8-2018

Efficacy of Multi-Threshold NULL Convention Logic in Low-Power Applications

Brent Bell
University of Arkansas, Fayetteville

Efficacy of Multi-Threshold NULL Convention Logic in Low-Power Applications

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering

by

Brent A. Bell
University of Arkansas
Bachelor of Science in Electrical Engineering, 2014
University of Arkansas
Master of Science in Computer Engineering, 2017

August 2018
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

_____
Jia Di, Ph.D.
Dissertation Director

_____
J. Patrick Parkerson, Ph.D.
Committee Member

_____
Michael S. Gashler, Ph.D.
Committee Member

_____
Jingxian Wu, Ph.D.
Committee Member

ABSTRACT

In order for an asynchronous design paradigm such as Multi-Threshold NULL Convention Logic (MTNCL) to be adopted by industry, it is important for circuit designers to be aware of its advantages and drawbacks especially with respect to power usage. The power tradeoff between MTNCL and synchronous designs depends on many different factors including design type, circuit size, process node, and pipeline granularity. Each of these design dimensions influences the active power and the leakage power comparisons. This dissertation analyzes the effects of different design dimensions on power consumption and the associated rational for these effects. Results show that while MTNCL typically uses more active power and less leakage power than an equivalent synchronous design, the magnitude of this difference can vary greatly and trends can be observed across each of these different design dimensions. Using the results and analysis found in this work, circuit designers will be able to choose between MTNCL and synchronous architectures for a given target application based on anticipated power consumption differences.

ACKNOWLEDGEMENTS

My graduate career would not have been possible without the support and guidance I have received from my advisor, Dr. Jia Di. Thank you so much for providing an environment where your graduate students can thrive and for your immense help in revising this work. I am also extremely grateful for my classmates and colleagues for their help on this dissertation and my previous projects. Finally, I wouldn't be where I am today without the support of my parents and family. Thank you so much for all you have done for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Chip designers face an ever growing challenge of designing integrated circuits that meet complex timing closure requirements across a wide range of operating conditions. This challenge is further compounded by fierce competition in the integrated circuit industry where being late to market for a design can have significant costs. In addition to the added complexity of timing closure, the proliferation of mobile devices and embedded technologies has made balancing performance and power consumption more important than ever before. Due to these challenges, quasi-delay insensitive (QDI) asynchronous design paradigms have received renewed attention for their clockless, correct-by-construction architecture which mitigates the need for timing closure analysis.

However, there are several major barriers to widespread adoption of asynchronous design methodologies in industry including but not limited to a lack of commercial electronic design automation (EDA) tools supporting asynchronous logic design, a lack of engineers trained in asynchronous design methodologies, and a poor understanding of what advantages can be obtained from asynchronous circuits for a specific application. It is unlikely that the first two of these problems will be solved without a concrete understanding of what advantages asynchronous design styles offer in each commercial application. Additionally, from an industry standpoint, any benefit obtained by pursuing an asynchronous design methodology must be large enough to offset any non-recurring engineering (NRE) costs associated with switching to a different design flow. This problem is compounded by the fact that there are many different asynchronous design styles with different characteristics and design challenges.

One asynchronous design paradigm that shows promise in low power scenarios is Multi-Threshold NULL Convention Logic (MTNCL). MTNCL is the application of Multi-Threshold

CMOS power gating to NULL Convention Logic (NCL) [1-2]. MTNCL uses a fine-grain power gating approach to "sleep" logic gates between data operations by turning off high threshold voltage (V_t) transistors in the power-ground path in order to reduce leakage current. Because of NCL's dual-rail architecture, it can be more easily modified for power gating than synchronous designs [3]. In a synchronous design, overhead is added in designing control circuitry to determine when it is safe to sleep certain parts of the circuit. This operation is also timing sensitive and requires careful design by the engineer. MTNCL on the other hand has power gating built directly into the architecture and therefore does not require additional control logic to determine when to sleep the design. The completion detection signals already present in NCL can be used to sleep each pipeline stage of the circuit when no computation is being performed. This action of sleeping bypasses the NULL propagation of NCL and turns off high-V_t transistors within each gate, which reduces the leakage power dissipated when data is not being processed.

When considering the adoption of asynchronous paradigms the designer must take into account both the advantages and disadvantages of the protocol. In some situations MTNCL may have a power benefit, but this is not necessarily the case in every circumstance. Many things could have an effect on the tradeoff between MTNCL and synchronous circuits including process technology node, performance target, operating conditions, architecture type, design flow, and circuit size.

It was previously believed that MTNCL designs were better than synchronous in terms of active and leakage power; however, there has been conflicting evidence supporting this [4-7]. For example, the data in Table 1 appears to indicate a dependency on process between two Finite Impulse Response (FIR) filters with respect to power consumption. For this data, two 16-tap FIR filters were compared across 3 different processes. The percent difference metric in Equation 1

shows how much more or less power the MTNCL design uses compared to the synchronous design with a negative number indicating that the synchronous design uses less power. It is calculated by dividing the difference between the MTNCL and synchronous power consumption by the average of the two and multiplying by 100. When the power for both circuits is greater than zero, this formula has a range of ±200% and does not bias against either circuit type.

$$\boldsymbol{Percent\ Difference} = \mathbf{200} \times \frac{(\boldsymbol{P_{sync}} - \boldsymbol{P_{MTNCL}})}{\boldsymbol{P_{sync}} + \boldsymbol{P_{MTNCL}}} \quad (1)$$

For this FIR filter, the dynamic power consumption of the MTNCL design quickly surpasses that of the synchronous design as the technology node scales down, while the leakage power for MTNCL is much better. In the table below, the power of three different process nodes were compared: GLOBALFOUNDRIES (GF) 32nm and 45nm processes, which are partially depleted SOI processes, and GF 130nm which is a bulk CMOS process. Simulation data in later sections will use the TSMC 90nm process as well.

Table 1: Effects of process node on MTNCL and synchronous power comparison

	Active Percent Difference	**Leakage Percent Difference**
32nm	-99.5%	167%
45nm	-23.4%	139%
130nm	7.11%	47.6%

However, this process effect may be design-dependent as this data is only for one design. Pinpointing the reason for the difference across process technology nodes is challenging as there are many different variables between two given process nodes, not only in the process itself, but in the design flow as well. This dissertation will seek to objectively analyze under what circumstances MTNCL shows benefits over synchronous in terms of power and the driving factors between this power tradeoff.

2 Background

2.1 MTNCL Overview

MTNCL is an asynchronous logic design paradigm based on a combination of NULL Convention Logic (NCL) and Multi-Threshold CMOS (MTCMOS) power gating. Like NCL, MTNCL uses dual-rail encoding to achieve quasi-delay insensitivity [3]. Table 2 shows the encoding scheme used for the NULL spacer and data values.

Table 2: Dual-rail signal

	Rail 1	Rail 0
NULL	0	0
DATA0	0	1
DATA1	1	0
Invalid	1	1

MTNCL also uses 27 fundamental sleepable threshold gates in place of typical Boolean logic gates. These gates arise from every possible combination of 4 input variables and will output a logic 1 value when the number of logic 1 input values meets or exceeds the threshold requirements of the gate. The generic symbol for an MTNCL threshold gate is shown in Figure 1.

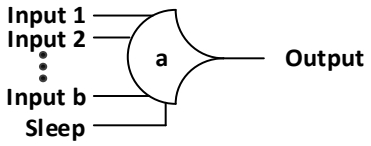

Figure 1: MTNCL threshold gate

Threshold gates are named according to the number of inputs and the threshold of the gate. For example, a TH23m gate is a 3 input gate with a threshold of 2. In order for this gate's output to be asserted, at least 2 of the inputs must be asserted. Some gates place a weight on one or more of the inputs. This is denoted by a 'w' in the gate name. For example, the TH23w2m gate is a 3-input gate with a threshold of 2 and the first input has a weight of 2. Unlike NCL gates these gates do not contain any hysteresis; therefore, even with the addition of sleep

transistors MTNCL gates tend to have a smaller area overhead than NCL gates as can be seen in Table 3.

Table 3: Transistor count of MTNCL and NCL fundamental gates

Gate Name	Boolean Function	NCL Transistor Count	MTNCL Transistor Count
TH12	A + B	6	8
TH22	AB	12	8
TH13	A + B + C	8	10
TH23	AB + AC + BC	18	14
TH33	ABC	16	10
TH23w2	A + BC	14	10
TH33w2	AB+ AC	14	10
TH14	A + B + C + D	10	12
TH24	AB + AC + AD + BC + BD + CD	26	20
TH34	ABC + ABD + ACD + BCD	24	22
TH44	ABCD	20	12
TH24w2	A + BC + BD + CD	20	16
TH34w2	AB + AC + AD + BCD	22	18
TH44w2	ABC+ ABD + ACD	23	16
TH34w3	A + BCD	18	12
TH44w3	AB + AC + AD	16	12
TH24w22	A + B + CD	16	12
TH34w22	AB + AC + AD + BC + BD	22	16
TH44w22	AB + ACD + BCD	22	16
TH54w22	ABC+ ABD	18	12
TH34w32	A + BC + BD	17	12
TH54w32	AB + ACD	20	16
TH44w322	AB + AC + AD + BC	20	16
TH54w322	AB +AC + BCD	21	16
THxor0	AB + CD	20	12
THand0	AB + BC + AD	19	14
TH24comp	AC + BC + AD + BD	18	12

Figure 2 shows the typical structure of an MTNCL threshold gate. Every MTNCL gate is comprised of an NMOS network to set the value of the output to a logic 1 value, a PMOS network to hold the output at zero, and a sleepable inverter. The complement of the Boolean functions in Table 3 are implemented by the *set* and *hold0* portions of each MTNCL gate. The two circled transistors in Figure 2 are high-threshold transistors as well as all transistors in the

hold0 network. There is also one high-V_t transistor in every path to ground within the *set* network. These transistors ensure that there is always an off high-V_t transistor in the path between the power and ground rails in order to reduce leakage power.

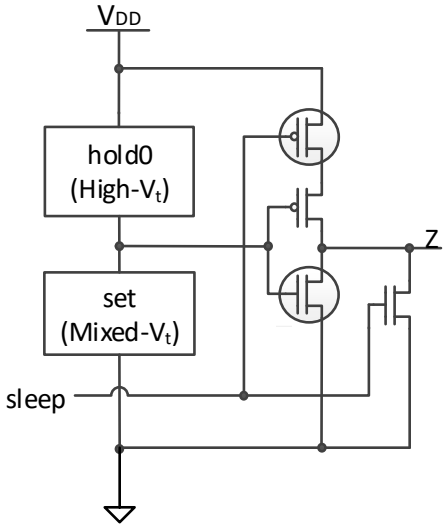

Figure 2: Basic structure of MTNCL gate

2.2 MTNCL Architecture

In an MTNCL circuit, logic alternates between DATA and NULL wavefronts. Completion detection blocks are used at each register stage to ensure DATA wavefronts are always separated by a NULL spacer. At the beginning of a data cycle, the corresponding stage's k_i signal is set to *request for data*. This signal will propagate towards the front of the pipeline through the feedback chain of completion detection registers until a register that contains DATA is reached. The combinational logic following the register containing DATA will then be unslept and the DATA wavefront will be allowed to propagate through the combinational logic until the next register stage. Once all signals at that register have transitioned to DATA, the completion detection logic will cause the k_o signal to transition to *request for null* which will in turn wake up the next pipeline stage. Once this k_o signal propagates through the previous stage's completion detection logic, the register and stage that generate it will be slept producing the NULL

wavefront behind the DATA wavefront. When the pipeline is full, adjacent stages will always alternate between DATA and NULL.

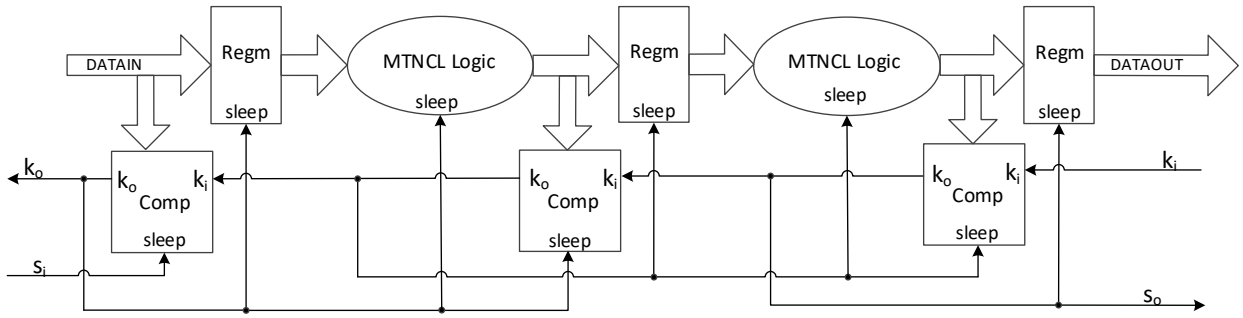

Figure 3: MTNCL framework

If performance is a concern, one can improve the performance of this architecture at the cost of additional MTNCL registers. Since the *sleep* signal bypasses the NULL wavefront propagation that exists in NCL, combinational blocks that are being put to sleep will typically be waiting on data instead of doing any processing due to the quick action of the sleep mechanism. This can be improved however by adding an additional register stage and completion logic block at each stage in the pipeline so that there are two registers back to back with no combinational logic in between. Since each stage alternates between DATA and NULL, this will cause all DATA wavefronts to be in pipeline stages that contain combinational logic at the same time. The next cycle, all combinational logic blocks will be slept very quickly while the DATA resides between the two adjacent registers. This drastically improves the performance of the MTNCL pipeline at the cost of additional power and area.

2.3 Timing Assumption of MTNCL

In recent years, issues with the MTNCL architecture have arisen that need to be addressed. A portion of the robustness in NCL designs, which comes from its QDI properties, is lost due to the fact that the architecture changes required to reduce the leakage power of MTNCL

also adds some delay sensitivities. These timing issues cannot be ignored, especially for smaller process nodes where process variation and routing delays are a larger concern.

The introduction of early-completion to MTNCL was thought to solve MTNCL's delay sensitivity problem [3]. However, early completion solves one delay sensitivity issue and introduces another. This timing issue in MTNCL is similar to a hold-time violation in synchronous logic. Figure 4 shows a single stage in an MTNCL pipeline, and the results of a simulation highlighting the problem are shown in Figure 5. Note that Figure 5 shows only a single rail of the MTNCL register.

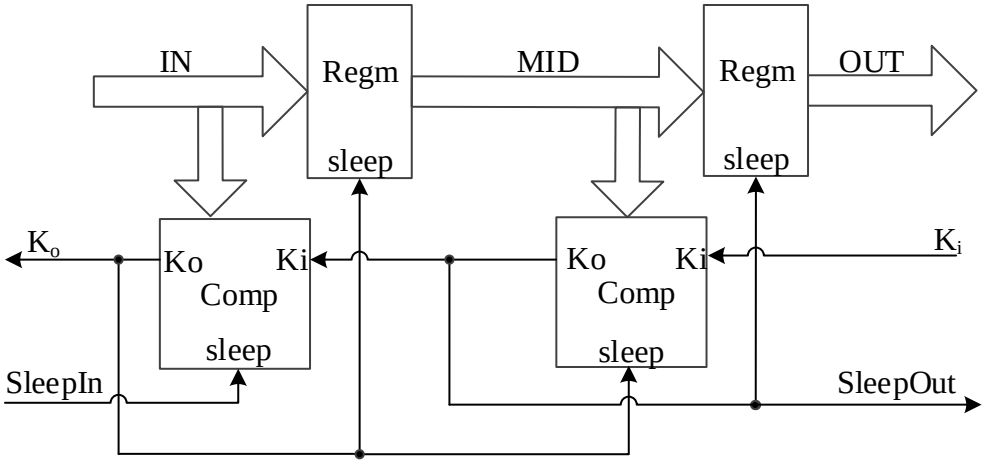

Figure 4: Single stage of MTNCL pipeline

Figure 5: Simulation of delay-sensitivity of MTNCL pipeline in Figure 4

Looking at Figures 4 and 5, when *MID* is NULL and K_i is logic 1, *SleepOut* will also be logic 1 (*rfd*). This means that the second register in the pipeline stage is asleep and *OUT* is NULL. Shortly after a DATA wavefront arrives at *MID*, the completion detection signal *SleepOut* will transition from logic 1 to logic 0. The transition of *SleepOut* to logic 0 along with a NULL wave at *IN* will cause the first completion detection block to output a *rfd* and K_o will transition to logic 1. The transition of *SleepOu*t to logic 0 also wakes up the second register in the pipeline and DATA should be latched from *MID* to *OUT*. At this moment, there a race condition in which the second register in Figure 4 must wake up and latch the DATA wavefront before the first register is slept. In Figure 5, *OUT* begins the transition from NULL to DATA, but due to a high capacitive load, the data is lost before it can be latched by the register.

This race condition can be split into two paths A and B where Path A involves waking the second register and Path B results in putting the first register to sleep. Both paths are initiated by the transition of *SleepOut* to logic 0. The delay of Path A, T_A, is described in the equation below where T_{wireA} is the wire delay from the output of the second completion logic block in

Figure 4 to the sleep input of the second register and T_{latch} depends on both the gate delay through a th12m gate and the load on the output of that gate.

$$T_A = T_{wireA} + T_{latch} \quad (2)$$

The delay of Path B, T_B, is described in the equation below where T_{wireB} is the wire delay between the output of the second completion detection register and the K_i of the first completion detection register, T_{TH22} is the gate delay through an NCL TH22 gate, T_{wireC} is the wire delay between the output of the first completion detection component and the sleep input of the first register, T_{sleep} is the transistor delay through the sleep transistors of the first register, and T_{wireD} is the wire delay from the output of the first register to the input of the second register.

$$T_B = T_{wireB} + T_{th22} + T_{wireC} + T_{sleep} + T_{wireD} \quad (3)$$

It should be noted that this equation makes the assumption that *IN* transitions to NULL before *SleepOut* transitions to logic 0 as is typically the case during normal operation. It also assumes that there is no combinational logic between the first and second registers. However, even if either or both of these assumptions are false, this would not necessarily mitigate the race condition. This means that the race condition exists not only in FIFO registers as previously thought, but in every pipeline stage of MTNCL. In order for MTNCL to function properly, T_A must always be less than T_B. If *MID* transitions from DATA to NULL before the data is fully latched at *OUT*, the data will be lost and the pipeline will not be able to recover.

It is clear from the comparison of the equations for the delay of Path A and B that Path A would typically be the shorter path. This is especially the case for larger process nodes where the gate delay is usually significantly larger than wire delays. However, if the rise time at the output of a register is large enough due to wiring capacitance or insufficient drive strength, Path A could easily be the longer path. This is the case for the simulation shown in Figure 5 where *OUT*

transitions to DATA more slowly than *MID* is slept to NULL. This problem is exacerbated by the fact that MTNCL is not supported by commercial tools and therefore is difficult to analyze for timing.

2.4 Solution to the MTNCL Timing Sensitivity Problem

Without the timing support from commercial synthesis tools, solving the timing problem in MTNCL can be difficult, but ensuring that none of the aforementioned race conditions will cause an issue for a given design is important to avoid loss of data or complete lockup of the pipeline. There are several solutions that could be used alone or in combination to ensure that the data is latched before it is lost.

First, the designer can ensure the delay T_A from Equation 2 is as short as possible. Since T_A is made up of the propagation delay from the output of the completion detection block to the register and the delay required to latch the data, one or both of these delays could be shortened to reduce the overall value of T_A. The propagation delay can be decreased by ensuring there are no additional buffers in the path between the combinational logic and the sleep input of the register and by reducing the length of the routing wire during place and route. The delay required for the delay insensitive (DI) register to latch the data can be reduced either by ensuring the drive strength of the register is large enough to quickly drive any load at the output, or a small buffer could be built into the gate at the library level to ensure the load capacitance at the output of the latch is always relatively constant. This would increase the area and power overhead of MTNCL, but would ensure the T_A delay is a relatively fixed quantity dependent only on T_{wireA} from Equation 2.

Second, the designer must ensure that T_B, shown in Equation 3, is larger than T_A. When T_A is minimized, this requirement will usually be met automatically as there are more gate delays

factored into T_B. However, if the reduction of T_A is not enough, the designer may decide to insert additional buffers to increase the delay in this second path. In the future, it may be helpful to modify the design flow of MTNCL to include timing analysis of these core paths either through commercial tools if possible or through the creation of additional custom tools.

3 MTNCL vs. Synchronous Power Tradeoff

The main focus of this work is to determine the power advantages and disadvantages of MTNCL designs as compared to equivalent synchronous designs and the driving factors behind them. While MTNCL typically uses more dynamic power than an equivalent synchronous design, this is not always the case and depends on several different factors. Additionally, the magnitude of this difference in dynamic power varies across designs. Like the dynamic power, the leakage power usage between the two design architectures also depends on many different factors. The first section of this chapter will detail the MTNCL synthesis method developed for this work as well as an overview of the cell library design and how it may affect the power comparison. Next, the designs compared in this work will be discussed. Then, the final 4 sections will give comparisons according to the active and leakage power across the different design dimensions: design type, circuit size, process node, and pipeline granularity. The data from these comparisons was analyzed to determine what possible factors influence the power tradeoff and why each of these factors contributes to any trends emerging from the data.

3.1 Design Flow

3.1.1 Synthesis of MTNCL Combinational Logic

One of the major barriers to widespread use of asynchronous logic is the lack of commercial design tools. Not only are commercial synthesis tools incapable of synthesizing most asynchronous architectures, but VHDL and Verilog are built around the idea of synchronous circuits making it difficult to describe asynchronous circuits in a meaningful way at a behavioral level. These two things together require asynchronous architectures such as NCL and MTNCL to be designed at the structural level. The time required to design an asynchronous circuit at the

structural HDL level is much greater than when the circuit can be designed at the behavioral HDL level and synthesized.

Some work has been done in the past to create asynchronous synthesis tools [8-10]. However, these synthesis tools do not support or provide only limited support for MTNCL. In order to quickly design MTNCL circuits for use in this work, a method of synthesis was developed using a combination of Python scripts and commercial synthesis tools, such as Cadence Genus Synthesis Solution.

For this method, the circuit designer must design the registration stages at the structural HDL level, while the MTNCL combinational logic can be described at the behavioral level. Any sections of behavioral code must not infer any latches or registration. Using this method, the designer can leverage the power of the commercial synthesis tool to optimize the combinational logic and take into account any timing, power, and load capacitance information present in the Liberty files.

After the HDL file has been written by the designer, any combinational logic blocks undergo a preliminary synthesis separately using Genus. During this stage, an intermediate equation-based Verilog netlist is generated using assign statements. A custom Python script then converts the single-rail Verilog netlist into a dual-rail VHDL behavioral netlist using Boolean algebra. After the dual-rail VHDL file has been generated, it is passed to Genus Synthesis tool along with the desired MTNCL timing libraries. The combinational logic can then be synthesized and mapped to the correct physical gates. This step does not route the sleep signals; however, this is trivial for a combinational logic block and is done with an additional Python script after synthesis has completed. After the combinational logic blocks have been created, they can be inserted back into the top-level HDL containing the MTNCL registers described at the structural

level. The user must then ensure that any register outputs and completion detection signals are properly buffered. There are additional Python scripts to aid in this process.

3.1.2 Cell Libraries

In order to properly understand the power information in the following sections, it is important to take into account the MTNCL and synchronous cell libraries for each process node. An important aspect of designing these libraries is choosing the appropriate threshold voltages for the MTNCL and synchronous libraries. These choices have major impacts on the power comparison between the synchronous and MTNCL designs especially with respect to leakage power.

The choice of transistor threshold voltage would typically come down to design constraints of speed and power, which may be specific to a certain target application. For the purpose of this work, the lowest threshold voltage available in each process was used for the synchronous libraries, which were provided by the vendors, as well as the low-V_t transistor for the MTNCL design. The high-V_t transistors used in the MTNCL library were chosen to approximately match the threshold voltage of the GF 130nm process standard cell library to provide the most accurate comparison between process nodes. The approximate threshold voltages for each of these transistor types are shown in Table 4. These values were obtained by a DC operating point simulation in Cadence ADE.

Table 4: Threshold voltages used in MTNCL and synchronous cell libraries

	PFET Threshold (V)		**NFET Threshold (V)**		**Average (V)**	
Process	**Low-V_t**	**High-V_t**	**Low-V_t**	**High-V_t**	**Low-V_t**	**High-V_t**
45nm	-0.345	-0.460	0.281	0.314	0.313	0.387
90nm	-0.245	-0.420	0.248	0.408	0.247	0.414
130nm	-0.297	-0.426	0.386	0.363	0.341	0.395

3.2 Designs Compared

A series of designs were created in order to determine the power tradeoff between MTNCL and synchronous designs: Finite State Machines (FSMs), Arithmetic Logic Units (ALUs), and Finite Impulse Response (FIR) filters. These designs were chosen because together they represent some of the typical differences in architecture that are thought to affect the power tradeoff between MTNCL and synchronous designs. The FSM is a control based architecture with feedback within the design structure; whereas, the ALU and FIR are both data processing architectures. The FIR was chosen due to the ease with which it can be pipelined and to explore the effects of input pattern between different data processing circuits, while the ALU was chosen for its different input data dependency characteristics and ease of scalability. Some or all of these designs were scaled in size, synthesized across different process nodes, and modified according to pipeline granularity. Each of these dimensions, separately or jointly, could have an effect on the power tradeoff between MTNCL and synchronous designs.

In order to compare MTNCL and synchronous power in a meaningful way, the power difference is calculated for each set of circuits according to Equation 4.

$$Power\ Difference = P_{MTNCL} - P_{Synchronous} \tag{4}$$

A negative power difference for the dynamic power would indicate the synchronous design uses more power than the MTNCL design; whereas, a positive power difference would indicate the MTNCL design uses more power. For the dynamic power comparison, this power difference is calculated using the power-delay product to better compare the energy use between designs of slightly different speeds. The synchronous designs were synthesized and simulated at the average speed of their equivalent MTNCL design in order to provide the most accurate power comparisons possible. For the leakage power comparisons in the following sections, the sign for

Equation 4 is reversed since the MTNCL architecture typically uses less leakage power. This is done so that the leakage power comparison charts are non-negative and easier to read.

3.2.1 Arithmetic Logic Unit (ALU)

A simple unpipelined 8-function ALU was designed in VHDL according to the operations in Table 5. The ALU was designed generically with respect to the width of the operands in order to allow a simple way of scaling the size of the circuit. Registers were placed at the inputs and outputs in order to perform timing analysis and clock tree synthesis (CTS) in the case of the synchronous design.

Table 5: Operations implemented in ALU

Control Input	Operation	Description
000	A + B	Addition
001	A – B	Subtraction
010	A – 1	Minus 1
011	A + 1	Plus 1
100	A && B	Logical AND
101	A || B	Logical OR
110	~ A	Negation
111	A ⊕ B	Logical XOR

Using the synthesis method described in Section 3.1, the MTNCL and synchronous designs were synthesized across all 3 different process nodes and 5 different circuit sizes shown in Table 6. The naming convention A*x* is used to denote the different ALU designs, where *x* is the bit width for the ALU. The gate count for each design after synthesis is shown for rough size estimation in Table 7. Note that comparing size this way does not take into account actual layout size or the size of transistors within the gates. In addition to synthesis, CTS was performed on each of the synchronous designs so that the clock tree power could be simulated as well. In order to perform CTS, the designs were preliminarily placed and routed.

Table 6: Different circuit sizes synthesized for ALU

Design Designation	Operand Width
A4	4
A8	8
A16	16
A32	32
A64	64

Table 7: Number of logic gates in each synthesized ALU design

	45nm		90nm		130nm	
	MTNCL	**Synchronous**	**MTNCL**	**Synchronous**	**MTNCL**	**Synchronous**
A4	171	57	173	73	174	55
A8	340	96	340	151	343	98
A16	699	177	693	280	702	177
A32	1530	448	1505	566	1530	483
A64	2771	1013	2780	1168	2786	1084

For each test case, the circuit was simulated using a random stream of input patterns in order to find the average power consumption. The leakage power of the circuit was also calculated to estimate the power draw when the circuit is idle. The results of these simulations are shown in Table 8 and Figures 6-8. The power-delay product (PDP), often referred to as energy per operation, is calculated by multiplying the average power by the cycle time. In the case of the MTNCL design, the average DATA/NULL cycle time is used; whereas, the PDP calculation for the synchronous design simply uses the clock period.

Table 8: Power delay product of ALU designs

Process	Design	MTNCL (fJ)	Synchronous (fJ)	Speed (GHz)
45nm	A4	393.36	184.89	2.107
	A8	840.51	343.20	1.854
	A16	1686.34	763.04	1.623
	A32	3657.34	1478.52	1.46
	A64	6507.26	3534.13	1.315
90nm	A4	588.60	374.63	0.838574
	A8	1158.83	668.53	0.764312
	A16	2476.71	1353.42	0.591693
	A32	5216.29	2767.04	0.530617
	A64	9670.95	5759.64	0.43726
130nm	A4	1184.37	1056.38	0.3854
	A8	2346.53	2090.09	0.3596
	A16	4845.95	4240.47	0.3286
	A32	10239.93	8757.85	0.275
	A64	18641.64	21144.83	0.2601

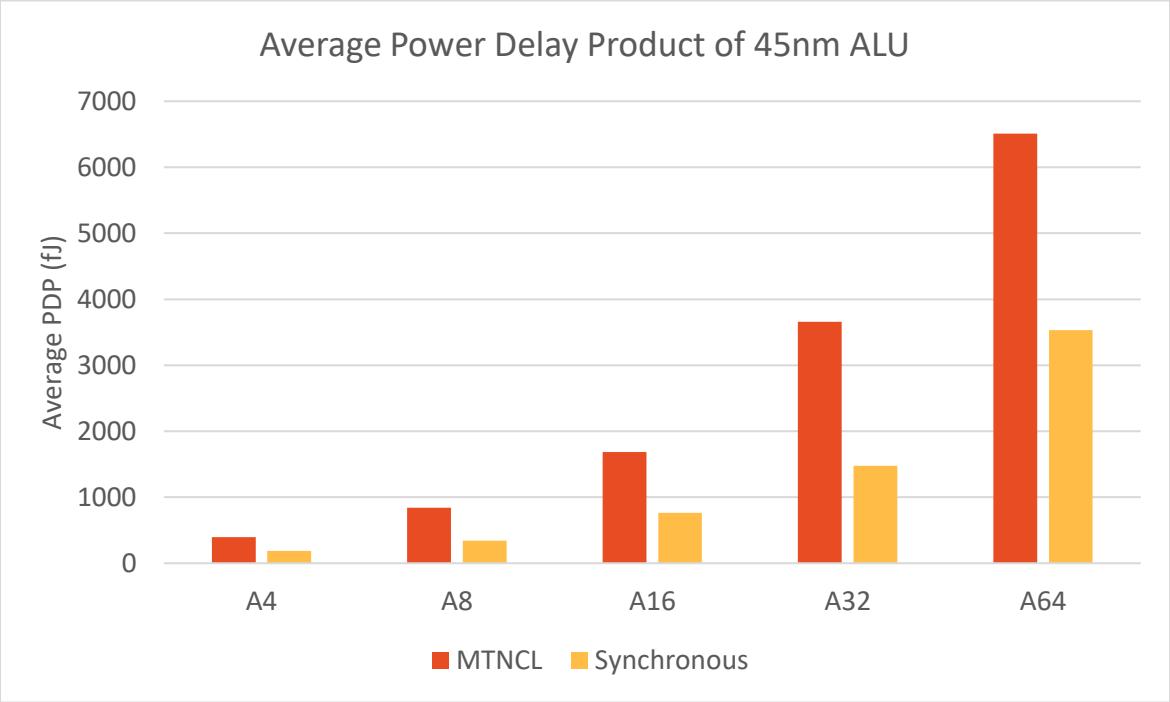

Figure 6: Average power delay product for ALU in GF 45nm process

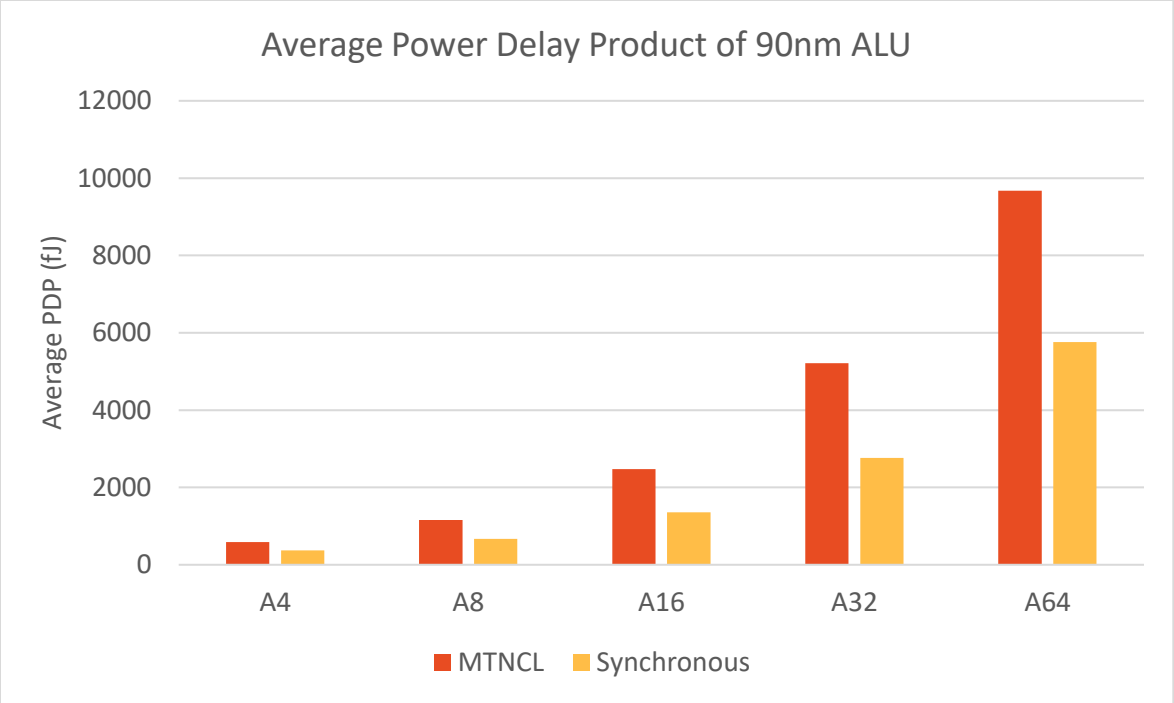

Figure 7: Average power delay product for ALU in TSMC 90nm process

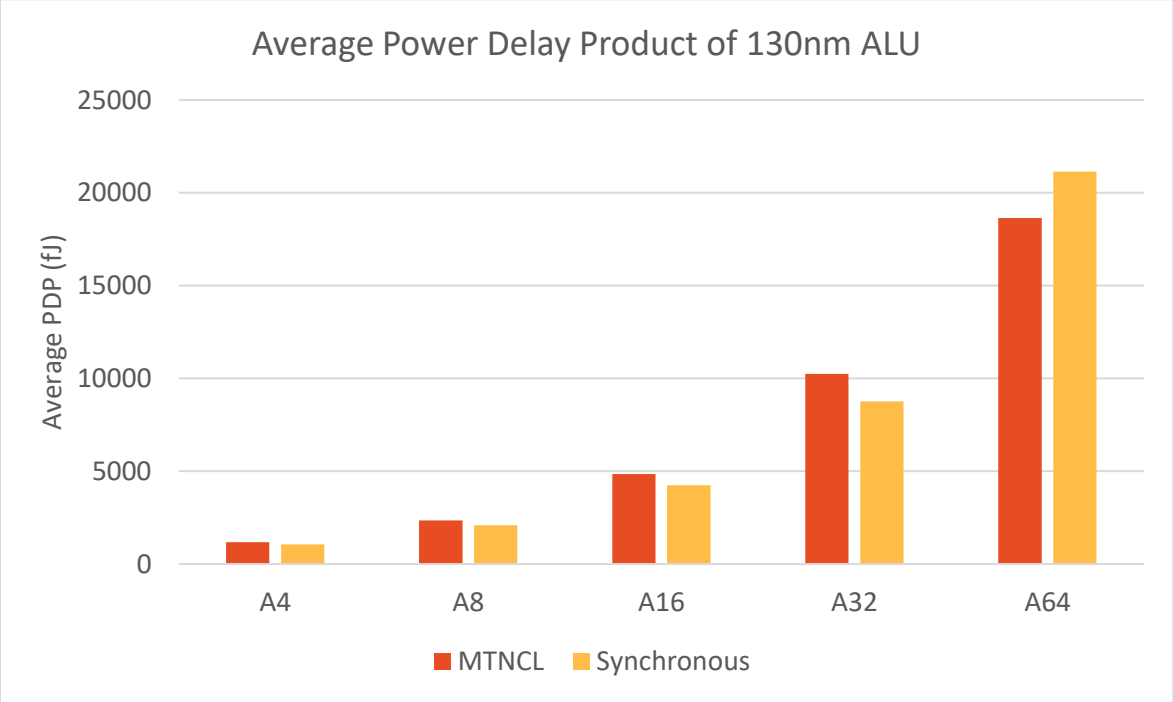

Figure 8: Average power delay product for ALU in GF 130nm process

For every case except for the 64-bit ALU in the 130nm process, the MTNCL design uses more dynamic power than its synchronous counterpart when simulated at the same average

speed. This higher power usage is primarily due to the higher activity factor and larger number of gates in the MTNCL design. The only case that synchronous ALU uses more power is also the only case where the synchronous design uses more logic gates. However, MTNCL does have better leakage characteristics due to its multi-threshold power gating. The benefits of MTNCL in terms of leakage power can be seen in Table 9 and Figures 9-11.

Table 9: Leakage power of MTNCL and synchronous ALU designs

Process	Design	MTNCL Leakage Power (µW)	Synchronous Leakage Power (µW)
45nm	**A4**	4.63	17.38
	A8	9.33	31.32
	A16	20.33	66.12
	A32	46.58	142.00
	A64	75.92	313.02
90nm	**A4**	0.73	6.23
	A8	2.96	11.50
	A16	4.86	22.07
	A32	11.24	40.62
	A64	19.93	83.45
130nm	**A4**	112.23	884.02
	A8	227.55	1503.15
	A16	458.17	4343.07
	A32	1012.55	9524.54
	A64	1812.45	19598.70

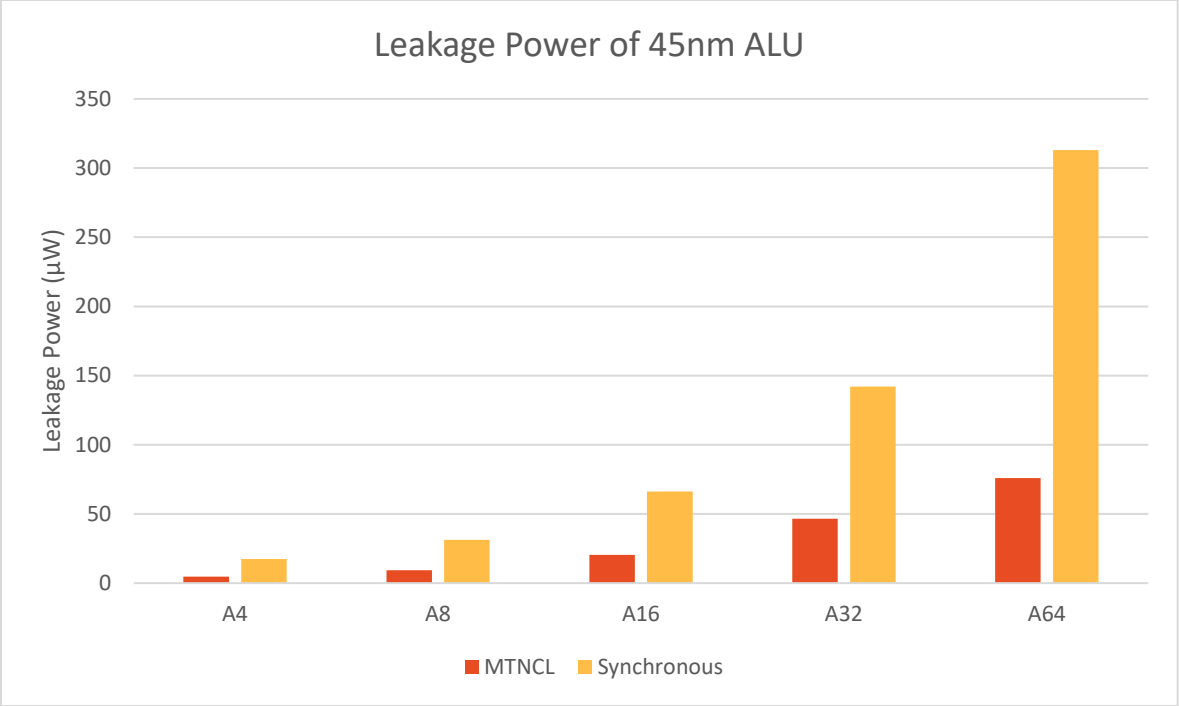

Figure 9: Leakage power of ALU in GF 45nm process

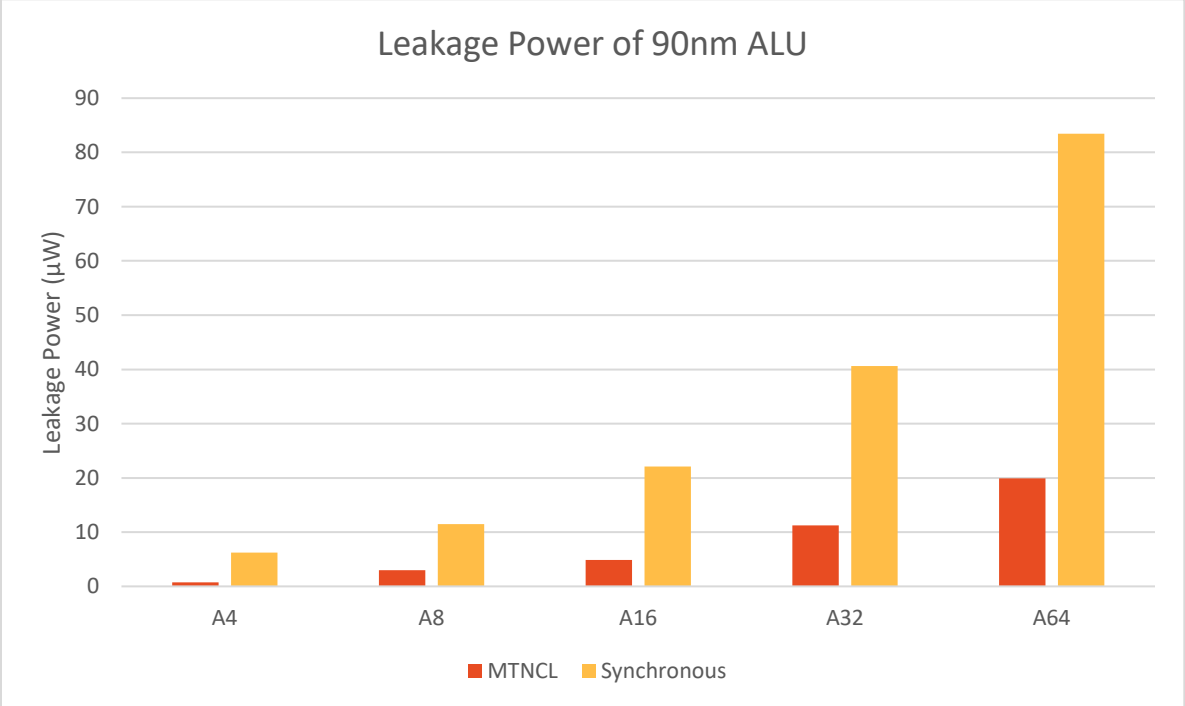

Figure 10: Leakage power of ALU in TSMC 90nm process

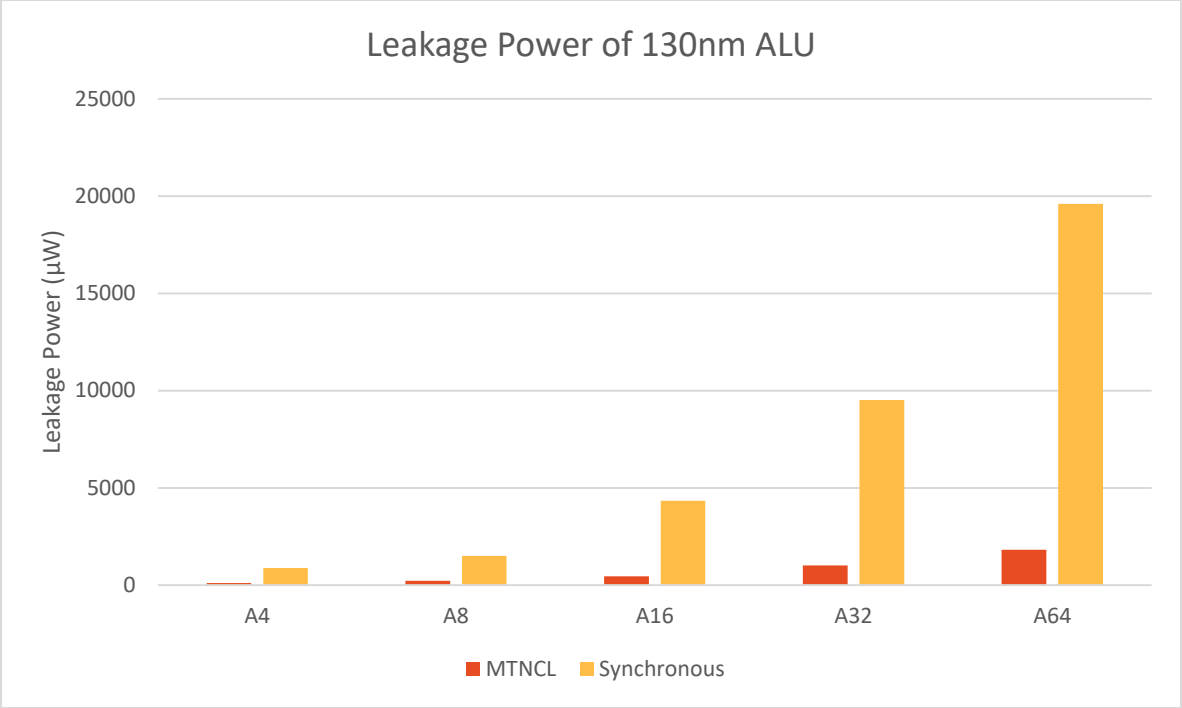

Figure 11: Leakage power of ALU in GF 130nm process

The leakage power of each MTNCL design is much lower than the corresponding synchronous design. This is true for all test cases and is logical since the synchronous designs use mostly low-threshold transistors; whereas, the MTNCL design uses a combination of low-threshold and high-threshold transistors to limit the amount of sub-threshold leakage current. The power tradeoffs due to circuit size and process node will be explored in more depth in Sections 3.4 and 3.5.

3.2.2 Finite State Machine (FSM)

A Mealy type FSM was designed using a randomly generated state table and scaled according to size in two different ways: by increasing the number of input bits and by increasing the number of state bits. Since the size of a state machine depends on the number of input bits and the number of states, these input vector widths were an obvious choice for scaling the size of the FSM. A total of 9 different FSMs were designed using a random state table dependent upon the appropriate number of input bits. These designs are denoted by the designation B*xy* where *x*

is the number of input bits and *y* is the number of state bits. A breakdown of the FSM designs is shown in Table 10. The number of gates in each design after synthesis for each process node is shown in Table 11.

Table 10: Breakdown of FSM designs

Design Designation	Number of Input Bits	Number of State Bits
B22	2	2
B24	2	4
B26	2	6
B42	4	2
B44	4	4
B46	4	6
B62	6	2
B64	6	4
B66	6	6

Table 11: Number of logic gates in each synthesized FSM design

	45nm		90nm		130nm	
	MTNCL	**Synchronous**	**MTNCL**	**Synchronous**	**MTNCL**	**Synchronous**
B22	50	13	53	18	53	12
B24	155	84	157	116	156	94
B26	703	415	705	586	707	445
B42	101	47	104	53	109	44
B44	547	281	556	408	514	322
B46	2533	1377	2541	2049	2528	1435
B62	305	164	313	239	317	184
B64	1669	976	1674	1362	1805	1042
B66	7813	5304	7850	7820	8054	5052

It appears that as the size of the design increases, the number of gates in the synchronous design approaches the number of gates in the MTNCL design. Additionally, the 90nm library tends to use more gates in the synchronous design especially for the larger FSMs.

After these designs were synthesized and implemented in all 3 processes, they were simulated for dynamic and leakage power using randomized input patterns similar to the ALU. The PDP and leakage power data for each of these designs is shown in Table 12 and Table 13 respectively.

Table 12: Power delay product of FSM designs

Process	Design	MTNCL (fJ)	Synchronous (fJ)	Speed (GHz)
45nm	**B22**	110.52	14.89	2.653
	B24	285.60	56.48	2.119
	B26	1090.18	226.03	1.383
	B42	219.00	37.41	2.292
	B44	900.24	153.94	1.519
	B46	3096.92	588.59	0.899
	B62	552.52	96.14	1.988
	B64	2309.71	427.24	0.896
	B66	10035.45	2923.87	0.425
90nm	**B22**	188.33	22.99	0.879
	B24	410.55	84.39	0.728
	B26	1365.15	293.15	0.490
	B42	297.98	66.11	0.797
	B44	1102.93	219.92	0.547
	B46	3912.96	750.26	0.329
	B62	727.91	142.74	0.646
	B64	2815.34	561.80	0.338
	B66	14034.47	3368.12	0.148
130nm	**B22**	358.01	62.72	0.449
	B24	803.32	286.21	0.387
	B26	2373.45	924.99	0.299
	B42	629.12	185.72	0.417
	B44	1862.54	763.66	0.319
	B46	6824.68	1852.96	0.201
	B62	1305.27	461.19	0.354
	B64	5204.20	1289.50	0.209
	B66	20774.25	4169.28	0.103

Figure 12 shows the average power delay product for the MTNCL and synchronous FSMs in GF 45nm SOI process. Similar to the ALU, the MTNCL FSM design uses significantly more energy per DATA/NULL cycle than the synchronous design during one clock period. The graphs of the data for the other two process nodes can be found in Section 3.5 as they follow a similar trend to the power data of the ALU in the previous section.

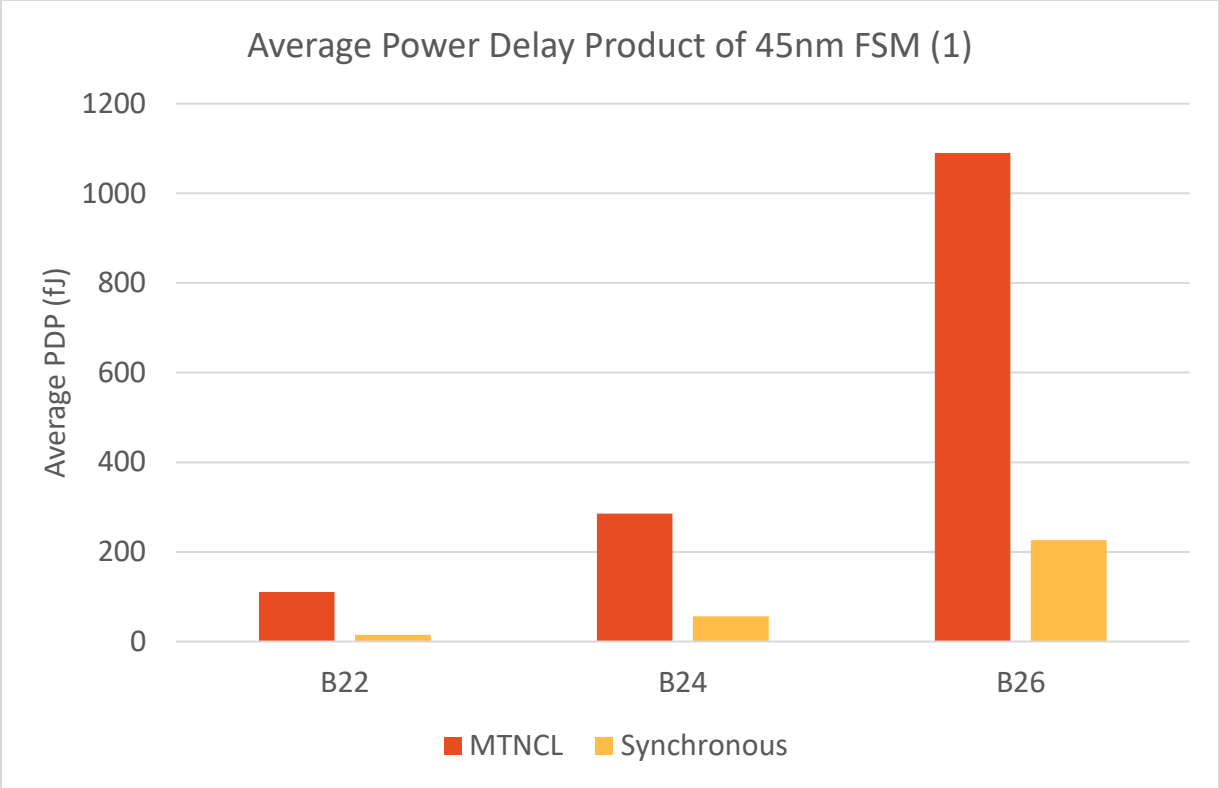

Figure 12: Average power delay product for FSM with 2 input bits in GF 45nm process

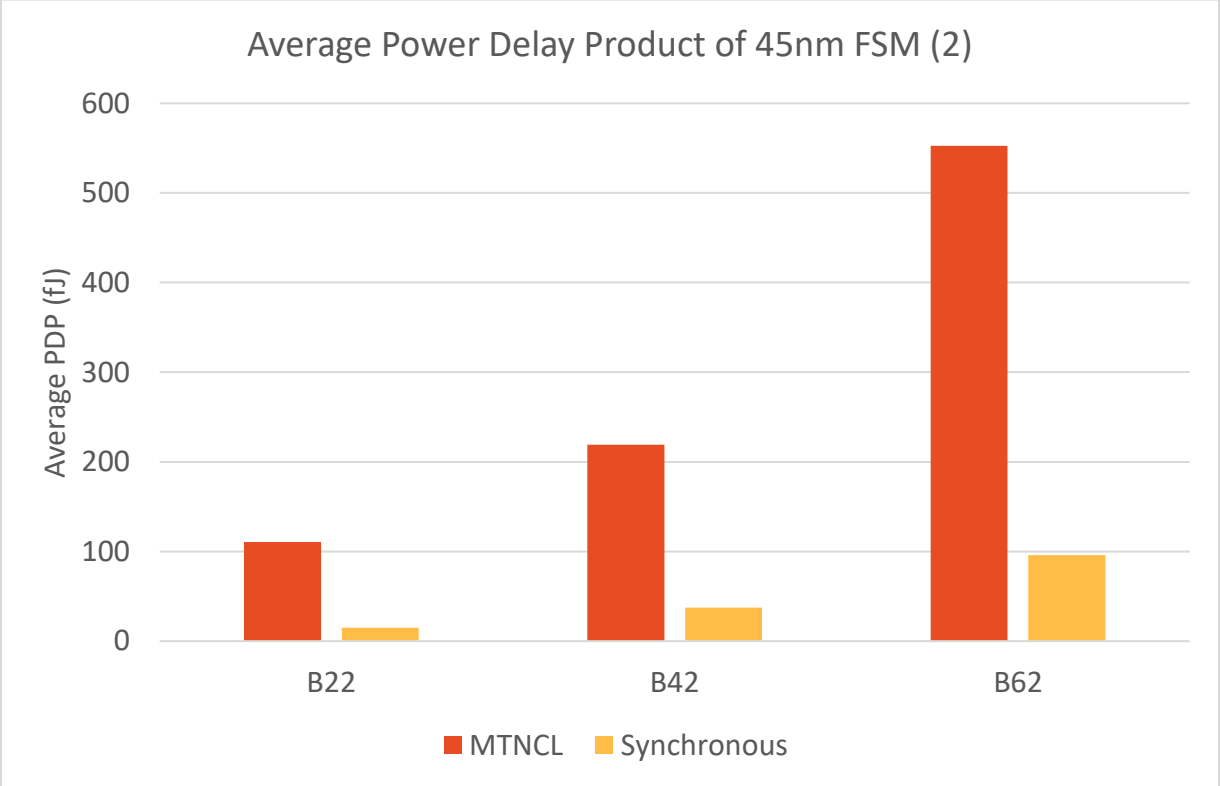

Figure 13: Average power delay product for FSM with 2 state bits in GF 45nm process

Table 13 shows the simulation results for the MTNCL and synchronous leakage of the FSM design. The MTNCL FSM also uses less leakage power in every case just as the ALU did.

Table 13: Leakage power of MTNCL and synchronous FSM designs

Process	Design	MTNCL Leakage Power (µW)	Synchronous Leakage Power (µW)
45nm	B22	3.22	5.85
	B24	6.62	18.79
	B26	24.05	68.87
	B42	4.58	12.88
	B44	17.29	50.32
	B46	77.29	195.80
	B62	9.11	30.27
	B64	52.91	134.90
	B66	244.33	831.00
90nm	B22	0.19	4.85
	B24	0.97	10.11
	B26	3.62	27.86
	B42	0.29	18.64
	B44	2.56	20.99
	B46	12.08	73.35
	B62	0.83	16.26
	B64	8.12	54.66
	B66	36.38	266.38
130nm	B22	0.049	2.035
	B24	0.119	2.791
	B26	0.535	6.203
	B42	0.076	2.136
	B44	0.327	6.016
	B46	1.841	13.736
	B62	0.160	3.404
	B64	1.036	10.897
	B66	5.914	42.873

3.2.3 Finite Impulse Response (FIR) Filter

The FIR filter was designed generically with respect to the number of taps so it could be easily scaled in size. It utilized 7-bit coefficients and an 11-bit input. The basic structure of the FIR filter is shown in Figure 14.

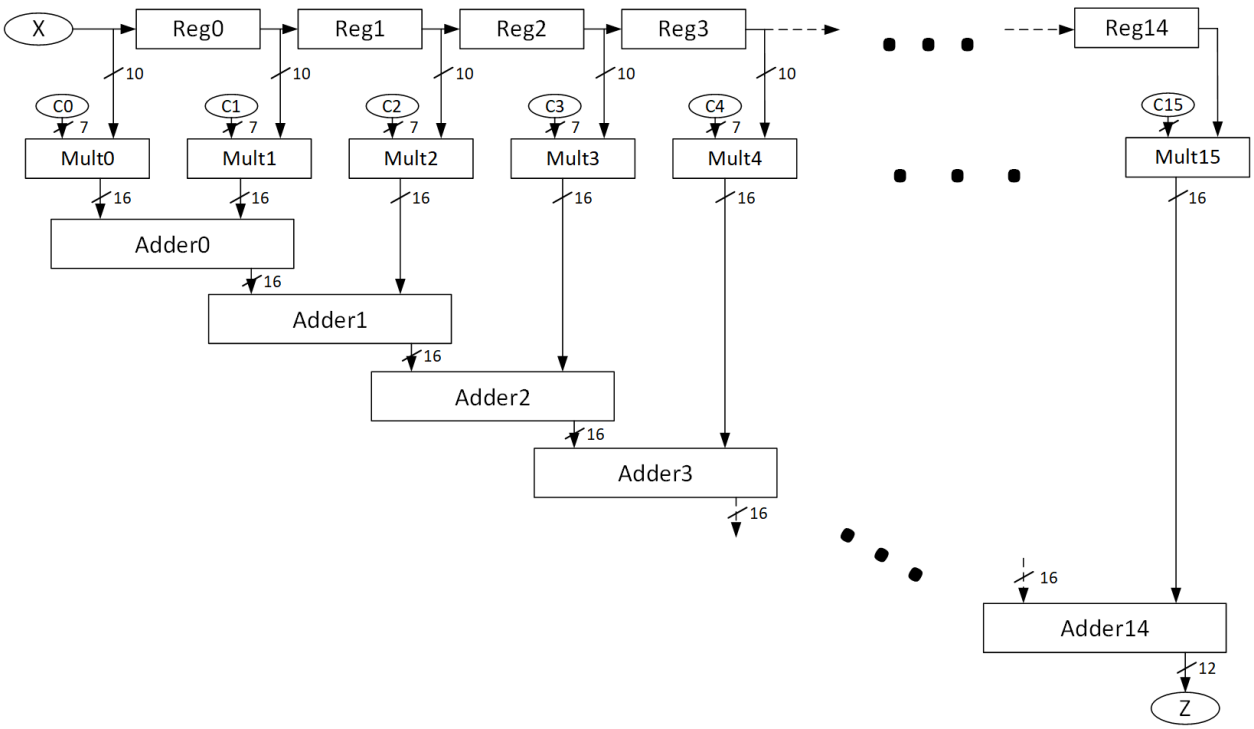

Figure 14: General structure of generic FIR filter

The power of the synchronous FIR filter, like all synchronous circuits, is data dependent with a large dependence on the value of the coefficients. These coefficient bits are typically pre-programmed constant values. Since these bits do not change during operation (or change very rarely), the synchronous circuit is able to take advantage of similarities between successive input patterns. This in turn reduces the switching activity in the circuit. Therefore, the synchronous FIR filter PDP is heavily dependent upon the coefficient values. This is in contrast to the MTNCL FIR filter which has approximately the same switching activity regardless of the input pattern due to its dual-rail architecture. In order to show the data dependency of the FIR, three different test cases were created. Case 1 has 50% of the coefficient bits set to logic 1 and 50% to logic zero (i.e., alternating zeros and ones), while the input to the FIR filter is random. Case 2 has all coefficient bits set to logic 1 and uses the same random stream of inputs as Case 1. Case 3 uses the same coefficient bits as Case 1 and alternates the input data to the FIR between -1

("1111111" in 2's complement) and 0 ("0000000" in 2's complement). The power simulation results after synthesis are shown for each process node in Tables 14-16 below. The number of gates for the unpipelined FIR filter is shown in Table 17.

Table 14: Power data for unpipelined FIR filter in GF 45nm process

	MTNCL PDP (pJ/op)	Sync PDP (pJ/op)	Speed (MHz)	MTNCL Leakage (µW)	Synchronous Leakage (µW)
Case 1	21.63	12.05	646.42	417.75	971.91
Case 2	23.66	13.17	642.63		
Case 3	21.50	14.50	729.98		

Table 15: Power data for unpipelined FIR filter in TSMC 90nm process

	MTNCL PDP (pJ/op)	Sync PDP (pJ/op)	Speed (MHz)	MTNCL Leakage (µW)	Synchronous Leakage (µW)
Case 1	57.58	31.39	164.58	69.04	506.74
Case 2	61.49	35.35	165.29		
Case 3	53.02	35.64	185.65		

Table 16: Power data for unpipelined FIR filter in GF 130nm process

	MTNCL PDP (pJ/op)	Sync PDP (pJ/op)	Speed (MHz)	MTNCL Leakage (µW)	Synchronous Leakage (µW)
Case 1	70.51	151.38	163.70	5.50	69.4
Case 2	77.26	165.40	163.78		
Case 3	70.75	154.63	181.47		

Table 17: Number of gates in MTNCL and synchronous unpipelined FIR filter

	MTNCL	Synchronous
45nm	9491	2908
90nm	9884	7637
130nm	9528	4182

The MTNCL PDP is higher than that of the synchronous FIR for the 45nm and 90nm processes, but is lower for the 130nm process. This is most likely a combination of the logic depth and the threshold voltages of the synchronous library. This discrepancy will be discussed in greater detail in following sections. While the leakage power is lower for the MTNCL design in every process node, the 45nm leakage power values differ by a smaller magnitude than the

other two process nodes. This is most likely due to the lower number of gates in the synchronous design, and the choice of transistors' threshold voltages.

3.3 Analysis of Design Type on Power Tradeoff

One major factor that can impact the tradeoff between MTNCL and synchronous designs is the type of design being compared. For example, control circuitry like the FSM has a very different architecture and gate composition than data processing circuits like the ALU. The effects of these different architectures will be compared in this section.

3.3.1 Dynamic Power and Activity Factor

Dynamic power in digital circuits can be decomposed into two main categories: switching power, which is dissipated during the charging and discharging of capacitive loads within the circuit, and short-circuit power, which is due to the current draw when both the PMOS and NMOS networks are momentarily on during switching. Short-circuit power is generally small when compared to the switching power in a circuit, so it is typically ignored when doing power estimations. The switching power in synchronous designs is often estimated according to Equation 5,

$$P_{switching} = \alpha C {V_{DD}}^2 f \tag{5}$$

where α is the activity factor, C is the capacitive load, V_{DD} is the supply voltage, and f is the frequency of the clock. When comparing MTNCL and synchronous designs within the same process node, V_{DD} is constant. Additionally, the synchronous designs were synthesized and simulated at the average frequency of the equivalent MTNCL circuit for each comparison in this work; therefore, according to Equation 5, any power differences within the same process node arise from either the activity factor or the load capacitance of signals in the circuit. The activity factor takes into account the fact most signals in a circuit do not transition with every transition

of the clock. For the case of the MTNCL design, the average frequency of the input completion detection signal k_i can be used in lieu of a clock frequency as this completion detection signal drives the transition of all other signals in the circuit. This power estimation is typically done node-by-node using a file containing the switching activity information such as a Switching Activity Interchange Format (SAIF) file or Vector Change Dump (VCD) file. For capacitance and timing information, a Synopsys Liberty file is typically used. The combination of Liberty and SAIF files can be used during the synthesis stage of the design flow to get early power information.

In order to estimate the differences in activity for the MTNCL and synchronous architectures, a SAIF file was generated containing the switching activity information for each node in the circuit. Using these values, the total number of node transitions and average activity factor for each given circuit can be calculated. Any differences in switching activity between designs could be a possible driver of any dynamic power differences between them.

Because of the dual-rail architecture MTNCL is based on, the switching activity is typically around 50%. This is due to the fact that during a given cycle, each dual-rail pair will transition from NULL to DATA and back to NULL, and since DATA is encoded using a one-hot scheme, one and only one rail of any dual-rail signal will transition in a given cycle. There is some deviation from this 50% estimate, however, because intermediate signals within an MTNCL combinational logic block do not necessarily maintain their dual-rail structure. By comparison, signals other than the clock within a synchronous design typically have an activity factor of significantly less than 50% due to the signal only transitioning at a maximum of once per clock cycle sometime after the rising edge of the clock. Note that in this analysis any switching do to glitches is ignored. Additionally, signals within a synchronous design will not

necessarily change every cycle if similarities in the input pattern do not require a transition on a particular node.

While the average activity factor for MTNCL does not vary greatly by design or input pattern, the average activity factor of synchronous designs can vary to a much greater degree. The dependence of the activity factor for synchronous designs on input pattern is best shown by the FIR filter test cases outlined in Section 3.2.3. The total average number of switches per cycle and average switching activity factor for both the MTNCL and synchronous designs across all three test cases are shown in Table 18 and Table 19 respectively.

Table 18: Average number of transitions per cycle for unpipelined FIR filter

	MTNCL	Synchronous
Case 1	4,925	960
Case 2	5,661	1,148
Case 3	4,925	1,467

Table 19: Average activity factor for unpipelined FIR filter

	MTNCL	Synchronous
Case 1	44.6%	12.3%
Case 2	51.3%	14.7%
Case 3	44.6%	18.8%

For reference, the MTNCL unpipelined FIR filter had a total of 11,043 signals while the synchronous design had only 7,818 signals. One possible reason for the difference in activity factor for the MTNCL design across the 3 cases is an optimization to the MTNCL FIR architecture, which takes advantage of constant coefficient bits. Without this optimization the activity factor would not depend on the coefficient bits to such a degree for the MTNCL design. As for the synchronous design, the number of transitions per cycle increases as more coefficient bits are set to logic 1 like in Case 2, or as the Hamming distance between successive input bits increases like in Case 3. This effect on power is logical because the coefficient bits are ANDed with input bits during the partial product generation stage of the FIR filter. Any coefficient bits

set to logic 0 will cause those partial product outputs to remain permanently at zero during operation.

From the large difference in number of transitions per cycle between the MTNCL and synchronous designs, it is easy to see why the MTNCL design uses so much more dynamic power than the synchronous design. However, activity factor is only part of the power equation. Capacitive load also plays a large part in the difference between MTNCL and synchronous power usage. These activity factors in this section are averaged across the entire design, but some nodes in the circuit with very different capacitive loads may have activity factors that deviate significantly from the mean. For example the clock in synchronous designs has a high load capacitance and an activity factor of 1. Nodes like this limit the accuracy of such a generalized analysis.

3.3.2 Fanout and Capacitive Load

The fanout and capacitive load of the two different architectures combine with the activity factor to make up the dynamic power difference according to Equation 5. The fanout of the two circuits are quite different especially with respect to their synchronization signals, *clock* for the synchronous designs and *sleep* for the MTNCL design. Both *clock* and the *sleep* signals can have a high fanout and must be timed properly for the circuit to function, albeit the *sleep* signal to a lesser degree. The fanout of the *sleep* signal in MTNCL logic increases with both the size of the combinational logic block of the pipeline stage as well as the number of registers in the pipeline stage. On the other hand, the clock signal fanout depends on the total number of registers in the design, as it is a global synchronization signal. In addition to the fanout of the clock, delay units or extra buffers are often added to certain paths to ensure setup and hold time, rise/fall time, and slew rate targets are met. Even taking into consideration the timing issues

discussed in Section 2.3 the MTNCL sleep signals will not require as many delay units because the MTNCL architecture is still inherently more robust.

Capacitive load depends not only on the fanout, but also on the size of the transistors that are being driven (input capacitance) and the wire routing capacitance. Since this work focuses mainly on pre-physical design and wire lengths are highly dependent on the place and route process, wire capacitances will be ignored except for the purpose of clock tree synthesis (CTS) for the synchronous design. The input capacitance of logic gates in the design depends on the process node and how the standard logic library itself was designed. Typically the gates with the lowest drive strength in the library are approximately minimum sized for the process node, with only small, sparse increases in transistor width to balance the rise-fall times of the gate. Depending on the fanout and speed of the design, either architecture may require larger gates or buffers within the design in order to maintain appropriate rise/fall times and delay targets.

For MTNCL, which operates in an event driven manner, slow rise/fall times may affect the performance of the design, but will not cause the circuit to stop functioning as long as they are not in the critical paths described in Section 2.3. Synchronous designs, on the other hand, are very susceptible to the additional delays caused by under-buffering and could malfunction without proper buffering. For this reason, MTNCL is often able to support smaller gate sizes than synchronous designs, which reduces the capacitive load on each net and therefore the power consumption.

3.3.3 Combinational Logic Gate Composition

The gate composition of the circuit can also have a large effect on the power tradeoff between MTNCL and synchronous designs. This difference arises mainly out of the way logic cells are created for each architecture type. For example, MTNCL circuits use threshold gates to

implement logic functions, while synchronous libraries typically have a wide range of different cell types. Not only do synchronous libraries implement the basic standard logic cells such as AND, OR, NAND, etc., but they also typically implement more complicated Boolean functions such as full adders, multiplexers, and combination gates. These additional gates in synchronous libraries allow for optimizations to be performed at the transistor level for more complex Boolean functions. While it is possible to create more complex gates for MTNCL as well, they are not typically developed as part of the cell library.

This difference in gate libraries can also be seen between the FSM and ALU designs. Since the FSM does not contain any addition or multiplexing, the synchronous design cannot take advantage of these optimized gates. Conversely, the ALU contains many opportunities for these specialized synchronous gates to be implemented. For comparison, an optimized fully-static Boolean full adder requires only 28 transistors; whereas, an MTNCL full adder uses 2 TH24compm gates, 1 TH12m gate, and 1 TH22m gate for a total of 40 transistors. The Boolean MUX gate in the 45nm process uses 14 transistors (without transmission gates), but to create this same gate using only threshold gates requires 2 THxorm gates for a total of 24 transistors. These increases in transistor number are due also in part to the dual-rail architecture of MTNCL since both *rail1* and *rail0* outputs must be implemented.

The average activity factor can also be used to compare across design type to see how differences in design structure affect the amount of switching in the circuit. For each design, a SAIF file was generated containing the number of transitions of each net for a given simulation. This data was then averaged across the total number of cycles and each design size to get the average number of transitions per cycle. The results of this calculation are shown in Table 20 and Table 21.

Table 20: Average activity data for synchronous designs

Design	Toggle Count	Number of Signals	Activity Factor
FIRu	238,301	7,818	15.2%
ALU	176,830	448	19.9%
FSM	41,173	237	11.8%

Table 21: Average activity data for MTNCL designs

Design	Toggle Count	Number of Signals	Activity Factor
FIRu	5,170,089	11,043	46.8%
ALU	517,700	1,150	46.5%
FSM	188,243	1,560	27.1%

One interesting thing to notice when comparing the activity factor across designs is the vastly lower average activity factor for the MTNCL FSM than other MTNCL circuits. This shows just how different the FSM structure is from the other designs. Since the other designs are made up mostly of adders and registers, the dual-rail nature of MTNCL is largely maintained throughout the combinational logic block. However, the FSM does not use adders and this changes the way internal signals are combined throughout the design. In fact, as the design gets larger the MTNCL activity factor decreases as more nodes in the circuit remain constant every cycle.

3.3.4 Leakage Power

The type of design can also have an impact on the leakage power tradeoff between MTNCL and synchronous designs, but only if the design type drastically affects the total transistor width of the circuit. The gates available in the cell library could affect the total transistor width of the design, which proportionally affects the leakage power. High fanout nets will also require transistors with larger widths in order to drive higher capacitive loads on these signal nets.

However, looking at the MTNCL and synchronous ALUs and FSMs, the leakage power depends more on the total number of MTNCL gates and varies very little between designs of the

same size. For example the leakage power difference of the 16-bit ALU is 45.8µW, and the difference for the B26 FSM is 44.8µW. The MTNCL ALU design has 699 gates and the MTNCL FSM has 703 gates. Interestingly, there is a much larger difference in the number of synchronous gates, 177 for the ALU and 415 for the FSM. This is because many of the ALU gates are complex gates like full adders or multiplexers, which typically have a higher transistor count.

3.4 Power Analysis of Scaling Circuit Size

Out of all the possible factors the power difference between MTNCL and synchronous designs can depend on, circuit size has one of the largest effects. The dynamic power difference increases in favor of synchronous as the size of the circuit increases for nearly every circuit compared. This is most likely due to the larger number of nodes and higher switching activity of MTNCL outpacing the additional power needed for the clock buffer tree. However, the leakage power difference scales in favor of MTNCL as the circuit size increases for all of the compared circuits. These two metrics will be discussed in greater depth in the following sections.

3.4.1 Dynamic Power

One theory for the dynamic power tradeoff between MTNCL and synchronous designs is that the power consumption would tip in favor of MTNCL as the size of the circuit increases due to increased buffering requirements of the clock tree. In order to test this, all 3 designs were scaled with respect to circuit size in order to determine the effects of increased circuit size on the power tradeoff between MTNCL and synchronous designs.

The first circuit scaled according to circuit size was the ALU. Looking at Figure 15, the difference in power roughly doubles as the number of input bits for the ALU doubles for every case except the 130nm 64-bit ALU. With the exception of this one case, it does not appear that

this increasing difference in power slows down across the cases tested. The reason for this discrepancy in A64 is explained in more detail in Section 3.5, but most likely has to do with the average speed of the MTNCL design being too fast for the synchronous design clock speed. For the ALU, the number of combinational logic gates and registers scale at approximately the same rate. In other words, doubling the size of the ALU also approximately doubles the number of registers and combinational logic gates for both designs. This means that the capacitive load of the clock and sleep signals scale at approximately the same rate.

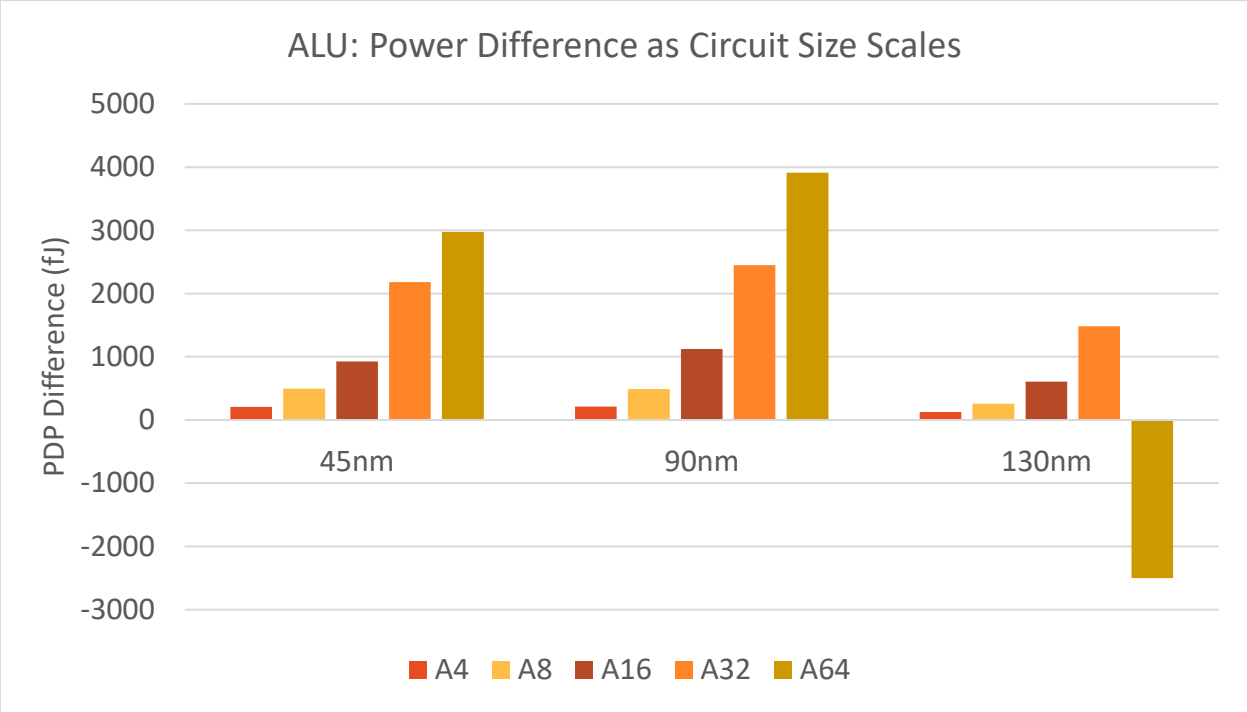

Figure 15: Effects of circuit size on ALU PDP difference

The pipelined FIR filter was also scaled with respect to tap size to see the effects of scaling a pipelined design. Since this design is highly pipelined, it has a much higher number of registers in relation to the number of combinational logic blocks; however, as the number of taps in the FIR filter increases, Figure 16 shows that the power difference continues to increase with an increase in circuit size but at a slightly slower rate than the ALU.

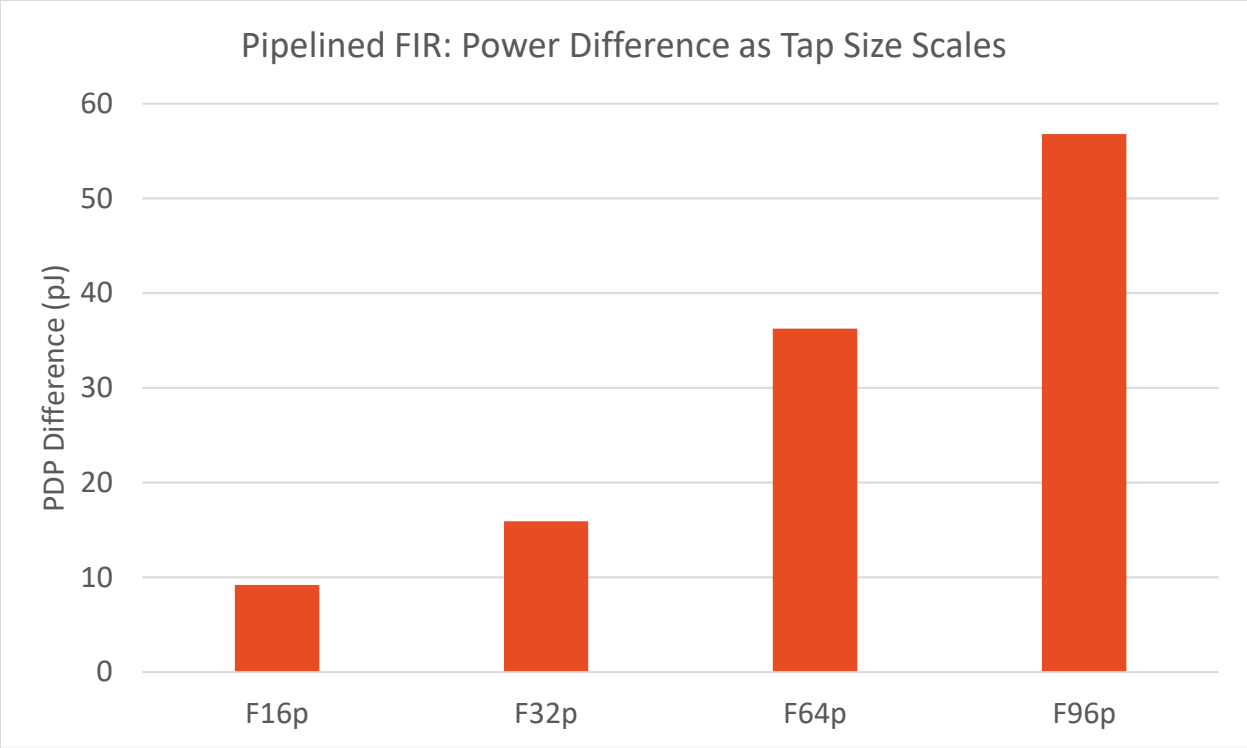

Figure 16: Effects of circuit size on pipelined FIR PDP difference in 45nm process

Finally, the FSM was scaled in terms of size in two different ways by scaling the number of input bits, which determine the next state values, and by scaling the number of state bits. Increasing the number of input bits increases the number of registers and combinational logic in a more proportional way; whereas, increasing the number of state bits increases the number of combinational logic gates exponentially faster than the increase in registers. As shown in Figure 17 and Figure 18, the power difference increases when both dimensions are scaled. In fact, the power difference scales slightly more quickly as the number of state bits increases. This is most likely due to the faster rate at which the combinational logic grows compared to the number of registers in the design. As mentioned previously, an increase in combinational logic increases the capacitive load of MTNCL's sleep completion detection signal in the corresponding pipeline potentially having a large impact on the dynamic power due to the *sleep* signal's activity factor of 1. Increasing the number of register in the design, however, increases the capacitive load of

the *clock* signal in the synchronous design. From these two effects, any difference in the way the combinational logic and registers scale with the circuit's size is bound to have an effect on the power tradeoff between MTNCL and synchronous designs.

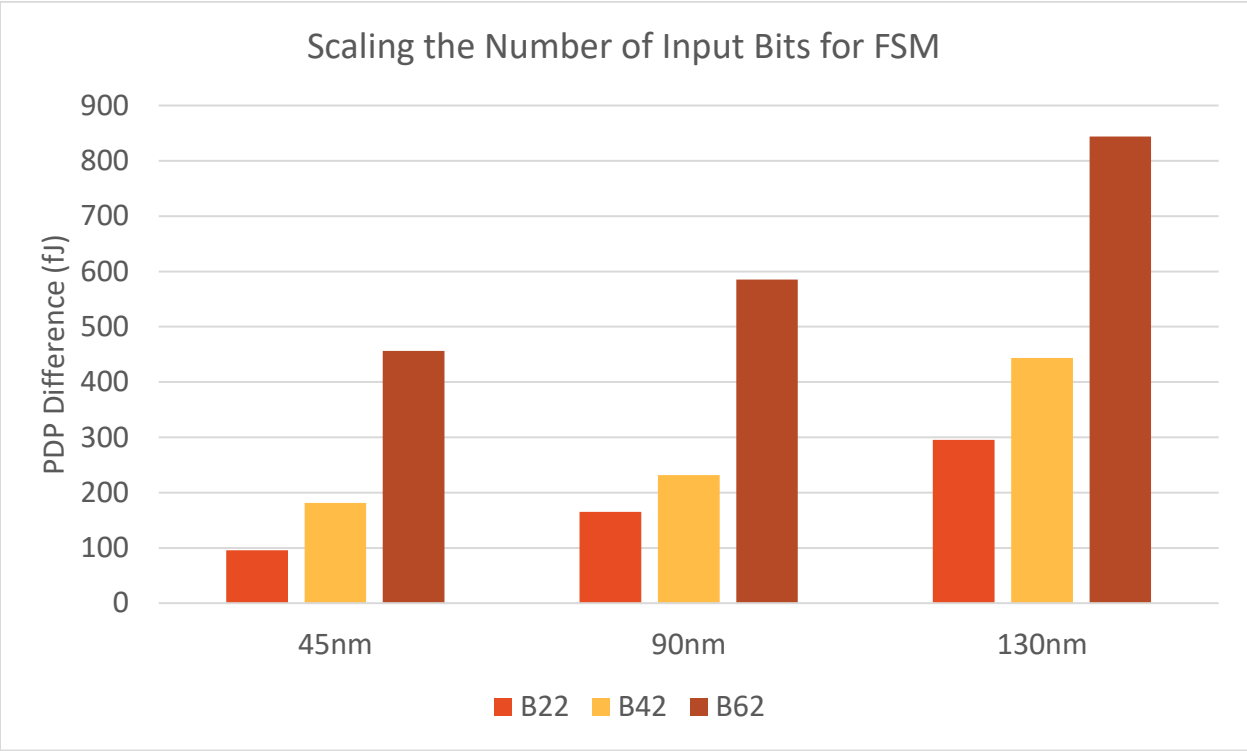

Figure 17: Effects of increasing number of input bits of FSM on PDP difference

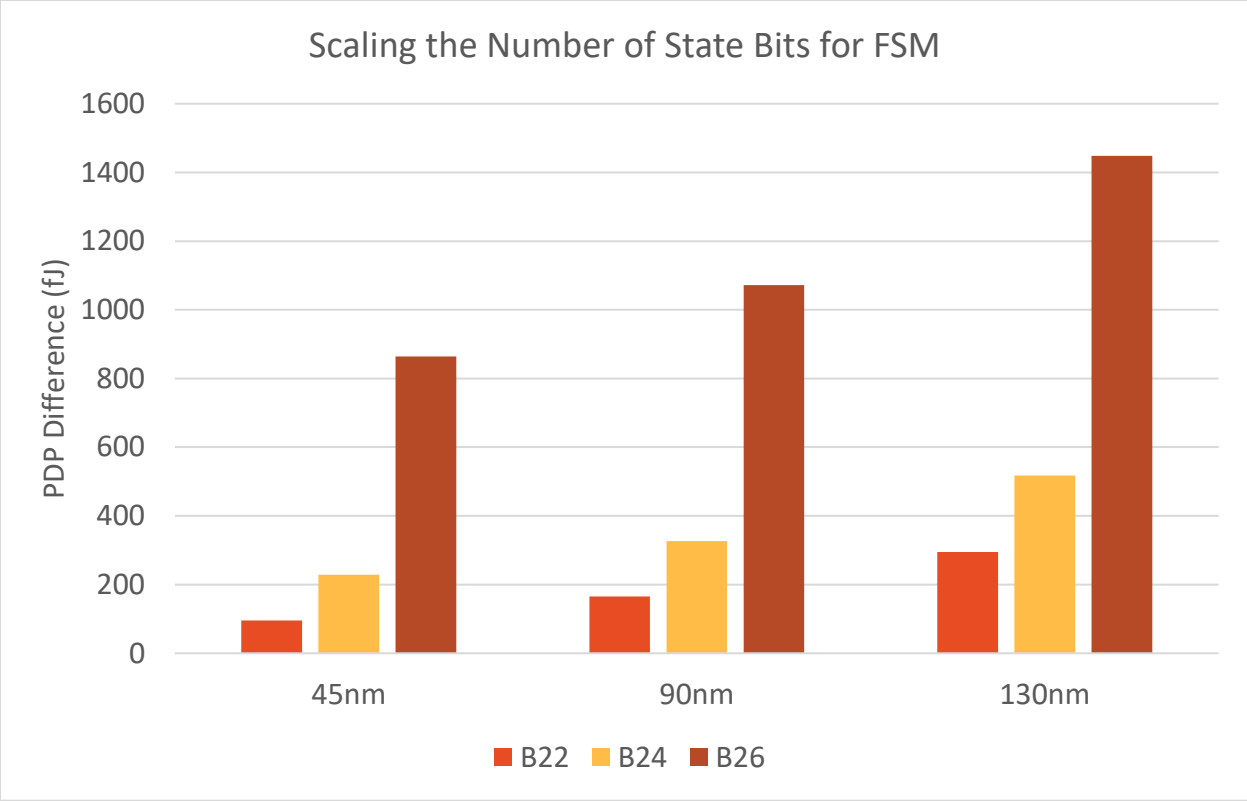

Figure 18: Effects of increasing number of state bits of FSM on PDP difference

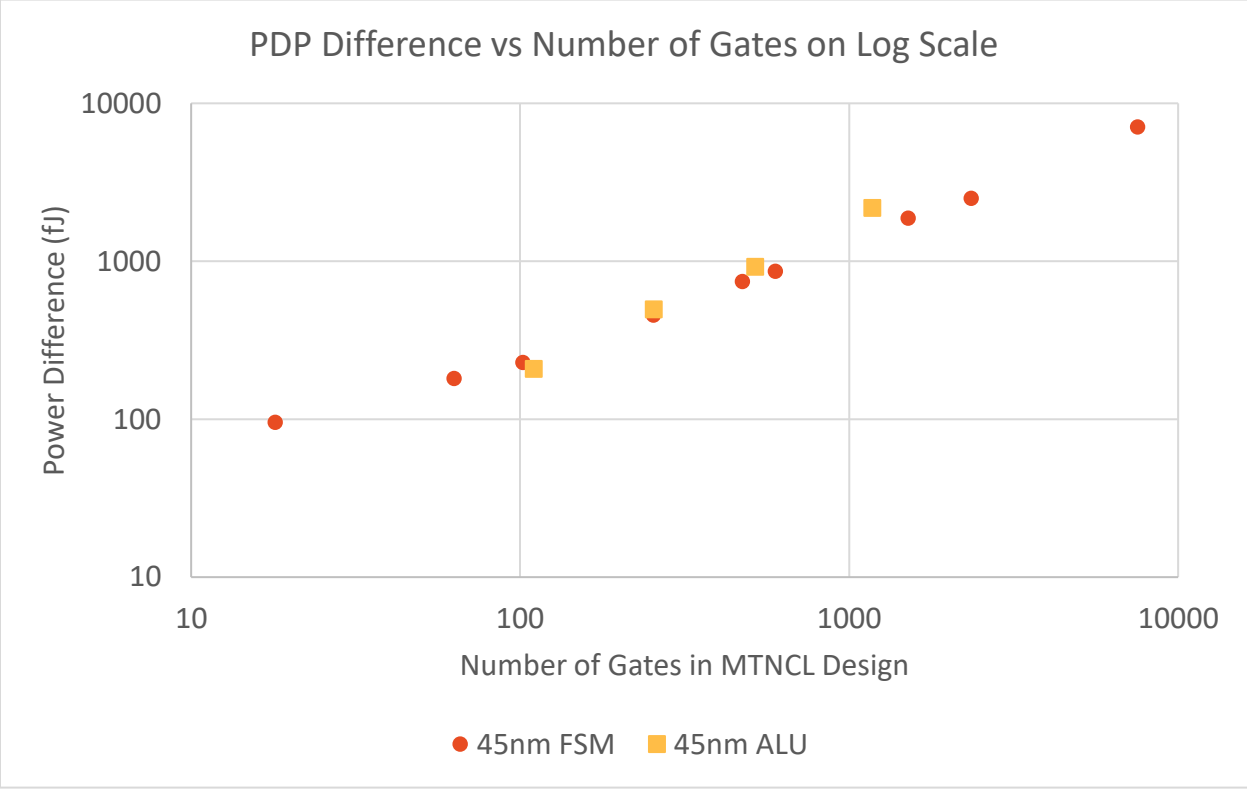

Figure 19: PDP difference vs number of gates in design

From the preceding data, it appears that the MTNCL design continues to use more dynamic power than an equivalent synchronous design even as the size of the circuit increases. Since the power consumed by the clock tree increases with respect to circuit size in synchronous designs, it must be the case that the MTNCL buffer trees for the combinational logic or other factors make up for the increased power of the clock tree. This leads to a relatively linear increase in the PDP difference between the two designs as the number of gates in the designs increase as shown in Figure 19.

Even though the power requirements of the clock tree do not appear to outpace the power consumption of the ALU for the designs compared, it is possible that there would be an effect at the chip level for extremely large designs. In this type of design, the local completion detection signals would have a significantly smaller buffer tree than the one required for the clock. Additionally, the clock would be routed across the entire circuit creating long traces and adding to the routing capacitance. More work needs to be done at the full chip level to see the power tradeoff after place and route.

3.4.2 Leakage Power

Leakage power consumption depends primarily on threshold voltages and transistor widths in the design. MTNCL gates almost universally use less leakage power than synchronous gates due to the high threshold transistors built into their logic. Therefore, it stands to reason that as the number of gates in the circuit increases, the benefit of the MTNCL design in terms of leakage power will also increase as long as the MTNCL gates are not significantly larger in size. In fact, this trend is seen clearly for every design compared. The following figures show the leakage power difference, with a positive number indicating the synchronous design using more power.

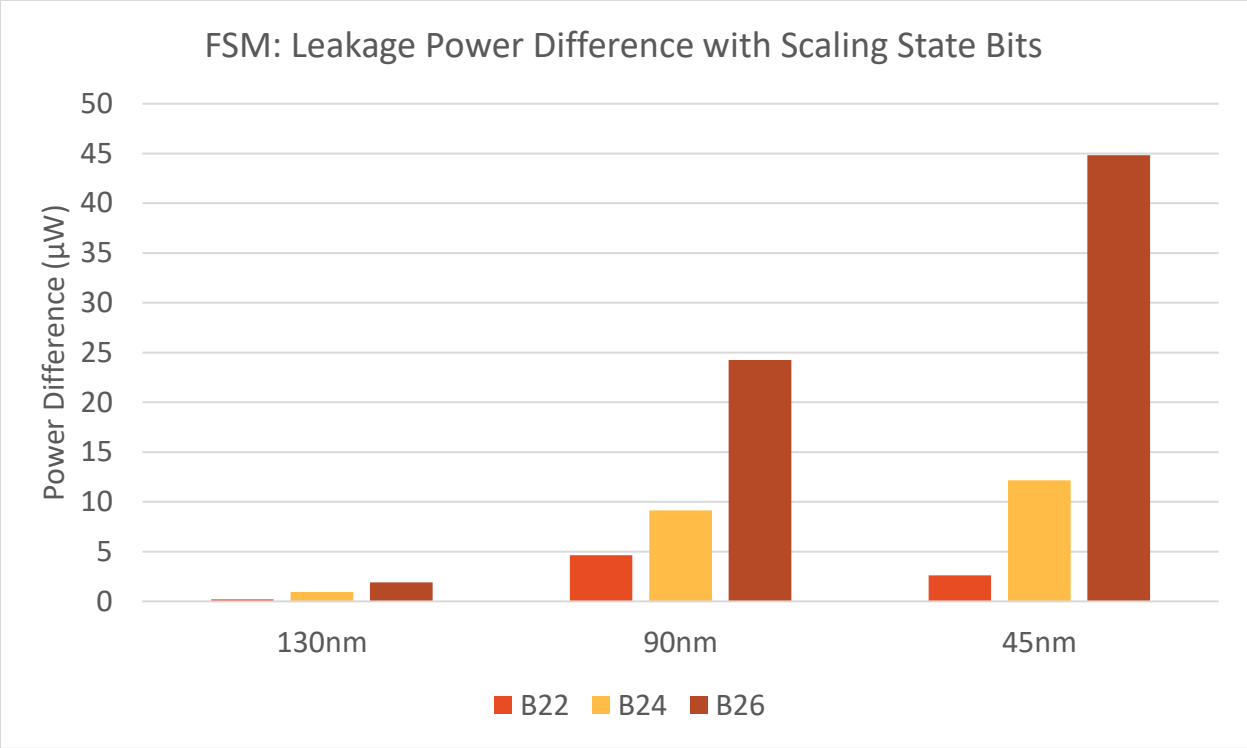

Figure 20: Effect of increasing number of state bits of FSM on leakage power difference

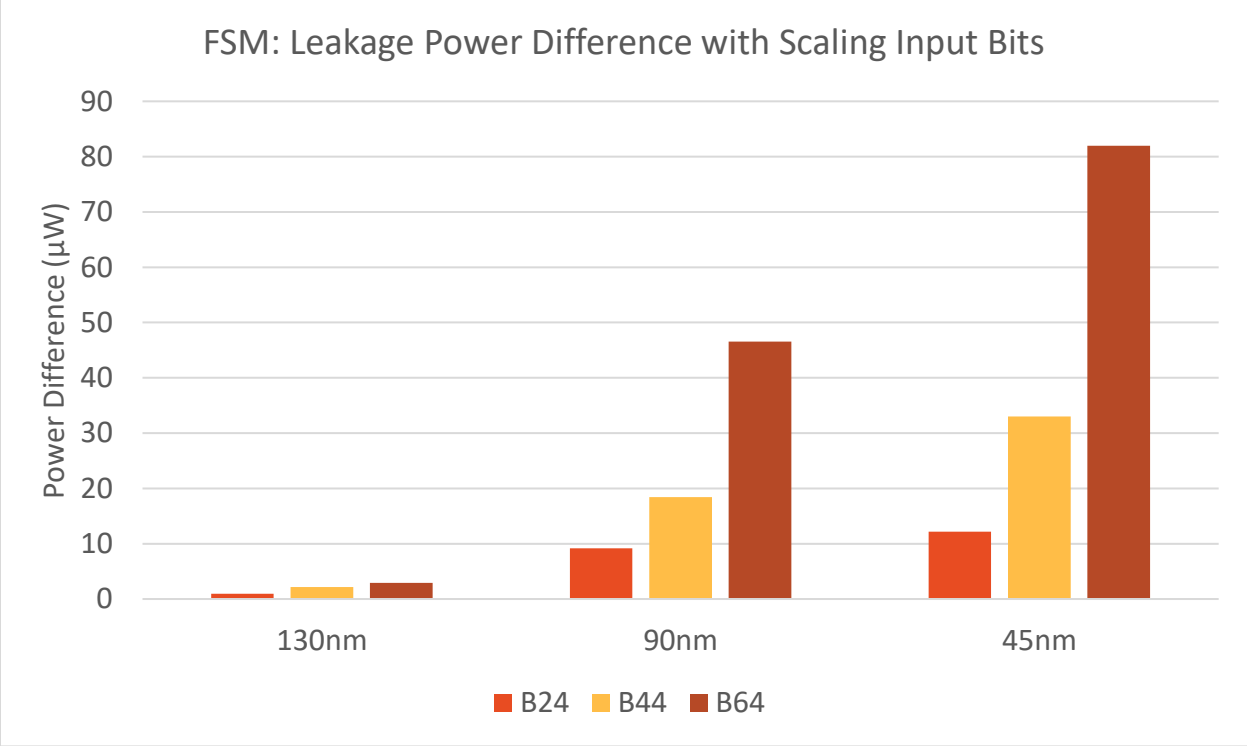

Figure 21: Effect of increasing number of input bits of FSM on leakage power difference

Like the PDP difference in the previous section, the leakage power difference appears to be roughly proportional to the number of gates in the design except in favor of MTNCL. This is as expected due to the higher threshold transistors in each gate of the MTNCL gate library.

3.5 Analyzing Power Tradeoff across Process Nodes

3.5.1 Dynamic Power

Process node also has a substantial impact on the power tradeoff between MTNCL and synchronous designs. However the absolute difference in the PDP for the two architectures does not show a clear trend across all the designs. For some circuit types, the power difference increases as the process size decreases, while for others the power difference decreases. This could explain in part why some research into MTNCL in the past found that MTNCL was better in terms of dynamic power than an equivalent synchronous circuit since much of the previous research is at larger process nodes like the GF 130nm process.

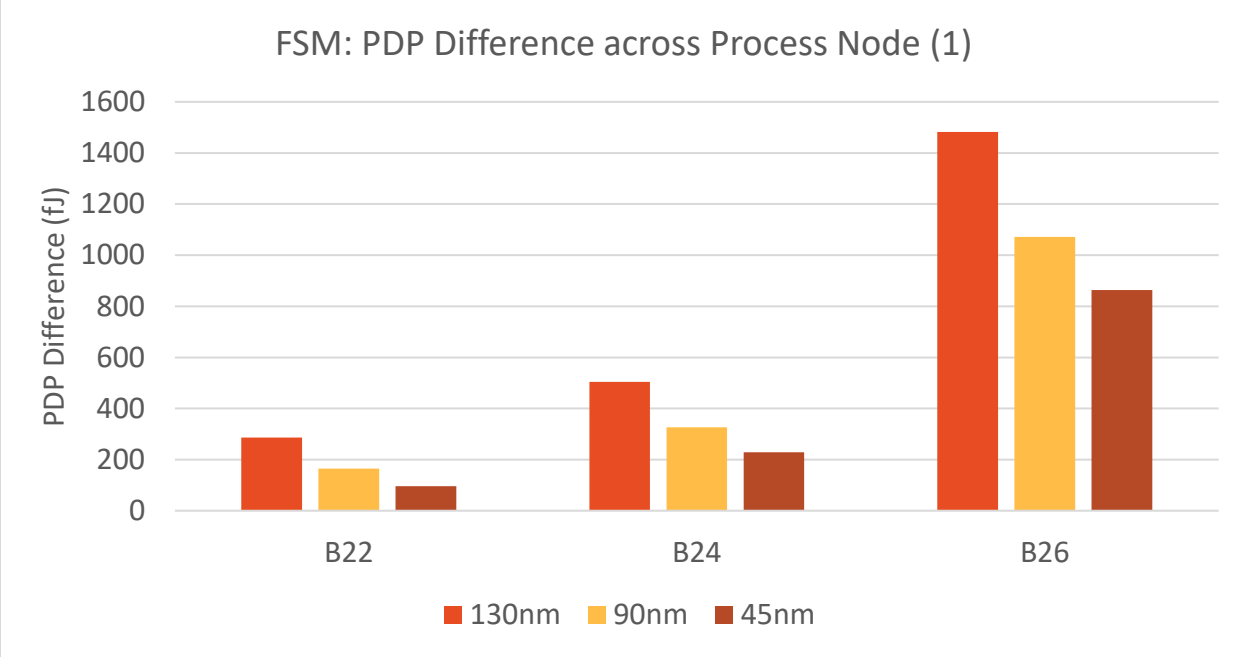

Figure 22: PDP difference for FSM designs with varying state bits

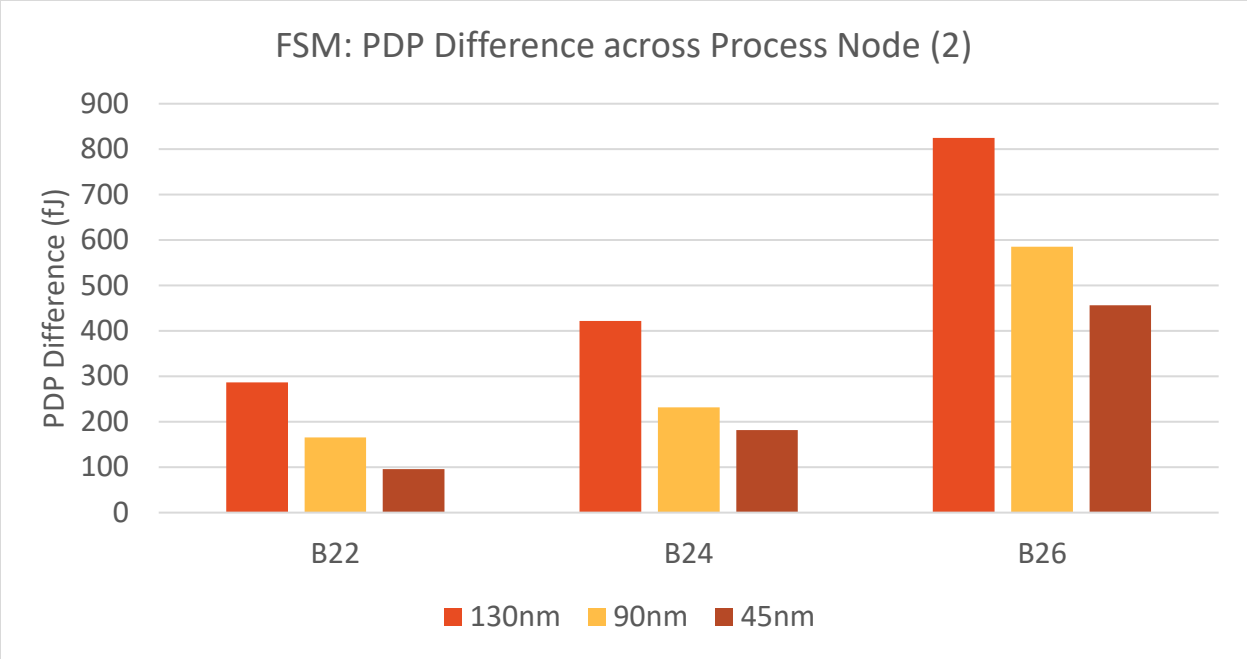

Figure 23: PDP difference for FSM designs with varying input bits

As shown in Figures 22 and 23, the PDP difference for the FSM decreases as the process node decreases for all cases. This is in contrast to the unpipelined FIR filter and the ALU, shown in Figure 24 and Figure 25 respectively, where the PDP differences do not show a clear trend. Since both the ALU and FIR filter have longer data depths, the speed target for the synchronous design could play a role in this difference between designs. One possible clue to this fact is the large number of iterations the 130nm synchronous FIR underwent during synthesis to meet the worst-case negative slack requirements. The synthesis tool performed many more optimization iterations for this design than any other of the designs and as a possible result, the power is much higher for the design. This theory could be tested in the future by synthesizing the 130nm synchronous unpipelined FIR filter at slower speeds to see the effect on the PDP. The fact that the 64-bit ALU also has a negative PDP difference also supports this theory since the ALU also has a large logic depth due to its adder based structure. During synthesis, many optimization iterations were needed for this design in the 130nm process as well. This increase in logic depth

could widen the gap between MTNCL's average-case performance and synchronous' worst-case performance making timing targets for the synchronous design more difficult to meet during synthesis.

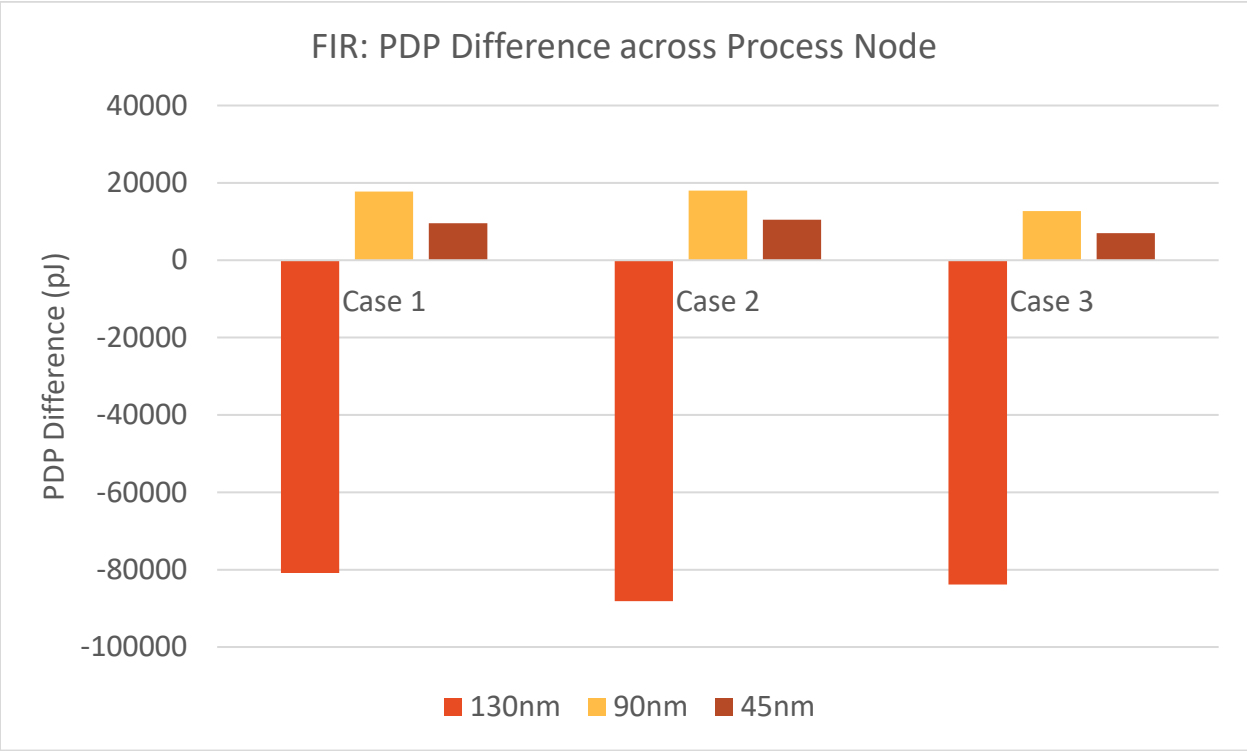

Figure 24: PDP difference for unpipelined FIR design

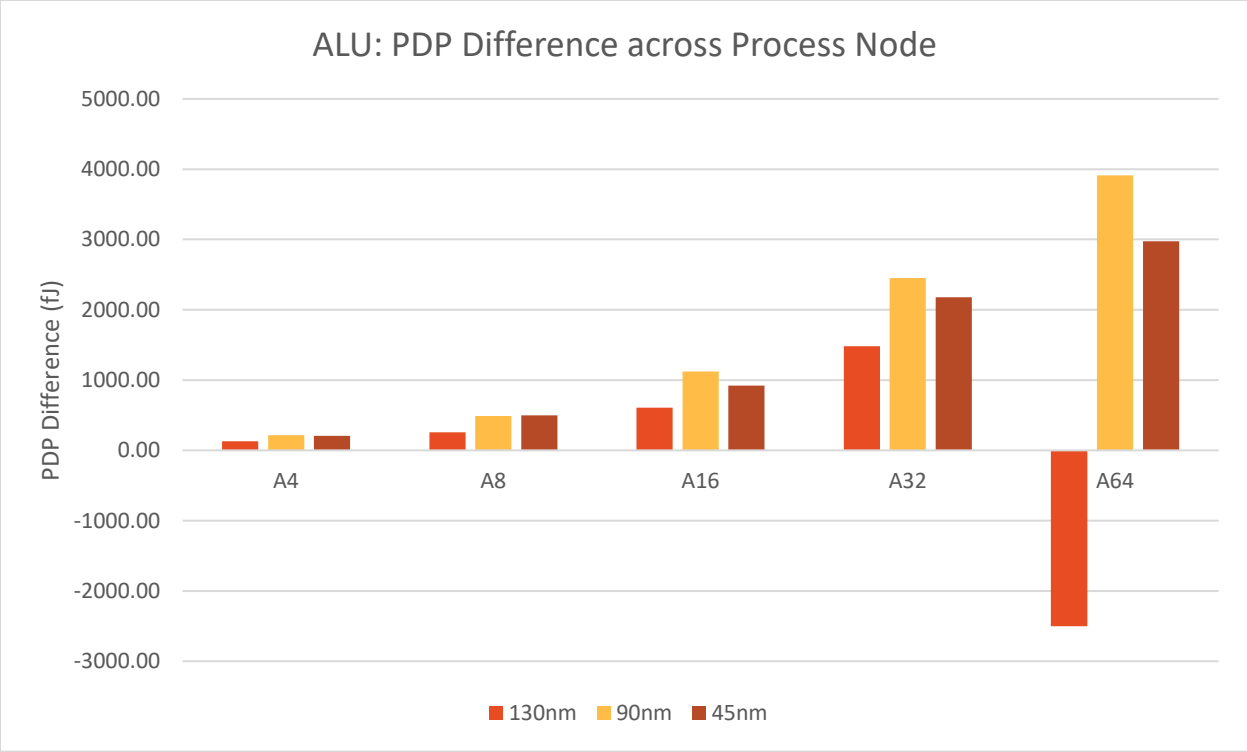

Figure 25: PDP difference for ALU designs

3.5.2 Leakage Power

One of the major issues the industry has faced scaling to lower process nodes is the large increase in leakage power. In order to mitigate this increased power usage, many different transistor thresholds are available at smaller process nodes so that the designer can balance leakage power draw and performance. The choice of threshold voltage in both the synchronous design and the MTNCL design has a huge effect on the leakage power comparison between them. However, MTNCL gate libraries are specifically designed to use less leakage than the synchronous standard cell libraries. As a result, all MTNCL designs use less leakage power than an equivalent synchronous design. As the design is scaled to smaller process nodes, the difference in leakage power between the MTNCL and synchronous designs increased across almost all designs tested.

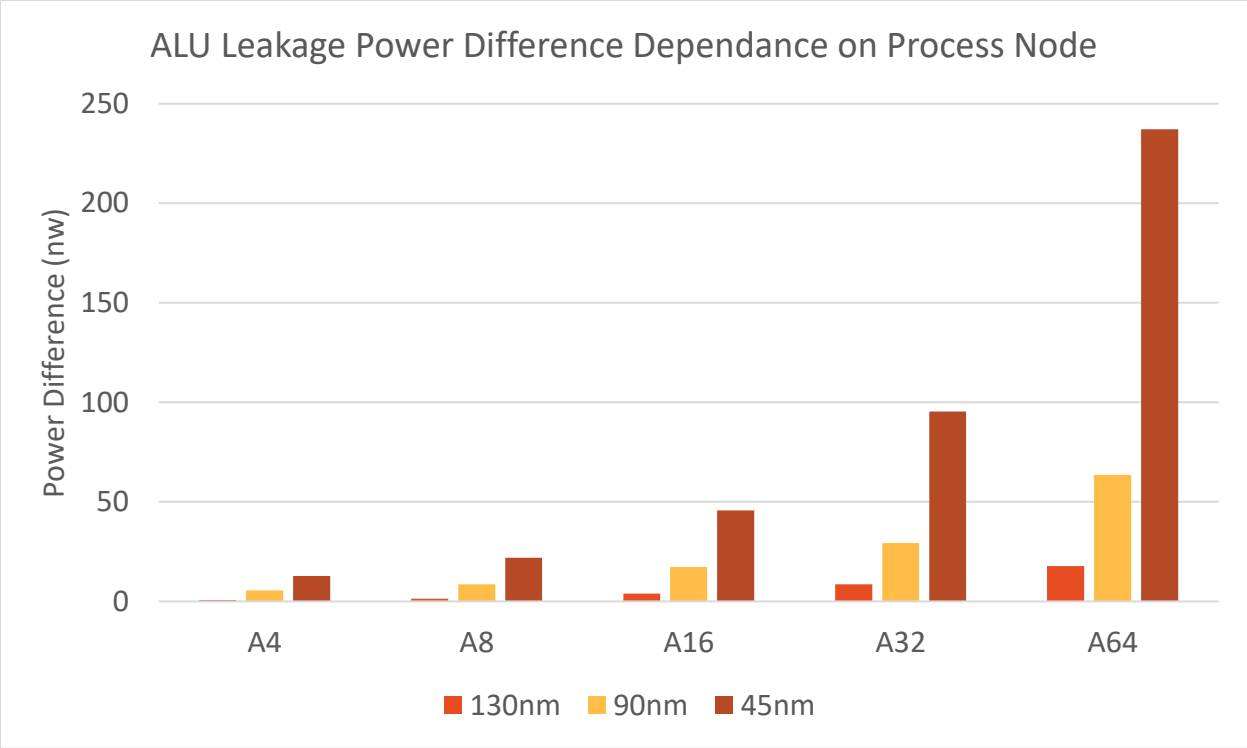

Figure 26: Leakage power difference for ALU designs

As shown in Figure 26, the leakage power difference more than doubled between each design node tested, showing that MTNCL becomes better than equivalent synchronous designs in terms of leakage power at smaller process nodes. The trend is similar with the FSM; however, for some of the smaller designs, the 45nm leakage power difference was smaller than the 90nm power difference. The data for the FSM leakage power can be seen in Figures 27-29. It is broken up into 3 figures to better show the difference in leakage power.

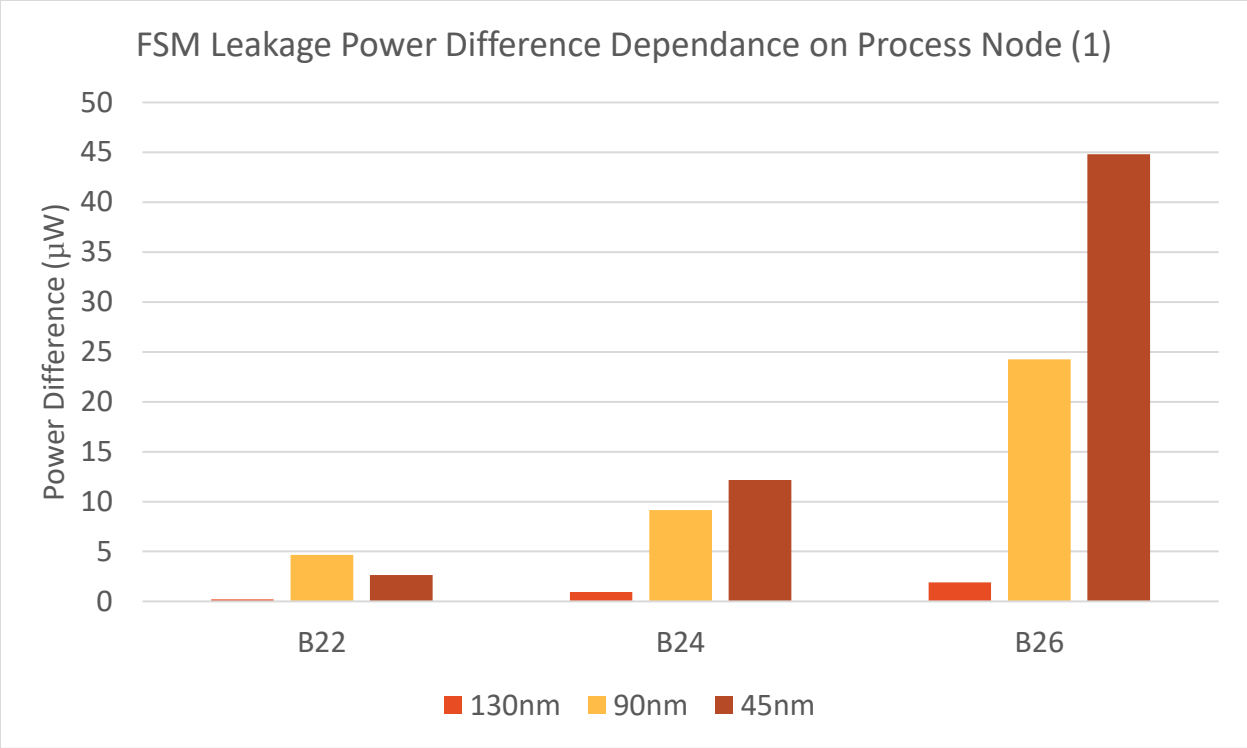

Figure 27: Leakage power difference for FSM designs (1)

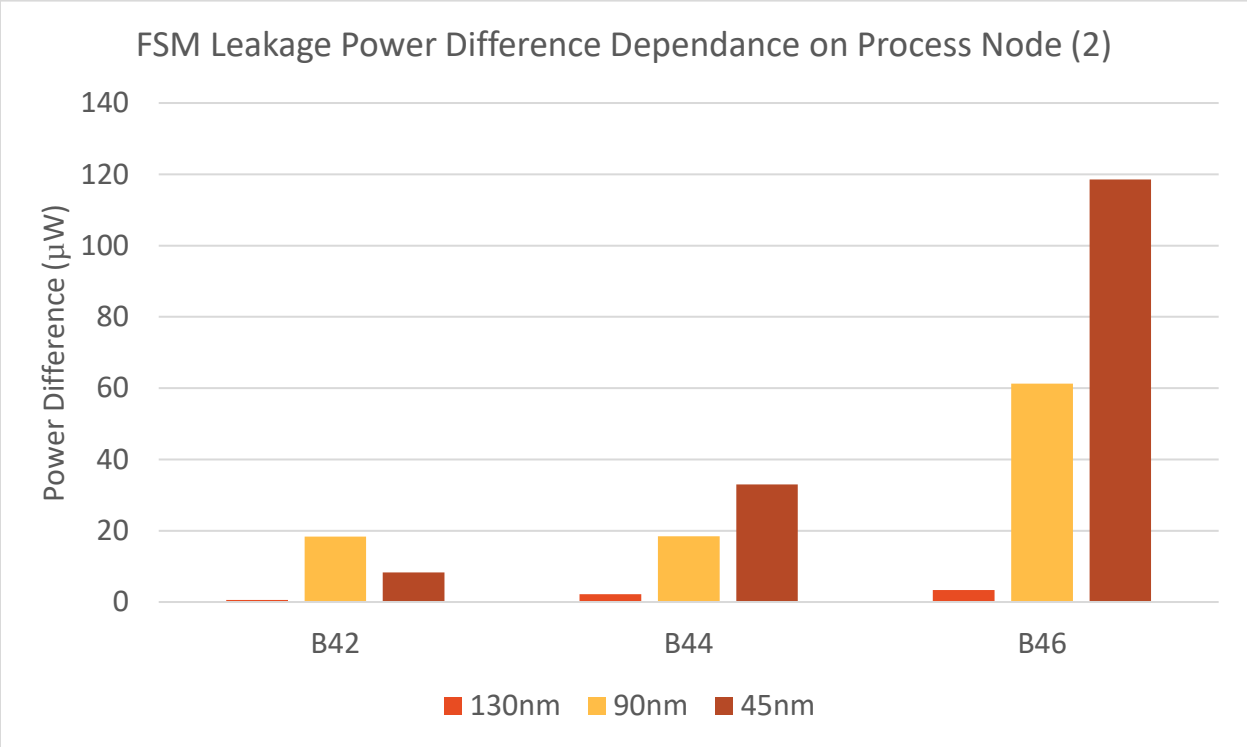

Figure 28: Leakage power difference for FSM designs (2)

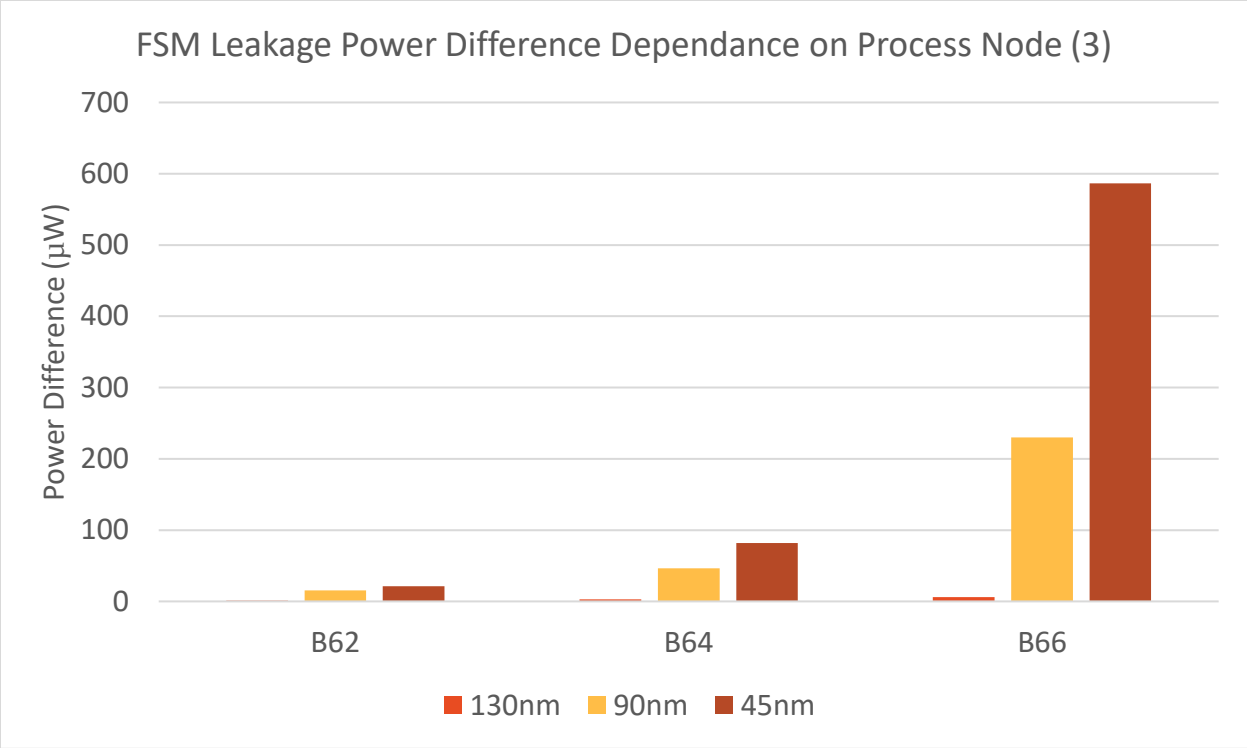

Figure 29: Leakage power difference for FSM designs (3)

While the MTNCL FSMs uses less leakage power than their synchronous counterparts across all design sizes, the 45nm leakage power difference was smaller than that of the 90nm for the 2 smallest designs of the 9 tested. This deviation from the trend seen in the rest of the circuits is most likely due to differences in the standard cell library and synthesized netlist. Since the leakage power difference is so small for these smaller designs, any differences in the netlist could impact the leakage power. In fact, the 90nm synchronous FSM netlist uses 18 gates; whereas, the 45nm synchronous FSM netlist uses only 13 gates. Therefore, it does not appear that the 2 cases in Figure 27 and Figure 28 which deviate from the trend disprove the theory that MTNCL circuits are generally better at smaller process nodes in terms of leakage. Rarely would a circuit be implemented with so few logic gates.

Finally, the leakage power of the unpipelined FIR filter is shown across process nodes in Figure 30.

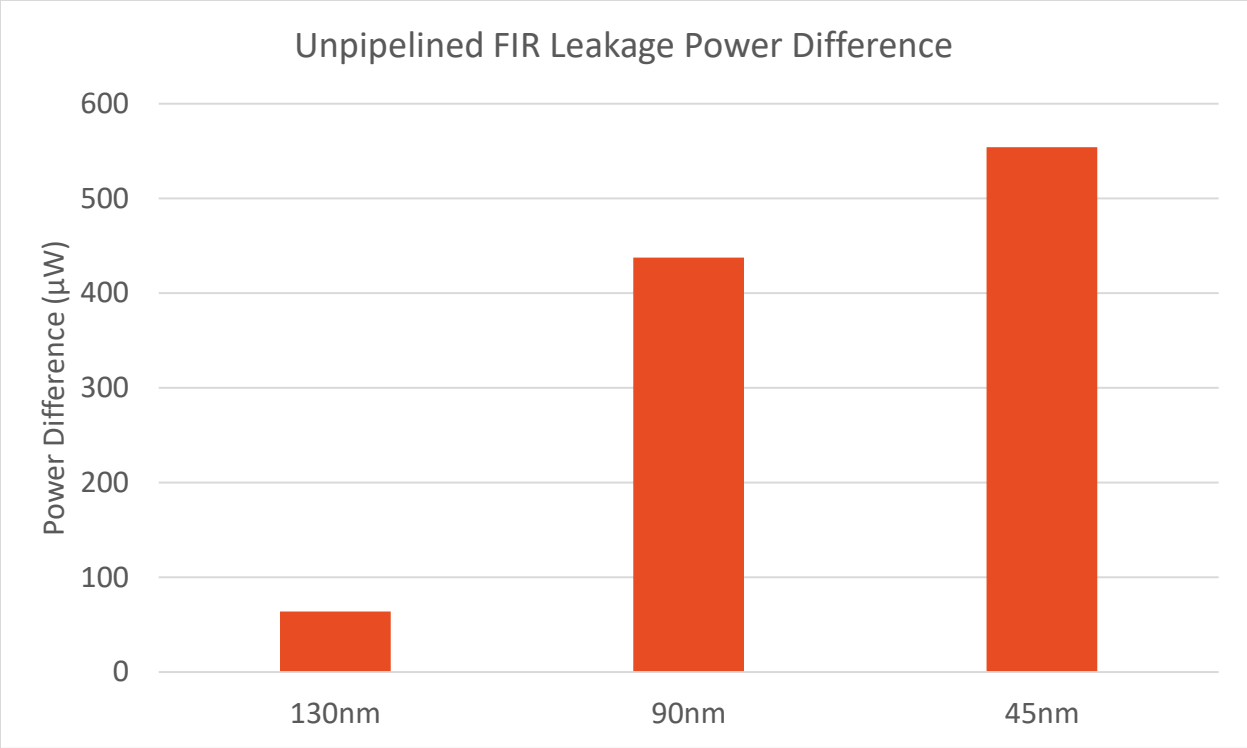

Figure 30: Leakage power difference for unpipelined FIR

The FIR filter follows the same trend in terms of the leakage power difference between the synchronous and MTNCL designs with the leakage power increasing as process size decreases. This trend is consistent across nearly all the designs, but it is important to note that this trend could be different if different threshold transistors are chosen. The threshold voltage choice when designing each of the cell libraries has a major impact on the leakage power tradeoff between them.

3.6 Analyzing Effects of Pipeline Granularity on Power Tradeoff

Because of the high power clock tree required in synchronous designs, it was expected that a design with more registers would be better suited to MTNCL than one with very few registers. In order to test this, two versions of the FIR filter were created, one with many pipeline stages and another with no pipelining.

3.6.1 Dynamic Power

To compare the dynamic power difference between the two FIR designs with respect to pipelining, both designs were simulated using the same 3 test cases described in Section 3.2.3. The results of these simulations are shown in Figure 31 below.

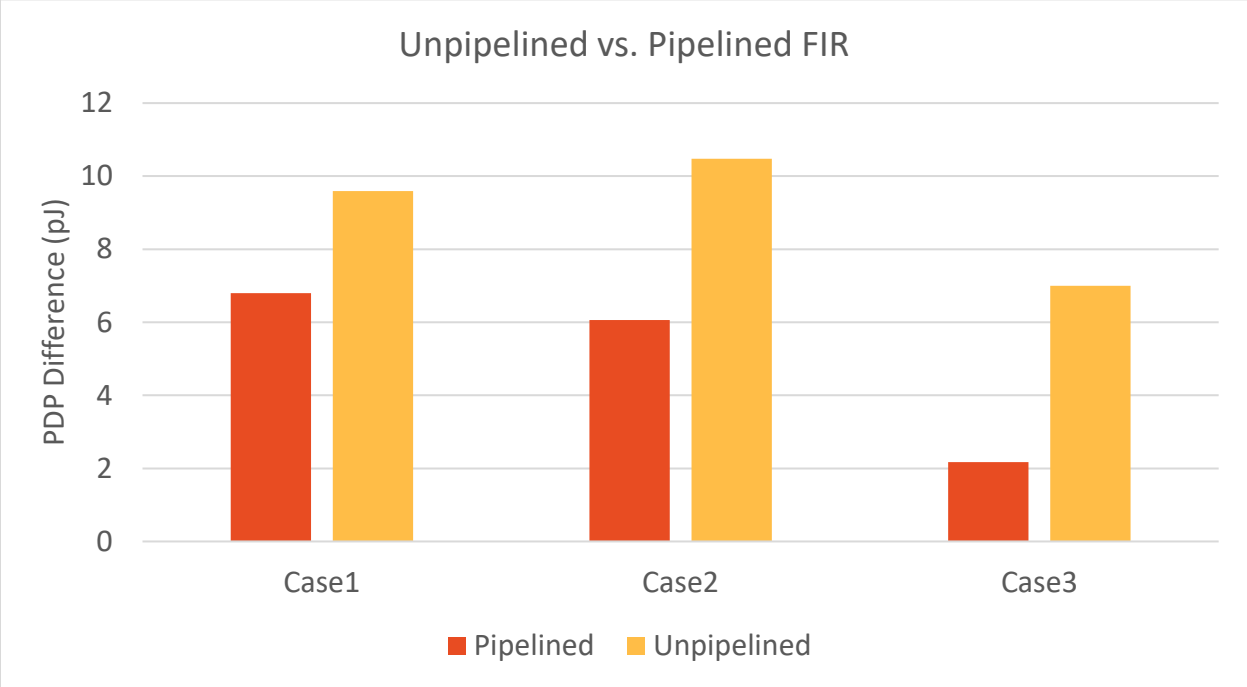

Figure 31: PDP difference for unpipelined and pipelined FIR filter in 45nm process

This data appears to support the hypothesis the power difference between MTNCL and synchronous designs will be lower for pipelined designs than for unpipelined designs. As mentioned previously, increasing the number of registers in the design also increases the complexity of the clock tree in the synchronous design, especially with a larger design like the FIR filter. While this increase in registers also has an impact on the MTNCL power draw, it is not as large of an increase as the synchronous design. Figure 32 shows the PDP for both the pipelined and unpipelined FIR filters for the Case 1 input pattern.

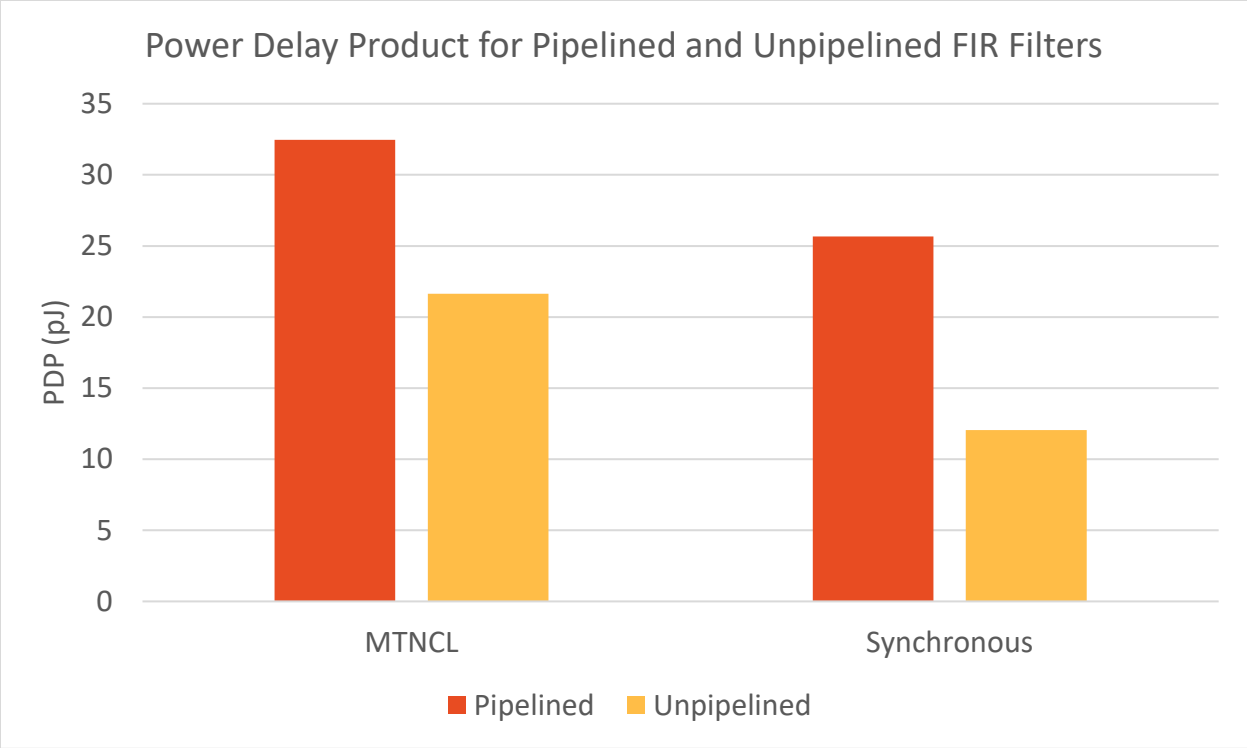

Figure 32: PDP for unpipelined and pipelined FIR filter in 45nm process

3.6.2 Leakage Power

Pipelining the design also had an effect on the leakage power tradeoff between the two designs. This effect is due in part to the change in design size, but the type of gates in the design and the increase in clock tree size also affect the leakage power tradeoff. The leakage power differences for the pipelined and unpipelined designs are shown in Figure 33.

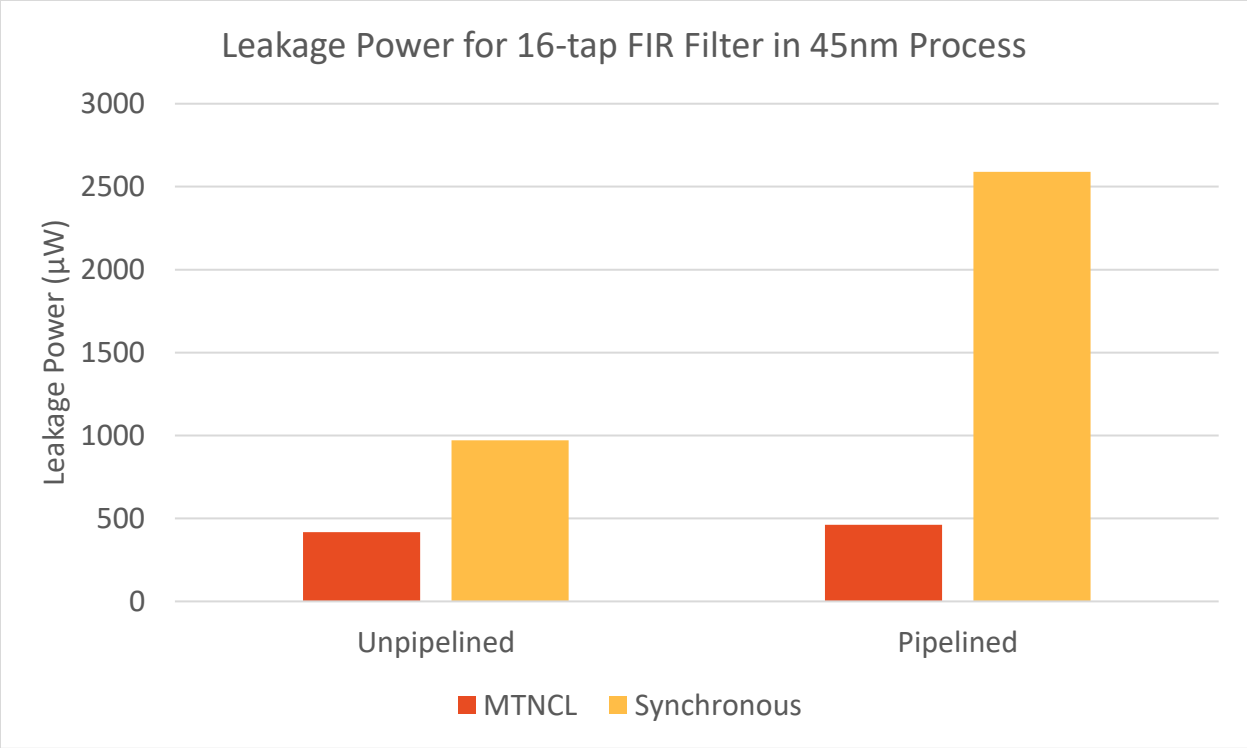

Figure 33: Leakage power difference for unpipelined and pipelined FIR in 45nm process

The pipelined design had a much higher difference in leakage power than the unpipelined design. Overall, MTNCL designs seem better suited to heavily pipelined designs where leakage power is a major concern.

4 Conclusion

In conclusion, MTNCL circuits are best suited for application scenarios where periods of high-speed computation are required followed by long periods of inactivity. This is because MTNCL circuits typically use more dynamic power when compared with equivalent synchronous circuits, especially as the process feature size decreases. In order to achieve these same low leakage benefits in synchronous circuits, additional circuitry must be added to determine when it is safe to sleep the circuit through multi-threshold CMOS power gating which adds additional paths for leakage current and consumes more dynamic power during operation. However, the method of sleeping MTNCL circuits is built directly into the architecture itself.

The sparsity of data waves required to make MTNCL beneficial over equivalent synchronous designs depends on the difference between the dynamic power of both designs as well as the difference between the MTNCL and synchronous leakage power consumption. Both of these power figures depend on process node, input data patterns, design type, circuit size, and pipeline granularity to varying degrees. Larger differences in dynamic power consumption between the equivalent MTNCL and synchronous designs will require either longer periods of inactivity or a larger difference in leakage power between the two designs in order to make MTNCL beneficial in terms of power.

In general, the difference in dynamic power consumption between MTNCL and synchronous increases when design size increases and decreases with an increase in pipeline stages. The effects of scaling process node are dependent on the design type. The MTNCL dynamic power consumption was higher than the equivalent synchronous design for nearly all test cases. The leakage power difference on the other hand increases in favor of MTNCL with increasing design size, decreasing process node, and an increase in pipeline stages. The MTNCL

leakage power was lower than the equivalent synchronous design for nearly all test cases. Figure 34 and Figure 35 show the general trend of the effect on PDP difference and leakage power difference for each of these design dimensions explored. For the leakage power only the design size and process node are shown since differences in design and pipeline stages affect leakage primarily through design size. Moving up along the y-axis on each graph indicates a widening gap between the MTNCL and synchronous power values in favor of the synchronous architecture for the PDP difference and in favor of MTNCL for the leakage power difference.

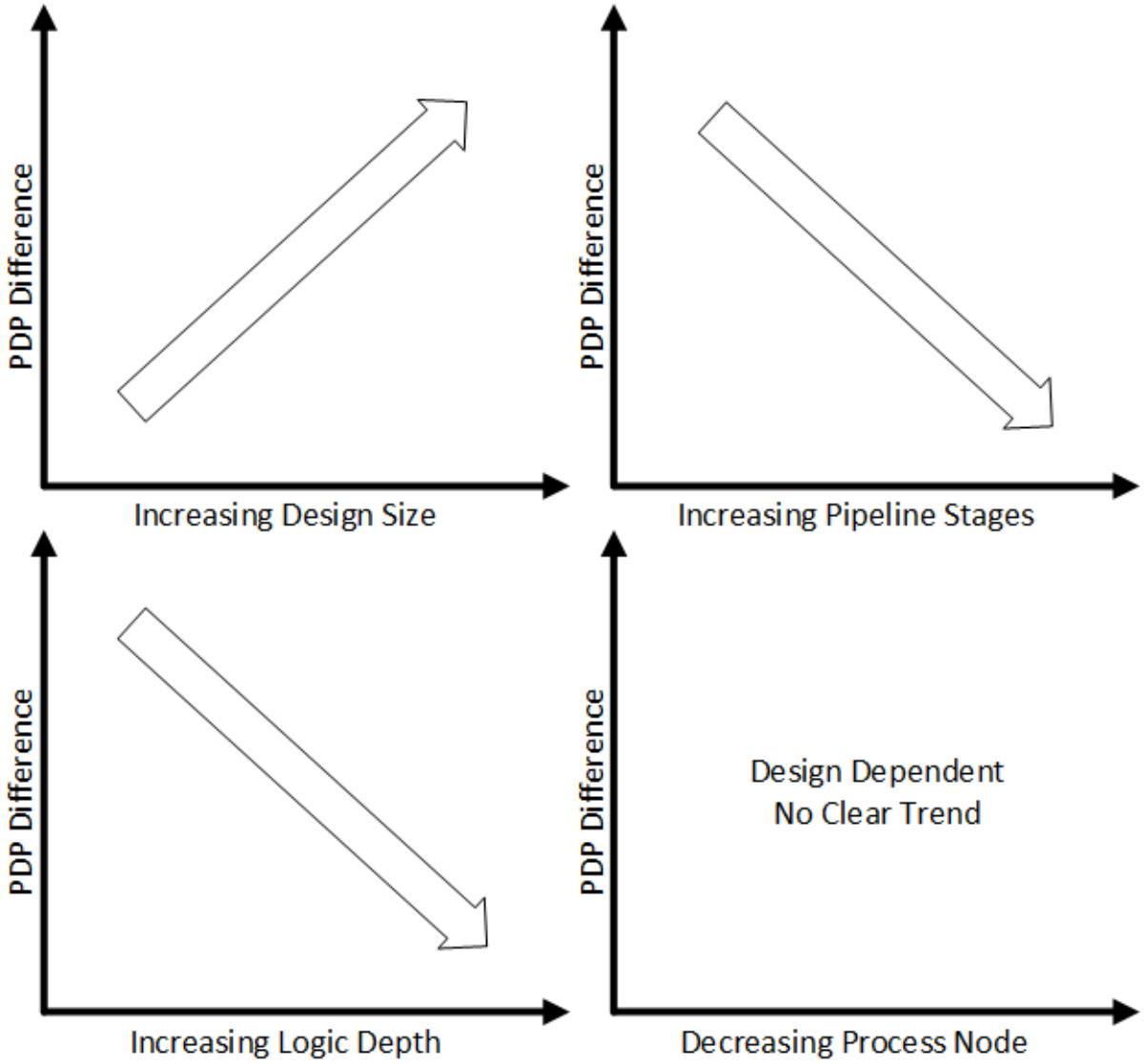

Figure 34: General trend of PDP difference for each dimension explored

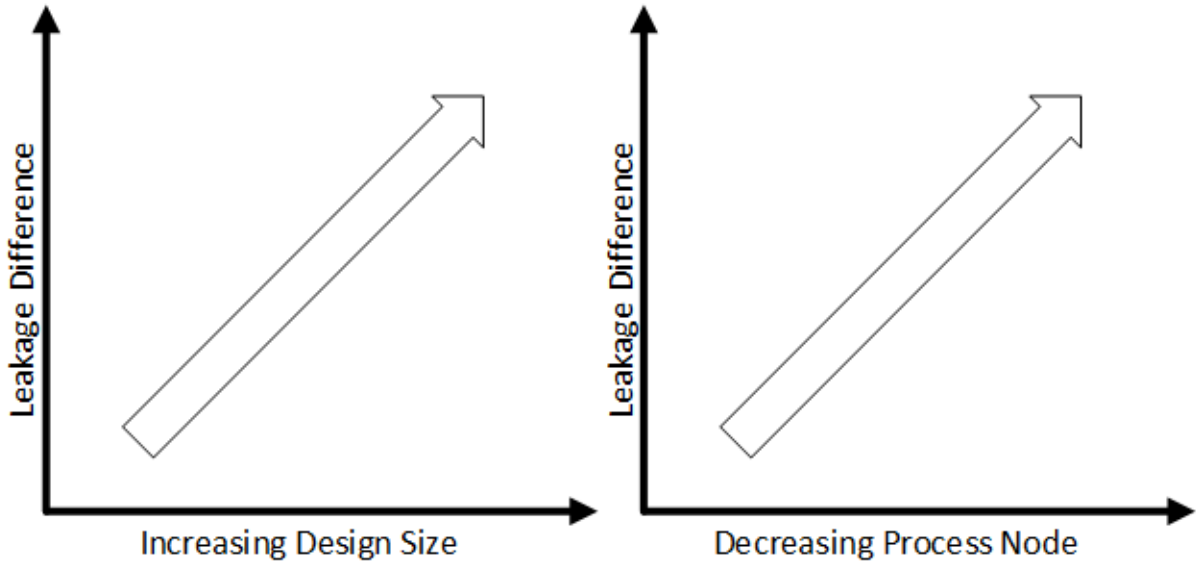

Figure 35: General trend of leakage difference with respect to design size and process node

In the future, it may be beneficial to use SAIF files to estimate power usage of asynchronous circuits earlier in the design flow without the need for time consuming transistor-level simulations. This power analysis method only requires the synthesized netlist, Liberty file, and testbench; however, first the accuracy of these results must be evaluated for asynchronous designs. If this is a viable method, it could drastically reduce the iteration time required when designing an MTNCL circuit for low power applications. Additionally, it would be interesting to see if MTNCL switching activity can be reduced by decreasing the number of internal nodes that must switch during a DATA/NULL cycle through synthesis optimizations as the higher switching activity of MTNCL is one of the driving factors for its higher dynamic power consumption.

5 References

[1] A. D. Bailey, J. Di, S. C. Smith and H. A. Mantooth, "Ultra-low power delay-insensitive circuit design," *51st Midwest Symposium on Circuits and Systems,* pp. 503-506, 2008.

[2] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *Proceedings of the International Conference on Application Specific Systems, Architectures, and Processors,* pp. 261-273, 1996.

[3] S. Smith and J. Di, Designing Asynchronous Circuits using NULL Convention Logic (NCL), Morgan & Claypool, 2009.

[4] M. Hinds, "Design and Analysis of an Asynchronous Microcontroller," ProQuest Dissertations Publishing, 2016.

[5] M. Hinds, B. Sparkman, J. Di and S. C. Smith, "An Asynchronous Advanced Encryption Standard Core Design for Energy Efficiency," *Journal of Low Power Electroncis,* vol. 9, no. 2, pp. 175-188, 2013.

[6] L. Zhou, R. Parameswaran, F. A. Parsan, S. C. Smith and J. Di, "Multi-Threshold NULL Convention Logic (MTNCL): An Ultra-Low Power Asynchronous Circuit Design Methodology," *Journal of Low-Power Electronics,* pp. 81-100, 2015.

[7] L. Zhou, S. C. Smith and J. Di, "Bit-Wise MTNCL: An Ultra-Low Power Bit-Wise Pipelined Asynchronous Circuit Design Methodology," *I5 Midwest Symposium on Circuits and Systems,* pp. 217-220, 2010.

[8] J. Kessels and A. Peeters, "The Tangram Framework: Asynchronous Circuits for Low Power," *Proceedings of the ASP-DAC,* pp. 255-260, 2001.

[9] A. Bardsley and D. A. Edwards, "The Balsa Asynchronous Circuit Synthesis System," 2000. [Online]. Available: http://apt.cs.manchester.ac.uk/ftp/pub/apt/papers/FDL00.pdf. [Accessed 28 06 2018].

[10] R. B. Reese, S. C. Smith and M. A. Thornton, "Uncle - An RTL Approach to Asynchronous Design," *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems,* pp. 65-72, 2012.

[11] B. A. Bell, W. Bouillon, S. Li and E. Logal, "Application of Multi-Threshold NULL Convention Logic to Adaptive Beamforming Circuits for Ultra-Low Power," in *Government Microcircuit Applications & Critical Technology Conference,* Orlando, 2016.