University of Arkansas, Fayetteville

ScholarWorks@UARK

Graduate Theses and Dissertations

12-2022

Design and Comparison of Asynchronous FFT Implementations

Julie Bigot
University of Arkansas, Fayetteville

Follow this and additional works at: https://scholarworks.uark.edu/etd

Part of the Graphics and Human Computer Interfaces Commons, and the Signal Processing Commons

Citation

Bigot, J. (2022). Design and Comparison of Asynchronous FFT Implementations. *Graduate Theses and Dissertations* Retrieved from https://scholarworks.uark.edu/etd/4720

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

Design and Comparison of Asynchronous FFT Implementations

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

by

Julie Bigot
University of Arkansas
Bachelor of Science in Computer Engineering, 2020

December 2022
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Jia Di, Ph.D.
Thesis Director

Dale Thompson, Ph.D.	Brajendra Panda
Committee Member	Committee Member

ABSTRACT

Fast Fourier Transform (FFT) is a widely used digital signal processing technology in a large variety of applications. For battery-powered embedded systems incorporating FFT, its physical implementation is constrained by strict power consumption, especially during idle periods. Compared to the prevailing clocked synchronous counterpart, quasi-delay insensitive asynchronous circuits offer a series of advantages including flexible timing requirement and lower leakage power, making them ideal choices for these systems. In this thesis work, various FFT configurations were implemented in the low-power Multi-Threshold NULL Convention Logic (MTNCL) paradigm. Analysis illustrates the area and power consumption trends along the changing of the number of points, data widths, and the number of pipeline stages.

ACKNOWLEDGEMENTS

First, I would like to give my deepest gratitude to my mentor Dr. Jia Di. His support and guidance have been invaluable to shaping me into the person I am now.

I would also like to thank the lab members I worked with, especially Cole Sherrill, Forrest Tennant, and Chandler Bernard.

I am very grateful to my parents Rita Janssens and Michael Bigot for their love and support through my graduate studies.

I am also very thankful for having spent this journey with my daughter, Catalina Ubaldo. She has been my strongest motivation to continue when things became difficult.

TABLE OF CONTENTS

1. Introduction

Digital signal processing (DSP) technologies are widely utilized in many everyday applications, ranging from thermostats to noise-cancelling headphones. Considering the large amount of DSP algorithms, the Fast Fourier Transform (FFT) algorithm is among the most commonly used. FFT computes the Discrete Fourier Transform (DFT) of a sequence, converting this signal from its original domain (mainly time or space) to a representation in the frequency domain, or its inverse (IDFT) from frequency domain to the original domain. FFT does such transformations rapidly by factorizing the DFT matrix into a product of sparse factors.

While for an electronic system, DSP algorithms including FFT can be implemented using digital signal processors [1], which are powerful and capable of executing multiple algorithms, many embedded systems cannot adopt these processors. Such systems include wearable electronics, edge computing devices, distributed sensors, etc., which are normally battery-powered and are under strict power/cost budget. Therefore, DSP algorithms implemented in Application-Specific Integrated Circuits (ASICs) become a top choice for these systems due to their low power consumption, low unit cost, and small footprint.

As the semiconductor technology advances, the feature size of transistors kept shrinking, which induces high leakage power as it becomes increasingly difficult to turn off transistors in idle mode. For most abovementioned embedded system application, leakage power is critical since their work duty cycles are normally low. FinFET and gate-all-around FET (GAAFET) technologies [2] have been invented to alleviate this problem. However, these leading semiconductor technologies are very expensive, increasing the cost of adoption. Moreover, for smaller transistors, process variation becomes more severe, amplifying all clock-related problems in synchronous circuits such as clock skew, thereby making timing closure much more difficult to achieve.

Asynchronous circuits, especially quasi-delay insensitive (QDI) paradigms, use local handshaking protocols instead of global clocks to coordinate the circuit operation. Little to none timing analysis is needed, making them much more robust to process variations. The Multi-Threshold NULL Convention Logic (MTNCL) paradigm incorporates high threshold voltage transistors inside each logic gate, which substantially reduces leakage power during idle mode. In this thesis work, a series of MTNCL FFT configurations have been implemented in the TSMC 65nm technology. These configurations include 2-, 4-, and 8-point, 4- and 8-bit data widths, and 1-, 2-, and 3-stage pipelines. Analysis illustrates the trend of area as well as active and leakage power consumption along the changing of these design parameters.

The thesis is organized as follows: Chapter 2 describes the background of asynchronous logic, NCL, and MTNCL, as well as FFT algorithm and the previous asynchronous FFT implementations; Chapter 3 introduces the MTNCL FFT designs; Chapter 4 shows the results with analysis; and Chapter 5 concludes the work.

2. Background

2.1 Asynchronous Logic, NULL Convention Logic, and Multi-Threshold NULL Convention Logic

Asynchronous logic circuits do not have clock; instead, they use handshaking protocols to control the circuit behavior. Different from the bounded-delay counterpart in which gate delays are bounded and the circuit will malfunction if any gate delay exceeds the bound, delay-insensitive (DI) or quasi-delay insensitive (QDI) style asynchronous circuits, such as the NULL Convention Logic (NCL) [3], do not assume delay bounds. Individual gate or wire delay has no impact on the correctness of the circuit's output. Since signal propagation is not time-dependent, NCL circuits require very little, if any, timing analysis. A NCL system consists of DI combinational logic sandwiched between DI registers, as shown in Figure 1(top). The Completion Detection block ensures all output bits of the corresponding register are in the same state before generating the combined handshaking signal (*Ko*) to the previous register.

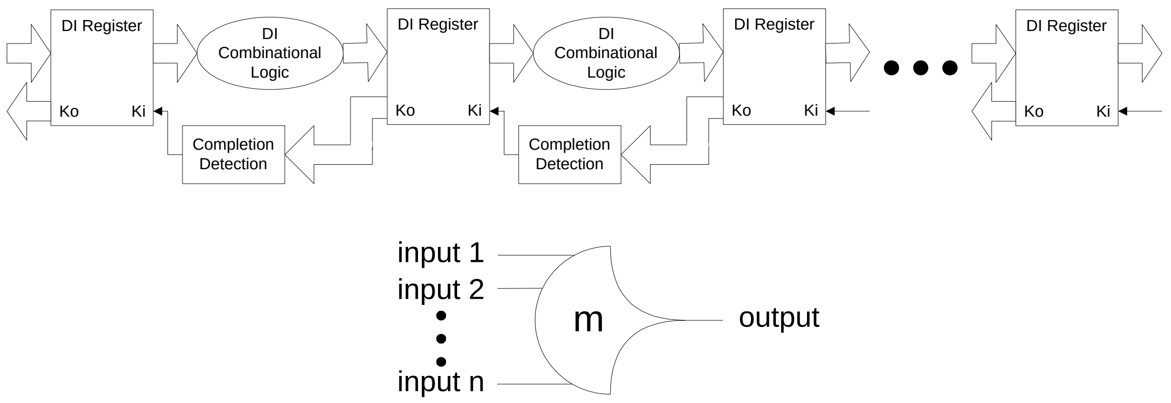

Figure 1: (top) NCL Pipeline (bottom) Thmn Threshold Gate

NCL circuits utilize multi-rail signals to achieve delay-insensitivity. The most prevalent multi-rail encoding scheme is dual-rail. A dual-rail signal, *S*, consists of two wires, S^0 and S^1. The DATA0 state (S^0 = 1, S^1 = 0) corresponds to a Boolean logic 0, the DATA1 state (S^0 = 0, S^1 = 1) corresponds to a Boolean logic 1, and the NULL state (S^0 = 0, S^1 = 0) corresponds to the empty set meaning that data is not yet ready. The invalid state (S^0 = 1, S^1 = 1) never occurs in normal NCL circuit operation; hence, the two rails are typically considered mutually exclusive. NCL logic family consists of 27 threshold gates. The standard threshold gate is the THmn gate, where $1 \leq m \leq n$, as shown in Figure 1(bottom). THmn gates have *n* inputs; at least *m* of the *n* inputs must be asserted before the output will become asserted. Once asserted, all *n* inputs must be deasserted for the output to be deasserted. NCL circuits communicate using request and acknowledge signals to prevent the current DATA from overwriting the previous DATA, by ensuring that the two DATA are always separated by a NULL state.

Originally proposed in 1950's and born with a series of advantages (e.g., no clock tree, flexible timing requirement, high modularity and scalability, high energy efficiency, robust circuit operation, low noise/emission), asynchronous logic, however, has not been developing nearly as fast as the synchronous counterpart, which dominates the digital IC market. This is due to the drawbacks of asynchronous logic, e.g., large area overhead, incompatible with commercial IC design CAD tools. For example, NCL requires the designed circuits to satisfy input-completeness (i.e., output transitions from DATA to NULL only after all inputs transition from DATA to NULL, and vice versa) and observability (i.e., all gate transitions must be observable at the output), which cannot be easily implemented in commercial synthesis tools.

In order to solve these problems, the Multi-Threshold NCL (MTNCL) paradigm was invented [4-13], which combines MTCMOS (Multi-Threshold CMOS) power-gating method with NCL. MTCMOS power-gating involves using high threshold voltage (high-V_t) transistors to gate the power supplies of a low threshold voltage (low-V_t) logic block. When the high-V_t transistors are turned on, the low-V_t logic is connected to virtual ground and power, and switching is performed through fast (low-V_t) devices. When the circuit enters the sleep mode, the high-V_t gating transistors are turned off, resulting in a very low subthreshold leakage current from V_{DD} to ground.

In MTNCL paradigm, MTCMOS power-gating structure is incorporated in each threshold gate. Due to the fact that floating nodes may result in substantial short circuit power consumption at the following stage, the output node of each gate is pulled down to ground during sleep mode. When all MTNCL gates in a pipeline stage are in sleep mode, such that all gates output logic0, this condition is equivalent to the pipeline stage being in the NULL state. Hence, after each DATA cycle, all MTNCL gates in a pipeline stage can be forced to output logic0 by asserting the sleep control signal instead of propagating a NULL wavefront through the stage, thereby speeds up the circuit. Since the handshaking signal indicates whether the corresponding pipeline stage is ready to undergo a DATA or NULL cycle, this signal can be naturally used as the sleep control signal for the corresponding combinational block, without requiring any additional hardware, in contrast to the complex *Sleep* signal generation circuitry needed for synchronous MTCMOS circuits. Early Completion mechanism [4] is used in MTNCL architecture, as shown in Figure 2, where each completion signal is used as *Sleep* signal for all threshold gates in the subsequent pipeline stage. The combinational logic will not be put to sleep until all inputs are NULL and the next stage is requesting for NULL; therefore the NULL wavefront is ready to propagate through the stage, so that this stage can instead be put to sleep without compromising delay-insensitivity. The stage will then remain in sleep mode until all inputs are DATA and the next stage is requesting for DATA, and is therefore ready to evaluate. Through this sleeping mechanism, MTNCL eliminates the requirements for input-completeness and observability [4], thereby significantly simplifies MTNCL circuit design, makes it compliant with standard IC design flow, and reduces area overhead.

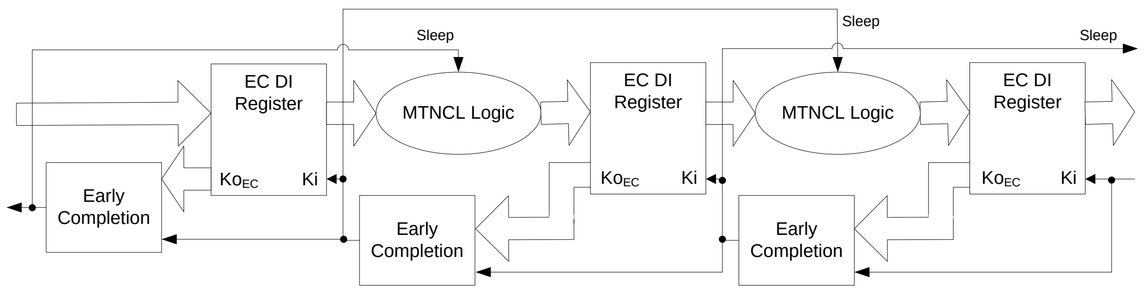

Figure 2: MTNCL Pipeline Architecture using Early Completion

As stated earlier, MTCMOS structure is incorporated inside each NCL threshold gate, and actually results in a number of the original transistors no longer being needed, which also contributes to the area overhead reduction. The original NCL threshold gate structure is shown in Figure 3(a). The *reset*/*set* circuitry is to force the output to be logic 0/1, and the hold0/1 circuitry is to keep the output as logic0/1 for hysteresis. For MTNCL, however, the *reset* circuitry is no longer needed, since the gate output will now be forced to logic0 by the MTCMOS sleep mechanism. Since all gates in a pipeline stage are forced to sleep by the same sleep control signal, NCL gate hysteresis is no longer required. Hence, the *hold1* circuitry and corresponding NMOS transistor are removed, and the PMOS transistor is removed to maintain the complementary nature of CMOS logic. After incorporating the MTCMOS power-gating mechanism, the general gate structure is shown in Figure 3(b). During active mode, the *Sleep* signal is logic0, such that the gate functions as normal. During sleep mode, *Sleep* is logic1, such that the output low-V_t pull-down transistor is turned on quickly to pull the output to logic0, while the high-V_t PMOS gating transistor in the output inverter and all high-V_t NMOS transistors (since all inputs are logic0) are turned off to reduce leakage. As an example, this MTNCL implementation of a TH23 gate is shown in Figure 3(c), which has four less transistors compared to the regular TH23.

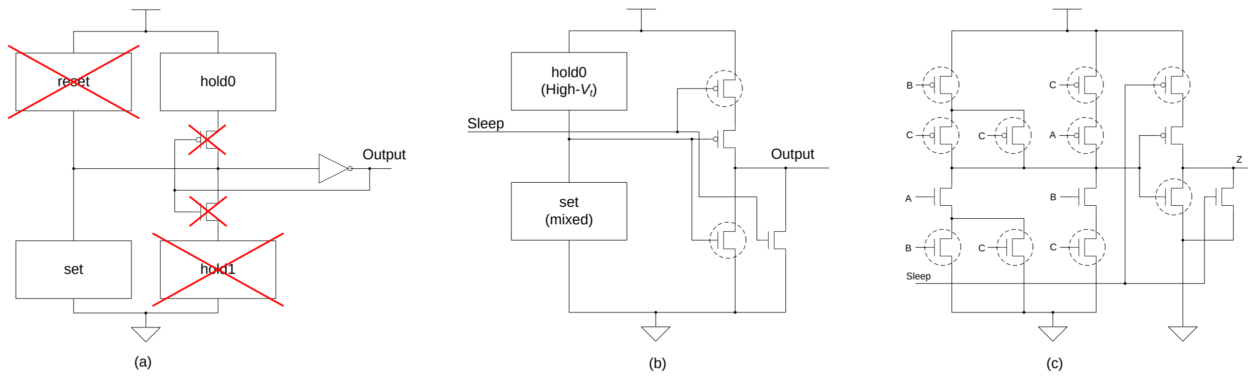

Figure 3: (a) Incorporating MTCMOS Power-Gating into NCL Threshold Gates, (b) MTNCL Gate Structure, (c) TH23 Implementation (Circled Transistors are High-V_t)

While retaining the features of NCL, MTNCL offers several unique and significant advantages over the MTCMOS synchronous and regular NCL counterparts, including leakage reduction in both active and idle modes, reduced area overhead and active energy, improved performance, and enhanced compliance with standard IC design flow. These advantages make MTNCL an ideal choice for implementing the FFT algorithm on power-constrained devices.

2.2 Fast Fourier Transform

The Discrete Fourier Transform (DFT) transforms N coefficients x[n] to N coefficients y[k] using the following definition:

$$y[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

The FFT is an efficient way of calculating the DFT to reduce the number of computations from $O(N^2)$ to $O(N\log N)$. The goal is to build a big DFT from smaller DFT's. The DFT can be divided into a sum of terms of even and odd indices for *n*. The indices need to be adjusted as follows:

$$\begin{cases} even: n = 2r \\ odd: n = 2r + 1 \end{cases}$$

where *r* is in [0, N/2-1]. As a result, the DFT can be written as:

$$y[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_N^{k2r} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r](W_N^2)^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1](W_N^2)^{kr}$$

From $W_N^2 = e^{\frac{-j2\pi 2}{N}} = e^{\frac{-j2\pi}{N/2}} = W_{N/2}$, it follows that:

$$y[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_{N/2}^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_{N/2}^{kr}$$

$$= y_{even[k]} + W_N^k y_{odd}[k]$$

where y_{even} is the N/2-point DFT of even samples and y_{odd} of the odd samples. So, the DFT can be written as a sum of two *N*/2-point DFT's. The splitting continues until $\frac{N}{2^m} = 1$, where $m = \log_2(N)$. This leads to the fundamental computation unit of the FFT, known as the butterfly unit. In general, this unit computes the following:

$$\begin{cases} y_i[s] = y_{i-1}[s] + W_N^l y_{i-1}[t] \\ y_i[t] = y_{i-1}[s] - W_N^l y_{i-1}[t] \end{cases}$$

where i is the stage number.

2.3 Prior Asynchronous FFT Implementations

In [14], 16-point and 64-point FFT circuits were implemented in both synchronous and asynchronous logic. Bundled data (BD) paradigm was adopted for the asynchronous design with a relative-timing based design flow. Due to the nature of BD, timing constraints and analysis were needed. All circuits were implemented using the IBM 65nm process. Simulation results showed the advantages of the asynchronous implementations in performance and energy consumption, with overheads in power and area.

A comprehensive FFT/IFFT processor was introduced in [15]. With SRAM and ROM, this processor was capable of performing 128-point FFT/IFFT computations on 16-bit data input sequence. A 4-phase QDI-like asynchronous handshaking protocol was incorporated. A synchronous baseline was also developed using the same 0.35μm CMOS process. Both circuits were taped out and the chip testing results showed that the asynchronous implementation consumed much lower energy across multiple supply voltages, with overheads in area and performance.

Another asynchronous 128-point FFT design was presented in [16]. QDI handshaking was adopted on 16-bit radix-8 data sequence. It also used SRAMs for storing constants. The circuit was implemented using a 65nm semiconductor node and compared with a synchronous counterpart, exhibiting significant advantages in energy consumption.

In [17], a GALS (Globally Asynchronous Locally Synchronous) architecture-based FFT processor was presented. It was implemented using the BD asynchronous paradigm. The circuit was synthesized onto an Altera FPGA. No comparison was provided with any prior work.

A NCL implementation of 64-point FFT was presented in [18]. It used a serial feedback architecture to minimize the logic area. Two single-rail sequencers were incorporated to control the feedback loop and output the data at the right time. Input completeness and observability were satisfied during the design process.

3. Design Methodology

3.1 Basic Components

The MTNCL FFT architecture consists of multiple basic dual-rail circuit components, i.e., AND function, OR function, XOR function, multiplexer (MUX), full adder, signed multiplier, and complex multiplier.

3.1.1 Basic Logic Functions

Dual-rail logic functions (e.g., AND) are fundamentally different from the Boolean logic counterparts in that their inputs and output are all dual-rail encoded. Therefore, two MTNCL threshold gates are needed to implement each function, except for the full adder which is more complex, as listed below.

AND Function

$$z^1 = a^1 b^1 = TH22(a^1, b^1)$$

$$z^0 = a^0 + b^0 = TH12(a^0, b^0)$$

OR Function

$$z^1 = a^1 + b^1 = TH12(a^1, b^1)$$

$$z^0 = a^0 b^0 = TH22(a^0, b^0)$$

XOR Function

$$z^1 = a^0 b^1 + a^1 b^0 = THXOR(a^0, b^1, a^1, b^0)$$

$$z^0 = a^0 b^0 + a^1 b^1 = THXOR(a^0, b^0, a^1, b^1)$$

MUX

$$z^1 = s^0 b^1 + s^1 a^1 = THXOR(s^0, b^1, s^1, a^1)$$

$$z^0 = s^0 b^0 + s^1 a^0 = THXOR(s^0, b^0, s^1, a^0)$$

Full Adder

$$cout^1 = cin^1a^1 + cin^1b^1 + a^1b^1$$

$$cout^0 = cin^0a^0 + cin^0b^0 + a^0b^0$$

$$s^1 = cout^0cin^1 + cout^0a^1 + cout^0b^1 + cin^1a^1b^1$$

$$s^0 = cout^1cin^0 + cout^1a^0 + cout^1b^0 + cin^0a^0b^0$$

3.1.2 Multipliers

There are two types of multipliers utilized in the MTNCL FFT architecture: one is a signed multiplier and the other is a complex number multiplier. Both multiplier architectures were designed in a generic manner to accommodate different data widths. A 4-bit signed multiplier structure is shown in Figure 4. It is a Baugh-Wooley multiplier with the OR of the two sign bits (i.e., the signal *t* in Figure 4) controlling the switch between signed and unsigned multiplications.

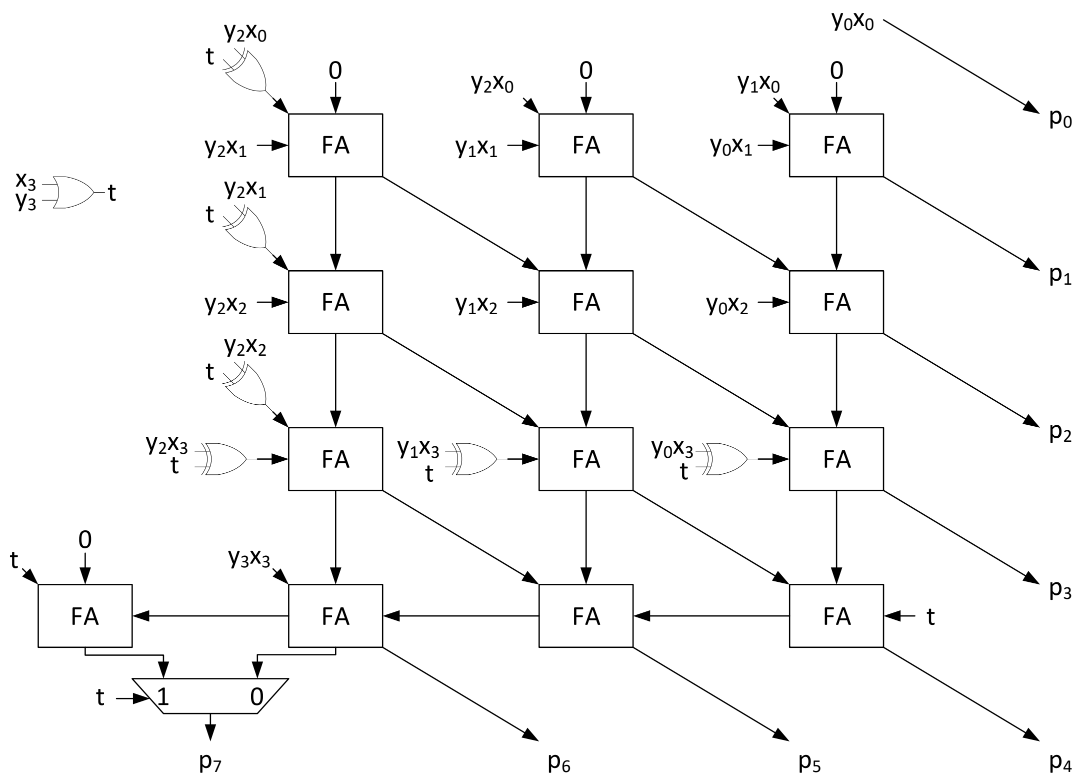

Figure 4: A 4-bit Signed Multiplier Structure

Figure 5 shows the structure of a generic N-bit complex number multiplier. The mathematical equation for multiplying two complex numbers x and w is:

$$(x_r + ix_i)(w_r + iw_i) = i(x_i w_r + x_r w_i) + x_r w_r - x_i w_i$$

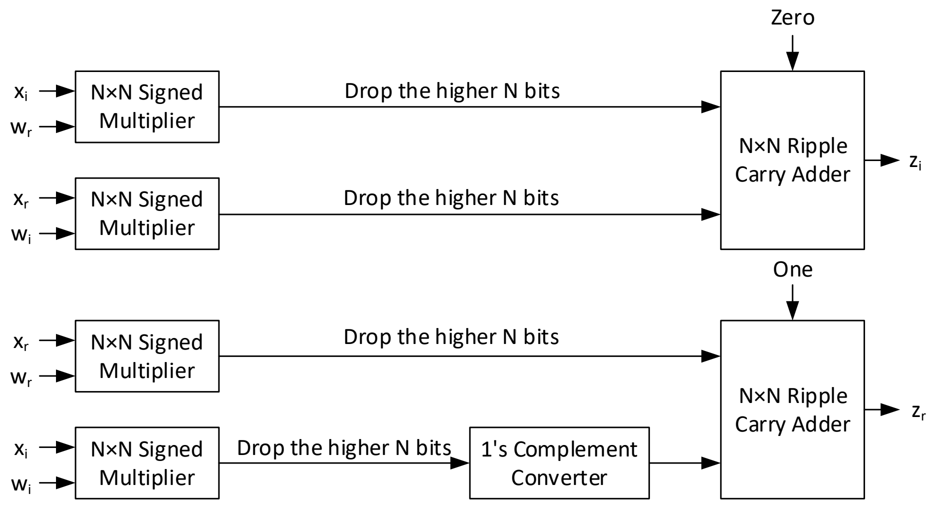

Figure 5: A Generic Complex Multiplier Structure

As shown in Figure 5, the two N-bit input data go through four separate signed multipliers. The products are truncated in half and only the second half (N bits) is passed to subsequent ripple carry adders, except the bottom product is converted to 2’s complement (i.e., the 1’s complement converter and the “one” carry-in) to implement subtraction.

3.1.3 Butterfly Units

As the top-level building block of FFT circuits, a butterfly unit consists of one complex number multiplier, two adders, and two subtractors implemented with adders and 1’s complement converters. A 4-bit butterfly unit structure is shown in Figure 6 below calculating:

$$(z_r + iz_i) = (b_r + ib_i)(w_r + iw_i)$$

$$(x_r + ix_i) = (a_r + z_r) + i(a_i + z_i)$$

$$(y_r + iy_i) = (a_r - z_r) + i(a_i - z_i)$$

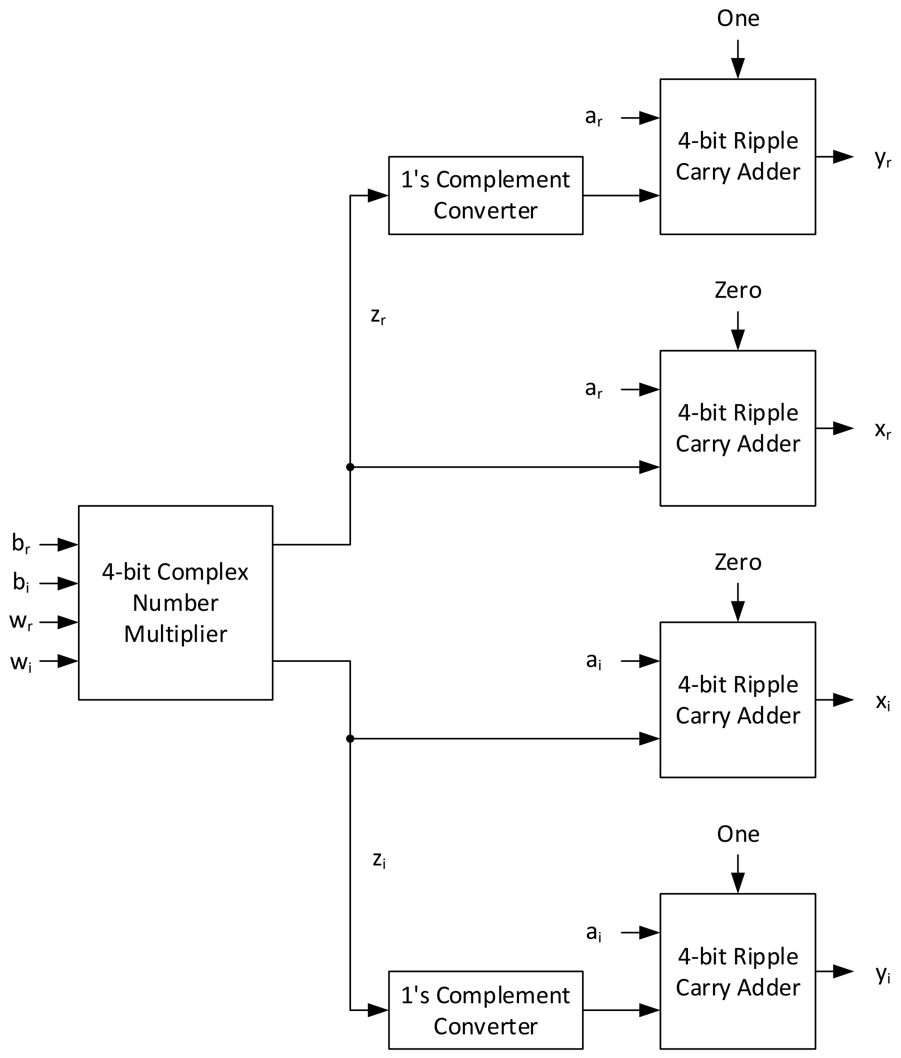

Figure 6: A 4-bit Butterfly Unit Structure

3.2 FFT Configurations

In order to comprehensively evaluate the area and power trend across different FFT parameters, a series of FFT configurations were included with three dimensions: number of points, data widths, and number of pipeline stages.

For number of points, 2-point (Figure 7), 4-point (Figure 8), and 8-point (Figure 9) FFT configurations were included to study the trend of circuit performance.

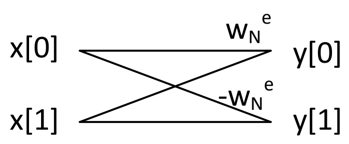

Figure 7: 2-point FFT

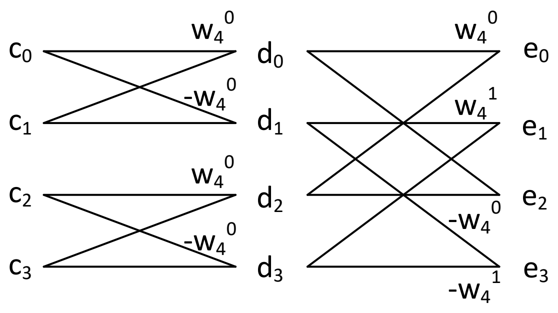

Figure 8: 4-point FFT

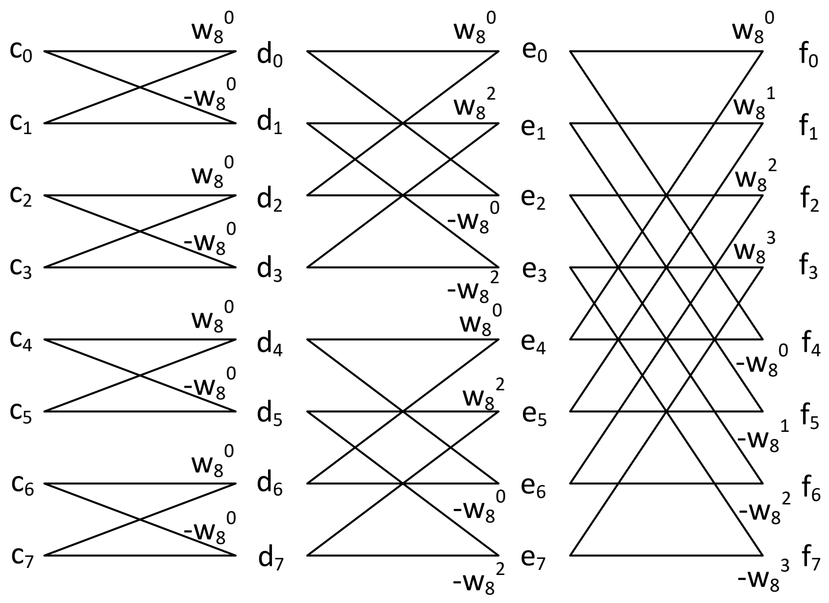

Figure 9: 8-point FFT

For data widths, 4-bit and 8-bit data inputs and weights were included in each FFT configuration. For the number of pipeline stages, all FFT configurations were either not pipelined (or 1-stage pipelined) or pipelined with a butterfly unit as the unit stage. Therefore, it is 1-stage pipeline for 2-point FFT, 2-stage or 1-stage pipeline for 4-point FFT, and 3-stage or 1-stage pipeline for 8-point FFT.

Each FFT configuration was implemented in MTNCL using the circuit components described in Section 3.1. Figures 10 and 11 show the diagrams of 2-stage 4-point FFT and 3-stage 8-point FFT circuits, respectively. The semiconductor technology used was TSMC 65nm bulk CMOS process.

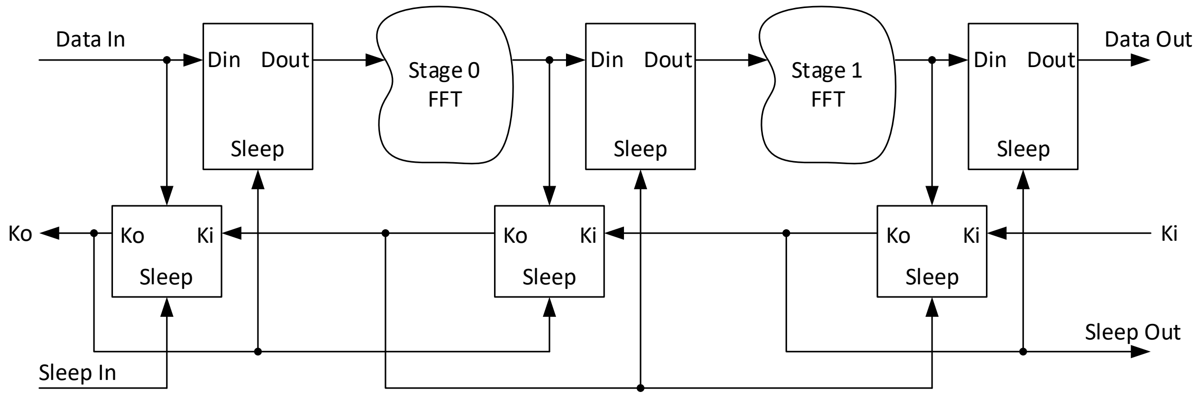

Figure 10: 2-stage 4-point MTNCL FFT Circuit Diagram

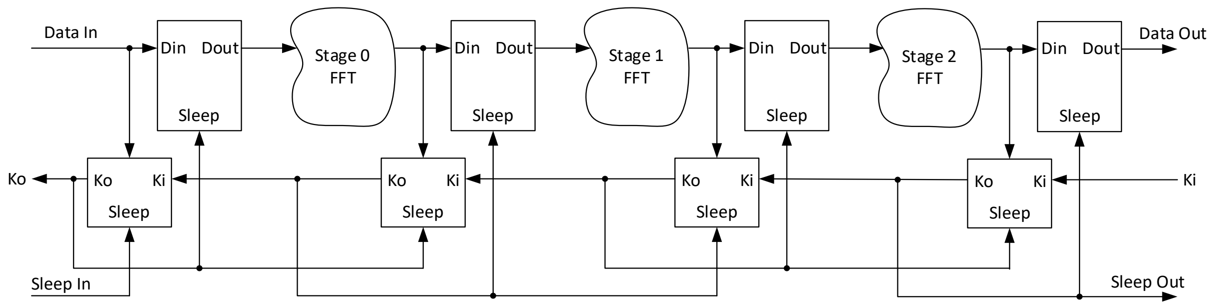

Figure 11: 3-stage 8-point MTNCL FFT Circuit Diagram

4. Results and Analysis

4.1 Data Collection Method

As a comparative analysis, all 10 MTNCL FFT implementations were analyzed for gate count, active power, and leakage power. While gate counts come directly from the flattened netlists, the other two power numbers were calculated from the Synopsys Liberty file. All MTNCL threshold gates, once constructed at transistor level, were characterized using the Synopsys SiliconSmart tool, which automatically simulates each gate for all possible input combinations while driving different load capacitances. Active (short-circuit) and leakage power are among the characterization data recorded in the Liberty file. For each FFT implementation, the power number for each type of MTNCL gate is multiplied by the number of this gate in the design; and then all power numbers are added together. Although this estimate is not as accurate as the traditional power analysis or analog simulations, it is sufficient for this comparative analysis.

4.2 Results and Analysis

Table 1 below summarizes active power, leakage power, and gate counts for all MTNCL FFT circuits. Note that 1-stage pipeline is the same as non-pipeline.

Table 1: Data for MTNCL FFT Circuits

Data Width	# of Point	# of Pipeline Stage	Active Power (mW)	Leakage Power (µW)	Gate Count
4-bit	2	1	0.214	9.222	611
	4	1	0.811	34.048	2254
		2	0.866	36.481	2434
	8	1	2.352	96.815	6390
		3	2.532	109.101	7170
8-bit	2	1	0.711	30.451	1969
	4	1	2.753	116.112	7498
		2	2.833	121.582	7846
	8	1	8.136	340.158	21966
		3	8.456	362.074	23350

4.2.1 Area (Gate Count) Analysis

Using the data in Table 1, Figures 12 and 13 show the gate count trends for non-pipelined and pipelined MTNCL FFT implementations, respectively.

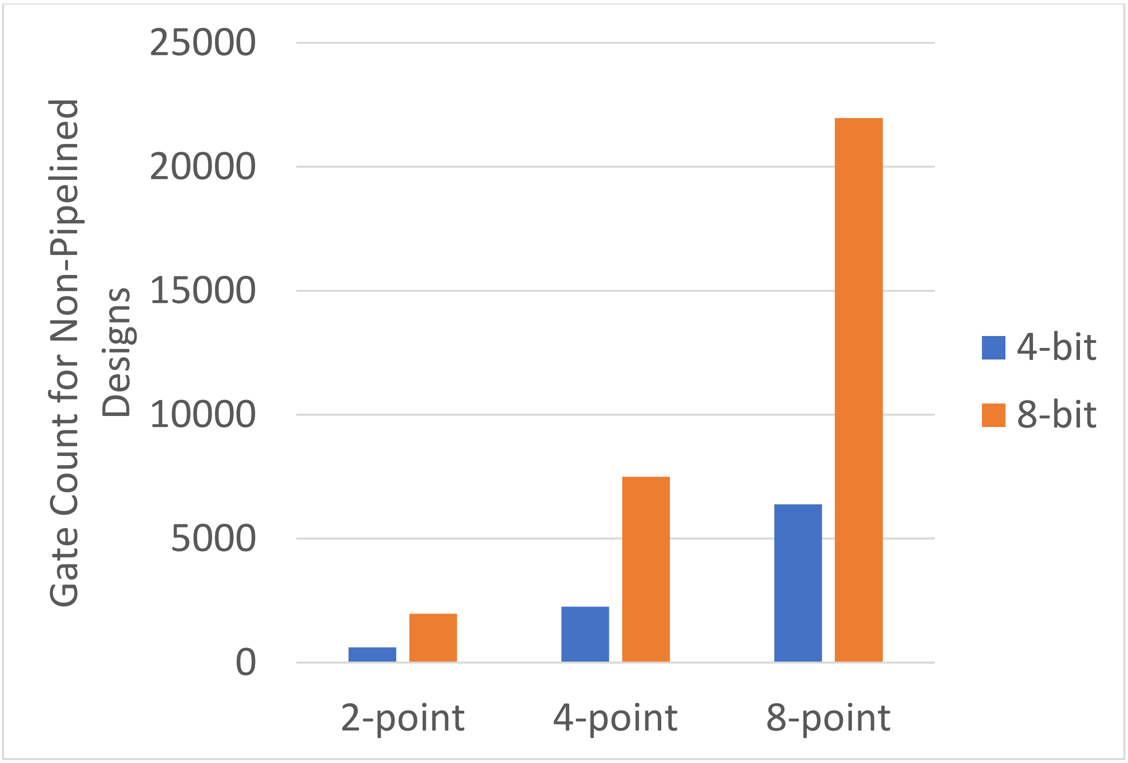

Figure 12: Gate Count Trend for Non-Pipelined MTNCL FFT Implementations

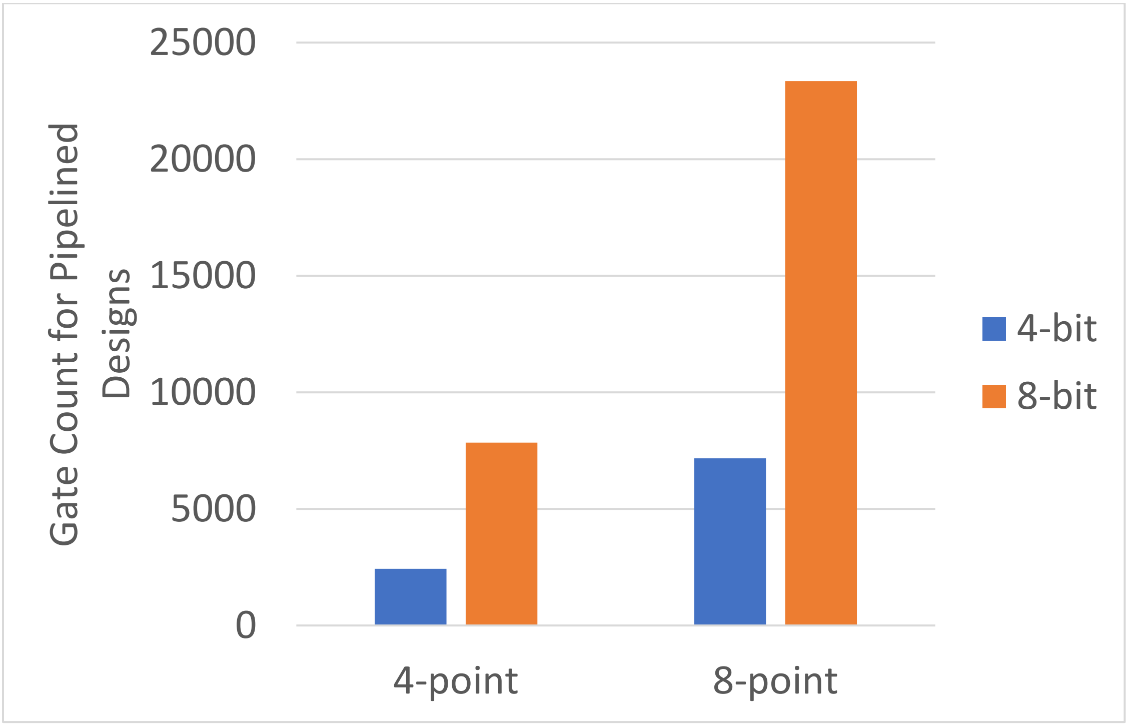

Figure 13: Gate Count Trend for Pipelined MTNCL FFT Implementations

As shown in the above two charts, the number of points poses significant impacts on the gate counts of MTNCL FFT implementations. For non-pipelined designs, the gate count increases by 3.68× and 3.81× when going from 2-point to 4-point for 4-bit and 8-bit data widths, respectively; and these two numbers are 2.83× and 2.93× when going from 4-point to 8-point. For pipelined designs, the gate count increases by 2.95× and 2.98× when going from 4-point to 8-point for 4-bit and 8-bit data widths, respectively. Data width is another strong factor impacting the gate count. For all MTNCL FFT circuits, when going from 4-bit to 8-bit, the gate count increases from 3.22× to 3.44×.

Pipelining, however, does not affect the gate count nearly as much. As shown in Figures 14 and 15 below, for 4-point and 8-point designs, pipelining only increases the gate count from 4.6% to 12.2%. This is because the added MTNCL registers are much smaller than the original computational logic.

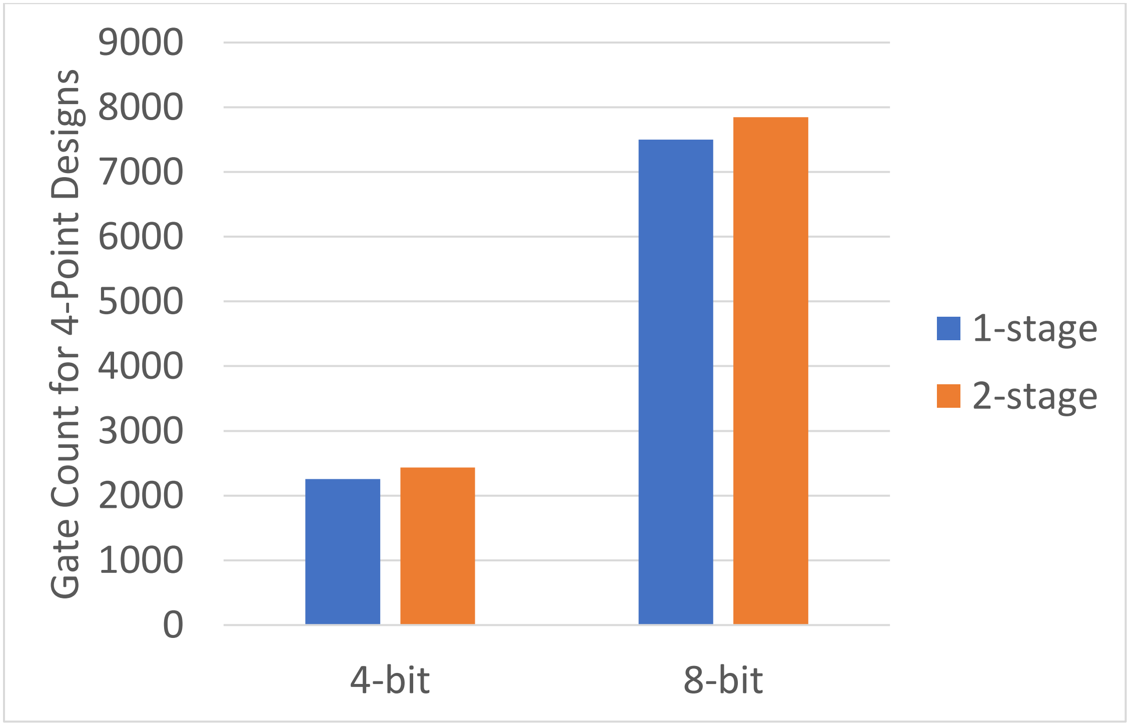

Figure 14: Gate Count Trend for 4-Point MTNCL FFT Implementations

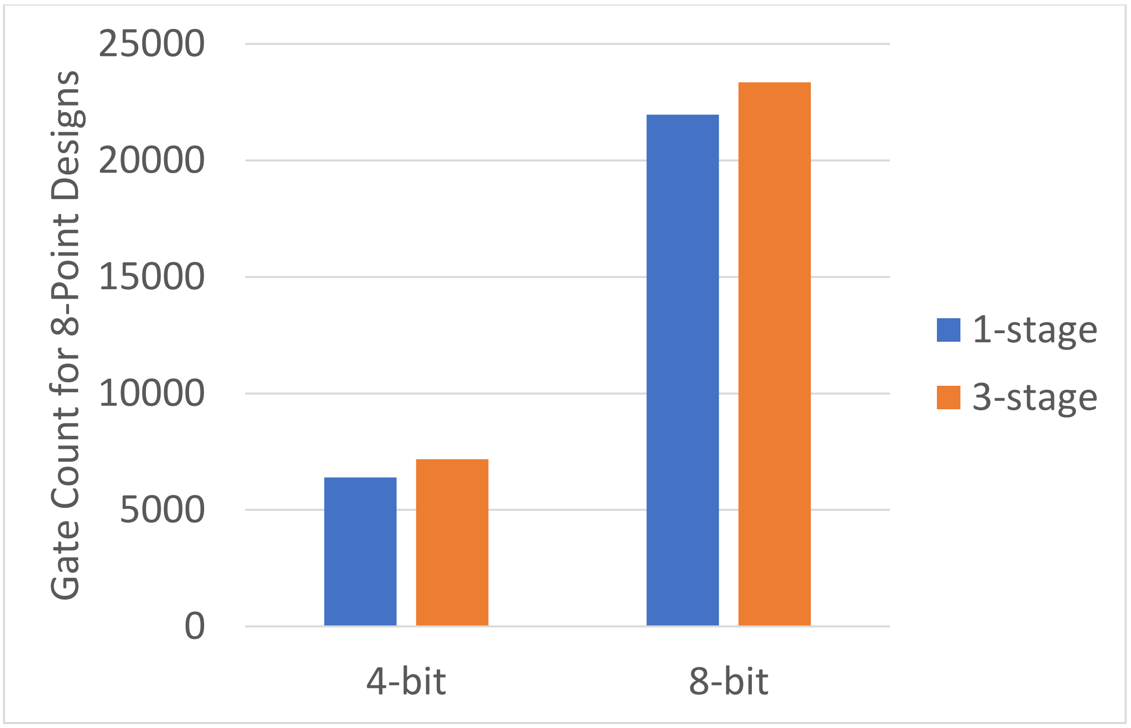

Figure 15: Gate Count Trend for 8-Point MTNCL FFT Implementations

4.2.2 Active Power Analysis

Using the data in Table 1, Figures 16 and 17 show the active power trends for non-pipelined and pipelined MTNCL FFT implementations, respectively.

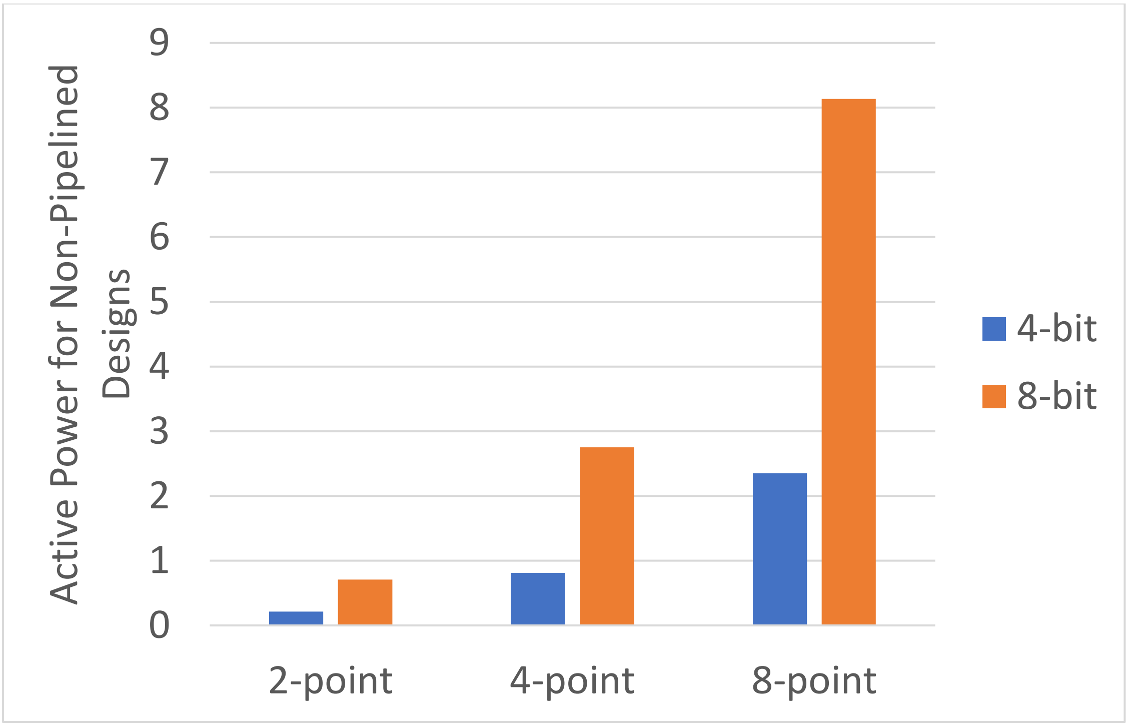

Figure 16: Active Power Trend for Non-Pipelined MTNCL FFT Implementations

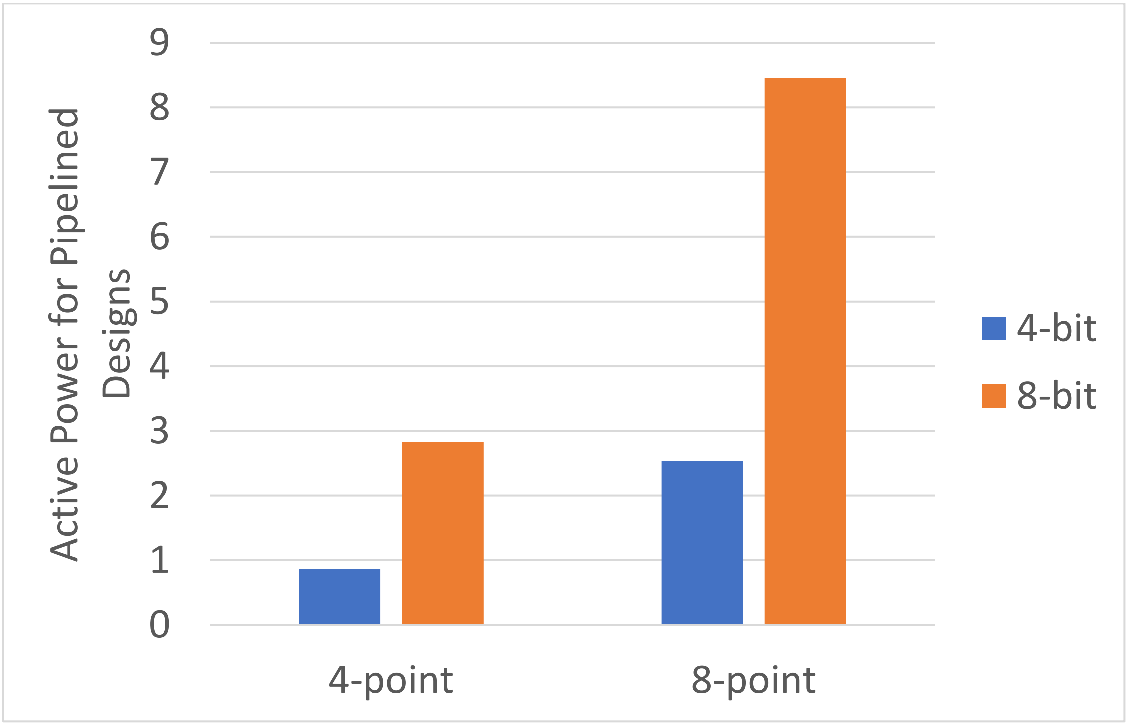

Figure 17: Active Power Trend for Pipelined MTNCL FFT Implementations

Similar to that of the gate count, the number of points poses significant impacts on the active power of MTNCL FFT implementations. For non-pipelined designs, the active power increases by 3.78× and 3.87× when going from 2-point to 4-point for 4-bit and 8-bit data widths, respectively; and these two numbers are 2.90× and 2.96× when going from 4-point to 8-point. For pipelined designs, the active power increases by 2.92× and 2.98× when going from 4-point to 8-point for 4-bit and 8-bit data widths, respectively. Data width is another strong factor impacting the active power. For all MTNCL FFT circuits, when going from 4-bit to 8-bit, the gate count increases from 3.27× to 3.46×.

Also similar to that of gate count, the impact of pipelining on active power is even smaller. As shown in Figures 18 and 19 below, for 4-point and 8-point designs, pipelining only increases the active power from 2.9% to 7.7%. This is also because the added MTNCL registers are much smaller than the original computational logic.

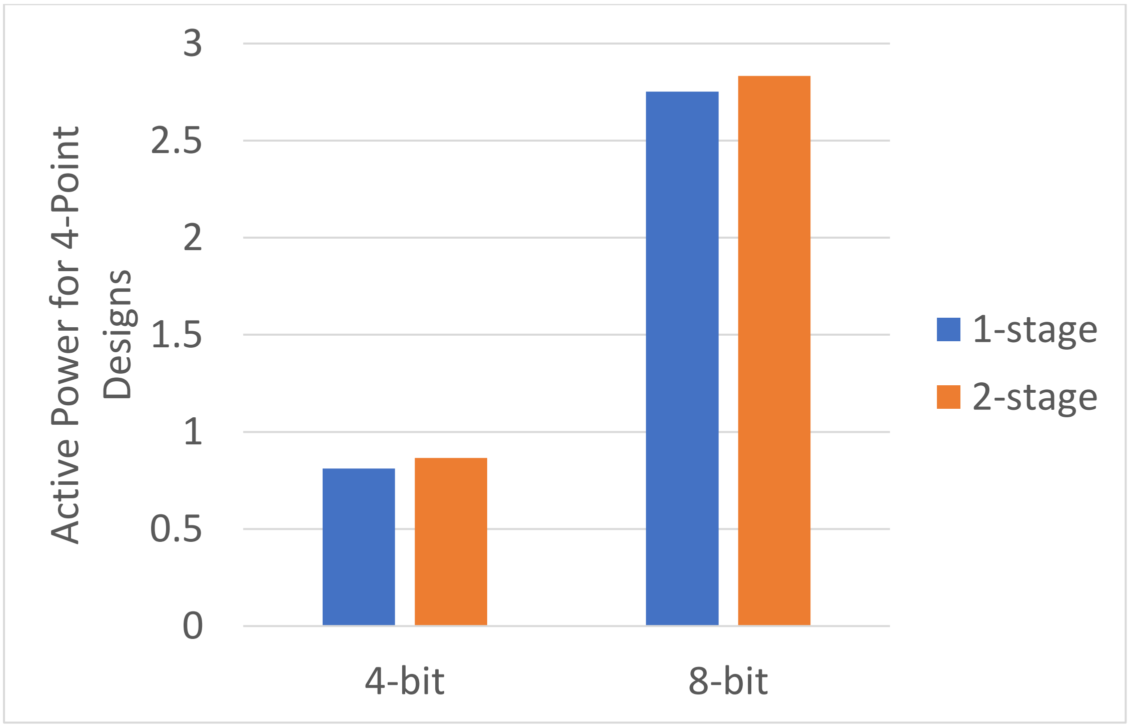

Figure 18: Active Power Trend for 4-Point MTNCL FFT Implementations

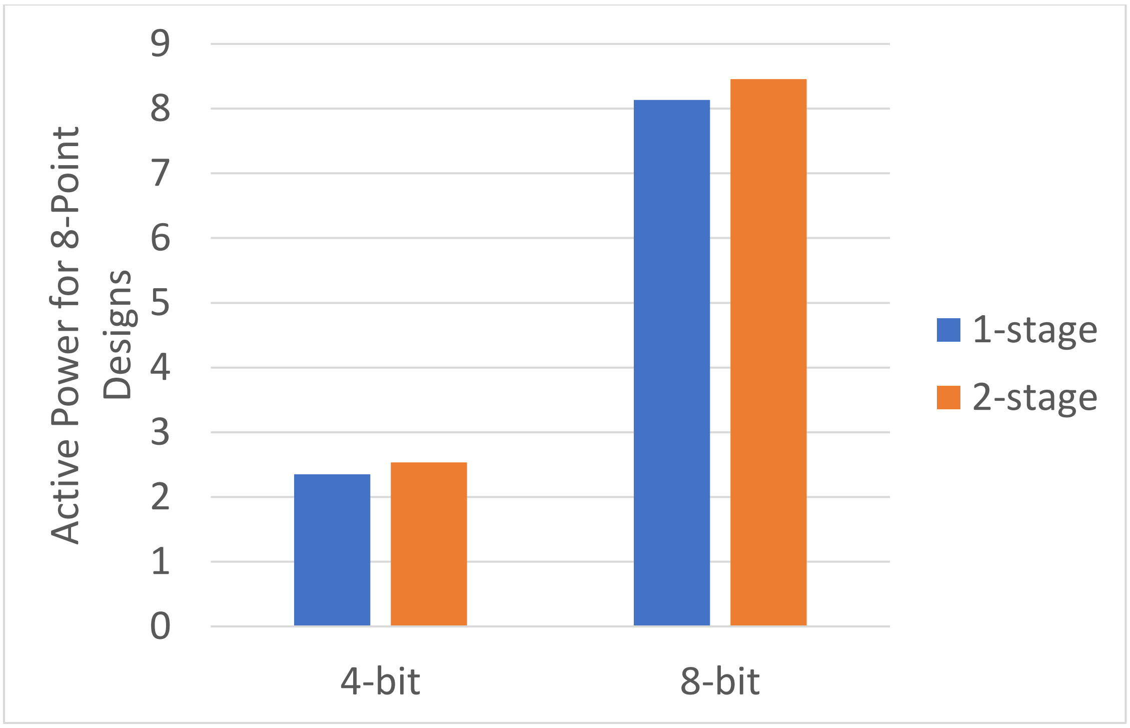

Figure 19: Active Power Trend for 8-Point MTNCL FFT Implementations

4.2.3 Leakage Power Analysis

Using the data in Table 1, Figures 20 and 21 show the leakage power trends for non-pipelined and pipelined MTNCL FFT implementations, respectively.

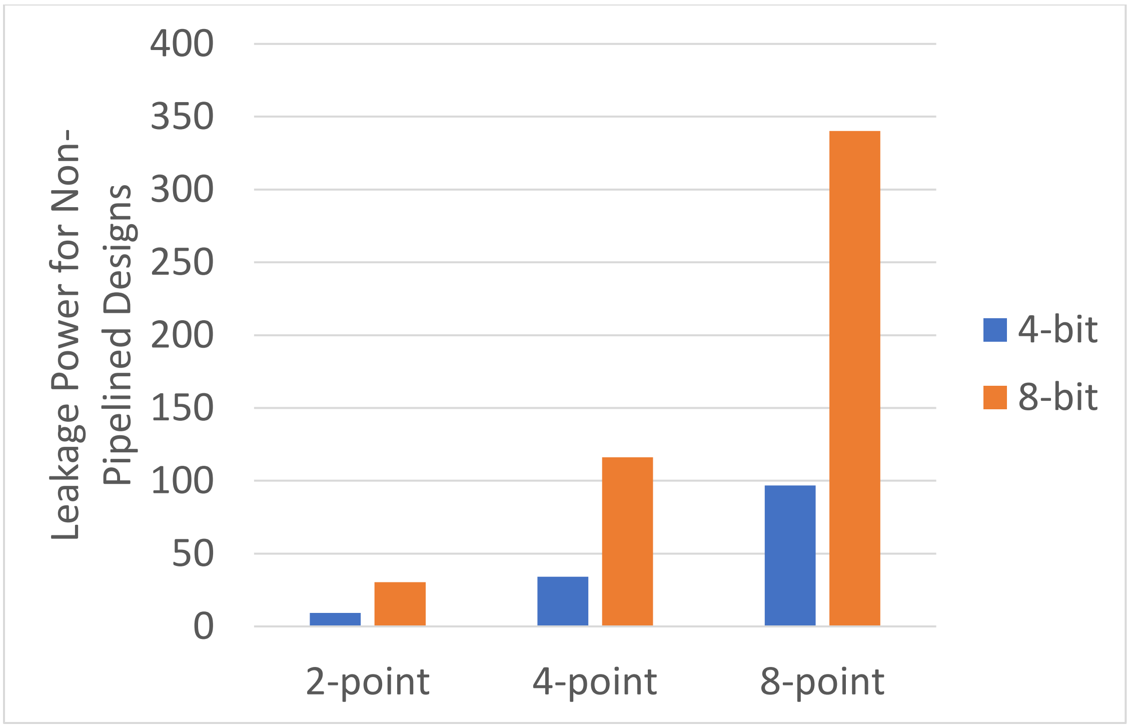

Figure 20: Leakage Power Trend for Non-Pipelined MTNCL FFT Implementations

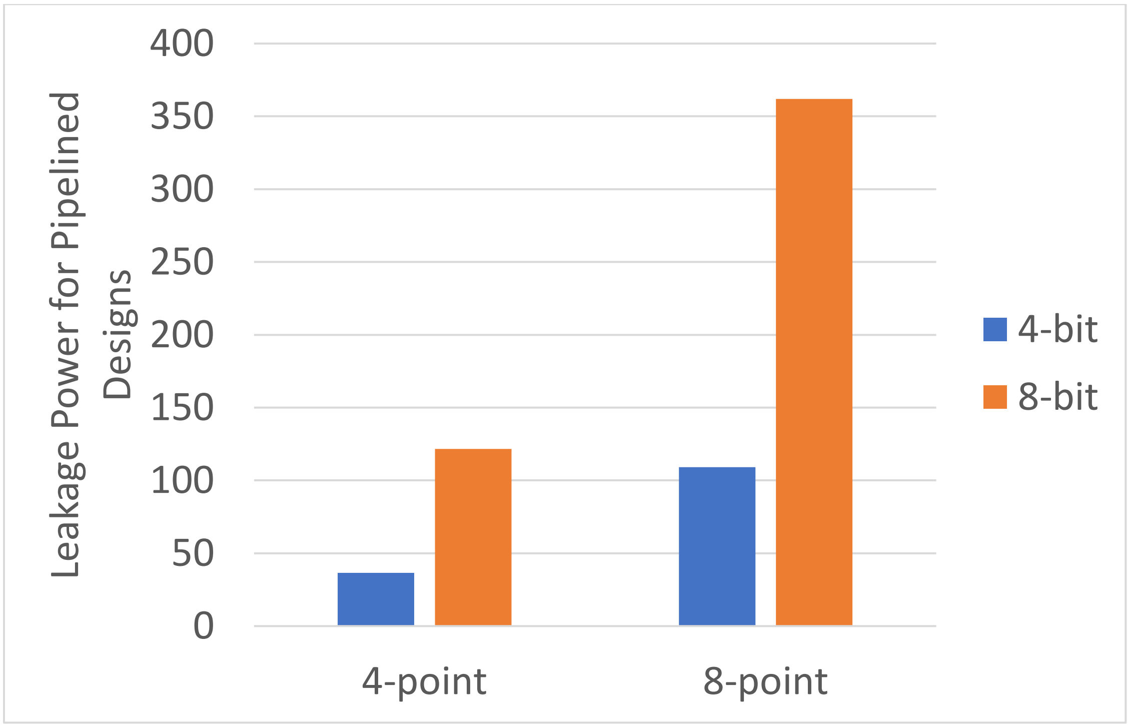

Figure 21: Leakage Power Trend for Pipelined MTNCL FFT Implementations

Leakage power follows the same trends as gate count since area is the most important factor of leakage. Therefore, the number of points poses significant impacts on the leakage power of MTNCL FFT implementations. For non-pipelined designs, the leakage power increases by 3.69× and 3.81× when going from 2-point to 4-point for 4-bit and 8-bit data widths, respectively; and these two numbers are 2.84× and 2.93× when going from 4-point to 8-point. For pipelined designs, the leakage power increases by 2.99× and 2.98× when going from 4-point to 8-point for 4-bit and 8-bit data widths, respectively. Data width is another strong factor impacting the leakage power. For all MTNCL FFT circuits, when going from 4-bit to 8-bit, the leakage power increases from 3.30× to 3.51×.

Also following the trend of gate count, the impact of pipelining on leakage power is small. As shown in Figures 22 and 23 below, for 4-point and 8-point designs, pipelining only increases the leakage power from 4.7% to 12.7%. This is also because the added MTNCL registers are much smaller than the original computational logic.

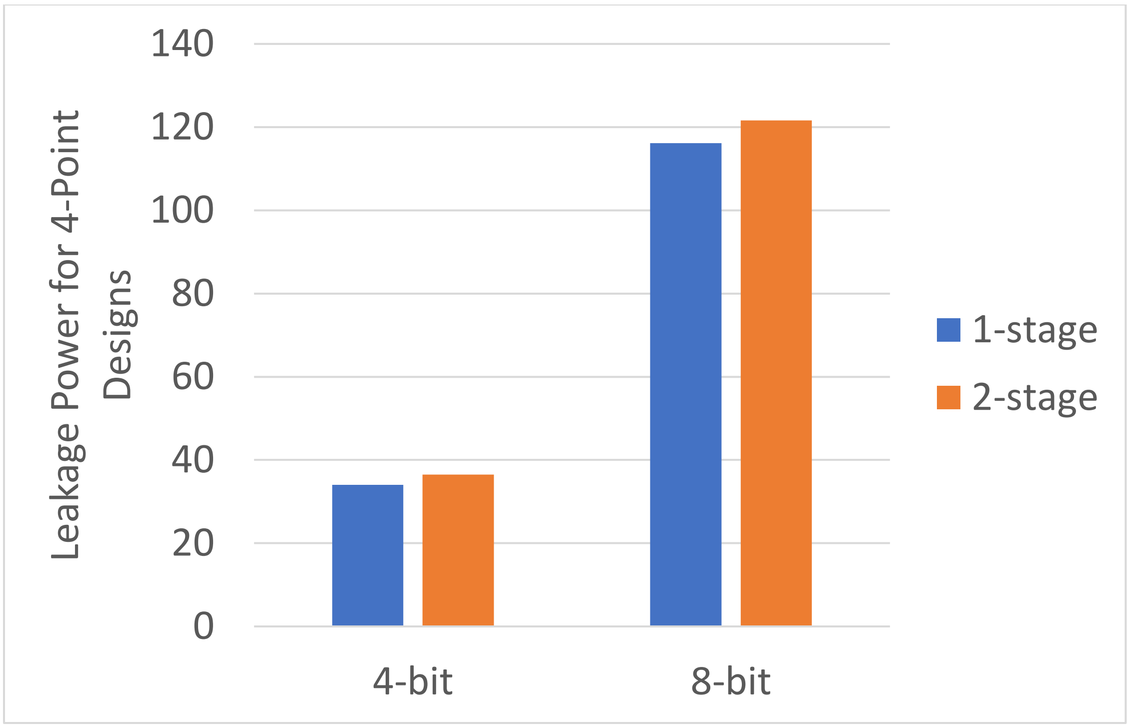

Figure 22: Leakage Power Trend for 4-Point MTNCL FFT Implementations

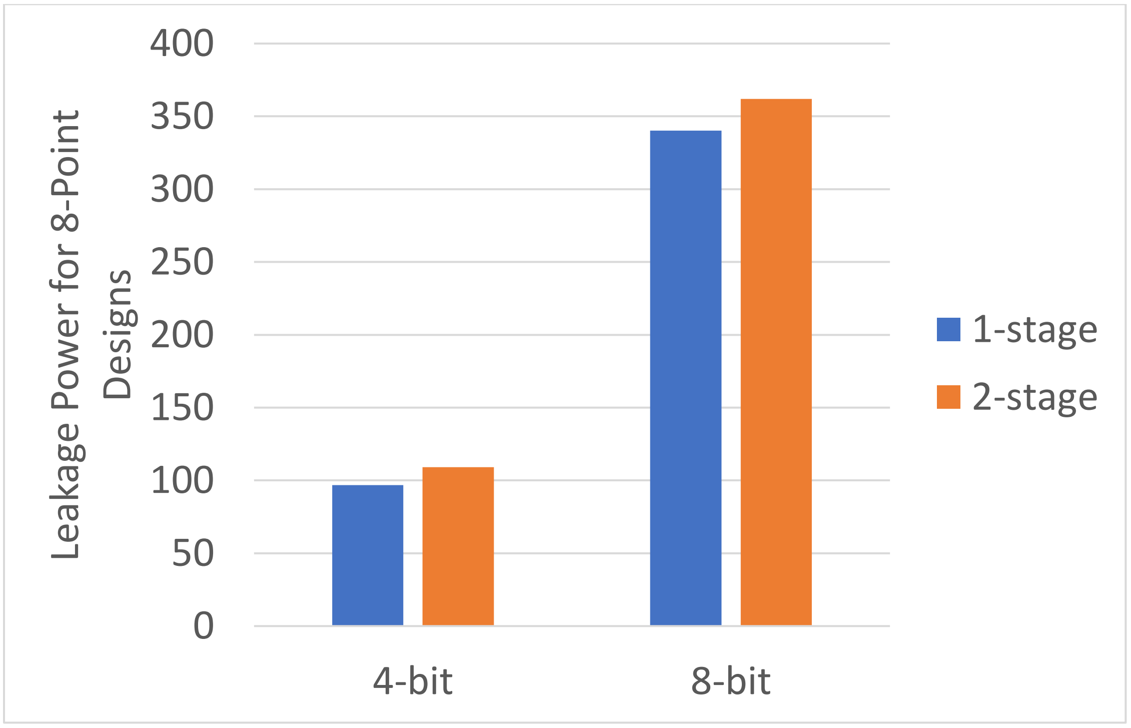

Figure 23: Leakage Power Trend for 8-Point MTNCL FFT Implementations

5. Conclusion

In this thesis work, 10 MTNCL FFT circuits were implemented using the TSMC 65nm technology. These circuits cover a spectrum from 2-point to 8-point, 4-bit to 8-bit, and 1-stage to 3-stage of pipelines. Power analysis shows that the active and leakage power of MTNCL FFT implementations is affected by these parameters differently. The number of points and data width pose much more significant impacts to the circuit area, which in turn affects the power consumption drastically. Pipelining, on the other hand, does not affect area much because the added MTNCL registers only occupy a small percentage of the entire logic. Therefore, pipelining is not a strong factor for power consumption. If MTNCL is used for implementing FFT algorithms in battery-powered embedded systems, pipelining could be considered for boosting up throughput, while using the fewest number of points and shortest data width to minimize active and leakage power consumption.

6. References

[1] https://www.ti.com/microcontrollers-mcus-processors/processors/digital-signal-processors/overview.html

[2] https://www.asml.com/en/news/stories/2022/what-is-a-gate-all-around-transistor#:~:text=In%20FinFET%20transistors%2C%20the%20gate,of%20current%20through%20the%20transistor.

[3] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," International Conference on Application Specific Systems, Architectures, and Processors, 1996.

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

[5] 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 and Applications, vol. 5, issue 2, pp. 81-100, May 2015

[6] L. Men and J. Di, "Asynchronous Parallel Platforms with Balanced Performance and Energy," Journal of Low Power Electronics, Vol. 10, No. 4, pp. 566-579, 2014

[7] L. Men and J. Di, "An Asynchronous Finite Impulse Response Filter Design for Digital Signal Processing Unit," IEEE Midwest Symposium on Circuits and Systems, Aug. 2014

[8] L. Men and J. Di, "Framework of Scalable Delay-Insensitive Asynchronous Platform Enabling Heterogeneous Concurrency," IEEE Midwest Symposium on Circuits and Systems, Aug. 2014

[9] L. Men, B. Hollosi, and J. Di, "Framework of an Adaptive Delay-Insensitive Asynchronous Platform for Energy Efficiency," IEEE International Symposium on VLSI, July 2014

[10] A. Bailey, A. Al Zahrani, G. Fu, J. Di, and S. Smith, "Multi-Threshold Asynchronous Circuit Design for Ultra-Low Power," Journal of Low Power Electronics, Vol. 4, NO. 3, pp. 337-348, December 2008

[11] L. Zhou, S. Smith, and J. Di, “Bit-Wise MTNCL: an Ultra-Low Power Bit-Wise Pipelined Asynchronous Circuit Design Methodology,” 2010 IEEE Midwest Symposium on Circuits and Systems, August 2010

[12] A. Alzahrani, A. Bailey, G. Fu, and J. Di, “Glitch-Free Design for Multi-Threshold CMOS NCL Circuits,” 2009 Great Lake Symposium on VLSI, May 2009

[13] A. D. Bailey, J. Di, S. C. Smith, and H. A. Mantooth, “Ultra-Low Power Delay-Insensitive Circuit Design,” 2008 IEEE Midwest Symposium on Circuits and Systems, Aug. 2008

[14] “Design of Low Energy, High Performance Synchronous and Asynchronous 64-Point FFT,” W. Lee, V. S. Vij, A. R. Thatcher, and K. S. Stevens, 2013 IEEE Design, Automation and TEst in Europe Conference (DATE)

[15] “Energy-Efficient Synchronous-Logic and Asynchronous-Logic FFT/IFFT Processors,” K. S. Chong, B. H. Gwee, and J. S. Chang, IEEE Journal of Solid-State Circuits, Vol. 42, No. 9, pp. 2034-2045, September 2007

[16] “Low Power QDI Asynchronous FFT,” B. Tang and F. Lane, 2016 IEEE International Symposium on Asynchronous Circuits and Systems

[17] “Implementation of an Asynchronous FFT Processor,” R. Seshasayanan, S. K. Srivatsa, and V. Sugavaneswaran, 2005 IEEE Annual India Conference

[18] “Implementation of Fast Fourier Transform Processor in NULL Convention Logic,” Zhen Song, M.S. thesis, University of Arkansas, 2011