ISSN: 2278-4632
Vol-13, Issue-11, November 2023

ENERGY EFFICIENT LOW LATENCY SIGNEDMULTIPLIERFOR FPGA BASED HARDWARE ACCELERATORS

[1] N.VENKATESWARA RAO, B. Tech, M. Tech, (Ph.D.), SAI KRISHNA VADIYALA [2], SHIVA KRISHNA SAGAR MARKU [3], SIDDHU GUGULOTH [4].

[1]ASSOCIATE PROFESSOR OF ECE IN MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE, MAISAMMAGUDA, MEDCHAL (M), HYDERABAD-500100, T. S.

[2, 3, 4, 5.] FINALYEAR STUDENTS FROM DEPT OF ECE IN MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE, MAISAMMAGUDA, MEDCHAL (M), HYDERABAD-500100, T. S.

Abstract:

In this project, a novel architecture for Booth multiplier is implemented. By using 6-input LUTs and associated fast carry chains of modern FPGAs, we present an architecture for signed multipliers that provides better performance than state-of-the-art designs. The proposed partial product encoding technique reduces the length of the carry chain in each partial product to further reduce the critical path of the multiplier. FPGA-based designs that exist are largely limited to unsigned numbers, which require extra circuits to support signed operations. To overcome these limitations for the FPGA-based implementations of applications utilizing signed numbers, this project presents an area-optimized, low-latency and energy-efficient architecture for accurate signed multiplier. This design is simulated and synthesized using Xilinx ISE 14.7.

Keywords: Energy-Efficient applications, Signed multipliers, FPGA-based designs.

INTRODUCTION

A PPLICATIONS in the domain of digital signal processing and machine learning extensively use multiplication as one of the basic arithmetic operations. The architecture of a selected multiplier and its implementation directly affect the overall performance, resource utilization and energy consumption of such applications. The FPGA synthesis tools tend to use DSP blocks for high-performance multiplication [1]. However, two points are worth noting concerning the DSP blocks utilization. 1) For many applications, such as artificial neural networks (ANNs), the 32-bit floating-point precision is often not necessary for obtaining acceptable quality results. As discussed in Section III, our 8-bit quantized implementation of an ANN reduces the classification accuracy only by 0.42% when compared with full-precision classification accuracy. For

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

implementing multipliers for these low precision numbers, the synthesis tools opt to use lookup tables (LUTs) instead of DSP blocks. 2) As noted by the authors in [2] and [3], due to the non-uniform distribution of these DSP blocks across the FPGA, the critical path delay could be adversely affected when many of them have to be concatenated for large multiplication operations. Moreover, the DSP resources are limited. On the other hand, the LUT resources are much larger. They also offer comparable performance with better energy-efficiency and flexibility than the DSP blocks for smallsized multipliers. Therefore, it is more advantageous to have the option to use the low-area, high performance and energyefficient LUT-based multiplier beside the DSP blocks. In this paper, we provide area-optimized, low latency, and energyefficient accurate, signed multipliers for FPGA-based systems. FPGA vendors, such as Xilinx and Intel, provide softcore LUT-based multipliers (signed and unsigned) as described in [4]. These multipliers can be either area- or speedoptimized. Booth's algorithm [5] is also a commonly used technique for multiplication because it reduces the total number of generated partial products by encoding the multiplier bits.

Booth's algorithm to present area-efficient radix-4 multiplier implementations for Xilinx FPGAs. However, these implementations do not use compressor trees for adding the generated partial products and have large critical path delays. More importantly, [7] has not discussed the implementations for signed numbers. Parandeh-Afshar et al. [11] have proposed a partial products compressor tree for Altera (now Intel) FPGAs. Nonetheless, their generalized parallel counters underutilize LUTs in two consecutive adaptive logic modules (ALMs). Their followup work, [9], has used the Booth's and Baugh-Wooley's multiplication [6] algorithms for area-efficient multiplier implementation. However, in order to reduce the effective length of the carry chains, their design limits the length of the ALM to five, resulting in the underutilization of the FPGA resources. On the other hand, the authors of [12] and [13] utilize smaller multiplier blocks for designing higher order multipliers. However, such techniques prove to be only useful for small bit-width multipliers; for higher bit-width multipliers, they consume more FPGA resources. For example, the logic-based implementation (using the '*' operation) of an accurate 8×8 multiplier on Virtex-7 FPGA in Xilinx Vivado, with default synthesis options, consumes 71 LUTs. Whereas, the modular implementation of an accurate 8×8 multiplier using accurate 4×4 multipliers consumes 82 LUTs.

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

EXISTING SYSTEM:

The proposed architecture computes all partial products in parallel and then adds the generated partial products using multiple 4:2 compressors and a ripple carry adder (RCA). The parallel generation of partial products significantly reduces the critical path delay of the multiplier. For an N X M multiplier, the length of the carry chain in each partial product row is N+4 bits. To improve the critical path delay of the multiplier can be modified with booth algorithm with radix length of the carry chain can be reduced to N+1 bits. A critical path delay optimized implementation of our novel multiplier is shown in below figure. The partial product terms pp(x, 0) and pp(x, 1), in each partial product row, require one and two bits of multiplicand respectively. These two partial product terms can be implemented by one single 6-input LUT. Similarly, pp(x, 2), in each partial product row, can be independently implemented using another 6-input LUT. A separate 6-input LUT, 'CG', can be used to compute the correct input carry for each partial product row. Thus, our proposed multiplier provides higher performance than state-of-the-art accurate and approximate multipliers.

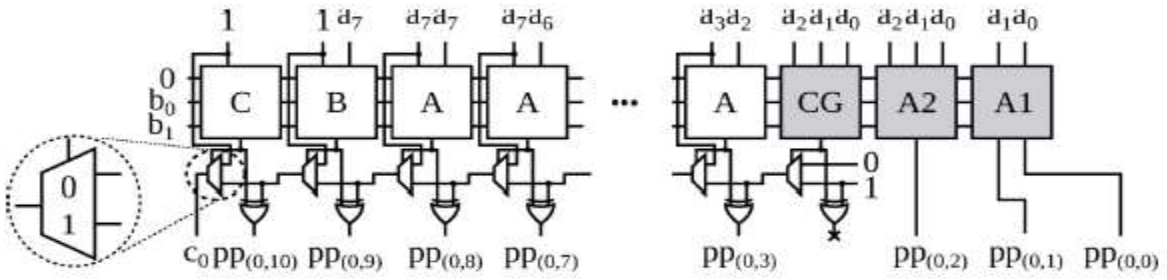


Fig: Architecture of Optimized version of multiplier

Advantages of Project:

- The proposed design requires less area.
- Less Critical Path delay
- Low Latency

Applications:

- Artificial Neural Networks
- Image processing

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

PROPOSED METHOD

Complex arithmetic operations are widely used in Digital Signal Processing (DSP) applications. In this work, we focus on optimizing the design of the fused Add-Multiply (FAM) operator for increasing performance. We investigate techniques to implement the direct recoding of the sum of two numbers in its Modified Booth (MB) form. We introduce a structured and efficient recoding technique and explore three different schemes by incorporating them in FAM designs. Comparing them with the FAM designs which use existing recoding schemes, the proposed technique yields considerable reductions in terms of critical delay, hardware complexity and power consumption of the FAM unit.

Radix-4 Booth Encoding

Booth encoding has been proposed to facilitate the multiplication of two's complement binary numbers [17]. It was revised as modified Booth encoding or radix-4 Booth encoding [18]. The MBE scheme is summarized in Table 1. The multiplier bits are grouped in sets of three adjacent bits. The two side bits are overlapped with neighbouring groups except the first multiplier bits group in which it is {b1, b0, 0}. Each group is decoded by selecting the partial product shown in Table 1, where 2A indicates twice the multiplicand, which can be obtained by left shifting. Negation operation is achieved by inverting each bit of A and adding '1' (defined as correction bit) to the LSB [10], [11], [12], [13]. Methods have been proposed to solve the problem of correction bits for NB radix-4 Booth encoding (NBBE-2) multipliers. However, this problem has not been solved for RB MBE multipliers.

RB Partial Product Generator

As two bits are used to represent one RB digit, then a RBPP is generated from two NB partial products [1], [2], [3], [4], [5], [6]. The addition of two N-bit NB partial products X and Y using two's complement representation can be expressed as follows [6]:

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

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

where Y is the inverse of Y , and the same convention is used in the rest of the paper. The composite number can be interpreted as a RB number. The RBPP is generated by inverting one of the two NB partial products and adding -1 to the LSB. Each RB digit Xi belongs to the set {1; 0; 1}; this is coded by two bits as the pair

TABLE 2
RB Encoding Used in This Work [6]

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

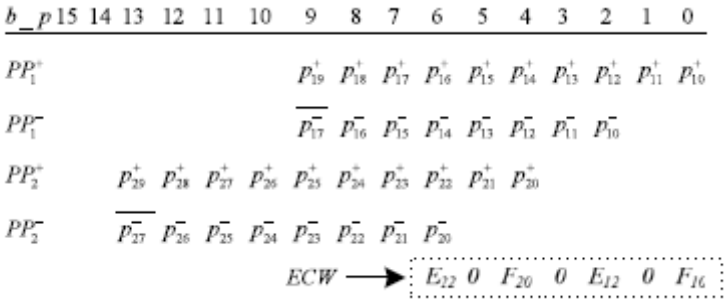


Fig. Conventional RBPP architecture for an 8-bit MBE multiplier

(X_i^- , X_i^+). RB numbers can be coded in several ways. Table 2 shows one specific RB encoding [6], where the RB digit is obtained by performing $X_i^+ - X_i^-$ Both MBE and RB coding schemes introduce errors and two correction terms are required: 1) when the NB number is converted to a RB format, -1 must be added to the LSB of the RB number; 2) when the multiplicand is multiplied by -1 or -2 during the Booth encoding, the number is inverted and þ1 must be added to the LSB of the partial product. A single ECW can compensate errors from both the RB encoding and the radix-4 Booth recoding.

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

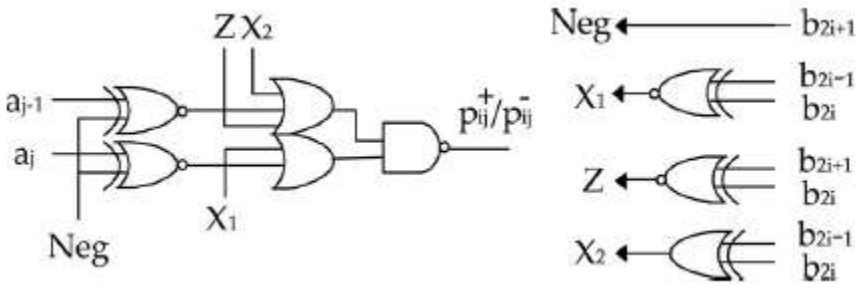
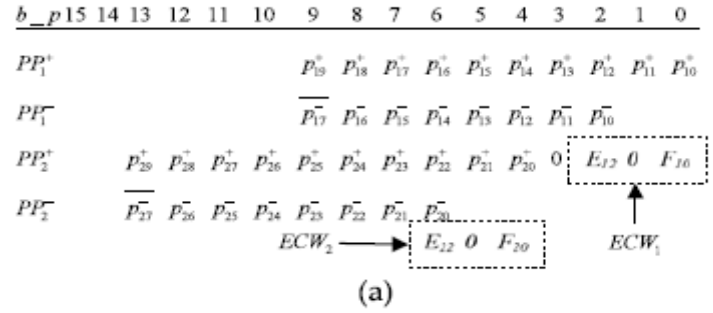
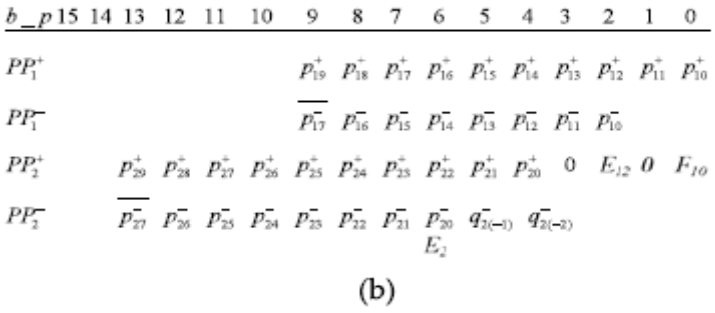


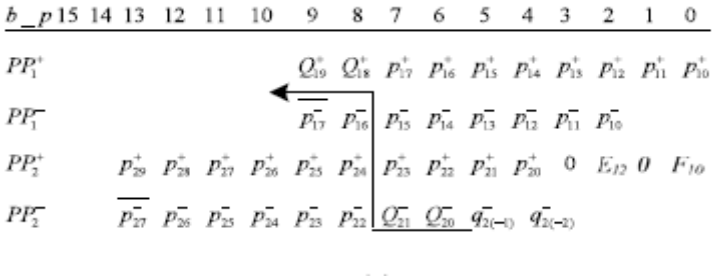
Fig. An encoder and decoder of the MBE scheme.



(a)



(b)



(C)

Fig. (a) The first new RBMPPG-2 architecture for an 8-bit MBE multiplier; (b) the further revised RBMPPG-2 architecture by replacing E_{22} and F_{20} with E_2, $q_{2(-2)}$,and $q_{2(-1)}$; (c) the final proposed RBMPPG-2 architecture by totally eliminatingECW2 an further combing E2 into Q^{+}_{19}, Q^{+}_{18}, Q^{-}_{21}, and Q^{-}_{20}.

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

For a 2n-bit CRBBE-2 multiplier, one additional RBPP accumulation stage is required due to the ECW. For a 64-bit RB multiplier, there are five RBPP accumulation stages; therefore, the number of RBPP accumulation stages can be reduced by 20 percent when eliminating the ECW in RB multiplier, which improves both the complexity and the critical path delay.

The circuit diagrams of the modified partial product variables Q^{+}_{19}, Q^{+}_{18}, $Q^{-}_{21}$are shown in Fig. It is clear that Q^{+}_{18} has the longest delay path. It is well known that the inverter, the 2-input NAND gate and the transmission gate (TG) are faster than other gates. So, it is desirable to use TGs when designing the multiplexer . As shown in Fig., the critical path delay (the dash line) consists of a 1-stage AND-OR-Inverter gate, a one-stage inverter, and two-stage TGs. Therefore, RBMPPG-2 just increases the TG delay by one-stage compared with the MBE partial product of Fig. . The above discussion is only an example; the above technique can be applied to design any 2n-bit RB multipliers. It eliminates the extra ECWN/4 and saves one RBPP accumulation stage, i.e., three XOR gate delays, while only slightly increasing the delay of the partial product generation stage. In general, an N-bit RB multiplier has N=4 RBPP rows using the proposed RBMPPG-2. The partial product variables $p^{+}_{1(N+1)}$, p^{+}_{1N}, $p^{-}_{(N/4)1}$ and $p^{-}_{(N/4)0}$ can be replaced by $Q^{+}_{1(N+1)}$, Q^{+}_{1N}, $Q^{-}_{(N/4)1}$ and $Q^{-}_{(N/4)0}$. The radix-4 Booth decoding of a PPR needs additional three-input OR gates (Fig. 4). Therefore, the extra ECWN/4 is removed by the transformation of four partial product variables $Q^{+}_{1(N+1)}$, Q^{+}_{1N}, $Q^{-}_{(N/4)1}$ and $Q^{-}_{(N/4)0}$ and one partial product row is saved in RB multipliers with any power-of two word-length.

The proposed RBMPPG-2 can be applied to any 2n-bit RB multipliers with a reduction of a RBPP accumulation stage compared with conventional designs. Although the delay of RMPPG-2 increases by one-stage of TG delay, the delay of one RBPP accumulation stage is significantly larger than a one-stage TG delay. Therefore, the delay of the entire multiplier is reduced. The improved complexity, delay and power consumption are very attractive for the proposed design.

A 32-bit RB MBE multiplier using the proposed RBPP generator is shown in Fig. 6. The multiplier consists of the proposed RBMPPG-2, three RBPP accumulation stages, and one RB-NB converter. Eight RBBE-2 blocks generate the RBPP; they are summed up by the RBPP reduction tree that has three RBPP accumulation stages. Each RBPP accumulation block contains RB full adders (RBFAs) and half adders (RBHAs). The 64-bit RB-NB converter converts the final

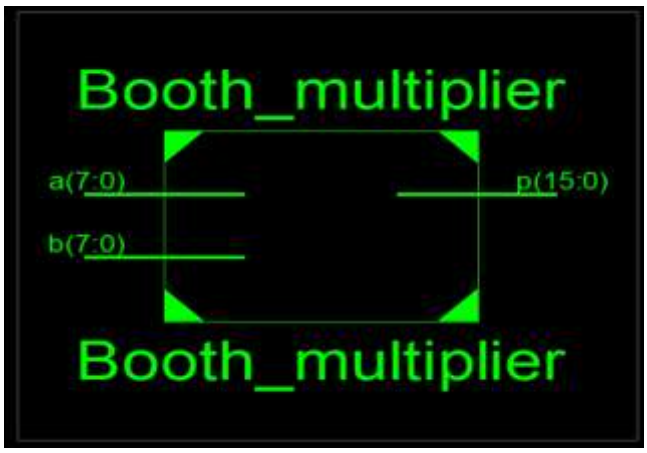
ISSN: 2278-4632
Vol-13, Issue-11, November 2023

accumulation results into the NB representation, which uses a hybrid parallel-prefix/carry select adder (as one of the most efficient fast parallel adder designs).

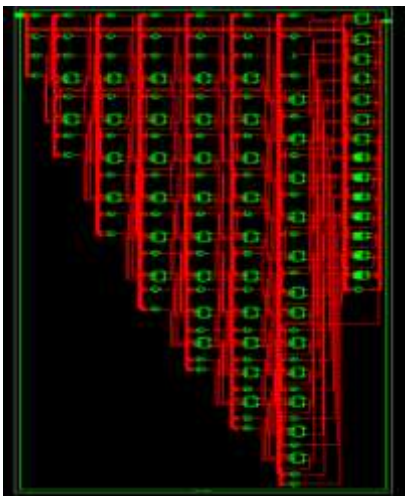
There are four stages in a conventional 32-bit RB MBE multiplier architecture; however, by using the proposed RBMPPG-2, the number of RBPP accumulation stages is reduced from 4 to 3 (i.e., a 25 percent reduction). These are significant savings in delay, area as well as power consumption. The improvements in delay, area and power consumption are further demonstrated in the next section by simulation.

SIMULATION RESULTS

RTL

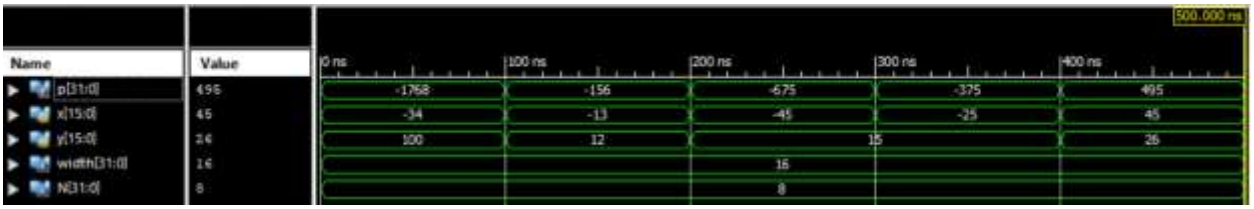


INTERNAL BLOCK DIAGRAM



ISSN: 2278-4632
Vol-13, Issue-11, November 2023

SIMULATION RESULTS



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

It can be seen that the comparison of the booth's multiplier in terms of operating speed indicates that modified booth's multiplier gives reduction in time delay. But in terms of hardware complexity, 16-bit modified booth's multiplier gives similar results in area by almost compared to the booth's multiplier.

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

CONCLUSION AND FUTURE SCOPE

The multiplier using the proposed algorithm achieves better power-delay products than those achieved by conventional Booth multipliers. Here, we have presented a method to reduce by one the maximum height of the partial product array with radix-4 Booth recoded magnitude multipliers. This reduction may allow more flexibility in the design of the reduction tree of the pipelined multiplier and achieved with no extra delay for $n \geq 32$ for a cell-based design. We believe that the proposed Booth algorithm can be broadly utilized in general processors as well as digital signal processors, mobile application processors, and various arithmetic units that use Booth encoding.

A general model is presented for array-based approximate arithmetic computing to guide the design of approximate Booth multipliers and squarers. To shed light on the design of ECU, which is the key of AAAC design, we develop four theorems to address two critical design problems of the ECU design, namely, determination of optimal error compensation values and identification of the optimal error compensation scheme. To further reduce energy consumption and area, we introduce don’t cares for ECU logic simplification.

ISSN: 2278-4632
Vol-13, Issue-11, November 2023

REFERENCES

[1] Xilinx. 2018. 7 Series DSP48E1 Slice, UG479.

[2] S. Ullah, et al., "Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators," in DAC 2018.

[3] I. Kuon et al., "Measuring the gap between FPGAs and ASICs," in IEEE TCADICS 2007.

[4] Xilinx. 2015. LogiCORE IP Multiplier v12.0, PG108.

[5] A. D. Booth, "A Signed Binary Multiplication Technique," in the Quarterly Journal of Mechanics and Applied Mathematics 1951.

[6] C. R. Baugh et al., "A two's complement parallel array multiplication algorithm," in IEEE TC, vol. 100, no. 12, 1973.

[7] M. Kumm, et al., "An efficient softcore multiplier architecture for Xilinx FPGAs," in Computer Arithmetic (ARITH), 2015.

[8] E. G. Walters, "Array Multipliers for High Throughput in Xilinx FPGAs with 6-Input LUTs," in Computers, MDPI, 2016.

[9] H. Parandeh-Afshar et al., "Measuring and reducing the performance gap between embedded and soft multipliers on FPGAs," in FPL, 2011.

[10] Xilinx. 2016. 7 Series FPGAs Configurable Logic Block, UG474.