ESTIMATION OF CARRY BITS IN FW BOOTH MULTIPLIER USING TRUNCATION ERROR TECHNIQUE

D. UMA MAHESWARI [1], RAMISETTI CHENNAKESAVA RAO [2], NELLURI SAILAJA[3], VANAPARLA AMULYA [4], MEKALA PRAKASH [5],PILLI VASANTH [6].
[1] ASSISTANT PROFESSOR,[2,3,4,5,6.] STUDENTSFrom Dept. of Electronics and Communication Engineering, Sree Vahini institute of Science and Technology, Tiruvuru, AP, 521235.

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

EXISTING SYSTEM:

The proposed architecture computes all partial products in parallel and then adds the generated partial products using multiple 4:2 compressorsand 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 chainin 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 carrychain can be reduced to N+1 bits. A critical path delay optimized implementation of our novel multiplier is shownin 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 proposedmultiplier provides higher performance than state-of-the-art accurate and approximatemultipliers.
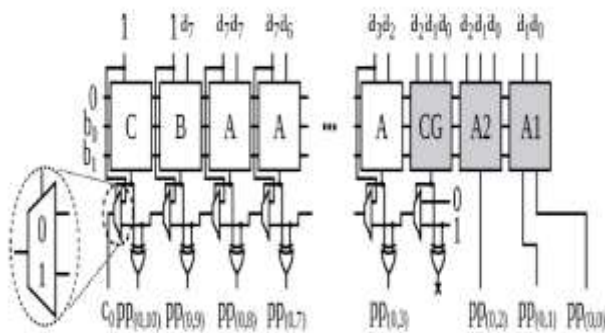
Architecture of Optimized version of multiplier

Advantages of Project:

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

Applications:

- Artificial Neural Networks
- Image processing

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 multiplicationof two's complement binary numbers [17]. It was revised as modifiedBooth encoding or radix-4 Booth encoding [18]. The MBEscheme is summarized in Table 1. The multiplier bits are grouped in sets of three adjacentbits. 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 shownin Table 1, where 2A indicates twice the multiplicand, which canbe obtained by left shifting. Negation operation is achieved byinverting each bit of A and adding '1' (defined as correction bit) tothe LSB [10], [11], [12], [13]. Methods have been proposed to solvethe problem of correction bits for NB radix-4 Booth encoding(NBBE-2) multipliers. However, this problem has not been solvedfor RB MBE multipliers.

RB Partial Product Generator

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

$$X + Y = X - \overline{Y} - 1$$

$$= \left(-x_N 2^N + \sum_{i=0}^{N-1} x_i 2^i\right) - \left(-\overline{y_N} 2^N + \sum_{i=0}^{N-1} \overline{y_i} 2^i\right) - 1$$

$$= -\left(x_N - \overline{y_N}\right)2^N + \sum_{i=0}^{N-1} \left(x_i - \overline{y_i}\right)2^i - 1$$

$$= \left(X, \overline{Y}\right) - 1,$$

(2) where Y is the inverse of Y , and the same convention is used in therest of the paper. The composite numbercan be interpretedas a RB number. The RBPP is generated by inverting one of the twoNB partial products and adding -1 to the LSB. Each RB digit Xibelongs to the set {1; 0; 1}; this is coded by two bits as the pair

TABLE 2
RB Encoding Used in This Work [6]

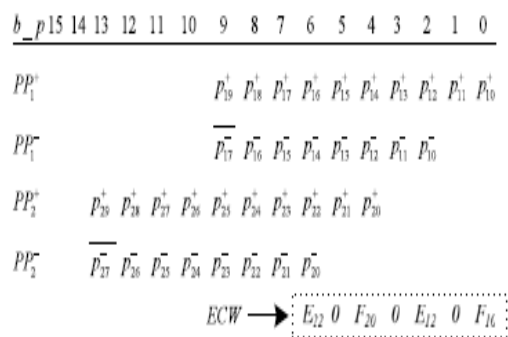| X_i^+	X_i^-	RB digit (X_i)
0	0	0
0	1	$\bar{1}$
1	0	1
1	1	0

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

(X_i^-, X_i^+). RB numbers can be coded in severalways. Table 2 shows one specific RB encoding [6], where the RBdigit is obtained by performing $X_i^+ - X_i^-$ Both MBE and RB coding schemes introduce errors and twocorrection terms are required: 1) when the NB number is convertedto a RB format, -1 must be added to the LSB of the RBnumber; 2) when the multiplicand is multiplied by -1 or -2during the Booth encoding, the number is inverted and þ1must be added to the LSB of the partial product. A single ECWcan compensate errors from both the RB encoding and theradix-4 Booth recoding.
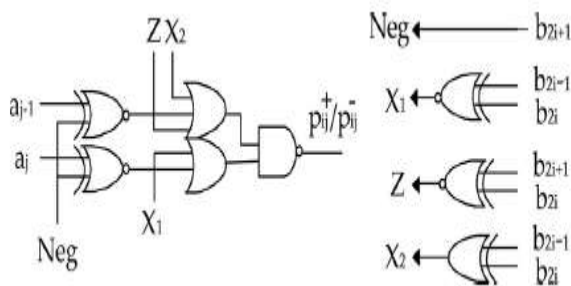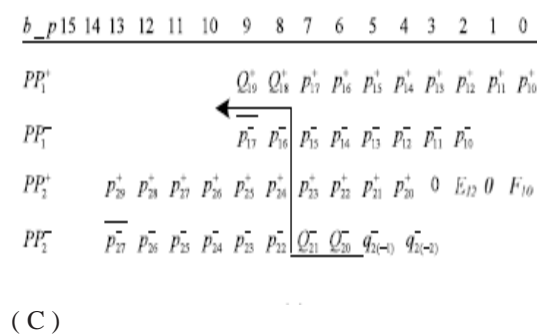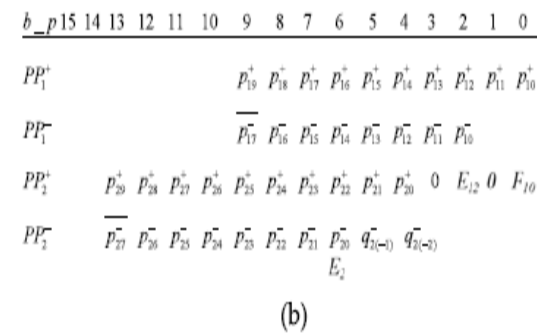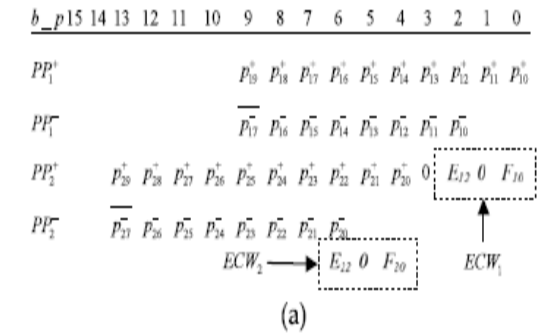

Fig. An encoder and decoder of the MBE scheme.

(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 eliminating ECW2 an further combing E2 into Q^{+}_{19}, Q^{+}_{18}, Q^{-}_{21}, and Q^{-}_{20}.

For a 2n-bit CRBBE-2 multiplier, one additional RBPP accumulationstage is required due to the ECW. For a 64-bit RB multiplier,there are five RBPP accumulation stages; therefore, the number ofRBPP accumulation stages can be reduced by 20 percent wheneliminating the ECW in RB multiplier, which improvesboth 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 longestdelay path. It is well known that the inverter, the 2-inputNAND 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 TGdelay by one-stage compared with the MBE partial product of Fig. .The above discussion is only an example; the above techniquecan be applied to design any 2n-bit RB multipliers. It eliminates theextra ECWN/4 and saves one RBPP accumulation stage, i.e., threeXOR gate delays, while only slightly increasing the delay of thepartial product generation stage. In general, an N-bit RB multiplierhas N=4 RBPP rows using the proposed RBMPPG-2. The partialproduct 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 aPPR needs additional three-input OR gates (Fig. 4). Therefore,the extra ECWN/4 is removed by the transformation of fourpartial product variables

$Q^{+}_{1(N+1)}$, Q^{+}_{1N}, $Q^{-}_{(N/4)1}$and $Q^{-}_{(N/4)0}$ and onepartial product row is saved in RB multipliers with any power-oftwoword-length.

Fig. The block diagram of a 32-bit RB multiplier using the proposed RBMPPG-2

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

A 32-bit RB MBE multiplier using the proposed RBPP generatoris shown in Fig. 6. The multiplier consists of the proposedRBMPPG-2, three RBPP accumulation stages, and one RB-NB converter.Eight RBBE-2 blocks generate the RBPP; they aresummed up by the RBPP reduction tree that has three RBPP accumulationstages. Each RBPP accumulation block contains RB fulladders (RBFAs) and half adders (RBHAs). The 64-bit RB-NBconverter converts the final 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 multiplierarchitecture; 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 indelay, area as well as power consumption. The improvementsin delay, area and power consumption are further demonstratedin the next section by simulation.
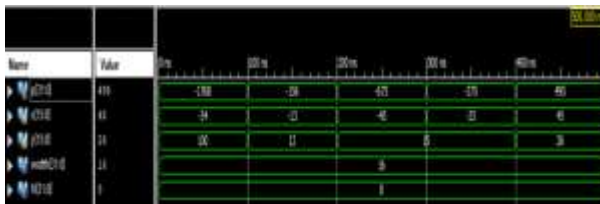
SIMULATION RESULTS

RTL
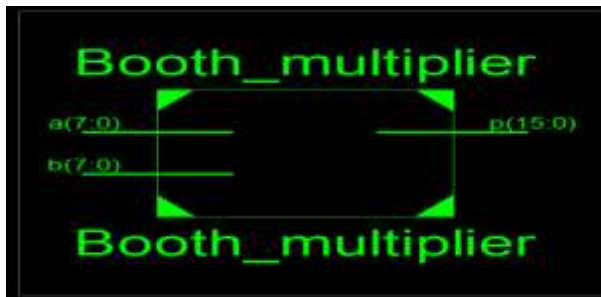
INTERNAL BLOCK DIAGRAM

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 system 16-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.

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.

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.