Journal of AI and Data Mining
Vol 7, No 3, 2019, 385-392
DOI: 10.22044/JADM.2018.7177.1853

Fast Mux-based Adder with Low Delay and Low PDP

H. Tavakolaee, Gh. Ardeshir* and Y. Baleghi

Dept. of Electrical and Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran.

Received 01 July 2018; Revised 23 July 2018; Accepted 16 November 2018
**Corresponding author: g.ardeshir@nit.ac.ir (G. Ardeshir).*

Abstract
Adders, as one of the major components of digital computing systems, have a strong influence on their performance. There are various types of adders, each of which uses a different algorithm to do addition with a certain delay. In addition to low computational delay, minimizing power consumption is also a main priority in an adder circuit design. In this paper, the proposed adder is divided into several sub-blocks, and the circuit of each sub-block is designed based on multiplexers and NOR gates to calculate the output carry or input carry of the next sub-block. This method reduces the critical path delay, and therefore, increases the speed of the adder. Simulation and synthesis of the proposed adder is done for cases of 8, 16, 32, and 64 bits, and the results obtained are compared with those of the other fast adders. The synthesis results show that the proposed 16- and 32-bit adders have the lowest computation delay and also the best power delay product among all the recent popular adders.

Keywords: *Carry Look Ahead Adder Carry Select Adder, Fast Adder, Low Power Adder.*

1. Introduction

Addition is one of the main mathematical operations, which is widely used in very large scale integration (VLSI) systems such as microprocessors and digital signal processers [1]. Addition is also applied in some other operations such as subtraction, multiplication, and addressing [2]. A fast and accurate operation in digital systems mainly depends on the adders [3]. Therefore, improving the performance of a digital adder is necessary for binary operations inside such systems [4].

One of the main objectives of mobile communications and computing systems is to reduce the power of their VLSI circuits. On the other hand, most of these VLSI systems are battery-based, which suffer from many limitations to provide the required power [5]. Hence, energy efficiency is important in portable electronic systems such as mobile phones, laptops, satellites, spacecraft, and aircraft [6]. Furthermore, the demands for portable electronic devices with a low power and a longer-life battery are increasing [7]. Adders, as the main operators of an ALU block, must be optimized in two aspects, i.e. power and speed of computations, to improve the overall system performance. Power consumption has a direct relation with the hardware used in a circuit. Therefore, reducing the hardware used in an adder results in a lower power consumption [8].

Fast adders speed up the computations by performing multi-bit operations in parallel. There are different types of fast adders with specialized applications. The simplest adder is Ripple Carry (RC), which is made up of some full adders (FAs) connecting in series [9]. The Carry Skip adder is a fast one that contains a simple RC adder with a special carry chain, which is called Skip Chain. In this adder, the circuit is divided into sub-blocks. If the output of an XOR gate, used in a sub-block, is '1' for all inputs, the input carry of that sub-block is equal to its output carry; otherwise, the output carry must be calculated as in RC [10]. Carry Look Ahead (CLA) is another fast adder, which is based upon the idea of calculating carries of all bits in parallel [11]. CLA consists of three operations:

1- Propagate /Generate carry based on

$P_i = A_i \quad XOR \quad B_i$ carry propagate (1)

$G_i = A_i B_i$ carry generate (2)

2- Carry generation unit according to

$C_{i+1}=G_i+P_iG_{i-1}+P_iP_{i-1}G_{i-2}+...+P_iP_{i-1}...P_2P_1G_0+P_iP_{i-1}...P_1P_0C_0$ (3)

3- Calculating the output with

$S_i = P_i \quad XOR \quad C_{i-1}$ (4)

Another fast adder is Carry Select adder, which is divided into sub-blocks each made up of RCAs. Figure 1 shows a typical 16-bit Carry Select adder [12]. According to this figure, two parallel sub-blocks work on '0' and '1' as their input carries, and their outputs are determined based on the input carry by a multiplexer.

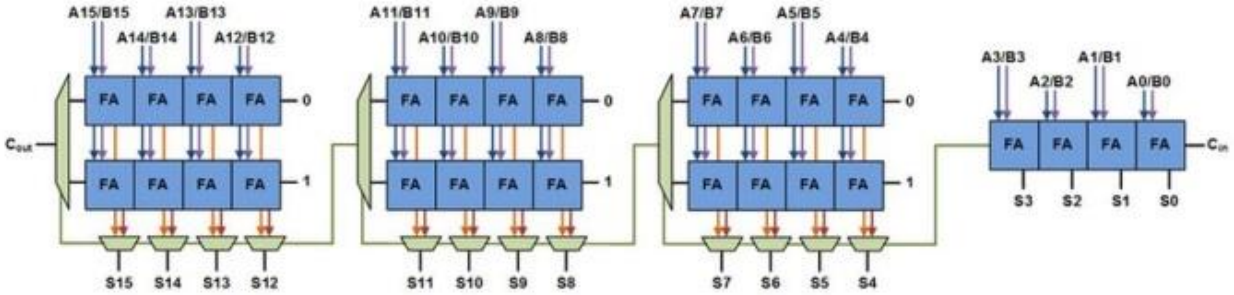

Figure 1. A typical 16-bit Carry Select adder.

Prefix is another fast adder made up of three blocks as CLA adder, in which the first and third blocks are the same. In the second block, the carry of all bits are calculated using graphs [13]. There are different types of graphs; however, the most common ones are Lander-fischer [14], Kogge-Stone [15], Brent-Kung [16], and Han-Carlson [17]. Figure 2 depicts the Kogge-Stone graph.

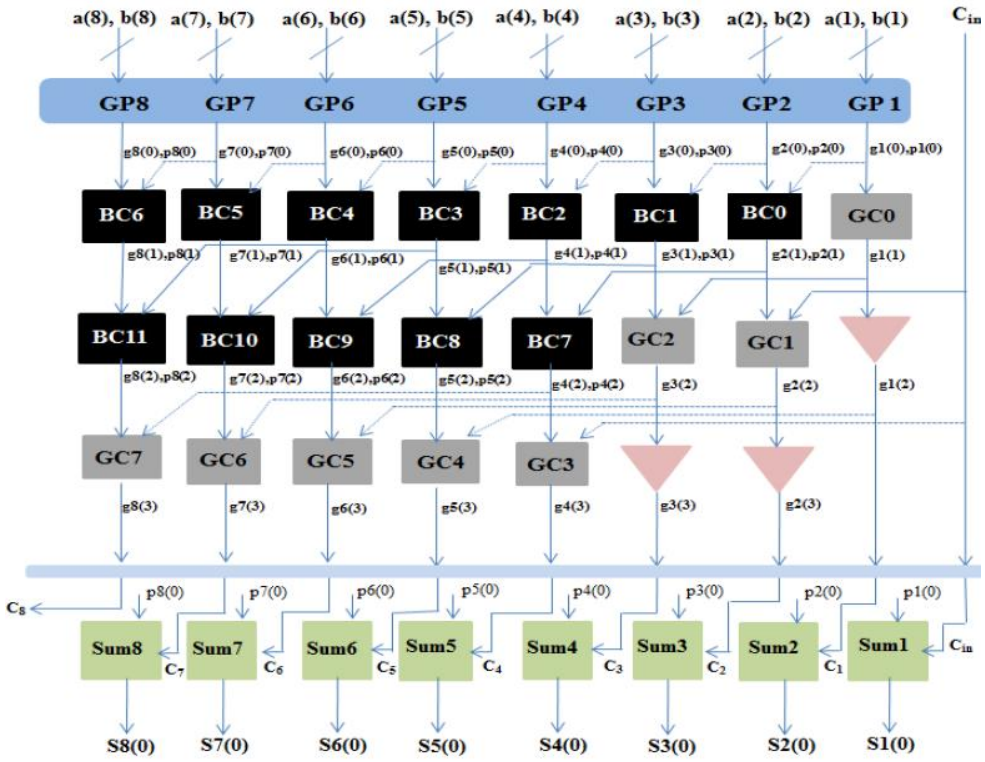

Figure 2. Kogge-Stone graph based on 8-bit Prefix adder [15].

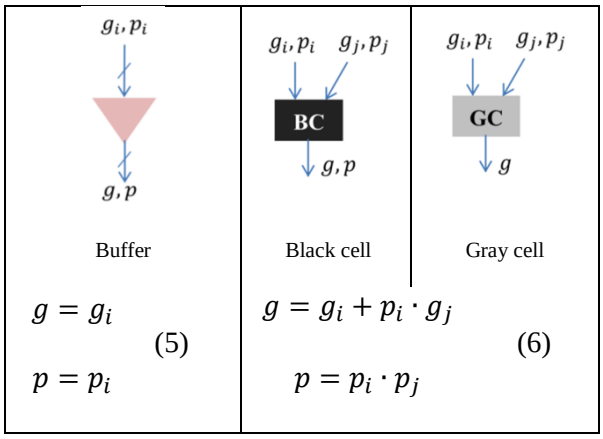

Figure 3. Cells of Perfix adder.

Each cell in figure 2 operates and generates an output according to (5) and (6) in figure 3.

In the proposed adder, a multiplexer-based circuit is designed that generates carries of more significant bits using a less logic length, which, in turn, results in faster computations and lower delays.

The proposed adder in this paper will be compared with the above adders in terms of critical path delay (CPD) and power consumption criteria. In [18], the delay and power products (PDPs) of 32-bit adders have been compared. According to the results obtained, the Prefix Kogge-Stone adder has the lowest delay and the highest power consumption, while the RC adder has the highest delay and the lowest power consumption. The rest of this paper is organized as what follows. The proposed fast adder is introduced in Section 2. Results and analyses are presented in Section 3. Finally, the paper is concluded in Section 4.

2. Proposed Fast Adder

The gate level implementation of a full adder is shown in figure 4. Applying this circuit to a 4-bit RC adder, the circuit shown in figure 5 is obtained.

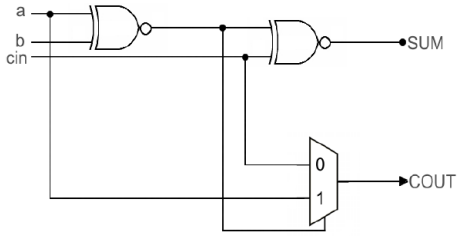

Figure 4. Gate level implementation of a full adder [19].

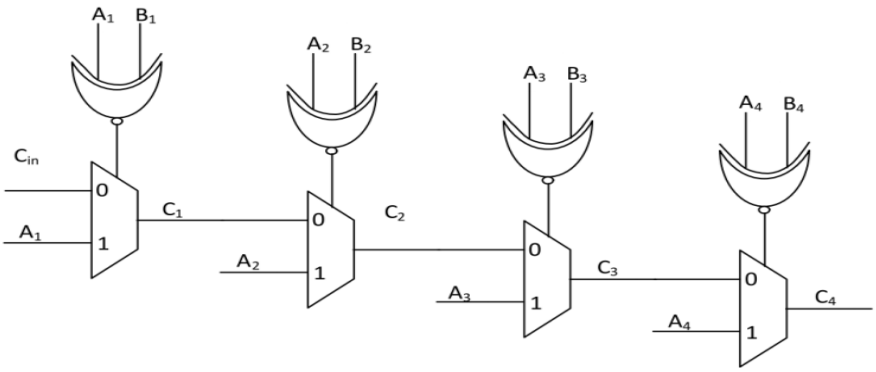

Figure 5. Generating carry out of a 4-bit RC adder based on multiplexers [19].

According to this circuit, the time required for calculating the C4 carry is equal to the computational time used by four multiplexers (neglecting XNOR). In order to reduce this time, we can reconsider the operation of adder circuits as follows:

$$X_1 = A_1 \ XOR \ B_1 \quad (7)$$

$$X_2 = A_2 \ XOR \ B_2 \quad (8)$$

$$X_3 = A_3 \ XOR \ B_3 \quad (9)$$

$$X_4 = A_4 \ XOR \ B_4 \quad (10)$$

Based on these equations, table 1 shows how C_{out} is calculated. As it can be seen in this table, in most cases (fifteen out of sixteen cases), the value for C_{out} can be calculated directly according to the inputs without knowing C_{in}. Therefore, we can propose the circuit shown in figure 6 for calculating C_{out}. In this circuit, the maximum calculation time is equal to the computational time of three multiplexers. This time reduces to that of four multiplexer for calculating the carry of the eighth bit, as shown in figure 7.

Table 1. Truth table for carry calculation of the forth bit of a 4-bit RC adder.

X_1	X_2	X_3	X_4	C_{out}
0 or 1	0 or 1	0 or 1	1	A_4
0 or 1	0 or 1	1	0	A_3
0 or 1	1	0	0	A_2
1	0	0	0	A_1
0	0	0	0	C_{in}

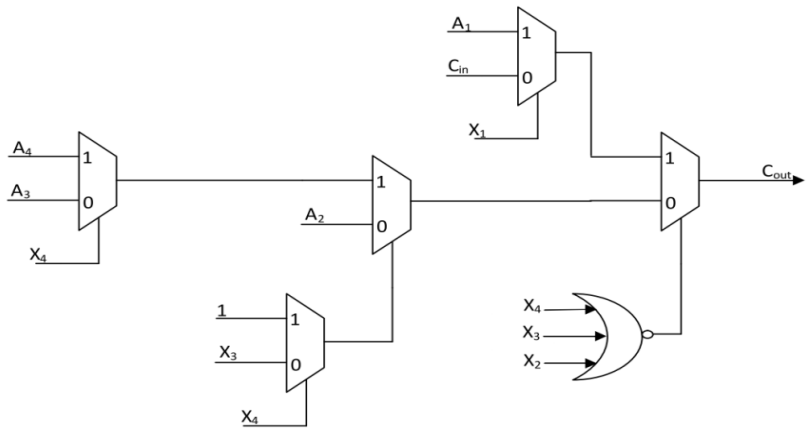

Figure 6. Proposed circuit for calculating output carry (i.e. fourth bit carry).

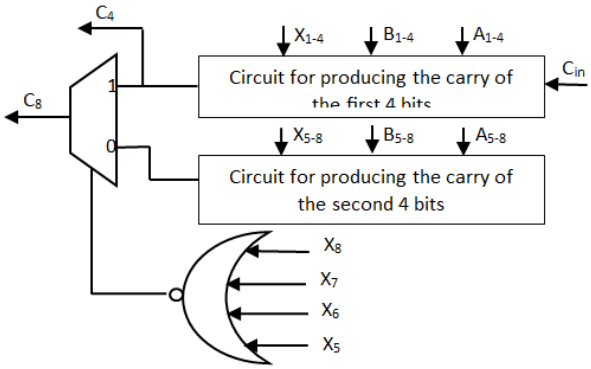

Figure 7. Proposed circuit for calculating the eighth bit carry.

The upper block in figure 7 produces the output carry for the first four bits, i.e. the same as the circuit shown in figure 6, and has an input carry, while the bottom block in this figure generates the output carry for the second four bits, i.e. the same as the circuit shown in figure 8, and has no input carry. Taken together these two blocks, the circuit generating the carry of the eighth bit is achieved. Similarly, the carry of the sixteenth bit with a maximum logic length of five multiplexers is obtained. This procedure can be used for higher-bit adders.

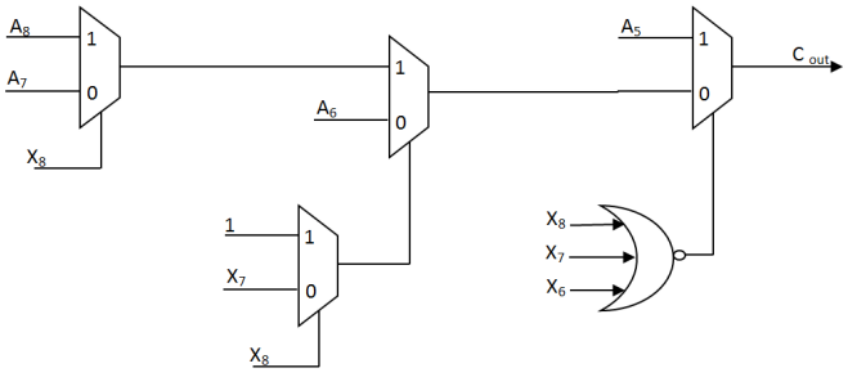

Figure 8. The circuit producing output carry for the second four bits in the circuit producing the eighth bit carry.

Using these fourth, eighth, and upper bits carry generating schemes, it is possible to design fast adders of higher bits. The design paradigm of these adders is shown in figure 9 as a block diagram. This figure shows the proposed sixteen-bit adder, in which the upper blocks are 4-bit RC adders and the bottom blocks are the proposed carry producing circuit as in figure 7. Therewith, the carry calculation process is speeded up by reducing the logic path.

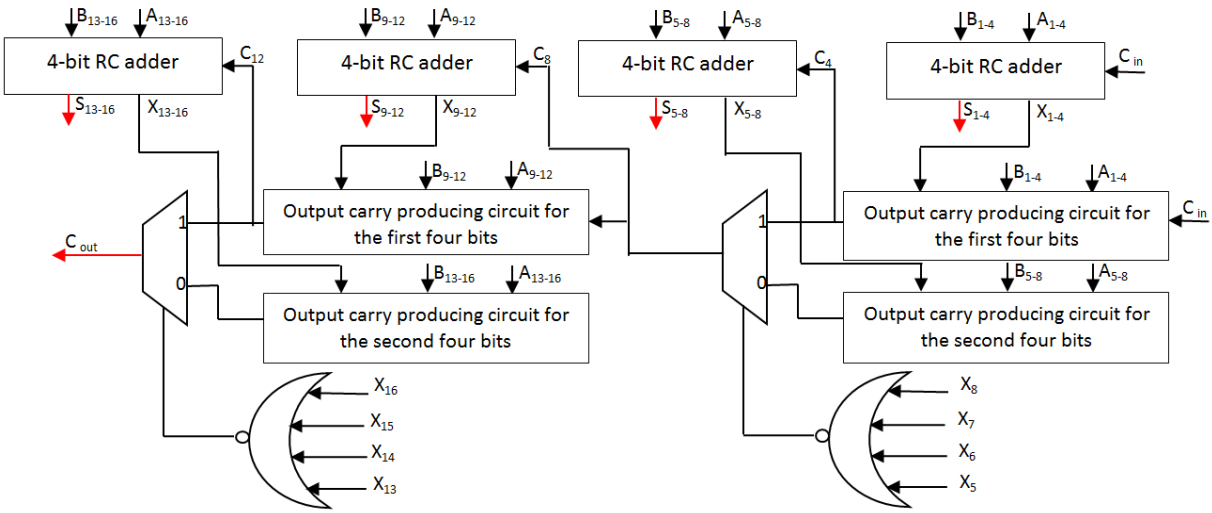

Figure 9. Proposed 16-bit adder.

3. Simulation results

In this section, the proposed adder was compared with other adders in terms of delay, power consumption, and area. All circuits were simulated in ModelSim V6.3, and their syntheses were done in Synopsys Design compiler version C-2009.06-SP5 for Linux.

The simulation and synthesis of the proposed adder and other adders were performed in three typical, fast, and slow conditions in the CMOS 180 nm technology. In the typical condition, the environmental temperature was 25° with a supply voltage of 1.8 V. In the fast condition, the environmental temperature was 0° with supply voltage of 1.98 V. In the slow condition, the environmental temperature was 125° with a supply voltage of 1.62 V.

The proposed adder was compared with the Ripple Carry, Carry select, Carry Skip, Carry LookAhead, and Prefix kogge-stone adders for the 8-, 16-, 32-, and 64-bit cases. The simulations of Ripple Carry, Carry Select, and Carry Skip for 8, 16, and 32 bits were done using 4-bit blocks, while in the case of 64 bits, the 8-bit blocks were applied. The simulation of Carry LookAhead adder was performed hierarchically for all bits in order to benefit from the maximum speed of this adder [11]. The Carry LookAhead adder is implemented with 4-bit blocks for 8- and 16-bit cases, and with 8-bit blocks for 32- and 64-bit cases.

The proposed 8-bit adder uses only one carry generating block, while the 16-bit and 32-bit adders use two blocks for generating carries, as shown in figure 9. 64-bit adders are made up of 8-bit blocks and each block contains eighth bit carry producing, as depicted in figure 7.

In digital circuits, the critical path delay (CPD) is the path that takes the longest time to reach from the input to the output. The CPD of circuits in figures 6, 7, and 9 are plotted in figures 10, 11, and 12, respectively.

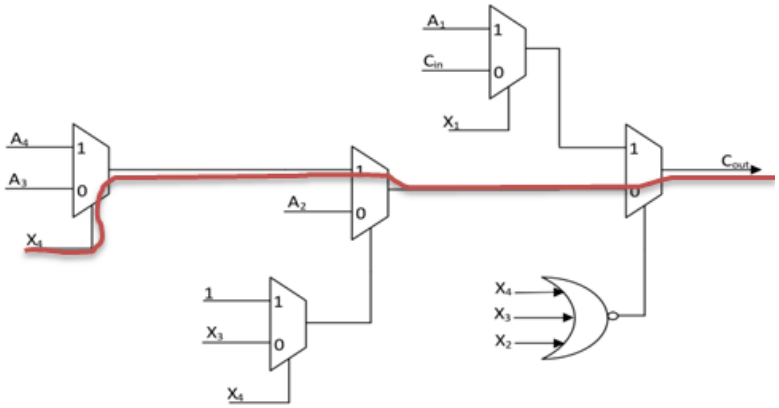

Figure 10. CPD of Figure 6.

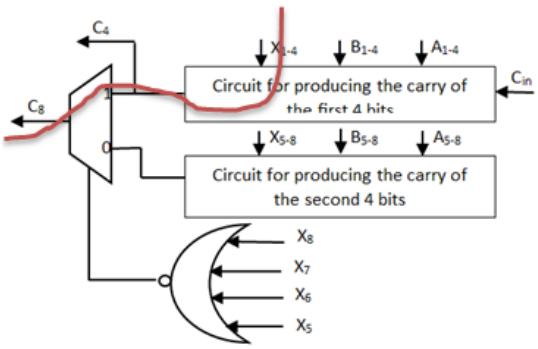

Figure 11. CPD of figure 7.

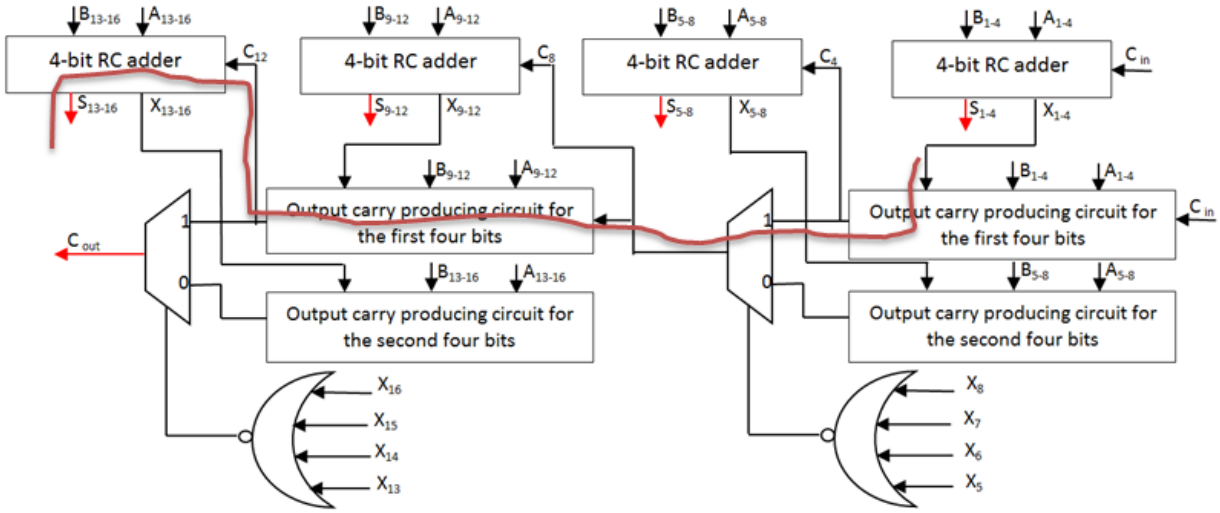

Figure 12. CPD of figure 9.

Figure 13 shows the comparison of the proposed adder with others in terms of critical path delay in the typical condition. As it can be seen, the proposed adder, for the cases of 16 and 32 bits, is placed in the first rank, and for the cases of 8 and 64 bits is placed in the second rank.

Delay optimization is always done according to the power optimization because the more is the power consumption, the less is the delay, and vice versa. Table 2 shows the comparison of the proposed adder with others in terms of power consumption, area, and delay in a typical condition. Tables 3 and 4 repeat the same comparisons, respectively, in fast and slow conditions.

The PDP criterion is used to compare the circuits, and the data is normalized before calculating PDP.

$$(a,b,c,d,e,f) \rightarrow \left(\frac{a}{\sqrt{a^2+b^2+c^2+d^2+e^2+f^2}}, \frac{b}{\sqrt{a^2+b^2+c^2+d^2+e^2+f^2}}, \frac{c}{\sqrt{a^2+b^2+c^2+d^2+e^2+f^2}}, \frac{d}{\sqrt{a^2+b^2+c^2+d^2+e^2+f^2}}, \frac{e}{\sqrt{a^2+b^2+c^2+d^2+e^2+f^2}}, \frac{f}{\sqrt{a^2+b^2+c^2+d^2+e^2+f^2}}\right)$$

in which a, b, c, d, e, and f stand for Ripple Carry, Carry Select, Carry Skip, Carry LookAhead, and Prefix kogge-stone, and the proposed adder respectively, as shown in table 2.

Then the normalized delay and power are multiplied to obtain PDP. The lower PDP indicates the better performance. Figure 14 shows the PDPs of the proposed adder and other adders in the typical condition. As it can be seen in this figure, PDP of the proposed fast adder is the lowest in all cases.

4. Conclusion

In this paper, we have proposed a new adder that generates carries of higher bits with a lower logic length. Therefore, it has the privilege of a higher computational speed and reduced delay. The proposed adder is designed using multiplexers and NOR gates. We minimized both delay and power consumption with less hardware than other adders that just improved the computational speed. PDP, which incorporates both delay and power, is applied as a criterion for comparison. The simulation and synthesis results for cases of 8, 16, 32, and 64 bits show that the proposed adder is

superior to the other ones in terms of delay, power consumption and also PDP.

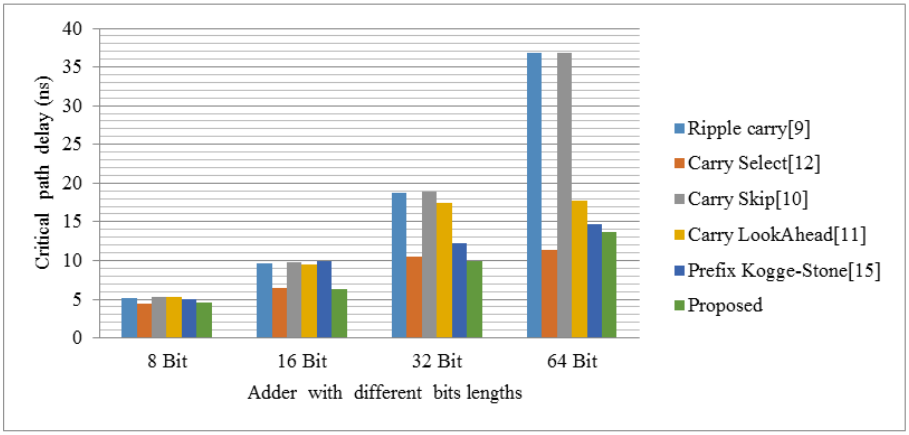

Figure 13. Comparison of the proposed fast adder with other fast adders in terms of critical path delay in a typical speed.

Table 2. Comparison of the proposed fast adder with other fast adders in terms of delay, area, and power consumption in a typical speed with t = 25 °C and V_{supply} = 1.80 v.

	8-bit adder			16-bit adder			32-bit adder			64-bit adder		
Adder type	Critical path delay (ns)	power (mW)	Area (μm^2)	Critical path delay (ns)	power (mW)	Area (μm^2)	Critical path delay (ns)	power (mW)	Area (μm^2)	Critical path delay (ns)	power (mW)	Area (μm^2)
Ripple Carry	5.13	0.83	13518	9.66	1.72	26850	18.72	3.46	53514	36.84	6.99	106843
Carry Select	4.37	1.36	21997	6.52	3.31	53034	10.56	7.17	115108	11.42	15.7	229390
Carry Skip	5.27	0.87	13715	9.79	1.79	27047	18.84	3.62	53711	36.89	7.06	107233
Carry LookAhead	5.32	0.79	14212	9.5	2.19	38327	17.42	4.75	82377	17.69	10.54	180564
Prefix Kogge-Stone	5.02	1.31	23011	9.98	3.13	53479	12.26	8.07	134952	14.64	18.05	313066
Proposed	4.54	0.92	15091	6.29	2.24	35043	9.97	4.56	69900	13.63	9.32	142813

Table 3. Comparison of the proposed fast adder with other fast adders in terms of delay, area, and power consumption in a fast speed with t =0 °C and V_{supply} = 1.98 v.

	8-bit adder			16-bit adder			32-bit adder			64-bit adder		
Adder type	Critical path delay (ns)	power (mW)	Area (μm^2)	Critical path delay (ns)	power (mW)	Area (μm^2)	Critical path delay (ns)	power (mW)	Area (μm^2)	Critical path delay (ns)	power (mW)	Area (μm^2)
Ripple Carry	3.30	1.07	13145	6.33	2.20	26104	12.40	4.46	52021	24.53	8.96	103856
Carry Select	2.52	1.76	21437	4.05	4.27	51724	7.06	9.32	112298	8.58	20.00	223560
Carry Skip	3.32	1.09	13332	6.45	2.28	26664	12.72	4.62	53328	25.27	9.33	106656
Carry LookAhead	3.75	1.00	14212	6.78	2.85	38703	13.09	5.76	77077	11.62	13.97	181234
Prefix Kogge-Stone	3.31	1.69	23011	6.83	4.05	53479	8.76	10.43	134958	9.46	23.34	312999
Proposed	2.82	1.18	15091	4.26	2.90	35043	6.81	5.91	69900	9.26	11.84	139826

Table 4. Comparison of the proposed fast adder with other fast adders in terms of delay, area, and power consumption in a low speed with t = 125°C and V_{supply} = 1.62 v.

	8-bit adder			16-bit adder			32-bit adder			64-bit adder		
Adder type	Critical path delay (ns)	power (mW)	Area (µm²)	Critical path delay (ns)	power (mW)	Area (µm²)	Critical path delay (ns)	power (mW)	Area (µm²)	Critical path delay (ns)	power (mW)	Area (µm²)
Ripple Carry	8.45	0.657	13518	16.21	1.35	26850	31.73	2.72	53514	62.77	5.48	106834
Carry Select	5.91	1.07	22001	9.19	2.60	53038	15.77	5.63	115111	18.34	12.32	229390
Carry Skip	8.63	0.682	13715	16.37	1.40	27047	31.87	2.83	53711	62.78	5.53	107233
Carry LookAhead	8.66	0.622	14212	15.65	1.71	38327	28.57	3.72	82377	29.41	8.28	180167
Prefix Kogge-Stone	8.10	1.02	23011	16.22	2.45	53479	20.31	6.30	134952	24.18	14.10	312879
Proposed	7.09	0.725	15091	10.18	1.75	35043	16.40	3.57	69900	22.06	7.28	142813

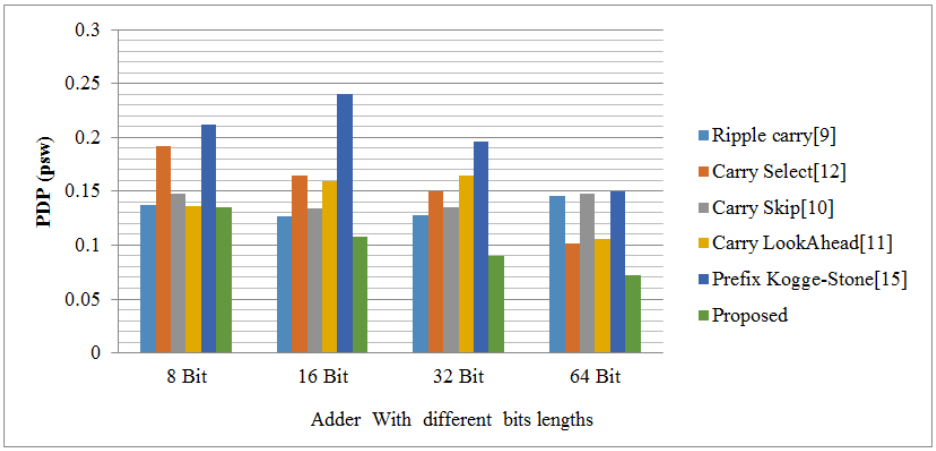

Figure 14. Comparing the PDP of the proposed adder and other adders.

References

[1] Balasubramanian, P. & Mastorakis, E. (2009). High speed gate level synchronous full adder designs, WSEAS Transactions on Circuits and Systems, vol. 8, no. 2, pp. 290-300.

[2] Ocheretnij, V., Marienfeld, E., Sogomonyan, S. & Gossel, (2004). Self-checking code-disjoint carry-select adder with low area overhead by use of add1-circuits, 10th IEEE International On-Line Testing Symposium, vol. 9, no. 1, pp. 31-36.

[3] Mousavi, A. M. & Khodadadi, M. (2018). A Fast and Self-Repairing Genetic Programming Designer for Logic Circuits." Journal of AI and Data Mining, vol. 6, no. 2, pp. 355-363.

[4] Papachatzopoulos, K. & Vassilis, V. (2018). Low-Power Addition with Borrow-Save Adders under Threshold Voltage Variability, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 4, pp. 572-576.

[5] Barani, F., & Nezamabadi-pour, H. (2018). BQIABC: A new Quantum-Inspired Artificial Bee Colony Algorithm for Binary Optimization Problems. Journal of AI and Data Mining, vol. 6, no. 1, pp. 133-143.

[6] Yezerla S. K. & Naik, B. R. (2014). Design and Estimation of delay, power and area for Parallel prefix adders, in Engineering and Computational Sciences (RAECS), vol. 20, no. 4, pp. 1-6.

[7] Saadtjoo, M. A., & Babamir, S. M. (2018). Optimizing Cost Function in Imperialist Competitive Algorithm for Path Coverage Problem in Software Testing. Journal of AI and Data Mining, vol. 6, no. 2, pp. 375-385.

[8] Rezvani, M. (2018). Assessment Methodology for Anomaly-Based Intrusion Detection in Cloud Computing. Journal of AI and Data Mining, vol. 6, no. 2, pp. 387-397.

[9] Weste, N. H. and Eshraghian, K. (1994). Principles of CMOS VLSI design, Addison-Wesley New York, vol. 188, no. 7, pp. 245-250.

[10] Kantabutra, V. (1993). Designing optimum one-level carry-skip adders, IEEE Transactions on Computers, vol. 42, no. 4, pp. 759-764.

[11] Pirsch, P. (1998). Architectures for digital signal processing, John Wiley & Sons, Inc. vol. 10, no. 3, pp. 245-260.

[12] Bedrij, O. Carry-select adder. (1962). IRE Transactions on Electronic Computers, vol. 10, no. 4, pp. 340-346.

[13] Beaumont-Smith, A. &b Lim, C. C. (2001). Parallel prefix adder design, in Computer Arithmetic, 15th IEEE Symposium on, vol. 24, no. 5, pp. 218-225.

[14] Ladner R. E. & Fischer, M. J. (1980). Parallel prefix computation, Journal of the ACM (JACM), vol. 27, no. 5, pp. 831-838.

[15] Kogge, P. M. & Stone, H. S. (1973). A parallel algorithm for the efficient solution of a general class of recurrence equations, IEEE transactions on computers, vol. 100, no. 4, pp. 786-793.

[16] Brent, R. P. & Kung, H.-T. (1979). A regular layout for parallel adders, vol. 5, no. 2, pp. 260-264.

[17] Han, T., & Carlson, D. A. (1987). Fast area-efficient VLSI adders, in Computer Arithmetic (ARITH), IEEE 8th Symposium on, vol. 8, no. 3, pp. 49-56.

[18] Bahadori, M., Kamal, M., Afzali-Kusha, A. & Pedram, M. (2016). A comparative study on performance and reliability of 32-bit binary adders, Integration, the VLSI Journal, vol. 53, pp. 54-67.

[19] Jiag, Y. & Al-Sheraidah, A. (2004). A novel multipleer-based low-power full adder, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 51, pp. 345-348.

ø ÍPDP Á € Ì y P e Ê Ź] Z € » ˆ ℃ Ę Ž À ^ » Ɇ ¼ ž n Ë € ‡ Ã | À À ¯

* Ê ¤ · Z] Á ¼ ‡ Ċ Z É Ë ¹ ½ v Ĉ w ĉ Ñ ¾ ö ù í w Ā ö £ á º Ċ ú ³

. ½ Y € Ë Y ¶] , Z ‖] Ê] ¿ , Y Á € Ì ‹ Â ¿ ∉ ê Â À b • » Ã Z ´ Á Œ © ¥ [Ê ‡ | À Æ » Ã | ° Œ ¿ Y {

ارسال ۲۰۱۸/۰۷/۰۱؛ بازنگری ۲۰۱۸/۰۷/۲۳؛ پذیرش ۲۰۱۸/۱۱/۱۶

چکیده

مدار جمع‌کننده یکی از اجزای اصـلی سـیسـتم‌های محاسـباتی دیجیتالی اسـت که تأثیر زیادی بر میزان کارایی آنها دارد، از اینرو بهینه‌سـازی مدار جمع‌کننده در بهینه‌سـازی سـیسـتم دیجیتالی تأثیر زیادی خواهد داشـت. انواع مختلفی از جمع‌کننده‌ها وجود دارد که هر کدام از آنها با بهره‌گیری از الگوریتم‌های خاصـی، عملیات جمع را با تأخیر کم محاسبه می‌نمایند. علاوه بر تأخیر محاسبات کم، توان مصرفی کمینه نیز به عنوان هدف اصلی طراحی مدار جمع‌کننده به حساب می‌آید. در این مقاله جمع‌کننده‌ای پیشـنهاد می‌شـود که مدار آن به زیربلوک‌هایی تقسـیم شده و برای محاسبه رقم‌نقلی هر زیربلوک، مداری مبتنی بر مالتی‌پلکسـر و گیت NOR طراحی شـده است. شبیه‌سازی و سنتز مدار جمع‌کننده پیشنهادی در بیت‌های ۸، ۱۶، ۳۲ و ۶۴ انجام شـده و با دیگر جمع‌کننده‌های سـریع رایج مقایسـه شـده اسـت. نتایج سـنتز نشـان می‌دهد جمع‌کننده ۱۶ و ۳۲ بیتی پیشنهادی کمترین تأخیر محاسبات را در بین دیگر جمع‌کننده‌ها داشته و مقدار PDP جمع‌کننده پیشنهادی نیز در همه بیت‌ها بهترین مقدار را دارد.

کلمات کلیدی جمع‌کننده سریع، جمع‌کننده توان پایین، جمع‌کننده Carry select، جمع‌کننده Carry LookAhead.