A MULTIPLIER'S ARCHITECTURE BASED ON APPROXIMATE 2- BIT LUT ADDERS

C. Yasmeen[1], Dr. S. Chandra Mohan Reddy[2]

[1]PG-Scholar, Department of ECE
Professor, Department of ECE
JNTUA COLLEGE OF ENGINEERING, ANANTAPUR,A.P.

Abstract – This paper delves into the creation and assessment of a new multiplier architecture that incorporates approximate 2-bit Look-Up Table (LUT) adders. Recognizing the demands of today's applications for swift operations with acceptable power consumption. The increasing advancement of technology has led to a corresponding rise in the demand for high-speed digital systems, primarily because to the crucial role played by multipliers in such systems. This paper presents the design of an 8x8 multiplier using 2-Bit LUT (Look Up Table) approximation adders, specifically the LEADx (Low Error & Area efficient) and APEx (Area & Power efficient) approximate adders. The proposed method incorporates the usage of approximate computing technology, which introduces a distinct aspect to digital design by effectively lowering area, power consumption, and latency. The current approach involves the utilization of 16-bit LEADx and APEx approximation adders to minimize error, area, power, and latency, which are the primary factors for reducing the rate of inaccuracy. Lookup tables (LUTs) serve as the primary components that comprise fields-programmable gate arrays (FPGAs). A user-defined truth table is stored in a Look Up Table (LUT) that is part of the Field-Programmable Gate Array (FPGA). This truth table is initialized when powering up the chip. This study primarily concerns the enhancement of multiplier efficiency through the integration of approximate 2-bit Look-Up Table (LUT) adders. The integration of these adders offers notable benefits in power consumption, space utilization, and processing speed. The proposed 8x8 multiplier demonstrated improvements in area, power, and latency metrics by incorporating 2-Bit LUT approximate adders and employing the approximate computing technique at the software level.

INDEX TERMS: - Approximate computing, LEADx, APEx, LUT, FPGA

1.INTRODUCTION:

The processing of multimedia content permits the accuracy to decline for a variety of reasons. One is that human sense organs are restricted in their ability to detect minute differences. The subsequent digital signal processing chain, on the other hand, must not evaluate the input data with the highest level of accuracy if the input data, such as that obtained from an image sensor, is of lower quality, such as because of less-than-ideal lighting circumstances. Utilizing approximation computing enables resource- and energy-efficient multimedia processing [1],[2]. Since a few years ago, this new paradigm has attracted a lot of interest from researchers.The use and exploitation of reconfigurable FPGAs in programmable and adaptive systems is comparatively young and the topic of ongoing research [3]. The relentless pursuit for increased performance in digital systems often comes at the expense of increased silicon area and increased power consumption. Approximate computing, an emerging paradigm, sacrifices some precision for increased performance and efficiency. This is particularly beneficial in error-tolerant applications like multimedia processing. In this study, We investigate the feasibility of including about 2-bit LUT adders in multiplier systems. It encompasses a wide range of computation methods that, rather than returning a result that is guaranteed to be accurate, may yield results that are approximative but nonetheless serve the intended goal.

1.1 OBJECTIVE OF THE STUDY:

The proposed design involves the utilisation of an approximate 2-bit adder (referred to as Add2 in the original work) to create an 8-bit approximate multiplier [5]. This approach allows for the generation of approximate results, resulting in improved performance metrics such as area, power, and time.

1.2. Background:

Approximate computing refers to a computational paradigm wherein systems are deliberately engineered to yield outcomes that may not be entirely precise, yet remain sufficiently satisfactory for the intended use. This frequently results in notable reductions in energy consumption, since the computational process can be executed at a faster rate and/or using less energy [1]. In this paper, we provide a novel methodology known as "Abacus" which aims to streamline the development of approximate computing circuits. Automated synthesis pertains to the process of generating circuit designs based on higher-level behavioural criteria. Instead of engaging in the design of circuits at the gate or transistor level, behavioural synthesis operates at a more abstract level, focusing on identifying the desired behaviour of a system rather than the specific implementation details of how it achieves that behaviour. The utilisation of tools facilitates the automatic generation of a circuit that serves the purpose of enabling the synthesis of circuits designed to purposefully incorporate approximation. The process of digital multiplication often yields outcomes that are both extensive and intricate, primarily because of the substantial values involved. Due to the intrinsic error tolerance of digital multipliers, designers have developed approximate multipliers. The objective of employing approximation multipliers is to diminish the dimensions and intricacy of the process while preserving its semantic significance. Compressors play a crucial function in diminishing the dimensions of intermediate or final outputs. Utilizing an XOR-MUX adder simultaneously minimises the quantity of logic gates used and the quantity of inputs processed [3]. In this paper, we aim to provide a thorough examination of the fundamental design ideas and approaches that are employed in the implementation of approximation computing across many the circuitry, architecture, and software layers that make up the computing stack. In a recent study [5], a novel 4:2 compressor design is introduced, which exhibits an error rate of around 18.75%. Furthermore, this proposed design demonstrates a notable reduction in energy consumption, averaging 15% less than the earlier designs that the literature has recorded. In this study, the multiplier is presented in two iterations, one of which uses only approximate compressors and the other of which uses both approximate and exact compressors. Circuits that are derived from the novel logic synthesis approach exhibit a reduction in circuit complexity and manufacturing expenses, while also demonstrating enhanced performance [6]. [7] Field Programmable Gate Arrays (FPGAs) hold significant prominence within the electronics sector as a pivotal device. Field-Programmable Gate Arrays (FPGAs) are commercially produced integrated circuits that possess the unique capability of being electrically reconfigurable, allowing them to effectively emulate a wide range of complex digital circuits or systems. The salient aspect of field-programmable gate arrays (FPGAs) is in their architectural design, which provides a comprehensive understanding of their programmable logic capabilities and interconnections.[8]With the possibility for cumulative gains from using various techniques across the system stack, approximation computing is a generally applicable paradigm. To take use of these advantages, it is necessary to redesign closely integrated approximate accelerators and to rethink several layers of the system stack to accept approximations from the ground up. By doing this, the applications will be able to operate in a setting where the architecture, programming model, and even the techniques used to implement the application are all fundamentally geared toward approximation computing. study that compares various parallel adders and suggests a hybrid adder. The Xilinx 14.7 ISE environment is used for all Adder results, and Verilog HDL is used for coding. For easier comparison, a specific graph and table of values for propagation delay, area, and needed transistor count are provided[10].

2.1 Methodology: -

An 8-bit multiplier architecture is designed by utilizing the 2-bit LUT approximate adders which are present in the base paper method to optimize the area, power & delay. The LUTs mainly define the behavior of the combinational logic designed with Verilog code. It simply generates output based on the input combination. When multiplicands are multiplied by each multiplier, partial products are formed. Approximate adders are used in reduction of partial products. In N-bit multiplier, NxN partial products will form and 2N is the result after reduction of partial products. In our proposed method Verilog code is framed & is simulated by using Xilinx vivado 2018 tool, Artix 7 AC01 board with speed -1 to design 8-bit multiplier architecture. Approximate adders are LEADx, APEx shown in the figure (1) & (2) respectively.

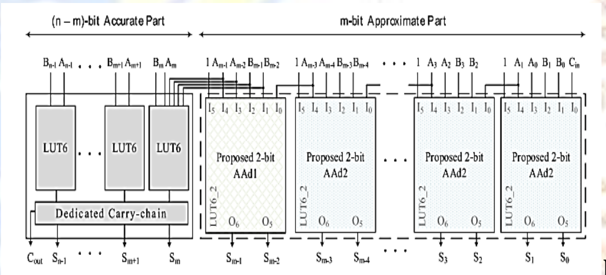


Fig 1: - N-bit Low error & Area Efficient Approximate adders (LEADx)

The 6-input LUTs used by FPGAs. Two 5-input functions may be implemented with the help of these LUTs. Performance of LUT-based implementation is unaffected by the implemented logic function's complexity. There are 5 inputs and 2 outputs on a 2-bit adder (AAd2 & AAd1). A 2-bit approximation adder can therefore be implemented using a LUT.

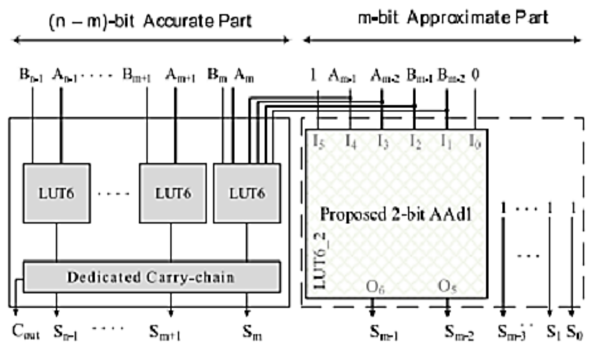


Fig 2: Area & Power Efficient Approximate Adder

In APEx, Finding a rough function that is not data dependent for the least significant m-2 bits of the LSP is the goal. It is not recommended to produce carry or to use carry when computing sums. In principle, any logic function with a 1-bit output can be used as a rough function to calculate the rough sum of 1-bit inputs at the ith bit location. Other acceptable approximations include functions with output constants of 0 or 1. Because no hardware will be needed for sum computation, fixing output 0 or output 1 will lower the approximate adder's size and power usage. In the majority of applications for digital signal processing, multiplication is a crucial operation.Speed, area, and power are the three key factors in any VLSI design. In high performance systems, multipliers are essential. As technology develops, there is a growing demand for high-speed digital systems because multipliers are a fundamental part of all digital systems.

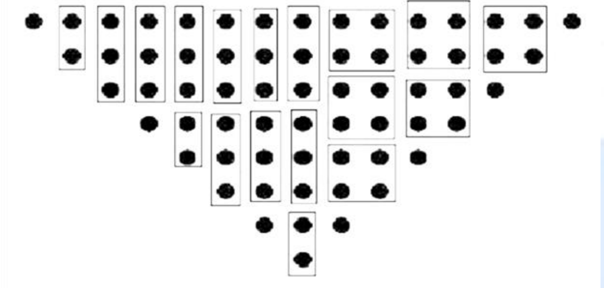
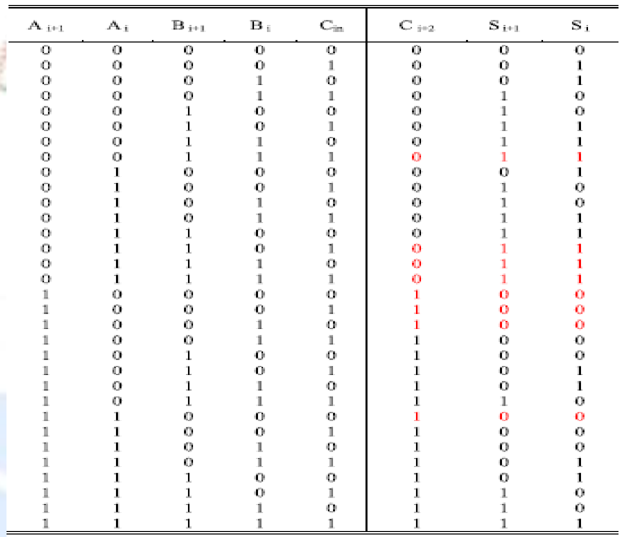


Fig 3: - Partial products of a multiplier in dot diagram

The proposed method incorporates the use of code pipelining in the approximate adder APEx. Instead of directly passing input values to the input ports, the inputs are stored in registers and then passed from the registers. The initialization of values is achieved through a reset signal, which clears any previously stored values. In APEx, we are approximating 8-bits of approximate part into constant 1's. It is shown as follows "assign s1[7:0] =8'b11111111 " to optimize area, power & delay which are main parameters for fast working of the multiplier. Whereas in the existing method only 6-bits in the approximate part are approximated directly into output constants as 1's. And 8-bits in the accurate part are computed by using 8 LUTs. Registers are used to store the sum values. The input of LUT is given as one of the inputs to the multiplexer and another input is from registers. All the carries are carried out by dedicated carry chain and given output as Cout. Outputs are obtained from the multiplexers.

The proposed method involves the design of a 16-bit LEADx, which is achieved by combining two 8-bit LEADx units. In this configuration, one of the 8-bit LEADx units is responsible for performing functions, while the other unit remains inactive, like a master-slave system. The reusability of an 8-bit LEADx is facilitated by the reusability feature of FPGA resources, specifically the Look-Up Table (LUT). In the 8-bit LEADx system, the computation of each 4-bit segment is performed using both an approximation component and an accurate component. In this context, 4-bit values are calculated using two adders, namely AAD1 and AAD2, each having a capacity of 2 bits. The inputs provided to the adder AAD1 should be about the same inputs as are given to the first lookup table (LUT) in the precise section. The ripple carry adder computes all the carries created in the accurate part by utilizing the XOR function, resulting in a reduction in area and delay.



A_{i+1}	A_i	B_{i+1}	B_i	C_{in}	C_{i+2}	S_{i+1}	S_i
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1
0	1	1	0	1	0	1	1
0	1	1	1	0	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	0	1	0	0
1	0	0	1	1	1	0	0
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	0
1	1	0	1	0	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

Table 1: - Truth table of the least important m-2 bits of LEADx, which are approximated by a 2-bit approximate adder (AAd2).

The truth table for the approximate 2-bit adder (Aad2) utilised for LEADx's least significant m-2-bit approximation is shown above.

3. Results and Discussion

3.1 Existing Method Results: -

	AREA(LUT)	POWER(W)	DELAY(ns)
LEADx	**15**	**10.342**	**6.499**
APEx	**12**	**7.344**	**6.499**

Table 1: - Area, Power, Delay values of LEADx, APEx

In the existing, 16 bit-LEADx, APEx approximate adders are designed in utilizing FPGA resources such as LUTs. 8 bit each are computed in approximate & accurate part. In LEADx, three 2 bit-AAD2 adders & one 2 bit-AAD1 adder are used for computation in approximate part. The inputs which given to AAD1 is same given to first LUT in the LSP (least significant part) of the accurate part. In accurate part 8 LUTs are used for computation in the accurate part. The main advantage of using FPGA resources like LUTs are reconfigurable & easily adaptable ones. And all the carries are carried out by dedicated carry chain. In APEx only 6 bits are directly approximated into constant 1 in the approximate part where as rest 2 bits are computed by using 2- bit LUT adder AAd1.

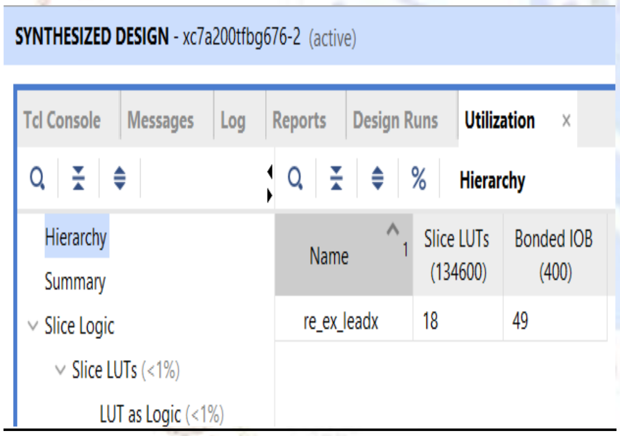
3.2 Proposed Results: -

	AREA(LUT)	POWER(W)	DELAY(ns)
LEADx	18	10.663	6.060
APEx	10	6.32	3.521

Table 2 :- Area, Power, Delay values

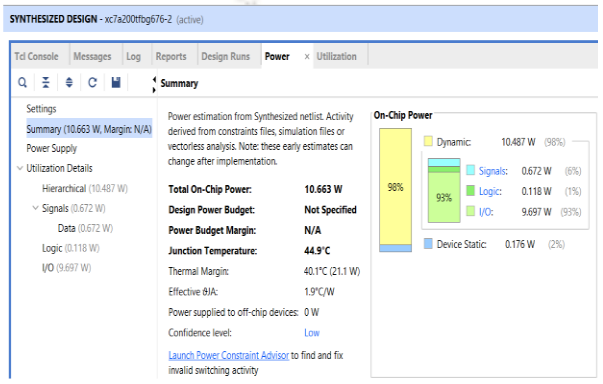
In the proposed method of 8-bit multiplier design, Area, power consumption of LEADx is optimized, Delay value is enhanced compared with existing method where as in APEx Area, Power, Delay all three parameters are enhanced compared with existing method.

LEADx: -


SYNTHESIZED DESIGN - xc7a200tfbg676-2 (active)

Name	Slice LUTs (134600)	Bonded IOB (400)
re_ex_leadx	18	49

Fig 4: Area Utilization


SYNTHESIZED DESIGN - xc7a200tfbg676-2 (active)

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

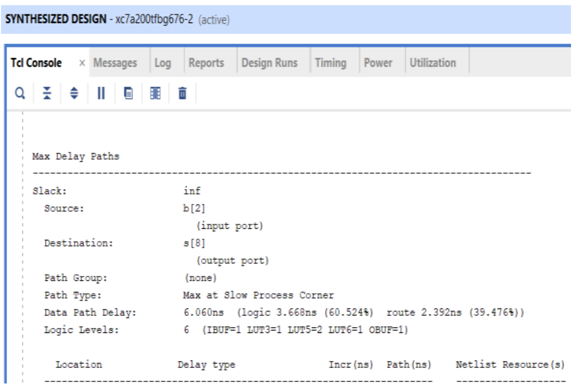
Total On-Chip Power:	10.663 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	44.9°C
Thermal Margin:	40.1°C (21.1 W)
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	10.487 W	(98%)
Signals:	0.672 W	(6%)
Logic:	0.118 W	(1%)
I/O:	9.697 W	(93%)
Device Static:	0.176 W	(2%)

Fig 5: Power Consumption


SYNTHESIZED DESIGN - xc7a200tfbg676-2 (active)

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack:                    inf
  Source:                 b[2]
                            (input port)
  Destination:            s[8]
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        6.060ns  (logic 3.668ns (60.524%)  route 2.392ns (39.476%))
  Logic Levels:           6  (IBUF=1 LUT3=1 LUT5=2 LUT6=1 OBUF=1)

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
```

Fig 6: Maximum delay path

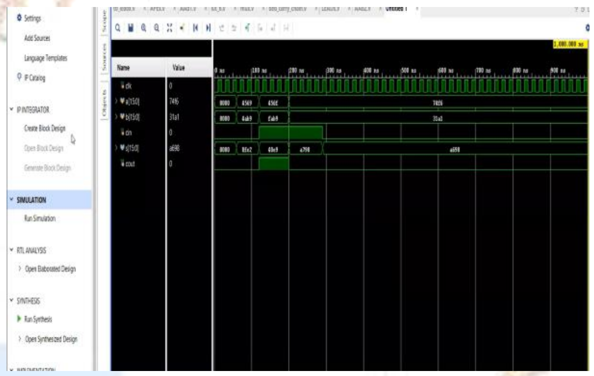


Fig 7: Simulation result of Results of simulation using the suggested approximation adder

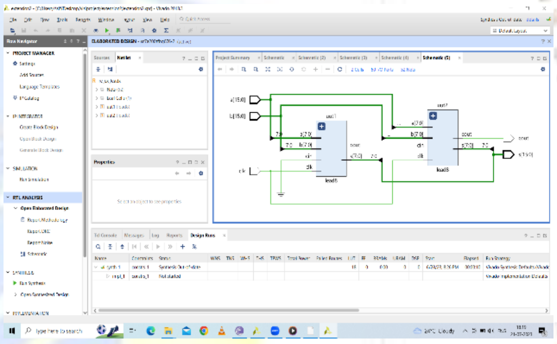
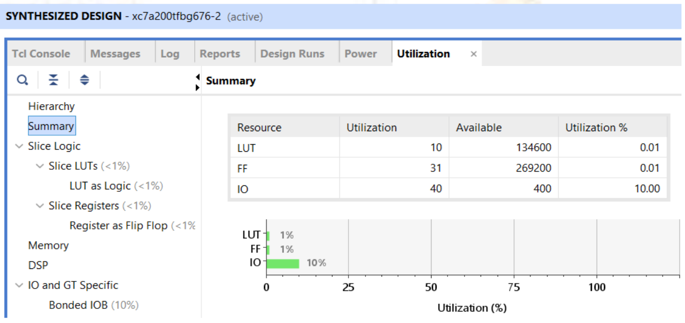


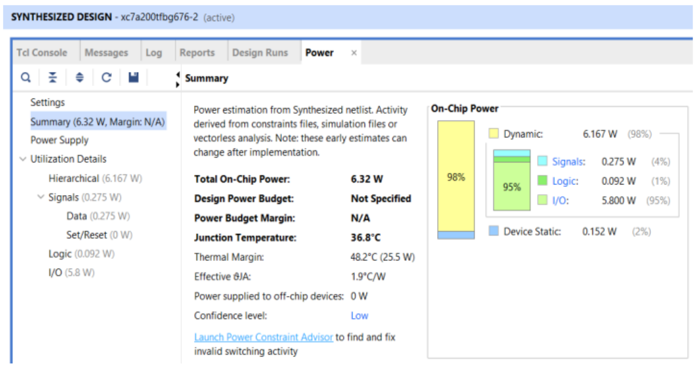
Fig 8: RTL An approximation adder's proposed schematic diagram

APEx: -


SYNTHESIZED DESIGN - xc7a200tfbg676-2 (active)

Resource	Utilization	Available	Utilization %
LUT	10	134600	0.01
FF	31	269200	0.01
IO	40	400	10.00

Fig 9: Area Utilization



SYNTHESIZED DESIGN - xc7a200tfbg676-2 (active)

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

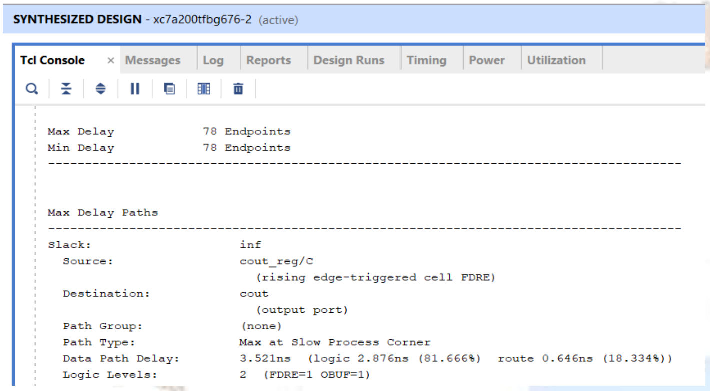
Total On-Chip Power:	**6.32 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**36.8°C**
Thermal Margin:	48.2°C (25.5 W)
Effective θJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	6.167 W	(98%)
🟦	Signals:	0.275 W	(4%)
🟩	Logic:	0.092 W	(1%)
🟩	I/O:	5.800 W	(95%)
🟦	Device Static:	0.152 W	(2%)

Fig 10: Power Consumption



SYNTHESIZED DESIGN - xc7a200tfbg676-2 (active)

```
Max Delay              78 Endpoints
Min Delay              78 Endpoints
------------------------------------------------------------------------------------


Max Delay Paths
------------------------------------------------------------------------------------
Slack:                          inf
  Source:                       cout_reg/C
                                  (rising edge-triggered cell FDRE)
  Destination:                  cout
                                  (output port)
  Path Group:                   (none)
  Path Type:                    Max at Slow Process Corner
  Data Path Delay:              3.521ns  (logic 2.876ns (81.666%)  route 0.646ns (18.334%))
  Logic Levels:                 2  (FDRE=1 OBUF=1)
```

Fig 11: Maximum delay path

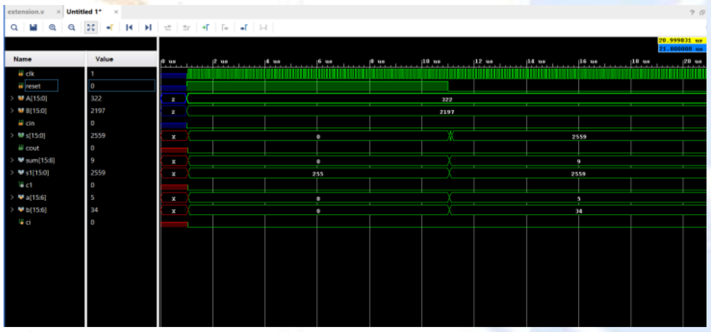


Fig 12: Simulation result of Simulation result of the suggested rough adder

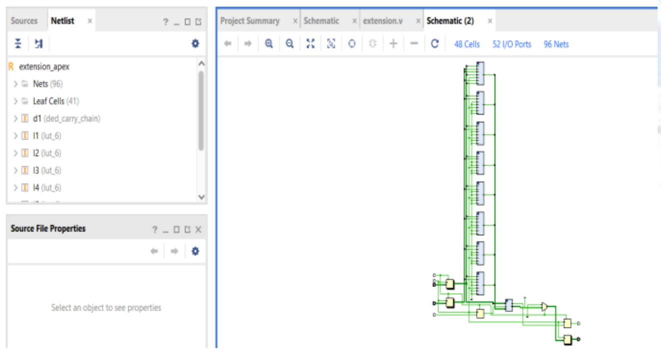


Fig 13: RTL An approximation adder's proposed schematic diagram

CONCLUSION

8-bit multiplier architecture based on 2-bit LUT approximate adders is designed to optimize area, power& delay values. For the fast working of a multiplier delay value should be low. Literature survey of Design of a rough multiplier employing a novel dual stage 4:2 compressor & in High speed improved approximate multiplier where transistors are used which are not reconfigurable ones. Area utilization & delay value is more compared with our proposed method. Above all results are implemented by framing verilog code & is simulated by using Xilinx vivado 2018 tool , Artix 7 AC01 board with speed -1at software computing level in order to design 8-bit multiplier architecture.

REFERENCES:

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in Test Symposium (ETS), 2013 18th IEEE European. IEEE, 2013, pp. 1–6.

[2] K. Nepal, Y. Li, R. Bahar, and S. Reda, "Abacus: A technique for automated behavioural synthesis of approximate computing circuits," in Proceedings of the conference on Design, Automation & Test in Europe. European Design and Automation Association, 2014, p. 361

[3] P. J. Edavoor, S. Raveendran and A. D. Rahulkar, "Approximate Multiplier Design Using Novel Dual-Stage 4:2 Compressors," in IEEE Access, vol. 8, pp. 48337-48351, 2020, doi: 10.1109/ACCESS.2020.2978773.

[4] T. Roshini, R. S. Krishna, P. K. Reddy and M. Vinodhini, "Improved High Speed Approximate Multiplier," 2020 4th International Conference on Electronics, Materials Engineering & Nanotechnology (IEMENTech), Kolkata, India, 2020, pp. 1-5.

[5] S. Venkataramani, K. Roy and A. Raghunathan, "Approximate Computing," 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), Kolkata, India, 2016, pp. 3-4, doi: 10.1109/VLSID.2016.128.

[6] L. H. Krishna, J. B. Rao, S. Ayesha, S. Veeramachaneni, and S. Noor Mahammad, "Energy Efficient. H. Krishna, J. B. Rao, S. Ayesha, S. Veeramachaneni, and S. Noor Mahammad, "Energy Efficient Approximate Multiplier Design for Image/Video Processing Applications," 2021 IEEE International Symposium on Smart Electronic Systems (iSES), Jaipur, India, 2021, pp. 210-215, doi: 10.1109/iSES52644.2021.00056.

[7] X. Wang, L. Wang, Z. Chu and Y. Xia, "Design and evaluation of approximate adders," 2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2020, pp. 201-204, doi: 10.1109/ASID50160.2020.9271703.

[8] X. Wang, L. Wang, Z. Chu and Y. Xia, "Design and evaluation of approximate adders," 2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2020, pp. 201-204, doi: 10.1109/ASID50160.2020.9271703.

[9] N. Vinod et al., "Performance Evaluation of LUTs in FPGA in Different Circuit Topologies," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 1511-1515.

[10] A. Agrawal et al., "Approximate computing: Challenges and opportunities," 2016 IEEE International Conference on Rebooting Computing (ICRC), San Diego, CA, USA, 2016, pp. 1-8, doi: 10.1109/ICRC.2016.7738674.

[11] P. V. Rao and C. P. Raj, "VLSI Design and Analysis of Multipliers for Low Power," 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kyoto, Japan, 2009, pp. 1354-1357, doi: 10.1109/IIH-MSP.2009.129.

[12] S. Sarkar and J. Mehedi, "Comparison of Various Adders and their VLSI Implementation," 2018 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2018, pp. 1-9, doi: 10.1109/ICCCI.2018.8441253.

[13] J. Han, "Introduction to approximate computing," 2016 IEEE 34th VLSI Test Symposium (VTS), Las Vegas, NV, USA, 2016, pp. 1-1, doi: 10.1109/VTS.2016.7477305.

[14] P. Gulati, H. Yadav and M. K. Taleja, "Implementation of an efficient multiplier using the vedic multiplication algorithm,"2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2016,pp.1440-1443,