NTNU | Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

TFE4152 Design of Integrated Circuits

Term project fall 2023

Multiply-Accumulate Unit

Authors:
Magnus Oscar Støleggen

Abstract

In this project, an 8-bit multiply-accumulate unit governed by a finite state machine has been successfully designed. Comprehensive assessments utilizing hardware descriptive language and Spice simulations affirm the system's reliable functionality, with the exception of an anomaly in the registers of the multiply-accumulate before the defining edge of the first clock cycle, although expected to not have an impact on a physical realization of the system.

The focal point of the design was the optimization of the multiply-accumulate unit's registers for maximum speed and energy efficiency, a goal that has been accomplished, although it could be further improved upon.

Table of Contents

1 Introduction

In this outlined project, the goal is to design an 8-bit multiply-accumulate unit using digital logic and 90 nm CMOS transistor technology. This unit will feature two 2-bit inputs for its multiplication stage and will operate under the control of specific signals generated by a dedicated finite state machine, also designed for this project. The finite state machine's instructions will orchestrate the multiply-accumulate unit's calculations over three clock cycles followed by an idle cycle. Additionally, it will ensure that the unit functions only when a designated "Run" signal is active, allowing the system's operational cycle to pause when this signal is set to low. Furthermore, a "Reset" signal will be incorporated to effectively reset both the finite state machine and the registers within the multiply-accumulate unit.

A primary emphasis will be placed on designing the registers of the multiply-accumulate circuit to prioritize reliability, power efficiency, and speed.

The motivation driving the development of such a unit stems from its ability to execute repetitive arithmetic tasks efficiently, utilizing minimal hardware resources. Particularly valuable in applications like digital signal processing, filtering algorithms, and neural networks, the multiply-accumulate operation serves as a fundamental and recurring function. By consolidating these operations into a dedicated unit, considerable enhancements in computational speed and resource utilization can be achieved.

Additionally, the 8-bit precision caters to scenarios where high precision is not a critical requirement. This characteristic reduces hardware complexity and power consumption, making it a fitting choice for applications where limitations in space, power, and cost play significant roles.

2 Theory

In this section, essential theoretical underpinnings for the realization of a multiply-accumulate (MAC) circuit and a finite state machine at the transistor level is presented.

2.1 Field-Effect Transistors (FETs)

Field-effect transistors represent a cornerstone in contemporary electronics design. Their ability to be fabricated at a minuscule scale renders them the predominant components in integrated circuit design. This dominance is attributed to their remarkable versatility, facilitating their deployment in diverse applications such as signal amplification and digital logic circuits [6].

The two primary types of FETs are NMOS and PMOS, which share a common structure but differ in the doping of their regions and substrate. NMOS devices employ a p-type substrate and n-doped regions for terminals, while PMOS devices feature an n-type substrate and p-doped regions, resulting in their complementary behavior [6].

2.1.1 Complementary Metal-Oxide-Semiconductor (CMOS)

CMOS-technology leverages both NMOS and PMOS transistors to construct integrated circuits for both digital and analog applications. CMOS has become the predominant IC technology, supplanting bipolar devices in numerous domains due to its versatility and efficiency [6].

To enable current flow through a PMOS transistor, a negative gate-source voltage exceeding a certain threshold is necessary. This establishes a conduction channel between the source and drain terminals. The condition for channel formation can be expressed as

$$|V_{gs}| \geq |V_t| \tag{1}$$

To induce a current i_D between the drain and source terminals, a negative voltage v_{DS} is applied to the drain terminal. The electron mobility within the channel is described by

$$\mu_p C_{ox} \tag{2}$$

and the transconductance parameter k_p is proportionate to the transistor's aspect ratio, denoted as

$$k_p = \mu C_{ox} \frac{W}{L} \tag{3}$$

[6].

An essential contrast between PMOS and NMOS transistors lies in their carrier mobility. PMOS transistors primarily utilize holes as carriers, which possess lower mobility than the electrons predominant in NMOS transistors. Consequently, PMOS transistors typically demonstrate slower switching speeds and reduced performance in comparison to NMOS transistors. Nonetheless, leveraging both transistor types in complementary configurations, such as in CMOS technology, offers compensatory benefits, a topic we will delve into further in this article.

2.2 Static power consumtion

Calculating the static power consumption in a CMOS integrated circuit is essential for estimating the device's power efficiency. Static power, also known as leakage power, is the power dissipated when the transistors are in their off state, which occurs when the circuit is not actively switching. It

is a significant component of the total power consumption in modern ICs. The primary contributors to static power are subthreshold leakage and gate leakage currents.

The static power consumption (P_{static}) in a CMOS integrated circuit can be estimated using the following equations:

1. **Subthreshold Leakage Power (P_{sub})**:

The subthreshold leakage power is due to the flow of subthreshold current through the transistors. It is given by the following equation:

$$P_{\text{sub}} = I_{\text{sub}} \cdot V_{\text{DD}} \tag{4}$$

where: I_{sub} is the subthreshold leakage current. V_{DD} is the supply voltage.

2. **Gate Leakage Power (P_{gate})**:

The gate leakage power is associated with the leakage current through the gate oxide. It can be estimated as:

$$P_{\text{gate}} = I_{\text{gate}} \cdot V_{\text{DD}} \tag{5}$$

where: - I_{gate} is the gate leakage current.

3. **Total Static Power (P_{static})**:

The total static power consumption is the sum of the subthreshold leakage power and the gate leakage power:

$$P_{\text{static}} = P_{\text{sub}} + P_{\text{gate}} \tag{6}$$

In practice, the values of I_{sub} and I_{gate} are dependent on the process technology, transistor characteristics, and the specific design of the CMOS circuit. To minimize static power, designers often employ various techniques such as power gating, voltage scaling, and transistor sizing to reduce leakage currents.

2.3 Process Variations

CMOS device parameters are sensitive to the fabrication process, resulting in variations in device properties even for identical designs. Factors like fabrication temperature can introduce variability, causing oxide thickness to fluctuate by approximately 5% and dopant concentrations to deviate by about 10%. Designers mitigate these variations by employing different device models. Corner analysis assesses the influence of "fast" and "slow" transistors on the design, categorizing corners as FF (Fast Fast), SS (Slow Slow), SF (Slow Fast), FS (Fast Slow), and TT (Typical Typical) [2].

2.4 Simulation Program with Integrated Circuit Emphasis (SPICE)

Simulation Program with Integrated Circuit Emphasis (SPICE) is an established open-source simulation language developed at the University of California, Berkeley, in the 1970s. Today, it is chiefly employed for simulating small integrated sub-circuits and discrete circuits, while Very Large Scale Integration (VLSI) necessitates faster simulation tools due to SPICE's relatively sluggish performance. SPICE supports a broad spectrum of simulations, including DC, AC, transient, and noise analysis, offering the flexibility to incorporate device models tailored to specific components. For CMOS designs, simulations encompass all process corners to account for production-related performance variations [5].

2.5 Hardware Description Language (HDL)

Designing intricate digital circuits manually becomes unmanageable due to their complexity. Hardware Description Language (HDL) emerges as the primary means for digital designers to create and describe hardware in a textual format, significantly reducing the risk of producing erroneous designs [4]. HDL defines circuits implicitly, detailing the relationships among signals rather than specifying the actual functions. VHDL and Verilog represent the most widely used HDLs [1] [7], providing high-level descriptions of digital designs. Notably, the release of SystemVerilog, a superset of Verilog, in 2005 further extended the capabilities of Verilog [3].

3 Method

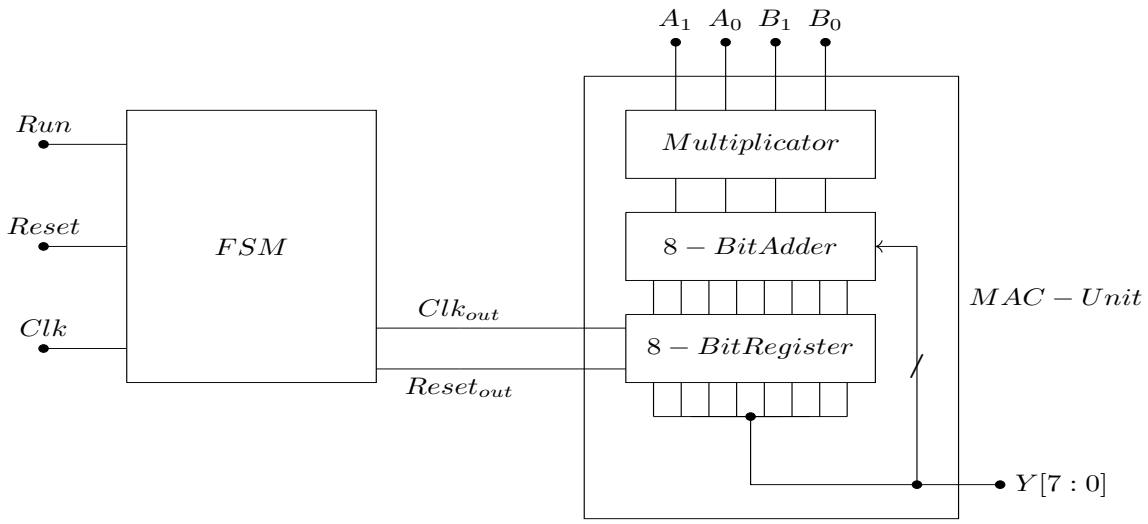

Figure 1: Block diagram of the system containing the finite state machine and MAC-unit with it's components

The system, as per specifications, comprises two distinct units: the finite state machine and the 8-bit Multiply-Accumulate (MAC) unit. The block diagram representing this system is detailed in Figure (1). Further specifics regarding the design of both the FSM- and MAC-units are delineated in sections (3.1) and (3.2) correspondingly.

3.1 Finite State Machine

The implementation of the finite state machine involves employing two positive edge-triggered master-slave D-flip-flops, as depicted in Figure (4), and a system of logic gates to process the three input signals. Within the FSM, the two registers are capable of retaining four distinct states. Among these states, three prompt the MAC-Unit to execute a calculation, while the fourth state commands it to remain idle for a single clock cycle. The gate-level schematics as well as the state diagram for the FSM is illustrated in Figure (2) and (3) respectively.

Three inputs govern the FSM: "Run," "Reset," and the clock signal. When the "Run" signal is active, the FSM progresses continuously through its four states, executing three calculations followed by an idle clock cycle. Contrastingly, when "Run" is inactive, the MAC-unit remains idle until the signal reactivates, at which point it resumes from its prior state.

To achieve this functionality, the clock signal that regulates both the FSM's internal registers and those of the MAC-unit is synchronized with the "Run" signal. As long as the "Run" signal remains low, no clock signal is dispatched to either the FSM's registers or those of the MAC-unit. Similarly, when the FSM enters its idle state, the FSM's output clock signal Clk_{Out} remains unchanged for the MAC-unit, while the FSM's registers continue to receive the clock signal as usual.

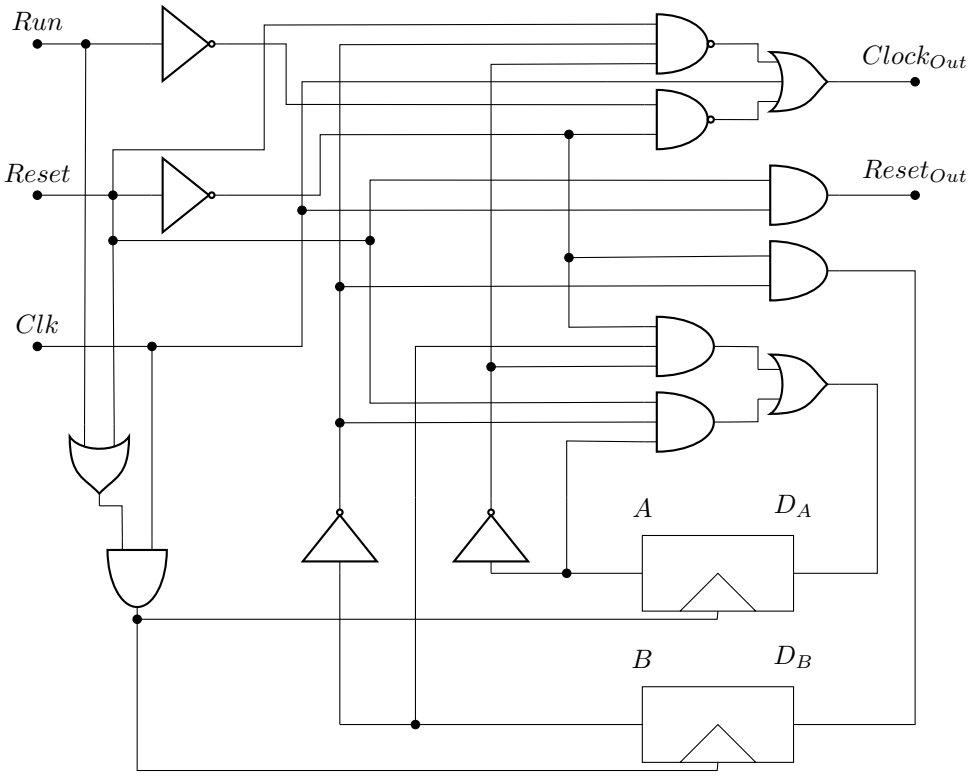

Figure 2: Gate-level schematic of the Finite State Machine using two positive edge triggered D-Flip-Flops.

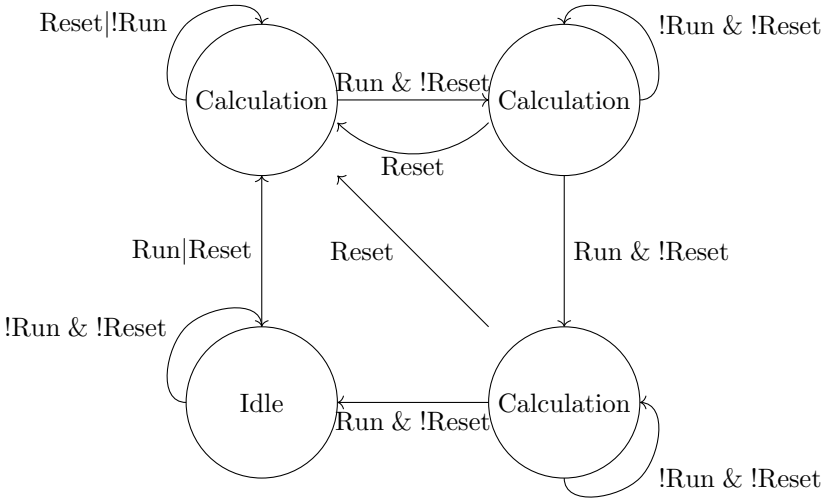

Figure 3: State diagram for the finite state machine.

The "Reset" signal serves the dual purpose of resetting both the FSM's internal registers back to the initial state and zeroing all registers within the MAC-Unit. Achieving this involves a sequence of logic operations applied to the input signal directed to the two internal registers. Consequently, when the "Reset" signal is activated, both internal registers receive a low value, initiating the reset process. The raw "Reset" signal also undergoes routing through an AND-gate alongside the raw clock signal and serves as the FSM's "$Reset_{Out}$" output. Subsequently, the resultant signal is directly conveyed to the reset-input of each of the eight registers within the MAC-Unit. This configuration effectively nullifies all stored values within the registers, ensuring a complete reset of the system.

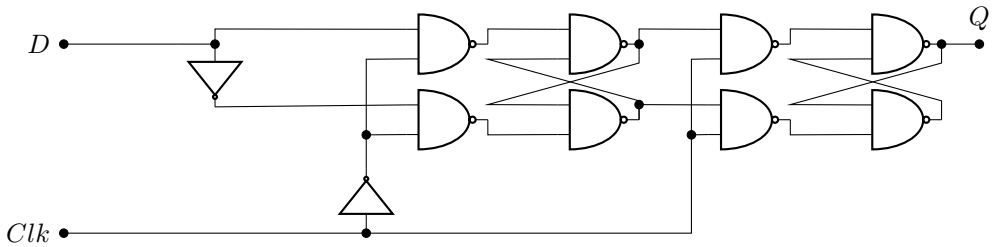

Figure 4: Logic gate schematic of positive edge triggered master-slave D-Flip-Flop.

3.2 Multiply-Accumulate unit

The MAC-Unit, as illustrated in Figure (1) comprises three essential components: a 2-bit multiplier, an 8-bit adder, and an 8-bit register equipped with reset functionality. These elements function in tandem to perform arithmetic operations within the system. The schemtics of the 2-bit multiplier and the 8-bit adder is illustrated in Figures (5) and (6) respectively.

The multiplier and adder are constructed using logic gates, while the design of the register operates on a transistor level, employing pass-logic and transmission gates. Further insights into this design are elaborated upon in section (3.3).

Functionally, the multiplier accepts two 2-bit numbers, $A[1:0]$ and $B[1:0]$, processing them to produce the product $X[3:0]$. This resultant product is subsequently channeled into the 8-bit adder, where it's combined with the ongoing output $Y[7:0]$ sourced from the register. The result of this summation $Z[7:0]$ is fed into the registers.

During each negative edge[1] of a clock cycle, the register undergoes an update to retain $Z[7:0]$. This updated value is then relayed back into the adder, constituting the fundamental "accumulate" stage within the MAC-unit's operations. This cyclic process enables the MAC-unit to execute its designated functions efficiently.

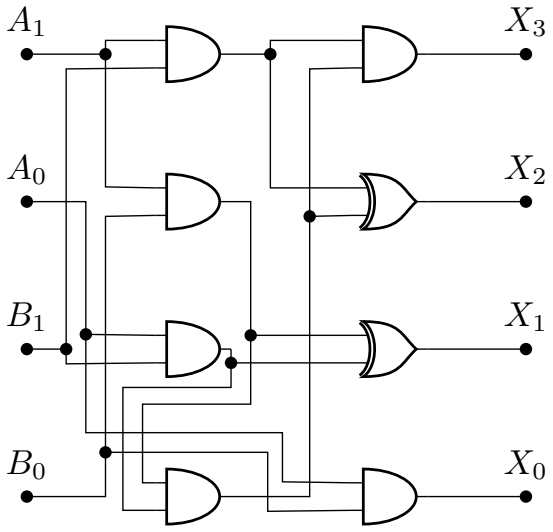

Figure 5: Gate-level schematic of the 2-bit multiplier within the MAC-Unit.

[1]Refer to section 3.3 for elaboration on why the MAC-unit employs negative, not positive, edge triggered registers.

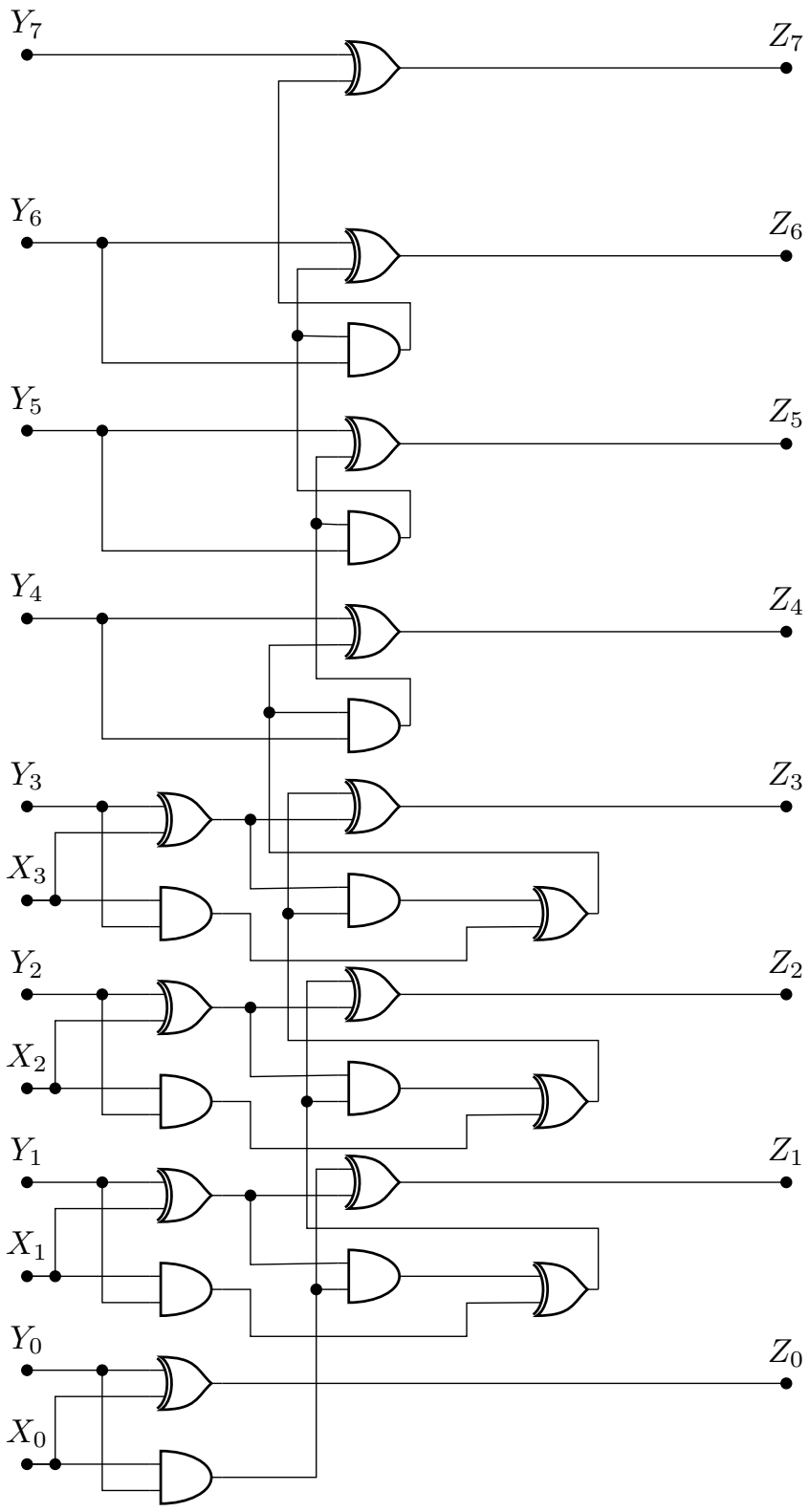

Figure 6: Gate-level schematic of the 8-bit adder within the MAC-Unit. Z[7:0] represent the output from the adder, Y[7:0] the output from the MAC-unit register, and X[3:0] the output product from the 2-bit multiplier.

3.3 MAC-Unit Registers

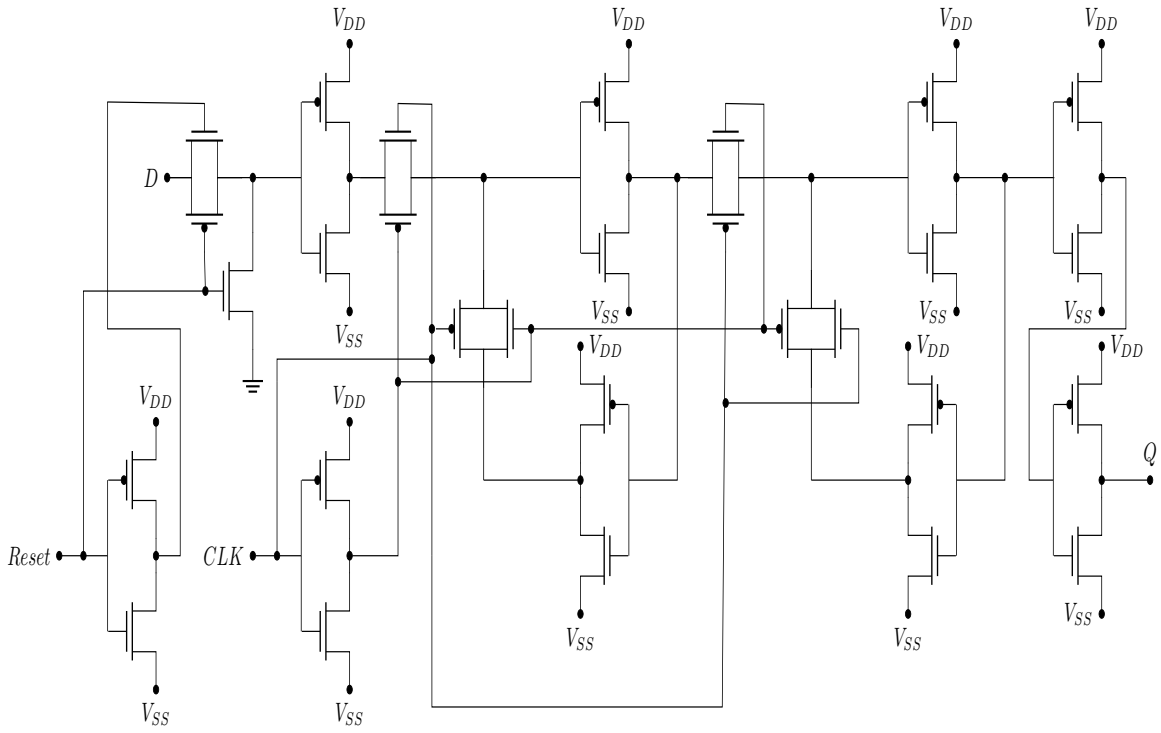

Figure 7: Transistor-level schematic of the 1-bit register. The schematic represents a negative edge triggered D-flip-flop with a reset signal.

The registers within the MAC unit are meticulously crafted at a transistor level rather than merely at the gate level. Leveraging pass-logic and transmission gates, this design emphasizes both low power consumption and rapid switching speeds. Employing a negative edge triggered master-slave D-Flip-Flop configuration, these registers demonstrate optimal performance at a clock frequency of 1 GHz.

Operating at such high clock frequencies poses challenges, particularly concerning signal delays originating from the FSM. These delays could potentially lead to undesirable behavior within the MAC unit. To address this concern, the registers within the MAC unit are intentionally designed to be negative edge triggered, while those within the FSM operate on positive edge triggers. This intentional distinction creates a buffer of half a clock cycle (0.5 ns) within the system.

For a comprehensive understanding of the circuit's topology, refer to the depicted outline in Figure (7).

The register outlined in Figure (7) underwent meticulous design and validation through SPICE simulations. A SPICE netlist simulating this 1-bit representation of the MAC-unit's register was devised, as detailed in section 6, utilizing provided device models. This simulation serves to confirm the register's functionality and explore its power consumption profile.

To optimize performance while minimizing voltage drop across the transistors, the lengths of PMOS transistors were tailored to be as short as possible, while maintaining appropriate behavior (Table 1). Conversely, PMOS transistor widths were maximized. Achieving a balanced configuration, NMOS transistors were set to match the strength of the PMOS transistors. Consequently, NMOS transistors shared identical lengths but with slightly smaller widths, aligning with the mobility relationship between these transistors (ref. section 2.1.1). To achieve the optimal functionality, supply voltage V_{DD}, as well as all input signals were set to be 1V.

Table 1: Transistor dimension specifications and supply voltage V_{DD} for the registers.

Transistor	Length (µm)	Width (µm)
NMOS	0.35	0.7
PMOS	0.35	1.5
V_{DD}	1 V	

The testing process involves simulating the circuit within SPICE using various squarewave signals with different periodic structures to excite inputs D, $Reset$, and Clk. Observing the behavior of the output signal Q serves as the primary indicator to confirm whether the circuit operates as expected or encounters discrepancies.

The SPICE-simulation also allows investigation of the impact of process variations. These simulations encompass specific corner types mentioned in Section 2.3, exploring the corners denoted as TT, FF, SS, SF, and FS at temperatures 0, 27, and 70 degrees centigrade.

3.4 Verilog simulation

To validate the overall functionality of the designed system, a simulation using hardware description language, specifically Verilog, was formulated. In this simulation, the negative edge triggered registers detailed in section (3.3) were represented at the gate-level, diverging from the previous transistor-level modeling. This approach aligns with the schematic illustrated in Figure (8). The code utilized to simulate the system is thoroughly documented in Appendix B.

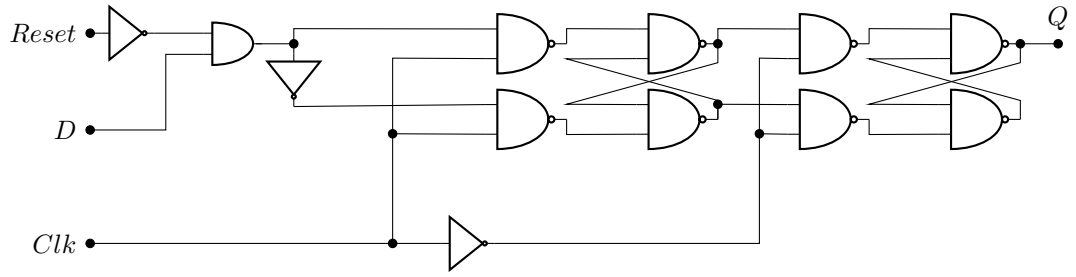

Figure 8: Logic gate schematic of negative edge triggered master-slave D-Flip-Flop with reset signal.

4 Results

The result section encompasses the simulations of the complete system using Verilog, as well as the SPICE-simulations of functionality of the 1-bit register from Figure (7) for all process parameters and temperatures 0℃, 27℃, and 70 ℃, as well as the static power consumption for the TT and FF corners at 27℃

4.1 SPICE-simulation of 1-bit register

All simulations are done utilizing the SPICE-code provided in Appendix A.

4.1.1 Process parameters: TT

Observing Figures (9), (10), and (11), it's evident that the register functions correctly. Notably, at 0℃ and 27℃, Q registers high at $t = 0$s in the simulations, but at 70℃, it registers low.

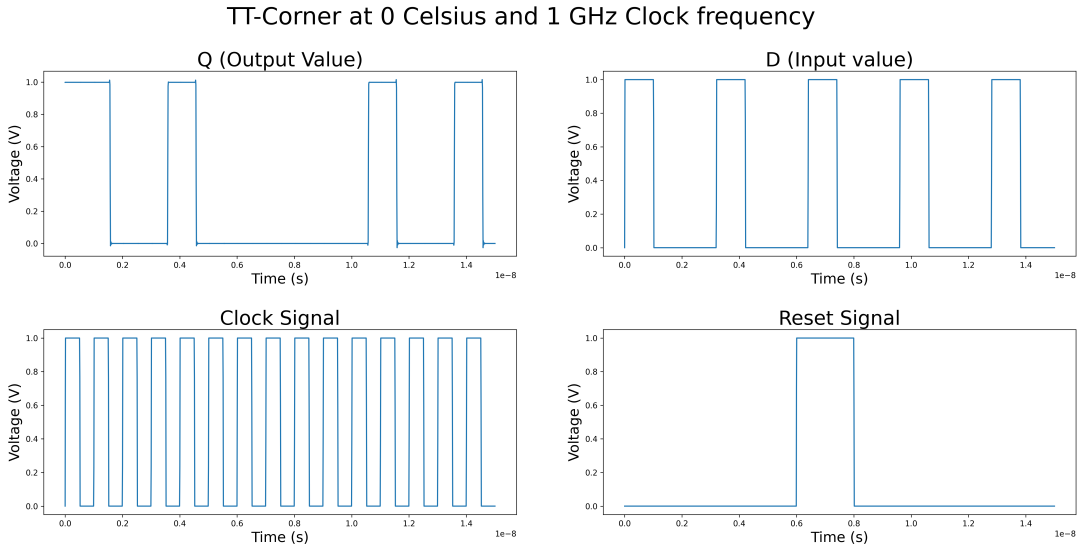

Figure 9: Functionality test for TT-corner at 0.

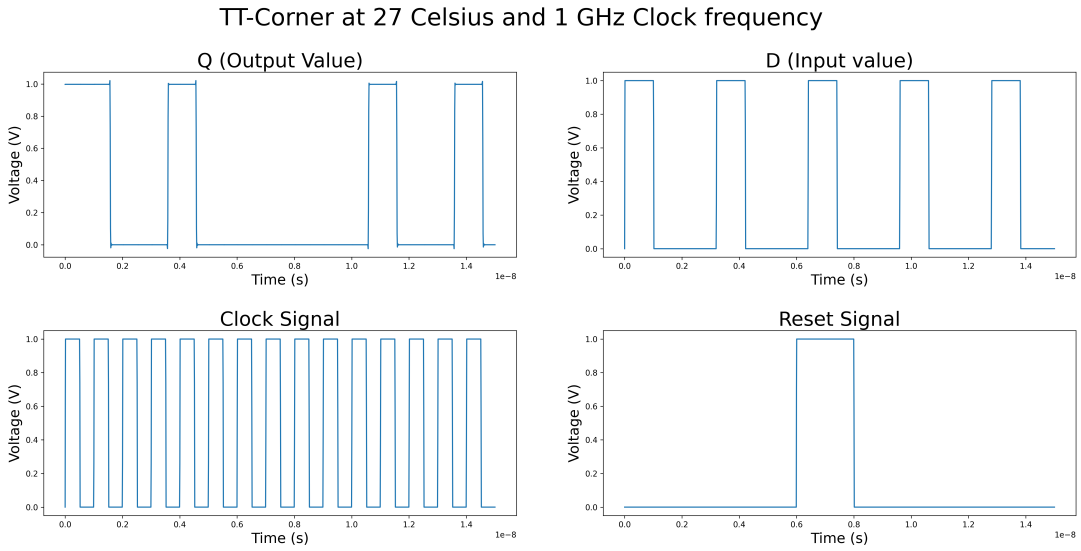

Figure 10: Functionality test for TT-corner at 27.

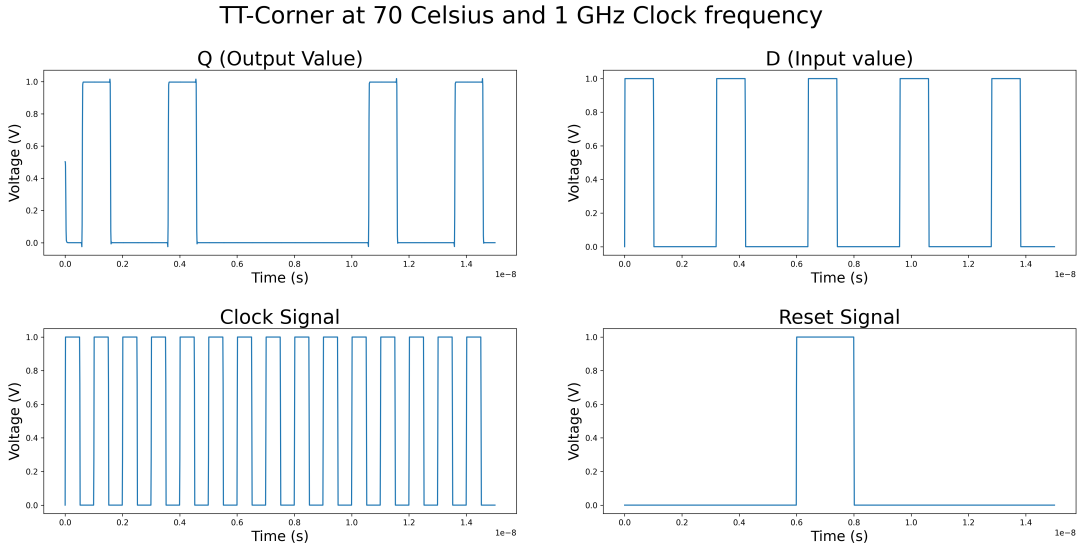

Figure 11: Functionality test for TT-corner at 70.

4.1.2 Process parameters: FF

Similar to the TT-corner observations, Figures (12), (13), and (14) depict the expected behavior of the register. However, in the FF-corner, a similar anomaly related to initial conditions emerges. Specifically, only the 0℃ simulation yields an initial high value for Q.

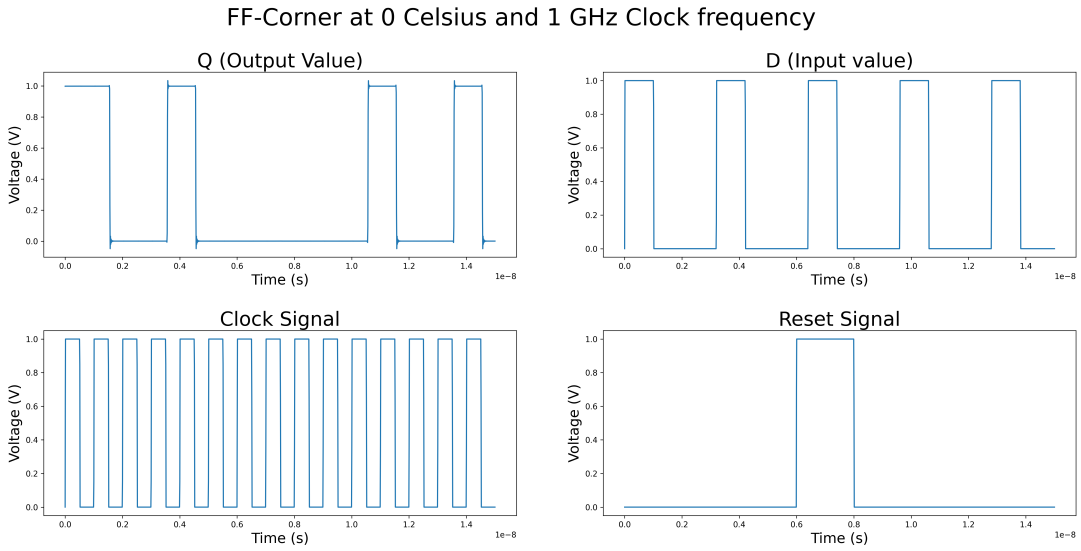

Figure 12: Functionality test for FF-corner at 0.

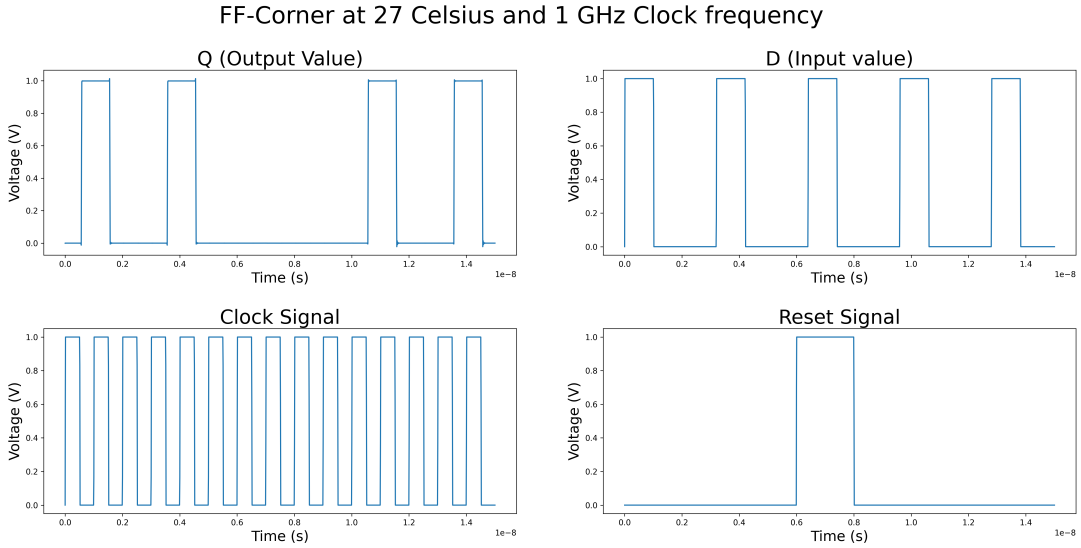

Figure 13: Functionality test for FF-corner at 27.

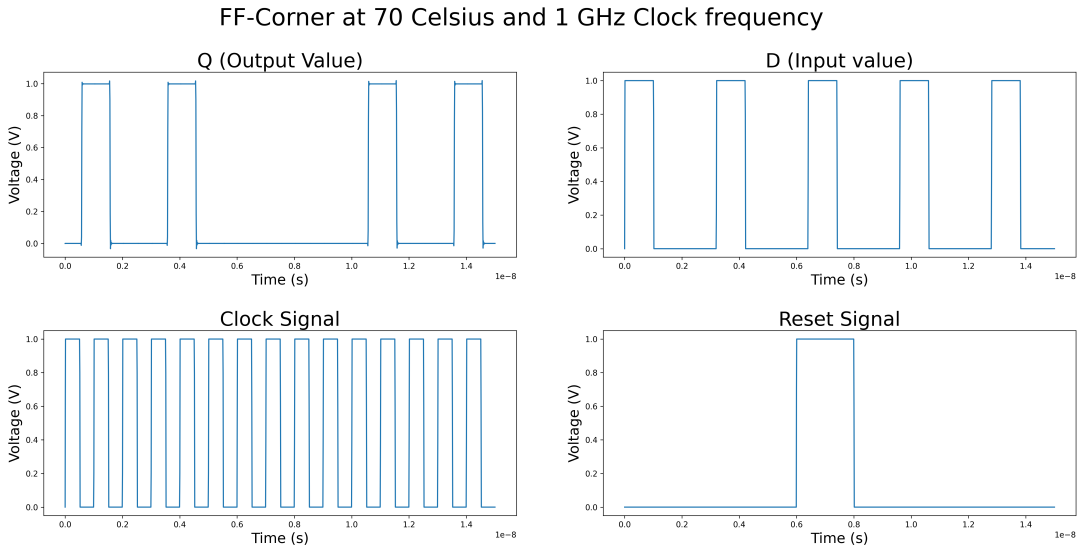

Figure 14: Functionality test for FF-corner at 70.

4.1.3 Process parameters: SS

Figures (15), (16), and (17) showcase the simulation results for the SS-corner, demonstrating the expected behavior of the register. Similar to the observations in the TT- and FF-corners, the SS-corner also presents an initial condition anomaly. However, in this case, the initial value of Q is low only for the simulation conducted at 0℃.

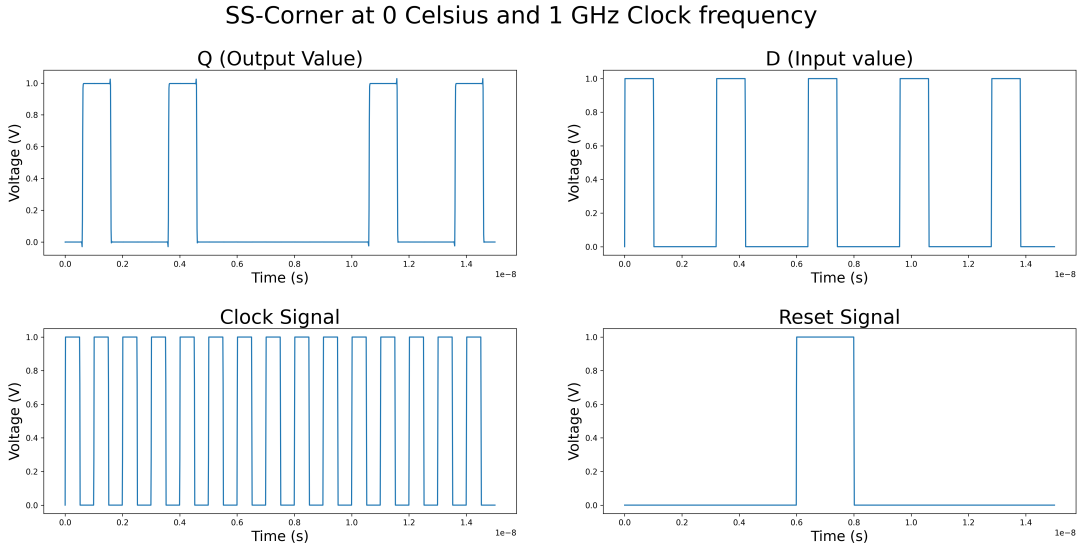

Figure 15: Functionality test for SS-corner at 0.

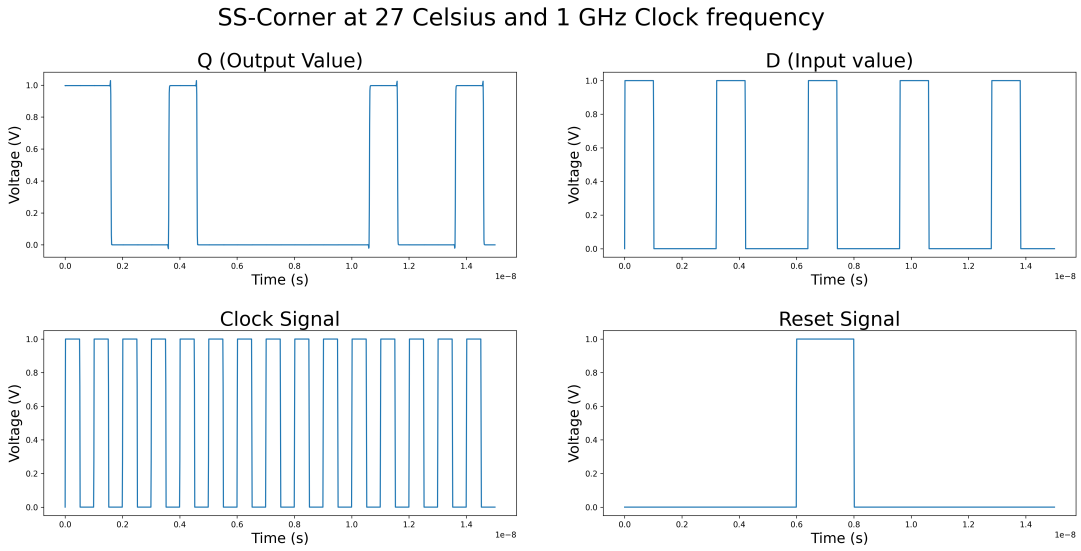

Figure 16: Functionality test for SS-corner at 27.

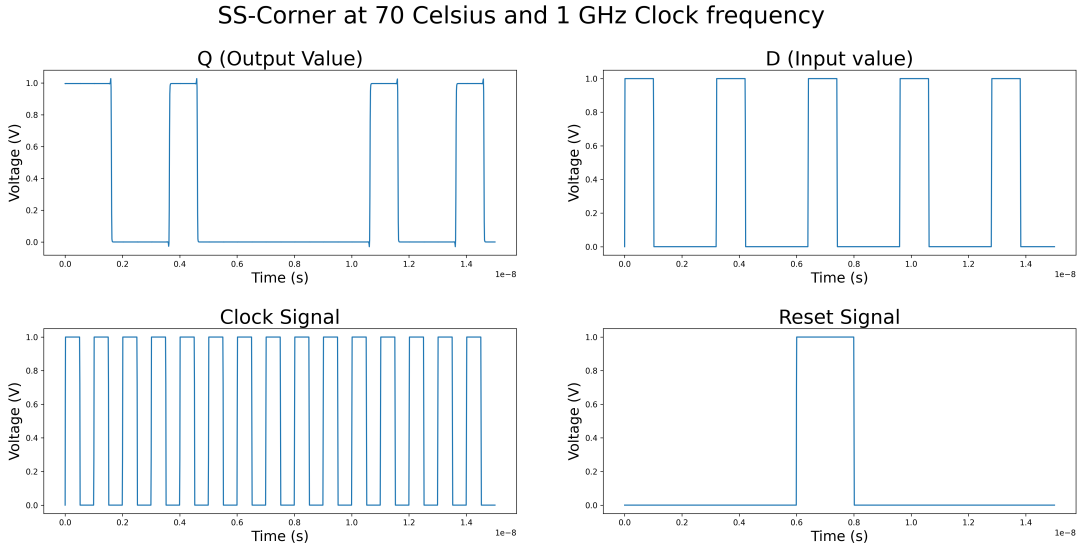

Figure 17: Functionality test for SS-corner at 70.

4.1.4 Process parameters: SF

Figures (18), (19), and (20) depict the simulation results for the SF-corner. Unlike the simulations conducted for the TT-, SS-, and FF-corners, these simulations demonstrate no indications of the initial condition anomaly. The register performs as expected at all times.

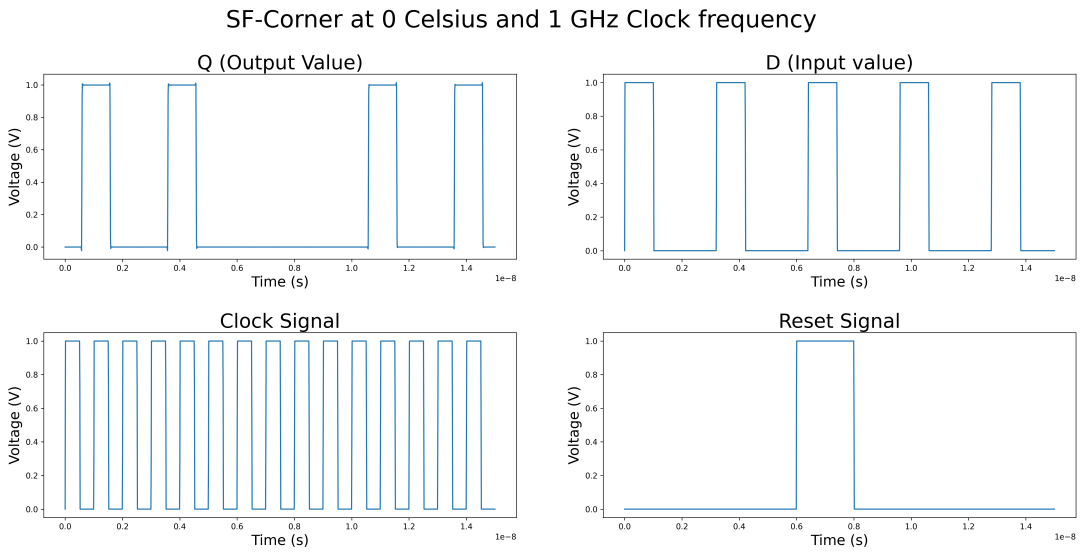

Figure 18: Functionality test for SF-corner at 0.

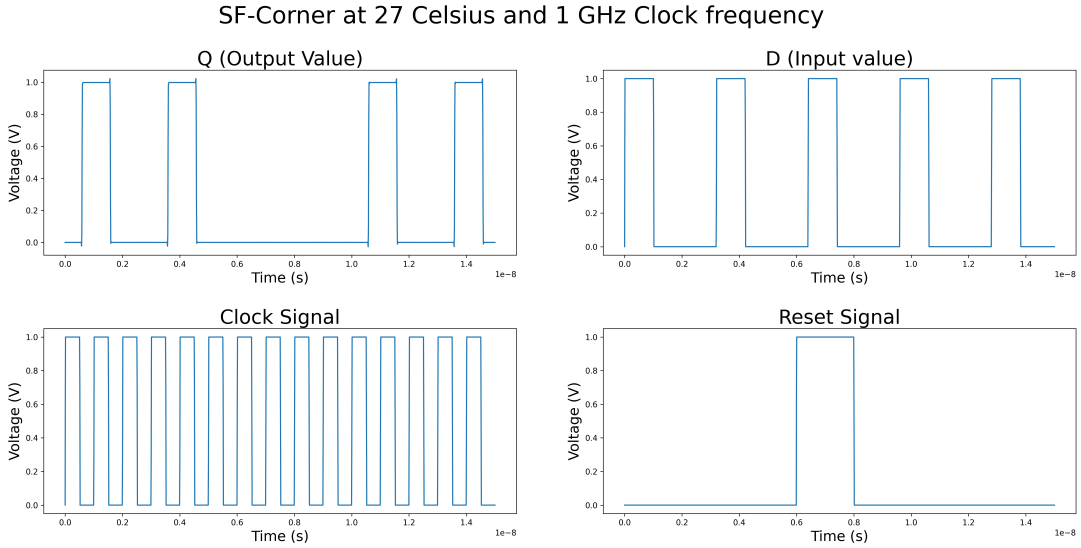

Figure 19: Functionality test for SF-corner at 27.

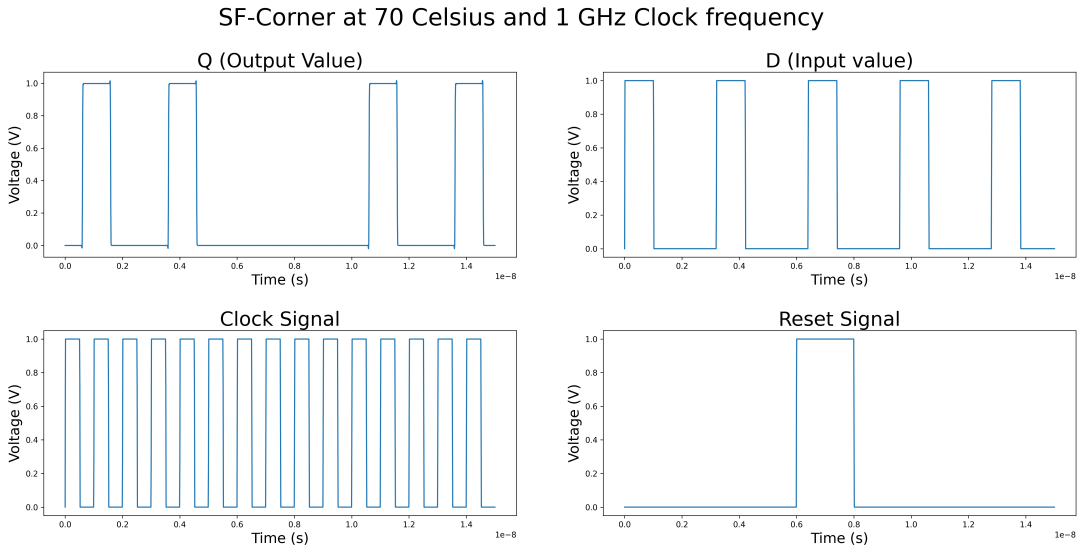

Figure 20: Functionality test for SF-corner at 70.

4.1.5 Process parameters: FS

Figures (21), (22), and (23) display the simulations for the FS-corner. While the register operates as intended, it exhibits the same initial condition anomaly observed in the TT-, FF-, and SS-corners, occurring consistently at $t = 0$s across all simulated temperatures.

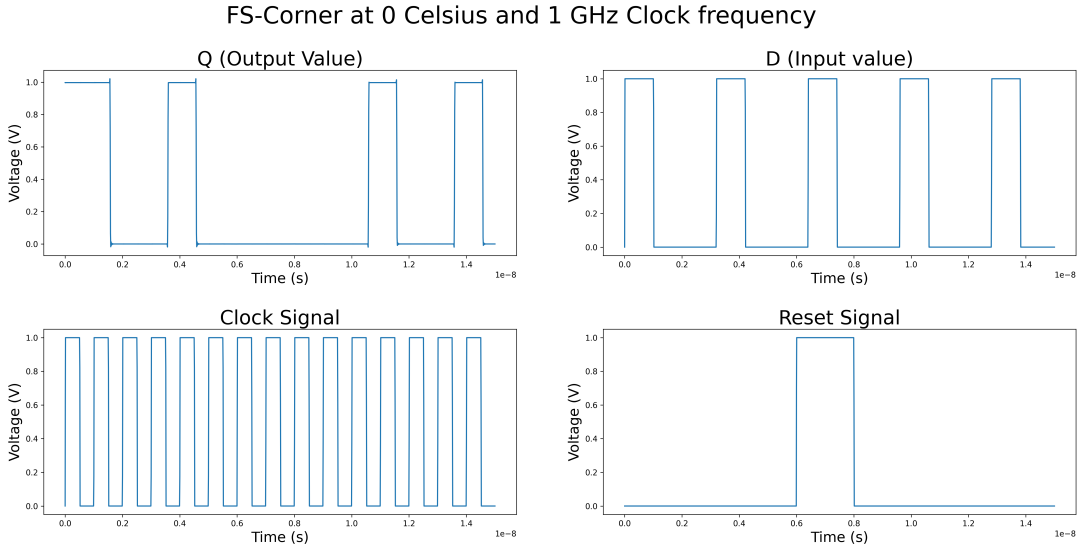

Figure 21: Functionality test for FS-corner at 0.

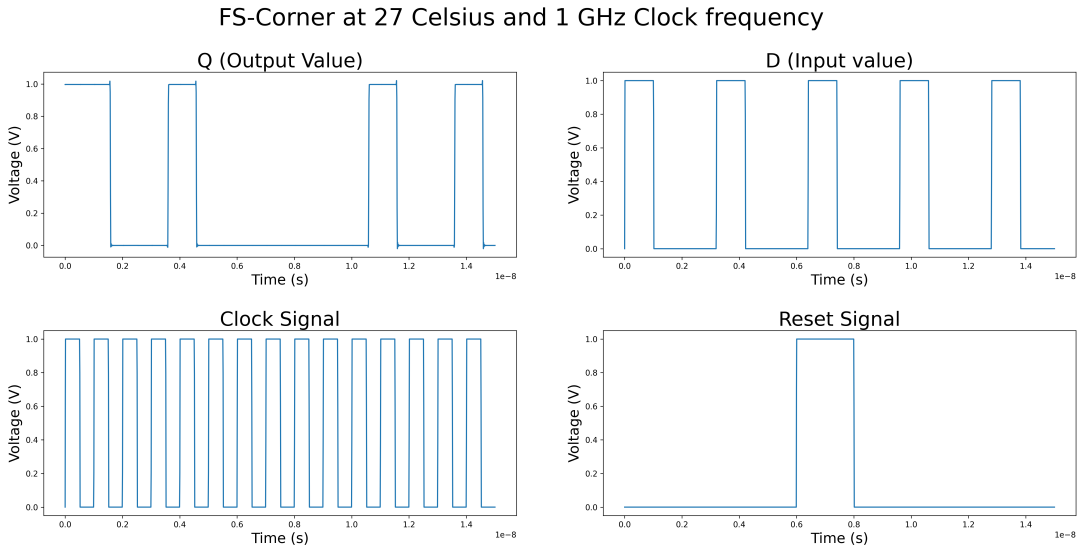

Figure 22: Functionality test for FS-corner at 27.

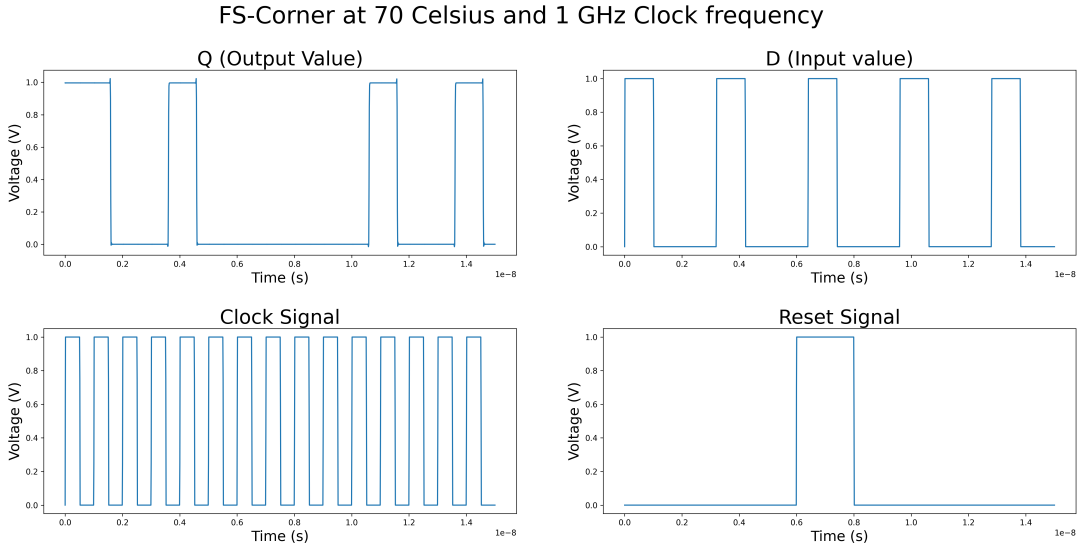

Figure 23: Functionality test for FS-corner at 70.

4.1.6 Static power consumption

Figures (25) and (24) illustrate the simulated static power consumption P_{static} for the TT- and FF-corners at 27 °C, respectively. The average static power consumption for the register, both in its on-state and off-state, as well as the average combined value, is outlined in Table (2). These values are calculated using equation (4). The data indicates that the TT-corner exhibits the lowest average static power consumption, primarily attributed to its lower power usage in the off-state. Overall, the static power consumption remains consistently low, aligning with the intended design objectives.

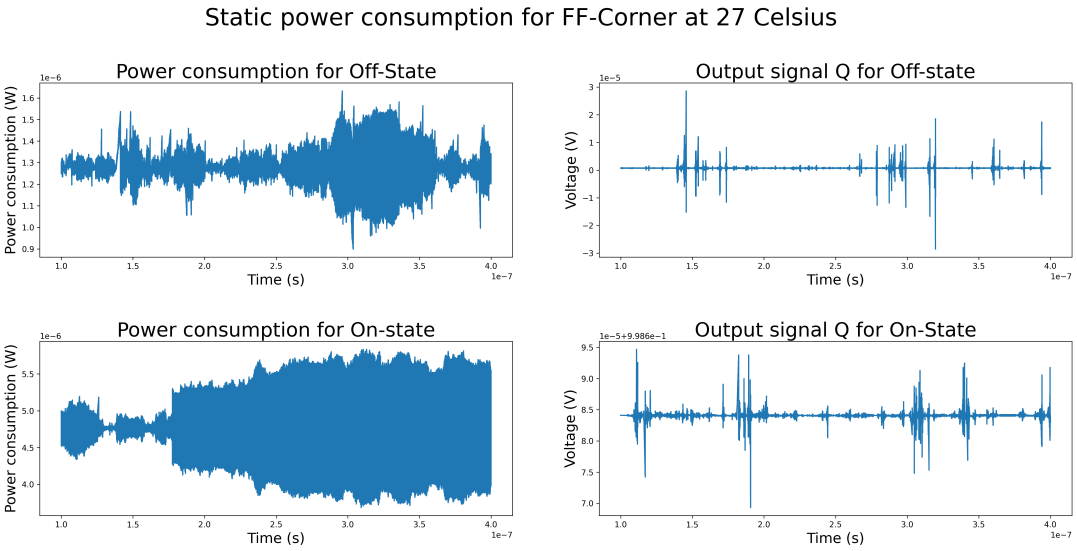

Figure 24: Static power consumption for FF-corner in On and Off states.

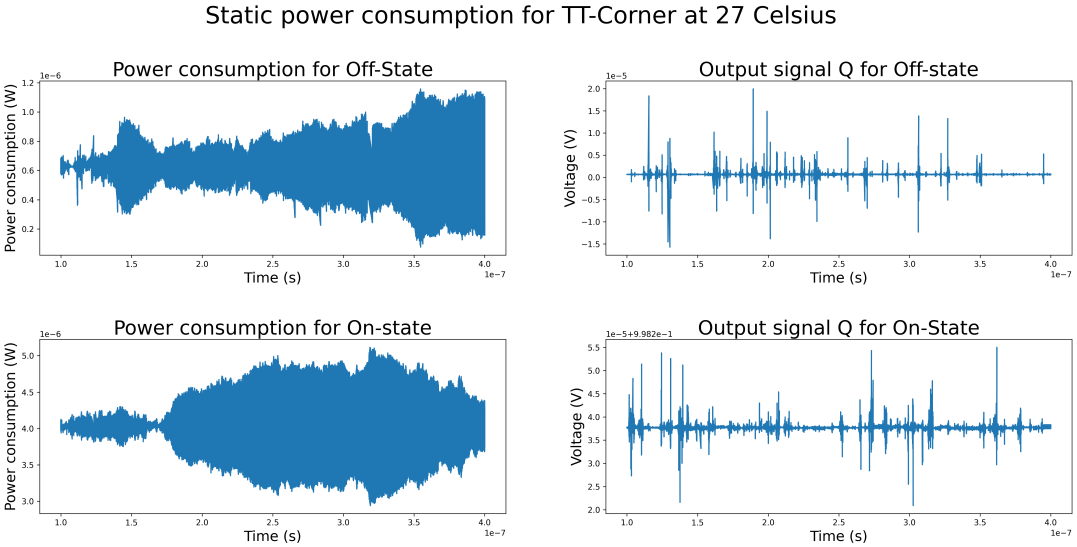

Figure 25: Static power consumption for TT-corner in On and Off states.

Table 2: Average static power consumption for TT and FF corners operating in On and Off states.

Corner	Avg. P_{static} Off-state	Avg. P_{static} On-state	Avg. P_{static} both on/off-states
TT	6.28e-07 W	4.03e-06 W	2.33e-06 W
FF	1.28e-06 W	4.77e-06 W	3.02e-06 W

4.2 Verilog Simulation of the complete system

The testing conducted on the Verilog code representation of the system (Appendix B) confirms its expected functionality. During the simulation illustrated in Figure (26), $A[1:0]$ was set to 0b'01, while $B[1:0]$ was configured as 0b'11. In Figure (27) both $A[1:0]$ and $B[1:0]$ was set to 0b'11.

As depicted in Figure (26), the output of the registers increments by 3 for each negative edge of the clock cycle, persisting for three consecutive cycles, followed by one cycle of idleness before resuming this pattern. Notably, at 40 ns, the active reset signal triggers the resetting of both $Y[7:0]$ and the FSM, initiating a new sequence of three calculation cycles and one idle cycle. Additionally, instances at 22 ns and 46 ns showcase the MAC-unit remaining idle due to an inactive run-signal until its reactivation.

The simulation depicted in Figure (27) exhibits a parallel behavior, albeit with an increment of 0b'101 (equivalent to decimal 9) for $Y[7:0]$ during each calculation cycle, as expected.

The simulated behavior affirms that the system operates according to its design specifications.

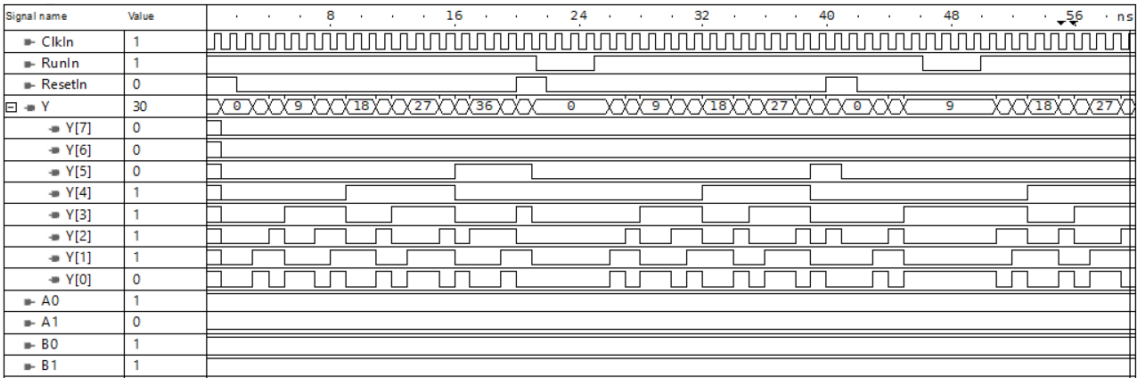

Figure 26: Result of the Verilog simulation of the complete system, including both the FSM and the MAC-unit, with $A[1:0]$ = 0b'01, and $B[1:0]$ = 0b'11.

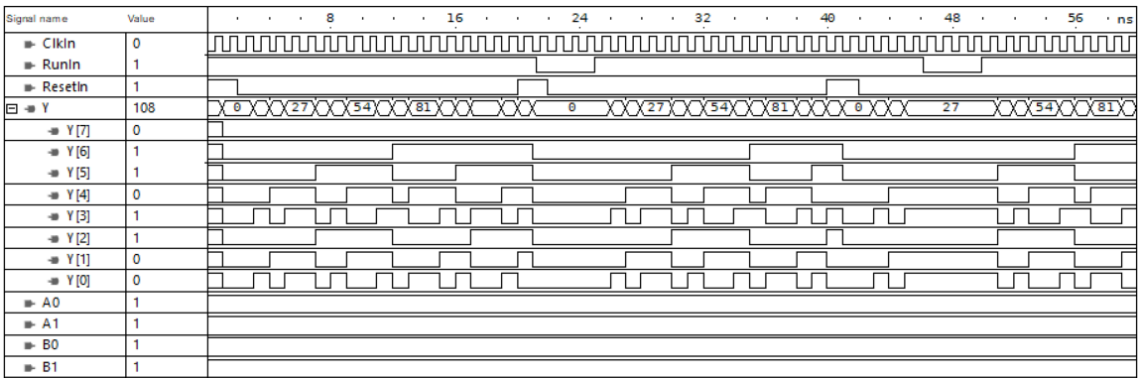

Figure 27: Result of the Verilog simulation of the complete system, including both the FSM and the MAC-unit, with $A[1:0]$ = 0b'11, and $B[1:0]$ = 0b'11.

5 Discussion

5.1 Verilog-simulation

The simulation outcomes detailed in section (4.2) validate the system's expected functionality, displaying no anomalous behavior.

5.2 SPICE-simulation

The figures presented in section 4.1 demonstrate that the 1-bit register operates as intended. At the negative edge of the clock signal, the output aligns with the input, maintaining this state until the subsequent negative clock edge unless a high reset signal intervenes. In such cases, the output value at the subsequent negative clock edge registers as low, irrespective of the preceding stored value. This functionality remains consistent across various process parameters and operational temperatures, ensuring a stable and distinctly defined high or low output signal.

However, a notable anomaly emerges concerning the initial output value, Q, across different simulations. The expected behavior dictates a low initial output until the first negative clock edge. Yet, inconsistencies arise based on transistor process corners and temperature. The discrepancy seems related to AIMSPICE's handling of initial conditions. The initial 0V value of the clock signal, perceived as a negative edge in certain simulations, leads to varied results. Even minor changes in transient analysis step size produce inconsistent outcomes regarding this anomaly. Nonetheless, the behavior following the initial clock cycle consistently aligns, suggesting this discrepancy might not significantly impact the register's functionality if physically implemented, although it introduces uncertainty.

The static power consumption of the register has been demonstrated to be minimal, approximately in the microwatt range. Potential enhancements could involve refining the supply voltage, further simplify the circuit topology, and addressing parasitic capacitance more meticulously within the circuit.

6 Conclusion

In conclusion, the project has successfully demonstrated the intended functionality of the designed 8-bit multiply-accumulate unit controlled by a finite state machine. Both Verilog-simulations and SPICE analyses have provided valuable insights into the system's behavior and functionality. The Verilog simulations affirmed the expected system operation without displaying any abnormal behavior, providing confidence in the unit's performance if physically implemented.

However, the SPICE simulations uncovered an anomaly concerning the initial output value (Q) of the register across various conditions, suspected to be linked to AIMSPICE's treatment of initial conditions. This inconsistency challenges the certainty of the register's operation during the initial clock cycle in a physical implementation. Despite this, the consistent behavior observed after the first clock cycle suggests that this anomaly might not impact the unit's overall functionality if physically implemented.

Additionally, the examination of the static power consumption highlighted the register's low power demands, an encouraging aspect that could be further optimized through fine-tuning the supply voltage and mitigating parasitic capacitance within the circuit.

In essence, while the project has showcased a robust design for the multiply-accumulate unit, further investigation and refinement in addressing the initial output anomaly is required.

Bibliography

[1] Steve Arar. *What Is VHDL? Getting Started with Hardware Description Language for Digital Circuit Design - Technical Articles*. Dec. 2017. URL: https://www.allaboutcircuits.com/technical-articles/hardware-description-langauge-getting-started-vhdl-digital-circuit-design/.

[2] Tony Chan. Carusone et al. *Analog integrated circuit design*. 2nd. John Wiley & Sons, 2014.

[3] DIGILENT. *Verilog HDL Background and History*. URL: https://reference.digilentinc.com/learn/fundamentals/digital-logic/verilog-hdl-background-and-history/start.

[4] M. Morris Mano and Michael D. Ciletti. *Digital design: with an introduction to the verilog hdl*. 5th. Pearson Education Limited, 2013.

[5] Laurence w. Nagel. *Is It Time for SPICE4?* URL: https://web.archive.org/web/20060926034314/http://www.cs.sandia.gov/nacdm/talks/Nagal_Larry_NACDM2004.pdf.

[6] Adel S. Sedra and Kenneth C. Smith. *Microelectronic circuits*. Oxford University Press, 2016.

[7] Arar Steve. *Getting Started with the Verilog Hardware Description Language - Technical Articles*. Jan. 2019. URL: https://www.allaboutcircuits.com/technical-articles/getting-started-with-the-verilog-hardware-description-language/.

Appendix A: SPICE-Code

```
1-BIT REGISTER WITH RESET SIGNAL

.include gpdk90nm_fs.cir

*********************************************************************
************** PARAMETERS ***************************************

.param  ln=350n
.param  lp=350n
.param  wn=700n
.param  wp=1500n

.param  supply=1V

*********************************************************************
*                          TRANSMISSION GATE                        *
*********************************************************************
.subckt TransmissionGate In PmosGate NmosGate Out VDD VSS

*       Drain:  Gate:       Source: Bulk:   Type:   Length: Width:
-------|-------|-----------|-------|-------|-------|-------|-------
xP1     Out     PmosGate    In      VDD     pmos1v  l=lp    w=wp
xN1     In      NmosGate    Out     VSS     nmos1v  l=ln    w=wn

.ends

*********************************************************************
*                              NOT-GATE                             *
*********************************************************************

*       Name:   Input:  Output: Power supply+:  Power supply-:
.subckt NOTGate A       B       VDD             VSS

*       Drain:  Gate:   Source: Bulk:   Type:   Length: Width:
-------|-------|-------|-------|-------|-------|-------|--------
xmp1    B       A       VDD     VDD     pmos1v  l=lp    w=wp
xmn1    B       A       VSS     VSS     nmos1v  l=ln    w=wn

.ends

*********************************************************************
*                              AND-GATE                             *
*********************************************************************
*       Name:       Input 1:    Input 2:    Output: Power+: Power-:
.subckt ANDGate     A           B           C       VDD     VSS

xTG1    A Nb B C VDD VSS TransmissionGate

*       Input   Output  Power+  Power-  Type
-------|-------|-------|-------|-------|-----------
xNotB B   Nb    VDD   VSS   NOT

*       Drain:  Gate:   Source: Bulk:   Type:   Length: Width:
-------|-------|-------|-------|-------|-------|-------|---------
xQ5     VSS     Nb      C       C       nmos1v  l=ln    w=wn
.ends
```

```
57
58
59 ********************************************************************
60 *       NEGATIVE EDGE TRIGGERED D-FLIP-FLOP WITH RESET SIGNAL      *
61 ********************************************************************
62 .subckt DFF   D Clk Reset Q VDD VSS
63
64 xNres    Reset   Nres    VDD   VSS   NOT
65 xRes     Nres    D       D   VDD     VSS     AND
66
67 xTG1     Dr  Nclk    Clk     1   VDD VSS     TransmissionGate
68 xTG2     4   Clk     Nclk    1   VDD VSS     TransmissionGate
69 xTG3     2   Clk     Nclk    5   VDD VSS     TransmissionGate
70 xTG4     8   Nclk    Clk     5   VDD VSS     TransmissionGate
71
72 xNot1    1   2       VDD     VSS     NOT
73 xNot2    2   4       VDD     VSS     NOT
74 xNot3    5   Q       VDD     VSS     NOT
75 xNot4    Q   8       VDD     VSS     NOT
76 xNClk    Clk Nclk    VDD     VSS     NOT
77
78 .ends
79
80
81 ********************************************************************
82 ************** SIMULATION ******************************************
83
84 V1       VDD     0   dc  sup
85
86 VClk     Clk 0   ac  PULSE(0 0 0     0 0 0.5ns   7s)
87 VData    D   0   ac  PULSE(0 0 0     0 0 2ns     3.2s)
88 VRes     Res 0   ac  PULSE(0 0 5.99n 0 0 2n      100s)
89
90 xDFF     D   Clk Res Qn  VDD 0   DFF
91 xNot2    Qn  Qq  VDD 0           NOT
92 xNot3    Qq  Q   VDD 0           NOT
93 R1       Q   0   300k
```

Appendix B: Verilog-Code

```
// 2x2 Bit Multiplicator

module MultiplicationUnit(input wire A1, A0, B1, B0,
                          output wire T3, T2, T1, T0);

    and                 G1(P1, B1, A1);
    and                 G2(P2, B0, A1);
    and                 G3(P3, B1, A0);
    and                 G4(P4, P3, P2);
    and                 G5(T3, P1, P4);
    xor                 G6(T2, P1, P4);
    xor                 G7(T1, P3, P2);
    and                 G8(T0, B0, A0);

endmodule

// 8-Bit Adder

module Adder(input wire Z0, Z1, Z2, Z3, Z4, Z5, Z6, Z7, X0, X1, X2, X3,
             output wire Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7);

    and                 G1(P1, Z0, X0);
    xor                 G2(Y0, Z0, X0);
    and                 G3(P3, Z1, X1);
    xor                 G4(P4, Z1, X1);
    and                 G9(P9, P1, P4);
    xor                 G10(Y1, P4, P1);
    or                  G11(Rest1, P3, P9);
    and                 G5(P5, Z2, X2);
    xor                 G6(P6, Z2, X2);
    and                 G12(P12, Rest1, P6);
    xor                 G13(Y2, Rest1, P6);
    or                  G14(Rest2, P12, P5);
    and                 G7(P7, Z3, X3);
    xor                 G8(P8, Z3, X3);
    and                 G14a(P14, Rest2, P8);
    xor                 G15(Y3, Rest2, P8);
    or                  G16(Rest3, P14, P7);
    and                 G17(P17, Z4, Rest3);
    xor                 G18(Y4, Z4, Rest3);
    and                 G19(P19, P17, Z5);
    xor                 G20(Y5, Z5, P17);
    and                 G21(P21, Z6, P19);
    xor                 G22(Y6,  Z6, P19);
    xor                 G23(Y7, Z7, P21);

endmodule
```

```
// Negative Edge triggered D-Flip-Flop

module negDFF(input wire Clk, Dn, Reset, output wire Q);

    not         Ga(Nres, Reset);
    and         Gb(D, Nres, Dn);

    not         G0(Nclk, Clk);
    not         G1(Nd, D);
    nand        G2(y1, D, Clk);
    nand        G3(y2, Nd, Clk);
    nand        G4(y3, y4, y1);
    nand        G5(y4, y3, y2);
    nand        G6(y5, y3, Nclk);
    nand        G7(y6, y4, Nclk);
    nand        G8(Q, y5, y7);
    nand        G9(y7, Q, y6);

endmodule

// Positive Edge Triggered D-Flip-Flop

module posDFF(input wire Clk, Dn, Reset, output wire Q);

    not         Ga(Nres, Reset);
    and         Gb(D, Nres, Dn);

    not         G0(Nclk, Clk);
    not         G1(Nd, D);
    nand        G2(y1, D, Nclk);
    nand        G3(y2, Nd, Nclk);
    nand        G4(y3, y4, y1);
    nand        G5(y4, y3, y2);
    nand        G6(y5, y3, Clk);
    nand        G7(y6, y4, Clk);
    nand        G8(Q, y5, y7);
    nand        G9(y7, Q, y6);

endmodule

// Multiply-Accumulate Unit

module MAC(input wire clk, Reset, A0, A1, B0, B1,
           output wire Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7);

    MultiplicationUnit      U1(A1, A0, B1, B0, X3, X2, X1, X0);
    Adder                   U2(Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7,
                               X0, X1, X2, X3, Z0, Z1, Z2, Z3,
                               Z4, Z5, Z6, Z7);

    negDFF  DFF0(clk, Z0, Reset, Y0);   negDFF  DFF1(clk, Z1, Reset, Y1);
    negDFF  DFF2(clk, Z2, Reset, Y2);   negDFF  DFF3(clk, Z3, Reset, Y3);
    negDFF  DFF4(clk, Z4, Reset, Y4);   negDFF  DFF5(clk, Z5, Reset, Y5);
    negDFF  DFF6(clk, Z6, Reset, Y6);   negDFF  DFF7(clk, Z7, Reset, Y7);

endmodule
```

```
// Finite State Machine

module FSM (input Run, Reset, Clk, output wire clkOut, resetOut);

    wire A;
    wire B;
    wire DA;
    wire ClkInternal;
    wire DB;

    assign  clkOut      = !( ( (A & Run)|(B & Run)|(Reset) ) & !Clk );
    assign  ClkInternal = ( ( Run|Reset ) & Clk );
    assign  resetOut    = Reset&Clk;
    assign  DA          = ( (!A & B & !Reset)|(A & !B & !Reset) );
    assign  DB          = ( !B & !Reset );

    posDFF  pDFF1(ClkInternal, DA, Reset, A);
    posDFF  pDFF2(ClkInternal, DB, Reset, B);

endmodule


// Complete System

module FullCircuit(input ClkIn, RunIn, ResetIn, A0, A1, B0, B1, output [7:0] Y);

    FSM     FSM1(RunIn, ResetIn, ClkIn, ClkOut, ResetOut);
    MAC     MAC1(ClkOut, ResetOut, A0, A1, B0, B1, y0, y1, y2, y3, y4, y5, y6, y7);

    assign  Y = {y7, y6, y5, y4, y3, y2, y1, y0};

endmodule
```