International Journal of Engineering Innovations in Advanced Technology

ISSN: 2582-1431 (Online), Volume-4 Issue-4, December 2022

Efficient Design for Fixed-Width Adder-Tree

G santhoshini

MTech Student, Department Of ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., India. Email: gundeboinasanthoshini@gmail.com

Dr. S. Kishore Reddy

Associate professor, HOD, Department Of ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., India. Email: kishorereddy416@gmail.com

P V Raju

Assistant professor, Department Of ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., India. Email: vijnatharaju@gmail.com

***Abstract* - There are many applications where multiplication is essential, including multimedia processing and artificial neural network (ANN. Multiplier is a substantial contribution to the energy usage, critical path latency, and resource usage in these applications. In FPGA-based designs, these impacts are more severe. ASIC-based systems, on the other hand, are the most up-to-date designs. Furthermore, the few Device designs that do exist are usually restricted to unsigned integers, requiring additional circuitry to provide signed operations. This work proposes an area-optimized, low-latency, and energy-efficient design for an accurate signed multiplier for FPGA-based solutions of applications that use signed integers. In order to speed up a multiplier, the best strategy is to limit the amount of incomplete products. With a modified Booth multiplier, adjustable arithmetic capacity and trade-offs in output accuracy are achieved.**

Keywords- ASCI, ANN, FPGA, Arithmetic capacity, Multiplier.

1.INTRODUCTION

Because of the increased complexity of VLSI design and application today, the manual approach of design is no longer a viable option. Automated design has become the standard practice. The present level of VLSI technology may be attributed to the rapid technological breakthroughs that have place over the course of the last two decades. Continuous growth in both the size and functionality of integrated circuits (ICs): The size of individual features has been steadily becoming smaller, which has resulted in a rise in both the speed and the gate device transistor density. Additionally, there has been a slow but steady improvement in the predictability of the behavior of circuits. Because of the aforementioned developments, there has been an explosion in the number of different approaches to VLSI design.

In the 1970s, when very large-scale integration (VLSI) was invented, complicated technologies for both semiconductors and communications were being developed. The VLSI technology was used in the construction of the microprocessor. Because of the dramatic increase in complexity brought about by the inclusion of hundreds of transistors in modern processors, this expression is not used nearly as often as it formerly was. This industry is becoming more obsessed with cramming more logic devices into ever-smaller spaces. At this time, very large scale integrated circuits can be built in very compact devices in a few of millimeters. VLSI circuits are used in a wide variety of modern electronic products including computers, automobiles, digital cameras, and mobile phones.

Interconnection variations: Because a huge number of transistors needed to be packed into a single chip in order to produce a functional design, it was necessary to approach this work with great attention to detail in terms of planning.

Using the technology that was available at the time, it was impossible to fit more than a few transistors on a single integrated circuit chip because of its vast size and the poor manufacturing yields that were available at the time. The design was simple because to the low number of links that were present. Because there are now a substantially greater number of transistors on a single chip than there were in the past, it is now significantly more difficult to create an efficient design.

Manuscript received Dec 10, 2022; Revised Dec 25, 2022; Accepted Dec 31, 2022

A. SSI Technology

Only very few transistors were used in the initial integrated circuits. Digital circuits with transistor counts in the tens offered a few logic gates, whereas early linear ICs like the Plessey SL201 or perhaps the Philips TAA320 contained as little as two transistors, a process known as "small-scale integration" (SSI). IBM engineer Rolf Landauer originally introduced the phrase "Large Scale Integration" while discussing the theoretical notion from which the names "SSI," "MSI," "VLSI," and "ULSI" were derived.

B. MSI Technology

Devices that had hundreds of transistor on each chip were known as "medium-scale integration" when they were first developed in the late Sixties (MSI).

C. Large Scale Integration

The same economic forces that resulted in "large-scale integration" (LSI) in the middle of the 1970s also resulted in thousands of transistors being packed onto one chip. Fewer than 4000 transistors were included in the first integrated circuits, which were manufactured in very limited numbers in the 1970s. In 1974, the first LSI circuits with more than 10,000 transistors were built for primary memory in computers and second-generation microprocessors. These circuits were developed in the United States.

As of 2009, very large-scale integration (VLSI), which was first created in the 1980s and is still being developed today, consists of more than a few billion transistors.

In 1986, consumers had the option of purchasing memory devices that included more than a million transistors. In 1989, a single computer chip included one million transistors; by 2005, that number had increased to one billion. Since 2007, when the first tens of billions of memory transistors were put into use, this pattern has been consistently maintained.

D. VLSI Design Flow

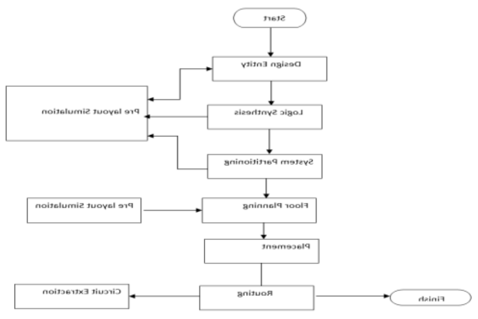


Fig 1.1: VLSI design flow

2. LITERATURE SURVEY

A. MULTIPLIERS

Multipliers are essential to the operation of a great number of extra applications in addition to digital signal processing. A substantial amount of focus and effort has been directed into the research and development of multipliers that are capable of achieving any of the following goals.

1. little power consumption,

2. lightning-fast speed

3.Due to the fact that they are so regular and take up such a little space, they are suitable for a wide variety of high-speed applications.

4. The use of very low power and very small integrated circuit technologies.

The "add and shift" method of multiplication is by far the most frequent approach. The performance of a parallel multiplier is significantly impacted when there is a greater number of sub-blocks that need to be added. The results of increasing parallelism and moving the partial products of intermediate sums that need to be added are a reduction in speed, an increase in the amount of silicon space required, and an increase in the amount of power required because of the increased number of connections caused by complex routing. On the other hand, multipliers that use the "serial-parallel" architecture sacrifice speed in order to save both space and power. It is possible that a parallel multiplier might be selected over a serial multiplier depending on the application in question. During this presentation, several multiplication algorithms and architectural frameworks will be discussed. We compare them using performance indicators such as the efficiency of their use of space, the amount of electricity they use, and the total amount of power they produce. It is possible to make the partial products by using AND gates (PP). You could find out how many partial products you have by multiplying an N-bit integer by an M-bit number. This might be done, for example, in Excel.

B. IMPLEMENTATION

An 8-bit accumulator is required to hold the result of the four bit multiplicand registers and the four bit multiplier registers that are included inside the MAC unit. The 8-bit result is stored in the product register, which can be seen in the picture. First, the multiplicand or even zero is shifted, and then, in the same manner as conventional binary multiplication, the result is added. This behavior is dependent on the multiplier bit. In order to do the same task, an adder with 8 bits would be required to be used. On the other hand, this configuration adds a multiplicand with a value of 5 and shifts the contents of the product register to the right by one space. The processing of multiplier values sequentially while adding in parallel is referred to as a "serial-parallel multiplier," and it is described by this term. A parallel array multiplier is the second kind of multiplier that may be used.

Attempts were made to speed up the pace of output production, which led to the development of this form of multiplier. The data in the serial-parallel multiplier is processed one bit at a time, one clock cycle at a time, as was previously described. An input of N bits would need about N clock cycles before it could be converted into the final output. As soon as the multiplier's inputs are provided, the result may be found in the output. The majority of the responsibility for this problem may be placed on the calculation of the 1-bit terms using an AND array structure. Because the adders have to total these partial product terms column by column in order to obtain the result, the process takes a little bit more time than it would otherwise. The illustration that may be seen below depicts a parallel array multiplier that has N equal to 8 bits.

3. EXISTING SYSTEM

It is shown how to make array multipliers by making use of a two-operand adding circuit. This circuit combines the production of radix-2 partial products with addition operations. Modern FPGAs manufactured by Xilinx may make use of this circuit if the LUT design in question has six inputs. As a result of the lower LUT footprint that array multipliers have, it is possible for them to be deployed in the same logic fabric as equivalent Logic core IP multipliers. When pipelined to a substantial degree, these multipliers are often slower than the multipliers found in Logic core IP.

Why It takes up more space to produce the signed multiplier, so be sure you have enough. It will take much longer.

A. Architecture of A Booth Multiplier

After all of the partial products have been computed in parallel, this configuration adds the resultant partial products with the help of a rippling carry adder and a number of 4:2 compressors (RCA). The critical route time of the multiplier may be greatly cut down when concurrent manufacture of partial products is used. The length of the carry chain in a N x M multiplier is always N+4 bits for each bit row. It is possible that the length of the carry chain might be shortened to N+1 bits in order to enhance the critical path latency of the multiplier. The perfect application of our brand new multiplier's essential route delay is shown in the following picture. Each row of partial products needs two bits of multiplicand for the partial product expressions, with one bit reserved for the pp(x, 0) operation and the other two bits devoted to the pp operation (x, 1). Using a lookup table (LUT) with six inputs allows for the feasible execution of both of these partial product phrases. Each row of the partial product may have its own unique implementation of the pp(x, 2) function by making use of a second 6-input LUT. To calculate the appropriate input carry for each bit row, one may make use of a separate 6-input LUT that was given the designation "CG." The multiplier that we have presented is superior than multipliers that are already in use since it has a higher level of precision and accuracy.

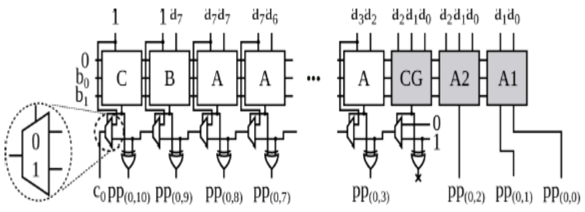


Fig.3.1: Architecture of Optimized version of multiplier

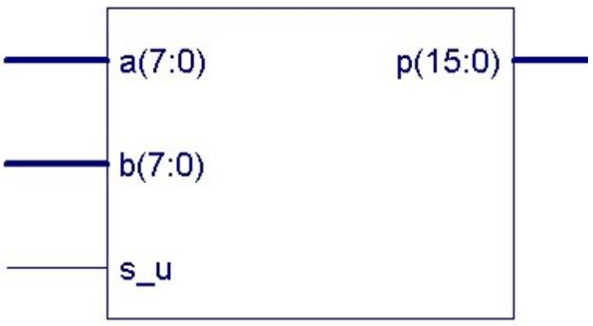


Fig.3.2: Block diagram of 8-bit booth multiplier

B. Partial Product Generation

In this work, signed-and-undated booth encoding multiplication is presented. Only signed values may be utilized with the modified booth encoding multiplier or the Baugh Wooley multiplier. When using the Braun array multiplier, unsigned values are the only kind of value that may be multiplied. When multiplying fixed numbers, every multiplier works in the same way. However, on current processors, we only need a single multiplier, regardless of whether the integers being multiplied are signed or unsigned. Efforts are being made to find a solution to this issue. The SUMBE multiplier may be found on the paper. One half of the partial product is represented by a revised Booth encoder, which acts as a parallel representation of the circuit for the other half. As a direct result of the expanded operand signal bit, the SUMBE multiplier will produce one more partial product than usual. The Carry save adder (CSA) tree and the final carry look ahead will be used in order to quicken the acceleration of the multiplier (CLA). Using the MBE approach, this multiplier is able to handle signed and unsigned data equally well.

4. PROPOSED SYSTEM

It has previously been established that the multiplier is a very important component of microprocessors, microcontrollers, and graphics systems. The multiplier level that was provided by Booth Furnisha serves as the foundation for the algorithms that will be utilized in the future generation of multipliers. These multipliers will have improved performance and quicker speeds. By using a small-scale function, negatively oriented fragmented products are ejected from the system. When it comes to first-level booths, making use of a multiplier results in a more reliable coding standard. The topic of this study is partial products, both positive and negative, in their many forms. The key aims of this study are to develop more efficient processes and to cut down on the number of incomplete goods that are generated.

The effort that was made during the research project to solve the problem of negative partial products. Calculating the 2's complement requires bypassing the extra adding1 step and generating a lengthy carry chain for this assignment. Reduce the unfinished product to the smallest possible size using the design that was supplied. Modified By using Booth's technique n/2, which outperforms n/2+1 partial products, the number of partial products may be reduced, hence improving overall performance. Verilog HDL is used to create the actual workings of the proposed system. In order to put the suggested plan into action, it is necessary to be aware of the total amount of bits that will be multiplied. There are many different approaches of putting a design through its paces. Xilinx's own synthesis tool is what's utilized for the actual synthesis process.

The technique of multiplication for producing a more useful partial product is the subject of this study's investigation. There is no longer a need for additional adders to perform an addition of 1'b1 at each step when the suggested approach requires a 2's complement since it seeks to prevent the construction of extended carry chains. It is feasible to produce the radix 4 multiplier with the same hardware by employing the ways that have been described, while simultaneously lowering the number of partial products to n/2+1 from the original n. The approaches that have been discussed work very well with a dataset that is 16 bits by 16 bits, but they can also be used to larger datasets that have greater bit counts.

Utilizing high-radix Booth encoding is an alternate way that may be used to cut down on the amount of goods that are missing necessary components. On the other hand, hard multiples, which aren't powers of two and can't be created by shifting and/or complementation, are becoming more commonplace. Hard multiples are defined as multiples that can't be obtained in this way. Besli and Desmukh came to the conclusion that some difficult multiples may be computed by subtracting two simple power-of-two multiplies from each other. The problem of hard multiples may be solved by using a radix-4-Booth coding scheme (also known as RBBE-4) that does not make use of ECW. It is possible to eliminate the hard multiple issue as well as the additional ECW by using a radix-2 RB Booth encoder; however, this requires using twice as many rows of RBPP. Both the MBE represented by radix-2 RBPP and the MBE represented by radix-4 RBPP contain the same amount of rows. Although the output of the RBPP generator and the MBE generator can be the same, the RBPP generator has a more complicated circuit design and runs at a slower speed than the MBE generator.

By using the RBPP generator, we are able to create the same number of total product rows in a 2n-bit RB multiplier while simultaneously reducing the number of partial product rows produced. MBE-based RBMPPG-2 is the name of the new product generator that is based on modifying existing partial products (RBMPPG). In the RBMPPG-2 that was designed, the ECW from each row is moved to the row that comes after it in the next column. This extra ECW is coupled with the ECW of the very final partial product row, in addition to the two MSBs, which are comprised of the very first partial product row and its two LSBs. This further simplifies the reasoning (the second partial product row). As a direct result of this, an RBPP accumulation stage is maintained despite the fact that the number of RBPP rows was decreased from N=4 1 to N=4. When compared to typical designs, the multiplier architecture being proposed needs each operand to be at least 32 bits long. This results in considerable savings in both space and power when the operand word length is at least 32 bits. Even though there is a 5% increase in delay for words that are larger than 32 bits, the suggested designs are still optimal in terms of power-delay product (PDP) at these lengths.

The next paragraphs will offer an explanation of the structure of this paper. In the second part of this chapter, the radix-4Booth encoding is discussed. The structure of the traditional RBPP generator is likewise subjected to intensive research and development. The RBMPPG diagram may be found in Section 3. The RBMPPG that has been suggested is presented in this section and is used by a variety of different word-length RB multipliers. New RB multipliers based on the proposed RBMPPG are evaluated and compared with prior best designs found in the technical literature for a variety of word lengths. These evaluations and comparisons are carried out.

A. Radix-4 Booth Encoding

In order to simplify the process of multiplying binary values with a two's complement, the Booth encoding has been devised. It was first implemented as either radix-4 Booth encoding or modified Booth encoding. Table 1 provides a concise overview of the MBE system. There are three sets of three multiplier bits that are next to one another. The two side bits overlap with the groups that are next to them, with the exception of the group that contains the first multiplier bit. According to Table 1, each group may be accessed by choosing the partial product that is shown there. In this context, the symbol 2A denotes the double of the multiplicand, which can be acquired by left-shifting the supplied value. By inverting each and every piece a The simple method is accomplished when a "1" (the correction bit) is added to the least significant bit (LSB). It has been suggested that correction bits might be used as a solution for problems with NB radix-4 Booth (NBBE-2) multipliers. On the other hand, this issue continues to have an impact on RB MBE multipliers.

B. RB Partial Product Generator

A RBPP is formed of two NB partial products since two bits are utilised to represent one RB digit. A two's complement representation may be used to indicate the integration of two N-bit NB partial products X and Y.

$$
\begin{aligned}
X + Y &= X - \overline{Y} - 1 \\
&= \left(-x_N 2^N + \sum_{i=0}^{N-1} x_i 2^i\right) - \left(-\overline{y_N} 2^N + \sum_{i=0}^{N-1} \overline{y_i} 2^i\right) - 1 \\
&= -(x_N - \overline{y_N}) 2^N + \sum_{i=0}^{N-1} (x_i - \overline{y_i}) 2^i - 1 \\
&= (X, \overline{Y}) - 1,
\end{aligned}
$$

(1)

There is a convention throughout the remainder of the paper where Y inverses Y. As an RB number, the composite number may be deduced. When one of two NB partial products is inverted, and the LSB is multiplied by 1, the RBPP is formed. Each RB digit Xi is encoded by 2 bits as the pair '1; 0; 1'.

TABLE 2
RB Encoding Used in This Work [6]

X_i^+	X_i^-	RB digit (X_i)
0	0	0
0	1	$\overline{1}$
1	0	1
1	1	0

Table 1: RB Encoding

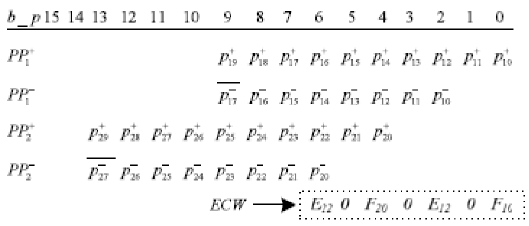

b_p 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
PP_1^+
PP_1^-
PP_2^+
PP_2^-
ECW
E_{12} 0 F_{20} 0 E_{12} 0 F_{16}

Fig.4.1: Conventional RBPP architecture for an 8-bit MBE multiplier

both negative and positive Xi effects. There are a variety of methods that may be used in order to encode an RB number. Table 2 presents a particular case of RB encoding; in this particular scenario, the RB digit is generated by using the Xi+ - Xi- procedure. The MBE and RB coding systems need the following terms to be changed as appropriate: When the modulated signal is multiplied by -1 or -2 during Booth encoding, the LSB of the second input must always be inverted and +1 must be added to it; consequently, -1 must be added to the LSB of the second input. This is because the LSB of the second input must always be inverted and +1 must be added to it. It's possible that a single ECW might handle both the radix-4 Booth recoding and the RB encoding. Figure 1 presents a common example of a partial product design for an 8-bit MBE multiplier that makes use of an encoder and a decoder. The produced bit position is denoted by the notation b p in this figure. A CRBBE-2 multiplier with N bits has four rows, with N equal to the number of RBPP rows plus one ECW row.

5. COMPARISION OF PARAMETERS

In the past, we have had an in-depth conversation on the modified Booth's algorithm as well as the Booth's multiplication algorithm. They were already working on their schematic implementation when they wrote down the techniques for estimating hardware parameters and modifying the bit width of input operands. This was done when they were working on their implementation. These parameters are compared by simulation in a virtex 4 xilinx area:

while conducting multiplication, the system output is optimized for area and has a maximum latency. These results are based on parameter comparisons and timing simulation results for the provided booth's multiplication. In order to compare the multipliers that are being offered here, it is necessary to compare the multipliers' parameters.

Table 5: Comparison of parameters

Booth Multiplier	Number of LUT's	Delay(ns)	Power(W)
Existing system 8-bit	144	23.333	0.081
Proposed system16-bit	297	5.116	0.065

Because of the new booth multiplier's slower working speed, it seems that the time delay will be reduced in this particular situation. However, the results that are produced by a 16-bit modified booth's multiplier are equivalent in area to the results that are produced by the booth's multiplier in terms of the complexity of the hardware and software.

6. SIMULATION AND SYNTHESIS REPORT

A. 8-Bit Unsigned Booth Multiplier

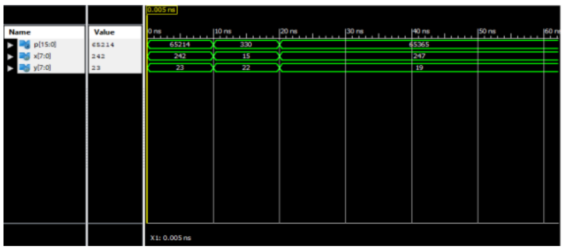


Fig.6.1: Simulation results of 8-bit unsigned Booth Multiplier

B.-Bit Signed Booth Multiplier

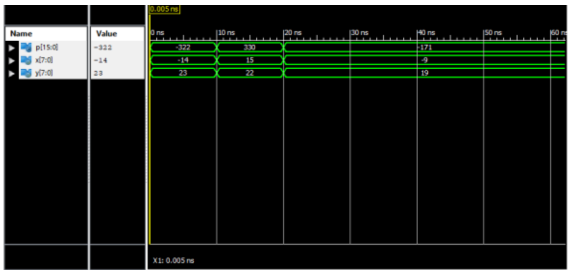


Fig.6.2: Simulation result of 8-bit signed Booth Multiplier

C. 16-Bit Unsigned Modified Booth Multiplier

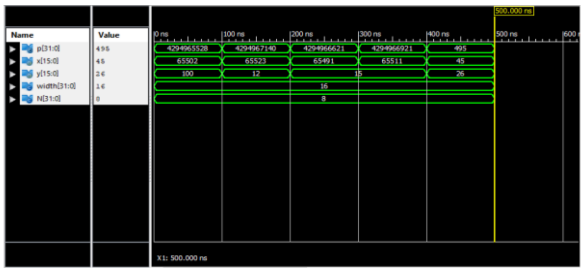


Fig.6.3: Simulation result of 16-bit unsigned Modified BM

D.16-Bit Signed Modified Booth Multiplier

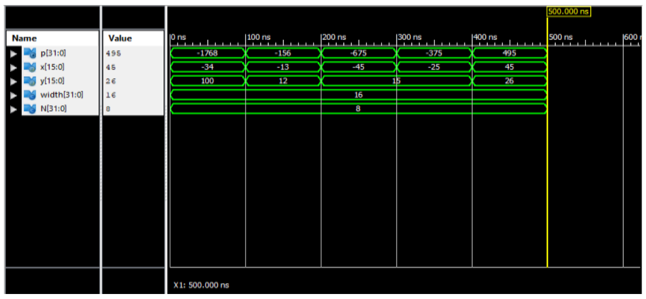
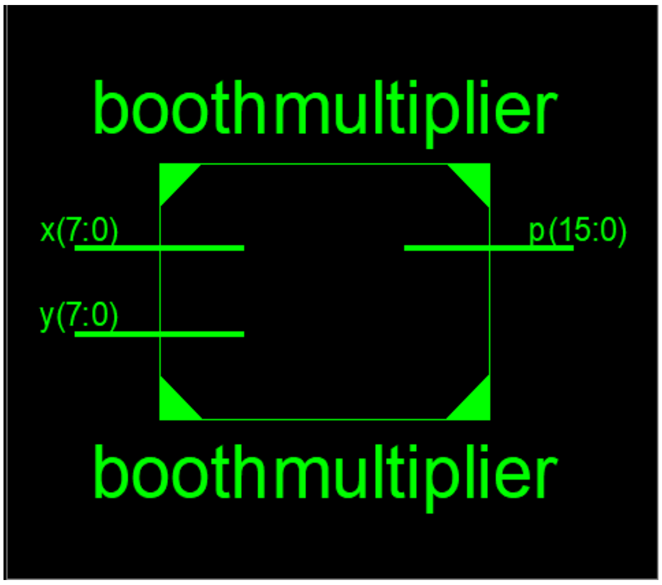


Fig.6.4: Simulation result of 16-bit signed Modified BM

E. RTL and Technology Schematics

8-Bit Booth Multiplier



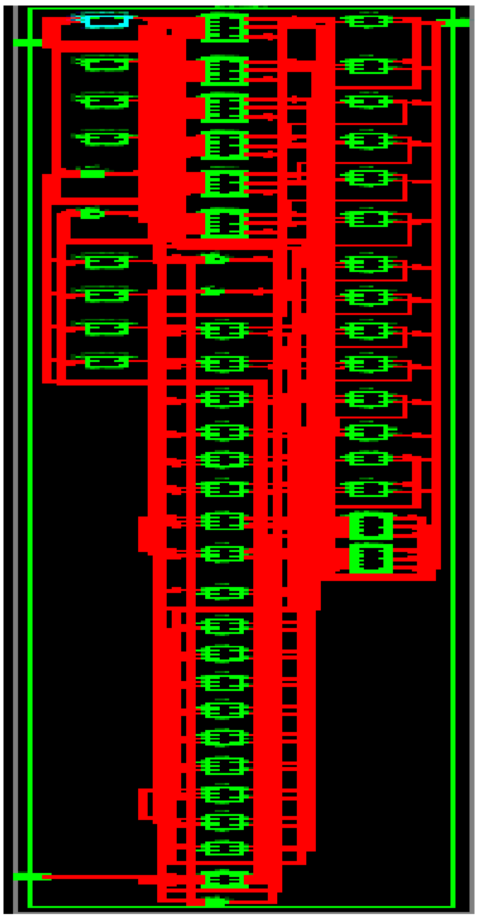
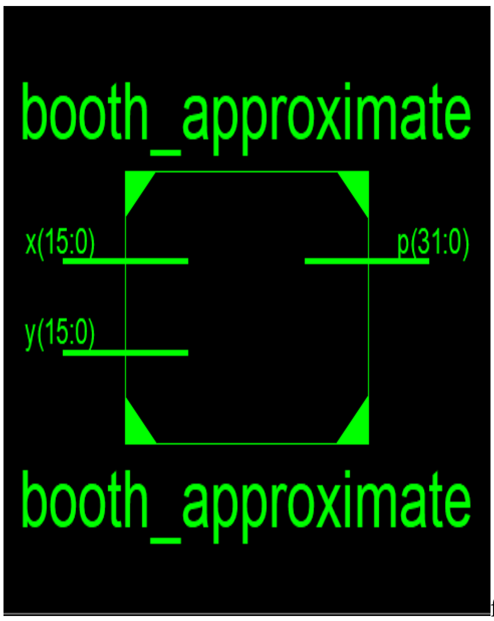


Fig.6.5: 8-bit RTL view of Booth Multiplier

F. 16-Bit Modified booth multiplier



f

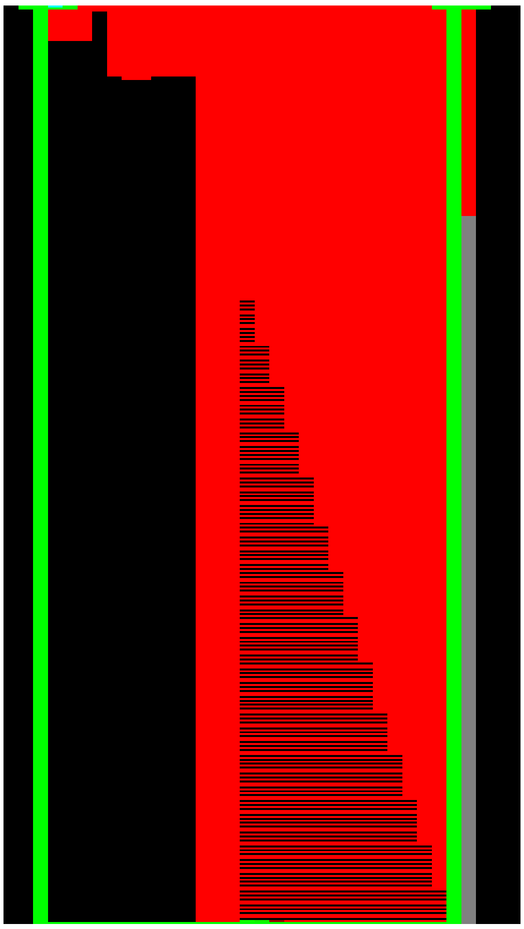
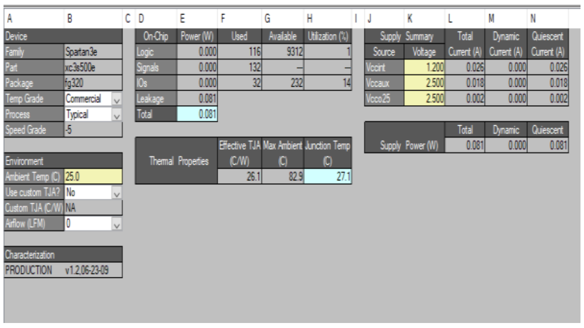


Fig.6.6: 16-bit RTL view of Modified Booth Multiplier



Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-5

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	116	9312	1
Signals	0.000	132	---	---
IOs	0.000	32	232	14
Leakage	0.081			
Total	0.081			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.026	0.000	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

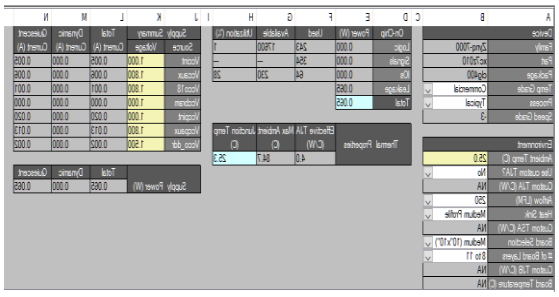
	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	82.9	27.1

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

Fig.6.7: 8-bit Booth Multiplier Power Result



Device	
Family	Zynq-7000
Part	xc7z010
Package	clg400
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	243	17600	1
Signals	0.000	364	---	---
IOs	0.000	64	230	28
Leakage	0.065			
Total	0.065			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.005	0.000	0.005
Vccaux	1.800	0.006	0.000	0.006
Vcco18	1.800	0.001	0.000	0.001
Vccbram	1.000	0.000	0.000	0.000
Vccpint	1.000	0.020	0.000	0.020
Vccpaux	1.800	0.013	0.000	0.013
Vcco_ddr	1.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.065	0.000	0.065

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.0	84.7	25.3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Fig.6.8: 16-bit Modified Booth Multiplier Power Result

CONCLUSION

The technique that has been presented produces power-delay products that are of a higher quality than those that are manufactured utilizing Booth multipliers. With the help of radix-4 Booth multipliers, we have seen how to bring the maximum height of the bits array down by one in this article. Because of this reduction, the pipelined multiplier's reduction tree may have access to an increased number of design alternatives in the case that a cell-based design is used. This method is one that may be used by conventional processors, digital signal chipmakers, mobile application processing units, and other arithmetic units. These are all examples of units that use Booth encoded data.

The incorporation of a comprehensive paradigm for approximate accumulator arithmetic computation makes it easier to develop approximation Booth multipliers and squares. In order to solve two key design challenges in the ECU (Electronic Control Unit) design, which is essential to the design of a AAAC (All Aluminum Alloy Conductor), it is necessary to have four theorems. These challenges are finding the best post-failure values and determining the best error compensation scheme. There is no need to be concerned about the ECU logic simplification that has been included since it will further lower the amount of energy used and take up less space.

REFERENCES

[1] 7 Series DSP48E1 Slice, document UG479, Xilinx, San Jose, CA, 321USA, 2018. 322

[2] S. Ullah et al., “Area-optimized low-latency approximate multipliers for 323FPGA-based hardware accelerators,” in Proc. ACM/ESDA/IEEE Design 324Autom. Conf. (DAC), 2018, pp. 1–6. 325

[3] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” 326IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 26, no. 2, 327pp. 203–215, Feb. 2007. 328

[4] LogiCORE IP Multiplier v12.0, document PG108, Xilinx, San Jose, 329CA, USA, 2015. 330

[5] A. D. Booth, “A signed binary multiplication technique,” Quart. J. 331Mech. Appl. Math., vol. 4, no. 2, pp. 236–240, 1951. 332

[6] C. R. Baugh and B. A. Wooley, “A two’s complement parallel array 333multiplication algorithm,” IEEE Trans. Comput., vol. C-22, no. 12, 334pp. 1045–1047, Dec. 1973. 335

[7] M. Kumm, S. Abbas, and P. Zipf, “An efficient softcore multiplier 336architecture for Xilinx FPGAs,” in Proc. IEEE Symp. Comput. Arithmetic 337(ARITH), 2015, pp. 18–25. 338

[8] E. G. Walters, “Array multipliers for high throughput in Xilinx FPGAs 339with 6-input LUTs,” Computers, vol. 5, no. 4, p. 20, 2016. 340

[9] H. Parandeh-Afshar and P. Ienne, “Measuring and reducing the 341performance gap between embedded and soft multipliers on FPGAs,” in 342Proc. Int. Conf. Field Program. Logic Appl. (FPL), 2011, pp. 225–231. 343

[10] 7 Series FPGAs Configurable Logic Block, document UG474, Xilinx, 344San Jose, CA, USA, 2016. 345

[11] H. Parandeh-Afshar, P. Brisk, and P. Ienne, “Exploiting fast carry-chains 346of FPGAs for designing compressor trees,” in Proc. Int. Conf. Field 347Program. Logic Appl. (FPL), 2009, pp. 242–249. 348