Multidimensional Systems and Signal Processing
https://doi.org/10.1007/s11045-021-00772-1

# A novel parallel prefix adder for optimized Radix-2 FFT processor

Garima Thakur[1] · Harsh Sohal[1] · Shruti Jain[1]

Received: 20 July 2020 / Revised: 18 February 2021 / Accepted: 22 February 2021

**Abstract**
The Fast Fourier Transform (FFT) is the basic building block for DSP applications where high processing speed is the critical requirement. Resource utilization and the number of computational stages in Radix-2 FFT structure implementation can be minimized by improving the performance of utilized multiplier and adder blocks. This work proposes a hardware design of an efficient Radix-2 FFT architecture using optimized multiplier and novel Parallel prefix (PP) adder. The designed FFT architecture results in low power and area with an increase in operation speed in comparison to the existing architectures. Our proposed Radix 2 FFT implementation results in 18.218 ns (6.030 ns logic delay and 12.118 ns router delay) in comparison with 24.003 ns delay for Wallace multiplier using Kogge Stone PP adder (M1P1), 24.162 ns delay for Wallace multiplier using Brent Kung PP adder (M2P2), 24.889 ns delay for Wallace multiplier using Landner Fischer PP adder (M3P3) and 22.827 ns delay for Wallace multiplier using Han Carlson PP adder (M4P4) algorithm. The proposed adder and hence the FFT processor can be used in different applications where high speed, low power, and less area is required. The novel PP architecture results in a 20.19% improvement in comparison with other state-of-art techniques.

**Keywords** PP adders · Wallace multiplier · Radix 2 structure · FFT algorithm

## 1 Introduction

Digital signal processing (DSP) plays a vital role in the everyday life of humans of all ages (Kogge & Stone, 1973). In signal processing, the Fast Fourier Transform (FFT) is the most usually used algorithm. FFT algorithm is utilized for improving the execution of Discrete Fourier Transform (DFT). The FFT calculations are well suitable for DSP applications where high-speed information is the foremost critical requirement. Nowadays, everyone demands portable gadgets with high-speed, less power consumption, and maximum components in a small area. To fulfill all these requirements it is important to design an algorithm that solves all these problems. Though these algorithms are fast, the actual

✉ Garima Thakur
garimathakur1994@gmail.com

[1] Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Solan, India

Published online: 15 March 2021

computation time is decided by the system clock at which the processor is operating. The processor mainly involves a hardware approach in designing Very Large Scale Integration (VLSI) architecture for implementing various transforms to optimize the performance metrics such as system throughput, hardware utilization (area), and power consumption. In the FFT processor, the complex calculation part is solved with the help of the Radix-2 algorithm. In this algorithm, different arithmetic operations like addition and multiplication are performed. Addition operation can be performed in two ways: serial or parallel. Because of the bit by bit operation, serial adders are slow, consume more power, and take more time for implementation where parallel adders are fast because bits are added simultaneously. It is important to design high-speed and less power consumption parallel prefix (PP) adders and multipliers. Serial adders like full adder (FA), ripple carry adder (RCA) uses more power and area, where PP adders like Kogge-Stone (KS) (Kogge & Stone, 1973), Brent-Kung (BK) (Brent & Kung, 1982), Landner-Fischer (LF) (Ladner & Fisher, 1980) and Han Carlson (HC) (Hans & Carlson, 1987) adders are widely used for the implementation of FFT algorithm.

### 1.1 Review of literature

Singh and Nandi (2017) discussed various FFT construction issues and designed efficient butterfly processing elements (BPE). BPE is constructed with the help of Vedic multiplier and Carry look-ahead adder (CLA) but the speed of the processor is not that much optimized. Dimitrakopoulos and Nikolos (2005) explained the systematic methodology for PP Ling adder which provides high-speed datapath with reduced delay and fan-out requirements. Waters and Swartzlander (2010) explained the modified Wallace multiplier which reduces 80 percent half the adders compared to the conventional Wallace multiplier. Akbar and Mosleh (2019) designed the two 8 bit reversible Wallace multiplier. The only difference between these two circuits is that one has the lowest delay and the other is better regarding many factors like fixed inputs, garbage output, and cost. Hussain et al. (2019) designed a modified multiplier that has low power by using the Wallace tree algorithm. This modified multiplier has a better power delay product (PDP), power consumption, and energy-delay product (EDP). Thakur et al. (2018a) designed a modified Wallace multiplier using Kogge stone adder(KSA) which results in a speed and power-efficient block. Kavya et al. (2017) designed an optimized radix -2 butterfly and then used part of this butterfly in designing a radix-4 butterfly which is power efficient. Santhosh and Thomas (2013) explained that radix $2^2$ requires less hardware in comparison to the radix 2 FFT algorithm. Jayakumar and Logashanmugam (2016) designed a proposed architecture of combined Radix 2, 4, and 8 which offers a decrease in slices, LUTs, delay, and power consumption compared to the conventional Radix 2, 4, and 8 architectures. Also, the proposed architecture reduces the computational stages and gives better performance. Thakur et al. (2020a) designed and analyzed the high speed parallel prefix adder for digital circuit design applications. Authors' main concern is to improve the speed for fast computation of applications. S. Liu and D. Liu (2019) proposed a CORDIC algorithm which is further used in memory-based FFT processor for increases its speed. The processor is synthesized using SMIC 65-nm technology. Tang et al. (2014) also proposed a high speed FFT processor. But the processor in it doesn't support the nonpower-of-two DFTs. Processors in Chen et al. (2015) is used for minimization of number of TF multiplication with the help of prime factor algorithm (PFA). PFA cannot access the parallel I/O data and for communication based applications (Future 5G), throughput is restricted which doesn't make it feasible to use. By using single-table approximation method (STAM) in Shih et al. (2018) 46 $2^m3^n5^k$ points is supported by SDF processor. But the SDF processor also faces limitation of single-path pipelined architecture which restricts the throughput. Akhil et al. (2020) uses CORDIC multiplier and KSA for the implementation of 8 point FFT processor. But the implementation requires more hardware for its utilization which increases the overall cost of the circuit. There is still a challenge to design a high speed, less area utilization and low power consumption processor. Thakur et al. (2020b) implemented Brent-Kung speculative adder for high speed application, but it requires more area for its processing which increases the cost of design.

### 1.2 Problem formulation

In the conventional circuits of multiplier and adders; area utilization, power consumption, and delay are not optimized. For implementation of high speed FFT processor efficient arithmetic circuits are designed. In this paper, an optimized multiplier circuit using an efficient adder is proposed and designed. The novelty of this work lies in the hardware implementation of an efficient Radix-2 FFT structure using a multiplier, and proposed novel PP adder which leads to low consumption of power and utilizes less area with high speed in comparison with the design of FFT using existing PP adders. Our proposed design incorporates a reconfigurable complex multiplier and parallel prefix adder for the power-of-2 radix style of FFT processors. Particularly, this paper presents an effective strategy of actualizing adders and multipliers on FPGAs that can be utilized as a fundamental building block to form different sorts of DSP filters.

The paper is organized as follows: In Sect. 2 materials and methods used in this paper are explained and implementation of Radix-2 structure using conventional PP adders and Wallace multiplier are discussed. In the end, an efficient Radix 2 DIT FFT structure is proposed using novel PP adder architecture which is followed by a conclusion in Sect. 3.

## 2 Methodology

With the increasing demand for the execution of computerized handling methods, DSP algorithms were designed that require a high level of computational throughput, especially for real-time application. This increase in demand is combined with the VLSI and leads to the development of VLSI DSP. FFT computing is the most useful technique in DSP. FFT computing requires different types of operations like storage, multiplication, and addition. The flow diagram of the proposed Radix-2 structure using different PP adders and multiplier is shown in Fig. 1.

The different existing PP adders and proposed novel PP adder is used to implement an efficient multiplier to reduce the complexity at the final stage of addition. Authors have implemented Radix 2 FFT structure using four different conventional PP adders, and a multiplier circuit. To overcome the disadvantages of the existing circuit, the author has proposed a novel PP adder and an optimized multiplier which results in high speed, and low power consumption. Anon Radix 2 FFT processor is realized using novel PP adder and the results are compared based on area, delay, and power. The hardware simulations were carried out in Xilinx ISE 14.7 design tool targeting the Spartan 6 FPGA family. Register Transfer Logic (RTL) and Technology design is obtained by viewing RTL Schematic and Technology Schematic. The evaluation environment consists of a PC with an Intel 2.4 GHz

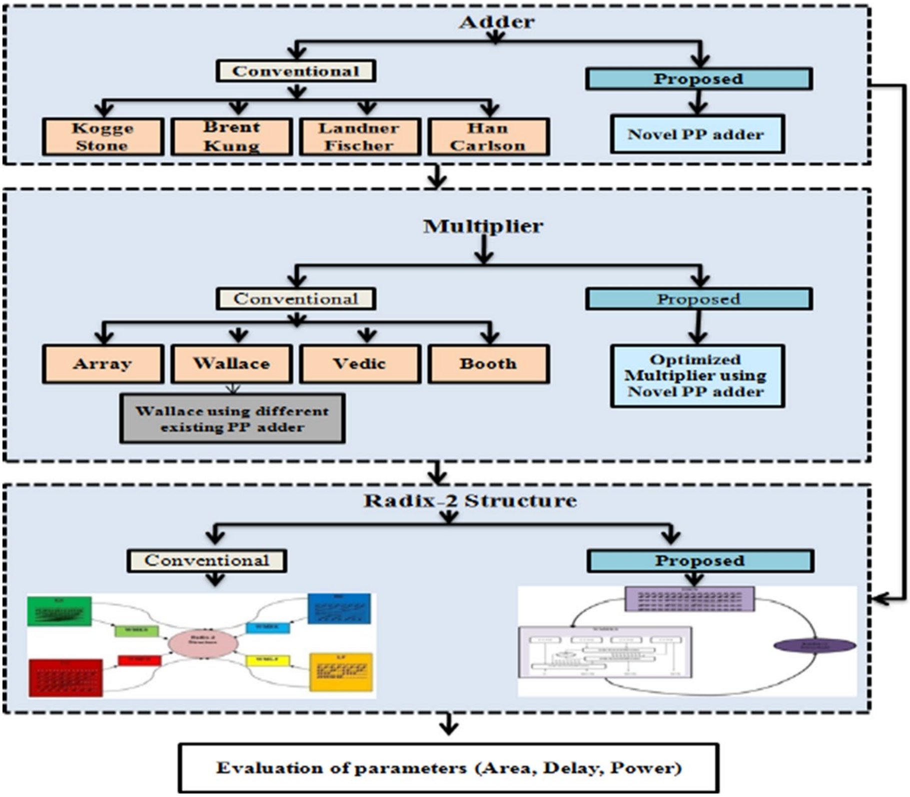

**Fig. 1** Flow diagram of proposed Radix-2 DIT FFT architecture

processor and 16 GB RAM. The structure of two-point radix components has one complex duplication and two complex augmentations. The prime objective of this paper is to implement an FFT structure with high computational speed and low power consumption with less area utilization. For efficient execution of Radix-2 FFT architecture different parameters are calculated which include IOBs, delay, power consumption, energy utilized. The parameters of the proposed architecture are compared with other state-of-art work. To check the efficiency of the proposed architecture comparison is important.

## 2.1 Implementation of FFT algorithm using conventional arithmetic structures

FFT algorithm can be used in many applications like DFT or signal processors. DFT is divided as Decimation In Time (DIT) and Decimation In Frequency (DIF). In DIT algorithm, initially multiplier operation is performed then adder but in DIF, adder operation is performed initially then multiplication. In this paper Radix-2 structure is implemented using DIT algorithm because in DIT the input is bit reversed while the output is in natural order whereas in DIF the input is in natural order but output is bit reversed order. Radix-2 structure is used in FFT to solve the complex calculations (Narvekar et al., 2019; Thakur et al., 2018b, 2018c). Figure 2 shows the flow diagram of Radix-2 FFT structure. The input *A* and *B* were initialized which used in the multiplier block.

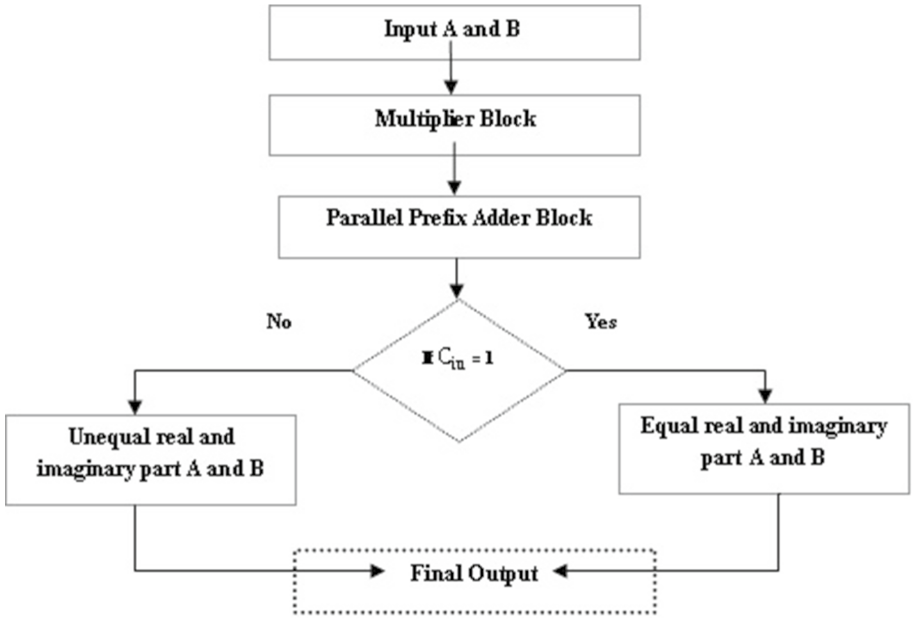
Input A and B
Multiplier Block
Parallel Prefix Adder Block
No
Yes
Cin = 1
Unequal real and imaginary part A and B
Equal real and imaginary part A and B
Final Output

**Fig. 2** Flow diagram of Radix 2 FFT structure

Multiplier block is followed by PP adder block in which addition of partial product generation is performed. The final output depends on the value of $C_{in}$, if $C_{in}$=1 then equal real and imaginary part is calculated otherwise unequal real and imaginary part is calculated.

For Radix-2 FFT structure, complex parts are solved by using DFT as expressed by (1).

$$X_P = \sum_{n=0}^{N-1} x_n\, e^{-j\left(2\Pi/N\right)np} \tag{1}$$

(1) can be splitted into two parts: even and odd. Even part is expressed by (2) whereas odd part is expressed by (3).

$$= \sum_{n=0}^{\left(N/2\right)-1} x_{2n}\, e^{-j\left(2\Pi/N\right)(2n)p} \tag{2}$$

$$= \sum_{n=0}^{\left(N/2\right)-1} x_{2n+1}\, e^{-j\left(2\Pi/N\right)(2n+1)p} \tag{3}$$

By summing (2) and (3) we get

$$\begin{aligned} X_P &= \sum_{n=0}^{(N/2-1)} x_{2n}\, e^{-j\left(2\Pi/N\right)(2n)p} + \sum_{n=0}^{(N/2-1)} x_{2n+1} e^{-j\left(2\Pi/N\right)(2n+1)p} \\ &= A_p + W^p B_p \end{aligned} \tag{4}$$

When frequency is p+$N/2$ then (1) is expressed by (5).

$$\begin{aligned} X_{P+N/2} &= \sum_{n=0}^{(N/2-1)} x_{2n} e^{-j\left(2\Pi/N/2\right)np} - e^{-j\left(2\Pi/N\right)p} \sum_{n=0}^{(N/2-1)} x_{2n+1} e^{-j\left(2\Pi/N\right)np} \\ &= A_p - W^p B_p \end{aligned} \tag{5}$$

This basic structure can be used in many applications such as signal processing, communication, DIP. To design an efficient Radix-2 DIT FFT structure, the most important building blocks (adders and multipliers) are required.

### 2.1.1 Conventional PP adders

PP adders are faster than serial adders because the serial addition is performed in sequential form (Thakur et al., 2018b). PP adders provide a good theoretical basis to make a wide range of design trade-off in terms of power consumption, speed and area occupied. Adders are one of the most frequently used circuit and their regularity makes them good candidates for VLSI synthesis, which can also be used to assess design trade-offs. The two binary numbers, $A$ ($a_{\mathrm{m}-1}$, $a_{\mathrm{m}-2}$, … $a_0$) and $B$ ($b_{\mathrm{m}-1}$, $b_{\mathrm{m}-2}$, … $b_0$) were added which results in the form of sum ($s_{\mathrm{m}-1}$, $s_{\mathrm{m}-2}$, …$s_0$) and carry ($c_{\mathrm{i}}$).

Figure 3 shows the flow diagram of 16 bit PP adder. Prefix adders are designed using three stages namely Pre-Computation stage, Prefix network and Post- Computation stage. Each stage performs different operation for calculating generate, propagate, sum and carry.

There are different types of PP adders. In this paper authors have worked on KS, BK, LF and HC PP adder.

**2.1.1.1 Kogge-Stone PP adder (KSPP)** The KSPP adder is designed as shown in Fig. 4 which consists of 5 stages. Stage 1 is a pre-processing stage where generate ($g_i$) and propagate ($p_i$) is calculated. Stage 2 to stage 4 is a prefix network from where final carry $c_i$ is computed by using $g_i$ and $p_i$ obtained from previous stage and sum $s_i$ is computed from the final stage. In the KSPP adder the number of carry stages are log $_2$ $n$ and cells are $n$(log $_2$ $n-1)+1$ with maximum fan-out is 2 (extra wiring).

**2.1.1.2 Brent-Kung PP adder (BKPP)** The BKPP adder is designed as shown in Fig. 5 that consists of 8 stages. Stage 1 is a pre-processing stage used for calculation of generate and propagate Stage 2 to stage 7 is prefix network from where final carry is computed by using generate and propagate that is generated in the previous stage. The sum is computed in the final stage. In the BKPP adder the number of carry stages are 2 log $_2$ ($n-1$) and cells are 2($n-1$)—log $_2$ $n$ with maximum fan-out is 2.

**2.1.1.3 Landner-Fischer PP adder (LFPP)** Figure 6 shows the LFPP adder in which stage 1 is a pre-processing stage where $g_i$ and $p_i$ is calculated. Stage 2 to stage 4 is a prefix

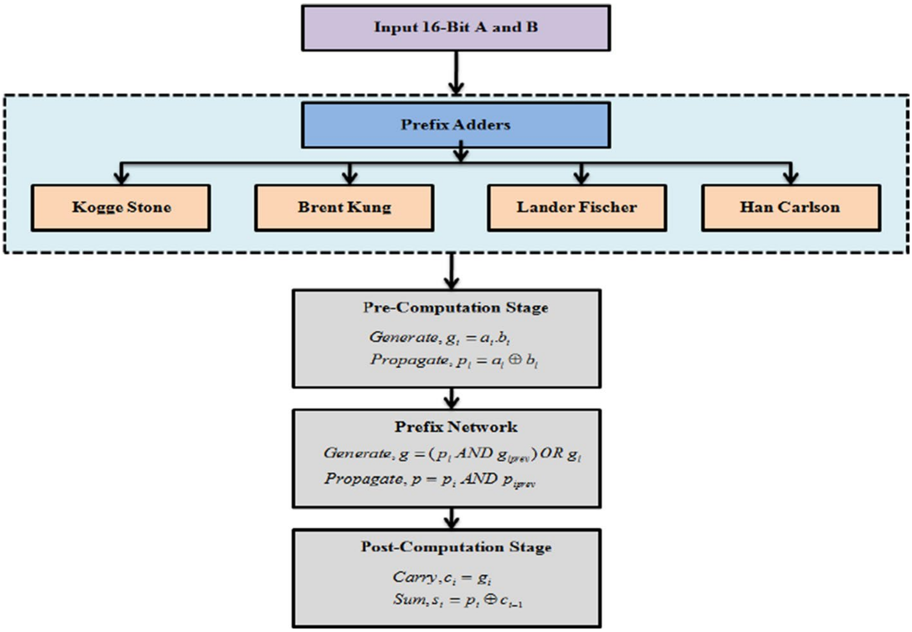
Input 16-Bit A and B
Prefix Adders
Kogge Stone
Brent Kung
Lander Fischer
Han Carlson
Pre-Computation Stage
Generate, $g_i = a_i.b_i$
Propagate, $p_i = a_i \oplus b_i$
Prefix Network
Generate, $g = (p_i\ AND\ g_{iprev})\ OR\ g_i$
Propagate, $p = p_i\ AND\ p_{iprev}$
Post-Computation Stage
Carry, $c_i = g_i$
Sum, $s_i = p_i \oplus c_{i-1}$

**Fig. 3** Flow diagram of Parallel Prefix adder

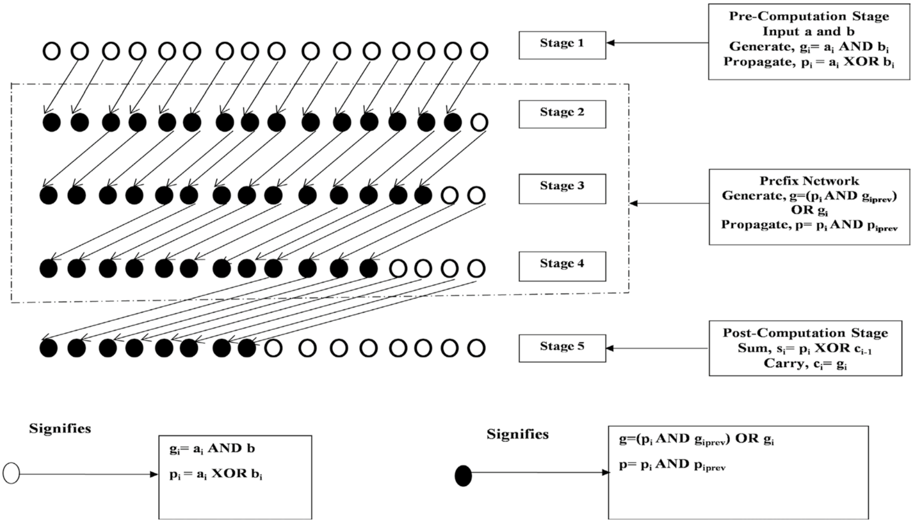
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Pre-Computation Stage
Input a and b
Generate, $g_i$= $a_i$ AND $b_i$
Propagate, $p_i$ = $a_i$ XOR $b_i$
Prefix Network
Generate, g=($p_i$ AND $g_{iprev}$) OR $g_i$
Propagate, p= $p_i$ AND $p_{iprev}$
Post-Computation Stage
Sum, $s_i$= $p_i$ XOR $c_{i-1}$
Carry, $c_i$= $g_i$
Signifies
$g_i$= $a_i$ AND b
$p_i$ = $a_i$ XOR $b_i$
Signifies
g=($p_i$ AND $g_{iprev}$) OR $g_i$
p= $p_i$ AND $p_{iprev}$

**Fig. 4.** 16 bit KSPP adder showing different stages

network stage and the sum is computed in the final stage. In the LFPP adder the number of carry stages are $\log_2$ n and cells are $(n/2)\log_2 n$ with maximum fan-out is $(n/2)$ (large fan-out long wiring).

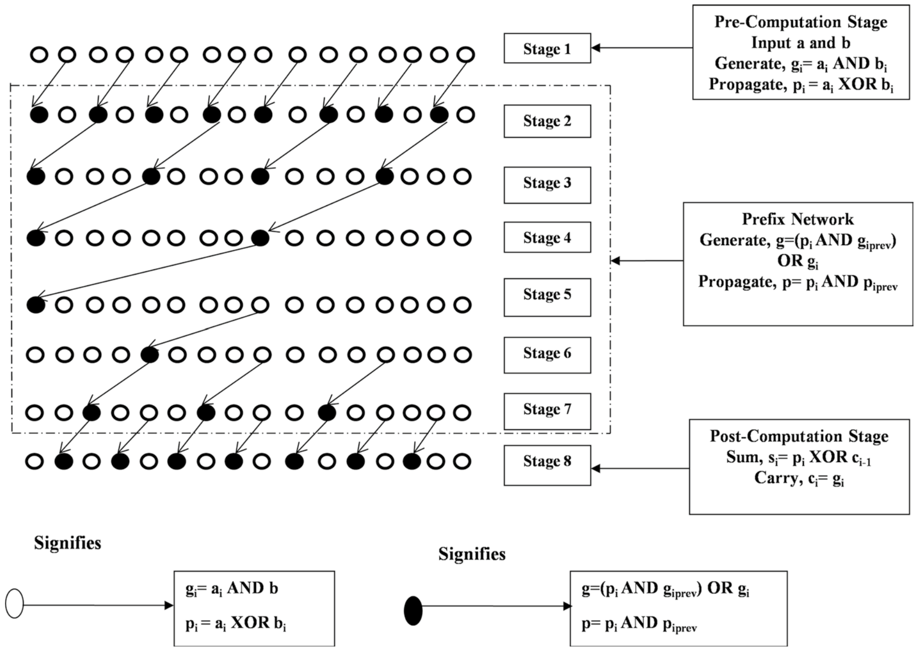

**Fig. 5.** 16 bit BKPP adder showing different stages

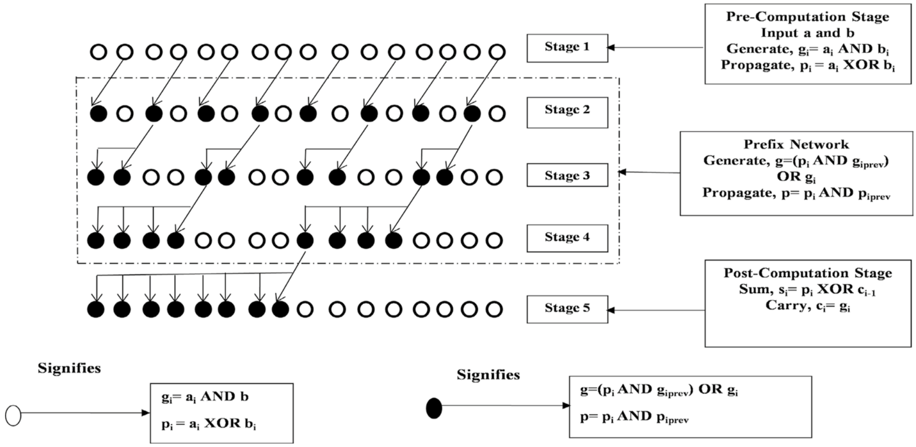
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Pre-Computation Stage
Input a and b
Generate, $g_i$= $a_i$ AND $b_i$
Propagate, $p_i$ = $a_i$ XOR $b_i$
Prefix Network
Generate, g=($p_i$ AND $g_{iprev}$) OR $g_i$
Propagate, p= $p_i$ AND $p_{iprev}$
Post-Computation Stage
Sum, $s_i$= $p_i$ XOR $c_{i-1}$
Carry, $c_i$= $g_i$
Signifies
$g_i$= $a_i$ AND b
$p_i$ = $a_i$ XOR $b_i$
Signifies
g=($p_i$ AND $g_{iprev}$) OR $g_i$
p= $p_i$ AND $p_{iprev}$

**Fig. 6.** 16 bit LFPP adder showing different stages

**2.1.1.4 Han Carlson PP adder (HCPP)** HC adder is combination of BK and KS PP adder. Figure 7 shows the HCPP adder where stage 1 represents the pre-processing stage, stage 2 to stage 5 is prefix network and final sum is computed from the final stage. In the HCPP adder the number of carry stages are $\log_2 n + 1$ with maximum fan-out is 2. Stage 1 and stage 6 of HCPP structure is same as BKPP adder while stage 2 to stage 5 the structure is the same as KSPP adder. In HCPP adder the wiring and gates are reduced but has one extra stage.

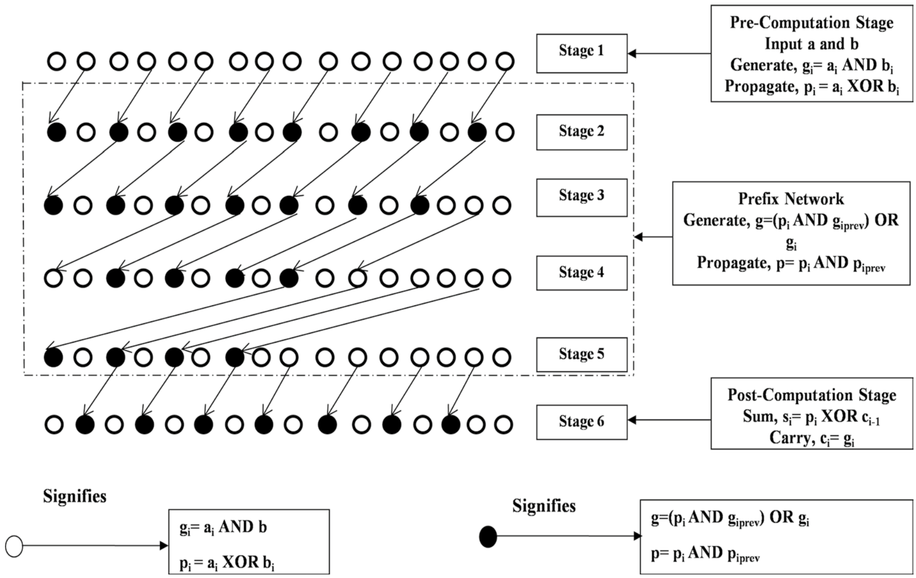

**Fig. 7.** 16 bit HCPP adder showing different stages

All the conventional adders were implemented using Xilinx software in Intel core processor. The results are tabulated in Table 1 which represents the area, speed, power and power delay product.

From the table it is interpreted that HCPP adder yields 12.147 ns delay (5.417 ns logic delay and 6.73 ns router delay) which is less in comparison with other PP adders. The number of slices used in HCPP is only 24 LUTs in comparison with 38 LUTs for KSPP, 33 LUTs for BKPP and 26 LUTs for LFPP. HCPP adder results better among all adders in terms of area and speed. BKPP adder has less delay compared to other PP adder but it takes more area.

### 2.1.2 Implementation of Multiplier with conventional PP adders

In the VLSI circuits the most prominent role is played by multiplier. Dimitrakopoulos and Nikolos (2005) explained different types of multipliers such as Array, Wallace and Vedic. Array multiplier has simple and efficient layout. Add/Shift algorithm is used for

**Table 1** Different types of 16 bit PP adders

| Adder | | No. of bonded IOB | | Number of Slice LUTs | | Delay(ns) | Total Power (W) | | Power Delay Product (Wns) |
|---|---|---|---|---|---|---|---|---|---|
| | | I=Buf | O-Buf | Logic | Router | | Leakage | IOs | |
| P1 | KSPP | 32 | 17 | 38 | 5.417 | 6.555 | 0.064 | 0.017 | 0.969 |
| P2 | BKPP | 33 | 17 | 33 | 5.159 | 6.070 | 0.064 | 0.017 | 0.909 |
| P3 | LFPP | 33 | 17 | 26 | 5.417 | 6.931 | 0.064 | 0.017 | 1.000 |
| P4 | HCPP | 32 | 17 | 24 | **5.417** | **6.730** | 0.064 | 0.017 | 0.983 |

multiplication. It consist of short wires when move from one block to adjacent block. In Vedic multiplier the word Vedic is derived from "Veda". Veda contains the house of knowledge with 16 sutras in it. Array and Vedic multiplier has certain drawbacks which are overcome by Wallace multiplier (WM). The layout of WM is complex but it is less time consuming process (Wallace, 1964). WM results remarkable values in comparison to other multipliers in terms of speed as explained by Thakur et al. (2018a, 2018b, 2018c). Multiplier consists of different stages of operation consisting partial product generation (PPG), partial product processing (PPP) and final addition. To increase the speed of WM is designed using PP adders, for addition of partial product at last stage. Figure 8 shows the flow diagram of Wallace multiplier using different types of PP adder. In this flow diagram initially four blocks of 8 × 8 bit of Wallace multiplier having input A and B is considered. After multiplication, multiplicand and multiplier partial products are generated. In the partial product different operations are performed for addition of partial products. Final addition of partial product is implemented by using different PP adders (KSPP, BKPP, LFPP, HCPP).

Figure 9 shows the block diagram representation of 16 bit WM. The structure consists of three 16 bit PP adders, one OR gate and four blocks of 8 × 8 WM.

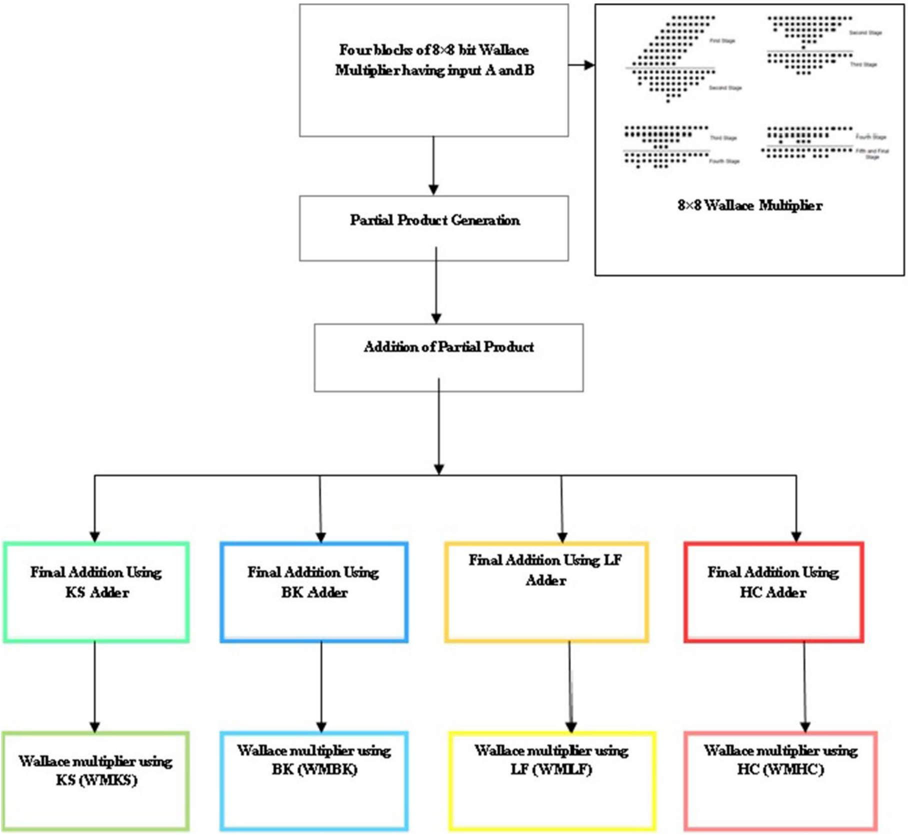
Four blocks of 8×8 bit Wallace Multiplier having input A and B
First Stage
Second Stage
Second Stage
Third Stage
Third Stage
Fourth Stage
Fourth Stage
Fifth and Final Stage
8×8 Wallace Multiplier
Partial Product Generation
Addition of Partial Product
Final Addition Using KS Adder
Final Addition Using BK Adder
Final Addition Using LF Adder
Final Addition Using HC Adder
Wallace multiplier using KS (WMKS)
Wallace multiplier using BK (WMBK)
Wallace multiplier using LF (WMLF)
Wallace multiplier using HC (WMHC)

**Fig. 8** Flow diagram Wallace multiplier using different types of PP adder

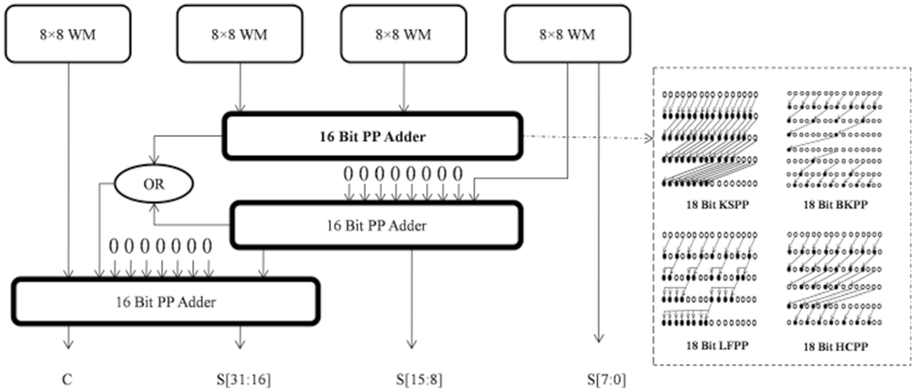

**Fig. 9** Block diagram of 16 bit Wallace multiplier

**Table 2.** 16 bit Wallace multiplier using different PP adders

| Multiplier circuit | | No. of bonded IOB | | Number of slice LUTs | Delay(ns) | | Total power (W) | | Power delay product (Wns) |
|---|---|---|---|---|---|---|---|---|---|
| | | I-Buf | O-Buf | | Logic | Router | Leakage | IOs | |
| M1 | WMKS | 32 | 32 | 517 | 7.258 | 15.856 | 0.064 | 0.025 | 2.057 |
| M2 | WMBK | 32 | 32 | 504 | 6.844 | 15.316 | 0.064 | 0.025 | 1.972 |
| M3 | WMLF | 32 | 32 | 501 | 7.457 | 17.203 | 0.064 | 0.025 | 2.194 |
| M4 | WMHC | 32 | 32 | 484 | 6.842 | 14.429 | 0.064 | 0.025 | 1.893 |

WM is simulated using different PP adder which is shown in Fig. 9. Each block of 8×8 WM consists of 16 bit input. Different PP adders are used to sum up the output of different blocks of 8×8 WM. Finally, 32 bit sum and 1 bit carry is generated. Table 2 shows the results of 16 bit Wallace multiplier using different PP adders in which WMHC (M4) shows the remarkable results in term of speed (21.271 ns) and LUTs in comparison to M1, M2 and M3.

In this paper an efficient, high speed and low power Radix-2 FFT structure is implemented using different PP adders and multipliers. Figure 10 represents Radix-2 structure which is implemented using different conventional adders and multiplier blocks.

Each block has its own characteristic for designing an efficient Radix-2 structure. There are four different types of adders which helps in designing four different types of multiplier block (Wallace multiplier using Kogge Stone adder (WMKS—M1), Wallace multiplier using Brent Kung adder (WMBK—M2), Wallace multiplier using Landner Fischer adder (WMLF —M3) and Wallace multiplier using Han Carlson adder (WMHC—M4)). All the simulated results using different adders and multipliers are divided into four different cases. The simulations were carried out in Xilinx ISE 14.7 design suite software and power is calculated by using XPower analyzer.

**2.1.2.1 Case 1: Radix-2 FFT structure using WMKS (M1) and different PP adders** The results were calculated by fixing the multiplier block and varying PP adder. Table 3 shows the results of Radix-2 FFT structure using WMKS and different types of PP adders. The results

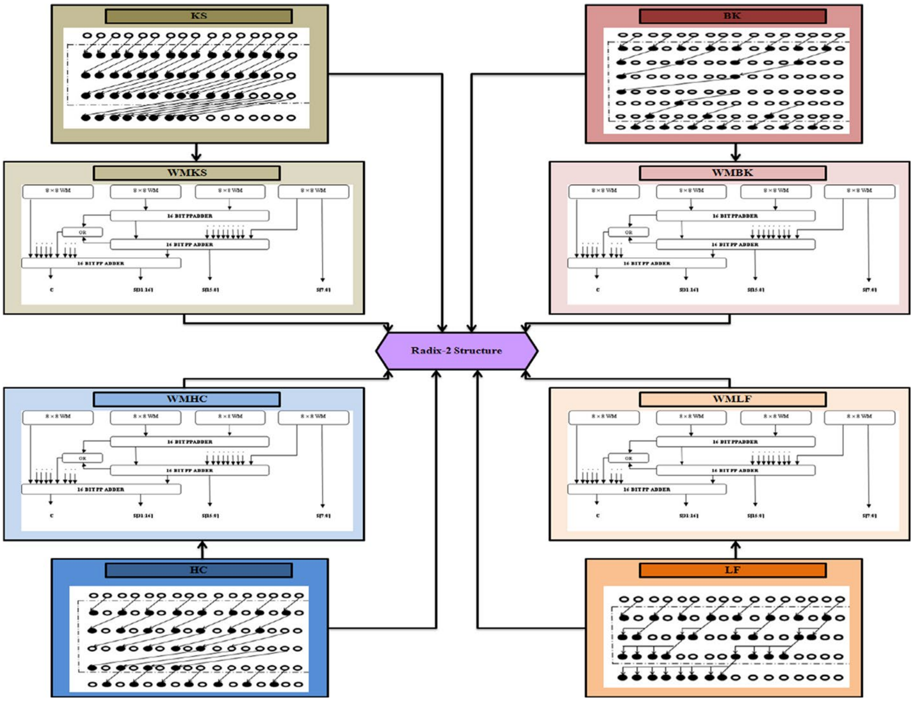

**Fig. 10** Existing Radix-2 DIT FFT structure

were compared in terms of speed, area and power. Radix-2 FFT structures using P1 adder results in 24.003 ns (7.254 ns logic delay and 16.749 ns router delay) and 2.760Wns Power Delay Product (PDP). It results in 2%, 2.94% and 2.94% improvement in speed, comparison with M1P2, M1P3 and M1P4 models.

**2.1.2.2 Case 2: Radix-2 FFT structure using WMBK (M2) and PP adders** The results were calculated by fixing the multiplier block and varying PP adder. Table 4 shows the result of Radix-2 FFT structure using WMBK and different types of PP adders. The results were compared in terms of speed, area and power. Radix-2 FFT structure using P2 results in 24.162 ns (7.254 ns logic delay and 16.908 ns router delay) and 2.778Wns PDP. It results in 2.8%, 1.6% and 1.8% improvement in speed, comparison with M2P1, M2P3 and M2P4 models.

**2.1.2.3 Case 3: Radix-2 FFT structure using WMLF (M3) and different PP adders** The results were calculated by fixing the multiplier block and varying PP adder. Table 5 shows the result of Radix-2 FFT structure using WMLF and different types of PP adders. The results were compared in terms of speed, area and power.Radix-2 FFT structure using P3 adder results in 24.889 ns (7.459 ns logic delay and 17.430 ns router delay) and 2.862Wns PDP. It results in 6.23%, 4.67% and 5.3% improvement in speed, comparison with M3P1, M3P2 and M3P4 models.

**Table 3** Radix-2 FFT structure using WMKS and different PP adders

| Butterfly structure using | | M1P1 | M1P2 | M1P3 | M1P4 |
|---|---|---|---|---|---|
| BELS | LUT2 | 583 | 546 | 547 | 546 |
| | LUT3 | 125 | 123 | 125 | 119 |
| | LUT4 | 165 | 159 | 164 | 158 |
| | LUT5 | 306 | 357 | 353 | 350 |
| | LUT6 | 1108 | 1014 | 999 | 999 |
| | MUXF7 | 1 | 1 | 1 | 1 |
| | Total BELS | 2288 | 2200 | 2189 | 2173 |
| Slice logic utilization | Number of Slice LUTs | 2287 | 2199 | 2188 | 2172 |
| | Number used as Logic | 2287 | 2199 | 2188 | 2172 |
| | % of utilization | 4% | 4% | 4% | 4% |
| IO buffers | IBUF | 96 | 96 | 96 | 96 |
| | OBUF | 86 | 86 | 86 | 86 |
| | Total IO Buffers | 182 | 182 | 182 | 182 |
| Total delay(ns) | Logic Delay | **7.254** | 7.467 | 7.457 | 7.457 |
| | Router Delay | **16.749** | 17.060 | 17.275 | 17.275 |
| Total power(W) | Leakage | 0.065 | 0.065 | 0.065 | 0.065 |
| | IOs | 0.050 | 0.050 | 0.050 | 0.050 |
| Power delay product (Wns) | | 2.760 | 2.820 | 2.844 | 2.844 |

*Note* M1P1 indicates WMKS with KS adder, M1P2 indicates WMKS with BK, M1P3 indicates WMKS with LF and M1P4 indicates WMKS with HC

**Table 4** Radix-2 FFT structure using WMBK and different PP adders

| Butterfly structure using | | M2P1 | M2P2 | M2P3 | M2P4 |
|---|---|---|---|---|---|
| BELS | LUT2 | 563 | 565 | 559 | 557 |
| | LUT3 | 111 | 101 | 105 | 100 |
| | LUT4 | 172 | 157 | 164 | 168 |
| | LUT5 | 299 | 314 | 318 | 318 |
| | LUT6 | 1048 | 1006 | 971 | 973 |
| | Total BELS | 2193 | 2143 | 2117 | 2116 |
| Slice logic utilization | Number of Slice LUTs | 2193 | 2143 | 2117 | 2116 |
| | Number used as Logic | 2193 | 2143 | 2117 | 2116 |
| | % of utilization | 4% | 4% | 4% | 4% |
| IO buffers | IBUF | 96 | 96 | 96 | 96 |
| | OBUF | 86 | 86 | 86 | 86 |
| | Total IO Buffers | 182 | 182 | 182 | 182 |
| Total delay(ns) | Logic Delay | 7.248 | **7.254** | 7.250 | 7.250 |
| | Router Delay | 17.630 | **16.908** | 17.323 | 17.356 |
| Total power(W) | Leakage | 0.065 | 0.065 | 0.065 | 0.065 |
| | IOs | 0.050 | 0.050 | 0.050 | 0.050 |
| Power delay product (Wns) | | 2.860 | 2.778 | 2.825 | 2.829 |

***Note*** M2P1 indicates WMBK with KS adder, M2P2 indicates WMBK with BK, M2P3 indicates WMBK with LF and M2P4 indicates WMBK with HC

**Table 5** Radix-2 FFT structure using WMLF and different PP adders

| Butterfly structure using | | M3P1 | M3P2 | M3P3 | M3P4 | M3P5 |
|---|---|---|---|---|---|---|
| BELS | LUT2 | 570 | 564 | 554 | 564 | 570 |
| | LUT3 | 138 | 129 | 107 | 127 | 121 |
| | LUT4 | 139 | 130 | 138 | 127 | 135 |
| | LUT5 | 302 | 325 | 330 | 322 | 297 |
| | LUT6 | 1037 | 985 | 950 | 966 | 997 |
| | Total BELS | 2186 | 2133 | 2079 | 2106 | 2120 |
| Slice logic utilization | Number of Slice LUTs | 2186 | 2133 | 2079 | 2106 | 2120 |
| | Number used as Logic | 2186 | 2133 | 2079 | 2106 | 2120 |
| | % of utilization | 4% | 4% | 4% | 4% | 3% |
| IO buffers | IBUF | 96 | 96 | 96 | 96 | 96 |
| | OBUF | 86 | 86 | 86 | 86 | 86 |
| | Total IO Buffers | 182 | 182 | 182 | 182 | 182 |
| Total delay(ns) | Logic Delay | 7.656 | 7.658 | **7.459** | 7.658 | 7.658 |
| | Router Delay | 18.887 | 18.451 | **17.430** | 18.628 | 18.312 |
| Total power(W) | Leakage | 0.065 | 0.065 | 0.065 | 0.065 | 0.029 |
| | IOs | 0.050 | 0.050 | 0.050 | 0.050 | 0.050 |
| Power delay product (Wns) | | 3.052 | 3.002 | 2.862 | 3.022 | **2.051** |

*Note* M3P1 indicates WMLF with KS adder, M3P2 indicates WMLF with BK, M3P3 indicates WMLF with LF and M3P4 indicates WMLF with HC

**Table 6** Radix-2 FFT structure using WMHC and different PP adders

| Butterfly structure using | | M4P1 | M4P2 | M4P3 | M4P4 |
|---|---|---|---|---|---|
| BELS | LUT2 | 546 | 541 | 542 | 548 |
| | LUT3 | 105 | 95 | 98 | 107 |
| | LUT4 | 155 | 147 | 148 | 123 |
| | LUT5 | 325 | 344 | 342 | 339 |
| | LUT6 | 991 | 943 | 929 | 956 |
| | Total BELS | 2122 | 2070 | 2059 | 2073 |
| Slice Logic Utilization | Number of Slice LUTs | 2122 | 2070 | 2059 | 2073 |
| | Number used as Logic | 2122 | 2070 | 2059 | 2073 |
| | % of utilization | 4% | 4% | 4% | 4% |
| IO Buffers | IBUF | 96 | 96 | 96 | 96 |
| | OBUF | 86 | 86 | 86 | 86 |
| | Total IO Buffers | 182 | 182 | 182 | 182 |
| Total Delay(ns) | Logic Delay | 7.244 | 7.246 | 7.246 | **7.049** |
| | Router Delay | 16.666 | 16.562 | 16.790 | **15.778** |
| Total Power(W) | Leakage | 0.065 | 0.065 | 0.065 | 0.065 |
| | IOs | 0.050 | 0.050 | 0.050 | 0.050 |
| Power Delay Product (Wns) | | 2.749 | 2.737 | 2.764 | 2.625 |

M4P1 indicates WMHC with KS adder, M4P2 indicates WMHC with BK, M4P3 indicates WMHC with LF, M4P4 indicates WMHC with HC and M4P5 indicates WMHC with HKS

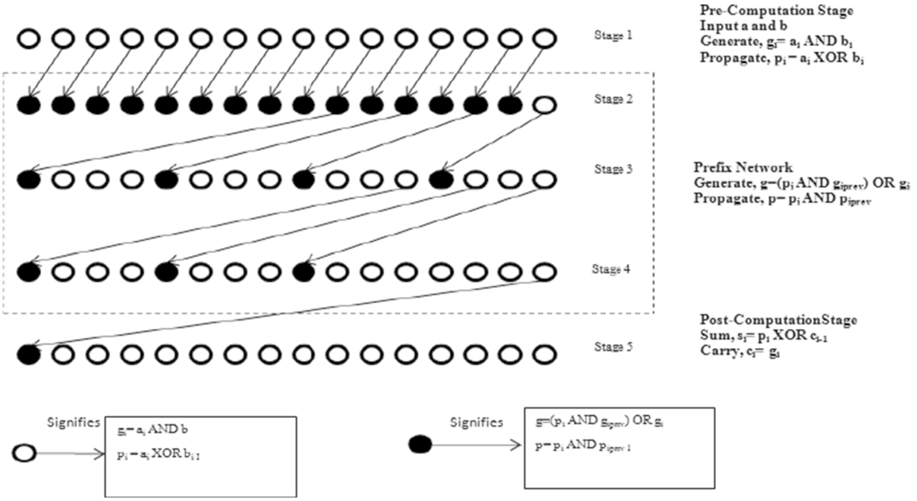
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Pre-Computation Stage
Input a and b
Generate, $g_i$= $a_i$ AND $b_i$
Propagate, $p_i$ – $a_i$ XOR $b_i$
Prefix Network
Generate, g–($p_i$ AND $g_{iprev}$) OR $g_i$
Propagate, p– $p_i$ AND $p_{iprev}$
Post-ComputationStage
Sum, $s_i$= $p_i$ XOR $c_{i-1}$
Carry, $c_i$= $g_i$
Signifies
$g_i$– $a_i$ AND b
$p_i$ – $a_i$ XOR $b_{i 1}$
Signifies
g=($p_i$ AND $g_{iprev}$) OR $g_i$
p– $p_i$ AND $p_{iprev 1}$

**Fig. 11.** 16 bit proposed NPP adder

**Table 7** Result of 16 bit NPP adder

| Adder | | No. Of Bonded IOB | | Number of Slice LUTs | Delay(ns) | | Total Power (W) | | Power Delay Product (Wns) |
|---|---|---|---|---|---|---|---|---|---|
| | | I-Buf | O-Buf | | Logic | Router | Leakage | IOs | |
| P5 | NPP | 32 | 17 | 29 | 4.609 | 3.852 | 0.029 | 0.017 | 0.380 |

#### 2.1.2.4 Case 4: Radix-2 FFT structure using WMHC (M4) and different PP adders

The results were calculated by fixing the multiplier block and varying PP adder. Table 6 shows the result of Radix-2 FFT structure using WMHC and different types of PP adders. The results were compared in terms of speed, area and power.Radix-2 FFT structure using P4 adder results in 22.827 ns (7.049 ns logic delay and 15.778 ns router delay) and 2.625Wns PDP. It results in 4.5%, 4.1% and 5% improvement in speed, comparison with M4P1, M4P2 and M4P3 models.

The conventional Radix-2 FFT structure using Han Carlson PP adder and Wallace multiplier results 22.827 ns delay which is showing remarkable results among other architectures. To further improve the power and speed of the structure, novel PP adder and optimized multiplier is proposed to realize efficient Radix-2 FFT structure.

## 2.2 Implementation of radix-2 using proposed arithmetic structures

In this paper, authors have proposed novel PP (NPP) adder. The proposed block diagram of NPP is shown in Fig. 11 which further decreases the intricacy with respect to KSPP adder. For proposed PP adder generate and propagate is calculated at stage 1.

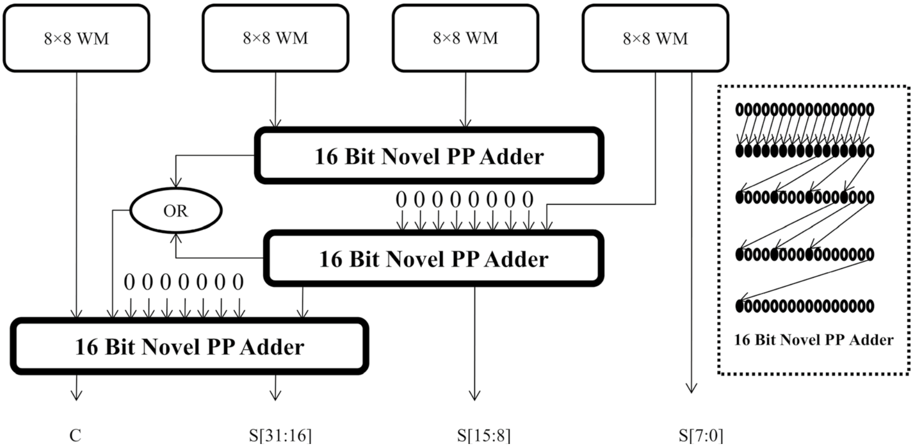

**Fig. 12** Block diagram of 16 bit multiplier using NPP adder

**Table 8.** 16 bit Wallace multiplier using different PP adders

| Multiplier wallace using | | No. of bonded IOB | | Number of Slice LUTs | Delay(ns) | | Total Power (W) | | Power delay product (Wns) |
|---|---|---|---|---|---|---|---|---|---|
| | | I-Buf | O-Buf | | Logic | Router | Leakage | IOs | |
| M5 | WMNPP | 32 | 32 | 496 | 6.032 | 10.585 | 0.029 | 0.025 | 0.897 |

Final carry using generate and propagate is computed from Stage 2 to stage 4 and sum is calculated from the last stage.

The novel parallel prefix adder show remarkable result as compared to state-of-art work. Novel parallel prefix adder used in applications where speed is the major requirement because it shows high speed as compared to other adders. The proposed novel adder is simulated and the results are tabulated in Table 7.

From the table it is interpreted that NPP adder results in optimized area with high speed and less power consumption. Power delay product for this adder is found to be 0.380Wns which makes it preferable choice for designing the different blocks of signal processing applications.

Multiplier circuit is simulated using NPP adder which is shown in Fig. 12. In this figure four 8 × 8 WM blocks, three 16 bit NPP adder and one OR gate is used. Each block of 8 × 8 WM consists of 16 bit input.

NPP adder is used at different stages to obtain the final sum.

A multiplier circuit using NPP adder results in 16.617 ns (6.032 ns logic delay and 10.585 ns router delay) delay which is tabulated in Table 8. Multiplier circuit is simulated using Xilinx 14.7 ISE design suite. Table also tabulates power, number of slices, LUTs, IOB and power delay product.

In comparison to other multiplier blocks, WMNPP shows remarkable results in terms of power utilization, area, speed and power delay product. WMNPP results in

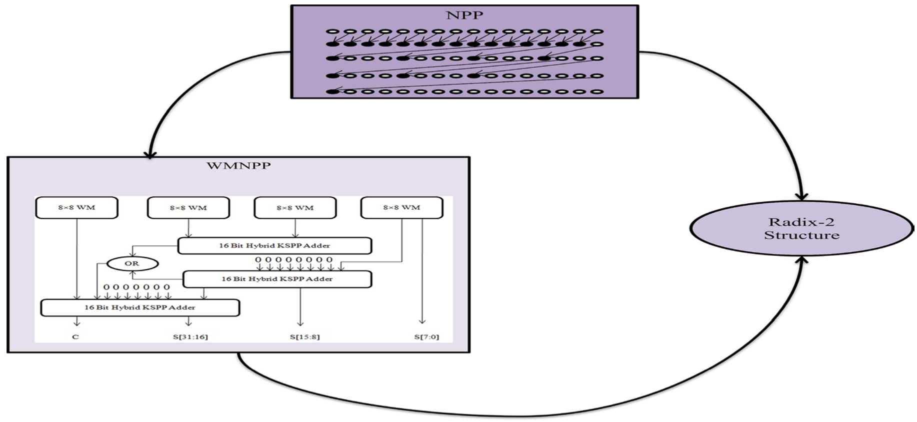

**Fig. 13** Proposed Radix-2 DIT FFT structure

**Table 9** Radix-2 FFT structure using WCHKS and different PP adders

| Butterfly structure using | | M5P1 | M5P2 | M5P3 | M5P5 | M5P5 |
|---|---|---|---|---|---|---|
| BELS | LUT2 | 571 | 566 | 566 | 557 | 567 |
| | LUT3 | 93 | 85 | 88 | 75 | 83 |
| | LUT4 | 177 | 166 | 163 | 162 | 161 |
| | LUT5 | 234 | 256 | 261 | 241 | 257 |
| | LUT6 | 1088 | 1036 | 1002 | 1050 | 1014 |
| | Total BELS | 2163 | 2109 | 2080 | 2085 | 2082 |
| Slice logic utilization | Number of slice LUTs | 2163 | 2109 | 2080 | 2085 | 2082 |
| | Number used as logic | 2163 | 2109 | 2080 | 2085 | 2082 |
| | % of utilization | 3% | 3% | 3% | 3% | 3% |
| IO buffers | IBUF | 96 | 96 | 96 | 96 | 96 |
| | OBUF | 86 | 86 | 86 | 86 | 86 |
| | Total IO buffers | 182 | 182 | 182 | 182 | 182 |
| Total delay(ns) | Logic delay | 6.641 | 6.440 | 6.237 | 6.237 | **6.030** |
| | Router delay | 14.258 | 13.145 | 12.568 | 12.159 | **12.188** |
| Total power(W) | Leakage | 0.029 | 0.029 | 0.029 | 0.029 | 0.029 |
| | IOs | 0.050 | 0.050 | 0.050 | 0.050 | 0.050 |
| Power Delay Product (Wns) | | 1.651 | 1.547 | 1.485 | 1.453 | **1.439** |

*Note* M5P1 indicates WMNPP with KS adder, M5P2 indicates WMNPP with BK, M5P3 indicates WMNPP with LF, M5P4 indicates WMNPP with HC and M5P5 indicates WMNPP with NPP adder

only 496 LUTs and 0.897 Wns PDP. From the table it is also interpreted that WMNPP multiplier results in optimized area with high speed, less power consumption and low area. Figure 13 shows the proposed structure of Radix-2 FFT structure using NPP adder and optimized multiplier.

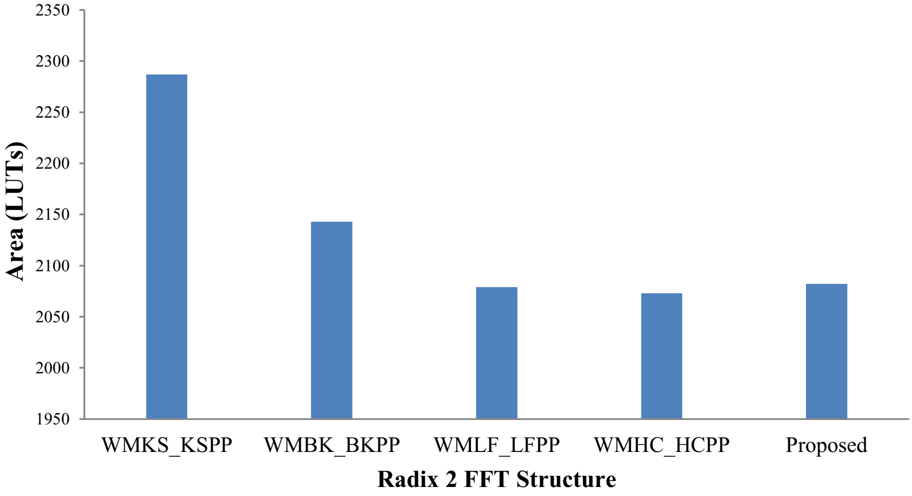

**Fig. 14** Chip Area for proposed Radix-2 DIT FFT structure

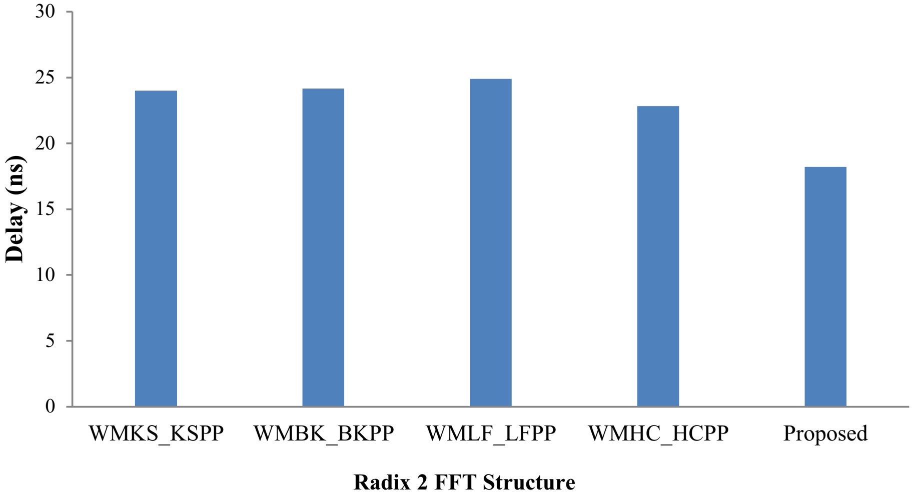

**Fig. 15** Delay (ns) for proposed Radix-2 DIT FFT structure

### 2.2.1 Radix-2 FFT structure using WMNPP (M5) and different PP adders

The results were calculating by fixing the multiplier block and varying PP adders. Table 9 tabulates the result of Radix-2 FFT structure using proposed NPP adder. The result of proposed Radix-2 FFT structure is compared with the conventional structure in terms of speed, area and power. Radix-2 FFT structures using NPP adder results in 18.218 ns (6.030 ns logic delay and 12.188 ns router delay) and 1.439Wns PDP. It results in 12.8%, 6.97%, 3.12% and 0.9% improvement in speed, with M5P1, M5P2, M5P3 and M5P4 models.

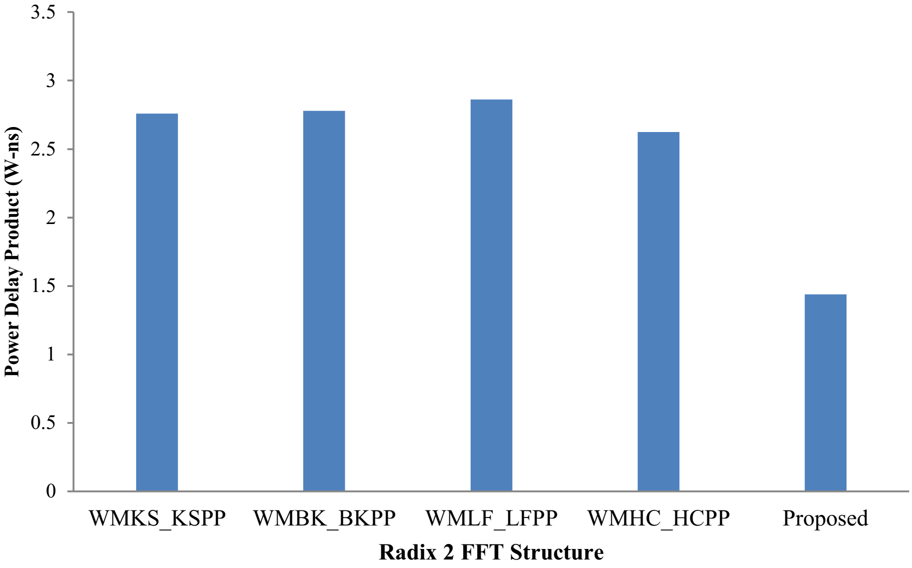

**Fig. 16** Power-Delay Product for proposed Radix-2 DIT FFT structure

The results of Radix-2 FFT structure using different PP adders in terms of area, speed and power-delay product are shown in Figs. 14, 15 and 16. The simulation result validates that M5P5 yields 18.218 ns delay (6.030 ns logic delay and 12.118 ns router delay) in comparison with others existing results (24.003 ns delay (7.254 ns logic delay and 16.749 ns router delay) for M1P1, 24.162 ns delay (7.254 ns logic delay and 16.908 ns router delay) for M2P2, 24.889 ns delay(7.459 ns logic delay and 17.430 ns router delay)for M3P3 and 18.218 ns delay(6.030 ns logic delay and 12.188 ns router delay)for M4P4). Our proposed Radix 2 structure (M5P5) using novel PP adder yields minimum power delay product (1.439Wns) in comparison with other architectures.

In VLSI design the performance is based on power consumption, speed of the design and area occupied. In this paper all these factor is taken very carefully. M5P5 results in an efficient Radix-2 structure for solving complex calculation with high speed and low power consumption. Different parameters were calculated such as LUTs, delay and power-delay product as shown in Figs. 14, 15 and 16. After simulation it has been concluded that among the entire adders' novel PP adder is more efficient than other PP adders because it takes less area and delay when used in multiplier block and proposed Radix-2 structure. Table 10 shows the percentage improvement of Radix-2 FFT structure in terms of delay (ns) comparison to conventional structure.

**Table 10** Percentage improvement of Proposed Radix-2 FFT structure

| Radix-2 Structure | Percentage improvement |
|---|---|
| M1P1 | 24.10% |
| M2P2 | 24.60% |
| M3P3 | 26.80% |
| M4P4 | 20.19% |

**Table 11** Comparison of FFT processors

| | Proposed | S. Liu and D. Liu (2019) | Tang et al. (2014) | Chen et al. (2015) | Shih et al. (2018) | Akhil et al. (2020) |
|---|---|---|---|---|---|---|
| Architecture | Memory-based | Memory-based | Memory-based | Memory-based | SDF | Memory-based |
| Technology | Spartan-6 | SMIC 65 nm | TSMC 180 nm | SMIC 180 nm | TSMC 40 nm | ZYNQ |
| Bit width | 16-Bit | 14-Bit | 12-Bit | 16-Bit | 14-bit | 16-Bit |
| Bonded IOB | 182 | – | – | – | – | 513 |
| SQNR (dB) | 23.6 | 66.1 | 48.4 | 23 | 35.84 | - |
| Area (mm$^2$) | 2.08 | 1.21 | 4.8 | 5 | 0.36 | 1.18 |
| Power (mW) | 79 | 68.64 | 156.2 | 320 | 48.46 | - |
| Energy (J) | 1.43 | 4.07 | 1.27 | 0.66 | 3.22 | - |

Table 11 show the comparison of proposed architecture from other state-of-work. Proposed architecture is memory-based design implemented by using Verilog HDL coding but state-of-work is designed using CMOS technology. If optimized FFT architecture is required for high speed signal processing application than proposed design is efficient as it provides high computational speed but lacks in power consumption.

## 3 Conclusion

This paper explains an efficient Radix-2 FFT implementation to exploit FPGA resources in an optimized way. We have presented Radix-2 FFT implementation with improved delay and power consumption. A novel parallel prefix (NPP) adder which has shown 29.32%, 24.65%, 31.47% and 30.34% improvement over P1, P2, P3 and P4 is proposed. This NPP adder is then used to implement optimized multiplier which has shown 28.10%, 25.01%, 32.61% and 21.87% improvement over M1, M2, M3 and M4. Further we used this NPP adder and optimized multiplier as building blocks for Radix-2 FFT structure. This implementation has shown improvement of 24.10%, 24.60%, 26.80% and 20.19% over M1P1, M2P2, M3P3 and M4P4. This is an effective strategy of actualizing adders and multipliers on FPGAs that can be utilized as fundamental building block to form various types of DSP applications. If optimized FFT architecture is required for high speed signal processing application than proposed design is efficient as it provides high computational speed but lacks in power consumption.

## References

Akbar, E. P. A., & Mosleh, M. (2019). An efficient design for reversible wallace unsigned multiplier. *Theoretical Computer Science, 773,* 43–52.

Akhil, R., Koleti, J. R., Vijaya Bhaskar, A., Sathish, V., & Goud, B. A. (2020). Delay and area analysis of hardware implementation of FFT using FPGA. In: *IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT):* 1–6.

Brent, R. P., & Kung, H. T. (1982). A regular layout for parallel adders. *IEEE Transactions on Computers, 31*(3), 260–264.

Chen, J., Hu, J., Lee, S., & Sobelman, G. E. (2015). Hardware Efficient Mixed Radix-25/16/9 FFT for LTE Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 23*(2), 221–229.

Dimitrakopoulos, G., & Nikolos, D. (2005). high-speed parallel-prefix VLSI ling adders. *IEEE Transactions on Computers, 54*(2), 225–231.

Hans, T., & Carlson, D. A. (1987). Fast area-Eeficient VLSI adders. In: *Proc. 8th IEEE Symposius on Computer Arithmetic*, pp 49–56.

Hussain, I., Pandey, C. K., & Chaudhury, S. (2019). Design and analysis of high performance multiplier circuit. *Devices for Integrated Circuit (DevIC)*: 23–24.

Jayakumar, D., & Logashanmugam, E. (2016). Design of combined Radix-2, Radix-4 and Radix-8 based single path delay feedback (SDF) FFT. *Indian journal of science and technology, 9,* 45.

Kavya, T., Deepa, K., & Jayamangala, S. (2017). Design and implementation of high performance Radix-2 and Radix-4 butterflies from FFT. *International Journal of Electronics, Electrical and Computational System, 6*(12), 363–367.

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, 22*(8), 786–793.

Ladner, R. E., & Fisher, M. J. (1980). Parallel Prefix Computation. *Journal of the AssoclaUon for Computing Machinery, 27*(4), 831–838.

Liu, S., & Liu, D. (2019). A High-Flexible Low-Latency Memory-Based FFT Processor for 4G, WLAN, and Future 5G. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27*(3), 511–523.

Narvekar, N. Raikar, S., Salkar, P., & Shirodkar, A. (2019). Implementation of FFT processor on FPGA using Vedic multiplier. *International journal of research and analytical reviews, 6*(2), 211–215.

Santhosh, L., & Thomas A. (2013). Implementation of Radix 2 and Radix 22 FFT algorithms on Spartan 6 FPGA. In: *International Conference on Computer Communication and Networking*.

Shih, X., Chou, H., & Liu, Y. (2018). VLSI Design and Implementation of Reconfigurable 46-Mode Combined-Radix-Based FFT Hardware Architecture for 3GPP-LTE Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers, 65*(1), 118–129.

Singh, A. K., & Nandi, A. (2017). Design of Radix 2 butterfly structure using Vedic multiplier and CLA on Xilinx. In: *Proc. IEEE Conference on Emerging Devices and Smart Systems*.

Tang, S., Jan, F., Cheng, H., Lin, C., & Wu, G. (2014). Multimode memory-based FFT processor for wireless display FD-OCT medical systems. *IEEE Transactions on Circuits and Systems I: Regular Papers, 61*(12), 3394–3406.

Thakur, G., Sohal, H., & Jain, S. (2018a). Design and comparative performance analysis of various multiplier circuit. *Journal of Scientific and Engineering Research, 5*(7), 340–434.

Thakur, G., Sohal, H., & Jain, S. (2018b). An efficient design of 8-bit high speed parallel prefix adder. *Research Journal of Science and Technology, 5*(7), 340–349.

Thakur, G., Sohal, H., & Jain, S. (2018c). High speed RADIX-2 butterfly structure using novel Wallace multiplier. *International Journal of Engineering & Technology, 7*(3.4), 213–217.

Thakur, G., Sohal, H., & Jain, S. (2020a). Design and analysis of high-speed parallel prefix adder for digital circuit design applications. In: *International Conference on Computational Performance Evaluation (ComPE)*, pp 095–100.

Thakur, G., Sohal, H., & Jain, S. (2020b). FPGA-based parallel prefix speculative adder for fast computation application. In: *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 206–210.

Wallace, C. S. (1964). A Suggestion for a Fast Multiplier. *IEEE Transactions on Electronic Computers, 13*(1), 14–17.

Water, R. S., & Swartzlander, E. E. (2010). A Reduced complexity wallace multiplier reduction. *IEEE Transactions on Computers, 59*(8), 1134–1137.

**Garima Thakur** has received M.Tech degree in Electronics and Communication from Jaypee university of information technology (JUIT), Waknaghat, in 2018. She is currently pursuing for her Ph.D. degree in Digital VLSI Circuits. Her current research interests are FPGA implementation of arthimetic logic circuits and approximate computing.

**Harsh Sohal** has Ph. D. in biomedical Engineering from Kyung Hee University, South Korea. He specialises in FPGA based instrumentation system design. His primary research interests are in the field of electronic instrumentation, circuits & systems design for medical electronics applications. He is a member of IEEE and is a life member of biomedical engineering society of India. In his free time he enjoys playing chess and reading scientific fiction.

**Shruti Jain** is an Associate Professor in the Department of Electronics and Communication Engineering at Jaypee University of Information Technology, Waknaghat, H.P, India and has received her Doctor of Science (D. Sc.) in Electronics and Communication Engineering. She has a teaching experience of around 15 years. She has filed three patents out of which one patent is granted and one is published. She has published more than 09 book chapters, and 100 research papers in reputed indexed journals and in international conferences. She has also published six books. She has completed two government-sponsored projects. She has guided 06 Ph.D. students and now has 02 registered student. She has also guided 11 M Tech scholars and more than 90 B Tech undergrads. Her research interests are Image and Signal Processing, Soft Computing, Bio-inspired Computing and Computer-Aided Design of FPGA and VLSI circuits. She is a senior member of IEEE, life member and Editor in Chief of Biomedical Engineering Society of India and a member of IAENG. She is a member of the Editorial Board of many reputed journals. She is also a reviewer of many journals and a member of TPC of different conferences. She was awarded by Nation Builder Award in 2018–19.