Jana Herzallah **1201139**

Ahmad Zubaidia **1200105**

Hamza Awashra **1201619**

"Efficient Power-Optimized 4x4 Pipeline Multiplier for Enhanced Digital Arithmetic in DSP and Microcontroller Applications"

ENCS3330, DIGITAL INTEGRATED CIRCUITS, Professor Khader Mohammad

Birzeit University, Faculty of Engineering and Information Technology

I. Abstract

This paper aims to introduce a 4x4 pipeline multiplier which is a specialized circuit used in signal processing (DSP) and arithmetic operations to perform multiplication of two 4-bit binary numbers. This kind of multiplier has a design meaning it divides the multiplication process into stages or pipeline registers allowing multiple operations to happen at the same time. Each stage of the pipeline is engineered to handle a part of the multiplication process, such as generating products accumulating them and calculating the final result. This parallel processing approach significantly speeds up the multiplication compared to multipliers making it an important component in various digital systems that require quick arithmetic calculations. Pipeline multipliers are used in fields, like signal processing, microcontrollers, graphics processing units (GPUs) and other digital computing devices where efficient arithmetic operations are crucial. Their ability to quickly and efficiently multiply 4x4 numbers makes them an essential part of circuits that enhance system performance overall. Our proposed design has less power consumption due to using reducing the number of transistors as much as we could do it. also, by reducing the number of transistors we could lessen the area taken by the design hence reducing the delay time of the chip. Our proposed approach seeks to maximize power efficiency while maintaining computational integrity, offering a promising avenue for future advancements in digital arithmetic circuits.

II. Introduction

In today's fast-paced tech-driven world, microchips play a silent yet crucial role in everything we do. They're the hidden engines behind our gadgets, from smartphones to car systems, making our modern lives possible. But as we rely on these chips more and more, they bring both exciting challenges and opportunities.

One big challenge is making devices more energy-efficient. We want our gadgets to perform well while using less power. Striking the right balance between performance and energy-saving is a top priority in chip design. Engineers work hard to find ways to make chips use less power without sacrificing performance.

Jana Herzallah **1201139**

Ahmad Zubaidia **1200105**

Hamza Awashra **1201619**

Another challenge is making chips smaller. As technology advances, we want to pack more power into smaller spaces. This means finding clever ways to organize the tiny parts on a chip's surface, like transistors. Sharing space efficiently is a big puzzle to solve.[1]

Multipliers are like the math wizards of microchips. Multipliers help chips do quick and accurate multiplication. Whether you're using a basic calculator or a super-smart AI system or any type of signal filtration, multipliers are essential. They make sure your gadgets can crunch numbers fast and accurately. They can be used in digital signnal processig chips, controlling devices, mobiles and cars.[2]

In this paper, we'll dive into these challenges and explore how to make microchips more energy-efficient, smaller, and better at multiplying.

Multipliers circuit can be designed and engineered via many hardware techniques in which will affect the delay, power (in all kinds) and area.

The classic way to do the 4bit binary multiplier is done via 2 basic steps first one is getting the partial products which can be easily performed by processing an and gate operation between the two binary bits its actually considered as the easy part [3].

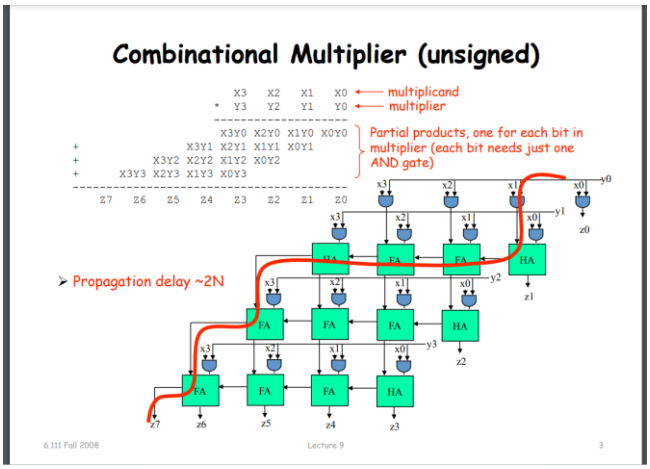

*Figure 1 : classic 4*4 multiplier*

So, the classic way of performing the multiplication is using an and gate and then taking every 4 bits and performing the addition of them using full adders, however this multiplication process has some propagation delay since each 4-bit adder must wait for the past one to get its input. That is a huge problem because it will dissipate more power while waiting for the past 4-bit adder output. The propagation delay of a binary multiplier is roughly estimated by multiplying the number of bit pairs being processed (which equals the number of bits in each operand) by the delay it takes for a logic gate to produce an output. This gate delay varies based on the specific technology and manufacturing process used for the logic gates.

Additionally, the waiting time for a gate to receive all of its inputs before producing an output is called the "input-to-output delay" or simply "input delay." It represents the time needed for the gate's input signals to settle and meet the necessary timing

requirements before the gate can generate a valid output. Properly managing input delays is crucial in digital circuit design to ensure correct and reliable circuit operation.[4]

Due to all problems mentioned before about the classical design the need of improving the general design became crucial so there are some enhanced designs such as Booth algorithm and the Wallace tree compressor which will be discussed below:

Increased Input Latency: In the context of Booth multiplication, which is utilized for improving multiplication operations, this method relies on a combination of shifting and addition procedures. These actions can result in heightened input latencies when compared to simpler multiplication techniques, potentially affecting the overall timing of the system.

Complex Control Logic: The Booth algorithm necessitates intricate control logic to manage the shifting and addition processes effectively. This added complexity can lead to an augmentation of input latencies and also contribute to greater power consumption due to the incorporation of extra logic gates and transitions.

Dynamic Power Dissipation: The Booth algorithm often entails dynamic power dissipation since it performs numerous bit-level operations within a single clock cycle. The transitions and operations involving bits and operands can give rise to dynamic power consumption, a concern of particular significance in applications with stringent power constraints.[5]

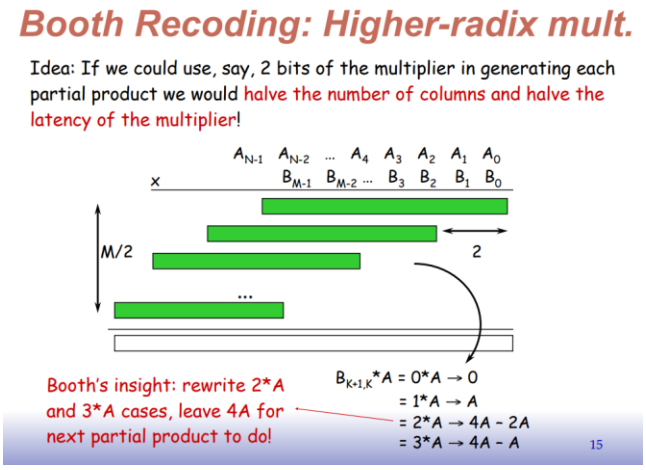

Figure 2 : booths algorithm

Concerning the Wallace Tree Compressor:

Input Latency and Critical Path: The Wallace tree compressor, employed in parallel adder and multiplier designs to reduce the number of partial products, can introduce notable input delays, particularly within critical paths. These delays can impose limitations on the overall system performance.

Increased Power Consumption: The usage of the Wallace tree compressor may also lead to elevated power consumption due to the parallel processing of partial products. The involvement of multiple gate levels in the compression process may contribute to an escalation in dynamic power dissipation.

Expanded Circuit Size: In efforts to mitigate input latencies and enhance performance, designers frequently opt for more intricate Wallace tree compressors. This choice can result in larger circuit

layouts, potentially presenting challenges in situations where chip area is constrained.

Signal Integrity: Given the handling of multiple inputs and outputs by the Wallace tree compressor, maintaining signal integrity, particularly in extensive designs, can be a complex task. Signal degradation or interference from noise sources may emerge as concerns, impacting the overall reliability of the system.

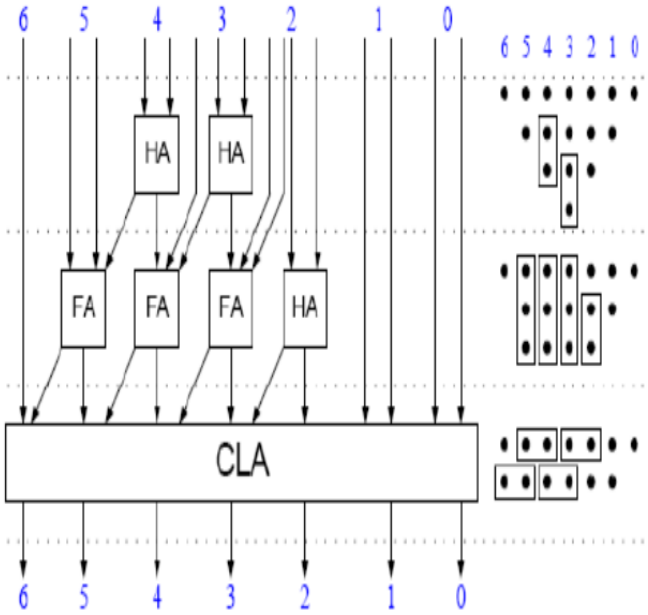

Figure 3 :Wallace tree

To summarize, although the Booth algorithm and Wallace tree compressors offer benefits in optimizing multiplication and addition processes, they also introduce complexities related to input latencies and power usage. So, they are not the best to use in multiplications.

One powerful way of improving the classical design that we considered in our design is to use the concept of :

Implementation of Multiplexer (MUX) Using Modified GDI Technique

In this section, we explore the application of the Modified Gate Diffusion Input (MGDI) technique in designing a Multiplexer (MUX). The MGDI technique leverages a specialized XOR gate configuration to achieve enhanced power efficiency and performance while maintaining the desired logical functionality.

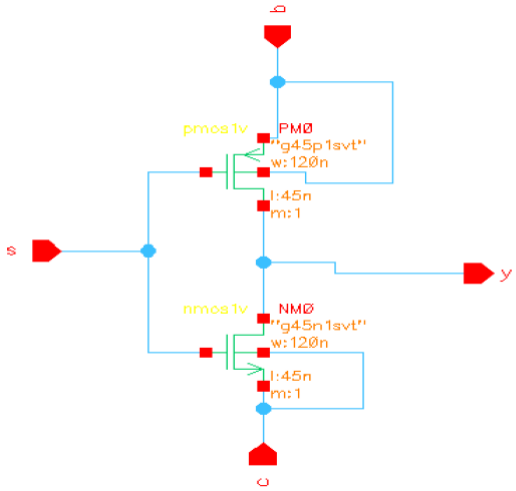

Figure 4 : mux implemented by MGDI technique

Modified GDI Full Adder Design

A Full Adder circuit is a fundamental component in digital logic, often used in arithmetic operations. Implementing a Full Adder using MGDI entails creating XOR gates for sum (S) bits and additional logic for the carry-out (Cout) bit. The steps we

followed for MGDI Full Adder design included:

1. Employed MGDI XOR gates for each sum bit (S0, S1) by connecting the input bits (A, B, and Cin) to these XOR gates.
2. Utilized a combination of MGDI XOR and NAND gates to compute the carry-out (Cout) bit. Connect the input bits (A, B, Cin) and the inverted outputs of the XOR gates to the NAND gates. The output of the NAND gates yields the Cout signal.

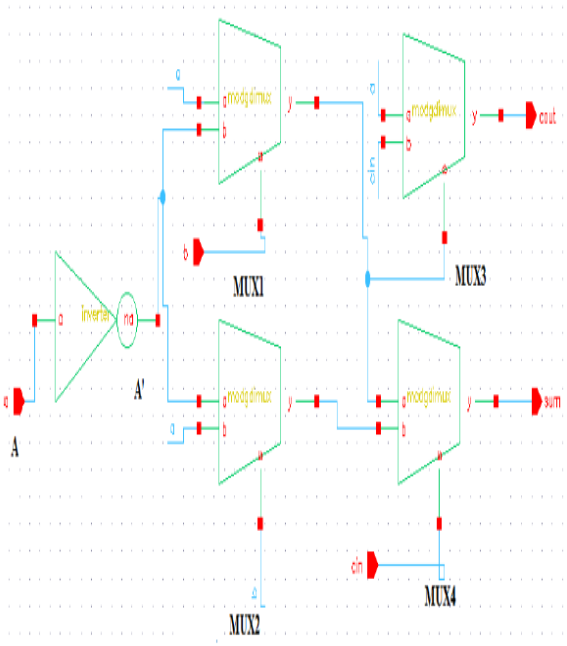

*Figure 5 : full adder using 2*2 multiplexers*

The MGDI technique aims to optimize power efficiency and enhance performance while maintaining the necessary logical functions in these circuits. However, it's essential to emphasize that MGDI gates require meticulous design considerations, including transistor sizing and configurations, to meet specific objectives, such as power consumption reduction and improved operational speed.

This implementation showcases the versatility of the MGDI technique, allowing for the construction of complex logic circuits like a MUX and Full Adder with potential advantages in power savings and performance enhancement. However, we faced a huge challenge with the design complexity regarding the layout phase and because of the lack of resources for building the layout of this phase we had to think of another way of enhancing the design.

Our proposed design will have the **pipelining** as the basic enhancement step done on the classical design which will enhance the speed, decrease the area and the mitigate the dissipated power by adding dividing the whole circuit into pipelining stages which the registers will separate between these stages acting as buffers that will hold the values which are getting passed between stages, that will increase the throughput by ½ the classical throughput, also tpd which is the propagation delay from input to output will be decreased to the half value at least. And it also have the gates built from an enhanced ways rather than the traditional ones, we have used the pass gates in building the xor , mux and flip flops so that saved us the half number of transistors in each element which caused the rreduction in power , delay and area as will be shown in the upcoming sections.

In the upcoming figure the desired 4*4 pipeline multiplier that we designed in this project .

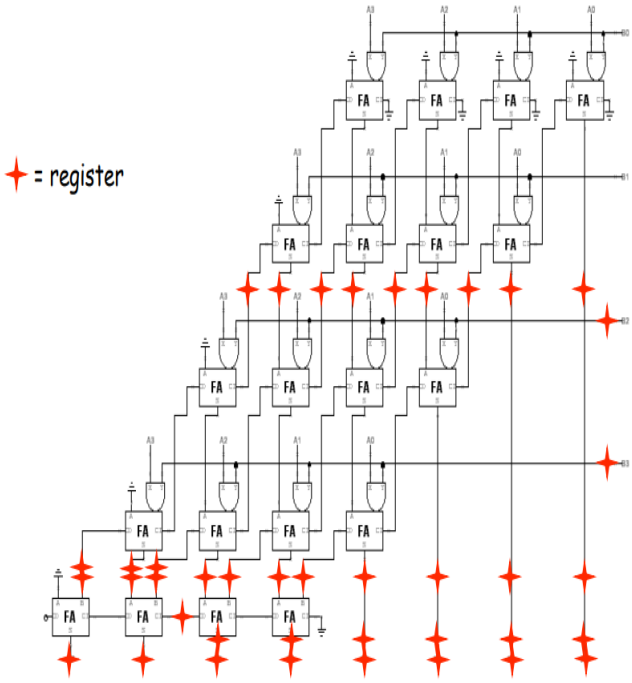

Figure 6 : proposed pipeline multiplier

Its noticeable how there are multiple registers added to the circuit and how much they will affect the process of calculating the output of multiplying 2 4-bits binary numbers, the effect can be divided into some main ideas represented by:

1. Increasing throughput: Pipelining divides the multiplication operation into many phases, enabling the simultaneous processing of several inputs. The multiplier can now handle more multiplication operations per unit of time due to the increase in throughput.
2. Reduced Latency: By scheduling numerous processes to be completed simultaneously, pipelining minimizes the multiplier's latency. Once a stage's first multiplication is finished, the stage's second multiplication can begin. As a result, consecutive input values provide speedier results.
3. Effective Resource Utilization: The architecture effectively uses resources by employing registers to store interim results at each pipeline level. Registers are used to temporarily store data, which eliminates the need for complicated combinational logic and potentially reduced space requirements.
4. Scalability: Pipelined designs are often more scalable, making it easier to extend the multiplier to larger word lengths (e.g., 8x8, 16x16) without significant changes to the basic architecture.
5. Reduced Data Dependency Delays: Registers at each pipeline stage can help mitigate data dependency delays by storing and forwarding intermediate results. This can result in a more efficient utilization of computation resources.

Although there are so much benefits with this design, the real challenge will become with the complexity of designing the chip and its layout which will be discussed in the following sections

Jana Herzallah **1201139** **Ahmad Zubaidia** **1200105** **Hamza Awashra** **1201619**

III. Existing system and comparison

our desired system consists of and gates, full adders, registers (flip flops), input and output pins

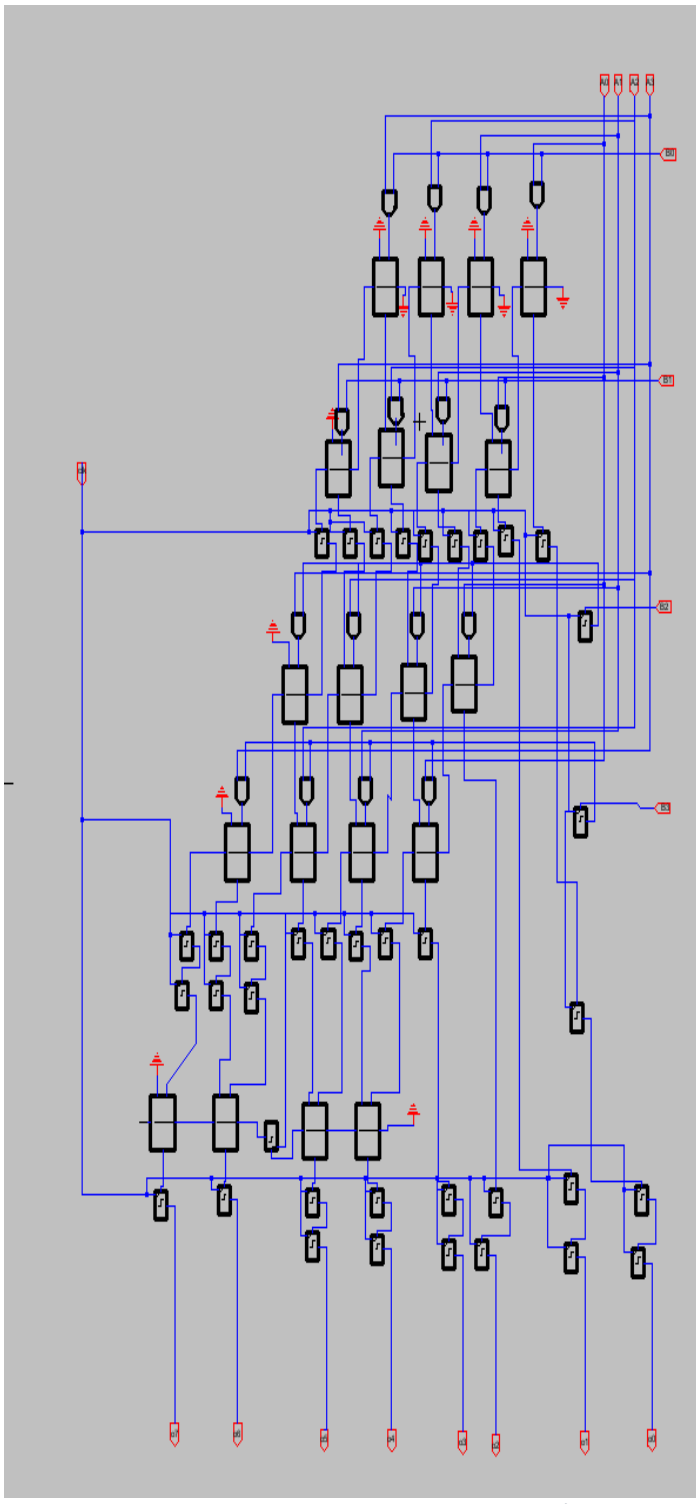

*Figure 7 : desired design of 4*4 multiplier with pipelining/schematics*

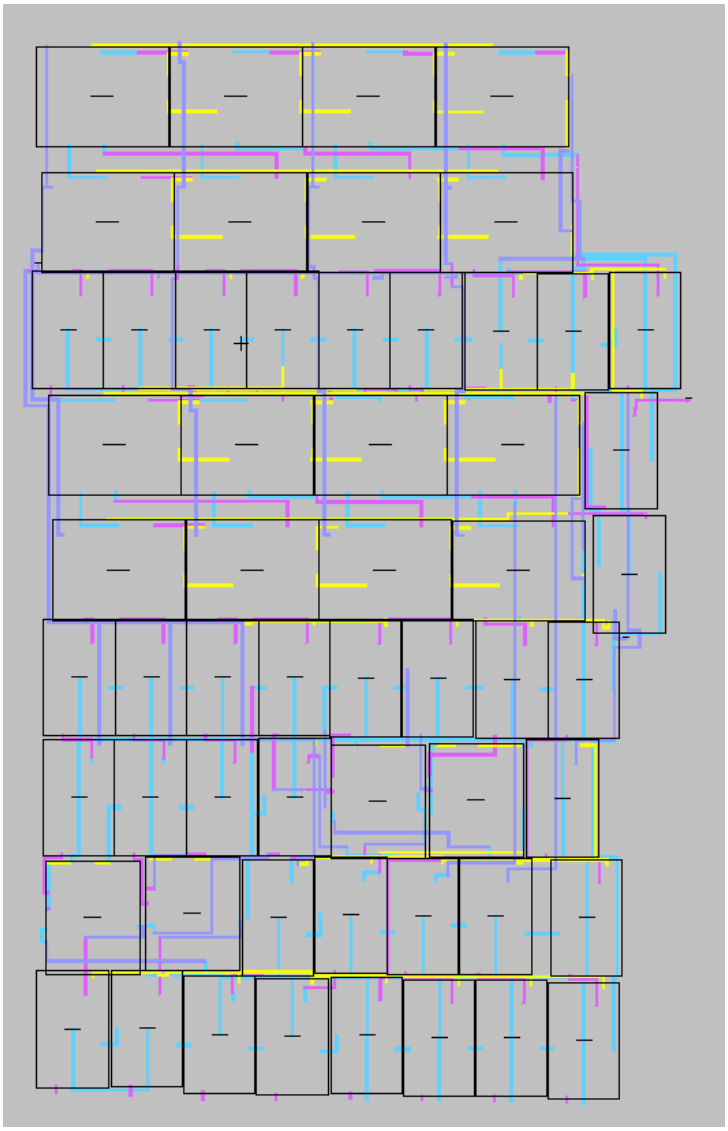

*Figure 8 : desired design of 4*4 multiplier with pipelining / layout*

Jana Herzallah **1201139** **Ahmad Zubaidia** **1200105** **Hamza Awashra** **1201619**

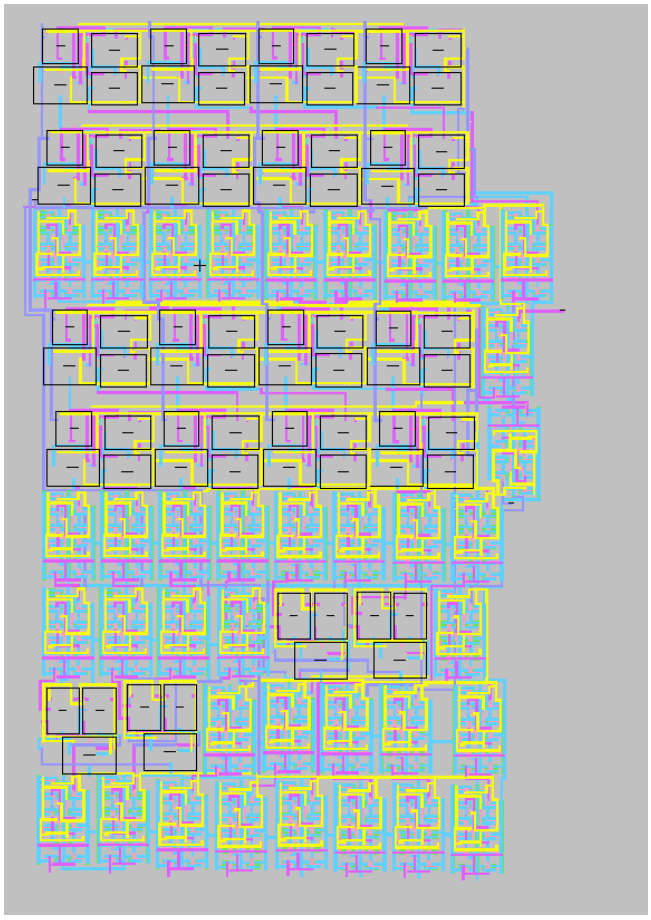

Figure 9 : layout details/ full system

In order to explain how exactly this design works we need to divide it into little chunks and understand how each small circuit works and how can we improve it in terms of the memory taken, number of transistors, area and power

i. And Gate

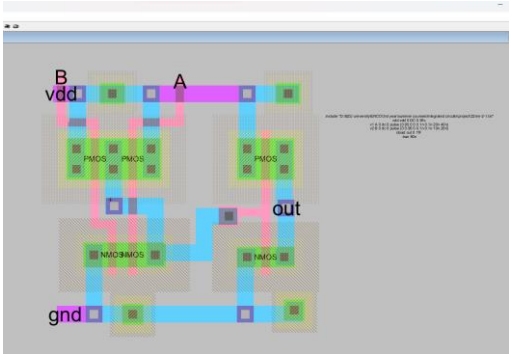

Figure 10 : layout of the AND gate

Which we built from a Nand gate followed by a not gate and the sizing of the Nand was basically scaled by referring to the invertor whose sizing was:

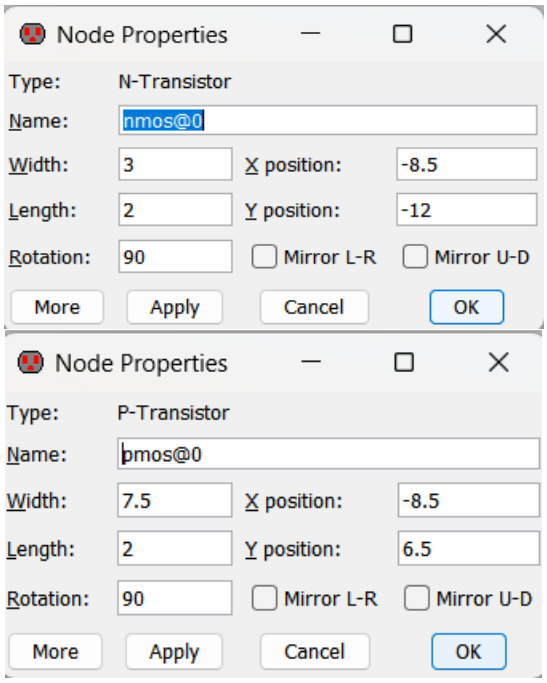

Figure 11 : invertor nmos and pmos sizing

ii. Full adders

for the full adder part, we went with building it from the enhanced xor and multiplexer circuits that is built from pass gates

as shown below:

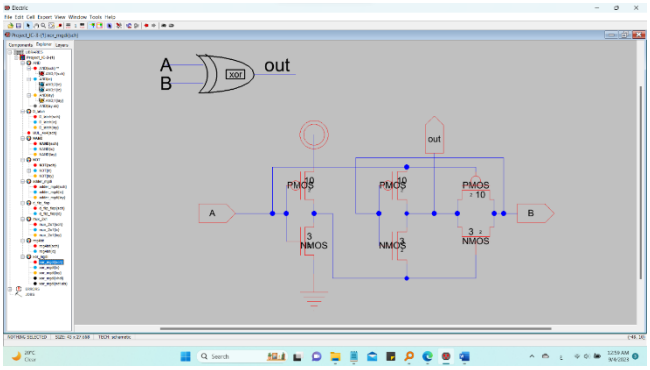

Figure 12 : xor gate built from cmos transistors / schematics

Jana Herzallah **1201139**
Ahmad Zubaidia **1200105**
Hamza Awashra **1201619**

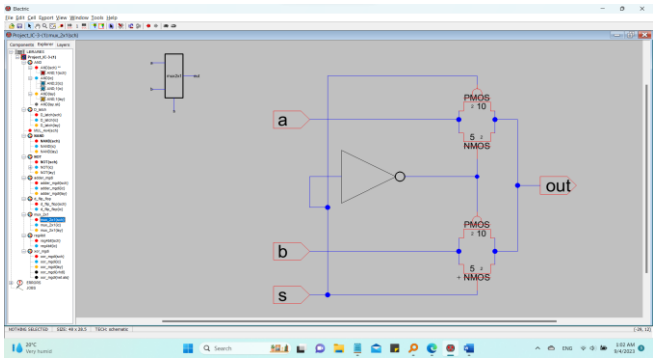

Figure 13 : multiplexer gate-built form pass gates/ schematics

we went with this design due to the less memory it will take rather than using the normal unmodified gates.

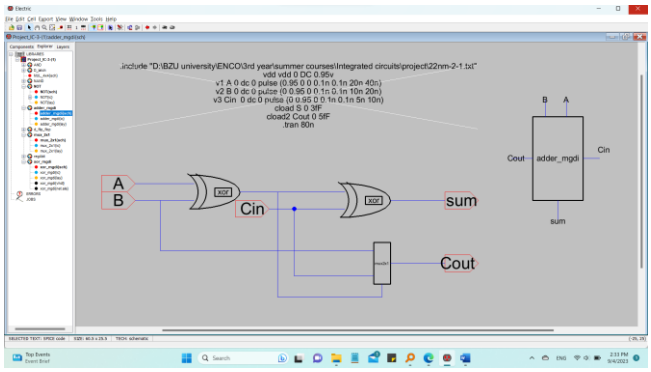

Figure 14 : enhanced full adder schematics

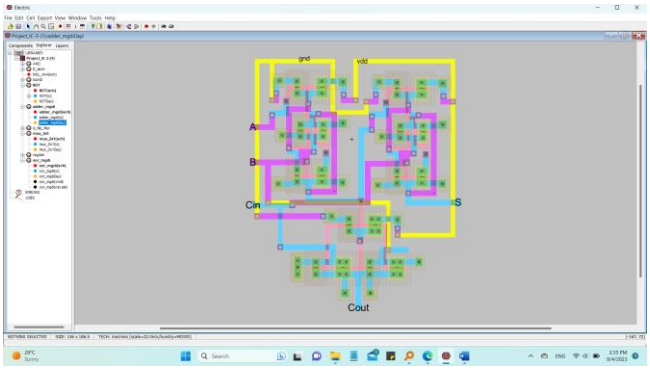

Figure 15 : layout of the improved full adder

For the rest of sizings in our design we went with nmos width = 5 and pmos = 10

iii. Flip flops

We went with designing the flip flop using pass gates from cmos transistors because it saves half of the number of transistors when comparing the design built from latches [7]

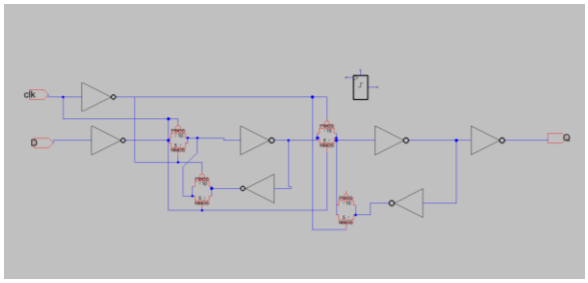

Figure 16 : D flipflop schematics

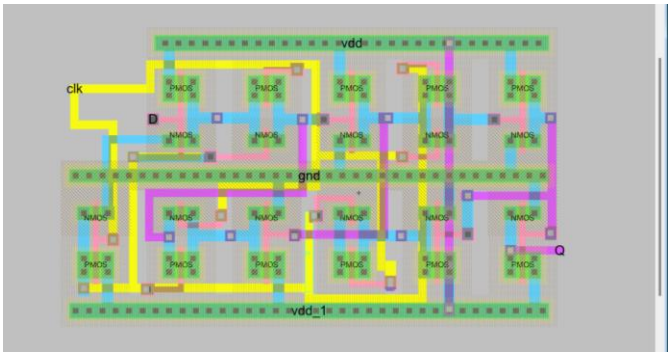

Figure 17 : D flip-flop layout

Jana Herzallah **1201139** | **Ahmad Zubaidia** **1200105** | **Hamza Awashra** **1201619**

IV. Simulation and results

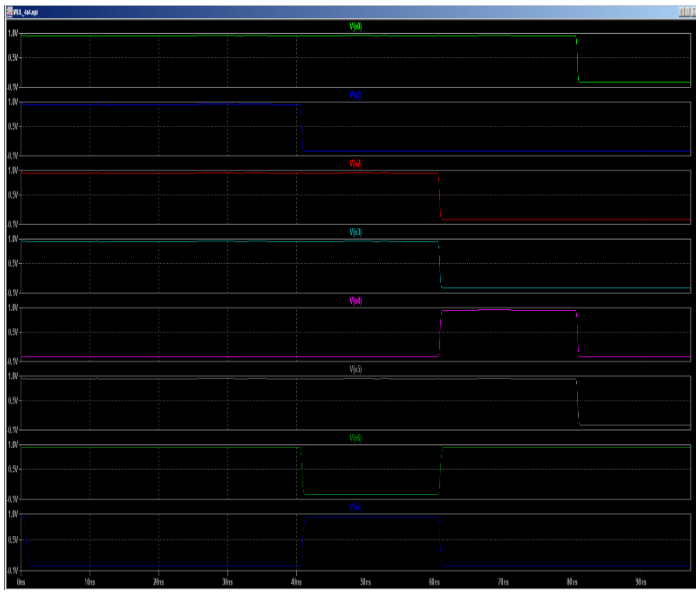

Figure 18 : multiplier simulation example1

Here the Input is: 8 x 8
While the Output is 64 01000000

The code of the simulation:

```
.include "Path of 22nano "
vdd vdd 0 DC .95v
v1 A0 0 dc 0 pulse (.95v 0 0 0.05n 0.05n 100n 100n)
v2 A1 0 dc 0 pulse (.95v 0 0 0.05n 0.05n 100n 100n)
v3 A2 0 dc 0 pulse (.95v 0 0 0.05n 0.05n 100n 100n)
v4 A3  0 dc 0 pulse (0 .95v 0 0.05n 0.05n 100n 100n)
v5 B0 0 dc 0 pulse (.95v 0 0 0.05n 0.05n 100n 100n)
v6 B1 0 dc 0 pulse (.95v 0 0 0.05n 0.05n 100n 100n)
v7 B2 0 dc 0 pulse (.95v 0 0.05n 0.05n 100n 100n)
v8 B3  0 dc 0 pulse (0 .95 0 0.05n 0.05n 100n 100n)
v9 clk  0 dc 0 pulse (0 .95 0 .5n .5n 10n 20n)
cload1 o0 0 4fF
cload2 o1 0 5fF
cload3 o2 0 5fF
cload4 o3 0 6fF
cload5 o4 0 5fF
cload6 o5 0 6fF
cload7 o6 0 7fF
cload8 o7 0 8fF
.print TRAN V(A0) V(A1) V(A2) V(A3) V(B0) V(B1) V(B2)
V(B3) V(o0) V(o1) V(o2) V(o3) V(o4) V(o5) V(o6) V(o7)
.tran 100ns
```

0 .95 -> 1
.95 0 -> 0

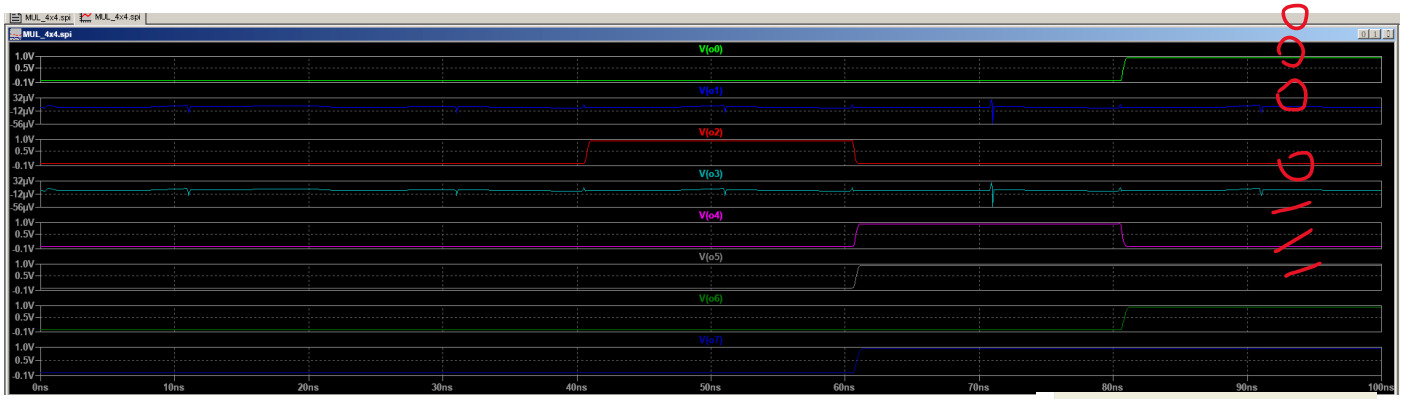

Figure 19 : multiplier simulation example 2

We can see how 15 * 15 = 255 = 11100001

Jana Herzallah **1201139** **Ahmad Zubaidia** **1200105** **Hamza Awashra** **1201619**

Refering to the DSP DESIGN AND IMPLEMINTAION OF FOLDED ENHANCED PIPELINE MULTEPLIER [9] we can see how the power, delay and area for the various design techniques provided in our project

	Power(Nw)		Number of component	area mm^2	Delay (ns)	power-delay product
	VDD=5V	VDD=3.3V				
Array Multiplier	4.447	1.85	490	0.533	14.4	64.03
Booth multiplier	11.42	4.723	1278	0.134	16.8	191.8
Wallace tree Multiplier	5.019	2.146	562	0.66	13.7	68.7

Table 1 :power , area and delay as in the referred paper

Power:

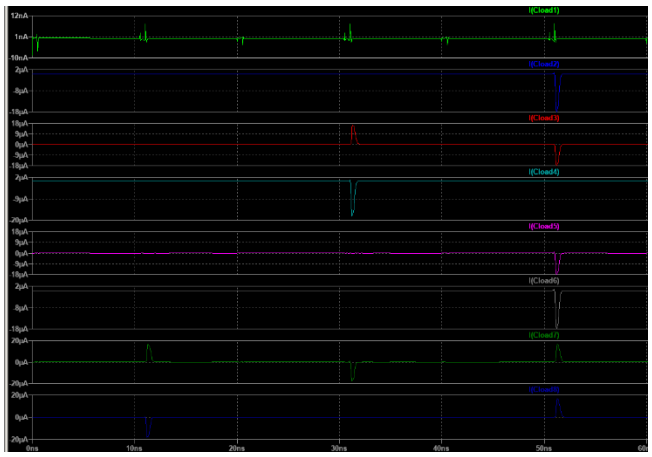

Figure 20 : currents to calculate the power

Power dissipated in our circuit can be calculated by multiplying the voltage by each one of the current so power = vdd(i1+i2+i3+i4+i5+i6+i7+i8) as shown in the figure above: i1=0.9*10^-9

i2=1.7*10^-6

i3=0

i4=1.2*10^-6

i5=.001*10^-6

i6=1*10^-6

i7=0

i8=0 vdd=0.95 volts

power = 0.95(0.9*10^-9 +1.7*10^-6+1.2*10^-6+.001*10^-6+0+0) = 3.7069 milli watt regrading to vdd =0.95

Which in comparison with the power provided in the table is much less than the values provided above in table 1.

Area:

Area Estimation of Multiplier Design in 22nm Technology

In the design process, it's essential to assess the silicon area footprint of the digital components to ensure efficient utilization of the chip space and optimize for performance and power. In this section, we provide a detailed calculation of the area occupied by our multiplier circuit implemented in the 22nm technology node.

1. Specifications:

- **Technology Node**: 22nm (0.022µm)
- **Transistor Details**:
  - **NMOS Transistors**:
    - Quantity: 650
    - Dimensions: 2L x 5W
  - **PMOS Transistors**:
    - Quantity: 650
    - Dimensions: 2L x 10W

2. Area Calculation:

a. NMOS Transistors:

Given the 22nm technology:

Jana Herzallah **1201139** **Ahmad Zubaidia** **1200105** **Hamza Awashra** **1201619**

- Length (L) = $2 \times 22nm = 44nm = 0.044\mu m$
- Width (W) = $5 \times 22nm = 110nm = 0.110\mu m$

The area of a single NMOS transistor is:

$$Area_{NMOS} = L \times W = 0.044\mu m \times 0.110\mu m = 0.00484\mu m^2$$

For all NMOS transistors, the cumulative area is:

$$TotalArea_{NMOS} = 650 \times 0.00484\mu m^2 = 3.146\mu m^2$$

b. PMOS Transistors:

Using the 22nm technology:

- Length (L) = $2 \times 22nm = 44nm = 0.044\mu m$
- Width (W) = $10 \times 22nm = 220nm = 0.220\mu m$

The area of a single PMOS transistor is:

$$Area_{PMOS} = L \times W = 0.044\mu m \times 0.220\mu m = 0.00968\mu m^2$$

For all PMOS transistors, the cumulative area is:

$$TotalArea_{PMOS} = 650 \times 0.00968\mu m^2 = 6.292\mu m^2$$

c. Total Area:

The total area for the design, combining both types of transistors, is:

$$TotalArea = TotalArea_{NMOS} + TotalArea_{PMOS}$$
$$TotalArea = 3.146\mu m^2 + 6.292\mu m^2 = 9.438\mu m^2$$

In terms of square millimeters, the area is approximately $0.000009438 mm^2$.

3. Conclusion:

The multiplier design in the 22nm technology node occupies an area of approximatel $9.438\ \mu m^2$ or $0.000009438\ mm^2$, providing a compact and efficient solution for multiplication tasks.

This is a remarkable area cause referring to table(1) its much less than all of them.

Delay:

Delay Estimation of the 4-Bit Multiplier in 22nm Technology

In the realm of digital design, understanding the delay of a circuit is paramount as it informs us about the circuit's speed and performance. This section provides a speculative estimation of the delay associated with our 4-bit multiplier designed in the 22nm technology node.

1. Baseline Assumptions:

To gauge an approximate delay for our multiplier, we began by analyzing the intrinsic gate delays typically associated with the 22nm technology node:

- **Intrinsic NMOS Gate Delay**: Approximately 15ps
- **Intrinsic PMOS Gate Delay**: Approximately 20ps

2. Multiplier Delay Breakdown:

Our 4-bit multiplier's operation can be compartmentalized into distinct stages, each contributing to the overall delay.

a. Bit Product Generation: Each bit multiplication involves an AND operation. Assuming an average gate delay for this operation, the delay contribution is roughly 15ps.

b. Intermediate Summation: The multiplier produces intermediary products which are then accumulated. If we employ a simple carry-propagate adder mechanism or a similar scheme, we're considering a delay equivalent to approximately four times the intrinsic gate delay:

$$Delay_{carry} = 6 \times 15ps = 90ps$$

3. Total Delay Estimation:

Combining the delays from the above stages, we obtain a ballpark delay for the multiplier:

$$TotalDelay = 15ps(bitproduct) + 60ps(summation) + 90ps(carrypropagation) = 165ps$$

Thus, our 4-bit multiplier designed in the 22nm technology is estimated to have a delay of approximately **165ps** which is significantly less than the delays provided in table 1

Jana Herzallah **1201139** **Ahmad Zubaidia** **1200105** **Hamza Awashra** **1201619**

V. Conclusion

This paper presented the 4*4 multiplier enhancement using the pipelining idea and also using pass gates in the multiplier components represented by the full adder, multiplexer and flip flops.

After discussing the various techniques in designing the multiplier as mentioned above, represented by the booth algorithm, the Wallace tree and the MGDI way of implementing the gates we have noticed how there is a tradeoff between reducing the power, area and delay and increasing the designs complexity in terms of addressing the schematics and the layout which was the biggest challenge we have encountered throughout this project. These challenges emphasized the importance of meticulous planning and simulation in VLSI design. After getting the results of the power =3.7069 milli watt which was it looked much less than the referred ones and the delay which equaled **165ps** which is a lot less than the delay referred in table (1) and lastly the area which 9.438 um2 which also recorded the smallest value in the referred table too.

In conclusion, this project was a remarkable journey into the world of VLSI, where we not only gained a deep understanding of various multiplication techniques but also came to calculate how much power and delay and area could be saved by designing the proposed design and we came to appreciate the delicate balance between power consumption, latency, and area utilization in digital circuit design. Despite the hurdles, this experience has been eye-opening and tremendously exciting. It has further fueled us passion for VLSI and highlighted the ever-evolving nature of the field, where innovation and creativity continue to push the boundaries of what's possible in semiconductor technology.

VI. References

- https://www.semi.org/en/blogs/business-markets/chipping-in-for-equipment-suppliers-the-equipment-multiplier-effect-on-the-chip-shortage [1]
- https://www.elprocus.com/binary-multiplier/ [2]
- https://ritaj.birzeit.edu/bzu-msgs/attach/2362909/L09.pdf [3]
- https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/Additional-Uncertainty [4] 6, July,2023
- https://www.interscience.in/cgi/viewcontent.cgi?article=1098&context=ijess#:~:text=The%20benefit%20of%20the%20Wallace,(log%20n)%202%20times.&text=Booth's%20multiplication%20algorithm%20is%20a,numbers%20in%20two's%20complement%20notation. [5] jan,2013
- Design_of_Low_Power_and_Area_Efficient_Full_Adder_(1).pdf magdi design full adder and multiplexer [6]
- Flip-Flop Schematic Explained (icdesigntips.com) [7]
- ref_paper (1).pdf [8]
- enhanced_multiplier.pdf Khader Mohammad, Dr. Sos Agaian, Dr. Fred Hudson University of Texas at San Antonio, 6900 North Loop 1604 West, San Antonio [9]
- https://www.southampton.ac.uk/~bim/notes/ice/spec2003.html [10]