Enhanced FIR Filter Design Using Hybrid Adders and Multipliers

Mr. N. Srinivasa Naidu,
Department of ECE,
Anil Neerukonda Institute Of Techonology And Sciences,
Visakhapatnam,
srinivasanaidu.ece@anits.edu.in

Paluru Tharun,
Department of ECE,
Anil Neerukonda Institute Of Techonology And Sciences,
Visakhapatnam,
palurutharun.21.ece@anits.edu.in

P.Rohan Kumar Raju,
Department of ECE,
Anil Neerukonda Institute Of Techonology And Sciences,
Visakhapatnam,
prohankumarraju.21.ece@anits.edu.in

Kovvuru Himakar,
Department of ECE,
Anil Neerukonda Institute Of Techonology And Sciences,
Visakhapatnam,
kovvuruhimakar.21.ece@anits.edu.in

Lankavalasa Sanjay,
Department of ECE,
Anil Neerukonda Institute Of Techonology And Sciences,
Visakhapatnam,
lankavalasasanjay.21.ece@anits.edu.in

ABSTRACT

Efficient arithmetic operations play an important role in high-performance digital circuits and systems, especially for signal processing applications. This paper presents a new Enhanced FIR Filter, which is realized with a new hybrid adder and hybrid multiplier to maximize power, speed, and area. The hybrid adder includes Sparse Kogge-Stone and Ling adders and results in a 25% improvement in Power Delay Product (PDP) over standard adders. Using this more efficient hybrid adder, a 4-chain LSKS hybrid multiplier is designed and produces a PDP reduction of 41% and 63% lower power usage than the standard designs. When these arithmetic units are implemented in the Enhanced FIR Filter design, the final chipset has a PDP reduction of 77% and is 27% lower in power consumption while supporting higher frequency operation (33.333MHz). Overall, the Improved Aberrated FIR Filter firmware provides an energy-efficient, fast design for real-time DSP applications, while being more efficient in power, speed, and computation than other traditional implementations.

Keywords— ***FIR filter, hybrid multiplier, hybrid adders, low-power design, digital signal processing.***

INTRODUCTION

FIR filters are a type of Finite Impulse Response filters which have a critical importance in the processing and analysing of digital signals, as they can be found commonly in the audio signal processing, telecommunications, and image filtering. The reason for the widespread use of FIR filter is its ability to maintain signal integrity, accuracy, precision, and/or measurement due to its stability and linear phase characteristics, which makes it possible to accomplish these tasks with minimal distortion to the signal. On the other hand, these advantages can be overshadowed by the fact that using traditional methods to design the FIR filters is inefficient, in terms of power consumption and silicon area, because of the convoluted multiplications and additions that consumes a lot of resources.

A pioneering attempt to minimize power and area in FIR filters incorporated hybrid adder along with Vedic multipliers [1]. Then work attempted to analyze the conditional performance of various hybrid adder structures [2] with further effort concentrating on high-speed multiplier designs based primarily on hybrid adder architectures [3]. To achieve better speed, area and power performance, the researchers eventually moved on to modified versions of the parallel prefix adders: the classical Kogge–Stone form [4], sparse-tree (sparse Kogge–Stone) prefix adders to create a more area efficient design [5], and the 32-bit ripple–Ling hybrid carry adder to achieve better speed and power performance [6].

Aside from prefix-based methods, hybrid multipliers that combine Vedic mathematics with the Karatsuba algorithm have been designed to simplify implementation and increase throughput [8], and compressor-based multiplier designs have achieved additional power and area savings [9]. At the system level, we have seen the development of FPGA based FIR filter implementations, including low power, reconfigurable architectures [10].

In this paper, we build on these developments to present a novel FIR- filter design in Xilinx Vivado, which implements a hybrid multiplier method that leverages the low order low time carry computation of a sparse Kogge–Stone adder, along with the area and power efficiency of a Ling adder, while achieving the goals of today's low power, high speed DSP environment.

Related Work:

I. ADDERS

Adders are essential arithmetic building blocks in digital systems because they are used at the root of most computations and are involved in different computational tasks, whether simple addition or more complex tasks like multiplication, divisions, and signal processing. Within VLSI applications, adders appear in probably all systems such as digital signal processors, microprocessors, embedded systems, etc., and their efficiency is significant to the overall performance, power, and area of a system.

A. Brent-Kung Adder:

The Brent–Kung adder uses a two-step procedure: first, it uses a hierarchical parallel-prefix network to compute group generate and propagate signals in $\lceil \log_2 N \rceil$ levels, and then it computes the sum, where each bit's propagate input is combined with the appropriate carry. It can only fan-out 2 times and only uses $2 \cdot \log_2 N - 1$ levels of logic, and this results in logarithmic delay growth with respect to the bit-width.[7]

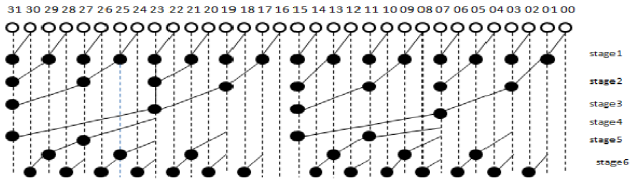

Figure 1:32-bit Brent-Kung Adder design

B. Kogge-Stone Adder:

The Kogge–Stone adder calculates addition in two parts: the prefix computation and sum calculation. In the prefix phase a dense parallel prefix tree calculates the carry signals, in minimal logic depth and maximum concurrency; then in the sum stage, each bit's propagate input is combined with the carry to create the sum. The Kogge–Stone adder facilitates ultra high speed by (1) fan out of only one, and (2) $\log_2 N$ depth[4].

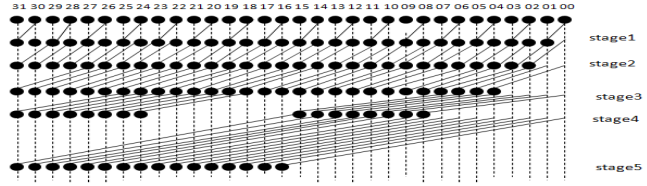

Figure 2: 32-bit Kogge-Stone Adder design

C. Sparse Kogge-Stone :

The Sparse Kogge – Stone adder keeps the standard Kogge – Stone parallel - prefix carry calculation but does so every so many bits. The approach is to utilize a reduced node, sparser prefix tree with reduced interconnectivity, reducing power consumption and silicon area while still offering near - logarithmic delay. Local carry resolve logic computes the other bits, striking an appropriate balance between speed and efficiency [5].

D. Ling Adder:

The Ling adder modifies the traditional generate (Gi) and propagate (Pi) signals into new signals:

$Hi = Gi + Pi \cdot Gi–1$

$Di = Pi \cdot Pi–1$

$Si = Pi \oplus Ci - 1$

- **Hi** predicts carry faster using current and previous stages.
- **Di** helps optimize multi-level carry structures.
- **Si** computes the sum using propagate and previous carry.

This reduces logic depth and simplifies the carry network. While this does add a preprocessing step, it will reduce the critical path, achieving a trade-off between speed and more hardware, and it remains a great fit for throughput-sensitive, but not area-sensitive designs [6].

II. HYBRID ADDER.

The 32-bit value will be split into 16-bit numbers, which will be given to different computation methods for efficiency. Computations for the lower and mixed 16-bits pieces will take place under a computation safe method, which reduces those computation times. The higher 16-bits pieces will be computed using the method with quick computation requirements.

32-bit adder hybrid architecture

A. Ling + Koggle-Stone Adder :

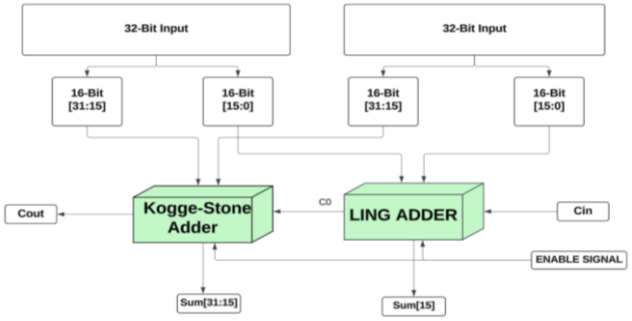

Figure 3:32-bit Ling + Kogge-Stone Adder Architecture

B. Brent-Kung + Sparse Kogge-Stone Adder:

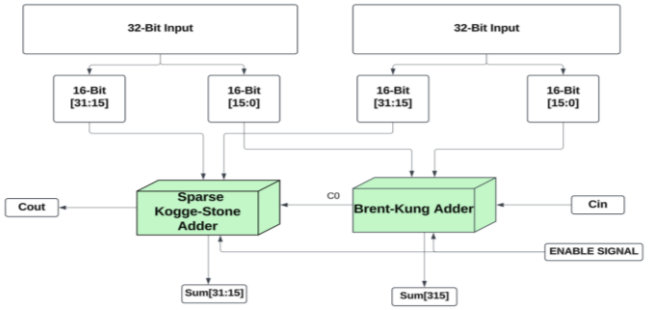

Figure 4:32-bit Brent-Kung + Sparse Kogge-Stone Adder Architecture

C. Ling+Brent-kung Adder:

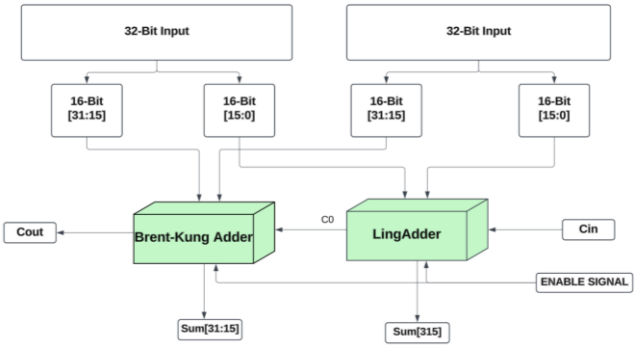

Figure 5:32-bit Ling+Brent-kung Adder Architecture

D. Ling+ Sparse Kogge-Stone:

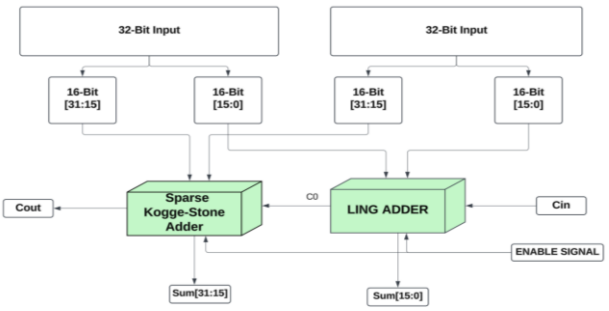

Figure 6:32-bit Ling+ Sparse Kogge-Stone Architecture

III. MULTIPLER

A. Vedic Multiplier:

The Vedic multiplier algorithm is a product of ancient Indian mathematics, most notably the "Urdhva Tiryabhyam" sutra, which translates to "Vertically and Crosswise". This multiplication algorithm allows multiplication to be performed more easily by creating partial products, which reduces delay. [1][8]

B. Karatsuba Multiplier:

The Karatsuba algorithm relies on leveraging the splitting of operands into small parts, which means the multiplication problem can be broken down into some smaller multiplications and additions.[8]

IV. PRATIAL PRODUCT GENRATION

PP (Partial Product) Generation is the first step in binary multiplication. It takes the multiplicand and multiplier, merges them, and will output intermediate binary products that will later be summed up in the final product. PP Generation, or Partial Product Generation, is one of the vital features of multipliers used in ALUs and digital signal processors (DSPs).[3]

For example, let's multiply A (Multiplicand) = 1011 (11 in decimal) and B (Multiplier) = 1101 (13 in decimal):

```
  1011          (Multiplicand – A)
  × 1101        (Multiplier – B)
------------
     1011       (B [0] × A)
+   00000       (B [1] × A, shifted 1 place)
+  101100       (B [2] × A, shifted 2 places)
+ 1011000       (B [3] × A, shifted 3 places)
  ------------
  10001111 (Final Product - 143 in decimal)
```

V. FIR FILTER

An FIR (finite impulse response) filter is a type of digital filter that operates on discrete-time signals. In comparison to IIR (infinite impulse response) filters, which will have an infinite number of coefficients, FIR filters have a finite number of coefficients, meaning that the impulse response will die out to zero after enough time.

A. 16-Tap 16-Bit Low-Pass FIR Filter Using Direct Arithmetic Operations

The title "16-Tap 16-Bit Low-Pass FIR Filter Using Direct Arithmetic Operations" describes the design with a Finite Impulse Response (FIR) filter:

- **16 taps**: When taps are used to describe a wet filter, taps are the coefficients of the filter.
- **16-bit**: The width of the input/output data.
- **Low-pass**: allows low frequency signals, and attenuates high frequencies.
- **Using Direct Arithmetic Operations**: as compared to special optimization methods (like the distributed arithmetic), which is the most efficient method to implement a FIR filter, you simply use a multiplication and an addition for each of your taps.

A digital 16-tap FIR (Finite Impulse Response) filter uses 16 past input samples to calculate each output value based on the signal that is being processed. Each tap consists of the filter's coefficient and the past input sample that is used to produce the output. In the case of a 16-tap filter, the output is based on the current input and the previous 15 inputs.

The filtering process involves the following steps:1) Save New Input: A new sample is saved to a shift register.2) Multiply by Coefficient: Each of the saved inputs is multiplied by a fixed coefficient.3) Take the Sum: The sum of all products is taken to form the new output sample.4) Shift the Buffer: When a new input sample arrives, all previous samples shift one position, with the oldest sample being removed from the shift register..

VI. PROPOSED WORK

A. PROPOSED HYBRID ADDER :

The 32-bit Hybrid Adder, creatively referred to as the LSSKA, ingeniously couples a Sparse Kogge-Stone Adder and a Ling Adder for both speed and power efficiency. It takes a 32-bit input and splits it into two 16-bit segments. The upper 16 bit are quickly calculated using the Sparse Kogge-Stone Adder since it is efficient when propagating carries, while the lower 16 bit are calculated with the Ling Adder which is meant to be power efficient. A carry bit travels between the two adder circuits to make sure the summation is correct. The total output includes the upper sum [31:16] and lower sum [15:0] as well as a final carry bit in the case of an overflow. This hybrid approach utilizes both types of adders in a single structure where high-speed VLSI designs and FIR filter (and DSP systems) are crucial in finding an optimal trade-off between performance, area, and power efficiency.

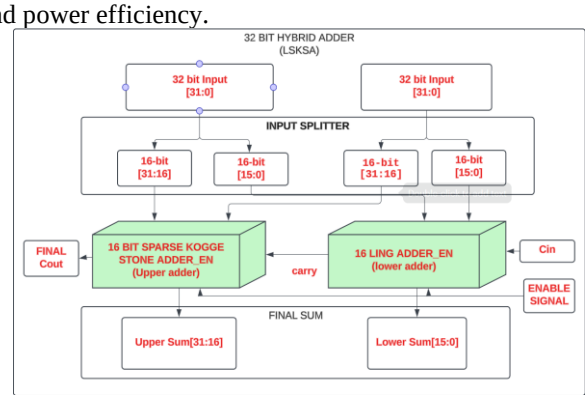

Figure 7:32-bit hybrid adder architecture

Stimulation Results:

Design and Implementation of a Hybrid Adder Using Verilog:

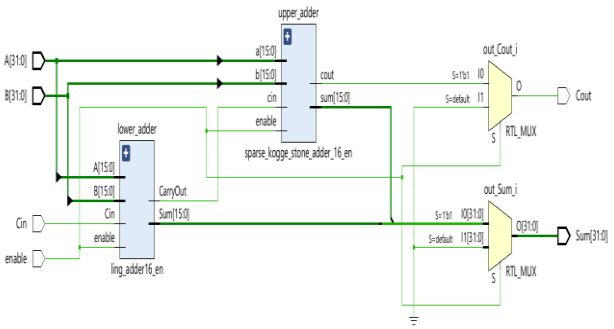

Fig 8. RTL schematic of 32-Bit Ling+ Sparse Kogge-Stone Adder

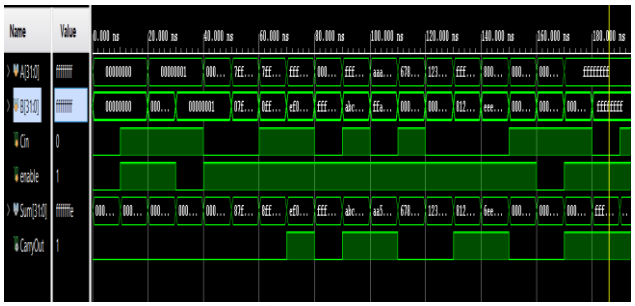

Fig 9. Program output result of 32-Bit Ling+ Sparse Kogge-Stone Adder

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	11	4	Cin	Cout	20.807	4.658	16.148	∞	input port clock
Path 2	∞	11	5	Cin	Sum[28]	20.683	4.646	16.038	∞	input port clock
Path 3	∞	11	5	Cin	Sum[30]	20.188	4.646	15.542	∞	input port clock
Path 4	∞	11	5	Cin	Sum[22]	20.126	4.646	15.480	∞	input port clock
Path 5	∞	11	5	Cin	Sum[26]	20.122	4.667	15.455	∞	input port clock
Path 6	∞	11	5	Cin	Sum[25]	20.074	4.654	15.420	∞	input port clock
Path 7	∞	11	5	Cin	Sum[31]	19.960	4.664	15.297	∞	input port clock
Path 8	∞	10	5	Cin	Sum[21]	19.858	4.522	15.337	∞	input port clock
Path 9	∞	11	5	Cin	Sum[23]	19.844	4.635	15.210	∞	input port clock
Path 10	∞	11	5	Cin	Sum[24]	19.730	4.650	15.079	∞	input port clock

Fig 10. Time delay of 32-Bit Ling+ Sparse Kogge-Stone Adder

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**15.095 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**100.5°C**
Thermal Margin:	-15.5°C (-3.0 W)
Ambient Temperature:	25.0 °C
Effective θJA:	5.0°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

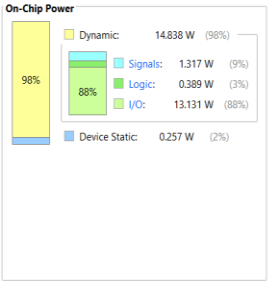

Fig 11. Power analysis of 32-Bit Ling+ Sparse Kogge-Stone Adder

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	31	0	0	8150	0.38
SLICEL	31	0			
SLICEM	0	0			
LUT as Logic	80	0	0	32600	0.25
using O5 output only	0				
using O6 output only	63				
using O5 and O6	17				
LUT as Memory	0	0	0	9600	0.00
LUT as Distributed RAM	0	0			
using O5 output only	0				
using O6 output only	0				
using O5 and O6	0				
LUT as Shift Register	0	0			
using O5 output only	0				
using O6 output only	0				
using O5 and O6	0				
Slice Registers	0	0	0	65200	0.00
Register driven from within the Slice	0				
Register driven from outside the Slice	0				
Unique Control Sets	0		0	8150	0.00

Fig 12. Slice logic distribution of 32-Bit Ling+ Sparse Kogge-Stone Adder

B. PROPOSED HYBRID MULTIPLIER:

The Proposed Hybrid Multiplier Architecture intelligently combines the two operations of multiplication and addition to offer both speed improvements and reduced power. The architecture begins with 2 16-bit inputs through a bitwise AND to generate partial products. The algorithm ensures that the partial products are left-shifted based upon significance of position. The left-shifted partial products are passed to the proposed hybrid adder, which takes advantage of Kogge-Stone and Ling adders, to reduce computation time. The final outcome of the multiplier is the final summation. Thus, the architecture produces results that are faster and consume reduced power that will predominantly benefit designs that require high speed and low power use.

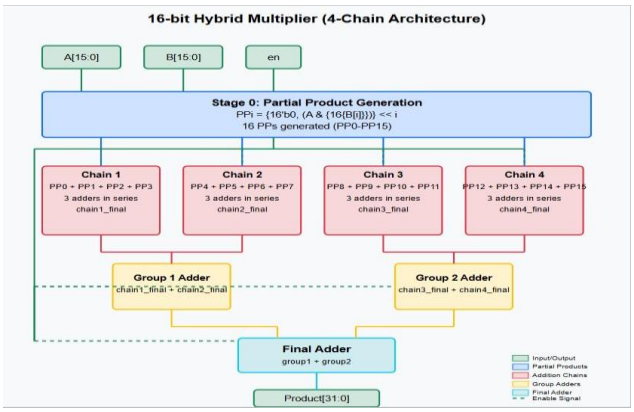

Figure 13:16-Bit Proposed Hybrid Multiplier Architecture

Stimulation Results:

Design and Implementation of a Hybrid Multiplier Using Verilog:

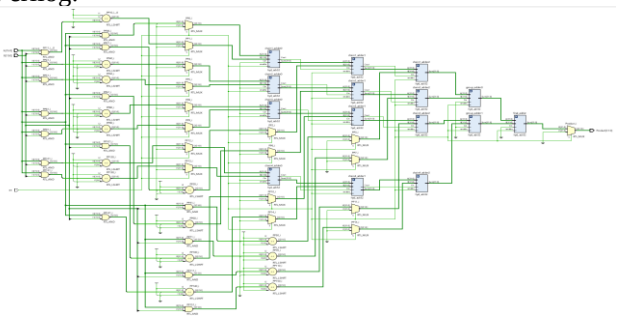

Fig 14. RTL schematic of 16-Bit Proposed Hybrid Multiplier

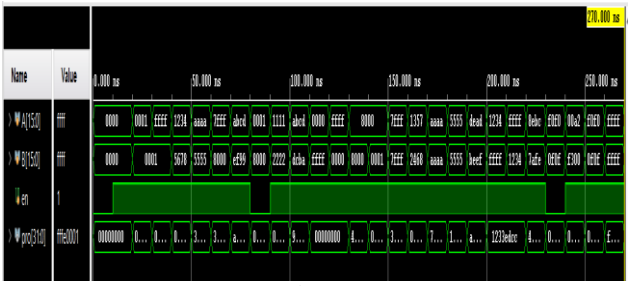

Fig 15. Program output result of 16-Bit Proposed Hybrid Multiplier

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	19	615	en	Product[29]	30.375	6.587	23.789	∞	input port clock
Path 2	∞	19	615	en	Product[30]	30.163	6.361	23.802	∞	input port clock
Path 3	∞	19	615	en	Product[28]	30.101	6.355	23.746	∞	input port clock
Path 4	∞	19	615	en	Product[27]	29.886	6.598	23.288	∞	input port clock
Path 5	∞	18	615	en	Product[25]	29.050	6.480	22.570	∞	input port clock
Path 6	∞	18	615	en	Product[24]	28.938	6.233	22.705	∞	input port clock
Path 7	∞	18	615	en	Product[26]	28.803	6.239	22.565	∞	input port clock
Path 8	∞	19	615	en	Product[31]	28.524	6.127	22.397	∞	input port clock
Path 9	∞	17	615	en	Product[23]	27.182	5.855	21.327	∞	input port clock
Path 10	∞	17	615	en	Product[22]	26.995	5.878	21.117	∞	input port clock

Fig 16. Time delay of 16-Bit Proposed Hybrid Multiplier

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**12.073 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**85.4°C**
Thermal Margin:	-0.4°C (-0.0 W)
Ambient Temperature:	25.0 °C
Effective θJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

On-Chip Power

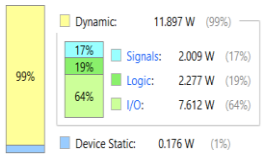

Fig 17. Power analysis of 16-Bit Proposed Hybrid Multiplier

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	150	0	0	8150	1.94
SLICEL	91	0			
SLICEM	67	0			
LUT as Logic	529	0	0	32600	1.62
using O5 output only	0				
using O6 output only	348				
using O5 and O6	181				
LUT as Memory	0	0	0	9600	0.00
LUT as Distributed RAM	0	0			
using O5 output only	0				
using O6 output only	0				
using O5 and O6	0				
LUT as Shift Register	0	0			
using O5 output only	0				
using O6 output only	0				
using O5 and O6	0				
Slice Registers	0	0	0	65200	0.00
Register driven from within the Slice	0				
Register driven from outside the Slice	0				
Unique Control Sets	0		0	8150	0.00

Fig 18. Slice logic distribution of 16-Bit Proposed Hybrid Multiplier

C. PROPOSED FIR FILTER:

We created a finite impulse response (FIR) filter using our proposed hybrid adder and hybrid multiplier. The hybrid adder consists of a sparse Kogge-Stone adder and Ling adder, which increases speed while reducing overall power and area usage. The hybrid adder was also used in the design of the hybrid multiplier which efficiently produces and adds the partial products.

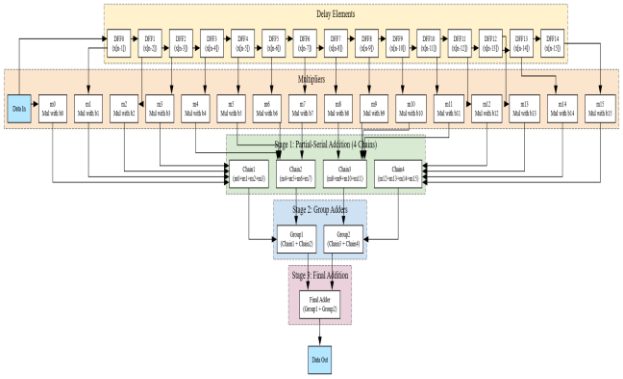

Figure 19:16-Bit Proposed Fir Filter Architecture

Stimulation Results:

Design and Implementation of a Hybrid Multiplier Using Verilog:

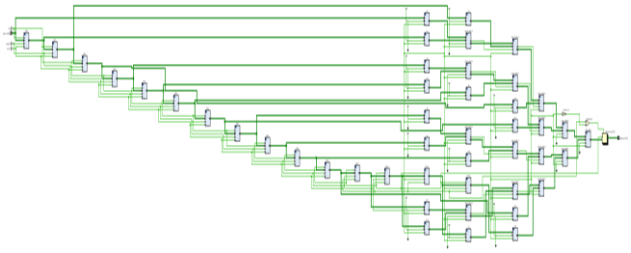

Fig 20. RTL schematic of 16-Bit Proposed Fir Filter

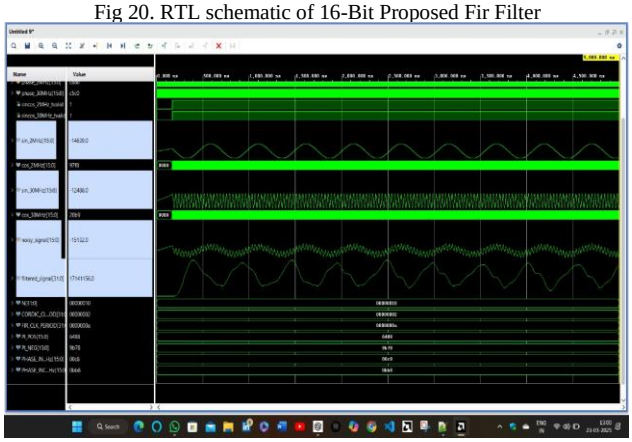

Fig 21. Program output result of 16-Bit Proposed Fir Filter

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	26	1056	enable	data_out_reg[24]/D	30.928	6.984	23.944	∞	input port clock
Path 2	∞	26	1056	enable	data_out_reg[23]/D	30.900	6.956	23.944	∞	input port clock
Path 3	∞	25	1056	enable	data_out_reg[22]/D	29.807	6.832	22.975	∞	input port clock
Path 4	∞	25	1056	enable	data_out_reg[21]/D	29.614	6.832	22.781	∞	input port clock
Path 5	∞	24	1056	enable	data_out_reg[20]/D	29.485	6.708	22.777	∞	input port clock
Path 6	∞	23	1056	enable	data_out_reg[19]/D	28.594	6.356	22.238	∞	input port clock
Path 7	∞	22	1056	enable	data_out_reg[18]/D	27.592	6.004	21.588	∞	input port clock
Path 8	∞	21	1056	enable	data_out_reg[17]/D	26.623	5.644	20.979	∞	input port clock
Path 9	∞	20	1056	enable	data_out_reg[16]/D	25.734	5.292	20.442	∞	input port clock
Path 10	∞	19	1056	enable	data_out_reg[15]/D	24.264	5.168	19.096	∞	input port clock

Fig 22. Time delay of 16-Bit Proposed Fir Filter

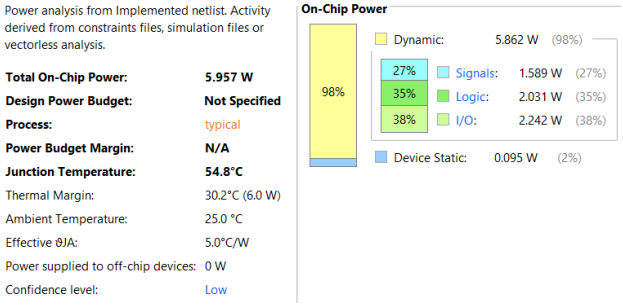

Fig 23. Power analysis of 16-Bit Proposed Fir Filter

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	267	0	0	8150	3.28
SLICEL	160	0			
SLICEM	107	0			
LUT as Logic	903	0	0	32600	2.77
using O5 output only	0				
using O6 output only	578				
using O5 and O6	325				
LUT as Memory	0	0	0	9600	0.00
LUT as Distributed RAM	0	0			
using O5 output only	0				
using O6 output only	0				
using O5 and O6	0				
LUT as Shift Register	0	0			
using O5 output only	0				
using O6 output only	0				
using O5 and O6	0				
Slice Registers	265	0	0	65200	0.41
Register driven from within the Slice	168				
Register driven from outside the Slice	97				
LUT in front of the register is unused	10				
LUT in front of the register is used	87				
Unique Control Sets	2		0	8150	0.02

Fig 24. Slice logic distribution of 16-Bit Proposed Fir Filter

VII. COMPARITIVE RESULTS

A detailed comparison of the power, speed, and area of the conventional Adder, and hybrid Adder is presented in Table 1.

Table 1. Performance comparison of various 32-bit Adder

Adder	Total Power (W)	Delay (ns)	Power delay product(W.ns)	Slice logic	LUT logic
BKA	22.991	22.469	516.584	24	64
KSA	24.034	17.409	418.407	41	134
LING	22.742	25.407	577.805	23	55
SKSA	22.744	24.585	559.161	23	55
Ling+BKA (Hybrid)	14.878	22.121	329.116	26	72
Ling+KSA (Hybrid)	15.225	21.822	332.239	30	95
BKA+SKSA (Hybrid)	14.764	24.430	360.684	26	71
LING+SKSA (Proposed)	**15.095**	**20.807**	**314.081**	**31**	**80**

The comparison of various 32-bit adders from Table-1, the proposed LING+SKSA hybrid adder clearly demonstrates the best overall performance. It achieves a Power Delay Product (PDP) of 314.081 W. ns, reflecting an improvement of approximately 25% over the best-performing conventional adder (PDP ≈ 418.407 W. ns). Furthermore, it maintains low total power and reduced delay, making it highly suitable for high-speed, low-power applications.

A detailed comparison of the power, speed, and area of the conventional multipliers, and hybrid multipliers is presented in Table 2.

Table 2. Performance comparison of various 16-bit multipliers

MULTIPLIER	Total Power (W)	Delay(ns)	Power delay product(W.ns)	Slice Logic	LUT logic
Dadda	37.424	25.773	964.528	102	350
Karatsuba	34.168	21.154	722.789	149	465
Wallace	39.189	23.546	922.744	110	387
Vedic	32.122	19.527	627.246	109	334
Parallel addition (4 adders)	33.827	25.573	865.057	183	644
Serial addition (4 adders)	13.898	42.897	596.182	231	778
4 chain tree (LKSA)	13.118	28.400	372.551	228	774
4 chain tree (LBKA)	32.572	25.490	830.260	156	543
4 chain LSKS (Proposed)	**12.073**	**30.375**	**366.717**	**158**	**529**

From Table-2, the proposed 4-chain LSKS multiplier (Power = 12.073 W, Delay = 30.375 ns, PDP = 366.717 W. ns) outperforms conventional multipliers in terms of power and overall efficiency. Comparing its Power Delay Product (PDP) of 366.717 W. ns to the best PDP among conventional designs (Vedic with 626.294 W. ns), the proposed design achieves approximately a 41% improvement. Moreover, it reduces total power by about 63% compared to Vedic (32.217 W → 12.073 W). These enhancements make the proposed multiplier highly suitable for applications demanding low power and efficient computation.

A detailed comparison of the power, speed, and area of the conventional Fir Filter, and hybrid Fir Filter is presented in Table 3.

Table 3. Performance comparison of various 16-bit Fir Filter

Type	Lowpass FIR Filter 16tap	Tot Power (w)	Delay (ns)	Power Delay Product (w. ns)	Slice logic	LUT logic	Slice Register
Without timing constraints	Direct arithmetic	36.929	35.436	1308.616	78	104	234
	parallel addition	26.076	30.127	785.591	269	883	265
	serial addition	23.806	38.870	925.339	285	970	267
	tree addition	23.316	31.102	725.174	279	913	265
	Enhanced Fir filter	**5.975**	**30.928**	**184.794**	**267**	**903**	**265**
With Timing constraints	Direct arithmetic 28.571 MHz (35ns period)	0.081	33.177	2.687	75	104	234
	Enhanced Fir filter 33.333 MHz (30nsperiod)	**0.075**	**22.065**	**1.654**	**201**	**921**	**265**

As shown in Table 3, our Enhanced FIR filter achieves a power-delay product (PDP) of 184.794 W. ns with no timing constraints while also realizing a 77% improvement compared to the best conventional design. The design also achieves a frequency of 33.333 MHz with power constrained to 0.075 W, allowing it to be used for real time DSP applications.

VIII.CONCLUSION

The proposed hybrid design showcases impressive improvements across all its components. The hybrid adder, which cleverly combines LING and SKSA techniques, achieves a power-delay product (PDP) of 314.081 W. ns, representing a 25% boost compared to the best conventional adder out there. Meanwhile, the 4-chain LSKS multiplier, boasting a PDP of 366.717 W.ns, surpasses traditional multipliers by delivering a remarkable 41% reduction in PDP and slashing power consumption from 32.217 W (Vedic) down to just 12.073 W. Lastly, the Enhanced FIR filter, crafted using these optimized arithmetic units, demonstrates a staggering 77% decrease in PDP (from 790.832 W.ns to 184.794 W.ns) and a 27% power saving (5.975 W compared to 8.18 W), all while running at a higher frequency of 33.333 MHz under timing constraints. Altogether, these findings confirm that the proposed hybrid design offers outstanding performance in terms of speed, power efficiency, and overall area, making it a fantastic choice for high-performance digital and real-time DSP applications.

IX. REFERENCES

[1] J. F. Sayed, B. H. Hasan, B. Muntasir, M. Hasan and F. 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), Dhaka, Bangladesh, 2021, pp. 748–752, doi:10.1109/ICREST51555.2021.9331063.

[2] R. R., B. Biju, A. Joy, V. Ganeshan and S. R., "Design and Performance Analysis of Various 32-bit Hybrid Adders using Verilog," 2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), Thiruvananthapuram, India, 2022, pp. 105–112, doi:10.1109/SPICES52834.2022.9774131.

[3] S. N. Leela, D. Keerthi Chandrika, K. Swetha, D. G. Kalali and G. Shanthi, "A Novel Design of High-Speed Multiplier Using Hybrid Adder Technique," 2024 3rd International Conference for Innovation in Technology (INOCON), Bangalore, India, 2024, pp. 1–5, doi:10.1109/INOCON60754.2024.10512237.

[4] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. Computers, vol. C-22, no. 8, pp. 786–793, Aug. 1973, doi:10.1109/T-C.1973.223610.

[5] S. K. Mathew, R. K. Krishnamurthy, M. A. Anders and S. Borkar, "130-nm address generation unit with 32-bit sparse-tree adder core," IEEE J. Solid-State Circuits, vol. 38, no. 5, pp. 690–697, May 2003, doi:10.1109/JSSC.2003.810056.

[6] N. Shang et al., "A 32-Bit Ripple-Ling Hybrid Carry Adder," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 71, no. 6, pp. 2709–2722, June 2024, doi:10.1109/TCSI.2024.3352139.

[7] Brent, R. P., & Kung, H. T. (1982). A Regular Layout for Parallel Adders. IEEE Transactions on Computers, C-31(3), 260–264. doi:10.1109/TC.1982.1675987

[8] S. Arish and R. K. Sharma, "An efficient binary multiplier design for high-speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm," 2015 Global Conference on Communication Technologies (GCCT), Thuckalay, India, 2015, pp. 192–196, doi:10.1109/GCCT.2015.7342650.

[9] K. Gunasekaran and M. Manikandan, "Low power and area efficient reconfigurable FIR filter implementation in FPGA," 2013 Int. Conf. Current Trends Eng. Technol. (ICCTET), Coimbatore, India, 2013, pp. 300–303, doi:10.1109/ICCTET.2013.6675970.

[10] S. S. Pujari, P. P. Muduli, A. Panda, R. Badhai, S. Nayak and Y. Sahoo, "Design & implementation of FIR filters using on-board ADC-DAC & FPGA," Int. Conf. Inf. Commun. Embedded Syst. (ICICES), Chennai, India, 2014, pp. 1–6, doi:10.1109/ICICES.2014.7033995.