Spring 5-15-2015

# Utilizing Magnetic Tunnel Junction Devices in Digital Systems

Michael James Hall
*Washington University in St. Louis*

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:
Dr. Roger D. Chamberlain, Chair
Dr. Ron Cytron
Dr. George Engel
Dr. Viktor Gruev
Dr. Robert Morley
Dr. Dave Richard

# Utilizing Magnetic Tunnel Junction Devices in Digital Systems

by
Michael J. Hall

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2015
Saint Louis, Missouri

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| 3M2P | 3 metal 2 poly |
| 5M1P | 5 metal 1 poly |
| AES | Advanced Encryption Standard |
| API | Application programming interface |
| ASIC | Application-specific integrated circuit |
| BJT | Bipolar junction transistor |
| CBC | Cipher-block chaining |
| CIMS | Current-induced magnetic switching |
| CMOS | Complementary metal-oxide semiconductor |
| DAC | Digital-to-analog converter |
| DRAM | Dynamic random-access memory |
| FIFO | First In, First Out |
| FIMS | Field-induced magnetic switching |
| FPGA | Field-programmable gate array |
| FPLD | Field-programmable logic device |
| GPU | Graphics processing unit |
| IID | Independent and identically distributed |
| M/D/1 | Markovian, or memoryless, arrival process; Deterministic service process; 1 server |
| M/G/1 | Markovian, or memoryless, arrival process; General service process; 1 server |
| MOSFET | Metal-oxide semiconductor field-effect transistor |
| MRAM | Magnetoresistive random-access memory |

| | |
|---|---|
| MSC | Magnetic sense circuit |
| MTJ | Magnetic tunnel junction |
| NMOS | n-channel MOSFET |
| PC | Personal computer |
| PCB | Printed circuit board |
| PLL | Phase-locked loop |
| PMOS | p-channel MOSFET |
| QQ | Quantile-quantile |
| R2V | Resistance-to-voltage |
| RA | Resistance-area |
| SHA | Secure Hash Algorithm |
| SMT | Simultaneous multithreading |
| SPI | Serial peripheral interface |
| SRAM | Static random-access memory |
| STT | Spin-torque transfer |
| TMR | Tunneling magnetoresistance ratio |
| USB | Universal serial bus |

# Acknowledgments

I am pleased to thank those who made this thesis possible as well as those who have supported me along the way. I am especially grateful to my research advisor Dr. Roger Chamberlain who has provided me with guidance, advice, and help during my years as a graduate student. He has been very patient and encouraging. I am also grateful to my co-advisor Dr. Viktor Gruev for the guidance he has given in anything pertaining to circuit design.

This thesis would not be possible without the support of funding sources. The research was supported by the Air Force Office of Scientific Research (AFOSR) under the Discovery Thrust Program, contract no. FA9550-08-1-0473; the National Science Foundation (NSF) through grants CCF-0427794, CNS-0751212, and CNS-0931693; Exegy, Inc.; and VelociData, Inc.

Thanks to the members of my dissertation committee for agreeing to serve on the committee; our collaborators at Organ State University who introduced us to magnetic tunnel junctions (MTJs) and provided us with a Verilog-A simulation model of the MTJ; Dr. Roch Guérin, the department chair, who steered me towards a vacation model to improve our queueing model; and VelociData with Dr. Joe Lancaster, my peer and supervisor, for the opportunity to work in a summer internship to build the SHA-2 application that is included in this dissertation.

It has been a pleasure also to work with Dr. Raj Jain periodically throughout my studies as a teaching assistant for his class and editor for his book. The knowledge and techniques learned from the class and book have been invaluable in my research work. I have applied

them to develop queueing models included in this dissertation and to analyze experimental results. These techniques, no doubt, will continue to be useful in my career, and I am especially grateful for the collaboration I have had with him.

Further, thanks to Washington University, the Computer Science & Engineering department, the staff and faculty, and those who have been supportive of me which include my parents, siblings, relatives, friends, and church community. Thanks also to my academic mentor Dr. George Engel from Southern Illinois University Edwardsville (SIUE) who encouraged me to pursue a Ph.D.

Finally, I would like to express my deepest thanks to God whom I love and who is an integral part of my life. God has been with me constantly throughout my Ph.D. studies and has always been one that I can turn to in prayer. Through prayer, I have daily offered up my work to God for sanctification.

Michael J. Hall

*Washington University in Saint Louis*
*May 2015*

Dedicated to the three people who have been tremendous influences in my life:
my mother, father, and academic mentor Dr. George Engel.

The people who have been the most influential in my life in pursuing a computer engineering career are my parents Gerald and Janet Hall, whom I dearly love, and my academic mentor Dr. George Engel. I am especially grateful to my parents for all of their love, help, and encouragement. My mother raised me in my Catholic faith and taught me morals and values that guide me through life. My father worked with me and helped me in my projects, and inspired me in my interest in computers. When I came to my parents in 2$^{nd}$ grade and asked them to give me math problems to solve, my dad said, "You can write a program to do that." Well, we did just that, and I have been playing with computers ever since. Finally, I thank my academic mentor Dr. George Engel, at SIUE, for encouraging me to go on to pursue a Ph.D.

ABSTRACT OF THE DISSERTATION

Utilizing Magnetic Tunnel Junction Devices in Digital Systems
by
Michael J. Hall
Doctor of Philosophy in Computer Engineering
Washington University in St. Louis, 2015
Professor Roger D. Chamberlain, Chair

The research described in this dissertation is motivated by the desire to effectively utilize magnetic tunnel junctions (MTJs) in digital systems. We explore two aspects of this: (1) a read circuit useful for global clocking and magnetologic, and (2) hardware virtualization that utilizes the deeply-pipelined nature of magnetologic.

In the first aspect, a read circuit is used to sense the state of an MTJ (low or high resistance) and produce a logic output that represents this state. With global clocking, an external magnetic field combined with on-chip MTJs is used as an alternative mechanism for distributing the clock signal across the chip. With magnetologic, logic is evaluated with MTJs that must be sensed by a read circuit and used to drive downstream logic. For these two uses, we develop a resistance-to-voltage (R2V) read circuit to sense MTJ resistance and produce a logic voltage output. We design and fabricate a prototype test chip in the 3 metal 2 poly 0.5 $\mu$m process for testing the R2V read circuit and experimentally validating its correctness. Using a clocked low/high resistor pair, we show that the read circuit can correctly detect the input resistance and produce the desired square wave output. The read circuit speed is measured to operate correctly up to 48 MHz. The input node is relatively insensitive to node capacitance and can handle up to 10s of pF of capacitance without changing the bandwidth of the circuit.

In the second aspect, hardware virtualization is a technique by which deeply-pipelined circuits that have feedback can be utilized. MTJs have the potential to act as state in a magnetologic circuit which may result in a deep pipeline. Streams of computation are then context switched into the hardware logic, allowing them to share hardware resources and more fully utilize the pipeline stages of the logic. While applicable to magnetologic using MTJs, virtualization is also applicable to traditional logic technologies like CMOS. Our investigation targets MTJs, FPGAs, and ASICs. We develop M/D/1 and M/G/1 queueing models of the performance of virtualized hardware with secondary memory using a fixed, hierarchical, round-robin schedule that predict average throughput, latency, and queue occupancy in the system. We develop three $C$-slow applications and calibrate them to a clock and resource model for FPGA and ASIC technologies. Last, using the M/G/1 model, we predict throughput, latency, and resource usage for MTJ, FPGA, and ASIC technologies. We show three design scenarios illustrating ways in which to use the model.

# Chapter 1

# Introduction

Thin-film magnetic devices based on the magnetic tunnel junction (MTJ) are actively being researched for applications in memory [1], field-programmable gate arrays (FPGAs) [2], and logic computation [3], and have been developed as early as the 1980s [4]. These devices can store information non-volatilely in a magnetic field that retains its value across power cycles. Fabrication of these devices is compatible with the complementary metal-oxide semiconductor (CMOS) process allowing them to be integrated on a CMOS chip. Integration improves area efficiency, lowers wire resistance and capacitance, and reduces power. Further, these devices can scale down with the CMOS process and have been shown to be fabricated in a 45 nm process node with a size of 40 nm along the short axis and an aspect ratio, long vs. short, of $\sim$ 2.5 to 3 [5]. The write endurance, or the number of times a stored value can change without failure in the device, is practically infinite. They also have potentially short read and write times, allowing stored data to be accessed quickly [6].

A basic illustration of the MTJ device is shown in Figure 1.1 with a write-strap that programs the MTJ using an induced magnetic field. The MTJ device is constructed using two ferromagnetic layers separated by a thin insulator such as MgO that is the tunnel barrier. The top ferromagnetic layer, called the free layer, can have a magnetic orientation in one of

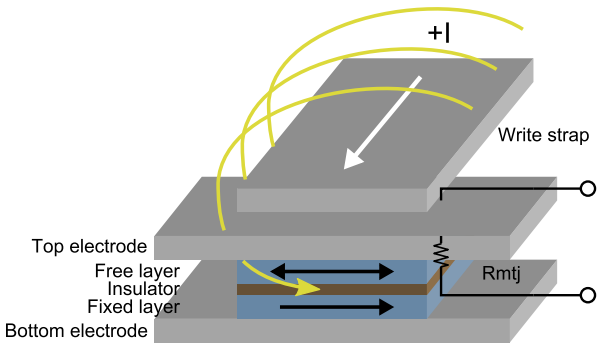



Figure 1.1: A magnetic tunnel junction (MTJ) device that encodes state information in the magnetic orientation of the free layer. The state is programmed with a current (+I) and accessed as a resistance ($R_{mtj}$) between the two electrodes.

two directions as indicated in the figure. The bottom ferromagnetic layer, called the fixed layer, has its magnetic orientation in one direction that is pinned during the manufacturing process. The resistance, $R_{mtj}$, as seen across the electrodes of the device, is dependent on the magnetic orientation of the free layer relative to the fixed layer. When the orientations of both layers are parallel (in the same direction), the resistance seen through the MTJ is low ($R_L$). In contrast, when the orientations of both layers are anti-parallel (in opposite directions), the resistance seen through the MTJ is high ($R_H$). Thus, the state is accessed as a resistance. The orientation of the free layer can be set by applying a magnetic field of sufficient strength through the free layer. The magnetic field can be generated externally or induced by sourcing a current through the write-strap [4].

## 1.1 Uses of MTJs

The first use of MTJs is memory. Memory technology has developed rapidly to serve different needs: static random-access memory (SRAM) for high-speed data storage, dynamic random-access memory (DRAM) for low-cost data storage, and Flash memory for high-density, non-volatile data storage. All three technologies have their advantages and disadvantages. Researchers are working to develop a universal memory technology that combines the best aspects of all three. That is, a memory that is non-volatile, has a high-write endurance, low read/write energy, and high read/write speed [7]. Magnetoresistive random-access memory (MRAM) using MTJs is one such promising universal memory technology [8, 9, 10]. MRAM is actively being researched and improved; Everspin Technologies, a spin-off company from Freescale Semiconductor formed in 2008, is actively commercializing MRAM memory and is today selling 64 Mb MRAM chips [11]. MRAM circuits typically use current conveyors for readout of the MRAM cells. While there is still research to be accomplished in the area of MRAM memory design, that is not the focus of this dissertation.

The second use of MTJs is configuration memory in FPGAs. FPGAs currently use SRAM cells to store the programmed configuration data used to configure the lookup tables in the reconfigurable logic. However, these memory cells are volatile and lose their data values across power cycles, requiring them be reprogrammed at power-up. If they are replaced with MRAM cells, then the configuration memory will be able to retain its data values across power cycles due to the non-volatile nature of MRAM. This consequently means that the configuration memory will only need to be programmed once [2]. The MRAM cells are based on the same structure as the SRAM cells. They use an SRAM-based sense amplifier, which is an unbalanced magnetic flip-flop in voltage-mode, to sense a stored data value from a

differential pair of MTJs [12]. As with MRAM, this dissertation will not investigate the use of MTJs as configuration memory in FPGAs.

The third use of MTJs is clocking which is a new use of MTJs not previously proposed in the literature. Synchronous logic circuits in digital systems need to receive a common clock signal at the same time across a chip. Clock distribution trees are the predominant way to distribute the clock from a single source. They use a tree-like structure and clock buffers to balance the signal propagation of the clock to every flip-flop in synchronous logic circuits. On modern processors, clock power is significant, consuming on the order of 25% of total chip power [13]. Clock skew with the clock distribution tree can be significant as the clock is driven at higher frequencies, contributing to 3.8% of the clock period in the Cell processor running at 3.2 GHz [13]. Replacing the clock distribution tree with an equivalent mechanism that can distribute the clock to all flip-flops has the potential to yield benefits in area, speed, and on-chip power dissipation.

A global external magnetic field combined with on-chip MTJs is an alternative mechanism for distributing the clock across a chip as illustrated in Figure 1.2. We proposed this in [14]. This is similar to optical clock distribution in free-space as proposed in the literature [15]. A resistance-to-voltage read circuit is necessary for sensing MTJ resistance state. The MTJs, when distributed across the chip, sense a global magnetic field, and, using a read circuit, produce local clock signals that drive nearby synchronous logic circuits. In this way, a global clock distribution tree is not needed, however, local clock distribution is still necessary. This should result in reduced on-chip power dissipation and area. If the global clock is relatively insensitive to process variations in the MTJ devices and corresponding read circuits such that the jitter between local clocks is small, then clock skew will decrease, resulting in increased circuit speed. The external magnetic field generated off-chip, however, will consume power. It is an open question as to whether or not there is a net power benefit in the system. This

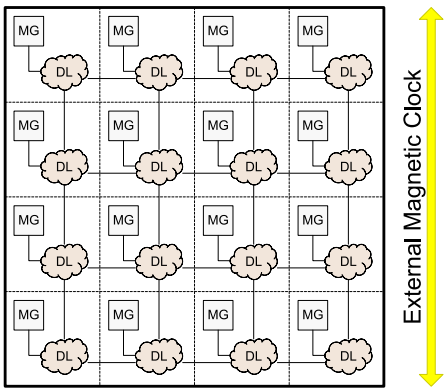



Figure 1.2: Magnetic clocking across a chip using an external magnetic field. A chip is partitioned into regions that contain a Magnetic Clock Generator (MG) and a Digital Logic (DL) region. The MG contains an MTJ and a read circuit that senses the external field and generates a local clock. The local clock drives nearby digital logic. Because the local clocks are synchronous, interconnections can be made from any DL region to any other DL region (only regular connections are illustrated).

dissertation will investigate the design of the resistance-to-voltage read circuit and present experimental results from a prototype test chip. The read circuit can be used to produce local clock signals generated from an external oscillating magnetic field. This dissertation will also describe the design of the prototype chip to be used for testing global clocking. However, due to the inaccessibility of obtaining MTJs, this dissertation will not present experimental results demonstrating a functional globally clocked chip.

The fourth use of MTJs is logic. Logic circuits designed using MTJs depend on the type of MTJ device used. These types include field-induced, toggle, thermally-assisted, and spin-torque transfer [4]. Several approaches have been proposed in the literature for constructing fundamental magnetologic circuits (e.g. gates) including device threshold logic [16], serially-connected devices for complex logic [3, 12, 17, 18], cascading device logic [19], and resistive-based lookup tables with MTJ storage elements [20]. This dissertation will not investigate how to build fundamental magnetologic circuits.

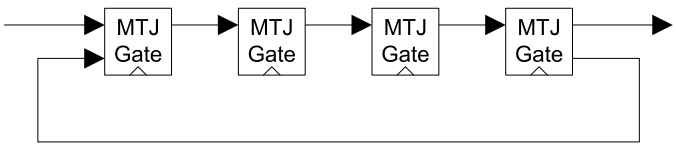



Figure 1.3: Pipelined magnetologic circuit with feedback. Four MTJ logic gates are shown, each acting as a pipeline stage with state. It takes 4 clock cycles to compute a single result with the feedback path.

There are commonalities in the way fundamental magnetologic circuits are built. First, each MTJ device acts as a latch to whatever data is written to it. This gives it the potential to act as state in the pipeline stage of a logic circuit as shown in Figure 1.3. We want to exploit the latching property of fundamental magnetologic circuits for constructing deeply-pipelined logic. Second, data is accessed as a resistance between the terminals of the MTJs. A read circuit, therefore, is commonly needed to sense this resistance.

To exploit the deeply-pipelined nature of magnetologic circuits when feedback is present, context switching can be used to fully utilize the pipeline stages of the deeply-pipelined logic, allowing them to compute multiple data streams. This allows the hardware (a.k.a. the logic) to be virtualized, meaning that each data stream shares the hardware resource. The circuits are sequential logic circuits with pipelined combinational logic. The pipelined combinational logic adds latency and decreases single stream throughput since it takes multiple clock cycles (corresponding to the number of pipeline stages) to compute a single result and feed it back to the input. If the number of pipeline stages is $C$, then this circuit is said to be $C$-slowed [21] since a single computation runs $C$ times slower. $C$-slow is a technique described by Leiserson by which every register is replaced by $C$ registers and then retimed to balance the registers throughout the combinational logic. Application of the $C$-slow technique is shown in Figure 1.4. Exploiting this characteristic allows processing multiple different contexts or data streams in a fine-grain way using the same hardware logic. The number of fine-grain

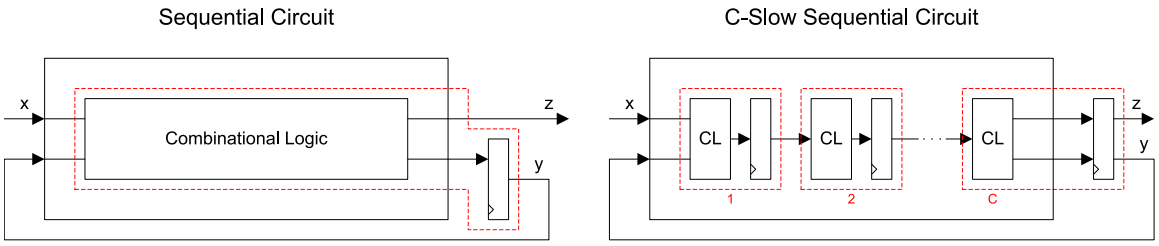



Figure 1.4: Application of the $C$-slow technique to a non-pipelined circuit. The non-pipelined sequential circuit is shown on the left. $C$-slow is applied to this circuit by replacing the register with $C$ registers and distributing it amongst the combinational logic (CL) shown on the right. $x$ is the input, $y$ is the feedback, and $z$ is the output.

contexts supported equals the pipeline depth. Context switching the hardware can utilize all the pipeline stages, and if they are evenly spaced, improve the clock frequency as well.

Virtualization of the hardware is akin to simultaneous multithreading (SMT) in a processor where fast context switches allow executing multiple threads simultaneously on the hardware, also sharing the hardware resource [22]. Graphics processing units (GPUs) regularly use this technique, frequently supporting large numbers of threads per core. Asymptotic models that employ this technique have been introduced by Ma et al. [23, 24]. This dissertation will investigate hardware virtualization using context switching as a way to exploit the deeply-pipelined nature of magnetologic circuits.

## 1.2 Virtualized hardware

Virtualized hardware using context switching is more widely applicable than its use in magnetologic. It can also be used in existing FPGA and ASIC technologies. Context switching is a technique by which hardware resources are reused to support processing multiple computations. It can be fine- or coarse-grain. Context switching using deeply-pipelined or $C$-slowed

logic circuits is fine-grain. Context switching using a secondary memory and explicitly swapping contexts in and out at an infrequent rate is coarse-grain.

There are three main benefits of context switching hardware: (1) the ability to reuse hardware resources for doing computation on multiple data streams, (2) an increase in the clock frequency when the combinational logic is evenly pipelined, and (3) an increase in the total throughput of all computations when total hardware resources are limited. For the first benefit, reusing hardware for computation is beneficial when available resources for a design are limited such as on an FPGA or when the cost to build hardware is high such as on an ASIC. For the second benefit, the clock frequency increases because the combinational logic is broken up more finely, allowing signals to propagate faster between pipeline stages. For the third benefit, total throughput increases because more computation can be done on a fixed amount of resources than could otherwise be done and as the clock frequency increases the throughput also increases.

This dissertation will investigate the design of virtualized hardware, model its performance, and optimize a schedule for context switching. The model will predict circuit performance and resource usage, and provide guidance in selecting design parameters.

## 1.3 List of contributions

In this dissertation, the following contributions are made:

- Noise analysis of a basic current-mode read circuit [25] and design guidance of performance tradeoffs
- Transistor sizing equations of several different current conveyor structures

- Design, layout, and simulation of a resistance-to-voltage (R2V) read circuit [26]
- Design, layout, and fabrication of a prototype test chip for testing the R2V read circuit and aspects of global clocking
- Design, layout, and fabrication of a printed circuit board (PCB) for testing the prototype chip
- Construction of the PCB and test setup for testing the prototype chip
- Experimental results and characterizations of the R2V read circuit in the prototype chip
- Automated tool scripts and infrastructure for synthesizing virtualized hardware logic circuits and exploring the design space
- Development of a clock period model based on a randomized model of logic delay
- Development of M/D/1 [27, 28] and M/G/1 queueing models for predicting performance in virtualized hardware logic circuits
- Design of 3 $C$-slow applications for virtualization
  - Synthetic cosine application implemented via a Taylor series expansion with added feedback
  - Advanced Encryption Standard (AES) cipher in cipher-block chaining (CBC) mode for encryption [27]
  - Secure Hash Algorithm (SHA-2) with 256 and 512 bit digests (SHA-256 and SHA-512)
- Calibration of the 3 $C$-slow applications to the clock period model and a resource model

- Demonstration of ways to use the performance model (queueing model + clock period model + resource model) to predict performance in virtualized hardware logic circuits [27]

## 1.4 Outline of the dissertation

The outline of the dissertation is as follows. Chapter 2 provides relevant background information about MTJ devices, MTJ read circuits, current conveyor operation, magnetologic circuits, $C$-slow transformation, hardware virtualization, queueing notation, and vacation models. Chapter 3 describes MTJ read circuit theory including a noise analysis of a current-mode read circuit, transistor sizing equations for several different current conveyor structures, and the design and simulation of a resistance-to-voltage (R2V) read circuit. Chapter 4 describes MTJ read circuit experimentation including the design of a prototype system and experimental results of the R2V read circuit in the prototype system. Chapter 5 investigates hardware virtualization which includes the design, modeling, and optimization of virtualized hardware logic circuits, and presents results with three example applications. Last, Chapter 6 presents conclusions and future work.

# Chapter 2

# Background and Related Work

## 2.1 MTJ devices

The magnetic tunnel junction (MTJ) device, illustrated in Figure 2.1, is a small thin-film magnetic device that is constructed using two ferromagnetic layers such as CoFeB (top layer) and CoFe (bottom layer) separated by a thin insulator such as MgO that is the tunnel barrier [4]. The figure is similar to the one illustrated in Figure 1.1, but without the write strap. The ferromagnetic layers can take on one of two magnetic orientations. Usually one of the layers is pinned during fabrication so that its magnetic orientation is fixed while the other layer is free to rotate. These layers are called the fixed layer and free layer, respectively. Pinning is done by placing an antiferromagnetic layer below the fixed layer (not shown) constructed using a material such as IrMn that prevents the fixed layer from switching at ambient temperature [4]. The free layer effectively stores one bit of state information determined by its magnetic orientation. Since it requires no power to sustain a particular magnetic orientation, the state information is non-volatile.

Reading the MTJ device state can be accomplished by sensing a resistance output. The output is sensed via the two electrodes of the device whose resistance is determined by the

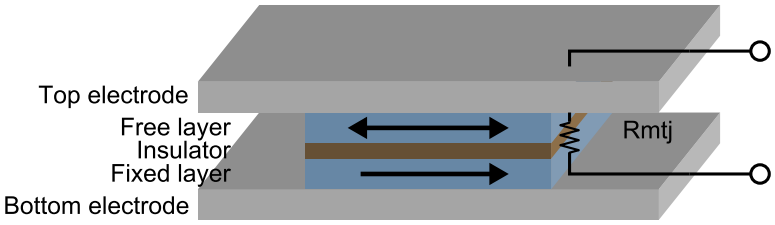



Figure 2.1: Magnetic tunnel junction (MTJ) device. Basic construction consists of two ferromagnetic layers (free and fixed layer) and an insulator. The fixed layer magnetic orientation is pinned whereas the free layer magnetic orientation is free to switch. For field-induced magnetic switching (FIMS), an additional write strap may be added above or below the MTJ. Current passing through the write strap induces a magnetic field to switch the free layer. For current-induced magnetic switching (CIMS), current instead passes through the MTJ device itself to switch the free layer. The state of the device is accessed as a resistance seen looking between the two electrodes.

relative magnetic orientations of the free and fixed layers. A device is characterized by its resistance-area (RA) product, which can range from $10\,\Omega \cdot \mu\text{m}^2$ [29] to $7.6\,\text{k}\Omega \cdot \mu\text{m}^2$ [1]. When the magnetic orientations are parallel, that is, oriented in the same direction, the resistance is low ($R_L$). Likewise, when they are anti-parallel, that is, oriented in opposite directions, the resistance is high ($R_H$). Sensing this low/high resistance provides a means for reading the device state. A tunneling magnetoresistance ratio (TMR) is defined as $TMR = \frac{R_H - R_L}{R_L}$ that characterizes the separation between resistance states. The TMR is dependent on the bias voltage, $V_{bias}$, across the electrodes of the device. Large bias voltages result in small TMR and a high rate of device failure [4].

Writing to the MTJ device can be done via one of two basic write mechanisms: field-induced magnetic switching (FIMS), and current-induced magnetic switching (CIMS). With FIMS, a write strap is added over the MTJ device and bidirectional current is applied through the write strap. This induces a magnetic field that permeates through the free layer and sets its magnetic orientation. The fixed layer, however, does not switch because the pinning layer below it prevents it from doing so. The applied current has to exceed the switching threshold of the free layer. Alternatively, this field might be induced by an externally generated global

field. With CIMS, a bidirectional current is applied directly through the electrodes of the MTJ device which becomes spin-polarized and sets the magnetic orientation of the free layer. A critical current has to be exceeded for switching to occur [4]. The field strength needed to switch an MTJ is at least 4 times the Earth's geomagnetic field [14].

**Field-induced magnetic switching (FIMS) devices:** There are three different types of MTJ devices that use the FIMS write mechanism: conventional, toggle, and thermally-assisted [4]. Conventional devices frequently have two write straps (easy and hard axis) that cross each other at a 90° angle. The hard axis lowers the threshold for magnetic switching, and the easy axis magnetically switches the free layer. Toggle devices are similar to conventional devices, but are constructed to prevent a device from being only half-selected (where a field is applied along one axis but not the other). This is done by performing a 4-phase rotation to toggle the device. Thermally-assisted devices, on the other hand, are constructed to lower the write energy required by preheating the device with a heat current prior to inducing a field to switch the free layer.

**Current-induced magnetic switching (CIMS) devices:** Spin-torque transfer (STT) devices utilize the CIMS write mechanism [4, 10]. They have a few benefits that do not exist in FIMS devices. One, STT devices do not suffer from cross-talk between fields, and therefore, they can be placed closer together for greater density. Two, STT devices scale well with power. As the process dimensions decrease, the power required to write to the device decreases as well. Many STT devices have two terminals which are used for both reading and writing. For reading, a large MTJ resistance is desirable for determining its state. For writing, a small MTJ resistance is desirable so that the voltage drop across the device due to the write current is small. A three-terminal STT device was constructed by Braganca et al. [30] that achieves this by providing a low-resistance write terminal and a

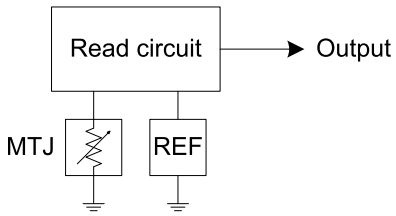



Figure 2.2: Read circuit block diagram for an MTJ. The read circuit senses an MTJ resistance and compares this to a reference to produce an output.

high-resistance read terminal. According to their simulations, the 3-terminal STT device requires approximately the same write current as a 2-terminal STT device, but with a lower voltage drop.

## 2.2 MTJ read circuits

An MTJ device is accessed as a resistance as seen through its tunnel junction. This resistance represents the logic state of the device, whether low or high. In order to read the state, a read circuit is necessary to convert the resistance into a corresponding logic voltage output. A read circuit can be designed in many ways. It can be voltage- or current-clamped, use a voltage- or current-mode comparator, use a single MTJ with a reference or a pair of differential MTJs, and be sampled or continuous in time. Several read circuits have been designed in the literature to accomplish this task. We describe several below. A block diagram for a basic read circuit is shown in Figure 2.2.

**Current conveyor with voltage comparator [31]:** This read circuit uses a differential current conveyor with a voltage comparator for sensing data in an MRAM cell. There are two inputs to the differential current conveyor: an MTJ device with a low/high resistance, and a reference with a midpoint resistance. The current conveyor clamps both inputs to a

fixed voltage and produces currents according to Ohm's law ($I = \frac{V}{R}$). These currents are then converted to voltages and compared by a voltage comparator.

**Current conveyor with current comparator [32]:** This read circuit is similar to the one in [31]. It uses a differential current conveyor with a current comparator for sensing data in an MRAM cell and is designed to be low power. There are two inputs to the differential current conveyor: an MTJ device with a low/high resistance and a reference with a midpoint resistance. The current conveyor clamps both inputs to a fixed voltage and produces currents according to Ohm's law. These currents are then compared by a current comparator to produce the final voltage output. This is the type of MTJ read circuit that is investigated in this dissertation.

**Voltage-mode sense amplifier [12]:** This read circuit is used to compare networks of MTJ devices in a structure called the basic field-programmable logic device (FPLD). It works by first applying a constant sense current, $I_{sens}$, to both the positive and negative terminals of the sense amplifier in the FPLD. This then produces corresponding terminal voltages across the MTJ networks according to Ohm's law. Comparing these terminal voltages, the sense amplifier then produces a final voltage output. The FPLD structure is used to construct complex Boolean logic expressions.

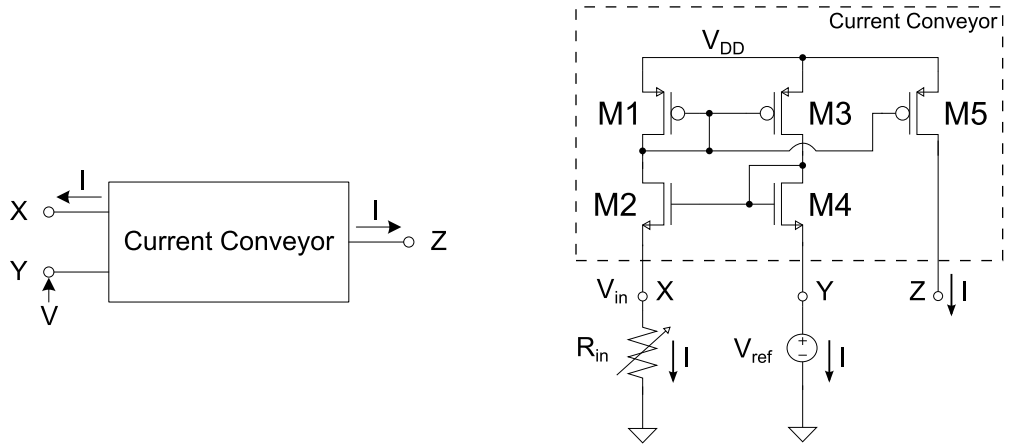
**Differential unbalanced flip-flop circuit [12]:** This read circuit is a reconfigurable static latch cell that consists of a differential pair of inverters connected with feedback to form a flip-flop. This is based on the static random-access memory (SRAM) cell circuit [33]. A complementary pair of MTJs are connected in series with the inverters to ground. This causes the flip-flop to be unbalanced due to the difference in MTJ resistance of the

complementary pair. A sense transistor, connected between the inverter outputs, shorts them together when closed and senses the MTJs when opened. When the sense transistor is opened, the inverters swing low and high based on the relative MTJ resistances. This samples the MTJ state discretely in time.

## 2.3 Current conveyor operation

The current conveyor, shown in a black box representation in Figure 2.3a, is a 3-terminal circuit introduced by Smith and Sedra [34] that provides current amplification with unity gain. A voltage $V$ applied at $Y$ appears at $X$, causing current $I$ to flow out at $X$. In a dual manner, the same current $I$ also flows out at $Y$. The current at $X$ is also conveyed out at $Z$ with high impedance. This circuit thus allows current to be conveyed from one source ($X$) to another ($Z$) while holding the voltage potential constant (at $X$).

An implementation of the basic current conveyor circuit using complementary metal-oxide semiconductor (CMOS) transistors [35, 36] is shown in Figure 2.3b which is used in our design of the MTJ read circuit for sensing MTJ resistance and producing an output current. The current conveyor is a feedback circuit consisting of two current mirror circuits between the left and middle branches connected together in a feedback loop and a high impedance output current out from the right branch. Transistors $M_{1,3,5}$ form the PMOS current mirror circuits and transistors $M_{2,4}$ form the NMOS current mirror circuit. The supplied $V_{ref}$ voltage then clamps the $V_{in}$ voltage to $V_{ref}$ potential over the input resistance $R_{in}$. This in turn produces a current $I_{in}$ through the left branch according to Ohm's Law that is inversely-proportional to the input resistance by the equation $I_{in} = {}^{V_{ref}}/_{R_{in}}$. The currents are mirrored between each branch (left, middle, and right) with a 1:1 gain which is achieved by matching the width




(a) Black box representation (b) Circuit implementation

Figure 2.3: Basic current conveyor.

and length of the transistors in the current mirrors [37], thus making $\left(\frac{W}{L}\right)_1 = \left(\frac{W}{L}\right)_3 = \left(\frac{W}{L}\right)_5$, $\left(\frac{W}{L}\right)_2 = \left(\frac{W}{L}\right)_4$, and $I_{in} = I_{sd,1} = I_{sd,3} = I_{sd,5} = I_{ds,2} = I_{ds,4}$.

## 2.4 Magnetologic circuits

There are several different ways to construct fundamental magnetologic circuits (e.g. basic gates), some of which may require CMOS support circuitry. These include device threshold logic, serially-connected devices for complex logic, cascading device logic, and lookup tables. Each approach is discussed separately; however, they are not all mutually exclusive. Some can be combined and used together.

**Device threshold logic [12, 16]:** This is a way to build simple magnetologic gates using threshold logic. Inputs combine as either currents or fields to switch the free layer of the device when the device threshold is exceeded. Since the free layer is only conditionally switched, threshold logic is implemented using a preset and evaluate phase. In the preset

phase, the MTJ device is preset to a value that selects the logic function to be evaluated (such as AND or OR). Then, in the evaluate phase, the logic is evaluated using threshold logic. The state of the free layer then represents the logic output which is accessed as a resistance. Device threshold logic needs read/write circuits to connect between gates.

**Serially-connected devices for complex logic [12, 18, 38]:** This is a way to build complex magnetologic gates using CMOS support circuitry. It is constructed using two strings of MTJ devices whose output resistances are serially connected to the V+ and V- nodes of a voltage sense amplifier. Logic inputs to this complex gate drive the MTJ devices' write lines to set their state. Based on their state, logic is evaluated when the strings of devices are sensed. A Boolean expression representing the logic function can be derived from the equation of the voltage sense amplifier. Simple logic functions can be implemented in multiple ways from this complex Boolean expression. Further, serially-connected devices can be combined with device threshold logic to construct even more complex logic gates [18].

**Cascaded device logic [19]:** This is a way to build simple magnetologic gates without CMOS support circuitry. It relies on device threshold logic with 3-terminal spin-torque transfer (STT) devices. With cascaded device logic, the read current of one device acts as the write current to the next device. Since device threshold logic conditionally switches the free-layer, cascaded device logic requires preset and evaluate phases to operate. Simple logic gates such as AND, OR, NAND, and NOR can be implemented.

**Lookup tables [20]:** This is a way to build reconfigurable magnetologic gates using CMOS support circuitry to implement arbitrary logic functions from a truth table. The truth table is specified by the values stored in MTJ devices and looked up via a multiplexor tree. The

multiplexer tree selects one of the devices based on a given input and accesses the device's resistance which is then sensed by a sense amplifier. The sense amplifier then produces the final output value. These lookup tables can be used in reconfigurable logic arrays such as field-programmable gate arrays (FPGAs) and in resistive computing [39].

## 2.5 $C$-slow transformation

$C$-slow is a transformation described by Leiserson [21] whereby every register in a digital logic circuit is replaced by $C$ registers as illustrated in Figure 1.4. This allows sequential logic circuits, which have feedback paths, to be pipelined and retimed. Retiming is a technique for improving the clock frequency of a circuit by moving pipeline registers forward and backwards through the combinational logic to shorten the critical path of the circuit. Retiming a $C$-slowed circuit can theoretically give up to $C$ times improvement in the clock frequency.

$C$-slow circuits, by their very nature, are multi-processing. They have $C$ pipeline registers in the feedback path allowing them to process multiple data streams interleaved into a single stream. It takes $C$ clock cycles to compute an output for a single data stream and to feed it back to the input to be processed with the next data value. Multiple data streams can be processed at different stages in the pipeline, allowing the pipeline to be more fully utilized. For a single stream, the throughput, measured as data values per clock cycle, will be $C$ times slower (in terms of clock cycles) than before the $C$-slow transformation. For all streams combined, the total throughput will be the same.

The design of $C$-slow circuits has several constraints relative to traditional synchronous circuit designs [40]. They cannot use asynchronous sets or resets. They cannot use a global synchronous reset as it creates too many constraints in the retiming process. And, they

cannot use the enable input of a flip-flop. Rather, reset and enable can be expressed as logic, allowing contexts to still be reset and enabled.

Several applications have been implemented using the $C$-slow technique. In 2003, Weaver et al. applied $C$-slow to three applications: AES encryption, Smith/Waterman sequence matching, and LEON 1 synthesized microprocessor core [40]. They designed an automatic $C$-slow retiming tool that would replace every register in a synthesized design with $C$ registers and retime the circuit. AES encryption achieved a speedup of 2.4 for a 5-slow by hand implementation. Smith/Waterman achieved a speedup of 2.2 for a 4-slow by hand implementation. And, the LEON 1 SPARC microprocessor core achieved a speedup of 2.0 for a 2-slow automatically $C$-slowed design implementation. In 2007, Su et al. applied $C$-slow to an LDPC decoder for a throughput-area efficient design [41]. In 2012, Akram et al. applied $C$-slow to a processor to execute multiple threads in parallel using a single datapath of an instruction set processing element. For a 3-slow microprogrammed finite-state machine, a speedup of 2.59 times in clock frequency was achieved [42].

## 2.6 Hardware virtualization

Plessl and Platzner in 2004 [43] wrote a survey paper on hardware virtualization on FPGAs where they described three different approaches: temporal partitioning, virtualized execution, and virtual machine. Chuang in 2008 [44] described another type of temporal partitioning whereby hardware logic can be reused by temporally shared state. A diagram illustrating the hardware virtualization approaches is shown in Figure 2.4.

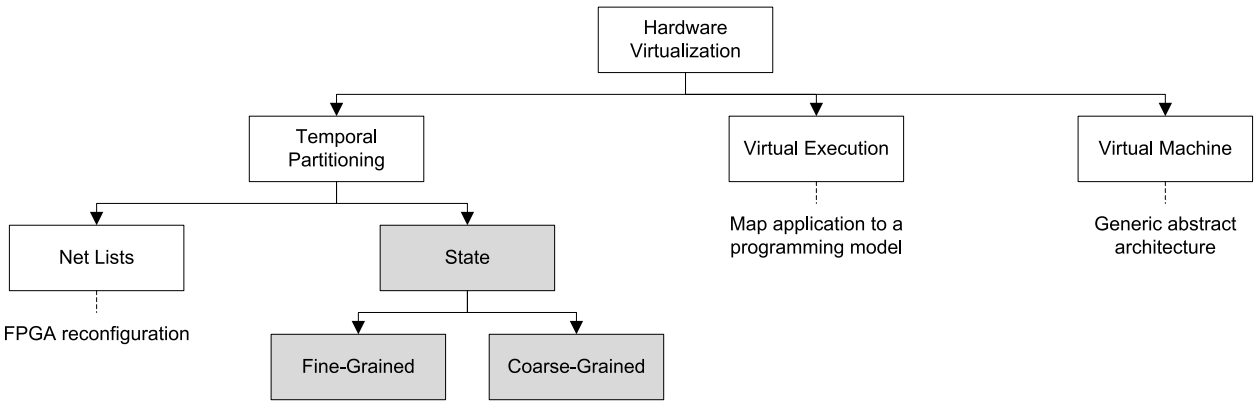



Figure 2.4: Hardware virtualization approaches. White boxes are described by Plessl and Platzner [43]. Gray boxes are described by Chuang [44].

**Temporal partitioning of net lists [43]:** This is described by Plessl and Platzner as a technique for virtualizing a hardware design described by a net list that would otherwise be too large to physically fit onto an FPGA. This is done by partitioning the net list and swapping it like virtual memory. Only one part of the computation is allowed to run at a time. Each part must, therefore, run sequentially to perform the complete computation with logic reconfiguration done between parts. This allows the design to better utilize resources on the FPGA and gives the illusion of having more resources than are physically available.

**Temporal partitioning of state [44]:** This is a way to share hardware by temporally swapping its state so as to compute multiple streams of computations on the same hardware (i.e., a single net list). The logic is fixed, and the state is swapped (context switched), allowing it to operate on independent streams. The context switch can be either fine- or coarse-grain. Fine-grain context switching is done by applying the $C$-slow transformation on the hardware logic, allowing different contexts to be processed in each pipeline stage of the $C$-slowed hardware. Coarse-grain context switching is done by swapping the state infrequently to and from a memory. Temporal partitioning of state is the type of virtualization that this dissertation investigates in Chapter 5.

**Virtualized execution [43]:** This is where a programming model is used to specify applications. Any application developed in this programming model can run on any hardware that supports this model of execution. One example programming model is the instruction set of a processor. Code written for this instruction set can execute on any processor that supports the instruction set. Another example is PipeRench [45], which has a pipelined streaming programming model where the application is decomposed into stripes and executes in a pipeline. An unlimited number of virtual stripes are supported by context switching the stripes in a pipelined manner on the hardware. Therefore, any hardware computation that is written for PipeRench can be temporally context switched in this virtual execution and can be arbitrarily large.

**Virtual machine [43]:** This is different from the rest in that it does not perform any kind of context switching of the hardware. A virtual machine defines a generic abstract architecture that hardware can be designed on. Designs targeted to a generic FPGA architecture are remapped to the actual architecture of a specific FPGA device.

## 2.7 Queueing notation

Queueing theory is an analytical study of waiting in queues. It models queue occupancy of different types of queues to predict queue lengths and waiting times. Kendall introduced a notation to classify queueing models [46]. This notation is of the form A/S/$c$ where A is the arrival process, S is the service time distribution, and $c$ is the number of servers.

**Arrival process (A):** The arrival process describes the arrival of jobs into the system. If jobs arrive at times $t_1$, $t_2$, ..., $t_j$, then interarrival times can be defined as $\tau_j = t_j - t_{j-1}$.

The interarrival times can be described as a random variable with a given distribution. It is generally assumed that the interarrival times, $\tau_j$, are independent and identically distributed (IID). The most common arrival process is Poisson with interarrival times that are IID and exponentially distributed [47].

**Service time distribution (S):** The time spent receiving service is called the service time. This is commonly modeled as a random variable that is IID with a given distribution.

**Number of servers ($c$):** The system may have one or more servers. This parameter sets the number of servers that are providing service in the queueing system.

The distributions for the interarrival and service times are denoted by a one letter symbol. A few of them listed are as follows:

| | |
|---|---|
| M | Exponential |
| D | Deterministic |
| G | General |

The exponential distribution is denoted with an M because it is memoryless, also called Markovian. Using Kendall's notation, we can describe a queue that has a *M*arkovian, or memoryless, arrival process; *D*eterministic service process; and *1* server as M/D/1.

## 2.8 Vacation model

The vacation model is an approach to analyzing systems where the server is not continuously available (e.g., the server is serving some other job). Bertsekas and Gallager [48] describe M/G/1 queues (*M*arkovian, or memoryless, arrival process; *G*eneral service process; 1 server) where the server can go on "vacation" for some random interval of time. This is illustrated in

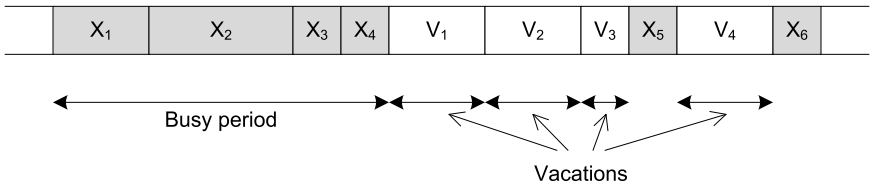



Figure 2.5: M/G/1 queueing model with vacations. During busy periods, the server is servicing jobs. During vacation periods, the server is "away" and jobs may be waiting in the queue.

Figure 2.5. Here, $X_j$ represents the service time of the $j$th job and $V_k$ represents the vacation time of the $k$th vacation. The model is as follows. When a job is waiting in the queue, the server will begin servicing the job and enter a busy period represented by $X_j$. When the queue is empty, the server will go on vacation and enter a vacation period represented by $V_k$. If a new arrival enters an idle system, rather than going immediately into service, it waits for the end of a vacation period, and then enters service. If there is no new arrival into the system, then the server, after returning from a vacation, will immediately go into another vacation period.

# Chapter 3

# MTJ Read Circuit Theory

Magnetic tunnel junction (MTJ) devices can be used in clocking and/or logic circuits, operating as sensors (to sense a magnetic field), memory (to store state), or logic gates (to evaluate logic expressions). For each of these uses, a read circuit is necessary to sense the resistance state of the MTJ device and convert it to a voltage. Our goal is to design a resistance-to-voltage read circuit for sensing the resistance state of the MTJ device and to produce a logic voltage output. We want this circuit to operate correctly and to perform fast in the presence of large input capacitance (which may come from off-chip wire bonding). We will be using this circuit in a prototype chip, which will be discussed further in Chapter 4. In this chapter, MTJ read circuit theory is defined as including both analytical and simulation results.

There are several basic designs for a resistance-to-voltage (R2V) read circuit. The read circuit can operate in the voltage- or current-mode, and it can have single or differential inputs. A voltage-mode read circuit operates by applying a constant current through the MTJ resistance and comparing the voltage output using a comparator to produce a logic voltage output. A current-mode read circuit, in contrast, operates by applying a constant voltage over the MTJ resistance and comparing the current output using a comparator to

produce a logic voltage output. A single input read circuit reads a single MTJ resistance and compares its value to a given threshold. A differential input read circuit, however, reads two complementary MTJ resistances (that have opposite resistance states) and compares them to each other to determine the logic voltage output.

In this chapter, we will design the resistance-to-voltage read circuit to operate in current-mode and use a single resistance input. Current-mode avoids actively charging and discharging the MTJ node and therefore can operate fast in the presence of large input capacitance (this will be shown in the analysis in Section 3.1). Reading a single input and thresholding it gives us more flexibility during experimentation of a prototype chip. The development of the read circuit will be presented in three parts. First, in Section 3.1, we will develop and analyze a current-mode read circuit (that reads an input resistance and produces an output current). Second, in Section 3.2, we will analyze several different current conveyor structures for sizing the transistors and improving the read circuit performance through the use of cascode structures. Last, in Section 3.3, we will present a completed design for the read circuit that uses a P-cascode current conveyor and includes an additional current comparator (to compare the output current to a threshold current) and output buffer to produce the desired logic voltage output. The read circuit performance is then characterized in simulation.

## 3.1 Analysis of a current-mode read circuit

To operate in current-mode, a current conveyor circuit is used. A current conveyor [35] is a current-mode circuit which can be used for reading the state of an MTJ device. It operates by clamping the voltage, $V_{bias}$, across the MTJ allowing the current $I = \frac{V_{bias}}{R_{mtj}}$ to flow through the MTJ. This current is then mirrored to the output, optionally with amplification, in order to use downstream.

Noise in the read circuit can affect the ability to correctly distinguish the logical state (0 or 1) of the MTJ device. It is therefore necessary to analyze circuit noise and to establish noise margins at the output for reliable operation.

A simple set of noise equations aimed at a broad family of circuits (both class A and AB current conveyors) were presented in [49] and then used to make qualitative conclusions. We present a noise analysis of a class A current conveyor circuit that is used for sensing the MTJ resistance. The resultant equations allow us to predict the output noise for the current conveyor and use them as design equations for tuning circuit parameters to meet noise performance requirements. First, the methodology employed in the noise analysis is given. Second, the noise analysis of the current-mode read circuit is described. Third, the analytical expressions derived in the noise analysis are validated via simulation. Fourth, design guidance into how to tune characteristics of the circuit for performance is described, and last, a general discussion of the noise model is given.

### 3.1.1 Methodology

The methodology used to perform the noise analysis of the current conveyor under consideration, shown in Figure 3.1, involves several steps. The first step is to extend the circuit with current sources added in parallel with every transistor and resistor that models the noise current of these circuit components. The second step is then to construct a small-signal model of the circuit from which a frequency-independent DC transfer function, $H_{dc,i}$, is derived for each current source $i$ to the output, making approximations as allowed to get a simplified form. The noise spectrum, $S_i(f)$, of each source is referred to the output, $S_{out,i}(f)$, by the equation

$$S_{out,i}(f) = |H_i(s)|^2 \cdot S_i(f) \tag{3.1}$$

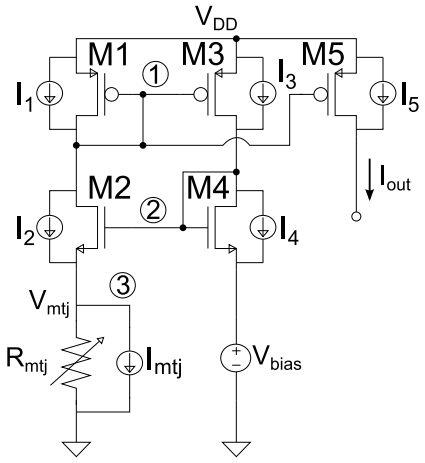



Figure 3.1: Noise model of the current-mode read circuit for sensing MTJ resistance. Current sources, $I_{1-5}$ and $I_{mtj}$, are added in parallel with transistors $M_{1-5}$ and resistor $R_{mtj}$ to model the noise contribution of each component, respectively. Circuit nodes $1-3$, which are referred to in the analysis, are labeled with a circled number.

where $s = j2\pi f$. At DC, $S_{out,i}(0) = |H_{dc,i}|^2 S_i(0)$.

The third step is to determine the pole frequency, $f_{p,m}$, at each node $m$ which limits the bandwidth of the circuit and bounds the circuit noise. The pole frequency requires determining the resistance and capacitance seen at each node:

$$f_{p,m} = \frac{1}{2\pi R_m C_m}. \tag{3.2}$$

The resistance seen at a node is determined by applying a test current, $I_{test}$, into the node and calculating

$$R_m = \frac{V_m}{I_{test}} \tag{3.3}$$

where $V_m$ is the voltage at the node. The capacitance seen at a node is determined by adding up all capacitance contributions at the node to ground. Capacitances that are coupled to other nodes are transformed to Miller capacitances that are a function of the capacitance and the small-signal voltage gain between the two nodes [37].

The frequency-dependent transfer function $H_i(s)$ is a 3-pole system that can be modeled as

$$H_i(s) = \frac{H_{dc,i}}{\left(1+\frac{s}{2\pi f_{p1}}\right)\left(1+\frac{s}{2\pi f_{p2}}\right)\left(1+\frac{s}{2\pi f_{p3}}\right)}. \tag{3.4}$$

The fourth step is to calculate the total integrated noise power, $\overline{I^2_{out,i,n}}$, of each source referred to the output:

$$\overline{I^2_{out,i,n}} = \int_{f=0}^{f=\infty} |H_i(j2\pi f)|^2 S_i(f)\, df. \tag{3.5}$$

There are two predominant noise spectra: thermal and $1/f$ noise. We only consider thermal noise for simplicity in validating our results. Thermal noise has a flat noise spectrum independent of frequency and $1/f$ noise has a spectrum with an inverse relationship to frequency. For a resistor, the thermal noise spectrum is $S_{i,t} = \frac{4kT}{R}\left[\frac{\mathrm{A}^2}{\mathrm{Hz}}\right]$ [37], where $k$ is Boltzmann's constant, $T$ is the temperature, and $R$ is the resistance. For a transistor, the thermal noise spectrum is $S_{i,t} = 4kT\gamma g_m\left[\frac{\mathrm{A}^2}{\mathrm{Hz}}\right]$ [37], where $\gamma$ is a coefficient that is typically equal to $2/3$ for long-channel transistors, and $g_m$ is the transconductance of the transistor.

The fifth step is to add all noise power contributions at the output to get the total noise power, $\overline{I^2_{out,tot,n}}$:

$$\overline{I^2_{out,tot,n}} = \sum_i \overline{I^2_{out,i,n}}. \tag{3.6}$$

The square root of the noise power gives the standard deviation of the noise current present at the output.

### 3.1.2 Noise analysis

The current-mode read circuit used for sensing the resistance of an MTJ device, as shown in Figure 3.1, is constructed using a current conveyor. The current conveyor ($M_{1-4}$) clamps the voltage, $V_{bias}$, from the source of $M_4$ to the source of $M_2$ across the resistance, $R_{mtj}$. This voltage produces a current through $R_{mtj}$ which is mirrored to the output using a current-mirror formed by $M_{1,5}$ with unity gain. Each transistor is a current source operating in the saturation region.

The small-signal model of the current-mode read circuit was constructed with several simplifications. First, all NMOS transistors $M_{2,4}$ have small signal parameters $g_{m,n}$ and $g_{ds,n}$, and all PMOS transistors $M_{1,3,5}$ have small signal parameters $g_{m,p}$ and $g_{sd,p}$, since the NMOS and PMOS transistors have the same aspect ratios $\left(\frac{W}{L}\right)_n$ and $\left(\frac{W}{L}\right)_p$, respectively, and the same drain currents. Second, the bulk-modulation effect of transistors $M_{2,4}$ is neglected. Third, the resistance $R_{mtj}$ is represented as a conductance, $g_{mtj} = \frac{1}{R_{mtj}}$.

From the small-signal model, the DC transfer function of each noise source, modeled by current sources $I_{1-5}$ and $I_{mtj}$ in Figure 3.1, to the output are presented in Table 3.1. In the table, two sets of equations are derived. The high accuracy equations are approximated with the assumptions $\{g_{m,n} \vee g_{m,p}\} \gg \{g_{ds,n} \vee g_{sd,p}\}$. The simple equations are approximated with additional assumptions that $g_{mtj} \lesssim \{g_{m,n} \vee g_{m,p}\}$ (i.e., of the same order or smaller). The transfer function is derived by considering one noise source at a time and calculating the current gain $H_{dc,i} = \frac{I_{out}}{I_i}$ from that source to the output. The output noise spectrum is then referred to the output by (3.1) at DC.

Next, the pole frequencies, $f_{p,1-3}$, are determined by deriving $R$ and $C$ at each node. The derived resistance equations for the current-mode read circuit are presented in Table 3.2. In

Table 3.1: DC transfer functions from noise source to output. The high accuracy equations are approximated with the assumptions $\{g_{m,n} \vee g_{m,p}\} \gg \{g_{ds,n} \vee g_{sd,p}\}$. The simple equations are approximated with additional assumptions $g_{mtj} \lesssim \{g_{m,n} \vee g_{m,p}\}$ (i.e., of the same order or smaller).

| | **Equations** | |
|---|---|---|
| **Term** | **High Accuracy** | **Simple** |
| $H_{dc,M1}$ | $\frac{-1}{\frac{g_{m,n}}{g_{m,n}+g_{mtj}}+g_{mtj}\frac{g_{ds,n}+g_{sd,p}}{g_{m,n}+g_{mtj}}\left(\frac{1}{g_{m,n}}+\frac{1}{g_{m,p}}\right)}$ | $-\left(1+\frac{g_{mtj}}{g_{m,n}}\right)$ |
| $H_{dc,M2}$ | $\frac{1}{\frac{g_{m,n}}{g_{mtj}}+\left(g_{ds,n}+g_{sd,p}\right)\left(\frac{1}{g_{m,n}}+\frac{1}{g_{m,p}}\right)}$ | $\frac{g_{mtj}}{g_{m,n}}$ |
| $H_{dc,M3}$ | $H_{dc,M2}$ | $H_{dc,M2}$ |
| $H_{dc,M4}$ | $-H_{dc,M2}$ | $-H_{dc,M2}$ |
| $H_{dc,M5}$ | 1 (exact) | 1 (exact) |
| $H_{dc,Rmtj}$ | $\frac{1}{1+g_{mtj}\frac{g_{ds,n}+g_{sd,p}}{g_{m,n}}\left(\frac{1}{g_{m,n}}+\frac{1}{g_{m,p}}\right)}$ | 1 |

this table, as before, two sets of equations are derived: high accuracy and simple. The same approximations, previously described, were applied to derive these equations.

The capacitance seen at each node is determined by considering the gate-to-source ($C_{gs}$), gate-to-drain ($C_{gd}$), source-to-bulk ($C_{sb}$), and drain-to-bulk ($C_{db}$) capacitances of each transistor connected to the node and the capacitance associated with the MTJ ($C_{mtj}$). The gate-to-bulk ($C_{gb}$) capacitance is neglected because all transistors are operating in saturation. Capacitances that are connected to a small-signal ground are simply added together. Those capacitances that are coupled to other nodes are replaced with corresponding Miller capacitances that are a function of the capacitance and the small-signal voltage gain ($\frac{v_{out}}{v_{in}}$) between the two nodes [37]. For a prototype chip, wire bond capacitance associated with the MTJ device is accounted for in $C_{mtj}$ which is typically about 3 pF [37]. The derived capacitance equations are presented in Table 3.3.

Table 3.2: Resistance at each node. The high accuracy equations are approximated with the assumptions $\{g_{m,n} \vee g_{m,p}\} \gg \{g_{ds,n} \vee g_{sd,p}\}$. The simple equations are approximated with additional assumptions $g_{mtj} \lesssim \{g_{m,n} \vee g_{m,p}\}$ (i.e., of the same order or smaller).

| | **Equations** | |
|---|---|---|
| **Resistance** | **High Accuracy** | **Simple** |
| $R_1$ | $\frac{1}{\frac{g_{m,n}g_{m,p}}{g_{m,n}+g_{mtj}}+\frac{g_{mtj}g_{sd,p}}{g_{m,n}+g_{mtj}}\left(1+\frac{g_{m,p}}{g_{m,n}}\right)+g_{ds,n}\left(\frac{g_{m,p}}{g_{m,n}}+\frac{g_{m,p}+g_{mtj}}{g_{m,n}+g_{mtj}}\right)}$ | $\frac{g_{m,n}+g_{mtj}}{g_{m,n}g_{m,p}}$ |
| $R_2$ | $\frac{1}{\frac{g_{m,n}^2}{g_{m,n}+g_{mtj}}+\left(g_{ds,n}+g_{sd,p}\right)\left(1+\frac{g_{m,n}g_{mtj}}{\left(g_{m,n}+g_{mtj}\right)g_{m,p}}\right)}$ | $\frac{g_{m,n}+g_{mtj}}{g_{m,n}^2}$ |
| $R_3$ | $R_{mtj} \parallel \left(\frac{\left(g_{ds,n}+g_{sd,p}\right)(g_{m,n}+g_{m,p})}{g_{m,p}g_{m,n}^2}\right)$ | $\frac{\left(g_{ds,n}+g_{sd,p}\right)(g_{m,n}+g_{m,p})}{g_{m,p}g_{m,n}^2}$ |

Table 3.3: Capacitance at each node.

| **Capacitance** | **Equation** |
|---|---|
| $C_1$ | $C_{gs1} + C_{db1}$ $+ C_{db2} + C_{gd2}\left(1 - \frac{v_{g2}}{v_{d2}}\right)$ $+ C_{gs3} + C_{gd3}\left(1 - \frac{v_{d3}}{v_{g3}}\right)$ $+ C_{gs5} + C_{gd5}$ |
| $C_2$ | $C_{gd2}\left(1 - \frac{v_{d2}}{v_{g2}}\right) + C_{gs2}\left(1 - \frac{v_{s2}}{v_{g2}}\right)$ $+ C_{db4} + C_{gs4}$ $+ C_{db3} + C_{gd3}\left(1 - \frac{v_{g3}}{v_{d3}}\right)$ |
| $C_3$ | $C_{mtj} + C_{sb2} + C_{gs2}\left(1 - \frac{v_{g2}}{v_{s2}}\right)$ |

The pole frequency at each node is then calculated using (3.2). Using the frequency-dependent transfer function in (3.4), the total integrated noise referred at the output is calculated by (3.5). For thermal noise, which has a flat noise spectrum independent of $f$, this is equal to

$$\overline{I_{out,i,nt}^2} = |H_{dc,i}|^2 \cdot S_{i,t} \cdot B_n \tag{3.7}$$

where the noise bandwidth [37], $B_n$, is

$$B_n = \frac{f_{p1} f_{p2} f_{p3} \left(f_{p1} + f_{p2} + f_{p3}\right)}{\left(f_{p1} + f_{p2}\right)\left(f_{p1} + f_{p3}\right)\left(f_{p2} + f_{p3}\right)} \frac{\pi}{2}. \tag{3.8}$$

### 3.1.3 Validation

We simulated the current-mode read circuit of Figure 3.1 using a 3 metal 2 poly (3M2P) 0.5 $\mu$m process model to verify the equations derived in Section 3.1.2. The circuit is modeled in the Cadence Design Environment and simulated using Spectre. Small-signal parameters, $g_m$ and $g_{ds}$, and capacitances $C_{gs}$, $C_{gd}$, $C_{sb}$, and $C_{db}$ are extracted from the DC operating points of the circuit. Small-signal voltage gains $\frac{v_{d2}}{v_{g2}}$, $\frac{v_{d3}}{v_{g3}}$, and $\frac{v_{s2}}{v_{g2}}$ are determined in an AC simulation by applying an AC test current in the circuit and measuring the AC voltage gain and phase between two nodes at DC (very low frequency). The sign of the gain is determined from the phase at DC, which is either 0° (positive gain) or 180° (negative gain). These parameters are then used in the analytical expressions for the output noise. The noise of devices $M_{1-5}$ and $R_{mtj}$ are modeled with thermal noise only. In the equation of the thermal noise spectrum, $S_t$, of a transistor, $\gamma = 1.0$ is chosen in agreement with simulation. The bias voltage, $V_{bias}$, across the MTJ is chosen to be 0.1 V in order to maintain a large tunneling magnetoresistance ratio (TMR). $R_{mtj} = 1\,\text{k}\Omega$ unless specified otherwise.

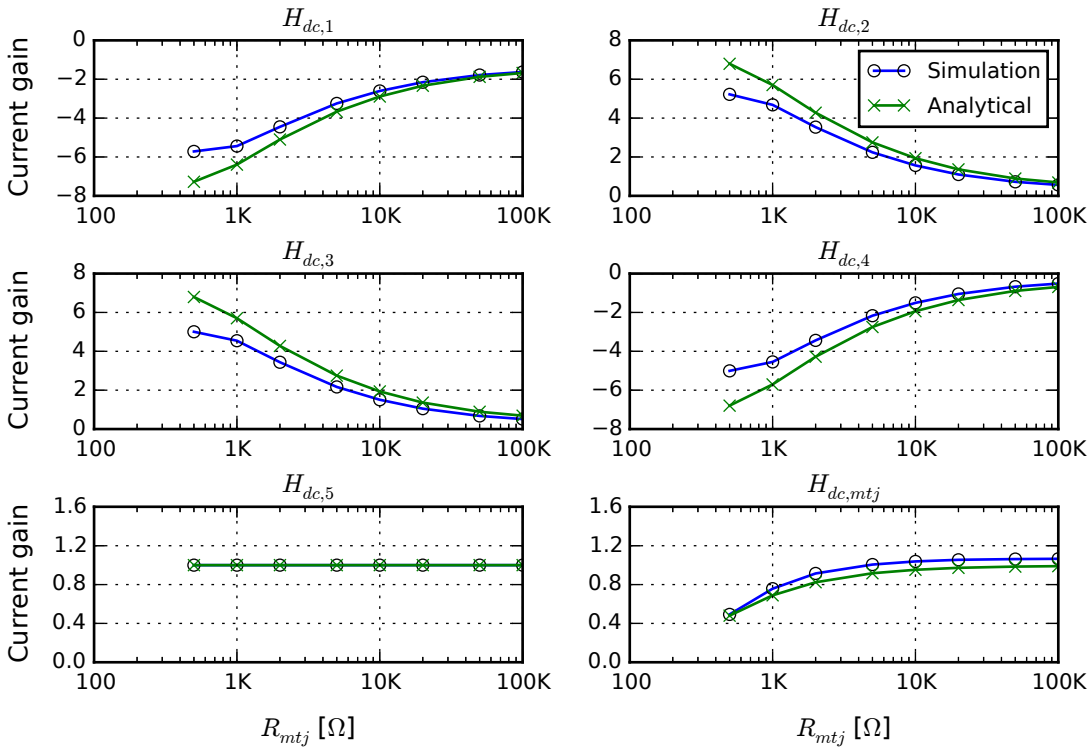



Figure 3.2: DC transfer function validation with analytical and simulation results. Six plots are presented, one for each transfer equation, $H_{1-5}$ and $H_{mtj}$, corresponding to the current gain from each current source to the output at DC. The X-axis shows the MTJ resistance, $R_{mtj}$, swept from 500 Ω to 100 kΩ. The Y-axis shows the current gain for simulation (blue with a ∘) and analytical (green with an ×) results.

### DC Transfer Function Validation

The DC transfer functions in Table 3.1 are validated in Figure 3.2 with analytical and simulation results. Simulation results are attained by measuring the current gain of current source, $I_i$, to the output in an AC analysis. The results are plotted versus the MTJ resistance, $R_{mtj}$, varied from 500 Ω to 100 kΩ. The six plots show the DC transfer function $H_{dc,i}$ for each source. The analytical results were derived using first-order approximations which show the same trend as simulation and remain within 3 dB throughout the range of MTJ resistances.

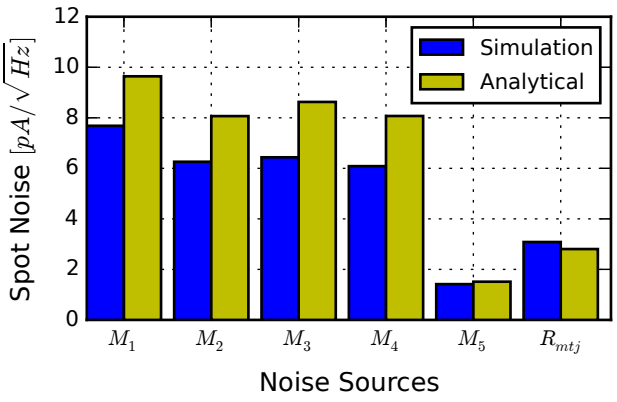



Figure 3.3: Thermal noise spectrum validation with analytical and simulation results. The noise spectrum of each source referred to the output is evaluated in simulation at 1 kHz.

### DC Spot Noise Validation

The thermal noise spectrum of each noise source is validated in Figure 3.3 with analytical and simulation results. Analytical results are attained by calculating the thermal noise spectrum of each noise source and referring it to the output using (3.1) at DC. Simulation results are attained by measuring the spot noise of each circuit component, modeled with thermal noise, at 1 kHz. The analytical results are shown to be within 3 dB of the simulation results for each noise source.

### Node Resistance Validation

The resistance equations in Table 3.2 are validated in Table 3.4 with analytical and simulation results. Simulation results are attained by applying an AC test current into the node in the small-signal model and calculating the resistance by (3.3). The analytical results are in agreement with the simulation results for each node.

Table 3.4: Validation of node resistances.

| **Result** | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Simulation | 39.4 kΩ | 44.6 kΩ | 231 Ω |
| Analytical | 45.9 kΩ | 51.9 kΩ | 311 Ω |

Table 3.5: Analytical calculation of pole frequencies.

| **Node** $m$ | $R_m$ | $C_m$ | $f_{p,m}$ |
|---|---|---|---|
| 1 | 45.9 kΩ | 136 fF | 25.5 MHz |
| 2 | 51.8 kΩ | 48 fF | 63.9 MHz |
| 3 | 311 Ω | 3.28 pF | 156 MHz |

**Dominant Pole Validation**

The dominant pole is the lowest frequency pole in the circuit. This can be determined by measuring the 3 dB bandwidth at the output. In simulation, we evaluated the dominant pole of the current-mode read circuit to be 23.6 MHz. To compare this to analytical results, we then calculated the resistance and capacitance seen at each node (assuming $C_{mtj} = 3$ pF for wire bond capacitance) and using these, determined the corresponding pole frequencies, $f_{p,m}$, using (3.2). The results are shown in Table 3.5. We can observe from the table that the analytical dominant pole is 25.5 MHz, which agrees well with 23.6 MHz from simulation. Further, using (3.8), we can determine the noise bandwidth from the calculated pole frequencies to be 27.5 MHz.

Wire bonding capacitance associated with the MTJ ($C_{mtj}$) will add to the capacitance on node 3 which we will assume dominants the node capacitance. A simple analysis using (3.2)

gives the minimum capacitance necessary to form a dominant pole on this node.:

$$C_{mtj} > \frac{1}{2\pi R_{3,sim} BW_{3\,\mathrm{dB},sim}} \tag{3.9}$$
$$\gtrapprox 29\,\mathrm{pF}$$

where $R_{3,sim} = 231\,\Omega$ is the resistance in simulation at node 3 and $BW_{3\,\mathrm{dB},sim} = 23.6\,\mathrm{MHz}$ is the 3 dB bandwidth in simulation at the output corresponding to the dominant pole.

**Total Integrated Output Noise Validation**

The total integrated output noise is validated in Figure 3.4 with analytical and simulation results. Analytical results are attained by calculating the thermal noise spectrum of each noise source, referring it to the output, integrating the spectrum using (3.7), and then computing the square root to get the noise current. Simulation results are attained by measuring the total integrated noise of each noise source, modeled with thermal noise, from 1 Hz to 1 GHz to get the noise power and then again computing the square root to get the noise current. The noise contribution of $M_5$ is not bounded because the current source $I_5$ is connected directly in series with the output in the noise model. A downstream circuit will, however, bound this noise term. The total noise of all noise sources is computed using (3.6) and is presented in the last column of the plot as a noise current. The analytical results differ from the simulation results for two reasons. One, a higher DC current gain, except $I_{mtj}$, is predicted as compared to simulation giving an increase in the total integrated output noise. Two, a higher dominant pole frequency is predicted as compared to simulation also contributing to an increase in the total integrated output noise. The results are within 2.5 dB of simulation for each noise source.

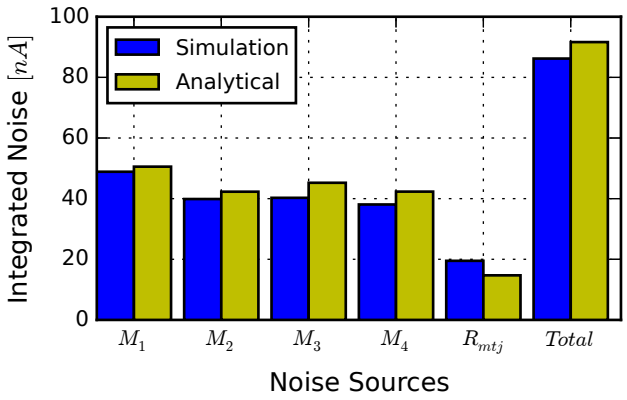



Figure 3.4: Total integrated output noise validation with analytical and simulation results. The noise spectrum of each source referred to the output is integrated from 1 Hz to 1 GHz.

### 3.1.4 Design guidance

The equations derived in the analytical model characterize the circuit and provide insight into how input parameters affect the circuit performance. However, even with approximations, they are still fairly complex. We can greatly simplify our intuition within the noise analysis by determining trends and relationships between input parameters and output circuit characteristics. This is shown summarized in Table 3.6.

In the table, input parameters are shown along the columns and output circuit characteristics are shown along the rows. Of the input parameters, $L$, $W$, and $V_{bias}$ are circuit parameters that the designer can adjust to tune circuit performance. $R_{mtj}$ is an external parameter determined by the MTJ resistance that the designer may not have direct control over its value. For the output circuit characteristics, a designer may be interested in tuning circuit bandwidth determined by the dominant pole at $f_{p1}$, tuning the range of input capacitances of the MTJ node determined by $C_{mtj,dp,min}$, and tuning total integrated output noise determined by $\sqrt{\overline{I^2_{out,tot,n}}}$.

Table 3.6: Design guidance for tuning circuit performance. Output circuit characteristics are given along the rows and input parameters along the columns. $R_{mtj}$ is an external parameter that the designer may not have direct control over. In the table, "S" means strong and "w" weak. "↑" means they trend together in the same direction. "↓" means they trend inversely to each other. A blank means the parameter may not have a significant impact.

| **Output Circuit Characteristic** | **Input Parameters** | | | |
|---|---|---|---|---|
| | $L$ | $W$ | $V_{bias}$ | $R_{mtj}$ |
| $f_{p1}$ | S↓ | w↓ | S↑ | w↓ |
| $R_3$ | | S↓ | | |
| $C_{mtj,dp,min}$ | S↑ | S↑ | S↓ | |
| $\sqrt{\overline{I^2_{out,tot,n}}}$ | S↓ | w↓ | w↑ | S↓ |

In the cross-section between the input parameters and output characteristics, the following notation is used. "S" means strong and "w" weak which indicates that the output characteristic is either strongly or weakly dependent on the input parameter. "↑" means that the output characteristic trends in the same direction as the input parameter. "↓" means that they trend inversely to each other. A blank means the parameter may not have a significant impact.

Input capacitance is relatively insensitive to the bandwidth of the circuit since node 1 forms the dominant pole. The term $C_{mtj,dp,min}$ is the minimum capacitance needed to form a dominant pole on node 3 containing the MTJ as determined in (3.9) and illustrates the range of input capacitances that can be present without affecting the circuit bandwidth. This term is a function of the $R_3$ node resistance and $f_{p1}$ dominant pole (which primarily determines the circuit bandwidth). As shown in the table, $C_{mtj,dp,min}$ can be increased by increasing $L$ or $W$ or decreasing $V_{bias}$. However, increasing $L$ or decreasing $V_{bias}$ will also drop the bandwidth. Therefore, tuning $W$ only will keep the bandwidth relatively constant (it only weakly decreases bandwidth) while increasing the input capacitance range.

The bandwidth, as determined primarily by the dominant pole $f_{p1}$, can be increased by decreasing $L$, $W$, or $R_{mtj}$, or increasing $V_{bias}$. It is strongly dependent on $L$ and $V_{bias}$ while only weakly dependent on $W$ and $R_{mtj}$. $V_{bias}$ strongly affects the bandwidth because it adjusts the $I_{ds}$ and $I_{sd}$ currents of the transistors. $R_{mtj}$ weakly affects the bandwidth due to some partial cancellation of the $R_{mtj}$ term.

The total integrated output noise, $\sqrt{\overline{I^2_{out,tot,n}}}$, can be decreased by increasing $L$, $W$, or $R_{mtj}$, or decreasing $V_{bias}$. It is strongly dependent on $L$ and $R_{mtj}$ while only weakly dependent on $W$ and $V_{bias}$.

The design guidance developed in the table was derived from the analytical equations and empirically validated from the analytical model using a $2^k$ factorial experimental design [47]. The experimental design is a statistical technique that analyzes a system containing $k$ factors with 2 levels per factor and determines variations in the output metrics due to each factor and their interactions. Validation of the design guidance is given further in Appendix A.

### 3.1.5 Discussion

We have provided a noise analysis of a current-mode read circuit for a magnetic tunnel junction device. The noise analysis presented here is only valid for nodes with low-pass transfer functions. The analytical expressions for noise at the output are validated against simulation results for the CMOS process we will be using for prototype fabrication.

While the numerical noise values presented in the simulation validation are limited to thermal noise for simplicity, the methodology generalizes for an arbitrary noise spectrum, $S_i(f)$. There are well known expressions for $S_i(f)$ to model $1/f$ noise, and as a general noise model becomes available for an MTJ device, it can be substituted for $S_i(f)$ as well.

One of the uses of this type of noise analysis is to bound the operating points of the system in terms of a noise margin. With an MTJ resistance of $R_H = 1\,\mathrm{k}\Omega$, the read circuit will output a low current, $I_L = \frac{V_{bias}}{R_H} = 100\,\mu\mathrm{A}$, and with an MTJ resistance of $R_L = 500\,\Omega$, the read circuit will output a high current, $I_H = 200\,\mu\mathrm{A}$. For this case, $I_H - I_L \gg 10\sigma$, which indicates that thermal noise in this circuit is not significantly limiting the noise margin. Note that noise margins must also take into account part-to-part variations and externally induced noise in addition to the internal noise analyzed here. The noise current present at the output bounds how close these two quantities can be, while still maintaining a safe distance between them, e.g., we might choose to bound $I_H - I_L > 6\sigma$.

One of the advantages of current-mode operation is the relative insensitivity to load capacitance that might be present connecting the MTJ device to the CMOS circuit. While production fabrication is integrated, our prototype will use wire bonding. Due to the low impedance at node 3 in Figure 3.1 (relative to node 1), parasitic capacitance of 1 to 5 pF will not significantly impact the noise performance or bandwidth of the circuit. This is consistent with node 1 rather than node 3 being the dominant pole in the circuit.

The analysis of the current-mode read circuit was simplified by using a non-cascoded current conveyor. The operation of the current conveyor circuit, however, can be improved by adding cascode structures. In the next section, a number of current conveyor structures will be discussed and analyzed, first starting with the non-cascoded structure and then building upon it with cascoded structures.

## 3.2 Current conveyor structures

The current-mode read circuit, previously discussed, uses a current conveyor to pin the voltage across an MTJ and read its resistance as a current. There are several variations of the current conveyor circuit that trade-off between area, speed, and linearity.

The current conveyor, for example shown in Figure 3.5, consists of two current mirror circuits that mirror the current between the left and right branches in a feedback loop [36]. The transistors in the current mirror act as current sources by operating in the saturation-mode. Their output can be modeled as an ideal current source in parallel with an output resistance. The higher the output resistance, the more closely the current source acts like an ideal source which improves linearity of the circuit. The output resistance of a current source can be improved in two ways. One, by increasing length $L$ of the transistor (which reduces the effects due to channel length modulation), and two, by adding a cascode stage to the current mirror circuit (which multiplies the output resistance) [37].

In this section, we consider basic, P-cascode, and NP-cascode current conveyors and develop design equations for sizing the transistors appropriately so that they operate in saturation-mode given a target $R_{mtj}$ resistance and $V_{bias}$ voltage. We further generalize the design equations for any number of N or P cascode stages added to the basic current conveyor circuit.

### 3.2.1 Basic

The basic current conveyor structure, shown in Figure 3.5, is formed by transistors $M_{1-4}$. Transistors $M_{1-2}$ form the PMOS current mirror circuit and transistors $M_{3-4}$ form the NMOS current mirror circuit. These are connected together in a feedback topology that clamps the

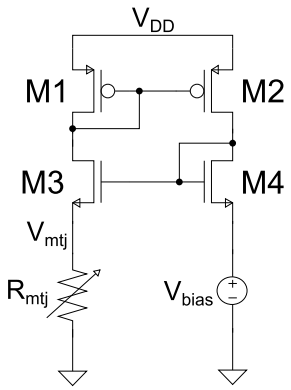



Figure 3.5: Basic current conveyor.

voltage $V_{mtj}$ to $V_{bias}$ potential, and, applying Ohm's Law, produces an $I_{mtj}$ current down through the left branch that is inversely-proportional to the input resistance by the equation $I_{mtj} = {}^{V_{bias}}/_{R_{mtj}}$. The currents are mirrored between branches with a 1:1 gain which is achieved by matching the widths and lengths of the transistors in the current mirrors [37], thus making $\left(\frac{W}{L}\right)_1 = \left(\frac{W}{L}\right)_2$, $\left(\frac{W}{L}\right)_3 = \left(\frac{W}{L}\right)_4$, and $I_{mtj} = I_{sd,1} = I_{sd,2} = I_{ds,3} = I_{ds,4}$.

We can now show that $V_{mtj} = V_{bias}$ by setting $I_{ds,3} = I_{ds,4}$:

$$\begin{aligned} \frac{1}{2}K_N\left(\frac{W}{L}\right)_3\left(V_{gs,3} - V_{TN}\right)^2 &= \frac{1}{2}K_N\left(\frac{W}{L}\right)_4\left(V_{gs,4} - V_{TN}\right)^2 \\ \left(V_{gs,3} - V_{TN}\right)^2 &= \left(V_{gs,4} - V_{TN}\right)^2 \\ V_{gs,3} &= V_{gs,4}. \end{aligned} \tag{3.10}$$

Then, equating $V_{g,3} = V_{g,4}$ (which are on the same node), we get

$$\begin{aligned} V_{mtj} + V_{gs,3} &= V_{bias} + V_{gs,4} \\ V_{mtj} &= V_{bias}. \end{aligned} \tag{3.11}$$

### Condition for saturation

In order for the transistors to operate in saturation-mode, they need to satisfy the condition for saturation [37]. For an NMOS transistor, the condition is

$$V_{ds} > V_{ds,sat} \text{ where } V_{ds,sat} = V_{gs} - V_{TN}. \tag{3.12}$$

And for a PMOS transistor, the condition is

$$V_{sd} > V_{sd,sat} \text{ where } V_{sd,sat} = V_{sg} - |V_{TP}|. \tag{3.13}$$

### Proof of saturation for diode-connected transistors

Diode-connected transistors have connected gate and drain terminals that force them to always be in saturation-mode. Transistors $M_{1,4}$ are diode-connected PMOS and NMOS transistors, respectively, and are therefore always in saturation. To prove this, set $V_d = V_g$ and then apply the condition for saturation to show that it is always true. For a diode-connected NMOS transistor,

$$\begin{aligned} V_{ds} &> V_{ds,sat} \\ V_{gs} &> V_{gs} - V_{TN} \\ V_{TN} &> 0. \end{aligned} \tag{3.14}$$

Since the threshold voltage $V_{TN}$ for the NMOS is always greater than 0 (which is true in the 3M2P 0.5 $\mu$m process we are targeting), this condition is always true. Thus, the transistor

is always in saturation. Similarly, for a diode-connected PMOS transistor,

$$\begin{aligned} V_{sd} &> V_{sd,sat} \\ V_{sg} &> V_{sg} + V_{TP} \\ V_{TP} &< 0. \end{aligned} \tag{3.15}$$

Again, since the threshold voltage $V_{TP}$ for the PMOS is always less than 0, this condition is always true. Thus, this transistor also is always in saturation. Note, it is typical to represent $V_{TP}$ as a positive number by taking the absolute value as $|V_{TP}|$. This allows equations for the NMOS and PMOS transistors to remain symmetrical and is the notation used in this chapter.

**Apply saturation condition for transistor** $M_3$

Next, we are going to apply the saturation condition for transistor $M_3$ and solve for the minimum shape factor that enables it operate in saturation-mode. To simplify the notation in this analysis, the shape factors $\left(\frac{W}{L}\right)_i$ will be replaced using the variable $s_i$. Also, we can define $V_{on}$ as the "on" voltage of a transistor, which is the extra voltage above a threshold that a transistor is turned on [50]. It is equivalent to the saturation voltage and is defined as $V_{on} = V_{ds,sat} = V_{gs} - V_{TN}$ for an NMOS and $V_{on} = V_{sd,sat} = V_{sg} - |V_{TP}|$ for a PMOS. For an NMOS transistor in saturation, the $V_{on}$ voltage can be determined from the $I_{ds}$ equation as

$$\begin{aligned} I_{ds} &= \frac{1}{2} K_N \cdot s \cdot (V_{gs} - V_{TN})^2 \\ I_{ds} &= \frac{1}{2} K_N \cdot s \cdot V_{on}^2 \\ V_{on} &= \sqrt{\frac{2I_{ds}}{K_N \cdot s}}. \end{aligned} \tag{3.16}$$

Likewise, for a PMOS, the $V_{on}$ voltage can be determined from the $I_{sd}$ equation as

$$\begin{aligned} I_{sd} &= \frac{1}{2} K_P \cdot s \cdot (V_{sg} - |V_{TP}|)^2 \\ I_{sd} &= \frac{1}{2} K_P \cdot s \cdot V_{on}^2 \\ V_{on} &= \sqrt{\frac{2I_{sd}}{K_P \cdot s}}. \end{aligned} \tag{3.17}$$

To apply the saturation condition, we need to determine the voltages $V_{on,3}$, $V_{d,3}$, and $V_{s,3}$. First, $V_{on,3}$ can be determined by setting $I_{ds,3}$ to $I_{mtj}$ and substituting it into back into (3.16):

$$I_{ds,3} = I_{mtj} = \frac{V_{bias}}{R_{mtj}}. \tag{3.18}$$

Thus,

$$V_{on,3} = \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_N \cdot s_3}}. \tag{3.19}$$

Next, we can determine $V_{d,3}$ by equating $I_{sd,1} = I_{ds,3}$ and solving for $V_{sg,1}$:

$$\begin{aligned} \frac{1}{2} K_P \cdot s_1 \cdot (V_{sg,1} - |V_{TP}|)^2 &= \frac{1}{2} K_N \cdot s_3 \cdot V_{on,3}^2 \\ V_{sg,1} &= \sqrt{\frac{K_N \cdot s_3}{K_P \cdot s_1}} V_{on,3} + |V_{TP}| \,. \end{aligned}$$

Then, we can calculate $V_{d,3}$ as

$$\begin{aligned} V_{d,3} &= V_{DD} - V_{sg,1} \\ V_{d,3} &= V_{DD} - \sqrt{\frac{K_N \cdot s_3}{K_P \cdot s_1}} V_{on,3} - |V_{TP}| \,. \end{aligned} \tag{3.20}$$

The source voltage, $V_{s,3}$, is simply $V_{bias}$. Applying the saturation conditions for $M_3$, we get

$$\begin{aligned} V_{ds,3} &> V_{ds,sat,3} \\ V_{d,3} - V_{bias} &> V_{on,3} \\ V_{DD} - \sqrt{\frac{K_N \cdot s_3}{K_P \cdot s_1}} V_{on,3} - |V_{TP}| - V_{bias} &> V_{on,3}. \end{aligned}$$

Rearranging terms,

$$V_{DD} - |V_{TP}| - V_{bias} > V_{on,3} \cdot \left(1 + \sqrt{\frac{K_N \cdot s_3}{K_P \cdot s_1}}\right).$$

Substituting $V_{on,3}$ using (3.19),

$$\begin{aligned} V_{DD} - |V_{TP}| - V_{bias} &> \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_N \cdot s_3}} \cdot \left(1 + \sqrt{\frac{K_N \cdot s_3}{K_P \cdot s_1}}\right) \\ V_{DD} - |V_{TP}| - V_{bias} &> \sqrt{\frac{2V_{bias}}{R_{mtj}}} \cdot \left(\frac{1}{\sqrt{K_N \cdot s_3}} + \frac{1}{\sqrt{K_P \cdot s_1}}\right) \\ \frac{1}{\sqrt{K_N \cdot s_3}} + \frac{1}{\sqrt{K_P \cdot s_1}} &< \frac{V_{DD} - |V_{TP}| - V_{bias}}{\sqrt{\frac{2V_{bias}}{R_{mtj}}}}. \end{aligned}$$

To simplify this, let $x = K_N \cdot s_3 = K_P \cdot s_1$:

$$\begin{aligned} \frac{2}{\sqrt{x}} &< (V_{DD} - |V_{TP}| - V_{bias}) \cdot \sqrt{\frac{R_{mtj}}{2V_{bias}}} \\ x &> \frac{8V_{bias}}{R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{bias})^2}. \end{aligned} \tag{3.21}$$

Finally, we can substitute the value of $x$ back in and calculate $s_3$ as

$$\begin{aligned} K_N \cdot s_3 &> \frac{8V_{bias}}{R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{bias})^2} \\ s_3 &> \frac{8V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{bias})^2} \end{aligned} \tag{3.22}$$

with

$$s_1 = \frac{K_N}{K_P} s_3. \tag{3.23}$$

**Apply saturation condition for transistor $M_2$**

Next, we are going to solve for the minimum shape factor needed to operate transistor $M_2$ in saturation-mode. To apply the saturation condition, we need to determine the voltages $V_{on,2}$, $V_{d,2}$, and $V_{s,2}$. $V_{on,2}$ can be determined by setting $I_{sd,2}$ to $I_{mtj}$ and substituting it into back into (3.17):

$$V_{on,2} = V_{sd,sat,2} = \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_P \cdot s_2}}. \tag{3.24}$$

To find $V_{d,2}$, we first solve the $I_{ds,4}$ current equation for $V_{gs,4}$ as

$$\begin{aligned} I_{ds,4} &= \frac{1}{2} K_N \cdot s_4 \cdot (V_{gs,4} - V_{TN})^2 \\ V_{gs,4} &= \sqrt{\frac{2I_{ds,4}}{K_N \cdot s_4}} + V_{TN} \end{aligned}$$

and then calculate $V_{d,2}$ as

$$\begin{aligned} V_{d,2} &= V_{g,4} \\ &= V_{gs,4} + V_{s,4} \\ V_{d,2} &= \sqrt{\frac{2I_{ds,4}}{K_N \cdot s_4}} + V_{TN} + V_{bias}. \end{aligned} \tag{3.25}$$

The source voltage, $V_{s,2}$, is $V_{DD}$. Now, we apply the saturation condition for transistor $M_2$:

$$\begin{aligned} V_{sd,2} &> V_{sd,sat,2} \\ V_{DD} - V_{d,2} &> \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_P \cdot s_2}}. \end{aligned} \tag{3.26}$$

Then, substituting (3.25) into (3.26) with $I_{ds,4} = {}^{V_{bias}}/_{R_{mtj}}$ gives

$$\begin{aligned} V_{DD} - \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_N \cdot s_4}} - V_{TN} - V_{bias} &> \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_P \cdot s_2}} \\ \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{1}{\sqrt{K_N \cdot s_4}} + \frac{1}{\sqrt{K_P \cdot s_2}} \right) &< V_{DD} - V_{TN} - V_{bias} \\ \frac{1}{\sqrt{K_N \cdot s_4}} + \frac{1}{\sqrt{K_P \cdot s_2}} &< \sqrt{\frac{R_{mtj}}{2V_{bias}}} \cdot (V_{DD} - V_{TN} - V_{bias}) . \end{aligned}$$

To simplify this, let $y = K_N \cdot s_4 = K_P \cdot s_2$:

$$\begin{aligned} \frac{2}{\sqrt{y}} &< (V_{DD} - V_{TN} - V_{bias}) \cdot \sqrt{\frac{R_{mtj}}{2V_{bias}}} \\ y &> \frac{8V_{bias}}{R_{mtj} \cdot (V_{DD} - V_{TN} - V_{bias})^2}. \end{aligned} \tag{3.27}$$

Finally, we can substitute the value of $y$ back in and calculate $s_4$ as

$$\begin{aligned} K_N \cdot s_4 &> \frac{8V_{bias}}{R_{mtj} \cdot (V_{DD} - V_{TN} - V_{bias})^2} \\ s_4 &> \frac{8V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - V_{TN} - V_{bias})^2} \end{aligned} \tag{3.28}$$

with

$$s_2 = \frac{K_N}{K_P} s_4. \tag{3.29}$$

**Summary of Design Equations**

A summary of all design equations for the basic current conveyor is as follows:

$$s_1 = s_2 = \frac{K_N}{K_P} s_3 = \frac{K_N}{K_P} s_4, \tag{3.30}$$

$$s_3 > \frac{8V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{bias})^2}, \tag{3.31}$$

$$s_4 > \frac{8V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - V_{TN} - V_{bias})^2}, \tag{3.32}$$

which can be combined as

$$s_{3,4} > \frac{8V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - \max(|V_{TP}|, V_{TN}) - V_{bias})^2}. \tag{3.33}$$

**Simulation Result**

To evaluate the performance of the basic current conveyor circuit, the circuit was designed and simulated in the 3M2P 0.5 $\mu$m process in the Cadence Design Environment using the

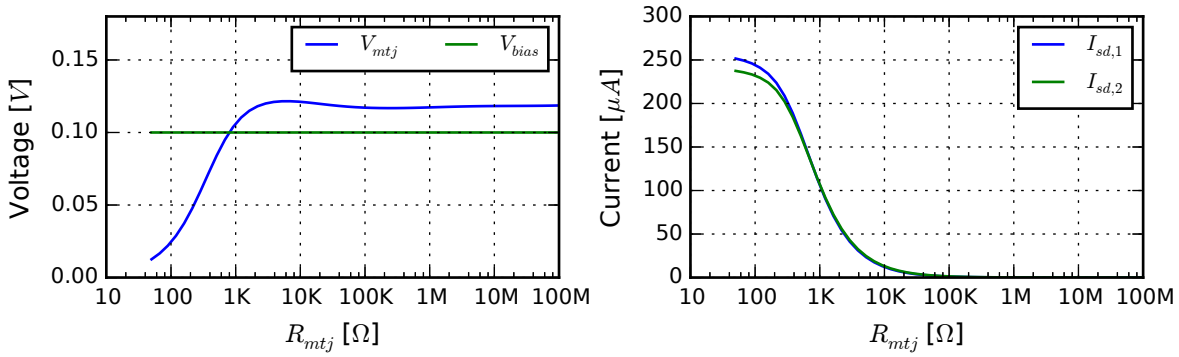



Figure 3.6: Simulation of the basic current conveyor. (Left) shows the $V_{mtj}$ and $V_{bias}$ voltages and (right) shows the $I_{sd,1}$ and $I_{sd,2}$ branch currents. For this circuit, $V_{bias} = 0.1\,\mathrm{V}$, $R_{mtj} = 1\,\mathrm{k\Omega}$, $L = 2\,\mu\mathrm{m}$, $s_N = 1.5$, and $s_P = 4.65$.

Spectre simulator. The design equations (developed above) were used to size the transistors, targeting $V_{bias} = 0.1\,\mathrm{V}$ and $R_{mtj} = 500\,\Omega$, and using transistor lengths of $2\,\mu\mathrm{m}$ for all transistors. The shape factors chosen were $s_N = 1.5$ (for the NMOS transistors) and $s_P = 4.65$ (for the PMOS transistors). The circuit was then simulated using $V_{bias} = 0.1\,\mathrm{V}$ and $R_{mtj} = 1\,\mathrm{k\Omega}$, producing the simulation result shown in Figure 3.6. Here, the $V_{mtj}$ and $V_{bias}$ voltages are compared in the left plot and the $I_{sd,1}$ and $I_{sd,2}$ currents are compared in the right plot to see how well they match. The resistance was swept from $50\,\Omega$ to $100\,\mathrm{M\Omega}$. This shows that at low $R_{mtj}$ resistance, the $V_{bias}$ voltage cannot be maintained across $R_{mtj}$ until about $500\,\Omega$. And, at high resistance, the $V_{mtj}$ node follows $V_{bias}$, but has an offset of about $18\,\mathrm{mV}$.

### 3.2.2 P-cascode

The P-cascode current conveyor, shown in Figure 3.7, is formed by transistors $M_{1-6}$. Transistors $M_{1-4}$ form a cascoded PMOS current mirror circuit and transistors $M_{5-6}$ form a basic current mirror circuit. The P-cascode current conveyor operates similarly to the basic current conveyor circuit. Voltage $V_{mtj}$ is clamped to $V_{bias}$ potential and produces current

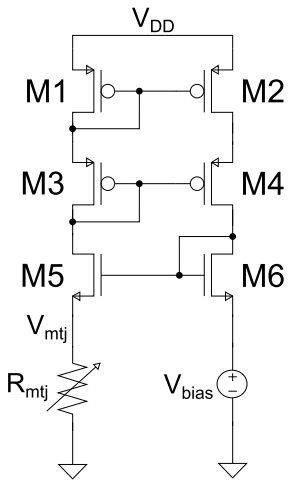



Figure 3.7: P-cascode current conveyor.

$I_{mtj}$ down through the left branch by equation $I_{mtj} = {}^{V_{bias}}/_{R_{mtj}}$. The currents are mirrored between branches with a 1:1 gain with the widths and lengths of the transistors matched, however, with the addition of the P-cascode structure, thus making $s_1 = s_2$, $s_3 = s_4$, $s_5 = s_6$, and $I_{mtj} = I_{sd,1} = I_{sd,2} = I_{sd,3} = I_{sd,4} = I_{ds,5} = I_{ds,6}$.

To solve for the conditions that will keep all transistors in saturation, we first observe that transistors $M_{1,3,6}$ are diode-connected and, according to (3.14, 3.15), are always in saturation.

We can show that $V_{mtj} = V_{bias}$ by setting $I_{ds,5} = I_{ds,6}$ and assuming that $M_{2,4,5}$ are in saturation:

$$\frac{1}{2}K_N \cdot s_5 \cdot (V_{gs,5} - V_{TN})^2 = \frac{1}{2}K_N \cdot s_6 \cdot (V_{gs,6} - V_{TN})^2$$
$$V_{gs,5} = V_{gs,6}. \tag{3.34}$$

Then,

$$V_{mtj} + V_{gs,5} = V_{bias} + V_{gs,6}$$
$$V_{mtj} = V_{bias}. \tag{3.35}$$

**Apply saturation condition for transistor** $M_5$

To apply the saturation condition, we need to determine voltages $V_{on,5}$, $V_{d,5}$, and $V_{s,5}$. $V_{on,5}$ can be determined by setting $I_{ds,5}$ to $I_{mtj}$ and substituting it back into (3.16):

$$V_{on,5} = V_{ds,sat,5} = \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot K_N \cdot s_5}}. \tag{3.36}$$

Next, we find the source to drain voltage of transistors $M_{1,3}$:

$$V_{sd,1} = V_{on,1} + |V_{TP}|,$$
$$V_{sd,3} = V_{on,3} + |V_{TP}|.$$

Then, we find the drain voltage of transistor $M_5$:

$$\begin{aligned} V_{d,5} &= V_{DD} - V_{sd,1} - V_{sd,3} \\ &= V_{DD} - V_{on,1} - V_{on,3} - 2|V_{TP}|. \end{aligned} \tag{3.37}$$

The source voltage, $V_{s,5}$ is $V_{bias}$. Now, we can apply the condition for saturation as

$$\begin{aligned} V_{ds,5} &> V_{ds,sat,5} \\ V_{d,5} - V_{bias} &> V_{on,5} \\ V_{DD} - V_{on,1} - V_{on,3} - 2\left|V_{TP}\right| - V_{bias} &> V_{on,5} \\ V_{DD} - 2\left|V_{TP}\right| - V_{bias} &> V_{on,1} + V_{on,3} + V_{on,5}. \end{aligned} \tag{3.38}$$

Note, this expression can be generalized as the following:

$$V_{DD} - \sum_{i=1}^{p} \left|V_{TP,i}\right| - V_{bias} > \sum_{i=1}^{p} V_{on,i} + V_{on,nmos} \tag{3.39}$$

where $p = \#$ of PMOSs on the left branch and $V_{on,nmos}$ is the "on" voltage for the NMOS transistor (which replaces $V_{on,5}$). Next, we can substitute in the value for each $V_{on,i}$ voltage and factor:

$$V_{DD} - 2\left|V_{TP}\right| - V_{bias} > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \cdot \left(\frac{1}{\sqrt{K_P \cdot s_1}} + \frac{1}{\sqrt{K_P \cdot s_3}} + \frac{1}{\sqrt{K_N \cdot s_5}}\right).$$

Let $x_5 = K_P \cdot s_1 = K_P \cdot s_3 = K_N \cdot s_5$, then

$$\begin{aligned} V_{DD} - 2\left|V_{TP}\right| - V_{bias} &> \sqrt{\frac{2V_{bias}}{R_{mtj}}} \cdot \left(\frac{3}{\sqrt{x_5}}\right) \\ x_5 &> \frac{18V_{bias}}{R_{mtj} \cdot \left(V_{DD} - 2\left|V_{TP}\right| - V_{bias}\right)^2}. \end{aligned} \tag{3.40}$$

Substituting $x_5 = K_N \cdot s_5$ back in, we get

$$s_5 > \frac{18V_{bias}}{K_N \cdot R_{mtj} \cdot \left(V_{DD} - 2\left|V_{TP}\right| - V_{bias}\right)^2} \tag{3.41}$$

with

$$s_1 = s_3 = \frac{K_N}{K_P} s_5. \tag{3.42}$$

**Apply saturation condition for transistor $M_2$**

To apply the saturation condition for transistor $M_2$, we must determine the voltage $V_{d,2}$ and $V_{s,2}$. Since $s_1 = s_2$ and $I_{sd,1} = I_{sd,2}$, then $V_{on,1} = V_{on,2}$ and $V_{sg,1} = V_{sg,2}$. Also, since $s_3 = s_4$ and $I_{sd,3} = I_{sd,4}$, then $V_{on,3} = V_{on,4}$ and $V_{sg,3} = V_{sg,4}$. The drain voltage of transistor $M_2$ can be determined as

$$\begin{aligned} V_{d,2} &= V_{DD} - V_{sg,1} - V_{sg,3} + V_{sg,4} \\ &= V_{DD} - V_{sg,1} \\ &= V_{DD} - V_{on,1} - |V_{TP}| \,. \end{aligned} \tag{3.43}$$

The source voltage, $V_{s,2}$, is $V_{DD}$. Now, we can apply the condition for saturation

$$\begin{aligned} V_{sd,2} &> V_{on,2} \\ V_{DD} - (V_{DD} - V_{on,1} - |V_{TP}|) &> V_{on,2} \\ |V_{TP}| &> 0, \end{aligned} \tag{3.44}$$

showing that transistor $M_2$ is always in saturation.

**Apply saturation condition of transistor $M_4$**

For this transistor, we need to know $V_{d,4}$ and $V_{s,4}$ to apply the saturation condition. We can determine $V_{d,4}$ as follows:

$$\begin{aligned} V_{d,4} &= V_{bias} + V_{gs,6} \\ &= V_{bias} + V_{on,6} + V_{TN}. \end{aligned} \tag{3.45}$$

The source voltage, $V_{s,4}$, is equal to $V_{d,2}$ which is given by (3.43). Now, we can apply the saturation condition as

$$\begin{aligned} V_{sd,4} &> V_{sd,sat,4} \\ V_{s,4} - V_{d,4} &> V_{on,4} \\ (V_{DD} - V_{on,1} - |V_{TP}|) - (V_{bias} + V_{on,6} + V_{TN}) &> V_{on,4} \\ V_{DD} - |V_{TP}| - V_{TN} - V_{bias} &> V_{on,2} + V_{on,4} + V_{on,6}. \end{aligned}$$

Substituting in for each $V_{on,i}$ voltage, we get

$$\begin{aligned} V_{DD} - |V_{TP}| - V_{TN} - V_{bias} &> \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{1}{\sqrt{K_P \cdot s_2}} + \frac{1}{\sqrt{K_P \cdot s_4}} + \frac{1}{\sqrt{K_N \cdot s_6}} \right) \\ V_{DD} - |V_{TP}| - V_{TN} - V_{bias} &> \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{3}{\sqrt{x_4}} \right) \end{aligned}$$

where $x_4 = K_P \cdot s_2 = K_P \cdot s_4 = K_N \cdot s_6$. Then

$$x_4 > \frac{18V_{bias}}{R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{TN} - V_{bias})^2}. \tag{3.46}$$

Substituting in $x_4 = K_N \cdot s_6$, we get

$$s_6 > \frac{18V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{TN} - V_{bias})^2} \tag{3.47}$$

with

$$s_2 = s_4 = \frac{K_N}{K_P} s_6.$$

**Summary of Design Equations**

A summary of all design equations for the P-cascode current conveyor is as follows:

$$s_1 = s_2 = s_3 = s_4 = \frac{K_N}{K_P} s_5 = \frac{K_N}{K_P} s_6, \tag{3.48}$$

$$s_5 > \frac{18V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - 2\,|V_{TP}| - V_{bias})^2}, \tag{3.49}$$

$$s_6 > \frac{18V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - |V_{TP}| - V_{TN} - V_{bias})^2}, \tag{3.50}$$

which can be combined as

$$s_{5,6} > \frac{18V_{bias}}{K_N \cdot R_{mtj} \cdot (V_{DD} - |V_{TP}| - \max(|V_{TP}|, V_{TN}) - V_{bias})^2}. \tag{3.51}$$

**Simulation Result**

To evaluate the performance of the P-cascode current conveyor circuit, the circuit was designed and simulated in the 3M2P 0.5 $\mu$m process in the Cadence Design Environment using the Spectre simulator. The above design equations were used to size the transistors, targeting

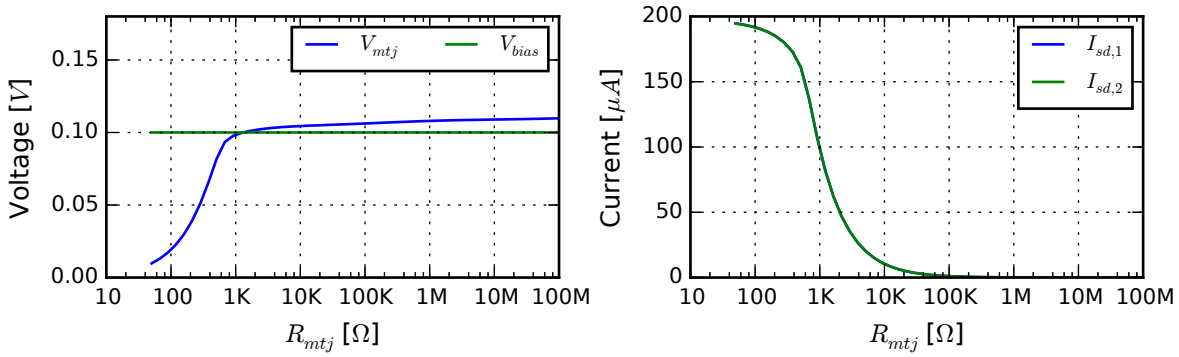



Figure 3.8: Simulation of the P-cascode current conveyor. (Left) shows the $V_{mtj}$ and $V_{bias}$ voltages and (right) shows the $I_{sd,1}$ and $I_{sd,2}$ branch currents. For this circuit, $V_{bias} = 0.1\,\mathrm{V}$, $R_{mtj} = 1\,\mathrm{k\Omega}$, $L = 2\,\mu\mathrm{m}$, $s_N = 4.0$, and $s_P = 12.4$.

$V_{bias} = 0.1\,\mathrm{V}$ and $R_{mtj} = 500\,\Omega$, and using transistor lengths of $2\,\mu\mathrm{m}$ for all transistors. The shape factors chosen were $s_N = 4.0$ (for the NMOS transistors) and $s_P = 12.4$ (for the PMOS transistors). The circuit was then simulated using $V_{bias} = 0.1\,\mathrm{V}$ and $R_{mtj} = 1\,\mathrm{k\Omega}$, producing the simulation result shown in Figure 3.8. The $V_{mtj}$ and $V_{bias}$ voltages are compared in the left plot and the $I_{sd,1}$ and $I_{sd,2}$ currents are compared in the right plot to see how well they match. The resistance was swept from $50\,\Omega$ to $100\,\mathrm{M\Omega}$. At low $R_{mtj}$ resistance, the $V_{bias}$ voltage cannot be maintained across $R_{mtj}$ until about $500\,\Omega$. And, at high resistance, the $V_{mtj}$ node follows $V_{bias}$, this time with a smaller offset than the basic current conveyor of up to $10\,\mathrm{mV}$.

### 3.2.3 NP-cascode

The NP-cascode current conveyor, shown in Figure 3.9, is formed by transistors $M_{1-8}$. Transistors $M_{1-4}$ form a cascoded PMOS current mirror circuit and transistors $M_{5-8}$ for a cascoded NMOS current mirror circuit. The NP-cascode current conveyor also operates similarly to the basic and P-cascode current conveyor circuits. Voltage $V_{mtj}$ is clamped to $V_{bias}$ potential and produces current $I_{mtj}$ down through the left branch with $I_{mtj} = {}^{V_{bias}}/_{R_{mtj}}$.

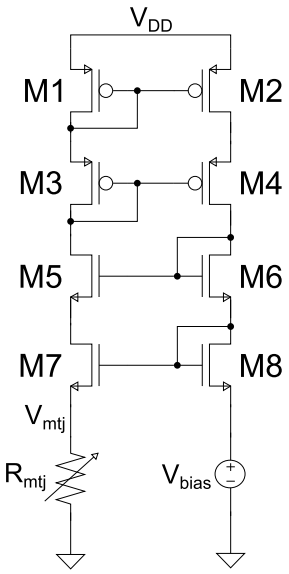



Figure 3.9: NP-cascode current conveyor.

The currents are mirrored between branches with a 1:1 gain with the widths and lengths of the transistors matched. With the addition of the N-cascode structure, $s_1 = s_2$, $s_3 = s_4$, $s_5 = s_6$, $s_7 = s_8$, and $I_{mtj} = I_{sd,1} = I_{sd,2} = I_{sd,3} = I_{sd,4} = I_{ds,5} = I_{ds,6} = I_{ds,7} = I_{ds,8}$.

To solve for the conditions that will keep all transistors in saturation, we first observe that transistors $M_{1,3,6,8}$ are diode-connected, and according to (3.14, 3.15), are always in saturation. Without going through a full-proof (as in the previous section), if $I_{sd,1} = I_{sd,2}$ and $s_1 = s_2$, then $M_2$ is in saturation. Also, if $I_{ds,7} = I_{ds,8}$ and $s_7 = s_8$, then $M_7$ is in saturation.

Setting $I_{ds,7} = I_{ds,8}$, then $V_{mtj} = V_{bias}$, without going through a full proof again.

**Apply saturation condition for transistor $M_5$**

First, find the drain voltage of transistor $M_5$:

$$\begin{aligned} V_{d,5} &= V_{DD} - V_{sg,1} - V_{sg,3} \\ &= V_{DD} - (V_{on,1} + |V_{TP}|) - (V_{on,3} + |V_{TP}|)\,. \end{aligned} \tag{3.52}$$

Next, find the source voltage of transistor $M_5$. We can note that $V_{gs,5} = V_{gs,6}$ because transistors $M_{5,6}$ are identical with the same current. This gives

$$\begin{aligned} V_{s,5} &= V_{bias} + V_{gs,8} + V_{gs,6} - V_{gs,5} \\ &= V_{bias} + V_{gs,8} \\ &= V_{bias} + (V_{on,8} + V_{TN})\,. \end{aligned} \tag{3.53}$$

Then apply the condition for saturation:

$$V_{ds,5} > V_{ds,sat,5}$$

$$V_{d,5} - V_{s,5} > V_{on,5}$$

$$V_{DD} - 2\,|V_{TP}| - V_{TN} - V_{bias} > V_{on,1} + V_{on,3} + V_{on,5} + V_{on,7}.$$

Note, $V_{on,8}$ was substituted with $V_{on,7}$ which it is equal to. Next, let $x = K_P \cdot s_1 = K_P \cdot s_3 = K_N \cdot s_5 = K_N \cdot s_7$. This gives

$$V_{DD} - 2\,|V_{TP}| - V_{TN} - V_{bias} > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{1}{\sqrt{K_P \cdot s_1}} + \frac{1}{\sqrt{K_P \cdot s_3}} + \frac{1}{\sqrt{K_N \cdot s_5}} + \frac{1}{\sqrt{K_N \cdot s_7}} \right)$$

$$V_{DD} - 2\,|V_{TP}| - V_{TN} - V_{bias} > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{4}{\sqrt{x}} \right).$$

Solving for $x$, we get

$$\begin{aligned}\sqrt{x} &> \sqrt{\frac{2V_{bias}}{R_{mtj}}}\left(\frac{4}{V_{DD}-2\left|V_{TP}\right|-V_{TN}-V_{bias}}\right)\\ x &> \frac{32V_{bias}}{R_{mtj}\cdot\left(V_{DD}-2\left|V_{TP}\right|-V_{TN}-V_{bias}\right)^2}.\end{aligned} \tag{3.54}$$

Substituting $x = K_N \cdot s_5$ back in, gives

$$s_5 > \frac{32V_{bias}}{K_N \cdot R_{mtj}\cdot\left(V_{DD}-2\left|V_{TP}\right|-V_{TN}-V_{bias}\right)^2} \tag{3.55}$$

with

$$s_1 = s_3 = \frac{K_N}{K_P}s_5 = \frac{K_N}{K_P}s_7. \tag{3.56}$$

**Apply saturation condition of transistor $M_4$**

First, find the drain voltage of transistor $M_4$:

$$\begin{aligned}V_{d,4} &= V_{bias} + V_{gs,8} + V_{gs,6}\\ &= V_{bias} + \left(V_{on,8} + V_{TN}\right) + \left(V_{on,6} + V_{TN}\right).\end{aligned} \tag{3.57}$$

Next, find the source voltage of transistor $M_4$. We can note that $V_{sg,3} = V_{sg,4}$ because transistors $M_{3,4}$ are identical with the same current. This gives

$$\begin{aligned}V_{s,4} &= V_{DD} - V_{sg,1} - V_{sg,3} + V_{sg,4}\\ &= V_{DD} - V_{sg,1}\\ &= V_{DD} - \left(V_{on,1} + \left|V_{TP}\right|\right).\end{aligned} \tag{3.58}$$

Then apply the condition for saturation:

$$V_{sd,4} > V_{sd,sat,4}$$

$$V_{s,4} - V_{d,4} > V_{on,4}$$

$$V_{DD} - |V_{TP}| - 2V_{TN} - V_{bias} > V_{on,2} + V_{on,4} + V_{on,6} + V_{on,8}.$$

Note, $V_{on,1}$ was substituted with $V_{on,2}$ which it is equal to. Next, let $x = K_P \cdot s_2 = K_P \cdot s_4 = K_N \cdot s_6 = K_N \cdot s_8$. This gives

$$V_{DD} - |V_{TP}| - 2V_{TN} - V_{bias} > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{1}{\sqrt{K_P \cdot s_2}} + \frac{1}{\sqrt{K_P \cdot s_4}} + \frac{1}{\sqrt{K_N \cdot s_6}} + \frac{1}{\sqrt{K_N \cdot s_8}} \right)$$

$$V_{DD} - |V_{TP}| - 2V_{TN} - V_{bias} > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{4}{\sqrt{x}} \right).$$

Solving for $x$, we get

$$\sqrt{x} > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \left( \frac{4}{V_{DD} - |V_{TP}| - 2V_{TN} - V_{bias}} \right)$$

$$x > \frac{32V_{bias}}{R_{mtj} \cdot (V_{DD} - |V_{TP}| - 2V_{TN} - V_{bias})^2}. \tag{3.59}$$

Substituting $x = K_P \cdot s_4$ back in, gives

$$s_4 > \frac{32V_{bias}}{K_P \cdot R_{mtj} \cdot (V_{DD} - |V_{TP}| - 2V_{TN} - V_{bias})^2} \tag{3.60}$$

with

$$s_2 = s_4 = \frac{K_N}{K_P} s_6 = \frac{K_N}{K_P} s_8. \tag{3.61}$$

### Summary of Design Equations

A summary of all design equations for the NP-cascode current conveyor is as follows:

$$s_1 = s_2 = s_3 = s_4 = \frac{K_N}{K_P} s_5 = \frac{K_N}{K_P} s_6 = \frac{K_N}{K_P} s_7 = \frac{K_N}{K_P} s_8, \tag{3.62}$$

$$s_5 > \frac{32 V_{bias}}{K_N \cdot R_{mtj} \cdot \left(V_{DD} - 2\left|V_{TP}\right| - V_{TN} - V_{bias}\right)^2}, \tag{3.63}$$

$$s_6 > \frac{32 V_{bias}}{K_N \cdot R_{mtj} \cdot \left(V_{DD} - \left|V_{TP}\right| - 2V_{TN} - V_{bias}\right)^2}, \tag{3.64}$$

which can be combined as

$$s_{5,6} > \frac{32 V_{bias}}{K_N \cdot R_{mtj} \cdot \left(V_{DD} - \left|V_{TP}\right| - V_{TN} - \max\left(\left|V_{TP}\right|, V_{TN}\right) - V_{bias}\right)^2}. \tag{3.65}$$

### Simulation Result

To evaluate the performance of the NP-cascode current conveyor circuit, the circuit was designed and simulated in the 3M2P $0.5\,\mu$m process in the Cadence Design Environment using the Spectre simulator. The above design equations were used to size the transistors, targeting $V_{bias} = 0.1\,\text{V}$ and $R_{mtj} = 500\,\Omega$, and using transistor lengths $2\,\mu$m for all transistors. The shape factors chosen were $s_N = 12.0$ (for the NMOS transistors) and $s_P = 37.2$ (for the PMOS transistors). The circuit was then simulated using $V_{bias} = 0.1\,\text{V}$ and $R_{mtj} = 1\,\text{k}\Omega$, producing the simulation result shown in Figure 3.10. The $V_{mtj}$ and $V_{bias}$ voltages are compared in the left plot and the $I_{sd,1}$ and $I_{sd,2}$ currents are compared in the right plot to see how well they match. The resistance was swept from $50\,\Omega$ to $100\,\text{M}\Omega$. At low $R_{mtj}$

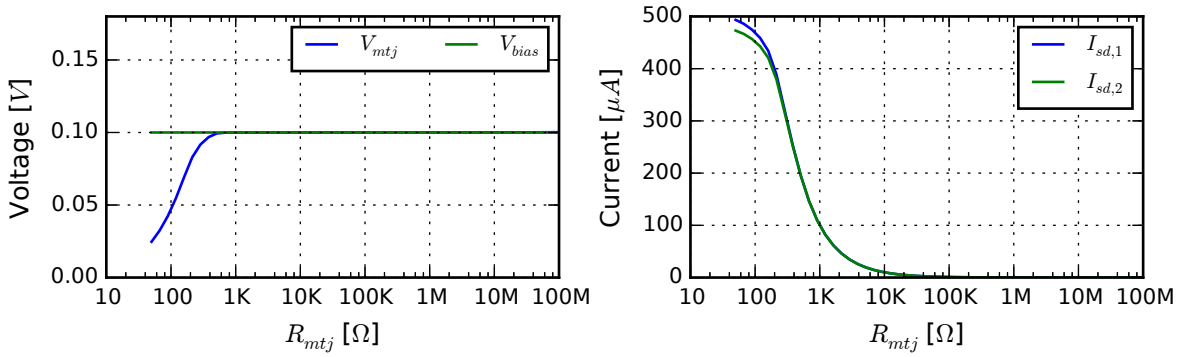



Figure 3.10: Simulation of the NP-cascode current conveyor. (Left) shows the $V_{mtj}$ and $V_{bias}$ voltages and (right) shows the $I_{sd,1}$ and $I_{sd,2}$ branch currents. For this circuit, $V_{bias} = 0.1\,\mathrm{V}$, $R_{mtj} = 1\,\mathrm{k}\Omega$, $L = 2\,\mu\mathrm{m}$, $s_N = 12.0$, and $s_P = 37.2$.

resistance, the $V_{bias}$ voltage cannot be maintained across $R_{mtj}$ until about $250\,\Omega$ or so. And, at high resistance, the $V_{mtj}$ node follows $V_{bias}$ very closely, with virtually no offset.

### 3.2.4 Generalized

We can generalize these results to produce a set of design equations for sizing the transistors for any number of N or P cascode stages. First, the saturation condition of transistor $M_i$ on a branch containing transistor set $T$ is

$$V_{DD} - \sum_{\substack{j \in T \\ j \neq i}} V_{T,j} - V_{bias} > \sum_{j \in T} V_{on,j}, \tag{3.66}$$

or

$$V_1 > \sum_{j \in T} V_{on,j}$$

where $V_1 = V_{DD} - \sum_{\substack{j \in T \\ j \neq i}} V_{T,j} - V_{bias}$. Expanding the $V_{on,j}$ terms, we get

$$V_1 > \sqrt{\frac{2V_{bias}}{R_{mtj}}} \cdot \sum_{j \in T} \left( \frac{1}{\sqrt{K_j \cdot s_j}} \right). \tag{3.67}$$

From this, we can get a generalized result:

$$\sum_{j \in T} \left( \frac{1}{\sqrt{K_j \cdot s_j}} \right) < \sqrt{\frac{R_{mtj} \cdot (V_1)^2}{2V_{bias}}}, \tag{3.68}$$

or

$$\sqrt{K_1 \cdot s_1} \,||\, \ldots \,||\, \sqrt{K_{|T|} \cdot s_{|T|}} > \sqrt{\frac{2V_{bias}}{R_{mtj} \cdot (V_1)^2}}. \tag{3.69}$$

Continuing from (3.67), let $x = K_j \cdot s_j$ for each $j = 1, \ldots, |T|$. Then

$$\begin{aligned} V_1 &> \sqrt{\frac{2V_{bias}}{R_{mtj}}} \cdot \sum_{j \in T} \left( \frac{1}{\sqrt{x}} \right) \\ V_1 &> \sqrt{\frac{2V_{bias}}{R_{mtj}}} \cdot \frac{|T|}{\sqrt{x}} \\ x &> \frac{2\,|T|^2\, V_{bias}}{R_{mtj} \cdot (V_1)^2}. \end{aligned} \tag{3.70}$$

Substituting $x = K_j \cdot s_j$ back in, we get

$$s_j > \frac{2\,|T|^2\, V_{bias}}{K_j \cdot R_{mtj} \cdot (V_1)^2}. \tag{3.71}$$

Assuming that $V_{TN}$ and $V_{TP}$ are constant for all NMOS and PMOS transistors, respectively, we can make a simplification of the design equations. Along one branch,

$$n = \#\text{ of NMOSs}$$

$$p = \#\text{ of PMOSs}$$

where $n + p = |T|$. Then,

$$s_N > \frac{2\left(n+p\right)^2 V_{bias}}{K_N \cdot R_{mtj} \cdot \left(V_1\right)^2} \tag{3.72}$$

where

$$V_1 = V_{DD} - (p-1)\left|V_{TP}\right| - (n-1)\,V_{TN} - \max\left(\left|V_{TP}\right|, V_{TN}\right) - V_{bias}, \tag{3.73}$$

$$s_P = \frac{K_N}{K_P} s_N. \tag{3.74}$$

To use these equations, simply set $n$ and $p$ appropriately for the chosen cascode structure. For the basic current conveyor, $n = 1$ and $p = 1$. For the P-cascode current conveyor, $n = 1$ and $p = 2$. And, for the NP-cascode current conveyor, $n = 2$ and $p = 2$. Other combinations of cascode structures can be explored by further changing $n$ and $p$. Once these two parameters are chosen, then the $s_N$ and $s_P$ equations will provide the necessary shape factors of the transistors so that they will operate with the given $V_{bias}$ voltage and $R_{mtj}$ resistance.

## 3.3 Read circuit design

We now present the design of a resistance-to-voltage (R2V) read circuit based on the current-mode read circuit design from Section 3.1. It works by performing a continuous read to sense an input resistance and produce a rail-to-rail logic voltage output. Then, we present area, transient response, power, and jitter characterizations with simulation results of the read circuit in the 3M2P 0.5 $\mu$m process and compare these results to a second implementation in a 5 metal 1 poly (5M1P) 0.18 $\mu$m process.

The design of the resistance-to-voltage (R2V) read circuit is shown in Figure 3.11 in the 3M2P 0.5 $\mu$m process. It consists of three parts: current conveyor, current comparator, and rail-to-rail output buffer. The current conveyor pins a voltage across the input resistance and produces an output current inversely proportional to the input resistance. That current is compared to a threshold current, $I_{th}$, by the current comparator, and then amplified rail-to-rail by the output buffer.

**Current conveyor:** The current conveyor circuit, formed by transistors $M_{1-6}$, is the P-cascode type from Section 3.2.2. It consists of two back-to-back current mirrors; one formed by a PMOS-cascode structure and the second one by an NMOS one. It is designed to have equal currents flowing through both branches.

Recapping the P-cascode current conveyor derivation in Section 3.2.2, this circuit operates by clamping the voltage $V_{mtj}$ to $V_{bias}$ over the resistance $R_{mtj}$ at the input. The current $I_{mtj}$ that flows through $M_{1-3}$ and $R_{mtj}$ is $I_{mtj} = \frac{V_{bias}}{R_{mtj}}$. $I_{mtj}$ is then mirrored to the current comparator through the current mirror formed by $M_{1,2}$ and $M_{7,8}$. All transistors operate in the saturation region satisfying the conditions $V_{ds} > V_{ds,sat}$ for NMOS and $V_{sd} > V_{sd,sat}$ for PMOS. Transistors $M_{1,2,6}$ are diode-connected and therefore always operate in saturation.

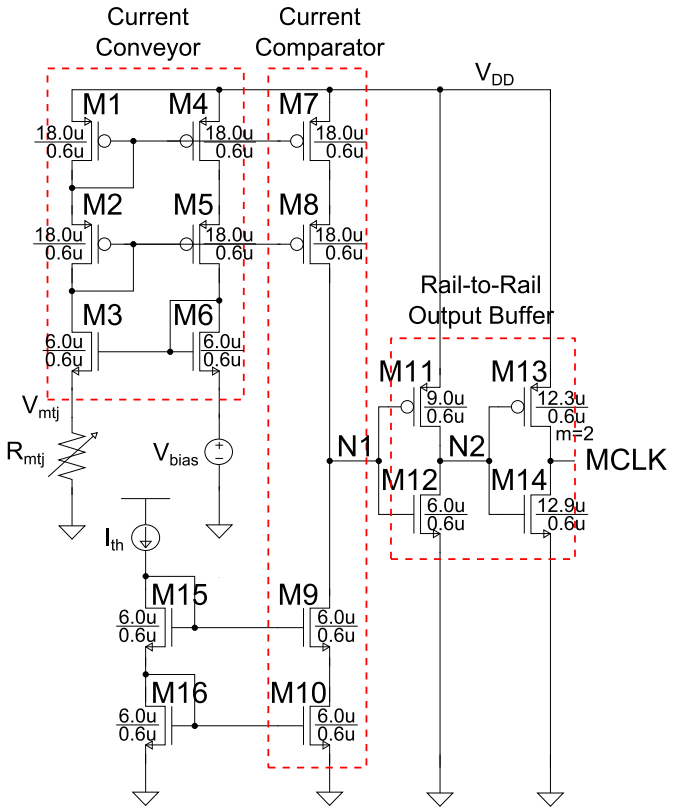



Figure 3.11: Resistance-to-voltage (R2V) read circuit in the 3M2P 0.5 $\mu$m process.

Transistors $M_{3,4,5}$ are determined by design equations and are biased such that they operate at the edge of the saturation region in order to get the smallest aspect ratio. Body and lambda effects are not considered:

$$\frac{W}{L}_3 > \frac{\frac{18V_{bias}}{K_N R_{mtj}}}{\left(V_{DD} - |V_{TP}| - \max\left(|V_{TP}|, V_{TN}\right) - V_{bias}\right)^2}, \tag{3.75}$$

$$\frac{W}{L}_{4,5} > \frac{\frac{18V_{bias}}{K_P R_{mtj}}}{\left(V_{DD} - |V_{TP}| - \max\left(|V_{TP}|, V_{TN}\right) - V_{bias}\right)^2}. \tag{3.76}$$

Here, all aspect ratios for NMOS transistors are equal ($\frac{W}{L}_3 = \frac{W}{L}_6$) and all aspect ratios for PMOS transistors are equal ($\frac{W}{L}_1 = \frac{W}{L}_2 = \frac{W}{L}_4 = \frac{W}{L}_5$). $V_{DD}$ is the supply voltage, $V_{TN}$ is the NMOS threshold voltage, $V_{TP}$ is the PMOS threshold voltage, $K_N$ is the NMOS

transconductance parameter, $K_P$ is the PMOS transconductance parameter, $V_{bias}$ is the input bias voltage, and $R_{mtj}$ is the input MTJ resistance.

It is good to note that since the read circuit is designed to distinguish between two discrete resistance inputs, it has significant tolerance for error in the read current. Small transistors may be able to be used while still allowing the read circuit to correctly distinguish between two input states.

**Current comparator:** The current comparator, formed by transistors $M_{7-10}$, compares an output current to a threshold current, $I_{th}$, and converts it to a logic voltage output [51]. $M_{1,2}$ current is copied and sourced by transistors $M_{7,8}$. Likewise, $M_{15,16}$ current is copied and sunk by transistors $M_{9,10}$. The output voltage $V_{N1}$ at node *N1* will swing depending on which current is greater. The voltage swing is determined analytically (using 1st order approximations) as

$$2\sqrt{\frac{2I_{th}}{\frac{W}{L}_{9,10} \cdot K_N}} + V_{TN} < V_{N1} < V_{DD} - 2\sqrt{\frac{2I_{mtj}}{\frac{W}{L}_{7,8} \cdot K_P}} - |V_{TP}| \tag{3.77}$$

where $I_{mtj}$ is the current through $R_{mtj}$, $I_{th}$ is the threshold current, and $\frac{W}{L}_i$ is the aspect ratio of the $i^{\text{th}}$ transistor.

**Rail-to-rail output buffer:** The last stage of the read circuit is the rail-to-rail output buffer formed by transistors $M_{11-14}$. This buffer performs voltage amplification of the current comparator output voltage $V_{N1}$ to get a rail-to-rail logic voltage output *MCLK* used to drive downstream logic. The buffer is designed to drive a 600 fF capacitive load with 1 ns rise/fall times (10–90%).

Table 3.7: Comparison of relevant process parameters for the R2V read circuit.

| | **Process** | |
|---|---|---|
| **Parameter** | **3M2P 0.5** $\mu$m | **5M1P 0.18** $\mu$m |
| $V_{DD}$ | 5.0 V | 1.8 V |
| $K_N/K_P$ | $110/32 \frac{\mu A}{V^2}$ | $342/74 \frac{\mu A}{V^2}$ |
| $V_{TN}/V_{TP}$ | 0.77/–0.95 V | 0.5/–0.49 V |

We laid out and simulated the R2V read circuit in the 3M2P 0.5 $\mu$m process and the 5M1P 0.18 $\mu$m process to compare how circuit characteristics change at smaller process dimensions. Table 3.7 compares the relevant process parameters. The circuit is characterized in terms of area, transient response, power, and jitter in the Cadence Design Environment using Spectre simulator.

### 3.3.1 Area

The R2V read circuit is designed using the smallest transistors that will operate in the saturation region and still perform well. The layout of the circuit is shown in Figure 3.12. The dimensions of the circuit are 31.35 $\mu$m by 40.20 $\mu$m.

For the 5M1P 0.18 $\mu$m process implementation, the read circuit is redesigned with transistor aspect ratios set according to (3.75) and (3.76) to ensure the current conveyor transistors operate in the saturation region. The current conveyor and comparator transistors change as follows: $\frac{W}{L}_p = \frac{18\,\mu m}{0.6\,\mu m}$ to $\frac{18\,\mu m}{0.18\,\mu m}$ and $\frac{W}{L}_n = \frac{6\,\mu m}{0.6\,\mu m}$ to $\frac{4.005\,\mu m}{0.18\,\mu m}$. The transistors stay at about the same width but scale down in length. This is primarily due to both process circuits supplying the same $I_{mtj}$ current for a constant $R_{mtj}$. The rail-to-rail output buffer transistors are scaled $0.5\,\mu m/0.18\,\mu m = 2.77$ times smaller which is the scaling factor between the two processes. The new layout has dimensions 8.28 $\mu$m by 27.27 $\mu$m. The area of the current conveyor and

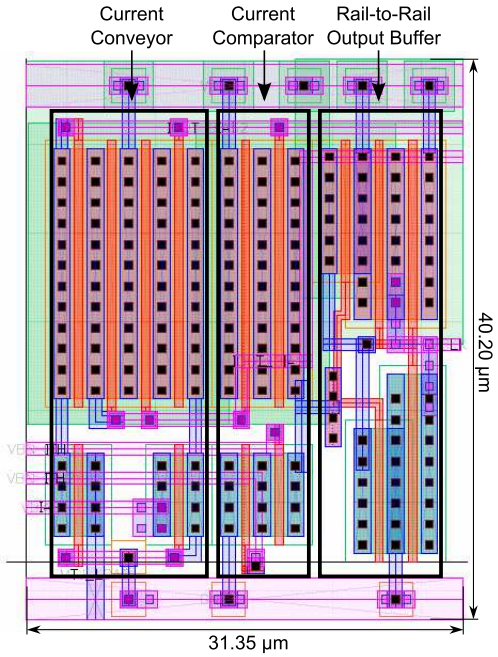



Figure 3.12: Layout of the R2V read circuit in the 3M2P 0.5 $\mu$m process. The sub-circuits highlighted in black boxes are (left) current conveyor, (middle) current comparator, and (right) rail-to-rail output buffer.

current mirror sub-circuits scale roughly linearly. The rail-to-rail output buffer may scale quadratically.

### 3.3.2 Transient response

The transient response of the circuit limits the maximum frequency at which the read circuit can be clocked by an external magnetic field and the performance of downstream circuits clocked by this read circuit. For the R2V read circuit implemented in the 3M2P 0.5 $\mu$m process, the transient response is shown in Figure 3.13. The input resistance $R_{mtj}$ is clocked at 10 MHz with values 500 $\Omega$ and 1 k$\Omega$. An appropriately chosen area and resistance-area (RA) product of an MTJ device will yield these values. The bias voltage $V_{bias}$ is set to 0.1 V which is pinned at $V_{mtj}$. The variation in $V_{mtj}$ is due to non-linearities in the current conveyor

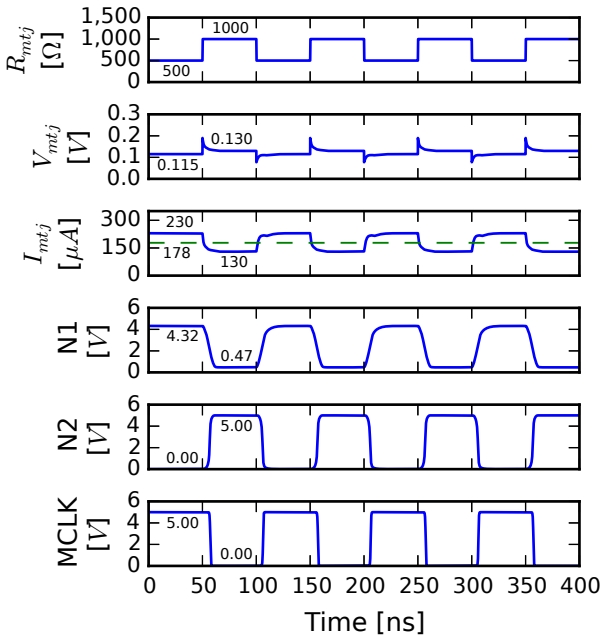



Figure 3.13: Transient response of the R2V read circuit at 10 MHz frequency. $R_{mtj}$ is the input resistance, $V_{mtj}$ is the voltage over $R_{mtj}$, $I_{mtj}$ is the current through $R_{mtj}$, *N1* is the current comparator output, *N2* is the first inverter output, and *MCLK* is the read circuit output. Threshold current, $I_{th}$, is indicated by the dashed line on $I_{mtj}$.

circuit. $I_{mtj}$ is simulated to be 230 and 130 $\mu$A for each $R_{mtj}$, respectively. These states are distinguished by comparing $I_{mtj}$ to a threshold current $I_{th}$ indicated by the dashed line in the figure. Node *N1* is not rail-to-rail since voltage drops are needed from drain-to-source of transistors $M_{7-10}$ to keep them turned on.

Rise/fall times can be used to identify which node limits the maximum clock frequency of the global external magnetic field. The rise/fall times, simulated from 10% to 90% between signal low and high, for *N1*, *N2*, and *MCLK* are shown in Table 3.8 for implementations in both the 3M2P 0.5 $\mu$m and 5M1P 0.18 $\mu$m process technologies. Node *N1* of the current comparator is shown to be the bottleneck node with a rise/fall time of 9.275/7.767 ns in the 3M2P 0.5 $\mu$m process. This is primarily due to the high impedance of the cascoded output

Table 3.8: Rise/fall times (10–90%) and propagation delay of the R2V read circuit.

| | **Process** | | | |
|---|---|---|---|---|
| | **3M2P 0.5 $\mu$m** | | **5M1P 0.18 $\mu$m** | |
| **Node** | $t_{rise}$ | $t_{fall}$ | $t_{rise}$ | $t_{fall}$ |
| *N1* | 9.275 ns | 7.767 ns | 5.430 ns | 5.240 ns |
| *N2* | 2.702 ns | 2.694 ns | 0.805 ns | 0.809 ns |
| *MCLK* | 0.977 ns | 0.964 ns | 0.225 ns | 0.230 ns |
| **Characteristic** | $t_{p,rise2fall}$ | $t_{p,fall2rise}$ | $t_{p,rise2fall}$ | $t_{p,fall2rise}$ |
| Prop. Delay | 7.215 ns | 6.691 ns | 4.625 ns | 2.403 ns |

stage of the current comparator that gives large RC time constants. If speed is important, a faster current comparator such as in [52] can be used to improve performance.

The propagation delay of the R2V read circuit assesses the latency of reading the input resistance as sensed at the output. These are shown in Table 3.8. $t_{p,rise2fall}$ is the propagation delay from a rising transition of $R_{mtj}$ to a falling transition of $MCLK$. Conversely, $t_{p,fall2rise}$ is the propagation delay from a falling transition of $R_{mtj}$ to a rising transition of $MCLK$. The propagation delay is closely related to the rise/fall time bottleneck of the current comparator node *N1*. Therefore, reducing the rise/fall time of *N1* should also decrease the propagation delay of the circuit.

The clock frequency is limited by the high impedance and moderate capacitance at node *N1*. This is observed in Figure 3.14 where the frequency is swept from 1 MHz to 100 MHz. The peak-to-peak voltage of node *N1* decreases as the frequency increases beyond about 30 MHz since the node can no longer fully charge and discharge. The 3 dB point is simulated to be about 67 MHz, which is an estimate of the maximum frequency at which the circuit can operate.

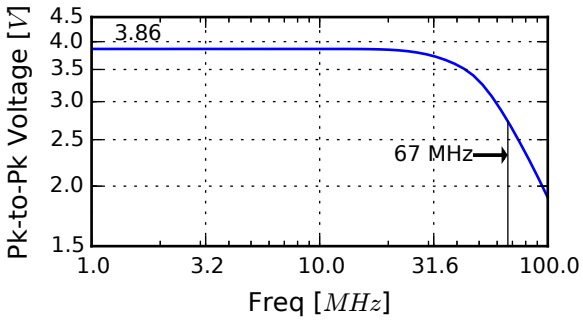


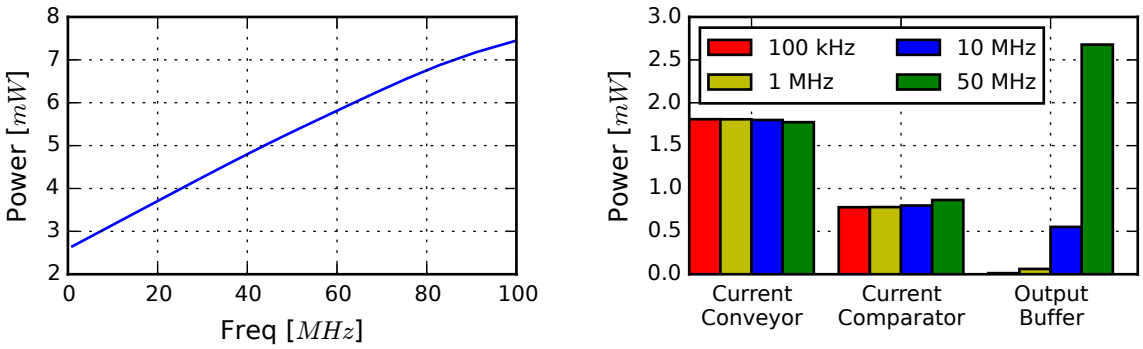

Figure 3.14: Node *N1* peak-to-peak voltage of the R2V read circuit from 1 MHz to 100 MHz frequency in the 3M2P 0.5 $\mu$m process. The 3 dB point is 67 MHz which indicates the maximum frequency of operation.

In the 5M1P 0.18 $\mu$m process, the rise/fall times and propagation delays are shorter with smaller transistor dimensions and node *N1* having less capacitance. The 3 dB point is simulated to be about 105 MHz.

### 3.3.3 Power

The average power of the R2V read circuit is shown in Figure 3.15a as a function of frequency from 1 MHz to 100 MHz. As the frequency increases, the total average power consumed increases roughly linearly. At 100 MHz, the circuit is exceeding its maximum frequency as determined earlier and is unable to switch fast enough. Since the internal node *N1* never fully charges or discharges, the subsequent output buffer begins to also not fully charge or discharge, thus there is overall less power consumed.

The power consumed by the read circuit limits the number of read circuits that can be built in a given power budget. The circuit continuously sinks current and consumes power. The power consumed is 3.21/1.98 mW when $R_{mtj}$ is low/high (500 $\Omega$/1 k$\Omega$). During switching, the instantaneous power spikes up due to dynamic power consumption in the output buffer. The average power consumed by this circuit at 10 MHz is 3.15 mW. If we exclude the output




(a) Average power plot

(b) Power breakdown plot

Figure 3.15: Average power and power breakdown plots of the R2V read circuit in the 3M2P 0.5 $\mu$m process. The average power plot is shown on the left and sweeps the frequency from 1 MHz to 100 MHz. The power breakdown plot is shown on the right.

buffer, the average power is 2.59 mW. A breakdown of the average power of each sub-circuit at varying clock frequencies is shown in Figure 3.15b.

In this figure, the current conveyor and current comparator consume about the same power independent of varying clock frequencies whereas the output buffer exhibits dynamic power that is linearly proportional to the clock frequency. The current conveyor consumes more than twice the power of the current comparator. This is due to current flowing through two branches in the current conveyor versus one in the comparator. Also, the average current in the comparator is a little less than that in the current comparator since the current flowing through the comparator cannot be greater than $I_{th}$.

For the 5M1P 0.18 $\mu$m process, the current flowing through each branch of the current conveyor and current comparator remains the same, but the power supply is now operating at 1.8 V instead of 5 V. This results in less overall power consumed. At 10 MHz, the total average power is 0.65 mW. Again, if we exclude the output buffer, the average power is 0.63 mW.

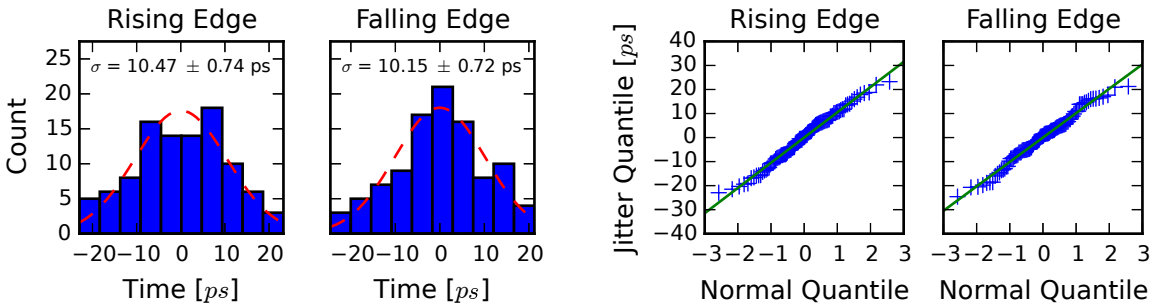
### 3.3.4 Jitter

Noise from the transistors in the R2V read circuit contribute to jitter, an uncertainty in time, in the *MCLK* output. Jitter on *MCLK* contributes to phase delay between two or more outputs. A Monte Carlo simulation is used to simulate the time-domain noise in the read circuit. Each noise source is replaced by a random variable and simulated in time. In the 3M2P 0.5 $\mu$m process technology, this simulation was done for 100 runs with thermal noise sources from 100 kHz to 1 GHz frequency. The jitter is calculated by measuring the delta time between when *MCLK* output crosses $V_{DD}/2$ compared to the average crossing time. A histogram of the time jitter is shown in Figure 3.16a for rising and falling edges with an overlaid Gaussian probability density function. The standard deviation, $\sigma$, of the jitter for the rising and falling edges is $10.47 \pm 0.74$ ps and $10.15 \pm 0.72$ ps, respectively.

A normal quantile-quantile plot, shown in Figure 3.16b, is a better way to test if the data are from a Gaussian normal distribution. In this plot, jitter quantiles are compared to Gaussian normal quantiles. Since the data form a straight line, it is determined to be Gaussian normal distributed. For a Gaussian normal distribution, 99.7% of all random samples occur within $\pm 3\sigma$. Therefore, the jitter can be as high as $6\sigma \approx 62.8$ ps for the rising edge or $6\sigma \approx 60.9$ ps for the falling edge.

### 3.3.5 Discussion of simulated circuits

We have designed an R2V read circuit to interface with MTJ devices. This circuit is characterized in terms of area, transient response, power, and jitter in the 3M2P 0.5 $\mu$m and 5M1P 0.18 $\mu$m process technologies. As the process scales down to smaller dimensions, the area decreases, rise/fall times decrease, propagation times decrease, maximum frequency increases,




(a) Jitter histogram

(b) Normal QQ plot

Figure 3.16: Jitter histogram and quantile-quantile (QQ) plot of the rising/falling edges of *MCLK* of the R2V read circuit in the 3M2P 0.5 $\mu$m process.

power consumption decreases, and jitter decreases. A summary of these results is given in Table 3.9.

From this investigation, we learned that the R2V read circuit has several areas for improvement. One, node *N1* in the current comparator has the highest rise/fall times, creating a performance bottleneck. A faster current comparator such as in [52] can be used to improve performance. And two, the power consumed and circuit area can be further reduced. Transistors $M_{1-3}$ must provide $I_{mtj}$ current flowing through it. But, transistors $M_{4-10}$ could operate on less current, thus allowing these transistors to be made smaller.

Table 3.9: Summary of results for the R2V read circuit.

| | **Process** | |
|---|---|---|
| **Characteristic** | **3M2P 0.5** $\mu$m | **5M1P 0.18** $\mu$m |
| Dim. of Layout | $31.35 \times 40.20\,\mu$m | $8.28 \times 27.27\,\mu$m |
| Area of Layout | $1,260\,\mu\text{m}^2$ | $226\,\mu\text{m}^2$ |
| $f_{N1,3dB}$ | 67 MHz | 105 MHz |
| $t_{N1,r/f}$ (10–90%) | 9.275/7.767 ns | 5.430/5.240 ns |
| $t_{N2,r/f}$ (10–90%) | 2.702/2.694 ns | 0.805/0.809 ns |
| $t_{MCLK,r/f}$ (10–90%) | 0.977/0.964 ns | 0.225/0.230 ns |
| $t_{p,r2f/f2r}$ | 7.215/6.691 ns | 4.625/2.403 ns |
| $P_{avg@10\,\text{MHz}}$ | 3.15 mW | 0.65 mW |
| $\sigma_{MCLK,jitter,r/f}$ | $10.47 \pm 0.74$ ps / $10.15 \pm 0.72$ ps | $8.67 \pm 0.62$ ps / $12.70 \pm 0.90$ ps |

r = rise, f = fall

# Chapter 4

# MTJ Read Circuit Experimentation in $0.5\,\mu$m Process

A test chip was designed and fabricated in the 3 metal 2 poly (3M2P) $0.5\,\mu$m process for testing the MTJ read circuit designed in Chapter 3 and to evaluate the feasibility of magnetic global clocking. The design was done using the Cadence Design Environment tools and simulated using the Spectre simulator. The chip was designed such that the magnetic tunnel junction (MTJ) devices can be replaced with resistors, thus allowing testing without MTJs. Due to us being unable to attain MTJs for testing, we are therefore limited to only experimentally testing the MTJ read circuit (which can use resistors) but not the magnetic global clocking (which needs actual MTJs for sensing the magnetic field).

In this chapter, we first describe the prototype chip containing test circuits for testing the read circuit and magnetic global clocking, and then describe the test setup and infrastructure which includes a custom printed circuit board (PCB), a field-programmable gate array (FPGA) board with custom firmware, and custom software to control the test setup. Last, we present performance results from experimentation of the MTJ read circuit.

## 4.1 Prototype chip

The prototype test chip, implemented in a complementary metal-oxide semiconductor (CMOS) process, is designed to test the MTJ read circuit and magnetic global clocking with circuitry dedicated to each. An overview of the chip is shown in Figure 4.1. This chip has dimensions 1.5 mm x 3.0 mm, which is the size of 2 tiny chips in the 3M2P 0.5 $\mu$m process. There are 58 total assigned pins with 8 pins on the bottom for MTJ wire bonded connections. Each pin can be defined as either analog (a) or digital (d), and as an input (in), output (out), or in/out (io). Therefore, e.g. a pin labeled as "(ain)" represents an analog input pin.

The test chip contains 4 quadrants and the common circuitry. Each quadrant contains a magnetic sense circuit (MSC), a shift register (SR), a 2x1 clock input select multiplexer for the shift register, and a 4x1 data input select multiplexer for the shift register. Collectively, a single quadrant forms a local "clock domain". The common circuitry contains biasing circuits (Biasing), a configuration shift register (Config SR), a bank of phase detectors (PD), a 4x1 voltage analog multiplexer (for probing the MTJ load voltage), and a 5x1 current analog multiplexer (for probing the MTJ current from the read circuit).

The chip contains a number of pins for power, biasing, configuration, test input/outputs, MTJ connections, and reset. Power pins include MSCVDDx (individual MSC power inputs which can be measured in testing), AVDD (analog power), DVDD (digital power), GND (ground), and MTJGND (the ground for the MTJ devices). One biasing pin, IBUF_BIAS, is a biasing current that biases the MTJ load voltage buffer. Configuration pins include SIN (serial input), SOUT (serial output), and SCLK (serial clock). Test input pins include CLKIN (clock input), SRINx (serial register input), ITHx (threshold current), and VBIAS (common bias or reference voltage for all read circuits). Test output pins include SROUTx

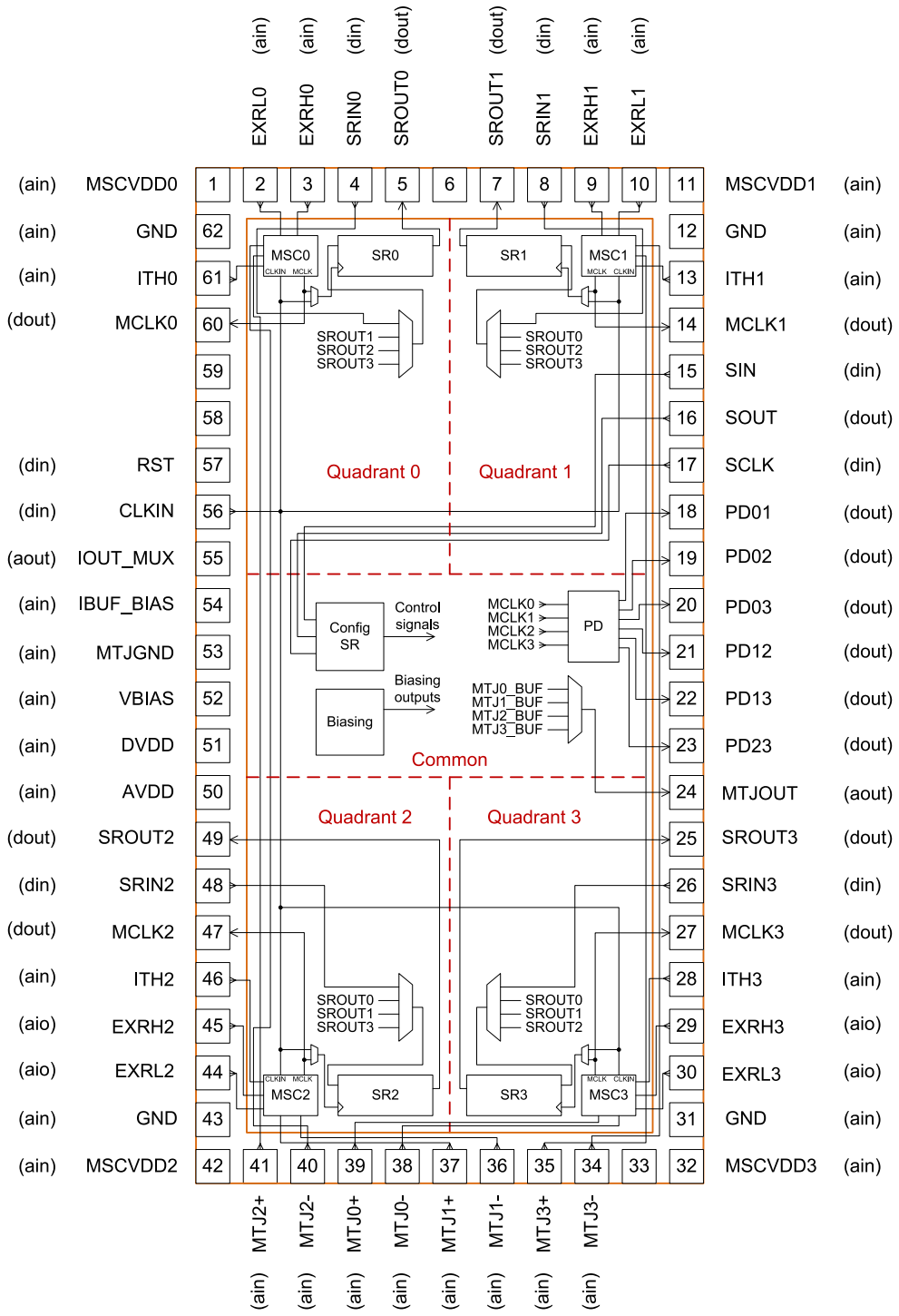



Figure 4.1: Prototype test chip schematic (overview). The test chip has dimensions 1.5 mm x 3.0 mm (2 tiny chips) with 58 total assigned pins and 8 pins on the bottom for MTJ wire bonded connections. The chip is divided into 4 quadrants, each containing a read circuit, and the common circuitry in the middle with a configuration register and additional test circuitry.

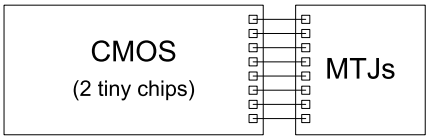



Figure 4.2: Chip bonding diagram. The CMOS test chip is fabricated in the size of 2 tiny chips (3.0 mm x 1.5 mm) and wire bonded to a die with fabricated MTJ devices.

(serial register output), MCLKx (magnetic clock output), PDxy (phase detector output), IOUT_MUX (multiplexed read circuit output current), and MTJOUT (multiplexed MTJ load voltage buffered output). MTJ connection pins are the bottom 8 MTJx+/- pins. And one reset pin, RST, performs a global, asynchronous reset of all digital circuits (which include the configuration register in the common circuitry and all shift registers in the quadrants).

The configuration register is located in the common circuitry and consists of a chain of D flip-flop registers that form a shift register. It is programmed and read via the SIN, SOUT, and SCLK pins. This register is used to configure multiplexers and switches on the chip that control test circuits for testing the read circuit and magnetic global clocking. The register is logically divided into sections for configuring the common circuitry and each quadrant. In the common circuitry, there are multiplexers for reading out the buffered MTJ load voltage and the MTJ read current from one of the quadrants. In each quadrant, there are switches that control the MSC, and multiplexers for selecting clock and data input sources for the shift registers.

The chip is designed to be wire bonded to MTJs that have been fabricated on another die. The setup for this is shown in Figure 4.2 where the CMOS chip is arranged longways on the left and the die containing the MTJs on the right. The MTJs are then wire bonded to the MTJ pins on the CMOS chip. This setup allows the CMOS chip and MTJs to be fabricated separately with connections made post-fabrication. Fabrication of MTJ devices directly on the CMOS die has been demonstrated by Durlam et al. [1].

### 4.1.1 Read circuit testing

The read circuit is contained in the magnetic sense circuit (MSC) of each quadrant. The MSC, shown in Figure 4.3, interfaces with an MTJ device or other resistive load and generates the output MCLK. The MTJ device is interfaced through the MTJx+/- pins located at the bottom of the chip. MTJ+ connects into the MSC whereas MTJ- connects to ground supplied by the MTJGND pin. The MSC consists of the read circuit, a bank of input sources, a bank of capacitors, and an output buffer. The read circuit, which is shown in Figure 3.11, has three inputs: the MTJ load (MTJ_LOAD), the VBIAS voltage, and the ITH threshold current. The MTJ load is connected to a network of switches that allow different resistive sources and load capacitances to be connected to the read circuit. This voltage can be monitored during testing via the MTJ_BUF output. The VBIAS voltage in the read circuit pins the voltage over the MTJ load. This in turn produces a current through the MTJ load which is compared to the ITH threshold current. The result of this comparison produces the digital logic voltage output at MCLK.

There are 11 switch options that are programmable in the MSC via the configuration register. Switches S0 – S2 control the input source selected. Switch S3 controls the connecting of the next section of switches. Switches S4 – S9 control the amount of extra load capacitance that can be added to the MTJ load. And, switch S10 controls the connecting of the output buffer for monitoring the MTJ load node voltage. Switches S1 and S2 connect a pair of integrated (i.e., internal) resistors and external resistors, respectively, that can be toggled within a pair via the CLKIN pin. This is done using decode logic, shown in Figure 4.4, which splits S1 and S2 into pairs of signals, producing S1L, S1H, S2L, and S2H. These in turn drive the switches in the MSC.

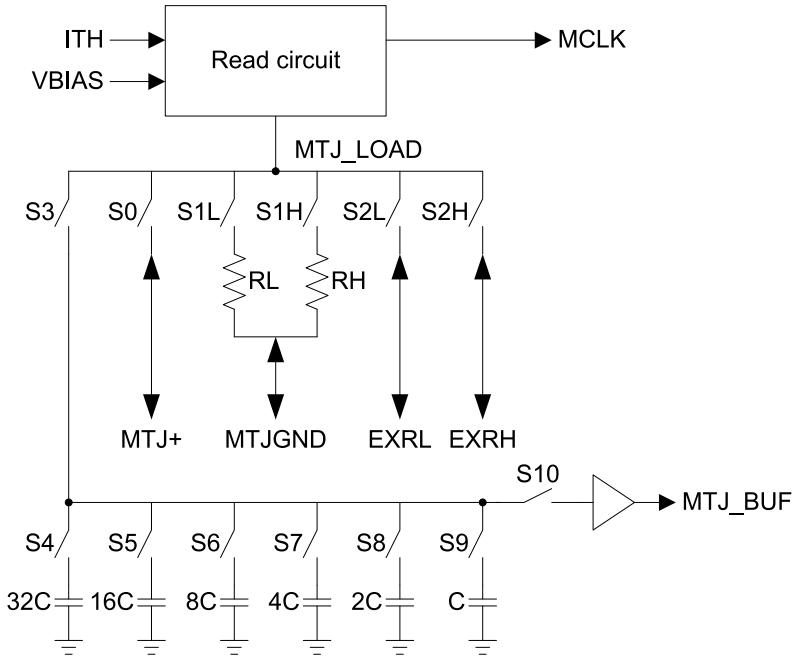



Figure 4.3: Magnetic sense circuit (MSC). This consists of a read circuit, a bank of input sources, a bank of capacitors, and an output buffer.

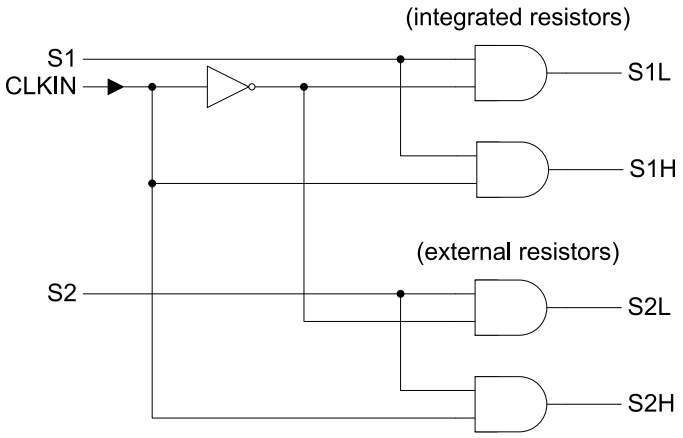



Figure 4.4: MSC decode logic. This logic decodes the S1 and S2 select signals further based on the logic value of the CLKIN signal to select which resistor in a pair is active.

Each switch has an associated "on" resistance that is added in series with the connection it makes. This resistance varies depending on the size of the transistor implementing the switch. Increasing $W/L$ of the transistor reduces this "on" resistance. This is only necessary, however, for switches that must carry current or need to switch quickly. Since switches S0 – S9 have this requirement, large transistors are used to implement them ($\frac{W}{L}$ is $\frac{45\,\mu\text{m}}{0.6\,\mu\text{m}}$) with an "on" resistance of $\approx 41\,\Omega$. Switch S10 carries no current and therefore is designed with minimum size.

In the MSC, there are three input sources that can be selected for the MTJ load: MTJ+ pin, integrated resistors, and external resistors. S0 selects the MTJ+ pin which is the wire bonded connection to the MTJ. S1 selects the integrated resistors which are integrated on the chip and have nominal values $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$. The bottom of the integrated resistors are connected to MTJGND. Last, S2 selects the external resistors which are settable off-chip via the EXRL (low resistance) and EXRH (high resistance) pins. These different modes of selecting the input provide the capability of testing the read circuit with and without MTJs.

The load capacitance bank provides the capability to add extra capacitance to the MTJ load by enabling switch S3 and programming the amount of capacitance by switches S4 – S9. The capacitance bank can be programmed from 0 to 31 with a unit capacitance of $C = 1\,\text{pF}$, thus allowing a range of 0 to 31 pF of extra capacitive load. This provides a way to test the relative insensitivity of the input node to capacitance as was determined as a result of the analysis in Section 3.1.

The MTJ load voltage can be monitored from the MTJ_BUF output by enabling switches S3 and S10. This in turn connects the buffer to the MTJ load voltage, which allows it to be read off-chip. The buffer is implemented as a P-source follower and incurs a gate-to-source offset in the voltage output. The amount of offset depends on the biasing of the P-source follower.

The P-source follower is biased by sourcing a current into the source of the transistor which is set via the current supplied into the IBUF_BIAS pin. Disabling S10, the buffer is driven with ground potential allowing the offset to be measured.

### 4.1.2 Global clock testing

In a magnetic global clocking system, read circuits distributed on a chip can be used to generate multiple clock signals that are synchronous to a global magnetic clock. Variations in the MTJ devices and transistors, circuit noise/jitter, and wire delays can all contribute to clock skew or phase delay between clock signals. To measure this phase delay and test global clocking, we have developed the test chip with multiple distributed read circuits and a phase detecting circuit. The test chip, shown earlier in Figure 4.1, is designed with four quadrants, each containing a read circuit (which is interfaced to an MTJ), a digital shift register, and two multiplexers (for selecting data and clock input sources to the shift register); and a phase detector circuit located in the center of the chip (for measuring phase delay between read circuits).

The read circuit senses an oscillating global magnetic field and produces a digital clock output called MCLK. The digital shift register shifts a bit pattern and is clocked by either CLKIN (from a pin) or MCLK (from the read circuit) through the clock multiplexer. The data input can be the SRINx pin of the quadrant or the output of the digital shift register of another quadrant selectable by the data multiplexer. This programmability in the data input allows interconnections to be made between quadrants, thus allowing global clocking to be tested with a long shift register formed by these interconnections.

The digital shift register in the quadrant, shown in Figure 4.5, consists of four D flip-flops which are clocked synchronously by CLK (the clock input set via the clock multiplexer) and

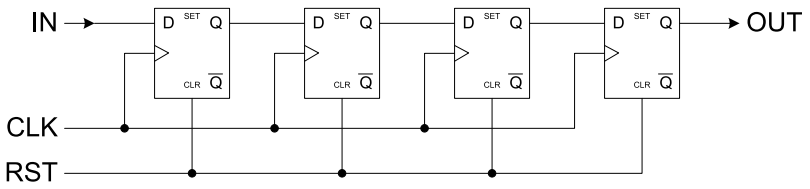



Figure 4.5: Digital shift register. This is constructed using a chain of four D flip-flops which are clocked synchronously by CLK and reset globally by RST.

reset globally by RST. The shift register can easily be tested in the lab for correctness by supplying a bit pattern at the input and observing the same bit pattern at the output. A string of shift registers formed by interconnections between each quadrant (for testing global clocking) can be tested for correctness in the same way.

Phase measurements are made between read circuits by the phase detector circuit located in the center of the chip. The main purpose of the phase detector is to measure large phase differences due to process variations in the MTJs or in the read circuits. The output of each read circuit is routed to the phase detector using equal length wire runs in order to have equal propagation delay of the distributed clock (i.e., the global magnetic field). The phase detector is designed using an array of symmetric XOR gates followed by buffers. The XOR gates are designed to be symmetric so that they will produce the same pulse output whether rising or falling edges occur, or whether one input arrives first or the other. The buffer is designed to drive a pin capacitance of 8 pF with 1 ns rise/fall time.

The range of the phase detector sets the smallest phase difference that can be detected. Since the speed of the buffer affects the pulse output, we consider the range of both the XOR gate and the buffer. A plot of the simulated phase detector response is shown in Figure 4.6. The input phase delay is set by adjusting the time between input resistance changes. The pulse width is measured by determining the difference in time of the rising and falling edges at the $V_{DD}/2$ voltage crossing point. In the plot, the response becomes non-linear at small pulse widths due to the pulse being too short to charge all the way to

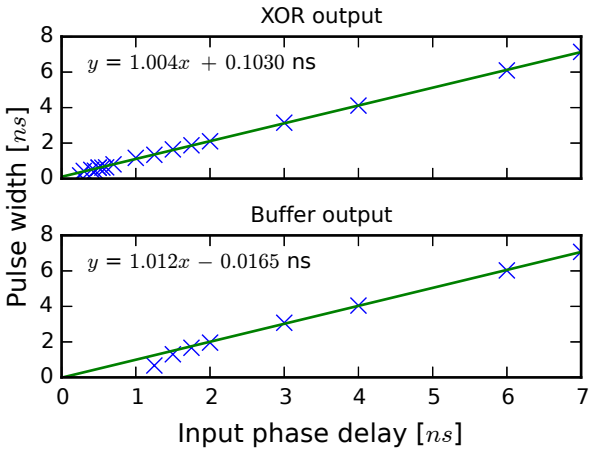



Figure 4.6: Simulated phase detector response plot in the 3M2P 0.5 $\mu$m process. The top plot shows the XOR output which detects as low as 0.25 ns and is linear above 0.4 ns. The bottom plot shows the buffer output which detects as low as 1.25 ns and is linear above 2 ns.

$V_{DD}$. With this in mind, the smallest detectable pulse width is 0.25/1.25 ns for the XOR and buffer outputs, respectively and are linear above 0.4/2 ns. The resistance-to-voltage (R2V) read circuit jitter, measured in simulation at $\sigma \approx 10$ ps (see Section 3.3.4) is insignificant compared to the smallest detectable phase delay. Hence, we anticipate any measured phase delay to be principally due to variations in MTJ devices.

### 4.1.3 Fabrication

The prototype test chip was fabricated in the 3M2P 0.5 $\mu$m process to produce five chips for testing. The layout of the chip is shown in Figure 4.7. Three chips were packaged and wire bonded by the manufacturer while the remaining two chips were left unpackaged (dies only). The three packaged, wire bonded chips are for testing the chip without MTJs with the MTJx+/- pads on the die wire bonded to package pins for testing purposes. The two unpackaged dies are for manually packaging and wire bonding the chip with MTJs connected directly to the MTJx+/- pads on the die.

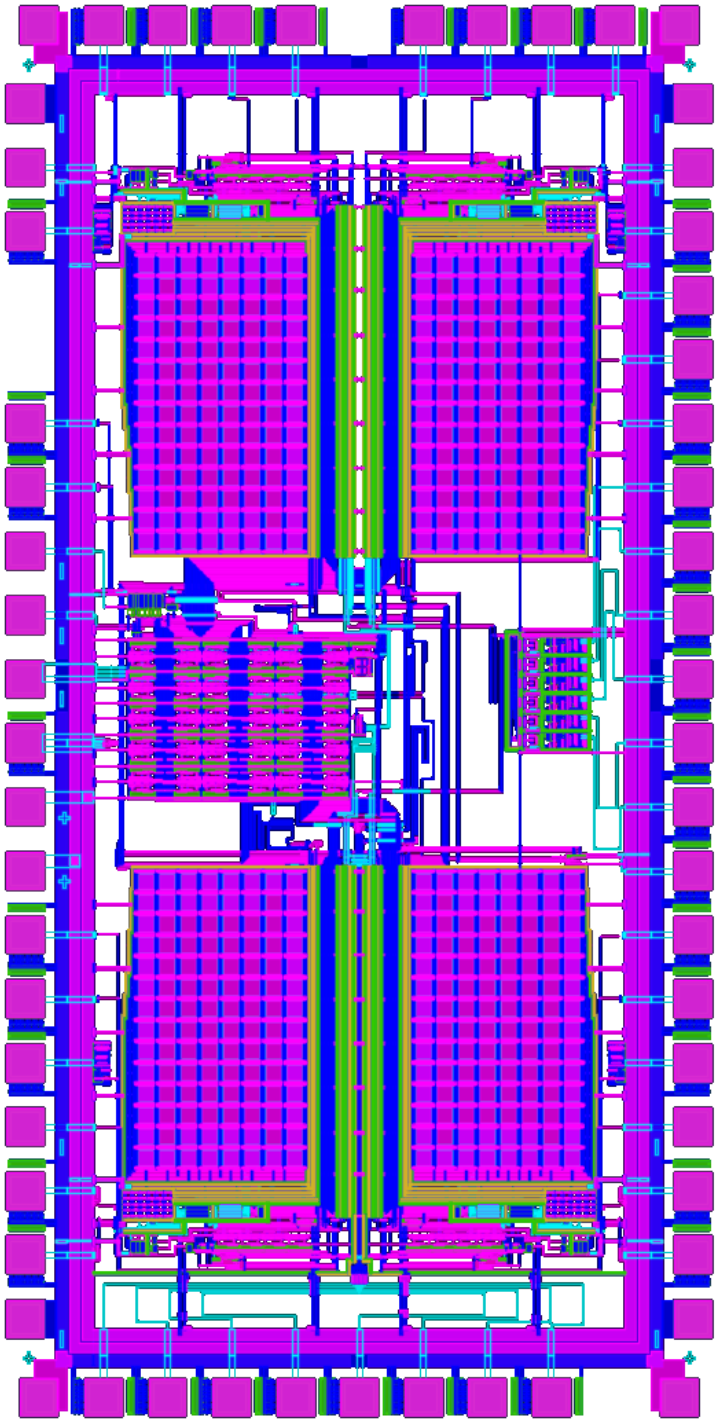


Figure 4.7: Prototype test chip layout.

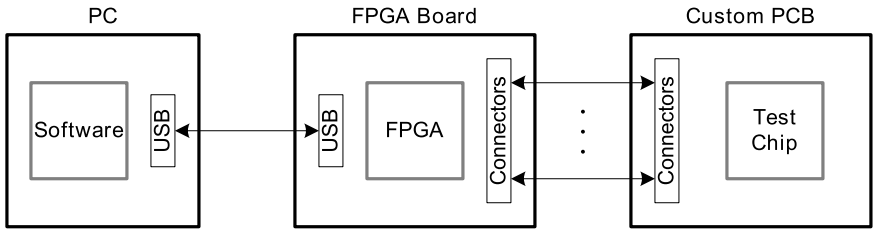



Figure 4.8: System overview of test setup. This includes a personal computer (PC) with software to control the test setup, an FPGA board with custom firmware to communicate with the PC and generate test signals to the custom printed circuit board (PCB), and the custom PCB to interface to the fabricated test chip.

## 4.2 Test setup

We have developed testing infrastructure for testing the fabricated prototype chip. An overview of the system is shown in Figure 4.8 which includes a personal computer (PC), an FPGA board, and a custom printed circuit board (PCB). The PC controls the entire test setup. Software written on the PC initializes the FPGA, PCB, and test chip; and controls experimentation of the circuits on the test chip. The PC interfaces to the FPGA board via a universal serial bus (USB) link. The FPGA board has I/O connectors that connect to the custom PCB that hosts the test chip. All test signals to the PCB and test chip are generated by the FPGA. The PCB hosts the test chip and contains test circuitry for supplying the necessary bias voltage, bias current, threshold currents, etc. to the test chip. Test points are provided on the PCB to observe signals to and from the test chip.

#### Parameters that can be set

There are several parameters that can be set in the system, shown in Table 4.1. The valid values of each parameter are given in the table. VBIAS is the bias voltage that is common to all read circuits in the test chip. This voltage pins the voltage over the MTJ load or resistance source, which sets the current flowing through the resistance. ITH is the threshold current

Table 4.1: Parameters that can be set.

| Parameter | Values | Description |
|---|---|---|
| VBIAS | 0 to 2 V | Common bias voltage |
| ITH<0:3> | 0 to 575 $\mu$A | Quadrant threshold current |
| IBUF_BIAS | 0 to 340 $\mu$A | Common buffer bias current |
| CLKIN | DC to 50+ MHz | Clock in test signal |
| Resistance source<0:3> | Integrated, external resistors, and/or MTJ | Quadrant resistance source(s); multiple selectable |
| Capacitance load<0:3> | 0 to 31 pF | Quadrant extra capacitance load added to MTJ load node |
| Data mux<0:3> | Local SRIN or another quadrant SROUT | Quadrant shift register data input multiplexer |
| Clock mux<0:3> | CLKIN, MCLK | Quadrant shift register clock input multiplexer |
| IOUT mux | Quadrants 0 to 3 or unconnected | Output current multiplexer |
| MTJ_OUT mux | Quadrants 0 to 3 | MTJ load output multiplexer |

which is settable for each quadrant, and used to compare the output current to produce a digital logic voltage output. IBUF_BIAS is a buffer biasing current that is supplied once into the chip and mirrored to each buffer (one in each quadrant) for reading out the MTJ load node voltage. CLKIN is the clock input signal and is programmable from DC to approximately 50 MHz.

The resistance source is selectable via switches in each quadrant as was previously shown in Figure 4.3. These switches (S0 – S2) are programmed via the configuration register and can allow 0, 1, or more resistance sources to be connected. The capacitance load is a bank of capacitors that can be added to the MTJ load node. The data multiplexer selects which data input to connect to the shift register in the quadrant. The clock multiplexer likewise selects which clock input to connect to the shift register in the quadrant. The IOUT

multiplexer selects which IOUT (output current from the read circuit) to supply out to a pin for measurement. Last, the MTJ_OUT multiplexer selects which MTJ_OUT (the buffered MTJ load node voltage) is brought out to a pin for measurement.

**Signals that can be measured**

The following is a (non-comprehensive) list of signals that we can measure in our test setup:

- Current flow into the MSC of a quadrant
- Direct voltage of the MTJ load node under certain conditions
- IOUT of a quadrant (the output current)
- MTJOUT of a quadrant (buffered MTJ load node voltage)
- MCLK of a quadrant (magnetic clock output from read circuit)
- SROUT of a quadrant (digital shift register output)
- PDxy (phase difference between xy combination of MCLKs)
- SOUT of the configuration register (serial output)

Direct voltage measurement of the MTJ load node is possible since it can be accessed through a switch with zero current flow. Therefore, activating a switch in Figure 4.3 that is unused and accessible outside of the chip (e.g. S0 or S2) can be used to take a direct voltage measurement.


Figure 4.9: Custom printed circuit board (PCB) for testing and interfacing to the prototype test chip. The test chip is populated into a PGA65 socket on the board. On the bottom side, the PCB is connected to an FPGA board.

### 4.2.1 PCB design

The PCB is a custom board designed in Cadence Allegro PCB Designer and manufactured using Sunstone PCBexpress. This board is needed in order to test and interface to the prototype test chip and is shown in Figure 4.9. It provides test circuitry for supplying the necessary power supplies, bias voltage, bias current, threshold currents, etc. needed to test the prototype test chip. The chip is seated into a PGA65 socket that is soldered onto the board. The socket allows the chip to be inserted and removed as needed which also allows multiple chips to be tested.

There are two serial-input, voltage-output digital-to-analog converter (DAC) chips on the PCB: a single-channel DAC that sources VBIAS, and an 8-channel DAC that sources voltages that generates the ITH<0:3> and IBUF_BIAS currents for the test chip. The currents are generated using bipolar junction transistors (BJTs). For an individual BJT, the output voltage from the 8-channel DAC drives the base of the BJT through a series resistor. This in turn produces an emitter current through the BJT that is a function of the base current. The DAC code needed to produce a desired current can be determined by performing a calibration of the DAC code to an observed output current. This is done later in software in Section 4.2.3.

Two 80-pin I/O connecters are placed on the PCB to connect to the FPGA board. The two boards then connect together vertically via these connectors. This allows test signals to be generated by the FPGA that can be used on the PCB. The FPGA operates at 3.3 V and some components, like the test chip, operate at 5 V. Therefore, logic level translation chips are needed to translate between these two voltage levels. The logic level translation chips used on the PCB are bidirectional.

The PCB supplies power using four voltage regulators. These are digital 3.3 V, analog 3.3 V, digital 5 V, and analog 5 V. The analog and digital power supplies are isolated so as to prevent digital circuits from inducing noise in the analog power supply due to switching. Power is distributed on the PCB via a partitioned power plane where each partition distributes a separate power net. The test chip also uses multiple power supplies that isolate analog and digital power, and the power to each MSC in the quadrants. Isolating the power to each MSC allows individual power measurements to be taken of the read circuit during experimentation. Jumpers are placed on the PCB for each MSC power supply to allow current measurements to be taken using a multimeter. Power can then calculated as voltage times current.

External resistors for the read circuit are wired on the PCB board via two 16-pin sockets. The $R_L$ resistors for all four quadrants are supplied via one socket, and the $R_H$ resistors for all four quadrants are supplied via the other socket. Equal length wire runs are made from the sockets to the test chip pins so as to keep equal wire resistance and propagation delay. This is done in the layout editor by specifying an electrical constraint on the layout.

The MTJ+ connection in the test chip is also brought out to a pin for each quadrant. To access these inputs, axial resistor footprints are placed on the PCB that are connected to the MTJ+/- pins of the chip. These connections are routed with equal length wire runs.

Test points are placed on the PCB that give access to various input and output signals that are desired during testing. These include MCLK<0:3>, SROUT<0:3>, SOUT, MTJOUT, CLKIN, and PDxy. The wire runs for MCLK<0:3> are also made using equal length wire runs.

### 4.2.2 FPGA board

The FPGA board is an XEM6010 USB integration module manufactured by Opal Kelly [53] based on the Xilinx Spartan-6 FPGA. This board interfaces to the PCB via two 80 pin I/O connectors that connect FPGA pins to PCB signals and to the PC via a USB controller that allows the FPGA to transfer data to/from the PC.

FPGA firmware is written to receive commands from the PC and to interface with the PCB to configure components and generate test signals. Hardware modules are written to accomplish this. Commands sent by the software on the PC are received by a command module on the FPGA. The command module contains a finite-state machine that processes and executes the commands. The commands include configuring two DACs on the PCB that program VBIAS, ITH<0:3>, and IBUF_BIAS; programming the configuration register in the test chip; generating the CLKIN signal used to stimulate the read circuit in the test chip for experimentation; and sending reset to the test chip. The two DACs and configuration register are programmed via a serial peripheral interface (SPI) which is handled by an SPI module on the FPGA for each of them. This module receives data from the command module and serially shifts the data into the target component which programs it.

The CLKIN signal is generated from one of five sources selectable via a multiplexer. The five sources are (1) a 250 kHz fixed frequency clock, (2) a programmable clock derived from the FPGA clock (running at 100 MHz) using a divide-by-$n$ counter, (3) the logic 0, (4) the

logic 1, and (5) a phase-locked loop (PLL) programmable clock. The PLL programmable clock is provided on the FPGA board and is programmable with parameters $P$, $Q$, and a divider, $Div$, such that the resultant frequency is $48\,\text{MHz} \times P/\left(Q \cdot Div\right)$. Appropriately selected parameters allow for a wide variety of frequencies that can be generated, particularly high frequencies above 1 MHz nearing the FPGA clock frequency that cannot be generated by the divide-by-$n$ counter.

### 4.2.3 Software

The PC interfaces with the FPGA board via a USB link and provides high-level test control for the system. The software on the PC is written in Python. Opal Kelly provides an application programming interface (API) library that contains methods to interface directly with the hardware modules on the FPGA. We use this library in our Python software to build the high-level test control of the system.

The Python software consists of three parts. The first part is a board class and sub-classes that provide high-level methods for interfacing to the FPGA, PCB and test chip. They provide high-level test control of the entire system. The second part is calibration. The DACs on the PCB are used to generate voltages and currents, however, the input to each DAC is a code. Calibration of the DAC code to the generated voltage or current allows us to specify a function that can map a desired voltage or current back to the necessary DAC code. This allows us to provide methods in the board class that can take these voltages and currents as input and directly produce them on the PCB for the test chip. The third part is testing. With a written board class, test functions can now use the board class to setup the test chip, configure parameters, and run an experiment. Measurements can then be taken using lab equipment.

The board class is the top class that interfaces to the FPGA, PCB, and test chip. It utilizes sub-classes for accessing and programming the DAC chips and the configuration register in the test chip. High-level methods are provided that program the CLKIN signal; that set VBIAS, IBUF_BIAS, and ITH<0:3>; and that program the configuration register in the test chip. CLKIN can be selected from one of five sources as was described in the previous section. If a programmable source is used (either the divide-by-$n$ counter or PLL), methods are provided that can set the desired frequency by calculating the appropriate count or algorithmically searching for the PLL parameters needed. The VBIAS, IBUF_BIAS, and ITH<0:3> signals are set using calibration functions that calculate the DAC code needed to produce the desired values. The configuration register is programmed from a bit string that is constructed from several classes that model the parameters of the chip. Once those parameters are set, the bit string can be assembled and programmed.

Calibration is accomplished by cycling through a set of DAC codes and measuring the observed output. When the output is a voltage (such as for VBIAS), the input/output relationship is linear because the voltage-output DAC is linear. Therefore, a simple linear regression can be used to obtain the input/output relationship. When the output is a current (such as for IBUF_BIAS and ITH<0:3>), the input/output relationship is non-linear. Therefore, linear regression cannot be used. Instead, interpolation using curves (C-spline) is applied to determine the values in-between measured points. Interpolation works well in the forward direction when interpolating the current, but not so well in the reverse direction when interpolating the DAC code. To work around this, we interpolate in the forward direction, but use a numeric solver to determine the reverse DAC code needed to get the desired current output.

## 4.3 Experimental results

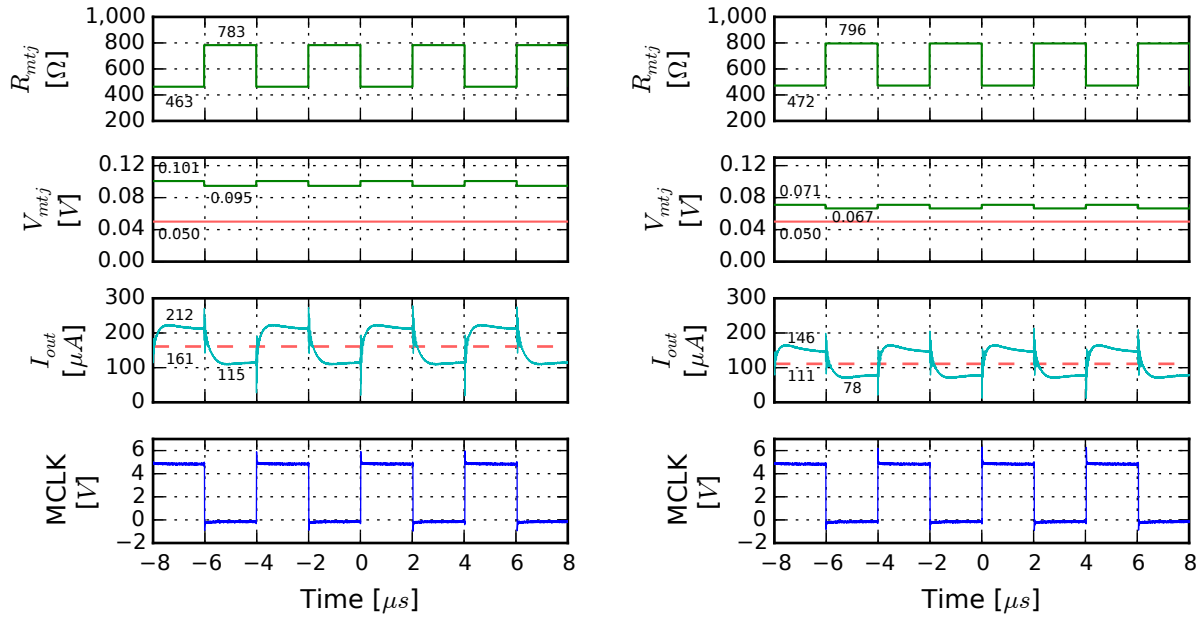
We now present experimental results for the prototype test chip on the MTJ read circuit.

### 4.3.1 Functional and performance testing

#### Functional testing

We first test the functionality of the MTJ read circuit in all four quadrants of the prototype test chip. The read circuit in each quadrant is connected to a pair of internal resistors that are integrated inside the chip with nominal resistance values $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$. The actual fabricated values of the internal resistors are measured empirically in Table 4.3. $V_{bias}$ is set to 50 mV and CLKIN is switched at $f_{CLKIN} = 250\,\text{kHz}$. There are four waveforms shown in Figure 4.10 for each quadrant of the test chip. The first, $R_{mtj}$, is the input resistance to the MTJ read circuit; the second, $V_{mtj}$, is the voltage over $R_{mtj}$; the third, $I_{out}$, is the output current (equal to the current flowing through $R_{mtj}$); and the fourth, MCLK, is the magnetic clock (digital) output voltage. The $R_{mtj}$ and $V_{mtj}$ waveforms are synthesized and shown in green. The $I_{out}$ waveform is a derived measurement and shown in dark cyan. And, the MCLK waveform is a direct measurement from the oscilloscope and shown in blue. All measurements captured from the oscilloscope are taken with 16 averages. The low/high values of each waveform, as well as the nominal and threshold values, are annotated on the plot adjacent to the waveforms.

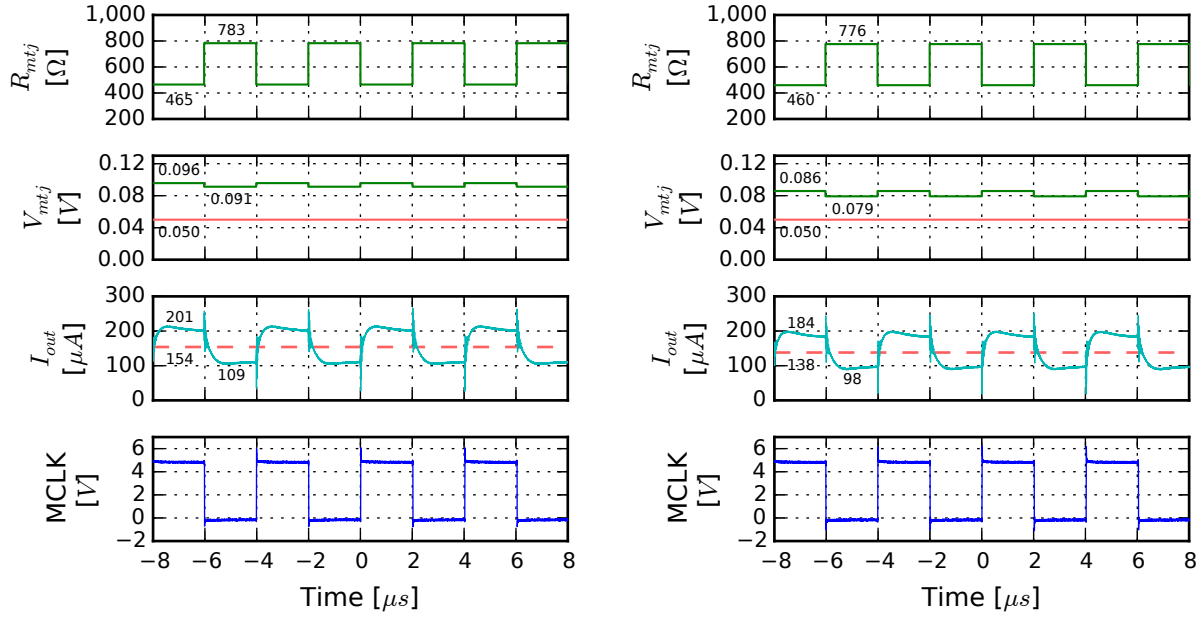
In the operation of the read circuit, the CLKIN signal switches the input resistance, $R_{mtj}$, low and high between two values ($R_L$ and $R_H$). Since $R_{mtj}$ cannot be directly measured, we instead determine these values individually at DC (see Section 4.3.2 below) and then create




(a) Quadrant 0




(b) Quadrant 1




(c) Quadrant 2




(d) Quadrant 3

Figure 4.10: Functional test of the MTJ read circuit in each quadrant. The read circuit is connected to a pair of internal resistors with $V_{bias} = 50\,\text{mV}$ and switched at $f_{CLKIN} = 250\,\text{kHz}$. Measurements were taken with 16 averages on the oscilloscope.

the $R_{mtj}$ waveform with rising/falling edges determined by the CLKIN waveform and with low/high values set to the calculated $R_L$ and $R_H$ values. The $V_{mtj}$ waveform is generated similarly with low/high values determined from DC measurements. $V_{mtj}$ should nominally be equal to $V_{bias}$ which is drawn as a solid red line. However, due to second-order effects and circuit variations, we can see in the plots that $V_{mtj}$ is higher than the nominal value and varies significantly between quadrants. This accounts for the variation in the output current which scales by this voltage.

The $I_{out}$ waveform is derived based on the voltage over a load resistor measured on the oscilloscope. In the test setup, IOUT_MUX is a pin that comes out of the test chip. It is connected to a load resistor with a value of $R_{load} = 4,669\,\Omega$. The current therefore is calculated as $V_{iout,mux}/R_{load}$ where $V_{iout,mux}$ is the voltage over the load resistor and at the IOUT_MUX pin. Inverting $R_{load}$, we get $1/R_{load} = 214.2\,\mu\mho$, which is a factor by which $V_{iout,mux}$ can be multiplied to calculate the $I_{out}$ current in $\mu A$. For each quadrant, the $I_{out}$ waveform is low or high, depending on $R_{mtj}$. An appropriate mid-way threshold current, $I_{th}$, is chosen for each and is indicated as a dashed red line. The MCLK waveform is directly measured from the oscilloscope which shows the successful logical reading of the $R_{mtj}$ input resistance.

We simulated the read circuit earlier in the transient response plot shown in Figure 3.13. The $V_{bias}$ used in that simulation was different, equaling 0.1 V. This means that we cannot do a quantitative comparison between the two results, but rather a qualitative one. We can see that the transient response of the measured results of the physical test chip are essentially the same as that shown in simulation. $R_{mtj}$ is stimulated in the same way. $V_{mtj}$, through feedback, is set near to the $V_{bias}$ voltage. $I_{out}$ shows a similar response. And, MCLK shows the same square wave response. Of some of the differences, the measured internal resistors on the test chip are all lower than the nominal values in simulation. The measured $V_{mtj}$

voltage on the test chip tends to show higher values above the nominal than in simulation. This consequently results in a higher measured $I_{out}$ current on the test chip. And, the measured $V_{mtj}$ voltage shows significant variation between quadrants on the test chip. With an appropriately chosen $I_{th}$ threshold, both simulation and measured results produce the same MCLK output.

**Performance testing**

Next, we test the performance of the read circuit using external and internal resistors to see how fast the read circuit can be clocked and still give a stable MCLK waveform output. Note, in our test setup, CLKIN is designed to operate at a maximum of 50 MHz.

The first performance test of the read circuit uses external resistors with values $R_L = 498\,\Omega$ and $R_H = 995\,\Omega$. In the test setup, $V_{bias} = 0.1\,\text{V}$, $I_{th} = 237.5\,\mu\text{A}$, and 64 averages are taken from the oscilloscope trace. These input clock frequency is gradually increased until the MCLK output waveform is no longer stable. The highest stable clock frequency found with no jitter on the oscilloscope trace was $f_{CLKIN} = 43\,\text{MHz}$. The results are shown in Figure 4.11. In this figure, CLKIN is the input clock which toggles the input resistance low/high, MCLK is the magnetic clock output signal from the read circuit, and the dashed gray lines are the power supply rails. The figure shows that the quality of the square wave for both CLKIN and MCLK is diminished at 43 MHz, but is still producing a correct result.

The second performance test of the read circuit uses internal resistors with measured values $R_L = 466\,\Omega$ and $R_H = 789\,\Omega$. In the test setup, $V_{bias} = 0.1\,\text{V}$, $I_{th} = 263\,\mu\text{A}$, and there is no averaging of the oscilloscope trace. The input clock frequency is again gradually increased until the MCLK output waveform is no longer stable. The highest stable clock frequency found with no jitter on the oscilloscope trace was $f_{CLKIN} = 48\,\text{MHz}$, which is

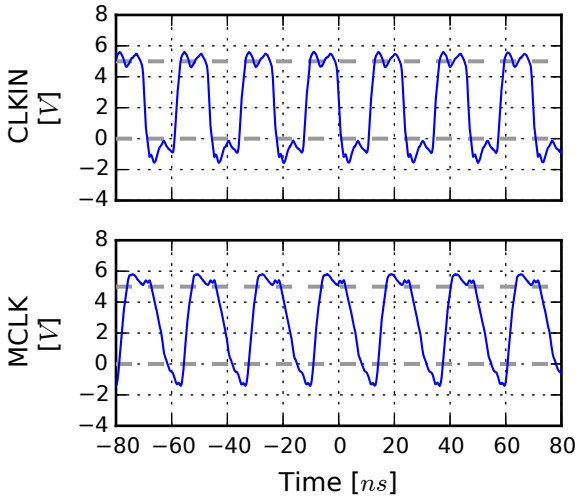
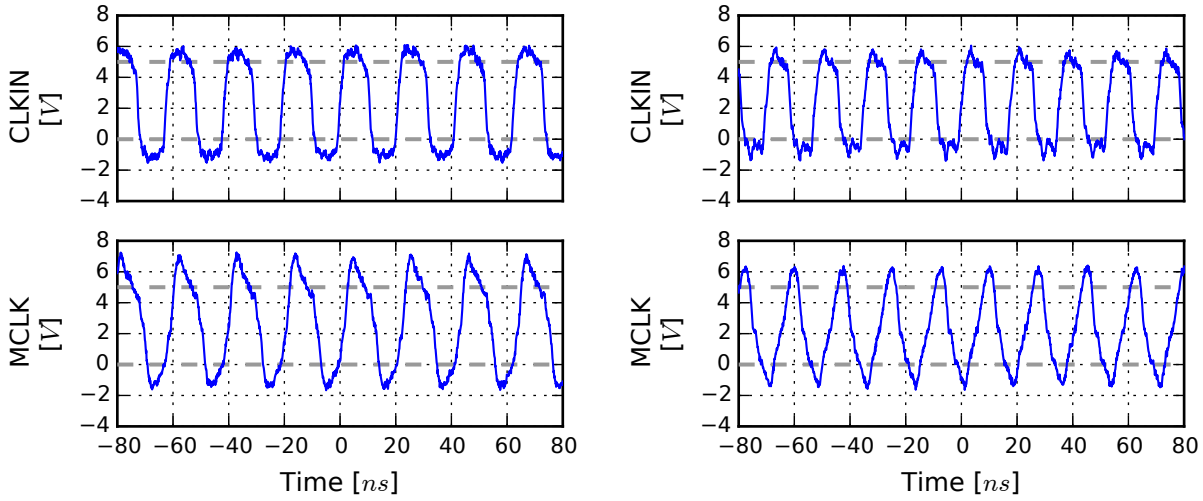



Figure 4.11: Performance test of the MTJ read circuit in quadrant 0 for external resistors at 43 MHz. The external resistors are measured at $R_L = 498\,\Omega$ and $R_H = 995\,\Omega$. $V_{bias} = 0.1\,\text{V}$ and $I_{th} = 237.5\,\mu\text{A}$ with 64 averages. CLKIN is the input clock, MCLK is the output of the read circuit, and the gray dashed lines are the power supply rails.

shown in Figure 4.12a. Pushing the clock frequency further, and above the maximum design specification of 50 MHz, we start to get jitter in the MCLK waveform. We are able to go as high as $f_{CLKIN} = 57\,\text{MHz}$ with jitter while still producing a valid MCLK waveform output that goes rail-to-rail. This is shown in Figure 4.12b. Again, in these figures, CLKIN is the input clock, MCLK is the magnetic clock output signal from the read circuit, and the dashed gray lines are the power supply rails. We can see in the 48 MHz plot that MCLK still resembles a square wave somewhat, but looks like a triangle wave in the 57 MHz plot.

Using these performance plots, we can now empirically measure the propagation delay from input to output and compare these results to those predicted in simulation. To do this, we measure two propagation delay values. One is from the rising edge of CLKIN to the falling edge of MCLK labeled as $t_{p,rise2fall}$. The other is from the falling edge of CLKIN to the rising edge of MCLK labeled as $t_{p,fall2rise}$. We do this for the performance plot with external resistors at 43 MHz and the performance plot with internal resistors at 48 MHz,




(a) $f_{CLKIN} = 48\,\text{MHz}$ (b) $f_{CLKIN} = 57\,\text{MHz}$

Figure 4.12: Performance test of the MTJ read circuit in quadrant 0 for internal resistors at two clock frequencies. The internal resistors are measured at $R_L = 466\,\Omega$ and $R_H = 789\,\Omega$, $V_{bias} = 0.1\,\text{V}$, and $I_{th} = 263\,\mu\text{A}$ with no averaging. CLKIN is the input clock, MCLK is the output of the read circuit, and the gray dashed lines are the power supply rails. On the left is the max stable frequency with no jitter in MCLK. On the right is the max frequency that MCLK is valid but has jitter.

which both have a stable MCLK waveform output. The results of these measurements are shown in Table 4.2. From these results, we can observe that the external resistors have higher propagation delays than the internal resistors. This is reasonable because the external resistors have additional wire resistance and capacitance for the connection off-chip that can slow down the read circuit. Comparing these results to the simulation results in Table 3.8, we can observe that the measured propagation delays are about twice as large as the simulation results. The simulation results, repeated here, are $t_{p,rise2fall} = 7.215\,\text{ns}$ and $t_{p,fall2rise} = 6.691\,\text{ns}$ in the 3M2P $0.5\,\mu\text{m}$ process. These measured propagation delays compared to simulation are also reasonable because there are many real-world effects missing in simulation that add delay in the physical measurement.

Table 4.2: Measured propagation delay of the read circuit in the test chip using external and internal input resistances.

| | **Performance Tests** | |
|---|---|---|
| **Characteristic** | **External @** 43 MHz | **Internal @** 48 MHz |
| $t_{p,rise2fall}$ | 16.99 ns | 12.43 ns |
| $t_{p,fall2rise}$ | 15.26 ns | 11.97 ns |

**Power measurements**

Last, we experimentally measure the power consumed by the read circuit by (1) doing a static analysis, and (2) dynamically generating CLKIN with a square wave and measuring the power via a pin. There are four MSCVDD power pins, one for each quadrant, that provide a way to take direct power measurements of the read circuit in each quadrant of the test chip. The MSCVDD power supply is internally connected to the read circuit and the output current transistors (which are accessible via the IOUT_MUX pin). Power measurements can be taken by measuring the average MSCVDD voltage and current during circuit operation. To exclude dynamic power effects due to switching, the circuit is tested at low frequency (1 kHz). This causes the power due to the output buffer to be negligible (with nearly zero current flow).

We start by doing a static analysis of the circuit. For our test setup, we use internal resistors in quadrant 0 with nominal values 500 Ω and 1 kΩ (the actual resistances are estimated to be $R_L = 466\,\Omega$ and $R_H = 789\,\Omega$). $V_{bias} = 0.1\,\text{V}$ and $I_{th} = 263\,\mu\text{A}$. Then, we directly measure the power supply voltage to be $V_{DD} = 4.989\,\text{V}$, and the current through the low/high resistances to be $I_{RL,out} = 328.1\,\mu\text{A}$ and $I_{RH,out} = 187.6\,\mu\text{A}$. Also in our test setup, the IOUT_MUX current is connected from the read circuit in the quadrant to the PCB via an internal multiplexer. This will contribute to the total power and must be taken into account.

We can calculate the total power as (current conveyor + current comparator + IOUT_MUX current)$\cdot V_{DD}$. In each term of this expression, we need to calculate the average current flowing.

The average current flowing through the current conveyor is the average of $I_{RL,out}$ and $I_{RH,out}$ times two (since there are two branches). This gives the following equation:

$$I_{current,conveyor} = I_{RL,out} + I_{RH,out}. \tag{4.1}$$

Next, the average current flowing through the current comparator is the average of the smallest current, $I_{RH,out}$, and the threshold current, $I_{th}$. This is because the current flow in the current comparator is limited by $I_{th}$. We then get the following equation:

$$I_{current,comparator} = \frac{I_{RH,out} + I_{th}}{2}. \tag{4.2}$$

Last, the average current flowing out from the IOUT_MUX pin is

$$I_{out,mux} = \frac{I_{RL,out} + I_{RH,out}}{2}. \tag{4.3}$$

From these three expressions, total power is then calculated as

$$\begin{aligned} P_{total} &= (I_{current,conveyor} + I_{current,comparator} + I_{out,mux}) \cdot V_{DD} \\ &= \left( (I_{RL,out} + I_{RH,out}) + \left( \frac{I_{RH,out} + I_{th}}{2} \right) + \left( \frac{I_{RL,out} + I_{RH,out}}{2} \right) \right) \cdot V_{DD} \\ &= \left( \frac{3}{2} I_{RL,out} + 2 I_{RH,out} + \frac{1}{2} I_{th} \right) \cdot V_{DD} \qquad (4.4) \\ &= 4.98\,\text{mW}. \qquad (4.5) \end{aligned}$$

Note, this equation does not include the output buffer which we can neglect because our power measurements are performed at low frequency.

Next, we measure the power using a dynamic test by generating CLKIN with a square wave and measuring the average current flowing into the MSCVDD pin for quadrant 0. For the CLKIN square wave, we set the frequency to $f_{CLKIN} = 1\,\text{kHz}$. We then measure the average current flowing into MSCVDD to be $994\,\mu\text{A}$. Multiplying this by the $V_{DD}$ power supply voltage measured above, we get a total power of 4.96 mW. This value is nearly identical to the total power calculated in the static analysis in (4.5), which verifies that our equations above are correct and that the read circuit is operating as expected.

What we are really interested in is the total power of the read circuit without the IOUT_MUX current. To calculate this, we merely exclude that term from the static analysis above. Doing so gives the following revised equation and result:

$$\begin{aligned} P_{total} &= (I_{current,conveyor} + I_{current,comparator}) \cdot V_{DD} \\ &= \left( (I_{RL,out} + I_{RH,out}) + \left( \frac{I_{RH,out} + I_{th}}{2} \right) \right) \cdot V_{DD} \\ &= \left( I_{RL,out} + \frac{3}{2} I_{RH,out} + \frac{1}{2} I_{th} \right) \cdot V_{DD} \qquad (4.6) \\ &= 3.70\,\text{mW}. \qquad (4.7) \end{aligned}$$

This result is comparable to our earlier simulation result in Section 3.3.3 of 2.59 mW, although it is higher. The measured power is higher because the input resistances and the $V_{mtj}$ voltage on the fabricated chip are higher than expected. This consequently produced high output currents, resulting in a higher average power measured on the chip.

### 4.3.2 Static measurements

Next, we take static measurements to learn about the static characteristics and response of the read circuit.

#### Internal resistor value estimates

The internal resistors in each quadrant of the test chip are designed to be nominally $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$ and are in series with a transistor switch. Due to the resistance added by the switch and variability in the fabrication process, their actual measured values are something different. We can determine the effective resistance values empirically by measuring the $V_{mtj}$ node voltage and the $I_{out}$ current, and then calculating the resistance as $R_{mtj} = \frac{V_{mtj}}{I_{out}}$. The $V_{mtj}$ node voltage is measured directly by first enabling the external resistor or MTJ connection in the MSC and then measuring the voltage via that connection. Since there is zero current flow, the measured voltage will accurately represent the $V_{mtj}$ node voltage. The $I_{out}$ current is measured directly using an ammeter via the IOUT_MUX pin.

The estimated internal resistor values for both low ($R_L$) and high ($R_H$) resistances are shown in Table 4.3. They are determined across values of $V_{bias}$ and all four quadrants of the test chip (indicated by $Q_{0\text{-}3}$) and are measured to be within $466 \pm 6\,\Omega$ and $787 \pm 11\,\Omega$, respectively. The estimated values for both $R_L$ and $R_H$ are shown to be less than their nominal values by varying amounts. This contributes to higher output currents than otherwise expected. As $V_{bias}$ varies, we can observe that the internal resistor values vary only slightly. This variation occurs due to the non-linear resistance of the series switch and second order effects in the $I_{out}$ current that are dependent on $V_{bias}$ (i.e., the $I_{out}$ current is not a perfect copy of the MTJ current). Between quadrants, we can also observe that the resistance values vary only

Table 4.3: Internal resistor value estimates in each quadrant of the test chip. The estimates are a function of $V_{bias}$ and vary between quadrants. The four quadrants are indicated by $Q_{0\text{-}3}$.

| | Nom. 500 Ω $R_L$ [Ω] | | | | Nom. 1 kΩ $R_H$ [Ω] | | | |
|---|---|---|---|---|---|---|---|---|
| $V_{bias}$ [V] | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ |
| 0.010 | 459 | 468 | 460 | 455 | 774 | 791 | 773 | 763 |
| 0.050 | 463 | 472 | 465 | 460 | 783 | 796 | 783 | 776 |
| 0.100 | 466 | 473 | 468 | 464 | 789 | 800 | 789 | 783 |
| 0.200 | 470 | 476 | 472 | 468 | 796 | 803 | 796 | 790 |

slightly and remain about the same. These results show that the effective internal resistance is relatively stable across both $V_{bias}$ and the chip quadrants.

### $V_{mtj}$ node and output buffer calibration

The output buffer in the MSC provides a way to read out the $V_{mtj}$ node voltage in an indirect way. The P-source follower incurs a gate-to-source offset in the voltage output that has to be compensated for. The amount of offset depends on the amount of biasing current that is passed through the P-source follower. For our test setup, we choose to use a biasing current of 10 $\mu$A. By measuring the direct $V_{mtj}$ node voltage and the $V_{mtj,buf}$ output voltage, a linear regression can be performed between the two to convert the output voltage $V_{mtj,buf}$ back to the original $V_{mtj}$ voltage. The $V_{mtj}$ node voltage is accessed by enabling the MTJ connection switch (S0) in the MSC. Using internal resistors, $V_{bias}$ is swept from 0 to 1 V, and the voltages $V_{mtj}$ and $V_{mtj,buf}$ are measured. This is performed for all four quadrants.

The result of the linear regression to calibrate $V_{mtj}$ is shown in Table 4.4 with equation

$$V_{mtj,Q} = m_Q \cdot V_{mtj,buf,Q} + b_Q \tag{4.8}$$

Table 4.4: $V_{mtj}$ node and output buffer calibration. A linear regression is performed to convert $V_{mtj,buf}$ back to $V_{mtj}$. The slope, intercept, and $R^2$ values are shown here for each quadrant.

| Quadrant, $Q$ | Slope, $m_Q$ | Intercept, $b_Q$ [V] | $R_Q^2$ |
|---|---|---|---|
| 0 | 1.0489 | -1.0375 | 0.999946 |
| 1 | 1.0530 | -1.0380 | 0.999951 |
| 2 | 1.0539 | -1.0440 | 0.999960 |
| 3 | 1.0525 | -1.0511 | 0.999984 |

where $Q$ is the quadrant number. The table shows that the slope is close to unity with a slight gain, the intercept is adjusting for offset in the P-source follower, and, $R^2$ is almost exactly 1 (out to 4 decimal places), which shows that the calibration result is a good fit.

**Linearity plots**

The read circuit is designed to be linear within its operating region. The current conveyor circuit is designed to handle the current required for $V_{bias} = 0.1\,\text{V}$ and $R_{mtj} = 500\,\Omega$ which is $200\,\mu\text{A}$. This is expected to be the most demanding case. Therefore, transistor sizes are chosen to handle this current and operate in the saturation region. We test the linearity of the read circuit by sweeping $V_{bias}$ and measuring directly the $V_{mtj,buf}$ voltage and $I_{out}$ current at DC using a voltmeter and ammeter. The $V_{mtj}$ node voltage is then computed using the calibration equations determined previously. The read circuit operates linearly when $V_{mtj} \approx V_{bias}$.

Linearity plots of the read circuit for internal resistors are shown in Figure 4.13. The nominal resistor values are $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$ with measurements taken for quadrant 0. The top set of subplots show the calculated $V_{mtj}$ node voltage (measured values are indicated by blue $\times$'s) with $V_{bias}$ overlaid as a solid green line for comparison. The bottom set of subplots

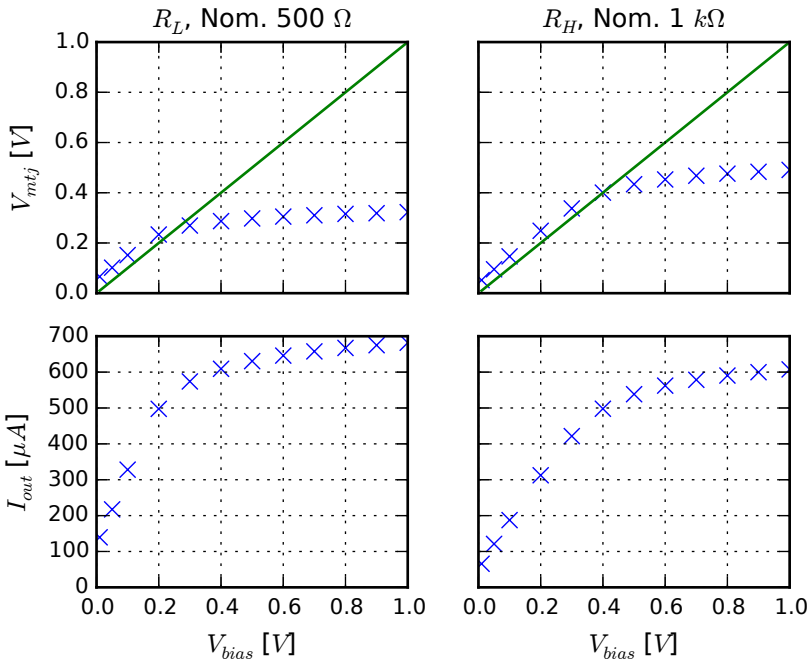



Figure 4.13: Linearity plots of the read circuit for internal resistors with nominal values $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$. These are for quadrant 0. The top plot shows the calculated $V_{mtj}$ node voltage and the bottom plot shows the measured $I_{out}$ current. In the top plot, the solid green line is the ideal $V_{bias}$ voltage which $V_{mtj}$ should correspond to when linear.

show the measured $I_{out}$ current. We can see in the top set of subplots that $V_{mtj}$ is initially linear (following $V_{bias}$), but eventually plateaus. For $R_L$, it becomes nonlinear above about 0.2 V of $V_{bias}$, and for $R_H$, it becomes nonlinear above about 0.4 V of $V_{bias}$. In the bottom set of subplots, we can see that the measured $I_{out}$ current follows the same curve and bend as the $V_{mtj}$ node voltage. For both values of $V_{bias}$ (for each resistor), $I_{out}$ corresponds to about 500 $\mu$A, above which it also becomes nonlinear.

Linearity plots of the read circuit for external resistors are shown in Figure 4.14. The nominal resistor values are 470 Ω, 1 kΩ, and 4.7 kΩ with measured values 459, 996, and 4,611 Ω, respectively. These are again for quadrant 0. We can see in the top set of subplots that $V_{mtj}$ is linear up until about $V_{bias} = 0.2$, 0.4, and 1.2 V, respectively. These correspond in the bottom set of subplots to about $I_{out} = 450$, 400, and 250 $\mu$A, respectively. These maximum

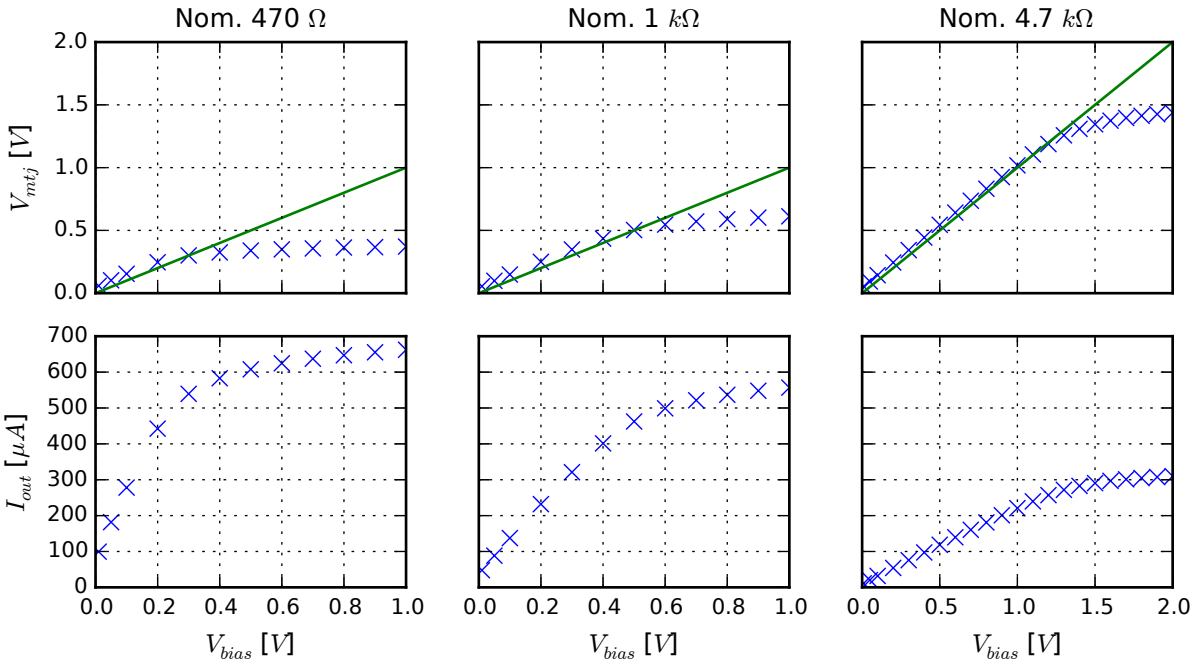



Figure 4.14: Linearity plots of the read circuit for external resistors with nominal values 470 Ω, 1 kΩ, and 4.7 kΩ. These are for quadrant 0. The top plot shows the calculated $V_{mtj}$ node voltage and the bottom plot shows the measured $I_{out}$ current. In the top plot, the solid green line shows the ideal $V_{bias}$ voltage which $V_{mtj}$ should correspond to when linear.

$I_{out}$ currents for linear operation vary between resistor values because the linearity of the read circuit is also a function of $V_{bias}$. $V_{bias}$ and $V_{mtj}$ are connected to the bottom of the current conveyor in the read circuit. Increasing these voltages consequently decreases the operating range of the current conveyor, which further limits $I_{out}$.

A summary of the linear range of $V_{bias}$ of the read circuit for each resistor is shown in Table 4.5. The $V_{bias}$ range given in the table is approximate based on the above plots. This shows that for low $R_{mtj}$ values, the read circuit must operate at low voltage for $V_{bias}$, but, for high $R_{mtj}$ values, it can operate correctly at low or high voltage for $V_{bias}$.

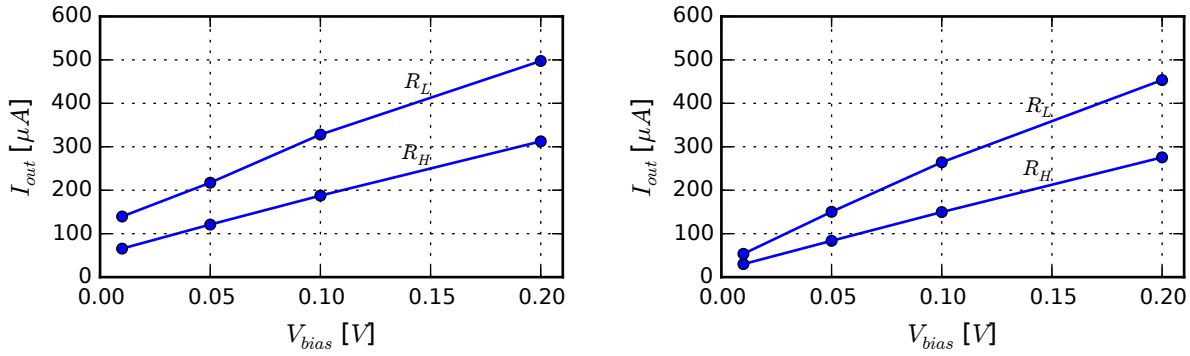
Table 4.5: Summary of the linear range of $V_{bias}$ of the read circuit for each resistor.

| Type | $R_{mtj}$ (nom.) | $V_{bias}$ range (approx.) |
|---|---|---|
| Internal | 500 Ω | 0 to 0.2 V |
| Internal | 1 kΩ | 0 to 0.4 V |
| External | 470 Ω | 0 to 0.2 V |
| External | 1 kΩ | 0 to 0.4 V |
| External | 4.7 kΩ | 0 to 1.2 V |

**Ideal current limits for a theoretical current comparator**

For a theoretical current comparator, we can predict the effective range of $I_{th}$ by plotting the $I_{out}$ current for both $R_L$ and $R_H$ resistors on the same plot. This is shown in Figure 4.15 for internal resistors for all four quadrants of the test chip with nominal values $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$. The $V_{bias}$ range is limited on the plots to the linear range of the read circuit. The data points each represent an experiment to measure the $I_{out}$ current at varying values of $V_{bias}$ and $R_{mtj}$. The data points are connected to form two piecewise lines (one for $R_L$ and the other for $R_H$). They represent the ideal current limits for the theoretical current comparator. The acceptable range of threshold currents is expected to be between these two lines.

We can compare all four quadrants in these plots. First, we can see that they all follow the same trend. They are each linear and increasing. And second, we can see that they are similar in value, although they do vary slightly. At low $V_{bias}$, the $I_{out}$ current can vary quite significantly, but at high $V_{bias}$, it varies much less. This is primarily due to differences in offset of the $V_{mtj}$ node voltage for each quadrant. At $V_{bias} = 50\,\text{mV}$, the $V_{mtj}$ offset is measured for $R_L$ for each quadrant to be 0.051, 0.021, 0.046, and 0.036 V, respectively. For $R_H$, the $V_{mtj}$ offset is measured for each quadrant to be 0.045, 0.017, 0.041, and 0.029 V, respectively.




(a) Quadrant 0

(b) Quadrant 1

(c) Quadrant 2

(d) Quadrant 3

Figure 4.15: Ideal current limits for a theoretical current comparator. These plots show all four quadrants using internal resistors with nominal values $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$ in the linear range of $V_{bias}$ (from 0 to 0.2 V).

Variation in these offsets account for significant deviation in $I_{out}$ between quadrants for low $V_{bias}$, but little deviation in $I_{out}$ at high $V_{bias}$.

### Current comparator threshold testing

We now test the physical current comparator in the read circuit using static measurements to determine the operating range of the threshold current, $I_{th}$, at DC. This is done by first selecting a value of $V_{bias}$ and $R_{mtj}$ to test with. This produces an output current (from the current conveyor) that is supplied to the top branch of the current comparator. The

Table 4.6: Measured operating ranges of the current comparator threshold current. These are for internal resistors in quadrant 0 with nominal resistance values $R_L = 500\,\Omega$ and $R_H = 1\,\mathrm{k}\Omega$.

| $V_{bias}$ [V] | $I_{RH,out}$ [$\mu$A] | $I_{th,L}$ [$\mu$A] | $I_{th,H}$ [$\mu$A] | $I_{RL,out}$ [$\mu$A] |
|---|---|---|---|---|
| 0.1 | 187.6 | 196.0 | 334.0 | 328.1 |
| 0.2 | 312.5 | 334.0 | 503.0 | 497.5 |

threshold current, which is controlled in our test setup, is supplied to the bottom branch of the current comparator. Then, $I_{th}$ is swept from low to high to determine the switching point of the current comparator to produce a valid, stable, logic output (through the output buffer). The result of this testing is summarized in Table 4.6.

The table shows the operating ranges of the current comparator threshold current measured at DC that gives stable output. These measurements are for internal resistors in quadrant 0. The output currents, $I_{RH,out}$ and $I_{RL,out}$, are reproduced in the table for both resistances, which represent the ideal current limits. The operating range of threshold currents are given by $I_{th,L}$ and $I_{th,H}$. Outside this range where $I_{th}$ is nearly equal to $I_{out}$, the current comparator output (through the output buffer) may become unstable and produce oscillations.

The results show that the $I_{th,L}$ and $I_{th,H}$ currents are near in value to the $I_{RH,out}$ and $I_{RL,out}$ currents, respectively. There is great overlap between the two ranges, however, the operating range of threshold currents is not a subset of the ideal current limits as might be expected. $I_{th,H}$ is greater than the upper current limit $I_{RL,out}$, which maybe due to an offset in the current comparator.

### 4.3.3 Stability and dynamic measurements

In this section, we will perform dynamic tests of the read circuit by stimulating it with a square wave CLKIN signal and then observing the stability of the read circuit to determine the stable region of $I_{th}$ currents that it can operate within. Note, these dynamic tests will give a stable region of $I_{th}$ currents that is different from those in the static measurement tests. The dynamic tests will capture the dynamic behavior of the read circuit.

Dynamic tests of the read circuit were performed for both internal and external resistors with nominal resistance values $R_L = 500\,\Omega$ and $R_H = 1\,\mathrm{k}\Omega$ for all four quadrants on the test chip. To do this, CLKIN is set to a low frequency, and $I_{th}$ is swept from low to high current. The first (low) current to produce a correct, stable MCLK output (no oscillations or glitching) is recorded as $I_{th,L}$. The last (high) current to produce a correct, stable MCLK output is recorded as $I_{th,H}$. These currents form the stability region of $I_{th}$ and are shown in Table 4.7 for each experiment. For these measurements, $V_{bias} = 0.1\,\mathrm{V}$. Also reproduced in the table are the output currents, $I_{RH,out}$ and $I_{RL,out}$, which are the ideal current limits of $I_{th}$. Outside of the stability region, the output shows glitching or oscillations on the MCLK output waveform at the edge where $I_{th}$ is near one of the output current limits.

We can make a number of observations in these results. One, the $I_{RH,out}$ and $I_{RL,out}$ currents vary quite significantly between quadrants, which is primarily due to variation in the $V_{mtj}$ node voltage. Two, stable ranges are found for internal resistors for all quadrants. Three, the stable threshold ranges of the internal resistors vary quite significant between quadrants. There is not a clear common threshold (around the middle of the ranges) that could be set among them all. Rather, each quadrant needs its own threshold current. This is partly due to the variation in the offset of the current comparator in each quadrant which has a significant effect on the output current at low $V_{bias}$. Four, the stable threshold ranges

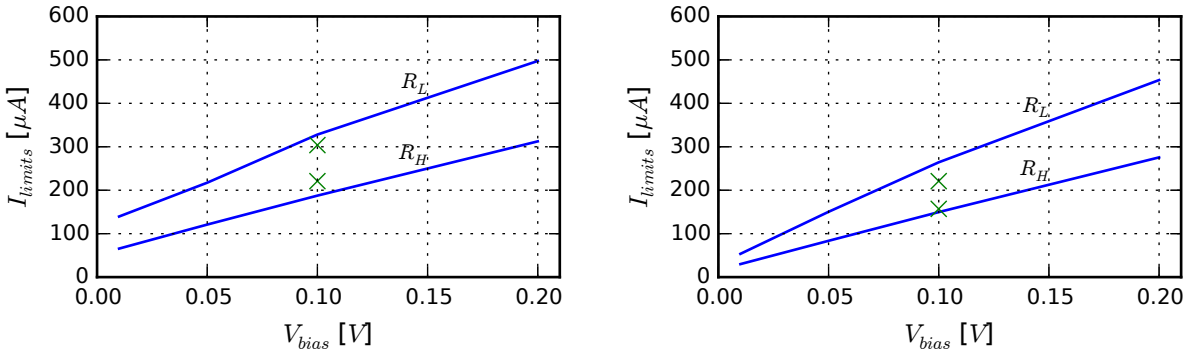
Table 4.7: Dynamic stability ranges. For these measurements, $V_{bias} = 0.1\,\mathrm{V}$, and the resistors (both internal and external) have nominal values $R_L = 500\,\Omega$ and $R_H = 1\,\mathrm{k}\Omega$. Testing includes both internal and external resistors for all four quadrants.

| Type | Quadrant | $I_{RH,out}$ [$\mu$A] | $I_{th,L}$ [$\mu$A] | $I_{th,H}$ [$\mu$A] | $I_{RL,out}$ [$\mu$A] |
|---|---|---|---|---|---|
| Internal | 0 | 187.6 | 221.0 | 304.0 | 328.1 |
| Internal | 1 | 149.7 | 157.0 | 221.0 | 264.3 |
| Internal | 2 | 182.0 | 202.0 | 258.0 | 314.4 |
| Internal | 3 | 170.8 | 195.0 | 275.0 | 302.2 |
| External | 0 | 137.5 | 235.0 | 240.0 | 258.8 |
| External | 1 | 108.9 | 105.0 | 122.0 | 209.2 |
| External | 2 | 133.8 | - | - | 249.4 |
| External | 3 | 125.0 | - | - | 239.9 |

of the internal resistors are pretty wide, making it easier to choose a threshold for correct operation. And five, the stable threshold ranges of the external resistors are pretty narrow in quadrants 0 and 1 and nonexistent in quadrants 2 and 3. This may in part be due to extra wire capacitance and resistance of the connections to the external resistors.

Overlaying the stable threshold ranges with the ideal current limit plots, we get the plots shown in Figure 4.16. These show all four quadrants and use internal resistors. Here, the ideal current limits are drawn as solid blue lines. These are simply the $I_{RL,out}$ and $I_{RH,out}$ currents plotted on the graph for $R_L$ and $R_H$ resistors, respectively. And, the measured current limits, $I_{th,L}$ and $I_{th,H}$, are drawn as green $\times$'s. We can see that the measured current limits remain between the ideal current limits. This quantifies the range of current threshold values over which the circuit operates as intended.

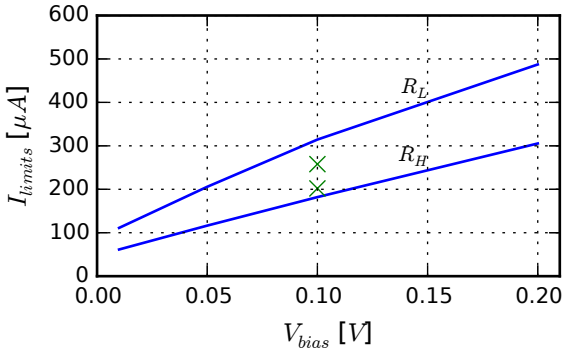
Next, we can drill down further to investigate what the MCLK output waveform looks like in the stable and unstable regions. To do this, we select three values of $I_{th}$. The first is below the low $I_{th,L}$ current, but above $I_{RH,out}$. The second is set midway between $I_{th,L}$ and $I_{th,H}$.



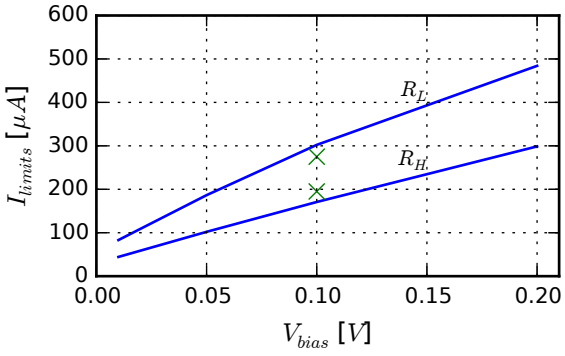

(a) Quadrant 0




(b) Quadrant 1




(c) Quadrant 2




(d) Quadrant 3

Figure 4.16: Dynamic stability plots showing the stable region of $I_{th}$ currents. These plots show all four quadrants using internal resistors with nominal values $R_L = 500\,\Omega$ and $R_H = 1\,\text{k}\Omega$. The lines are the ideal current limits for a theoretical current comparator. The green $\times$'s are the measured current limits of the stable range for the actual current comparator in the dynamic test.

And the third is above the high $I_{th,H}$ current, but below $I_{RL,out}$. We zoom in on the rising and falling edges to observe the oscillations that occur in the unstable waveforms. These waveforms, me

asured using an oscilloscope at the MCLK output, are shown in Figure 4.17. The rising and falling edges of MCLK are aligned relative to the falling and rising edges of CLKIN, respectively.

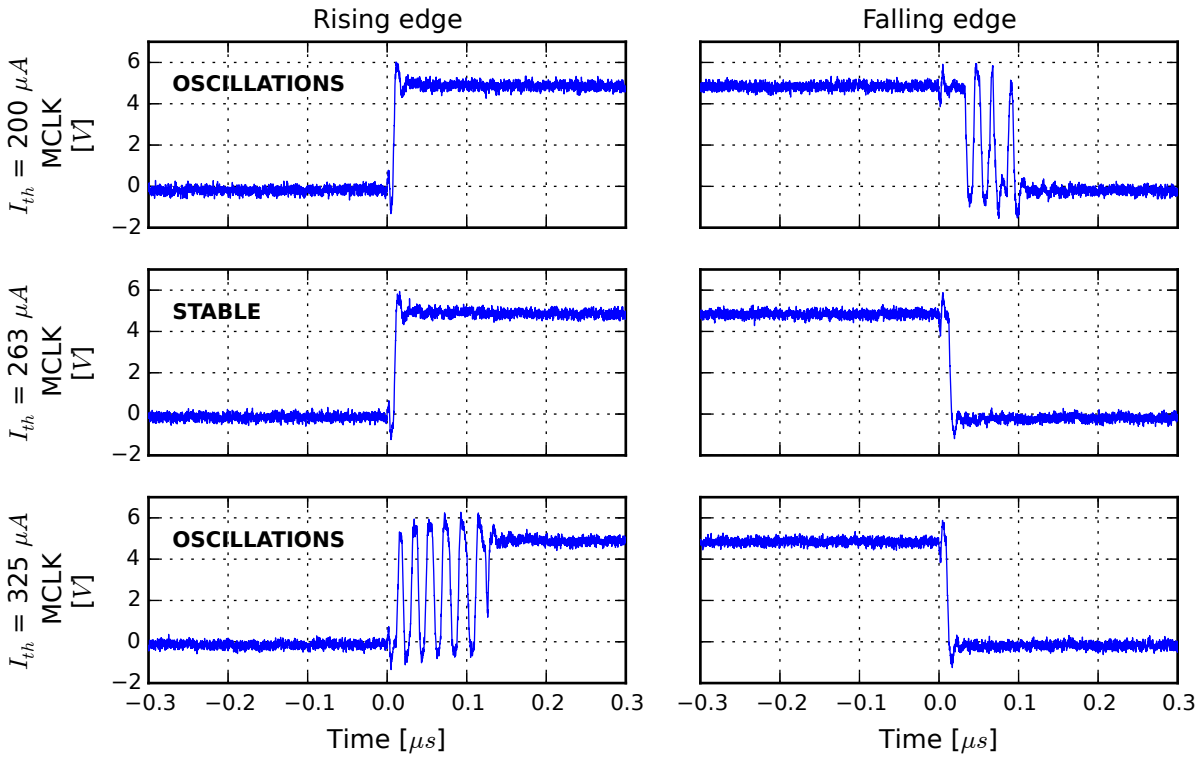



Figure 4.17: Output waveforms demonstrating oscillations and stable operation for multiple $I_{th}$ currents at the rising/falling edges. This is using internal resistors in quadrant 0 with nominal resistance values $R_L = 500\,\Omega$ and $R_H = 1\,\mathrm{k}\Omega$. $V_{bias} = 0.1\,\mathrm{V}$.

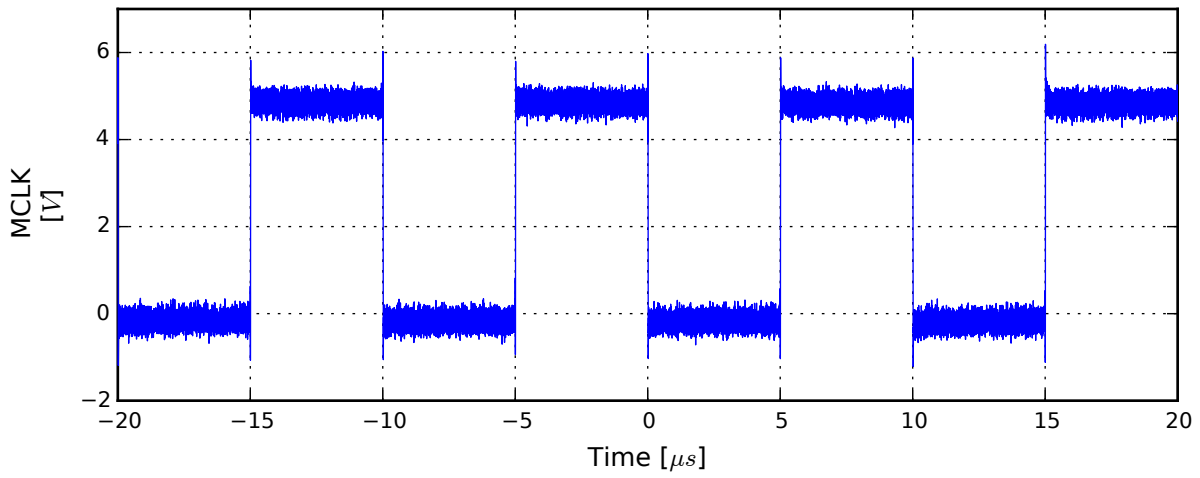



Figure 4.18: Output waveform demonstrating stable operation at midway $I_{th} = 263\,\mu\mathrm{A}$. This is using internal resistors in quadrant 0 with nominal resistance values $R_L = 500\,\Omega$ and $R_H = 1\,\mathrm{k}\Omega$. $V_{bias} = 0.1\,\mathrm{V}$.

In the figure, the three values of $I_{th}$ chosen are 200, 263, and 325 $\mu$A. Internal resistors are used in quadrant 0 of the test chip with $V_{bias} = 0.1$ V. At $I_{th} = 200\,\mu$A, the threshold is set low. This causes oscillations when $I_{out}$ falls low to $I_{RH,out}$ which occurs on the falling edge of MCLK. This is observed in the plot. At $I_{th} = 325\,\mu$A, the threshold is set high. This again causes oscillations when $I_{out}$ rises high to $I_{RL,out}$ which occurs on the rising edge of MCLK. For $I_{th} = 263\,\mu$A, which is the midway threshold point, MCLK is stable on both edges. The full waveform for this threshold current is shown in Figure 4.18. We can see that there is no glitching or oscillations and that the read circuit is working as intended.

The current comparator in the read circuit is designed using a current source and current sink that pull against each other. A buffer (which is two series inverters) thresholds the internal voltage of the current comparator to produce the digital MCLK output. There is no hysteresis built into this circuit design. Therefore, the circuit output is sensitive to the effects of noise in the current comparator which can cause the output buffer to flip multiple times if the internal voltage of the current comparator is near the threshold of the output buffer. Redesigning the current comparator with hysteresis may help alleviate this problem and extend the stable region of threshold currents in the read circuit. This may be particularly helpful for external resistors in cases where no stable region could be found.

## 4.4 Summary

We were able to test the resistance-to-voltage read circuit using our fabricated prototype test chip. The read circuit was shown to operate correctly. In the results, we see that $V_{mtj}$ follows $V_{bias}$, but with an offset. That offset accounts for a higher than expected output current. The offset also varies between quadrants on the chip. Testing the read circuit for performance, we found that it can operate as fast as 48 MHz using internal resistors in

quadrant 0 of the test chip (which was fabricated in a 3M2P 0.5 $\mu$m process). Above this frequency of switching, the output waveform is unstable.

We measured the propagation delay of the read circuit using the internal resistors to be about 12 ns. This is on par with the simulation results at about 7 ns. We further measured the average power consumed by the read circuit to be 3.70 mW. This also is on par with the simulation results at about 2.59 mW. It is higher because the output currents on the test chip were higher than expected.

We estimated empirically the internal resistor values and found that their measured values are lower than nominal, within $466 \pm 6\,\Omega$ and $787 \pm 11\,\Omega$. This is one factor that accounts for the higher than expected output currents on the chip.

Last, we found a range of $I_{th}$ over which the read circuit operates correctly to produce a good stable waveform output. Outside of this range, $I_{th}$ approaches one of the current limits and causes the output of the current comparator to oscillate.

# Chapter 5

# Virtualization of Hardware Logic Circuits

Virtualization of computational resources provides a way by which hardware resources can be reused, which is commonly known as resource sharing. In magnetologic technology, an MTJ device, which is used to construct a magnetologic gate, acts as a latch to whatever data is written to it. This gives it the potential to act as state in the pipeline stage of a logic circuit. We want to exploit the latching property of these fundamental magnetologic circuits for constructing deeply-pipelined circuits, using virtualization as a means of effectively utilizing the hardware logic. Sharing is a common technique used for utilizing available hardware resources for computing such as DSP blocks [54], memory controllers, memory bandwidth, and IP cores.

In this chapter, we model the performance of a virtualized fixed logic computation and tune a schedule for optimal performance. Our interest is in supporting a set of distinct data streams that all wish to perform the same computation. Virtualizing the logic computation involves sharing hardware resources and context switching each distinct data stream into the hardware. This context switch can be either fine-grained (in which context is changed

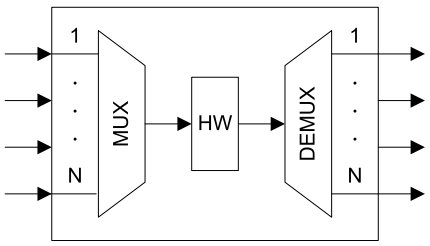



Figure 5.1: Hardware virtualization for $N$ distinct data streams that perform the same computation. The $N$ streams are multiplexed into a shared hardware (HW) block, processed, and then demultiplexed back into $N$ streams.

each clock cycle) or coarse-grained (in which the state of the computation is swapped out to a secondary memory). Good candidates for virtualization are logic computations with long combinational paths (i.e., substantial computational requirements). We will approach virtualization from this perspective of pipelining combinational logic, however, for magnetologic, the computation is naturally pipelined and the pipeline depth is fixed.

To virtualize a function, consider the hardware block (HW) shown in Figure 5.1 that has an Nx1 input multiplexer and 1xN output demultiplexer. This HW block is a function implemented in custom logic. $N$ distinct data streams (each with a dedicated input and output port) share the single instance of the HW block. These data streams are then multiplexed into the custom logic block, processed, and then demultiplexed back into independent streams. For our purposes, we are not considering I/O bound computations, but rather assume there is sufficient bandwidth at the input and output ports of Figure 5.1. When the logic function is purely combinational (i.e., feed-forward), any input from any data stream can be presented to the HW block at any clock cycle, even if it is deeply-pipelined. In this case, there are no constraints on scheduling. When the logic function is sequential (i.e., has feedback) and has been deeply-pipelined, this imposes scheduling constraints. Once a data element from a particular stream has been delivered to the HW block, the stream has to wait a number of clock ticks equal to the pipeline depth before it can provide a subsequent data element from that same stream.

Pipelined logic circuits with feedback can be context switched to compute multiple data streams concurrently. Essentially, the circuit can be thought of as a sequential logic circuit with pipelined combinational logic. The pipelined combinational logic adds latency and decreases single stream throughput since it takes multiple clock cycles (corresponding to the number of pipeline stages) to compute a single result and feed it back to the input. If the number of pipeline stages is $C$, then this circuit is said to be $C$-slowed since a single computation takes $C$ times more clock cycles (often mitigated by a higher clock *rate*). $C$-slow is a technique described by Leiserson and Saxe [21] by which each register is replaced by $C$ registers and then retimed to balance the registers throughout the combinational logic. Exploiting this characteristic allows processing multiple different contexts or data streams in a fine-grain way using the same hardware logic. The number of fine-grain contexts supported equals the pipeline depth.

When the number of contexts to be supported, $N$, is greater than the pipeline depth, $C$, coarse-grained context switching can be used, swapping out whatever state is stored in the circuit to a secondary memory. In general, this will incur some cost, representing the overhead of a context switch. While the fine-grained context switching of the $C$-slowed circuit naturally uses a round-robin schedule, there are a richer set of scheduling choices available when building a coarse-grain context switched design. In this work, we constrain our consideration to round-robin schedules and explore the performance impact of the schedule period.

The general virtualized hardware configuration that we consider is shown in Figure 5.2a. An arbitrary sequential logic circuit (i.e., the HW block) with input $x$, state $y$, and output $z$ has been $C$-slowed and augmented with a secondary memory that can load and unload copies of state $y$ to/from the "active" state register. $N$ FIFO, or First In, First Out, buffers are present at the inputs to store data stream elements that are awaiting being scheduled. The

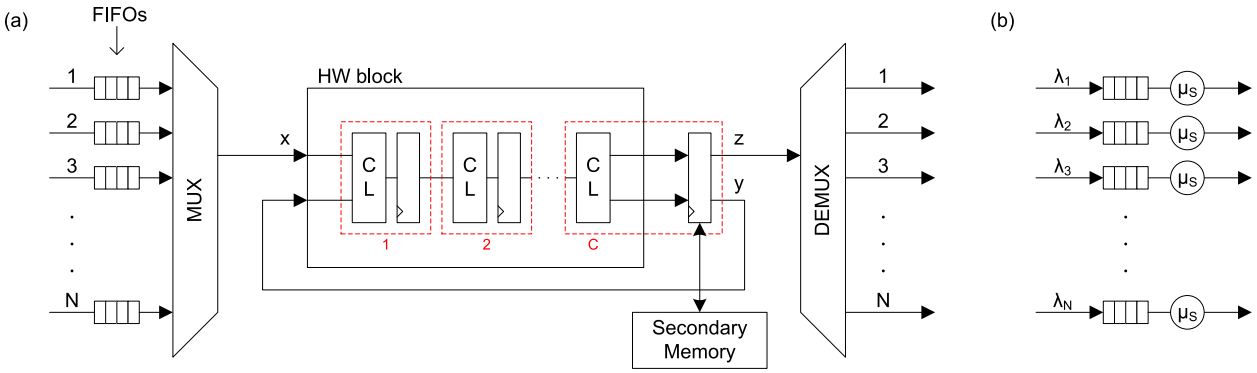



Figure 5.2: (a) General virtualized hardware configuration with a $C$-slowed sequential logic circuit and secondary memory supporting $N$ data input/output streams. (b) Queueing model of this circuit with $N$ queueing stations (one for each data stream). Each queueing station consists of a FIFO queue and associated server representing one virtual copy of the hardware computation.

circuit consumes one data element (from an individual input specified by the schedule) each clock tick.

A queueing model of this circuit can be developed to predict system performance. Inputs to the model and model assumptions are described first. The number of input data streams is denoted by $N$. Each data stream, $i$, is assumed to provide elements with a known distribution and given mean arrival rate $\lambda_i$ $^{\text{elements}}/_{\text{s}}$. We will assume the input distribution is Poisson. The HW block logic is characterized as follows. The total combinational propagation delay is given as $t_{CL}$. The pipelining depth is $C$ (corresponding to a $C$-slowed design). We model the combinational propagation delay between the pipeline registers as a random variable $X$ with mean $\mu_X = t_{CL}/C$ and standard deviation $\sigma_X$. The clock period, $t_{CLK}$, is predicted by the expectation of the maximum of the stage-to-stage delay. We assume that both the secondary memory and input buffers operate at the clock rate of the pipeline. State transfers to/from secondary memory take a given $S$ clock cycles (enabling the model to support a range of context switch overheads), and the buffers are assumed to have single-cycle enqueue and dequeue capability (i.e, they do not limit the performance of the system).

With the above inputs available, we represent the performance of the context switched hardware via an open queueing network model with effective service rates determined by the clock frequency achievable by the $C$-slowed circuit. This queueing model is illustrated in Figure 5.2b. Each individual queueing station represents one virtual copy of the hardware computation. The arrival rate at each queue is $\lambda_i$ $^{\text{elements}}/_{\text{s}}$.

In the sections that follow, we investigate and model the clock period of a $C$-slowed circuit which will be used in the queueing model. Second, we develop queueing model equations to predict system performance. Third, we measure empirical data for three $C$-slowed applications on two technologies and calibrate the clock period model and a resource model to the data. Last, we present analytic performance model prediction results for three design scenarios, showing different ways in which to apply the model.

## 5.1 Clock model

The clock period will be modeled for a $C$-slowed circuit. A $C$-slowed circuit, shown in Figure 5.3, consists of a sequential logic circuit (i.e., a circuit with combinational logic and a feedback path) that is pipelined with $C$ pipeline stages. The pipeline stages are to be distributed evenly throughout the combinational logic and have a stage-to-stage delay modeled as a random variable. If $t_{CL}$ is the total combinational logic delay for a non-pipelined circuit, then the mean stage-to-stage combinational logic delay will be $t_{CL}/C$. There is variation in the stage-to-stage delay due to the placement of logic and the routing of signals between logic, which we will model stochastically. The clock period, $t_{CLK}$, is then determined by the worst-case logic path, which is the maximum stage-to-stage delay in the $C$-slowed circuit. Therefore, we can model $t_{CLK}$ as the expectation of the maximum of $C$ samples of a random variable $X$ where $X$ is the stage-to-stage delay.

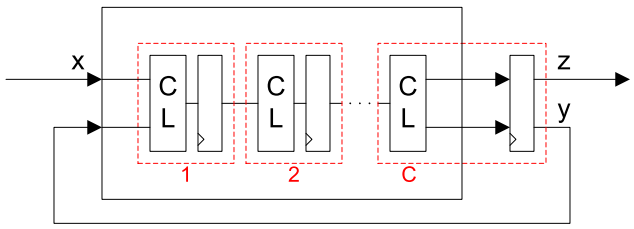



Figure 5.3: $C$-slowed circuit.

In the clock period model development that will follow, we start by formulating an initial model for the expectation of the maximum of $X$ by first using order statistics to establish an upper bound on $X$ based on a square root function. This bound is a general result and gives us a starting place for developing a model. Empirical simulation will show that this model does not match the shape of the expectation of the max very well, particularly at a low value of $C$. This can be improved, however, by adding a power term to bend the shape of the model curve. Next, we assume that $X$ is a normal (or Gaussian) distribution and establish a tighter bound on the expectation of the max based on the square root of a logarithmic function. We again add a power term to allow bending the shape of the model curve and remove the square root (which is subsumed in the power term). This model is shown to better match the shape of the expectation of the max and is used to build the $t_{CLK}$ model. The $t_{CLK}$ model is then evaluated with various random distributions to ensure that it is a good curve-fit for each distribution. Last, a study is performed on $t_{CL}$ and $\sigma$ for each random distribution to determine their effect on the fitted power term in the model. It is found that the power term is distribution dependent and, for some distributions, constant.

### 5.1.1 Model development of the expectation of the maximum of $C$ samples of random variable $X$

In the model development of the expectation of the max, we model the stage-to-stage combinational logic delay for $C$ pipeline stages in Figure 5.4. Here, there are $C$ random samples, $x_1, x_2, \ldots, x_C$, one for each stage-to-stage delay drawn from a distribution defined by random variable $X$ with mean $\mu_X$ and standard deviation $\sigma_X$. The maximum delay is then $\max(x_1, x_2, \ldots, x_C)$, which corresponds to the critical path.

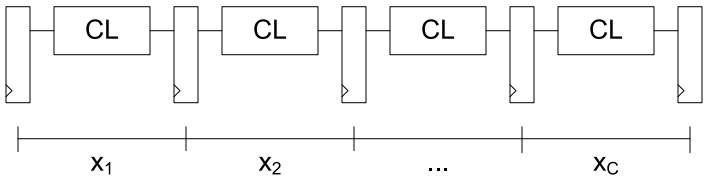



Figure 5.4: Stage-to-stage delay model. Delays are modeled with random variable $X$.

#### Using bound equation from order statistics

Using order statistics [55], we can establish a bound on the expectation of the maximum of $C$ samples from $X$ as

$$E[\max(X)] \leq \mu_X + \sigma_X\sqrt{C-1}. \tag{5.1}$$

Note, this bound is general for any probability distribution of $X$ and does not assume independence or identical distribution of samples.

To see how this bound equation compares to the estimated expectation of the maximum of $X$, we estimate the expectation of the max using a Monte Carlo simulation. In the simulation, $X$ is generated from a normal distribution, $\mathcal{N}(10, 5^2)$, across 10,000 ensembles (where $\mu_X = 10$ and $\sigma_X = 5$). For each ensemble, $\max(X)$ is calculated and then averaged across all ensembles to estimate $E[\max(X)]$. The estimated $E[\max(X)]$ and the bound

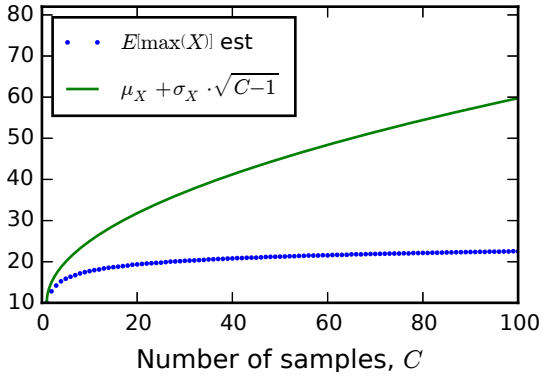



Figure 5.5: Empirical simulation of $E\left[\max\left(X\right)\right]$ (blue) and its upper bound (green) for $C$ random samples. $X$ is generated from a normal distribution, $\mathcal{N}(10, 5^2)$. For each value of $C$, $E\left[\max\left(X\right)\right]$ is calculated from 10,000 ensembles.

equation is shown in Figure 5.5. The simulation shows that the bound curve trends with $E\left[\max\left(X\right)\right]$.

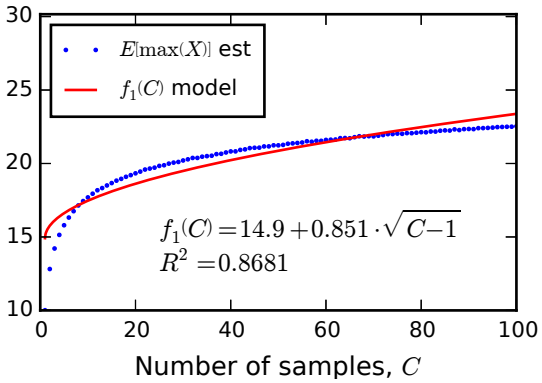
Using the bound in (5.1) as a guide, the expectation of the maximum of $X$ can be modeled (Model 1) as

$$f_1\left(C\right) = k_1 + k_2\sqrt{C-1} \tag{5.2}$$

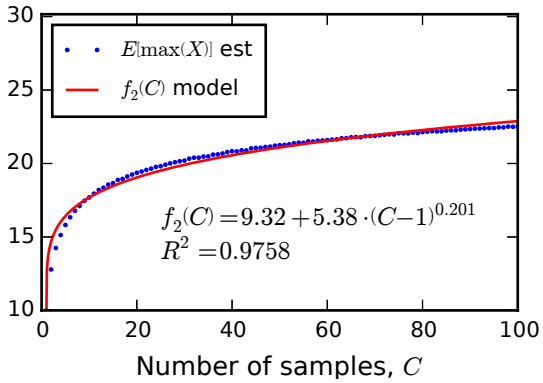
where $k_1$ and $k_2$ are curve-fit parameters of the model. Note, we can observe that when $C = 1$ in this model, then $k_2\sqrt{C-1}$ will be 0, and only $k_1$ will be returned. Therefore $k_1$ should curve fit to approximately $\mu_X$.

In Figure 5.6a, we curve fit the model in (5.2) to the simulated $E\left[\max\left(X\right)\right]$. The resulting plot shows that the model does not match the shape of $E\left[\max\left(X\right)\right]$ very well and has an $R^2$ value of only 0.8681. Also, $k_1 = 14.9$ which overestimates $\mu_X = 10$. We observe that the shape of the model curve needs to be bent down to match $E\left[\max\left(X\right)\right]$. This can be done by replacing the square root term with an arbitrary power $p$ that can bend the curve as needed. This gives the following revised model equation (Model 2):

$$f_2\left(C\right) = k_1 + k_2\left(C-1\right)^p. \tag{5.3}$$




(a) Model 1: $f_1(C) = k_1 + k_2\sqrt{C-1}$




(b) Model 2: $f_2(C) = k_1 + k_2(C-1)^p$

Figure 5.6: Curve-fit of Model 1 and 2 with samples drawn from normal distribution $\mathcal{N}(10, 5^2)$.

In Figure 5.6b, we now curve fit the revised model in (5.3) to the simulated $E[\max(X)]$. This curve-fit is much better with an $R^2$ value of 0.9758. The power, $p$, determined by the curve-fit regression is 0.201 which is approximately $1/5$, and $k_1 = 9.32$ which is closer to $\mu_X = 10$, making it a better estimator than the previous model. However, the shape of the curve is still off, particularly at low values of $C$.

### Using bound equation for a Gaussian distribution

If we assume that the distribution is normal (or Gaussian), then we can compute a tighter bound on the expectation of the maximum of random variable $X$ as

$$E[\max(X)] \leq \mu_X + \sigma_X\sqrt{2 \cdot \ln(C)}. \tag{5.4}$$

Note, this bound is based on the square root of a natural logarithmic function of $C$ instead of square root of $C-1$ which will give a different shape for the model curve.

To derive the bound equation, we can apply Jensen's inequality [56]. If we define $Z = \max_{1 \leq i \leq C}(X_i)$ where $X_i$ are $C$ i.i.d. random variables of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, then we can apply Jensen's inequality as follows:

$$e^{t \cdot E[Z]} \leq E\left[e^{t \cdot Z}\right] \tag{5.5}$$

$$= E\left[\max\left(e^{t \cdot X_i}\right)\right]. \tag{5.6}$$

Using the union bound [57], we can bound the maximum as a sum:

$$e^{t \cdot E[Z]} \leq \sum_{i=1}^{C} E\left[e^{t \cdot X_i}\right]. \tag{5.7}$$

Last, this summation and expectation can be evaluated from the moment generating function of a Gaussian as $C \cdot e^{t \cdot \mu + t^2 \sigma^2 / 2}$. This simplifies to

$$E[Z] \leq \mu + \frac{\ln(C)}{t} + \frac{t \cdot \sigma^2}{2}. \tag{5.8}$$

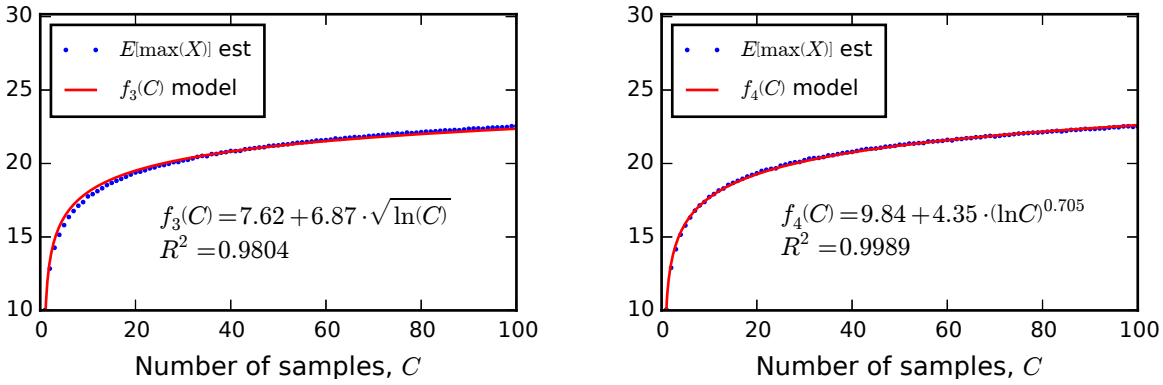
We can further minimize the expression by setting $t = \frac{\sqrt{2 \cdot \ln(C)}}{\sigma}$, which is computed by calculating the derivative and setting it equal to 0. The simplified expression then becomes:

$$E[Z] \leq \mu + \sigma\sqrt{2 \cdot \ln(C)}. \tag{5.9}$$

Replacing the constants with variables $k_1$ and $k_2$, we get the following model equation (Model 3):

$$f_3(C) = k_1 + k_2\sqrt{\ln(C)}. \tag{5.10}$$

Note, as was the case in (5.2), when $C = 1$, then $k_2\sqrt{\ln(C)}$ will be 0, and only $k_1$ will be returned. $k_1$ will continue to curve fit to $\mu_X$.




(a) Model 3: $f_3(C) = k_1 + k_2\sqrt{\ln(C)}$ (b) Model 4: $f_4(C) = k_1 + k_2(\ln C)^p$

Figure 5.7: Curve-fit of Model 3 and 4 with samples drawn from normal distribution $\mathcal{N}(10, 5^2)$.

In Figure 5.7a, we curve fit the model in (5.10) to the simulated $E[\max(X)]$. Compared to Figure 5.6b, the model curve more closely matches the data and $R^2$ improves from 0.9758 to 0.9804. However, visually, we can still see there is room for improvement. The model curve raises too quickly and flattens more than it should. This indicates a similar problem as with the $\sqrt{C-1}$ model in (5.2) in that the model needs to be bent or reshaped. Therefore, we can accomplish this by replacing the square root in (5.10) with an arbitrary power $p$. This results in the following revised model (Model 4):

$$f_4(C) = k_1 + k_2 \cdot (\ln C)^p . \tag{5.11}$$

In Figure 5.7b, we curve fit the revised model in (5.11) to the simulated $E[\max(X)]$. Here, the $p$ parameter reshapes the curve so that the model curve more closely matches the data for all values of $C$. The resulting $R^2$ is 0.9989. The optimal $p$ value turned out to be 0.705 ($\approx \sqrt{1/2}$) instead of 1/2 (for a square root).

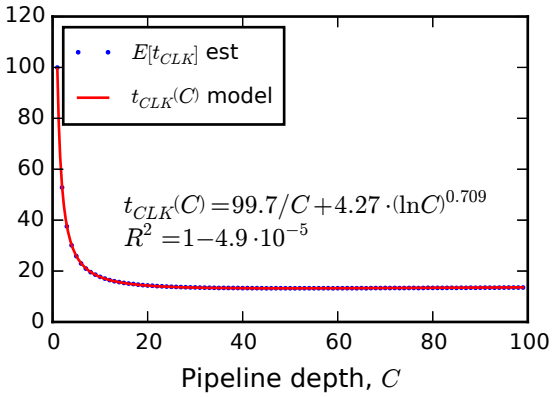



Figure 5.8: Curve-fit of $t_{CLK}(C)$ model with samples drawn from a normal distribution ($t_{CL} = 100$, $\sigma = 5$).

### 5.1.2 Model of $t_{CLK}$ and investigation of model

Using Model 4 in (5.11), we can now model the clock period, $t_{CLK}$, as

$$t_{CLK}(C) = \frac{k_1}{C} + k_2 \cdot (\ln C)^p, \tag{5.12}$$

replacing $k_1$ with $k_1/C$. Here, $C$ is the number of pipeline stages, and $k_1$, $k_2$, and $p$ are curve-fit parameters of the model. $k_1$ represents the curve-fit total combinational logic delay of the circuit (which should be approximately equal to $t_{CL}$), so that as $C$ increases, $k_1/C$ decreases. $k_1/C$ is the mean stage-to-stage delay for combinational logic that is evenly pipelined with $C$ pipeline stages.

In Figure 5.8, we curve-fit the $t_{CLK}$ model in (5.12) to the simulated $t_{CLK}$. In the simulation, 10,000 ensembles were generated to compute the expectation of the maximum of the delay for $C$ pipeline stages with $t_{CL} = 100$ and $\sigma = 5$. This plot shows that the model matches the simulation data with an $R^2$ value of $1 - 4.9 \cdot 10^{-5}$. The fitted $p$ value is 0.709 which is similar to that in Figure 5.7b. We can also observe that $k_1$ approximately estimates $t_{CL} = 100$.

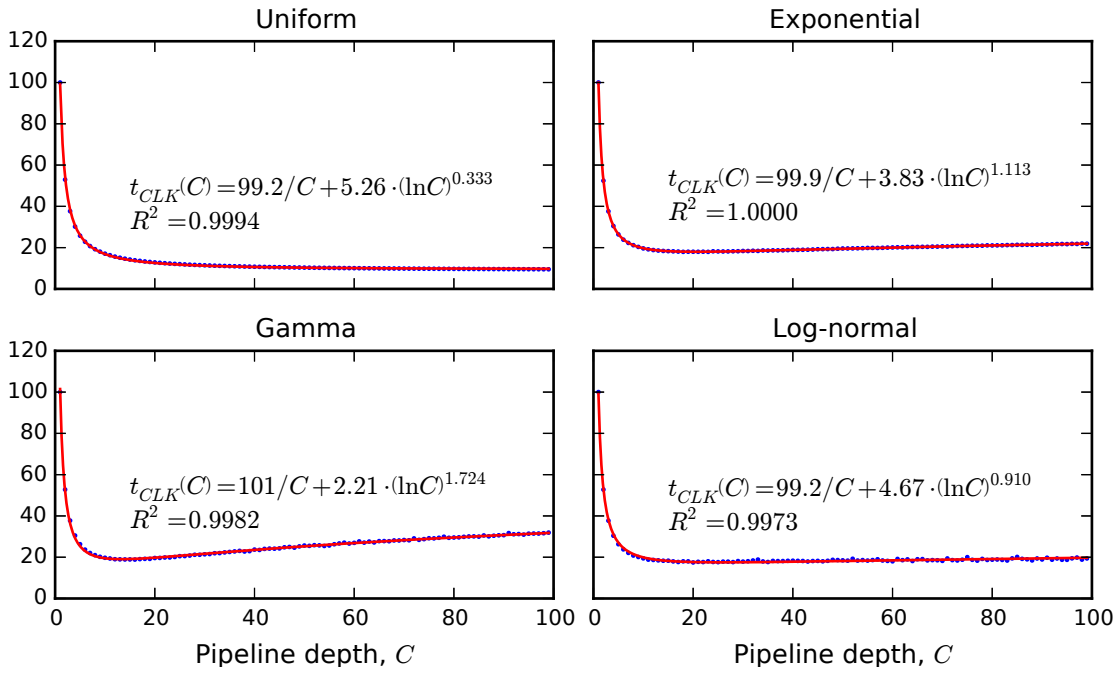



Figure 5.9: Curve-fit of $t_{CLK}(C)$ model with samples drawn from various distributions ($t_{CL} = 100$, $\sigma = 5$).

So far, the model has been tested only for a Gaussian distribution. Our next question is, if we vary the distribution, will the model still fit the empirical simulation results? To determine this, we simulate the the stage-to-stage delays using uniform, exponential, gamma, and log-normal distributions. For each distribution, $t_{CL} = 100$ and $\sigma = 5$. The simulation results and corresponding curve-fits to the $t_{CLK}$ model are shown in Figure 5.9. These plots show that the shape of the curve changes based on the distribution. The $p$ parameter, as expected, changes to match the shape of the curves accordingly. All curve-fits match the data well with $R^2$ values $\geq 0.9973$.

Last, we investigate the effect of varying $t_{CL}$ and $\sigma$ on the curve-fit parameter $p$. To do this, we simulate each distribution again with 3 different sets of values for $t_{CL}$ and $\sigma$, and report the fitted $p$ parameter. The experimental results are shown in Table 5.1. We can observe that $p$ is distribution dependent and, for some distributions, constant. For normal

Table 5.1: Curve-fit $p$ values of the $t_{CLK}(C)$ model for various distributions and parameters.

| | **Parameters** | | |
|---|---|---|---|
| | $t_{CL} = 10$ $\sigma = 1$ | $t_{CL} = 100$ $\sigma = 1$ | $t_{CL} = 100$ $\sigma = 10$ |
| **Distribution** | **Curve-fit $p$ values** | | |
| Normal | 0.7096 | 0.7115 | 0.7108 |
| Uniform | 0.3339 | 0.3326 | 0.3332 |
| Exponential | 1.1185 | 1.1132 | 1.1172 |
| Gamma | 1.0782 | 2.0024 | 1.0782 |
| Log-normal | 0.5557 | 2.0696 | 0.5454 |

(Gaussian), uniform, and exponential distributions, $p$ is approximately constant. However, for gamma and log-normal distributions, it is approximately constant only when $t_{CL}/\sigma$ is constant.

We anticipate that the distribution will likely be a Gaussian distribution, and that we can use a constant $p$ value of approximately 0.7 for modeling $t_{CLK}$. This gives the following modified equation that will be used in Section 5.4 as the basis for calibrating $t_{CLK}$:

$$t_{CLK}(C) = \frac{k_1}{C} + k_2 \cdot (\ln C)^{0.7}. \tag{5.13}$$

## 5.2 M/D/1 queueing model development

In this section, we will develop an M/D/1 queueing model of the general virtualized hardware configuration ($C$-slow + secondary memory) of Figure 5.2(a) to predict system performance. Outputs of the model include total achievable throughput, latency, and queueing occupancy.

### 5.2.1 Queueing model

To model the effective service rate, we first consider the case where $N = C$ (i.e., only fine-grained context switching is employed, and there is no secondary memory). This uses a fixed, round-robin schedule (since there are no context switches to secondary memory) with data elements dequeued from each input at a fixed rate of $\mu_s = f_{CLK}/C$. This corresponds to a deterministic service process for each server, with mean service rate $\mu_s$ $^{\text{elements}}/_{\text{s}}$. This implies we can treat each context as an independent $M/D/1$ queueing station ($M$arkovian, or memoryless, arrival process; $D$eterministic service process; 1 server) [58]. For this $M/D/1$ system, the maximum achievable throughput (per stream) is

$$\mu_s = \frac{1}{C \cdot t_{CLK}} \tag{5.14}$$

and the total achievable throughput is

$$T_{TOT} = C \cdot \mu_s. \tag{5.15}$$

The average (mean) waiting time of each data element in the queue is [58]

$$W_q = \frac{1}{\mu_s} \cdot \frac{\rho}{2\,(1-\rho)}, \tag{5.16}$$

where $\rho = \lambda/\mu_s$ (the utilization). The (deterministic) time in the pipelined circuit is

$$W_s = \frac{1}{\mu_s} = C \cdot t_{CLK}, \tag{5.17}$$

and therefore the average latency, $W_T$, (elapsed time from arrival to completion of processing; we assume one output is generated per input) for each data element is

$$W_T = W_q + W_s = \frac{1}{\mu_s} \cdot \frac{\rho}{2\,(1-\rho)} + C \cdot t_{CLK}. \tag{5.18}$$

The average (mean) occupancy of each queue is [58]

$$N_q = \lambda W_q = \frac{\rho^2}{2\,(1-\rho)}. \tag{5.19}$$

The second case to consider is when $N > C$, and we are exploiting both fine- and coarse-grained context switching. We will make the simplifying assumption that $N$ is an integer multiple of $C$. The schedule is a fixed, hierarchical, round-robin schedule with period $R_S$ (for each stream) that acts as follows:

1. a set of $C$ input streams is chosen to share the hardware resource and within that set, a round-robin schedule is used;

2. after $R_S$ rounds, the current set of input streams' state is swapped out to secondary memory and the next set of $C$ input streams' state is swapped in (the time required to complete this operation is given as $S$ clock cycles), this set is then scheduled to use the hardware in round-robin fashion;

3. the entire collection of $N/C$ input stream sets is also chosen in round-robin fashion (hence the label hierarchical, round-robin schedule), such that once every individual input stream has had $R_S$ input elements processed the high-level schedule returns to the first set of $C$ input streams.

The impact of $N > C$ on the queueing model starts with the effective service rate expression. The number of clock cycles to complete a full round of the hierarchical schedule (during which each server services $R_S$ elements) is $R_S N + SN/C$. This implies the effective service rate is

$$\mu_s = \frac{R_S}{(R_S N + SN/C) \cdot t_{CLK}} \text{ elements/s}, \tag{5.20}$$

which simplifies to (5.14) when $S = 0$ (i.e., context switches are free) and $N = C$. The total achievable throughput, $T_{TOT}$, is now $N \cdot \mu_s$, or

$$\begin{aligned} T_{TOT} &= \frac{N \cdot R_S}{(R_S N + SN/C) \cdot t_{CLK}} \\ &= \frac{R_S}{(R_S + S/C) \cdot t_{CLK}} \text{ elements/s}. \end{aligned} \tag{5.21}$$

To develop an expression for the average (mean) time that each data element waits in one of the input buffers, we start by using (5.16), the $M/D/1$ expression for mean queue waiting time, and add a term, $W_h$, to reflect the additional time waiting in the queue due to the hierarchical, round-robin schedule. The additional waiting time is experienced by the fraction of data elements that arrive when their input stream is swapped out (i.e., not receiving service). This fraction is $\frac{R_S(N-C)+SN/C}{R_S N+SN/C}$, or the number of clocks a stream is swapped out divided by the number of clocks in a full schedule round. The average additional time experienced by this fraction of input elements is one half of the time the stream is swapped out, $(R_S (N - C) + SN/C) \cdot t_{CLK}/2$. Multiplying the additional time by the fraction that experience the additional time yields

$$\begin{aligned} W_h &= \frac{(R_S (N - C) + SN/C) \cdot t_{CLK}}{2} \cdot \frac{R_S (N - C) + SN/C}{R_S N + SN/C} \\ &= \frac{(R_S (N - C) + SN/C)^2 \cdot t_{CLK}}{2 (R_S N + SN/C)}. \end{aligned} \tag{5.22}$$

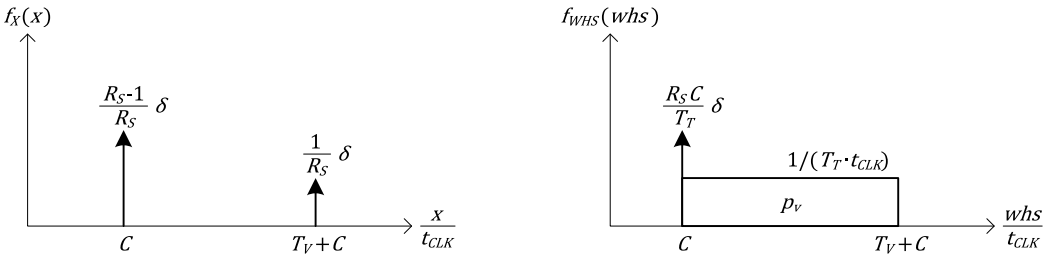
The time in the pipelined circuit does not change from $W_s = C \cdot t_{CLK}$, and therefore the average latency from arrival to completion of processing is

$$\begin{aligned} W_T &= W_q + W_h + W_s \\ &= \frac{1}{\mu_s} \cdot \frac{\rho}{2\left(1-\rho\right)} + \frac{\left(R_S\left(N-C\right) + SN/C\right)^2 \cdot t_{CLK}}{2\left(R_S N + SN/C\right)} + C \cdot t_{CLK}. \end{aligned} \tag{5.23}$$

Note that the expression for time in the server, $W_s$, no longer equals $1/\mu_s$, because the waiting time due to coarse-grained context switching is accounted for in the expression for $W_h$ rather than $W_s$. The expression for the average (mean) queue occupancy, $N_q$, does not change from (5.19); however, the expression for $\mu_s$, the effective service rate, has changed to (5.20). The mean number of elements waiting in the buffer due to the hierarchical, round-robin scheduling is $N_h = \lambda W_h$.

The above discussion provides analytic performance expressions for total achievable throughput, $T_{TOT}$, and the average latency experienced by each data element, $W_T$, both in the buffer, $W_q + W_h$, and in computation, $W_s$. Also available is the average occupancy of the buffer, $N_q + N_h$. For a summary of these queueing model equations, see Table B.2 in Appendix B.

The effective service rate, $\mu_s$, in (5.20) can also be derived from an equivalent service time distribution shown in Figure 5.10a with random variable $X$ where $\mu_s = \frac{1}{E[X]}$. Also, the sum of the additional time waiting in the queue and the time in the server, $W_h + W_s$, in (5.17, 5.22) can be derived from an equivalent distribution shown in Figure 5.10b with random variable $WHS$ where $W_h + W_s = E\left[WHS\right]$. In both figures, $T_V$ is the total vacation time for a full round of the hierarchical schedule equal to $R_S\left(N-C\right) + SN/C$, $T_T$ is the total time for a full round of the hierarchical schedule equal to $R_S N + SN/C$, and $p_v$ is the fraction of time that the server is in a vacation period equal to $\frac{T_V}{T_T}$.



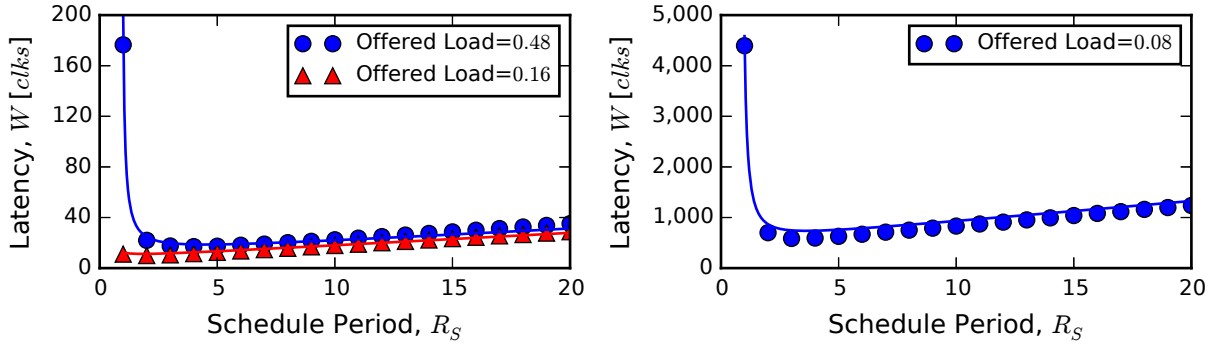

(a) Equivalent service time distribution, $X$

(b) Equivalent $W_h$+$W_s$ distribution, $WHS$

Figure 5.10: Equivalent distributions for M/D/1 model equations. (a) $\mu_s = \frac{1}{E[X]}$, and (b) $W_h + W_s = E\,[WHS]$.

### 5.2.2 Validation

To validate these modeling expressions, we developed a cycle-accurate discrete-event simulation of the system and measured the average latency of data elements from when they enter the input queue to when they exit the system. Consider a candidate design with 4 fine-grained contexts ($C = 4$), 8 total contexts ($N = 8$), and a 4 clock overhead to perform a coarse-grained context switch ($S = 4$). Figure 5.11a plots the total latency, $W_T$, predicted by (5.23) vs. the schedule period, $R_S$, for two different offered loads. The points on the graph correspond to empirical results from the discrete-event simulation run with the same parameters. The offered load is defined as the ratio of the aggregate arrival rate (of all streams) to the peak service rate of the system (i.e., when $S = 0$). Offered load then evaluates to $N \cdot \lambda \cdot t_{CLK}$. We draw two conclusions from this figure. First, there is good correspondence between the analytical model and the empirical simulation results. This bolsters our confidence that the analytic model is reasonable. Second, as is readily apparent in the graph, the schedule period that optimally minimizes latency is different for different offered loads. At low offered load (0.16) minimum latency is experienced with schedule period $R_S = 2$; and at a higher offered load (0.48) one must increase the schedule period to $R_S = 4$ to achieve minimum latency.




(a) $C = 4$, $N = 8$, $S = 4$ (b) $C = 10$, $N = 100$, $S = 100$

Figure 5.11: Discrete-event simulation of latency vs. schedule period for two sets of parameters for the M/D/1 queueing model. The curves are analytically generated from (5.23) and the points are empirically measured via the discrete-event simulation.

Likewise, we consider a second candidate design with 10 fine-grain contexts ($C = 10$), 100 total contexts ($N = 100$), and a 100 clock cycle overhead to perform a coarse-grained context switch ($S = 100$) to validate the analytic model against a different set of parameters. Figure 5.11b plots the total latency, $W_T$, with an offered load of 0.08. Again, we see that there is good correspondence between the analytical model and the empirical simulation results with minimum latency occurring at $R_S = 3$.

## 5.3 M/G/1 queueing model development with vacation model

We now consider a model of the system with an independent M/G/1 queueing station (*M*arkovian, or memoryless, arrival process; *G*eneral service process; 1 server) where the server can go on vacation for some time. In this model, the server distribution is general and, during a vacation, the server is not servicing any of its queued data elements.

Referring back to the queueing model in Figure 5.2b, the system consists of $N$ queueing stations which all share a single physical server which is represented as being $N$ "virtual" servers. We employ the same fixed, hierarchical, round-robin schedule as was used in the M/D/1 model. A set of $C$ input streams share the hardware resource and execute in a round-robin fashion in the server (in the computation pipeline). After $R_S$ rounds, the current set of $C$ input streams' state is swapped out (context switched) to secondary memory with cost $S$ and the next set is swapped in. The collection of input stream sets are also context switched in a round-robin fashion.

When the server takes a vacation, it is one of two kinds: short or long. Short vacations are due to fine-grain context switching and are taken when the server is idle (i.e., the FIFO queue is empty). They occur for $C$ clock cycles at a time. Long vacations are due to coarse-grain context switching and are taken when the schedule has completed $R_S$ rounds of the current set of $C$ input streams and context switches to execute the other $N - C$ input streams, all according to the fixed schedule. They occur for the duration of the vacation period. During these times, no new elements are processed until the vacation is complete.

A single queueing station of the system, shown in Figure 5.12, consists of a FIFO queue and "virtual" server. Here, $W_q$ is the mean waiting time in the queue and $W_s$ is the time spent in the server. When the server takes a vacation, this produces an additional waiting time at the head of the queue, $W_h$. $W_h$ is included in $W_q$ and emphasized in bold in the figure. It is derived from a vacation waiting time $V$ whose distribution is determined by the fixed, hierarchical, round-robin schedule and defined below. $W_s$ is modeled by a fixed service time $X$.

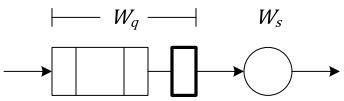
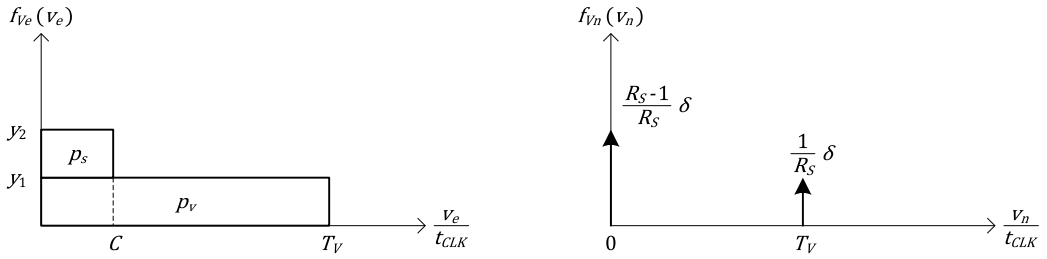



Figure 5.12: Single queueing station of system.

### 5.3.1 Vacation waiting time model

To model the vacation waiting time, $V$, we define two sub-model distributions. The first is for an empty queue shown in Figure 5.13a for waiting time, $V_e$, and the second is for a non-empty queue shown in Figure 5.13b for waiting time, $V_n$.

For the case of an empty queue, the probability density function, $f_{V_e}(v_e)$, in Figure 5.13a, shows the distribution of the vacation waiting time for continuous random variable $V_e$. In this figure, $T_V$ is the time in a long vacation period (i.e., during a coarse-grain context switch), $p_s$ is the fraction of time in a service period, and $p_v$ is the fraction of time in the long vacation period. Additionally, $y_1$ and $y_2$ are height levels to be solved for, which depend on the areas of the two rectangles. Along the $x$-axis, $v_e$ is measured in time, but is normalized to clocks (by dividing by $t_{CLK}$).




(a) For an empty queue

(b) For a non-empty queue

Figure 5.13: Sub-model distributions used in the derivation of the vacation waiting time model.

We start the model by defining the total number of clocks to complete a full round of the hierarchical schedule as

$$T_T = R_S N + SN/C. \tag{5.24}$$

The number of clocks in a long vacation period, which is the time during which the server has context switched to process other contexts, is

$$\begin{aligned} T_V &= T_T - R_S C \\ &= R_S (N - C) + SN/C. \end{aligned} \tag{5.25}$$

The fraction of time in a service period is

$$p_s = R_S C / T_T, \tag{5.26}$$

and the fraction of time in a long vacation period is

$$\begin{aligned} p_v &= 1 - p_s \\ &= 1 - R_S C / T_T. \end{aligned} \tag{5.27}$$

Returning to Figure 5.13a, $f_{V_e}(v_e)$, is defined as two stacked rectangles with areas $p_s$ and $p_v$ and height levels $y_1$ and $y_2$. Since the sum of the areas must equal 1, we can use this to calculate the height levels.

We first calculate $y_1$ from the $p_v$ area as

$$\begin{aligned} p_v &= (T_V \cdot t_{CLK}) \cdot y_1 \\ y_1 &= \frac{p_v}{T_V \cdot t_{CLK}} \\ &= \frac{1 - p_s}{T_V \cdot t_{CLK}}. \end{aligned} \tag{5.28}$$

Likewise, we calculate $y_2$ from $y_1$ and the $p_s$ area as

$$\begin{aligned} p_s &= C \cdot t_{CLK} \cdot (y_2 - y_1) \\ y_2 &= \frac{p_s}{C \cdot t_{CLK}} + y_1 \\ &= \left(\frac{p_s}{C} + \frac{1 - p_s}{T_V}\right) \cdot \frac{1}{t_{CLK}}. \end{aligned} \tag{5.29}$$

From these results, we can define the probability density function [59], $f_{V_e}(v_e)$, as

$$f_{V_e}(v_e) = \begin{cases} \left(\frac{p_s}{C} + \frac{1-p_s}{T_V}\right) \cdot \frac{1}{t_{CLK}} & 0 \leq v_e \leq C \cdot t_{CLK}, \\ \frac{1-p_s}{T_V \cdot t_{CLK}} & C \cdot t_{CLK} < v_e \leq T_V \cdot t_{CLK}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.30}$$

Writing this using step functions ($u(x)$ notation) [59], we get

$$f_{V_e}(v_e) = \left(\frac{1 - p_s}{T_V \cdot t_{CLK}}\right) \cdot [u(v_e) - u(v_e - T_V \cdot t_{CLK})] + \left(\frac{p_s}{C \cdot t_{CLK}}\right) [u(v_e) - u(v_e - C \cdot t_{CLK})] \tag{5.31}$$

where

$$u(x) = \begin{cases} 0 & x < 0, \\ 1 & x \geq 0. \end{cases}$$

Integrating $f_{V_e}(v_e)$ across all time should give a total probability of 1:

$$\begin{aligned}\int_{-\infty}^{\infty} f_{V_e}(v_e)\, dv_e &= \left(\frac{1-p_s}{T_V \cdot t_{CLK}}\right)[T_V \cdot t_{CLK}] + \left(\frac{p_s}{C \cdot t_{CLK}}\right)[C \cdot t_{CLK}] \\ &= 1 - p_s + p_s \\ &= 1 \\ &True.\end{aligned}$$

Next, for the case of a non-empty queue, the probability density function, $f_{V_n}(v_n)$, in Figure 5.13b, shows the distribution of the vacation waiting time for continuous random variable $V_n$. In this figure, the impulses defined at 0 and $T_V \cdot t_{CLK}$ are Dirac delta functions ($\delta(x)$ notation) [59] where

$$d_\epsilon(x) = \begin{cases} 1/\epsilon & -\epsilon/2 \leq x \leq \epsilon/2, \\ 0 & otherwise, \end{cases}$$

$$\delta(x) = \lim_{\epsilon \to 0} d_\epsilon(x),$$

and $\int_{-\infty}^{\infty} \delta(x)\, dx = 1$. Along the $x$-axis, $v_n$ is measured in time, but is normalized to clocks (by dividing by $t_{CLK}$).

Here, the two impulses define the vacation waiting time (and are normalized to the total number of jobs, $R_S$, in a full round of the hierarchical schedule). The first impulse, at $v_n = 0$ (i.e., no wait time), defines a group of $R_S - 1$ jobs being serviced consecutively. When these jobs complete, the server goes on a long vacation, and the next job waits for the vacation to complete. The wait time of the one job is then modeled by the second impulse, at $v_n = T_V \cdot t_{CLK}$.

The probability density function, $f_{V_n}(v_n)$, is then

$$f_{V_n}(v_n) = \frac{R_S - 1}{R_S}\delta(v_n) + \frac{1}{R_S}\delta(v_n - T_V \cdot t_{CLK}). \tag{5.32}$$

Integrating $f_{V_n}(v_n)$ across all time should give a total probability of 1:

$$\begin{aligned}\int_{-\infty}^{\infty} f_{V_n}(v_n)\, dv_n &= \frac{R_S - 1}{R_S} + \frac{1}{R_S} \\ &= 1 \\ &True.\end{aligned}$$

Since the queueing station is using the fixed, hierarchical schedule, the probability of the queue being empty changes depending on whether the server is in a service or vacation period. Although it might change, we will assume that this probability is fixed for the purposes of combining the probability density functions, $f_{V_e}(v_e)$ and $f_{V_n}(v_n)$, into a single approximate function, $f_V(v)$. When we do this, we get

$$\begin{aligned}f_V(v) &= p_0 \cdot f_{V_e}(v) + (1 - p_0) \cdot f_{V_n}(v) \\ &= p_0 \cdot \left[\frac{1 - p_s}{T_V \cdot t_{CLK}}\left[u(v) - u(v - T_V \cdot t_{CLK})\right] + \frac{p_s}{C \cdot t_{CLK}}\left[u(v) - u(v - C \cdot t_{CLK})\right]\right] \\ &\quad + (1 - p_0) \cdot \left[\frac{R_S - 1}{R_S}\delta(v) + \frac{1}{R_S}\delta(v - T_V \cdot t_{CLK})\right]\end{aligned} \tag{5.33}$$

where $p_0 = 1 - \rho$, the probability of an empty queue.

Integrating $f_V(v)$ across all time should give a total probability of 1:

$$\begin{aligned}\int_{-\infty}^{\infty} f_V(v)\,dv &= p_0 \cdot \left[\frac{1-p_s}{T_V \cdot t_{CLK}} T_V \cdot t_{CLK} + \frac{p_s}{C \cdot t_{CLK}} C \cdot t_{CLK}\right] + (1-p_0)\left[\frac{R_S - 1}{R_S} + \frac{1}{R_S}\right] \\ &= p_0 \cdot [1 - p_s + p_s] + (1-p_0) \cdot [1] \\ &= p_0 + 1 - p_0 \\ &= 1 \\ &True.\end{aligned}$$

Calculating the expected value [59], $E[V]$, of the vacation waiting time to get the mean vacation waiting time, we get the following:

$$\begin{aligned}E[V] = \int_{-\infty}^{\infty} v \cdot f_V(v)\,dv &= p_0 \cdot \left[\frac{1-p_s}{T_V \cdot t_{CLK}} \frac{1}{2}\left[(T_V \cdot t_{CLK})^2\right] + \frac{p_s}{C \cdot t_{CLK}} \frac{1}{2}\left[(C \cdot t_{CLK})^2\right]\right] \\ &\quad + (1-p_0) \cdot \left[0 + \frac{1}{R_S} \cdot T_V \cdot t_{CLK}\right] \\ &= \left[\frac{1}{2} p_0 \cdot [(1-p_s) \cdot T_V + p_s \cdot C] + (1-p_0) \cdot \left[\frac{T_V}{R_S}\right]\right] \cdot t_{CLK}. \end{aligned} \tag{5.34}$$

The mean vacation waiting time will be denoted as $\overline{V}$ and will be used later in the derivation of the queueing model equations.

### 5.3.2 Service time model

The model for the service time, $X$, is fixed and deterministic. For every job that enters the computational pipeline, it always takes exactly $C$ clock cycles to complete service of that job. Therefore, the service time can be modeled as a single impulse function at $x = C \cdot t_{CLK}$.

The resulting probability density function, $f_X(x)$, is

$$f_X(x) = \delta(x - C \cdot t_{CLK}). \tag{5.35}$$

Integrating $f_X(x)$ across all time should give a total probability of 1:

$$\int_{-\infty}^{\infty} f_X(x)\,dx = 1$$

$$True.$$

Next, we can calculate $E[X]$, the expected service time, as

$$E[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)\,dx = C \cdot t_{CLK}. \tag{5.36}$$

As part of the queueing model, we also need to know the second moment of the service time, $E[X^2]$. This is calculated as

$$E\left[X^2\right] = \int_{-\infty}^{\infty} x^2 f_X(x)\,dx = C^2 \cdot t_{CLK}^2. \tag{5.37}$$

We will denote $E[X]$ as $\overline{X}$ and $E[X^2]$ as $\overline{X^2}$ and will use them in the development of the queueing model equations that follow.

### 5.3.3 Queueing model

The queueing equations developed here will be for the general case ($N > C$) and will include effects due to vacations. They can easily be reduced to the $N = C$ case by setting $S = 0$.

First, we start by defining the (deterministic) time in the pipelined circuit as

$$W_s = \overline{X} = C \cdot t_{CLK}. \tag{5.38}$$

The effective service rate, $\mu_s$, of a "virtual" server is not equal to $1/W_s$, but has to take into account the long vacation time due to a coarse-grain context switch. We can do this by defining the service rate as the number of jobs processed by a stream in one full period of the hierarchical schedule. Therefore, for this M/G/1 system, the service rate, which is also the maximum achievable throughput (per stream), is

$$\begin{aligned} \mu_s &= \frac{R_S}{T_T \cdot t_{CLK}} \\ &= \frac{R_S}{(R_S N + SN/C) \cdot t_{CLK}}. \end{aligned} \tag{5.39}$$

The total achievable throughput is then $T_{TOT} = N \cdot \mu_s$, or

$$T_{TOT} = \frac{R_S}{(R_S + S/C) \cdot t_{CLK}}. \tag{5.40}$$

The average (mean) waiting time of each data element for an M/G/1 system (without vacations) is determined by the Pollaczek-Khinchin (P-K) formula as $W_q = \frac{\lambda \overline{X^2}}{2(1-\rho)}$ where $\rho = \lambda/\mu_s$, $\lambda$ is the arrival rate, and $\overline{X^2}$ is the second moment of the service time [48]. To account for vacations in this formula, we need to add an additional waiting time at the head of the queue ($W_h$):

$$W_q = \frac{\lambda \overline{X^2}}{2\,(1-\rho)} + W_h.$$

To determine $W_h$, we need to use equation 3.46 in [48] of the P-K formula derivation which uses a mean residual time, $R$, to solve for the wait time in the queue. The wait time formula then becomes $W_q = R + \frac{1}{\mu_s} N_q$ where $R = \frac{1}{2}\lambda \overline{X^2}$ and $N_q = \lambda W_q$ (Little's Theorem [58]).

Now, we can add the mean vacation waiting time, $\overline{V}$, to $W_q$ and apply Little's Theorem:

$$W_q = R + \frac{1}{\mu_s}\lambda W_q + \overline{V}.$$

Solving for $W_q$ then gives

$$\begin{aligned} W_q &= \frac{R + \overline{V}}{1-\rho} \\ &= \frac{\lambda \overline{X^2}}{2\left(1-\rho\right)} + \frac{\overline{V}}{1-\rho}. \end{aligned} \tag{5.41}$$

Therefore, we can determine that $W_h = \frac{\overline{V}}{1-\rho}$.

Next, the average latency (elapsed time from arrival to completion of processing; we assume one output is generated per input) for each data element is

$$\begin{aligned} W_T = W_q + W_s &= \frac{\lambda \overline{X^2}}{2\left(1-\rho\right)} + \frac{\overline{V}}{1-\rho} + \overline{X} \\ &= \left[\frac{C^2\left(\lambda \cdot t_{CLK}\right)}{2\left(1-\rho\right)} + \frac{1}{2}\left[\left(1-p_s\right)\cdot T_V + p_s \cdot C\right] + \frac{\rho}{1-\rho}\cdot\left[\frac{T_V}{R_S}\right] + C\right]\cdot t_{CLK}. \end{aligned} \tag{5.42}$$

The average (mean) occupancy of each queue is $N_q = \lambda W_q$ [58].

The expression for the average latency consists of 5 terms that model the delay through the system. The first term ($\frac{\lambda C^2 t_{CLK}^2}{2(1-\rho)}$) models service queueing delay. The second term ($\frac{1}{2}\left(1-p_s\right)T_V \cdot t_{CLK}$) models long vacation delay during a vacation period. The third term ($\frac{1}{2}p_s \cdot C \cdot t_{CLK}$) models short vacation delay during a service period for an empty queue. The fourth term ($\frac{\rho}{1-\rho}\frac{T_V \cdot t_{CLK}}{R_S}$) models vacation queueing delay. And the fifth term ($C \cdot t_{CLK}$) models service time delay. Of these delays, those due to the long vacation period (the second and fourth terms) have the greatest impact on the average latency. The vacation queueing delay (fourth term) accounts for large initial latency at low $R_S$ due to the effect

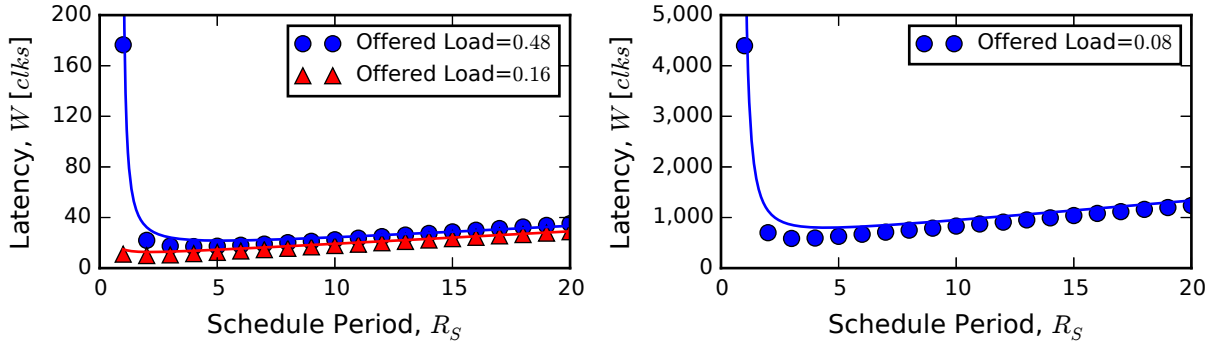
of $S$, but decreases when $R_S$ increases (by amortizing the effect of $S$). The long vacation delay (second term) accounts for a gradual increase in latency with increasing $R_S$ since $T_V$ increases with $R_S$.

Referring back to the M/D/1 model, the average latency expression in (5.23) models queueing delay (with both the service and vacation periods lumped together), long vacation delay, and service time delay. These account for the greatest effects to the latency. In comparison, the M/G/1 model separately models queueing delay for the service period (in the first term of (5.42)) and vacation period (in the fourth term). It also models the additional effect of short vacation delay during a service period for an empty queue (in the third term).

Analytical performance expressions were derived using an M/G/1 model with vacations by defining distributions for the vacation waiting time and service time. These include additional effects that were not considered in the M/D/1 model. We have performance equations for the total achievable throughput, $T_{TOT}$, the average latency, $W_T$, experienced by a data element, and the average occupancy of the buffer, $N_q$. For a summary of these queueing model equations, see Table B.3 in Appendix B.

### 5.3.4 Validation

To validate these modeling expressions, we compare in Figure 5.14a the analytical expression for latency, $W_T$, in (5.42) to the empirical simulation results of the cycle-accurate discrete-event simulation of the system (first introduced in Section 5.2.2) for the candidate design with 4 fine-grain contexts ($C = 4$), 8 total contexts ($N = 8$), and 4 clock overhead due to a coarse-grain context switch ($S = 4$). We can observe that the analytical curves and empirical results are in good alignment. Compared to Figure 5.11a, the M/G/1 curves predict slightly




(a) $C = 4$, $N = 8$, $S = 4$ (b) $C = 10$, $N = 100$, $S = 100$

Figure 5.14: Discrete-event simulation of latency vs. schedule period for two sets of parameters for the M/G/1 queueing model. The curves are analytically generated from (5.42) and the points are empirically measured via the discrete-event simulation.

higher latency. This is expected because the M/G/1 model includes more effects in the latency equation.

We also consider the second candidate design in Figure 5.14b with 10 fine-grain contexts ($C = 10$), 100 total contexts ($N = 100$), and 100 clock overhead for a coarse-grain context switch ($S = 100$). We can observe again that the analytical curve and empirical results are in good alignment, with the analytical curve being slightly higher than the curve in Figure 5.11b.

## 5.4 Calibration of clock and resource models for three $C$-slowed applications

We use three applications across two technologies to validate and calibrate (5.13), the model for $t_{CLK}$, and a resource model. Once calibrated, they can be used in analytic model prediction results presented in Section 5.5. The three applications, listed in Table 5.2, are (1) a

Table 5.2: Applications implemented using $C$-slow techniques.

| Abbr. | Name | Description |
|---|---|---|
| COS | Cosine application | Synthetic cosine application implemented via a Taylor series expansion with added feedback |
| AES | AES application | Advanced Encryption Standard (AES) cipher in cipher-block chaining (CBC) mode for encryption |
| SHA | SHA-2 application | Secure Hash Algorithm (SHA-2) with 256 and 512 bit digests (SHA-256 and SHA-512) |

synthetic cosine application implemented via a Taylor series expansion with added feedback, (2) the Advanced Encryption Standard (AES) cipher in cipher-block chaining (CBC) mode for encryption, and (3) the Secure Hash Algorithm (SHA-2) with 256 and 512 bit digests (SHA-256 and SHA-512). These applications were chosen because they each have a long combinational logic path with feedback from output to input, thus making simple pipelining alone insufficient to effectively utilize their logic blocks. This makes them good candidates for $C$-slow (pipelining) with virtualization where independent streams are scheduled for execution in the pipeline stages.

The applications are implemented on field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) technologies with measurements taken of the design speed and resource usage. The logic is $C$-slowed to support $C$ streams of computation with $N = C$ total streams (no secondary memory). For FPGA technology, we target two different parts and use two different tools. For the Cosine and AES applications, we target a Xilinx Virtex-4 XC4VLX100 FPGA and use the Xilinx ISE 13.4 tools for synthesis, place, & route of the hardware designs. For the SHA-2 application, we target a Xilinx Virtex-7 XC7VX485T FPGA and use the Xilinx Vivado 2013.4 tools. For ASIC technology, we target a 5M1P 0.18 $\mu$m process using the Virginia Tech VLSI for Telecommunications (VTVT) standard cell library and use the Cadence RTL Compiler 8.10 and Cadence Encounter 9.14 tools to

synthesize, place, & route the hardware designs. Only the Cosine and AES applications are implemented in ASIC technology. The clock period is unconstrained in the runs for both technologies.

Depending on the application, we use one of two interfaces. The first is the parallel stream interface in Figure 5.1 with $N$ parallel streams multiplexed into the virtualized hardware and demultiplexed back out. In this interface, as the number of streams grows, resource usage grows due to the multiplexer/demultiplexer growing in size. The second is a single stream interface with one input stream. The input stream contains fields within record-oriented data that are independent of each other. Each field then can be processed independently of any other field in the virtualized hardware. An upfront buffer queues up data elements for each field and extracts them using a multiplexer from any position in the buffer. A reorder queue at the output can reorder the field results back into their original order. The Cosine and AES applications use the parallel stream interface, and the SHA-2 application uses the single stream interface.

We calibrate a clock and resource model for each application and technology. The clock model is calibrated based on (5.13). The resource model is calibrated based on the equation

$$Resource\,(L, C) = k_1 + k_2 \cdot L + k_3 \cdot C \tag{5.43}$$

(valid only for a $C$-slowed circuit with no secondary memory; that is, $N = C$) where $L$ is a parameter proportional to the combinational logic path length, and $C$ is the number of streams supported in the system or the pipeline depth. The resource being modeled depends on the technology. For FPGA technology, the resource is the total slices used to implement the design. For ASIC technology, the resource is the total area of the core (i.e., the space inside the ring of pads). In this equation, $k_1$ models the fixed slices/area of the design, $k_2$

models the additional slices/area due to the combinational logic length, $L$, of the virtualized computation, and $k_3$ models the additional slices/area due to the number of streams, $C$, being supported or pipeline depth. For $C$-slowed applications that use the parallel interface, we expect $k_3$ to model the delta growth in slices/area of the multiplexer and demultiplexer in the interface as $C$ increases.

### 5.4.1 Synthetic cosine application with added feedback (COS)

The Cosine application, illustrated in Figure 5.15, consists of a cosine function, an output register, a feedback path, and an adder to mix the input with the output feedback. This is a synthetic application built to have a long combinational logic path through the cosine function via a configurable number of Taylor series terms, $N_t$, that approximate the cosine. The cosine function is pipelined with a configurable number of pipeline stages, $C$, to improve the clock period and to support $C$ virtual streams of computation. These streams are provided in parallel to the cosine function (via the parallel stream interface) and are accessed one at a time via an input multiplexer. They are fed back out via a demultiplexer at the output. To accommodate for different values of $N_t$, which controls the length of the combinational logic path, the clock period model in (5.13) is modified as follows:

$$t_{CLK}\left(N_t, C\right) = \frac{k_{11} + k_{12} \cdot N_t}{C} + \left(k_{21} + k_{22} \cdot N_t\right) \cdot \left(\ln C\right)^{0.7}. \tag{5.44}$$

In this equation, coefficients $k_1$ and $k_2$ are expanded with linear terms to model the effect of $N_t$ on the clock period. Thus, $k_1 \rightarrow k_{11} + k_{12} \cdot N_t$ and $k_2 \rightarrow k_{21} + k_{22} \cdot N_t$.

In the resource model of (5.43), we can replace $L$ with $N_t$ for the Cosine application:

$$\mathit{Resource}\left(N_t, C\right) = k_1 + k_2 \cdot N_t + k_3 \cdot C. \tag{5.45}$$

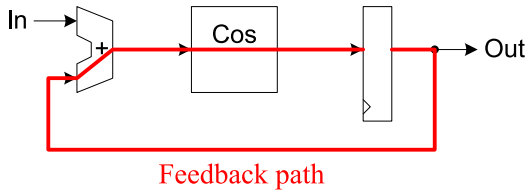



Figure 5.15: Block diagram of synthetic cosine application with added feedback. The cosine function is approximated with a Taylor series expansion of $N_t$ terms which form a long combinational logic path. With the added feedback path, this will determine the achievable clock rate.

### Clock model

In an experiment, we measure $t_{CLK}$ on 10 independent runs first for an FPGA with $N_t$ ranging from 2 to 24 terms, and $C$ ranging from 1 to 44 streams. Curve fitting the model in (5.44) across all of the data sets concurrently yields:

$$t_{CLK}\left(N_t, C\right) = \left[\frac{-11.5 + 11.8 \cdot N_t}{C} + \left(1.47 + 0.0079 \cdot N_t\right) \cdot \left(\ln C\right)^{0.7}\right] \text{ns}. \tag{5.46}$$

Comparing this expression to (5.13) shows that we are modeling the curve-fit total combinational logic delay, $k_1$, as $-11.5\,\text{ns} + 11.8\,\frac{\text{ns}}{\text{term}} \cdot N_t$, and $k_2$ as $1.47\,\text{ns} + 0.0079\,\frac{\text{ns}}{\text{term}} \cdot N_t$.

To validate this model, we compute the total achievable throughput, $T_{TOT}$, as $1/t_{CLK}$ (which is equivalent to (5.40) when $S = 0$ and $N = C$), and plot the observed data values from the synthesis, place, & route runs with the model prediction in Figure 5.16. Blue $\times$'s are included in the model and green $\times$'s are excluded from the model. The reason for excluding the green data points is because the pipeline stages are uneven and retiming does not do a good job at evenly distributing the pipeline registers in the combinational logic. We can see that the green $\times$'s consequently plateau until the last data point (which is at the maximum $C$ and uses all the pipeline registers in the design).

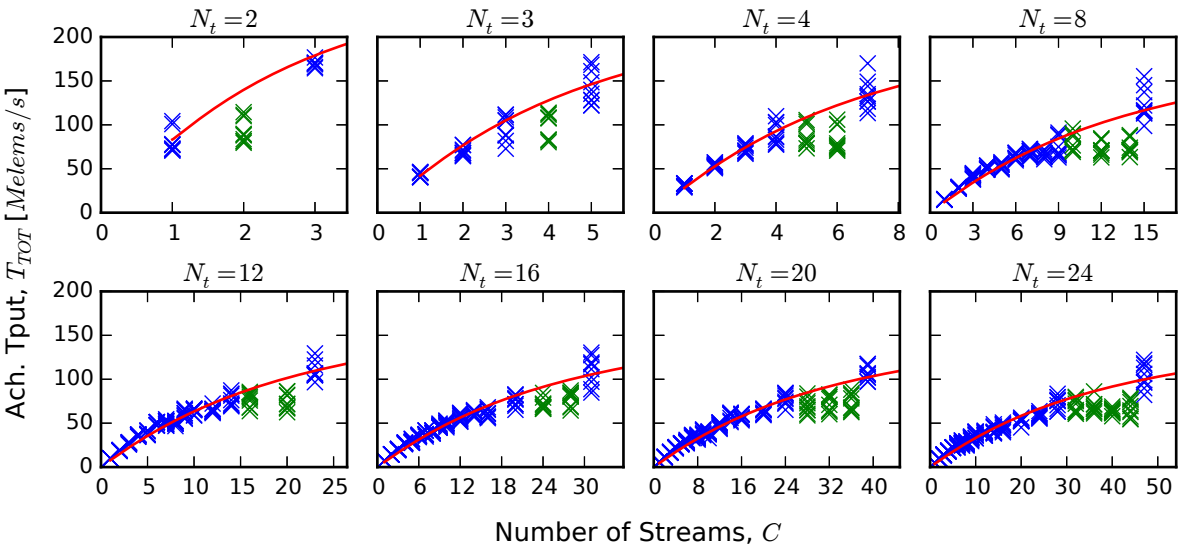



Figure 5.16: Calibrated total achievable throughput plot of the virtualized Cosine application with feedback on an FPGA. $N_t$ is the number of Taylor series terms in the cosine function. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. Blue ×'s are included in the model. Green ×'s are excluded from the model.

We make several observations about the total achievable throughput of the virtualized designs. First, the model does a reasonably good job of characterizing the shape of the curve. Second, throughput initially increases linearly (at low stream counts) but eventually starts to level off and adding additional streams does not provide as significant throughput gains. Essentially, the clock rate gains (due to deeper pipelining) are approaching their maximum benefit.

Next, we curve fit $t_{CLK}$ for the ASIC technology. The resulting model yields:

$$t_{CLK}\left(N_t, C\right) = \left[\frac{-40.4 + 28.5 \cdot N_t}{C} + \left(6.52 - 0.13 \cdot N_t\right) \cdot \left(\ln C\right)^{0.7}\right] \text{ns}. \tag{5.47}$$

Comparing this expression to (5.13) shows that we are modeling the curve-fit total combinational logic delay, $k_1$, as $-40.4\,\text{ns} + 28.5\,\frac{\text{ns}}{\text{term}} \cdot N_t$, and $k_2$ as $6.52\,\text{ns} - 0.13\,\frac{\text{ns}}{\text{term}} \cdot N_t$. The plot of the empirical performance results with the model prediction of the total achievable

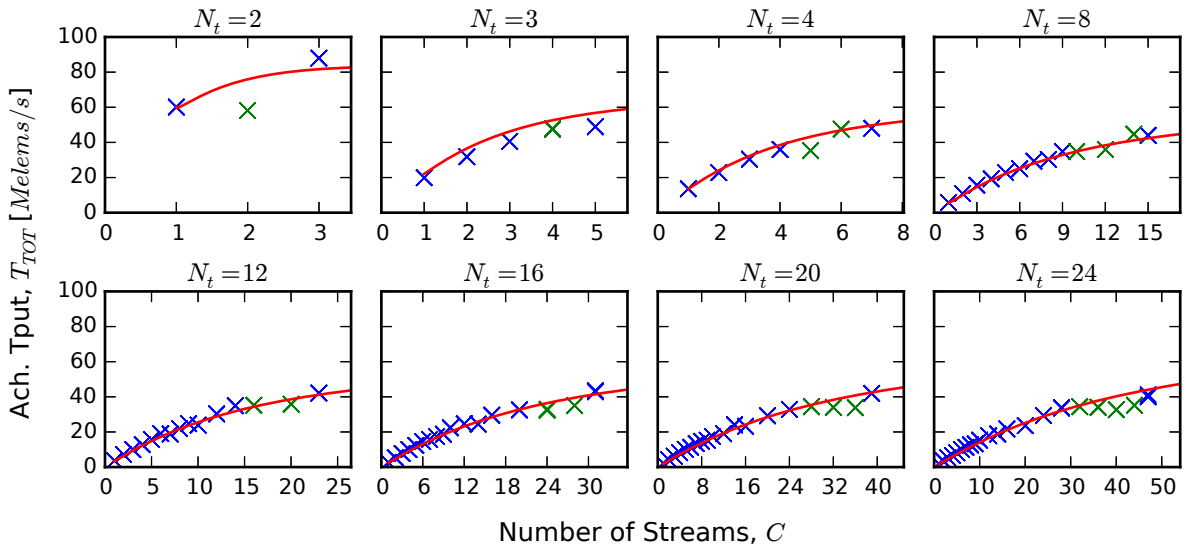



Figure 5.17: Calibrated total achievable throughput plot of the virtualized Cosine application with feedback on an ASIC. $N_t$ is the number of Taylor series terms in the cosine function. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. Blue ×'s are included in the model. Green ×'s are excluded from the model.

throughput for the ASIC design is shown in Figure 5.17. The observations made with FPGA technology clearly still hold with the ASIC technology.

### Resource model

For FPGA technology, we curve fit the total slices to (5.45) to yield the following model:

$$Slices\,(N_t, C) = 91.0 + 12.6 \cdot N_t + 90.1 \cdot C. \tag{5.48}$$

We can see in this equation that the fixed number of slices used is 91, that every additional term in the Taylor series of the cosine function uses 12.6 slices, and that every additional stream (which requires more multiplexer/demultiplexer logic) uses 90.1 slices. The plot of the empirical results with the model prediction of the slices is shown in Figure 5.18. We can

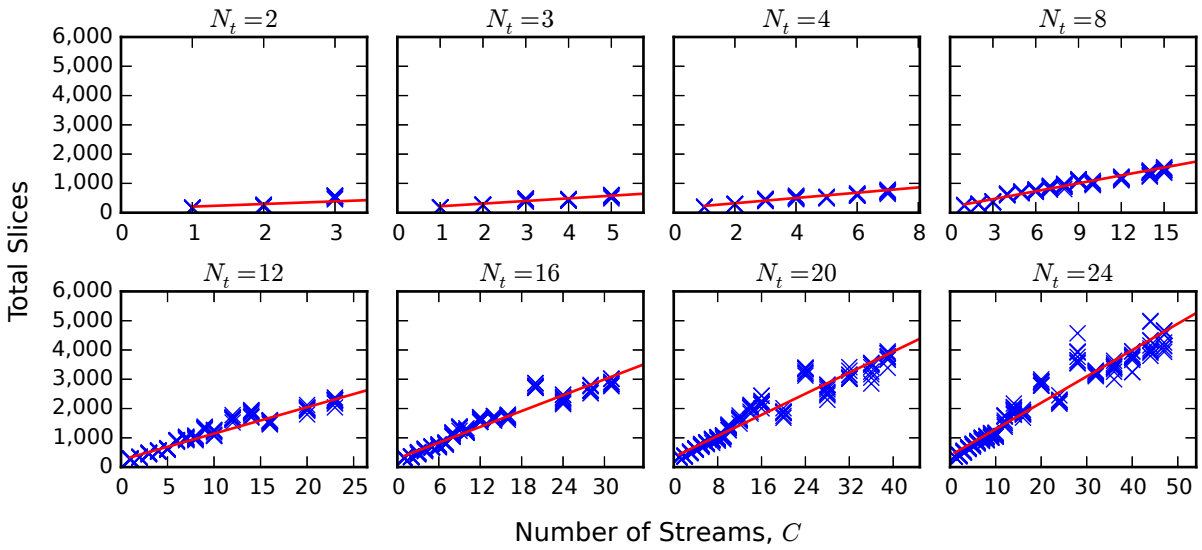



Figure 5.18: Calibrated total slices plot of the virtualized Cosine application with feedback on an FPGA. $N_t$ is the number of Taylor series terms in the cosine function. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. All data points are included in the model.

observe that the slices follow a linear trend with the number of streams ($C$) and terms ($N_t$), and that the model prediction aligns well with the measured slices data.

Next, we curve fit the total core area for ASIC technology to yield the following area model:

$$Area\,(N_t, C) = [-0.365 + 0.225 \cdot N_t + 0.069 \cdot C]\,\text{mm}^2. \tag{5.49}$$

Here, we can see that the fixed area cost is $-0.365\,\text{mm}^2$, that every additional term in the Taylor series of the cosine function accounts for $0.225\,\text{mm}^2$ of the area, and that every additional stream accounts for $0.069\,\text{mm}^2$ of the area. The plot of the empirical results with the model prediction of the area is shown in Figure 5.19. The observations made with FPGA technology still hold with the ASIC technology.

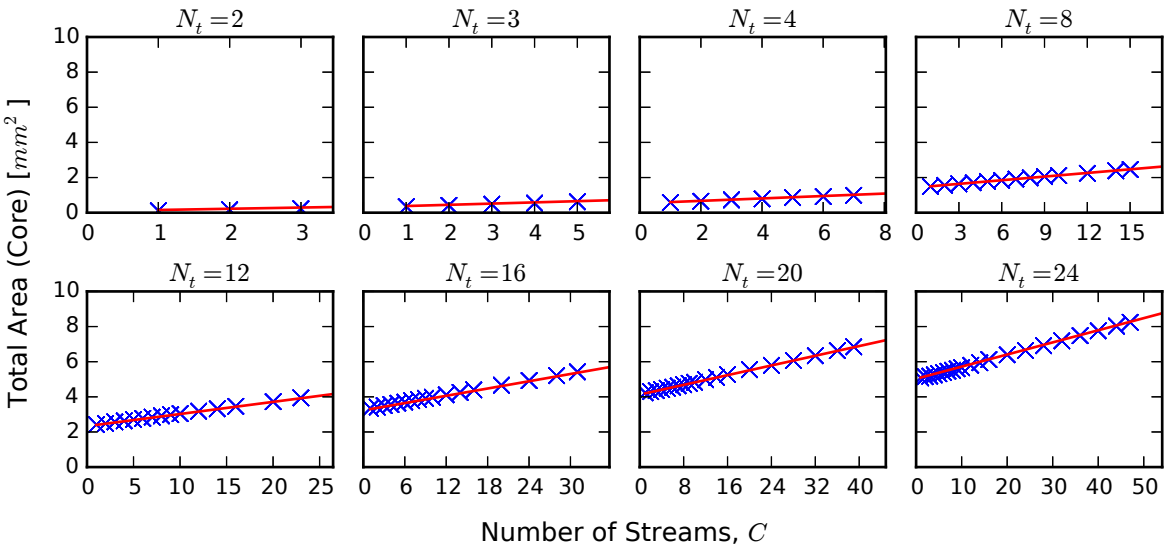



Figure 5.19: Calibrated total core area plot of the virtualized Cosine application with feedback on an ASIC. $N_t$ is the number of Taylor series terms in the cosine function. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. All data points are included in the model.

### 5.4.2 Advanced Encryption Standard (AES) cipher in cipher-block chaining mode

Next, we use an AES encryption cipher [60] in CBC block mode (that has a feedback path) illustrated in Figure 5.20 to calibrate the clock and resource models. In our implementation, the AES encryption cipher is fully unrolled forming a long combinational logic path with up to 14 rounds (the AES 256-bit standard), enabling us to investigate the impact of short vs. deep combinational logic functions. The number of rounds, $N_r$, in the cipher is configurable, which controls the length of the combinational logic. The AES block cipher is shown in the middle of the figure. Operating in cipher-block chaining (CBC) mode, an initialization vector (IV) is XOR'd with a plaintext block to produce the input to the cipher. The output is registered which contains the ciphertext output. The ciphertext is then fed back, through a multiplexer, to be mixed again with the next block of plaintext. Also, the AES encryption

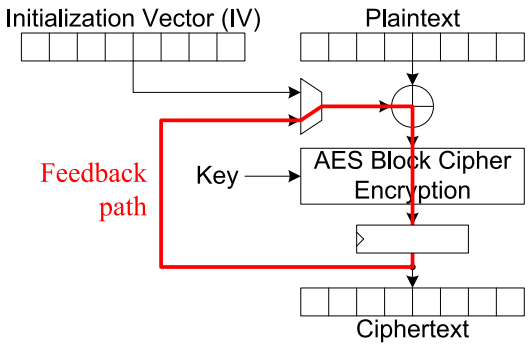



Figure 5.20: Block diagram of AES encryption cipher application in the CBC block mode. With a fully unrolled block cipher and no pipelining (initially), the highlighted feedback path will determine the achievable clock rate.

cipher has a configurable number of pipeline stages, $C$, to improve the clock period and to support $C$ virtual streams of computation. This is implemented via outer loop pipelining [61]. These streams are provided in parallel to the AES encryption cipher (via the parallel stream interface) and are accessed one at a time via an input multiplexer and fed back out via a demultiplexer at the output. To accommodate for different values of $N_r$, the clock period model in (5.13) is modified as follows:

$$t_{CLK}\left(N_r, C\right) = \frac{k_{11} + k_{12} \cdot N_r}{C} + \left(k_{21} + k_{22} \cdot N_r\right) \cdot \left(\ln C\right)^{0.7}. \tag{5.50}$$

In this equation, coefficients $k_1$ and $k_2$ are expanded with linear terms to model the effect of $N_r$ on the clock period. Thus, $k_1 \rightarrow k_{11} + k_{12} \cdot N_r$ and $k_2 \rightarrow k_{21} + k_{22} \cdot N_r$.

In the resource model of (5.43), we can replace $L$ with $N_r$:

$$Resource\left(N_r, C\right) = k_1 + k_2 \cdot N_r + k_3 \cdot C. \tag{5.51}$$

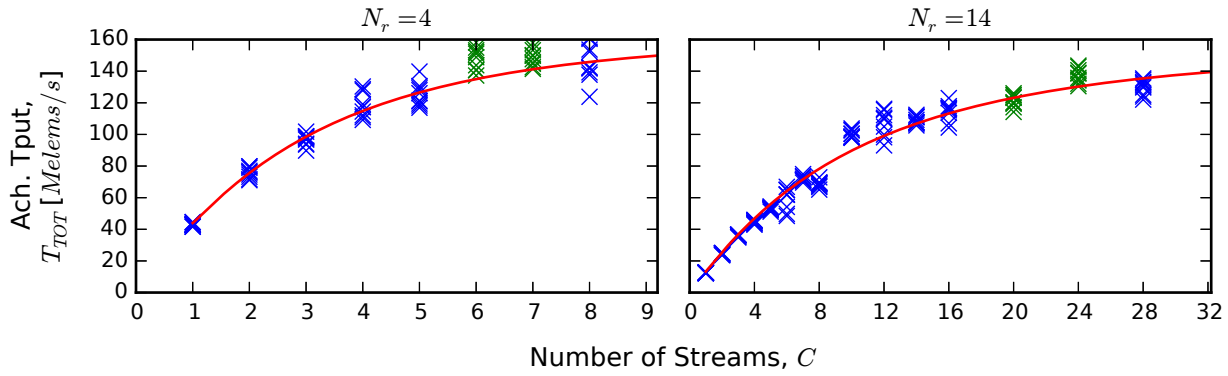



Figure 5.21: Calibrated total achievable throughput plot of the virtualized AES encryption cipher application on an FPGA. $N_r$ is the number of rounds in the cipher. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. Blue $\times$'s are included in the model. Green $\times$'s are excluded from the model.

### Clock model

In an experiment, we measure $t_{CLK}$ on 10 independent runs for an FPGA for $N_r = 4$ and 14, and $C$ ranging from 1 to 28 streams (i.e., up to 2 pipeline registers are included per round). Curve fitting the model in (5.50) across all of the data sets concurrently yields:

$$t_{CLK}\left(N_r, C\right) = \left[\frac{1.8 + 5.2 \cdot N_r}{C} + \left(2.56 - 0.038 \cdot N_r\right) \cdot \left(\ln C\right)^{0.7}\right] \text{ns}. \tag{5.52}$$

Comparing this expression to (5.13) shows that we are modeling the curve-fit total combinational logic delay, $k_1$, as $1.8\,\text{ns} + 5.2\,\frac{\text{ns}}{\text{rnd}} \cdot N_r$, and $k_2$ as $2.56\,\text{ns} - 0.038\,\frac{\text{ns}}{\text{rnd}} \cdot N_r$. The plot of the empirical performance with the model prediction of the total achievable throughput for this FPGA design is shown in Figure 5.21. We can observe that the model matches closely to the observed data values.

Next, we curve fit $t_{CLK}$ for the ASIC technology. The resulting model yields:

$$t_{CLK}\left(N_r, C\right) = \left[\frac{6.6 + 6.3 \cdot N_r}{C} + \left(5.34 + 0.026 \cdot N_r\right) \cdot \left(\ln C\right)^{0.7}\right] \text{ns}. \tag{5.53}$$

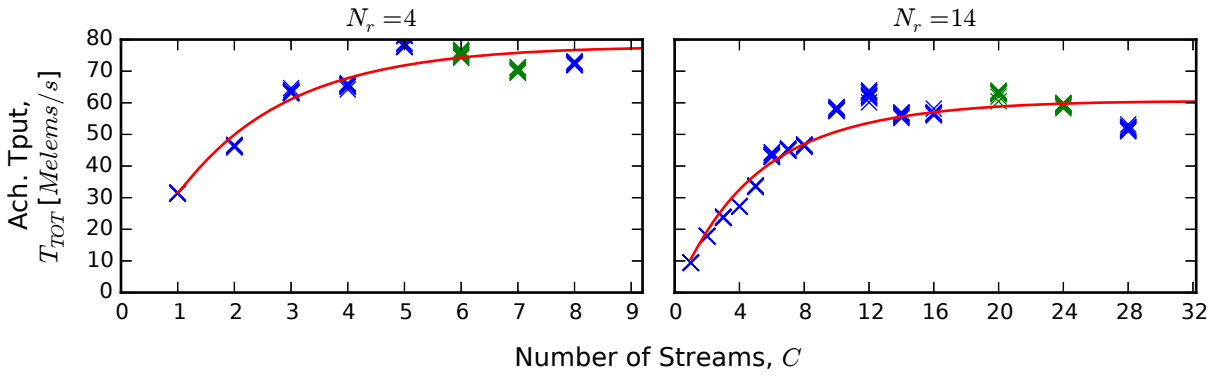



Figure 5.22: Calibrated total achievable throughput plot of the virtualized AES encryption cipher application on an ASIC. $N_r$ is the number of rounds in the cipher. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. Blue $\times$'s are included in the model. Green $\times$'s are excluded from the model.

Comparing this expression to (5.13) shows that we are modeling the curve-fit total combinational delay, $k_1$, as $6.6\,\text{ns} + 6.3\,\frac{\text{ns}}{\text{rnd}} \cdot N_r$, and $k_2$ as $5.34\,\text{ns} + 0.026\,\frac{\text{ns}}{\text{rnd}} \cdot N_r$. The plot of the empirical performance results with the model prediction of the total achievable throughput for the ASIC design is shown in Figure 5.22. The observations made with FPGA technology again still hold with the ASIC technology.

### Resource model

For FPGA technology, we curve fit the total slices to (5.51) to yield the following model:

$$Slices\,(N_r, C) = 1{,}097 + 1{,}434 \cdot N_r + 799 \cdot C. \tag{5.54}$$

We can see in this equation that the fixed number of slices used is 1,097, that every additional round of the AES encryption cipher uses 1,434 slices, and that every additional stream (which requires more multiplexer/demultiplexer logic) uses 799 slices. The plot of the empirical results with the model prediction of the slices is shown in Figure 5.23. We can observe that

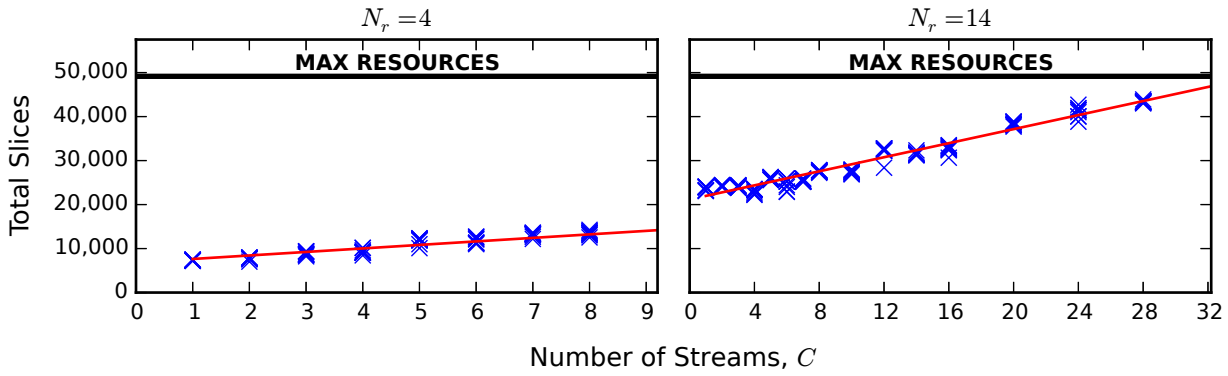



Figure 5.23: Calibrated total slices plot of the virtualized AES encryption cipher application on an FPGA. $N_r$ is the number of rounds in the cipher. $C$ refers to the number of streams supported and the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. All data points are included in the model.

the slices follow a linear trend with the number of streams, $C$, and the number of rounds, $N_r$, and that the model prediction aligns well with the measured slices data.

Next, we curve fit the total core area for ASIC technology to yield the following model:

$$Area\,(N_r, C) = [0.11 + 3.73 \cdot N_r + 0.56 \cdot C]\,\text{mm}^2. \tag{5.55}$$

Here, we can see that the fixed area cost is $0.11\,\text{mm}^2$, that every additional round of the AES encryption cipher accounts for $3.73\,\text{mm}^2$ of the area, and that every additional stream accounts for $0.56\,\text{mm}^2$ of the area. The plot of the empirical results with the model prediction of the area is shown in Figure 5.24. The observations made with FPGA technology still hold with the ASIC technology.

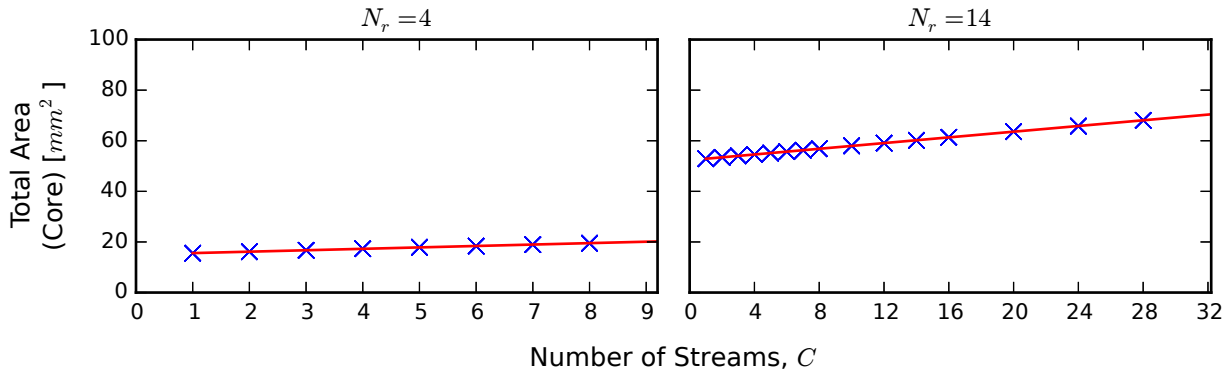



Figure 5.24: Calibrated total core area of the virtualized AES encryption cipher application on an ASIC. $N_r$ is the number of rounds in the cipher. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 10 tool flow runs. All data points are included in the model.

### 5.4.3 Secure Hash Algorithm (SHA-2) with 256 and 512 bit digests

Last, we use an SHA-2 cryptographic hash application [62] (that has a feedback path for processing multiple blocks in a stream) illustrated in Figure 5.25 to calibrate the clock and resource models. In our implementation, the SHA-2 core is fully unrolled forming a long combinational logic path with 64 rounds (for SHA-256) and 80 rounds (for SHA-512). The core is shown in the middle of the figure. To start, an initialization vector (IV) is provided with the first block of data to the input of the core (or hash function). The output is registered and eventually becomes the hash output after all data blocks have been processed. While processing each block, the intermediate hash values are fed back, through a multiplexer, to be mixed again with the next data block. The SHA-2 core has a configurable number of pipeline stages, $C$, to improve the clock period and to support $C$ virtual computations at a time. The SHA-2 application uses the single stream interface where the input stream contains arbitrary fields within record-oriented data that can be processed independently of each other. An upfront content addressable queue then buffers

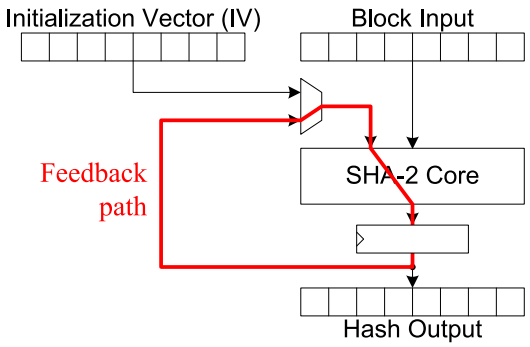



Figure 5.25: Block diagram of SHA-2 cryptographic hash application. With a fully unrolled SHA-2 core and no pipelining (initially), the highlighted feedback path will determine the achievable clock rate.

block data from each field which can be accessed at any position via a multiplexer. The buffer is fixed in size, independent of $C$.

We fix the number of rounds in SHA for the SHA-256 and SHA-512 hash standards. Therefore, we directly calibrate the clock period model to (5.13). And, for the resource model, we drop the $L$ term giving the modified equation:

$$Resource\,(C) = k_1 + k_2 \cdot C. \tag{5.56}$$

Since the buffer size is fixed and therefore independent of $C$, we expect the resource model to be flat for SHA.

### Clock model

In an experiment, we measure $t_{CLK}$ on 10 independent runs for an FPGA for SHA-256 and SHA-512 with $C$ ranging from 1 to 81 streams (i.e., up to 1 pipeline register per round

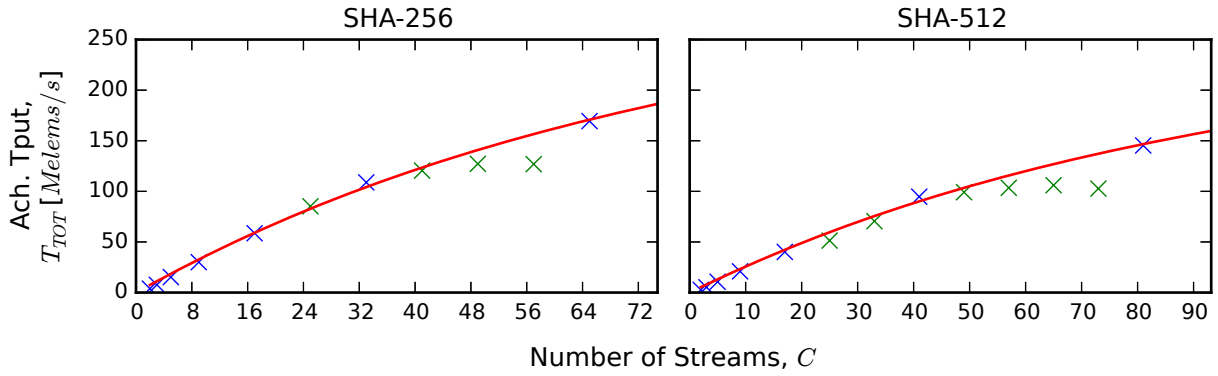



Figure 5.26: Calibrated total achievable throughput plot of the virtualized SHA-2 cryptographic hash application on an FPGA for SHA-256 and SHA-512. $C$ refers to the number of streams supported or the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 1 tool flow run. For SHA-256, the maximum number of pipeline stages is 65. For SHA-512, the maximum number of pipeline stages is 81. Blue $\times$'s are included in the models. Green $\times$'s are excluded from the models.

plus 1). Curve fitting the model in (5.13) across data sets concurrently yields:

$$t_{CLK,SHA256}(C) = \left[\frac{264.6}{C} + 0.66 \cdot (\ln C)^{0.7}\right] \text{ns}, \tag{5.57}$$

$$t_{CLK,SHA512}(C) = \left[\frac{375.1}{C} + 0.78 \cdot (\ln C)^{0.7}\right] \text{ns}. \tag{5.58}$$

Comparing this expression to (5.13) shows that, for SHA-256, we are modeling the curve-fit total combinational logic delay, $k_1$, as 264.6 ns, and $k_2$ as 0.66 ns, and for SHA-512, we are modeling the curve-fit total combinational logic delay, $k_1$, as 375.1 ns, and $k_2$ as 0.78 ns. The plot of the empirical performance with the model prediction of the total achievable throughput for this FPGA design is shown in Figure 5.26. We can observe that the model matches closely to the observed data values.

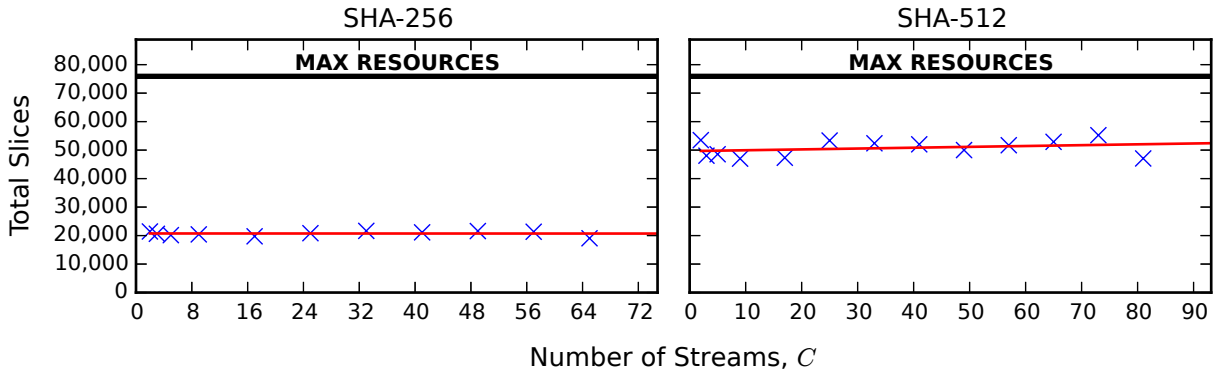



Figure 5.27: Calibrated total slices plot of the virtualized SHA-2 cryptographic hash application on an FPGA for SHA-256 and SHA-512. $C$ refers to the number of streams supported and the pipeline depth of the computation with $N = C$ (i.e., no secondary memory). For each value of $C$, data points were taken from 1 tool flow run. For SHA-256, the maximum number of pipeline stages is 65. For SHA-512, the maximum number of pipeline stages is 81. All data points are included in the models.

### Resource model

For FPGA technology, we curve fit the total slices for SHA-256 and SHA-512 to (5.56) to yield the following models:

$$Slices_{SHA256}(C) = 20{,}737, \tag{5.59}$$

$$Slices_{SHA512}(C) = 49{,}652 + 30 \cdot C. \tag{5.60}$$

We can see in this equation that the fixed number of slices used for SHA-256 is 20,737 with no additional slices per stream, and that the fixed number of slices used for SHA-512 is 49,652 with an additional 30 slices per stream. The plot of the empirical results with the model prediction of the slices is shown in Figure 5.27. We can observe that the slices are approximately constant as a function of $C$, and that the model prediction aligns well with the measured slices data.

## 5.5 Analytic model results

We have application and technology independent queueing model equations, and calibrated clock and resource sub-models for three applications (COS, AES, and SHA) and two technologies (FPGA and ASIC). We are going to use the M/G/1 queueing model equations previously developed in Section 5.3 with the calibrated clock and resource models curve-fitted in Section 5.4 to predict the performance of virtualized logic computations for fine- and coarse-grain contexts for a variety of design goals. We will predict the performance of a theoretical MTJ technology where every logic gate is potentially a pipeline stage by determining the number of logic gate levels needed to implement an application and setting the pipeline depth, $C$, to this number accordingly. We will present results for three design scenarios. First, we will fix $C$ and then optimize for minimum latency. This will emulate an MTJ technology with a fixed pipeline depth or an already implemented hardware design where $C$ has been chosen and is fixed. Second, we will fix $N$, tune $C$, and then optimize again for minimum latency. In this case, the hardware is being designed and $C$ is a tunable design parameter. Third, we will tune $C$, fix $N = C$, and then co-optimize throughput and slices/area for an efficient design. In this case, throughput and slices/area are important but not latency.

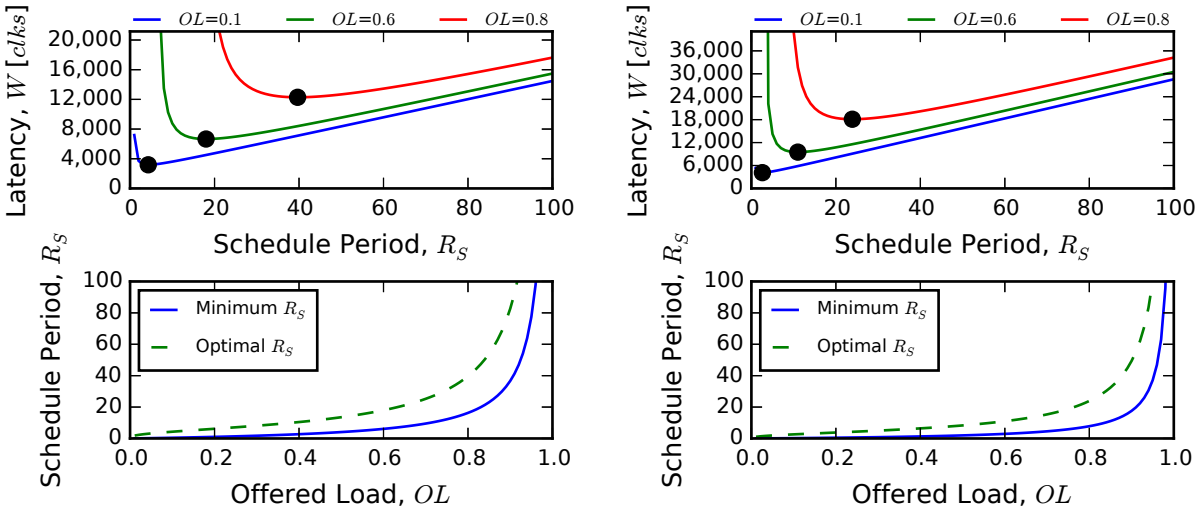
### 5.5.1 Design Scenario 1: Fix $C$, optimize for latency ($W_T$)

For the first design scenario, we are given a circuit, technology, the total number of contexts ($N$), the pipeline depth ($C$), and the cost of a context switch ($S$), and have a varying offered load ($OL$). We can tune the schedule period ($R_S$). We now want to optimize for minimum latency ($W_T$). For this design scenario, we present latency and optimization results for MTJ, FPGA, and ASIC technologies.

The first result we present will be for MTJ technology (magnetologic) which is a natural fit for this design scenario. As a reminder, magnetologic refers to logic circuits built using MTJ devices. Since MTJs are magnetic and are capable of storing information in a magnetic field, then every logic gate in magnetologic has the potential to be a pipeline stage in combinational logic. On this basis, we can then set $C$ equal to the number of logic gate levels in the longest combinational logic path of an application circuit. For this technology, we target the SHA application. For SHA-256, we estimate 491 logic levels. For SHA-512, we estimate 1,025 logic levels. (These estimates were taken from post-route timing summary reports for SHA on a Xilinx Virtex-7 XC7VX485T FPGA part using the Xilinx Vivado 2013.4 tools.)

The results for MTJ technology are shown in Figure 5.28a for SHA-256 and Figure 5.28b for SHA-512. In these results, we assume one output is generated per input (as in, hashes are computed immediately from each block input into SHA). In each figure, there is a latency and optimization plot. In the latency plot, the queueing model predicts mean latency ($W_T$) across a range of schedule periods ($R_S$) and three offered loads ($OL$). Since we are predicting theoretical results for an MTJ technology, $t_{CLK}$ is unknown but is expected to be constant. Therefore, we measure latency in clocks (i.e., setting $t_{CLK} = 1$ which effectively drops $t_{CLK}$ out of the equations). Offered load, as was previously defined in Section 5.2.2 and is repeated here, is the ratio of the aggregate arrival rate (of all streams) to the peak service rate of the system (i.e., when $S = 0$). It evaluates to $N \cdot \lambda$ (where $\lambda$ is in $\frac{\text{elements}}{\text{clock}}$).

Returning to the latency plot, there are three regions of interest. The first region is the high latency on the left which drops steeply. This high latency is due to queueing delay. The beginning of this region marks the minimum $R_S$ needed for a given offered load. As $R_S$ increases, total achievable throughput increases, causing queueing to decrease. The second region is at the knee of the curve. In this region, $R_S$ is optimal and gives minimum latency performance. The third region is to the right where latency gradually increases.



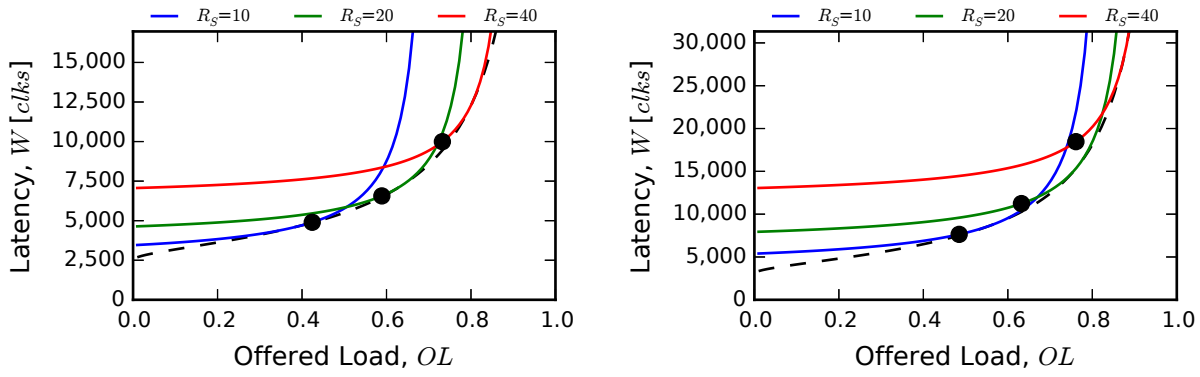

(a) SHA-256 with $C = 491$

(b) SHA-512 with $C = 1{,}025$

Figure 5.28: Analytic latency prediction (top) and optimization (bottom) plots for MTJ technology with the SHA application for design scenario 1. The optimal $R_S$ for minimum latency is indicated by a dot. In these plots, $C$ is calibrated to the logic levels of the SHA-256 and SHA-512 implementations, $N = 2C$, and $S = 2{,}000$.

This increase is due to the wait time incurred during a vacation period by the hierarchical, round-robin schedule as $R_S$ increases. It now takes longer for a data stream to be serviced. We can observe for SHA-256 with an $OL = 0.8$, the minimum latency is about 12,500 clocks, whereas for SHA-512, it is about 18,000 clocks. This larger latency is expected due to the additional logic levels in SHA-512.

In the optimization plot, latency is optimized continuously across a range of offered loads swept from 0 to 1. There are two curves. The solid blue curve shows the minimum $R_S$ needed to service a given offered load (that is, $\rho < 1$). The dashed green curve shows the optimal $R_S$ that minimizes latency at the given offered load. We can see that at low offered load, the optimal $R_S$ is small, then gradually increases until about 0.8 (left) or 0.9 (right), and finally increases steeply.




(a) SHA-256 with $C = 491$

(b) SHA-512 with $C = 1{,}025$

Figure 5.29: Analytic latency prediction plots vs. offered load for MTJ technology with the SHA application for design scenario 1. $C$ is calibrated to the logic levels of the SHA-256 and SHA-512 implementations, $N = 2C$, and $S = 2{,}000$. A 3 dB point is drawn for each curve above the minimum latency to approximate the knee of the curve. The dashed black curve is a lower bound of the optimal latency for $R_S$ optimized at each offered load.

Suppose we choose a value of $R_S$ from the optimization plot. We can then plot the latency for this fixed $R_S$ versus the offered load to show a performance curve across all loads. This is shown in Figure 5.29a for SHA-256 and in Figure 5.29b for SHA-512 for three values of $R_S$. We can see that the latency is initially flat at low offered loads, but then increases steeply at some point. The "knee" of these curves, indicated by a black dot, we can approximate to be 3 dB above the minimum latency of each curve. Beyond the knee, the latency increases sharply. As $R_S$ varies, the latency curve changes in two ways. First, it changes in the value of the latency in the flat part of the curve (low $R_S$ gives low latency). Second, it changes the position of the knee of the curve (high $R_S$ pushes the knee out further, allowing a large range of loads to be handled). This indicates a tradeoff. If we expect the load on the system to be small, we can sacrifice range to get low latency. Otherwise, we can sacrifice low latency for a large range. Optimizing $R_S$ continuously across all offered loads, we can attain the minimum latency for each offered load. This is shown as the dashed black curve, which establishes a lower bound on the latency.

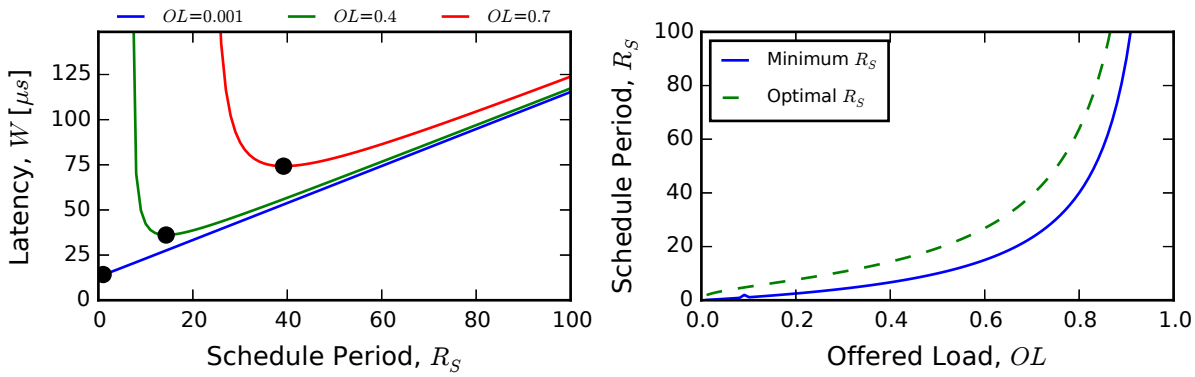



Figure 5.30: Analytic latency prediction (left) and optimization (right) plots for FPGA technology with the COS application for design scenario 1. In these plots, $C = 10$, $N = 100$, $S = 100$, and $N_t = 20$.

Next, we present results for FPGA technology with $t_{CLK}$ calibrated to (5.46) for the COS application. The latency and optimization plots are shown in Figure 5.30 for offered loads of 0.001, 0.4, and 0.7. In these plots, $C = 10$, $N = 100$, $S = 100$, $N_t = 20$ (the number of Taylor series terms approximating the cosine function), and $t_{CLK}$ evaluates to 25.3 ns (which is 39.5 MHz). The general trends observed with MTJ technology are clearly present here as well.

Last, we present results for ASIC technology with $t_{CLK}$ calibrated to (5.53) for the AES application. The latency and optimization plots are shown in Figure 5.31 for offered loads of 0.001, 0.4, and 0.7. In these plots, $C = 10$, $N = 200$, $S = 150$, $N_r = 14$ (the number of cipher rounds), and $t_{CLK}$ evaluates to 19.6 ns (which is 50.9 MHz). The general trends observed with MTJ technology are clearly present here as well.

### 5.5.2 Design Scenario 2: Fix $N$, tune $C$, optimize for latency ($W_T$)

For the second design scenario, we are given a circuit, technology, the total number of contexts ($N$), the cost of a context switch ($S$), and the arrival rate ($\lambda$). We can tune the

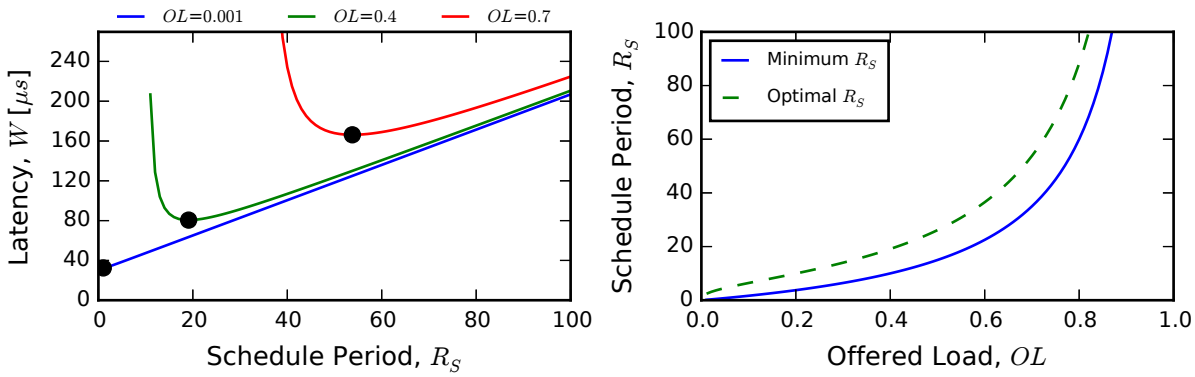
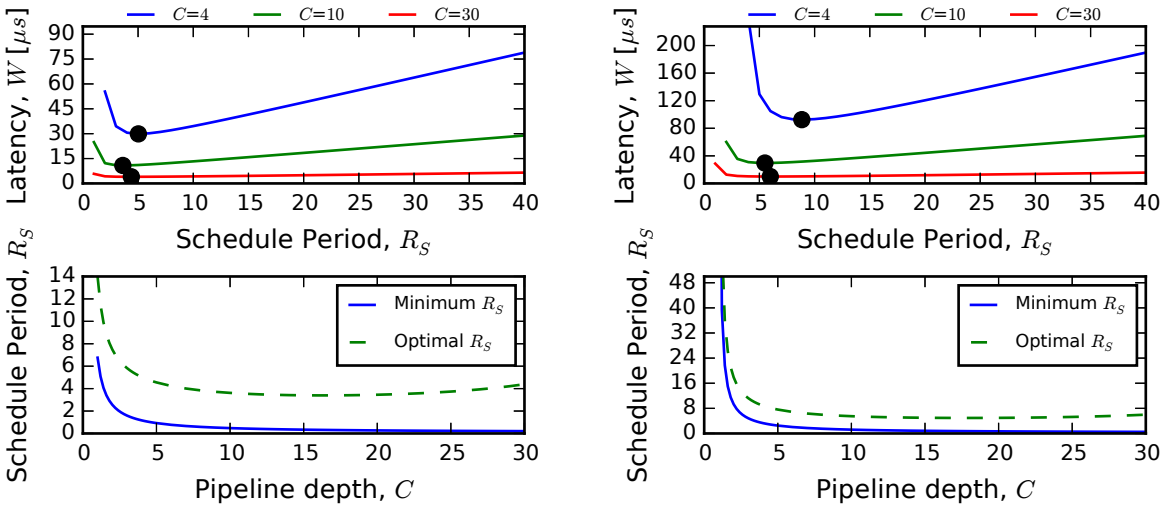



Figure 5.31: Analytic latency prediction (left) and optimization (right) plots for ASIC technology with the AES application for design scenario 1. In these plots, $C = 10$, $N = 200$, $S = 150$, and $N_r = 14$.

pipeline depth ($C$) and the schedule period ($R_S$). We now want to optimize for minimum latency ($W_T$). For this design scenario, we present latency and optimization results for the COS, AES, and SHA applications on FPGA and ASIC technologies.

First, we present results for the COS application in Figure 5.32a for FPGA technology with the clock period, $t_{CLK}$, calibrated to (5.46) and in Figure 5.32b for ASIC technology with $t_{CLK}$ calibrated to (5.47). In each figure, there is a latency and optimization plot with parameters $N = 60$, $S = 10C$, $N_t = 20$ (the number of Taylor series terms approximating the cosine function), and $\lambda = 30 \frac{\text{Kelems}}{\text{s}}$. $t_{CLK}$ depends on the value of $C$.

In the latency plot, the queueing model again predicts mean latency, $W_T$, across a range of schedule periods, $R_S$, however, this time for three values of $C$ (instead of offered load). The shape of the latency plot is the same as before (initially high, drops steeply to an optimal point, then increases gradually). However, we can see that as $C$ increases, overall latency decreases (lower is better). This is due to having more pipeline stages running more contexts concurrently in the hardware, thus parallelizing the computation. Because there are more stages (evenly distributed throughout the combinational logic), stage-to-stage delay is reduced. This improves overall throughput, and therefore reduces latency.



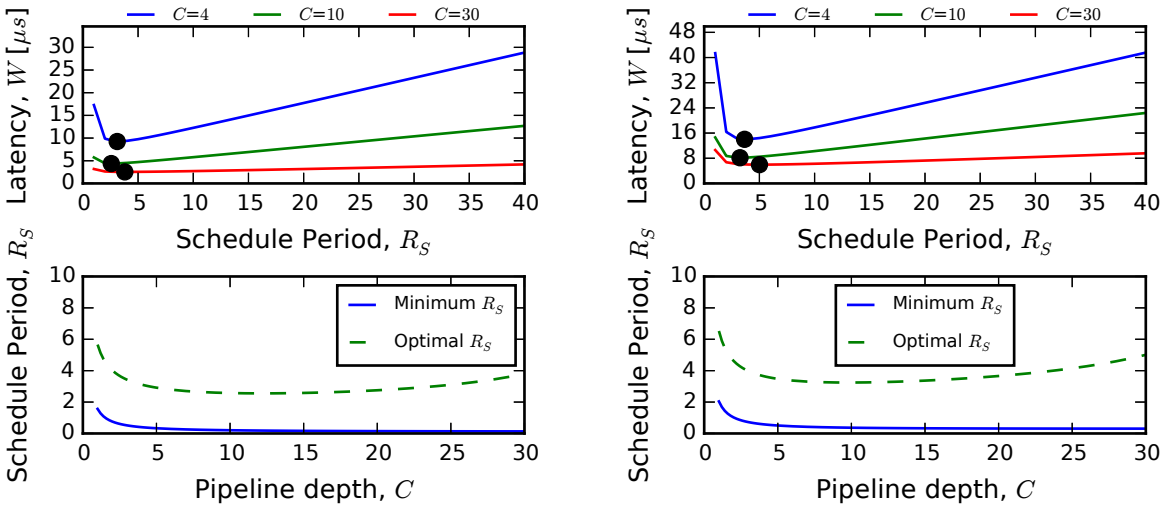

(a) FPGA

(b) ASIC

Figure 5.32: Analytic latency prediction (top) and optimization (bottom) plots for the COS application for design scenario 2. In these plots, $N = 60$, $S = 10C$, $N_t = 20$, and $\lambda = 30\,\frac{\text{Kelems}}{\text{s}}$.

In the optimization plot, the schedule period, $R_S$, is optimized for minimum latency, this time as a function of $C$. We can see that the optimal $R_S$ is initially high at low $C$, then decreases as $C$ increases. This is because when $C$, the number of pipeline stages, increases, the overall throughput also increases. When this happens, $R_S$ no longer needs to be as high to handle the load. We also notice that the optimal $R_S$ starts to curve back up slightly at high $C$. We can understand this by making a simple observation in the latency plot. As $C$ increases in the latency plot, the slope of the region to the right of the optimal $R_S$ point decreases. Since this slope is less pronounced, it causes the optimal $R_S$ to shift to the right (increase) slightly.

Next, we present results for the AES application in Figure 5.33a for FPGA technology with $t_{CLK}$ calibrated to (5.52) and in Figure 5.33b for ASIC technology with $t_{CLK}$ calibrated to (5.53). In these plots, $N = 60$, $S = 10C$, and $\lambda = 30\,\frac{\text{Kelems}}{\text{s}}$. The number of rounds, $N_r$, is set to 14 which is the standard for AES-256. The value of $t_{CLK}$ again depends on the



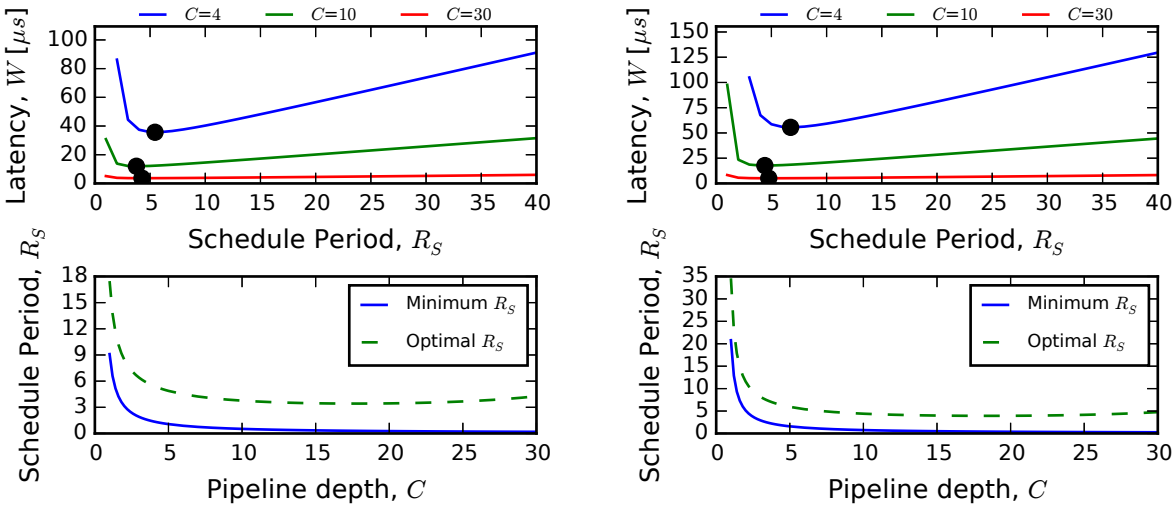

(a) FPGA

(b) ASIC

Figure 5.33: Analytic latency prediction (top) and optimization (bottom) plots for the AES application for design scenario 2. In these plots, $N = 60$, $S = 10C$, $N_r = 14$, and $\lambda = 30\,\frac{\text{Kelems}}{\text{s}}$.

value of $C$. We can see in these figures that the general shape and observations made of the latency and optimization curves remain the same except that the optimal $R_S$ (in the optimization plot) does not appear as high at low $C$.

Last, we present results for the SHA application for FPGA technology in Figure 5.34a for SHA-256 with $t_{CLK}$ calibrated to (5.57) and in Figure 5.34b for SHA-512 with $t_{CLK}$ calibrated to (5.58). In these plots, $N = 60$, $S = 10C$, and $\lambda = 30\,\frac{\text{Kelems}}{\text{s}}$. The value of $t_{CLK}$ again depends on the value of $C$. We can also see in these figures that the general shape and observations made of the latency optimization curves are the same as for the COS application.




(a) SHA-256

(b) SHA-512

Figure 5.34: Analytic latency prediction (top) and optimization (bottom) plots for the SHA application in FPGA technology for design scenario 2. In these plots, $N = 60$, $S = 10C$, and $\lambda = 30\,\frac{\text{Kelems}}{\text{s}}$.

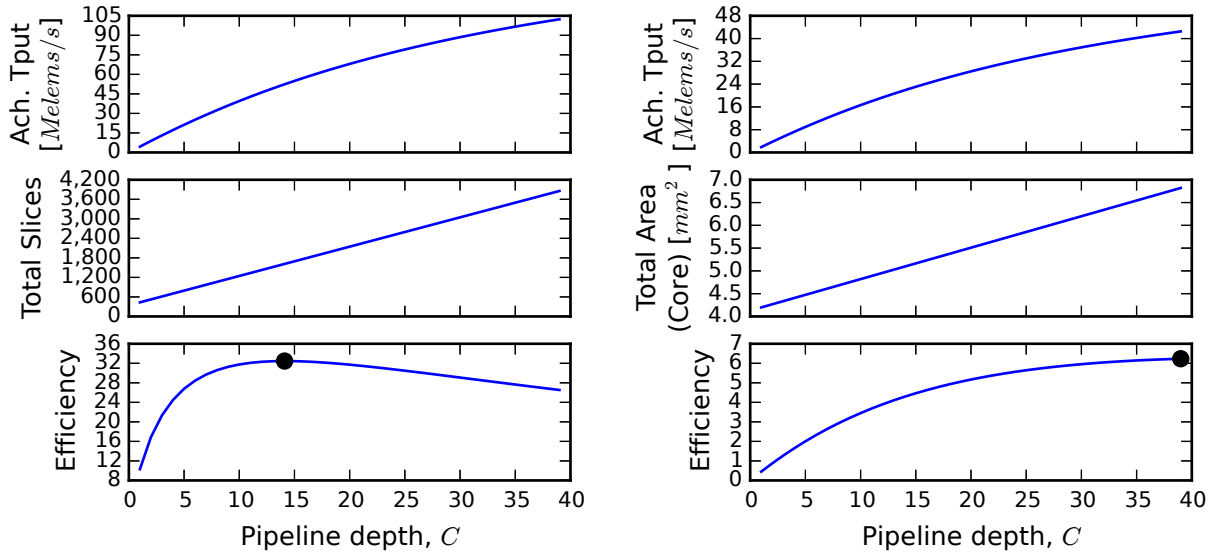
### 5.5.3 Design Scenario 3: Optimize for throughput-slice/area efficiency

For the third design scenario, we are given a circuit and technology and will only be using fine-grain context switching (i.e., no secondary memory). Thus, $N = C$ (total contexts equals the pipeline depth) and $S = 0$ (no context switch cost). We can tune the pipeline depth ($C$). We now want to optimize for the maximum throughput-slice/area efficiency of the design. Throughput-slice/area efficiency is defined as the ratio of the total achievable throughput to the total slices (FPGA) or area (ASIC). A more efficient design is one that has a high throughput and low slice count or circuit area. For this design scenario, we present throughput, slices/area, and efficiency results for the COS, AES, and SHA applications on FPGA and ASIC technologies. $C$ will be constrained up to the maximum number of pipeline stages that are implemented for each application.

First, we present results for the COS application in Figure 5.35a for FPGA technology and in Figure 5.35b for ASIC technology. In each figure, there is a total achievable throughput, total slices/area, and efficiency plot. In these plots, $N_t = 20$ (the number of Taylor series terms approximating the cosine function). We model two parameters in this design scenario: clock period ($t_{CLK}$) and total slices/area. $t_{CLK}$ is calibrated using (5.46, 5.47) and total slices/area is calibrated using (5.48, 5.49). Since $C$ varies, $t_{CLK}$ also varies because it depends on $C$. In the total achievable throughput plot, the queueing model predicts $T_{TOT}$ across a range of pipeline depths, $C$, up to the maximum pipeline depth $C_{max} = 2N_t - 1 = 39$ (for COS). We can see that the throughput increases almost linearly as $C$ increases, but starts to bend over. In the total slices/area plot, the slices/area increases linearly with $C$. We then derive the efficiency plot as the ratio of these two plots. A higher efficiency is better which co-optimizes for high throughput and low slices/area. In Figure 5.35a, we can see that the efficiency is initially low, increases sharply to an optimal point at about $C = 14$, and then gradually decreases with larger $C$. In Figure 5.35b, there is a similar trend, except that the pipeline depth, $C$, reaches its maximum before reaching a peak. Therefore, the optimal point is at $C_{max} = 39$.

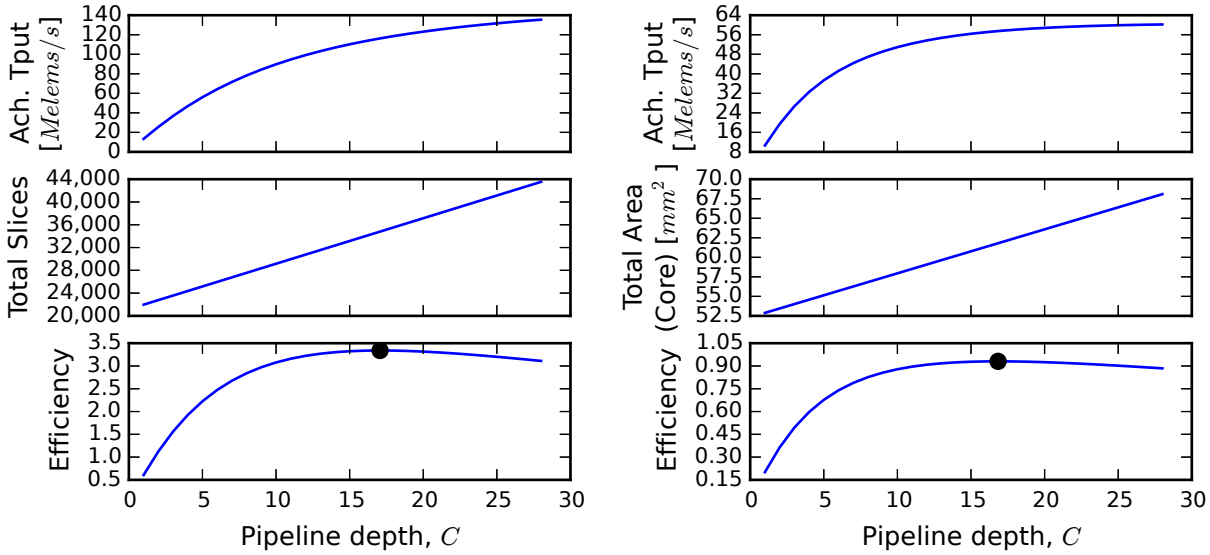
Next, we present results for the AES application in Figure 5.36a for FPGA technology and in Figure 5.36b for ASIC technology. In these plots, the number of rounds, $N_r$, is set to 14 which is the standard for AES-256. $t_{CLK}$ is calibrated using (5.52, 5.53), and total slices/area is calibrated using (5.54, 5.55). The maximum pipeline depth, $C_{max}$, is $2N_r = 28$. We can see that the optimal efficiency in both Figures 5.36a and 5.36b is about $C = 17$.

Last, we present results for the SHA application (implemented only on FPGA technology) in Figure 5.37a for SHA-256 and in Figure 5.37b for SHA-512. $t_{CLK}$ is calibrated using (5.57, 5.58) and total slices are calibrated using (5.59, 5.60). The maximum pipeline depth, $C_{max}$, depends on the number of rounds, $N_r$, for each hash type of the SHA-2 standard. For



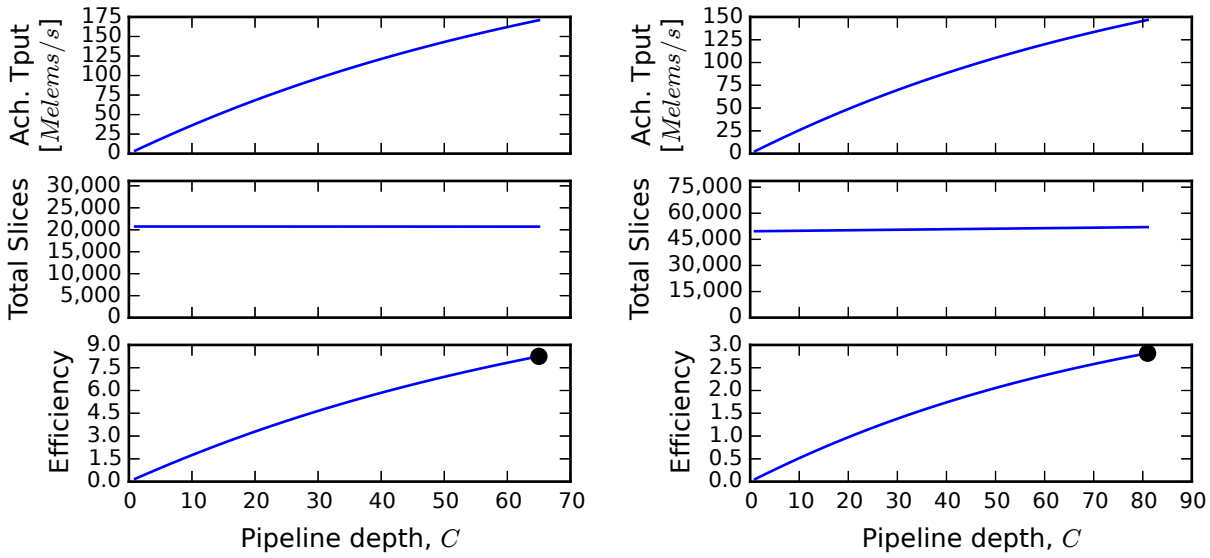

(a) FPGA (b) ASIC

Figure 5.35: Analytic total achievable throughput (top), total slices/area (middle), and efficiency (bottom) plots for the COS application for design scenario 3. $N = C$, $S = 0$, and $N_t = 20$. Efficiency is the ratio of throughput to total slices (left) or area (right).




(a) FPGA (b) ASIC

Figure 5.36: Analytic total achievable throughput (top), total slices/area (middle), and efficiency (bottom) plots for the AES application for design scenario 3. $N = C$, $S = 0$, and $N_r = 14$. Efficiency is the ratio of throughput to total slices (left) or area (right).




(a) SHA-256 (b) SHA-512

Figure 5.37: Analytic total achievable throughput (top), total slices (middle), and efficiency (bottom) plots for the SHA application in FPGA technology for design scenario 3. $N = C$, and $S = 0$. Efficiency is the ratio of throughput to total slices.

SHA-256, there are $N_r = 64$ rounds making $C_{max} = N_r + 1 = 65$, and for SHA-512, there are $N_r = 80$ rounds making $C_{max} = N_r + 1 = 81$. Observing the total slices plot, we can see that the slices are relatively constant, or independent of the pipeline depth, $C$. This is due to the SHA-2 implementation having a single stream interface containing fields (to be hashed) within record-oriented data. Elements from each field are then accessed via a content addressable queue. This logic is independent of $C$. Therefore, the efficiency plot always shows that the optimal pipeline depth for maximum efficiency will be at the same point for maximum throughput which is $C_{max}$.

## 5.6 Summary

We developed virtualized hardware using $C$-slow techniques to effectively utilize hardware and to compute multiple data streams concurrently. The virtualized hardware is pipelined with $C$ pipeline stages which we were able to model to develop a clock period model. Using queueing theory, we modeled the performance of the virtualized hardware using M/D/1 and M/G/1 queueing models to predict total throughput, latency, and queue occupancy. These models were validated via a discrete-event simulation.

We developed three $C$-slow applications: COS, AES, and SHA. Using the clock period model and a resource model, we calibrated these applications using simulation results to the models. The models were shown to fit well to the simulation results.

Last, we used the M/G/1 queueing model to predict the total throughput, latency, and queue occupancy of our three applications for MTJ, FPGA, and ASIC technologies. We showed several ways in which to use the model that allow a designer to optimize the performance of a virtualized circuit.

# Chapter 6

# Conclusions and Future Work

The research described in this dissertation is motivated by the desire to effectively utilize magnetic tunnel junctions (MTJs) in digital systems. Two potential uses of MTJs are clocking and logic. In clocking, a global external magnetic field combined with on-chip MTJs is used as an alternative mechanism for distributing the clock signal across the chip. This requires a read circuit to sense the MTJ and produce the clock output. In logic, MTJs are used to construct logic circuits, also called magnetologic. MTJs are written via a current or a magnetic field and are accessed via a resistance. This therefore requires support circuitry in order to read the MTJ and write to a downstream device. For both of these uses, a read circuit is needed to determine the state of the MTJ. For this purpose, we developed a resistance-to-voltage (R2V) read circuit to sense the MTJ resistance and produce a logic voltage output.

Magnetologic circuits have the potential to act as state in the pipeline stage of a logic circuit. This is because each MTJ device acts as a latch to whatever data is written to it. This consequently means that magnetologic circuits have the potential to become very deeply-pipelined. Since most applications are sequential (i.e., they have feedback), this can make digital design very challenging. We want to exploit the latching property of

magnetologic circuits. Therefore, we investigated virtualization of hardware as a way to more fully utilize deeply-pipelined circuits. Virtualization involves context switching the hardware logic to allow computing multiple data streams in the same logic circuit. While applicable to magnetologic using MTJs, virtualization is also applicable to traditional logic technologies like CMOS. Our virtualization investigation targets MTJs, FPGAs, and ASICs.

## 6.1 Conclusions

**Resistance-to-voltage (R2V) read circuit**

We designed, analyzed, simulated, laid out, fabricated, and tested an R2V read circuit. In the analysis, we developed a noise model for a current-mode read circuit designed with a basic current conveyor to predict its noise performance at the output. The model relates several circuit parameters to circuit performance. We learned from the model that the input node is relatively insensitive to node capacitance. This allows the input node to have high capacitance without degradation of overall circuit performance.

From the model, we developed design guidance for tuning circuit performance. This tells us that the output noise current depends strongly on transistor length and $R_{mtj}$, and weakly on transistor width and $V_{bias}$. Therefore, to decrease the output noise, the transistor length and $R_{mtj}$ (if tunable), which yield the greatest effect, should be increased.

We analyzed several current conveyor structures and developed design equations for sizing their transistors. We learned that linearity is improved by adding cascode structures and/or increasing transistor length (while preserving the shape factor). The tradeoffs are that (1) cascode structures increase the number of transistors in the design and require wider

transistors in order to keep operating in the saturation region; and (2) increasing transistor length (while preserving the shape factor) increases the transistor size in quadrature.

We designed the R2V read circuit that was fabricated using a P-cascode current conveyor with minimum length transistors, a P-cascode current comparator, and an output buffer. This design balances tradeoffs in linearity, area, and speed.

We simulated the R2V read circuit and characterized it in terms of area, transient response, power, and jitter. We found that when the process scales down to smaller dimensions, the area decreases, rise/fall times decrease, propagation times decrease, maximum frequency increases, power consumption decreases, and jitter decreases. Further, we found that the performance bottleneck of the read circuit is in the current comparator. This indicates that read circuit performance could be improved with a faster current comparator circuit. Also, we observed that the current comparator current need not be the same current as the current conveyor current. Rather, it could be scaled down to save power and area.

We designed and fabricated a prototype test chip in the 3 metal 2 poly 0.5 $\mu$m process for testing the R2V read circuit and to evaluate the feasibility of magnetic global clocking. However, due to being unable to attain MTJs, magnetic global clocking was not tested.

We experimentally tested the R2V read circuit on the prototype test chip to evaluate its functionality and performance. Using a clocked low/high resistor pair, the read circuit is shown to correctly detect the input resistance and produce the desired square wave output for MCLK. We observed second-order effects in the $V_{mtj}$ node voltage that caused an offset, from $V_{mtj}$ to $V_{bias}$, to appear on this node in the order of 17 to 51 mV. At low $V_{bias}$, such as 50 mV, this offset is significant because it is of a similar order to $V_{bias}$. This consequently causes the $I_{out}$ current to vary significantly from its nominal value. Between quadrants, we observed significant variation between $V_{mtj}$ node offsets, causing significant variation in the

ranges of low/high $I_{out}$ currents at low $V_{bias}$. This prevents a single common threshold to be selected between all the quadrants and requires that each quadrant be set individually. We tested the read circuit for its maximum operating frequency and found it to be about 48 MHz for internal resistors with $V_{bias} = 0.1\,\mathrm{V}$.

We empirically measured the actual values of the internal resistors to be within $466 \pm 6\,\Omega$ and $787 \pm 11\,\Omega$ across $V_{bias}$ and all quadrants. These values are less than their nominal values of $500\,\Omega$ and $1\,\mathrm{k}\Omega$, which results in higher output current than expected. The variation in the internal resistors is small, making the resistor values close to constant across $V_{bias}$ and all quadrants.

We tested the linearity of the read circuit across a range of $V_{bias}$ and for multiple resistors. The read circuit remains linear up to a maximum $V_{bias}$ and then plateaus in the $I_{out}$ current. In the linear region, the transistors operate in saturation and $V_{mtj}$ correctly follows $V_{bias}$. This in turn produces linear $I_{out}$ current. The circuit is designed to operate correctly at $R_{mtj} = 500\,\Omega$ and $V_{bias} = 0.1\,\mathrm{V}$, however, it is shown to operate correctly up to twice the range of $V_{bias} = 0.2\,\mathrm{V}$ for the same resistor value.

We established current limits for the read circuit that the threshold current must be between. These are defined to be the values of the $I_{out}$ current measured for the low/high resistors. We then measured statically a range of stable threshold currents over which the current comparator operates correctly at DC. We last tested the dynamic operation of the read circuit (by clocking the input resistance between low/high values) and determined a range of parameters over which the read circuit operates correctly. For internal resistors, this range was wide, but for external resistors, it was narrow or non-existent. Outside of the stable range, we observed that the MCLK output oscillates when $I_{th}$ is close to the current limits.

**Virtualization of hardware logic circuits**

We investigated virtualization techniques for deeply-pipelined hardware logic circuits. We developed a clock period model and empirically validated it via Monte Carlo simulations. To better fit the model to the empirical result, a power term $p$ was added to bend the shape of the model curve. The shape depends on the random distribution and, for some distributions, their parameters as well. For a Gaussian distribution, $p$ is $\approx 0.7$ and is constant regardless of the parameters of the distribution.

We developed M/D/1 and M/G/1 queueing models of the performance of virtualized hardware with secondary memory using a fixed, hierarchical, round-robin schedule. These models predict average throughput, latency, and queue occupancy in the system.

We discovered when applying the $C$-slow technique (i.e., replacing every register with $C$ registers and then retiming) to virtualize a hardware circuit that the retiming tools did not do a good job at evenly distributing logic throughout the pipeline stages. Evenly distributed pipeline stages are necessary to improve the clock frequency to maximize total throughput performance. To accomplish this, manual pipelining was used instead.

We developed three $C$-slow applications and calibrated them to the clock model and a resource model for FPGA and ASIC technologies. Both models show good correspondence to the empirical results. The clock model is input to the queueing models. The resource model is combined with the queueing models to evaluate the efficiency of performance to resource usage.

Using the M/G/1 model, we predict throughput, latency, and resource usage for MTJ, FPGA, and ASIC technologies. We show three design scenarios illustrating ways in which to use the model. The first two predict latency. We are able to determine an optimal schedule

in each that minimizes the latency in each design scenario. The third design scenario predicts throughput and total slices/area. We are able to determine an optimal pipeline depth that maximizes throughput-slice/area efficiency.

## 6.2 Future work

Since we were unable to attain MTJs for experimentation, future work is to attain actual MTJ devices, wire bond them to the prototype test chip, and measure the effectiveness of magnetic global clocking in the presence of an oscillating magnetic field using the global clocking circuits on the chip.

During experimentation of the read circuit, we observed oscillations near threshold and offset in the $V_{mtj}$ node voltage. There were also aspects of the read circuit that were not fully explored such as the impact of load capacitance on the input node. Further investigation is needed on the properties of the read circuit. Future work is to determine the cause of the oscillations observed near threshold, investigate the use of hysteresis in the current comparator as a way to reduce the oscillations, investigate the separation (offset) between $V_{mtj}$ and $V_{bias}$ both within a quadrant and across quadrants, determine the impact of load capacitance on the input node, and from these determine ways in which the read circuit can be improved.

In hardware virtualization, we explored applying the $C$-slow technique to a hardware circuit to improve performance and utilize deeply-pipelined logic. Our investigation primarily used a parallel interface (i.e., streams are in parallel at the input and output) and a fixed schedule, and explored the use of secondary memory in the development of a queueing model.

Future work is to expand the set of context switch designs which includes supporting secondary memory, adding flow control in the data stream, and context switching using dynamic schedules; and to develop queueing models for these designs. In the SHA-2 application, an upfront content addressable queue was used to buffer the input data stream. Future work is to model the content addressable queue and its workload, and from the model determine how large it should be sized. Last, in the queueing model, we assume the arrival process is exponential. This may not be the case in many real world applications. Future work is to evaluate this assumption and determine its impact on the accuracy of the queueing models.

# References


[1] M. Durlam, P. Naji, A. Omair, M. DeHerrera, J. Calder, J. Slaughter, B. Engel, N. Rizzo, G. Grynkewich, B. Butcher, C. Tracy, K. Smith, K. Kyler, J. Ren, J. Molla, W. Feil, R. Williams, and S. Tehrani, "A 1-Mbit MRAM based on 1T1MTJ bit cell integrated with copper interconnects," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 5, pp. 769–773, May 2003.

[2] W. Zhao, E. Belhaire, C. Chappert, B. Dieny, and G. Prenat, "TAS-MRAM-based low-power high-speed runtime reconfiguration (RTR) FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 2, pp. 1–19, 2009.

[3] S. Lee, G. Lee, H. Lee, S. Lee, and H. Shin, "Design of reconfigurable logic circuits based on Single-Layer Magnetic-Tunnel-Junction elements," *Japanese Journal of Applied Physics*, vol. 47, pp. 3264–3268, Apr. 2008. [Online]. Available: http://jjap.ipap.jp/link?JJAP/47/3264/

[4] J. Slaughter, "Materials for magnetoresistive random access memory," *Annual Review of Materials Research*, vol. 39, no. 1, pp. 277–296, Apr. 2009.

[5] C. Lin, S. Kang, Y. Wang, K. Lee, X. Zhu, W. Chen, X. Li, W. Hsu, Y. Kao, M. Liu *et al.*, "45nm low power CMOS logic compatible embedded STT MRAM utilizing a reverse-connection 1T/1MTJ cell," in *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2009, pp. 1–4.

[6] W. Reohr, H. Honigschmid, R. Robertazzi, D. Gogl, F. Pesavento, S. Lammers, K. Lewis, C. Arndt, Y. Lu, H. Viehmann, R. Scheuerlein, L.-K. Wang, P. Trouilloud, S. Parkin, W. Gallagher, and G. Muller, "Memories of tomorrow," *IEEE Circuits and Devices Magazine*, vol. 18, no. 5, pp. 17–27, Sep. 2002.

[7] J. Åkerman, "Toward a universal memory," *Science*, vol. 308, no. 5721, pp. 508–510, 2005. [Online]. Available: http://www.sciencemag.org/content/308/5721/508.short

[8] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," in *DAC '08: Proceedings of the 45th Annual Design Automation Conference*. New York, NY, USA: ACM, 2008, pp. 554–559.

[9] M. Durlam, P. Naji, M. DeHerrera, S. Tehrani, G. Kerszykowski, and K. Kyler, "Non-volatile RAM based on magnetic tunnel junction elements," in *IEEE International Solid-State Circuits Conference. Digest of Technical Papers.*, 2000, pp. 130–131.

[10] R. Takemura, "Low-power NV-RAM," in *Green Computing with Emerging Memory*. Springer, 2013, pp. 111–140.

[11] Everspin Technologies, Inc. [Online]. Available: http://www.everspin.com

[12] W. C. Black and B. Das, "Programmable logic using giant-magnetoresistance and spin-dependent tunneling devices (invited)," *Journal of Applied Physics*, vol. 87, no. 9, pp. 6674–6679, May 2000.

[13] N. Ranganathan and N. Jouppi, "Evaluating the potential of future on-chip clock distribution using optical interconnects," *Hewlett-Packard Development Company, Tech. Rep. HPL-2007-163, Oct*, 2007. [Online]. Available: http://www.hpl.hp.com/techreports/2007/HPL-2007-163.html

[14] M. Hall, A. Jander, R. D. Chamberlain, and P. Dhagat, "Globally clocked magnetic logic circuits," in *Digest of Papers Int'l Conf. on Magnetics*, May 2009.

[15] J. Goodman, F. Leonberger, S.-Y. Kung, and R. Athale, "Optical interconnections for VLSI systems," *Proceedings of the IEEE*, vol. 72, no. 7, pp. 850–866, Jul. 1984.

[16] A. Ney, C. Pampuch, R. Koch, and K. H. Ploog, "Programmable computing with a single magnetoresistive element," *Nature*, vol. 425, no. 6957, pp. 485–487, Oct. 2003. [Online]. Available: http://dx.doi.org/10.1038/nature02014

[17] S. Lee, S. Choa, S. Lee, and H. Shin, "Magneto-logic device based on a single-layer magnetic tunnel junction," *IEEE Transactions on Electron Devices*, vol. 54, no. 8, pp. 2040–2044, Aug. 2007.

[18] S. Lee, S. Seo, S. Lee, and H. Shin, "A full adder design using serially connected single-layer magnetic tunnel junction elements," *IEEE Transactions on Electron Devices*, vol. 55, no. 3, pp. 890–895, Mar. 2008.

[19] B. Buford, A. Jander, and P. Dhagat, "Digital logic using 3-terminal spin transfer torque devices," *Arxiv preprint arXiv:1101.3222*, 2011.

[20] D. Suzuki, M. Natsui, S. Ikeda, H. Hasegawa, K. Miura, J. Hayakawa, T. Endoh, H. Ohno, and T. Hanyu, "Fabrication of a nonvolatile lookup-table circuit chip using magneto/semiconductor-hybrid structure for an immediate-power-up field programmable gate array," in *Symposium on VLSI Circuits*, Jun. 2009, pp. 80–81.

[21] C. Leiserson and J. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5–35, Jun. 1991.

[22] D. M. Tullsen *et al.*, "Exploiting choice: instruction fetch and issue on an implementable simultaneous multithreading processor," in *Proc. of 23rd Int'l Symp. on Computer Architecture*, May 1996, pp. 191–202.

[23] L. Ma, K. Agrawal, and R. D. Chamberlain, "A memory access model for highly-threaded many-core architectures," *Future Generation Computer Systems*, vol. 30, pp. 202–215, Jan. 2014.

[24] L. Ma, K. Agrawal, and R. D. Chamberlain, "Analysis of classic algorithms on GPUs," in *Proc. of the 12th ACM/IEEE Int'l Conf. on High Performance Computing and Simulation (HPCS)*, 2014.

[25] M. J. Hall, V. Gruev, and R. D. Chamberlain, "Noise analysis of a current-mode read circuit for sensing magnetic tunnel junction resistance," in *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2011.

[26] ——, "Performance of a resistance-to-voltage read circuit for sensing magnetic tunnel junctions," in *2012 IEEE 55th International Midwest Symp. on Circuits and Syst. (MWSCAS)*, Aug. 2012, pp. 639–642. [Online]. Available: http://dx.doi.org/10.1109/MWSCAS.2012.6292101

[27] M. J. Hall and R. D. Chamberlain, "Performance modeling of virtualized custom logic computations," in *Proc. of 24th ACM Int'l Great Lakes Symposium on VLSI*, 2014.

[28] ——, "Performance modeling of virtualized custom logic computations," in *IEEE 25th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Jun. 2014, pp. 72–73.

[29] W. Zhao, C. Chappert, V. Javerliac, and J.-P. Noziere, "High speed, high stability and low power sensing amplifier for MTJ/CMOS hybrid logic circuits," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 3784–3787, Oct. 2009.

[30] P. Braganca, J. Katine, N. Emley, D. Mauri, J. Childress, P. Rice, E. Delenia, D. Ralph, and R. Buhrman, "A three-terminal approach to developing spin-torque written magnetic random access memory cells," *IEEE Transactions on Nanotechnology*, vol. 8, no. 2, pp. 190–195, Mar. 2009.

[31] P. Naji, M. Durlam, S. Tehrani, J. Calder, and M. DeHerrera, "A 256 kb 3.0 V 1T1MTJ nonvolatile magnetoresistive RAM," in *IEEE International Solid-State Circuits Conference. Digest of Technical Papers.*, 2001, pp. 122–123, 438.

[32] E. K. S. Au, W.-H. Ki, W. H. Mow, S. T. Hung, and C. Y. Wong, "A novel current-mode sensing scheme for magnetic tunnel junction MRAM," *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 483–488, Mar. 2004.

[33] S.-M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, 3rd ed. McGraw-Hill, 2003.

[34] K. C. Smith and A. Sedra, "The current conveyor–a new circuit building block," *Proc. IEEE (Letters)*, vol. 56, pp. 1368–1369, Aug. 1968.

[35] E. Bruun, "Class AB CMOS first-generation current conveyor," *Electronics Letters*, vol. 31, no. 6, pp. 422–423, Mar. 1995.

[36] A. Ismail and A. Soliman, "Wideband CMOS current conveyor," *Electronics Letters*, vol. 34, no. 25, pp. 2368–2369, Dec. 1998.

[37] B. Razavi, *Design of Analog CMOS Integrated Circuits*. New York, NY, USA: McGraw-Hill, Inc., 2001.

[38] H. Meng, J. Wang, and J.-P. Wang, "A spintronics full adder for magnetic CPU," *IEEE Electron Device Letters*, vol. 26, no. 6, pp. 360–362, Jun. 2005.

[39] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2010, pp. 371–382. [Online]. Available: http://doi.acm.org/10.1145/1815961.1816012

[40] N. Weaver, Y. Markovskiy, Y. Patel, and J. Wawrzynek, "Post-placement C-slow retiming for the Xilinx Virtex FPGA," in *Proc. of 11th Int'l Symp. on Field Programmable Gate Arrays*, 2003, pp. 185–194.

[41] M. Su, L. Zhou, and C.-J. Shi, "Maximizing the throughput-area efficiency of fully-parallel low-density parity-check decoding with C-slow retiming and asynchronous deep pipelining," in *Proc. of 25th Int'l Conf. on Computer Design*, Oct. 2007, pp. 636–643.

[42] M. A. Akram, A. Khan, and M. M. Sarfaraz, "C-slow technique vs multiprocessor in designing low area customized instruction set processor for embedded applications," *International Journal of Computer Applications*, vol. 36, no. 7, 2012. [Online]. Available: http://arxiv.org/abs/1204.1179

[43] C. Plessl and M. Platzner, "Virtualization of hardware – Introduction and survey," in *Proc. of 4th Int'l Conf. on Engineering of Reconfigurable Systems and Algortihms*, 2004, pp. 63–69.

[44] K. K. Chuang, "A virtualized quality of service packet scheduler accelerator," Master's thesis, Georgia Institute of Technology, Aug. 2008.

[45] S. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. Taylor, "PipeRench: a reconfigurable architecture and compiler," *Computer*, vol. 33, no. 4, pp. 70–77, Apr. 2000.

[46] M. Kendall and W. Buckland, *A Dictionary of Statistical Terms*. Edinburgh and London: Published for the International Statistical Institute by Oliver & Boyd, Ltd., 1957.

[47] R. Jain, *The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.* New York: John Wiley & Sons, Inc., 1991.

[48] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, Inc., 1992.

[49] E. Bruun, "Analysis of the noise characteristics of CMOS current conveyors," *Analog Integrated Circuits and Signal Processing*, vol. 12, pp. 71–78, 1997.

[50] P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, 2nd ed. Oxford Univ. Press, 2002.

[51] D. Freitas and K. Current, "CMOS current comparator circuit," *Electronics Letters*, vol. 19, no. 17, pp. 695–697, 18 1983.

[52] L. Chen, B. Shi, and C. Lu, "A robust high-speed and low-power CMOS current comparator circuit," in *The IEEE Asia-Pacific Conference on Circuits and Systems*, 2000, pp. 174–177.

[53] Opal Kelly XEM6010. [Online]. Available: https://www.opalkelly.com/products/xem6010

[54] A. Canis, J. H. Anderson, and S. D. Brown, "Multi-pumping for resource reduction in FPGA high-level synthesis," in *Proc. of Conf. on Design, Automation and Test in Europe*, 2013, pp. 194–197. [Online]. Available: http://dl.acm.org/citation.cfm?id=2485288.2485338

[55] B. C. Arnold and N. Balakrishnan, *Relations, Bounds, and Approximations for Order Statistics.* Springer-Verlag, 1989.

[56] J. Jensen, "Sur les fonctions convexes et les inégalités entre les valeurs moyennes," *Acta Mathematica*, vol. 30, no. 1, pp. 175–193, 1906. [Online]. Available: http://dx.doi.org/10.1007/BF02418571

[57] C. E. Bonferroni, *Teoria statistica delle classi e calcolo delle probabilita.* Libreria internazionale Seeber, 1936.

[58] L. Kleinrock, *Queueing Theory, Volume 1.* Wiley-Interscience, 1975.

[59] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical & Computer Engineers*, 2nd ed. John Wiley & Sons, Inc., May 2004.

[60] NIST, "FIPS-197: Advanced Encryption Standard," *Federal Information Processing Standards (FIPS) Publications*, Nov. 2001.

[61] P. R. Chodowiec, "Comparison of the hardware performance of the AES candidates using reconfigurable hardware," Master's thesis, George Mason University, 2002. [Online]. Available: http://bass.gmu.edu/reports/Pawel_Chodowiec_MS_Thesis.pdf

[62] NIST, "FIPS 180-4: Secure Hash Standard (SHS)," *Federal Information Processing Standards (FIPS) Publications*, Mar. 2012.

# Appendix A

# Read Circuit Design Guidance Validation

The design guidance presented in Section 3.1.4 is derived based on a $2^k$ factorial experimental design [47] for the noise analysis of the basic current conveyor circuit in Section 3.1. The experimental design is a statistical technique that analyzes a system containing $k$ factors with 2 levels per factor and determines variations in the output metrics due to each factor and their interactions.

In the experimental design, we have five factors, listed in Table A.1. These factors are input parameters $W$, $L$, $R_{mtj}$, $C_{mtj}$, and $V_{bias}$ from the noise analysis and are assigned letters A – E. For each factor, there are two levels such that they are a multiple of two. $W$ and $L$ are the width and length of the transistors in the current conveyor circuit. $R_{mtj}$ is the MTJ resistance. $C_{mtj}$ is the additional capacitance on the MTJ node. And, $V_{bias}$ is the bias voltage set over the MTJ.

We choose four metrics or output characteristics, listed in Table A.2, that a designer may be interested in tuning. These metrics include $f_{p1}$, $R_3$, $C_{mtj,dp,min}$, and $\sqrt{\overline{I^2_{out,tot,n}}}$. $f_{p1}$ is the dominant pole frequency of the circuit which contributes the most to the overall circuit

Table A.1: Factors used in the $2^k$ factorial experimental design.

| Factor | Input Parameter | Description |
|---|---|---|
| A | $W$ | Transistor width |
| B | $L$ | Transistor length |
| C | $R_{mtj}$ | MTJ resistance |
| D | $C_{mtj}$ | MTJ node capacitance |
| E | $V_{bias}$ | Bias voltage over MTJ |

Table A.2: Metrics used in the $2^k$ factorial experimental design.

| Metric | Description |
|---|---|
| $f_{p1}$ | Dominant pole frequency of circuit |
| $R_3$ | Node 3 resistance |
| $C_{mtj,dp,min}$ | Minimum MTJ capacitance for a dominant pole |
| $\sqrt{\overline{I^2_{out,tot,n}}}$ | Total integrated output noise |

bandwidth. $R_3$ is the equivalent resistance of node 3. $C_{mtj,dp,min}$ is the minimum capacitance needed to form a dominant pole on node 3 containing the MTJ, and is a function of $f_{p1}$ and $R_3$. And, $\sqrt{\overline{I^2_{out,tot,n}}}$ is the total integrated output noise of the circuit.

Applying the experimental design technique, results are shown in Figures A.1 through A.4 for each output metric. Effects are along the x-axis and represent the effect of each factor and their interactions on the output metric. There are two subplots. The top subplot shows the quantity of the effects with "I" representing the mean quantity of all effects and all others being the difference from the mean effect. The bottom subplot shows the percent variation explained by each effect. (Note, percent variation is meaningless for mean "I".)

Design guidance is determined by observing the effects with the greatest percent variation and determining their trend (increasing or decreasing) by the sign of the quantity of the

effect. For example, we can observe in Figure A.1 for output characteristic $f_{p1}$ that factors B and E exhibit the greatest percent variation in the output. B is the transistor length which decreases $f_{p1}$ when increased. E is the $V_{bias}$ voltage which increases $f_{p1}$ when increased.

The relative dependence of output characteristics to input factors can be estimated empirically by observing the change in output due to a change in input, and then fitting this to power $p$ in the equation $y_2 = y_1 \left(\frac{x_2}{x_1}\right)^p$ where $x_2/x_1$ is the ratio change in the input parameter, $y_2/y_1$ is the ratio change in the output characteristic, and $p$ is a power term that determines the strength of the dependence of the output characteristic to the input factor. If $|p|$ is 1, then the ratio change at the input is the same ratio change at the output. If $|p|$ is greater than one, then the input parameter has a stronger than linear effect on the output. If $|p|$ is less than one, then the output is not affected as much by the change in input. If $p$ is positive, this indicates a direct trend between the input factor and the output characteristic. In contrast, if $p$ is negative, this indicates an inverse trend.

A summary of the relative dependence of output characteristics to input factors is shown in Table A.3. (Note, input parameter $C_{mtj}$ is not shown because its effect is negligible.) The values in the table are estimated values of power $p$. To make an observation from the table, we can see that for output circuit characteristics $f_{p1}$ and $C_{mtj,dp,min}$, the transistor length, $L$, has nearly a square relationship to them. In contrast, for $V_{bias}$, it is about linear.

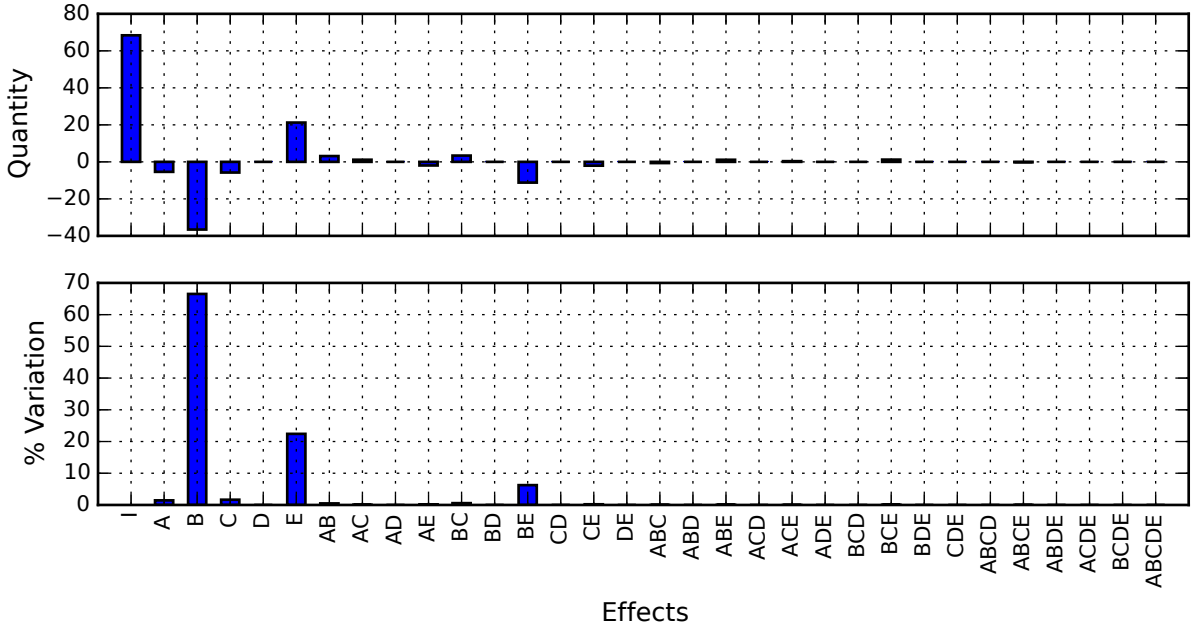



Figure A.1: $2^k$ factorial experimental design for output characteristic $f_{p1}$.

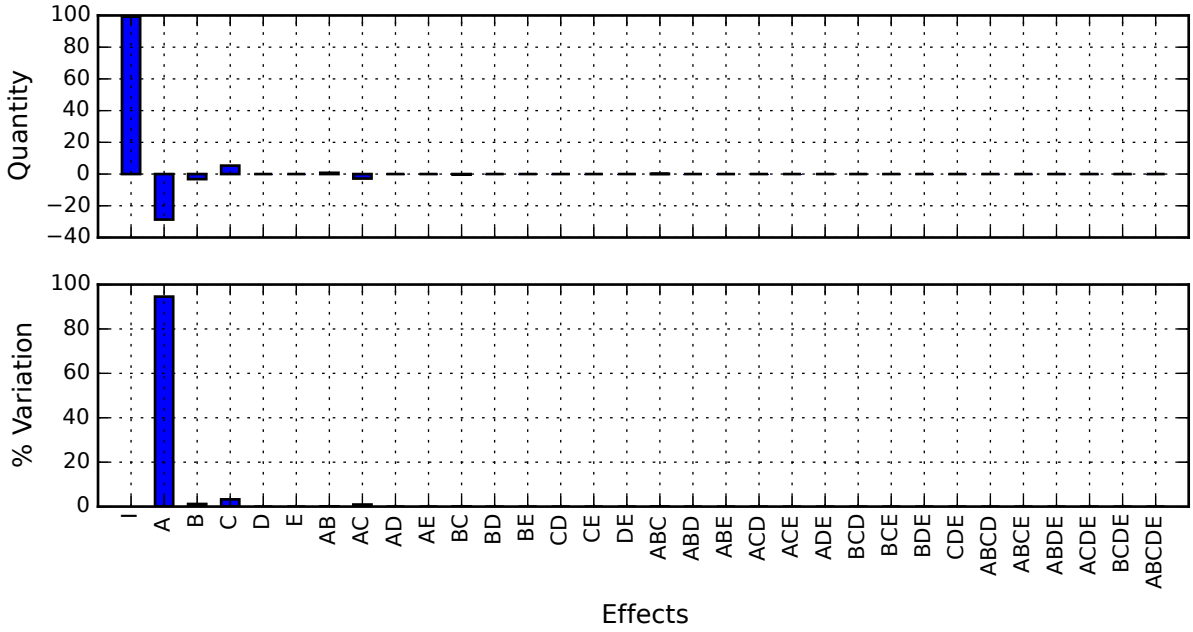



Figure A.2: $2^k$ factorial experimental design for output characteristic $R_3$.

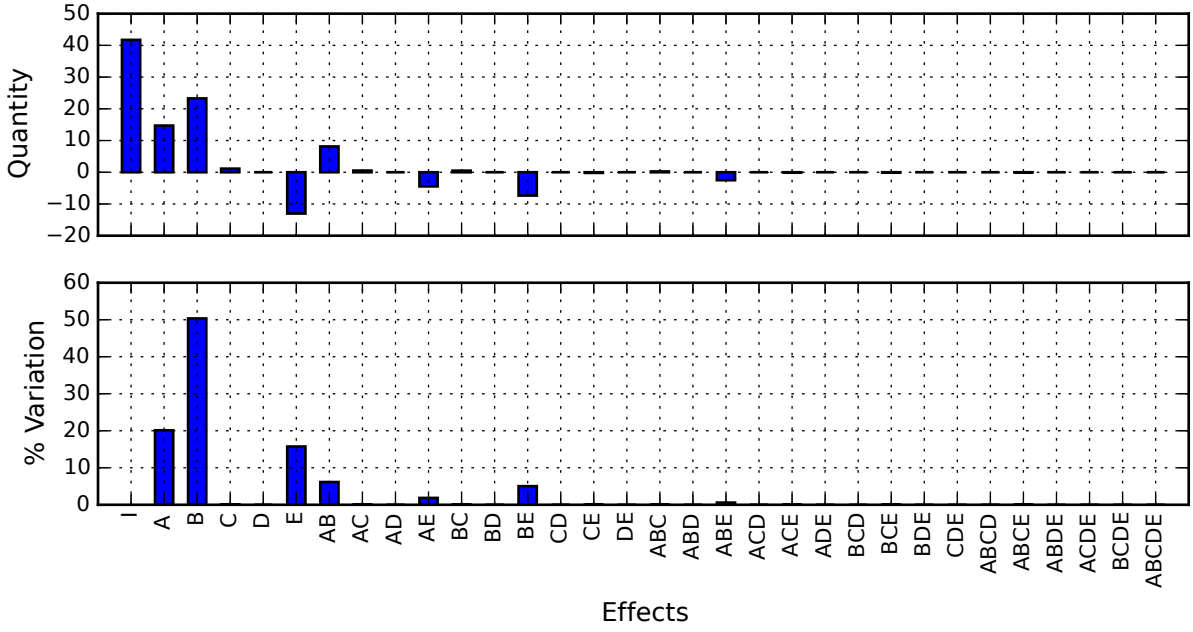



Figure A.3: $2^k$ factorial experimental design for output characteristic $C_{mtj,dp,min}$.

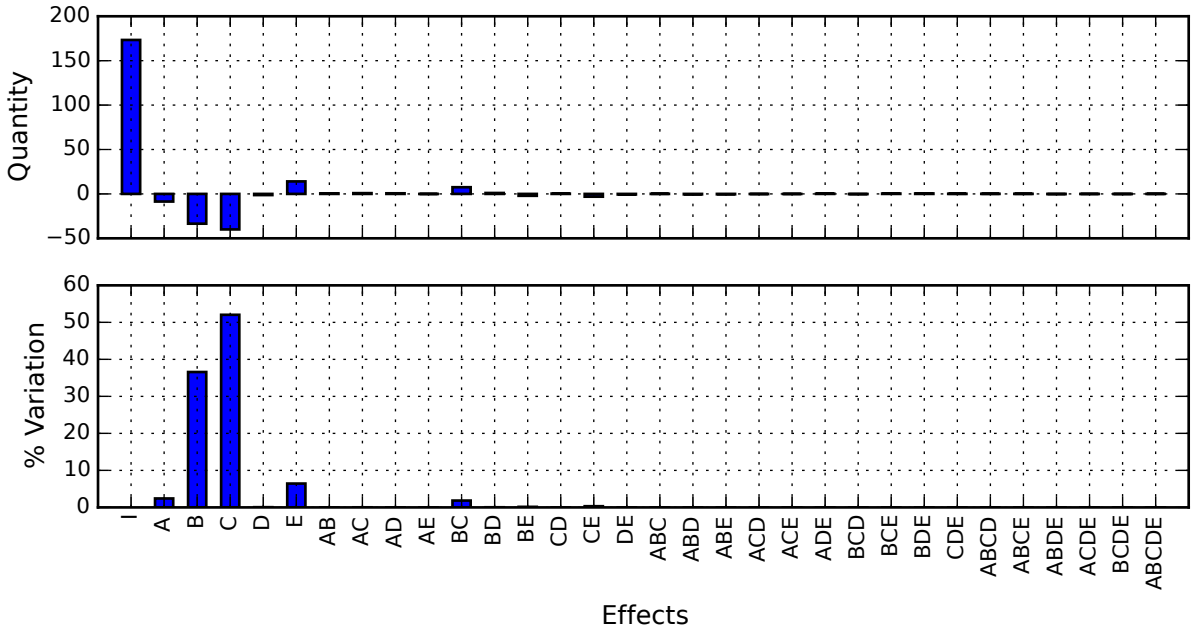



Figure A.4: $2^k$ factorial experimental design for output characteristic $\sqrt{\overline{I^2_{out,tot,n}}}$.

Table A.3: Summary of the relative dependence of output characteristics to input factors. The values in the table are estimated powers, $p$, in the equation $y_2 = y_1 \left(\frac{x_2}{x_1}\right)^p$ where $x_2/x_1$ is the ratio change in an input parameter, and $y_2/y_1$ is the ratio change in the output characteristic.

| **Output Circuit Characteristic** | **Input Parameters** | | | |
|---|---|---|---|---|
| | $L$ | $W$ | $V_{bias}$ | $R_{mtj}$ |
| $f_{p1}$ | -1.7 | -0.2 | 0.9 | -0.2 |
| $R_3$ | -0.1 | -0.9 | 0.0 | 0.1 |
| $C_{mtj,dp,min}$ | 1.8 | 1.1 | -0.9 | 0.1 |
| $\sqrt{\overline{I^2_{out,tot,n}}}$ | -0.6 | -0.2 | 0.2 | -0.7 |

# Appendix B

# Summary of Queueing Model Equations

Presented in this appendix is a definition of terms used in the M/D/1 and M/G/1 queueing model equations that were developed in Chapter 5 for the general virtualized hardware configuration in Figure 5.2. This is shown in Table B.1. Next, a summary of the M/D/1 and M/G/1 queueing model equations are shown in Tables B.2 and B.3.

Table B.1: Queueing model definition of terms.

| **Term** | **Label** | **Definition** |
|---|---|---|
| $N$ | Number of data streams | Total number of data streams or contexts in the system |
| $C$ | Pipeline depth | Total number of pipeline stages or fine-grain contexts in the system |
| $S$ | Context switch cost | Number of clocks to context switch a group of fine-grain contexts |
| $R_S$ | Schedule period | Number of rounds that a group of fine-grain contexts are executed before doing a coarse-grain context switch |
| $\lambda$ | Mean arrival rate | Mean arrival rate at the queue of an individual stream |

| | | |
|---|---|---|
| $t_{CLK}$ | Clock period | Length of the critical path of the stage-to-stage delay |
| $T_T$ | Total schedule time | Total number of clocks to complete a full round of the hierarchical schedule |
| $T_V$ | Vacation time | Number of clocks in a long vacation period |
| $p_s$ | Service time fraction | Fraction of time in a service period for a group of fine-grain contexts |
| $p_0$ | Empty queue probability | Probability that the input queue is empty for an individual stream |
| $\overline{V}$ | Mean vacation waiting time | Mean waiting time of vacations for an individual virtual server |
| $\overline{X}$ | Mean service time | Mean service time for an individual virtual server |
| $\overline{X^2}$ | Service time second moment | Service time second moment for an individual virtual server |
| $\mu_s$ | Service rate | Mean service rate of an individual virtual server |
| $\rho$ | Utilization | Utilization of an individual virtual server |
| $T_{TOT}$ | Total achievable throughput | Total aggregate achievable throughput of all streams in the system |
| $W_q$ | Queue wait time | Latency in the queue for an individual stream |
| $W_h$ | Head of queue wait time | Additional latency at the head of the queue due to coarse-grain context switching or vacations; for M/D/1 this is separate; for M/G/1 this is included in $W_q$ |
| $W_s$ | Service wait time | Latency in the server (i.e., the pipelined circuit) for an individual stream |
| $W_T$ | Total wait time | Latency in the system for an individual stream |
| $N_q$ | Number in queue | Number of data elements in the queue for an individual stream |
| $N_s$ | Number in service | Number of data elements in the server for an individual stream |

| | | |
|---|---|---|
| $N_T$ | Number in system | Total number of data elements in the system for an individual stream |

Table B.2: Summary of M/D/1 queueing model equations.

| | **Case** | |
|---|---|---|
| **Term** | $N = C^*$ | $N > C$ |
| $\mu_s$ | $\frac{1}{C \cdot t_{CLK}}$ | $\frac{R_S}{(R_S N + SN/C) \cdot t_{CLK}}$ |
| $\rho$ | $\frac{\lambda}{\mu_s}$ | same |
| $T_{TOT}$ | $C \cdot \mu_s = \frac{1}{t_{CLK}}$ | $N \cdot \mu_s = \frac{R_S}{(R_S + S/C) \cdot t_{CLK}}$ |
| $W_q$ | $\frac{1}{\mu_s} \cdot \frac{\rho}{2(1-\rho)}$ | same |
| $W_h$ | 0 | $\frac{(R_S(N-C)+SN/C)^2 \cdot t_{CLK}}{2(R_S N + SN/C)}$ |
| $W_s$ | $\frac{1}{\mu_s} = C \cdot t_{CLK}$ | $C \cdot t_{CLK}$ (note, not $1/\mu_s$) |
| $W_T$ | $W_q + W_h + W_s$ | same |
| $N_q$ | $\lambda W_q = \frac{\rho^2}{2(1-\rho)}$ | same |
| $N_h$ | $\lambda W_h = 0$ | $\lambda W_h = \rho \cdot \frac{R_S \cdot (R_S(N-C)+SN/C)^2}{2(R_S N + SN/C)^2}$ |
| $N_s$ | $\lambda W_s = \rho$ | $\lambda W_s = \rho \cdot \frac{R_S C}{R_S N + SN/C}$ |
| $N_T$ | $N_q + N_h + N_s = \lambda W_T$ | same |

* Note, when $N = C$, set $S = 0$.

Table B.3: Summary of M/G/1 queueing model equations.

| | **Case** | |
|---|---|---|
| **Term** | $N = C^*$ | $N > C$ |
| $T_T$ | $R_S N$ | $R_S N + SN/C$ |
| $T_V$ | 0 | $T_T - R_S C = R_S (N - C) + SN/C$ |
| $p_s$ | 1 | $\frac{R_S C}{T_T} = \frac{R_S C}{R_S N + SN/C}$ |
| $p_0$ | same | $1 - \rho$ |
| $\overline{V}$ | $\frac{1}{2} p_0 \cdot C \cdot t_{CLK}$ | $\left[\frac{1}{2} p_0 \cdot \left[(1 - p_s) T_V + p_s \cdot C\right] + (1 - p_0) \left[\frac{T_V}{R_S}\right]\right] t_{CLK}$ |
| $\overline{X}$ | $C \cdot t_{CLK}$ | $C \cdot t_{CLK}$ |
| $\overline{X^2}$ | $C^2 t_{CLK}^2 = \frac{1}{\mu_s^2}$ | $C^2 t_{CLK}^2$ |
| $\mu_s$ | $\frac{1}{C \cdot t_{CLK}} = \frac{1}{\overline{X}}$ | $\frac{R_S}{T_T \cdot t_{CLK}} = \frac{R_S}{(R_S N + SN/C) \cdot t_{CLK}}$ |
| $\rho$ | same | $\frac{\lambda}{\mu_s}$ |
| $T_{TOT}$ | $C \cdot \mu_s = \frac{1}{t_{CLK}}$ | $N \cdot \mu_s = \frac{R_S}{(R_S + S/C) \cdot t_{CLK}}$ |
| $W_q$ | $\frac{\lambda}{2\mu_s^2 (1-\rho)} + \frac{\overline{V}}{1-\rho}$ | $\frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{\overline{V}}{1-\rho}$ |
| $W_h$ | same | $\frac{\overline{V}}{1-\rho}$ |
| $W_s$ | $\overline{X} = C \cdot t_{CLK}$ | $\overline{X} = C \cdot t_{CLK}$ (note, not $1/\mu_s$) |
| $W_T$ | same | $W_q + W_s$ |
| $N_q$ | $\lambda W_q = \frac{\rho^2}{2(1-\rho)} + \frac{\lambda \overline{V}}{1-\rho}$ | $\lambda W_q = \frac{\lambda^2 \overline{X^2}}{2(1-\rho)} + \frac{\lambda \overline{V}}{1-\rho}$ |
| $N_s$ | $\lambda W_s = \rho$ | $\lambda W_s = \rho \cdot \frac{R_S C}{R_S N + SN/C}$ |
| $N_T$ | same | $N_q + N_s = \lambda W_T$ |

* Note, when $N = C$, set $S = 0$.

# Vita

Michael J. Hall

**Degrees**

Ph.D. Computer Engineering, May 2015
M.S. Electrical Engineering, December 2007
B.S. Computer Engineering, May 2006

**Professional Societies**

Institute of Electrical and Electronics Engineers

**Publications**

**Hall, Michael** and Roger Chamberlain. "Performance modeling of virtualized custom logic computation," in *Proc. of IEEE 25$^{th}$ Int'l Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, Jun. 2014, pp. 72–73.

**Hall, Michael J.** and Roger D. Chamberlain. "Performance modeling of virtualized custom logic computations," in *Proc. of 24$^{th}$ Great Lakes Symp. on VLSI (GLSVLSI)*. New York, NY, USA: ACM, 2014, pp. 89–90.

**Hall, Michael J.**, Viktor Gruev, and Roger D. Chamberlain. "Performance of a resistance-to-voltage read circuit for sensing magnetic tunnel junctions," *Proc. of IEEE 55$^{th}$ Int'l Midwest Symp. on Circuits and Syst. (MWSCAS)*, Aug. 2012, pp. 639–642.

**Hall, Michael J.**, Viktor Gruev, and Roger D. Chamberlain. "Noise analysis of a current-mode read circuit for sensing magnetic tunnel junction resistance," *Proc. of IEEE Int'l Symp. on Circuits and Systems (ISCAS)*, May 2011.

Chamberlain, Roger D., Mark A. Franklin, Eric J. Tyson, James H. Buckley, Jeremy Buhler, Greg Galloway, Saurabh Gayen, **Michael Hall**, E.F. Berkley Shands, and Naveen Singla. "Auto-pipe: Streaming applications on architecturally diverse systems," *Computer*, vol. 43, pp. 42–49, Mar. 2010.

Engelbrecht, Linda M., Albrecht Jander, Pallavi Dhagat, **Michael J. Hall**. "A toggle MRAM bit modeled in Verilog-A," *Solid-State*

*Electronics*, vol. 54, no. 10, pp. 1135–1142, Oct. 2010. Selected Papers from ISDRS 2009.

Engel, George, **Michael J. Hall**, Justin Proctor, Jon Elson, Lee Sobotka, R. Shane, and Robert Charity. "Design and performance of a multi-channel, multi-sampling, PSD-enabling integrated circuit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 612, no. 1, pp. 161–170, Dec. 2009.

Njuguna, Raphael, **Michael Hall**, and Viktor Gruev. "Low power CMOS image sensor with programmable spatial filtering," in *Proc. of IEEE Sensors*, Oct. 2009, pp. 189–192.

**Hall, Michael**, Albrecht Jander, Roger D. Chamberlain, and Pallavi Dhagat. "Globally clocked magnetic logic circuits," Digest of Papers International Conference on Magnetics, May 2009.

Singla, Naveen, **Michael Hall**, Berkley Shands, and Roger D. Chamberlain. "Financial monte carlo simulation on architecturally diverse systems," in *Workshop on High Performance Computational Finance, 2008. WHPCF 2008.*, 16-16 2008, pp. 1–7.

**Hall, Michael J.** "Design Considerations in Systems Employing Multiple Charge Integration for the Detection of Ionizing Radiation," M.S. Thesis, Dept. of Electrical Engineering, Southern Illinois University Edwardsville, Dec. 2007.

May 2015