iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** IV **Month of publication:** April 2024

DOI: https://doi.org/10.22214/ijraset.2024.60615

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue IV Apr 2024- Available at www.ijraset.com

Design and Implementation of FIR Filters for Biomedical Applications

Dr. T Kali Raja[1], B. Bharath Kumar Reddy[2], M. Sunny[3], C. Hemanth[4], C. Bhanu Prakash[5], C. Shyam Pavan Kumar[6]
Electronics and Communication Engineering S V College of Engineering Tirupati, India

Abstract: ***Digital signal processing (DSP) greatly facilitates the continuous capture, monitoring, processing, and analysis of signals in real-world applications, particularly in biomedical or wearable devices. Within DSP systems, the design of Finite Impulse Response (FIR) filters is pivotal. In scenarios demanding intricate calculations and high precision, higher-order filters are employed. Multipliers are crucial components in filter design, consuming significant chip space and introducing extra computation overhead. Designers strive to optimize multipliers to enhance performance.***
In this project we are going to design pipelined based FIR filter by applying pipelining concept to the FIR filter and following the previous implementations such as the FIR filter with high accuracy compared to existing method. The proposed system architectures are known to be efficient for real signal processing, offering advantages over conventional based designs. The proposed design is anticipated to yield benefits in terms of area, power consumption, and timing performance. The synthesis and simulation of the existing and proposed designs are implemented using Cadence Virtuoso.
Keywords: Finite Impulse Response (FIR) filter, Multiply and Accumulate (MAC) unit, Distributed Arithmetic (DA), Look-Up Table (LUT). etc.

I. INTRODUCTION

Multiplication operations are fundamental in various computational tasks, ranging from signal processing to scientific computing. However, these operations often come at a significant cost in terms of area consumption and critical path delay. In many applications, where a certain degree of error tolerance is acceptable, the use of advanced system has emerged as a promising approach to mitigate these challenges.

In this paper, we propose the design of a pipelined-based FIR filter for biomedical application, leveraging the concept of pipelining to enhance performance while maintaining a trade-off between area, timing, and accuracy. Building upon previous implementations, particularly focusing on the development of an FIR filter with superior accuracy compared to existing solutions, our objective is to create a FIR filter capable of meeting the stringent requirements of biomedical applications.

This proposed system offer the flexibility to adjust accuracy, time consumption, and area utilization to achieve higher performance compared to existing system. By incorporating pipelining into our design, we aim to further improve performance by parallelizing computation and reducing critical path delay.

Furthermore, our approach introduces an data aware computation module to meet our requirements. This is one of the important step towards achieving our requirements.

To mitigate the disadvantages that are introduced by existing system, we propose a system that can be helps to enhance area, power and timing efficiency.

Experimental results demonstrate the efficiency of our proposed designs. Compared to existing system our proposed system can achieves significant reductions in parameter values, highlighting its efficiency in terms of resource utilization. Moreover, when compared to non-pipelined FIR filter designs, the performance improvements afforded by our pipelined implementation are evident.

The synthesis and simulation of our proposed designs are facilitated using Cadence Virtuoso, providing a robust platform for development and optimization. Through rigorous synthesis and simulation processes, we validate the effectiveness and feasibility of our proposed approaches, paving the way for their practical implementation in various computational applications.

The organizational framework of this study divides the research work in the different sections. The Literature survey is presented in section 2. In section 3 and 4 discussed about the existing system and proposed system methodologies. Further, in section 5 the Results are discussed and Conclusion is presented by last section 6.

©IJRASET: All Rights are Reserved

II. LITERATURE SURVEY

In 2021, Juthi Farhana Sayed et al.[1] proposed the Design and Evaluation of a FIR Filter Using Hybrid Adders and Vedic Multipliers. In this paper, FIR filter of 45nm technological node has been presented, which is a basic filter in DSP applications. Hybrid Adder has been introduced to improve cost and power consumption of the circuit. A Vedic multiplier and D-type register have also been introduced in the proposed FIR filter. 2-bit 4 tap direct and transposed form of FIR filter have been designed for computational data comparison. Results show that the hybrid adder design has almost 6 times lower power consumption than our conventional Adder using complementary CMOS logic. The smaller number of Delay elements of direct-from FIR further reduces power consumption, area and transistor numbers. Therefore, by using our circuits, the overall performance and power consumption of FIR filter has been improved significantly. To implement the circuits, DSCH software has been used and to design the layout, Microwind has been used.

In 2020, Swathi Dayanand et al.[2] proposed Low Power High Speed Vedic Techniques in Recent VLSI Design — A Survey. Advancement in the Artificial Intelligence (AI) and Machine Learning (ML) has influenced complex designs to be integrated in Very Large-Scale Integration (VLSI) Design. Designers are concentrating on high speed and low power techniques to facilitate the needs of the technology requirements. In multiple AI applications, Digital Signal Processor is the building block, optimization of it may solve the issues related to computation of the data signal at faster rate consuming less power using Vedic mathematics. In this paper, a detailed review is made on recent applications of Vedic Mathematics in the domain of VLSI to yield novel design, efficient architecture for Squarer, Multiplier, Arithmetic unit, Cubic and divider circuits along with their crucial performance criteria. It is deduced that the use of Vedic Sutras in formulating algorithms for digital logic circuit design has led to simplified architecture and yielded higher speed, low power consumption and enhanced efficiency of operation.

In 2020, M. Bharathi et al.[3] proposed Performance evaluation of Distributed Arithmetic based MAC Structures for DSP Applications MAC is an essential core which is used in every Digital signal processor. The primary focal point of this article is to introduce a high performance Distributed based (DA) MAC and offset binary coding Distributed Arithmetic (DA) based MAC for real time Signal Processing Applications. Addition and multiplication are the two hardware resources widely used to design any arithmetic blocks in many fields like video processing, audio processing, speech processing and medical image processing applications. In this article, a literature survey is done on different MAC [2] units with different multipliers to generate partial products and to perform accumulation.

Developed a DA based and offset binary coding DA based MAC cores which offers greater speed compared with different conventional MAC's using various multipliers. The coding for DA and offset based architectures are done using Verilog and simulation, synthesis are performed in Xilinx 14.7 Integrated Simulation Environment version. It achieves best area and less delay result when compared with previous approximate adder designs. The results of DA based MAC cores gives much more efficient in delay whereas offset binary coding-based DA offers both speed and area optimization.

In 2013, Ashish B. Kharate et al.[4] proposed VLSI Design and Implementation of Low Power MAC for Digital FIR Filter. In the majority of digital signal processing (DSP) applications the critical operations are the multiplication and accumulation. Multiplier-Accumulator (MAC) unit that consumes low power is always a key to achieve a high performance digital signal processing system. Finite impulse response (FIR) filters are widely used in various DSP applications. The purpose of this work is to design and implementation of Finite impulse response (FIR) filter using a low power MAC unit with clock gating and pipelining techniques to save power.

In 2011, Narendra singh Pal et al.[5] proposed Implementation of High Speed FIR Filter using Serial and Parallel Distributed Arithmetic Algorithm. This paper describes the implementation of highly efficient multiplier less serial and parallel distributed arithmetic algorithm for FIR filters. Distributed Arithmetic (DA) had been used to implement a bit-serial scheme of a general symmetric version of an FIR filter due to its high stability and linearity by taking optimal advantage of the look-up table (LUT) based structure of FPGAs. The performance of the bit-serial and bit-parallel DA technique for FIR filter design is analyzed and the results are compared to the conventional FIR filter design techniques. The proposed algorithm has been synthesized with Xilinx ISE 10.1i and implemented as a target device of Spartan3E FPGA.

III. EXISTING SYSTEM

In Very Large-Scale Integration (VLSI), the most popular bits' serial approach is Distributed Arithmetic (DA). In this approach the values are calculated in advance based on the number of address bits and stored in the database in the form of a lookup table (LUT) and these values are considered for the computation instead of resultant values from the multiplication.

©IJRASET: All Rights are Reserved

The DA based design implementation on field programmable gate arrays (FPGAs) is easier and takes less computational time, but they are hindering due to excessive utilization of area when the bits increase.

This is the one of the main deciding factors to choose between the multiplier-based MAC or multiplier-less MAC. The entered coefficients are numbered concurrently and are used as input bits to the LUT; its end result is added to the gathered partial products.

To calculate the dot product, it makes use of N clocks where in N is the range of bits entered and is independent of the range of the entered variables. Figure 1 denotes the LUT based DA design architecture.

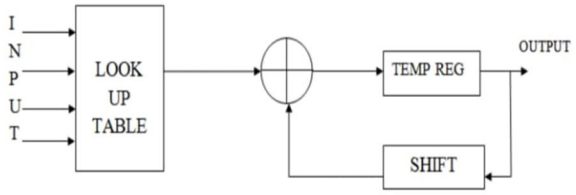


Fig 1: Architecture of LUT based DA design

The flow chart shown in figure 2 gives the design implementation methodology followed for getting the meaningful results which are discussed in the later part of the paper. Any new design will start with the detailed design specifications which will give clarity about the algorithm used, number of sub modules present in it, the complexity of the design, hardware and software requirement and no. of input and output signals and description of the signals.

It also figures out the control and interfacing unit requirement which decides the capability of the design to enhance or expand on need basis. The design is coded in Verilog starting from the low-level going up in the direction to integrate them at high level which gives the required results as per the application requirements.

The verification is performed at every stage of the design flow to check the functionality. Performance reviews are generated with the use of EDA tools and comparison is made between the proposed design and the conventional design to conclude which is better for the chosen application.

DA uses Lookup Tables (LUTs) to store the multiplication results. The Look-Up Table may be designed as per table 2. The existing system has low-level modules such as rom_rtl, adder, PIPO and PISO. They are used to perform operations as shown in table 3.

The table 4 details of the input/output signals of the design with number of bits, directions, and description.

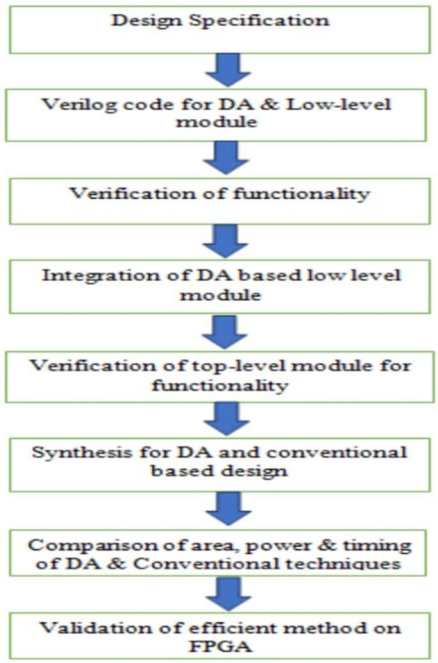


Fig 2: Step-by-Step Design Flow carried out for existing design implementation

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Sl. No.	Address	Data	Sl. No.	Address	Data
1	0000	0	9	1000	b_0
2	0001	b_3	10	1001	b_0+b_3
3	0010	b_2	11	1010	b_0+b_2
4	0011	b_2+b_3	12	1011	$b_0+b_2+b_3$
5	0100	b_1	13	1100	b_0+b_1
6	0101	b_1+b_3	14	1101	$b_0+b_1+b_3$
7	0110	b_1+b_2	15	1110	$b_0+b_1+b_2$
8	0111	$b_1+b_2+b_3$	16	1111	$b_0+b+b_2+b_3$

Table 1: Look Up Table

Sl. No.	Signal	Width	Direction	Description
1.	A_0-A_3	4-bits	Input	Co-efficient of the FIR filter
2.	ADDR	4-bits	Input	Input signal to the FIR filter
3.	clken	1-bit	Input	Clock and clock enable for synchronization
4.	Clk	-	Input	
5.	Q	16-bits	Output	Output of the 4 Tap FIR Filter using MAC Unit

Table 2: Input and Output

Module Name	Operation Performed
rom_rtl	Fetch the data from Look Up Table
Adders	Adds input DATA, e and cin
PISO	Sends data bit-by-bit
PIPO	Accumulate and send filter output

Table 3: Details of Module Name and Operation Performed

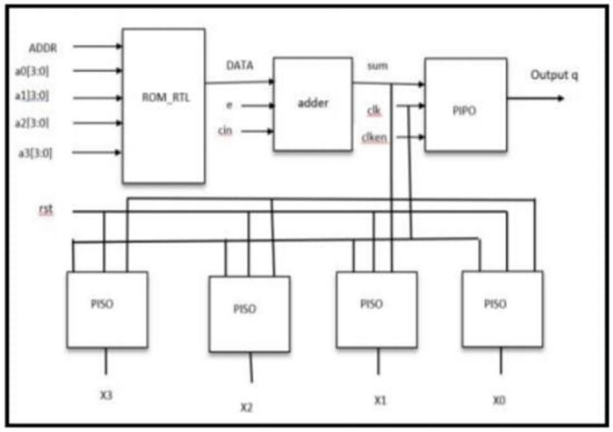


Fig 3: Architecture of existing system

IV. PROPOSED SYSTEM

In the proposed design four order filter is considered, it will have four coefficients. The inputs considered are X and Y are of 8-bits in size and output is 16-bits wide. The design accommodates the non-positive numbers as well, in two's complement format and output changes accordingly to the input subjected to the design. The structural design architecture of FIR filter is as depicted in figure.

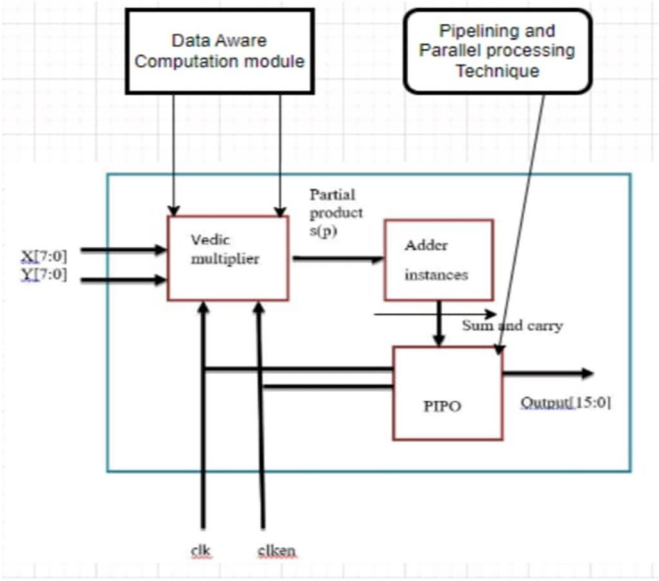


Fig 4: Architecture of proposed system

A data-aware computation module is a component within a system or algorithm that operates while being cognizant of the characteristics and structure of the data it processes. This module intelligently adapts its computation strategies based on the data it encounters, aiming to optimize performance, efficiency, and accuracy. Overall, a data-aware computation module combines adaptability, efficiency, and intelligence to perform computations optimally in various data-driven applications and scenarios.

Its ability to understand, adapt to, and leverage data characteristics makes it a crucial component in modern computing systems and algorithms. Pipelining and parallel processing are two techniques used in computer architecture and system design to improve performance and efficiency by overlapping and executing multiple tasks simultaneously. While they share the goal of enhancing throughput and reducing latency, they operate in different ways.

Pipelining is a technique where multiple tasks are divided into smaller stages, and each stage is executed concurrently.

The input data flows through the pipeline, with each stage processing a different part of the data simultaneously.

Once a stage completes its operation on one piece of data, it passes that data to the next stage without waiting for the entire process to finish. Pipelining reduces the overall processing time by overlapping the execution of different stages, effectively increasing throughput.

Common examples of pipelining include instruction pipelines in processors, where instructions are divided into fetch, decode, execute, and write-back stages, each operating concurrently on different instructions. Parallel processing involves the simultaneous execution of multiple tasks or computations using multiple processing units. Unlike pipelining, parallel processing executes independent tasks concurrently rather than breaking a single task into smaller stages.

Parallel processing can be achieved using multiple processors, cores within a processor, or specialized hardware accelerators. Tasks are distributed among the available processing units, and each unit operates independently on its assigned task.

Parallel processing offers significant performance gains for tasks that can be divided into parallelizable subtasks, such as large-scale simulations, data analytics, image processing, and scientific computations.

However, parallel processing may require synchronization and coordination mechanisms to manage data dependencies and ensure correct execution.

In summary, pipelining breaks a single task into smaller stages executed concurrently, while parallel processing involves executing multiple independent tasks simultaneously using multiple processing units. Both techniques aim to improve performance and efficiency by leveraging concurrency and overlapping execution.

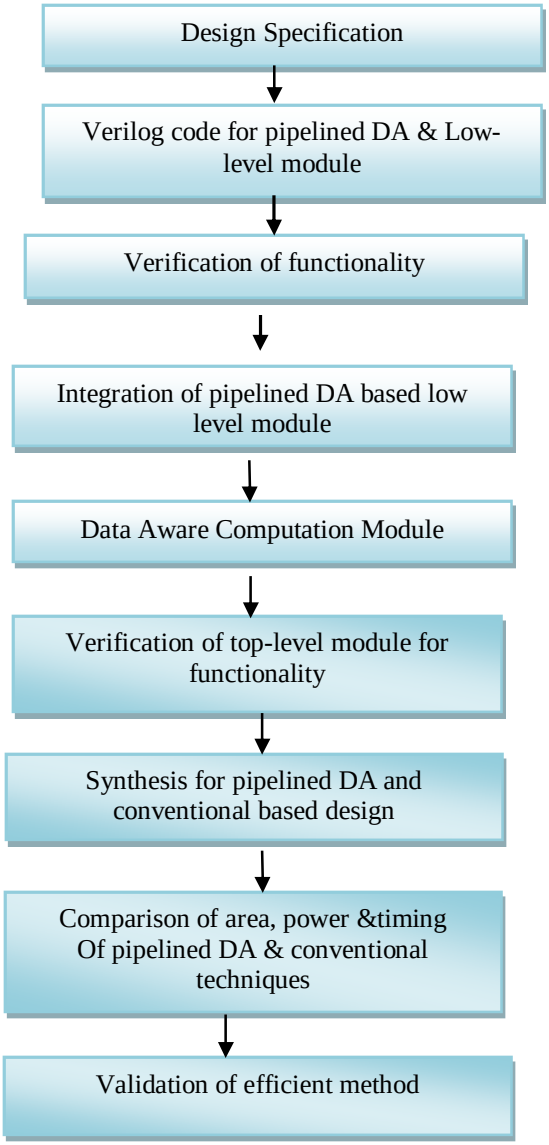


Fig 5: Flow chart of proposed system

V. RESULTS AND DISCUSSION

1) RTL schematic

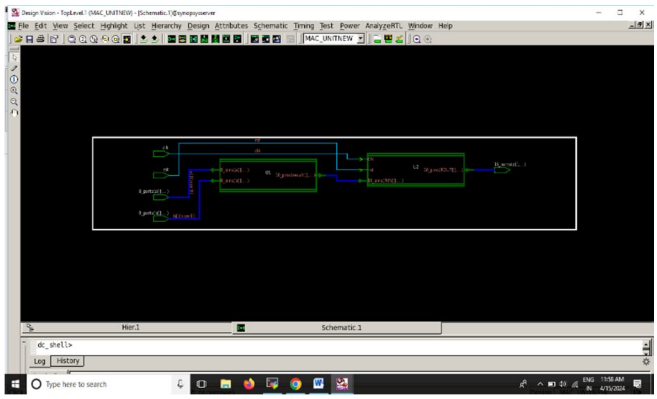


Fig 6: RTL Schematic 1 of Proposed System

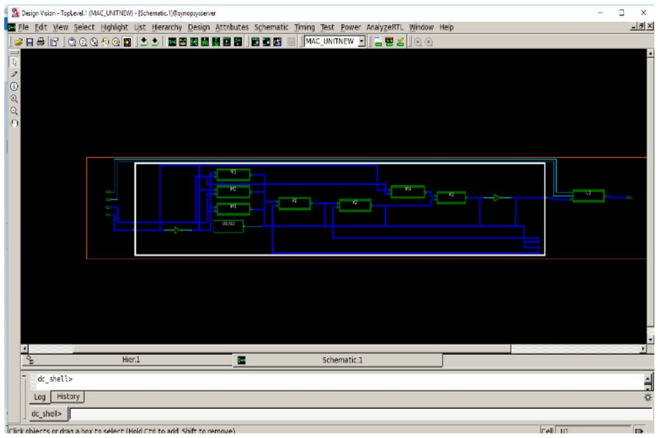


Fig 7: RTL schematic 2 of Proposed System

2) Area:

```
Placed design
 Design Summary:
Total Standard Cell Number   (cells) : 82
Total Block Cell Number      (cells) : 0
Total I/O Pad Cell Number    (cells) : 0
Total Standard Cell Area     ( nm^2) : 202.08
Total Block Cell Area        ( nm^2) : 0.00
Total I/O Pad Cell Area      ( nm^2) : 0.00
```

Fig 8: Showing Area of Proposed System

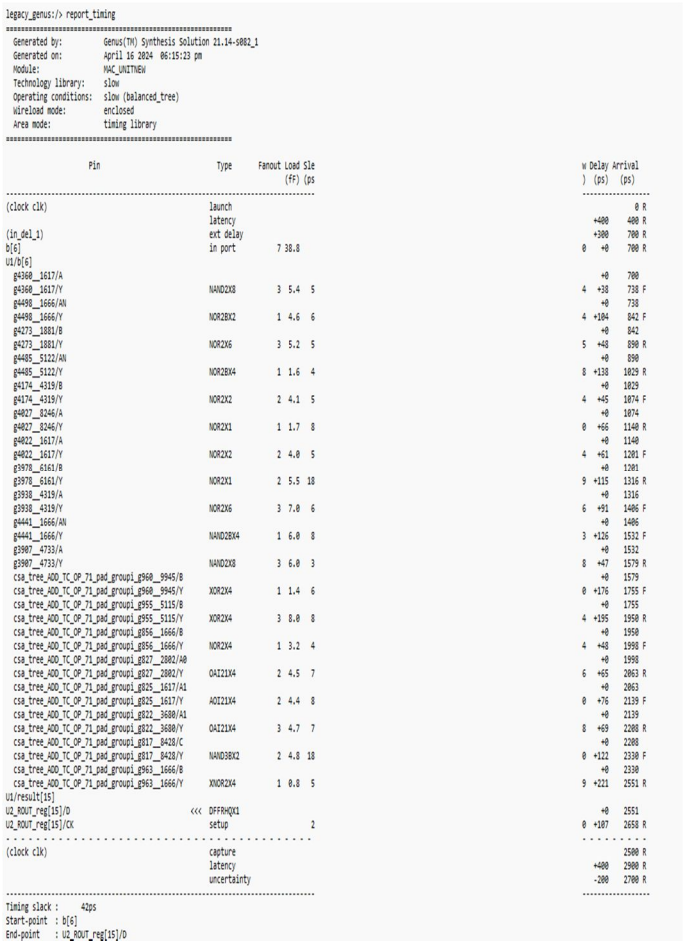
3) Power:

```
Power Report


Instance: /MAC_UNITNEW
Power Unit: W
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------
    Category        Leakage     Internal    Switching       Total    Row%
  ------------------------------------------------------------------------
       memory    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     register    4.67494e-09  1.01743e-08  0.00000e+00  1.01748e-08  15.31%
        latch    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        logic    1.35710e-07  3.67754e-08  1.90544e-08  5.58433e-08  84.02%
         bbox    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        clock    0.00000e+00  0.00000e+00  4.47898e-08  4.47898e-08   0.67%
          pad    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
           pm    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  ------------------------------------------------------------------------
     Subtotal    1.40385e-07  4.69497e-08  1.95023e-08  6.64660e-08 100.00%
   Percentage          0.02%       70.64%       29.34%      100.00% 100.00%
```

Fig 9: Showing Power of Proposed System

4) Timing:



```
legacy_genus:/> report_timing
============================================================
  Generated by:           Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:           April 16 2024  06:15:23 pm
  Module:                 MAC_UNITNEW
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                    Pin                        Type       Fanout Load Slew                                    w Delay Arrival
                                                           (fF) (ps)                                          ) (ps)  (ps)
-------------------------------------------------------------------------------------------------------------------------
(clock clk)                                    launch                                                                0 R
                                               latency                                                            +400   400 R
(in_del_1)                                     ext delay                                                          +300   700 R
b[6]                                           in port         7 38.8                                           0   +0   700 R
U1/b[6]
  g4360__1617/A
  g4360__1617/Y                                NAND2X8         3  5.4   5                                       4  +38   738 F
  g4498__1666/AN                                                                                                   +0   738
  g4498__1666/Y                                NOR2BX2         1  4.6   6                                       4 +104   842 F
  g4273__1881/B                                                                                                    +0   842
  g4273__1881/Y                                NOR2X6          3  5.2   5                                       5  +48   890 R
  g4485__5122/AN                                                                                                   +0   890
  g4485__5122/Y                                NOR2BX4         1  1.6   4                                       8 +138  1029 R
  g4174__4319/B                                                                                                    +0  1029
  g4174__4319/Y                                NOR2X2          2  4.1   5                                       4  +45  1074 F
  g4027__8246/A                                                                                                    +0  1074
  g4027__8246/Y                                NOR2X1          1  1.7   8                                       0  +66  1140 R
  g4022__1617/A                                                                                                    +0  1140
  g4022__1617/Y                                NOR2X2          2  4.0   5                                       4  +61  1201 F
  g3978__6161/B                                                                                                    +0  1201
  g3978__6161/Y                                NOR2X1          2  5.5  18                                       9 +115  1316 R
  g3938__4319/A                                                                                                    +0  1316
  g3938__4319/Y                                NOR2X6          3  7.0   6                                       6  +91  1406 F
  g4441__1666/AN                                                                                                   +0  1406
  g4441__1666/Y                                NAND2BX4        1  6.0   8                                       3 +126  1532 F
  g3907__4733/A                                                                                                    +0  1532
  g3907__4733/Y                                NAND2X8         3  6.0   3                                       8  +47  1579 R
  csa_tree_ADD_TC_OP_71_pad_groupi_g960__9945/B                                                                    +0  1579
  csa_tree_ADD_TC_OP_71_pad_groupi_g960__9945/Y XOR2X4         1  1.4   6                                       0 +176  1755 F
  csa_tree_ADD_TC_OP_71_pad_groupi_g955__5115/B                                                                    +0  1755
  csa_tree_ADD_TC_OP_71_pad_groupi_g955__5115/Y XOR2X4         3  8.0   8                                       4 +195  1950 R
  csa_tree_ADD_TC_OP_71_pad_groupi_g856__1666/B                                                                    +0  1950
  csa_tree_ADD_TC_OP_71_pad_groupi_g856__1666/Y NOR2X4         1  3.2   4                                       4  +48  1998 F
  csa_tree_ADD_TC_OP_71_pad_groupi_g827__2802/A0                                                                   +0  1998
  csa_tree_ADD_TC_OP_71_pad_groupi_g827__2802/Y OAI21X4        2  4.5   7                                       6  +65  2063 R
  csa_tree_ADD_TC_OP_71_pad_groupi_g825__1617/A1                                                                   +0  2063
  csa_tree_ADD_TC_OP_71_pad_groupi_g825__1617/Y AOI21X4        2  4.4   8                                       0  +76  2139 F
  csa_tree_ADD_TC_OP_71_pad_groupi_g822__3680/A1                                                                   +0  2139
  csa_tree_ADD_TC_OP_71_pad_groupi_g822__3680/Y OAI21X4        3  4.7   7                                       8  +69  2208 R
  csa_tree_ADD_TC_OP_71_pad_groupi_g817__8428/C                                                                    +0  2208
  csa_tree_ADD_TC_OP_71_pad_groupi_g817__8428/Y NAND3BX2       2  4.8  18                                       0 +122  2330 F
  csa_tree_ADD_TC_OP_71_pad_groupi_g963__1666/B                                                                    +0  2330
  csa_tree_ADD_TC_OP_71_pad_groupi_g963__1666/Y XNOR2X4        1  0.8   5                                       9 +221  2551 R
U1/result[15]
U2_ROUT_reg[15]/D                          <<<  DFFRHQX1                                                             +0  2551
U2_ROUT_reg[15]/CK                              setup                 2                                          0 +107  2658 R
-------------------------------------------------------------------------------------------------------------------------
(clock clk)                                    capture                                                           2500 R
                                               latency                                                           +400  2900 R
                                               uncertainty                                                       -200  2700 R
-------------------------------------------------------------------------------------------------------------------------
Timing slack :    42ps
Start-point  : b[6]
End-point    : U2_ROUT_reg[15]/D
```

Fig 10: Showing timing of proposed system

5) Comparison Table

S.No	Parameter	Existing Method	Proposed Method
1	Area (in nm^2)	386.1	202
2	Power (in Watts)	8.22E-09	6.64E-08
3	Timing (in ps)	6210	2700

This comparison table evaluates the performance of the existing and proposed systems for implementing FIR filter.
The parameters compared are area, power and timing consumption. Let's discuss the meaning of each parameter and the comparison between the existing system and proposed system.

a) Area (in nm^2*):* This parameter measures the amount of space required for implementing the FIR filter. It is typically measured in nm^2. The existing system takes more area than proposed system. This suggests that the area efficiency is more in proposed system.

b) Power (in W): Power consumption measures the amount of electrical power consumed by the FIR filter during its operation. It is typically measured in watts (W). Despite the slight increase in power consumption, the proposed method's advantage in terms of timing might outweigh this increase, depending on the specific application's requirements.

c) *Timing (in ps):* This is the time it takes for the FIR filter to produce an output after receiving input. It is measured in picoseconds (ps). The existing system having more timing than the proposed system. This indicates that the proposed system offers improved speed performance compared to the existing system.

Overall, the proposed system seems to offer faster performance with comparable area efficiency but slightly higher power consumption compared to the existing system. The choice between the two systems would depend on the specific priorities of the design project, such as speed, power efficiency, or resource utilization etc.

VI. CONCLUSION

In conclusion, the proposed FIR filter design incorporating Data Aware Modules (DAMs) and pipelining techniques represents a significant advancement in digital signal processing. This innovative approach holds great potential for various applications, including communications, audio processing, and biomedical signal analysis, where real- time processing and high-performance filtering are paramount.

By addressing the challenges of traditional FIR filter implementations, this proposal opens new avenues for advanced signal processing systems capable of meeting the demands of modern data-intensive applications. With further research and development, the integration of DAMs and pipelining techniques into FIR filter designs promises to revolutionize digital signal processing, unlocking new capabilities and performance benchmarks previously thought unattainable.

VII. FUTURE SCOPE

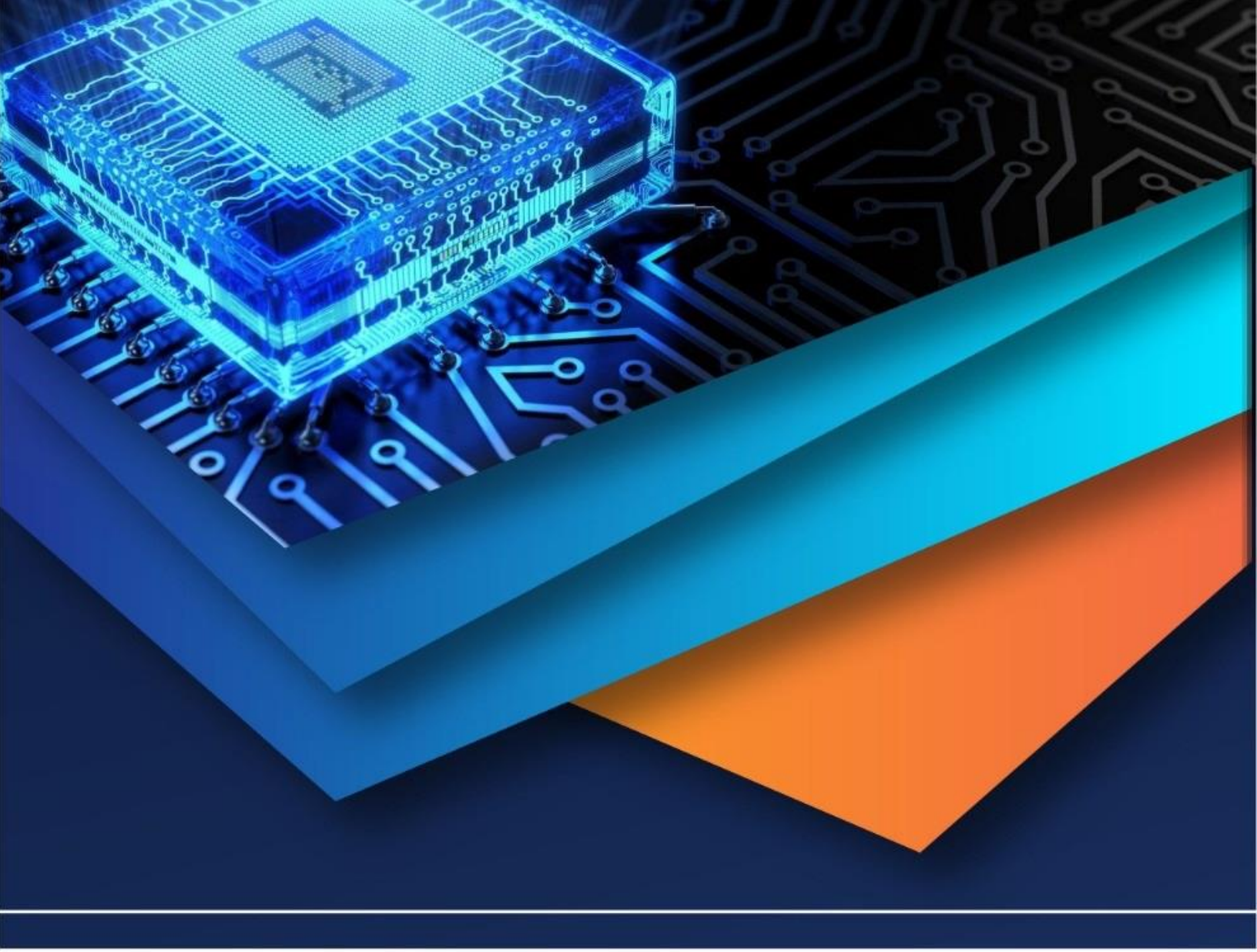
The proposed system can be extended with further exploration of optimization techniques that can refine the proposed FIR filter design. This can be consist of using advanced blocks to achieve better performance metrics such as area, power consumption, and speed.

VIII. ACKNOWLEDGMENT

It gives us great pleasure in presenting the preliminary project report on. I would like to take this opportunity to thank my guide Dr. T. Kali Raja, Ph.D., Professor and Dr. D. Srinivasulu Reddy, Ph.D., Professor, & Head of the Department (HOD) of Electronics and Communication Engineering, S V College of Engineering, Tirupati, Andhra Pradesh, India, for giving me all the help and guidance I needed. I am really grateful for their kind support and valuable suggestions were very helpful. Thank you all!

REFERENCES

[1] N. Pal, H. Singh, R. Sarin, and S. Singh, “Implementation of High-Speed FIR Filter using Serial and Parallel Distributed Arithmetic Algorithm,” International Journal of Computer Applications, vol. 25, 07 2011.
[2] Bharathi, M., Shirur, Y.J., & Lahari, P.L. (2020). Performance evaluation of Distributed Arithmetic based MAC Structures for DSP Applications. 2020 7th International Conference on Smart Structures and Systems (ICSSS), 1-5.
[3] S. Dayanand, V. K R, R. T, Y. J. M. Shirur, and J. R. Munavalli, “Low Power High Speed Vedic Techniques in Recent VLSI Design — A Survey”, pices, vol. 4, no. 6, pp. 147-156, Oct. 2020.
[4] Ashish B. Kharate and Prof. P.R. Gumble, “VLSI Design and Implementation of Low Power MAC for Digital FIR Filter”, International Journal of Electronics Communication and Computer Engineering Volume 4, Issue (2) REACT-2013, ISSN 2249—071X. June 2013, PP 604 — 605.
[5] Juthi Farhana Sayed, Bhuiyan Hasibul Hasan, Babul Muntasir, Mehedi Hasan, Farhadur Arifin, “Design and Evaluation of a FIR Filter Using Hybrid Adders and Vedic Multipliers”, 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST).
[6] D. Maskell, “Design of efficient multiplier less FIR filters,” IET Circuits, Devices & Systems, vol. 1, pp. 175—180(5), 2007.
[7] Haw-Jing Lo, “Distributed Arithmetic” in Design of a reusable Distributed Arithmetic Filter and Its Application to The Affine Projection Algorithm” Georgia, UMI Microform, 2009, PP 3-12.
[8] Xiumin Wang, “Implementation of FIR Filter on FPGA Using DAOBC Algorithm“ , in IEEE,2010.
[9] Shunwen Xiao,Yajun chen, "The design of FIR filter based on improved DA algorithm and its FPGA implementation ",IEEE International conference on computer and automation engineering(ICCAE’10), 2010,Vol.2,PP.589-591.
[10] New Approach to Look-up-Table Design and Memory-Based Realization of FIR Digital Filter Pramod Kumar Meher, Senior Member, IEEE.
[11] CHALLIS, R. E. andKITNEY, R. I. (1982) The design of digital filters for biomedical signal processing (in three parts). Part 1.J. Biomed. Eng.,**4**, 267–278.
[12] CHALLIS, R. E. andKITNEY, R. I. (1983a) The design of digital filters for biomedical signal processing (in three parts). Part 2. —Ibid.,**5**, 19–30.
[13] CHALLIS, R. E. andKITNEY, R. I. (1983b) The design of digital filters for biomedical signal processing (in three parts). Part 3. —Ibid.,**5**, 91–102.
[14] Preeti D, Sasikala M, Gnana Prakash V, Satyasri B, Archana V “Performance Analysis of FIR Filters using High Speed VLSI Adders for FECG monitoring” 2023 9th international conference on smart structures and systems (ICSSS).
[15] Design & implementation of FPGA based digital filters Ankit Jairath Department of Electronics & Communication, Gyan Ganga Institute of Technology and Sciences, Jabalpur (M.P) Issue 7, September 2012 199 All Rights Reserved © 2012 IJARCE.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)