DOI: 30.0485/ijearst.10.03.3480 https://ijearst.co.in/

ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2386-7876
Vol.02, Issue.03
May-2026
Pages: 962-973

HIGH-PERFORMANCE STATIC SEGMENTED APPROXIMATE MAC WITH CARRY SKIP ADDER

[1]ADINA YAMINI SATYA SAI DEVI, [2]Dr. BH. V. V. S. R. K. K. PAVAN
[1]M. Tech, Dept. of ECE, BVC Institute of Technology & Science, Batlapalem, Indupalli, AP
[2]Professor, Dept. of ECE, BVC Institute of Technology & Science, Batlapalem, Indupalli, AP

ABSTRACT: This project proposes a high-speed and energy-efficient two-cycle multiply-accumulate (MAC) architecture that supports whole numbers. MAC consists multiplier and adder units. This multiplier uses a new partial-product reduction format which consecutively reduces the maximum output delay. The proposed architecture uses a unique carry-propagate adder and performs segmentation on the three operands A, B, and C, to reduce hardware cost. The circuit can be configured at design-time by two parameters. The first one controls the segmentation on A and B, while the second one controls the segmentation on C and the adder length. An error compensation technique is also employed, to reduce the approximation error. Further, this project is modified using carry select adder for addition operation in segmented MAC unit in order to reduce both dynamic power and latency constraints.

Keywords: Approximate Computing, Approximate MAC Unit / Ax MAC Error-Resilient Applications, Low-Power/Small-Area Design, Accuracy-Configurable/Scalable.

INTRODUCTION: THE multiply-accumulate (MAC) unit is a common digital block used extensively in microprocessors and digital signal processors for data-intensive applications. For example, many filters, orthogonal frequency-division multiplexing algorithms, and channel estimators require FIR or FFT/IFFT computations that MAC units can accelerate efficiently. **M**ODERN consumer electronics make extensive use of Digital Signal Processing (DSP) providing custom accelerators for the domains of multimedia, communications etc. Typical DSP applications carry out a large number of arithmetic operations as their implementation is based on computationally intensive kernels, such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Finite Impulse Response (FIR) filters and signals' convolution. As expected, the performance1 of DSP systems is inherently affected by decisions on their design regarding the allocation and the architecture of arithmetic units. Recent research activities in the field of arithmetic optimization [1], [2] have shown that the design of arithmetic components combining operations which share data, can lead to significant performance improvements. Based on the observation that an addition can often be subsequent to a multiplication (e.g., in symmetric FIR filters), the Multiply-Accumulator(MAC) and Multiply-Add (MAD) units were introduced [3] leading to more efficient implementations of DSP algorithms compared to the conventional ones, which use only primitive resources [4]. Several architectures have been proposed to optimize the performance of the MAC operation in terms of area occupation, critical path delay or power consumption [5]–[7]. As noted in [8], MAC components increase the flexibility of DSP data path synthesis as a large set of arithmetic

operations can be efficiently mapped onto them. Except the MAC/MAD operations, many DSP applications are based on Add-Multiply (AM) operations (e.g., FFT algorithm [9]). Since the last decade the semiconductor industry has experienced an exponential growth of integration of sophisticated multi-media applications into portable gadgets. The major concern of portable gadgets is the battery life, which influences the real-time processing applications and their dynamic range of input signals for additive features. It is the high time to explore the challenging criteria of these emerging low power, low area and high performance digital signal processing chips [1]. In digital VLSI circuits, computation is the critical part and it decides the power consumption and operating speed of the designs. For computations arithmetic circuits involves adders and multipliers; which are the most copiously used components.

LITERATUTRE SURVEY: Song Wen-miao, Liu Yan-ming, Zhang Shu-e [3]: This paper describes about SMAC protocol which is an energy efficient protocol of WSN. According to this paper SMAC adopts a periodic listen and sleep mechanism to save energy from idle listening. Energy is saved when the node goes to sleep interval. A complete cycle is called a frame. All nodes can choose their own listen/sleep schedules. Neighbor nodes synchronize together to reduce control overhead. To avoid collision and overhearing, SMAC protocol uses RTS/CTS mechanism. To reduce the control overhead of RTS and CTS SMAC reserves the medium for transmitting all the fragments in burst. 3.2 Rohan Parmar, Dr. R C Poonia "A Literature Survey on Energy Efficient MAC Protocols for WSN" [4]: This paper describes TMAC protocol which is a successor of SMAC. TMAC protocol is used to overcome the problem faced by SMAC i.e. performance under variable traffic load. In T-MAC listening period ends when no activation event has occurred for a time threshold TA. The decision for TA is presented along with some solutions to the early sleeping problem defined in. Variable load in sensor networks are expected, since the nodes that are closer to the sink must relay more traffic. Although T-MAC gives better results under these variable loads, the synchronization ofthe listen periods within virtual clusters is broken. This is one of the reasons for the early sleeping problem. 3.3 Changsu Suh and Young-Bae Ko [5] : This paper describes a MAC protocol named as TEEM (Traffic aware Energy Efficient MAC) in which the duration of listen and sleep modes are not fixed like SMAC rather they are adaptive by utilizing traffic information of each node which results in low consumption of power. By utilizing the traffic information the listen time of nodes can be reduced by putting them into sleep state earlier when they expect no traffic to occur. It is done by modifying SMAC in two aspects: Firstly, the nodes are turned off when they expect no traffic to occur. Secondly, by eliminating separate RTS control packet when data transfer is likely to occur. Also the size of control packet is reduced by combining the RTS and SYNC packet. The combined packet is called SYNCrts. 3.4 R. Ramya, G. Saravanakumar and S. Ravi "MAC Protocols for Wireless Sensor Networks" [6]: This paper describes about a MAC protocol named as μ-MAC which is a type of TDMA based MAC protocol. In μ-MAC high sleep ratio are obtained which is retaining the message reliability and latency. It is based on a schedule-based approach by which shared medium is accessed, which is predicted by behavior of traffic. Single time-slotted channel is used in μ-MAC protocol. Operations of this protocol alternate between a contention-free period and contention period. 3.5 Zahra Rezaei , Shima Mobininejad "Energy Saving in Wireless Sensor Network" [7]: In this paper a MAC protocol named as DEE-MAC has been explained which lets the idle listening nodes go into sleep using synchronization performed at cluster head. It is a TDMA based MAC protocol. A round is the time duration between a node disseminates its interest to the event and receives the response from the event. Each round comprise of a cluster formation and transmission phases. DEE-MAC operations comprise of

these two phases. Each of the rounds includes a cluster formation phase and a transmission phase. In the cluster formation phase, a node decides whether to become the cluster head based on its remaining power. The node with the highest power level is elected as the cluster head. Each new round introduces formation of another cluster with different group of nodes based on the current node power level and the network structure changes. After the successful cluster head election, the system enters the transmission phase.

EXISTING METHOD:

STATIC SEGMENT METHOD: The SSM employs an m × m multiplier for realizing an n × n product, with n/2 ≤ m < n. Let us consider the operand A. The signal is segmented in a lower portion (LA), comprising its m least significant bits (LSBs), and an upper portion (HA), comprising its m most significant bits (MSBs), that is:

$$L_A = A[m-1:0]$$
$$H_A = A[n-1:n-m] \qquad (1)$$

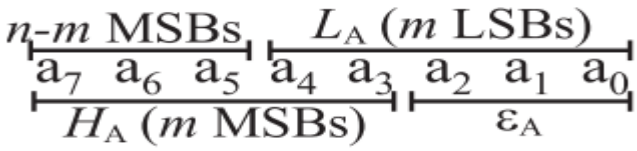

Fig. 1. Segmentation for the input A.

The Fig. 1 shows an example for n = 8, m = 5. When the (n−m) MSBs of A are low, LA is selected for the multiplication. Otherwise, HA is chosen at the cost of an accuracy loss (since the least significant part εA is discarded, see Fig. 1). Let us define αA as the OR of the (n-m) MSBs of A and let us name Assm the segmented input (Assm = LA if αA = 0 and Assm = HA if αA = 1). The following relation holds:

$$A \simeq A_{ssm} \cdot 2^{K_A} \text{ where : } K_A = \begin{cases} 0 & \text{if } \alpha_A = 0 \\ n-m & \text{if } \alpha_A = 1 \end{cases} \qquad (2)$$

where an approximation error occurs when αA = 1, due to discarding εA. A similar segmentation is applied to B and the output of the multiplier is approximated as follows:

$$P_{SSM} = A \times B \simeq (A_{ssm} 2^{K_A}) \times (B_{ssm} 2^{K_B})$$
$$= A_{ssm} \times B_{ssm} \cdot 2^{K_S} \qquad (3)$$

$$K_S = \begin{cases} 0 & \text{if } (\alpha_A,\ \alpha_B) = 00 \\ n-m & \text{if } (\alpha_A,\ \alpha_B) = 01,\ 10 \\ 2 \cdot (n-m) & \text{if } (\alpha_A,\ \alpha_B) = 11 \end{cases} \qquad (4)$$

The (3) clearly shows that the SSM requires only a small m × m multiplier to compute the inner product Assm × Bssm. The SSM implementation is reported in the dashed box of Fig. 2(a). To obtain a multiplier-accumulator we can simply add the operand C to the output of the SSM, as follows:

$$Y_{SSMAC} = \left(A_{ssm} \times B_{ssm} \cdot 2^{K_S}\right) + C \qquad (5)$$

Note that the word length of C is nC ≥ 2n since in most applications the multiply-accumulate unit is used sequentially to accumulate the output of several multiplications.

The implementation of the above equation is shown in Fig. 2(a). The left-shift needed to calculate PSSM does not allow to merge the final adder with the inner m × m multiplier Assm × Bssm. Therefore, two cascaded carry-propagate adders are required: a first 2m-bits adder is used to compute Assm × Bssm, while a second nC-bits adder sums the operand C.

STATIC SEGMENTED MAC

The presence of the two adders in cascade negatively affects the performance of the circuit in Fig. 2(a). To solve this problem, we rewrite (5) as follows:

$$Y_{SSMAC} = \left(A_{ssm} \times B_{ssm} + C \cdot 2^{-K_S}\right) \cdot 2^{K_S} \qquad (6)$$

he product Assm × Bssm. This allows to eliminate the multiplexer between the multiplier and the adder, and to merge the term C · 2−KS in the PPM of the multiplier. The Fig. 2(b) shows the architecture that implements (6), highlighting in red the differences with respect to the MAC of Fig. 2(a). The two OR-gates compute the flags αA and αB, that in turn program two 2-1 multiplexers used to segment A and B. A third multiplexer, marked in red in the figure, applies the right-shift on C, choosing between the portions C[nC-1:0], C[nC-1:n-m], and C[nC-1:2·(n-m)]. The multiplexer on the output rearranges the result including the bits C[n-m-1:0] and C[2·(n-m)-1:0] at the least significant positions in case of right-shift. Thus, the architecture in Fig. 2(b) allows to merge the inner multiplication and the final sum in a fused MAC structure, again highlighted in red, requiring a unique nC-bits adder. There is a small overhead due to the introduction of the multiplexer on C input, but this overhead is more than compensated by the elimination of the 2m-bits adder required in Fig. 2(a).

Segmentation of the Addend C:

As observed before, the architecture in Fig. 2(b), while more effective compared to Fig. 2(a), requires an nC-bit adder to compute the output. We can reduce the adder length by segmenting also the input C. By following an approach like Fig. 1, we subdivide the input C in a lower portion (LC) and an upper portion (HC) which comprises mC bits, with nC/2 ≤ mC < nC:

LC = C[mC − 1 : 0]

HC = C[nC − 1 : nC − mC] (7)

Thus, the proposed static-segmented MAC (SSMAC) can be configured at design time with two parameters: m that controls the segmentation of A and B, and mC that controls the segmentation of C. In the following we will assume mC ≥ 2m. Let us name αC the OR of the (nC-mC) MSBs of C and let us define Cssm the segmented C input (Cssm = LC if αC = 0 and Cssm = HC if αC = 1). Like (2), we have:

$$C \simeq C_{ssm} \cdot 2^{K_C} \text{ where : } K_C = \begin{cases} 0 & \text{if } \alpha_C = 0 \\ n_C - m_C & \text{if } \alpha_C = 1 \end{cases} \quad (8)$$

From (6), the output of the SSMAC is written as:

$$Y_{SSMAC} = \left(A_{ssm} \times B_{ssm} + C_{ssm} \cdot 2^{K_C - K_S}\right) \cdot 2^{K_S} \quad (9)$$

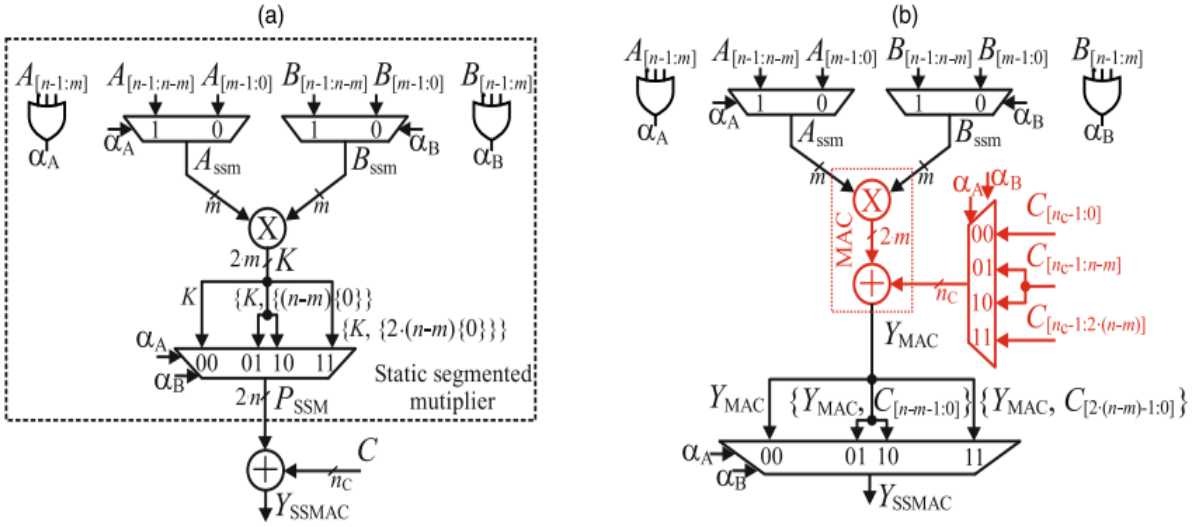

Fig. 2. MAC realized (a) by cascading a static segmented multiplier (in the dashed line) and an adder, and (b) by implementing the proposed

static segmented multiplier-accumulator, according to (6). In the MAC of (a), two cascaded carry-propagate adders are required: The first one in the inner multiplier that computes Assm × Bssm, and the second one that adds the operand C. In the MAC of (b), the inner multiplication and the addition are merged, requiring a unique carry propagate adder. The multiplexer on C input introduces only a small overhead.

The word length of the product Assm × Bssm is 2m bits, and the word length of Cssm is mC. Unfortunately, due to the term 2KC −KS (that can imply a left or a right shift ofCssm), we cannot use a simple mC-bits adder to compute the output in (9). We thus consider the various cases that can occur in (9), depending on the values of αC, αB, αA. When (αC, αB, αA) = 000, we have KS = KC = 0 and: YSSMAC = Assm × Bssm + Cssm (10) where Assm = LA, Bssm = LB and Cssm = LC. The last equation requires a mC-bits adder, as desired. In the other cases, we perform some additional approximations by truncating some of the terms in (9); these approximations are different depending on the sign of KC−KS. To fix ideas, let us assume: (nC-mC) ≥ 2(n−m), so thatKC-KS is negative only when KC = 0 and KS 0. Let us consider the case in which KC-KS is positive, that is KC 0. We are in this case when αC = 1, independently on the values of αA and αB. We rewrite (9) as follows:

$$Y_{SSMAC} = \left(A_{ssm} \times B_{ssm} 2^{-(K_C - K_S)} + C_{ssm}\right) \cdot 2^{K_C} \quad (11)$$

We approximate this equation by truncating the term Assm × Bssm 2−(KC −KS) :

$$Y_{SSMAC} \simeq \left(\left\lfloor A_{ssm} \times B_{ssm} 2^{-(K_C - K_S)} \right\rfloor + C_{ssm} \right) \cdot 2^{K_C} \quad (12)$$

having indicated as the floor operator. Let us assume, for the sake of simplicity, that KC-KS is even. The (12) can be rewritten as follows:

$$Y_{SSMAC} \simeq (A'_{ssm} \times B'_{ssm} + C_{ssm}) \cdot 2^{K_C} \quad (13)$$

where:

$$A'_{ssm} = \left\lfloor A_{ssm} \cdot 2^{-(K_C - K_S)/2} \right\rfloor$$

$$B'_{ssm} = \left\lfloor B_{ssm} \cdot 2^{-(K_C - K_S)/2} \right\rfloor \quad (14)$$

Thus, the approximation (13) can be implemented by truncating the (KC-KS)/2 rightmost bits of Assm and Bssm. In other words, A ssm is simply a sub-segment of HA or LA, and a similar observation holds for B ssm. In the case in which KC-KS is odd, we can still use (13), but A ssm and B ssm will have a number of bits that differ by one, and are given by the following equation:

$$A'_{ssm} = \left\lfloor A_{ssm} \cdot 2^{-\lceil (K_C - K_S)/2 \rceil} \right\rfloor$$

$$B'_{ssm} = \left\lfloor B_{ssm} \cdot 2^{-\lfloor (K_C - K_S)/2 \rfloor} \right\rfloor \quad (15)$$

where · is the ceil operator. The Fig. 3 shows an example for n = 8, m = 5 and nC = 20, mC = 14. In Fig. 3(a) the entire (exact) PPM of the multiplieraccumulator is shown. The Fig. 3(b) highlights the portion of PPM considered for (αC, αB, αA) = 111. In this case A ssm and B ssm include the five most-significant bits of A and B, therefore the bits a2..a0, b2..b0 are truncated. The segmentCssm, including the 14 most-significant bits of C, is employed and hence the bits c5..c0 are also truncated. The Fig. 3(c) shows the portion of PPM considered for (αC, αB, αA) = 101. In this case A ssm includes the three most significant bits of A, while B ssm includes the bits b4b3b2b1 (the bit b0 is truncated, while b7b6b5 are zero since αB = 0). Let us now consider the case in which KC-KS is negative, that is KC = 0 and KS 0. We are in this case when αC = 0, and one of αA and αB, or both, are equal to 1. In these conditions,

$$Y_{SSMAC} = (A_{ssm} \times B_{ssm} + C_{ssm} \cdot 2^{-K_S}) \cdot 2^{K_S} \quad (16)$$

This equation can be approximated by truncating $C_{ssm} \cdot 2^{-K_S}$ as follows:

$$Y_{SSMAC} \simeq (A_{ssm} \times B_{ssm} + C'_{ssm}) \cdot 2^{K_S} \quad (17)$$

where:

$$C'_{ssm} = \lfloor C_{ssm} \cdot 2^{-K_S} \rfloor \quad (18)$$

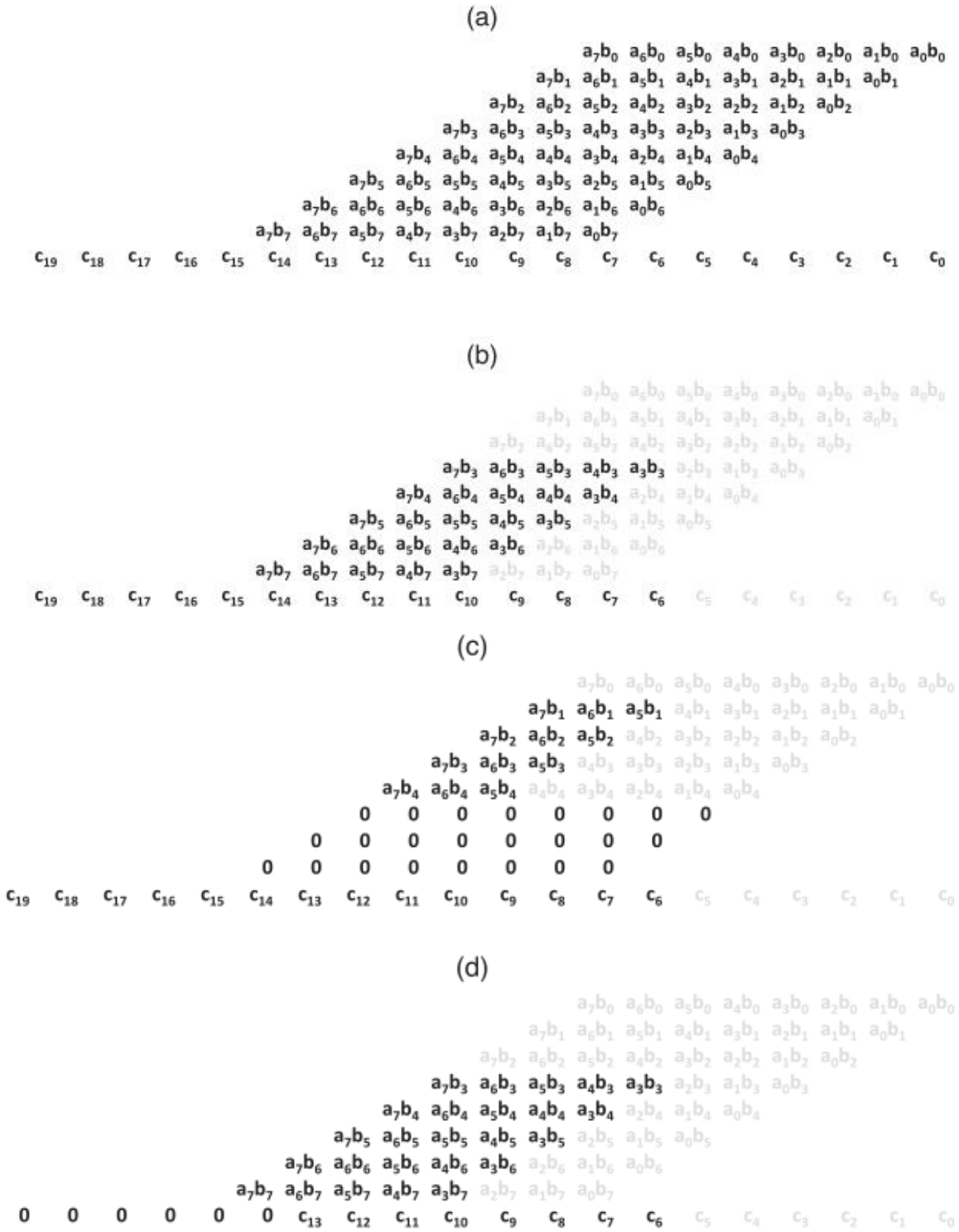

Fig. 3. Multiplier accumulator example, with n = 8, m = 5, nC = 20, mC = 14. (a) Partial product matrix of the entire (exact) multiplier accumulator. (b) Portion of the partial product matrix considered for (αC, αB,

αA) = 111. Bits a2..a0, b2..b0, c5..c0 are truncated. (c) Portion of the partial product matrix considered for (αC, αB, αA) = 101. Bits a4..a0, b0, c5..c0 are truncated. Bits b7b6b5 are zero. (d) Portion of the partial product matrix considered for (αC, αB, αA) = 011. Bits a2..a0, b2..b0, c5..c0 are truncated. Bits c19 …c14 are zero.

PROPSOED METHOD:

Carry Skip Adder (CSKA), or carry-bypass adder, is a high-speed digital adder that reduces propagation delay by allowing the carry signal to "skip" blocks of bits when the block propagate signal is high. It balances the simplicity of a Ripple Carry Adder (RCA) with the speed of a Carry Lookahead Adder (CLA)

- **Structure:** Composed of multiple full-adder blocks, a carry propagation check (using AND gates), and a multiplexer to bypass the block.
- **Skip Mechanism:** If all bits (\(A_i, B_i\)) within a block are not identical (i.e., \(A_i \oplus B_i = 1\) for all \(i\)), the carry-in (\(C_{in}\)) of the block can directly propagate to the carry-out (\(C_{out}\)) of that block, bypassing the ripple carry chain.
- **Block Propagate Signal (\(P_{B}\)):** Calculated for each block, \(P_{B}\) is true if the block allows the carry to skip. It is the output of an AND gate receiving propagate signals from each full adder.
- **Performance:** Significantly faster than RCA due to reduced critical path, particularly for larger bit sizes (e.g., 16-bit, 32-bit).

- **Block Division:** Divide the \(n\)-bit addition into groups (e.g., 4-bit blocks).
- **Local Addition:** Use RCA within each block to generate local sum bits and a block carry-out (\(C_{4}\)).
- **Propagate Logic:** Generate a "skip" signal (Block Propagate \(P_{B}\)) by ANDing all individual \(P_i = A_i \oplus B_i\) signals in the block.
- **Multiplexer (Skip Logic):** Use the \(P_{B}\) signal to drive a multiplexer. If \(P_B=1\), the carry skips the block (\(C_{out} = C_{in}\)). If \(P_B=0\), the multiplexer selects the ripple carry-out (\(C_{4}\)).

The conventional structure of the CSKA consists of stages containing chain of full adders (FAs) (RCA block) and 2:1 multiplexer (carry skip logic). The RCA blocks are connected to each other through 2:1 multiplexers, which can be placed into one or more level structures [19]. The CSKA configuration (i.e., the number of the FAs per stage) has a great impact on the speed of this type of adder . Many methods have been suggested for finding the optimum number of the FAs. The techniques presented in [19] make use of VSSs to minimize the delay of adders based on a singlelevel carry skip logic. Some methods to increase the speed of the multilevel CSKAs are proposed. The techniques, however, cause area and power increase considerably and less regular layout. The design of a static CMOS CSKA where the stages of the CSKA have a variable sizes was suggested in [18]. In addition, to lower the propagation delay of the adder, in each stage, the carry look-ahead logics were utilized. Again, it had a complex layout as well as large power consumption and area usage. In addition, the design approach, which was presented only for the 32-bit adder, was not general to be applied for structures with different bits lengths. Alioto and Palumbo [19] propose a simple strategy for the design of a single-level CSKA. The method is based on the VSS technique

where the near-optimal numbers of the FAs are determined based on the skip time (delay of the multiplexer), and the ripple time (the time required by a carry to ripple through a FA). The goal of this method is to decrease the critical path delay by considering a noninteger ratio of the skip time to the ripple time on contrary to most of the previous works, which considered an integer ratio [17], [20]. In all of the works reviewed so far, the focus was on the speed, while the power consumption and area usage of the CSKAs were

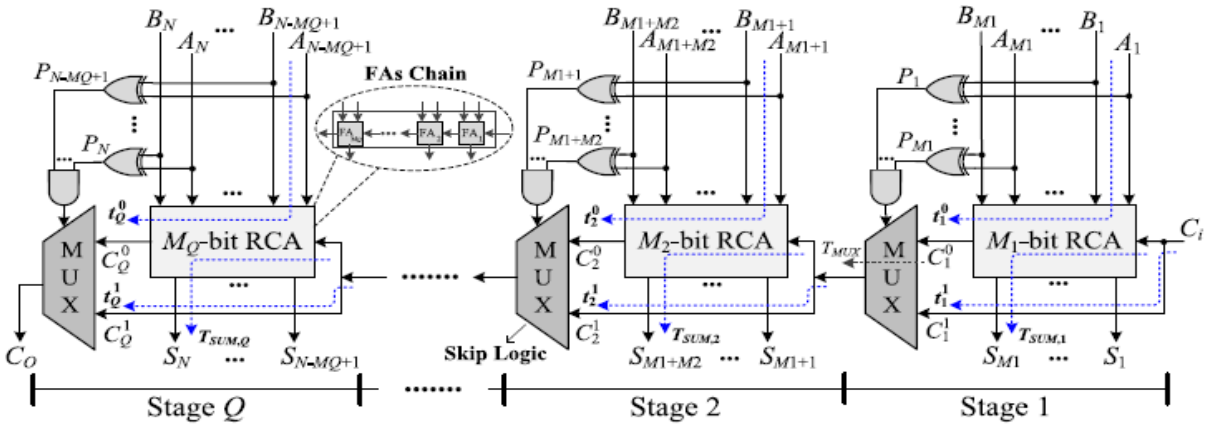

Fig. 4 Proposed structure of the CSKA

not considered. Even for the speed, the delay of skip logics, which are based on multiplexers and form a large part of the adder critical path delay [19], has not been reduced.

RESULT:

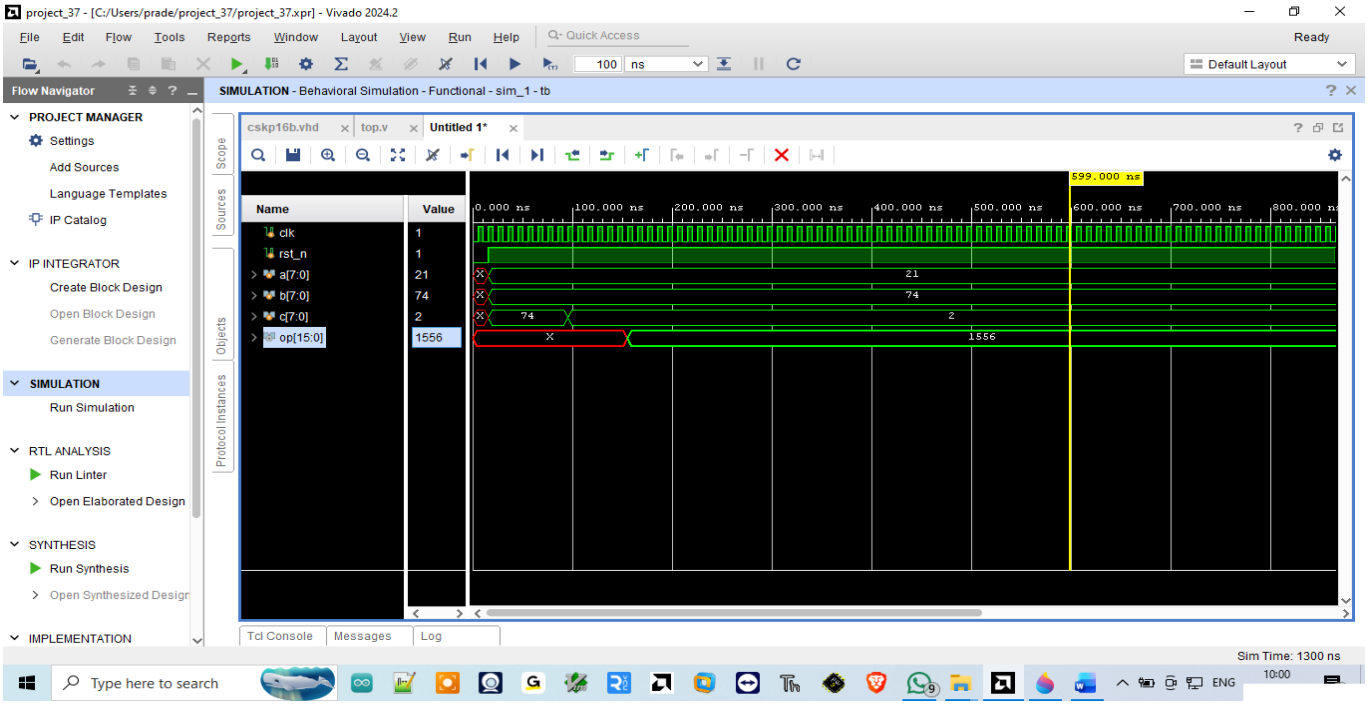

Fig a: Proposed Simulation Result

CONCLUSION:

This paper focuses on optimizing the design of multiply accumulation unit (MAC) operator. The proposed architectures have yielded better efficiencies in than existing architectures. The developed architecture uses a unique carry-propagate adder and performs segmentation on the three operands, by using a reduced word length adder. A correction technique is also proposed for recovering precision. The proposed circuit can be configured at design-time by two parameters. The first one controls the segmentation on A and B, while the second one controls the segmentation on C and the adder length.

Future Scope

Integration of the proposed approximate MAC architecture into advanced AI accelerators and neural network processors for energy-efficient deep learning applications. Development of adaptive or dynamically reconfigurable segmentation techniques where the approximation level changes according to workload requirements. Extension of the static segmentation concept to higher bit-width MAC units such as 16-bit, 32-bit, and floating-point arithmetic units. Application of the architecture in edge computing and IoT devices where ultra-low power consumption is critical.

REFERENCES:

[1] Chang, Chip-Hong, Jiangmin Gu, and Mingyan Zhang. "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits." Circuits and Systems I: Regular Papers, IEEE Transactions on 51.10 (2004): 1985-1997.

[2] Tung Thanh Hoang; Sjalander, M.; Larsson-Edefors, P., "A High-Speed, Energy-Efficient Two-Cycle MultiplyAccumulate (MAC) Architecture and Its Application to a Double-Throughput MAC Unit," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.57, no.12, pp.3073,3081, Dec. 2010.

[3] Chen Ping-hua; Zhao Juan, "High-speed Parallel 32×32-b Multiplier Using a Radix-16 Booth Encoder," Intelligent Information Technology Application Workshops, 2009. IITAW '09. Third International Symposium on , vol., no., pp.406,409, 21-22 Nov. 2009

[4] Kiwon Choi; Minkyu Song, "Design of a high performance 32×32-bit multiplier with a novel sign select Booth encoder," Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on , vol.2, no., pp.701,704 vol. 2, 6-9 May 2001.

[5] Rajput, R.P.; Swamy, M.N.S., "High Speed Modified Booth Encoder Multiplier for Signed and Unsigned Numbers," Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on , vol., no., pp.649,654, 28-30 March 2012.

[6] Yangbo Wu; Weijiang Zhang; Jianping Hu, "Adiabatic 4-2 compressors for low-power multiplier," Circuits and Systems, 2005. 48th Midwest Symposium on , vol., no., pp.1473,1476 Vol. 2, 7-10 Aug. 2005.

[7] Jaina, D.; Sethi, K.; Panda, R., "Vedic Mathematics Based Multiply Accumulate Unit," Computational Intelligence and Communication Networks (CICN), 2011 International Conference on, vol., no., pp.754,757, 7-9 Oct. 2011.

[8] Aliparast, Peiman, Ziaadin D. Koozehkanani, and Farhad Nazari. "An Ultra High Speed Digital 4-2 Compressor in 65-nm CMOS." International Journal of Computer Theory & Engineering 5.4 (2013).

[9] N. Weste and David Harris, “CMOS VLSI Design- A Circuits & System Perspective”, Pearson Education, 2008.

[10] ChandraMohan U, "Low Power Area Efficient Digital Counters", Proceedings of the 7th VLSI Design and Test Workshops, VDAT, August 2003.
[11] Narendra C P & Ravi K M Kumar, "Efficient Comparator based Sum of Absolute Differences Architecture for Digital Image Processing Applications", Foundation of Computer Science, New York, USA, International Journal of Computer Applications, 96(4):17-24, June 2014.
[12] W.-C. Yeh, "Arithmetic Module Design and its Application to FFT," Ph.D. dissertation, Dept. Electron. Eng., National Chiao-Tung University, , Chiao-Tung, 2001.
[13] R. Zimmermann and D. Q. Tran, "Optimized synthesis of sum-of-products," in Proc. Asilomar Conf. Signals, Syst. Comput., Pacific Grove, Washington, DC, 2003, pp. 867–872.
[14] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs. Oxford: Oxford Univ. Press, 2000.
[15] O. L. Macsorley, "High-speed arithmetic in binary computers," Proc. IRE, vol. 49, no. 1, pp. 67–91, Jan. 1961.
[16] N. H. E. Weste and D. M. Harris, "Datapath subsystems," in CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Readington: Addison-Wesley, 2010, ch. 11.
[17] S. Xydis, I. Triantafyllou, G. Economakos, and K. Pekmestzi, "Flexible datapath synthesis through arithmetically optimized operation chaining," in Proc. NASA/ESA Conf. Adaptive Hardware Syst., 2009, pp. 407–414.
[18] [Online]. Available:
http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCUltra/Pages/default.aspx
[19] [Online]. Available:
http://www.synopsys.com/Tools/Implementation/SignOff/PrimeTime/Pages/default.aspx
[20] Z. Huang, "High-Level Optimization Techniques for Low-Power Multiplier Design," Ph.D., University of California, Department of Computer Science, Los Angeles, CA, 2003.
[21] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, 1964