LOW POWER AND HIGH SPEED DADDA MULTIPLIER USING APPROXIMATE FULL ADDER COMPRESSORS

[1]A.Sivasai kumar[2] Challa Mahidhanush Reddy[3] Konidina Kiran[4] Thodeti Pranay[5] Mukiti Uday Kumar

[1]Assistant Professor,ECE Department,NECN, Nellore,AP, India

[2345]UG Scholor, ECE, NECN, Nellore, AP, India

Abstract: *Approximate computing is widely used to achieve energy-efficient system design in Very Large-Scale Integration (VLSI). This approach is particularly suitable for signal processing and multimedia applications where low power consumption is the primary concern. Approximate computing enables faster and significant results with a trade-off in reduced accuracy. In this project, a novel design methodology is proposed based on monolithic 4:2 compressors. The proposed approach minimizes the number of stages in partial product multiplication. The monolithic compressor demonstrated superior performance compared to other 4:2 compressors. Proposed design employs majority-logic-based techniques with Dadda multiplication, implementing a new partial product reduction format that significantly reduces maximum output delay. This method also reduces the utilization of MOSFETs compared to other multipliers, such as Wallace Tree Multipliers. Simulation results validate that the proposed approximate computing-based Dadda multiplier achieves substantial reductions in area utilization, dynamic power consumption, and processing time compared to conventional designs. The proposed methodology highlights the efficiency of majority-logic-based designs for VLSI applications, particularly in scenarios where minor accuracy losses are acceptable.*

Keywords: *Approximate Computing, Dadda Multiplier, Majority Logic 4:2 Compressor, Low Power VLSI, High-Speed Multiplication, Energy-Efficient Design.*

I. INTRODUCTION

Very-large-scale integration (VLSI) is the process of creating an integrated circuit (IC) by combining thousands of transistors into a single chip. VLSI began in the 1970s. When complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device.

Before the introduction of VLSI technology, most ICs had a limited set of functions could perform. An electronic circuit might consist of a CPU, ROM, RAM and other glue logic. VLSI lets IC designers add all of these into one chip.

The electronics industry has achieved a phenomenal growth over the last few decades, mainly due to the rapid advances in large scale integration technologies and system design applications [1]. With the advent of very-large scale integration (VLSI) designs, the number of applications of integrated circuits (ICs) in high-performance computing, controls, telecommunications, image and video processing, and consumer electronics has been rising at a very fast pace [4].

The current cutting-edge technologies such as high resolution and low bit-rate video and cellular communications provide the end-users a marvelous amount of applications, processing power and portability [5]. This trend is expected to grow rapidly, with very important implications on VLSI design and systems design.

THE DESIGN PROCESS OF A VLSI IC

Overall, VLSI IC design incorporates two primary stages or parts:

- **Front-End Design:** This includes digital design using a hardware description language, for example, Verilog, System Verilog, and VHDL. Furthermore, this stage encompasses design verification via simulation and other verification techniques. The entire process also incorporates designing, which starts with the gates and continues through to design for testability.
- **Back-End Design:** This consists of characterization and CMOS library design. Additionally, it involves fault simulation and physical design.

The entire design process follows a step-by-step approach, and the following are the front-end design steps:

- Problem Specification: This is a high-level interpretation of a system. Key parameters are addressed, such as design techniques, functionality, performance, fabrication technology, and physical dimensions. The final specifications include the power, functionality, speed, and size of the VLSI system.
- Architecture Definition: This includes fundamental specifications such as floating-point units and which system to use, such as RISC or CISC and ALU's cache size.
- Functional Design: This recognizes the vital functional units of a system and, thus, enables identification of each unit's physical and electrical specifications and interconnect requirements.
- Logic Design: This step involves control flow, Boolean expressions, word width, and register allocation.
- Circuit Design: This step performs the realization of the circuit in the form of a netlist. Since this is a software step, it utilizes simulation to check the outcome.
- Physical Design: In this step, the layout is created by converting the netlist into a geometrical depiction. This step also follows some preconceived static rules, such as the lambda rules, which afford precise details of the ratio, spacing between components, and size.

The following are the back-end design steps for hardware development:

- Wafer Processing: This step utilizes pure silicon melted in a pot at 1400^{o} C. Then, a small seed comprising the required crystal orientation is injected into liquefied silicon and gradually pulled out, 1mm per minute. The silicon crystal is manufactured as a cylindrical ingot and cut it into discs or wafers before polishing and crystal orientation [6].
- Lithography: This process (photolithography) includes masking with photo etching and a photographic mask. Next, a photoresist film applied on the wafer. A photo aligner then aligns the wafer to a mask. Finally, expose the wafer to ultraviolet light, thus highlighting the tracks through the mask [6].
- Etching: Here, selectively removing material from the surface of the wafer to produce patterns. With an etching mask to protect the essential parts of the material, additional plasma or chemicals is used to remove the remaining photoresist[6].
- Ion Implantation: Here, a method utilized to achieve a desired electrical characteristic in the semiconductor, i.e., a process of adding dopants. The process uses a beam of high-energy dopant ions to target precise areas of the wafer. The beam's energy level determines the depth of wafer penetration.

- Metallization: In this step, a thin layer of aluminum applied over the entire wafer.
- Assembly and Packaging: Every one of the wafers contains hundreds of chips. Therefore, a diamond saw used to cut the wafers into single chips. Afterward, receive electrical testing, and discard the failures. In contrast, chips that pass receive a thorough visual inspection utilizing a microscope. Finally, package the chips that pass the visual inspection as well as recheck chips.

VLSI technology is ideally suited to the demands of today's electronic devices and systems. With the ever-increasing demand for miniaturization, portability, performance, reliability, and functionality, VLSI technology will continue to drive electronics advancement.

Multipliers:

Multipliers are extensively used in digital signal processing where the speed of multiplication and area of power consumption are important. The objective of optimal multiplier is to provide low power consumption and high speed. If appropriate optimized multiplier is not chosen it ultimately produces lag in digital circuits.

Multiplication plays an important role in order to carry out the operations in Digital filters, DSP, image processing, etc. The primary block performing such operations are known as multipliers, which consume more power and operating time. In any conventional multiplier, there are three basic stages like

1. Generation of partial product
2. Reduction of Partial product and
3. Addition of propagated carries of previous stage.

Partial product reduction is very complicated while using compressors. The functionality of compressor is to count the number of ones in the given input. Also, these are the fundamental processing elements (PEs) for the collection of partial products in the multiplication operation. Compressors provide knowledge regarding the progressive increasing demand of reduced power and high speed in the overall critical path of the circuit.

II. FUNCTIONAL OVERVIEW

Many high end surveillance and medical applications based on image and video processing algorithms need high reliability and most important utmost precision. But on other hand there many arithmetic logic based processing algorithms in which human visibility it seems to be accurate though it remains numerically approximate [2]. The numerically inaccurate results provides some freedom of relaxation to proceed with an approximate computation. This kind of requirement give the new ear of research idea that incorporates low-power designs with reduced area and less propagation delay in different abstraction levels.

There is been a tradeoff between delay and power for the design consideration of the circuit. To have well designed circuit with high-speed performance, which could only be achieved at the expense of more energy consumption and also larger delays are seen with low energy consumption. However less attention is given to design the approximate multiplier because in general multiplication is thought as a repeated sum of partial products, but it is not viable to have straight forward use of approximate adders for designing [3].

As this could lead to inefficient hardware complexities and other performance measures. Some other works on designing area and power efficient approximate multipliers showed power savings compared to an exact multiplier; which considered altered partial products in two variants of 16-bit multipliers.

The novelty approach for error resilient image and signal processing applications have been designed with approximate compressors. The significant reductions in area and power with high signal to noise ratio were the achievements as results in these kind of algorithms [3].

In this project, new approximate arithmetic computing based approach is proposed for the Dadda multiplier. In initial stages of multiplier, an approximate adder is proposed i.e. Almost full adder based adder compressor and

later stages majority logic based hybrid combination is proposed for Dadda multiplier design [3].

The moderate digital design logic with less power consumption and minimal de- vice utilization are the two main objectives of this implementation proposed in this project, which maintains the performance nearly same as exact multiplication process. In this project, a novel design approach based on majority 4:2 compressor also proposed to have reduction in partial product stage of multiplication.

III. DESIGN

Proposed majority logic based 4:2 Adder compressor

The proposed Majority logic based 4:2 compressor is demonstrated in Fig 1.

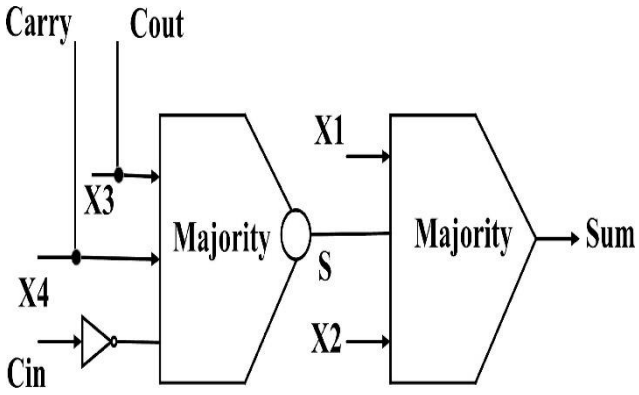


Fig 1 Majority logic based 4:2 Adder compressor

XOR–XNOR module's logic equations are as follows in Eq. 2, Eq. 3a to Eq. 3c. Majority based approximate adder's sum and carry equations are as Eq. 4a to Eq. 4c

$$C_{out} = (x1 \oplus x2) \cdot x3 + x1 \cdot x2 \quad (2)$$
$$= (x1 \oplus x2) \cdot x3 + (x1 \oplus x2) \cdot x1$$

$$S = (x1 \oplus x2 \oplus x3) \quad (3a)$$
$$sum = (S \oplus x4 \oplus C_{in}) \quad (3b)$$
$$= (x1 \oplus 2 \oplus x3 \oplus x4 \oplus C_{in})$$
$$Carry = S \oplus x4) \cdot C_{in} + S \cdot x4 \quad (3c)$$
$$= (x1 \oplus 2 \oplus x3 \oplus x4) \cdot C_{in} + x1 \oplus 2 \oplus x3 \cdot x4$$

$$S = Maj(X1,X2,X3) \quad (4a)$$
$$Carry = X4 \quad (4b)$$

$$Sum = Cin \quad (4c)$$

Majority logic-based approach suffice the problem of the above mentioned all 4:2 compressors having AND- OR and XOR combination. Proposed approach will reduce the circuit latency and hardware complexity at the cost of exploiting more negligible error. The following equation explains the logic of the proposed imprecise 4:2 compressor. Fig 2, shows the proposed 16-bit Dadda multiplier employing all 4:2 compressors with Full Adders, Half Adders and 4:2 majority- based compressors. The 4:2 compressors have been utilized since the proposed compressor has turned into 4:2 compressor as described in proposed compressors part of this section.

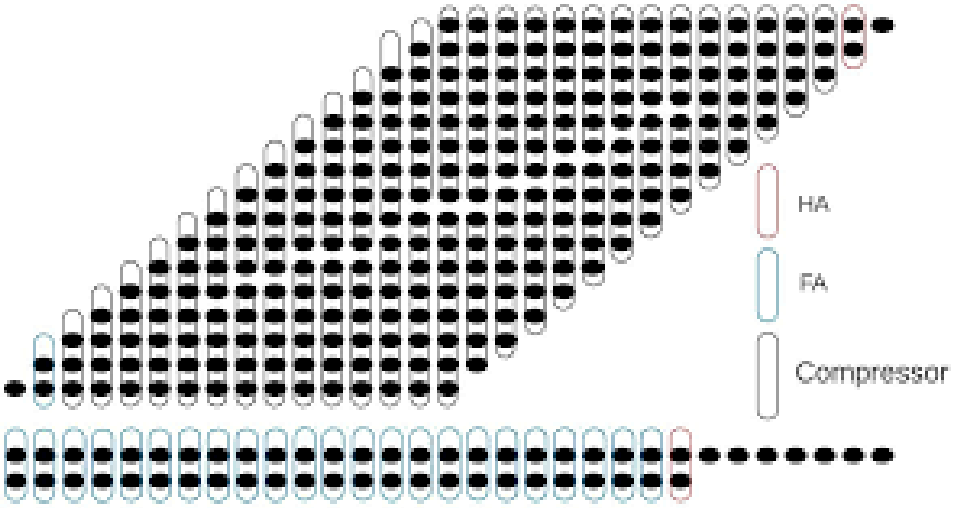


Fig 2 16-bit approximate Dadda Multiplier using majority logic based compressor

IV. WORKING

PARTIAL PRODUCT REDUCTION FOR MAJORITY LOGIC BASED DADDA MULTIPLIER

Generally, tree (parallel) multipliers are considered to be the efficient in delivering the circuit performance. AND operation is used to multiply the multiplicand bit and multiplier bit in first phase of partial product generation in the tree. Dadda Multiplier is designed to reduce the complexity of an operation. As mentioned earlier, Dadda multiplier performs 3 to 5 stages for multiplication process as other conventional multiplier.

Generation of partial product with shifting and AND-ing operation of the Dadda multiplier is performed in following way as shown in Fig 3

P3	P2	P1	P0			
			Q3	Q2	Q1	Q0
				P3Q0	P2Q0	P1Q0	P0Q0
			P3Q1	P2Q1	P1Q1	P0Q1	
		P3Q2	P2Q2	P1Q2	P0Q2		
	P3Q3	P2Q3	P1Q3	P0Q3			
S6	S5	S4	S3	S2	S1	S0	

Fig 3 Generation of Partial product for Dadda multiplier

After completion of first stage of generation of partial products, there is second stage of the partial products reduction can be implemented with used of carry propagation based full adder and half adder digital logic designs. This stage is the heart of the multiplication process which has been divided into the different stages of the partial product reduction by implementing various approximate adder compressors and finally the final product answer

is obtained by addition. There are mainly two types of tree multi- plier exists namely Wallace and Dadda. The difference between both the multiplier is its reduction algorithm employed for partial product reduction stage. Addition is performed with the help of Carry Propagate Adder (CPA) to obtain the final result of reduced bits. with help of Wallace Dadda are tree multipliers, circuit performance is drastically improved in terms of speed. There are various techniques to design Approximate Dadda which has been already discussed in literature review part.

Three different configurations of approximate Dadda multipliers using Almost full adder are implemented where it is mainly focusing on device utilization and power consumption of the multiplier. In all the configurations instead of all only few of the accurate adders have been replaced with the approximate almost full adders. It ensures the drastic reduction in the device utilization at the cost of accuracy. So, to avoid such a situation, the adders are replaced in such a manner that the final outcome of the results will maintain the accuracy along with a noticeable reduction in power consumption and device utilization. To achieve this, proposed almost full adder is used at first two stages and all the other stages are kept as it is for the designed Dadda multiplier configuration. It shown in Fig 4 to Fig 8.

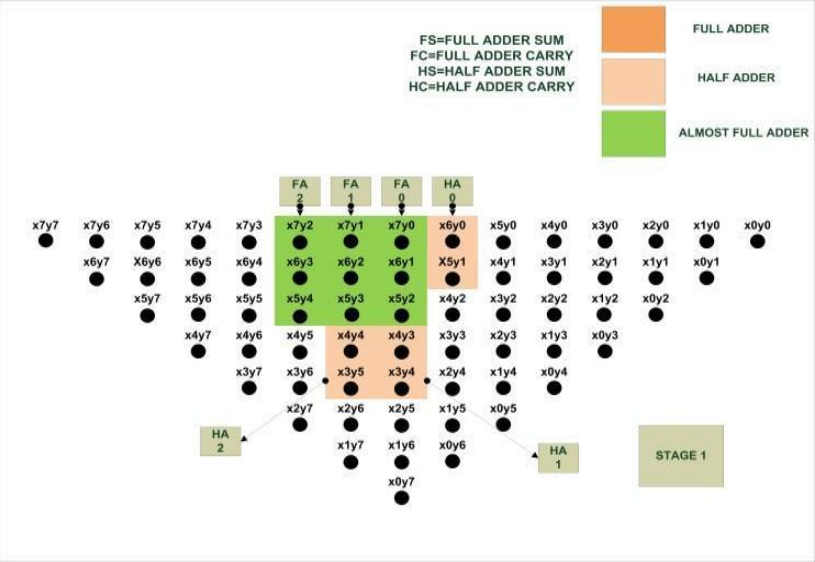


Fig 4 Stage 1 of Dadda Multiplication

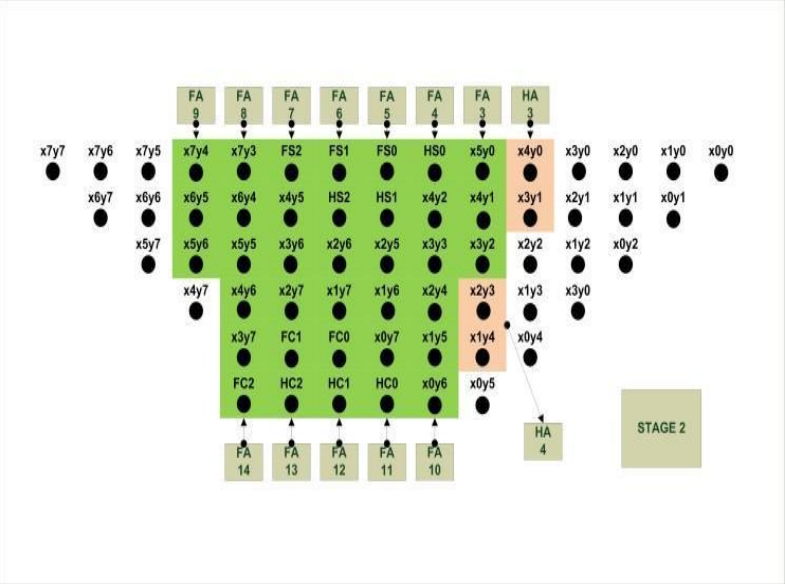


Fig. 5 Stage 2 of Dadda Multiplication

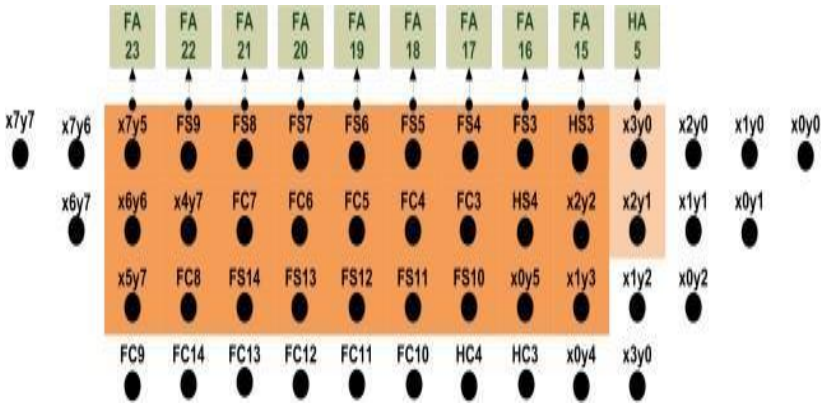


Fig 6 Stage 3 of Dadda Multiplication

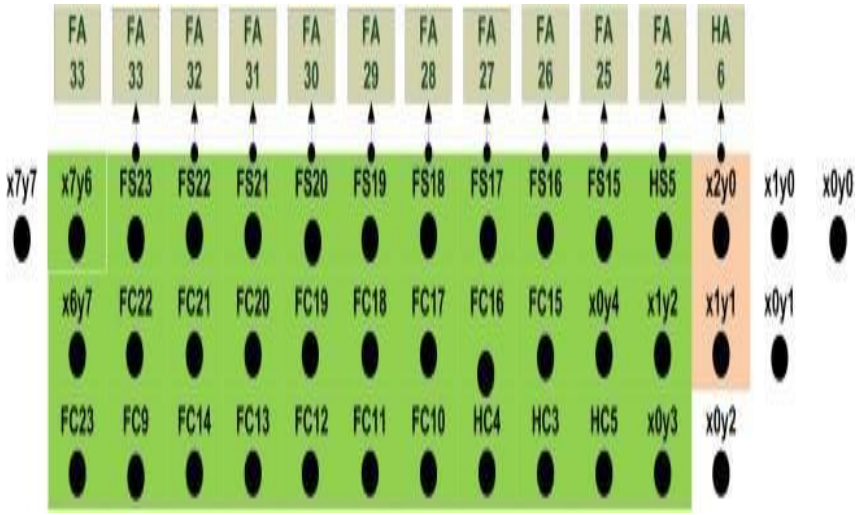


Fig 7 Stage 4 of Dadda Multiplication

An algorithm proposed by Dadda to achieve reduced number of reduction stage. The fundamental concept of this algorithm is working back from two rows having predetermined matrix heights for N x N multipliers. The height of each intermediate stage is kept as 1.5 times the height of the next stage which is shown in Eq. 5,

Height of stage i = (3/2) ∗ Height of stage i+1 (5)

Hence, the sequence of stage heights will become 2,3,4,6,9,13.The five stage 8x8 Dadda tree multiplier is shown in Fig 4.10. Dadda multiplier uses recursive algorithm for the partial product reduction. The maximum height of this multiplier is 8 bits and hence the height of next stage will be 6. The maximum column (Column) height will be 6 after reduction.

Following are the steps of Dadda multiplier (here Column1, Column2, represents the column number): The following steps elaborate the design of majority logic and almost full adder based hybrid combination of Dadda multiplier which is demonstrated in Fig 4.4.

Step 1: There is no change is desired for initial column 1 to 6.

Step 2: 7th Bit is utilized by column 7 while reduced 6 bits are incorporated with half adder

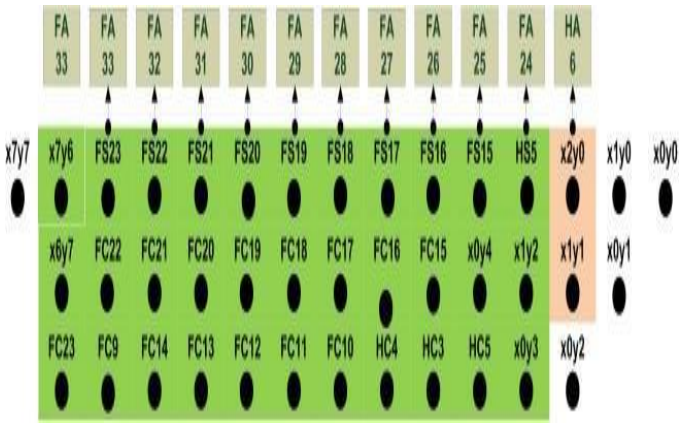


Fig 8 Stage 5 of Dadda Multiplication

Step 3: Carry generated during column 7 along with 8 bits are unitized in Column 8. So there are reduction of 3 bits in half adder and full adder as total 6 bits instead of 9 bits.

Step 4: In Column 9 two carry propagated in column 8 and total 7 bits are utilized and only 6 bits are needed for half adder and full adder.

Step 5: Same step 4 is repeated for column 10.

V. IMPLEMENTATION

DIGITAL CIRCUIT DESIGN USING XILINX TOOLS

Xilinx Tools is a suite of software tools used for the design of digital circuits implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The design procedure consists of (a) design entry, (b) synthesis and implementation of the design, (c) functional simulation and (d) testing and verification. Digital designs can be entered in various ways using the above CAD tools: using a schematic entry tool, using a hardware description language (HDL) – Verilog or VHDL or a combination of both.

In this lab, design flow is used that involves the use of Verilog HDL.

The CAD tools enable to design combinational and sequential circuits starting with Verilog HDL design specifications. The steps of this design procedure are listed below:

1. Create Verilog design input file(s) using template driven editor.
2. Compile and implement the Verilog design file(s).
3. Create the test-vectors and simulate the design (functional simulation) without using a PLD (FPGA or CPLD).
4. Assign input/output pins to implement the design on a target device.
5. Download bitstream to an FPGA or CPLD device.
6. Test design on FPGA/CPLD device

A Verilog input file in the Xilinx software environment consists of the following segments:

- **Header:** module name, list of input and output ports.
- **Declarations:** input and output ports, registers and wires.
- **Logic Descriptions:** equations, state machines and logic functions.
- **End:** endmodule

All designs for this lab must be specified in the above Verilog input format. Note that the state diagram segment does not exist for combinational logic designs.

XILINX VIVADO

Xilinx Vivado is an integrated development environment (IDE) provided by AMD Xilinx, designed specifically for Field Programmable Gate Array (FPGA) and System-on-Chip (SoC) development. It is a next-generation tool that enables engineers to implement, analyze, and optimize their digital designs efficiently.

ROLE OF XILINX VIVADO IN THE PROJECT

In this project, Xilinx Vivado was utilized for the design, implementation, and analysis of the [mention specific module/project, e.g., "low-power ALU for MIPS32"].

The tool was instrumental in:

- Design Entry: The HDL (Hardware Description Language) code for the [module/system] was written in Verilog and entered into Vivado for synthesis and implementation.
- IP Integration: Vivado's IP catalog was used to incorporate [mention specific IP blocks, if any].
- Synthesis and Implementation: The HDL code was synthesized into a gate-level netlist and implemented on the target FPGA board ([mention board name, e.g., Xilinx Artix-7]).
- Simulation: Functional and timing simulations were conducted using Vivado’s integrated simulator to verify the correctness of the design.
- Timing and Power Analysis: Post-implementation timing and power analysis were performed to ensure the design met the desired specifications.
- Bitstream Generation: A bitstream file was generated for programming the FPGA with the designed logic.
- Debugging and Validation: The Vivado Logic Analyzer was used to debug and validate the functionality of the implemented design.

VI. RESULTS

The multiplier using exact compressor power consumption details is as shown in Fig 9

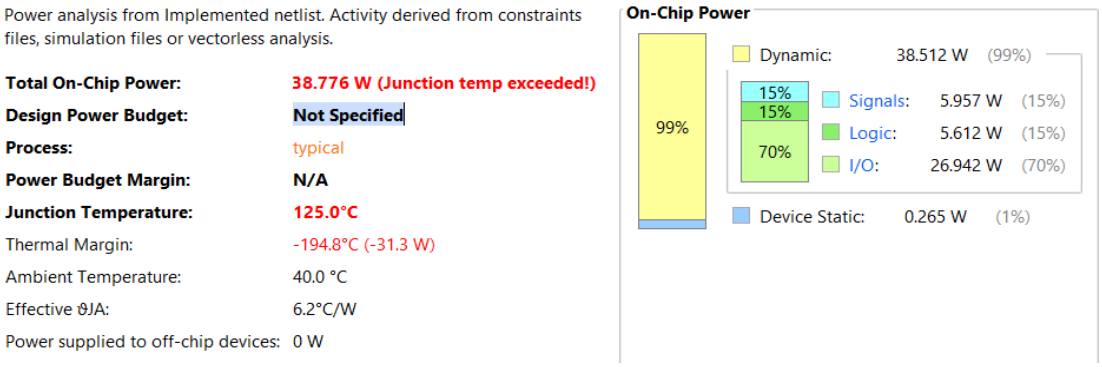


Fig 9 Power consumption of 16 – bit DADDA multiplier using Exact compressor

The multiplier using exact compressor LUT utilization details is as shown in Fig 10.

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	373	14600	2.55
IO	64	112	57.14

Fig 10 LUT utilization of 16 – bit DADDA multiplier using Exact compressor

The multiplier using exact compressor delay details is as shown on Fig 11.

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destinat
Path 1	∞	19	32	b[1]	y[31]	27.766	8.579	19.187	∞	input port clock	
Path 2	∞	19	32	b[1]	y[28]	27.405	8.577	18.828	∞	input port clock	
Path 3	∞	19	32	b[1]	y[29]	27.224	8.566	18.658	∞	input port clock	
Path 4	∞	19	32	b[1]	y[30]	27.123	8.341	18.781	∞	input port clock	
Path 5	∞	18	32	b[1]	y[27]	26.719	8.224	18.495	∞	input port clock	
Path 6	∞	17	32	b[1]	y[26]	25.389	7.862	17.527	∞	input port clock	
Path 7	∞	17	32	b[1]	y[25]	25.384	7.857	17.527	∞	input port clock	
Path 8	∞	16	32	b[1]	y[24]	24.061	7.509	16.552	∞	input port clock	
Path 9	∞	15	32	b[1]	y[23]	23.567	7.384	16.183	∞	input port clock	
Path 10	∞	14	32	b[1]	y[22]	22.252	7.022	15.230	∞	input port clock	

Fig 11 Delay of 16 - bit DADDA multiplier using Exact compressor

The multiplier using Majority logic compressor power consumption details is as shown in Fig 12.

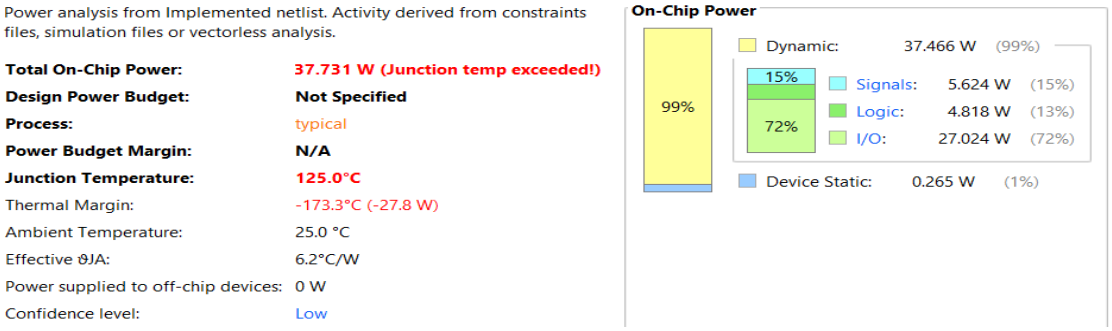
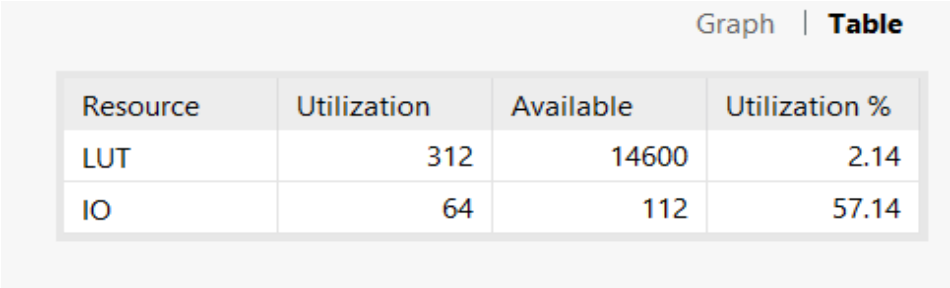


Fig 12 Power consumption of 16 – bit DADDA multiplier using Majority logic compressor

The multiplier using Majority logic compressor LUT utilization details is as shown in Fig 13.

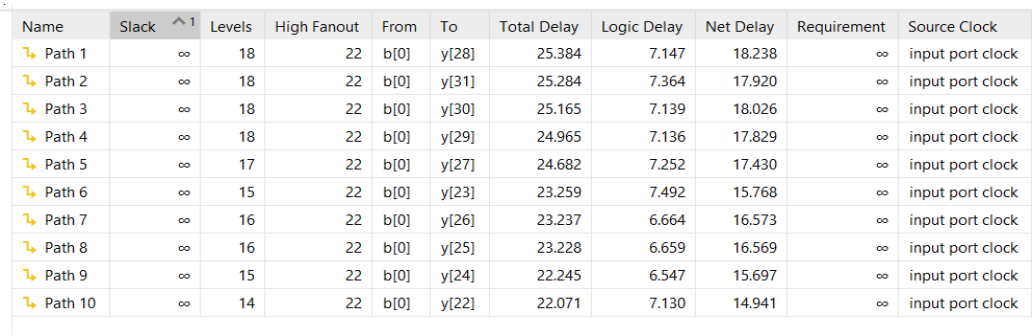


Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	312	14600	2.14
IO	64	112	57.14

Fig 13 LUT utilization of 16 – bit DADDA multiplier using Majority logic compressor

The multiplier using Majority logic compressor delay details is as shown in Fig 14.



Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	18	22	b[0]	y[28]	25.384	7.147	18.238	∞	input port clock
Path 2	∞	18	22	b[0]	y[31]	25.284	7.364	17.920	∞	input port clock
Path 3	∞	18	22	b[0]	y[30]	25.165	7.139	18.026	∞	input port clock
Path 4	∞	18	22	b[0]	y[29]	24.965	7.136	17.829	∞	input port clock
Path 5	∞	17	22	b[0]	y[27]	24.682	7.252	17.430	∞	input port clock
Path 6	∞	15	22	b[0]	y[23]	23.259	7.492	15.768	∞	input port clock
Path 7	∞	16	22	b[0]	y[26]	23.237	6.664	16.573	∞	input port clock
Path 8	∞	16	22	b[0]	y[25]	23.228	6.659	16.569	∞	input port clock
Path 9	∞	15	22	b[0]	y[24]	22.245	6.547	15.697	∞	input port clock
Path 10	∞	14	22	b[0]	y[22]	22.071	7.130	14.941	∞	input port clock

Fig 14 Delay of 16 – bit DADDA multiplier using Majority logic compressor

Table I: Comparison of parameters between Exact DADDA multiplier and Approximate DADDA multiplier

PARAMETERS	**EXACT DADDA MULTIPLIER**	**APPROXIMATE DADDA MULTIPLIER**
LUT's	**373**	**312**
Power (W)	**38.776**	**37.731**
Delay (ns)	**27.766**	**25.384**

The results obtained from the implementation and analysis of the proposed low-power, high-speed Dadda multiplier using approximate full adders and majority logic-based 4:2 compressors reveal significant improvements over the traditional exact Dadda multiplier as shown in Table I.

One of the key performance parameters analysed was the Look-Up Table (LUT) utilization. The approximate Dadda multiplier recorded a LUT count of 312 compared to 373 for the exact Dadda multiplier. This reduction indicates a decrease in hardware complexity, making the approximate design more area-efficient.

Power consumption is another critical factor in VLSI design, particularly for portable and energy-constrained applications. The approximate Dadda multiplier achieved a power consumption of 37.731 W, which is slightly lower than the 38.776 W recorded for the exact Dadda multiplier. While the reduction may seem marginal, it is significant in large-scale implementations where power efficiency is crucial.

The most notable improvement is seen in the delay parameter. The approximate Dadda multiplier demonstrated a delay of 25.384 ns, which is considerably lower than the 27.766 ns observed in the exact Dadda multiplier. This substantial reduction in delay highlights the effectiveness of the proposed approximate compressors in enhancing computational speed.

The trade-off for these improvements is a minor loss of accuracy, which is acceptable for many applications like image and video processing where approximate results do not significantly impact overall performance. The integration of almost full adders and majority logic-based 4:2 compressors optimally balances power, area, and speed, making the proposed design a viable option for energy-efficient, high-speed computing systems.

In conclusion, the proposed approximate Dadda multiplier successfully reduces hardware complexity and power consumption while significantly improving processing speed. These enhancements make it a compelling choice for applications that can tolerate slight accuracy loss in favour of better performance and efficiency.

VII. CONCLUSION

In this project, an energy-efficient 16-bit Dadda multiplier design is proposed using approximate computing techniques and novel monolithic 4:2 compressors. The proposed design effectively minimizes area utilization by reducing the transistor count through the use of approximate 4:2 compressors. Additionally, the optimized partial product reduction stages significantly enhance processing speed. This approach highlights the potential of majority-logic-based designs in achieving a balance between energy efficiency and computational accuracy. The proposed design is particularly advantageous for applications in signal processing and multimedia, where slight accuracy trade-offs are acceptable in exchange for substantial improvements in power and area efficiency.

VIII. REFERENCES

[1] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, SubThreshold Design for Ultra Low-Power Systems, vol. 95. New York, NY, USA:Springer, 2006.

[2] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[3] M. H. Moaiyeri, F. Sabetzadeh, and S. Angizi, "An efficient majority based compressor for approximate computing in the nano era," Microsyst. Technol., vol. 24, no. 3, pp. 1589–1601, Mar. 2018.

[4] S. Mittal, "A survey of techniques for approximate computing," ACM Comput. Surv., vol. 48, no. 4, p. 62, May 2016.

[5] C.H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

[6] Pathak, Ketki C., Jignesh N. Sarvaiya, Anand D. Darji, Shreya Diwan, Anjali Gangadwala, Zinal Bhatt, and Azba Patel. "An Efficient Dadda Multiplier using Approximate Adder." In 2020 IEEE REGION 10 CONFERENCE (TENCON), pp. 176-181. IEEE, 2020.