FPGA Implementation of High Performance Accurate and Approximate Signed and Unsigned Multipliers using Structure of LUT configurations

Saravanan V.[1,*], Elarmathi S.[2], Rajalakshmi V. R.[3]

[1]Associate Professor, Department of ECE, Knowledge Institute of Technology, Salem, Tamil Nadu, India

[2]Assistant Professor, Department of ECE, Knowledge Institute of Technology, Salem, Tamil Nadu, India

[3]PG Scholar, Department of ECE, Knowledge Institute of Technology, Salem, Tamil Nadu, India

Emails: vsece@kiot.ac.in; seece@kiot.ac.in; 2k22vlsi12@kiot.ac.in

Abstract

A recent study examined the applications of multiplication and division in video and image manipulation and there has been mention of machine learning. DSP blocks that function as high performance multipliers are given by FPGA providers. However routing lag time and inefficiencies, particularly for lower bit width multiplications, might emerge from the fixed placements and restricted number of these FPGA multipliers, raising power consumption. FPGA companies offer IP cores that are soft made especially for multiplication to solve this problem. Even if these IP cores have improved over time, they can yet be improved. This can be accomplished by creating low latency, accurate, and core multiplier topologies that maximize the space of FPGA and take advantage of its architectural characteristics, like rapid carry chains and look up table structures. These architectures seek to improve overall efficiency by lowering the crucial path delay and multiplier resource consumption. Here a proposed method for building accurate and approximate signed and unsigned multipliers for an eight bit configuration is presented. This entails changing the LUT 6 architecture to use a one LUT 5 with multiplexers in place of a dual LUT 5 with multiplexers. Using Xilinx software, the design was built in Verilog HDL and synthesized. At the conclusion of the process, variables including area, delay and power were compared.

Keywords: FPGA; Multipliers; Power consumption; LUT; HDL

1. Introduction

A basic mathematical operation used frequently in processing of digital images and signal is multiplication. Manufactures of FPGAs like Xilinx provide specific DSP blocks for effective multiplication. Even while these blocks in DSP offer excellent performance, for some uses they might not always be the best option like efficiency and resource use. Here compares the use of crucial path delays and lookup tables in two distinct implementations of JPEG and Reed-Solomon encoders on Virtex-7 series FPGAs with Xilinx Vivado. Implementing the DSP based Reed-Solomon encoder has become more delayed as a result of routing delays resulting from the DSP blocks' location. In some cases, manual floor planning may be necessary for smaller applications. It is crucial to maximize the FPGA resource allocation to increase application performance. However, in the case of complex applications that have conflicting resource requirements, manual optimization is not be feasible to maximize performance enhancements. For instance, when implementing a JPEG encoder, a significant portion of the accessible DSP blocks (56%) are utilized. This can lead to a depletion of DSP blocks, limiting their availability for critical operations in other applications running simultaneously on the similar FPGA. This highlights the necessary for LUT-based multipliers to address resource constraints and improve overall application performance. Similar findings on DSP block utilization and application performance have been documented by other researchers. Therefore, incorporating soft multipliers based on logic in addition to DSP blocks is crucial for achieving optimal

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

performance across various implementation scenarios. This is why leading companies like Intel and Xilinx offer soft multipliers based on logic as part of their FPGA solutions [1].

Methods such as utilize a modular approach to create larger FPGA-based multipliers by utilizing smaller blocks. While effective for small bit-width multipliers, these techniques can be resource-intensive for larger bit-width multipliers. For instance, implementing an 8x8 multiplier on a Virtex-7 FPGA using default synthesis in vivado settings requires 61 LUTs [2]. In contrast, a modular approach using 4x4 multipliers to create an accurate 8x8 multiplier consumes 82 LUTs. Researchers Walters [12] and Kumm, as well as Parandeh-Afshar [7] and their team, have investigated the applications of altered Booth algorithm for Altera and FPGAs respectively for effective radix 4 multiplier designs. Walters and Kumm utilized carry chains and look up tables with 6 inputs in their Xilinx FPGA implementation to stay away from partial products trees, but faced challenges with significant delays in the crucial path. On the other hand, Parandeh-Afshar and team employed Booth's and BaughWooley's algorithms on Altera FPGAs for area-efficient multiplier designs, limiting the duration of modules using adaptive logic to five to reduce carry chain lengths. However, this resulted in underutilization of FPGA resources. The limitation of five ALMs in the carry chain was not feasible with other FPGA vendors like Xilinx. Additionally, while employing Altera FPGAs to put forth compressor of partial product tree encountered difficulties when attempting to create a generic parallel counter.

Among the various options for designing multipliers, traditional shift-and-add, serial, and serial/parallel multipliers prioritize minimizing space requirements but suffer from lengthy delays on the crucial paths. On the other hand, popular parallel multipliers like Wallace and Dadda designs demand more area to achieve faster output times through parallel processing of partial products. By examining ASIC-based systems now have access to a novel multiplier architecture that uses Radix-4 recoding to combine the best aspects of the Wallace/Dadda and Booth multiplier approaches. This work shows that using effective techniques for encoding and reducing partial products can improve the performance of these software based logic multipliers [3]. For example, our suggested accurate multiplier design reduces critical path delays to 52%, which is better than current implementations.

2. Related Work

Recently, approximate computing has gained popularity as a viable method for designing digital systems with minimal energy use. The foundation of approximate computing is the tolerance of many systems and applications for a certain amount of loss of optimality or quality in the computed result. Improved energy efficiency can be achieved significantly with approximation computing approaches, as they eliminate the need for highly accurate or deterministic operations. The design of approximation arithmetic blocks, relevant error and quality measurements, and algorithm-level approaches for approximate computing are among the recent advancements in the field that are reviewed in this work.

Martin Kumm et al. (2018) developed a method to massive multiplier design for FPGAs that maximizes resource usage [4]. These are made up of smaller multipliers, which can be logic-based multipliers. The problem's viable solutions are described using a previously suggested multiplier tiling methodology. After that, the issue is framed as an integer linear programming (ILP) problem, which is solvable using conventional ILP solvers. It can be applied to exchange the LUT cost against the DSP cost or to reduce the overall implementation cost. The optimal solutions be discovered for most practical multiplier sizes up to 64×64, though the problem is NP complete [5-6]. Slice reductions of up to 47.5% are observed in synthesis studies on appropriate multiplier sizes when compared to the most advanced heuristic methods.

Pilipović et al. (2020) developed the logarithmic multiplier design with radix-4Booth encoding [8]. These advantages come at cost of decreased accuracy. RCCMs multiply input values through constrained selection of coefficients with adders, subtractors, bit shifts, and multiplexers (MUX). These RCCMs adapted to FPGA logic elementsto guarantee its efficient use. To diminish the loss of quantification information, training techniques were developed. This allows the use of RCCMs on hardware, maintaining high precision. The advantages of processes with AlexNet, ResNet-18, and ResNet-50 networks. The outcoming performance accomplish up to 50% resource savings in contrast to conventional 8bit quantized networks, resulting in major accelerations and power savings. The minimum resource requirement outperforms 6-bit fixed point precision, as the entire performances through RCCM realize precision at least equal with 8-bit uniformly quantized design.

Ullah et al. developed approximate multipliers in 2021 for FPGA-based hardware accelerators. Improved designs are required for high performance in FPGAs with soft multiplier IP cores [9]. By taking advantage of the lookup table structures (LUT) and fast carry chains found in FPGAs, this least latency, precise and approximate soft core multiplier optimized for generic areas reduces resource and general delay of the critical route application of multipliers. The multiplier sizes, equivalent to Xilinx's Logi CORE IP Up to 23% and 55% less LUT usage is possible with unsigned and signed architecture, respectively, for various multiplier. Additionally, by achieving

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

negligible loss at output precision comparable to Logi CORE IP, unsigned approximation multiplier architectures may reduce critical path time by up to 51%.

Liang et al. (2018) Developed a binarized neural network (BNN) for FPGAs, radically reduces hardware consumption as maintaining acceptable precision [10]. One method of resource-aware model analysis (RAMA) is to eliminate restricted access to bit-level XNOR multipliers and variable operations, and bottleneck of parameter access through data quantization and optimized storage on chip. This evaluates the FP-BNN accelerator designs for MNIST multi-layer Perceptrons (MLP), Cifar-10 ConvNet, and AlexNet on Stratix-V FPGA system.

Mittal (2020) Developed Convolutional neural networks can benefit from FPGA-based accelerators to improve integer and binary operations in FP operations and boost system performance. In this case, the multiplier was utilized to lower the quantity of multiplications throughout the convolution process [11]. This algorithm maximizes that number of additions, the total cost was diminished based on greater relative overhead of multiplier likened to adder. They realize that1D CONV engine under the form of a four-stage pipeline.

Faraone et al. (2019) Developed Addnet: Neural networks with multipliers optimized for FPGAs [13]. In this method, the proven reconfigurable constant coefficient multipliers (RCCM) suggest best option to save silicon area using low precision arithmetic. RCCMs multiply input values through constraint selection of coefficients with only adders, subtractors [14-15], bit shifts, and multiplexers (MUX). To lessen the loss of quantification information, developed novel training processes map the probable coefficient representations of RCCMs. To reveal that advantages of techniques with AlexNet, ResNet-18, and ResNet-50networks. This RCCM of minimum resource requirements outperforms 6-bit fixed point precision, as the entire executions using RCCM accomplish accuracy at least equal to 8-bituniformly quantized design.

3. Existing System

As a fundamental mathematical operation that is frequently employed in the within the domain of image and digital signal processing, multiplication plays a significant part in the process. Multipliers may be quickly created with blocks of DSP from FPGA providers like Intel and Xilinx. Even while these blocks of DSP which are reasonably priced, their excellent performance does not necessarily translate into efficiency in terms of overall effectiveness and the amount of space needed for particular uses. The positioning of DSP blocks causes routing delays, which significantly increase latency in the implementation based on DSP of the Reed-Solomon encoder. For smaller applications, manual floor planning can improve overall performance by reducing size. However, for more complex applications with competing FPGA resource needs, manual optimization might not be feasible to enhance performance advantages, especially in embedded applications. A large percentage of the DSP blocks that are available are used in the JPEG encoder implementation, which may limit their availability for other performance-critical processes on the same FPGA. This may require the use of LUT-based multipliers to compensate. To improve overall performance in different implementation scenarios, a strategy incorporating logic-based soft multipliers and DSP is crucial. Both Xilinx and Intel offer logic-based soft multipliers to address this need and enhance program performance.

To construct larger FPGA-based multipliers with smaller blocks, new techniques are being developed. These methods work better, though, with smaller bit-width multipliers. Larger bit-width multipliers ultimately result in more FPGA resource consumption. For example, 71 LUTs are used in the implementation of an 8x8 multiplier on the Virtex-7 FPGA utilizing a logic-based method with Vivado. However, 82 LUTs are required to create the same 8x8 multiplier using 4x4 multipliers. Walters and Kumm have developed area-efficient implementations of radix-4 multipliers with carry chains and 6-input LUTs on Xilinx FPGAs by utilizing a modified version of Booth's technique. Although their method avoids compressor trees for partial products, it causes longer crucial path delays. Parandeh-Afshar and associates have also utilized the multiplication algorithms of Booth and Baugh-Wooley to develop area-efficient multipliers on Altera FPGAs. Nevertheless, they restrict the number of adaptive logic modules to five in order to shorten the carry chains, which results in an underuse of FPGA resources.

Furthermore, a issue with the existing FPGAs from manufacturers like Xilinx is that carry chains can only support five ALMs without wasting resources. Using Altera FPGAs, Parandeh-Afshar and colleagues have also introduced a compressor tree for partial products. Nevertheless, their use of GPCs, or generalized parallel counters, results in an underutilization of LUTs in subsequent ALMs as shown in figure 1.

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

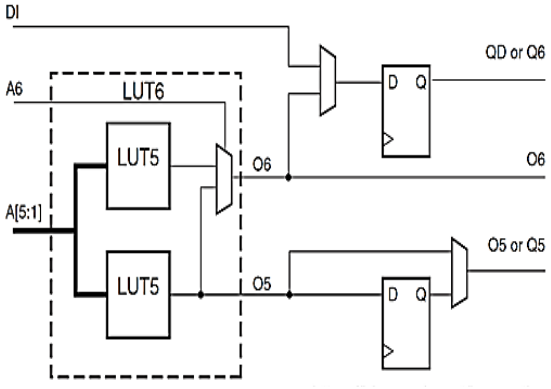

Figure 1. LUTs in subsequent ALMs

Among the various options available for designing multipliers, traditional shift-and-add, serial, and serial/parallel multipliers are known for their ability to meet low area requirements but suffer from high critical path delays. On the other hand, parallel multipliers based on Wallace and Dadda designs are effective in reducing output latencies by adding partial products in parallel, but they come with high area requirements. Recognizing the strengths and weaknesses of multiplier schemes such as Booth and Wallace/Dadda, a new hybrid multiplier architecture has been proposed for ASIC-based systems. Radix-4 recoding is used in this design to boost efficiency. The work shows that effective methods for encoding and reducing partial products can improve logic-based soft multipliers even more.

4. Proposed System

In a recent study, Multiplication and division are now often used in a wide range of applications, including machine learning and image/video processing. High-performance DSP blocks with multipliers are available from FPGA suppliers, however the quantity and arrangement of these blocks on FPGAs are constrained. This may result in inefficient lower bit-width multiplications and routing delays, which raise power consumption. FPGA companies offer soft IP cores made especially for multiplication to solve this problem. Even if the designs of these soft multiplier IP cores have evolved over time, more may be done to increase resource and performance efficiency. In this work, we suggest creating low-latency, approximate, accurate, and area-optimized soft-core multiplier architectures by utilizing FPGA architectural features including rapid carry chains and look-up table structures.

Our goal in doing this is to lower the multipliers' resource consumption and crucial path latency. Using a unique strategy, we build Approximate and Accurate Signed and Unsigned Multipliers for 8-bit setups. Additionally, we propose reworking the LUT6 architecture by substituting a single LUT5 with multiplexers for a dual LUT5 with multiplexers as shown in the figure 2. The suggested designs were synthesized with Xilinx software and implemented in Verilog HDL. The area, delay, and power consumption of the multipliers were used to evaluate their performance; the results were encouraging for potential future uses.

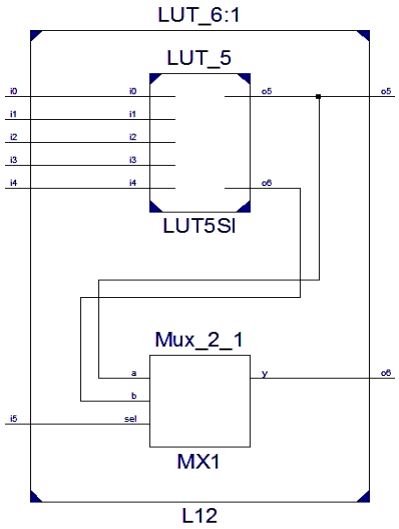

Figure 2. Proposed LUT 6 Slice Structure

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

The earlier discussed binary and ternary adders for adding partial products are included in the proposed design of a precise multiplier. Resource usage can be made more effectively by using ternary adders. However, because each element in the carry chain depends on the carry-generate signal from the previous cell, the efficiency of a ternary adder is reduced. For instance, an 8-bit ternary adder with three operands written on a Virtex-7 FPGA using Vivado has a 36% greater critical path latency than an 8-bit binary adder with two operands. Kumm et al. have observed similar results on the loss in performance in ternary adders with varying bit-widths. To address this issue, we employ various approximation techniques to create high-performance and resource-efficient approximate multipliers as shown in the figure 3 and figure 4.

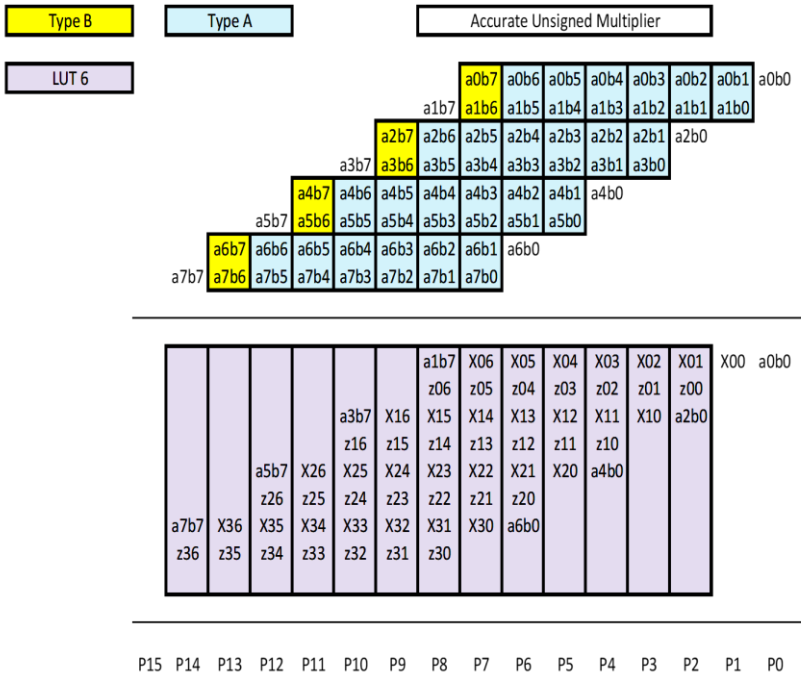

Figure 3. Proposed Unsigned_Accurate and Approximate Multiplier

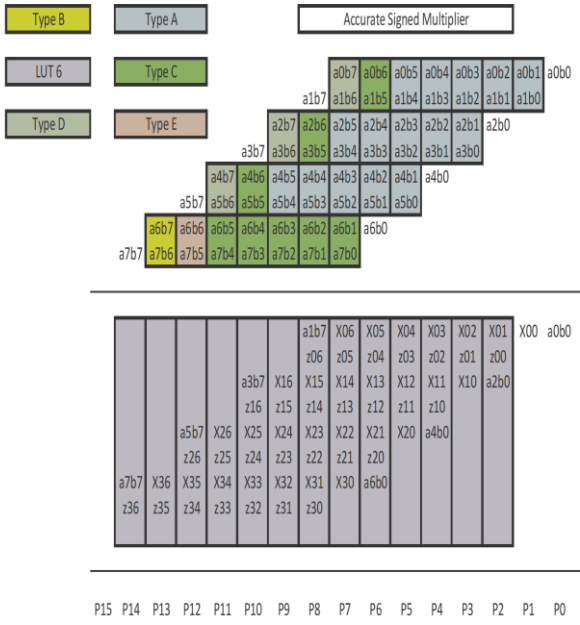

Figure 4. Proposed Signed Accurate and Approximate Multiplier

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

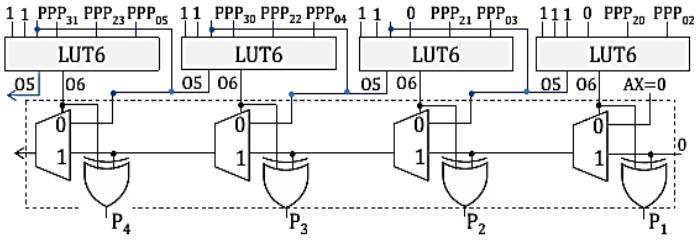

Figure 5. Ternary adder calculation of final product

The earlier discussed binary and ternary adders for adding partial products are included in the proposed design of a precise multiplier as shown in the figure 5. Resource usage can be made more effectively by using ternary adders. Nevertheless, a ternary adder's effectiveness is diminished since every component in the carry chain depends on the carry-generate signal from the preceding cell. For example, compared to an 8-bit binary adder (with two operands), an 8-bit ternary adder (with three operands) written on a Virtex-7 FPGA using Vivado has a 37% longer critical path latency. Kumm et al. have found similar results on the decline in performance in ternary adders with varying bit-widths and higher order multipliers with sub multipliers. In order to solve this problem, we develop highly-performing and resource-efficient approximate multipliers using a variety of approximation techniques. The construction of an approximate adder for creating multipliers of higher orders utilizing sub multipliers will be covered in the parts that follow, after which we will provide designs for simple approximation multipliers.

A. Total assistance of the work

A Precise Unsigned Multiplier Architecture: It refers to a digital circuit design that performs multiplication of two unsigned binary numbers with full precision, meaning it produces the exact result without any loss of accuracy and is area-optimized,. Generation of One Step Partial Products and their Addition: The generation of one-step partial products and their addition is a key part of the multiplication process in digital systems. In unsigned binary multiplication, one-step partial products are generated by ANDing each bit of the multiplier with the entire multiplicand. Each partial product is shifted left based on the position of the corresponding multiplier bit. After generating the partial products, we need to add them together to get the final product. The choice of addition method affects the speed and complexity of the multiplier.

Effective Partial Product Reduction Tree: A key element of high-speed multiplier designs is the effective partial product reduction tree, which aims to effectively decrease the number of partial products to two final rows that can be added with a fast adder. The Wallace Tree and Dadda Tree methods are the most often used techniques for this. Equation 1 tells us how many steps are required to find the exact product.

$$No.\ of\ stages\ =\ \left\lceil log_3(\frac{M}{2})\right\rceil + 1 \qquad (1)$$

Signed Accurate Multiplier: signed accurate multiplier is a digital circuit designed to perform multiplication of signed binary numbers while maintaining full precision in the result. The multiplier can handle both positive and negative numbers, typically represented in two's complement format. The result is mathematically exact, with no rounding or truncation errors and the circuit that performs multiplication. The main challenge in signed accurate multiplication is handling the sign bits correctly throughout the multiplication process while maintaining the efficiency and speed of unsigned multiplication techniques.

Unsigned Approximate 4x2 Multiplier: A foundational element in order to use higher-order approximation multipliers is an unsigned approximate 4x2 multiplier. The 6 inputs of a LUT of the most advanced FPGAs are fully utilized by the 4x2 approximate multiplier that has been suggested.

Unsigned Approximate 4x2 Multiplie: We build an asymmetric and approximate 4×4 multiplier by applying several FPGA-specific optimizations to the 4x2 approximate multiplier in order to minimize the amount of output errors.

An Estimated Ternary Adder for Calculating the Sum of the Approximate Partial Products Generated: Our recommended method for achieving these contributions is summarized in a Ternary Approximate Adder for Summation of the Approximate Partial Products that Are Generated in Figure 5. We demonstrate the proper multiplier design utilizing FPGAs' primary logic resources. Next, we examine the used three is to one compressors and employ more LUT-level optimization strategies to create distinct approximations of multipliers to lower the multiplier crucial path time. Ultimately, a comprehensive examination of the suggested multipliers' output performance and accuracy improvements over the most advanced multipliers is provided. For varying multiplier sizes, our approximate and accurate topologies reduce the whole number of LUTs used by up to 26% in contrast

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

to the LogiCORE IP from Xilinx IP that has been optimized for area. Furthermore, in contrast to The LogiCORE IP with area optimization, the suggested approximation architecture reduces the crucial path delay of the multiplier by up to 51%. The following quality metrics have been used for our proposed multipliers' error Characterisation: (a) The total number of errors; (b) The highest error magnitude; (c) The average relative error; and (d) The total number of maximum errors.

B. Slice Structure of Xilinx FPGA

Modern FPGAs, such those made available by Intel and Xilinx, create sequential as well as combinational circuits using 6-input LUTs. All the designs mentioned have been implemented using Xilinx FPGAs. Nonetheless, our suggested approach is universal and may be used to FPGAs from different suppliers, such Intel, which likewise employs carry chains and factorable 6-input LUTs. Xilinx FPGA slice structure is a fundamental building block in Xilinx FPGAs. The slice structure has evolved over different FPGA families. This slice typically contains Look-Up Tables, Flip-flops, multiplexers, carry logic, arithmetic logic and wide multiplexers. Key features of this structure are each element can be configured for different functions, 6-input LUTs can be split into two 5-input LUTs and here it is further reduced to one LUT, Ability to use LUTs and FFs independently, Dedicated routing within and between slices, Dedicated carry chain for fast arithmetic operations chain implements a carry-look ahead adder in accordance with Equations 2 and 3, using O5 as the carry-generate signal and O6 as the carry-propagate signal. The external bypass signals AX – DX can also provide the carry-generate signals for the carry chain.

$$S_i = P_i \oplus C_i \tag{2}$$

$$C_{i+1} = G_i + P_i \cdot C_i \tag{3}$$

C. Baugh-Wooley's Multiplication Algorithm

In contrast to unsigned multiplication, accurate product computation in a signed multiplication requires precise sign-extension of all partial products. The multiplication algorithm developed by Baugh-Wooley encodes the sign information in the partial products that are produced, so eliminating the requirement for sign-extension bit computation and communication.

$$\begin{aligned} A &= -a_{N-1}2^{N-1} + \sum_{n=0}^{N-2} a_n 2^n \\ B &= -b_{M-1}2^{M-1} + \sum_{m=0}^{M-2} b_m 2^m \end{aligned} \tag{4}$$

$$\begin{aligned} P = {} & a_{N-1}b_{M-1}2^{N+M-2} + \sum_{n=0}^{N-2}\sum_{m=0}^{M-2} a_n b_m 2^{n+m} \\ & - 2^{N-1}\sum_{m=0}^{M-2} a_{N-1}b_m 2^m - 2^{M-1}\sum_{n=0}^{N-2} b_{M-1}a_n 2^n \end{aligned} \tag{5}$$

$$\begin{aligned} P = {} & a_{N-1}b_{M-1}2^{N+M-2} + \sum_{n=0}^{N-2}\sum_{m=0}^{M-2} a_n b_m 2^{n+m} \\ & + 2^{N-1}\sum_{m=0}^{M-2} \overline{a_{N-1}b_m} 2^m + 2^{M-1}\sum_{n=0}^{N-2} \overline{b_{M-1}a_n} 2^n \\ & + 2^{N-1} + 2^{M-1} + 2^{N+M-1} \end{aligned} \tag{6}$$

$$\begin{aligned} P_8 = {} & a_7 b_7 2^{14} + \sum_{n=0}^{6}\sum_{m=0}^{6} a_n b_m 2^{n+m} + 2^7 \sum_{m=0}^{6} \overline{a_7 b_m} 2^m \\ & + 2^7 \sum_{n=0}^{6} \overline{b_7 a_n} 2^n + 2^8 + 2^{15} \end{aligned} \tag{7}$$

For a N × M signed multiplier, Equation 4 specifies the corresponding operands in the complement representation of 2. Equation 5 illustrates the procedure of generating signed partial products in order to determine the final product "P". By using Baugh-Wooley's multiplication method, the negative partial product terms are rewritten as provided in Equation 6, eliminating the requirement for explicit sign-extension bits. In the equation, "axby" stands for the 1's complement of the corresponding partial product term for x ∈ [0.. N − 1] and y ∈ [0.. M − 1. Equation 7 for an 8x8 signed multiplier, for example, expresses signed partial products using Baugh-Wooely's method.

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

6. Results and Discussion

A. Signed Accurate Multiplier

The implemented accurate signed 8x8 multiplier uses LUT6 (6-input lookup tables) which is employed using single LUT5s and multiplexers (MUXs) and the logic that could be employed for the execution are sign extension, booth's encoding, partial product generation, partial product accumulation and final addition and sign correction as shown in the figure 6.

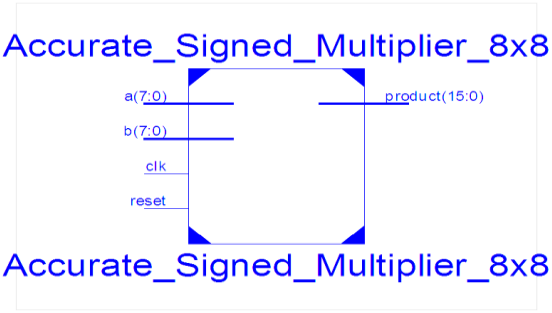

Figure 6. Signed Accurate Multiplier

B. Timing Report of Signed Accurate Multiplier

```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 14962 / 16
-------------------------------------------------------------------------
Delay:               13.818ns (Levels of Logic = 16)
  Source:            a<2> (PAD)
  Destination:       product<15> (PAD)

  Data Path: a<2> to product<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             17   0.694   0.736  a_2_IBUF (a_2_IBUF)
     LUT4:I0->O             2   0.086   0.666  TA12/Mxor_o6_Result1 (x11)
     LUT6:I2->O             2   0.086   0.905  L3/LUT5SI/Mxor_x1_Result1 (L3/LUT5SI/x1)
     LUT6:I0->O             4   0.086   0.613  L3/LUT5SI/Mxor_x3_Result1 (L3/si)
     LUT4:I1->O             1   0.086   0.901  n089 (n0)
     LUT6:I0->O             4   0.086   0.500  n11 (n1)
     LUT4:I2->O             1   0.086   0.412  n279 (n2)
     LUT6:I5->O             4   0.086   0.914  n389 (n3)
     LUT6:I0->O             5   0.086   0.505  n4 (n4)
     LUT4:I2->O             3   0.086   0.496  n5 (n5)
     LUT4:I2->O             4   0.086   0.914  n6 (n6)
     LUT6:I0->O             5   0.086   0.837  L17/LUT5SI/o51 (y16)
     LUT6:I1->O             4   0.086   0.425  L18/LUT5SI/o51 (y17)
     LUT6:I5->O             2   0.086   0.666  L19/LUT5SI/o51 (y18)
     LUT6:I2->O             1   0.086   0.286  Mxor_product<15>_Result (product_15_OBUF)
     OBUF:I->O                  2.144          product_15_OBUF (product<15>)
    ----------------------------------------
    Total                      13.818ns (4.042ns logic, 9.776ns route)
                                        (29.3% logic, 70.7% route)
```

Figure 7. Timing Report of Signed Accurate Multiplier

Timing report generated by the synthesis and place-and-route tools used for implementation, and contain detailed information specific to its design and target device as shown in the figure 7. This includes gate delay and net delay where Gate delays refer to the propagation delays associated with the elements LUT6s, LUT5s, and MUXs) used in multiplier implementation. Each LUT6 or LUT5 has a certain propagation delay from its inputs to the output, which depends on the logic function being implemented and the internal structure of the LUT.

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

These gate delays are typically reported in the timing report, along with the specific logic elements and paths where they occur. Net delays refer to the delays associated with the interconnect routing between the logic elements (LUT6s, LUT5s, and MUXs) in multiplier implementation.

The interconnect delays depend on the physical routing of the signals in the FPGA or ASIC device, including factors such as wire length, fan out, and capacitive loading. The net delays can vary significantly that depends on the placement and routing of the multiplier components on the target device. In the timing report, net delays are typically reported for each net (interconnect) between logic elements, indicating the propagation delay for signals travelling through that specific net.

The Power Report of Signed Accurate Multiplier and Output Waveform of Signed Accurate Multiplier as shown in the figure 8 and figure 9.

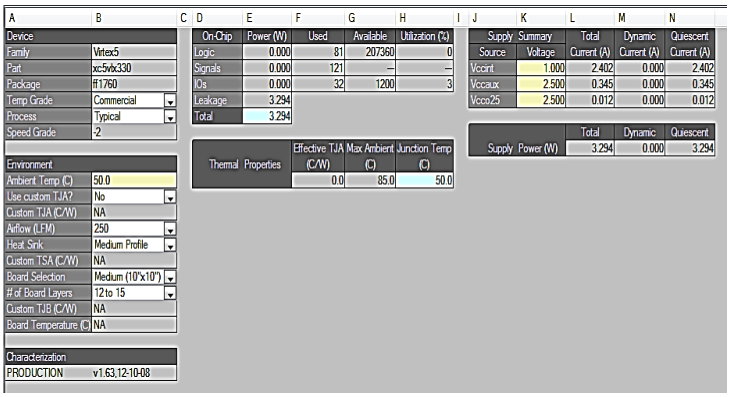

Figure 8. Power Report of Signed Accurate Multiplier

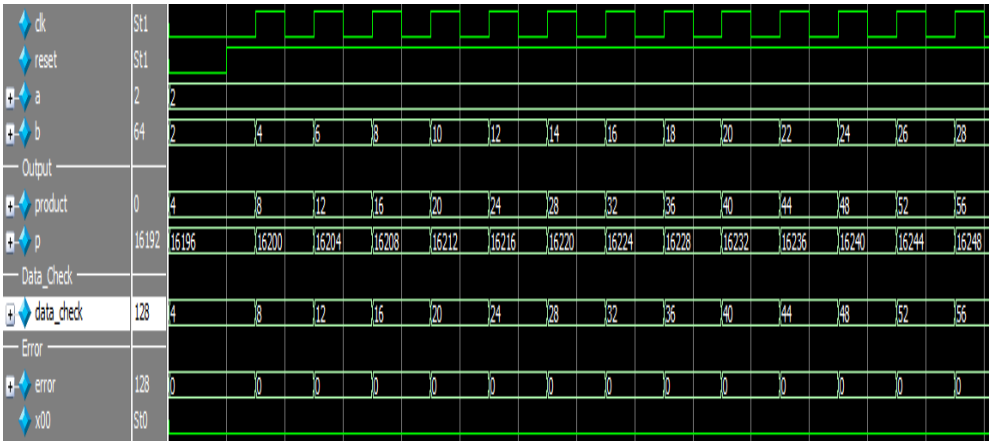

Figure 9. Output Waveform of Signed Accurate Multiplier

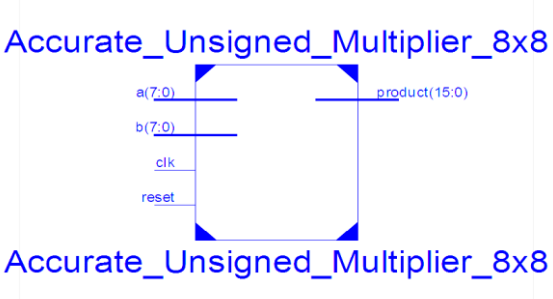

Figure 10. Signed Accurate Multiplier

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

================================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 11 / 2
--------------------------------------------------------------------------------
Delay:               4.110ns (Levels of Logic = 3)
  Source:            i5 (PAD)
  Destination:       o6 (PAD)

  Data Path: i5 to o6
                                   Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             1   0.694   0.901  i5_IBUF (i5_IBUF)
     LUT6:I0->O            1   0.086   0.286  o61 (o6_OBUF)
     OBUF:I->O                 2.144          o6_OBUF (o6)
    ----------------------------------------
    Total                      4.110ns (2.924ns logic, 1.186ns route)
                                       (71.1% logic, 28.9% route)

================================================================================
```

Figure 11. Timing Report of Unsigned Accurate Multiplier

The above figure 11 represents the Timing Report of Unsigned Accurate Multiplier. The Power Report of Unsigned Accurate Multiplier and output waveform are shown in the figure 12 and figure 13.

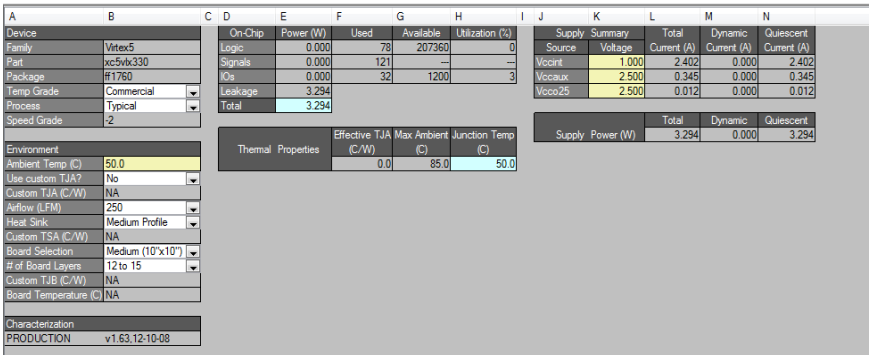

Figure 12. Power Report of Unsigned Accurate Multiplier

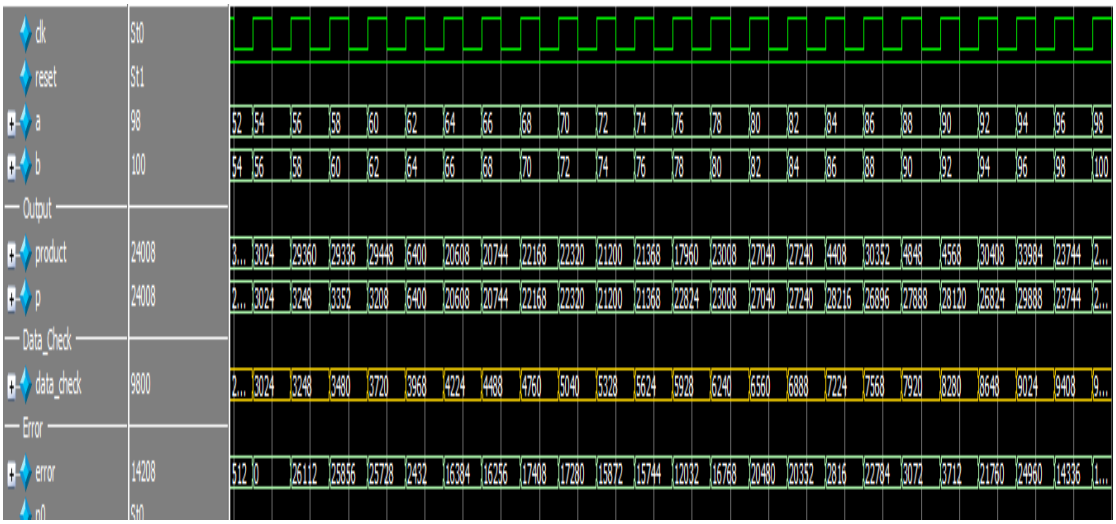

Figure 13. Output Waveform of Unsigned Accurate Multiplier

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

7. Conclusion and Future Scope

This work provides area-optimized, high performance soft-core exact and approximation multipliers that make use of the LUT6 tables and supporting sum and the carry chains seen in current FPGAs. It is recommended that LUT6 with multiplexers be used instead of numerous LUT5s to create accurate and approximation signed and unsigned multipliers for 8-bit configurations. Compared to Xilinx's area-efficient multiplier IP, we propose accurate signed and unsigned multipliers that may lower the number of LUTs utilized. As a result, performance is increased and critical path delays are at most decreased when the LUT6 architecture is executed using a single LUT5 with multiplexers rather than a dual LUT5 with multiplexers.

The future scope would be to increase the accuracy and further decrease in the number of crucial path delays. This would improve the power consumption and area which plays an important role in the field of designing the circuits and is promising and encompasses several areas of advancement and application like future research may focus on refining the placement and routing algorithms to minimize delays and maximize throughput. Future FPGA designs could explore techniques to reduce power consumption while maintaining or enhancing performance. This could explore techniques to reduce power consumption while maintaining or enhancing performance. Future advancements might include tools and methodologies to automatically generate LUT-based multiplier structures optimized for particular tasks or datasets, making FPGA implementation more accessible to a wider range of applications.

References

[1] A. Kakacak, A. E. Guzel, O. Cihangir, S. Gören, and H. F. Ugurdag, "Fast Multiplier Generator for FPGAs with LUT based Partial Product Generation and Column/Row Compression," *Integr. VLSI J.*, vol. 57, pp. 147–157, 2017.

[2] J. Beuchat and J. Muller, "Automatic Generation of Modular Multipliers for FPGA Applications," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1600–1613, Dec. 2008.

[3] N. Brunie, F. de Dinechin, M. Istoan, G. Sergent, K. Illyes, and B. Popa, "Arithmetic Core Generation Using Bit Heaps," in *Proc. 23rd Int. Conf. Field Program. Log. Appl. (FPL)*, Porto, Portugal, 2013, pp. 1–8.

[4] M. Kumm, "Optimal constant multiplication using integer linear programming," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 5, pp. 567–571, May 2018, doi: 10.1109/TCSII.2017.2739185.

[5] P. Paz and M. Garrido, "Efficient Implementation of Complex Multipliers on FPGAs Using DSP Slices," *J. Signal Process. Syst.*, vol. 95, pp. 543–550, Apr. 2023. [Online]. Available: https://doi.org/10.1007/s11265-023-01867-7

[6] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, "High Performance Accurate and Approximate Multipliers for FPGA-Based Hardware Accelerators," presented at the *2021 Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Feb. 2021.

[7] H. Parandeh-Afshar and P. Ienne, "Measuring and Reducing the Performance Gap between Embedded and Soft Multipliers on FPGAs," in *Proc. 21st Int. Conf. Field Program. Log. Appl. (FPL)*, Chania, Greece, 2011, pp. 225–231.

[8] R. Pilipovic and P. Bulic, "The Logarithmic Multiplier Design with Radix-4 Booth Encoding," presented at the *2020 Int. Conf. Comput. Design (ICCD)*, Hartford, CT, USA, Oct. 2020.

[9] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "FP-BNN: Binarized Neural Network on FPGA," presented at the *2018 Int. Conf. Field Program. Technol. (FPT)*, Naha, Japan, Dec. 2018.

[10] S. Mittal, "A Survey of FPGA-Based Accelerators for Convolutional Neural Networks," *Neural Comput. Appl.*, vol. 32, pp. 1109–1139, Jan. 2020.

[11] R. Pilipovic, P. Bulic, and W. E. G. Walters, "Array Multipliers for High Throughput in Xilinx FPGAs with 6-Input LUTs," *Computers*, vol. 5, no. 4, Dec. 2020.

[12] J. Faraone, M. Kumm, and H. W. Leong, "AddNet: Deep Neural Networks Using FPGA-Optimized Multipliers," presented at the *2019 Int. Conf. Field Program. Technol. (FPT)*, Tianjin, China, Dec. 2019.

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025

[13] B. Thiyaneswaran, S. Kumarganesh, M. Dharmalingam, P. N. Palanisamy, L. Vasanth, and A. Immanuvel, "Environmental Pollution and Weather Data Monitoring Using LoRa Low Power VLSI Solution," presented at the *9th Int. Conf. Sci. Technol. Eng. Math. (ICONSTEM)*, Chennai, India, Apr. 2024, doi: 10.1109/ICONSTEM60960.2024.10568664.

[14] S. P. Anthoniraj, A. K. Anguraj, S. Kumarganesh, and B. Thiyaneswaran, "Development of Keyless Biometric Authenticated Vehicle Ignition System," *Mater. Today: Proc.*, vol. 81, no. 2, pp. 464–469, 2021.

[15] D. Kalaiyarasi and M. Saraswathi, "Design of an Efficient High Speed Radix-4 Booth Multiplier for Both Signed and Unsigned Numbers," presented at the *2016 Int. Conf. Adv. Electron. Comput. Commun. (ICAECC)*, Bangalore, India, Oct. 2016.

DOI: https://doi.org/10.54216/IJWAC.090202
Received: January 17, 2025 Revised: February 13, 2025 Accepted: March 14, 2025