

DISSERTATION

RAPID EARLY DESIGN SPACE EXPLORATION USING LEGACY DESIGN
DATA, TECHNOLOGY SCALING TREND AND IN-SITU MACRO MODELS

Submitted by

Charles V.K. Thangaraj

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall, 2009

UMI Number: 3401003

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3401003

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

COLORADO STATE UNIVERSITY

Nov 9, 2009

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY CHARLES V.K. THANGARAJ ENTITLED RAPID EARLY DESIGN SPACE EXPLORATION USING LEGACY DESIGN DATA, TECHNOLOGY SCALING TREND AND IN-SITU MACRO MODELS BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work



George Collins



Anthony Maciejewski



Phillip Chapman



Advisor: Tom Chen



Department Head: Anthony Maciejewski

ABSTRACT OF DISSERTATION

RAPID EARLY DESIGN SPACE EXPLORATION USING LEGACY DESIGN DATA, TECHNOLOGY SCALING TREND AND IN-SITU MACRO MODELS

CMOS technology scaling trend, i.e. the doubling of the operating frequency and the doubling of the number of transistors on a die every eighteen months, also known as Moore's Law has been a fundamental driver for the semiconductor industry for well over three decades. Scaling CMOS technologies into deep sub micron especially into sub 100 nm dimensions have caused a significant shift in business and design philosophy, and methodology. In addition to the semiconductor industry maturation there are seven key disruptive trends impacting the semiconductor industry. They are competitive landscape changes, technology convergence, greater global connectedness, increased design complexity, commoditization, consumerization, and the soaring research, development and engineering costs. These disruptions have made traditional business models increasingly ineffective and the benefits of Moore's Law insufficient for sustained competitiveness [1]. 'More-than-Moore' approach to heterogeneous system integration and holistic system optimization strategies in addition to the benefits of technology scaling are necessary for future success [2] [3].

Embedded computation systems and microprocessor designs have significantly benefitted from "cramming" more transistor on a single die. When memory is included on die (with large amounts of cache on die) the latency incurred in moving data and instructions to the computation units reduces sharply, increasing the overall

instruction execution rate and ultimately increasing performance. Increase in operating frequency being another aspect of technology scaling, improves the number of instruction executed per unit time. In a superscalar execution pipeline, increasing operating frequency increases overall instruction execution rate and throughput. The conventional design flow for computation engines (embedded computation systems and microprocessors) starts with architectural design followed by physical design. In nanometer CMOS technologies, successful physical implementation of a highly optimized architectural design is not guaranteed due to power consumption variations, signal integrity and processing challenges.

Consequently, design convergence in both power and performance have become increasing difficult with increasing levels of system integration, design complexity and technology scaling related uncertainties. As a result, traditional compartmentalized design methodologies are no longer sufficient as they lead to designs that are pessimistic, slow and/or power hungry. A holistic and systematic understanding of the various design tradeoffs and exploring the design solution space extensively early in the design phase improves design convergence. Some design challenges can be best addressed at the circuit level, others are most effectively addressed at the architecture or system level. With increasingly competitive business conditions dictating design cycle times and time-to-market window, a thorough design space exploration at an early stage of a design can put the design in an optimal subspace for better convergence and for avoiding costly redesigns later on in the design cycle. Rapid and effective design space exploration at all stages of a design process enables faster design convergence and meeting time-to-market stipulation. Design space exploration is important and particularly effective during the early stage of a design where design decisions can have a significant impact on design convergence. A holistic approach to system design is possible only when design tools and aids that incorporate high

level system models are easily available to perform tradeoff analysis and design space exploration.

This work proposes a system level design framework for early design space exploration with a focus on power and performance tradeoffs using analytical power and performance prediction models. The analytical prediction models are driven by legacy design data, technology scaling trend, low level physical design parameters and in-situ simulations. Experiments on ISCAS benchmark circuits validate the feasibility of the proposed approach and yielded power centric designs that improved power by 7% - 32% for a corresponding 0% - 9% performance impact; or performance centric designs with improved performance of 11.25% - 17% for a corresponding 2% - 3.85% power penalty. Evolutionary algorithm based Pareto analysis on an industrial 65 nm design uncovered design tradeoffs which are not obvious to designers and optimize both power and performance. The high performance design option of the industrial design improved the straight-ported design's performance by 29% with a 2.5% power penalty, whereas the low power design option reduced the straight-ported design's power consumption by 40% for a 9% performance penalty.

The design framework and methodology developed and demonstrated in this work form the foundational steps for early design space exploration utilizing technology scaling trends, process dependent parameters and in-situ simulations. Analytical prediction models are currently limited only to predicting power and performance. Prediction models for yield, chip area and system reliability are seen as valuable future additions to EIDAs capability. Modeling the impact of process variation and the ability to incorporate statistical inputs and outputs are seen as another incremental improvement to EIDAs value as a design tool. In addition to the above improvements, a macromodel based critical path delay calculation technique including clock and signal uncertainties, incorporating special libraries, RF and analog modules in

the system model and, improving the evolutionary algorithm used for design space exploration are salient direction for future research.

Charles V.K. Thangaraj
Department of Electrical and
Computer Engineering
Colorado State University
Fort Collins, Colorado 80523
Fall, 2009

ACKNOWLEDGEMENTS

"If I have seen further than others, it is by standing upon the shoulders of giants." - Sir Isaac Newton

"The horse is made ready for the day of battle, but victory rests with the LORD." - Proverbs 21:31

I am forever grateful to my graduate advisor, Dr. Tom Chen, for his constant support and guidance in the pursuit of my academic goals.

I thank Dr. George Collins, Dr. Anura Jayasumana, Dr. Anthony Maciejewski and Dr. Philip Chapman, members of my graduate committee, for their support throughout the program. I would also like to specifically thank all members of the VLSI System Architecture Lab, Colorado State University, for their invaluable help and support.

Special thanks to Bea & Carl Mohr and the Mohr family, Lynn & Jeffrey Smith, for their well wishes and many friends who were with me through thick and thin. I would also like to thank all my teachers from Stanes High School, Kumaraguru College of Technology and Colorado State University on whose shoulders I stand. Last but not the least, I thank my lovely parents and brother, for being my constant source of inspiration and for having faith in me.

Above all, I humbly give all glory to my Lord Almighty for His grace and blessings to undertake this arduous journey, without whom nothing is possible !

CONTENTS

1 Introduction	1
1.1 CMOS Technology Scaling	1
1.1.1 Technology Scaling and System Level Design	4
1.2 Semiconductor Industry Trends and Challenges	6
1.2.1 Business Trends	6
1.2.2 Manufacturing Technology Cost Challenges	7
1.2.3 Design and EDA Tool Challenges	9
1.3 Motivation and Objective	11
2 Background Information, Existing Methodologies and Approaches	13
2.1 Overview	13
2.2 System Level Power Performance Optimization	14
2.2.1 SimpleScalar Toolset	16
2.2.2 SimplePower Toolset	19
2.2.3 Wattch Toolset	22
2.2.4 AccuPower Toolset	25
2.2.5 PowerTimer Toolset	27
2.3 RT-Level Power Performance Optimization	31
2.4 Physical Level Power Performance Optimization	35
2.4.1 BACPAC Toolset	38
2.5 Shortcomings of Existing Tools and Methodologies	41
3 The Proposed Approach	45
3.1 Overview	45
3.2 Proposed High Level Modeling Methodology	45
3.3 Module Descriptor Vector Elements	49
3.3.1 Legacy Design Descriptors	50
3.3.2 Target Process Technology Descriptors	51
3.3.3 In-situ Simulations and Descriptors	52
3.4 Proposed Analytical Power and Performance Modeling Methodology	54
3.4.1 Dynamic Power	54
3.4.2 Leakage Power	56
3.4.3 Operating Frequency	58
3.4.4 Effect of V_{dd} Scaling	59

3.4.5	Procedure to Find Coefficients X and Y in Eqn 3.26	61
3.4.6	Procedure to Find ABBC, ABBPC, STC, STPC and DVTC Descriptors	62
3.5	Proposed Methodology for Rapid Early Design Space Exploration	66
3.5.1	Estimating Module Power and Performance	66
3.5.2	System Design Target Prediction	67
3.5.3	Design Space Exploration	67
3.5.4	Evolutionary Algorithms for Design Space Exploration	70
4	Proposed Design Space Exploration Methodology: Experimental Setup	71
4.1	Overview	71
4.2	Experiments in Technology Node Migration	72
4.2.1	Applying Module Granular Circuit Level Design Choice	74
4.3	Design Target Prediction Accuracy	75
4.3.1	Successive Design Porting From 180 nm to 65 nm Technologies	76
4.3.2	Design Space Exploration of the Test Circuit in 32 nm Technology	78
4.4	Evolutionary Algorithm Based Design Space Exploration	79
4.4.1	Design Migration	80
4.4.2	Pareto-Analysis Using Randomized Design Generation	80
4.4.3	Pareto-Analysis Using EA Based Design Generation	85
5	Experimental Results, Discussion of the Results and Future Work	89
5.1	Results of Experiments In Technology Node Migration	89
5.1.1	Technology Node Migration Experiment Observations	98
5.2	Results of Design Target Prediction Accuracy	100
5.2.1	Results of Successive Design Porting From 180 nm to 65 nm Technologies	100
5.2.2	Results of Design Space Exploration of the Test Circuit in 32 nm Technology	102
5.3	Results of Evolutionary Algorithm Based Design Space Exploration	103
5.3.1	Results of Pareto-Analysis Using Randomized Design Generation	103
5.3.2	Results of Pareto-Analysis Using EA Based Design Generation	109
5.3.3	Pareto-Front Decrowding Replacement Schemes	110
5.3.4	Figures of Merit: Stopping Criteria	112
5.3.5	Results of Pareto-Analysis Using the IRRR Scheme on ISCAS89 Circuit	117
5.4	Discussion of the Results	119
5.4.1	Impact of ABB Design Choice	119
5.4.2	Prediction Model Complexity: Impact of Model Parameters	121
5.4.3	Pareto-front Quality: Impact of Evolutionary Algorithm	124
5.5	Conclusion	125
5.6	Future Work	127
6	Acknowledgement	129

A	Common Design Techniques Incorporated into the EIDA Tool	130
B	Code for System Design Target Prediction	136
C	Code for EA Based Design Space Exploration	142
D	GUI Implementation of EIDA	148
D.1	Screen Captures of the GUI	148
D.2	TCL/Tk Code for the GUI	155
	Bibliography	184

LIST OF FIGURES

1.1	ITRS clock frequency trend up to the year 2020	2
1.2	Clock rate for high performance microprocessors from Dec '92 to May '02	3
1.3	Transistor L, V_{dd} and V_t through 2020	4
1.4	ITRS Roadmap for gate and wire delay through process nodes	5
2.1	Design flexibility, solution time and tool complexity	14
2.2	System level optimization flows	15
2.3	An overview of SimpleScalar toolset	17
2.4	SimpleScalar pipeline model	18
2.5	SimpleScalar internal organization	18
2.6	SimplePower internal organization	20
2.7	SimplePower switching capacitance table for bit-dependent microarchitec- tural blocks	22
2.8	Wattch internal organization	23
2.9	AccuPower internal organization	26
2.10	PowerTimer internal organization	27
2.11	PowerTimer pipeline model	28
2.12	PowerTimer microarchitectural block power model	30
2.13	Typical RTL power estimation flow	32
2.14	Interconnect parasitics model	39
2.15	Steps to estimated delay in BACPAC	40

2.16	Steps to estimated dynamic power in BACPAC	41
3.1	Critical paths and modules	47
3.2	Fanout of four configuration	48
3.3	A partitioned system shown with module #3 abstracted with F04 inverters and its corresponding descriptor vector	49
3.4	Experiment to obtain STC and STPC factors	62
3.5	Experiment to obtain ABBC and ABBPC factors	64
3.6	Experiment to obtain DVTC factor	66
3.7	Application of a design choice to a module	67
3.8	Rapid early design space exploration flow chart	69
4.1	Benchmark circuit partitioning with critical path shown	73
4.2	Test system to determine prediction accuracy	76
4.3	Procedure to compare EIDA and SPICE	78
4.4	Block diagram of an modern microprocessor. B/I/M/FP: branch/ integer/ memory/ floating point units; ALAT: advanced load address table; TLB: translation look-aside buffer	81
4.5	a) Simple randomizer algorithm b) Complete randomizer algorithm . . .	84
4.6	(a) Chromosome for evolutionary algorithm based pareto analysis. A com- plete list of all design choices is listed in Table 4.6 (b) A valid chromo- some (12,9,3,0,.....,9,7,10,11)	86
5.1	C5315 design choices	91
5.2	C6288 design choices	92
5.3	C7552 design choices	93
5.4	S38584 design choices	94
5.5	S132007 design choices	95

5.6	S38417 design choices	95
5.7	S15850 design choices	97
5.8	S9234 design choices	98
5.9	Observed technology scaling trends for power and performance	100
5.10	Observed prediction error with respect to SPICE	101
5.11	Pareto-front analysis results	104
5.12	Design #9 details	106
5.13	Design #11 details	106
5.14	Design 14 details	107
5.15	Design 13 details	108
5.16	EA pareto-front progression - baseline with no replacement	109
5.17	EA pareto-front progression - with EER	111
5.18	EA pareto-front progression - with IRRR	111
5.19	Figure of merit for pareto-front solution quality	113
5.20	Figure of Merit for pareto-front solution spread	113
5.21	Pareto-front analysis results	115
5.22	Details of system design A from Fig 5.21	116
5.23	Details of system design B from Fig 5.21	117
5.24	Pareto-front with IRRR at 50K iteration for ISCAS89 s38584 and s38417 circuits	118
5.25	Figure of merit for ISCAS89 s38584 and s38417 circuits Pareto-front with IRRR solution quality	118
5.26	Figure of Merit for ISCAS89 s38584 and s38417 circuits Pareto-front with IRRR solution spread	119
5.27	Impact of ABB design choice	120
A.1	An illustration for using sleep transistors for power gating.	131

A.2	An illustration for multiple V_{dd} zones in a design.	132
A.3	An illustration for clock gating technique.	133
A.4	Decoupling capacitance allocation in a standard cell design	134
D.1	Invoking the EIDA tool's GUI from the command prompt	148
D.2	Entering the number of modules in the design	149
D.3	Confirming the number of modules and choosing interactive run type . .	149
D.4	Entering the process dependent descriptors	150
D.5	Entering the command file name to save the entered data	151
D.6	Entering VDD scaling descriptors	151
D.7	Entering modules descriptors for each module in the system	152
D.8	Entering design choices for each module in the system	153
D.9	System power and performance for the chosen module design choices . .	153
D.10	EIDA tool used in batch mode from the command prompt. A command file corresponding to the modules in the system and their respective design choices has to be created prior to invoking EIDA from the command prompt.	154

LIST OF TABLES

3.1	Descriptors from legacy design	50
3.2	Target process technology descriptors	52
3.3	Circuit-level design choices requiring In-situ simulations	53
3.4	Descriptor vector elements from in-situ simulations	54
3.5	Algorithm ST_size_evaluate	63
3.6	Algorithm ABB_evaluate	65
4.1	Benchmark circuit details	72
4.2	Assumed descriptor and coefficient values for 180 <i>nm</i> TSMC to 130 <i>nm</i> PTM technology porting	73
4.3	Module-granular circuit-level design choices	74
4.4	Assumed descriptor and coefficient values for 180 <i>nm</i> to 130 <i>nm</i> PTM & 130 <i>nm</i> to 90 <i>nm</i> PTM & 90 <i>nm</i> to 65 <i>nm</i> PTM & 65 <i>nm</i> to 32 <i>nm</i> PTM	77
4.5	Assumed descriptor and coefficient values for 65 <i>nm</i> to 32 <i>nm</i> PTM . . .	82
4.6	Valid available additional design choices	83
4.7	Seed recipes for pareto-front analysis	83
4.8	Summary of evolutionary algorithm based pareto analysis	87
5.1	Technology mode migration results	90
5.2	Circuit C5315 migration results	91
5.3	Circuit C6288 migration results	92

5.4	Circuit C7552 migration results	93
5.5	Circuit S38584 migration results	94
5.6	Circuit S132007 migration results	96
5.7	Circuit S38417 migration results	96
5.8	Circuits S15850 & S9234 migration results	97
5.9	EIDA and SPICE comparison	102
5.10	Modular design choices for designs #14, #13, #8 and #10	105
5.11	Power and performance sensitivity of selected model parameters	122
5.12	Power and performance sensitivity of selected model parameters with re- duced de-coupling capacitance	123

Chapter 1

Introduction

1.1 CMOS Technology Scaling

CMOS being the dominant technology used in VLSI systems has scaled well into deep sub-micron (DSM) sub 100 nm feature sizes. CMOS technology scaling is expected to continue the current trend and scale into sub 10 nm feature sizes in the coming decades. Current state-of-the-art production CMOS process has a minimum feature size of 45 nm and 32 nm technologies are currently being piloted for production. The semiconductor industry road map predicts that CMOS technologies with a minimum feature size of 6 nm will be developed by the year 2020 [4]. The benefits of technology scaling include reduced manufacturing cost per transistor, reduction in energy per logic function implemented on chip, and increase in the number of logic functions (i.e. transistors) that can be integrated on a single die. As feature sizes reduce, the evaluation time required for a logic function implemented in these technologies reduce due to faster transistor switching times. Reduction in evaluation time translates into an increase in the number of evaluations completed per unit time, in other words, an increase in performance. Improved performance and the reduced cost of integrating logic functions on chip are very favorable for business. The economic

benefits of scaling have been very tangible and companies want this trend to continue well into the future.

Moore's Law [5] states that the number of transistors in a chip and the chip operating frequency double every eighteen months. Global semiconductor industry competitive landscape dictate that semiconductor companies abide by this law and scale their manufacturing process to stay relevant in the globalized market place. In addition to device size the operating frequency also doubles every eighteen months. As a result, designs in scaled technologies become faster. Due to the higher integration potential of the scaled technologies, designs have become bigger, more powerful and more capable in terms of functionality. For example, technology scaling has enabled microprocessor companies to design microprocessors with higher operating frequency compared to their preceding generation designs. A recent high performance microprocessor designed in a 65 nm SOI process operates at a core clock frequency of 4.7 GHz [6].

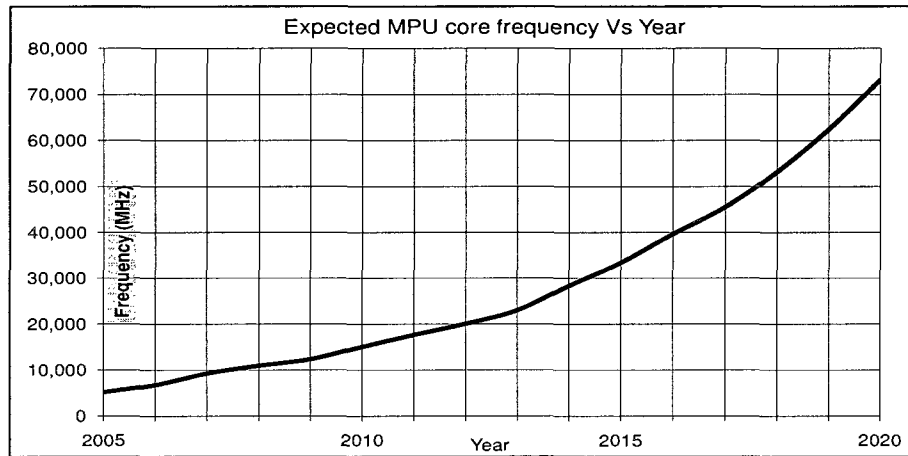


Figure 1.1: ITRS clock frequency trend up to the year 2020

The ITRS (international technology road-map for semiconductors, predicts the core operating frequency for high performance microprocessor as shown in Fig 1.1. At present CMOS technologies with minimum feature size in the range of few tens

of nanometers are becoming relatively commonplace allowing higher operating frequencies [7, 8]. As shown in Fig 1.1 the projected microprocessor core operating frequency for the next decade is upwards to 73 GHz. Aggressive scaling enables designer to design chip that meet the ever increasing demands of increased functionality by scaling designs to newer and advanced CMOS processes. In addition to increased functionality the newer technologies enable designs with higher operating frequencies. Fig 1.2 [9] shows the microprocessor core operating frequency trends in the past, i.e. from the December 1995 to May 2002.

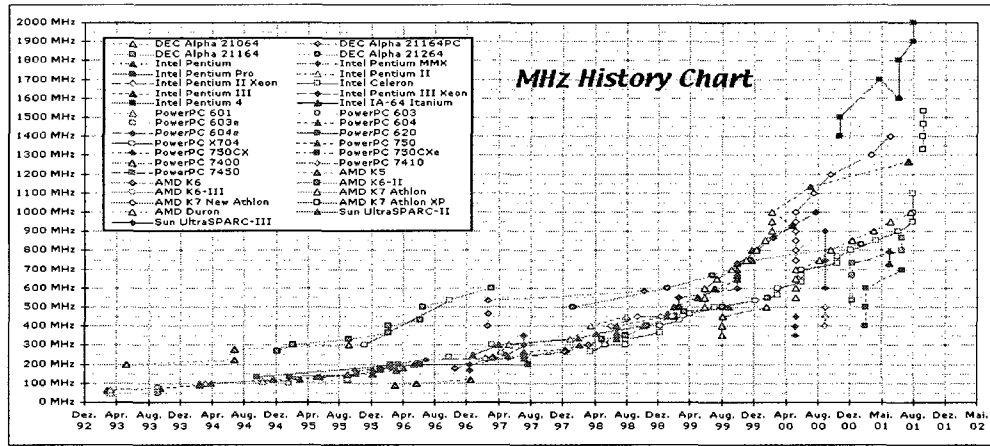


Figure 1.2: Clock rate for high performance microprocessors from Dec '92 to May '02

Technology scaling along with its benefits has numerous challenges. A very important challenge is the closing gap between the supply voltage V_{dd} and threshold voltage V_t as shown in Fig 1.3. This means that the transistor cannot be turned off effectively and thus lead to a dramatic increase in leakage currents. The increase in leakage is further compounded by the decreasing transistor lengths which lead to a decrease in V_t . Leakage power consumption in state-of-the-art high performance microprocessor designs can be as much as 30 to 40% of the total power consumption. Due to the higher levels of integration the total switching capacitance load (normalized to technology) increase. Higher operating frequencies leading to higher

switching rates result in an increase in thermal induced instabilities and switching related signal integrity uncertainties. Another direct impact of scaling is the explosion in the number of interconnects, particularly long global interconnects as the die sizes increase. Increasing interconnect densities will lead to a reduction in signal integrity and preventive measures to improve signal integrity, which are costly are often necessary. With increasing operating frequency and distances, global signals have to travel faster and longer than they did earlier resulting in an increased need for repeaters in global interconnects. Consequently resulting in tighter delay tolerance, design tolerance and an increase in repeater power consumption [10–12].

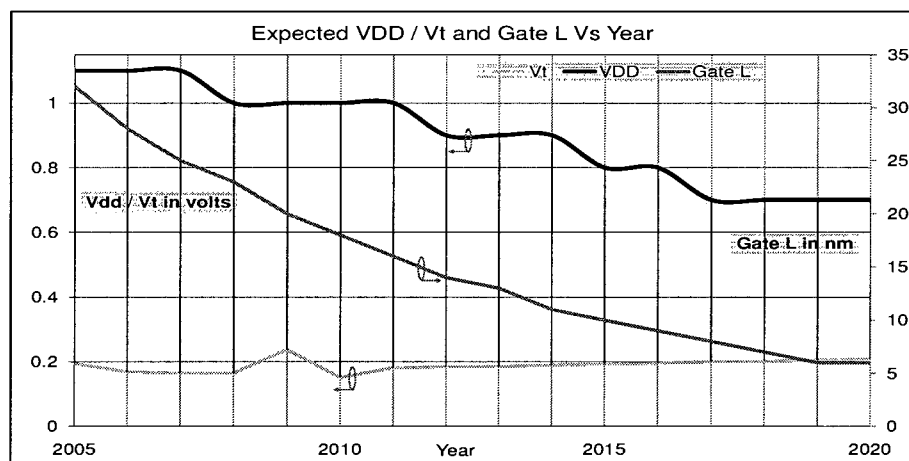


Figure 1.3: Transistor L , V_{dd} and V_t through 2020

1.1.1 Technology Scaling and System Level Design

The effects of technology scaling impact the design process of a VLSI system in a number of different ways. Notably technology scaling exasperates the effect of manufacturing process variation. When the critical dimension or the minimum feature sizes (CD) reduces, the tolerance in manufacturing variation can approach the CD or be the same order of magnitude as the CD. Therefore manufacturing variations can significantly alter gate delay, interconnect delay, threshold voltage V_t , gate oxide

thickness t_{ox} , gate area, leakage current variations and SRAM stability among many others, leading to increased design uncertainties. In addition to manufacturing variability lithographical limitations impact device and interconnect characteristic and composition.

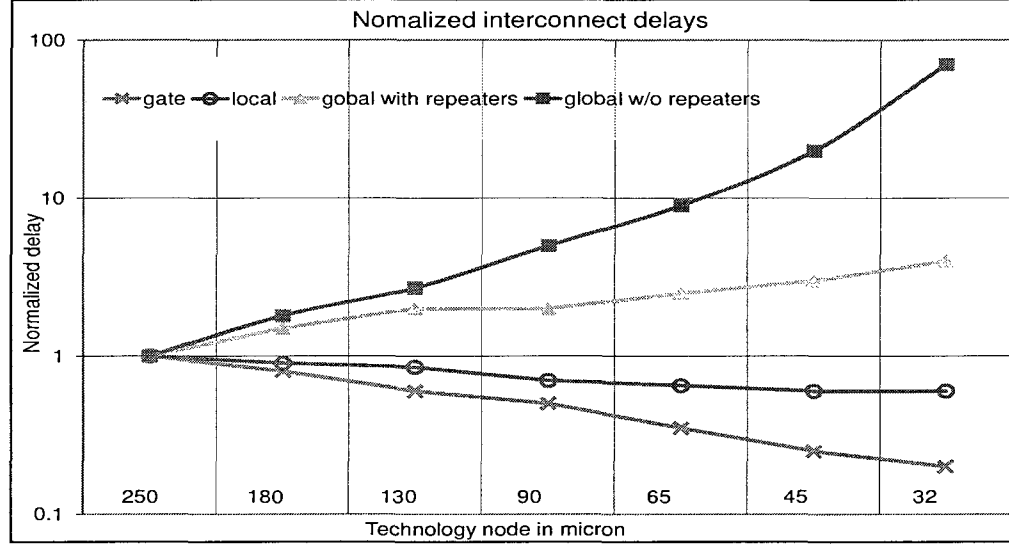


Figure 1.4: ITRS Roadmap for gate and wire delay through process nodes

Fig 1.4 shows the growth in wire delay through various process technologies. Global interconnects have become performance limiters at the system level due to exponentially increasing wire delays. Interconnect composition (width, pitch and thickness) variations causes uncertainties in interconnect delays [13]. To include interconnect uncertainties designers focus on the worst case which results in pessimistic over designing of interconnect drivers and increased power consumption. It is very important to accurately estimate interconnect delay as the critical path delay has a significant interconnect delay component which ultimately determines system performance. The same is true for gate delay as well. The variations in gate L or W, V_t and t_{ox} manifests as uncertainties in switching speed, drain or on current (I_{ds}) as well as leakage current. From a power consumption point of view it is very important to be able to estimate the leakage currents to ensure that the design falls within the

power envelope specification [14]. SRAM stability, negative and positive bias temperature instability [15] and SER (soft error rate) [16, 17] vulnerability (all of which are strongly influenced by scaling) will determine the minimum power supply voltage V_{dd} [18], the system reliability [19] and the implementation of system level architectural error protection schemes [20]. Clearly technology scaling greatly influences key design parameters which in turn impact system level design parameters and/or system architecture.

1.2 Semiconductor Industry Trends and Challenges

1.2.1 Business Trends

The semiconductor industry has grown approximately 16% in the past decade, with 9 to 10X revenue growth to around 200 billion dollars annually. The projected growth for this industry in the next ten years is 6% annually with a projected revenue of approximately 700 billion dollars annually by 2017 [21]. Microprocessors are the most important of all semiconductor products and represents the high end of the semiconductor industry's product line. The fierce competition in this market segment compels companies to maintain an edge over their rivals to retain market share. In the past decade, quick and successive introduction of new microprocessor architectures or process technology improvements have proved to be a successful strategy to maintain market share and profitability. Current leading microprocessor design companies try to introduce new process technology or architectural advancements once every two years.

The two year cycle in the microprocessor industry is increasingly difficult to maintain, the reasons are attributed to the exponentially increasing cost of building a new manufacturing or fabrication facility (fab). The cost of a 90 nm fab was US\$2

billion whereas the cost for building a 45 nm, 300 mm wafer fab which is two generations ahead of the 90 nm fab is well over US\$5 billion. Moreover the cost incurred in developing a new 45 nm process technology was 30% higher than the 90 nm process. The cost of research and development of newer (i.e. 32 nm, 22 nm, 16 nm and so on) 450 mm CMOS technology is expected to grow exponentially with each progressive technology generation [21]. Given the high capital cost of developing newer fabs and technologies, it is important for semiconductor companies to have very short design cycles, smaller time-to-market, fewer re-spins (prototyping for testing and validation before marketing), high yield and minimal operating cost to maintain market segment share and profit margins. Design leveraging is a strategy used by most semiconductor companies to achieve better return on investment. Design leveraging is the process of porting an existing high end design to a newer technology to release the legacy design as a newer incremental product, swiftly. Design leveraging also includes reusing parts of the existing high end design to create a product with a subset of features on the original design targeted to the lower end of the market segment or a completely different market segment. Design leveraging strategy attempts to reduce the time-to-market for keeping the company's product lines fresh and its business attractive to investors who continually seek value and good business fundamentals.

1.2.2 Manufacturing Technology Cost Challenges

The lithography process, especially in the sub 100 nm CMOS technology domain had increased in complexity many folds in order to maintain high patterning quality [22] [23]. Sub-wave-length lithography (where the wave length of the light source used in the manufacturing process is larger than the intended pattern size on the wafer) and decreasing feature size tend to decrease patterning quality. The wavelength of the illumination source used in lithography has changed very little over the

years and reticle enhancement techniques (RET) [24] have become commonplace to improve lithographical patterning quality. Phase-shift masks, optical proximity corrections (OPC) [25] and off-axis illumination (OAI) [26] are three major resolution enhancement technologies that have enabled optical lithography extend into the nano meter era. These techniques considerably increase the complexity of lithographical masks. OAI in particular limits the pitches and sizes of the shapes that can be patterned effecting transistor designs. Among new proposed technologies to improve lithography, extreme-UV (EUV) lithography is promising however the cost of this technology has remained prohibitive till date. An other cheaper alternative is immersion lithography which is shown to be practical, however there are many mechanical problems with this technology that needs further development [27, 28].

Sub-wave-length lithography and the additional complications introduced by phase-shift masks, OPC, OAI and other sub-resolution assist features (SRAF) increase the variability in semiconductor manufacturing. Process variations (PV) in doping densities, gate oxide thickness, field oxide thickness and gate dimension alters the characteristic of transistors by introducing uncertainties in transistor on-off currents, V_t , gate leakage and junction leakage currents [29]. Smaller the feature size more pronounced are the effect of process variation motivating the need for expensive RETs. Designers (as opposed to the technology developers) have tackled the undesirable outcome of PV by employing design for manufacturing (DFM) techniques during the design process. DFM techniques usually involve additional design rules to help improve pattern quality and reduce the impact of PV. The limitation of this technique is that as CMOS technology scales, there will be an explosion in DFM rules further restricting and complicating the design process.

Given the high cost of manufacturing, challenges remain in reducing manufacturing cost. However until a suitable and relatively inexpensive solution is identified,

semiconductor companies have to contend with existing technologies and rely upon efficient design methodologies to keep cost to a minimum and thus remain profitable.

1.2.3 Design and EDA Tool Challenges

As explained in Section 1.2.1, design leveraging and re-optimizing existing designs have become common design approaches to reduce design cost. Since leveraged or re-optimized designs have very specific goal of either boosting performance or reducing power or optimizing both; huge cost savings are possible when designs are leveraged or re-optimized. While some of the design goals may be obtained through porting a design to a newer technology, often additional tweaking of the ported design in the newer process is necessary. Lithography challenges in the newer technology generations result in physical device topology restrictions making design convergence more difficult. In addition to these restrictions, since many features do not scale well, co-optimization and design of the devices, the circuits and the layout are absolutely essential to successfully port designs to newer technologies.

With companies opting for quick and successive introduction of newer products in the market, the time available for product design (i.e. design cycle time) reduces. This implies that design teams have to increase in size (which may not be possible or desirable) or need to be more efficient and turn out high quality designs that reduce the need for expensive re-designs and debugging. A thorough design space exploration at an early stage of a design can put the design in an optimal subspace for better convergence whereby avoiding expensive redesigns later on and improves the design team's efficiency.

Conventional design flow starts with the system architectural design followed by physical design [30]. System architects base their design decisions on expert knowledge and assumptions regarding many aspect of the physical design process.

In nano-meter CMOS, however, physical implementation and design convergence of a highly optimized architectural design is not guaranteed due to increasing leakage power, process variation effects, thermal density issues, signal integrity degradation and lithographical challenges [4]. The physical design process tries to optimize and often tradeoff design objectives such as performance, power consumption, reliability and yield despite the fact that the design objective are deeply intertwined with each other [31, 32]. Under stiffer time-to-market stipulation dictated by business needs, making optimal and correct design choices at each design phase becomes imperative. This is especially true at the architectural design phase where design decision have a greater impact on design convergence. Since the physical design convergence of a highly optimized architectural design cannot be guaranteed, a quick feasibility analysis of all the architectures considered will help in choosing an architectural solution that is implementable and has a higher chance of design convergence.

Existing EDA tools typically address specific design aspect such as timing or power in an accurate and detailed manner. They do not address the overall system level design trade-off, implementation feasibility analysis, and fast turn-around what-if analysis often required for assisting designers to meet design quality and time-to-market requirement [33]. Even though a collection of existing tools may be utilized to perform the above tasks, inter-operability overhead and their nature of lower level detailed analysis makes such a concoction too slow for quick feasibility analysis required for effective design space exploration. Therefore newer design tools and design aids that incorporate system level models and are fast and sufficiently accurate need to be developed. These tools can then be used during the early design phase or architectural design phase for design space exploration and tradeoff analysis and can help in choosing optimal system design for implementation. Doing this would improve design convergence and help meeting time-to-market stipulation.

1.3 Motivation and Objective

Business, technology and tool challenges faced by the semiconductor industry have resulted in tighter design cycle time, increased difficulty in design convergence and a need for newer design tools and aids to improve first pass design successes. A thorough design space exploration at an early stage of a design puts the design in an optimal design subspace enabling better design convergence. Existing conventional design flows lack the ability for performing early rapid design space exploration capable of reducing the need and extent of expensive last-minute re-designs.

Most modern high performance designs tend to improve and build on comparable existing designs. Sophisticated design exploration methodology and tools are especially suited for such leveraged designs since the parameters in the system level models have higher confidence level than those for ground-up designs. The work described in this dissertation is motivated by the lack of fast and effective design space exploration tools and the shortcomings of the existing approaches. The goal of the proposed method is two folds. First, to develop a high level system modeling methodology which includes low-level physical design parameters to provide more realistic constraints for design space exploration. Second, to develop a design framework for early design space exploration consisting of the high level system models and analytical models that can be used to estimate design targets such as power consumption and (maximum operating frequency) performance.

Designing large systems such as high-end multi-core microprocessors and complex system-on-chips (SOCs) are often performed by multiple design teams in parallel with each team focusing on a portion (sub-system) of the larger system. In such an environment, compartmentalized design optimizations done by individual design teams do not guarantee global design optimality. To ensue and achieve global design optimality, a holistic approach to design tradeoff and optimization is needed [2,3]. This

work focuses on performance and power optimization and tradeoff analysis during the early design space exploration phase when complete bottom-up implementation data is not yet available. The proposed analytical models include prediction models for leakage power consumption, dynamic power consumption and maximum operating frequency. Utilizing legacy design data, technology scaling trend data and allowing in-situ macro-model generation and simulation, the proposed framework is positioned for more realistic estimates of the impact of circuit level design choices for a given design. The proposed evolutionary algorithm based design space exploration methodology and the modeling of a system as a collection of modules (or sub-systems or sub-design) where the modules are independently characterized allows for modeling and analyzing large designs with large design spaces without significantly disproportionate increase in system modeling efforts.

Chapter 2

Background Information, Existing Methodologies and Approaches

2.1 Overview

Design convergence in both power and performance have become increasing difficult with increasing levels of system integration, design complexity and technology scaling related uncertainties. Consequently, early design phase design validation through power performance tradeoff analysis and design space exploration have thus become an integral part of the standard design flow. This can be performed at various stages in the design process such as;

- Power performance optimization at the system level
- Structural or logic optimization at the RTL level
- Library optimization at the physical level
- Process technology (SPICE model) optimization at the foundry level

The physical design convergence of a highly optimized architectural design in nanometer CMOS is not guaranteed. The likelihood of physical design convergence

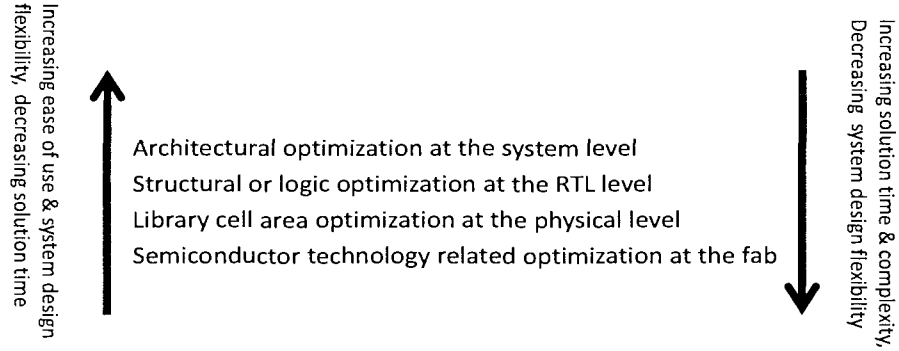


Figure 2.1: Design flexibility, solution time and tool complexity

of the highly optimized architectural design can be increased by performing design space exploration and design optimization at the system level based on models representing low level implementation during the early design phase. The scope and opportunities for power and performance tradeoff are maximum when the optimization is performed at the highest level of design abstraction, as illustrated in Fig 2.1. Moreover, discovering a system level power consumption excursion from the intended design target when the physical implementation is complete is far too late in the design cycle for significant remedial redesign effort without delaying time to market. Design teams recognize the potential harm in encountering this situation and try to avoid it by employing many techniques and tools. The following section reviews existing methodologies and tools for design space exploration and power performance tradeoff at various levels of abstraction.

2.2 System Level Power Performance Optimization

In a conventional design flow the system architectural design precedes physical implementation. System architects often use high level architectural models to explore the architectural design space to perform architectural design optimization.

Fig 2.2 shows two procedures where a high level architectural model for a computational engine is utilized for architectural design optimization. These architectural models are implemented in traditional programming languages or hardware description languages. They serve to emulate the execution of standard benchmark programs on the computational engines they model. By compiling and executing well known benchmark programs, the architectural correctness and the efficiency of an architectural design can be determined very early on in the design phase.

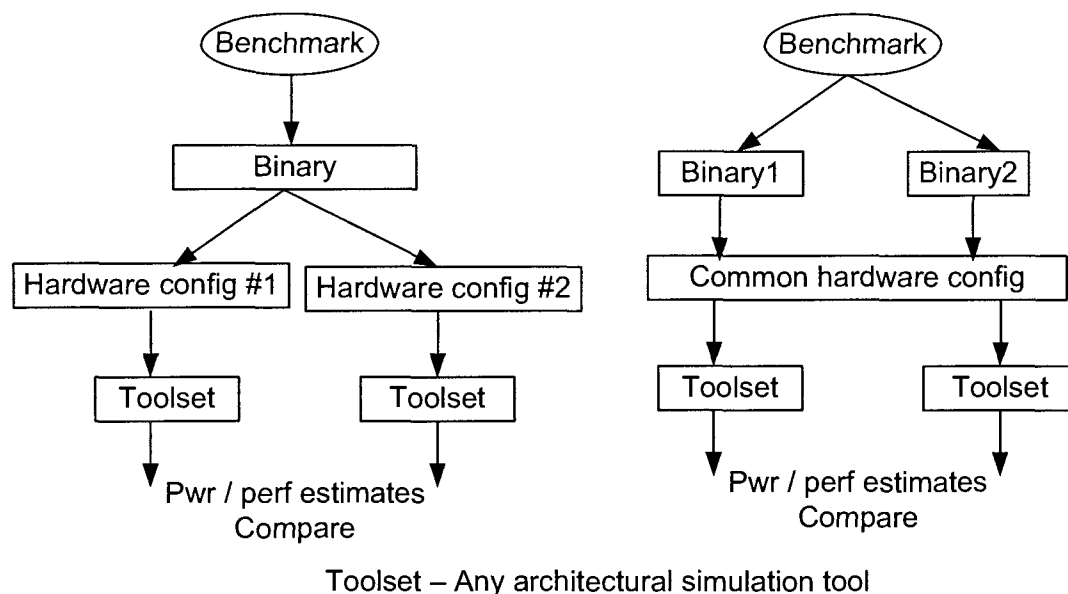


Figure 2.2: System level optimization flows

High level architectural models have three critical and conflicting characteristics, they are model fidelity, model flexibility and model detail. Model fidelity refers to the model's ability to capture the computational engine's features such that the emulated execution of the various types workloads are close to the actual execution of the various workloads on the computational engine. Flexibility indicates the model's ability to model a wide variety of architectural designs and the relative ease of making incremental changes to the architecture and emulating the execution of benchmark

programs. Model detail pertains to the level of architectural detail incorporated in the model. For example, a execution unit may be modeled as a black box or can be modeled at a lower level of abstraction as a unit consisting of sub-units or blocks such as program counter, decode logic, ALU, multiplier unit etc.

Practically however, maximizing model fidelity, model flexibility and model detail in tandem has been proven to be difficult. Most existing system level architectural models developed to study and optimize computation engine architecture, maximize two of the three critical model characteristic often at the expense of the third. The earliest notable toolset for architectural optimization is the SimpleScalar toolset [34]. SimpleScalar toolset is a fast, flexible and accurate simulator of microprocessors based on the MIPS architecture. SimpleScalar implements a parameterized modular microprocessor model based on the MIPS-4 instruction set architecture (ISA). In the following sections existing solutions for system level design optimization are discussed.

2.2.1 SimpleScalar Toolset

The SimpleScalar toolset provides an infrastructure for architectural design, simulation and optimization. The parameterized microprocessor model used in SimpleScalar has good fidelity and flexibility and, capable of modeling a variety of architectures ranging from a simple un-pipelined microprocessor to a complex multi cycle multiple issue out-of-order dynamic scheduling microprocessor with multiple levels of cache. It is also easy to extend the built in microprocessor model to include additional microarchitectural features and/or to modify existing features. This allows for sufficiently detailed architectural modeling. SimpleScalar emulates the computation process in the microarchitected processor by executing the benchmark program's instructions using an instruction interpreter. Software workloads designed for popular processor architectures such as Alpha, Power PC, x86, and ARM can also be executed by using appropriate instruction interpreters for the various workloads.

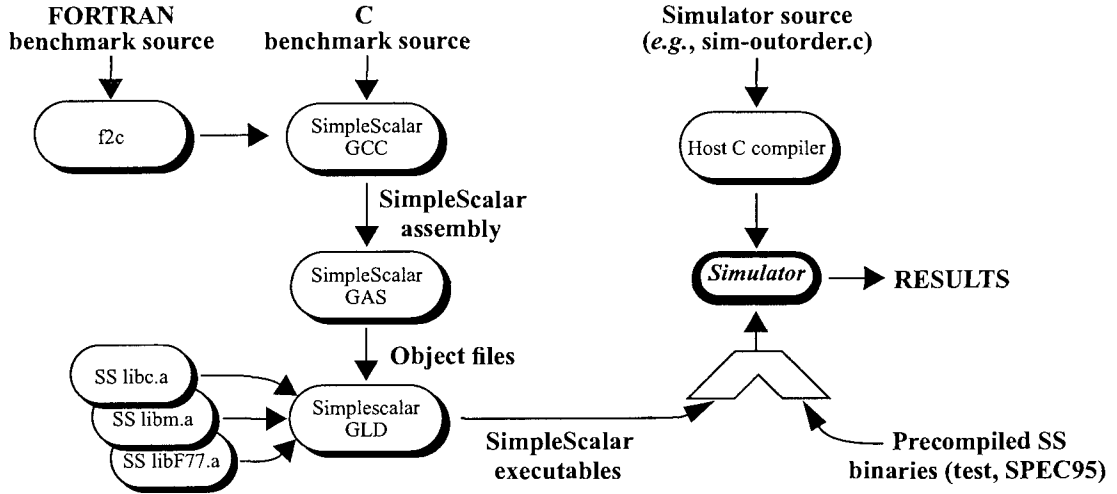


Figure 2.3: An overview of SimpleScalar toolset

SimpleScalar compiles binaries of the workloads (as in Fig 2.3) for the modeled architecture and emulates program execution. Thus verifying instruction execution, determining cache miss rates, estimating or summarizing execution profiles and execution time in number of processor clock cycles. The execution pipeline model used in SimpleScalar is shown in Fig 2.4. Figure 2.5 shows the SimpleScalar’s internal software organization. Software workloads run on the modeled microarchitectural model using a technique called execution-driven simulation. An instruction-set emulator and an I/O emulator are utilized to interpret the workloads instructions to execute using the host platform. The instruction-set emulator interprets each compiled instruction and directs the microarchitectural models activity through callback interfaces built into the instruction interpreter.

The interpreter comprehends the nature and functioning of all the instructions in the ISA and directs the architectural model to update appropriate registers and memory state. A preprocessor uses these machine definitions to synthesize the interpreter, the dependence analyzer and the microcode generator that SimpleScalar models need to emulate program execution. The I/O emulation module provides interface to the

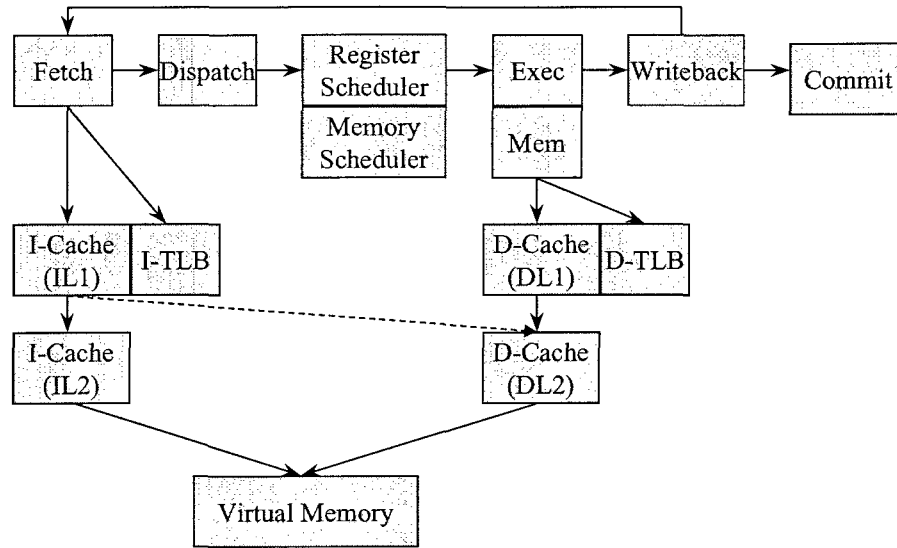


Figure 2.4: SimpleScalar pipeline model

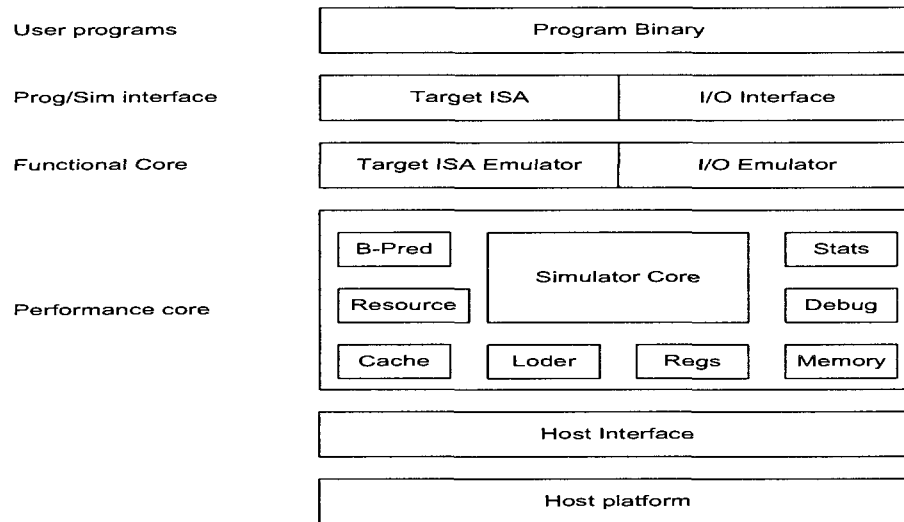


Figure 2.5: SimpleScalar internal organization

external host's input and output sources. The I/O interface translates a system call in the emulated program execution into an equivalent host operating system's system call. After executing the system call the I/O interface handles the returning of the results back to the emulated program. In this manner any program written for a

particular instruction set architecture (ISA) can be interpreted and executed on any host platform. This is called execution driven microarchitectural simulation.

Execution driven approach provides access to all data produced and consumed during program execution. These values are crucial to the understanding data and control flow, prediction optimizations, memory compression and dynamic power analysis. In dynamic power analysis, the simulation must monitor the data values sent to all microarchitectural components such as the arithmetic logic units and the caches to gauge switching activity which consumes power. Execution driven microarchitectural simulation also permits greater accuracy in the modeling of speculative branch prediction or load address speculation. Speculative execution causes miss-predictions, when a miss-prediction is detected later on during program execution, the pipeline is flushed and restarted with the instruction preceding the miss-prediction. Speculative instruction executions cause resource conflicts with nonspeculative instructions potentially slowing the program, this can be studied only with execution driven microarchitectural simulation. Trace-driven techniques cannot model speculative code execution because instruction traces record only correct program execution. Thus, execution driven simulation faithfully reproduces the speculative computation and correctly models its impact on program performance. Execution driven microarchitectural simulation, as in SimpleScalar, is therefore considered better than the leading alternative i.e. trace based microarchitectural simulation. The use of SimpleScalar in evaluating and optimizing different system architectures for throughput and chip area is illustrated in [35].

2.2.2 SimplePower Toolset

A major limitation of the SimpleScalar toolset is the lack of a power estimation tool in the toolset. It may be recalled that SimpleScalar does provide activity factors

for various microarchitectural blocks under a variety of workloads. However just activity factors alone are not sufficient to estimate power consumption. SimplePower [36] an extension to SimpleScalar, is an input transition-sensitive execution-based cycle-accurate power estimation tool. Essentially SimplePower is a set of C-based post processing procedures to perform simulator independent process technology dependent power estimation. SimplePower assumes a five stage pipelined data path, consisting of the fetch stage (IF), the instruction decode stage (ID), the execution stage (EXE), the memory access stage (MEM), and the write-back stage (WB). Figure 2.6 shows the organization of SimplePower toolset [37].

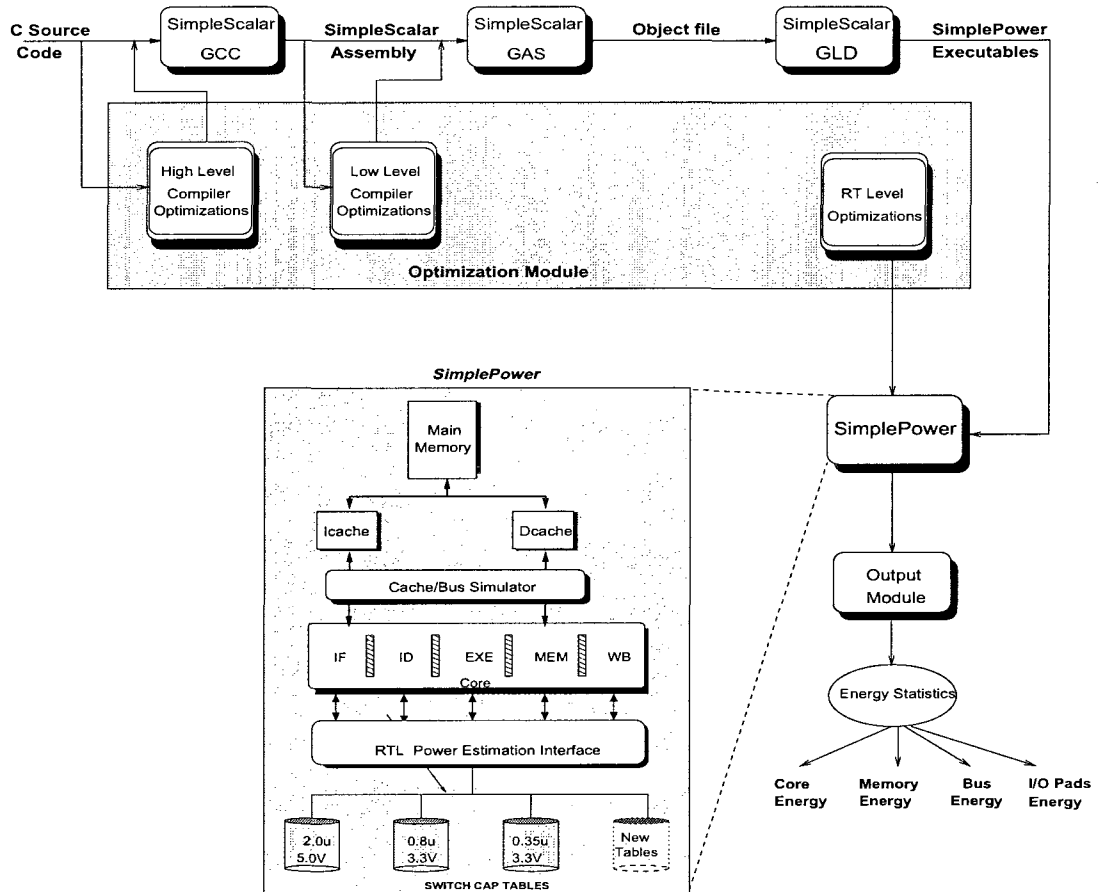


Figure 2.6: SimplePower internal organization

While executing a compiled benchmark workload, SimpleScalar, at each clock cycle, simulates the execution of all active instructions, while SimplePower monitors the activity and calls corresponding to the power estimation interfaces for all activated microarchitectural blocks. SimplePower maintains a pre-calibrated technology dependent switching capacitance table for each microarchitectural block in the design such as the adders, the ALUs, the multipliers, the shifter, the controllers, the register file, the pipeline registers and the multiplexors. The built-in bus simulator snoops and records the total number of accesses and the number of transitions on the instruction cache address bus, the instruction cache data bus, the data cache address bus, and the data cache data bus. The recorded number of accesses are combined with analytical interconnect power models to compute the effective switch capacitance of the on-chip buses. The cache simulator simulates the cache access activity and records them. SimplePower then estimates the power consumption by using a lookup table containing the switching capacitance for each input transition for every microarchitectural block activated.

2.2.2.1 Switching Capacitance Table Construction

SimplePower's power estimation accuracy depends on the accuracy of the switching capacitance tables. The construction of these tables is based on the structure of the microarchitectural block. All microarchitectural blocks fall into one of the following types: bit-independent microarchitectural block or bit-dependent microarchitectural block. In a bit-independent microarchitectural block, the bit slices operate independent of each other. Thus only a small bit slice switching capacitance table is needed. The total energy consumed by the microarchitectural block can be calculated by summing the energy consumed by each bit slice during transition. Bit-independent functional units include the pipeline registers, the logic unit in the ALUs, latches and

buses. In a bit-dependent microarchitectural block, the bit slices are not independent of each other. In such cases energy characterization is based on input vector differences i.e. the total bit flips between two adjacent inputs; as shown in Fig 2.7 [36].

Index		Switch Capacitance (pF)
previous input vector	current input vector	
$0_1 \dots 0_n$	$0_1 \dots 0_n$	cap_0
$0_1 \dots 0_n$	$0_1 \dots 1_n$	cap_1
$0_1 \dots 0_n$	$0_1 \dots 10_n$	cap_2
$0_1 \dots 0_n$	$0_1 \dots 11_n$	cap_3
...
$1_1 \dots 1_n$	$1_1 \dots 10_n$	$cap_{2^n - 2}$
$1_1 \dots 1_n$	$1_1 \dots 11_n$	$cap_{2^n - 1}$

Figure 2.7: SimplePower switching capacitance table for bit-dependent microarchitectural blocks

For large microarchitectural blocks with a large input vector size, observing that the size of this table grows exponentially, two remedial measures were included. They are; analytical transition independent power modeling and partitioning of large microarchitectural blocks into smaller ones. Analytical modeling involves approximating the switching capacitance based on other observable parameters depending on the microarchitectural block. Partitioning of the microarchitectural blocks into smaller blocks does not require any additional changes to the toolset, only the number of microarchitectural blocks will increase.

2.2.3 Wattch Toolset

The Wattch toolset [38] similar to SimplePower is an extension of SimpleScalar and can be used to analyze and optimize microprocessor architectures and power dissipation. Wattch features a parameterized power model for common microarchitectural blocks found in modern superscalar microprocessors. These power models are

integrated into the SimpleScalar architectural simulator toolset to form the Wattch toolset.

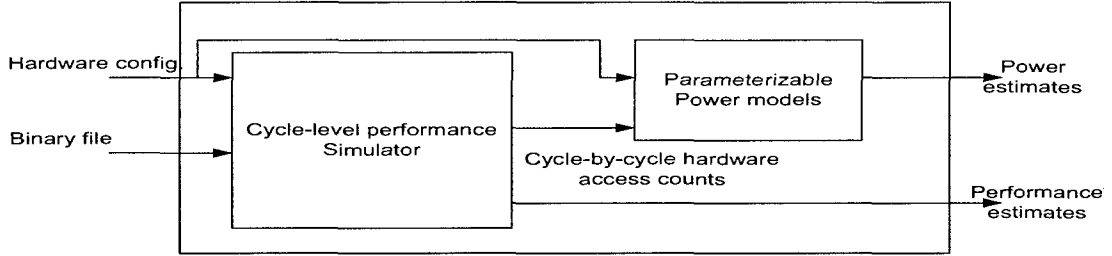


Figure 2.8: Wattch internal organization

Figure 2.8 shows the illustrates the internal organization of Wattch, showing the interface between the architectural simulator and the power models. Wattch’s power modeling methodology classifies microarchitectural block commonly found in a microprocessor into four categories. They are;

- Array Structures: Data and instruction caches, cache tags, register files, register mapping table, branch history tables, and instruction / data load store queue.
- Fully Associative Content Addressable Memories: Instruction reorder buffer and translation look aside buffers.
- Combinational Logic and Wires: Control logic, dependency check logic and all signal buses.
- Clocking: Clock buffers, clock wires, and capacitive loads.

The power for each microarchitectural block is given by the product of the total load capacitance (C), frequency (f), square of supply voltage (V_{dd}) and activity factor (α) i.e. $(P_d = C \times \alpha \times f \times V_{dd}^2)$. Power supply voltage and frequency depend on the process technology used for the design. The activity factor for all microarchitectural blocks are estimated while the cycle accurate simulator, i.e. SimpleScalar, executes the benchmark programs. The exception to obtaining the activity factor from the

cycle accurate simulator is when some blocks that use dynamic logic circuits that pre-charge and evaluate every cycle are used in the design. For such microarchitectural blocks an activity factor of 1 (unity) is assumed. For blocks where measuring activity with the cycle accurate simulator is not possible and are not using dynamic logic circuits, a base activity factor of 0.5 (random switching activity) is assumed. When clock-gating is used in the design, higher level power models modify the activity count for microarchitectural blocks selectively (based on whether the clock to a particular block is gated or not) whereby effectively lowering overall activity factor for the corresponding blocks.

2.2.3.1 Calculating Switching Capacitance

Switching capacitance estimation varies for the four types of microarchitectural blocks commonly found in a microprocessor. The array structure power model is parameterized based on the number of rows (entries), columns (width of each entry), and the number of read/write ports as they affect the size and number of decoders, word lines, bit lines, length of pre-decoder wires, word lines and bit lines. Power consumption of the array consists of the following components; decoder power, word line driver power, bit line discharge and output sense amplifier powers. Word line capacitance includes the diffusion capacitance of the word line driver, the gate capacitance of the memory cell and the capacitance of the word line's metal wire. The bit line capacitance includes the pre-charge transistor's diffusion capacitance, the diffusion capacitance of the memory cell and the capacitance of the bit line's metal wire. The total switching capacitance of the array will be the sum of all the line capacitances and the gate capacitance of the transistors in the decode and sense amplifier.

Content addressable memory structures are analyzed very similar to the array structures. However, in the content addressable memory structure tag lines and match lines are used instead of bit lines and word lines. Switching capacitance for

complex logic blocks are obtained from existing published design literature [39] [40]. Switching capacitance for buses are estimated by multiplying metal capacitance per unit length and length of the wires in the buses. The bus lengths are calculated by assuming microarchitectural block sizes based on published design literature [41]. Clock network on high performance microprocessors are a significant source of power consumption. Clock power consumption consists of long clock net switching capacitance, clock buffers and clock load switching capacitance. An example showing how the switching capacitance values were obtained from the design manual of the Alpha processor is shown in [42].

2.2.4 AccuPower Toolset

AccuPower toolset which is also based on SimpleScalar, modifies the SimpleScalar toolset and includes a power estimation tool. The AccuPower tool consists of three parts, microarchitectural simulator (modified version of the SimpleScalar toolset), physical layouts for major data path components & caches and, the power estimation module that uses coefficients obtained from SPICE simulations and transition counts obtained from the microarchitectural simulator to compute energy/power. Figure 2.9 shows the internal organization and the overall power/energy estimation methodology of the AccuPower toolset [43].

AccuPower incorporates a detailed architectural model including microarchitectural blocks such as the issue queues, the register files, the reorder buffers, the load store queues, the pipeline by-pass mechanisms, multiple levels of onchip caches, interconnections, arbitration blocks, chiplevel I/O traffic blocks and the clock distribution network. Since there is a SPICE engine involved in this toolset, design techniques such as clock gating, voltage and frequency scaling can be incorporated into the toolset allowing detailed design space exploration. For best accuracy, coefficients obtained

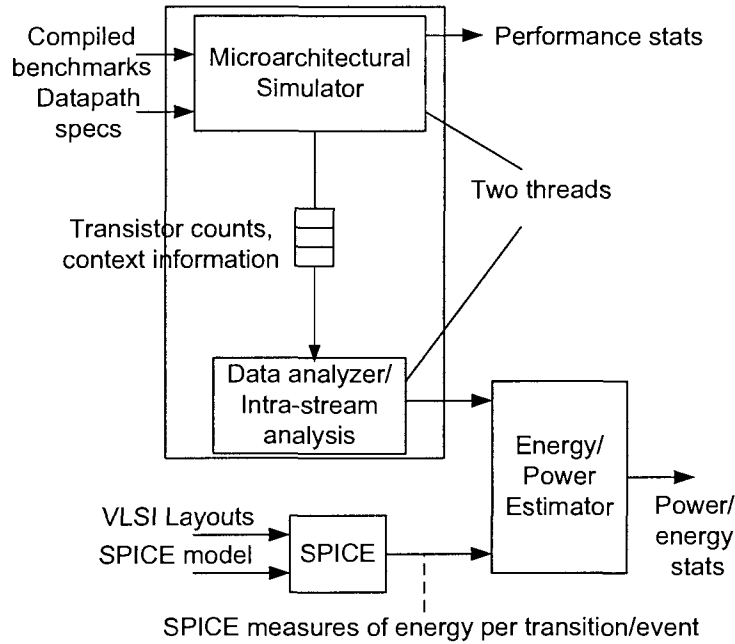


Figure 2.9: AccuPower internal organization

from SPICE measurements of actual VLSI layouts are used. AccuPower uses parameterized models for energy dissipation for major data path components. Pipeline model improvements over SimpleScalar incorporated in Accupower include the splitting of the monolithic cache model into a two level cache model to support multiple cycle cache reads and writes, incorporating level-1 instruction and data cache, a unified level-2 cache, level-1 and level-2 cache contention, level-2 and off-chip memory contention and, a realistic multi stage multi cycle model for dispatch, register rename and operand register and read operations (these were lumped into one cycle in SimpleScalar). Therefore the microarchitectural model incorporated in Accupower is more advanced than the model used in SimpleScalar.

The focus of the AccuPower toolset is to facilitate design space exploration and gauging the impact of well understood circuit design techniques intended for saving power consumption such as clock gating, dynamic voltage and frequency scaling. AccuPower can be used to obtain realistic measurements of bit level data path activity

on the interconnects and dedicated transfer links and, the read and write activities for the register files that form the data path storage components. In addition, the data analyzer modifies the switching counts based on signal bit invariance and measures the average occupancy rates for various microarchitectural blocks. The occupancy rates can be used for dynamic resource allocation studies or just resizing microarchitectural blocks. However, a major drawback of Accupower is its reliance on detailed physical layouts for all microarchitectural blocks in the design, this clearly is not possible at the early design phase when physical implementation has not been commenced and only preliminary layouts may exist.

2.2.5 PowerTimer Toolset

The PowerTimer [44] toolset is an early design phase microarchitectural level power performance analysis tool for modern microprocessors developed by IBM. PowerTimer consists of parameterizable energy functions that can be used in conjunction with any cycle accurate microarchitectural simulator. PowerTimer's internal organization is shown in Fig 2.10. Typically the cycle accurate microarchitectural simulator will target a particular microarchitecture, however for general architectural studies a parameterize microprocessor model, as shown in Fig 2.11, is used along with a cycle accurate simulator called Turandot.

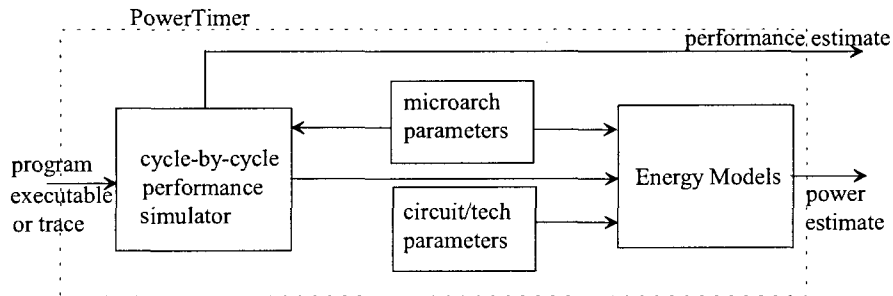


Figure 2.10: PowerTimer internal organization

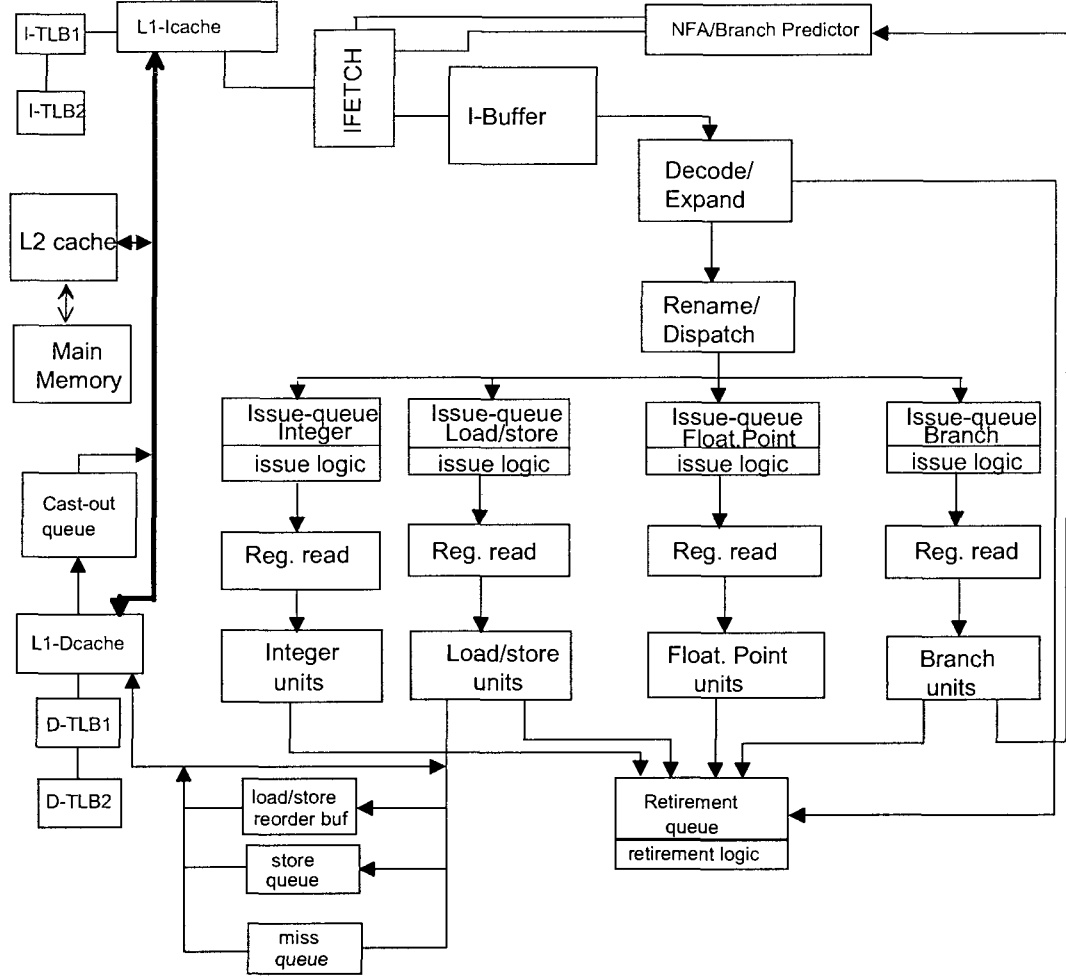


Figure 2.11: PowerTimer pipeline model

The modeled pipeline structure of the parameterized Turandot microprocessor is similar in complexity to modern microprocessors. The model has an in-order front end. The instruction fetch unit on any given cycle, accesses the level-1 instruction cache to fetch the next sequential group of instructions into the instruction buffer. The per cycle fetch window width is a parameter that the user sets. The decode/expand unit is parameterized and can decode up to five instructions per cycle to form a basic instruction dispatch group. Some complex instructions are broken down into micro operations. After the register renaming and dispatch the instructions are issued to one

of the four queues; namely the integer queue, the load/store queue, the floating-point queue and the branch queue. Each instruction issue logic supports out-of-order issue to the execution units. Up to two instructions can be issued per cycle. The model also supports out-of-order execution with in-order retirement using a reorder buffer mechanism. Turandot incorporates a two level cache hierarchy i.e. a split instruction and data L1 caches and a unified L2 cache. Instruction and data translation look aside buffers are also included in the model. The main memory is considered as an infinite perfect storage with a constant parameterized access latency.

PowerTimer uses a variety of sources for developing the power models, such as detailed circuit level power analysis results, extraction tool based estimator results and analytical models derived in a bottom-up modeling methodology. Energy models can be developed based on the following methodologies;

- Microarchitectural level energy models, used in early conceptual design phase is based block level latch counts estimated by the design team. These latch counts are estimated from logic level specifications or area and latch density based projections from prior designs, suitably scaled by technology upgrade parameters. Observing that clocked latches account for 70 to 80% of logic power, a latch based energy model for non array portions of the design is adequate during conceptual phase design space exploration.

- Microarchitectural level energy models can be build on detailed macro level power (SPICE) simulation data of prior existing design. Macros that are reused in subsequent designs are characterized using detailed SPICE simulation. This method is appropriate for both the early design phase as well as the early implementation phase.

Fig 2.12 represents the hierarchial macro based power characterization methodology. Where SF (switching factor) is the average rate at which a particular microarchitectural block is called while executing benchmark workload suites. HoldPower

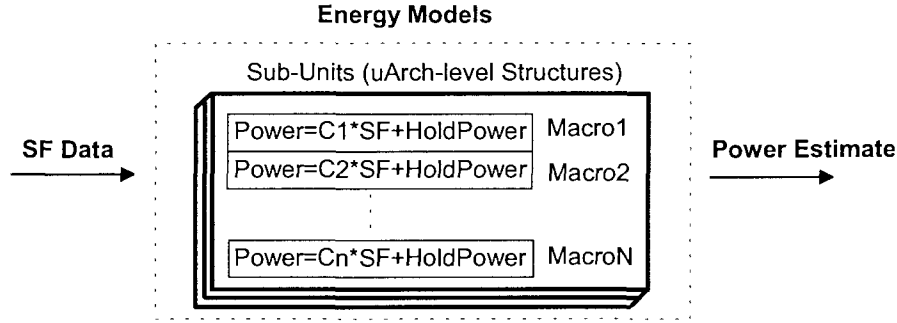


Figure 2.12: PowerTimer microarchitectural block power model

is the quantity used to represent the leakage power a particular microarchitectural block. C_1 through C_n are microarchitectural block dependent slope estimates of the linear power model, assuming N microarchitectural blocks in a design. The total power is the summation of individual microarchitectural block power values.

- Macros that are new and not available in pervious design are characterized by PowerSpice based detailed circuit simulation experiments to obtain energy data for primitive building blocks such as latches, clock-buffers, multiplexors, and interconnects. Analytical equations in terms of high level organizational and technology parameters are formulated to model various combinations and sizes of the primitive blocks when used to build other microarchitectural blocks. The macros are characterized with and without clock gating included.

- Finally, independent analytical power models are used to model power in regular structures such as SRAM array macros. A special tool was developed with the goal of modeling the energy and delay characteristics of IBM-specific SRAM array designs that implement cache macros. But currently designer provided energy models for customized SRAM array macros are used.

The power models can be integrated into the simulator function and run on a cycle-by-cycle basis, calculating the switching factor (SF) along with other perfor-

mance metrics such as the average microarchitectural switching factor, the CPI, the cycle-by-cycle power and the power swing (di/dt). Average power and performance can be obtained by post processing the collected data from the cycle accurate simulator. Examples for using PowerTimer from [44] show that optimizing the quantity cycles per instructions - CPI)³ \times power consumption gives the best power-performance architectural solution.

2.3 RT-Level Power Performance Optimization

Register-transfer level (RTL) power estimation [45] is a popular technique for system power estimation often employed by in-house by designs teams. An RTL description is typically structurally well defined and can be directly mapped onto standard library cells. Tools that operate at the RT-level typically estimate power by aggregating power estimates for its constituent RTL components, such as standard logic gate cells, latches, registers units, memory cells, etc. Accuracy is achieved through a combination of special purpose estimation algorithms and carefully crafted modeling techniques specific to a particular sub-system such as clock sub-system, IO, etc. Power savings achievable with complex fine-grain power management techniques that cannot be modeled at the system level, can be analyzed at the RT-level quickly; enabling identification of localized power hot spots that otherwise would be too difficult to find. A typical RTL power estimation flow is shown in Fig 2.13.

Some of the power saving design techniques that can be evaluated at the RT-level include [46] [47];

- Re-timing - Purely combinational logic between sequential latches can be further partitioned into smaller logic, consequently reducing path delay between latches

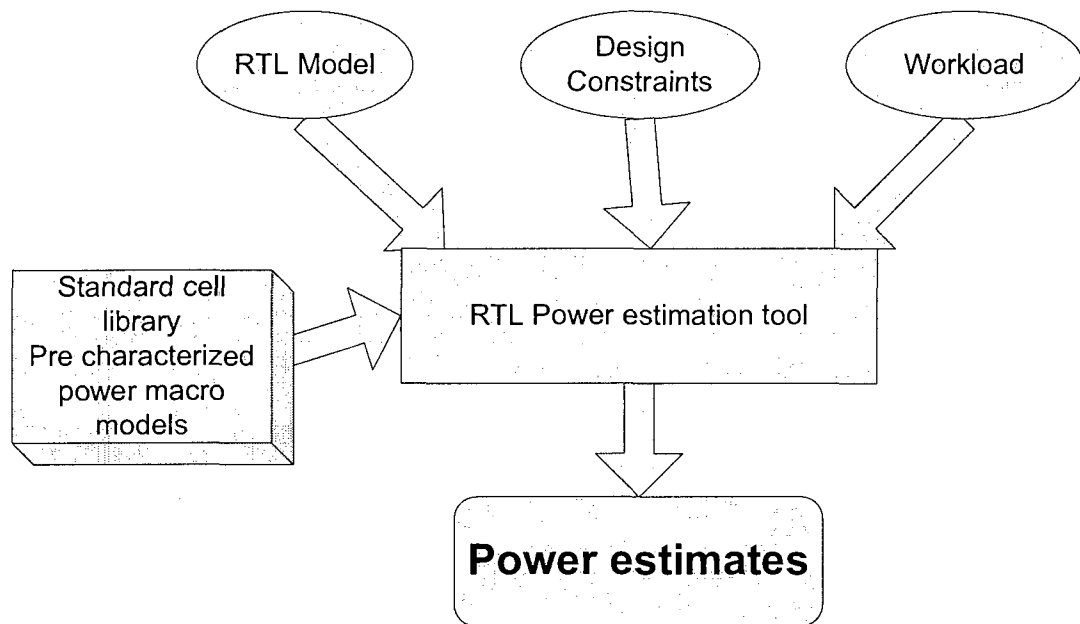


Figure 2.13: Typical RTL power estimation flow

and hence increasing clock speed. When pipe stages in a microprocessor are re-timed the clock rate increases, improving through put but with an increase in dynamic power.

- Bus encoding - Large buses can be encoded or bit-compressed to a smaller number of bits. The encoded information is passed on to be decoded or uncompressed when needed. This strategy help lower the number of toggling bits in a bus, thus reducing transaction dynamic power. However additional logic is required to compress, uncompress, encode and decode the bits.
- Asynchronous designs - Designs can be completely clock-less. This technique removes bulky clock buffers, reduce clock delivery power. However designing hand-shaking logic adds complexity and uncertainty to the design.
- Clock gating and power-aware clock implementation - An alternate technique to asynchronous designs is to turn off clock signals when determined not necessary.

This technique cuts down unnecessary clock power, however there is a delay penalty and a power plane integrity issue due to $\frac{di}{dt}$ during clock signal turning on when required. This technique suits low activity designs better.

- Pulsed latches - A conventional flip-flop is composed of two latches, master and slave. A pulse clock triggered latch is similarly to edge-triggered flip-flop, but half the logic, this saves power. However meeting timing may become an issue with the stringent edge triggered behavior.
- Signal gating - A technique where all signals are gated along with clock signal when a functional module is under a period of inactivity.
- Memory partitioning - Banking memory structures into smaller partitions and activating these smaller partitions as necessary will reduce power by avoiding turning on the entire memory unit at once. Also a smaller partitions will lead to smaller read and write currents.
- Logic partitioning - Partitioning a larger logic block into smaller blocks and sequentially gating power reduces the over all power consumption. This technique and a strong impact on computation delay.
- Low power FSM encoding
- Power-aware synthesis, placement and routing
- Minimum leakage vector technique to reduce leakage current
- Pre-computation logic
- Data path reordering to reduce glitches

The list above lists some of the design choices a designer can make in order to tradeoff power and performance to achieve a design goal. Typically these design choices are handled by RTL optimization tools in the following ways.

- Design choices involving library cell mapping for low power or high performance designs are evaluated by selectively instantiating pre-characterized library power macro models. Analyzing the power models in the context of the entire design improves design optimality.
- Design techniques which modifying input signals such as clock or signal gating, pulse latching etc. require suitable test bench environments so as to evaluate their impact. At the RT-level such test bench environments are simple to develop and quick to test.
- Other design techniques involving changes to the original design under investigation, such as asynchronous designs, memory or logic partitioning, pipeline re-timing, bus encoding etc. require modifying the underlying RTL model to reflect these changes. This is the most tedious of RT-level power performance optimization procedures, however this flexibility is not present at any other level and provides very powerful power performance optimization.

Power estimation and optimization at the RT-level is accurate since at the RTL stage of a design, typically, the library cells are very well defined and well characterized. RTL power estimation is a well understood and mature design technique and many commercial tools such as PowerTheater [48], PowerPro CG [49], Design Compiler Ultra [50], Talus Power [51] and Encounter RTL Compiler [52] to mention a few, are available. A designer iteratively using any one of these tools can perform structural RTL based power performance optimization.

2.4 Physical Level Power Performance Optimization

Power consumption in digital CMOS circuits consists of three components, dynamic power, short-circuit power and leakage power. Dynamic power relates to charging and discharging of the load capacitance at the gate output. Short-circuit power is due to the period of time during signal switching, where both the PMOS and the NMOS trees are on creating a path from VDD to ground and leaking power by shorting them. Static power is due to gate leakage, gate to source and drain tunneling leakage and source to drain leakage in the sub-threshold conduction region. Leakage (static power) is growing exponentially especially in the nanometer CMOS era. Power savings at the physical level is achieved primarily by targeting leakage power and dynamic power. Performance is improved by making transistors switch faster. Through a combination of circuit design choices and processing technology tweaks, power and performance can be optimized.

At the physical level, power and performance can be optimized by techniques such as the ones shown below [46], [47], [53], [54] and [55];

- Transistor sizing - adjusting the size of logic gates in the design so as to optimize power and or performance. By trading critical path transistors in a positive slack path with low leakage transistors, power saving can be achieved without violating timing.
- Dynamic voltage scaling - altering the supply voltage dynamically to meet peak performance requirement when need and the subsequent supply voltage reduction to save power when performance demand has elapsed.
- Voltage islands - multiple supply voltage for various functional blocks based on criticality to optimize power and performance.

- Multiple threshold voltages - low- V_t , nominal and high- V_t FETs are selectively used to improve performance in critical parts of the design, in non-critical parts of the design and high leakage parts of the design; to optimize power and performance.
- Power gating - High- V_t low leakage sleep transistors gate the power supply and puts parts of the design to “sleep” when these parts are not required to switch.
- Non-min-L transistors - selective use of transistors with non minimum (larger than min L) length that have lower leakage and performance.
- Stacking and parking states - stacking off-transistors in series with on-transistors when not switching to reduce leakage.
- Logic design styles - choosing design styles (static, dynamic and pass-transistor logic) based on power-performance targets.
- Process tweak - modifying metal line widths and compositions to reduce series resistance.
- Process tweak - modifying inter metal dielectrics and line pitch to reduce parasitic capacitance.
- Process tweak - modifying threshold voltage, mobility, channel strain, doping density etc., to modify drain current.

Power performance optimization at the transistor or circuit level is the most effective of all techniques to achieve system optimality. However at this level of abstraction there is no flexibility in the models. A complete transistor level power estimation for a full chip may take days if not weeks of computation resources. Moreover such a detailed simulation is very complicated to build and is usually error

prone. When a modular approach to simulation is undertaken the complexity can be greatly reduced. Modular simulations however cannot simulate inter-modular communications and the power associated with such communications. The most common use of circuit level power performance characterization is when the results are rolled up to develop power and performance models at higher levels of abstraction and then analyzed. The best accuracy is obtained if such a process is used. Most of the design techniques in the above list are evaluated using detailed circuit simulator such as SPICE. Process tweaks are reflected in the SPICE model cards used to perform the circuit simulations. One must be careful not to generalize this method, as it can be applied only to designs that are similar to existing designs or designs that have a family of generational designs such as microprocessors of a particular instruction set architecture. Leveraged designs are the most suitable for such a methodology for optimizing power and performance.

Low physical level simulations are unsuitable for system level design optimizations, however empirical analytical models developed based on the low physical level simulation results from previous generation have been found useful for system level design optimization. These analytical models quickly and fairly accurately predict power and performance for any newer design. Moreover these models being parameterized based on physical and process level quantities, are very useful in predicting power and performance of newer leveraged designs in an advanced process technology. Since, the newer physical implementation layouts of majority of the microarchitectural block are scaled versions of older implementation. Speed paths also tend to be similar in structure in the newer designs. Many in-house proprietary solutions exist to perform power performance estimations from circuit level parameters. In the public and academic domains the most notable tool for system optimization based on low level parameters is (Berkeley Advanced Chip Performance Calculator) BACPAC [56] [57]. The following section discusses some salient features of BACPAC.

2.4.1 BACPAC Toolset

BACPAC is a system level power performance analysis tool based on low physical level design parameters. BACPAC comprise of a set of empirical analytical models to collectively form a new system power performance prediction model. Without the need for detailed circuit level SPICE simulations, BACPAC focuses on low level physical design parameters and attempts compile the model equations. BACPAC consists of the following analysis types;

- Delay Analysis
- Noise Analysis
- Dynamic Power Analysis
- Leakage Power
- Short-circuit Power
- Packaging Issues
- Yield Modeling

For delay analysis, BACPAC uses minimum feature size, metal line widths, dielectric constants, metal pitch, line thickness, inter metal dielectric thickness, dielectric constant and metal resistivity. BACPAC then proceeds to calculate the resistance and capacitance of each metal layer based on a three dimensional interconnect model shown in Fig 2.14. The product of unit length metal resistance and capacitance give unit length time constant or delay. BACPAC considers module sizes of 50,000 to 100,000 gates. That is, a design is partitioned into modules of fixed user defined size between 50,000 and 100,000 gates. Figure 2.15 shows the step used by BACPAC to

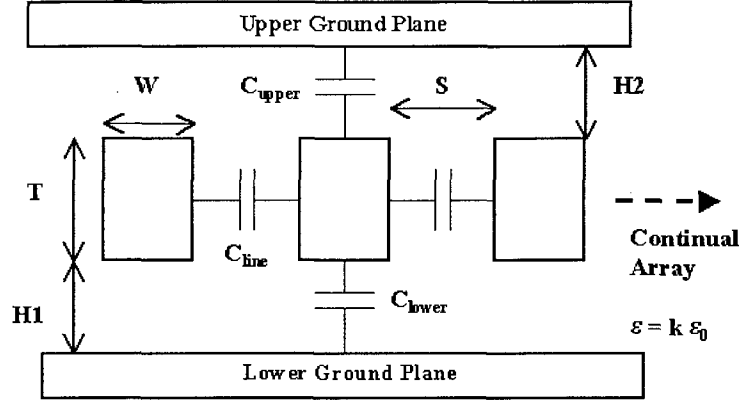


Figure 2.14: Interconnect parasitics model

calculate critical path delay.

Critical path delay = (Total global wire length *times* unit length delay) + (# of gates in critical path \times gate delay) \times (1 + 0.05 + 0.1).

Noise analysis is performed by using Miller capacitance approximation to add coupling capacitance in the equations leading up to calculating the gate delay. Noise analysis estimates the effect of noise on critical path delay and hence system performance.

Figure 2.16 shows the step used by BACPAC to calculate a designs dynamic power. Total dynamic power is given by;

$$P_{dyn.tot} = P_{blocks} + P_{global} + P_{clock} + P_{memory} + P_{IO.pads}.$$

Where alpha (α) is the activity factor. C_{wire} , C_{device} , C_{clock} , C_{pad} and C_{mem} are wire, device, clock network, IO pad and memory switching capacitances.

Leakage power is estimated by the product of the device width, a scaling factor and unit width leakage measured at nominal operating conditions. Thus,

$$Power_{leakage} = W_{device} \times 10\mu A/\mu m \times 10^{-V_t/95mV}.$$

Short circuit power is estimated as,

$$Power_{short_circuit} = \# \text{ of gates} \times 1/2 \times T_{short_circuit} \times I_{peak} \times V_{dd} \times f \times \alpha$$

Where $T_{short_circuit}$ and I_{peak} are the overlap switching time and peak switching

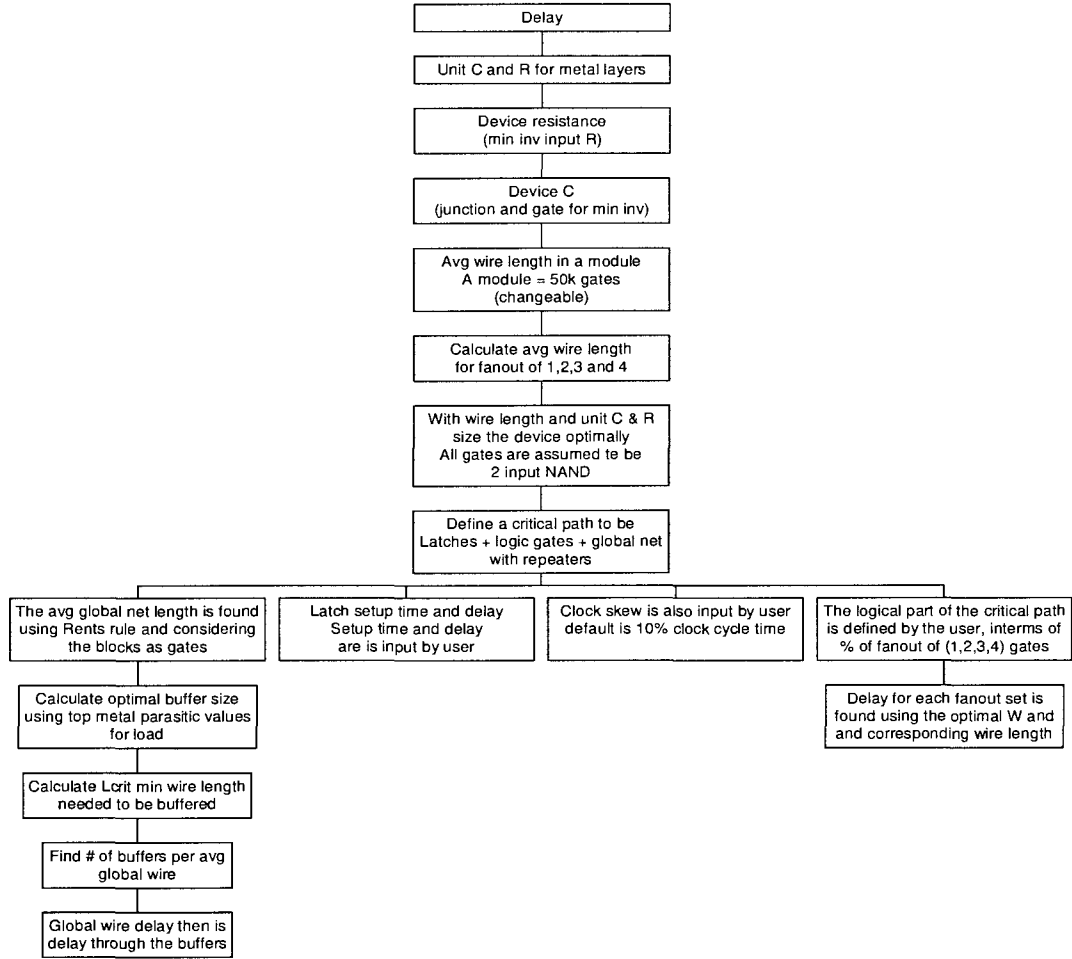


Figure 2.15: Steps to estimated delay in BACPAC

current. Physical parameters such as minimum feature size, metal pitches, dielectric constants and circuit related input parameters including gates per module and number of modules also occur in the model's equations. A design when modeled using BACPAC can be quickly evaluated to obtain power and performance estimates, by performing these evaluations iteratively, a designer can optimize the design for power and performance. Packing issues and yield modeling have been included in BACPAC however they do not pertain directly to system level power performance optimization hence not discussed here in this work.

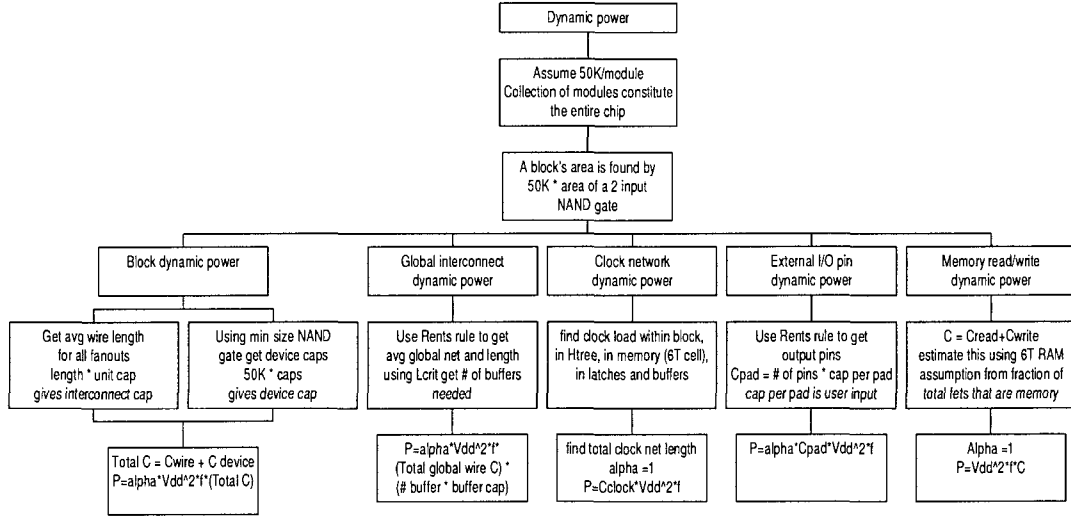


Figure 2.16: Steps to estimated dynamic power in BACPAC

2.5 Shortcomings of Existing Tools and Methodologies

The scope and opportunities for power and performance optimization are maximum when design optimization is performed at the highest level of abstraction. Existing system design optimization tools fall into two broad categories, the first category are tools that are specifically intended for microprocessors or computation engines. Tools in this category include SimplePower, SimpleScalar, Wattch toolset and PowerTimer. These tools model an underlying computation architecture and emulate code execution on the modeled architecture. They collect activity rates, instruction execution rates, miss rates, and prediction efficiency to estimate cycles per instruction (CPI) and other performance metrics. These tools primarily optimize microarchitectural features to maximize computational performance (CPI). Power estimation is a post processing step on the collected metrics and predetermined per-transaction power estimates. The second category of tools are applicable primarily to ASICs to optimize physical implementation. Tools in this category include BACPAC (in the

public domain) and other in-house proprietary tools. These tools estimate power and performance based on low-level physical design parameters and key process parameters.

SimpleScalar was the earliest well recognized toolset for performance estimation of computation engines. SimpleScalar (Sec 2.2.1) essentially is a cycle accurate execution emulator. Emulating the execution of benchmark suites on a targeted processor architecture for a specific instruction set architecture and monitoring every instruction's life in the pipeline; an accurate execution performance is obtained. However a major shortcoming of SimpleScalar was its inability to estimate power consumption of the architectural design being analyzed. Cycle accurate performance simulators like SimpleScalar, when coupled with microarchitectural activity monitors embedded in the toolset can be used to estimate system power consumption. This technique is applicable to microprocessors, embedded processors and computation engines. Typically in such cases performance is measured as number of computation cycles required to execute one instruction - cycles per instruction (CPI). System power however is estimated through the collected activity counts, when unit transaction and unit activity power requirements are known. SimplePower (Sec 2.2.2) toolset an add-on to SimpleScalar, estimates system power consumption by monitoring microarchitectural block activity and using a unit activity power look up table to estimate total system power.

Power estimates from SimplePower include just the dynamic power component of system power. Leakage component of power consumption was omitted. This is a major weakness of the SimplePower toolset, especially in nanometer CMOS era. Secondly, the static nature of the power tables make such an approach unsuitable for design space exploration with dynamically varying circuit level design choices or implementation details. Another major drawback of SimplePower is that clock distribution network power is not included in the tool, which can account for up to 30% of

the total dynamic power consumption in modern designs [58–61]. Wattch toolset (Sec 2.2.3) similar to SimplePower only estimates the dynamic power component of the total power, this is Wattch’s main drawback. These shortcomings make SimplePower and Wattch less attractive for early system level design space exploration.

AccuPower toolset (Sec 2.2.4) which is an extension to the SimpleScalar toolset incorporates library cell characterization data and detailed physical layouts to estimate dynamic and leakage power. Parameterized analytical power models are used in lieu of physical layouts when their implementation does not exist. A similar approach is used in the PowerTimer toolset (Sec 2.2.5) as well. PowerTimer is a ground up power modeling tool which builds power models from the bottom up and has the ability to evaluate dynamically varying design choices during design space exploration. PowerTimer and AccuPower toolsets are have high fidelity, accuracy and detail. However, they are less flexible and are not suitable for “quick” power performance predictions which are required for design space exploration. This is due to the nature of the power models which are build from the bottom up. During design space exploration when a design is modified, power estimates cannot be obtained without re-simulating each module and this results in large time penalty between iterations. As a result these two toolsets are not suitable for rapid early design space exploration.

Design space exploration and power performance optimization performed at the RT-level using a complete RTL description of the design can be relatively efficient for designs of smaller size. However for larger designs such as a modern microprocessor, design space exploration and power performance optimization performed at the RT-level becomes time consuming and inefficient. RTL model based power estimation depends on the availability of fully characterized library cell and macros. While their availability is possible further along the design cycle, they may not exist during the early design exploration phase.

The BACPAC toolset (Sec 2.4.1), uses parameterized analytical power and performance models for system power and performance estimation. BACPAC attempts to recreate a design as a collection of **equal sized** modules, which is an acceptable approach for ASIC designs but not suitable for modeling or optimizing highly modular designs such as microprocessors and other high performance designs. Moreover BACPAC models leakage power as a linear function of the total gate width in the module, which is too simple to capture the exponential increase in leakage power in nanometer CMOS. Furthermore BACPAC's analytical models lack the ability to access the the impact of applying additional circuit level design choices which are applied to microarchitectural blocks to either reduce power or improve their performance. The BACPAC toolset provides the correct direction for a system level design optimization tool. However, due to the insufficiencies in its modeling methodology it is not suitable for rapid early design space exploration of large designs such as microprocessors and other high performance designs.

The work described in this dissertation is motivated by the lack of fast and effective design space exploration tools and the shortcomings of the existing approaches. Chapter 3 introduces the proposed approach, design methodology and the envisioned framework (tool) to perform rapid early design space exploration of modern VLSI designs.

Chapter 3

The Proposed Approach

3.1 Overview

Due to challenges faced by designs in nanometer CMOS arising from business and cost aspects, the design cycle time and time-to-market stipulation are becoming progressively more stringent. It was shown that a thorough design space exploration using system level models during the early design phase is capable of placing a design in an optimal design sub-space and improve design convergence. The challenge in performing such design space exploration was the lack of suitable design tools and aids and, design tools capable of modeling and analyzing large systems such as microprocessors and high performance designs were needed. To this affect, a new methodology for modeling large systems and performing rapid early design phase design space exploration was proposed. In this chapter the proposed high level modeling methodology, the proposed design target prediction models and the proposed methodology for design space exploration are detailed.

3.2 Proposed High Level Modeling Methodology

EIDA (Early Integrated circuit Design Assist) is a framework for early design space exploration. It consists of two components; the high level models (modeling

methodology) and the analytical models for design target prediction. Design space exploration using EIDA involves iteratively building modular level models, assigning module-granular circuit-level design choices, and estimation of the expected achievable system targets when the design solution is implemented. A system is defined as a collection of modules and each module is characterized by a set of module descriptors, called descriptor vector. A collection of descriptor vectors form the system model.

Consider a leverage redesign of an existing system design in a newer process, where portions of the system critical path lie in a subset of modules called critical modules, as shown in Fig 3.1. To estimate system performance, system critical path delay i.e. sum of modular critical path delays of critical modules are necessary. Legacy data can be used to get an initial estimate of the modular critical path delay values. When complete bottom-up design data is not available, abstracting the underlying design is necessary so as to estimate critical path delay. A module's critical path delay depends on number of gates and interconnect length along the path and typical gate delay. Other factors such as additional circuit-level design choices influence critical path delay. Therefore factors such as legacy performance, total interconnect length and the ratio of gate to interconnect delay are elements in the module descriptor vector and in the performance estimation model (Section 3.4). To estimate power consumption, dynamic and leakage power for all modules are necessary. Dynamic power depends on total switching capacitance, operating frequency, power supply and switching factor. Leakage power depends on gate area for gate leakage, junction area for junction leakage and total width of active devices that are off for sub-threshold leakage. These factors among others are elements in the module descriptor vector and in the power estimation models (Section 3.4).

The module critical path can be modeled as a series of inverters or NAND gates. An equivalent logic gate is defined as a standard sized inverter with a fanout of four

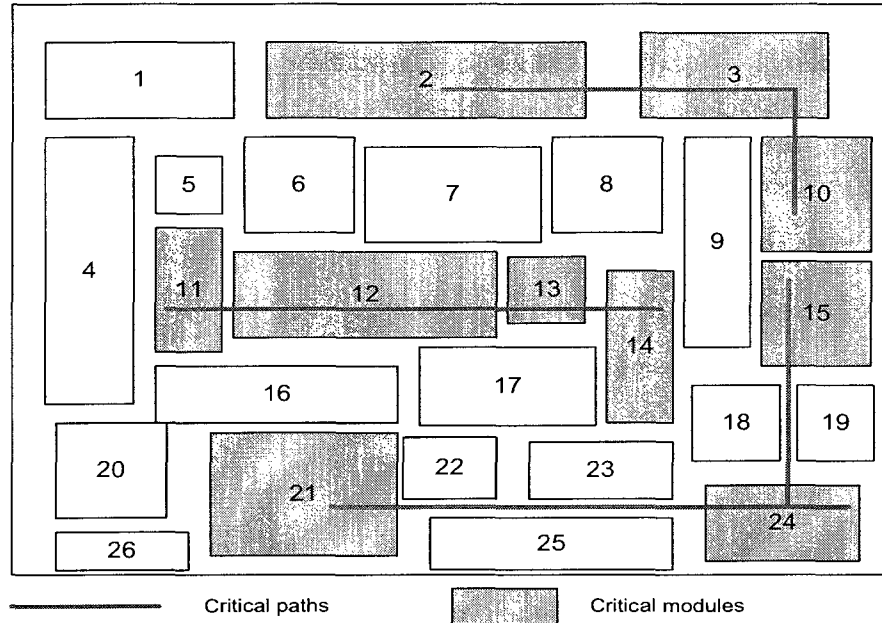


Figure 3.1: Critical paths and modules

(FO4) load, its propagation delay is defined by equivalent logic gate delay (EGD). The delay of any logic gate can be expressed in terms of EGDs and the critical path delay is the sum of EGDs of all gates along the path. That is, a non-standard sized inverter may have a delay of 3.75 EGDs and if the critical path consists of 10 of these then, the critical path delay is 37.5 EGDs. For cells in a given library their corresponding EGDs are characterized and known. Using EGDs to express delay normalizes process technology, and the critical path delay can be expressed in a technology independent manner. The use of EGD captures logic delay including nominal load on each logic gate. Significant RC delays on a path is not captured by the EGD metric. These RC delays need to be added to the logic delay to get total path delay. For leveraged redesigns the ratio of RC delay to logic delay for each module in the legacy design is known. Therefore the RC delay can be estimated as a function of the logic delay and the ratio of RC delay to logic delay.

For power estimation, a module's underlying physical implementation is abstracted as an inverter (the module inverter) with a FO4 load, as shown in Fig 3.2. Note that this representation is different from the standard sized inverter FO4 configuration used for EGD calculation. The module inverter sizing is proportional to the total P and N FET widths in the module which can be obtained by scaling the legacy data. Total power consumption consists of dynamic and leakage power. Leakage power consumption for a design is primarily related to the total gate area, total junction area and total device width in the design. These physical design descriptors can be obtained by scaling legacy design data. Also, leakage is a strong function of the transistor stacking effect, which is captured by including the stacking factor in the analytical model (Eqn 3.14). Dynamic power consumption on the contrary, depends on physical design descriptors and operating frequency; where operating frequency is estimated using the critical path delay. The FO4 connected module inverter representation forms the basis for macro-model generation used to estimate power and performance impact of technology scaling and the application of additional circuit-level design choices; through in-situ simulations.

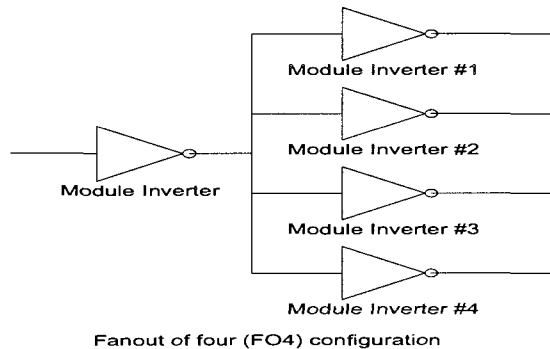


Figure 3.2: Fanout of four configuration

Elements in a module descriptor vector are primarily obtained from legacy data from previous generation designs or estimates through other sources. Power and performance of leveraged designs can be estimated by modeling switching capacitance

scaling and power supply scaling. For example, the module inverter size which reflects the module's active transistor area and total switching capacitance obtained from legacy data is used to estimate scaled active transistor area and total switching capacitance. Some descriptor values such as supply voltage, min FET length etc. are obtained from the new process technology specs.

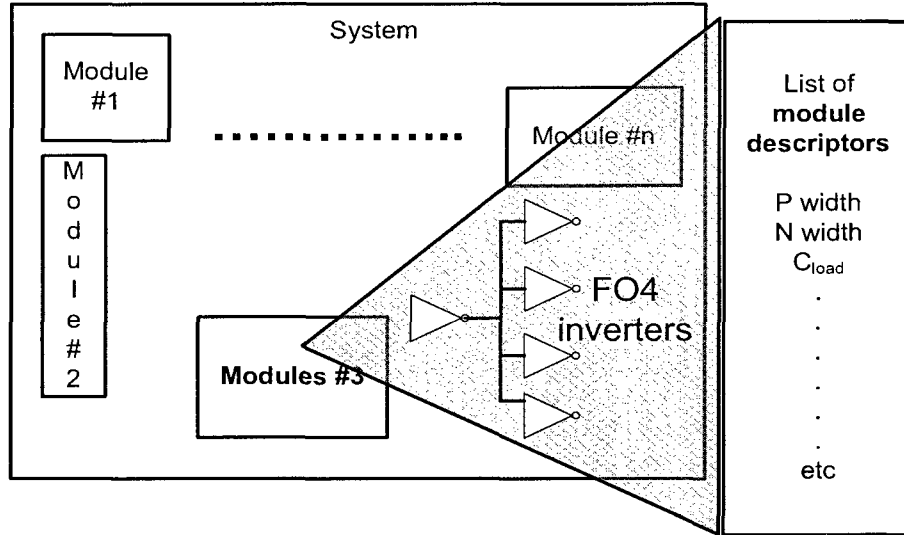


Figure 3.3: A partitioned system shown with module #3 abstracted with FO4 inverters and its corresponding descriptor vector

In-situ SPICE simulations are used to ascertain circuit-level design-choice dependent descriptors such as sleep transistor performance correction (STPC) etc. Fig 3.3 illustrates the proposed approach of using FO4 module inverters to abstract the physical implementation along with module descriptors to estimate power and performance from analytical models.

3.3 Module Descriptor Vector Elements

The descriptor vector is a set of parameters from multiple sources that are used to model and predict system power and performance. Elements in the descriptor

vector enable the inclusion of physical constraints while performing system level optimizations.

3.3.1 Legacy Design Descriptors

Table 3.1 lists descriptor vector elements obtained from prior/legacy designs along with their explanations.

Table 3.1: Descriptors from legacy design

Descriptor	Explanation
C_{orig}	The total load capacitance in legacy design
TWL	The total wire length in legacy design
ULC	Unit length capacitance in the legacy design (all metal layers)
$W_{total}/(P/N)$	The total/(P/N) fet width in legacy design
L_{min}	The minimum fet length in legacy design
f_{old}	operating frequency of the legacy design
ASF	Average switching factor in the legacy design
CGF	Clock gating correction to activity factor
DECAP_SENS	supply voltage change due to unit de-coupling capacitance insertion
C_{de_cap}	De-coupling capacitance added in legacy design
sf	Stacking factor
HVR	Ratio of # of high V_t to total fets in legacy design
LVR	Ratio of # of low V_t to total fets in legacy design
C_{wire_buff}	Total buffer capacitance added in legacy design
TPFR	Typical path fet ratio i.e. ratio of logic delay to total path delay in a typical speed path
USPE	Useful skew allocation performance enhancement factor
BSUF	Speed up factor due to buffer insertion
C_{unit_old}	Unit area gate capacitance in legacy design
I_{ds_old}	min W&L, drain current in legacy design
temp	Typical operating temperature in legacy design

For example, with the rate of supply voltage scaling diminishing and drain induced barrier lowering (DIBL) effects becoming stronger with technology scaling, the effectiveness of leakage reduction by stacks becomes higher. In complex CMOS gates

where FETs are stacked, this effect captured by the stacking factor descriptor (sf) defined as the ratio of single device leakage to stack leakage. The sf descriptor is typically obtained from legacy design. Average switching activity of a module (ASF) and clock gating (CGF) used in legacy design are very important for dynamic power consumption estimation. Clock gating factor is calculated as the average fraction of time the clock signal is gated. Interconnect and gate delay component proportions of the critical path delay are described by the typical path FET ratio (TPFR) descriptor which is the ratio of critical path gate delay to the total critical path delay. C_{wire_buff} is the additional switching capacitance introduced due to interconnect buffer insertion and the buffer speed up factor (BSUF) captures the speed up in critical path interconnect delay due to buffer insertion. The useful skew allocation performance enhancement (USPE) descriptor (unity by default) reflects critical path timing improvement as a result of positive skew allocation and combinatorial circuit re-timing and redesign.

3.3.2 Target Process Technology Descriptors

Table 3.2 lists descriptor vector elements and their explanations which are obtained from the new process technology specifications, to which the design is to be ported. Scaling factors, unit capacitance and supply voltage are typically well defined for any process. Some of the descriptors can be obtained from the SPICE model for the new process. For example, the supply voltage at the highest metal power grid on-chip is lower than the voltage at the package power supply terminals due to package power network's IR drop. This effect is characterized by the η descriptor which characterizes the expected drop due to the package power network. The RC slowdown factor (RCSF) descriptor captures the change in unit length (the most representative interconnect length in the legacy design) interconnect delay in the new process com-

pared to legacy interconnect delay, when driven by comparably drivers. This factor is typically obtained from process characterization data.

Table 3.2: Target process technology descriptors

Descriptor	Explanation
V_{dd_spec}	Voltage at package supply in new chip
η	Package IR drop factor
V_{dd_bump}	Voltage at supply bumps in new chip ($V_{dd_spec} \times \eta$)
s_{gate_cap}	scaling factor for gate capacitance in new process
s_{wire_cap}	scaling factor for wire capacitance in new process
s_{width}	Width scaling factor in new process
I_{leak_junc}	Unit junction leakage current in new process
I_{leak_gate}	Unit gate leakage current of in new process
I_{othv}	Typical I_{off} for high V_t fet in unit linear μm
I_{owhv}	Worst I_{off} for high V_t fet in unit linear μm
I_{otlv}	Typical I_{off} for low V_t fet in unit linear μm
I_{owlv}	Worst I_{off} for low V_t fet in unit linear μm
$I_{gate_per_w}$	Gate tunneling current for unit linear μm
RCSF	RC slowdown factor - slow down due to wire scaling
C_{unit_new}	Unit area gate capacitance in new process
I_{ds_new}	min W&L, drain current in new process

3.3.3 In-situ Simulations and Descriptors

Leveraged redesigns often involve additional features and use additional circuit-level design choices. Under such circumstances scaling legacy design data for design space exploration alone is not sufficient. Therefore, macro models are generated for in-situ simulation to determine the relevant descriptor values for various circuit-level design that were not available from the legacy design but may be needed in the new design. Using macro models to allow designers to **quickly** evaluate the impact of applying a particular circuit-level design choice or a technology process tweak. Additionally, in-situ simulation improve the accuracy of design target (power and performance) prediction models. Macro-model generation and in-situ simulations are

necessary and performed when the situations listed in Table 3.3 occur during design exploration. The corresponding descriptor vector elements and their explanations are listed in Table 3.4.

Table 3.3: Circuit-level design choices requiring In-situ simulations

S.No	Description of design choice
1	Using multiple threshold FETs i.e. using nominal V_t as non-critical and low V_t as critical path FETs, to improve performance. [54]
2	Inserting sleep transistor and turning off power supply to a module when it is inactive to reduce power consumption. [54]
3	Using adaptive body biasing; PFET's body i.e. nwell tied to V_{dd} or forward biasing (FB) to improve performance. [53]
4	Reverse biasing (RB) nwell to reduce leakage power.

Design options listed in Table 3.3 affect both power and performance. The STC descriptor captures the effective leakage power saving obtained when the module's power is gated by the sleep transistor. Using a FO4 inverter macro model, in-situ SPICE simulations are used to calculate this descriptor. Similarly the STPC descriptor captures the device performance degradation due to sleep transistor wake-up time. In-situ simulations on FO4 inverter macro model are used to determine the effects of using back body biasing design technique. The ABBC descriptor capture the change in device leakage currents when using back body biasing and ABBPC descriptor captures the change in device performance. When the additional circuit-level design choice of using low- V_t FETs along the critical path to improve critical path delay is applied, both performance and leakage power consumption increases. Leakage power increases due to excessive sub-threshold leakage in the low- V_t FETs. To determine the impact of using low- V_t FETs on power and performance, in-situ simulations on macro model formulated as nine stage ring oscillator with all nominal and all low- V_t FETs are performed. The performance improvement achieved through using low- V_t

FETs (the DVTC descriptor) can be calculated from the observed operating frequencies of the two ring oscillator macro models. DVTC descriptor alters the estimated operating frequency and thus the dynamic power consumption. The impact on leakage power is indirect and achieved through the HVR and LVR descriptors and does not require in-situ simulations. Low- V_t FET leakages are typically pre characterized and using them additionally in a design does not alter their leakage characteristics. Table 3.4 lists the descriptors needed to include the effect of applying these design choices while design target prediction. The effect of varying supply voltage for what-if analysis is described in next section.

Table 3.4: Descriptor vector elements from in-situ simulations

Descriptor	Explanation
STC	Leakage power correction factor due to power savings obtained by sleep transistor insertion.
ABBC	Leakage power correction factor due to change in leakage power by applying adaptive body biasing.
DVTC	Performance improvement in critical path delay due to dual- V_t FETs in the critical path.
ABBPC	Performance correction factor due to change in transistor delays by applying adaptive body biasing.
STPC	Performance impact for the corresponding power saving due to sleep transistor insertion.

3.4 Proposed Analytical Power and Performance Modeling Methodology

3.4.1 Dynamic Power

The dynamic power of a generic block is given by equation 3.1.

$$P_{dyn} = C_{new} \times V_{dd_new}^2 \times f_{new} \times RPSF \quad (3.1)$$

$$C_{new} = \{C_{orig} \times ((s_{gate_cap} \times frac_{fet}) + ((1 - frac_{fet}) \times s_{wire_cap}))\} + C_{wire_buff} \quad (3.2)$$

$$V_{dd_new} = (V_{dd_bump} \times SLPZD) + DECAPC \quad (3.3)$$

$$f_{new} = (f_{predict}) \times ASF \times CGF \quad (3.4)$$

$$SLPZD = \frac{(V_{dd_bump} - NADSP)}{V_{dd_bump}} \quad (3.5)$$

$$RPSF = \frac{\sum_{i=1}^{no_of_library_cells} W_i \times \frac{Cell_i_power_original}{Cell_i_power_after_redesign}}{no_of_library_cells} \quad (3.6)$$

$$DECAPC = DECAP_SENS \times C_{de_cap} \quad (3.7)$$

$$C_{orig} = C_{fet} + C_{wire} \quad (3.8)$$

$$C_{wire} = TWL \times ULC \quad (3.9)$$

$$Area_{fet} = W_{total} \times s_{width} \times L_{min} \quad (3.10)$$

$$C_{fet} = Area_{fet} \times C_{ox} \quad (3.11)$$

$$frac_{fet} = \frac{C_{fet}}{C_{orig}} \quad (3.12)$$

The re-design power savings factor (RPSF, Eqn 3.6) captures the power savings obtained through improvements to standard library cells. Average power savings from individual library cells measured using simulations are averaged with appropriate weights (W_i in Eqn 3.6) to obtain the RPSF descriptor value. The total switching capacitance in the module (C_{new} , estimated by Eqn 3.2) is a function of the total gate switching capacitance and interconnect load capacitances. Total gate switching capacitance obtained from scaling the original legacy switching capacitance (C_{orig}) by the gate capacitance scaling factor (s_{gate_cap}), wire capacitance scaling factor (s_{wire_cap})

and the fraction of gate capacitance to interconnect capacitance (frac_{fet}). Any additional switching capacitance introduced due to the addition of interconnect buffers is added to the scaled value and given by C_{wire_buff} .

V_{dd_new} is the effective supply voltage at the supply rails, which can be altered by inserting de-coupling capacitance and power supply noise due to switching activity. This value is estimated using Eqn 3.3 and depends on the supply bump voltage (V_{dd_bump}), estimated supply droop due to a switching activity (SLPZD) and correction to supply voltage due to de-coupling capacitance insertion (DECAPC). Supply droop due to switching is estimated using Eqn 3.5, where NADSP is the average measured power supply droop under typical switching activity rates of a power supply network in the new process. Supply voltage correction due to de-coupling capacitance insertion is estimated using supply network sensitivity to unit de-coupling capacitance (DECAP_SENS) and total de-coupling capacitance inserted (C_{de_cap}) as in Eqn 3.7.

Switching activity and clock gating both affect the switching activity of a module, this is included in dynamic power calculations by altering the predicted operating frequency ($f_{predict}$) for calculation purposes by the average switching factor (ASF) and clock gating factor (CGF). This results in an equivalent operating frequency (f_{new} , Eqn 3.4) which reflects the actual switching activity of the module; dynamic power is estimated using the equivalent operating frequency.

3.4.2 Leakage Power

Leakage power for a generic block is given by Eqn 3.13.

$$\begin{aligned}
 P_{leak} = & \{[(V_{dd_bump} \times I_{off}) + (V_{dd_bump} \times (I_{gate} + BUFFLC))] \\
 & + (V_{dd_bump} \times I_{junc})\} \times \varphi + [V_{dd_bump} \times DECAPLC]
 \end{aligned} \tag{3.13}$$

$$I_{off} = \frac{1}{2} \times \left[\left(\frac{W_{total} \times s_{width}}{sf} \times HVR \times A \right) + \left(\frac{W_{total} \times s_{width}}{sf} \times LVR \times B \right) \right] \quad (3.14)$$

$$A = e^{\left(\frac{\ln(I_{othv}) + \ln(I_{owhv})}{2} \right)} \quad (3.15)$$

$$B = e^{\left(\frac{\ln(I_{otlv}) + \ln(I_{owlv})}{2} \right)} \quad (3.16)$$

$$I_{gate} = A_{gate} \times s_{width} \times I_{leak_gate} \quad (3.17)$$

$$I_{junc} = A_{sd} \times I_{leak_junc} \quad (3.18)$$

$$A_{sd} = \tau \times A_{gate} \quad (3.19)$$

$$DECAPLC = A_{de_cap} \times I_{leak_gate} \quad (3.20)$$

$$BUFFLC = \frac{1}{2} \times A_{wire_buff} \times I_{leak_gate} \times ASF \quad (3.21)$$

$$\varphi = STC \times ABBC \quad (3.22)$$

The total module sub-threshold leakage current (I_{off}), which is the average of the P and N tree leakage currents, is estimated using Eqn 3.14. Sub-threshold leakage is a function of the total device width (W_{total}), type of FET i.e. nominal or low- V_t , typical unit width leakage values ($I_{othv}/owhv/otlv/owtv$), process width scaling factor (s_{width}), average stacking factor for cell in the new process library (sf) and the fraction of low- V_t FETs to nominal FETs (LVR , HVR). The total module gate leakage current (I_{gate}) is estimated using Eqn 3.17 and is a function of the total gate area in the module (A_{gate}), width scaling factor (s_{width}), and unit width gate leakage current (I_{leak_gate}). The total module junction leakage current (I_{junc}) is estimated using Eqn 3.18 and is a function of source and drain junction area (A_{sd}) and unit junction area leakage (I_{leak_junc}). Values for the unit width sub-threshold leakage and unit area gate and junction leakages are determined from the new process' foundry characterization data.

An empirical constant determined by process and device layout rules (τ in Eqn 3.19) is used to estimate the average source and drain junction areas based on gate area. The impact of applying either sleep transistors or back body biasing to the module is captured by the effective leakage modifier (φ) as defined in Eqn 3.22. Additional gate leakage due to interconnect buffers insertion (to improve interconnect delay) and de-coupling capacitances insertion (to improve power grid integrity) are captured by the factors BUFFLC and DECAPLC, respectively.

3.4.3 Operating Frequency

The max operating frequency of a generic block can be calculated from Eqn 3.23.

$$f_{predict} = \frac{((f_{old} \times HVR) + (f_{old} \times LVR \times DVTC)) \times DIF}{(TPFR \times FSF) + ((1 - TPFR) \times RCSF_{\ddagger})} \times \Theta \quad (3.23)$$

$$RCSF_{\ddagger} = \frac{RCSF}{BSUF} \quad (3.24)$$

$$\Theta = STPC \times ABBPC \times USPE \quad (3.25)$$

$$FSF = \left[\frac{V_{dd_spec} - X}{V_{dd_bump} \times SLPZD \times (SLPZD - X)} \right]^Y \quad (3.26)$$

$$DIF = \frac{C_{unit_old}}{C_{unit_new} \times s_{width} \times \frac{L_{min_new}}{L_{min_old}}} \times \frac{I_{ds_new}}{I_{ds_old}} \quad (3.27)$$

Design choice dependent factors affecting $f_{predict}$ are, legacy operating frequency (f_{old}), nominal to low- V_t FET ratios (HVR, LVR), critical path speed up due to low- V_t FETs (DVTC), performance impact correction due to sleep transistor insertion (STPC) and performance impact correction due to back body biasing (ABBPC). Empirical factors affection $f_{predict}$ are, expected interconnect slow down due to interconnect RC scaling (RCSF), expected interconnect buffer insertion speed up attainable (BSUF) and the expected ratio of a typical critical path gate delay to total critical path delay (TPFR). Process dependent factors affection $f_{predict}$ are, ratio of

unit area gate capacitance in the old and new process (DIF), ratio of minimum size device drive currents in old and new process (DIF) and the impact of supply voltage droop on FET delay (FSF). FSF (FET slowdown factor) is a factor that captures FET switching speed dependence on power supply voltage and environmental power grid noise. Propagation delay measurements from SPICE simulations on standard logic gates in the new process technology are used to obtain X and Y parameters in Eqn 3.26, by curve fitting the SPICE delay measurement data (Section 3.4.5.). The operating frequency (reciprocal of the critical path delay) for a module is estimated by Eqn 3.23. For modules that have new features with no legacy data, critical path delay estimated using the number of equivalent gate delays and TPFR are used instead of legacy data in Eqn 3.23.

3.4.4 Effect of V_{dd} Scaling

Lowering supply voltage to reduce total power consumption or elevating supply voltage to improve performance is a standard design technique [53]. When a module's supply voltage changes, that affects module dynamic, leakage power, and operating frequency. The impact of supply voltage change on dynamic power can be directly estimated from the expression for dynamic power (Eqn 3.1). However, the impact of supply voltage change on gate leakage, junction leakage, sub-threshold leakage, and device slowdown/speedup (operating frequency) are indirect and have to be determined individually. Supply voltage does not directly appear in the expressions for total sub-threshold current (Eqn 3.14), total gate leakage current (Eqn 3.17) and total junction leakage current (Eqn 3.18). Moreover, unit width or unit area leakage values at nominal supply voltage are used to estimate leakage currents and the device slowdown/speedup (i.e. FSF in Eqn 3.26) was calculated at nominal supply voltage.

Clearly, leakage and gate delay are supply voltage dependent. Detailed models for leakage currents such as models described in [62] exists and can be used to accurately

calculate the leakage currents under varying supply voltages. However, these models require detailed physical device analysis and utilize additional process parameters that are hard to extract or not readily available. Therefore the task of determining the impact of supply voltage on leakage current can be simplified by using simple empirical equations that capture the general relationship of the detailed models to the first order, as shown in Eqns 3.28 - 3.34.

$$V_{dd_bump} = \eta \times V_{dd_spec} \quad (3.28)$$

$$a = \frac{V_{dd_spec}}{V_{dd_applied}} \quad (3.29)$$

$$I_{leak_junc_new} = I_{leak_junc} \times e^{\alpha \times (a-1)} \quad (3.30)$$

$$I_{leak_gate_new} = I_{leak_gate} \times e^{\beta \times (a-1)} \quad (3.31)$$

$$I_{off_new} = I_{off} \times e^{\gamma \times b} \quad (3.32)$$

$$b = \frac{a^\delta + a^{2\delta} + a^{4\delta} + a^{6\delta}}{4} - 1 \quad (3.33)$$

$$FSF_new = FSF \times \frac{(1/a)^{2\epsilon} + (1/a)^{\epsilon/2}}{2} \quad (3.34)$$

Suitable test circuits are simulated using the new process SPICE model to measure the leakage currents and device delay by sweeping the supply voltage. The measured values are then used to determine the analytical approximation coefficients ($\alpha, \beta, \gamma, \delta$ and ϵ) by curve fitting. Establishing the analytical approximation prior to design space exploration, obviates the need for repetitive characterization runs, thus speeding up the estimation of power supply scaling on leakage currents and device delays. V_{dd_spec} is the nominal power supply voltage and $V_{dd_applied}$ is the scaled/alterd supply voltage applied. η is a factor characterizing package supply network's IR drop. $I_{leak_junc_new}$, $I_{leak_gate_new}$, I_{off_new} and FSF_new are unit area junction, unit area gate, unit width sub-threshold leakages and FET slowdown

expected at the altered supply voltage and I_{leak_junc} , I_{leak_gate} , I_{off} and FSF are the corresponding leakages and FET slowdown at nominal supply voltage. When a module's supply voltage changes, the new unit leakage values and FET slowdown are calculated and then these values are used instead of the nominal values to estimate module leakage, dynamic powers and operating frequency.

3.4.5 Procedure to Find Coefficients X and Y in Eqn 3.26

The values of these curve fitting parameters (X and Y) are obtained from SPICE simulations performed on a test circuit using the appropriate device model. The test circuit consists of inverters of varying sizes with appropriate FO4 load. Inverters sizing starts with minimum the size (1X). Other sizes used in the simulations are 10X, 100X and 1000X the minimum size where the 100X and 1000X inverters are suitable fingered. Propagation delay of these devices is defined as the average of high-to-low and low-to-high 50%-to-50% V_{dd} transition times. Propagation delay for the inverters for V_{dd} sweep of $V_{dd} \pm 40\%V_{dd}$ in steps of 1% V_{dd} is measured.

The FSF (FET slowdown factor) is a measure of the change in propagation delay of the devices under various applied supply voltages, with respect to nominal supply voltage. Analytically FSF is given by Eqn 3.35. By curve fitting the measured propagation delay values for the various supply voltages, the coefficient values of X and Y in Eqn 3.35 are obtained.

$$FSF = \left[\frac{V_{dd_spec} - X}{V_{dd_bump} \times SLPZD \times (SLPZD - X)} \right]^Y \quad (3.35)$$

$$V_{dd_bump} = V_{dd_spec} \times 0.95 \quad (3.36)$$

$$SLPZD = \frac{(V_{dd_bump} - NADSP)}{V_{dd_bump}} \quad (3.37)$$

3.4.6 Procedure to Find ABBC, ABBPC, STC, STPC and DVTC Descriptors

Applying module granular circuit level design choices, namely, adaptive body biasing, sleep transistor insertion and, using dual threshold FETs require background SPICE simulation to ascertain their impact on module power and performance.

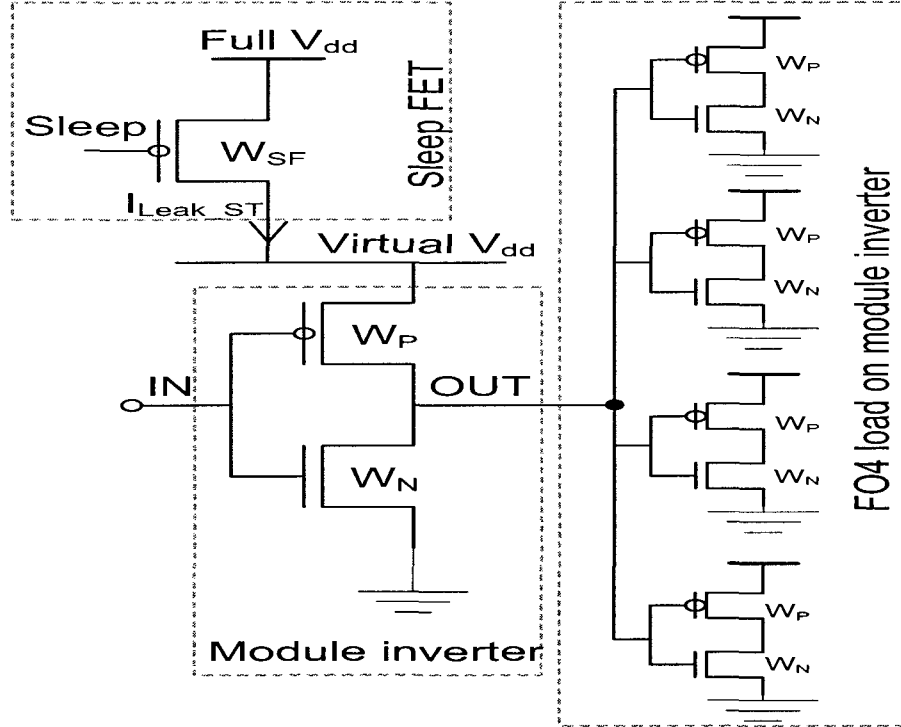


Figure 3.4: Experiment to obtain STC and STPC factors

Fig 3.4 shows the setup used to determine the impact of applying sleep transistor insertion to a module. The width of the sleep transistor (W_{SF}) has tremendous influence on the benefits of applying this design choice to the module, it is hence a very important design specification. To optimize sleep transistor size and its impact on a block, an algorithm (described in Table 3.5) which uses the relation in Eqn 3.38 was developed.

$$\left(\frac{W}{L}\right)_{SleepFET} = \frac{I_{sleep}}{\mu_n \times C_{ox} \times (V_{DD} - V_t) \times V_{sleep}} \quad (3.38)$$

Where, I_{sleep} and V_{sleep} are the sleep leakage current and the voltage drop across the sleep FET.

Table 3.5: Algorithm ST_size_evaluate

Step	Explanation
1	Maximum switching current of the block without ST is calculated using SPICE.
2	Using Eqn 3.38, assuming a fixed value for V_{sleep} ($FullVdd/2$) and setting I_{sleep} equal to the value in step 1; the initial sleep transistor W is calculated
3	The estimated W_{SF} (sleep transistor width) is used with SPICE to measure V_{sleep} and I_{sleep} a new width is calculated using the latest values as in Eqn 3.38
4	STC_i = Leakage with STs (most recent I_{sleep} from SPICE) / Leakage without any STs
5	$STPC_i$ = Propagation delay without STs / Propagation delay with STs (most recent delay value from SPICE)
6	Steps 3 to 5 are iteratively repeated until a user defined stopping criteria is met, this is the optimal W_{SF} value
7	In the iterative process when the stopping criteria is reached the STC_i and $STPC_i$ factors correspond to the optimal W_{SF} and are the optimal STC and STPC.

The decision to stop the iterative sleep transistor width sizing is based on the stopping criteria parameter (STOP_ST) given by Eqn 3.39, which is calculated in step 6 in Table 3.5.

$$STOP_{ST} = (((STC_i - STC_{i-1}) \times \alpha_{ST}) + (STPC_i - STPC_{i-1}) \times (1 - \alpha_{ST})) / 2 \quad (3.39)$$

where α_{ST} is a user provided weight to trade-off power and performance, it is set to 0.5 by default. STC_i and STC_{i-1} are the leakage power correction descriptors for the i^{th} and $(i+1)^{th}$ iterations, respectively. $STPC_i$ and $STPC_{i-1}$ are the performance correction descriptors for the i^{th} and $(i+1)^{th}$ iterations, respectively. The

iterations are stopped if and when the value of $STOP_{ST}$ becomes less than an user provided threshold value ξ_{ST} .

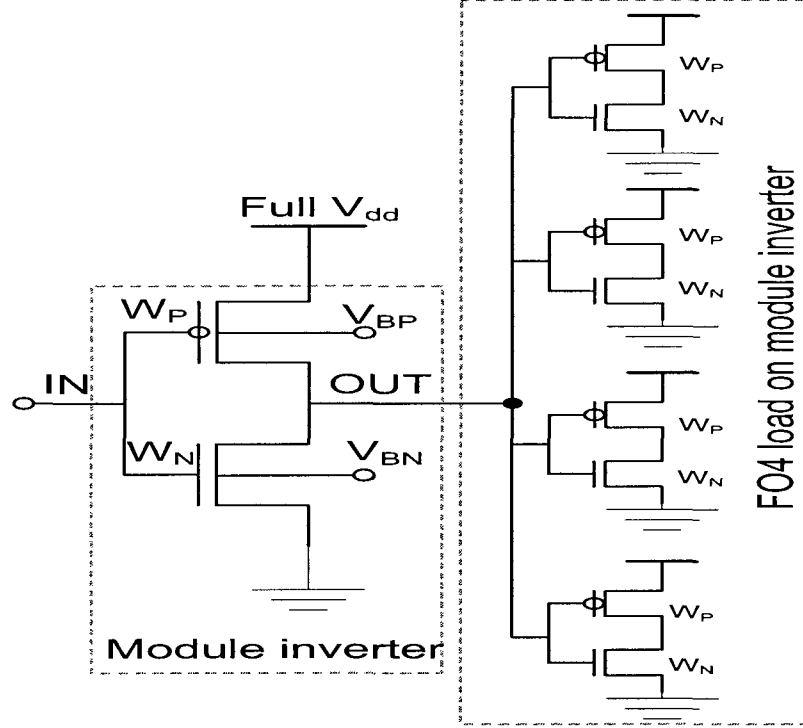


Figure 3.5: Experiment to obtain ABBC and ABBPC factors

Fig 3.5 shows the setup used to determine the impact of applying adaptive back body biasing to a module. To determine the optimal bias voltages, an algorithm (described in Table 3.6) was developed. The body voltage sweep ranges are $V_{dd} \pm 50\%$ of V_{dd} for PFETs and $GND \pm 50\%$ of V_{dd} for NFETs.

A figure of merit value (FOM_{ABB}) as defined in Eqn 3.40 is used to determine the optimal bias voltages.

$$FOM_{ABB} = (ABBPC_i / ABBC_i) \quad (3.40)$$

where, $ABBPC_i$ and $ABBC_i$ are the performance and leakage power correction descriptors values at the i^{th} step. The user can choose to bypass the algorithms de-

Table 3.6: Algorithm ABB_evaluate

Step	Explanation
1	Leakage current (sub-threshold) and propagation delay without ABB are calculated using SPICE
2	A complete two variable nested sweep of the body voltage in 10% of V_{dd} steps is started, recording the leakage and propagation delay value at each sweep point
3	$ABBC_i = \text{Leakage with ABB (most recent leakage current from SPICE)} / \text{Leakage without ABB}$
4	$ABBPC_i = \text{Propagation delay without ABB} / \text{Propagation delay with ABB (most recent delay value from SPICE)}$
5	Calculate FOM_ABB as in Eqn 3.40
6	Loop steps 2-4 for all combinations of bias voltages
7	The bias voltage combination which maximizes FOM_ABB is chosen as the optimal bias condition
8	$ABBC_i$ and $ABBPC_i$ factors correspond to the optimal bias conditions and are the optimal ABBC and ABBPC.

scribed in Tables 3.5 and 3.6 a custom sleep transistor size or bias voltages may be used respectively in the background simulations to calculate the descriptors.

To improve performance a design technique commonly employed is using dual threshold FETs i.e. low- V_t FETs along the critical path and nominal V_t FETs elsewhere. The increase in leakage due to low- V_t FETs is captured by Eqn 3.14 and the performance improvement is captured by the DVTC descriptor in Eqn 3.23.

Fig 3.6 shows the setup used to determine the impact of using dual threshold FETs. The nominal operating frequency i.e. with all nominal FETs of the test setup is measured. The nominal FETs are replaced with low- V_t FETs and the operating frequency is measured again. The increase in performance i.e. DVTC factor is then calculated as in Eqn 3.41.

$$DVTC = \frac{\text{Operating frequency with low-} V_t \text{ FETs}}{\text{Operating frequency with nominal FETs}} \quad (3.41)$$

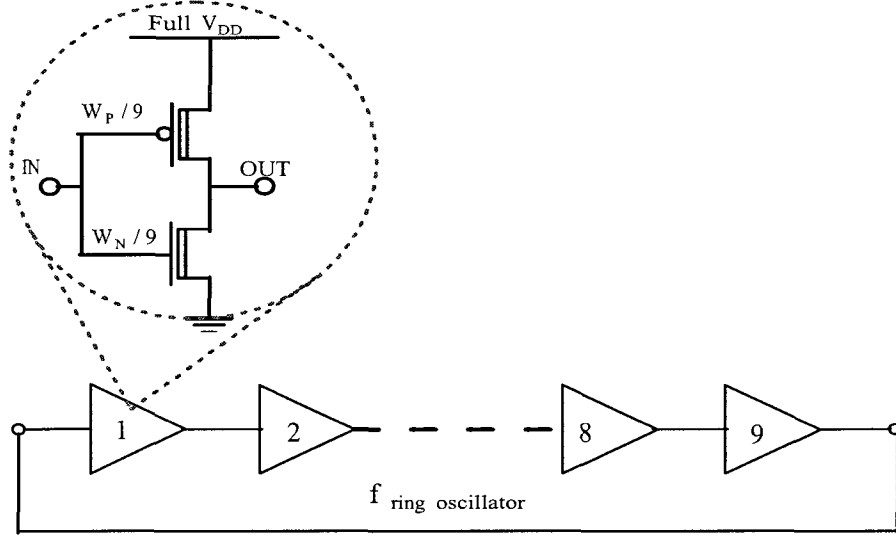


Figure 3.6: Experiment to obtain DVTC factor

3.5 Proposed Methodology for Rapid Early Design Space Exploration

3.5.1 Estimating Module Power and Performance

Eqns 3.3 though 3.1 represent the analytical design target prediction models to estimate a module's power and performance. Module descriptors are primary variables in these models. When an existing design is ported to a new technology or when different module-granular circuit-level design choices are applied, appropriate module descriptors are modified to reflect the changes; as shown in Fig 3.7. When a design choice is applied to the module which requires in-situ simulations, such as the choices described in section 3.3.3; simulations on the macro models are performed to calculate the corresponding descriptor values. Once all necessary descriptors are obtained, evaluating the analytical design target prediction models (Eqns 3.1 though 3.27) results in module power and performance estimates.

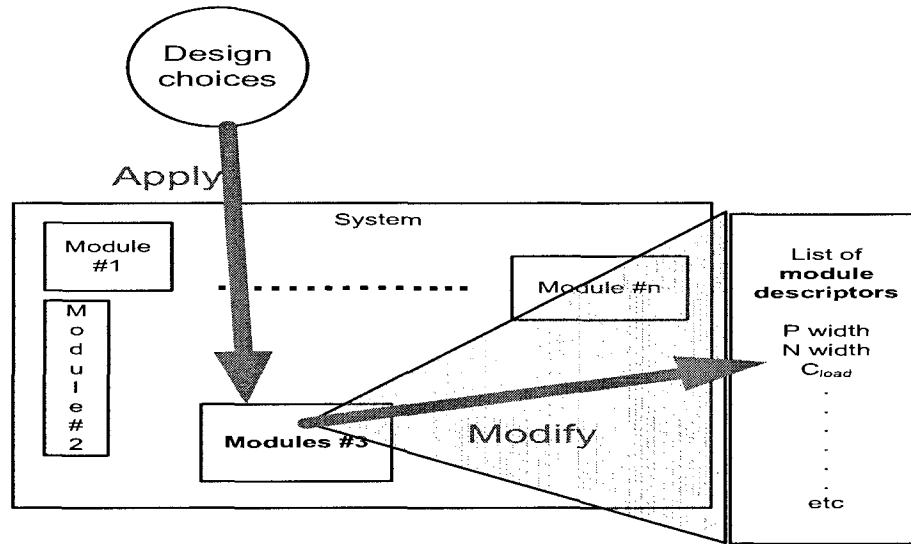


Figure 3.7: Application of a design choice to a module

3.5.2 System Design Target Prediction

For a system with multiple modules, design choices are independently assigned to each module in the system. With non-homogeneous design choice applied to modules, power and performance of all modules in the system are sequentially estimated. Once this is complete, the predicted system power is as the sum of all the individual module power estimates. All modules which contain portions of the system critical path are flagged as “critical modules”. The predicted system performance i.e. critical path delay, is the sum of all critical module delay estimates.

3.5.3 Design Space Exploration

Design space exploration involves evaluating system designs iteratively by varying circuit-level design choices applied to various modules in the system. The module granular circuit level design choice that can be applied to modules for design space exploration are described in detail in Appendix A. Fig 3.8 shows how the proposed

methodology and tool for design space exploration (EIDA) fits in a standard system design flow.

Given a system architecture optimized for computation efficiency or for functionality in a SoC, generating the optimal power, performance (i.e. operating frequency) system implementation scheme involves the following steps.

1. Generating the modular system model and descriptor vectors for all modules.
2. Initializing all non circuit-level design choice dependent descriptor values. They include all legacy design descriptors, target process technology dependent descriptors and analytical approximation coefficients.
3. Making module-granular circuit-level design choice assignments to all modules in the system.
4. Generating in-situ macro models and initiating SPICE simulations to measure the relevant parameters to calculate the corresponding descriptor values.
5. Predicting system power and performance using the analytical design target prediction models.

The analytical prediction models for system power and performance provide a path to link the physical level behavior to the high level system specifications, making the proposed approach for design space exploration more meaningful. Furthermore, using path ‘A’ in Fig 3.8, the architect may modified the system architecture based on EIDA’s power-performance trade-off and what-if analyses; such physical implementation driven architectural optimization leads to a correct-by-design system architecture. Thus, given a system architecture, early design phase power-performance optimization as in Fig 3.8 improves design convergence and help meet time-to-market requirement.

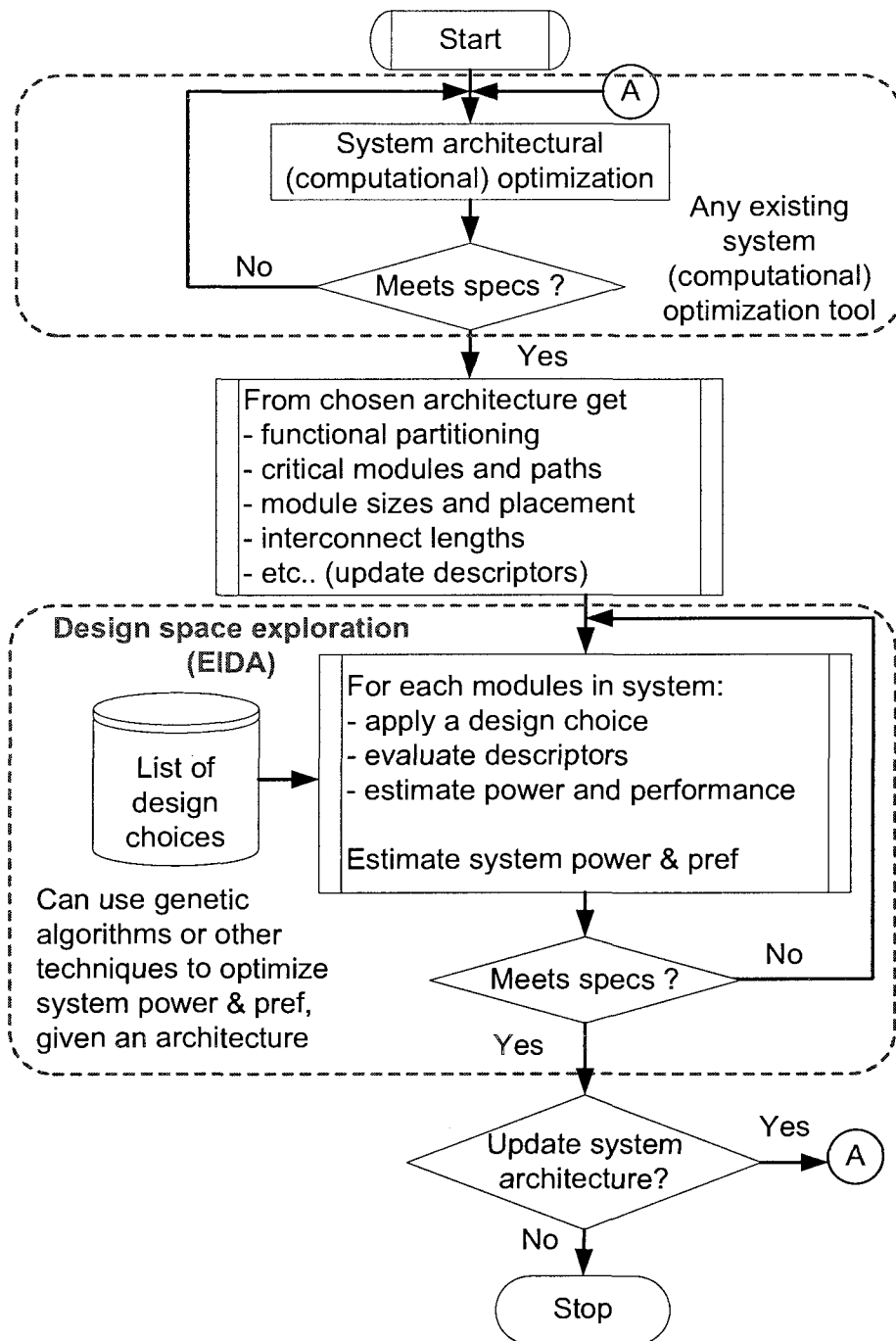


Figure 3.8: Rapid early design space exploration flow chart

3.5.4 Evolutionary Algorithms for Design Space Exploration

An evolutionary algorithms (EA) is a search technique that follows natural selection and survival of the fittest phenomenon observed in the biological world. EAs are unlike traditional optimization techniques by searching, at each iteration, within a collection of potential solutions called “population” and not from a single solution. At the end of every iteration a competitive selection process “weeds out” unfit solutions from the population so that in the next iteration these solutions do not generate more unfit solutions. The solutions with high fitness that remain in the population, they then mutate or recombine with each other to generate additional solution. Recombination and mutation are used to generate new solutions that are biased towards the fittest solutions (elite solutions) in the current population. EAs can be applied to many types of problems and it is particularly suited for multi objective optimization such as power performance optimization in modern VLSI designs.

In [63] an evolutionary algorithm based design space exploration was demonstrated on a larger VLIW-microprocessor based platform. It was shown that an evolutionary algorithm based design space exploration is immune to increase in system size and maintained high levels of efficiency. The proposed methodology of incorporating an evolutionary algorithm for design space exploration and the use of analytical models for design target prediction (design fitness estimation) result in high quality Pareto fronts suitable for performing power performance tradeoffs. The modeling of a system as a collection of modules (or sub-design) structurally lends itself very well to be represented as a chromosome, which is an integral part of any EA. Also by modularizing a system and independently characterizing the individual modules, permits modeling and analyzing large designs with large design spaces without significant disproportionate increase in system modeling efforts.

Chapter 4

Proposed Design Space Exploration Methodology: Experimental Setup

4.1 Overview

The overall goal of the proposed design framework is to enable designers to perform sufficiently accurate early design space exploration to improve design convergence and meeting time-to-market schedule. Validation of the proposed method for design space exploration is demonstrated through a technology node migration experiment. This experiment generates normalized predicted power vs performance charts, while applying standard circuit-level design techniques to modularized benchmark circuits. Following methodology validation, analytical system design target prediction model accuracy is verified against SPICE simulation results. Finally, the scalability of the proposed approach is shown by applying a Pareto-front analysis to two ISCAS89 benchmark circuits and an industrial microprocessor based design.

4.2 Experiments in Technology Node Migration

Technology node migrations of ISCAS85 [64,65] circuits C5315, C6288 and C7552 [66] and ISCAS89 [67,68] circuits S132007, S15850, S38417, S38584 and S9234 from a 180 nm process to a 130 nm process technology [69–71] are performed. Table 4.1 list the characteristics of the ISCAS benchmark circuits used in this experiment.

Table 4.1: Benchmark circuit details

- ISCAS85 c5315 : 178 inputs, 123 outputs, 2406 logic gates
- ISCAS85 c6288 : 32 inputs, 32 outputs, 2406 logic gates
- ISCAS85 c7552 : 207 inputs, 108 outputs, 3512 logic gates
- ISCAS89 s9234 : 36 inputs, 39 outputs, 211 DFF, 5597 logic gates
- ISCAS89 s13207 : 62 inputs, 152 outputs, 638 DFF, 7951 logic gates
- ISCAS89 s15850 : 77 inputs, 150 outputs, 534 DFF, 9772 logic gates
- ISCAS89 s38584 : 38 inputs, 304 outputs, 1426 DFF, 19253 logic gates
- ISCAS89 s38417 : 28 inputs, 106 outputs, 1636 DFF, 22179 logic gates

Structural Verilog description of these circuits were mapped to gates from a 180 nm standard logic gate library from [72] and the corresponding transistor level SPICE netlists were generated. The circuits were each randomly partitioned into four partitions, each partition with its input/output signal (nets) is considered as a module in a system of four interconnected modules; as shown in Fig 4.1. Critical paths within each module and within the whole circuit were determined using NanoSim [73]. These circuits were relatively small and with random partitioning the system critical path fell along and included all four partitions, but with varying degrees of contribution to the system critical path delay. The transistor level netlists for the benchmark circuits were used to determine each partition’s (module’s) total P and N FET widths. Process dependent descriptors were obtained from the 130 nm process specs.

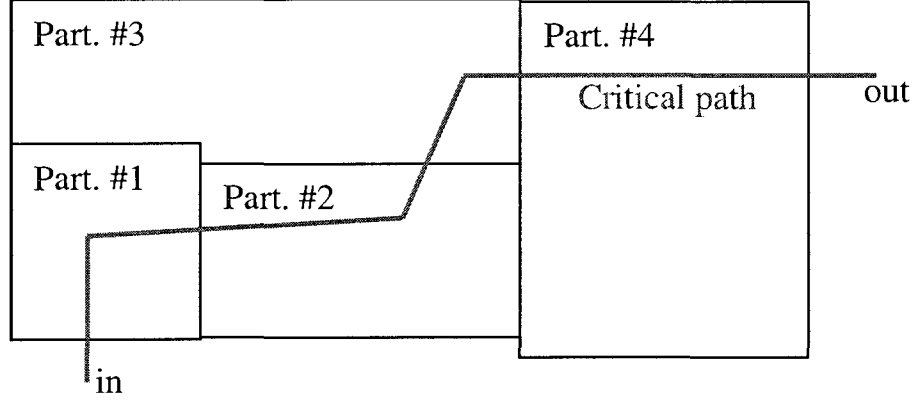


Figure 4.1: Benchmark circuit partitioning with critical path shown

Table 4.2: Assumed descriptor and coefficient values for 180 *nm* TSMC to 130 *nm* PTM technology porting

- s_{width} was set to 0.7
- sf was set to 2.5
- $RPSF$ was set at 1
- 10% V_{dd} droop i.e. $NADSP$ was set at 0.1
- ASF was set to 0.2
- HVR was set to 0.9
- $RCSF$ was set to 0.15 i.e. 15% interconnect degradation
- $BSUF$ was set to 1.3
- $TPFR$ was set to 0.75
- $DECAP_SENS$ was set to 0.05 V/nF (legacy data)
- Fitted $\alpha, \beta, \gamma, \delta$ and ϵ in Eqns 3.30 - 3.34 were 10, 8, 1.1, 1 and 2 respectively.
- Fitted X and Y in Eqn 3.35 were 0.2737 and 0.4305 respectively
- Interconnect length (TWL) was set at 1 m and divided among partitions according to the ratio of partition FET width to total FET width.

Table 4.2 shows the list of descriptors and analytical approximation coefficients and their values used for this experiment. Once all the descriptor values (in-situ simulation descriptors appropriately initialized to unity or zero) have been determined, analytical evaluations of Eqns 3.1 to 3.27 are performed, resulting in predicted power and performance numbers for the scaled circuits. The scaled circuits with no changes to any design choices (i.e. straight port) are used as the reference circuits to compare with scaled designs with a variety of different choices. The results of the design space exploration are normalized to the “straight port” designs for all circuits considered in the experiments.

4.2.1 Applying Module Granular Circuit Level Design Choice

Design space exploration is performed using a set of design techniques for either reducing power or improving performance. A design “assignment” is considered valid and complete when all modules in a system has been assigned a circuit-level design choice or a valid combination of design choices. Table 4.3 lists the design choices

Table 4.3: Module-granular circuit-level design choices

S.No	Design choice
1	No change to the original design
2	Lowering V_{dd} by 200 mV to reduce power consumption
3	Elevating V_{dd} by 200 mV to improve performance
4	Using Low V_t FETs in critical path to improve performance
5	Using sleep transistors to reduce leakage power
6	Adaptive body biasing to PFETs (FB) to improve performance
7	Adaptive body biasing PFETs (RB) to reduce leakage power

applied to the scaled design on a per module basis. The design choices are divided into two subsets, the first subset contained design choices for improving performance and second subset contained design choices for reducing power. Combinations of design techniques (recipes) within a subset are applied to the partitions of all three circuits. Design choices that are available for each module are listed in Table 4.3.

Lowering supply voltage to reduce power and using low- V_t FETs to improve critical path delay may not be a desirable combination. Incidentally, adaptive body biasing and sleep transistor together may not be a desirable combination since body bias has no effect when the supply is turned off. Design choices and their combinations are then assigned to the modules based on the module’s criticality; i.e. critical modules were assigned design choices from the high performance subset and non-critical modules were assigned design choices from the low power subset. All possible assignments under criticality constraints were generated and evaluated. This was done to mimic the intuitive design practice followed by design teams. When using sleep transistors, operating temperatures differ depending on whether a module is turned on and off [74]. This is accounted for in the in-situ simulations performed by changing temperature settings accordingly. Active operational temperature was set to 90° centigrade and inactive temperature was set to 25°.

4.3 Design Target Prediction Accuracy

Prediction accuracy is the foundation of the applicability of the proposed methodology. The feasibility of the proposed approach was established by the experiment described in Section 4.2. Design target prediction model accuracy with respect to SPICE has to be estimated in order to validate the applicability of the proposed methodology. SPICE simulation and measurement on the selected benchmark circuits were not possible due to large input vectors sizes. A test circuit in a reference technology is chosen for SPICE runs to verify prediction accuracy. The test circuit’s input vector size is smaller and more manageable for SPICE simulation to validate the applicability of the proposed methodology. Two experiments were conducted on the test circuit to ascertain prediction accuracy. In the first experiment the test circuit was successively ported from 180 nm to 130 nm to 90 nm to 65 nm technologies and

at each technology node the predicted power and performance and SPICE measurements are compared. In the second experiment the design space of the test circuit in a 32 nm process is explored to obtain a power and a performance centric solution. There solution were them implemented in SPICE and their power and performance were measured. The measured power and performance values were compared with the power and performance values predicted by the proposed prediction models.

4.3.1 Successive Design Porting From 180 nm to 65 nm Technologies

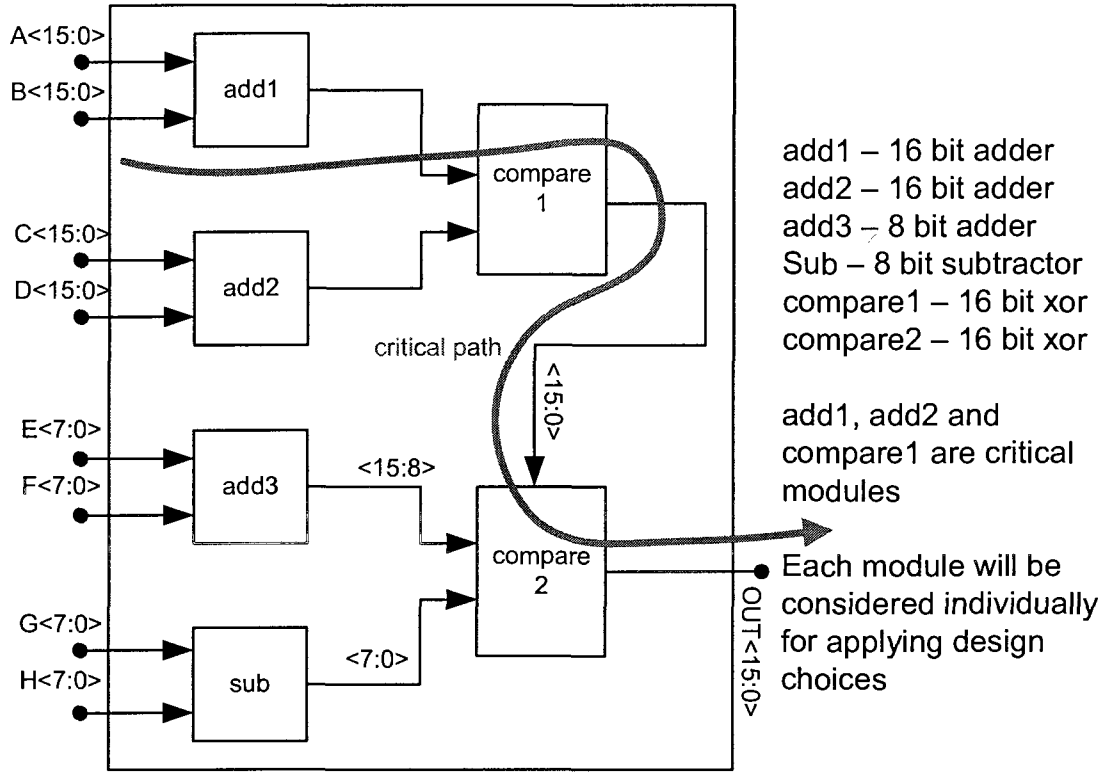


Figure 4.2: Test system to determine prediction accuracy

Table 4.4 lists the descriptor values and analytical approximation coefficients for 180 nm to 130 nm, 130 nm to 90 nm and 90 nm to 65 nm technologies used in this experiment.

Table 4.4: Assumed descriptor and coefficient values for 180 nm to 130 nm PTM & 130 nm to 90 nm PTM & 90 nm to 65 nm PTM & 65 nm to 32 nm PTM

(a) Common values of descriptors used

- s_{width} was set to 0.7
- sf was set to 2.5
- RPSF was set at 0.9
- 10% V_{dd} droop i.e. NADSP was set at 0.1
- ASF was set to 0.2
- HVR was set to 0.75
- RCSF was set to 0.1 i.e. 10% interconnect degradation
- BSUF was set to 1
- TPFR was set to 0.5
- DECAP_SENS was set to 0.05 V/nF
- Interconnect length (TWL) was set at 0.01 m and divided among modules according to the ratio of module FET width to total FET width.

(b) Technology port dependent descriptor values

Descriptor	180 to 130 nm	130 to 90 nm	90 to 65 nm	65 to 32 nm
α Eqn 3.30	10	8.5	6	5
β Eqn 3.31	8	7.25	6.25	5.25
γ Eqn 3.32	1.1	1.02	0.9	1
δ Eqn 3.33	1	0.94	0.65	0.65
ϵ Eqn 3.34	2	1.8	1.22	1.4
X Eqn 3.35	0.2737	0.3103	0.5262	0.5529
Y Eqn 3.35	0.4305	0.4625	0.5235	0.5448

Fig 4.2 shows the chosen test circuit, consisting of six modules, two 16-bit adders, one 8-bit adder, one 8-bit subtractor and two 16-bit comparators. The test circuit is implemented in 180 nm TSMC technology and its power consumption and performance (critical path delay) are measured by SPICE simulations. This implementation is considered as the legacy design and the measured power and delay form legacy de-

sign data. This legacy design is then manually scaled to 130, 90 and 65 nm PTM technologies. SPICE simulations on the scaled circuits are used to determine their power and performance. Simultaneously, the legacy design is modeled as described in Section 3.2 and ported successively to 130, 90 & 65 nm PTM technologies and the ported design's power and performance are estimated using the design target prediction model in Section 3.4. The SPICE measured power and performance is compared to the predicted power and performance.

4.3.2 Design Space Exploration of the Test Circuit in 32 nm Technology

Successive scaling experiment validates the prediction model accuracy as well as establishes the predicted result's conformity to comparable results in the literature. To further validated and demonstrate the applicability and accuracy of the proposed

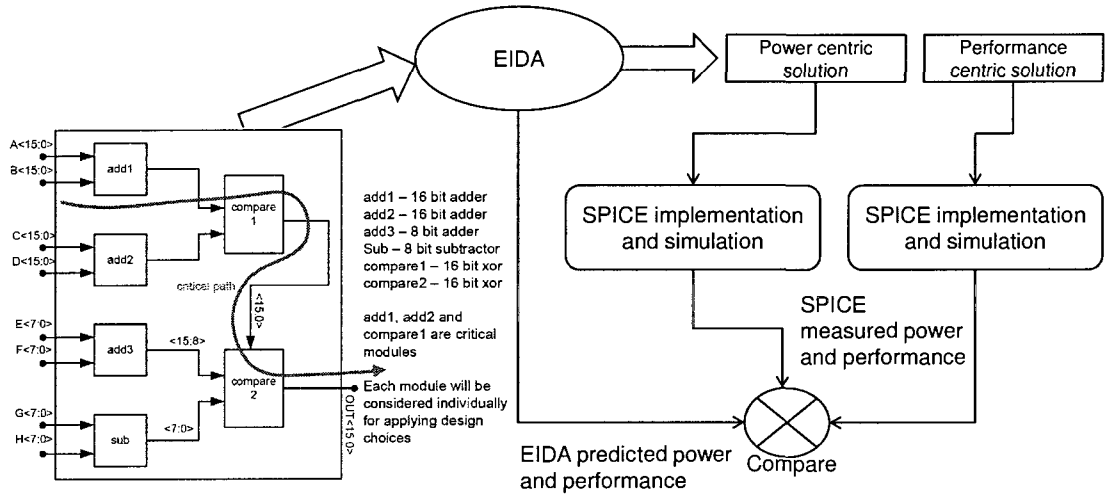


Figure 4.3: Procedure to compare EIDA and SPICE

methodology for design space exploration, the test circuit is implemented in a 65 nm PTM process and the ported from 65 nm to 32 nm PTM technology. Fig 4.3 illustrates this experiment. A design space exploration of the ported circuit in 32 nm

technology using the circuit level design choices listed in Table 4.3 was performed. Table 4.4 lists the descriptor values and analytical approximation coefficients for 65 nm to 32 nm technologies used in this experiment.

4.4 Evolutionary Algorithm Based Design Space Exploration

The validity of the proposed modeling methodology for design space exploration and design target prediction accuracy are addressed through technology node migration (Section 4.2) and design target prediction accuracy (Section 4.3) experiments respectively. In this section the evolutionary algorithm (EA) based design space exploration, pareto-front analysis [75] and the scalability of the modeling methodology are demonstrated on ISCAS89 s38584 and s38417 circuits, and a larger design based on an existing microprocessor design in a 65 nm CMOS technology [76].

The ISCAS89 circuits were randomly partitioned into four modules as described in Section 4.2. The circuits critical paths fell along all four partitions. The microprocessor based design is partitioned into 26 modules based on micro-architectural functionality. Partitioning is not a trivial task since in addition to functionality, floorplaning, power network integrity, and performance requirements all influence partitioning. Partitioning of a microprocessor is often a complex and manual process. In the process of partitioning the microprocessor, the optimization techniques discussed here played a minor role. Fig 4.4 shows the partitioning of a modern microprocessor similar to the partitioning scheme used for this experiment [77] [78] [79]. The detailed discussion of the partitioning process is beyond the scope of this report. The system critical path fell along 10 out of the 26 modules in the system.

To demonstrate design space exploration it is assumed that;

1. A hypothetical microprocessor (based on the existing design in a 65 nm PTM technology) is to be redesigned with no micro-architectural changes in 32 nm PTM process. Following the procedure outlined in Fig 3.8 and using the existing functional partitioning, all relevant legacy descriptor values are extracted. The design is then ported (migrated) to the 32 nm PTM process and the system power and performance (operating frequency) are predicted. Then an EA based design optimization using Pareto-analysis is performed on the ported design to complete the design space exploration.
2. The two ISCAS89 circuits (s38584 and s38417) are to be ported (migrated) from a 180 nm process to a 130 nm process technology ([69,70]) similar to the experiment described in Section 4.2. Then an EA based design optimization using Pareto-analysis is performed on the ported design to complete the design space exploration.

4.4.1 Design Migration

The 65 nm microprocessor based design is ported to a 32 nm PTM design [69,70]. Descriptor values for this experiment are shown in Table 4.5. Similarly, the ISCAS89 circuits are ported from 180 nm to 130 nm PTM. Descriptor values and analytical approximation coefficients for this experiment are shown in Table 4.2.

4.4.2 Pareto-Analysis Using Randomized Design Generation

Table 4.6 lists the module granular circuit level design choices for the ported 32 nm design. Design space exploration of the ported design is performed by assigning these design choices independently to all the 26 modules [80]. A design solution or recipe is considered “assigned” and valid when all twenty six modules have been given

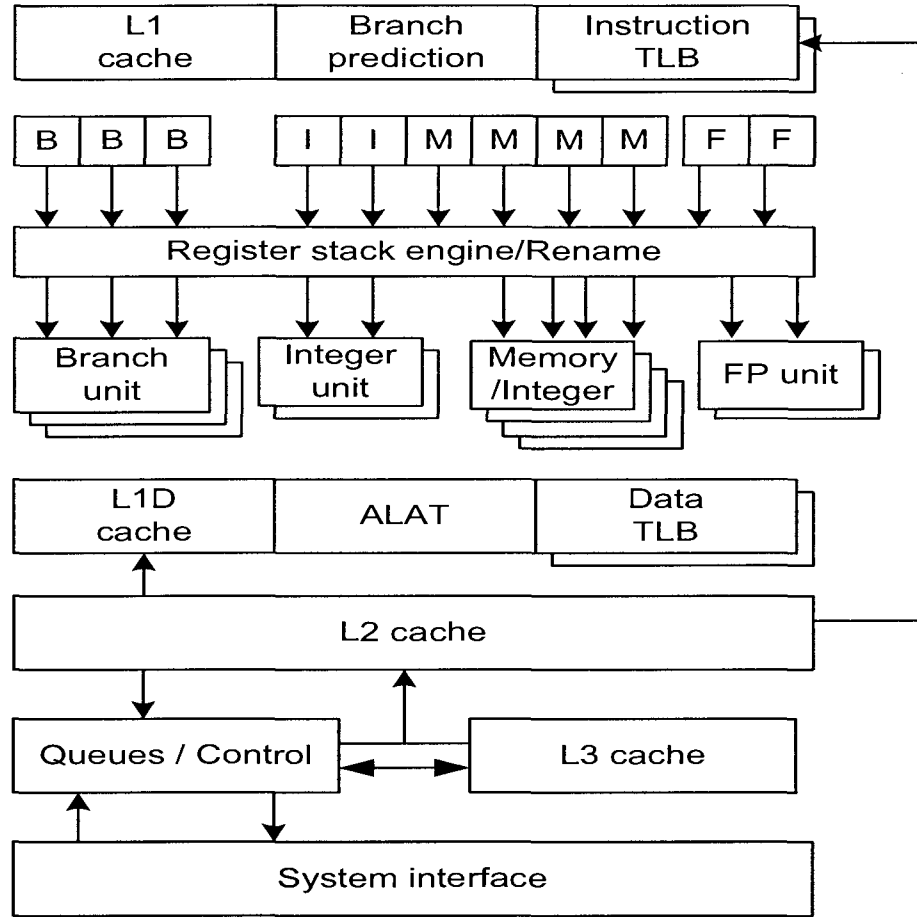


Figure 4.4: Block diagram of a modern microprocessor. B/I/M/FP: branch/ integer/ memory/ floating point units; ALAT: advanced load address table; TLB: translation look-aside buffer

one of the thirteen design choices from Table 4.6. The power and performance of a recipe can be estimated using the design target prediction model.

The number of possible recipes that are possible when thirteen design choices are independently assigned to twenty six modules is very large (i.e. $26! \times 13! = 2.51130432 \times 10^{36}$). Given such a large design space, it is not possible to evaluate all the recipes. Moreover not all recipes may be meaningful. However, design space exploration involves generating recipes for evaluation and choosing an optimal solution within the generated recipes. Therefore, two algorithms, simple randomizer and

Table 4.5: Assumed descriptor and coefficient values for 65 nm to 32 nm PTM

- s_{width} was set to 0.7
- sf was set to 2.0
- RPSF was set at 0.9
- 10% V_{dd} droop i.e. NADSP was set at 0.1
- ASF was set to 0.2
- HVR was set to 0.75
- RCSF was set to 0.1 i.e. 10% interconnect degradation
- BSUF was set to 1
- TPFR was set to 0.5
- DECAP_SENS was set to 0.05 V/nF
- Fitted $\alpha, \beta, \gamma, \delta$ and ϵ in Eqns 3.30 - 3.34 were 5, 5.25, 1, 0.65 and 1.4 respectively.
- Fitted X and Y in Eqn 3.35 were 0.5529 and 0.5448 respectively
- Interconnect length (TWL) was set at 10 m and divided among modules according to the ratio of module FET width to total FET width.

complete randomizer outlined in Fig 4.5a) and b) respectively to generate recipes for design space exploration were developed. These algorithms both utilize twelve standard design solutions called “seed recipes” in Table 4.7 to initialize recipe generation.

The simple randomizer algorithm chooses a seed recipe and generates a coin-flip based random bit for each of the twenty six modules. When the random bit corresponding to a module is ‘1’ then that module is marked for modification. When all the twenty six random bits have been generated, one of the thirteen design choices from Table 4.6 is randomly selected and applied to the modules marked for modification. The complete randomizer algorithm uses the same random bit generation technique to

Table 4.6: Valid available additional design choices

No.	Design choice
1	No additional design choices applied
2	Reduce Vdd by 100 mV
3	Reduce Vdd by 100 mV & apply sleep transistor for power gating
4	Reduce Vdd by 100 mV & ABB RB for critical path transistors
5	Increase Vdd by 100 mV
6	Increase Vdd by 100 mV & low V_t transistors in critical path
7	Increase Vdd by 100 mV & ABB FB for critical path transistors
8	Increase Vdd by 100 mV, Low V_t transistors & ABB FB in critical path transistors
9	Low V_t transistors in critical path
10	Low V_t transistors & ABB FB in critical path transistors
11	ABB FB in critical path transistors
12	ABB RB in critical path transistors
13	Inserting sleep transistors for power gating

Table 4.7: Seed recipes for pareto-front analysis

No.	Design choice
1	All non critical modules power gated with sleep transistors
2	All non critical module critical path transistors applied ABB-RB
3	All critical module critical path transistors made low- V_t
4	All critical module critical path transistors applied ABB-FB
5	All critical module critical path transistors applied ABB-FB and made low- V_t
6	All critical modules applied 100 mV higher VDD
7	All critical module critical path transistors made low- V_t and to critical modules applied 100 mV higher VDD
8	All critical module critical path transistors applied ABB-FB and critical modules applied 100 mV higher VDD
9	All critical module applied 100 mV higher VDD, critical path transistors made low- V_t and applied ABB-FB
10	All non critical modules applied 100 mV lower VDD
11	All non critical modules applied 100 mV lower VDD and module transistors applied ABB-RB
12	All non critical modules applied 100 mV lower VDD and power gated with sleep transistors

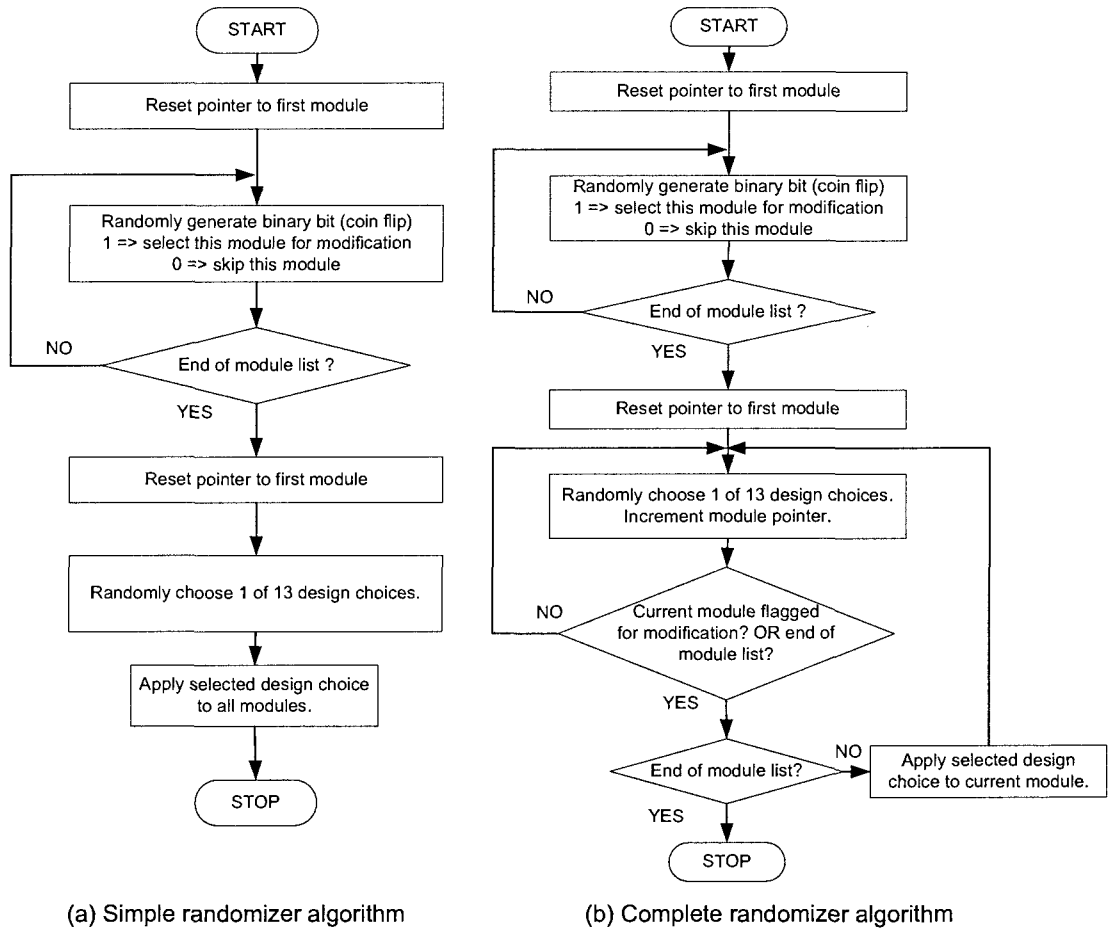


Figure 4.5: a) Simple randomizer algorithm b) Complete randomizer algorithm

mark modules for modification. However instead of choosing one design choice to be applied to all modules marked for modification, the complete randomizer algorithm randomly chooses one of the thirteen design choices from Table 4.6 individually for each module marked for modification. Starting with the initial twelve seeds, a total of two hundred and thirty recipes were generated. The power and performance of the generated recipes were estimated using the design target prediction models and are normalized to the straight ported values.

4.4.3 Pareto-Analysis Using EA Based Design Generation

Randomizer algorithm based design space exploration yielded a sparse pareto-front, Fig 5.11. Moreover the randomizer algorithms are not capable of tracking designs which optimize both power and performance. An evolutionary algorithm (EA) based design optimization using pareto-analysis of the ported design will yield a well populated and diverse pareto-front. In addition to improving the pareto-front, EAs are well suited to track solutions that optimize multiple criteria and evolve them to generate subsequent solutions. The original design in 65 nm is ported to 32 nm as described in Section 4.4.1. Table 4.6 lists the module-granular circuit-level design choices that are used for design space exploration.

4.4.3.1 Chromosome Definition

Design space exploration using multi-criteria evolutionary algorithms ([81]) requires a “chromosome” mapping scheme to represent the system to be optimized. The microprocessor based design consists of 26 modules, where each module can be independently assigned one of twelve design choices from Table 4.6. Therefore the design’s chromosome mapping is defined as a vector of length 26, where each vector element, or gene, can have an integer value 0 through 12. For the ISCAS89 circuits the chromosome mapping is defined as a vector of length 4, where each vector element, or gene, can have an integer value 0 through 12. Fig 4.6(a) shows the chromosome (for the microprocessor based design) with 26 elements, one for each module in the system with possible gene values for module 1 expanded.

4.4.3.2 Chromosome Fitness Estimation

Every chromosome has fitness values associated with it, which in this experiment corresponds to a vector of size two (power,performance) i.e. total power consumption and performance of the system with design choices applied as represented by the

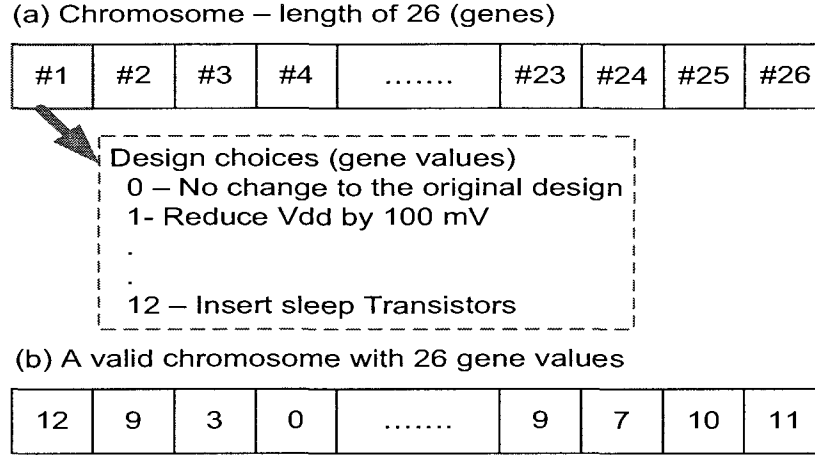


Figure 4.6: (a) Chromosome for evolutionary algorithm based pareto analysis. A complete list of all design choices is listed in Table 4.6 (b) A valid chromosome (12,9,3,0,.....,9,7,10,11)

chromosome itself. Fig 4.6(b) shows a valid chromosome with design choices applied to all the modules in the design. The fitness of a given chromosome is evaluated using the analytical prediction models. Power and performance for all modules in the system are estimated considering its corresponding gene value, i.e. design choice. Then from them the system power and performance are calculated which forms the chromosome fitness vector. The starting point for design space exploration is the ported design to the new process with no additional design choices applied, represented by the straight-ported chromosome (0,0) for the microprocessor based design and (0,0,0,0) for the ISCAS89 circuits. All power and performance results presented here are normalized to the straight-ported chromosome fitness values respectively.

4.4.3.3 Chromosome Encoding, Generation and Optimization

Binary encoding of the chromosome is used to transform the chromosome into a string of 1s and 0s. Individual gene value can be an integer between 0 and 12, therefore a 4-bit binary encoding for each gene is used. Each chromosome will be transformed

into a (26×4) 104-bit long binary number. For example, the chromosome in Fig 4.6(b) will be represented as a binary string:

$$\begin{array}{cccccccc} 12 & 9 & 3 & 0 & & 9 & 7 & 10 & 11 \\ \underbrace{1100} & \underbrace{1001} & \underbrace{0011} & \underbrace{0000} & \dots\dots & \underbrace{1001} & \underbrace{0111} & \underbrace{1010} & \underbrace{1011} \end{array}$$

The parameters used in the evolutionary algorithm based design space exploration are summarized in Table 4.8.

Table 4.8: Summary of evolutionary algorithm based pareto analysis

Characteristic	Description
Population size	100
Crossover	Uniform crossover
Selection method	Two random chromosomes
Mutation probability	1%
Replacement policy	Dominant child replaces one dominated chromosome in the current population

The initial population is selected in such a way that none of the chromosomes in the initial population dominate any other chromosome in the population. After the initial population is chosen, the iterative process to optimize the design is carried out. At each iteration two new chromosomes are generated by uniform crossover of two randomly selected chromosomes in the current population. Invalid chromosomes are discarded. A generated valid chromosome's fitness is evaluated and compared to all existing chromosome in the current population. If any chromosome in the current population is dominated by the generated one, then the dominated chromosome is replaced with the generated chromosome. Only one replacement is allowed per iteration to maintain a constant population size. If any chromosome in the current population dominates the generated chromosome then the generated one is discarded.

Chromosome domination is defined in Eqn 4.5. Consider a multi-criteria EA optimization problem with, n optimization criteria, chromosome length of m and population size p . Let C , S_i and F_i be the criteria, a generic chromosome and fitness vectors respectively.

$$C \in \mathbb{R}^n \text{ and } C = [C_1, C_2, \dots, C_n]^T \quad (4.1)$$

$$S_i \in \mathbb{R}^m \text{ and } S_i = [S_{i1}, S_{i2}, \dots, S_{im}]^T \quad (4.2)$$

$$F_i \in \mathbb{R}^n \text{ and } F_i = [F_{i1}, F_{i2}, \dots, F_{in}]^T \quad (4.3)$$

The criteria vector can be grouped as,

$$C = [C_1, C_2, \dots, C_{\vartheta}, C_{\vartheta+1}, C_{\vartheta+2}, \dots, C_{\lambda}, C_{\lambda+1}, C_{\lambda+2}, \dots, C_n]^T \quad (4.4)$$

Where, criteria 1 to ϑ are minimization, $\vartheta + 1$ to λ are maximization and $\lambda + 1$ to n are criteria considered don't care for problem relaxation purposes (optional).

Now, let S_1, S_2 and F_1, F_2 be two chromosomes and their corresponding fitness vectors. Chromosome S_1 dominates S_2 if and only if the following is true;

$$\begin{aligned} F_{1j} &< F_{2j} \quad \forall \quad j : 1 \dots \vartheta \\ F_{1k} &> F_{2k} \quad \forall \quad k : (\vartheta + 1) \dots \lambda \end{aligned} \quad (4.5)$$

The results of the experiments to validate the proposed modeling methodology for design space exploration, to establish the design target prediction accuracy and, to demonstrate the scalability of EA based design space exploration are presented next in Chapter 5.

Chapter 5

Experimental Results, Discussion of the Results and Future Work

5.1 Results of Experiments In Technology Node Migration

The eight chosen benchmark circuit, each partitioned into four modules were ported from 180 nm TSMC to 130 nm PTM process technology, designated as “straight port” design. The ported circuit’s power and performance were estimated using the analytical design target prediction model. For each circuit an exhaustive module granular design choice assignment list (generated design) is generated and their power and performance are estimated using the analytical design target prediction models. Figs 5.1 to 5.8 show the power vs performance curve for all generated designs normalized to the “straight port” design for each of the eight chosen benchmark circuits. In Figs 5.1 to 5.8 design labeled A & B are performance centric and design labeled C,D & E are power centric. The power and performance centric assignment details for each chosen benchmark circuit are listed in Tables 5.2 to 5.8, where assignments subscripted “solution” are the chosen power/performance centric solution. Table 5.1 summarizes results of the technology migration experiment followed by the normalized power performance plots for all the circuits. For each circuit considered, system

power and performance impact for the power and performance centric assignment solutions are shown in Table 5.1.

Table 5.1: Technology mode migration results

Circuit name	Assignment name	%pwr impact	%pref impact
C5315	A	2.35	11.25
	D	-32	-9
C6288	A	2.43	10.31
	E	-17	-4.3
C7552	A	3.85	15.3
	E	-32	-1.3
S38584	B	2	9
	C	-16	-3
S13207	B	2	11
	D	-23	-3
S38417	B	3	17
	C	-7	< -1
S15850	B	5	8
	D	-8	-6
S9234	B	4	6
	C	-4	-2

-----*Intentionally left blank.*-----

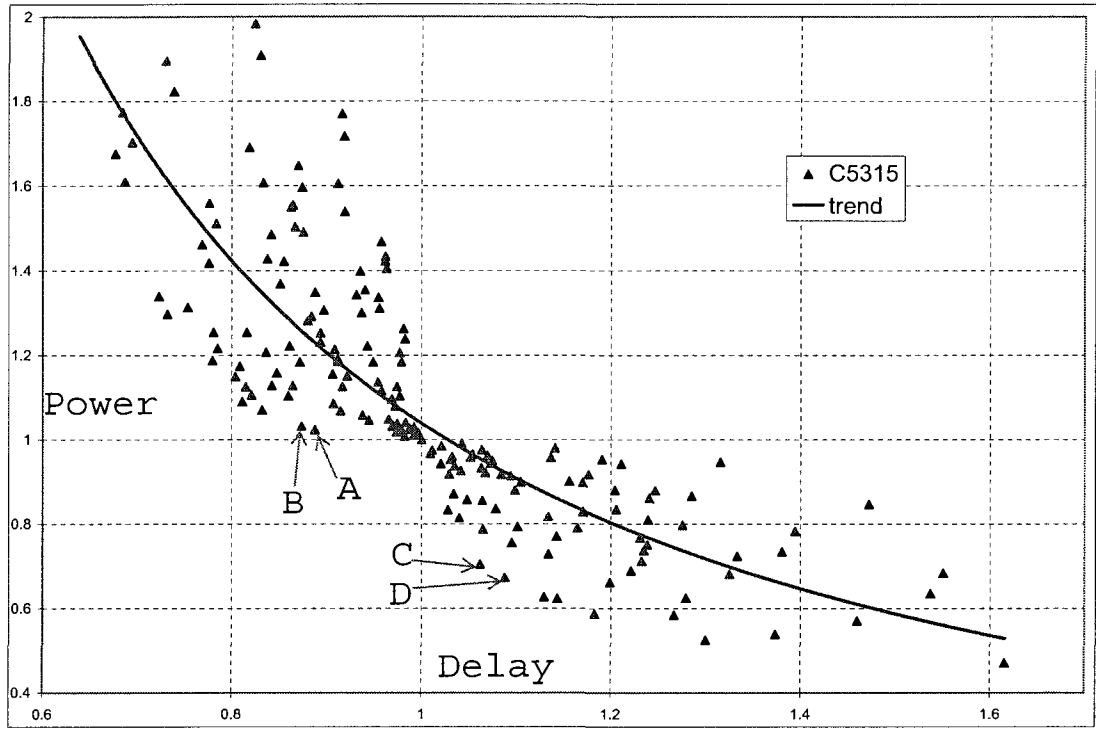


Figure 5.1: C5315 design choices

Table 5.2: Circuit C5315 migration results

Assignment	Module 1	Module 2	Module 3	Module 4
<i>A solution</i>	low- V_t	low- V_t	low- V_t	low- V_t
B	low- V_t & ABB-FB	low- V_t & ABB-FB	low- V_t & ABB-FB	low- V_t & ABB-FB
C	low- V_{dd}	low- V_{dd}	none	none
<i>D solution</i>	low- V_{dd} & ST	low- V_{dd} & ST	none	none

For circuit C5315 in Fig 5.1, performance centric assignment A offers a 2.35% increase in power for a 11.25% improvement on performance and power centric assignment D offers a 32% reduction in power for a 9% performance penalty.

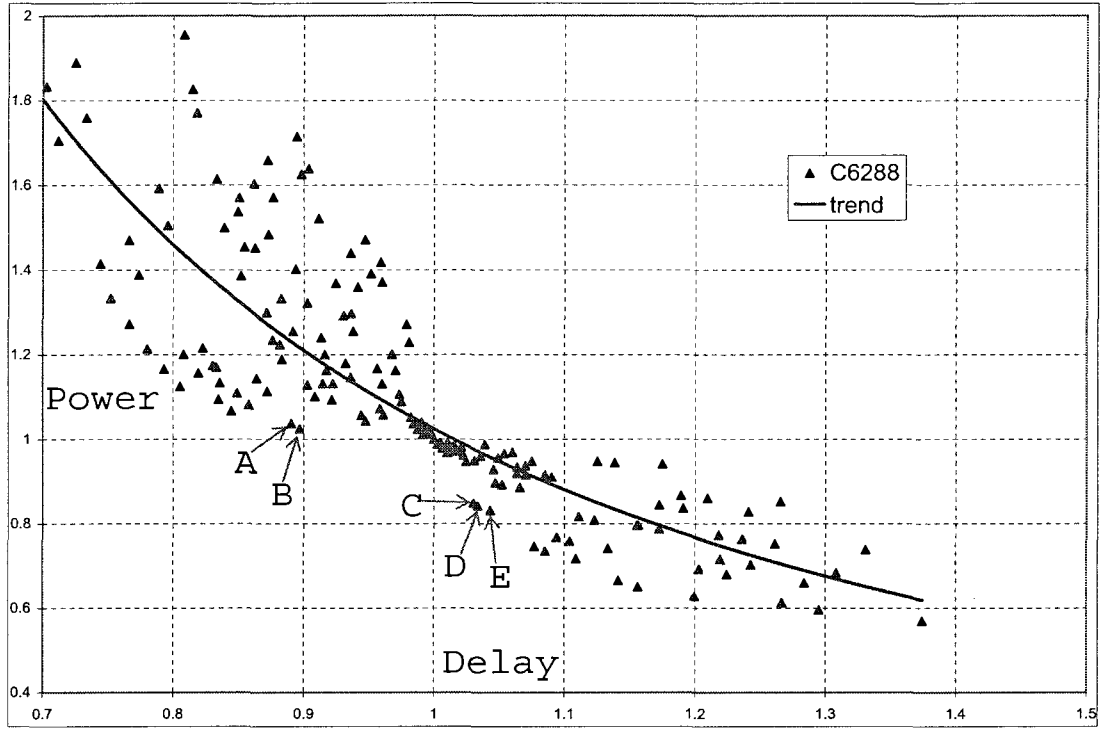


Figure 5.2: C6288 design choices

Table 5.3: Circuit C6288 migration results

Assignment	Module 1	Module 2	Module 3	Module 4
A <i>solution</i>	low- V_t	none	none	none
B	low- V_t & ABB-FB	none	none	none
C	low- V_{dd}	none	none	none
D	low- V_{dd} & ABB-RB	none	none	none
E <i>solution</i>	low- V_{dd} & ST	none	none	none

For circuit C6288 in Fig 5.2 performance centric assignment A offers a 2.43% increase in power for a 10.31% performance improvement and power centric assignment E offers a 17% reduction in power for a 4.3% performance penalty.

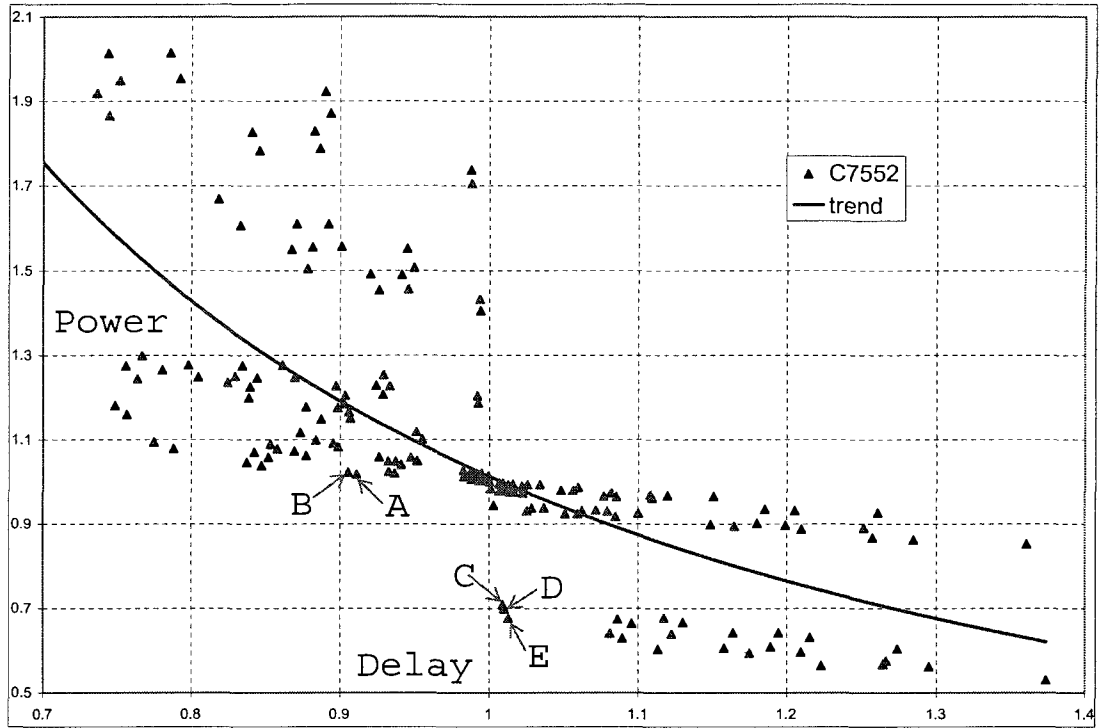


Figure 5.3: C7552 design choices

Table 5.4: Circuit C7552 migration results

Assignment	Module 1	Module 2	Module 3	Module 4
A <i>solution</i>	low- V_t	none	none	low- V_t
B	low- V_t & ABB-FB	none	none	low- V_t & ABB-FB
C	none	low- V_{dd}	none	none
D	none	low- V_{dd} & ABB-RB	none	none
E <i>solution</i>	none	low- V_{dd} & ST	none	none

For circuit C7552 in Fig 5.3 performance centric assignment A offers a 3.85% increase in power for a 15.3% performance improvement and power centric assignment E offers a 32% reduction in power for a 1.3% performance penalty.

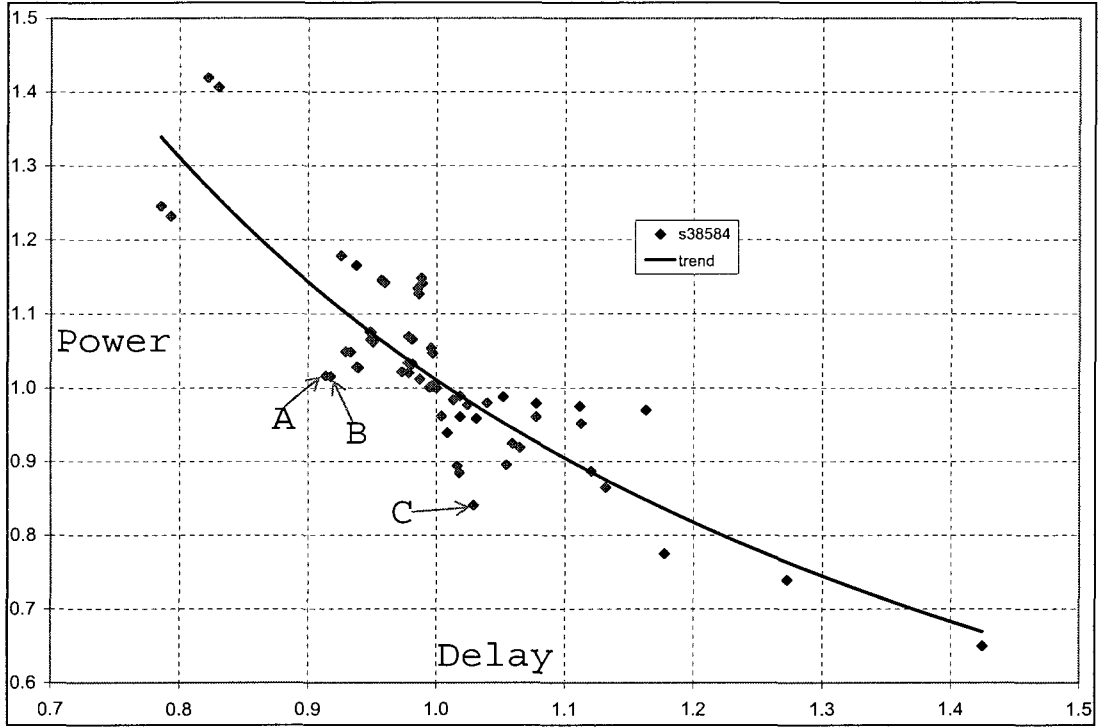


Figure 5.4: S38584 design choices

Table 5.5: Circuit S38584 migration results

Assignment	Module 1	Module 2	Module 3	Module 4
A	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB
B <i>solution</i>	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t
C <i>solution</i>	none	none	none	ST

For circuit S38584 in Fig 5.4, performance centric assignment B offered a 9% performance improvement for a 2% increase in power and power centric assignment C offered a 16% reduction in power with a 3% performance penalty. Module 4 in circuit S38584 contributed the least to the critical path delay, therefore design choice that lower power when applied to module 4 have a significant impact on power with minimal impact on performance.

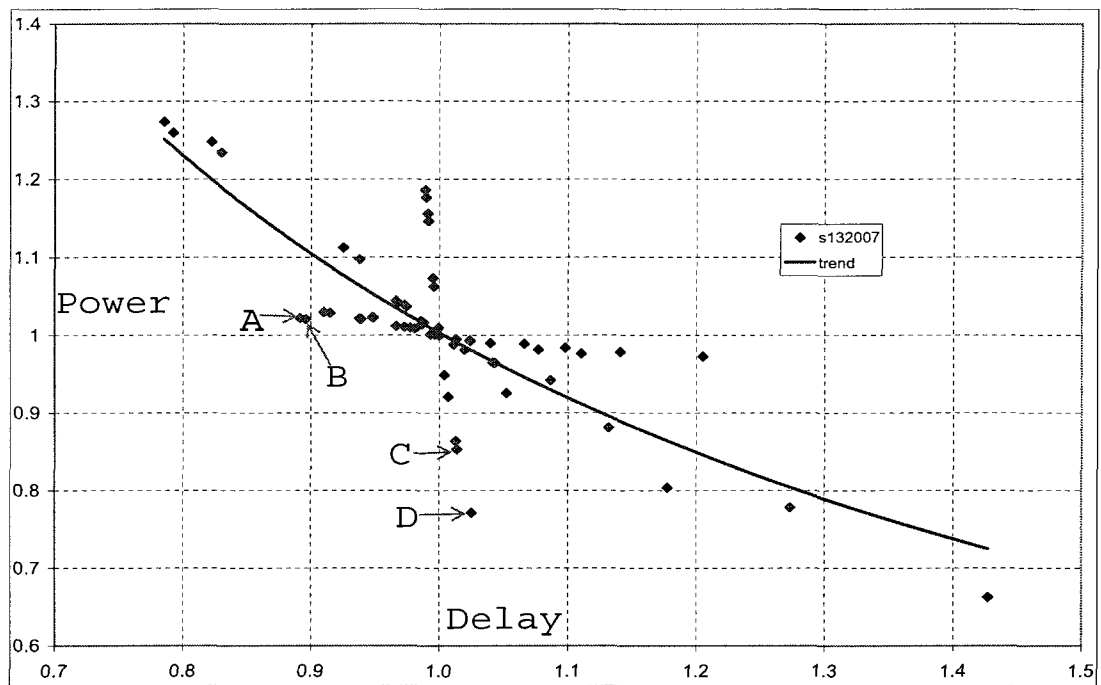


Figure 5.5: S132007 design choices

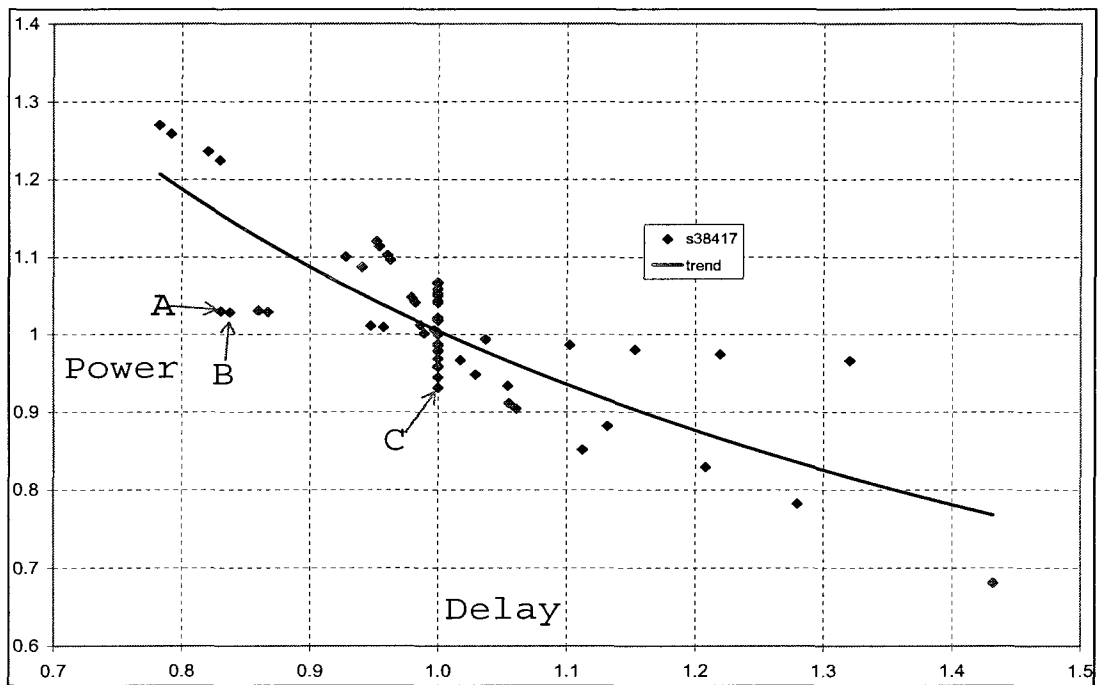


Figure 5.6: S38417 design choices

Table 5.6: Circuit S132007 migration results

Assignment	Module 1	Module 2	Module 3	Module 4
A	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB
B <i>solution</i>	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t
C	none	none	none	low- V_{dd}
D <i>solution</i>	none	none	none	ST

Table 5.7: Circuit S38417 migration results

Assignment	Module 1	Module 2	Module 3	Module 4
A	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB
B <i>solution</i>	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t
C <i>solution</i>	none	none	ST	none

Partition 4 in S132007 and partition 3 in S38417 contributed $< 3\%$ to the system critical path delay and approx 23% to the total system power. The manifestation of this underlying circuit condition can be seen in the power performance plots where assignments with design choices targeting module 4 in S132007 and module 3 in S38417 fall along a straight line with the power centric assignment dominating other assignments. For circuit S132007 in Fig 5.6 performance centric assignment B offers a 11% improvement in performance for a 2% increase in power and power centric assignment D offers a 23% power reduction for a 3% performance penalty. For circuit S38417 in Fig 5.4 performance centric assignment B offered a 17 % performance improvement for a 3% increase in power and power centric assignment C offered a 7% power reduction for no performance penalty.

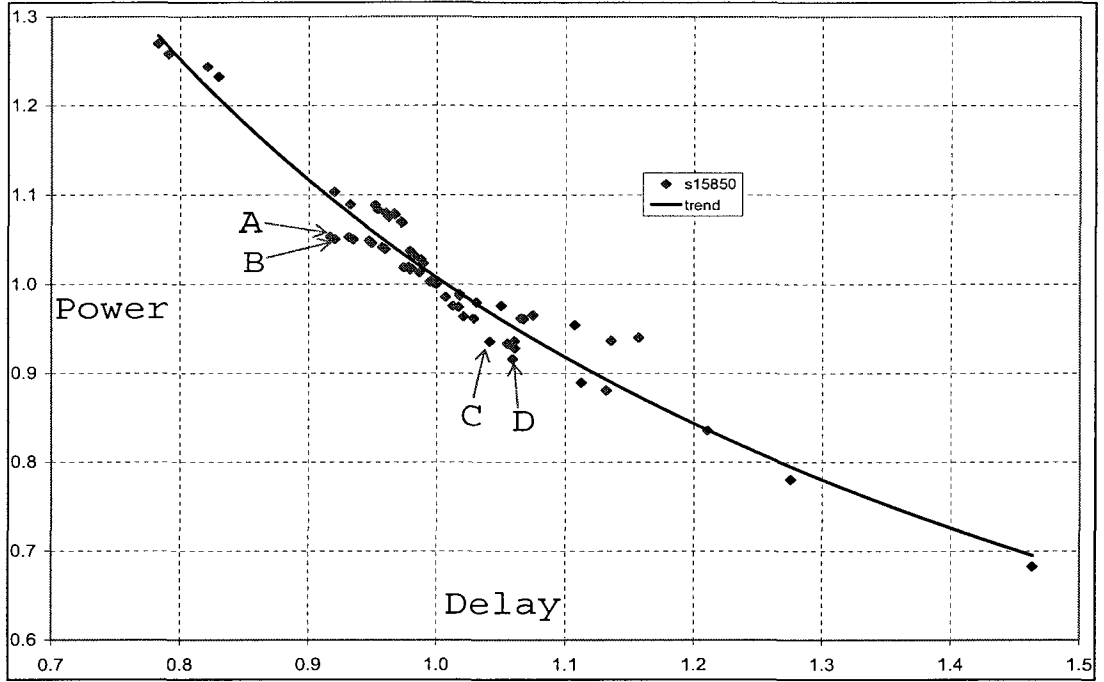


Figure 5.7: S15850 design choices

Table 5.8: Circuits S15850 & S9234 migration results

Assignment	Module1	Module2	Module3	Module4
S15850				
A	lo- V_t , hi- V_{dd} , ABB-FB	lo- V_t , hi- V_{dd} , ABB-FB	lo- V_t , hi- V_{dd} , ABB-FB	lo- V_t , hi- V_{dd} , ABB-FB
B solution	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t
C	none	low- V_{dd}	none	none
D solution	none	ST	none	none
S9234				
A	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB	low- V_t , hi- V_{dd} & ABB-FB
B solution	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t	hi- V_{dd} & low- V_t
C solution	low- V_{dd}	low- V_{dd}	low- V_{dd}	low- V_{dd}

Circuits S15850 in Fig 5.7 and S9234 in Fig 5.8, were unique with all modules contributing equally to both system power and system critical path delay. As a

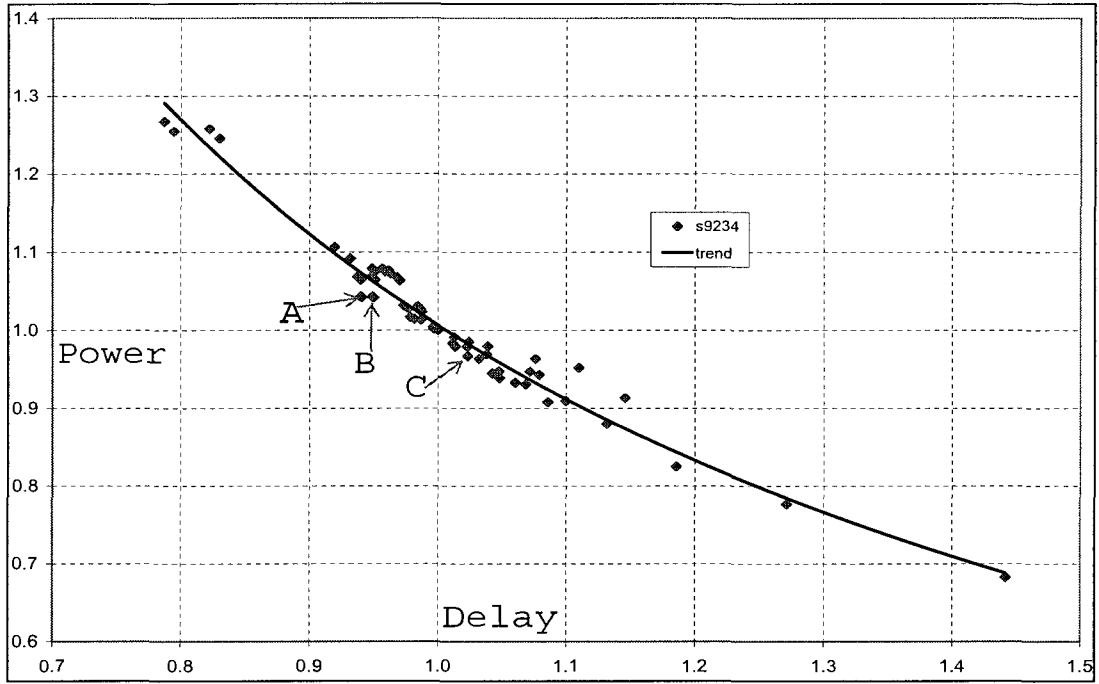


Figure 5.8: S9234 design choices

result these circuits had fewer opportunities for power performance trade-offs, this can be seen in the power-performance plots where the assignment spread is distinctly subdued with fewer assignments dominating other assignments. For circuit S15850 performance centric assignment A offered a 8% improvement in performance for a 5% increase in power and power centric assignment D offered 8% power savings for a 6% performance penalty. Similarly for circuit S9234 performance centric assignment A offered a 6% improvement in performance for a 4% increase in power and power centric assignment C offered 4% power savings for 2% performance penalty.

5.1.1 Technology Node Migration Experiment Observations

1. Normalized power vs performance plots showed expected trends with the application of known power-centric and performance-centric circuit-level design choice assignments.

2. Power vs performance plots for all but two circuits considered (S15850,S9234), exhibited trends leading to an assignment optimizing both system power and performance.
3. Circuits S15850 in Fig 5.7 and S9234 in Fig 5.8, were unique with all modules contributing equally to both system power and system critical path delay. As a result these circuits had fewer opportunities for power performance trade-offs, this can be seen in the power-performance plots where the assignment spread is distinctly subdued and fewer assignments dominating other assignments. The charts for these circuit expose the difficulty in optimizing both power and performance early in the design phase, thus avoiding pitfall redesigns.
4. In circuits S132007 and S38417, one module (#4 in S132007 and #3 in S38417) contributed less than 3% to the system critical path delay. This underlying circuit condition leads to a reduction in system power consumption with little impact on performance, which was observed in the power performance plots, Figs 5.5 and 5.6 respectively.
5. An industrial study on a 16-bit multiplier implemented in a 90 nm process, reported a 7X reduction in leakage power using sleep transistors compared to the active state [82]. Similar trends are observed here, with a 20% system wide activity factor the system power saving predicted here is between 16-32% (2.5-6X) on average and is comparable to results in [82].
6. An industrial study performed power measurements on an ALU in 130 nm process for an typical activity profile and reported a 9% and 15% reduction in power using ABB and ST respectively [83]. Power savings predictions with a 20% activity factor, using ABB for the circuits are 3-6% on average and are

similar to the reported savings in [83]. This further validates the proposed methodology.

5.2 Results of Design Target Prediction Accuracy

5.2.1 Results of Successive Design Porting From 180 nm to 65 nm Technologies

Fig 5.9 shows the technology scaling trends observed for power and performance when the test circuit is scaled from 180 nm to 130 nm to 90 nm to 65 nm technology. The technology scaling trend of power and performance exhibited by the EIDA results are consistent with the results shown for the reference ALU design in [84] and the results for porting from 180 nm TSMC to a 130 nm technology from [85].

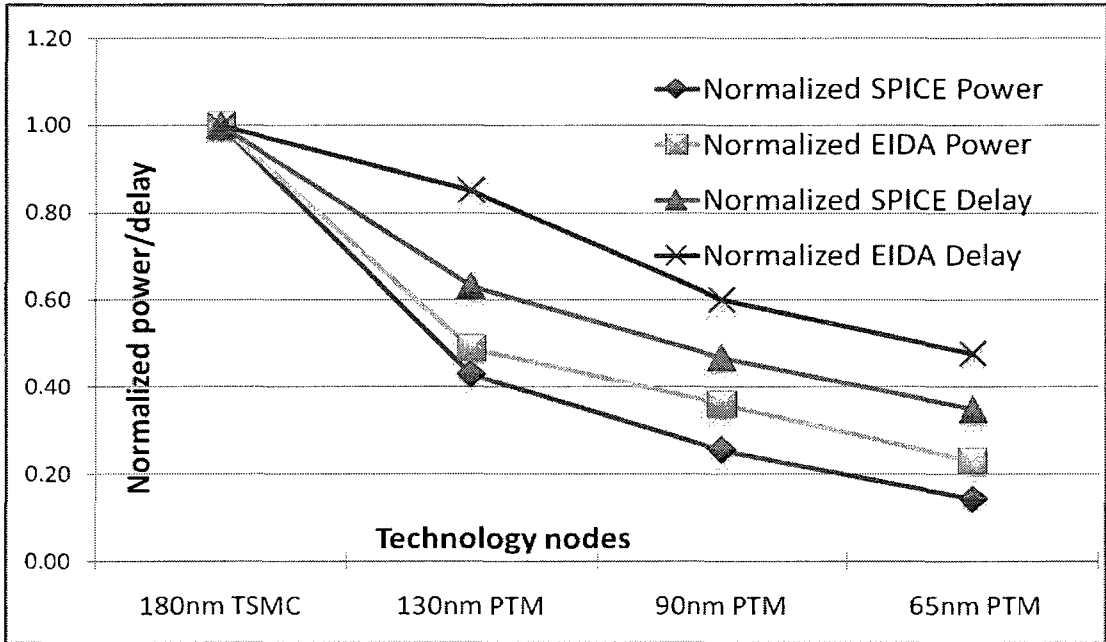


Figure 5.9: Observed technology scaling trends for power and performance

Fig 5.10 shows the observed power and performance prediction errors with respect to SPICE. Power simulations are performed using a set of input vectors that

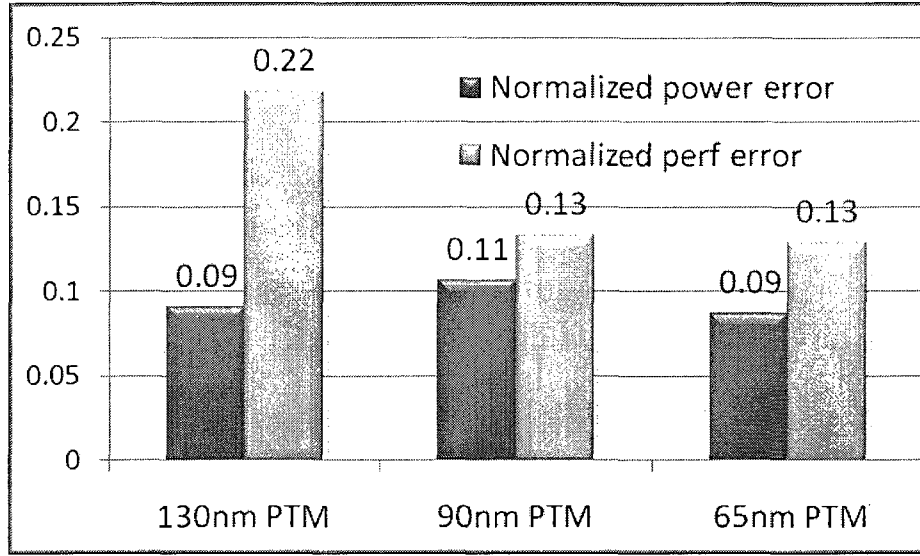


Figure 5.10: Observed prediction error with respect to SPICE

emulate a typical system switching activity of 20%. The prediction errors for power range from 9% to 11% with respect to SPICE. The errors in power prediction is well controlled given that some circuit aspects such as load capacitances may not be accurately modeled at the system level. The errors in performance prediction range from 13% to 22% with respect to SPICE. This is mainly due to the unavailability of detailed layout interconnect parasitic values included in the SPICE netlist. However the interconnect RC delay contribution is estimated and included in EIDA through the TPFR descriptor in Eqn 3.23. TPFR was set to 0.5 as shown in Table 4.4. The errors in performance prediction are reasonable given the nature of system level modeling. The accuracy of static timing analysis using traditional signal propagation was shown to be within 14% of SPICE in [86] in some cases. In terms of average errors, state of the art static timing tools from leading commercial EDA vendors report having typical error within 5% of SPICE [87] [88]. These tools operate on detailed transistor level netlist. Experiments on ISCAS circuits implemented in a 90 nm industrial process using a lumped capacitance model and the most commonly used Thevenin based

flow for timing analysis yielded errors between 10-15% (reported $\mu+\sigma$ error quantile of 7.5%) [89]. With this context, operating at the highest level of abstraction and with power and performance prediction errors in the range of 9 to 22% compared to SPICE results makes the usage of EIDA for high level design tradeoffs practical, especially for the early design phase when complete bottom up data is **not** available yet.

5.2.2 Results of Design Space Exploration of the Test Circuit in 32 nm Technology

Design space exploration by EIDA yields power and performance centric design assignment solutions. The design is manually scaled to 32 nm PTM and the module granular circuit level design choices generated by EIDA are applied. Then SPICE simulations on these circuits yield actual power and performance of the EIDA generated power and performance centric solutions. EIDA predicted power and performance for the power centric and performance centric design solutions are compared with SPICE simulation based measured power and performance. Table 5.9 tabulates the average error percentages between the predicted power and performance and SPICE measured values.

Table 5.9: EIDA and SPICE comparison

Measurement	EIDA	SPICE	% error
straight-ported design	1.7924 mW	1.6084 mW	-11.4%
straight-ported design	392 MHz	451 MHz	13.1%
power-centric design	1.0456 mW	0.937 mW	-11.5%
power-centric design	406 MHz	368 MHz	-10.3%
performance-centric design	1.5523 mW	1.375 mW	-12.9%
performance-centric design	442 MHz	513 MHz	13.8%

EIDA predictions for power and performance were within 14% of SPICE and consistent with the results of the successive scaling experiment. Early design exploration studies typically require a relative measure of the “goodness” of different

design solutions and absolute accuracy is not necessary. Therefore, along with the successive scaling experiment, design space exploration experiment completely validates the applicability and accuracy of the proposed approach for early rapid design space exploration.

5.3 Results of Evolutionary Algorithm Based Design Space Exploration

The straight ported power and performance predicted by the design target prediction models are **2.661492 watts** and **2.94 GHz**, respectively. Porting the microprocessor based design to 32 nm PTM in a straight manner (i.e. no circuit changes etc.) improves operating frequency by 30% and reduces total power consumption by 11%, compared to the legacy design.

5.3.1 Results of Pareto-Analysis Using Randomized Design Generation

Fig 5.11 shows the power vs performance plot of the straight ported, seed and generated recipes, normalized to the straight ported design. Fig 5.11 clearly shows the power-performance “cloud” of the various designs evaluated. A well known technique for multi-criteria optimization is the use of a pareto-front based technique. Solutions that form the pareto-front are all considered optimal that offer a certain optimality in each of the objectives, rather than an absolute minimum or maximum. The optimization criteria in this experiment is to minimize power and maximize performance.

The pareto-front shown in Fig 5.11, forms an inverted “S” and is sparsely populated. Due the sparsity of the pareto-front a 5% region around the pareto-front called the desired solution region is defined and designs falling along the pareto-front or in the desired solution region are considered optimal. This improves flexibility to trade-off power and performance. Twenty designs, labeled #1 through #20 in Fig

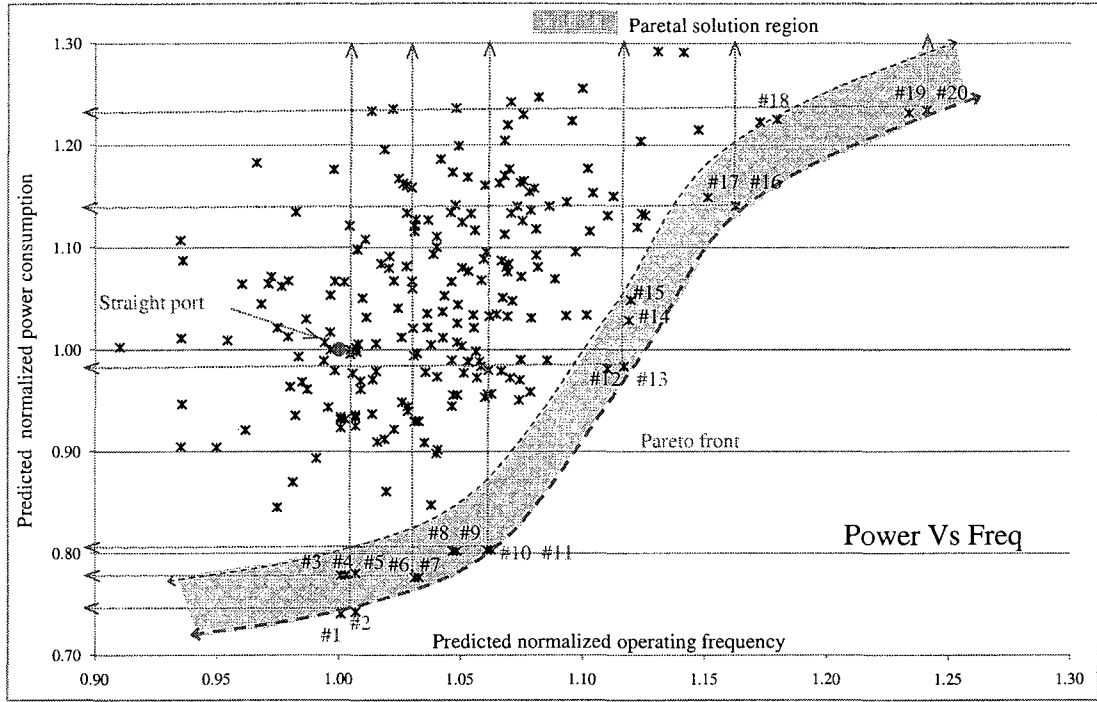


Figure 5.11: Pareto-front analysis results

5.11 lie on or within the desired solution region. To illustrate design optimization process, two pairs of designs #13 & #14 and #9 & #11 are analyzed. The modular design choices corresponding to these designs are shown in Table 5.10.

Designs #1 through #13 lie below unity normalized power ($Y = 1$) line and are considered power centric designs. Designs #12 through #20 lie above unity normalized power line and are considered performance centric designs. Designs #1, #2, #5, #12, #14, #18, #19 and #20 are seeds, #9 was generated by the simple randomizer algorithm. The remaining designs were generated by the complete randomizer algorithm.

Intentionally left blank.

Table 5.10: Modular design choices for designs #14, #13, #8 and #10

Module	Design #14	Design #13	Design #9	Design #11
1	None applied	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$	$\uparrow V_{dd}$
2	Low- V_t FETs	$\uparrow V_{dd} \& \text{Low-} V_t \text{FETs}$	None applied	None applied
3	Low- V_t FETs	ABB-FB	None applied	$\downarrow V_{dd} \& ST$
4	None applied	$\downarrow V_{dd}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
5	None applied	$\uparrow V_{dd} \& \text{ABB-FB}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
6	None applied	$\uparrow V_{dd}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
7	None applied	$\downarrow V_{dd} \& \text{ABB-RB}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
8	None applied	$\downarrow V_{dd} \& \text{ABB-RB}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
9	None applied	Low- V_t FETs	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
10	Low- V_t FETs	ABB-FB	None applied	None applied
11	Low- V_t FETs	$\uparrow V_{dd}$	None applied	$\uparrow V_{dd} \& \text{Low-} V_t \text{FETs}$
12	Low- V_t FETs	$\downarrow V_{dd} \& \text{ABB-RB}$	None applied	$\downarrow V_{dd} \& ST$
13	Low- V_t FETs	$\downarrow V_{dd} \& ST$	None applied	$\downarrow V_{dd} \& ST$
14	Low- V_t FETs	ABB-FB	None applied	$\uparrow V_{dd} \& \text{Low-} V_t \text{FETs}$
15	Low- V_t FETs	Low- V_t FETs	None applied	$\downarrow V_{dd}$
16	None applied	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$	Low- V_t FETs
17	None applied	$\uparrow V_{dd}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
18	None applied	$\uparrow V_{dd}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
19	None applied	ST	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
20	None applied	$\downarrow V_{dd} \& \text{ABB-RB}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
21	Low- V_t FETs	$\uparrow V_{dd} \& \text{ABB-FB}$	None applied	None applied
22	None applied	ABB-RB	$\downarrow V_{dd} \& ST$	$\uparrow V_{dd} \& \text{ABB-FB}$
23	None applied	ST	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
24	Low- V_t FETs	Low- V_t FETs	None applied	$\uparrow V_{dd} \& \text{Low-} V_t \text{FETs}$
25	None applied	ST	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$
26	None applied	$\uparrow V_{dd}, \text{Dual-} V_t \& \text{ABB-FB}$	$\downarrow V_{dd} \& ST$	$\downarrow V_{dd} \& ST$

5.3.1.1 Analysis of Power Centric Designs #9 and #11

Among power centric designs, design #7 dominates #3, #4, #5 and #6, design #9 dominates #8 and design #11 dominates #10. Therefore the power centric designs are #11, #9, #7, #2 and #1. Design pair #9 and #11 has the most increase

in performance with least increase in power. Figs 5.12 and 5.13 show the modular

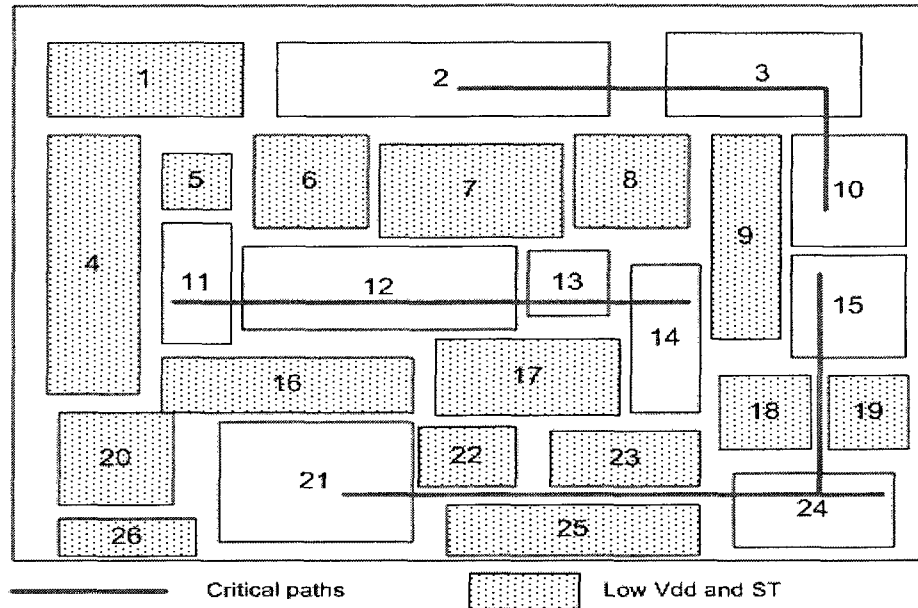


Figure 5.12: Design #9 details

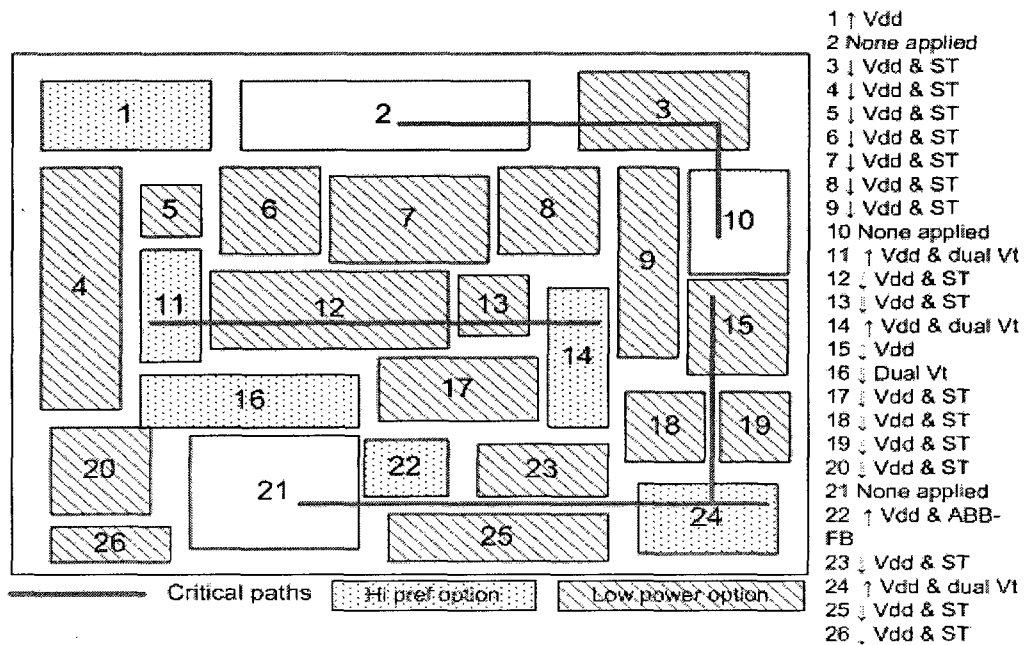


Figure 5.13: Design #11 details

design choices and the critical path for designs #9 and #11 respectively. Compared to the straight ported design, generated designs #9 and #11 both optimize the design power and performance. However design #11 with respect to #9 improves performance with a negligible increase in power. Power centric design solution i.e. design #11 offers 19.6% power savings with 6.3% improvement in performance, optimizing both power and performance with respect to the straight ported design.

5.3.1.2 Analysis of Performance Centric Designs #13 and #14

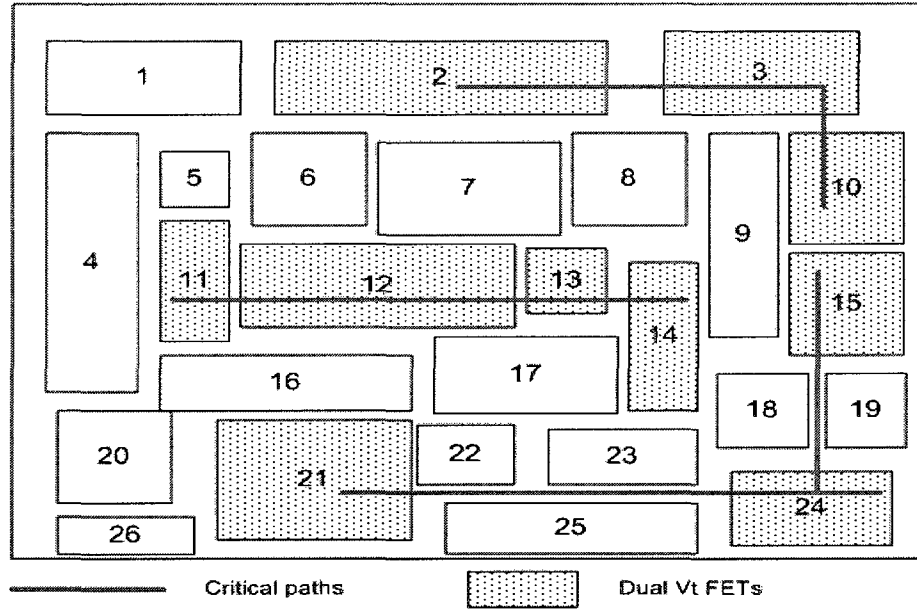


Figure 5.14: Design 14 details

Among performance centric designs, design #14 dominates #15, design #16 dominates #17 and design #19 dominates #18. Therefore the power centric designs are #14, #16, #19 and #20. Designs #14 and #13 straddle the unity normalized power line, and design #13 with respect to #14 offers 4.5% power reduction for a negligible performance degradation of 0.21%. Design #16, with respect to #13 offers a 4.6% performance improvement for a 15.7% power penalty. Designs #19 and #20,

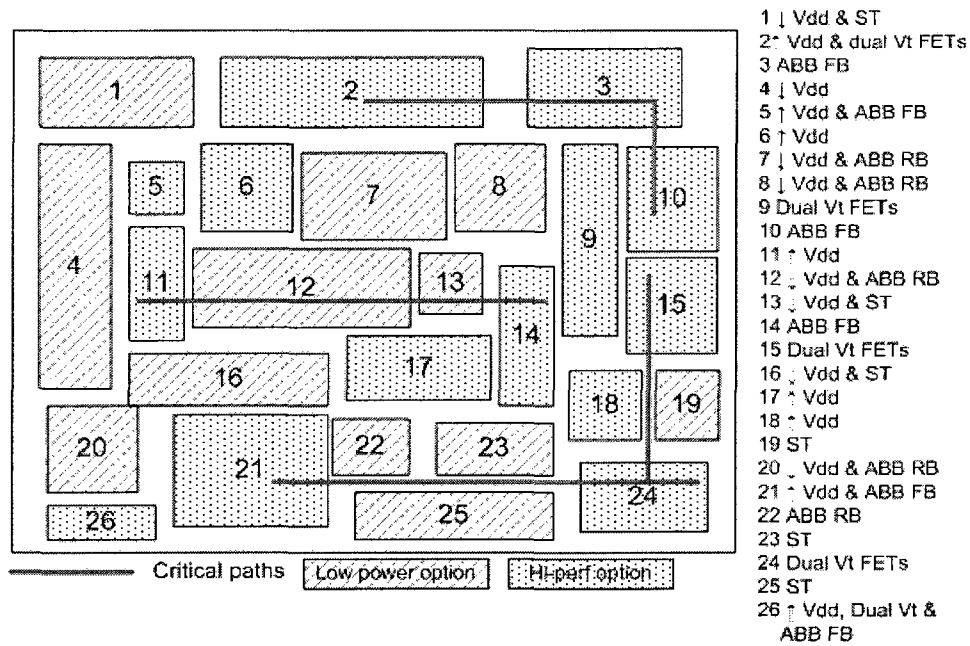


Figure 5.15: Design 13 details

with respect to #13 offer about 12% performance improvement for about a 25% power penalty. Therefore design pair #14 and #13 have the most reduction in power for the least performance impact.

Figures 5.14 and 5.15 show the modular design choices and the critical path for designs #14 and #13 respectively. Compared to the straight ported design, designs #14 and #13 both improve performance. Design #13 offers a 11.7% performance improvement and a 1.63% decrease in power consumption. Performance centric design solution i.e. design #13 offers 11.7% improvement in performance and reduces power consumption by 1.63%, optimizing both power and performance with respect to the straight ported design.

5.3.2 Results of Pareto-Analysis Using EA Based Design Generation

Fig 5.16, shows the pareto-front progression at intermediate points for a EA optimization with 50K iterations. Fig 5.16 shows that the pareto-front progression

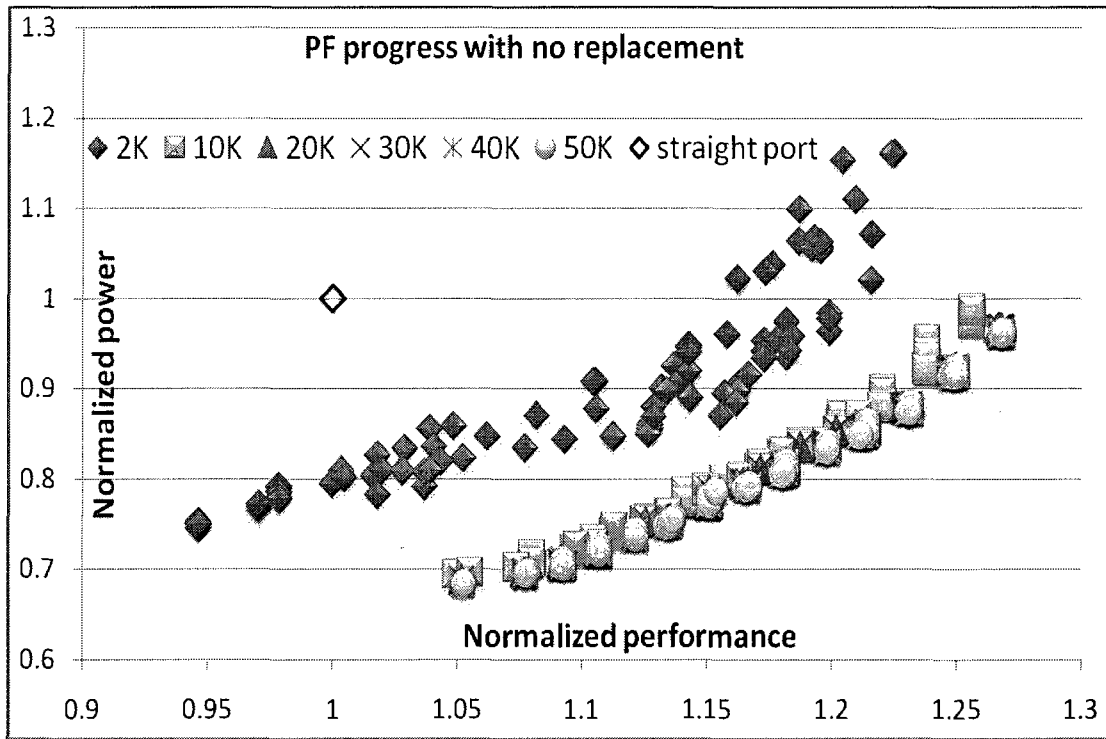


Figure 5.16: EA pareto-front progression - baseline with no replacement

quickly settles around 10K iterations and the progression beyond 10K is minimal. Moreover, the solutions along the final pareto-front (50K) appear to have crowded, leading to reduced bio-diversity in the population. Design space exploration aims to generate a well spread pareto-front with high bio-diversity which enable various trade-off analysis. Various techniques such as normal boundary insertion can be used to improve pareto-front spread and increasing bio-diversity [90]. These techniques are in general computationally intensive involving many steps of Pareto-front calculations.

Here two simple chromosome replacement schemes are used to improve pareto-front spread.

5.3.3 Pareto-Front Decrowding Replacement Schemes

The two chromosome replacement schemes for pareto-front decrowding are, edge extension replacement (EER) and interior redundancy reduction replacement (IRRR). In the standard iterative procedure (i.e. without replacement, NOR), a newly generated chromosome which happens to be a pareto optimal but lies outside the boundary of the pareto-front formed by the current population will get “effectively” discarded¹. This is because, no chromosome in the current population will dominate or be dominated by the newly generated chromosome. The decrowding replacement schemes identify such chromosomes as “outside-pareto” chromosomes and **forces** the outside-pareto chromosomes into the current population. The two replacement schemes differ in choosing which chromosome in the current population the forced outside-pareto chromosome replaces.

When an outside-pareto chromosome is encountered, the EER scheme chooses the chromosome in the current population which has the least geometric distance to the outside-pareto chromosome for replacement. Thus extends the current pareto-front’s edge outward in the direction specified by the outside-pareto chromosome. When an outside-pareto chromosome is encountered, the IRRR scheme examines the current population to find a pair of “candidate” chromosomes with the least amount of difference in their fitness values, for replacement. One of the two candidate chromosomes is then arbitrarily selected to be replaced with the outside-pareto chromosome.

¹The boundary of the pareto-front of is defined as the area of the design space between maximum and minimum fitness values for each criteria (axis), for the chromosomes in a given population

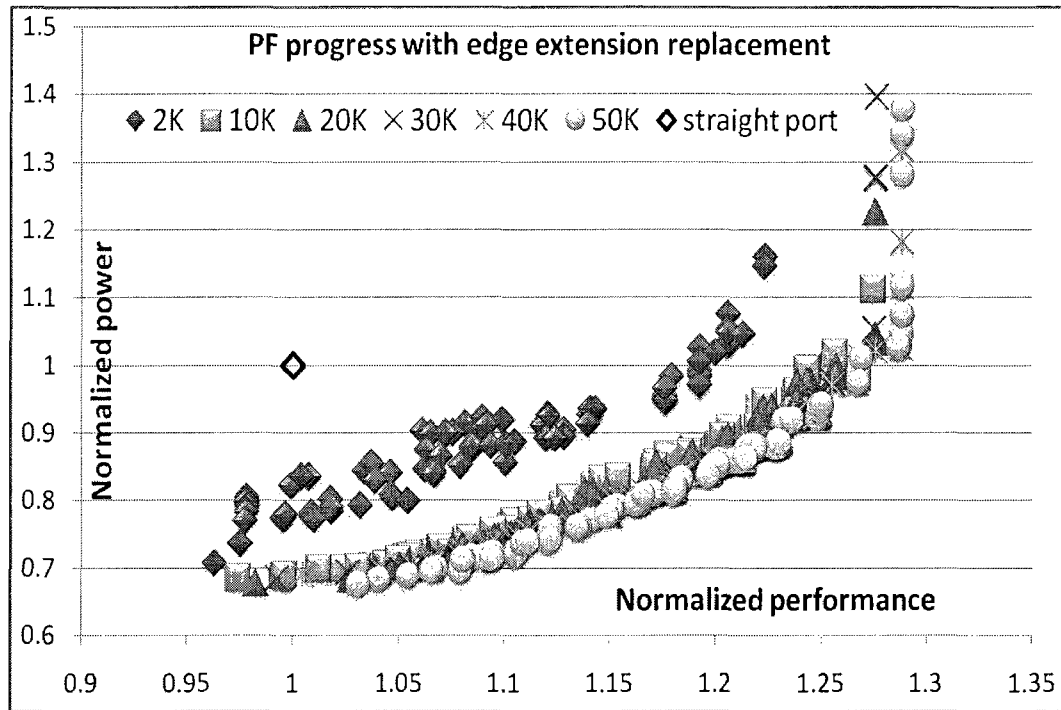


Figure 5.17: EA pareto-front progression - with EER

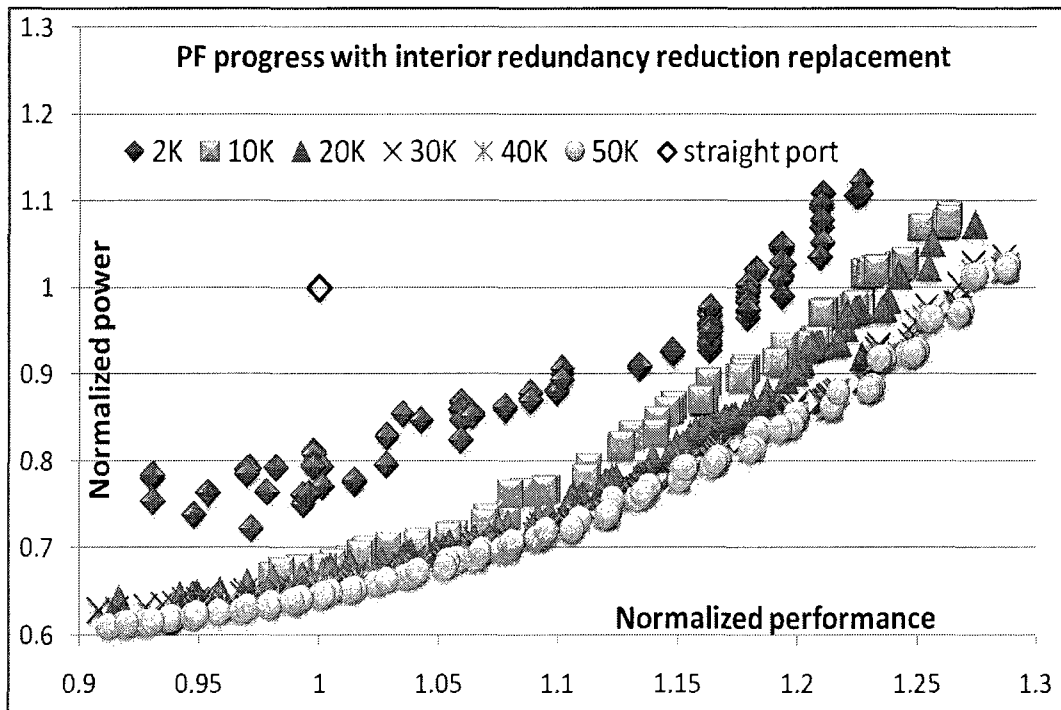


Figure 5.18: EA pareto-front progression - with IRRR

Figs 5.17 and 5.18 show the pareto-front progression at intermediate points for a EA optimization with 50K iterations with ERR and IRRR schemes respectively. Figs 5.17 and 5.18 show that IRRR scheme generates pareto-front that are well spread with higher bio-diversity. In order to quantitatively analyze the generated pareto-fronts two figure of merit (FOM) formulations are defined. The FOM formulation and the analysis of the generated pareto-fronts are explained next.

5.3.4 Figures of Merit: Stopping Criteria

The two FOM formulations i.e. FOM for pareto-front solution quality (FOM_SQ) and FOM for pareto-front solution spread FOM_SS are used to quantitatively analyze the generated pareto-fronts of the different schemes. Also these metrics can be used to determine the ideal number of iterations and the speed of convergence for different schemes. Consider a pareto-front with p chromosomes (population size of p). Then the two FOMs are defined in Eqns 5.1 and 5.4. F is the fitness vector with λ objectives (elements) for a chromosome, where objectives 1 through ϑ are minimization objectives and $(\vartheta + 1)$ through λ are maximization objectives. Note that these definitions assume that the pareto-front points are normalized to a suitable reference and are positive.

$$FOM_SQ = Median\left(\left[\frac{\phi_\nu}{\chi_\nu}\right]\right); \forall \quad \nu : 1 \dots p \quad (5.1)$$

$$\chi_\nu = \prod_{j=1}^{\vartheta} F_{\nu j} \quad (5.2)$$

$$\phi_\nu = \prod_{k=(\vartheta+1)}^{\lambda} F_{\nu k} \quad (5.3)$$

$$FOM_SS = \prod_{x=1}^{\lambda} \sigma_x \quad (5.4)$$

$$\sigma_x = Std\ Dev(F_{1x}, F_{2x}, \dots, F_{px}) \quad (5.5)$$

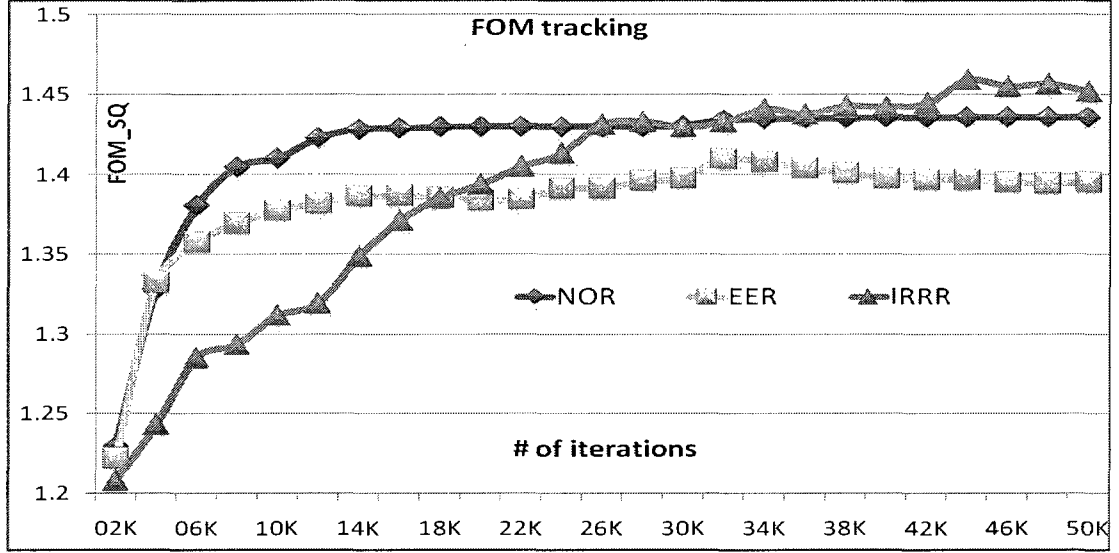


Figure 5.19: Figure of merit for pareto-front solution quality

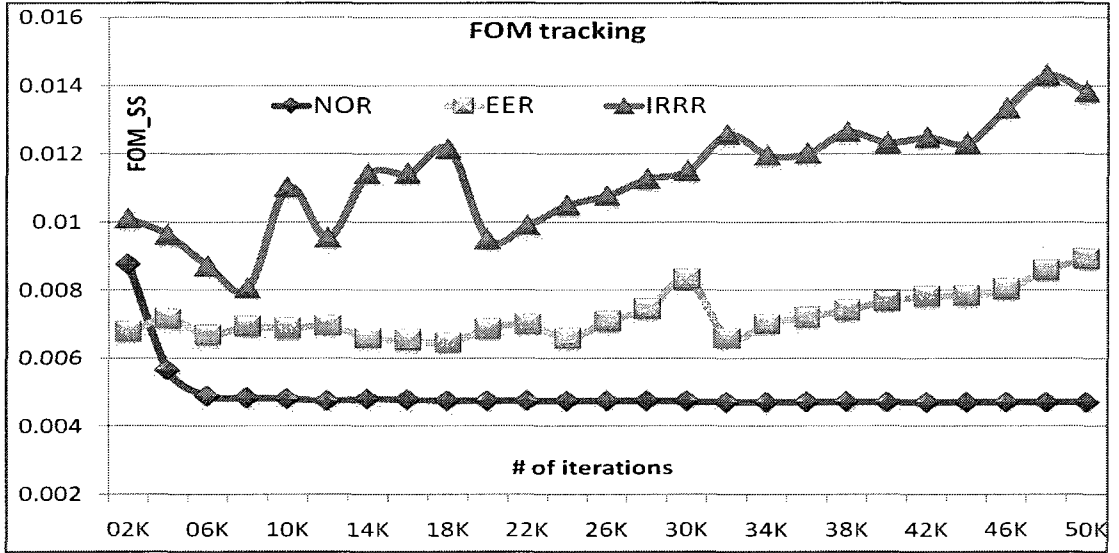


Figure 5.20: Figure of Merit for pareto-front solution spread

Figs 5.19 and 5.20 show FOM for pareto-front solution quality and FOM for pareto-front solution spread respectively for EA runs with the two replacement schemes

(EER and IRRR) and no replacement (NOR). The FOMs for the generated pareto-fronts for every 2K iterations are calculated and plotted. Solution quality for NOR and EER increase quickly and saturate quickly around 22K and 12K iterations respectively. Moreover from Fig 5.17 it is clear that the EER solution includes many “outliers” (note the Y axis range) since the EER scheme simply extends the edge out. As a result solutions generated by EER have higher spread (due to outliers) but are of low quality as confirmed by the curve for ERR in Fig 5.19 which is below the other two schemes from 14K iteration and above. The IRRR scheme generates solutions with fewer outlying chromosomes (Fig 5.18) since the most redundant chromosome for replacement can be located anywhere in the pareto-front, this prevents successive replacements at the edge and improves the chance of generating a chromosome that dominates the outlying chromosomes. However when interior redundant chromosomes are replaced, population diversity increases but rate of increase in solution quality decreases. Hence the solution quality for IRRR scheme achieves parity with the other two schemes only after 10K iterations, but continues to increase almost monotonically before saturating around 48K iterations.

Solution spread for EER and NOR are low compared to IRRR, with NOR saturating around 12K iterations and EER displaying a general decreasing trend until 40K iteration where the pareto-front particularly at high performance regions appears to be vertically spreading, increasing the solution spread as seen in Fig 5.17 but decreasing solution quality. Fig 5.20 show that the IRRR schemes’s solution spread is generally higher than the other schemes, but appearing to have an oscillatory behavior. This is because with more iterations the randomness in the initial/current population reduces converging to a pareto-front (which can be further improved with more iterations) reducing the overall solution spread. For instance from 2K to 4K and 8K to 10K for IRRR in Fig 5.20. Then the pareto-front spreads along the direction of

the generated outside-pareto chromosome, which increases the solution spread. That is from 4K to 8K and 10K to 16K for IRRR in Fig 5.20. Moreover during the standard iterative process with IRRR, the solution spread reduces when newly generated chromosomes dominate and replace relatively outlying chromosome in the current population. The solution spread increases when more “outside-pareto” points are generated. Once the current population’s pareto-front is sufficiently close to the edge of the “true” pareto-front, then with more iterations the solution spread will saturate before starting to decrease and the solution quality will saturate. The IRRR scheme exhibits this expected behavior around 46-50K iterations which forms the necessary stopping criteria for design space exploration using EA and IRRR.

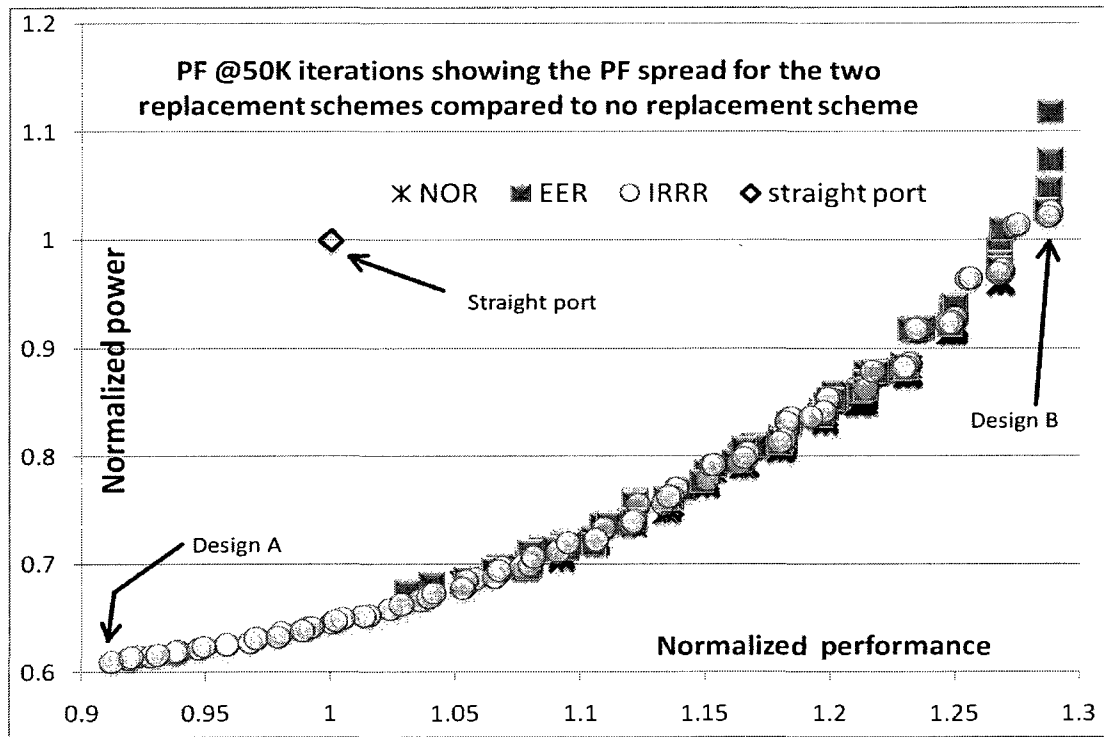


Figure 5.21: Pareto-front analysis results

Fig 5.21 compares the final pareto-front generated at 50K iterations by the NOR, EER and IRRR schemes. It clearly shows the spread and quality for IRRR scheme

to be better compared to the other two schemes and far superior to the pareto-front obtained from the randomizer algorithms in section 4.4.2. Moreover the IRRR pareto-front has the highest range from high performance designs to low power designs and designs that optimize both power and performance. Thereby providing a wider choice for design space exploration and power performance trade-offs.

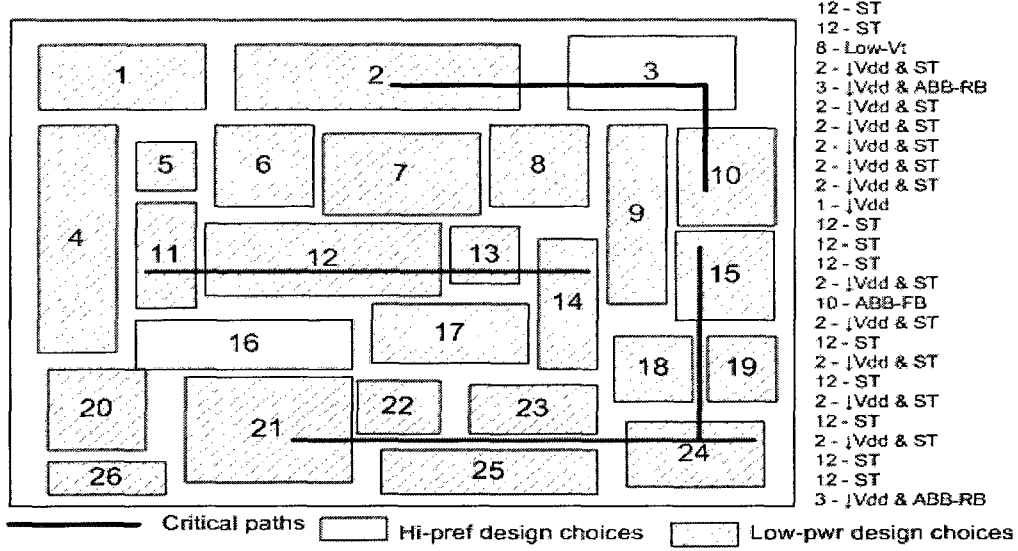


Figure 5.22: Details of system design A from Fig 5.21

Figs 5.22 and 5.23 show the details of system design A and B from Fig 5.21, respectively. Design A improves the straight-port design with 40% power reduction with only 9% performance impact. Similarly design B improves the straight-port design with 29% improvement in performance with only 2.5% power penalty. EA based design space exploration with IRRR generates Pareto-fronts that optimize both system power and performance. Solutions uncovered henceforth are non-intuitive and are not immediately obvious, thus, enabling designers to perform quick, relatively accurate design space exploration and trade-off analysis early in the design phase. This ability is a key contribution of the proposed methodology.

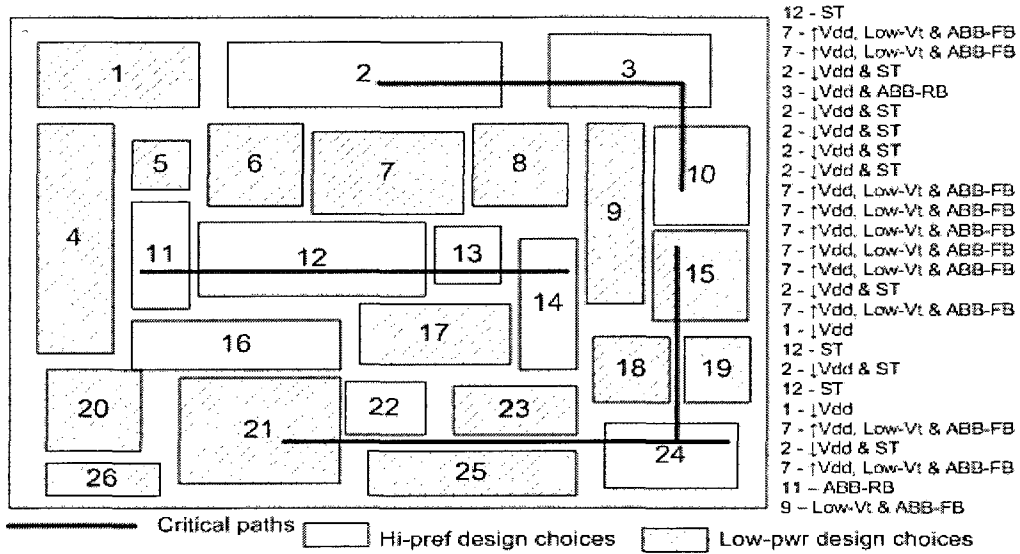


Figure 5.23: Details of system design B from Fig 5.21

5.3.5 Results of Pareto-Analysis Using the IRRR Scheme on ISCAS89 Circuit

With the effectiveness of the IRRR scheme demonstrated in the previous section, two large ISCAS89 circuits are used to further validate IRRR scheme's effectiveness in this section. Fig 5.24 shows the Pareto-fronts for the two ISCAS89 circuits after 50K iteration with IRRR scheme. The Pareto-front for s38417 circuit was expected to be better of the two since (as pointed out in Section 5.1.1.3) one module in s38417 contributed less than 3% to the critical path delay. Therefore power consumption could be significantly reduced with minimal impact on performance. Since s38584 circuit did not have such an advantage and the critical path was nearly equally divided between all four modules, the final Pareto-front for this circuit was inferior as shown in Fig 5.24. This proves that the proposed design framework allows for such opportunities to be uncovered and subsequently generating solutions that optimize both system power and performance.

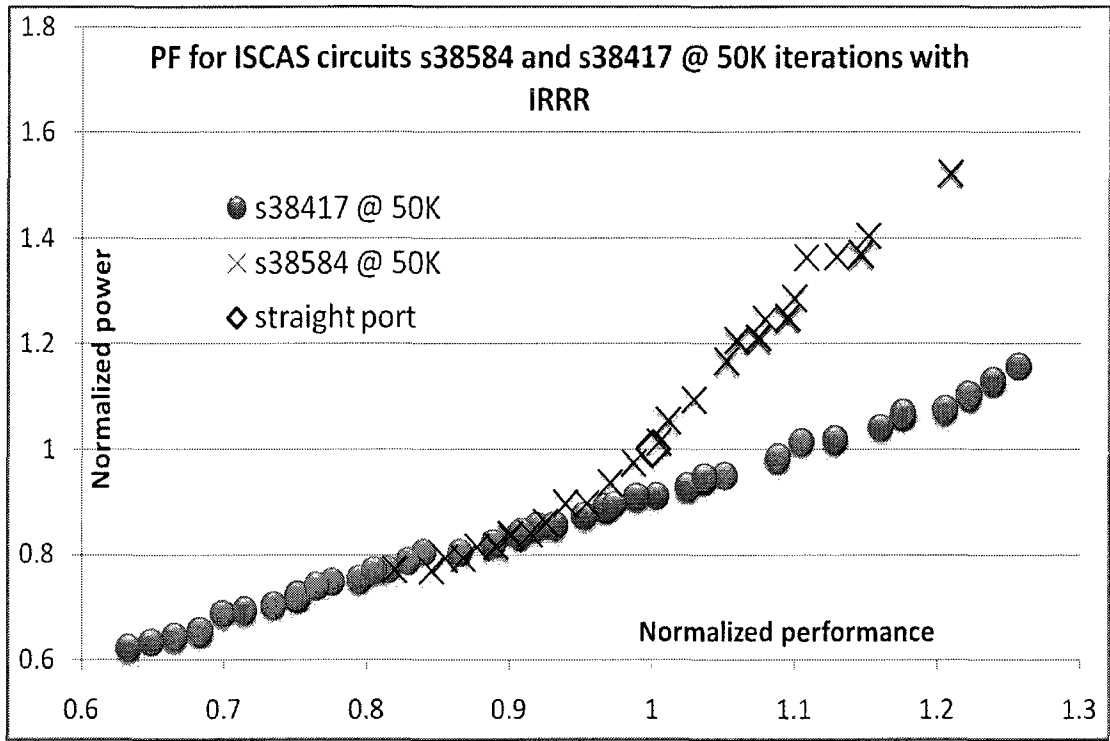


Figure 5.24: Pareto-front with IRRR at 50K iteration for ISCAS89 s38584 and s38417 circuits

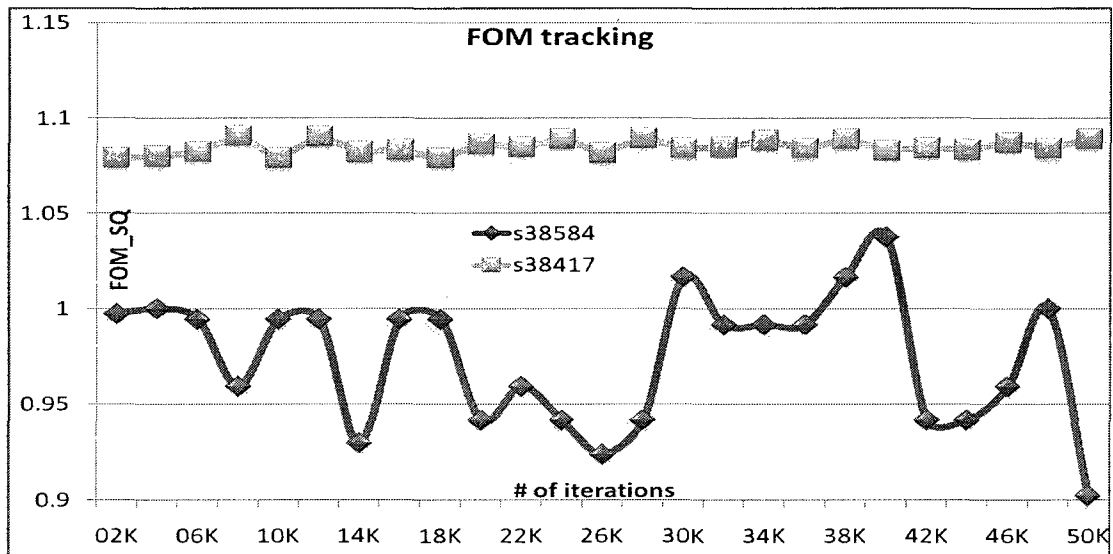


Figure 5.25: Figure of merit for ISCAS89 s38584 and s38417 circuits Pareto-front with IRRR solution quality

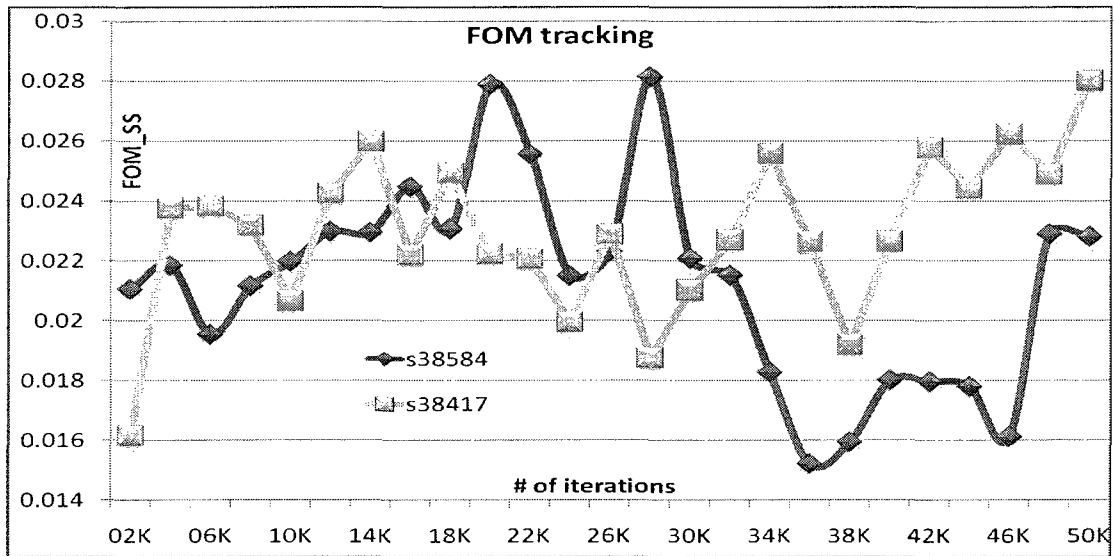


Figure 5.26: Figure of Merit for ISCAS89 s38584 and s38417 circuits Pareto-front with IRRR solution spread

Figs 5.25 and 5.26 show the FOM for Pareto-front solution quality and FOM for Pareto-front solution spread, respectively, for the ISCAS89 circuits with IRRR. As expected the solution quality for the s38417 circuit is better compared to s38584 circuit since the latter circuit had fewer opportunities to optimize both power and performance simultaneously. The FOM for solution quality for s38417 circuit saturates quickly compared to the IRRR solution quality for the microprocessor based design. This is due to the difference in the problem size i.e. chromosome length of 4 as opposed to 26 respectively. However, the solution spread FOM indicates that the spread of the Pareto-front consistently improves only later around 38K iterations.

5.4 Discussion of the Results

5.4.1 Impact of ABB Design Choice

Large area overhead and increased design complexity of using the ABB technique may make the ABB choices undesirable, particularly when ABB is used in some

modules are mixed with other modules with no ABB on the same chip. To ascertain the impact of ABB, an EA run with IRRR and 50K iterations without any ABB design choices was performed and its result was compared to the result of the EA run with IRRR and 50K iterations with all design choices.

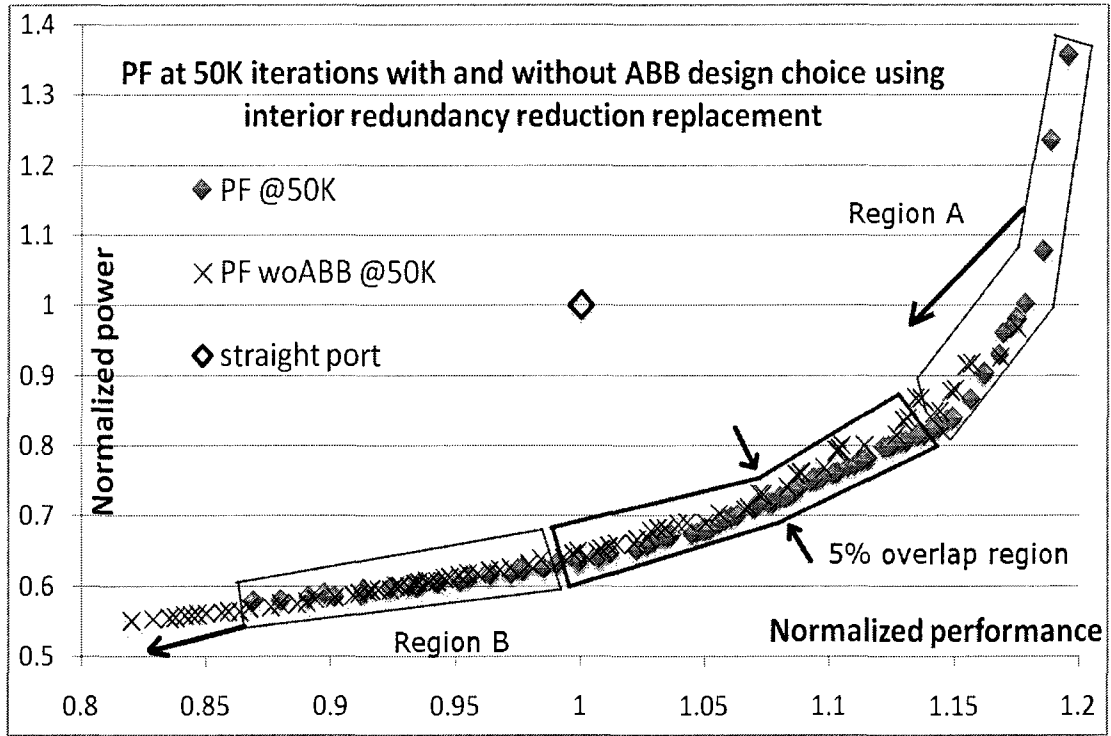


Figure 5.27: Impact of ABB design choice

Fig 5.27 shows the impact of ABB design choice on the final pareto-front at 50K. High performance designs (region A in Fig 5.27) predominantly use ABB-FB to improve performance, where removal of ABB-FB leads to reduction in both power and performance, as illustrated in Fig 5.27. In general, ABB-RB can be used to reduce power consumption, however power consumption reduction using sleep transistors (ST) is much larger. Low power designs (region B in Fig 5.27) predominantly use ST (in addition to ABB-RB) to reduce power and use ABB-FB to offset the performance impact of ST. Removal of ABB (both FB and RB) in low power designs lead to

reduction in performance as well as power consumption, as illustrated in Fig 5.27. As a result the pareto-front without ABB in comparison with the pareto-front with ABB, is shifted along the direction of lower power and performance. Interestingly, there exists remarkable overlap in the two pareto-fronts where the same design goals can be met with much less design complexity. Moreover as shown in Fig 5.27 if a 5% power penalty region is considered, designs with reduced complexity (without ABB) can be uncovered with sufficient performance and substantial power savings.

5.4.2 Prediction Model Complexity: Impact of Model Parameters

The analytical design target prediction models (Eqns 3.1 through 3.27) described earlier in the paper have many parameters contained in them. Majority of these parameters are easy to obtain. For example, technology dependent model parameters and legacy design dependent parameters. These parameters are mainly obtained from SPICE characterization which are similar to routine library characterization and from legacy design data and new process information, respectively. However several parameters, such as, η - package in Table 3.2, BUFLC in Eqn 3.13, BSUF in Eqn 3.24 and USPE in Equ 3.25, are difficult to obtain.

One could argue that eliminating some of these parameters from the equations would simplify the models without sacrificing prediction accuracy. In order to better understand the impact of those "more-difficult-to-obtain" parameters on the prediction models, a set of sensitivity analysis is performed using the test circuit in Fig 4.2. Descriptor values the various technologies used in this experiment are the same as in Table 4.4.

Table 5.11 shows the normalized power and performance sensitivity for 130nm, 90nm and 65 nm PTM, when the corresponding parameter value is changed by 10%

Table 5.11: Power and performance sensitivity of selected model parameters

Descriptor / Technology	Power Sensitivity			Perf. Sensitivity		
	130nm	90nm	65nm	130nm	90nm	65nm
η (Table 3.2)	2.23	2.19	2.23	0.59	0.70	0.66
DECAP_SENS (Eqn 3.7)	1.68	1.79	1.75	0.74	0.79	0.77
β (Eqn 3.31)	-0.96	-0.68	-0.65	-0.25	-0.30	-0.21
γ (Eqn 3.32)	-0.96	-0.68	-0.65	-0.25	-0.30	-0.21
δ (Eqn 3.33)	-0.96	-0.68	-0.65	-0.25	-0.30	-0.21
α (Eqn 3.30)	-0.94	-0.67	-0.64	-0.25	-0.30	-0.21
USPE (Eqn 3.25)	0.91	0.94	0.87	1.00	1.00	1.00
RPSF (Eqn 3.24)	0.91	0.94	0.87	n/a	n/a	n/a
ϵ (Eqn 3.34)	-0.82	-0.47	-0.48	-0.09	-0.07	-0.02
Y (Eqn 3.35)	-0.65	-0.53	-0.60	-0.22	-0.27	-0.25
X (Eqn 3.35)	-0.63	-0.51	-0.60	-0.20	-0.25	-0.25
BSUF (Eqn 3.24)	0.49	0.56	0.48	0.54	0.60	0.55
RCSF (Eqn 3.1)	-0.48	-0.55	-0.48	-0.53	-0.59	-0.55
sf (Eqn 3.14)	-0.03	-0.02	-0.06	n/a	n/a	n/a
n/a - not applicable						

from its default value. Model parameters are listed according to the absolute magnitude sensitivity to power and performance, in the descending order for 130nm PTM process. Chip packaging quality i.e. the IR drop in the package interconnects (η) and change in supply voltage due to de-coupling capacitance insertion (DECAP_SENS), have the highest impact on power consumption and performance as the power grid voltage is a strong function of η and DECAP_SENS.

De-coupling capacitance insertion improves supply grid voltage and leads to performance improvement. However, leakage power due to the de-coupling capacitance as well as dynamic power due to performance improvement both increase, increasing total power consumption. For designs that are power constrained, power consumption can be reduced by reducing de-coupling capacitance and with a performance penalty. Provided, the initial amount of de-coupling capacitance inserted was not large enough to cause a saturation in achievable power grid voltage improvement. With incremental de-coupling capacitance added there is diminishing return, i.e. the

power grid's sensitivity to unit de-coupling capacitance would decrease. To illustrate how a designer would study the impact of a change in the power grid's sensitivity and may choose to simplify the model, the normalized power and performance sensitivities for the test circuit were recalculated after reducing the added de-coupling capacitance by half and assuming that the power grid's sensitivity increases by 30%. This assumption is in line with experimental results in [91].

Table 5.12: Power and performance sensitivity of selected model parameters with reduced de-coupling capacitance

Descriptor	Power Sensitivity			Perf. Sensitivity		
	130nm	90nm	65nm	130nm	90nm	65nm
η (Table 3.2)	2.38	2.46	2.42	0.69	0.91	0.80
RPSF (Eqn 3.24)	0.92	0.94	0.89	n/a	n/a	n/a
USPE (Eqn 3.25)	0.92	0.94	0.89	1.00	1.00	1.00
β (Eqn 3.31)	-0.89	-0.72	-0.63	-0.31	-0.41	-0.28
γ (Eqn 3.32)	-0.89	-0.72	-0.63	-0.31	-0.41	-0.28
δ (Eqn 3.33)	-0.89	-0.72	-0.63	-0.31	-0.41	-0.28
α (Eqn 3.30)	-0.87	-0.71	-0.63	-0.31	-0.41	-0.28
ϵ (Eqn 3.34)	-0.77	-0.54	-0.49	-0.18	-0.22	-0.12
Y (Eqn 3.35)	-0.61	-0.57	-0.59	-0.26	-0.36	-0.31
X (Eqn 3.35)	-0.59	-0.55	-0.59	-0.24	-0.34	-0.31
DECAP_SENS (Eqn 3.7)	0.51	0.61	0.58	0.11	0.17	0.14
BSUF (Eqn 3.24)	0.47	0.52	0.46	0.51	0.55	0.52
RCSF (Eqn 3.1)	-0.46	-0.51	-0.45	-0.50	-0.55	-0.51
sf (Eqn 3.14)	-0.04	-0.03	-0.04	n/a	n/a	n/a
n/a - not applicable						

Table 5.12 (similar to Table 5.11) shows the model parameter sensitivities resulting from this experiment. Reducing the de-coupling capacitance in half and increasing the power grid's sensitivity by 30%, leads to a 26% reduction in total power with a 6% performance impact. Here, power and performance impact corresponding to a unit change in DECAP_SENS parameter value are low and hence in this case default values may be used with an acceptable prediction accuracy. Sensitivities for other

parameters are very similar to values in Table 5.11. For designs where altering supply voltage is not a design option, default values for parameters α , β , γ , δ and ϵ can be used without sacrificing prediction accuracy. Similarly, other parameters can use default values depending on design constraints that allow using default values, exists. For the test circuit used, total power consumption is least sensitive to stacking factor (sf) since dynamic power dominated leakage power (by approx 9X) and hence the sensitivity of the total power for a 10% change in sf is small. Therefore using default values for sf parameter in this case would not considerably impact power predictions unlike parameter η . Furthermore with different process technologies the sensitivities for some parameters reduce, for example ϵ . Therefore, for a given design in a given process, the tool user can choose to use default values for such parameters to simplify the analytical prediction model. Given this flexibility, the models presented in this paper are therefore sufficiently accurate without being excessively complex to capture various design choices' impact on system power and performance.

5.4.3 Pareto-front Quality: Impact of Evolutionary Algorithm

The IRRR scheme (section 5.3.4) used to improve pareto-front spread, is influenced by the concept of “crowding distance” based pareto-front spread improvement outlined in [92]. However, the implementation is much simpler compared to the NSGA-II algorithm in [92]. Similar to the NSGA-II, the IRRR scheme normalizes the objective function values, does not require any user defined sharing parameters and, can be applied to problems with more than two objectives.

In [93] three metrics to compare non-dominated sets (i.e. a Pareto-front at the end of an evolutionary algorithm run) were proposed. They are the D-metric for accuracy, Δ -metric for pareto-front uniformity and the ∇ -metric for pareto-front extent. The proposed FOM_SS metric for solution spread is similar to the ∇ -metric

in the sense that both measure the extend (spread) of the pareto-front set in all dimensions to ascertain the hyper-volume of the pareto-front. The proposed FOM_SQ metric for solution quality measures the quality of a pareto-front as opposed to the D-metric which requires two pareto-fronts to compute the metric. Therefore to compare two pareto-fronts (from two EA runs for instance) a third reference pareto-front is needed to compare each of the two pareto-fronts individually to the reference pareto-front. In the proposed FOM_SQ formulation there is no need for a reference pareto-front, since choosing this reference is one additional source of variability which is can be avoided. The δ -metric for pareto-front uniformity is unique and the proposed FOM formulations does not have a metric similar to the δ -metric. However such a metric will be redundant since the uniformity of a pareto-front can be ascertained from the fitness standard deviations.

The strict one child one parent replacement policy used in this work limits the elitism offered by the evolutionary algorithm. The focus of the work at this time is establishing the applicability of the proposed design framework for design space exploration; hence a simpler evolution algorithm with IRRR is used. Integrating a more rigorous algorithm such as the NSGA-II to improve the design framework for design space exploration is a natural extension of the work presented here.

5.5 Conclusion

System power performance optimization is most effective early in the design phase when design space exploration is performed. Design convergence in both power and performance, especially in nanometer CMOS, has become increasingly difficult and choosing an optimal implementation target is imperative for meeting time-to-market requirement. The proposed framework for design space exploration using legacy design data, technology scaling trends, and in-situ simulations provides design

aid in choosing optimal implementation targets for better design convergence. The technology node migration experiment on ISCAS benchmark circuits established the feasibility of the proposed methodology. For these circuits, the results from design space exploration offer 7-32% power reduction with 0-9% performance degradation or 11.25-17% performance improvement with 2-3.85% power penalty.

Furthermore, power and performance prediction model accuracy in successive CMOS technologies from 180 nm to 65 nm were experimentally estimated and compared to similar estimates from existing literature. The accuracy is inline and comparable to results from other previously published works. Evolutionary algorithm based design space exploration using Pareto analysis of a microprocessor based design yielded Pareto optimal solutions that optimized both power and performance. The evolutionary algorithm was improved by a replacement schemes for decrowding to generate well spread Pareto-fronts to allow better trade-off analysis. The design space exploration generated designs that improved the straight-port design by reducing power by 40% with 9% performance impact or by improving performance by 29% with 2.5% power penalty. In addition to generating Pareto optimal designs, ways to reduce system implementation complexity with bounded impact on power and performance were presented.

The experimental results illustrate the ability of the proposed framework to uncover complex non-intuitive solutions using evolutionary algorithm based Pareto analysis. By incorporating the proposed methodology as part of a standard system design flow, quick what-if analysis can be performed at a high level and underlying design risks can be exposed very early in the design phase. The key contributions of the proposed framework are, a) the ability to uncover Pareto optimal, complex and non-intuitive system designs. b) The ease of performing high level tradeoff and what-if analysis. c) The potential to expose any underlying design risks very early in the

design phase and d) Design complexity reduction by generating solution without complex design choices such as ABB with a bounded power and performance impact. Thus, improving system design convergence and meeting time-to-market schedule.

5.6 Future Work

The work presented here are the first few steps in developing a design framework and methodology for early design space exploration utilizing technology scaling trends, process dependent parameters and in-situ simulations. The following areas are identified as natural extensions to the work presented in this dissertation. In addition to the design target prediction models for power and performance, the envisioned design framework will be greatly benefitted if models for predicting the system reliability (FIT - failure in time) and design choices affecting reliability are included. Models to predict the chip area and yield, a closely related design target would improve the proposed design framework's value as a design tool.

The proposed framework does not include any estimates for the impact of process variation on design targets which are becoming important design considerations. Special libraries, RF and analog macro modules that does not follow standard scaling trends are not included in the proposed framework. Critical path delay and clock signal arrival uncertainties ([94]) are not currently accounted for in the proposed framework. A macro model based interconnect delay estimation and needs to be included to extend the proposed methodology well into "More-than-Moore" design era.

Improving the evolutionary algorithm and including algorithms such as NSGA-II [92] or SPEA2 [95] to explore the design space and generate the pareto fronts will aid in further optimizing system designs. Integrating more rigorous algorithms such as the NSGA-II or SPEA2 to improve the design framework for design space exploration

and comparing the relative merits of IRRR scheme and the various algorithms, potentially using the PISA framework [96], are natural extensions of the work presented here. NSGA-II improves the pareto-front spread while SPEA2 provides a better distribution of points when the number of objectives increases [95]. Since the problem that we apply the evolutionary algorithm on has no absolute known pareto-front, it will be interesting to see how much improvement these advanced algorithms achieve compared to the IRRR scheme. PISA's assessment metrics and the two proposed FOM can be used to compare the Pareto-fronts generated by the various algorithms. Additionally, the uniformity (Δ -metric) metric proposed in [93] can be included to compare the Pareto-fronts generated by the various algorithms.

Another area of study that remains ripe for new research is the development of statistical design target prediction models. These models are extension of current models capable of estimating confidence intervals for the predicted design targets, given a set of descriptor uncertainties. This ability is seen by as a key developmental area for the envisioned design framework.

Chapter 6

Acknowledgement

Dr. Cengiz Alkan, VLSI System Lab, Colorado State University, assisted in developing and testing the evolutionary algorithm for design space exploration.

Appendix A

Common Design Techniques Incorporated into the EIDA Tool

A design when modeled using the proposed system modeling methodology described in Section 3.2, can be optimized for power and performance by applying module granular circuit level design choices. The analytical design target prediction models described in Section 3.4 are used to estimate power consumption and performance of the ported design (ported from one process to an other). The initial estimate of a ported design is called as “**straight-ported**” power and performance values. Power and performance optimization of the ported design can be performed (as described in Section 3.5) by applying module granular circuit level design choices to the ported design. Module granular circuit level design choices that are included in the proposed tool are described in the remaining sections of this chapter.

Using Sleep Transistor Insertion for Power Gating

Selectively gating power supply to non-active modules in a design leads to reduction in power consumption. Fig A.1 shows an example for using sleep transistors for power gating where VDD is the power supply and VVDD is the gated virtual power supply [97]. Power gating can be applied potentially with no performance impact, however the activity in a design is workload dependent. As a result applying sleep

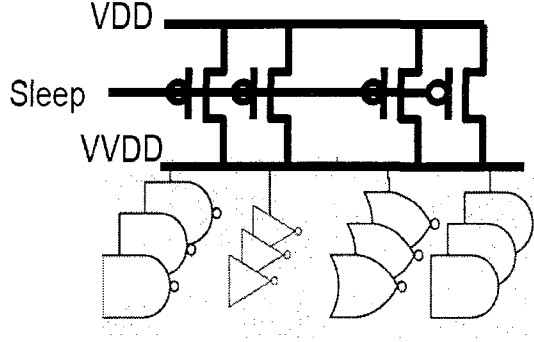


Figure A.1: An illustration for using sleep transistors for power gating.

transistors to gate the power supply impacts both power and performance due to the module being deactivated and the finite “wake-up” time involved when the sleep transistor is turned on to activate the module, respectively. The power and performance impact of applying sleep transistors are included through the STC and STPC descriptors respectively, which are estimated for each module as described in Section 3.4.6.

Using Dual/Multiple Threshold Transistors

Low threshold FETs are faster and leakier than their nominal/high threshold counterparts. Performance can be improved by selectively using low threshold FETs for gates along the critical path. This will lead to an increase in leakage power consumption. Similarly if available, high threshold FETs can be used in place of nominal FETs in gates not in the critical path to reduce power consumption. A dual threshold scheme i.e. nominal and low threshold FETs similar to the scheme described in [98] is assumed. The power and performance impact of applying dual threshold transistors is captured in EIDA through the DVTC (Section 3.4.6), HVR, LVR, I_{othv} , I_{owhv} , I_{otlv} and I_{owlv} descriptors.

Using Multiple Supply Voltage Zones

Varying the power supply voltage of a module is a design procedure that can be used to either improve performance or reduce power by increasing or decreasing the supply voltage respectively [99]. The impact of changing supply voltage on power and performance are estimated using analytical models as described in Section 3.4.4. Fig A.2 shows an example for multiple supply zones implemented within a design [100].

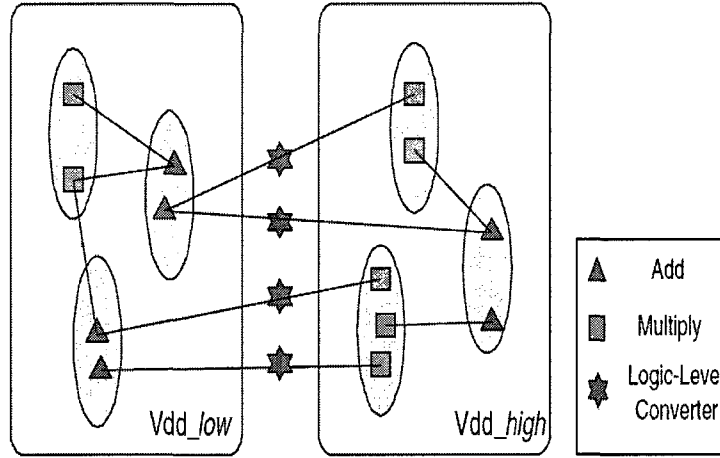


Figure A.2: An illustration for multiple V_{dd} zones in a design.

Using Adaptive Body Biasing - ABB

Applying a back bias to the body of a transistor changes its leakage and delay characteristics. With respect to the source terminal the body of a transistor can be either forward or reverse biased. When forward biased, device leakage currents increase and propagation delay decreases and vice versa for reverse bias. Biasing the body impacts device leakage in two ways, by reduction/increasing source & drain to body junction leakage and by increasing/decreasing threshold voltage to reduce/increase sub-threshold leakage [101]. The use of body biasing is captured in EIDA through the ABBC and ABBPC descriptors extracted for each module as described in section 3.4.6.

Clock Gating

Power saving can be achieved by turning off clock signal to a module when it is not active. Fig A.3 shows an implementation scheme for clock gating [102]. By turning off clock the load on the external clock tree is reduced and unnecessary switching inside the blocks need not occur, reducing dynamic power consumption. Clock gating is captured in EIDA explicitly by the CGF (Eqn 3.4) descriptor which modifies the module activity factor.

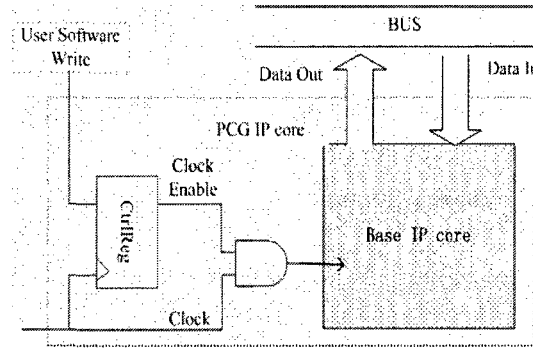


Figure A.3: An illustration for clock gating technique.

De-coupling Capacitance Insertion

Power supply grid voltage integrity in the midst of switching activity impacts performance as well as power of a design. It is desirable to maintain a constant power supply grid voltage in spite of switching activity. Inserting de-coupling capacitances between the power supply rails (V_{dd} and GND) reduces droop in power supply grid voltage. The downside of inserting de-coupling capacitances is the increase in power consumption due to leakage in de-coupling FETs which are used as capacitances. Fig A.4 shows an example of decoupling capacitance allocation in a standard cell design [103]. The performance improvement (power supply voltage improvement) due to de-coupling capacitances insertion is captured in EIDA through the DECAPC (Eqn

3.3) and DECAP_SENS (Eqn 3.7) descriptors, and the increase in power consumption is captured through the DECAPLC (Eqn 3.13) descriptor.

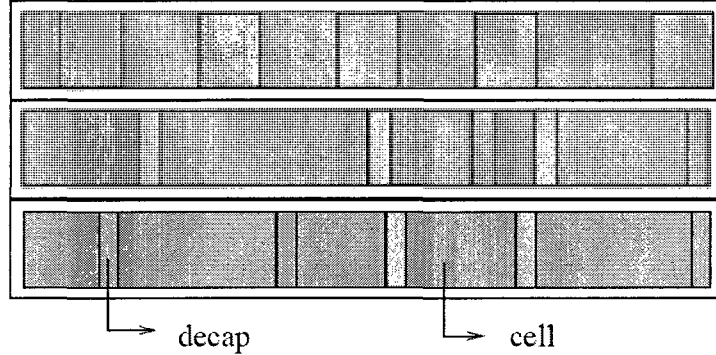


Figure A.4: Decoupling capacitance allocation in a standard cell design

Interconnect Repeater/Buffer Insertion

Global and long interconnect delay can be improved by repeating the interconnects [104]. Repeaters divide the interconnect into smaller portions. Interconnect delay reduces quadratically with length and hence repeater insertion improves overall interconnect delay but increases power consumption due to additional load of the repeaters themselves. The performance impact of repeater insertion is included in EIDA through the BSUF (Eqn 3.24) descriptor. The leakage power impact of repeater insertion is captured through the BUFLC (Eqn 3.13) descriptor and the dynamic power impact is captured through adding the additional switching capacitance C_{wire_buff} (Eqn 3.2) to the total module switching capacitance.

Useful Skew and Time Borrowing

A module may have multiple logic paths from input to output where some are critical paths. A critical path in a module may contain both combinatorial and sequential elements. Logic paths in a module have either positive or negative timing

slack. By utilizing positive timing slack and re-timing portions of logic paths typically lead to an improvement in performance. A procedure to optimize timing slack is described in [105]. Performance improvement due to useful skew allocation is included in EIDA by the USPE (Eqn 3.25) descriptor.

Appendix B

Code for System Design Target Prediction

File name: eida.pl

Function: To predict the performance and the power consumption of a design based on module descriptor values, technology constants and in-situ SPICE simulations.

Input: Command file with system model and module descriptors.

Output: Prediction power consumption in watts and performance in GHz.

Reference: Fig 3.8 second loop, looping through all modules in a system.

```
#!/usr/bin/perl
#use warnings;
$args = @ARGV;
if ($args < 1)
{
    print "Improper usage encountered \n \n";
    print "Proper usage - eida.pl <command file name> \n";
    print "Retry , now exiting.....\n";
    exit;
}

$CMD_FILE_NAME = $ARGV[0];
$CMD_FILE_RESULTS = "$CMD_FILE_NAME".".results";
$MODULE_RESULTS = "$CMD_FILE_NAME".".module.results";
##system("awk '\{ if (\$1 != \"\".com_des\") \{print \"\".com_des \" \"\$NEW_PARAMETER_LIST\"} else \{print \$0\} \}' $CMD_FILE_NAME ");
# Use method1 or method 2
#method 1
#system("grep -v \"\".mods_choices\" $CMD_FILE_NAME > tmp_cmd_filename");
#end method 1
#system("rm -f tmp_cmd_filename");
#method 2

open(OUT1, ">$CMD_FILE_RESULTS") or die "Can't create results file : Check folder permissions \n";
open(OUT2, ">$MODULE_RESULTS") or die "Can't create module results file : Check folder permissions \n";
open(CMD_FILE, "$CMD_FILE_NAME") or die "Unable to open command file $CMD_FILE_NAME : Check folder permissions \n";
while(<CMD_FILE>)
{
    $fl_line = $_;
```

```

chomp($fl_line);
@fl_line = split(" ", $fl_line);
#len = @fl_line;
$line_length = @fl_line;

if($line_length == 0)
{
next;
}
else
{
if($fl_line =~ m/~#/)
{
next;
# print "$_\\n";
}
else
{
if($fl_line[0] eq ".mods")
{
$no_modules = $fl_line[1];
next;
}
elseif($fl_line[0] eq ".com_des")
{
@com_des = reverse(@fl_line);
pop(@com_des);
@com_des = reverse(@com_des);
next;
}
elseif($fl_line[0] eq ".pro_des")
{
@pro_des = reverse(@fl_line);
pop(@pro_des);
@pro_des = reverse(@pro_des);
next;
}
else
{
next;
}
}
}
close CMD_FILE;

print OUT1 "# of modules in design -- $no_modules\\n\\n";
#print "Common descriptors -- @com_des\\n";
#print "Process parameters -- @pro_des\\n";

# Initialize hash $module_des
for($i=0; $i < $no_modules; $i++)
{
$jj = $i + 1;
$key = "des_". "$jj";
$module_des{$key} = "description for module $jj entered here";
}

open(CMD_FILE, "$CMD_FILE_NAME") or die "Unable to open command file $CMD_FILE_NAME : Check folder permissions \\n";
while(<CMD_FILE>)
{
$fl_line = $_;
chomp($fl_line);
@fl_line = split(" ", $fl_line);
$line_length = @fl_line;
if($fl_line =~ m/~\\.mods_des/)
{
#print "$_";
$key = $fl_line[1];
$key = "des_". $key;
@fl_line = reverse(@fl_line);
pop(@fl_line);
pop(@fl_line);
@fl_line = reverse(@fl_line);
#print "KEY:$key --- $fl_line\\n";
$module_des{$key} = "@fl_line";
}
if($fl_line =~ m/~\\.mods_choices/)
{
$key = $fl_line[1];
$key = "choices_". $key;
@fl_line = reverse(@fl_line);
pop(@fl_line);
pop(@fl_line);
@fl_line = reverse(@fl_line);
$module_des{$key} = "@fl_line";
}
}
close CMD_FILE;

for($i=0; $i < $no_modules; $i++)
#{
$jj = $i + 1;
#$key = "des_". "$jj";
#print "Module -- $module_des{$key}\\n";
#}

# Assign common descriptor variables from array
$vdd_spec = $com_des[0];
$eta = $com_des[1];
$min_1 = $com_des[2];

```

```

$gate_cap_scaling_factor = $com_des[3];
$wire_cap_scaling_factor = $com_des[4];
$width_sclaing_factor = $com_des[5];
$sold_unit_gate_cap = $com_des[6];
$sold_min_size_fet_ids = $com_des[7];
$new_unit_gate_cap = $com_des[8];
$new_min_size_fet_ids = $com_des[9];
$redesign_power_saving_factor = $com_des[10];
$rc_slowdown_factor = $com_des[11];
$stacking_factor = $com_des[12];
$decap_sensitivity = $com_des[13];
$unit_junc_leakage = $com_des[14];
$unit_gate_leakage = $com_des[15];
$per_unit_width_gate_leakage = $com_des[16];
$typical_hi_vt_ioff = $com_des[17];
$worst_hi_vt_ioff = $com_des[18];
$typical_lo_vt_ioff = $com_des[19];
$worst_lo_vt_ioff = $com_des[20];
# Assign common descriptor variables from array - END

# Assign process parameter variables from array
$alpha = $pro_des[0];
$beta = $pro_des[1];
$gamma = $pro_des[2];
$delta = $pro_des[3];
$epsilon = $pro_des[4];
$fsf_x = $pro_des[5];
$fsf_y = $pro_des[6];
# Assign process parameter variables from array - END

# Test printing of variables
print OUT1 "*****\n";
print OUT1 "***Common descriptors***\n";
print OUT1 "*****\n";
print OUT1 "vdd_spec = $vdd_spec\n";
print OUT1 "eta = $eta\n";
print OUT1 "min_l = $min_l\n";
print OUT1 "gate_cap_scaling_factor = $gate_cap_scaling_factor\n";
print OUT1 "wire_cap_scaling_factor = $wire_cap_scaling_factor\n";
print OUT1 "width_sclaing_factor = $width_sclaing_factor\n";
print OUT1 "old_unit_gate_cap = $old_unit_gate_cap\n";
print OUT1 "old_min_size_fet_ids = $old_min_size_fet_ids\n";
print OUT1 "new_unit_gate_cap = $new_unit_gate_cap\n";
print OUT1 "new_min_size_fet_ids = $new_min_size_fet_ids\n";
print OUT1 "redesign_power_saving_factor = $redesign_power_saving_factor\n";
print OUT1 "rc_slowdown_factor = $rc_slowdown_factor\n";
print OUT1 "stacking_factor = $stacking_factor\n";
print OUT1 "decap_sensitivity = $decap_sensitivity\n";
print OUT1 "unit_junc_leakage = $unit_junc_leakage\n";
print OUT1 "unit_gate_leakage = $unit_gate_leakage\n";
print OUT1 "per_unit_width_gate_leakage = $per_unit_width_gate_leakage\n";
print OUT1 "typical_hi_vt_ioff = $typical_hi_vt_ioff\n";
print OUT1 "worst_hi_vt_ioff = $worst_hi_vt_ioff\n";
print OUT1 "typical_lo_vt_ioff = $typical_lo_vt_ioff\n";
print OUT1 "worst_lo_vt_ioff = $worst_lo_vt_ioff\n";

print OUT1 "*****\n";
print OUT1 "***Process parameters***\n";
print OUT1 "*****\n";
print OUT1 "alpha = $alpha\n";
print OUT1 "beta = $beta\n";
print OUT1 "gamma = $gamma\n";
print OUT1 "delta = $delta\n";
print OUT1 "epsilon = $epsilon\n";
print OUT1 "fsf_x = $fsf_x\n";
print OUT1 "fsf_y = $fsf_y\n";
# Test printing of variables - END

# Calculations begin
$vdd_bump = $vdd_spec * $eta;
$dif = device_improvement_factor
$dif = ($old_unit_gate_cap/$new_unit_gate_cap) * ($new_min_size_fet_ids/$old_min_size_fet_ids);
print OUT1 "*****\n";
print OUT1 "***Intermediate calculations***\n";
print OUT1 "*****Module independent*****\n";
print OUT1 "*****\n";
print OUT1 "Vdd_bump = $vdd_bump\n";
print OUT1 "Device_improvement_factor = $dif\n";

$total_power_leak = 0;
$total_power_dyn = 0;
$total_delay = 0;

for($i=0; $i < $no_modules; $i++)
{
    $j = $i + 1;
    $key = "des_" . $j;
    $key1 = "choices_" . $j;
    $key2 = "results_" . $j;
    #module_des = $module_des{$key};
    #print OUT1 "Module -- $module_des{$key}\n";
    @tmp_array = split(" ", $module_des{$key});
    #print OUT1 "Module -- $module_des{$key}\n";
    @tmp_array1 = split(" ", $module_des{$key1});

    $nads = pop(@tmp_array);
    $if_critical_module = pop(@tmp_array);
    $temperature = pop(@tmp_array);
    $buffer_speed_up_factor = pop(@tmp_array);
    $useful_skew_performance_enhancement = pop(@tmp_array);
    $typical_path_FET_ratio = pop(@tmp_array);

```

```

$wire_buff_cap = pop(@tmp_array);
$ratio_hi_vt_FETS = pop(@tmp_array);
$de_cap_added = pop(@tmp_array);
$clock_gating_factor = pop(@tmp_array);
$average_switching_factor = pop(@tmp_array);
$original_operating_freq = pop(@tmp_array);
$total_NFET_W = pop(@tmp_array);
$total_PFET_W = pop(@tmp_array);
$unit_wire_length_cap = pop(@tmp_array);
$total_wire_length = pop(@tmp_array);
$original_load_cap = pop(@tmp_array);

$use_st = pop(@tmp_array);
$use_abb_rb = pop(@tmp_array);
$use_abb_fb = pop(@tmp_array);
$use_abb = pop(@tmp_array);
$use_dual_vt = pop(@tmp_array);
$vdd_applied = pop(@tmp_array);
$vary_vdd = pop(@tmp_array);

print OUT1 "*****\n";
print OUT1 "***Module $j details***\n";
print OUT1 "*****\n";

print OUT1 "nads = $nads\n";
print OUT1 "if_critical_module = $if_critical_module\n";
print OUT1 "temperature = $temperature\n";
print OUT1 "buffer_speed_up_factor = $buffer_speed_up_factor\n";
print OUT1 "useful_skew_performance_enhancement = $useful_skew_performance_enhancement\n";
print OUT1 "typical_path_FET_ratio = $typical_path_FET_ratio\n";
print OUT1 "wire_buff_cap = $wire_buff_cap\n";
print OUT1 "ratio_hi_vt_FETS = $ratio_hi_vt_FETS\n";
print OUT1 "de_cap_added = $de_cap_added\n";
print OUT1 "clock_gating_factor = $clock_gating_factor\n";
print OUT1 "average_switching_factor = $average_switching_factor\n";
print OUT1 "original_operating_freq = $original_operating_freq\n";
print OUT1 "total_NFET_W = $total_NFET_W\n";
print OUT1 "total_PFET_W = $total_PFET_W\n";
print OUT1 "unit_wire_length_cap = $unit_wire_length_cap\n";
print OUT1 "total_wire_length = $total_wire_length\n";
print OUT1 "original_load_cap = $original_load_cap\n";

print OUT1 "*****\n";
print OUT1 "***Module $j design options***\n";
print OUT1 "*****\n";

@result = @module_evaluate($original_load_cap, $total_wire_length, $unit_wire_length_cap, $total_PFET_W, $total_NFET_W, $original_operating_freq,
$average_switching_factor, $clock_gating_factor, $de_cap_added, $ratio_hi_vt_FETS, $wire_buff_cap, $typical_path_FET_ratio,
$useful_skew_performance_enhancement, $buffer_speed_up_factor, $temperature, $if_critical_module, $nads, $vary_vdd, $vdd_applied, $use_dual_vt,
$use_abb, $use_abb_fb, $use_abb_rb, $use_st,$j);

# Push results into hash
$module_des{$key2} = "@result";
#print OUT1 "Dynamic power for module $module_no = $result[0]\n";
#print OUT1 "Leakage power for module $module_no = $result[1]\n";
#print OUT1 "Performance for module $module_no = $result[2]\n";
$total_power_dyn = $total_power_dyn + $result[0];
$total_power_leak = $total_power_leak + $result[1];

if($if_critical_module == 1)
{
$total_delay = $total_delay + (1/$result[2]);
# Use the line above to calculate total delay if you are modeling at a fub level with multiple fubs in the critical path. Added 06/21/2007

#if( (1/$result[2]) > $total_delay )
#{
#total_delay = (1/$result[2]);
#}
# Use the above if block to calculate total delay if you are modeling above the fub level, here the critical path may be
#contained completely inside a module. We need to do this to make sure the operating frequency for the modules
#are correct when calculating dynamic power. Added 06/21/2007
}

print OUT1 "use_st = $use_st\n";
print OUT1 "use_abb_rb = $use_abb_rb\n";
print OUT1 "use_abb_fb = $use_abb_fb\n";
print OUT1 "use_abb = $use_abb\n";
print OUT1 "use_dual_vt = $use_dual_vt\n";
print OUT1 "vdd_applied = $vdd_applied\n";
print OUT1 "vary_vdd = $vary_vdd\n";
}

sub module_evaluate {
@list = @_;
@list = reverse(@list);

$original_load_cap = pop(@list);
$total_wire_length = pop(@list);
$unit_wire_length_cap = pop(@list);
$total_PFET_W = pop(@list);
$total_NFET_W = pop(@list);
$original_operating_freq = pop(@list);
$average_switching_factor = pop(@list);
$clock_gating_factor = pop(@list);
$de_cap_added = pop(@list);
$ratio_hi_vt_FETS = pop(@list);
$wire_buff_cap = pop(@list);
$typical_path_FET_ratio = pop(@list);
$useful_skew_performance_enhancement = pop(@list);
$buffer_speed_up_factor = pop(@list);
$temperature = pop(@list);
$if_critical_module = pop(@list);

```

```

$ndasp = pop(@list);
$vary_vdd = pop(@list);
$vdd_applied = pop(@list);
$use_dual_vt = pop(@list);
$use_abb = pop(@list);
$use_abb_fb = pop(@list);
$use_abb_rb = pop(@list);
$use_st = pop(@list);
$module_no = pop(@list);

$ratio_lo_vt_FETS = 1 - $ratio_hi_vt_FETS;
$w_total = $total_NFET_W + $total_PFET_W;
$area_fet = $w_total * $width_sclaing_factor * $min_l;
$c_fet = $area_fet * $old_unit_gate_cap * 1e12;
$c_wire = $total_wire_length * $unit_wire_length_cap * 1e6;
if ($original_load_cap != 0)
{
    $original_load_cap = $c_fet + $c_wire;
    # This if statement was added to facilitate user input original load cap value override. Added 06/23/2007
}
$frac_fet = $c_fet / $original_load_cap;
$decapc = $decap_sensitivity * $de_cap_added;
#$decapc = $decap_sensitivity * 1e9 * $de_cap_added * $c_fet;
# need to check why $c_fet is present in the equation.. not sure if that is correct. 09/30/2008
$a_de_cap = $de_cap_added * $c_fet / $new_unit_gate_cap;
$slpzd = ($vdd_bump - $ndasp)/($vdd_bump);
$fsf = ( ($vdd_spec - $fsf_x)/($vdd_bump * $slpzd * ($slpzd - $fsf_x)) )**$fsf_y;
$vdd_new = ($vdd_bump * $slpzd) + $decapc;

#Design choices -- to be changed depending on user choices
if($vary_vdd == 0)
{
    $vdd_applied = $vdd_bump;
}

$a = $vdd_spec / ($vdd_applied + $decapc);
$one_over_a = $vdd_applied / $vdd_spec;
# FSP vdd scaling
$fsf_scaled = $fsf * (($a**(2*$sepilon)) + ($a**($sepilon/2)))/2;
$src_slowdown_factor_prime = $src_slowdown_factor/$buffer_speed_up_factor;

if($use_dual_vt == 0)
{
    $dvtc = 1;
}
else
{
    print "EIDA info: Calculating Dual-Vt correction factor.....\n";
    $dvtc = `perl /home/charles/EIDA_gui/scripts/get_dvtc.pl $total_PFET_W $total_NFET_W $vdd_applied`;
    print "EIDA info: Calculating Dual-Vt correction factor.....Done!\n";
    print OUT1 "Calculated DVTc factor - $dvtc\n";
}

if($use_st == 0)
{
    $stc = 1;
    $stpc = 1;
}
else
{
    print "EIDA info: Calculating Sleep transistor correction factors.....\n";
    $st = `perl /home/charles/EIDA_gui/scripts/get_st_params.pl $total_PFET_W $total_NFET_W $vdd_applied $vdd_spec`;
    @st_params = split(" ", $st);
    $stc = $st_params[0];
    $stpc = $st_params[1];
    print "EIDA info: Calculating Sleep transistor correction factors.....Done!\n";
    print OUT1 "Calculated STC factor - $stc\n";
    print OUT1 "Calculated STPC factor - $stpc\n";
}

if($use_abb == 0)
{
    $abbc = 1;
    $abbpc = 1;
}
else
{
    if($use_abb_fb == 0)
    {
        print "EIDA info: Calculating Active Body Bias correction factors.....\n";
        $abb = `perl /home/charles/EIDA_gui/scripts/get_abb_params.pl $total_PFET_W $total_NFET_W $vdd_applied "RB"`;
        @abb_params = split(" ", $abb);
        $abbc = $abb_params[0];
        $abbpc = $abb_params[1];
        print "EIDA info: Calculating Active Body Bias correction factors.....Done!\n";
        print OUT1 "Calculated ABBC factor - $abbc\n";
        print OUT1 "Calculated ABBCPC factor - $abbpc\n";
    }
    else
    {
        print "EIDA info: Calculating Active Body Bias correction factors.....\n";
        $abb = `perl /home/charles/EIDA_gui/scripts/get_abb_params.pl $total_PFET_W $total_NFET_W $vdd_applied "FB"`;
        @abb_params = split(" ", $abb);
        $abbc = $abb_params[0];
        $abbpc = $abb_params[1];
        print "EIDA info: Calculating Active Body Bias correction factors.....Done!\n";
        print OUT1 "Calculated ABBC factor - $abbc\n";
        print OUT1 "Calculated ABBCPC factor - $abbpc\n";
    }
}

#print "EIDA info: Design choice factor\n";

```

```

# print "EIDA info: DVTC = $dvtc\n";
# print "EIDA info: STC = $stc\n";
# print "EIDA info: STPC = $stpc\n";
# print "EIDA info: ABBC = $abbc\n";
# print "EIDA info: ABBCP = $abbcpc\n";

$psi = $stc * $abbc;

$f_predict_numerator = (($original_operating_freq * $ratio_hi_vt_FETS) + ($original_operating_freq * $ratio_lo_vt_FETS * $dvtc) * $dif * $stpc * $abbcpc * $useful_skew_performance_enhancement);

$f_predict_denominator = ($sf_scaled * $typical_path_FET_ratio) + ((1 - $typical_path_FET_ratio) * $rc_slowdown_factor_prime);
$f_predict = $f_predict_numerator / $f_predict_denominator; # Predicted operating frequency

$f_new = $f_predict * $average_switching_factor * $clock_gating_factor;
$original_load_cap_prime = $original_load_cap * $gate_cap_scaling_factor;
$sc_wire_buff = $wire_buff_cap * $original_load_cap;
$sa_wire_buff = $sc_wire_buff / $new_unit_gate_cap;
$sc_new = ($original_load_cap_prime * (($frac_fet * $gate_cap_scaling_factor) + ((1 - $frac_fet) * $wire_cap_scaling_factor))) + $sc_wire_buff;
$pdyn = $sc_new * $f_new * $redesign_power_saving_factor * (($vdd_new)**2); # Dynamic power

$B = (((one_over_a**$delta) + (one_over_a**(2*$delta)) + (one_over_a**(4*$delta)) + (one_over_a**(6*$delta)))/4 - 1);
$t1 = exp((log($typical_hi_vt_ioff) + log($worst_hi_vt_ioff))/2);
$t2 = exp((log($typical_lo_vt_ioff) + log($worst_lo_vt_ioff))/2);
$t3 = ($w_total * $width_sclaing_factor * 1e6) / $stacking_factor;
$I_off_unscaled = ($t3 * $ratio_hi_vt_FETS * $t1) + ($t3 * $ratio_lo_vt_FETS * $t2);
$I_off = $I_off_unscaled * exp($B * $gamma); # I_off
$unit_junc_leakage_scaled = $unit_junc_leakage * exp($beta * (one_over_a - 1));
$unit_gate_leakage_scaled = $unit_gate_leakage * exp($alpha * (one_over_a - 1));
$bufflc = $sa_wire_buff * $unit_gate_leakage_scaled;
$decaplc = $sa_de_cap * $unit_gate_leakage_scaled;
$I_gate = $area_fet * 1e12 * $unit_gate_leakage_scaled; # I_gate
$area_sd = 4 * $area_fet * 1e12;
$I_junc = $area_sd * $unit_junc_leakage_scaled; # I_junc

$t4 = (($vdd_bump * $I_off) + ($vdd_bump * ($I_gate + $bufflc)) + ($vdd_bump * $I_junc));
$t5 = ($t4) * ((1 - $average_switching_factor) * $psi) + $average_switching_factor;
$pl_leakage = $t5 + ($vdd_bump * $decaplc); # Leakage power

return ($p_dyn, $p_leakage, $f_predict, $module_no);
}

print OUT1 "*****\n";
print OUT1 "*****Module results summary*****\n";
print OUT1 "*****\n";
print OUT1 "[ Module # ][ Dynamic power (W) ][ Leakage power (W) ][ Performance (Hz) ]\n\n";
print OUT2 "*****\n";
print OUT2 "*****Module results summary*****\n";
print OUT2 "*****\n";
print OUT2 "[ Module # ][ Dynamic power (W) ][ Leakage power (W) ][ Performance (Hz) ]\n\n";
for($i=0; $i < $no_modules; $i++)
{
    $j = $i + 1;
    $key = "results_" . $j;
    @tmp = split(" ", $module_des{$key});

    $t0 = int($tmp[0] * 1e6);
    $t0 = $t0 / 1e6;

    $t1 = int($tmp[1] * 1e6);
    $t1 = $t1 / 1e6;

    $t2 = int($tmp[2] * 1e11);
    $t2 = $t2 / 1e20;
    $t2 = int($t2 * 1e6);
    $t2 = $t2 / 1e6;
    $t2 = "t2"."e9";
    print OUT1 "[ $j ] [$t0] [$t1] [$t2] \n";
    print OUT2 "[ $j ] [$t0] [$t1] [$t2] \n";
}

$total_power_dyn = int($total_power_dyn * 1e6);
$total_power_dyn = $total_power_dyn / 1e6;

$total_power_leak = int($total_power_leak * 1e6);
$total_power_leak = $total_power_leak / 1e6;

$total_freq = 1/$total_delay;
$total_freq = int($total_freq * 1e11);
$total_freq = $total_freq / 1e20;
$total_freq = int($total_freq * 1e6);
$total_freq = $total_freq / 1e6;
$total_freq = "total_freq"."e9";

print OUT1 "\n*****\n";
print OUT1 "[ Total ] [$total_power_dyn] [$total_power_leak] [$total_freq] \n";
print OUT1 "*****\n";
print OUT2 "\n*****\n";
print OUT2 "[ Total ] [$total_power_dyn] [$total_power_leak] [$total_freq] \n";
print OUT2 "*****\n";
close OUT1;
close OUT2;
exit;

```

Appendix C

Code for EA Based Design Space Exploration

File name: genetic_new.py

Function: Aspects of the EA based design space exploration including design generation, chromosome mutation, chromosome crossover, EER, IRRR and population maintenance are implemented in this Python program.

Input: Command file with chromosome size, objective goals (minimize or maximize), initial population if any and EA options.

Output: Pareto-front for the EA run.

Reference: Section 4.4.3.

```
#!/usr/bin/env python

from time import * from subprocess import * from random import *
from math import * from glob import * import sys import os
import os.path

#INITIALIZE_POP = 1 #INITIAL_POP_FILE = "./initial_pop_100.txt"
#EIDA_SCRIPT_WRAPPER = "../evaluate_designs_analytical.pl"
#REPLACEMENT_TYPE = 0 ## Type -1 : No replacement ## Type 0
: replaces the chromosome in the current pop which has the min
distance to the new one ## Type 1 : replaces the chromosome in the
current pop which has the min distance with other chromosomes in
the current pop, with the new one ## Type 2 : not implemented yet

##Enter the design choices you want to exclude from the chromosome
#EXCLUDE_DESIGN = 1 #EXCLUDE_DESIGNS = [3,6,7,9,10,11] gobi = []

LF = open
('/home/charles/EIDA_gui/ASP_DAC_2009_GA/genetic/ \
global_log_file.txt','w')
print >>LF, 'Starting the GA code at:', asctime() LF.flush()

if len(sys.argv) <= 1:
    print 'usage : prg def.py' sys.exit()
else:
    mod,ext = os.path.splitext(sys.argv[1]) try:
        defm = (__import__(mod)) xdef =
        getattr(defm,'xdef') ydef = getattr(defm,'ydef')
        NUMITER = getattr(defm,'NUMITER',10000)
        POPSIZE = getattr(defm,'POPSIZE',100)
        NBITS = getattr(defm,'NBITS',8) MUTATION =
        getattr(defm,'MUTATION',0.01) UNIFORMCROSSOVER
        = getattr(defm,'UNIFORMCROSSOVER',True)
        PUREPARETO = getattr(defm,'PUREPARETO',False)
        INF = getattr(defm,'INF',1E308) IGNORE
        = getattr(defm,'IGNORE',0) MINIMIZE =
        getattr(defm,'MINIMIZE',1) MAXIMIZE =
        getattr(defm,'MAXIMIZE',2)

        INITIALIZE_POP = getattr(defm,'INITIALIZE_POP',0)
        INITIAL_POP_FILE = getattr(defm,'INITIAL_POP_FILE',
        './initial_population.txt') EIDA_SCRIPT_WRAPPER
        = getattr(defm,'EIDA_SCRIPT_WRAPPER',
        '../evaluate_designs_analytical.pl
        ") REPLACEMENT_TYPE =
```

```

getattr(defm, 'REPLACEMENT_TYPE', 0) EXCLUDE_DESIGN =
getattr(defm, 'EXCLUDE_DESIGN', 0) EXCLUDE_DESIGNS =
getattr(defm, 'EXCLUDE_DESIGNS', []) REPLACE_DESIGNS =
getattr(defm, 'REPLACE_DESIGNS', []) NO_OF_CHECKPOINTS
= getattr(defm, 'NO_OF_CHECKPOINTS', 10)

except ImportError:
print 'Could not load definition file ', sys.argv[1]
sys.exit()
except AttributeError:
print 'Could not find definitions in module',
mod sys.exit()
except:
print 'Unknown error while loading module',
mod sys.exit()

class Gene:
def __init__(self, name, defval, min, max, bits,
parent=None):
self.name = name self._min = float(min) self._max =
float(max) self._defval = float(defval) if defval <
min or defval > max:
raise ValueError
self._bits = bits[:] self._len = len(bits) self._bval
= self._calcBVal() self._val = self._calcVal()
self._parent = parent

@classmethod def random(cls, name, defval, min, max, len):
bits = [] for i in range(len):
bit = choice([0,1]) bits.append( bit )
g = Gene(name, defval, min, max, bits) return g

def parent(self):
return self._parent

def setParent(self, parent):
self._parent = parent

def name(self):
return self._name

def setName(self, name):
self._name = name

def min(self):
return self._min

def setMin(self, min):
self._min = float(min) self._val = self._calcVal()

def max(self):
return self._max

def setMax(self, max):
self._max = float(max) self._val = self._calcVal()

def defVal(self):
return self._defval

def setDefVal(self, defVal):
if defVal < self._min or defVal > self._max:
raise ValueError
self._defval = float(defVal)

def bits(self):
return self._bits[:]

def setBits(self, bits):
if len(bits) != self._len:
raise ValueError
self._bits = bits[:] self._bval = self._calcBVal()
self._val = self._calcVal()

def val(self):
return float(self._val)

def setVal(self, val):
if val < self._min or val > self._max:
raise ValueError
bval = float(2**self._len) / float(self._max -
self._min) * float(val - self._min) self.setBVal(
bval )

def _calcVal(self):
bval = self._calcBVal() val = float(self._min) val +=
float(self._max - self._min) / float(2**self._len)
* float(bval) return val

def bVal(self):
return self._bval

def setBVal(self, bval):
if bval < 0 or bval >= 2**self._len:
raise ValueError
r = range(self._len) r.reverse() bits = [] for i
in r:
v = 2**i if bval >= v:
bits.append( 1 ) bval = bval - v
else:
bits.append( 0 )
bits.reverse() self._bits = bits self._bval =
self._calcBVal() self._val = self._calcVal()

def _calcBVal(self):
bval = 0 for i in range(self._len):
bit = self._bits[i] bval += bit * 2**i
return float(bval)

def mutated(self, rat):
bits = [] for bit in self._bits:
if random() < rat:
if bit == 1:
bits.append( 0 )
else:
bits.append( 1 )
else:
bits.append( bit )
g = Gene(self._name, self._defval, self._min,
self._max, bits) return g

def __xor__(self, other):
if UNIFORMCROSSOVER:
return self._uniformCrossOver( other )
else:
return self._singleCrossOver( other )

def _uniformCrossOver(self, other):
c1 = [] c2 = [] for p1,p2 in
zip(self._bits, other._bits):
if random() < 0.5:
c1.append(p1) c2.append(p2)
else:
c1.append(p2) c2.append(p1)
g1 = Gene(self._name, self._defval, self._min,
self._max, c1) g2 = Gene(self._name, self._defval,
self._min, self._max, c2) return g1,g2

def _singleCrossOver(self, other):
c1 = [] c2 = [] crossAt =
randint(0, len(self._bits)-1) i = 0 while i < crossAt:
p1 = self._bits[i] p2 = self._bits[i]
c1.append(p1) c2.append(p2) i += 1
while i < len(self._bits):
p1 = self._bits[i] p2 = self._bits[i]
c1.append(p2) c2.append(p1) i += 1
g1 = Gene(self._name, self._defval, self._min,
self._max, c1) g2 = Gene(self._name, self._defval,
self._min, self._max, c2) return g1,g2

def __str__(self):
return self._name + ' ' + str(self._val)

def __repr__(self):
s = '<Gene ' + self._name + ' ' + str(self._val)
for bit in
self._bits:
s += '%d' % bit
s += '>' return s

def __len__(self):
return self._len

def __getitem__(self, key):
if key < 0 or key >= self._len:
raise IndexError
return self._bits[key]

def __setItem__(self, id, val):
if key < 0 or key >= self._len:
raise IndexError
self._bits[key] = val self._bval = self._calcBVal()
self._val = self._calcVal()

def __iter__(self):
return self._bits.__iter__()

def __int__(self):
return self._bval

def __float__(self):
return self._val

def __eq__(self, other):
return self._name == other._name and self._bits ==
other._bits

def __ne__(self, other):
return self._name != other._name or self._bits !=
other._bits

class Fitness:
powerSaved = 0 perfSaved = 0 def __init__(self, name, min,
max, mode, func, parent=None):
self._name = name self._min = min self._max = max
self._mode = mode self._func = func self._val =
None if parent:
self._val = self._eval(parent)
self._parent = parent

def update(self):
self._val = self._eval(self._parent)

def name(self):
return self._name

```



```

def setName(self, name):
    self._name = name

def parent(self):
    return self._parent

def setParent(self, parent):
    self._parent = parent
    self._val = self._eval(parent)

def min(self):
    return self._min

def setMin(self, min):
    self._min = min

def max(self):
    return self._max

def setMax(self, max):
    self._max = max

def mode(self):
    return self._mode

def setMode(self, mode):
    self._mode = mode

def func(self):
    return self._func

def setFunc(self, func):
    self._func = func
    self._val = self._eval(self._parent)

def isValid(self):
    if self._val < self._min or self._val > self._max:
        return False
    return True

def _eval(self, chro):
    cmd = EIDA_SCRIPT_WRAPPER #cmd = './charles.py
    ' arg = self._name + ' ' first = True for gene in
    chro._genes:
    if first:
        arg += str(int(gene.val())) first
        = False
    else:
        arg += ',' + str(int(gene.val()))
        _cmd = cmd + arg print 'calling ' + _cmd, asctime()
        print '>>LF, 'calling ' + _cmd, asctime() LF.flush()
        p = Popen(_cmd, shell=True, stdout=PIPE) #print
        "CMD:" + cmd + arg for l in p.stdout:
        print 'read ', l print '>>LF, 'read', l
        LF.flush() # (power, perf) = l.split(',')
        #if self._name == 'power': # return
        float(power) #elif self._name ==
        'performance': # return float(perf)
        return float(l)

    return None

# for gene in chro._genes: #
    exec(gene.name() + '=' + str(gene.val())) # # return
    eval( self._func )

def __float__(self):
    return float(self._val)

def __sub__(self, other):
    return self._val - other._val

def __str__(self):
    return self._name + ' = ' + str(self._val)

def __repr__(self):
    s = '<Fitness ' + self._name + ' = ' + str(self._val)
    + '>'

def __lt__(self, other):
    if self._mode == IGNORE:
        return False
    elif self._mode == MINIMIZE:
        return self._val > other._val
    elif self._mode == MAXIMIZE:
        return self._val < other._val

def __le__(self, other):
    if self._mode == IGNORE:
        return False
    elif self._mode == MINIMIZE:
        return self._val >= other._val
    elif self._mode == MAXIMIZE:
        return self._val <= other._val

def __eq__(self, other):
    if self._mode == IGNORE:
        return False
    elif self._mode == MINIMIZE or self._mode ==
    MAXIMIZE:
        return self._name == other._name and
        self._val == other._val

```

```

def __ne__(self, other):
    if self._mode == IGNORE:
        return False
    elif self._mode == MINIMIZE or self._mode ==
    MAXIMIZE:
        return self._name != other._name or self._val
        != other._val

def __gt__(self, other):
    if self._mode == IGNORE:
        return False
    elif self._mode == MINIMIZE:
        return self._val < other._val
    elif self._mode == MAXIMIZE:
        return self._val > other._val

def __ge__(self, other):
    if self._mode == IGNORE:
        return False
    elif self._mode == MINIMIZE:
        return self._val <= other._val
    elif self._mode == MAXIMIZE:
        return self._val >= other._val

class Chromosome:
    def __init__(self, parent=None):
        self._genes = [] self._fitness = [] self._parent
        = parent

    def appendGene(self, g):
        self._genes.append( g ) g.setParent( self ) for f
        in self._fitness:
        f.update()

    def appendFitness(self, f):
        self._fitness.append( f ) f.setParent( self )

    def parent(self):
        return self._parent

    def setParent(self, parent):
        self._parent = parent

    def isValid(self):
        for f in self._fitness:
        if not f.isValid():
        #print f.name(), float(f) return False
        if EXCLUDE_DESIGN == 1:
        for g in self._genes:
        #print 'gene =', int(g._val) if
        int(g._val) in EXCLUDE_DESIGNS:
        #return False place =
        EXCLUDE_DESIGNS.index(int(g._val))
        g.setVal(REPLACE_DESIGNS[place])
        #print place, '-->',
        REPLACE_DESIGNS[place]

        return True

    def distance(self, other):
        distance = 0 for ff1, ff2 in zip(self._fitness,
        other._fitness):
        distance = distance + ( ( float(ff1) -
        float(ff2) ) ** 2 )
        distance = sqrt(distance) return distance

    @classmethod def random(cls, xdef, ydef, genelen):
        c = Chromosome() keys = xdef.keys() keys.sort()
        for k in keys:
        name = k defval, min, max = xdef[k] g =
        Gene.random(name, defval, min, max, genelen)
        c.appendGene( g )
        keys = ydef.keys() keys.sort() for k in keys:
        name = k yrange, func, mode = ydef[k] min, max
        = yrange f = Fitness(name, min, max, mode, func)
        c.appendFitness( f )
        return c

    def mutated(self, rat):
        c = Chromosome() for gene in self._genes:
        g = gene.mutated(rat) c.appendGene( g )
        for f in self._fitness:
        f =
        Fitness(f.name(), f.min(), f.max(), f.mode(), f.func())
        c.appendFitness( f )
        return c

    def __xor__(self, other):
        if UNIFORMCROSSOVER:
        return self._uniformCrossOver( other )
        else:
        return self._singleCrossOver( other )

    def _uniformCrossOver(self, other):
        ch1 = Chromosome() ch2 = Chromosome() for p1, p2
        in zip(self._genes, other._genes):
        c1, c2 = p1 ^ p2 ch1.appendGene( c1 )
        ch2.appendGene( c2 )
        for f in self._fitness:
        f1 =

```

```

Fitness(f.name(),f.min(),f.max(),f.mode(),f.func())
f2 =
Fitness(f.name(),f.min(),f.max(),f.mode(),f.func())
ch1.appendFitness( f1 ) ch2.appendFitness(
f2 )
return ch1,ch2

def _singleCrossOver(self, other):
ch1 = Chromosome() ch2 = Chromosome() crossAt =
randint(0,len(self._genes)-1) i = 0 while i <
crossAt:
p1 = self._genes[i] p2 = self._genes[i]
c1 = Gene(p1._name, p1._defval, p1._min,
p1._max, p1._bits[:]) c2 = Gene(p2._name,
p2._defval, p2._min, p2._max, p2._bits[:])
ch1.appendGene( c1 ) ch2.appendGene( c2 )
i += 1

p1 = self._genes[i] p2 = self._genes[i] c1,c2 = p1 -
p2 ch1.appendGene( c1 ) ch2.appendGene( c2 ) i += 1

while i < len(self._genes):
p1 = self._genes[i] p2 = self._genes[i]
c1 = Gene(p1._name, p1._defval, p1._min,
p1._max, p1._bits[:]) c2 = Gene(p2._name,
p2._defval, p2._min, p2._max, p2._bits[:])
ch1.appendGene( c2 ) ch2.appendGene( c1 )
i += 1

for f in self._fitness:
f1 =
Fitness(f.name(),f.min(),f.max(),f.mode(),f.func())
f2 =
Fitness(f.name(),f.min(),f.max(),f.mode(),f.func())
ch1.appendFitness( f1 ) ch2.appendFitness(
f2 )
return ch1,ch2

def __str__(self):
s = 'Chromosome:\n' genes = {} for g in self._genes:
genes[ g.name() ] = g
keys = genes.keys() keys.sort() for key in keys:
g = genes[key] s += '%10s %9.5f\n' %
(key,float(g))
s += '-----\n' fs = {} for f in
self._fitness:
fs[f.name()] = f
keys = fs.keys() keys.sort() for key in keys:
f = fs[key] s += '%10s %9.5f\n' %
(key,float(f))
return s

def __repr__(self):
s = '<Chromosome ' for g in self._genes:
s += repr(g)
for f in self._fitness:
s += repr(f)
return s

def __len__(self):
return len(self._genes)

def __getitem__(self, key):
if key < 0 or key >= len(self._genes):
raise IndexError
return self._genes[key]

def __setitem__(self, id, val):
if key < 0 or key >= len(self._genes):
raise IndexError
self._genes[key] = val val.setParent( self ) for
f in self._fitness:
f.update()

def __contains__(self, item):
for g in self._genes:
if g == item:
return True
return False

def __iter__(self):
return self._genes.__iter__()

def __lt__(self, other):
for f1,f2 in zip(self._fitness,other._fitness):
if f1 >= f2:
return False
return True

def __le__(self, other):
for f1,f2 in zip(self._fitness,other._fitness):
if f1 > f2:
return False
return True

def __eq__(self, other):
for g1,g2 in zip(self._genes,other._genes):
if g1 != g2:
return False
return True

def __ne__(self, other):
for g1,g2 in zip(self._genes,other._genes):
if g1 != g2:
return True
return False

def __gt__(self, other):
for f1,f2 in zip(self._fitness,other._fitness):
if f1 <= f2:
return False
return True

def __ge__(self, other):
for f1,f2 in zip(self._fitness,other._fitness):
if f1 < f2:
return False
return True

def writeGnuplot(self,file):
fd = open(file,'w') for g in self:
print >>fd,int(float(g)),
print >>fd,':', for f in self._fitness:
print >>fd,float(f),
print >>fd fd.close()

class Population:
def __init__(self, size, parent=None):
self._chros = [] self._size = size self._parent
= parent

def appendChro(self, chro):
if not chro.isValid():
return False
if chro in self:
return False

for c in self._chros[:]:
if c > chro:
return False
isoutsideparpoint = 1 minchro_distance = 10000000000

for c in self._chros[:]:
if c < chro:
self._chros.remove( c )
isoutsideparpoint = 0 if not
PUREPARETO:
break

if isoutsideparpoint == 1:
if REPLACEMENT_TYPE == 0:
for c in self._chros[:]: # for
fff in c._fitness: # if
(fff.name() == str("power")):
# currchropwr
= float(fff) # if
(fff.name() == str("performance")): #
currchroper = float(fff)
chro_distance =
c.distance(chro)
if chro_distance <
minchro_distance:
minchro_distance =
chro_distance
#print >>LF, "Chromosomal
distance: " chro_distance
print >>LF, "Min chromosomal
distance: ", minchro_distance

for c in self._chros[:]:
chro_distance =
c.distance(chro)
if chro_distance ==
minchro_distance:
self._chros.remove( c
) self._chros.append(
chro) print >>LF,
"This is an outside
pareto point -
inserted into current
population_set type 0
"

if REPLACEMENT_TYPE == 1:
for _t in range( len(self._chros)):
for _tt in range
(len(self._chros)):
chro_distance =
self._chros[_t].distance(self._chros[_tt])
#print >>LF,
"Chromosomal
distance: ",
chro_distance if
self._chros[_t]
!= self._chros[_tt]:
if
chro_distance
<
minchro_distance:
minchro_distance
=
chro_distance

```

```

#for c in self._chros[:]:
#for c1 in self._chros[:]:
#chro_distance =
c1.distance(c) #print
>>LF, "Chromosomal
distance: ",
chro_distance #if
c != c1:
#if
chro_distance
<
minchro_distance:
#minchro_distance
=
chro_distance

print >>LF, "Min chromosomal
distance: ",minchro_distance print
>>LF, "----"

#for c in self._chros[:]:
#for c1 in self._chros[:]:
#chro_distance =
c1.distance(c) #print
>>LF, "Chromosomal
distance: ",
chro_distance #if
c != c1:
#if
chro_distance
==
minchro_distance:
#self._chros.remove(
c1 )
#self._chros.append(
chro)
#print
>>LF,
"This
is an
outside
pareto
point
-
inserted
into
current
population_set
type
1"
#break

for _t in range (len(self._chros)):
for _tt in range
(len(self._chros)):
chro_distance =
self._chros[_t].distance(self._chros[_tt])
#print >>LF,
"Chromosomal
distance: ",
chro_distance if
self._chros[_t]
!= self._chros[_tt]:
if
chro_distance
==
minchro_distance:
self._chros.remove(
self._chros[_tt]
)
self._chros.append(
chro)
print
>>LF,
"This
is an
outside
pareto
point
-
inserted
into
current
population_set
type
1"
break

#print len(self._chros) if len(self._chros) <
self._size:
self._chros.append( chro ) chro.setParent(
self ) return True
return False

def parent(self):
return self._parent

def setParent(self, parent):
self._parent = parent

@classmethod def random(cls, xdef, ydef, popSize, geneLen,
chros=[]):

p = Population(popSize) for chro in chros:
p._chros.append( chro ) chro.setParent( p )
while len(p) < popSize:
c = Chromosome.random(xdef, ydef, geneLen)
if not c.isValid():
print 'notValid' print >>LF,
'notValid' LF.flush() continue
if c in p:
print 'already in' print >>LF,
'already in' LF.flush() continue
add = True for chro in p._chros[:]:
if chro > c:
print 'dominated' print >>LF,
'dominated' LF.flush()
add = False break
if not add:
continue
if PUREPARETO:
for chro in p._chros[:]:
if chro < c:
p._chros.remove(
chro )
print 'size',len(p),popSize print >>LF,
'size',len(p),popSize print >>LF,
p LF.flush() p._chros.append( c )
sys.stdout.flush() c.setParent( p )
print >>LF, p LF.flush() return p

def __len__(self):
return len(self._chros)

def __getitem__(self, key):
if key < 0 or key >= len(self._chros):
raise IndexError
return self._chros[key]

def __setitem__(self, key, value):
if key < 0 or key >= len(self._chros):
raise IndexError
self._chros[key] = value value.setParent( self )

def __delitem__(self, key):
if key < 0 or key >= len(self._chros):
raise IndexError
del self._chros[key]

def __iter__(self):
return self._chros.__iter__()

def __contains__(self, item):
for c in self._chros:
if c == item:
return True
return False

def __str__(self):
s = 'Population:\n' c = self._chros[0] genes = {}
for g in c._genes:
genes[g.name()] = g
fs = {} for f in c._fitness:
fs[f.name()] = f
keysg = genes.keys() keysg.sort() keysf = fs.keys()
keysf.sort() for k in keysg:
s += '%9s' % k
s += '|' for k in keysf:
s += '%9s' % k
s += ' '*(9*(len(keysg)+len(keysf))+1) s +=
'\n' for c in self._chros:
genes = {} for g in c._genes:
genes[g.name()] = g
for k in keysg:
g = genes[k] s += '%9.3f' % float(g)
fs = {} s += '|' for f in c._fitness:
fs[f.name()] = f
for k in keysf:
f = fs[k] s += '%9.3f' % float(f)
s += '\n'
return s

def writeGnuplot(self,file):
fd = open(file,'w') for c in self._chros:
for g in c:
print >>fd,int(float(g)),
print >>fd,':', for f in c._fitness:
print >>fd,float(f),
print >>fd
fd.close()

def writeGobi(self, file, org):
fd = open(file,'w') print >>fd,'<?xml
version="1.0"?>' print >>fd,'<!DOCTYPE ggobidata
SYSTEM "ggobi.dtd">' print >>fd,'<ggobidata>'
print >>fd,'<data name="GA optimization">' print
>>fd,'<description>' print >>fd,'This data is the
entire GA population'' print >>fd,'</description>'

varCount = len(self._chros[0]._fitness)
varCount += len(self._chros[0]._genes) + 2 print
>>fd,'<variables count="%d">' % varCount for f in
self._chros[0]._fitness:
print >>fd,'<realvariable name="%s"' %

```

```

f.name(), if f.min() > -INF:
print >>fd,'min="%f"' % f.min(),
if f.max() < INF:
print >>fd,'max="%f"' % f.max(),
print >>fd,'>'
for g in self._chros[0]._genes:
print >>fd,'<realvariable name="%s" min="%f"
max="%f"/>' % (
g.name(),g.min(),g.max())
print >>fd,'<categoricalvariable name="category">'
print >>fd,'<levels count="4">' print >>fd,'<level
value="0">original</level>' print >>fd,'<level
value="1">dominates</level>' print >>fd,'<level
value="2">normal</level>' print >>fd,'</levels>'
print >>fd,'</categoricalvariable>' print
>>fd,'<category name="pareto">' print
>>fd,'<levels count="2">' print >>fd,'<level
value="0">normal</level>' print >>fd,'<level
value="1">pareto</level>' print >>fd,'</levels>'
print >>fd,'</categoricalvariable>' print
>>fd,'</variables>'

recCount = len(self)+1 print >>fd,'<records
count="%i">' % recCount print >>fd,'<record color="2"
glyph="fc 3">', for f in org._fitness:
print >>fd,float(f),
for g in org._genes:
print >>fd,float(g),
print >>fd,'0', print >>fd,'0', print
>>fd,'</record>'

for c in self._chros:
if c > org:
dominates = True
else:
dominates = False
pareto = True for o in self._chros:
if o == c:
continue
if c < o:
pareto = False break
print >>fd,'<record>', if dominates:
print >>fd,'color="4"',
else:
print >>fd,'color="8"',
if pareto:
print >>fd,'glyph="plus 3"',
else:
print >>fd,'glyph="fc 3"',
print >>fd,'>'

for f in c._fitness:
print >>fd,float(f),
for g in c._genes:
print >>fd,float(g),
if dominates:
print >>fd,'1',
else:
print >>fd,'2',
if pareto:
print >>fd,'1',
else:
print >>fd,'0',
print >>fd,'</record>'
print >>fd,'</records>'

print >>fd,'</data>' print >>fd,'</ggobidata>'
fd.close()

class Genetic:
def __init__(self, pop):
self._pop = pop pop.setParent( self )

def evolve(self, numIter, org):
i = 0 while i < numIter:
print 'iteration',i print >>LF, 'iteration',i
LF.flush() pop = self._pop parents =
sample(pop,2) p1 = parents[0] p2 = parents[1]
c1,c2 = p1 ~ p2 res = pop.appendChro( c1 )
if c1 > org:
print print c1 gobi.append( c1 )
if res:
print 'X', sys.stdout.flush()
res = pop.appendChro( c2 ) if c2 > org:
print print c2 gobi.append( c2 )
if res:
print 'X', sys.stdout.flush()
i += 1 yield i parents = sample(pop,i) p =
parents[0] c = p.mutated(MUTATION) res =
pop.appendChro( c ) if c > org:
print print c gobi.append( c )
if res:
print 'M', sys.stdout.flush()
#print >>LF, 'Iteration# ',i, ' population'
#print >>LF, pop #LF.flush() i += 1 yield i

raise StopIteration

if __name__ == '__main__':
for f in glob('*.xml'):
os.unlink(f)

# if os.path.isfile
("/home/charles/EIDA_gui/ASP_DAC_2009_GA/genetic/ \
previous_run_chromosome_new.txt"):
#os.unlink("/home/charles/EIDA_gui/ASP_DAC_2009_GA/\
#genetic/previous_run_chromosome_new.txt")

if INITIALIZE_POP == 1:
init = []

for line in open(INITIAL_POP_FILE):
a = line.split() print >>LF,
'Chromosome initialized from file'
print >>LF, a LF.flush() org =
Chromosome() keys = xdef.keys()
keys.sort() iiii = 0 for k in keys:
name = k val,min,max
= xdef[k] g =
Gene(name,val,min,max,[0
for i in range(NBITS)])
#print float(a[iiii]), iiii,
a g.setVal( float(a[iiii]) )
org.appendGene( g ) iiii += 1
keys = ydef.keys() keys.sort()
for k in keys:
name = k yrange,func,mode =
ydef[k] min,max = yrange f =
Fitness(name,min,max,mode,func)
org.appendFitness( f )

init.append( org )
org.writeGnuplot('orig.dat') print print 'Original:'
print >>LF, 'Original:' LF.flush() print org print
'Initializing and generating initial population'
print >>LF, 'Initializing and generating initial
population'

else:
org = Chromosome() keys = xdef.keys() keys.sort()
for k in keys:
name = k val,min,max = xdef[k]
g = Gene(name,val,min,max,[0 for i
in range(NBITS)]) g.setVal( val )
org.appendGene( g )

keys = ydef.keys() keys.sort() for k in keys:
name = k yrange,func,mode = ydef[k] min,max =
yrange f = Fitness(name,min,max,mode,func)
org.appendFitness( f )
org.writeGnuplot('orig.dat') print print 'Original:'
print >>LF, 'Original:' LF.flush() print org
print 'Generating initial population' print >>LF,
'Generating initial population'

if INITIALIZE_POP == 1:
pop = Population.random( xdef, ydef, POPSIZE,
NBITS, init)
else:
pop = Population.random( xdef, ydef, POPSIZE, NBITS,
[org])

print 'complete' print >>LF, 'complete' LF.flush()
pop.writeGnuplot('init.dat') pop.writeGobi('init.xml', org)

ga = Genetic( pop ) step = int(NUMITER/NO_OF_CHECKPOINTS)
for i in ga.evolve(NUMITER,org):
if i % step == 0:
pop.writeGobi(str(i)+'_xml',org) print
>>LF, 'Iteration# ',i, ' population' print
>>LF, str(pop) LF.flush() # if os.path.isfile
("/home/charles/EIDA_gui/ASP_DAC_2009_GA/genetic/global_log_file.txt"):
#os.unlink("/home/charles/EIDA_gui/ASP_DAC_2009_GA/genetic/global_log_file.txt")
pop.writeGnuplot('final.dat') pop.writeGobi('final.xml', org)

print print ' ', w = 4 for f in org._fitness:
print '%10s' % f.name(), w += 10
for g in org._genes:
print '%10s' % g.name(), w += 10
print print 'org:', print >>LF, 'org:', for f in
org._fitness:
print '%10.4f' % float(f), print >>LF,
'%10.4f' % float(f),
for g in org._genes:
print '%10d' % int(float(g)), print >>LF,
'%10d' % int(float(g)),
print print >>LF print ' '*w print >>LF, ' '*w for chro
in pop:
if chro > org:
print 'imp:', print >>LF, 'imp:',
else:
print 'par:', print >>LF, 'par:',
for f in chro._fitness:
print '%10.4f' % float(f), print >>LF,
'%10.4f' % float(f),
for g in chro._genes:
print '%10d' % int(float(g)), print
>>LF, '%10d' % int(float(g)),
print print >>LF

#gpob = Population( len(gobi) ) #for c in gobi: #
gpob._chros.append( c ) #gpob.writeGobi('inter.xml', org)
LF.close()

```

Appendix D

GUI Implementation of EIDA

D.1 Screen Captures of the GUI

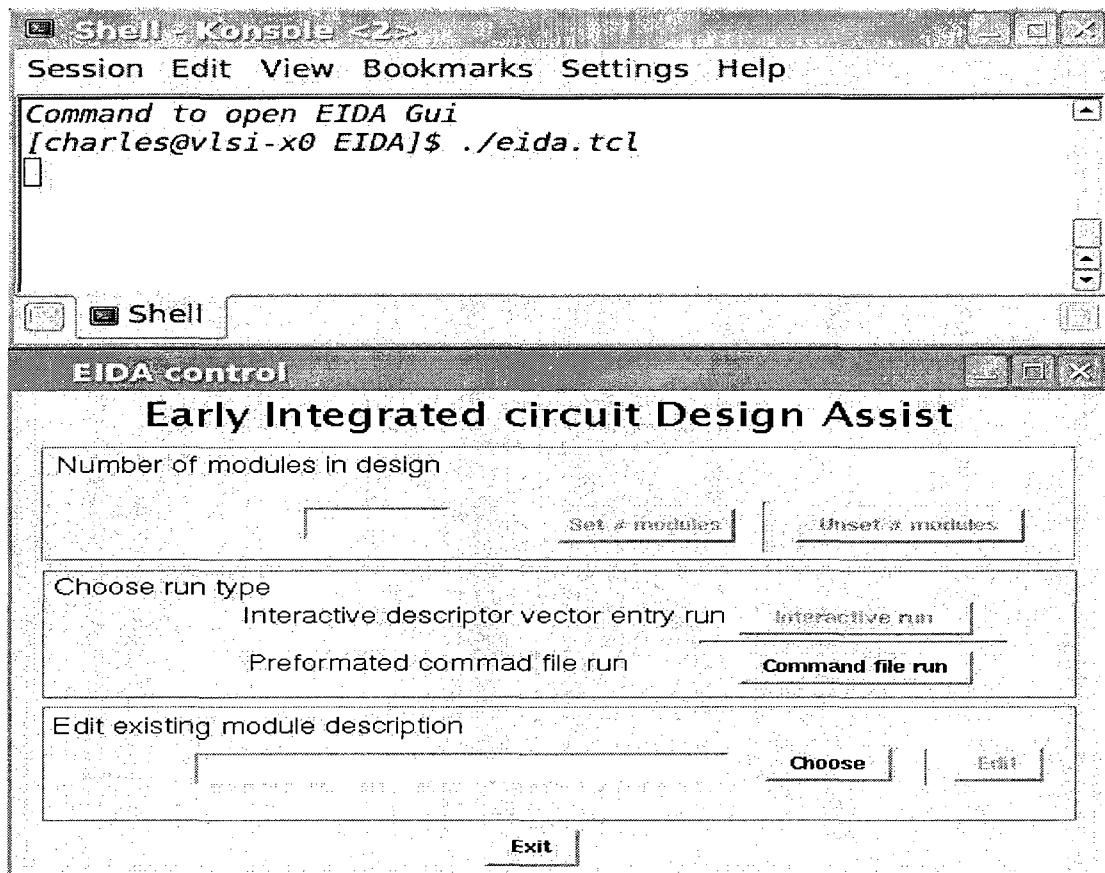


Figure D.1: Invoking the EIDA tool's GUI from the command prompt

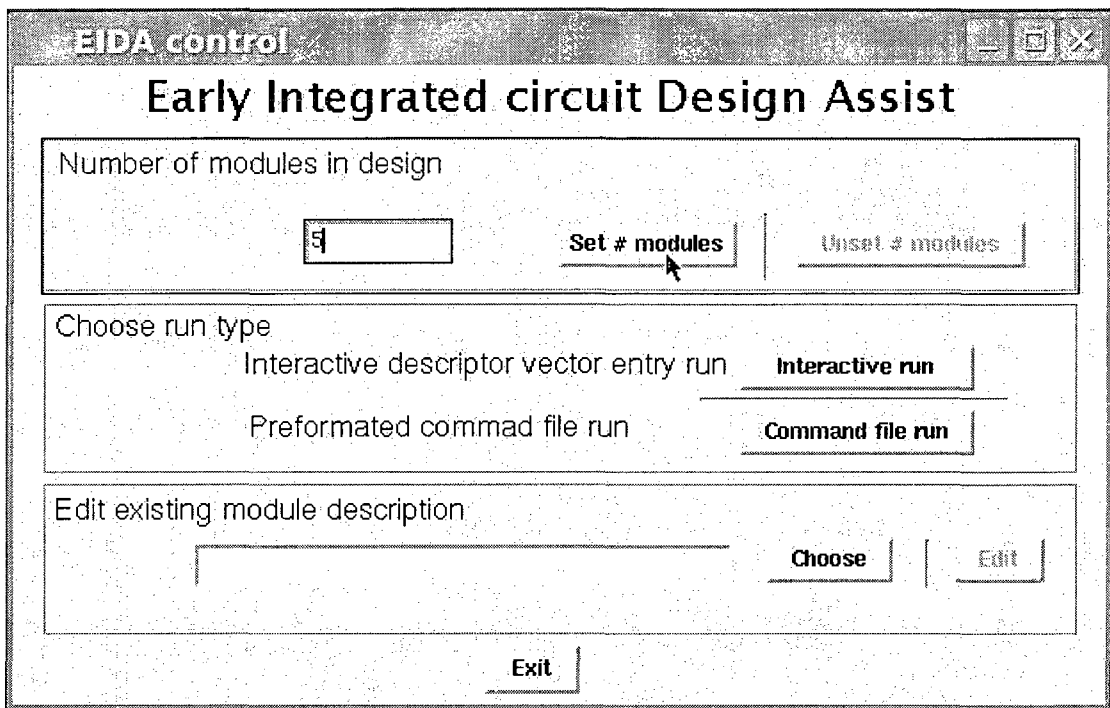


Figure D.2: Entering the number of modules in the design

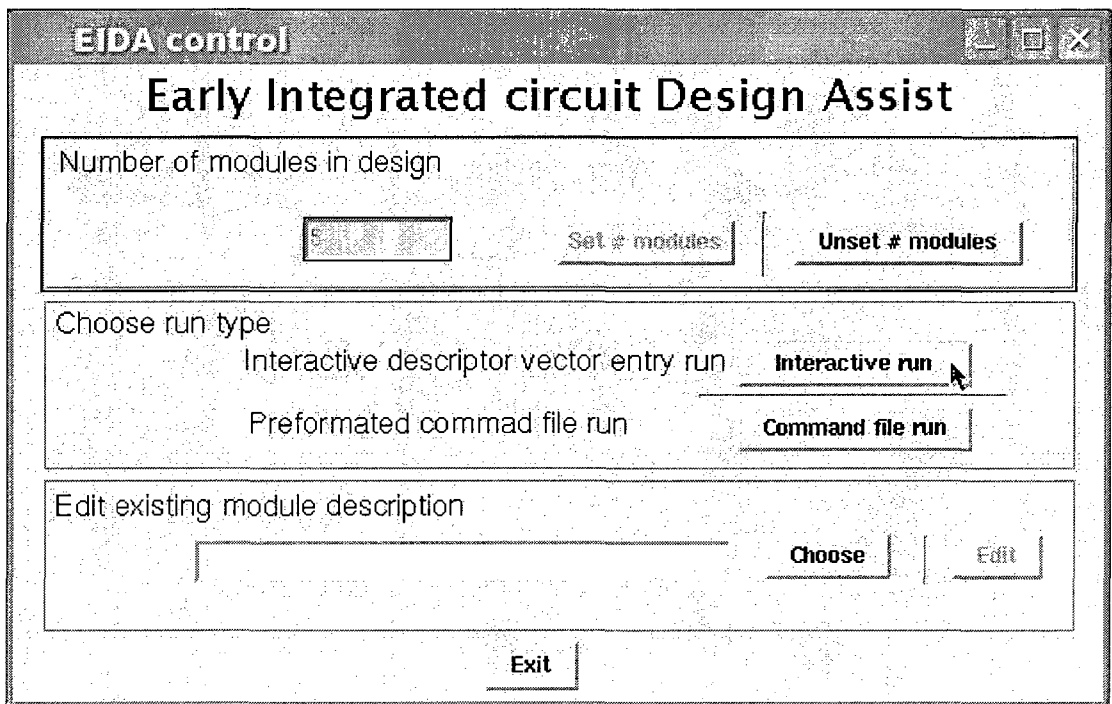


Figure D.3: Confirming the number of modules and choosing interactive run type

Common descriptors

Descriptors common to all modules

Supply voltage of new process (V)	1.2
Package IR drop factor (eta)	0.95
$v_{ds_bump} \times 1 / v_{ds_supply} \times eta$	1.14
Min length in new process (m)	32e-9
Gate cap scaling factor	1.2
Wire cap scaling factor	1.2
Gate width scaling factor	1.1
Original unit area gate cap (F/um2)	1e-12
Original min W/L drain current (A)	15e-6
New chip's unit area gate cap (F/um2)	0.7e-12
New chip's min W/L drain current (A)	30e-6
Library redesign power saving factor	1
RC interconnect slowdown factor	1
Average statcking factor for new design	0.8
Unit decap sensitivity (V/nF)	0.01

New process's leakage current estimates
from foundry or SPICE simulations

Unit junction area leakage (A/um2)	1e-7
Unit gate area leakage (A/um2)	1e-5
Unit linear gate leakage current (A/um)	1e-7
Unit linear, typical I_off for high Vt FETs (A/um)	100e-8
Unit linear, worst I_off for high Vt FETs (A/um)	350e-8
Unit linear, typical I_off for low Vt FETs (A/um)	250e-8
Unit linear, worst I_off for low Vt FETs (A/um)	800e-8

Clear All
Save and proceed

Figure D.4: Entering the process dependent descriptors

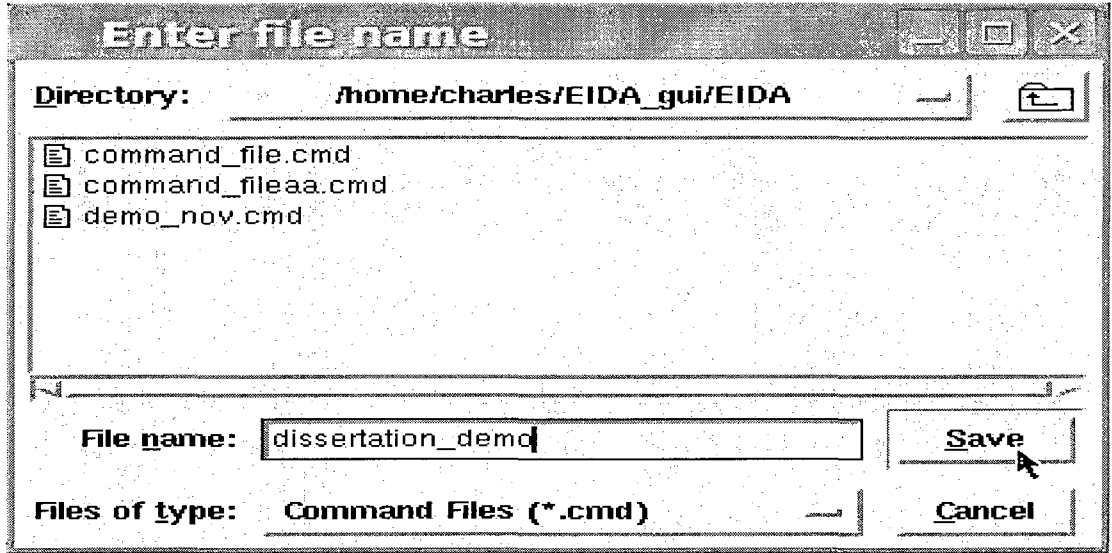


Figure D.5: Entering the command file name to save the entered data

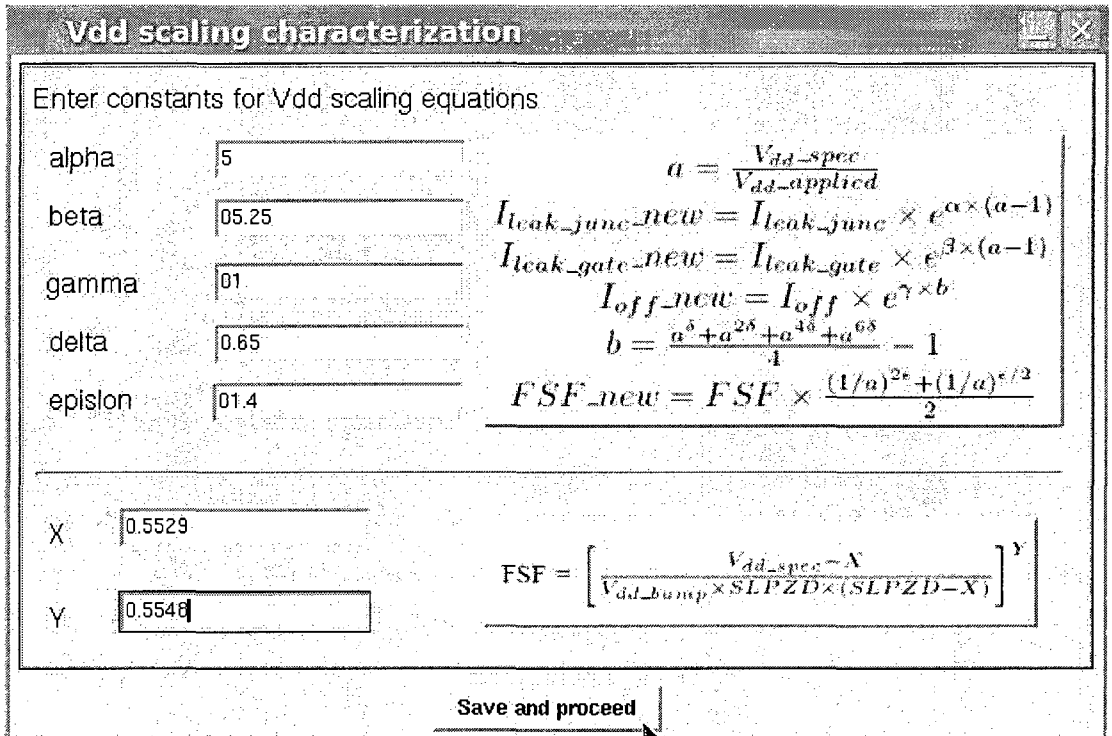


Figure D.6: Entering VDD scaling descriptors

Individual module descriptors

Individual module descriptors

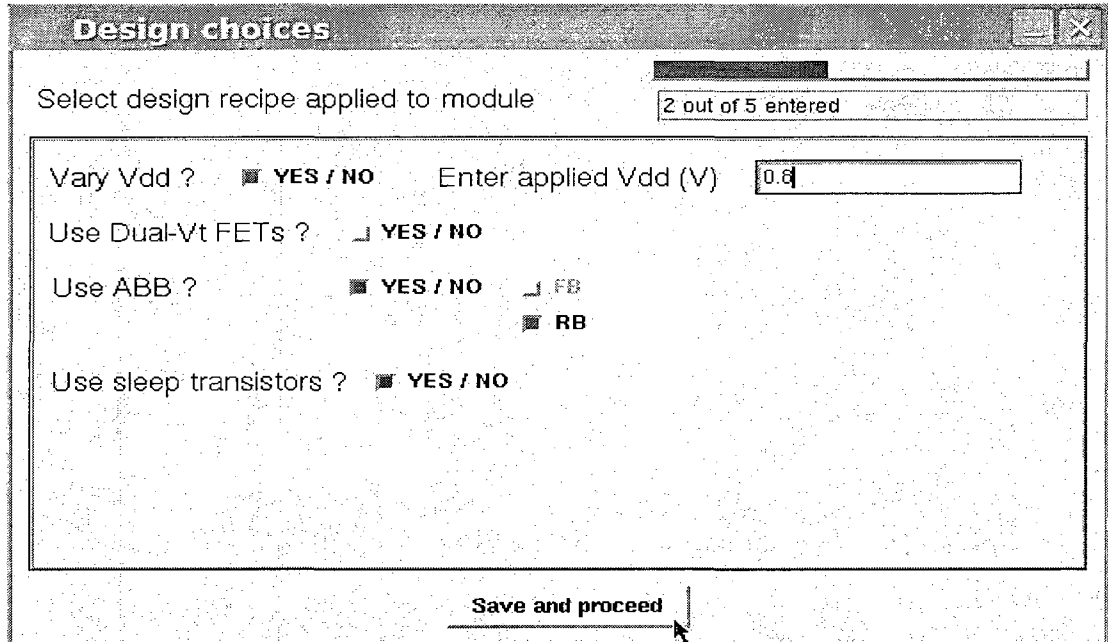
2 out of 5 entered

Original load capacitance (F)	2.05e-12
Total wire length (m)	6.94
Unit length wire cap, all metal layers (F)	2.5e-17
Total PFET width (m)	0.46
Total NFET width (m)	0.31
Original operating frequency (Hz)	2258064516
Delay-(1/freq)	0
Average switching factor	0.2
Clock gating factor	0.9
Decap added (% of total FET width)	0.05
Ratio of Hi-to-Lo Vt FETS	0.85
Ratio of Lo-to-Hi Vt FETS	0.15
Wire buffers (% of total FET width)	0.1
Typical path FET ratio	0.5
Useful skew perf enhancement	01
Interconnect speedup due to buffers	01
Temperature of operation	25
Critical module?	<input checked="" type="checkbox"/> YES / NO
NADSP - Vdd droop (% of Vdd)	0.2

Clear All

Save and proceed

Figure D.7: Entering modules descriptors for each module in the system



Design choices

Select design recipe applied to module 2 out of 5 entered

Vary Vdd ? ☒ YES / NO Enter applied Vdd (V)

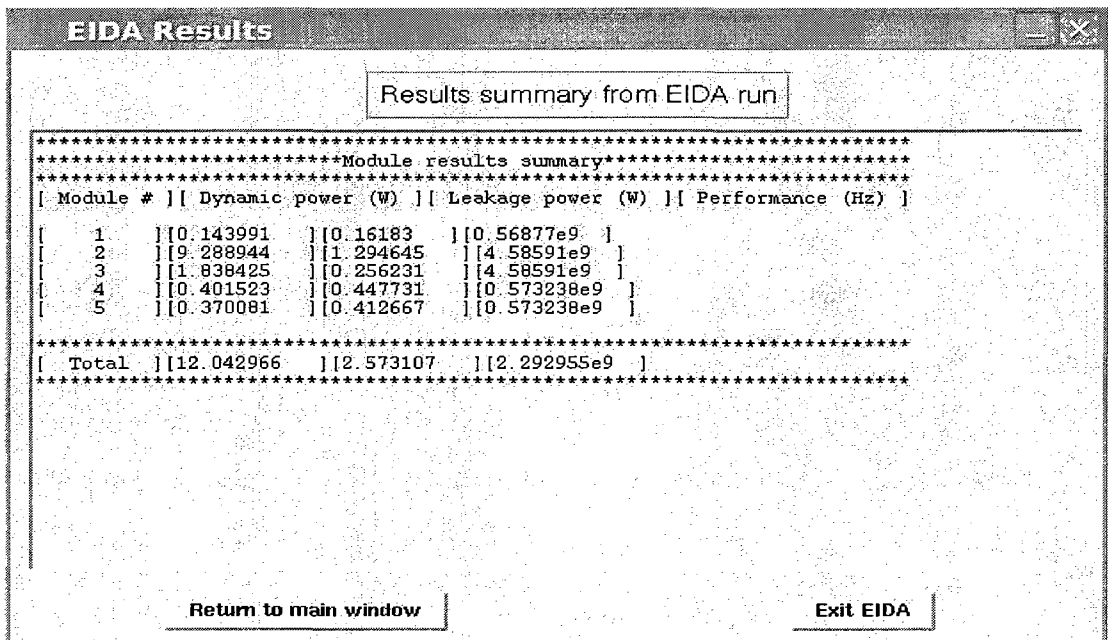
Use Dual-Vt FETs ? ☐ YES / NO

Use ABB ? ☒ YES / NO ☐ FB ☒ RB

Use sleep transistors ? ☒ YES / NO

Save and proceed

Figure D.8: Entering design choices for each module in the system



EIDA Results

Results summary from EIDA run

```

*****
*****Module results summary*****
*****
[ Module # ][ Dynamic power (W) ][ Leakage power (W) ][ Performance (Hz) ]
[ 1 ][ 0.143991 ][ 0.16183 ][ 0.56877e9 ]
[ 2 ][ 9.288944 ][ 1.294645 ][ 4.58591e9 ]
[ 3 ][ 1.838425 ][ 0.256231 ][ 4.58591e9 ]
[ 4 ][ 0.401523 ][ 0.447731 ][ 0.573238e9 ]
[ 5 ][ 0.370081 ][ 0.412667 ][ 0.573238e9 ]

[ Total ][ 12.042966 ][ 2.573107 ][ 2.292955e9 ]
*****

```

Return to main window
Exit EIDA

Figure D.9: System power and performance for the chosen module design choices

```
Shell - Konsole <2>
Session Edit View Bookmarks Settings Help

Using EIDA from command prompt with command file
[charles@vlsi-x0 EIDA]$ /home/charles/EIDA_gui/scripts/eida.pl ./dissertation_demo.cmd
[charles@vlsi-x0 EIDA]$ more ./dissertation_demo.cmd.module.results
*****
*****Module results summary*****
*****
[ Module # ][ Dynamic power (W) ][ Leakage power (W) ][ Performance (Hz) ]

[ 1 ][0.143991 ][0.16183 ][0.56877e9 ]
[ 2 ][9.288944 ][1.294645 ][4.58591e9 ]
[ 3 ][1.838425 ][0.256231 ][4.58591e9 ]
[ 4 ][0.401523 ][0.447731 ][0.573238e9 ]
[ 5 ][0.370081 ][0.412667 ][0.573238e9 ]

*****
[ Total ][12.042966 ][2.573107 ][2.292955e9 ]
*****
[charles@vlsi-x0 EIDA]$
```

Figure D.10: EIDA tool used in batch mode from the command prompt. A command file corresponding to the modules in the system and their respective design choices has to be created prior to invoking EIDA from the command prompt.

D.2 TCL/Tk Code for the GUI

File name: eida.tcl

Function: Implements the GUI interface for the EIDA design framework.

Input: Optional for interactive run. A command file corresponding to the modules in the system and their respective design choices is required for the command file run.

Output: System (and module) performance and power consumption estimates.

Reference: Section D.1.

```
exec wish "$0" "$@"
append auto_path
"/home/charles/my_softwares/AvtiveTcl/lib/tcl8.4
/home/charles/my_softwares/AvtiveTcl/lib
/home/charles/my_softwares/AvtiveTcl/lib/tklib0.4
/home/charles/my_softwares/AvtiveTcl/lib/tcllib1.8
/home/charles/my_softwares/AvtiveTcl/lib/tk8.4"
if {[info exists
vTcl(sourcing)]} {
    catch {package require bogus-package-name}
    set packageNames {package
names}
    package require BWidget
    switch $tcl_platform(platform) {
    windows {
    }
    default {
        option add *ScrolledWindow.size 14
    }
    }

    package require Tk
    switch $tcl_platform(platform) {
    windows {
        option add *Button.padY 0
    }
    default {
        option add *Scrollbar.width 10
        option add
        *Scrollbar.highlightThickness 0
        option add
        *Scrollbar.elementBorderWidth 2
        option add
        *Scrollbar.borderWidth 2
    }
    }

    if {[info exist vTcl(sourcing)]} {
        proc ::vTcl:rename {name} {
            regsub -all "\\" $name "_" ret
            regsub -all "\\" $ret "_" ret
            regsub -all "/" $ret "_" ret
            regsub -all ":" $ret "_" ret
            return [string tolower $ret]
        }

        proc ::vTcl:image:create_new_image {filename {description {no
description}}} {type {}} {data {}} {
            if {[info exists ::vTcl(images,files)]} {
                if {[lsearch -exact $::vTcl(images,files) $filename] > -1} {
                    return
                }
            }
            if {[info exists ::vTcl(sourcing)]} && [string length $data] >
0 {
                set object [image create [vTcl:image:get_creation_type $filename]
-data $data]
            } else {
                if {[file exists $filename]} {
                    set script [file dirname [info script]]
                    set filename [file
                    join $script [file tail $filename] ]
                }
                if {[file exists $filename]} {
                    set description "file not found!"
                    set object [image create
                    photo -width 1 -height 1]
                } else {
                    set object [image create [vTcl:image:get_creation_type
$filename] -file $filename]
                }
            }
            set reference [vTcl:rename $filename]
            set
            ::vTcl(images,$reference,image) $object
            set
            ::vTcl(images,$reference,description) $description

            set ::vTcl(images,$reference,type) $type
            set
            ::vTcl(images,filename,$object) $filename
            lappend
            ::vTcl(images,files) $filename
            lappend ::vTcl(images,$type) $object

            return $object
        }

        proc ::vTcl:image:get_image {filename} {
            set reference [vTcl:rename $filename]
            if {[info exists
            ::vTcl(images,$reference,image)]} {
                set imageTail [file tail $filename]
                foreach oneFile
                $::vTcl(images,files) {
                    if {[file tail $oneFile] == $imageTail} {
                        set reference [vTcl:rename $oneFile]
                        break
                    }
                }
                return $::vTcl(images,$reference,image)
            }
            proc ::vTcl:image:get_creation_type {filename} {
                switch [string tolower [file extension $filename]] {
                .ppm -
                .jpg -
                .bmp -
                .gif {return photo}
                .xbm {return
                bitmap}
                default {return photo}
                }
            }
            foreach img {
            } {
                eval set _file [lindex $img 0]
                vTcl:image:create_new_image\
                $_file [lindex $img 1] [lindex $img 2] [lindex $img 3]
            }
        }
    }
}
```

```

}
catch {package require Img}
foreach img {
  {(file join / home charles EIDA_gui images eqn1.s.jpg)} {user
image} user {}
  {(file join / home charles EIDA_gui images
eqn2.s.jpg)} {user image} user {}
  {(file join / home charles
EIDA_gui images hour_glass.jpg)} {user image} user {}
} {
  eval set _file [lindex $img 0]
  vTcl:image:create_new_image\
  $ _file [lindex $img 1] [lindex $img 2] [lindex $img 3]
}
if {[info exist vTcl(sourcing)]} {
  set vTcl(fonts,counter) 0
  proc
::vTcl:font:add_font {font_descr font_type {newkey {}}} {
  if {[info exists ::vTcl(fonts,$font_descr,object)]} {
    return $::vTcl(fonts,$font_descr,object)
  }
  incr ::vTcl(fonts,counter)
  set newfont [eval font create
    $font_descr]
  lappend ::vTcl(fonts,objects) $newfont
  if {$newkey == ""} {
    set newkey vTcl:font$::vTcl(fonts,counter)
  } while
  {[vTcl:font:get_font $newkey] != ""} {
    incr ::vTcl(fonts,counter)
    set newkey
      vTcl:font$::vTcl(fonts,counter)
  }
  set ::vTcl(fonts,$newfont,type) $font_type
  set ::vTcl(fonts,$newfont,key) $newkey
  set ::vTcl(fonts,$newfont,font_descr) $font_descr
  set ::vTcl(fonts,$font_descr,object) $newfont
  set
    ::vTcl(fonts,$newkey,object) $newfont
  lappend
    ::vTcl(fonts,$font_type) $newfont
  return $newfont
}
proc ::vTcl:font:getFontFromDescr {font_descr} {
  if {[info exists ::vTcl(fonts,$font_descr,object)]} {
    return $::vTcl(fonts,$font_descr,object)
  } else {
    return ""
  }
}
vTcl:font:add_font \
  "-family helvetica -size 12" \
stock \
vTcl:font1
vTcl:font:add_font \
  "-family lucida -size 18" \
stock \
vTcl:font8
}
if {[info exists vTcl(sourcing)]} {
  proc ::Window {args} {
    global vTcl
    foreach {cmd name newname} [lrange $args 0 2] {
      set rest [lrange $args 3 end]
      if {$name == "" || $cmd == ""} {
        return
      }
      if {$newname == ""} { set newname $name }
      if {$name ==
        "."} { vm withdraw $name; return }
      set exists [winfo exists $newname]

      switch $cmd {
        show {
          if {$exists} {
            vm deiconify $newname
          } elseif {[info procs vTclWindow$name] != ""} {
            eval "vTclWindow$name $newname $rest"
          }
          if {[winfo exists $newname] && [vm state $newname] ==
            "normal"} {
              vTcl:FireEvent $newname <<Show>>
            }
        }
        hide {
          if {$exists} {
            vm withdraw $newname
            vTcl:FireEvent $newname <<Hide>>
          }
        }
        iconify { if $exists {vm iconify $newname; return} }
        destroy {
          if $exists {destroy $newname; return}
        }
      }
    }
    proc ::vTcl:DefineAlias {target alias widgetProc top_or_alias cmdalias} {
      proc ::vTcl:toplevel {args} {
        global widget
        set widget($alias) $target
        set widget(rev,$target)
        $alias
        if {$cmdalias} {
          interp alias {} $alias {} $widgetProc $target
        }
        if {$top_or_alias != ""} {
          set widget($top_or_alias,$alias) $target
          if {$cmdalias} {
            interp alias {} $top_or_alias.$alias {} $widgetProc $target
          }
        }
      }
      proc ::vTcl:DoCmdOption {target cmd} {
        set parent $target
        while {[winfo class $parent] == "Menu"} {
          set parent [winfo parent $parent]
        }
        regsub -all {\%/widget} $cmd $target cmd
        regsub -all {\%/top} $cmd
          [winfo toplevel $parent] cmd
        uplevel #0 [list eval $cmd]
      }
      proc ::vTcl:FireEvent {target event {params {}}} {
        if {[winfo exists $target]} return
        foreach bindtag [bindtags
          $target] {
          set tag_events [bind $bindtag]
          set stop_processing 0
          foreach
            tag_event $tag_events {
              if {$tag_event == $event} {
                set bind_code [bind $bindtag $tag_event]
                foreach rep
                  "\%/W $target\)" $params {
                    regsub -all [lindex $rep 0] $bind_code [lindex $rep 1]
                    bind_code
                  }
                set result [catch {uplevel #0 $bind_code} errortext]
                if {$result == 3} {
                  set stop_processing 1
                } elseif {$result != 0} {
                  berror $errortext
                }
                break
              }
            }
        }
        if {$stop_processing} {break}
      }
      proc ::vTcl:Toplevel:WidgetProc {w args} {
        if {[llength $args] == 0} {
          return $w
        }
        set command [lindex $args 0]
        set args [lrange $args 1 end]

        switch -- [string tolower $command] {
          "setvar" {
            foreach {varname value} $args {
              if {$value == ""} {
                return [set ::${w}::${varname}]
              } else {
                return [set ::${w}::${varname} $value]
              }
            }
          }
          "hide" - "show" {
            Window [string tolower $command] $w
          }
          "showmodal" {
            Window show $w; raise $w
            grab $w; tkwait window $w; grab
            release $w
          }
          "startmodal" {
            Window show $w; raise $w
            set ::${w}::_modal 1
            grab $w;
            tkwait variable ::${w}::_modal; grab release $w
          }
          "endmodal" {
            set ::${w}::_modal 0
            Window hide $w
          }
          default {
            uplevel $w $command $args
          }
        }
      }
      proc ::vTcl:WidgetProc {w args} {
        if {[llength $args] == 0} {
          return $w
        }
        set command [lindex $args 0]
        set args [lrange $args 1 end]
        uplevel $w $command $args
      }
    }
  }
}

```

```

    uplevel #0 eval toplevel $args
    set target [lindex $args 0]
    namespace
        eval ::$target {set _modal 0}
    }
    if {[info exists vTcl(sourcing)]} {
        proc vTcl::project::info {} {
            set base .command_file
            namespace eval ::widgets::$base {
                set set,origin 1
                set set,size 1
                set runvisible 1
            }
            namespace eval ::widgets::$base.button1 {
                array set save {-command 1 -disabledforeground 1 -text 1}
            }
            namespace eval ::widgets::$base.frame1 {
                array set save {-font 1 -foreground 1 -highlightcolor 1 -text 1}
            }
            set site_3_0 $base.frame1
            namespace eval
                ::widgets::$site_3_0.entry1 {
                    array set save {-background 1 -disabledforeground 1 -font 1
                        -insertbackground 1 -textvariable 1}
                }
                namespace eval ::widgets::$site_3_0.button_choose_file {
                    array set save {-command 1 -disabledforeground 1 -text 1}
                }
                namespace eval ::widgets::$base.label1 {
                    array set save {-disabledforeground 1 -font 1 -text 1}
                }
                set base .control
                namespace eval ::widgets::$base {
                    set set,origin 1
                    set set,size 1
                    set runvisible 1
                }
                namespace eval ::widgets::$base.canvas {
                    array set save {-borderwidth 1 -closeenough 1 -height
                        1 -insertbackground 1 -relief 1 -selectbackground 1
                        -selectforeground 1 -width 1}
                }
                namespace eval ::widgets::$base.label1 {
                    array set save {-disabledforeground 1 -font 1 -text 1}
                }
                namespace eval ::widgets::$base.cpd80 {
                    array set save {-activebackground 1 -activeforeground 1 -command
                        1 -disabledforeground 1 -text 1}
                }
                namespace eval ::widgets::$base.canvas_run_type {
                    array set save {-borderwidth 1 -closeenough 1 -height
                        1 -insertbackground 1 -relief 1 -selectbackground 1
                        -selectforeground 1 -width 1}
                }
                namespace eval ::widgets::$base.canvas3 {
                    array set save {-borderwidth 1 -closeenough 1 -height
                        1 -insertbackground 1 -relief 1 -selectbackground 1
                        -selectforeground 1 -width 1}
                }
                set base .design_choice
                namespace eval ::widgets::$base {
                    set set,origin 1
                    set set,size 1
                    set runvisible 1
                }
                namespace eval ::widgets::$base.button1 {
                    array set save {-command 1 -disabledforeground 1 -text 1}
                }
                namespace eval ::widgets::$base.cpd82 {
                    array set save {-background 1 -foreground 1 -height 1 -maximum
                        1 -relief 1 -troughcolor 1 -variable 1}
                }
                namespace eval ::widgets::$base.cpd83 {
                    array set save {-background 1 -disabledforeground 1
                        -insertbackground 1 -relief 1 -state 1 -textvariable 1}
                }
                namespace eval ::widgets::$base.canvas1 {
                    array set save {-borderwidth 1 -closeenough 1 -height
                        1 -insertbackground 1 -relief 1 -selectbackground 1
                        -selectforeground 1 -width 1}
                }
                namespace eval ::widgets::$base.title {
                    array set save {-disabledforeground 1 -font 1 -text 1}
                }
                set base .interactive_common
                namespace eval ::widgets::$base {
                    set set,origin 1
                    set set,size 1
                    set runvisible 1
                }
                namespace eval ::widgets::$base.frame1 {
                    array set save {-borderwidth 1 -height 1 -relief 1 -width 1}
                }
                set site_3_0 $base.frame1
                namespace eval
                    ::widgets::$site_3_0.label1 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label2 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label3 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label4 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.line1 {
                        array set save {-background 1 -orient 1}
                    }
                    namespace eval ::widgets::$site_3_0.label5 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label6 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label7 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label8 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry1 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry2 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.templ {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -state 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry3 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1 -validate 1 -validatecommand
                            1}
                    }
                    namespace eval ::widgets::$site_3_0.entry4 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry5 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry6 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry7 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry8 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.cpd75 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.cpd77 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.label_templ {
                        array set save {-disabledforeground 1 -font 1 -state 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label10 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry10 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.label11 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry11 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.cpd73 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.cpd74 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.label12 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.entry12 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }
                    namespace eval ::widgets::$site_3_0.label13 {
                        array set save {-disabledforeground 1 -font 1 -text 1}
                    }
                    namespace eval ::widgets::$site_3_0.cpd78 {
                        array set save {-background 1 -disabledforeground 1
                            -insertbackground 1 -textvariable 1}
                    }

```


[illegible]


```

array set save {-command 1 -disabledforeground 1 -text 1}
}
set base .interactive_process_edit
namespace eval ::widgets::$base {
    set set,origin 1
    set set,size 1
    set runvisible 1
}
namespace eval ::widgets::$base.canvas1 {
    array set save {-borderwidth 1 -closeenough 1 -height
1 -insertbackground 1 -relief 1 -selectbackground 1
-selectforeground 1 -width 1}
}
namespace eval ::widgets::$base.butt1 {
    array set save {-command 1 -disabledforeground 1 -text 1}
}
set base .results_display
namespace eval ::widgets::$base {
    set set,origin 1
    set set,size 1
    set runvisible 1
}
namespace eval ::widgets::$base.canvas1 {
    array set save {-borderwidth 1 -closeenough 1 -height
1 -insertbackground 1 -relief 1 -selectbackground 1
-selectforeground 1 -width 1}
}
namespace eval ::widgets::$base.label1 {
    array set save {-disabledforeground 1 -font 1 -relief 1 -text 1}
}
namespace eval ::widgets::$base.butt1 {
    array set save {-command 1 -disabledforeground 1 -text 1}
}
namespace eval ::widgets::$base.butt2 {
    array set save {-command 1 -disabledforeground 1 -text 1}
}
set base .wait
namespace eval ::widgets::$base {
    set set,origin 1
    set set,size 1
    set runvisible 1
}
namespace eval ::widgets::$base.canvas1 {
    array set save {-borderwidth 1 -closeenough 1 -height
1 -highlightthickness 1 -insertbackground 1 -relief 1
-selectbackground 1 -selectforeground 1 -width 1}
}
namespace eval ::widgets::bindings {
    set taglist {TopLevel vTclBalloon}
}
namespace eval ::vTcl::modules::main {
    set procs {
        init
        main
    }
    set compounds {
    }
    set projectType single
}
}
}
proc ::main {argc argv} {
    wm withdraw .command_file
    wm
    withdraw .interactive_common
    wm withdraw .interactive_individual_1
    wm
    withdraw .interactive_process
    wm withdraw .design_choice
    wm withdraw
    .interactive_common_edit
    wm withdraw .interactive_individual_1_edit
    wm withdraw .results_display
    wm withdraw .wait
    set no_modules 0
}
}
proc ::init {argc argv} {
}
init $argc $argv
proc vTclWindow. {base} {
    if {$base == ""} {
    }
    set base .
}
wm focusmodel $top passive
wm geometry $top 1x1+0+0; update
wm
    maxsize $top 1265 994
    minsize $top 1 1
    overriddenirect $top
    0
wm resizable $top 1 1
wm withdraw $top
wm title $top "vTcl.tcl"
bindtags $top "$top vTcl.tcl all"
vTcl::FireEvent $top <<Create>>
wm
    protocol $top WM_DELETE_WINDOW "vTcl::FireEvent $top <<DeleteWindow>>"
vTcl::FireEvent $base <<Ready>>
}
proc vTclWindow.command_file {base} {
    if {$base == ""} {
        set base .command_file
    }
    if {[winfo exists $base]} {
        wm deiconify $base; return
    }
    set top $base
    vTcl::TopLevel $top -class TopLevel \
-relief raised -highlightcolor black
    wm withdraw $top
    wm focusmodel $top passive
    wm geometry $top
        616x172+303+540; update
    wm maxsize $top 1265 994
    wm minsize $top
        1 1
    wm overriddenirect $top 0
    wm resizable $top 0 0
    wm title
        $top "Command file entry"
    vTcl::DefineAlias "$top" "TopLevel2"
    vTcl::TopLevel::WidgetProc "" 1
    bindtags $top "$top TopLevel all
        _TopLevel"
    vTcl::FireEvent $top <<Create>>
    wm protocol $top
        WM_DELETE_WINDOW "vTcl::FireEvent $top <<DeleteWindow>>"
    button
        $top.butt1 \
    \
    -command {wm withdraw .command_file
set edit_design_choices [tk_messageBox -title "EIDA question" -message
"Edit design choices?" -type yesno -icon question]
    if {[string equal
$edit_design_choices "yes"]} {
        catch {exec more $cmd_filename | grep
".mods_des" | wc -l} no_modules
        puts "Modules read $no_modules"
        set x
        0
        set new_cmd_filename $cmd_filename
        set vdd_bump_for_cmd_file_run 1
        set vdd_applied 0
        set vary_vdd 0
        set use_dual_vt 0
        set use_abb 0
        set
        use_st 0
        catch {exec \.\./scripts/clear_design_choices.pl $cmd_filename
} junk
        wm deiconify .design_choice
    } else {
        wm deiconify .wait
        catch
        {exec \.\./scripts/eida.pl $cmd_filename} junk1
        wm withdraw .wait
        set
        result_ext ".module.results"
        set cmd_filename1 $cmd_filename
        append
        cmd_filename1 $result_ext
        catch {exec more $cmd_filename1} results
        wm
        deiconify .results_display
        set sw [text .results_display.canvas1.sw
-wrap char -height 20]
        pack $sw -fill both -expand 1
        $sw insert end
        $results
    }
    \
    -disabledforeground #a1a1a1 -text {Execute command file}
    vTcl::DefineAlias "$top.butt1" "Button1" vTcl::WidgetProc "TopLevel2"
    1
    labelframe $top.frame1 \
    -font [vTcl::font:getFontFromDescr "-family helvetica -size 12"]
    \
    -foreground black -text {Module descriptors} -highlightcolor
    black
    vTcl::DefineAlias "$top.frame1" "Labelframe1" vTcl::WidgetProc
    "TopLevel2" 1
    set site_3_0 $top.frame1
    entry $site_3_0.entry1 \
    -background white -disabledforeground #a1a1a1 \
    -font
    [vTcl::font:getFontFromDescr "-family helvetica -size 12"] \
    \
    -insertbackground black -textvariable cmd_filename
    vTcl::DefineAlias "$site_3_0.entry1" "Entry1" vTcl::WidgetProc
    "TopLevel2" 1
    button $site_3_0.butt_choose_file \
    \
    -command {wm withdraw .command_file
set types {
    {(Command Files) {.cmd}
    {(All Files) *
    }
    }
    set cmd_filename ""
    while {[string equal $cmd_filename ""]} {
        set
        cmd_filename [tk_getOpenFile -filetypes $types -initialdir ./ -initialfile

```

```

command_file -title "Choose command file"
}
wm deiconify .command_file}
\
-disabledforeground #a1a1a1 -text Browse
vTcl::DefineAlias "$site_3.0.butt_choose_file" "Button2"
vTcl::WidgetProc "Toplevel2" 1
place $site_3.0.entry1 \
-in $site_3.0 -x 30 -y 60 -width 428 -height 25 -anchor nw \

-bordermode ignore
place $site_3.0.butt_choose_file \
-in $site_3.0 -x 483 -y 58 -width 76 -height 28 -anchor nw \

-bordermode ignore
label $stop.label1 \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Descriptor file path}
vTcl::DefineAlias "$stop.label1" "Label1" vTcl::WidgetProc "Toplevel2" 1
place $stop.butt1 \
-in $stop -x 230 -y 135 -width 161 -height 28 -anchor nw \

-bordermode ignore
place $stop.frame1 \
-in $stop -x 15 -y 20 -width 586 -height 101 -anchor nw \

-bordermode ignore
place $stop.label1 \
-in $stop -x 30 -y 50 -width 144 -height 20 -anchor nw \

-bordermode ignore
vTcl::FireEvent $base <<Ready>>
}
proc vTclWindow::control {base} {
if {$base == ""} {
set base .control
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl::toplevel $stop -class Toplevel \
-relief groove -highlightcolor black
wm focusmodel $stop passive
wm geometry $stop 605x355+264+167; update

wm maxsize $stop 1265 994
wm minsize $stop 1 1
wm overrideredirect
$stop 0
wm resizable $stop 0 0
wm deiconify $stop
wm title $stop "EIDA
control"
vTcl::DefineAlias "$stop" "Toplevel1" vTcl::Toplevel::WidgetProc
"" 1
bindtags $stop "$stop Toplevel all _TopLevel"
vTcl::FireEvent
$stop <<Create>>
wm protocol $stop WM_DELETE_WINDOW "vTcl::FireEvent
$stop <<DeleteWindow>>"
canvas $stop.canvas \
-borderwidth 2 -closeenough 1.0 -height 87 -insertbackground black \
-relief groove -selectbackground #c1c2c1 -selectforeground
black \
-width 575
vTcl::DefineAlias "$stop.canvas" "Canvas1" vTcl::WidgetProc "Toplevel1" 1
entry $stop.canvas.entry1 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black \
-textvariable no_modules
-validate key \
-validatecommand {.control.canvas.butt_set_blocks
config -state normal
.control.canvas_run_type.butt_run_type1 config -state normal
return 1}
vTcl::DefineAlias "$stop.canvas.entry1" "Entry1" vTcl::WidgetProc
"Toplevel1" 1
label $stop.canvas.label_no_of_modules \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Number of modules in design}
vTcl::DefineAlias "$stop.canvas.label_no_of_modules" "Label3"
vTcl::WidgetProc "Toplevel1" 1
button $stop.canvas.butt_set_blocks \
-activebackground #f6f7f6 -activeforeground black \
-command
{.control.canvas.entry1 config -state disabled
.control.canvas.butt_unset_modules config -state normal
.control.canvas.butt_set_blocks config -state disabled
set msg
$no_modules
append msg " module descriptor vectors expected"
tk_messageBox -title "Confirm # modules" -type ok -message
$msg -icon info
set x 0
set y [expr {($x/$no_modules) * 100}]
set
module_entry_percentage $y
set module_entry_step "Descriptor vector
entry step" \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-state disabled -text {Set # modules}
vTcl::DefineAlias "$stop.canvas.butt_set_blocks" "Button1"
vTcl::WidgetProc "Toplevel1" 1
button $stop.canvas.butt_unset_modules
\
-activebackground #f6f7f6 -activeforeground black \
-command
.control.canvas.entry1 config -state normal
set .control::entry1 0
.control.canvas.butt_unset_modules config -state
disabled
.control.canvas.butt_set_blocks config -state normal} \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-state disabled -text {Unset # modules}
vTcl::DefineAlias "$stop.canvas.butt_unset_modules" "Button4"
vTcl::WidgetProc "Toplevel1" 1
Separator $stop.canvas.line2 \
-background #d6cbbb
vTcl::DefineAlias "$stop.canvas.line2" "Separator2" vTcl::WidgetProc
"Toplevel1" 1
bind $stop.canvas.line2 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $stop.label1 \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family lucida -size 18"] \
-text {Early Integrated circuit
Design Assist}
vTcl::DefineAlias "$stop.label1" "Label1" vTcl::WidgetProc "Toplevel1"
1
button $stop.cpd80 \
-activebackground #f6f7f6 -activeforeground black \
-command
{#puts $tcl_pkgPath
catch {exec rm -f tmp_cmd_filename} junk3
catch {exec rm -f
temp_cmd_file} junk3
catch {exec rm -f abb_spice_run_junk} junk3
exit}
\
-disabledforeground #a1a1a1 -text Exit
vTcl::DefineAlias "$stop.cpd80" "Button6" vTcl::WidgetProc "Toplevel1"
1
canvas $stop.canvas_run_type \
-borderwidth 2 -closeenough 1.0 -height 97 -insertbackground
black \
-relief ridge -selectbackground #c1c2c1 -selectforeground
black \
-width 575
vTcl::DefineAlias "$stop.canvas_run_type" "Canvas2" vTcl::WidgetProc
"Toplevel1" 1
label $stop.canvas_run_type.label_run_type \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text {Choose run type}
vTcl::DefineAlias "$stop.canvas_run_type.label_run_type"
"Label5" vTcl::WidgetProc "Toplevel1" 1
label
$stop.canvas_run_type.label_run_type1 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Interactive descriptor vector entry run}
vTcl::DefineAlias "$stop.canvas_run_type.label_run_type1"
"Label6" vTcl::WidgetProc "Toplevel1" 1
label
$stop.canvas_run_type.label_run_type2 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Preformatted command file run}
vTcl::DefineAlias "$stop.canvas_run_type.label_run_type2"
"Label7" vTcl::WidgetProc "Toplevel1" 1
button
$stop.canvas_run_type.butt_run_type1 \
-activebackground #f6f7f6 -activeforeground black \
-command
{#puts "executing scripts"

```

```

wm withdraw .control
set vdd_spec 0
set eta 0.95
set vdd_bump 0

set l_min 0
set s_gate_cap 0
set s_wire_cap 0
set s_width 0
set
c_unit_old 0
set i_ds_old 0
set c_unit_new 0
set i_ds_new 0
set rpsf
0
set rcsf 0
set sf 0
set decap_sens 0
set nadsp 0
set i_leak_junc
0
set i_leak_gate 0
set i_gate_per_w 0
set i_othv 0
set i_owhv 0
set i_otlv 0
set i_owlv 0
wm deiconify .interactive_common \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-state disabled -text {Interactive run}
vTcl::DefineAlias "$stop.canvas_run_type.butt_run_type1"
"Button9" vTcl::WidgetProc "Toplevel1" 1
button
$stop.canvas_run_type.butt_run_type2 \
-activebackground #f6f7f6 -activeforeground black \
-command
{wm withdraw .control
wm deiconify .command_file \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-text {Command file run}
vTcl::DefineAlias "$stop.canvas_run_type.butt_run_type2" "Button10"
vTcl::WidgetProc "Toplevel1" 1
Separator $stop.canvas_run_type.line1 \
-background #d6cddb
vTcl::DefineAlias "$stop.canvas_run_type.line1" "Separator1"
vTcl::WidgetProc "Toplevel1" 1
bind $stop.canvas_run_type.line1
<Destroy> {
Widget::destroy %W; rename %W {}
}
canvas $stop.canvas3 \
-borderwidth 2 -closeenough 1.0 -height 87 -insertbackground
black \
-relief ridge -selectbackground #c1c2c1 -selectforeground
black \
-width 575
vTcl::DefineAlias "$stop.canvas3" "Canvas3" vTcl::WidgetProc "Toplevel1" 1
label $stop.canvas3.label8 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Edit existing module description}
vTcl::DefineAlias "$stop.canvas3.label8" "Label2" vTcl::WidgetProc
"Toplevel1" 1
entry $stop.canvas3.entry2 \
-background white -disabledforeground #a1a1a1 -foreground
black \
-highlightcolor black -insertbackground black \
-selectbackground #c1c2c1 -selectforeground black \
-textvariable
cmd_file_to_edit
vTcl::DefineAlias "$stop.canvas3.entry2" "Entry2" vTcl::WidgetProc
"Toplevel1" 1
button $stop.canvas3.butt5 \
-activebackground #f6f7f6 -activeforeground black \
-command
{set types {
{{Command Files} {*.cmd}
}}
}
}
set cmd_file_to_edit ""
while {[string equal $cmd_file_to_edit ""]} {
set cmd_file_to_edit [tk_getOpenFile -filetypes $types -initialdir
./ -initialfile command_file -title "Choose command file"]
}
.control.canvas3.butt5 config -state disabled
.control.canvas3.butt6
config -state normal \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-text Choose
vTcl::DefineAlias "$stop.canvas3.butt5" "Button7" vTcl::WidgetProc
"Toplevel1" 1
button $stop.canvas3.butt6 \
-activebackground #f6f7f6 -activeforeground black \
-command
{catch {exec more $cmd_file_to_edit | grep ".com_des" | wc -l}
read_common_des
catch {exec more $cmd_file_to_edit | grep ".pro_des" | wc -l}
read_process_des
catch {exec more $cmd_file_to_edit | grep
".mods_des" | wc -l} read_modules_des
tk_messageBox -title
"Command file details" -type ok -message "$read_common_des -
common module description,\n$read_process_des - process description
and\n$read_modules_des - module description detected in command file"
-icon info
catch {exec more $cmd_file_to_edit | grep ".com_des"}
read_common_des
set tt [split $read_common_des " "]
set vdd_spec
[lindex $tt 1]
set eta [lindex $tt 2]
set vdd_bump [expr [lindex $tt
1] * [lindex $tt 2]]
set l_min [lindex $tt 3]
set s_gate_cap [lindex
$tt 4]
set s_wire_cap [lindex $tt 5]
set s_width [lindex $tt 6]
set
c_unit_old [lindex $tt 7]
set i_ds_old [lindex $tt 8]
set c_unit_new
[lindex $tt 9]
set i_ds_new [lindex $tt 10]
set rpsf [lindex $tt 11]
set rcsf [lindex $tt 12]
set sf [lindex $tt 13]
set decap_sens [lindex
$tt 14]
set i_leak_junc [lindex $tt 15]
set i_leak_gate [lindex $tt 16]

set i_gate_per_w [lindex $tt 17]
set i_othv [lindex $tt 18]
set i_owhv
[lindex $tt 19]
set i_otlv [lindex $tt 20]
set i_owlv [lindex $tt 21]

wm deiconify .interactive_common_edit
.control.canvas3.butt6 config
-state disabled
.control.canvas3.butt5 config -state normal
wm withdraw
.control \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-state disabled -text Edit
vTcl::DefineAlias "$stop.canvas3.butt6" "Button8" vTcl::WidgetProc
"Toplevel1" 1
Separator $stop.canvas3.cpd75 \
-background #d6cddb
vTcl::DefineAlias "$stop.canvas3.cpd75" "Separator3" vTcl::WidgetProc
"Toplevel1" 1
bind $stop.canvas3.cpd75 <Destroy> {
Widget::destroy %W; rename %W {}
}
place $stop.canvas \
-in $stop -x 15 -y 40 -width 575 -height 87 -anchor nw \
-bordermode ignore
place $stop.canvas.entry1 \
-in $stop.canvas -x 145 -y 45 -width 83 -height 25 -anchor nw \
-bordermode ignore
place $stop.canvas.label_no_of_modules \
-in $stop.canvas -x 7 -y 3 -width 218 -height 24 -anchor nw \
-bordermode ignore
place $stop.canvas.butt_set_blocks \
-in $stop.canvas -x 285 -y 45 -width 101 -height 28 -anchor nw \
-bordermode ignore
place $stop.canvas.butt_unset_modules \
-in $stop.canvas -x 416 -y 45 -width 130 -height 28 -anchor nw \
-bordermode ignore
place $stop.canvas.line2 \
-in $stop.canvas -x 400 -y 42 -width 2 -height 37 -anchor nw \
-bordermode ignore
place $stop.label1 \
-in $stop -x 70 -y 0 -anchor nw -bordermode ignore
place $stop.cpd80 \
-in $stop -x 260 -y 320 -anchor nw -bordermode inside
place $stop.canvas_run_type \
-in $stop -x 15 -y 130 -width 575 -height 97 -anchor nw \
-bordermode ignore
place $stop.canvas_run_type.label_run_type \
-in $stop.canvas_run_type -x 5 -y 3 -width 126 -height 24 -anchor
nw \

```

```

-bordermode ignore
place $top.canvas_run_type.label_run_type1 \
-in $top.canvas_run_type -x 110 -y 24 -width 273 -height 24 \

-anchor nw -bordermode ignore
place $top.canvas_run_type.label_run_type2 \
-in $top.canvas_run_type -x 112 -y 59 -width 217 -height 24 \

-anchor nw -bordermode ignore
place $top.canvas_run_type.butt_run_type1 \
-in $top.canvas_run_type -x 385 -y 23 -width 133 -height 28 \

-anchor nw -bordermode ignore
place $top.canvas_run_type.butt_run_type2 \
-in $top.canvas_run_type -x 385 -y 59 -width 133 -height 28 \

-anchor nw -bordermode ignore
place $top.canvas_run_type.line1 \
-in $top.canvas_run_type -x 365 -y 54 -width 171 -height 2
-anchor nw \
-bordermode ignore
place $top.canvas3 \
-in $top -x 15 -y 230 -width 575 -height 87 -anchor nw \

-bordermode ignore
place $top.canvas3.label18 \
-in $top.canvas3 -x 5 -y 5 -width 234 -height 24 -anchor nw \

-bordermode ignore
place $top.canvas3.entry2 \
-in $top.canvas3 -x 85 -y 35 -width 298 -height 25 -anchor nw \

-bordermode ignore
place $top.canvas3.butt5 \
-in $top.canvas3 -x 400 -y 30 -width 73 -height 28 -anchor nw \

-bordermode ignore
place $top.canvas3.butt6 \
-in $top.canvas3 -x 504 -y 30 -width 53 -height 28 -anchor nw \

-bordermode ignore
place $top.canvas3.cpd75 \
-in $top.canvas3 -x 490 -y 33 -width 2 -height 27 -anchor nw \

-bordermode ignore
vTcl:FireEvent $base <<Ready>>
}
proc vTclWindow.design_choice {base} {
    if {$base == ""} {
        set base .design_choice
    }
    if {[winfo exists $base]} {
        wm deiconify $base; return
    }
    set top $base
    vTcl:toplevel $top -class Toplevel \
    -highlightcolor black
    wm withdraw $top
    wm focusmodel $top passive
    wm geometry $top
    609x378+231+378; update
    wm maxsize $top 1265 994
    wm minsize
    $top 1 1
    wm overriddenirect $top 0
    wm resizable $top 0 0
    wm
    title $top "Design choices"
    vTcl:DefineAlias "$top" "Toplevel6"
    vTcl:Toplevel:WidgetProc "" 1
    bindtags $top "$top Toplevel all
    _Toplevel"
    vTcl:FireEvent $top <<Create>>
    wm protocol $top
    WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"
    button
    $top.butt1 \
    \
    -command [list vTcl:DoCmdOption $top.butt1 {incr x
    set new_cmd_file_id [open $new_cmd_filename a]
    puts $new_cmd_file_id
    ".mods_choices $x $vary_vdd $vdd_applied $use_dual_vt $use_abb
    $use_abb_fb $use_abb_rb $use_st\n"
    close $new_cmd_file_id
    set
    vdd_applied $vdd_bump
    if {$vdd_bump_for_cmd_file_run == 1} {
        set
        vdd_applied 0
    }
    set vary_vdd 0
    .design_choice.canvas1.entry1 config
    -state disabled
    set use_dual_vt 0
    set use_abb 0
    set use_abb_fb 0
    set
    use_abb_rb 0
    set use_st 0
    if {$x<[expr $no_modules + 1]} {
        set al
        [expr 0.01/$x]
        set bl [expr 0.01/$no_modules]
        set b [expr {$bl/$al}
        * 100]
        set module_design_precentage $b
        set module_design_step "$x
        out of $no_modules entered"
    }
    if {$x == $no_modules} {
        wm withdraw
        .design_choice
        tk_messageBox -title "EIDA information" -type ok -message
        "Ready for What-if analysis" -icon info
        wm deiconify .wait
        catch
        {exec \\.\/scripts/eida.pl $cmd_filename} junk1
        wm withdraw .wait
        set
        result_ext ".module.results"
        set cmd_filename1 $cmd_filename
        append
        cmd_filename1 $result_ext
        catch {exec more $cmd_filename1} results
        wm
        deiconify .results_display
        set sw [text .results_display.canvas1.sw
        -wrap char -height 20]
        pack $sw -fill both -expand 1
        $sw insert end
        $results
    }
    }
    -disabledforeground #a1a1a1 -text {Save and proceed}
    vTcl:DefineAlias "$top.butt1" "Button1" vTcl:WidgetProc "Toplevel6"
    1
    Progressbar $top.cpd82 \
    -background #d6cddb -foreground #000099 -height 15 -maximum
    100 \
    -relief raised -troughcolor #d9d9d9 \
    -variable
    module_design_precentage
    vTcl:DefineAlias "$top.cpd82" "Progressbar1" vTcl:WidgetProc
    "Toplevel6" 1
    entry $top.cpd83 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -relief groove -state readonly -textvariable
    module_design_step
    vTcl:DefineAlias "$top.cpd83" "Entry3" vTcl:WidgetProc "Toplevel6"
    1
    canvas $top.canvas1 \
    -borderwidth 2 -closeenough 1.0 -height 277 -insertbackground
    black \
    -relief ridge -selectbackground #c1c2c1 -selectforeground
    black \
    -width 590
    vTcl:DefineAlias "$top.canvas1" "Canvas1" vTcl:WidgetProc "Toplevel6"
    1
    label $top.canvas1.label1 \
    -activebackground #f6f7f6 -activeforeground black \
    -disabledforeground #a1a1a1 \
    -font [vTcl:font:getFontFromDescr
    "-family helvetica -size 12"] \
    -foreground black -highlightcolor
    black -text {Vary Vdd ?}
    vTcl:DefineAlias "$top.canvas1.label1" "Label2" vTcl:WidgetProc
    "Toplevel6" 1
    checkbox $top.canvas1.check1 \
    -activebackground #f6f7f6 -activeforeground black \
    -command
    {if {$vary_vdd == 1} {
        .design_choice.canvas1.entry1 config -state normal
        .design_choice.canvas1.label1a config -state normal
    }
    }
    else {
        .design_choice.canvas1.entry1 config -state disabled
        .design_choice.canvas1.label1a config -state disabled
    }
    }
    -disabledforeground #a1a1a1 -foreground black -highlightcolor
    black \
    -text {YES / NO} -variable vary_vdd
    vTcl:DefineAlias "$top.canvas1.check1" "Checkbutton1" vTcl:WidgetProc
    "Toplevel6" 1
    label $top.canvas1.label1a \
    -activebackground #f6f7f6 -activeforeground black \
    -disabledforeground #a1a1a1 \
    -font [vTcl:font:getFontFromDescr
    "-family helvetica -size 12"] \
    -foreground black -highlightcolor
    black -state disabled \
    -text {Enter applied Vdd (V)}
    vTcl:DefineAlias "$top.canvas1.label1a" "Label3" vTcl:WidgetProc
    "Toplevel6" 1
    entry $top.canvas1.entry1 \
    -background white -disabledforeground #a1a1a1 -foreground black \
    -highlightcolor black -insertbackground black \
    -selectbackground
    #c1c2c1 -selectforeground black -state disabled \
    -textvariable
    vdd_applied
    vTcl:DefineAlias "$top.canvas1.entry1" "Entry1" vTcl:WidgetProc
    "Toplevel6" 1
    label $top.canvas1.label2 \

```

```

-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text {Use Dual-Vt FETs ?}
vTcl:DefineAlias "$top.canvas1.label2" "Label4" vTcl:WidgetProc
"Toplevel6" 1
checkboxbutton $top.canvas1.check2 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-text {YES / NO} -variable use_dual_vt
vTcl:DefineAlias "$top.canvas1.check2" "Checkbox2" vTcl:WidgetProc
"Toplevel6" 1
label $top.canvas1.label3 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text {Use ABB ?}
vTcl:DefineAlias "$top.canvas1.label3" "Label5" vTcl:WidgetProc
"Toplevel6" 1
checkboxbutton $top.canvas1.check3 \
-activebackground #f6f7f6 -activeforeground black \
-command
{if {$use_abb == 1} {
.design_choice.canvas1.check3a config -state normal
.design_choice.canvas1.check3b config -state normal
}
else {
.design_choice.canvas1.check3a config -state disabled
.design_choice.canvas1.check3b config -state disabled
set use_abb_fb
0
set use_abb_fb 0
}} \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-text {YES / NO} -variable use_abb
vTcl:DefineAlias "$top.canvas1.check3" "Checkbox3" vTcl:WidgetProc
"Toplevel6" 1
checkboxbutton $top.canvas1.check3a \
-activebackground #f6f7f6 -activeforeground black \
-command
{if {$use_abb_fb == 1} {
.design_choice.canvas1.check3b config -state disabled
set use_abb_fb
0
.design_choice.canvas1.check3a config -state normal
} else {
.design_choice.canvas1.check3a config -state disabled
set use_abb_fb
0
.design_choice.canvas1.check3b config -state normal
}} \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-state disabled -text FB -variable use_abb_fb
vTcl:DefineAlias "$top.canvas1.check3a" "Checkbox4" vTcl:WidgetProc
"Toplevel6" 1
checkboxbutton $top.canvas1.check3b \
-activebackground #f6f7f6 -activeforeground black \
-command
{if {$use_abb_fb == 1} {
.design_choice.canvas1.check3a config -state disabled
set use_abb_fb
0
.design_choice.canvas1.check3b config -state normal
} else {
.design_choice.canvas1.check3b config -state disabled
set use_abb_fb
0
.design_choice.canvas1.check3a config -state normal
}} \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-state disabled -text RB -variable use_abb_fb
vTcl:DefineAlias "$top.canvas1.check3b" "Checkbox5" vTcl:WidgetProc
"Toplevel6" 1
label $top.canvas1.label4 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Use sleep transistors ?}
vTcl:DefineAlias "$top.canvas1.label4" "Label6" vTcl:WidgetProc
"Toplevel6" 1
checkboxbutton $top.canvas1.check4 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-text {YES / NO} -variable use_st
vTcl:DefineAlias "$top.canvas1.check4" "Checkbox6" vTcl:WidgetProc
"Toplevel6" 1
label $top.title \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Select design recipe
applied to module}
vTcl:DefineAlias "$top.title" "Label1" vTcl:WidgetProc "Toplevel6"
1
place $top.button1 \
-in $top -x 240 -y 340 -anchor nw -bordermode ignore
place $top.cpd82 \
-in $top -x 355 -y 5 -width 245 -height 15 -anchor nw \
-bordermode inside
place $top.cpd83 \
-in $top -x 358 -y 25 -width 243 -height 22 -anchor nw \
-bordermode ignore
place $top.canvas1 \
-in $top -x 9 -y 55 -width 590 -height 277 -anchor nw \
-bordermode ignore
place $top.canvas1.label1 \
-in $top.canvas1 -x 9 -y 15 -width 87 -height 24 -anchor nw \
-bordermode ignore
place $top.canvas1.check1 \
-in $top.canvas1 -x 115 -y 15 -anchor nw -bordermode ignore
place $top.canvas1.label1a \
-in $top.canvas1 -x 225 -y 15 -anchor nw -bordermode ignore
place $top.canvas1.entry1 \
-in $top.canvas1 -x 405 -y 15 -anchor nw -bordermode ignore
place $top.canvas1.label2 \
-in $top.canvas1 -x 8 -y 50 -width 152 -height 24 -anchor nw \
-bordermode ignore
place $top.canvas1.check2 \
-in $top.canvas1 -x 175 -y 50 -anchor nw -bordermode ignore
place $top.canvas1.label3 \
-in $top.canvas1 -x 10 -y 85 -anchor nw -bordermode ignore
place $top.canvas1.check3 \
-in $top.canvas1 -x 175 -y 85 -anchor nw -bordermode ignore
place $top.canvas1.check3a \
-in $top.canvas1 -x 270 -y 85 -anchor nw -bordermode ignore
place $top.canvas1.check3b \
-in $top.canvas1 -x 273 -y 108 -width 41 -height 22 -anchor nw \
-bordermode ignore
place $top.canvas1.label4 \
-in $top.canvas1 -x 10 -y 145 -anchor nw -bordermode ignore
place $top.canvas1.check4 \
-in $top.canvas1 -x 190 -y 145 -anchor nw -bordermode ignore
place $top.title \
-in $top -x 10 -y 20 -anchor nw -bordermode ignore
vTcl:FireEvent $base <<Ready>>
}
proc vTclWindow.interactive_common {base} {
if {$base == ""} {
set base .interactive_common
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl:Toplevel $top -class Toplevel \
-highlightcolor black
wm withdraw $top
wm focusmodel $top passive
wm geometry $top
492x913+381+10; update
wm maxsize $top 1265 994
wm minsize $top
1 1
wm overrideredirect $top 0
wm resizable $top 0 0
wm title
$top "Common descriptors"
vTcl:DefineAlias "$top" "Toplevel3"
vTcl:Toplevel:WidgetProc "" 1
bindtags $top "$top Toplevel all
_TopLevel"
vTcl:FireEvent $top <<Create>>
wm protocol $top
WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"
frame
$top.frame1 \
-borderwidth 3 -relief groove -height 810 -width 470
vTcl:DefineAlias "$top.frame1" "Frame1" vTcl:WidgetProc "Toplevel3"
1
bindtags $top.frame1 "$top.frame1 Frame $top all _TopLevel"
set site_3_0 $top.frame1
label $site_3_0.label1 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Supply voltage of new
process (V)}
vTcl:DefineAlias "$site_3_0.label1" "Label1" vTcl:WidgetProc
"Toplevel3" 1
label $site_3_0.label2 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Package IR drop factor

```

```

(eta)
vTcl::DefineAlias "$site_3_0.label2" "Label2" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label3 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Min length in new process
(m)}
vTcl::DefineAlias "$site_3_0.label3" "Label4" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label4 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Gate cap scaling factor}
vTcl::DefineAlias "$site_3_0.label4" "Label5" vTcl::WidgetProc
"Toplevel3" 1
Separator $site_3_0.line1 \
-background #d6cddb -orient horizontal
vTcl::DefineAlias "$site_3_0.line1" "Separator1" vTcl::WidgetProc
"Toplevel3" 1
bind $site_3_0.line1 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3_0.label5 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Wire cap scaling factor}
vTcl::DefineAlias "$site_3_0.label5" "Label6" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label6 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Gate width scaling
factor}
vTcl::DefineAlias "$site_3_0.label6" "Label7" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label7 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Original unit area gate
cap (F/um2)}
vTcl::DefineAlias "$site_3_0.label7" "Label8" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label8 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Original min W/L drain
current (A)}
vTcl::DefineAlias "$site_3_0.label8" "Label9" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry1 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable vdd_spec
vTcl::DefineAlias "$site_3_0.entry1" "Entry1" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry2 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable eta
vTcl::DefineAlias "$site_3_0.entry2" "Entry2" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.temp1 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state readonly -textvariable vdd_bump
vTcl::DefineAlias "$site_3_0.temp1" "Entry3" vTcl::WidgetProc
"Toplevel3" 1
bindtags $site_3_0.temp1 "$site_3_0.temp1 Entry $top
all _Toplevel"
entry $site_3_0.entry3 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable l_min -validate focusin \
-validatecommand
{set vdd_bump [expr $eta * $vdd_spec]}
return 1}
vTcl::DefineAlias "$site_3_0.entry3" "Entry4" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry4 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable s_gate_cap
vTcl::DefineAlias "$site_3_0.entry4" "Entry5" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry5 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable s_wire_cap
vTcl::DefineAlias "$site_3_0.entry5" "Entry6" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry6 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable s_width
vTcl::DefineAlias "$site_3_0.entry6" "Entry7" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry7 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable c_unit_old
vTcl::DefineAlias "$site_3_0.entry7" "Entry8" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry8 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_ds_old
vTcl::DefineAlias "$site_3_0.entry8" "Entry9" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.cpd75 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable c_unit_new
vTcl::DefineAlias "$site_3_0.cpd75" "Entry10" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.cpd77 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_ds_new
vTcl::DefineAlias "$site_3_0.cpd77" "Entry11" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label_temp1 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-state disabled -text {Vdd_bump =
Vdd_supply x eta}
vTcl::DefineAlias "$site_3_0.label_temp1" "Label11" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label10 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Library redesign power
saving factor}
vTcl::DefineAlias "$site_3_0.label10" "Label3" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry10 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable rpsf
vTcl::DefineAlias "$site_3_0.entry10" "Entry12" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label11 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {RC interconnect slowdown
factor}
vTcl::DefineAlias "$site_3_0.label11" "Label12" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry11 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable rcsf
vTcl::DefineAlias "$site_3_0.entry11" "Entry13" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.cpd73 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {New chip's unit area
gate cap (F/um2)}
vTcl::DefineAlias "$site_3_0.cpd73" "Label13" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.cpd74 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {New chip's min W/L drain
current (A)}
vTcl::DefineAlias "$site_3_0.cpd74" "Label14" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label12 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Average statcking factor
for new design}
vTcl::DefineAlias "$site_3_0.label12" "Label15" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.entry12 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable sf
vTcl::DefineAlias "$site_3_0.entry12" "Entry14" vTcl::WidgetProc
"Toplevel3" 1
label $site_3_0.label13 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit decap sensitivity
(V/nF)}
vTcl::DefineAlias "$site_3_0.label13" "Label16" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3_0.cpd78 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \

```

```

-textvariable decap_sens
vTcl::DefineAlias "$site_3.0.cpd78" "Entry15" vTcl::WidgetProc
"Toplevel3" 1
Separator $site_3.0.line2 \
-background #d6cddb
vTcl::DefineAlias "$site_3.0.line2" "Separator2" vTcl::WidgetProc
"Toplevel3" 1
bind $site_3.0.line2 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3.0.heading2 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {New process's leakage
current estimates
from foundry or SPICE simulations}
vTcl::DefineAlias "$site_3.0.heading2" "Label17" vTcl::WidgetProc
"Toplevel3" 1
Separator $site_3.0.cpd79 \
-background #d6cddb
vTcl::DefineAlias "$site_3.0.cpd79" "Separator3" vTcl::WidgetProc
"Toplevel3" 1
bind $site_3.0.cpd79 <Destroy> {
Widget::destroy %W; rename %W {}
}
entry $site_3.0.entry16 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_leak_junc
vTcl::DefineAlias "$site_3.0.entry16" "Entry16" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3.0.cpd80 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_leak_gate
vTcl::DefineAlias "$site_3.0.cpd80" "Entry17" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3.0.cpd81 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_gate_per_w
vTcl::DefineAlias "$site_3.0.cpd81" "Entry18" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3.0.cpd82 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_othv
vTcl::DefineAlias "$site_3.0.cpd82" "Entry19" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3.0.cpd83 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_owhv
vTcl::DefineAlias "$site_3.0.cpd83" "Entry20" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3.0.cpd84 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_otlv
vTcl::DefineAlias "$site_3.0.cpd84" "Entry21" vTcl::WidgetProc
"Toplevel3" 1
Separator $site_3.0.line4 \
-background #d6cddb -orient horizontal
vTcl::DefineAlias "$site_3.0.line4" "Separator4" vTcl::WidgetProc
"Toplevel3" 1
bind $site_3.0.line4 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3.0.label17 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit junction area leakage
(A/um2)}
vTcl::DefineAlias "$site_3.0.label17" "Label18" vTcl::WidgetProc
"Toplevel3" 1
label $site_3.0.cpd86 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit gate area leakage
(A/um2)}
vTcl::DefineAlias "$site_3.0.cpd86" "Label19" vTcl::WidgetProc
"Toplevel3" 1
label $site_3.0.cpd87 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear gate leakage
current (A/um)}
vTcl::DefineAlias "$site_3.0.cpd87" "Label20" vTcl::WidgetProc
"Toplevel3" 1
label $site_3.0.cpd88 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, typical
I_off for high Vt FETs (A/um)}
vTcl::DefineAlias "$site_3.0.cpd88" "Label21" vTcl::WidgetProc
"Toplevel3" 1
entry $site_3.0.cpd91 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_owlv
vTcl::DefineAlias "$site_3.0.cpd91" "Entry22" vTcl::WidgetProc
"Toplevel3" 1
label $site_3.0.cpd93 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, worst I_off
for high Vt FETs (A/um)}
vTcl::DefineAlias "$site_3.0.cpd93" "Label22" vTcl::WidgetProc
"Toplevel3" 1
label $site_3.0.cpd94 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, typical
I_off for low Vt FETs (A/um)}
vTcl::DefineAlias "$site_3.0.cpd94" "Label23" vTcl::WidgetProc
"Toplevel3" 1
label $site_3.0.cpd95 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, worst I_off
for low Vt FETs (A/um)}
vTcl::DefineAlias "$site_3.0.cpd95" "Label24" vTcl::WidgetProc
"Toplevel3" 1
place $site_3.0.label1 \
-in $site_3.0 -x 5 -y 15 -width 258 -height 20 -anchor nw \
-bordermode ignore
place $site_3.0.label2 \
-in $site_3.0 -x 7 -y 40 -width 213 -height 20 -anchor nw \
-bordermode ignore
place $site_3.0.label3 \
-in $site_3.0 -x 4 -y 95 -width 228 -height 20 -anchor nw \
-bordermode ignore
place $site_3.0.label4 \
-in $site_3.0 -x 5 -y 127 -width 173 -height 20 -anchor nw \
-bordermode ignore
place $site_3.0.label1 \
-in $site_3.0 -x 299 -y 5 -width 2 -height 456 -anchor nw \
-bordermode ignore
place $site_3.0.label5 \
-in $site_3.0 -x 5 -y 155 -width 169 -height 24 -anchor nw \
-bordermode ignore
place $site_3.0.label6 \
-in $site_3.0 -x 6 -y 185 -width 184 -height 24 -anchor nw \
-bordermode ignore
place $site_3.0.label7 \
-in $site_3.0 -x 2 -y 215 -width 264 -height 24 -anchor nw \
-bordermode ignore
place $site_3.0.label8 \
-in $site_3.0 -x 5 -y 245 -width 248 -height 24 -anchor nw \
-bordermode ignore
place $site_3.0.entry1 \
-in $site_3.0 -x 305 -y 15 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3.0.entry2 \
-in $site_3.0 -x 305 -y 45 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3.0.templ \
-in $site_3.0 -x 305 -y 70 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3.0.entry3 \
-in $site_3.0 -x 305 -y 95 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3.0.entry4 \
-in $site_3.0 -x 305 -y 125 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3.0.entry5 \
-in $site_3.0 -x 305 -y 155 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3.0.entry6 \
-in $site_3.0 -x 305 -y 185 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3.0.entry7 \
-in $site_3.0 -x 305 -y 215 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3.0.entry8 \
-in $site_3.0 -x 305 -y 245 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3.0.cpd75 \
-in $site_3.0 -x 305 -y 274 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3.0.cpd77 \
-in $site_3.0 -x 305 -y 303 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3.0.label_templ \
-in $site_3.0 -x 60 -y 65 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3.0.label10 \

```



```

-in $site_3_0 -x 7 -y 330 -width 266 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry10 \
-in $site_3_0 -x 305 -y 332 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.label11 \
-in $site_3_0 -x 6 -y 360 -width 245 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry11 \
-in $site_3_0 -x 8 -y 305 -y 360 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd73 \
-in $site_3_0 -x 7 -y 388 -width 280 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd74 \
-in $site_3_0 -x 6 -y 303 -width 273 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.label12 \
-in $site_3_0 -x 7 -y 388 -width 292 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry12 \
-in $site_3_0 -x 305 -y 388 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.label13 \
-in $site_3_0 -x 9 -y 418 -width 201 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd78 \
-in $site_3_0 -x 305 -y 415 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.line2 \
-in $site_3_0 -x 10 -y 475 -width 456 -height 2 -anchor nw \
-bordermode ignore
place $site_3_0.heading2 \
-in $site_3_0 -x 90 -y 490 -width 307 -height 42 -anchor nw \
-bordermode ignore
place $site_3_0.cpd79 \
-in $site_3_0 -x 95 -y 540 -width 296 -height 2 -anchor nw \
-bordermode ignore
place $site_3_0.entry16 \
-in $site_3_0 -x 360 -y 575 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd80 \
-in $site_3_0 -x 360 -y 610 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd81 \
-in $site_3_0 -x 360 -y 640 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd82 \
-in $site_3_0 -x 360 -y 670 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd83 \
-in $site_3_0 -x 360 -y 705 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd84 \
-in $site_3_0 -x 360 -y 735 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.line4 \
-in $site_3_0 -x 355 -y 575 -width 2 -height 216 -anchor nw \
-bordermode ignore
place $site_3_0.label17 \
-in $site_3_0 -x 7 -y 580 -width 257 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd86 \
-in $site_3_0 -x 6 -y 610 -width 238 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd87 \
-in $site_3_0 -x 5 -y 640 -width 349 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd88 \
-in $site_3_0 -x 4 -y 675 -width 349 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd91 \
-in $site_3_0 -x 360 -y 765 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd93 \
-in $site_3_0 -x 7 -y 703 -width 345 -height 29 -anchor nw \
-bordermode ignore
place $site_3_0.cpd94 \

-in $site_3_0 -x 5 -y 735 -anchor nw -bordermode inside
place $site_3_0.cpd95 \
-in $site_3_0 -x 5 -y 765 -anchor nw -bordermode inside
button $top.butt1 \
\
-command {wm withdraw .interactive_common
set vdd_bump [expr $vdd_spec * $eta ]
set types {
{{Command Files} {cmd}
{{All Files} *
}
}
set new_cmd_filename [tk_getSaveFile -filetypes $types
-initialdir ./ -initialfile command_file -title "Enter file
name"]
wm deiconify .interactive_common
set new_cmd_file_id
[open $new_cmd_filename w]
puts $new_cmd_file_id "\# Begin
module descriptions \n"
puts $new_cmd_file_id "\# number of
modules \n"
puts $new_cmd_file_id ".mods $no_modules \n"
puts
$new_cmd_file_id "\# Common descriptor list \n"
puts $new_cmd_file_id "\#
Vdd_spec,ETA,Min_L,Gate_cap_scaling_factor,Wire_cap_scaling_factor,width_sclaing_factor,
+"
puts $new_cmd_file_id "\#
Old_unit_gate_cap,old_min_size_FET_ids,New_unit_gate_cap,new_min_size_FET_ids,Redesign_po
+"
puts $new_cmd_file_id "\#
RC_slowdown_factor,stacking_factor,Decap_sensitivity,unit_junc_leakage,unit_gate_leakage,
+"
puts $new_cmd_file_id "\#
per_unit_width_gate_leakage,typical_hi_vt_ioff,worst_hi_vt_ioff,typical_lo_vt_ioff,worst_
\n\n"
puts $new_cmd_file_id ".com.des $vdd_spec $eta $l_min $s_gate_cap
$s_wire_cap $s_width $c_unit_old $i_ds_old $c_unit_new $i_ds_new
$rcsf $rcsf $sf $decap_sens $i_leak_junc $i_leak_gate $i_gate_per_w
$i_othv $i_othv $i_othv $i_othv\n\n"
puts $new_cmd_file_id "\#
Begin individual module descriptions\n"
puts $new_cmd_file_id
"\# Individual module descriptor list\n"
puts $new_cmd_file_id "\#
Original_load_cap,total_wire_length,unit_wire_length_cap,total_PFET_W,
+"
puts $new_cmd_file_id "\#
total_NFET_W,original_operating_freq,average_switching_factor,
+"
puts $new_cmd_file_id "\#
clock_gating_factor,de_cap_added,ratio_hi_vt_FETS,wire_buff_cap,typical_path_FET_ratio,
+"
puts $new_cmd_file_id "\#
useful_skew_performance_enhancement,buffer_speed_up_factor,temperature,if_critical_module

close $new_cmd_file_id
wm withdraw .interactive_common
set c_orig
0
set twl 0
set ulc 0
set w_total_p 0
set w_total_n 0
set f_old 1

set asf 0
set cgf 0
set c_de_cap 0
set hvr 0
set c_wire_buff 0
set tprf 0
set uspe 0
set bsuf 0
set temperature 25
set cm_yes 1
set nadsp 0
wm deiconify .interactive_individual_i } \
-disabledforeground #a1a1a1 -text {Save and proceed}
vTcl::DefineAlias "Stop.butt1" "Button1" vTcl::WidgetProc "Toplevel3"
1
button $top.butt2 \
\
-command {set vdd_spec 0
set eta 0.95
set l_min 0
set s_gate_cap 0
set s_wire_cap 0
set s_width
0
set c_unit_old 0
set i_ds_old 0
set c_unit_new 0
set i_ds_new 0
set rpsf 0
set rcfsf 0
set sf 0
set decap_sens 0
set i_leak_junc 0
set i_leak_gate 0
set i_gate_per_w 0

```

```

set i_othv 0
set i_ovhv 0
set
i_othv 0
set i_othv 0 \
-disabledforeground #alalal -text {Clear All}
vTcl::DefineAlias "$stop.button2" "Button2" vTcl::WidgetProc "Toplevel3" 1
label $stop.heading \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Descriptors common to
all modules}
vTcl::DefineAlias "$stop.heading" "Label10" vTcl::WidgetProc "Toplevel3" 1
place $stop.frame1 \
-in $stop -x 10 -y 45 -width 470 -height 810 -anchor nw \

-bordermode ignore
place $stop.button1 \
-in $stop -x 295 -y 875 -anchor nw -bordermode ignore
place $stop.button2 \
-in $stop -x 70 -y 875 -anchor nw -bordermode ignore
place $stop.heading \
-in $stop -x 124 -y 19 -width 260 -height 24 -anchor nw \

-bordermode ignore
vTcl::FireEvent $base <<Ready>>
}
proc vTcl::Window::interactive_common_edit {base} {
if {$base == ""} {
set base .interactive_common_edit
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl::toplevel $stop -class Toplevel \
-highlightcolor black
wm withdraw $stop
wm focusmodel $stop passive
wm geometry $stop
492x906+413+11; update
wm maxsize $stop 1265 994
wm minsize $stop 1
1
wm overrideredirect $stop 0
wm resizable $stop 1
wm title $stop
"Edit common descriptors"
vTcl::DefineAlias "$stop" "Toplevel7"
vTcl::Toplevel::WidgetProc "" 1
bindtags $stop "$stop Toplevel all
_TopLevel"
vTcl::FireEvent $stop <<Create>>
wm protocol $stop
WM_DELETE_WINDOW "vTcl::FireEvent $stop <<DeleteWindow>>"
frame
$stop.frame1 \
-borderwidth 3 -relief groove -height 795 -width 470
vTcl::DefineAlias "$stop.frame1" "Frame1" vTcl::WidgetProc "Toplevel7" 1
bindtags $stop.frame1 "$stop.frame1 Frame $stop all _TopLevel"

set site_3_0 $stop.frame1
label $site_3_0.label1 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Supply voltage of new
process (V)}
vTcl::DefineAlias "$site_3_0.label1" "Label1" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label2 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Package IR drop factor
(eta)}
vTcl::DefineAlias "$site_3_0.label2" "Label2" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label3 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Min length in new process
(m)}
vTcl::DefineAlias "$site_3_0.label3" "Label4" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label4 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Gate cap scaling factor}
vTcl::DefineAlias "$site_3_0.label4" "Label5" vTcl::WidgetProc
"Toplevel7" 1
Separator $site_3_0.line1 \
-background #d6cddb -orient horizontal
vTcl::DefineAlias "$site_3_0.line1" "Separator1" vTcl::WidgetProc
"Toplevel7" 1
bind $site_3_0.line1 <Destroy> {

Widget::destroy %W; rename %W {}
}
label $site_3_0.label5 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Wire cap scaling factor}
vTcl::DefineAlias "$site_3_0.label5" "Label6" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label6 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Gate width scaling
factor}
vTcl::DefineAlias "$site_3_0.label6" "Label7" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label7 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Original unit area gate
cap (F/um2)}
vTcl::DefineAlias "$site_3_0.label7" "Label8" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label8 \
-disabledforeground #alalal \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-text {Original min W/L drain
current (A)}
vTcl::DefineAlias "$site_3_0.label8" "Label9" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry1 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable vdd_spec
vTcl::DefineAlias "$site_3_0.entry1" "Entry1" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry2 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable eta
vTcl::DefineAlias "$site_3_0.entry2" "Entry2" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.temp1 \
-background white -disabledforeground #alalal -insertbackground
black \
-state readonly -textvariable vdd_bump
vTcl::DefineAlias "$site_3_0.temp1" "Entry3" vTcl::WidgetProc
"Toplevel7" 1
bindtags $site_3_0.temp1 "$site_3_0.temp1 Entry $stop
all _TopLevel"
entry $site_3_0.entry3 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable l_min
vTcl::DefineAlias "$site_3_0.entry3" "Entry4" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry4 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable s_gate_cap
vTcl::DefineAlias "$site_3_0.entry4" "Entry5" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry5 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable s_wire_cap
vTcl::DefineAlias "$site_3_0.entry5" "Entry6" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry6 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable s_width
vTcl::DefineAlias "$site_3_0.entry6" "Entry7" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry7 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable c_unit_old
vTcl::DefineAlias "$site_3_0.entry7" "Entry8" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.entry8 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable i_ds_old
vTcl::DefineAlias "$site_3_0.entry8" "Entry9" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd75 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable c_unit_new
vTcl::DefineAlias "$site_3_0.cpd75" "Entry10" vTcl::WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd77 \
-background white -disabledforeground #alalal -insertbackground
black \
-textvariable i_ds_new
vTcl::DefineAlias "$site_3_0.cpd77" "Entry11" vTcl::WidgetProc
"Toplevel7" 1
label $site_3_0.label_temp1 \

```

```

-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-state disabled -text {Vdd_bump =
Vdd_supply x eta}
vTcl:DefineAlias "$site_3_0.label_temp1" "Label11" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.label10 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Library redesign power
saving factor}
vTcl:DefineAlias "$site_3_0.label10" "Label3" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.entry10 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable rpsf
vTcl:DefineAlias "$site_3_0.entry10" "Entry12" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.label11 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {RC interconnect slowdown
factor}
vTcl:DefineAlias "$site_3_0.label11" "Label12" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.entry11 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable rcsf
vTcl:DefineAlias "$site_3_0.entry11" "Entry13" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd73 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {New chip's unit area
gate cap (F/um2)}
vTcl:DefineAlias "$site_3_0.cpd73" "Label13" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd74 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {New chip's min W/L drain
current (A)}
vTcl:DefineAlias "$site_3_0.cpd74" "Label14" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.label12 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Average statcking factor
for new design}
vTcl:DefineAlias "$site_3_0.label12" "Label15" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.entry12 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable sf
vTcl:DefineAlias "$site_3_0.entry12" "Entry14" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.label13 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit decap sensitivity
(V/nF)}
vTcl:DefineAlias "$site_3_0.label13" "Label16" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd78 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable decap_sens
vTcl:DefineAlias "$site_3_0.cpd78" "Entry15" vTcl:WidgetProc
"Toplevel7" 1
Separator $site_3_0.line2 \
-background #d6cddb
vTcl:DefineAlias "$site_3_0.line2" "Separator2" vTcl:WidgetProc
"Toplevel7" 1
bind $site_3_0.line2 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3_0.heading2 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {New process's leakage
current estimates
from foundry or SPICE simulations}
vTcl:DefineAlias "$site_3_0.heading2" "Label17" vTcl:WidgetProc
"Toplevel7" 1
Separator $site_3_0.cpd79 \
-background #d6cddb
vTcl:DefineAlias "$site_3_0.cpd79" "Separator3" vTcl:WidgetProc
"Toplevel7" 1
bind $site_3_0.cpd79 <Destroy> {
Widget::destroy %W; rename %W {}
}

entry $site_3_0.entry16 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_leak_junc
vTcl:DefineAlias "$site_3_0.entry16" "Entry16" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd80 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_leak_gate
vTcl:DefineAlias "$site_3_0.cpd80" "Entry17" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd81 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_gate_per_w
vTcl:DefineAlias "$site_3_0.cpd81" "Entry18" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd82 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_othv
vTcl:DefineAlias "$site_3_0.cpd82" "Entry19" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd83 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_owhv
vTcl:DefineAlias "$site_3_0.cpd83" "Entry20" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd84 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_otlv
vTcl:DefineAlias "$site_3_0.cpd84" "Entry21" vTcl:WidgetProc
"Toplevel7" 1
Separator $site_3_0.line4 \
-background #d6cddb -orient horizontal
vTcl:DefineAlias "$site_3_0.line4" "Separator4" vTcl:WidgetProc
"Toplevel7" 1
bind $site_3_0.line4 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3_0.label17 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit junction area leakage
(A/um2)}
vTcl:DefineAlias "$site_3_0.label17" "Label18" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd86 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit gate area leakage
(A/um2)}
vTcl:DefineAlias "$site_3_0.cpd86" "Label19" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd87 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear gate leakage
current (A/um2)}
vTcl:DefineAlias "$site_3_0.cpd87" "Label20" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd88 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, typical
I_off for high Vt FETs (A/um)}
vTcl:DefineAlias "$site_3_0.cpd88" "Label21" vTcl:WidgetProc
"Toplevel7" 1
entry $site_3_0.cpd91 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable i_owlv
vTcl:DefineAlias "$site_3_0.cpd91" "Entry22" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd93 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, worst I_off
for high Vt FETs (A/um)}
vTcl:DefineAlias "$site_3_0.cpd93" "Label22" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd94 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, typical
I_off for low Vt FETs (A/um)}
vTcl:DefineAlias "$site_3_0.cpd94" "Label23" vTcl:WidgetProc
"Toplevel7" 1
label $site_3_0.cpd95 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit linear, worst I_off

```

```

for low Vt FETs (A/um)}
vTcl:DefineAlias "$site_3_0.cpd95" "Label24" vTcl:WidgetProc
"Toplevel7" 1
place $site_3_0.label11 \
-in $site_3_0 -x 5 -y 15 -width 258 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.label12 \
-in $site_3_0 -x 7 -y 40 -width 213 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.label13 \
-in $site_3_0 -x 4 -y 95 -width 228 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.label14 \
-in $site_3_0 -x 5 -y 127 -width 173 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.line1 \
-in $site_3_0 -x 299 -y 5 -width 2 -height 466 -anchor nw \
-bordermode ignore
place $site_3_0.label15 \
-in $site_3_0 -x 5 -y 155 -width 169 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.label16 \
-in $site_3_0 -x 6 -y 185 -width 184 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.label17 \
-in $site_3_0 -x 2 -y 215 -width 264 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.label18 \
-in $site_3_0 -x 5 -y 245 -width 248 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry1 \
-in $site_3_0 -x 305 -y 15 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.entry2 \
-in $site_3_0 -x 305 -y 45 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.temp1 \
-in $site_3_0 -x 305 -y 70 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.entry3 \
-in $site_3_0 -x 305 -y 95 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.entry4 \
-in $site_3_0 -x 305 -y 125 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3_0.entry5 \
-in $site_3_0 -x 305 -y 155 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3_0.entry6 \
-in $site_3_0 -x 305 -y 185 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3_0.entry7 \
-in $site_3_0 -x 305 -y 215 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3_0.entry8 \
-in $site_3_0 -x 305 -y 245 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3_0.cpd75 \
-in $site_3_0 -x 305 -y 274 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd77 \
-in $site_3_0 -x 305 -y 303 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.label_temp1 \
-in $site_3_0 -x 60 -y 65 -width 148 -height 22 -anchor nw -bordermode ignore
place $site_3_0.label10 \
-in $site_3_0 -x 7 -y 330 -width 266 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry10 \
-in $site_3_0 -x 305 -y 332 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.label11 \
-in $site_3_0 -x 6 -y 360 -width 245 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry11 \
-in $site_3_0 -x 305 -y 360 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd73 \
-in $site_3_0 -x 8 -y 274 -width 280 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd74 \
-in $site_3_0 -x 6 -y 303 -width 273 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.label12 \
-in $site_3_0 -x 7 -y 388 -width 292 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.entry12 \
-in $site_3_0 -x 305 -y 388 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.label13 \
-in $site_3_0 -x 9 -y 418 -width 201 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd78 \
-in $site_3_0 -x 305 -y 415 -width 148 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.line2 \
-in $site_3_0 -x 7 -y 480 -width 456 -height 2 -anchor nw \
-bordermode ignore
place $site_3_0.heading2 \
-in $site_3_0 -x 95 -y 490 -width 307 -height 42 -anchor nw \
-bordermode ignore
place $site_3_0.cpd79 \
-in $site_3_0 -x 100 -y 540 -width 296 -height 2 -anchor nw \
-bordermode ignore
place $site_3_0.entry16 \
-in $site_3_0 -x 358 -y 580 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd80 \
-in $site_3_0 -x 358 -y 610 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd81 \
-in $site_3_0 -x 358 -y 640 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd82 \
-in $site_3_0 -x 358 -y 670 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd83 \
-in $site_3_0 -x 358 -y 700 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd84 \
-in $site_3_0 -x 358 -y 730 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.line4 \
-in $site_3_0 -x 355 -y 570 -width 2 -height 216 -anchor nw \
-bordermode ignore
place $site_3_0.label17 \
-in $site_3_0 -x 9 -y 580 -width 257 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd86 \
-in $site_3_0 -x 7 -y 610 -width 238 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd87 \
-in $site_3_0 -x 6 -y 643 -width 296 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd88 \
-in $site_3_0 -x 6 -y 670 -width 349 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd91 \
-in $site_3_0 -x 358 -y 759 -width 100 -height 22 -anchor nw \
-bordermode ignore
place $site_3_0.cpd93 \
-in $site_3_0 -x 7 -y 698 -width 345 -height 29 -anchor nw \
-bordermode ignore
place $site_3_0.cpd94 \
-in $site_3_0 -x 9 -y 728 -width 343 -height 24 -anchor nw \
-bordermode ignore
place $site_3_0.cpd95 \
-in $site_3_0 -x 5 -y 755 -width 148 -height 22 -anchor nw -bordermode inside
button $stop.button1 \
\
-command {wm withdraw .interactive_common_edit
set vdd_bump [expr $vdd_spec * $eta ]
set newline "$vdd_spec $eta
$1_min $s_gate_cap $s_wire_cap $s_width $c_unit_old $i_ds_old
$c_unit_new $i_ds_new $rpsf $rcsf $sf $decap_sens $i_leak_junc
$i_leak_gate $i_gate_per_w $i_othv $i_ovhv $i_otlv $i_owlv"
catch {exec
\.\./scripts/swapline.pl com $newline $cmd_file_to_edit > temp_cmd_file
} junk
catch {exec cp temp_cmd_file $cmd_file_to_edit } junk2
set x
0
set curr_mod.is_final 0
catch {exec more $cmd_file_to_edit | grep
"mods_des 1" } read_mod_des
set uu [split $read_mod_des " "]
set

```

```

c_orig [lindex $uu 2]
set twl [lindex $uu 3]
set ulc [lindex $uu 4]

set w_total_p [lindex $uu 5]
set w_total_n [lindex $uu 6]
set f_old
[lindex $uu 7]
set delay [expr 1/$f_old]
set asf [lindex $uu 8]
set
cgf [lindex $uu 9]
set c_de_cap [lindex $uu 10]
set hvr [lindex $uu
11]
set lvr [expr 1-$hvr]
set c_wire_buff [lindex $uu 12]
set tpfr
[lindex $uu 13]
set uspe [lindex $uu 14]
set bsuf [lindex $uu 15]

set temperature [lindex $uu 16]
set cm_yes [lindex $uu 17]
set
nadsp [lindex $uu 18]
set choose_edit "Edit / Skip"
wm deiconify
.interactive_individual_1_edit
-disabledforeground #a1a1a1 -text {Save and proceed}
vTcl:DefineAlias "$stop.butt1" "Button1" vTcl:WidgetProc "Toplevel4" 1
button $stop.butt2 \
\
-command {set vdd_spec 0
set eta 0.95
set l_min 0
set s_gate_cap 0
set s_wire_cap 0
set s_width
0
set c_unit_old 0
set i_ds_old 0
set c_unit_new 0
set i_ds_new 0
set rpsf 0
set rcsf 0
set sf 0
set decap_sens 0
set i_leak_junc 0
set i_leak_gate 0
set i_gate_per_w 0
set i_othv 0
set i_owhv 0
set
i_otlv 0
set i_owlv 0} \
-disabledforeground #a1a1a1 -text {Clear All}
vTcl:DefineAlias "$stop.butt2" "Button2" vTcl:WidgetProc "Toplevel4" 1
label $stop.heading \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Descriptors common to
all modules}
vTcl:DefineAlias "$stop.heading" "Label10" vTcl:WidgetProc "Toplevel4" 1
place $stop.frame1 \
-in $stop -x 10 -y 55 -width 470 -height 795 -anchor nw \
-bordermode ignore
place $stop.butt1 \
-in $stop -x 295 -y 865 -width 137 -height 28 -anchor nw \
-bordermode ignore
place $stop.butt2 \
-in $stop -x 70 -y 865 -anchor nw -bordermode ignore
place $stop.heading \
-in $stop -x 124 -y 19 -width 260 -height 24 -anchor nw \
-bordermode ignore
vTcl:FireEvent $base <<Ready>>
}
proc vTclWindow.interactive_individual_1 {base} {
if {$base == ""} {
set base .interactive_individual_1
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl:toplevel $stop -class Toplevel \
-highlightcolor black
wm withdraw $stop
wm focusmodel $stop passive
wm geometry $stop
488x697+366+128; update
wm maxsize $stop 1265 994
wm minsize $stop
1 1
wm overrideredirect $stop 0
wm resizable $stop 0 0
wm title
$stop "Individual module descriptors"
vTcl:DefineAlias "$stop"
"Toplevel4" vTcl:Toplevel:WidgetProc "" 1
bindtags $stop $stop
Toplevel all _Toplevel
vTcl:FireEvent $stop <<Create>>
wm protocol
$stop WM_DELETE_WINDOW "vTcl:FireEvent $stop <<DeleteWindow>>"
frame
$stop.frame1 \
-borderwidth 3 -relief groove -height 590 -width 470
vTcl:DefineAlias "$stop.frame1" "Frame1" vTcl:WidgetProc "Toplevel4" 1
bindtags $stop.frame1 "$stop.frame1 Frame $stop all _Toplevel"

set site_3_0 $stop.frame1
label $site_3_0.label1 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Original load capacitance
(F)}
vTcl:DefineAlias "$site_3_0.label1" "Label1" vTcl:WidgetProc
"Toplevel4" 1
label $site_3_0.label2 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Total wire length (m)}
vTcl:DefineAlias "$site_3_0.label2" "Label2" vTcl:WidgetProc
"Toplevel4" 1
label $site_3_0.label3 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit length wire cap,
all metal layers (F)}
vTcl:DefineAlias "$site_3_0.label3" "Label4" vTcl:WidgetProc
"Toplevel4" 1
label $site_3_0.label4 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Total PFET width (m)}
vTcl:DefineAlias "$site_3_0.label4" "Label5" vTcl:WidgetProc
"Toplevel4" 1
Separator $site_3_0.line1 \
-background #d6cddb -orient horizontal
vTcl:DefineAlias "$site_3_0.line1" "Separator1" vTcl:WidgetProc
"Toplevel4" 1
bind $site_3_0.line1 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3_0.label5 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Total NFET width (m)}
vTcl:DefineAlias "$site_3_0.label5" "Label6" vTcl:WidgetProc
"Toplevel4" 1
label $site_3_0.label6 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Original operating
frequency (Hz)}
vTcl:DefineAlias "$site_3_0.label6" "Label7" vTcl:WidgetProc
"Toplevel4" 1
label $site_3_0.label7 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Average switching
factor}
vTcl:DefineAlias "$site_3_0.label7" "Label8" vTcl:WidgetProc
"Toplevel4" 1
label $site_3_0.label8 \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Clock gating factor}
vTcl:DefineAlias "$site_3_0.label8" "Label9" vTcl:WidgetProc
"Toplevel4" 1
entry $site_3_0.entry1 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable c_orig
vTcl:DefineAlias "$site_3_0.entry1" "Entry1" vTcl:WidgetProc
"Toplevel4" 1
entry $site_3_0.entry2 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable twl
vTcl:DefineAlias "$site_3_0.entry2" "Entry2" vTcl:WidgetProc
"Toplevel4" 1
entry $site_3_0.entry3 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-textvariable ulc -validate none
vTcl:DefineAlias "$site_3_0.entry3" "Entry4" vTcl:WidgetProc

```

```

    "Toplevel4" 1
    entry $site_3_0.entry4 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable w_total_p
    vTcl::DefineAlias "$site_3_0.entry4" "Entry5" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry5 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable w_total_n
    vTcl::DefineAlias "$site_3_0.entry5" "Entry6" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry6 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable f_old
    vTcl::DefineAlias "$site_3_0.entry6" "Entry7" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry7 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable asf -validate focusin \
    -validatecommand {set delay [expr 1/$f_old]}
    return 1}
    vTcl::DefineAlias "$site_3_0.entry7" "Entry8" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry8 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable cgf
    vTcl::DefineAlias "$site_3_0.entry8" "Entry9" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.cpd75 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable c_de_cap
    vTcl::DefineAlias "$site_3_0.cpd75" "Entry10" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.cpd77 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable hvr
    vTcl::DefineAlias "$site_3_0.cpd77" "Entry11" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.label10 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -state disabled -text {Ratio
    if Lo-to-Hi Vt FETS}
    vTcl::DefineAlias "$site_3_0.label10" "Label3" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry10 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -state readonly -textvariable lvr
    vTcl::DefineAlias "$site_3_0.entry10" "Entry12" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.label11 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Wire buffers (% of total
    FET width)}
    vTcl::DefineAlias "$site_3_0.label11" "Label12" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry11 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable c_wire_buff -validate focusin \
    -validatecommand {set lvr [expr 1-$hvr]}
    return 1}
    vTcl::DefineAlias "$site_3_0.entry11" "Entry13" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.cpd73 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Decap added (% of total
    FET width)}
    vTcl::DefineAlias "$site_3_0.cpd73" "Label13" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.cpd74 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Ratio of Hi-to-Lo Vt
    FETS}
    vTcl::DefineAlias "$site_3_0.cpd74" "Label14" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.label12 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Typical path FET ratio}
    vTcl::DefineAlias "$site_3_0.label12" "Label15" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry12 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable tpfr
    vTcl::DefineAlias "$site_3_0.entry12" "Entry14" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.label13 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Useful skew perf
    enhancement}
    vTcl::DefineAlias "$site_3_0.label13" "Label16" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.cpd78 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable uspe
    vTcl::DefineAlias "$site_3_0.cpd78" "Entry15" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry16 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable bsuf
    vTcl::DefineAlias "$site_3_0.entry16" "Entry16" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.cpd80 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable temperature
    vTcl::DefineAlias "$site_3_0.cpd80" "Entry17" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.label17 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Interconnect speedup
    due to buffers}
    vTcl::DefineAlias "$site_3_0.label17" "Label18" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.cpd86 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Temperature of
    operation}
    vTcl::DefineAlias "$site_3_0.cpd86" "Label19" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.cpd85 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -state disabled -text {Delay
    (1/freq)}
    vTcl::DefineAlias "$site_3_0.cpd85" "Label11" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.cpd79 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {Critical module?}
    vTcl::DefineAlias "$site_3_0.cpd79" "Label20" vTcl::WidgetProc
    "Toplevel4" 1
    checkbox $site_3_0.check1 \
    -disabledforeground #a1a1a1 -text {YES / NO} -variable cm_yes
    vTcl::DefineAlias "$site_3_0.check1" "Checkbox1" vTcl::WidgetProc
    "Toplevel4" 1
    bindtags $site_3_0.check1 $site_3_0.check1 Checkbutton
    stop all _vTclBalloon"
    bind $site_3_0.check1 <<SetBalloon>> {
    set ::vTcl::balloon::W {Check if critical module}
    }
    entry $site_3_0.cpd76 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -state readonly -textvariable delay
    vTcl::DefineAlias "$site_3_0.cpd76" "Entry7a" vTcl::WidgetProc
    "Toplevel4" 1
    entry $site_3_0.entry18 \
    -background white -disabledforeground #a1a1a1 -insertbackground
    black \
    -textvariable nadsp
    vTcl::DefineAlias "$site_3_0.entry18" "Entry18" vTcl::WidgetProc
    "Toplevel4" 1
    label $site_3_0.cpd90 \
    -disabledforeground #a1a1a1 \
    -font [vTcl::font:getFontFromDescr
    "-family helvetica -size 12"] \
    -text {NADSP - Vdd droop (%
    of Vdd)}
    vTcl::DefineAlias "$site_3_0.cpd90" "Label21" vTcl::WidgetProc
    "Toplevel4" 1
    place $site_3_0.label1 \
    -in $site_3_0 -x 3 -y 15 -width 213 -height 20 -anchor nw \
    -bordermode ignore
    place $site_3_0.label2 \
    -in $site_3_0 -x 7 -y 43 -width 148 -height 20 -anchor nw \
    -bordermode ignore
    place $site_3_0.label3 \
    -in $site_3_0 -x 5 -y 72 -width 293 -height 20 -anchor nw \
    -bordermode ignore

```

```

        place $site_3.0.label14 \
-in $site_3.0 -x 5 -y 100 -width 163 -height 20 -anchor nw \
        -bordermode ignore
        place $site_3.0.line1 \
-in $site_3.0 -x 299 -y 5 -width 2 -height 576 -anchor nw \
        -bordermode ignore
        place $site_3.0.label15 \
-in $site_3.0 -x 5 -y 130 -width 164 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.label16 \
-in $site_3.0 -x 5 -y 160 -width 244 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.label17 \
-in $site_3.0 -x 2 -y 215 -width 189 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.label18 \
-in $site_3.0 -x 5 -y 245 -width 143 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry1 \
-in $site_3.0 -x 305 -y 15 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry2 \
-in $site_3.0 -x 305 -y 45 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry3 \
-in $site_3.0 -x 305 -y 72 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry4 \
-in $site_3.0 -x 305 -y 100 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry5 \
-in $site_3.0 -x 305 -y 130 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry6 \
-in $site_3.0 -x 305 -y 160 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry7 \
-in $site_3.0 -x 305 -y 215 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry8 \
-in $site_3.0 -x 305 -y 245 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd75 \
-in $site_3.0 -x 305 -y 274 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd77 \
-in $site_3.0 -x 305 -y 303 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.label10 \
-in $site_3.0 -x 7 -y 330 -width 186 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry10 \
-in $site_3.0 -x 305 -y 332 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.label11 \
-in $site_3.0 -x 6 -y 360 -width 255 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry11 \
-in $site_3.0 -x 305 -y 360 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd73 \
-in $site_3.0 -x 8 -y 274 -width 260 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd74 \
-in $site_3.0 -x 6 -y 303 -width 188 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.label12 \
-in $site_3.0 -x 4 -y 390 -width 167 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry12 \
-in $site_3.0 -x 305 -y 390 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.label13 \
-in $site_3.0 -x 4 -y 418 -width 231 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd78 \
-in $site_3.0 -x 305 -y 419 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry16 \
-in $site_3.0 -x 305 -y 449 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd80 \
-in $site_3.0 -x 305 -y 479 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.label17 \
-in $site_3.0 -x 5 -y 449 -width 267 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd86 \
-in $site_3.0 -x 5 -y 479 -width 188 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd85 \
-in $site_3.0 -x 4 -y 188 -width 106 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd79 \
-in $site_3.0 -x 7 -y 511 -width 121 -height 24 -anchor nw \
        -bordermode ignore
        place $site_3.0.check1 \
-in $site_3.0 -x 335 -y 510 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd76 \
-in $site_3.0 -x 304 -y 187 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.entry18 \
-in $site_3.0 -x 305 -y 540 -width 148 -height 22 -anchor nw \
        -bordermode ignore
        place $site_3.0.cpd90 \
-in $site_3.0 -x 5 -y 540 -width 148 -height 22 -anchor nw \
        -bordermode inside
        button $stop.but1 \
        \
        -command {list vTcl:DoCmdOption $stop.but1 {incr x
set delay [expr 1/$f_old]
set lvr [expr 1-$hvr]
set new_cmd_file_id
[open $new_cmd_filename a]
puts $new_cmd_file_id ".mods_des $x
$c_orig $twl $ulc $w_total_p $w_total_n $f_old $asf $cgf $c_de_cap
$hvr $c_wire_buff $tpfr $suspe $bsuf $stemp $cm_yes $nadsp\n"
close $new_cmd_file_id
if {[expr $no_modules + 1]} {
set xi
[expr 0.01/$xi]
set yi [expr 0.01/$no_modules]
set y [expr {$yi/$xi}
* 100]
set module_entry_percentage $y
set module_entry_step "$x out
of $no_modules entered"
}
if {$x == $no_modules} {
wm withdraw
.interactive_individual_1
tk_messageBox -title "EIDA information"
-type ok -message "Module description complete !\nBegin Vdd scaling
characterization" -icon info
set alpha 0
set beta 0
set gamma 0

set delta 0
set epsilon 0
set fsf_x 0
set fsf_y 0
wm deiconify
.interactive_process
set x 0
set cm_yes 1
} else {
set c_orig 0

set twl 0
set ulc 0
set w_total_p 0
set w_total_n 0
set f_old 1

set delay [expr 1/$f_old]
set asf 0
set cgf 0
set c_de_cap 0
set
hvr 0
set lvr [expr 1-$hvr]
set c_wire_buff 0
set tpfr 0
set uspe 0

set bsuf 0
set temperature 25
set cm_yes 1
set nadsp 0
wm withdraw
.interactive_individual_1
wm deiconify .interactive_individual_1
}}
\
-disableforeground #a1a1a1 -text {Save and proceed}
vTcl:DefineAlias $stop.but1 "Button1" vTcl:WidgetProc "Toplevel4"
1

```

```

button $top.butt2 \
\
-command {set vdd_spec 0
set eta 0.95
set i_min 0
set s_gate_cap 0
set s_wire_cap 0
set s_width 0
0
set c_unit_old 0
set i_ds_old 0
set c_unit_new 0
set i_ds_new 0
set rpsf 0
set rcsf 0
set sf 0
set decap_sens 0
set i_leak_junc 0
set i_leak_gate 0
set i_gate_per_w 0
set i_othv 0
set i_owhv 0
set
i_otlv 0
set i_owlv 0} \
-vTcl::DefineAlias "$top.butt2" "Button2" vTcl::WidgetProc "Toplevel4" 1
1
label $top.heading \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Individual module
descriptors}
vTcl::DefineAlias "$top.heading" "Label10" vTcl::WidgetProc "Toplevel4" 1
1
Progressbar $top.cpd73 \
-background #d6cddb -foreground #000099 -height 15 -maximum
100 \
-relief raised -troughcolor #d9d9d9 -variable
module_entry_percentage
vTcl::DefineAlias "$top.cpd73" "Progressbar1" vTcl::WidgetProc
"Toplevel4" 1
entry $top.cpd74 \
-background white -disabledforeground #alalal -insertbackground
black \
-relief groove -state readonly -textvariable
module_entry_stop
vTcl::DefineAlias "$top.cpd74" "Entry3" vTcl::WidgetProc "Toplevel4" 1
1
place $top.frame1 \
-in $top -x 10 -y 50 -width 470 -height 590 -anchor nw \
-bordermode ignore
place $top.butt1 \
-in $top -x 295 -y 660 -width 137 -height 28 -anchor nw \
-bordermode ignore
place $top.butt2 \
-in $top -x 70 -y 660 -width 78 -height 28 -anchor nw \
-bordermode ignore
place $top.heading \
-in $top -x 15 -y 10 -width 205 -height 24 -anchor nw \
-bordermode ignore
place $top.cpd73 \
-in $top -x 265 -y 5 -width 220 -height 15 -anchor nw \
-bordermode inside
place $top.cpd74 \
-in $top -x 265 -y 25 -width 218 -height 22 -anchor nw \
-bordermode inside
vTcl::FireEvent $base <<Ready>>
}
proc vTclWindow.interactive_individual_1_edit {base} {
if {$base == ""} {
set base .interactive_individual_1_edit
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl::toplevel $top -class Toplevel \
-highlightcolor black
wm withdraw $top
wm focusmodel $top passive
wm geometry $top
488x693+341+114; update
wm maxsize $top 1265 994
wm minsize
$top 1 1
wm overrideredirect $top 0
wm resizable $top 1 1
wm
title $top "Edit individual module descriptors"
vTcl::DefineAlias
"$top" "Toplevel8" vTcl::Toplevel::WidgetProc "" 1
bindtags $top
"$top Toplevel all _TopLevel"
vTcl::FireEvent $top <<Create>>

wm
protocol $top WM_DELETE_WINDOW "vTcl::FireEvent $top <<DeleteWindow>>"

frame $top.frame1 \
-borderwidth 3 -relief groove -height 590 -width 470
vTcl::DefineAlias "$top.frame1" "Frame1" vTcl::WidgetProc "Toplevel8"
1
bindtags $top.frame1 "$top.frame1 Frame $top all _TopLevel"

set site_3_0 $top.frame1
label $site_3_0.label1 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Original load capacitance
(F)}
vTcl::DefineAlias "$site_3_0.label1" "Label1" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label2 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Total wire length (m)}
vTcl::DefineAlias "$site_3_0.label2" "Label2" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label3 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Unit length wire cap,
all metal layers (F)}
vTcl::DefineAlias "$site_3_0.label3" "Label4" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label4 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Total PFET width (m)}
vTcl::DefineAlias "$site_3_0.label4" "Label5" vTcl::WidgetProc
"Toplevel8" 1
Separator $site_3_0.line1 \
-background #d6cddb -orient horizontal
vTcl::DefineAlias "$site_3_0.line1" "Separator1" vTcl::WidgetProc
"Toplevel8" 1
bind $site_3_0.line1 <Destroy> {
Widget::destroy %W; rename %W {}
}
label $site_3_0.label5 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Total NFET width (m)}
vTcl::DefineAlias "$site_3_0.label5" "Label6" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label6 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Original operating
frequency (Hz)}
vTcl::DefineAlias "$site_3_0.label6" "Label7" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label7 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Average switching
factor}
vTcl::DefineAlias "$site_3_0.label7" "Label8" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label8 \
-disabledforeground #alalal \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Clock gating factor}
vTcl::DefineAlias "$site_3_0.label8" "Label9" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry1 \
-background white -disabledforeground #alalal -insertbackground
black \
-state disabled -textvariable c_orig
vTcl::DefineAlias "$site_3_0.entry1" "Entry1" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry2 \
-background white -disabledforeground #alalal -insertbackground
black \
-state disabled -textvariable tw1
vTcl::DefineAlias "$site_3_0.entry2" "Entry2" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry3 \
-background white -disabledforeground #alalal -insertbackground
black \
-state disabled -textvariable ulc -validate none
vTcl::DefineAlias "$site_3_0.entry3" "Entry4" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry4 \
-background white -disabledforeground #alalal -insertbackground
black \
-state disabled -textvariable w_total_p
vTcl::DefineAlias "$site_3_0.entry4" "Entry5" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry5 \

```



```

-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable v_total_n
vTcl::DefineAlias "$site_3_0.entry5" "Entry6" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry6 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable f_old
vTcl::DefineAlias "$site_3_0.entry6" "Entry7" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry7 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable asf -validate focusin \

-validatecommand {set delay [expr 1/$f_old]}
return 1}
vTcl::DefineAlias "$site_3_0.entry7" "Entry8" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry8 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable cgf
vTcl::DefineAlias "$site_3_0.entry8" "Entry9" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.cpd75 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable c_de_cap
vTcl::DefineAlias "$site_3_0.cpd75" "Entry10" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.cpd77 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable hvr
vTcl::DefineAlias "$site_3_0.cpd77" "Entry11" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label10 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-state disabled -text {Ratio
of Lo-to-Hi Vt FETS}
vTcl::DefineAlias "$site_3_0.label10" "Label3" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry10 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state readonly -textvariable lvr
vTcl::DefineAlias "$site_3_0.entry10" "Entry12" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label11 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Wire buffers (% of total
FET width)}
vTcl::DefineAlias "$site_3_0.label11" "Label12" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry11 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable c_wire_buff -validate
focusin \
-validatecommand {set lvr [expr 1-$hvr]}
return 1}
vTcl::DefineAlias "$site_3_0.entry11" "Entry13" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.cpd73 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Decap added (% of total
FET width)}
vTcl::DefineAlias "$site_3_0.cpd73" "Label13" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.cpd74 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Typical path FET ratio}
vTcl::DefineAlias "$site_3_0.cpd74" "Label14" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label12 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Ratio of Hi-to-Lo Vt
FETS}
vTcl::DefineAlias "$site_3_0.cpd74" "Label14" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label12 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Typical path FET ratio}
vTcl::DefineAlias "$site_3_0.label12" "Label15" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry12 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable tpfr
vTcl::DefineAlias "$site_3_0.entry12" "Entry14" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label13 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Useful skew perf
enhancement}
vTcl::DefineAlias "$site_3_0.label13" "Label16" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.cpd78 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable uspe
vTcl::DefineAlias "$site_3_0.cpd78" "Entry15" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.entry16 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable bsuf
vTcl::DefineAlias "$site_3_0.entry16" "Entry16" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.cpd80 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable temperature
vTcl::DefineAlias "$site_3_0.cpd80" "Entry17" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.label17 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Interconnect speedup
due to buffers}
vTcl::DefineAlias "$site_3_0.label17" "Label18" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.cpd86 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Temperature of
operation}
vTcl::DefineAlias "$site_3_0.cpd86" "Label19" vTcl::WidgetProc
"Toplevel8" 1
entry $site_3_0.cpd76 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state readonly -textvariable delay
vTcl::DefineAlias "$site_3_0.cpd76" "Entry18" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.cpd85 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-state disabled -text {Delay
(1/freq)}
vTcl::DefineAlias "$site_3_0.cpd85" "Label11" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.cpd79 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Critical module?}
vTcl::DefineAlias "$site_3_0.cpd79" "Label20" vTcl::WidgetProc
"Toplevel8" 1
checkbox $site_3_0.check1 \
-disabledforeground #a1a1a1 -state disabled -text {YES / NO} \
-variable cm_yes
vTcl::DefineAlias "$site_3_0.check1" "Checkbutton1" vTcl::WidgetProc
"Toplevel8" 1
bindtags $site_3_0.check1 "$site_3_0.check1 Checkbutton
$top all vTcl::Balloon"
bind $site_3_0.check1 <<SetBalloon>> {
set ::vTcl::balloon::%W {Check if critical module}
}
entry $site_3_0.entry18 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-state disabled -textvariable nadsp
vTcl::DefineAlias "$site_3_0.entry18" "Entry19" vTcl::WidgetProc
"Toplevel8" 1
label $site_3_0.cpd91 \
-disabledforeground #a1a1a1 \
-font [vTcl::font:getFontFromDescr
"-family helvetica -size 12"] \
-text {NADSP - Vdd droop (%
of Vdd)}
vTcl::DefineAlias "$site_3_0.cpd91" "Label21" vTcl::WidgetProc
"Toplevel8" 1
place $site_3_0.label1 \
-in $site_3_0 -x 3 -y 15 -width 213 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.label2 \
-in $site_3_0 -x 7 -y 43 -width 148 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.label3 \
-in $site_3_0 -x 5 -y 72 -width 293 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.label4 \
-in $site_3_0 -x 5 -y 100 -width 163 -height 20 -anchor nw \
-bordermode ignore
place $site_3_0.line1 \
-in $site_3_0 -x 299 -y 5 -width 2 -height 571 -anchor nw \

```

```

-bordermode ignore
  place $site_3.0.label5 \
-in $site_3.0 -x 5 -y 130 -width 164 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.label6 \
-in $site_3.0 -x 5 -y 160 -width 244 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.label7 \
-in $site_3.0 -x 2 -y 215 -width 189 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.label8 \
-in $site_3.0 -x 5 -y 245 -width 143 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.entry1 \
-in $site_3.0 -x 305 -y 15 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.entry2 \
-in $site_3.0 -x 305 -y 45 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.entry3 \
-in $site_3.0 -x 305 -y 72 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.entry4 \
-in $site_3.0 -x 305 -y 100 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.entry5 \
-in $site_3.0 -x 305 -y 130 -width 148 -height 22 -anchor nw \
-bordermode ignore
  place $site_3.0.entry6 \
-in $site_3.0 -x 305 -y 160 -width 148 -height 22 -anchor nw \
-bordermode ignore
  place $site_3.0.entry7 \
-in $site_3.0 -x 305 -y 215 -width 148 -height 22 -anchor nw \
-bordermode ignore
  place $site_3.0.entry8 \
-in $site_3.0 -x 305 -y 245 -width 148 -height 22 -anchor nw \
-in $site_3.0 -x 305 -y 274 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd77 \
-in $site_3.0 -x 305 -y 303 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.label10 \
-in $site_3.0 -x 7 -y 330 -width 186 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.entry10 \
-in $site_3.0 -x 305 -y 332 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.label11 \
-in $site_3.0 -x 6 -y 360 -width 255 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.entry11 \
-in $site_3.0 -x 305 -y 360 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd73 \
-in $site_3.0 -x 8 -y 274 -width 265 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd74 \
-in $site_3.0 -x 6 -y 303 -width 188 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.label12 \
-in $site_3.0 -x 4 -y 390 -width 167 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.entry12 \
-in $site_3.0 -x 305 -y 390 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.label13 \
-in $site_3.0 -x 4 -y 418 -width 231 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd78 \
-in $site_3.0 -x 305 -y 419 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.entry16 \
-in $site_3.0 -x 305 -y 449 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd80 \
-in $site_3.0 -x 305 -y 480 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.label17 \
-in $site_3.0 -x 5 -y 449 -width 267 -height 24 -anchor nw \

-bordermode ignore

  place $site_3.0.cpd86 \
-in $site_3.0 -x 5 -y 479 -width 188 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd76 \
-in $site_3.0 -x 304 -y 187 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd85 \
-in $site_3.0 -x 4 -y 188 -width 106 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd79 \
-in $site_3.0 -x 7 -y 511 -width 121 -height 24 -anchor nw \

-bordermode ignore
  place $site_3.0.check1 \
-in $site_3.0 -x 335 -y 510 -width 148 -height 22 -anchor nw \
-bordermode ignore
  place $site_3.0.entry18 \
-in $site_3.0 -x 305 -y 540 -width 148 -height 22 -anchor nw \

-bordermode ignore
  place $site_3.0.cpd91 \
-in $site_3.0 -x 5 -y 540 -width 148 -height 22 -anchor nw \
  button $stop.button1 \
\
-command [list vTcl:DoCmdOption $stop.button1 {incr x
wm withdraw .interactive_individual_1_edit
set edit_curr_mod
[tk_messageBox -title "Please confirm edit" -message "Edit
current (\# $x) module?" -type yesno -icon question]
wm deiconify
.interactive_individual_1_edit
if {[string equal $edit_curr_mod "yes"]} {
  set choose_edit "Proceed"
  .interactive_individual_1_edit.button1 config
-state disabled
.interactive_individual_1_edit.button3 config -state normal

.interactive_individual_1_edit.frame1.entry1 config -state normal
.interactive_individual_1_edit.frame1.entry2 config -state normal
.interactive_individual_1_edit.frame1.entry3 config -state normal
.interactive_individual_1_edit.frame1.entry4 config -state normal
.interactive_individual_1_edit.frame1.entry5 config -state normal
.interactive_individual_1_edit.frame1.entry6 config -state normal
.interactive_individual_1_edit.frame1.entry7 config -state normal
.interactive_individual_1_edit.frame1.entry8 config -state normal
.interactive_individual_1_edit.frame1.cpd75 config -state normal
.interactive_individual_1_edit.frame1.cpd77 config -state normal
.interactive_individual_1_edit.frame1.entry11 config -state normal
.interactive_individual_1_edit.frame1.entry12 config -state normal
.interactive_individual_1_edit.frame1.cpd78 config -state normal
.interactive_individual_1_edit.frame1.entry16 config -state normal
.interactive_individual_1_edit.frame1.cpd80 config -state normal
.interactive_individual_1_edit.frame1.check1 config -state normal
.interactive_individual_1_edit.frame1.entry18 config -state normal
}
} else {
  if {$x < [expr $read_modules_des + 1]} {
    set x1 [expr 0.01/$x]

    set y1 [expr 0.01/$read_modules_des]
    set y [expr {$y1/$x1} *
100]
    set module_entry_precentage $y
    set module_entry_step "$x
out of $read_modules_des entered"
    if {$x != $read_modules_des} {

      catch {exec more $cmd_file_to_edit | grep ".mods_des [expr $x+1]"
} read_mod_des
      set uu [split $read_mod_des " "]
      set c_orig [lindex
$uu 2]
      set twl [lindex $uu 3]
      set ulc [lindex $uu 4]
      set w_total_p
[lindex $uu 5]
      set v_total_n [lindex $uu 6]
      set f_old [lindex
$uu 7]
      set delay [expr 1/$f_old]
      set asf [lindex $uu 8]
      set cgf
[lindex $uu 9]
      set c_de_cap [lindex $uu 10]

```

```

set hvr [lindex $uu 11]

set lvr [expr 1-$hvr]
set c_wire_buff [lindex $uu 12]
set tpfr
[lindex $uu 13]
set uspe [lindex $uu 14]
set bsuf [lindex $uu 15]

set temperature [lindex $uu 16]
set cm_yes [lindex $uu 17]
set
nads [lindex $uu 18]
wm withdraw .interactive_individual_1_edit

wm deiconify .interactive_individual_1_edit
}
}
}
if {$x ==
$read_modules_des} {
if {[string equal $edit_curr_mod "no"]} {

wm withdraw .interactive_individual_1_edit
tk_messageBox -title
"EIDA information" -type ok -message "Module description edit complete
!\nEdit Vdd scaling characterization" -icon info
catch {exec more $cmd_file_to_edit | grep ".pro_des" } read_mod_des
set yy [split
$read_mod_des " "]
set alpha [lindex $yy 1]
set beta [lindex $yy
2]
set gamma [lindex $yy 3]
set delta [lindex $yy 4]
set epsilon
[lindex $yy 5]
set fsf_x [lindex $yy 6]
set fsf_y [lindex $yy 7]
wm
deiconify .interactive_process_edit
} else {
set curr_mod_is_final 1
}
}

set choose_edit "Edit / Skip"
-disabledforeground #a1a1a1 -text Proceed -textvariable
choose_edit
vTcl:DefineAlias "$top.butt1" "Button1" vTcl:WidgetProc "Toplevel8"
1
button $top.butt2 \
\
-command {set vdd_spec 0
set eta 0.95
set l_min 0
set s_gate_cap 0
set s_wire_cap 0
set s_width
0
set c_unit_old 0
set i_ds_old 0
set c_unit_new 0
set i_ds_new 0
set rpsf 0
set rcsf 0
set sf 0
set decap_sens 0
set i_leak_junc 0
set i_leak_gate 0
set i_gate_per_w 0
set i_othv 0
set i_owhv 0
set
i_otlv 0
set i_owlv 0} \
-disabledforeground #a1a1a1 -text {Clear All}
vTcl:DefineAlias "$top.butt2" "Button2" vTcl:WidgetProc "Toplevel8"
1
label $top.heading \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-text {Individual module
descriptors}
vTcl:DefineAlias "$top.heading" "Label10" vTcl:WidgetProc "Toplevel8"
1
ProgressBar $top.cpd73 \
-background #d6cbbb -foreground #000099 -height 15 -maximum
100 \
-relief raised -troughcolor #d9d9d9 -variable
module_entry_percentage
vTcl:DefineAlias "$top.cpd73" "ProgressBar1" vTcl:WidgetProc
"Toplevel8" 1
entry $top.cpd74 \
-background white -disabledforeground #a1a1a1 -insertbackground
black \
-relief groove -state readonly -textvariable
module_entry_step
vTcl:DefineAlias "$top.cpd74" "Entry3" vTcl:WidgetProc "Toplevel8"
1

button $top.butt3 \
\
-command [list vTcl:DoCmdOption $top.butt3 {set delay [expr
1/$f_old]
set lvr [expr 1-$hvr]
set newline "$c_orig $twl $ulc $w_total_p
$w_total_n $f_old $asf $cgf $c_de_cap $hvr $c_wire_buff
$tpfr $suspe $bsuf $temperature $cm_yes $nads"
catch {exec
\.\./scripts/swapline.pl $x $newline $cmd_file_to_edit >
temp_cmd_file } junk
catch {exec cp temp_cmd_file $cmd_file_to_edit
} junk3
.interactive_individual_1_edit.butt1 config -state normal
.interactive_individual_1_edit.butt3 config -state disabled
.interactive_individual_1_edit.frame1.entry1 config -state disabled
.interactive_individual_1_edit.frame1.entry2 config -state disabled
.interactive_individual_1_edit.frame1.entry3 config -state disabled
.interactive_individual_1_edit.frame1.entry4 config -state disabled
.interactive_individual_1_edit.frame1.entry5 config -state disabled
.interactive_individual_1_edit.frame1.entry6 config -state disabled
.interactive_individual_1_edit.frame1.entry7 config -state disabled
.interactive_individual_1_edit.frame1.entry8 config -state disabled
.interactive_individual_1_edit.frame1.cpd75 config -state disabled
.interactive_individual_1_edit.frame1.cpd77 config -state disabled
.interactive_individual_1_edit.frame1.entry11 config -state disabled
.interactive_individual_1_edit.frame1.entry12 config -state disabled
.interactive_individual_1_edit.frame1.cpd78 config -state disabled
.interactive_individual_1_edit.frame1.entry16 config -state disabled
.interactive_individual_1_edit.frame1.cpd80 config -state disabled
.interactive_individual_1_edit.frame1.check1 config -state disabled
.interactive_individual_1_edit.frame1.entry18 config -state disabled

if {$x<[expr $read_modules_des + 1]} {
set x1 [expr 0.01/$x]
set y [expr {$y1/$x1} * 100]

set module_entry_percentage $y
set module_entry_step "$x out of
$read_modules_des entered"
}
if {$curr_mod_is_final == 1} {
wm withdraw
.interactive_individual_1_edit
tk_messageBox -title "EIDA information"
-type ok -message "Module description edit complete !\nEdit Vdd scaling
characterization" -icon info
catch {exec more $cmd_file_to_edit | grep
".pro_des" } read_mod_des
set yy [split $read_mod_des " "]
set alpha
[lindex $yy 1]
set beta [lindex $yy 2]
set gamma [lindex $yy 3]
set
delta [lindex $yy 4]
set epsilon [lindex $yy 5]
set fsf_x [lindex $yy
6]
set fsf_y [lindex $yy 7]
wm deiconify .interactive_process_edit
}
else {
catch {exec more $cmd_file_to_edit | grep ".mods_des [expr $x+1]"
} read_mod_des
set uu [split $read_mod_des " "]
set c_orig [lindex
$uu 2]
set twl [lindex $uu 3]
set ulc [lindex $uu 4]
set w_total_p
[lindex $uu 5]
set w_total_n [lindex $uu 6]
set f_old [lindex $uu 7]

set asf [lindex $uu 8]
set cgf [lindex $uu 9]
set c_de_cap [lindex
$uu 10]
set hvr [lindex $uu 11]
set c_wire_buff [lindex $uu 12]
set
tpfr [lindex $uu 13]
set uspe [lindex $uu 14]
set bsuf [lindex $uu 15]

```

```

set temperature [lindex $uu 16]
set cm_yes [lindex $uu 17]
set naddp
[lindex $uu 18]
}}} \
-disabledforeground #a1a1a1 -state disabled -text {Save edit}
vTcl::DefineAlias "$stop.button3" "Button3" vTcl::WidgetProc "Toplevel8"
1
place $stop.frame1 \
-in $stop -x 10 -y 50 -width 470 -height 590 -anchor nw \
-bordermode ignore
place $stop.button1 \
-in $stop -x 385 -y 655 -width 67 -height 28 -anchor nw \
-bordermode ignore
place $stop.button2 \
-in $stop -x 70 -y 655 -width 78 -height 28 -anchor nw \
-bordermode ignore
place $stop.heading \
-in $stop -x 15 -y 10 -width 205 -height 24 -anchor nw \
-bordermode ignore
place $stop.cpd73 \
-in $stop -x 265 -y 5 -width 220 -height 15 -anchor nw \
-bordermode inside
place $stop.cpd74 \
-in $stop -x 265 -y 25 -width 218 -height 22 -anchor nw \
-bordermode inside
place $stop.button3 \
-in $stop -x 220 -y 655 -anchor nw -bordermode ignore
vTcl::FireEvent $base <<Ready>>
}
proc vTcl::Window::interactive_process {base} {
if {$base == ""} {
set base .interactive_process
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl::toplevel $top -class Toplevel \
-relief groove -highlightcolor black
wm withdraw $top
wm focusmodel $top passive
wm geometry $top
651x408+118+246; update
wm maxsize $top 1265 994
wm minsize $top 1
1
wm overrideredirect $top 0
wm resizable $top 0 0
wm title $top
"Vdd scaling characterization"
vTcl::DefineAlias "$top" "Toplevel5"
vTcl::Toplevel::WidgetProc "" 1
bindtags $top "$top Toplevel all
_Toplevel"
vTcl::FireEvent $top <<Create>>
wm protocol $top
WM_DELETE_WINDOW "vTcl::FireEvent $top <<DeleteWindow>>"
canvas
$stop.canvas1 \
-borderwidth 2 -closeenough 1.0 -height 362 -insertbackground
black \
-relief ridge -selectbackground #c1c2c1 -selectforeground
black \
-width 640
vTcl::DefineAlias "$stop.canvas1" "Canvas1" vTcl::WidgetProc "Toplevel5"
1
label $stop.canvas1.label1 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Enter constants for Vdd scaling equations}
vTcl::DefineAlias "$stop.canvas1.label1" "Label1" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.label2 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text {alpha}
vTcl::DefineAlias "$stop.canvas1.label2" "Label2" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.cpd77 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text beta
vTcl::DefineAlias "$stop.canvas1.cpd77" "Label3" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.cpd78 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text gamma
vTcl::DefineAlias "$stop.canvas1.cpd78" "Label4" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.cpd79 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text delta
vTcl::DefineAlias "$stop.canvas1.cpd79" "Label5" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.cpd80 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text epsilon
vTcl::DefineAlias "$stop.canvas1.cpd80" "Label6" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.cpd81 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text X
vTcl::DefineAlias "$stop.canvas1.cpd81" "Label7" vTcl::WidgetProc
"Toplevel5" 1
label $stop.canvas1.cpd82 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text Y
vTcl::DefineAlias "$stop.canvas1.cpd82" "Label8" vTcl::WidgetProc
"Toplevel5" 1
Separator $stop.canvas1.line1 \
-background #d6d6db
vTcl::DefineAlias "$stop.canvas1.line1" "Separator1" vTcl::WidgetProc
"Toplevel5" 1
bind $stop.canvas1.line1 <Destroy> {
Widget::destroy %W; rename %W {}
}
entry $stop.canvas1.entry1 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable alpha
vTcl::DefineAlias "$stop.canvas1.entry1" "Entry1" vTcl::WidgetProc
"Toplevel5" 1
entry $stop.canvas1.cpd85 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable beta
vTcl::DefineAlias "$stop.canvas1.cpd85" "Entry2" vTcl::WidgetProc
"Toplevel5" 1
entry $stop.canvas1.cpd86 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable gamma
vTcl::DefineAlias "$stop.canvas1.cpd86" "Entry3" vTcl::WidgetProc
"Toplevel5" 1
entry $stop.canvas1.cpd87 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable delta
vTcl::DefineAlias "$stop.canvas1.cpd87" "Entry4" vTcl::WidgetProc
"Toplevel5" 1
entry $stop.canvas1.cpd88 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black \
-textvariable epsilon
vTcl::DefineAlias "$stop.canvas1.cpd88" "Entry5" vTcl::WidgetProc
"Toplevel5" 1
entry $stop.canvas1.cpd89 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable fsf_x
vTcl::DefineAlias "$stop.canvas1.cpd89" "Entry6" vTcl::WidgetProc
"Toplevel5" 1
entry $stop.canvas1.cpd90 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable fsf_y
vTcl::DefineAlias "$stop.canvas1.cpd90" "Entry7" vTcl::WidgetProc
"Toplevel5" 1
button $stop.canvas1.button2 \

```

```

-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-image [vTcl:image:get_image [file join / home charles
EIDA_gui images eqn1.s.jpg]] \
-text button
vTcl:DefineAlias "$top.canvas1.butt2" "Button2" vTcl:WidgetProc
"Toplevel5" 1
button $top.canvas1.cpd73 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-image [vTcl:image:get_image [file join / home charles
EIDA_gui images eqn2.s.jpg]] \
-text button
vTcl:DefineAlias "$top.canvas1.cpd73" "Button3" vTcl:WidgetProc
"Toplevel5" 1
button $top.butt1 \
\
-command {set new_cmd_file_id [open $new_cmd_filename a]
puts $new_cmd_file_id "\# Vdd scaling characterization parameters\n"
puts
$new_cmd_file_id "\# alpha,beta,gamma,delta,epsilon,fsf_x,fsf_y\n"

puts $new_cmd_file_id ".pro_des $alpha $beta $gamma
$delta $epsilon $fsf_x $fsf_y\n\n"
puts $new_cmd_file_id
"\# Begin module design recipe\n"
puts $new_cmd_file_id "\#
vary_vdd,vdd_applied,use_dual_vt,use_abb,use_abb_fb,use_abb_rb,use_st\n"

close $new_cmd_file_id
wm withdraw .interactive_process
tk_messageBox
-title "EIDA information" -type ok -message "Command file creation
complete" -icon info
set vdd_bump_for_cmd_file_run 0
set vdd_applied
$vdv_bump
.design_choice.canvas1.entry1 config -state disabled
set
vary_vdd 0
set use_dual_vt 0
set use_abb 0
set use_st 0
wm deiconify
.design_choice} \
-disabledforeground #a1a1a1 -text {Save and proceed}
vTcl:DefineAlias "$top.butt1" "Button1" vTcl:WidgetProc "Toplevel5" 1
place $top.canvas1 \
-in $top -x 5 -y 5 -width 640 -height 362 -anchor nw \

-bordermode ignore
place $top.canvas1.label1 \
-in $top.canvas1 -x 5 -y 10 -anchor nw -bordermode ignore
place $top.canvas1.label2 \
-in $top.canvas1 -x 15 -y 45 -width 50 -height 24 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd77 \
-in $top.canvas1 -x 15 -y 80 -anchor nw -bordermode inside
place $top.canvas1.cpd78 \
-in $top.canvas1 -x 13 -y 120 -width 61 -height 24 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd79 \
-in $top.canvas1 -x 15 -y 155 -anchor nw -bordermode inside
place $top.canvas1.cpd80 \
-in $top.canvas1 -x 15 -y 190 -anchor nw -bordermode inside
place $top.canvas1.cpd81 \
-in $top.canvas1 -x 15 -y 270 -anchor nw -bordermode inside
place $top.canvas1.cpd82 \
-in $top.canvas1 -x 15 -y 320 -anchor nw -bordermode inside
place $top.canvas1.line1 \
-in $top.canvas1 -x 10 -y 244 -width 611 -height 2 -anchor nw \

-bordermode ignore
place $top.canvas1.entry1 \
-in $top.canvas1 -x 116 -y 45 -width 150 -height 25 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd85 \
-in $top.canvas1 -x 116 -y 80 -width 150 -height 25 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd86 \
-in $top.canvas1 -x 116 -y 118 -width 150 -height 25 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd87 \
-in $top.canvas1 -x 116 -y 155 -width 150 -height 25 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd88 \
-in $top.canvas1 -x 115 -y 190 -width 150 -height 25 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd89 \
-in $top.canvas1 -x 60 -y 265 -width 150 -height 25 -anchor nw \

-bordermode inside

place $top.canvas1.cpd90 \
-in $top.canvas1 -x 60 -y 315 -width 150 -height 25 -anchor nw \

-bordermode inside
place $top.canvas1.butt2 \
-in $top.canvas1 -x 275 -y 41 -width 348 -height 178 -anchor nw \

-bordermode ignore
place $top.canvas1.cpd73 \
-in $top.canvas1 -x 275 -y 270 -width 333 -height 68 -anchor nw \

-bordermode inside
place $top.butt1 \
-in $top -x 250 -y 375 -anchor nw -bordermode ignore
vTcl:FireEvent $base <<Ready>>
}
proc vTclWindow.interactive_process_edit {base} {
if {$base == ""} {
set base .interactive_process_edit
}
if {[winfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl:toplevel $top -class Toplevel \
-relief groove -highlightcolor black
wm withdraw $top
wm focusmodel $top passive
wm geometry $top
651x408+44+128; update
wm maxsize $top 1265 994
wm minsize $top
1 1
wm overriddenirect $top 0
wm resizable $top 0 0
wm title
$top "Edit Vdd scaling characterization"
vTcl:DefineAlias "$top"
"Toplevel9" vTcl:Toplevel:WidgetProc "" 1
bindtags $top $top
Toplevel all _Toplevel"
vTcl:FireEvent $top <<Create>>
wm protocol
$top WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"
canvas
$top.canvas1 \
-borderwidth 2 -closeenough 1.0 -height 362 -insertbackground
black \
-relief ridge -selectbackground #c1c2c1 -selectforeground
black \
-width 640
vTcl:DefineAlias "$top.canvas1" "Canvas1" vTcl:WidgetProc "Toplevel9" 1
label $top.canvas1.label1 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {Enter constants for Vdd scaling equations}
vTcl:DefineAlias "$top.canvas1.label1" "Label1" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.label2 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text {alpha}
vTcl:DefineAlias "$top.canvas1.label2" "Label2" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.cpd77 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text beta
vTcl:DefineAlias "$top.canvas1.cpd77" "Label3" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.cpd78 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text gamma
vTcl:DefineAlias "$top.canvas1.cpd78" "Label4" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.cpd79 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text delta
vTcl:DefineAlias "$top.canvas1.cpd79" "Label5" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.cpd80 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \

```

```

-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text epsilon
vTcl:DefineAlias "$top.canvas1.cpd80" "Label6" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.cpd81 \
-activatebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text X
vTcl:DefineAlias "$top.canvas1.cpd81" "Label7" vTcl:WidgetProc
"Toplevel9" 1
label $top.canvas1.cpd82 \
-activatebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl:font:getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black -text Y
vTcl:DefineAlias "$top.canvas1.cpd82" "Label8" vTcl:WidgetProc
"Toplevel9" 1
Separator $top.canvas1.line1 \
-background #d6d6db
vTcl:DefineAlias "$top.canvas1.line1" "Separator1" vTcl:WidgetProc
"Toplevel9" 1
bind $top.canvas1.line1 <Destroy> {
Widget::destroy %W; rename %W {}
}
entry $top.canvas1.entry1 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable alpha
vTcl:DefineAlias "$top.canvas1.entry1" "Entry1" vTcl:WidgetProc
"Toplevel9" 1
entry $top.canvas1.cpd85 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable beta
vTcl:DefineAlias "$top.canvas1.cpd85" "Entry2" vTcl:WidgetProc
"Toplevel9" 1
entry $top.canvas1.cpd86 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable gamma
vTcl:DefineAlias "$top.canvas1.cpd86" "Entry3" vTcl:WidgetProc
"Toplevel9" 1
entry $top.canvas1.cpd87 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable delta
vTcl:DefineAlias "$top.canvas1.cpd87" "Entry4" vTcl:WidgetProc
"Toplevel9" 1
entry $top.canvas1.cpd88 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black \
-textvariable epsilon
vTcl:DefineAlias "$top.canvas1.cpd88" "Entry5" vTcl:WidgetProc
"Toplevel9" 1
entry $top.canvas1.cpd89 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable fsf_x
vTcl:DefineAlias "$top.canvas1.cpd89" "Entry6" vTcl:WidgetProc
"Toplevel9" 1
entry $top.canvas1.cpd90 \
-background white -disabledforeground #a1a1a1 -foreground black \
-highlightcolor black -insertbackground black \
-selectbackground
#c1c2c1 -selectforeground black -textvariable fsf_y
vTcl:DefineAlias "$top.canvas1.cpd90" "Entry7" vTcl:WidgetProc
"Toplevel9" 1
button $top.canvas1.butt2 \
-activatebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-image [vTcl:image:get_image [file join / home charles
EIDA_gui images eqn1.s.jpg]] \
-text button
vTcl:DefineAlias "$top.canvas1.butt2" "Button2" vTcl:WidgetProc
"Toplevel9" 1
button $top.canvas1.cpd73 \
-activatebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-image [vTcl:image:get_image [file join / home charles
EIDA_gui images eqn2.s.jpg]] \
-text button
vTcl:DefineAlias "$top.canvas1.cpd73" "Button3" vTcl:WidgetProc
"Toplevel9" 1
button $top.butt1 \
-command {set newline "$alpha $beta $gamma $delta $epsilon
$fsf_x $fsf_y"
catch {exec \.\./scripts/swapline.pl pro $newline $cmd_file_to_edit >
temp_cmd_file } junk
catch {exec cp temp_cmd_file $cmd_file_to_edit }
junk2
wm withdraw .interactive_process_edit
tk_messageBox -title "EIDA
information" -type ok -message "Command file edit complete" -icon info
}
wm deiconify .control} \
-disabledforeground #a1a1a1 -text {Save and proceed}
vTcl:DefineAlias "$top.butt1" "Button1" vTcl:WidgetProc "Toplevel9"
1
place $top.canvas1 \
-in $top -x 5 -y 5 -width 640 -height 362 -anchor nw \
-bordermode ignore
place $top.canvas1.label1 \
-in $top.canvas1 -x 15 -y 10 -anchor nw -bordermode ignore
place $top.canvas1.label2 \
-in $top.canvas1 -x 15 -y 45 -width 50 -height 24 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd77 \
-in $top.canvas1 -x 15 -y 80 -anchor nw -bordermode inside
place $top.canvas1.cpd78 \
-in $top.canvas1 -x 13 -y 120 -width 61 -height 24 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd79 \
-in $top.canvas1 -x 15 -y 155 -anchor nw -bordermode inside
place $top.canvas1.cpd80 \
-in $top.canvas1 -x 15 -y 190 -anchor nw -bordermode inside
place $top.canvas1.cpd81 \
-in $top.canvas1 -x 15 -y 270 -anchor nw -bordermode inside
place $top.canvas1.cpd82 \
-in $top.canvas1 -x 15 -y 320 -anchor nw -bordermode inside
place $top.canvas1.line1 \
-in $top.canvas1 -x 10 -y 244 -width 611 -height 2 -anchor nw \
-bordermode ignore
place $top.canvas1.entry1 \
-in $top.canvas1 -x 116 -y 45 -width 150 -height 25 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd85 \
-in $top.canvas1 -x 116 -y 80 -width 150 -height 25 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd86 \
-in $top.canvas1 -x 116 -y 118 -width 150 -height 25 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd87 \
-in $top.canvas1 -x 116 -y 155 -width 150 -height 25 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd88 \
-in $top.canvas1 -x 115 -y 190 -width 150 -height 25 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd89 \
-in $top.canvas1 -x 60 -y 265 -width 150 -height 25 -anchor nw \
-bordermode inside
place $top.canvas1.cpd90 \
-in $top.canvas1 -x 60 -y 315 -width 150 -height 25 -anchor nw \
-bordermode inside
place $top.canvas1.butt2 \
-in $top.canvas1 -x 275 -y 41 -width 348 -height 178 -anchor nw \
-bordermode ignore
place $top.canvas1.cpd73 \
-in $top.canvas1 -x 275 -y 270 -width 333 -height 68 -anchor nw \
-bordermode inside
place $top.butt1 \
-in $top -x 250 -y 375 -anchor nw -bordermode ignore
vTcl:FireEvent $base <<Ready>>
}
proc vTclWindow.results_display {base} {
if {$base == ""} {
set base .results_display
}
if {[wininfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl:toplevel $top -class Toplevel \
-highlightcolor black
wm withdraw $top
wm focusmodel $top passive
wm geometry $top
646x419+0+0; update
wm maxsize $top 1265 994
wm minsize $top
1 1
wm overrideredirect $top 0
wm resizable $top 0 0
wm
title $top "EIDA Results"

```

```

vTcl::DefineAlias "$top" "Toplevel10"
vTcl::Toplevel::WidgetProc "" 1
bindtags $top "$top Toplevel all
Toplevel"
vTcl::FireEvent $top <<Create>>
wm protocol $top
WM_DELETE_WINDOW "vTcl::FireEvent $top <<DeleteWindow>>"
canvas
$top.canvas1 \
-borderwidth 2 -closeenough 1.0 -height 312 -insertbackground
black \
-relief ridge -selectbackground #c1c2c1 -selectforeground
black \
-width 625
vTcl::DefineAlias "$top.canvas1" "Canvas1" vTcl::WidgetProc "Toplevel10"
1
label $top.label1 \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-relief groove -text {Results
summary from EIDA run}
vTcl::DefineAlias "$top.label1" "Label1" vTcl::WidgetProc "Toplevel10"
1
button $top.button1 \
-command {wm withdraw .results_display
wm deiconify .control} \
-disabledforeground #a1a1a1 -text {Return to main window}
vTcl::DefineAlias "$top.button1" "Button1" vTcl::WidgetProc "Toplevel10"
1
button $top.button2 \
-command {catch {exec rm -f tmp_cmd_filename} junk3
catch {exec rm -f temp_cmd_file} junk3
catch {exec rm -f
abb_spice_run_junk} junk3
exit} \
-disabledforeground #a1a1a1 -text {Exit EIDA}
vTcl::DefineAlias "$top.button2" "Button2" vTcl::WidgetProc "Toplevel10"
1
place $top.canvas1 \
-in $top -x 10 -y 55 -width 625 -height 312 -anchor nw \

-bordermode ignore
place $top.label1 \
-in $top -x 210 -y 15 -width 251 -height 34 -anchor nw \

-bordermode ignore
place $top.button1 \
-in $top -x 90 -y 380 -anchor nw -bordermode ignore
place $top.button2 \
-in $top -x 460 -y 380 -anchor nw -bordermode ignore
vTcl::FireEvent $base <<Ready>>
}
proc vTcl::Window.wait {base} {
if {$base == ""} {
set base .wait
}
if {[wininfo exists $base]} {
wm deiconify $base; return
}
set top $base
vTcl::Toplevel $top -class Toplevel \
-highlightcolor black
wm withdraw $top
wm focusmodel $top passive
wm geometry $top
416x187+296+466; update
wm maxsize $top 1265 994
wm minsize $top
1
wm overrideredirect $top 0
wm resizable $top 0 0
wm title
$top "EIDA busy please wait"
vTcl::DefineAlias "$top" "Toplevel11"
vTcl::Toplevel::WidgetProc "" 1
bindtags $top "$top Toplevel all
Toplevel"
vTcl::FireEvent $top <<Create>>
wm protocol $top
WM_DELETE_WINDOW "vTcl::FireEvent $top <<DeleteWindow>>"
canvas
$top.canvas1 \
-borderwidth 5 -closeenough 1.0 -height 172 -highlightthickness
2 \
-insertbackground black -relief ridge -selectbackground
#c1c2c1 \
-selectforeground black -width 395
vTcl::DefineAlias "$top.canvas1" "Canvas1" vTcl::WidgetProc "Toplevel11"
1
button $top.canvas1.button1 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 -foreground black -highlightcolor
black \
-image [vTcl::image::get_image [file join / home charles
EIDA_gui_images hour_glass.jpg]] \
-text button
vTcl::DefineAlias "$top.canvas1.button1" "Button1" vTcl::WidgetProc
"Toplevel11" 1
label $top.canvas1.label1 \
-activebackground #f6f7f6 -activeforeground black \
-disabledforeground #a1a1a1 \
-font [vTcl::font::getFontFromDescr
"-family helvetica -size 12"] \
-foreground black -highlightcolor
black \
-text {SPICE running in background
Please wait !!}
vTcl::DefineAlias "$top.canvas1.label1" "Label1" vTcl::WidgetProc
"Toplevel11" 1
place $top.canvas1 \
-in $top -x 10 -y 10 -width 395 -height 172 -anchor nw \

-bordermode ignore
place $top.canvas1.button1 \
-in $top.canvas1 -x 20 -y 35 -width 128 -height 93 -anchor nw \

-bordermode ignore
place $top.canvas1.label1 \
-in $top.canvas1 -x 155 -y 65 -anchor nw -bordermode ignore
vTcl::FireEvent $base <<Ready>>
}
bind "_Toplevel" <<Create>> {
if {[info exists _topcount]} {set _topcount 0}; incr _topcount
}
bind "_Toplevel" <<DeleteWindow>> {
if {[set ::WM::modal]} {
vTcl::Toplevel::WidgetProc $WM endmodal
} else {
destroy $WM; if {$_topcount == 0} {exit}
}
}
bind "_Toplevel" <Destroy> {
if {[wininfo toplevel $WM] == "$WM"} {incr _topcount -1}
}
bind "_Toplevel" <Enter> {
if {[info exists vTcl::sourcing]} {
find "_vTclBalloon" <<KillBalloon>> {
namespace eval ::vTcl::balloon {
after cancel $id
if {[wininfo exists .vTcl.balloon]} {
destroy .vTcl.balloon
}
set set 0
}
}
bind "_vTclBalloon" <<vTclBalloon>> {
if {[set ::vTcl::balloon::first] != 1} {break}
namespace eval
::vTcl::balloon {
set first 2
if {[wininfo exists .vTcl]} {
toplevel .vTcl; wm withdraw .vTcl
}
if {[wininfo exists .vTcl.balloon]} {
toplevel .vTcl.balloon -bg black
}
wm overrideredirect .vTcl.balloon 1
label .vTcl.balloon.l
-text {[%W]} -relief flat -bg #ffffff -fg black -padx 2 -pady 0
-anchor w
pack .vTcl.balloon.l -side left -padx 1 -pady 1
wm
geometry .vTcl.balloon +[expr {[wininfo rootx $W]+[wininfo width
$W]/2}]+[expr {[wininfo rooty $W]+[wininfo height $W]+4}]
set set 1
}
}
bind "_vTclBalloon" <Button> {
namespace eval ::vTcl::balloon {
set first 0
}
vTcl::FireEvent $W <<KillBalloon>>
}
bind "_vTclBalloon" <Enter> {
namespace eval ::vTcl::balloon {
if {[info exists $W]} {
vTcl::FireEvent $W <<SetBalloon>>
}
set set 0
set first 1
set id [after 500 {vTcl::FireEvent $W
<<vTclBalloon>>}]
}
}
bind "_vTclBalloon" <Leave> {
namespace eval ::vTcl::balloon {
set first 0
}
vTcl::FireEvent $W <<KillBalloon>>
}
bind "_vTclBalloon" <Motion> {
namespace eval ::vTcl::balloon {
if {[set]} {
after cancel $id
set id [after 500 {vTcl::FireEvent $W
<<vTclBalloon>>}]
}
}
}
}

```

```
Window show .
Window show .command_file
Window show .control
Window show .design_choice
Window show .interactive_common
Window show
.interactive_common_edit
Window show .interactive_individual_1
Window
```

```
show .interactive_individual_1_edit
Window show .interactive_process

Window show .interactive_process_edit
Window show .results_display
Window show .wait
main $argc $argv
```


Bibliography

- [1] G. Bailey and W. Huang, *More than "Moore" to win: Optimization strategies for success in a maturing semiconductor industry*, ser. IBM Global Business Services. IBM Institute for Business Value, 2008.
- [2] B. Dennington, "Low power design from technology challenge to great products," in *Proc. of Intl. Sym. on Low power electronics and design*. New York, NY, USA: ACM, 2006, pp. 213–213.
- [3] J. Lin, "Design technology challenges for system and chip level designs in very deep submicron technologies," in *Proc. of 1st IEEE/ACM/IFIP Intl. Conf. on Hardware/software codesign and system synthesis*. New York, NY, USA: ACM, 2003, pp. 194–194.
- [4] <http://www.itrs.net/>, "The International Technology Roadmap for Semiconductors," [Online].
- [5] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, Apr 1965.
- [6] John Buscemi and Vince Smith, "IBM POWER6 Microprocessor and IBM System p570," *IBM Corporation press release*, May 2007.
- [7] B. Wong, A. Mittal, Y. Cao, and G. W. Starr, "Nano-CMOS Circuit and Physical Design," *ISBN: 978-0-471-46610-9*, Nov 2004.
- [8] L. Chang, Y. Choi, D. HA, P. Rande, S. Xiong, J. Bokor, C. Hu, and T. King, "Extremely Scaled Silicon Nano-CMOS Devices," *Proceedings of the IEEE*, vol. 91, pp. 1860–1873, Nov 2003.
- [9] S. Bobba and I. Hajj, "Maximum voltage variation in the power distribution network of vlsi circuits with rlc models," in *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2001, pp. 376–381.
- [10] S. Kubicek and K. De Meyer, "Cmos scaling to 25 nm gate lengths," *The Fourth International Conference on Advanced Semiconductor Devices and Microsystems*, pp. 259–270, Oct. 2002.

- [11] H. Iwai, "Cmos scaling toward sub-10 nm regime," *11th IEEE International Symposium on Electron Devices for Microwave and Optoelectronic Applications*, pp. 30–34, Nov. 2003.
- [12] H. Iwai, "Cmos scaling for sub-90 nm to sub-10 nm," *Proceedings of the 17th International Conference on VLSI Design*, pp. 30–35, 2004.
- [13] U. Narasimha, B. Abraham, and N. NS, "Statistical analysis of capacitance coupling effects on delay and noise," *7th International Symposium on Quality Electronic Design*, pp. 6 pp.–, March 2006.
- [14] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, "Considering process variations during system-level power analysis," *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, pp. 342–345, Oct. 2006.
- [15] G. Ribes, M. Rafik, and D. Roy, "Reliability issues for nano-scale cmos dielectrics," *Microelectronic Engineering*, vol. 84, no. 9-10, pp. 1910 – 1916, 2007, iNFOS 2007. [Online]. Available: <http://www.sciencedirect.com/science/article/B6V0W-4NVM053-P/2/5bc1e475c7c4072dfd6dd6955591e1fa>
- [16] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," *International Conference on Dependable Systems and Networks*, pp. 389–398, 2002.
- [17] D. Mavis and P. Eaton, "Soft error rate mitigation techniques for modern microcircuits," *Reliability Physics Symposium Proceedings, 2002. 40th Annual*, pp. 216–225, 2002.
- [18] R. Kapre, K. Shakeri, H. Puchner, J. Tandigan, T. Nigam, K. Jang, M. Reddy, S. Lakshminarayanan, D. Sajoto, and M. Whately, "Sram variability and supply voltage scaling challenges," *Proceedings. 45th annual. IEEE International Reliability Physics Symposium*, pp. 23–28, April 2007.
- [19] S. Basu and R. Vemuri, "Process variation and nbtI tolerant standard cells to improve parametric yield and lifetime of ics," *IEEE Computer Society Annual Symposium on VLSI*, pp. 291–298, March 2007.
- [20] Q. Ding, R. Luo, and Y. Xie, "Impact of process variation on soft error vulnerability for nanometer vlsi circuits," *6th International Conference On ASIC*, vol. 2, pp. 1117–1121, Oct. 2005.
- [21] M. Chang, "Foundry future: Challenges in the 21st century," *Digest of Technical Papers. IEEE International Solid-State Circuits Conference*, pp. 18–23, Feb. 2007.

- [22] C. Young, R. Leu, C. Hung, S. Chen, and K. Huang, "Critical dimension management of photo resister by lens heating compensation and statistical process control," *5th International Workshop on Statistical Metrology*, pp. 23–26, 2000.
- [23] K. Aoyama, H. Kunitomo, K. Tsuneno, H. Sato, K. Mori, and H. Masuda, "Rigorous statistical process variation analysis for quarter- μm cmos with advanced tcad metrology," *2nd International Workshop on Statistical Metrology*, pp. 8–11, Jun 1997.
- [24] A. K. Wong, *Resolution Enhancement Techniques in Optical Lithography*. SPIE Press, March 2001.
- [25] S. Raghvendra and P. Hurat, "Dfm: linking design and manufacturing," *18th International Conference on VLSI Design*, pp. 705–708, Jan. 2005.
- [26] P. Gupta, A. B. Kahngt, and C.-H. Park, "Detailed Placement for Improved Depth of Focus and CD Control," *Proc. of the Asia and South Pacific Design Automation Conf.*, pp. 343–348, 2005.
- [27] S. Sivakumar, "Lithography challenges for 32nm technologies and beyond," *International Electron Devices Meeting*, pp. 1–4, Dec. 2006.
- [28] T. Furuyama, "Deep sub-100 nm design challenges," *9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, pp. 9–16, 0-0 2006.
- [29] W. Zhao, Y. Cao, F. Liu, K. Agarwal, D. Acharyya, S. Nassif, and K. Nowka, "Extraction and Modeling of Process Variations for Robust Nanoscale Design," *Arizona State University and IBM Austin Research Laboratory*, 2007.
- [30] G. A. Northrop and P.-F. Lu, "A semi-custom design flow in high-performance microprocessor design," in *Proc. of DAC*. ACM, 2001, pp. 426–431.
- [31] C. Constantinescu, "Trends and challenges in vlsi circuit reliability," *Micro, IEEE*, vol. 23, no. 4, pp. 14–19, July-Aug. 2003.
- [32] S. Rusu, M. Sachdev, C. Svensson, and B. Nauta, "T3: Trends and challenges in vlsi technology scaling towards 100nm," in *Proc. of ASP-DAC*. IEEE Computer Society, 2002, p. 16.
- [33] T. Karn *et al.*, "Eda challenges facing future microprocessor design," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 19, no. 12, pp. 1498–1506, Dec 2000.
- [34] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: an infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, Feb 2002.

- [35] J. Burns and J.-L. Gaudiot, "Area and system clock effects on SMT/CMP throughput," *IEEE Transactions on Computers*, pp. 141–152, Feb 2005.
- [36] W. Ye and et al., "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," *37th DAC*, pp. 340–345.
- [37] N. Vijaykrishnan and et al., "Energy-driven integrated hardware-software optimizations using SimplePower," *Proceedings of the 27th International Symposium on Computer Architecture*, pp. 340–345, 2000.
- [38] D. Brooks *et al.*, "Wattch: a framework for architectural-level power analysis and optimizations," *Proc. of the 27th Int. Sym. on Comp. Arch.*, pp. 83–94, 2000.
- [39] B. Bishop, T. Kelliher, and M. Irwin, "The design of a register renaming unit," *Proceedings. Ninth Great Lakes Symposium on VLSI*, pp. 34–37, Mar 1999.
- [40] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective superscalar processors," *SIGARCH Comput. Archit. News*, vol. 25, no. 2, pp. 206–218, 1997.
- [41] S. Palacharla, N. Jouppi, and J. Smith., "Quantifying the Complexity of Superscalar Processors." *Univ. of Wisconsin Computer Science Tech. Report 1328*, 1997.
- [42] H. Fair and D. Bailey., "Clocking Design and Analysis for a 600MHz Alpha Microprocessor." *ISSCC Digest of Technical Papers*, pp. 398–399, 1998.
- [43] D. Ponomarev, G. Kucuk, and K. Ghose, "AccuPower: An Accurate Power Estimation Tool for Superscalar Microprocessors," *Proceedings of Design Automation and Test in Europe*, pp. 124–129, March 2002.
- [44] D. Broooks *et al.*, "New methodology for early-stage, microarchitecture-level powerperformance analysis of microprocessors," *IBM J. RES. & DEV.*, no. 5/6, pp. 653–670, Sept/Nov 2003.
- [45] L. Zhong, S. Ravi, A. Raghunathan, and N. Jha, "Rtl-aware cycle-accurate functional power estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2103–2117, Oct. 2006.
- [46] M. Pedram and A. Abdollahi, "Low-power rt-level synthesis techniques: a tutorial," *IEE Proceedings of Computers & Digital Techniques*, pp. 333–343, May 2005.
- [47] A. Khan, P. Watson, G. Kuo, D. Le, T. Nguyen, S. Yang, P. Bennett, P. Huang, J. Gill, C. Hawkins, J. Goodenough, D. Wang, I. Ahmed, P. Tran, H. Mak, O. Kim, F. Martin, Y. Fan, D. Ge, J. Kung, and V. Shek, "A 90-nm power

- optimization methodology with application to the arm 1136jf-s microprocessor,” *IEEE Journal of Solid-State Circuits*, vol. 41, no. 8, pp. 1707–1717, Aug. 2006.
- [48] <http://www.sequencedesign.com>, [Online].
 - [49] <http://www.calypto.com>, [Online].
 - [50] <http://www.synopsys.com>, [Online].
 - [51] <http://www.magma-da.com>, [Online].
 - [52] <http://www.cadence.com>, [Online].
 - [53] T. Chen and S. Naffziger, “Comparison of adaptive body bias (abb) and adaptive supply voltage (asv) for improving delay and leakage under the presence of process variation,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 11, no. 5, pp. 888–899, 2003.
 - [54] R. Vilangudipitchai and P. Balsara, “Power switch network design for mtcmos,” *18th Intl. Conf. on VLSI Design*, pp. 836–839, Jan 2005.
 - [55] H. Ramakrishnan, K. Maharatna, S. Chattopadhyay, and A. Yakovlev, “Impact of strain on the design of low-power high-speed circuits,” *IEEE International Symposium on Circuits and Systems*, pp. 1153–1156, May 2007.
 - [56] D. Sylvester *et al.*, “System-level performance modeling with BACPAC Berkeley advanced chip performance calculator,” *ACM Int. Workshop on System-Level Interconnect Prediction*, pp. 109–114, 1999.
 - [57] <http://www.eecs.umich.edu/dennis/bacpac/index.html>, [Online].
 - [58] D. Duarte, V. Narayanan, and M. Irwin, “Impact of technology scaling in the clock system power,” *Proc. of IEEE Computer Society Annual Symposium on VLSI*, pp. 52–57, 2002.
 - [59] P. Restle and A. Deutsch, “Designing the best clock distribution network,” *IEEE Symposium on VLSI Circuits Digest of Technical Papers*, pp. 2–5, Jun 1998.
 - [60] P. Mahoney, E. Fetzer, B. Doyle, and S. Naffziger, “Clock distribution on a dual-core, multi-threaded itanium-family processor,” *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, vol. 1, pp. 292–599, Feb. 2005.
 - [61] E. S. Fetzer, “Using adaptive circuits to mitigate process variations in a microprocessor design,” *IEEE Des. Test*, vol. 23, no. 6, pp. 476–483, 2006.

- [62] S. Mukhopadhyay, A. Raychowdhury, and K. Roy, "Accurate estimation of total leakage in nanometer-scale bulk cmos circuits based on device geometry and doping profile," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, no. 3, pp. 363–381, March 2005.
- [63] G. Ascia, V. Catania, and M. Palesi, "A multi-objective genetic approach for system-level exploration in parameterized systems-on-a-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 635–645, April 2005.
- [64] F. Brglez and H. Fujiwara, "Special Session on ATPG (Also introducing 'A Neutral Netlist of 10 Combinational Benchmark Circuits')," in *IEEE International Symposium On Circuits and Systems*, 1985.
- [65] <http://www.eecs.umich.edu/~jhayes/iscas/benchmark.html>, [Online].
- [66] C. Thangaraj and T. Chen, "Power and performance analysis for early design space exploration," in *Proc. of the IEEE Computer Society Annual Symposium on VLSI*, 2007, pp. 473–478.
- [67] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *IEEE International Symposium on Circuits and Systems*, pp. 1929–1934 vol.3, May 1989.
- [68] <http://www.fm.vslib.cz/~kes/asic/iscas/>, [Online].
- [69] <http://www.eas.asu.edu/~ptm/>, "Predictive technology model (ptm)," [Online].
- [70] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *Proceedings of the 7th International Symposium on Quality Electronic Design*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 585–590.
- [71] W. Zhao and Y. Cao, "Predictive technology model for nano-cmos design exploration," *J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 1, p. 1, 2007.
- [72] <http://avatar.ecen.okstate.edu/projects/scells/>, [Online].
- [73] <http://www.synopsys.com/>, [Online].
- [74] K. Shi and D. Howard, "Challenges in sleep transistor design and implementation in low-power designs," in *Proceedings of the 43rd annual conference on Design automation*. New York, NY, USA: ACM, 2006, pp. 113–116.

- [75] S. K. Tiwary, P. K. Tiwary, and R. A. Rutenbar, "Generation of yield-aware pareto surfaces for hierarchical circuit design space exploration," in *Proceedings of the 43rd annual conference on Design automation*. New York, NY, USA: ACM, 2006, pp. 31–36.
- [76] B. Stackhouse, S. Bhimji, C. Bostak, D. Bradley, B. Cherkauer, J. Desai, E. Francom, M. Gowan, P. Gronowski, D. Krueger, C. Morganti, and S. Troyer, "A 65 nm 2-billion transistor quad-core itanium processor," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 18–31, Jan. 2009.
- [77] C. McNairy and R. Bhatia, "Montecito: a dual-core, dual-thread itanium processor," *Micro, IEEE*, vol. 25, no. 2, pp. 10–20, March–April 2005.
- [78] C. McNairy and D. Soltis, "Itanium 2 processor microarchitecture," *Micro, IEEE*, vol. 23, no. 2, pp. 44–55, March–April 2003.
- [79] S. Naffziger, B. Stackhouse, T. Grutkowski, D. Josephson, J. Desai, E. Alon, and M. Horowitz, "The implementation of a 2-core, multi-threaded itanium family processor," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 1, pp. 197–209, Jan. 2006.
- [80] C. Thangaraj and T. Chen, "Design target exploration for meeting time-to-market using pareto analysis," *IEEE International Symposium on Circuits and Systems*, pp. 364–367, May 2008.
- [81] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, ser. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.
- [82] S. K. Hsu and et al., "A 110 GOPS/W 16-bit Multiplier and Reconfigurable PLA Loop in 90-nm CMOS," *IEEE Journal Of Solid-State Circuits*, pp. 256–264, 2006.
- [83] J. W. Tschanz and et al., "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors," *IEEE Journal Of Solid-State Circuits*, pp. 1838–1845, 2003.
- [84] B. Chatterjee, M. Sachdev, and R. Krishnamurthy, "A cpl-based dual supply 32-bit alu for sub 180 nm cmos technologies," *Proc. of Intl. Sym. on Low Power Electronics and Design*, pp. 248–251, Aug. 2004.
- [85] S. Mathew et al., "Sub-500 ps 64 b alus in 0.18 um soi/bulk cmos: Design & scaling trends," *IEEE International Solid-State Circuits Conference*, pp. 318–319, 460, 2001.

- [86] D. Blaauw, V. Zolotov, S. Sundareswaran, C. Oh, and R. Panda, "Slope propagation in static timing analysis," *IEEE/ACM international conference on Computer-aided design*, pp. 338–343, 2000.
- [87] SYNOPSYS, "<http://www.synopsys.com/Tools/Implementation/SignOff/Pages/NanoTime.aspx>," 2009.
- [88] CADENCE, "http://www.celestry.com/products_nautilusdc.shtml," 2009.
- [89] V. Veetil, D. Sylvester, and D. Blaauw, "Fast and accurate waveform analysis with current source models," *International Symposium on Quality Electronic Design, ISQED*, pp. 53–56, March 2008.
- [90] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM J. on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.
- [91] H. Sanchez and et al, "Increasing microprocessor speed by massive application of on-die high-k mim decoupling capacitors," *IEEE International Solid-State Circuits Conference*, pp. 2190–2199, Feb. 2006.
- [92] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.
- [93] C. Erbas, S. Cerav-Erbas, and A. Pimentel, "Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 358–374, June 2006.
- [94] S. Rusu, *Interconnect-Centric Design for Advanced SOC and NOC*, 7th ed., J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, Eds. Springer, 2004, vol. Chapter 5.
- [95] E. Zitzler, K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, T. F. (eds, E. Z. Ler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," www.tik.ee.ethz.ch/sop/publicationListFiles/zlt2001a.pdf, 2002.
- [96] E. Zitzler and et al, "<http://www.tik.ethz.ch/~sop/pisa/>," *A Platform and Programming Language Independent Interface for Search Algorithms*, 2008.
- [97] K. Shi and D. Howard, "Sleep transistor design and implementation simple concepts yet challenges to be optimum," Synopsys Inc. and ARM Ltd., Tech. Rep.

- [98] A. Kumar and M. Anis, "Dual-vt design of fpgas for subthreshold leakage tolerance," in *ISQED '06: Proceedings of the 7th International Symposium on Quality Electronic Design*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 735–740.
- [99] W. Hung, Y. Xie, V. Narayanan, M. Kandemir, M. J. Irwin, and Y.-F. Tsai, "Total power optimization through simultaneously multiple-vdd multiple-vth assignment and device sizing with stack forcing," in *Proceedings of the International Symposium of Low Power Electronics and Design*, August 2004. [Online]. Available: <http://www.gigascale.org/pubs/592.html>
- [100] D. Kang, M. Johnson, and K. Roy, "Multiple-vdd scheduling/allocation for partitioned floorplan," in *Computer Design, 2003. Proceedings. 21st International Conference on*, Oct. 2003, pp. 412–418.
- [101] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *ICCAD*, November 2002. [Online]. Available: <http://www.gigascale.org/pubs/160.html>
- [102] X. Chang, M. Zhang, G. Zhang, Z. Zhang, and J. Wang, "Adaptive clock gating technique for low power ip core in soc design," in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.*, May 2007, pp. 2120–2123.
- [103] H. Su, S. Sapatnekar, and S. Nassif, "Optimal decoupling capacitor sizing and placement for standard-cell layout designs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 428–436, Apr 2003.
- [104] N. Menezes and C.-P. Chen, "Spec-based repeater insertion and wire sizing for on-chip interconnect," in *VLSI Design, 1999. Proceedings. Twelfth International Conference On*, Jan 1999, pp. 476–482.
- [105] V. Nawale and T. W. Chen, "Optimal useful clock skew scheduling in the presence of variations using robust ilp formulations," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. New York, NY, USA: ACM, 2006, pp. 27–32.