International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Sep-2022)

https://dx.doi.org/10.12785/ijcds/120164

Design of Serial Multiplier Circuit Based on a Variable Length Conditional Binary Counter for Improved Critical Path Delay and Slack Time

Mangal Deep Gupta [1] and R. K. Chauhan [2]

[1, 2] Department of Electronics and Communication Engineering, MMMUT, Gorakhpur, Uttar Pradesh, India

Received 21 Mar. 2021, Revised 01 Apr. 2022, Accepted 05 Apr. 2022, Published 30 Sep. 2022

Abstract: In this paper, a variable-length conditional counter algorithm is proposed to count or add the partial products of serial multiplier circuits for minimizing computational delay. The proposed variable-length conditional counter enables optimizing critical path delay and improving its slack time at different input frequencies. The evaluation of the slack time of any circuit is essential to estimate the highest frequency of operation. In this work, the slack time of the proposed architecture is computed and compared with the related work. The result shows that slack time is higher as compared to the existing circuit, which implies that the proposed architecture of the serial multiplier operates at high frequency. The proposed variable-length conditional counter design and its serial multiplier architecture of 8, 16, 32 and 64-bit has been designed using Verilog HDL. The simulation of the proposed design has been done on the mentor Graphics EDA Tool. The physical layout design of the proposed 8-bit multiplier circuit using 180-nm CMOS technology is obtained in this work.

Keywords: Variable-Length Conditional Binary Counter, Critical Path Delay, Slack Time, Multiplier, Semi-Custom Layout, Verilog HDL

1. Introduction

The multiplier is the core component of any digital signal processing (DSP) unit, encryption and decryption algorithm in cryptography, data rendering in medical imaging, and other logical computations [1]- [4]. Performance of various operations such as arithmetic and logic unit in microprocessors [5], In DSP applications [6]-[7], various operations like convolution, Fast Fourier Transform (FFT), filtering, intensive arithmetic functions [8], and other operations are mostly dependent on multiplier design architecture. For higher throughput arithmetic operations, it is necessary to achieve the desired performance. In arithmetic operations, binary multiplication is one of the most important key components. Therefore, the development of low hardware and high-speed multiplier architecture has become a subject of interest over the decades. Low chip area, less power consumption and high speed are the basic requirement for the design of multiplier circuits. Since multiplication dominates the execution time of most DSP processors, cryptography algorithms, ALU operations, etc. the performance of these systems mainly depends on the speed of the multiplier circuit. So, a high-speed multiplier circuit with the optimum required area and power dissipation is needed in VLSI systems.

The minimization of timing performance, chip area and power dissipation of binary multiplier circuits is a major challenge for researchers. To estimate the frequency constraints, Slack time analysis is also done by design engineers. Its consideration is important because one needs to understand how fast the chip is going to run and interact with the other chips. Slack time is generally defined as the difference between the desired arrival time of a signal and its actual arrival time. It indicates the violation of timing constraints in VLSI circuits. A violation may occur if the slack time is less than the threshold value i.e. specified by the user. A timing constraint is violated when its value is negative. In the electronic design automation (EDA) tool, static time analysis (STA) is used to determine the designed circuit

E-mail: mangaldeepgcet@gmail.com, rkchauhan27@gmail.com

will reliably operate or not at a specified frequency constraint [9]- [10].

Binary multiplication of two integers gives partial products which are added for producing final results. The addition of that partial product dominates the critical path delay or speed of serial multiplier architecture. To combine the partial product efficiently, a column compressor is generally used. Various techniques have been proposed in earlier literature to minimize the performance of partial product addition, such as Wallace tree [11]- [12], Dadda tree multiplier [13]. These techniques use full adders (as a binary counter) to reduce each 3-bits (with the same weight column's element) into 2-bits of different weight and these bits are further added using a carry-save adder tree. But these techniques do not optimize carry propagation delay.

A 7:3 parallel counter built from full and half adder was presented by Mehta et al to minimize carry propagation delay [14] of [12]- [13]. It accepts 7-bit of the same weight column elements that give a 3-bit output of different weight column's elements. These counter are then designed as an efficient counter-based Wallace tree multiplier. The delay observed in such design was due to the chain of the XOR gate on the critical path. In [15]- [16], presented a booth multiplier architecture with significantly improved delay. It uses a multi-stage parallel addition rather than sequential addition for the addition of partial products. It is also used carry-save adder with carry look-ahead logic for adder implementation and bubble pushing techniques are used for booth encoder optimization. A 7:3 multi-bit counter to count the partial products of binary multiplication was proposed by *Aloke Saha et. al.* [17]. This design includes the half-adders rather than full-adders. The input operands are divided into multiple groups (each of 2-bit) that count using a half-adder circuit. *Fritz and Fam* in [18] were presented a counting method that uses the stacking circuit to stack the number of ones in given binary data and these stack data are thereafter counted using this binary counter, producing a 7:3 binary counter. Also, they proposed a multiplier circuit.

To optimize the carry propagation delay of partial products in multiplier architecture, a variable-length conditional counter to count or add the partial product is proposed to improve its critical path delay and slack time with the different input frequencies. To demonstrate the proposed variable-length conditional counter, a multiplier circuit of different sizes has been designed using the proposed binary counter and stacking-based counter for critical path delay and slack time analysis at the different input frequencies. The rest of the paper is organized as follows: Section-2, the concept of column converter is summarized. The design of the variable-length conditional counter is discussed in Section-3. The proposed algorithm for binary multiplication is described in Section-4. The discussion on simulated results is mentioned in Section-5. Finally, Section-6 concludes the work.

2. Column Converter

In a column converter, partial product in binary multiplication operations is arranged in separate columns. In the proposed architecture of a multiplier circuit, a column converter is one of the primary steps, in which partial products are converted into a separate column that has the same weight element in the form of 0's and 1's. Vedic mathematics [19] is used to generates columns of partial products using this technique. This column converter requires less time as compared to serial operation. In conventional multiplication operation these phenomena are executed using serial left shifting of multiplicand then logic AND with multiplier data and MSB's bit in every shifting operation, which is a more time-consuming operation compared to column conversion of partial products.

An example is taken here to show how different columns with elements corresponding to the partial product of 8-bit data give the 16 different columns with variable elements size in symmetric order.

$c_0 = A_0\&B_0;$

$c_1 = \{A_1\&B_0 , A_0\&B_1\};$

$c_2 = \{A_2\&B_0 , A_1\&B_1, A_0\&B_2, y1_1\};$

$c_3 = \{A_3\&B_0 , A_0\&B_3, A_2\&B_1, A_1\&B_2, y2_1\};$

$c_4 = \{A_4\&B_0 , A_0\&B_4, A_2\&B_2, A_3\&B_1, A_1\&B_3, y2_2, y3_1\};$

$c_5 = \{A_5\&B_0 , A_0\&B_5, A_4\&B_1, A_1\&B_4, A_3\&B_2, A_2\&B_3, y3_2, y4_1\};$

$c_6 = \{A_6\&B_0 , A_0\&B_6, A_5\&B_1, A_1\&B_5, A_4\&B_2, A_2\&B_4, A_3\&B_3, y4_2, y5_1\};$

$c_7 = \{A_7\&B_0 , A_0\&B_7, A_1\&B_6, A_6\&B_1, A_5\&B_2, A_2\&B_5, A_3\&B_4, A_4\&B_3, y4_3, y5_2, y6_1\};$

$c_8 = \{A_7\&B_1 , A_1\&B_7, A_2\&B_6, A_6\&B_2, A_3\&B_5, A_5\&B_3, A_4\&B_4, y5_3, y6_2, y7_1\};$

$c_9 = \{A_7\&B_2 , A_2\&B_7, A_3\&B_6, A_6\&B_3, A_4\&B_5, A_5\&B_4, y6_3, y7_2, y8_1\};$

$c_{10} = \{A_7\&B_3 , A_3\&B_7, A_4\&B_6, A_6\&B_4, A_5\&B_5, y7_3, y8_2, y9_1\};$

$c_{11} = \{A_7\&B_4 , A_4\&B_7, A_5\&B_6, A_6\&B_5, y8_3, y9_2, y10_1\};$

$c_{12} = \{A_7\&B_5 , A_5\&B_7, A_6\&B_6, y9_3, y10_2, y11_1\};$

$c_{13} = \{A_7\&B_6 , A_6\&B_7, y10_3, y11_2, y12_1\};$

$c_{14} = \{A_7\&B_7 , y12_2, y13_1\};$

$c_{15} = \{y13_2 , y14_1\};$

3. Design of Variable Length Conditional Counter

To minimize the carry propagation delay in multiplier architecture, this section develops the variable-length conditional counter that counts the logic 1's in each column of partial products. The counting operation of this counter depends on the column's element. In the partial product of multiplier circuits, each column requires a variable length of the element, so one needs a variable-length conditional counter. Using this counter, one can design a multiplier architecture that is efficient in terms of carry propagation delay.

Algorithm 1 shows the step for the proposed variable-length conditional counter in which 'X' is the n-bit register that store the 'n' elements of binary input, Y is $(m+1)$ bit register that is used to store the count value, variable 'i' is $\log_2 n$ bit register which is used to select the input bit in register 'X'. The output of this counter is $(C_m, \dots, C_1, S)$. This algorithm is designed by hardware description language (HDL) in which conditional assignment is used, if X[i] a bit of register 'X' is zero or one, depend on that increment in 'Y' and 'i' have been executed. This operation is executing 'n' times in the loop with the clock signal. When register (i) value equal to input bit size (n), the value of the register (Y) assign to the output variable $(C_m, \dots, C_1, S)$.

Algorithm-1: Stage for proposed general variable-length conditional counter

Initialize: $i \leftarrow 0, Y \leftarrow 0, X \leftarrow$ input data, $n \leftarrow$ number of elements in the X register
while $(i \le n)$
do
if $(C[i] = 0)$
$Y \leftarrow Y,\ \ i \leftarrow i+1$
else
$Y \leftarrow Y+1, i \leftarrow i+1$
end
$\{C_m, \dots, C_1, S\} \leftarrow Y$

To design the gate level architecture for the above algorithm, this section proposes a sequential circuit that counts or add a binary sequence based on the clock signal. This architecture containing T-flip flops holds the value of the variable register (Y) and the number of flip flops is depended on the size of this variable. The output of flip flops are control by the value of a variable (i) and it is assigned to the output variable $(C_m, \dots, C_1, S)$. The flip-flops input is control by the input binary sequence and relevant logic gates which are described in Eq. (1) to Eq. (9). Assume output of flip flops to be $Q_n, Q_{n-1}, \dots . Q_1, Q_0$ and corresponding inputs as $T_n, T_{n-1}, \dots . T_1, T_0$.

Generalized input combinational logic at each flip-flop is given in Eq. (1) to Eq. (5) and the count value of this counter is given in Eq. (6) to Eq. (9) for designing the general variable-length conditional counter. The count value will be reflected by the specific value of a variable (i) and its value will depend on the size of the binary counter. Eq. (6) represents the LSB of the count value and the rest of the bits of the count value are given by Eq. (7) to Eq. (9).

For designing the 7:3 binary counter, extract the logic expression from the generalized equation, which is given in Eq. (10) to Eq. (15). It requires three flip flops to design this architecture and each flip flop's input is controlled by the relevant combinational logic, which is given in the equation. Fig. 1 presents the proposed 7:3 binary counter. Similarly, Fig. 2 presents the proposed 5:3 binary counter. It is similar to a 7:3 binary counter but controlling the output signal is different from that. The output is controlled by $I[2].\bar{I}[1].I[0]$, as shown in Fig. 2.

$$T_n = In.\bar{Q}_n Q_{n-1} \dots Q_1.Q_0 \tag{1}$$

$$T_{n-1} = In.\bar{Q}_n Q_{n-2} \dots Q_0.Q_0 + In.\bar{Q}_{n-1}.Q_{n-2} \dots Q_1.Q_0 \tag{2}$$

$$T_{n-2} = In.\bar{Q}_n.Q_{n-3} \dots Q_1.Q_0 + In.\bar{Q}_{n-1}.Q_{n-3} \dots Q_1.Q_0 + In.\bar{Q}_{n-2}.Q_{n-3} \dots Q_1.Q_0 \tag{3}$$

.

.

$$T_1 = In.\bar{Q}_n.Q_1.Q_0 + In.\bar{Q}_{n-1}.Q_1.Q_0 + \cdots + In.\bar{Q}_3.Q_1.Q_0 + In.\bar{Q}_2.Q_1.Q_0 \tag{4}$$

$$T_0 = In.\bar{Q}_n + In.\bar{Q}_{n-1} + In.\bar{Q}_{n-2} + \cdots + In.\bar{Q}_1 + In.\bar{Q}_0 \tag{5}$$

$$S = Q_n.I[n-1].I[n-2] \dots .I[1].I[0] \tag{6}$$

$$C_n = Q_{n-1}.I[n-1].I[n-2] \dots .I[1].I[0] \tag{7}$$

$$C_{n-1} = Q_{n-2}.I[n-1].I[n-2] \dots .I[1].E[0] \tag{8}$$

.

.

$$C_1 = Q_0.I[n-1].I[n-2] \dots .I[1].I[0] \tag{9}$$

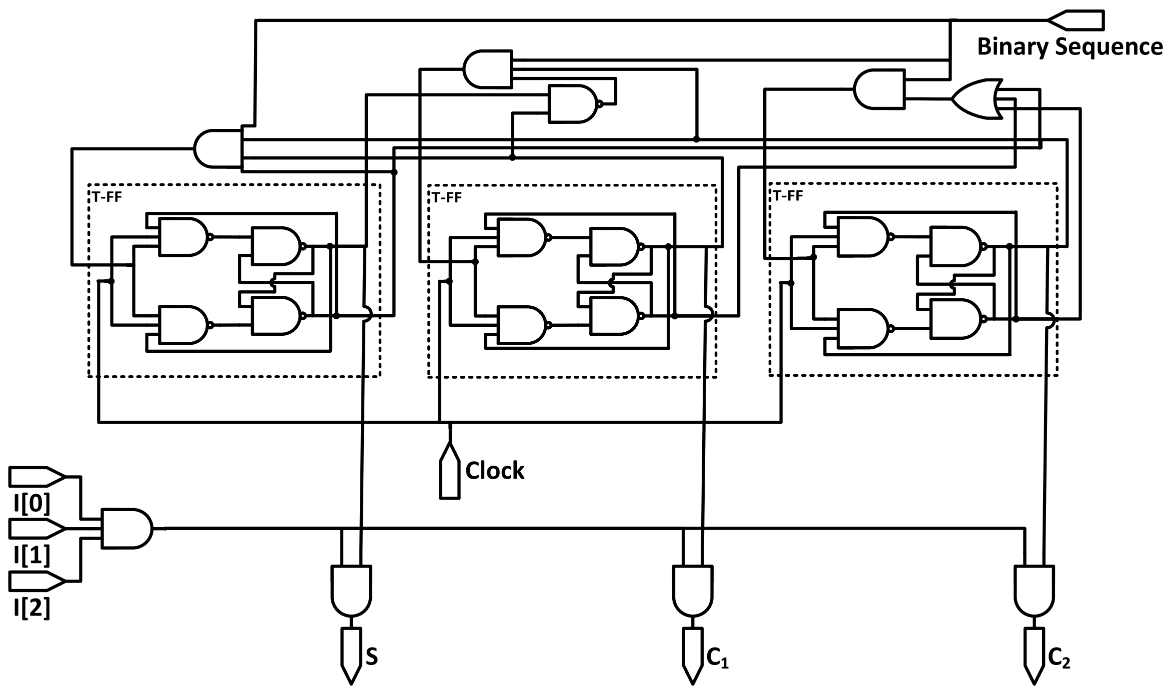

Figure 1. Gate level Schematic of 7:3 binary conditional counter using Proposed Algorithm.

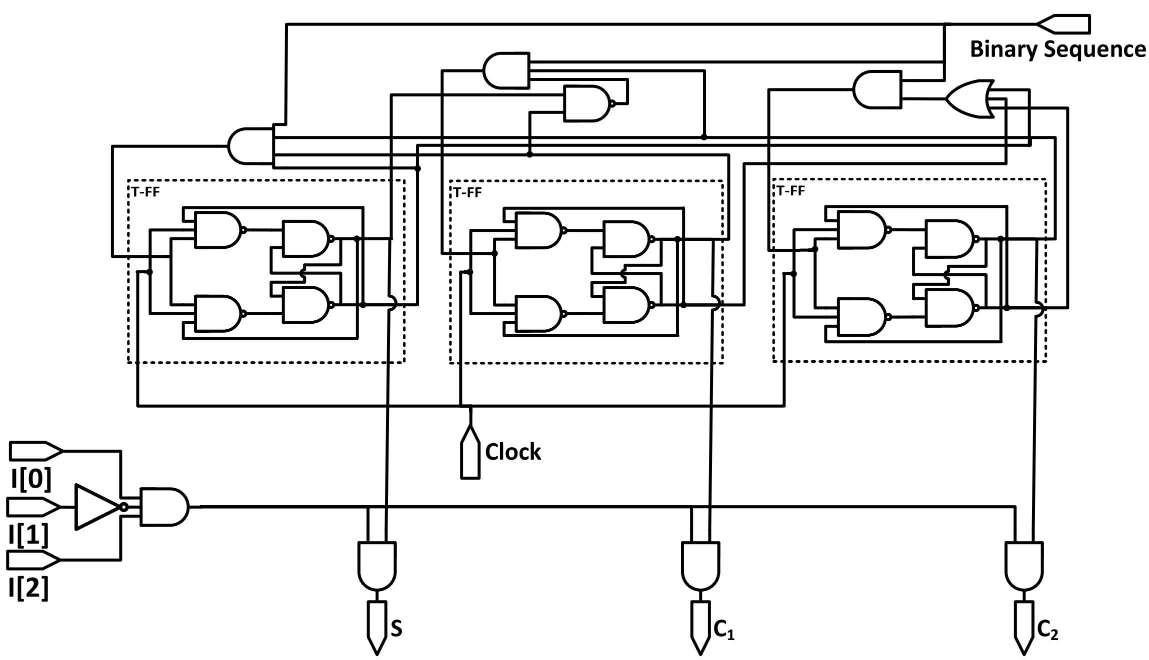

Figure 2. Gate level Schematic of 5:3 binary conditional counter using Proposed Algorithm.

For the 7:3 binary counter,

$$T_2 = In.\bar{Q}_2.Q_1.Q_0 \tag{10}$$

$$T_1 = In.\bar{Q}_2.Q_0 + In.\bar{Q}_1.Q_0 \tag{11}$$

$$T_0 = In.\bar{Q}_2 + In.\bar{Q}_1 + In.\bar{Q}_0 \tag{12}$$

$$S = Q_2.I[2].I[1].I[0] \tag{13}$$

$$C_2 = Q_1.I[2].I[1].I[0] \tag{14}$$

$$C_1 = Q_0.I[2].I[1].I[0] \tag{15}$$

4. Proposed Algorithm for Binary Multiplication

The proposed algorithm is realized with the concept of the column converter of partial product and variable-length conditional counter. The first step of this algorithm to generate the columns of partial product using Vedic mathematics through input A and B such that every column element group together after column conversion. After that select each column from C_0 to C_{2P-1} one by one, count the number of ones in a selected column in each iteration using the proposed conditional counter based on the column's elements. In algorithm $C_r[i]$ is the i^{th} an element of selected column in r^{th} iteration, if its value is logic one, count register Y is an increment to 1 i.e. $Y \leftarrow Y + 1$ otherwise count register will preserve their previous value i.e. $Y \leftarrow Y$ and 'i' will be an increment to 1 in both cases i.e. $i \leftarrow i + 1$. This process will be continuing until 'i' is less than or equal to 'n' $(i \leq n)$ whenever 'i' is equal to 'n', the count register value Y moves into the register D and LSB bit of register 'D' in r^{th} iteration, move into the output register bit M[r] where M is 2P-bit register which store the multiplication result and $'r'$ is the bit position of register M and simultaneously rest of $(log_2 n - 1)$ bits of the register(D), move into the next columns, and group together with the same weight elements. These operations have been done in each iteration and for the complete operation of this algorithm, these iterations will be repeats 2P times where P is the bit size of multiplicand and multiplier.

Fig.3 shows the data paths and control signals of the proposed multiplier architecture. Control and data path signal i.e. Sel column, Condition, Multiplicand (A), Multiplier (B), Output, column's element, clock, etc. have their individual control on multiplication operation. Column converter module gives the columns $(C_n \ldots . C_1, C_0)$ with different weight elements of a partial product of Multiplicand (A) and Multiplier (B). For counting operations 'Sel Column' signal is used to choose the column from the rightmost column and the condition signal is used to find the column element is zero or one, based on that counter register (Y) is increment or preserve to their previous value. When's complete the counting operation, MSB bit (Y_0) of count value $(Y_n \ldots Y_1 Y_0)$ moves into output register at corresponding position and rest of bits $(Y_n \ldots Y_1)$ merge into the next column with the same weight elements.

Algorithm-2: Stage for proposed binary multiplier using variable-length conditional counter

Initialize: $A, B \leftarrow$ **P**-bit input data
for $r = 0$ to $2P - 1$
Initialize: $i \leftarrow 0, Y \leftarrow 0, n \leftarrow$number of elements in r^{th} column
 while $(i \leq n)$
 do
 if $(C_r[i] = 0)$
 $Y \leftarrow Y,\ i \leftarrow i + 1$
 else
 $Y \leftarrow Y + 1,\ i \leftarrow i + 1$
 ends
 $D \leftarrow Y$
 $R[r] \leftarrow D[0]$
 $\{C_{2P-1}, \ldots, C_r\} \leftarrow D[\log_2 n : 1]$
end

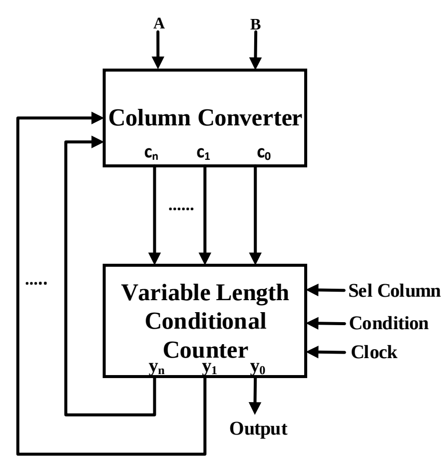

Figure 3. Data path and control signal of proposed conditional counter-based Multiplier

A dot representation of binary data operation, for 8-bit input data flow, is shown in Fig. 4. In this representation, the first two rows are indicating the multiplicand and multiplier and their partial products are arranged in separate columns $(C_{15}, C_{14}, \ldots, C_1, C_0)$ and each column has the same weight with a different number of elements which are represented by dots(.). At every iteration count, the number of ones in the entire column and counter register value is denoted by a symbol $(*_n)$ where 'n' denotes the counting operation for n^{th} column. LSB of '$*$' is moved into the output register and the rest of the bit combines with the next column which is shown in Fig. 4.

								.	.	.	.	.	.	.	.
								.	.	.	.	.	.	.	.
C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
$*_{13}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$*_{14}$	$*_{12}$	.	.	.	.	.	.	.	.	.	.	.	.	.	
	$*_{13}$	$*_{10}$	.	.	.	.	.	.	.	.	.	.	.		
		$*_{11}$	$*_9$	.	.	.	.	.	.	.	.	.	$*_1$		
		$*_{12}$	$*_{10}$	$*_8$	.	.	.	.	.	.	.	$*_2$			
			$*_{11}$	$*_9$	$*_7$	.	.	.	.	.	$*_2$				
				$*_{10}$	$*_8$	$*_6$	.	.	.	$*_3$	$*_3$				
					$*_9$	$*_7$	$*_5$	.	$*_4$	$*_4$					
						$*_8$	$*_6$	$*_5$	$*_5$						
							$*_7$	$*_6$							
$*_{15}$	$*_{14}$	$*_{13}$	$*_{12}$	$*_{11}$	$*_{10}$	$*_9$	$*_8$	$*_7$	$*_6$	$*_5$	$*_4$	$*_3$	$*_2$	$*_1$	$*_0$

Figure 4. Dot representation of a proposed algorithm for 8-bit data.

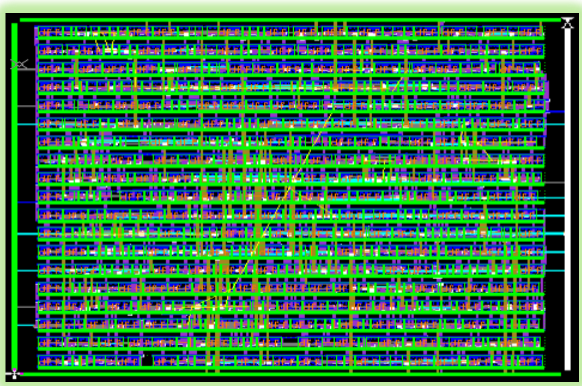

Figure 5. The semi-custom layout of 8-bit Multiplier of proposed architecture using 180-nm CMOS technology.

5. RESULT AND DISCUSSION

The proposed variable-length conditional counter and its serial multiplier architecture have been designed using Verilog HDL and behavior level simulation has been done using Questa Simulator by Mentor Graphics. The Leonardo Spectrum Tool has been used for the synthesis of the proposed design. For this simulation, 350-nm, 180-nm and 45nm CMOS technology are used. For validation of this work, a Stacking-based binary counter and its multiplier design [18] were implemented using the same process file. The simulation of the 7:3 binary counter using the proposed variable-length conditional counter and stacking-based counter have been done to analyze and compare its critical path delay and slack time at the different input frequencies. To demonstrate the proposed variable-length Conditional counter, a Multiplier circuit of different sizes has been designed using the proposed binary counter and stacking-based counter for critical path delay and slack time analysis at a different frequency. The semi-custom layout of the 8-bit proposed multiplier Architecture using 180-nm CMOS technology is shown in Fig. 5.

Table 1 shows the simulation result of the proposed 7:3 binary counter and stacking-based counter in terms of critical path delay and slack time with a variation of the input frequency. Based on the result, the critical path delay of the proposed counter and stacking-based counter are obtained as 1.11 ns and 1.65 ns (for 350-nm CMOS technology) respectively. It shows that the proposed counting operation is faster than a stacking-based counter. This is because the concept of parallelization has been used in our counter design. Data required time and slack time decrease with input frequency constraint. It can be seen in Table-1 that, at the 590-MHz frequency, the slack time of the stack-based counter is near to zero, which means it is near to critical working frequency constraint however, at the same operating frequency, the slack time of the proposed counter is 0.58 ns, which indicates that the design meets the timing constraint and still it can be improved for higher frequency signal. The proposed design operates successfully till 800-MHz input frequency. Fig. 6 shows the graphical representation of the slack time variation with the operating frequency of the 7:3 binary counter.

Table 2 shows the simulation result of the 8-bit proposed multiplier and stacking-based multiplier in terms of critical path delay and slack time at the different input frequencies. Based on simulation results, the critical path delay of the proposed variable-length conditional counter and stack counter-based 8-bit multiplier is obtained as 2.14ns and 6.82ns (for 350-nm CMOS technology) respectively. This shows that the proposed multiplier is faster than the stacking counter-based multiplier. It is due to the use of a variable-length conditional counter which can count or add the partial product with minimum carry propagation delay without using a carry-save adder. It also minimizes the computational time of the multiplication. Data required time and slack time decrease with input frequency, as seen in Table-2. At 145-MHz frequency, the slack time of the stack-based counter closes to zero, which means, it reaches critical working frequency, but at the same frequency, the slack time of the proposed multiplier is 4.67ns. It indicates that the proposed design meets the timing constraint and it can further be improved to 465 MHz frequency. The slack time observed for the stacked counter-based multiplier at 465 MHz is negative. So it concludes that the proposed multiplier architecture operates at a higher input frequency compared to the conventional multiplier. Slack time at different frequency constraints of both multipliers is shown in Fig. 7.

Table 3 presents the simulation result of 8, 16, 32, and 64-bit proposed and stacking-based multipliers design in terms of critical path delay and slack time at 20MHz input frequency. As mention in this table, the critical path delay of proposed variable-length conditional counter-based multipliers is less as compared to stacking counter-based multipliers [18]. Slack time of proposed architecture is higher than conventional architecture for all sizes of the multiplier (i.e. 8, 16, 32, and 64-bit). This table demonstrates that the proposed work can be operated at a higher frequency constraint. The graphical representation of results is displayed in Fig. 8.

Table 1. Synthesis report of 7:3 binary counter

Frequency constraint (MHz)	Date required time (ns)	350-nm CMOS				180-nm CMOS				45-nm CMOS			
		7:3 counter [18]		Proposed 7:3 counter		7:3 counter [18]		Proposed 7:3 counter		7:3 counter [18]		Proposed 7:3 counter	
		Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)
510	1.96	1.65	0.31	1.11	0.85	0.868	1.092	0.584	1.376	0.217	1.743	0.146	1.814
520	1.92	1.65	0.27	1.11	0.81	0.868	1.052	0.584	1.336	0.217	1.703	0.146	1.774
530	1.89	1.65	0.23	1.11	0.77	0.868	1.022	0.584	1.306	0.217	1.673	0.146	1.744
540	1.85	1.65	0.2	1.11	0.74	0.868	0.982	0.584	1.266	0.217	1.633	0.146	1.704
550	1.82	1.65	0.17	1.11	0.71	0.868	0.952	0.584	1.236	0.217	1.603	0.146	1.674
560	1.79	1.65	0.13	1.11	0.67	0.868	0.922	0.584	1.206	0.217	1.573	0.146	1.644
570	1.75	1.65	0.1	1.11	0.64	0.868	0.882	0.584	1.166	0.217	1.533	0.146	1.604
580	1.72	1.65	0.07	1.11	0.61	0.868	0.852	0.584	1.136	0.217	1.503	0.146	1.574
590	1.69	1.65	0.04	1.11	0.58	0.868	0.822	0.584	1.106	0.217	1.473	0.146	1.544
800	1.25	1.65	-0.40	1.11	0.27	0.868	0.382	0.584	0.666	0.217	1.033	0.146	1.104

Table 2. Synthesis report of 8-bit multiplier

Frequency constraint (MHz)	Date required time (ns)	350-nm CMOS				180-nm CMOS				45-nm CMOS			
		8-bit multiplier [18]		proposed 8-bit multiplier		8-bit multiplier [18]		proposed 8-bit multiplier		8-bit multiplier [18]		proposed 8-bit multiplier	
		Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)
50	20	6.82	13.18	2.14	17.86	3.589	16.411	1.126	18.874	0.897	19.10	0.281	19.719
60	16.67	6.82	9.85	2.14	14.53	3.589	13.081	1.126	15.544	0.897	15.77	0.281	16.389
70	14.29	6.82	7.47	2.14	12.14	3.589	10.701	1.126	13.164	0.897	13.39	0.281	14.009
80	12.50	6.82	5.68	2.14	10.36	3.589	8.911	1.126	11.374	0.897	11.60	0.281	12.219
90	11.11	6.82	4.29	2.14	8.97	3.589	7.521	1.126	9.984	0.897	10.21	0.281	10.829
100	10	6.82	3.18	2.14	7.86	3.589	6.411	1.126	8.874	0.897	9.103	0.281	9.719
110	9.09	6.82	2.27	2.14	6.95	3.589	5.501	1.126	7.964	0.897	8.193	0.281	8.809
120	8.33	6.82	1.51	2.14	6.19	3.589	4.741	1.126	7.204	0.897	7.433	0.281	8.049
130	7.69	6.82	0.87	2.14	5.55	3.589	4.101	1.126	6.564	0.897	6.793	0.281	7.409
140	7.14	6.82	0.32	2.14	5	3.589	3.551	1.126	6.014	0.897	6.243	0.281	6.859
145	6.9	6.82	0.08	2.14	4.76	3.589	3.311	1.126	5.774	0.897	6.003	0.281	6.619
465	2.15	6.82	-4.67	2.14	0.01	3.589	-1.439	1.126	1.024	0.897	1.253	0.281	1.869

Table 3. Binary multiplier synthesis report at 20 MHz

Size	Date required time (ns)	350-nm CMOS				180-nm CMOS				45-nm CMOS			
		Multiplier [18]		Proposed Multiplier		Multiplier [18]		Proposed Multiplier		Multiplier [18]		Proposed Multiplier	
		Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)	Date arrival time (ns)	Slack time (ns)
8-bit	40	6.82	33.18	2.14	37.86	3.589	36.410	1.126	38.873	0.897	39.102	0.281	39.718
16-bit	40	10.03	29.97	3.88	36.12	5.278	34.721	2.042	37.957	1.319	38.680	0.510	39.489
32-bit	40	18.81	21.19	13.03	26.97	9.9	30.1	6.857	33.142	2.475	37.525	1.714	38.285
64-bit	40	34.89	5.11	29.11	10.89	18.363	21.636	15.321	24.679	4.590	35.409	3.830	36.169

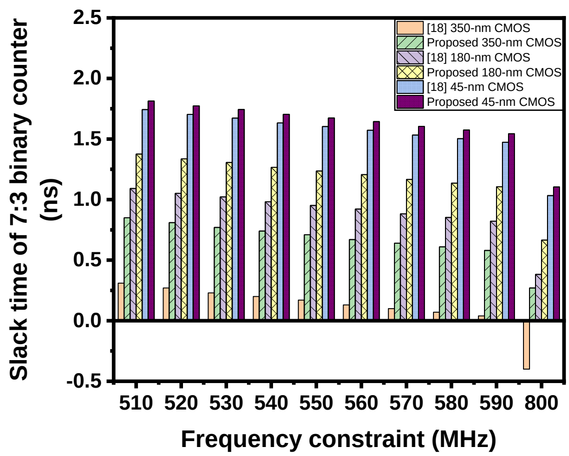

Figure 6. Comparison analysis of slack time for 7:3Binary stack counter [18] and proposed 7:3 conditional binary counter.

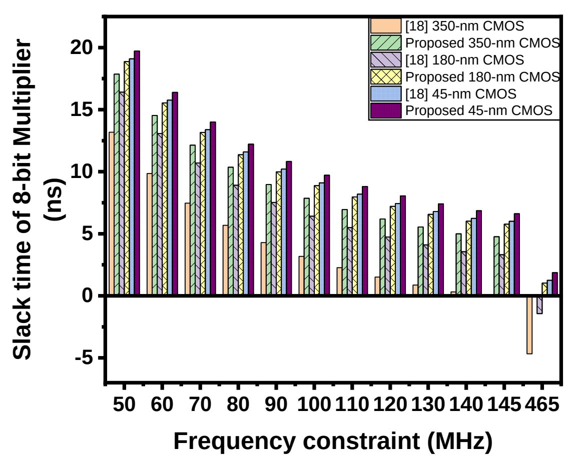

Figure 7. Comparison analysis of slack time for 8-bit multiplier using 7:3 binary stack counter [18] and proposed conditional binary counter.

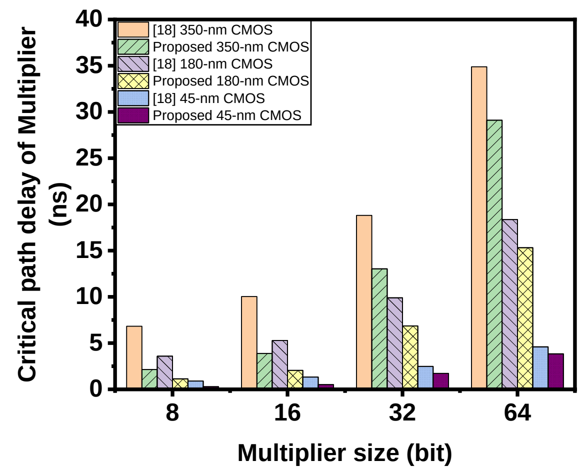

Figure 8. Comparison of analysis of critical path delay for multipliers.

6. Conclusion

This paper introduces a general variable-length conditional counter algorithm to count or add the partial products for the minimum computational delay. The results of the proposed design are compared and contrasted with the results obtained for the stacking-based [18] counter. The analysis has been done for critical path delay and slack time at different working frequencies. It demonstrates that the proposed counting operation is faster than a stacking-based counter. It was also suggested that the proposed counter can be operated at a high-frequency constraint compared to a conventional binary counter. Analysis has been done in this work to demonstrate the usage of the proposed variable-length conditional counter-based multiplier circuit at different operand sizes. Critical path delay and slack time were obtained from the proposed design as well as of the stack counter-based multiplier reported in the literature [18]. The physical layout for the 8-bit proposed serial multiplier circuit is also obtained in this work using 180-nm CMOS technology. This work demonstrates that the proposed variable-length conditional counter-based multiplier architecture can significantly improve the slack time by 14% and delay is reduced by 68% of multiplier circuit and it can successfully be operated at a higher frequency compared to the earlier reported designs. The simulation results show that the slack time of the multiplier circuit increase by 18% and critical path delay is reduced by 87% when 45 nm is compared with 350-nm CMOS technology. The performance of the proposed multiplier design was therefore superior compared to its contemporary design. If the dimensions of the device are further reduced from 45nm, it is expected that the proposed design will show better performance. The proposed binary multiplier can be used to design modern processors or digital systems for low-area and high-speed applications.

REFERENCES

[1] Gupta MD & Chauhan RK. Efficient Hardware Implementation of Pseudo-Random Bit Generator Using Dual-CLCG Method. *Journal of Circuits, Systems and Computers* (2021).

[2] A. T. Abebe, Y. N. Shiferaw, and P. G. V. S. Kumar, Compact and High Speed Point Multiplication Architecture for Elliptic Curve Diffie-Hellman Algorithm on FPGA, *Int. J. Comput. Digit. Syst.*, (2021).

[3] U. Iqbal and A. H. Mir, Efficient and dynamic access control mechanism for secure data acquisition in IoT environment, *Int. J. Comput. Digit. Syst.*, 10, 9–28 (2021).

[4] Gupta MD & Chauhan RK. Coupled variable-input LCG and clock divider-based large period pseudo-random bit generator on FPGA, *IET Computer and Digital Techniques*. 1–13 (2021). doi:10.1049/cdt2.12027

[5] Tang, G. M., Takagi, K. & Takagi, N. 32 × 32-Bit 4-Bit Bit-Slice Integer Multiplier for RSFQ Microprocessors. in *IEEE Transactions on Applied Superconductivity* (2017). doi:10.1109/TASC.2017.2662700

[6] Warrier, R., Shreejith, S., Zhang, W., Vun, C. H. & Fahmy, S. A. Fracturable DSP Block for Multi-Context Reconfigurable Architectures. *Circuits, Syst. Signal Process.* (2017). doi:10.1007/s00034-016-0445-x

[7] Meher, P. K. Seamless Pipelining of DSP Circuits. *Circuits, Syst. Signal Process.* **35**, 1147–1162 (2016).

[8] Kim, E. J., Lee, J. H. & Sunwoo, M. H. Novel shared multiplier scheduling scheme for area-efficient FFT/IFFT processors. *IEEE Trans. Very Large Scale Integr. Syst.* **23**, 1689–1699 (2015).

[9] Vygen, J. Slack in static timing analysis. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **25**, 1876–1885 (2006).

[10] Gupta, M.D. & Chauhan, R.K. Performance Investigation of Binary Counter with Different Clock Gating Networks. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 12(2), 59-67 (2020).

[11] Wallace, C. S. A Suggestion for a Fast Multiplier. *IEEE Trans. Electron. Comput.* **EC-13**, 14–17 (1964).

[12] Asif, S. & Kong, Y. Analysis of different architectures of counter based Wallace multipliers. in *Proceedings - 2015 10th International Conference on Computer Engineering and Systems, ICCES 2015* 139–144 (2016). doi:10.1109/ICCES.2015.7393034

[13] Blankenship, P. E. Comments on “a two’s complement parallel array multiplication algorithm”. in *Computer Arithmetic: Volume I* 159 (2015). doi:10.1142/9789814651578

[14] Mehta, M., Parmar, V. & Swartzlander, E. High-speed multiplier design using multi-input counter and compressor circuits. in *Proceedings - Symposium on Computer Arithmetic* 43–50 (1991).

[15] Boppana, N. V. V. K., Kommareddy, J. & Ren, S. Low-Cost and High-Performance 8 × 8 Booth Multiplier. *Circuits, Syst. Signal Process.* **38**, 4357–4368 (2019).

[16] Mohanty, B. K. & Choubey, A. Efficient Design for Radix-8 Booth Multiplier and Its Application in Lifting 2-D DWT. *Circuits, Syst. Signal Process.* **36**, 1129–1149 (2017).

[17] Saha, A., Pal, R., Naik, A. G. & Pal, D. Novel CMOS multi-bit counter for speed-power optimization in multiplier design. *AEU - Int. J. Electron. Commun.* **95**, 189–198 (2018).

[18] Fritz, C. & Fam, A. T. Fast Binary Counters Based on Symmetric Stacking. *IEEE Trans. Very Large Scale Integr. Syst.* **25**, 2971–2975 (2017).

[19] Bansal, Y., Madhu, C. & Kaur, P. High speed vedic multiplier designs-A review. in *2014 Recent Advances in Engineering and Computational Sciences, RAECS 2014* (2014). doi:10.1109/RAECS.2014.6799502

Rajeev Kumar Chauhan is currently a Professor & Head in the ECE Department, MMMUT, Gorakhpur-273010, U.P, INDIA, and serving the same institute for the last 26 years. He has received his Ph.D. in Electronics Engineering, from IIT-BHU, Varanasi, INDIA, in 2002, M.E. from MNNIT-Allahabad in 1993, and B.Tech from G.B.P.U.A.T – Pantnagar, in 1989. He has also served as a professor at Addis Ababa University, Addis Ababa under the UNDP program for two years (2003- 2005). His research area includes VLSI Technology & Design.
E-mail: rkchauhan27@gmail.com

Mangal Deep Gupta is a Teaching and Research Fellow, in the ECE Department, MMMUT, Gorakhpur-273010, U.P. INDIA. He has received his M.Tech. in VLSI Design, from C-DAC, Noida, INDIA, in 2018. His research area includes: Microelectronics & VLSI Design.
E-mail: mangaldeepgcet@gmail.com