A Second Generation GENEric SYstems Simulator (GENESYS) for a Gigascale System-on-a-Chip (SoC)

A Dissertation
Presented to
The Academic Faculty

by

Steven P. Nugent

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2005

Copyright © 2005 by Steven P. Nugent

A Second Generation GENEric SYstems Simulator (GENESYS) for a Gigascale System-on-a-Chip (SoC)

Approved by:

Prof. James D. Meindl, Advisor
Electrical and Computer Engineering
Georgia Institute of Technology

Prof. D. S. Wills, Co-Advisor
Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Jeffrey A. Davis
Electrical and Computer Engineering
Georgia Institute of Technology

Prof. M. Swaminathan
Electrical and Computer Engineering
Georgia Institute of Technology

Prof. William R. Callen
Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Paul Kohl
Biomolecular and Chem. Engineering
Georgia Institute of Technology

Date Approved: March 30, 2005

This work is dedicated to my beloved parents,

Michael J. and Beryl J. Nugent

ACKNOWLEDGEMENTS

I would like to take this opportunity to offer my sincere gratitude to those who have made this work possible and who have provided support and guidance for my efforts. First, Profs. James D. Meindl and D. Scott Wills for their time and efforts in directing and encouraging the course and aims of this research. The examples that they have set for me will remain influential for the remainders of my professional and personal life. I would also like to acknowledge the other members of the reading committee for assisting in the academic process.

The GSI administrative assistant, Jennifer Tatham, whose tireless guidance and support have helped to make the GSI group a coherent 'family' of researchers. She has greatly enhanced the graduate experience for every student fortunate enough to be associated with her.

My parents, Michael and Beryl Nugent, brothers, Mark and Robert, have provided ample encouragement in my personal and academic development. Their continued love and support will aid me in all my future endeavors.

Dr. John Eble deserves recognition for his ground breaking work in system modeling which provided the framework upon which my contributions are built. Many stimulating conversations early in my academic career set the course for much of my academic efforts.

Finally, I would like to acknowledge both the Defense Advanced Research Projects Agency (DARPA) and the Semiconductor Research Corporation (SRC) for sponsoring the bulk of this research.

TABLE OF CONTENTS

DEDICATION **iii**

ACKNOWLEDGEMENTS **iv**

LIST OF TABLES **viii**

LIST OF FIGURES **x**

SUMMARY **xiv**

I INTRODUCTION **1**
1.1 Motivation and Background 1
1.2 Problem Statement 3
1.3 Summary of Contributions 4
1.4 Dissertation Outline 5

II SYSTEM LEVEL MODELING **7**
2.1 Introduction 7
2.2 Related Work 7
2.3 GENESYS 11
2.3.1 Throughput Estimation in GENESYS 12
2.3.2 GSI Interconnect Architectures in GENESYS 28
2.4 Extending GENESYS to Heterogeneous SoCs 36

III HETEROGENEOUS SOC METHODOLOGY **37**
3.1 Introduction 37
3.2 Block Modeling Methodology 38
3.3 Global Interconnect Methodology in GENESYS 2k4 43
3.4 Cell Placement 47
3.4.1 Manual Cell Placement 47
3.4.2 Automated cell placement 49
3.4.3 Placement effects on efficiency 52
3.5 Validation of the block SoC methodology 57
3.5.1 Methodology 58
3.5.2 Results 59

3.6 Conclusion 59

IV PHYSICAL ON-CHIP BUS MODELING 64
4.1 Introduction 64
4.2 A Generic On-Chip Bus Model 66
4.2.1 Bus specification in GENESYS 2004 66
4.2.2 Bus Link Models 67
4.2.3 Calculating the Bus Length 70
4.2.4 Bus Delay 73
4.2.5 Bus Power dissipation 76
4.2.6 Bus Routing Area 78
4.2.7 Bus placement optimization 81
4.3 Bus limitations on system throughput 83
4.3.1 Transaction model 84
4.3.2 Transaction rate limit 89
4.4 CoreConnect Chip Example 96
4.4.1 The IBM CoreConnect Architecture 96
4.4.2 440GP Simulation results 97
4.4.3 Performance scaling and trends for the 2004 ITRS technologies 99
4.5 Conclusions 103

V ON-CHIP DRAM MODEL 106
5.1 Introduction 106
5.2 DRAM Cell Model 107
5.2.1 Cell area model 108
5.2.2 Cell energy model 109
5.2.3 Cell delay 118
5.3 DRAM Array model 119
5.3.1 DRAM Array area 119
5.3.2 DRAM array power dissipation 120
5.4 A Gigabit on-chip DRAM 122
5.5 Conclusions 122

VI GIGASCALE CELLULAR ARRAY ARCHITECTURES 126
6.1 Introduction 126
6.2 Stochastic global network for gigascale array processors 126
6.2.1 Simulation methodology 127
6.2.2 Global and local interconnect resources 128
6.2.3 Local and global clock frequency 133
6.2.4 Power dissipation 137
6.3 Design space for global communication 138
6.3.1 Simulation methodology 138
6.3.2 Shared global bus 139
6.3.3 Nearest neighbor wiring approach 139
6.4 Conclusion 141

VII CONCLUSION AND FUTURE RESEARCH **143**
7.1 Overview 143
7.2 Summary of Results 145
7.3 Future Research 148
7.3.1 Core Modeling Enhancements 148
7.3.2 System Modeling Enhancements 149

APPENDIX A — 2003 ITRS TECHNOLOGY NODES **151**

APPENDIX B — GENESYS 2004 MANUAL **153**

APPENDIX C — GENESYS HELP FILE **170**

APPENDIX D — MACHINE DESCRIPTION **172**

REFERENCES **179**

VITA **184**

LIST OF TABLES

1 ITRS feature size 15

2 Global interconnect for the 2013 ITRS microprocessor. 19

3 Cost analysis parameters 21

4 Parameter values for optimal zone size analysis 24

5 GSI microprocessor parameters 27

6 Parameters for repeater analysis 34

7 Relative LDI performance 34

8 Cell placement efficiencies for generic SoC example containing 18 cells for manual, in-order, and size based placement. 55

9 Megacell Placement Efficiency for Commercial SoCs. 57

10 Validation results for block methodology 61

11 Bus length calculations for various fanouts and configurations for generic SoC floorplan of Figure 44. 73

12 Bus delay at $5\mu m$ line width for the three generic bus line types from Figure 41. 75

13 Average bus power comparisons with actual data 77

14 Bus area requirements for example from Figure 45 80

15 Simulation results for 32 bit shared, global bus connecting all cells for a $300mm^2$, 20 megacell generic SoC. 81

16 Simulation results for bus length and delay for various cell placement schemes. 82

17 Optimal pipeline depth 87

18 Effects of bus pipelining on peak data rate 89

19 Parameters for bus throughput analysis 95

20 Comparison of GENESYS results for PPC440 processor core. 104

21 GENESYS simulation results for the IBM 440GP SoC. 104

22 Maximum PLB bus bandwidth for the IBM 440GP SoC 104

23 Technology scaling results for the 440GP example 105

24 Technology scaling simulation results for the 440GP PLB bus illustrated in Figure 55. 105

25 Voltage swing dependencies for a generic DRAM cell 114

26 DRAM cell leakage states 116

27 Array area comparison 120

28 DRAM area comparison 120

29 DRAM simulation results 121

30 Giga-bit embedded DRAM 123

31 2003 ITRS technology 129

32 Shared bus statistics for processor array 140

33 Bus performance for a nearest neighbor communication network 141

LIST OF FIGURES

1 GENESYS Hierarchy 11

2 CPI dependencies 13

3 Parameter extraction for empirical CPI model 14

4 Throughput projections for the ITRS 16

5 Random logic delay model 17

6 Global interconnect delay model 17

7 Throughput vs die size for the 2013 (18nm) ITRS microprocessor. 19

8 Power and clock frequency for 2013 ITRS microprocessor 20

9 Performance-Cost index for 2013 ITRS microprocessor 22

10 Minimum die size vs. gate count 23

11 CPI variation with gate count 25

12 Clock frequency variation with gate count 25

13 Raw throughput vs. gate count 26

14 Throughput vs. gate count for various cache sizes 27

15 Throughput vs gate count for 2013 generation 28

16 Throughput vs tier organization 31

17 Normalized global interconnect cross-sectional area vs. tier organization . 31

18 Throughput vs. number of wiring levels 33

19 Frequency increase with die size for repeater & single driver schemes . . 35

20 Repeater and single driver die size 36
21 Block diagrams for Uniprocessor and SoC approaches 38
22 Heterogeneous Simulation Flow 40
23 Global interconnect sizing algorithm 42
24 Homogeneous interconnect distribution 43
25 Heterogeneous global interconnect distribution 44
26 Cell placement grid example 48
27 Example cell placement specification 49
28 Auto-placement algorithm 53
29 Row assembly for auto-placement 53
30 Generic SoC for efficiency comparison 54
31 Generic SoC floorplan with in-order placement 55
32 Generic SoC floorplan for size based cell placement 56
33 Intel Itanium2 die micrograph 58
34 Emotion engine die micrograph 59
35 UltraSPARC die micrograph 60
36 PowerPC die micrograph 61
37 Die area comparison 62
38 Clock frequency comparison 62
39 Power dissipation comparison 63
40 Speed trends in on-chip/off-chip connections 65
41 Generic Bus Line Models 68
42 Physical model for bus lines 69
43 Illustration of bounding box calculation 72
44 Bus length example floorplan 72

45 Delay trends for generic bus line types 74
46 Average bus power consumption for three link types 78
47 Average bus power consumption at a clock frequency of 200MHz 79
48 Generic SoC processor-memory cores and on-chip bus 84
49 Non pipe-lined bus transaction model 85
50 Pipelined bus transaction models 86
51 Bus transaction limit vs latency 90
52 Bus transaction rate vs miss rate 93
53 Bus transaction rate vs processor core frequency 94
54 Bus limitations on system throughput 95
55 Block diagram of an SoC implementation based on the IBM 440GP chip 98
56 Power and Frequency for technology scaling example 101
57 Power density technology scaling example 102
58 Breakdown of power dissipation for technology scaling example 102
59 Basic circuit model for a DRAM cell 107
60 DRAM feature size vs calendar year 109
61 DRAM cell area vs calendar year 110
62 DRAM cell area in feature sizes vs calendar year 110
63 Generic DRAM cell layout 111
64 DRAM cell area model comparison with ITRS data 112
65 Cross-sectional diagram of DRAM cell with leakage currents 117
66 Tunneling current model for DRAM leakage 124
67 Generic DRAM array layout 125
68 Generic cellular array floorplan 127
69 Node sizing vs Node count 128

70 Local interconnect demand vs number of nodes for selected ITRS technology generations . . . 130

71 Global interconnect demand vs number of nodes for selected ITRS technology generations . . . 131

72 Global interconnect pitch vs number of nodes for selected ITRS technology generations . . . 132

73 Global clock frequency vs number of nodes and technology generation . . 134

74 Local clock frequency vs number of nodes and technology generation . . 135

75 Ratio of local to global clock frequency . . . 136

76 Processor array power dissipation . . . 137

77 Global network types for processor array . . . 138

78 Technology characteristics for near term years . . . 152

79 Technology characteristics for long term years . . . 152

SUMMARY

It is the fundamental thesis of the gigascale integration (GSI) research group that the performance of future multi-billion transistor systems is governed by a hierarchy of limits. The ability to leverage advancements in materials, devices, and circuit related technologies for future designs is heavily dependent on limitations imposed by system level constraints. The emerging dominance of System-on-a-Chip solutions for complex IC's necessitates the development of an early performance estimation tool specifically geared towards SoC type designs. Future SoC designs will incorporate many peripherals that were previously off-chip resulting in system which depart significantly from traditional architectures. A newly enhanced generic systems simulator, GENESYS 2004, is offered as a fast, flexible approach to exploring system level limitations for complex GSI systems. GENESYS 2004 incorporates a hierarchical block-based system modeling methodology based on physical system level design parameters to rapidly project performance.

The new system methodology is verified by comparing the simulated performance and characteristics of existing microprocessors/SoCs against actual data. Several simulation methods incorporating the projected system characteristics from the 2003 edition of the ITRS show that a near term limitation on the growth potential of gigascale SoCs is limited by excessive power dissipation. Beyond the 2006 time frame the projected power dissipation for GSI systems rapidly exceeds the ITRS maximum allowable values.

This thesis presents a methodology for rapidly projecting the performance of gigascale systems-on-a-chip and exploring limitations on the growth potential of these future systems.

CHAPTER I

INTRODUCTION

1.1 Motivation and Background

Beginning with the introduction of the integrated circuit in 1958, technology-scaling trends have driven an exponential increase in on-chip integration over the previous four decades. Using the Intel line of microprocessors as a key example, the 8086 processor introduced in 1978 supported 29K transistors while the next generation Itanium processor boasts on the order of 220M transistors [1][2]. This trend represents an increase of 4 orders of magnitude over the past two decades. With this increasing integration, the complexity of chip designs has progressed from the single issue, sequential processors of the 1970's to the multiple-pipelined, out-of-order, speculative systems of the present [2]. The 2003 ITRS projects that by 2018 the number of transistors integrated on-chip will grow from several hundred million today to more than 14 billion for high performance processors [3]. The cost of debugging and validating high complexity designs make reducing design complexity a key goal for chip architects [4]. In response to the increasing integration and complexity the semiconductor industry has moved steadily towards System-on-a-Chip (SoC) solutions for VLSI systems. This shift in recent years towards SoC designs for complex systems presents significant challenges for the design and test community [5][6]. While these issues are of paramount importance to the future viability of SoC designs in GSI systems, the impact of technology scaling on key system performance metrics such as

clock frequency, power dissipation, and chip area cannot be ignored.

The 2003 edition of the ITRS projects that the 14 billion transistor chip of 2018 will operate at a local clock frequency of approximately 53 GHz, occupy an area of 310mm^2, and consume 300W [3]. The challenges to achieving such performance are daunting when the general tradeoffs are assessed. For example, reaching the target power dissipation numbers for this node is problematic due to both the increased integration and higher clock frequencies. Given the relative inflexibility of limitations arising from lower levels of the GSI hierarchy, the best opportunities for overcoming barriers to continued performance growth lie at the system level.

SoC design methodologies promise to play a dominant role in the future of GSI systems. Given the inherent challenges in the design and production of such systems, it is vital that the interactions between technology and architecture are explored to identify optimal design spaces for GSI SoCs. It is theorized that potential performance improvements for GSI systems are governed by a key set of limits that are organized hierarchically as fundamental, material, device, circuit, and system [7]. The GENEric SYstems Simulator (GENESYS) is designed to take a system description incorporating each level of the GSI hierarchy and produce estimates of key system performance metrics [8].

As chip designs become increasingly based on SoC methodologies the emergence of intellectual property (IP) core methodologies in which systems are composed of previously defined blocks or macro-cells from design libraries, results in systems that are inherently heterogeneous in nature. A traditional homogeneous system model with an interconnect distribution based on the assumption of a uniform gate-array is no longer adequate for describing the system architecture on the interconnect level

[9][10]. A new system modeling methodology has been developed for explorations of the SoC design space. This methodology incorporates a system description which mimics the physical layout of a SoC combined with a dual interconnect distribution for global and local interconnects to more accurately estimate system performance. In addition this model allows for novel investigations into the effects of macro-cell placement on the global interconnect distribution and its impact on the system clock frequency, power consumption, and throughput for future technology generations. The system methodology is further extended by the incorporation of explicitly defined interconnect/busses into the architectural description of the SoC. Another critical advancement over the generic homogeneous modeling approach is the incorporation of a new on-chip DRAM model for high density memory storage a key feature on numerous SoCs. GENESYS 2004 uniquely preserves a rich core of empirical and analytical modeling while providing a highly flexible and generic system level methodology for making projections of SoC performance into the next decade.

1.2 Problem Statement

Projections of future system performance for gigascale SoCs based on a detailed hierarchy of limits requires a system level methodology that is sufficiently descriptive and flexible enough to provide an accurate basis for simulating systems that may vary widely in characteristics from one implementation to another. The enabling observation is that no matter the complexity and design of the system in question, all of the constituent parts are derived from increasingly simpler components. Devices are arrangements of semiconductor materials (silicon,dopants,oxides...). Circuits are collections of interconnected devices. Functional units are groupings of circuits. Systems are collections of inter-related functional units. Changes made at the material,

device, or circuit levels may filter up to impact the performance at higher levels of the hierarchy, but alterations in the system design such as the placement of the megacells does not modify the behavior at lower levels of the hierarchy. Therefore, a more flexible system methodology may 'layer' onto the existing framework.

Based on this observation, a new methodology for describing a fully realized SoC and then engaging the necessary body of hierarchical modeling to produce projections that correspond with high fidelity to the specified system is required. This thesis address the question of how to estimate the system performance of complex GSI SoCs with a flexible, but generic description, as to enable the exploration of opportunities and limitations for future designs.

This thesis describes 1) a hierarchical block-based system methodology for a gigascale SoC that significantly improves upon the performance of the previous system model featuring a dual distribution approach based upon global and local interconnection networks, 2) a set of physical models for on-chip DRAM, 3) a methodology for explicitly specifying and simulating the performance of critical global interconnect networks such as system busses to enhance throughput projection for SoCs.

1.3 Summary of Contributions

The following list is a summary of the contributions presented in this dissertation.

- Definition, implementation, and engagement of a machine description for a heterogeneous system-on-a-chip.
- Implementation, validation, and engagement of a methodology for projecting

the system performance of an SoC implementation.

- Integration of a key global interconnect distribution and algorithm for generating a complete stochastic global net distribution for GSI SoCs.
- Evaluation of existing SoC designs showing improved fidelity over a previous system model.
- Derivation, implementation and engagement of bus modeling for critical global interconnects.
- Development and implementation of a novel automated cell placement routine for heterogeneous SoCs with constraints for maximizing cell clustering density and minimizing the length of on-chip busses.
- Derivation, verification, and integration of an on-chip DRAM model into GENESYS 2004.

1.4 Dissertation Outline

Chapter II summarizes previous work in generic system modeling including the early generation of GENESYS and its capabilities. The early generation simulator is engaged in original studies to highlight key features of the homogeneous system model utilized.

Chapter III introduces the concept of the hierarchical block-based methodology for SoCs. The system description and simulation methodology are discussed in detail. The algorithm for assessing the global routing requirements is presented. The impact of cell placement on the global distribution and system performance is examined from a stochastic viewpoint. The new system methodology is verified by comparison with existing commercial chips and contrasted with the results from the

homogeneous model. A novel ultra-fast automated cell placement algorithm with optimization constraints for cell clustering density and bus length minimization is discussed.

Chapter IV presents a novel on-chip bus model for assessing the performance impact of critical global interconnects. The impact on global routing resources and delay is examined. The impact of bus performance on system throughput is also discussed.

The on-chip DRAM modeling is detailed in Chapter V. The generic DRAM model is introduced and verified by comparison to commercial implementations. Integration into the GENESYS 2004 system framework is discussed.

Chapter VI explores system performance for cellular array architectures using the 2003 ITRS as a starting point for estimating system configurations. The tradeoff between local and global interconnect with increasing cell count is examined. The suitability of three distinct wiring schemes for array architectures is examined. The power dissipation and clock frequency for each technology generation is evaluated against projected ITRS data.

The dissertation is summarized and the major conclusions presented in chapter VIII. Promising subjects for future research and extension of the GENESYS 2004 framework are discussed.

Finally, the appendices gather ancillary material such as the GENESYS user's guide, sample help file, and example machine description.

CHAPTER II

SYSTEM LEVEL MODELING

2.1 Introduction

System performance modeling can be described as consuming key system design parameters to produce estimates of performance for prediction or early design feedback [11]. As semiconductor technologies continue to scale toward the nanometer regime, the impact of device technology, interconnect, and system architecture on chip performance makes chip/system level modeling vital to identifying key challenges and opportunities for future GSI systems. Numerous attempts to model the system level performance of complex ICs have been undertaken with varying degrees of rigor.

This chapter presents a brief overview of earlier research efforts in the area of chip modeling and exercises the first generation generic systems simulator (GENESYS) to explore various components of microprocessor performance in a homogeneous environment.

2.2 Related Work

Bakoglu presented the first system model combining elements of technology, architecture, and system packaging in SUSPENS (the Stanford University System Performance Simulator) [12]. This system model is comprised primarily of eleven key

expressions and associated parameters. By exercising this system model, the author estimated the clock frequency, die size, and power dissipation for both CMOS and GaAs technologies. However, this system model, with its sparse parameter set, focuses almost exclusively on the circuit and system levels. This model ignores key contributions made at the lower levels of the GSI hierarchy [7], the design of the memory hierarchy, and throughput estimation.

Sai-Halasz introduced a generic cycle-time model for projecting performance trends in uniprocessors [13]. Unlike SUSPENS, this work focused primarily upon the estimation of maximum clock frequency rather then complete system modeling. However, Sai-Halasz significantly extended the modeling effort by taking into account both multi-tier wiring architectures with via-blockage modeling and area requirements for on-chip memory.

A 1990 effort by Hoppe and Neuendorf [14] focused on circuit level optimizations but included analytical models for frequency, area, and power. The detailed critical path model takes into account various gate geometries (NAND,NOR...etc.) and estimates the wiring capacitance from layout information. The area model is also based on empirical data for estimating the footprint of a logic gate. The power dissipation is estimated via the product of the CV^2 energy and clock frequency. The methodology is highly detailed but the extrapolation from circuit level optimization and modeling to the system level area and power models neglects the impact of interconnect scaling on system performance. The modeling presented in this work is well suited for circuit level estimations only.

A better example of a system modeling methodology is presented by Goel and Schuermeyer [15]. The authors developed a simulator, NCHIPSIM, for projecting the performance indicators of an integrated chip. Like SUSPENS, a key aspect of the system modeling is the calculation of the wiring distribution and average interconnect length from Rents parameters and the Donath distribution [16][17]. The authors applied NCHIPSIM to an existing microprocessor design and demonstrated accurate estimation of the system clock frequency, chip size, and power consumption. The system modeling in NCHIPSIM is very similar to SUSPENS, but is limited by the assumption of NMOS as the circuit implementation technology. This model does not significantly extend the system modeling work presented in [12].

The Rennsselaer Interconnect Performance Estimator (RIPE) was developed to provide early analysis of the system performance constraints introduced by interconnect related issues [18][19]. In this capacity RIPE takes an interconnect centric approach to system modeling. The average wire length is calculated via the Rents parameters. The interconnect distribution is reduced to average length wire and a number of maximum length wires. Several IBM, Intel, and Alpha processors are simulated with RIPE and the results show accurate estimations of clock frequency and power dissipation. RIPE does not provide a complete system model as the chip area is an input rather than an output. The primary drawback to RIPE is that the input set consisting of system, technology, and interconnect requires detailed information regarding the capacitance, resistance, and area. When projecting performance for future technology generations, this information may not be readily available.

Another recent system modeling effort is the Berkeley Advanced Chip Performance Calculator (BACPAC) developed by Sylvester [11]. BACPAC incorporates a set of inputs codified as interconnect, device, and system to produce estimates of

the chip area, frequency, and power. Unlike most previous system models, BACPAC models both the local and global interconnects. The system model assumes that the wire-length distribution derived by Donath applies to both the local and global interconnect distributions. Furthermore, the BACPAC model assumes that the system architecture is partitioned into blocks of between 50,000 to 100,00 gates. The power modeling utilizes a hierarchical approach in which different components of the power consumption are individually modeled. This approach makes BACPAC one of the more comprehensive system modeling tools available. The major drawbacks of the methodology are its reliance on an outmoded interconnect distribution for both local and global interconnect modeling and a broad assumption regarding the partitioning of logic at the system level.

The generic model (GenM) for SoC architectures is used for the rapid estimation of performance in terms of energy and latency [20]. The core system model utilized in this work consists of a processor supporting dynamic voltage scaling, reconfigurable logic, and memory . It takes as its key input parameter set an array of operating states for the processor and memory. Unlike the other modeling approaches discussed above, the GenM model replaces the technology modeling with a component level abstraction. The time and energy costs for specific state transitions and interactions between the system components are entered as input. With this framework in place, it is possible to map an application to the GenM model and evaluate its execution time and energy cost. This approach is shown to generate reasonably accurate results for several sample processors, however the interactions between technology and architecture are hidden from the user. The GenM system model focuses solely on energy and execution time without any estimates of physical system characteristics such as die size or clock frequency. In order to utilize this simulator the application, processor states, and transition time/energy costs must be known. This

makes this simulation framework unsuitable for investigating the performance of future billion transistor systems.

2.3 GENESYS

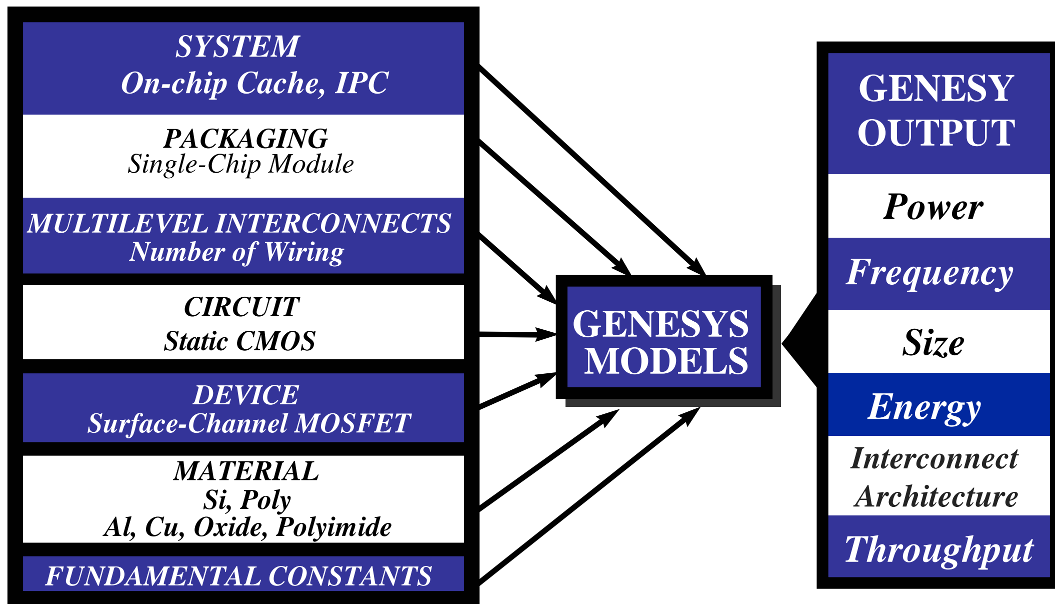

Figure 1: Visualization of GENESYS hierarchical modeling approach

The first system modeling tool to fully incorporate the entire GSI hierarchy in its simulation methodology was GENESYS [8][21]. GENESYS established a uniprocessor system model consisting of a large block of random logic with between 0 to 3 cache memories. Under this model, a newly derived stochastic interconnect distribution [9][10] improving upon the previously derived method in [17] is utilized to derive the average wire length for the random logic. In addition to an improved interconnect distribution, GENESYS introduced detailed modeling of the on-chip memory. These models include area, access time and power dissipation for numerous physical and logical cache organizations. Furthermore, GENESYS was the first

such system modeling tool to incorporate a compact CPI model for estimating the system throughput [22]. This CPI model incorporates contributions from both the random logic network (via an empirical model) and the memory hierarchy (as a function of miss rate tables, bus frequency, and DRAM access time). A diagram of the GENESYS approach is illustrated in Figure 1.

The following sections provide relevant information regarding GENESYS modeling in the context of original studies of microprocessor performance characteristics. The primary areas of focus for these studies are throughput estimation, and GSI interconnect architectures.

2.3.1 Throughput Estimation in GENESYS

Delivered instruction throughput, measured in completed instructions per second (IPS), is a widely used metric for comparing processor performance. It is typically computed using standard benchmarks (e.g. SPEC95...). Instruction throughput is the product of cycle time (T_c) and cycles-per-instruction (CPI), both of which are processor implementation dependent.

$$IPS = (T_c \cdot CPI)^{-1} \tag{1}$$

Projections of future microprocessor performance have historically centered on clock frequency as the primary performance metric [23][24]. However, clock frequency alone does not fully determine the processor performance. In this section a previously derived empirical CPI model is exercised to give insight into performance limits for future GSI microprocessors [25]. The logic-memory model takes into account the CPI of a random logic block and the impact of the memory subsystem. Figure 2 shows the globally observable metrics that are correlated to the CPI in this model.

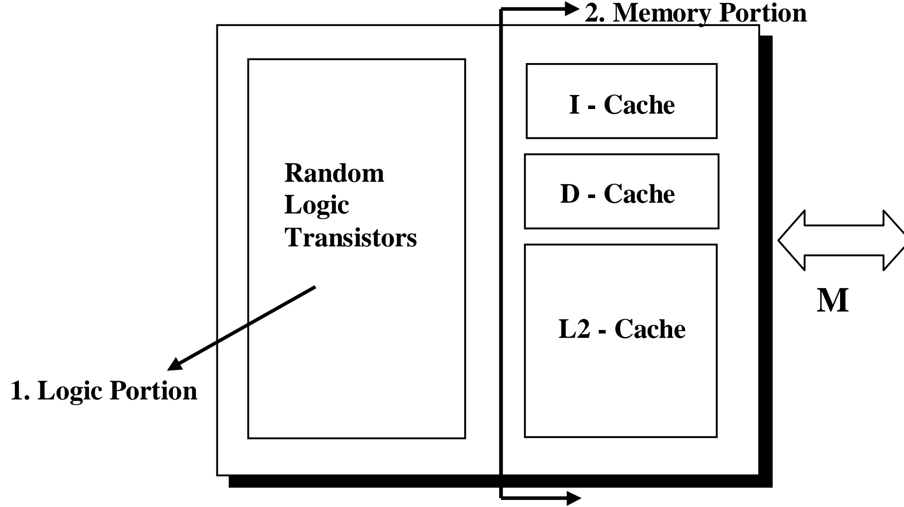

Figure 2: A global view of CPI dependencies for the logic-memory CPI model.

2.3.1.1 Empirical CPI Model for random logic

The first correlation parameter is the number of logic or non-cache transistors on the chip. This factor influences the logic component of the CPI which is computed via the following expression:

$$CPI_{logic} = E_c N_{gate}{}^{-E_e} \quad (2)$$

Where E_c is the empirical coefficient, N_{gate} is the gate count, and E_e is the empirical exponent. The formulation of the relationship between the gate count and the CPI derives from the observation that historically increasing on-chip integration has been utilized to implement architectural enhancements that reduce the CPI. The values for E_c and E_e are found by fitting the power law model to actual data taken from an existing line of microprocessors as shown in Figure 3. The values used in this study are $E_c = 51829$ and $E_e = 0.7725$.

2.3.1.2 Memory CPI Model

The memory component of the CPI model captures the impact of the cache memory (capacity and logical organization) and the main memory (off-chip memory

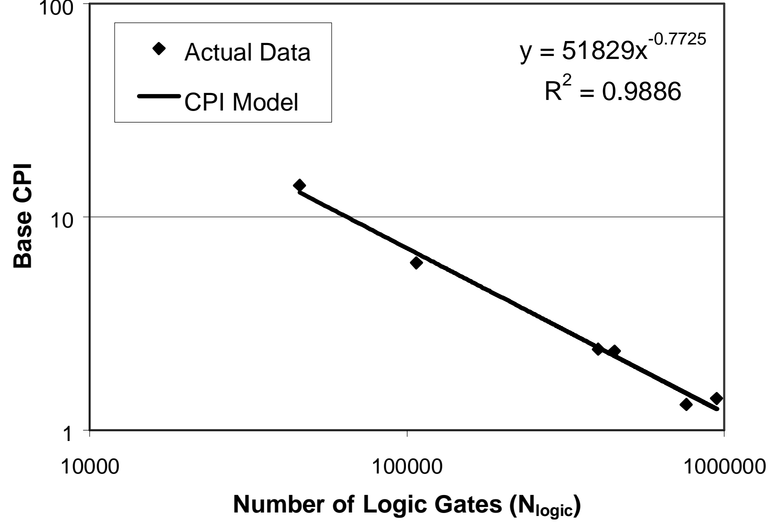

Figure 3: Parameter extraction for empirical CPI logic model for Intel x86 processor family.

access). A simple expression for the memory CPI contribution is listed in the equation below:

$$CPI_{mem} = M_{refs} M_{rate} M_{penalty} \tag{3}$$

Where M_{refs} is the reference frequency expressed as a % of the instructions that make a reference to the memory subsystem, M_{rate} is the % of references resulting in a cache miss, and $M_{penalty}$ is the delay in cycles taken to access the main memory to retrieve data. GENESYS is capable of considering three distinct cases: a single unified cache, a split cache (instruction and data),and a split cache with unified level 2 cache.

$$CPI_{mem} = (M_{iref} + M_{dref}) m_{rate1} \left(\lceil t_{mm} f_c \rceil + \left\lceil \frac{8B_1}{b_{bus}} \right\rceil \left\lceil \frac{f_c}{f_{bus}} \right\rceil \right) \tag{4}$$

Equation 4 is a more precise calculation of the memory CPI component for the single unified cache where M_{iref} and M_{dref} are the instruction and data reference frequencies, m_{rate1} is the miss rate of the cache, t_{mm} is the main memory access time, f_c is the processor clock frequency, B_1 is the size of a cache line in bytes, b_{bus} is the width of the memory data bus in bits, and f_{bus} is the bus frequency. The expressions for the

Table 1: Minimum feature size for ITRS technology generations

1999	180nm
2001	150nm
2003	130nm
2005	100nm
2011	50nm

2 and 3 cache cases are similar to Eq. 4 but with additional terms. The cache miss rate is determined from the Design Target Miss Rate (DTMR) tables introduced by Smith and updated by Flynn [26][27][28].

2.3.1.3 Simulation Results for a GSI Microprocessor

The CPI for three different processor implementations (x86, AMD, and RISC) is calculated from Eqs. 2 and 3 then combined with ITRS projections for cycle time to determine the throughput trend for the roadmap. The results are shown in Figure 4 taken from [22]. The feature size for each technology generation is given in Table 1. As bench-marked against the 2x trend the throughput trend for each processor degrades significantly at the end of the roadmap. The logic CPI and clock frequency are linear on a log scale. Therefore, the cause of the degradation must be the memory component of the CPI. In fact, the contribution of the memory to the total CPI increases from approximately 35% in 1999 to about 65% in 2011. In order to maintain the 2x/24 month trend the memory CPI must scale at a rate comparable to the logic component. The increasing gap between core logic and memory performance indicates that the design and implementation of the memory hierarchy will play a vital role in efforts to increase the performance of future GSI systems.

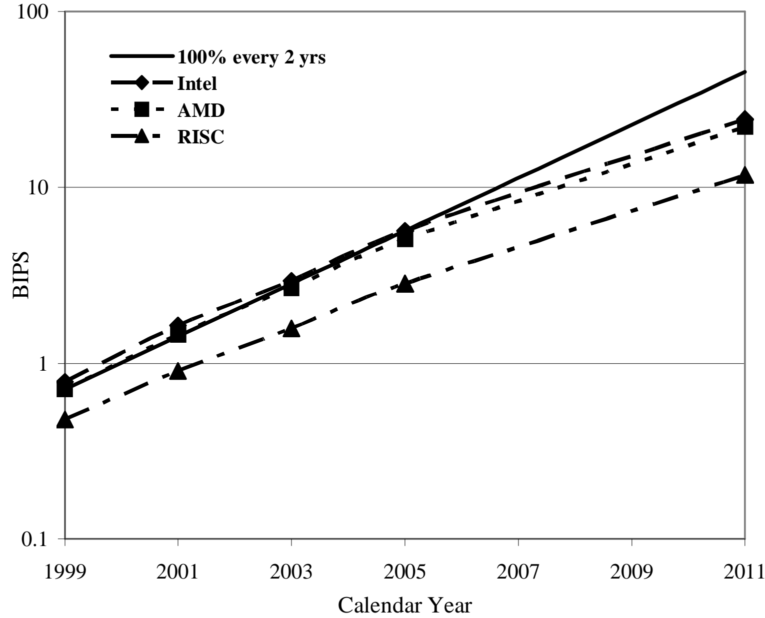

Figure 4: Throughput projections for Intel, AMD, and RISC processor families for the ITRS versus 2x improvement every 2 years [22].

2.3.1.4 Effects of Die Size on System Performance

In this study GENESYS is utilized to explore the impact of varying die size on system performance for a GSI Microprocessor, to develop a cost performance index for identifying optimal performance per unit cost, and to determine the optimal size of a random logic block for maximum throughput. The architectural component of performance is determined via the empirical throughput model discussed in section 2.3.1.1 and section 2.3.1.2. The clock frequency is determined by either the delay through a chain of gates wired by average length interconnects or the delay of the longest global interconnect [25]. The basic delay models are illustrated in Figures 5 and 6. Repeater insertion is assumed unless otherwise noted.

GENESYS Optimization and Die Size

GENESYS utilizes several optimization methods and metrics. The methods include gate, interconnect, and both. The gate method makes the device width the primary variable through which the optimization is performed. The interconnect

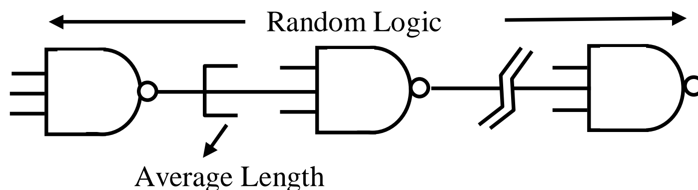

Figure 5: Random logic delay model utilized in GENESYS. A chain of canonical gates connect by average length wires.

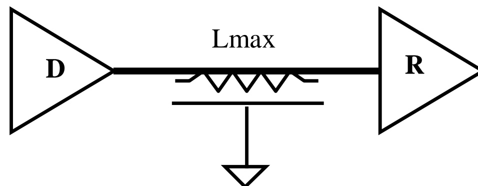

Figure 6: A schematic of the global distributed RC interconnect delay model. The value Lmax is approximately twice the chip edge length and D/R denote the driver and receiver respectively. The single driver case is shown but optimal repeater insertion is also an option.

method sets the wiring dimensions as the optimization variable. The metrics are performance and die size. The constraint for the performance metric is a target die size specified by the user. GENESYS will attempt to find the gate width/interconnect dimensions which maximize performance without violating the constraint. The die size metric constraint is given as a target clock frequency. In this case the simulator attempts to find the smallest die size for which the target clock frequency is met.

Die Size vs. Throughput

Two of the key metrics determining the viability of a microprocessor design are the cost per die and the performance. The complete processing of a wafer can be modeled as a fixed cost, so the first metric is closely related to the die size as the number of dies per wafer is inversely proportional to the die size. Therefore for this study the die size will function as our cost metric with larger die sizes indicating increasing cost. The second metric is performance. The ultimate performance metric for

a microprocessor is throughput (in instructions/operations per second). GENESYS incorporates an empirical model for projecting the CPI of a GSI processor. When combined with maximum clock frequency projections the CPI model allows for estimation of the system throughput that will serve as the performance metric in this study.

Simulation Methodology

These simulations are performed for a GSI Microprocessor conforming to the technology and system characteristics specified in the 2001 ITRS for the 2013 technology generation [23]. The minimum feature size is 18nm while the pertinent system characteristics are 1.56 billion transistors with 12 metalization layers. For this study, GENESYS will optimize both the interconnect dimensions and the average device W/L ratio for maximum clock frequency. The target die size is varied from $55mm^2$ to $275mm^2$. This approach allows for the investigation of the trade-offs between die size and overall system performance.

Simulation Results

Figure 7 shows the response of the system throughput to increasing die size. A substantial increase in performance of nearly 200% comes at a cost of a 150% increase in the die size of the processor. The throughput increases sharply between 55-$70mm^2$ with no appreciable performance gain beyond $140mm^2$. The $55mm^2$ point represents the smallest die size for which GENESYS could meet the wiring demand while $275mm^2$ is the largest die size practical due to performance saturation. Where does the performance increase come from? The CPI component of the throughput is independent of the die size, so the increase in system throughput is the result of increasing clock frequency. The higher clock frequencies driving the trend from Figure 7 is primarily the result of GENESYS optimization of interconnect dimensions. As the target die size is increased, the additional routing area allows for an increase

Table 2: Global interconnect for the 2013 ITRS microprocessor.

Die Size	$55mm^2$	$275mm^2$
Length	1.28cm	2.99cm
R (Ohms)	161K	90
Total Delay	2.66ns	0.30ns
ToF Delay	0.06ns	0.14ns

in the cross-sectional area of interconnects. This reduces the resistance of the wire resulting in lower RC delay. This effect is most prevalent in the global interconnects because the delay based optimizations run by GENESYS target the wiring levels with the highest delays. Therefore, the increased routing area is reducing the global interconnect delay.

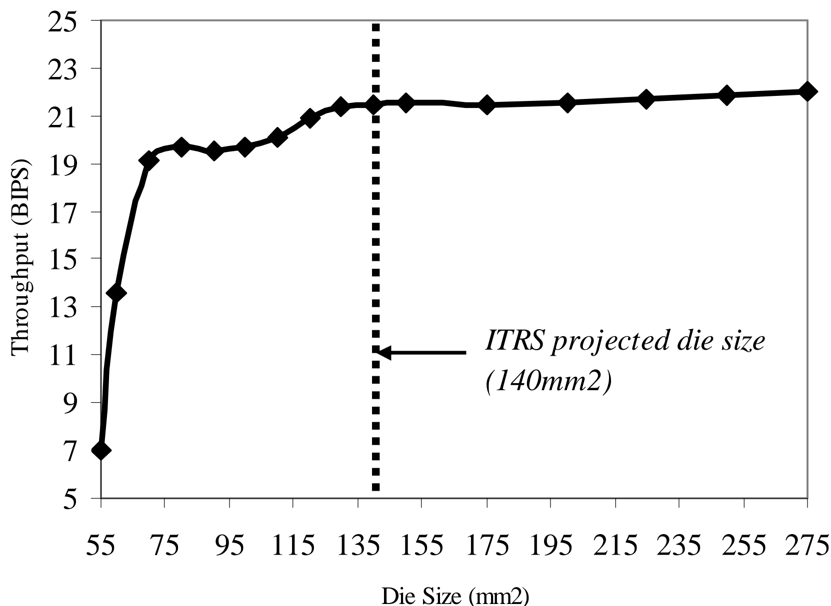

Figure 7: Throughput vs die size for the 2013 (18nm) ITRS microprocessor.

The data of Table 2 show that even through the interconnect length increases by more than 100%, the total delay drops by 89%. For a $55mm^2$ die size the time of flight delay accounts for only about 2% of the total delay. For the $275mm^2$ die size example the time-of-flight delay comprises nearly 47% of the total delay. Figure 8

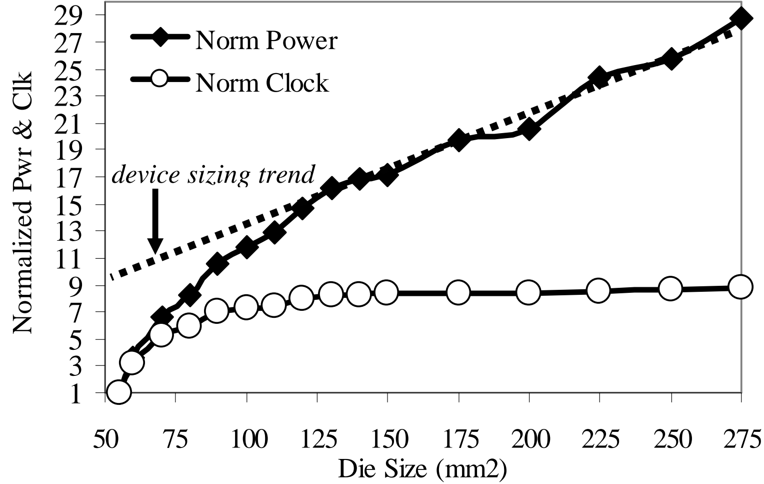

Figure 8: Normalized power dissipation and clock frequency vs die size for the 2013 ITRS microprocessor. The dotted shows the power dissipation trend due to increasing device W/L ratio.

shows the normalized clock frequency and power dissipation for these simulations. The clock frequency saturates as the random logic critical path becomes the dominant contributor to the processor cycle time. The increasing die size allows for larger width gates. However, the increase in average interconnect length and gate loading offsets the speed advantage of higher drive current. The power dissipation trend continues to show increases in the region where the clock frequency has saturated because GENESYS increases the gate W/L ratios to optimize the critical path delay. Neither of the two cases presented in Table 2 are optimal. The small die size is low cost but low performance and the large die size is high performance but high cost. The initially steep increase in the performance indicates that there may be an optimal die size for maximizing the performance with respect to cost.

2.3.1.5 Performance-Cost Index

In extension to the work performed in section 2.3.1.4 a performance-cost index is developed to find the optimal die size for the processor of section 2.3.1.4.

Table 3: Cost analysis parameters for microprocessor cost performance index.

Parameter	Value
r	15cm
G	0.01
D	$0.05cm^{-2}$
Wfr Cost	$1,000.00

Cost Model for Processor Die

In order to obtain an index that couples the microprocessor performance with the economic cost rather than the raw die size the cost per die must be evaluated. The cost calculation begins with the assumption that the total cost for processing a complete wafer is fixed. Under this assumption the cost per die can be modeled as the wafer cost divided by the number of good die per wafer. The number of good die per wafer is the product of two related components: die size and yield. The die size determines the total number of die per wafer while the yield predicts what percentage of them are good. The raw die count and yield are defined in the following equations:

$$N = \left\lfloor \frac{\pi r^2}{A} \right\rfloor \tag{5}$$

Where N is the die count, r is the radius of the wafer, and A is the individual die area.

$$Y = (1 - G)e^{-A \cdot D} \tag{6}$$

Where Y is the yield percentage, G is the gross failure factor representing the portion of the wafer upon which all circuits fail, A is as in Eq. 5, and D is the defect density [29]. Table 3 lists the values utilized in this cost analysis. With the dependence of raw die count and yield on die area established, the cost per die is computed as:

$$DieCost = \frac{C_{wfr}}{N \cdot Y} \tag{7}$$

Where DieCost is the cost per die, C_{wfr} is the fixed wafer cost, N is the raw die count, and Y is the yield [30]. As can be clearly seen in Eq. 7 the die size is a crucial

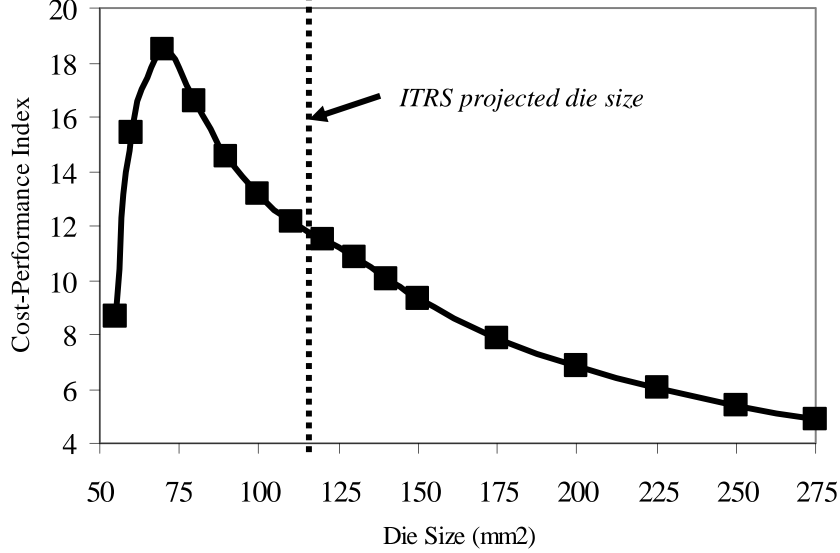

Figure 9: Performance-Cost index VS die size for 2013 ITRS microprocessor.

component of the cost per die. An increase in die size decreases both the raw die count and the yield.

Optimal Die Size for Performance Cost Index

The Performance-Cost index is formulated as the ratio of the raw throughput in billions-of-instructions per second (BIPS) to the cost per die as formulated in Eq. 7. The units of this index are BIPS per dollar ($BIPS \cdot \$^{-1}$). This index, as calculated in Eq. 8, rewards high performance and low cost systems.

$$PC_{index} = \frac{throughput}{cost_per_die} \tag{8}$$

Applying this formulation to the data from Figure 7, the PC index for the ITRS processor corresponding to the 2013 (18nm) technology generation is obtained. Figure 9 clearly shows that there is an optimal value for the PC index at a die size of approximately $70mm^2$. At die sizes below $70mm^2$ an increase in the die area results in performance increases that outstrip the resulting increase in cost. At die sizes above the optimal value the opposite is true and the performance comes at increasing cost. The optimal value for the die size as determined via the PC index is about half of

the projected ITRS die size of $140mm^2$[23].

2.3.1.6 Optimal Zone Size for Throughput

This analysis is motivated by the observation that the two components of throughput (clock frequency and CPI) compete against one another. In sections 2.3.1.4 and 2.3.1.5 the die size is varied directly and its effects on throughput are examined. The increase in throughput for the larger die sizes is due to reduced interconnect delay, the CPI remains relatively constant for those analyses. From section 2.3.1.1 we see that the CPI is primarily dependent on the gate count. In this section, the gate count rather than the die size is varied. Figure 10 illustrates the correlation between gate count and die size. The die size increases linearly for gate counts up to $5x10^6$ after which the die size becomes interconnect limited which forces the die size to grow at an increased rate as the gate count increases. The effects of increasing gate count on the throughput of a random logic block with and without memory are examined to determine if an optimal zone size (in number of gates) exists that maximizes the system throughput.

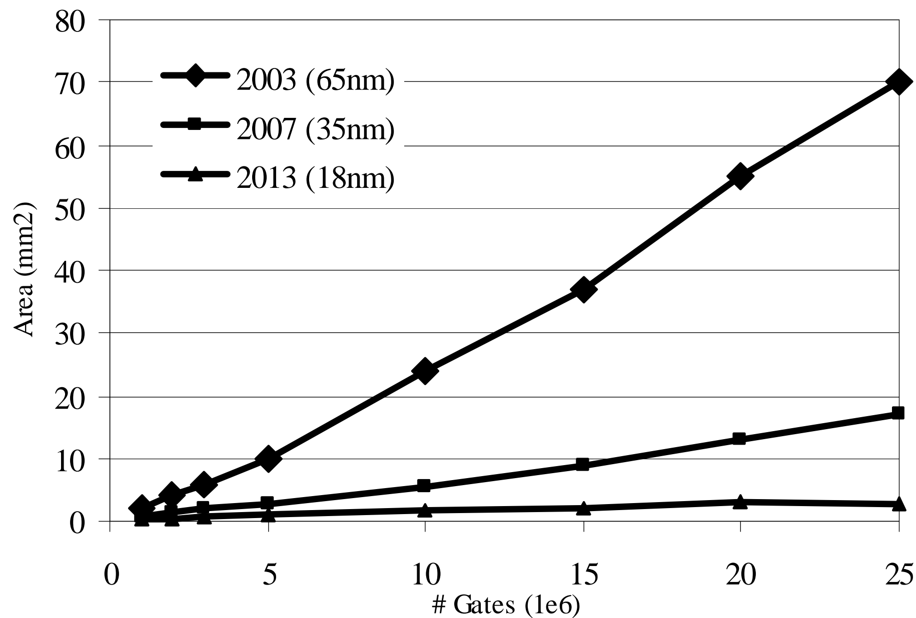

Figure 10: Minimum die size vs gate count for three ITRS technology generations.

Table 4: Parameter values for optimal zone size analysis. E_e and E_c are the exponent and coefficient for the empirical throughput relationship.

Device Width (W/L)	10.0
Logic Depth	12
Num. Caches	0
Interconnects	Opt.
Die Size	Min.
Ee	51829
Ec	-0.7725

Simulation Methodology

This analysis is run under the conditions set in Table 4 for the technology generations corresponding to the ITRS 2003 (65nm) and 2013 (18nm) years. The device width to length ratio and logic depth are held constant within each technology generation for consistency and to maintain a constant packing density. This approach ensures that the die size increase monotonically increases with respect to gate count. The interconnect architecture is optimized for maximum clock speed and the die size held to the minimum allowed by gate layout or interconnect routing concerns. The empirical CPI parameters are set to values corresponding to the Intel x86 microprocessor family. Finally, the gate count is varied from 1M to 25M gates.

Without Cache present

The effects of gate count variation on the raw throughput of a random logic block are examined by setting the number of caches present equal to zero. This eliminates the memory CPI contribution from section 2.3.1.2. This corresponds to an assumption of perfect data availability. The CPI and clock frequency variation with gate count are illustrated in Figures 11 and 12. For the 2013 generation parameters as listed in Table 5, Figures 11 and 12 show the system delay increases by a factor of three, but the CPI reduces by a factor of six. The historic improvement in the CPI due to increasing gate count overpowers the opposing increase in cycle time.

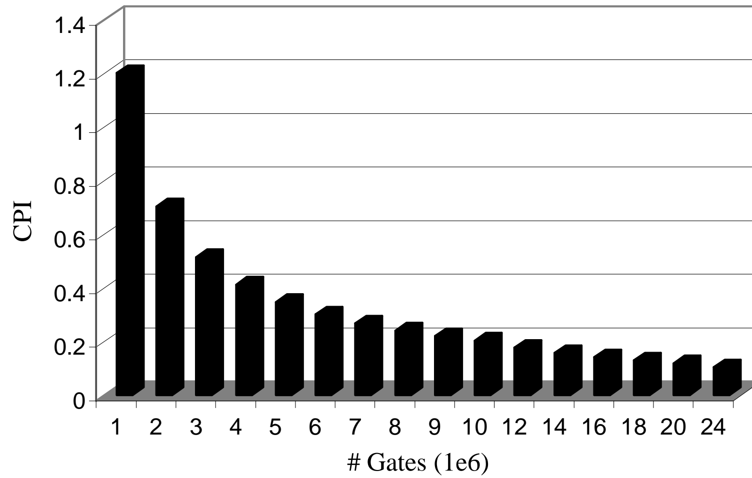

Figure 11: CPI variation with gate count for Intel x86 architecture for the empirical throughput model.

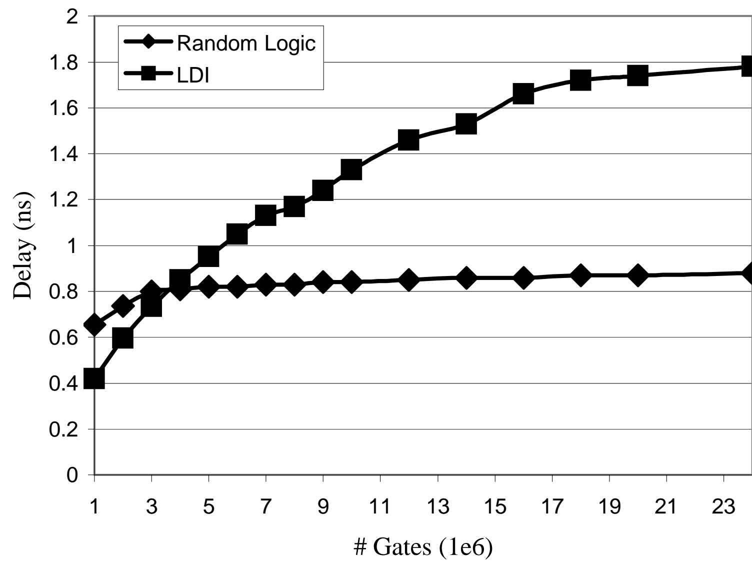

Figure 12: Clock frequency variation with gate count for Intel x86 architecture showing both the random logic (chain of gates and long distance interconnect (LDI) delay. The system delay is set by the greater of the two.

This result indicates that there is no optimal gate count/die size for throughput for any gate count between 1 and 25 million gates. If the range is extended to 250M gates the results do not change as shown in Figure 13. This behavior is due to the

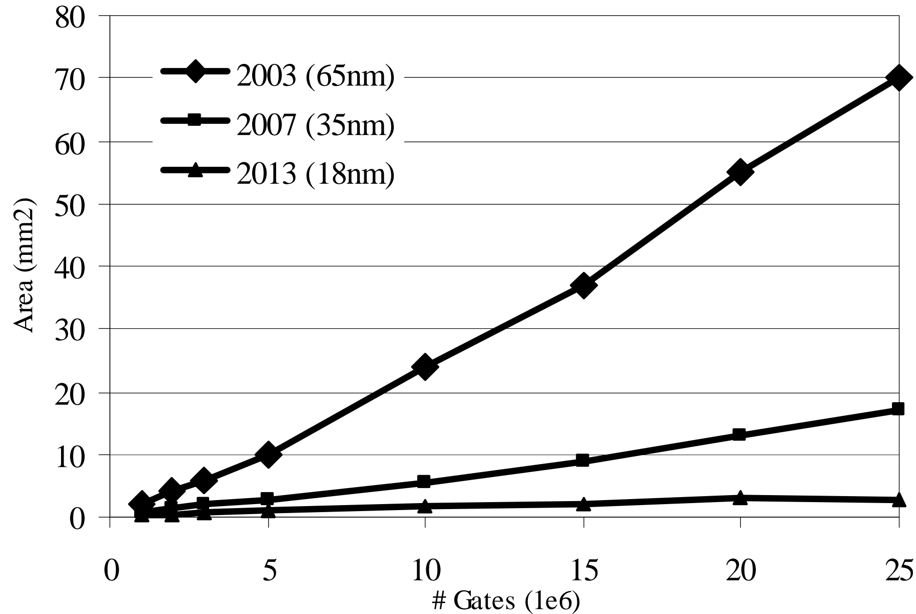

Figure 13: Raw throughput (BIPS) VS gate count for Intel x86 architecture. Ee = -0.7725.

empirical throughput coefficient, E_e. The extracted value of -0.7725 indicates that the Intel x86 architecture aggressively takes advantage of higher gate counts to reduce the CPI. Therefore the CPI trends downward at a faster rate than the cycle time increases with increasing gate count for any given technology generation.

With Cache

To more accurately gauge the dependence of throughput on the gate count it is necessary to include the contribution to the CPI from the memory hierarchy. Section 2.3.1.6 indicated that the throughput would rise monotonically with respect to increasing gate count. To account for memory CPI degradation the 2003 (65nm) technology generation was simulated for a range of gate counts from 1 to 100M gates with cache sizes of 8, 16, 32, and 64KB [31]. The results are shown in Figure 14.

Table 5: System and technology parameters for GSI microprocessor corresponding to the 2013 ITRS node.

GSI Microprocessor	
Technology	18nm
# Transistors	1.54B
# Caches	2
Threshold Voltage	0.2V
Supply Voltage	0.5V
Die Size	$140mm^2$
# Interconnect Levels	12

There appears to be an optimal zone size of approximately 25M gates for the 8, 16,

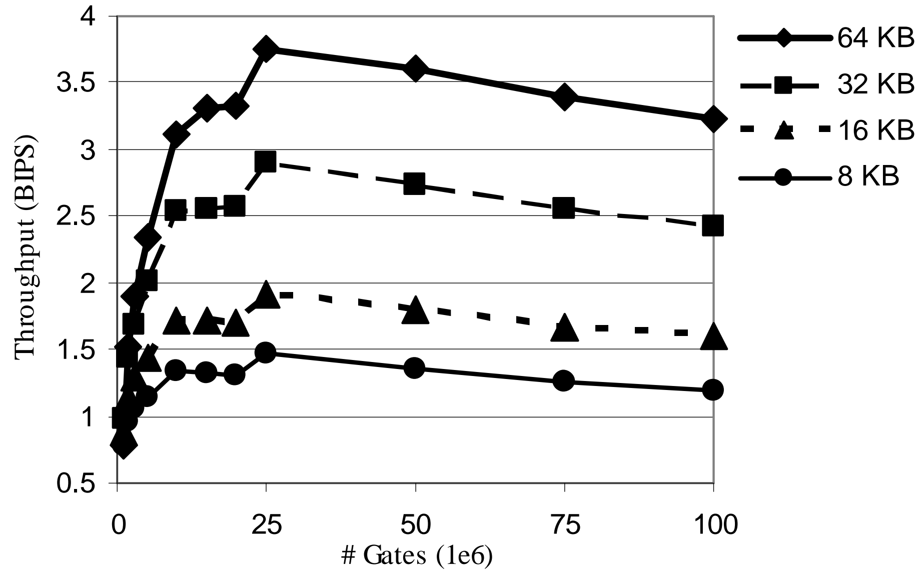

Figure 14: Throughput vs gate count for 2003 (65nm) ITRS technology generation with various cache sizes.

and 32 and 64KB cache sizes. Beyond this optimal value there is a substantial decline in performance. Too check for any technology dependence, a simulation for the 2013 ITRS generation with a 64KB cache reveals in Figure 15 similar behavior with no substantial performance increase beyond a gate count of 25M.

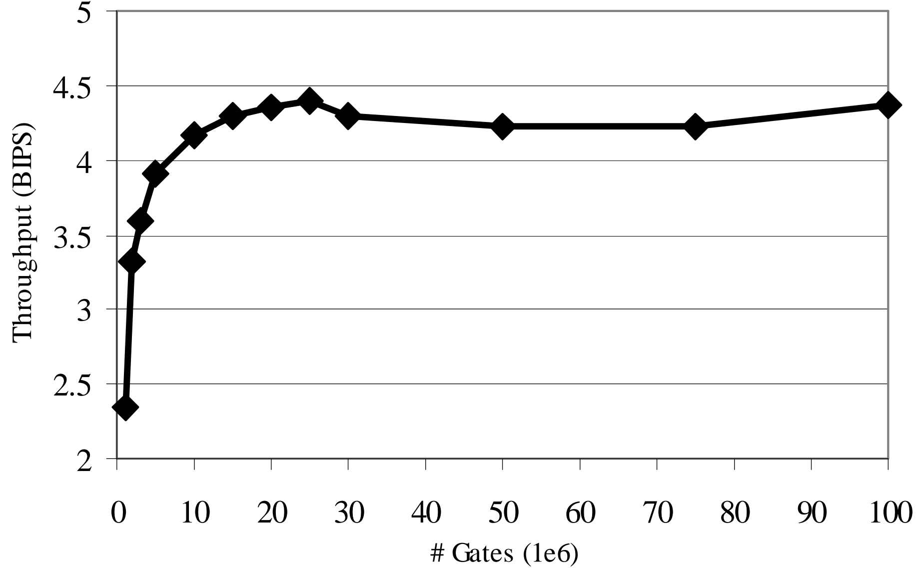

Figure 15: Throughput vs gate count for 2013 (18nm) technology generation with a 64KB cache.

2.3.2 GSI Interconnect Architectures in GENESYS

A primary constraint on the performance of future GSI systems is the increasingly restrictive limits imposed by the interconnect architecture [32]. A key feature of GENESYS is modeling of the interconnects at the system level. This section analyzes the performance and characteristics of these GSI interconnect architectures including: optimal tier organization, number of wiring levels, effects of repeater insertion, and power dissipation.

2.3.2.1 Overview of the GENESYS Interconnect Architecture

The GENESYS model of the system level interconnect architecture is based on five key observations/assumptions:

- GENESYS assumes that the length distribution of the interconnect architecture obeys the stochastic interconnect distribution derived in[9].
- An interconnect level represents a single physical plane on which a metal layer is routed. An interconnect tier represents a grouping of such levels on a logical

basis. For example, an architecture with 6 metal levels may be grouped into a many as 3 tiers (two orthogonal levels per tier) or as few as one tier (6 levels). GENESYS constrains all interconnect levels in a tier to the same pitch and are adjacent to one another.

- A key assumption of GENESYS is that the level on which an interconnect is routed is determined by the length of the interconnect. The shortest interconnects are routed on the lowest level/tier while the longest interconnects are routed on the uppermost level/tier. This approach is based upon the use of reverse scaled interconnects in VLSI systems in which the longer interconnects are routed on the upper tiers with larger cross-sectional dimensions [33].
- GENESYS utilizes a supply and demand algorithm for determining routing constraints for each interconnect tier.
- The longest interconnect length (global) is assumed to be approximately twice the chip edge length.

2.3.2.2 Optimal Tier Organization for GSI Systems

The tier organization of the interconnect architecture is an important optimization constraint in GENESYS. A single tier wiring scheme represents a simple non-reverse scaled architecture with little opportunity for optimizing wiring dimensions due to supply limitations. A multi-tiered system permits GENESYS to optimize by sizing the tiers and finding the optimal partition lengths by mapping the on-chip supply to the stochastic distribution. Here the effects of tier organization on an optimized GSI interconnect architecture are examined.

Simulation Methodology

GENESYS simulations for the GSI microprocessor summarized in Table 5 were conducted for maximum performance at a die size of $140mm^2$ with 12 levels of

interconnect. Optimal repeater insertion is assumed. The key results of interest were the throughput and cross-sectional areas of the interconnects in the uppermost tier. Four different tier organizations were considered:

1. **1 tier:** All interconnects have the same cross-sectional dimensions.

2. **2 tiers:** The first six, second six levels occupy the first, and second tiers respectively.

3. **3 tiers:** The four lowest levels occupy the first tier with the four mid levels and four upper levels occupying the second and third tiers.

4. **6 tiers:** Each orthogonal pair occupies a separate tier.

Simulation Results

Figures 16 and 17 show the system throughput and interconnect cross-sectional area for each tier organization. As can be seen in Figure 16 the full-optimization case (6 tiers) maximizes the performance. The worst case (1 tier) performance is the result of the small cross-sectional interconnect area as illustrated in comparison with the other tier organizations in Figure 17. For the 1 tier case the RC delay dominates due to high interconnect resistance. The 3 and 4 tier organizations show substantial improvement, but are still less than optimal. Figures 16 and 17 indicate that the majority of the performance increase for the fully optimized case comes from increased sizing of the global interconnects. Figure 17 shows a substantial 2.5x increase in the cross-sectional area from case three to four. The conclusion drawn from these results is that tier organizations emphasizing the optimization of all orthogonal pairs and global interconnects (upper tiers) in particular result in the greatest performance gains. If the assumption of optimal repeater insertion is relaxed, the same trend is observed, however, the throughput is greatly reduced due to increased interconnect delay.

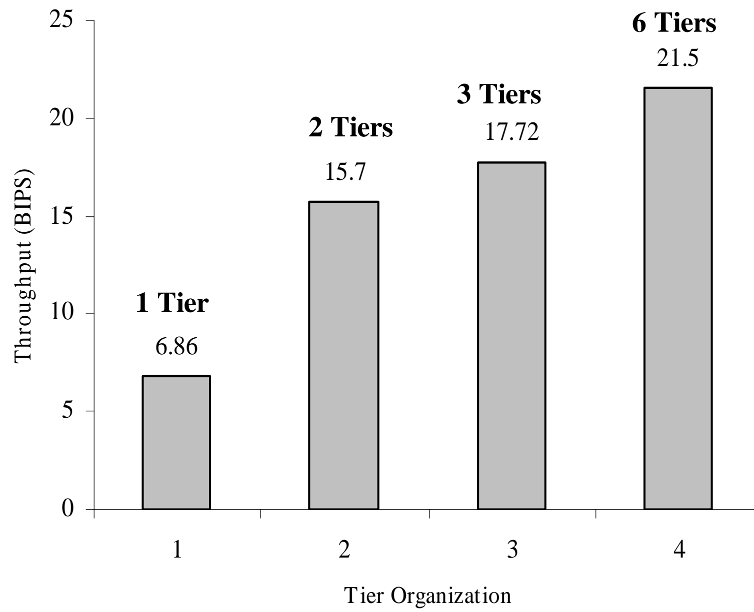

Figure 16: Throughput for each tier organization is listed in section 2.3.2.2.

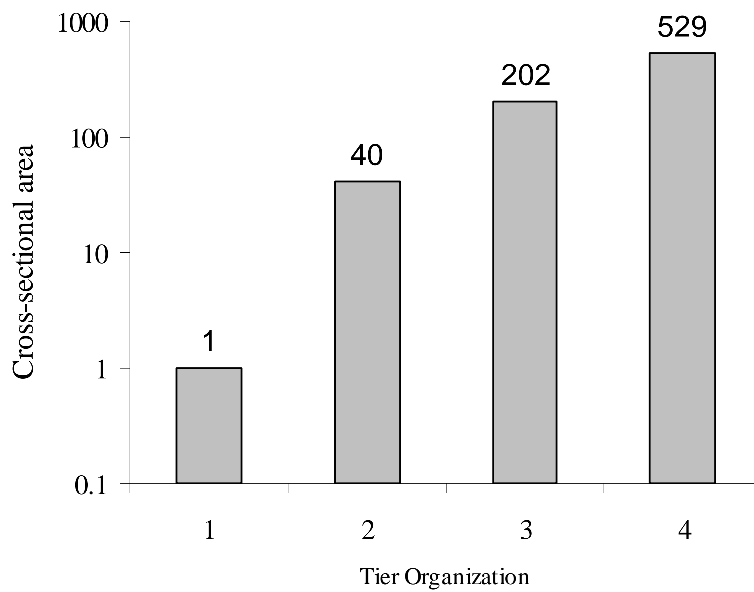

Figure 17: Normalized global interconnect cross-sectional area for each tier organization is listed in section 2.3.2.2.

2.3.2.3 Performance Dependence on Number of Int. Levels

Section 2.3.2.2 examined the relationship between tier organization (focus of optimization) and performance. Another equally important design variable influencing the performance of GSI systems is the number of interconnect levels available. The maximum cross-sectional area of the global interconnects is heavily dependent on the available on-chip supply. Increased supply allows for heavier interconnect pitch at the global levels. The interconnect supply, or the maximum interconnect length available for routing purposes, is estimated by:

$$Supply = \frac{N_w \cdot A_{chip}}{p_w} \cdot \eta_w \tag{9}$$

Where N_w is the number of wiring levels, A_{chip} is the chip area/die size, p_w is the wiring pitch, and η_w is the wiring efficiency. From Eq. 9, the on-chip supply is directly proportional to the number of interconnect levels. From Eq. 1, the system throughput increases linearly with clock frequency for any given value of the CPI. The CPI is held constant for this analysis, therefore, the throughput represents an indirect measure of the clock frequency. Figure 18 shows that the throughput increases from less than three BIPS for two levels of wiring to more than twenty BIPS with twelve wiring levels. For the ITRS projection of twelve interconnect levels by 2013 the projected system performance is approximately twenty-two BIPS.

2.3.2.4 Effect of Repeater Insertion on System performance

One of the primary approaches to optimizing the performance of long distance interconnects (LDI) is the insertion of repeaters to linearize the RC component of the delay. This section explores the performance benefits of repeater insertion in a multilevel GENESYS interconnect architecture. The device technology and system parameters utilized in this analysis are listed in Table 6.

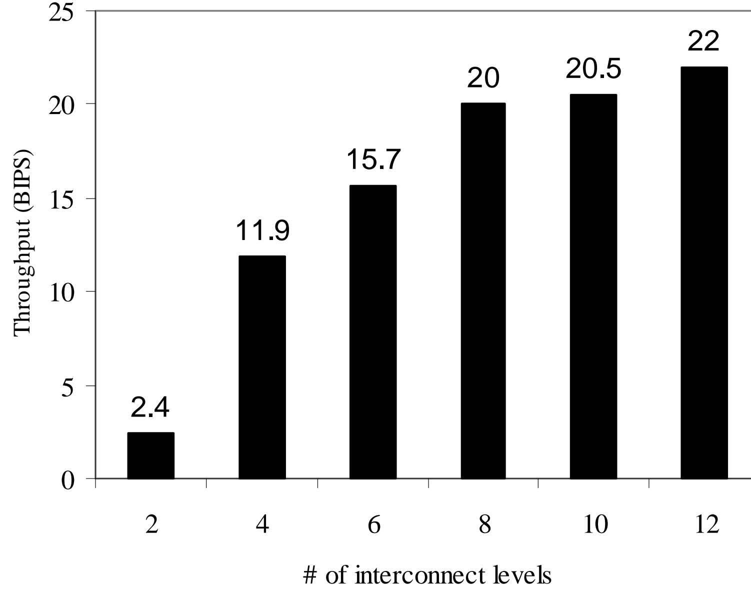

Figure 18: System performance in BIPS vs the number of interconnect levels for a GSI microprocessor.

Comparison at Common Die Size

The performance of repeater driven and single driver LDIs is compared at a common die size. The point of comparison is the die size for which a system with repeater driven LDIs achieves a clock frequency of 1 GHz. This die size is held constant with the only variable being the insertion of repeaters. Given the system parameters listed in Table 6 the minimum die size for maintaining the target clock frequency of 1GHz for repeater driven interconnects is approximately $85mm^2$. The results listed in Table 7 show that a single driver scheme for LDIs results in a 97% decline in the clock frequency. For this example, the insertion of repeaters results in a more than 30x increase in the clock frequency.

Comparison for Large Die Sizes

A comparison of the single driver and repeater LDI schemes at larger die sizes makes the necessity of repeater insertion plainly clear. Again, the results are shown for a common die size. This time the ITRS projected die sizes of $140mm^2$ (production)

Table 6: Technology and system simulation parameters for GENESYS repeater analysis.

Technology & System Parameters	
Device	
Feature Size (nm)	18
Oxide Thickness (nm)	0.5
Threshold (V)	0.2
Supply V	0.5
System	
Xtrs (10^6)	1546
Num Caches	0
Int. Levels	12
Int. Tiers	6
Target Die Size	$85mm^2$

Table 7: Repeater and single driver LDI performance for minimum die size for a 1GHz target clock frequency (repeater).

Repeater/Single Driver Comparison		
	Repeater	Single Driver
$Fclk_{max}$ (MHz)	1000	30
Die Size (mm^2)	85	85
Global Delay (ns)	0.88	29.0

and $280mm^2$ (introduction). In comparison with the data in Table 6, Figure 19 shows that the performance of the repeater driven interconnects increases by 110% while the single driver performance logs a 400% gain if the die size is increased to the ITRS projected $140mm^2$. If the die size is further increased to $280mm^2$, the performance of the repeater driven LDI increases by 140% and the single driver case by more than 650%. Despite this relatively large % increase in the single driver clock frequency, the repeater driven interconnects remains an order of magnitude greater.

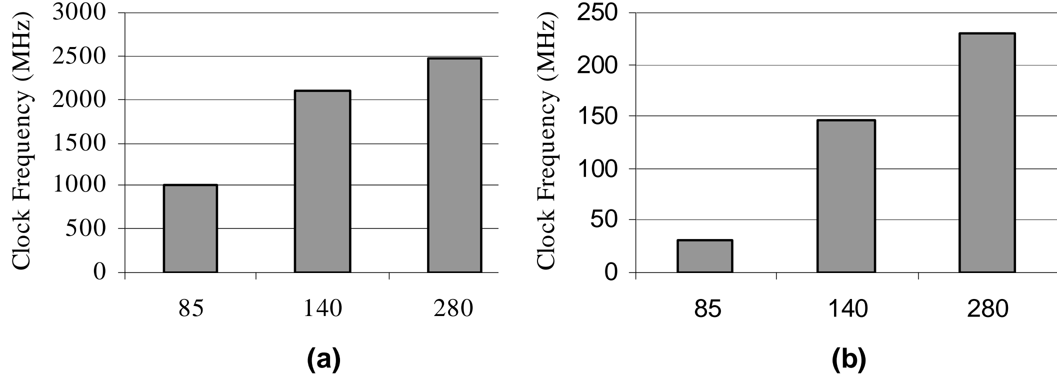

Figure 19: Clock frequency for (a) repeater driven and (b) single driver LDIs at 85, 140, and $280mm^2$ die sizes.

Comparison at Common Clock Frequency

Comparison at common die sizes illustrates the performance advantage of repeater insertion. Another advantage of repeater insertion is highlighted by comparison at common clock frequencies. This relaxes the earlier assumption of constant die size and allows interconnect dimensions to vary. For the processor implementation utilized for the above analysis there is no overlap in performance; the minimum clock frequency for the repeater driven example is 253MHz while the maximum frequency for the single driver case is less than 230MHz. However, Figure 20 for 2001 ITRS generation (90nm) shows significant (horizontal) overlap in the range of allowable operating frequencies. At every level of performance from 500 to 1GHz the repeater driven system has a die size between 40 to 60% than that of the single driver system. Therefore, over a range of die sizes, the repeaters can provide either smaller die size for equal performance (if the design is interconnect limited) or higher performance for equal die size.

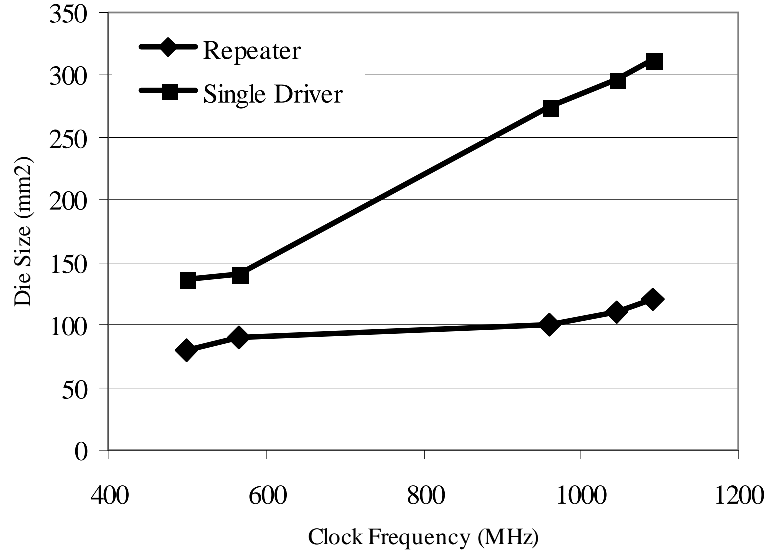

Figure 20: Die size comparison of repeater and single driver LDI schemes at common frequency.

2.4 Extending GENESYS to Heterogeneous SoCs

As can be clearly seen from the above examples, GENESYS is a flexible and powerful tool for exploring system level limits and the interaction between technology and architecture in determining potential system performance. However, even with the depth of modeling at every level of the GSI hierarchy and the novel throughput modeling, the uni-processor system model is inadequate for accurately modeling emerging SoC design methodologies. In the following chapter, a key advancement of the GENESYS modeling framework is discussed in detail.

CHAPTER III

HETEROGENEOUS SOC METHODOLOGY

3.1 Introduction

GENESYS 2004 is designed to project future trends and performance metrics for gigascale systems by applying a core set of analytical and physical models derived from first principles and established empirical knowledge to a set of input parameters spanning all levels of the GSI hierarchy [7][8][21]. Through this set of models and parameters GENESYS assimilates the structure of the GSI hierarchy and assesses key performance metrics such as die size, power dissipation, and clock frequency. An earlier system level model utilized in GENESYS assumed a homogeneous uniprocessor architecture consisting of a large core of random logic and on-chip cache as illustrated in Figure 21a. A key feature of the homogeneous assumption for a block of random logic is that the interconnect wire length distribution obeys a stochastic model [9][10]. This allowed for the estimation of clock frequency and power based on an average wire length model derived from [9]. In relation to SoC designs, this model has several drawbacks. First, SoCs are not sufficiently modeled as homogeneous systems. The emerging dominance of IP core methodologies for SoC design results in inherently heterogeneous systems. This is an important consideration when evaluating the wiring resources. The global wire-length distribution for a cluster of heterogeneous megacells (cores) obeys a different distribution [34]. Second, the homogeneous assumption 'flattens' the system eliminating all micro-architectural detail and allowing only the

calculation of average values for metrics such as power dissipation that may experience significant variation across the chip (i.e. hot spots). Therefore, a new model for describing and analyzing heterogeneous SoCs via GENESYS 2004 is developed. Figure 21 below provides comparison between the homogeneous system model and an actual chip implementation [35].

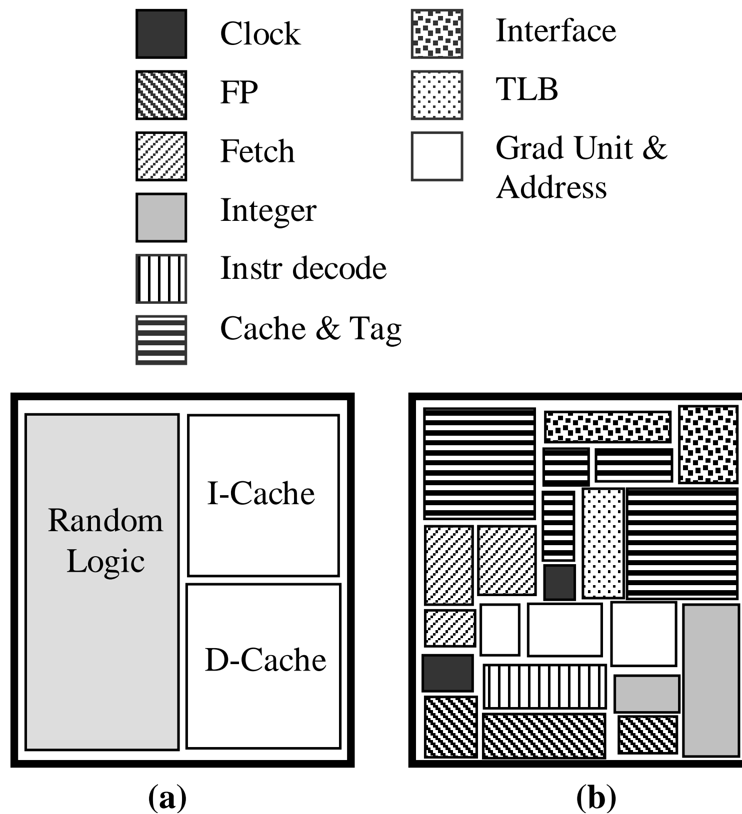

Figure 21: Part (a) homogeneous system description in GENESYS with little micro-architectural detail and part (b) a commercial chip implementation with 20 megacells on-chip from [35].

3.2 Block Modeling Methodology

Many SoC systems are designed from pre-defined megacells in order to simplify and speed up the design process. The choice of a block-based modeling scheme is natural. Figure 21b illustrates a heterogeneous block diagram. The modeling of an

SoC via GENESYS begins with the definition of the global/technology parameters that provide a listing of all GENESYS inputs for use as default values. Any of these parameters may be overridden, but most of them such as feature size, oxide thickness...etc. will not change from cell to cell. These GENESYS inputs are descriptive of the technology resources allocated to the system. The next step is the formulation of a system description for the SoC as a collection of megacells. Each megacell is defined by a set of inputs and/or instances of other cells. This is where the hierarchical nature of the block-based methodology takes shape. A simple example definition of a block looks similar to this:

```
    Entity block B1 is
    num_transistors = 1.5e6;
    logic_depth = 20;
    num_wiring_levels = 6;
    device_width = 10;
    block A1 a1;
    block A2 a2:(logic_depth = 15, device_width = 15);
end Entity B1;
```

The above entry defines a block of random logic containing 1.5M transistors, 6 levels of wiring, an average device W/L ratio of 10, and logic depth (number of gate delays between latches) of 20. Additionally, the block B1 contains two instances of other blocks (A1 and A2). The instance a1 of block A1 is exactly as defined elsewhere in the SoC description, but instance a2 overrides several of the parameters inherited from parent A2 with new values. A1 and A2 are referred to as the parent cells for instances a1 and a2. In this manner the SoC may be composed of a large number of smaller cells each with its own set of GENESYS parameters. The hierarchical

structure of the machine description can be utilized to describe the on-chip resources of the SoC with varying degrees of granularity.

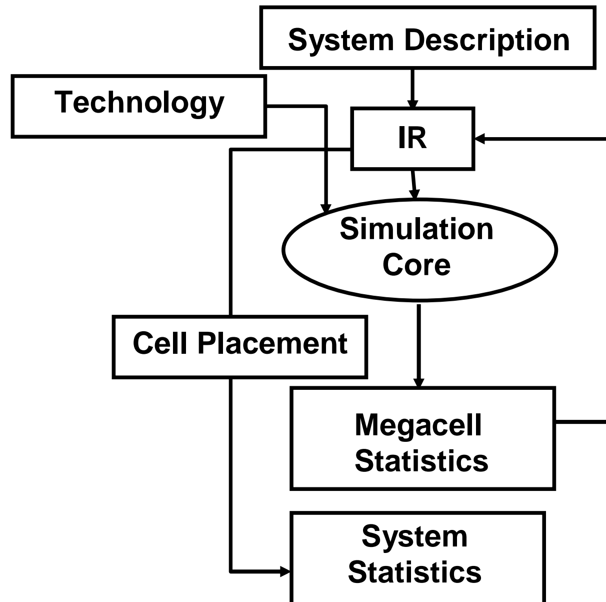

Figure 22: Block diagram of the simulation flow for the GENESYS 2003 heterogeneous methodology. IR is internal representation.

Figure 22 illustrates the basic program flow of a simulation run, beginning with the system description. Unlike the global and technology parameter files, which are simply listings of parameter names and values, the SoC description has an associated grammar. This requires a front-end compiler for reading the SoC specification and generating an internal representation (IR) of the SoC structure for use in GENESYS. The IR maintains the hierarchical structure as listed in the specification. Each cell defined is a separate object in the IR. The cell object contains all of the parameters associated with that cell, a list of any other cells that are instantiated within, and the simulation results for the cell. Because the entire specification is read into the IR before any simulation steps occur, all instances are linked to their parent cells before running the GENESYS core. This removes restriction on the ordering of cell entries

in the machine specification file.

Once the IR is constructed, GENESYS 2004 loads the global input parameters then queries the first cell object for its input parameters. GENESYS then runs a simulation for the individual cell as a homogeneous system utilizing its core set of material, device, and circuit models and the stochastic interconnect distribution from [9][10]. On the first simulation pass all cells with instances of other cells within them are skipped. Once all parent cells are simulated, a second run generates simulation results for any instance that contains overridden parameters. Instances without override parameters assume the simulation results from the parent cell.

When the initial simulation runs are complete, GENESYS compiles the simulation results for any composite cells (cells containing instances of other cells). The performance statistics for the three primary metrics (area, speed, and power) are determined via the following expressions. The cell area is simply the sum of the areas for each of its constituent cells:

$$A_{cell} = \sum_{i \leq n} A_i \tag{10}$$

Where A_{cell} is the cell area and A_i is the contribution from the ith constituent cell. The clock frequency is assumed to be limited by the slowest constituent cell such that:

$$f_{cell} = min(f_i) \tag{11}$$

Where f_{cell} is the maximum clock frequency for the composite cell and f_i is the clock frequency of the ith cell. The power dissipation in the composite cell must be adjusted to account for differences in the maximum clock frequency between cells:

$$P_{cell} = \sum_{i \leq n} P_i \frac{f_{cell}}{f_i} \tag{12}$$

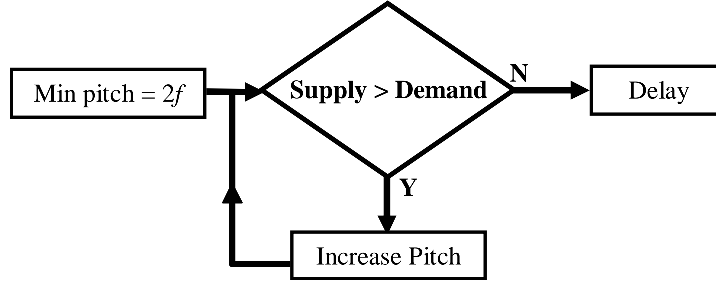

Figure 23: Supply and Demand sizing algorithm for global interconnects

Where P_i is the power dissipation of cell i operating at frequency f_i and f_{cell} is the frequency of the composite cell.

The final step in evaluating the performance of the SoC is the construction of the heterogeneous global wiring distribution. The number of gates and Rents parameters in the megacell combined with placement and routing information as given in [34] allows GENESYS to calculate the global wire length distribution for the design. A key property of the heterogeneous methodology is the consideration of cell placement. GENESYS 2004 takes cell placement information and calculates the average cell placement efficiency for the system. This heterogeneous distribution (demand) is then mapped to the global wiring resources (supply). The dimensions of the global wires are determined via a simplified supply/demand optimization routine as illustrated in Figure 23. The clock frequency for a SoC under the assumption of a single global clock is determined via the maximum delay of either the random logic or the global interconnect (whichever is larger).

The relevant output statistics at this stage include power, chip area, maximum clock frequency, and numerous others regarding technology related parameters and the interconnect architecture. Because the simulation results are available for every level of the SoC specification, inhomogeneities in maximum clock frequencies

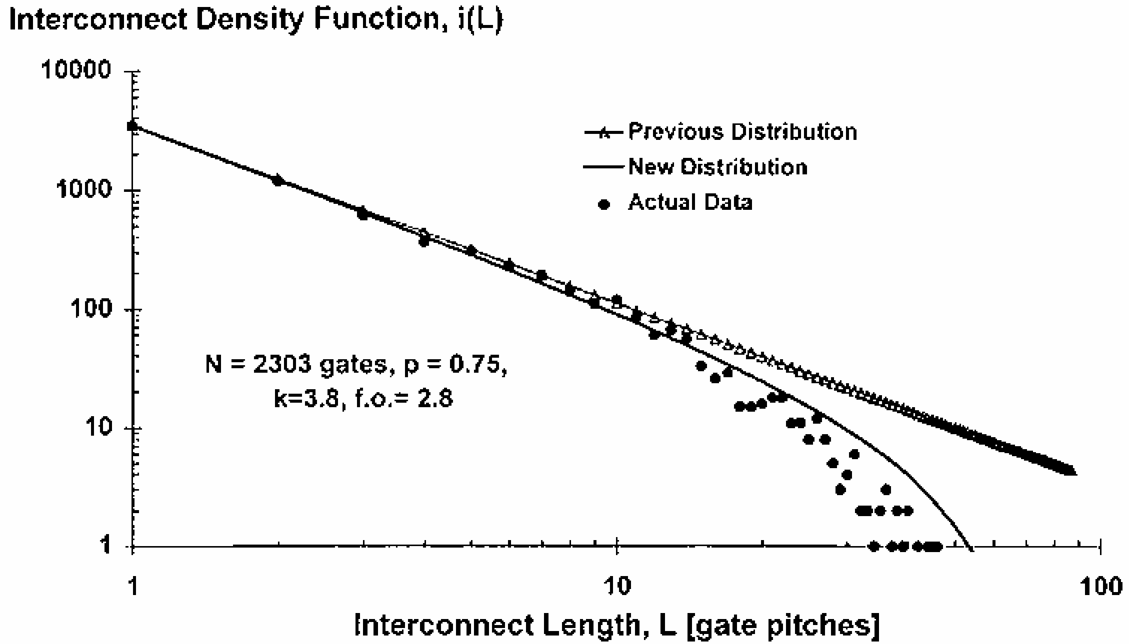

Figure 24: Interconnect density function derived under assumption of a homogeneous gate array. Expected number of interconnects vs. length in gate pitches [9].

and power dissipation can now be addressed. This represents an important advance of the previous homogeneous modeling approach.

3.3 Global Interconnect Methodology in GENESYS 2k4

Under the homogeneous uni-processor model originally advanced in GENESYS the interconnects are expected to obey the stochastic distribution derived by Davis [9]. This assumption is good for relatively small blocks of random logic in which the assumption of homogeneity holds approximately true. The properties of this distribution are illustrated in Figure 24.

This distribution has proven useful in estimating routing requirements for projecting system performance, however, for large systems constructed from groupings

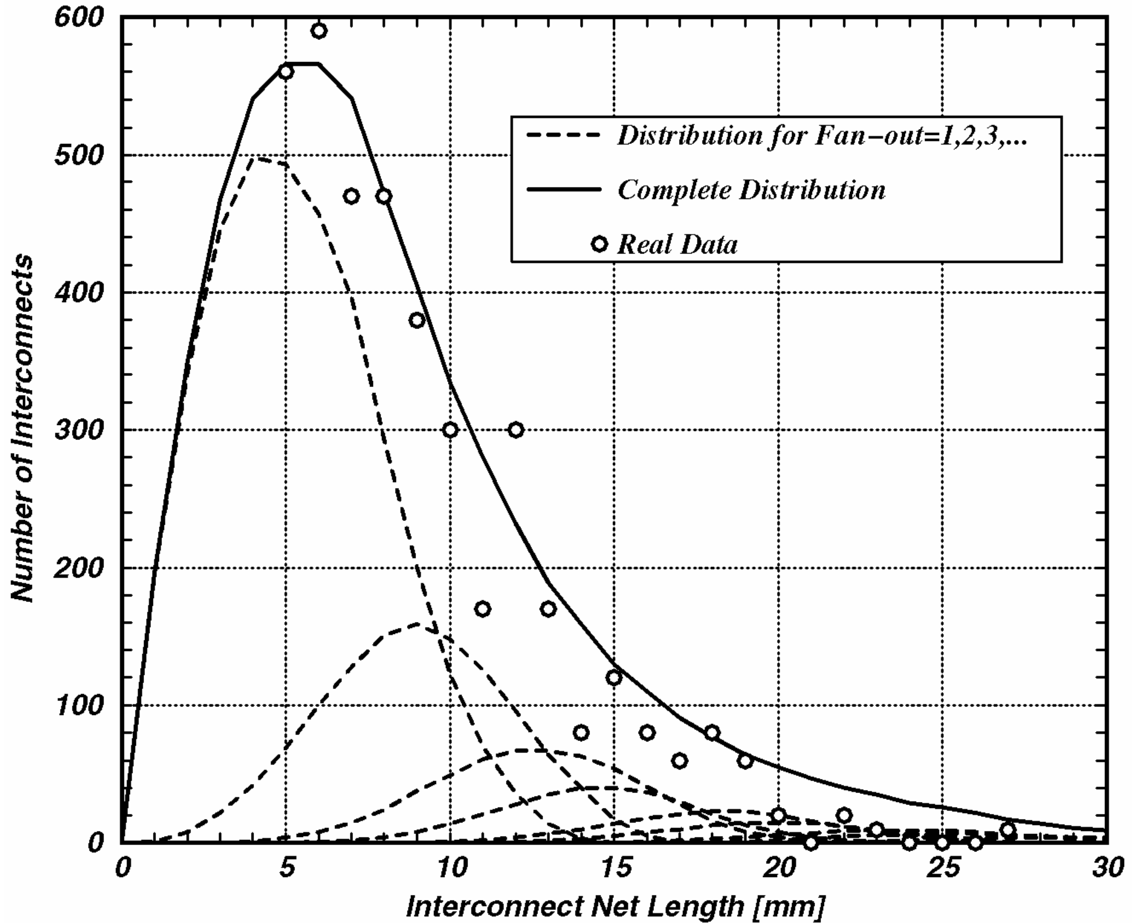

Figure 25: Interconnect net length distribution: Expected number of interconnects vs. physical wire length [34].

of heterogeneous megacells such as the system in Figure 21b the homogeneous assumption no longer holds true. The most promising solution for estimating system-level routing requirements for SoCs is a previously derived stochastic distribution for multi-terminal global nets [34]. The characteristics of this distribution are illustrated in Figure 25.

The homogeneous distribution is dominated by short interconnects, but still predicts large numbers of global interconnects up to the maximum of two chip-edge lengths. The heterogeneous distribution on the other hand is dominated by mid-length global nets with very few short or long nets.

The block modeling methodology for heterogeneous SoCs takes advantage of both distributions to improve the accuracy of system performance projections. The homogeneous distribution is assumed to apply only to interconnects within cells (local) interconnects and the heterogeneous distribution is reserved for inter cell nets which are designated to be global.

The local distribution can be represented by a closed-form expression allowing for relative ease in integration to calculate total or average interconnect length. The global distribution, however, requires the calculation of routing requirements for multiple fanouts and the determination of the global placement efficiency. The calculation of the net distribution (number of expected nets for each fanout from 2 to N_{meg} (number of megacells) is relatively straightforward, but the global placement efficiency requires megacell placement and global net list information. The cell placement information can be reasonably approximated apriori but the net list information is not assumed to be available. The dependence of the global interconnect length on the cell placement efficiency demands that a suitable algorithm be developed for estimating the efficiency in the absence of global net list information. From [34] the average megacell placement efficiency (η_p) is computed via the following expression:

$$\overline{\eta_p} = \frac{1}{N_{net}} \sum_{i \in Netlist} \frac{1 - \frac{Bounding_Area}{Total_Area}}{1 - \frac{m_i}{N_m}} \tag{13}$$

where N_{net} is the total number of global nets, the bounding area is the area of a box drawn about the cells connected by a net, the total area is the sum of the individual cell area, m_i is the net fanout, and N_m is the number of megacells in the system.

The average placement efficiency is the weighted average of the placement efficiencies of the individual nets. The resulting placement efficiency influences the bounding area of a net in the following manner:

$$\hat{a} = \hat{b} = \sqrt{\overline{A_{meg}}(m\eta_p + N_m(1 - \eta_p))} \tag{14}$$

$\overline{A_{meg}}$ is the average megacell area, and m is the fanout of the net. The calculation of the average net length for a given fanout is a function of the net bounding area computed above and empirically determined parameters:

$$L_{ave} = (\alpha m^{\gamma} - \beta)\frac{\hat{a}\hat{b}}{\hat{a} + \hat{b}} + (\hat{a} + \hat{b}) \tag{15}$$

Where $\alpha = 1.1$, $\beta = 2.0$, and $\gamma = 0.5$.

GENESYS 2004 handles the calculation of the average global placement efficiency by generating a random cluster of nets for each fanout and calculating the individual placement efficiencies for each net in the cluster. This placement efficiency is then averaged and weighted by the total number of nets of that fanout. Once the weighted efficiencies are known for all fanouts, the average placement efficiency is calculated as follows:

$$\overline{\eta_p} = \frac{1}{N_{net}} \sum_i N_{neti}\eta_{pi} \tag{16}$$

Where N_{neti} is the number of nets of fanout i, and η_{pi} is the average efficiency for fanout i. Once this calculation is complete the derivation of the complete stochastic global distribution is enabled by calculating the average bounding box dimensions for each fanout as in Equation 14. Next, GENESYS 2004 computes the average net

length from Eq. 15. The total global interconnect demand is expressed in Equation 17.

$$L_{total} = \sum_i N_{neti} L_{avei} \tag{17}$$

This procedure allows GENESYS 2004 to accurately gauge the required global routing resources for an arbitrary heterogeneous collection of cells with nothing more than the cell placement information. With knowledge of the routing requirements estimates of global interconnect dimensions, parasitics, energy consumption, and delay become possible. The key feature of this algorithm is that apriori knowledge of the net list information is not necessary.

3.4 Cell Placement

One of the key design considerations in core based SoCs is the physical placement of the megacells. The placement efficiency metric discussed in section 3.3 requires information regarding the physical layout of the SoC. With this information, GENESYS 2004 can compute the placement efficiency for estimating the required global interconnect resources. GENESYS 2004 provides several methods for specifying the cell placement: manual and automated.

3.4.1 Manual Cell Placement

The manual cell placement for an existing SoC implementation is constructed by superimposing a sufficiently fine grid over a floorplan/die micrograph and recording the upper-left and lower-right coordinates of the corners of each cell in the design. The grid is numbered from its lower-left corner [0,0] to its upper-right [N,N], where N is the grid factor indicating the number of rows/columns. The grid should be chosen

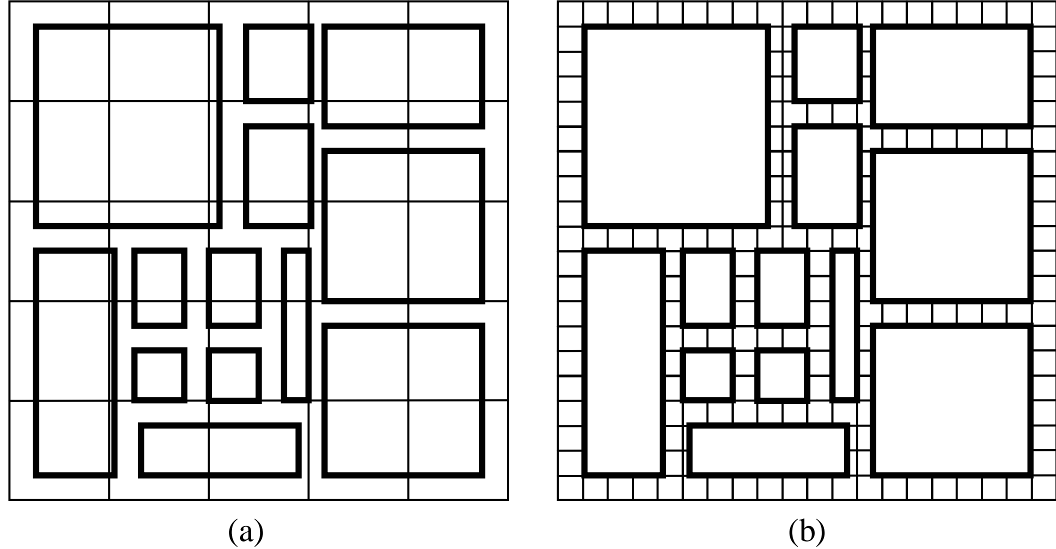

Figure 26: Example of grid choice for manual cell placement: (a) an grid value of 5 is insufficient to accurately locate the cell corners, (b) a grid factor of 20 provides a much closer fit to the SoC floorplan allowing for an accurate assignment of cell-corner coordinates.

sufficiently fine that the coordinates for the cell corners are distinct and representative of the position and dimensions of the actual SoC layout. Figures 26a and b illustrate the grid placement for N=5 and N=20 grids. From inspection, the N=5 grid of Figure 26a is too coarse to place all the cells in their proper positions with respect to one another or to maintain the proper cell dimensions. The N=20 grid, however, is fine enough to properly specify the physical layout of the SoC with acceptable accuracy. In general, any grid factor of N=20 or greater is sufficient for an accurate placement specification.

An example placement specification is illustrated in Figures 27a and b. The format utilized to specify the cell coordinates is: **cell_name (upper,left:lower,right)**. GENESYS 2004 reads the specification in Figure 27b and constructs an internal list which is compared against the list of megacells comprising the system entity. With the specification in place GENESYS utilizes the information to calculate the average cell placement efficiency as described in section 3.3.

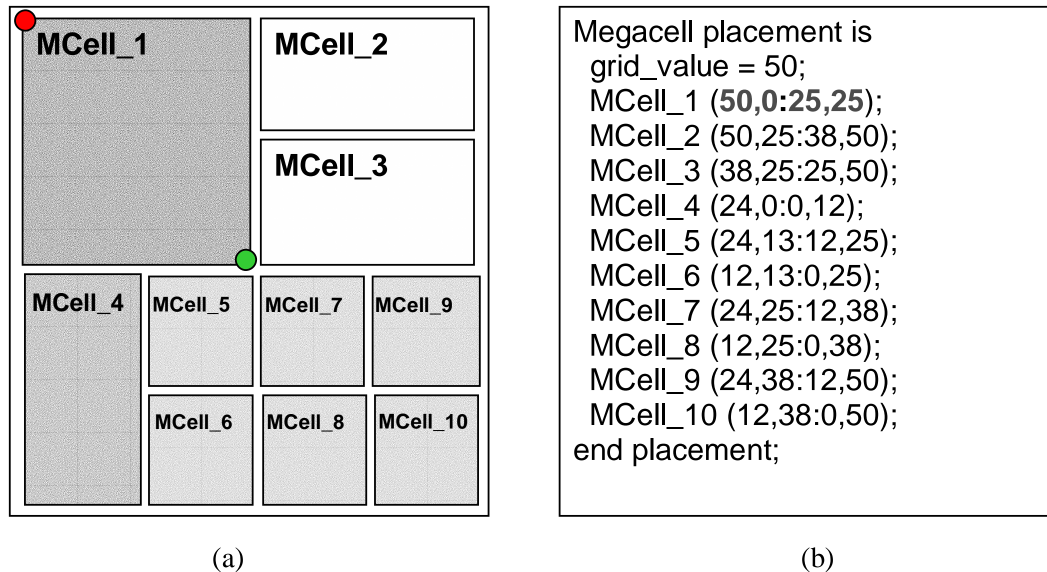

Figure 27: Cell placement specification: (a) layout with cell Mcell_1 corners marked, (b) GENESYS 2004 megacell placement specification for (a) with a grid factor of N=50.

3.4.2 Automated cell placement

The alternative to manually specifying the coordinates of each cell in an SoC design is to utilize the auto-placement algorithm to generate an approximate SoC floorplan. The primary inputs to the auto-placement algorithm is the grid factor (as defined in section 3.4.1) and a placement constraint. The placement constraint allows the user to specify several different criteria for creating the SoC floorplan. The available placement criteria are listed below:

- **in order**
- **cell size**
- **bus length optimization**

The in-order constraint forces GENESYS 2004 to place the cells in the same order that the cell instances are listed in the system level entity of the block SoC description. The resulting floorplan is organized from left to right and top to bottom.

An SoC with M megacells would have cell_1 occupying the upper left corner of the chip and cell_N in the lower right corner. This constraint gives the user the ability to fine-tune the placement by altering the ordering without having to manually alter cell coordinates. The cell-size constraint orders the cells from smallest to largest. The goal of size based cell ordering is to cluster small cells together to increase the effective placement density and reduce interconnect length for connections within the cluster. The bus length optimization constraint is used in conjunction with the bus specification as described in Chapter 4. The goal of this constraint is to minimize the physical length of the bus by placing all cells connected by the bus in adjacent positions. Simulation results on the effectiveness of this technique are given in Chapter 4.

The algorithm for generating the individual cell coordinates is illustrated in Figure 28. Once the simulations for each block in the system description are completed the auto-placement algorithm accesses the system level entity and retrieves the cell names and area. Because no information regarding the geometry of the cells is given, the aspect ratio of each cell is initially assumed to be unity. The height and width of each cell are calculated and stored. Next, the placement constraint is applied and the ordering of the cells is adjusted. Once the cells are ordered for placement, the required number of cell rows is estimated via the following expression:

$$N_{row} = \frac{\sum_{i=1}^{N_{cell}} Wcell_i}{\sqrt{A_{chip}}} \tag{18}$$

where N_{row} is the number of rows that cells will be grouped into, N_{cell} is the total number of megacells present in the SoC, $Wcell_i$ is the width of cell i, and A_{chip} is the total chip area. Once the row count is known, the cells are grouped and tiled across until the row is filled and the next row is assembled beginning with the next cell in

the placement list. The newly assembled rows contain cells with varying dimensions as illustrated in Figure 29. After the rows are assembled a vertical squeeze operation is performed to equalize the height of the cells by compressing the larger cells to equal height with the smallest cell. Because the area of each cell must remain constant, the aspect ratios of the larger cells are altered to compensate. The vertically squeezed row may now have a width that is substantially longer than the width of the chip. To return the row to its proper width an horizontal squeeze operation is performed to reduce the width of the cells. Again, because the area must remain constant, the aspect ratios of the cells increase as the cell height increases. This second squeeze operation reduces the row width to match the width of the chip. If the row width is less than the chip width, it is stretched to match the chip width. The final step is to stack the rows and assign the corner coordinates. The upper cell coordinates for the first (upper) row is assigned the maximum value of N (grid value) and the first cell in each row is given a left coordinate of 0. The remaining coordinates are assigned in the following manner:

$$Upper_i = Upper_{i-1} - \lfloor \frac{H_{i-1}}{d_{grid}} \rfloor \tag{19}$$

$$Lower_i = Upper_i - \lfloor \frac{H_i}{d_{grid}} \rfloor \tag{20}$$

$$Left_j = Left_{j-1} + \lfloor \frac{W_{j-1}}{d_{grid}} \rfloor \tag{21}$$

$$Right_j = Left_j + \lfloor \frac{W_j}{d_{grid}} \rfloor \tag{22}$$

Where $Upper_i$, $Lower_i$, $Left_j$, $Right_j$ are the cell coordinates for row i and cell j respectively. The value d_{grid} is the grid dimension used to convert the physical height and width of the cells (in units of mm) to grid units. It is calculated via the following expression:

$$d_{grid} = \sqrt{\frac{A_{chip}}{N^2}} \tag{23}$$

This methodology is intended to produce an approximate floorplan for estimating the placement efficiency in lieu of empirical cell placement data. In typical SoC designs the individual macrocells are optimized and placed according to specific design criteria. The auto-placement routine does not preserve cell geometry and (depending on the placement constraint and cell count/area) will typically produce suboptimal efficiencies. If information regarding cell geometries is available, the manual cell placement technique is utilized to explicitly define the SoC floorplan for calculating the placement efficiency with a higher degree of accuracy.

Once all of the cell coordinates have been defined, GENESYS 2004 is ready to generate the average cell placement efficiency. The following section examines the effects of the various cell placement schemes on the placement efficiency.

3.4.3 Placement effects on efficiency

The dependence of the cell placement efficiency on the placement scheme is assessed by constructing a generic SoC description and engaging GENESYS 2004 to produce estimates of the placement efficiency and global interconnect demand (total length of global interconnect). In this case, the SoC is assumed to consist of several large megacells and numerous smaller megacells in various arrangements. Figure 30 illustrates the floorplan of the example SoC.

area

Collect cell statistics

$$width = \sqrt{cell_area}$$
$$height = \sqrt{cell_area}$$

Calculate cell dimensions

assume unity aspect ratio

Apply constraint criteria and re-order cells

system
area
bus

$$rows \approx \frac{N_{cell} \cdot W_{cell_ave}}{W_{chip}}$$

Estimate row count and assemble rows

chip width

Squeeze operations (vertical and horizontal)

$$C_{upper} = \frac{row_height}{\sqrt{\frac{A_{chip}}{grid_value^2}}}$$

Assign row/cell coordinates

grid value

Stack rows and re-assign coordinates

Output coordinates

Figure 28: Algorithm for generating the cell coordinates for an SoC floorplan. On the left are key expressions and illustrations, the middle block diagram describes each step of the algorithm, and the inputs to the algorithm are shown to the right.

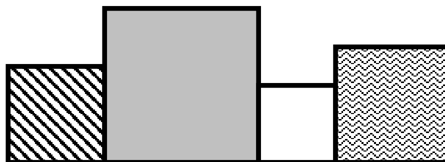

Figure 29: An assembled row of four cells of varying sizes prior to row squeezing operations. The total width of the row is approximately equal to the chip width.

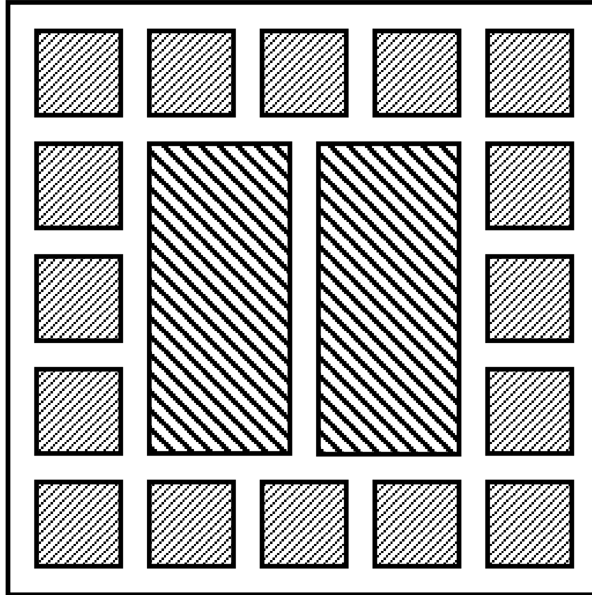

Figure 30: A generic SoC floorplan for examining the effects of placement schemes on the cell placement efficiency metric. The SoC consists of 18 megacells with a manually entered floorplan.

The smaller cells contain 20K gates each while the two larger cells contain 300K gates each. The smaller cells are arranged around the periphery so that the any global nets connecting groups of small cells will likely be spread out, yielding a relatively low placement efficiency for this approach. In addition to this placement scheme, two other placements are examined: the in-order and size based schemes. The in-order scheme is specified to place a grouping of smaller cells between the two larger cells. The resulting floorplan is shown in Figure 31. The cell size placement scheme attempts to maximize the placement efficiency for the generic SoC by producing a floorplan in which the smaller cells are tightly clustered and the larger cells remain adjacent. This floorplan is illustrated in Figure 32.

The simulation results for the placement efficiencies of each placement scheme are listed in Table 8:

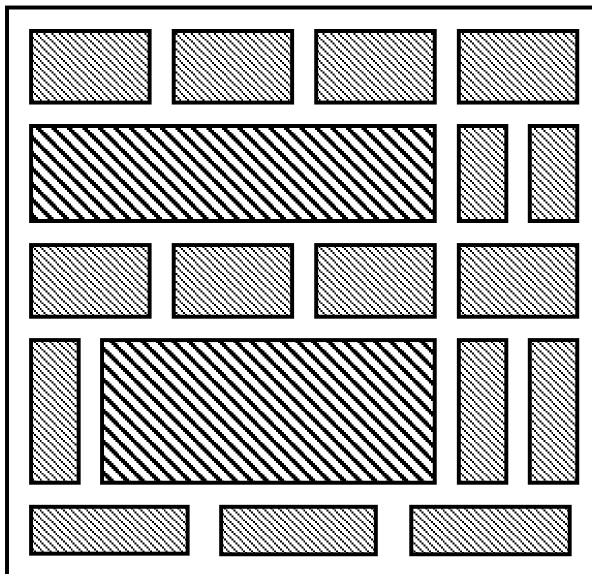

Figure 31: The floorplan of the generic SoC from Figure 30 under the in-order placement strategy where the two larger cells are separated by a grouping of smaller cells.

Table 8: Cell placement efficiencies for generic SoC example containing 18 cells for manual, in-order, and size based placement.

	Cell Placement Efficiency
Placement Type	$N_{cell} = 18$
Manual	73%
In Order	62%
Size	81%

The size ordered cell placement produces the highest placement efficiency of 81% for the example SoC. The clustering of the small cells takes advantage of locality by reducing the length of nets connecting the smaller cells without substantially increasing the average distance from the larger cells in the design. The in-order placement scheme in which the larger cells are separated while maintaining a wide spread between the smaller cells produces a placement efficiency of only 62%. The original floorplan of Figure 30 has an intermediate efficiency of 73% because the distance between the large cells and the smaller cells is minimized while maintaining adjacency

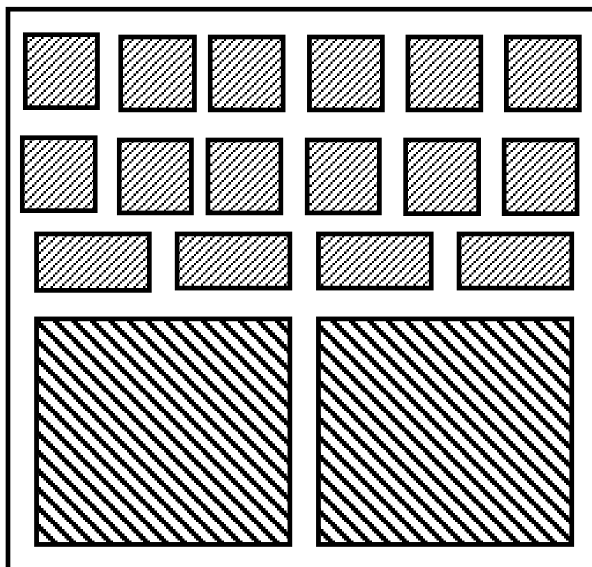

Figure 32: The floorplan for the generic SoC from Figure 30 under the size based placement strategy in which the smaller cells are closely clustered together.

between the large cells. Another manually entered floorplan placing the large cells at the top and bottom of the chip and clustering the smaller cells between them produces a placement efficiency of 72%. This value is very close to the results for the original floorplan, indicating that the impact of separating either the large or small cells in a fully interconnected stochastic distribution is roughly equal.

The size based automated placement scheme produces optimal cell placement efficiencies for the generic SoC illustrated in Figure 30. However, most SoC designs do not exhibit the same characteristic mix of cell sizes. The question remains as to whether these placement schemes produce similar results for more realistic SoC implementations as shown in Figure 21. The five system implementations featured in section 3.5 were simulated with each of the placement schemes utilized above. The results are listed in Table 9.

Table 9: Megacell Placement Efficiency for Commercial SoCs.

	Placement Type		
Processor	**Manual**	**In Order**	**By Block Size**
MIPS R2000	86%	83%	84%
Itanium2	83%	78%	81%
PowerPC	78%	74%	73%
Emotion	78%	79%	89%
UltraSPARC	81%	78%	80%

These results indicate that the manual placement taken from the die micrographs of each chip produces the highest placement efficiency in almost every case. For the MIPS R2000, Itanium2, PowerPC, and UltraSPARC systems there is little variation in the cell placement efficiency for the different placement schemes. Only the Emotion chip shows any significant improvement in placement efficiency. An inspection of the floorplan in Figure 34 shows a number of smaller cells spread out across the bottom and top edges of the chip. In this situation the placement efficiency is increased by reordering the cells.

3.5 Validation of the block SoC methodology

In order to develop confidence in the block modeling methodology the relative accuracy of the resulting performance projections with respect to the homogeneous model are examined. Both the heterogeneous and homogeneous system models are applied to five commercial SoC implementations, a Mips R2000 RISC microprocessor [35], the Intel Itanium2 processor, the Emotion engine graphics renderer, an UltraSPARC implementation, and a PowerPC RISC processor. The Mips processor is depicted in Figure 21b. The block diagrams of the remaining 4 processors are illustrated in Figures 33 through 34.

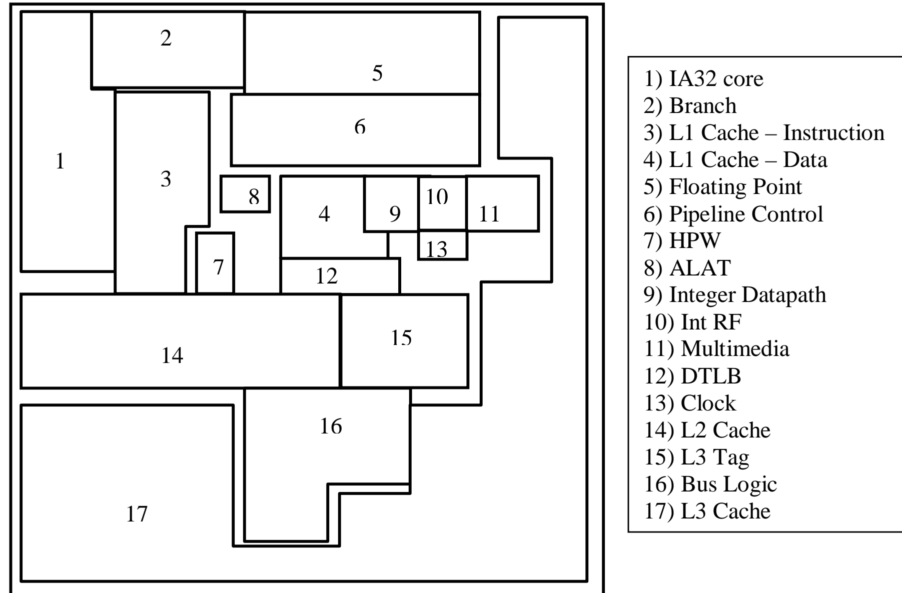

Figure 33: Die micrograph for a 1.1 GHz Intel Itanium2 microprocessor

3.5.1 Methodology

The homogeneous simulations are run with the proper technology and system level inputs to estimate the die size, clock frequency, and power dissipation for each of the designs. The heterogeneous system description is built by observing a die mircrograph for the block level layout. The cache transistor budget is subtracted from the total transistor count to estimate the number of logic transistors remaining. If no data on individual cell transistor counts is known, it is estimated by comparing the area of the cell to the total remaining chip area and assuming that the cell transistor budget is proportional. The individual cell placement is determined by superimposing a grid over the die micrograph and determining the coordinates of each cell. Once all cell data is compiled in the system description, simulations are run to yield the final results as described in the following section.

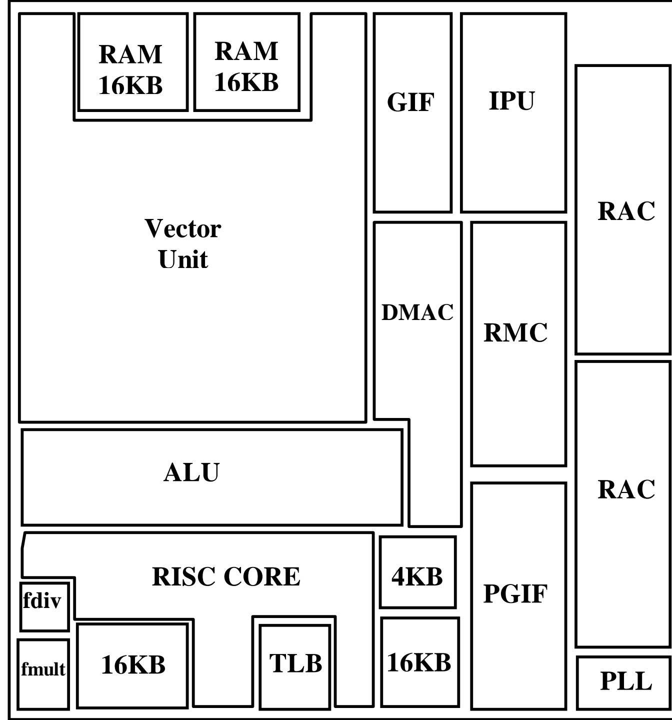

Figure 34: Die micrograph for a 300MHz Emotion graphics renderer

3.5.2 Results

The results of simulations of the five example SoCs for both system models are shown in Table 10. When compared against actual data the heterogeneous block-based system model produces estimates with higher accuracy than the homogeneous modeling. In particular, the homogeneous system model tends to underestimate the clock frequency. This is due to an overestimation of the average interconnect length resulting in increased critical path delay in the random logic network. These same results are presented in graphical form in Figures 37 through 39.

3.6 Conclusion

This chapter introduces the block modeling methodology utilized in GENESYS 2004 to accurately model the power, area, and clock frequency for complex SoCs. This

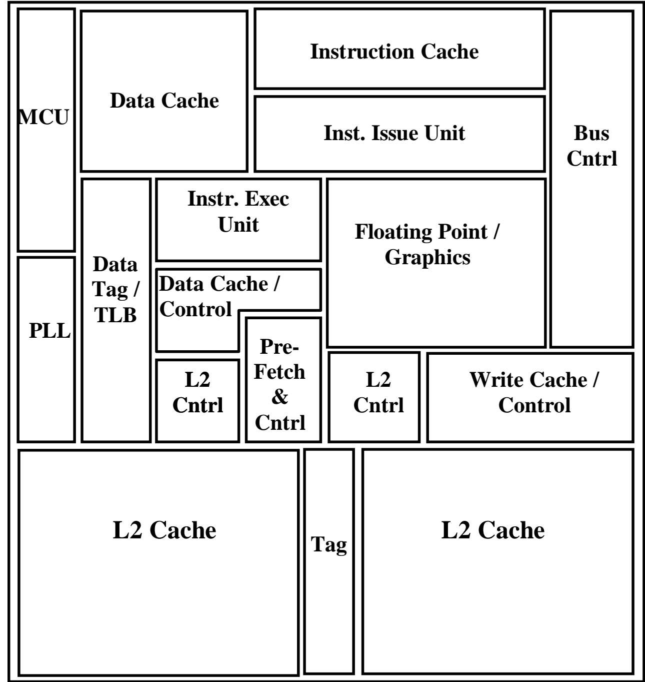

Figure 35: Die micrograph for a 1GHz UltraSPARC microprocessor

methodology preserves the core modeling at the fundamental, material, device, and circuit levels of the GSI hierarchy while extending the system level methodology to a broad class of system designs. The application of a stochastic distribution for modeling global interconnect resources is highlighted. A methodology for specifying the SoC floorplan to generate an average cell placement efficiency is introduced. An automated cell placement algorithm for floorplanning utilizing three different design constraints is introduced and applied to several example systems. Results show that the manual placement approach typically produces the highest placement efficiency. Finally, the performance data for five example systems is compared against simulation data for both the block SoC methodology and the older homogeneous system model with results for the power, area, and clock frequency showing significant improvement in accuracy for the block SoC methodology.

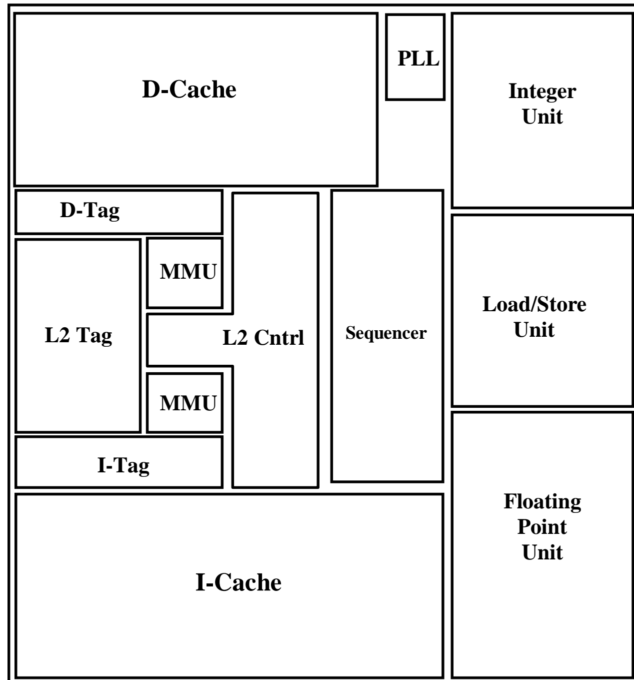

Figure 36: Die micrograph for a 250MHz PowerPC microprocessor

Table 10: Comparison of GENESYS simulation results for the heterogeneous system modeling methodology against actual data and homogeneous simulation results. The die size is in mm^2, the frequency is in MHz, and the power dissipation is in Watts.

Chip Simulation Results									
	Actual			Homogeneous			SoC		
Chip	Size	Freq.	Power	Size	Freq.	Power	Size	Freq.	Power
RISC	295	200	30	350	180	33	304	197	28.5
Itanium	421	1,000	130	345	810	105	411	1,065	131
Emotion	225	300	18	318	245	31	200	311	20
Ultra SPARC	180	1100	53	248	790	36	188	1150	60
Power PC	67	250	5	109	194	8.4	60	240	7.4

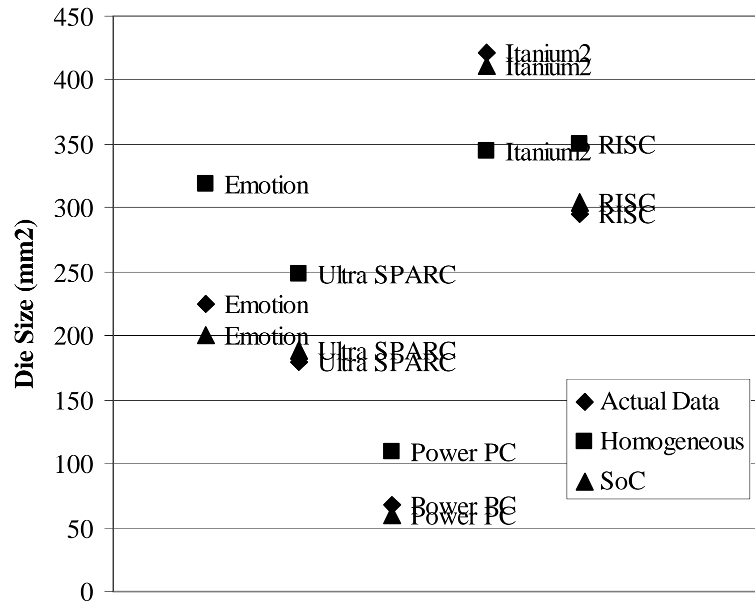

Figure 37: Validation of die area $[mm^2]$ projections for the homogeneous and block based system methodology (SoC)

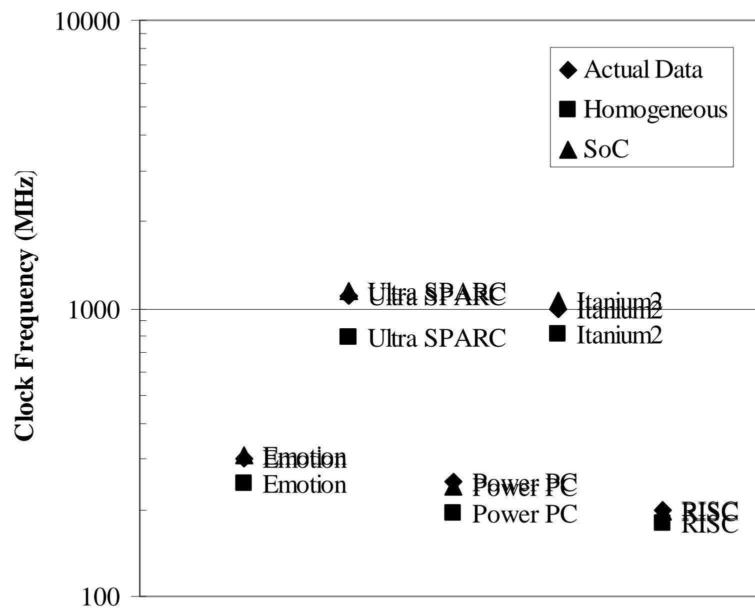

Figure 38: Validation of clock frequency [MHz] projections for the homogeneous and block based system methodology (SoC)

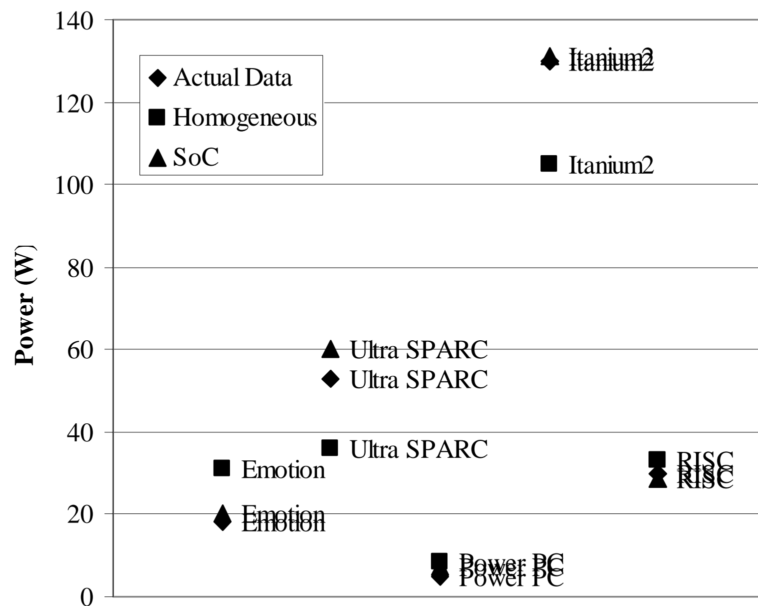

Figure 39: Validation of power dissipation [W] projections for the homogeneous and block based system methodology

CHAPTER IV

PHYSICAL ON-CHIP BUS MODELING

4.1 Introduction

Traditional system bus architectures are primarily designed for implementation at the printed circuit board (PCB) level to connect various peripherals and memory to the central processing unit. With increasing on-chip integration density and functionality, the ultimate performance of these systems is limited less by the processor than by the speed of the system bus. The design and performance of the off-chip system bus is limited by large PCB trace widths, limited pin count, and the distance between on-board components. Historically, the growth rate for processor speed has greatly outstripped speed increases for the off-chip interface as illustrated in Figure 40 [36] [37]. The peripheral component interface (PCI) bus has remained at 33MHz since its introduction in 1993. Memory bus speeds have increased from 33MHz to 200MHz in the time period captured in Figure 40. In the same time, however, processor core speeds have sky-rocketed by a factor of nearly 50 from 66MHz to over 3 GHz in the preceding decade. SoC is intended to provide a solution to the increasing gap between on-chip and off-chip communication speeds.

The SoC design methodology places processor(s), memory, and peripherals on a single-chip. This allows the system designer to take advantage of small wire

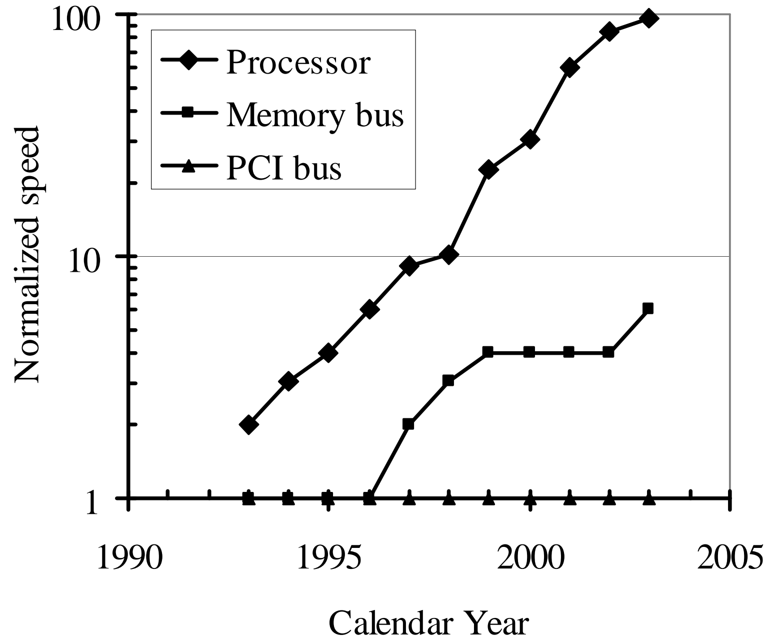

Figure 40: Historical speed trends for the growth of on-chip vs off-chip frequencies normalized to the 1993 calendar year.

dimensions, virtually unlimited pin count, and the tightest possible integration between components. The prevalent method for designing SoCs involves the use of pre-designed intellectual property (IP) cores/megacells. Under this design methodology numerous megacells are placed on-chip and wired together often utilizing custom glue logic to coordinate communication between megacells. The drawback of glue logic is the verification time required to ensure proper operation between the custom logic and the IP cores connected to it [38]. Several commercial implementations of on-chip bus architectures such as the IBM CoreConnect, PalmChip CoreFrame, and ARM AMBA are designed to reduce or eliminate the use of custom glue logic by providing IP based bus cores for rapid design and verification [39][40][41]. The performance of future SoC systems is heavily dependent upon the ability of the on-chip bus to provide sufficient bandwidth for data flow between the processor(s) and peripheral components.

Chapter 3 outlined the global interconnect architecture for a generic SoC based upon a stochastic distribution for multi-terminal nets. This global distribution assumes full interconnection between all of the cells in the SoC. This approach is useful for a quick estimation of global wiring requirements in the absence of detailed wiring information such as a global netlist, but the global wiring of IP based SoCs utilizing a bus based protocol is more reflective of system design choices such as the design of the bus architecture. In order to project the resource requirements and potential performance of on-chip bus architectures, GENESYS must specifically model this critical class of interconnects. This chapter details the methodology for modeling on-chip system buses in GENESYS 2004.

4.2 A Generic On-Chip Bus Model

In GENESYS 2004 buses are considered as a specific class of interconnects which are explicitly defined by the user/system designer. In this case, the bus is modeled as a grouping of global interconnects or nets with a defined set of sources/sinks. The observation behind this generic model is that no matter the specifics of the architectural design of the bus, it consists of devices and wires.

4.2.1 Bus specification in GENESYS 2004

On-chip buses are specified in a manner similar to the declaration of blocks and megacells discussed in Section 3.2. The example below illustrates the declaration of a bus:

entity BUS sys_bus **is**

physical_width = 1.0 [um];

logical_width = 32 [bits];

```
aspect_ratio = 1.0 [];
spacing = 1.0 [um];
driver_size = 25.0 [W/L];
line_type = 2 [0-unbuffered,1-buffered/unidirectional,2-buffered/bidirectional];
bus_list (cell_1,cell_2,cell_3...cell_n);
end entity sys_bus;
```

The above definition differs from the declaration of a megacell only by the parameter names and the special parameter bus_list. The parameters physical_width, spacing, and aspect_ratio govern the physical dimensions of the bus wires. The parameter logical_width sets the number of bit lines for the bus and driver_size adjusts the width to length ratio of the gate(s) driving the bus lines. The parameter line_type selects one of the three link types discussed in the following section and bus_list parameter is a listing of the cell instances connected to the bus. In short, the above declaration can be viewed as a bus netlist. Multiple buses may be specified on chip to model the physical structure of hierarchical bus architectures such as the IBM CoreConnect standard. Bus entities are not instantiated within the system level entity because GENESYS 2004 handles all bus related calculations after completing the simulations for individual cells.

4.2.2 Bus Link Models

GENESYS 2004 models three basic types of bus lines: an unbuffered bi-directional link with multiple fan-out, a buffered unidirectional link with conventional repeater insertion, and a buffered bi-directional link utilizing coupled bi-directional repeaters. The basic circuit model for each type of bus line is shown in Figures 41a-41c.

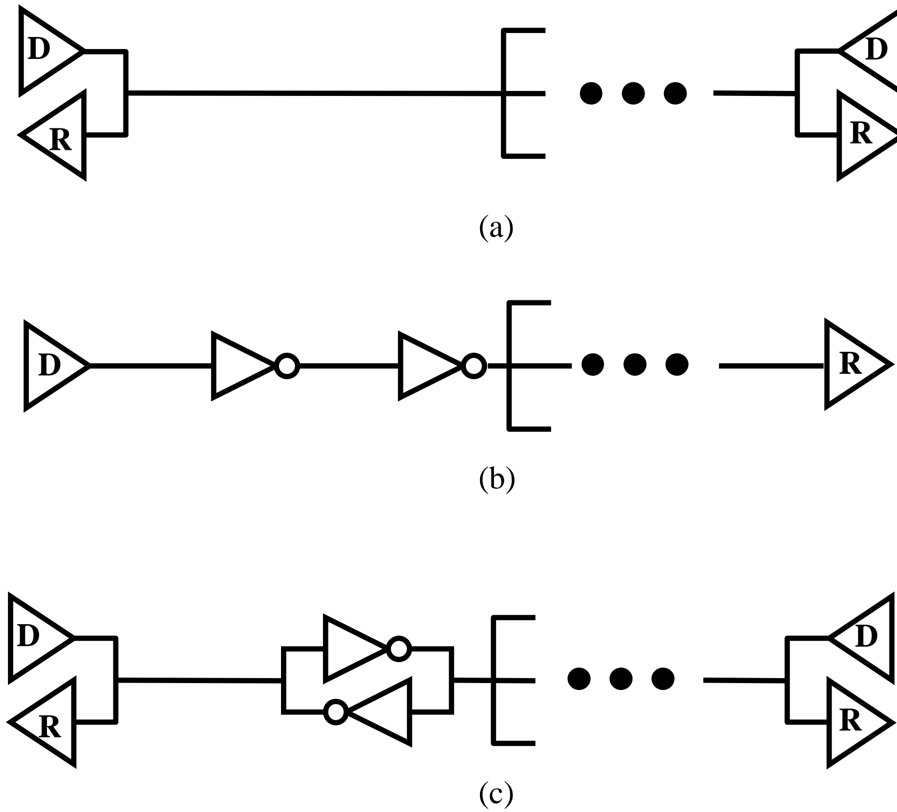

Figure 41: The circuit level model for three generic bus line types: (a) an unbuffered bi-directional line, (b) a buffered uni-directional line, and (c) a buffered bi-directional bus line utilizing coupled repeaters.

Each type of bus has advantages and disadvantages. The unbuffered bus provides bi-directionality at no further cost, but is substantially slower than a buffered line. The buffered bus with standard repeaters inserted is the fastest option but is uni-directional. It may be operated in a bi-directional mode if the number of lines is doubled to provide outgoing and incoming channels, however, this option doubles the expected bus power dissipation and area utilization. The buffered bus with coupled repeaters provides bi-directionality with half the bus lines of the unbuffered case, but experiences greater delay due to increased loading from the coupled repeaters. The bus line segment between repeaters (if a buffered line is chosen) is modeled as a distributed RC line. Each node of the bus is terminated with a twin driver/repeater.

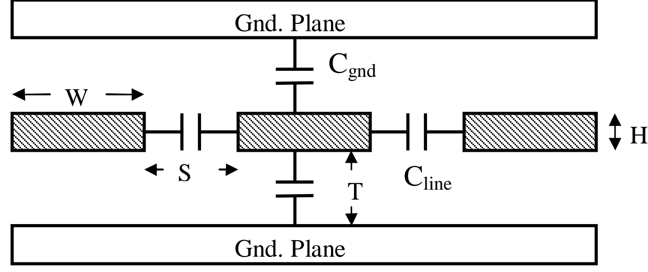

Figure 42: The physical model for the bus lines assumes ground planes above and below as well as coupling to the nearest neighbors.

The physical model for the bus lines segments between coupled repeaters is similar to the standard pt-pt interconnects for the local interconnect distribution. Figure 42 illustrates the structure and environment of a bus line. This model idealizes the on-chip wiring with the assumption of ground planes above and below the active lines as opposed to orthogonal wires.

Where C_{gnd} and C_{line} are the ground capacitance and line to line parasitic capacitance respectively. The total capacitance of each type of bus line including the bi-directional repeaters is calculated as in Equations 24-26:

$$C_{bus1} = 2(C_{line} + C_{gnd})L_{net} + N_n(Cin_{rec} + Cout_{drv}) \tag{24}$$

$$C_{bus2} = 2(C_{line} + C_{gnd})L_{net} + N_{rep}(C_{in} + C_{out}) + N_n Cin_{rec} \tag{25}$$

$$C_{bus3} = 2(C_{line} + C_{gnd})L_{net} + 2N_{rep}(C_{in} + C_{out}) + N_n(Cin_{rec} + Cout_{rec}) \tag{26}$$

Where L_{net} is the length of the bus line, N_{rep} is the number of repeaters, N_n is the number of nodes connected by the bus line, and C_{in}/C_{out} are the input and output capacitances of the repeaters. As can be seen from the above equations, the loading on the bus lines increases with the addition of repeaters and receiver/driver circuits.

4.2.3 Calculating the Bus Length

Before any estimation of the required routing resources or bus performance can be made, the physical length of the bus lines must be estimated. The length of a bus line is determined by two criteria: the number of cells connected and the placement of those cells within the SoC. The first step in calculating the bus length is to collect the cell placement information described in Section 3.4. The location of all the cells in the SoC floorplan is now known. The second step is to collect information regarding the cell connected to the bus. Once the connected cells and their coordinates are known, a bounding box enclosing the correct region is found via the following expressions utilizing the cell placement information:

$$a = max(r_i) - min(l_i) \tag{27}$$

$$b = max(u_i) - min(lwr_i) \tag{28}$$

where a and b are the bounding box dimensions, and r_i, l_i, u_i, lwr_i are the right, left, upper and lower coordinates respectively. The index i refers to all cells within the bus netlist.

These dimensions entirely enclose every megacell connected to the bus. The likelihood of the bus terminals begin placed at the far edges of the bounding box is not high. Therefore some adjustment needs to be made to the bounding box. The net bounding box, the box enclosing the actual terminals of the bus, is estimated by applying the formula for the dimensions of an m node net with randomly placed terminals within the megacells. The expressions below compute the expected net bounding dimensions from [34]:

$$\hat{a} = \frac{m-1}{m+1}a \tag{29}$$

$$\hat{b} = \frac{m-1}{m+1}b \tag{30}$$

Where a and b are the net bounding dimensions. Given this information, the length of the bus lines can be calculated via Equation 15:

$$L_{line} = (\alpha m^{\gamma} - \beta)\frac{\hat{a} \cdot \hat{b}}{\hat{a} + \hat{b}} + (\hat{a} + \hat{b})$$

Where L_{line} is the physical line length, m is the number of bus terminals, and α, γ, and β are empirical parameters. The first term containing the empirical parameters represents the expected length of the lines to the branch terminals while the second term, $\hat{a} + \hat{b}$, is the corner to corner length of the net bounding box. From [34] the proper values for good agreement with observed data are: $\alpha = 1.1$, $\gamma = \frac{1}{2}$ and $\beta = 2$. An example of the bounding box calculation for a generic SoC is shown in Figure 43.

An example of the variation in bus length for various fanouts and bus netlists is presented below. The generic SoC floorplan from Figure 43 is utilized. The bus length is evaluated for m values from 2 (minimum) to 5 (maximum) for various configurations. The five cells present in the generic SoC are numbered according to the block diagram of Figure 44.

The configuration numbers such as (1,2,3) mean that cells 1, 2, and 3 are connected on the bus. For this example 12 different configurations are examined. The results are presented in Table 11.

The bus length increases with the number of terminals from a minimum of $5mm$ for

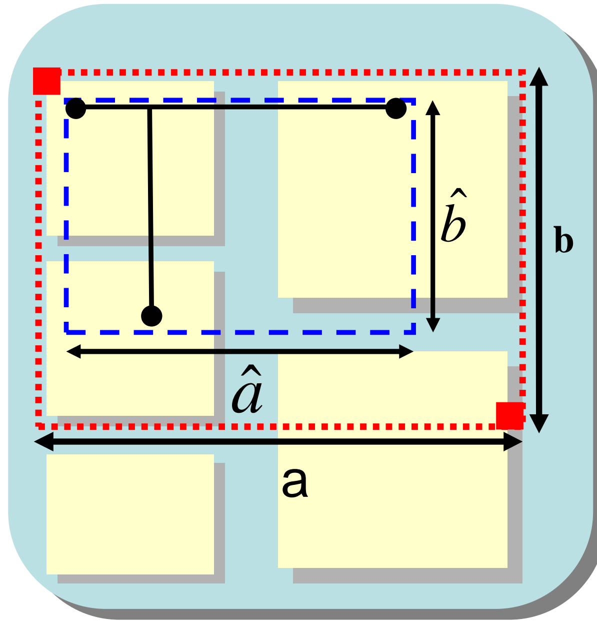

Figure 43: An illustration of the determination of the net bounding box. The cell bounding box dimensions are determined by enclosing the bus connected cells to yield the dimensions a and b. Equations 29 and 30 are applied to get the final net bounding box dimensions $\hat{a}$ and $\hat{b}$.

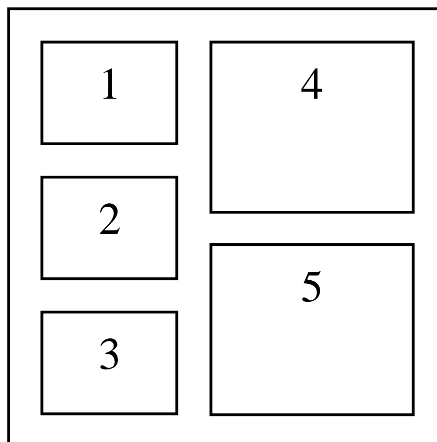

Figure 44: Example of bus length calculation for generic SoC floorplan

Table 11: Bus length calculations for various fanouts and configurations for generic SoC floorplan of Figure 44.

# Terminals	Config.	Length $[mm]$
$m = 2$	(1,2)	5.0
	(1,3)	6.5
	(1,4)	6.8
	(1,5)	8.7
$m = 3$	(1,2,3)	10.8
	(1,2,4)	12.2
	(1,3,5)	14.6
	(2,4,5)	14.6
$m = 4$	(1,2,3,4)	18.9
	(2,3,4,5)	18.9
	(1,3,4,5)	18.9
$m = 5$	(1,2,3,4,5)	22.3

$A_{chip} = 225mm^2$ (15x15mm)

two adjacent cells to 22.3mm for a bus connecting all five cells. For each given fanout, the maximum bus length is found for any combination of cells which span the entire chip. With $m = 4$ and only 5 cells, there is no combination that does not span the chip, therefore, the expected length of the bus remains constant. The increase in length for the $m = 5$ case is due to both the extra terminal and the increase in the expected dimensions of the net bounding box.

4.2.4 Bus Delay

Depending upon the choice of line type chosen from section 4.2.2 the delay of the bus line is calculated in one of two ways. If the unbuffered line from Figure 41a is used, the delay is calculated taking the branch capacitances into account along the full length of the bus line. If either of the buffered lines from Figures 41b/c are chosen, the delay is calculated through the longest point to point path between the furthest cells.

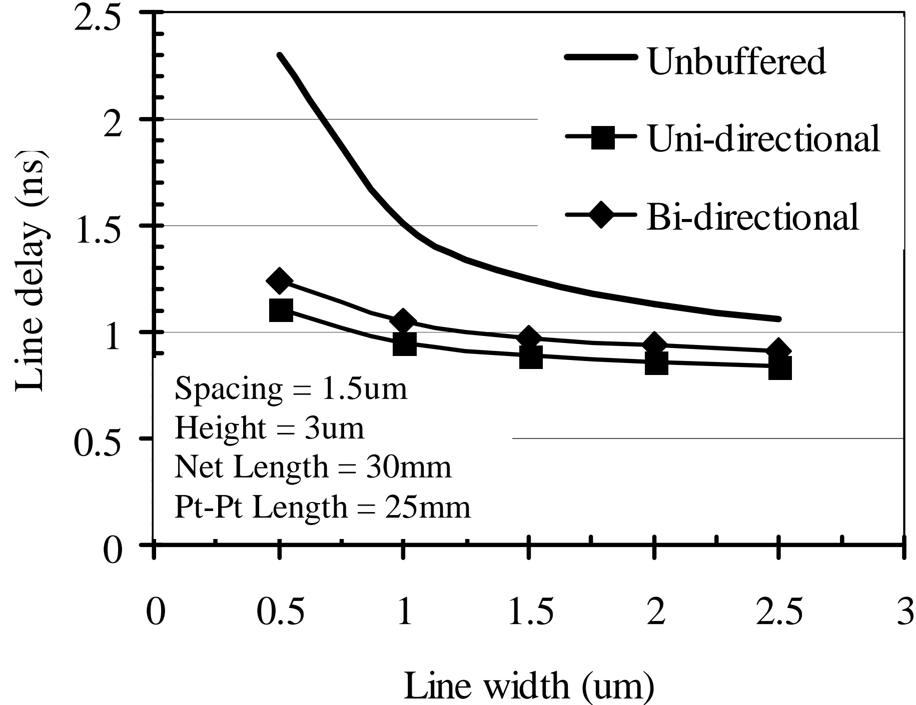

Figure 45: The delay vs line width for the three different types of bus lines from Figure 41. The spacing, height, and length of the bus are kept constant as shown in the lower corner. The unbuffered line shows significantly greater delay. The pt-pt length is the distance between the two farthest nodes. The number of nodes connected by the bus in this example is 7.

The delay trends for each type of bus are plotted against increasing line width in Figure 45. The chip area is $300mm^2$ with a total of 20 megacells of which seven are connected via a bus. The line with the lightest capacitive loading, the unbuffered, case also exhibits the greatest delay. There are two primary effects which penalize the unbuffered line. first, the delay is proportional to the square of the length in an unbroken RC line. Additionally, the branch capacitances from the N_n nodes produce additional delay at the farthest node. The buffered lines produce lower delay times because they break up the square dependence on the length and because the buffers allow the branches to charge in parallel with one another. This permits us to calculate the delay of the multi-terminal net along its longest point to point path. For the

Table 12: Bus delay at $5\mu m$ line width for the three generic bus line types from Figure 41.

Line Config.	Delay [ns]	% increase
Unbuffered	9.53	13.8
Uni-directional	8.37	–
Bi-directional	8.9	6.3

$L_{net} = 30mm$, Spacing = 1.5μm, Height = 3μm

example shown in Figure 45 this corresponds to $25mm$ vs $30mm$ for the unbuffered case. For small line dimensions, the buffered lines have an obvious advantage over the unbuffered line, but for larger dimensions this advantage is eroded. For example, at a line width of $5\mu m$ the delay of the unbuffered line is only 14% greater as shown in Table 12. The results from these simulations indicate that the preferred generic bus link for connecting large clusters of cells.

If the delay and logical width of the bus are known, the peak bandwidth is calculated via the following expression:

$$BW_{peak} = N_{bit} \cdot f_{bus} \tag{31}$$

Where N_{bit} is the number of bus lines and f_{bus} is the maximum bus frequency. The peak bandwidth for the example of Figure 45 assuming a 32 bit bus ranges from less than 14 Gbps for the unbuffered line to more than 37Gbps for the buffered uni-directional bus.

4.2.5 Bus Power dissipation

The average power dissipation in an active bus line can be calculated by assessing the energy requirements for transmission of a bit. The general formula for calculating the average dynamic power is given below:

$$P = \frac{1}{2} C V^2 f_c a_f \tag{32}$$

Where C is the total switching capacitance, V is the voltage, f_c is the operating frequency, and a_f is the activity factor or percentage of cycles where active switching occurs. If Equations 24 - 26 are substituted, the expressions for calculating the bus power dissipated for each type of bus link under active switching are as follows:

$$P_{bus1} = \frac{N_{bit}}{2} \{2(C_{line} + C_{gnd}) L_{net} + N_n (Cin_{rec} + Cout_{drv})\} {V_{dd}}^2 \cdot \frac{a_f}{\tau_{bus}} \tag{33}$$

$$P_{bus2} = \frac{N_{bit}}{2} \{2(C_{line} + C_{gnd}) L_{net} + N_{rep}(C_{in} + C_{out}) + N_n Cin_{rec}\} {V_{dd}}^2 \cdot \frac{a_f}{\tau_{bus}} \tag{34}$$

$$\begin{aligned} P_{bus3} = {} & \frac{N_{bit}}{2} \{2(C_{line} + C_{gnd}) L_{net} + 2N_{rep}(C_{in} + C_{out}) \\ & + N_n (Cin_{rec} + Cout_{rec})\} {V_{dd}}^2 \cdot \frac{a_f}{\tau_{bus}} \end{aligned} \tag{35}$$

Where τ_{bus} is the delay of the bus line. GENESYS 2004 calculates the power dissipation for a bus operating at maximum frequency. If the bus speed is faster than the system clock, the bus may be constrained to operate at the system frequency. In this case, the power would be adjusted dividing by the ratio of the bus frequency to

Table 13: Average bus power comparison with data for different signaling techniques from [42] with GENESYS simulation. The logical bus width is 32 bits implemented in a 0.35μm technology.

Bus type/mode	**Power Dissipation (mW)**
Current mode	225
Voltage mode	125
Adaptive mode	100
GENESYS	175

a_f = 15, L_{bus} =1 cm, F_{bus} = 1GHz

the system clock frequency. For random logic an activity factor of 10% is typically assumed and provides excellent agreement with empirical data for chip power dissipation. The activity factor for individual bus lines depends heavily upon numerous factors, but data obtained from simplescalar simulations and reported in literature [42] indicates an average activity factor of about 15% for individual bus lines under the transmission of instruction address streams. The behavior for data address streams which exhibit more random behavior with an average activity factor of approximately 20%. An examination of bit patterns for both instruction and data indicates that a large percentage of bits are static (retaining the same value for numerous cycles) so that even with high bus utilization the average activity factor will often remain low. Using an activity factor of 15% for a 32 bit bus with a line length of 1cm operating at 1GHz produces bus power results that agree with published data from [42].

For the example from Figure 45 the power is calculated for various line widths and plotted in Figures 46 and 47. Clearly, from Figure 46, the unbuffered line gives the lowest power dissipation of the three cases. The buffered bi-directional link experiences the greatest average power dissipation due to additional loading from the coupled repeaters. The peak power for the bi-directional buffered line is nearly 600mW at 1GHz. Much of the difference in power dissipation between the three is accounted

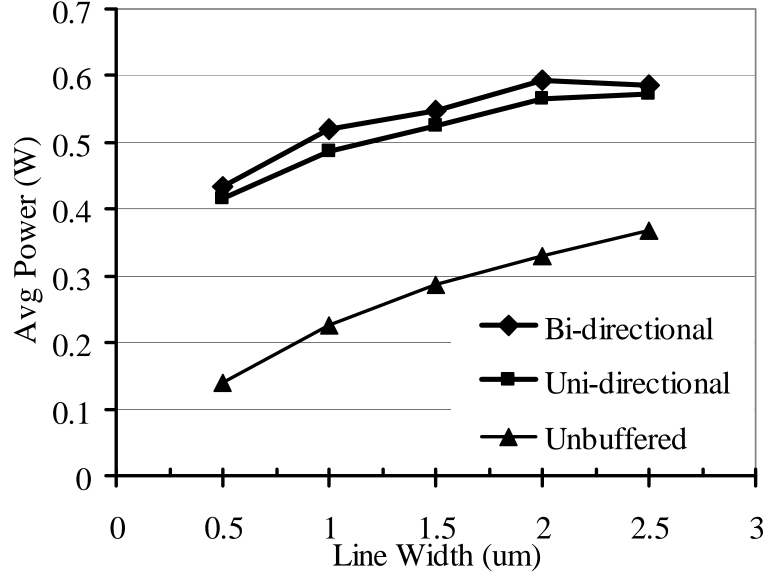

Figure 46: The average bus power consumption (W) for a 32 bit, $30mm$ bus for each link type detailed in section 4.2.2 plotted against physical line width (μm). The activity factor is set to 15%.

for by the respective line delays for each link type. Figure 47 adjusts for this difference by assuming that the link delay is limited to the system clock delay of 5ns (200MHz). The results show that the unbuffered line dissipates approximately 25mW less than the uni-directional line at the same frequency and $W_{line} = 0.5\mu m$. The effective bandwidth is 6.4 Gbps for all three cases, therefore, under the assumption of a limited bus frequency speed the unbuffered bus is the superior choice due to lower power dissipation at equal performance. This is true up to a clock frequency of 950MHz and peak bandwidth of 30Gbps where the unbuffered bus achieves its maximum speed for the maximum line width of $2.5\mu m$, if more than 30Gbps is desired, buffered lines must be utilized.

4.2.6 Bus Routing Area

A complete bus (or bus segment) is a collection or grouping of individual bus lines sharing identical node destinations. The logical width of the bus is the number

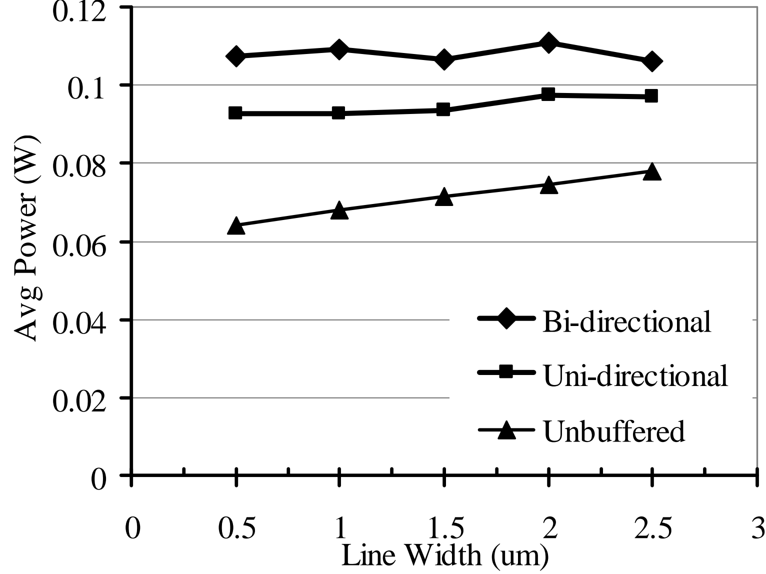

Figure 47: The average bus power consumption (W) for a 32 bit, 30mm bus for each link type at a common clock frequency of 200MHz. The activity factor is set to 15%.

of bits or bus lines comprising the bus segment. The routing resources required to wire the bus are dependent upon the length of the bus lines, the physical width and spacing of the lines, and the logical width of the bus. The expression for calculating the bus wiring area is given in Equation 36:

$$A_{bus} = N_{bit}(W_{line} + S_{line})L_{line} \tag{36}$$

Where N_{bit} is the logical bus width, W_{line}/S_{line} are the physical width and spacing, and L_{line} is the length of the bus line. From 36 it is clear that the area requirement increases linearly for each of the key dimensions. The area requirements for the example of Figure 45 and Table 12 for a 32 bit, 30mm bus are given in Table 14.

GENESYS 2004 assumes that any specified bus wiring is routed in addition to the stochastic wiring distribution detailed in Section 3.3. This means that the area reserved for bus wiring is subtracted from the available area to route any remaining

Table 14: Bus area requirements for example from Figure 45

Line width $[\mu m]$	**Bus area** $[mm^2]$
0.5	1.91
1.0	2.38
1.5	2.86
2.0	3.34
2.5	3.81
5.0	6.19

Spacing = $1.5\mu m$, Height = $3\mu m$

global interconnects. The total available routing area, and supply (maximum length of routable interconnect) for global interconnects is calculated via the following expressions:

$$A_{route} = A_{chip} N w_{glb} \quad (37)$$

$$Supply = e_w \frac{A_{route}}{p_w} \quad (38)$$

Where A_{chip} and Nw_{glb} are the chip area and number of global wiring levels. e_w and p_w are the wiring efficiency (usually about 40%) and the wiring pitch. The wiring pitch for the stochastic global interconnect is determined via the algorithm from Figure 23. The impact of the bus routing on the global interconnect routing area is determined as follows:

$$A_{route} = A_{chip} N w_{glb} - A_{bus} \quad (39)$$

The impact of bus routing in a generic SoC is examined by defining a single shared global bus connecting all 20 megacells in the example from Figure 45 and observing the performance of global interconnects. Because this is a maximum length bus, the link type chosen is the buffered bi-directional line from Figure 41c.

Table 15: Simulation results for 32 bit shared, global bus connecting all cells for a $300mm^2$, 20 megacell generic SoC.

Width [μm]	A_{bus} [mm^2]	**bus delay** [**ns**]	**global pitch** [μm]	$globaldelay[ns]$
0.35	1.2	3.0	7.0	1.0
0.5	1.7	2.3	7.0	1.0
1.0	3.4	1.44	7.0	1.0
2.0	6.8	1.0	7.0	1.0
3.0	10.7	0.93	7.0	1.0
4.0	13.6	0.9	7.0	1.0
5.0	17.0	0.89	6.8	1.0

$W_{line} = S_{line}$, Length = $53mm$,F_c=200MHz, Nw_{glb}=2

These results show that for reasonable line dimensions the area required for routing the 32bit global bus does not significantly impact the performance of the stochastic global network. Further simulations with an increased logical bus width show that the global distribution can support a maximum length bus of up to 256 bits before any significant increase (10%) in the global delay. For this example the processor clock speed is limited by the delay through the random logic to 200MHz, therefore the bus if operated independently of the core processor, could achieve up to 5 bus cycles per processor cycle with a bus line pitch of $4\mu m$.

4.2.7 Bus placement optimization

As discussed briefly in Section 3.4.2, GENESYS provides a method for generating automatic floorplans with a placement constraint designed to reduce the length of the bus by placing bus connected cells with a high degree of adjacency. The effectiveness of this technique is examined by applying the various cell placement schemes to the floorplans of Figures 21 and 33, the MIPS R2000 and Intel Itanium2 respectively. The technology implementations for the two examples are $0.35\mu m$ and $0.18\mu m$. The

Table 16: Simulation results for bus length and delay for various cell placement schemes.

Chip	placement	efficiency	bus length	bus delay	bus area
R2000	manual	85.7%	29.8mm	0.77ns	13.3mm^2
	inorder	82.3%	28.6mm	0.75ns	12.8mm^2
	size	84.6%	29.1mm	0.75ns	13.0mm^2
	bus driven	85.6%	21.53mm	0.64ns	9.7mm^2
Itanium2	manual	83.4%	29.3mm	0.68ns	6.8mm^2
	inorder	82.9%	34.0mm	0.77ns	7.8mm^2
	size	81.8%	34.4mm	0.77ns	7.9mm^2
	bus driven	80.5%	21.8mm	0.56ns	5.0mm^2

generic bus superimposed upon these floorplans is assumed to connect the functional units/processor core to the memory subsystem. The number of nodes connected by the R2000 and Itanium2 floorplan buses are 7 and 6 respectively. Both buses are wired at a pitch equal to 20 times the minimum pitch afforded by the minimum feature size. Simulation results are presented in Table 16.

These simulation results show that the bus driven placement scheme substantially reduces the length of bus signal lines in the above example. The reduced bus delay for each of the two cases presented results in an increase in the peak bandwidth of 20%. This method is most effective in SoCs where a bus connects a small enough number/area that a higher degree of cell adjacency results in a cell bounding area substantially smaller than the unoptimized case. The example from Table 15 cannot benefit from the bus driven placement scheme because all of the cells are connected leaving no room for optimization.

4.3 Bus limitations on system throughput

As discussed in Section 4.1 the performance of a bus-based SoC architecture is heavily dependent upon the performance of the bus. In traditional microprocessor systems, GENESYS determines the expected throughput via the use of an empirically based logic-memory model detailed in Section 2.3.1. This logic-memory model is applicable to standard uni-processor families such as the Intel and AMD x86 based systems, but does not extend to a generic SoC model in which there may be multiple processors and integrated peripherals, nor does it consider the performance of on-chip busses. The models developed in this section address the impact of on-chip bus performance on the system level throughput measured in instructions/ops per second.

The architectural design space for SoCs is extraordinarily wide, ranging from uni-processor cores with integrated peripherals to mixed signal/RF applications to complex multiprocessor systems. The physical system level modeling utilized in GENESYS 2004 works for a large portion of the SoC design space because there is commonality with regard to the underlying technology. However, there is no generic throughput model that adequately addresses the vast difference in design and functionality for the SoC design space because there is such little commonality at the architectural or programming level. Therefore, this section focuses on a specific class of SoC designs in which a processor core and on-chip memory are the key components along with on-chip peripherals. A block diagram of the generic SoC core is illustrated in Figure 48.

The throughput model developed for exploring the effects of the bus design on the system performance focuses on the interaction of the processor core and memory. The processor core can be treated in a manner similar to the uni-processor model and the overall throughput for the processor core is determined by evaluating the

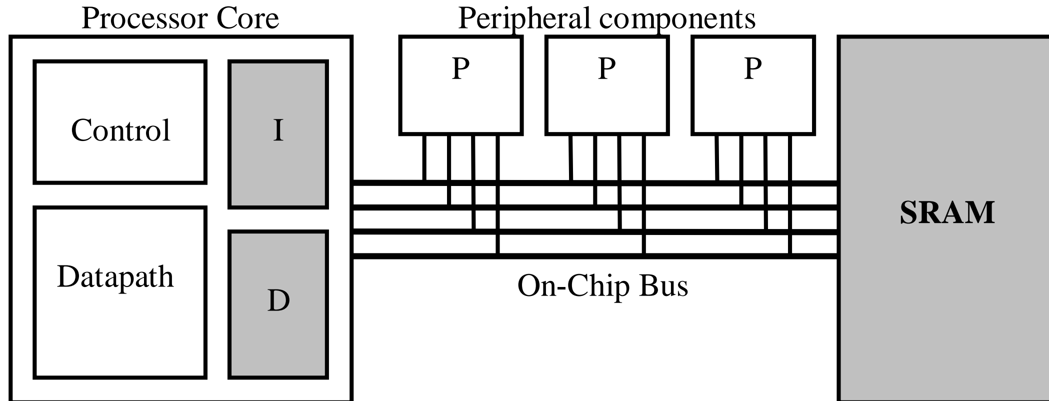

Figure 48: A generic SoC processor-memory core connected to peripheral components and SRAM memory via an on-chip bus.

average penalty for bus transactions arising from instruction or data references that miss in the processor local memory. The primary assumption for this analysis of bus limitations on system throughput is that the processor core is the key performance determining component such that the relative performance for the SoC mirrors the processor core. The basic model for the processor core throughput is listed in Equation 40.

$$Throughput = F_c(CPI_{base} + C_{bus} + C_{mem}) \tag{40}$$

Where F_c is the core clock frequency, CPI_{base} is the ideal cycles-per-instruction or operation, C_{bus} is the average penalty for bus accesses per op, C_{mem} is the average penalty for memory access.

4.3.1 Transaction model

A model for a generic bus transaction is needed to evaluate the individual components of the throughput model. The transaction model adopted for use in GENESYS 2004 considers both pipelined and non-pipelined bus transactions.

4.3.1.1 Non pipelined bus transactions

Figure 49 illustrates the bus transaction model for a non-pipelined (only one outstanding transaction permitted) bus. The tenure in units of bus cycles is formulated for each component. B_{word} is the number of bytes per word, W_{bus} is the logical width of the bus in bits, F_{bus} is bus operating frequency, T_{access} is the memory access latency, B_{data} is the size of the data stream transferred over the bus, and the X following the data transfer represents the null cycle terminating the data tenure. The transaction width is the total number of cycles between the request (A) and the end of the data tenure (X).

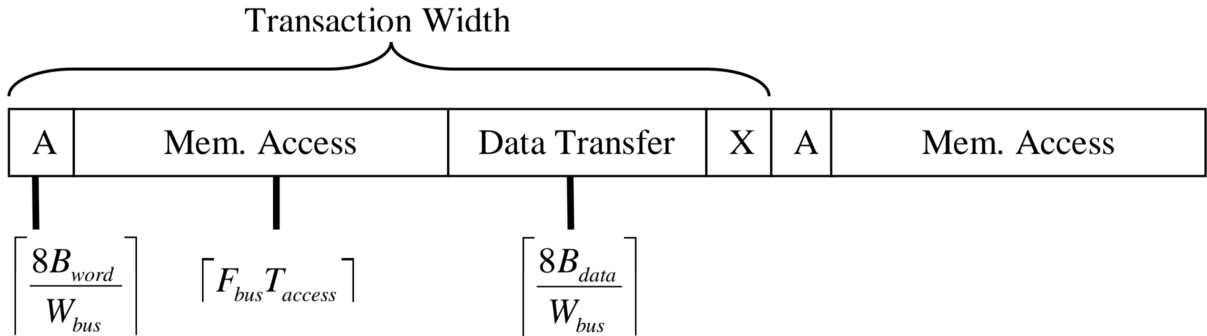

Figure 49: A generic bus transaction model for a non-pipelined bus. The transaction begins with the transmittal of the address information followed by the memory access latency and the return of multiple bytes of data. The transaction ends with a null cycle to separate data and address tenures. The expressions for the tenure (in bus cycles) for each component are listed.

In a non-pipelined bus, a bus transaction can only be initiated when there are no other outstanding transactions. For this case the total penalty assessed for a bus transaction is simply the sum of the individual components. Every subsequent transaction is assessed the same penalty, but any contention for bus resources results in additional penalties while subsequent transactions await completion of preceding transactions.

4.3.1.2 Pipelined bus transactions

A pipelined bus utilizes separate address and data buses to allow multiple outstanding transactions as a method for hiding the latency of preceding transactions. This method is illustrated in Figure 50.

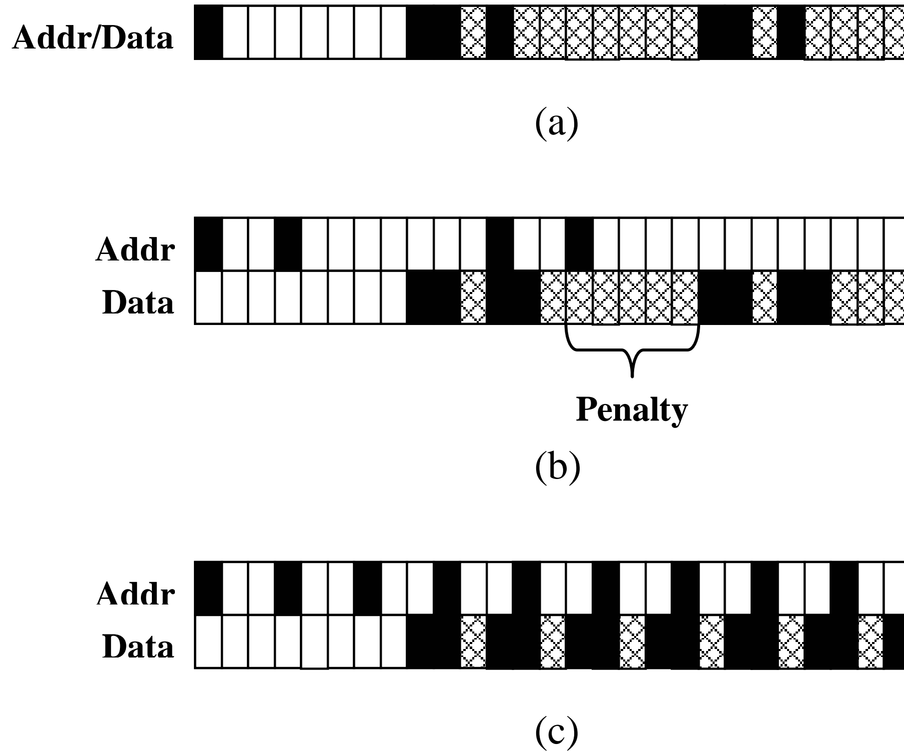

Figure 50: A generic bus transaction model for a pipelined bus. (a) non-pipelined bus for comparison, (b) a bus with suboptimal (level 1) pipelining allowing up to 2 outstanding transactions, and (c) an bus with optimal pipelining.

An optimally pipelined bus is capable of supporting enough outstanding bus transactions to fully hide the latency of previously scheduled transactions as in Figure 50c. Sub-optimal pipelining occurs when the number of permitted address requests cannot cover the transaction latency as illustrated in 50b. Ideally, each address request would be separated by a number of bus cycles no greater than data tenure. If only two simultaneous transactions are supported and the latency is greater

Table 17: The optimal pipeline depth assuming 32 byte data transfers over a 200MHz bus for various bus widths and memory latencies.

T_{access} (ns)	Bus width (bits)			
	32	64	128	256
50	2	3	4	6
45	2	2	4	5
40	1	2	3	5
35	1	2	3	4
30	1	2	3	4
25	1	2	2	3
20	1	1	2	3
15	1	1	2	2
10	1	1	1	2
5	1	1	1	1

than the ideal spread, a third transaction would have to await completion of the first. The optimal pipeline depth is a function of the memory latency and data tenure and can be calculated via the following expression:

$$D_{pipe} = \left\lfloor \frac{(T_{access}F_{bus}) + \left(\frac{8B_{data}}{W_{bus}}\right) + 1}{\left(\frac{8B_{data}}{W_{bus}}\right) + 1} \right\rfloor \quad (41)$$

From equation 41 it can be seen that the optimal pipeline depth increases with increasing memory latency or decreasing data tenure (due to either smaller data transfers or wider buses). The optimal pipeline depth for a number of bus widths and memory latencies is shown in Table 17.

Deep pipelines are most advantageous for buses with high latency memories (with respect to the bus speed) and fast data transfers (a 32 byte data transfer requires only a single cycle on a 256 bit bus). The peak data rate for an optimally pipelined bus can be computed via a simple relationship between the bus frequency and data tenure by multiplying the ideal bus bandwidth by the fraction of a bus transfer spent

actively transferring data:

$$D_{rate} = \frac{F_{bus} B_{data} (\frac{8B_{data}}{W_{bus}})}{(\frac{8B_{data}}{W_{bus}}) + 1} \tag{42}$$

Using the data from Table 17 as a reference, the peak data rate for a 32 bit bus with a pipeline depth of 2 and a 10 cycle memory latency transferring 32 bytes of data is approximately 711MB/s. By comparison, the peak bandwidth of a 32bit 200MHz bus is 800MB/s. A 256 bit bus with a 1 cycle transfer penalty has a peak data rate of 3.2GB/s or about half the peak bandwidth. The limit set by the peak bandwidth is not reached in either case because of the null cycle terminating each data tenure.

For pipelined buses that have a depth less than the optimal value determined via equation 41 the peak data rate must be determined by evaluating the penalty paid for sub-optimal pipelining. This is clearly illustrated in the example of Figure 50b with a pipeline depth of one. Because the third transaction could not initiate until the first completed, it issued later and could not hide its latency with the data tenures of the previous transactions. This results in a 5 cycle penalty. This pattern repeats for every two transactions, so the average penalty is 2.5 bus cycles per transaction. The formula for computing the average penalty and peak data rate for a sub-optimal pipeline is:

$$P_{pipe} = \frac{T_{access} F_{bus} + (\frac{8B_{data}}{W_{bus}}) + 2 - (D_{pipe} + 1)(\frac{8B_{data}}{W_{bus}}) + 1)}{D_{pipe} + 1} \tag{43}$$

$$D_{rate} = \frac{F_{bus} B_{data}}{(\frac{8B_{data}}{W_{bus}}) + 1 + P_{pipe}} \tag{44}$$

The effect of sub-optimal pipelining on the peak data rate of the bus is shown in Table 18. The worst case is the non-pipelined bus with a peak data rate of 2.5GB/s. Allowing just two pending transactions nearly doubles the peak data rate. Every increase in the pipeline depth significantly increases the performance of the bus. This is particularly true for data streaming operations that pass large amounts of data between cores. The fully pipelined bus will be nearly eight times as fast.

Table 18: The transaction penalty (in bus cycles) and peak data rate for a 1 GHz, 256 bit bus transferring 32 Byte lines from memory with a 10 cycle access penalty.

Optimal pipeline depth = 6		
Depth	Penalty	Data Rate (GB/sec)
0	11	2.5
1	4.5	4.9
2	2.33	7.4
3	1.25	9.8
4	0.6	12.3
5	0.17	14.7
6	0	16.0

4.3.2 Transaction rate limit

The maximum number of bus transactions per processor clock cycle is determined in a manner similar to the peak data rate from equation 42. This value is the maximum rate at which the processor core and other peripherals can issue bus requests without incurring additional penalties due to contention for bus resources. The bus transaction limit for an arbitrarily pipelined bus is defined below:

$$T_{limit} = \frac{D_{rate}}{B_{data} F_c} \tag{45}$$

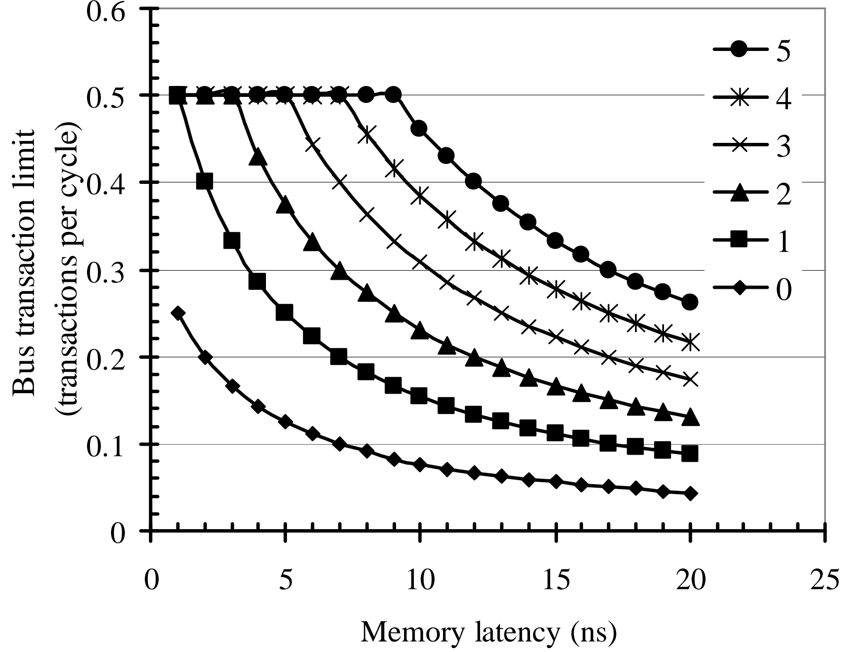

Figure 51: The bus transaction limit (transactions per cycle) for a bus with $F_c = F_b$ = 1GHz, B_{data} = 32 Bytes, and W_{bus} = 256 bits. The memory latency is varied from 1 to 20 cycles for pipeline depths ranging from 0 (non-pipelined) to 5.

Figure 51 illustrates the dependence of the transaction rate limit on memory latency and pipeline depth. Clearly, the pipelined buses show a considerably higher rate limit for increasing pipeline depth up to the optimal value of 0.5 transactions per cycle. The buses with deeper pipelining are able to maintain the optimal data rate for higher memory latencies.

There is assumed to be no additional penalty for bus access if the total transaction rate is less than the maximum set by the rate limit. However, if the total bus loading from the processor core and peripherals exceeds the transaction rate limit, then an average penalty per access is assessed in the following manner:

$$P_{rate} = \frac{T_{rate}}{T_{limit}} \tag{46}$$

The value of the rate penalty,P_{rate}, is assumed to be 1 if required transaction rate, T_{rate}, is less than T_{limit}. For example, if the required transaction rate is twice the limit, then every bus request made by the processor core must wait an additional bus penalty before it may issue. Substituting the various components for bus access penalty into equation 40 yields an expression for the relative system throughput.

$$TP_{sys} = F_c\Big(CPI_{base} + M_{rate}\Big[P_{rate}\Big(\frac{8B_{data}}{W_{bus}} + \frac{B_{word}}{W_{bus}}\Big)\Big\lceil\frac{F_c}{F_{bus}}\Big\rceil + \lceil T_{access}F_c\rceil\Big]\Big) \tag{47}$$

The overall transaction rate generated by the processor core is dependent primarily upon the effective miss rate, M_{rate} from the processor local memory, the bus/access penalty, and the relative speed of the processor core with respect to the bus operating frequency. The transaction rate produced by the processor core is self-limiting. Because bus accesses contribute to processor stall cycles, an increased transaction rate results in a greater number of clock cycles between transactions. The effective transaction rate for the processor core to memory arising from misses in the processor local cache is estimated by dividing the miss rate (effective rate at which the processor issues read transactions) by the average number of cycles per instruction/operation) as in equation 48.

$$T_{rate} = \Big(\frac{CPI_{ave}}{M_{rate}}\Big)^{-1} \tag{48}$$

The effect of bus transaction rate limits on the processor-memory transaction rate with respect to increasing miss rate is illustrated in Figure 52. The non-pipelined bus reaches the limit of 1 transaction every 4 processor cycles at a miss rate of about 50 percent. Beyond that value the saturation penalty from equation 46 is assessed

resulting in increased CPI. The transaction rate limit for the pipelined bus is considerably higher (1 transaction every 3 processor cycles) such that the transaction rate continues to increase permitting a higher total system throughput. It should be noted however, that the miss rates for which this effect is felt are very high. The expected miss rate from a 32KB instruction or data cache averages at 2%, so the bus transaction rate should remains below the limit. In this case, the primary benefit of the the bus pipelining is to increase the bus resources available for communication with peripheral components. Figure 53 shows a similar analysis with a fixed miss rate and varying processor clock frequency. Unlike the previous case where the transaction rate limit was static with respect to the miss rate, the limit is directly proportional to the processor cycle time as in equation 45. The transaction rate from the processor core is limited by the increasing penalty for bus accesses as the gap between the bus and processor speed increases. The transaction rate asymptotically approaches the limit for the non-pipelined case. This result implies that for systems with a large mismatch between the core speed and bus speed the processor core may monopolize the system bus severely reduce the bus resources available to the peripheral cores.

The effect of finite bus resources on throughput is illustrated in Figure 54. The key system and bus metrics used to generate this plot are listed in Table 19. The base CPI of 0.5 is the ideal value for the example SoC from Section 4.4.2. The average throughput for each bus width at 1GHz is approximately 1.6BIPS (billions of instructions/operations per second). For smaller bus sizes (32bits) the throughput quickly saturates at a value of approximately 4BIPS as the processor and bus speeds diverge. The smaller bus transfers data slowly and the bus transfer penalty becomes the dominant contributor to the increase in the processor CPI. The larger bus sizes resist saturation much more effectively, but still begin to lag noticibly at higher clock frequencies. The 256bit bus experiences a nearly ideal speed up of 1.81 as the clock

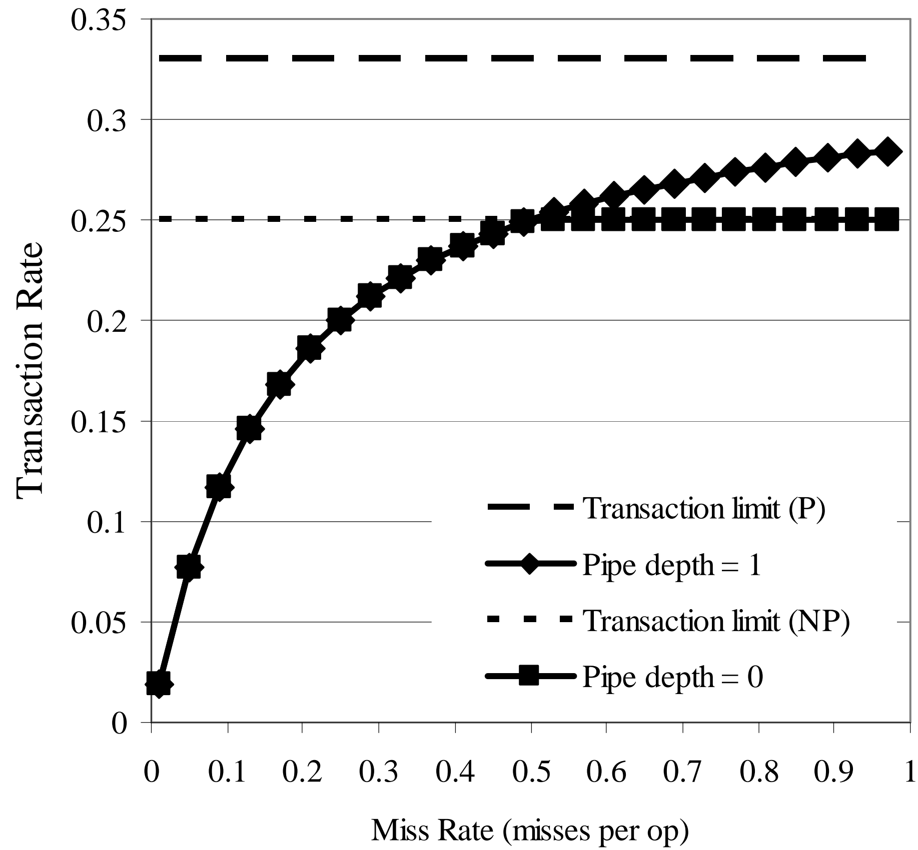

Figure 52: Bus transaction rates vs miss rate for pipelined (depth = 1) and non pipelined (depth = 0) buses compared to the transaction rate limits for each type of bus.

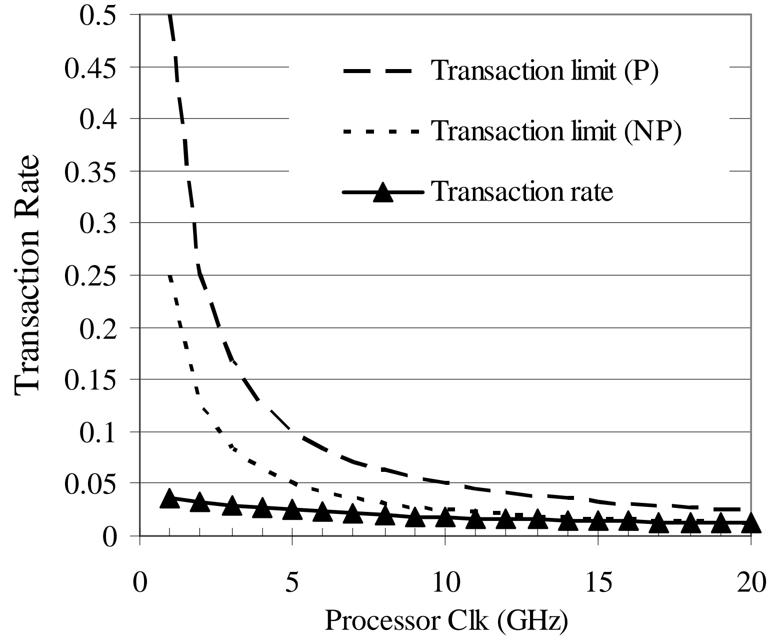

Figure 53: Bus transaction rates vs processor core frequency compared to the transaction rate limits for each type of bus (P=pipelined, NP = non-pipelined). The miss rate, memory access time, bus speed, and bus width are 2%, 1ns, 1GHz, and 256 bits respectively

frequency doubles from 1 to 2 GHz, but the speed up from 10 to 20 GHz is only 1.24.

In order for multi-GHz systems to properly leverage the advantages offered by continued technology scaling, the performance of the on-chip bus must keep pace with the performance of the system cores. There are numerous potential solutions for the specific SoC design space explored in this section. Obviously, increasing the bus width and frequency to minimize the penalty incurred for each bus transaction is a straightforward approach. Additionally, the processor local cache size may be increased to reduce the effective miss rate. Another approach is to arrange the larger memory such that the processor core has direct access for handling memory transactions without putting the traffic on the shared bus. The ideal throughput for a 32 bit system with a base CPI of 0.5 operating at 20 GHz is 40BIPS. An ideal bus design for this system has a 256 bit width (equal to the size of the data transfers)

Table 19: Parameters for bus throughput analysis

Parameter	**Value**
F_{bus}	1GHz
B_{data}	32bytes
T_{access}	1ns
Cache	32KB I/D
M_{rate}	0.02
CPI_{base}	0.5

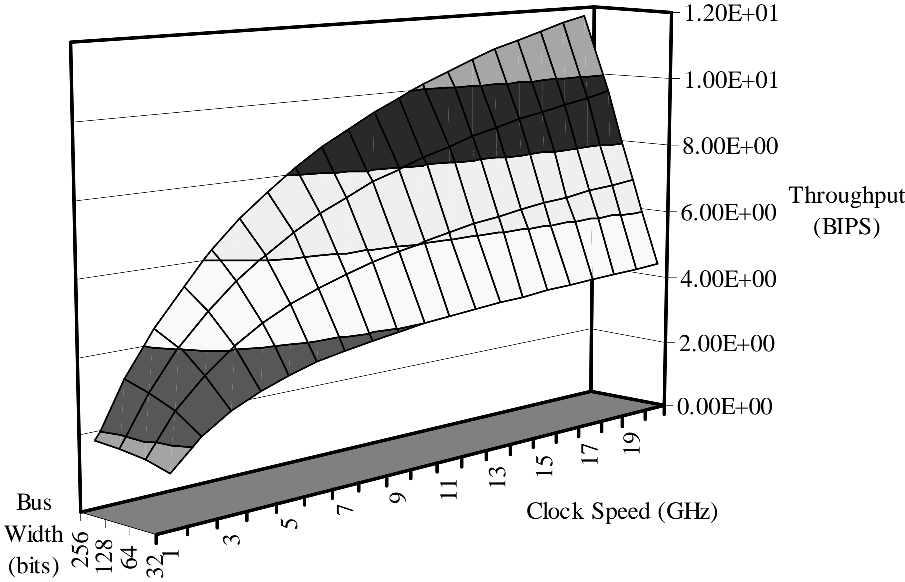

Figure 54: Bus limited throughput for increasing processor clock speed (1-20 GHz) and bus width (32-256bits). The colored bands indicate the regions between z-axis gridlines as projected onto the surface.

operating at the core clock frequency of 20 GHz. If a 4 cycle penalty is assessed for memory access at a miss rate of 0.02, the peak throughput is approximately 32BIPS. Clearly, the performance of the on-chip bus is a key component in determining the overall system performance for a gigascale SoC. Aggressive design practices for these on-chip buses for gigascale SoCs will be required to realize the performance potential inherent in the underlying technology.

4.4 CoreConnect Chip Example

In this section, the hierarchical block methodology introduced in Chapter 3 is applied to an existing implementation of a bus-based SoC, the IBM 440GP. The simulation results for both the processor core and chip are compared with actual data. The simulated bus statistics are given for both operation at specified frequency and at maximum potential bus frequency. Finally, GENESYS 2004 is exercised to determine the performance scaling of this design for various technology implementations.

4.4.1 The IBM CoreConnect Architecture

The IBM CoreConnect bus architecture is a commercially available solution for developing SoC systems utilizing on-chip buses for communication between component cores [39]. The CoreConnect bus architecture utilizes three on-chip buses: the processor local bus (PLB), on-chip peripheral bus (OPB), and the device control register (DCR) bus to exchange data between the core processor and peripheral components. The primary components of the CoreConnect architecture are described below:

- **PLB:** The processor local bus provides the high performance communications pathway between an embedded processor core and peripherals. The PLB has separate read and write buses with widths of 32, 64, and 128 bits. The bus operates at frequencies of 66, 133, and 183MHz.

- **OPB:** The on-chip peripheral bus is designed to alleviate performance bottlenecks by providing a secondary bus for connecting additional peripherals to the processor core and memory via a bridge unit connecting the OPB and PLB. The OPB provides separate 32 bit read and write buses operating at a frequency of 50MHz.

- **DCR:** The device control register bus is a low performance bus designed for communicating system configuration data between the processor core and PLB slaves. The DCR bus provides 10 address and 32 data lines.

4.4.2 440GP Simulation results

Figure 55 illustrates the block diagram for an example SoC implementation based upon the IBM 440GP chip [43]. The system consists of a PowerPC440 processor core connected with various peripherals via the CoreConnect bus architecture. The embedded processor core is composed of the PPC440 microprocessor and two 32KB instruction and data caches. The example of Figure 55 is implemented in a $0.18\mu m$ process utilizing 4 levels of wiring at a supply voltage of 1.8V. GENESYS simulations for the 440 processor core (PPC440 CPU in Figure 55) is listed in Table 20 against actual data. These results indicate that block SoC methodology produces relatively close agreement with the actual data for the processor core. The homogeneous system model estimates the total local interconnect demand at 72 meters of wiring as opposed to 11 meters for the heterogeneous SoC model (IBM does not provide data for the wiring demand). The higher estimation of wiring demand for the homogeneous

system model results in the increased die size, greater power dissipation and slower clock speed as seen in Table 20. The area estimate from the SoC methodology is double the indicated $4mm^2$ from, however, this data was listed as estimated by IBM, so a precise comparison of accuracy is not possible.

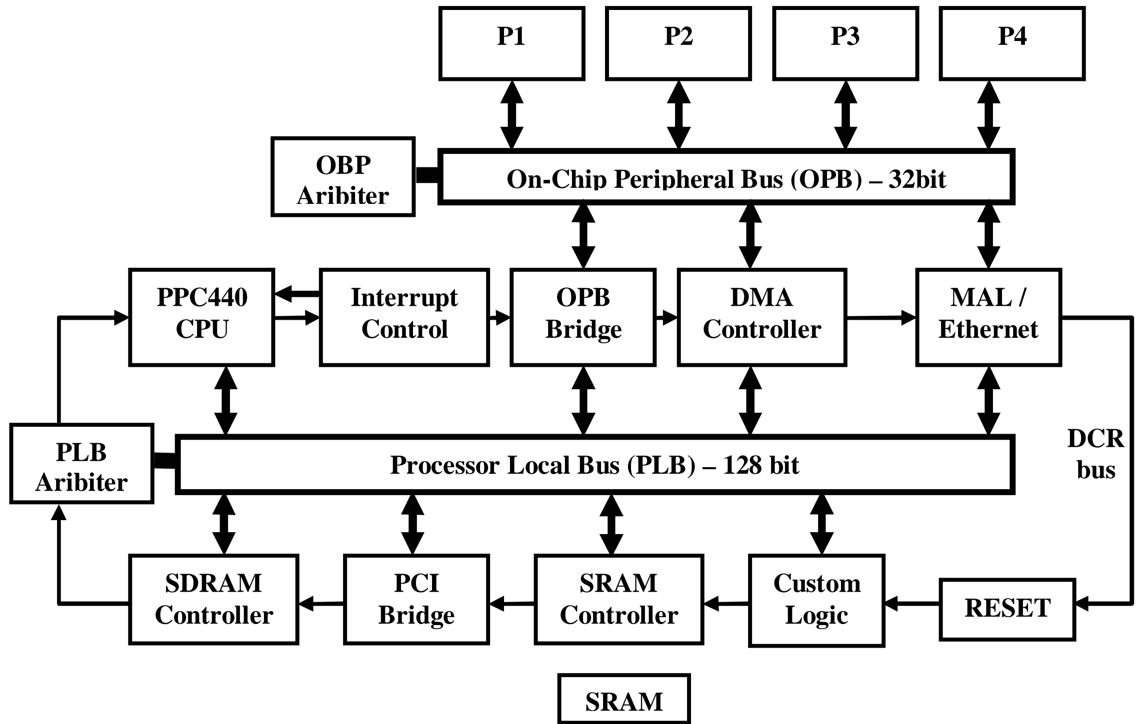

Figure 55: A block diagram for an example SoC implementation based on the IBM 440GP using the CoreConnect bus architecture.

Simulation results for the complete SoC illustrated in Figure 55 is simulated via GENESYS and listed against actual performance data in Table 21. The simulated area, frequency, and power dissipation for the example system are shown to be in good agreement with the empirical data. This application of the GENESYS SoC modeling methodology to a non-traditional microprocessor design is indicative of the flexibility afforded by the block modeling methodology introduced in Chapter 3. The uni-processor system model supported by the earlier generation tool again overestimates the wiring demand (280 meters) when compared to the block methodology (66

meters for local and global interconnect).

The bus statistics for the PLB of the simulated 440GP SoC are listed in Table 22. The bus operating frequencies are held to the specified frequencies permitted by the CoreConnect specification. The expected average power dissipation for the $15mm$ bus length increases by a factor of 10 as the number of bit lines and bus frequency are increased.

Many current on-chip bus standards are limited to operating at frequencies well below the core processor speed. If the bus is permitted to operate at the maximum frequency permitted by the delay of the bus line, the bus frequency increases to about 1.5GHz with a peak bandwidth ranging from nearly 50Gbps for the 32 bit bus to nearly 200Gbps with a 128 bit bus. The price in increased power dissipation is $217mW$ and $885mW$ respectively. The 1.5GHz frequency exceeds the core processor clock by a factor of three. Theoretically, this would permit up to three bus transactions per bus for every one clock cycle. This indicates that the potential for exploiting the increases in bandwidth afforded by on-chip integration is substantial.

4.4.3 Performance scaling and trends for the 2004 ITRS technologies

A key feature of the GENESYS 2003 simulator is the ability to rapidly asses the impact of technology changes on key performance metrics of integrated systems. The SoC of Figure 55 is simulated for the technology implementations corresponding to selected ITRS [3] technology generations. All system and circuit level parameters remained unchanged with the exception of interconnect dimensions which are assumed to scale with the minimum feature size such that the first two interconnect levels are routed at minimum pitch ($2F$) and the upper tier at 2x minimum pitch to reflect

typical interconnect scaling (longer wires have larger dimensions) [33]. The bus line dimensions are set to 10x minimum pitch for consistency with previous simulation results. The results for the critical system performance metrics are listed in Tables 23 and 24.

The projected performance of the simulated IBM 440GP increases rapidly with each advancing technology implementation. The maximum system clock frequency and average power dissipation are plotted in Figure 56. The clock frequency increases steadily, but the power dissipation saturates at a value of close to 7 Watts. The decrease in feature size between technology generations results in a decrease in the total power dissipation for the 2003 to 2005 nodes, but beyond 2005 the static power (leakage) increases by three orders of magnitude as the oxide thickness reduces from 1.8nm to 1.3nm. Beyond 2007, the oxide thickness stabilizes and increases in the gate leakage current density are offset by reduced gate area. Therefore, as the dynamic power continues to decrease, the static power increases at a rate sufficient to offset the improvement in dynamic power. A power dissipation of only 7W does not seem problematic until the resulting power density is taken into account as illustrated in Figure 57. The power density for the example SoC exceeds a reasonable limit of $100W/cm^2$ beyond the 2005 time frame. The expected average power dissipation for the PLB is seen to decline monotonically for each successive generation despite a five fold increase in the maximum operating frequency. This effect follows from two key effects: a reduction in the bus length as the chip area scales down, and advances in low-k dielectrics for reducing line capacitance.

Figure 58 displays a breakdown of the power dissipation into its dynamic and static components. The static contribution percentage of the total system power for the 2003 and 2005 nodes is expected to be less than 10%, but by 2007 this increases

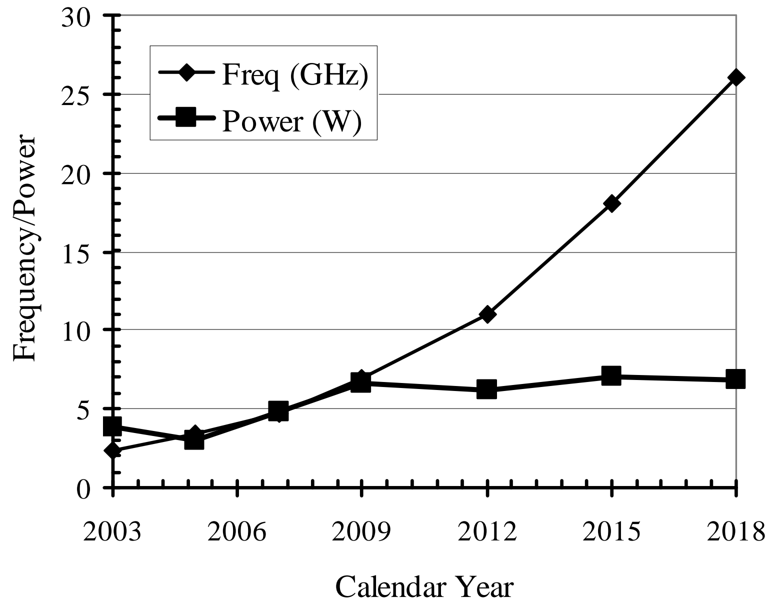

Figure 56: Power dissipation and maximum clock frequency for selected ITRS technology nodes applied to the 440GP example. See Table 23 for feature size information.

sharply and continues until fully 80% of the total power is from leakage. This implies that merely slowing the clock and sacrificing performance is not enough. The majority of the leakage current is electron and hole tunneling through the gate dielectric [44].

Simulations for the 2018 technology node applied to the example SoC indicate that increasing the effective oxide thickness to 2.1nm (from 0.9) and increasing the effective threshold voltage to 250mV (from 200) results in a total static power dissipation of $5mW$. This approach requires the use device structures resistant to drain induced barrier lowering in order to control sub-threshold leakage current. Additionally, high permitivitty materials may permit an increase in the physical thickness without reducing the gate capacitance. The new total power dissipation is $590mW$ at a die size of $0.16mm^2$. The power density is still too high at $368W/cm^2$. The clock frequency must be reduced to 4.8GHz in order to achieve a power density of $100W/cm^2$. The expected performance for the example is reduced by 80% in order to achieve a realistic power budget. The key result to be taken from this analysis is

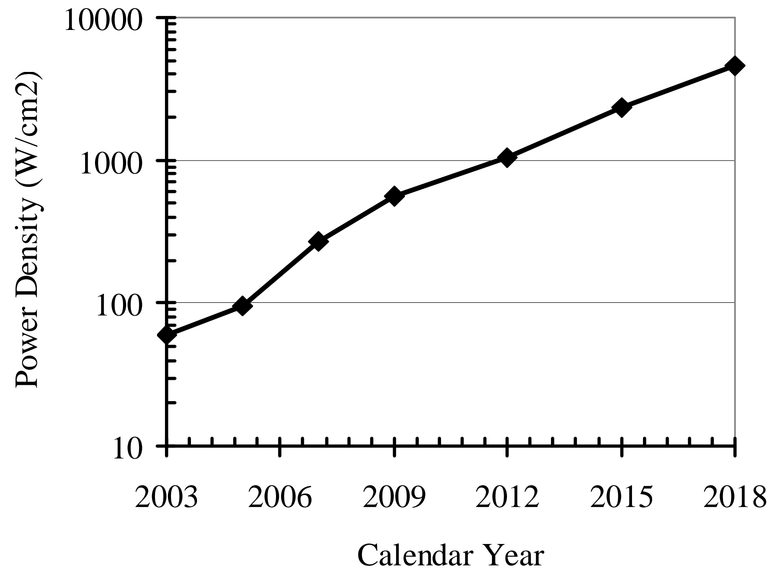

Figure 57: Power density for selected ITRS technology nodes applied to the 440GP example.

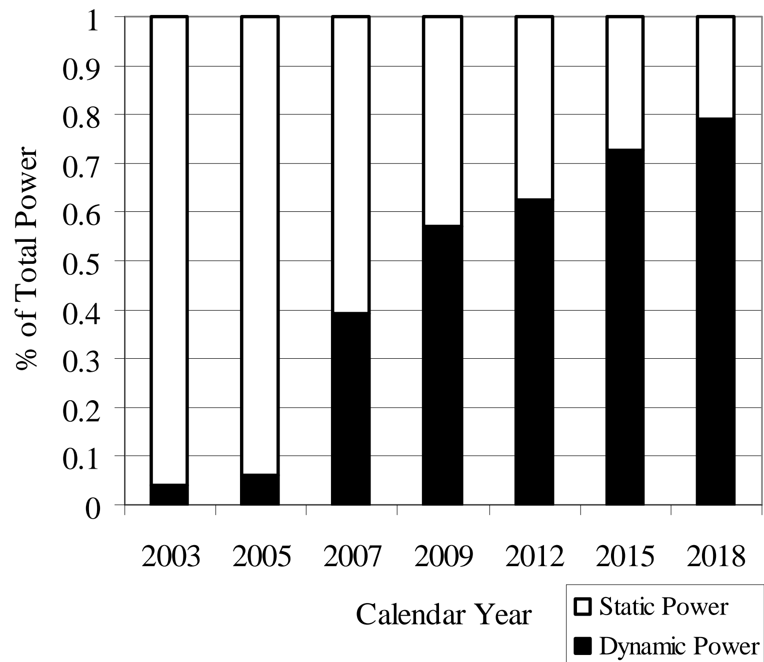

Figure 58: A breakdown of the total power dissipation between the dynamic and static components for the 440GP example.

that beyond the 2007 technology node (max $F_c = 4.7$GHz), there is no appreciable gain in performance due to limitations imposed by power density requirements. In this instance, GENESYS has been utilized to simulated the projected performance of a complex SoC for future technology generates and identify a key limitation on the scaling potential of the system and assess the performance trade-off required to a achieve a practical solution.

4.5 Conclusions

This chapter introduces the modeling of generic on-chip buses for SoC applications via the hierarchical block-based SoC methodology introduced in Chapter 3. The model treats the on-chip bus (or bus segments) as a set of performance critical interconnects requiring definition of the physical (line width/spacing) and logical resources (bus netlist). Three different types of generic bus links are introduced with an examination of the power/delay trade-offs associated with each. Methods for estimating the length, and average power dissipation for the generic bus lines are introduced. An automated cell placement method for optimizing bus bandwidth is introduced and shown to reduce the length of bus lines by up to 30% and increase the bandwidth by approximately 25%. A model for the transaction rate and bus limited throughput was introduced and utilized to explore the effects of bus performance for a processor-core based SoC with an on-chip bus. The throughput analysis confirms that the bus performance is a key target of optimization for maximizing the performance for a subset of bus-based SoC designs. The block-based SoC model was applied to an existing bus-based SoC with a high degree of accuracy in projecting the area, clock speed, and power dissipation when compared to the previous uni-processor system model.

Table 20: Comparison of GENESYS simulation results for the PowerPC440 embedded processor core. Both the block methodology (SoC model) and the homogeneous (uni-processor) system model are utilized for comparison purposes. The core contains 5.5M transistors, approximately 1.5M for logic and 4M for cache. The $4mm^2$ die size listed is only an estimate, but the heterogeneous SoC model produces much closer agreement than does the homogeneous model.

	PPC440	**SoC Model**	**Homogeneous Model**
Area [mm^2]	4.0*	8.0	59.6
Freq. [MHz]	500	502	390
Power [W]	1.25	1.24	6.8

*Estimated by IBM

Table 21: Comparison of GENESYS simulation results for the IBM 440GP SoC implementation comprised of 17 megacells connected via the CoreConnect bus architecture. The simulation results for the uni-processor system model are listed for comparison purposes.

	440GP	**SoC Model**	**Uni-processor Model**
Area [mm^2]	64	59.1	238
Freq. [MHz]	500	502	330
Power [W]	5	5.9	19.6

Table 22: Bus maximum bandwidth, average power dissipation, and area utilization for the PLB for various PLB widths for the IBM 440GP SoC. The bus lines are assumed routed at 10x minimum pitch (W=S=1.8/mum).

	Processor Local Bus		
PLB width	BW [Gbps]	Avg. Pwr [mW]	Area [mm^2]
32bit	2.11	9.56	3.53
64bit	8.5	38.7	7.05
128bit	23.4	108	14.1

f_{bus}=66, 133, and 183MHz for the 32, 64, and 128 bit bus.

Table 23: Technology scaling simulation results for the 440GP example of Figure 55.

440GP CHIP				
Year	F	F_c [GHz]	A_{chip} $[mm^2]$	Pwr [W]
2003	$65nm$	2.3	6.4	3.8
2005	$45nm$	3.4	3.1	3.0
2007	$35nm$	4.8	1.8	4.8
2009	$28nm$	6.9	1.2	6.7
2012	$20nm$	11	0.6	6.23
2015	$14nm$	18	0.3	7.0
2018	$10nm$	26	0.15	6.8

Table 24: Technology scaling simulation results for the 440GP PLB bus illustrated in Figure 55.

PLB BUS				
Year	F	F_{bus} [GHz]	L_{bus} $[mm]$	Pwr [mW]
2003	$65nm$	5.3	5.2	450
2005	$45nm$	7.2	3.6	320
2007	$35nm$	9	2.8	270
2009	$28nm$	11	1.7	230
2012	$20nm$	15	1.6	170
2015	$14nm$	22	1.1	120
2018	$10nm$	27	0.8	70

CHAPTER V

ON-CHIP DRAM MODEL

5.1 Introduction

Since the introduction of on-chip caching with the Intel 80386 processor in 1985, the design space for on-chip memory arrays has been dominated by SRAM. The size of on-chip memory storage has grown considerably in that time, from a mere 16Bytes to 9MB for the Intel Itanium2 microprocessor [45]. For much of this time, the integration of DRAM and logic on-chip was hindered by the difficulties in merging two different process technologies. However, the incorporation of DRAM on-chip has been demonstrated as early as 1996 [46]. As the demand for on-chip memory capacity continues to increase, the incorporation of DRAM on-chip is becoming a common design practice for achieving high storage density [47] [48]. Therefore, this chapter attempts to extend the GENESYS cache modeling framework to incorporate on-chip DRAM.

The key observation concerning the modeling of on-chip memory arrays for SRAM and DRAM is that the surrounding support logic (line drivers, multiplexors, comparators, sense amplifiers...etc) for addressing/accessing the array is the same. The primary difference between the two types of storage is the design of the memory storage cell. The SRAM cache routines developed for GENESYS in [25] exhaustively model the area, power, and delay for each of these components. The approach taken

in this chapter focuses on the characterization of a standard single transistor storage cell and extrapolating the area, power, and delay at the array level.

5.2 DRAM Cell Model

The single transistor DRAM cell forms the basic unit of memory storage in a DRAM array. The cell consists of single transistor and capacitor pair with bit and word line connections. The capacitor forms a simple storage node and the transistor provides access to the charge stored on the node. The cell is accessed via the word line connection to the gate of the transistor and the stored charge is sampled via the bit line connection to the drain of the access transistor. A basic circuit model for this standard DRAM cell is illustrated in Figure 59. The primary characteristics of the cell targeted for modeling are the layout footprint (area), switching and storage energy, and the capacitor discharge rate.

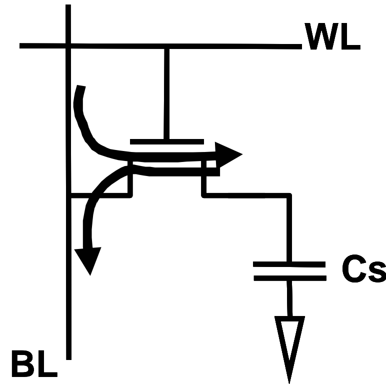

Figure 59: The basic circuit model utilized for the characterization of a simple DRAM storage cell. BL = bit line, WL = word line, and C_s is the storage node capacitance.

5.2.1 Cell area model

Figures 60 through 62 show the historical sizing trends in DRAM cells [49]. The feature size and cell area reduce rapidly with advancing technology generations, but the area in terms of square feature sizes has remained steady at an average of approximately $10F^2$. The cell area model utilized in GENESYS 2004 is based upon the layout footprint for a standard storage cell. The layout for a typical DRAM cell utilizing a trench capacitor as the storage node is illustrated in Figure 63. The total length of the cell layout is simply the sum of the transistor and capacitor dimensions:

$$L_{cell} = L_s + \alpha F + L_d + L_{cap} \tag{49}$$

Where L_s and L_d are the source and drain lengths respectively, α is a factor relating the transistor channel length to the minimum feature size ($\alpha > 1$), F, and L_{cap} is the layout length of the trench capacitor. The channel length of the DRAM access transistor is typically greater than the minimum in order to control leakage from the storage node, therefore, the α parameter is utilized to adjust the drawn channel length. From 49 the bounding area for the individual cell is calculated using the width of the DRAM cell as in the following expression:

$$A_{cell} = L_{cell} W_{cell} \tag{50}$$

The actual cell footprint for a DRAM cell in an array must take into account the design rule specifications of at least 1F spacing between adjacent bit lines and isolation between the storage capacitors of adjacent cells. The modified expression for the cell footprint area is given in equation 51.

$$A_{cellfoot} = (L_{cell} + Lbit_{iso})(W_{cap} + Wcap_{iso}) \tag{51}$$

Where $Lbit_{iso}$ and $Wcap_{iso}$ are the widths of the cell isolation regions between bit lines and storage capacitors. If typical values for the cell dimensions ($L_s = L_d = 1F$, $\gamma = 1.5$, $L_{cap} = 2F$, $W_{cap} = 2F$, and $Wbit_{iso} = Wcap_{iso} = 1F$) are chosen, the respective cell and footprint areas are $11F^2$ and $19.5F^2$ respectively. The $11F^2$ feature size is in agreement with the historical trend illustrated in Figure 62. cell footprint is plotted against the ITRS projections of the area for each of the roadmap years in Figure 64. The results obtained from 51 closely matches the ITRS projections.

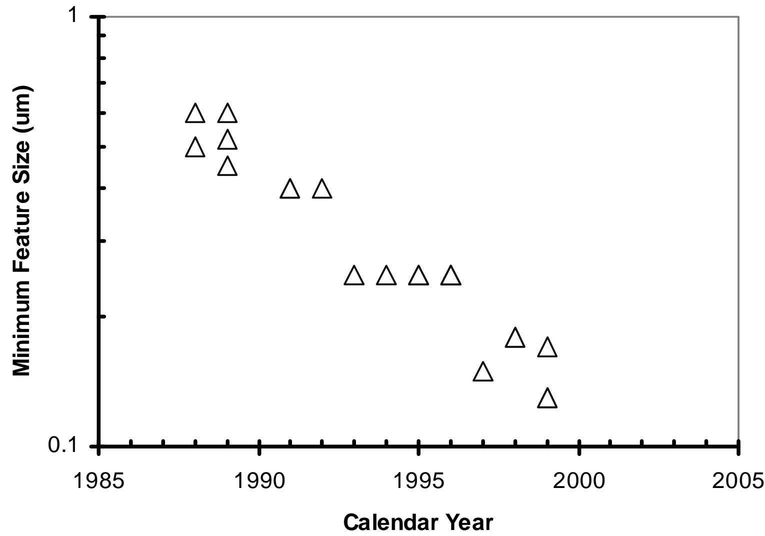

Figure 60: The historical trend for minimum feature size for DRAM technologies vs calendar year.

5.2.2 Cell energy model

The energy/power consumption of the DRAM cell is a function of the device and circuit level technologies (the operating voltage, oxide thickness, device dimensions...etc). The overall cell power is determined by evaluating both the static and

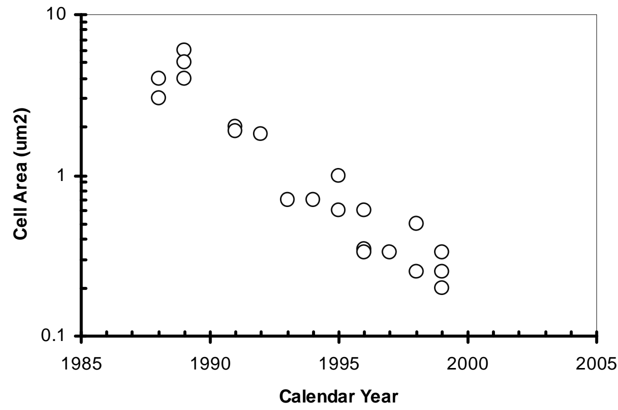

Figure 61: The historical trend for DRAM cell area vs calendar year.

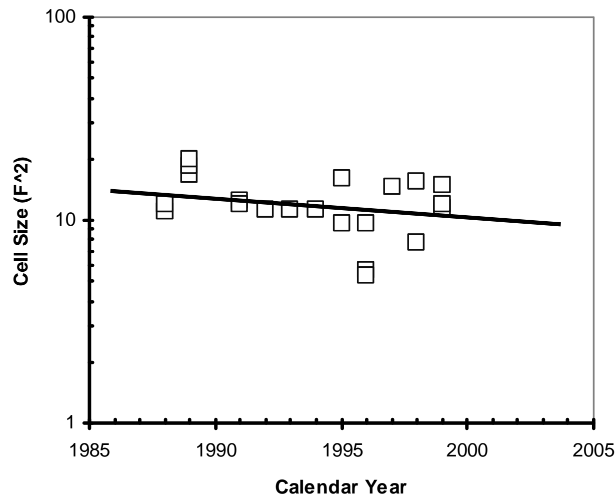

Figure 62: The historical trend for DRAM cell area in units of square feature sizes (F^2) vs calendar year.

dynamic components. The dynamic component is the energy required to operate the DRAM cell while the static power is the average power lost due to leakage. The dynamic energy is only dissipated during switching transitions while the static power is present at all times while charge is stored.

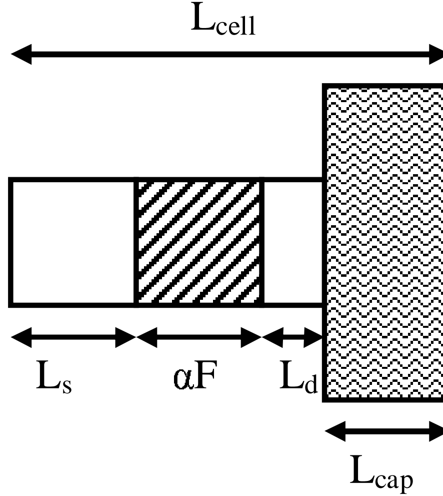

Figure 63: A layout for a generic DRAM cell with trench capacitor. L_s is the length of the source junction, αF is the drawn gate length of the access transistor, L_d is the drain length, and L_{cap} is the length of the trench capacitor. All the lengths are specified in units of the feature size (F).

5.2.2.1 Dynamic energy

The first step in determining the average dynamic power dissipation in a DRAM cell is to define the operating conditions (voltages and procedures for reading/writing data into the cell). The energy model utilized in this chapter is derived under the following assumptions:

- Word lines are charge pumped to a voltage of Vdd + Vth (device threshold voltage) on all cell accesses to ensure that the storage node holds Vdd when a '1' is written.
- Bit lines are precharged to Vdd/2 prior to read operations.
- Writing to the cell: the word line is charged to Vdd+Vth and the bit line (Vdd/0) depending on the value of the data.
- Reading from the cell: the bit line is charged to Vdd/2 and the word line to Vdd+Vth.

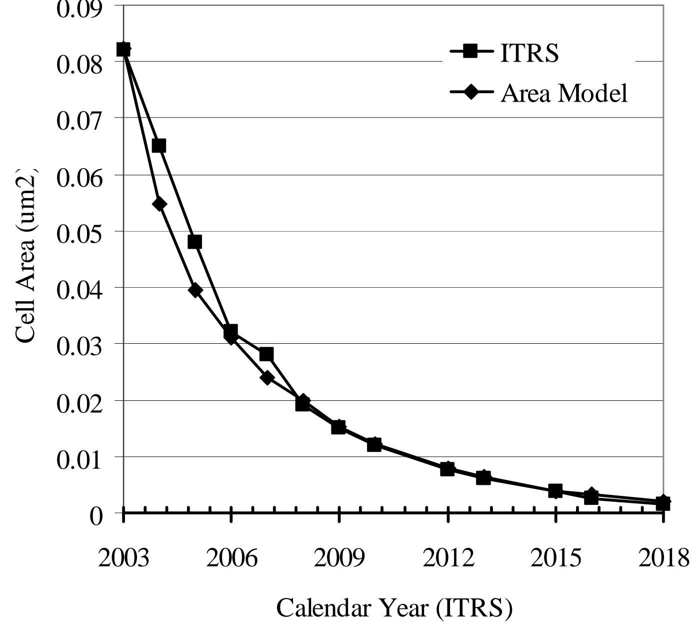

Figure 64: A comparison of the cell footprint model from 51 with ITRS 2003 projections for cell area (μm^2).

- Cell refresh: a read operation is conducted and the result is fed back to the bit line while the word line is held high. An automatic refresh is performed after every read operation.

The various contributors to the cell dynamic energy are the bit and word line capacitances, the gate and junction capacitances of the access transistor, and the storage capacitor. The expressions used to calculated each of these components are given below:

$$Cl_{bit} = C_{line}(W_{cell} + Wcap_{iso}) + C_j L_s W_{cell} \tag{52}$$

$$Cl_{word} = C_{line}(L_{cell} + Lbit_{iso}) + \frac{\epsilon_{ox}}{t_{ox}} \alpha F W_{cell} \tag{53}$$

$$C_s = C_j L_d W_{cell} + \frac{\epsilon_{ox}}{t_{ox}} (2t_d + L_{cap}) W_{cell} \tag{54}$$

Where Cl_{bit} the total bit line capacitance, C_{line} is the line capacitance per unit length, C_j is the junction capacitance per unit area, ϵ_{ox} and t_{ox} are the permitivitty

and thickness of the oxide respectively, and t_d is the depth of the trench capacitor.

Once the cell capacitance is determined, the cell dynamic energy is assessed via the $\frac{1}{2}CV^2$ metric for switching energy. The expressions for the dynamic switching energy for an individual DRAM cell are given below:

$$E_{bl} = \frac{Cl_{bit}{V_{bl}}^2}{2} \tag{55}$$

$$E_{wl} = \frac{Cl_{word}(V_{dd} + V_{th})^2}{2} \tag{56}$$

$$E_s = \frac{C_s {V_s}^2}{2} \tag{57}$$

Where the storage capacitor voltage swing V_s is dependent upon the operation (read/write) and the data value on the storage node. For example, the value of V_s for writing a '1' onto a cell already holding a '1' is 0 (no charge is transfered). Table 25 shows the total voltage swing across the bit and storage capacitor for each different operation and data value. The bit line voltage swing is $V_{dd}/2$ for operations reading or writing a 0, but the energy penalty is incurred twice: once for the precharge operation and when the bit line is discharged to refresh the cell after the destructive read operation.

Utilizing the data in Table 25 the energies for the read and write operations performed on a single DRAM cell are summarized in the following expressions:

$$E_{write00} = \frac{Cl_{bit}{V_{dd}}^2}{4} \tag{58}$$

Table 25: The voltage swing for the bit lines and storage capacitor for each type of operation and data value.

Op	Data	V_{bl}	V_s	V_{wl}
WRITE	$0 \rightarrow 0$	$V_{dd}/2$	0	$V_{dd}+V_{th}$
	$1 \rightarrow 1$	V_{dd}	0	$V_{dd}+V_{th}$
	$1 \leftrightarrow 0$	V_{dd}	V_{dd}	$V_{dd}+V_{th}$
READ	0	$V_{dd}/2$	ΔV_c	$V_{dd}+V_{th}$
	1	V_{dd}	ΔV_c	$V_{dd}+V_{th}$
REFRESH	0	$V_{dd}/2$	ΔV_c	$V_{dd}+V_{th}$
	1	V_{dd}	ΔV_c	$V_{dd}+V_{th}$

$$E_{write11} = \frac{Cl_{bit}{V_{dd}}^2}{2} \tag{59}$$

$$E_{write01} = \frac{Cl_{bit}{V_{dd}}^2}{4} + \frac{C_s{V_{dd}}^2}{2} \tag{60}$$

$$E_{write10} = \frac{Cl_{bit}{V_{dd}}^2}{2} + \frac{C_s{V_{dd}}^2}{2} \tag{61}$$

$$E_{read0} = \frac{Cl_{bit}{V_{dd}}^2}{8} + \frac{Cl_{bit}+C_s}{2}\big(\frac{V_{dd}}{2} - \Delta V_{bl}\big)^2 \tag{62}$$

$$E_{read1} = \frac{Cl_{bit}{V_{dd}}^2}{8} + \frac{Cl_{bit}+C_s}{2}\big(\frac{3}{4}{V_{dd}}^2 - V_{dd}\Delta V - \Delta V^2\big) \tag{63}$$

$$E_{refresh0/1} = E_{read0/1} \tag{64}$$

The value ΔV is the difference in the voltage on the bit line before and after the data in the cell is sampled. This value is calculated below in equation 65. If it is deemed that the data read out of or written into a cell is equally like to be a one or a zero the average dynamic energy for a DRAM cell read and write operation is computed via equations 66 and 67.

$$\Delta V = \left(\frac{Cl_{bit}}{Cl_{bit} + C_s}\right)\frac{V_{dd}}{2} \tag{65}$$

$$E_{write} = \frac{E_{write00} + E_{write01} + E_{write10} + E_{write11}}{4} \tag{66}$$

$$E_{read} = \frac{E_{read0} + E_{read1}}{2} \tag{67}$$

The word line energy consumption E_{wl} does not change with respect to the operational state of the cell and is considered separately from the read and write average power when calculated the total array power dissipation.

5.2.2.2 Static power dissipation

The previous section dealt with determining the dynamic power dissipation for read and write operations for a single DRAM cell. In this section the static power drain for a DRAM cell under several conditions is calculated. The leakage current requires that every DRAM cell in the array be periodically refreshed. The refresh period (time between updates) is governed primarily by the magnitude of the leakage

Table 26: The leakage state of a DRAM cell with regards to the storage value and bit line bias. 1 means that there is an applied bias to the bit line or a '1' stored on the capacitor, 0 means no voltage or no charge on the capacitor.

Storage	BL	Tunneling	Subthreshold
1	1	YES	NO
	0	YES	YES
0	1	NO	YES
	0	NO	NO

current and the size of the storage capacitor. A high leakage current and small capacitor requires a short refresh period which negatively impacts the dynamic power consumption. This is an effect that standard SRAM memories do not have to contend with. Therefore, determining the leakage currents in the DRAM cell is vital to assessing the power dissipation of the entire DRAM array.

The primary sources of leakage considered in this work is the sub-threshold leakage across the source/drain junctions of the access transistor and the tunneling leakage current through the storage capacitor oxide. The cross-sectional diagram of a generic trench capacitor DRAM cell of Figure 65 illustrates the flow of these leakage currents [50]. The leakage currents only flow when there is a potential difference across the source/drain terminals or the storage capacitor plates. Therefore the leakage is dependent upon the state of the DRAM cell. The leakage state of the DRAM cell with respect to the operating state is summarized in Table 26. The state of interest for assessing the impact of the leakage currents on the power and refresh period is the quiescent state when the node is storing a 1. This is assumed to be that state in which the cell spends the most time. All other states either experience no discernible leakage or are transient.

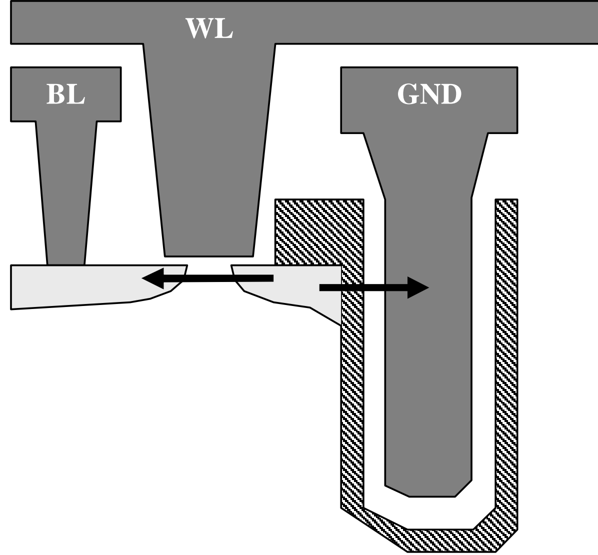

Figure 65: A cross-sectional view of a trench capacitor DRAM cell with the subthreshold leakage and direct tunneling current shown.

The required ΔV_{sense} for sensing a read on the cell sets a limit on minimum storage capacity for the cell. Typically, the sense amps require on the order of a few hundred mV to properly sample the cell [57] [58]. The limit on the storage capacity is formulated in the following expression:

$$C_{smin} = \frac{\Delta V_{sense} Cl_{bit}}{\frac{V_{dd}}{2} - \Delta V_{sense}} \tag{68}$$

This expression, however, leaves no room for error. If any charge leaks off, there will not be enough left on the node to reach $\Delta V_{bl} >= \Delta V_{sense}$. Therefore in order to ensure proper operation the actual storage capacitance should be larger. With a lower bound set on the capacity of the storage cell, the maximum rate at which the voltage decays in the quiescent state is determined via equation 69.

$$\frac{dV}{dt} = \frac{I_{leak}}{C_s} \tag{69}$$

If the maximum voltage decay permitted is ΔV_r, the maximum time between refresh operations, ΔT_r, for the DRAM cell is:

$$\Delta T_r = \frac{\Delta V_r C_s}{I_{leak}} \tag{70}$$

Where the leakage current, I_{leak} is the sum of the sub-threshold and direct tunneling currents. The sub-threshold leakage current through the access transistor can be determined either via the use of a transregional MOSFET drain current model [51] or estimated via the ITRS specifications for the device off current. The direct tunneling current through the capacitor oxide is assessed via a previously derived model that takes into account electron and hole tunneling in both the valence and conduction bands [44]. The tunneling current model is listed in Figure 66.

5.2.3 Cell delay

The complex components of the DRAM critical path delay (decoders, comparator, sense amps) are already modeled extensively with respect to the cache SRAM modeling. Because the bit and word lines connect to the DRAM cell via the source and gate of the access transistor in the same manner as a 6 transistor SRAM cell, the bit line and word line delay models are identical to the existing cache SRAM models in GENESYS [25]. A key advantage of on-chip DRAM is that it can leverage the same high performance transistors and aggressive design techniques such as splitting word and bit lines utilized in SRAM design. Simulation results for the DRAM access time are compared against actual data in the following section.

5.3 DRAM Array model

The DRAM Array model is an extension of the cell models introduced in the previous sections. The total area, power, and delay for the DRAM array is found by extrapolating up from the cell level models.

5.3.1 DRAM Array area

The total DRAM array area is the sum of the individual components such as the address drivers, decoders, sense amps, output drivers, and the active cell area. Figure 67 shows a generic array layout for two cases. Figure 67a shows a single 1Mb DRAM array with an aspect ratio of 1. Figure 67b is the same cache divided into 4 sub-arrays. As in SRAM caches, dividing the array into sub-arrays by splitting the bit and word lines is done to reduce the access time by shortening the length of the lines. The SRAM area model assumes that the 6 transistor cell has a fixed footprint of $250F^2$. The footprint for the DRAM cell as modeled in Section 5.2.1 is an order of magnitude smaller. The total active array area for the on-chip DRAM is:

$$N_{cell}A_{cellfoot} \tag{71}$$

Where N_{cell} is the size of the array in bits. For a feature size of $0.1\mu m$ and a cell footprint of $15F^2$ the projected area for a 1Mb array is $0.157mm^2$ vs $2.62mm^2$ for the equivalent SRAM cell. Table 27 show a comparison of total DRAM array area against an equivalent SRAM for various array sizes. Initially, the array size for both the SRAM and DRAM cells is dominated by the area consumed by the support logic, but as the cache size increases beyond 4KB, the size of the SRAM cell begins to grow linearly with cache size. The DRAM array size begins increasing in size linearly beyond a cache size of 32KB. For a 1MB array the DRAM achieves a relative density of approximately 7X when compared to the equivalent SRAM. This is in agreement

Table 27: A comparison of DRAM array area against cache SRAM area. The implementation technology is $0.13\mu m$.

	Area (mm^2)	
(KB)	SRAM	DRAM
1	0.32	0.21
2	0.41	0.22
4	0.53	0.25
8	0.9	0.3
16	1.5	0.39
32	2.71	0.56
64	5.1	0.93
128	9.9	1.6
256	19	2.9
512	38	5.5
1024	75	11

Table 28: A comparison of estimated DRAM area with reported data from [52] for a $0.18/mum$ technology. C is the cache size in MB, and A is the associativity.

DRAM (C,A)	HPA-RISC-2.0	GENESYS
12MB-6way	$195mm^2$	$180mm^2$
20MB-5way	$117mm^2$	$111mm^2$

with data reported in [52].

5.3.2 DRAM array power dissipation

The power dissipation for an embedded DRAM array is calculated via the expressions for the per cell energy dissipation. The primary differences in the total array power dissipation for a DRAM arise from the requirements for refreshing destructive reads and the periodic refresh. The power dissipation in the active DRAM array (cells only) is calculated by combining the energy costs in Equations 56, 64, 66, and 67 with array geometry parameters.

Table 29: GENESYS simulation of on-chip DRAM compared with embedded macro implementations at 0.18 and 0.13μm technologies. The cores used for comparison purposes are the Samsung LD18 and LD13 eDRAM.

	Metric	Actual	GENESYS 2K4
LD18 - 4Mb	Area (mm^2)	3.87	3.44
	T_{access}(ns)	50	38
	Power (mW)	54	88
LD18 - 4Mb	Area (mm^2)	2.64	2.17
	T_{access}(ns)	50	38
	Power (mW)	36	47

refresh period = 4ms

$$Parray_{ave} = \alpha_f f_c N_{subarray} \Big(\frac{8Bn_{spr}E_{wl}}{n_{dwl}} + \frac{C(E_{read} + E_{write})}{2Bn_{spr}n_{dbl}} \Big) + \frac{8CE_{refresh}}{\Delta T_r} \qquad (72)$$

Where α_f is the memory activity factor, f_c is the DRAM cycle time, B is the set/block size in bytes, n_{spr} is the number of sets per row, n_{dbl} and n_{dwl} are the bit and word line divisions. The number of sub-arrays, $N_{subarray}$, is the product of the word and bit line divisions as illustrated in Figure 67. The above expression assumes that the frequency of reads and writes are equal, so that their energy is weighted equally when averaged. If necessary, separate parameters for the read and write frequency (% of operations that are reads/writes). A typical value for the activity factor of random logic is approximately 0.1, but accesses to memory may occur at a much higher frequency. Therefore, the recommended activity factor for a memory array is given a range of 0.4 to 0.5 [25]. The cost for the periodic refresh is determined by averaging the energy required to refresh all bits in the array over a period of Δt_r. The results of simulations for all three components of the on-chip DRAM modeling (power, area, access time) are compared with an actual embedded DRAM macro in Table 29.

5.4 A Gigabit on-chip DRAM

Utilizing the modeling developed in the previous sections of this chapter, the performance for a 1Gb on-chip DRAM is projected for selected ITRS technology generations. The simulated DRAM is a 128MB, 8-way set associative array with 1024 byte blocks. The total area, power dissipation, and DRAM cycle time for each ITRS generation are collected in Table 30. The projected area for a 128MB DRAM implemented in 65nm technology is approximately $110mm^2$. If the $0.18\mu m$ 4Mb DRAM used from Table 29 is scaled down to 65nm and adjusted for the difference in capacity, the resulting area is $120mm^2$, a value close to the GENESYS projection. The power dissipation remains relatively steady across the roadmap. This implies that the power reducing advantages of small feature sizes are not enough to overcome the increased power dissipation from the lower cache cycle time afforded by the reduction in the access time. From these results it can be seen that the cache power is proportional to the area covered by the cache and inversely proportional to the access time. If the array area and the access time are halved there is little change in the overall dynamic power dissipation. The use of low-K materials results in some reduction in the dynamic power from the 2003 to the 2012 node. Beyond 2012 increasing static power dissipation results in an increase in the total power dissipation. Just as in the scaling case for the 440GP example from Chapter 4, the power density rises rapidly increasing from an already high 227W/cm^2 for the 65nm node to over 1KW/cm^2 at the 2018 technology node.

5.5 Conclusions

In this chapter, a DRAM model for on-chip memories is developed for use with the GENESYS 2K4 SoC modeling methodology. The modeling approach focuses upon the characterization of the individual DRAM cell and leverages pre-existing models

Table 30: GENESYS 2K4 simulation results for a 1Gb embedded DRAM

C = 128MB, B = 1024, A = 8						
Year	2003	2006	2009	2012	2015	2018
F_{min} (nm)	65	40	28	20	14	10
Area (mm^2)	110	42	20.3	10.3	5.0	2.6
Pwr (W)	25	22	18	17	21	24
T_c (ns)	1450	510	220	100	50	20

for the logic components and driver circuits of the memory array. The modeling for the DRAM array is physically based upon the properties of a trench capacitor single transistor DRAM cell. The model takes into account numerous aspects of the DRAM cell topology to calculate a cell area that closely matches the historical trends for cell sizing. The energy models introduced for estimating the power dissipation in a DRAM array are derived from the physical properties of the cell layout and the DRAM read/write procedures. Simulation results for the cell area show significant reductions in the silicon usage for larger array sizes when compared to a cache SRAM. The area, access time, and power dissipation are shown to be in good agreement with an existing embedded DRAM macro. The on-chip DRAM model is then applied to a theoretical 1Gb embedded DRAM showing significant improvement in the area and access time for each successive ITRS technology node. The results for the power dissipation indicate that increasing power density may be a key barrier to taking full advantage of the opportunities afforded by technology scaling for large on-chip memories.

$$J_n = \frac{q^3}{8\pi h \phi_b \varepsilon_{ox}} \cdot C(V_g, V_{ox}, T_{ox}, \phi_b) EXP\left(\frac{-8\pi\sqrt{2m_{ox}}\phi_b^{3/2}\left[1-\left(1-\frac{|V_{ox}|}{\phi_b}\right)^{3/2}\right]}{3hq|E_{ox}|} \right) \quad \text{(a)}$$

$$C(V_g, V_{ox}, T_{ox}, \phi_b) = EXP\left[\frac{20}{\phi_b}\left(\frac{|V_{ox}|-\phi_b}{\phi_{bo}}+1\right)^{\alpha}\left(1-\frac{|V_{ox}|}{\phi_b}\right)\right]\frac{V_g}{T_{ox}}N \quad \text{(b)}$$

$$N = \frac{\varepsilon_{ox}}{T_{ox}}\left\{ n_{inv} v_t \ln\left(1+EXP\left[\frac{(V_{ge}-V_{th})}{n_{inv} v_t}\right]\right) + n_{acc} v_t \ln\left(1+EXP\left[\frac{-(V_g - V_{FB})}{n_{acc} v_t}\right]\right)\right\} \quad \text{(c)}$$

$$V_{ge} + 2\phi_f + q\varepsilon_{si} N_{poly} T_{ox}^2 \left(\sqrt{1+\frac{2\varepsilon_{ox}^2 (V_g - V_{FB} - 2\phi_f)}{q\varepsilon_{si} N_{poly} T_{ox}^2}} - 1\right) \quad \text{(d)}$$

$$V_{ox} = V_{ge} - \phi_s - V_{\mathrm{FB}} \quad \text{(e)}$$

$$\phi_s = \left[\frac{\gamma}{2}\left(\sqrt{\frac{1+4(V_g - V_{ge} - V_{FB})}{\gamma^2}} - 1\right)\right]^2 \quad \text{(f)}$$

$$\gamma = \frac{\sqrt{2\varepsilon_{si} q N_{sub}}}{C_{ox}} \quad \text{(g)}$$

Figure 66: The tunneling current model developed in [44] for tunneling through ultra-thin oxides.

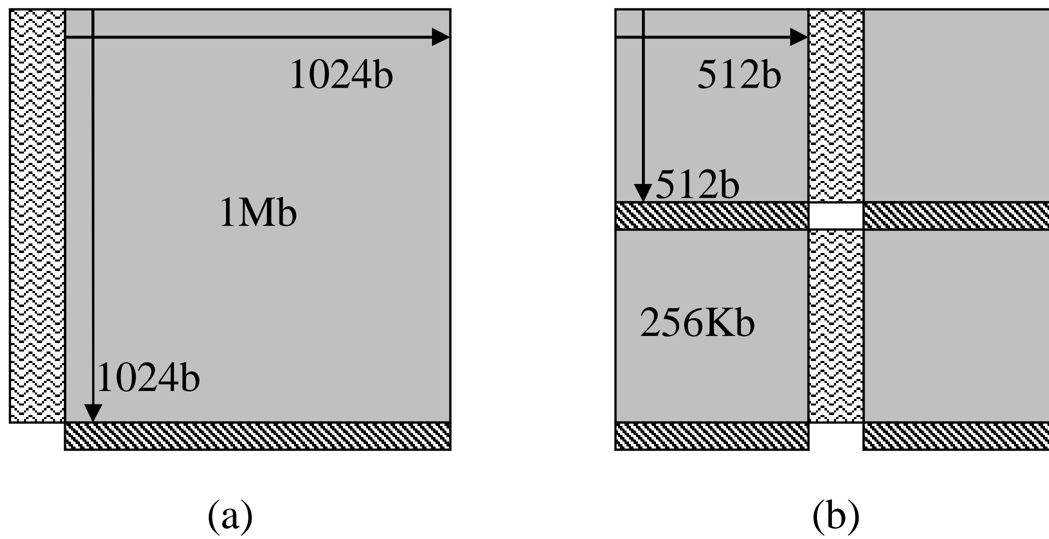

Figure 67: A generic layout for a 1Mb DRAM array for: (a) 1 sub-array, and (b) 4 256Kb sub-arrays.

CHAPTER VI

GIGASCALE CELLULAR ARRAY ARCHITECTURES

6.1 Introduction

The previous chapters developed a modeling framework for projecting the performance of SoC based designs, taking into account core-IP design methodologies, on-chip buses, and embedded DRAM. In this chapter, the system model is utilized to explore the design space for a gigascale SoC based on a cellular array architecture. The generic system type under evaluation is illustrated in Figure 68. The system consists of a large array of interconnected processing elements, each possessing their own local memory. The system performance is evaluated for several interconnect schemes: a stochastic global net list, a single shared bus and fully systolic network of local buses. ITRS projections for future technology generations are utilized to forecast operating frequencies and power dissipation for this system.

6.2 Stochastic global network for gigascale array processors

The primary assumption made for this analysis is that the stochastic global netlist discussed in Section 3.3 applies to the system architecture of Figure 68. Therefore, the wiring distribution is a series of multi-terminal nets of varying fanouts and

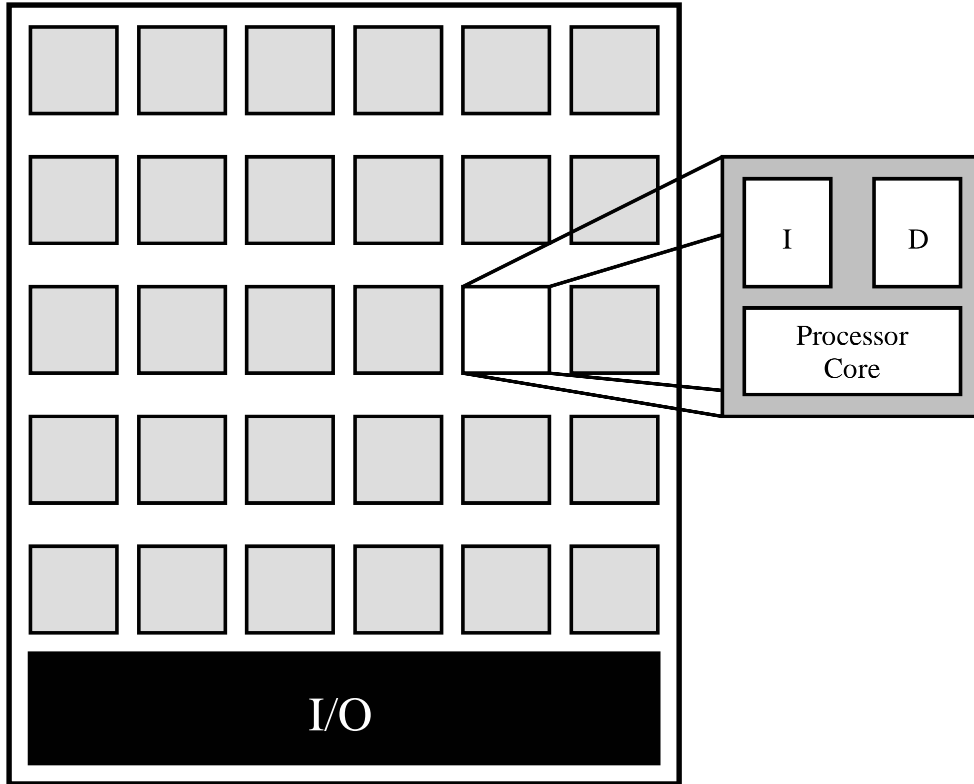

Figure 68: The generic floorplan for a cellular array of processors with local memory.

lengths connecting clusters of chips together.

6.2.1 Simulation methodology

The block modeling methodology implemented in GENESYS 2K4 is ideal for describing an array of identical cells because it is only necessary to define the cell characteristics once and they can be instantiated in the system entity as many times as needed, enabling a rapid exploration of the design space for the generic processor array. The key parameter for characterizing the system of Figure 68 is the number of processing nodes comprising the array. Simulations were run for selected ITRS nodes and the number of nodes is varied from 4 to 128. The key assumptions made regarding the simulation of the array are listed:

- The ITRS projects that nearly 80% of on-chip devices will be devoted to memory storage, the chip array does not contain any large memory cells (only local memory), so the total chip transistor count for each technology generation is: $Xtr_{chip} = 0.8Xtr_{itrs}$. The data values for each ITRS node are summarized in Appendix A.

- The chip area and transistor count remain constant as the number of nodes is increased. This means that the number of transistors per node declines as the node count increases. The approximate transistor count for each node is: $Xtr_{node} = Xtr_{chip}/N_{node}$. This is illustrated in Figure 69.

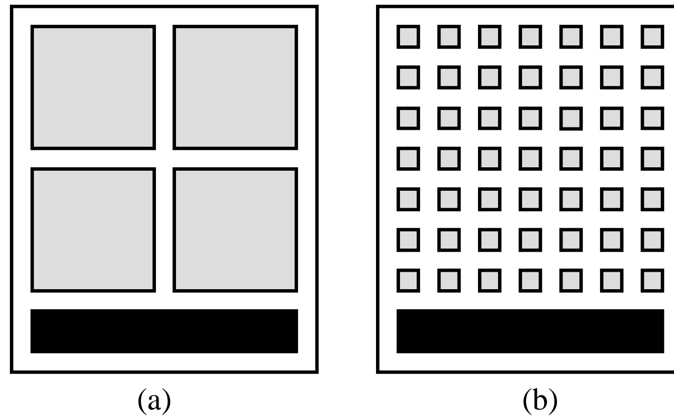

Figure 69: A block diagram of the generic architecture from Figure 68 for various cell counts: (a) four large cells, and (b) numerous smaller cells.

The minimum feature size data for each of the following figures utilizing the calendar year on an axis is summarized in Table 31.

6.2.2 Global and local interconnect resources

The total length of interconnect routed on chip is divided between local interconnects within the individual nodes and global interconnects between nodes. The trends for local and global wiring resources are plotted in Figures 70 and 71.

Table 31: Minimum feature size for the 2003 ITRS technology generations adapted from Table A.

.

2003	65nm
2006	40nm
2009	28nm
2012	20nm
2015	14nm
2018	10nm

The local interconnect demand increases sharply with increasing transistor count and shows a weak dependence on the cell count. The increase is due to both the increasing number of transistors and the reduction in gate pitch as the minimum feature size is reduced. As the node count increases and each node is reduced in size the average interconnect length also shortens. This means that a large number of small cells results in reduced wiring demand for any given technology generation.

Unlike the local wiring demand, the global interconnect length increases with both the gate and node count. The increase in the gate count from one technology node to the next requires more interconnects between any pair of cells to satisfy the stochastic distribution. Additionally, the increase in the cell count also requires more global interconnects to fully wire the chip. The net effect is that the global interconnect resources become more constricted with increasing system complexity (where system complexity is defined as proportional to the number of nodes and total gate count). This effect is illustrated in Figure 72. As the system complexity increases the global interconnect pitch is reduced by a factor of 5. Unlike the local interconnect distribution, there is no reduction in the interconnect length to offset the smaller wiring dimensions. Additionally, the high fanout nodes are penalized with increasing length due to increasing fanout.

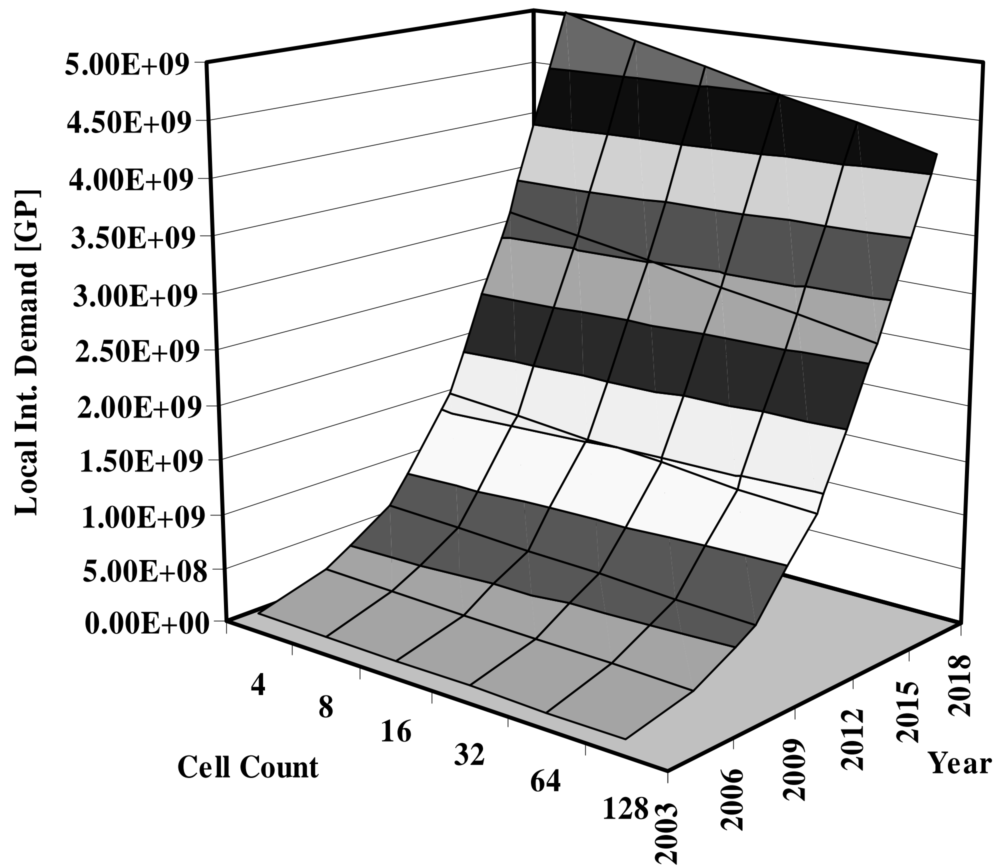

Figure 70: The total length (in terms of the gate pitch) of interconnect within nodes vs the node count and technology generation. The colored bands represent the regions between z-axis grid lines as projected onto the surface. Minimum feature size information is listed in Table 31.

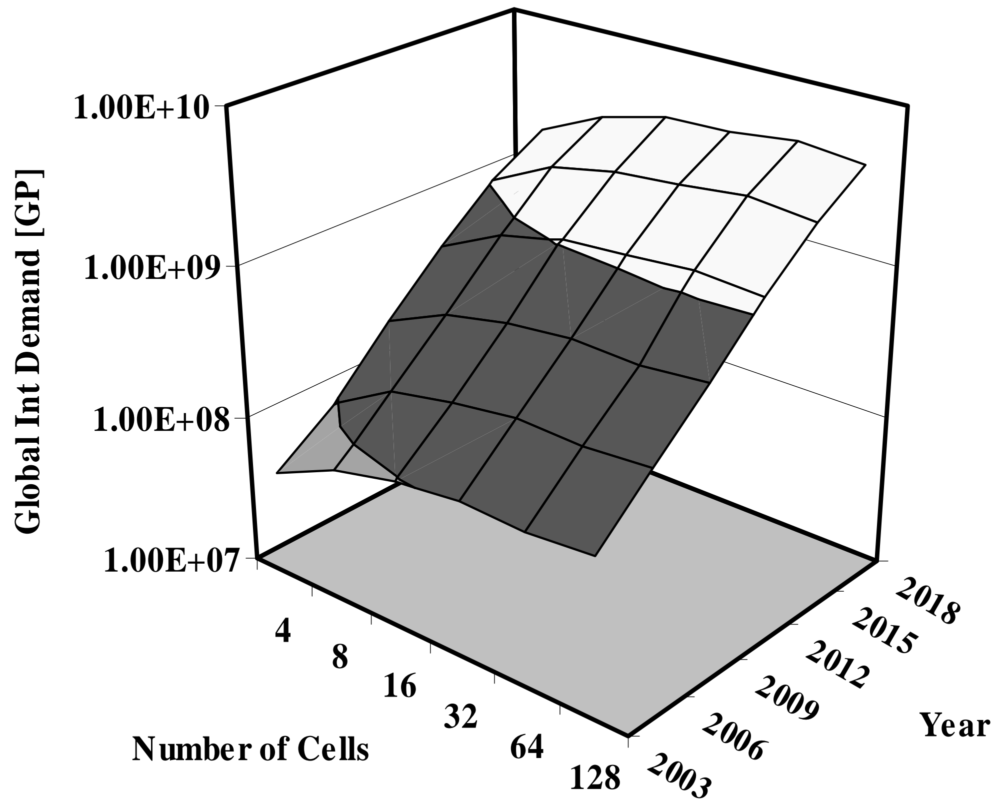

Figure 71: The total length (in terms of the gate pitch) of interconnects between nodes vs the node count and technology generation. The colored bands represent the regions between z-axis grid lines as projected onto the surface. Minimum feature size information is listed in Table 31.

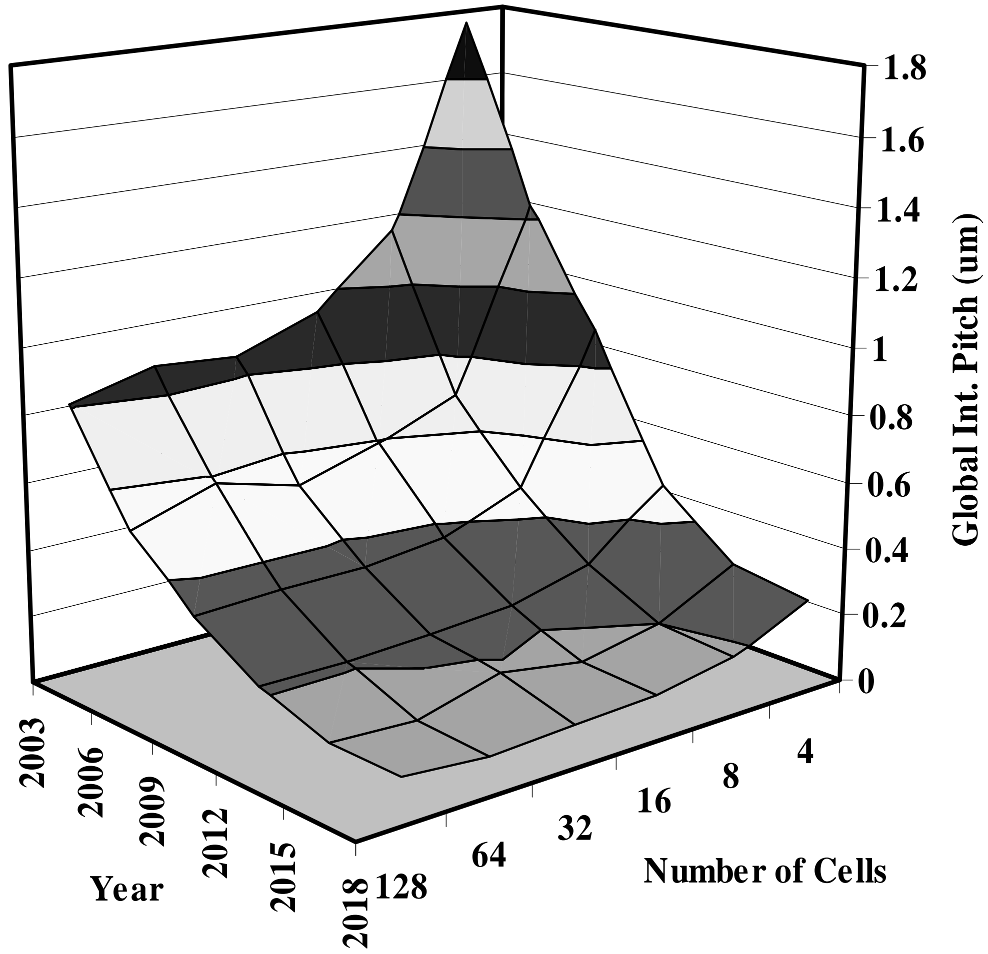

Figure 72: The wiring pitch for interconnects between nodes vs the node count and technology generation. The colored bands represent the regions between z-axis grid lines as projected onto the surface. Minimum feature size information is listed in Table 31.

6.2.3 Local and global clock frequency

The wiring resource trends discussed in the previous section have a significant impact on the overall performance for the processor array. Figures 73, 74, and 75 plot the global frequency , the local frequency, and the ratio of the local to global clock. The global clock frequency plotted in Figure 73 clearly does not scale with increasing system complexity. The reduction in pitch and increasing line length result in a nearly 90% reduction in the maximum speed of across chip communication. The local clock frequency, however, increases by a factor of 40 with increasing system complexity. The increasing gap between global and local clock frequency illustrated in Figure 76 shows that by the 2018 ITRS node, the local clock is over two orders of magnitude faster. This reduction in performance is due to both increasing wire length for high fanout nets and reduced line dimensions required by the increasing demand. The only viable solution to this problem is the use of shorter wires and/or lower density wiring schemes. Observations made in Chapter 4 regarding limitations on throughput with regard to global communication imply that a stochastic net-length distribution does not meet requirements for the efficient use of system resources. The assumption that the net-length distribution applies to this system appears to be unfounded. A cellular array architecture like the one considered here is the product of high level system design in that the organization and placement of system resources and communications channels have been defined by the system designers rather than the type of place and route tools used to wire random logic inside custom blocks. Stochastic distributions tend to break down when the wiring schemes begin to reflect the design choices of system architects.

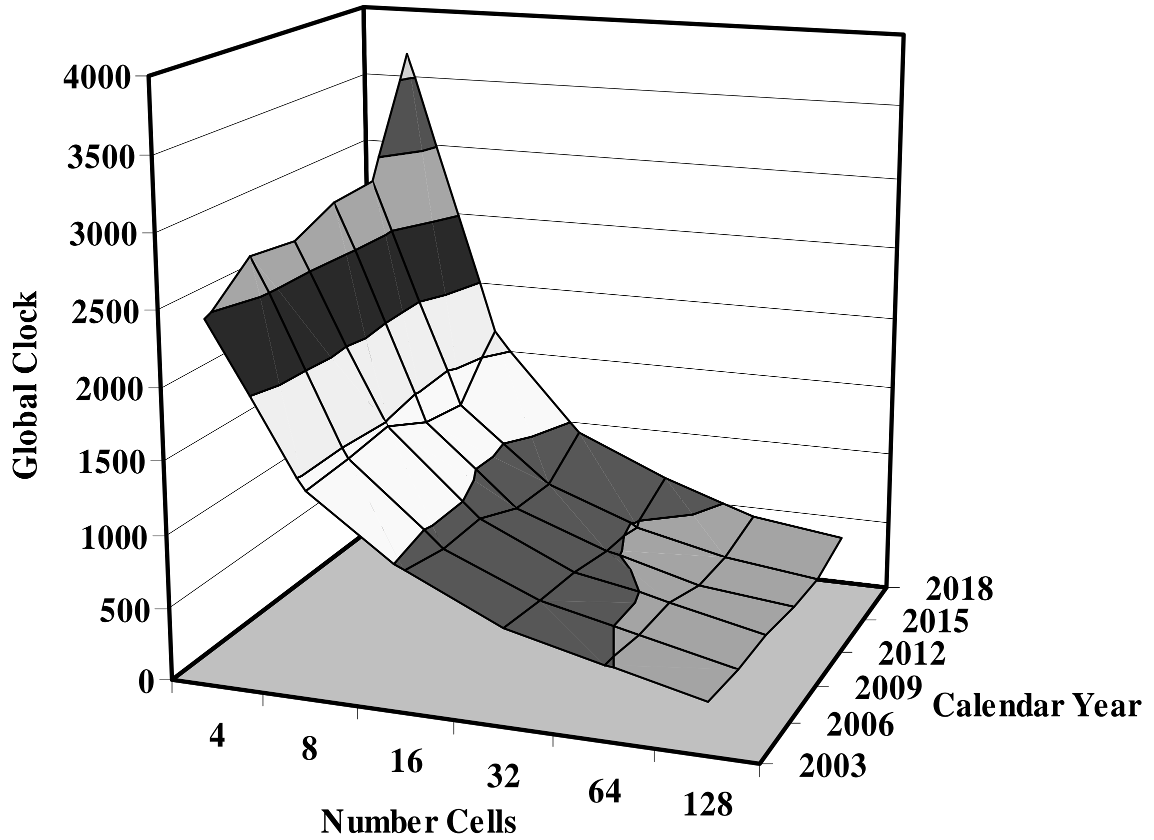

Figure 73: The global clock speed in MHz for increasing node count and ITRS technologies. The colored bands represent the regions between z-axis grid lines as projected onto the surface. Minimum feature size information is listed in Table 31.

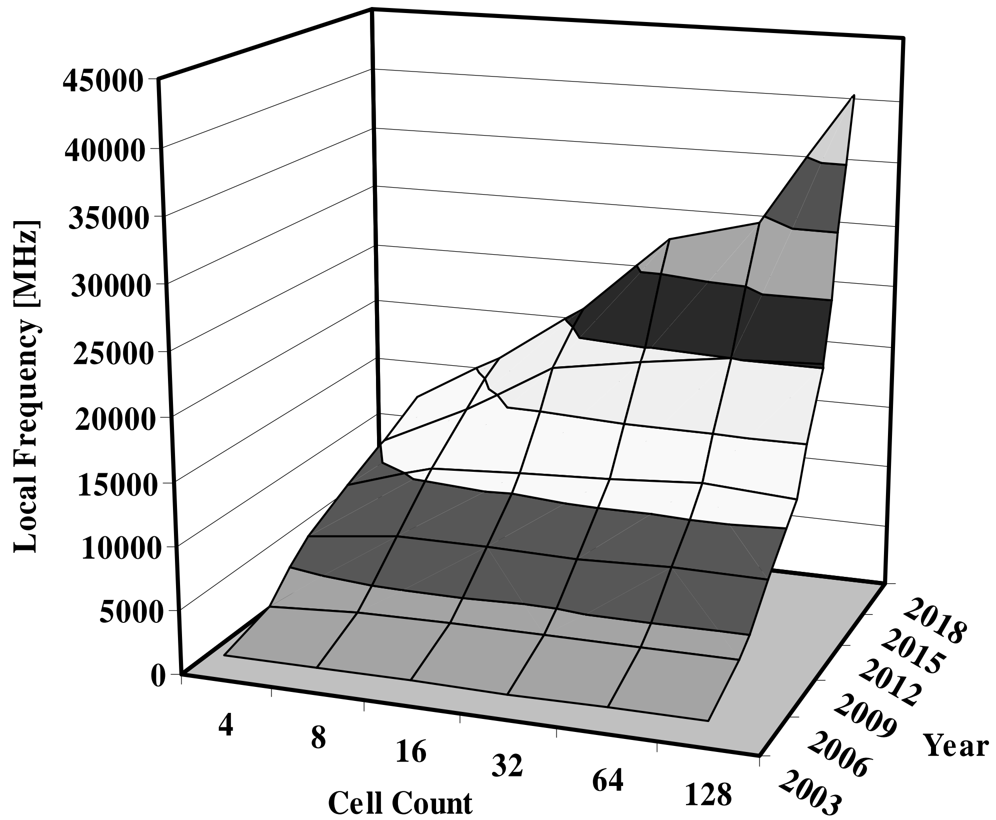

Figure 74: The maximum frequency of the local clock vs the node count and technology generation. The colored bands represent the regions between z-axis grid lines as projected onto the surface. Minimum feature size information is listed in Table 31.

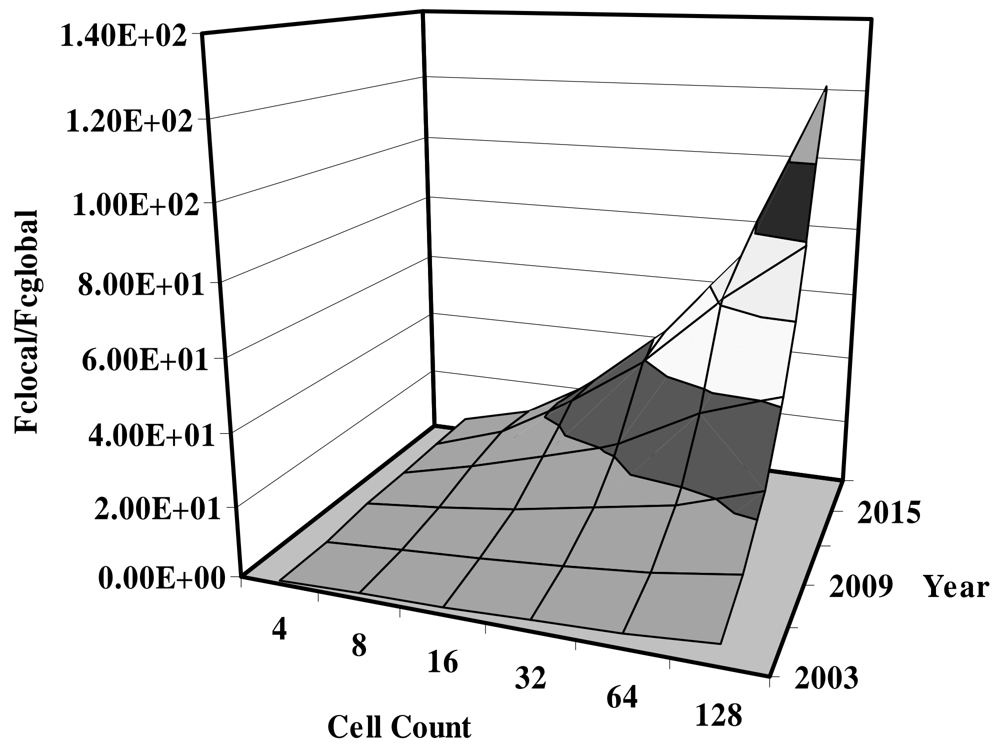

Figure 75: The ratio of the local to global clock frequency vs node count and ITRS technologies. The value of the ratio at N = 4 for the 2003 ITRS generation is 0.82. Minimum feature size information is listed in Table 31.

6.2.4 Power dissipation

The expected power dissipation for the array architecture is plotted in Figure 76. Clearly the dominant contribution to the power is the increase in gate count and clock speed as the technology advances. The peak power of 1KW is unsustainable and it is reasonable to assume that the performance of this system will be limited primarily by the requirement that the average power dissipation not exceed 300W as noted in the ITRS.

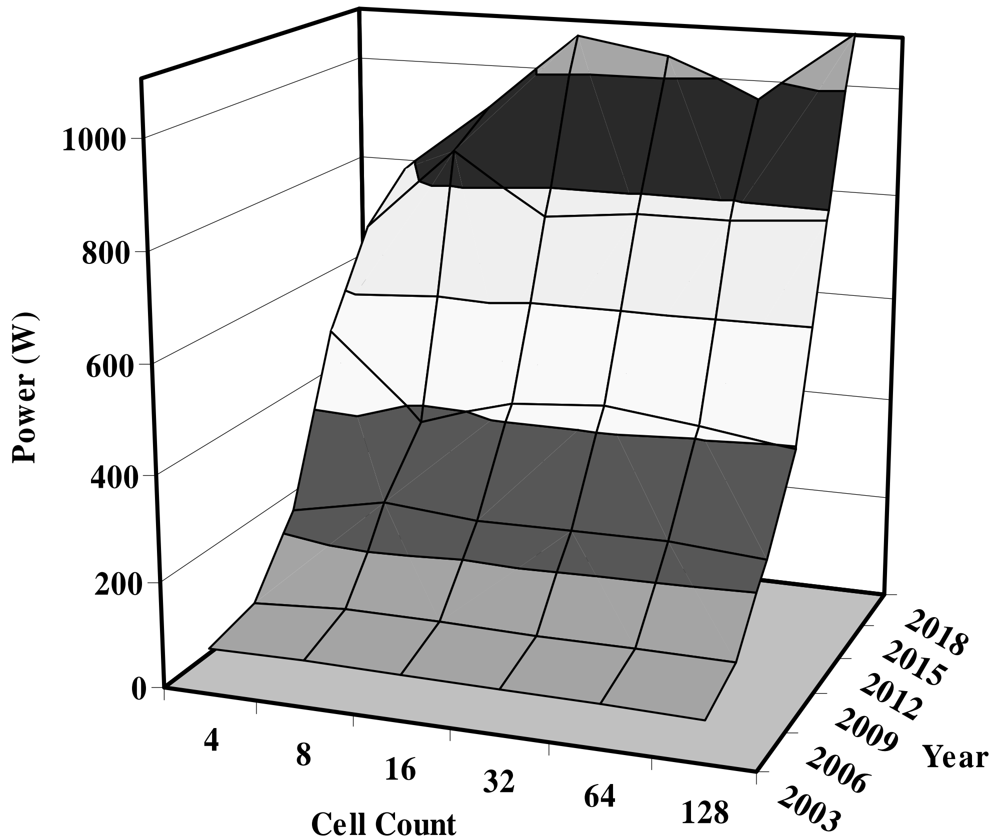

Figure 76: The average power dissipation for the array processor of Figure 68 vs node count and ITRS technology generation. Minimum feature size information is listed in Table 31.

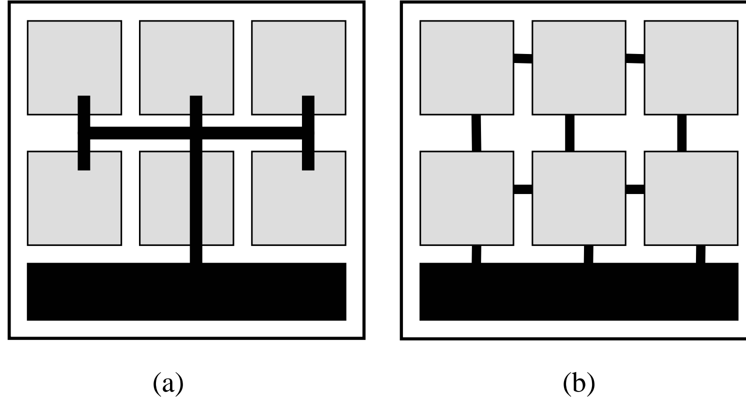

Figure 77: The two global interconnect networks for the processor array: (a) a single shared global bus, and (b) a systolic array network.

6.3 Design space for global communication

The limitations of the stochastic approach to modeling system level interconnects is apparent from Section 6.2.3. Realisticly, the nodes in this type of processor array would be connected via a bus or switched network. The bus modeling developed in Chapter 4 permits exploration of the global communication design space. In this section the processor array architecture of Figure 68 is simulated for two different global networks that span the design space: a shared bus connecting all nodes to one another and a nearest neighbor network in which only neighboring nodes communicate directly. These two approaches are illustrated in Figure 77.

6.3.1 Simulation methodology

In Section 6.2 simulations of the array processor were run under the assumption of a stochastic net-length distribution for both various ITRS technologies and node counts. Here, the technology will be restricted to the 2012 node (28nm) while the number of processor nodes is increased from 4 to 128. The wiring scheme is set via the bus specification defined in Section 4.2.1. The global shared bus is created by defining a bus entity with a list of every cell instantiated in the system entity. The

nearest neighbor case is simulated by defining a single bus between adjacent cells and extrapolating to determine the aggregate bandwidth and power consumption.

6.3.2 Shared global bus

The simulations are run for a 32 bit bus with both buffered and unbuffered bus lines. The relevant system statistics are collected in Table 32. The processor frequency saturates at 10.6 GHz due to the delay through the random logic network. The performance of the unbuffered bus line, however, degrades considerably as the node count increases. Even with only 8 nodes on the bus, the frequency is less than 200MHz. The bus length increases by a factor of 4 accounting for much of the performance degradation with the rest due to increased capacitive loading at the terminals. The buffered bi-directional bus is able to maintain a 1GHz frequency even with 128 nodes. Clearly, unbuffered bus lines quickly become prohibitively expensive for communicating over all but the shortest distances. If there are any more than 4 nodes on the bus, the system will experience a significant slowdown. Even the 1GHz frequency achieved by the buffered bus will be inadequate for hosting a large number of nodes as the processor speed is an order of magnitude greater than the maximum bus frequency.

6.3.3 Nearest neighbor wiring approach

The relative performance of the processor array vs the global communication bandwidth for a shared bus indicates that single shared bus architectures are not suited for efficiently wiring the chip for high performance. An alternative to this approach is the use of a bus architecture that reduces line length by breaking the bus into separate bus segments. At the extreme end of this design space is a nearest neighbor wiring scheme in which only adjacent nodes are connected as in Figure 77.

Table 32: The simulation results for a single shared bus connecting N_{node} processors.

System		Unbuffered Bus				Buffered Bus	
N_{node}	Power(W)	F_c(GHz)	F_{bus}(GHz)	$P_{ave}(mW)$	L_{bus}	F_{bus}(GHz)	$P_{ave}(mW)$
4	533	8.56	1.0	131	14.1	3.72	102
8	374	10.6	0.18	8.2	19.9	2.2	153
16	414	10.6	0.05	4.36	27.3	1.5	196
32	414	10.6	0.02	2.69	37.4	1.22	242
64	406	10.6	0.01	2.3	54	1.2	293
128	378	10.6	0.005	1.3	68.4	1.08	357

This wiring scheme closely resembles the wiring of a systolic array. Communication across the chip is handled by routing data from node to node in multiple clock cycles. The simulation results for the bus performance are listed in Table 33. The total bus power and aggregate bandwidth, BW_{total}, are calculated by multiplying the power and effective bandwidth of a bus segment by the number of bus segments between nodes in the array (a 32 bit bus segment operating at 1GHz has a 32 Gbps bandwidth, if there are 4 bus segments the total bandwidth is 128 Gbps). The expression for computing the aggregate bandwidth is given in Equation 74. The effective bandwidth, BW_{eff}, is calculated by assuming a 32 bit non-pipelined bus transferring 32 byte data with a 1 cycle node access penalty and applying equations 43 and 44. The number of bus segments for an m by n array is calculated as:

$$N_{seg} = 2(mn) - m - n \tag{73}$$

$$BW_{total} = N_{seq} f_{bus} W_{bus} \tag{74}$$

The latency of global communication for the global shared bus is determined

Table 33: The bus performance statistics for an processor array implemented in the ITRS 2012 technology with a processor local clock of 10GHz. The effective bandwidth is calculated via the aggregate bandwidth (Equation 74) and

N_{node}	$L_{bus}(mm^2)$	F_{bus}(GHz)	P_{ave}(mW)	P_{total}(W)	BW_{eff}(Gbps)
4	5.6	6.5	70	0.28	604
8	3.9	6.6	83	0.99	1843
16	2.5	7.1	83	2.0	2883
32	1.7	8.7	76	3.72	7215
64	1.3	10.1	67	7.5	19141
128	0.8	13.4	61	14.2	52741

primarily by the delay of the bus, so that a 10GHz array running on a 1GHz bus takes 10 processor clock cycles to communicate with other nodes. The maximum global communication latency for the nearest neighbor wiring scheme is dependent on the maximum number of nodes traversed to reach the opposite corner of the m by n array. For the 8x8 array of 64 nodes the latency, assuming 1 cycle per hop at 10GHz, is 1.6ns. For the 128 node array the required number of hops is approximately 24 for a total delay of 2.4ns. This provides a clear performance advantage over the shared global bus. The data in Table 33 indicates that the potential performance of a system wired with multiple, fast bus segments provides a substantial benefit in the maximum speed and bandwidth of global communication for a modest increase in the total system power.

6.4 Conclusion

In this chapter the performance potential for a generic gigascale array of processors is simulated via GENESYS 2K4 to determine the suitability of three different wiring schemes for a system of increasing complexity. The stochastic net-length distribution is found to be prohibitively expensive with respect to the global wiring resources and delay. Furthermore, it is determined that the stochastic network does

not adequately reflect the impact of architectural design on the system level distribution. A shared global bus is simulated and is shown to exhibit poor scaling properties with increasing system complexity. The other end of the design spectrum, a nearest neighbor wiring scheme is shown to scale well with increasing complexity. While the performance advantages of this wiring scheme are obvious, many system applications and implementations do not support the level of synchronization required for efficient communication between nodes. The most likely global wiring solution for a generic gigascale SoC is a hierarchical bus design [59] that accomplishes the goal of reducing the bus length and loading while providing a communication data-path suitable for the SoC architecture and target application. This example shows that GENESYS is capable of estimating system characteristics and providing an indication of the future direction of bus-based high performance SoCs.

CHAPTER VII

CONCLUSION AND FUTURE RESEARCH

7.1 Overview

The continued scaling of semiconductor technologies into the nanometer regime will enable the design and implementation of high performance systems containing multiple billions of transistors integrated on a single substrate [3]. Concurrent with the exponential growth of the chip transistor count is an increase in relative system complexity as chip designers attempt to take full advantage of larger and larger transistor budgets. Costs associated with the validation of these highly complex systems are driving a significant portion of the semiconductor industry to adopt SoC design solutions for VLSI systems. The GSI Hierarchy promotes the basic thesis that future limitations and opportunities for gigascale systems are codified as fundamental, material, device, circuit, and system. The system level limitations can shift significantly with fundamental changes to the architectural implementation and design methodology at the system level. This dissertation asserts the thesis that a next generation generic systems simulator (GENESYS 2K4) is a viable platform for exploring system limits for a broad spectrum of the SoC design space. GENESYS 2K4 leverages earlier work in regarding the modeling of lower levels of the GSI hierarchy as codified in the original GENESYS simulator [25]. The development of a new system level model suitable for SoC designs permits the rapid exploration of chip level performance trends with respect to changes at any level of the hierarchy.

A key element for a wide class of present and future SoC implementations is the design of the on-chip communication networks for connecting processors and peripheral components. The substantial increase on available resources and maximum clock frequency means the ultimate performance of complex, high performance SoCs will be governed by the ability to rapidly transfer data to and from system components. Historically, the design of system buses has been at the printed circuit board (PCB) level. As formerly off-chip components are integrated in a single chip, the design of the system bus becomes an issue for chip level design. The advantages in on-chip integration for buses include ultra-high available pin count for implementing wider data channels and higher on-chip frequencies. This work attempts to explore limits on the performance for an important class of SoC designs by correlating the system throughput to the relative performance of on-chip buses with respect to an integrated processor core. A methodology for specifying design and modeling the physical performance limits of on-chip buses has been derived to model the primary chip communication paths separate from stochastically derived interconnect distributions in either random logic blocks or globally between cells in an SoC design.

The limitations and opportunities for future performance trends of SoC applications are governed primarily at the system level by the properties of the SoC design methodology and effectiveness of on-chip busing. The goal of this work is to develop and exercise a modeling framework for providing a quantitative analysis of the impact these have on system performance.

7.2 Summary of Results

To date, no generic methodology for modeling the physical performance of SoC taking into account the impact of core-based design methodologies and on-chip busing on the overall system throughput has been reported. A quantitative assessment of the impact of on-chip bus designs for SoC implementations provides insight into the performance scaling properties of SoCs into the next decade.

To model a core-based heterogeneous SoC, a new system methodology is implemented. The new system methodology incorporates a hierarchical system description based on the definition of individual megacells and the collections of megacells comprising the chip level system. The simulation of the SoC requires the application of the core GENESYS models for the fundamental, material, device and circuit levels of the GSI hierarchy to each defined megacell for evaluating the maximum operating frequency, power dissipation, and area for each type of cell in the specified design. The system level performance statistics for the collection of megacells is determined by combining the performance statistics for the simulated megacells. The global wiring resources are estimated via the use of a previously derived stochastic global net-list distribution [34]. The estimation of the global net length incorporates a description of the physical chip floorplan for estimating the impact of cell placement on the global wiring resources. The projections of frequency, power, and area for 5 commercial SoCs against both actual data and projections utilizing the earlier GENESYS uniprocessor system model confirm that the block-based methodology significantly improves the accuracy of the performance projections.

In order to assess the impact of on-chip bus performance on overall system throughput, GENESYS 2K4 combines an explicitly bus definition with a compact set of analytical and empirical models for calculating the length, area, delay, average

power dissipation, and peak bandwidth for a bus. The bus description defines the physical attributes (line width and spacing, and driver circuit sizing) as well as the logical attributes (data width, and net-list). The modeling supports several different physical implementations (unbuffered, uni-directional, and bi-directional) for the individual bus lines. Simulation results show significant performance advantages for buffered bus lines over unbuffered lines. A novel cell placement algorithm is adapted for optimizing the length of on-chip buses using adjacent placement of cells on the bus net-list. Applying an optimal bus driven cell placement to several examples shows significant reductions in bus length and delay when compared to several different placement schemes. A generic model for bus transactions is introduced for determining the optimal bus pipelining and maximum data rate. A modified expression for the average cycles-per-instruction/operation is introduced to determine the effects of bus design based on penalties for memory access and bus data transfer. This model is applied to an example generic processor-bus-memory system to show that the relative bus performance strongly limits the maximum throughput. An optimal bus design is proposed and is shown to permit a significant increase in the maximum achievable throughput. The modeling is then applied to an example bus-based SoC (IBM 440GP) utilizing a commercially available on-chip bus architecture and shows agreement with published performance specifications. The performance scaling of the example design with respect to the ITRS technology nodes shows significant increases in the maximum operating speed for both the core megacells and bus lines, with corresponding reductions in the area and power consumption. The average power density, however, is shown to increase beyond acceptable limits. This result indicates that managing power dissipation is a key barrier to achieving the optimal performance afforded by nano-scale technology.

A feature of many embedded systems is the incorporation of high density DRAM memory arrays on-chip to increase performance in systems where high storage capacity is desired without incurring the high area cost of cache SRAM. A generic model for on-chip DRAM is introduced and implemented in GENESYS 2K4. The DRAM array model is based upon the characterization of an individual trench capacitor single transistor DRAM cell. An area model based upon physical layout parameters is developed and shown to match historical trends in DRAM cell sizing. The power model for the on-chip array is determined via analytical models for the average energy cost for the basic DRAM read/write operations. The DRAM methodology leverages existing models for the array access, decoding, and sensing, and output logic to produce estimates of the DRAM array area, average power, and access time. Comparisons against data for several commercial embedded DRAM macros show that the model projections agree reasonably well with published data. The modeling is then applied to project the performance of a 1Gb DRAM array for selected road map years. The simulation results show significant reductions in the array access time, and area, but no corresponding improvement in the power dissipation. Again, power dissipation is projected to be a limiting factor on the achievable performance improvement from technology scaling.

Finally, the GENESYS 2K4 SoC model is applied to a generic gigascale array of processors utilizing different global interconnection schemes. The scaling properties of the stochastic net-list distribution for global interconnects are shown to be poor. This calls in to question the suitability of applying stochastic distributions to multi-processor based SoCs where the actual wiring schemes are a function of high-level architecture design choices. Simulations for a global shared bus likewise indicate that course bus architecture cannot maintain acceptable performance with respect to increasing system complexity. A fine grain approach based on connections

between adjacent cells shows substantial improvements in scalability and maximum on-chip bandwidth. These results indicate that the most efficient bus architectures will be hierarchical in nature with multiple on-chip bus segments to reduce line length and increase the available bandwidth for providing on-chip cores with adequate communication resources.

In conclusion, the GENESYS 2K4 SoC modeling methodology is shown to be suitable for exploring system performance limits for a broad class of SoC designs and identifying roadblocks and potential solutions for future gigascale systems.

7.3 Future Research

Future opportunities for enhancement of the GENESYS 2K4 framework can be divided into two primary areas: core modeling and system methodology. Core modeling efforts relate to the development of new models at the material, device, and circuit levels of the GSI hierarchy. System methodology involves extending or augmenting the GENESYS SoC/Bus modeling framework at the architectural level.

7.3.1 Core Modeling Enhancements

The following elements have been identified as key areas of development for future efforts in modeling at the lower levels of the GSI hierarchy.

- Circuit and device level modeling for mixed-signal analog applications
- A more detailed analysis of Rent parameters for megacells based on cell functionality rather than assuming one value for all random logic and another for memory

- Modeling for multiple threshold logic for low power circuit operation
- More accurate modeling of interconnect parasitics including fringing and relaxation of the assumption of ground planes capacitances for multilevel interconnect architectures
- Improved repeater delay/sizing models taking into account sub-optimal placement and floor-planning
- Device modeling support for complex device structures and material systems other than bulk silicon
- Modeling for key circuit macros such as latches and clock driver circuits for estimation of area utilization and power
- Improvements to the bus line modeling including tri-state logic and distributed multiplexor implementation

7.3.2 System Modeling Enhancements

The following areas are key research goals required for expanding the applicability and accuracy of the GENESYS 2K4 system methodology.

- Modeling for bus traffic based upon connectivity and net list cell functionality (i.e processor core, memory, i/o, control, DMA blocks...)
- Bus modeling support for hierarchical bus architectures (support logic, bus bridges)
- Modeling of on-chip switched networks (cross bar, ring, butterfly...) for multiprocessor throughput

- Explicit modeling for chip multi-processor CPI based on processor size, number, and bus/network
- Incorporation of power reducing techniques for overcoming key system level power density limits
- Performance modeling for mixed/signal applications
- System level optimization routines for power density constraints
- Improved floor-planning/cell placement algorithms for optimal wire-lengths and performance

The key area for investigation is the modeling of chip-multiprocessor performance. Support for modeling the CPI in conjunction with the bus architecture and complexity of the system would make GENESYS uniquely qualified for assessing the impact of technology and architectural design choices on the key performance metrics for an important class of gigascale SoC designs. Insights into the performance scaling of billion transistor architectures gained utilizing the approach taken in GENESYS 2K4 should yield compelling evidence for the direction of future SoC designs.

APPENDIX A

2003 ITRS TECHNOLOGY NODES

Year	2003	2004	2005	2006	2007	2008	2009
F (nm)	65	53	45	40	35	32	28
Xtrs 10^6 (CP)	180	226	285	360	453	571	719
Xtrs 10^6 (HP)	439	553	697	878	1106	1393	1756
Die Size (mm2)	280	280	280	280	280	280	280
Die Size (HP)	310	310	310	310	310	310	310
Wiring Levels (max)	13	14	15	15	15	16	16
Wiring Levels (min)	9	10	11	11	11	12	12
Local Clock (MHz)	2976	4171	5204	6783	9285	10972	12369
Vdd	1.2	1.2	1.1	1.1	1.1	1	1
Max Power (HP)	149	158	167	180	189	200	210
Max Power (CP)	80	84	91	98	104	109	114
Oxide Thickness (nm)	1.3	1.2	1.1	1	0.9	0.8	0.8
Elec. Ox. Thick. (nm)	2.1	2	1.8	1.7	1.3	1.2	1.2
Sub. Current (mA/um)	0.03	0.05	0.05	0.05	0.07	0.07	0.07
Gate Leakage (A/cm2)	220	450	520	600	930	1100	1200
Drive Current (A/m)	980	1110	1090	1170	1510	1530	1590
Dielectric constant	3.3	3.1	3.1	3.1	2.7	2.7	2.7
Resistivity (um-cm)	2.2	2.2	2.2	2.2	2.2	2.2	2.2
Expected Global (MHz)	1064.7	1492.2	1861.8	2426.7	3321.8	3925.3	4425.1

Figure 78: Technology characteristics for near term years

Year	2010	2012	2013	2015	2016	2018
F (nm)	25	20	18	14	13	10
Xtrs 10^6 (CP)	1546	2454	3092	4908	6184	9816
Xtrs 10^6 (HP)	2212	3511	4424	7022	8848	14045
Die Size (mm2)	280	280	280	280	280	280
Die Size (HP)	310	310	310	310	310	310
Wiring Levels (max)	16	16	16	17	18	18
Wiring Levels (min)	12	12	12	13	14	14
Local Clock (MHz)	15079	20065	22980	33403	39683	53207
Vdd	1	0.9	0.9	0.8	0.8	0.7
Max Power (HP)	218	240	251	270	288	300
Max Power (CP)	120	131	138	148	158	168
Oxide Thickness (nm)	0.7	0.7	0.6	0.6	0.5	0.5
Elec. Ox. Thick. (nm)	1.1	1.1	1	1	0.9	0.9
Sub. Current (mA/um)	0.1	0.1	0.3	0.3	0.5	0.5
Gate Leakage (A/cm2)	1900	2400	7700	10000	19000	24000
Drive Current (mA/mm)	1900	1790	2050	2110	2400	2190
Dielectric constant	2.3	2.3	2	2	1.7	1.7
Resistivity (um-cm)	2.2	2.2	2.2	2.2	2.2	2.2
Expected Global (MHz)	5394.7	5620	6020	6020	6530	6530

Figure 79: Technology characteristics for long term years

APPENDIX B

GENESYS 2004 MANUAL

CONTENTS

1 Introduction

1.1 GSI Heirarchy

1.2 GENESYS

2 GENESYS System Modeling

2.1 Homogeneous chip model

2.2 Heterogeneous chip model

3 Installation

3.1 Requirements

3.2 Installing GENESYS

4 Running GENESYS

4.1 Command line switches

4.2 Homogeneous simulation

4.3 Heterogeneous SoC simulation

4.4 Creating plot files

5 GENESYS Input Files

5.1 Technology File

5.2 Global Parameters File

5.3 Machine Description

5.3.1 Syntax and keywords

5.3.2 Specifying tech and glb files

5.3.3 Declaring Entities

5.3.4 Instantiation

5.3.5 Declaring Busses

5.3.6 The System Entity

5.3.7 Cell Placement

5.4 Plot Configuration File

1. Introduction

1.1 GSI Heirarchy

It is the fundamental thesis of the gigascale intergration (GSI) group that future opportunities for multi-billion transistor chips/systems are governed by a heirachy of limits that can be codified as:

- fundamental
- material
- device
- circuit
- system

Each of the levels of the GSI hierarchy are described in more detail below.

Fundamental:

The fundamental limits arise from thermodynmics, quantum mechanics, and electro-magnetics. A key example from electro-magnetics is the fundamental time- of-flight limit on signal propogation. The minimum propogation delay of a signal is determined by the maximum velocity of the signal, in this case the speed of light in free space. This limitation is fundamental in nature becuase it applies universally without consideration from any other level of the hierarchy.

Material:

The material limits comprise the next level of the heirarchy and are more restrictive than the fundamental limits. A clear example is the dielectric properties of a material. The speed of light in a medium is inversely proportional to the square root of the relative dielectric constant of the material. Therefore, the material limit imposes an additional restriction on the minimum propogation time.

Device:

The device limits arise from the geometry and physics of semiconductor devices. A key device limit for MOSFET devices is minimum channel length. This is the minimum source/drain seperation for which the gate-stack maintains adequate control of the channel to turn the device off when zero gate voltage is applied. This limit is often evaluated via the measurment/projection of the drain-induced-barrier-lowering (DIBL) effect.

Circuit:

The circuit limits are much more numerous than the limits at the lower levels of the hierarchy. A comparison of two key circuit designs utilized in current methodologies serves to highlight the relative limitations. CMOS logic is noted for being

relatively low power, but domino logic is inherently faster although it consumes more power. Other key circuit design limits involve noise margin and stability which are dependent to some extent upon the circuit design methodology.

System:

The system level of the GSI hierarchy is the most nebulous and ill-defined of the set of limitations governing future prospects for GSI chips. Some system design considerations affecting key performance metrics is the global/local interconnect distribution, the degree of pipelining/logic depth, the number of metalization levels... etc.

1.2 GENESYS

The GENEric SYstems Simulator (GENESYS) was developed to explore future trends and tradeoffs between technology and architecture. The core concept behind GENESYS is that a system can be described by a set of inputs that encapsulate the GSI heirarchy. A key set of analytical and empirical models consume these inputs to project key system performance metrics such as power dissipation, die size, and clock frequency in addition to a host of other information.

2. GENESYS System Modeling

A core consideration in GENESYS simulations is the assumptions made in selecting a system model for projecting the key performance metrics. Currently, there are two different system modeling methodolgies incorporated in GENESYS. These are discussed briefly in the following subsections. The Homogeneous and Heterogeneous system models draw thier names from the type of stochastic interconnect distribution

utilized by the model.

2.1 Homogeneous chip model

The homogeneous chip model is the original system model used in earlier versions of the simulator. A fixed generic uniprocessor architecture is modeled. Under this model, the system consists of cache and random logic. There are key assumptions made concerning both the memory and logic portions of the system.

Between 0 and 3 caches are permitted in this system model and specific assumptions regarding the organization of the cache hierarchy are made according to the number of caches present. If only one cache is specified, it is assumed to be a unified level one cache for both instruction and data. If two caches are present, they are still assumed to be level one, but are split caches: one resevered for instruction address information, the other for data. With three caches present the assumption is that there is the split level one cache and a single unified level 2 cache. These distinctions are not vitally important for estimating the physcial performance of the system, but are neccessary for assessing the impact of the memory heirarchy on the overall system throughput.

The far more important assumption of this system model concerns the random logic portion of the chip. The key assumption here is that the processor's logic can be modeled as a homogeneous Sea-of-Gates. The interconnect distribution utilized by this model was derived under the assumption that the local wiring characteristics of small groupings of gates does not vary across the chip. This means that the Rents parameters derived from an examination of one section of the random logic are applicable to the entire random logic network. At the very least this supposition

implies that there exists an average value for the Rent's exponent and coefficient that adequately describes the wiring distribution from the local level to the global level.

Under this system model, detail is sacrificed for simplicity. This modeling is compact and powerfull because the user can quickly asses the projected impact of changes to the technology at the system level, but only within a confined design space. The homogeneous model is also less flexible when examining the impact of changes made to the system architecture as it is confined to specific design space and too generic within that space to transparently compare different design methdolgies.

2.2 Heterogeneous chip model

A newer modeling methodology was developed for GENESYS to overcome the limitations of the homogeneous uniprocessor model. There were two primary goals to be met by this new methodology: that it be capable of describing a wide class of architectures and that it provide adequate detail for assessing architectural trade-offs between architectures.

Both the homogeneous and heterogeneous modeling methodologies are based upon the same premise: that no matter the system architecture, the technology is the foundation upon which it is built and any such system can be described as a collection of devices and wires for the purpose of electrical modeling. This means that the core models used to calculate the system performance metrics are unchanged and that the new system model is superimposed over the homogeneous modeling.

The hierarchical block modeling methodology provides a framework for describing the system as a collection of distinct components or megacells. Each megacell

is itself described with the full set of GENESYS inputs. These cells are then simulated in GENESYS seperately and the total system metrics collected from the results for each indvidual cell. At the lowest level of the specification where a defined block/cell contains no other blocks, it is assumed to be homogeneous in nature. Once the initial simulation is complete, GENESYS applies a heterogeneous wiring distribution to the global interconnects between cells to estimate the total wiring length and interconnect dimensions. Becuase the system description allows for cells to contain other cells, the user has the flexibility of determinng the fineness of the system specification.

This methodology closely approximates the physical design aspect of large scale systems-on-a-chip (SoC) in which pre-designed megacells are fabricated on the chip to form the complete system. A primary advantage of this methodology is that each cell may have its own distinct design parameters. Additionally, almost any sort of architecture may be specified. For example a user may describe an on-chip multiprocessor by defining a single processing element, replicating it as needed, and adding as many or as few caches as desired (distributed or shared memory). A uniproccessor can be modeled with higher fidelity than the homogeneous model by explicitly defining each of the functional components such as integer and floating point units.

Further reading on the block modeling methodology and the heterogeneous wiring distribution can be found in [53]

3. Installation

3.1 Requirements

GENESYS was developed on SUN Ultra workstations but may be compiled from

source code to run on any system. An installation script is provided for ease in creating the GENESYS executable. This script is specific to UNIX systems and may require slight modification to run properly.

The following applications are required to use the installation script (UNIX/Linux)

- make
- gzip
- tar
- g++

3.2 Installation

If the system (UNIX/Linux) and application requirements are met then GENESYS may be installed by invoking the installation script included with this release. First, copy the installation script and archive file to a directory of your choice. Then simply type the name of the script (geninstall) at the command prompt.

Example: running the installation script

```
> geninstall
```

The installation script performs the following actions:

1) unzips the compressed archive
2) extracts the source files from the archive
3) runs the make utility to compile and link the executable

4) cleans up installation files

The end result should be the GENESYS executabe, genesys, residing in the same directory from which the installation script was invoked.

4. Running GENESYS

Once installation has been completed, the application is ready to run. This version of GENESYS was developed to run from the command prompt.

4.1 Command line switches

A listing and explanation of the GENESYS command line arguments is given in the help file. To access the help file simply type either of the following at the command prompt:

```
> genesys -h
> genesys -help
```

This command will list the GENESYS help file on screen. To save the help information to a file type the following:

```
> genesys -h > help.txt
```

4.2 Homoegeneous simulation

If the -soc option is not selected GENESYS uses the homogeneous system modeling and requires the specification of the *.tech and *.gen input files.

4.3 Heterogeneous simulation

If the -soc option is selected GENESYS utilizes the heterogenous sytem modeling and requires only the name of the machine description file as further input.

4.4 Creating plot files

When used in the homogeneous mode GENESYS can create plot data files. These data files contain data for plotting 4 different x-y plots. This option is selected by setting the variable plotting parameter to a value of 1 and by specifying the name of the plot configuration file in the *.gen input file. The plot configuration file is explained in more detail in section 5.4.

5. GENESYS input files

5.1 Technology file

The GENESYS technology file (*.tech) contains the key input parameters describing the technology in which the simulated system is implemented. There are a total of 15 associated inputs in the technology file. This file contains a simple list of parameter names and associated values.

5.2 Global parameters/*.gen file

These files contain all of the GENESYS input parameters that are not in the technology file. The global parameters file (*.glb) specified in the machine description for the heterogenous system modeling lists the default values for any GENEYSS parameters not specified in the definition of a block or entity. In homogeneous system modeling the file is named *.gen. Like the technology file, the global parameters file is a listing of parameter names and values.

5.3 Machine description

Unlike the *.tech and *.glb files the machine description requires a syntax and grammar for specifying system. This grammar and syntax are described in the following subsection.

5.3.1 Keywords and Syntax

The keywords used in the machine description language are as follows:

- **use** specifies the use of another file.
- **global** file contains global parameters.
- **tech** file contains technology parameters.
- **entity** used to define new component.
- **block,bus,megacell** defines the entity type. Used in both the definition and instantiation.
- **is,end** used with **entity** to begin and close definitions
- **placement** used to begin specification of megacell placement for the system entity.

- **bus** used for definition of a chip level bus.
- **bus-list** defines a set of cells connected to a bus.
- **system** used for definition of the chip level system entity.

The syntax of a GENESYS machine description is modeled loosely on standard VHDL. All parameter assignments, declarations, and instantiations in the machine file are terminated with a semi-colon.

5.3.2 Specifying tech and global files

Before running any simulations, GENESYS must first load a set of appropriate parameters and values for both the technology inputs (feature size, oxide thickness...) and other global parameters (number of interconnect tiers,inverter area...). The two valid file types are loaded in the following manner:

use tech <filename1>;
use global <filename1>;

NOTE: The technology and global parameters file must be terminated with the *.tech and *.glb extensions respectively for use with the hetergeneous system model.

5.3.3 Declaring entities

An entity is declared in the machine description file by opening the definition using the **entity** and **is** keywords followed by a series of entity specific parameter assignments. The entity definition is closed with the **end** and **entity** keywords. A simple entity declaration is shown below:

entity block <name> **is**

<parameter1> = <value>;

<parameter2> = <value>;

end entity <name>;

The above example creates a simple entity of type **block** and redefines two parameters with values that are specific to that entity. All other parameters remain as defined in the global and technology parameter files.

5.3.4 Instantiation

A newly defined entity is not limited to containing a series of parameter/value strings. In addition to redefined parameters, an entity may contain any number of instances of other entities. An example is used to illustrate this below:

entity block <name> **is**

<parameter1> = <value>;

<parameter2> = <value>;

block <name2> <instance name1>;

block <name3> <instance name2>;

end entity <name>;

The above example differs from the previous in that it contains two instances of other entities. These instances will contribute to the power and area results for the defined entity. To change parameters for an entity instance define it in the following manner:

block <name2>(<parameter1> = <value>,...);

The list of parameters and values in the parentheses overides any values for those parameters either in the entity definition or in the global parameters file.

5.3.5 Declaring buses

While much of the global interconnect resources are treated stochastically via a previously defined distribution [34], the user may specify explicit connections between megacells in a system. This is accomplished via the definition of an entity of type **bus**.

entity bus <name> **is**

<parameter1> = <value>;

<parameter2> = <value>;

bus-list (<name1>,<name2>,<name3>,. . .);

end entity <name>;

In the above example the bus is defined similarly to an entity with the exception of the special keyword, **bus-list**. This keyword is used to begin the definition of a list of cells that are connected to the net. The list is contained in parentheses with commas seperating cell names. Becuase bus entities are simulated outside of the GENESYS core, they are not instantiated within the system entity. Bus definitions act as their own instances.

5.3.6 The system entity

The chip level system entity is essentially a list of cell/block instances comprising the complete System-on-a-Chip. The definition follows the following form:

entity system <name> **is**

  megacell <name2> <instance name1>;

  megacell <name2> <instance name2>;

  megacell <name3> <instance name3>;

  megacell <name4> <instance name4>;

  megacell <name5> <instance name5>;

end entity <name>;

The above system is comprised of five megacells. The first two are identical (instances of the same entity). As in all other instantiations, the parameters may be redefined for each megacell in the system entity. Also, the **block** and **megacell** keywords are interchangeable.

5.3.7 Megacell Placement

In order to properly estimate the global wiring requirements, knowledge of the placement of megacells on the chip is required. The placement is specifed by superimposing a grid on the chip surface and specifying the lower-left and upper-right coorinates of the cell corners (cells are assumed to be roughly rectangular in shape). There are several specific parameters associated with the placement specification. An example is given below:

megacell placement is

  grid_value = 100;

  placement_type = <type>

constraint = <constrain>

<cell_name1>

<cell_name2>

<cell_name2>

<cell_name3>

...

<cell_nameN>

end placement

5.4 Plot Configuration File

The plot configuration file is used only with the homogeneous simulation mode and controls the output of plot data to a specified file. The configuration file is utilized by setting the parameter Plotting_Parameter to one and specifying the name of the configuration file. The data specified in the configuration file is shown in the example below:

Plot_Type <value> [0-Length/Delay, 1-Recip/Delay, 2-Power/Delay,3-user selected]

Start <value> [0-1 meters, 2 seconds, 3 user selected]

Stop <value> [0-1 meters, 2 seconds, 3 user selected]

Num_points <value>

Scale <value> [0-linear, 1-log]

Input_Parameter <parameter name>

Output_Parameter <parameter name>

Plot_File <filename>

The Plot_Type variable allows the user to select from among four types of plots:

interconnect length vs. delay, the reciprocal length^2-delay plane, the power-delay plane, and user selected. The reciprocal length^2-delay plot is usefull for assessing the performance of local and global interconnect with respect to various limits related to the GSI heirarchy. The power-delay plane similarly evaluates the energy cost with respect to the GSI limits. The user selected option activiates the Input and Output parameters in which the user specifies which input will vary and which output will be monitored. The Start and Stop parameters set the range overwhich the input variable(s) is varied. The Num_Points parameter adjusts the density of collected data points. The type of plot scale for the x-axis data can be chosen as either log or linear. Finally, the Plot_File parameter is used to specify the filename under which the resulting plot data will be stored.

Once the plot data is collected, the results may be plotted via a third party application such as Grace (xmgr) or Excel.

APPENDIX C

GENESYS HELP FILE

GENESYS 2004 Help File

Command Line switches: -i, -t, -o, -default, -info, -help -soc

-i required if -soc not used:

This switch instructs GENESYS to load the following file as input.

USAGE: genesys -i <filename>

-t required if -soc not used:

This switch is used to specify the name of the input

file containing the technology (global) inputs.

USAGE: genesys -t <filename>

-o optional:

This option allows the user to specify the name of

the file in which the output will be saved. If not

specified the name of the output file will be the

the same as the -i input file but with the .out extension

USAGE: genesys -o <filename>

-default optional:

This option allows the user to create a default input file

NOTE: This will create a .gen and .tech file

USAGE: genesys -default

-info optional:

The -info switch instructs GENESYS to print intermediate

simulation information to the screen. This may also be

redirected to file output by the user.

USAGE: genesys -info

-help optional:

This option displays the help page.

USAGE: genesys -help

-soc required if *.soc file used:

This option enables genesys for SoC analysis using a properly

GENESYS machine description file <*.soc>.

All other arguments except for the -info switch are ignored

USAGE: genesys -soc <filename>.soc

EXAMPLE: the following examples run GENESYS. The first example saves the

output to test.out while redirecting the run-time

information to test.log. The second example does the

same but for a heterogeneous system description.

```
> genesys -i in.gen -t in.tech -o test.out -info > test.log
> genesys -soc machine.soc -o test.out -info > test.log
```

APPENDIX D

MACHINE DESCRIPTION

SOC Description for 200 MHz RISC Example

```
use global risc.glb;
use tech risc.tech;

entity block I_Cache is
    type cache;
    num_transistors = 2.28e6;
    external_rent = 0.2;
    ext_rent_const = 4.12;
    internal_rent = 0.2;
    rent_const = 4.12;
    num_caches = 1;
    Cache_C_1 = 32768;
    Cache_B_1 = 32;
    Cache_A_1 = 2;
    Cache_ndwl_1 = 4;
    Cache_ndbl_1 = 1;
    Cache_nspd_1 = 1;
    Cache_nfoldd_1 = 1;
    Cache_ntwl_1 = 1;
    Cache_ntbl_1 = 1;
    Cache_nspt_1 = 1;
    Cache_nfoldt_1 = 1;
    initial_width = 25;
    inverter_size = 3600;
end entity I_Cache;

entity block D_Cache is
    type cache;
    num_transistors = 2.1e6;
```

```
    external_rent = 0.2;
    ext_rent_const = 4.12;
    internal_rent = 0.2;
    rent_const = 4.12;
    num_caches = 1;
    Cache_C_1 = 32768;
    Cache_B_1 = 32;
    Cache_A_1 = 2;
    Cache_ndwl_1 = 4;
    Cache_ndbl_1 = 1;
    Cache_nspd_1 = 1;
    Cache_nfoldd_1 = 1;
    Cache_ntwl_1 = 1;
    Cache_ntbl_1 = 1;
    Cache_nspt_1 = 1;
    Cache_nfoldt_1 = 1;
    initial_width = 25;
    inverter_size = 3600;
end entity D_Cache;

entity block I_Cache_Tag is
    num_transistors = 1.08e5;
    external_rent = 0.47;
    ext_rent_const = 3.80;
    activity_factor = 12.0;
    internal_rent = 0.47;
    rent_const = 3.8;
end entity I_Cache_Tag;

entity block D_Cache_Tag is
    num_transistors = 1.53e5;
    external_rent = 0.47;
    ext_rent_const = 3.80;
    activity_factor = 12.0;
    internal_rent = 0.47;
    rent_const = 3.8;
end entity D_Cache_Tag;

entity block TLB is
    num_transistors = 1.344e5;
    external_rent = 0.35;
    ext_rent_const = 3.80;
    activity_factor = 15.0;
    internal_rent = 0.35;
    rent_const = 3.8;
```

```
end entity TLB;

entity block Cache_Cntrl is
    num_transistors = 9.42e4;
    external_rent = 0.6;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Cache_Cntrl;

entity block Ext_Interface is
    num_transistors = 1.104e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Ext_Interface;

entity block Sys_Buffers is
    num_transistors = 1.356e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Sys_Buffers;

entity block Free_List is
    num_transistors = 5.88e4;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Free_List;

entity block Grad_Unit is
    num_transistors = 1.578e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Grad_Unit;

entity block Instr_Fetch_Addr is
    num_transistors = 9.9e4;
    external_rent = 0.60;
```

```
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Instr_Fetch_Addr;

entity block Instr_Fetch_DP is
    num_transistors = 8.28e4;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Instr_Fetch_DP;

entity block Instr_Fetch_Cntrl is
    num_transistors = 5.7e4;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Instr_Fetch_Cntrl;

entity block Addr_Queue is
    num_transistors = 1.32e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Addr_Queue;

entity block Instr_Decode is
    num_transistors = 2.71e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    activity_factor = 15.0;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Instr_Decode;

entity block Int_DP is
    num_transistors = 2.62e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Int_DP;
```

```
entity block Int_Queue is
    num_transistors = 1.182e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    activity_factor = 15.0;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Int_Queue;

entity block Float_DP is
    num_transistors = 1.956e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    activity_factor = 12.0;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Float_DP;

entity block Float_Queue is
    num_transistors = 3.06e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    activity_factor = 12.0;
    LDI_Scheme = 2;
    Target = 1.0;
    Opt_Method = 0;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Float_Queue;

entity block Float_Mult is
    num_transistors = 1.158e5;
    external_rent = 0.60;
    ext_rent_const = 3.20;
    activity_factor = 12.0;
    LDI_scheme = 2;
    internal_rent = 0.6;
    rent_const = 3.2;
end entity Float_Mult;

entity BUS sys_bus is
    physical_width = 2.5 [um];
    logical_width = 32 [bits];
    aspect_ratio = 1.1 [ ];
```

```
    spacing = 1.5 [um];
    activity_factor = 0.15 [ ];
    driver_size = 25.0 [w/l];
    bus_list (float_q,float_mult,float_dp,int_dp,int_q,grad_unit,cache_cntrl);
end entity sys_bus;

entity system RISC_CHIP is
    block I_Cache i_cache;
    block D_Cache d_cache;
    block I_Cache_Tag i_tag;
    block D_Cache_Tag d_tag;
    block TLB tlb;
    block Cache_Cntrl cache_cntrl;
    block Ext_Interface ext_inter;
    block Sys_Buffers sys_buffer;
    block Free_List free_list;
    block Grad_Unit grad_unit;
    block Instr_Fetch_Addr int_addr;
    block Instr_Fetch_DP int_fetch_dp;
    block Instr_Fetch_Cntrl int_fetch_cntrl;
    block Addr_Queue addr_q;
    block Instr_Decode instr_decode;
    block Int_DP int_dp;
    block Int_Queue int_q;
    block Float_DP float_dp;
    block Float_Queue float_q;
    block Float_Mult float_mult;
end entity RISC_CHIP;

Megacell placement is
    grid_value = 70;
    placement_type = manual;
    constraint = inorder;
    i_cache (69,1:48,26);
    d_cache (54,44:33,69);
    ext_inter (69,27:63,54);
    sys_buffer (69,55:55,69);
    i_tag (63,27:55,37);
    cache_cntrl (63,38:55,54);
    d_tag (54,27:42,34);
    tlb (54,35:33,43);
    int_addr (47,13:33,26);
    int_fetch_dp (47,1:33,12);
    int_fetch_cntrl (32,1:24,12);
    free_list (32,13:22,22);
```

```
    grad_unit (32,23:22,41);
    addr_q (32,42:20,56);
    int_dp (32,57:1,69);
    instr_decode (21,13:12,41);
    int_q (19,42:12,56);
    float_mult (15,1:1,12);
    float_dp (11,13:1,41);
    float_q (11,42:1,56);
end placement;
```

REFERENCES

[1] S. Morse, B. Ravenel, S. Mazor, and W. Pohlman, "Intel microprocessors 8008-8086," *Computer*, vol. 13, pp. 42–60, Oct. 1980.

[2] S. Naffziger and G. Hammond, "The implementation of the next-generation 64b itanium microprocessor," *Proc. 2002 Int. Solid-State Circuits Conf.*, vol. 1, pp. 344–472, Feb. 2002.

[3] C. Semiconductor Research Association, San Jose, "International technology roadmap for semiconductors," tech. rep., 2003.

[4] M. Tremblay, G. Grohoski, B. Burgess, *et al.*, "Challenges and trends in processor design," *Computer*, vol. 31, pp. 39–48, Jan. 1998.

[5] G. Gao, "Bridging the gap between isa compilers and silicon compilers: a challenge for future soc design," *Int. Sym. on System Synthesis*, p. 93, 2001.

[6] K. Ruparel, "Soc test: the devil is in the details," *Proc. Int. Test Conf.*, p. 1143, 2001.

[7] J. Meindl, "Low power microelectronics: Retrospect and prospect," *Proc. of the IEEE*, vol. 83, pp. 619–635, Apr. 1995.

[8] J. Eble, V. De, and J. Meindl, "A first generation generic system simulator (genesys) and its relation to the itrs," *Proc. 11th Biennial University/Goverment/Industry/Microelectronics Symposium*, pp. 147–154, 1995.

[9] J. Davis, V. De, and J. Meindl, "A stochastic wire-length distribution for gigascale integration(gsi). i. derivation and validation," *IEEE Transactions on Electron Devices*, vol. 45, pp. 580–9, Mar. 1998.

[10] J. Davis, V. De, and J. Meindl, "A stochastic wire-length distribution for gigascale integration(gsi). i. applications to clock frequency, power dissipation, and chip size esitmation," *IEEE Transactions on Electron Devices*, vol. 45, pp. 590–7, Mar. 1998.

[11] D. Slyvester and C. Hu, "Analytical modeling and characterization of deep-submicron interconnect," *Proc. of the IEEE*, vol. 89, pp. 634–663, May 2001.

[12] H. Bakoglu and J. Meindl, "A system level circuit model of multi- and single chip cpus," *1987 Int. Solid State Circuits Conf.*, pp. 308–309 439–440, 1987.

[13] G. Sai-Halasz, "Performance trends in high-end processors," *Proc. of the IEEE*, vol. 83, pp. 20–36, Jan. 1995.

[14] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, and W. Specks, "Optimization of high-speed cmos logic circuits with analytical models for signal delay, chip area, and dynamic power dissipation," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 236–47, Mar. 1990.

[15] A. Goel and F. Schuermeyer, "Nchipsim - a microcomputer simulator of nmos chip performance indicators," *Proc. First Great Lakes Symposium on VLSI*, pp. 332–333, 1991.

[16] B. Landman and R. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. on Computers*, vol. C-20, pp. 1469–79, Dec. 1971.

[17] W. Donath, "Placements and average interconnection lengths of computer logic," *IEEE Trans. on Circuits and Systems*, vol. CAS-26, pp. 272–277, Apr. 1979.

[18] R. Mangaser and K. Rose, "Facilitating interconnect-based vlsi design," *Proc. 1997 IEEE Int. Conf. on Microelectronic Systems Education*, pp. 139–140, 1997.

[19] P. Ghosh, R. Mangaser, C. Mark, and K. Rose, "Interconnect dominated vlsi design," *Proc. 20th Annual Conf. on Advanced Research in VLSI*, pp. 114–121, 1999.

[20] S. Mohanty and V. Prasanna, "Rapid system-level performance evaluation and optimization for application mapping onto soc architectures," *Proc. IEEE Int. ASIC/SoC Conf.*, vol. 15, pp. 160–67, Sept. 2002.

[21] J. Eble, V. De, and J. Meindl, "A generic system simulator (genesys) for asic technology and architecture beyond 2012," *Proc. 9th Annual IEEE ASIC Conference*, pp. 193–196, 1996.

[22] S. Nugent, J. Eble, D. Wills, and J. Meindl, "An ultra-compact empirical model for throughput projection for gigascale integration," *Proc. TechCon 2K*, Sept. 2000.

[23] C. Seminconductor Research Association, San Jose, "International technology roadmap for semicondictors," tech. rep., 2001.

[24] G. Sai-Halasz, "Performance trends in high-end processors," *Proc. of the IEEE*, vol. 83, pp. 20–36, Jan. 1995.

[25] J. Eble, *A Generic System Simulator with Novel On-Chip Cache and Throughput Models for Gigascale Integration.* PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, 1998.

[26] A. Smith, "Cache evaluation and the impact of workload choice," *Intl. Symposium on Computer Architecture*, pp. 64–73, June 1985.

[27] A. Smith, "Line (block) size choices for cpu cache memories," *IEEE Transactions on Computers*, vol. C-36, pp. 1063–1075, Sept. 1987.

[28] M. Flynn, *Computer Architecture: Pipelined and Parrallel Processor Design.* Boston, MA: Jones and Bartlett, 1995.

[29] J. D. Plummer, M. D. Deal, and P. B. Griffin, *Silicon VLSI Technology: Fundamentals, Practice, and Modeling.* New Jersey: Prentice Hall, 2000.

[30] P. R. Gray and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits.* New York: John Wiley & sons, inc., 1993.

[31] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach.* San Francisco, CA: Morgan Kaufmann, 1996.

[32] J. Davis, R. Venkatesan, A. Kaloyeros, M. Bylansky, S. Souri, K. Banerjee, A. Rahman, A. Reif, and J. Meindl, "Interconnect limits on gigascale integration (gsi) in the 21st century," *Proc. IEEE*, vol. 89, pp. 305–324, Mar. 2001.

[33] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990.

[34] P. Zarkesh-Ha, J. Davis, and J. Meindl, "Prediction of global net-length distribution for global interconnects in a heterogeneous system-on-a-chip," *IEEE Transactions on Very Large Scale Integration*, vol. 8, pp. 649–659, Dec. 2000.

[35] N. Vasseghi, K. Yeager, E. Sarto, and M. Seddighnezhad, "200-mhz superscalar risc microprocessor," *IEEE Journal of Solid State Circuits*, vol. 31, pp. 649–59, Nov. 1996.

[36] D. Lundell, "Squeesing through the von neumann bottleneck," *Available online at: http://www.knozall.com/squeezingthrougthevonneuman.htm*, 2001.

[37] T. Womack, "Tom's x86 faq," *Available online at: http://www.tom.womack.net/x86FAQ/faq_time.html*, 2004.

[38] W. Badawy, "System-on-chip: Issues, challenges and trends," *Canadian Journal of Electrical and Computer Engineering*, vol. 26, pp. 85–90, Oct. 2001.

[39] IBM, "Coreconnect bus architecture," *Available Online: http://www-306.ibm.com/chips/products/coreconnect/.*

[40] B. Cordan, "An efficient bus architecture for systems-on-chip design," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 623–6, 1999.

[41] R. Weiss, "Advanced microprocessor bus architecture (amba) bus system," *Electronic Design*, vol. 49, pp. 114–15, Mar. 2001.

[42] R. Bashirullah, “Low power design methodology for an on-chip bus with adaptive bandwidth capability,” *Design Automation Conf.*, pp. 628–33, June 2003.

[43] IBM, “Powerpc 440 embedded processor,” *Available Online: http://www-306.ibm.com/chips/products/powerpc/processor/processors.html*.

[44] W. Lee and C. Hu, “Modeling cmos tunneling current through ultrathin gate oxide due to conduction- and valence-band electron and hole tunneling,” *IEEE Transactions on Electron Devices*, vol. 48, pp. 1366–72, July 2001.

[45] J. Chang, “A 0.13um triple-vt 9mb third level on-die cache for the itanium2 microprocessor,” *Proc. IEEE Int. Solid State Circuits Conf.*, pp. 496–52, 2004.

[46] T. Shimizu and J. Korematu, “A multimedia 32b microprocessor with 16mb dram,” *Proc. IEEE Int. Solid State Circuits Conf.*, pp. 216–217, 1996.

[47] J. Gebis, “Trends in merged dram-logic computing,” *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 4109, pp. 198–205, July 2000.

[48] K. Hardee, F. Jones, D. Butler, M. Parris, and M. M. et al, “A 0.6v 205mhz 19.5ns trc 16mb embedded dram,” *IEEE Int. Solid-State Circuits Conf.*, pp. 320–322, Feb. 2004.

[49] J. Mandelman and R. Denard, “Challenges and future directions for the scaling of dynamic random access memory (dram),” *IBM Journal of Research and Development*, no. 2/3, pp. 187–212, 2002.

[50] T. Rajeevakumar and G. Bronner, “A novel trench cpacitor structure for ulsi drams,” *Symposium on VLSI Technology*, pp. 7–8, May 1991.

[51] B. Austin, K. Bowman, X. Tang, and J. Meindl, “A low power transregional mosfet model for complete power-delay analysis of cmos gigascale integration,” *Proc. Int. ASIC Conf.*, pp. 125–9, Sept. 1998.

[52] P. Kelcher, S. Richardson, and S. Siu, “An equal area comparison of embedded dram and sram memory architectures for a chip multi-processor,” tech. rep., HP Labs Palo Alto, Apr. 2000.

[53] S. Nugent, D. Wills, and J. Meindl, “A hierarchical block-based modeling methodology for soc in genesys,” *Proc. IEEE Int. ASIC/SoC Conf.*, vol. 15, pp. 239–243, Sept. 2002.

[54] S. Rusu, “Vlsi scaling trends and challenges,” *IEEE int. symposium on circuits and systems*, pp. 10.1.1–10.1.10, May 2001.

[55] P. Zarkesh-Ha, *Global Interconnect Modeling for a Gigascale System-on-a-Chip (GSoC)*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, 2001.

[56] M. Ghoneima, “Optimum positioning of interleaved repeaters in bi-directional buses,” *Design Automation Conf.*, pp. 586–91, June 2003.

[57] T. Inaba, "A 250mv bit-line swing scheme for 1-v operating gigabit scale drams," *IEICE Transactions on Electronics*, vol. E79-C, pp. 1699–706, Dec. 1996.

[58] J. Poulton, "An embedded dram for cmos asics," *Proceedings. 17th Conf. on Advanced Research in VLSI*, vol. 17, pp. 288–302, Sept. 1997.

[59] K. Ryu, E. Shin, and V. Mooney, "A comparison of five different multiprocessor soc bus architectures," *Proc. Euromicro symposium on digital systems design*, pp. 202–9, 2001.

VITA

Steven Paul Nugent was born in Charelston S.C. to Michael and Beryl Nugent in September 1973. His family moved to Lawrenceville, Georgia in 1977, where he attended high school graduating in 1993. In December 1997 he received a Bachelor in Electrical Engineering with highest honors from the Georgia Institute of Technology. In May of 2002 he graduated with a designated Masters degree in Electrical Engineering. While in graduate school, he worked as a graduate research assistant in the Gigascale Integration Group in the School of Electrical and Computer Engineering at Georgia Tech. His research interests include computer architecture, semiconductor devices, and VLSI circuit design. His doctoral research focused on the extension of a generic system simulator, GENESYS, to heterogeneous systems-on-a-chip for projecting key performance metrics for future billion transistor systems.