DESIGN OF HIGH PERFORMANCE MAC UNIT BY USING CARRY SKIP ADDER

[1]Mr. G. SHIVA KUMAR, [2] Mr .P. VENKATESWARLU, [3] Dr.Mr.ARVIND KUNDU
[1]PG Student, [2]Assistant professor, [3]Associate Professor
[1]Department of ECE, [1]SCIENT INSTITUTE OF TECHNOLOGY, HYDERABAD

ABSTRACT: A design of high performance16-bit Multiplier-and-Accumulator (MAC) is implemented in this project.MAC unit performs important operation in many of the digital signal processing (DSP) applications. The multiplier is designed using array multiplier and the adder is done with carry skip adder. The total design is coded with verilog-HDL and the synthesis is done using Cadence RTL complier using typical libraries of Xilinx 13.2 Version.

Key words: Carry skip adder (CSKA), energy efficient, high performance, hybrid variable latency adders, voltage scaling.

I.INTRODUCTION:

MAC unit is an inevitable component in many digital signal processing (DSP) applications involving multiplications and/or accumulations. MAC unit is used for high performance digital signal processing systems. The DSP applications include filtering, convolution, and inner products. Most of digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) or discrete wavelet transforms (DWT). Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic determines the execution speed and performance of the entire calculation. Multiplication-and-accumulate operations are typical for digital filters. Therefore, the functionality of the MAC unit enables high-speed filtering and other processing typical for DSP applications. Since the MAC unit operates completely independent of the CPU, it can process data separately and thereby reduce CPU load. The application like optical communication systems which is based on DSP , require extremely fast processing of huge amount of digital data. The Fast Fourier Transform (FFT) also requires addition and multiplication.

A MAC unit consists of a multiplier and an accumulator containing the sum of the previous successive products. The MAC inputs are obtained from the memory location and given to the multiplier block.

Recently, the near-threshold region has been considered as a region that provides a more desirable tradeoff point between delay and power dissipation compared with that of the subthreshold one, because it results in lower delay compared with the subthreshold region and significantly lowers switching and leakage powers compared with the superthreshold region. In addition, near-threshold operation, which uses supply voltage levels near the threshold voltage of transistors [11], suffers considerably less from the process and environmental variations compared with the subthreshold region.

The dependence of the power (and performance) on the supply voltage has been the motivation for design of circuits with the feature of dynamic voltage and frequency scaling. In these circuits, to reduce the energy consumption, the system may change the voltage (and frequency) of the circuit based on the workload requirement [12]. For these systems, the circuit should be able to operate under a wide range of supply voltage levels. Of course, achieving higher speeds at lower supply voltages for the computational blocks, with the adder as one the main components, could be crucial in the design of high-speed, yet energy efficient, processors. In addition to the knob of the supply voltage, one may choose between different adder structures/families for optimizing power and speed. There are many adder families with different delays, power consumptions, and area usages. Examples include ripple carry adder (RCA), carry increment adder (CIA), carry skip adder (CSKA), carry select adder (CSLA), and parallel prefix adders (PPAs). The descriptions of each of these adder architectures along with their characteristics may be found in [1] and [13]. The RCA has the simplest structure with the smallest area and power consumption but with the worst critical path delay. In the CSLA, the speed, power consumption, and area usages are considerably larger than those of the RCA. The PPAs, which are also called carry look-ahead adders, exploit direct parallel prefix structures to generate the carry as fast as possible [14]. There are different types of the parallel prefix algorithms that lead to different PPA structures with different performances. As an example, the Kogge–Stone adder (KSA) [15] is one of the fastest structures but results in large power consumption and area usage. It should be noted that the structure complexities of PPAs are more than those of other adder schemes.

The CSKA, which is an efficient adder in terms of power consumption and area usage, was introduced in [17]. The critical path delay of the CSKA is much smaller than the one in the RCA, whereas its area and power consumption are similar to those of the RCA. In addition, the power-delay product (PDP) of the CSKA is smaller than those of the CSLA and PPA structures [19]. In addition, due to the small number of transistors, the CSKA benefits from relatively short wiring lengths as well as a regular and simple layout [18]. The comparatively lower speed of this adder structure, however, limits its use for high-speed applications.

In this paper, given the attractive features of the CSKA structure, we have focused on reducing its delay by modifying its implementation based on the static CMOS logic. The concentration on the static CMOS originates from the desire to have a reliably operating circuit under a wide range of supply voltages in highly scaled technologies [10]. The proposed modification increases the speed considerably while maintaining the low area and power consumption features of the CSKA. In addition, an adjustment of the structure, based on the variable latency technique, which in turn lowers the power consumption without considerably impacting the CSKA speed, is also presented. To the best of our knowledge, no work concentrating on design of CSKAs operating from the superthreshold region down to near-threshold region and also, the design of (hybrid) variable latency CSKA structures have been reported in the literature. Hence, the contributions of this paper can be summarized as follows.

1) Proposing a modified CSKA structure with MAC by combining the concatenation and the incrementation schemes to the conventional CSKA (Conv-CSKA) structure for enhancing the speed and energy efficiency of the adder. The modification provides us with the ability to use simpler carry skip logics based on the AOI/OAI compound gates instead of the multiplexer.

2) Providing a design strategy for constructing an efficient CSKA structure with MAC based on analytically expressions presented for the critical path delay

3) Investigating the impact of voltage scaling on the efficiency of the proposed CSKA structure (from the nominal supply voltage to the near-threshold voltage).

4) Proposing a hybrid variable latency CSKA structure based on the extension of the suggested CSKA, by replacing some of the middle stages in its structure with a PPA, which is modified in this paper.

II.CARRY-SKIP ADDER

A **carry-skip adder** (also known as a **carry-bypass adder**) is an adder implementation that improves on the delay of a ripple-carry adder with little effort compared to other adders. The improvement of the worst-case delay is achieved by using several carry-skip adders to form a block-carry-skip adder.

2.1Single carry-skip adder: The worst case for a simple one level carry-ripple-adder occurs, when the propagate-condition is true for each digit pair (a_i, b_i). Then the carry-in ripples through the n-bit adder and appears as the carry-out after $\tau_{CRA}(n) \approx n \cdot \tau_{VA}$.

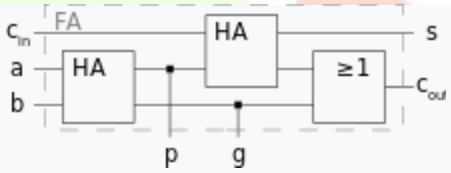

Fig1: Full adder with additional generate and propagate signals.

For each operand input bit pair (a_i, b_i) the propagate-conditions $p_i = a_i \oplus b_i$ are determined using an XOR-Gate (see). When all propagate-conditions are *true*, then the carry-in bit c0 determines the carry-out bit.

The *n*-bit-carry-skip adder consists of a *n*-bit-carry-ripple-chain, a *n*-input AND-gate and one multiplexer. Each propagate bit p_i, that is provided by the carry-ripple-chain is connected to the *n*-input AND-gate. The resulting bit is used as the select bit of a multiplexer that switches either the last carry-bit Cn or the carry-in C0 to the carry-out signal Cout

$$s = p_{n-1} \wedge p_{n-2} \wedge \cdots \wedge p_1 \wedge p_0 = p_{[0:n-1]}$$

This greatly reduces the latency of the adder through its critical path, since the carry bit for each block can now "skip" over blocks with a *group* propagate signal set to logic 1 (as opposed to a long ripple-carry chain, which would require the carry to ripple through each bit in the adder). The number of inputs of the AND-gate is equal to the width of the adder. For a large width, this becomes impractical and leads to additional delays, because the AND-gate has to be built as a tree. A good width is achieved, when the sum-logic has the same depth like the *n*-input AND-gate and the multiplexer.

2.2 Carry-skip optimization

The problem of determining the block sizes and number of levels required to make the physically fastest carry skip adder is known as the 'carry-skip adder optimization problem'. This problem is made complex by the fact that a carry-skip adders are implemented with physical devices whose size and other parameters also affects addition time.

The carry-skip optimization problem for variable block sizes and multiple levels for an arbitrary device process node was solved by Thomas W. Lynch in.[2] This reference also shows that carry-skip addition is the same as parallel prefix addition and is thus related to, and for some configurations identical to, the Hans Carlson, Brent and Kung, Kogge-Stone adderand a number of other adder types.

2.3 Implementation overview

Breaking this down into more specific terms, in order to build a 4-bit carry-bypass adder, 6 full adders would be needed. The input buses would be a 4-bit *A* and a 4-bit *B*, with a carry-in (*CIN*) signal. The output would be a 4-bit bus X and a carry-out signal (*COUT*).

The first two full adders would add the first two bits together. The carry-out signal from the second full adder (C1)would drive the select signal for three 2 to 1 multiplexers. The second set of 2 full adders would add the last two bits assuming C1 is a logical 0. And the final set of full adders would assume that C1 is a logical 1.

The multiplexers then control which output signal is used for *COUT*, X1 and X3

III. MULTIPLY–ACCUMULATE OPERATION

In computing, especially digital signal processing, the multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC, or MAC unit); the operation itself is also often called a MAC or a MAC operation. The MAC operation modifies an accumulator *a*:

$$a \leftarrow a + (b \times c)$$

When done with floating point numbers, it might be performed with two roundings (typical in many DSPs), or with a single rounding. When performed with a single rounding, it is called a fused multiply–add (FMA) or fused multiply–accumulate (FMAC).

Modern computers may contain a dedicated MAC, consisting of a multiplier implemented in combinational logic followed by an adder and an accumulator register that stores the result. The output of the register is fed back to one input of the adder, so that on each clock cycle, the output of the multiplier is added to the register. Combinational multipliers require a large amount of logic, but can compute a product much more quickly than the method of shifting and adding typical of earlier computers. The first processors to be equipped with MAC units were digital signal processors, but the technique is now also common in general-purpose processors.

3.1 In floating-point arithmetic

When done with integers, the operation is typically exact (computed modulo some power of two). However, floating-point numbers have only a certain amount of mathematical precision. That is, digital floating-point arithmetic is generally not associative or distributive. (See Floating point#Accuracy problems.) Therefore, it makes a difference to the result whether the multiply–add is performed with two roundings, or in one operation with a single rounding (a fused multiply–add). IEEE 754-2008 specifies that it must be performed with one rounding, yielding a more accurate result.[1]

3.2 Fused multiply–add

A *fused* multiply–add (sometimes known as *FMA* or *fmadd*)[2] is a floating-point multiply–add operation performed in one step, with a single rounding. That is, where an unfused multiply–add would compute the product $b \times c$, round it to *N* significant bits, add the result to *a*, and round back to *N* significant bits, a fused multiply–add would compute the entire expression $a+b \times c$ to its full precision before rounding the final result down to *N* significant bits.

A fast FMA can speed up and improve the accuracy of many computations that involve the accumulation of products:

- Dot product, Matrix multiplication,Polynomial evaluation (e.g., with Horner's rule), Newton's method for evaluating functions. Convolutions and artificial neural networks

Fused multiply–add can usually be relied on to give more accurate results. However, William Kahan has pointed out that it can give problems if used unthinkingly.[3] If $x^2 - y^2$ is evaluated as $((x \times x) - y \times y)$ using fused multiply–add, then the result may be negative even when $x = y$ due to the first multiplication discarding low significance bits. This could then lead to an error if, for instance, the square root of the result is then evaluated.

When implemented inside a microprocessor, an FMA can actually be faster than a multiply operation followed by an add. However, standard industrial implementations based on the original IBM RS/6000 design require a $2N$-bit adder to compute the sum properly.[4][5]

A useful benefit of including this instruction is that it allows an efficient software implementation of division (see division algorithm) and square root (see methods of computing square roots) operations, thus eliminating the need for dedicated hardware for those operations.

IV.PROPOSED CSKA STRUCTURE

Based on the discussion presented in Section III, it is concluded that by reducing the delay of the skip logic, one may lower the propagation delay of the CSKA significantly. Hence, in this paper, we present a modified CSKA structure that reduces this delay

4.1.General Description of the Proposed Structure

The structure is based on combining the concatenation and the incrementation schemes [13] with the Conv-CSKA structure, and hence, is denoted by CI-CSKA. It provides us with the ability to use simpler carry skip logics. The logic replaces 2:1 multiplexers by AOI/OAI compound gates (Fig. 2). The gates, which consist of fewer transistors, have lower delay, area, and smaller power consumption compared with those of the 2:1 multiplexer [37]. Note that, in this structure, as the carry propagates through the skip logics, it becomes complemented. Therefore, at the output of the skip logic of even stages, the complement of the carry is generated. The structure has a considerable lower propagation delay with a slightly smaller area compared with those of the conventional one. Note that while the power consumptions of the AOI (or OAI) gate are smaller than that of the multiplexer, the power consumption of the proposed CI-CSKA is a little more than that of the conventional one. This is due to the increase in the number of the gates, which imposes a higher wiring capacitance (in the noncritical paths).

Now, we describe the internal structure of the proposed CI-CSKA shown in Fig. 2 in more detail. The adder contains two N bits inputs, A and B, and Q stages. Each stage consists of an RCA block with the size of Mj (j = 1,..., Q). In this structure, the carry input of all the RCA blocks, except for the first block which is Ci , is zero (concatenation of the RCA blocks). Therefore, all the blocks execute their jobs simultaneously. In this structure, when the first block computes the summation of its corresponding input bits (i.e., SM1 ,..., S1), and C1, the other blocks simultaneously compute the intermediate results [i.e., {ZK j+Mj,..., ZK j+2, ZK j+1} for K j = j−1 r=1 Mr(j = 2,..., Q)], and also Cj signals. In the proposed structure, the first stage has only one block, which is RCA. The stages 2 to Q consist of two blocks of RCA and incrementation. The incrementation block uses the

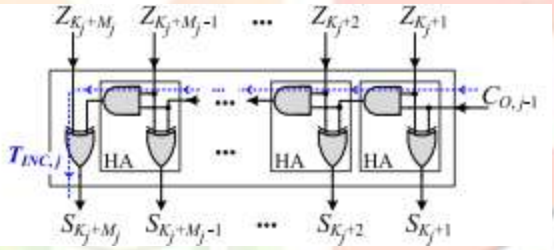

Fig. 2. Internal structure of the jth incrementation block, K j = j−1 r=1 Mr (j = 2,..., Q).

Intermediate results generated by the RCA block and the carry output of the previous stage to calculate the final summation of the stage. The internal structure of the incrementation block, which contains a chain of half-adders (HAs), is shown in Fig. 3. In addition, note that, to reduce the delay considerably, for computing the carry output of the stage, the carry output of the incrementation block is not used. As shown in Fig. 2, the skip logic determines the carry output of the jth stage (CO,j) based on the intermediate results of the jth stage and the carry output of the previous stage (CO,j−1) as well as the carry output of the corresponding RCA block (Cj) . When determining CO,j , these cases may be encountered. When Cj is equal to one, CO,j will be one. On the other hand, when Cj is equal to zero, if the product of the intermediate results is one (zero), the value of CO,j will be the same as CO,j−1 (zero).

The reason for using both AOI and OAI compound gates as the skip logics is the inverting functions of these gates in standard cell libraries. This way the need for an inverter gate, which increases the power consumption and delay, is eliminated. As shown in Fig. 2, if an AOI is used as the skip logic, the next skip logic should use OAI gate. In addition, another point to mention is that the use of the proposed skipping structure in the Conv-CSKA structure increases the delay of the critical path considerably. This originates from the fact that, in the Conv-CSKA, the skip logic (AOI or OAI compound gates) is not able to bypass the zero carry input until the zero carry input propagates from the corresponding RCA block. To solve this problem, in the proposed structure, we have used an RCA block with a carry input of zero (using the concatenation approach). This way, since the RCA block of the stage does not need to wait for the carry output of the previous stage, the output carries of the blocks are calculated in parallel.

4.2.Stage Sizes Consideration

Similar to the Conv-CSKA structure, the proposed CI-CSKA structure may be implemented with either

FSS or VSS. Here, the stage size is the same as the RCA and incrementation blocks size. In the case of the FSS (FSS-CI-CSKA), there are Q = N/M stages with the size of M. The optimum value of M, which may be obtained using (11), is given by

$$Mopt = N(TAOI + TOAI)\ 2(TCARRY + TAND)\ .$$

In the case of the VSS (VSS-CI-CSKA), the sizes of the stages, which are M1 to MQ , are obtained using a method similar to the one discussed in Section III-B. For this structure, the new value for TSKIP should be used, and hence, α becomes (TAOI+TOAI) / (2×TCARRY). In particular, the following steps should be taken.

1)The size of the RCA block of the first stage is one.

2) From the second stage to the nucleus stage, the size of jth stage is determined based on the delay of the product of the sum of its RCA block and the delay of the carry output of the (j − 1)th stage. Hence, based on the description given in Section III-B, the size of the RCA block of the jth stage should be as large as possible, while the delay of the product of the its output sum should be smaller than the delay of the carry output of the (j − 1)th stage. Therefore, in this case, the sizes of the stages are either not changed or increased.

3) The increase in the size is continued until the summation of all the sizes up to this stage becomes larger than N/2. The last stage, which has the largest size, is considered as the nucleus (pth) stage. There are cases that we should consider the stage right before this stage as the nucleus stage (Step 5).

4) Starting from the stage (p + 1) to the last stage, the sizes of the stage i is determined based on the delay of the incrementation block of the ith and (i − 1)th stages (TINC,i and TINC,i−1, respectively), and the delay of the skip logic. In particular

$$TINC,i \leq TINC,i-1 - TSKIP,i-1;\ \text{for}\ i \geq p + 1. \quad (13)$$

In this case, the size of the last stage is one, and its RCA block contains a HA.

5) Finally, note that, it is possible that the sum of all the stage sizes does not become equal to N. In the case, where the sum is smaller than N by d bits, we should add another stage with the size of d. The stage is placed close to the stage with the same size. In the case, where the sum is larger than N by d bits, the size of the stages should be revised (Step 3). For more details on how to revise the stage sizes.

Now, the procedure for determining the stage sizes is demonstrated for the 32-bit adder. It includes both the conventional and the proposed CI-CSKA structures. The number of stages and the corresponding size for each stage, which are given in Fig. 4, have been determined based on a 45-nm static CMOS technology. The dashed and dotted lines in the plot indicate the rates of size increase and decrease. While the increase and decrease rates in the conventional structure are balanced, the decrease rate is more than the

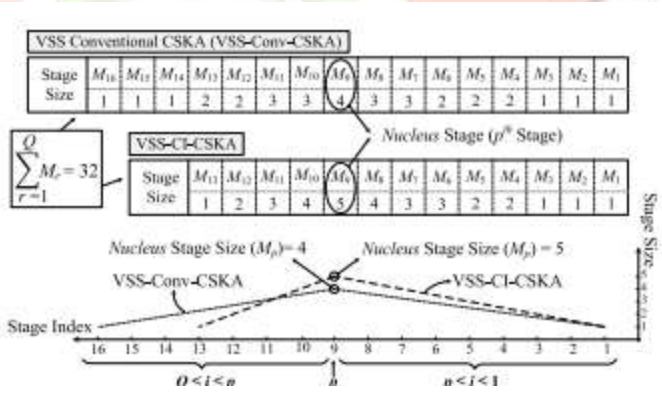

Fig. 3. Sizes of the stages in the case of VSS for the proposed and conventional 32-bit CSKA structures in 45-nm static CMOS technology.

increase one in the case of the proposed structure. It originates from the fact that, in the Conv-CSKA structure, both of the stages size increase and decrease are determined based on the RCA block delay [according to (4) and (5)], while in the proposed CI-CSKA structure, the increase is determined based on the RCA block delay and the decrease is determined based on the incrementation block delay. The imbalanced rates may yield a larger nucleus stage and smaller number of stages leading to a smaller propagation delay

V.PROPOSED HYBRID VARIABLE LATENCY CSKA WITH MAC

In this section, first, the structure of a generic variable latency adder, which may be used with the voltage scaling relying on adaptive clock stretching, is described. Then, a hybrid variable latency CSKA structure based on the CI-CSKA structure with MAC described.

A.Variable Latency Adders Relying on Adaptive Clock Stretching

The basic idea behind variable latency adders is that the critical paths of the adders are activated rarely. Hence, the supply voltage may be scaled down without decreasing the clock frequency. If the critical paths are not activated, one clock period is enough for completing the operation. In the cases, where the critical paths are activated, the structure allows two clock periods for finishing the operation. Hence, in this structure, the slack between the longest off-critical paths and the longest critical paths determines the maximum amount of the supply voltage scaling. Therefore, in the variable latency adders, for determining the critical paths activation, a predictor block, which works based on the inputs pattern.

The concepts of the variable latency adders, adaptive clock stretching, and also supply voltage scaling in an N-bit RCA adder may be explained using Fig. 5. The predictor block consists of some XOR and AND gates that determines the product of the propagate signals of considered bit positions. Since the block has some area and power overheads, only few middle bits are used to predict the activation of the critical paths at price of prediction accuracy decrease. In Fig. 5, the input bits (j + 1)th–(j + m)th have been

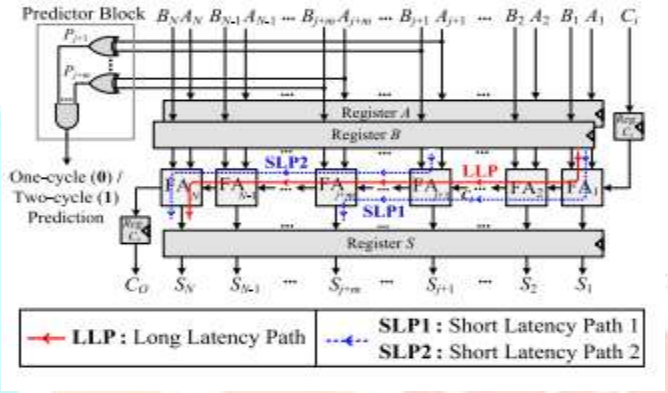

Fig. 5, the input bits (j + 1)th–(j + m)th

exploited to predict the propagation of the carry output of the jth stage (FA) to the carry output of (j + m)th stage. For this configuration, the carry propagation path from the first stage to the Nth stage is the longest critical path (which is denoted by Long Latency Path (LLP), while the carry propagation path from first stage to the (j+m)th stage and the carry propagation path from (j +1)th stage to the Nth stage (which are denoted by Short Latency Path (SLP1) and SLP2, respectively) are the longest off-critical paths. It should be noted the paths that the predictor shows are (are not) active for a given set of inputs are considered as critical (off-critical) paths. Having the bits in the middle decreases the maximum of the off-critical paths. The range of voltage scaling is determined by the slack time, which is defined by the delay difference between LLP and max(SLP1, SLP2). Since the activation probability of the critical paths is low

B.Proposed Hybrid Variable Latency CSKA Structure

The basic idea behind using VSS CSKA structures was based on almost balancing the delays of paths such that the delay of the critical path is minimized compared with that of the FSS structure. This deprives us from having the opportunity of using the slack time for the supply voltage scaling. To provide the variable latency feature for the VSS CSKA structure, we replace some of the middle stages in our proposed structure with a PPA modified in this paper. It should be noted that since the Conv-CSKA structure has a lower speed than that of the proposed one, in this section, we do not consider the conventional structure. The proposed hybrid variable latency CSKA structure is shown in Fig. 6 where an Mp-bit modified PPA is used for the pth stage (nucleus stage). Since the nucleus stage, which has the largest size (and delay) among the stages, is present in both SLP1 and SLP2, replacing it by the PPA reduces the delay of the longest

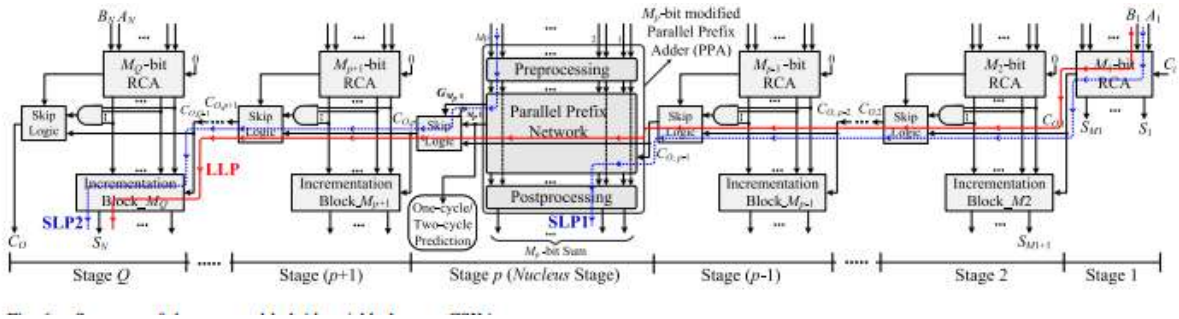

Fig. 6. Structure of the proposed hybrid variable latency CSKA.

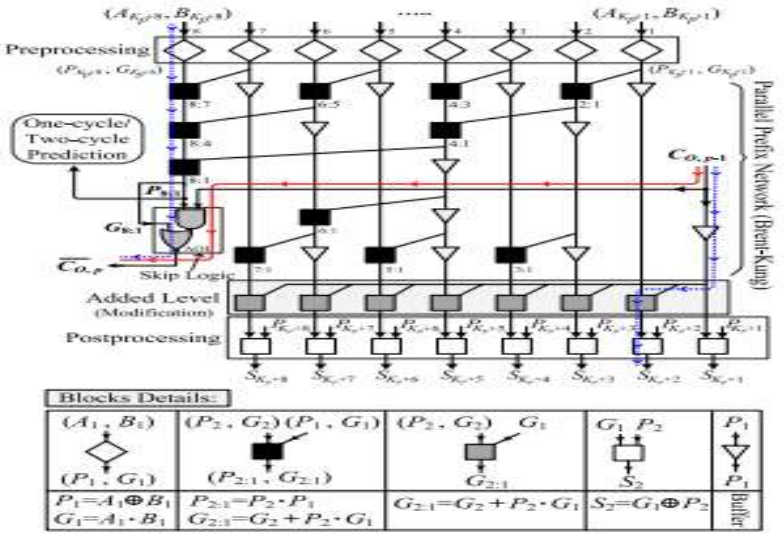

Fig. 7. Internal structure of the pth stage of the proposed hybrid variable latency CSKA. Mp is equal to 8 and Kp=p−1 r=1 Mr .

off-critical paths. Thus, the use of the fast PPA helps increasing the available slack time in the variable latency structure. It should be mentioned that since the input bits of the PPA block are used in the predictor block, this block becomes parts of both SLP1 and SLP2.

In the proposed hybrid structure, the prefix network of the Brent–Kung adder is used for constructing the nucleus stage (Fig. 7). One the advantages of the this adder compared with other prefix adders is that in this structure, using forward paths, the longest carry is calculated sooner compared with the intermediate carries, which are computed by backward paths. In addition, the fan-out of adder is less than other parallel adders, while the length of its wiring is smaller. Finally, it has a simple and regular layout. The internal structure of the stage p, including the modified PPA and skip logic, is shown in Fig. 7. Note that, for this figure, the size of the PPA is assumed to be 8 (i.e., Mp = 8).

VI. RESULTS:

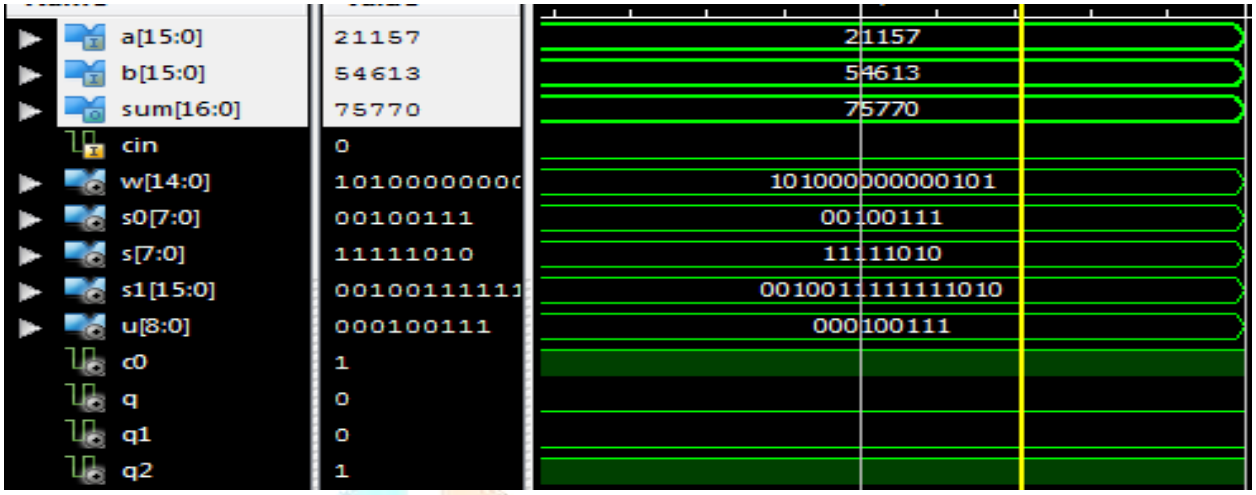

Fig8: simulation Results for proposed carry skip adder

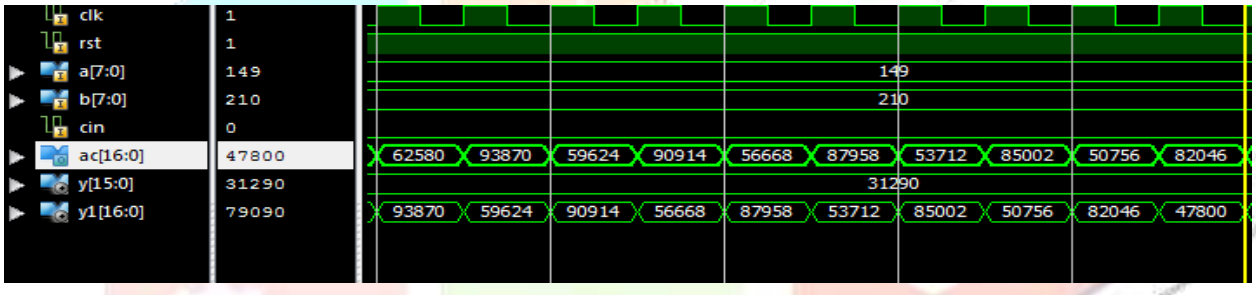

Fig9: Simulation Results for MAC by using proposed carry skip adder

VII.CONCLUSION

In this paper, A design of high performance16-bit Multiplier-and-Accumulator (MAC) is implemented in this project and a static CMOS CSKA structure called CI-CSKA was proposed with MAC, which exhibits a higher speed and lower energy consumption compared with those of the conventional one. The speed enhancement was achieved by modifying the structure through the concatenation and incrementation techniques. .MAC unit performs important operation in many of the digital signal processing (DSP) applications. The multiplier is designed using array multiplier and the adder is done with carry skip adder. In addition, AOI and OAI compound gates were exploited for the carry skip logics. The efficiency of the proposed structure for both FSS and VSS was studied by comparing its power and The total design is coded with verilog-HDL and the synthesis is done using Cadence RTL complier using typical libraries of Xilinx 13.2 Version . It exploited a modified parallel adder structure at the middle stage for increasing the slack time, which provided us with the opportunity for lowering the energy consumption by reducing the supply voltage. The efficacy of this structure was compared versus those of the variable latency RCA, C2SLA, and hybrid C2SLA structures. Again, the suggested structure showed the lowest delay and PDP making itself as a better candidate for high-speed low-energy applications.

REFERENCES

[1] I. Koren, Computer Arithmetic Algorithms, 2nd ed. Natick, MA, USA: A K Peters, Ltd., 2002.
[2] R. Zlatanovici, S. Kao, and B. Nikolic, "Energy–delay optimization of 64-bit carry-lookahead adders with a 240 ps 90 nm CMOS design example," IEEE J. Solid-State Circuits, vol. 44, no. 2, pp. 569–583, Feb. 2009.
[3] S. K. Mathew, M. A. Anders, B. Bloechel, T. Nguyen, R. K. Krishnamurthy, and S. Borkar, "A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS," IEEE J. Solid-State Circuits, vol. 40, no. 1, pp. 44–51, Jan. 2005.
[4] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy, "Comparison of high-performance VLSI adders in the energy-delay space," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 6, pp. 754–758, Jun. 2005.
[5] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 2, pp. 371–375, Feb. 2012.
[6] M. Vratonjic, B. R. Zeydel, and V. G. Oklobdzija, "Low- and ultra low-power arithmetic units: Design and comparison," in Proc. IEEE Int. Conf. Comput. Design, VLSI Comput. Process. (ICCD), Oct. 2005, pp. 249–252.
[7] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 10, pp. 689–702, Oct. 1996.
[8] Y. He and C.-H. Chang, "A power-delay efficient hybrid carrylookahead/carry-select based redundant binary to two's complement converter," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 1, pp. 336–346, Feb. 2008.
[9] C.-H. Chang, J. Gu, and M. Zhang, "A review of 0.18 µm full adder performances for tree structured arithmetic circuits," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 6, pp. 686–695, Jun. 2005.
[10] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," Proc. IEEE, vol. 98, no. 2, pp. 237–252, Feb. 2010.
[11] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," Proc. IEEE, vol. 98, no. 2, pp. 253–266, Feb. 2010.
[12] S. Jain et al., "A 280 mV-to-1.2 V wide-operating-range IA-32 processor in 32 nm CMOS," in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC), Feb. 2012, pp. 66–68.
[13] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. dissertation, Dept. Inf. Technol. Elect. Eng., Swiss Federal Inst. Technol. (ETH), Zürich, Switzerland, 1998.

Author's Details:

Mr.G. SIVA KUMAR, He received B.Tech degree in ECE dept from JNTUH. Presently He is pursuing M.TECH in BRANCH of VLSI&ES at SCIENT INSTITUTE OF TECHNOLOGY, IBRAHIMPATNAM

Mr. P Venkateswarlu, He received B.Tech From JNTU Kakinada in Electronics and Communication Engineering. He did M.Tech from Andhra University in Communication System. He is working as Assistant Professor in SCIENT Institute of technology.His area of research in Microstrip Patch antenna and Data encoding techniques, venkat.p22@gmail.com.

Dr. Arvind Kundu, he did B. Tech from H.P. University (SHIMLA) in Electronics & Communication. He did M. Tech from M.D. University (ROHTAK) in Electronics & Communication Engineering. He did Ph. D from Ranchi University and area of research is ADHOC Networks, EMBEDDED System, Cryptography, Message authentication Protocol, Image Processing, Routing protocol etc. He is working as HOD ECE Department at SCIENT INSTITUTE OF TECHNOLOGY, IBRAHIMPATNAM, dr.arvindkundu@gmail.com.