Enhanced FIR Filter Design Using Hybrid Adders and Multipliers for Low Power Applications

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

P. 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

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

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

ABSTRACT:

Finite Impulse Response (FIR) filters are one of the key building blocks in digital signal processing for various applications such as audio processing, image filtering, telecommunications, biomedical applications, etc. Conventional FIR filters with standard adders and multipliers suffer from high power consumption and propagation delay. This work proposes an Enhanced FIR filter architecture based on a novel hybrid adder, a merger of Sparse Kogge Stone and Ling adder structures, with an enhanced hybrid multiplier that utilizes a 4-chain tree addition technique. Proposed 32-bit hybrid LKSA adder results in 45.64% improvement in Power Delay Product (PDP) over 32-bit conventional Ling adder and 43.69% improvement in PDP over 32-bit conventional Sparse Kogge Stone adder. A 16-bit hybrid multiplier is designed using the proposed 32-bit LKSA hybrid adder with 4-chain parallel tree addition, which results in a 41.53% reduction in PDP compared to the conventional 16-bit Vedic multiplier. When these arithmetic units are replaced in the conventional FIR Filter design, the final FIR filter has an 85.87% improvement in PDP. Overall, the Improved Aberrated FIR Filter firmware provides an energy-efficient and fast design for real-time DSP applications while being more efficient in power, speed, and computation than other traditional implementations. In this paper, we demonstrate and validate the proposed designs with Xilinx Vivado.

Keywords— ***FIR filter, Ling adder, Sparse Kogge Stone Adder, hybrid adders, hybrid multiplier, Enable Signal, low-power design.***

INTRODUCTION

Finite Impulse Response (FIR) filters play a vital role in digital signal processing (DSP) systems and are used in the widest range of applications, comprising audio processing, image filtering, communications, biomedical, etc., due to their stability and linear phase.[1] Because of the characteristic linear phase of FIR filters, signal distortion is minimized, making them extremely suitable for applications requiring more precision.[9] However, traditional FIR filter designs are prone to higher power consumption and larger propagation delay due to the computationally intensive convolution operations, which involve a large number of multiplications and additions.[3] Though existing work [1]–[10] has proposed hybrid adders and multiplier designs, most solutions lack in realizing an optimal power-delay-area tradeoff. This paper bridges the gaps by combining a hybrid adder with a 4-chain multiplier and shows considerable performance improvement over conventional solutions.

Traditional FIR filters are normally afflicted with excessive power consumption and lengthy propagation delays mainly because they are based on regular adder and multiplier circuits.[9] In this contribution, we try to overcome such limitations through a new hybrid design that combines a 32-bit hybrid adder comprising Sparse Kogge–Stone and Ling adder architectures with an optimized 16-bit hybrid multiplier based on a parallel 4-chain tree addition method.

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, division, 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 leverages its structure to two stages of performance gain: prefix computation and sum generation. In the Brent-Kung Adder, the carry computation is structured in a hierarchy which leads to a logarithmic depth overall delay thus, it is highly scalable in terms of bit width.[3] [4]

B. Kogge-Stone Adder:

The Kogge-Stone adder can perform its task efficiently in two different phases of computation: The prefix computation phase and the addition of the sum. The prefix computation phase is the phase in which the adder computes the carry signals using a dense tree structure, which will maximize parallelism and minimize the logic level depth.[3][5]

C. Sparse Kogge-Stone :

The sparse Kogge-Stone adder follows the original Kogge-Stone adder principles, but there are fewer interconnects in the prefix tree. The sparse Kogge-Stone adder was designed to alleviate that with an alternate prefix tree implementation, which has fewer nodes and interconnects, giving it a more

area and power-efficient implementation but still having a low delay.[3]

D. Ling Adder:

The advantages of the Ling Adder come from the original GG and PP signals being modified into a new set of signals that simplifies the carry computation. While the design may be a bit more complex, the Ling Adder provides a great option when a balance between speed and hardware is needed.[3] [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-bit pieces will take place under a computation-safe method, which reduces those computation times. The higher 16-bit pieces will be computed using the method with quick computation requirements.
Proposed design implementation, the upper 16 bits are processed by a Sparse Kogge–Stone adder offering fast parallel carry propagation, and the lower 16 bits are processed by a Ling adder that reduces switching power by minimizing carry generation. This partitioning scheme makes for an optimal speed-power trade-off. An Enable signal is introduced to control each block, which helps in minimizing the power by reducing the unnecessary switching activity known as Power gating or Power Isolation.

A. Ling + Kogge-Stone Adder :

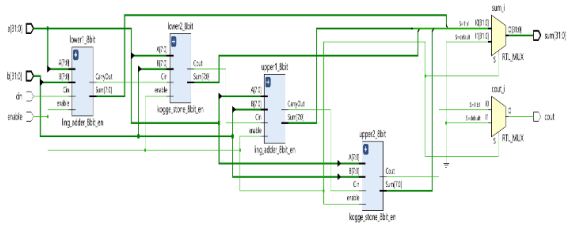

Figure 1: 32-bit Ling + Kogge-Stone Adder Schematic

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

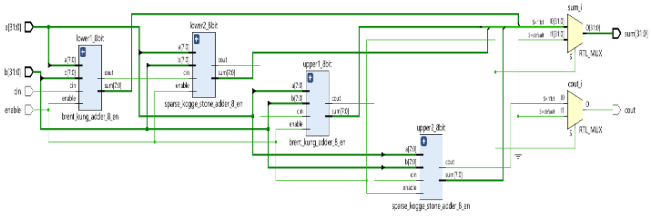

Figure 2: 32-bit Brent-Kung + Sparse Kogge-Stone Adder Schematic

C. Ling+Brent-kung Adder:

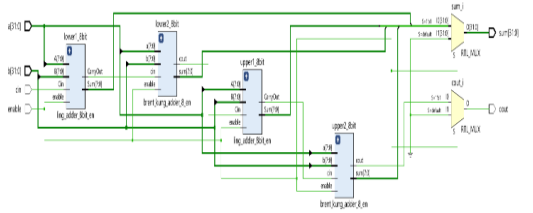

Figure 3: 32-bit Ling+Brent-kung Adder Schematic

D. Ling+ Sparse Kogge-Stone:

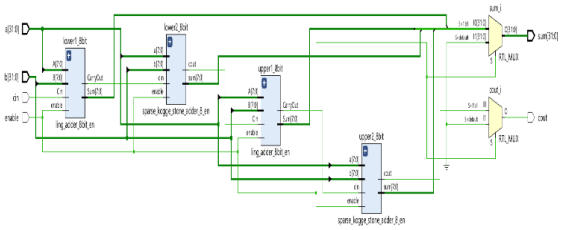

Figure 4: 32-bit Ling+ Sparse Kogge-Stone Adder Schematic

III. MULTIPLER

Effective multiplication processes are essential in the fields of high-performance computing and digital signal processing. In response to the growing need for multiplication that is both quicker and more economical, the idea of a hybrid multiplier design is becoming more popular. Multipliers are basic digital circuits that are utilized in many several uses for carrying out multiplication tasks. They are essential to attaining effective and high-performance computing across several domains. In the case of multiplication, the delay value will be more. We know that in this fast-growing technological world there is a need for speed, less area.[3]

IV. PRATIAL PRODUCT GENERATION

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).
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. TECHNIQUES USED FOR THE ADDITION OF PARTIAL PRODUCTS IN THE HYBRID MULTIPLIER DESIGN:

A. Parallel Addition with 32-bit Hybrid Adder:

In this process, seven hybrid adders work together in parallel for adding two partial products together in stage 1. Later on, their summation is achieved through the application of 4 hybrid adders in stage 2. Results are combined employing two hybrid adders in stage 3, and finally, the result comes into existence by stage 4, applying a hybrid adder. The process is optimized with utmost parallelism to result in quicker summation but potentially having more hardware devices than series addition.[5]

B. Serial Addition with 32-bit Hybrid Adder:

The partial products are added sequentially by 15 hybrid adders in a ripple-carry configuration. Hybrid adder adds first two partial products in the first step its result is carried as one input to the succeeding adder and the other input is the next partial product. The approach typically requires fewer hardware resources but more delay, compared to sequential addition than parallel addition.[5]

C. 2-Chain Tree Addition with 32-bit Hybrid Adder

In this approach,16 partial products are divided into two parallel chains of addition: chain 1 adds PP0 to PP7, and chain 2 adds PP8 to PP15 at stage 1, which generates two distinct sum results where they are added at stage 2 to obtain the final result. Each chain employs a serial (ripple) addition.[7]

D. 4-Chain Tree Addition with 32-bit Hybrid Adder

In this approach,16 partial products are divided into four parallel chains of addition: chain 1 adds from PP0 to PP3, chain 2 adds from PP4 to PP7, chain 3 adds from PP8 to 11, and chain 4 adds from 12 to PP15, which yields four distinct sum results at stage 1. Each chain employs a serial (ripple)

addition. The four chain results at stage 1 are combined at stage two in 2 groups, where each group is added again to get two different sum results, which are added at the last stage to find the final result. This approach minimizes propagation delay and hardware needs by efficiently combining multiple partial sums in parallel.[5] [7].

VI. 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.[8] [9]

16-Tap 16-Bit Low-Pass FIR Filter :

The 16-tap, 16-bit low-pass FIR filter normally computes its output as the sum of the weighted current input sample and the 15 previous input samples, where each is scaled by a preset coefficient. [8]

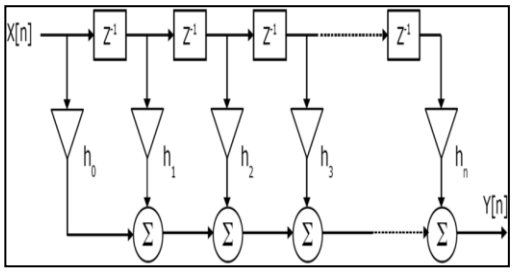

Figure 5: Block diagram of a Finite Impulse Response (FIR) filter.[11]

The structure of it has three dominant stages:
(1) A series of D Flip-Flops that trails behind the input samples,
(2) A step to generate the 32-bit partial products through multiplication with corresponding coefficients,
(3) A stage to summate these partial products to generate the final output.

Though straightforward, this process results in high power dissipation, increased propagation delay, and silicon area utilization, which requires a superior design.[1]

VII. PROPOSED WORK

A. PROPOSED HYBRID ADDER :

The 32-bit Hybrid Adder, creatively referred to as the LSKSA, 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 bits are quickly calculated using the Sparse Kogge-Stone Adder since it is efficient when propagating carries, while the lower 16 bits 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. An enable signal is used as a control signal. 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. An Enable signal is driven to control the adder which in result reduced the power consumption.

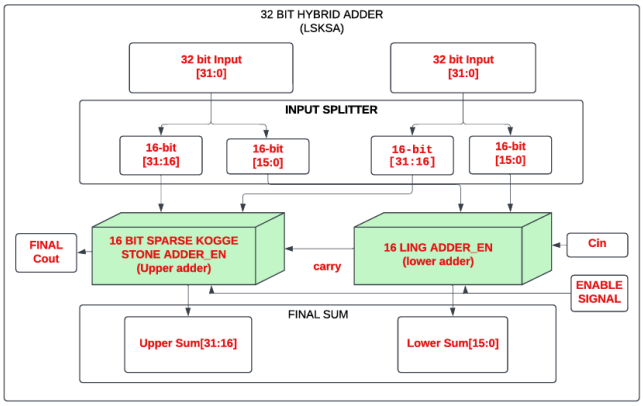

Figure 6: Proposed 32-bit hybrid adder architecture (Ling+ Sparse Kogge-Stone Adder)

VIII. SYNTHESIS AND IMPLEMENTATION RESULTS:

- Design and Implementation of a Hybrid Adder Using Verilog:

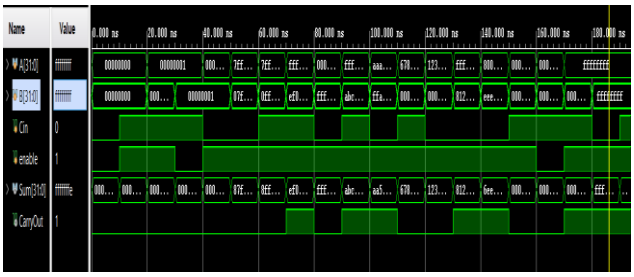

Fig 7: Program simulation output of proposed 32-bit Hybrid Adder

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 8: Time delay of 32-bit Ling+ Sparse Kogge-Stone Adder

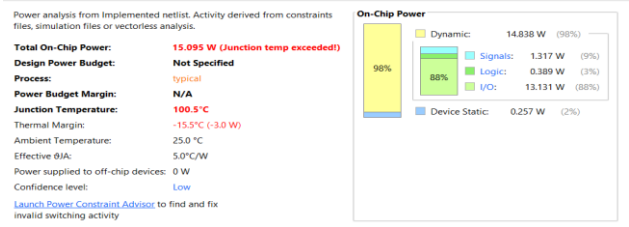

Fig 9: Power analysis of 32-bit Ling+ Sparse Kogge-Stone Adde

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				

Fig 10: 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 two 16-bit inputs through a bitwise AND to generate partial products. The algorithm ensures that the partial products are left-shifted based upon the significance of position. The left-shifted partial products are passed to the proposed hybrid adder using parallel four-chain

tree addition, which takes advantage of Sparse Kogge-Stone and Ling adders to reduce computation time. The outcome of the multiplier is the final summation. Thus, the architecture produces results that are faster and consume less power, which will predominantly benefit designs that require high speed and low power use.

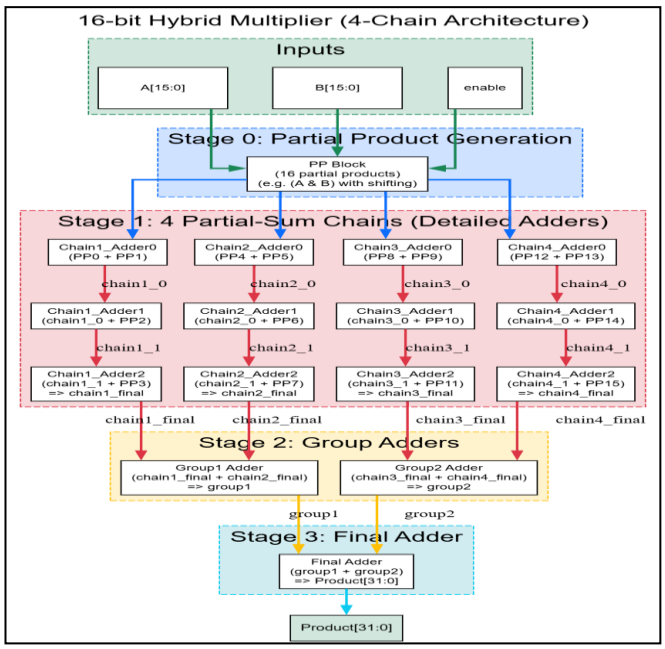

Figure 11: 16-Bit Proposed Hybrid Multiplier Architecture

- Design and Implementation of a Hybrid Multiplier Using Verilog:

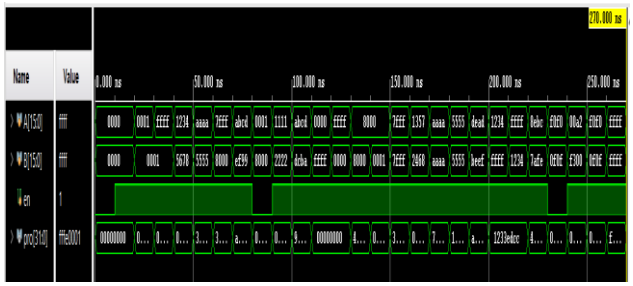

Fig 12: Program simulation output of 16-bit Proposed Hybrid Multiplier

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 13: 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

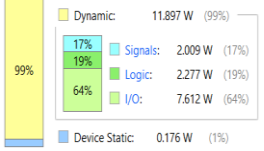

Fig 14: Power Analysis of 16-Bit Proposed Hybrid Multiplier

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	158	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				

Fig 15: 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. Coefficients of the design are selected so as to pass the low frequencies. The hybrid adder consists of a sparse Kogge-Stone adder and a 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. It uses a parallel four–chain addition to add the multiplication results of the delay elements and coefficients. An Enable signal is driven among the delay unit, Multiplier unit, and Adder unit to reduce the power consumption by minimizing the unwanted switching activities. This technique is called power isolation, which is a part of the spurious power suppression technique (SPSE).[4]

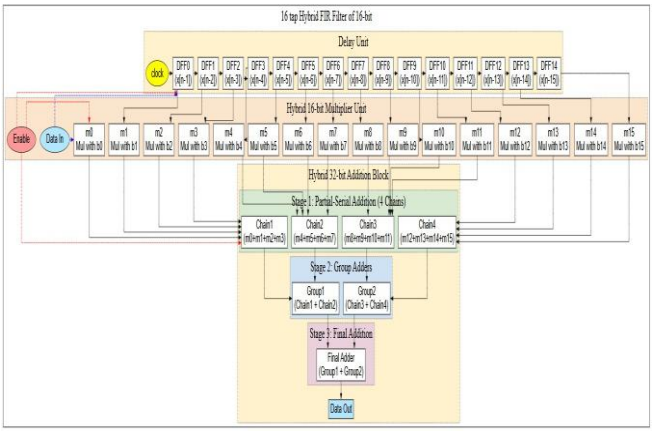

Figure 16: Proposed FIR Filter Architecture

- Design and Implementation of Enhanced FIR filter Using Verilog:

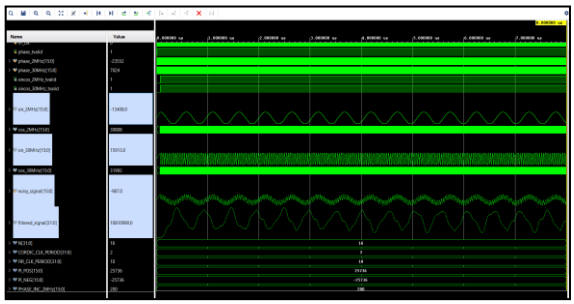

Fig 17: Program simulation output of Proposed Low Pass FIR Filter

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 18: Time delay of 16-bit Proposed FIR Filter

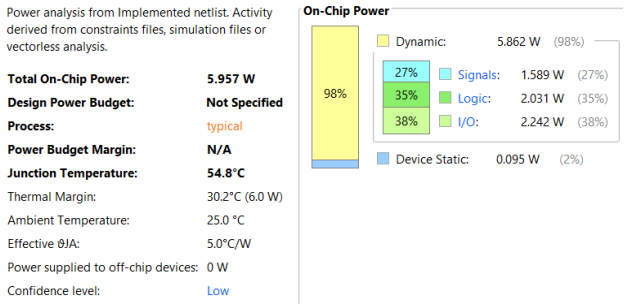

Fig 19: Power analysis of 16-bit Proposed FIR Filter without constraints

Slice	279	0	0	8150	3.42
SLICEL	164	0			
SLICEM	115	0			
LUT as Logic	913	0	0	32600	2.80
using O5 output only	0				
using O6 output only	806				
using O5 and O6	107				
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	60				
Register driven from outside the Slice	205				
LUT in front of the register is unused	20				
LUT in front of the register is used	185				
Unique Control Sets	2		0	8150	0.02

Fig 20: Slice logic distribution of 16-bit Proposed FIR Filter

IX. RESULTS COMPARISION

Table 1 presents the performance improvement of the hybrid designs proposed in terms of Power and Delay compared with the conventional designs.

Table 1 Improvements in Proposed Design

Component	Metric	Conventional Design	Proposed Hybrid Design Improvement
32-bit Adder	Power Consumption	Ling adder	33.63% reduction
	Delay	Ling adder	18.6% reduction
	Power Consumption	Sparse Kogge–Stone adder	33.63% reduction
	Delay	Sparse Kogge–Stone adder	15.36% reduction
16-bit Multiplier	Power Consumption	Vedic Multiplier	62.415% reduction
	Power Consumption	Wallace Tree Multiplier	69.192% reduction
16-bit 16 tap FIR Filter	Power Consumption	Arithmetic Operators	83.82% reduction
	Delay	Arithmetic Operators	12.721% reduction
	Power-Delay Product (PDP)	Arithmetic Operators	85.87% reduction

These improvements result in an energy-efficient, high-speed FIR filter design appropriate for real-time DSP applications, as confirmed using Xilinx Vivado.

Table 2 presents a detailed comparison of the power, speed, and area of the conventional Adder and the hybrid Adder.

Table 2: Performance comparison of various 32-bit adders

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**

In 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, reflecting an improvement of approximately 43.69% over the conventional 32-bit SKSA adder. Furthermore, it maintains low total power and reduced delay, making it highly suitable for high-speed, low-power applications.

Table 3 presents a detailed comparison of the power, speed, and area of the conventional multipliers and hybrid multipliers.

Table 3: 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
Multiplier with hybrid adders in parallel	33.827	25.573	865.057	183	644
Multiplier with hybrid adders in serial	13.898	42.897	596.182	231	778
Multiplier with 4 chain tree addition using LKSA	13.118	28.400	372.551	228	774
Multiplier with 4 chain tree addition using LBKA	32.572	25.490	830.260	156	543
Hybrid multiplier 2 chain tree addition with all adders	13.555	31.761	430.520	224	780
Multiplier with 4 chain tree addition using LSKSA (Proposed)	**12.073**	**30.375**	**366.717**	**158**	**529**

From Table 3, the proposed 4-chain LSKSA multiplier (Power = 12.073 W, Delay = 30.375 ns, PDP = 366.717) outperforms conventional multipliers in terms of power and overall efficiency. Comparing its Power Delay Product (PDP) of 366.717 to the best PDP among conventional designs (Vedic with 626.294). The proposed design achieves a 41.53% improvement in PDP. Moreover, it reduces total power by 62.415% 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.

Table 4: Performance Comparison of various 16-bit 16 tap FIR filters

Type	Lowpass FIR Filter 16tap	Tot Power (w)	Delay (ns)	Power Delay Product	Slice logic	LUT logic	Slice Register
Without timing constraints	Conventional filter	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 (proposed)	**5.975**	**30.928**	**184.794**	**267**	**903**	**265**
With Timing constraints	Conventional filter 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

Table 4 presents a detailed comparison of the power, speed, and area of the conventional FIR filter and the hybrid FIR filter.

As shown in Table 4, our Enhanced FIR filter achieves a power-delay product (PDP) of 184.794, achieving an 85.87% reduction in PDP, 83.82% reduction in power, and 12.721% reduction in delay compared to that of a conventional FIR filter using arithmetic operations without timing constraints. The design also achieves a 7.40% reduction in Power, a 33.49% reduction in delay, and a 38.44% reduction in PDP, with timing constraints having a frequency of operation at 33.33MHz, allowing it to be used for real-time DSP applications.

X. FUTURE SCOPE

The proposed design of the hybrid FIR filter shows a quite achievements in both speed and delay efficiency, making it ideal for real-time DSP applications. Scale the design to handle larger bit-widths for high-end applications. We might also adjust the tap values, size, and number, which could help reduce noise and create a smoother output. These enhancements will make the design suitable for various real word applications such as biomedical applications, audio and video noise cancellation applications.[10]

XI. CONCLUSION

The proposed hybrid design in this work exhibits remarkable enhancement in all of its components. Our 32-bit hybrid LKSA adder achieves a 45.64% decrease in power–delay product (PDP) compared to conventional Ling adders and a 43.69% decrease compared to traditional Sparse Kogge–Stone adders. Additionally, the 16-bit hybrid multiplier realized by a 4-chain tree addition technique achieves a 62.415% decrease in power dissipation compared to a standard 16-bit Vedic multiplier. When applied to FIR filter design, these optimized arithmetic units yield a filter that realizes an overall 85.87% reduction in PDP, with the power consumption reduction being 83.82% and delay reduction being 12.721% relative to a regular FIR filter employing basic arithmetic operations. Besides, under timing constraints, the optimized design achieves an extra 7.40% power saving, a 33.49% delay saving, and a 38.44% PDP saving at 33.33 MHz. These results show that the proposed design is suitable for low-power applications.

XII. 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] V. M B, P. Akkamanchi and V. Uttarkar, "Low Power High Speed Brent Kung Adder Using SPST," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-6, doi: 10.1109/GCAT55367.2022.9971848.

[5] T. Shanmugaraja. and N. Kathikeyan, "Power Effective Multiply Accumulation Configuration For Low Power Applications Using Modified Parallel Prefix Adders," 2023 International Conference on Applied Intelligence and Sustainable Computing (ICAISC), Dharwad, India, 2023, pp. 1-7, doi: 10.1109/ICAISC58445.2023.10200142.

[6] N. Shang et al., "A 32-Bit Ripple-Ling Hybrid Carry Adder," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 71, no. 6, pp. 2709-2722, June 2024, doi: 10.1109/TCSI.2024.3352139.

[7] 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.

[8] K. Gunasekaran and M. Manikandan, "Low power and area efficient reconfigurable FIR filter implementation in FPGA," 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), Coimbatore, India, 2013, pp. 300-303, doi:10.1109/ICCTET.2013.6675970.

[9] 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," International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 2014, pp. 1-6, doi: 10.1109/ICICES.2014.7033995.

[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,” International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 2014, pp. 1-6, doi: 10.1109/ICICES.2014.7033995.

[11] Firmansyah, Iman & Yamaguchi, Yoshiki. (2022). Real-Time FPGA Implementation of FIR Filter Using OpenCL Design. Journal of Signal Processing Systems. 94. 1-13. 10.1007/s11265-021-01723-6.