Design of a N X N Signed Multiplier Based ON Radix-8 (3-Bit Grouping)

Department of Electrical and Computer Engineering
Birzeit University

Sewar Abu Eid [1], *Maha Mali* [2], *Lama Nasser* [3]

1200043@*student.birzeit.edu*[1], 1200746@*student.birzeit.edu*[2], 1200190@*student.birzeit.edu*[3]

Abstract

The central purpose of this work is to design a Multiplier Based on radix-8 (3-bit grouping) structure Design. this design presents a reduction in the total amount of the partial products. This work has been done by 32-nm process specifications. This Optimizes the performance of the radix-8 multiplier. the crucial feature for this project includes the implementation of muxes, adders and shifters using pass gate, low power consumption, decrease the latency and the hardware space.

1. Introduction

1.1. Digital Multiplier

In digital computing devices, Multipliers are greatly used after addition and subtraction. It's important to note that all the digital signal processors utilize multipliers for computationally comprehensive tasks such as filtering and convolution. In addition, high-performance devices like CPU's and GPU's use these fundamental arithmetic operations. The real reason behind the huge growth of multiplication comprehensive algorithms is the increasing demand for low-power portable devices in the markets as many market research reports show.

1.2. Motivation

All the mentioned demands derived to advancements in chip manufacturing like 10nm,7nm and 5nm. This leads to higher transistor density and decrease the power dissipation as a result. So, the objective of this work is to design a low-power, low cost and high speed signed integer multiplier with the proposed algorithm by using 32nm CMOS technology.

2. Literature Review

In general, there are two ways in order to enhance the speed of intrinsic multi-operand addition: high radix multipliers which plays a vital role in reducing the number of operands to be added, tree and array multipliers to compose multi-operand adders to reduce latency then increase the throughput.

2.1. Multiplier Architectures

2.1.1. Array Multiplier

This algorithm approach is for multiplying two binary numbers, by using an array of full and half adders in synchronous addition of the product terms. Each stage processes its corresponding partial products before reaching the adder, with the carry-out propagating to the subsequent row. In the following figure an example of this multiplier architecture:

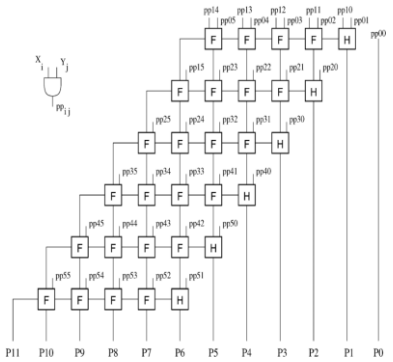


Figure 1:*Array Multiplier*

2.1.2. The Tree Multiplier Description

Tree multiplier name refers to the way that the tree multiplier is designed in a hierarchical way, and it is a type of digital circuit that is used in performing mathematical operations to perform the multiplication process for binary numbers.

The idea of tree multiplier is to divide the multiplication process into a series of parts and then add the parts as group to produce the last outcome. The structure resembles a tree, where the inputs are at the bottom like the roots of the tree and the outputs are at the top branching like the branches in the tree.

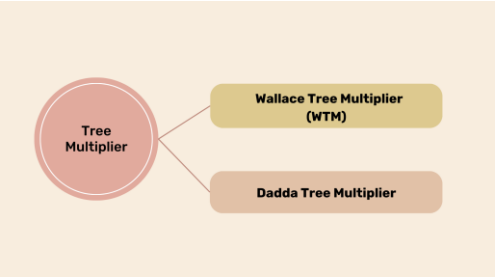


Figure 2:*Chart of Tree Multiplier*

2.1.2.1 The Wallace Tree Multiplier Description (WTM)

An important part the Wallace tree plays in high-speed applications since it is an area efficient and high-speed multiplier. Additionally, it is an actual hardware version for digital circuits that multiplies two numbers efficiently. Incomplete products are produced initially in a typical WTM. Then these are combined in different phases. This process will be repeated until there are two columns in the final stage.[2]

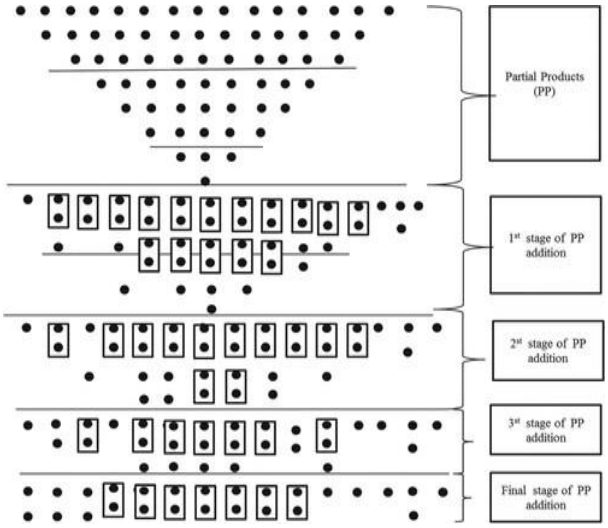


Figure 3: *The steps of Wallace Tree Multiplier*

2.1.2.2 The Dadda Tree Multiplier Description

The Dadda multiplier which use compressors to reduce part of the product, are very important in high-speed applications because they are low power and have high speed and area efficiency. Between the tree multiplier the dadda multipliers are more efficient and need less space than the Wallace tree multiplier. The dadda multiplier includes a cheaper decrease stage since it also aims to reduce the number of gates used in addition to input and output delays. [3]

2.1.3. The Booth Multiplier Description

In computer digital circuit, the booth multiplier is a technique for multiplication 2 signed binary integers. The main goal of this algorithm is to decrees the quantity for product terms via looking at neighbor bit pairs in multiplier. It is also used to increase multiplication process speed. It only requires to move all the bits that are near to the correct position since it operates on a sequence of zeros in the multiplier, as well as a series of ones in a value of 2^k to value of 2^m multiplier bits that may be seen as 2^k + 1-2^m. This procedure keeps on once its multiplier's values are all used, resulting in the final product. [4]

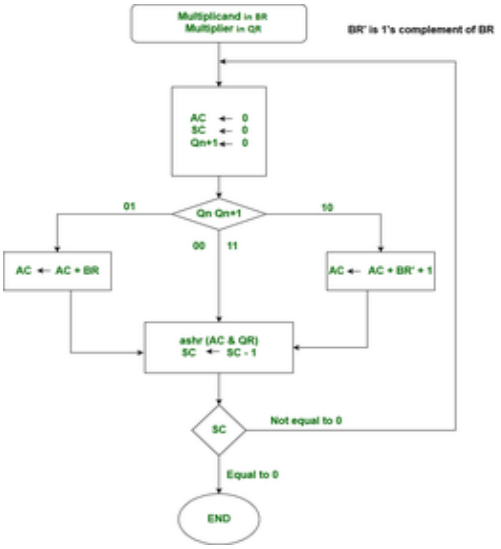


Figure 4: *Booth Multiplier*

2.1.4. Baugh Wooley Two's Complement for signed Multiplier

Baugh wooely multiplier does not need to use the sign extension method since it uses two's complement method. In order of that, it was used in the multiplication of signed numbers. Where each partial products have to be added for multiplying two integers with two's complement.

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-1}(a_i 2^i)$$

$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-1}(b_i 2^i)$$

$$P = A \times B$$

$$P = [-a_{n-1}2^{n-1} + \sum_{i=0}^{n-1}(a_i 2^i)] \times [-b_{n-1}2^{n-1} + \sum_{i=0}^{n-1}(b_i 2^i)]$$

Baugh Wooley method can be used for both signed and unsigned multiplication. Another advantage for Baugh Wooley technique is that it is a parallel multiplier. Thus, it is fast and compactible if it we compare it with other parallel multipliers. It is also having smaller zone, minimal power scattering and less duration. The baugh Wooely architecture depends on the method of carry saving calculation.

2.1.5. The Vedic Multiplier

One of the key tools for growing technology in the immense domain for image processing techniques, digital signal processing (DSP) and real time signal. A very important block in digital systems and digital design is the multipliers. As well as the accuracy domain for reducing the time required, area, power and delay of the processer by improving the speed of it. Vedic algorithms and mathematics rules produce partial products concurrently that saves the time.

2.1.6. Low Power Delay Product for Radix-4 8x8 Booth Multiplier

Booth multiplication method deals with signed and unsigned numbers in uniform way. The idea of Booth multiplier is to decrease the number of products decreasing the number of bits required. In order to achieve that, various recoding technique is used such as radix-2, radix-4 and radix-8. Aue at el 2018 proposed low power delay product multiplier that uses CMOS 90 nm technology. The low power delay can be generated due to the low delay that is reduced through using the low complexity adder. This low complexity adder is caused by the use of a modified low complexity binary to two's complement converter and multiplexer instead of conventional adder.

3. Design and Implementation

The main methodology in our work is to reduce the number of transistors in the designing process, we worked on 32 nm process file specifications. The first step to achieve this, pass gate was used instead of any other approach, in order to minimize the hardware area and have an affective optimization. Also, to guarantee precise timing and moderate signal skew.

In Order to build the radix-8-bit multiplier, then some of main functional units should be implemented, like muxes (2x1 Mux) and then (16x8 Mux) as will be shown. Then 3-bit Booth multiplier was built. To take all the outputs of these circuits as inputs of the 16-bit adder. Finally, shift the result by 2 to the right and store it in a 16-bit register to feed it back again to the adder. All these components work together in order to get the Block diagram of the 8 bit- radix 4-Booth multiplier.

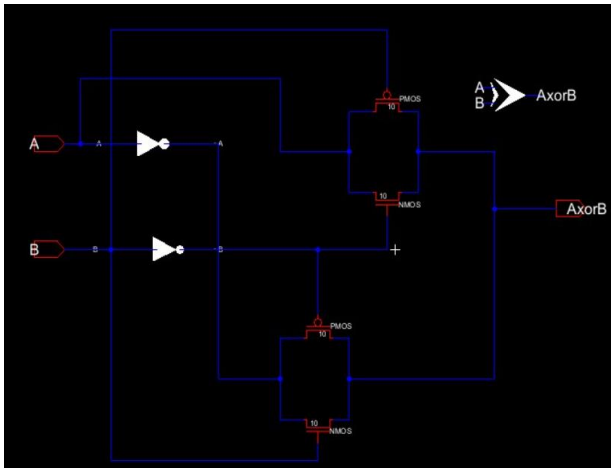


Figure 7: *Xor gate Schematic*

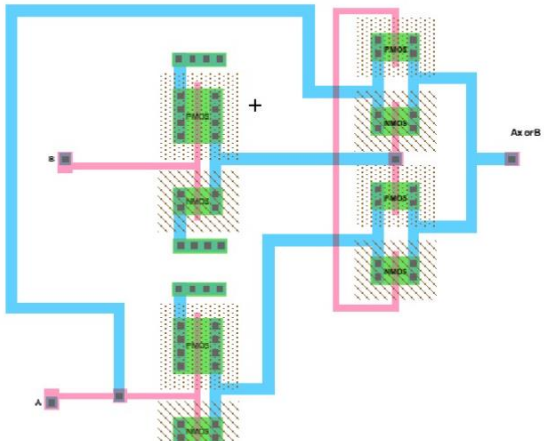


Figure 5: *Xor Gate Layout*

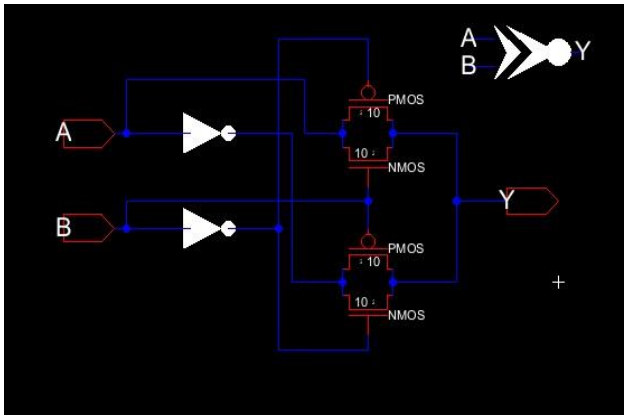


Figure 6: *Xnor gate schematic*

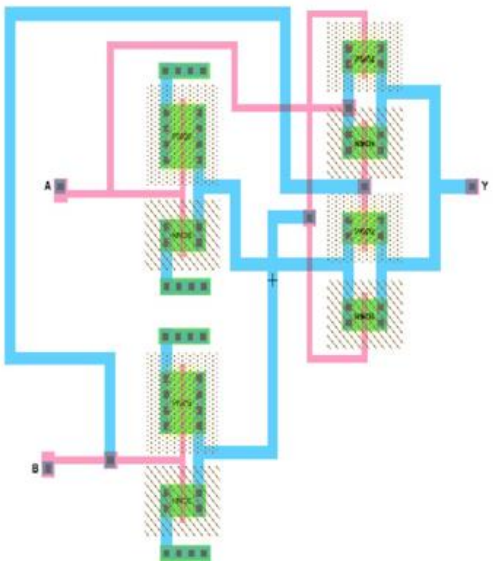


Figure 8:*Xnor gate Layout*

Now, this is the implementation of the nor gate which was used also to create the Booth Encoder.

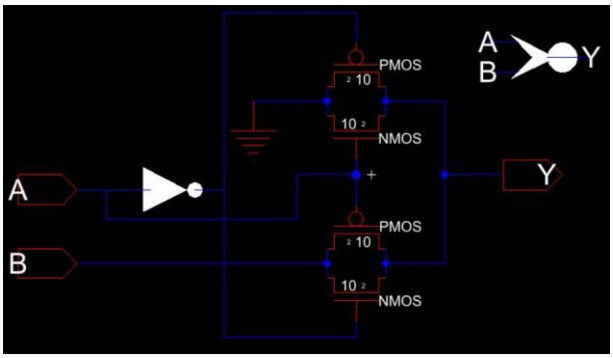


Figure 9:*Nor gate schematic*

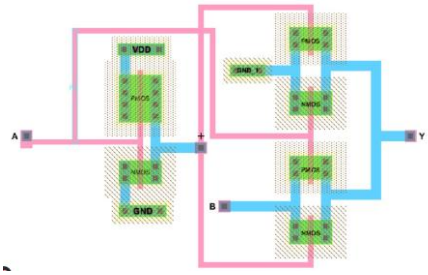


Figure 10 : *Nor gate Layout*

All these gates were be instantiating to build the Booth encoder in an area optimized design for it

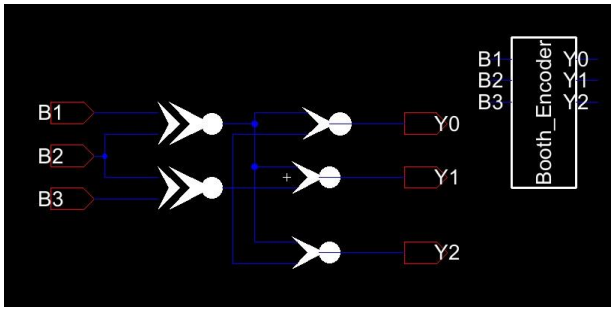


Figure 11: *3-bit Booth encoder*

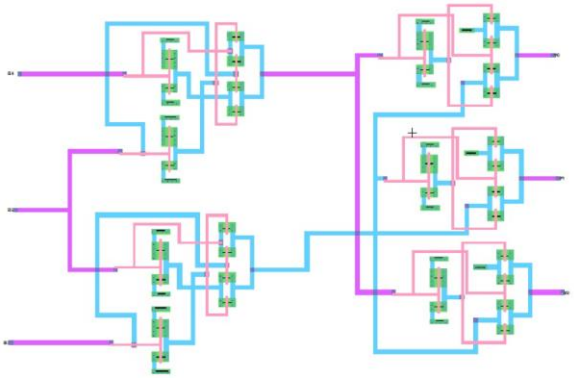


Figure 12: *3-bit Booth Encoder*

Now, the following table indicates the output from the Booth Encoder if we try to trace its functionality, we get the following results:

B1	B2	B3	Y0	Y1	Y2
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	0

Table 1:*Both Encoder Truth Table*

After Building the Booth multiplier from the mentioned logical gates that were designed by the pass gate, it's the time for new step to build the second main functional unit which is the muxes. The basic stone is to design 2x1 mux, then instantiate it for expanded design.

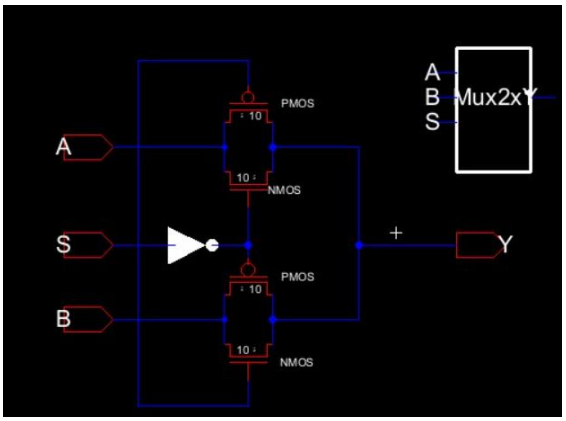


Figure 13: *mux 2x1 Schematic*

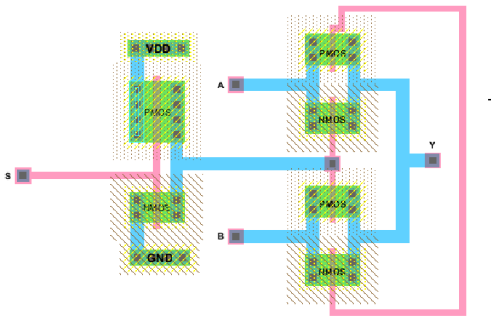


Figure 14: *mux 2x1 Layout*

This mux was used for 8 times to build the expanded 16x8 mux this will add a very strong optimization to the design, so instead of build the new mux from the beginning and use new complex designs with more devices, it was implemented as in the following circuit diagram.

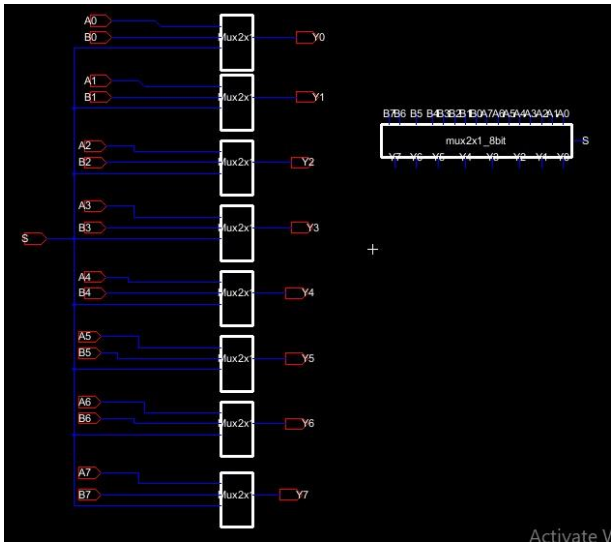


Figure 15: *mux2x1_bit Extended schematic*

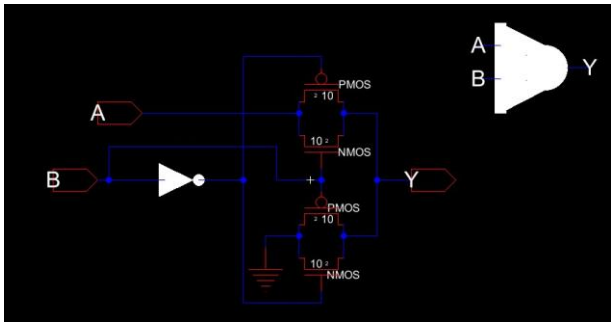


Figure 16: *AND Gate Schematic*

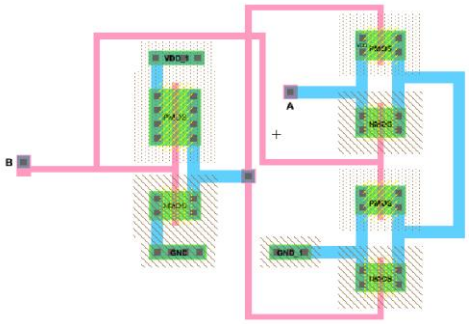


Figure 17 : *AND Gate Layout*

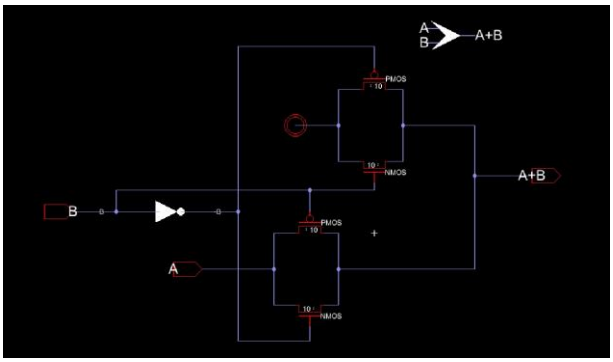


Figure 18:*OR Gate Schematic*

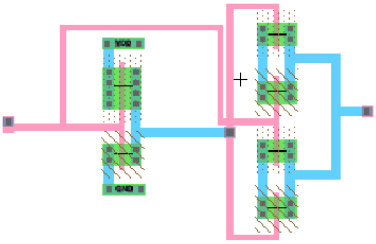


Figure 19: *OR Gate Layout*

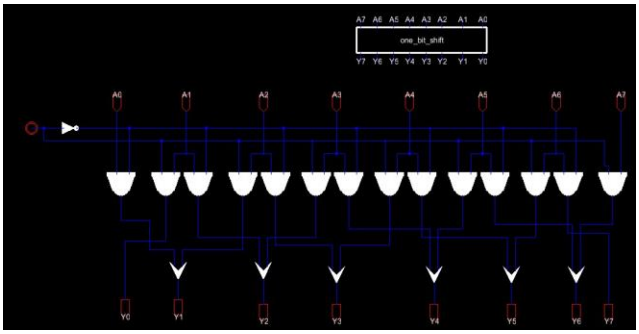


Figure 20: *One-bit left shift for 8-bit input*

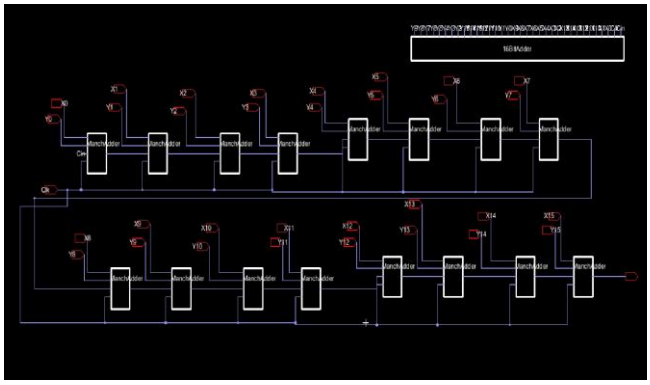


Figure 21: *16-bit Adder*

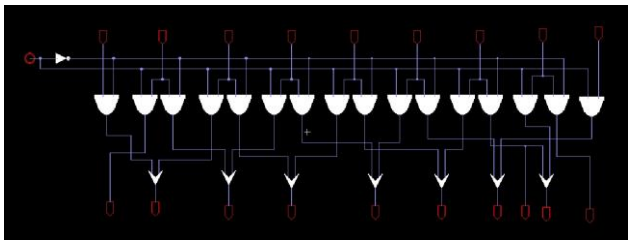


Figure 22: *One-bit left shift for 9-bit input*

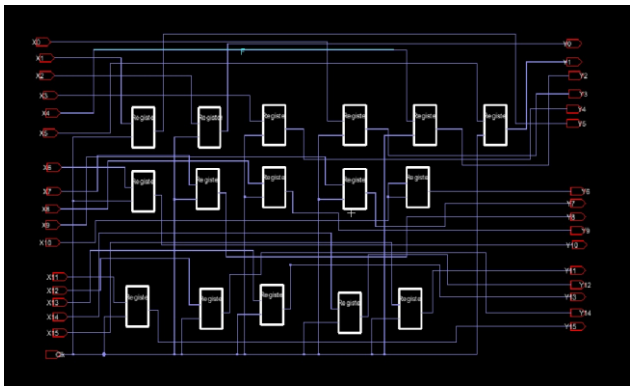


Figure 23: *16- Bit Register*

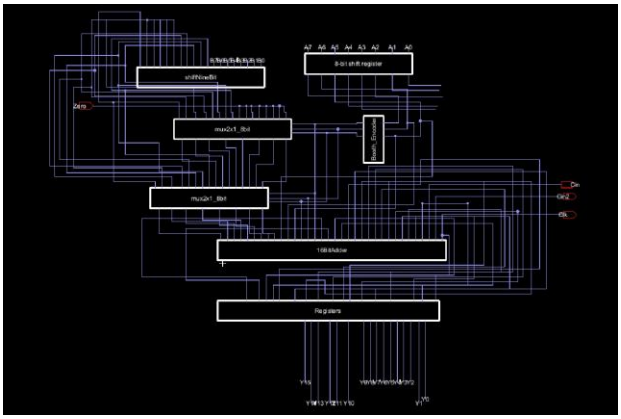


Figure 24:*Whole System*

4. Results

In our simulation we used the first and the second input equal to 240 in decimal which means 11110000 in binary. To verify that our system is working as expected, the multiplication of the two inputs 240*240=57,600 which is 1110000100000000 in binary as shown in the following figures.

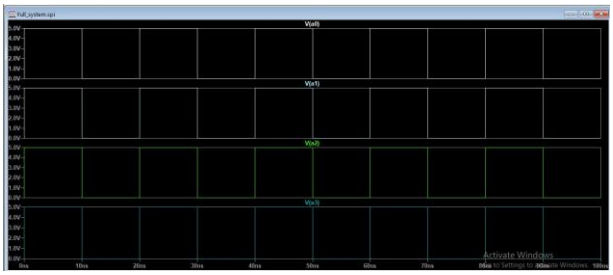


Figure 25:*Most Significant 4 bits for input A*

Figure 26: *Least Significant 4 bits for input A*

Due to the usage of the flip flops which were implemented by the latches, there are some delay issues in our simulation as the next results shown. like clock to out delay as well as to the hold and set up time. however, we tried to follow the min and max delay constraint but some issues remain with reduced effects after we did the enhancement.

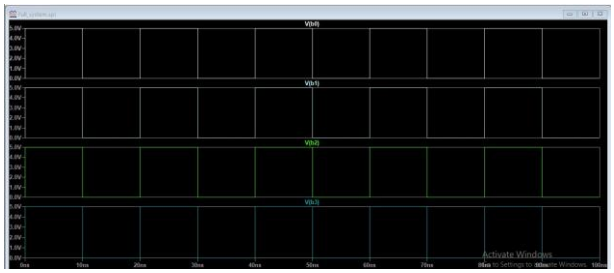


Figure 27: *Most Significant 4 bits for input B*

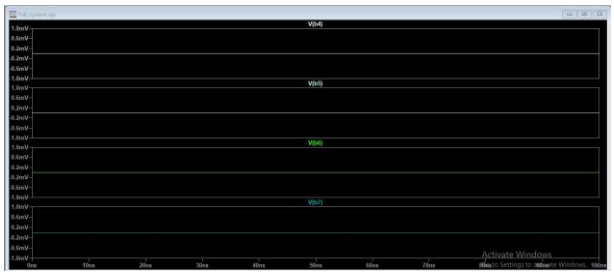


Figure 28: *Least Significant 4 bits for input B*

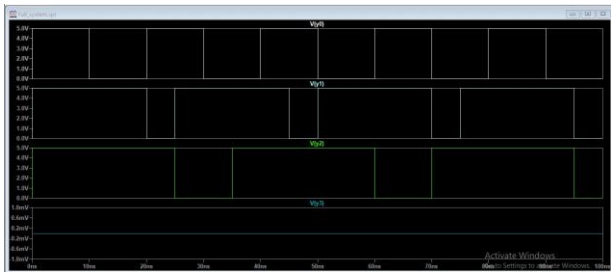


Figure 29: *Most Significant 4 bits for output Y*

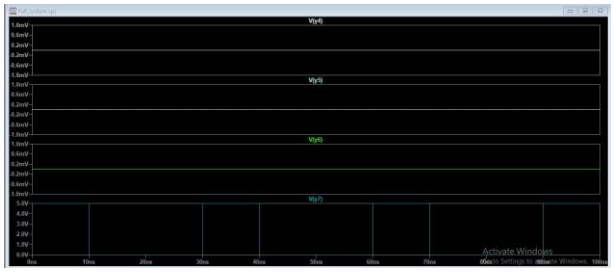


Figure 30: *The next 4 bits for output Y*

Figure 31: *The next 4 bits for output Y*

Figure 32: *Least Significant 4 bits for output Y*

5. Area, Power, and Delay Optimization

In our Designed system, we used the pass gate in the functional units like muxes, adder and shifter due to the area optimization and power consumption compared to CMOS gates. it achieves this by maximize the non-critical paths in order to have low-activity regions of the multiplier. In addition, the switching time of transmission gate is too faster than other type of CMOS gates as mentioned.

Any suggestion for future enhancements of the 8-radix multiplier should focus on parallel processing. Also, the flexibility is one of the most important improving factors which leads to optimize the power consumption. as well as the importance of the crucial role that the pipelined algorithms play in order to improve the operations concurrently. this will require adding registers between each pipeline stages in order to save the inner values.

6. Design Improvements

Power simulation was performed deeply were static power due to leakage current increases significantly. The proposed multiplier consumes the power from 0.013mW to 0.006mW and reduced the hardware space from 0.13µm to be 0.09µm technologies, respectively, so it reduces power consumption to 0.07%~0.08% of active mode.

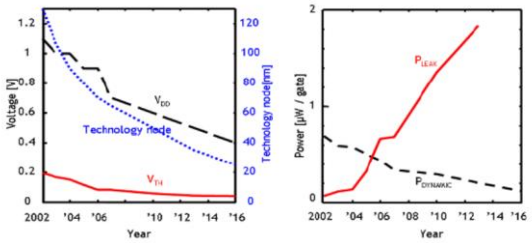


Figure 33:*Power vs Voltage*

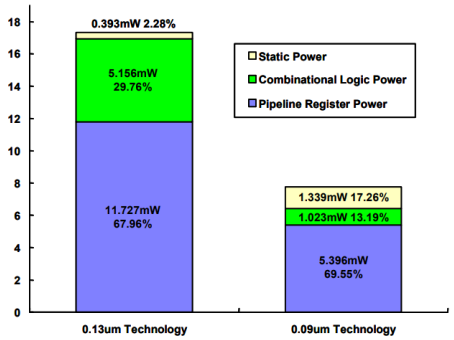
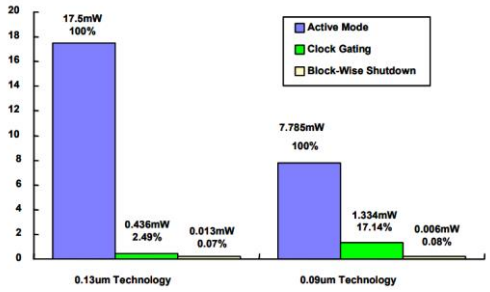


Figure 34: *Different Type of power Consumption*



7. Performance Comparison of Multipliers

Analysis of Area Performance

OF GENERIC 4 BIT MULTIPLIERS

COMPARSION / ALGORITHM	LUTS	SLICES	GATE COUNT
ARRAY	33	18	198
BOOTH	39	23	352
COLUMN	24	13	144
WALLACE TREE	46	32	695

Table 2 :*Analysis of Area Performance 4-bit Multipliers*

Other Performance Evaluation

4 BIT MULTIPLIERS

COMPARSION / ALGORITHM	PEAK MEMORY USAGE	NUMBER OF BONDED IOBS	MAPPER VERSION
ARRAY	103 MB	16 OUT OF 141 11%	SPARTAN3 -- $REVISION: 1.16 $
BOOTH	104 MB	17 OUT OF 141 12%	SPARTAN3 -- $REVISION: 1.16 $
COLUMN	102 MB	16 OUT OF 141 11%	SPARTAN3 -- $REVISION: 1.16 $
WALLACE TREE	104 MB	17 OUT OF 141 12%	SPARTAN3 -- $REVISION: 1.16 $

Table 4:*Other Performance Evaluation 4-bit*

Analysis of Area Performance

OF GENERIC 16 BIT MULTIPLIERS

COMPARSION / ALGORITHM	LUTS	SLICES	GATE COUNT
ARRAY	525	268	3150
BOOTH	623	321	5563
COLUMN	480	241	2880
WALLACE TREE	438	240	6217

Table 3: *Analysis of Area Performance 16-bit Multipliers*

Other Performance Evaluation

16 BIT MULTIPLIERS

COMPARSION / ALGORITHM	PEAK MEMORY USAGE	NUMBER OF BONDED IOBS	MAPPER VERSION
ARRAY	105 MB	64 OUT OF 141 45%	SPARTAN3 -- $REVISION: 1.16 $
BOOTH	106 MB	65 OUT OF 141 46%	SPARTAN3 -- $REVISION: 1.16 $
COLUMN	105 MB	64 OUT OF 141 45%	SPARTAN3 -- $REVISION: 1.16 $
WALLACE TREE	105 MB	65 OUT OF 141 33%	SPARTAN3 -- $REVISION: 1.16 $

Table 5 : *Other Performance Evaluation 16-bit*

8. Drawbacks of pass gate implementation

The pass gate reduces the hardware area spacing as we mentioned above. Nevertheless, you may face some issues while implementing your designed circuit like it complex the reliability of the system once you need to modify your load or to reduce it too other. or even if you want to change its position. So, the pass gate implementation implies to rebuild the circuit again and it is not flexible in this subject and add complexity to the circuit editing.

Conclusions

This paper presented the 8x8 radix- 3-bit grouping multiplier enhancement using the hardware minimization area by applying 32nm with the idea of implementing all the functional units and their relevant logical gates from transmission gate which plays a vital role in reducing the number of required transistors.

After discussing the various techniques in designing the multiplier as mentioned above and different multipliers algorithms, our design was the enhancement version of them.

To sum up, this undertaking proved to be an extraordinary exploration into the realm of Very Large-Scale Integration (VLSI). Throughout this endeavor, we not only acquired a profound comprehension of diverse multiplication techniques but also conducted assessments to quantify the potential savings in power, delay, and area achieved by implementing the proposed design. This experience further fostered our appreciation for the intricate equilibrium required in digital circuit design, where considerations of power consumption, latency, and area utilization intricately intersect.

9. References

[1] https://www.inass.org/2021/2021022803.pdf.
Accessed on 3-1-2024 at 4:34 PM.

[2] https://www.ijert.org/research/asic-implementation-of-dadda-multiplier-IJERTV8IS070198.pdf .
Accessed on 4-1-2024 at 5:34 PM.

[3] https://www.geeksforgeeks.org/computer-organization-booths-algorithm/ .
Accessed on 4-1-2024 at 7:39 PM.

[4] Performance analysis and implementation of approximate multipliers on spartan 6 FPGA | International journal of health sciences (sciencescholar.us).
Accessed on 6-1-2023 at 3:34 PM.

[5] https://iopscience.iop.org/article/10.1088/1742-6596/2225/1/012003/meta .
Accessed on 12-1-2024 at 4:34 PM.

[6] https://journal.uob.edu.bh/handle/123456789/4426?show=full .
Accessed on 13-1-2023 at 4:34 PM.

[7] https://www.researchgate.net/publication/283841595_A_study_of_performance_comparison_of_digital_multipliers_using_22nm_strained_silicon_technology.
Accessed on 14-1-2024 at 4:34 PM.

[8] https://www.researchgate.net/publication/283230862_High_speed_low_power_64-bit_comparator_designed_using_current_comparison_based_domino_logic.
Accessed on 15-1-2024 at 4:34 PM.

[9] CMOS two-input NAND and AND gates (uni-hamburg.de) .
Accessed on 18-1-2024 at 4:34 PM.