A Structured Design Methodology for High Performance VLSI Arrays

by

Satendra Maurya

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2012 by the
Graduate Supervisory Committee:

Lawrence Clark, Chair
Keith Holbert
Sarma Vrudhula
David Allee

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

The geometric growth in the integrated circuit technology due to transistor scaling also with system-on-chip design strategy, the complexity of the integrated circuit has increased manifold. Short time to market with high reliability and performance is one of the most competitive challenges. Both custom and ASIC design methodologies have evolved over the time to cope with this but the high manual labor in custom and statistic design in ASIC are still causes of concern.

This work proposes a new circuit design strategy that focuses mostly on arrayed structures like TLB, RF, Cache, IPCAM etc. that reduces the manual effort to a great extent and also makes the design regular, repetitive still achieving high performance. The method proposes making the complete design custom schematic but using the standard cells. This requires adding some custom cells to the already exhaustive library to optimize the design for performance. Once schematic is finalized, the designer places these standard cells in a spreadsheet, placing closely the cells in the critical paths. A Perl script then generates Cadence Encounter compatible placement file. The design is then routed in Encounter. Since designer is the best judge of the circuit architecture, placement by the designer will allow achieve most optimal design.

Several designs like IPCAM, issue logic, TLB, RF and Cache designs were carried out and the performance were compared against the fully custom and ASIC flow. The TLB, RF and Cache were the part of the HEMES microprocessor.

DEDICATION

Dedicated to my parents *Shri Mahendra Kumar Maurya* and *Smt. Vinod Kumari*

ACKNOWLEDGMENTS

First of all, I would like to thank my parents and family for the support they have given throughout my studies. They have stood by me through the ups and downs of my graduate life and I am really grateful for that.

I am deeply indebted to my advisor Dr. Lawrence Clark, for the guidance and support that he has given me throughout my masters. Rarely have I walked into Dr. Clark’s office with a problem and walked out without a solution. He has been a tremendous source of counseling and inspiration for me. I truly believe that I have become a better engineer under his guidance and I thank him for that. I also thank Dr. David Allee, Dr. Keith Holbert and Dr. Sarma Vridulla for being on my committee.

I thank Dan Patterson, Nathan Hindman, Thomas Mozdzen, Jerin Xavier, Srivatsan Chelleppa, Sandeep Shambhulingaiah for their support in the research and for their technical and non-technical ideas without which the successful completion of the work would not have been possible. Also my friends Nishant Chandra, Ashutosh Singuraur and Vinay Chinti for their moral and immoral support both academia and non-academia.

Finally, I would like to thank the Almighty for His grace. It would not have been possible without His blessings.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER 1. **INTRODUCTION**

Integrated circuits (IC) have demonstrated a compound annual growth of 53% over 50 years as measured by transistor count [Weste10]. Digital circuits progress with technology scaling roughly following the Moore's Law [Moore65] which has become a self-fulfilling prophecy for this enormous growth. It is driven primarily by scaling down the transistor size and to minor extent by building larger chips. In general, each generation shrinks the linear dimension by 0.7. The development of new CMOS technology nodes has been primarily motivated by the rapidly growing demand for high performance in digital circuits. Table 1.1 and Figure 1.1 show how the transistor count and integration density has dramatically increased in the microprocessors over the time [Moore03] [Ogdin75]. Smaller transistor feature sizes make it possible for digital circuits to run faster and consume less power. Moreover, it makes them cheaper to manufacture with each generation. Both speed and power efficiency has improved geometrically (by the approximated 0.7× per generation) in the past two decades, resulting in a huge overall performance improvement.

The increasing density of IC's has led to the emergence of system-on-chip (SOC) design with a concomitant increase in design size and complexity. However, this has resulted in significant increases in the cost of IC design and corresponding increases in mask costs on the order of millions of dollars. High performance IC designs have stringent area, speed, and power requirements. In the past this has been dealt with by having large design teams, but must increasingly be automated as design team cannot be made larger.

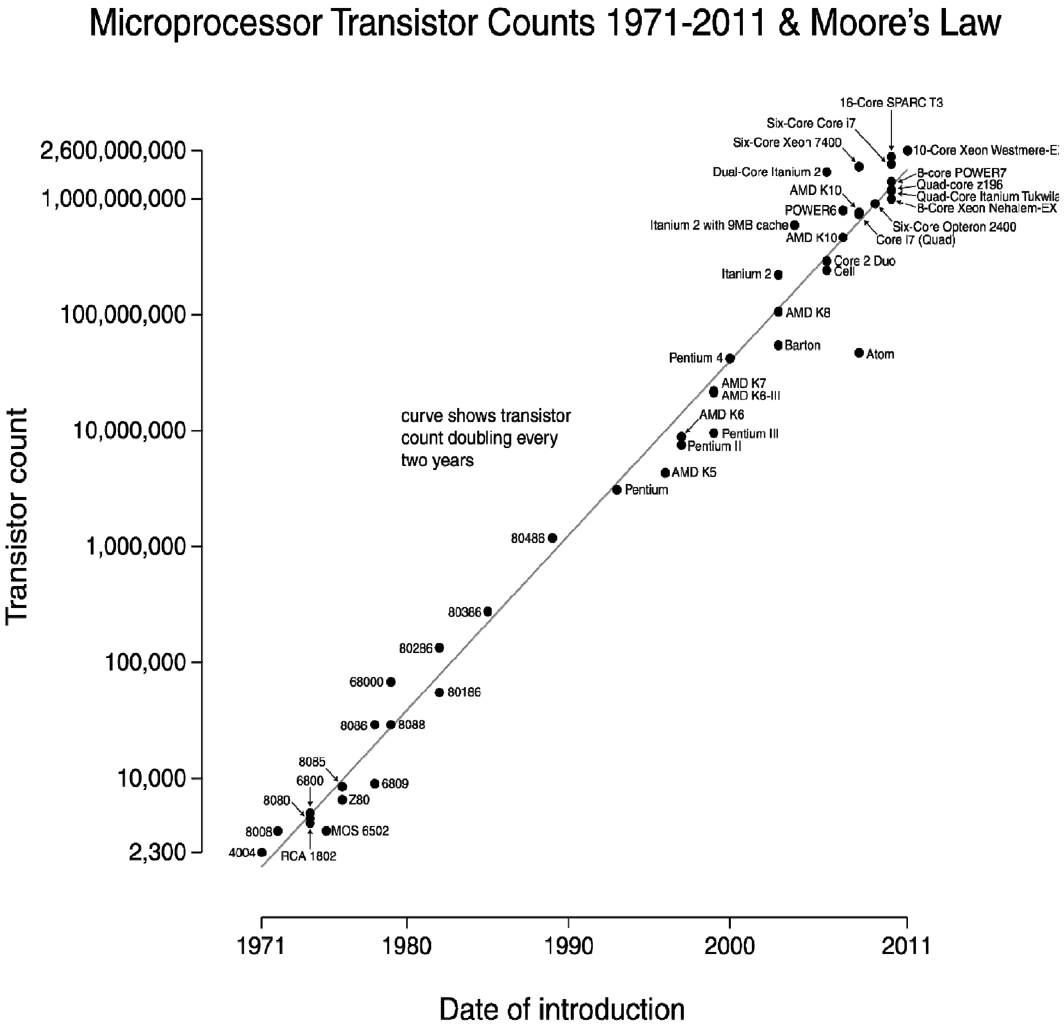


Figure 1.1 Figure showing the CPU transistor count against the Moore's law

1.1 ***Design Challenges***

With the improving design performance and reduced cost of the IC, increasing system complexity poses a major challenge. Modern SOC designs combine memories, processors, high speed I/O interfaces, and dedicated application-specific logic on a single chip. Partitioning the design to simplify the implementation process is necessary. However, the interdependence between the structures complicates partitioning. The practice of structured design relies on hierarchy, regularity, modularity and locality to manage complexity.

Table 1.1 Microprocessor Scorecard [Ogdin75] [Microprocessor12]

Processor	Transistor count	Area	Year
Intel 4004	2300	12 mm²	1971
Intel 8008	3500	14 mm²	1972
Zilog Z80	8500	18 mm²	1976
Intel 8086	29000	33 mm²	1978
Pentium	3,100,000		1993
AMD K5	4,300,000		1996
AMD K7	22,000,000		1999
Pentium 4	42,000,000		2000
POWER6	789,000,000	341 mm²	2007
Atom	47,000,000		2008
AMD K10	758,000,000		2008
Core i7 (Quad)	731,000,000	263 mm²	2008
Six-Core Opteron 2400	904,000,000	346 mm²	2009
10-Core Xeon Westmere-EX	2,600,000,000	512 mm²	2011

Digital VLSI design is often partitioned into five levels of abstraction: architecture, micro-architecture, logic, circuits, and physical design. Architecture describes the user visible function of the design. Partitioning the design into registers and other functional blocks is determined by the micro-architecture level. Logic describes how the functional units are constructed. Use of transistors to implement the logic comprises circuit design. Finally, layout and placement of these transistors are part of physical design level. These elements are relatively independent and all influence each of the design objectives. Figure 1.2 shows the various levels of abstraction in the modern VLSI design.

The viability of VLSI design depends on a number of conflicting factors, e.g., performance in terms of speed or power consumption, cost, and production volume. Performance excellence at low cost can only be achieved using volume production. With ultimate performance as the primary design goal, high performance custom design techniques are often desirable. However, the cost of a sufficiently large team for this approach is becoming untenable. Reducing the system size through integration, not performance, is the major objective in most consumer applications. Under these circumstances, the design cost can be reduced substantially by using advanced design-automation techniques. VLSI architectures should exploit the potential of the VLSI technology and also take into account the design constraints introduced by the technology. Some of the key design requirements are summarized below:

Simplicity and regularity: Cost effectiveness has always been a major concern in designing VLSI architectures. A structure, if decomposed into a few types of building blocks which are used repetitively with simple interfaces, results in great savings. In VLSI, there is an emphasis on keeping the overall architecture as regular and modular as possible, thus reducing the overall complexity. For example, memory and processing power will be relatively cheap as a result of high regularity and modularity.

Design reuse: It improves the design productivity and also saves time and effort. Effective reuse of existing designs requires proper representation, abstraction, and characterization in terms of its functionality, performance, reliability, and possible interactions with the environment, so that it can be seamlessly integrated with the rest of design for synthesis, simulation and verification. Such representation and abstraction should also support efficient update of the design when migrating from one technology generation to another, from one foundry to another, and from one design environment to another. Design reuse should also include the development of reusable design process,

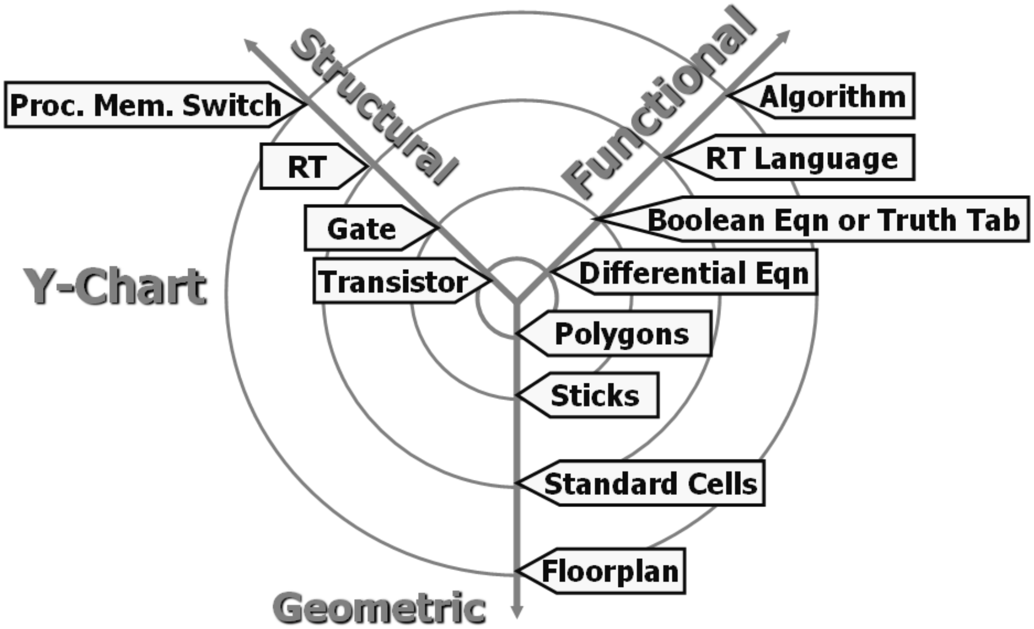


Figure 1.2 Traditional Hardware Levels of Abstraction

methodology, and tools so that they can be retargeted for different technology generations and easily shared among different design projects.

Scalability and optimization: Complexity management requires improvement for global performance, power, area, and reliability optimization. Innovations in highly scalable optimization algorithms which can handle complex design constraints, multiple design objectives, and rapidly increasing design sizes will significantly improve the capability, efficiency, and quality of the design tools for future ICs.

1.2 ***Logic design methodologies***

Over the last three decades, several logic design methodologies have evolved to cope with technological advancements in semiconductor circuit design.

Based on the design approach they can be broadly classified into full-custom and ASIC (application-specific integrated circuit) methodology.

1.2.1 ***Full-custom design***

Full-custom design relies extensively on manual effort for most design decisions. For example, transistor sizing, transistor layout, device placement and routing are all carried out manually with the aid of computer-aided design (CAD) tools. The circuit is partitioned into a collection of sub-circuits based on functionality creating several levels of hierarchy. Each functional block can be of any size.

This technique offers the greatest flexibility from a designer perspective, because circuits can be tailored to specifications with superior performance in terms of area, delay or power [Hurst99]. When designing a custom IC, the designer has a full range of choices in design style and benefits from the ability to optimize across domains. These include micro-architecture, logic design, floor-planning and physical placement, and most importantly choice of logic family. Circuits can be carefully optimized and use special circuit styles and arbitrary sizing of the transistors for high speed, lower power, and lower area. Custom designs may also show superior logic-level design of regular structures such as adders, multipliers, and other data-path elements. They achieve fewer levels of logic on the critical path with more compact, complex logic cells and by combining logic with the latches [Hwang93]. Due to lack of any constraints on the physical design perspective, the custom design achieves very compact layout design.

However, there is a high engineering cost overhead involved. The amount of effort required for full-custom design scales linearly with the number of unique circuits in the design [Chen06]. Furthermore, given shrinking time-to-market windows and market lifetime of IC products, it becomes increasingly difficult to depend on full-custom techniques for IC design. For example, a large IC design house like Intel requires large teams of designers working for the equivalent over a thousands of man-years to deliver high-performance full-custom products such as the Pentium 4 chip on schedule. This results in huge expenses that has an impact on the pricing of such chips and the products that use them (e.g., personal computers) [Hurst99]. This limits the custom design to be used when the final design must have high performance requirement and the design time is less of a factor.

1.2.2 ***Application specific integrated circuit (ASIC)***

In order to reduce engineering effort, some vendors resort to automated design techniques to deliver new products much faster and at lower cost [Costa84]. They rely exclusively on electronic design automation (EDA) tools to implement logic and circuits [Loos96]. These EDA tools use libraries of pre-designed logic cells, standard cells, blocks, and or cores, to implement circuits in much shorter time but with some overhead costs, and significant restrictions on the IC designer [Loos96].

This flow is known as the application specific integrated circuit (ASIC) flow as illustrated in Figure 1.3. This approach of circuit design has become immensely popular and is used for the implementation of virtually all logic elements in today’s integrated circuits. This has been made possible because of the advent of sophisticated logic-synthesis tools and increased quality of automatic cell placement and routing tools. It is a structured methodology that theoretically limits the space of exploration, yet still achieves superior results in the fixed time constraints of the design [Vincentelli04].

In standard-cell-based designs, vendors develop cell libraries that implement generic logic gates such as NOR and NAND gates with different drive strengths and in accordance with a constrained physical layout, typically with fixed cell height [Yoshida04]. These libraries of logic gates are then mapped from register transfer level (RTL) descriptions of desired hardware behavior written in VHDL or Verilog [Zhang01], into gates using logic synthesis tools such as Synopsys Design Compiler and Cadence RTL Compiler [Stine05]. A more aggressive form of cell-based design allows designers to build custom cells and tools according to their own specifications and tailored for a particular circuit or group of circuits [Dally00] [Chinnery02] [Bhattacharya04] [Scott94].

Synthesis mapping is automated and can be constrained for area and performance [Dunlop85]. Placement involves reading a netlist of gates generated through logic synthesis and then arranging the corresponding standard cells in rows. This high degree of automation is made possible by placing strong restrictions on the layout options. Standard cells are arranged in rows so that power (V_{DD}) and ground (V_{SS}) rails of adjacent cells are connected by abutment since all the standard cells are of same height [Xing02]. Routing tools connect nodes (with metal wires) to implement the desired logic function. A substantial

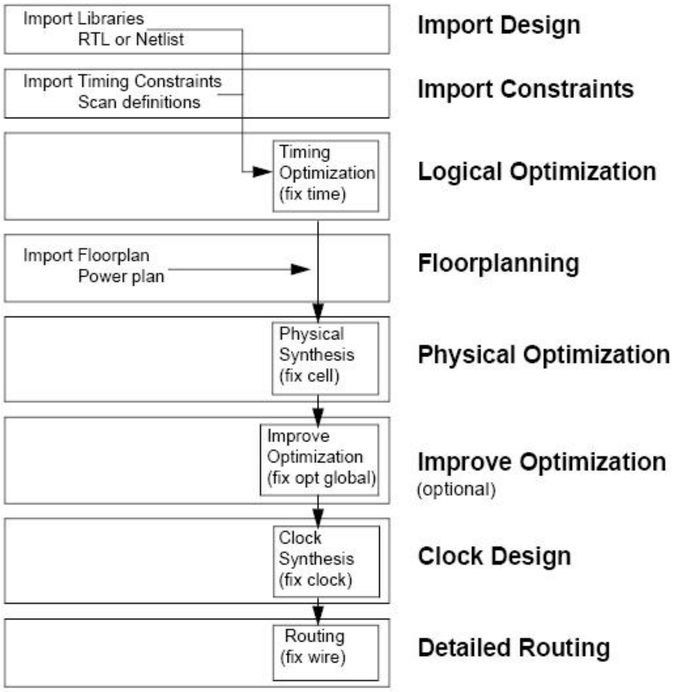


Figure 1.3 Figure showing the different stages of ASIC design flow

fraction of area still must be devoted to signal routing [Gyurcsik89] [Keyes82]. The minimization of interconnect overhead is the most important goal of the standard cell placement and routing tools. The routing area, for design rules with two metal layers, is about 50% of the chip area [Moraes98]. For design rules with three or more metal layers, it is possible to aggressively use the over-the-cell routing [Kim96] to reduce the routing area to 25% of the chip area. The approach "zero routing footprint" [Sherwani95] (no routing area) is possible because the cells are designed to be transparent to several routing tracks, increasing in this way the cell height. When more than three metal layers are available for routing, they must be used to connect different blocks, not inside them (transparency over blocks). There are cells without any transistors, called feed-through or filler cells. They can be added between cells to allow vertical connectivity when there are no more routing resources over the cell, thus reducing long wires [Clein99]. More interconnect layers also reduce wiring area.

The speed of average ASIC design lags that of the fastest custom circuits in the same processing geometry by factors of as much as six to eight [Chinnery02]. The standard cell layout is inherently non-hierarchical. ASIC designs are limited by the algorithms in the synthesis. The performance achieved solely depends on the efficiency these algorithms to select right logic implementation, placement and finally routing the standard cells. The algorithms read the design data for each of the standard cells that consist of dimensions, timing, drive strength and power dissipation and try to meet the specification. The optimization scale is limited by the granularity of the standard cell library. Today's libraries contain from several hundred to more than a thousand cells comprising of range that vary in terms of drive strengths (2×, 4× etc.), logic styles (static, dynamic, sequential etc.), and logic type (NAND, NOR etc.). These cells have to be redesigned with every migration to a new technology, and technology sub-nodes must also be supported. A number of iterations of the synthesis flow shown in Figure 1.3 are needed until the specifications are matched [McFarland88]. Depending on the design and the target specification, several iterations need to be done that can take longer [Tiri04].

Cell based "soft-IP" microprocessors are not typical, because they bear more similarity to custom microprocessors, but they present a good mid-point in the comparison between custom design and a typical ASIC design. This methodology is mostly used to implement random or control logic part of the

Table 1.2 Performance comparisons for custom and ASIC micro-architectures

Custom			
Name	Frequency(GHz)	Power(W)	Area(mm^2)
Pentium III(Tualatin)	1.4	31.0	80.0
Pentium 4 (Northwood)	2.2	55.0	146.0
ASIC			
Lexra LX4380	0.420	0.05	0.8
ARM1022E	0.325	0.23	7.9

micro-processor design. Table 1.2 exemplifies some of the microprocessor designs carried out at 0.13 µm technology node using both custom and full synthesis approach. At the outset, it can be observed that custom microprocessor operate 3× to 4× faster than those fully synthesized in the same process [Chinnery02]. This gap seems staggering– the speed improvement due to one process generation (e.g. 0.18µm to 0.13µm) is 1.5× [Moores] then this gap is equivalent to that of two process generations.

However, in the future, with increasing design rule complexity and restrictions, imposed regularity, there will be very little room for improvement with manual design. Furthermore, due to increased complexity of the design rules, manual designs may gravitate towards local optimizations rather than global, and thus may even be suboptimal [Borkar09]. Custom design will be limited to memory arrays, register files and large arrays. This is where the current area of this research concentrates and proposes methods to optimize the design time without deteriorating the performance.

*1.3 **Designing comparing ASIC and full-custom techniques***

In order to completely understand the two design approaches and their feasibility for different types of designs, several works has been published that compare ASIC designs against those using full-custom techniques.

*1.3.1 **Full adder case analysis***

A case study carried for adder design by Henrik *et al.* quantifies the performance gap between the two techniques for two types of adder structures [Eriksson03]. The adders used in the comparison are one 64-bit ordinary ripple-carry adder (RCA) and one 64-bit Han-Carlson tree adder (HCA) [Han87]. These adder topologies were chosen since they represent two adder implementation extremes. The interconnect wire run in RCA is very small as the carry propagates only to the neighboring adder cell. However the small interconnect run comes at the cost of poor performance. The performance was improved by using dynamic circuit. The fully custom RCA design leads to much higher performance over the unstructured standard cell implementation both in terms of delay and area. The HCA, on the other hand has, as all tree adders do, complex interbitslice interconnections to achieve high performance [Knowles01]. The performance of the design carried out using the custom placement or using ASIC flow was close. Table 1.3 summarizes the two adders performance built using the two methodologies.

Thus the ASIC tool performs place and route as efficiently as custom designer would do for implementaion with inter-bitslice interconnections, but that full-custom designs reach higher performance through the use of advanced circuit

Table 1.3 Performance comparisons for Adders

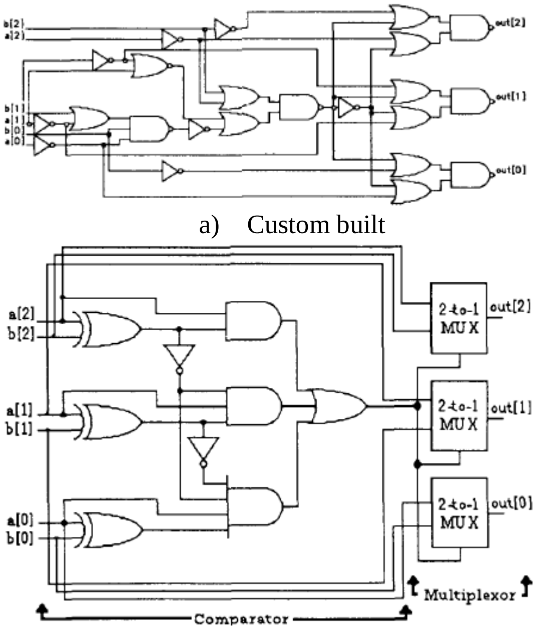
Adders	ASIC		Custom	
	Delay(ns)	Area(mm^2)	Delay(ns)	Area(mm^2)
RCA	30.24	0.045	23.54	0.015
HCA	3.92	0.110	2.13	0.416

technique. This is due to the fact that the length of the critical wires in full-custom bitsliced layouts is proportional to width of adder, but in standard-cell based layouts the length is proportional to the square root of the width. For small tree adders full-custom layouts are preferable, but for wide adders standard-cell based layouts are better since the routing-dependent delay scales better with the adder width [Eriksson03].

1.3.2 ***Viterbi Decoder***

A similar analysis was carried out by Grey *et al.* for a Viterbi decoder that assumed trellis encoded modulation with an 8 PSK signal set, rate 2/3 and memory length 2 encoder [Gray91]. At each transmission interval, the decoder processes a measure for each of the eight possible transmit signals and decides upon the most likely transmitted symbol. This decoder architecture is highly parallel to optimize for speed. One symbol is decoded at every clock cycle. The Viterbi decoding algorithm leads to an implementation comprised of flip-flop storage elements and combinational logic elements to create fast adders, comparators, multiplexors and control circuits which are the main constituents of digital circuit designs.

The synthesized decoder is 16% smaller and 13% faster than the hand optimized design possibly because synthesis could optimize larger circuits better than custom optimization. Approximately 120 man-hours were used to enter the



a) Custom built

b) Synthesized

Figure 1.4 Comparison of 3-bit comparator/multiplexer circuit

schematic and simulate the decoder for the "hand" design. This does not include the time spent simulating the algorithm, finding the minimal logic and minimal gate level implementation. For comparison, the total time required to learn the ASIC synthesis tool, Verilog HDL, write and debug the Verilog code, test the netlist and compile the decoder took the same amount of time, approximately 120 man-hours. The performance comparison for the design using the two methodologies is shown in Table 1.4.

The performance degradation for custom design in terms of both area and delay could be attributed to simple design approach of using two input NAND/NOR/INV gates for the implementation. However the synthesis tool used complex XOR gates and multi-input gates and multiplexers to optimize the design. This difference is clearly inferred from the two schematics of the 3-bit

Table 1.4 Performance comparisons for Viterbi decoder

	Delay(ns)	No. of Std cells	Transistor Count	Area(mils^2)
Custom	41.6	1359	16545	21877
ASIC	36.3	1493	11946	18322

comparator shown in Figure 1.4 for design using the custom and synthesized. The 3-bit comparator compares two 3-bit inputs and outputs the smallest.

1.3.3 ***Microprocessor Design***

For microprocessors the custom design methodologies have proven to have much better performance than synthesized designs. One of the most acclaimed high performance fully custom microprocessors was the Dec Alpha 21064 introduced in 1992 [Gronowski98]. It was the culmination of three generations of Alpha microprocessors that achieved high performance through process advancements, architectural improvements, and very aggressive circuit design techniques. Figure 1.5 shows the integer benchmark performance of as a function of time. The Alpha chips showed that manual circuit design applied to a simpler, cleaner architecture allowed for much higher operating frequencies than those that were possible with the more automated design methodologies. These chips caused a renaissance of custom circuit design within the microprocessor design community at that time (as it is now) the microprocessor industry was dominated by automated design and layout tools [Bolotoff07].

As mentioned, the DEC Alpha microprocessor designs used full-custom circuit design methodologies. All circuits were designed at the transistor level, and were uniquely sized to meet speed and area goals. Custom layout design techniques optimized parasitic capacitances for all circuits. Automatic synthesis approaches for logic and circuit design were used for fewer than 10% of the circuits. The use of a full-custom design methodology gave the designer flexibility. The Alpha microprocessors were implemented with a wide range of circuit styles including conventional complementary CMOS logic, single and dual rail dynamic logic, cascode logic, pass transistor logic, and ratioed static logic [Benschneider95]. Dynamic circuits were one of the most commonly used circuit styles, and were present in both data path and random control structures. Another circuit style that was widely used in the microprocessors was dual-rail cascode voltage switch logic (CVSL) [Dobberpuhl92]. Although full custom circuit techniques allowed designers to build very high speed circuits, the verification and validation of the design at process corners presented a challenge that lead them to develop many in-house CAD tools. Also migrating from one technology node to other took much more time than required for move automated methodologies.

Soft-cores are microprocessors whose architecture and behavior are fully described using a synthesizable subset of a hardware description language (HDL). The use of soft-cores provides many advantages for the embedded system designer. First, soft-core processors are flexible and can be customized for a specific application with relative ease. Second, since soft-core processors are technology independent and can be synthesized for any given target technology, they are more immune to becoming obsolete when compared with circuit or logic level descriptions of a processor. Finally, since a soft-core processor's architecture and behavior are described at a higher abstraction level using an HDL, it becomes much easier to port the design. As against the soft-core fully

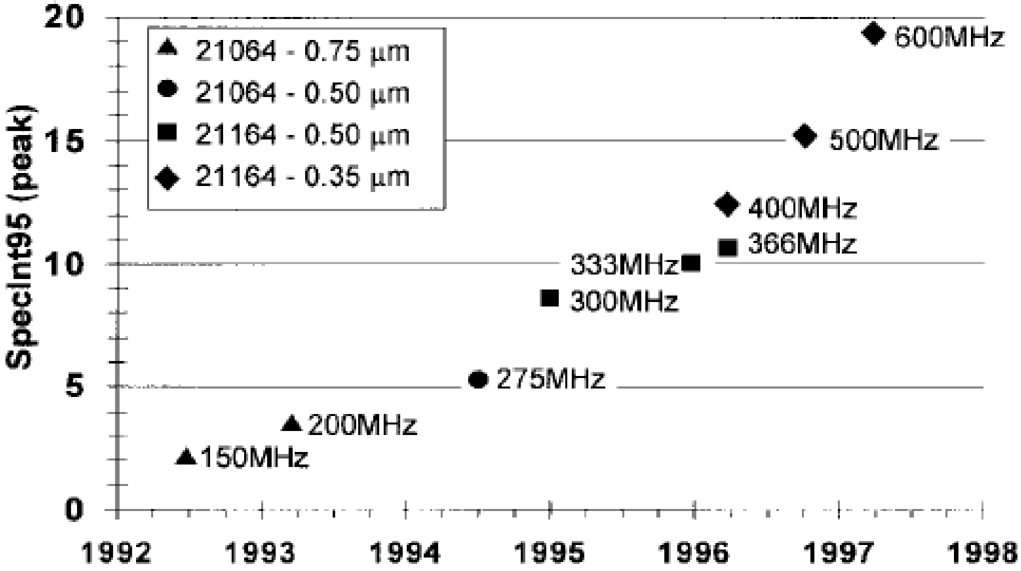


Figure 1.5 Alpha performance versus time

synthesized RISC microprocessors, which are standard in the current embedded market, we can see from the Figure 1.6, for the 0.35 µm technology node the performance significantly lags the Alpha processor at the same technology node [Toshiba].

1.3.3.1 ***iCORE to bridge ASIC and custom performance***

The iCORE project by STMicroelectronics tried to prove that a proPerly configured ASIC design flow could give a soft CPU core custom performance [EEtimes02]. The goal was to create a very high performance version of the company's ST20-C2 embedded CPU architecture, but it also show that the design could be portable as a soft-core across existing and future technologies. This ruled out the extensive use of custom circuits, and led to the adoption of a methodology close to a traditional ASIC design flow, but one tuned to the aggressive performance goals demanded by the project.

Firstly, the performance gap between custom and synthesized embedded cores was closed by using deep, well-balanced pipelines, coupled with careful partitioning, and mechanisms that compensate for increased pipeline latencies. This enables the use of simple and well-structured logic functions in each pipeline stage that are amenable to synthesis. Secondly, the performance gains are consolidated by use of placement-driven synthesis, thirdly, by using more careful clock-tree design. Delays were also reduced by creating larger, single level cells for high-drive gates. Physical compiler was used for cell placement. Timing simulation indicated that a good balance of delays between the pipeline stages was achieved. Silicon testing showed functional performance of the design from 475 MHz at V_{DD}= 1.7 V to 612 MHz at V_{DD}= 2.2 V and 25 °C in 0.18µm CMOS process [Richardson03]. This performance is close comparison to the DEC Alpha 21164PC implementation which was implemented at 0.28µm technology node rather than the iCore which was on 0.18µm [David09], lag of one process technology node.

1.4 ***Comparison of different design styles***

Thus we have found that the performance of a design depends on choice of methodology adopted which intern depend on the intended functionality of the chip, time-to-market and total number of chips to be manufactured. For designs that are more random or control logic type, ASIC methodology is well suited whereas for structures that are regular and hierarchical, custom design has proven results. But still the time and effort of the custom design and the inconsistent and ergodic implementation of the ASIC methodology are the major cause of concern.

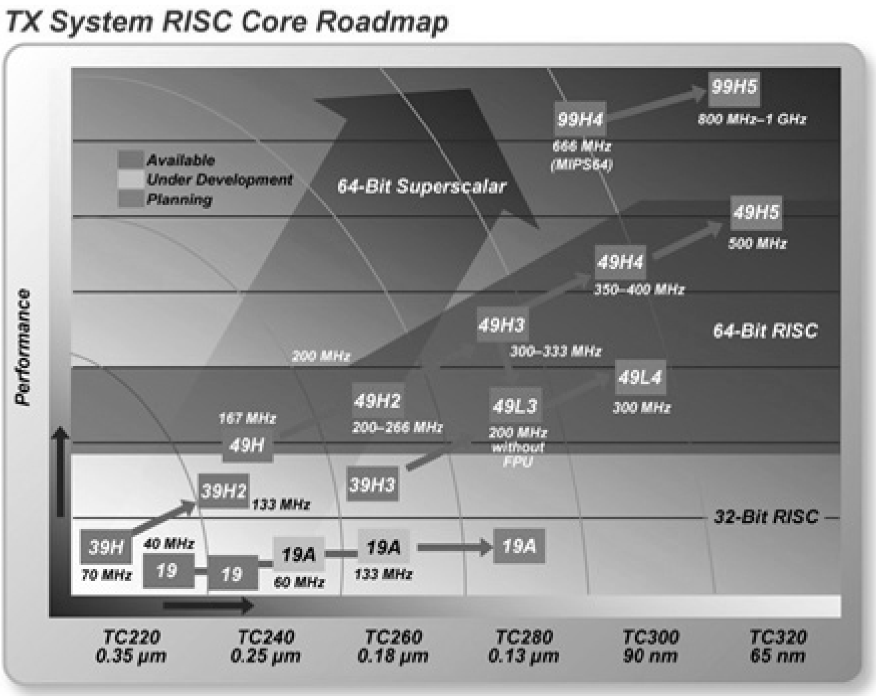


Figure 1.6 RISC softcore roadmap

A mid-way between the two where the implementation of the design is done similar to using the custom methodology in which the designer chooses the cells (both size and implementation kind) and also carry out the floorplanning of the placement of these cells along with the highly proven optimized routing using the ASIC methodology would result in high performance designs.

Thus high performance design can be obtained by judiciously employing number of custom design techniques including floorplanning, prerouting critical signals, tiling datapaths, and generating crafted cells [Dally00]. These techniques all structure the design by routing the critical wires first and then placing the devices. This ‘key wires first’ approach exploits the structure of the logic to reduce wire loads, provide early visibility into the timing and power dissipation of a design, and gives the designer control of the key wiring.

Custom designs have proven performance for regular, repetitive and hierarchical designs like VLSI Arrays (TLB, RF, cache etc). These array structures require high performance implementation as they are time critical for the microprocessor. They are mostly implemented using the custom flow requiring large implementation team and man hours and also migrating from one technology node to other requires significant rework. However if these blocks are implemented by incorporating special standard cells, a method similar to the ASIC flow and the placement is directed by the designer instead of random placement of synthesis flow and finally routed using the ASIC efficient algorithms, the output design will be regular, easy to amend and migrate across technology nodes could be easy while maintaining high performance.

1.5 ***Contribution of this work***

In the proposed methodology, we put forward a structured IC design approach where the regularity of custom IC design, together with the fast layout and routing of the ASIC approach, using the EDA tools are exploited. The proposed design approach is targeted for VLSI arrayed structures however other designs can be implemented using this flow too. In the structured approach, the design is implemented using the standard cells with designer crafted schematics. The design is then verified across various corners using the timing tools to ensure design robustness. The standard cells are then automatically placed in the floor-plan using programs developed as a part of this work. Since the designer is the best judge of how the cells should be placed to reduce the critical path delays, the placement is crafted so that the drive strengths, path delays and regularity of the design can be exploited. The fast and optimal routing of the synthesis tools are then exploited by routing the placed cells using the auto-routing tools. Fully routed design can be imported back as GDSII file.

The structured approach of building the digital circuits make the design more controllable for timing and area constraints. Since moving the standard cells in the spreadsheet is easy and comfortable. It is also repeatable and several iterations to meet the specifications can be run in few hours. Portability of the design across technology nodes is very easy. Replacing the standard cells in the spreadsheet with the equivalent driver cells of new technology makes the design ready for the next technology. Robustness is guaranteed as each step is checked using the ASIC tools like timing (Prime Time) and placement routing (Encounter). These tools can easily be run at various corners to ensure margin checks and other failure analysis.

1.6 ***Thesis Organization***

This chapter gives an introduction to custom and ASIC design methodologies. Chapter 2 highlights various ASIC and custom designs features and how to manipulate them to achieve high performance design. The proposed structured methodology of IC design which achieves performance benefits in terms of speed, area and power as of the custom design is comprehended in chapter 3. Chapter 4 discusses how to characterize the complex static and dynamic gates using the Synopsys NCX tool. Chapter 5 discusses various designs examples viz. Internet protocol content addressable memories (IPCAM) and issue logic design where proposed methodology is used to estimate the performance of the design. Chapter 6 focus on building static design like translation look-aside buffer (TLB) design and register file design; that exploits the structured approach in their implementation. A detail performance comparison in terms of area, delay and power is also presented. Chapter 7 explains yet another highly complicated cache design. Chapter 8 concludes.

CHAPTER 2. **HIGH PERFORMANCE DESIGN TECHNIQUES**

Implementation of a SOC design depends on broad range of factors:

i) Performance, power and cost constraints

ii) Design complexity

iii) Testability

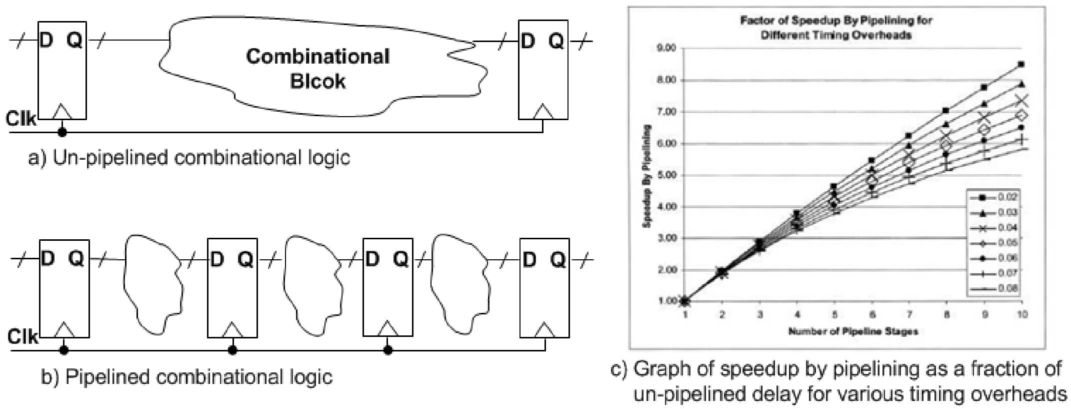
iv) Time to market and time to revenue

v) Application range covered by the design

A number of these factors tend towards more flexible, programmable components that can be reused and that can be modified even after manufacturing i.e. reconfigurable components. Finding the balance between these extremes is the ultimate challenge for the modern designer. New implementation styles have emerged over the last decades, presenting the designer with wide variety of options. Various methodologies have been proposed in order to push the performance of ASIC designs closer to that of custom designs. Since the ASIC designs are tool based, the performance improvement techniques can be grouped into two major groups. Firstly, achieving high frequency design and secondly, ensuring low power implementation.

2.1 ***Achieving high frequency design***

The speed of a digital circuit is determined by the delay of its longest critical timing path. The length of the critical timing path is a function of gate delays, wiring delays, set-up and hold-times, clock-to-Q (the delay from when a clock signal arrives at a latch to when the latch output stabilizes), and clock skew [Weste92]. As previously explained in Chapter 1, the speed of ASIC designs lags



a) Un-pipelined combinational logic

b) Pipelined combinational logic

c) Graph of speedup by pipelining as a fraction of un-pipelined delay for various timing overheads

Figure 2.1 Pipelined and un-pipelined logic implementation and the speedup from pipelining

that of the custom design in the same technology node by factors of 1.5 to 2. To improve the speed of an integrated circuit requires reducing the delay of one or more of these elements. High frequency design requires the ability to apply costly (as measured by effort) local optimizations to the design critical paths.

*2.1.1 **Micro-architecture and hardware implementation***

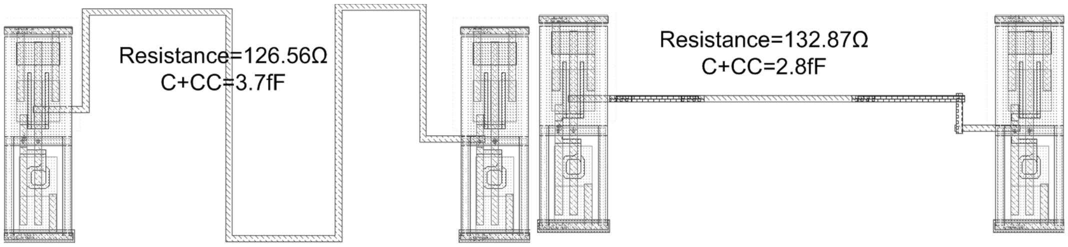
Pipelining can be productive if multiple tasks can be initiated in parallel or if there is one signal flow direction. Splitting a large operation into smaller independent operations by placing storage elements between them allows the circuit to operate at higher frequency. Figure 2.1 shows the basic un-pipelined and pipelined combinatorial logic implementation. The graph shows the estimated speedup by pipelining, assuming that the combinational logic can be split equally and flip-flops are balanced which is overly optimistic.

The increase in the throughput is at the cost of latency and complexity. Simply increasing the clock speed by adding latches would only increase latency due to the additional latch setup and hold times. By placing additional latches or registers in long chains of logic, the number of gates in critical path in each stage can be reduced and hence higher operating frequency can be obtained. With flip-flops the timing overhead is between 0.06 and 0.08 times, and this larger timing overhead substantially reduces the benefit of having more pipeline stages (refer to Figure 2.1) [Chinnery02]. Pipelining ASICs is also limited by the speed of registers in the pipeline, and greater automated clock tree generation produced skew than can be obtained in carefully designed custom clock trees.

Custom designs may show superior logic-level design of regular structures such as adders, multipliers, and other datapath elements. They achieve fewer levels of logic on the critical path with more compact, complex logic cells and by combining logic with the latches. A careful custom design can balance the logic in pipeline stages after placement, ensuring that the delays in each stage are close, whereas an ASIC may have unbalanced pipeline stages resulting in more levels of logic on the critical path.

Fast datapath designs, such as carry-lookahead and carry-select adders and other regular elements, do exist in pre-designed libraries, and are easily invoked in RTL logic synthesis of ASICs. Use of these predefined macro cells for an ASIC can significantly improve the resulting design, by reducing the number of logic levels for implementing complex logic functions and reducing the area taken up by logic [Chang98]. However these have to be compatible with the pipeline micro-architecture organization. All the operations in the macro block must complete within that particular pipeline stage.

Design of memory macro blocks like register file (RF), translation lookaside buffer (TLB), cache etc. are some of the most critical blocks to



a) Routing a net using single M2 in zigzag fashion

b) Routing a net using multiple metals in straight line

Figure 2.2 Routing patterns and resistance/capacitance associated with the net

incorporate in the pipeline stage. Since these blocks are very time critical, they are usually custom crafted in high performance processor; however all vendors have RAM compilers that generate memory blocks [Shinohara91]. Most of these blocks, being in the critical timing path of the pipeline stage, decide the maximum clock period. Consequently, dynamic logic is incorporated in the worst case delay paths to optimize the delay, which is achievable using custom circuits only.

2.1.2 ***Wire-delay Optimization***

Wire-delays associated with "global" wires between physical modules can be a dominant portion of the total path delay [Xing02]. The delay associated with wires depends on the length of the wire, the width and aspect ratios of the wire, and on proper drive [Gyurcsik89]. Properly driving a wire requires not only sizing of drivers, but insertion of repeaters. However, the primary factor in wire delay is wire length. Wire length is obviously dependent on placement, which in turn depends on floor-planning. However, it is also influenced by the quality of routing. Most reported systems use classical routing algorithms such as maze, line-probe, river routing and the left edge algorithm [Lefebvre97]. Routers based on the above combination generally rely on some net classification scheme based on the location of terminals associated with each net [Ong89]. Routing on a single

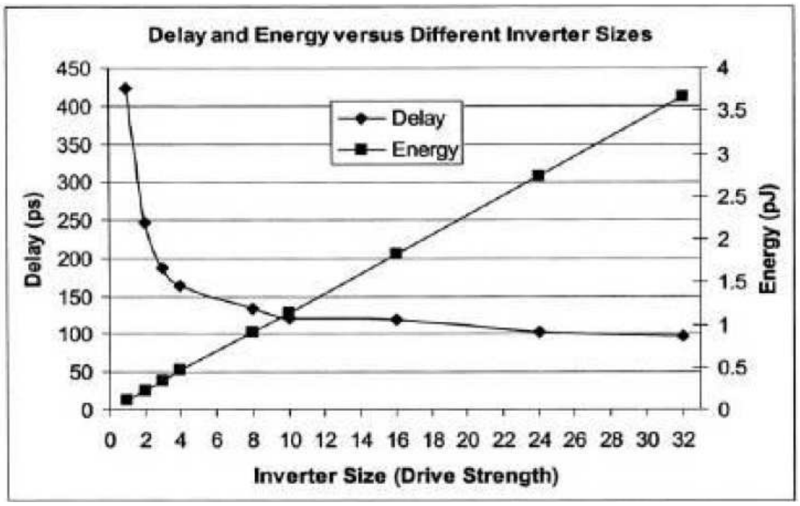


Figure 2.3 Delay and energy for increasing inverter sizes

metal layer incurs less delay. Similarly for zigzag as against straight wires with several metal layers switching in between. This is because of vias that are required to shift between metal layers. These vias have higher resistance and capacitance. Use of several of these can incur higher path delay. Figure 2.2 shows an example for such layout. The total capacitance (including the coupling capacitance) is less for the straight routing with many metals but the resistance is higher that eventually increases the path delays.

Custom ICs are typically manually floor-planned. Tools with the capacity to identify similar structures that may be abutted or placed next to each other appropriately will reduce area, reducing wire lengths and increasing performance. A bit slice may be laid out automatically then tiled, rather than the circuitry being placed without considering that it may be abutted [Chang98]. Regular data paths can be best laid out by hand or by tiling slices for abutment.

2.1.3 ***Transistor sizing***

Based on the timing constraints, each transistor in a custom design is sized to meet the drive requirements of the capacitive load it faces. Increasing the sizes lowers the delay until the intrinsic capacitances start dominating as shown in Figure 2.3. Wires may be widened to reduce the delays (proportional to the product of resistance and capacitance) by reducing the resistance. Additional buffers may be included to drive large capacitive loads that would otherwise be charged and discharged too slowly. However, ASIC methodology requires cell selection from a fixed library, where transistor sizes and drive strengths are constrained to the library, and routing wire widths are usually fixed. Fundamentally, the discrete transistor sizes of a library only approximate the continuous transistor sizing of a custom design. ASIC cells typically include design guard banding, such as fully buffered flip-flop inputs, which introduces overhead. Hence a rich library of sizes with varied drive strengths and buffer sizes, as well as dual polarities for functions (gates with and without negated output) [Scott94] can impact the performance of ASIC design drastically. A richer library also reduces circuit area [Keutzer87].

Initial logic synthesis chooses drive strengths using estimated wire lengths and the net load a gate has to drive, but this will differ from that in the final layout. After placement gates are resized accounting for the drive strengths required to send signals across the circuit, and buffers are inserted or removed as necessary. Smaller transistors are used to reduce power consumption. However on critical paths transistors are optimally sized to meet speed requirements. They can make a speed difference of 20% or more [Fishburn85]. Later arriving signals can be routed closer to the gate output and transistors moved to maximize the adjacent drains and sources for diffusion sharing [Hill85]. Iterative gate resizing and resynthesis can improve speeds by 20% [Gavrilov97].

2.1.4 ***Design using Dynamic Gates***

Dynamic logic is significantly faster than static CMOS logic and requires less area, but requires very careful design to ensure there is no glitching of input signals, limited cross talk between neighboring signals, no charge sharing and latching at the end to hold the evaluated value during the precharge phase. Nonetheless, they can be used to speed up critical paths within the circuit.

Dynamic logic libraries are not available for ASIC design, because of the difficulties and care required by dynamic logic. Design with dynamic logic requires careful consideration of noise and power consumption. Dynamic logic is particularly susceptible to noise, as any glitches on input voltages may cause a discharge of the charge stored, which should only occur when the logic function evaluates to false. Additionally, dynamic logic has higher power consumption, requiring careful design of the power distribution, and of the clock distribution as well; the clock determines when precharging occurs, and inputs must not glitch during or after the precharge. These problems become more pronounced with deeper submicron technologies.

2.1.5 ***Reducing uncertainty in the design process***

A higher frequency of operation can be achieved by accurately determining which paths in the design are critical and then applying local optimization to those parts [Stephen01]. In order to differentiate the critical from the non-critical paths, design must have minimal uncertainties in the design process. This enables one to apply global optimizations to the non-critical portions of the design and limit the costs of local optimizations to the critical paths.

Design uncertainty is caused by inaccuracy in the design analysis tools or unpredictable variations in the design between iterations. The unforeseen nature of the routing causes significant variations from iteration to iteration which may lead to reordering of the paths. This reduces the confidence that the critical path in any iteration will continue to be the critical path in future iterations. As the frequency increases, the tolerance for design uncertainty drops dramatically as the uncertainty becomes a larger percentage of the total cycle time. The sources of design uncertainty grow as the frequency increases and second order electrical effects become first order effects [Bai02]. By using a better design practice, such as prerouting, the uncertainty of routing can be significantly reduced between iterations. This helps to stabilize the path ordering attributed to such variations [Kheterpal04].

2.2 ***Achieving low power design***

For a specific application, the energy efficiency of different implementations can differ by multiple orders of magnitude [Chang05]. Implementation that uses dynamic logic dissipates more power due to higher activity factor than those that uses static logic to implement the same logic. Also, circuits sized with higher fanout also dissipate lesser power due to lower m wiring capacitance. Every design has a unique power versus performance characteristic. Maximizing the energy efficiency of a design enables the minimization of power dissipation by creating the largest range of trade-offs between performance and power. Increasing the efficiency of a design and reducing the power necessary to deliver the required application performance are achieved by accomplishing one or more of the following three basic goals:

i) Moving along the power versus performance curve towards a more efficient operating point.

ii) Reducing power dissipation by operating at a lower performance and lower-dissipation point.

iii) Moving to a different power-performance curve by changing either the architecture or the process node.

For a fixed process technology, architectural choices have the greatest impact on the system's energy and power efficiency as they potentially enable the design to operate on an improved power-performance curve [Dally00] [Chinnery02].

2.2.1 ***Dynamic Energy Optimization***

Dynamic power dissipation is due to charging and discharging load capacitance as gates switch, as well as and short-circuit current while both PMOS and NMOS stacks are partially ON. Digital circuit dynamic power consumption (assuming constant frequency clock and balanced number of 0-to- 1, 1-to-0 transitions) is

$$P_{dyn} = \alpha C V_{DD}{}^{2} f \tag{2.1}$$

where α is the activity factor, and *f* is the switching frequency.

Hence the dynamic energy can be reduced by reducing any of these parameters. However the exponential dependence on V_{DD} makes it the more appealing parameter to minimize. The design can have dynamic lowering of supply voltage or creation of distinct voltage islands to reduce the energy dissipation. Lowering the supply voltage or gating the supply to the blocks are some of the techniques adopted to achieve lower energy [Borkar01]. Explicitly disabling unnecessary portions of the chip through clock-gating and/or block enables or dynamic frequency scaling are the methods to lower the frequency and the activity factor of the design. Redundant transitions in the address or data bus can be reduced by bit encoding [Weng05]. Elimination of glitches by optimizing the delays in converging paths will reduce the activity factor and subsequently the energy dissipation. Load matching, where each gate is sized according to its load driving capability, and parasitic reduction reduces the capacitive loads.

Switching capacitance comes from the wires and transistors in a circuit. Wire capacitance is minimized through good floorplaning and placement. Device switching capacitance is reduced by choosing fewer stages of logic and smaller transistors. Near minimum sized gates can be used on non-critical paths. Although logical effort finds that the best stage effort is 4, using a larger stage effort increases delay only slightly and greatly reduces the transistor sizes. Therefore, gates that are larger or have a high activity factor and thus dominate the power can be downsized with only a small performance impact as is depicted in Figure 2.3.

2.2.2 ***Optimizing the Static power dissipation***

In nanometer processes with low threshold voltages and thin gate oxides, leakage can account for as much as a third of total active power. The main sources of static power dissipation in the digital ICs are:

i) Sub-threshold leakage through OFF transistors

ii) Gate leakage through gate dielectric

iii) Junction leakage from source/drain diffusions

In semiconductor devices, the leakage power is an increasingly important contributor to overall design power and at sub-micron level; leakage power can be the dominant component of power consumption in specific applications. Two main contributors are sub-threshold leakage current (I_{sub}) and gate-oxide leakage current (I_{ox}). The basic equations for digital circuit static power consumption [Chandrakasan01] are:

$$P_{static} = V_{DD} \times (I_{sub} + I_{ox}) \tag{2.2}$$

$$I_{sub} = K_1 We\left(-\frac{V_t}{nV_\theta}\right)(1 - e\left(-\frac{V_{gs}}{V_\theta}\right)) \tag{2.3}$$

$$I_{ox} = K_2 W\left(\frac{V_{gs}}{t_{ox}}\right)^2 e(-\frac{\alpha t_{ox}}{V_{gs}}) \tag{2.4}$$

where K_1, K_2, α and n are empirically determined and W is the transistor width, V_{DD} is the supply voltage, V_{gs} is the gate-to source voltage, V_t is the threshold voltage, and V_θ is the thermal voltage (kT/q, 25mV at 25 °C).

Similar to dynamic energy optimization, V_{DD} reduction also assists in reducing the static power. Power gating is a prominent technique in which power supply is gated off to the sleeping blocks [Agarwal06]. However, it requires the state to be saved or reset upon power-up [Little92]. Substituting high threshold devices in non-critical logic paths or by employing transistor stacking to generate negative body-to-source voltages, negative V_{gs} and reduce the effect of drain-induced barrier lowering (DIBL) on V_t and introducing body-bias (either static or active) to increase the effective V_t which eventually reduces the static power. The leakage through two or more series transistors is dramatically reduced on account of the stack effect.

2.3 ***Optimizing placement of standard cell***

Analyzing the various methodologies in the previous sections, interconnect routing seems to be one of dominant criteria for both speed and power. Large wire load requires either large driving devices that increase power dissipation or more buffering that incurs higher delay. For high performance chips, it is important to route each net such that it meets the timing requirement.

It has been well known that the placement quality has a profound impact on the routing complexity and hence the layout compactness. The length of the interconnect route directly depends on the placement of the cells. Timing assurance placement and routing methods have been reported previously [Dunlop84] [Burstein85] [Ogawa86] [Teig86] [Hauge87] [Jackson89]. Priority weight assignment for controlling wire length in automatic layout was proposed in [Dunlop84]. When a signal delay is above a specified upper bound after layout design, a high priority weight is assigned to the corresponding net and the placement and routing are rerun using the updated weight. The greater the weight of a net connecting two cells together, the closer they tend to be placed. However, in practice, when many nets are given high weights, the wire lengths of nets with high weights are not always short in the resulting layout. Thus, it needs various weights to control wire lengths, resulting in long design time. The methods proposed in [Burstein85] [Ogawa86] [Teig86] [Hauge87] are variations of net weighting. In a recent study [Jackson89], a performance-driven placement approach using LP (linear programming) optimization was proposed [Terai90].

Min-cut placement is one of the most practical and widely used placement techniques. It performs two dimensional placements [Dun83] to reduce the wiring density by partitioning the cells into two groups. In order to partition the cells above and below a horizontal line, the algorithm first determines the set of nets that must exist above and below that line. It then partitions the cells into two groups of approximately equal size. Once the partition in Y has been accomplished, it repeats the process in X, then in Y again, etc. At each step, it calculates the set of necessary connections on each side of the partition and uses this to guide the min-cut [Hill88].

For a net whose pin-pairs are too far apart, the constraint will not be meet even if it is interconnected with the shortest routing path. Thus the constraint becomes more important in the placement process than in the routing process. For this reason, a placement algorithm that controls wire lengths of nets is essential for timing assurance post-layout. Since placement has been proven to be computationally difficult, most modem placement is done in consecutive steps. First a global placement is carried out that tries to roughly spread the cells on the chip as evenly as possible. Then legalization of these cells, to align them in rows without overlaps follows. Finally, a detailed placement that further optimizes the legal placement to improve the placement quality according to certain criteria is carried out [Yanwu10].

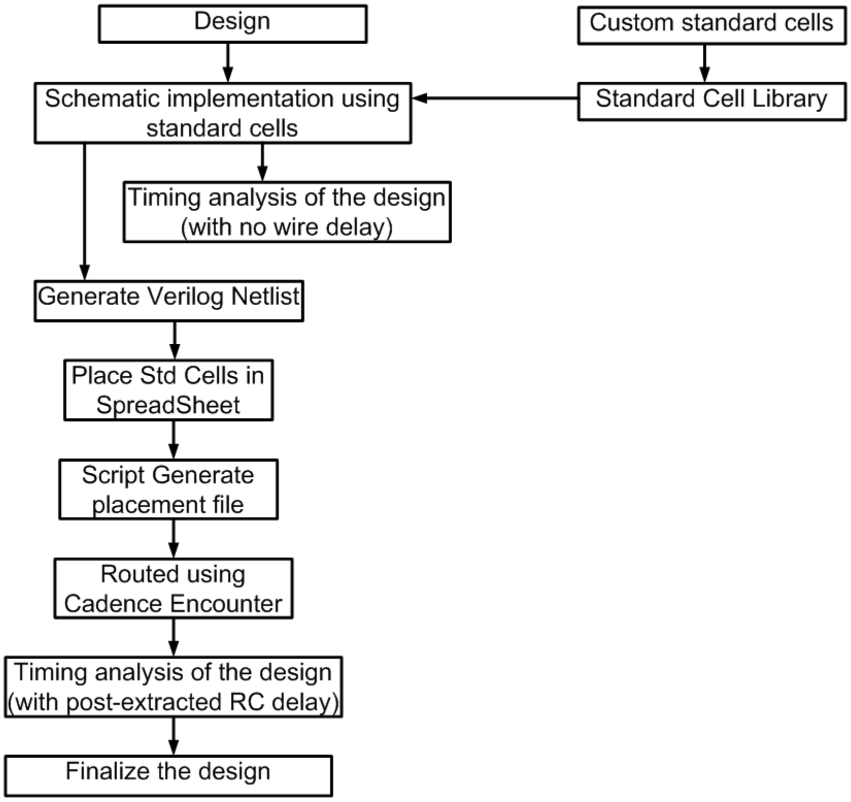
The primary disadvantage of using this approach is the amount of CPU time required. The common weaknesses of the aforementioned, and most other existing cell placement algorithms, is that they all ignore the circuit properties. So in order to cope with this issue, the proposed structured methodology relies completely on the knowledge of the circuit designer for the placement. The design is built semi-custom using the standard cells and the designer places each of these standard cells in a spreadsheet as per the logic and also comprehends the timing of the design. Since the designer is the best judge for circuit functionality and optimization for the structured flow, the design is highly regular and can be repeated with minimal efforts as compared to the stochastic floor-plan of the standard cells every time placement effort is made in the ASIC flow. All the algorithms in the ASIC flow seeds a random number to start an initial placement and then optimize the overall floor-plan based on the constraints. Details of the proposed flow are highlighted in the following chapters.

CHAPTER 3. **PROPOSED STRUCTURED METHODOLOGY OF IC DESIGN**

Achieving custom performance using ASIC methodologies is a big challenge for the IC industry. However, the required time, reliability and resource requirements of custom design are again a challenge. The blend of the two techniques where the ease of circuit design using standard cells with appropriate sizing and optimal logic implementation, along with the ease of placement and routing of the standard cells, seems to be a captivating solution. They not only allow the designer achieve higher productivity, circuit speed, lower power and lower area, but also facilitates the portability of the design from one technology node to other in a short span of time. In the proposed IC design methodology, both the custom techniques and the ASIC design steps are jointly exploited to achieve the desired performance with low complexity, ease of repeatability and methods to verify the design both functional and timing so as to develop a robust design.

3.1 ***Overall flow of the structured methodology***

The custom design requires analyzing the complete design architecture and micro-architecture level to gain maximum optimization. The design is then implemented in the schematic editor using the standard cell libraries. The regularity of the array based design makes this approach productive. Custom standard cell required to meet the requirement are also added to the libraries. Analysis of the design across multiple corners is subsequently carried out for timing verification using timing tools like Synopsys Prime Time. At this point,



Design
Custom standard cells
Schematic implementation using standard cells
Standard Cell Library
Timing analysis of the design (with no wire delay)
Generate Verilog Netlist
Place Std Cells in SpreadSheet
Script Generate placement file
Routed using Cadence Encounter
Timing analysis of the design (with post-extracted RC delay)
Finalize the design

Figure 3.1: The proposed structured design flow

since the design is not routed, the wire delay is ignored. However, the timing tool checks for the proper drive strengths of standard cells by analyzing the transitions at every node in the critical path. Optimal use of the standard cells and proper signal transition rates helps not only to achieve optimal delay but also lower energy dissipation in the design.

Once design at the schematic level is finalized, the floor-planning of these standard cells is carried out. This is done by placing the cells in a spreadsheet, each cell of which corresponds to each instantiation of the standard cell in the design. Since global wires between physical modules are a dominant portion of the total path delay, proper placement of the standard cells and their drivers make the path delay within the design limit, thus achieving high speed. The spreadsheet

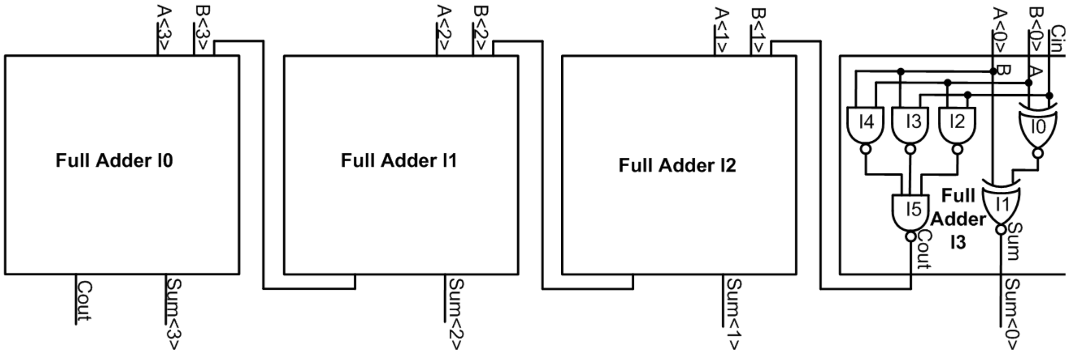


Figure 3.2: Schematic of the 4-bit adder

is then read by Perl scripts that convert it into a placement file readable by the Cadence Encounter routing tool for actual placement of the standard cells in its floor-plan. The floor-plan in the Cadence Encounter can be analyzed and the standard cells can be replaced to obtain better area. By simply swapping or shifting the placed cells in the spreadsheet and re-running the Perl scripts to obtain the new placements, the task is productive and allows designer flexibility of moving the cells across for area and delay optimization.

As soon as the Encounter floor-plan is settled, the routing of all the cells is carried out using the Encounter auto-route tool. This is the step where the ASIC design methodology is used. Since the algorithms for routing are resonably optimal and regular, they can give similar results as manual routing but at far less effort. The netlist and the timing information (that captures the estimated wire delays) from the routed block can then be extracted. The timing tool then re-analyzes the block but this time with estimated wire RC delays. Changes, if required, can be carried out in the spreadsheet for placement changes or in the schematic for better drive strengths in case the cell is driving long wire. Since

```
// Verilog HDL and netlist files of
// "4-bit_adder 4-bit schematic"

// Netlisted models

// netlist file - RHBDcellLibSF_NDHv4, nand3x0018, schematic.
ihnl/cds0/netlist
// netlist file - RHBDcellLibSF_NDHv4, nand2x010, schematic.
ihnl/cds1/netlist
// netlist file - RHBDcellLibSF_NDHv4, xnor2x010, schematic.
ihnl/cds2/netlist
// netlist file - 4-bit_adder, full_adder, schematic.
ihnl/cds3/netlist
// netlist file - 4-bit_adder, 4-bit, schematic.
ihnl/cds4/netlist


// End HDL models

// globals file
hdlFilesDir/cds_globals.v
```

Figure 3.3 Output of the verilog.infile

these steps are fast and require less effort, the proposed methodology generates a design with better performance, regularity, repeatability, and ease of portability. The main labor required by the designer is only in making the micro-architecture of the design and circuit design as embodied by the schematic.

*3.2 **Generating the placement file from Perl scripts***

Details of the Perl script usage and steps to obtain the final placement file are explained in the following sections. For better understanding the methodology, an example with 4-bit adder design is also presented.

*3.2.1 **Creating the flatten netlist file***

Once the micro-architecture of the design using the designated standard cell library is completed, the schematic of the design is created in the Virtuoso Schematic editor. Verilog-XL is invoked by selecting tools → simulations. In order to set the netlisting options, select the setup tab and Netlist in the Virtuoso Verilog Environment for Verilog-XL integration window. In the Verilog Netlisting Options window check *Generate Test Fixture Template*, and *Netlist Explicitly.*All other options must be unchecked. *Generate Test Fixture Template*

```
standard cell
ihnl/cds0/netlist
ihnl/cds1/netlist
ihnl/cds2/netlist
macro cell
ihnl/cds3/netlist
ihnl/cds4/netlist
```

Figure 3.4 Content of netlist_file

```
I3:I5:nand3x0018
I3:I4:nand2x010
I3:I3:nand2x010
I3:I2:nand2x010
I3:I1:xnor2x010
I3:I0:xnor2x010
I2:I5:nand3x0018
I2:I4:nand2x010
I2:I3:nand2x010
I2:I2:nand2x010
I2:I1:xnor2x010
I2:I0:xnor2x010
I1:I5:nand3x0018
I1:I4:nand2x010
I1:I3:nand2x010
I1:I2:nand2x010
I1:I1:xnor2x010
I1:I0:xnor2x010
I0:I5:nand3x0018
I0:I4:nand2x010
I0:I3:nand2x010
I0:I2:nand2x010
I0:I1:xnor2x010
I0:I0:xnor2x010
```

Figure 3.5 Content of hierarchy.txt

option generates a verilog.infiles file that has path to all the verilog netlist of the modules in the design and *Netlist Explicitly* option instantiates the modules and connects the signals by name.

In the stimulus tab in Virtuoso integration window select *verilog*. In the directory specified, there will be a file named verilog.inpfiles. It has the entire models name with the verilog netlist file paths.

Finally, create a new text file with the first line as “standard cell”. Then copy all the netlist file paths from verilog.inpfiles that are the basic standard cells in subsequent lines. Following this enter “macro cell” and copy all the remaining netlist file paths. The top level module verilog path must be at the bottom of the

```
module nand3x0018 ( o, a, b, c );
output  o;
input  a, b, c;
endmodule

module nand2x010 ( o, a, b );
output  o;
input  a, b;
endmodule

module xnor2x010 ( o, a, b );
output  o;
input  a, b;
endmodule

module full_adder ( Cout, Sum, A, B, Cin );
output  Cout, Sum;
input  A, B, Cin;

nand3x0018 I5 ( .c(a_nand_b), .b(b_nand_cin), .a(a_nand_cin),
     .o(Cout));
nand2x010 I4 ( .a(A), .o(a_nand_b), .b(B));
nand2x010 I3 ( .a(Cin), .o(b_nand_cin), .b(B));
nand2x010 I2 ( .a(Cin), .o(a_nand_cin), .b(A));
xnor2x010 I1 ( .b(B), .a(a_xor_b), .o(Sum));
xnor2x010 I0 ( .b(A), .a(Cin), .o(a_xor_b));
endmodule

module 4-bit_adder ( Cout, Sum_0_, Sum_1_, Sum_2_, Sum_3_,
A_0_, A_1_,
     A_2_, A_3_, B_0_, B_1_, B_2_, B_3_, Cin );
output  Cout, Sum_0_, Sum_1_, Sum_2_, Sum_3_;
input  A_0_, A_1_, A_2_, A_3_, B_0_, B_1_, B_2_, B_3_, Cin;
full_adder I3 ( .Cin(Cin), .B(B_0_), .A(A_0_), .Sum(Sum_0_),
     .Cout(co_0_));
full_adder I2 ( .Cin(co_0_), .B(B_1_), .A(A_1_), .Sum(Sum_1_),
     .Cout(co_1_));
full_adder I1 ( .Cin(co_1_), .B(B_2_), .A(A_2_), .Sum(Sum_2_),
     .Cout(co_2_));
full_adder I0 ( .Cin(co_2_), .B(B_3_), .A(A_3_), .Sum(Sum_3_),
     .Cout(Cout));

endmodule
```

Figure 3.6 Content of netlist.v

file. This text file is then given as input to the Perl script *netlist_merge.pl*. The script generates two output files namely *netlist.v* and *hierarchy.txt.*

*3.2.2 **Parsing the Library Exchange Format (LEF) file***

Since the proposed methodology uses Cadence Encounter for layout and routing, the LEF file is needed as an input to the tool. It represents physical layout information on components of an integrated circuit in an ASCII format. It explains all required information for a place and route of the components without the need to include the full physical design. Besides the component information it also stores some technology information on conducting layers and via layers. The LEF of all the standard cells in the library is usually generated using the Cadence Abstractgen tool.

The program *lef_parse.pl* reads the LEF files of the standard cell libraries and the custom cells and generates the file *"lef_detail.txt"* file which has information about the cells height and width. The Perl script file can read multiple LEF files, each separated by a space. This information is used by the subsequent scripts to actually place the cells in the floor-plan.

3.2.3 ***Creating the layout plan in the spreadsheet***

The whole crux of the proposed methodology lies on the ability to place the standard cells used in the design in a spreadsheet and then finally read this spreadsheet to the actual floor-plan in the Cadence Encounter.

A spreadsheet was chosen since it is easily editable in two dimensions. The details of the standard cells used in the design are obtained from the *hierarchy.txt* generated by the Perl script *Netlist.pl* with proper hierarchy information of each cell as in the schematic design. The Perl script *excel_parse.pl* then reads this spreadsheet and generates and output file *layout.txt* that captures the row-wise and column-wise information of the standard cells arranged in the spreadsheet. If the design is very large all the cells cannot be accommodated in a single spreadsheet. In this case, placement of the standard cells may extend to multiple sheets. The parameter *$split_sheet* is used to let the script know if the sheet is extended column-wise (*$split_sheet=1)* or row-wise (*$split_sheet=0)*.

3.2.4 ***Creating the placement file***

Once the *netlist.v, hierarchy.txt, lef_detail.txt* and *layout.txt* files are generated, the Perl script *Place.pl* then uses these files to generate the final placement file named <*top_module_name*>*.place*. This file can be zipped and loaded as a placement file in Encounter. The Perl script has three configuration variables viz., *$layout_margin, $std_cell_height and $column. $layout_margin* sets the margin across the core area in the layout floor-plan. Since the complete design is carried out using the standard cell which has specific height and variable width, *$std_cell_height* sets this height. It is also the distance between successive power rails. The variable *$column* is used to pack a group of columns together. The group of cells are then compacted together to lower the wire delays between them. This feature is extensively used in design where the design can be split into multiple columns, such as a translation look-aside buffer design (6.1) or register file (6.2) design.

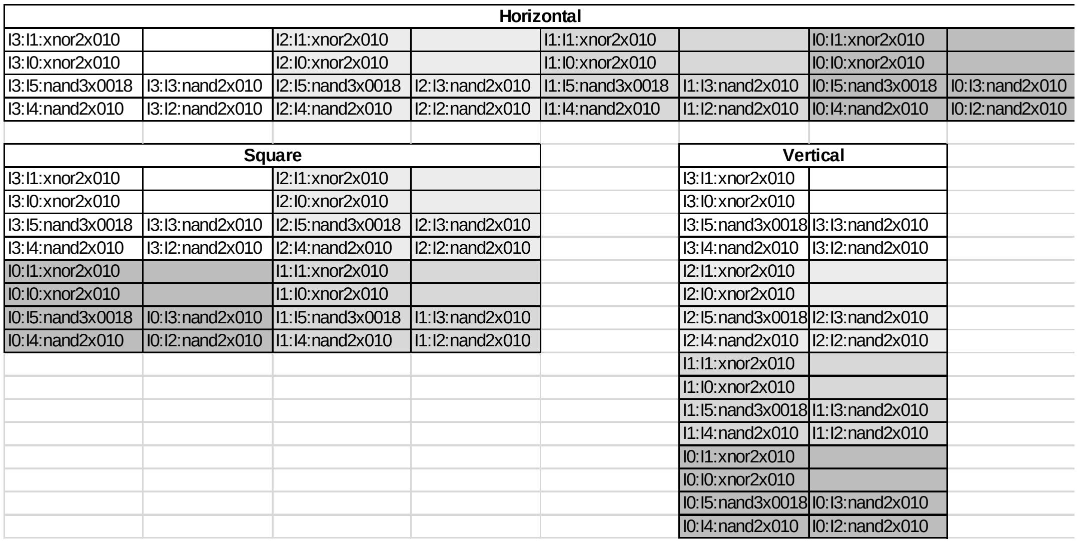
The Perl script *Place.pl* includes built-in checkers to ensure that cells placed in the spreadsheet match with those of the schematic design. The script checks for the following discrepancies:→

i) Any cell that is not in the design but is placed in the spread-sheet

ii) Any cell that is in the design but not in the spread-sheet

iii) Multiple placement of same cell in the spreadsheet.

An estimate of the floor-plan height and width is also printed on the console that can be used to set the floor-plan height and width once in the Encounter.



Horizontal							
I3:I1:xnor2x010		I2:I1:xnor2x010		I1:I1:xnor2x010		I0:I1:xnor2x010	
I3:I0:xnor2x010		I2:I0:xnor2x010		I1:I0:xnor2x010		I0:I0:xnor2x010	
I3:I5:nand3x0018	I3:I3:nand2x010	I2:I5:nand3x0018	I2:I3:nand2x010	I1:I5:nand3x0018	I1:I3:nand2x010	I0:I5:nand3x0018	I0:I3:nand2x010
I3:I4:nand2x010	I3:I2:nand2x010	I2:I4:nand2x010	I2:I2:nand2x010	I1:I4:nand2x010	I1:I2:nand2x010	I0:I4:nand2x010	I0:I2:nand2x010

Square			
I3:I1:xnor2x010		I2:I1:xnor2x010	
I3:I0:xnor2x010		I2:I0:xnor2x010	
I3:I5:nand3x0018	I3:I3:nand2x010	I2:I5:nand3x0018	I2:I3:nand2x010
I3:I4:nand2x010	I3:I2:nand2x010	I2:I4:nand2x010	I2:I2:nand2x010
I0:I1:xnor2x010		I1:I1:xnor2x010	
I0:I0:xnor2x010		I1:I0:xnor2x010	
I0:I5:nand3x0018	I0:I3:nand2x010	I1:I5:nand3x0018	I1:I3:nand2x010
I0:I4:nand2x010	I0:I2:nand2x010	I1:I4:nand2x010	I1:I2:nand2x010

Vertical	
I3:I1:xnor2x010	
I3:I0:xnor2x010	
I3:I5:nand3x0018	I3:I3:nand2x010
I3:I4:nand2x010	I3:I2:nand2x010
I2:I1:xnor2x010	
I2:I0:xnor2x010	
I2:I5:nand3x0018	I2:I3:nand2x010
I2:I4:nand2x010	I2:I2:nand2x010
I1:I1:xnor2x010	
I1:I0:xnor2x010	
I1:I5:nand3x0018	I1:I3:nand2x010
I1:I4:nand2x010	I1:I2:nand2x010
I0:I1:xnor2x010	
I0:I0:xnor2x010	
I0:I5:nand3x0018	I0:I3:nand2x010
I0:I4:nand2x010	I0:I2:nand2x010

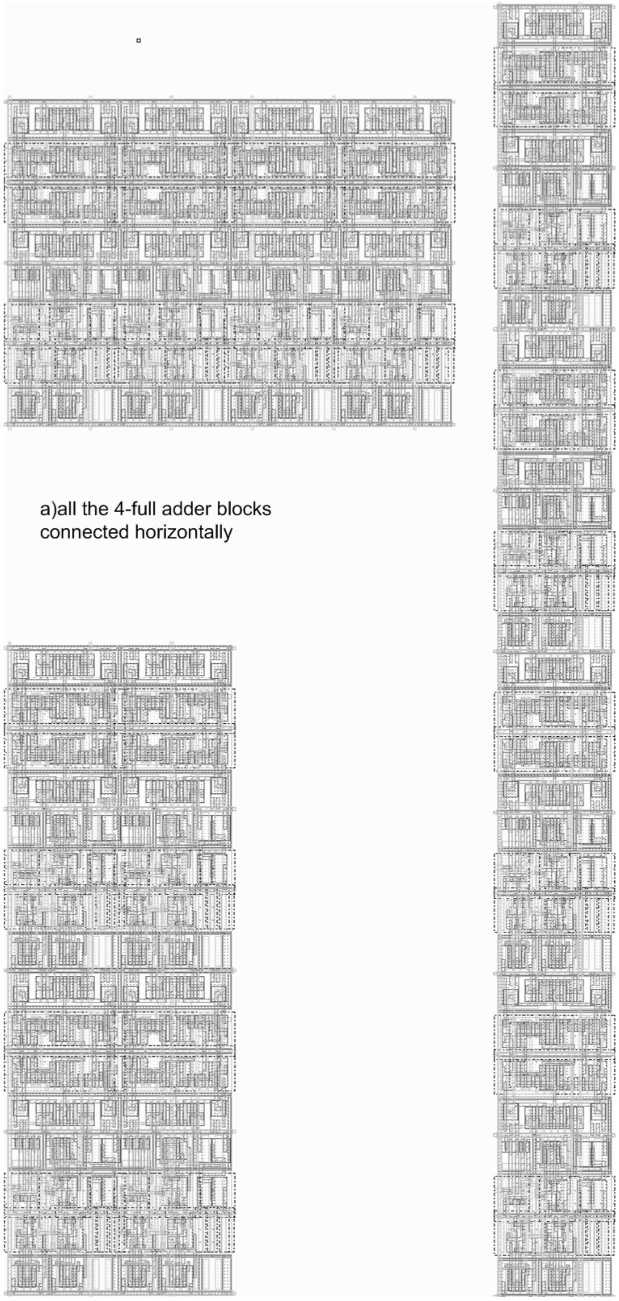
Figure 3.7 Spreadsheet for various layouts of the 4-bit adder block. Each full adder block is shown with different shades to differentiate with other blocks

3.3 ***4-bit adder design example***

In order to better understand the methodology, an example of a simple 4-bit ripple carry adder is presented in this section. The schematic for the design is shown in Figure 3.2. The detailed instance number is also shown for all the blocks and standard cells. The schematic is built in the Virtuoso Schematic Editor and the verilog netlists of each module is extracted using the Verilog-XL simulator. The content of the *verilog.inpfiles* generated for this design looks like as shown in Figure 3.3. From this, another text file *netlist_file* as shown in Figure 3.4 is created that defines the *standard_cell* and *macro_cell* verilog netlist file locations.

This file is read by the program *netlist_merge.pl* that generates two files *hierarchy.txt* and *netlist.v*, the output of which are shown in the Figure 3.5 and Figure 3.6, respectively. Referring to *hierarchy.txt* and comparing it against Figure 3.2, the names of all the standard cells are followed by their exact hierarchy locations. The file *hierarchy.txt* is the list of the instance names and cell types for each instance. Also, in the *netlist.v* file, the details of the standard cells are not elaborated. Only the interface input and output pin declaration is given. However, for the macro block, complete details with instantiation order of the standard cells and sub-blocks are outlined. The instantiation of the standard cells in the macro block use net connection by name.

The *hierarchy.txt* file is used to place the layout in the spreadsheet. The complete names of the standard cells with their instantiation details are used in the spreadsheet. Various placement styles viz., horizontal, square and vertical placement of the full-adder blocks are shown in the Figure 3.7. By simple cut and paste operation, different styles can be realized. The spreadsheet is read by the *excel_parge.pl* program that generates the *layout.txt* file. This along with *netlist.v*, *hierarchy.txt* and *lef_details.txt* (generated using *lef_parge.pl* script and all LEF files as input) are used by the *place.pl* to generate the placement file *4-bit_adder.place.* Finally, this file is zipped and loaded into Cadence Encounter. Referring to the Figure 3.7, each of the full adder block occupies two columns, hence the variable *$column* is set to 2 so that the two columns are compacted together. Auto routing is then invoked to connect all these blocks or cells together. Saving the GDSII file and importing it in to Virtuoso gives the complete layout. The layouts of all the different styles were carried out and are shown in Figure 3.8. The standard cells used in this example are the RHBD cells developed for the HERMES microprocessor on the 90nm IBM foundry process.



a)all the 4-full adder blocks connected horizontally

b)all the 4-full adder blocks connected in square fashion

c)all the 4-full adder blocks connected vertically

Figure 3.8 Different layouts of the 4-bit adder cells when the four full-adder blocks are placed differently

CHAPTER 4. **CHARACTERIZING THE COMPLEX GATES**

The basic building block of the ASIC flow is the standard cell (also called the primitive blocks) which are used to build the complete design. Standard cells are logic units of similar geometry that have same height but variable widths. Even for the proposed structured methodology, the standard cells are an essential block. However analyzing the design using the HSPICE netlist or GDSII format is very cumbersome as the designs are complicated. Functional and delay simulations takes a long time for a large design. Power estimation for the block is difficult as it depends on variety of simulations to estimate the average power. The detection of the timing constraints such as setup and hold for a particular block needs specific scenario simulations. All these consume time and also the results may be biased towards the simulation condition chosen Timing and delay information is important to the ASIC design process because it allows a level of realism to be incorporated into the circuit model. Typically, a Verilog hardware model does not include this timing information; the outputs of modules are in effect resolved instantaneously. This strips the designer of important details such as propagation delay, pulse width, setup and hold times. Incorrect state transitions from setup and hold violations could very easily lead to the design not functioning.

In order to accelerate the timing analysis of the designs built using standard cells, the ASIC flow uses timing tools like Synopsys Prime Time. These tools reads in a simple model for delay, function, constraints and power on cell level embodied in a cell characterization file (.lib). The simulator then automatically sets the appropriate delays inside the specify block. Essentially, Verilog and the simulator have a built in ability to handle the changing of delays within a specify block. Complete timing closure of the design is depends on the accuracy of the characterization file.

4.1 ***Timing analysis delay models***

The accuracy and speed of the timing analysis tool depends solely on the delay models of the cells in the characterization file. These delay models are much faster than the HSPICE simulations and are also more accurate than the simple linear delay model. Timing, area, power and noise models for each gate in a standard cell library are stored in .lib file that adhere the Liberty standard [Opensource]. Logical effort parameters for standard cells can be obtained by interpolation of the timing models, assuming equal delays and rise/fall times for the previous stage.

4.1.1 ***Generic CMOS delay model***

The Generic CMOS delay model evaluates the delay based on the intrinsic delay, slope delay, transition delay and interconnect delay. Assuming the slope of the input is proportional to the delay of the previous stage, the delays for rising and falling outputs can be expressed as:

Delay = intrinsic_delay + transition delay + slope_delay + interconnect delay (4.1)

Intrinsic delay is a fixed value which models delays independent of the surroundings. Slope delay is produced by the slew of the input signal. The input slew is multiplied with a sensitivity factor to compute the slope delay. Transition delay is the time required to charge the capacitance of the next stage input pins. It is equal to the product of the driving resistance as determined by the driving current and input capacitance. Interconnect delay is delay produced by the RC value of the wire to the next stage input pin and the capacitance of the next stage input pins.

Linear delay models are not accurate enough to handle the wide range of slopes and loads found in synthesized circuits, so they have largely been superseded by nonlinear delay models.

4.1.2 ***Nonlinear delay model***

The nonlinear delay model is the most widely used model for timing analysis. The delay and transition time are modeled as functions of input slew and output load. The data is stored as a two dimensional lookup table. It looks up the delay from the table based on the load capacitance and the input slope. Separate tables are used to lookup rising and falling delays and output slopes. The data points in the table are usually not equidistant. The timing analysis tool uses interpolation when a specific load capacitance or slope is not in the table.

The nonlinear models also do not contain enough information to characterize the delay of a gate driving a complex RC interconnect network with the accuracy desired by some designers. They also lack the accuracy to fully characterize noise events. Timing analysis also must be done at different process corners to ensure robustness of the design. This requires different supply voltage and operating temperature. The nonlinear models must be created for each voltage and temperature at which the chip needs to be characterized, since device characteristics vary with these parameters.

4.1.3 ***Current source model***

The current source model was developed to overcome the shortcomings of the nonlinear model. A current source model express the output DC current as a nonlinear function of the input and output voltages of the cell. A timing analyzer numerically integrates the output current to find the voltages as a function of time into an arbitrary RC network and to solve for the propagation delay. The current source model scales the current waveform for the highest accuracy. It supports IR drop, multi-V_{DD} and dynamic voltage and frequency scaling characterization.

The Liberty composite current source model (CCSM) stores output current as a function of time for a given input slew rate and output capacitance. The competing effective current source model (ECSM) stores output voltage as a function of time. The two representations are equivalent and can be synthesized into a true current source model.

Unfortunately, this model is not fully incorporated as industrial standard. Thus, most characterization library files still have NLDM formats.

4.2 ***Characterizing the standard cells***

For characterizing the standard cells used in the design employs Synopsys NCX tool to characterize them. The library model developed is of NDLM type. It accepts the post extracted netlist of the standard cells. The process corner parameters are specified as PVT-corners. A template library format file is also given as input that defines the wire-load models and input transition and output

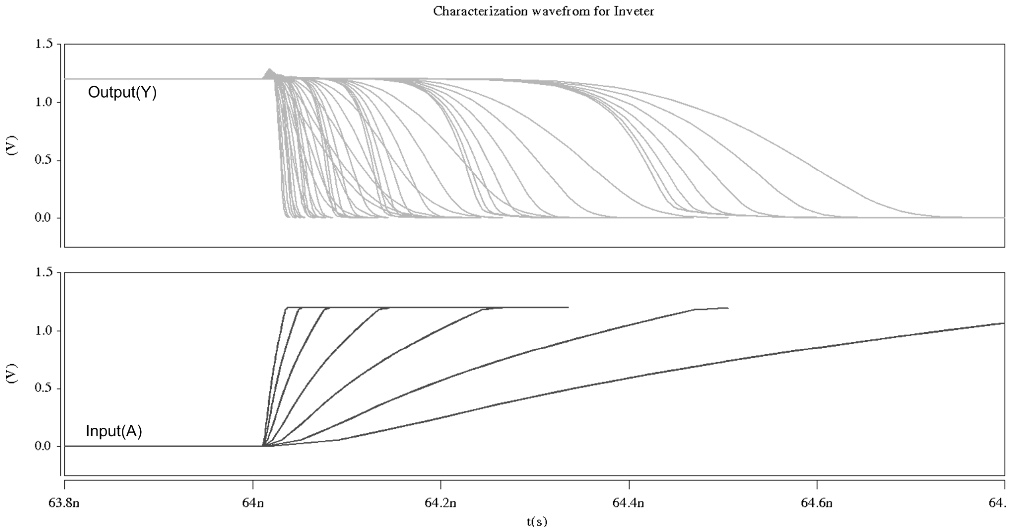


Figure 4.1: Input transition vs output load waveform for characterizing an inverter

load variation information. A .opt file of the standard cell that defines the output to input relation is also given as input.

The tool then simulates the netlist with various input transition (called arcs) and output load and creates a 7 × 7 table that has the delay information. An example of the standard cell inverter characterization waveform for the output rise transition is shown in Figure 4.1. The inverter is simulated for various input transition and output waveform for each input transition versus various load combination as shown in waveform. The same steps are repeated for output fall transitions.

For the sequential cells, like flop-flops and latches, the setup and hold times for each combination is also computed and tabulated. Clocks, clear and preset information for these sequential cells are given in the .opt file in addition to the input and output information.

4.3 ***Characterizing the complex dynamic gates***

Most of the current characterization tool only characterizes the static CMOS gates or sequential cells. They do not support characterization of dynamic cells. This is because the input to output relation for the dynamic gates is not straight forward. The output can only rise by the precharge signal and can fall by input signal. This makes the tool unable to automatically understand the functionality and hence cannot characterize them in a normal flow. Also, the dynamic gates often have high fan-in *i.e.*, a 16 input dynamic gate is common. The tool generates arcs for each combination of inputs that can toggle the output. This makes for a very large number of simulations. For the example of inverter shown in Figure 4.1 with only one input, the tool does 98 ($=2\times7^2$) simulations for both rising and falling output transitions. Hence for the case of dynamic gates with 16 input the number of simulations would be about 6 million ($=2^{16}\times98$). This makes the characterization of the dynamic gates impractical.

So in order to characterize the dynamic gates, we simplified the problem in two steps:

i) Exploited the truth table format of the input output relation

ii) Reduced the number of input to output relations by regularity

These characterization tools support truth tables for defining the output to input relation. The output rise is defined by the precharge signal and the fall is defined by the input. However, when neither the precharge signal nor the input is driving the output, it is held by the keeper and stays at the previous stage. When both input and precharge is driving the output, the output is defined by the input

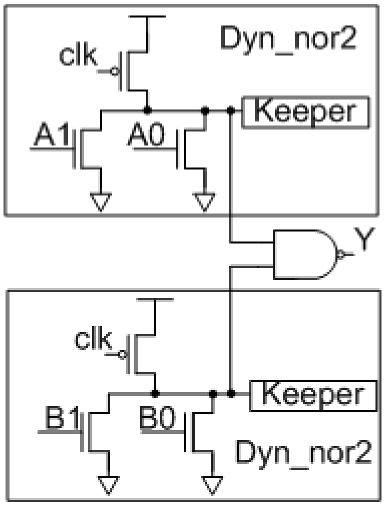


Figure 4.2: Dynamic 4-input OR gate

combination assuming the pullup network is weaker than the pulldown network. In reality, such situation never arises and hence are exploited to reduce the table.

The dynamic cells can be treated as sequential circuits rather than as combinational blocks. This is because of the keeper at the output of the dynamic gates that act as a storage node. However the problem with this is that when the static timing analysis tools, *i.e.*, Prime Time, Encounters a sequential block on its path, it stops going further and gives the timing information upto that block assuming it as a clock boundary. So the only way to properly treat the dynamic blocks into the timing analysis tool is to treat them as combinational block with prechn as preset signal and all the pull down inputs as combinational input. This allows the Prime Time tool to analyze the path correctly and compute the critical timing path.

The following example explains how this can be achieved. The tools used are NCX and Prime Time. For simplicity a 4 input dynamic OR gate is presented. The circuit is shown in the Figure 4.2 implemented using two 2-input dynamic

```
statetable ("a0 a1 clk", " IOUT") {
  table :      " L  L  L   : - :  H, \
                 L  L  H  : - :  N, \
                  -  H  L  : - :  L, \
                 H  -   L  : - :  L, \
                  -  H  H : - :  L, \
                  H  -  H : - :  L ";
}
```

Figure 4.4: 2-input dynamic NOR gate state-table

```
statetable (“A0 A1 A2 A3 clk", " IOUT") {
       table : " L   L   L    L    L  : - :  L, \
                 L   L   L    L   H  : - :  N, \
                 H  -    -     -    H  : - :  H, \
                 H  -    -     -    L   : - :  H, \
                 -   H  -     -    H  : - :  H, \
                 -   H  -     -    L   : - :  H, \
                 -    - H     -    H  : -  :  H, \
                 -    - H     -     L  : - :  H, \
                 -    - -     H    H  : - :  H ";
                 -    - -     H    L  : - :  H ";
}
```

Figure 4.3: 4-input dynamic NOR gate state-table

NOR gates followed by a static CMOS NAND gate. Characterization of this 4-input OR gates was carried out in two ways and their correctness is checked using circuit simulation.

In one method, the whole 4-input OR gate is taken as a single stage gate and the characterization of the block is carried out using the NCX. It is done using the truth-table shown in Figure 4.3. One note-worthy point in the statetable is that, although it’s a contention for dynamic block when clk (precharge) is low and any of the input is driven high, the output is driven low implying the pull-down always win in such cases. This is done in order to make the state-table complete. The characterization tool then simulates the dynamic block and generates the timing characteristic file that can be used by the static timing tool.

In the second approach, a divide and conquer method is applied. The 2-input dynamic NOR gate and the static NAND gate is first characterized using the Synopsys NCX tool independently and then the .lib file generated is fed to Prime

```
statetable (“A0 A1 A2 A3 clk", " IOUT") {
      table : " L   L   L    L    L  : - :  L, \
                L   L   L    L   H  : - :  N, \
                H  -    -    -    H  : - :  H, \
                H  -    -    -    L   : - :  H, \
        }
```

Figure 4.5: Simplified 4-input dynamic NOR gate truth-table

Time with a verilog netlist file explaining the connection between the different instances. Prime Tool can then generate an overall timing information file (.lib) from this information using the *extract model* command.

While characterizing the dynamic blocks, the setup timing needs to be ignored. This can be done be setting the following two switches in NCX

*NCX_create_arcs: <cell_name> * * setup_rising ignore all;*

*NCX_create_arcs: <cell_name> * * setup_falling ignore all;*

This setting of the switches are required, otherwise the timing analysis tool will infer this block as a latch and only the timing related to the clock input will be propagated across. Hence the overall timing file will not be generated otherwise.

The two lib files for the 4-input dynamic OR gates were analyzed and the results were compared. They are found to be accurate within 10% accuracy. Thus, the characterization of the large complex gates with many inputs can be achieved by a divide-and-conquer method as explained above. The complex blocks are split into smaller blocks and individual characterization is carried out to generate the timing information for these smaller blocks. Then the verilog netlist file with the connection details of these smaller blocks fed to the Prime Time will give the overall timing file.

However, sometimes these methods still cannot characterize the high fan-in dynamic gates. The characterization tool takes a long time due to innumerable simulations. The dynamic gates have regularity, i.e. the outputs to each input relation is not complex and are mostly behave similarly. As can be seen in the truth table in Figure 4.3., most of the combinations are don't care indicated by '-'in the truth table. Thus the complex truth table can be simplified as shown in Figure 4.4 where only the output relation is defined for one input (a_0_). When the characterization tool is executed with this truth table, it gives the timing information of the output with relation to this particular input. But since the input relation defined is only one, the number of arcs generated will be two and only 98 simulations would be executed for generating the characterizing file. This step can be repeated for each of the other input one at a time and four timing file will be generated for this particular case. These files need to be merged into one file manually to obtain the complete timing file for the 4-input dynamic OR gate. Thus the complete timing file is obtained by 392 (=4×98) simulation rather than 1568 (2^4×98) simulations. The number of simulations increases exponentially with the input in the otherwise full truth-table case. The same strategy can be applied for characterizing the 16-input dynamic gates and so on.

Having explained the various methodologies to achieve near custom performance using the various ASIC tools, the following chapter explains how this was achieved. Various designs that includes, internet protocol content addressable memory (IPCAM), issue logic, translation look-aside buffer (TLB) and register file (RF) are carried out using the structured design flow.

Performance comparison against both the pure custom and fully soft-core designs are carried out and the analysis is presented.

CHAPTER 5. **ESTIMATING DESIGN PERFORMANCES**

In order to evaluate the strength of the structured design flow, fully custom designs that were already developed in the research lab are designed using this new methodology and the performances are compared against each other. Two of the designs that were already developed here in the research lab namely internet protocol content addressable memories and issue logic were analyzed and complete design was carried out using the structured design flow.

5.1 ***Internet protocol content addressable memories (IPCAM)***

This work was the part of my Master of Science thesis work. A brief description of the design followed by performance evaluation is presented here.

5.1.1 ***Router Functionality***

The large and constantly increasing internet traffic volume depends on continuously increasing router performance, measured not only by throughput, but also by cost and power dissipation. Packets in the IP domain are routed on a next-hop basis, i.e., the router sends an incoming packet to the next hop only—the packet reaches its final destination in multiple hops. For this the router has to perform address lookup, buffering, scheduling and finally, sending the packet to the next hop address through the appropriate port. The key router performance bottleneck is the associative determination of the appropriate outgoing link for each packet. The mask bits are set based on the length of the valid address. The addresses bits for which mask bits are '1' are valid and the rest of the address is ignored. Hence mask bits are set to '1' in at least part of the MSB and to '0' for the unused IP address LSB bits. The destination address of an incoming packet is

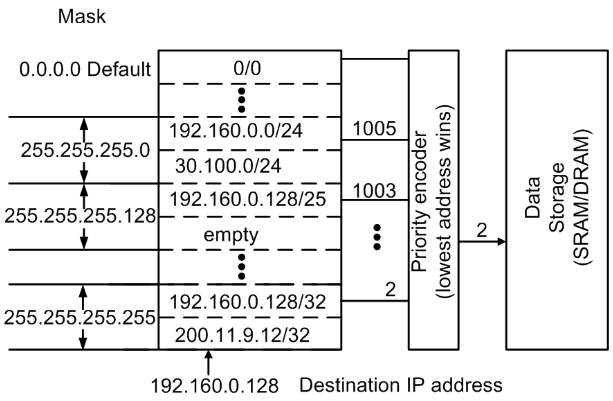


Figure 5.1 Basic IP router logical structure. The values after the slashes indicate the mask values. Final hops are in external memory as shown.

compared with all the current prefixes in the routing table and the next hop associated with the longest matching prefix is determined for the packet. This forwarding information is stored in the router forwarding table as shown in Figure 5.1. Since many addresses may have matching prefixes, to facilitate determination of the longest one, the addresses are conventionally sorted by their prefix length, which is determined by the number of bits masked. Once the forwarding information is computed, the router switches the packet from the incoming link to the appropriate outgoing link. In the case of Figure 5.1, the matching address nearest the bottom is the next hop for the incoming packet.

The proposed static IPCAM (SIPCAM) circuits follow the basic multi-sized match approach of the dynamic IPCAM (D-IPCAM [Maurya10]) but use only static CMOS gates and incorporate micro-architectural improvements. Since high fan-in dynamic flip-flops circuits are prone to failure due to leakage and noise [Chaudhary06] and are not amenable to auto place and route (APR), the proposed design is entirely static to reduce power and make it amenable to automation. The circuits are simplified by assuming that the smallest usable

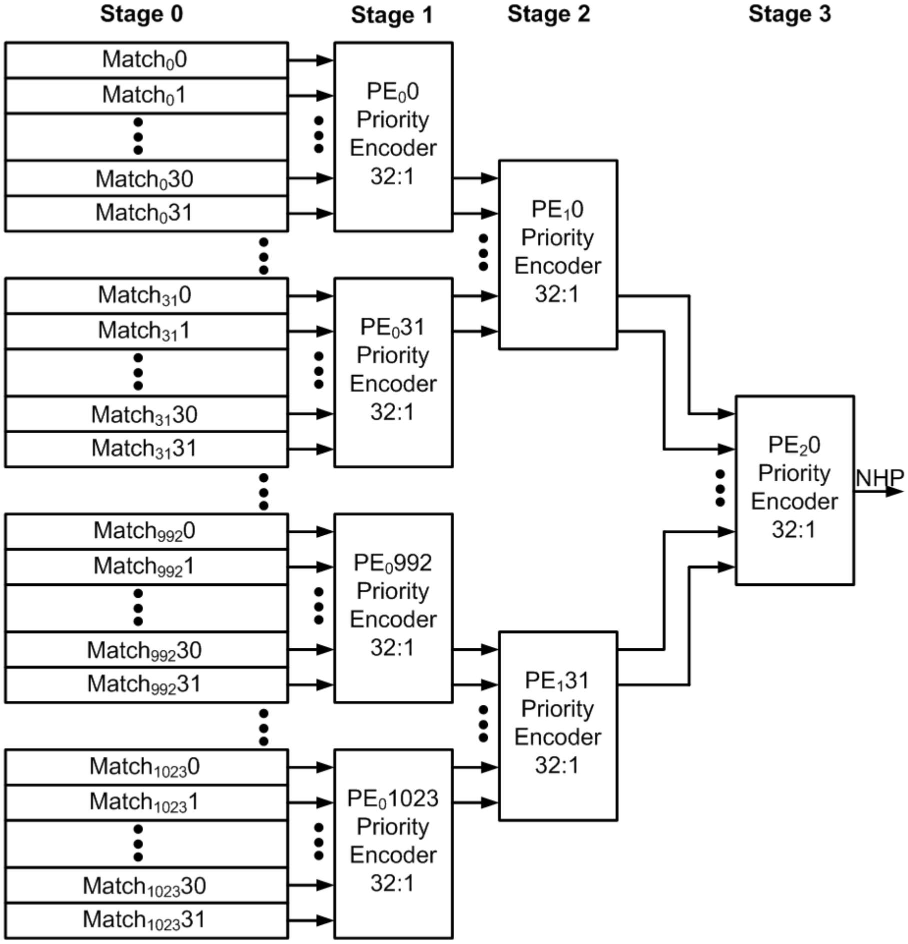


Figure 5.2 Architectural details of proposed routing table circuit. The match block is composed of static IPCAM circuits, followed by the priority encoder. The next hop address pointer NHP, corresponding to the location of the best match, is output

match is 8-bits, while the previous D-IPCAM allowed matches down to one bit. Moreover, removal of the pre-charge clock phase allows the proposed circuit to operate at higher clock frequencies than the dynamic version. This section describes the SIPCAM in detail using a 32K entry routing table. A faster priority encoder design, which completes the longest prefix match look up by determining the longest of multiple matches, is also shown.

5.1.2 ***Overall Architecture***

The complete architecture for the proposed next hop routing table is shown in Figure 5.2. The table shown has 32K entries of the IP address with the corresponding NHP associated with each addresses. This is sufficient to hold the BGP table [BGP]. Using the input address, each entry in the proposed SIPCAM match block directly computes the longest matching contiguous bits from a single stored address and mask word. The match block operates on all the 32K entries in parallel. Each entry determines the number of most significant bits (MSBs) of the stored address that match the input destination address. The result is passed to the PE to determine the best LPM address. Each PE block operates on 32 match outputs as shown. The output from each PE block is then passed to the next set of PE blocks until the final matching block with longest prefix match is obtained. Since the proposed SIPCAM is implemented using static CMOS gates, it can be pipelined into multiple stages as shown in Figure 5.2, where four stages of pipeline are needed for implementing a 32K entry routing table. When pipelined, a lookup can be carried out every clock and the NHP is determined after a latency of four clocks.

5.1.3 ***Match block***

For every unique address (mask prefix pattern), there can be a maximum of one entry with a given prefix length matching external search key in the entire routing table. Thus, for a word width of 32 bits, there can be a maximum of 24 entries (no entries under 8-bits) matching external search key data in the entire routing table, which is exploited in the SIPCAM circuit to determine the LPM. The circuit architecture for one row required to implement IPv4 is shown in Figure 5.3. The complete operation is divided into four blocks, viz., A, B, C and D in order to lower the fan-out of each static CMOS gate.

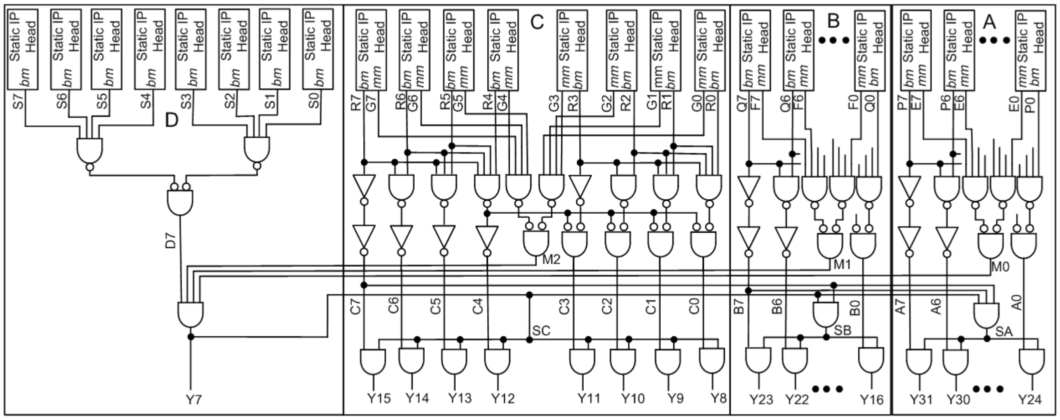


Figure 5.3 Proposed SIPCAM circuit for one entry. One row for implementing IPv4 is shown

The first most significant 8-bit block, i.e. block D, shown in Figure 5.3 is different from the others where only the MSB D7 is computed. Signal D7 is asserted only when all of the first eight bits of the address match; otherwise the entry mismatches. Each of these blocks (A-D) operates on eight bits of the address in parallel to generate the output. The signals P(7-0) through S(7-0) represent the single bit matches (bm) of the stored IP address with the destination IP address. They are converted to eight bit thermometric codes A(7-0) through C(7-0), respectively using NAND, inverter and NOR gates as shown. The signals E(7-0) through G(7-0) represent the match bit (mm) signals. The mm signals from the CAM head circuit are ANDed together to generate M(2-0), one from each group. Thus each of the three groups (A-C) generate nine output signals, eight thermometric match information (A-C)(7-0) and one mask bit information M(2-0) respectively. The select signals S(A-C) are generated either from mask bits M(2-0) or from the MSB outputs (B-C)7 from each of the blocks depending upon the block position. Thus for the C-group, the select signal SC is generated from the match information M(2-0) and D7, however for group B, the select SB is controlled by MSB from C-group and the match signals. The final match information for the 32-bit block is obtained by ANDing the select signals S(A-C) with the corresponding block outputs as shown.

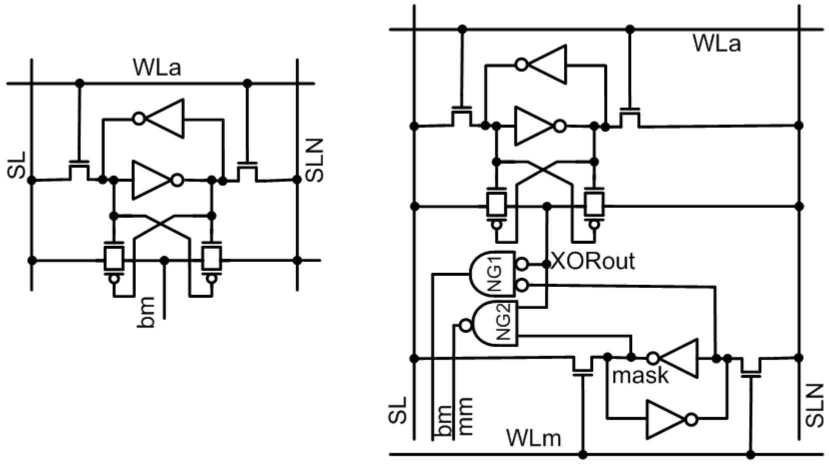
The matching circuitry in each group (A-C) of eight columns is built in two 4-bit groups, similar to carry lookahead circuits. Within the lower group, the most significant bit column's output controls the match output of the other less significant bits as shown in Figure 5.3. Thus, in each group the leftmost column, e.g., R7 in the C group, controls any of the eight match lines. The second, e.g., R6 controls 7 match lines. The rightmost controls only the match line corresponding to a single match on the rightmost bit, e.g., R0. Thus, the number of consecutive matching bits (the prefix) is encoded by C7-C0.

When an entire group matches, i.e., all 8 bits in the group match the incoming address, that group signals out on one of the signals (A-D)7 match that this has occurred. For example, if 8-bit groups C and D fully match, but there is a mismatch at the 5th bit in group B, i.e., B(7-0)= 0001_1111. The select signals S(C-B) are high and select signal SA is driven low. Since group D and C fully match, output Y(15-7) are all high. Output Y(23-16) driven by B(7-0) is equal to = 0001_1111. Since SA is low, outputs Y(31-24) are also low. Thus, the output generated by the match block (Y(31-0) = 0000_0000_0001_1111_1111_1111_1111_1111) is a thermometer code. In a thermometer code, whenever a bit is asserted high, all the lower order bits are also high. In the above example, signals Y(20-0) are all high and rests are low indicating a 21-bit match. The NHP address is the one with the maximum match output.

5.1.3.1 ***Head Circuit***

The head circuit provides the CAM function by XORing the search lines with the stored address under the control of the mask bit. Two designs of the CAM head circuit is shown; Figure 5.4(a) shows the CAM head circuit for the first 8-bit address D-block and Figure 5.4(b) for the lower 24-bits blocks (A-C). Since all the mask bits for the MSB D-block will always be set, the NOR/NAND gate and mask storage is redundant for this block. The XOR network in the CAM head cell in each column determines if the stored address matches the incoming address bit. Two input NOR and NAND gates (NG1 and NG2) are used for the LSB blocks (A-C) to generate XORout and mask signals bm and mm. When mask is set, the bm signals signify if the incoming bit matches the stored bit. The mm signal is asserted low if mask is set but there is no match. This signal for each bit is ANDed together and controls the final match output as shown in Figure 5.3. The mm signal ensures that there is only one match length for a specific mask set. It drives all the outputs to zero for which there is a mismatch and the corresponding mask bit is set. This simplifies the search for the longest prefix match as there is one (exclusive) match for each row in the table.

In operation, one of the clock controlled differential search lines for each of the 32 columns is asserted high in every clock cycle, starting the match operation. The XOR network in the CAM head cell in each column determines if the stored address matches the incoming address bit for that column. If it does not,



a)CAM head circuit for first 8-bits b)CAM head circuit for lower 24-bits

Figure 5.4 The CAM head circuitry driving each column of the Static IPCAM row

the signal XORout (see Figure 5.4) which drives one of the NOR and NAND gate inputs, is asserted high, thereby asserting the column bit-wise match signal bm low. The CAM head cells are written and read by placing the data to be stored on the differential search/bit lines SL and SLN (see Figure 5.4) and asserting WLa to write the address storage or WLm to write the mask storage. However when WLa and WLm are zero, the BL/BLn act as the search lines. The mask storage controls the match value, i.e., whenever the mask is zero, the match will be low for that cell.

Since the matching network (see Figure 5.3) is implemented using static CMOS gates, the match lines are not pre-charged and the search lines are not clock-ANDed. This allows a complete clock phase for the match operation. The differential search lines are generated using the rising edge triggered, differential output flip-flop circuit shown in Figure 5.5. Each flip-flop drives 32 match blocks. The master latch is a conventional active low latch circuit. However, the slave latch is fully differential to achieve equal rise and fall times on the differential

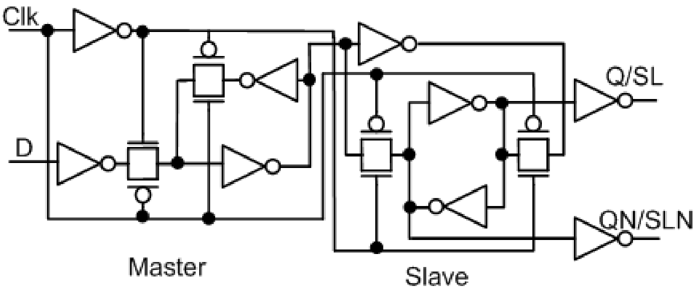
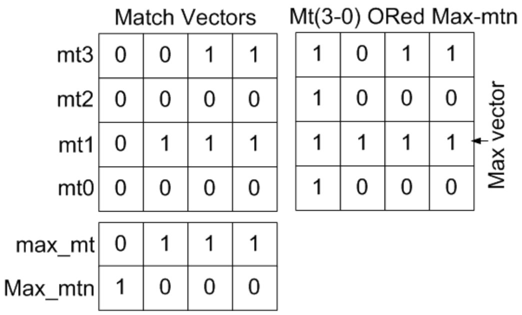


Figure 5.5 The flip-flop circuitry showing both master and slave circuit driving each search line of the Static IPCAM row.

search lines during the high phase of the clock. This ensures similar timings through all match blocks.

5.1.3.2 ***Masking***

To describe the SIPCAM operations we assume that the mask bits in the CAM head circuit are all high. Referring to Figure 5.3, prefixes are stored with the least significant bit to the right and most significant bit to the left. The mask bits are thus set in order, depending on the address length, from left to right. For instance, if the prefix length is 24 bits, then group A is left out of the prefix search for that entry. The longest prefix that can match is 24 bits, so these match lines are permanently asserted low by the mask bits in the CAM head cell. These match lines never get asserted as one of the input of the head circuit NOR gate is high (NG1 in Figure 5.4(b)). No power is dissipated in masked columns, since the bm signal is fixed. Returning to the 24-bit address length example where all bits in group A are masked, the maximum output code is then Y(31-7) = 0000_0000_1111_1111_1111_1111_1.



	Match Vectors			
mt3	0	0	1	1
mt2	0	0	0	0
mt1	0	1	1	1
mt0	0	0	0	0
max_mt	0	1	1	1
Max_mtn	1	0	0	0

Mt(3-0) ORed Max-mtn			
1	0	1	1
1	0	0	0
1	1	1	1
1	0	0	0

Figure 5.6 Priority encoder example

5.1.4 ***Priority Encoder***

The thermometric output from the match block, combined with the fact that there is a unique matching output from each row simplifies the design of the priority encoder. This can be explained considering an example as shown in Figure 5.6, which for simplicity considers four rows of matching information from match blocks each consisting of 4-bits. There can be multiple rows with zero matching but otherwise the length of the matches in each row is unique (otherwise the shorter entries would match).

Since all the match vectors mt(3-0) are thermometric, the maximum match vector max_mt can be obtained by ORing all the bits column wise. The max_mt is same as mt1 in case of example in Figure 5.6. In order to detect the maximum match row, the second column must be output. With the known fact that the max_mt is either equal to or greater than any match vector mt(3-0), the complex XOR operation can be avoided. By inverting the max_mt, a complement thermometric vector, i.e., vector with all ‘1’ towards MSB and ‘0’ towards LSB, can be generated. When this is ORed with each of the match vectors mt(3-0) only one output with all bits set (the one matching the max_mt) will be asserted or at

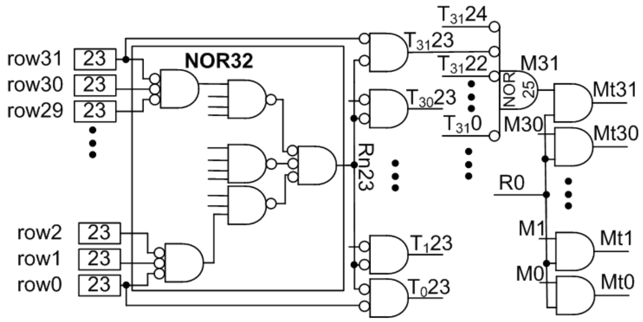


Figure 5.7 Logic implementation of the proposed priority encoder

least one bit is asserted low as shown in Figure 5.6. Thus the matching entry can be obtained by ANDing all these outputs together. However, a special case arises when none of the entries matches, i.e., all entries are zero. This will cause no entry matching selection; i.e., max_mt match all the entries. This can be avoided by ANDing the max_mt(0) with all the output select signals because max_mt(0) signifies at least one bit match.

Figure 5.7 shows the logic implementation of the proposed scheme. The 32 rows of Match (31-0) are the vector output from the match block shown in Figure 5.3. Each of these is a 25-bit thermometric code. The NOR32 gates carries out 32-bit column wise NOR operations to generate the complementary maximum match vector Rn(24-0). The Rn(24-0) is ORed with each of the match vectors to generate T(31-0)(24-0). T31(23) corresponds to row/entry 31 and column 23. The active low T signals are combined in a 25-bit NOR gate to generate the match signals M(31-0). The final match signal Mt(31-0) is generated by ANDing M(31-0) with R0 signal in order to avoid no entry matching selection. The high fan-in NOR32 and NOR25 gates are implemented in three static CMOS stages.

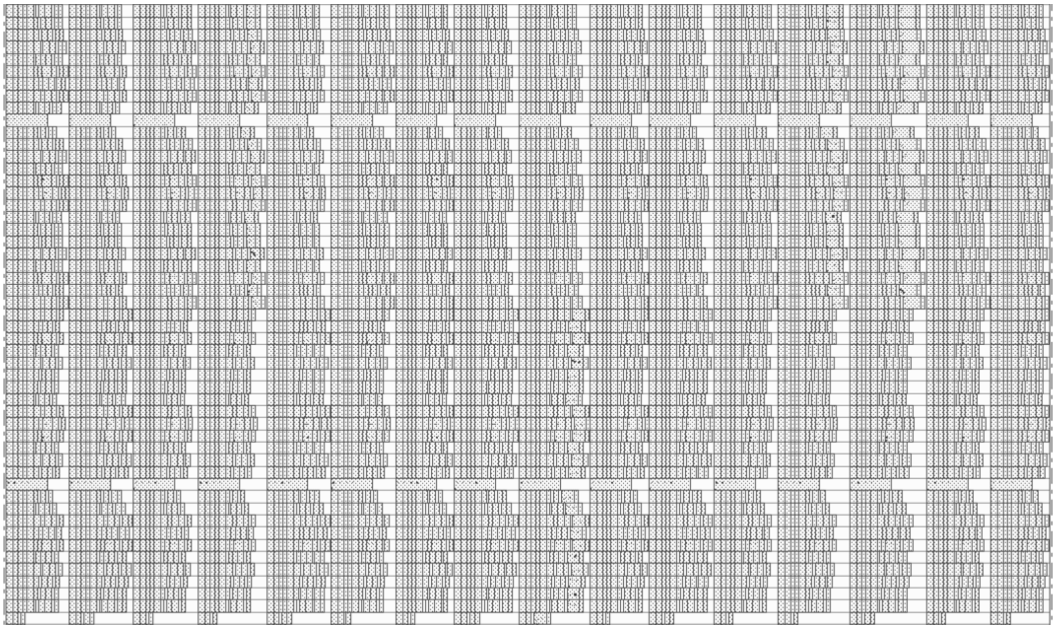


Figure 5.8 Placement of standard cells for 32-entry SIPCAM array including both match block and PE.

5.1.5 ***Layout of design***

The complete SIPCAM is implemented using the standard cells from the Artisan 45nm library. Some custom standard cells, *e.g.*, SRAM and XOR gates, were also added for efficient implementation. All these standard cells were then placed using the spreadsheet which is then read by set of scripts to generate the placement in Encounter floorplan as shown in Figure 5.8. The placement is highly regular. Since all the entries of the SIPCAM are similar, copying the first column multiple times and replacing its instance numbers allows the complete placement.

The layout of the proposed SIPCAM block for both match and priority encoder blocks is shown in Figure 5.9. The 32-rows are split into two sets, one above the other. The PE cells are immersed within the match row/columns in order to minimize wire lengths. The empty spaces in the MSB D-block can be

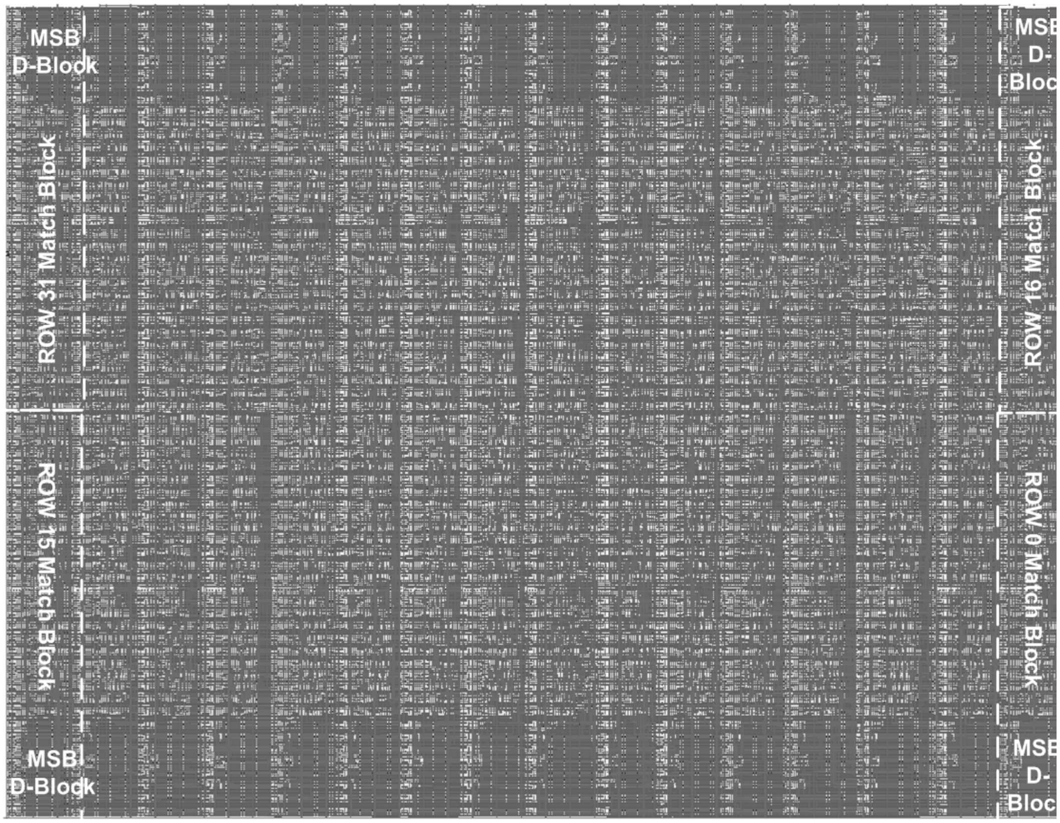


Figure 5.9 Layout of 32-entry SIPCAM array including both match block and PE.

filled with the flip-flop circuitry (shown in Figure 5.5) to drive the 32-columns. The initial design was carried out in IBM 65 nm foundry process. Portability from our initial 65 nm to the 45 nm library was straightforward. Special differential XOR and SRAM cells were added to the available library gates, corresponding to the cells shown in Figure 5.4. The cells were placed using scripts to organize the row and column placement, before using Encounter to wire all connections. This allowed rapid design and the systematic array layout shown.

5.1.6 ***Performance analysis***

The use of the static CMOS gates aids in the use of the ASIC static timing tools for the timing analysis. Timing analysis of the complete design used

Synopsys Primetime. Characterization of the differential XOR and SRAM cells was carried out using Synopsys NCX. The SIPCAM match block has a propagation delay of 145.6 ps at V_{DD} = 1.1 V at -40°C (fast-fast corner) and 364.3 ps at V_{DD} = 0.72 V at 125 °C (slow-slow corner) and hence the design can achieve well over 2 GHz clock rates. Energy dissipation, determined using Ultrasim to simulate the post parasitic extraction netlist is 1.97 fJ/bit/search.

5.2 ***Out of order Issue Logic***

This work was the part of Master of Science thesis work by Siddhesh Mhambrey [Siddhesh10]. A brief description of the design followed by performance evaluation is presented here.

5.2.1 ***Instruction Issue Logic***

Microprocessor instruction streams contain instructions that can potentially execute in parallel. This characteristic, termed instruction level parallelism (ILP) is the foundation of superscalar processing. ILP is determined by the number of true data dependencies and number of branches in relation to other instructions and is limited by the existence of anti-dependencies and associated hazards. Speculation techniques give the processor an ability to look beyond the current point of execution and extract ILP. The process of selecting the highest priority ready instructions requires a buffer to store instructions, a dependency tracking mechanism to generate ready signals (ready indicates that a given instruction inputs will be valid in the next cycle) and a selection mechanism to pick instructions according to a selection policy. These requirements are fulfilled by the instruction issue logic that has wakeup and selection logic blocks

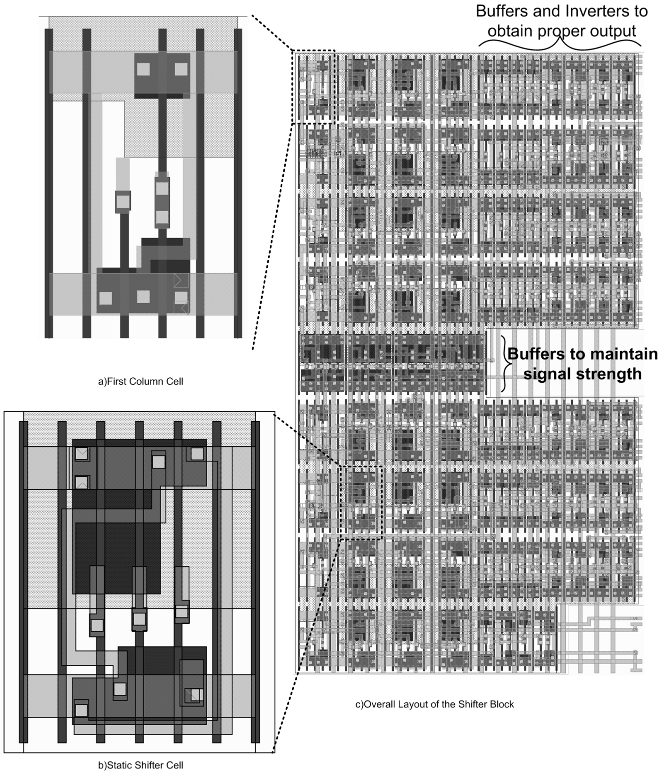

Buffers and Inverters to obtain proper output
Buffers to maintain signal strength
a)First Column Cell
b)Static Shifter Cell
c)Overall Layout of the Shifter Block

Figure 5.10 Layout of the shifter1 structure implemented on the 45 nm foundry process. Details of the gates are shown to the left (a) and (b), while a full entry is shown to the right

as shown in Figure 5.10. The wakeup logic is comprised of a queue that stores the renamed instruction registers, tracks their dependencies and generates ready signals based on the dependencies. The selection logic prioritizes the ready instructions. The update logic then refreshes the queue to accept new instructions in the next cycle, while maintaining the ordering of existing entries and replacing issued entries with new instructions at the clock edge.

5.2.2 ***Layout of the shifter1 structure***

Since shifter block is very regular, layout of the block was automated using the Cadence Encounter APR tool (see Figure 5.10). This design has pass-gate shifter cells rather than the normal static CMOS gates shown in Figure 5.10(a-b). The basic individual cells were laid out in the standard cell library height. The cells were placed in the Encounter floor-plan using the Perl scripts. APR (using Encounter) then connected the cells and routed to the block input and output pins, so the only manual effort was the shifter cell design. Cell layouts, i.e., the first column cell and shifter cell are shown in Figure 5.10(a) and Figure 5.10(b). To maintain the most direct (fastest) internal wiring, the cells are sparse. The complete layout is shown in Figure 5.10(c).

Timing analysis of the shifter block of Figure 5.10 was carried out using Primetime using the post extracted nextlist.. Since this part of the issue logic only uses transmission gate multiplexer, timing analysis for the shifter block was carried out separately. Cell characterization was carried out using Synopsys NCX. The shifter block has a propagation delay of 94 ps at V_{DD} = 1.1V and temperature of 125ºC and 169 ps at V_{DD}=0.81V and temperature of 125ºC.

CHAPTER 6. **FULLY STATIC ARRAY BLOCKS**

Having set the floor for the structured design flow and tested it using two of the already existing design, more sophisticated designs are carried out. Design of the fully static translation look-aside buffer and register file has been carried out and its performance is compared against the fully soft-core and dynamic design.

6.1 ***Translation look-aside buffer (TLB)***

This design was a part of the HERMES microprocessor memory management unit block. The design uses RHBD cell libraries developed by Nathan Hindman in 90nm IBM foundry process. The same design was later carried out using Artisan 45nm libraries.

6.1.1 ***Memory management unit (MMU)***

Most modern computers use virtual addressing and demand paging, i.e., dividing the memory into pages, which are dynamically allocated to different processes, thereby operating as if each process had a complete isolated memory system. The virtual-to-physical address translation is a large lookup table. Translation look-aside buffers (TLBs) are small cache memories that speed up the virtual-to-physical address translation in microprocessors. The TLB thus contains the most recently used virtual-to-physical address mappings and page attributes, making it a key component of most memory management units (MMUs). The most common TLB circuit implementation makes use of a fully associative content addressable memory (CAM) affording high hit rates, the ability to handle multiple page sizes, and high speed.

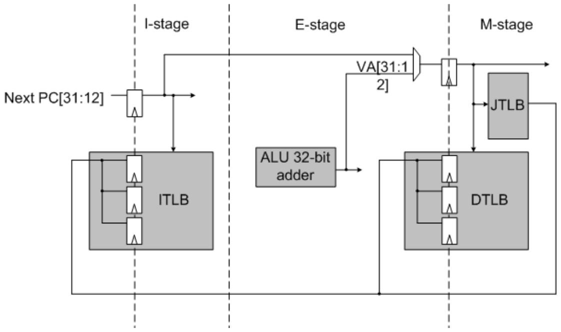


Figure 6.1 Microprocessor pipeline stages with integrated MMU unit.

In the last decade, power consumption has become the primary factor in high performance processor design. Since it is accessed for every instruction and data fetch, the TLB can be costly in terms of power consumption. For a fully associative dynamic TLB, power consumption increases approximately linearly as the number of entries increases [Juan97]. Another important feature of modern MMUs is their support of variable page sizes. Smaller pages are preferred in embedded systems, as the small page sizes allow better utilization of smaller system memories. Measured data from embedded low power processors such as the StrongARM and SH-3 showed that as much as 17% on-chip power is consumed in the TLBs [Juan97] [Kadayif02].

6.1.2 ***Design Specification***

This TLB is intended for a five stage embedded microprocessor. Figure 6.1 shows the MMU in the core pipeline. The MMU consists of instruction and data micro-TLBs and a joint TLB (JTLB). The former provide a first level of caching and hold only instruction or data translations, respectively, with three 4-kB page entries. The second level JTLB holds instruction and data page mappings in 16 variable sized dual-page entries. Instruction translations in mapped memory

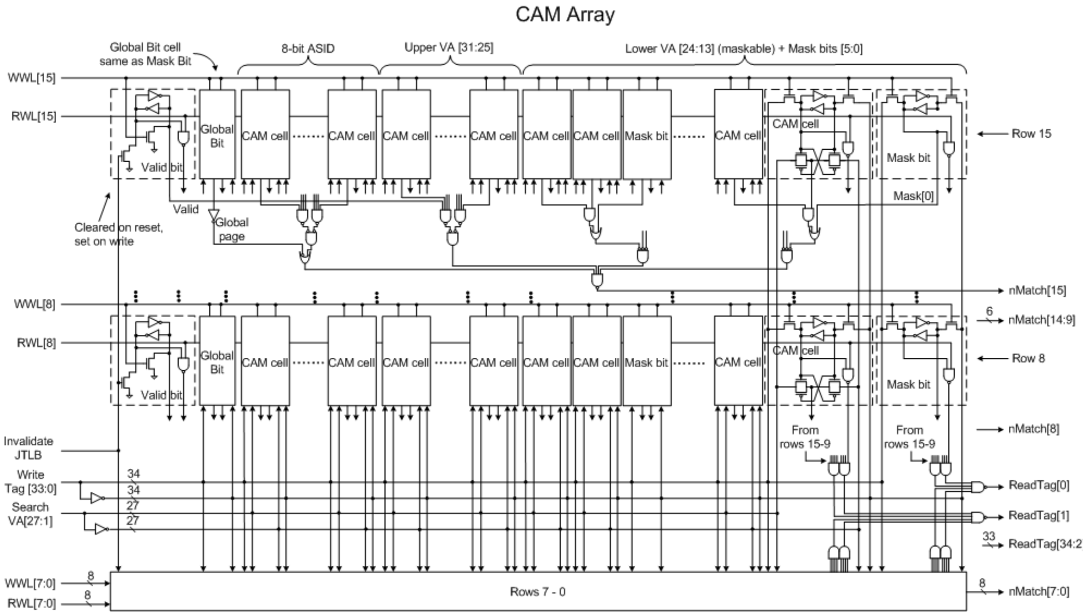


Figure 6.2 JTLB CAM array showing the timing/logic critical path gates.

are first accessed in the micro-instruction TLB (ITLB) while data translations, with higher miss rates, access both the micro-data TLB (DTLB) and JTLB so the latter can translate on a DTLB miss with only a one clock penalty. Microprocessor TLBs can have signifycant power dissipation since the CAM (for lookup operation) and RF (for storing the physical address) structures are traditionally dynamic circuits, with high activity factors. These circuits are generally a critical timing path, so power savings must not be obtained at the expense of increased circuit delay [Clark03].

The JTLB responds to the following events (1) invalidations, (2) writes, (3) reads, (4) probes and (5) lookups. On invalidations, the CAM valid bit for each entry is cleared. The valid bit is written on subsequent write operations. The CAM array is comprised of two halves, with 8 entries in the top half and 8 entries in the bottom half to limit RC wire delay and loading (see Figure 6.2). Each CAM entry

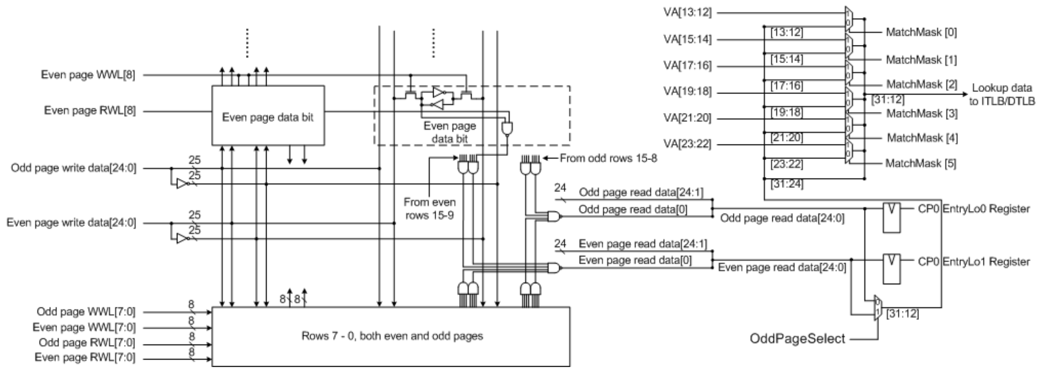


Figure 6.3 Bottom half of the JTLB data array logic.

is 35 bits wide consisting of the valid and global bits, eight address space identifier (ASID) bits, a 19 bit virtual address and 6 mask bits. Of these, only the ASID and VA bits use CAM cells. The mask bits control matching of the 12 lower VA bits. The output other gates to nMT[x] comprise a distributed wide OR gate i.e., the static match lines. Along with the match output, the odd-page select signal is also generated, which controls whether to read the even or odd RF bank as above. The mask bits are also read out from the match block to select the final physical address multiplexers using circuits identical to those in the static RF. Static logic determines which data array entry to read out on a lookup match. For JTLB writes, the CAM is looked up in the M-stage to ensure the entry is not already present. The CAM entry is written in the high phase of the W pipeline stage. A CAM entry is read out statically with the read triggered on the rising edge of the M-stage clock. The search line drivers are placed towards the left side and the read (RWL) and write (WWL) word line drivers are placed mid-way as shown in the Figure 6.2 in order to reduce the capacitance loading and wire length.

The static RF storing the page translations and permission bits is shown in Figure 6.3. It consists of two banks, viz. even and odd, each of 16 rows of SRAM

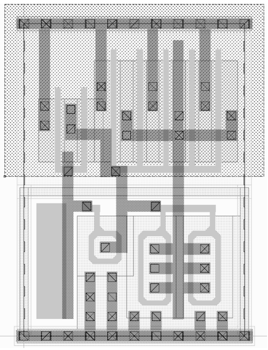


Figure 6.4 RHBD buffer layout showing the annular NMOS and guard rings for PMOS.

array like the match block, only one of which is read from the data array on a CAM match. The even and odd page mappings are interleaved across rows. Each entry is 25 bits comprised of a 20-bit physical frame number, a 3-bit cacheability attribute, a dirty and a valid bit. The static NAND gate within each cell controls the cell read, which is combined by distributed gates comprising a large OR. Similar to the match block, the bit line drivers are placed towards the left and the read and write word lines are mid-way as shown in the Figure 6.3. A data array entry is written in the high phase of the W-stage clock and is read out statically on the rising edge of the M–stage clock.

6.1.3 ***Operation***

Writes are conventional as evident in the RF cell detail in Figure 6.3. To search, address bits 31 to 13 are driven by complimentary output flip-flops to the search lines SL and SLn that drive the XOR cells in the CAM. The compare outputs are ANDed using the static CMOS gates to generate the entry match signal (e.g., nMT[15] in Figure 6.2) taking the mask bits and global bits into consideration. For a matching entry, the corresponding odd-page select signal whether to read the odd bank or even bank in the static register file. The output from each SRAM cell in the RF cell is ORed together using NAND-NAND

Table 6.2 TLB Energy per operation (pJ)

	NOP	Read	Write	Lookup
DC	0.64	7.2	10.0	13.8
RHBD Structured	0.55	3.16	7.67	4.89

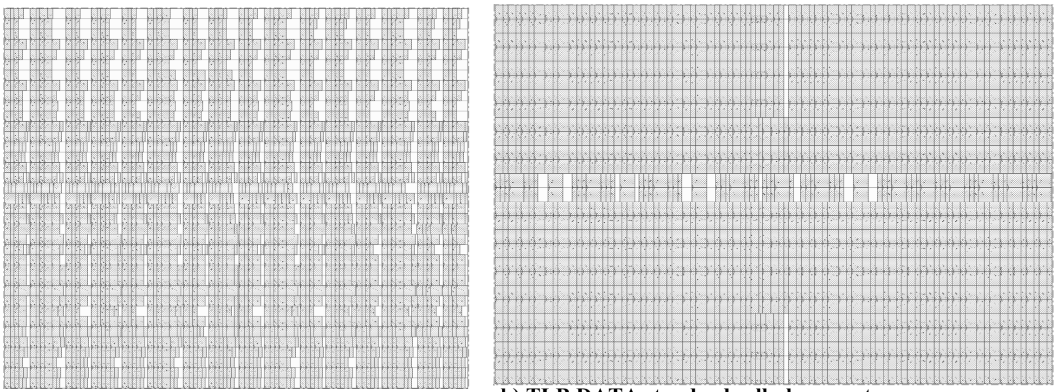
Table 6.1 TLB delay comparisons

Process	Time(ps) (no parasitics)		Time(ps)(extracted and 90% density)	
	Commercial	RHBD	Design Compiler	RHBD Structured
FF	567.3	314.7	707.0	381.0
SS	2554.0	1695.8	3273.6	1988.4
Typical	1184.0	748.3	1367.9	879.6

operation to generate the physical address which are muxed using the mask bits to generate the final output.

6.1.4 ***Radiation hardened TLB design***

An ultralow power radiation hardened TLB design implemented on 90 nm bulk CMOS is presented here. The design is implemented using a 90 nm RHBD cell library that has been proven to have no significant leakage increase in accelerated TID testing to above 2 Mrad(Si). The library uses N well guard rings to mitigate single event latchup (SEL) as well. Figure 6.4 shows the RHBD layout of a buffer circuit depicting the annular NMOS gates and N well guard rings. The SEE hardening is through micro-architecture, which uses DMR pipelines with dual TLB copies. SET and SEU induced errors are determined by implicitly comparing the outputs produced by the TLBs. The TLB power is limited to the bare minimum through an all static logic design that has no power dissipation for accesses to the same page, as validated through benchmarks run on the design with extracted parasitics. By careful (but design-effort efficient) design, we



a) TLB CAM standard cell placement

b) TLB DATA standard cell placement

Figure 6.5 Standard cell placement details for CAM and Data part of TLB

achieve full hardening with better than standard commercial design performance and power dissipation.

Since much of the processor is protected from SEE induced errors by DMR, the lowest possible power must be achieved to mitigate the power impact of doubling of the logic. Lowering the activity factor is critical, as is physical design to minimize the area and parasitic capacitances (and resistances) of each block. In order to achieve low power dissipation and maximum SEE robustness, the JTLB is a completely static design (see Figure 6.2 and Figure 6.3). Static logic primarily helps the former, but does avoid CAM dynamic match and RF bitline discharges, which are not recoverable when dynamic.

6.1.4.1 ***TLB Radiation Hardening***

The MMU is DMR with physically separated instances to prevent an ionizing radiation particle from simultaneously corrupting both. MMU error checking occurs indirectly through micro-architecture error checking and correction mechanisms that exist in other places within the processor; the two MMU instances are not directly compared, so there is no direct checking cost in the MMU. The point where an error is flagged depends on the nature of the error. There are three places where a mismatch between the two MMU instances may be detected: (1) at the crossover stage between the dual redundant pipeline and the TMR fill buffers used to retrieve instructions or data from the external memory system; (2) at the crossover stage between the DMR pipeline and the TMR stage where exception handling takes place; (3) at the DMR cache comparator outputs. Once an SEE induced mismatch error is detected, an exception handler is invoked that, among other things, invalidates all TLBs. Subsequent MMU lookups result in an access to a page table residing in memory where uncorrupted translation and attribute information are retrieved and again cached in the MMU.

6.1.4.2 ***Results and Analysis***

The proposed TLB is designed using a proven TID hard RHBD standard cell library, as well as a commercial (unhardened) standard cell library available for the same process for comparison. The baseline design uses engineered placement and automated place and route (APR) using the Perl scripts to achieve near custom circuit power and performance. The baseline design is built using all the standard cells with few more custom standard cells like SRAM and XOR that were added to the library for efficient implementation in terms of both delay and power. The placements of these cells were first carried out in the spreadsheet and then Encounter compatible placement file is generated using the Perl script. The placements of these standard cells are shown in Figure 6.5. With high level of regularity and consistent placement, placement density of 88% and 97% were achieved for the TLB CAM and Data design. The placement looks evenly spread across the floorplan. Since each of the 16-rows of both CAM and Data are similar, coping the first row and repeating it for 16-times makes the complete placement done, thereby building the complete design quickly still achieving the performance goal.

Comparisons are also made between the standard soft-core, full synthesis and APR approach that would be available to a hardened processor implemented using commercially available register transfer level (RTL) and the hard-core APR assisted design here. The timing results using static timing analysis reveal that the RHBD library implementation improves delay more than 35% across process corners as compared to the unhardened commercial standard cell implementation (see Table 6.1). Power dissipation comparison to a (soft-core) synthesized and APR TLB design, with both fully routed, and using extracted parasitics is shown in Table 6.2. The soft-core JTLB is 190% the size of our RHBD design (using the same RHBD cells), the layout of which is shown in Figure 6.6. Static timing analysis shows considerable improvement in delay. Table 6.1 shows the details of the actual delay values for the various JTLB implementations.

The TLB power dissipation is estimated using circuit simulations of the actual layout with extracted interconnect parasitic RCs running the operations shown in Table 6.2 at V_{DD} = 1.2 V and 100 MHz. Benchmarks were run through an architectural simulator to determine the actual JTLB operation frequencies. The data show that 96% of TLB operations are lookups, 3% are writes (to add an entry for a miss during the TLB lookup) and the rest is for reading. Based on these statistics and the values in the Table 6.2, the structured design energy

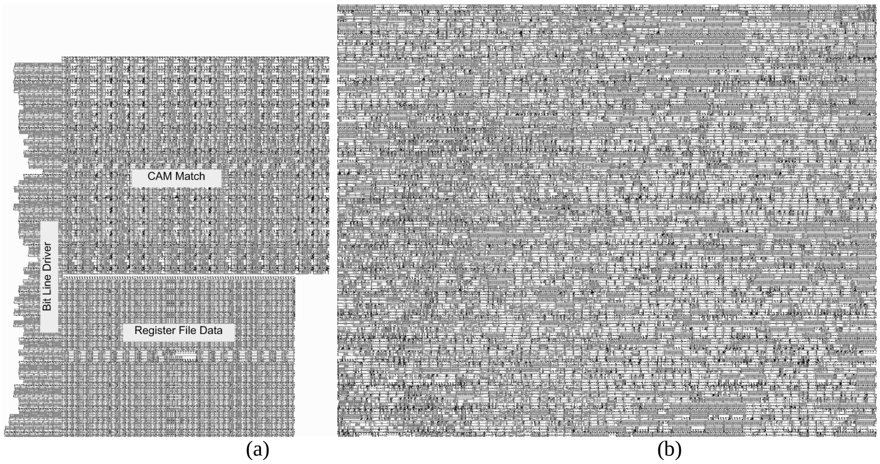


Figure 6.6 Single instance TLB layouts using both structured flow layout (engineer controlled placement in the arrays) and using standard DC/APR approach. With engineered placement, less cells are used and the delay and area allow DMR TLB in the same footprint as a conventionally done soft-core.

dissipation is 37% lower than that of the static synthesized design. Figure 6.7 shows the actual signal timings for the structured TLB design. The power simulations used Cadence Ultrasim, stimulating the JTLB with the vectors as determined by the architectural simulator. The energy delay product (EDP) of our structured design is 4.37 times less than the synthesized design using the post extracted layout simulations.

*6.1.5 **TLB design using commercial library***

The above TLB design was also implemented using the commercial library cells instead of the radiation hardened cells. We used Artisan 45nm standard cell library for the implementation. Comparison of the standard dynamic TLB to static designs such as those produced by the synthesis of soft-cores is presented here. Additionally, we compare to a structured approach, which has directed cell selection and placement using the Perl script, but retain the static circuits of the synthesis approach. The structured approach for static TLB requires

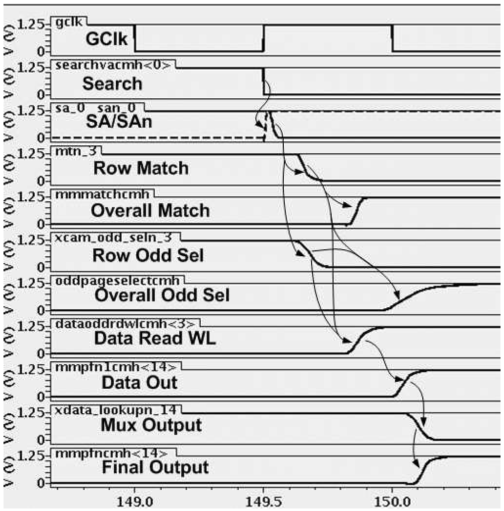


Figure 6.7 TLB simulation waveform showing assertion of match-line and data read word-line

considerably less design effort than is needed for the dynamic design, and is superior to both the dynamic and synthesized TLBs in terms of power, area and energy-delay product (EDP).

6.1.5.1 ***Dynamic TLB design***

The reference TLB tag is implemented as a conventional dynamic fully associative CAM with a dynamic RF data memory containing the PAs and permissions [Haigh04] as shown in Figure 6.8.

6.1.5.1.1 ***Micro-architecture details***

In a dynamic TLB (see Figure 6.8) the critical timing path starts with the tag access, i.e., the CAM compare, during the first clock phase. The resulting match signal must then be latched, since the RF access in the next clock phase will occur while the CAM pre-charges. The hard clock edges preclude time borrowing, and sufficient timing margin must be included to allow the CAM operation to complete. In the CAM, the clocked search line drivers controlling SL

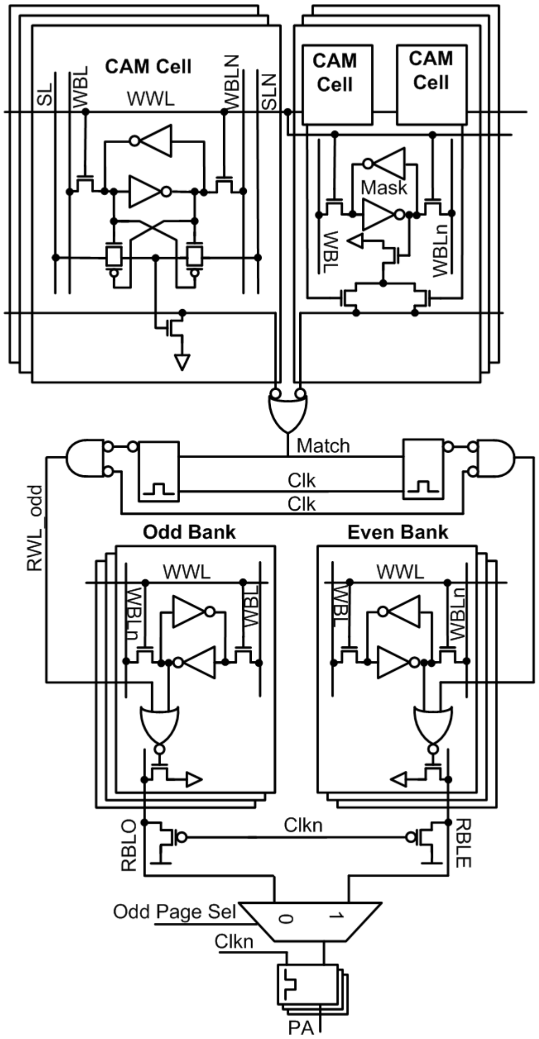


Figure 6.8 The fully associative dynamic TLB reference circuits

and SLN act as the D1 domino stage, and the CAM cells are D2 (footless) domino gates driving the high fan-in NOR match lines. The CAM power is primarily due to the clocked search lines and match line discharges. On a hit, N-1 of N match lines discharge.

On a CAM tag match the data memory RF is read in the second clock phase to output the PA and permissions. The RF consists of two banks, providing 32 pages with 16 entries, organized in pairs, selected by the LSB of the CAM match output (see Table 6.3). The Odd-Page Sel signal is dynamically generated

Table 6.3 Mask bits corresponding to Page size and Odd Page select signals

Mask(5:0)	Page Size(KB)	Odd Page Select
000000	4	Addr(12)
000001	16	Addr(14)
000011	64	Addr(16)
000111	256	Addr(18)
001111	1024	Addr(20)
011111	4096	Addr(22)
111111	16834	Addr(24)

along with the match operation. Each RF entry is 25 bits, comprised of a 20-bit physical frame number, a 3-bit cacheability attribute, a dirty and a valid bit. In the domino RF the read word-line (WL) acts as a clocked domino (D1) stage and the cell read bit-line (BL) acts as a footless domino (D2) stage. Again, the high RF BL activity factor leads to high dynamic power dissipation. The dynamic circuit activity factor propagates downstream, raising that of the output bus and all attached circuitry whether dynamic or static, e.g., the cache address bus. The RF read speed is dominated by the required read bit line (labeled RBLE and RBLO) discharge time. The RBLE/O capacitance is reduced in this design by the NOR gate driving a single pull-down transistor. The RF output drives a second (dynamic to static conversion) latch in the TLB path.

6.1.5.1.2 ***Multiple page size support***

Multiple page size support is easily accommodated in a fully associative design by including in each entry mask bits that indicate the page size of that particular entry. During a TLB CAM search, these bits mask off portions of the address that should not participate in the comparison. The page sizes supported in the reference design are shown in the Table 6.3. The output of a single page size

Table 6.4 Performance analysis of the TLB designs

Design	Delay(ps)			Energy per operation (pJ)				Area(μm^2)
	Fast-fast T=-40C V_{DD}=1.1	Typical T=27C V_{DD}=1.0	Slow-slow T=125C V_{DD}=0.72	Read	Write	Lookup	Overall	
Reference Dynamic TLB		326.5		26.61	27.86	37.14	36.76	9217.53
Design Compiler	476.4	528.4	1472.8	13.21	13.86	14	13.99	8552.11
RTL Compiler	491.5	541.8	1486.6	11.25	12	11.21	11.24	8778.4
Structured Static	318.2	348.9	945.3	8.4	9.43	8.88	8.89	7746.03

bit is used to directly mask the compare of the appropriate set of CAM cells (see Figure 6.8). RF cells store the decoded page size mask bits that disable the CAM match line pull-down circuits in the corresponding CAM cells.

6.1.5.2 ***Structured static low power TLB design***

Lowering the activity factor is a key advantage for a fully static TLB design. If the page is constant, none of the TLB circuits transition. However static circuits have greater capacitances than their dynamic counterparts, particularly for wide OR functions. Structured physical design minimizes the area and parasitic capacitances (and resistances) of each block, particularly if the combining gates, i.e., those in the high fan-in match and bit lines, are placed cleverly to mitigate some of the performance impact of using static circuits where dynamic circuits are the norm.

The details of the static TLB design are shown in the Figure 6.2 and Figure 6.3. Both the match block implemented using CAM blocks and the resister file consisting of the conventional SRAM cells are shown respectively. The complete implementation is similar to the radiation hardened one explained in section 6.1.4, except for this design uses commercial Artisan 45nm standard cell library instead of the RHBD library.

6.1.5.3 ***Performance analysis of the TLB design using commercial library***

As mentioned, all the designs were completely implemented using the same foundry 45 nm SOI process; fairness is ensured by meeting the same 90% cell density for all cases. The foundry library was used for cells where possible, i.e., most non-domino cells. CAM and differential write SRAM (latch) cells were added to the static library. Characterization of the differential XOR and SRAM cells was carried out using Synopsys NCX. Comparison in terms of delay, power and EDP use the post extraction netlist to include the parasitics, which vary dramatically by design style. The results comprise Table 6.4. The conventional TLB design uses domino circuits. In addition to affecting the power, dynamic circuits require considerably greater design effort as they must be simulated for proper keeper sizing (write-ability) noise and charge sharing issues over process, temperature and voltage corners. Increasing leakage also poses significant challenges to high-fan-in nodes such as the tag CAM match lines and RF RBLs.

6.1.5.3.1 ***Physical design and Area Comparison***

The structured TLB is built similar to the RHBD TLB explained previously. The standard cells used were from the Artisan 45nm cell library. The SRAM and XOR custom cells were then added to the library. The same spreadsheet used in the RHBD TLB is used again by replacing the equivalent standard cells in Artisan 45nm library. The detailed placement of the standard cells is shown in Figure 6.9. The figure shows both CAM and Data of the TLB

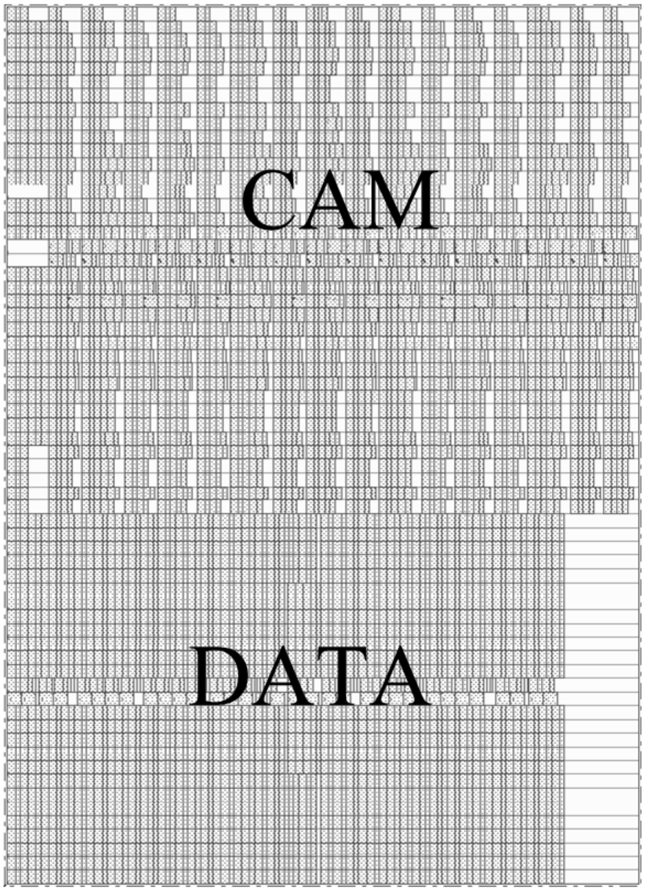


Figure 6.9 Standard cell placement for the 45-nm TLB showing both CAM and DATA

together. The design was built very fast and the migration from RHBD 90nm library to Artisan 45nm was just replacement of the corresponding standard cells.

The layout of all the four approaches is shown in the Figure 6.10. For the structured design and dynamic designs, the area is the sum of the match block and RF block and not the overall rectangle taken together. The layout has been carried out for these designs using the Perl script. The structured approach utilizes 9.5% and 11.8% less area than the DC and RC synthesized designs, respectively. Surprisingly, the area gain is 19% over the dynamic design. Although dynamic CAMs and RFs employ fewer gates and minimize PMOS transistors, the required phase latches and clocked driving circuits eliminate the advantage.

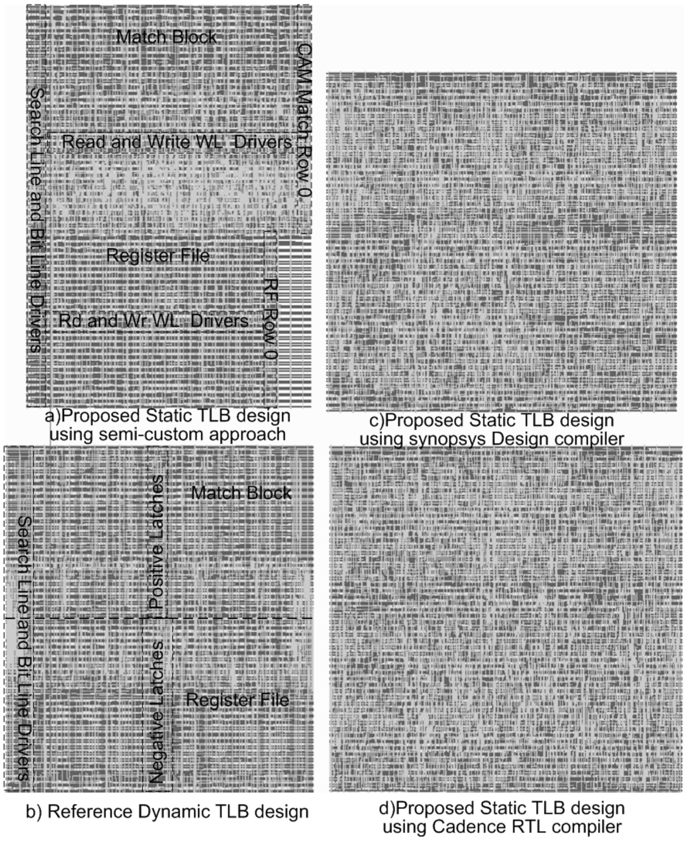

Match Block
CAM Match Row 0
Search Line and Bit Line Drivers
Read and Write WL Drivers
Register File
RF Row 0
Rd and Wr WL Drivers
a)Proposed Static TLB design using semi-custom approach
c)Proposed Static TLB design using synopsys Design compiler
Search Line and Bit Line Drivers
Positive Latches
Match Block
Negative Latches
Register File
b) Reference Dynamic TLB design
d)Proposed Static TLB design using Cadence RTL compiler

Figure 6.10 Layout of the stuctured and the dynamic TLB design. a) shows the layout for the proposed TLB design using the structured approach. b) shows the layout of the reference TLB design. c) and d) shows the synthesis layout using the design and rtl compiler respectively of the proposed TLB design.

Using only static CMOS gates makes the design more scalable and portable, while structured layout preserves these advantages and improves the speed and power dissipation. To ensure the best delay possible, the netlist was hand optimized (as it was for the dynamic version). The 16-rows are physically split into two sets, one above the other for both match and the RF block. The odd and the even rows in the RF file are interleaved to minimize wire lengths to the final multiplexing operation. The cells were placed using Perl scripts to organize the row and column placement, before using Encounter to wire all connections. This allowed rapid design, but with optimized placement and minimal physical design effort.

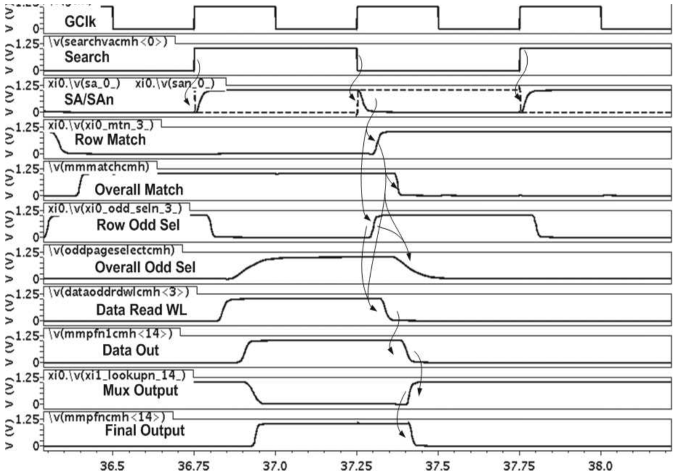


Figure 6.11 Simulated waveforms for the structured static design at 2GHz.

Figure 6.10 shows all the layouts with the key structures labeled for the structured static TLB and the dynamic reference design. The dynamic TLB has similar layout as the structured layout with the phase latches labeled. The RF output latches make its size similar to that of the match block. Figure 6.10 also shows the layout of the synthesized static TLB using Design Compiler (Figure 6.10c) and RTL Compiler (Figure 6.10d) which achieve similar sizes and delays.

6.1.5.3.2 ***Delay***

Timing analysis of the static designs was carried out with Primetime, with the critical paths verified using Ultrasim. Timing analysis was carried out at the Fast-Fast (-40°C and V_{DD} = 1.1 V), typical (27°C and V_{DD} = 1.0 V) and Slow-Slow (125 °C and V_{DD} = 0.72 V) corners (see Table 6.4). The structured structured approach provides 33% and 35% delay improvements at all corners compared to synthesized TLBs. The simulated waveforms with a 2 GHz clock for the structured TLB is shown in Figure 6.11 with the critical path signal transitions marked.

Since the reference TLB design uses dynamic gates, the delay and power measurements were determined using circuit simulation only (Ultrasim). The match block takes 171 ps to generate the odd/even read world line select and the register file takes 155 ps to generate the final output at typical corner. This includes the intermediate and final latch setup times, but no clock skew. Even neglecting clock skew (important given the hard clock edges, i.e., the dynamic circuits cannot time-borrow) the gain in the overall delay as compared to the structured design is only 6%. Since the design was carried out using the conventional 45 nm libraries where the gates are not skewed for lower delays, the gain could have been higher at the cost of some noise margin. Nonetheless, this result is surprising.

6.1.5.3.3 ***Power Dissipation***

In order to measure the energy dissipation, memory traces were collected from the SPEC benchmarks for several programs. These gave the percentage of time the TLB is engaged and the operation performed e.g., lookup, write or read. The data show that 96% of the JTLB operations are lookups, 3% are writes (to add an entry for a miss during the TLB lookup) and the remainder reads. For power comparisons, the post extracted netlists were simulated in Ultrasim at V_{DD} = 1V at a 100 MHz clock frequency. The energy results are tabulated in Table 6.4. The overall energy per cycle of the structured design is 36.5% and 21% lower than that of the static synthesized designs. The dynamic TLB design dissipates 4.5× more energy than the structured static design, due to its high activity factor.

6.2 ***Register File (RF)***

A static Register file is implemented using the techniques described in Chapter 3. Comparison is carried out against the dynamic register file built for the HERMES RHBD processor built using in IBM 90nm foundry process. The same design is also synthesized and performance analysis against the structured register file is also presented.

6.2.1 ***Multiported RF***

Multiported register files (RF) are performance-critical processor components with single cycle read/write latency and high throughput requirement providing buffered communication of values between instructions. Also in applications like digital signal processing for in-place algorithms to perform the fast Fourier transform, the RF are the key components. For these applications, the speed and power are the major concerns for the design.

Increasing leakage currents combined with reduced noise margins are seriously degrading the robustness of the dynamic circuits. Keepers are used to maintain signal level of pre-charged nodes. Size of these keepers can significantly impact the performance of the dynamic gates. Conventional multi-ported RF are implemented in dynamic logic and exploits the popular 6T CMOS memory cell by adding several pass transistors to provide additional ports. Unfortunately, cell stability can be compromised when multiple ports access the same cell, as several pass transistors can each draw read currents from the memory cell disrupting the cross-coupled latch in the memory cell. Thus the pass transistors must be made smaller to prevent cell disrupt on read: for two simultaneous read ports, the size of the pass transistors must be half compared to that of 6T memory cell. These half-size pass transistors will increase the read access time by about 2×. The silicon area of a multiported memory, built using conventional approaches, grows quadratically in the number of ports due to wordline and bitline wiring [Tremblay95]. Reducing the number of physical access ports in a memory cell can thus lead to significant area and power savings as well as latency improvement.

6.2.2 ***Multiported RF design***

The register file design presented here is intended for an embedded five pipeline stage microprocessor implemented primarily in DMR logic. The data path results checking are carried out at the write back (WB) stage. Both the circuit architecture and the error checking and recovery are explained in this section.

6.2.2.1 ***Microarchitecture details***

The complete design is carried out on the IBM foundry 9SF 90 nm bulk CMOS process. The complete architecture of the proposed register file implementation is shown in Figure 6.12. The core of the design consists of dual redundant register file blocks A and B, each consisting of 32 rows of 4-byte registers. Two parity bits are added for each byte making each register 40-bit wide. Each of these blocks consists of 3R1W ports. Of the three read ports, only two of the read ports viz. Rs and Rt are used for instruction operand read. The third read port Rs/Rt is used for the error correction mechanism as shown in the Figure 6.12.

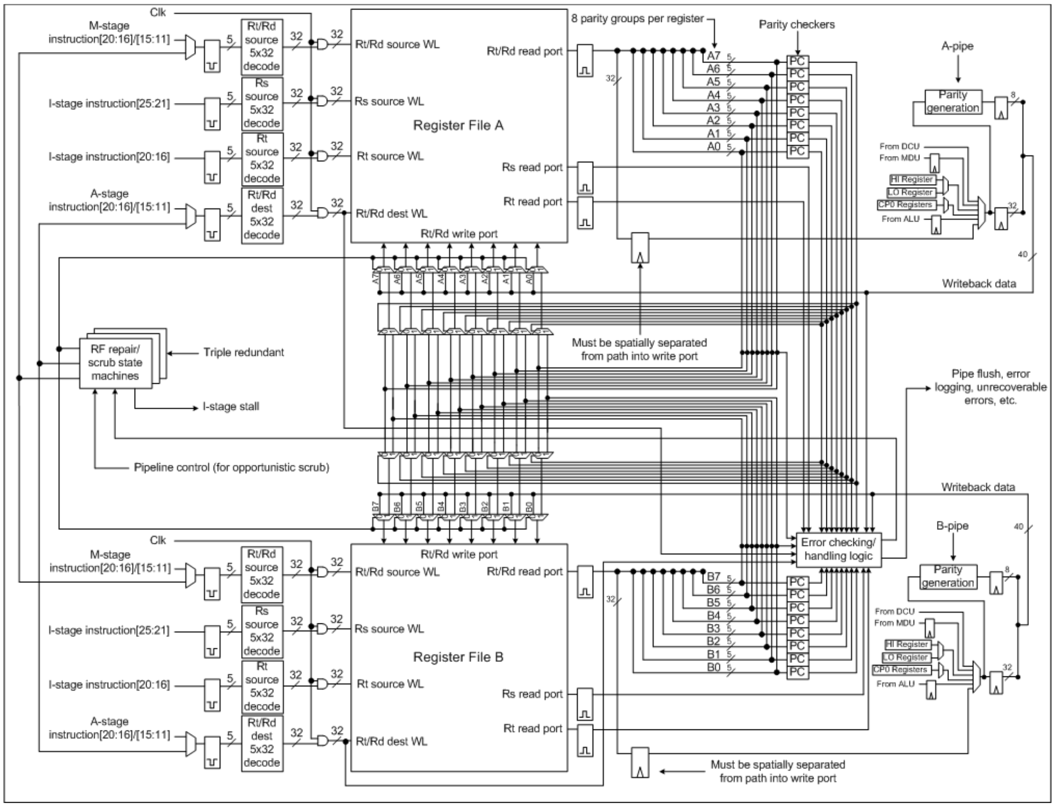


Figure 6.12 Schematic of the complete register file implementation. The dual instance of the core register file block and the error correction architecture is shown.

The detail floorplan of the register file blocks is shown in Figure 6.13. The RF cell design differs from that of a conventional dynamic RF in two respects. First, the write requires simultaneous assertion of two write word lines (WWLs)—this mitigates the possibility of inadvertent writes to RF locations. Secondly, a standard load-store CPU would have two read ports and one write port, supporting the ALU operands and results, respectively. This design adds a third (Rt/Rd) read port, used to read a copy of the destination register contents, about to be written in the next cycle WB stage. Thus, the overwritten RF data is put back if the store to the RF is cancelled due to a detected data path error. Since

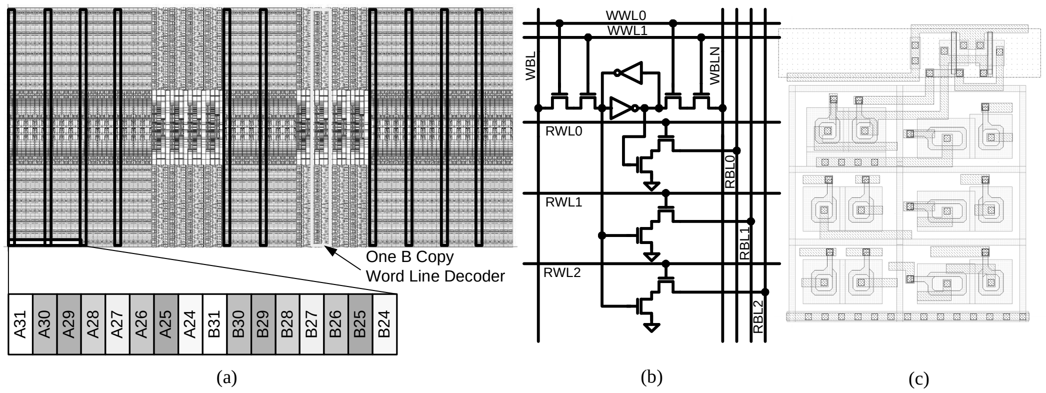


Figure 6.13 32 entry 40-bit DMR RF layout with parity group interleaving (color coded) and one parity group's bits outlined to show critical node separation (a); RF cell schematic showing the unconventional dual WWL connections (b) and cell layout through metal 1 (c). The latter is non-rectangular with the adjacent cell storage node PMOS load transistors sharing the same well. The A and B decoders are separated as well.

DMR does not indicate which copy is in error, the original data is returned to the RF and then the instruction with non-matching results is restarted.

6.2.3 ***Performance Analysis***

The performance analysis of the proposed register file design is carried out against the implemented using the static designs that used standard cells. These standard cells were developed in IBM 90 nm foundry process for radiation hardening by design (RHBD). For comparisons, only the core of the register file blocks A and B shown in Figure 6.12 is implemented. Fairness is ensured by meeting the same 90% cell density for all cases. The foundry library (RHBD standard cells) was used for cells where possible, i.e., most non-domino cells. Differential write SRAM (latch) cells were added to the static library. Characterization of the differential SRAM cells was carried out using Synopsys NCX. Comparison in terms of delay, power and EDP use the post extraction netlist to include the parasitics, which vary dramatically by design style. The results comprise Table 6.5 and Table 6.6.

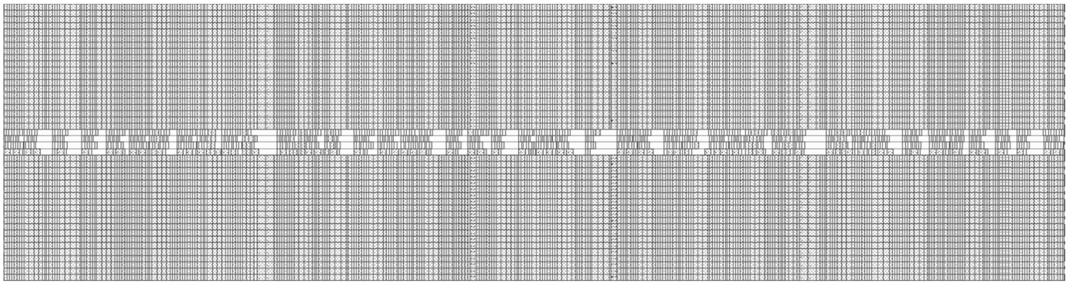


Figure 6.14 Standard cell placement for the dual modular register file

6.2.3.1 *Physical design and Area Comparison*

The design for the structured RF file was carried out by first placing all the standard cells in the spreadsheet. The complete design is built using the RHBD standard cells with added custom SRAM cells to make the implementation more efficient. The Perl script then generates the Encounter compatible placement files. The standard cell placement is shown in Figure 6.14. The regularity of the floorplan allowed to achieve 98% placement density and also with less effort as all the rows are similar to one another. Successive rows can be built from the first row just by replacing the instance number.

The layout of all the four approaches is shown in the Figure 6.15. In the structured approach, the standard cells are placed in the floorplan using the PERL script and then the Encounter routes the complete design using the ARR tool. However for the synthesized design, the Synopsys Design Compiler and the Cadence RTL compiler reads the behavioral model of the register file and generates the layout. The struc

tured register file utilizes 25.1% and 20.3% less area than the DC and RC synthesized designs, respectively. However, compared to the dynamic design, the structured requires 19.8% more area. This is reasonable because dynamic RFs

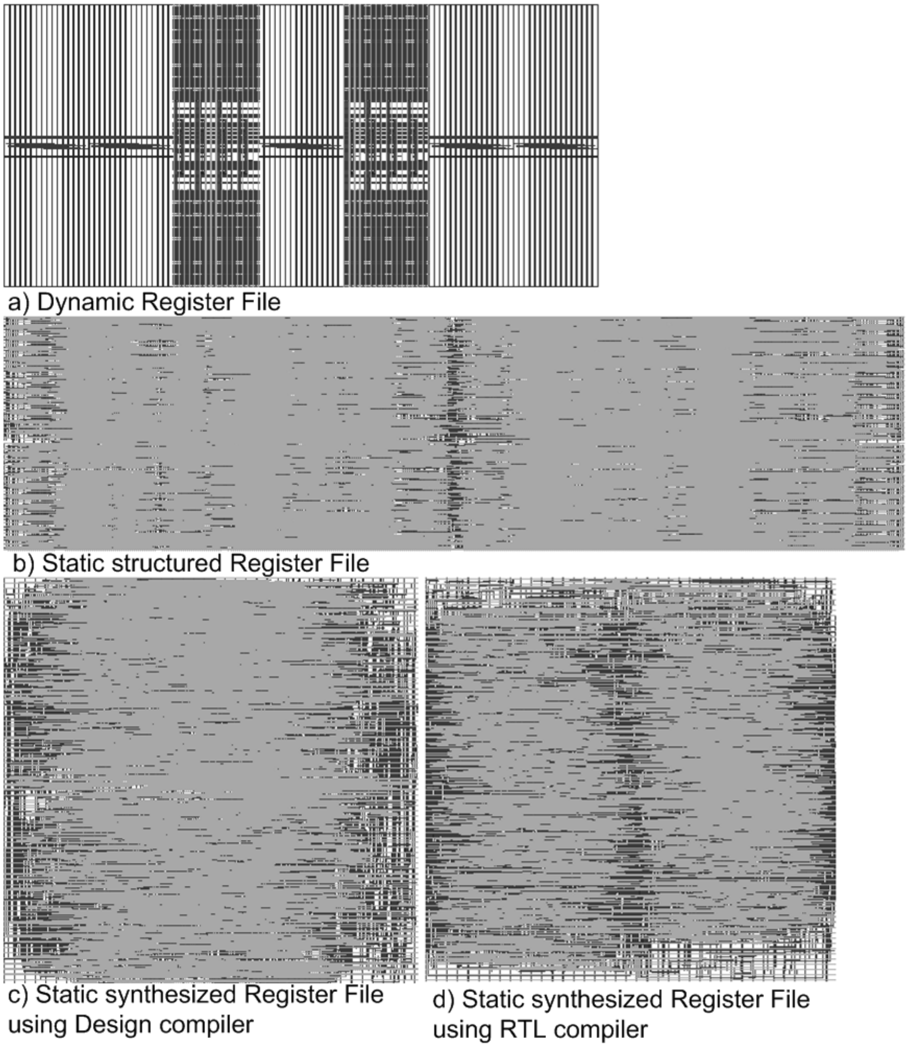

a) Dynamic Register File
b) Static structured Register File
c) Static synthesized Register File using Design compiler
d) Static synthesized Register File using RTL compiler

Figure 6.15 Layout of the dynamic and static RF design. a) Shows the layout for the proposed dynamic RF design b) shows the layout of the static RF using the structured approach c) and d) shows the synthesis layout using the design and rtl compiler respectively of the proposed RF design

employ fewer gates and minimize PMOS transistors and hence can be compacted better.

Using only static CMOS gates makes the design more scalable and portable, while structured layout preserves these advantages and improves the speed and power dissipation. To ensure the best delay possible, the netlist was hand optimized (as it was for the dynamic version). The 32-rows are physically split into two sets, one above the other for structured static RF block. The A and B

Table 6.6 Energy comparisons for various RF design

	1W	1R	2R	3R	1W1R	1W2R	1W3R	Average
Dynamic RF	47.34	40.66	48.67	54.86	54.39	62.87	69.07	53.98
Structured Static RF	12.27	6.52	11.92	22.41	16.62	21.97	32.05	17.68
Design Compiler	16.76	10.69	15.60	27.48	25.04	31.57	38.15	23.61
RTL Compiler	19.46	15.32	20.84	30.41	34.50	44.34	34.17	28.43

Table 6.5 Delay and Area comparisons for various RF design

Design	Delay(ns)			Area(μm^2)
	Fast-fast (T=-40C, V_{DD}=1.1)	Typical (T=27C, V_{DD}=1.0)	Slow-slow (T=125C, V_{DD}=0.72)	
Dynamic RF		0.3902		120700.32
Structured Static RF	0.4150	0.9664	2.1211	144620.85
Design Compiler	1.0942	2.8838	7.0897	193010.62
RTL Compiler	0.5623	1.0372	2.4002	181417.28

rows in the RF file are interleaved to minimize wire lengths to the final multiplexing operation. The cells were placed using scripts to organize the row and column placement, before using Encounter to wire all connections. This allowed rapid design, but with optimized placement and minimal physical design effort.

Figure 6.13(a) shows all the layouts with the key structures labeled for the dynamic design. The structured RF has similar layout as the dynamic layout. Figure 6.15 also shows the layout of the synthesized static TLB using Design Compiler (Figure 6.15c) and RTL Compiler (Figure 6.15d) which achieve similar sizes and delays.

6.2.3.2 ***Delay***

Timing analysis of the static designs were carried out using Primetime, with the critical paths verified using Ultrasim. Timing analysis was carried out at the Fast-Fast (-40°C and V_{DD} = 1.32 V), typical (27°C and V_{DD} = 1.2 V) and Slow-Slow (125 °C and V_{DD} = 0.72 V) corners (see Table 6.5). Since the dynamic design uses domino gates, the delay measurements were determined using circuit simulation only (Ultrasim). The structured design shows the delay improvement of 66.5% and 7% at all corners compared to synthesized RFs. Compared to the dynamic design; the delay is 2.47× more. Although this delay improvement is tremendous, the gain in the operating frequency is lessened by the fact that domino logic requires half phase of the clock for pre-charge and rest for the evaluation. Nonetheless, the gain in the operating frequency is 19% as against the structured approach.

6.2.3.3 ***Energy Dissipation***

In order to measure the energy dissipation, different stimulus were driven to all the designs to measure the energy dissipation for different scenarios like reading through only one port (1R) or reading and writing (1R1W) etc as mentioned in Table 6.6. Since different benchmarks will engage the RFs differently, the energy dissipation will differ. For energy comparisons, the post extracted netlists were simulated in Ultrasim at V_{DD} = 1.2 V at a 100 MHz clock frequency. The energy results are tabulated in Table 6.6. Due to the high activity factor of the dynamic design, the energy dissipation is very high. It dissipates 3× more energy on an average as compared to the structured static approach. Compared to the synthesized designs, the energy dissipation of the structured is 1.3× and 1.6× less for DC and RTL designs.

CHAPTER 7. **CACHE DESIGN: COMPLEX CIRCUIT COMPRISING OF BOTH STATIC AND DYNAMIC LOGIC**

The initial design of the proposed cache architecture was carried out by Xiaoyin Yao as part of his PhD program. The design was carried out entirely using dynamic logic to achieve high speed operation. But the dynamic design was very difficult to port and characterize for use in the overall design. Additionally, the design changes were exceedingly difficult. So the same architecture was implemented, replacing many of the dynamic circuits with static circuits using the standard cells. The flow proposed in this dissertation rectified many of the original circuit's drawbacks.

7.1 ***Introduction***

This chapter shows that large blocks, including large sub-blocks, can also be handled by the proposed methodology and tools. Cache memory comprises the second level memory hierarchy after register file in a modern microprocessor design. It can be either data cache or instruction cache. The principle of locality is important for cache operation. Cache relies on both the spatial locality and the temporal locality of memory accesses. Spatial locality states "that items whose addresses are near one another tend to be referenced close together in time" [Patterson96]. Temporal locality means "that recently accessed items are likely to be accessed in the near future" [Patterson96].

Cache memory is used to store most recently used data and instructions. If the cache has the data or instruction that CPU is going to execute, it is a hit. Otherwise, it is a miss. The miss rate (or hit rate) is an important measure of cache design. It is affected by the cache size and the write policies. There are two options in the write policies: write-through and write-back. The write-through policy requires “the information is written to both the block in the cache and to the block in the lower-level” [Patterson96]. A write-back policy requires the information is written to cache only, *i.e.*,the main memory is only updated when the line is evicted . Cache is updated every time miss happens by replacing an entry with the desired entry from the lower level memory. There are many strategies employed in deciding which block in cache will be replaced. The block can be randomly selected or the Least Recently Used.

There are three categories of cache organization: direct mapped, fully associative and set associative. “If each block has only one place it can appear in the cache” [Patterson96], it is direct mapped. “If a block can be placed anywhere in the cache” [Patterson96], the cache is fully associative. “If a block can be placed in restricted set of places in the cache” [Patterson96], the cache is set associate. Suppose there are n blocks in a set, the cache is n-way set associative.

The cache performance is a major factor that affects the speed of a microprocessor. It is measured in terms of average memory access time [Patterson96]:

Average memory access time = Hit time + Miss rate × Miss penalty

where hit time is the time to hit in a cache, miss rate is the percentage that a miss happens, miss penalty is the additional time needs when a miss happens. Average memory access time can measured either in absolute time or in the number of clock cycles.

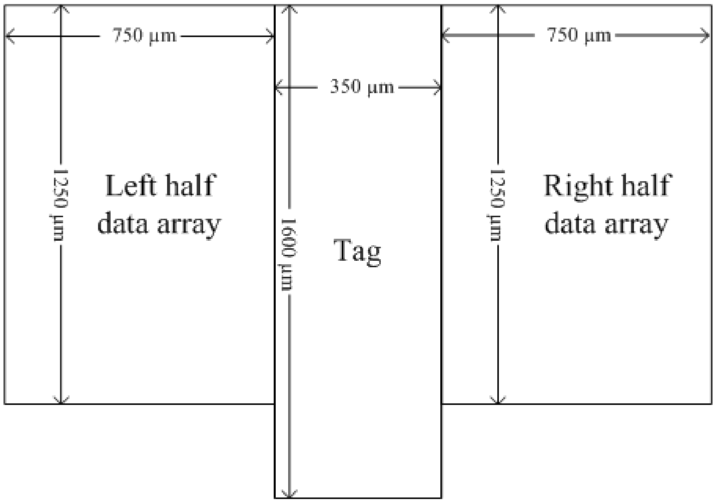


Figure 7.1 Floor plan of cache

The cache is also split into tag and data arrays so they can be addressed independently. In this way, the hit generation and data read out are speed up. A write hit is generally slower than the read hit because the tag must be checked before writing the data. In the design used here, they occur in separate clock cycles.

7.2 ***Cache architecture***

The proposed design of the cache is a part of the targeted HERMES microprocessor. The design is used for both data or instruction caches. Since HERMES is a radiation hardened microprocessor, RHBD techniques are used in the cache design: Annular NMOS transistors and guard rings around NMOS transistors are used to achieve TID hardness. A detect-invalidation-reload scheme is used to achieve SEE harness.

The proposed cache design is a 16 KB cache and 4 way-associative, using write-though and no-write allocate policies. For every eight data bits, there is one parity bit. The cache has 1024 cache lines, with 16 bytes in each cache line. The cache is virtually indexed and physically tagged. A cache line is defined as the smallest division of a cache memory for which there is a distinct tag [Handy98]. In this cache, a line consists of four words (with four bytes in each word) that have the same set address and way. There are 256 cache lines (or 4 KB of data) per way.

The most significant 20 virtual address bits are fed into TLB to generate physical address, which is used for comparison in tag. Other virtual address bits are sent to tag and data array decoders. The way selection signal from the tag select which way of data array generates final cache data out.

The cache supports four operations: lookup, read, write, and global invalidation. In a lookup operation, all 4 ways are read and the tag address is compared with the physical address from translation lookaside buffer (TLB). If there is a match, the selected way is output and a hit signal is generated. Otherwise, a miss signal is generated. This supports an instruction fetch (instruction cache) or a load instruction (data cache). The read operation can read a specified set and way from data array or tag. In the data array, the minimum unit that can be read is a word. In the tag, the tag address and parity bits from a certain set and way can be read out. Meanwhile, the other tag status bits of all other ways are read out at the same time. A write operation writes a specified set and way of data array or tag. The minimum unit written is a byte. In a line fill, a whole line is written. Global cache invalidation invalidates the whole cache. This operation clears the dual redundant valid bits in every tag entry. This occurs when the cache is reset after a power up, reset or a cache SEE-induced error is detected.

7.3 ***Design details***

These cache designs achieve high speed and low power, as well as better SEE hardness, by using simple design approaches. For instance, there is no sense amplifier on the path of bit lines (BL). There are no column multiplexers due to the BL circuit architecture and word line (WL) arrangement. Both data array and tag use dynamic circuits to precharge and read their SRAM arrays. Gated clocks are used as much as possible to reduce active power.

The floor plan of the cache is shown in Figure 7.1. The data array is divided into two halves, with the most significant 18 bits of each byte parity protected word on the left and the least significant 18 bits on the right. The tag is in the middle, minimizing the distance from the way hit signals to the way multiplexers in the data array. This reduces the wire length of the way select signals and thus helps reduce power and speed up the lookup operation.

The cache is composed of tag and data array. They are described in following sections.

7.3.1 ***Data Array***

In left or right half data array, there are 4 words. Each word has 4 banks, which are composed of a top and bottom sub-bank. A sub-bank comprises 32 rows and 72 columns of SRAM cells. In the data array, there are a total 16 KB of data. Since the smallest writable unit is a byte, it must be parity protected. Thus, there is one parity bit for every 8 data bits.

There are 4 banks for one word. Two of them are for way 3 and 2. The other two are for way 1 and 0. In each row, there are two word lines (WL) and two ways. In each WL of each way, there are two bytes. Each byte is composed of eight data bits and one parity bit. Figure 7.2 shows major circuits in the data array. The data array single-cycle fills, since all the words can be written simultaneously by activating the four banks at once.

There are two reasons to include two WLs and two ways in a row. The first is to avoid column multiplexers which are usually used in a cache design. Therefore the BL development path is simplified, which helps the cache achieve high speeds. The second reason is that by interleaving bits belonging to two bytes, two WLs and two ways, the bits belonging to the same parity group (same byte, WL and way) are separated by a distance equal to the width of seven SRAM cells, rather than three. This helps obtain SEE hardness against multi-bit upsets.

The major component circuits in data array includes the WL decoder, SRAM cell, precharge for BL, write control and way multiplexer, as shown in Figure 7.2. During the precharge stage, the BL is pulled up to V_{DD} by a PMOS transistor. When the clock is asserted and there is an operation to this bank, the WL decoder asserts one WL high. One SRAM cell in each column is selected for a read or a write. On a write operation, the BLs are driven by the write buffer and the selected SRAM cell is written. During a read, the BLs are driven by the selected SRAM cell. The read out value propagates to the GBL through the NAND4 and NOR gate. The GBL of the top four banks and bottom four banks are combined through an AND2 gate. For each way, the result is the read out. These are passed to the way multiplexer. The way selection generated by tag decides

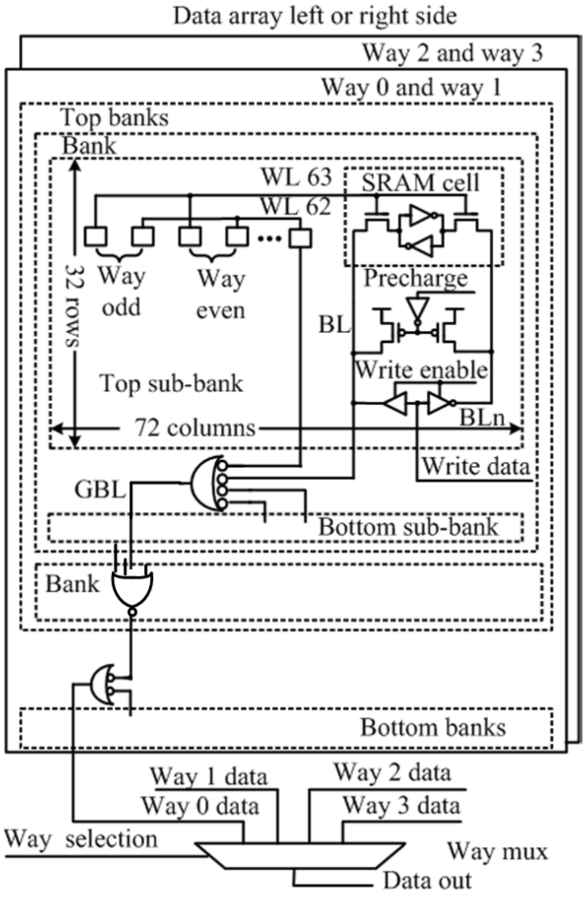


Figure 7.2 Basic diagram of data array (after [Yao09])

which way to forward the cache output. The final read out data leaves the cache after being passed to an inverter SDL.

The WL decoder drives 64 WLs in a sub-bank. There are two WLs in each row, with 36 SRAM cells on each WL. The SRAM cell uses annular NMOS pulldown transistors, which are much bigger than in a standard unhardened SRAM cell. The WL is long (same as the width of 72 SRAM cells) and heavily loaded (with the gates of the large access NMOS transistors for the 36 SRAM cells).

7.3.1.1 **Layout of data array**

The complete layout of the data array was carried out in two steps. First, the banks were built independently and then the banks, along with standard cells were put together, to complete the layout. Since all the 16 banks used in the design are identical, these banks are full custom. Each of the data banks consists of two sub-banks: top and bottom sub-banks as shown in Figure 7.2. The custom design achieves a highly compact design. Moreover, there is dynamic logic in the bank, which is best laid out manually. The fully laidout bank was characterized to generate the library characterization file (*i.e.,* .lib) for the bank. The characterization was first carried out using the Synopsys Nanotime static timing tool and then the values were further verified using the actual HSPICE simulation of the post extracted netlist.

The rest of the cache layout was carried out by placing the banks and the standard cells in the flow outlined above. The banks were treated as macro blocks and their placement was defined in the spreadsheet. Precaution was taken to ensure the RHBD required bit interleaving and maintenance of the spatial distance between the signals traversing from these banks and merging into the standard cells. This spreadsheet is read by the Perl script that generates the Encounter compatible placement file. Figure 7.3 shows the placement details of the data array. The standard cells are placed in between every two banks. This reduces the wire run for the signals and hence reduces the excessive loading of standard cells. The standard cells are placed in a manner to maintain a laminar flow of the signals to and from the banks. Since the banks maintain the proper bit interleaving

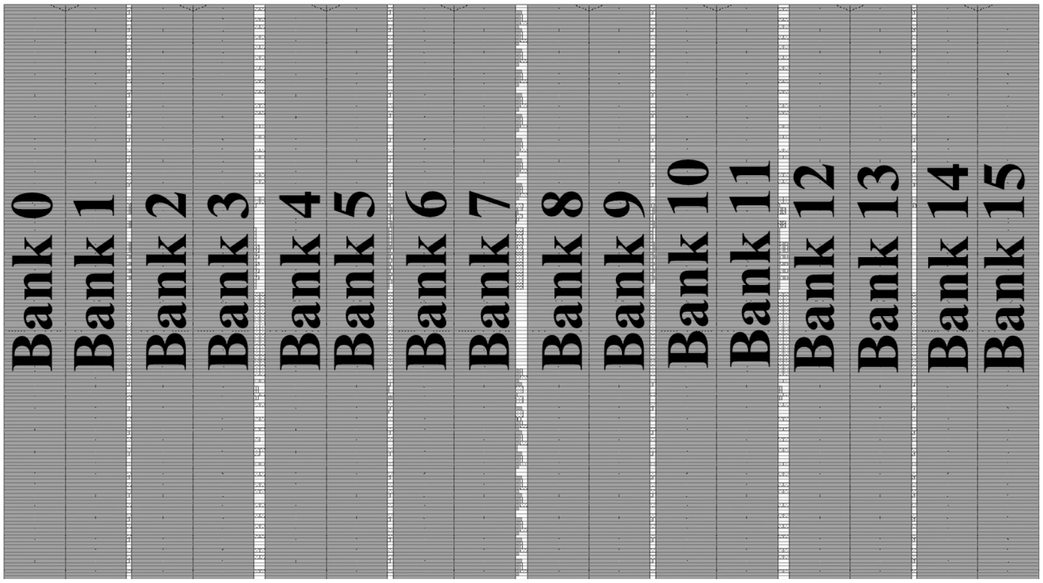


Figure 7.3 Data bank and standard cell placement for the cache data array

and spatial separation between the SEE susceptible signals, this way of placing the standard cells automatically ensure the radiation hardening criteria.

Once the placement of the standard cells and the banks are finalized in the Encounter floorplan, the design is routed using the naoroute in Encounter. The design is saved in gdsII format and is then imported back into the Cadence Virtuoso environment. The complete layout of the data array is shown in Figure 7.4. The layout is compact, regular and also since it is generated through the script, any modifications can be applied to the design with reasonable design and characterization effort.

*7.3.2 **Tag array***

The tag is composed of four ways, with four banks in each way and two sub-banks (top and bottom sub-bank) in each bank. Each way has dual redundant tag comparison and hit generation logic, so that SETs on them can be detected.

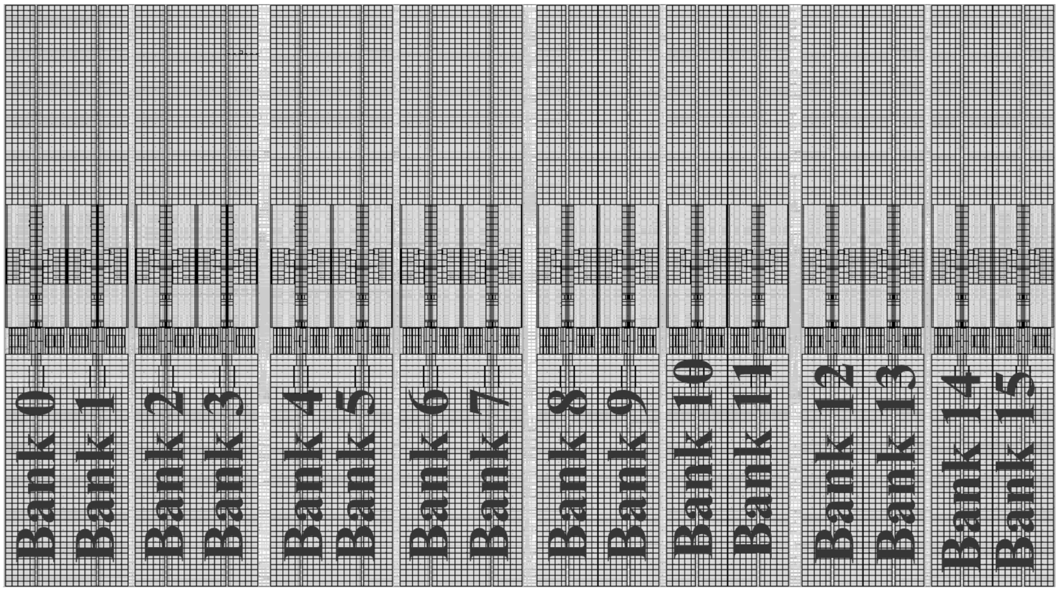


Figure 7.4 complete layout of cache data array showing the banks and the standard cells

There are 32 rows and 28 columns in a sub-bank. The components of each way are shown in Figure 7.5. The tag rows use interleaved layout. There are 24 data bits and 4 parity bits. The data bits consist of a lock bit, a least recently filled (LRF) bit, two valid bits, and 20 tag address bits. The lock bit indicates a line what should not be replaced, making the cache suitable for real-time systems. The LRF bit indicates the least recently filled line. It is used to determine which line to replace after a miss happens. The valid bits are dual redundant and indicate the corresponding cache line is valid. For an invalid cache line, if either valid bit is set by SEE, the other one still indicates the cache line is invalid.

The tag array stores the address affiliated with each line in the cache. It must be faster than the data array, since it compares the stored address and selects the requisite way in the data array in parallel with the data read. Similar to the data array, the tag array has WL decoders, SRAM cells columns, precharge for BL and write circuits. In each sub-bank, there are 32 rows and 28 columns of

SRAM cells. To speed up the BL development, BLs are split into two groups of 16 top (or bottom) SRAM cells in the same column. Therefore, the load on the tag BL is about half of that in the data array. Since the critical path in the cache includes the hit generation, the faster BL development is necessary to speed up the way selection signals generation. The outputs of the tag include way selection, four status bits (lock, LRF and two valid bits) of all four ways, as well as the tag address and parity bits of the selected way. Figure 7.5 shows the major components and the structure of the tag array.

There are 28 bits in a row of the tag, including 20 address bits, four status bits (lock, LRF, and two valid bits), and four parity bits. They are divided into four parity groups. The distance between bits of the same parity group is the width of three cells, i.e., 21.3 µm. The valid bits are protected from SEE by dual redundancy. The reason is that they are cleared independently from other bits in the tag, during power up initialization or global cache invalidation operation. When the cache is read out, the dual valid bits are output at the same time. If either bit is low, the output is invalid. The distance between the dual valid bits are separated by three SRAM cells. The distance between them is enough so that a single ionizing particle strike should only affect one, potentially invalidating the set.

The tag address read from SRAM cells is compared to the physical address in the hit generation circuits using the static XOR gates. Each The XOR gate compares one of the 20 bits in the address. Way-hit signal is generated by a logically ANDing these 20-bit XOR output.

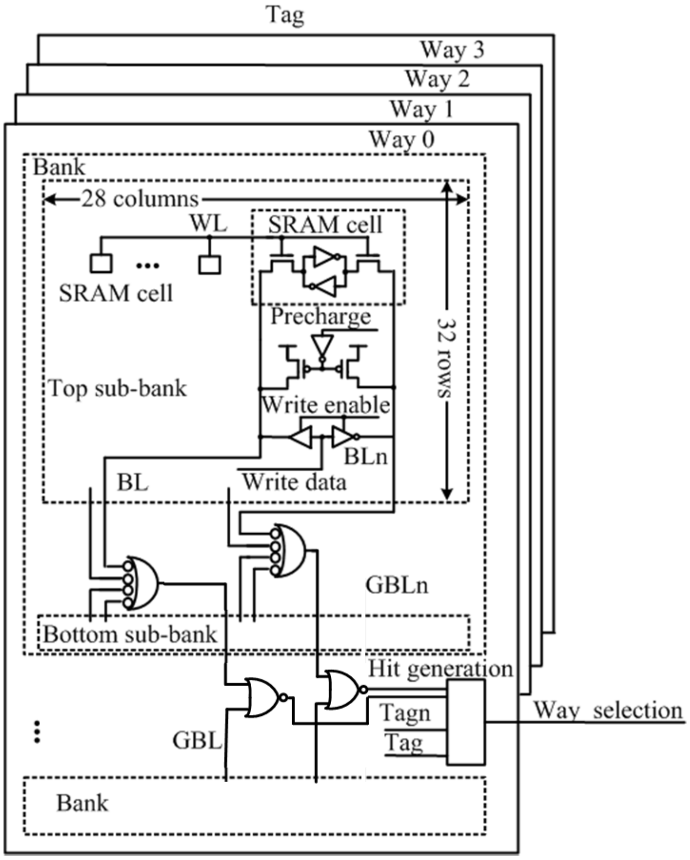


Figure 7.5 Basic diagram of tag (after [Yao09])

The error checking circuits in the tag are similar to those in the data array. To detect an SEE-induced error in the hit generation circuits, the hit generation logic is dual redundant in each way, named Match A and Match B as illustrated in Figure 7.6 [Yao09]. The way selections from the dual redundant hit generation can then be checked against each other. If there is a mismatch, an error is flagged. This dual hit generation does not imply that the area is double that of a single instance hit generation logic. Because there are 18 way-multiplexers on each half side of the data array and the way selection signals must travel a considerable distance, the way selection drivers are large.

7.3.2.1 **Layout of tag array**

The complete layout of the tag array was carried out similar to the data array layout, in two steps. First banks were independently built and then banks along with standard cells were put together to get the complete layout. Since all the 16 banks used in the design are similar, first these banks were fully custom built. Each of the tag banks consists of two sub-banks: top and bottom sub-banks as shown in Figure 7.5. The custom design lets to achieve highly compact design and also because there are dynamic logics in the bank which are best laid out manually. Once the layout of the bank is finalized, it was characterized to generate the library characterization file for the bank. The characterization was first carried out using the Nanotime Synopsys library characterization tool and then the values were further verified using the actual HSPICE simulation of the post extracted netlist.

Once the banks are ready, the rest of the layout was carried out by placing the banks and the standard cells into the spreadsheet. The banks were treated as macro blocks and their placement was defined in the spreadsheet as per their instance names. Precaution was taken to ensure bit interleaving and spatial distance between the signals traversing from these banks and merging into the standard cells. This spreadsheet is read by the Perl script that generates the Encounter compatible placement file. The Figure 7.7 shows the placement details of the data array. The standard cells are placed in between every two banks. This reduces the wire run for the signals and hence reduces the excessive loading of standard cells. The standard cells are placed in a manner to maintain a laminar

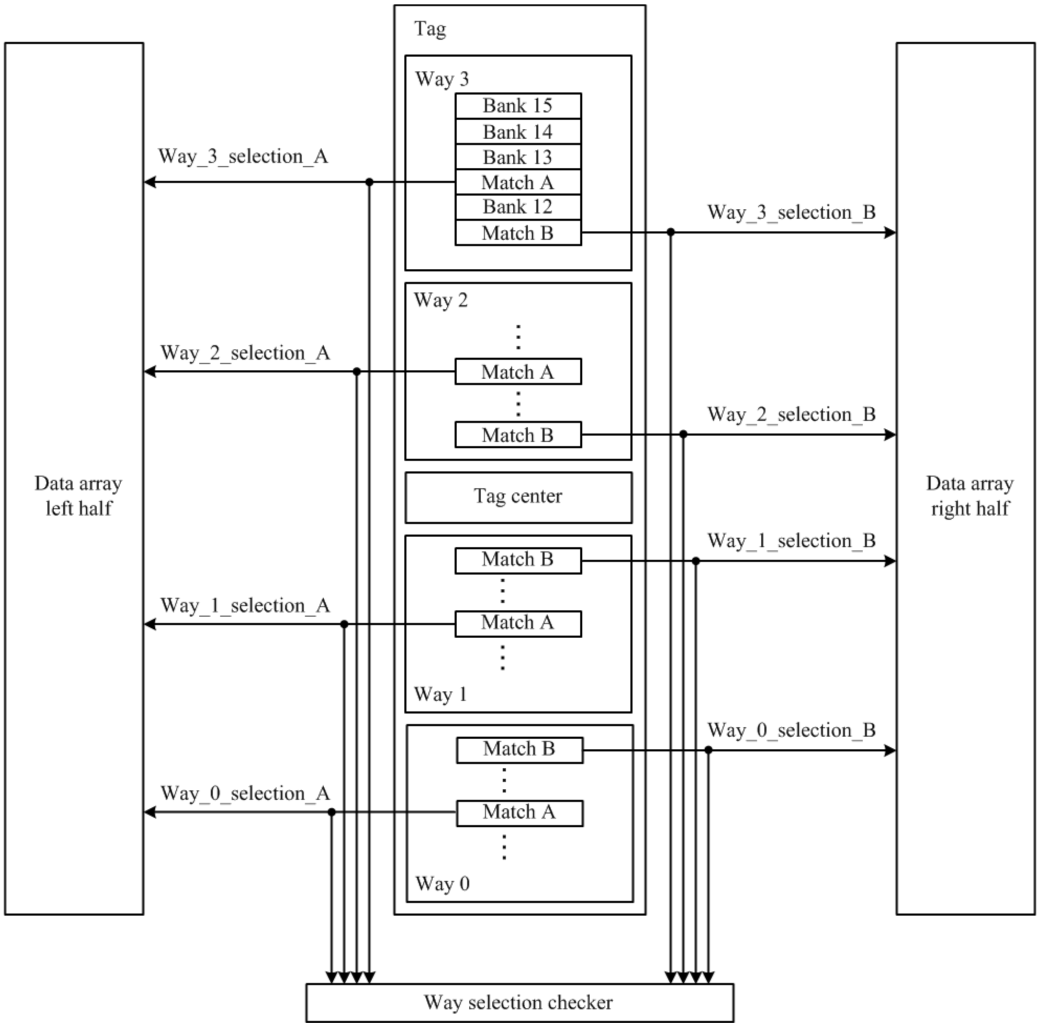


Figure 7.6 Basic diagram of tag (after [Yao09])

flow of the signals to and from the banks. Since the banks maintain the proper bit interleaving and spatial separation between the SEE susceptible signals, this way of placing the standard cells automatically ensure the radiation hardening criteria.

7.3.3 ***Overall cache layout***

The sub-bank clock drivers are the clock gated cells that generates the clocks whenever enabled for the respective sub-banks in each array. There are 96 sub-bank clock drivers 32 for each array and two per bank in an array (both top

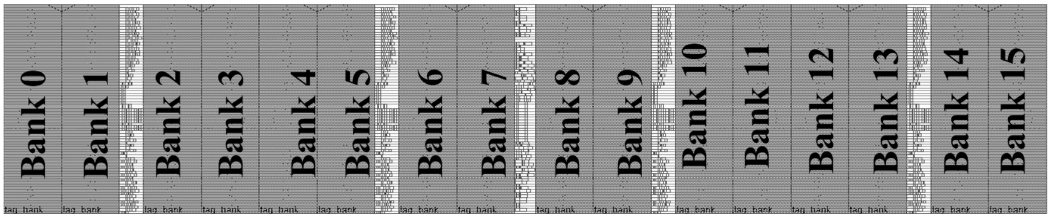


Figure 7.7 Tag bank and standard cell placement for the cache tag array

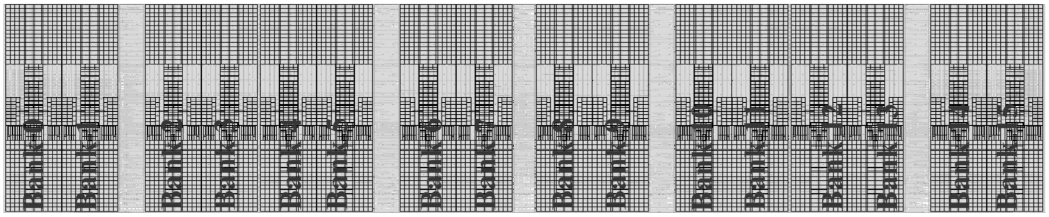


Figure 7.8 Complete layout of cache tag array showing the banks and the standard cells

and bottom sub-bank). TLB pfn circuits drive the input TLB address differentially to be compared against the cache tag address within the tag array. In order to ensure proper data is written to the data and tag array several data and tag write checkers are implemented. They are all implemented using dynamic logic since they need to be evaluated quickly before the actual data is written to ensure data integrity.

These blocks are then characterized using the Synopsys NCX tool to generate the library file. The .lib files for the data and tag arrays, the periphery blocks and the standard cells are then given to the Synopsys Design compiler along with the RTL code of the overall cache to synthesize the cache. The synthesized netlist is then given to Cadence Encounter for placement and routing purpose. In order to reduce the skew these sub-bank clock drivers are placed either equidistantly from the corresponding target sub-bank clock or by up/downsizing the sub-bank clock driver cells. The placement of the overall cache is shown in Figure 7.10. The six grey blocks in line just below the right and left

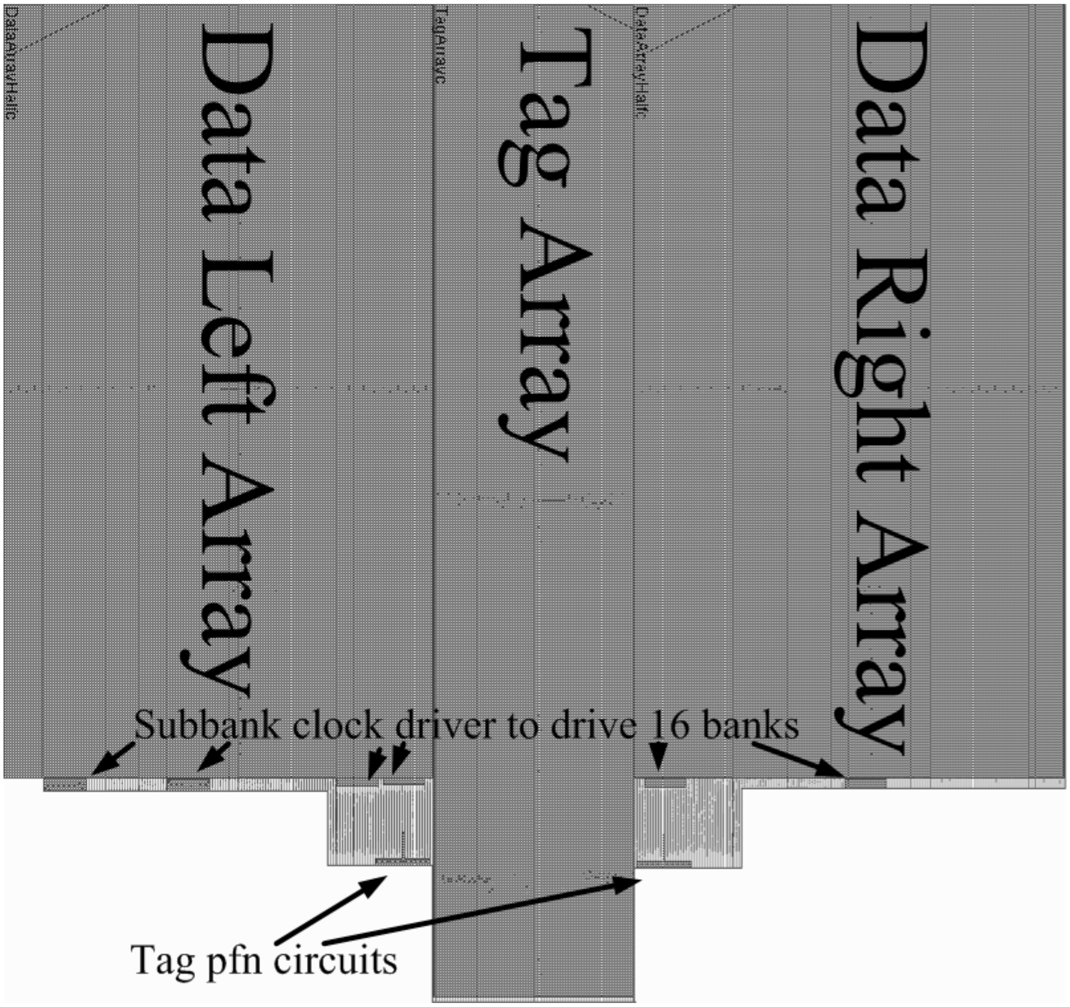


Figure 7.9 Tag and data array along with standard cell and periphery block placement for the overall cache design

data arrays are the sub-bank clock driver each consisting of 16 sub-bank clock driver sets. Hence each array is driven by two such blocks. The data and tag checkers are placed keeping in mind the spatial separation required for the A and B copy. Rests of the standard cells are placed by the Encounter and are optimized by the tool.

Once the placements of blocks are finalized in the Encounter floorplan, routing is done using the nanoroute tool. Since the clocks, especially the sub-banks clocks, are timing critical, these are routed in higher metals (M5-M6) with

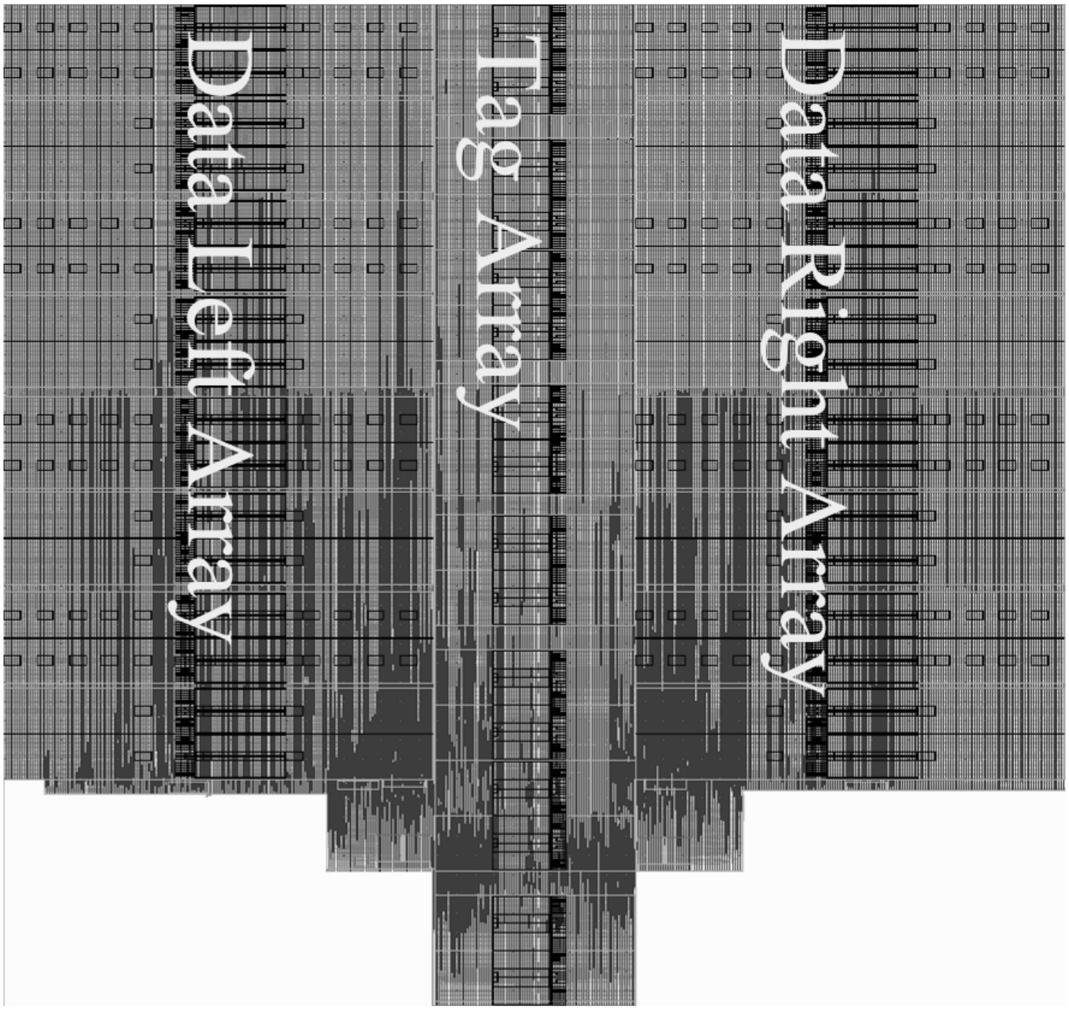


Figure 7.10 Complete layout of the overall cache design showing the tag and data array along with standard cell and periphery block

wire width of 500 nm. This ensures lower resistance on the wire and hence smaller driver cells and optimal delay. All other routing is carried using the default metal width of 140nm. The overall cache layout is shown in Figure 7.11.

7.3.3.1 Performance analysis

The delay analysis of the fully assembled cache was carried out using Prime Time. Since all Library files for the banks, periphery blocks and standard cells were already generated, these along with the synthesized netlists, when input to the tool, give the worst case timing for the cache. As per the tool, the cache can

Table 7.1 Maximum power consumption estimation during a read and write cycle at V_{DD} = 1.2 V

	Simulation (mW)		Switched Capacitor (mW)	
	Read	Write	Read	Write
Half data array	53	49	56	47
Tag array	21	19	22	18
Cache	127	117	134	112

run at 800 MHz at V_{DD} of 1.2 and temperature 85 °C. The original dynamic cache can be simulated at 1GHz across the similar corner. However, it does not have the peripheral circuits.

The critical path for this overall cache starts from the TLB pfn circuit that takes the address from the former TLB block and generates the differential address. The differential address along with the address read from the tag array using tag WL decoder is used for the way-hit generation in the tag array that eventually generates the way hit read signal for the data array. The critical path ends at the way multiplexers in the data array terminating at the inverter SDL logic.

The power consumed by the cache is estimated here by using circuit simulation. Another approach is to calculate the capacitances that are precharged and discharged during an operation. The capacitances include transistor and wire capacitance. Leakage is not considered in the latter approach. However, it provides a good guideline to check the simulation results.

Table 7.1 lists the results from both approaches for the power consumed in a read or write operation. The power dissipated by a read operation at 800 MHz in the cache is 127 mW. The power dissipation for a write operation is 117 mW. The estimation based on switched capacitors gives almost same results as that from simulation, providing some comfort for the designer that the results are correct. Among cache operations, lookup consumes more power than other operations. It requires reads of four data array sub-banks (two in left, and two in right half data array), and four sub-banks in tag array (one sub-bank for each way).

CHAPTER 8. **CONCLUSION**

The ever-increasing levels of CPU performance demanded by embedded applications and product design cycles that have often been reduced to only a few months have made it important to produce processor cores capable of execution speeds heretofore only achievable by complex custom solutions. However, in many cases, it is impractical to use custom design flows.

The proposed structured methodology aims at achieving the custom performance at a much reduced design time. The proposed flow has been shown to provide much better performance compared to a fully soft core design methodology. Incorporating the best of both custom and ASIC flow, the structured design flow has an edge not only in terms of ease of design but also testing the design against the timing and functional verification. However, full custom performance was achieved only when the precharge phase could otherwise be utilized, *i.e.,*in the IPCAM and issue queue designs.

The goal is to create a very high performance design, but it also been shown that the resulting design is more portable, similar to a soft-core, across existing and future technologies. This ruled out the use of extensive custom circuits, and led to the adoption of a methodology close to a traditional ASIC design flow, but one tuned to achieving aggressive performance goals.

The use of standard cells from the ASIC library is incorporated to limit design effort. If needed, more custom standard cells are added to the library. This gives the designer freedom to use more complex gates (like SRAM, CAM, TCAM etc.), which are generally not available in the standard ASIC library but significantly optimize the critical timing paths. Judicious use of dynamic gates or set domino latch (SDL) etc. can make the design highly optimized in terms of area, power and delay. SDL's have been a key component in the design as it limits the signal flow from dynamic logic domain to static logic domain. The SDL's only allow the evaluated logic to flow through them and block the precharged logic to flow through.

A simulation comparison of the behavioral verilog netlist extracted from the schematic against the frontend RTL code of the design ensures the functional verification of the circuit design. The timing verification is carried out by giving the same behavioral verilog code to Synopsys Prime time along with the library model files. Characterization of the custom built standard cells used in the design is carried out before running the Prime time. This gives the worst case timing for the design and if it does not meets the specification; the critical path is optimized again to meet the specification. Some delay margin is given for the interconnect delay even when the wire delay model is incorporated in the Prime time static timing analysis.

Once the functional and timing verification is proven, the placement of the standard cells is facilitated by a spreadsheet input format. The cells in the critical paths are placed locally to reduce the wire delay. This ensures the critical timing delay is within limits. The Perl program, developed as a part of this work then create the placement file for Encounter to route the design. In order to ensure the design specifications are met even with the actual layout of the design, delay information from the Encounter is extracted as *.spef* format file that is given as input to the Prime time to reanalyze the timing but this time with the actual wire delay. The designer iterates the design procedures to meet the performance and power goal.

The most appealing part of the structured design flow is the ease with which any modification in the design can be carried out – not only when bugs are found, but also if there is any modifications in the design specification or even for a technology node port. The most cumbersome part of the methodology is the schematic of the design and to some extent placing the cells in the spreadsheet. These steps not only reduce the time to make or change the design as against the fully custom design but also ensure superior performance in compliance with the fully custom design. However, as against the ASIC design flow, the design time remains similar. However, the improvement in performance is dramatic compared to pure synthesis. In addition to this, the structured design is regular and ordered as against the stochastic soft-core design.

Various designs have been carried out using the structured methodology and its performance comparison against the soft-core and fully custom designs carried out and presented. These designs are carried out across different technology nodes from 90nm to 45nm, proving the design portability along with the performance. The design of the internet protocol content addressable memory (IPCAM) which was designed in 65nm IBM process in both dynamic and static logic was again carried out using the static gates in IBM 45nm node. The performance in terms of area power and delay was analyzed. Both issue queue and IPCAM design establish the structured methodology.

The design of the fully static translation look-aside buffer (TLB) was carried out which was a part of the HERMES 32-bit MIPS microprocessor. In order to compare the static design, most popular implementation of the TLB using dynamic circuits was also carried out using the structured flow. The performance against the static, dynamic and soft-cores are presented. The designs were also carried out using the commercial standard cell library and RHBD (radiation hardened by design) library cells.

A static register file (RF) design is also carried out using the proposed flow. The comparison was made against the already existing dynamic RF design which is again the part of the HERMES microprocessor was carried out.

Finally, to prove the flow on a large, hierarchical block, the design a cache has been carried out part of the block is dynamic and part is implemented using static logic. This cache was initially built using fully dynamic logic but in the revised implementation, only the individual banks are dynamic. Performance is 80% of the original, but portability and revisability were achieved.

For all of these designs, using the structured approach the performance obtained was near that of a fully custom design, but with design time similar to an ASIC flow. Both the static and dynamic design was carried out. However the performance of the dynamic design was not near to the fully custom block. This is due to lack of skewed static gates that follow the dynamic blocks to enhance the performance.

REFERENCES

[Agarwal06] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, “Power gating with multiple sleep modes,” *in Proc. Int. Symp. Quality Electronic Design*, Mar. 2006, pp. 633–637.

[Bai02] X. Bai, C. Visweswariah, P. Strenski, and D. Hathaway, “Uncertainty aware circuit optimization,” *Proc. ACM/IEEE Design Automation* Conf., pp. 58–63, 2002.

[Benschneider95] B. Benschneider et al., “A 300-MHz 64-b quad-issue CMOS RISC microprocessor,” *IEEE J. Solid-State Circuits*, vol. 30, pp. 1203–1214, Nov. 1995

[BGP] BGP Routing Table Analysis Reports [Online]. Available: http://bgp.potaroo.net.

[Bhattacharya04] D. Bhattacharya, “Design-Specific standard Cells yield custom performance”, EEdesign, 2004. [online] http://www.eedesign.com/article/showArticle.jhtml?articleId=20301126.

[Bolotoff07] P. Bolotoff, “Alpha: The History in Facts and Comments”, [online]: http://alasir.com/articles/alpha_history/index.html.

[Borkar01] S. Borkar, “Low power design challenges for the decade” *In Conference on Asia South Pacific Design Automation*, pp. 293–296, Yokohama, Japan, January–February 2001.

[Borkar09] S. Borkar, “Design perspectives on 22nm CMOS and beyond”, *in IEEE International Electron Devices Meeting (IEDM)*, 2009, Baltimore, MD, pp. 1-1, Dec 2009.

[Burstein85] M.Burstein and M.N.Youssef. "Timing Influenced Layout Design," *in Proc. 22nd DAC*, 1985. pp. 124-130.

[Buti05] T.N. Buti, R. McDonald, Z. Khwaja, A. Ambekar, H. Le, W. Burky, and B. Williams, “Organization and Implementation of the Register Renaming Mapper for Out-of-Order IBM Power4 Processors,” IBM J. Research and Development, vol. 49 (1), pp. 167-188, Jan. 2005.

[Chandrakasan01] A. Chandrakasan, , W. Bowhill, and F. Fox, “Design of High- Performance Circuits” IEEE Press 2001.

[Chang05] A. Chang and W. Dally, “Explaining the gap between ASIC and custom power: a custom perspective”, *in 42nd Proceedings of Design Automation Conference*, pp. 281 – 284, 2005.

[Chang98] A. Chang, “VLSI Datapath Choices: Cell-Based Versus Full-Custom", SM Thesis, Massachusetts Institute of Technology, February 1998. [online] ftp://cva.stanford.edu/ pub/publications/achang_masterworks980427.pdf.

[Chaudhary06] V. Chaudhary and L. Clark, “Low-power high-performance NAND match line content addressable memories” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (2006), pp. 895–905.

[Chen06] W. Chen, “The VLSI handbook”, 2nd Ed. , CRC Press.

[Chinnery01] D.G. Chinnery, B. Nikolic, and K.Keutzer, “Achieving 550MHz in an ASIC Methodology,” *in Proc. of the Design Automation Conference (DAC)*, 2001, pp. 420–425.

[Chinnery02] D. Chinnery, and K. Keutzer, “Closing the Gap between ASIC and Custom”, Kluwer Academic Publishers, 2002.

[Clark03] L. Clark, M. Wilkerson and B. Choi, “Reducing translation look-aside buffer active power”, *in Proc. Int. Symp. Low Power Electronics and Design*, pp. 10 -13, 2003.

[Clein99] D. Clein, “CMOS IC layout: concepts, methodologies, and tools”, vol 1, Newnes Pub., December 22, 1999

[Costa84] E. Costa et al., “On The Integration of a CAD System for IC Design”, by, European Conference on EDA, Mar. 1984, pp. 40-45.

[Dally00] W. Dally, and A. Chang, “The Role of Custom Design in ASIC Chips”, *Design Automation Conference 2000*.

[David09] David Mosberger. "Overview of Alpha Family". Retrieved Dec 9 2009.

[Dobberpuhl92] D. Dobberpuhl et al., “A 200 MHz 64-bit Dual Issue CMOS Microprocessor,” *IEEE J. Solid State Circuits*, vol. 27, No. 11, Nov. 1992, pp. 1,555–1,567.

[Dunlop84] A.Dunlop, V. Agrawal, D. Deutsch, M. Jukl, P. Kozak and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting," *in Proc. 21st DAC*, 1984. pp.133-136.

[Dunlop85] A. Dunlop and B. Kernighan, “A procedure for placement of standard-cell VLSI circuits,” *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 92-98, Jan. 1985.

[EEtimes02] EE Times Design Article [online] http://www.eetimes.com/electronics-news/4142815/ASIC-design-flow-gives-CPU-core-custom-performance

[Eriksson03] H. Eriksson, P. Larsson-Edefors, T. Henriksson, C. Svensson, “Full-custom vs. standard-cell design flow - an adder case study”, *in Proceedings of the Design Automation Conference*, 2003. Asia and South Pacific, pp. 507 – 510

[Farrell98] J. A. Farrell and T. C. Fischer, “Issue logic for a 600-mhz out-of-order execution microprocessor*,” IEEE J. Solid-State circuits*, vol. 33 (5), pp. 707 – 712, May 1998.

[Fishburn85] J. Fishburn, and A. Dunlop, “A Polynomial Programming Approach to Transistor Sizing”. *Proc. of ICCAD ’85*, pp. 326-328.

[Gavrilov97] S. Gavrilov, A. Glebov, S. Moore, A. Dharchoudhury, R. Panda, G. Vijayan, and D. Blaauw, “Library-Less Synthesis for Static CMOS Combinational Logic Circuits*”, Proc. of ICCAD* ’97, pp. 658-663.

[Gray91] C. Gray, M. Smith, J. Rowson and M. O’Brian, “A Comparison of an ASIC Synthesized Design to a Schematic Entry Design for a Viterbi Decoder*”, in Proceedings of the 1991 Custom Integrated Circuits Conference*, 1991, pp. 11.1/1-1 1.1/4.

[Gronowski98] P. Gronowski, W. Bowhill, R. Preston, M. Gowan, and R. Allmon, “High-Performance Microprocessor Design*”*, *IEEE Journal of Solid-state Circuit*s, vol. 33(5), May 1998, pp. 676-686.

[Gyurcsik89] R. Gyurcsik and J. Jeen, “A generalized approach to routing mixed analog and digital signal nets in a channel*,” IEEE J. Solid-State Circuits*, vol. 24, pp. 436–442, Apr. 1989.

[Haigh04] J. Haigh, M. Wilkerson, J. Miller, T. Beatty, S. Strazdus and L. Clark, “A Low-Power 2.5 GHz 90 nm Level 1 Cache and Memory Management Unit,” *in IEEE Jnl. of Solid State Circuits*, 40(5), pp. 1190-1199, IEEE Press, 2004.

[Han87] T. Han and D. Carlson “Fast Area-Efficient VLSI Adders”, *In Proceedings of the 8th IEEE Symposium on Computer Arithmetic*, 1987, pp. 49-56.

[Handy98] J. Handy, *The Cache Memory Book,* San Diego: Academic Press, Inc, 1998.

[Hauge87] P.Hauge, R.Nair and E.J.Yoffa, "Circuit Placement for Predictable Performance,"*in Proc. IEEE ICCAD*, 1987, pp.88-91.

[Havemann89] R. Havemann, J. Hutchby, "High-performance interconnects: An integration overview.*" Proceedings of the IEEE 89*, 5 (May), pp. 586–601.

[Hill85] D. Hill, "A Hybrid Automatic Layout System". Proc. of ICCAD '85, pp. 172-174.

[Hill88] D. Hill, "Alternative Strategies for Applying Min-cut to VLSI Placement*", in Proceedings of the 1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 3-5 Oct 1988, pp. 440 – 444

[Hurst99] S. Hurst, "VLSI Custom Microelectronics: Digital: Analog, and Mixed-Signal", Marcel Dekker Inc.

[Hwang93] K. Hwang, "Advanced computer architecture : parallelism, scalability, programmability", Publisher New York [etc.] : McGraw-Hill, cop. 1993.

[Jackson89] M.Jackson and E.Kuh, "Performance-Driven Placement of Cell Based IC's," *in Proc. 26th DAC*, 1989, pp. 370-375.

[Juan97] T. Juan, T. Lang and J. Navarro, "Reducing TLB power requirements*", in Proc. of the Intl. Symp. on Low Power Electronics and Design*, 1997, pp. 196–201.

[Kadayif02] I. Kadayif, A. Sivasubramaniam, M. Kandemir, G. Kandiraju, and G. Chen, "Generating Physical Addresses Directly for Saving Instruction TLB Energy,".*in Proc. of the Intl. Symp. on Microarchitecture*, 2002. (MICRO-35), pp. 185 – 196.

[Keutzer87] Keutzer, K., Kolwicz, K., and Lega, M. Impact of Library Size on the Quality of Automated Synthesis. Proc. of ICCAD '87, pp. 120-123.

[Keyes82] R. Keyes, "The wire-limited logic chip," *IEEE J. Solid-State Circuits*, vol. 17, pp. 1232–1233, Dec. 1982.

[Kheterpal04] V. Kheterpal, A. J. Strojwas, L. Pileggi, "Routing Architecture Exploration for Regular Fabrics", *Proceedings of the ACM/IEEE DAC,* June 2004, pp. 204-207

[Kim96] J.Kim, S.M.Kang “A New Triple-Layer OTC Channel Router” *in IEEE Transactions on CAD*, Vol. 15, no. 9,September 1996, pp. 1059-1070.

[Knowles01] S. Knowles, “A Family of Adders”, *In Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 277-281.

[Ledalla04] R. Ledalla, J. Soreff and F. Yang, “Method for reducing RC parasitics in interconnect networks of an integrated circuits”, US Patent No. 6,763,504 B2.

[Lefebvre97] M. Lefebvre, D. Marple, C. Sechen. ‘The Future of Custom Cell Generation in Physical Synthesis,” *in proc. 1997 Design Automation Conference*, pp. 446-451

[Little92] W. Little, “Integrated circuit with watchdog timer and sleep control logic which places IC and watchdog timer into sleep mode”, US Patent 5,175,845, 1992

[Loos96] J. Loos, C. Wang, M. Mahmood, “Electronic design automation apparatus and method utilizing a physical information database”, US Patent 5,487,018, 1996.

[Maurya10] S. K. Maurya, and L. T. Clark, “A Dynamic Longest Prefix Matching Content Addressable Memory for IP Routing”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (2010), pp. 1 - 10 .

[McFarland88] M. C. McFarland, A. C. Parker, and R. Camposano, “Tutorial on high-level synthesis,” *in Proc. 25th Design Automation Conf.* , June 1988, pp. 330-336.

[Mhambrey10] S. Mhambrey, L. Clark, S. Maurya and K. Berezowski, “Out of order issue logic using sorting networks,” *Proc. of 20th Great Lakes Symposium on VLSI*, pp. 385-388, May 2010.

[Microprocessor12] Microprocessor chronology. [online]: http://en.wikipedia.org/wiki/Microprocessor_chronology.

[Mohr07] K. Mohr, G. Samson and L. Clark "A radiation hardened by design register file with low latency and area cost error detection and correction," *in IEEE Trans. Nuc. Sci.*, vol. 54, no. 4, pp. 1335-1342, Aug. 2007.

[Moores] Moore's law - methods to study new environments. [online] http://www.hcibook.com/e3/online/moores-law/

[Moore03] G. Moore, “No Exponential is Forever: But ‘Forever’ can be delayed!,” *ISSCC Dig. Techn. Papers*, pp. 21-23, Feb. 2003.

[Moore65] G. Moore, “Cramming more components into Integrated Circuits,” Electronics, Vol. 38, Nr 8, April 1965.

[Moraes98] F.Moraes, L.Torres, M.Robert and D.Auvergne “Estimation of Layout Densities for CMOS Digital Circuits”, *PATMOS '98 - International Workshop on Power and Timing Modeling, Optimization and Simulation,* October 07-09 1998, Denmark, pp. 61-70.

[Ogawa86] Y.Ogawa, T. Ishii, Y. Shiraishi, H. Terai, T. Kozawa, K. Yuyama and K. Chiba, "Efficient Placement Algorithms Optimizing Delay for High-speed ECL Masterslice LSI's,*" in Proc. 23rd DAC*, 1986, pp. 404-410

[Ogdin75] Ogdin, Jerry (January 1975). "Microprocessor scorecard". Euromicro Newsletter 1 (2), pp. 43–77. doi:10.1016/0303-1268(75)90008-5

[Ong89] C. Ong, J. Li, and C. Lo, “GENAC: An automatic cell synthesis tool”, *IEEE 26th Design Automation Conference*, July 1989, pp. 239-244.

[Opensource] Open Source Liberty, [online] www.opensourceliberty.org

[Patterson96] D.A. Patterson, John. L. Hennessy, *Computer Architecture: A quantitative Approach*. 2nd ed., Morgan Kaufmann Publishers, Inc., 1996, pp. 375-390.

[Peter81] K. Peter, “The Architecture of Pipelined Computers”, McGraw-Hill, Taylor & Francis Pub., January 1981.

[Richardson03] N. Richardson, L.B. Huang, R. Hossain, J. Lewis, T. Zounes, N. Soni, and J. Lewis “The iCore 520-MHz syntheesizable CPU core”, *in Proceedings of the 39th annual Design Automation Conference*, 2002, pp. 640-645.

[Scott94] K. Scott, K. Keutzer, “Improving Cell Libraries for Synthesis*”, IEEE Custom Integrated Circuits Conference*, 1994, pp. 128 – 131.

[Sherwani95] N.Sherwani. “ Algorithm for VLSI physical design automation”. Kluwer Academic Publisher, 538p. 1995.

[Shinohara91] H. Shinohara et al., “A Flexible Multiport RAM Compiler for Data Path,” *IEEE J. of Solid-State Circuits*, vol. 26(3), Mar. 1991, pp. 343 - 349.

[Siddhesh10] S. Mahambrey, "Low Complexity Out of Order Instruction Issue Logic", MS Thesis, Arizona State University, 2010

[Skotnicki05] T. Skotnicki, J. Hutchby, T. King, H. Wong and F. Boeuf F, "The end of CMOS scaling: toward the introduction of new materials and structural changes to improve MOSFET performance*", IEEE Circuits Devices Mag.*, vol 21(1), pp. 16–26, 2005.

[Stephen01] R. Stephen, P. Matthew and S. Jim, "Reducing the Frequency Gap Between ASIC and Custom Designs: A Custom Perspective", *in Proceedings of Design Automation Conference*, pp. 432 – 437, 2001.

[Stine05] J. Stine, J. Grad, I. Castellanos, J. Blank, V. Dave, M. Prakash, N. Iliev, and N. Jachimiec, "A Framework for High-Level Synthesis of System-on-Chip Designs", *In International Conference on Microelectronic Systems Education*, pp. 67–68. IEEE Computer Society Press, 2005

[Sylvester98] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron", *in ICCAD '98*, pp. 203-211.

[Teig86] S.Teig. R.L.Smith and J.Seaton.'Timing-Driven Layout of Cell-Based ICs," *in VLSI System Design*, May 1986, pp. 63-73.

[Hill88] D. Hill, "Alternative Strategies for Applying Min-cut to VLSI Placement*", in Proceedings of the 1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 3-5 Oct 1988, pp. 440 – 444

[Terai90] M. Terai, K. Takahashis and K. Sato, "A new min-cut placement algorithm for timing assurance layout design meeting net length constraint", *in Proceedings of Design Automation Conference*, 1990, pp. 96 – 102

[Tiri04] K. Tiri and I. Verbauwhede, "Place and route for secure standard cell design,*" in Proc. Smart Card Research and Advanced Application IFIP Conf. (CARDIS),* Toulouse, France, 2004, pp. 143–158.

[Toshiba] Toshiba Intellectual Property [online] http://www.toshiba.com/taec/Catalog/Line.do?familyid=1&lineid=7179

[Tremblay95] M. Tremblay, B. Joy and K. Shin, "A Three Dimensional Register File For Superscalar Processors." *In Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, pp. 191–201, January 1995.

[Vincentelli04] A. Vincentelli, L. Carloni, F. Bernardinis, and M. Srroi, "Benefits and challenges for platform-based design", *In Proceedings of the 41st annual Design Automation Conference,* pp. 409–414, 2004

[Weste92] Weste, N., and Eshraghian, K. Principles of CMOS VLSI Design. Addison-Wesley, 1992.

[Weste10] N. Weste and D. Harris, "CMOS VLSI Design: A circuit and systems perspective", 4th Ed., Addison-Wesley publication.

[Weng05] T. Weng, W. Chiao, J. Shann, C. Chung, and J. Lu, "Low-Power Data Address Bus Encoding Method", *in Proc. of CDES'05*, Las Vegas, USA, pp.204-210, June 2005

[Xing02] Z. Xing and R. Kao, "Method and apparatus for performing power routing in ASIC design", US Patent 6,446,245, 2002

[Yanwu10] G. Yanwu Gu and Z. Yanmin, "An Efficient Approach to Placement Legalization within Standard Cell Circuits", *in International Conference on Computer Design and Applications*, 2010, pp. 477-480

[Yao09] X. Yao, D.W. Patterson, L.T. Clark, and K.E. Holbert, "A 90 nm bulk CMOS radiation hardened by design cache memory," *presented at the 10th European Conference on Radiation Effects on Components and Systems*, 2009, September 14-18.

[Yao209] X. Yao, "Radiation hardened high performance microprocessor cache design", Phd dissertation, Arizona State University, 2009

[Yoshida04] H. Yoshida, K. De, and V. Boppana, "Accurate pre-layout estimation of standard cell characteristics," *in Proceedings of the 41st annual conference on Design automation*. New York, NY, USA: ACM Press, 2004, pp. 208–211.

[Zhang01] Y. Zhang, X. Hu and D. Chen, "Cell Selection from Technology Libraries for Minimizing Power", *in Integration, the VLSI Journal*, Vol.31 (2), May 2002, pp. 133-158