International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

Modified Dadda Multipliers and Compressors Designed Using Approximate Multiplier Algorithm

Mtech Scholar Sidhharth Yadav, HOD & Professor Dr Bharti Chourasia
SRK University Bhopal(M.P)

Abstract- The multiplier is a crucial component in digital signal processing. Many scientists have attempted—and continue to attempt—to construct multipliers that satisfy the two flowing pan criteria of fast speed, low power consumption, consistent design, and fewer zones. This is made possible by technological advancements. They are appropriate for a range of applications needing high speed, low power, and less VSI consumption because they can even combine these two objectives into a single multiplier. In This paper present modified dadda multipliers using approximate multiplier with high speed and energy economy is discussed. This way, speed and energy efficiency are increased at the cost of a slight inaccuracy, as the computationally intensive part of the multiplication is bypassed. Whereas Isim Simulator is used for simulation, Xilinx 14.7 is used for implementation. Data from test bench validation indicates that it provides a higher accuracy than the others. Based on the simulation results, the suggested multiplier design outperforms earlier designs in terms of space, latency, speed, and power. Unlike prior proposals which could only construct 16 or 32 bit multipliers, the proposed multipliers can be constructed with 64 bits.

Index Terms- Dadda multiplier, approximate multiplier, Xilinx, vlsi.

I. INTRODUCTION

Today's digital signal processing and many other applications heavily rely on multipliers. As technology has advanced, numerous researchers have attempted and are now attempting to design multipliers. These types of multipliers can provide fast speed, low power consumption, and layout regularity—all of which are desirable design goals. They are therefore appropriate for a variety of high speed, low power and compact VSI implementations because they require less space, or even a combination of them in a single multiplier. IC designers may now design with less area thanks to VLSI. Typically, a single PCBA holding a CPU, RAM, ROM, and extra peripherals makes up a digital circuit. Nevertheless, very large-scale integration (VLSI) technology allows an integrated circuit designer to combine several of them into a single chip.
Various arithmetic operations, such as subtraction and duplication, are important components of sophisticated circuits that speed up processor calculations. In any case, the processor's multiplier unit has a major influence on its speed. This increases the interest in improved sign processors and AU fast multiplier design. Over the past few decades, a few innovative multiplier engineering techniques have been introduced. These days, speed is a major concern when designing processors, thus new engineering that is more rapidly than the multiplier previously mentioned must be given. This work aims to introduce a 64-bit modified dadda multiplier using approximate multiplier with low latency and high speed for advanced digital signal processing.

II. MULTIPLICATION ALGORITHM

Researchers have created a wide variety of multipliers. Here, the fundamental multiplication algorithm is covered. The following illustrates the multiplication procedure for an N-bit multiplicand by an N-bit multiplier:

Y= Yn-1 Yn-2 .....Y2 Y1 Y0 Multiplicand

X= Xn-1 Xn-2 ..... X2 X1 X0 Multiplier

Additionally, the Fractional Items, PP, are created using gates. In the unlikely event that the multiplicand is N bits and the multiplier is M bits, there will be a midway item of N * M. Different multipliers have different modes of operation that determine how fractional items are generated or tallied up. It is possible to break down duplicate paired numbers into increments. The 16-bit item P can be created by increasing the two 8-bit values, An and B.

A7 A6 A5 A4 A3 A2 A1 A0

X B7 B6 B5 B4 B3 B2 B1 B0

© 2024 IJSRET

International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

A7.B0 A6.B0 A5.B0 A4.B0 A3.B0 A2.B0 A1.B0 A0.B0
+ A7.B1 A6.B1 A5.B1 A4.B1 A3.B1 A2.B1 A1.B1 A0.B1
+ A7.B2 A6.B2 A5.B2 A4.B2 A3.B2 A2.B2 A1.B2 A0.B2
+ A7.B3 A6.B3 A5.B3 A4.B3 A3.B3 A2.B3 A1.B3 A0.B3
+ A7.B4 A6.B4 A5.B4 A4.B4 A3.B4 A2.B4 A1.B4 A0.B4
+ A7.B5 A6.B5 A5.B5 A4.B5 A3.B5 A2.B5 A1.B5 A0.B5
+ A7.B6 A6.B6 A5.B6 A4.B6 A3.B6 A2.B6 A1.B6 A0.B6
+A7.B7 A6.B7 A5.B7 A4.B7 A3.B7 A2.B7 A1.B7 A0.B7

P15 P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 P0

The equation for the addition is:

$$P(m+n) = A(m)B(n) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_i b_j 2^{i+j}$$

.

- Incorporate the multiplicand into a collector if the multiplier's SB is '1'.
- Move multiplicand one piece to one side and the multiplier one piece to the right.
- When the multiplier's bits are zero, stop.

The expansion of numbers has replaced the augmentation from above. When incomplete items are included one after the other, east equipment is used in conjunction with a sequential adder. Using a parade multiplier, it is possible to incorporate midway objects with a single combinational circuit.

Flow Chart of Proposed Method: In this section flow chart of proposed methods are discussed.

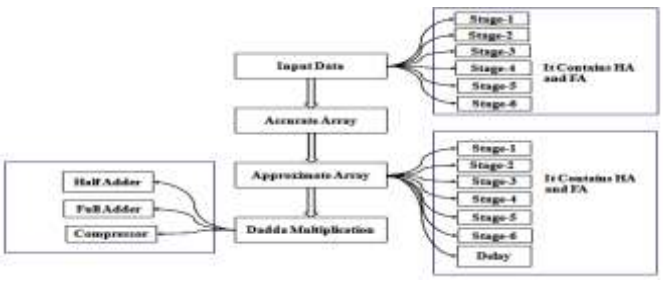

Figure 1: Flow Chart of proposed method

The suggested flow chart is displayed in Figure 1. It is evident from this working flow that the suggested approximate multiplier is designed and executed in accordance with the subsequent sub modulesCarry Save Adder.,Dadda Multiplier,Compressor,Full Adder and Half Adder. The number of adder stages needed to complete the midway item summation is limited by the dadda pot basecoat. To do this, the number of Ines in the grid of bits at each summation arrange is decreased by using full and half adders. Modified Dadda multipliers are an improved version of Wallace's parallel multipliers. The modified dadda multiplier is divided into three stages. N2 AND phases create the unfinished item grid during the primary stage. This reduces the halfway item lattice to a tan less of two in the next step. Dadda replaced the Wallace Pseudo adders with (n, m) counters as parameters. The circuit known as a Parae (n, m) counter comprises n inputs and generates m outputs, each of which represents a double of the ONEs that are present at the inputs. A fu adder is a function that accepts three inputs and produces two outputs by using a (3, 2) counter. Similarly, a half adder accepts two inputs and produces two by executing a (2, 2) counter.

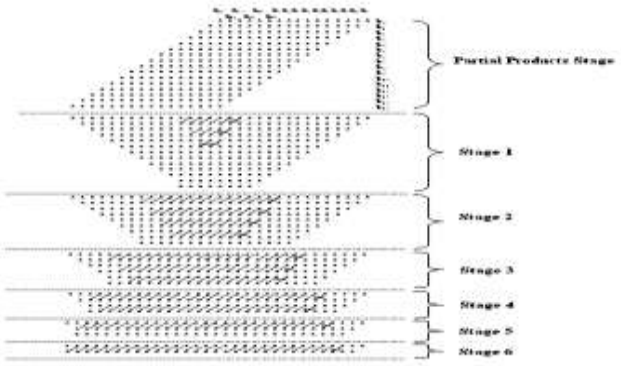

Figure 2: Dot diagram of proposed 64-bit modified dadda multiplier

III. SOFTWARE DESCRIPTION

Xilinx ISE[1] (Integrated Software Condition) is a software tool designed by Xilinx for the combination and analysis of HD pans. It allows the engineer to manipulate (or "arrange") their structures, conduct timing analysis, check RT outlines, simulate a structure's reaction to different boosts, and collaborate with the developer to design the desired device.

With the ISE Pan Suite: Rationale Version, you can move from device programming from the direction line or within the tied together condition of the ISE Venture Pilot, all the way from structure passage to usage and check.

© 2024 IJSRET

International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

IV. IMPLEMENTATION AND RT VIEW

In this section implementation and rt view are described

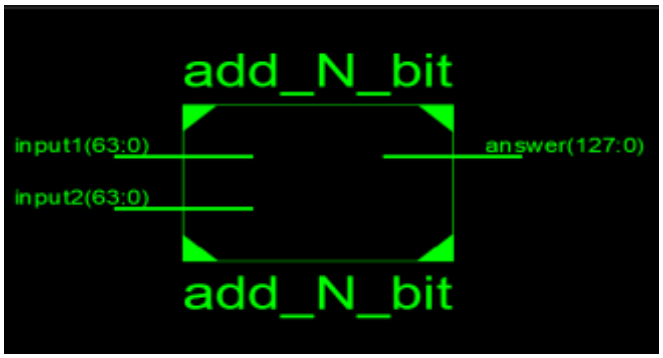

Figure 3: Top eve View

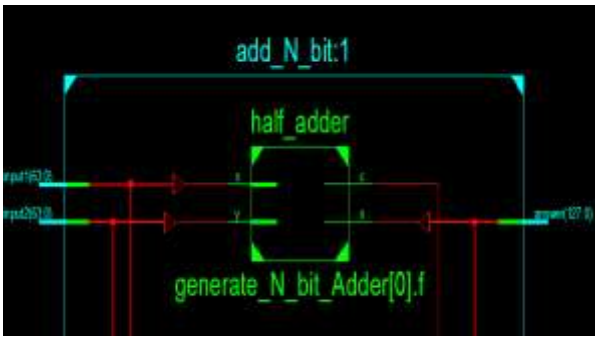

Figure 4: Half adder RT View

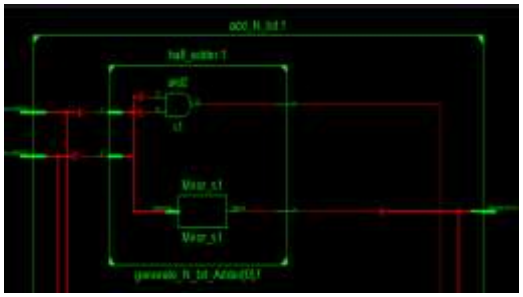

Figure 5: Internal architecture of half adder

The suggested 64-bit approximation multiplier is shown in top perspective view in figure 3. Inputs "a" and "b" each have 64 bits, respectively. This multiplier will provide a 128-bit output, or "c." The output bit of a digital multiplier is the sum of the input bits added.

Figure 4 presents the proposed multiplier from the half adder register transfer eve perspective. X and Y are the input bits and C and S are the output bits in a half adder, where c is the carry bit and s is the sum bit. The internal architecture of the half adder is depicted in Figure 5. This is where the XOR gate RT see is defined.

V. SIMULATION IN XILINX TEST BENCH

The proposed 64-bit approximation multiplier is implemented in Verilog using Xilinx 14.7 version. test bench results are simulated and validated using the Isim simulator. the binary numbers 1 and 0 represent the active signal and bank space mean, respectively, for the input values of 'a' and input values of 'b', here binary number 1 shows the active signal and 0 shows the bank space means.

1. Binary Number Input-1 [64-bit]

Input (a) =
1010111111000011010101010111110000111110000111100001111111111110
Input (b) =
1101011111111100000011111000001111111000000111111000000111111100

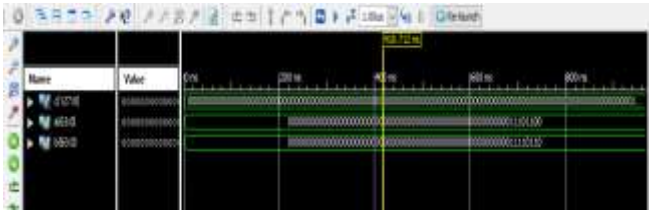

Figure 6: Result validation in Test Bench-1

Isim simulator is currently used for simulation. Figure 6 displays the window of the test bench. Here, the 64-bit inputs are "a" and "b," while the 128-bit output is "c." Values for "a" and "b" were previously mentioned. Now, "c" multiplies "a" and "b" to produce its output.

Output (c) =
0111000101100100001001010001001010000110111011101010001010001100111100100000000000000000000000000000000000000000000000000000000000000000000000000000

2. Binary Number Input-2 [32-bit]

Input (a) = 10111100001111000011111111111110
Input (b) = 11111000000111111000000111111100

Figure 7: Result validation in Test Bench-2

Here, the inputs "a" and "b" are 32 bits, and the output "c" is 64 bits. Values for "a" and "b" were previously mentioned. Now, "c" multiplies "a" and "b" to produce its output.

Output (c) =
0101101100111000110001000110111010111011101010000000000000000

© 2024 IJSRET

International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

3. Hexadecimal Number Input-1

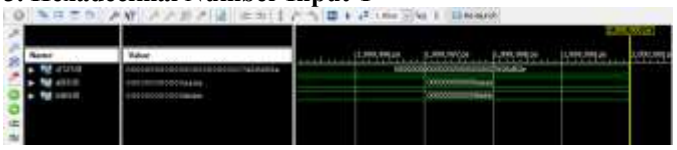

Figure 8 Result validation in Test Bench-3

In this case, the 64-bit hexadecimal inputs are "a" and "b," while the 128-bit hexadecimal output is "c." The values of "a" and "b" are aaaa and bbbb, respectively. "A" and "B" are multiplied to get "c" as the output. Therefore, 7D26D82E is the value of "c".

4. Hexadecimal Number Input-2

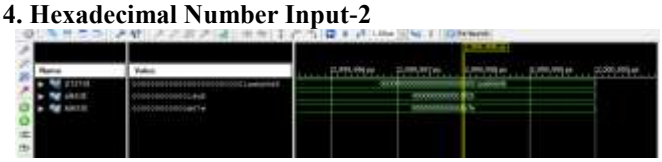

Figure 9: Result validation in Test Bench-4

Here, hexadecimal inputs "a" and "b" are 64 bits, and outputs "c" is 128 bits. 1fd5 is the value of "a," while d67e is the value of "b." Multiplying "a" with "b" is the result of "c." Thus, 1AABB8D6 is the value of 'c'.

5. Octal Number Input-1

Input (a) = 734546465477474446544
Input (b) = 646714646171646746161

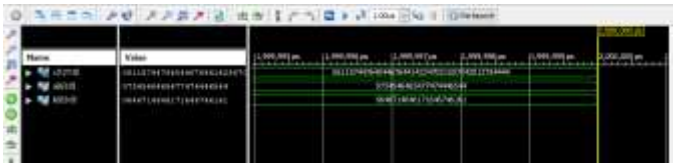

Figure 10 Result validations in Test Bench-5

Here, octal inputs "a" and "b" are 64 bits, and outputs "c" is 128 bits. The values of "a" and "b" were mentioned previously. "C" produces a multiplied version of "a" and "b."
Output (c) = 611574476404467644136156567640373062455O

6. Decima Number Input-1

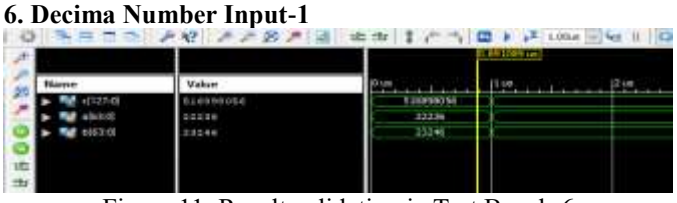

Figure 11: Result validation in Test Bench-6

Input (a) = 22236
Input (b) = 23246
Afterwards, 516898056 is the outcome of the multiplication operation.

VI. RESULT PARAMETER CALCULATION

In this section result of proposed methods are calculated.

1. Area

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	95	63400	0%
Number of fully used LUT-FF pairs	0	95	0%
Number of bonded IOBs	192	210	91%

Figure 12 Device utilization summary

A summary of the components utilizing the duration recommended approximate multiplier implementation is shown in Figure 512. There have been 95 lookups since the table was utilized, out of 63400 total available. Flip-flop pairings used 0 while availability is 95 according to the Look Up Table. 192 bonded input/output blocks were used, but there were 210 available. From this utilization summary, the total area is now calculated. Therefore, the proposed 64 bit approximate multiplier is implemented using 30.33% of the available area.

2. Power

Figure 13 X-power Anayzer

In Figure 13, the Xilinx power analyzer is displayed. There are several FPGA IC families available, including Spartan, Vertex, Kintex, and Artix. Both input and output power is measured in milliwatts. As a result, the total average power consumed is roughly 0.082 mW.

Table 1: Primitive and Back Box Usage summary

Sr No.	Bock Name	Quantity
1	BES	95
2	UT2	1
3	UT3	31
4	UT4	1
5	UT5	60
6	UT6	2
7	IO Buffers	192
8	IBUF	128
9	OBUF	64

© 2024 IJSRET

3. Timing Details

A values displayed in nanoseconds (ns)
Timing constraint: Default path analysis
Total number of paths / destination ports: 4160 / 64
Delay: 13.745ns (eves of logic = 34)
Source: input1<2> (PAD)
Destination: answer<62> (PAD)
Data Path: input1<2> to answer<62>

Gate Ce:in->out	Net fanout	Delay	Delay	logical Name (Net Name)
IBUF:I->O	2	0.001	0.698	input1_2_IBUF (input1_2_IBUF)
UT6:I0->O	3	0.097	0.305	generate_N_bit_Adder[2].f/c_out1 (carry<2>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[4].f/c_out1 (carry<4>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[6].f/c_out1 (carry<6>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[8].f/c_out1 (carry<8>)
UT3:I2->O	2	0.097	0.300	generate_N_bit_Adder[9].f/c_out1 (carry<9>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[11].f/c_out1 (carry<11>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[13].f/c_out1 (carry<13>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[15].f/c_out1 (carry<15>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[17].f/c_out1 (carry<17>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[19].f/c_out1 (carry<19>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[21].f/c_out1 (carry<21>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[23].f/c_out1 (carry<23>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[25].f/c_out1 (carry<25>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[27].f/c_out1 (carry<27>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[29].f/c_out1 (carry<29>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[31].f/c_out1 (carry<31>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[33].f/c_out1 (carry<33>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[35].f/c_out1 (carry<35>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[37].f/c_out1 (carry<37>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[39].f/c_out1 (carry<39>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[41].f/c_out1 (carry<41>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[43].f/c_out1 (carry<43>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[45].f/c_out1 (carry<45>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[47].f/c_out1 (carry<47>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[49].f/c_out1 (carry<49>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[51].f/c_out1 (carry<51>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[53].f/c_out1 (carry<53>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[55].f/c_out1 (carry<55>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[57].f/c_out1 (carry<57>)
UT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[59].f/c_out1 (carry<59>)
UT5:I4->O	2	0.097	0.515	generate_N_bit_Adder[61].f/c_out1 (carry<61>)
UT3:I0->O	1	0.097	0.279	generate_N_bit_Adder[62].f/Mxor_s_xo<0>1 (answer_62_OBUF)
OBUF:I->O		0.000		answer_62_OBUF (answer<62>)
Total				13.745ns (3.105ns logic, 10.640ns route) (22.6% logic, 77.4% route)

Total REA time to Xst completion: 19.00 secs
Total CPU time to Xst completion: 19.27 secs

4. Memory

Totall memory usage is 4625700 kilobytes

Table 2: Simulation Parameter

Sr No.	Parameters	Proposed Work
1	Type of Multiplier	64-bit Approximate-Multiplier
2	Area	30.33%
3	Delay	13.745ns
4	Accuracy rate	97%
5	Simulation Time	19.00 Secs
6	Power	0.082mW
7	PDP (Power delay product)	1127
8	Frequency	72.7 MHz
9	Throughput	4.6 GHz
10	Memory	4625700 kilobytes
11	Goba Maximum Fan-out	100000

© 2024 IJSRET

Table 3: Comparison with Previous and proposed work

Sr No.	Parameters	Previous work	Proposed work
1	Type of Multiplier	16-bit Multiplier	64- Bit multiplier
2	Area	3319.20 micrometer^2	30.33%
3	Delay	4ns	13.745ns
4	Power	0.112mW	0.082mW
5	PDP (Power delay product)	727	1127

Therefore proposed 64-bit approximate multiplier gives better result in term of calculated parameters. So it can be used in high speed, low area and latency.

Figure 14: Comparison of multiplier bit

Proposed multiplier is designed for 64 bit modified dadda multiplication using approximate multipier while in previous work it is designed only for 16-bit.So it can be say that proposed 64-bit multiplier is advanced of previous multiplier in terms of calculated parameters.

VII. CONCLUSION

Multiplier is an important part in arithmetic processors, the current mobile applications and DSP applications need ICs with high-speed operations but low power consumption.

This paper presents various parameters analysis like power, area, latency, throughput, frequency and power delay product to identify the improved architectures for high speed applications. It was also found that the power consumption of proposed multipliers and the basic building block of more CMOS VLSI circuits depend on the switching activities of the adders. As a result, a proposed multiplier is developed, and the Xilinx software successfully validates the results.

The suggested multiplier pans save a significant amount of power and space while being used in applications where yield quality misfortune is minimal.

Binary and hexadecimal 64-bit inputs will be provided, and the output will be a 64-bit multiplier. The suggested multiplier uses 0.08 mW of electricity and saves 70% of the area.

REFERENCES

1. K. MOUNIKA REDDY , A. LAKSHMI CHENNA KESHAVA , M. SAI KIRAN REDDY , K. KRISHNA SREEKAR , P. SOWJANYA, “Approximate Multiplier Design Using Novel Dual Stage 4:2 Compressor” Jornal of engineering sciences , Vol 13,Issue 05, May /2022 ISSN NO:0377-9254.
2. K. C. Pathak, A. D. Darji and J. N. Sarvaiya, "Low power Dadda multiplier using approximate almost full adder and Majority logic based adder compressors," 2022 IEEE Region 10 Symposium (TENSYMP), Mumbai, India, 2022, pp. 1-6, doi: 10.1109/TENSYMP54529.2022.9864428.
3. A. Kumar and R. Thakur, "Design and Analysis of 64-Bit Approximate Multiplier for Accurate and High-Level Processing," 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2021, pp. 1788-1792, doi: 10.1109/I-SMAC52330.2021.9640815.
4. .P. J. Edavoor, S. Raveendran and A. D. Rahulkar, "Approximate Multiplier Design Using Novel Dual-Stage 4:2 Compressors," in IEEE Access, vol. 8, pp. 48337-48351, 2020, doi: 10.1109/ACCESS.2020.2978773.
5. F. Sabetzadeh, M. H. Moaiyeri and M. Ahmadinejad, "A Majority-Based Imprecise Multiplier for Ultra-Efficient Approximate Image Multiplication," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 11, pp. 4200- 4208, Nov. 2019.
6. H. Saadat, H. Bokhari and S. Parameswaran, "Minimally Biased Multipliers for Approximate Integer and Floating-Point Multiplication," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2623- 2635, Nov. 2018.
7. V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang and J. Han, "Scalable Construction of Approximate Multipliers With Formally Guaranteed Worst Case Error," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 11, pp. 2572-2576, Nov. 2018.
8. S. Venkatachalam and S. Ko, "Design of Power and Area Efficient Approximate Multipliers," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 5, pp. 1782-1786, May 2017.
9. S. Mazahir, O. Hasan, R. Hafiz and M. Shafique, "Probabilistic Error Analysis of Approximate Recursive Multipliers," in IEEE Transactions on Computers, vol. 66, no. 11, pp. 1982-1990, 1 Nov. 2017.
10. W. Liu, L. Qian, C. Wang, H. Jiang, J. Han and F. Lombardi, "Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing," in IEEE Transactions on Computers, vol. 66, no. 8, pp. 1435-1441, 1 Aug. 2017.
11. A. Mokhtari, W. Shi, Q. Ling and A. Ribeiro, "DQM: Decentralized Quadratically Approximated Alternating

© 2024 IJSRET

Direction Method of Multipliers," in IEEE Transactions on Signal Processing, vol. 64, no. 19, pp. 5158-5173, 1 Oct.1, 2016.
12. H. Jiang, J. Han, F. Qiao and F. Lombardi, "Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation," in IEEE Transactions on Computers, vol. 65, no. 8, pp. 2638-2644, 1 Aug. 2016. 57.
13. B. Shao and P. Li, "Array-Based Approximate Arithmetic Computing: A General Model and Applications to Multiplier and Squarer Design," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 62, no. 4, pp. 1081-1090, April 2015.
14. Z. Zhi-Qing Lü and X. An, "Non-conforming finite element tearing and interconnecting method with one Lagrange multiplier for solving large-scale electromagnetic problems," in IET Microwaves, Antennas & Propagation, vol. 8, no. 10, pp. 730-735, 15 July 2014.
15. D. De Caro, N. Petra, A. G. M. Strollo, F. Tessitore and E. Napoli, "Fixed-Width Multipliers and Multipliers-Accumulators With Min-Max Approximation Error," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 9, pp. 2375-2388, Sept. 2013.
16. J. Wang, S. Kuang and S. Liang, "High-Accuracy Fixed-Width Modified Booth Multipliers for Lossy Applications," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 1, pp. 52-60, Jan. 2011.
17. F. Auger, Z. Lou, B. Feuvrie and F. Li, "Multiplier-Free Divide, Square Root, and Log Algorithms [DSP Tips and Tricks]," in IEEE Signal Processing Magazine, vol. 28, no. 4, pp. 122-126, July 2011.
18. I. Park and T. Kim, "Multiplier-less and table-less linear approximation for square and square-root," 2009 IEEE International Conference on Computer Design, Lake Tahoe, CA, 2009, pp. 378-383.
19. P. Bettiol and R. Vinter, "Existence of feasible approximating trajectories for differential inclusions with obstacles as state constraints," Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, Shanghai, 2009, pp. 8624-8629.
20. M. Rentzsch, F. Gleisberg, H. Guldner, F. Benecke and C. Ditmanson, "Closed analytical model of a 20 kV output voltage, 800 W output power series-parallelresonant converter with Walton Cockroft multiplier," 2008 IEEE Power Electronics Specialists Conference, Rhodes, 2008, pp. 1923-1929.
21. I. W. Hauser, "A Multiplier-less Algorithm for Evaluating Intrinsic and Nonlinear Functions," 2006 49th IEEE International Midwest Symposium on Circuits and Systems, San Juan, 2006, pp. 525-529.
22. G. Jovanovic-Dolecek and S. K. Mitra, "Multiplier-free FIR filter design based on IFIR structure and rounding," 48th Midwest Symposium on Circuits and Systems, 2005., Covington, KY, 2005, pp. 559-562 Vol. 1. 58
23. A. Drozd, R. Al-Azzeh, J. Drozd and M. Lobachev, "The logarithmic checking method for on-line testing of computing circuits for processing of the approximated data," Euromicro Symposium on Digital System Design, 2004. DSD 2004., Rennes, France, 2004, pp. 416-423.
24. J. Hauser and C. Purdy, "Approximation of Nonlinear Functions with Genetic Algorithms" Embedded Systems Programming Magazine - Web Exclusive. http://www.embedded.com/story/OEG20030129S0058 Source code and other related information may be found at ftp://ftp.embedded.com/pub/2003/hauser-purdy. February 2003.
25. A. Drozd, "On-line Testing of Computing Circuits at Approximate Data Processing," Radioelectronics & Informatics, Vol. 3, pp. 113 - 116, 2003.
26. A.Drozd, "Base concepts of on-line testing of the computing circuits for the approximated data processing," Electronic Simulation, Vol. 25, No 1, pp. 73-79, 2003.
27. J. Drozd, "Fast performance of operation of the units amount division for a parallel code," Works of the Odessa polytechnic university, Odessa, Ukraine, Vol. 1, pp. 110 - 113, 2002

© 2024 IJSRET