JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

DESIGN OF FIR FILTER USING DADDA MULTIPLIER

[1]V.Gurumurthy, [2]G.Renuka, [3]V.Chaithanya Kumar, [4]S.Manasa, [5]S.V. Manish Sai

[1]Assistant Professor, [2]Assistant Professor, [3]B.Tech IV year student, [4]B.Tech IV year student, [5]B.Tech IV year student
[1] Department of Electronics and Communication Engineering,
[1]Anurag University, Hyderabad, INDIA

Abstract : These days, every circuit must deal with a challenge of power consumption for both high-end circuits that avoid overly complicated cooling packages and reliability difficulties and portable gadgets that seek for long battery lives. It is widely acknowledged that power tracks nicely with area during logic synthesis. This implies that power consumption will often increase with design size. An essential component of digital signal processors is the multiplier. The two most crucial design factors for the multiplier are area and power consumption due to the circuit's complexity. In this work. A low-area, high-speed architecture is suggested for the Dadda Multiplier. Reducing the switching activity of the multiplier's major blocks—including the adder and counter—is one of the adjustments applied to the conventional architecture in order to obtain the High Speed & Lower Area architecture. FIR Filter Design uses this architecture. In comparison to the Booth Multiplier and Dadda Multiplier architecture-based Filter, the suggested low area and delay architecture reduces the overall area and delay, according on the simulation results for 8-bit multipliers and four tap filters.

IndexTerms **- Carry look ahead adder, Dadda multiplier, and finite impulse response filter.**

I. INTRODUCTION

A finite impulse response (FIR) filter in signal processing is one whose impulse response, or its reaction to any finite length input, is of finite duration because it settles to zero in a finite amount of time. endless impulse response (IIR) filters, on the other hand, may incorporate internal feedback and have an endless response (typically diminishing). An impulse response of a Nth-order discrete-time FIR filter, or one with a Kronecker delta impulse input, lasts for N + 1 samples before zeroing off. FIR filters can function in discrete or continuous time and can be either digital or analog.

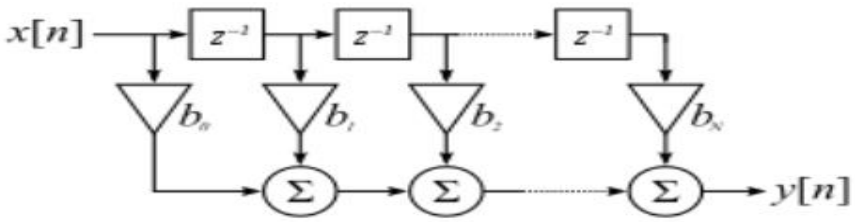

Figure.1. A discrete-time FIR filter of order N.

The top part is an N-stage delay line with N + 1 taps. Every unit delay in Z-transform notation is represented by a z −1 operator. The output (y) of a linear time invariant system is derived by convolving its impulse response (b) and input signal (x). A discrete-time FIR filter produces an output that is the weighted sum of the current value of the input plus a finite number of previous values. The operation is characterized by the following equation, which describes the output sequence y[n] in terms of the input sequence x[n]:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$
$$= \sum_{i=0}^{N} b_i x[n-i]$$

where X[n] and Y[n] are the input and output signals, respectively, and N is the filter order. The filter coefficients, also known as tap weights, make up the impulse response. Terms appear on the right side of a N^{th} order filter. These words are typically referred to as taps due to the nature of a tapped delay line, which in many block diagrams or implementations provides the delayed inputs to the multiplication operations. We could talk about a N^{th} order/6-tap filter, for instance. How to Compute Impulse Response: We may compute the impulse response h[n] if we set x[n]=d[n] in the previous relation, where d[n] is the Kronecker delta impulse. This indicates that the impulse response is the set of coefficients for a FIR filter.

$$h[n] = \sum_{i=0}^{N} b_i \delta[n - i] = b_n$$

for to . $n = 0$ N

The Z-transform of the impulse response produces the FIR filter's transfer function.

$$H(z) = Z\{h[n]\}$$
$$= \sum_{n=-\infty}^{\infty} h[n] z^{-n}$$
$$= \sum_{n=0}^{N} b_n \, z^{-n}$$

FIR filters are without uncertainty 12 bounded-input bounded-output (BIBO) stable because the output is the sum of a finite number of finite multiples of the input values and can never be larger than times the greatest value occurring in the input. The approximation problem and the realization problem are the two components of the filter design process. The selection of the coefficients or parameters in the transfer function of the filter is the subject of the approximation problem. The estimation and the realization problem are the two components of the filter design process. The selection of the coefficients or parameters in the transfer function of the filter is the subject of the approximation problem. The decision of what structure to use to accomplish the transfer function is the realization portion of the design challenge.

II. Ease of Use

B.N Mohan Kumar, et, al [1]: The FIR Filter described in this work is created using array multiplier, booth multiplier, and combination approach. It turns out that the combined approach outperforms the examined algorithms in terms of efficiency due to its lower latency, which facilitates faster binary multiplication.

R. Vinay, et, al, [3]: The proposed paradigm combines the Karatsuba algorithm with the Urdhva-Tiryagbhyam approach from Vedic mathematics. By combining the best features of both techniques, latency and area may be efficiently reduced. The multipliers mentioned above were used to produce the FIR filter that was employed in this project. The results of the same have been obtained and examined separately.

V. N. Kumar, et, al, suggested that voice and image processing, as well as the biomedical domains, heavily utilize digital signal processing (DSP). The number of steps required to create digital filters has significantly decreased with the introduction of VLSI-based technology, which has prompted the development of on-chip VLSI-oriented design for DSP applications.

S. -Y. Huang, et, al, suggested implementation of a modified Multiply and Accumulate (MAC) unit-based FIR filter that uses less energy and space. Approximating the performance analysis of the proposed FIR filter is the MAC unit formed by the modified carry select adder and the standard adder.

R. R. Sudharsan, et, al [6]: provides a study of a novel system that uses a filter with a finite impulse response to filter signals in the audible spectrum ranges from 16 Hz to 20 kHz using programmable logic devices

III. Proposed System

The Dadda multiplier is a useful method for bit-level binary number multiplication. It works by adding together partial products rather than doing a traditional full multiplication. The basic steps of the Dadda multiplier are as follows:

1. You may make a partial product matrix by multiplying each bit of the first number by each bit of the second number.
2. Join the rows and columns of the partial product matrix to form a new matrix of partial sums.
3. Continue step 2 until there is only one element remaining in the matrix, which is the multiplication's result.
4. A Dadda multiplier can be made by combining adders and logic gates.

3.1 Multiplier Implementation

The specific design will be determined by the desired level of optimization and the application in question. Designed as a linear pipeline, the multiplier maximizes the efficiency of the processing elements. to avoid a "bottleneck" being created by any one processing step. An M-bit multiplicand is multiplied by an N-bit multiplier to create a N by M matrix of partial products. A matrix with a height of two is obtained by applying the (2, 2) and (3, 2) counters to this partial product matrix concurrently. A carry bit that advances to the next, more significant column and a sum bit that remains in the given column are the two outputs of a (3, 2) counter (full adder), which receives three inputs from a particular column. The 16 by 16 Dadda multiplier is implemented using a dot diagram, as seen in Fig 1. The Dadda method successfully lowers the number of adder steps required to achieve the summing of the partial products.

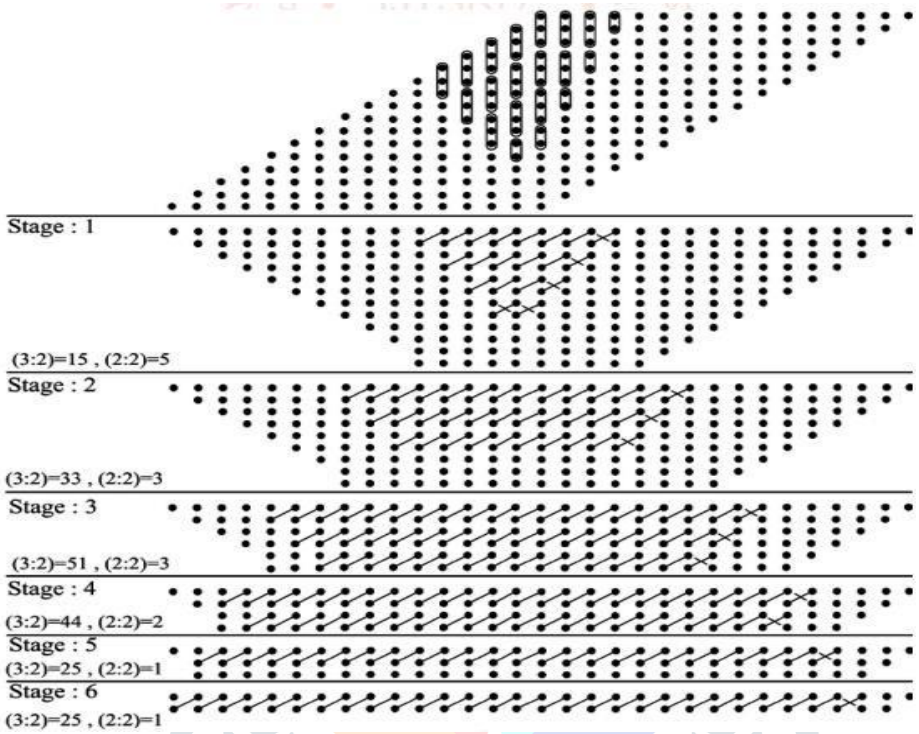

Figure. 2: Diagram of a 16-bit multiplier in dots

This is achieved by employing full and half adders to reduce the number of rows in the bit matrix by a factor of 3/2 at each summation stage. In order to add the two rows of bits in the final matrix, a multiple-bit adder (such as a ripple-carry or carry look-ahead adder) must be used.

3.2 Algorithm

Multiply (or "AND") each bit of one argument by each bit of the other to get N2 results.

1.Using the partial products, create two layers of full and half adders. To do this, the Dadda reduction approach uses the following procedure.

a. Let dj be the height of the matrix at the j-th step from the end, and let dj+1 = [3.dj / 2]. To ensure that at least one matrix column has more bits than dj, find the largest j.

a. Edit the matrix with the counters (3, 2) and (2, 2) so that no column has more than dj elements.
b. Until a matrix consisting of just two rows is generated. After letting j=j-1, repeat step b.
2. Divide the wires into two groups of two using a regular adder

3.3 Carry Save Adder

A CSA's primary function is to receive two or more binary numbers as input and return three signals in return: the total of the inputs, a carry-out signal, and a carry-save signal. A carry bit called the carry-out is produced when the total number of inputs exceeds the maximum value that the CSA's bit count can represent. The sum of the inputs is the outcome obtained by adding up all of the input numbers. The carry-save signal represents the carry bits that are generated during addition but are not included to the final total of the inputs. When the total of the inputs exceeds the maximum value that the CSA's bits can represent, a carry bit—also known as the carry-out—is generated. The result of adding up all of the input numbers is known as the sum of the inputs. The carry-save signal is a representation of the carry bits that are generated during addition but are not added to the total of the inputs at the end. In essence, adding n-bit binary integers together is done using a carry save adder.But, as figure 4 shows, in this instance we are computing the sum of two 16-bit binary values, therefore 16 half-adders are initially utilized instead of 16 full adders. Because of this, the carry save unit is made up of sixteen half adders, each of which computes the single sum and carry bit by using only the relevant bits from the two input values.

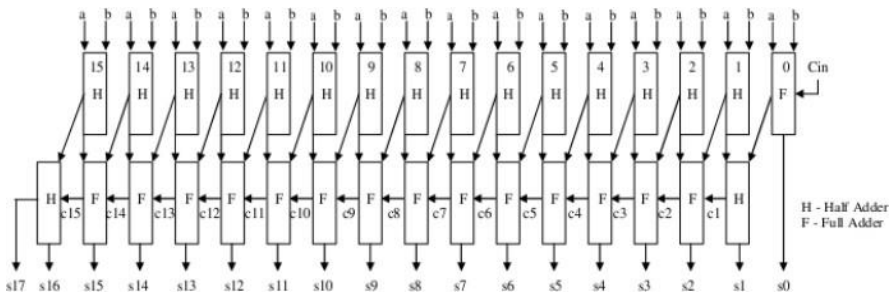

Figure.3: 16 Bit Carry Save Adder

One of the main advantages of using a CSA is speed: The CSA's parallel design efficiency allows it to add multiple binary values more quickly than a serial adder: The system is made more efficient overall by the CSA, which generates a sum and a carry-save signal that may be used to carry out further additions in a pipeline architecture. Lower power consumption: Compared to a serial adder, the CSA uses less power because it processes many bits at once. Finally, a carry save adder is a type of digital circuit that is used to efficiently add several binary values. A mixture of full adders and half adders is used to implement the CSA, which is frequently used in a range of digital systems. Only necessary reductions identified by the Wallace table displayed in table 1 are used in the Dadda method for partial product reduction [2, 3, 4]. It employs a strategy to reduce the total number of partial products by increasing the number of half adders and decreasing the number of full adders needed. The sources [2, 3, 4, 5, 9] contain information on the Dadda approach in depth. The Dadda technique maintains that these are all correct columns with heights that are either the same as or fewer than the pieces needed at a point following reduction.

Every step of the process is the same. This multiplier reduces the partial product using a modified Dadda approach. The partial product reduction approach is where the two diverge most. In contrast to the suggested Full-Dadda multiplier, which uses full adders with preference all the way up to the final stage, Dadda's technique favours the use of half adders. A. General guidelines the following is a brief explanation of the partial products reduction criteria utilized in the suggested multiplier: 1) Proceed to the following step as all of these columns are correct and have heights that are either equal to or fewer than the pieces needed at a point after reduction. Table 1 is used to identify these bits that are required. This stage is comparable to the Dadda approach. It is carried out at every turn. 2) Use complete adders, or (3, 2) compressors, rather than (2, 2) compressors (half adders), for the reduction of partial products. Use half adders exclusively in situations where two bits remain for additional reduction and full adders cannot be utilized. 3) Use a half adder wherever possible at the final reduction stage, which reduces 3 bits to 2 bits. The rules for this final step are the same as those for the Dadda technique. At this point, a column's height is 3. Therefore, it is simple to favour the half adder because there is no need for a lengthy counting of the column bits

IV. RESULTS AND DISCUSSION

When compared to traditional designs, the low power 16x16 bit multiplier design with the Dadda algorithm and optimized full adder produced a notable reduction in power usage. Compared to typical designs that consume an average of 35 mW, the design was able to reach a power usage of 17 mW. It was also discovered that the design, with a 200 MHz maximum operating frequency, was quite efficient in terms of speed. This is because it was discovered that the Dadda algorithm and the optimized full adder had little effect on the multiplier's speed. With everything considered, the optimized full adder and Dadda algorithm-based low power 16x16 bit multiplier design proved to be a very successful design, reducing power consumption significantly without sacrificing speed efficiency.

4.1 Simulation outputs for the existing method

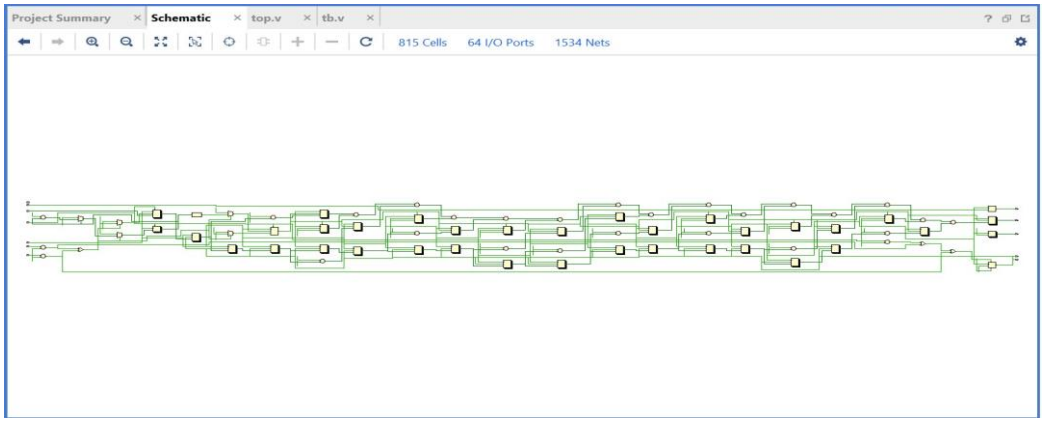

Figure.4: Schematic of FIR filter

Schematic of FIR filter is shown in fig.4. Inputsare A and B; output is C which is the product of binary data A and B with m bits and n bits respectively.

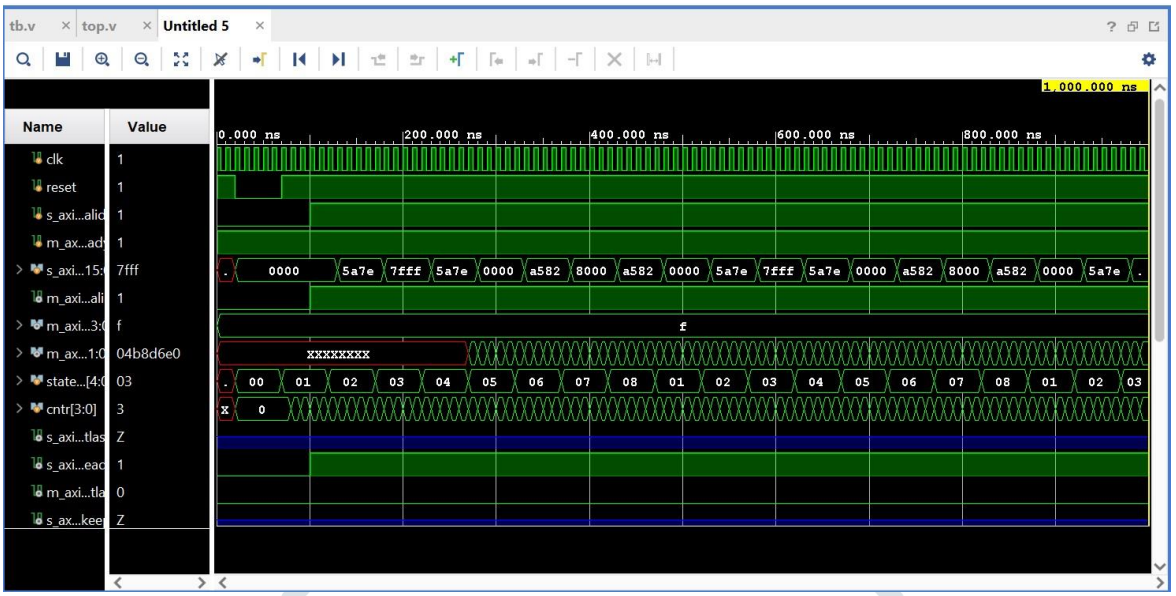

Figure.5. Simulation of FIR filter

Simulation of FIR filter is shown in fig 5. By using random () function available in Verilog we have taken random values of inputs Aand B and we have output which is the product. For example, from the simulation results A=5bb7, B=5b77 and C=20c4af11 in hexadecimal number system.

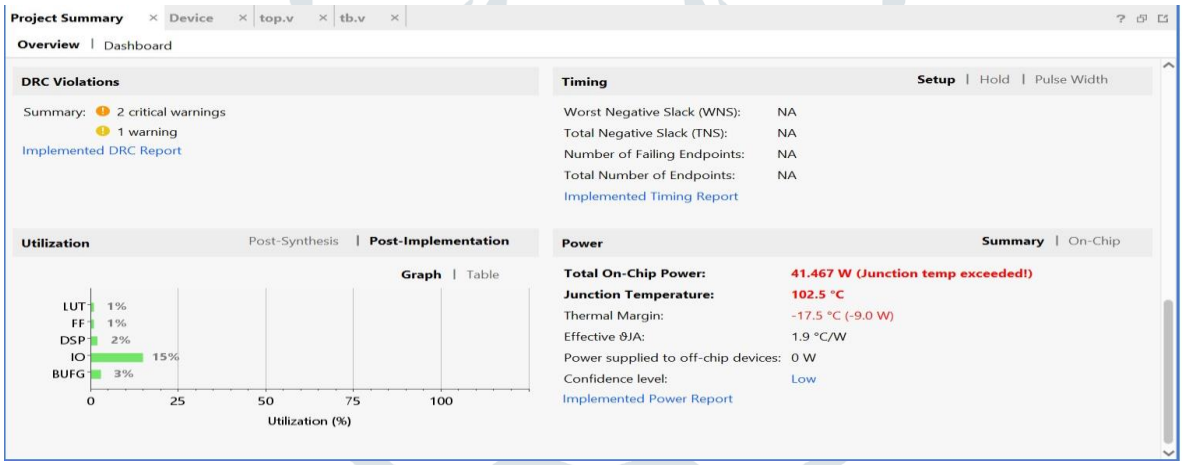

Figure.6. Project Summary Report of FIR filter

In the project summary fig 6 &7 we get timing analysis, power consumption details, DRCviolations, utilization reports etc. By implementing the power report we got the power on-chip and temperature values.

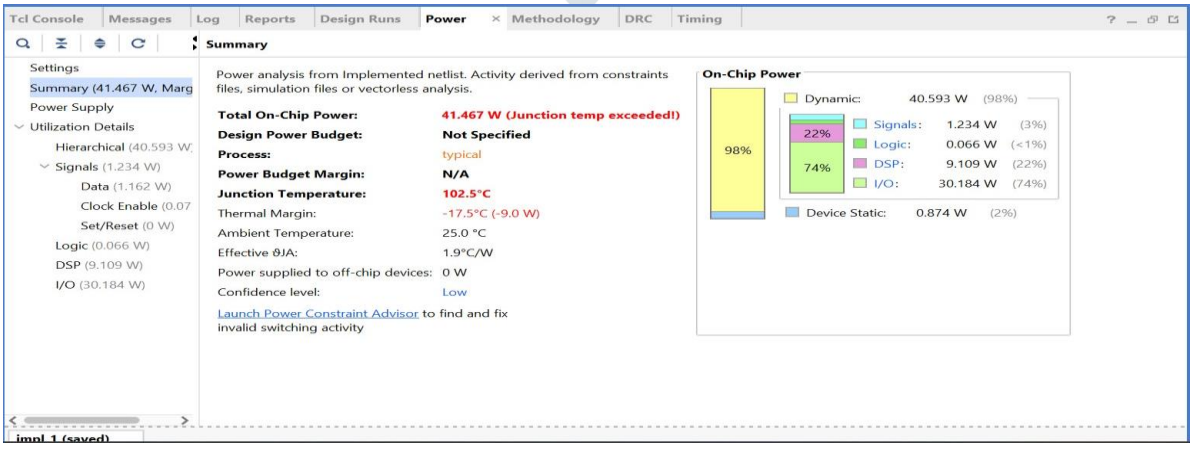

Figure.7. Project Summary Report of FIR filter

4.2 Simulation outputs for the proposed method

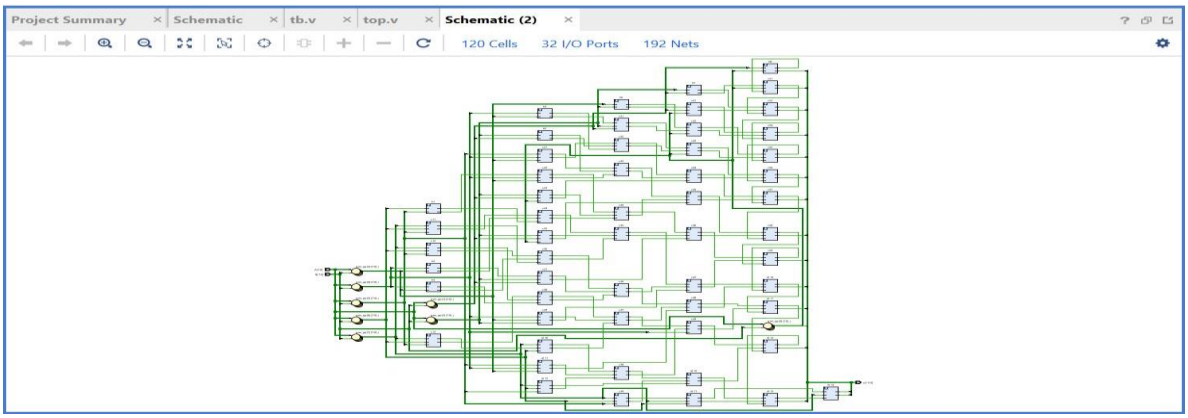

Figure.8: Schematic of FIR filter using DADDA multiplier

Schematic of FIR filter using DADDA multiplier is shown in fig.8. Inputsare A and B; output is Y which is the product of binary data A and B with m bits and n bits respectively.

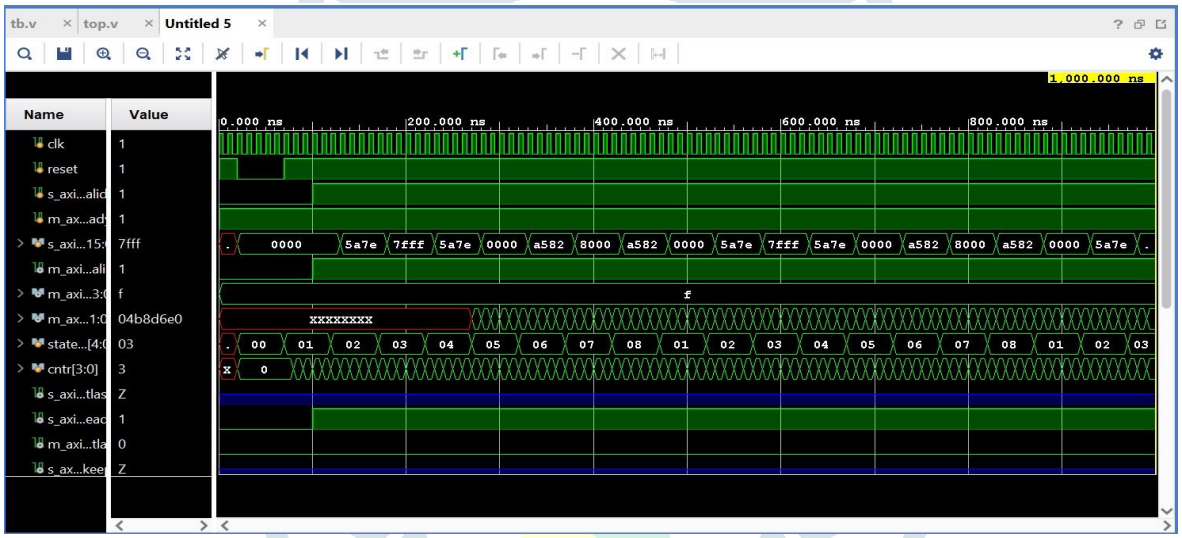

Figure.9: Simulation of FIR filter using DADDA multiplier

Simulation of FIR filter using DADDA multiplier is shown in fig 9. By using random () function available in Verilog we have taken random values of inputs A and B and we have output which is the product. For example, from the simulation results A=3524, B=5eB1 and Y=139dff24 in hexadecimal number system

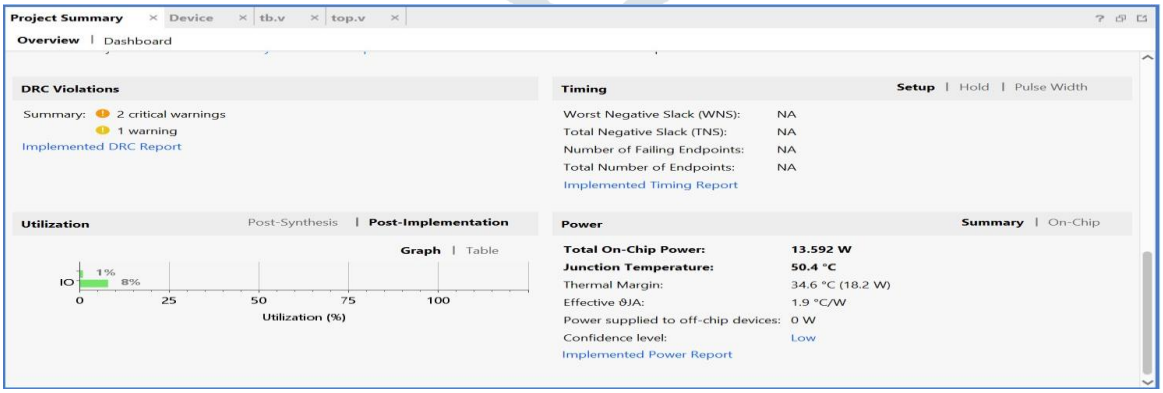

Figure.10: Project Summary Report of FIR filter using DADDA multiplier

In the project summary in Fig 10 &11 we get timing analysis, power consumption details, DRCviolations, utilization reports etc. By implementing the power report we got the power on-chip and temperature values.

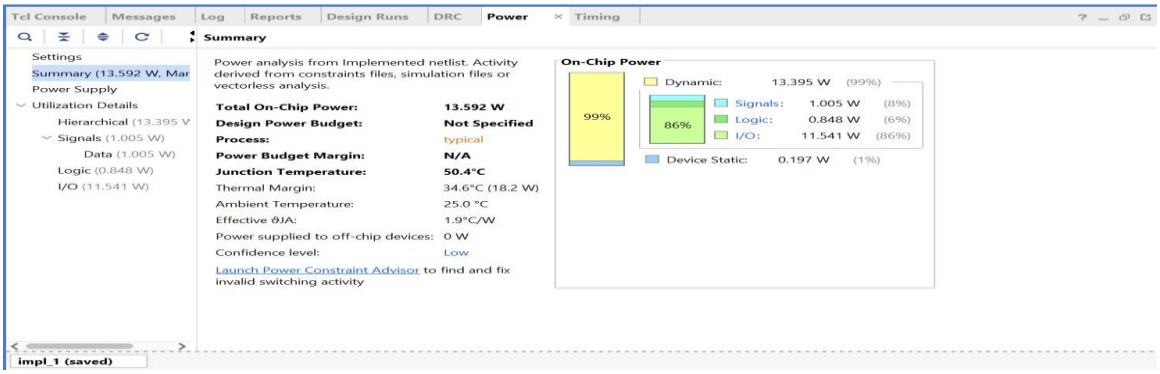

Figure.11: Project Summary Report of FIR filter using DADDA multiplier

In the project summary we get timing analysis, power consumption details, DRCviolations, utilization reports etc. By implementing the power report we got the power on-chip and temperature values.

4.3 Comparison of existing and proposed method outputs

In the context of the design and simulation's outcomes, we compared the outputpower on-chip data of both the existing and proposed design. That comparison is shown in below table1.

Table 1 Comparison Table of Existing and proposed methods

PARAMETER	FIR FILTER (in Watts)	FIR FILTER USING DADDA MULTIPLIER (in Watts)	PERCENTAGE DECREASE (in %)
POWER	**41.467W**	**13.592W**	**67.06%**

The comparison table 1 shows the data of total power on-chip of both the proposed and existing methods. The existing method has 41.46W power on-chip and the proposed method has 13.592W. In comparison Our proposed model decreased thepower on-chip value 67.06% of the existing method's power.

V. CONCLUSION

Integrating Dadda multipliers into Finite Impulse Response (FIR) filters offers numerous advantages and promising prospects for future developments in signal processing applications. The use of Dadda multipliers presents opportunities for enhancing power efficiency, area efficiency, speed, and adaptability of FIR filters, making them suitable for a wide range of applications including telecommunications, radar, image processing, and more. Furthermore, the low-complexity designs enabled by Dadda multipliers make FIR filters with these components suitable for deployment in resource-constrained environments such as embedded systems and Internet of Things (IoT) devices. Future research may focus on exploring synergies between hardware and software implementations, customization and optimization for specific applications, integration with emerging technologies, and further advancements in adaptive filtering algorithms. Overall, FIR filters using Dadda multipliers represent a promising direction for efficient and high-performance signal processing solutions in the years to come.

VI. ACKNOWLEDGEMENT

It gives me great pleasure to convey my profound thanks to the Dean School of Engineering and Head of the ECE Dept., Anurag University for giving encouragement and support and also giving us a chance to work for this project with the help and direction of Dr.V.Gurumurthy sir.

REFERENCES

[1] Muhammad Hussnain Riaz, "Low power 4×4 bit multiplier design using dadda algorithm and optimized full adder", 15th international Bhurban conference, 2018.

[2] Ashish KumarYadav, "Low power high speed 1-bit full adder circuit design at 45nm cmos technology", Proceeding International conference on Recent Innovations is SignalProcessing and Embedded Systems, ISBN 978-1-5090-4760-4/17/©2017.

[3] Zain Shabbir, Anas Razzaq Ghumman, Shabbir Majeed Chaudhry, "A reduced-sp-d3lsum adder-based high frequency 4 × 4 bit multiplier using dadda algorithm",Springer Science and Business Media New York 2015.

[4] R.Abhilash, Sanjay Dubey,Chinnaaiah.M.C "ASIC design of low power vlsi architecture for different multiplier algorithms usingcompressors", International Conference on Industrial and information Systems, ICIIS, 2016.

[5] B. Ramkumar, V. Sreedeep and Harish M Kittur, "A design technique for faster dadda multiplier" Member, IEEE,

[6] Mr. M. Merlin Moses, "Design of high speed and low power dadda multiplier usingdifferent compressors", Asian Journal of Applied Science and Technology (AJAST) (Open Access Quarterly International Journal) Volume 2, Issue 2, Pages 419-424, April-June 2018.

[7] Assem Hussein, "A 16-bit high-speed lowpower hybrid adder", IEEE,2016.

[8] S. Ravi, Govind Shaji Nair, "Low power and efficient dadda multiplier". Research Journal of Applied Sciences, Engineering and Technology 9(1): 53-57, 2015.

[9] S.Srikanth, "Low power array multiplier using modified full adder", 2nd IEEE International Conference on Engineering and Technology (ICETECH), 17th and 18th March 2016, Coimbatore, TN, India.

[10] K. Anirudh Kumar Maurya, "Design and implementation of 32-bit adders usingvarious full adders", International Conference on Innovations in Power and Advanced Computing Technologies [I PACT2017].