INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2022, VOL. 68, NO. 2, PP. 307-313
Manuscript received June 21, 2021; revised May, 2022. DOI: 10.24425/ijet.2022.139883

An Efficient Two-phase Clocked Sequential Multiply -Accumulator Unit for Image Blurring

Rashmi Samanth, and Subramanya G. Nayak

***Abstract*—The multiply-accumulator (MAC) unit is the basic integral computational block in every digital image and digital signal processor. As the demand grows, it is essential to design these units in an efficient manner to build a successful processor. By considering this into account, a power-efficient, high-speed MAC unit is presented in this paper. The proposed MAC unit is a combination of a two-phase clocked modified sequential multiplier and a carry-save adder (CSA) followed by an accumulator register. A novel two-phase clocked modified sequential multiplier is introduced in the multiplication stage to reduce the power and computation time. For image blurring, these multiplier and adder blocks are subsequently incorporated into the MAC unit. The experimental results demonstrated that the proposed design reduced the power consumption by 52% and improved the computation time by 4% than the conventional architectures. The developed MAC unit is implemented using 180*nm* standard CMOS technology using CADENCE RTL compiler, synthesized using XILINX ISE and the image blurring effect is analyzed using MATLAB.**

***Keywords*—Multiply-accumulator (MAC) unit; modified sequential multiplier; finite state machine (FSM); two-phase clockin; carry-save adder (CSA); image blurring**

I. INTRODUCTION

Multiply-accumulator units are the essential computational blocks that perform complex operations in the majority of the digital signal and image processors. Apart from their widespread use in processors, they are also used in a variety of other applications.

A typical MAC consists of a multiplier, adder, and accumulator register. The multipliers are the slowest component in terms of computing, requires a large area in the hardware resources, and consumes more power in the architecture. There have been numerous studies conducted on developing different types of multipliers to meet the requirements of reducing power, area consumption, and speeding up the computation. Most of the time, researchers were able to accomplish a reduction in power with less area, but at the expense of processing speed, and vice versa. The traditional method of multiplication follows add- shift method, which requires less area with certain power limitations. In [1], a novel energy-efficient approximate MAC architecture is developed with an overhead of area for error-resilient, image processing applications. The presented idea was to replace the simple shift operation. Instead of the conventional shift method, an input- aware conditional approximate multipliers were used for power reduction. To speed up the multiplication, a novel iterative self-timed clocking scheme is introduced [2] on a modified booth recoding algorithm. The developed architecture was performed based on self- timed independent form; the time wasted on the clock was reduced to a minimum with a small area. To improve the computational speed, authors of [3] developed a technique for high-speed parallel booth encoded multipliers. By using parallel multipliers, this architecture achieved an 8% reduction in delay. The parallel multiplication method enhances the computation speed with compromission of hardware utilization and power consumption. The authors of [4] have proposed pipelining technique to reduce the power and hardware utilization along with fast multiplication. Radix-based modified booth multipliers with 3:2 compressor adders were introduced in the architecture. The preceding multiplication strategies particularly targeted to enhance the speed of the architecture using combinational logic. The booth multiplier algorithm can be modified while encoding bit patterns. Those input bit patterns must be scanned and checked before encoding. The combinational logic will work depending on the current values of the multiplier inputs and it does not have to be activated by a clock. Therefore, it is completely time independent. There is a motive to construct a flexible architecture, which includes digital logic that decides automatically with high-speed processors. Sequential logic must be considered in the design to achieve this. The benefit of this logic is time-dependent, and it requires triggering.

The FSM-based booth design was developed [5] in the multiplication stage. The state transition controlled by a clock if the multiplier bits were shifted to a two-bit window, which leads to a change in the bit pattern. In each iteration of the booth algorithm, the arithmetic operation was evaluated whenever a new bit shifted in. Precomputation-based sequential multipliers [6] were developed using priority encoders to reduce the switching activities. It showed 30% clock count with improved switching activity as compared to existing multiplication designs. Similarly, radix-4 based sequential circuit is implemented for booth multiplication [7] to impose the state transitions. Even though sequential multipliers were intended to save computing time, the resulted outcome was a considerable delay reduction in the circuit. To improve the speed and accuracy of these sequential designs, two-phase clocking scheme was developed from a single clock signal [8 -10]. The general-purpose field impulse response (FIR) filter was developed using the two-phase clocking technique [11]. In this research, demonstrations were made in transistor-level simulations to compare two-phase [12] clocks with single-phase

Authors are with Department of Electronics and Communication Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal-576 104, India (e-mail: rashmi.samanth@gmail.com, gs.nayak@manipal.edu).

clocks. As a result, the two-phase clock sequential design generated faster outputs.

The main contribution of this work is to implement an efficient sequential circuit with a modified booth multiplier by taking speed limits from the literature into account. The novel design uses a two-phase clocking approach to minimize the multiplier architecture's computation time and power. The developed multiplier and adder blocks are integrated on the MAC unit for image blurring applications.

II. Structure of the Proposed MAC

Multiply-accumulate is a typical computation method for calculating the product of two numbers in a sequence. The product is then added to an accumulator. In this work, the proposed MAC is constructed in three parts: a sequential modified booth encoder with two-phase clocking, a carry-save adder, and an accumulator register.

A. *Developed Modified Sequential Multiplier*

An 8-bit multiplication architecture based on a modified booth encoder is implemented and explained in this section. Register A and B are used to store the multiplicand and the multiplier bits respectively. The traditional modified booth encoder is developed priorly as a fundamental block that reduces the partial product generation. Initially, the LSB bit is added with '0' (right to left) of the multiplier then grouped into three input bits. After this, multiplicand will operate based on the encoding scheme that is mentioned in Table I.

Table I
Modified Booth Recoder

$i+1$	i	$i-1$	Operation
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

The multiplier input bits are scanned to group into 3 bits with LSB introduced with 0. To generate the partial product, a set of operations needs to be performed on the multiplicand that is represented as 0, 1, 2, -2 and -1. If the bits of the multiplier are grouped as 000 and 111, then 0 is multiplied by the multiplicand. Similarly, the grouped input bits grouped as 001 and 010, the generated partial product will be added to the preceding output of the multiplicand that is +1. The two's complemented version of the multiplicand is assigned for the result for the 101 and 110 bits. For 011 and 100 inputs the result will be shifted to the left by 1 bit (+2) of the multiplicand and for the next case, the multiplicand is assigned as (-2) which is complemented and shifted left by one bit. The result will be a summation of all the partial products.

A Moore finite state machine (FSM) is designed to perform booth multiplication as shown in Fig. 1. Hence the power required is less and the speed is high when compared to all other existing techniques. In each iteration, the add, sub and shift operation is performed, and it is decided by the memory bits shifted from left to the present state. The state decides the output and the next state depends on the current state and input.

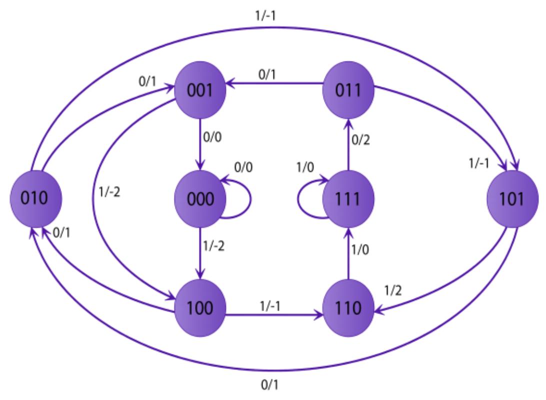

Fig. 1. State transition diagram of modified sequential multiplier

Table II represents the state of input (i/p), output (o/p), current state (CST) and next state (NST) of the FSM. The detailed algorithm steps show the working of the modified sequential multiplication.

Table II
Modified Sequential Multiplier State Table

i/p	CST	NST	O/p
0	000	000	0
0	001	000	0
0	010	001	+1
0	011	001	+1
0	100	010	-2
0	101	010	-2
0	110	011	-1
0	111	011	-1
1	000	100	-2
1	001	100	-2
1	010	101	-1
1	011	101	-1
1	100	110	-1
1	101	110	-1
1	110	111	0
1	111	111	0

The control logic in the booth algorithm is as follows:

Step 1: Append a '0' to the multiplier's LSB.
Step 2: Starting with the LSB, group three adjacent bits to form the current state register.
Step 3: The memory is used to store the next two bits, which are then provided as input to the FSM.
Step 4: Based on memory, determine the next state.
Step 5: The next state output is set as select lines of the booth encoder.
Step 6: Multiplicand operands are changed (shift/add/sub) depending on the encoding technique.
Step 7: The reduced partial products are combined to obtain the final product.

B. Developed Two-phase clocking technique

With a single clock, MBE's FSM operates as described in the preceding section. However, proposing a two-phase clock scheme in the architecture was the key work that was done to make the design fast. The toggling of a single clock's 50% duty cycle yields two non-overlapping clocks named ϕ_1 and ϕ_2. To obtain these clock pulses, the clocks were sent to a JK flip-flop, as indicated in Fig. 2. To accomplish toggling mode, the JK inputs are set to logic 1.

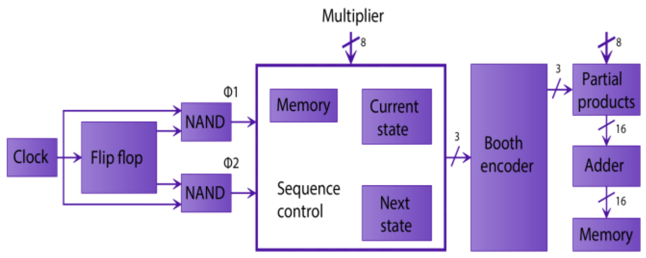

Fig. 2. Block diagram of the developed multiplier

The flip-flop and clock outputs are fed as input to the NAND gates to produce two-phase toggling clocks. The sequence control is controlled by these clocks ϕ_1 and ϕ_2. A state transition occurs whenever the clocks ϕ_1 or ϕ_2 reach a positive edge. We introduced an architecture to compute fast multiplication by utilizing a modified sequential multiplier with two-phase clocking in the current work.

The three stages designed in the proposed method are explained as follows:

Stage 1: Generation of two-phase clocks

With the help of a JK flip-flop, a single clock of 50 MHz is used to produce two-phase clocks of 25 MHz each at the first stage, as illustrated in Fig. 3.

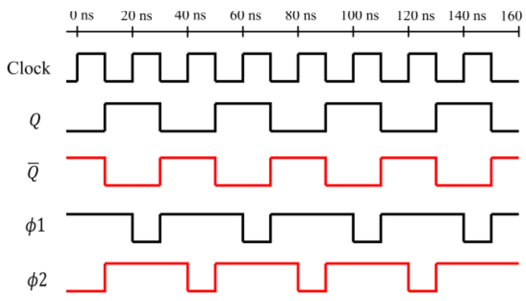

Fig. 3. Generation of two-phase clocks

The outputs Q and $\bar{Q}$ of JK are connected to the NAND gate. As the JK enters the toggle mode, Q switches the logic at every negative edge of the clock. The flip-flop and clock outputs are sent as independent inputs to the NAND gates. The output of the NAND gate gives ϕ_1 and makes a '0' logic when Q and clocks are at '1'. Similarly, whenever $\bar{Q}$ and clock becomes '1', the NAND reaches logic '0', thereby producing ϕ_2.

Stage 2: FSM Modules

In this stage, a modified booth algorithm is implemented in the form of the finite state machine. The multiplier grouped to three bits along with appended '0' is passed to the state machine. This stage is synchronized with the generated two-phase clocks. The FSM module includes a memory of two bits to retain the grouped bits of the multiplier's current state which helps in deciding the next state. The state transitions occur at every subsequent rising edge of the two-phase clocks.

Stage 3: Generation of partial products

The state determined at the previous module is the input to this stage. In this stage, the partial products are generated based on the booth conditions for the occurred input state. The conditions of the encoders are applied to the multiplicand bits. Finally, the partial products are added using carry save adders to obtain the product.

C. Carry Save Adder

In the partial product stage, the modified booth multiplier design employs carry-save adders. Three operand additions can be done at the same time by using CSA [13]. To calculate the partial product terms in an 8-bit architecture, four blocks are required as illustrated in Fig. 3. These logic blocks are built as a combination of an encoder and a multiplexer that provide the multiplicand bits in accordance with the three-bit condition of multiplier. Three adders are used to obtain the final product, one of which is a ripple carry adder and remaining two are carry-save adders. The use of CSA blocks allows for simultaneous three-term addition. In the ripple carry adder stage, separately generated sum and carry terms are combined.

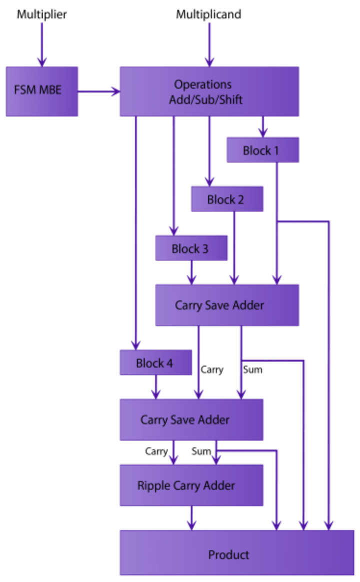

Fig. 4. CSA blocks in Proposed MAC

III. PROPOSED MAC FOR IMAGE BLURRING

The main function of the MAC unit is to multiply, add, shift, and store. Fig. 5 depicts the proposed MAC architecture. The following typical equation represents the main function the proposed MAC for 2D convolution scheme:

$$Z(x,y) = \sum_{m=0}^{K-1}\sum_{n=0}^{K-1}(x+m, y+n)\,.\,h(m,n) + Y(x,y) \qquad (1)$$

Where $I(x,y)$ stands for input pixels, $h(m,n)$ stands for kernels and window lengths $m,n = 0\!:\!K-1$, image width $x = 0\!:\!M-1$, $y = 0\!:\!N-1$, and $Y(x,y)$ stands for accumulator registers. $Z(x,y)$ generates the filtered image in a first in, first out (FIFO) manner. Image pixels are stored in D-flipflop registers. In FIFO, $M - k$ shift registers are employed. This module's main target is to shift the window plane $h(m, n)$ over the image $I(x, y)$. At each clock cycle, the convolver applies K^2 MAC units to the image. For image pixels and kernels, 8-bit signed integers are used as operands.

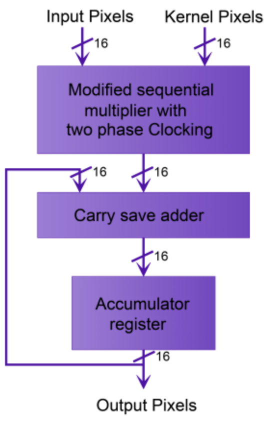

Fig. 5. Proposed MAC unit

A. Simulation steps

MATLAB:

1. An image comprising 8-bit signed integers is convolved with a desired filter kernel with an 8-bit fixed point or 8-bit integer. The Gaussian kernel is used in the current study.
2. To retain the background pixels while performing the convolution, the image matrices are padded with zeros.
3. The kernel and image matrices are transformed to text format as vectors h_i and I_j, with $i = 0$ equaling $K - 1$ and $j = 0$ equaling $MN - 1$ respectively. The image vectors are in hexadecimal format.

Xilinx ISE:

1. The convolution test bench module imports the hex file containing image pixels from MATLAB into memory locations.
2. To convolve the image with the specified kernels, MAC units are invoked from the test bench.
3. The convolved results are written to a text file in hexadecimal format and kept in a separate memory block.

IV. Result Analysis

The proposed MAC unit was implemented using 180nm standard CMOS technology to compare the power, delay. The simulation was carried out using XILINX ISE. For evaluating the image blurring effect MATLAB is used. This section explains the findings of the developed MAC performance metrics.

A. Sequential modified multiplier results

The proposed sequential multiplier design simulation and test bench waveforms with and without two-phase clock phasing were examined using XILINX ISE. The Fig. 6 shows the multiplication output of the inputs 27 and 53 with a regular clock. The testbench status is changed depend on the single clock. The multiplication cycle took roughly $800ns$ to complete.

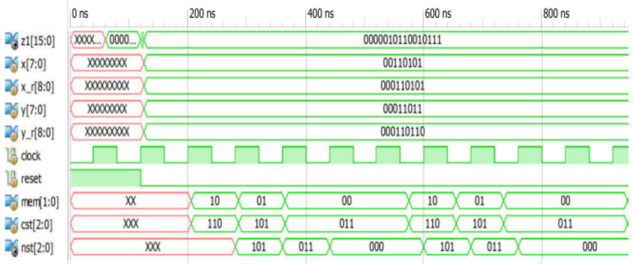

Fig. 6. A clock-synchronized modified sequential multiplier

Fig. 7 shows another example of multiplication, this time with the operand's multiplier and multiplicand. The multiplier is scanned for three bits with a '0' attached to the LSB. The first three bits of the operand '126' are set to 100, which is the current state.

```
                0 1 1 1 1 1 1 0 | 126  (Mxr)
              x 1 1 0 1 0 1 1 1 | 215  (Mcnd)
0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 | pp 1 : -2
0 0 0 0 0 0 0 0 0 0 0 0 0 0     | pp 2 :  0
0 0 0 0 0 0 0 0 0 0 0 0         | pp 3 :  0
0 1 1 0 1 0 1 1 1 0             | pp 4 : +2
0 1 1 0 1 0 1 1 1 1 0 1 0 0 1 0 | 27090 (Product)
```

Fig. 7. Example of modified sequential multiplication

Booth encoding is used to condition the partials. Because the state was '100,' the first partial product is two's compliment of the multiplicand bits in this case. Similarly, partial products are generated for all states until the grouped multiplier bits are completed. It must be noticed that the partial products have been reduced to just four subparts. To obtain the final product, the decreased partial products are added using CSA.

Fig. 8 illustrates the results of multiplication with two-phase clocking of the same bits with inputs 27 and 53. We can observe from the test bench waveform that the transition of states is dependent on both clocks ϕ_1 and ϕ_2. As a result, it is possible to compute faster multiplication with change of states.

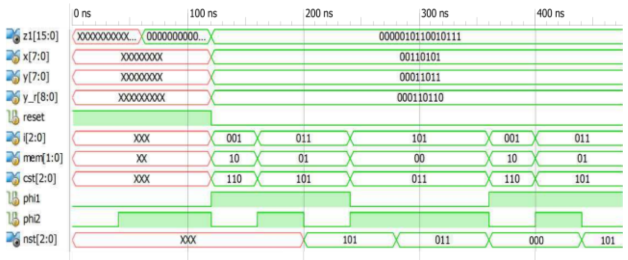

Fig. 8. Example of modified sequential multiplication

TABLE III
MODIFIED BOOTH RECODER

Integer i	Current state	Memory	Next state	Operation
1	100	11	111	0
3	111	11	111	0
5	111	01	011	+2

Fig. 9 shows the modified sequential multiplier with and without two-phase clocking for the multiplier '126.' In this example, a time scale of 0 to $160ns$ is used. The first 'Clock' corresponds to the traditional approach [14] in which the multiplier changes state with each rising edge of the clock. The multiplier's three bits grouping states took the entire cycle to complete.

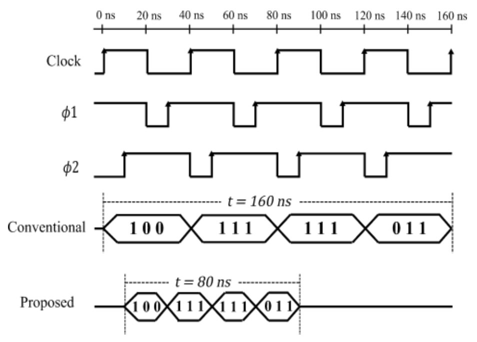

Fig. 9. Comparison of developed sequential multiplier with and without two-phase clocking

Depending on both the rising edges of the two-phase clocks ϕ_1 and ϕ_2, the suggested technique transits the state. The waveforms show that the proposed method follows both clocks, the timing of the process completion was minimized by 50% compared to the existing method.

A. Performance comparison of the proposed MAC

In Table IV and V shows the comparison of proposed MAC results with the conventional MAC. It demonstrates that the developed structure is both power efficient and fast. The design of a sequential multiplier is primarily responsible for this improvement.

B. Results of Proposed MAC for image blurring

The PSNR for developed MAC unit is evaluated using gaussian blurring. A normal distribution is used to construct the Gaussian kernel $h(x, y)$, where x and y denote the kernel's pixel coordinates.

$$h(x,y) = \frac{1}{\sigma^2 2\pi}\, e^{-\frac{(x^2+y^2)}{2\sigma^2}} \qquad (2)$$

Table VI shows the kernels used were Gaussian filter with varying σ measure from 0.5 to 2.

TABLE VII
POWER COMPARISON OF THE PROPOSED MAC WITH DIFFERENT MULTIPLIER

MAC structure with conventional multiplier	Power(mW)
[15]	9.11
[17]	3.74
[18]	2.74
Proposed MAC with sequential multiplier	1.29

$$PSNR = 10\log_{10}\frac{N^2}{MSE} \qquad (3)$$

where N is the maximum pixel value of the image. Since an example of 8-bit integer image was used in this study, the measure is calculated using the following equation:

TABLE VII
RESULTS OF 2D GAUSSIAN FILTER

σ	$PSNR_M$ (dB)	$PSNR_V$ (dB)	$E_M - E_V$ (dB)
0.5	23.94	24.04	0.1
1	25.09	25.14	0.05
1.5	25.27	25.30	0.03
2	25.32	25.35	0.03

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[I_o(x,y) - I_R(x,y)]^2 \qquad (4)$$

where I_o represents the original image and I_R represents the filtered image.

TABLE VI
KERNEL MATRICES REPRESENTATION FOR DIFFERENT σ VALUES

σ	$h(x,y)$
0.5	$\begin{bmatrix}0.011 & 0.083 & 0.011\\ 0.083 & 0.619 & 0.083\\ 0.011 & 0.083 & 0.011\end{bmatrix}$
1	$\begin{bmatrix}0.075 & 0.123 & 0.075\\ 0.123 & 0.204 & 0.123\\ 0.075 & 0.123 & 0.075\end{bmatrix}$
1.5	$\begin{bmatrix}0.094 & 0.118 & 0.094\\ 0.118 & 0.147 & 0.118\\ 0.094 & 0.118 & 0.094\end{bmatrix}$
2	$\begin{bmatrix}0.101 & 0.115 & 0.101\\ 0.115 & 0.130 & 0.115\\ 0.101 & 0.115 & 0.101\end{bmatrix}$

The PSNR in dB of processed images retrieved from the Verilog $PSNR_V$ and MATLAB $PSNR_M$ platforms is compared to the error values using the following expression:

$$PSNR_{error} = PSNR_M - PSNR_V \qquad (5)$$

The PSNR (dB) values were found to have a minimum error $E_M - E_V$ of 0.005 to 0.1 for varying σ of the kernels that is evaluated using MATLAB and Verilog platforms.

Table VII shows the minimum error measurement of the processed image.

Fig. 10 shows the comparison between the smoothened images computed in Xilinx and validated in MATLAB. Four subplots in Fig. 10 [a] represent the original and [b] is the filtered images with different kernels (k=5,7 and 9) respectively.

The intensity values at location [38, 40] are extracted from both the outputs to illustrate the performance of the MAC unit. It was found that there was a minimum error of fixed-point intensities between MATLAB and Verilog ranging from 0.05 to 0.1.

The histograms are displayed and compared with the MATLAB findings to obtain the difference between the pixel values to validate the MAC findings, as demonstrated in Fig. 11 [a], [b] and [c].

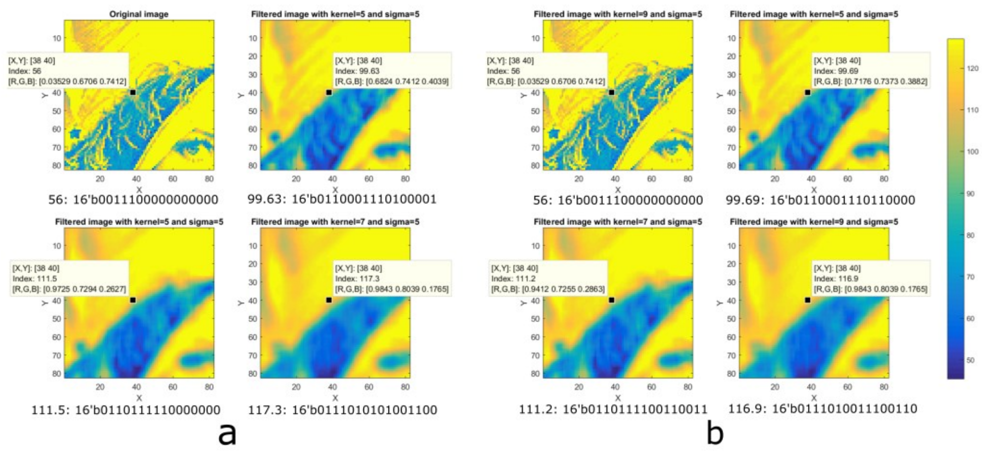

Fig. 10. a. Input image, b. Blurred image with σ = 5 in MATLAB, c. Blurred image with σ = 5 in Verilog

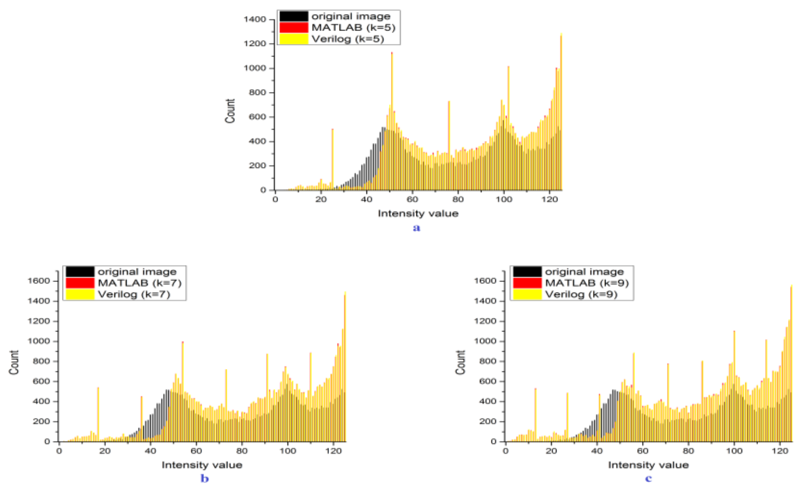

Fig. 11. Comparison in terms of histograms of filtered image a. kernel 5 x 5, b. kernel 7 x 7, c. kernel 9 x 9

V. Conclusion

The current work proposes an efficient two-phase clocked sequential multiply -accumulator unit with the benefit of parallelism to considerably reduce computing power as well as time. Over manual clock tuning, a two-phase clocking system is used to reduce delay time. The findings of the suggested design reduced delay as well as power consumption compared to the standard architecture. The experimental results showed that the new design decrease the power consumption *by* 52% with increase in computation time by 4% than the conventional architectures. The attained speed is attributable to state transitions that happened on both the positive and negative edges of the two-clock phases.

References

[1] M. Masadeh, O. Hasan and S. Tahar, “Input-conscious approximate multiply-accumulate (mac) unit for energy-efficiency,” IEEE Access, 7, pp.147129-147142. https://doi.org/10.1109/ACCESS.2019.2946513

[2] M.C. Shin, S.H. Kang, and I.C Park, “An area-efficient iterative modified-booth multiplier based on self-timed clocking,” In Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001 Sep 23: IEEE. pp. 511-512. https://doi.org/10.1109/ICCD.2001.955079

[3] W.C. Yeh and C.W Jen, “High-speed Booth encoded parallel multiplier design,” IEEE transactions on computers, 49(7):692-701, 2000. https://doi.org/10.1109/12.863039

N. Kaur and R.K Patial, “Implementation of Modified Booth Multiplier using Pipeline Technique on FPGA,” International Journal of Computer Applications. 975:8887, 2013. http://dx.doi.org/10.5120/11666-7261

[4] S. Shrivastva and Pankaj Gulhane, “Comparative Analysis on Power and Delay Optimization of Various Multipliers using VHDL,” International Journal of Electrical and Electronics Research, 2014 2(3):182-91.

[5] N. Honarmand, M.R. Javaheri, N. Sedaghati-Mokhtari and A. Afzali-Kusha, “Power efficient sequential multiplication using pre-computation,” In 2006 IEEE International Symposium on Circuits and Systems. pp. 4. https://doi.org/10.1109/ISCAS.2006.1693183

[6] K. Babulu and G. Parasuram, “FPGA Realization of Radix-4 Booth Multiplication Algorithm for High-Speed Arithmetic Logics,” International Journal of Computer Science and Information Technologies. 2(5), 2102-7, 2011.

[7] R. Das, G. K Singh and R.M. Mehra. “Two-phase clocking scheme for low-power and high-speed VLSI,” Int. J. Adv. Eng. Sci. Technol, 2(2):225-30, 2013.

[8] N. A. Nayan, Y. Takahashi and T. Sekine, “LSI implementation of a low-power 4× 4-bit array two-phase clocked adiabatic static CMOS logic multiplier,” Microelectronics Journal, 43(4):244-9,2012. https://doi.org/10.1016/j.mejo.2011.12.013

[9] K. Kajstura and D. Kania, “Low power synthesis of finite state machines-state assignment decomposition algorithm,” Journal of Circuits, Systems and Computers,27(03):1850041, Mar 7, 2018. https://doi.org/10.1142/S021812661850041X

[10] F. Carbognani, F. Bürgin, N. Felber, H. Kaeslin and W. Fichtner, “Two-phase clocking and a new latch design for low-power portable applications,” In International Workshop on Power and Timing Modeling, Optimization and Simulation. Springer, Berlin, Heidelberg. 2005 Sep 20, pp. 446-455. https://doi.org/10.1007/11556930_46

[11] K. Kato, Y. Takahashi, and T. Sekine, “Two phase clocking subthreshold adiabatic logic,” In 2014 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE 2014 Jun 1 pp. 598-601. http://dx.doi.org/10.1109/ISCAS.2014.6865206

[12] S. Erniyazov and J.C Jeon, “Carry save adder and carry look ahead adder using inverter chain based coplanar QCA full adder for low energy dissipation,” Microelectronic Engineering. 211:37-43, Apr 15, 2019. https://doi.org/10.1016/j.mee.2019.03.015

[13] Y. Zhang Y, L. Okamura, and T. Yoshihara, “An energy efficiency 4-bit multiplier with two-phase non-overlap clock driven charge recovery logic,” IEICE transactions on electronics, 94(4):605-12, Apr 1, 2011. https://doi.org/10.1587/transele.E94.C.605

[14] K.C. Kuo and Chou CW, “Low power and high-speed multiplier design with row bypassing and parallel architecture,”. Microelectronics Journal,41(10):639-50, Oct 1, 2010. https://doi.org/10.1142/S021812661750030X

[15] M.S. Kumar, D.A Kumar and P. Samundiswary, “Design and performance analysis of Multiply-Accumulate (MAC) unit” In International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], 2014 Mar 20 (pp. 1084-1089). IEEE. https://doi.org/10.1109/ICCPCT.2014.7054782

[16] M. Madheswaran and S. Saravanan, “Design and Analysis of Low Power Multiply and Accumulate Unit Using Pixel Properties Reusability Technique for Image Processing Systems,” ICTACT Journal on Image and Video Processing, 459-66, Aug. 2012. https://doi.org/10.21917/ijivp.2012.0065

[17] P.I. Khan and R.S. Mishra, “Comparative analysis of different algorithm for design of high-speed multiplier accumulator unit (MAC),” Indian Journal of science and technology, 9(8), Feb. 2016. https://doi.org/10.17485/ijst/2016/v9i8/83614

[18] K. Singh and D. Kumar, “Modified booth multiplier with carry select adder using 3-stage pipelining technique,” International Journal of Computer Applications, 44(14):35-8, Apr. 2012. https://doi.org/10.5120/6334-8710

[19] Y.N Rao, G.S. Raju, P. Raja, “Design and performance evaluation of high-speed MAC unit with parallel pipeline technology,” International Journal of Computer Applications,1;106(4), Jan. 2014 https://doi.org/10.5120/18506-9575