International Journal of Engineering Innovations in Advanced Technology
ISSN: 2582-1431 (Online), Volume-4 Issue-4, December 2022

Design High Speed Area Efficient Rounding Based Multiplier Fir DSP Applications

R Hari Kumar Reddy
MTech Student, Department Of ECE, VLSI & ES, GVR&S College of Engg. & Tech., India.
Email: ramapuramhari383@gmail.com

M.B. Narendra
Associate professor, Department Of ECE, VLSI & ES, GVR&S College of Engg. & Tech., India.
Email: mbn399@gmail.com

G Venkata Ramana Reddy
Associate professor, HOD, Department Of ECE, VLSI & ES, GVR&S College of Engg. & Tech., India.

***Abstract*— Compression of data is a common practice in the signal - processing & digital image analysis fields, and is often employed for multimedia & image processing purposes. Approximate computation is a prominent design approach in arithmetic. New high-speed areas may be opened up by high-speed multimedia applications. Error-tolerant circuits that use approximate computations. At the same time, these applications give great production at a reduced cost of accuracy In addition, the system's complexity is reduced as a result of their implementations. Power consumption and latency in the system's architecture are the main factors. It is proposed that two compressors be designed and analysed that have smaller size, less delay, and more power than the present systems, all while maintaining precision that is equivalent to the current systems.**

***Key Words*- image processing, Error-tolerant, multimedia**

Manuscript received Nov 10, 2022; Revised Dec 25, 2022; Accepted Dec 31, 2022

I. INTRODUCTION

Exact computing units aren't necessarily essential in applications like data mining and multimedia signal processing. They may be substituted with a similar item. Research into error-tolerant applications using approximation computation is on the increase. these applications rely heavily on adders and multipliers. In digital signal processing, approximate complete adders are suggested at the transistor level. Partial product accumulation in multipliers is handled by their suggested full-adder design

The use of truncation in fixed-width multiplication designs is common to simplify the circuitry. In order to compensate again for quantization error induced by the reduced portion, a variable correction component is added

Accumulation of bits is critical in terms pf power usage when using approximation methods in multipliers. If the least relevant bits of the inputs can be trimmed, then partial products may be formed in order to decrease hardware complexity. In partial product accumulations, the suggested multiplier saves just a few adder circuits. A partial product reduction trees of four 8 x 8Dadda multiplier variations is given and applied with two types of roughly 4-2 compressors.

Mean relative error (MRE) is greatly affected by the suggested compressors in that they produce nonzero output for zero-valued inputs, which is a severe negative. In this brief, an estimated design is presented to address the current issue. As a result, accuracy is improved. When a segment multiplier (SSM) is used, the leading 1 bit of each operand is used to generate an output of m-bit segments. Then, instead of n n multiplication, m m multiplication is used. A partial product perforation (PPP) multiplier in an n-bit multiplier omits subsequent partial products beginning at position j, where j [0,n-1] and k [1, min (n-j, n-1)]. Multiplying an element in the Karnaugh map by 2 is used as a building block to produce 4 4 and 8 8 multiplications. Power-efficient Wallace tree multipliers might benefit from an inaccurate counter design. The multiplier partial product accumulation is handled by a new approximation adder. Compared to an accurate multiplier, a 16-bit approximation multiplier achieves a 26% decrease in power.

Voltage over-scaling (VOS) is used to approximate an 8-bit Wallace tree multiplier. Errors may be caused by lowering the supply voltage, which produces routes that do not match delay restrictions. In the past, logic complexity reduction has been achieved by simply applying approximation adders and compressor to the partial products. Various probabilities are included into the partial products in this short. Systematic approximation is used to examine the likelihood statistics of the changing partial products.

Approximation is suggested using half-adder, full-adder, and 4-2 compressors. The complexity of the arithmetic units has been lowered, but the error value has also been taken into consideration. Systemic approximation improves accuracy, while lower logic architecture of approximation arithmetic units reduces power and area consumption. Image processing applications benefit from improved peak signal-to-noise ratio (PSNR) values achieved by the suggested multipliers over previous designs. The arithmetic separation between a proper output and an approximation output for just a given input may be characterised as the error distance (ED).

II. EXISTING METHOD

A redundant binary (RB) format may be utilised to create high-performance multipliers because of its flexibility and carry-free addition. The traditional RB multiplier adds an extra RB partial product (RBPP) row, since an error-correcting word (ECW) is produced both by the radix-4 Modified Booth encoding (MBE) or the RB encoding. For the MBE multiplier, this means an extra RBPP accumulation step. To save one RBPP accumulation step, a novel RB altered bits generator (RBMPPG) has been developed in this study. RBMPPG is less wasteful than RB MBE multiplier because it creates fewer incomplete product rows. When the length of each multiplier operand is at least 32 bits, simulation findings demonstrate that the proposed RBMPPG-based designs greatly lower area and power consumption; these reductions over earlier NB multiplier designs result in a minor latency gain (approximately 5 percent). The suggested RB multipliers may minimise the power-delay product by up to 59 percent when compared to current RB multipliers.

Exact computing units aren't usually required in applications that can tolerate mistake, including multimedia signals processing and data mining. They may be substituted with a similar item. Research into error-tolerant applications using approximation computation is on the increase. These applications rely heavily on adders and multipliers. In digital signal processing, approximate complete adders are suggested at the transistor level. Partial product accumulation in multipliers is handled by their suggested full-adder design The use of truncation in repaired multiplier designs is common to simplify the circuitry.

A. Modified Booth encoding

Two signed binary values in 2 ’s complement notation are multiplied using Booth's multiplication method. In 1950, while working on diffraction at Birkbeck University in Bloomsbury, London, Andrew Donald Booth came up with the algorithm. Algorithm for shifting was developed by Booth, who utilised desk computers that were more efficient in shifting than at adding. In the field of computer architecture, Booth's method is of interest.

An implicit bit underneath the most significant bit, y1 = 0, is examined using Booth's procedure for the N-bit multiplication Y in sign two's complement form. Bits yi and yi1 are examined for every bit yi, scale from zero to N - 1 for i. The product accumulation P remains unaffected if these two items are equivalent. The multiplicand times 2i is given to P when yi is zero or one, and the multiplyand times 2i is removed from P when yi is one or zero. The signed product is P's ultimate value.

However, the representations of a multiplicand & product are not defined and may be any number system that allows the addition and subtraction of numbers; as is the case with the multiplier. The sequence of the stages is left to the reader's imagination, as indicated before. This is usually done by shifting the P accumulator rightward in small stages, beginning at I = 0, and then working its way up from there, starting with the lowest N bits of P. Then, the multiplying by 2i is often substituted. In terms of these specifics, there are several modifications and optimizations available.

Strings of 1s inside the multiplier are typically characterised as being converted to high-order+1 and low-order-1 at the endpoints of the string using this process. In this case, there is no elevated +1 and the total consequence is that the appropriate value is interpreted as negative.

Step 1: Booth Encoder and Partial Product Generator stage (BEPPG stage**):** The partial product generator's efficiency is influenced by the booth encoder. The cost, performance, & energy usage of the RB summation tree or the multiplier everywhere are affected by the number of bits that may be avoided at this step. There are 16 CRBBE-4 slices used to regulate the multiplier in stage one. There is a five-fold increase in difficulty. Shifting and selecting the multiplicand bits results in 16 rows of RBPPG partial products.

Step 2: Redundant Binary Adder summing tree stage (RBA summing stage): A total of 128 bits are generated by the partial products. There are four Redundant Binary Adders (RBAs) that add these bits together: 1, 2, 3, and 4. Blocks of 128 bits each had been created using RBA.

Step 3: Redundant binary to NB conversion stage (RB-to- NB stage): The final aggregate result is converted to NB representation using an RB-NB converter. It is possible to do the conversion in groups of successive digits based on their arrival time due to the uneven delay pattern of the final Rbs result bits An RBA tree summation may be used to assess the carry production of the following set of digits because the summing results do not affect the carry generation. When two binary integers need to be multiplied, a binary multiplication is a digital electrical circuit, such as a computer. Binary adders are used to construct it.

A digital multiplier may be implemented using a number of computer arithmetic approaches. In most cases, a collection of partial products is computed, and then the partial products are added together. A base-2 (binary) numeral system has been adapted from the one taught to primary school pupils for lengthy multiplication on base-10 integers.

B. Disadvantages of existing system are given below

- More logic complexity
- More power and more delay

III. SOFTWARE AND HARDWARE COMPONENTS

A. VERILOG

HDL (hardware description language) Verilog is used to simulate electronic systems in the semiconductor & electronic design sector. At the registration level of abstraction, Verilog HDL is most widely used in digital logic chip design, verification, and implementation. Analog and made by mixing circuits are also tested using this technique.

In contrast to software programming languages, hardware description languages like as Verilog contain methods for specifying the transmission of time & signal dependencies (sensitivity). A blocked assignment (=) and a non-blocking assignment (=) are both available. As a result of the non-blocking assignment, designers may express a government update without the necessity for declaring and using temporary storage. Designers may construct short and succinct circuit descriptions using Verilog's language semantics, which incorporates these ideas. Because of its release in 1984, Verilog gave circuit designers who had previously used graphical schematic capturing software and custom software programmes to record and test electronic circuits a huge productivity boost in the field of electronic circuit design.

Because C was already extensively used in the production of engineering software, the Verilog creators sought to create a syntax that would be comparable to that of C. A simple preprocessor exists in Verilog, although it lacks the sophistication of ANSI C/C++. It has the same flowing verbs (if/else, for, while, case, etc.) and the same operator precedence as the other language's version. Differences in syntax include variable definition (Verilog needs bit-widths for net/reg types), procedural block demarcation (begin/end rather than curly braces), and a slew of other tidbits.

In a Verilog design, the modules are organised into a hierarchy. Through a set of stated input, output and bidirectional ports, all modules may interact with each other. In terms of code, a module may have any combination of net/variable declarations (such as "wire," "register," "integer," and so on), simultaneous and sequential phrase blocks, or instances of many other modules (sub-hierarchies). Within a begin/end block, consecutive statements are encapsulated and performed one after the other. Verilog, on the other hand, is a dataflow language since the blocks itself are performed simultaneously.

Both the signal values ("1, 0, floating, indeterminate") and the intensity of a signal are included in Verilog's definition of "wire" (strong, weak, etc.). Sharing signal lines may be abstractly modelled using this technique. Wire value is determined by the source drivers & their respective strengths when numerous drivers are present.

The Verilog language has a subset containing statements that may be synthesised. It is possible to physically implement RTL (register-transfer level) Verilog modules using synthesis software. Verilog's (abstract) source is algorithmically transformed into a net list (a logically similar description comprising solely of basic logic primitives, including AND, OR, NOT, flip-flops, and so on) that is accessible in a given FPGA or VLSI technology. A circuit fabrication design may be generated from further modifications of the netlist (such as a photo mask set for an ASIC or a bit stream file for an FPGA).

B. Introduction to Xilinx-ISE

One of Xilinx's software tools for creating HDL designs, the ISE (Integrated Software Environment) enables the developer to create (or "compile") their designs, do timing analysis (RTL diagrams), simulate the response of a design to various stimuli, and programme the target device.

This is the Xilinx® designing environment that enables you to move your design through design input to Xilinx devices programming using the ISE® Design Suite Designers of logic, embedded processors, or DSP systems may all benefit from the ISE Design Suite, which offers editions adapted to their individual requirements.

- Achieve Greater Designer Productivity

The goal of process design is to better use the FPGA's resources by integrating multiple domains. The Proteus Design Suite: System Edition, which permits the usage of accelerators that the processor may use, can provide considerable performance gains for applications that increasingly depend on Digital Signal Processing (DSP) operations. Indeed, one of the key advantages of Xilinx specific system architecture is the flexibility to segment the system and manage the tradeoffs between software and software. Algorithms developed for low-level HDL languages may not be optimal for all users.

- Attain Breakthrough Performance, Power and Cost Benefits

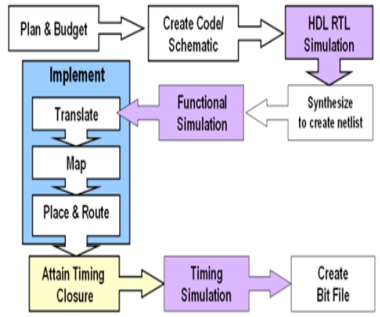
By unifying external functions inside an FPGA, you may reduce the cost of your system while creating a specialised processing platform. Your system should have the right mix of features and size, as well as the optimum hardware/software design trade-offs to satisfy your rigorous needs.

- Focus on Design Differentiation

With the ISE Create Suite System Edition, you get an integrated development environment, software tools, wizards, and intellectual property (IP) to help you design and make the most of a programmable platform's versatility. With the ISE Design Suite, you have access to Xilinx FPGA Intellectual Properties (IP) through the Xilinx CORE GeneratorTM System, which speeds up design time by giving access to highly parameterized IP. The user-customizable IP functions vary in complexity from simple functions like memory and FIFOs to more complicated system-level building blocks like filters and transformations. These IP blocks may save you anything from a few days to many months of development time. It is easier for FPGA designers to concentrate on generating ideas quickly and bringing products to market more quickly with the support of highly optimised IP.

C. Xilinx Design Flow Overview

The following steps are involved in the realization of a digital system using Xilinx FPGAs, as illustrated by the following figure.



- Design Entry

Our design is the initial stage. The creation of "Source" files is a simple method for accomplishing this goal. A schematic or a Hardware Description Language (HDL) such as VHDL or Verilog may be used to build source files. It is common for a project to include a top-level source file and a number of lower-level source files. A schematic or an HDL file may be generated from any of these files.

- Synthesis of Concepts

Synthesis is the process of assembling the multiple source files into a single netlist file. You may use these netlist files in the implementation module.

- Verification of the design (simulation)

At different points in the design process, this is an essential phase. Your circuit's functioning, behaviour, and timing may all be tested using a simulator (functional simulation). In order to determine the circuit's precise speed and timing, timing simulation must be done once the circuit has been implemented in the FPGA.

- Conceptualization and Execution

As soon as you have generated the netlist file (synthesis stage), implementation will transform the logic design to a physical format which can be installed on the target machine (e.g. Vertex FPGA).

Translating the netlist, Mapping, and Place&Route are all part of this process.

- Configuration of a device

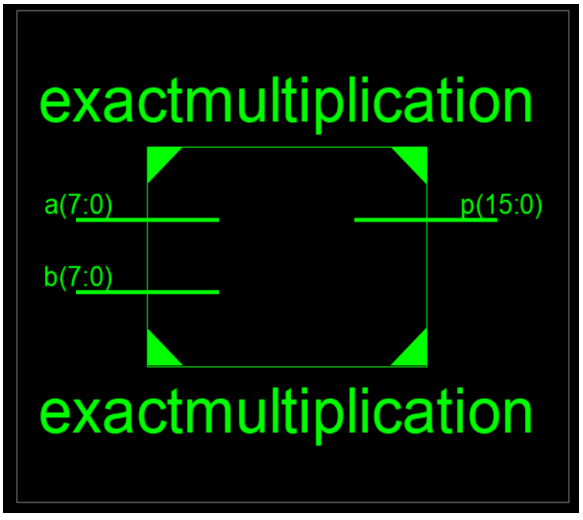
This refers to uploading the program file towards the Xilinx FPGA and programming the target FPGA.

- Simulation

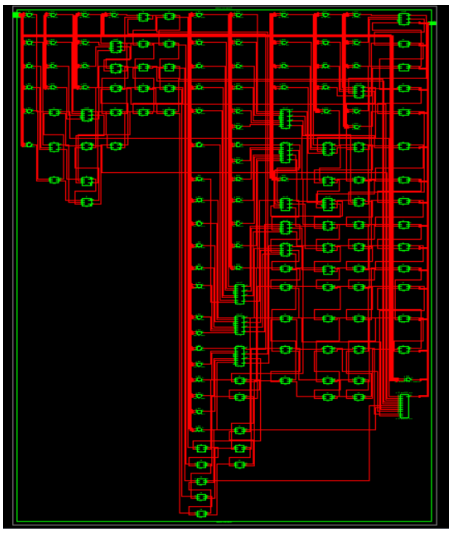
This is when the fun really begins.modelsim is indeed a verification device simulations tool for VHDL, Verilog, Verilog System Verilog and mixed-language designs. Simulating logic circuits has never been easier thanks to Modelsim, a robust simulator. Mentor Graphics' Modelsim is an intuitive but powerful VHDL/(System) Verilog/System C simulator. It allows for behavioural, register transfer, and gate-level modelling. A wide range of platforms are supported by Modelsim, including Linux, Solaris, Windows, and a variety of other operating systems.

IV. SIMULATION RESULTS

A. *RTL*



B. *Internal Block Diagram*



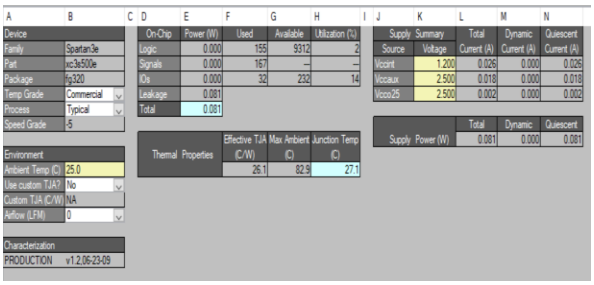
C. *Area*

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	155	9,312	1%	
Number of occupied Slices	87	4,656	1%	
Number of Slices containing only related logic	87	87	100%	
Number of Slices containing unrelated logic	0	87	0%	
Total Number of 4 input LUTs	155	9,312	1%	
Number of bonded IOBs	32	232	13%	
Average Fanout of Non-Clock Nets	3.28			

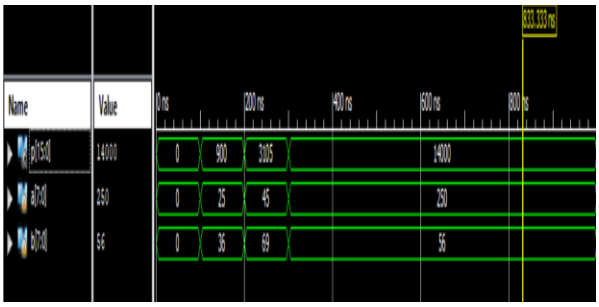
D. *Delay*

```
---------------------------------------
Total                     21.286ns (12.872ns logic, 8.414ns route)
                                   (60.5% logic, 39.5% route)

=========================================================================


Total REAL time to Xst completion: 6.00 secs
Total CPU time to Xst completion: 6.21 secs
```

E. Power



F. Simulation



V. CONCLUSION

Using signals generated and propagated, this paper proposes efficient approximation multipliers. A basic OR gate is used to create changed partial products for approximation. Half-adder, full-adder, and 4-2 compressors are all recommended to minimise the leftover partial products in the final product.. In Multiplier1, approximations are applied to all n bits, but in Multiplier2, they are applied just to the n 1 least significant bit. The space and power consumption of Multiplier1 and Multiplier2 are significantly reduced compared to exact designs. Both Multiplier1 and Multiplier2 save 87 and 58 percent, respectively, from accurate multipliers in APP compared to previous approximation solutions. When compared to previous approximation multiplier designs, they are proven to be more precise. Using the suggested multiplier designs, output quality may be maintained while substantial power and space are saved.

REFERENCES

[1] Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digitalsignal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137,Jan. 2013.

[2] J. King and E. E. Swartzlander, Jr., "Data-dependent truncationscheme for parallel multipliers," in Proc. 31st Asilomar Conf. Signals,Circuits Syst., Nov. 1998, pp. 1178–1182.

[3] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design low-error fixed-width modified booth multiplier," IEEE Trans. VeryLarge Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531,May 2004.

[4] R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspiredimprecise computational blocks for efficient VLSI implementation ofsoft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers,vol. 57, no. 4, pp. 850–862, Apr. 2010.

[5] Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design andanalysis of approximate compressors for multiplication," IEEE Trans.Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[6] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim,"Energy-efficient approximate multiplication for digital signal processingand classification applications," IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[7] Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi,"Design-efficient approximate multiplication circuits through partialproduct perforation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,vol. 24, no. 10, pp. 3105–3117, Oct. 2016.