JOURNAL OF BASIC SCIENCE AND ENGINEERING

DESIGN OF ROUNDING AREA AND POWER EFFICIENT BASED MULTIPLE FOR SIGNED AND UNSIGNED OPERATIONS

[1]Srinivasarao Udara, [2]Harish H M, [3] Niranjana Kumara M, [4] Annaiah H,

[1]Professor, Department of E & C, S T J Institute of Technology, Ranebennur, India
[2]Assistant Professor, Department of ECE, Government Engineering College Haveri, India
[3]Assistant Professor, Department of ECE, Government Engineering College Hassan, India
[4]Assistant Professor, Department of CSE, Government Engineering College Hassan, India

DOI: https://doie.org/10.0721/Jbse.2024635798

Article History: Received:14-04-2024 Revised: 24-06-2024 Accepted: 22-07-2024

ABSTRACT
The IEEE 754 Standard for Floating-Point Arithmetic has been the dominant method for representing real numbers in computer systems for many years. Recently, John L. Gustafson introduced the posit number format, which is touted to offer higher precision using the same or fewer bits and simpler hardware compared to IEEE 754. This new format is suggested as an alternative to the widely used IEEE 754 standard. This research investigates the posit number format, analyzing its features and properties as described in existing literature, and compares it to the standard floating-point numbers. This study evaluates the potential of posits to replace traditional floating-point numbers, as suggested in existing research. A posit arithmetic multiplier is designed at a low level using the Xilinx tool to generate synthesizable Verilog code, which is effective for unsigned number multiplication. Addressing both signed and unsigned numbers is crucial for practical applications. Therefore, we introduce a new method called the RoBA (Rounding Based Approximate) multiplier, which achieves significant reductions in area, delay, and power consumption, by 10%, 40%, and 54%, respectively. To conclude this study, we propose a low-level design for a posit arithmetic RoBA (Rounding Based Approximate) multiplier that supports both signed and unsigned operations. This design is created using the Xilinx tool to produce synthesizable HDL code. The Xilinx ISE 14.3 software is used for this purpose, while the Genus synthesis tool is employed for performing synthesis using 90nm technology, ensuring optimized area and low power consumption.

Keywords: Computer arithmetic, Floating point, Posit number system, Numerical error, Roba Multiplier.

1.1 Existing System

The IEEE-754 standard specifies a range of formats for representing, formatting, and computing with real numbers, providing flexibility from 16-bit to 256-bit precision. Different fields utilize various precisions to suit their needs; for instance, deep learning frequently employs single-precision (32-bit), while image processing can leverage lower precisions for efficiency. Additionally, applications like GROMACS may use a combination of different precisions to optimize performance [1-2].

Disadvantages:

- It occupies more Area
- It consumes more power

1.2 Proposed System

The proposed design for the posit multiplier involves a parameterized data path, which includes several key stages in its critical path: extracting posit components, performing mantissa multiplication with an (nb−es) bit radix-4 modified Booth multiplier, carrying out final addition and normalization, packing the posit components, and executing rounding. Moreover, the design applies distinct processing for the sign and exponent. To improve power efficiency, two key optimizations are implemented: generating a control signal for the mantissa multiplier to activate only the necessary segments, and dividing the mantissa multiplier into smaller sections, each managed by the corresponding control signal [3-5].

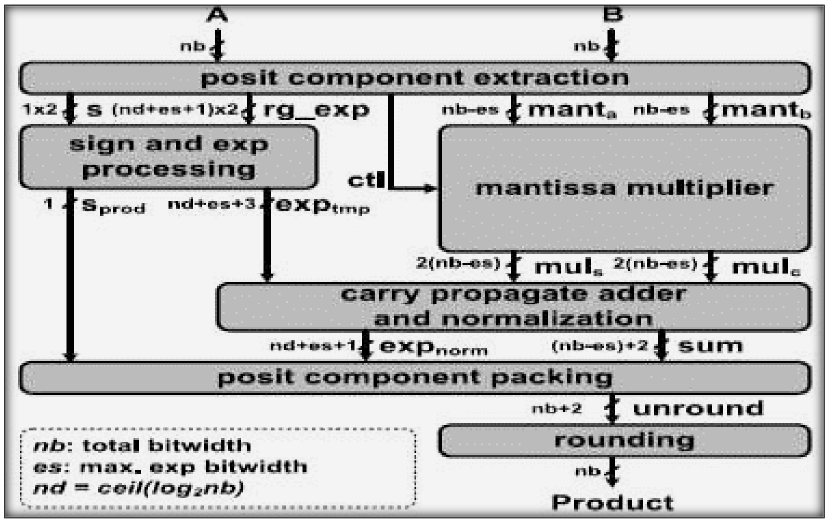


Figure 1.1: Flow for Proposed multiplier

Advantages:

- The multiplier structure was designed with Less area and it consumes less power

Applications:

- Deep Learning
- DSP
- MAC

To complete our multiplier design, generating a control signal is crucial. The 16-bit input is divided into four segments. By identifying the first occurrence of the binary digit '1', we derive the control signal for both inputs. This position informs the comparison between segmented groups and input digits, helping determine the appropriate control signal for each input. This ensures the optimal selection of a smaller multiplier for the operation. Overall, the control signal is vital for optimizing the segmented multiplier selection process.

2. Proposed Method

Beyond image and video processing, approximate computing proves advantageous in fields where perfect arithmetic precision is not critical. It allows designers to balance accuracy, speed, and power consumption effectively. This flexibility extends across different design layers, including circuitry, logic, architecture, algorithms, and software. Techniques for approximation may involve intentional timing violations such as voltage over-scaling or over clocking, function approximation methods like modifying Boolean functions in circuits, or a combination of these approaches. Function approximation techniques are applied to various arithmetic components, such as adders and multipliers, across different design layers. This research introduces a high-speed, low-power approximate multiplier specifically designed for robust digital signal processing (DSP) applications [6]. The proposed multiplier, which is also area-efficient, modifies traditional multiplication methods at the algorithmic level by assuming fixed input values. This design is known as the Rounding-Based Approximate (RoBA) multiplier.

The proposed multiplication technique accommodates both signed and unsigned operations and introduces three optimized architectures. These architectures are assessed based on delay, power consumption, energy-delay product (EDP), and area, providing comparisons with various approximate and exact multipliers. This study's key contribution is: 1) the introduction of a novel Rounding-Based Approximate (RoBA) multiplication scheme by modifying traditional methods, and 2) the development of three hardware architectures specifically designed for approximate multiplication in both signed and unsigned contexts [7-9].

The main concept behind the proposed approximate multiplier centers on leveraging the efficiency achieved by operating on numbers that are powers of two (2^n). This approach involves rounding the input values A and B to their nearest representations, A_round and B_round. The multiplication operation can then be reformulated using these rounded values to approximate the product of A and B. This method not only aims to reduce computational complexity but also optimizes power consumption and area utilization, aligning with the growing demand for efficient computational techniques in digital signal processing and other related applications.

$$A \times B = (A_r - A) \times (B_r - B) + A_r \times B + B_r \times A - A_r \times B_r .$$

The core insight here is that multiplication operations such as Ar × Br, Ar × B, and Br × A can be efficiently executed using shift operations, leveraging their computational efficiency in hardware implementations. However, the term (Ar − A) × (Br − B) introduces complexity in hardware due to its dependency on the differences between exact and rounded values, contributing minimally to the final result. To streamline the multiplication process and optimize hardware efficiency, it is proposed to omit this term from the computation. Thus, the multiplication operation can be effectively simplified using the following expression:

$$A \times B \cong A_r \times B + B_r \times A - A_r \times B_r$$

In this approach, we prioritize identifying the closest values of A and B as 2^n. When A (or B) equals 3 × 2^(p-2) (where p is any positive integer greater than one), it aligns with two nearby 2^n values, differing by 2^p and 2^(p-1) in absolute terms. Both values equally impact the accuracy of the proposed multiplier. Choosing the larger value (except when p = 2) simplifies hardware implementation for determining the nearest rounded value, which is why it is emphasized in this research.

This method capitalizes on numbers of the form 3 × 2^(p-2) being treated as "don't care" values in rounding operations, which simplifies the computational process. Employing these values for rounding up can result in smaller logical expressions. An exception arises with the number three, for which two is considered the nearest value in the proposed approximate multiplier [10-11].

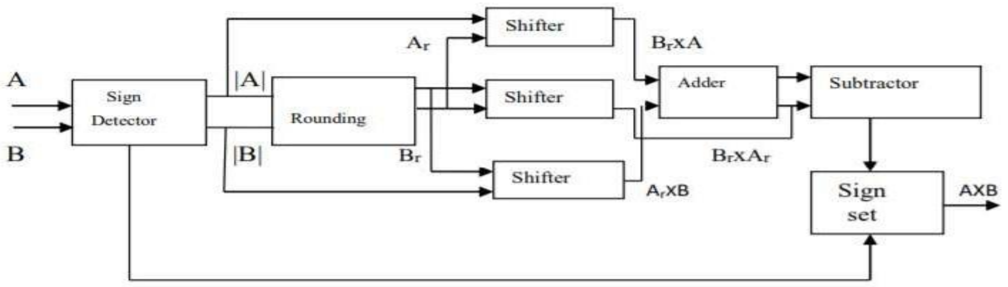
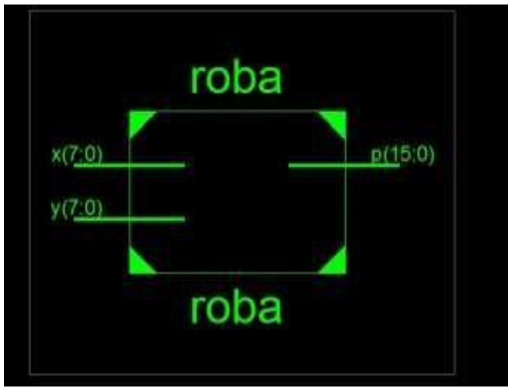


Figure 1.2: Block Diagram for the Hardware Implementation of the ROBA Multiplier

Unlike previous studies, where approximate results were consistently smaller than exact values, the RoBA multiplier can produce outcomes that vary in magnitude relative to the exact result. This variation hinges on the relative sizes of Ar and Br compared to A and B. Specifically, if one operand (like A) is smaller than its rounded value and the other operand (like B) is larger than its rounded value, the approximate result may exceed the exact result. This phenomenon occurs due to the negative product of (Ar − A) and (Br − B).

The RoBA multiplier yields an approximate result that surpasses the exact one due to the nature of the difference being the product itself. Conversely, when both operands A and B exceed or fall short of their rounded values Ar and Br, respectively, the approximate result will be less than the exact result. It's crucial to highlight that the benefits of the RoBA multiplier apply primarily to positive inputs, as the rounded values of negative inputs under two's complement representation do not align with the 2n format.

To ensure accurate multiplication in signed magnitude representation, it is advised to first ascertain the absolute values of both input operands and then determine the sign of the multiplication result based on the signs of these operands. This approach allows the multiplication operation to proceed using unsigned numbers, after which the appropriate sign is applied to the resultant unsigned value, ensuring correctness and consistency in the computation process.



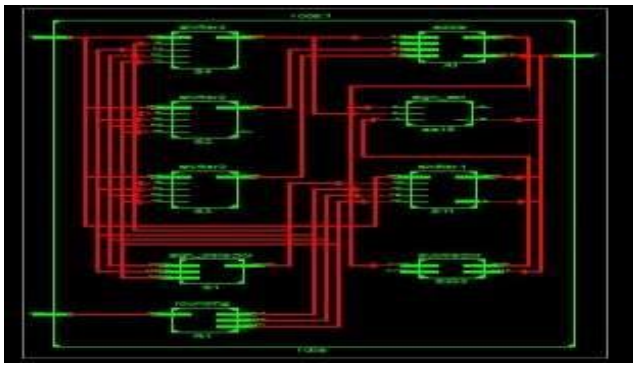
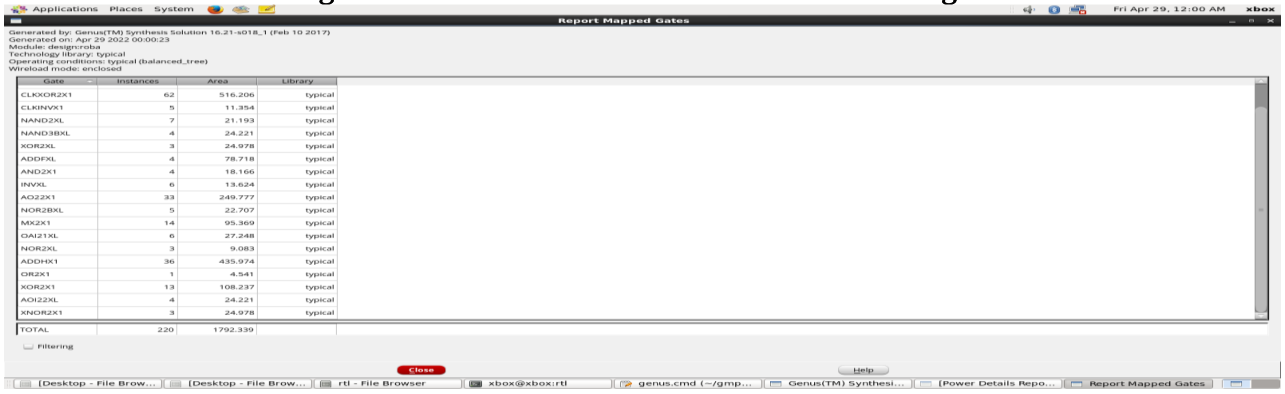


Figure 1.3: RTL schematic & internal block diagram


Applications Places System
Report Mapped Gates
Fri Apr 29, 12:00 AM
xbox
Filtering
Close
Help
[Desktop - File Brow... [Desktop - File Brow... rtl - File Browser xbox@xbox:rtl genus.cmd (~/gmp... Genus(TM) Synthesi... [Power Details Repo... Report Mapped Gates

```
Generated by: Genus(TM) Synthesis Solution 16.21-s018_1 (Feb 10 2017)
Generated on: Apr 29 2022 00:00:23
Module: design:roba
Technology library: typical
Operating conditions: typical (balanced_tree)
Wireload mode: enclosed
```

Gate	Instances	Area	Library
CLKXOR2X1	62	516.206	typical
CLKINVX1	5	11.354	typical
NAND2XL	7	21.193	typical
NAND3BXL	4	24.221	typical
XOR2XL	3	24.978	typical
ADDFXL	4	78.718	typical
AND2X1	4	18.166	typical
INVXL	6	13.624	typical
AO22X1	33	249.777	typical
NOR2BXL	5	22.707	typical
MX2X1	14	95.369	typical
OAI21XL	6	27.248	typical
NOR2XL	3	9.083	typical
ADDHX1	36	435.974	typical
OR2X1	1	4.541	typical
XOR2X1	13	108.237	typical
AOI22XL	4	24.221	typical
XNOR2X1	3	24.978	typical
TOTAL	220	1792.339	

Figure 1.4: Area report of the Roba multiplier

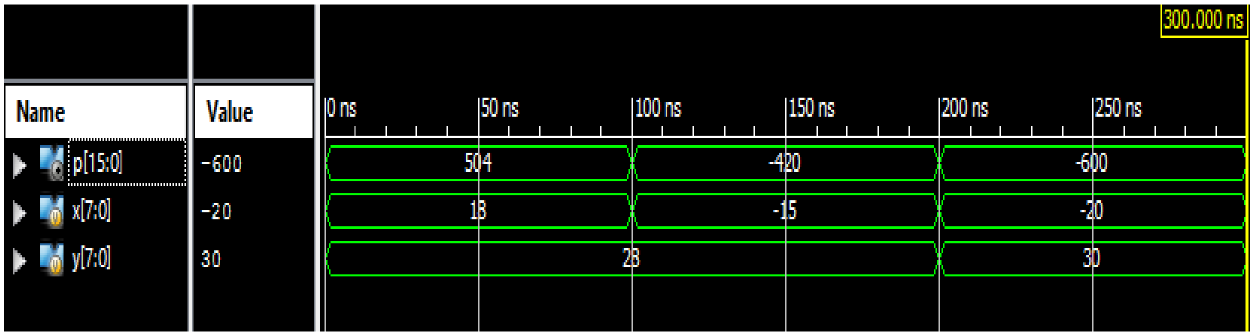
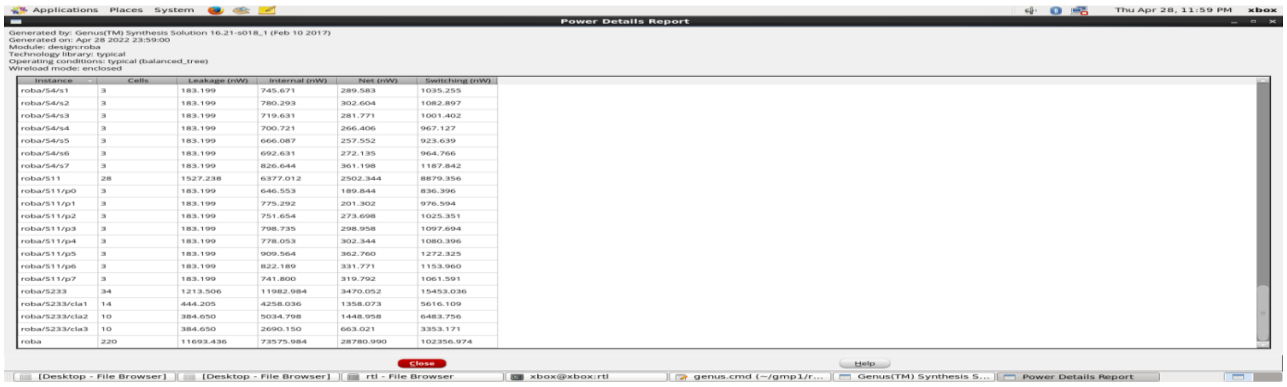


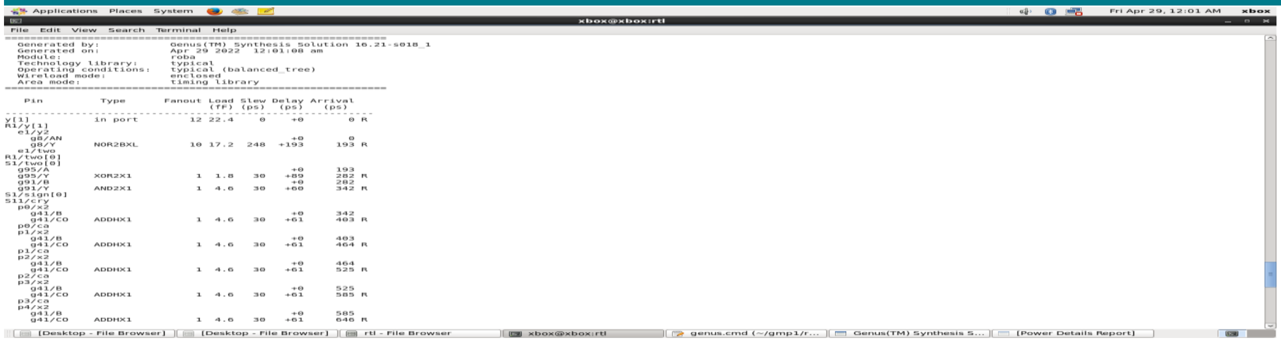
Figure 1.5: Simulation of the Roba multiplier


Applications Places System
Power Details Report
Thu Apr 28, 11:59 PM
xbox
Close
Help
[Desktop - File Browser] [Desktop - File Browser] rtl - File Browser xbox@xbox:rtl genus.cmd (~/gmp1/r... Genus(TM) Synthesis S... Power Details Report

```
Generated by: Genus(TM) Synthesis Solution 16.21-s018_1 (Feb 10 2017)
Generated on: Apr 28 2022 23:59:00
Module: design:roba
Technology library: typical
Operating conditions: typical (balanced_tree)
Wireload mode: enclosed
```

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
roba/S4/s1	3	183.199	745.671	289.583	1035.255
roba/S4/s2	3	183.199	780.293	302.604	1082.897
roba/S4/s3	3	183.199	719.631	281.771	1001.402
roba/S4/s4	3	183.199	700.721	266.406	967.127
roba/S4/s5	3	183.199	666.087	257.552	923.639
roba/S4/s6	3	183.199	692.631	272.135	964.766
roba/S4/s7	3	183.199	826.644	361.198	1187.842
roba/S11	28	1527.238	6377.012	2502.344	8879.356
roba/S11/p0	3	183.199	646.553	189.844	836.396
roba/S11/p1	3	183.199	775.292	201.302	976.594
roba/S11/p2	3	183.199	751.654	273.698	1025.351
roba/S11/p3	3	183.199	798.735	298.958	1097.694
roba/S11/p4	3	183.199	778.053	302.344	1080.396
roba/S11/p5	3	183.199	909.564	362.760	1272.325
roba/S11/p6	3	183.199	822.189	331.771	1153.960
roba/S11/p7	3	183.199	741.800	319.792	1061.591
roba/S233	34	1213.506	11982.984	3470.052	15453.036
roba/S233/cla1	14	444.205	4258.036	1358.073	5616.109
roba/S233/cla2	10	384.650	5034.798	1448.958	6483.756
roba/S233/cla3	10	384.650	2690.150	663.021	3353.171
roba	220	11693.436	73575.984	28780.990	102356.974

Figure 1.6: power report of the Roba multiplier



```
  Generated by:            Genus(TM) Synthesis Solution 16.21-s018_1
  Generated on:            Apr 29 2022  12:01:08 am
  Module:                  roba
  Technology library:      typical
  Operating conditions:    typical (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==============================================================

   Pin          Type        Fanout Load Slew Delay Arrival
                                   (fF) (ps)  (ps)   (ps)
------------------------------------------------------------
y[1]           in port          12 22.4    0    +0       0 R
R1/y[1]
  e1/y2
    g8/AN                                     +0       0
    g8/Y       NOR2BXL          10 17.2  248  +193     193 R
  e1/two
R1/two[0]
S1/two[0]
  g95/A                                       +0     193
  g95/Y        XOR2X1            1  1.8   30   +89     282 R
  g91/B                                       +0     282
  g91/Y        AND2X1            1  4.6   30   +60     342 R
S1/sign[0]
S11/cry
  p0/x2
    g41/B                                     +0     342
    g41/CO     ADDHX1            1  4.6   30   +61     403 R
  p0/ca
  p1/x2
    g41/B                                     +0     403
    g41/CO     ADDHX1            1  4.6   30   +61     464 R
  p1/ca
  p2/x2
    g41/B                                     +0     464
    g41/CO     ADDHX1            1  4.6   30   +61     525 R
  p2/ca
  p3/x2
    g41/B                                     +0     525
    g41/CO     ADDHX1            1  4.6   30   +61     585 R
  p3/ca
  p4/x2
    g41/B                                     +0     585
    g41/CO     ADDHX1            1  4.6   30   +61     646 R
```

Figure 1.7: Timing report of the Roba multiplier

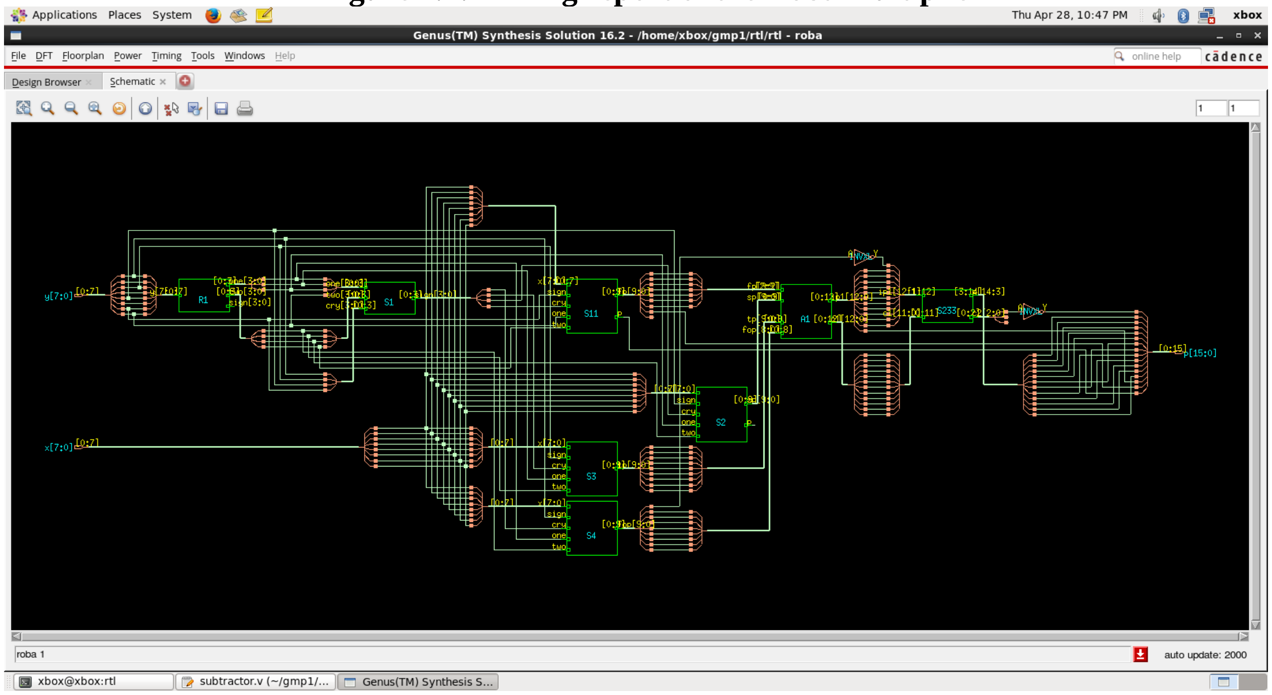

Genus(TM) Synthesis Solution 16.2 - /home/xbox/gmp1/rtl/rtl - roba
File DFT Floorplan Power Timing Tools Windows Help
Design Browser
Schematic
y[7:0]
x[7:0]
R1
S1
S11
S2
S3
S4
p[15:0]
roba 1
auto update: 2000

Figure 1.8: Synthesis of the Roba multiplier

3. Results and discussions

To ensure equitable comparison, uniform Verilog HDL coding practices are employed using the Xilinx 14.7 ISE tool to design each logic block. Subsequently, all designs undergo synthesis with the Genus tool utilizing the SAED 90nm CMOS technology library. This process evaluates key metrics such as core area, timing characteristics, and power consumption across different word sizes. The analysis includes critical parameters such as maximum combinational gate delay, core area utilization, power consumption metrics like area-delay-product (ADP), and power-delay-product (PDP), which are detailed in Table 1. The observed outcomes are subject to variation based on the HDL coding methodology and the optimization configurations applied in the Genus tool. Utilizing a 16-bit input sequence, the implementation of the High-Speed and Low-Power RoBA multiplier illustrates notable advantages including reduced power consumption, minimized footprint, and enhanced operational speed. Detailed analysis through area-delay-product (ADP) and power-delay-product (PDP) plots for the 16-bit input sequence are presented in Figures 1.6 and 1.7, respectively. Additionally, Figure 1.8 provides the RTL perspective of the 32-bit HC binary adder.

Existing method using Xilinx 4.7 tool

Name of the module	Power (W)	Area (µ)	Timing
Robo multiplier	3.5678	189	12.567 ns

Current method using Genus tool

Name of the module	Power (nW)	Area (nm)	Timing (fs)
Robo multiplier	120.5678	1782	387.567

4. Conclusion

The RoBA multiplier is an approximate multiplier designed to prioritize high speed and energy efficiency in digital signal processing. It operates by rounding input values to the nearest power of 2, significantly simplifying the multiplication process. This method reduces computational complexity, leading to enhanced speed and reduced energy consumption, although it introduces minor approximation errors. Notably, the RoBA multiplier supports both signed and unsigned multiplication operations, making it versatile for various applications in computational systems.

References

1) Hao Zhang and Seok-Bum Ko, ―Design of Power Efficient Posit Multiplier, ‖IEEE on Circuits and Systems—ii: express briefs, vol. 67, no.5, May2020.Transactions

2) J.L. Gustafson and I. Yonemoto, ―Beating floating point at its own game: Positarithmetic, ‖Supercomput.Front.Innovat.Int. J., vol. 4,no. 2, pp. 71– 86, Jun. 2017.

3) Sneha Singh and Prachi Singh, ―Implementation of Radix-4 Booth Multiplier by VHDL,‖ IJRREEE, vol 4, issue 1,pp 1-11, Jan 2017.

4) Sukhmeet Kaur, Suman and Manpreet Singh Manna, ―Implementation of Modified Booth Algorithm(radix-4) and its comparison with Booth Algorithm(radix-2),‖ Advances Electric and Electronic Engineering, ISSN 2231-1297, vol 3, no 6,pp683-690, 2013.

5) Bikash Chandra Sahoo, Sanjay Kumar Samant, ―Design and Power Estimation of Booth Multiplier Using Different Adder Architectures, ‖2013.

6) S.ShafiullaBasha,Syed,JahangirBadashah,―DesignandImplementationofRadix-4Based High Speed Multiplier for ALU‘s using Minimal Partial Products, ‖IJAET,ISSN 2231-1963,vol4, issue1, pp 314-325.

7) D.R. Kelly, B.J. Phillips, and S. Al-Sahrawi,―Approximate signed binary integer multipliers for arithmetic data value speculation,‖ in Proc. Conf. Design Archit. Signal Image Process.,2009, pp.97–104.

8) K.Y. Kyaw, W. L.Goh, and K.S.Yeo,―Low-power high-speed multiplier for error-tolerant

application,‖inProc.IEEEInt.Conf.ElectronDevicesSolid-StateCircuits(EDSSC),Dec.2010, pp.1–4.

9) A. Momeni,J.Han,P.Montuschi,andF.Lombardi, ―Design and analysis of approximate compressors formultiplication,‖IEEETrans.Comput.,vol.64,no.4,pp. 984–994,Apr.2015.

10) K. Bhardwaj and P.S. Mane, ―ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip, ‖ in Proc. 8th Int. Workshop Reconfigurable Commun. –Centric System. -Chip,2013, pp. 1–6.

11) K. Bhardwaj, P.S. Mane, and J. Henkel, ―Power-and area-efficient approximate wallace tree multiplier for error-resilient systems, ‖ in Proc. 15th Int. Symp. Quality Electron. Design(ISQED),2014, pp. 263–269.