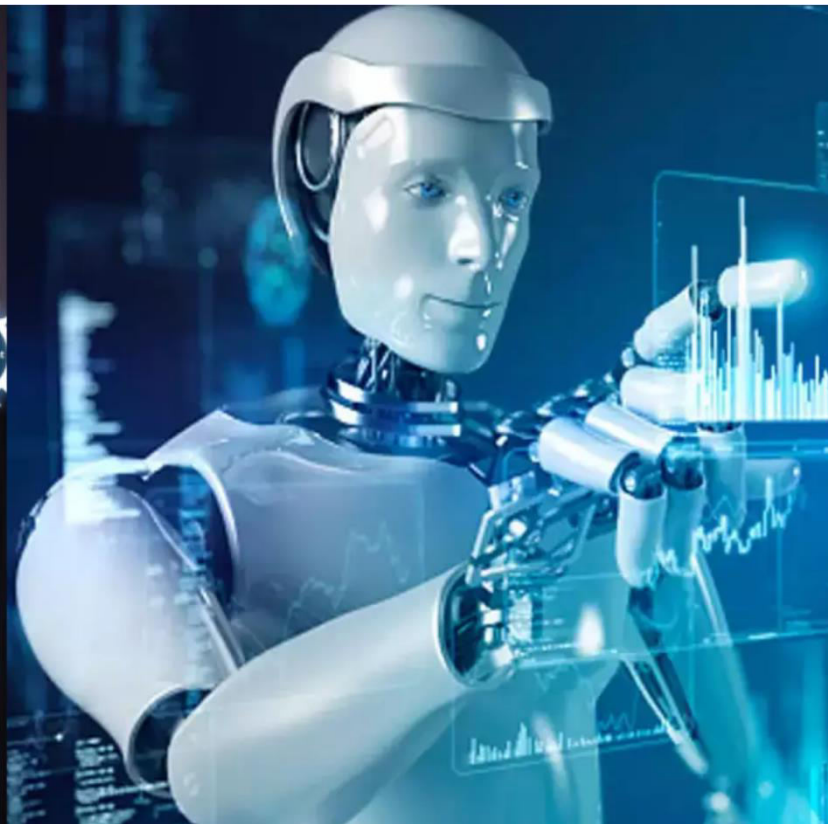




International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 6, June 2025

VLSI Architecture for Parallel FIR Filter based on Brent Kung Adder and Computational Sharing Multiplier

Sonam Shukla¹, Prof. Aditya Mishra²

VLSI M. Tech. Scholar, Department of Electronics and Communication Engineering, Vidhyapeeth Institute of Science
and Technology, Bhopal, M.P., India¹

Assistant Professor, Department of Electronics and Communication Engineering Vidhyapeeth Institute of Science and
Technology, Bhopal, M.P., India²

ABSTRACT: Finite Impulse Response (FIR) filters are critical components in modern digital signal processing (DSP) systems due to their inherent stability and linear phase characteristics. However, implementing FIR filters efficiently in hardware remains a challenge due to the high computational complexity and resource utilization. This paper presents a novel VLSI architecture for a parallel FIR filter design that integrates the Brent-Kung adder and the Computational Sharing Multiplier (CSM) technique to enhance performance and area efficiency. The Brent-Kung adder, known for its logarithmic delay and minimal fan-out, reduces the latency and power consumption in accumulation stages. Simultaneously, the CSM technique optimizes multiplication operations by reusing partial product computations, thereby reducing redundant operations and lowering power and area overhead. The proposed architecture is implemented using Verilog HDL and synthesized using Xilinx/Virtex FPGA tools. Simulation results demonstrate significant improvements in delay, power consumption, and area utilization when compared to traditional FIR filter implementations using ripple carry adders and conventional multipliers. The design is particularly suitable for real-time DSP applications such as audio processing, image enhancement, and wireless communications. This research contributes a scalable and high-speed FIR filter architecture, paving the way for more efficient DSP hardware solutions.

KEYWORDS: Brent Kung Adder, Parallel FIR Filter, Delay, Slice Flip Flop

I. INTRODUCTION

In the modern era of high-speed digital communication and real-time signal processing, the demand for efficient hardware implementation of digital filters has significantly increased. Finite Impulse Response (FIR) filters are widely used in a variety of digital signal processing (DSP) applications due to their inherent stability, linear phase response, and ease of implementation. These filters are fundamental in systems such as audio processing, wireless communication, biomedical signal analysis, and image enhancement [1, 2]. However, the computational complexity associated with higher-order FIR filters, particularly in terms of multiplications and additions, poses a major challenge for hardware designers. This complexity becomes even more critical when the filter must operate at high speed with low power and minimal area consumption — the core goals of Very-Large-Scale Integration (VLSI) design [3].

Traditionally, FIR filters are implemented using basic multipliers and ripple-carry adders, which, although simple in design, are not optimal for performance in VLSI circuits. These conventional arithmetic components suffer from long propagation delays and high switching activities, which directly affect the speed and power efficiency of the system. To address these limitations, the present research focuses on an advanced VLSI architecture for parallel FIR filter implementation that utilizes two efficient components: the Brent-Kung adder and the Computational Sharing Multiplier (CSM) [4].

The Brent-Kung adder is a type of parallel prefix adder known for its logarithmic delay and minimal fan-out, making it highly suitable for high-speed VLSI systems. Unlike ripple-carry or carry-lookahead adders, the Brent-Kung adder achieves faster addition by using a tree-structured approach to compute carry bits, resulting in lower delay and improved throughput. This feature is particularly advantageous in FIR filters, where multiple additions are performed in

every filtering operation. Integrating the Brent-Kung adder into the filter architecture substantially improves the timing performance and overall speed of computation [5, 6].

In parallel, the Computational Sharing Multiplier (CSM) is employed to reduce the redundant partial product computations in multiplication operations. In conventional FIR filter designs, multipliers contribute significantly to the area and power consumption of the circuit. The CSM technique identifies common sub-operations among different multiplication steps and reuses them, thereby minimizing the hardware complexity and reducing dynamic power dissipation. This optimization is especially effective in fixed-coefficient FIR filters, where the constant coefficients allow pre-computation and reuse of results.

Combining these two optimized components — the Brent-Kung adder for addition and the CSM for multiplication — enables the design of a high-speed, area-efficient, and low-power FIR filter architecture. Additionally, implementing the design in a parallel configuration allows multiple output samples to be computed simultaneously, further increasing the throughput of the filter. This parallel processing approach is particularly beneficial in real-time DSP applications that demand rapid data processing without sacrificing accuracy [7].

The proposed architecture is implemented using hardware description languages (HDL) like Verilog and synthesized on FPGA platforms such as Xilinx or Altera. The simulation and synthesis results demonstrate considerable improvements in performance metrics such as propagation delay, area utilization, and power consumption when compared to traditional FIR filter implementations. Thus, the integration of the Brent-Kung adder and CSM within a parallel FIR filter framework offers a promising solution for next-generation high-performance DSP systems [8, 9].

II. FIR FILTER

The term Finite Impulse Response (FIR) filter arises the filter output is computed as a weighted sum, finite term sum of past, present and perhaps future values of the filter input. The output of FIR filter is represented in Punskeya (2005) as,

$$y(n) = \sum_{k=-M_1}^{M_2} b_k x(n-k) \quad (1)$$

Where, both M_1 and M_2 are finite.

In one of the simplest FIR filters, we may consider 3-term moving average filter of the form,

$$y(n) = \frac{1}{3} [x(n-1) + x(n) + x(n+1)] \quad (2)$$

As, mentioned in early section, FIR filters consist only feedforward term. This is illustrated in Equation (2). Feedforward means, the present output does not depends on previous outputs and also there is no feedback of past or future outputs. It means that the output of FIR filter just contain only input related terms. The FIR filter is also termed as recursive filter. The literal meaning for recursive is running back and technically refers, the previously calculated outputs are going back to the latest output. The every stage of FIR filter uses previous output values in addition to input values for processing filtering operation. Hence, it is called as recursive filter. Therefore, in FIR filter, previous input values are stored in the processors memory. From above consecution, it is clear that more calculation is required to perform the process of FIR filter, since the filter expression consist of previous output terms as well as present input terms. There are two types of FIR filters widely used. One is direct form FIR filter and other is transposed FIR filter. The structural architecture of direct form FIR filter is shown in Figure 1. This structure consists of delay unit, adder and multiplier units to perform filtering operations.

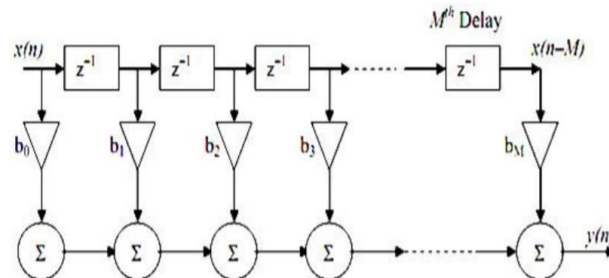


Figure 1: Direct Form FIR filter

III. METHODOLOGY

Computational Sharing Multiplier

We consider two N-bit operands $x_{N-1}, x_{N-2}, \dots, x_2, x_1, x_0$ and $y_{N-1}, y_{N-2}, \dots, y_2, y_1, y_0$ for N by N multiplier, the partial products of two N-bit numbers are $x_i y_j$ where i, j go from 0,1,..N-1. The longest two columns in the middle of the partial products contribute to the maximum delay. We then proceed to sum up each column of the two parts in parallel.

Partition Multiplier Method: - Let M and N be numbers of b-bits such that multiplication of the inputs M and N gives the 2b bits product. Here M and N are split into t numbers as $M_0, M_1, M_2, \dots, M_{t-1}$ and

$N_0, N_1, N_2, \dots, N_{t-1}$ each consisting of r bits

Where

$$t = \frac{b}{r} \quad (3)$$

Consider the value of b = 8, r = 2

Then

$$t = 4 \quad (4)$$

M and N can therefore be expressed as

$$M_0 = M\left(\frac{n}{2} - 1 \text{ to } 0\right) \quad (5)$$

$$M_1 = M\left(n - 1 \text{ to } \frac{n}{2}\right) \quad (6)$$

$$N_0 = N\left(\frac{n}{2} - 1 \text{ to } 0\right) \quad (7)$$

$$N_1 = N\left(n - 1 \text{ to } \frac{n}{2}\right) \quad (8)$$

a. Brent Kung Adder

The Brent-Kung adder is a parallel prefix adder (PPA) and is modified form of carry-look ahead adder (CLA). Proposed by Richard Peirce Brent and Hsiang Te Kung in 1982 it acquainted higher normality with the structure of the adder and has less wiring issues, reduces complexities, provides better execution and less chip region which is not the case with the Kogge-Stone adder (KSA). It is also very faster than ripple-carry adders (RCA).

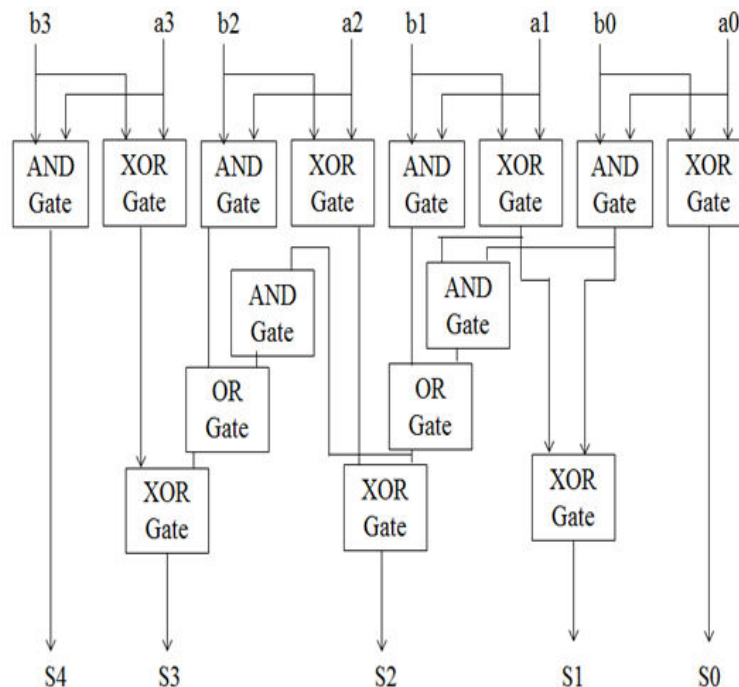


Figure 2: Block Diagram of Brent Kung Adder

IV. IMPLEMENTATION METHODOLOGY

Following are the equations used to design the two parallel FIR filter with two inputs $x(2k)$, $x(2k+1)$ and two outputs $y(2k)$, $y(2k+1)$. For implementing this filter three FIR sub-filter blocks has been used as compare to traditional two FIRs sub-block filter, having length $N/3$. Two of three sub-filters $2k$ and $2k+1$ are having symmetric coefficient which reduces the number of multiplier and adders. Here two preprocessing and four post-processing adders have been used along with delay equipment. The symmetric sub-filter block has been implemented at the cost of two additional adders among those one is pre-processing and other one is post-processing for $L=2$.

Digital Signal Processing (DSP) operations are increasingly being implemented on Field Programmable Gate Array (FPGA) platform.

FPGA supports all signal processing operations like audio processing, video processing, filtering, frequency transformations and so on. On the other hand, Application Specific Integrated Circuits (ASICs) are also used to implement the signal processing operations. ASIC level of design is used to implement only specific application. Hence, flexibility and configurability is not possible in ASIC level of design. Therefore, FPGA is better than ASIC by all means. The analog operations can be implemented in FPGA by compiling Verilog Hardware Description Language (HDL) code for corresponding operation into register transfer logic (RTL) net list. In addition, synthesis process is used to produce bits which control logic gates and fills the registers and memories in an FPGA.

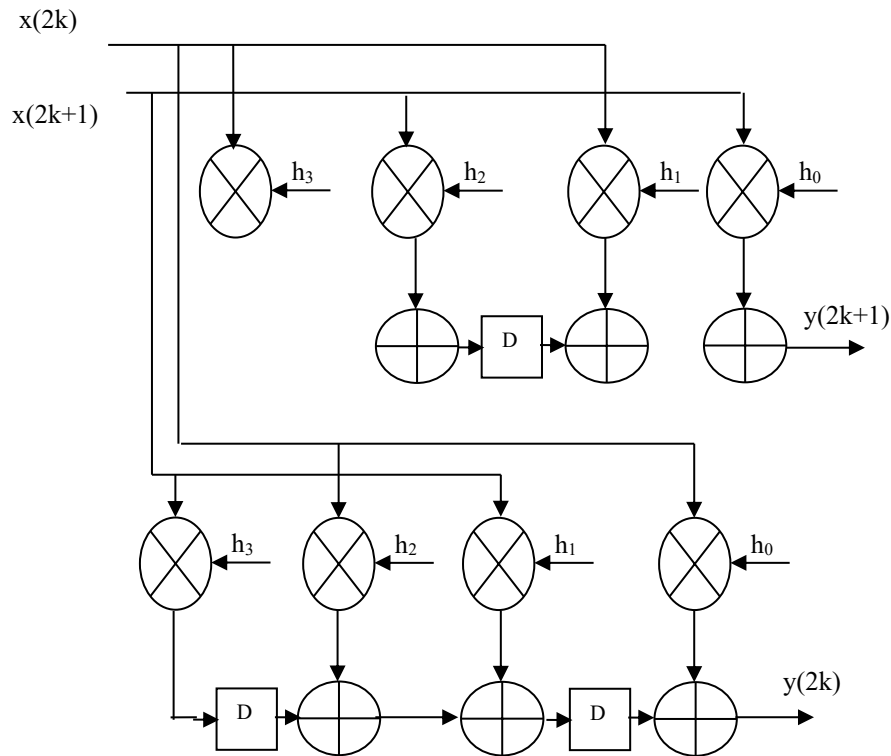


Figure 3: Block Diagram of 2x2 Parallel FIR Filter

Parallel FIR Filter used in Image System

Field Programmable Gate Arrays (FPGAs) are pre-fabricated silicon devices that can be electrically programmed in the field to become almost any kind of digital circuit or system. For low and medium productions, FPGAs provide cheaper solution and faster time to market as compared to the ASIC circuits.

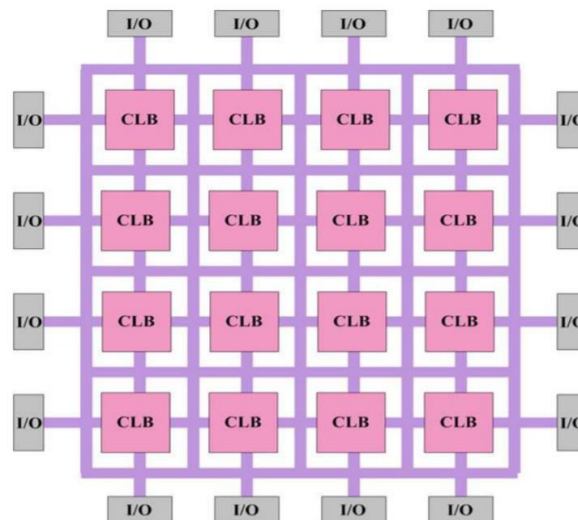


Figure 4: Architecture of FPGA

One alternative solution to SRAM based FPGA is the use of flash memory based FPGA. Flash memory or Electrically Erasable Programmable Read Only Memory (EEPROM) offers many advantages in FPGA. These memories are non-volatile in nature. Also flash memory based FPGAs are more area efficient than SRAM based FPGA. However, flash memory based FPGA has its own limitation of configurability and configurability. When compared to reconfigurable property, the SRAM based FPGA is considered as better than FPGA. In other hand, flash memory based FPGA uses non-standard CMOS process.

V. SIMULATION RESULT AND DISCUSSION

All the designing and experiment regarding algorithm that we have mentioned in this paper is being developed on Xilinx 14.2i updated version. Xilinx 9.2i has couple of the striking features such as low memory requirement, fast debugging, and low cost. The latest release of ISETM (Integrated Software Environment) design tool provides the low memory requirement approximate 27 percentage low. ISE 14.2i that provides advanced tools like smart compile technology with better usage of their computing hardware provides faster timing closure and higher quality of results for a better time to designing solution. ISE 14.1i Xilinx tools permits greater flexibility for designs which leverage embedded processors. The ISE 14.2i Design suite is accompanied by the release of chip scope ProTM 14.2i debug and verification software. By the aid of that software we debug the program easily.

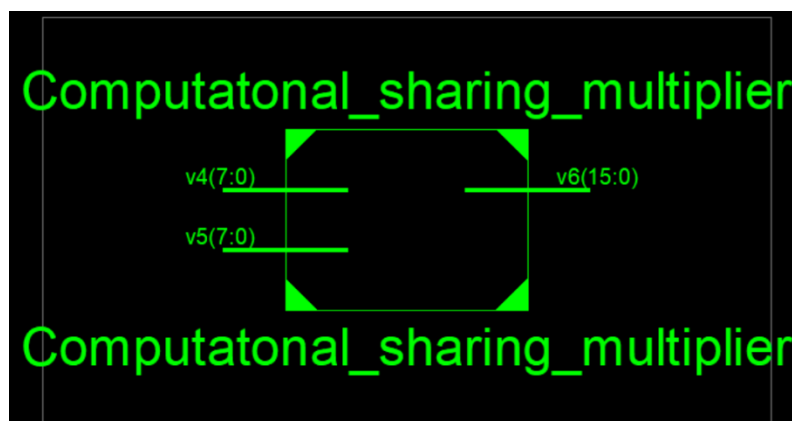


Figure 5: View Technology Schematic CSM

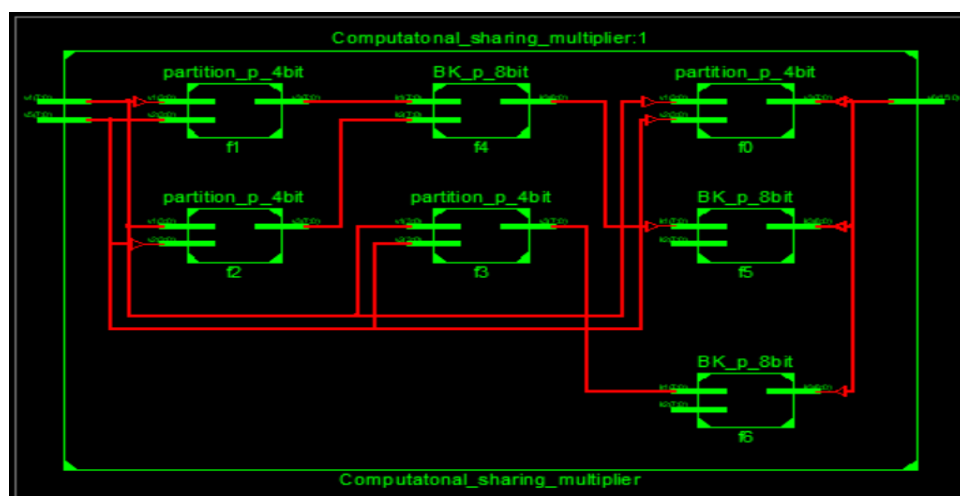


Figure 6: RTL View of CSM

```

Advanced HDL Synthesis Report

Macro Statistics
# Xors                                     : 149
 1-bit xor2                               : 137
 1-bit xor3                               : 12

Device utilization summary:
-----
Selected Device : 5v1x20tff323-2

Slice Logic Utilization:
Number of Slice LUTs:          128 out of 12480    1%
Number used as Logic:         128 out of 12480    1%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 128
Number with an unused Flip Flop: 128 out of 128    100%
Number with an unused LUT:        0 out of 128     0%
Number of fully used LUT-FF pairs: 0 out of 128     0%
Number of unique control sets:    0

IO Utilization:
Number of IOs:                 32
Number of bonded IOBs:         32 out of 172     18%

Timing Summary:
-----
Speed Grade: -2

Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 10.250ns
  
```

Figure 7: Design Summary of CSM



Figure 8: VTS for 4-bit BK Adder

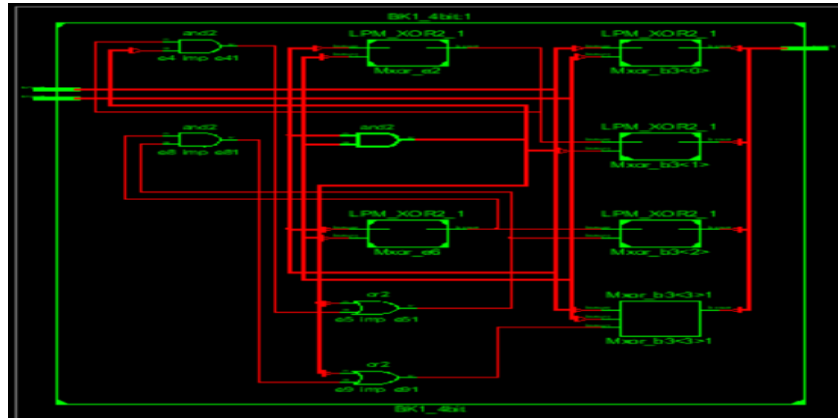


Figure 9: RTL View of 4-bit BK Adder

HDL Synthesis Report

Macro Statistics

# Xors	:	6
1-bit xor2	:	5
1-bit xor3	:	1

Device utilization summary:

Selected Device : 5v1x20tff323-2

Slice Logic Utilization:

Number of Slice LUTs:	6	out of	12480	0%
Number used as Logic:	6	out of	12480	0%

Slice Logic Distribution:

Number of LUT Flip Flop pairs used:	6			
Number with an unused Flip Flop:	6	out of	6	100%
Number with an unused LUT:	0	out of	6	0%
Number of fully used LUT-FF pairs:	0	out of	6	0%
Number of unique control sets:	0			

IO Utilization:

Number of IOs:	13			
Number of bonded IOBs:	13	out of	172	7%

Timing Summary:

Speed Grade: -2

Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 4.453ns

Figure 10: Design Summary of BK Adder

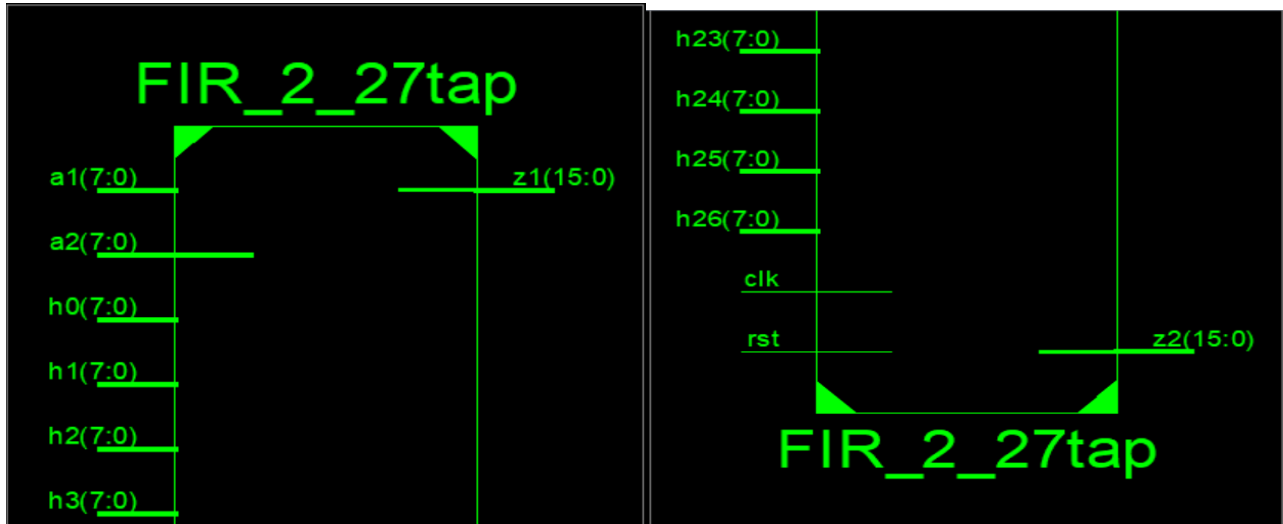


Figure 11: VTS for Parallel Filter

```

Advanced HDL Synthesis Report

Macro Statistics
# MACs                                     : 55
  4x4-to-16-bit MAC                       : 55
# Multipliers                             : 165
  4x4-bit multiplier                      : 165
# Adders/Subtractors                     : 110
  16-bit adder                           : 110
# Registers                               : 256
  Flip-Flops                             : 256
# Xors                                    : 764
  1-bit xor2                             : 648
  1-bit xor3                             : 116

Device utilization summary:
-----
Selected Device : 5v1x20tff323-2

Slice Logic Utilization:
Number of Slice Registers:          96 out of 12480    0%
Number of Slice LUTs:              1450 out of 12480  11%
  Number used as Logic:            1450 out of 12480  11%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 1509
  Number with an unused Flip Flop: 1413 out of 1509   93%
  Number with an unused LUT:        59 out of 1509    3%
  Number of fully used LUT-FF pairs: 37 out of 1509    2%
  Number of unique control sets:    1

IO Utilization:
Number of IOs:                      266
Number of bonded IOBs:              130 out of 172   75%

Timing Summary:
-----
Speed Grade: -2

Minimum period: 27.855ns (Maximum Frequency: 35.900MHz)
Minimum input arrival time before clock: 29.943ns
Maximum output required time after clock: 2.826ns
Maximum combinational path delay: No path found
  
```

Figure 12: Design Summary of Parallel Filter

VI. CONCLUSION

This work presents an efficient VLSI architecture for a parallel FIR filter using a Brent-Kung adder and a Computational Sharing Multiplier (CSM). The Brent-Kung adder significantly enhances the speed of accumulation operations due to its logarithmic delay and low fan-out properties, making it well-suited for high-speed DSP applications. Simultaneously, the CSM technique reduces hardware redundancy by sharing partial computations across multiple multipliers, leading to reduced area and power consumption. The parallel implementation further boosts throughput, making the architecture ideal for real-time applications such as image processing, audio filtering, and wireless communication. The proposed design, synthesized and evaluated using FPGA tools, demonstrates superior performance in terms of delay, power efficiency, and logic resource utilization compared to conventional FIR architectures. Overall, this architecture contributes a scalable, high-performance, and energy-efficient solution for digital signal processing systems, with potential extensions into adaptive filtering and ASIC implementations for commercial deployment.

REFERENCES

- [1] S Narendran and B T Geetha, "Performance Analysis of Parallel FIR Digital Filter Based on Even Symmetric Fast FIR Algorithm using Different Adders", 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE 2021.
- [2] K. Anjali Rao;Abhishek Kumar;Neetesh Purohit, "Efficient Implementation for 3-Parallel Linear-Phase FIR Digital Odd Length Filters", IEEE 4th Conference on Information & Communication Technology (CICT), IEEE 2020.
- [3] Yi Zheng;Ping Zheng, "Case Teaching of Parallel FIR Digital Filter Design Combined Matlab with FPGAs", International Conference on Artificial Intelligence and Education (ICAIE), IEEE 2020.
- [4] S. Sreekanth; Pratima B. Shinde; G. Vijaya Durga, "Performance Analysis of Higher Order FIR Polyphase Filter", Second International Conference on Intelligent Computing and Control Systems (ICICCS), IEEE 2018.
- [5] Qiaoyu Tian;Yinan Wang;Guiqing Liu;Xiangyu Liu;Jietao Diao, Hui Xu, "Hardware Efficient Parallel FIR Filter Structure Based on Modified Cook-Toom Algorithm", IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), IEEE 2018.
- [6] Payal Paliwal;Janki Ballabh Sharma, "Efficient FPGA Implementation Architecture of Fast FIR Algorithm Using Han-Carlson Adder Based Vedic Multiplier", International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE 2018.
- [7] Swetha Annangi;Ravisankar Puli, "ASIC implementation of efficient 16-parallel fast FIR algorithm filter structure", 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE 2017.
- [8] Shalina Percy Delicia Figuli;Peter Figuli;Jürgen Becker, "A reconfigurable high-speed spiral FIR filter architecture", 40th International Conference on Telecommunications and Signal Processing (TSP), IEEE 2017.
- [9] Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner, "FPGA Implementation of High Speed FIR Filters Using Add and Shift Method", 1-4244-9707-X/06/\$20.00@2006 IEEE.
- [10] Amina Naaz.S, Mr.Pradeep M.N, Satish Bhairannawar and Srinivas halvi, "FPGA Implementation Of High Speed Vedic Multiplier using CSLA For Parallel Fir Architecture", 2014 2nd International Conference on Devices, Circuits and Systems (ICDCS).
- [11] Laxman P.Thakre, Suresh Balpande, Umesh Akare, Sudhir Lande, "Performance Evaluation and Synthesis of Multiplier used in FFT operation using Conventional and Vedic algorithms," Third international conference on emerging trends in Engineering and Technology , IEEE, 2010.
- [12] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur and Girish V. A., "Implementation of Vedic Multiplier for Digital Signal Processing," International Conference on VLSI ,Communication & Instrumentation (ICVCI), 2011.
- [13] G.Vaithyanathan, K.Venkatesan, S.Sivaramakrishnan, S.Sivaand, S.Jayakumar, "Simulation and implementation of Vedic multiplier using VHDL code," International Journal of Scientific & Engineering Research, vol.4, 2013.
- [14] Pushpalata Verma and K. K. Mehta, "Implementation of an Efficient Multiplier based on Vedic Mathematics Using EDA Tool," International Journal of Engineering and Advanced Technology(IJEAT), vol.1, June 2012.

- [15] C. Cheng and K. K. Parhi, "Further complexity reduction of parallel FIR filters," in Proc. IEEE ISCAS, May 2005, vol. 2, pp. 1835–1838.
- [16] C. Cheng and K. K. Parhi, "Low-cost parallel FIR structures with 2-stage parallelism," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 2, pp. 280–290, Feb. 2007.
- [17] J. G. Chung and K. K. Parhi, "Frequency-spectrum-based low-area low-power parallel FIR filter design," EURASIP J. Appl. Signal Process., vol. 2002, no. 9, pp. 444–453, Jan. 2002.
- [18] K. K. Parhi, VLSI Digital Signal Processing systems: Design and Implementation. New York: Wiley, 1999.
- [19] Nivedita A. Pande, Vaishali Niranjane, Anagha V. Choudhari, "Vedic Mathematics for Fast Multiplication in DSP," International Journal of Engineering and Innovative Technology (IJEIT), vol.2, 2013.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details