SCIREA Journal of Electrics, Communication

http://www.scirea.org/journal/ECAC

May 5, 2019

Volume 2, Issue 1, February 2019

Realization of a Low Power and Area-Efficient VLSI Architecture for Carry Select Adder using Multiplexer

Bala Sindhuri Kandula[1],K.Padma Vasavi[2], I.Santi Prabha[1]

[1]Department of Electronics and communication Engineering, University College of Engineering,JNTUK,Kakinada, India

[2]SVECW, Bhimavram, India

Abstract

Carry Select Adder (CSLA) is the most popular choice for multiply and accumulate operations because of its high performance in fast computations. However, the major drawback for CSLA is resource utilization as it occupies more area and power when compared to Ripple Carry Adder (RCA). Low Power and area efficiency can be achieved by using multiplexer based adders as the switching activity reduces the resource utilization. The proposed architecture is designed by using two bit adders using 4:1 multiplexers and synthesized in cadence RTL compiler using 90nm technology. The performance evaluation of the proposed architecture in terms of area and power is compared with Square Root Carry Select Adder (SQRT CSLA), Square Root Carry Select Adder using Kogge-stone Adder (SQRT-KSA), Square Root Carry Select Adder using Binary to Excess-1 Converter

(SQRTCSLA-BEC),Kogge-stone Square Root Carry Select Adder using Binary to Excess-1 Converter (SQRTKSA-BEC)architectures for different bit depths ranging from 16 bits to 64 bits. The Proposed architecture is proved to be efficient both in terms of area and power when compared to SQRTCSLA, SQRT-KSA, SQRTCSLA-BEC, SQRTKSA-BEC architectures

Keywords: Carry Select Adder (CSLA), Multiplexer Based Adder,Area and Power Efficient, Two Bit Adder.

1. Introduction

The egress Internet of Things (IoT) [1]-[2] demands the digitization because of its pliancy. Signal Conditioning circuits are necessary in order to eliminate the repellent noises and retardations [3]–[10]. Finite-Impulse-Response (FIR) filter accomplished attention in IoT because of its enticing characteristics such as linear phase property and low coefficient sensitivity. But the implementation cost of FIR Filter is high when compared with IIR Filter approaching the identical magnitude response specifications because of multiplier and adder cost. If the order of the filter is increased, then the cost of multiplier and adder is also further increased. Enhancements in the implementation of multiplier and adder can be achieved by switching techniques during run-time. Switching action between different constants and their mapping to the FPGA is best performed by multiplexers [11]. Major components in the implementation of multipliers are partial product generators and adders. The generated partial products are added by using reduction techniques. Many approaches have been proposed for reducing the number of adders in the multipliers. The most commonly used adders are Ripple Carry Adder (RCA), Kogge-stone adder, Square Root Carry Select Adder (SQRT CSLA). Among these, RCA occupies less area but the speed of the adder is very less. In order to improve the speed, CSLA is designed but the area is very high. In order to scale down the area in CSLA, one stage of RCA with input carry is equal to 'one' is replaced with binary to an excess-1 converter. In all these adders, the major digital components are XOR gates, AND gates, OR gates. Multiplexer based adders [12]-[15] gained popularity because of less gate count when compared to XOR gate. In this Paper, Low Power and Area-Efficient VLSI

Architecture for Carry Select Adder using Multiplexer Based Adders are proposed. The rest of the paper is structured as follows. Section 2 deals with CSLA using Multiplexer Based Adder. Section 3 presents the simulation results. Section 4 presents the synthesis results. Finally, the work is concluded in Section 5.

2. Proposed Architecture

The design of VLSI architecture for each of the building blocks of the proposed adder is described in this section. The Proposed adder for 16-bit CSLA is designed by using two-bit adders using 4:1 Multiplexer with input carry is equal to ‘zero’ and input carry is equal to ‘one’ and multiplexers as shown in Fig.1.The adder consists of total 8 groups. Group1 consists of the two-bit adder with inputs a (1:0) and b(1:0) and the input carry Cin. Outputs from the Group1 are sum (1:0) and C (1). From Group2 onwards each group consists of 2 sets of two-bit adders and multiplexer .1st set consists of two-bit adder using 4:1 multiplexer with input carry is equal to ‘zero’ and 2nd set consists of two-bit adder using 4:1 multiplexer with input carry is equal to ‘one’. Carry from the preceding group is given a selection input for the next group multiplexer i.e. group n Selection line is from group n-1, where n is a natural number. Depending on the carry the final sum and carry are selected either from set1 or set2. Similarly, n-bit adder consists of n/2 groups and each group consists of Two-bit adders and a multiplexer. The architecture is further extended to 64 bits. The basic building blocks for this architecture are explained below.

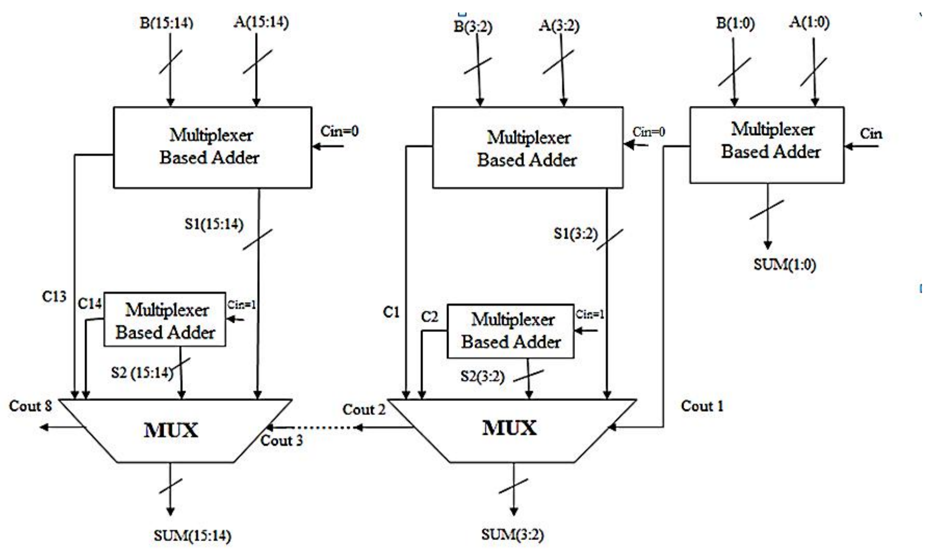

Fig.1. CSLA using Multiplexer Based Adders

2.1 Two bit adder using 4:1 Multiplexer with Input Carry Is Equal To 'Zero':

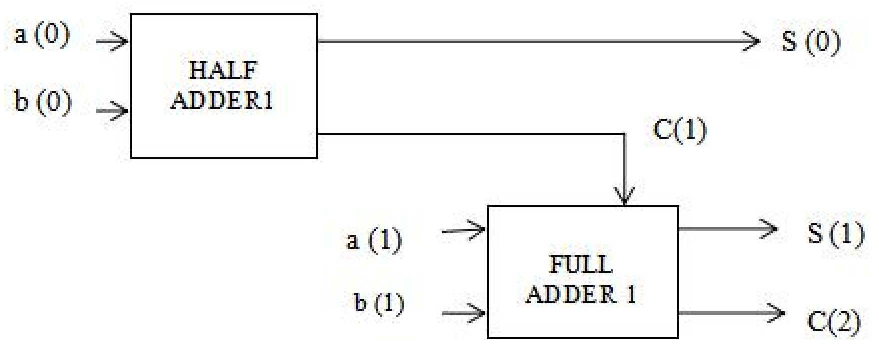

Fig.2. Two bit adder with input Carry is equal to 'Zero'

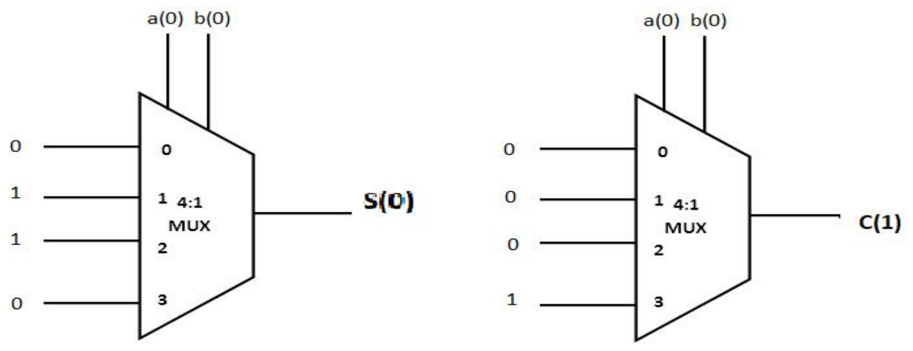

Fig.3. Multiplexer unit for generation of S(0) and C(1)

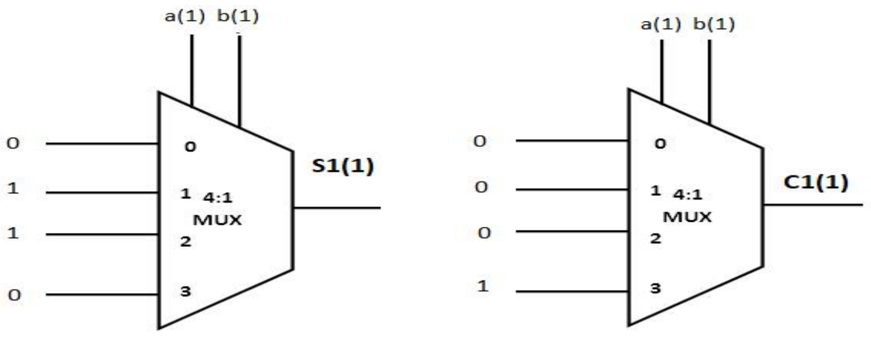

Fig.4. Multiplexer unit for generation of S1(1) and C1(1)

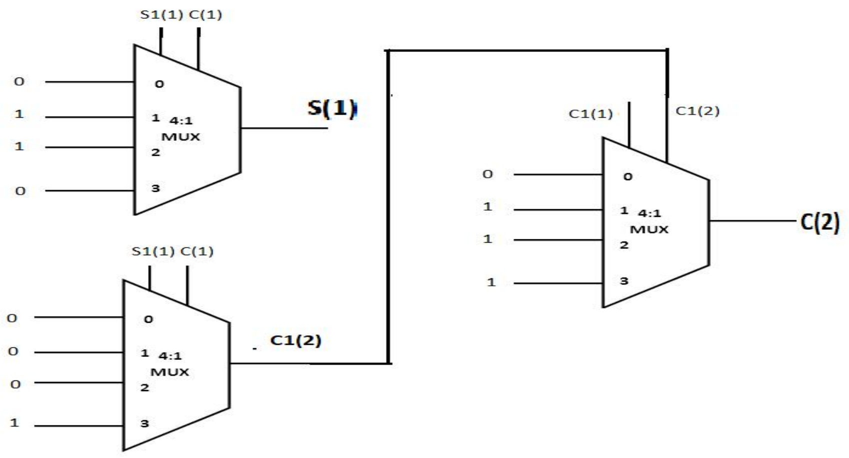

Fig.5. Multiplexer unit for generation of S(1) ,C1(1) and C(2)

Two-bit adder with input Carry is equal to 'Zero' consists of half adder and the full adder as shown in Fig.2.The proposed two-bit adder is designed using 4:1multiplexers.Selection lines for the first and second multiplexers are a(0) and b(0).The inputs to the first multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,0 in order to obtain sum S(0).The equations for obtaining S(0) and C(1) is shown in Eq.(1) and Eq.(2). During runtime, any one of the paths is active depending on the selection lines. The inputs to the second multiplexer at selection input 0, input 1, input 2, input 3 are taken as 0,0,0,1 in order to obtain carry C(1) as shown in Fig.3. Selection lines for the third and fourth multiplexers are a(1) and b(1). The inputs to the third multiplexer at selection input 0, input 1, input 2, input 3 are taken as 0,1,1,0 in order to obtain S1(1). The inputs to the fourth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,0,0,1 in order to obtain C1(1) as shown in Fig.4. The equations for obtaining S1(1) and C1(1) is shown in Eq.(3) and Eq.(4). Selection lines for the fifth and sixth multiplexers are S1(1) and C(1). The inputs to the fifth multiplexer at selection input 0, input 1, input 2, input 3 are taken as 0,1,1,0 in order to obtain S(1). The inputs to the sixth multiplexer at selection input 0, input 1, input 2, input 3 are taken as 0,0,0,1 in order to obtain carry C1(2). Selection lines for the seventh multiplexer are C1(1) and C1(2). The inputs to the seventh multiplexer at selection input 0, input 1,input 2,input 3 are taken as 0,1,1,1 in order to obtain carry C(2) as shown in Fig.5. The equations for obtaining S(1), C1(2) and C(2) is shown in Eq.(5), Eq.(6) and Eq.(7). The final architecture for two-bit adder using 4:1 multiplexers with input Carry is equal to 'Zero' is shown in Fig.6.

$$\boldsymbol{S(0) = \overline{a(0)b(0)}0 + \overline{a(0)}b(0).1 + a(0).\overline{b(0)}1 + a(0).b(0).0 = \overline{a(0)}b(0) + a(0).\overline{b(0)}} \qquad \textbf{(1)}$$

$$\boldsymbol{C(1) = \overline{a(0)}.\overline{b(0)}.0 + \overline{a(0)}.b(0).0 + a(0).\overline{b(0)}.0 + a(0).b(0).1 = a(0).b(0)} \qquad \textbf{(2)}$$

$$\boldsymbol{S1(1) = \overline{a(1)}.b\overline{(1)}.0 + \overline{a(1)}.b(1).1 + a(1).\overline{b(1)}.1 + a(1).b(1).0 = \overline{a(1)}.b(1) + a(1).\overline{b(1)}} \qquad \textbf{(3)}$$

$$\boldsymbol{C1(1) = \overline{a(1)}.\overline{b(1)}.0 + \overline{a(1)}.b(1).0 + a(1).\overline{b(1)}.0 + a(1).b(1).1 = a(1).b(1)} \qquad \textbf{(4)}$$

$$\boldsymbol{S(1) = \overline{S1(1)}.\overline{C(1)}.0 + \overline{S1(1)}.C(1).1 + S1(1).\overline{C(1)}.1 + S1(1).C(1).0 = \overline{S1(1)}.C(1) + S1(1).\overline{C(1)} = S1(1) \oplus C(1) = a(1) \oplus b(1) \oplus C(1)} \qquad \textbf{(5)}$$

$$\boldsymbol{C1(2) = \overline{S1(1)}.\overline{C(1)}.0 + \overline{S1(1)}.C(1).0 + S1(1).\overline{C(1)}.0 + S1(1).C(1).1 = S1(1).C(1) = \left(\overline{a(1)}.b(1) + a(1).\overline{b(1)}\right).C(1)} \qquad \textbf{(6)}$$

$$\boldsymbol{C(2) = \overline{C1(1)}.\overline{C1(2)}.0 + \overline{C1(1)}.C1(2).1 + C1(1).\overline{C1(2)}.1 + C1(1).C1(2).1 = C1(1) + C1(2) = a(1)b(1) + \left(\overline{(a(1)}.b(1) + a(1).\overline{b(1)}\right).C(1) = a(1).b(1) + a(1).\,\overline{b(1)}.C(1) + \overline{a(1)}.b(1).C(1) = a(1)[b(1) + C(1)] + \overline{a(1)}.b(1).C(1) = a(1)b(1) + a(1)C(1) + \overline{a(1)}.b(1).C(1) = a(1)b(1) + \left[a(1) + \overline{a(1)}.b(1)\right]C(1) = a(1)b(1) + b(1)C(1) + a(1)C(1)} \qquad \textbf{(7)}$$

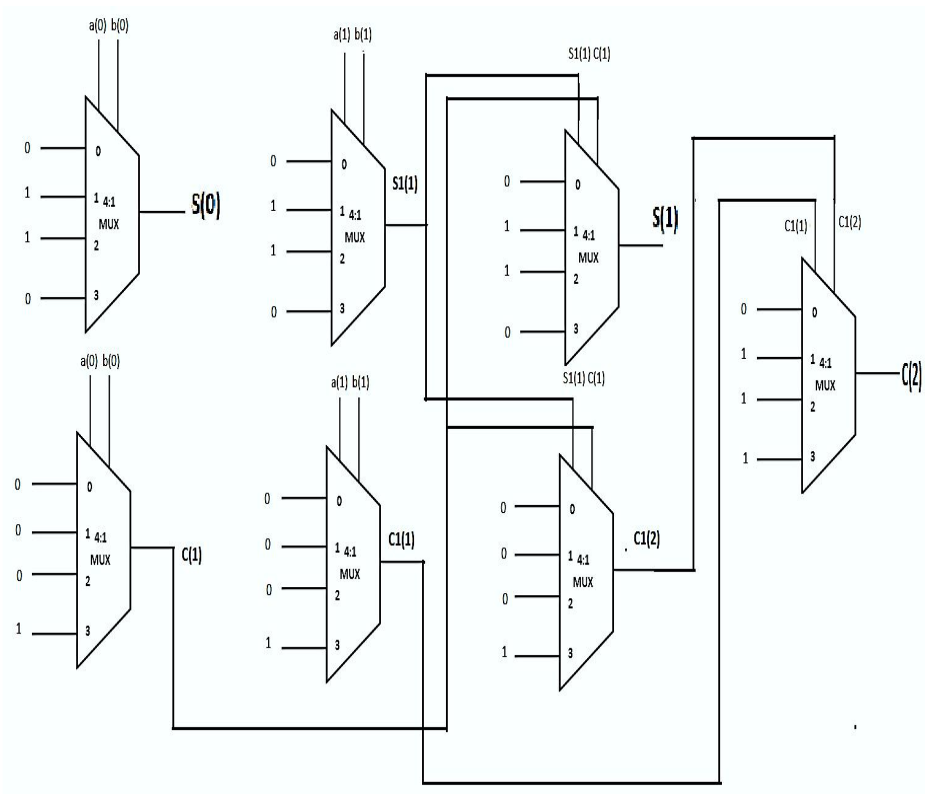

Fig.6. Two bit adder using 4:1 Multiplexers with input Carry is equal to 'Zero'

2.2 Two bit adder using 4:1 Multiplexer with input carry is equal to 'one'

Two bit adder with input Carry is equal to 'one' consists of full adders as shown in Fig.7.The proposed two bit adder is designed using multiplexers. Selection lines for the first and second multiplexers are a(0) and b(0).The inputs to the first multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,0 in order to obtain sum S1(0). The inputs to the second multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,0,0,1 in order to obtain carry C1(0) as shown in Fig.8. The equations for obtaining S1(0) and C1(0) is shown in Eq.(8) and Eq.(9).Selection lines for the third ,fourth and fifth multiplexers are S1(0) and Cin. The inputs to the third multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,0 in order to obtain S(0).The inputs to the fourth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,0,0,1 in order to obtain C2(0). The inputs to the fifth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,1 in order to obtain C(1) as shown in Fig.9. The equations for obtaining S(0) , C2(0) and C(1) is shown in Eq.(10), Eq.(11) and Eq.(12).Selection lines for the sixth and seventh multiplexers are a(1) and b(1). The inputs to the sixth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,0 in order to obtain S2(0). The inputs to the seventh multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,0,0,1 in order to obtain carry C3(0) as shown in Fig.10.The equations for obtaining S2(0) and C3(0) are shown in Eq.(13) and Eq.(14).

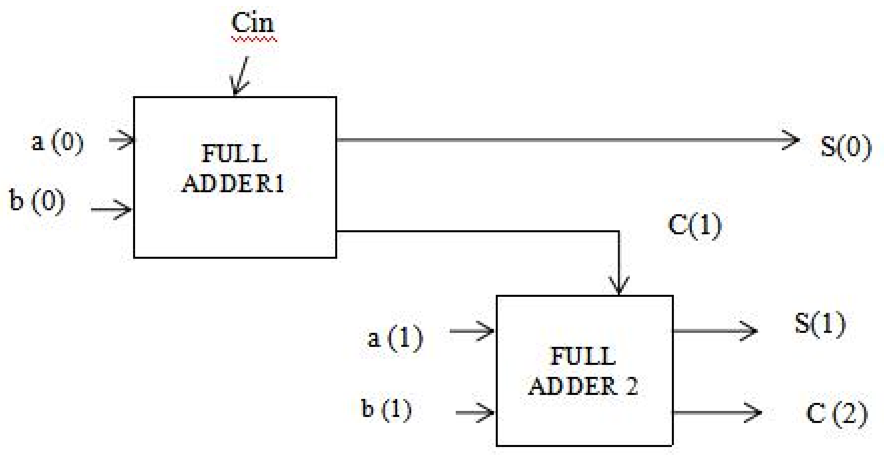

Fig.7. Two bit adder with input Carry is equal to 'One'

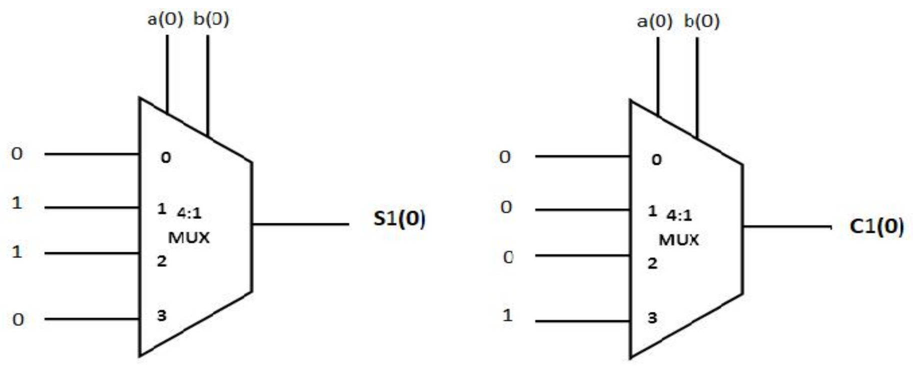

Fig.8. Multiplexer unit for generation of S1 (0), C1(0) .

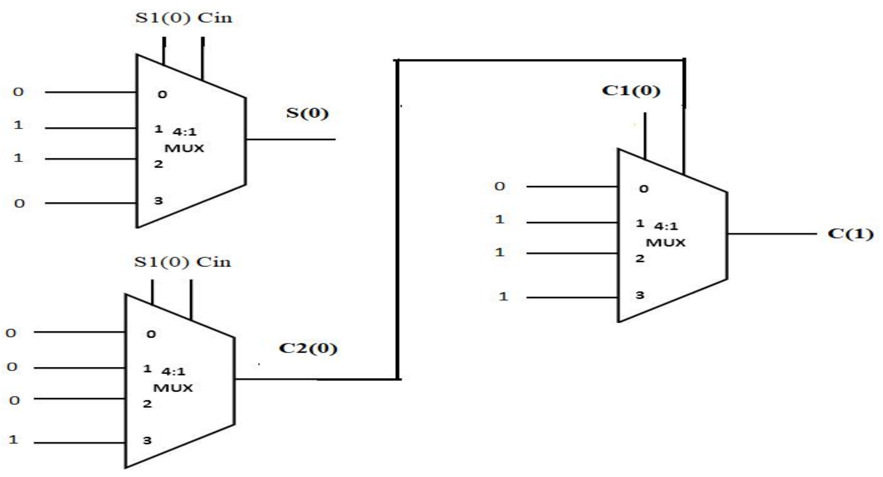

Fig.9. Multiplexer unit for generation of S(0) ,C2(0) and C(1).

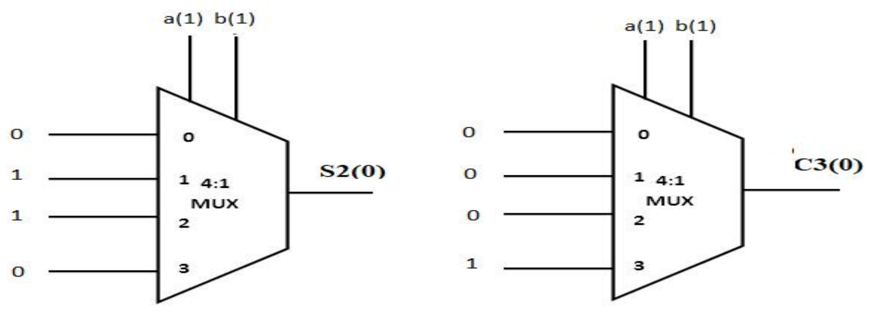

Fig.10. Multiplexer unit for generation of S2(0) and C3(0).

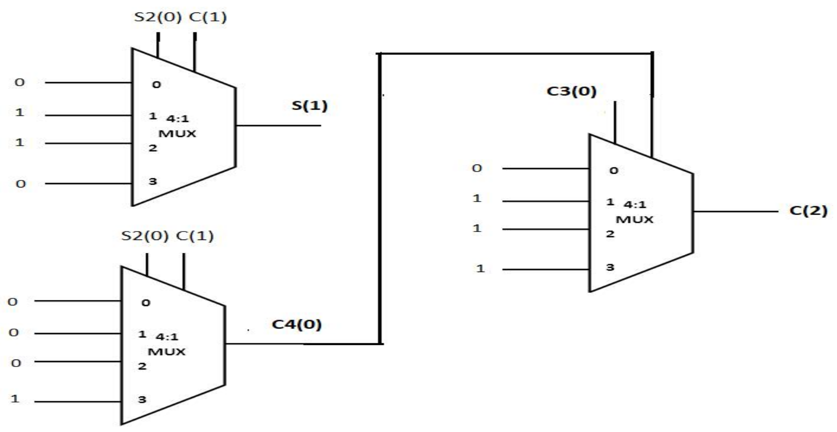

Fig.11. Multiplexer unit for generation of S(1) ,C4(0) and C(2).

Selection lines for the eighth, ninth multiplexers are S2(0) and C(1). The inputs to the eighth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,0 in order to obtain carry S(1). The inputs to the ninth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,0,0,1 in order to obtain carry C4(0). The inputs to the tenth multiplexer at selection input 0,input 1,input 2,input 3 are taken as 0,1,1,1 in order to obtain carry C(2) as shown in Fig.11. The equations for obtaining S(1),C4(0) and C(2) are shown in Eq.(15) , Eq.(16) and Eq.(17). The final architecture for two bit adder using 4:1 multiplexers with input Carry is equal to 'One' is shown in Fig.12.

$$\boldsymbol{S1(0) = \overline{a(0)}.\overline{b(0)}.0 + \overline{a(0)}.b(0).1 + a(0).\overline{b(0)}.1 + a(0).b(0).0 = \overline{a(0)}.b(0) + a(0).\overline{b(0)}} \quad \textbf{(8)}$$

$$\boldsymbol{C1(0) = \overline{a(0)}.\overline{b(0)}.0 + \overline{a(0)}.b(0).0 + a(0).\overline{b(0)}.0 + a(0).b(0).1 = a(0).b(0)} \quad \textbf{(9)}$$

$$\boldsymbol{S(0) = \overline{S1(0)}.\overline{Cin}.0 + \overline{S1(0)}.Cin.1 + S1(0).\overline{Cin}.1 + S1(0).Cin.0 = \overline{S1(0)}.Cin + S1(0).\overline{Cin} = S1(0) \oplus cin = a(0) \oplus b(0) \oplus cin} \quad \textbf{(10)}$$

$$\boldsymbol{C2(0) = \overline{S1(0)}.\overline{Cin}.0 + \overline{S1(0)}.Cin.0 + S1(0).\overline{Cin}.0 + S1(0).Cin.1 = S1(0).Cin = a(0).\,\overline{b(0)}.Cin + \overline{a(0)}.b(0).Cin = (a(0) \oplus b(0)).Cin} \quad \textbf{(11)}$$

$$\boldsymbol{C(1) = \overline{C1(0)}.\overline{C2(0)}.0 + \overline{C1(0)}.C2(0).1 + C1(0).\overline{C2(0)}.1 + C1(0).C2(0).1 = C1(0) + C2(0) = a(0).b(0) + [a(0).\,\overline{b(0)} + a\overline{(0)}.b(0)].Cin = a(0).b(0) + a(0).\,\overline{b(0)}.Cin + \overline{a(0)}.b(0).Cin = a(0)[b(0) + Cin] + \overline{a(0)}.b(0).Cin = a(0)b(0) + a(0)cin + \overline{a(0)}.b(0).Cin = a(0)b(0) + \left[a(0) + \overline{a(0)}.b(0)\right]Cin = a(0)b(0) + [a(0) + b(0)]Cin = a(0)b(0) + a(0)Cin + b(0)Cin.} \quad \textbf{(12)}$$

$$\boldsymbol{S2(0) = \overline{a(1)}.\overline{b(1)}.0 + \overline{a(1)}.b(1).1 + a(1).\overline{b(1)}.1 + a(1).b(1).0 = \overline{a(1)}.b(1) + a(1).\overline{b(1)} = a(1) \oplus b(1)} \quad \textbf{(13)}$$

$$\boldsymbol{C3(0) = \overline{a(1)}.\overline{b(1)}.0 + \overline{a(1)}.b(1).0 + a(1).\overline{b(1)}.0 + a(1).b(1).1 = a(1).b(1)} \quad \textbf{(14)}$$

$$\boldsymbol{S(1) = \overline{S2(0)}.C\overline{(1)}.0 + \overline{S2(0)}.C(1).1 + S2(0).C\overline{(1)}.1 + S2(0).C(1).0 = \overline{S2(0)}.C(1) + S2(0).C\overline{(1)} = S2(0) \oplus C(1) = a(1) \oplus b(1) \oplus C(1)} \quad \textbf{(15)}$$

$$\boldsymbol{C4(0) = \overline{S2(0)}.C\overline{(1)}.0 + \overline{S2(0)}.C(1).0 + S2(0).C\overline{(1)}.0 + S2(0).C(1).1 = (a(1) \oplus b(1)).C(1)} \quad \textbf{(16)}$$

$$\boldsymbol{C(2) = \overline{C3(0)}.\overline{C4(0)}.0 + \overline{C3(0)}.C4(0).1 + C3(0).\overline{C4(0)}.1 + C3(0).C4(0).1 = C3(0) + C4(0) = a(1).b(1) + [a(1).\,\overline{b(1)} + \left(\overline{a(1)}.b(1)\right].C(1) = a(1).b(1) + a(1).\,\overline{b(1)}.C(1) + \overline{a(1)}.b(1).C(1) = a(1)[b(1) + C(1)] + \overline{a(1)}.b(1).C(1) = a(1)b(1) + a(1)C(1) + \overline{a(1)}.b(1).C(1) = a(1)b(1) + \left[a(1) + \overline{a(1)}.b(1)\right]C(1) = a(1)b(1) + [a(1) + b(1)]C(1) = a(1)b(1) + a(1)C(1) + b(1)C(1).} \quad \textbf{(17)}$$

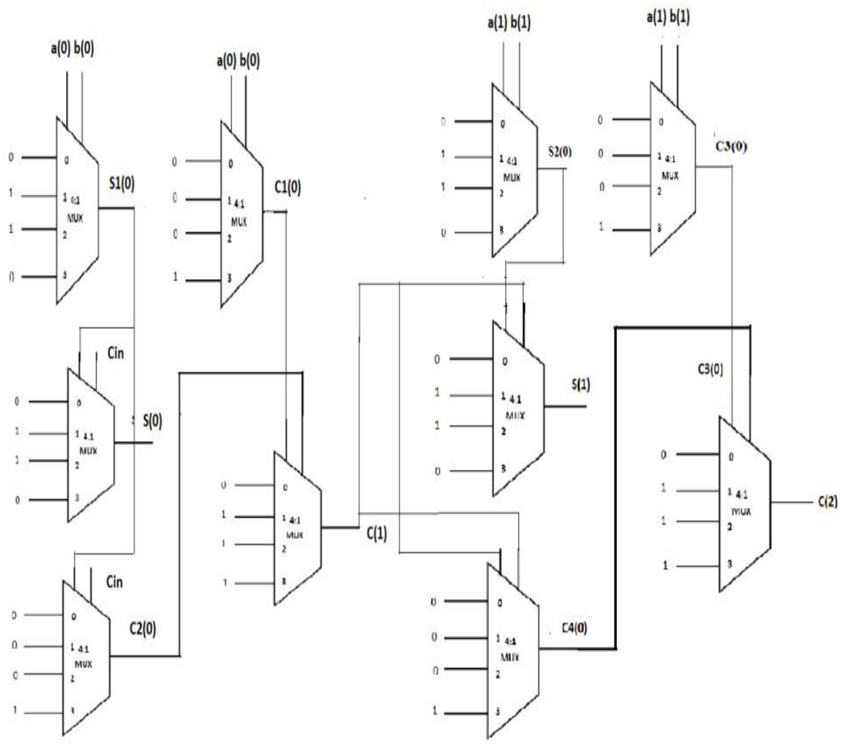

Fig.12. Two bit adder using 4:1 Multiplexers with input Carry is equal to 'One'

2.3 Simulation Results

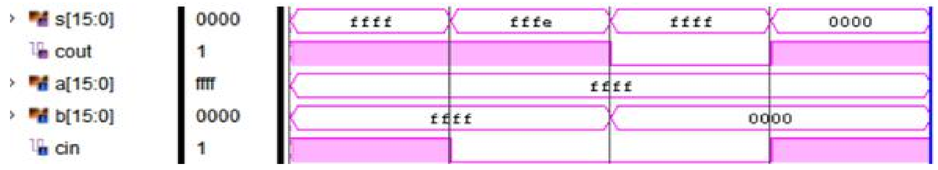

Fig.13.Simulation results for 16 bit CSLA using Multiplexed Based Adders

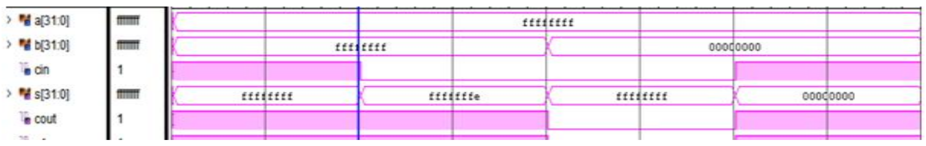
a[31:0] ffffffff
b[31:0] ffffffff 00000000
cin 1
s[31:0] ffffffff ffffffff fffffffe ffffffff 00000000
cout 1

Fig.14.Simulation results for 32 bit CSLA using Multiplexed Based Adders

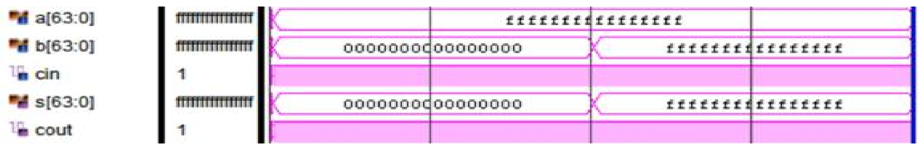
a[63:0] ffffffffffffffff
b[63:0] 0000000000000000 ffffffffffffffff
cin 1
s[63:0] 0000000000000000 ffffffffffffffff
cout 1

Fig.15.Simulation results for 32 bit CSLA using Multiplexed Based Adders

QThe proposed architecture and internal sub-modules are simulated and synthesized using Vivado2017.3. The proposed architecture for CSLA for 16 bit is simulated for which inputs are a(15:0),b(15:0) and cin. The resultant sum s(15:0) and cout for various combinations are as shown in the Fig.13. The proposed architecture for CSLA for 32 bit is simulated for which inputs are a(31:0),b(31:0) and cin. The resultant sum s(31:0) and cout for various combinations are as shown in the Fig.14. The proposed architecture for CSLA for 64 bit is simulated for which inputs are a(63:0),b(63:0) and cin. The resultant sum s(63:0) and cout for various combinations are as shown in the Fig.15.

3. Results & Discussion

The proposed adder and the conventional architectures for data width ranging from 16 bit to 64 bit are synthesized in cadence Encounter RTL compiler using the slow library of 90nm standard cell technology. The proposed architecture consumes less area and power when compared to conventional architectures for various word sizes. 2 bit Conventional RCA with input carry is equal to 'zero' consists of 1 Half adder and 1 Full adder. Each full adder is designed using 2 Half adders and 1 OR gate. Thus 2 bit RCA with input carry is equal to 'zero' is designed using 3 Half adders and 1 OR gate. Half adder consumes 12 units of cell area since multiplexer based XOR gate consumes 6 units of area and AND gate consumes 6 units of area. Similarly 6 units of cell area for OR gate during synthesis. Thus the overall cell area occupied by 2 bit RCA with input carry is equal to 'zero' is 12*3+5=41 units of cell area. 2 bit RCA with input carry is equal to 'one' consists of 2 Full adders. Each full adder is designed using 2 Half adders and 1 OR gate. Thus 2 bit RCA with input carry is equal to 'one' is designed using 4 Half adders and 2 OR gates.Thus the overall cell area occupied by 2 bit RCA with input carry is equal to 'one' is 12*4+2*5=58 units of cell area. Similarly multiplexer based 2 bit adder with input carry is equal to 'cin' occupies 39 units of cell area during synthesis since multiplexer based XOR gate consumes 6 transistors and multiplexer based OR gate and AND gate consumes 4 units of cell area. Thus the overall area is reduced for multiplexer based adder when compared to RCA. Table I indicates the cell area and delay results for both Normal and Modified Architectures. The area calculations are evaluated in terms of the cell area. Area for 16 bit Proposed architecture is reduced by 44.8877658 %,19.7007087%,10.4737132% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC architectures and increased by 0.7538689 % when compared to SQRTKSA-BEC architecture. Area for 32 bit Proposed architecture is reduced by 53.98997%, 26.80797 %,15.83536% and 3.366533 % when compared to Conventional SQRT CSLA, SQRT-KSA,SQRTCSLA-BEC ,SQRTKSA-BEC architectures. Area for 64 bit Proposed architecture is reduced by 59.35164%, 39.86908%,18.79677% and 14.7444% when compared to Conventional SQRT CSLA, SQRT-KSA,SQRTCSLA-BEC ,SQRTKSA-BEC architectures. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of cell area is shown in Fig.16. Performance evaluation for the proposed architecture with respect to different CSLA architectures in terms of delay is shown in Fig.17.

Table I Comparison of Area and Delay for different CSLA Architectures

Adder /Parameter	Area(µm$^{2)}$			Delay (psec)		
Word Size	16-Bit	32-Bit	64-Bit	16-Bit	32-Bit	64-Bit
SQRT-CSLA[20]	879.518	1869.543	3869.273	3832	4644	6029
SQRT-KSA[21]	726.624	1539.535	3396.21	3840	4652	6038
SQRTCSLA- BEC [20]	670.613	1406.32	2884.546	3817	4627	6010
SQRTKSA-BEC [21]	602.492	1254.94	2786.149	3836	4660	6255
PROPOSED	607.034	1214.068	2428.135	4817	7641	13291

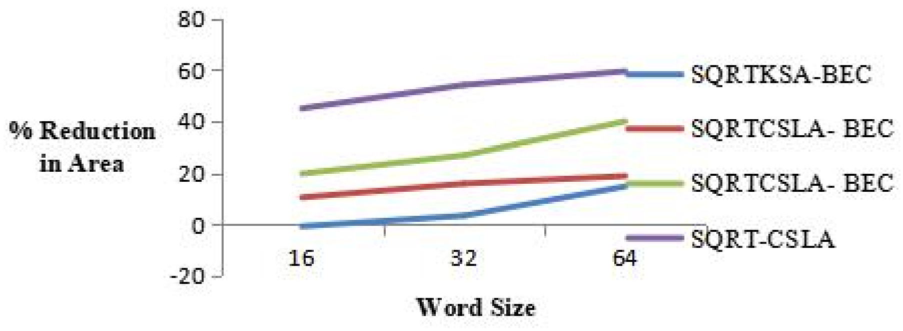

Fig.16.Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of cell area.

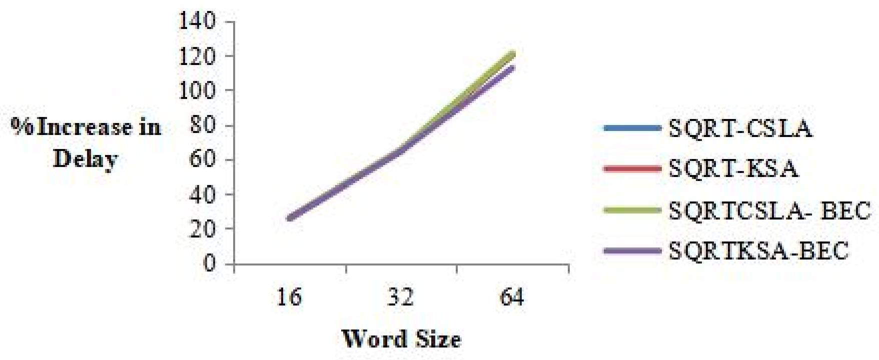

Fig.17. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of delay.

Table II & III indicates the Leakage Power, Dynamic Power and Total Power for different CSLA Architectures. The total Power is the sum of Leakage power and Dynamic Power. Leakage Power for 16 bit Proposed architecture is reduced by 97.13535%,37.499974%,38.816802%, 8.8625051% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. Leakage Power for 32 bit Proposed architecture is reduced by 109.7318%,42.44941%,43.31021%,10.06154% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. Leakage Power for 64-bit proposed architecture is reduced by 117.1699%, 68.05759%, 45.53647%, and 33.67389 % when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. The usage of multiplexer based adders in the Proposed architecture reduces the leakage power since the probability of a getting a direct path from VDD to ground is very less [16]-[19]. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of Leakage Power is shown in Fig.18.Dynamic Power for 16 bit Proposed architecture is reduced by 36.11468%,30.37478 %,15.39073%,16.47633% when compared to Conventional SQRT CSLA, SQRT-KSA,SQRTCSLA-BEC and SQRTKSA-BEC architectures. Dynamic Power for 32 bit Proposed architecture is reduced by 52.25854 %, 45.0262 %, 27.16637 %, 26.85307% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. Dynamic Power for 64-bit proposed architecture is reduced by 56.74388%, 69.17959%, 27.9172%, and 47.47071% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of Dynamic Power is shown in Fig.19.Total Power for 16 bit Proposed architecture is reduced by 43.8077%,31.27307%,18.34276%,15.51644% when compared to Conventional SQRT CSLA, SQRT-KSA,SQRTCSLA-BEC and SQRTKSA-BEC architectures. Total Power for 32 bit Proposed architecture is reduced by 59.45213 %, 44.70368 %, 29.187 % and 24.75137% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. Total Power for 64-bit proposed architecture is reduced by 64.34607%, 69.03764%, 30.13349% and 45.73408% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC and SQRTKSA-BEC architectures. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of total Power is shown in Fig.20.

Table II Comparison of Leakage Power, Dynamic Power for different CSLA Architectures

Parameter	Leakage Power (nW)			Dynamic Power (nW)		
Data Width	16-Bit	32-Bit	64-Bit	16-Bit	32-Bit	64-Bit
SQRT-CSLA [20]	6638.803	14126.01	29253.98	31775.03	71677.62	146695.8
SQRT-KSA [21]	4630.5	9594.358	22638.28	30435.09	68272.91	158334.3
SQRTCSLA-BEC [20]	4674.846	9652.335	19604.56	26937.17	59865.17	119717.1

SQRTKSA-BEC [21]	3666.094	7412.946	18006.61	27190.59	59717.68	138017.1
PROPOSED	3367.637	6735.274	13470.55	23344.31	47076.26	93589.51

Table III Comparison of Total Power for different CSLA Architectures

Parameter	Total Power (nW)		
Data Width	16-Bit	32-Bit	64-Bit
SQRT-CSLA [20]	38413.83	85803.64	175949.8
SQRT-KSA [21]	35065.59	77867.26	180972.6
SQRTCSLA- BEC [20]	31611.65	69517.5	139321.6
SQRTKSA-BEC [21]	30856.69	67130.62	156023.7
PROPOSED	26711.95	53811.53	107060.6

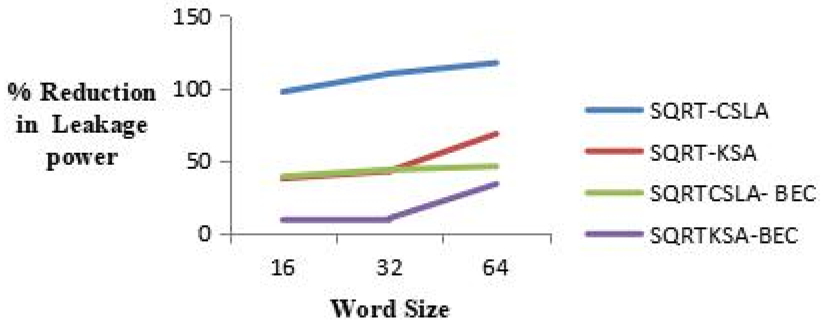

Fig.18. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of Leakage Power.

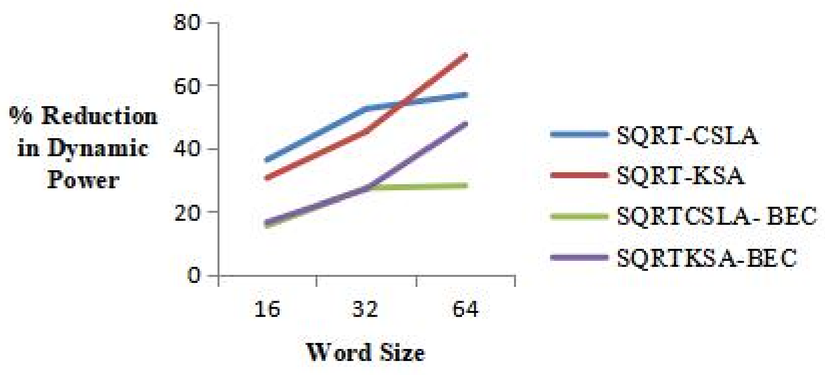

Fig.19. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of Dynamic Power.

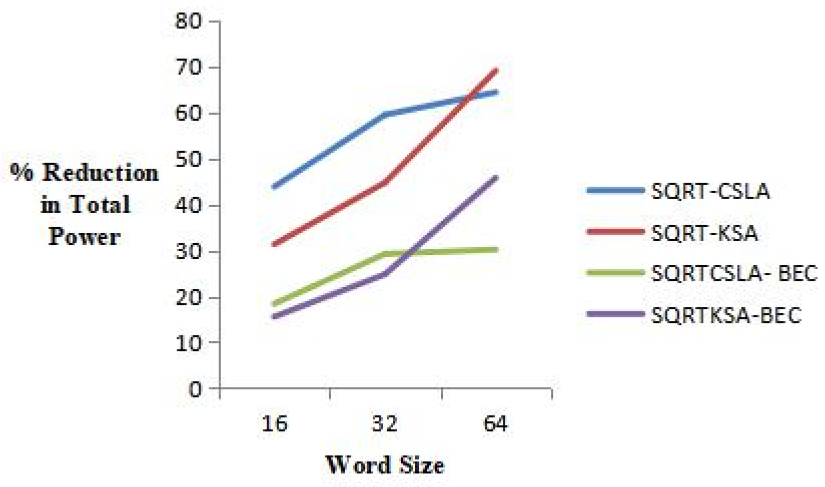

Fig.20. Performance evaluation for the proposed architecture with respect to different CSLA Architectures in terms of Total Power.

4. Conclusion

In this paper, a Low power and area efficient VLSI Architecture are Proposed. ASIC implementation of the proposed adder for bit depths ranging from 16 bits to 64 bits proved that the architecture is efficient in terms of area and power. Area for the proposed adder for 64 bit is reduced by 59.35164 %, 39.86908 %, 18.79677 %, 14.7444 % when compared to Conventional SQRT CSLA, SQRT-KSA,SQRTCSLA-BEC,SQRTKSA-BEC architectures. Leakage Power for the proposed adder for 64 bit is reduced by 117.1699%, 68.05759%, 45.53647%, 33.67389% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC, SQRTKSA-BEC architectures. Dynamic Power for the proposed adder for 64 bit is reduced by 56.74388%, 69.17959%, 27.9172%, 47.47071% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC, SQRTKSA-BEC architectures. Total Power for the proposed adder for 64 bit is reduced by 64.34607%, 69.03764%, 30.13349%, 45.73408% when compared to Conventional SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC, SQRTKSA-BEC architectures. Thus the Proposed adder is area efficient and power efficient when compared to SQRT CSLA, SQRT-KSA, SQRTCSLA-BEC, SQRTKSA-BEC architectures which is apt for designing area and power efficient FIR filter

References

[1]R. W. Stewart et al., ‘‘A low-cost desktop software defined radio design environment using MATLAB, simulink, and the RTL-SDR,’’ IEEE Commun. Mag., vol. 53, no. 9, pp. 64–71, Sep. 2015. DOI: 10.1109/MCOM.2015.7263347

[2]XIN CAI , MINGDA ZHOU ,XINMING HUANG “ Model-Based Design for Software Defined Radio on an FPGA”.pp.8276-8283,vol.5,IEEEACCESS, DOI: 10.1109/ACCESS.2017.2692764

[3]N. Kumar, K. R. Nalluri and G. Lakshminarayanan, "Design of area and power efficient digital FIR filter using modified MAC unit," 2015 2nd International Conference on Electronics and Communication Systems (ICECS) Pp: 884 – 887, DOI: 10.1109/ECS.2015.7125041.

[4]A. P. Vinod, E. M-k. Lai, S. Emmanuel “Low power and high-speed implementation of FIR filters for software defined radio receivers”, 2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, DOI:10.1109/PIMRC.2006.254369.

[5]S. Mirzaei, A. Hosangadi and R. Kastner, “FPGA Implementation of High Speed FIR Filters Using Add and Shift Method”. In IEEE Proc. International Conference on Computer Design, San Jose, CA, pp. 308- 313, Oct. 2006,DOI: 10.1109/ICCD.2006.4380833.

[6] R. I. Hartley, “Subexpression sharing in filters using canonical signed digit multipliers”, IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 10, pp. 677-688, 1996, DOI: 10.1109/82.539000.

[7] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, “A new algorithm for elimination of common subexpressions”, IEEE Trans. Computer-Aided Design, vol. 18, no. 1, pp .58-68, 1999, DOI: 10.1109/43.739059.

[8] C. Y. Yao, H. H. Chen, C. J. Chien and C. T. Hsu, “A novel common subexpression elimination method for synthesizing fixed point FIR filters”, IEEE Trans. Circuits Syst. I, vol. 51, no. 11, pp. 2215-2221, 2004, DOI: 10.1109/TCSI.2004.836853.

[9] Abhijit Chandra , Sudipta Chattopadhyay “Design of hardware efficient FIR filter: A review of the state-of-the-art approaches” Engineering Science and Technology, an International Journal,pp.212-226. University. http://dx.doi.org/10.1016/j.jestch.2015.06.006.

[10] K. Muhammad and K. Roy, “A graph theoretic approach for synthesizing very low-complexity high-speed digital filters”, IEEE Trans. Computer-Aided Design, vol. 21, no. 2, pp. 204-216, 2002, DOI: 10.1109/43.980259.

[11] O. Gustafsson and L. Wanhammar, “A novel approach to multiple constant multiplication using minimum spanning tree”. In Proc. IEEE Mediterranean Electrotechnical Conf., Dubrovnik, Croatiamum spanning trees”. In Proc. IEEE Midwest Symp. Circuits Syst., Tulsa, OK, vol. 3, pp. 652-655, 2002, DOI: 10.1109/MWSCAS.2002.1187124.

[12] H. Ohlsson, O. Gustafsson and L. Wanhammar, “Implementation of low-complexity FIR filters using a minimum spanning tree”. vol. 1, pp. 261-264, May 2004, DOI: 10.1109/MELCON.2004.1346826.

[13] G. Dempster and M. D. Macleod, “Use of minimum-adder multiplier blocks in FIR digital filters”, IEEE Trans. Circuits Syst.-II, vol. 42, no. 9, pp. 569-577, 1995, DOI: 10.1109/82.466647.

[14] Parhami, Computer Arithmetic, Algorithms and hardware Designs, Oxford University.Press, London, U.K, 2000.

[15] H. R. Lee. C. W. Jen, and C. M. Liu, “A new hardware-efficient architecture for programmable FIR filters”, IEEE Trans. Circuits Syst. II, vol. 43, no. 9, pp. 637-644, 1996, DOI: 10.1109/82.536760.

[16] S. Mitra, L. J. Avya and E. J. McCluskey, "Efficient multiplexer synthesis techniques," in IEEE Design & Test of Computers, vol. 17, no. 4, pp. 90-97, Oct.-Dec. 2000.
doi: 10.1109/54.895009

[17] M. Faust and C. H. Chang, “Minimal logic depth adder tree optimization for multiple constant multiplication”. In Proc. IEEE Int. Symp. On Circuits Syst.(ISCAS), Paris, France, pp. 457-460, June 2010, DOI: 10.1109/ISCAS.2010.5537658.

[18] Anubhuti Mittal; Ashutosh Nandi; DishaYadav “Comparative study of 16-order FIR filter design using different multiplication techniques” IET Circuits, Devices & Systems,pp.196-200,Issue-3,2017, DOI: 10.1049/iet-cds.2016.0146.

[19] Yingtao Jiang, Abdulkarim Al-Sheraidah, Yuke Wang, Edwin Sha, and Jin-Gyun Chung “A novel multiplexer based low power full adder” IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 51, NO. 7, JULY 2004, DOI: 10.1109/TCSII.2004.831429.

[20] B.Ram kumar,Harish M kittur “Low power and area efficient carry select adder”pp.371-375, IEEE Transactions on Very Large Scale Integration (VLSI) Systems ,Volume: 20, Issue: 2, Feb. 2012, DOI: 10.1109/TVLSI.2010.2101621.

[21] K.Bala Sindhuri; N.Udaya Kumar; A.Durga Prasad; K.V.S.S.Lalitha Siva Jyothi,”Area Efficient VLSI Architecture for square root Carry Select Adder using Kogge-Stone Adder and Binary to Excess-1 Converter” 2017 Innovations in Power and Advanced Computing Technologies (i-PACT)