International Journal of Engineering Innovations in Advanced Technology
ISSN: 2582-1431 (Online), Volume-4 Issue-4, December 2022

Design Of High-Speed Area Efficient Mac Unit Using Reversible Logic

K Vamshinadh
MTech Student, Department Of ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., India.
Email: kvamshinadh94@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

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

Abstract—we propose a low-power high-speed pipeline multiply-accumulate (MAC) architecture. In a conventional MAC, carry propagations of additions (including additions in multiplications and additions in accumulations) often lead to large power consumption and large path delay. To resolve this problem, we integrate a part of additions into the partial product reduction (PPR) process. In the proposed MAC architecture, the addition and accumulation of higher significance bits are not performed until the PPR process of the next multiplication. To correctly deal with the overflow in the PPR process, a small-size adder is designed to accumulate the total number of carries. Compared with previous works, experimental results show that the proposed MAC architecture can greatly reduce both power consumption and circuit area under the same timing constraint.

***Index Terms*— MAC architecture, bits, adder, power consumption**

I. Introduction

The multiply-accumulate (MAC) unit is a fundamental block for digital signal processing (DSP) applications. Especially, in recent years, the development of real-time edge applications has become a design trend. Thus, there is a strong demand for high-speed low-power MAC units. A conventional MAC unit is composed of two individual blocks: a multiplier and an accumulator (i.e., an accumulate adder). An N-bit MAC unit includes an N-bit multiplier and a (2N+α-1)-bit accumulator (adder), where α is the number of guard bits used to avoid overflow (caused by long sequences of multiply-accumulate operations). A lot of previous works paid attention to the optimization of multiplier and the optimization of adder, respectively. A multiplier usually has three steps. The first step is the partial product generation (PPG) process. For example, AND gates can be used to generate a partial product matrix (PPM) for an unsigned multiplication. The second step is the partial product reduction (PPR) process. By using the Dadda tree approach or the Wallace tree approach, the PPM can be reduced to become two rows. The third step is the final addition. An adder (called the final adder) is used to perform the summation of the final two rows.

Manuscript received Oct 10, 2022; Revised Oct 25, 2022; Accepted Nov 4, 2022

A. Multiplier

Multiplication is a fundamental operation in most signal processing algorithms. Multipliers have large area, long latency and consume considerable power. Therefore low-power multiplier design has an important part in low-power VLSI system design. A system is generally determined by the performance of the multiplier because the multiplier is generally the slowest element and more area consuming in the system. Hence optimizing the speed and area of the multiplier is one of the major design issues. However, area and speed are usually conflicting constraints so that improvements in speed results in larger areas. Multiplication is a mathematical operation that include process of adding an integer to itself a specified number of times. A number (multiplicand) is added itself a number of times as specified by another number (multiplier) to form a result (product). Multipliers play an important role in today's digital signal processing and various other applications. Multiplier design should offer high speed, low power consumption. Multiplication involves mainly 3 steps

1. Partial product generation

2. Partial product reduction

3. Final addition

II. Existing Method

In this section, we present the proposed two-stage (i.e., two cycle) MAC architecture. The first stage performs the PPG process, the PPR process (based on the PPM that combines the PPG result and the accumulation result), the (2N-k-1)-bit addition (i.e., a part of the final addition) and the αbit addition (for dealing with the overflow in the PPR process). Then, the second stage performs the (k+α)-bit addition to produce the accumulation result. The main features of the proposed architecture are below. λ To reduce the lengths of carry propagations, we integrate a part of additions into the PPR process. λ To handle overflow in the PPR process, a α-bit adder is used to count the total number of carries. λ By applying the gating technique, the second stage can only be executed in the last cycle (of the entire sequence of multiply-accumulate operations) for power saving. The proposed two-stage pipeline MAC unit is displayed in Fig. 2. Our PPM (for the PPR process) is composed of two PPMs: one PPM is derived by the PPG and the other PPM is derived by the accumulation. For an unsigned MAC unit, in the PPG process, "AND" gates can be directly used to generate the PPM. For a signed MAC unit, because the influences of the sign bit should be taken into account, several PPG algorithms have been proposed to generate the signed PPM. In the proposed architecture, the Baugh-Woolley algorithm is adopted in the PPG process to generate the signed PPM.

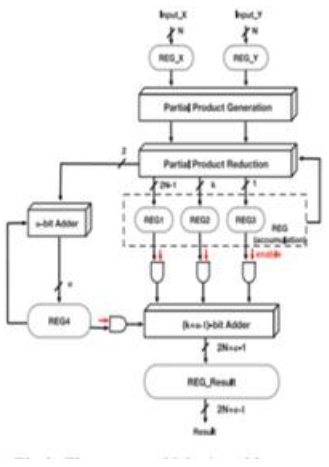

Figure 1.MAC architecture

We have implemented a tool (a C++ program) to automatically generate the proposed N-bit MAC in Verilog RTL description. The users can specify the value of N and the value of k for automatic generation, where k denotes the number of higher significance bits whose additions (accumulation) are not performed in the final addition. Note that the value of k is equal to the bit width of register REG2. In our experiments, we specify the value of N to be 16 (i.e., 16-bit MAC). Besides, we assume that the maximum number of multiplications in each multiply-accumulate operation is 256. Thus, the number of guard bits (i.e., the value of α) is set to be 8. We have implemented several different configurations of the proposed MAC architecture. For the convenience of presentation, we use the term Ours_k for the naming of each configuration, where k represents the bit width of register REG2. In our experiments, these Verilog RTL descriptions are synthesized to gate-level netlists and targeted to TSMC 40 nm cell library by using Synopsys Design Compiler.For comparisons, we also implemented the following two MAC architectures: the conventional MAC architecture and the state-of-the-art MAC architecture. In the conventional MAC architecture, the MAC unit is composed of two individual blocks (i.e., a multiplier and an accumulator). On the other hand, in the state-of-the-art MAC architecture, the multiplier and the accumulator are tightly integrated (i.e., a carry-save format is sent to the accumulator without being added to only one vector).

The systolic array has been widely used in the hardware acceleration for matrix multiplication. In recent years, several research efforts have been paid to map the inference of a convolutional neural network to a systolic array. Note that a systolic array is composed of multiple processing elements (PEs). Each PE corresponds to a MAC unit. In this section, we address the application of the proposed MAC architecture to a systolic array. Figure gives the block diagram of the PE based on the conventional MAC architecture. Note that the PE is a two-stage (i.e., two-cycle) pipeline design. The inputs of the PE are x and y. The block MUL denotes the multiplier. In the first stage, the multiplier performs the multiplication. Then, the output of the multiplier is stored in a register. In the second stage, the accumulator performs the accumulation. Then, the accumulation result is stored in register result.

III.PROPOSED METHOD

In data transmission applications, the widely used public-key cryptosystem is a simple and efficient Montgomery multiplication algorithm such that the low-cost and high-performance. In which includes encryption and decryption process. The Montgomery multiplier receives and outputs the data with binary representation and uses only one-level carry-save adder (CSA) to avoid the carry propagation at each addition operation. This CSA is also used to perform operand pre-computation and format conversion from the carry save format to the binary representation, leading to a low hardware cost and short critical path delay at the expense of extra clock cycles for completing one modular multiplication. To overcome the weakness, A configurable CSA (CCSA), which could be one full-adder or two serial half-adders, is proposed to reduce the extra clock cycles for operand pre-computation and format conversion by half. When modular multiplier is done with CCSA technique and it has some drawbacks. The drawbacks are short critical path, high power consumption. To overcome the drawbacks the CCSA is replaced with PASTA (Parallel Self Timed Adder) in the Montgomery modular multiplier. The PASTA adder can achieve less power consumption.

Modular Multiplication is the central operation in many application areas including public key cryptography for encryption and decryption. The widely used method for modular multiplication is Montgomery modular multiplier. In which there will be a carry save adder . X´Y mod M is the operation to be performed. In which X and Y are the inputs. It is necessary to find the value of mod M, henceforth going for this algorithm. Comparing all previously occurring algorithms, this algorithm will produce the optimized output. There are two cases, semi carry save addition and full carry save addition. In this semi carry save addition, the given inputs are in binary and the inter outputs alone in carry save. Whereas in full carry addition, both inputs and inter outputs are in carry save. On comparing, it can be seen that semi carry save is the most advantageous one because it has only one carry save and hence it has less area and high speed which is required for designing an VLSI based multipliers.

Consider the modulus N to be a k-bit odd number and an extra factor R is to be defined as 2k mod N, where 2k−1 ≤ N < 2k. Given two integers a and b, where a, b

$$A = a\times R \pmod N \qquad (1)$$
$$B = b\times R \pmod N \qquad (2)$$

In this existing system, carry save addition with semi-carry approach is described. In which all the multiplicands are not recycled, that is whatever the multiplicand is needed to be multiplied at that time alone is used for determining the output. The carry save approach has higher benefits since it is the basic key for operating a Montgomery modular multiplier. In such a way, using this semi carry save type only one carry level adder is implemented which may be two serial half adders or a full adder can be used based on the requirement. It thereby reduces the number of clock cycles and hence less delay. So the output will be optimized and it can be implemented using Verilog coding.

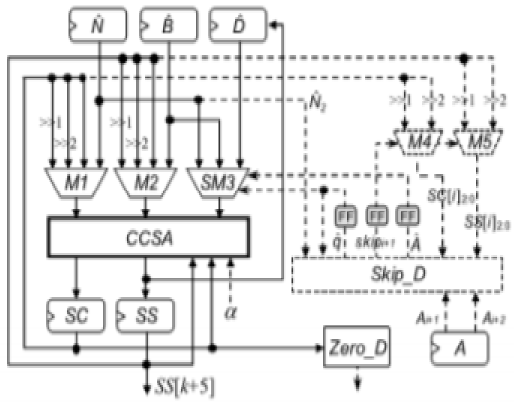

Figure 2. Block diagram of Montgomery Modular Multiplication using CCSA

The above architecture is the semi-carry save based Montgomery multiplier. In which the loop is reduced on comparing to the existing one. It consists of two multiplexers, one multiplier, one configurable carry save adder, flip-flops, skip detector and zero detector.

Illustrates the block diagram of proposed semi carry save multiplier. It is first used to precompute the four-to-two carry save additions. Then the required multiplication can be performed. The modulus N and inputs will be allowed inside the twomultiplexers. This partial product is then allowed inside the multiplier. Those partial outputs then enter into configurable carry save adder, where the carry save addition operation is performed. They are stored in the flip flops temporarily. When another partial output is executed, then that will be stored in the flip flop. The Skip detector will skip the previous multiplication which is not required in the operation so as to reduce the number of clock cycles. The partial product from SM3 is allowed to the multiplexers M4 and M5. Later on it allows inside the flip flops for temporary storage, then to the skip detector. The output can be obtained from semi carry. This process is repeated until the output is obtained. The zero detectors can also be used to detect zero in many situations, which is most required. The complexity is very less compared to the previous one.

Critical Path Delay Reduction In order to reduce the critical path delay, the operations in semi carry save and full carry save is performed jointly. The carry save format conversion as well as the binary format should be taken place. Then pre-computation must be done in order to reuse the multiplicand values. Another method is using zero detectors. SC will produce the output only when the zero is detected. Then the pre computation can be done i-1 iteration.

Clock Cycle Number Reduction In order to decrease the clock cycle number, a configurable carry save architecture to perform one three-input carry-save addition or two serial two-input carry-save addition is used. Furthermore the number of iteration can also be reduced to reduce number of clock cycles. Then a signal skip is used. In order to verify whether i+1 is required or not to be happen in the upcoming events. This can be found in the previous i iteration itself. By again using the same condition, signal skip will use i+1, so that it increases by a factor 2. Hence it directly goes to i+2. So that clock cycle gets reduced.

Quotient Pre-computation: Ai+1, Ai+2 and qi+1, qi+2 should be known already in order that the unwanted steps in the (i +1) iteration can be reduced by determining i iteration. So as to pre compute the quotients. Another method is using skip detector so that it will pre computes the values. And also since the shortest path in this multiplier is lengthened, it has to be minimized. As modulus N is an odd number, it can be used directly for the multiplication. So that time is consumed highly.

To increase the Speed of Operation we are replacing the CSA with PASTA (Parallel self timed adder) in the proposed architecture.

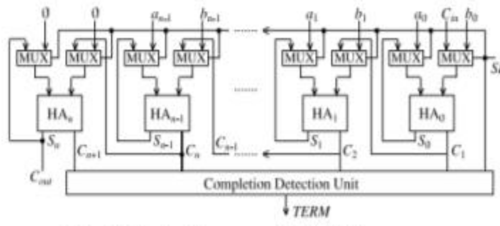

Fig.3.Block diagram of PASTA

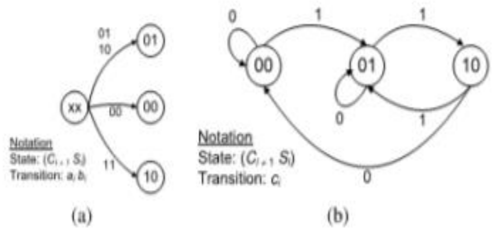

Fig.4. State diagram of PASTA (a).Initial Phase (b).Iterative Phase

Montgomery multiplication is to perform fast modular multiplication(MM).PASTA adder using in Montgomery modular multiplication is to reduced area and clock cycles.To design a simple and efficient radix-2 Montgomery Modular multiplication with Parallel Self Timed Adder (PASTA).The design of PASTA is uses half adders (HAs) along with multiplexers requiring minimal interconnections. The selection input for two-input multiplexers corresponds to the Request handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL=0andwill switch to feedback/carry paths for subsequent iterations using SEL=1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values are show in Fig .3. In Fig. 4, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (Ci+1 Si) pair where Ci+1, Si represent carry out and sum values, respectively, from the ith bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state (11) cannot appear.

The proposed architecture of Montgomery Modular Multiplication using PASTA adder, which consists of one one-level Parallel Self Timed Adder(PASTA) architecture, two 4-to-1 multiplexers (M1 and M2) one simplified multiplier SM3, one skip detector Skip_D, one zero detector Zero_D, and six registers. Zero detector Zero_D is used to detect SC is equal to zero. The Skip_D is composed of four XOR gates, three AND gates, one NOR gate, and two 2-to-1 multiplexers the skip detector is used to detect the unnecessary multiplication operations.

The design has been implemented using Xilinx Verilog coding. For further verification, the design can be done using Cadence. It can be clearly understand by the waveform shown below. It can be proven that it has reduced area complexity and speed complexity on comparing to all other multipliers. The method has been implemented using a configurable carry save adder so as to prove the maximum delay to be less comparing all. The delay and area can be minimized as much as possible as comparing to all other previous existing architectures.

IV.CONCLUSION

This paper presents low-power high-speed two-stage pipeline MAC architecture for real-time DSP applications. Our basic idea is to integrate a part of additions (including a part of the final addition in the multiplication and a part of the addition in the accumulation) into the PPR process. As a result, critical path delays and power dissipations caused bycarry propagations can be reduced. To correctly deal with the overflow during the PPR process, an α-bit accumulator is used to count the total number of carries. Experimental results consistently show that the proposed approach works well in practice. The proposed MAC architecture is applicable to both the design of an unsigned MAC unit and the design of a signed MAC unit. Note that the only differences between the unsigned MAC unit and the signed MAC unit are the PPM structure and the α-bit addition mechanism. Moreover, the proposed MAC architecture is also applicable to the systolic array (for performing the matrix multiplication). Implementation data show that, compared with the systolic array based on the conventional PE (i.e., the conventional MAC architecture), the systolic array based on the proposed PE (i.e., the proposed MAC architecture) can greatly reduce both circuit area and power consumption under the same timing constraint.SCS-based multipliers maintain the input and Output operands of the Montgomery MM in the carrysave format to escape from the format conversion, leading to fewer clock cycles but smaller area than FCS-based multiplier. In the existed architecture disadvantages are carry propagation delay and extra clock cycles. To overcome the disadvantages we go for PASTA adder. The PASTA adder is using in Montgomery Modular Multiplier in these advantages are low hard ware cost short critical path delay and required clock cycles are reduced for completing one MM operation.

REFERENCES

[1] V.Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137,Jan. 2013.

[2] [2] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., Nov. 1998, pp. 1178–1182.

[3] [3] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[4] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[5] Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[6] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large Scale
Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[7] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, “Design-efficient approximate multiplication circuits through partial product perforation,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[8] P. Kulkarni, P. Gupta, and M. D. Ercegovac, “Trading accuracy for power in a multiplier architecture,” J. Low Power Electron., vol. 7, no. 4, pp. 490–501, 2011.

[9] C.-H. Lin and C. Lin, “High accuracy approximate multiplier with error correction,” in Proc. IEEE 31st Int. Conf. Comput. Design, Sep. 2013, pp. 33–38.

[10] Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” in Proc. Conf. Exhibit. (DATE), 2014, pp. 1–4.