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

https://dx.doi.org/10.12785/ijcds/120142

# Distributed Memory based Architecture for Multiplier

**S. Aruna Mastani**[1] **and S. Kannappan**[2]

[1]*Department of Electronics and Communication Engineering, JNTUA College of Engineering, Anantapuramu, India*
[2]*Department of Electronics and Communication Engineering, GATES Institute of Technology, Gooty, India*
[1,2]*E-Mail address: aruna_mastani.ece@jntua.ac.in, usk310587@gmail.com*

*Received 2 Apr.2021, Revised 18 Jul. 2022, Accepted 31 Jul. 2022, Published 6 Aug. 2022*

**Abstract:** Novel multiplier architecture is proposed based on the concept of memory-based computing in contrast to logic computations, thus making it efficient to implement both on Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Array (FPGAs). Unlike the traditional approaches of using column bits compressors/ counters for partial product reduction, in this work, the partial products are added using basic 'm-operand adders' where m= 2, 3, 4, 5. This reduces the depth of pipelining that result in long carry propagation. These adders are designed using only registers and small ROMs and optimized the performance w.r.t. area and delay. The Partial product generator used is an AND gate-array for unsigned multiplier, and for signed multiplier radix-4 Booth encoding is used. The architecture can be extended to 'N-bit multipliers' by re-use of basic m-operand adder modules. The proposed unsigned multiplier utilizes 15.52% less LUTs and 41.033% less delay compared to existing 16-bit multiplier [1]. The proposed 16-bit signed multiplier has 36.17% less LUTs but 5.5% of more delay than that of existing 16-bit multiplier [2] respectively. After exhaustive experimentation and analysis made by varying the bit-length and number of operands it's evident that proposed multiplier outperforms existing ones.

**Keywords:** multiplier; arithmetic unit; distributed memory; column bits compressors; counter; multi-operand adder; look-up-table.

## 1. Introduction

Multiplication is one of the mostly used arithmetic operations which involve addition of multiple numbers i.e. the partial products, and this is the crucial part that decides the performance of the Multiplier w.r.t area and delay. Generally the use of compressor trees has a long history in the design of arithmetic units [3], [4], [5], [6]. The basic idea is to avoid the slow carry propagation by passing (saving) the carry to the next compressor stage instead of propagating it within the same stage. While this guarantees substantial improvement in Application Specific Integrated Circuits (ASICs) or custom ICs, but the use of carry save arithmetic on Field Programmable Gate Arrays (FPGAs) was regarded as unsuitable, and more over efficient mapping of compressor trees on to LUTs decide the performance. In 2008, Parandeh-Afshar et al., [7] introduced the concept of so-called Generalized Parallel Counters (GPCs) is adapted in FPGAs. Parandeh-Afshar in his publication titled “Efficient Synthesis of Compressor Trees on FPGAs” has reported the first method that synthesizes compressor trees on FPGAs with GPCs as basic element and found drastic improvement both in terms of resource utilization and delay. Matsunaga et al. [8] in their research publication of 2013 titled “An Exact Approach for GPC-Based Compressor Tree Synthesis” formulated the mapping of GPCs as an Integer Linear Programming (ILP) which resulted in reduced count of GPCs in compressor tree.

Many of the inventions prior have targeted to develop logic for efficient compressor tree in multi operand adders [9]. The logic for compressors used is different in ASICs and FPGAs. A compressor efficient in the ASIC may not map on to LUT fabric i.e. FPGA efficiently. Hence the motive of this work is to bridge the gap between ASICs and FPGAs by providing a soft core of a multiplier architecture that uses only registers and ROMs / LUTs so that the difficulty of mapping the logic on to LUTs of FPGAs is simplified. As LUTs are also a memory elements, in this architecture the LUTs can be used instead to ROMs. Different FPGA family has different size of LUTs, hence to prove the efficacy of the proposed architecture design, it is synthesized on different FPGAs and corresponding comparison is made with the most recent existing work.

The long carry chain propagation is a major hurdle in partial product addition of larger size multipliers. This is optimized in the proposed architecture as the carry propagate only within a basic block of m- operand adder module (m=2, 3, 4, 5). Even these blocks are connected in multiple stages, the carry bits are stored within the module

*E-mail address: aruna_mastani.ece@jntua.ac.in, usk310587@gmail.com* to add to next higher weight bits coming serially, and only sum bits are propagated from stage to stage in such a way that pipelining and parallelism are implemented implicitly thus eliminating the long carry chain. The multi-operand adder concept used in the proposed Multiplier architecture has been filed/ published and is with the Indian patent office bearing application number 201941024136 pending [10]. From the above discussion the proposed multiplier architecture made of memory blocks is equally efficient for both ASIC and FPGA implementation more over as carry propagation is limited to individual modules, the bottlenecks of long carry chain and complexity of routing to pipelining as in conventional multipliers is removed. The proposed architecture can be extended to larger bit sizes by the replicating the m-operand modules thus supporting modularity and regularity which is a figure of merit in IC fabrication. The total delay in number of clock cycles increases logarithmically with increased in bit length in contrast to linear increase in existing multipliers.

## 2. Related Work

The proposed multiplier is synthesized on different FPGAs to compare the performance with the existing multipliers that were implemented on same FPGA. In this section the existing techniques are discussed in brief. In 2015, Mani Kunnathettu et al., [11] have implemented an efficient high speed Wallace Tree Multiplier on Artix-7 FPGA. In 2016, K. M. Gaikwad et al., [12] have analysed array and vedic multipliers using Xilinx. In 2018, Arpita Singh et al., [1] have implemented “High Speed Multiplier with Optimized Reduction Phase” on Artix-7 FPGA. In this technique Carry Look Ahead Adders were used to reduce the partial products. By using carry look-ahead adder (CLA) in Wallace Tree reduction method for addition, the delay has reduced further than Wallace Tree Adders. In 2018, Prashant R. Deshmukh et al., [13] have presented “Development of Concurrent Architecture of Vedic Multiplier” on Virtex-5 FPGA, wherein to reduce the overall time delay and area overhead a novel data base analyzer has included with built in self test block. The pre-computed values are stored in database. If the input samples are matched with value in local database the performance of the multiplier is outstanding. If the matching probability is less than 50% then the emphasis is given specifically on optimization of concurrent Vedic Multiplier architecture. In this method both memory based approach and a conventional multiplier are used. So, the mapping on to FPGA device is become complex; the performance of delay and area increases in worst case.

In 2017, ku. Dhamini et al., [2] have implemented high speed 16-bit Vedic and Booth Multiplier using a modified Carry Save Adder tree on Virtex-6 FPGA to reduce the partial products and to compare over the existing two techniques i.e.,Wallace and Booth Multiplier of 16-bit. In 2011, S. A. Shinde et al., [14] have implemented Booth Multiplier on Virtex-5 FPGA using Wallace tree reduction scheme to reduce the hardware utilization. By using Booth Multiplier the number of partial products is intended to reduce and the addition depth has reduced by using Wallace Tree reduction [15]. In 2017, Xiaoqing Liu et al., [16] had implemented an asynchronous 64-bit Booth Multiplier on Virtex-7 FPGA by isolating the functional blocks from control circuit therefore they made the complete circuit operate asynchronously to increase the replication of blocks to reduce power consumption.

The main drawback of memory based multipliers is the memory size and hence the accessing time, as the bit size increases the memory size has to increases linearly to cover all the combinations of multiplicand and multiplier. Hence mostly in FIR filter design a memory based multiplier is used to multiply a variable input with constant coefficient [17] so that pre-computed products of inputs is reduced. Promod Kumar Mehar [18], [19], [20] has implemented various techniques for LUT optimization to perform memory based multiplication. To the best of our knowledge the proposed multiplier is the one which adopts memory based approach when both multiplier and multiplicand are considered as variables yet utilizing small distributed memories.

The proposed architecture for multiplier has an optimized performance with respect to area and delay compared the existing multipliers discussed in this section. The detailed description of multiplier along with multi-operand adder used for partial product addition is explained in the further sections.

## 3. Multi-Operand Adder

The invention in patent mentioned above relates to architecture for 2-operand, 3-operand, 4-operand, 5-operand N-bit binary adder modules and also a method of interconnecting column array of these modules in stages to build architecture for addition of larger number of operands. The architecture of 4-operand adder is as shown in Figure 1. The novelty of this work is that ROMs are stored with binary representation of count of number of 1’s in their address. For example, if the input address to a $2^6 \times 3$ ROM shown in Figure 1 is ‘101011’, the value stored in that address location is ‘100’ representing the count of four 1’s in the specified address.

The complete architecture is built using only memories (ROMs) and registers, hence making the architecture suitable both for ASICs and FPGAs. The advantages of the proposed architecture also lie in hierarchal usage of the basic modules to extend it to sum a plurality of ‘K’ operands of any arbitrary size of N-bits each by just replicating the proposed 2-operand, 3-operand, 4-operand, 5-operand adder architecture modules supporting modularity and regularity.

For adding ‘K’ number of N-bit operands it requires $\left(\left\lceil \frac{N+\lceil \log_2 K \rceil}{3} \right\rceil + no.of\,stages - 1\right)$ clock cycles. Here one stage represents one complete set of column array modules for which inputs are given simultaneously. This approach is used to add all the partial products simultaneously. The

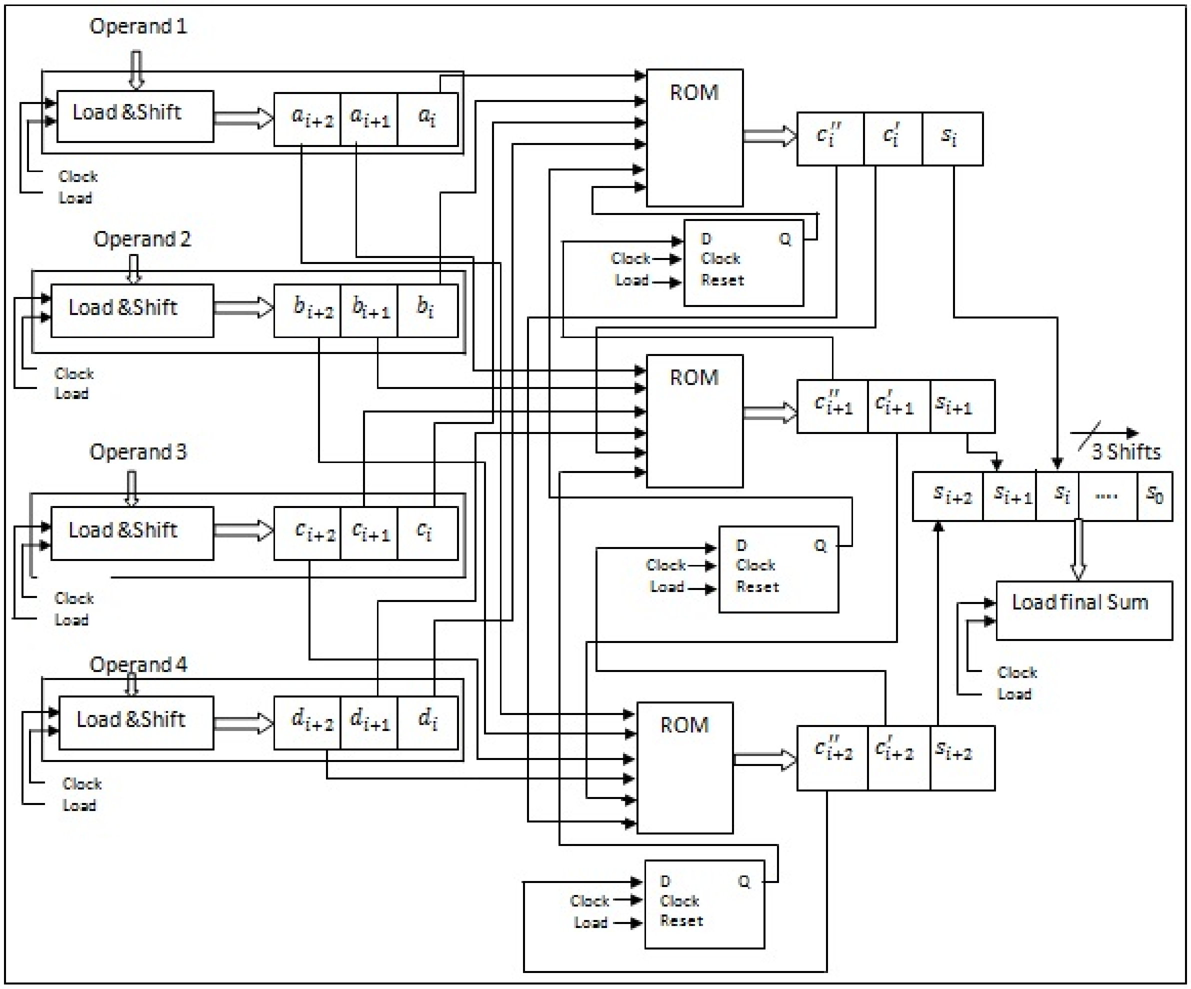
Operand 1
Load &Shift
$a_{i+2}$ $a_{i+1}$ $a_i$
Clock
Load
Operand 2
Load &Shift
$b_{i+2}$ $b_{i+1}$ $b_i$
Operand 3
Load &Shift
$c_{i+2}$ $c_{i+1}$ $c_i$
Operand 4
Load &Shift
$d_{i+2}$ $d_{i+1}$ $d_i$
ROM
$c''_i$ $c'_i$ $s_i$
D Q Clock Reset
$c''_{i+1}$ $c'_{i+1}$ $s_{i+1}$
3 Shifts
$s_{i+2}$ $s_{i+1}$ $s_i$ .... $s_0$
Load final Sum
$c''_{i+2}$ $c'_{i+2}$ $s_{i+2}$

Figure 1. Architecture of 4-Operand Adder

logic components, the interconnections between them which forms the architecture and also the method of its operation is same for the 4-operand, 3-operand, 5-operand, 2-operand adder modules, however the sizes of the ROM and the number of input shift registers varies with the number of operands. Hence the detail of the architecture along with the operating process of only 4-operand adder is illustrated through Figure 1.

### A. Method of Operation

Let 'K' number of binary operands of N-bit size each is taken and each binary operand is represented as $'b_{N-1}b_{N-2}....b_0'$ where $b_0$ is the *Least Significant Bit* (LSB), and $b_{N-1}$ is the *Most Significant Bit* (MSB). A binary digit $b_i$ represents a single bit with weight 'i'. Given a set of 'K' N-bit operands to add, all the bits from 'K' operands with the same weight form one column. In one clock cycle three columns of bits starting with the LSB are added using three ROMs along with three delay elements to get corresponding 3-bit sum. The process of adding three columns of bits at a time is repeated until all the columns of input operands are added using the same hardware. In this work, the inputs to the ROMs are column of bits with same weight. For example, to add four 8-bit operands, all the least significant bits of four operands form one column. In the same way from LSB to MSB it can be viewed as eight columns and a group of three columns are processed in one clock cycle. The four 8-bit operands are processed in three clock cycles and the carry generated from second, third clock cycles are added in fourth clock cycle, totally four clock cycles are required to get final sum.

## 4. PROPOSED UNSIGNED MULTIPLIERS

In the proposed multiplier, the main concern is to develop new architecture for partial product addition through the concept of multi-operand adders avoiding the complexity of deep pipelining built with multi-bit column compressors / counters called Generalized Parallel Counters (GPCs) [9] used in FPGAs and even with traditional carry save adders, carry Look-ahead adders, carry propagate adders used in ASIC that result in long carry chains leading to increase in power consumption, and extra area for

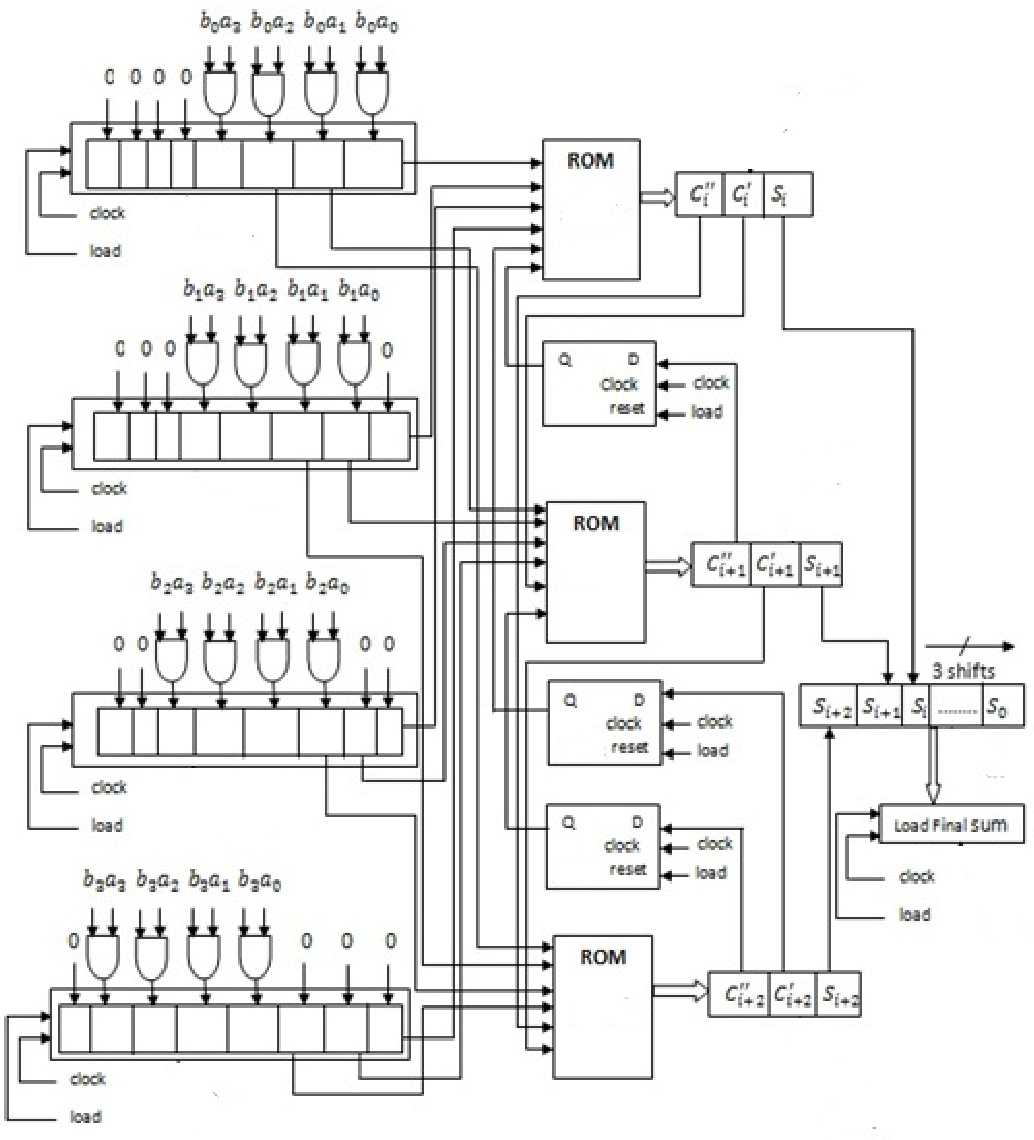
$b_0a_3$ $b_0a_2$ $b_0a_1$ $b_0a_0$
$b_1a_3$ $b_1a_2$ $b_1a_1$ $b_1a_0$
$b_2a_3$ $b_2a_2$ $b_2a_1$ $b_2a_0$
$b_3a_3$ $b_3a_2$ $b_3a_1$ $b_3a_0$
0
clock
load
ROM
$C_i''$ $C_i'$ $S_i$
$C_{i+1}''$ $C_{i+1}'$ $S_{i+1}$
$C_{i+2}''$ $C_{i+2}'$ $S_{i+2}$
Q D clock reset
3 shifts
$S_{i+2}$ $S_{i+1}$ $S_i$ ........ $S_0$
Load Final sum

Figure 2. Architecture of 4-bit Multiplier

pipelining. However, in this invention all these bottlenecks of conventional multipliers are addressed by using basic m-operand adders for m=2,3,4,5 to build the compressor tree instead to GPCs of FPGA and traditional adders of ASIC approach. An N-bit multiplier as can be interpreted from Figure 2. representing a 4-bit multiplier. The partial products are generated simultaneously by ANDing of the multiplicand with multiplier bits, with just by an AND gate delay through AND gate array. The N number of N-bit partial products are generated using NxN AND gate array.

The unoccupied Least Significant Bits (LSB) and Most Significant Bits (MSB) in the 2N-bit partial products as per their shifting are appended with zeros before loading them in to the input registers of m-operand adders using additional Flip-Flops in the Parallel in Serial Out Shift registers of multi-operand adder, resulting in a new architecture for the multiplier with slight increase in Flip- Flops but optimizing the overall area and delay/latency.

The 'N' number of partial products, considered as input operands, are grouped in sets of 'm' operands (m=2, 3, 4, 5), are given to respective m-operand adders forming the first stage called the input stage. For example, if there are 12 partial products from three 4-bit multipliers then one can use either three 4-operand adders (4+4+4) or four 3-operand adders (3+3+3+3) or any combination covering all 'N' partial products, however maintaining regularity in structure is given preference.

In every clock cycle an m-operand adder process 3-bits of all the 'm' operands starting from least significant bits

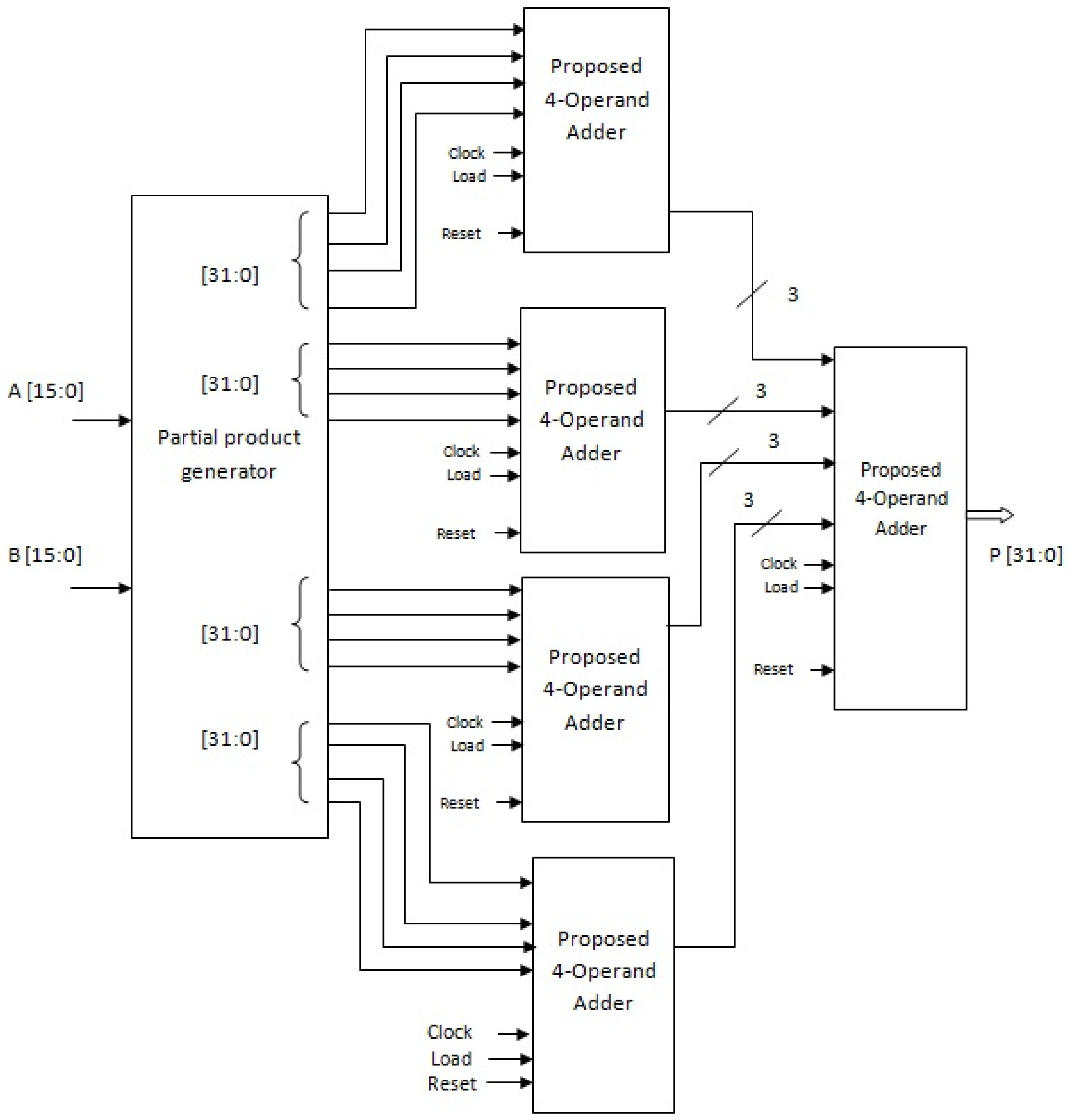

Figure 3. Architecture of 16-bit unsigned Multiplier

i.e. bits with weights $'i+2, i+1, i'$ at $i^{th}$ iteration. All the bits with the same weight form one column. In one clock cycle three columns of bits are added in set of any of the above said m-operand adder modules to generate 3-bit sum output $'s_{i+2}, s_{i+1}, si'$ at $i^{th}$ iteration. These 3-bit outputs stored in the pipeline registers, are grouped again and given as inputs to next stage of set of m-operand adders, during the addition of these bits in this stage called intermediate stage, the next three least significant bits with weights $'i+5, i+4, i+3'$ the all input operands are added in the first stage. Similarly, all the 2N-bits are added in a pipelined manner with required multiple stages. However, in each clock cycle the final stage shift the 3-bit sum to a SIPO shift register, giving the final sum after completing all the 2N-bits at the input in $\left(\left\lceil \frac{2N}{3} \right\rceil + no.of\,stages - 1\right)$ clock cycles to be loaded in to readout register, which represent the 2N-bit output of multiplier. The resources are greatly reduced due to serial-parallel architecture of 'm-operand' adder modules built with small memory blocks.

Coming to pipelining, as three columns of bits are processed at a time using multi-operand adders instead to multi-bit counters the area overhead is less compared to conventional multipliers and more over as the carry

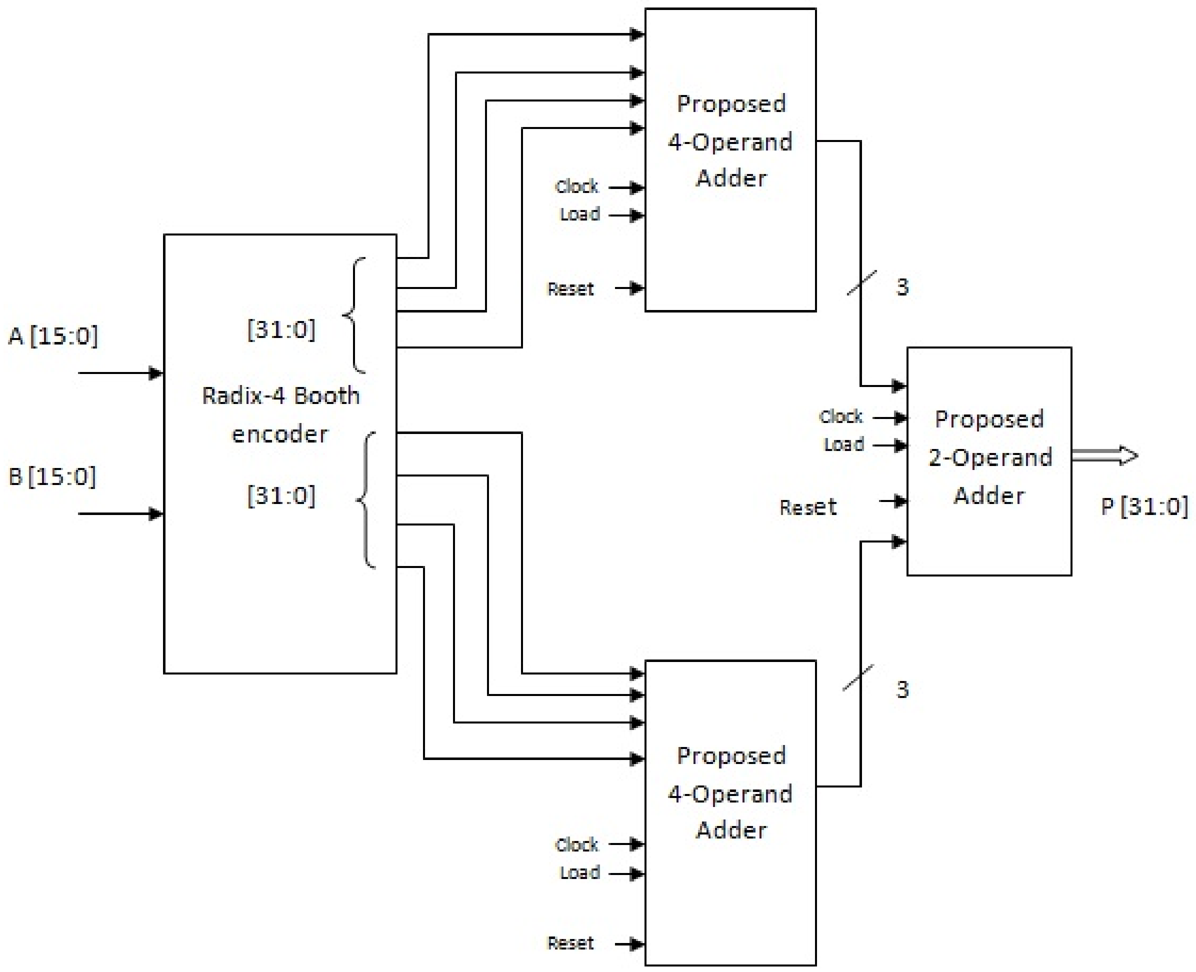

Figure 4. Architecture of 16-bit Signed Multiplier

propagation is confined to individual m-operand adder modules, long carry propagation is avoided leading to low power consumption and less complexity in pipelining. The resources can be further reduced if high radix approach of partial products generation is used instead to AND gate array, so that a less operand adder is required for further process. High radix approach is must for signed numbers.

## *A. 16-bit Unsigned Multiplier*

To make the invention more understandable, architecture of '16-bit multiplier' is analyzed further as an example. The block diagram of the architecture for '16-bit multiplier' is shown in Figure 3. To perform multiplication operation on 16-bit operands, multiplicand and multiplier are given to 16 x 16 AND gate array. Sixteen partial products of 32-bit each are generated simultaneously. These 16 partial products are loaded onto the Parallel-In-Serial-Out shift register of 16-operand adder as input. The 16-operand adder includes four 4-operand adder modules in the input stage to cover all the operands as inputs; and one 4-operand adder module at the output stage. In general each 4-operand adder module at the input stage receives three bits from each four input operands starting from the LSB position and generates 3-bit sum output in each clock cycle; hence four 3-bit outputs are generated from input stage. These four 3-bit outputs from the input stage are grouped as sets of four operands, and each set of four operands are given as input to one of the 4-operand adder modules in the following stage, which once again generate four 3-bit outputs after second clock cycle, these four 3-bit outputs are given as input operands to one 4-operand adder modules at the output stage to get sum output of three column of bits of all operands after the third clock cycle, this process repeats until the thirty two column of bits are added to get the final sum in 12 clock cycles. To be more specific if the input stage is processing input operands to generate three intermediate sum bits in $i^{th}$ clock cycle, then the intermediate stage will be processing the output bits of input stage in $(i + 1)^{th}$ clock cycle to generate another three intermediate sum bits, and then the output stage will process the output bits of intermediate stage in the $(i + 2)^{th}$ clock cycle in a pipeline manner.

TABLE I. Comparison of Unsigned Multiplier (8-bit, 16-bit, 32-bit) in terms of Area (LUTs), Delay (ns)

| Multiplier Type | FPGA Device | Input Bit Size | | | | | |
|---|---|---|---|---|---|---|---|
| | | 8-bits | | 16-bits | | 32-bits | |
| | | LUT | Delay (ns) | LUT | Delay (ns) | LUT | Delay (ns) |
| Array Multiplier A. Singh et al., [1](2018) | Artix-7 xc7a200-3sbg484 | 132 | 45.743 | 520 | 61.429 | ___ | ___ |
| Wallace Multiplier A. Singh et al., [1](2018) | | 164 | 26.264 | 683 | 44.870 | ___ | ___ |
| Dadda Multiplier A. Singh et al., [1](2018) | | 207 | 25.471 | 610 | 45.363 | ___ | ___ |
| Wallace Multiplier using CLA A. Singh et al., [1](2018) | | 191 | 24.350 | 541 | 40.131 | ___ | ___ |
| Improved Wallace Tree Multiplier Mani Kunnathettu et al., [11](2016) | | 128 | 11.314 | ___ | ___ | ___ | ___ |
| Proposed Unsigned Multiplier | | 126 | 13.804 | 457 | 23.664 | 1666 | 47.328 |
| Vedic Multiplier R Deshmukh et al., [13] | Virtex-5 xc5vlx50t-ff1136-3 | ___ | 13.252 | 556 | 29.068 | 1561 | 59.643 |
| Proposed Unsigned Multiplier | | 118 | 15.75 | 427 | 27 | 1626 | 54 |

## B. *16-bit Signed Multiplier*

The proposed Signed Multiplier has a booth encoder (radix-4) for generation of partial products; addition of partial products is made with similar architecture and working principle as that of unsigned. A block diagram for a 16-bit signed multiplier is shown in Figure 4 for illustration. The input multiplicand and multiplier operands are represented in 2's complement form and the final product also generated in 2's complement form. In an N-bit multiplier, N/2 partial products are generated and are added by using multi-operand Adder. As the number of partial products generated is half of that of the unsigned multiplier, it has one stage less than that of unsigned multiplier thus reducing the latency by one clock cycle thus still enhancing the performance in terms of Area-Delay product. The number of clock cycles required to get the product output is equal to $\left(\left\lceil \frac{2N}{3} \right\rceil + no.of\,stages - 1\right)$.

The proposed modular architecture for 16-bit signed multiplier has eight partial products of 32 bits each and is added by using 8-operand adder in two stages, and as per defined formula, after 12 clock cycles the SIPO is read to the output 32-bit product register. Similarly, the concept can be extended to N-bit multiplier.

## 5. RESULTS AND DISCUSSION

The synthesis and implementation of proposed architectures for all the multipliers discussed are done using `Virtex-5`, `Virtex-6`, `Artix-7` FPGA families in order to make apt comparison with the existing multipliers. As the choice of FPGA vendor and family can significantly influence the design architecture because of FPGA internal architecture and resources available, the comparison is made with existing methods which are implemented with same FPGA family. The performance of the proposed multiplier is compared with most recent state of art architectures for multipliers namely Wallace Multiplier using CLA by A. Singh et al., [1], Improved Wallace Tree Multiplier by Mani Kunnathettu et al., [11], Concurrent Vedic Multiplier by R Deshmukh et al., [13], Booth Asynchronous Multiplier by Xiaoqing Liu et al., [16]. The parameters considered for comparison are area (in terms of LUTs) and delay/latency i.e. Area-Delay product, percentage of reduction in LUTs and delay w.r.t. highest utilizing method. In the results tabulated here some slots are kept as '___' , this indicate that the existing method in the literature has not implemented multiplier with those specifications.

The designs are coded in VHDL and verified the functionality by applying different input vectors using Xilinx ISE 14.7. Component based coding strategy is used to have a better control over mapping. Analysis is done for each basic module to verify the functionality. The synthesis result of the proposed Signed, Unsigned multiplier is tabulated for input operands of 8-bit, 16-bit, 32-bit word sizes. As our architecture supports modular design, it can be extended to plurality of operands of 64, 128 bit sizes also. For fair

TABLE II. Performance of proposed Signed Multiplier (8-bit, 16-bit,32-bit,64-bit) in terms of Area(LUTs), Delay(ns)

| FPGA Device | Input Size | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 8-bit | | 16-bit | | 32-bit | | 64-bit | |
| | LUTs | Delay (ns) | LUTs | Delay (ns) | LUTs | Delay (ns) | LUTs | Delay (ns) |
| Virtex-6 (XC6VCX75T-2FF84) | 107 | 11.676 | 480 | 23.352 | 1436 | 46.70 | 5609 | 91.464 |
| Virtex-7 (xc7vx550tffg1158-2) | 111 | 10.272 | 488 | 20.544 | 1919 | 39.38 | 7560 | 80.464 |

TABLE III. Comparison of Signed Multiplier (8-bit, 16-bit, 32-bit,64-bit) in terms of Area(LUTs), Delay(ns)

| Multiplier Type | FPGA Device | Input Size | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 8-bit | | 16-bit | | 32-bit | | 64-bit | |
| | | LUT | Delay (ns) | LUT | Delay (ns) | LUT | Delay (ns) | LUT | Delay (ns) |
| Booth-Wallace S.A.Shinde et al. [14] (2011) | Virtex-5 (xc5vlx50 -3ff324) | 222 | 7.405 | ——— | ——— | ——— | ——— | ——— | ——— |
| Proposed Signed Multiplier | | 97 | 13.5 | ——— | ——— | ——— | ——— | ——— | ——— |
| Radix-4 Booth Ku. Daminiet al. [2] (2017) | Virtex-6 (XC6VCX75T -2FF84) | ——— | ——— | 752 | 22.12 | ——— | ——— | ——— | ——— |
| Proposed Signed Multiplier | | ——— | ——— | 480 | 23.352 | ——— | ——— | ——— | ——— |
| Booth Asynchronous Xiaoqing Liu et al. [16] (2018) | Virtex-7 (xc7vx550t -2ffg1158) | ——— | ——— | ——— | ——— | ——— | ——— | 3814 | 180 |
| Proposed Signed Multiplier | | ——— | ——— | ——— | ——— | ——— | ——— | 7560 | 80.46 |

comparison the proposed multiplier architectures for above said multipliers are implemented on the same FPGA devices as that in the existing literature taken as reference in this work.

## A. Performance of Unsigned Multiplier

The performance of proposed unsigned multiplier is compared with existing unsigned multipliers given in [11], [1], [13], and results of comparison are tabulated in Table I. From the results in Table I, it is observed that the proposed architecture utilizes least number of LUTS and has minimum delay. There is an exception for improved Wallace [11], in which for 8-bit multiplier LUTs are increased by 1.5%, and reduction in delay by 18.03% than proposed multiplier. However for higher bit size multipliers our approach outperforms as the clock cycles reduces due to serial – parallel architecture. In terms of LUTs the proposed method utilize 34.03%, 15.52% less LUTs than that of best existing multiplier reported in [1] and in terms of delay our architecture achieves 43.31%, 41.033% of reduction for 8-bit and 16-bit multiplication operation respectively. Synthesis and implementation of the proposed multiplier are done using `'XC5VLX50T- FF1136-3'` device from `Virtex-5` FPGAs for fair comparison with existing Concurrent Vedic Multiplier which was implemented by R Deshmukh et al. [13] in 2018 using the same FPGA device package. The performance of proposed multiplier in terms LUTs, time delay is reported at the bottom of Table I for 8-bit, 16-bit and 32-bit word size multiplications.

The proposed multiplier is more effective for higher word size multipliers as the design is aimed to compress large number of partial products with less time. So, it is well suited for applications with higher word length multiplications. The proposed unsigned multiplier is also compared with recent implementation of Concurrent Vedic Multiplier

[13] using `Virtex-5 xc5vlx50t-3ff1136FPGA` device.

The formula for percentage of reduction is given by $\left(\frac{\text{existing value}-\text{new value}}{\text{existing value}}\right) * 100$. The negative values given in the table indicates the parameters i.e. area and/or delay is increased in percentage over the existing method and positive values indicates the percentage of reduction over existing method. Comparing the Concurrent Vedic Multiplier to ours the delay is 18% higher for 8-bit, however the delay is best for the proposed i.e. it is reduced by 7.11%, 9.46% with a slight increase in LUTs for 16-bit, 32-bit respectively.

### B. Performance of Signed Multiplier

The proposed Signed Multiplier is implemented on `Virtex-5`, `Virtex-6`, `Virtex-7` FPGA for 8-bit, 16-bit, 32-bit, 64-bit and its performance is made in terms of utilized LUTs, delay is tabulated in Table II, the comparison of the proposed multiplier with existing Booth multipliers given in [2], [14], [16] is tabulated in Table III. From the performance table, it is observed that delay of proposed signed multiplier is reduced for higher bit size multiplication due to higher frequency of the architecture. The proposed 8-bit signed multiplier is compared with the Booth – Wallace Multiplier [14] implemented on `Virtex-5 (xc5vlx50-3ff324)` FPGA. It utilizes 56.30% less LUTs, with increase in delay, however when bit size increased to 16-bits the delay is approximately same, but LUTs remains less.

The proposed 16-bit signed multiplier is compared with Radix-4 Booth multiplier implemented on `Virtex-6 (XC6VCX75T-2FF84)` FPGA device by Ku. Damini et al. [2]. It utilize 36.17% less LUTs with 5.5% increase in delay. But there is a drastic fall in delay for the proposed 64-bit multiplier compared to existing multiplier [16] by 55.3%. But in parallel the LUT's are increased to almost double of existing 64-bit multiplier i.e existing multiplier has less LUT than the proposed multiplier by 49.55%. Therefore the proposed multiplier for lower bit length to higher bit length the delay is reducing drastically and area is increasing compared to existing multipliers. But when compared to the existing multipliers the proposed multiplier has less area-delay product by 20.34%, 32.618%, 11.39% for 8-bit, 16-bit and 64-bit respectively.

## 6. CONCLUSION

The proposed multiplier architecture made of memory blocks and registers is equally efficient for both ASIC and FPGA implementation, more over as carry propagation is limited to individual m- operand adder modules, the bottlenecks of long carry chain and complexity of routing to pipelining as in conventional multipliers is removed. The registers at the input and output of the m-operand adder modules can be used to take the advantage of parallelism and pipelining of data. The architecture of our multiplier is planned to involve this advantage in the addition of partial products, resulting in serial-parallel architecture with pipelining. This made our multiplier outperform existing architecture as the bit size increases.

## References

[1] A. Singh, A. Sharma, and P. Kumari, "Fpga Implementation of High Speed Multiplier with Optimized Reduction Phase," in *Intelligent Communication, Control and Devices.*, ser. Advances in Intelligent Systems and Computing, vol. 624. Singapore: Springer Nature, Apr. 2018, pp. 187–195.

[2] C. Damini Dandade and R. Prashant Indurkar, "Design of High Speed 16-Bit Vedic and Booth Multiplier," *International Journal for Research in Applied Science & Engineering Technology*, vol. 5, no. VIII, pp. 689–696, 2017.

[3] C. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 14–17, Feb. 1964. [Online]. Available: https://ieeexplore.ieee.org/document/4038071

[4] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 45, no. 5, pp. 349–356, 1965.

[5] K. Bickerstaff, M. Schulte, and E. Swartzlander, "Reduced area multipliers," in *Proceedings of International Conference on Application Specific Array Processors (ASAP '93)*, Oct 1993, pp. 478–489.

[6] A. Meo, "Arithmetic networks and their minimization using a new line of elementary units," *IEEE Transactions on Computers*, vol. C-24, no. 3, pp. 258–280, March 1975.

[7] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Efficient Synthesis of Compressor trees on FPGA," in *2008 Asia and South Pacific Design Automation Conference*. Seoul, Korea (South): IEEE, Mar. 2008, pp. 138–143.

[8] T. Matsunaga, S. Kimura, and Y. Matsunaga, "An Exact Approach for GPC-Based Compressor Tree Synthesis," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E96-A, no. 12, pp. 2553–2560, Dec. 2013.

[9] M. Kumm and J. Kappauf, "Advanced compressor tree synthesis for fpgas," *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1078–1091, Aug 2018.

[10] S. Aruna Mastani, S. Kannappan, "Memory based Multi-Operand Adder," Indian patent application 201 941 024 136, june 18, 2019.

[11] M. Kunnathettu Rajee, T. Thomas, A. Manuel, A. Rachel, and R. Cheriyan, "Fpga Implementation of an Efficient High Speed Wallace Tree Multiplier," *International Journal of Computer Applications*, pp. 9–14, 2015. [Online]. Available: https://research.ijcaonline.org/icettas2015/number1/icettas2565.pdf

[12] K. M. Gaikwad, M. S. Chavan, "Analysis of Array Multiplier and Vedic Multiplier using Xilinx," *Communications on Applied Electronics (CAE)*, vol. 5, no. 1, pp. 13–16, May 2016. [Online]. Available: https://www.caeaccess.org/archives/volume5/number1/gaikwad-2016-cae-652140.pdf

[13] P. R. Deshmukh and J. S. Edle, "Development of concurrent architecture of vedic multiplier," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. pune, India: IEEE, 2018, pp. 1–6.

[14] S. A. Shinde, R. K.Kamat, "Fpga based improved hardware implementation of booth wallace multiplier using handel c," *Elektronika Ir Elektrotechnika*, vol. 109, no. 3, pp. 71–74, Mar. 2011.

[15] S. B. Suhas, S. Ramakrishna, and S. B. Rajashekar, "Design and implementation of adders and multiplier in fpga using chipscope:

A performance improvement," in *Proceedings of Information and Communication Technology for Competitive Strategies*, ser. Lecture Notes in Networks and Systems, vol. 40. Springer, Aug 2018, pp. 11–19.

[16] X. Liu, A. He, C. Li, G. Feng, and J. Zhang, "Study of 64-bit booth asynchronous multiplier based on fpga," in *2017 IEEE 12th International Conference on ASIC (ASICON)*, 2017, pp. 263–266.

[17] P. K. Meher, "New approach to look-up-table design and memory-based realization of fir digital filter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 3, pp. 592–603, March 2010. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5356201&isnumber=5424135

[18] P. K. Meher, "Lut optimization for memory-based computation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 4, pp. 285–289, April 2010. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5447668&isnumber=5452111

[19] P. K. Meher, "New approach to lut implementation and accumulation for memory-based multiplication," in *2009 IEEE International Symposium on Circuits and Systems*, May 2009, pp. 453–456.

[20] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in *Proceedings of the 2009 12th International Symposium on Integrated Circuits*, Dec 2009, pp. 663–666.

**S. Aruna Mastani** S. Aruna Mastani received the B.Tech. and M.Tech. degrees in Electronics and Communication engineering from Jawaharlal Nehru Technological University Anantapur(JNTUA), Ananthapuramu, Andhra Pradesh, India. She was awarded with Ph.D. degree in Image Processing from JNTUA, Ananthapuramu, Andhra Pradesh, in 2008. She is working as Assistant Professor in the department of Electronics and Communication Engineering, JNTUA, Ananthapuramu., India. She has 20 years of experience in teaching for Electronics and Communication Engineering. She filled three complete Indian Patent applications. One of the patent applications is published in the Journal of Indian Patent and is in the process of examination by the Indian Patent office. She published research articles in 35 international journals. Being a supervisor she is guiding 8 Ph.D scholar students. Her areas of interest covers Image Processing, Very Large Scale Integration, VLSI circuits and systems.

**S. Kannappan** S. Kannappan was born in Sathrawada village, Nagari, Andhra Pradesh, India in 1987. He successfully completed the Ph.D VIVA defence on december 2020 in Jawaharlal Nehru Technological University Anantapur(JNTUA), Andhra Pradesh. His research area is VLSI circuits and systems and more particularly design of digital circuits and systems, design of Processor. He received the B.Tech degree in Electronics and Communication Engineering from Narayana Engineering College, Nellore, Andhra Padesh. He received M.Tech degree in Digital Electronics and Communication Systems fron JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh. He filled two complete Indian patent applications along with Dr. S. Aruna Mastani, Assistant Professor, JNTUA, Ananthapuramu, Andhra Pradesh. He publicshed papers in Five International journals and presented papers in three international conferences.