Performance and Energy Efficient Building Blocks for Network-on-Chip Architectures

Sriram R. Vangal
Abstract

The ever shrinking size of the MOS transistors brings the promise of scalable Network-on-Chip (NoC) architectures containing hundreds of processing elements with on-chip communication, all integrated into a single die. Such a computational fabric will provide high levels of performance in an energy efficient manner. To mitigate emerging wire-delay problem and to address the need for substantial interconnect bandwidth, packet switched routers are fast replacing shared buses and dedicated wires as the interconnect fabric of choice. With on-chip communication consuming a significant portion of the chip power and area budgets, there is a compelling need for compact, low power routers. While applications dictate the choice of the compute core, the advent of multimedia applications, such as 3D graphics and signal processing, places stronger demands for self-contained, low-latency floating-point processors with increased throughput. Therefore, this work focuses on two key building blocks critical to the success of NoC design: high performance, area and energy efficient router and floating-point processor architectures.

This thesis first presents a six-port four-lane 57 GB/s non-blocking router core based on wormhole switching. The router features double-pumped crossbar channels and destination-aware channel drivers that dynamically configure based on the current packet destination. This enables 45% reduction in crossbar channel area, 23% overall router area, up to 3.8X reduction in peak channel power, and 7.2% improvement in average channel power, with no performance penalty over a published design. In a 150nm six-metal CMOS process, the 12.2mm$^2$ router contains 1.9 million transistors and operates at 1GHz at 1.2V.
We next present a new pipelined single-precision floating-point multiply accumulator core (FPMAC) featuring a single-cycle accumulate loop using base 32 and internal carry-save arithmetic, with delayed addition techniques. Combined algorithmic, logic and circuit techniques enable multiply-accumulates at speeds exceeding 3GHz, with single-cycle throughput. Unlike existing FPMAC architectures, the design eliminates scheduling restrictions between consecutive FPMAC instructions. The optimizations allow removal of the costly normalization step from the critical accumulate loop and conditionally powered down using dynamic sleep transistors on long accumulate operations, saving active and leakage power. In addition, an improved leading zero anticipator (LZA) and overflow detection logic applicable to carry-save format is presented. In a 90nm seven-metal dual-\(V_T\) CMOS process, the 2mm\(^2\) custom design contains 230K transistors. The fully functional first silicon achieves 6.2 GFLOPS of performance while dissipating 1.2W at 3.1GHz, 1.3V supply.

It is clear that realization of successful NoC designs require well balanced decisions at all levels: architecture, logic, circuit and physical design. Our results from key building blocks demonstrate the feasibility of pushing the performance limits of compute cores and communication routers, while keeping active and leakage power, and area under control.
Preface

This licentiate thesis presents my research as of April 2006 at the Electronic Devices group, Department of Electrical Engineering, Linköping University, Sweden. The following publications are included:


As a staff member of Microprocessor Technology Laboratory at Intel Corporation, Hillsboro, OR, USA, I am also involved in research work, with several publications not discussed as part of this thesis:


# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Alternating Current</td>
</tr>
<tr>
<td>AR</td>
<td>Aspect Ratio</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal-Oxide-Semiconductor</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic Random Access Memory</td>
</tr>
<tr>
<td>DSM</td>
<td>Deep SubMicron</td>
</tr>
<tr>
<td>FF</td>
<td>Flip-Flop</td>
</tr>
<tr>
<td>FP</td>
<td>Floating-Point</td>
</tr>
<tr>
<td>FPADD</td>
<td>Floating-Point Addition</td>
</tr>
<tr>
<td>FPMAC</td>
<td>Floating-Point Multiply Accumulator</td>
</tr>
<tr>
<td>FPU</td>
<td>Floating-Point Unit</td>
</tr>
<tr>
<td>FO4</td>
<td>Fan-Out-of-4</td>
</tr>
<tr>
<td>GB/s</td>
<td>Gigabyte per second</td>
</tr>
<tr>
<td>IEEE</td>
<td>The Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>ILD</td>
<td>Inter-Layer Dielectric</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ITRS</td>
<td>International Technology Roadmap for Semiconductors</td>
</tr>
<tr>
<td>LZA</td>
<td>Leading Zero Anticipator</td>
</tr>
<tr>
<td>MOS</td>
<td>Metal-Oxide-Semiconductor</td>
</tr>
<tr>
<td>MOSFET</td>
<td>Metal-Oxide-Semiconductor Field Effect Transistor</td>
</tr>
<tr>
<td>MSFF</td>
<td>Master-Slave Flip-Flop</td>
</tr>
<tr>
<td>MUX</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>NMOS</td>
<td>N-channel Metal-Oxide-Semiconductor</td>
</tr>
<tr>
<td>NoC</td>
<td>Network-on-Chip</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Locked Loop</td>
</tr>
<tr>
<td>PMOS</td>
<td>P-channel Metal-Oxide-Semiconductor</td>
</tr>
<tr>
<td>RC</td>
<td>Resistance-Capacitance</td>
</tr>
<tr>
<td>RF</td>
<td>Radio-Frequency</td>
</tr>
<tr>
<td>RLC</td>
<td>Resistance-Inductance-Capacitance</td>
</tr>
<tr>
<td>SoC</td>
<td>System-on-a-Chip</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very-Large Scale Integration</td>
</tr>
</tbody>
</table>
Acknowledgments

I would like to express my sincere gratitude to my principal advisor and supervisor Prof. Atila Alvandpour, for motivating me and providing the opportunity to embark on research at Linköping University, Sweden. I also thank Dr. Peter Caputa, and Lic. Eng. Martin Hansson, Anna Folkesson and Arta Alvandpour at Linköping University for their assistance.

I am thankful for support from all of my colleagues at Microprocessor Technology Labs, Intel Corporation. Special thanks to my manager, Nitin Borkar, and Matthew Haycock, for their support; Shekhar Borkar, and Justin Rattner for their technical leadership and for funding this research. I also thank Dr. Yatin Hoskote, Dr. Vivek De, Dr. Dinesh Somasekhar, Dr. Tanay Karnik, Dr. Siva Narendra, Dr. Ram Krishnamurthy, Dr. Sanu Mathew and Jim Tschanz for numerous technical discussions. I am also grateful to the silicon design and prototyping team: Howard Wilson, Vasantha Erraguntla, Jason Howard, Greg
Ruhl, Saurabh Dighe, Arvind Singh, Venkat Veeramachaneni, Amaresh Pangal, Venki Govindarajulu, David Finan and all mask designers for providing an ideal research and prototyping environment for making silicon realization of the research ideas possible. Greg Dermer and Brad Bloechel deserve credit for their invaluable support in the lab with silicon testing.

All of this was made possible by the love and encouragement from my family. I am indebted to my inspirational grandfather Sri. V. S. Srinivasan, and my parents, for their constant motivation from the other side of the world. I sincerely thank Prof. K. S. Srinath, for being my mentor since the early years. Without his visionary guidance and support, my career dreams would have remained unfulfilled.

Finally, I am exceedingly thankful for the patience, love and support from my soul mate: Reshma.

Sriram Vangal
Linköping, April 2006
Contents

Abstract iii
Preface v
Abbreviations ix
Acknowledgments xi
List of Figures xvii

Part I Introduction 1

Chapter 1 Introduction 3
1.1 The Microelectronics Era and Moore’s Law.........................4
1.2 Low Power CMOS Technology .........................................5
  1.2.1 CMOS Power Components 5
1.3 Technology scaling trends and challenges ...........................7
  1.3.1 Technology scaling: Impact on Power 8
  1.3.2 Technology scaling: Impact on Interconnects 11
  1.3.3 Technology scaling: Summary 12
1.4 Motivation and Scope of this Thesis ..................................13
1.5 Organization of this Thesis.............................................14
Part II NoC Building Blocks

Chapter 2 On-Chip Interconnection Networks

2.1 Interconnection Network Fundamentals

2.1.1 Network Topology
2.1.2 Message Switching Protocols
2.1.3 Virtual Lanes
2.1.4 A Generic Router Architecture

2.2 A Six-Port 57GB/s Crossbar Router Core

2.2.1 Introduction
2.2.2 Router Micro-architecture
2.2.3 Packet structure and routing protocol
2.2.4 FIFO buffer and flow control
2.2.5 Router Design Challenges
2.2.6 Double-pumped Crossbar Channel
2.2.7 Channel operation and timing
2.2.8 Location-based Channel Driver
2.2.9 Chip Results

2.3 Summary and future work

2.4 References

Chapter 3 Floating-point Units

3.1 Introduction to floating-point arithmetic

3.1.1 Challenges with floating-point addition and accumulation
3.1.2 Scheduling issues with pipelined FPUs

3.2 Single-cycle Accumulation Algorithm

3.3 CMOS prototype

3.3.1 High-performance Flip-Flop Circuits
3.3.2 Test Circuits

3.4 References

Part III Papers

Chapter 4 Paper 1

4.1 Introduction
4.2 Double-pumped Crossbar Channel with LBD
4.3 Comparison Results
4.4 Acknowledgments
4.5 References

Chapter 5 Paper 2
5.1 FPMAC Architecture ................................................................. 64
5.2 Proposed Accumulator Algorithm ............................................. 65
5.3 Overflow detection and Leading Zero Anticipation .................... 67
5.4 Simulation results ..................................................................... 69
5.5 Acknowledgments ..................................................................... 70
5.6 References ............................................................................... 71

Chapter 6 Paper 3

6.1 Introduction ............................................................................... 74
6.2 FPMAC Architecture ............................................................... 76
  6.2.1 Accumulator algorithm ......................................................... 77
  6.2.2 Overflow prediction in carry-save ........................................ 79
  6.2.3 Leading zero anticipation in carry-save ................................. 82
6.3 Conditional normalization pipeline ............................................. 83
6.4 Design details ........................................................................... 85
6.5 Experimental Results ............................................................... 88
6.6 Conclusions ............................................................................. 93
6.7 Acknowledgments ..................................................................... 94
6.8 References ............................................................................... 94
List of Figures

Figure 1-1: Transistor count per Integrated Circuit ......................................................... 4
Figure 1-2: Basic CMOS gate showing dynamic and short-circuit currents ....................... 6
Figure 1-3: Main leakage components for a MOS transistor ............................................. 7
Figure 1-4: Power trends for Intel microprocessors ........................................................... 8
Figure 1-5: Dynamic and Static power trends for Intel microprocessors ............................... 9
Figure 1-6: Sub-threshold leakage current as a function of temperature ............................ 9
Figure 1-7: Three circuit level leakage reduction techniques ............................................. 10
Figure 1-8: Intel 65 nm, 8-metal copper CMOS technology in 2004 .................................... 11
Figure 1-9: NoC Architecture ............................................................................................ 12
Figure 2-1: Direct and indirect networks .......................................................................... 20
Figure 2-2: Popular on-chip fabric topologies ................................................................... 21
Figure 2-3: A fully connected non-blocking crossbar switch ............................................ 22
Figure 2-4: Reduction in header blocking delay using virtual channels ............................... 24
Figure 2-5: Canonical router architecture ......................................................................... 25
Figure 2-6: Six-port four-lane router block diagram ......................................................... 27
Figure 2-7: Router data and control pipeline diagram ....................................................... 28
Figure 2-8: Router packet and FLIT format ....................................................................... 29
Figure 2-9: Flow control between routers .......................................................................... 30
Figure 2-10: Crossbar router in [19] (a) Internal lane organization. (b) Corresponding layout. (c) Channel interconnect structure ............................................................................ 30
Figure 2-11: Double-pumped crossbar channel .................................................................. 32
Figure 2-12: Crossbar channel timing diagram ................................................................. 33
Figure 2-13: Location-based channel driver and control logic ......................................... 34
Figure 2-14: (a) LBD Schematic. (b) LBD encoding summary ........................................... 34
Figure 2-15: Double-pumped crossbar channel results compared to work reported in [19]... 35
Figure 2-16: Simulated crossbar channel propagation delay as function of port distance .... 36
Figure 2-17: LBD peak current as function of port distance ............................................. 37
Part I

Introduction
Chapter 1

Introduction

For four decades, the semiconductor industry has surpassed itself by the unparalleled pace of improvement in its products and has transformed the world that we live in. The remarkable characteristic of transistors that fuels this rapid growth is that their speed increases and their cost decreases as their size is reduced. Modern day high-performance Integrated Circuits (ICs) have more than one billion transistors. Today, the 65 nanometer (nm) Complementary Metal Oxide Semiconductor (CMOS) technology is in high volume manufacturing and industry has already demonstrated fully functional static random access memory (SRAM) chips using 45nm process. This scaling of CMOS Very Large Scale Integration (VLSI) technology is driven by Moore’s law. CMOS has been the driving force behind high-performance ICs. The attractive scaling properties of CMOS, coupled with low power, high speed and noise margins, reliability, wider temperature and voltage operation range, overall circuit and layout implementation and manufacturing ease, has made it the technology of choice for digital ICs. CMOS also enables monolithic integration of both analog and digital circuits on the same die, and shows great promise for future System-on-a-Chip (SoC) implementations.

This chapter reviews trends in Silicon CMOS technology and highlights the main challenges involved in keeping up with Moore’s law. CMOS device
scaling increases transistor sub-threshold leakage. Interconnect scaling coupled with higher operating frequencies requires careful parasitic extraction and modeling. Power delivery and dissipation are fast becoming limiting factors in product design.

1.1 The Microelectronics Era and Moore’s Law

There is hardly any other area of industry which has developed as fast as the semiconductor industry in the last 40 years. Within this relatively short period, microelectronics has become the key technology enabler for several industries like information technology, telecommunication, medical equipment and consumer electronics [1].

In 1965, Intel co-founder Gordon Moore observed that the total number of devices on a chip doubled every 12 months [2]. He predicted that the trend would continue in the 1970s but would slow in the 1980s, when the total number of devices would double every 24 months. Known widely as “Moore’s Law,” these observations made the case for continued wafer and die size growth, defect density reduction, and increased transistor density as manufacturing matured and technology scaled. Figure 1-1 plots the growth number of transistors per IC and shows that the transistor count has indeed doubled every 24 months [3].

During these years, the die size increased at 7% per year, while the operating frequency of leading microprocessors has doubled every 24 months [4], and is
1.2 Low Power CMOS Technology

The idea of CMOS Field Effect Transistor (FET) was first introduced by Wanlass and Sah [5]. By the 1980’s, it was widely acknowledged that CMOS is the dominant technology for microprocessors, memories and application specific integrated circuits (ASICs), owing to its favorable properties over other IC technologies. The biggest advantage of CMOS over NMOS and bipolar technology is its significantly reduced power dissipation, since a CMOS circuit has almost no static (DC) power dissipation.

1.2.1 CMOS Power Components

In order to understand the evolution of CMOS as one of the most popular low-power design approaches, we first examine the sources of power dissipation in digital CMOS circuit. The total power consumed by a static CMOS circuit consists of three components, and is given by the following expression [6]:

\[ P_{\text{total}} = P_{\text{dynamic}} + P_{\text{short-circuit}} + P_{\text{static}} \]  \hspace{1cm} (1.1)

\( P_{\text{dynamic}} \) represents the dynamic or switching power, i.e., the power dissipated in charging and discharging the physical load capacitance contributed by fan-out gate loading, interconnect loading, and parasitic capacitances at the CMOS gate outputs. \( C_L \) represents this capacitance, lumped together as shown in Figure 1-2. The dynamic power is given by Eq. 1.2, where \( f_{\text{clk}} \) is the clock frequency with which the gate switches, \( V_{dd} \) is the power supply, and \( \alpha \) is the switching activity factor, which determines how frequently the output switches per clock-cycle.

\[ P_{\text{dynamic}} = \alpha \cdot f_{\text{clk}} \cdot C_L \cdot V_{dd}^2 \]  \hspace{1cm} (1.2)

\( P_{\text{short-circuit}} \) represents the short-circuit power, i.e., the power consumed during switching between \( V_{dd} \) and ground. This short-circuit current \( (I_{sc}) \) arises when both PMOS and NMOS transistors are simultaneously active, conducting current directly from \( V_{dd} \) to ground for a short period of time. Equation 1.3 describes the short-circuit power dissipation for a simple CMOS inverter [7], where \( \beta \) is the gain factor of the transistors, \( \tau \) is the input rise/fall time, and \( V_T \) is the transistor threshold voltage (assumed to be same for both PMOS/NMOS).

\[ P_{\text{short-circuit}} = \frac{\beta}{12} \left( V_{dd} - 2V_T \right)^3 \cdot \tau \cdot f_{\text{clk}} \]  \hspace{1cm} (1.3)
Figure 1-2: Basic CMOS gate showing dynamic and short-circuit currents.

It's important to note that the switching component of power is independent of the rise/fall times at the input of logic gates, but short-circuit power depends on input signal slope. Short-circuit currents can be significant when the rise/fall times at the input of the gate is much longer when compared to the output rise/fall times.

The third component of power: $P_{static}$ is due to leakage currents and is determined by fabrication technology considerations, and consists of (1) source/drain junction leakage current; (2) gate direct tunneling leakage; (3) sub-threshold leakage through the channel of an OFF transistor and is summarized in Figure 1-3.

(1) The junction leakage ($I_1$) occurs from the source or drain to the substrate through the reverse-biased diodes when a transistor is OFF. The magnitude of the diode’s leakage current depends on the area of the drain diffusion and the leakage current density, which, in turn, determined by the process technology.

(2) The gate direct tunneling leakage ($I_2$) flows from the gate thru the “leaky” oxide insulation to the substrate. Its magnitude increases exponentially with the gate oxide thickness and supply voltage. According to the 2005 International Technology Roadmap for Semiconductors [8], high-$K$ gate dielectric is required to control this direct tunneling current component of the leakage current.

(3) The sub-threshold current is the drain-source current ($I_3$) of an OFF transistor. This is due to the diffusion current of the minority carriers in the channel for a MOS device operating in the weak inversion mode (i.e., the sub-threshold region.) For instance, in the case of an inverter with a low input voltage, the NMOS is turned OFF and the output voltage is high. Even with $V_{GS}$ at 0V, there is still a current flowing in the channel of the OFF NMOS transistor since $V_{GS} = V_{dd}$. The magnitude of the sub-threshold current is a function of the temperature, supply voltage, device size, and the process parameters, out of
which, the threshold voltage \( V_T \) plays a dominant role. In today’s CMOS technologies, sub-threshold current is the largest of all the leakage currents, and can be computed using the following expression [9]:

\[
I_{\text{SUB}} = K \cdot \left( 1 - e^{-v_{DS}/v_T} \right) \cdot e^{V_{GS}-V_T+\eta v_{DS}/nv_T}
\]  

(1.4)

where \( K \) and \( n \) are functions of the technology, and \( \eta \) is drain induced barrier lowering (DIBL) coefficient, an effect that manifests short channel MOSFET devices. The transistor sub-threshold swing coefficient is given by \( n \) and \( v_T \) is the thermal voltage given by \( KT/q \approx 33\text{mV} \) at \( 110^\circ C \).

![MOS transistor diagram](image)

**Figure 1-3: Main leakage components for a MOS transistor.**

In the sub-threshold region, the MOSFET behaves primarily as a bipolar transistor, and the sub-threshold is exponentially dependent on \( V_{GS} \) (Eq. 1.4). Another important figure of merit for low-power CMOS is the sub-threshold slope, which is the amount of voltage required to drop the sub-threshold current by one decade. The lower the sub-threshold slope, the better since the transistor can be turned OFF when \( V_{GS} \) is reduced below \( V_T \).

### 1.3 Technology scaling trends and challenges

The scaling of CMOS VLSI technology is driven by Moore’s law, and is the primary factor driving speed and performance improvement of both microprocessors and memories. The term “scaling” refers to the reduction in transistor width, length and oxide dimensions by 30%. Historically, CMOS technology, when scaled to the next generation, (1) reduces gate delay by 30% allowing a 43% increase in clock frequency, (2) doubles the device density (as shown in Figure 1-1), (3) reduces the parasitic capacitance by 30% and (4) reduces the energy and active energy per transition by 65% and 50%
respectively. The key barriers to continued scaling of supply voltage and technology for microprocessors to achieve low-power and high-performance has been well documented in [10]-[13].

1.3.1 Technology scaling: Impact on Power

The rapid increase in the number of transistors on chips has enabled a dramatic increase in the performance of computing systems. However, the performance improvement has been accompanied by an increase in power dissipation; thus, requiring more expensive packaging and cooling technology. Figure 1-4 shows the power trends of Intel microprocessors, with the dotted line showing the power trend with classic scaling. Historically, the primary contributor to power dissipation in CMOS circuits has been the charging and discharging of load capacitances, often referred to as the dynamic power dissipation. As shown by Eq. 1.2, this component of power varies as the square of the supply voltage ($V_{dd}$). Therefore, in the past, chip designers have relied on scaling down $V_{dd}$ to reduce the dynamic power dissipation. Despite scaling $V_{dd}$, the total power dissipation of microprocessors is expected to increase exponentially (in logarithmic scale) from 100W in 2005 to over 2KW by end of the decade, with a substantial increase in sub-threshold leakage power [10].

![Figure 1-4: Power trends for Intel microprocessors.](image)

Maintaining transistor switching speeds (constant electric field scaling) requires a proportionate downscaling of the transistor threshold voltage ($V_T$) in lock step with the $V_{dd}$ reduction. However, scaling $V_T$ results in a significant increase of leakage power due to an exponential increase in the sub-threshold
leakage current (Eq. 1.4). Figure 1-5 illustrates the increasing leakage power trend for Intel microprocessors in different CMOS technologies; with leakage power exceeding 50% of the total power budget in 70nm technology node [13]. Figure 1-6 shows estimated sub-threshold leakage currents for future technologies as a function of temperature. Borkar in [11] predicts a 7.5X increase in the leakage current and a 5X increase in total energy dissipation for every new microprocessor chip generation!

![Figure 1-5: Dynamic and Static power trends for Intel microprocessors](image)

![Figure 1-6: Sub-threshold leakage current as a function of temperature.](image)

Note that it is possible to substantially reduce the leakage power, and hence the overall power, by reducing the die temperature. Therefore better cooling
techniques would be more critical in the advanced deep submicron technologies to control both active leakage and total power. Leakage current reduction can also be achieved by utilizing either process techniques and/or circuit techniques. At process level, channel engineering is used to optimize the device doping profile for reduced leakage. The dual-$V_T$ technology [14] is used to reduce total sub-threshold leakage power, where NMOS and PMOS devices with both high and low threshold voltages are made by selectively adjusting well doses. The technique adjusts high performance critical path transistors with low-$V_T$ while non-critical paths are implemented with high-$V_T$ transistors, at the cost of additional process complexity. Over 80% reduction in leakage power has been reported in [15], while meeting performance goals.

In the last decade, a number of circuit solutions for leakage control have been proposed. One solution is to force a non-stack device to a stack of two devices without affecting the input load [16], as shown in Figure 1-7(a). This significantly reduces sub-threshold leakage, but will incur a delay penalty, similar to replacing a low-$V_T$ device with a high-$V_T$ device in a dual-$V_T$ design. Combined dynamic body bias and sleep transistor techniques for active leakage power control have been described in [17]. Sub-threshold leakage can be reduced by dynamically changing the body bias applied to the block, as shown in Figure 1-7(b). During active mode, forward body bias (FBB) is applied to increase the operating frequency. When the block enters idle mode, the forward bias is withdrawn, reducing the leakage. Alternately, reverse body bias (RBB) can be applied during idle mode for further leakage savings.

![Figure 1-7: Three circuit level leakage reduction techniques.](image_url)

Supply gating or sleep transistor technique uses a high threshold transistor, as shown in Figure 1-7(c), to cut off the supply to a functional block, when the
1.3 Technology scaling trends and challenges

A design is in an “idle” or “standby” state. A 37X reduction in leakage power, with a block reactivation time of less than two clock cycles has been reported in [17]. While this technique can reduce leakage by orders of magnitude, it causes performance degradation and complicates power grid routing.

1.3.2 Technology scaling: Impact on Interconnects

In deep-submicron designs, chip performance is increasingly limited by the interconnect delay. With scaling the width and thickness of the interconnections are reduced. As a result, the resistance increases, and as the interconnections get closer, the capacitance increases, increasing RC delay. As a result, the cross-coupling capacitance between adjacent wires is also increasing with each technology generation [18]. New designs add more transistors on chip and the average die size of a chip has been increasing over time. To account for increased RC parasitics, more interconnect layers are being added. The thinner, tighter interconnect layers are used for local interconnections, and the new thicker and sparser layers are used for global interconnections and power distribution. Copper metallization has been used to reduce the resistance of interconnects and Fluorinated SiO\(_2\) as inter-level dielectric (ILD) to reduce the dielectric constant (k=3.6). Figure 1-8 is a cross-section SEM image showing the interconnect structure in the 65nm technology node [19].

![Figure 1-8](image)

**Figure 1-8: Intel 65 nm, 8-metal copper CMOS technology in 2004.**

Inductive noise and skin effect will get more pronounced as frequencies are reaching multi-GHz levels. Both circuit and layout solutions will be required to contain the inductive and capacitive coupling effects. In addition, Interconnects now dissipate an increasingly larger portion of total chip power [20]. All of these trends indicate that interconnect delays and power in sub-65nm technologies will continue to dominate the overall chip performance.
1.3 Technology scaling trends and challenges

To mitigate the global interconnect problem, new structured on-chip communication fabrics, called networks on chips (NoCs), have emerged for use in SoC designs. The basic concept is to replace today’s shared buses with on-chip packet-switched interconnection networks [21]. The NoC architecture, shown in Figure 1-9, consists of the basic building block, the “network tile”. These tiles are connected to an on-chip network that routes packets between them. Each tile may consist of one or more compute cores (microprocessor) or memory cores and would also have routing logic responsible for routing and forwarding the packets, based on the routing policy of the network. The structured network wiring of such a NoC design gives well-controlled electrical parameters that simplifies timing and allows the use of high-performance circuits to reduce latency and increase bandwidth.

![Network Tile Diagram](image)

Figure 1-9: NoC Architecture.

1.3.3 Technology scaling: Summary

We discussed trends in CMOS VLSI technology. The data indicates that silicon performance, integration density, and power have followed the scaling theory. The main challenges to VLSI scaling include lithography, transistor scaling, interconnect scaling, and increasing power delivery and dissipation. Modular on-chip networks are required to resolve interconnect scaling issues. As the MOSFET channel length is reduced to 45nm and below, suppression of the ever increasing off-state leakage currents becomes crucial, requiring leakage-aware circuit techniques and newer bulk CMOS compatible device structures. While these challenges remain to be overcome, no fundamental barrier exists to scaling CMOS devices well into the nano-CMOS era, with a physical gate length of under 10nm. Predictions from the 2005 IRTS roadmap [8] indicate that “Moore’s law” should continue well into the next decade before the ultimate device limits for CMOS are reached.
1.4 Motivation and Scope of this Thesis

Very recently, chip designers have moved from a computation-centric view of chip design to a communication-centric view, owing to the relative importance of interconnect delay over gate delay in deep-submicron technologies. As this discrepancy between device delay and on-chip communication latency becomes macroscopic, the practical solution is to use scalable NoC architectures. To address the need for substantial interconnect bandwidth, packet switched routers are fast replacing shared buses and dedicated wires as the interconnect fabric of choice. With on-chip communication consuming a significant portion of the chip power (about 40% [22]) and area budgets, there is a compelling need for compact, low power routers. While applications dictate the choice of the compute core, the advent of multimedia applications, such as 3D graphics and signal processing, places stronger demands for self-contained, low-latency floating-point processors with increased throughput. Therefore, this work focuses on two key building blocks critical to the success of NoC design: high performance, area and energy efficient router and floating-point processor architectures.

This thesis first presents a six-port four-lane 57GB/s non-blocking router core based on wormhole switching. In this design, we opportunistically leverage available data timing slack in the crossbar switch by double-pumping the data buses, significantly reducing router area. The router features double-pumped crossbar channels and destination-aware channel drivers that dynamically configure based on the current packet destination. This enables 45% reduction in crossbar channel area, 23% overall router area, up to 3.8X reduction in peak channel power, and 7.2% improvement in average channel power, with no performance penalty over a published design. In a 150nm six-metal CMOS process, the 12.2mm² router contains 1.9 million transistors and operates at 1GHz at 1.2V.

We next present a new pipelined single-precision floating-point multiply accumulator core (FPMAC) featuring a single-cycle accumulate loop using base 32 and internal carry-save arithmetic, with delayed addition techniques. Combined algorithmic, logic and circuit techniques enable multiply-accumulates at speeds exceeding 3GHz, with single-cycle throughput. In this design, we successfully obtain higher frequency and performance by increased parallelism, and at the expense of increased area. Unlike existing FPMAC architectures, the design eliminates scheduling restrictions between consecutive FPMAC instructions. The optimizations allow removal of the costly normalization step from the critical accumulate loop and conditionally powered down using dynamic sleep transistors on long accumulate operations, saving active and
leakage power. In addition, an improved leading zero anticipator (LZA) and overflow detection logic applicable to carry-save format is presented. In a 90nm seven-metal dual-$V_T$ CMOS process, the 2mm$^2$ custom design contains 230K transistors. The fully functional first silicon achieves 6.2 GFLOPS of performance while dissipating 1.2W at 3.1GHz, 1.3V supply.

It is clear that realization of successful NoC designs require well balanced trade offs at all levels: architecture, logic, circuit and physical design. Our results from key building blocks demonstrate the feasibility of pushing the performance limits of compute cores and communication routers, while keeping active and leakage power, and area under control.

1.5 Organization of this Thesis

This thesis is organized into three parts:

- Part I - Introduction
- Part II - NoC Building Blocks
- Part III - Papers

Part I, provides the necessary background for the concepts used in the papers. This chapter reviews trends in Silicon CMOS technology and highlights the main challenges involved in keeping up with Moore’s law. Properties of CMOS devices, sources of leakage power and the impact of scaling on power and interconnect are discussed. This is followed by a brief discussion of NoC architectures.

In Part II, we describe the two NoC building blocks in detail. Introductory concepts specific to on-die interconnection networks, including a generic router architecture is presented in Chapter 2. A more detailed description of the six-port four-lane 57 GB/s non-blocking router core design (Paper 1) is also given. Chapter 3 presents basic concepts involved in floating-point arithmetic, reviews conventional floating-point units (FPU), and describes the challenges in accomplishing single-cycle accumulate on today’s FPUs. A new pipelined single-precision FPMAC design, capable of single-cycle accumulation, is described in Paper 2 and Paper 3. Paper 2 first presents the FPMAC algorithm, logic optimizations and preliminary chip simulation results. Paper 3 introduces the concept of “Conditional Normalization”, applicable to floating point units, for the first time. It also describes a 6.2 GFLOPS Floating Point Multiply-Accumulator, enhanced with conditional normalization pipeline and with detailed results from silicon measurement.

Finally in Part III the papers, included in this thesis, are presented in full.
### 1.6 References


Part II

NoC Building Blocks
Chapter 2

On-Chip Interconnection Networks

As VLSI technology scales, and processing power continues to improve, inter-processor communication becomes a performance bottleneck. On-chip networks have been widely proposed as the interconnect fabric for high-performance SoCs [1] and the benefits demonstrated in several chip multiprocessors (CMPs) [2]–[3]. Recently, NoC architectures are emerging as the candidate for highly scalable, reliable, and modular on-chip communication infrastructure platform. The NoC architecture uses layered protocols and packet-switched networks which consist of on-chip routers, links, and well defined network interfaces. With the increasing demand for interconnect bandwidth, on-chip networks are taking up a substantial portion of system power budget. A case in point: the MIT Raw [2] on-chip network which connects 16 tiles of processing elements consumes 36% of total chip power, with each router dissipating 40% of individual tile power. The routers and the links of the Alpha 21364 microprocessor consume about 20% of the total chip power [4]. These numbers indicate the significance of managing the interconnect power consumption. In addition, any NoC architecture should fit into the limited silicon budget with an optimal choice of NoC fabric topology providing high bisection bandwidth, efficient routing algorithms and compact low-power router implementations.
This chapter first presents introductory concepts specific to on-chip networks. Topics include network topologies, crossbar switches, and message switching techniques. The internal functional blocks for a canonical router architecture are also described. A more detailed description of a six-port four-lane 57 GB/s non-blocking router core design (Paper 1), with area and peak-energy benefits, is also presented.

2.1 Interconnection Network Fundamentals

An interconnection network consists of nodes (or routers) and links (or fabric), and can be broadly classified into direct or indirect networks [5]. A direct network consists of a set of nodes, each one being directly connected to a (usually small) subset of other nodes in the network. A common component of each node in such a network is a dedicated router, which handles all message communication between nodes, as shown in Figure 2-1. Usually two neighboring nodes are connected by a pair of uni-directional link in opposite directions. As the number of nodes in the system increases, the total communication bandwidth also increases. This excellent scaling property has made direct networks very popular in constructing large-scale NoC designs. The MIT Raw [2] design is an example of direct network. In an indirect network, communication between any two nodes must go through a set of switches.

![Figure 2-1: Direct and indirect networks](image)

All modern day on-chip networks are buffered, i.e., the routers contain storage for buffering messages when they are unable to obtain an outgoing link.
An interconnection network can be defined by four parameters: (1) its topology (2) the routing algorithm governing it, (3) the message switching protocol, and (4) the router micro-architecture.

### 2.1.1 Network Topology

The topology of a network concerns how the nodes and links are connected. A topology dictates the number of alternate paths between nodes, and thus how well the network can handle contention and different traffic patterns. Scalability is an important factor in the selection of the right topology for on-chip networks. Figure 2-2 shows various popular on topologies which have been commercially adopted [5].

Shared bus networks (Figure 2-2a) are the simplest, consist of a shared link common to all nodes. All nodes have to compete for exclusive access to the bus. While simple, bus-based systems scale very poorly as more nodes are added. In a ring (or 1D torus) every node has exactly two neighbors and allows more concurrent transfers. Since all nodes are not directly connected, messages will have to hop along intermediate nodes until they arrive at the final destination. This causes the ring to saturate at a lower network throughput for most traffic patterns. The 2D mesh/torus, crossbar and hypercube topologies are examples of direct networks, and thus provide tremendous improvement in performance, but at a cost, typically increasing as the square of the number of nodes.

![Figure 2-2: Popular on-chip fabric topologies](image)

A crossbar (Figure 2-2e) is a *fully connected* topology, i.e., the interconnection allows any node to directly communicate with any other node.
A crossbar network example connecting $p$ processors to $b$ memory banks is shown in Figure 2-3. Note that it is a *non-blocking* network, i.e., a connection of one processor to a given memory bank *does not block* a connection of another processor to a different memory bank.

The above crossbar uses $p \times b$ switches. The hardware cost for crossbar is high, at least $O(p^2)$, if $b = p$. An important observation is that the crossbar is not very scalable and cost increases quadratically, as more nodes are added.

### 2.1.2 Message Switching Protocols

The message switching protocol determines when a message gets buffered, and when it gets to continue. The goal is to effectively share the network resources among the many messages traversing the network, so as to approach the best latency-throughput performance.

1. **Circuit vs. packet switching:** In circuit switching a physical path from source to destination is reserved prior to the transmission of data, as in telephone networks. The complete message is transmitted along the pre-reserved path that is established. While circuit switching reduces network latency, it does so at the expense of network throughput. Alternatively, a message can be decomposed into packets which share channels with other packets. The first few bytes of a packet, called the *packet header*, contain routing and control information. Packet
switching improves channel utilization and extends network throughput. Packet-based flow control methods dominate interconnection networks today because in interconnection networks, buffers and channels are limited resources that need to be efficiently allocated.

2) **Virtual Cut-Through (VCT) Switching:** To reduce packet delay at each hop, virtual cut-through flow switching allows transmission of a packet to begin before the entire packet is received. Latency experienced by a packet is thus drastically reduced. However, bandwidth and storage are still allocated in packet-sized units. Packets can move forward only if there is enough storage to hold the entire packet. Examples of routers adopting VCT switching are the IBM SP/2 switch [6] and Mercury router described in [7].

3) **Wormhole Switching:** In a wormhole routing scheme, a packet is divided into “FLIT’s” or “FLow control unIT’s”, for transmission. A FLIT is the minimum divisible amount of data transmitted within a packet that the control logic in the router can process. As the header flit containing routing information advances along the network, the message flits follow in a pipelined manner. The lower packet latency and greater throughput of wormhole routers increases the efficiency of inter-processor communication. In addition, wormhole routers have substantially reduced buffering requirements [8], thus enabling small, compact and fast router designs. The simplicity, low cost, and distance-insensitivity of wormhole switching are the main reasons behind its wide acceptance by manufacturers of commercial parallel machines. Several high-bandwidth low-latency routers based on wormhole switching have been presented in [8]–[10].

2.1.3 **Virtual Lanes**

Virtual-lanes (or logical channels) [11] improves upon the channel utilization of wormhole flow control, allowing blocked packets to be passed by other packets. This is accomplished by associating several virtual channels, each with a separate flit queue, with each physical channel. Virtual channels arbitrate for physical channel bandwidth on a flit-by-flit basis. When a packet holding a virtual channel gets blocked, other packets can still traverse the physical channel through other virtual channels. Virtual lanes were originally introduced to solve the deadlock avoidance problem, but they can be also used to improve network latency and throughput, as illustrated in Figure 2-4. Assume \( P_0 \) arrived earlier and acquired the channel between the two routers first. In absence of virtual channels, packet \( P_1 \) arriving later would be blocked until the transmission of \( P_0 \) has been completed. Assume that the physical channels implement two virtual channels. Upon arrival of \( P_1 \), the physical channel is multiplexed between them on a flit-by-flit basis and both packets proceed with half speed, and both messages continue to make progress. In effect, virtual lanes decouples the
physical channels from message buffers allowing multiple messages to share the same physical channel in the same manner that multiple programs may share a central processing unit (CPU).

![Figure 2-4: Reduction in header blocking delay using virtual channels.](image)

On the downside, the use of virtual channels, while reducing header blocking delays in the network, increases design complexity of the link controller and flow control mechanisms.

### 2.1.4 A Generic Router Architecture

Router architecture is largely determined by the choice of switching technique. Most modern routers used in high performance multiprocessor architecture utilize packet switching or cut-through switching techniques (wormhole and virtual cut-through) with either a deterministic or an adaptive routing algorithm. Several high-bandwidth low-latency routers have been designed [12]–[14]. A generic wormhole switched router (Figure 2-5) has the following key components [5]:

1. **Link Controller (LC).** This block implements flow control across the physical channel between adjacent routers. The link controllers on either side of the link coordinate to transfer flow control units. In the presence of virtual channels, this unit also decodes the destination channel of the received flit.

2. **Buffers.** These are first-in-first out (FIFO) buffers for storing packets in transit and are associated with input and output physical links. Routers may have buffers only on inputs (input buffered) or outputs (output buffered). Buffer size is critical and must account for link delays in propagation of data and flow control signals.

3. **Crossbar switch.** This component is responsible for connecting router input buffers to the output buffers. High-speed routers utilize crossbar networks with full connectivity, enabling multiple and simultaneous message transfers. In this thesis, we also use the term *crossbar channel* to represent the crossbar switch.
4. **Routing and arbitration unit.** This unit implements the routing function and the task of selecting an output link for an incoming message. Conflicts for the same output link must be arbitrated. Fast arbitration policies are crucial to maintaining a low latency through the switch. This unit may also be replicated to expedite arbitration delay.

![Figure 2-5: Canonical router architecture.](image)

5. **Virtual channel controller (VC).** This unit is responsible for multiplexing the contents of the virtual channels onto the physical links. Several efficient arbitration schemes have been proposed [15].

6. **Processor Interface.** This block implements a physical link interface to the processor and contains one or more injection or ejection channels to the processor.

### 2.2 A Six-Port 57GB/s Crossbar Router Core

We now describe a six-port four-lane 57GB/s router core that features double-pumped crossbar channels and destination-aware channel drivers that dynamically configure based on the current packet destination. This enables 45% reduction in channel area, 23% overall chip area, up to 3.8X reduction in peak channel power, and 7.2% improvement in average channel power, with no performance penalty. In a 150nm six-metal CMOS process, the 12.2mm² core contains 1.9 million transistors and operates at 1GHz at 1.2V.
2.2.1 Introduction

As described in Section 2.1.2, crossbar routers, based on wormhole switching are a popular choice to interconnect large multiprocessor systems. Although small crossbar routers are easily implemented on a single chip, area limitations have constrained cost-effective single chip realization of larger crossbars with bigger data path widths [16]–[17], since switch area increases as a square function of the total number of I/O ports and number of bits per port. In addition, larger crossbars exacerbate the amount of on-chip simultaneous-switching noise [18], because of increased number of data drivers. This work focuses on the design and implementation of an area and peak power efficient multiprocessor communication router core.

The rest of this chapter is organized as follows. The first three sections (2.2.2–2.2.4) present the router architecture including the core pipeline, packet structure, routing protocol and flow control descriptions. Section 2.2.5 describes the design and layout challenges faced during the implementation of an earlier version of the crossbar router, which forms the motivation for this work. Section 2.2.6 presents two specific design enhancements to the crossbar router. Section 2.2.7 describes the double-pumped crossbar channel, and its operation. Section 2.2.8 explains details of the location-based channel drivers used in the crossbar. These drivers are destination-aware and have the ability to dynamically configure based on the current packet destination. Section 2.2.9 presents chip results and compares this work with prior crossbar work in terms of area, performance and energy. We conclude in Section 2.3 with some final remarks and directions for future research.

2.2.2 Router Micro-architecture

The architecture of the crossbar router is shown in Figure 2-6, which is based on the router described in [19], and forms the reference design for this work. The design is a wormhole switched, input-buffered, fully non-blocking router. Inbound packets at any of the six input ports can be routed to any output ports simultaneously. The design uses vector-based routing. Each time a packet enters a crossbar port, a 3-bit destination ID (DID) field at the head of the packet determines the exit port. If more than one input contends for the same output port, then arbitration occurs to determine the binding sequence of the input port to an output. To support deadlock-free routing, the design implements four logical lanes. Notice that each physical port is connected to four identical lanes (0–3). These logical lanes, commonly referred to as “virtual channels”, allow the physical ports to accommodate packets from four independent data streams simultaneously.
Data flow through the crossbar switch is as follows: data input is first buffered into the queues. Each queue is basically a first-in-first out (FIFO) buffer. Data coming out of the queues is examined by routing and sequencing logic to generate output binding requests based on packet header information. With six ports and 4 lanes, the crossbar switch requires a 24-to-1 arbiter and is implemented in 2 stages, a 6-to-1 port Arbiter followed by a 4-to-1 lane Arbiter. The first stage of arbitration within a particular lane essentially binds one input port to an output port, for the entire duration of the packet. If other packets in different lanes are contending for the same physical output resource, then a second level of arbitration occurs. This level of arbitration occurs at a FLIT level, a finer level of granularity than packet boundaries. It should be noted that both levels of arbitration are accomplished using a balanced, non-weighted, round-robin algorithm.

Figure 2-7 shows the 6-stage data and control pipelines, including key signals used in the router crossbar switch. The pipe stages use rising edge-triggered sequential devices. Stages one, two and three are used for the queue. Stage four, for the route decoding and sequencing logic. Stage five, for data in the port multiplexer and port arbitration. Stage six, for data in the lane multiplexer and lane arbitration. The shaded portion between stages four and five represents the primary physical crossbar routing channel. Here, data and control buses are connected between the six ports within a lane. The layout area of this routing
channel increases as a square function of the total number of ports and number of bits per port.

Figure 2-7: Router data and control pipeline diagram.

The shaded portion between stages five and six represents a second physical routing channel. In this case, data and control buses are connected between the four corresponding lanes. The total die area of this routing channel increases only linearly as a function to the total number of bits per port. These two main routing channels run perpendicular to each other on the die. For high performance and symmetric layout, this basic unit of logic in Fig.2 is duplicated 24 times throughout the crossbar, six within each lane and across four lanes.

2.2.3 Packet structure and routing protocol

Figure 2-8 shows the packet format used in the router. Each packet is subdivided into FLITs. Each FLIT contains six control bits and 72 data bits. The control bits are made up of packet header (H), valid (V) and tail (T) bits, two encoded lane ID bits which indicate one of four lanes, and flow control (FC) information. The flow control bits contain receiver queue status and helps create “back pressure” in the network to help prevent queue overflows.

The crossbar uses vector based routing. Data fields in the “header” flit of a packet are reserved for routing destination information, required for each hop. A “hop” is defined as a flit entering and leaving a router. The router uses the 3-bit destination ID [DID] field as part of the header to determine the exit port. The
least significant 3 bits of the header indicates the current destination ID and is updated by shifting out the current DID after each hop. Each header flit supports a maximum of 24 hops. The minimum packet size the protocol allows for is two flits. This was chosen to simplify the sequencer control logic and allow back-to-back packets to be passed through the router core without having to insert idle flits between them. The router architecture places no restriction on the maximum packet size.

![Router packet and FLIT format.](image)

2.2.4 FIFO buffer and flow control

The queue acts as a FIFO buffer and is implemented as a large signal 1-read 1-write register file. Each queue is 76 bits wide and 64 flits deep. There are four separate queues [20]–[21] per physical port to support virtual channels. Reads and writes to two different locations in the queue occur simultaneously in a single clock cycle. The circuits for reading and writing are implemented in a single-ended fashion. Read sensing is implemented using a two level local and global bit-line approach. This approach helps to reduce overall bit-line loading as well as speeding up address decoding. In addition, data-dependent, self-timed pre-charge circuits are also used to lower both clock and total power.

Flow control and buffer management between routers is debit-based using almost-full bits (Figure 2-9), which the receiver FIFO signals via the flow control bits (FC) bits, when its buffers reach a specified threshold. This mechanism is chosen over conventional credit and counter based flow control schemes for improved reliability and recovery in the presence of signaling errors. The protocol is designed to allow all four receiving queues across four lanes to independently signal the sender when they are almost full regardless of the data traffic at that port. The FIFO buffer almost-full threshold can be programmed via software.


2.2.5 Router Design Challenges

This work was motivated to address the design challenges and issues faced while implementing an earlier version of the router built in six-metal, 180nm technology node \((L_{\text{eff}} = 150\text{nm})\) and reported in [19]. Figure 2-10(a) shows the internal organization of the six ports within each lane. We focused our attention on the physical crossbar channel between pipe stages four and five where data and control busses are connected between the six ports. The corresponding reference layout for one (of four lanes) is given in Figure 2-10(b). The crossbar channel interconnect structure used in the original full custom design is also shown in Figure 2-10(c). Each wire in the channel is routed using metal 3 (vertical) and metal 4 (horizontal) layers with a 1.1\(\mu\text{m}\) pitch.

![Diagram](image)

**Figure 2-10:** Crossbar router in [19] (a) Internal lane organization. (b) Corresponding layout. (c) Channel interconnect structure.

In each lane, 494 signals (456 data, 38 control) traverse the 3.3mm long wire-dominated channel. More importantly, the crossbar channel consumed 53% of
core area and presented a significant physical design challenge. To meet GHz operation, the entire core used hand-optimized data path macros. In addition, the design used large bus drivers to forward data across the crossbar channel, sized to meet the worst-case routing and timing requirements and replicated at each port. Over 1800 such channel drivers were used across all four lanes in the router core. When a significant number of these drivers switch simultaneously, it places a substantial transient current demand on the power distribution system. The magnitude of the simultaneous-switching noise spike can be easily computed using the expression:

$$\Delta V = L M \frac{\Delta I}{\Delta t}$$

Where $L$ is the effective inductance, $M$ the number of drivers active simultaneously and $\Delta I$ the current switched by each driver over the transition interval $\Delta t$. For example, with $M = 200$, a power grid with a parasitic inductance of 1nH, and a current change of 2mA per driver with an overly conservative edge rate of half a nanosecond, the $\Delta V$ noise is approximately 0.8 volts. This peak noise is significant in today’s deep sub-micron CMOS circuits and causes severe power supply fluctuations and possible logic malfunction. It is important to note that the amount of peak switching noise, while data activity dependent, is independent of router traffic patterns and the physical spatial distance of an outbound port from an input port. Several techniques have been proposed to reduce this switching noise [17]–[18], but the work has not been addressed in the context of packet-switched crossbar routers.

In this work, we present two specific design enhancements to the crossbar channel and a more detailed description of the techniques described in Paper 1. To mitigate crossbar channel routing area, we propose double-pumping the channel data. To reduce peak power, we propose a location based channel driver (LBD) which adjusts driver strength as a function of current packet destination. Reducing peak currents has the benefit of reduced demand on the power grid and decoupling capacitance area, which in-turn results in lower gate leakage.

**2.2.6 Double-pumped Crossbar Channel**

Now we describe design details of the double pumped crossbar. The crossbar core is a fully synchronous design. A full clock cycle is allocated for communication between stages 4 and 5 of the core pipeline. As shown in Figure 2-11, one clock phase is allocated for data propagation across the channel. At pipe-stage 4, the 76-bit data bus is double-pumped by interleaving alternate data bits using dual edge -triggered flip-flops. The master latch M0 for data input (i0) is combined with the slave latch S1 for data input (i1) and so on. The slave latch S0 is moved across the crossbar channel to the receiver. A 2:1 multiplexer,
enabled using clock is used to select between the latched output data on each clock phase.

Figure 2-11: Double-pumped crossbar channel.

This double pumping effectively cuts the number of channel data wires in each lane by 50% from 456 to 228 and in all four logical lanes from 1824 to 912 nets. There is additional area savings due to a 50% reduction in the number of channel data drivers. To avoid timing impact, the 38 control (request + grant) wires in the channel are left untouched.

2.2.7 Channel operation and timing

The timing diagram in Figure 2-12 summarizes the communication of data bits D[0-3] across the double-pumped channel. The signal values in the diagram match the corresponding node names in the schematic in Figure 2-11. To transmit, the diagram shows in cycle T2 and T3, master latch M0 latching input data bits D[0] and D[2], while the slave latch S1 retains bits D[1] and D[3]. The 2:1 multiplexer, enabled by clock is used to select between the latched output data on each clock phase, thus effectively double-pumping the data. A large driver is used to forward the data across the crossbar channel.

At the receiving end of the crossbar channel, the delayed double-pumped data is first latched by slave latch (S0), which retains bits D[0] & D[2] prior to capture by pipe stage 5. Data bits D[1] and D[3] are easily extracted using a rising edge flip-flop. Note that the overall data delay from Stage 4 to Stage 5 of the pipeline is still one cycle, and there is no additional latency impact due to double pumping. This technique, however, requires a close to 50% duty cycle requirement on the clock generation and distribution.
2.2.8 Location-based Channel Driver

To reduce peak power, the crossbar channel uses location based drivers as a drop-in replacement to conventional bus drivers. Figure 2-13 shows the LBD driver and the associated control logic highlighted. A 3-bit unique hard-wired location ID (LID) is assigned to provide information about the spatial location of the driver in the floor-plan, at a port level granularity. By comparing LID bits with the DID bits in the header flit, an indication of the receiver distance from the driver is obtained. A 3-bit subtractor accomplishes this task. The subtraction result is encoded to control the location based drivers. To minimize delay penalty to the crossbar channel data, the encoder output control signals (en[2:0]) are re-timed to pipe stage 4.

The LBD schematic Figure 2-14(a) shows a legged driver with individual enable controls for each leg, which is binary weighted. The LBD encodings are carefully chosen to allow full range of driver sizes and communication distances, without exceeding the path data delay budget.
Figure 2-13: Location-based channel driver and control logic.

Figure 2-14: (a) LBD Schematic. (b) LBD encoding summary.

LBD encoding is summarized in Figure 2-14(b), and is chosen in such a way that the smallest leg of the driver is turned on when a packet is headed to its own port, a feature called “loopback”. A larger leg is on if the packet is destined to the adjacent ports. All legs are enabled only for the farthest routable distances in.
the channel, specifically ports \(0 \rightarrow 5\) or \(5 \rightarrow 0\). Hard-wiring of the LID bits enables a modular design and layout that is reusable at each port. It is important to note that the LBD enable bits change only at packet boundaries and not at flit boundaries. The loopback capability, where a packet can be routed from a node to itself, is supported to assist with diagnostics and driver development.

### 2.2.9 Chip Results

A 6-port four-lane router core utilizing the proposed double-pumped crossbar channel with LBD has been designed in a 150nm six-metal CMOS process. All parasitic capacitance and resistance data from layout was extracted and included in the circuit simulations, and the results are compared to reference work [19] in terms of area, performance and power and summarized in Figure 2-15. It is important to note that we compare the designs on the same process generation.

![Figure 2-15: Double-pumped crossbar channel results compared to work reported in [19].](image)

Double-pumping the crossbar channel reduces all four channel widths from 560µm to 310µm, and full-chip area from 15.8 mm\(^2\) to 12.2mm\(^2\), enabling a 45% reduction in channel area and 23% overall chip area, with no latency penalty over the original design. The smaller channel enables an 8.3% reduction
in worst-case interconnect length and a 7.7% improvement in total signal propagation delay due to reduced capacitance. This also results in a 7.2% improvement in average channel power, independent of traffic patterns. The technique however increases the clock load by 4% primarily because of the 2:1 multiplexer. Note that this is reported as a percentage of the clock load seen in Stage 4 and 5 of the router pipeline. This penalty relative to the clock load of the entire core is less than 1%. In addition, the 2:1 multiplexer increases signal propagation delay by 6%, which could be successfully traded for the 7.7% improvement in channel delay due to reduced double-pumped interconnect length.

Figure 2-16 plots the simulated logic delay, the wire delay and the total crossbar channel signal propagation delay as a function of distance of the receiver from the driver. The logic delay includes a fixed logic delay component, and a varying location-based driver delay. A port distance of “6” indicates the farthest routable port and “1” indicates packet loop-back on the same port. Notice that the total crossbar signal delay remains roughly constant over the range of port-distances. This is because the location based driver dynamically adjusts driver logic delay and wire delay to the specific destination port, without exceeding the combined delay budget of 500ps.

Figure 2-16: Simulated crossbar channel propagation delay as function of port distance.

Figure 2-17 shows the improvements in peak channel driver currents with LBD usage for various router traffic patterns. All 1824 channel drivers across all
four lanes were replaced by location based drivers. The graph plots the peak current per driver (in mA) for varying port distance and shows the current change from 5.9mA for the worst case port distance of 6 down to 1.55mA for the best case port distance of 1. The LBD technique achieves up to 3.8X reduction in peak channel currents, depending on the packet destination. As a percentage of logic in pipe stage 4, LBD control layout overhead is 3%.

![Graph showing LBD peak current as function of port distance.]

**Figure 2-17: LBD peak current as function of port distance.**

<table>
<thead>
<tr>
<th>Port</th>
<th>L0</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 0</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Port 1</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Port 2</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Port 3</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Port 4</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Port 5</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

**Figure 2-18: Router layout and characteristics.**

The router core layout and a summary of chip characteristics are shown in Figure 2-18. In a 150nm six-metal process, the 12.2mm² full custom core contains 1.92 million transistors and operates at 1GHz at 1.2V. The tiled nature of the physical implementation is easily visible. The six ports within each lane set run vertically, and the four lane sets are placed side by side. Note the placement of the four lanes, which are mirrored to reduce the number of routing
combinations across lanes. Also note the locations of the inbound queues in the crossbar and their relative area compared to the routing channel in the crossbar. De-coupling capacitors fill the space below the channel and occupy approximately 30% of the die area.

2.3 Summary and future work

This work has demonstrated the area and peak energy reduction benefits of a six port four lane 57 GB/sec communications router core. Design enhancements include four double-pumped crossbar channels and destination-aware crossbar channel drivers that dynamically configure based on the current packet destination. Combined application of both techniques enables a 45% reduction in channel area, 23% overall core area, up to 3.8X reduction in peak crossbar channel power, and 7.2% improvement in average channel power, with no performance penalty, when compared to an identical reference design on the same process. The crossbar area and peak power reduction also allows for larger crossbar networks to be integrated on a single die. This router design, when configured appropriately in a multiple-node system, yields a total system bandwidth in excess of one Terabyte/sec.

As future work we propose the application of low-swing signaling techniques [22] to router links and internal crossbar to significantly alleviate energy consumption. In addition, encoding techniques can also be applied to the crossbar channel data buses to further mitigate peak power based on [23]–[24].

2.4 References


Chapter 3

Floating-point Units

Scientific and engineering applications require high performance floating-point units (FPU). The advent of multimedia applications, such as 3D graphics and signal processing, places stronger demands for self-contained, low-latency FPUs with increased throughput. This chapter presents basic concepts involved in floating-point arithmetic, reviews conventional floating-point units (FPU), and describes the challenges in accomplishing single-cycle accumulate on today’s FPUs. A new pipelined single-precision FPMAC design, capable of single-cycle accumulation, is described in Paper 2 and Paper 3. Special flip-flop circuits used in the FPMAC design and test circuits are also presented.

3.1 Introduction to floating-point arithmetic

The design of FPUs is considered more difficult than most other arithmetic units due to the relatively large number of sequentially dependent steps required for a single FP operation and the extra circuits to deal with special cases such as infinity arithmetic, zeros and NaNs (Not a Number), as demanded by the IEEE-754 standard [ref]. As a result, there is opportunity for exploring algorithms, logic and circuit techniques that enable faster FPU implementations. An excellent introduction to FP arithmetic can be found in Omondi [1] and
Goldberg [2], which describe logic principles behind FP addition, multiplication and division. Since the mid 1980s, almost all commercial FPU designs have followed the IEEE-754 standard [3], which specifies two basic floating-point formats: single and double. The IEEE single format consists of three fields: a 23-bit fraction, $f$; an 8-bit biased exponent, $e$; and a 1-bit sign, $s$. These fields are stored contiguously in one 32-bit word, as shown in Figure 3-1.

![Figure 3-1: IEEE single precision format. The exponent is biased by 127.](image)

3.1.1 Challenges with floating-point addition and accumulation

FP addition is based on the sequence of mantissa operations: swap, shift, add, normalize and round. A typical floating-point adder (Figure 3-2) first compares the exponents of the two input operands, swaps and shifts the mantissa of the smaller number to get them aligned. It is necessary to adjust the sign if either of the incoming number is negative. The two mantissas are then added; with the result requiring another sign adjustment if negative. Finally, the adder renormalizes the sum, adjusts the exponent accordingly, and truncates the resulting mantissa using an appropriate rounding scheme [4]. For extra speed, FP adders use leading zero anticipatory (LZA) logic to carry out predecoding for normalization shifts in parallel mantissa addition. Clearly, single-cycle FP addition performance is impeded by the speed of the critical blocks: comparator, variable shifter, carry-propagate adder, normalization logic and rounding hardware. To accumulate a series of FP numbers (e.g., $\Sigma A_i$), the current sum is looped back as an input operand for the next addition operation, as shown by the dotted path in Figure 3-2. Over the past decade, a number of high-speed FPU designs have been presented [5]–[7]. Floating-point (FP) is dominated by multiplication and addition operations, justifying the need for fused multiply-add hardware. Probably the most common use of FPUs is performing matrix operations, and the most frequent matrix operation is a matrix multiplication, which boils down to computing an inner product of two vectors.

$$X_i = \sum_{j=0}^{n-1} A_i \times B_j$$  \hspace{1cm} (3.1)
3.1 Introduction to floating-point arithmetic

Computing the dot product requires a series of multiply-add operations. Motivated by this, the IBM RS/6000 first introduced a single instruction that computes the fused multiply-add (FPMADD) operation.

\[ X_i = (A \times B) + C \quad (3.2) \]

Although this requires a three operand read in a single instruction, it has the potential for improving the performance of computing inner products. In such cases, one option is to design the FP hardware to accept up to three operands for executing MADD instructions as shown in Figure 3-3(a), while other FP instructions requiring fewer than three operands may utilize the same hardware by forcing constants into the unused operands. For example, to execute the floating-point add instruction, \( T = A + C \), the B operand is forced to the constant 1.0. Similarly, a multiply operation would require forcing operand C to zero. Examples of fused multiply-add FPUs in this category include the CELL processing element, and are described in [8]–[10]. A second option is to optimize the FP hardware for dot product accumulations by accepting two operands, with an implicit third operand (Figure 3-3b). We define a fused multiply-accumulate (FPMAC) instruction as:

\[ X_i = (A \times B) + X_{i-1} \quad (3.3) \]
An important figure of merit for FPUs is the *initiation* (or repeat) interval [2], which is the number of cycles that must elapse between issuing two operations of a given type. An ideal FPU design would have an initiation interval of 1.

### 3.1.2 Scheduling issues with pipelined FPUs

To enable GHz operation, all state-of-art FPUs are pipelined, resulting in multi-cycle latencies on most FP instructions. An example of a five stage pipelined FP adder [11] is shown in Figure 3-4(a) and a six-stage multiply-add FPU [10] is given in Figure 3-4(b). An important observation is that neither of these implementations are optimal for accomplishing a steady stream of dot product accumulations. This is because of pipeline data hazards. For example, the six-stage FPU in Figure 3-4(b) has an initiation interval of 6 for FPMAC instructions i.e., once a FPMAC instruction is issued, a second one can only be issued a minimum of six cycles apart due to *read after write* (RAW) hazard [2]. In this case, a new accumulation can only start after the current instruction is complete. Schedulers of multi-cycle FPUs detect such hazards and place code scheduling restrictions between consecutive FPMAC instructions, resulting in throughput loss. The goal of this work is to eliminate such limitations by demonstrating single-cycle accumulate operation and a sustained FPMAC ($X_i$) result every cycle at GHz frequencies.

Several solutions have been proposed to minimize data hazards and stalls due to pipelining. Result *forwarding* (or *bypassing*) is a popular technique to reduce the effective pipeline latency. Notice that the *result bus* is forwarded as a possible input operand by the CELL processor FPU in Figure 3-4(b).
Multithreading has emerged as one of the most promising options to hide multi-cycle FPU latency by exploiting thread-level parallelism (TLP). Multithreaded systems [12] provide support for multiple contexts and fast context switching within the processor pipeline. This allows multiple threads to share the same FPU resources by dynamically switching between threads. Such a processor has the ability to switch between the threads, not only to hide memory latency but also to avoid stalls resulting from data hazards. As a result, multithreaded systems tend to achieve better processor utilization and improved performance. These benefits come with the disadvantage that multithreading can add significant complexity and area overhead to the architecture, thus increasing design time and cost.

### 3.2 Single-cycle Accumulation Algorithm

In an effort to achieve fast single-cycle accumulate operation, we first analyzed each of the critical operations involved (Figure 3-2) in conventional FPUs, with the intent of eliminating, reducing and/or deferring the amount of logic operations inside the accumulate loop. The proposed FPMAC algorithm is given in Figure 3-5, and employs the following optimizations:
(1) **Swap and Shift:** To minimize the interaction between the incoming operand and the accumulated result, we choose to self-align the incoming number every cycle rather than the conventional method of aligning it to the accumulator result. The incoming number is converted to base 32 by shifting the mantissa left by an amount given by the five least significant exponent bits (Exp[4:0]) thus extending the mantissa width from 24-bits to 55-bits. This approach has the benefit of removing the need for a variable mantissa alignment shifter inside the accumulation loop. The reduction in exponent from 8 to 3 bits (Exp[7:5]) expedites exponent comparison.

(2) **Addition:** Accumulation is performed in base 32 system. The accumulator retains the multiplier output in carry-save format and uses an array of 4-2 carry-save adders to “accumulate” the result in an intermediate format [13], and delay addition until the end of a repeated calculation such as accumulation or dot product. This idea is frequently used in multipliers, built using Wallace trees [14], and removes the need for an expensive carry propagate adder in the critical path. Expensive variable shifters in the accumulate loop are replaced with constant shifters, and are easily implemented using a simple multiplexer circuit.

![Figure 3-5: Single-cycle FPMAC algorithm.](image)

(3) **Normalize and round:** The costly normalization step is moved outside the accumulate loop, where the accumulation result in carry-save is added, the sum normalized and converted back to base 2, with necessary rounding.
3.3 CMOS prototype

With a single-cycle accumulate loop, we have successfully eliminated RAW data hazards due to accumulation and enabled scheduling of FPMAC instructions every cycle. Note that we could still have a RAW data hazard between consecutive FPMAC instructions if either input operands to the multiplier (A, B) as part of the current FPMAC instruction, is dependent on the previous accumulated result. In such cases, the second FPMAC instruction issue must be delayed (in cycles) by the amount of the entire FPMAC pipeline depth. A better option is to have a compiler re-order the instruction stream on such dependent operations and improve throughput. With this exception, the proposed design allows FPMAC instructions with an initiation interval of 1, significantly increasing throughput. A detailed description of the work is presented in Paper 2 and Paper 3.

3.3 CMOS prototype

A test chip using the proposed FPMAC architecture has been designed and fabricated in a 90nm communication technology [15]. The 2mm² full-custom design contains 230K transistors, with the FPMAC core accounting for 151K (67%) of the device count, and 0.88mm² (44%) of total layout area.

3.3.1 High-performance Flip-Flop Circuits

To enable fast performance, the FPMAC core uses implicit-pulsed semi-dynamic flip-flops [16]–[17], with fast clock-to-Q delay and high skew tolerance. When compared to a conventional static master-slave flip-flop, the semi-dynamic flip-flops provide both shorter latency and the capability of incorporating logic functions with minimum delay penalty, properties which make them very attractive for high-performance digital designs. Critical pipe stages like the FPMAC accumulator registers (Figure 3-5), are built using inverting rising edge-triggered semi-dynamic flip-flop with synchronous reset. The flip-flop (Figure 3-6) has a dynamic master stage coupled to a pseudo-static slave stage. As is shown in the schematic, the flip-flops are implicitly pulsed, with several advantages over non-pulsed designs. One main benefit is that they allow time-borrowing across cycle boundaries due to the fact that data can arrive coincident with, or even after, the clock edge. Thus negative setup time can be taken advantage of in the logic. Another benefit of negative setup time is that the flip-flop becomes less sensitive to jitter on the clock when the data arrives after clock. They thus offer better clock-to-output delay and clock skew tolerance than conventional master-slave flops. However, pulsed flip-flops have some important disadvantages. The worst-case hold time of this flip-flop can exceed clock-to-output delay because of pulse width variations across process, voltage, and temperature conditions. A selectable pulse delay option is available, as
shown in Figure 3-6, to avoid failures due to pulse-width variations over process corners and consequent min-delay failures.

![Figure 3-6: Semi-dynamic resetable flip flop with selectable pulse width.](image)

### 3.3.2 Test Circuits

To aid with testing the FPMAC core, the design includes three 32-bit wide, 32-deep first-in first-out (FIFO) buffers, operating at core speed (Figure 3-7).

![Figure 3-7: Block diagram of FPMAC core and test circuits.](image)

FIFOs A and B provide the input operands to the FPMAC and FIFO C captures the results. A 67-bit scan chain feeds the data and control words.
Output results are scanned out using a 32-bit scan chain. A control block manages operations of all three FIFOs and scan logic on chip.

Each FIFO is built using a register file unit that is 32-entry by 32b, with single read and write ports (Figure 3-8). The design is implemented as a large signal memory array [18]. A static design was chosen to reduce power and provide adequate robustness in the presence of large amounts of leakage. The RF design is organized in four identical 8-entry, 32b banks. For fast, single-cycle read operation, all four banks are simultaneously accessed and multiplexed to obtain the desired data. An 10-transistor, leakage-tolerant, dual-$V_T$ optimized RF cell with 1-read/1-write ports is used. Reads and writes to two different locations in the RF occur simultaneously in a single clock cycle. To reduce the routing and area cost, the circuits for reading and writing registers are implemented in a single-ended fashion. Local bit lines are segmented to reduce bit-line capacitive loading and leakage. As a result, address decoding time, read access time, as well as robustness improve. RF read and write paths are dual-$V_T$ optimized for best performance with minimum leakage. The RF RAM latch and access devices in the write path are made high-$V_T$ to reduce leakage power. Low-$V_T$ devices are used everywhere else to improve critical read delay by 21% over a fully high-$V_T$ design.

Figure 3-8: 32-entry x 32b dual-$V_T$ optimized register file.
3.4 References


