International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-12 Issue-1, May 2023

Design & Implementation of Approximate 7:2 Compressor Based 16-bit Dadda Multiplier using Verilog

P V V Satyanarayana, M. Venkanna, S. Jayasri, J. Vasavi Kalyani, B. Ramjee

Abstract: ***Nowadays, technology is growing at a rapid rate. Notably, the use of electronics is increasing in various ways, depending on their intended purpose and individual preferences. In this regard, multipliers play a vital role because they enable us to perform complex arithmetic operations involving large numbers more efficiently. Instead of performing a series of addition or subtraction operations, a multiplier allows us to operate a single step within no time, which is the challenge of today's world. In addition to being more efficient, multipliers also have practical applications in fields such as engineering, computer science, and cryptography, which are also used, for example, in the design of digital circuits and the encryption and decryption of data. Overall, multipliers play a crucial role in mathematics and its applications, serving as essential tools for performing complex computations efficiently. Compressors play a vital role in achieving high-speed multipliers. In error-resilient applications such as Image processing, Multimedia, and Matrix multiplication, approximate computing is used, which provides meaningful results faster with lower power consumption. In previous work, the compressors were designed using full adders, which provide accurate results. The 4:2 and 5:2 approximate compressors are then introduced with 18% delay reduction and ADP reduction up to 52%. The further work concentrated on implementing a 7:2 Approximate Compressor-based multiplier to enhance the performance of multipliers. The proposed design is expected to provide the maximum reduction in area, delay, and power consumption, while achieving a speed improvement compared to the 4:2 and 5:2 compressor-based approximate multipliers.***

Keywords: Adder, Multiplier, 4:2 Compressor, 5:2 Compressor, 7:2 Compressor and Verilog.

Manuscript received on 27 March 2023 | Revised Manuscript received on 08 March 2023 | Manuscript Accepted on 15 May 2023 | Manuscript published on 30 May 2023.

*Correspondence Author(s)
P V V Satyanarayana, Department of Electronics & Communication Engineering, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem. (A.P), India E-mail: vvsnarayanapudi@gmail.com, ORCID ID: https://orcid.org/0009-0001-6971-1391
M. Venkanna*, Department of Electronics & Communication Engineering, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem. (A.P), India. E-mail: madasuvenky263@gmail.com, https://orcid.org/0009-0007-3996-7995
S. Jayasri, Department of Electronics & Communication Engineering, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem. (A.P), India. E-mail: jayasrisampangi@gmail.com, https://orcid.org/0009-0006-6567-5502
J. Vasavi Kalyani, Department of Electronics & Communication Engineering, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem. (A.P), India. E-mail: vasavijangam@gmail.com, https://orcid.org/0009-0007-0496-05706
B. Ramjee, Department of Electronics & Communication Engineering, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem. (A.P), India. E-mail: ramjeebuddarapu@gmail.com, https://orcid.org/0009-0002-4547-4253

I. INTRODUCTION

Multipliers are one of the most fundamental components in computer arithmetic and are frequently used in various digital signal processors, computer graphics, scientific calculations, image processing, and other applications. Three sequential phases are present in the process of multiplication. 1. Partial product generation 2. Partial product reductions. 3. Propagating addition is the final computation. A significant number of compressors are used in multiplication to carry out partial product addition in high-speed, error-resilient applications. Typically, several different types of compressors, such as 3:2, 4:2, and 5:2, have been extensively used to achieve partial product addition and overall reduction of partial products during multiplication. However, this primarily contributes to the overall delay, power consumption, and area. So the latency of this stage is decreased by using compressors [11, 12, 13, and 14]. Most of the time, multipliers cause overhead to the computer arithmetic units. The multiplier unit consumes the maximum power and also occupies more space in the filter design. Therefore, the compressor-based multipliers were designed to minimize these effects. These multipliers will perform electrically more effectively in terms of area, power, and delay. A compressor with the size x: y denotes that it has 'x' number of inputs and 'y ' number of outputs [1].

High-order compressors have the potential to simultaneously reduce both the vertical critical path and stage operations, resulting in improved power and speed performance [2]. Exact and accurate calculations are not necessarily required in many Signal or Image processing and Multimedia applications. Since these systems are error-tolerant and generate output suitable for human perception. Approximate compressors are used to improve the performance of circuits created for high-speed applications. Xilinx ISE is the computer program used to simulate and synthesise multipliers and compressors. For the synthesis and analysis of HDL designs, Xilinx ISE is a discontinued software package from the company. It was primarily used to generate embedded firmware for the FPGA (field programmable gate array) and CPLD (complex programmable logic device) product families as well [3].

II. EXISTING WORK

The approximate multipliers using 4:2 compressors, 5:2 compressors and the exact 7:2 compressors were proposed earlier.

Mostly used 3:2 compressors (full adders) require five steps in 8-bit multiplication. It is known that 4-2 compression can reduce the number of steps in partial product reductions, offering efficient hardware costs for designing fast arithmetic units. Similarly, the 5:2 Compressor-based multipliers have better optimised performance as compared to the 4:2 Compressor-based multipliers. These compressors are commonly implemented by using the full adders. The number of full adders that are used in implementing the compressors depends on the number of inputs that they have. The compressors, which are designed using full adders, i.e., 3:2 compressors, are referred to as exact compressors [4].

2.1 4:2 Compressor:

The general structure of a 4:2 compressor using full adders is shown in Figure 1. It consists of five inputs, three outputs and two cascaded full adders. A1, A2, A3, A4, and Cin are the inputs, and Cout, Carry, and Sum are the outputs on applying approximation to 4:2 compressors. The output can be reduced to two partial products at the last stage. Approximation is done by eliminating Cout. The 4:2 compressor shows a delay of 1.065 ns and device utilisation of up to 3%.

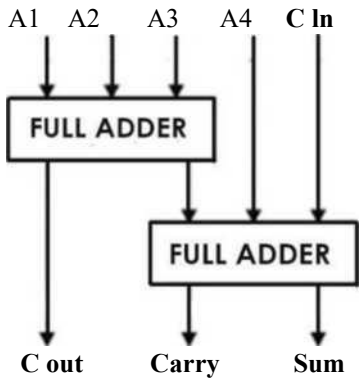


Figure 1. 4:2 compressor using a full adder

2.2 5:2 compressor:

The basic structure of the 5:2 Compressor using full adders is shown in Figure 2. This general block diagram of a 5:2 compressor consists of three full adders cascaded together. It has five inputs (X1, X2, X3, X4, X5), along with carry inputs Cin1 and Cin2, and two outputs (Sum and Carry), along with a carry output Cout. The 5:2 compressor shows a delay of 1.295 ns and device utilisation of up to 5%. So that the 5:2 compressor gives more delay than the 4:2 Compressor as it contains one more full-adder in addition [5].

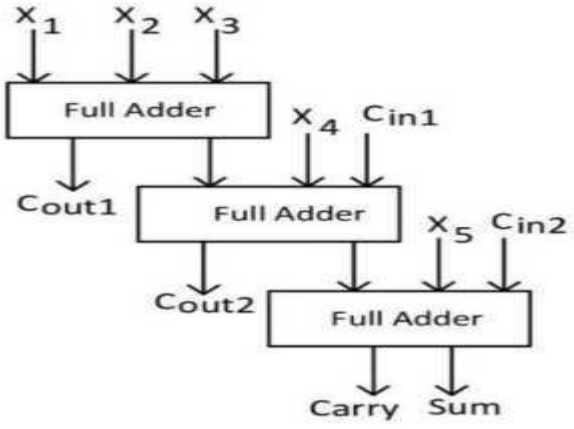


Figure 2 5:2 compressor using a full adder

2.3 7:2 compressor:

The general block diagram of a 7:2 compressor is shown in Figure 3. The 7:2 Compressor using full adders is shown in Figure 3 below. It consists of 7 inputs (A1, A2, A3, A4, A5, A6, A7) and 2 carries from previous stage (Cin1, Cin2). It has 4 outputs consisting of one sum and three carries. The simulation results of 7:2 show a delay of 1.859 ns and device utilisation of up to 8% compared to 4:2 and 5:2 exact compressors. The 7:2 compressor shows an increase in delay, as it requires more full adders compared to the 4:2 and 5:2 compressors, due to the increased number of inputs. To reduce the delay, the approximate compressors are introduced [7, 8]. The general block diagram of a 7:2 compressor is shown in Figure 3.

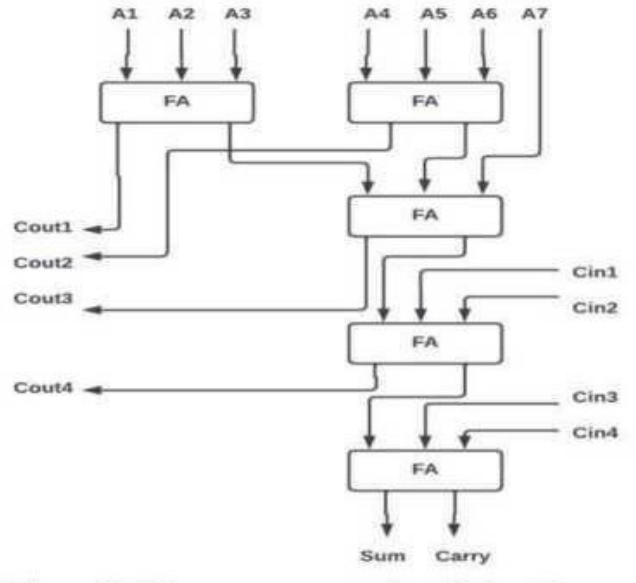


Figure 3. 7:2 compressor using a full adder

2.4 Dadda Multiplier Method:

The Dadda multiplier is, of course, a more efficient and faster method of multiplication than traditional methods, such as the shift-and-add method or the Booth algorithm. The Dadda multiplier is one type of digital circuit that multiplies two unsigned integers. Dadda multiplier uses less space and is faster than the Wallace tree multiplier. The area reduction in this multiplier is caused by a general decrease in partial product levels, achieved by using the Dadda compression technique, which is often employed in digital multipliers.

The Dadda multiplier uses a tree-like structure to perform the multiplication operation. The tree is built up of several stages, with each stage consisting of several parallel adders. The first stage of the tree consists of half adders, which take two bits as inputs and produce a sum and carry bit as outputs. The second stage consists of full adders, which take three bits as inputs and produce a sum and carry bit as outputs. Each subsequent stage has more full adders than the previous stage. To perform a multiplication using the Dadda multiplier, the two binary numbers are first represented as bit arrays of equal length. The two bit arrays are then placed at the top of the tree, with the bits flowing down the tree from the top to the bottom.

Retrieval Number: 100.1/ijrte.A75700512123
DOI: 10.35940/ijrte.A7570.0512123
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

At each stage of the tree, the bits are added together using the parallel adders, and the results are passed down to the next stage of the tree. The final result is obtained from the output of the lowest stage of the tree.

Wallace tree and Dadda multipliers both have the same number of partial product addition levels for 2 bit operand multiplication, but as the operand bit size increases for 4-bit 8 bit, and more, the partial product addition levels of the Dadda multiplier significantly decrease compared to the Wallace tree multiplier, and the Dadda multiplier is also faster [6].

Table 1: No. of reduction stages for Dadda Multiplier

Bits in Multiplier(N)	Number of Stages
3	1
4	2
5 < N< 6	3
7 < N < 9	4
10 <N< 13	5
14 <N< 19	6
20 < N < 28	7
29 < N < 42	8
43 < N < 63	9
63 < N < 94	10

The reduction rules are as follows in terms of rules:

Rule 1: When three wires of identical weight are fed into a full adder, the output will consist of one wire of the same weight as the inputs and an additional wire of higher weight for every group of three input wires.

Rule 2: If two wires of equal weight remain and the current count of output wires with that weight is divisible by 3 with a remainder of 2, then feed them into a half adder. Otherwise, simply forward them to the next layer.

Rule 3: When there is only one wire remaining, it should be connected to the next layer. This process performs the required number of additions to maintain the output weights close to a multiple of 3, which is ideal when utilising full adders as 3:2 compressors.

Rule 4: If a layer contains a maximum of three input wires for each weight, then it will be the final layer in the Dadda tree. In such cases, the Dadda tree will utilize half adders more extensively (though not to the same extent as in a Wallace multiplier) to guarantee that each weight has only two outputs.

Consequently, the second rule mentioned earlier undergoes a modification as follows: If there are two wires of the same weight left, and the current number of output wires with that weight is equal to 1 or 2 (modulo 3), input them into a half adder of the digital circuit. Otherwise, it will pass to the next layer [9]. Some similar compressor circuits can be observed in [

Figure 4 below illustrates the 16-bit by 16-bit data multiplier, showing how the partial product reduction is performed in six consecutive stages using a flowchart to obtain the final addition, which is stored in the output register. Finally, the output will be available, as shown in the flow chart.

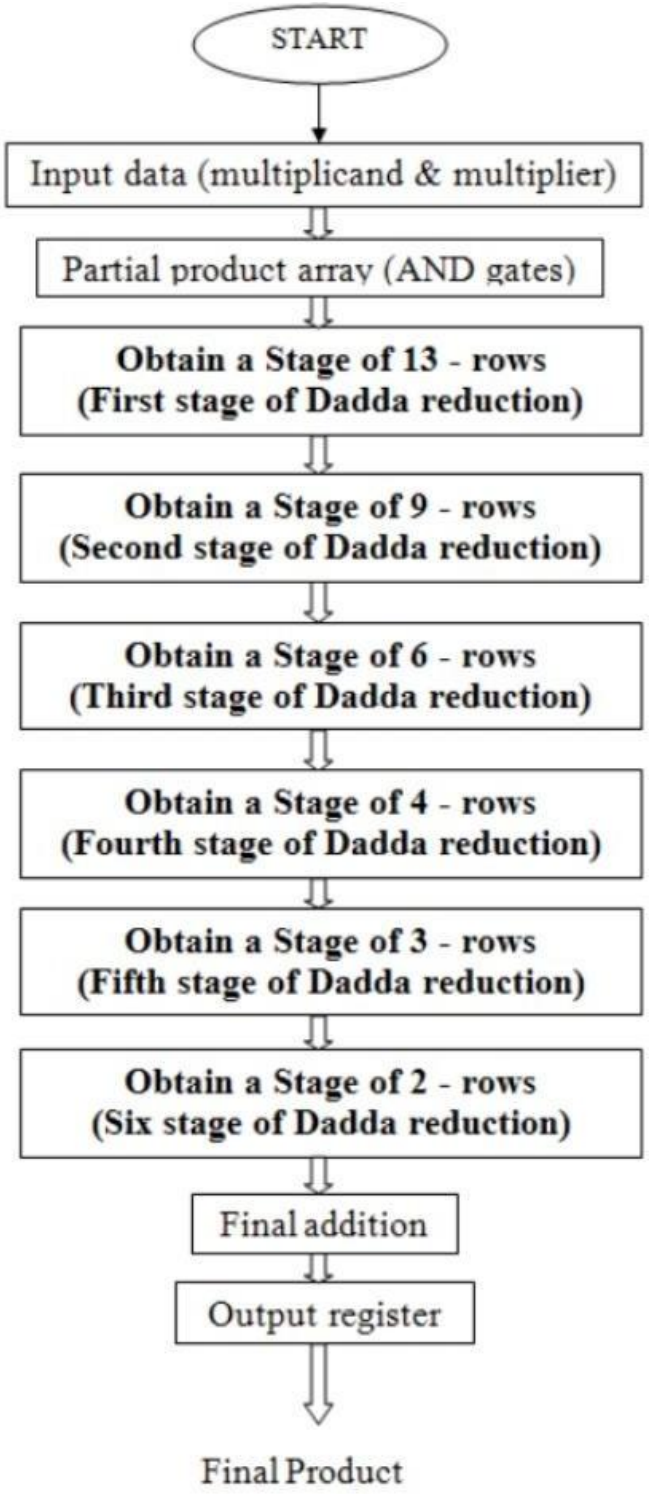


Figure 4: Flow chart of 16*16 Dadda multiplier

The Dadda multiplier algorithm efficiently reduces the number of partial products and performs the multiplication with fewer logic gates than traditional multiplication methods. This 16-bit Dadda multiplier requires six stages of reduction to obtain the final product. The partial product reduction is represented in terms of a dot diagram, as shown in Figure 5.

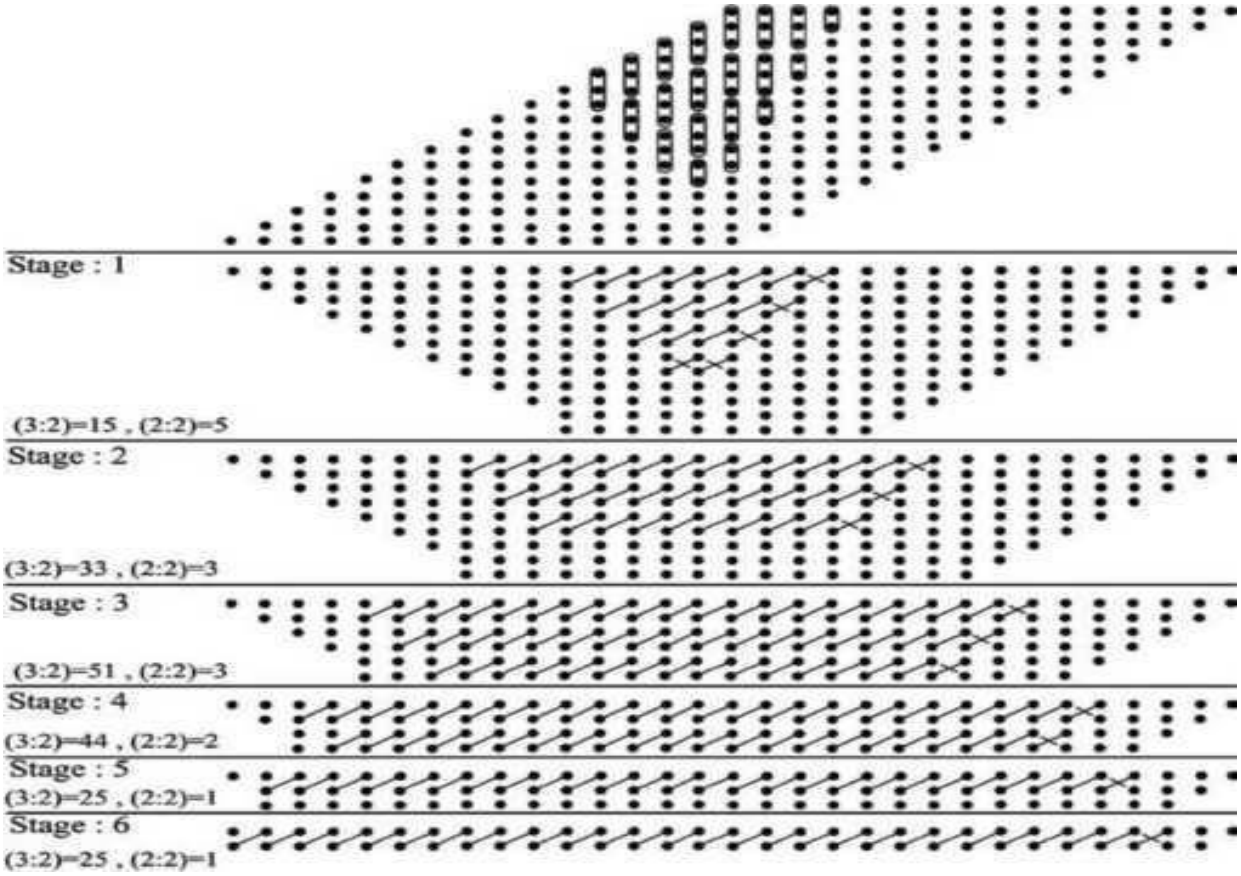


Figure 5: Dot diagram of 16*16 Dadda multiplier

III. PROPOSED METHODS

3.1 Proposed Modified 7:2 Compressor

As the number of full adders increases compared to the number of inputs, the delay of the compressor will also increase. The 7:2 Compressor can be implemented in another way, i.e., by using the 4:2 Compressor, a full adder, and a half adder. To design a 7:2 Compressor, two 4:2 compressors, two full adders and one half-adder are required. As compared to the 7:2 compressor using full adders, the 7:2 compressor using the 4:2 compressor shows better performance in terms of delay and area. Therefore, the proposed work focuses on implementing the Dadda multiplier using a modified 7:2 compressor for improved electrical performance.

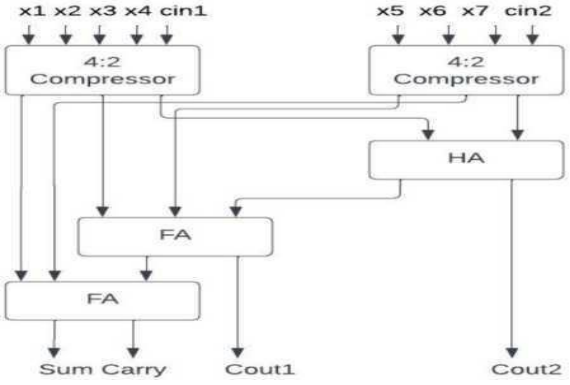


Figure 6 Design of 7:2 Compressor Using 4:2 Compressor

Figure 6 above shows the 7:2 compressor using a 4:2 compressor, which has inputs x1 to x7 and cin. The sum, carry, and cout1, cout2 outputs are generated between operations using a half adder.

Now, the comparisons of the modified 7:2 compressor with the other compressors are shown in Table 2 below.

Table 2 Comparison Table with modified compressor

Parameter	Using Full adders			Modified 7:2 compressor
	4:2 [8]	5:2 [9]	7:2 [10]	
Number of full adders used	2	3	5	2
Delay	1.065ns	1.295ns	1.895ns	1.849ns
Device Utilization	3%	5%	8%	6%

3.2 Proposed 16-bit Dadda multiplier using 4:2 compressor:

The following design describes how the suggested architecture uses significant driven logic compression to minimise delay and power consumption with a tiny error: The higher importance weights employ exact (4:2) compressors, the medium weights use approximate high-order compressors, and the lower weights use approximate (4:2) compressors that are erroneous. Circuits for PPM reduction have two stages. All of the weights are in the first phase. Only the higher significance weights are eligible for the second step. The maximum number of product terms for each weight after the second stage is two. Each lower significance weight has a maximum of two product phrases once the first stage is complete. We apply our approximation n:2 compressor for each middle significance weight to save energy, where n represents the number of product terms in this weight. The maximum height of the PPM is reduced using 4:2 compressors to achieve excellent precision.

The rightmost precise 4:2 compressor's carry bit, Cin, is configured to be 0. Each higher importance weight has two product words once the second stage is complete. Figure 7 below shows the schematic of the DADA multiplier using a 4:2 compressor extracted from Xilinx ISE software.

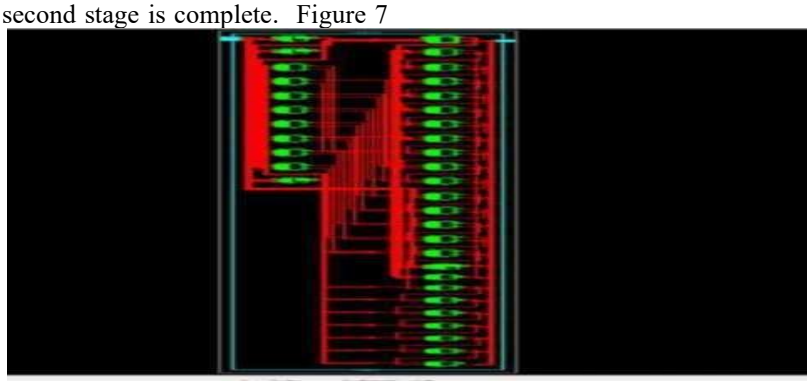


Figure 7 Schematic diagram of Dadda multiplier using 4:2 Compressor

The device utilization summary shown in Figure 8 here for understanding how many Slices, LUTs and IOBs are occupied.

```
Device utilization summary:
---------------------------

Selected Device : 3s100evq100-5

 Number of Slices:                        338  out of     960     35%
 Number of 4 input LUTs:                  589  out of    1920     30%
 Number of IOs:                            64
 Number of bonded IOBs:                    64  out of      66     96%

---------------------------
```

Figure 8: Device utilization summary

The timing details of the DDA multiplier are shown in Figure 9

```
Delay:               37.978ns (Levels of Logic = 32)
  Source:            m<0> (PAD)
  Destination:       y<30> (PAD)

  Data Path: m<0> to y<30>
                                        Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             26   1.106   1.223  m_0_IBUF (m_0_IBUF)
     LUT2:I0->O             2   0.612   0.410  p_3_and00001 (p<3>)
     LUT4:I2->O             3   0.612   0.454  c69/a1/Mxor_sum_Result1 (c69/x)
     LUT4:I3->O             2   0.612   0.532  c69/a2/carry1 (c<87>)
     LUT3:I0->O             2   0.612   0.410  f12/carry1 (c<117>)
     LUT3:I2->O             2   0.612   0.410  f13/carry1 (c<118>)
     LUT3:I2->O             2   0.612   0.410  f14/carry1 (c<119>)
     LUT3:I2->O             2   0.612   0.532  f15/carry1 (c<120>)
     LUT3:I0->O             2   0.612   0.410  f16/carry1 (c<121>)
     LUT3:I2->O             2   0.612   0.410  f17/carry1 (c<122>)
     LUT3:I2->O             2   0.612   0.410  f18/carry1 (c<123>)
     LUT3:I2->O             2   0.612   0.410  f19/carry1 (c<124>)
     LUT3:I2->O             2   0.612   0.410  f20/carry1 (c<125>)
     LUT3:I2->O             2   0.612   0.410  f21/carry1 (c<126>)
     LUT3:I2->O             2   0.612   0.410  f22/carry1 (c<127>)
     LUT3:I2->O             2   0.612   0.410  f23/carry1 (c<128>)
     LUT3:I2->O             2   0.612   0.532  f24/carry1 (c<129>)
     LUT3:I0->O             2   0.612   0.532  f25/carry1 (c<130>)
     LUT3:I0->O             2   0.612   0.532  f26/carry1 (c<131>)
     LUT3:I0->O             2   0.612   0.532  f27/carry1 (c<132>)
     LUT3:I0->O             2   0.612   0.532  f28/carry1 (c<133>)
     LUT3:I0->O             2   0.612   0.532  f29/carry1 (c<134>)
     LUT3:I0->O             2   0.612   0.532  f30/carry1 (c<135>)
     LUT3:I0->O             2   0.612   0.532  f31/carry1 (c<136>)
     LUT3:I0->O             2   0.612   0.532  f32/carry1 (c<137>)
     LUT3:I0->O             2   0.612   0.532  f33/carry1 (c<138>)
     LUT3:I0->O             2   0.612   0.532  f34/carry1 (c<139>)
     LUT3:I0->O             2   0.612   0.532  f35/carry1 (c<140>)
     LUT3:I0->O             2   0.612   0.532  f36/carry1 (c<141>)
     LUT3:I0->O             2   0.612   0.410  f37/carry1 (c<142>)
     LUT4:I2->O             1   0.612   0.357  f281/Mxor_sum_Result1 (y_30_OBUF)
     OBUF:I->O                  3.169          y_30_OBUF (y<30>)
    ----------------------------------------
    Total                      37.978ns (22.635ns logic, 15.343ns route)
                                        (59.6% logic, 40.4% route)
```

Figure 9: Timing details summary

3.3 16-bit Dadda Multiplier using 5:2 Compressor

Usually, a multiplier is made up of three parts. To create partial products in the first section, AND gates are used. Compressors are used in the second stage to lower the PPM maximum height (partial product matrix).

To obtain the final output, a carry-propagation adder is employed in the third step. The PPM reduction circuitry is therefore a key contributor to the multiplier's design complexity. The PPM reduction circuitry's optimization is the main emphasis of designs [1-6]. We present a 16 × 16 multiplier design in this section. According to their significance, weights are divided into three categories: higher significance weights, medium significance weights, and lower significance weights.

The number of higher significance weights, intermediate significance weights, and lower significance weights can all be adjusted by the designers to trade off between power consumption and computational accuracy. Our PPM reduction circuitry uses significance-driven logic compression, which consumes less power with a small amount of error:

While the lower weights employ inaccurate (5:2) area-efficient compressors, the higher weights use exact (5:2) compressors. The middle weights use two-stage approximate (5:2) compressors (OR-tree-based approximation). In the second and third stages, significance weights are taken into consideration. When the second and third phases are complete, there are no more than two product phases for each weight. As a result, a carry propagation adder may be used to create the final output, which is shown in Figure 10, using the Dadda multiplier with a 5:2 compressor.

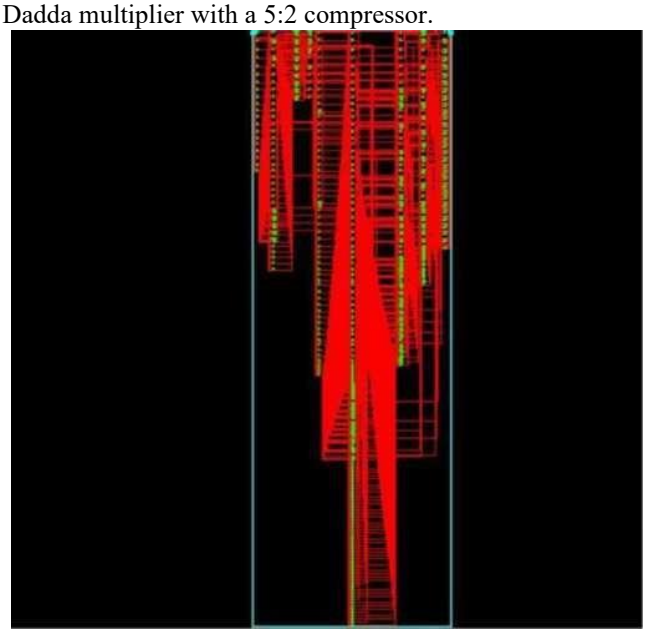


Figure 10 Schematic diagram of Dadda multiplier using 5:2 Compressor

Device utilization summary is given below in Figure 11

```
Device utilization summary:
---------------------------

Selected Device : 7z010clg400-3


Slice Logic Utilization:
 Number of Slice LUTs:                  416  out of  17600     2%
    Number used as Logic:               416  out of  17600     2%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    416
   Number with an unused Flip Flop:     416  out of    416   100%
   Number with an unused LUT:             0  out of    416     0%
   Number of fully used LUT-FF pairs:     0  out of    416     0%
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                          64
 Number of bonded IOBs:                  64  out of    100    64%

Specific Feature Utilization:
```

Figure 11: Device utilization summary

Timing Details: The timing details of the Dadda multiplier with a 5:2 compressor are shown here in Figure 12

```
Delay:               24.408ns (Levels of Logic = 20)
  Source:            a<2> (PAD)
  Destination:       y<31> (PAD)

  Data Path: a<2> to y<31>
                                   Gate     Net
    Cell:in->out      fanout       Delay    Delay  Logical Name (Net Name)
    ----------------------------------------------  ------------
     IBUF:I->O             32    1.222    1.656  a_2_IBUF (a_2_IBUF)
     LUT6:I0->O             1    0.203    0.944  cp52/m1/Mmux_y12 (cp52/m1/Mmux_y11)
     LUT6:I0->O             2    0.203    0.961  cp52/m1/Mmux_y13 (r4)
     LUT5:I0->O             2    0.203    0.981  ha7/c1 (x7)
     LUT6:I0->O             3    0.203    0.651  fa14/carry1 (L5)
     LUT5:I4->O             3    0.205    0.898  fa16/carry1 (L7)
     LUT6:I2->O             2    0.203    0.981  fa18/carry1 (L9)
     LUT6:I0->O             3    0.203    0.879  fa19/carry1 (L10)
     LUT6:I3->O             3    0.205    0.898  fa21/carry1 (L12)
     LUT6:I2->O             2    0.203    0.845  fa23/carry1 (L14)
     LUT4:I1->O             2    0.205    0.864  fa24/carry1 (L15)
     LUT5:I1->O             3    0.203    0.755  fa25/carry1 (L16)
     LUT5:I3->O             2    0.203    0.961  fa27/carry1 (L18)
     LUT6:I1->O             3    0.203    0.879  fa28/carry1 (L19)
     LUT6:I3->O             3    0.205    0.898  fa30/carry1 (L21)
     LUT6:I2->O             4    0.203    0.684  fa32/carry1 (L23)
     LUT6:I5->O             3    0.205    0.651  fa35/carry1 (L26)
     LUT5:I4->O             2    0.205    0.981  fa37/carry1 (L28)
     LUT6:I0->O             1    0.203    0.579  fa38/carry1 (y_31_OBUF)
     OBUF:I->O                   2.571           y_31_OBUF (y<31>)
    ----------------------------------------------
    Total                       24.408ns (7.459ns logic, 16.949ns route)
                                         (30.6% logic, 69.4% route)
```

Figure 12: Timing details of the 16-bit Dadda Multiplier using 5:2 Compressor

3.3 16-bit Dadda Multiplier using Modified 7:2 Compressor

The 7:2 compressor-based Dadda multiplier is a form of digital multiplier circuit used to multiply two binary values together. It is based on the Dadda algorithm, a powerful technique for multiplying binary numbers. The 7:2 compressor-based Dadda multiplier's fundamental idea is to divide the multiplication problem into smaller sub-problems, each of which can be resolved using a straightforward 7:2 compressor circuit. A larger compressor circuit receives the output from each compressor circuit and then mixes the outputs to produce the finished product.

The fundamental benefit of the Dadda multiplier based on the 7:2 compressor is that it uses fewer adders and partial products than other multipliers, like the Wallace tree multiplier.

To develop a 16-bit Dadda multiplier based on a 7:2 compressor, follow these steps:

- Step 1: The 16-bit multiplicand and multiplier are first divided into four groups of four bits each.
- Step 2: Multiply the appropriate group of four multiplicand bits to generate the partial products for each multiplier bit.
- Step 3: Organise the unfinished items into three rows of information: level 0, level 1, and level 2. Each level will have a different number of rows, and each row will contain a varied number of partial products.
- Step 4: Create the sum and carry signals for each row using the 7:2 compressor.
- Step 5: Propagate the carry signals through the rows to obtain the result, then combine the sum signals.

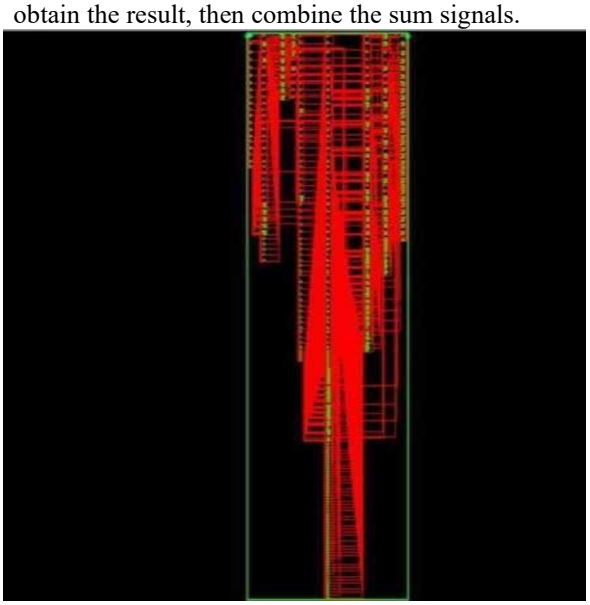


Figure 13: Schematic diagram of Dadda multiplier using 7:2 Compressor

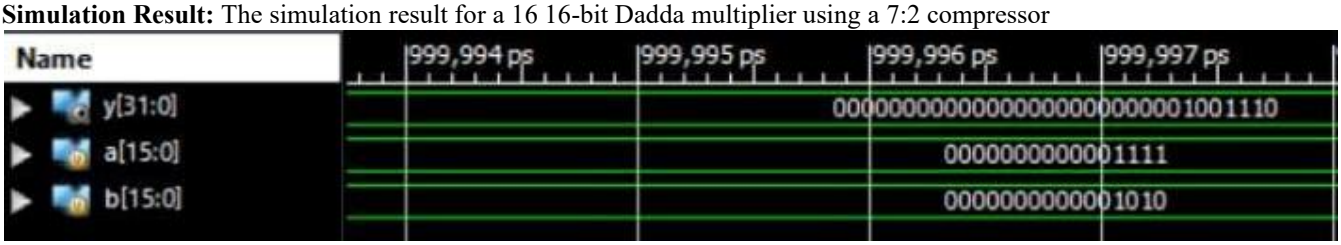
The above Figure 13 shows the RTL schematic diagram of the Dadda multiplier using a 7:2 compressor **Device.Utili.Station Summary: The figure below 14 shows the device utilisation summary, which includes the number of slices, LUTs, IOBs, etc., that are** occupied.

```
Delay:               10.549ns (Levels of Logic = 23)
  Source:            b<1> (PAD)
  Destination:       y<29> (PAD)

  Data Path: b<1> to y<29>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             32   0.000   0.563  b_1_IBUF (b_1_IBUF)
     LUT4:I0->O             1   0.043   0.428  cp41/m1/Mmux_y1_SW0 (N8)
     LUT5:I2->O             2   0.043   0.347  cp41/m1/Mmux_y1 (t2)
     LUT3:I1->O             2   0.043   0.554  fa11/carry1 (L2)
     LUT6:I0->O             2   0.043   0.555  fa12/carry1 (L3)
     LUT6:I0->O             3   0.043   0.438  fa13/carry1 (L4)
     LUT6:I3->O             2   0.043   0.555  fa15/carry1 (L6)
     LUT6:I0->O             3   0.043   0.466  fa16/carry1 (L7)
     LUT5:I1->O             2   0.043   0.432  fa18/carry1 (L9)
     LUT3:I0->O             2   0.043   0.293  fa19/carry1 (L10)
     LUT6:I5->O             3   0.043   0.438  fa20/carry1 (L11)
     LUT5:I2->O             3   0.043   0.438  fa22/carry1 (L13)
     LUT5:I2->O             3   0.043   0.438  fa24/carry1 (L15)
     LUT5:I2->O             3   0.043   0.438  fa26/carry1 (L17)
     LUT5:I2->O             2   0.043   0.293  fa28/carry1 (L19)
     LUT6:I5->O             2   0.043   0.348  fa29/carry1 (L20)
     LUT5:I3->O             2   0.043   0.555  fa30/carry1 (L21)
     LUT6:I0->O             2   0.043   0.461  fa31/carry1 (L22)
     LUT4:I0->O             3   0.043   0.466  fa32/carry1 (L23)
     LUT6:I2->O             3   0.043   0.299  fa34/carry1 (L25)
     LUT5:I4->O             3   0.043   0.560  fa36/carry1 (L27)
     LUT6:I0->O             1   0.043   0.279  fa37/Mxor_sum_xo<0>1 (y_29_OBUF)
     OBUF:I->O                  0.000          y_29_OBUF (y<29>)
    ----------------------------------------
    Total                      10.549ns (0.903ns logic, 9.646ns route)
                                        (8.6% logic, 91.4% route)


Device utilization summary:
---------------------------

Selected Device : 3s250eft256-5

 Number of Slices:                      339  out of   2448    13%
 Number of 4 input LUTs:                590  out of   4896    12%
 Number of IOs:                          64
 Number of bonded IOBs:                  64  out of    172    37%
```

Figure 14: Timing and device utilization details Summary

IV. RESULTS AND DISCUSSIONS

Simulation Result: The simulation result for a 16 16-bit Dadda multiplier using a 7:2 compressor



Now Table 3 shows the comparison table for the Dadda multiplier with the proposed method for 4:2, 5:2 and 7:2 proposed compressors

Table. 3. Comparison table for the Dadda multiplier of the proposed method

Parameter	16-bit Dadda Multiplier		
	4:2 [5]	5:2 [6]	7:2 Proposed
Delay	37.978ns	24.408ns	10.549ns
Device Utilization	96%	64%	37%

Application of Compressor-based Dadda Multiplier

Some possible applications of the 7:2 compressor-based Dadda multiplier include:

Digital signal processing (DSP): Multipliers are a key component in various DSP applications, including audio and video processing, speech recognition, and image processing. The 7:2 compressor-based Dadda multiplier can be used in these applications to perform high-speed multiplication operations. Cryptography: Cryptographic algorithms, such as RSA and elliptic curve cryptography, rely heavily on large integer multiplication. The 7:2 compressor-based Dadda multiplier can be used to efficiently implement these algorithms, thereby reducing the computation time required for encryption and decryption. Computer vision: Many computer vision algorithms, such as object recognition and tracking, require intensive matrix operations. The 7:2 compressor-based Dadda multiplier can be used to perform these operations efficiently, enabling real-time performance on embedded systems. Artificial intelligence (AI): Machine learning algorithms, such as neural networks and deep learning, require a large number of multiplication operations. The 7:2 compressor-based Dadda multiplier can be used in these applications to speed up the computation time needed for training and inference.

V. CONCLUSIONS

A new design of a 7:2 Compressor-based 16-bit Dadda multiplier is implemented in this paper. The proposed design is a flexible and effective circuit with a wide range of applications in artificial intelligence, computer vision, and digital signal processing. It is designed to reduce the area and power by reducing the number of adder units. According to the results, a significant improvement is achieved in terms of delay and area. The delay occurred in the multiplier while using a 7:2 compressor is less compared to a 4:2 compressor and a 5:2 compressor.

DECLARATION

Funding/ Grants/ Financial Support	No, we did not receive.
Conflicts of Interest/ Competing Interests	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence
Avail.abilit of Data and Material/ Data Access Statement	(ieeexplore.ieee.org) and scholor.google.com
Authors Contributions	I, M. Venkanna, have completed this paperwork under the guidance of Sri P. V. V. Satyanarayana, Sr. Assistant. Professor, Sri Vasavi Engineering College.

REFERENCES

1 Tianqi Kong and Shuguo Li," Design and Analysis of Approximate 4-2 Compressors for High Accuracy Multipliers." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 29, NO. 10 OCTOBER 2021. [CrossRef]
2 B. Sree Mangamma and M. Madhu Babu, "Design of Approximate Multiplier using 5:2 Compressors." International Journal of Reconfigurable and Embedded Systems, Volume 8, Issue no. 11, April 2022.
3 Pranose J. Edavoor, Sithara Raveendran, and Amol D. Rahulkar," Approximate Multiplier Design Using Novel Dual-Stage 4 X 2 Compressors", VOLUME 8, pp.48337_48551, Feb.2020. [CrossRef]
4 Manikantta Reddy, M. H. Vasantha, Y. B. N. Kumar, and D. Dwivedi, 'Design and analysis of multiplier using approximate 4-2 compressor,' AEU-Int. J. Electron. Commun. vol. 107, pp. 89_97, Jul. 2019. [CrossRef]
5 D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra," Approximate multipliers based on new approximate compressors," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 12, pp. 4169_4182, Dec. 2018. [CrossRef]
6 Gorantla and P. Deepa, 'Design of approximate compressors for multiplication,' ACM J. Emerg. Technol. Comput. Syst., vol. 13, no. 3, pp. 1_17Apr. 2017. [CrossRef]
7 K. Hari Kishore, K. Akhil, G. Viswanath, N. Pavan Kumar, "Design and Implementation of 8x8 Multiplier using 4-2 Compressor and 5-2 Compressor", International Journal of Reconfigurable and Embedded Systems (IJRES) Vol. 5, No. 3, November 2016. [CrossRef]
8 A. Momeni, J. Han, P. Montuschio, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984_994, Apr. 2015. [CrossRef]
9 Sanjeev Kumar, Manoj Kumar 4-2 Compressor design with New XOR-XNOR Module, 4th International Conference on Advanced Computing

10 C.H. Lin, J.C. Lin, "High accuracy approximate multiplier with error correction," IEEE 31st International Conference on Computer Design (ICCD), 2013
11 SATTI, VV SATYANARAYANA, and Sridevi Sriadibhatla. "Hybrid self-controlled precharge-free CAM design for low power and high performance." *Turkish Journal of Electrical Engineering and Computer Sciences* 27.2 (2019): 1132-1146. [CrossRef]
12 Satyanarayana, S. V. V., et al. "Dual-chirality GAA-CNTFET-based SCPF-TCAM cell design for low power and high performance." *Journal of Computational Electronics* 18 (2019): 1045-1054. [CrossRef]
13 Satti, VV Satyanarayana, and Sridevi Sriadibhatla. "Dual-bit control low-power dynamic content addressable memory design for IoT applications." *Turkish Journal of Electrical Engineering and Computer Sciences* 29.2 (2021): 1274-1283. [CrossRef]
14 Satyanarayana, Satti VV, and Sriadibhatla SRIDEVI. "Design of TCAM architecture for low power and high performance applications." *Gazi University Journal of Science* 32.1 (2019): 164-173.

AUTHORS PROFILE

PVV Satyanarayana, Sr. Assistant Professor in the Electronics & Communication Department at Sri Vasavi Engineering College (SVEC), Pedatadepalli, Tadepalligudem. He has a graduation and master's degree in VLSI &Embedded Systems. He also guides the students in their research and projects. He has a good knowledge of Digital arithmetic and floating-point architectures and has over 10 years of teaching experience. He has also mentored many undergraduate students to help them complete their final-year projects. He also focuses on the research of the practical application of VLSI, floating-point arithmetic, and computer arithmetic techniques. PVV Satyanarayana is a permanent member of the Institute of Electronics and Communications Engineering (IETE)

M. Venkanna is currently pursuing the Bachelor's Degree in Electronics & Communication Department at Sri Vasavi Engineering College (SVEC), Pedatadepalli, and Tadepalligudem, Andhra Pradesh. His research interest is in VLSI design. He has excellent practical knowledge with the ability to identify fine points of VLSI circuits and Design. He has a keen interest in programming languages such as Python, VHDL, and Verilog, seeking to bring technical expertise and an open-minded approach to the field of VLSI circuits. He is always eager to enhance his professional capabilities through learning. He actively participates in cultural activities and group discussions at the college.

S. Jayasri is a student of the Electronics & Communication Department at Sri Vasavi Engineering College (SVEC), Pedatadepalli, and Tadepalligudem. Andhra Pradesh. Her research interest is in VLSI design. She has excellent knowledge, with the ability to code for highly sophisticated problems in VLSI circuits and MOS circuit design. She has a keen interest in programming languages such as VHDL and Verilog, as well as expertise in Mentor Graphics and Vivado high-level synthesis tools. She is always eager to enhance his professional capabilities through learning and is also an active member in technical events and cultural activities.

B. Ramjee is currently pursuing the Bachelor's Degree in Electronics & Communication Department at Sri Vasavi Engineering College (SVEC), Pedatadepalli, and Tadepalligudem, Andhra Pradesh. His research interest is in VLSI design.. He is interested in programming languages like Python and C, seeking to bring technical expertise and an open-minded approach to the field of VLSI circuits. He is always eager to enhance his professional capabilities through learning. He actively participates in the cultural activities and group discussions in the college

J. Vasavi Kalyani is currently pursuing the Bachelor's Degree in Electronics & Communication Department at Sri Vasavi Engineering College (SVEC), Pedatadepalli, and Tadepalligudem, Andhra Pradesh. His research interest is in VLSI design. He is interested in bringing technical expertise and an open-minded approach to the field of VLSI circuits. He actively participates in cultural activities.