ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(17), DOI: 10.17485/ijst/2016/v9i17/93033, May 2016* ISSN (Online) : 0974-5645

Analysis of Low Power Conditional Sum Adder

M. Siva Kumar*, Syed Inthiyaz, V. Narsimha Nayak, M. Bhavani, K. Charan Teja, S. J. S. Rajesh, K. Eswar Reddy and G. Sruthi Keerthana

Department of Electronics and Communication Engineering, KL University, Vijayawada - 522502, Andhra Pradesh, India;
siva4580@kluniversity.in, Syedinthiyaz@kluniversity.in, narasimhanayak@kluniversity.in, bhavanimathi.147@gmail.com, charan.charantejakodela@gmail.com, saladijsrajesh1234@gmail.com, eshwarreddy226@gmail.com, g.keerthana6@gmail.com

Abstract

Objective: To implement high speed arithmetic systems especially conditional sum adder is used. It consists mainly conditional cells and sum cells. **Method/Analysis:** It contains sum and carry with input 1 and 0. These are used to reduce delay generated by the carry propagator. The advantage using conditional sum adder is addition is much faster. **Finding:** In this project we are going to deal with the low power conditional sum addition rule. By doing this it will basically reduce the nodes which are present internally and number of multiplexer in the adder design. By implementing this adder in cadence software the power will be reduced and the process can be done by applying clock gating and by without applying clock gating. **Novelty/Improvement:** For implementing this adder Verilog code is used. And this will be executed in cadence software. Because of using Verilog code, using the digital cadence software.

Keywords: Conditional Carry Addition (CCA), Clock Gating, Conditional Sum Adder (CSA), Low Power

1. Introduction

In real time to maintain swift operation, low power dissipation and to allow data High-performance Digital Signal Processing (DSP) systems are required. It is efficient method for organizing data as well as for structures also. So addition plays a vital role in DSP. The propagation of the carry always occurs in an *n*-bit adder. For a simple structure design ripple carry adder is mainly used.

By using this adder mainly delay will be reduced. The main limitation in this adder is it has a linear growth problem[1]. To avoid this limitation we are going to the carry look ahead adder. CLA can be produced in parallel and fed to all carry generators with a carry in a signal. When the carry bit height increase then there will be an increase in the number of stacked MOS transistors. The bit length of CLA is 4 because of the delay in the higher carry bit. This is created mainly in the cell based IC designs, but it mainly requires high area and demand regular layout. When the propagation delay of CLA is reduced then it's bit length is proportional. Another method to increase the speed of the operation is the usage of carry select adder[1].

This carry select adder[2] operation done along with different carry-ins and then it selects the correct one to perform operation. Such pairs are repeated for 4-bit adder module and then the propagation delay is reduced to n/4 adder modules. By using the multiplexer gate the previous carry selects the correct sum. When compared with the CLA multiplexer delay of each module is fast. In each bit m-bit module-bit adders are present in carry save adder. So the cost of layout area is high. Irregular structures and complex carry select networks are present in these designs. In high speed applications conditional sum adder has high performance when compared with other adder so it is mainly preferable. For high speed performance this CSA uses higher number of multiplexer gates. For

* *Author for correspondence*

this reason it is more irregular. The main limitation for this CSA is it exhibits a large layout area and a delay in multiplexer and power dissipation. This limitation can be solved by increasing the addition bit length of the adder.

For high speed applications in this paper the 32-bit Conditional Carry Adder[3] (CCA), is proposed. It reduces number of multiplexers and internal nodes in the adder. The proposed method has less power dissipation and low capacitive load and operating speed is also high. To implement novel 32 bit adder static cmos logic is used. The supply voltage given is 3.3 v to 1.5v and fabricated using cmos technology. It was finally founded that our proposed CCA has small layout area and good operating faster than the CSA. This paper explains the architecture of 32 bit CCA and implementations are done.

2. Adders

To perform addition in electronics[4] adders are used usually. Adders are not only used for ALU operation but also uses in other processors. Adders are mainly divided into two types they are

- Half adder.
- Full adder.

For numerical representations, such as binary-coded decimal or excess-3, the most common adders mainly operate on binary numbers where two's and one's complement's added to an adder. A more complex adder is required by other signed numbers.

They are different types of complex adders. They are

- Ripple carry adder.
- Carry save adder.
- Carry select adder.
- Conditional sum adder.
- Carry look ahead adder etc.

For any type of these adders we need two inputs and one carry for addition. It produces output for those two inputs by performing addition operation[5].

3. Conditional Sum Adder

The fundamental procedure of CSA is taking two inputs for adding by giving the two outputs with sum is one output and carry will be other output. Here we can sum with single Bit CSA, two bit CSA, three bit CSA adder and So On. Here we will not wait for carry because we are generating the outputs with sum and carry and from that we will get the sum result of our given input.

It performs carry select adder for single piece operation[6]. For instance 16 bit CSA the initial two lines, full adder figures the whole and complete for every piece accepting carry ins 0 and 1, respectively[7]. In the following two lines, multiplexer's sets select the aggregate and do of the upper piece of every square of two, again expecting carries ins of 0 and 1. In the following two columns, multiplexers select the whole and complete of the upper two bits of every piece of four, etc.

At the point when you come to operation of CSA in real life for N = 16 with Cin = 0. In the piece width 1 row[8], a pair of full adders processes the total and complete for every segment. One adder works expecting carry into the section is 0, while alternate accept it is 1. I the square width 2 push, the adder choose the total for the upper portion of the ach piece in view of the do of the lower half. It likewise processes the do of the pair of bits. Again this is done twice, for both conceivable outcomes of carry into the piece[9]. In the square width 4, the adder again chooses the total for the upper half in light of the complete of the whole piece. This procedure is rehashed in ensuing lines until the 16 bit aggregate and the last complete are chosen.

The CSA[10] includes 2N Full adders and 2N log 2N Multiplexers. Similarly as with carry select, the contingent whole adder can be enhanced by figuring out of aggregate XOR and utilizing AND-OR gates as a part of the spot of multiplexers.

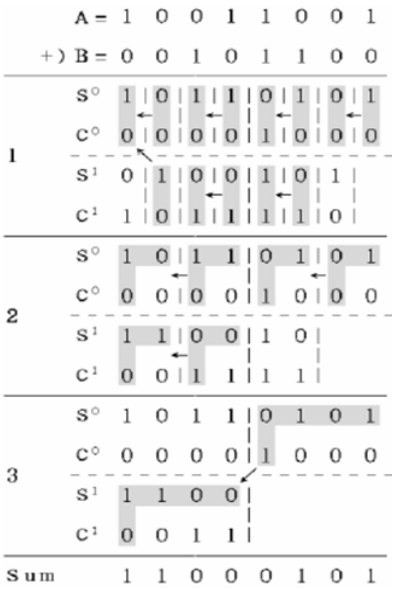

Figure 1. Operation of CSA.

Figure 1 explains the operation of the conditional sum adder by taking sum as 0 and carry as 1 for the above operation.

4. Schematic for Conditional Sum Adder

Figure 2 explains the schematic obtained for the conditional sum adder.

5. Comparison of CSA and CCA

The difference in the proposed conditional carry addition (CCA) and CSA is to create a carry signal and multiplexer in the CCA in every network. Thus due to multiplexer capacitive load it lessen the high speed operation, low power dissipatation and high over load. The operation[11] speed for CCA is faster than the CSA due to the network multiplexer reduction process. The difference between the CSA and the CCA is CCA is expelled from CSA to select sum bits which is shown in Figure 2. The 8 bit CSA equation is given as below

$$S_7 = C_3\,[(C_4^0 .C_5^0 + C_4^0 .C_5^1)(C_6^0 .S_7^0 + C_6^0 .S_7^1) + (C_4^0 .C_5^0 + C_4^0 .C_5^1)(C_6^1 .S_7^0 + C_6^1 .S_7^1] + C_3\,[(C_4^1 .C_5^0 + C_4^1 .C_5^1)(C_6^0 .S_7^0 + C_6^0 .S_7^1) + (C_4^1 .C_5^0 + C_4^1 .C_5^1)(C_6^1 .S_7^0 + C_6^1 .S_7^1)]$$

In the proposed CCA, the XOR gates are moved from the front-end to the back-end and the equation for 8 bit is given below as

$$S_7 = S_7^0\ \{C_3\,[(C_4^0 .C_5^0 + C_4^0 .C_5^1)C_6^0 + (C_4^0 .C_5^0 + C_4^0 .C_5^1)C_6^1] + C_3\,[(C_4^1 .C_5^0 + C_4^1 .C_5^1)C_6^0 + (C_4^1 .C_5^0 + C_4^1 .C_5^1)C_6^1]\}$$

The transistor number of the conditional unit of the CSA is 84p + 84n (= 28 × 3), and the transistor check of the conditional carry unit of the proposed CCA is 51p + 51n (= 17 × 3). Table 1 displays the amounts of 2-to-1 multiplexers used as a part of the n-bit prohibitive aggregate adder (CSA) and the n-bit proposed CCA. The 32-bit CSA requires 186 multiplexers, keeping in mind the proposed 32-bit CCA requires just 129 multiplexers. In like manner, the operation speed and power dispersal are both made moved forward.

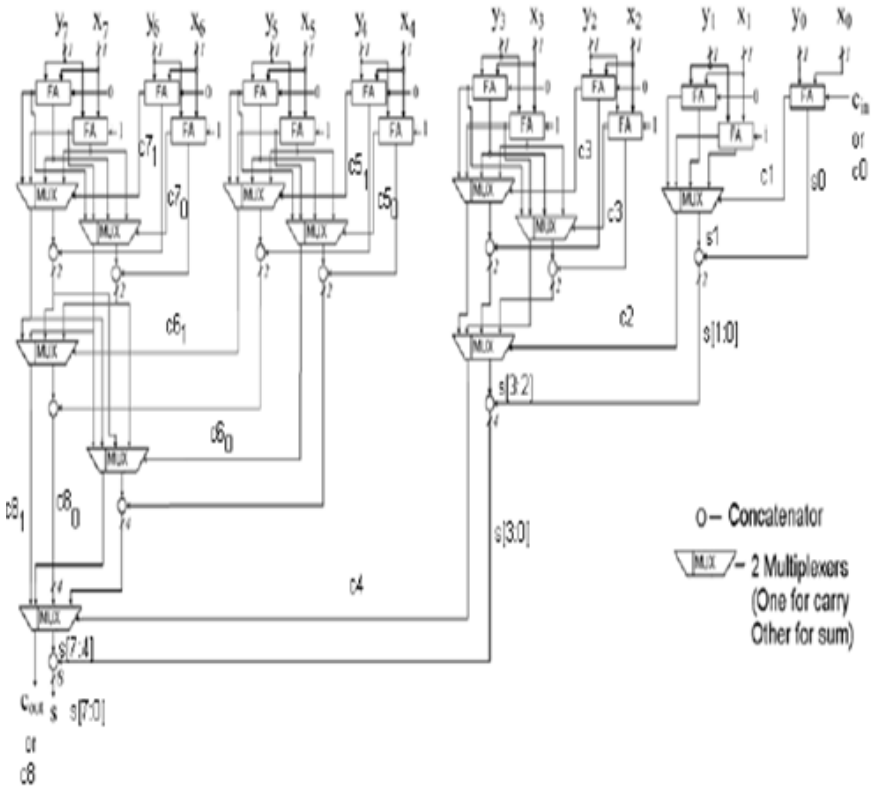

Figure 2. Schematic of CSA.

Table 1. Comparision of various adders

Adder	V_{DD}	Tech.	Sim. delay/ (meas.)
32 bit CSA using static CMOS	3.3 V	0.5 µm CMOS	3.38/(7.03) ns
Proposed 32 bit CCA using static CMOS	3.3 V	0.5 µm CMOS	2.73/(6.09) ns
32 bit CSA using CPL	3.3 V	0.5 µm CMOS	4.05/(5.32) ns
Proposed 32 bit CCA using CPL	3.3 V	0.5 µm CMOS	3.70/(4.42) ns
Proposed 32 bit CCA using CPL	3.3 V	0.35 µm CMOS	2.10 ns
32 bit HSAC [9] (2002)	5 V	1.0 µm BiCMOS	4.0 ns
32 bit [7] (1997)	5 V	1.2 µm CMOS	3.28 ns
16 bit SICNB CCS [8] (2000)	1.5 V	0.8 µm CMOS	14.56 ns

Here Table 1 shows the comparison for various adders when applying the different voltages and different technologies and then obtained different delays.

6. Results

The below shows the output of conditional sum adder results

Figure 3 explains the output obtained for the operation done and the 0 represents sum and 1 represents carry. Figure 4 represents the Schematic of conditional sum adder without applying clock gating.

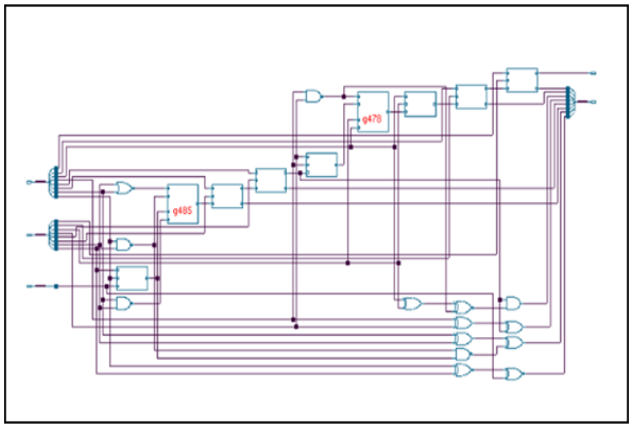

Figure 4. Schematic for CSA without clock gating.

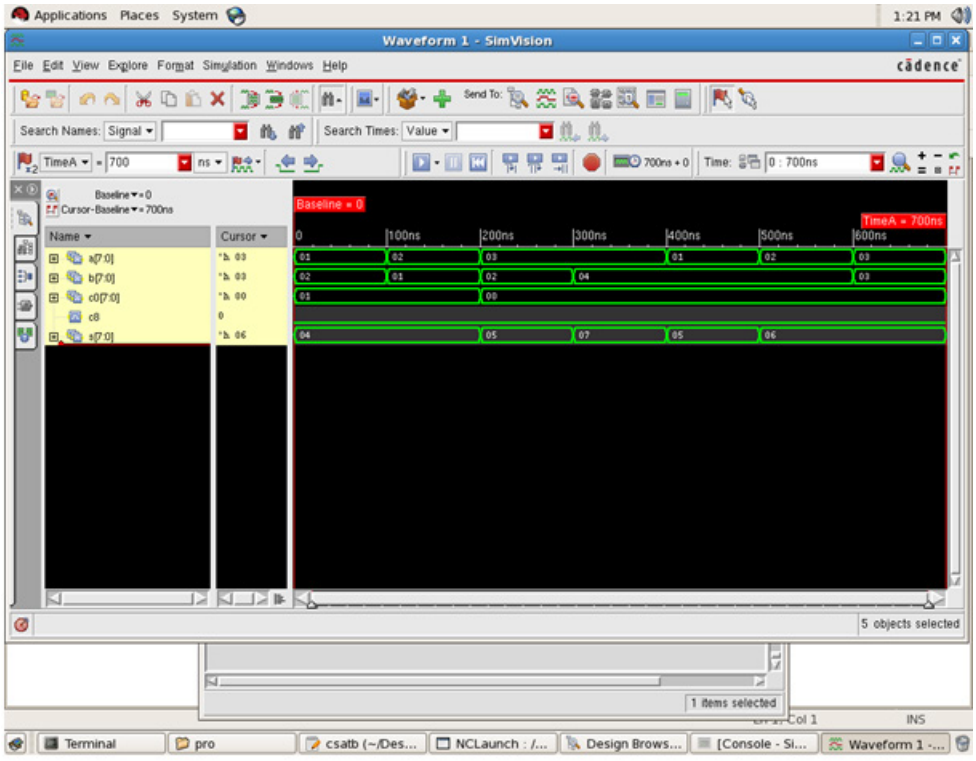

Figure 3. Output for CSA.

Figure 5 represents the gates obtained that are obtained with out clock gating.

```
 Type Instances   Area  Area %
------------------------------
logic        23 154.526  100.0
------------------------------
total        23 154.526  100.0
```

Figure 5. Gates without clock gating.

Figure 6 represents the power obtained that are obtained with out clock gating.

```
=======================================================
  Generated by:           Encounter(R) RTL Compiler v09.10-p104_1
  Generated on:           May 12 2013  04:28:44 PM
  Module:                 csa
  Technology library:     slow_normal 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=======================================================

              Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------
csa         23   613.941  4239.061  4853.002
```

Figure 6. Power without clock gating.

Figure 7 represents the area obtained that are obtained with out clock gating.

```
=======================================================
  Generated by:           Encounter(R) RTL Compiler v09.10-p104_1
  Generated on:           May 12 2013  04:28:19 PM
  Module:                 csa
  Technology library:     slow_normal 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=======================================================

Instance  Cells  Cell Area  Net Area   Wireload
-----------------------------------------------
csa          23        155         0    <none> (D)
```

Figure 7. Area without clock gating.

Figure 8 represents the timing diagram obtained that are obtained with out clock gating.

```
-------------------------------------------------------------
  Pin       Type       Fanout Load Slew Delay Arrival
                          (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------
b[0]        in port         3  4.0    0    +0        0 F
g489/A1                                  +0        0
g489/Y      OAI21XL         2  2.4  131   +88       88 R
g485/B0                                  +0       88
g485/Y      OAI211XL        1  2.5  165  +135      224 F
g483/CI                                  +0      224
g483/CO     ADDFX1          1  2.5   78  +202      426 F
g482/CI                                  +0      426
g482/CO     ADDFX1          3  3.4   84  +186      611 F
g481/B0                                  +0      611
g481/Y      OAI21XL         1  1.0   88   +45      657 R
g478/A1N                                 +0      657
g478/Y      OAI2BB2XL       1  1.0   95   +94      750 R
g477/B0                                  +0      750
g477/Y      OAI2BB1XL       1  2.5   94   +78      828 F
g476/CI                                  +0      828
g476/CO     ADDFX1          1  2.5   78  +182     1011 F
g475/CI                                  +0     1011
g475/CO     ADDFXL          1  0.0   60  +146     1157 F
c8          out port                     +0     1157 F
-------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : b[0]
End-point    : c8
```

Figure 8. Timing without clock gating.

Figure 9 represents the schematic for conditional sum adder that are obtained with applying clock gating.

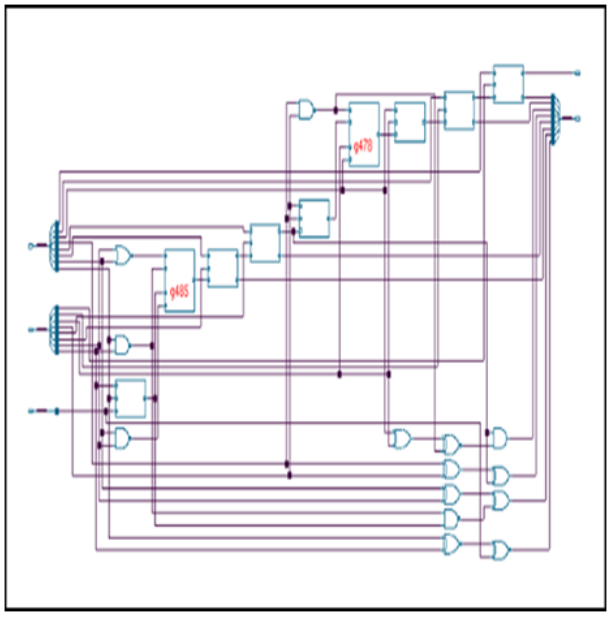

Figure 9. Schematic for CSA with clock gating.

Figure 10 represents the gates for conditional sum adder that are obtained with applying clock gating.

```
 Type Instances   Area  Area %
------------------------------
logic        23 154.526  100.0
------------------------------
total        23 154.526  100.0
```

Figure 10. Gates with clock gating.

Figure 11 represents the power for conditional sum adder that are obtained with applying clock gating.

```
=======================================================
  Generated by:          Encounter(R) RTL Compiler v09.10-p104_1
  Generated on:          Apr 29 2013  02:28:54 PM
  Module:                csa
  Technology library:    slow_highvt 1.0
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
=======================================================

             Leakage  Dynamic   Total
Instance Cells Power(nW) Power(nW) Power(nW)
-----------------------------------------
csa       23  151.744 4157.650 4309.394
```

Figure 11. Power with clock gating.

Figure 12 represents the area for conditional sum adder that are obtained with applying clock gating.

```
rc:/> report area
=======================================================
  Generated by:          Encounter(R) RTL Compiler v09.10-p104_1
  Generated on:          May 12 2013  04:38:13 PM
  Module:                csa
  Technology library:    slow_normal 1.0
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
=======================================================

Instance  Cells  Cell Area  Net Area   Wireload
-----------------------------------------------
csa          23        155         0     <none> (D)
```

Figure 12. Area with clock gating.

Figure 13 represents the timing diagram for conditional sum adder that are obtained with applying clock gating.

```
  Pin        Type       Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
-------------------------------------------------------
b[0]       in port         3  4.0    0    +0      0 F
g489/A1                                 +0      0
g489/Y     OAI21XL         2  2.4  131   +88     88 R
g485/B0                                 +0     88
g485/Y     OAI211XL        1  2.5  165  +135    224 F
g483/CI                                 +0    224
g483/CO    ADDFX1          1  2.5   78  +202    426 F
g482/CI                                 +0    426
g482/CO    ADDFX1          3  3.4   84  +186    611 F
g481/B0                                 +0    611
g481/Y     OAI21XL         1  1.0   88   +45    657 R
g478/A1N                                +0    657
g478/Y     OAI2BB2XL       1  1.0   95   +94    750 R
g477/B0                                 +0    750
g477/Y     OAI2BB1XL       1  2.5   94   +78    828 F
g476/CI                                 +0    828
g476/CO    ADDFX1          1  2.5   78  +182   1011 F
g475/CI                                 +0   1011
g475/CO    ADDFXL          1  0.0   60  +146   1157 F
c8         out port                     +0   1157 F
-------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : b[0]
End-point    : c8
```

Figure 13. Timing with clock gating.

7. Advantages

- Addition is much speedier.
- Used for rapid low power applications.
- The expansion is much speedier as higher piece expansion operation doesn't depend on carry propagation.
- However, the determination of right result sits tight for the past carry to be set.

8. Disadvantages

- It is very complex design.
- It is very costly.
- This is a generally costly plan, concerning N = 2n-bit expansion, it requires 2N-1 full-adders and 2n+1 – n – 2 multiplexers, which are not present in traditional ripple carry adders.

9. Conclusion and Future Scope

The conditional sum adder is designed and simulated by using verilog code and test bench in cadence digital software which is used for high speed low power applications and also very faster adder.

10. Acknowledgement

We sincerely thank to my project guide, who helped me

in all aspects of my project to complete in short term. We also thank KL University for providing necessary facilities towards carrying out this work.

11. References

1. Cheng K-H, Cheng S-W. Improved 32-bit conditional sum adder for low-power high-speed applications. Journal of Information Science and Engineering. 2006; 22(4):975–89.
2. Anjana R, Kanoji V, Somkumar A. Low power conditional sum adder using modified ripple carry adder. Global Journal of Researches in Engineering Electrical and Electronics Engineering. 2014; 14(5):19–23.
3. Lo JC. A fast binary adder with conditional carry generation. IEEE Transactions on Computer. 1997; 46(2):248–53.
4. Wang Y, Pai C, Song X. The design of hybrid carry-look ahead/carry-select adders. IEEE Transactions on Circuit and Systems II: Analog and Digital Signal Processing. 2002; 49(1):16–24.
5. Kumar HAA, Umadevi S. Implementation of fast radix-10 BCD multiplier in FPGA. Indian Journal of Science and Technology. 2015; 8(19):1–6.
6. Praveena R, Nirmala S. Realization of efficient multiplier for low power biomedical signal processing system-on-chip design for portable ECG monitoring systems. Indian Journal of Science and Technology. 2015; 8(24):1–7.
7. Malekpour A, Ejlali A. Improving the energy/power consumption of parallel decimal multipliers. Indian Journal of Science and Technology. 2014; 7(3):1–6.
8. Brent RP, Kung HT. A regular layout for parallel adders. IEEE Transactions on Computers. 1982; C-31(3):260–4.
9. Saleem D, Semicond N, Al-Khalili D. Low power conditional sum adder using pass logic topology. IEEE Candian Conference Electrical and Computer Engineering; Waterloo, Ont. 1998. p. 9–12.
10. Kenney RD, Schulte MJ. High-speed multioperand decimal adders. IEEE Trans Comput. 2005 Aug; 54(8):953–63.
11. Gisamore RT, Swartzerland EE. Negative save sign extension for multiterm adders and multipliers. Journal of Signal Processing Systems. 2008; 52(1):1–11.