Energy Efficient Rounding Technique Approximate Multiplier Configuration with Approximate Compressors

Hima Bindu.D [1] Himabindudattada@gmail.com

Dr. Mohammed Jabirullah[2] drjabirullah@lords.ac.in

Narsaiah Domala [3] dnarsaiah@lords.ac.in

[1]PG Scholar, Dept of ECE, Lords Institute of Engineering & Technology,
Hyderabad, India
[2]Associate Professor & HOD, Dept of ECE, Lords Institute of Engineering & Technology,
Hyderabad, India
[3] Associate Professor r, Dept of ECE, Lords Institute of Engineering & Technology,
Hyderabad, India

ABSTRACT: Accurate computing may help with data processing for error-resistant applications including signal and image processing, computer vision, and data mining. The expense of improving the circuit characteristics limits the precision that may be used for approximations. The circuit designer uses the target accuracy as a criterion for balancing precision with the circuit's capabilities. In this situation, using the rounding method is a great way to keep the transaction under control. According to the simulation findings, the proposed multiplier outperforms its counterparts in terms of strength, range, speed, and energy. The degree of accuracy must be monitored while spending as little money on hardware as feasible while dealing with an increasing amount of data.

Keywords**—**Data Processing, Digital Arithmetic, Approximate computing, Energy efficient, Hi-performance, Rounding Technique.

I.INTRODUCTION

According to the simulation findings, the proposed multiplier outperforms its counterparts in terms of strength, range, speed, and energy. The degree of accuracy must be monitored while spending as little money on hardware as feasible while dealing with an increasing amount of data. Countless projects have to make a choice between accuracy and delay energy in this regard because to budget constraints. The multiplier's basic building blocks are partial product production, partial product reduction, and packaging. This article offers a modern solution for input blocks prior to partial product creation: a rounding method. As a way to manage and control the error rate, accuracy curves are essential tools. Various multiplier block rates call for different implementations of the same set of algorithms. The input block employs either a 16-bit or 32-bit rounding technique, depending on the required degree of accuracy. Products that have been partially produced fall into one of two categories: active products or passive products. Because they are both zeros, there is no requirement for any inactive partial products in the compressor reduction phase. As a consequence, compression is no longer necessary for as many objects. To improve the latency of the circuit, less time is spent isolating active partial goods. OR gates provide correctness while also lowering hardware in the compressed output block, which includes both precise and estimated compressor blocks.

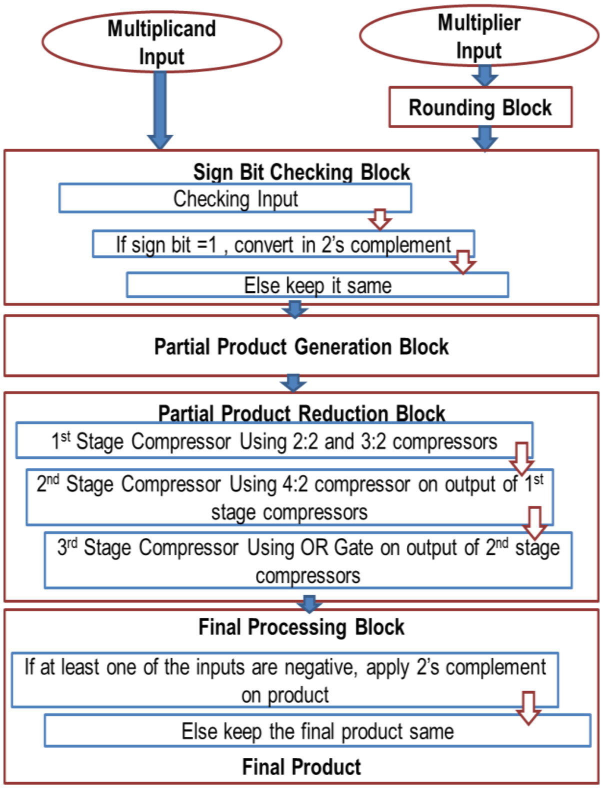
II.RELATED STUDIES:

The minimising or truncation of partial products was the focus of experiments that were significant in multiplying estimations. Hardware-level methodologies that employ fewer but more precise and highly effective energy components may be divided into two main categories: software-level procedures that limit measurements or memory accesses in order to increase efficiency at the cost of output precision. One of the fastest and most energy-efficient multipliers ever devised was recently presented[1]. It was possible to improve performance while also reducing energy usage by eliminating the computationally expensive multiplication step (by as much as 65 percent). According to [2]'s hardware interpretation, the design has configurable kernels and an overflow-resistant limiter. [2] Recently, many methods for loosening up a single computer device component (such as a functional unit[3]) have been described in the literature to allow for better design. The propagation time and energy consumption are both reduced by using the indirect multiplier architecture suggested by the authors in [4]. There was no particular emphasis on pre-computing discoveries in any of the research. Before temporary multipliers are applied, the value of I is significantly reduced by rounding the input numbers first.

III. PROPOSED Architecture OF MULTIPLIER APPROXIMATES

A. Diagram of a schematic

By using an approximate multiplier, rounded data may be used in multiplication. The proposed technique begins with a partial product generation rounding approach before moving on to the input. This multiplier building method has a detailed specification map in Figure 1. Initially, it is rounded up by sending its two inputs via a rounding tube (Multiplicand and Multiplier). The Sign bits of both inputs are retained until the multiplication begins, and the multiplication value's output sign is calculated using the input signs.. Finally, on the test, the correct symbol is used. Positive number combinations should be transformed to their 2's complement before being used as input blocks. When given an N-bit input, traditional multipliers generate partial products (half products). However, the rounding technique generates a mixture of active and inactive component products. "1" denotes an active partial product in the multiplier equation. As a result, after rounding, a complete Multiplicand row is generated. Partial items with all 0s are inactive. So, throughout the cutting process, they're left exposed.


Multiplicand Input
Multiplier Input
Rounding Block
Sign Bit Checking Block
Checking Input
If sign bit =1 , convert in 2's complement
Else keep it same
Partial Product Generation Block
Partial Product Reduction Block
1st Stage Compressor Using 2:2 and 3:2 compressors
2nd Stage Compressor Using 4:2 compressor on output of 1st stage compressors
3rd Stage Compressor Using OR Gate on output of 2nd stage compressors
Final Processing Block
If at least one of the inputs are negative, apply 2's complement on product
Else keep the final product same
Final Product

B. Rounding data

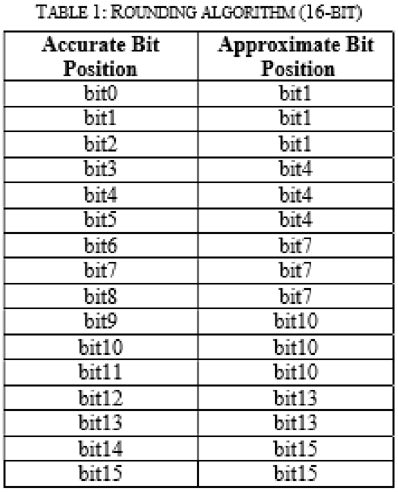
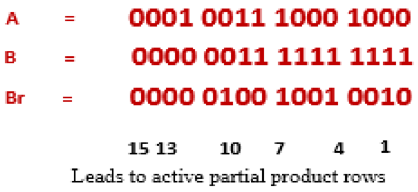


TABLE 1: ROUNDING ALGORITHM (16-BIT)

Accurate Bit Position	Approximate Bit Position
bit0	bit1
bit1	bit1
bit2	bit1
bit3	bit4
bit4	bit4
bit5	bit4
bit6	bit7
bit7	bit7
bit8	bit7
bit9	bit10
bit10	bit10
bit11	bit10
bit12	bit13
bit13	bit13
bit14	bit15
bit15	bit15


A = 0001 0011 1000 1000

B = 0000 0011 1111 1111

Br = 0000 0100 1001 0010

To ensure consistency, data rounding on inputs requires significant work. A basic assumption says that rounding lower bits reduces inaccuracy more than rounding higher bits. As a consequence, bit location rounding weights have been given depending on the magnitude of the suggested technique. Figures 2 and 3 illustrate this method's 16-bit error location curve, in which the lower bits are given less weight and the higher bits are given more. As a result of rounding issues, the exact bit position may vary somewhat from the chip's actual location. Table 1 shows the rounded bit values for each precise bit. The error gap widens as the value of the bit position increases. Let's use the following as an illustration: 3 A and B are specified as inputs in the scenario, with B as the rounded integer. The "Rounding Technique" looks for a "1" in the "X" bit position and either assigns a "1" properly to the "Y" bit position or not.

C. Partial product reduction

Various types of compressors are used to compress partially reduced products throughout the reduction process. The proposed approach makes certain that the number of component rows is always the same no matter how many components are added. Figure 6 shows an alternate 16-bit architecture for reducing partial products to a maximum of six rows. In all conventional techniques, N-bit inputs are combined to produce partial N/N products that may be used in many applications. As the bit count increases, the length of Partial products becomes shorter since the computation cost goes down by O(N2). The proposed technique has 16 bits O(N6) and 32 bits O in computational complexity (N113).

In order to make the design easier to understand, input values A, B, and Br (from fig. 6(right)) are used. The multiplier value is 'B,' and it is rounded to 'Br,' initially. By mixing the inputs, NN partial products are created.. In the context of multiplier data rounding, NN partial products combine active and inactive partial products. Figure 6 shows the whole row of Multiplicand multiplied by the coefficient "1," which is the consequence of rounding. On the other hand, partially inactive products are the zero value lines that have been multiplied by "0." As a consequence, they aren't included in the reduction process. Products that are only partially active, on the other hand, can only increase

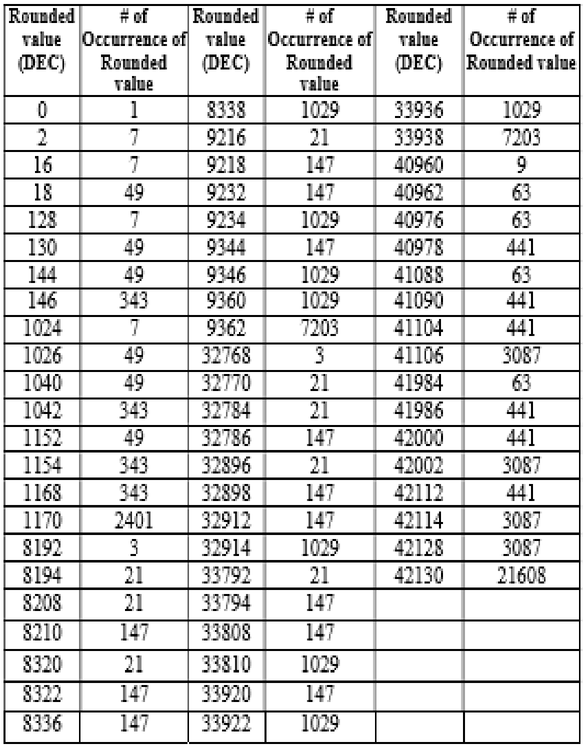
This means that before packaging any unfinished items, we had to take them out of service completely. With this method, you may utilise more power, save more space, and save more time, all while increasing the system's performance. Before being transmitted, the active partial products are compressed and packed using a three-level compression method. Partially compressed objects are initially compressed using full and half adders. A 4:2 compressor with 16bit inputs and a 9:2 compressor with 32bit inputs are utilised to further decrease the 1st stage compression performance.

Illustration. 6 is an example of a comparable procedure on a real-world situation (to the right). To create a completed product, the output of the second stage compressor is compressed using an OR gate. Complete adders, rather than OR, are used in its place. The OR gate replaces the whole adder, resulting in a significant reduction in the amount of space and energy used.

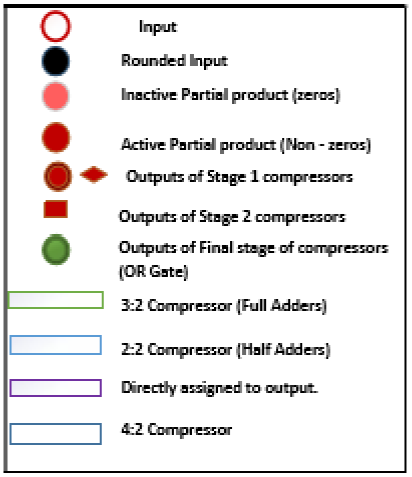
IV. ROUNDING ERROR ANALYSIS

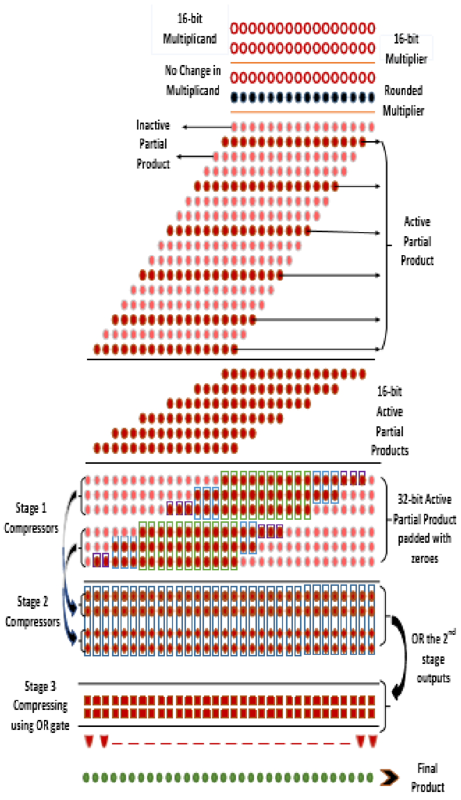
As shown in the previous sections, there is only one rounded Input (the Multiplier). As an example, look for a 16-bit multiplier to estimate the rounding error. There are two inputs, Multiplicand and Multiplier, and the answer of Multiplier is rounded to the closest whole number. In Fig. 2 from Section III, the rounding technique produces 16-bit integer rounded values ranging from 0 to 65535. Figure 4 illustrates the relationship between rounded values and their occurrences as an illustration of this. The resultant phase diagram shows how comparable the rounded values are. As a result, phase sizes are often small, indicating lower error and thus higher accuracy. However, increasing the step size may lead to somewhat more inaccuracy. In order to improve accuracy, it's important to perform efficient data processing before multiplication so that you can devote all of your time and resources to this one area.

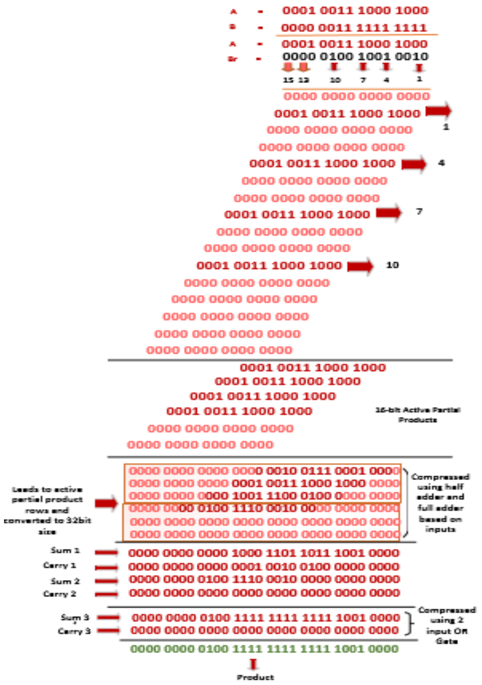
TABLE 2: ROUNDED VALUES AND THEIR OCCURRENCES RATE



Rounded value (DEC)	# of Occurrence of Rounded value	Rounded value (DEC)	# of Occurrence of Rounded value	Rounded value (DEC)	# of Occurrence of Rounded value
0	1	8338	1029	33936	1029
2	7	9216	21	33938	7203
16	7	9218	147	40960	9
18	49	9232	147	40962	63
128	7	9234	1029	40976	63
130	49	9344	147	40978	441
144	49	9346	1029	41088	63
146	343	9360	1029	41090	441
1024	7	9362	7203	41104	441
1026	49	32768	3	41106	3087
1040	49	32770	21	41984	63
1042	343	32784	21	41986	441
1152	49	32786	147	42000	441
1154	343	32896	21	42002	3087
1168	343	32898	147	42112	441
1170	2401	32912	147	42114	3087
8192	3	32914	1029	42128	3087
8194	21	33792	21	42130	21608
8208	21	33794	147		
8210	147	33808	147		
8320	21	33810	1029		
8322	147	33920	147		
8336	147	33922	1029		


Input
Rounded Input
Inactive Partial product (zeros)
Active Partial product (Non - zeros)
Outputs of Stage 1 compressors
Outputs of Stage 2 compressors
Outputs of Final stage of compressors (OR Gate)
3:2 Compressor (Full Adders)
2:2 Compressor (Half Adders)
Directly assigned to output.
4:2 Compressor


16-bit Multiplicand
16-bit Multiplier
No Change in Multiplicand
Rounded Multiplier
Inactive Partial Product
Active Partial Product
16-bit Active Partial Products
Stage 1 Compressors
32-bit Active Partial Product padded with zeroes
Stage 2 Compressors
OR the 2nd stage outputs
Stage 3 Compressing using OR gate
Final Product


A - 0001 0011 1000 1000
B - 0000 0011 1111 1111
A - 0001 0011 1000 1000
Br - 0000 0100 1001 0010
15 13 10 7 4 1
0000 0000 0000 0000
0001 0011 1000 1000
1
4
7
10
16-bit Active Partial Products
Leads to active partial product rows and converted to 32bit size
0000 0000 0000 0000 0010 0111 0001 0000
0000 0000 0000 0001 0011 1000 1000 0000
0000 0000 0000 1001 1100 0100 0000 0000
0000 0000 0100 1110 0010 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
Compressed using half adder and full adder based on inputs
Sum 1 0000 0000 0000 1000 1101 1011 1001 0000
Carry 1 0000 0000 0000 0001 0010 0100 0000 0000
Sum 2 0000 0000 0100 1110 0010 0000 0000 0000
Carry 2 0000 0000 0000 0000 0000 0000 0000 0000
Sum 3 0000 0000 0100 1111 1111 1111 1001 0000
Carry 3 0000 0000 0000 0000 0000 0000 0000 0000
Compressed using 2 input OR Gate
0000 0000 0100 1111 1111 1111 1001 0000
Product

Because the phase sizes are smaller, the inaccuracy is lower, and the precision is higher. However, increasing the step size may lead to somewhat more inaccuracy. In order to improve accuracy, it's important to perform efficient data processing before multiplication so that you can devote all of your time and resources to this one area.

A plan has developed as a consequence of the extensive research. After that, we computed the probability that an integer between 0 and 65535 would be rounded up or down to the next whole number. The yellow dotted lines in Figure 4 show how probabilities spread rounded values. Up to a rounded value radius of 9360, there is a small possibility of higher accuracy. In spite of this, the chance will drop from 40960 onwards. Changing the rounding pattern near the centre, where likelihood is greatest, has the potential to increase accuracy. Future research and study will concentrate on these areas in order to improve the algorithm at the expense of adding additional hardware. According to Table 2, rounded numbers were utilised more often than the original input values, as can be seen.

V.Extension:

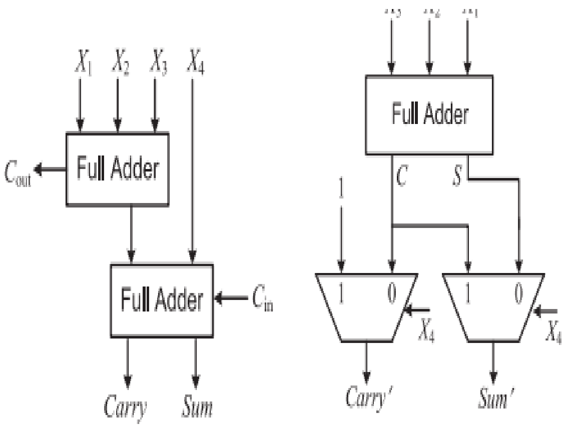
When Cin is fed into the compressor, the outputs are Cout and Carry, which are both order one binary bit higher. This design uses a 4–2 compressor, as seen in Figure 1a. A better 4–2 compressor design than the one proposed in [9] is used to compare the projected output of compressors. Equations (1) and (2) depict the compressor's functioning (3).

In order to reduce the approximation rounding multiplier's latency, we used a carry propagate adder in conjunction with an exact compressor as part of this project's fourth phase.

$$Sum = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{in} \qquad (1)$$

$$C_{out} = (X_1 \oplus X_2)X_3 + \overline{(X_1 \oplus X_2)}X_1 \qquad (2)$$

$$Carry = (X_1 \oplus X_2 \oplus X_3 \oplus X_4)C_{in} + \overline{(X_1 \oplus X_2 \oplus X_3 \oplus X_4)}X_4 \qquad (3)$$



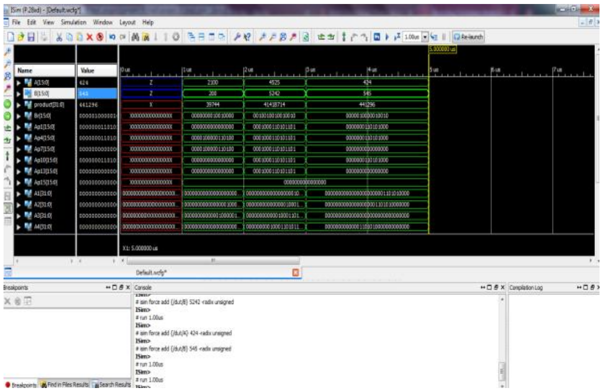
(a) Exact compressor (b) Approximate Compressor

VI.RESULTS

The new project's functionality is tested in a virtual environment. After functional verification, the RTL model is utilised in the synthesis process using the Xilinx ISE tool. The RTL model may be transformed into a certain technology library's gate level net list by utilising an RTL-to-gate level net list converter. The ISE tool from Xilinx worked with a broad range of Spartan 3E chips. Device speed "-5" was used for this design to synthesise the XC3S500E and FG320 package.

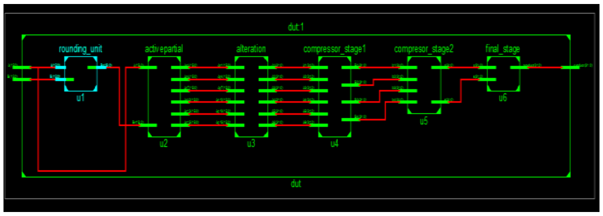
Based on the information produced by this design, the following analysis was carried out on it:

SIMULATION RESULTS:

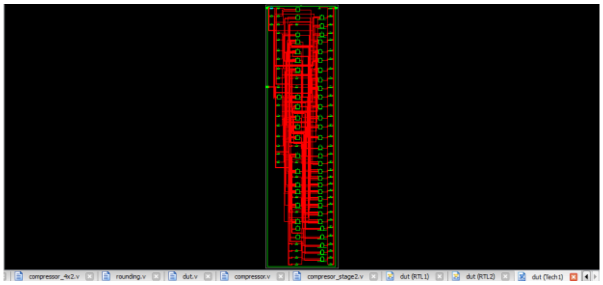


```
# isim force add {/dut/B} 5242 -radix unsigned
ISim>
# run 1.00us
ISim>
# isim force add {/dut/A} 424 -radix unsigned
ISim>
# isim force add {/dut/B} 545 -radix unsigned
ISim>
# run 1.00us
ISim>
# run 1.00us
```

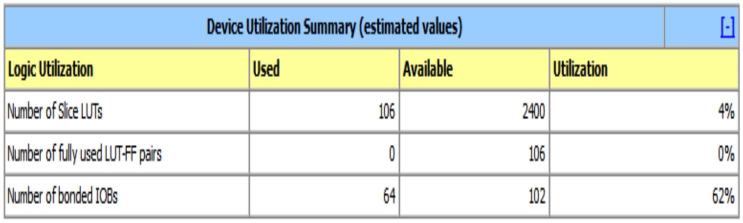
RTL SCHEMATIC DIAGRAM



TECHNOLOGICAL SCHEMATIC

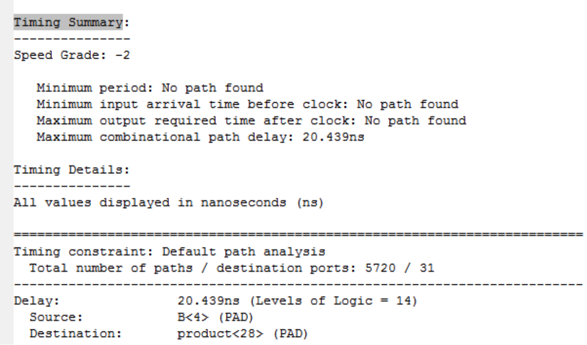


DESIGN SUMMARY:



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	106	2400	4%
Number of fully used LUT-FF pairs	0	106	0%
Number of bonded IOBs	64	102	62%

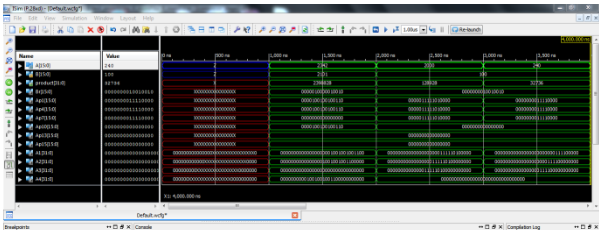
TIMING SUMMARY:



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 20.439ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 5720 / 31
-------------------------------------------------------------------------
Delay:               20.439ns (Levels of Logic = 14)
  Source:            B<4> (PAD)
  Destination:       product<28> (PAD)
```

EXTENSION RESULTS

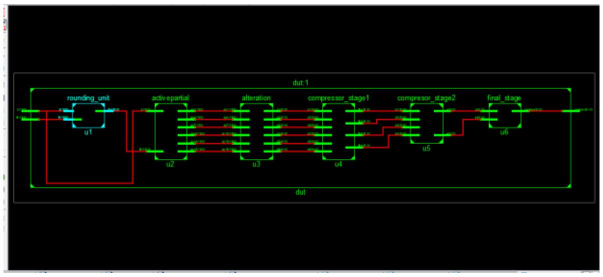
SIMULATION RESULTS



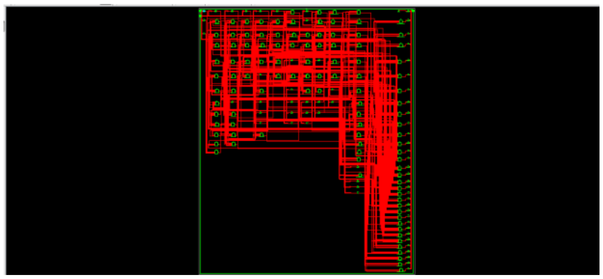
ISim (P.28xd) - [Default.wcfg*]

Name	Value
A[15:0]	240
B[15:0]	100
product[31:0]	32736
Br[15:0]	0000000010010010
Ap1[15:0]	0000000011110000
Ap4[15:0]	0000000011110000
Ap7[15:0]	0000000011110000
Ap10[15:0]	0000000000000000
Ap13[15:0]	0000000000000000
Ap15[15:0]	0000000000000000
A1[31:0]	0000000000000000
A2[31:0]	0000000000000000
A3[31:0]	0000000000000000
A4[31:0]	0000000000000000

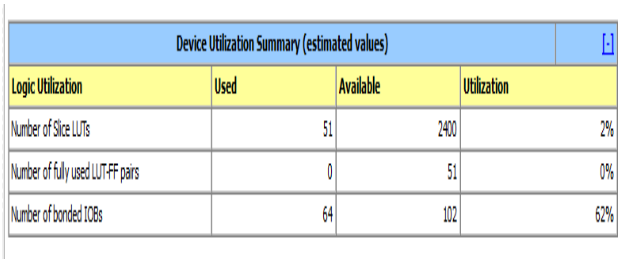
RTL BLOCK DIAGRAM:



TECHNOLOGICAL SCHEMATIC:

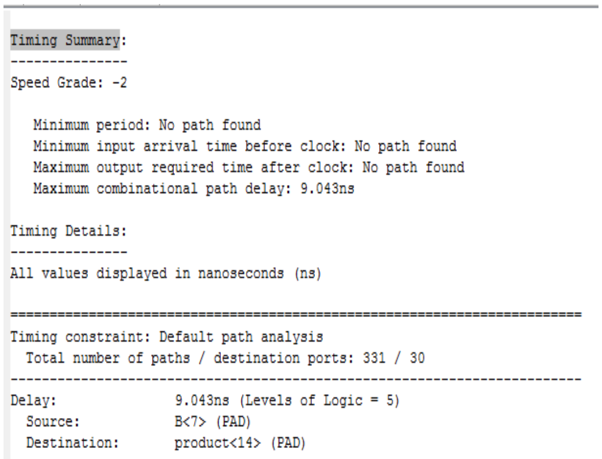


DESIGN SUMMARY:



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	51	2400	2%
Number of fully used LUT-FF pairs	0	51	0%
Number of bonded IOBs	64	102	62%

TIMING SUMMARY:



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 9.043ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 331 / 30
-------------------------------------------------------------------------
Delay:               9.043ns (Levels of Logic = 5)
  Source:            B<7> (PAD)
  Destination:       product<14> (PAD)
```

COMPARISON TABLE

MODULE	DELAY
ROUNDING MULTIPLIER WITH EXACT MUTIPLIER	20.43ns
ROUNDING MULTIPLIER WITH APPROXIMATE MUTIPLIER	**9.043ns**

VII.CONCLUSION

The suggested Algorithm is shown to be the most power-area delay and PDP efficient when compared to existing algorithms for signed and unsigned results (16-bit and 32-bit). This is the first time that the rounding method has been used to an approximation multiplier with just one rounding step (as seen in fig. 2 and Table 1). As you can see from the pattern, there are regions with less accuracy and areas with higher precision where rounding is more likely (fig. 4 and Table. 2). To fine-tune the rounding patterns, only a bit additional hardware is required. Dynamic rounding vs. fixed rounding are both options for rounding The suggested method outperforms existing algorithms for signed and unsigned data in terms of delayed power space and PDP performance (16-bit and 32-bit). First time a rounding approach using fixed active, partial product lines on an approximated multiplier has been used to analyse the redness methodology (as seen in fig. 2 and Table 1). With this pattern of rounding, there is a possibility that certain regions will be less accurate than others (fig. 4 and Table. 2). Even additional hardware is not needed to alter the rounding patterns. The pattern of rounding may be changed dynamically or continuously.

VI. REFERENCES

[1]R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, M. Pedram, "RoBA multiplier: A rounding-based approximate multiplier for high- speed yet energy-efficient digital signal processing", IEEE Trans. Very Large-Scale Integer. (VLSI) Syst., vol. 25, no. 2, pp. 393-401, Feb. 2017.

[2] M. Van Leussen, J. Huisken, L. Wang, H. Jiao, J. P. De Gyvez, "Reconfigurable Support Vector Machine Classifier with Approximate Computing", 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, pp. 13-18, jul 2017.

[3] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, Massoud Pedram, LETAM, Computers and Electrical Engineering, Science Direct, v.63 n.C, p.1-17, October 2017

[4] S. Hashemi, R. I. Bahar, and S. Reda, “DRUM: A Dynamic Range Unbiased Multiplier for approximate applications,” IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 418- 425, Nov. 2015.

[5] Bharat Garg, G, K Sharma: Low Power Signal Processing via Approximate Multiplier for Error Resilient Applications, 11th International Conference on Industrial and Information System (ICIIS), pp. 546-551, 2016.

[6] H.Bessalah, K.Messaoudi, M.Issad, N.Anane, M.Anane: Left to Right Serial Multiplier for Large Numbers on FPGA, Proceedings of the 2009 IEEE International Conference on Mechatronics Malaga, Spain, pp. 1-6, April 2009 [7] A. Kahng and S. Kang, “Accuracy-configurable adder for approximate arithmetic designs,” in Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, pp. 820–825, june 2012.