


8-2013

Analysis of Parameter Tuning on Energy Efficiency in Asynchronous Circuits

Justin Thomas Roark
University of Arkansas, Fayetteville

Follow this and additional works at: <http://scholarworks.uark.edu/etd>

 Part of the [Digital Circuits Commons](#), and the [VLSI and Circuits, Embedded and Hardware Systems Commons](#)

Recommended Citation

Roark, Justin Thomas, "Analysis of Parameter Tuning on Energy Efficiency in Asynchronous Circuits" (2013). *Theses and Dissertations*. 862.
<http://scholarworks.uark.edu/etd/862>

This Thesis is brought to you for free and open access by ScholarWorks@UARK. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu, ccmiddle@uark.edu.

Analysis of Parameter Tuning on Energy Efficiency in Asynchronous Circuits

Analysis of Parameter Tuning on Energy Efficiency in Asynchronous Circuits

A thesis submitted in partial
fulfillment of the requirements for the degree of
Master of Science in Computer Engineering

by

Justin Roark
University of Arkansas
Bachelor of Science in Electrical Engineering, 2011

August 2013
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Dr. Jia Di
Thesis Director

Dr. Dale R. Thompson
Committee Member

Dr. Scott C. Smith
Committee Member

ABSTRACT

Power and energy consumption are the primary concern of the digital integrated circuit (IC) industry. Asynchronous logic, in the past several years, has increased in popularity due to its low power nature. This thesis analyzes a collection of array multipliers with different parameters to compare two asynchronous design paradigms, NULL Convention Logic (NCL) and Multi-Threshold NULL Convention Logic (MTNCL). Several commercially available pieces of software and custom scripts are used to analyze the asynchronous circuits and their components to provide the energy consumption estimation on various parts of each circuit. The analysis of the software results revealed that MTNCL circuits are more energy efficient for any size provided the number of pipeline stages does not become too great. Otherwise NCL would consume less energy. A combinational logic gate count to register gate count ratio of 3 was given to help determine when an MTNCL circuit would have too many pipeline stages for circuits designed with IBM's 130nm 8RF-DM design kit.

ACKNOWLEDGEMENTS

I thank my advisor, Dr. Di, and my committee for their support on my thesis.

I also thank my family without whose support this would not have been possible and my friends for their tireless encouragement.

TABLE OF CONTENTS

1.	INTRODUCTION	1
2.	BACKGROUND	4
2.1	Asynchronous Circuit Design.....	4
2.1.1	NULL Convention Logic.....	4
2.1.2	Multi-Threshold NULL Convention Logic	9
2.2	Integrated Circuit Power and Energy Measurement.....	12
3.	TECHNICAL APPROACH.....	14
3.1	Circuits.....	14
3.2	Data Gathering Methodology	16
3.2.1	Commercial Software	17
3.2.2	Custom Script.....	20
4.	RESULTS AND ANALYSIS.....	26
4.1	Non-Cascaded Circuit Results	26
4.2	Cascaded Circuit Results	35
4.3	Analysis.....	44
5.	CONCLUSIONS.....	51
5.1	Summary.....	51
5.2	Conclusions.....	51
5.3	Future Work.....	52
	REFERENCES	53

LIST OF FIGURES

Figure 1. NCL TH23w2 Threshold Gate [2]	6
Figure 2. NCL Threshold Gate Transistor Schematic Diagram [7]	8
Figure 3. NCL 2-Bit Register with Completion Logic [2]	9
Figure 4. MTNCL Threshold Gate [8]	10
Figure 5. MTNCL 1-Bit Register	11
Figure 6. MTNCL Pipeline Architecture	12
Figure 7. Generic 4-Bit Array Multiplier Architecture	15
Figure 8. Data Gathering Flowchart	17
Figure 9. Virtuoso Screen Capture of I_{vdd} (top), I_{gnd} (middle), and V_{output} (bottom)	19
Figure 10. DA Primary Data Structure	23
Figure 11. Register Gate Count for Non-Cascaded Circuits	27
Figure 12. Register Energy Consumption for Non-Cascaded Circuits	28
Figure 13. Combinational gate Count for Non-Cascaded Circuits	30
Figure 14. Combinational Energy Consumption for Non-Cascaded Circuits	31
Figure 15. Sleep Tree Buffer Count for Non-Cascaded Circuits	32
Figure 16. Sleep Tree Energy Consumption for Non-Cascaded Circuits	33
Figure 17. Total Gate Count for Non-Cascaded Circuits	34
Figure 18. Total Energy Consumption for Non-Cascaded Circuits	35
Figure 19. Register Gate Count for All Cascaded Circuits	37
Figure 20. Register Energy Consumption for Cascaded Circuits	38
Figure 21. Combinational Gate Count for Cascaded Circuits	39
Figure 22. Combinational Energy Consumption for Cascaded Circuits	40

Figure 23. Sleep Tree Buffer Gate Count for Cascaded Circuits	41
Figure 24. Sleep Energy Consumption for Cascaded Circuits	42
Figure 25. Total Gate Count for Cascaded Circuits.....	43
Figure 26. Total Energy Consumption for Cascaded Circuits	44
Figure 27. MTNCL Combinational Logic to Register Ratio for Non-Cascaded Circuits	48
Figure 28. MTNCL Combinational Logic to Register Ratio for Cascaded Circuits	49

LIST OF TABLES

Table 1. Dual-rail NCL Logic Values [2].....	5
Table 2. NCL Fundamental Gate List [2].....	7
Table 3. Register Gate Count for All Non-Cascaded Circuits.....	27
Table 4. Energy Consumed by Registers in All Non-Cascaded Circuits	28
Table 5. Combinational Logic Gate Count for All Non-Cascaded Circuits.....	29
Table 6. Combinational Logic Gate Energy Consumption for All Non-Cascaded Circuits.....	30
Table 7. Sleep Tree Buffer Count for All Non-Cascaded Circuits.....	32
Table 8. Sleep Tree Energy Consumption for All Non-Cascaded Circuits	33
Table 9. Total Gate Count for All Non-Cascaded Circuits.....	34
Table 10. Total Energy Consumption for All Non-Cascaded Circuits.....	35
Table 11. Register Gate Count for All Cascaded Circuits.....	36
Table 12. Register Energy Consumption for All Cascaded Circuits	37
Table 13. Combinational Logic Gate Count for All Cascaded Circuits	39
Table 14. Combinational Logic Energy Consumption for All Cascaded Circuits	40
Table 15. Sleep Tree Buffer Count for All Cascaded Circuits.....	41
Table 16. Sleep Tree Buffer Energy Consumption for All Cascaded Circuits.....	42
Table 17. Total Gate Counts for All Cascaded Circuits	43
Table 18. Total Energy Consumption for All Cascaded Circuits.....	44
Table 19. MTNCL Combinational Logic to Register Ratio for Non-Cascaded Circuits	47
Table 20. MTNCL Combinational Logic to Register Ratio for Cascaded Circuits.....	49

1. INTRODUCTION

Recently, the digital integrated circuit (IC) industry has shifted its primary focus from increasing speed to decreasing energy consumption. There are many factors that have led to this shift. Digital electronics have become ubiquitous increasingly in places where the availability of power is limited. Smart phones and other mobile devices are prime examples. The market for mobile devices continues to grow, which is a strong driving force for lower power electronics since these devices have restricted on energy capacity. As the size of transistors decrease, their density on chip increases. This has led to a rise in increasingly complex circuits which require more power. The energy density of batteries has not increased at the same rate as the power demand of ICs. Batteries are not able to keep up with the power demands of denser circuits. Reduction in energy consumption is necessary for digital circuits to make better use the limited energy available in a mobile environment. Heat dissipation is also a concern. Smaller feature sizes lead to increased heat concentrations. Excess heat lowers the performance of circuits and shortens their lifespan. The market for digital electronics is expected to continue growing as is the need for lower power devices [1].

Synchronous circuits have been the main focus of the digital IC design industry. With the shift towards lower power consumption, however, asynchronous circuits are beginning to grow in popularity. Asynchronous circuits boast lower power consumption and robustness towards process and environment variation. Currently, most designers and Computer-Aided Design (CAD) tools are focused on synchronous circuits. This is a challenge that asynchronous circuits must overcome to become more widely adopted [2].

Asynchronous circuits fall into two categories, bounded delay and delay-insensitive. This work is concerned with delay-insensitive circuits. NULL Convention Logic (NCL) is a quasi-delay insensitive asynchronous design methodology. NCL uses multi-rail signals. The multi-rail signals allow the addition of a third state beyond Boolean '1' and '0' called NULL. The NULL state acts as a buffer between DATA states. The multi-rail encoding causes NCL gates to be larger than synchronous logic gates [2].

Another asynchronous design paradigm, Multi-Threshold NULL Convention Logic (MTNCL) combines NCL with Multi-Threshold CMOS (MTCMOS) power gating [3] [4]. MTNCL uses transistors with different thresholds voltages to perform power gating inside a logic gate. Most MTNCL gates are smaller than NCL gates, but require large buffers to drive the sleep signals used for power gating [5]. This work analyzes the trends in NCL and MTNCL energy consumption in circuits of different sizes and number of pipeline stages.

This thesis uses gate-level energy models and logical activity models to estimate energy consumption in NCL and MTNCL circuits. NCL and MTNCL gates were simulated to create energy consumption models. The circuits to be compared were simulated to determine gate switching activity. These two sets of data were then combined to estimate the energy consumption of the entire circuit. Array multipliers ranging in size and number of pipeline stages were analyzed using this method to illustrate the trends in NCL and MTNCL energy consumption.

This thesis is organized into five chapters. Chapter 1 is the introduction. Chapter 2 provides information on asynchronous circuits, focusing on NULL Convention Logic and Multi-Threshold NULL Convention Logic. Chapter 3 contains a detailed description of the simulation and calculation processes used to gather the resulting data. Chapter 4 presents the gathered data

and the analysis. Chapter 5 summarizes the work accomplished, discusses future work, and provides a conclusion.

2. BACKGROUND

2.1 Asynchronous Circuit Design

In the past year, the digital IC industry has been primarily focused on synchronous circuits. With decreased transistor feature size and increasing design complexity, clock management has become a major issue. Power consumption has increased with faster clock speeds and larger circuit sizes. Clock distribution is also a significant challenge as larger circuits are prone to increased clock skew. Asynchronous circuits recently have begun growing in popularity since they remove the needs for clocks. Handshaking protocol is used in place of clocks to control the circuit operation. Asynchronous circuits benefit from using less power, having less electromagnetic interference, and producing less noise than their synchronous counterparts [2].

Asynchronous circuits also boast delay insensitivity. Delay-insensitive circuits do not need to take into account wire and gate delays. They function correctly regardless of delay fluctuations inside individual logic gates. Delay insensitivity provides robustness to asynchronous circuits in the form of tolerance to process, supply voltage, and temperature variations. As long as the transistors are able to function the asynchronous circuit will operate correctly [2].

2.1.1 NULL Convention Logic

NULL Convention Logic (NCL) is a quasi-delay insensitive asynchronous circuit design paradigm. NCL is considered quasi-delay insensitive because it assumes wire forks are isochronic. This assumption only needs to be applied to wires within a basic component and not

to wires connecting components. The isochronic fork delay assumes that the delay caused by a wire is much less than the delay caused by a logic element [2].

NCL is considered symbolically complete; meaning the logic itself can express its validity. Synchronous circuits are symbolically incomplete because they rely on time, which is external to the logic, to determine when the data is valid. NCL accomplishes symbolic completeness by adding a third state to its logic set, NULL. The NULL state indicates the current signals are not data. The NULL state is used as a buffer between two DATA states. To provide both the standard Boolean 0 and 1 logic values and the NULL value, multiple wires, or rails, must be used for a single signal. One of the most common forms of NCL multi-rail signals is dual-rail. Dual-rail uses two wires to represent the three possible logic states shown in Table 1 [2]. DATA 0 indicates a Boolean logic 0 and DATA 1 indicates a Boolean logic 1. When both rails are low the signal is in a NULL state. If both rails are high then it is an illegal state [6].

Table 1. Dual-rail NCL Logic Values [2]

	NULL	DATA 0	DATA 1	Illegal
Rail 0	0	1	0	1
Rail 1	0	0	1	1

NCL circuits are designed using the 27 fundamental gates in Table 2. These gates represent all possible Boolean combinations using four variables or less [2]. These gates are known as Threshold Gates and use the TH_{mn} notation, where n is the number of inputs to the gate and m is the threshold. At least m of the n inputs must be high for the output to rise, otherwise it remains low. Inputs can also be given weights. The notation for a gate with weighted inputs is $TH_{mnw}\{x_1 x_2 \dots x_n\}$ where x is an integer greater than 1. For example, a TH_{23w2} has three inputs where the first input carries the weight of two inputs towards the threshold value.

Since the threshold is 2 the first pin is able to raise the output by itself, whereas the second and third inputs would each require one other input to be high in order to raise the output. The fundamental gates also have resettable and inverting variants that are used in storage and control logic. TH_{mnd} and TH_{mn} denote reset high and reset low, respectively. An inverting gate takes the form of TH_{mb} . Figure 1 shows the symbol used to represent a TH_{23w2} gate [2]. The number in the center of the symbol is the threshold value. The first input forks indicated that it has a weight of 2.

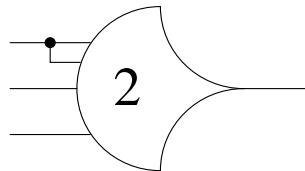


Figure 1. NCL TH_{23w2} Threshold Gate [2]

Table 2. NCL Fundamental Gate List [2]

NCL Threshold Gate	Boolean Function
TH12	$A + B$
TH22	AB
TH13	$A + B + C$
TH23	$AB + AC + BC$
TH33	ABC
TH23w2	$A + BC$
TH33w2	$AB + AC$
TH14	$A + B + C + D$
TH24	$AB + AC + AD + BC + BD + CD$
TH34	$ABC + ABD + ACD + BCD$
TH44	$ABCD$
TH24w2	$A + BC + BD + CD$
TH34w2	$AB + AC + AD + BCD$
TH44w2	$ABC + ABD + ACD$
TH34w3	$A + BCD$
TH44w3	$AB + AC + AD$
TH24w22	$A + B + CD$
TH34w22	$AB + AC + AD + BC + BD$
TH44w22	$AB + ACD + BCD$
TH54w22	$ABC + ABD$
TH34w32	$A + BC + BD$
TH54w32	$AB + ACD$
TH44w322	$AB + AC + AD + BC$
TH54w322	$AB + AC + BCD$
THxor0	$AB + CD$
THand0	$AB + BC + AD$
TH24comp	$AC + BC + AD + BD$

NCL gates also have a hysteresis property. Once the output is high it will remain high until all of the inputs have fallen. The hysteresis is necessary to maintain delay insensitivity [2]. Figure 2 is the generic transistor schematic diagram for an NCL threshold gate [7]. The *set* and *reset* blocks are responsible for changing the output of the gate to high and low, respectively.

The *hold0* and *hold1* blocks are in series with the two hysteresis transistors. Together, they maintain the last output until the *set* and *reset* blocks force a new output value.

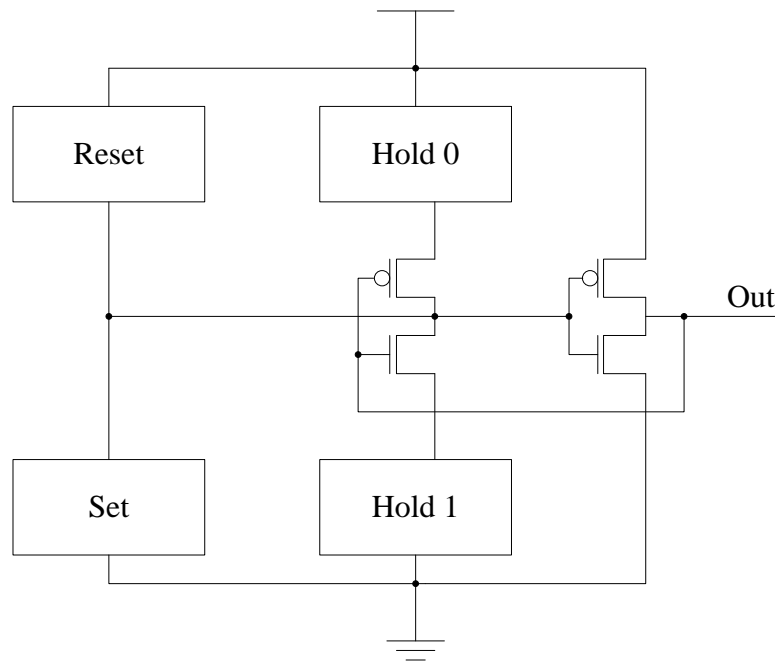


Figure 2. NCL Threshold Gate Transistor Schematic Diagram [7]

The handshaking signal in NCL is generated by completion logic. The completion logic's purpose is to detect when all of the signals at a particular pipeline stage have become either all DATA or all NULL. Once all of the signals are DATA the completion logic will request NULL from the previous pipeline stage. Conversely, it will request DATA once all of the signals have become NULL. The previous stage will propagate the requested wavefront, DATA or NULL, once it has become available. The request for DATA or NULL is handled by the K_o signal which is generated by the completion logic. Each NCL register consists of two TH22 gates, one for each rail of the signal. The outputs of both TH22 are connected to a TH12b gate as shown in Figure 3. The TH12b gate determines if the signal at the register is DATA or NULL. The output of the TH12b gate is the K_o signal for each individual register. All of the registers' K_o signals are

fed into the completion logic. The completion logic is a tree composed of the necessary number of TH22, TH33, and TH44 gates required to create one K_o signal. Figure 3 is an example of a 2-bit register [2]. The dashed box contains the completion logic which in this case is just a single TH22 gate. The K_o signal for the current pipeline stage is sent to the previous pipeline stage. The K_i signal receives the K_o signal coming from the next pipeline stage.

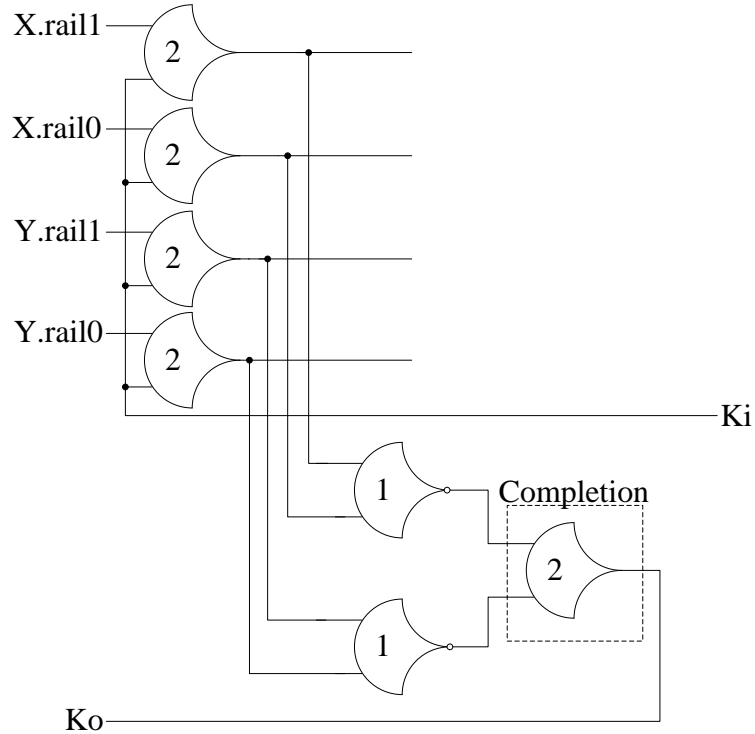


Figure 3. NCL 2-Bit Register with Completion Logic [2]

2.1.2 Multi-Threshold NULL Convention Logic

Multi-Threshold NULL Convention Logic (MTNCL) is a lower power and smaller area version of NCL. MTNCL incorporates the power gating principles of Multi-Threshold CMOS (MTCMOS) into the NCL framework [4] [5]. MTNCL does not propagate a NULL wavefront in the same fashion as regular NCL does. Instead, it puts all of the gates in the corresponding

pipeline stage into a sleep mode. The sleep mode forces the output of the gates low, which creates the NULL. The sleep mode reduces leakage power during the NULL cycle by gating the power with high threshold transistors. The addition of the sleep signal removes the need for hysteresis logic in MTNCL threshold gates. Two extra transistors, however, are needed in each gate to implement the sleep logic. Most MTNCL gates are smaller than their equivalent NCL gate [5]. The MTNCL threshold gate in Figure 4 has only the *hold0* and *set* logic blocks of the NCL threshold gate [8]. The *reset* block is obsolete since the sleep signal forces the output low and the lack of hysteresis removes the need for the *hold1* block. The *hold0* and *set* blocks are complementary which ensures that the gate's internal node will never be floating.

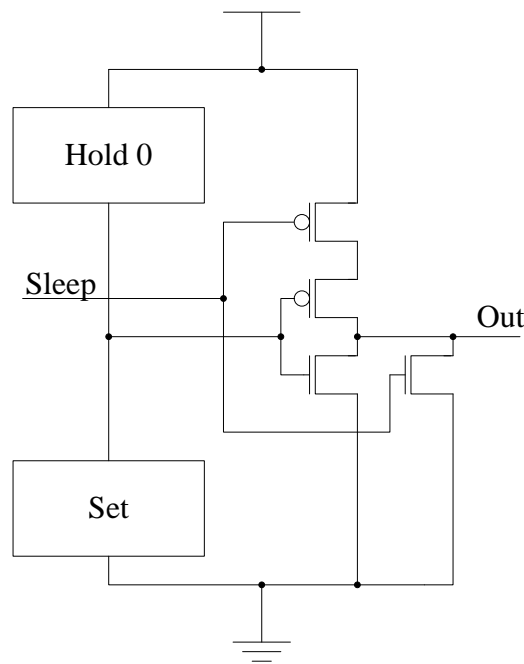


Figure 4. MTNCL Threshold Gate [8]

MTNCL registers are made from the MTNCL variant of the TH12 gate, the TH12m. The “m” denotes an MTNCL gate. Since MTNCL gates lack hysteresis, which is a necessary function for a register, the output is fed back to one of the TH12m inputs, as illustrated in Figure 5. The

second input to the TH12m gate is the input to the register. Once the register's input goes high the output will go high and remain high until the register is slept.

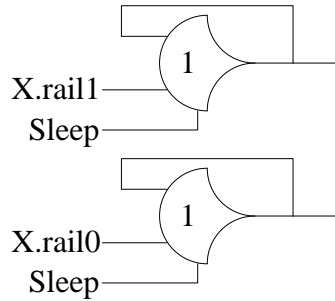


Figure 5. MTNCL 1-Bit Register

In MTCMOS synchronous circuits, generating the sleep signals requires additional logic. The logic overhead is often complex since it must synchronize with the circuit to prevent glitches and maintain data integrity [9]. However, the K_o signal in the NCL architecture provides a natural sleep signal. MTNCL uses early completion logic. Early completion logic is driven by the input signals going to the registers instead of the outputs, as in NCL. Early completion logic prevents partially formed DATA wavefronts from propagating through the combinational logic [2]. Figure 6 illustrates the MTNCL architecture. The K_o generated by the early completion logic is used to sleep the current stage's registers and the combinational and completion logic of the next stage.

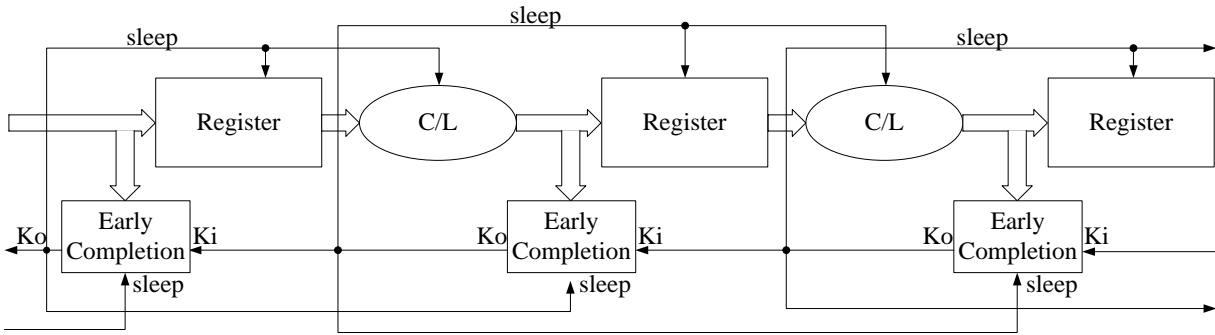


Figure 6. MTNCL Pipeline Architecture

2.2 Integrated Circuit Power and Energy Measurement

In a digital CMOS circuit, power is consumed when the circuit is active and when it is inactive or in standby. During active mode there are two types of power, dynamic and short circuit. Dynamic power is the power used by charging and discharging the load capacitance on a logic gate. Dynamic power is data dependent. Short circuit current is the power used when both the p-network and n-network in a CMOS gate are both on. The short circuit condition occurs while an input to a logic gate is rising or falling and it reaches the mid-point. The mid-point is around half of the supply voltage which is enough to bring the all of the transistors connected to the input signal into saturation. This provides a direct connection from power to ground. Short circuit power is affected by the rise and fall time of the input signal and the load capacitance. The slower the rise or fall time is or the greater the load capacitance the longer both the PFETs and NFETs are simultaneously saturated [1].

When the circuit is inactive leakage power is consumed. Leakage power is the result of transistors not being able to turn off completely due to decreased size and lowered threshold voltages. Sub-threshold current is the current that flows between the source and drain while the

voltage potential across gate and source (V_{GS}) is below the threshold voltage. Sub-threshold current causes the majority of the leakage power. Leakage power is generally smaller than active power, but its effect can be substantial on the total power usage if the circuit spends a significant portion of time inactive while the circuit is being supplied with power [1].

For synchronous circuits power is a natural figure of merit since the circuits by definition operate on a specific time interval. Delay-insensitive asynchronous circuits do not operate within a specific time interval so power measurements do not accurately represent their energy usage. For asynchronous circuits, total energy and energy per operation are standard figures of merit. For synchronous circuits, power measurements can easily be converted to energy by integrating power over the total run time or operation time. Energy can be calculated by measuring the current drawn by the circuit while it is operating on a set number of inputs, integrating the current over time, and multiplying it by the supply voltage.

3. TECHNICAL APPROACH

3.1 Circuits

All of the circuits used to compare the NCL and MTNCL paradigms were array multipliers with varying configurations. Array multiplier architecture was chosen because it is a very regular circuit that can be expanded easily. The bit-width and number of pipeline stages were the parameters altered between each circuit. In addition, several multipliers were cascaded in series to create larger circuits. There are 11 non-cascaded circuits. These circuits were then cascaded to create several larger circuits. Each circuit has an NCL and MTNCL version.

For the 11 basic circuits there are four bit-widths, i.e., 4, 8, and 16 bits. There are also several different pipeline granularities. There are 1-, 2-, and 4-pipeline stage variants of the 4-bit multiplier. The 8-bit multiplier has 1-, 2-, 4-, and 8-stage pipeline designs. For the 16-bit multiplier there are 1-, 2-, 4-, and 8-stage versions.

There are 11 cascaded multiplier circuits. Each cascaded circuit is 4 copies of one of the 11 single multipliers. The four multipliers are connected in series using the output of one as the input to another. The output is split in half with each half feeding one of the two inputs of the next multiplier. The output is also inverted in between the multipliers. The inverters prevent a result equaling zero from propagating through the entire chain and resulting in very minimal circuit activity. Without the inverters zeros appear frequently. All together there are 22 NCL and 22 MTNCL circuits for a total of 44 circuits.

The array multiplier is composed of half adders, full adders and AND gates. The generic array multiplier in Figure 7 has a bit-width of 4. The first three rows of the multiplier create and sum partial products. The final row is a ripple carry adder that sums the remaining partial

products. The number of rows is equal to the bit-width of the multiplier and the total number of half and full adders in each row is one less than the bit-width. Both the NCL and MTNCL paradigms follow this architecture.

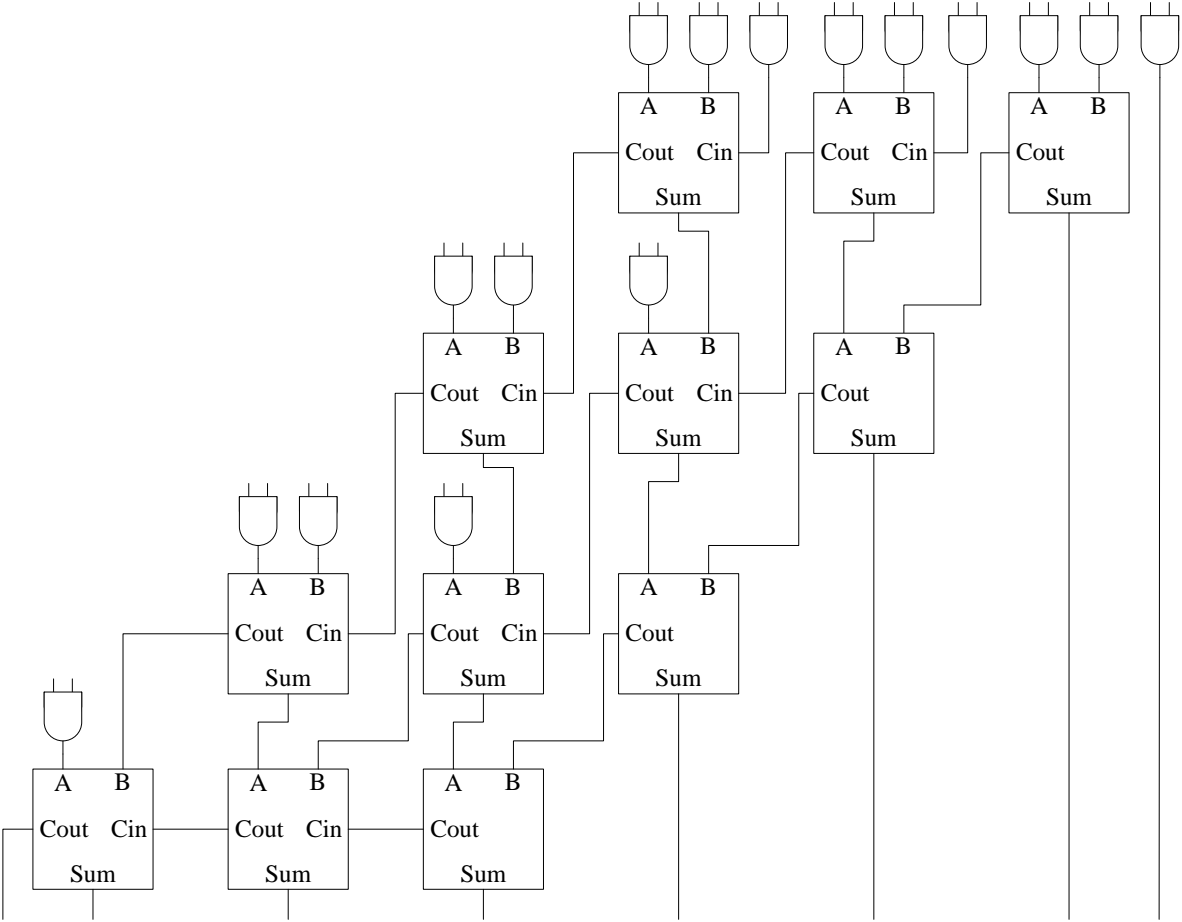


Figure 7. Generic 4-Bit Array Multiplier Architecture

NCL and MTNCL require registers at the inputs and outputs of the circuit. When counting pipeline stages the registers at the output are not counted. The number of pipeline stages is determined by the number of combinational blocks. Register stages are inserted between the rows of half and full adders. The register stages are also spaced evenly throughout the multiplier. A 4-bit multiplier with 2 stages will have registers before row 1, between row 2

and 3, and after the final row. There are two groups of combinational logic between the registers which is why it is considered a two pipeline stage design.

NCL and MTNCL circuits do not require a timing analysis like synchronous circuits. All asynchronous threshold gates are sized solely on capacitive loads they drive. All of the gates in the libraries used in this thesis work have three different drive strengths. The gate libraries also include two buffers with greater drive strengths than the combinational gates. A custom script is used to calculate the capacitive loads and select the appropriate gate or buffer size. If a load is too great for the largest buffer, the script will create a buffer tree. The buffer tree is created by dividing the total load capacitance equally among the minimum number of gates required to drive the entire load. If the total load capacitance to drive all of the newly added buffers is too great for the combinational gate then another layer of buffers is added to drive the first layer of buffers. Buffer layers are added until the input capacitance of the buffer tree is small enough for a combinational gate to drive.

3.2 Data Gathering Methodology

Four pieces of software were primarily used to calculate the energy usage of each circuit. Cadence Virtuoso was used to capture gate-level energy consumption. Synopsys Liberty NCX was used to find the input capacitance of each gate. Mentor Graphics Modelsim recorded the switching activity of every net in each circuit during simulation. Finally, a custom script was developed to combine the data from Virtuoso and Modelsim with a Verilog netlist to provide an energy consumption breakdown of the circuit. All of the circuit netlists are required to be in the Verilog format and flattened down to the gate-level. This is necessary for the analysis. The flattened netlists used in this thesis were generated with Synopsys Design Compiler. Figure 8 is a

flowchart depicting the order programs were run and the files they generated. This method is similar to the approach used by Venkat Satagopan et al. in [10].

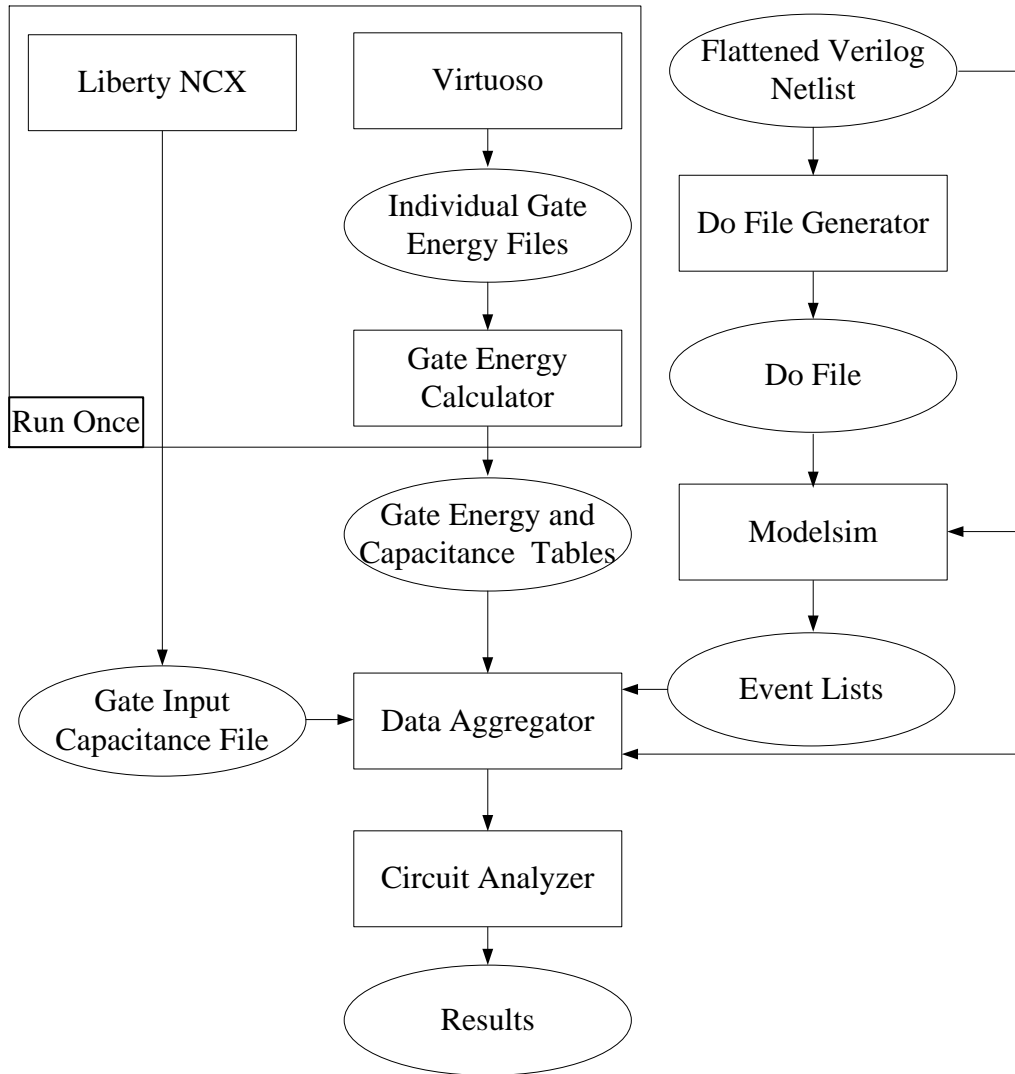


Figure 8. Data Gathering Flowchart

3.2.1 Commercial Software

In Virtuoso, every gate used in each circuit was individually simulated for energy consumption. All of the gates were designed using IBM's 130nm 8RF-DM design kit. A test vector was used to exercise every possible input pattern to a gate. Each time the output of a gate

rose or fell the currents at the power and ground pins were recorded. Each current was then integrated over time and multiplied by the source voltage resulting in energy consumption, as illustrated by Equation 1. It was necessary to capture both the power and ground current because of how the simulator operated. The simulator would display a current spike on the power pin when the output rose and a current spike on the ground pin when the output fell. Figure 9 is a screen capture from Virtuoso displaying the current spikes and output waveform. This is how the simulator showed energy flowing into the capacitive load as it was being charged during a rising output and energy flowing from the load to ground as the output fell and the load discharged. Each spike was recorded separately. The current flowing through the ground pin was negative since it represented current flowing out of the gate. Thus, the ground current was inverted during the calculation to accurately represent energy being consumed and not generated. The simulation was repeated for a range of capacitive loads. The capacitances were selected for each gate based on the capacitive range that gate is rated to drive. The capacitance and energy consumption information was written to a file. Once all of the gates had been simulated this program was finished and would need to be re-run for each circuit, as indicated by the dashed box in Figure 8. The same gate energy and capacitance table file is used by every circuit.

$$E = V \int I(t)dt$$

Equation 1. Energy Equation

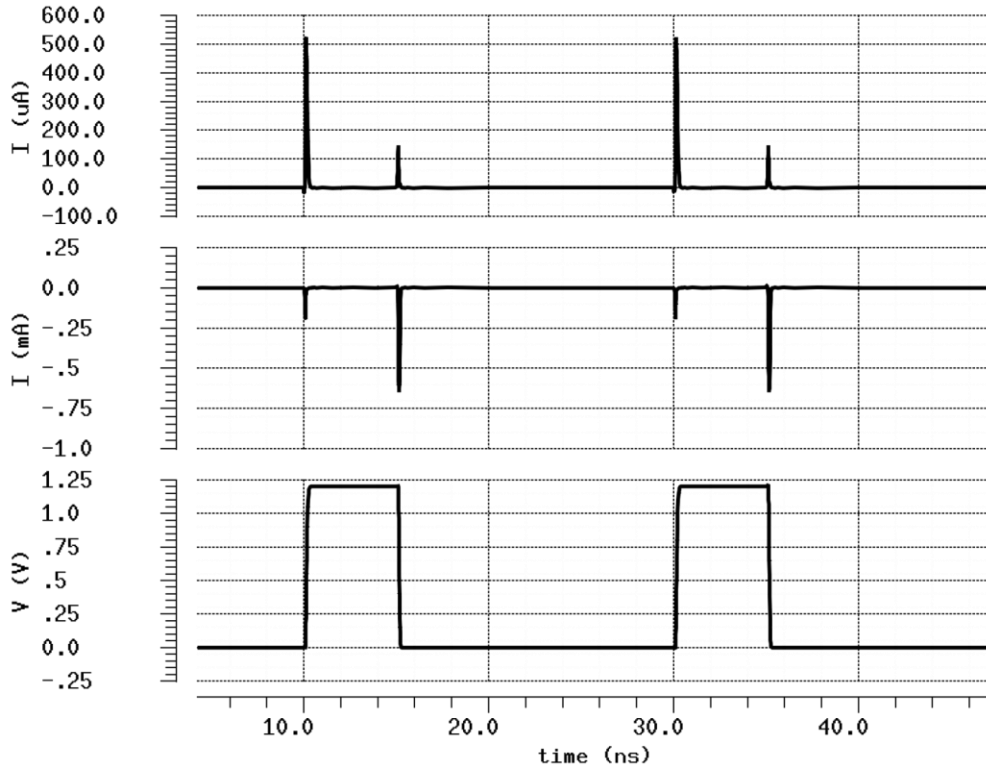


Figure 9. Virtuoso Screen Capture of I_{vdd} (top), I_{gnd} (middle), and V_{output} (bottom)

Liberty NCX is used to characterize logic gates. It runs simulations on each gate and provides information on rise and fall times with respect to capacitance. The software also provides the input capacitance on every input to a gate. Each gate's input capacitance and respective pin were extracted from Liberty NCX' .lib files and stored in an array written to the gate input capacitance file. This file was only generated once because it contained all of the possible gates the circuits of interest used. Thus, Liberty NCX only needed to be run once, as detailed in Figure 8.

The Modelsim simulations provide the switching activity of each circuit. The digital waveform for every net in the circuit being simulated was recorded. A list of all the nets in a circuit must be given to Modelsim. The custom script that will be discussed in Section 3.2.2 has

a function, called the Do File Generator, that generates a list of signals in a “Do” file for Modelsim. The Do File Generator runs separately from the main parts of the custom script and, thus, should be considered its own program. Even though it is a custom build function, the Do File Generator is in the Commercial Software section because its sole purpose is to generate files for Modelsim. The Do file compiles all of the VHDL and Verilog files required to simulate the circuit. It also selects which signals the simulator needs to record. The function that generates the Do file reads in a flattened Verilog netlist and a Do file containing the paths to the testbench and gate description files used by the circuit. The Do File Generator adds every output of every gate to the Do file. The testbench simulations consisted of random input patterns. All of the simulations used the same seeds for the random number function. The result of the random function was then scaled to match the bit-width of circuit currently being simulated. Using the same seeds allowed the input patterns to be unbiased, but still remain the same for all circuits with the same bit-width. After the simulation had completed, an “event list” was exported from Modelsim. The event list only records when the output of a gate changes and the time the change occurred. The resulting file is significantly smaller than a “tabular list” which contains the value of every gate output at every time step. An event list needs to be generated for every circuit that is processed and must be regenerated if any changes are made to the netlist.

3.2.2 Custom Script

The custom script was written in Python and has three main components: the Gate Energy Calculator (GEC), the Data Aggregator (DA), and the Circuit Analyzer (CA). The first component, the GEC, is concerned with processing the Virtuoso files. The program reads in a file which contains the energy data for a single gate. The energy used during a rising output and

the corresponding falling output are added together. The number of rising and falling outputs is determined by the logic function of the gate. An average is taken of all the different outputs' energy consumptions that were produced by the comprehensive input patterns. The average is taken only for a specific capacitive load. The average energy consumption and capacitive load are stored in a table for that gate. This is repeated for every capacitive load contained in the Virtuoso output file. The program then reads the next file which contains the energy data for a different gate. After the program has read and processed every file, a new file is created. The table containing the capacitive loads and related energy consumption for each gate is then written to the new file. Figure 8 refers to this file as the Gate Energy and Capacitance Tables. This file was only generated once since it contains all of the possible gates the circuits of interest, which is why the GEC is inside of the dashed box in Figure 8.

The purpose of the DA is to collect all of the information from the various sources and put it into one data structure. The DA starts with the Modelsim event list. From this file, the DA collects gate names, the total number of times a gate output switches, and the number of times the gate output goes high. Next the flattened Verilog netlist is read in. The DA builds an array that is organized by the type of gate. Each type of gate element in the array contains another array that holds every specific instance of that gate type. Each entry for a specific instance contains the gate name and any output nets. This forms the base of the data structure. Next the gate input capacitance information is read in from the file generated by Liberty NCX. This information is used with the netlist to calculate the capacitive load on each net, as well as the fanout. The program takes the output net of every gate and finds all of the input pins it drives. The capacitance on the input pins is looked up in the gate input capacitance array. The capacitances are totaled and stored as the load capacitance for the gate driving the net. All of the

gates being driven are counted as the script finds them and this number is the fanout. Next the gate energy data from the GEC is read in and stored temporarily. Finally, all of the information is collected so that an entry for a specific gate contains its name, output nets, load capacitance on each output net, fanout, and the number of times the output signal rose. All of these entries are categorized by their type of gate. Stored with the gate type is its table containing capacitive loads and energy consumption. The data structure is illustrated in Figure 10. Only rising outputs are kept because the gate energy data contains the energy used in both rising and falling. This is acceptable because a rising gate removes the total rising and falling energy from the power source during the rising transition. The falling energy is just stored in the load until the fall occurs. A second data structure is then created to represent all of the connections in the netlist. This data structure is used by the CA to quickly traverse the netlist. These two data structures are then passed to the CA.

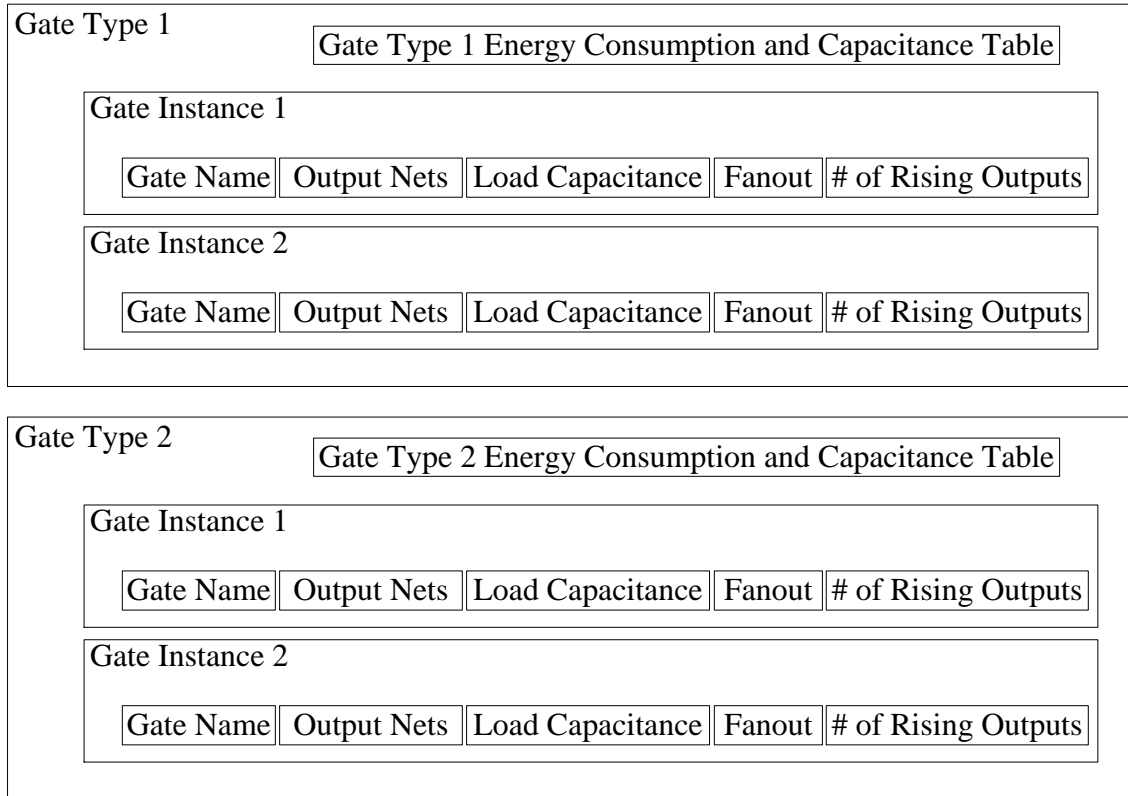


Figure 10. DA Primary Data Structure

The CA's primary purpose is to calculate the energy consumption of various parts of a circuit and ultimately the whole circuit. The CA breaks each circuit into three parts: registers, combinational logic, and sleep trees. The three parts are added together to give the total. The CA returns energy consumption and a gate count. First, the CA reads in the primary data structure and the secondary helper structure created by the DA. The CA separates all of the gates into the three categories previously mentioned. The sleep tree section is only relevant to MTNCL since NCL does not have any slept logic. To find the sleep signals, the sleep input signal on every sleep-able gate is examined and the root signal is determined from net name. The sleep signal can also be reverse traced through the buffer tree to the root, in the absence of useful net names. Once all of the sleep nets have been collected, the sleep nets' buffer trees are recursively

traversed and all of the buffer gate names are collected in a list. The secondary data structure is used for the recursive buffer tree traversal. Buffers not used in a sleep net are considered part of the combinational logic. Next, the registers are found by searching for the fundamental gates that are used as NCL and MTNCL registers. Since the fundamental gates can be used as combinational logic the gates, the connections are checked to make sure they are in the register configurations in Figure 3, for NCL, and Figure 4, for MTNCL. For NCL, the function locates TH22 gates. It then checks for a TH12 gate that is driven by the output of the TH22. It then back traces the second input of the TH12 gate to verify it is driven by another TH22 gate. This confirms the dual-rail register. The gates are then added to the register list. For MTNCL, the function verifies that the TH12m gate is driving one of its own inputs. In both cases, the function will search through a buffer tree, if there is one on the output, to verify if the gate is in a register configuration. The completion logic for both NCL and MTNCL is not included in the register list. The combinational logic is simply the remaining gates not in the sleep tree or register lists.

The energy is then calculated for each of the lists. For each gate, the load capacitance is referenced against the gate energy consumption and capacitance table to find the per output rise energy consumption. The two capacitance values in the table that the load capacitance falls between are used with the equation of a line, Equation 2, to find a linear approximation of the energy used. In Equation 2, x is capacitance and y is energy consumption. The subscripts 1 and 2 represent the two table entries and the prime markings are the values for the current gate. The energy use per rise is multiplied by the rising output count to produce the total energy used by the gate during the simulation. All of the energy consumption for the gates in each list is totaled to give a total for each part of the circuit. The three sums are then added together to get the total

circuit energy consumption. The length of each list, which is the gate count, and the energy consumption totals are all written out to the results file.

$$y' = y_2 - \frac{(y_2 - y_1)(x_2 - x')}{x_2 - x_1}$$

Equation 2. Equation of a Line

4. RESULTS AND ANALYSIS

The results produced by the technical approach detailed in Chapter 3 are included in this section. Section 4.1 and 4.2 contain the raw results with brief descriptions. Section 4.3 is a detailed analysis of the results.

4.1 Non-Cascaded Circuit Results

This section contains the results for the 22 non-cascaded multiplier circuits for both NCL and MTNCL. The results are separated into the three main circuit components and then a total. In each section there is a table and chart for gate count and a table and chart for energy consumption. 100 random test patterns were used to generate the energy data in each simulation.

The register gates include only gates that store values. Since all of the designs are dual-rail, both gates that comprise a dual-rail register are included in these tables and charts. The completion logic is not included in this category. Table 3 and Figure 11 contain the register counts for all of the non-cascaded circuits. The counts between NCL and MTNCL are nearly identical. The slight differences seen in the 4-bit, 2-stage; 4-bit, 4-stage; 8-bit, 4-stage; 8-bit, 8-stage; and 16-bit, 8-stage circuits are from minor optimizations made by the software. Table 4 and Figure 12 contain the energy consumption of all the register gates. The notable increase in register energy consumption in the NCL 16-bit, 1-stage multiplier is due to the large fanout some of the registers are required to drive. The registers have less fanout in the NCL 16-bit, 2-stage multiplier.

Table 3. Register Gate Count for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Registers	MTNCL Registers
4	1	32	32
	2	60	58
	4	110	108
8	1	64	64
	2	124	124
	4	244	242
	8	470	468
16	1	128	128
	2	252	252
	4	500	500
	8	996	994

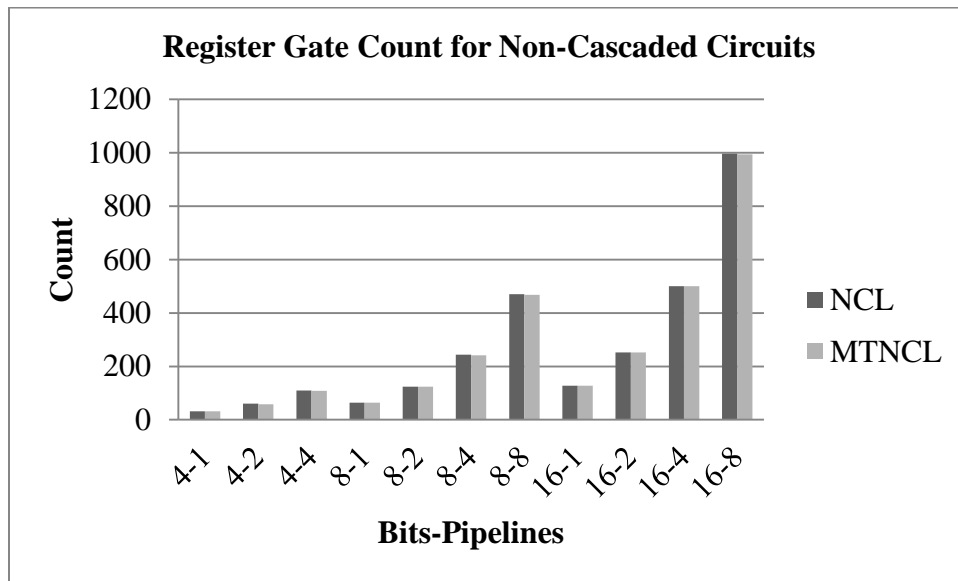


Figure 11. Register Gate Count for Non-Cascaded Circuits

Table 4. Energy Consumed by Registers in All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Register Energy Used (J)	MTNCL Register Energy Used (J)
4	1	3.06E-11	2.59E-11
	2	4.47E-11	4.65E-11
	4	7.10E-11	8.60E-11
8	1	1.03E-10	5.89E-11
	2	1.27E-10	1.06E-10
	4	1.90E-10	2.00E-10
	8	3.13E-10	3.79E-10
16	1	5.56E-10	1.88E-10
	2	3.86E-10	2.65E-10
	4	5.10E-10	4.62E-10
	8	8.30E-10	8.52E-10

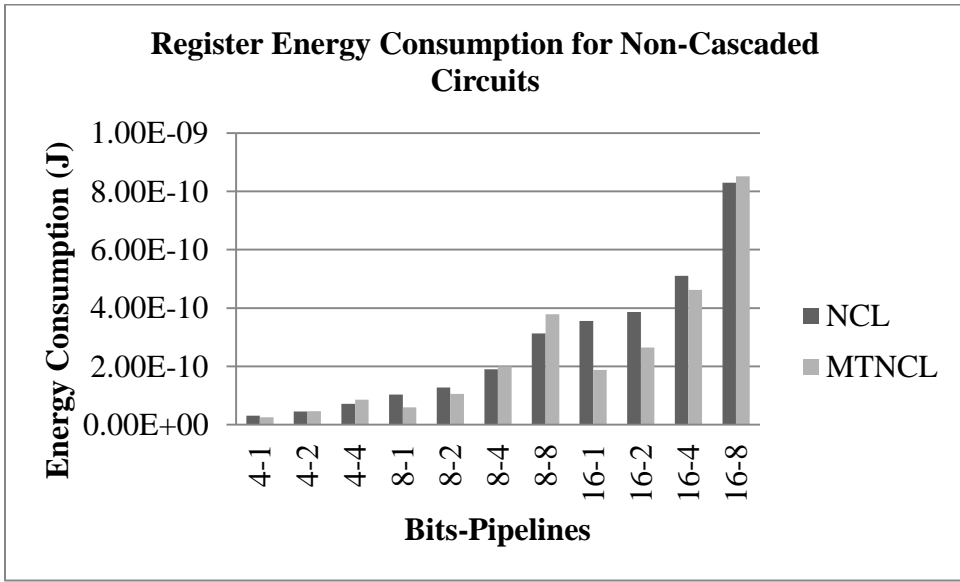


Figure 12. Register Energy Consumption for Non-Cascaded Circuits

The combinational logic (C/L) gates include gates that are not registers, nor are part of a sleep tree. All of the non-cascaded combinational gate counts are found in Table 5 and Figure 13. The combinational logic gate count for MTNCL is lower than NCL in every circuit. Table 6 and Figure 14 contain the combinational logic gate energy consumption for all of the non-cascaded

circuits. MTNCL combinational logic gates in each circuit used less energy than NCL combinational logic gates in the equivalent circuit. The increase in gates and their respective energy use between the different numbers of pipeline stages of a particular bit-width is mostly due to the completion logic.

Table 5. Combinational Logic Gate Count for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Combinational Logic	MTNCL Combinational Logic
4	1	151	96
	2	170	108
	4	206	132
8	1	593	381
	2	630	406
	4	717	455
	8	872	545
16	1	2337	1525
	2	2422	1569
	4	2594	1663
	8	2944	1854

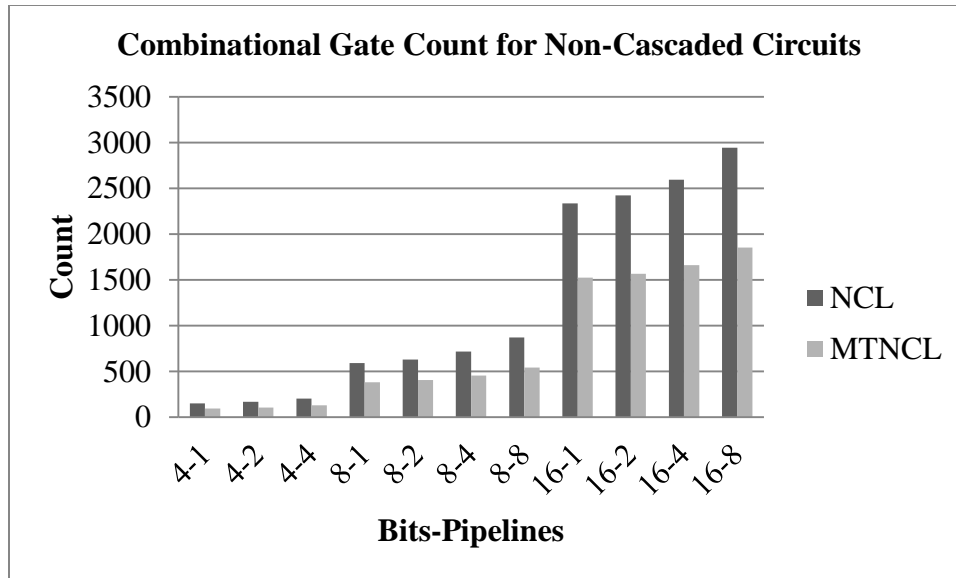


Figure 13. Combinational gate Count for Non-Cascaded Circuits

Table 6. Combinational Logic Gate Energy Consumption for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL C/L Energy Used (J)	MTNCL C/L Energy Used (J)
4	1	6.86E-11	4.53E-11
	2	8.78E-11	5.28E-11
	4	1.22E-10	6.62E-11
8	1	2.73E-10	1.78E-10
	2	3.06E-10	1.93E-10
	4	3.74E-10	2.21E-10
	8	5.02E-10	2.76E-10
16	1	1.09E-09	6.95E-10
	2	1.16E-09	7.35E-10
	4	1.29E-09	7.95E-10
	8	1.58E-09	9.05E-10

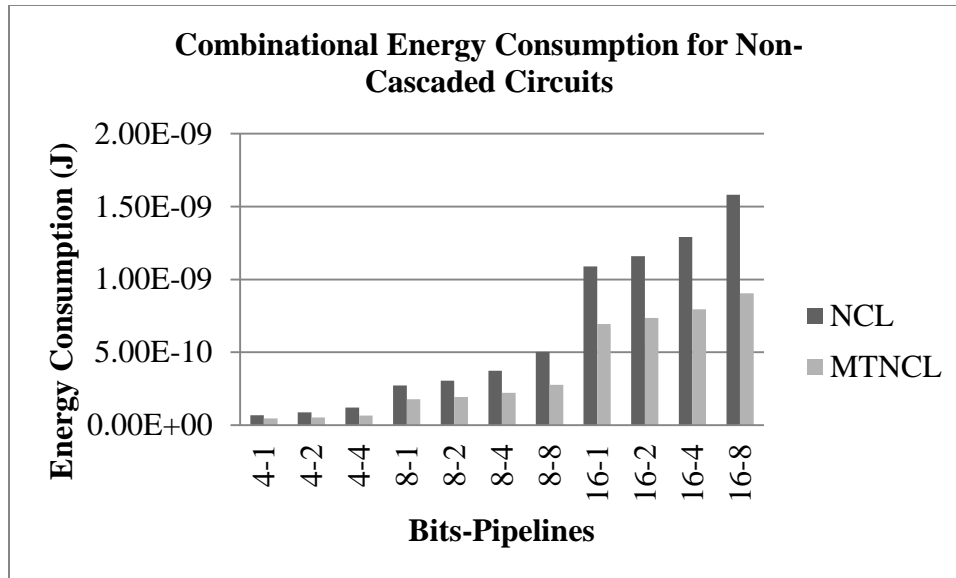


Figure 14. Combinational Energy Consumption for Non-Cascaded Circuits

The sleep trees contain only buffers that are used to drive the sleep nets, which are only in MTNCL designs. The gate counts are in Table 7 and Figure 15. The energy consumption is in Table 8 and Figure 16. Buffers of any size are included in this category, which is why the 16-bit, 2-stage circuit can have fewer buffers than the 16-bit, 1-stage circuit. By default, MTNCL has more gates and use more energy in this category.

Table 7. Sleep Tree Buffer Count for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Sleep Tree	MTNCL Sleep Tree
4	1	0	3
	2	0	5
	4	0	9
8	1	0	9
	2	0	11
	4	0	17
	8	0	36
16	1	0	41
	2	0	37
	4	0	51
	8	0	94

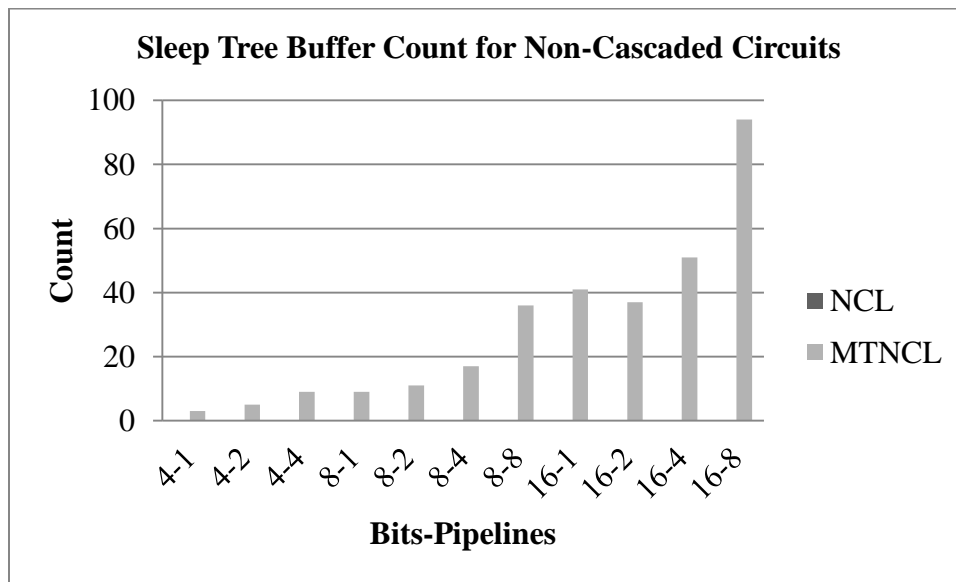


Figure 15. Sleep Tree Buffer Count for Non-Cascaded Circuits

Table 8. Sleep Tree Energy Consumption for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Sleep Tree Energy (J)	MTNCL Sleep Tree Energy (J)
4	1	0	2.49E-11
	2	0	3.56E-11
	4	0	5.44E-11
8	1	0	8.30E-11
	2	0	1.05E-10
	4	0	1.50E-10
	8	0	2.35E-10
16	1	0	3.34E-10
	2	0	3.56E-10
	4	0	4.51E-10
	8	0	6.31E-10

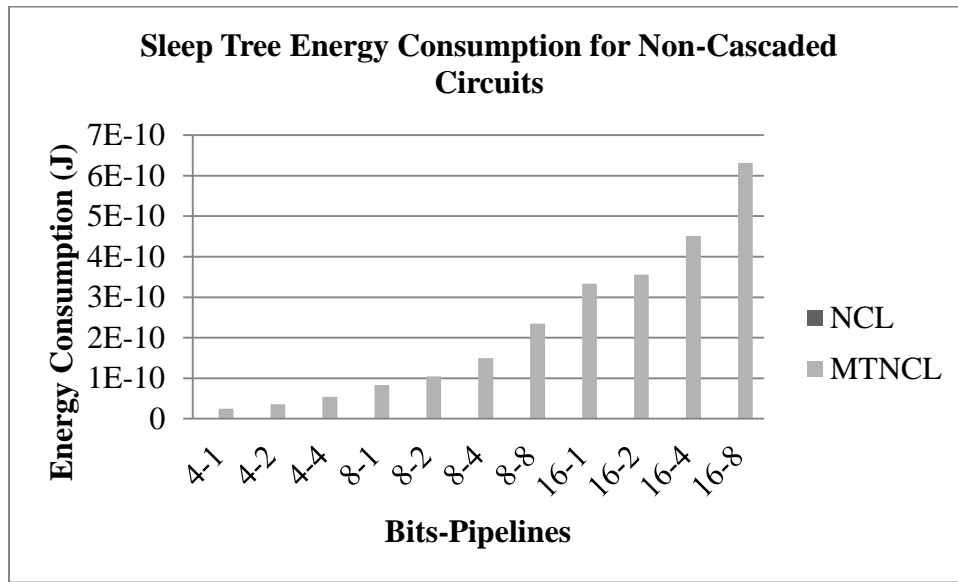


Figure 16. Sleep Tree Energy Consumption for Non-Cascaded Circuits

Every gate in a circuit is included in the total. The three previous sections added together are equal to the total. Table 9 and Figure 17 contain the total gate counts for every non-cascaded circuit. All of the MTNCL gate counts are smaller than the NCL gate counts for each circuit.

Table 10 and Figure 18 contain the total energy consumption for every non-cascaded circuit. The result is mixed. Different design paradigms use less energy for different circuits.

Table 9. Total Gate Count for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Gate Total	MTNCL Gate Total
4	1	183	131
	2	230	171
	4	316	249
8	1	657	454
	2	754	541
	4	961	714
	8	1342	1049
16	1	2465	1694
	2	2674	1858
	4	3094	2214
	8	3940	2942

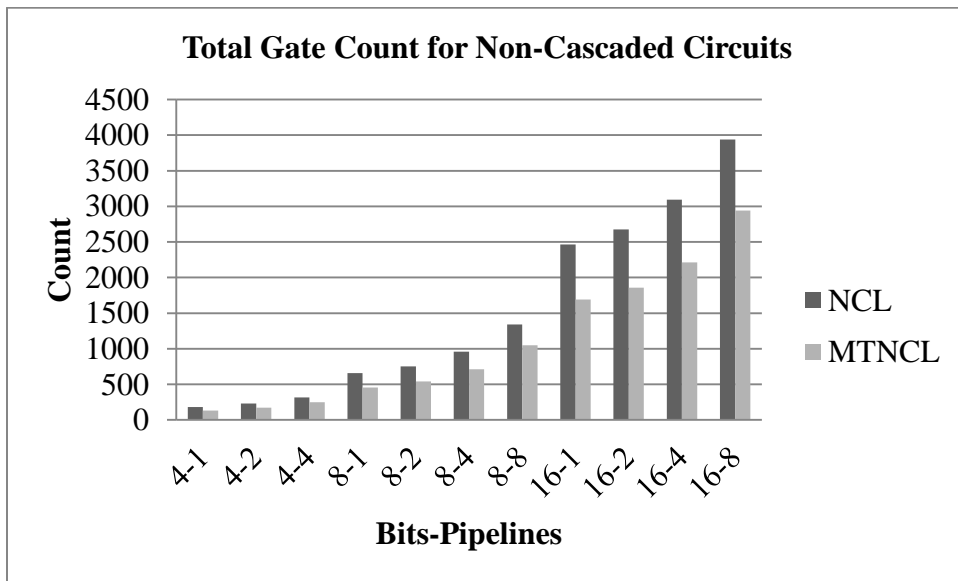


Figure 17. Total Gate Count for Non-Cascaded Circuits

Table 10. Total Energy Consumption for All Non-Cascaded Circuits

Bit-width	Pipeline Stages	NCL Total Energy Used (J)	MTNCL Total Energy Used (J)
4	1	9.92E-11	9.61E-11
	2	1.32E-10	1.35E-10
	4	1.93E-10	2.07E-10
8	1	3.76E-10	3.20E-10
	2	4.33E-10	4.04E-10
	4	5.65E-10	5.71E-10
	8	8.15E-10	8.90E-10
16	1	1.45E-09	1.22E-09
	2	1.54E-09	1.36E-09
	4	1.80E-09	1.71E-09
	8	2.41E-09	2.39E-09

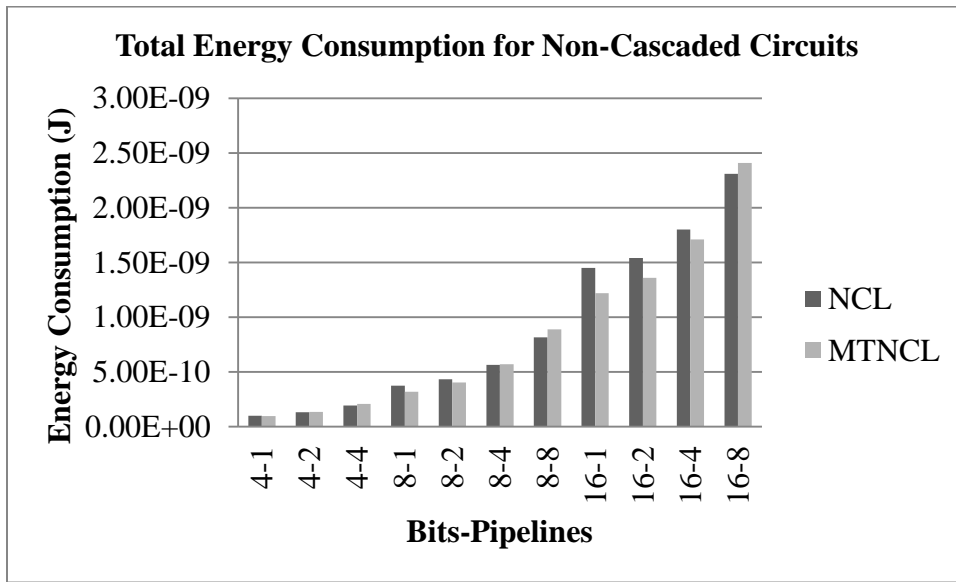


Figure 18. Total Energy Consumption for Non-Cascaded Circuits

4.2 Cascaded Circuit Results

This section contains the results for the 22 cascaded multiplier circuits for both NCL and MTNCL. Each circuit is four copies of one of the 22 non-cascaded circuits. The four copies are

connected in series. The bit-width and pipeline stage columns for all of the cascaded circuits reference the base non-cascaded circuit. Similar to Section 4.1, the results are separated into the three main circuit components and then a total. In each section there is a table and chart for gate count and a table and chart for energy consumption. 500 random test patterns were used to generate the energy usage in order to cover more switching activity profiles in the circuits, but the results were normalized to 100 for comparison with the non-cascaded circuits.

The gate count for the registers in the cascaded circuits is in Table 11 and Figure 19. The slight difference in register counts between NCL and MTNCL is due to the same software optimization. The register energy consumption is found in Table 12 and Figure 20. The cascaded NCL 16-bit, 1-stage multiplier has higher energy consumption due to the large loads on the registers as explained in Section 4.1.

Table 11. Register Gate Count for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Registers	MTNCL Registers
4	1	128	128
	2	240	232
	4	440	432
8	1	256	256
	2	496	496
	4	976	968
	8	1880	1872
16	1	512	512
	2	1008	1008
	4	2000	2000
	8	3984	3976

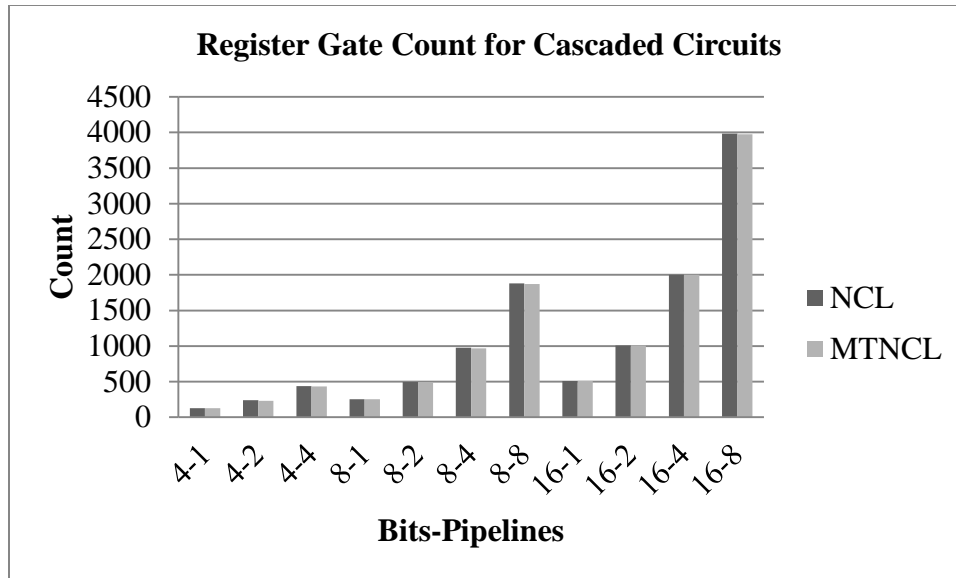


Figure 19. Register Gate Count for All Cascaded Circuits

Table 12. Register Energy Consumption for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Register Energy Used (J)	MTNCL Register Energy Used (J)
4	1	1.65E-10	1.04E-10
	2	2.20E-10	1.85E-10
	4	3.26E-10	3.40E-10
8	1	4.88E-10	2.36E-10
	2	5.06E-10	4.22E-10
	4	8.38E-10	7.90E-10
	8	1.33E-09	1.50E-09
16	1	1.35E-09	7.54E-10
	2	1.54E-09	1.05E-09
	4	2.02E-09	1.83E-09
	8	3.22E-09	3.36E-09

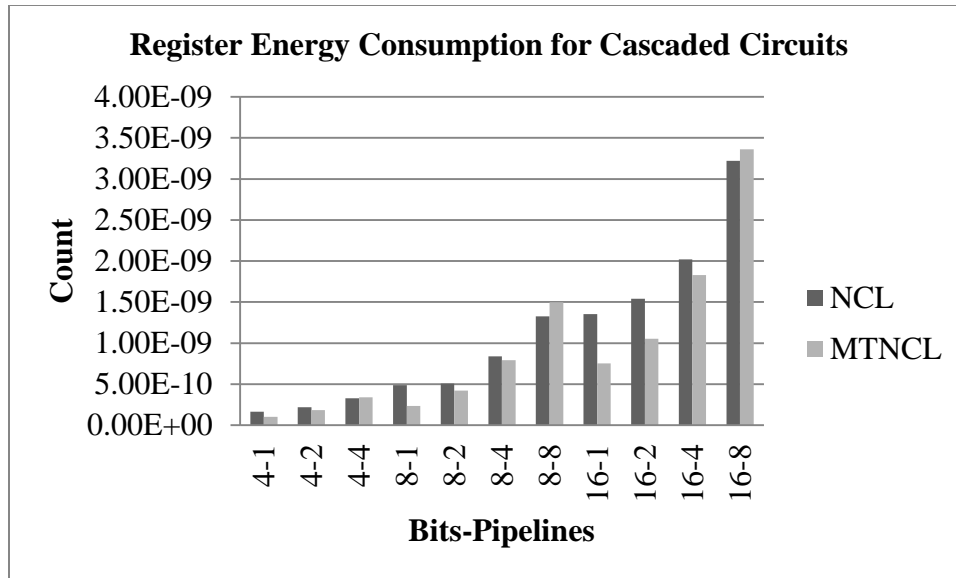


Figure 20. Register Energy Consumption for Cascaded Circuits

Table 13 and Figure 21 are the gate count for the combinational logic in all of the cascaded circuits. MTNCL has fewer gates in every case. Table 14 and Figure 22 are the energy consumed by the combinational logic gates in all of the cascaded circuits. MTNCL uses less energy in every case. As explained in Section 4.1 most of the gate count and energy consumption increase within a particular bit-width is due to completion logic.

Table 13. Combinational Logic Gate Count for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Combinational Logic	MTNCL Combinational Logic
4	1	604	385
	2	680	433
	4	824	529
8	1	2372	1525
	2	2520	1625
	4	2868	1821
	8	3488	2181
16	1	9348	6120
	2	9688	6288
	4	10376	6672
	8	11776	7428

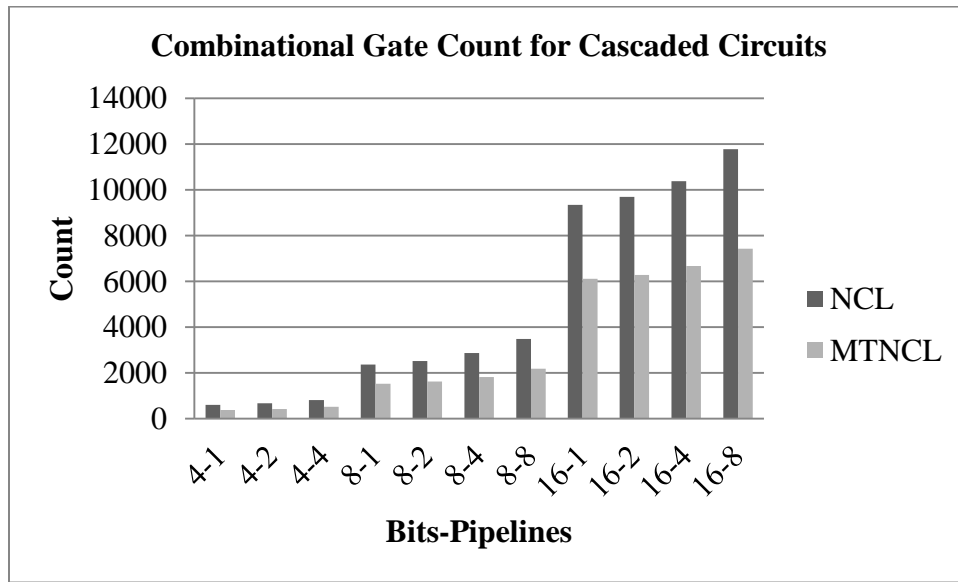


Figure 21. Combinational Gate Count for Cascaded Circuits

Table 14. Combinational Logic Energy Consumption for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL C/L Energy Used (J)	MTNCL C/L Energy Used (J)
4	1	2.56E-10	1.88E-10
	2	3.14E-10	2.16E-10
	4	4.22E-10	2.70E-10
8	1	9.96E-10	7.20E-10
	2	1.23E-09	7.78E-10
	4	1.27E-09	8.90E-10
	8	1.61E-09	1.10E-09
16	1	4.00E-09	2.84E-09
	2	4.58E-09	2.98E-09
	4	5.12E-09	3.22E-09
	8	5.30E-09	3.66E-09

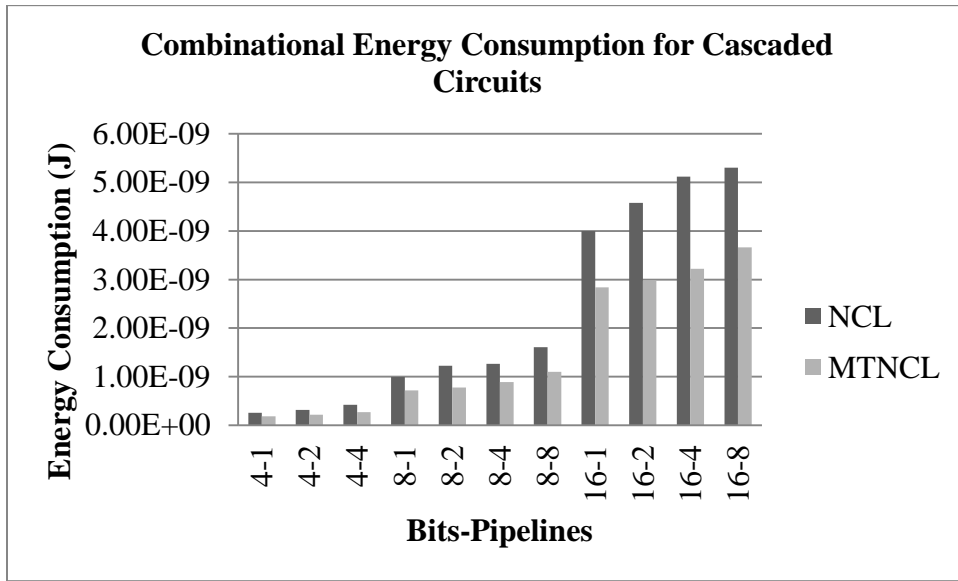


Figure 22. Combinational Energy Consumption for Cascaded Circuits

The sleep tree buffer count is in Table 15 and Figure 23. The energy consumption for the sleep tree buffers is in Table 16 and Figure 24. NCL does not have any sleep-able gates and thus has no sleep nets that would require buffering. Therefore, MTNCL has more gates and uses more energy.

Table 15. Sleep Tree Buffer Count for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Sleep Tree	MTNCL Sleep Tree
4	1	0	11
	2	0	19
	4	0	35
8	1	0	35
	2	0	43
	4	0	67
	8	0	143
16	1	0	144
	2	0	136
	4	0	184
	8	0	364

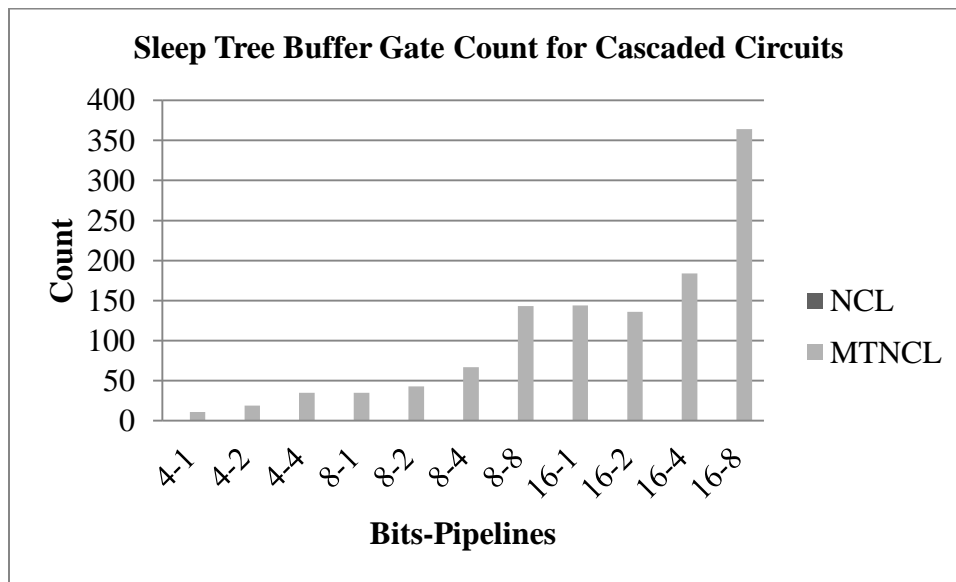


Figure 23. Sleep Tree Buffer Gate Count for Cascaded Circuits

Table 16. Sleep Tree Buffer Energy Consumption for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Sleep Tree Energy (J)	MTNCL Sleep Tree Energy (J)
4	1	0.00E+00	9.28E-11
	2	0.00E+00	1.35E-10
	4	0.00E+00	2.10E-10
8	1	0.00E+00	3.16E-10
	2	0.00E+00	4.04E-10
	4	0.00E+00	5.82E-10
	8	0.00E+00	9.16E-10
16	1	0.00E+00	1.22E-09
	2	0.00E+00	1.31E-09
	4	0.00E+00	1.68E-09
	8	0.00E+00	2.40E-09

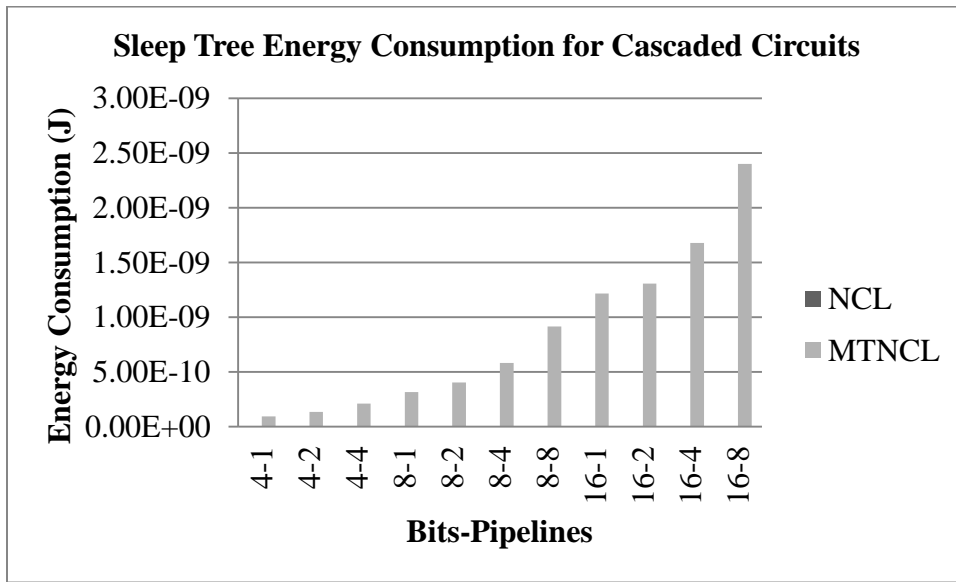


Figure 24. Sleep Energy Consumption for Cascaded Circuits

The total gate count for all cascaded circuits is in Table 17 and Figure 25. Table 18 and Figure 26 contain the total energy consumed by all of the cascaded circuits. MTNCL has lower total gate counts for every cascaded circuit. The lowest energy consumption is mixed between NCL and MTNCL.

Table 17. Total Gate Counts for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Gate Total	MTNCL Gate Total
4	1	732	524
	2	920	684
	4	1264	996
8	1	2628	1816
	2	3016	2164
	4	3844	2856
	8	5368	4196
16	1	9860	6776
	2	10696	7432
	4	12376	8856
	8	15760	11768

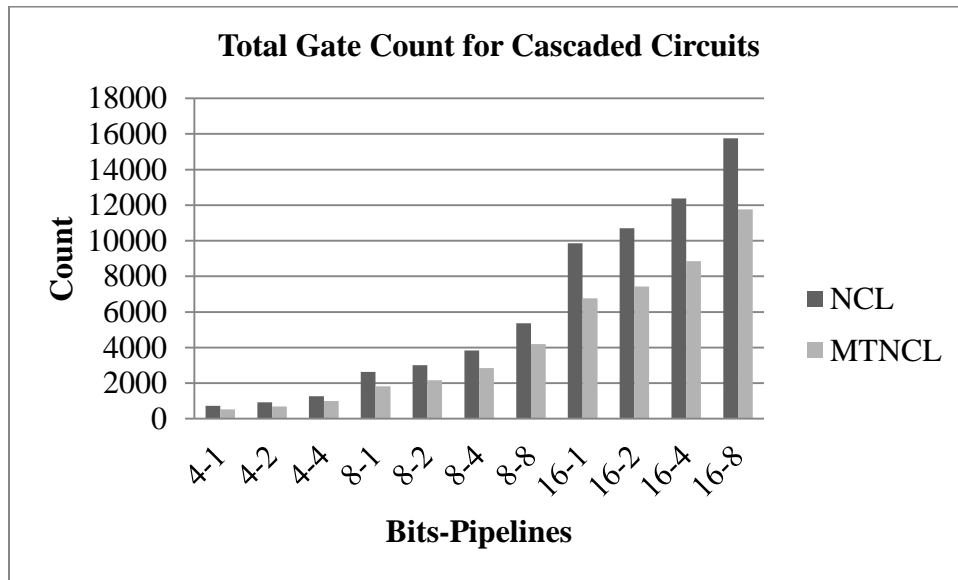


Figure 25. Total Gate Count for Cascaded Circuits

Table 18. Total Energy Consumption for All Cascaded Circuits

Bit-width	Pipeline Stages	NCL Total Energy Used (J)	MTNCL Total Energy Used (J)
4	1	4.22E-10	3.84E-10
	2	5.36E-10	5.36E-10
	4	7.48E-10	8.18E-10
8	1	1.48E-09	1.27E-09
	2	1.73E-09	1.60E-09
	4	2.10E-09	2.26E-09
	8	2.94E-09	3.52E-09
16	1	5.34E-09	4.80E-09
	2	6.12E-09	5.34E-09
	4	7.14E-09	6.72E-09
	8	8.52E-09	9.40E-09

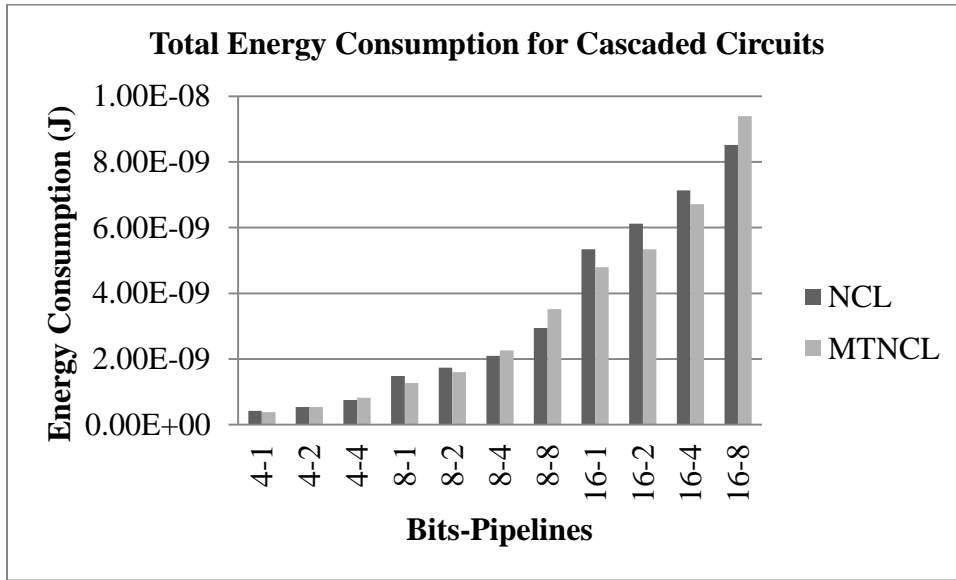


Figure 26. Total Energy Consumption for Cascaded Circuits

4.3 Analysis

The analysis will first focus on the non-cascaded circuits. Since these are the base of all the cascaded circuits, understanding how they compare to each other is important for

understanding the cascaded circuits. The addition of pipeline stages to the multiplier circuit has a complicated effect on energy consumption. The comparison between NCL and MTNCL is not straightforward. Overall, the results in Table 9, Table 10, Figure 17, and Figure 18 indicate that as the gate count in the non-cascaded circuits grow so does the energy consumption when compared to circuits designed with the same paradigm. From Table 9 and Figure 17, the results show that MTNCL always used fewer gates in an equivalent circuit. In contrast, Table 10 and Figure 18 indicate MTNCL does not always consume less energy than NCL. Some of the circuits, such as the MTNCL 16-bit, 8-stage multiplier, have a smaller gate count, but higher energy usage than the NCL version. However, with the 16-bit, 4-stage multiplier the MTNCL version has both a lower gate count and less energy usage.

The combinational logic section of the multiplier circuits clearly favors the MTNCL paradigm over the NCL paradigm. Table 5, Table 6, Figure 13, and Figure 14 both support MTNCL as the better paradigm for this section. MTNCL uses fewer gates and less energy. The combinational logic is largely comprised of the actual multiplier logic. This multiplier logic does not change significantly when pipeline stages are added. However, extra completion logic is needed for each additional pipeline stage. These gates are added to the combinational logic section and make up the majority of gates added to the combinational logic section when pipeline stages are inserted. Buffers outside of the sleep trees will also be categorized into this section. MTNCL has the advantage in this area because its gates are generally smaller and fewer of them are needed since it does not need to propagate a NULL signal. The reduced complexity gives MTNCL a reduced energy consumption advantage over NCL.

The register comparison between MTNCL and NCL shows a distinct switch in energy efficiency between paradigms. Table 3 and Figure 11 detail the gate count as nearly identical.

The slight difference in gate count does not necessarily indicate lower energy consumption. In Table 4 and Figure 12, for example, the NCL 4-bit, 2-stage multiplier has slightly less energy consumption than the MTNCL version even though it has two more registration gates. In each of the bit-widths, once MTNCL reaches a certain pipeline stage it becomes less energy efficient than NCL. This point corresponds to when the total energy switches from being in favor of MTNCL to being in favor of NCL.

The sleep tree buffers are clearly a disadvantage for MTNCL. NCL does not have large sleep nets that require buffer trees to drive. The sleep tree buffer counts in Table 7 and Figure 15 generally increase with pipeline stage and bit-width. The MTNCL 16-bit, 2-stage multiplier, however, does not follow this trend. This circuit uses mostly the larger sized buffer. This makes the gate count lower, but it is still driving a larger capacitance as indicated by the energy consumption in Table 8 and Figure 16. In this MTCNL paradigm all of the combinational logic and registers are sleep-able. As discussed in Section 2.1.2, the sleep signal generated by the early completion logic sleeps the current pipeline stage's registers and the next pipeline stage's combinational and completion logic. By adding a pipeline stage, the combinational logic is split into two groups. Thus, instead of one large sleep net there are two. This reduces the buffers needed to drive the signal since there are now two smaller signals. Additional pipeline stages continue to separate the combinational logic into smaller groups. The downside, however, is each new pipeline stage adds registers and completion logic gates. These gates must also be slept. Eventually, the additional load of the new register and completion logic over takes the gains achieved by breaking the combinational logic into smaller groups with different sleep signals.

The easiest method to detect when MTNCL is going to start using more energy than NCL is by the combinational logic gate count to register gate count ratio for MTNCL. The ratio is

simply the number of MTNCL combinational logic gates divided by the number of MTNCL register gates. The data in Table 19 and Figure 27 empirically suggests that the turnover point is when the ratio drops below 3. This ratio, however, does indicate what number of pipeline stages is the ideal number for that bit-width. Finding the ideal number of pipeline stages is best determined experimentally so far. The reduction in buffers as the result of increasing the number of pipeline stages is highly dependent on the distribution of the combinational logic. There are many factors that must be taken into consideration such as the number of registers required to maintain all of the signals, the number of gates on either side of the new pipeline, the size of the gates on either side of the pipeline. For example, in the MTNCL 16-bit, 4-stage multiplier the sleep trees are not even. The first stage sleep tree has a larger capacitive load on its sleep net than the other three sleep trees. Analyzing the location of pipeline register stages is outside the scope of this thesis. The ratio, however, provides a baseline for when too many stages have been added.

Table 19. MTNCL Combinational Logic to Register Ratio for Non-Cascaded Circuits

Bit-width	Pipeline Stages	Combinational/Register Ratio
4	1	3.00
	2	1.86
	4	1.22
8	1	5.95
	2	3.27
	4	1.88
	8	1.16
16	1	11.91
	2	6.23
	4	3.33
	8	1.87

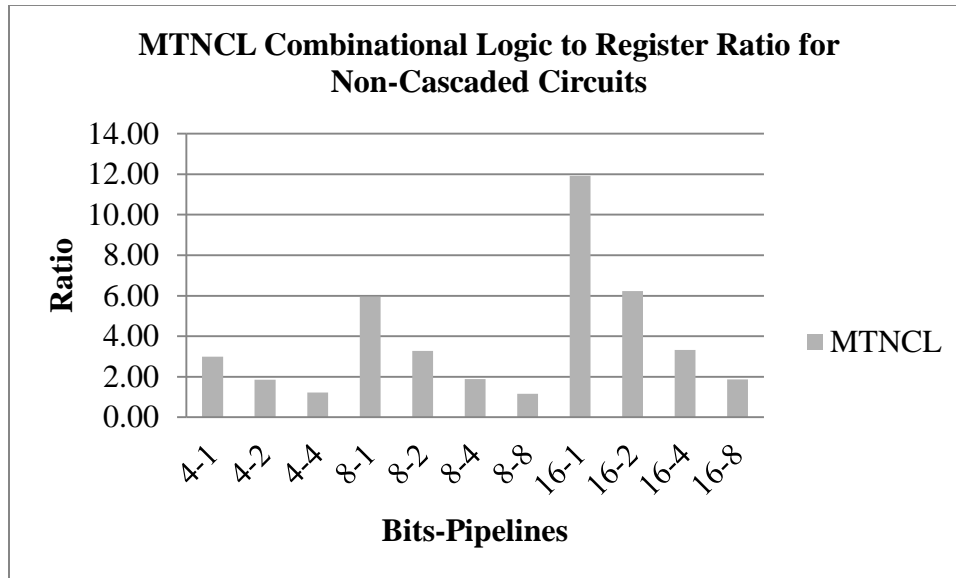


Figure 27. MTNCL Combinational Logic to Register Ratio for Non-Cascaded Circuits

The purpose of the cascaded circuits was to confirm the findings in the non-cascaded circuits. One of the notable results was the MTNCL register trend changed slightly, as found in Table 12 and Figure 20. The MTNCL registers started using more energy than the NCL registers only in the multipliers that had the greatest number of pipeline stages for a bit-width. This deviates from the previous trend where the register energy consumption followed the total energy consumption. The combinational logic energy consumption trend in Table 14 and Figure 22 remained the same as the non-cascaded results. The gate counts in Table 11, Table 13, Table 15, Table 17, Figure 19, Figure 21, Figure 23, and Figure 25 are all directly proportional to the non-cascaded multiplier gate counts. Despite the change in the register trend the total energy consumption, in Table 18 and Figure 26, follows the same trend as the non-cascaded multipliers. The combinational to register gate count ratio of 3 works for these circuits as well.

Table 20. MTNCL Combinational Logic to Register Ratio for Cascaded Circuits

Bit-width	Pipeline Stages	Combinational/Register Ratio
4	1	3.01
	2	1.87
	4	1.22
8	1	5.96
	2	3.28
	4	1.88
	8	1.17
16	1	11.95
	2	6.24
	4	3.34
	8	1.87

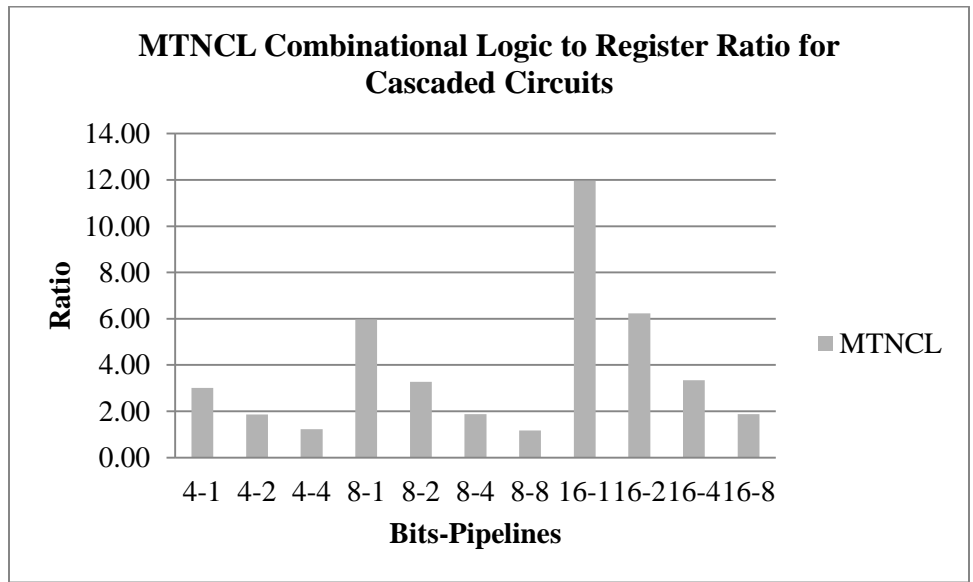


Figure 28. MTNCL Combinational Logic to Register Ratio for Cascaded Circuits

The advantages of MTNCL in terms of active energy can be lost if the circuit becomes overly pipelined. The advantage of MTNCL is in the combinational logic. Too many registers will cause an increase in the sleep tree which will cause MTNCL circuits to become less energy

efficient than NCL. The combinational logic to register ratio can be used as a guide to determine when an MTNCL circuit will consume more energy than its NCL variant.

5. CONCLUSIONS

5.1 Summary

This thesis analyzed the energy efficiency of NCL and MTNCL asynchronous circuit design paradigms across a collection of array multiplier of different sizes and with different numbers of pipeline stages. By utilizing commercially available software energy consumption information and circuit activity data were gathered. Custom scripts were then used to estimate the energy consumption of various parts of each circuit. The data produced by the custom script was then analyzed.

5.2 Conclusions

Care must be taken when designing MTNCL to emphasize its more energy efficient components. The less complex combinational logic gives MTNCL large energy savings over NCL. These savings, however, can be lost if too many pipeline register stages are added. The added stages increase the amount of registers in a design and, consequently, add to the load on a sleep net. Conversely, adding pipeline stages also splits a sleep net into multiple sleep nets, thus, allowing the loads to be driven by smaller buffer trees. A balance must be found between these two effects. The MTNCL combinational logic to register ratio provides a ratio of 3 as a guide to judge when an NCL design will consume less energy than an MTNCL design. This information will be valuable to guide designers when creating NCL and MTNCL pipelined circuits.

5.3 Future Work

There are many avenues for future work to continue from this thesis. Adding synchronous circuits to this comparison would be beneficial in allowing a designer to see the crossovers in energy consumption between circuit size and number of pipeline stages between the various design paradigms. An analysis of pipeline register stage placement will aid in optimizing MTNCL energy efficiency. The complex interplay between the combinational logic, registers, and the sleep trees needs in-depth study to determine practical guidelines. Finally, the custom scripts can be expanded to provide a more detailed breakdown of a circuit. The script has the potential to group and analyze components in numerous ways, such as by pipeline stage or by component activity.

REFERENCES

- [1] J. Rabaey, *Low Power Design Essentials*, New York: Springer Science+Business Media, LLC, 2009.
- [2] S. C. Smith and J. Di, *Designing Asynchronous Circuits using NULL Convention Logic (NCL)*, San Rafael, CA: Morgan & Claypool Publishers, 2009.
- [3] A. Bailey, J. Di, S. Smith and H. Mantooth, "Ultra-low power delay-insensitive circuit design," *IEEE Midwest Symposium on Circuits and Systems*, August 2008.
- [4] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu and J. Yamada, "'1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, August 1995.
- [5] A. Bailey, A. Al Zahrani, G. Fu, J. Di and S. C. Smith, "Multi-Threshold Asynchronous Circuit Design for Ultra-Low Power," *Journal of Low Power Electronics*, vol. 4/3, pp. 337-348, 2008.
- [6] K. Fant and S. Brandt, "NULL Convention Logic™: a complete and consistent logic for asynchronous digital circuit synthesis," *Proceedings of International Conference on Application Specific Systems*, 1996.
- [7] G. E. Sobelman and K. Fant, "CMOS Circuit Design for Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 61-64, May 1998.
- [8] L. Zhou, S. C. Smith and J. Di, "Bit-Wise MTNCL: An Ultra-Low Power Bit-Wise Pipelined Asynchronous Circuit Design Methodology," *IEEE Midwest Symposium on Circuits and Systems*, 2010.
- [9] J. T. Kao and A. P. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 7, pp. 1009–1018, July 2000.
- [10] V. Satagopan, B. Bhaskaran, A. Singh and S. C. Smith, "Automated energy calculation and estimation for delay-insensitive," *Elsevier's Microelectronics Journal*, vol. 38, no. 10-11, pp. 1095-1107, 2007.