



AFRL-RI-RS-TR-2012-134

EXPLORATION OF NANOMETER COGNITIVE REASONING VERY LARGE SCALE INTEGRATION (VLSI) COMPUTER ARCHITECTURES

OKLAHOMA STATE UNIVERSITY

APRIL 2012

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2012-134 HAS BEEN REVIEWED AND IS APPROVED FOR
PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ s /
THOMAS RENZ
Work Unit Manager

/ s /
PAUL ANTONIK, Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE*Form Approved*
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**1. REPORT DATE (DD-MM-YYYY)**

APR 2012

2. REPORT TYPE

Final Technical Report

3. DATES COVERED (From - To)

NOV 2008 – NOV 2011

4. TITLE AND SUBTITLE**EXPLORATION OF NANOMETER COGNITIVE
REASONING VERY LARGE SCALE INTEGRATION
(VLSI) COMPUTER ARCHITECTURES****5a. CONTRACT NUMBER**

FA8750-09-2-0036

5b. GRANT NUMBER

N/A

5c. PROGRAM ELEMENT NUMBER

61102F

6. AUTHOR(S)

James E. Stine, Jr.

5d. PROJECT NUMBER

OKLC

5e. TASK NUMBER

CC

5f. WORK UNIT NUMBER

09

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)Oklahoma State University
School of Electrical and Computer Engineering
202 Engineering South
Stillwater, OK 74078**8. PERFORMING ORGANIZATION
REPORT NUMBER**

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)Air Force Research Laboratory/RITB
525 Brooks Road
Rome NY 13441-4505**10. SPONSOR/MONITOR'S ACRONYM(S)**

AFRL/RI

**11. SPONSORING/MONITORING
AGENCY REPORT NUMBER**

AFRL-RI-RS-TR-2012-134

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88 ABW-2012-2306

Date Cleared: 16 Apr 2012

13. SUPPLEMENTARY NOTES**14. ABSTRACT**

The objectives of this work were to design, develop, and evaluate support for the design of low-power hardware computer architectures at the Very Large Scale Integration (VLSI) level. The objectives were realized by achieving complete design flow integration with commercial and open-source Electronic Design Automation tools. The design flow takes as inputs a high-level system-level architecture description, along with area, critical path delay, and power dissipation constraints. Based on the System on Chip architecture description and design constraints, the tools automatically generate synthesizable Hardware Descriptive Language (HDL) models, embedded memories, and custom components to implement the specified VLSI architecture. Simulation results showed significant improvement over previous approaches with respect to power dissipation and leakage reduction.

15. SUBJECT TERMS

Design Flow, Design Tools, Electronics Design Automation tools, VLSI design, System on a Chip design, Low Power Processor design

16. SECURITY CLASSIFICATION OF:**17. LIMITATION OF
ABSTRACT****18. NUMBER
OF PAGES****19a. NAME OF RESPONSIBLE PERSON**

THOMAS E. RENZ

a. REPORT

U

b. ABSTRACT

U

c. THIS PAGE

U

UU

28

19b. TELEPHONE NUMBER (Include area code)

N/A

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES.....	ii
1. SUMMARY	1
2. INTRODUCTION.....	2
3. METHODS, ASSUMPTIONS and PROCEDURES	5
3.1 IBM Standard Cell Libraries	8
3.2 Architecture Design Decisions for Lower Power Dissipation.....	15
4. RESULTS AND DISCUSSION	18
5. CONCLUSIONS	20
6. REFERENCES.....	21
LIST OF ACRONYMS	23

LIST OF FIGURES AND TABLES

Figure 1. Pitch Examples	10
Figure 2. Stacked Vias	11
Figure 3. Example Pitch Definitions.....	11
Figure 4. Proposed Design Flow using Commercial-based EDA tools.....	13
Figure 5. 32-bit x 32 word Single Write/Read Port Layout.....	14
Figure 6. Place/Routed 500 MHz MIPS Processor Using Proposed Design Flow.....	17
Figure 7. Final Layout for July 2010 T using IBM CMOS 10LPe Technology.....	19
Table 1. IBM CMOS10SF 65 nm Pitch Definitions.....	13
Table 2. Basic Foundation Flow Script Flow (in order of running)	16
Table 3. Area/Delay Results for Sample Designs.....	18

1. SUMMARY

The objectives of this work were to design, develop, and evaluate support for design of low-power hardware computer architectures at the Very Large Scale Integration (VLSI) level. The objectives were realized by achieving complete design flow integration with commercial and open-source Electronic Design Automation tools. The design flow takes as inputs a high-level system-level architecture description, along with area, critical path delay, and power dissipation constraints. Based on the System on Chip architecture description and design constraints, the tools automatically generate synthesizable Hardware Descriptive Language (HDL) models, embedded memories, and custom components to implement the specified VLSI architecture. Simulation results showed significant improvement over previous approaches with respect to power dissipation and leakage reduction.

2. INTRODUCTION

Advances in parallel processing, Very Large Scale Integration (VLSI) technology, and computer architecture have led to high performance computer systems. Modern microprocessors are capable of executing over several hundred million floating-point operations per second, and supercomputer systems, capable of executing more than a trillion operations per second, are currently being developed [1]. Along with this, greater reliance is being placed on the simulation results produced by these computer systems. Computer simulations play an important role in almost every aspect of science and engineering, including research in computational fluid dynamics, weather forecasting, VLSI circuit design, manufacturing, and modeling in chemistry, physics, and biology.

The large number of arithmetic operations and the reliance placed on computer systems make it extremely important to design, evaluate, and implement architectures efficiently for a given objective. Unfortunately, most computer systems are complicated by the requirement to provide significant processing performance while still maintaining adequate environmental constraints for power dissipation and power/clock distribution. This is further complicated by the fact that many systems are now resorting to an architecture with more than one core in parallel in order to increase its performance capabilities. These multi-core systems use complex interconnection networks to parallelize code and increase latency performance. To overcome the challenge of designing complex architectures, several software tools for creating silicon implementations of computer architectures and digital systems have been developed. Any one tool covers just a small part of the simulation and design process needed to create a complete chip design. A design flow is a combination of tools to cover a significant portion of the design process. These tools include design flows for sub-micron technologies targeted at education repositories, such as Metal Oxide Semiconductor Implementation Service (MOSIS) generic technology tool sets for corporate instruction, and research-targeted, variation-aware nanometer generic technologies, and limited public-domain toolsets using open-source Electronic Design Automation (EDA) tools.

The main disadvantage of EDA tools for the implementation of VLSI architectures is that they don't readily support flexible design flows for the implementation of high performance architectures. Corporations that specialize in the design and implementation of high performance computer architectures usually have an inordinate number of engineers at their disposal in order to target systems for a given performance metric. However, sharing or dissemination of the EDA tool integration methods and specific design flows corporations use to create a desired computer architecture objectives are not available for the general public, making further improvements upon computer architectures difficult or impossible to reproduce. More importantly, many corporations can afford to sponsor application engineers (AE), from the EDA software vendors, on site to assist in completing a given design objective.

According to International Technology Roadmap for Semiconductors (ITRS) 2003 [2], two of the most daunting System on Chip (SOC) challenges are (1) design productivity improvement by more than 100% per technology node and (2) management of power

especially for low-power, wireless, multimedia applications. Current design tools and flows, however, require excessive time, effort and design costs to achieve the power, area, and performance requirements of future SOC solutions. The semiconductor industry needs new design techniques, tools and flows for high-level synthesis that provide the ability to quickly develop and evaluate complex SOC solutions. These techniques, tools and flows should provide accurate area, delay, and power estimates from high-level SOC architecture descriptions to facilitate design space exploration and the evaluation of new power management techniques. They should also provide support for a wide variety of components (e.g., standard cell and custom circuits, programmable processors, embedded memory, and buses) and facilitate design reuse. Furthermore, the design tools and flows should be well documented, easy to use, and publicly available to encourage new research and innovation in SOC design.

Presently, there are several tools available including the SimpleScalar architecture simulator [3] and Watch Power Analysis Framework [4] for estimating the performance or power dissipation of processors and their memory system. These tools, however, do not facilitate accurate modeling of complete SOC solutions, which often include hardware accelerators, peripherals, and bus interfaces. Furthermore, since these tools do not provide access to information about the underlying circuit and silicon implementation, the accuracy with which they can model dynamic and static power dissipation is significantly limited. Consequently, accurate modeling and evaluation of complex systems has been marked as a significant challenge for system architecture and digital system designers by a recent National Science Foundation panel [5]. Ultimately, the NSF panel argues that simulation and benchmarking will require a leap in capability within the next few years to maintain ongoing innovations in computer systems.

This paper details the research and development of high-level synthesis tools for System-on-Chip platforms, specifically targeted at IBM 65 nm Complementary Metal Oxide Semiconductor (CMOS) technologies, that (1) provide the ability to efficiently integrate embedded memories, low-power/high-performance circuits and processors, and communications structures, (2) combine synthesis and layout information to accurately estimate area, delay, and power from high-level SOC architecture descriptions, (3) facilitate rapid design-space exploration of SOC solutions, and (4) are well-documented, easy-to-use, and publicly available for the Air Force Research Laboratory (AFRL) personnel. It is also anticipated that many of the outcomes of this project will aid in the development and deployment of future silicon architectures for any user that employs trusted foundry fabrication capabilities.

This paper describes a number of techniques and design flows related to lowering power dissipation within the design of computer architectures resulting from work done at the AFRL in Rome, New York from 2007 until 2011. In this paper, a low-power digital standard-cell library and design flow is discussed for IBM's 65 nm CMOS technology and its use in the creation of low-power structures in the design of high-performance computer architectures. Several techniques, both circuit and architecture-based, are discussed and results are shown using the design flow implemented for this research. Specifically, multi-threshold CMOS (MTCMOS) is implemented within the library and

compared versus a library supplied by ARM that only works with regular threshold voltage transistors.

This report is organized as follows. Section 3 discusses the overall effort and the background behind power dissipation and its relationship to design flows for nanometer technologies. . Section 3.1 discusses a standard-cell library in IBM CMOS 10SF 65 nm feature sizes for complex CMOS technologies, (10SF denoted the specific design technology for 2010 fabrication runs) and the library's effect on routing methodology choice. Section 3.2 discusses some of the architecture ideas utilized to lower the total power dissipation within digital circuits. Section 4 discusses the results from these design choices and compares them versus the IBM-supplied ARM library. Finally, Section 5 presents some brief conclusions.

3. METHODS, ASSUMPTIONS and PROCEDURES

The design and fabrication process used to produce silicon structures for state of the art computer architectures is long and complicated. During the 1970s and up through the early 1990s, most new computer architectures created were for ever higher performance application specific systems or for high performance general processors [5]. In the late 1980's, Field Programmable Gate Arrays (FPGAs) were created. Although FPGAs are easy and far cheaper to design than most traditional computer architectures, they consume significantly more area, delay and power for a given performance [6]. Consequently, for computer applications which demand a high amount of performance, traditional but high design complexity architectures continue to be used.

The demand for increased speed, decreased energy consumption, improved memory utilization, and better compilers for processors has become paramount to the design of the next generation of computer architectures. To make matters worse, the traditional challenges of designing digital devices with semiconductor technology has drastically changed with the introduction of deep submicron technology. Designs that have long been expanding in complexity through Moore's Law have run up against severe technological limitations below 130 nm. Where once it was easy to improve a design by scaling the minimum feature size of a transistor, new basic features are now required. Quality simulation adds new levels of design complexity.

Because silicon technologies are so small, designs can now implement millions and even billions of transistors on a reasonably small die. Unfortunately, this leads to power density and total power dissipation that is at the limits of what packaging, cooling, and other infrastructure can support [8]. More importantly, in CMOS technologies below 90 nm, leakage current almost matches or surpasses that of dynamic power, making power dissipation a major obstacle to designing complex SOC designs.

Although power dissipation complicates the design process for integrated circuits it does not necessarily mean that designs cannot be efficiently accomplished. Energy consumption and clock speed are closely linked and engineering choices or sacrifices are normally required if a design requires a lower power factor or high clock rates.

Today some of the highest power microprocessors can dissipate close to 150 watts in an area close to 1 cm square [8]. In fact, as the International Technology Roadmap for Semiconductors (ITRS) predicted, the power for microprocessor reached a maximum of 198 watts in 2008 [2]. For virtually all applications, including general-purpose computer architectures, reducing the power consumed by an SOC is essential to allow new features and add performance to improve technology. Consequently, it is important to understand how power consumption affects SOC designs to improve upon power efficiency.

The total power consumption of a digital logic circuit consists of two major factors. The first part consists of dynamic power which is the power that is consumed when a device is active. Typically, dynamic power is consumed when devices are active and are switching back and forth. That is, they are based on what is supplied at the input of a

circuit. If, for example, a circuit has lots of activity (e.g. within a router for the Internet), it will typically consume lots of dynamic power. Conversely, applications that only switch on during critical events, (e.g. sensors for abnormal events within automobiles), typically consume very low amounts of dynamic power.

Dynamic power's main impetus is the amount of switching that occurs during an event. Since most CMOS circuits are composed of layers of Silicon Dioxide, which is an excellent charge storage material, a majority of the switching power stems from the power that is charged and discharged in the process of turning the transistor on and off, respectively. This results in dynamic power with a squared dependence on the voltage:

$$P_{dyn} = C_L \cdot V_{DD}^2 \cdot P_{trans} \cdot f_{clock} \quad (1)$$

where C_L is the load capacitance, V_{DD} is the supply voltage, f is the frequency of the system clock, and P_{trans} is the probability of an output transition. In addition to switching power, internal power also contributes to dynamic power. Internal power is related to the speed with which CMOS gate is turned from on to off and back to on. This switching causes both NMOS and PMOS transistors to be ON momentarily resulting in a short circuit or "crowbar" current. Although the short circuit period can be small, it can contribute to the total dynamic power if the input is ramped up too quickly [10]. Internal or short-circuit power can be described as:

$$P_{ss} = t_{sc} \cdot V_{DD} \cdot I_{peak} \cdot f_{clock} \quad (2)$$

where t_{sc} is the time duration of the short-circuit current and I_{peak} is the total internal switching current. Although short-circuit current will not be discussed in this paper, for lower-power consumption it is important to make sure gates are not floating on an output when turning on certain power-gating circuits. That technique is discussed later, in the results section.

The second major power factor, static power dissipation, is the power consumed when devices are powered on but no signals are changing values. In the past, static power dissipation, which is mainly dominated by leakage current in a gate, was either non-existent or did not significantly impact a design. However, as the voltage and minimum feature size of a transistor became smaller, the pronounced effect of leakage within a gate made static power dissipation greater than dynamic power dissipation below the 90nm technology node [9].

In the past, traditional designs resorted to lowering the power supply voltage to get an exponential decrease in the power. This decision has been substantiated by Equation 1's power dissipation being dependent on the square of the supply voltage. The real problem is that lowering the supply voltage causes the drive or drain to source current of a transistor to decrease. The drain to source current can be approximated by:

$$I_{DS} = \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot \frac{(V_{GS} - V_T)^2}{2} \quad (3)$$

where μ is the carrier mobility, C_{ox} is the gate capacitance, W and L are the dimensions of the transistor, V_T is the threshold voltage, and V_{GS} is the gate-source voltage. Since deep submicron technologies have low supply voltages, having a low threshold voltage allows CMOS designs to maintain good performance [10]. Unfortunately, as the threshold voltage gets smaller, an exponential increase in the sub-threshold leakage current (I_{SUB}) occurs.

The subthreshold leakage current is the dominant element of static power dissipation. It occurs when a CMOS gate is not turned completely off. A good approximation to the subthreshold equation is shown in Eq. 4, where k is Boltzmann's constant, T is the temperature in kelvin, q is the charge of an electron, and n is a function of the device fabrication process. The subthreshold leakage current for sub-90 nm transistors is a major source of concern within current technologies, such as the IBM CMOS10SF 65 nm technology originally considered for this work.

$$I_{SUB} = \mu \cdot C_{ox} \cdot \frac{k \cdot T}{q} \cdot \frac{W}{L} \cdot e^{\frac{(V_{GS} - V_T) \cdot q}{n \cdot k \cdot T}} \quad (4)$$

Eq. 4 indicates that sub-threshold leakage, which is the predominant factor in static power dissipation, depends exponentially on the difference between V_{GS} and V_T . Therefore, as technology scales the power supply and V_T down to limit the dynamic power, leakage power grows exponentially, as was shown in [9]. To make matters worse, sub-threshold voltage current increases exponentially with temperature, which also complicates the process for low-power design.

Transistors are usually defined by their length and width, however, the length usually establishes the minimum feature size of a transistor [6]. As technology moves towards smaller feature sizes, the thickness of the oxide below the gate of a transistor also decreases in thickness. Unfortunately, in current semiconductor processes the thickness of the oxide is only several atoms thick. Consequently, the thinness of the oxide allows a current that tunnels through the gate towards the channel of a transistor, so much so that in current sub 90 nm technologies, gate leakage can be nearly 1/3 as much as sub-threshold leakage [8]. In order to reduce the gate leakage, some manufacturers have resorted to high-k dielectric materials, such as Hafnium, to keep the gate leakage in check [11].

Another technique to reduce the leakage current is to use multi threshold voltage transistors. Using this technique, high V_T cells can be utilized wherever performance goals allow the power dissipation to be kept in check. And, lower V_T cells can be used on a critical path to meet a specific timing. Specifically, having transistors that can utilize different threshold voltages allows the reduction of the substrate current, as shown in Eq. 4. The technology for IBM's CMOS10LPe (2010 Low Power) 65 nm process utilized for this work enabled the use of Regular V_T , High V_T , and Low V_T transistors to reduce the gate leakage [12].

3.1 IBM Standard Cell Libraries

A high priority of any computer architecture created in silicon is to facilitate “paper to product” in an efficient and pragmatic method. To accomplish this goal many designs are created through complex and elaborate software programs that write netlists or a structural description of silicon objects by means of a concise software system. When engineers first started creating these silicon structures, the number of transistors was small and integration that existed within a system design was simple and straightforward. However, as the complexity of computer architectures increased, many of the software tools became extremely elaborate and complex but narrowly focused on one or just a few aspects of the complete architecture design. Therefore, engineers must expend significant energy and time to design efficient streams of design tools, or design flows, to accomplish the task of producing Very Large Scale Integration (VLSI) architectures.

Many of these design flows involve translating structural or behavioral descriptions of computer architecture into a working silicon mask layer that can be fabricated. Although this process is just an evolution of processing one netlist to another, the process has dramatically changed from early designs involving several hundred transistors to current SOC designs that encompass close to or exceed 1 billion transistors [14]. To make matters worse, power and performance optimization issues have complicated the entire process [15].

Standard-cell designs involve taking pre-made layout elements, such as an AND or NAND gate, and having software stitch the elements together via placing each routing wire between known pins. Early layout editors, such as the Magic Layout Editor, had built-in routers to allow designers to avoid having to worry about laying out wire between two points [16]. However, as more points and pins were created, the performance cost for a given route increased and there was a dramatic need for more efficient algorithms to deal with congestion and optimization [17].

Although the use of standard-cell libraries is not new, their implementation into a design flow has not been widely studied, mainly because much of the design process has been become a black art. In fact, many of the standard cells from initial creators can only be utilized by instantiating or inserting their layout at the foundry when a design is complete. Details of the standard cell’s characteristics are not given to designers outside the foundry. A designer, who wants to create a new layout library, must learn all the ins and outs of a given standard cell by trial and error. Unfortunately, most designers in industry cannot afford this time consuming process and, thus, resort to using libraries that are purchased from companies that specialize in this area.

Fortunately, there are several limited design flows available, often free of charge from educational institutions. Unfortunately, most of these design flows only contain a limited number of standard cells and components, and are not geared toward complete SOC-based design. Virginia Tech offers an extensive library for the Taiwan Semiconductor Manufacturing Corporation, TSMC 0.25 μm process [18]. Although these standard-cell libraries are helpful at learning some of the basics, they tend to skip over many of the details of more proprietary, optimized commercial standard-cell libraries. Furthermore, since current design flows are often not well integrated with design tools for high-level

synthesis or design space exploration, there is a significant barrier to high-quality research and education in this area. This project began with adapting standard-cells using the IBM CMOS10SF 65 nm process while utilizing the author's experience in creating design flows both academically [19-22] and with industrial partners Cadence Design Systems, Synopsys, and Mentor Graphics.

The IBM CMOS 10SF is a 65 nm fabrication process created by IBM. This process provides 9 metal layers (M1, M2, M3, M4, B1, B2, B3, EA, EB) plus LB (for transfer off chip) to DV (glass cut). The technology is a nanometer-based technology that gives high-performance, while still trying to maintain somewhat low power designs. IBM originally created this process for static random access memory, digital logic, mixed-signal, and mixed-voltage input/output applications [23]. Although this technology is robust, it presents certain challenges, since its operating voltage is relatively low at a V_{DD} of 1.0 Volt.

Standard-cell libraries usually utilize two methodologies for determining where a pin can route. The first methodology is called "grid-less placement and routing" which treats every routing point as a Cartesian coordinate [23], [24]. Although this method accomplishes the task well with high-performance computers capable of processing billions of operations per second, its algorithm has a high processing overhead. Consequently, grid-less placement and routing tends to be avoided in commercial placement and routing software tools. Although grid-less placement and routing works well for grid-based placement and routing, implementation is an NP-hard problem that can cause problems for cases when limited area or high amounts of constraints are given for a problem [24]. On the other hand, "grid-based placement and routing" works well with most tools and can handle large numbers of transistors with a surprising amount of efficiency. Therefore, for this work, grid-based placement and routing was chosen to minimize the complexity as well as have the ability to interface to commercial-based EDA tools.

Grid-based placement and routing is organized into grids that are chosen by design. It is important that there is a sufficient size for each grid so that wires can follow the grid and connect around each junction. For this paper, it is assumed that connections are orthogonal, however, they can easily be non-orthogonal with more complexity added to the placement and routing algorithms. Standard-cell libraries are designed such that each cell is set at the same height. Although cells can be at different heights, placement and routing can get more congested with this approach. Since most libraries are designed such that they have many gates within them, including ones with different drive strengths, it is important to make the height of the cell relatively large. However, making the height too large can cause low amounts of density within the design.

Placement and routing algorithms work with two basic elements to create a connection. The first element is a contact, called a via, where a large majority of connections are between two metal layers. The second element is wires, which connect between each pin or contact. A contact or via is utilized to connect to a pin, which is either a final destination or beginning of a connection, or at an intermediate connection that may be part of a larger route. To set the correct grid distance, or pitch, between two wires, it is important to factor the via and wire rules into the computation. There are two basic rules

related to pitch. The first pitch rule is called via-to-via distance and although it is a common definition for pitch, it can waste space between two wires that do not have a connection. Consequently, for this project we chose a line-to-via pitch to allow the wires to be closer together and allow more routing within a given area. Line-to-via is more common in industry. Figure 1 shows an example of the two pitch designations.

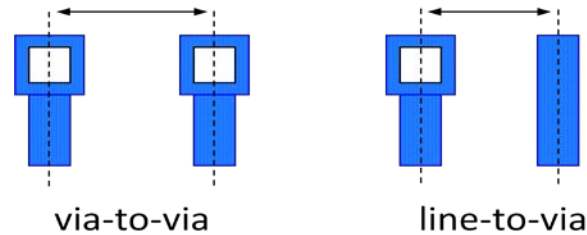


Figure 1. Pitch Examples

In this design process, all grids are assumed to be equal for both horizontal and vertical distances. This was chosen to simplify the computation for each distance, however, many design flows use different dimensions for pitches in the 2D horizontal and vertical directions for individual metal layers. To make the wires less congested, each wire in a layer is assumed to navigate either horizontally or vertically and subsequent layer wires are at 90 degrees, resulting in either HVH (Horizontal-Vertical-Horizontal) or VHV (Vertical-Horizontal-Vertical) routing. In this work, odd metal layers are assumed to be routed horizontally (e.g. metal1), whereas, even metal layers are assumed to be routed vertically (e.g. metal4). That is, a HVH routing methodology is utilized.

Contacts or vias are usually connected so that they minimize the connection between two wires [6]. In order to save space it is preferable to have contacts, between metal layers that are more than one metal layer apart, aligned above each other regardless of the metal layer. For example, if a route is to connect between metal4 and metal1, it would be easiest to have the connection happen straight downwards through the levels of metal 3 and metal 2. Because this connection happens on-top of each layer, it is called a stacked via. In early designs, such as those with only 2-3 layers of metal, stacked vias were physically difficult to implement. However, with newer technology and machinery to fabricate integrated circuits, stacked vias are extremely common. The IBM CMOS 10SF 65 nm process has stacked vias, therefore, the design is chosen such that contacts or vias can be situated on top of each, as shown in Figure 2.

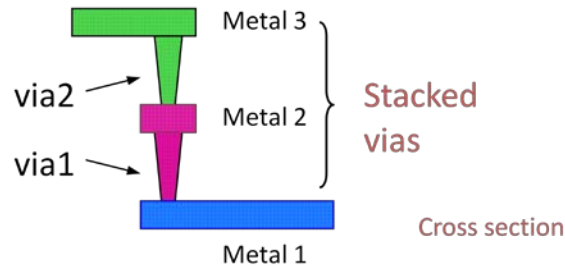


Figure 2. Stacked Vias

Although grids are common in standard-cell designs, designers of common standard-cell libraries tend to forget to keep the grids in multiples of each other for each subsequent metal layer. Consequently, some metal layers (e.g. metal1 and metal3) may not align and, therefore, a stacked via cannot be connected nicely without a jog. Although a jog will work, it tends to increase the capacitance within a design. Therefore, it is highly advisable, as with this design, to keep each metal direction (i.e. metals in the same direction – metal1, metal3, metal5, ...) within a multiple of each other to allow each metal layer to overlap for stacked vias if needed. For example, if M3:M1 layer have a track pitch ratio of 11:8, both horizontal tracks would seldom be on top of each other making it difficult or even impossible to create a M3 → M1 contact. Therefore, for any standard-cell library track pitch ratios should use simple ratios such as 1:1, 1:2, 2:3, or 3:4 between adjacent same direction routing tracks. Figure 3 shows an example of a 1:2 M3 → M1 ratio.

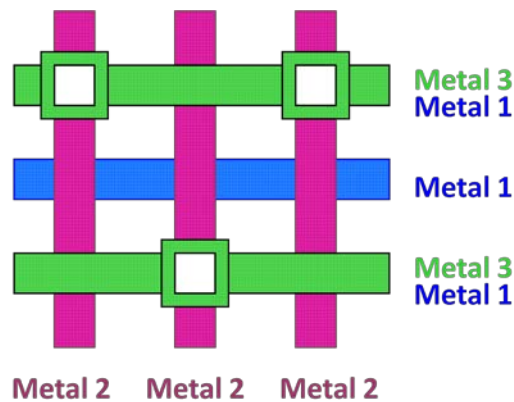


Figure 3. Example Pitch Definitions

In this design, the tracks were designed to have 300 nm pitch distances both vertically and horizontally. To allow an ample amount of room between both power rails, 9 grid distances were chosen between V_{DD} and GND rails allowing a 2.70 μm cell height. This height allows simpler gates to be designed while not wasting a lot of space. However, the space is just enough to allow more complicated gates like a Full Adder cell, to easily be integrated within the rails.

The IBM CMOS 10SF 65 nm CMOS technology has the ability to adapt to many different parameters. One particular parameter is called the metal stack and allows a customer to choose the number of metal layers for 1X, 2X, and 4X metal layers within the context of a design. Since it is easy to add more layers later, the stack can easily be extended to more metal layers.

The pitch definition is chosen such that it always has enough give-and-take to allow extra routing between the layers, but still overlap to allow stacked vias. The cell's width resolution (multiples of 0.x μm) is determined by the contact CA (CA width + chosen CA spacing). The CA width is 0.09 μm and the CA spacing is 0.21 μm giving a resolution of 0.3 μm . The other constraint is that the cells' width needs to be multiples of 0.2 μm to get rid of the gaps between cells. Due to rule 203aR in the IBM design manual, the minimum CA spacing is 0.16 μm (center to center = 0.25 μm) [12]. This plus the CA width = 0.16 μm + 0.09 μm = 0.25 μm and, therefore, in this work 0.3 μm is chosen. Nevertheless rule 203 says spacing can be as small as is 0.105 μm , so it might be possible to select 0.2 μm resolution (CA width + chosen CA spacing = 0.195) however, 0.3 μm was chosen just to make sure there is enough room for a route.

The metal pitch rules come in two forms: one is designed minimum and the other is considered a conservative rule denoted by appending a R after the rule (e.g. Rule 203R). IBM recommends that all conservative rules be utilized, therefore, this work chose to use the conservative R rules [12]. Consequently, the metal pitch was defined by a via-to-line distance assuming $\frac{1}{2}$ the via distance on one side and $\frac{1}{2}$ the metal line. For example, for metal1, the pitch is defined as:

$$\frac{1}{2} \cdot \text{Via} + \text{M1 via overlap} + \text{M1 spacing} + \frac{1}{2} \cdot \text{M1 wire} \quad (5)$$

which gives

$$0.05 + 0.04 + 0.1 + 0.5 = 0.24 \mu\text{m} \quad (6)$$

As stated before, to keep the M1 $\rightarrow$ M3 ratios simple along with the other layers, a 2:3 pitch was chosen which augments this distance from 0.24 μm to 0.30 μm , just to be conservative. The same procedure was completed for the other layers. The final values for pitch are shown in Table 1. It is important to understand that the values in Table 1 can be easily augmented to add more metal layers within the stack.

Once a pitch was defined, a standard-cell library and its design flow were created. Unfortunately, many tools still rely on different file formats to set up the geometry to allow all placement and routing to be done efficiently. In order to allow the standard-cell library to be easily defined and implemented, an automatic standard-cell library generation tool called Synopsys Cadabra was utilized. Cadabra takes an architecture description of a cell, defined as a spice netlist, and generates layout based on a defined rule set. Although the automatic library generation is efficient, the rules still need to be added manually and tested for several different scenarios to see if they work correctly. However, if a designer adequately defines the pitch, as explained previously, the job of defining the rules within the automatic layout generation is easier to implement.

Table 1. IBM CMOS10SF 65 nm Pitch Definitions

Metal Layer	Direction	Pitch [μm]	Ratio
M1	H	0.30	-
M3	H	0.20	2:3
M5	H	0.20	2:3
M2	V	0.20	-
M4	V	0.20	1:1
M6	V	0.20	1:1

The design flow shown in Figure 4 was then integrated with parameters cells provided by IBM. Cadence Design System's Virtuoso was used for layout creation and schematic entry. However, the design flow was designed to work with Synopsys and Cadence Design System synthesis engines. The standard-cell library was also created with baseline cells within its library, so that Synopsys DesignWare Intellectual Property could also be utilized and integrated within a design [25].

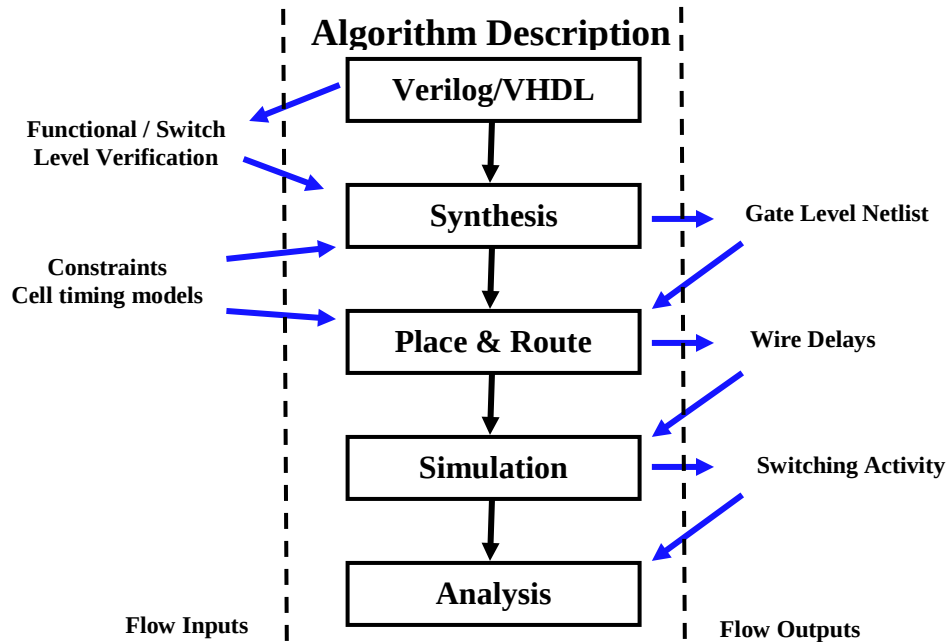


Figure 4. Proposed Design Flow using Commercial-based EDA tools

Cadence Design Systems' SOC Encounter was utilized for place & route, and Mentor Graphics' Calibre was used for Design Rule Checking, Electrical Rule Checking, Layout

versus Schematic, and Parasitic Extraction (PEX). The PEX was extremely important for power dissipation estimation, since knowledge of the capacitance in the wire was required to adequately address the total static and dynamic power dissipation within a design.

The design flow was planned to easily integrate within the Process Design Kit (PDK) that contains all the Parameterized Cells or p-cells within the CMOS 10SF library. More importantly, Regular Voltage Transistors, Low Voltage Transistors, and High Voltage Transistors were all included in the library amounting to a 93 cell library that was completely characterized for power and delay. This allowed simulations to accurately predict power dissipation and its associated propagation delay for various computer architectures.

To add to the robustness of the standard-cell library, six custom parts were also included that allowed memory storage to be integrated within the design structure. Each memory device was designed to be a robust register file with fast decoding and fast access. The six register files were all sized to provide a cycle time of 2 ns and also have several read and write ports to allow simultaneous reading and writing in common within microprocessors. Figure 5 shows the layout of a single read and write port 32-bit, 32-word register file designed for this library. Each register file was designed to have no Design Rule Check (DRC) violations and completely verify through Layout Versus Schematic (LVS) checks.

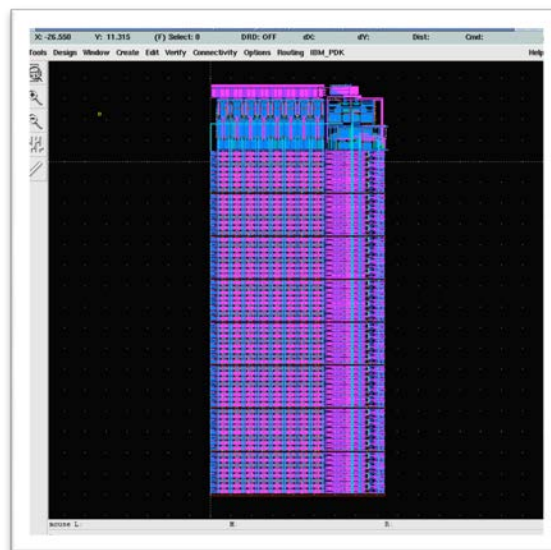


Figure 5. 32-bit x 32 word Single Write/Read Port Layout

3.2 Architecture Design Decisions for Lower Power Dissipation

Although power dissipation is an important factor within the design of computer architectures, choosing technology that impacts switching activity or utilizing low-leakage transistors is not the only option for the pragmatic designer. If designers are going to be successful at obtaining their goals, it is important to utilize complete architectures for lower power dissipation. It was also found to be important to have robust and repeatable scripts that allowed designs to be analyzed and properly.

In 2008, although the standard-cell library was created and appropriately characterized there were questions related to the leakage that this technology could produce. To offset concerns and potentially limit losses, changes in the technology were enacted to move towards a low-power technology, IBM CMOS 10LPe. Although the 10SF and 10LPe technologies are similar, they had significantly different design rules, thus, potentially, causing problems with the standard-cell designs. To offset these problems, standard-cell and memory libraries from Virage Logic were acquired. Regardless of the standard-cell library and design rule changes, the problem related to design flow integration remained to be explored and solved.

The instruction command sequence in EDA tools can be complicated. For example, Cadence Design Systems had over 3500 pages of text references for many commands that were actually either outdated, retired, or just repetitive. This can complicate the structure in which EDA tools are employed. For example, some commands can be non-existent or even be limited in their execution, because they are obsolete.

To simplify the process, commands were emulated based on Cadence Design System examples called the Foundation Flow Template System. The Foundation Flow is a derived set of current commands that allow basic structures to be repeated without the problems caused by outdated or obsolete command structures. Although the Foundation Flow scripts are well written, they are not incorporated with commands to target a specific design task or to obtain a power figure. Therefore, the Foundation Flows were heavily modified in Tcl to be robust, easy to edit, and repeatable with the IBM CMOS10LPe design library.

The main structure of this design flow allows decisions for low-power dissipation to be a healthy design choice and facilitates accurate simulation. The key element of this new flow is the use of the public-domain tool Makefile to aid in a repeatable sequence of steps within the backend design process. The basic flow structure is broken down into nine Tcl programs that run in succession. The nine Tcl programs are shown in Table 2.

It is important to emphasize that the new scripts are easily configurable and can be augmented for specialized instructions, such as bump-cell insertion in flip-chip designs. To add extra scripts, modifications are added to the setup.tcl file to post-process after each subsequent Tcl script. For example, if a user wanted to run a script after place.tcl, there is a location in setup.tcl to run a post-place.tcl, provided it is available. The setup.tcl contains default parameters, such as pitch and location of Layout Extraction Format files. A secondary Tcl script called init.tcl was written which contains routines to help with power rail adjustments and additions.

Table 2. Basic Foundation Flow Script Flow (in run order)

Program	Task
Init.tcl	Main initialization and Floor planning
Place.tcl	Placement and Partitioning of design
Pre-cts.tcl	Pre Clock-To-Synthesis Optimization
Cts.tcl	Clock-To-Synthesis (CTS)
Post-cts.tcl	Post CTS Fixing
Route.tcl	Global/Detail Routing
Postroute.tcl	Post route Optimization and Hold fixing
Postroute_si.tcl	SI Hold Fixing and Optimization
Signoff.tcl	Final signoff timing and verification

The scripts were tested on several design structures and sample netlists of various degrees of complexity. The key element to any of the scripts is that they were designed to be simple and easy to use for any design. That is, any design could be deployed and reloaded from any part of the design flow. For example, if a design had several errors within the route stage, the design could be loaded at the stage before the route stage (i.e., postcts.tcl), and errors could be corrected and integrated within the flow. In fact, this scenario of modifications had been previously been proposed for design flows to get the best optimizations within a design for a particular constraint [26]. Figure 6 demonstrates a sample design that has been placed and routed through the scripts as well as verified through DRC and LVS.

A script was also written to allow subsequent tools to adequately address power and energy consumption. The key to creating these results originally comes from the simplicity and repeatability of the original scripts created within this grant. Another key finding is that the scripts not only allow power results to be easily computed, they also allow many designs to be integrated into a stable set of design scripts and numbers to be targeted for a design. Currently, the scripts were designed for both the Synopsys and Cadence Design System (CDS) toolsets allowing mixing of program flows and vendor-driven EDA interactions.

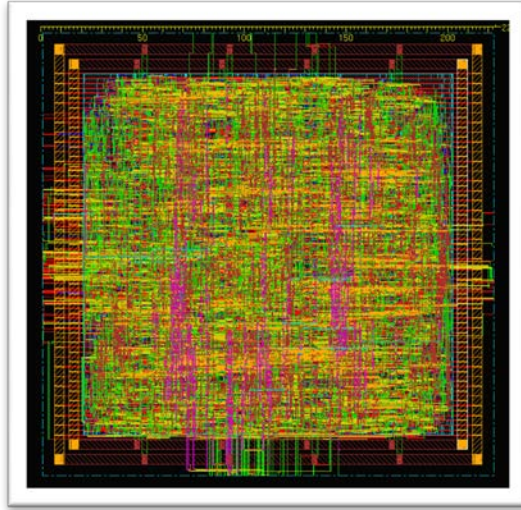


Figure 6. Place/Routed 500 MHz MIPS Processor Using Proposed Design Flow

4. RESULTS AND DISCUSSION

The design flow was created in such a way that environmental variables were utilized to enable it to be installed anywhere. For example, all the libraries refer to an environmental variable called \$OSU_IBM65 which points to the release directory. The library created for this research, as well as the library created by ARM for the IBM CMOS 10SF 65 nm technology was located in this directory. As stated previously, the ARM library does not invoke any low threshold voltage transistors and, thus, could potentially consume significantly more power.

Scripts were designed to work with the design flow, so that any design could be easily created from a HDL design into a mask layout. All standard-cell designs and memory elements were created using scripts that either generated a place and route design or initiated a language within the Cadence Design System tools to create a memory array.

As previously stated, power dissipation is an important element within any computer architecture, however, the computation of power can be difficult to compute because of the complexity produced by current designs. Consequently, the design flow presented here has several different power level estimation tools that can either estimate the power during synthesis with no parasitic extraction of the wires (e.g. through Synopsys PowerCompiler or PrimeTime) or computed more accurately using an extracted SPEF file (e.g. through Synopsys nanosim). Scripts were created that allowed all designs to be tested effortlessly for timing, area, and power parameters. More importantly, all the scripts can be modified, so that power, delay, or area could be targeted as an optimized constraint for a given architecture. Table 3 shows some typical numbers achieved for the IBM CMOS10LPe 65 nm developed design flow.

Table 3. Area/Delay Results for Sample Designs

Design	Synthesis		Post-Synthesis	
	Gates	Delay [ns]	Area [μm^2]	Delay [ns]
IEEE 754R Floating-Point Adder	4,493	2.00	24,115	3.070
MIPS R2000	4,984	1.05	22,563	2.122
64-bit CLA	568	1.99	2,613	2.287

All designs were created and completed through the use of scripts created for this project to interface the HDL definitions. Also, many of the designs in Table 2 were created using Synopsys DesignWare Intellectual Property engines giving the flow extended processing capabilities.

Although the design flow created for this work was easy to apply for any computer architecture, the real power is the ability to apply the design parameters for achieving lower power dissipation for designs that have multiple viewpoints, such as memory and processors. The scripts were successful in rapidly creating a design for a July 2010 fabrication run using the IBM CMOS10LPe library and several memory blocks from Virage Logic. The design was taped out at 125 MHz and preliminary tests indicated the design powered up correctly. In addition to the processor design created at the Air Force Research Laboratory, the device also included an asynchronous Field-Programmable Gate Array created by Cornell University.

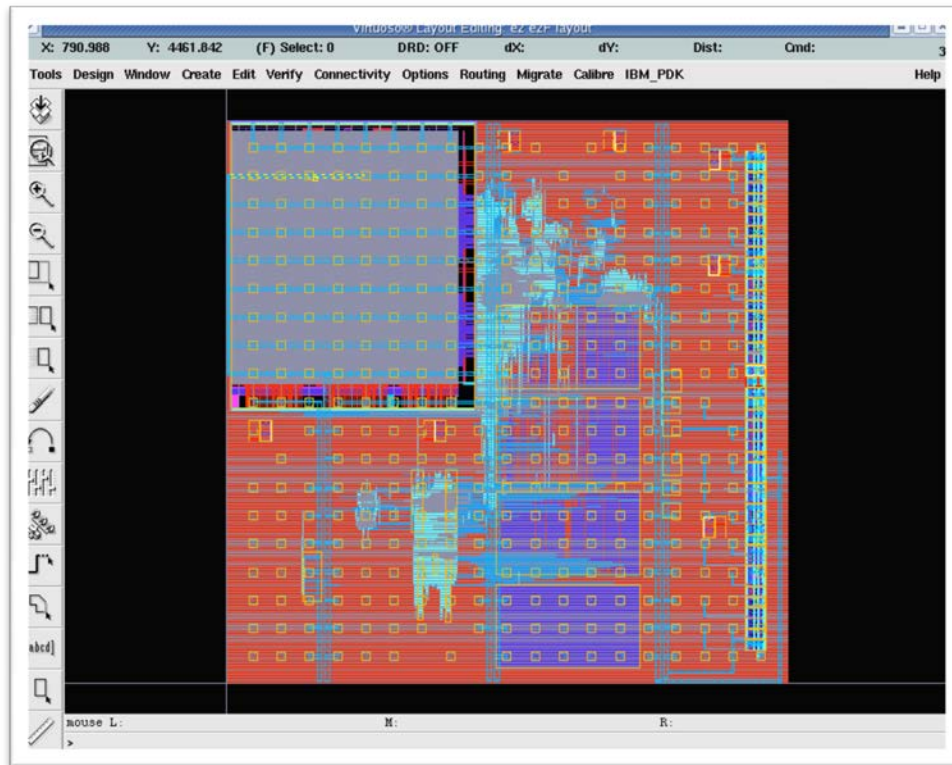


Figure 7. Final Layout for July 2010 T using IBM CMOS 10LPe Technology

5. CONCLUSIONS

This report describes research that was conducted for the Air Force Research Laboratory in designing low power libraries for computer architectures. With the combination of architecture and circuit-based enhancements, significant improvements over previous techniques can be seen.

The design flow was designed to work out-of-the-box and allow anyone to design fast and efficient designs with the IBM CMOS 10SF or CMOS10 LPe 65 nm libraries. More importantly, it allows multiple Electronic Design Automation tools to be efficiently and effortlessly utilized to create designs and verify their timing and/or power constraints. There is still more significant work that can be done by modeling better power variations as well as incorporating design choice what-if scenarios. Future work will be ongoing in this area and produce an environment that can be integrated easily with any trusted foundry customer.

Over the course of three years, specific achievements resulted in the following items:

- Design flows created that were enhanced for Cadence Design System and Synopsys based EDA toolsets.
- A vibrant and modular standard-cell library and six register files for the IBM CMOS10SF technology.
- Design Verification scripts for timing, voltage, and power considerations.
- Modular Foundation-based script vehicles that allow rapid development of Silicon structures.
- Tape-Out procedures and methodologies for the IBM CMOS10LPe technology.

6. REFERENCES

- [1] Intel Corporation, "DOE accelerated strategic computing initiative TFLOPS system," 1996.
- [2] *International Technology Roadmap for Semiconductors*, Semiconductor Industry Association, 2003.
- [3] D. Burger and T. M. Austin, "The Simple Scalar Tool Set, version 2.0," University of Wisconsin at Madison Technical Report, TR 1342, 1997.
- [4] D. Brooks, V. Tiwari, and M. Martonosi, "Watch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proceedings of the International Symposium on Computer Architecture*, 2000, pp. 83-94.
- [5] K. Skadron, M. Martonosi, D. I. August, M.D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in Computer Architecture Evaluation," *IEEE Computer*, 2003, pp. 30-36.
- [6] N. H. E. Weste and D. Harris, **CMOS VLSI Design: A Circuits and Systems Perspective**, 3rd Edition, Addison Wesley, New York, 2004, pp. 117
- [7] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *Proceedings of the ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, 2006, pp. 21-30.
- [8] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, **Low Power Methodology Manual: For System on Chip Design**, Springer, New York, 2007, pp. 314
- [9] J. E. Stine and J. Grad, "Low-Power and High-Speed Addition Strategies for VLSI," *International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2006, pp. 1610-1613.
- [10] K. Roy and S. C. Prasad, "Low-Power CMOS VLSI Circuit Design," Wiley-Interscience, 2000, pp. 118
- [11] R. Chau, J. Brask, S. Datta, G. Dewey, M. Doczy, B. Doyle, J. Kavalieros, B. Jin, M. Metz, A. Majumdar, and M. Radosavljevic, "Application of High-K Gate Dielectrics and Metal Gate Electrodes to Enable Silicon and Non-Silicon Logic Nanotechnology," *Microelectronic Engineering*, **Vol. 80**, 2005, pp. 1-6.
- [12] *CMOS 10SF (CMS10SF) Technology Design Manual: ES 70P3848*. IBM, 2007.
- [13] I. Sutherland, R. Sproull, and D. Harris, **Logical Effort: Designing Fast CMOS Circuits**, Morgan Kaufman Publishers, New York, 1999, pp. 328
- [14] P. R. Groenveld, "Physical Design Challenges for Billion Transistor Chips," *Proceedings of the International Conference on Computer Design*, 2002, pp. 78-83.

- [15] M. Pedram and J. M. Rabaey, **Power Aware Design Methodologies**, Kluwer Academic Publishers, Norwell, MA, 2002.
- [16] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor, "Magic: A VLSI Layout System," University of California Berkeley Technical Report, 1983.
- [17] N. A. Sherwani, **Algorithms for VLSI Physical Design Automation**, Kluwer Academic Publishers, Norwell, MA, 1998.
- [18] J. B. Sulistyo and D. S. Ha, "Developing Standard-Cells for TSMC 0.25um Technology Under MOSIS DEEP Rules," Technical Report Virginia Polytechnical Institute and State University, VISC-2002-02, 2002.
- [19] J. Grad, J. E. Stine, and D. D. Neiman, "Real-World SOC Experience for the Classroom," *International Conference on Microelectronic System Education*, 2005, pp. 49-50.
- [20] J. E. Stine, J. Grad, I. D. Castellanos, J. M. Blank, V. B. Dave, M. Prakash, N. Iliev, and N. Jachimiec, "A Framework for High-Level Synthesis of System on Chip Designs," *International Conference on Microelectronic System Education*, 2005, pp. 67-68.
- [21] J. Grad and J. E. Stine, "A Standard-Cell Library for Student Projects," *International Conference on Microelectronics System Education*, 2003, pp. 98-99.
- [22] J. E. Stine, I. D. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, and R. Jenkal, "FreePDK: An Open-Source Variation-Aware Design Kit," *International Conference on Microelectronic Systems Education*, 2007, pp. 173-174.
- [23] IBM, "CMOS 10SF Technology Design Manual," Technical Report, 2009.
- [24] S. H. Gerez, **Algorithms for VLSI Design Automation**, Wiley and Sons, New York, 1998.
- [25] M. Sarrafzadeh and C. K. Wong, **An Introduction to VLSI Physical Design**, McGraw-Hill, New York, 1996.
- [26] A. Kuhlmann and L. P. P. P. Van Ginneken, "Grammar-based optimization of synthesis scenarios," in *IEEE International Conference on Computer Design*, pp. 20-25, 1994.

LIST OF ACRONYMS

ACT	Asynchronous Circuit Tools
AFPGA	Asynchronous Field Programmable Gate Array
AFRL	Air Force Research Laboratory
ASIC	Application Specific Integrated Circuit
CB	Connection box
CMOS	Complementary Metal Oxide Semiconductor
DARPA	Defense Advanced Research Projects Agency
DRC	Design Rule Checker
EDA	Electronic Design Automation
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
ITRSI/O	International Technology Roadmap for Semiconductors Input/Output
I/OIP	Input/Output Intellectual Property
IPLB	Intellectual Property Logic block
LBLUT	Logic block Lookup Table
LUT	Lookup Table
LVS	Layout versus Schematic
MOSISSBSRAM	Metal Oxide Semiconductor Implementation Service Switch box Static Random Access Memory
SBSRAMVHDL	Switch box Static Random Access MemoryVery High Speed Integrated Circuit Hardware Description Language
SRAMVHDLVLSI	Static Random Access MemoryVery High Speed Integrated Circuit Hardware Description Language Very Large Scale Integration
TSMCVHDLVLSIVPR	Taiwan Semiconductor CorporationVery High Speed Integrated Circuit Hardware Description Language Very Large Scale Integration Versatile Place and Route
VHDLVLSIVPR	Very High Speed Integrated Circuit Hardware Description Language Very Large Scale Integration Versatile Place and Route
VLSIVPR	Very Large Scale Integration Versatile Place and Route
VPR	Versatile Place and Route