2581-4575

International Journal For Recent Developments in Science & Technology

Crossref

A Peer Reviewed Research Journal

SKLANSKY AND KOGGE STONE ADDER DIGITAL FILTER DESIGN AND IMPLEMENTATION

[1]Kalikivayi Ajay Kumar,[2]Mr.Ch. Arun Prakash, [3]Doopam Mahesh Babu, [4]Kota Thirumala Eswar kumar,[5]Nulaka Sudheer Kumar Reddy

[2]Assistant Professor, ECE Dept, RISE Krishna Sai Prakasam Group of Institutions,Ongole,AP

[1,3,4&5]B.Tech final year students, ECE Dept, Rise Krishna Sai Prakasam Group of Institutions,Ongole-523001,AP

([1]ajaykumarkalikivai@gmail.com;[2]arungvpe@gmail.com;[3]maheshdoopam21@gmail.com;[4]eswarkuma308@gmail.com;[5]nulakasudheerkumarreddy@gmail.com)

ABSTRACT

The most important component of any electronic gadget has always been the arithmetic and logic unit. According to current developments, an arithmetic and logic unit must have an effective algorithmic operation, like addition and multiplication, in order to be relevant. The most fundamental operation in artificial intelligence, machine learning, neural networks, and digital signal processing is multiplication. Filters can be employed as memory components to store data in the digital age. The complete adders are the adders that are most frequently employed to implement digital filters. In this project, PPA is used to implement the digital filter. The Sklansky adder is the most efficient of the many PPAs. The hardware complexity increases when the Kogge Stone Adder is implemented.Therefore, it is required to simplify the Sklansky adder's hardware architecture. Numerous characteristics, including delay and the number of logic gates, were compared. Using a number of parameters, the architecture shown above tends to reduce the hardware complexity.

Keywords: Sklansky,Kogge Stone, Fir Filter.

INTRODUCTION

A complex DSP system consists of multiple adders and multipliers [2], and the efficient design of a DSP machine enhances the system's performance. An adder, a fundamental component, is frequently employed in many networks that are used in system like controllers and processing chips [3]. In these systems, a performance is improved by the running ability of adder and multiplier. Generally, it is necessary to redesign the frequencies, calculate the parameters, and adjust the filter at each time. In rare cases, the redesign calls for filters to be changed with their filter types or with their length to fit according to their requirements.

However, sampling can also happen in other ways. Equal space interval sampling is the second most popular method. Consider, for instance, receiving simultaneous measurements from a number of strain sensors spaced down the length of an airplane wing at one-centimeter intervals. Time and space are by far the most common domains, while there are many others that may be used. When the word "time domain" appears in DSP, keep in mind that it might refer to samples that were taken throughout a period of time or it could refer to any domain in which the samples were obtained. Each linear filter has three responses: frequency, step, and impulse.

LITERATURE SURVEY

Fast FIR Filter Design Using Carry Select Adder and Compressor. Subject 1: In the digital age, speed and area are now two of the most important design considerations. Increasing speed during addition or multiplication operations has always been a fundamental need for advanced system and application design. Carry Select Adder (CSA) is a fastest adder utilized in various processors to accomplish rapid arithmetic function. Many alternative adder architecture designs have been devised to boost the efficiency of the adder. In the semiconductor industry, it is well known that any processor can perform millions of work functions per second. Therefore, one of the primary criteria that should be considered when constructing multipliers is performance speed. In this study, we provide a multiplier-based compressor and carry select adder approach for building FIR filters.

Kogge Stone Adder implementation for signal processing applications.

Designing low power systems has emerged as a key performance objective. An effective part of digital signal processing applications is the finite impulse response filter. Multipliers and adders are crucial to the [R1] FIR filter's construction. Kogge stone adder and booth multiplier are used in the design of the suggested FIR filter. The Booth multiplier is a multiplication algorithm that is used to accomplish multiplication using 2's complement notation, while the Kogge stone adder is a high speed adder that is used to construct high performance circuits.

Dadda multiplier and parallel prefix adder are used in the implementation of a programmable fir filter.

Digital filters play an amazing role in many signal processing applications. The popularity of DSP can be attributed to the filters' performance. Filtering is typically used to manipulate the input data in order to achieve the desired output. To help create various applications that benefit the globe, numerous kinds of filters are utilized to control the data. The multiplication block is the most important component of any filter design since the way the multiplication is done determines how well the filter performs. The Dadda multiplier is one of several multipliers; it is significant since it does multiplication with a relatively small number of gates. In light of this, the MAC (Multiply and Accumulate) unit has been used to design and construct a programmable FIR filter.

Fast FIR Filter Design Using Carry Select Adder and Compressor. Dr. MinalSaxena, RakshaChouksey, and Deepak Kumar Patel are the authors.

In the digital age, speed and area are now two of the most important design considerations. Increasing speed during addition or multiplication processes has always been a fundamental need for sophisticated system and application design. Many processors use the quickest adder, the Carry Select Adder (CSA), to perform quick arithmetic operations. To improve the adder's efficiency, numerous adder architecture designs have been created. It is well known that processors in the semiconductor industry do millions of work operations every second**.**

EXISTING SYSTEM

Digital circuits employ the Sklansky Adder, a type of parallel prefix adder, for high-speed arithmetic operations, especially addition. It was first used by J. Sklansky in his work on carry propagation techniques in binary addition, and it bears his name. Structure of Features: It is a parallel prefix adder, which means that it uses a tree structure to compute the carry signals in parallel. By organizing bits and performing carry signal calculations in a hierarchical fashion, the Sklansky adder efficiently generates carry signals.

Carry Computation: A Generate-Propagate (G-P) is used by the adder to calculate carry signals. Reasoning: Benefits: Generate (Gig_I): Bit Ii Generates a Carry; Propagate (Pip_I): Bit Ii Propagates a Carry Low Logic Depth: By lowering the number of sequential logic gates on the critical path, the tree structure minimizes the propagation delay. High-Speed Performance: It is appropriate for high-speed arithmetic in processors because to its logarithmic time complexity (Log2n\Log_2 N, where Nn is the number of bits).

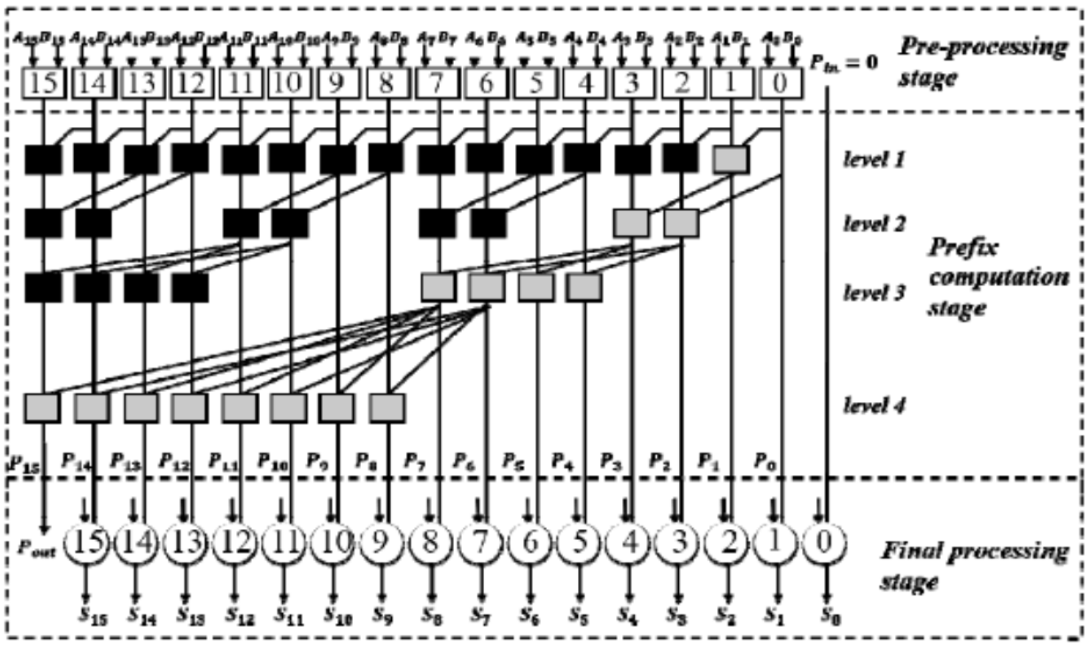


Fig:1 Architecture of Sklansky adder

Drawbacks: High Fan-Out: Some nodes have a higher fan-out because the structure needs multiple signals to branch out and connect to multiple gates. Area Consumption: In comparison to other adders, such as the Brent-Kung Adder, the complexity of interconnects may result in a larger area for implementation.

PROPOSED SYSTEM

The Kogge Stone Adder is a high-performance parallel prefix adder that is frequently used in digital circuits for quick arithmetic operations, especially in high-speed computing systems and processors. In their 1973 paper on carry propagation, Peter M. Kogge and Harold S. Stone introduced it. Features Structure: It is a parallel prefix adder that uses a prefix tree

structure to compute carry signals in parallel. To ensure minimal delay, the Kogge-Stone Adder uses a balanced binary tree for both carry generation and propagation.
Carry Calculation: The Generate-Propagate (G-P) is what the adder depends on. Reasoning:

Generate (Gig_I): Bit Ii Creates a Carry.

Propagate (Pip_I): Bit Ii Is Used to Spread a Carry.

Stages: The adder consists of several computational levels that gradually combine carry signals. In order to produce carry signals for every bit, each stage computes intermediate results for smaller groups of bits.

Benefits: Low Logic Depth: Log⁡2n\Log_2 N, where Nn is the number of bits, determines the carry computation time. This guarantees fast performance.

Typical Structure: The regularity makes it easier to design and optimize hardware implementation.

Drawbacks: High Interconnect Complexity: Area and power consumption are increased by the numerous wires and connectors. Area Cost: More hardware resources are needed than with simpler adders like the Brent-Kung Adder or the Ripple Carry Adder.

Comparison With Other Adders

Feature	**Kogge-Stone Adder**	**Sklansky Adder**	**Brent-Kung Adder**
Logic Depth	O(Log⁡2n)O(\Log_2 N)	O(Log⁡2n)O(\Log_2 N)	O(2log⁡2n)O(2\Log_2 N)
Area/Interconnect Cost	Very High	High	Moderate
Fan-Out Complexity	Moderate	High	Low
Speed	Fast	Fast	Moderate

TABLE:1 Comparison of Adders

Calculate the Generate (Gig_I) and Propagate (Pip_I) Signals for Every Bit in the **Preprocessing Stage:** A_I \Cdot B_I = Gi=Ai·Big_I A_I \O plus B_I = Pi=Ai⊕Bip_I
Take with you Propagation Stage: To calculate all carry signals, combine the gg and pp signals over all bits using the prefix tree.
Stage of Summation: Utilizing the Propagate Signals and Carry-Out Values, compute the sum bits: I = P_I \Oplus C_{I-1}
Si = Pi⊕Ci−1s_I

Because of its regular structure and speed, the Kogge-Stone adder is frequently preferred for systems where performance is crucial. However, for designs with limited resources, its area and connectivity costs may be a limiting factor.

Using a Parallel Prefix Adder for the First Filter

A parallel prefix adder and a shift and add multiplier are used in the design of the FIR filter.

The Multiply and Accumulate operation (MAC) is necessary for the design of a FIR filter in direct implementation.

When using a single tap filter, the output is allocated the result of simply multiplying the filter coefficient h0 by the variable x0. A 4-tap filter adds the results of four multipliers and assigns them to the result after multiplying the filter coefficient by the associated variables.

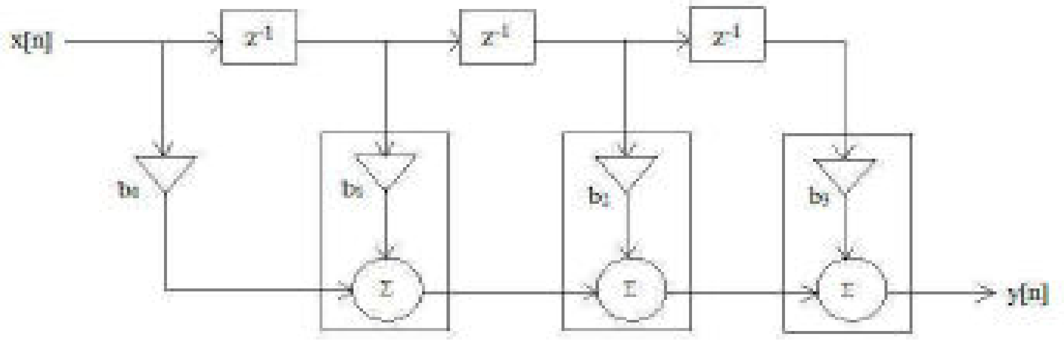


Fig:24-tap FIR Filter (Direct form)

The data flow graph schematic for a 4-tapFIR filter is displayed in Fig. 1. The longest path latency rises in tandem with the number of taps. Additionally, the architecture's need for multipliers and adders grows. Fig. 1 displays a data flow graph in direct form.

At the RTL level, the Multiplier/Result module was designed.

Figure 3-7 displays the module's block diagram together with the specifics of the register assignment. When the LOAD_cmd is asserted, the multiplier, B_in, is loaded into bits 0 through 7 of the register and serves as the module's input. The module additionally receives inputs from the adder block outputs (adder_out and Cout). Bit 0 of the register (LSB), bits 8 to 15 of the register (RB), and bits 0 to 15 of the register (RC) are the module's outputs. While RB is fed into the adder to be added to the multiplicand, LSB is fed back to the controller to decide the next state. Only when the controller asserts the STOP signal is the final multiplication result, or RC, regarded as legitimate.

A multi-bit carry-propagate adder called a parallel prefix adder (PPA) is used to add two multi-bit values in parallel. PPA speeds up adding by extending the carry look-ahead adder's generated and propagated logic. The three steps of the perspective architecture—pre-processing, prefix calculation, and final processing—are the fundamental schematic structure of the many PPAs that are examined. Let's take a closer look at each step.

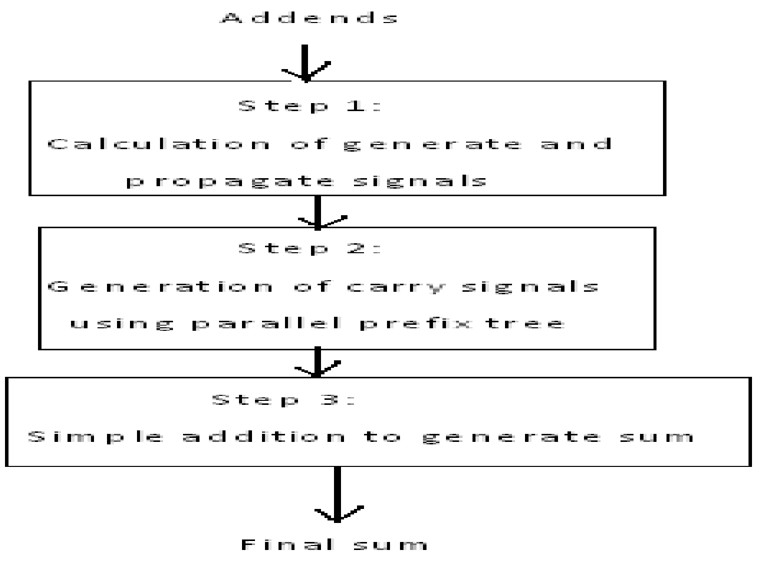

Addends
Step 1:
Calculation of generate and propagate signals
Step 2:
Generation of carry signals using parallel prefix tree
Step 3:
Simple addition to generate sum
Final sum

Fig:3 Addition procedure using Parallel Prefix tree structures

The two input signals (ai and bi) are added to the corresponding carry-in signal (carryi) in each bit (i) of the two operand block to generate the sum output (sumi). The formula that generates the sum output is Sumi = ai ^ bi ^ carryi (1).

The most important and time-consuming process is calculating the carry-in signals at each bit. Designing the carry-in signals for individual bit additions is the main goal of the carry-look ahead scheme of adders (CLA). This is accomplished by utilizing the following equations to create two signals: generate (gi) and propagate (pi): Gi = ai^ bi (2) Pi = ai^ bi (3) For any adder block, the carry-in signal is determined using the following formula: Ci+1 = gi V (pi) (4)

Hence, at any addition level, ci must be enlarged in order to compute ci+1

Parallel Prefix adders combine create and propagate signals in a different way to calculate carry-in at each addition level. Figure 2(a) and Figure 2(b) illustrate the employment of two operators, black and gray, in parallel prefix trees, respectively.

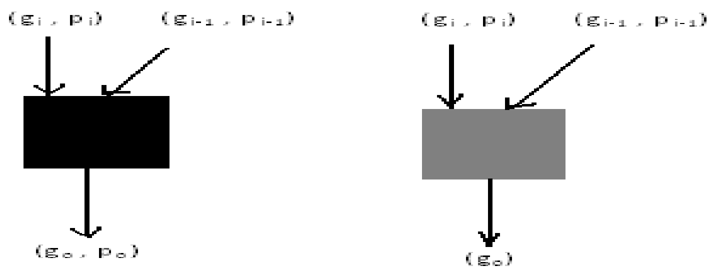


Fig:4 Operators used in Parallel Prefix trees

The black operator uses the following equations to calculate one set of generate and propagate signals (go, po) after receiving two sets of generate and propagate signals (gi, pi), and (gi-1, pi-1).

The gray operator computes only one generate signal using the same equation as in equation after receiving two sets of generate and propagate signals (gi, pi), (gi-1, pi-1).

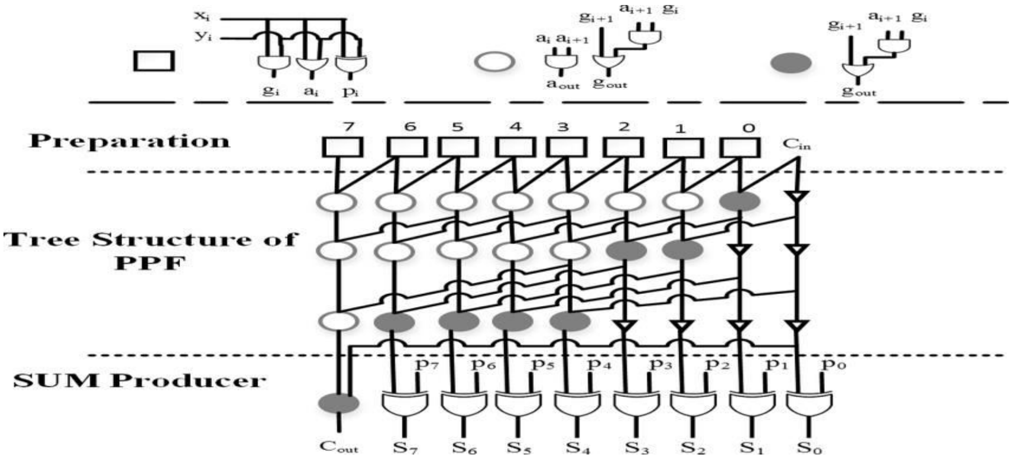
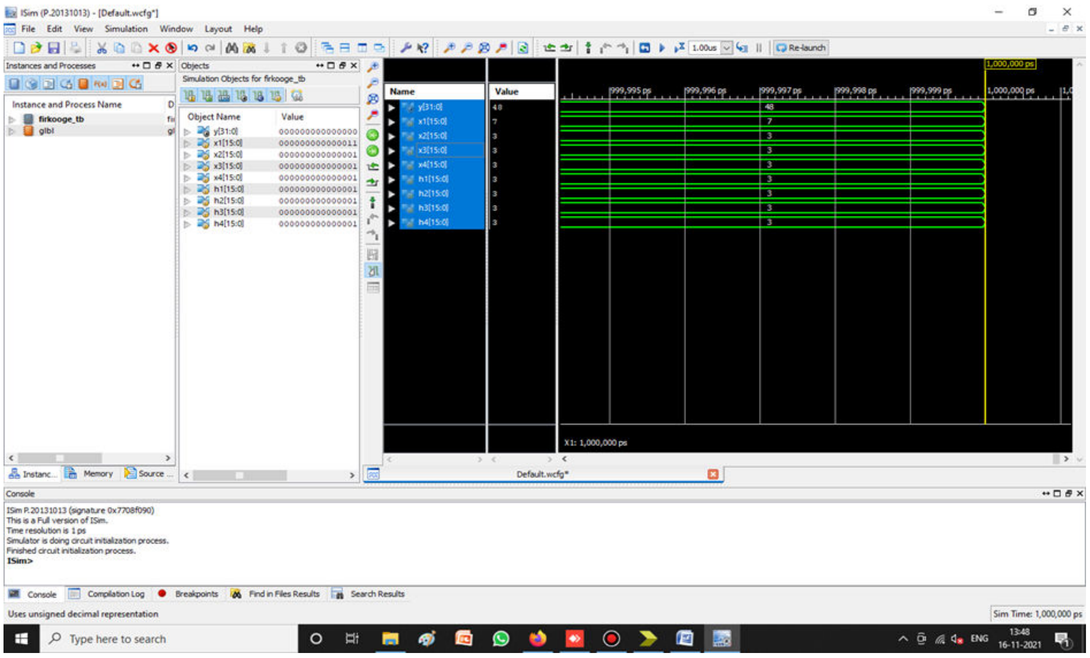

x_i
y_i
g_i
a_i
p_i
a_i a_{i+1}
g_{i+1} a_{i+1} g_i
a_{out}
g_{out}
g_{i+1} a_{i+1} g_i
g_{out}
Preparation
7 6 5 4 3 2 1 0
C_{in}
Tree Structure of PPF
SUM Producer
p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0
C_{out} S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0

Fig :516 Bit Kogge Stone Adder

The Kogge-Stone adder's construction is comparable to that of this adder's first level of the prefix tree. The second level of the prefix tree is where the primary structural difference starts. Two schematic node groups are constructed at the second level of the prefix tree, four schematic nodes are composed at the third level, eight schematic nodes are included at the fourth level, and so on.

RESULTS:

SIMULATIONRESULT OF FIR FILTER:


```
Simulation Objects for firkooge_tb
Object Name   Value
y[31:0]       0000000000000000
x1[15:0]      0000000000000011
x2[15:0]      0000000000000001
x3[15:0]      0000000000000001
x4[15:0]      0000000000000001
h1[15:0]      0000000000000001
h2[15:0]      0000000000000001
h3[15:0]      0000000000000001
h4[15:0]      0000000000000001

Name       Value
y[31:0]    48
x1[15:0]   7
x2[15:0]   3
x3[15:0]   3
x4[15:0]   3
h1[15:0]   3
h2[15:0]   3
h3[15:0]   3
h4[15:0]   3

X1: 1,000,000 ps

Console
ISim P.20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>

Uses unsigned decimal representation
Sim Time: 1,000,000 ps
```

FIG:6 Simulation Result For fir Filter

Here we can give the inputs as x1 = 7,x2 = 3 x3 = 3 x4 = 3,h1 = 3 h2 = 3 h3 = 3 h4 = 3 and result as y = 48.

RTL SCHEMATIC PF MULTIPLIER:

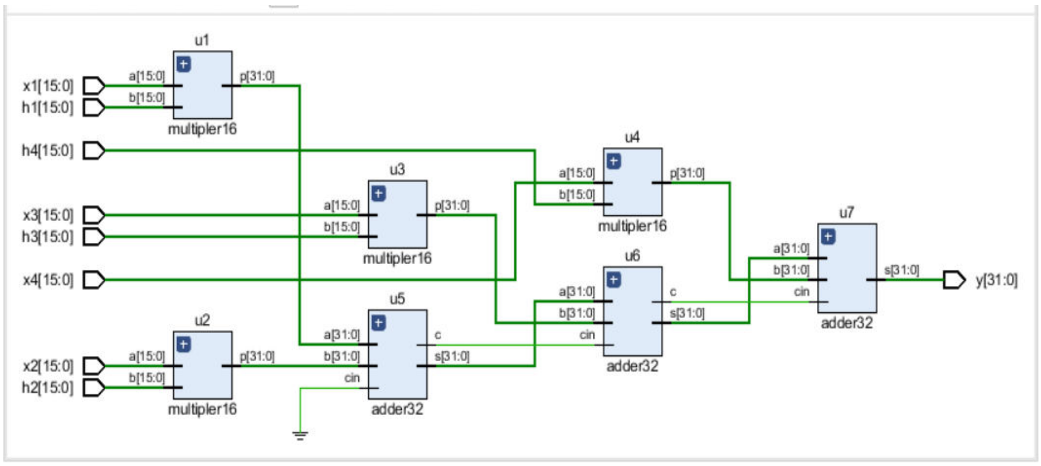


Fig:7RTL Schematic of Multiplier

Internal Diagram Of RTL Schematic:

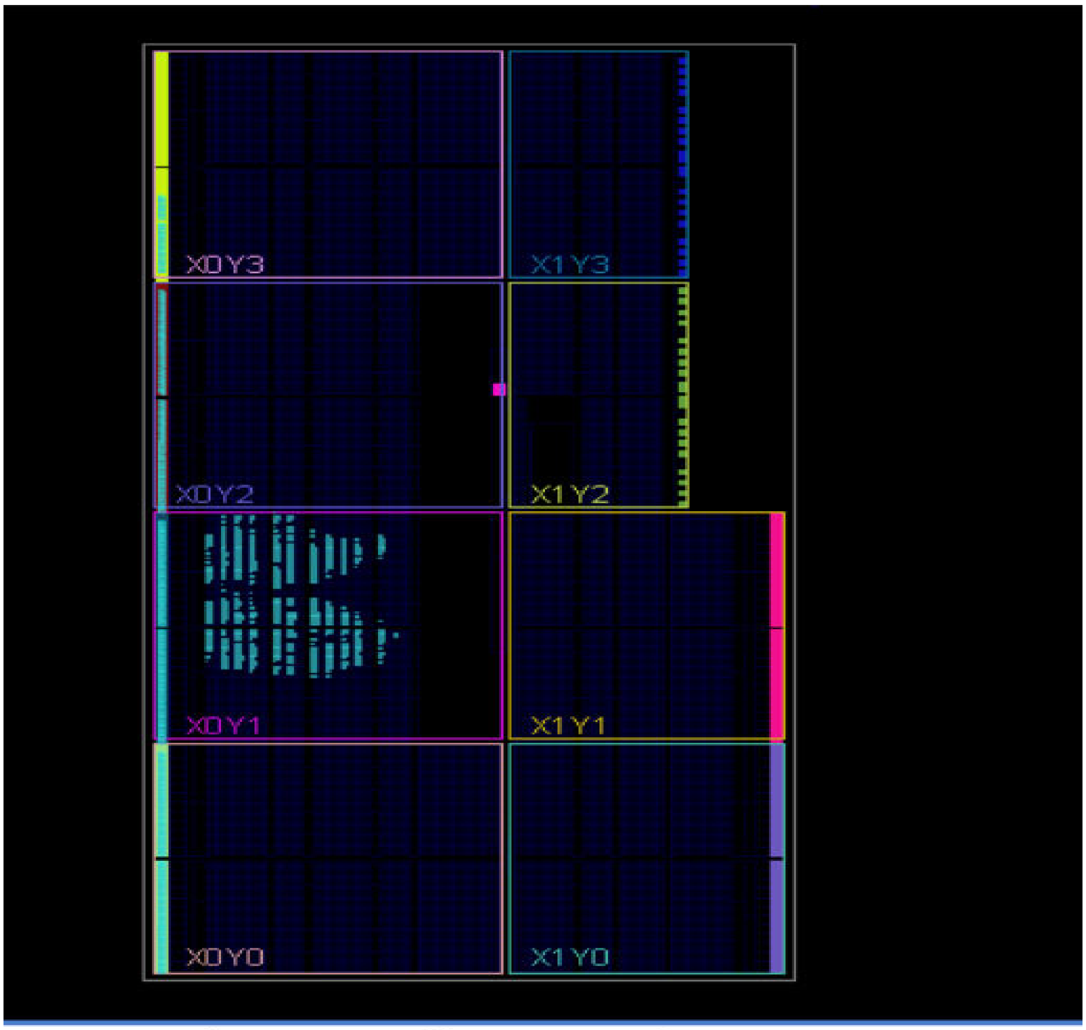
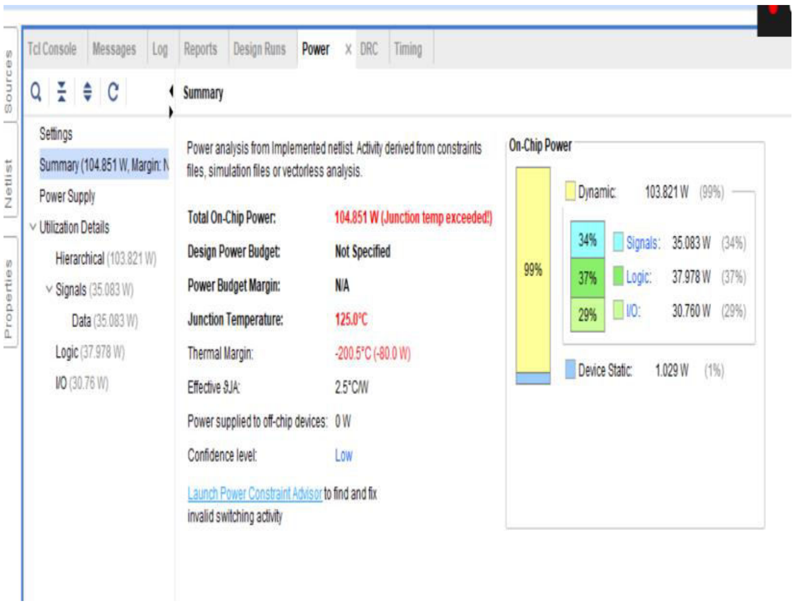


Fig:8 Internal diagram of RTL schematic

Estimation of power:



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

Total On-Chip Power:	104.851 W (Junction temp exceeded!)
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	125.0°C
Thermal Margin:	-200.5°C (-80.0 W)
Effective ϑJA:	2.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

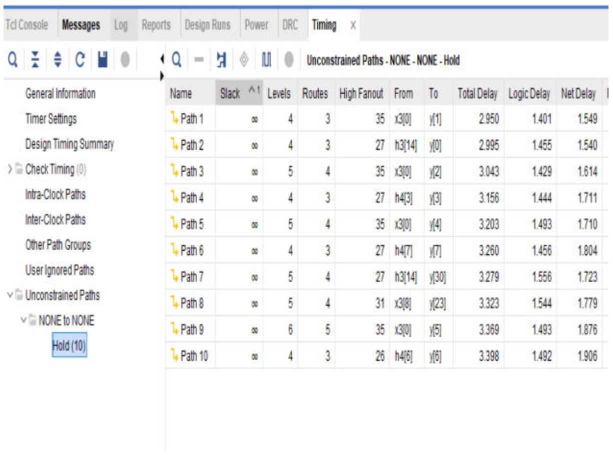
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

99%	Dynamic:	103.821 W	(99%)
34%	Signals:	35.083 W	(34%)
37%	Logic:	37.978 W	(37%)
29%	I/O:	30.760 W	(29%)
	Device Static:	1.029 W	(1%)

Fig:9 Estimation of power

Estimation of Delay:

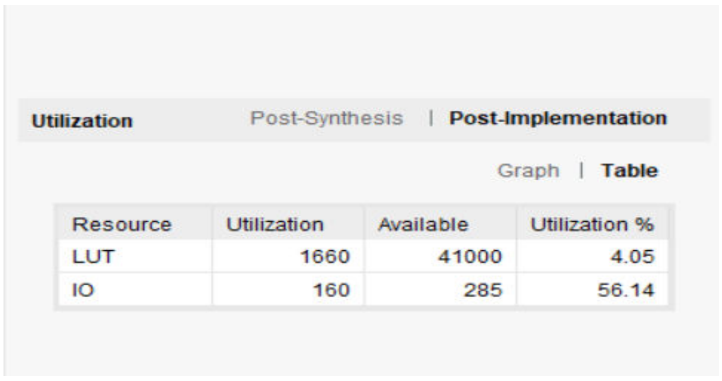


Unconstrained Paths - NONE - NONE - Hold

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	4	3	35	x3[0]	y[1]	2.950	1.401	1.549
Path 2	∞	4	3	27	h3[14]	y[0]	2.995	1.455	1.540
Path 3	∞	5	4	35	x3[0]	y[2]	3.043	1.429	1.614
Path 4	∞	4	3	27	h4[3]	y[3]	3.156	1.444	1.711
Path 5	∞	5	4	35	x3[0]	y[4]	3.203	1.493	1.710
Path 6	∞	4	3	27	h4[7]	y[7]	3.260	1.456	1.804
Path 7	∞	5	4	27	h3[14]	y[30]	3.279	1.556	1.723
Path 8	∞	5	4	31	x3[8]	y[23]	3.323	1.544	1.779
Path 9	∞	6	5	35	x3[0]	y[5]	3.369	1.493	1.876
Path 10	∞	4	3	26	h4[6]	y[6]	3.398	1.492	1.906

Fig:10 Estimation of Delay

Estimation of Area:



Resource	Utilization	Available	Utilization %
LUT	1660	41000	4.05
IO	160	285	56.14

Fig:11 Estimation of Area

Synthesis Report:

	Fir filter using sklansky	Fir filter using KSA
Area	**7355**	**7184**
Delay	**111.454ns**	**108.642ns**
Speed	**89.72Mhz**	**92.04Mhz**

Table:2 Comparison between power and delay

CONCLUSION

Xilinx ISE 14.7 was used to create the FIR filter and display the results. The KSA FIR filter's output is contrasted with the Sklanksky adder FIR filter's. After comparison, we discovered that the Kogge stone adder architecture FIR filter is faster than the Sklansky adder FIR filter.

REFERENCES

[1] Haichen Zhao, Shaolu Hu, Linhua Li, Xiaobo Wan. "NLMS AdaptiveFIR Filter Design Method".

[2] B. Ramkumar and Harish M Kittur, "Low-Power and AreaEfficient Carry Select Adder", IEEE Transsactions on Very Large ScaleIntegration (VLSI) Systems, VOL. 20, No. 2 Feb 2012.

[3] Deepak Kumar Patel, RakshaChouksey, Dr.MinalSaxena "Design ofFast FIR Filter Using Compressor and Carry Select Adder", 2016 3rdInternational Conference on Signal Processing and Integrated Networks(SPIN).

[4] A.Abinaya , M.Maheswari. "Implementation of Kogge Stone Adder forSignal Processing Applications", International Journal of AdvancedResearch in Computer and Communication Engineering Vol. 8, Issue 5,May 2019.

[5] S. Madhavi, K. Rasagna, N. Kavya, M. Sindhu. "Implementation ofprogrammable fir filter using dadda multiplier and parallel prefixadder",IEEE Xplore Compliant Part Number:CFP18N67-ART;ISBN:978-1-5386-2456-2.

[6] V. Jamuna, P. Gomathi and A. Arun, "Design and Implementation ofFIR Filter Architecture using High Level

Transformation Techniques"Indian Journal of Science and Technology.

[7] M.Moghaddam, M. B. Ghaznavi-Ghoushchi. "A New Low-Power, Lowarea,Parallel Prefix Sklansky Adder with Reduced Inter-StageConnections Complexity".

[8] Aung Myo San, Alexey N. Yakunin "Reducing the HardwareComplexity of a parallel prefix adder".