© Universiti Tun Hussein Onn Malaysia Publisher's Office

IJIE

http://penerbit.uthm.edu.my/ojs/index.php/ijie
ISSN : 2229-838X e-ISSN : 2600-7916

The International Journal of Integrated Engineering

Design of Complex Multiplier Using Vedic Mathematics

Hasliza Hassan[1]*, K. B. Hwa[2], S. I. M. Akhbal[3], M.F. Kambas[4], I. I. Jamaludin[5]

[1]Faculty of Engineering Techology,
Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor,84600, MALAYSIA

[2]Besgrade Plywood Sdn Bhd,
Batu 10, Jalan Pokok Sena, Alor Setar, Kedah, 06400, MALAYSIA

[3]Hosiden Electronics (M) Sdn. Bhd,
Bangi Industrial Estate, Bandar Baru Bangi, Selangor, 43650, MALAYSIA

[4]Department of Petrochemical Engineering,
Politeknik Tun Syed Nasir Syed Alwi, Hub Pendidikan Pagoh, Muar, Johor, 84600, MALAYSIA

[5]Intel Technology Sdn.Bhd.
Jalan Hi-Tech 2/3 Kulim Hi-Tech Park, Kulim, Kedah, 09000, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2023.15.03.021
Received 20 June 2023; Accepted 18 July 2023; Available online 2 October 2023

Abstract: In this project, a 4x4 multiplier is implemented that utilizes the Urdhava Tiryakbhyam sutra method in Vedic mathematics. This method is applicable in all two decimal number multiplications which offers high speed calculation and improved efficiency. Thus, the design of a 4x4 Vedic-based multiplier is solely aimed at performing faster multiplications and achieving quicker processing speeds than the traditional multipliers. The architecture of the Vedic multiplier consists of four 2x2 multipliers and three adders of different bit sizes that are assembled using the Wallace tree implementation. The coding for the multipliers and adders is written in Verilog Hardware Description Language (HDL) in the Quartus Prime 17 Software. Functional simulation is then carried out to ensure that the Vedic multiplier performs the accurate multiplication operations, while the Verilog Compiled Simulator is employed to compile and simulate the multiplier design. Following this, the Design Compiler (DC) and Integrated Circuit Compiler (ICC) command scripts are then composed to allow the logic and physical synthesis to be performed on the Vedic and traditional multipliers. From there, the performance level of both these multipliers are assessed through reference to several key parameters such as timing, area, power consumption, overflow percentage and congestion statistics. Based on the results obtained in the synthesis process, the Vedic multiplier possesses faster operational speed than the traditional multiplier (due to a shorter processing time), but ultimately exhibits a greater power consumption and wider area coverage.

Keywords: Vedic mathematics, multiplier, faster operational, speed.

*Corresponding author: hliza@uthm.edu.my

1. Introduction

The method of Vedic Mathematics formula Urdhva Tiryagbhyam Sutra is one of the methods of 16 Sutra its means vertical and crosswise calculation. In ancient time in India, people used this Sutra for decimal number mathematical functions effectively. They apply even to complex problems involving many mathematical operations. The Urdhava Tiryakbhyam method is one of the methods of 16 Sutra its means vertical and crosswise calculation, is used for two decimal number multiplication. The meaning of this sutra is "Vertically and crosswise" and it is applicable to all the multiplication operation [1]. This sutra is general formula to all applicable of multiplication operation with greatest advantage in speed as compared to other multipliers. The Urdhva Tiryagbhyam sutra under Vedic mathematics is the general formula applicable to all cases of multiplication [2]. This Sutra was traditionally used in ancient India for the multiplication of two decimal numbers in relatively less time [3]. The speed of a processor determines its performance [4,5,6,7,8]. High speed processing is an essential requirement for all the systems. Multiplication is a significant operation in Digital signal processors and Arithmetic-Logic Unit (ALU), and thus the demand for high-speed multiplication is continuously increasing in modern Very Large-Scale Integration (VLSI) design [12,13,18,19]. In addition, research by [20], In comparison to the Vedic and Array multipliers, the complex floating-point multiplication with single precision using the combined integer and floating point multiplier has a smaller amount of delay and uses less power. This is advantageous for quick multiplication operations that can be used for DSP applications. However, the focus of this project is on high-speed multiplication for better performance using Vedic Mathematics.

2. Mathematical Formulation

The method that used for this project is Vedic mathematics implantation [9,14,15, 16,17]. This method faster in term of calculation process, time and speed. The comparison will carry out the differences between the Vedic multiplier and traditional multiplier to investigate the speed, area, time and efficiency. Quartus software used to design the Verilog code of adder and multiplier while the Synopsys tools used to run the simulation with test bench. The design compiler part will set the timing analysis and convert the Register Transfer Level (RTL) to gate-level netlist. Then, IC Compiler (ICC) will carry out the physical synthesis. For the last part, the design of performance will be analyzed and check the output whether tally with input. After all the processes are completed, the performance gap between the Vedic multiplier and traditional multiplier is analyzed. The Application Specific Integrated Circuit (ASIC) design methodology will be justified for the following topic.

Fig. 1 shows an example of the basic 4x4 bit multiplication by using multiple adders and logic gates [4,5,6,7,8,9,10,11]. The process of multiplication is based on addition, and many of the techniques useful in addition carry over to multiplication [4,5]. The multiplication of Y0 and X0 which will produce a partial product Y0X0 that can be gained by using an AND gate. Others multiplication also using AND gates. For example, the values at the third until sixth row are sum multiplication of the values of X and Y. The values P7, P6, P5, P4, P3, P2, P1 and P0 are final answer from the multiplication process at Fig. 1. Lastly, all the multiplication results from third until sixth row will used addition to sum up by using OR gate. The result is added to the next partial product with carry out and it will come out a final partial product, lastly it will produce 8-bit sum which indicates the multiplication value of the two binary numbers. This type of multiplication takes time and not efficiency.

$$
\begin{array}{cccccccc}
 & & & & Y = Y_3 & Y_2 & Y_1 & Y_0 \\
 & & & & X = X_3 & X_2 & X_1 & X_0 \\
\hline
 & & & & Y_3X_0 & Y_2X_0 & Y_1X_0 & Y_0X_0 \\
 & & & Y_3X_1 & Y_2X_1 & Y_1X_1 & Y_0X_1 & \\
 & & Y_3X_2 & Y_2X_2 & Y_1X_2 & Y_0X_2 & & \\
 & Y_3X_3 & Y_2X_3 & Y_1X_2 & Y_0X_2 & & & \\
\hline
\boldsymbol{P_7} & \boldsymbol{P_6} & \boldsymbol{P_5} & \boldsymbol{P_4} & \boldsymbol{P_3} & \boldsymbol{P_2} & \boldsymbol{P_1} & \boldsymbol{P_0}
\end{array}
$$

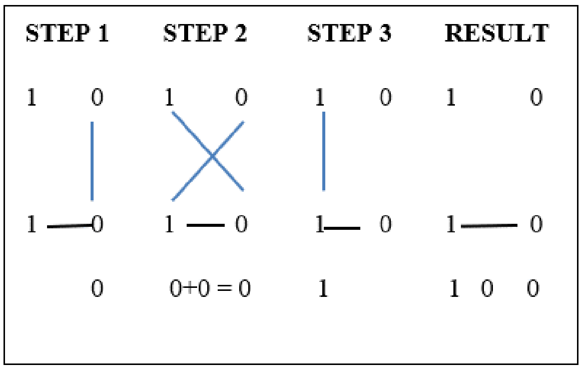


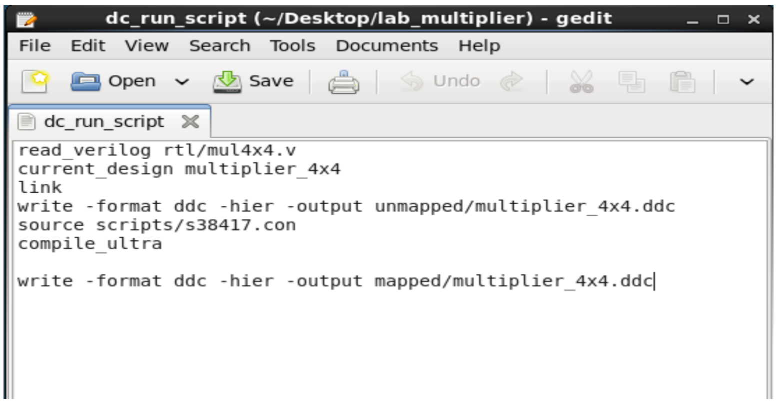
Fig. 1 - Multiplication using Vedic mathematics

The main method that used for this project is Vedic mathematics. This method faster in term of calculation process, time, and speed. The comparison will carry out the differences between the Vedic multiplier and traditional multiplier to investigate the speed, area, time and efficiency. The design of performance analysed and check the output whether tally with input. After all the processes are completed, the performance gap between the Vedic multiplier and traditional multiplier is analysed.

3. Design Method

Quartus software is used to design the Verilog code of adder and multiplier while the Synopsys tools used to run the simulation with test bench. Then for design compiler part will set the timing analysis and convert the Register Transfer Level (RTL) to gate-level netlist. Then, IC Compiler (ICC) will carry out the physical synthesis. For the last part, the design of performance will be analyzed and check the output whether tally with input. After all the processes are completed, the performance gap between the Vedic multiplier and traditional multiplier is analyzed. First and foremost, the read_verilog rtl/mul4x4.v command enables the Synopsys to load the default and specified libraries stored within its software memory. It is also capable of reading all Raster Transfer Language (RTL) files and translating them into a technology-independent design (GTECH) in a single step. Furthermore, it loads the unmapped ddc design in terms of DC memory. Once this is done, the complete design of the mul4x4.v can be properly read by the software which sets the Vedic 4x4 multiplier (mul4x4.v) as the current design. This setting is further reaffirmed in the following line where the current_design multiplier_4x4 command is used to specify the current design of the Vedic multiplier. Subsequently, the link command enables the synthesised design to be checked for its consistency via a separate check_design command. This check is also important in identifying the presence of any unresolved subdesigns (components) that have been referenced more than once (unresolved) or whether the hierarchy is recursive. It also reports all multiply instantiated designs together with their instance names and associated attributes. Therefore, it can prevent the design from facing any connectivity and hierarchy issues, for example missing ports, unconnected input pins, recursive hierarchy or multiple instantiations.

Logic synthesis is conducted under the DC command script in order to gain an insight on the parametric performances of the 4x4 Vedic and traditional multiplier. The full script is shown in Fig. 2 which encompasses the necessary steps required to run the DC command smoothly. This script is applicable in both multiplier circuits.


dc_run_script (~/Desktop/lab_multiplier) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
dc_run_script

```
read_verilog rtl/mul4x4.v
current_design multiplier_4x4
link
write -format ddc -hier -output unmapped/multiplier_4x4.ddc
source scripts/s38417.con
compile_ultra

write -format ddc -hier -output mapped/multiplier_4x4.ddc
```

Fig. 2 - DC command script of the Vedic and traditional multipliers

Next, the write -format ddc -hier -output unmapped/multiplier_4x4.ddc command allows all ddc format files to store the design netlist, constraints and attributes. This is an efficient format to either re-read the design into DC or to read the design into ICC which will be used later in the project. The source scripts/s38417.con in the following line enables the software to source the scripts of the constraints file named s38417.con, which is the Vedic multiplier file After that, the compile_ultra command is used as an optimization feature which performs three main levels of optimization. These levels include architecture level or high-level synthesis, logic level or GTECH optimization, as well as gate-level or mapping optimization. In addition, the optimization process aims to minimize the area coverage of the designed circuit while attempting to meet timing, power and Design Rule Check (DRC) constraints. The main difference between the compile and compile_ultra command is the inclusion of an additional high-performance optimization algorithm in the latter command. Last but not least, the write -format ddc -hier -output mapped/multiplier_4x4.ddc command has similar functions to its unmapped counterpart where it is able to read the mapped 4x4 Vedic multiplier design into DC or ICC using a more effective and efficient format.

The Integrated Circuit Compiler (ICC) or IC Compiler is another vital tool in the Synopsys software. It is a physical tool that can perform many functions. These functions include receiving a gate-level netlist from the DC, generating the required floorplans, timing constraints, and physical libraries as inputs. In the IC Compiler, the data setup is initially executed followed by the design planning. The floorplan can then be viewed clearly through the enlargement of its layout. Next, the placement will be set to perform iterative placement and logic optimization. The objective of this step is to fix timing violations and congestion. Furthermore, clock tree synthesis is run to fix and hold time violations and perform inter-clock balancing. The routing step is also important as it can concurrently perform logic, placement, and routing optimizations. The main objective of this process is to complete routing and meet the timing requirements. All of the mentioned steps can be performed via the insert command to enable the IC Compiler to run these essential steps. Moreover, the functions of the IC Compiler also include power optimization, concurrent timing analysis, as well as area, timing and power reduction. In addition, adjustments can be made to the connection paths within the floorplan to allow the current to flow smoothly through all the paths while greatly reducing the area coverage.

In this project, the physical synthesis executed under the ICC command script allows the parametric performances of both the 4x4 Vedic and traditional multiplier to be viewed which is similar to that of the DC command. However, an additional perk offered by the ICC script is the display of the overall design floorplan and global route congestion map of these multipliers modes. In manual mode, the power and ground nets have to be specified by the programmer. Whereas the automatic mode derives the power and ground nets from the power domain connections. Automatic mode can only be implemented in multi-voltage designs with Unified Power Format (UPF) descriptions and requires logic libraries with power and ground pins. The next command creates a floorplan in a sequential order around the circuit with a chip boundary, core, rows, and wire tracks. The size of the floorplan is slightly varied between the two multipliers, with 15 and 20 measurement units being allocated to the Vedic and traditional multiplier respectively.

All in all, the DC and ICC command scripts are essential programs that are needed to run the logic and physical synthesis in order for the parametric performances of the Vedic and traditional multipliers to be reported in a clear and concise manner. From there, the researcher is able to gain an insight on the performance levels of both multiplier circuits in terms of several key parameters such as timing, area and power consumption. The results from the synthesis process will serve as an accurate gauge of the performance standards of both multipliers in regards to their area, power, timing, overflow percentage and congestion statistics.

4. Result and Discussion

From the functional simulation results, two samples of simulation results are taken to compare and check either correct on not. The waveform has two input values that are randomly set as well as an output value. After the input is inserted, the functional simulation is executed. The output waveform will be displayed after the multiplication of the two input values is carried out. Fig. 3 shows the output results of multiplier circuit using Vedic technique.

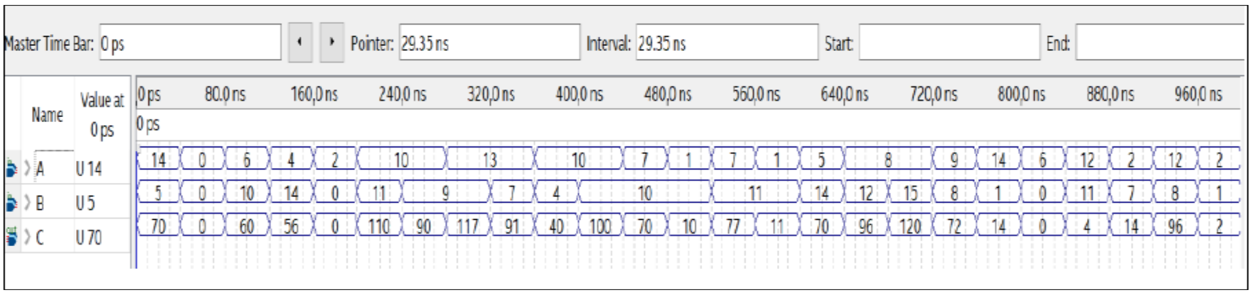

Master Time Bar: 0 ps
Pointer: 29.35 ns
Interval: 29.35 ns
Start:
End:
Name
Value at 0 ps
0 ps 80.0 ns 160.0 ns 240.0 ns 320.0 ns 400.0 ns 480.0 ns 560.0 ns 640.0 ns 720.0 ns 800.0 ns 880.0 ns 960.0 ns
0 ps
A U14 14 0 6 4 2 10 13 10 7 1 7 1 5 8 9 14 6 12 2 12 2
B U5 5 0 10 14 0 11 9 7 4 10 11 14 12 15 8 1 0 11 7 8 1
C U70 70 0 60 56 0 110 90 117 91 40 100 70 10 77 11 70 96 120 72 14 0 4 14 96 2

Fig. 3 - Output waveform of the functional simulation

The results of the logic and physical synthesis are carried out under the DC and ICC command script enables to measure the overall performance of the Vedic Multiplier in terms of important performance indicators such as the timing (and speed), maximum area, total power consumption, total number of cells, as well as the design floorplan and overflow. This assessment is carried out in order to evaluate the efficacy of the proposed multiplier circuit as well as the pros and cons that are inherent within the circuit design.

Following this, the results of the physical synthesis enable the researcher to analyse the overall design floorplan of the multiplier circuit. This step is essential in ensuring that the placement of all the cells is optimally feasible in terms of minimal cell congestion. Fig. 4 illustrates the complete design floorplan of the circuit, with the global routing for congestion analysis performed to produce the congestion map shown in Fig. 5. By default, a medium-effort global routing is carried out under the 'routing_stage global' command, which generates the congestion map based on the specified routing stage. The coloured boxes in the bottom portion of the map represent the different types of metals used in the floorplan layout in Fig. 5, namely METAL1, METAL2, etc. The numbers on the right side of the boxes correspond to the overflow percentage of each individual metal. Based on these figures, it can be noted that there is null percentage of overflow (and thus congestion) from METAL3 to METAL7, while only a minimum overflow percentage is detected from METAL0 to METAL2, all of which are within the 2% tolerated range. Therefore, the design floorplan of the Vedic multiplier circuit is considered feasible as there is little or no overflows present within the layout.

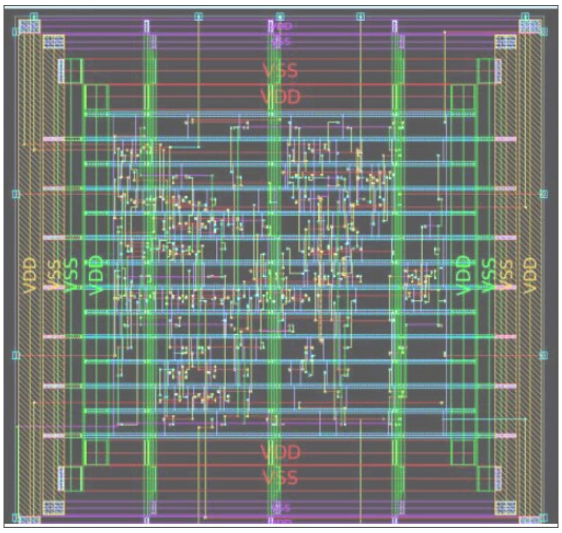
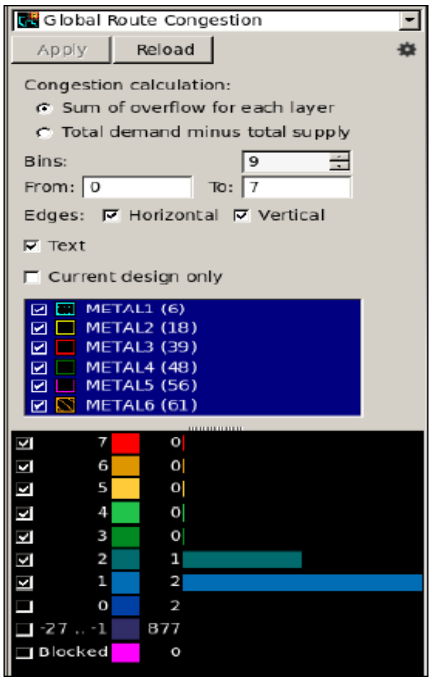


Fig. 4 - Design floorplan of the Vedic multiplier circuit



Congestion calculation:
[✓ Sum of overflow for each layer]
Total demand minus total supply

Bins: 9
From: 0 To: 7
Edges: ☑ Horizontal ☑ Vertical
☑ Text
☐ Current design only

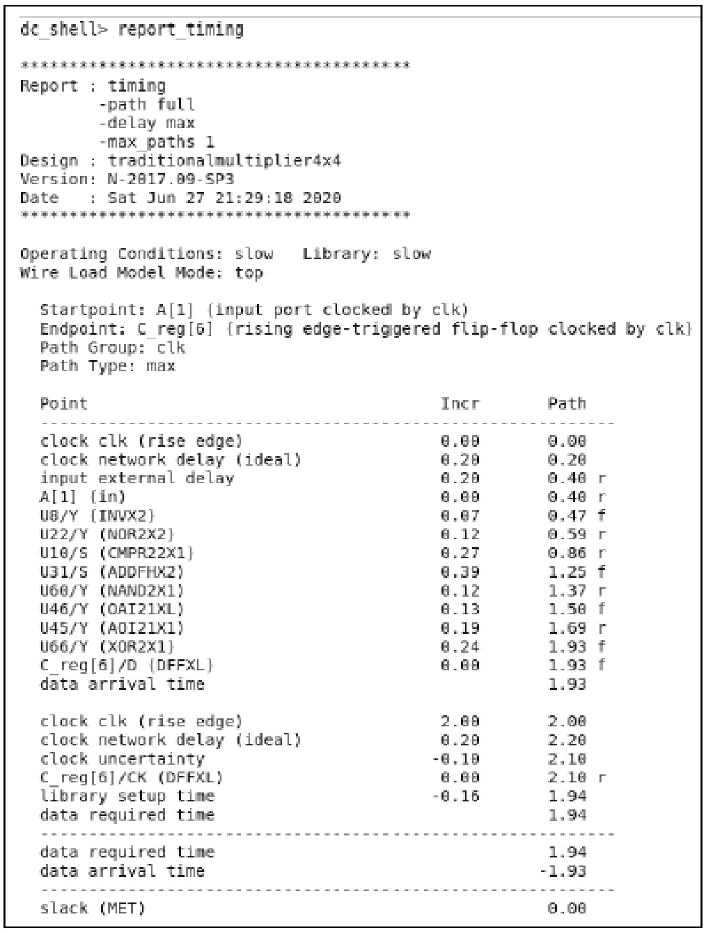
☑ METAL1 (6)
☑ METAL2 (18)
☑ METAL3 (39)
☑ METAL4 (48)
☑ METAL5 (56)
☑ METAL6 (61)

☑	7	🟥	0
☑	6	🟧	0
☑	5	🟨	0
☑	4	🟩	0
☑	3	🟩	0
☑	2	🟦	1
☑	1	🟦	2
☐	0	🟦	2
☐	-27 .. -1	🟪	877
☐	Blocked	🟪	0

Fig. 5 - Global route congestion map of the Vedic multiplier circuit

Furthermore, it is important to note that the overflow value is defined as the total number of wires in the global routing cells that do not have a corresponding accessible track, i.e. the wires that are over utilized. Therefore, a global routing cell is deemed to be overly congested should the actual number of wires that pass through the cell exceed the number of tracks available in that same routing cell. Whereas the GRCs value corresponds to the total number of overly congested global routing cells in the multiplier design. Based on the congestion report of the proposed multiplier circuit, both the overflow and GRCs values are zero which means that the placement of routing cells within the design is feasible and no congestion is present. Therefore, the proposed Vedic multiplier circuit that employs a 4x4 layout concept is an optimum design that features no overflow and no congestion properties.

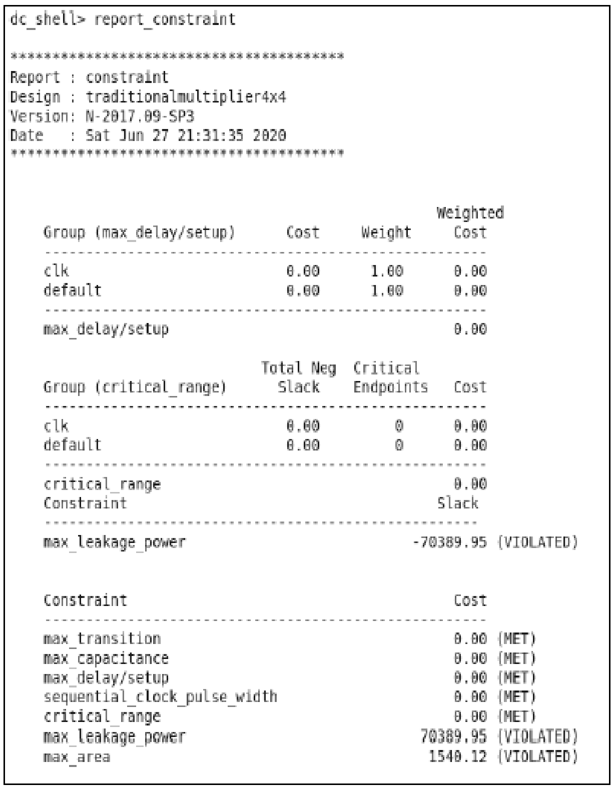
An identical simulation to that of the Vedic multiplier is used to evaluate the overall timing required by the traditional multiplier to complete the tasks given under the same circumstances. The reported timing results in Fig 6 show that the circuit needs a total of 1.93 units of time for accurate data arrival to be obtained. The breakdown of the processes is required to complete the whole multiplication operation as well as the time consumed by each individual process are also provided in the figure.



```
dc_shell> report_timing

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : traditionalmultiplier4x4
Version: N-2017.09-SP3
Date   : Sat Jun 27 21:29:18 2020
****************************************

Operating Conditions: slow   Library: slow
Wire Load Model Mode: top

  Startpoint: A[1] (input port clocked by clk)
  Endpoint: C_reg[6] (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  ----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.20       0.20
  input external delay                     0.20       0.40 r
  A[1] (in)                                0.00       0.40 r
  U8/Y (INVX2)                             0.07       0.47 f
  U22/Y (NOR2X2)                           0.12       0.59 r
  U10/S (CMPR22X1)                         0.27       0.86 r
  U31/S (ADDFHX2)                          0.39       1.25 f
  U60/Y (NAND2X1)                          0.12       1.37 r
  U46/Y (OAI21XL)                          0.13       1.50 f
  U45/Y (AOI21X1)                          0.19       1.69 r
  U66/Y (XOR2X1)                           0.24       1.93 f
  C_reg[6]/D (DFFXL)                       0.00       1.93 f
  data arrival time                                   1.93

  clock clk (rise edge)                    2.00       2.00
  clock network delay (ideal)              0.20       2.20
  clock uncertainty                       -0.10       2.10
  C_reg[6]/CK (DFFXL)                      0.00       2.10 r
  library setup time                      -0.16       1.94
  data required time                                  1.94
  ----------------------------------------------------------
  data required time                                  1.94
  data arrival time                                  -1.93
  ----------------------------------------------------------
  slack (MET)                                         0.00
```

Fig. 6 - Data arrival time of the traditional multiplier and breakdown of time for each individual process

The total area coverage of the traditional circuit that employs a 4x4 layout concept is similar to that of the Vedic multiplier is shown in Fig. 7. The maximum area constituted by this circuit is 1540.12 squared units which also corresponds to the total number of cells used in the design. The total cell utilization of the traditional multiplier as well as their individual areas are discussed in the following.



```
dc_shell> report_constraint

****************************************
Report : constraint
Design : traditionalmultiplier4x4
Version: N-2017.09-SP3
Date   : Sat Jun 27 21:31:35 2020
****************************************


                                                Weighted
    Group (max_delay/setup)      Cost     Weight     Cost
    -----------------------------------------------------
    clk                          0.00      1.00      0.00
    default                      0.00      1.00      0.00
    -----------------------------------------------------
    max_delay/setup                                  0.00

                              Total Neg  Critical
    Group (critical_range)      Slack    Endpoints   Cost
    -----------------------------------------------------
    clk                          0.00        0       0.00
    default                      0.00        0       0.00
    -----------------------------------------------------
    critical_range                                   0.00
    Constraint                                      Slack
    ----------------------------------------------------
    max_leakage_power                           -70389.95 (VIOLATED)


    Constraint                                       Cost
    -----------------------------------------------------
    max_transition                                   0.00 (MET)
    max_capacitance                                  0.00 (MET)
    max_delay/setup                                  0.00 (MET)
    sequential_clock_pulse_width                     0.00 (MET)
    critical_range                                   0.00 (MET)
    max_leakage_power                            70389.95 (VIOLATED)
    max_area                                      1540.12 (VIOLATED)
```

Fig. 7 - Total area covered by the traditional multiplier circuit

The total power consumption of all the circuit components within the traditional multiplier, where the internal power consumption is 0.4137 mW and the aggregate power consumption of the whole circuit is 0.4643 mW after the switching and leakage power is taken into account. Based on the information provided, it can be noted that the bulk of the power consumption is utilized by the register, whereby it consumes a 0.3195 mW or 68.81% of the total power consumption. Whereas the remaining 0.1448 mW or 31.19% of power is expended by the combinational circuit.

Nevertheless, it is important to note that a trade-off exists between the circuit parameters, where an advancement made to a certain parameter would inevitably compromise the performance of another parameter within the same circuit. This aspect is particularly evident in the Vedic multiplier, whereby a shorter data arrival time is achieved through the utilization of an additional number of cells within the circuit as a wider array of components, enable more tasks to be carried out simultaneously, which in turn allows the multiplication operations to be completed over a shorter period of time. A greater number of cells that are built into the multiplier will consequently increase the area coverage and power consumption of the whole circuit. Therefore, the trade-off between speed, power and area is a relatively conspicuous attribute of the proposed Vedic multiplier. As the objectives of this project is mainly focused on enhancing the speed and efficiency of the Vedic multiplier, a greater emphasis has to be placed on reducing the data arrival time in order to increase the relative speed of the proposed multiplier circuit. At the same time, an optimum balance has to be achieved between the parametric trade-offs to ensure that an adjustment made to improve the efficiency of the multiplier does not cause an exponential increase in the area coverage or power consumption that is somewhat unrealistic or unachievable. This argument also supported by [21,22], the processor's performance has improved and delays have decreased using Vedic multiplier in a few ways. The complexity of the total unit is substantially lower than that of a traditional multiplier.

After all, it can be reasonably concluded that the Vedic multiplier has achieved a higher operational speed as compared to the traditional multiplier, albeit with many cells, area coverage and power consumption. Thus, both these multipliers have their own advantages and drawbacks, which are useful for different circuit designs depending on the parametric attributes that are taken into consideration as the main priority for achieving the objectives. In [21], comparison to other multipliers, the 8 x 8 Vedic multiplier using the Brent Kung parallel prefix adder (BKA) decreases complexity, latency, delay, and area required while increasing system speed.

5. Conclusion

The functionality of the proposed Vedic multiplier was also tested through the execution of the functional simulation in the Quartus software which allowed checking is done on the accuracy of the multiplier circuit operation. Once the functionality of the Vedic multiplier has achieved acceptable standards, the performance level of the circuit was then assessed in terms of a few crucial parameters that were derived from the logic and physical synthesis process. This step is important in identifying the strengths and weaknesses of the Vedic multiplier circuit as well as its performance standards in comparison to that of the traditional multiplier. Based on the analysis carried out on the performance of the Vedic multiplier, it can be concluded that the main advantage of this circuit is its speed and efficiency as it is able to carry out the required tasks and generate an accurate output data waveform in a relatively short period of time. However, the inherent trade-offs between speed, power and area mean that the Vedic multiplier circuit ultimately consumes a greater amount of power and covers a wider extent of area.

Acknowledgement

The authors wish to express their gratitude to Faculty of Engineering Technology, University Tun Hussein Onn Malaysia for its support.

References

[1] D. K. Kahar and H. Mehta, "High speed vedic multiplier used vedic mathematics," Proc. 2017 Int. Conf. Intell. Comput. Control Syst. ICICCS 2017, vol. 2018-Janua, pp. 356–359, 2018.

[2] P. Mehta and D. Gawali, "Conventional versus Vedic mathematical method for hardware implementation of a multiplier," ACT 2009 - Int. Conf. Adv. Comput. Control Telecommun. Technol., pp. 640–642, 2009.

[3] P. M, S. K. Patil, Shivukumar, S. K. P, and S. H, "Implementation of Multiplier using Vedic Algorithm," Int. J. Innov. Technol. Explor. Eng., vol. 2, no. 6, pp. 219–223, 2013.

[4] E. Jaya and K. C. Rao, "Power , Area and Delay Comparision of Different Multipliers," vol. 5, no. 6, pp. 2093–2100, 2016.

[5] L. S. Jie and S. H. Ruslan, "A 2x2 bit Vedic multiplier with different adders in 90nm CMOS technology," AIP Conf. Proc., vol. 1883, no. September 2017, 2017.

[6] M. . QI LIU, B.S, B.A, M.E, "Design of multiplier and its VLSI implementation," vol. 65, no. 8, pp. 1141–1146, 1999.

[7] J. V. Suman, D. G. Jignash, and B. I. Neelgar, "Design of Digital FIR Filter Based on MCMAT for 12 bit ALU using DADDA & WALLACE Tree Multiplier," Int. J. Hybrid Inf. Technol., vol. 7, no. 6, pp. 325–336, 2014.

[8] S. Kumar and Soniya, "Multiplier-Accumulator Unit .," Int. J. Emerg. Trends Technol. Comput. Sci., vol. 2, no. 4, 2013.

[9] M. H. Kumar and M. Ramana Reddy, "High Speed 4x4 Bit Vedic Multiplier based on Vertical and Crosswise Methods," Int. J. Sci. Enginnering Technol. Res., vol. 03, no. 03, pp. 408–412, 2014.

[10] A. C. Swathi, T. Yuvraj, J. Praveen, and R. R. A, "A Proposed Wallace Tree Multiplier Using Full Adder and Half Adder," vol. 4, no. 5, pp. 472–474, 2016.

[11] R. Kumaravel, "Design of Radix-8 Mbe-Multiplier Based on Efficient Parallel Multiplier Accumulator," no. December, 2018.

[12] D. Chandel, G. Kumawat, P. Lahoty, V. V. Chandrodaya, and S. Sharma, "Booth Multiplier : Ease of multiplication," vol. 3, no. 3, pp. 2–6, 2013.

[13] D. Bordiya and L. Bandil, "Comparative Analysis of Multipliers (serial and parallel with radix based on booth algoritham)," Int. J. Eng. Res. Technol., vol. 2, no. 9, pp. 1437–1441, 2013.

[14] S. Shembalkar, "Vedic Mathematics Sutras -A Review," vol. 1, no. 21, 2017.

[15] M. S. N. Gadakh and A. S. Khade, "FPGA implementation of high speed vedic multiplier," IET Conf. Publ., vol. 2016, no. CP700, pp. 184–187, 2016.

[16] W. B. Vasantha, K. F. Smarandache, and W. B. Vasantha Kandasamy, "VEDIC MATHEMATICS - 'VEDIC' OR 'MATHEMATICS': A FUZZY &amp; NEUTROSOPHIC ANALYSIS INTRODUCTION TO VEDIC MATHEMATICS," 2006.

[17] S. Koushighan, K. Hariharan, and V. Vaithiyanathan, "Design of an optimized high speed multiplier using vedic mathematics," Contemp. Eng. Sci., vol. 7, no. 9–12, pp. 443–448, 2014.

[18] D. K. Tala, "Verilog Tutorial Updated," pp. 1–227, 2013.

[19] S. Code, E. Marks, E. Hours, and L. Hours, "Verilog HDL Verilog HDL [As per Choice Based Credit System (CBCS) scheme]," pp. 1–114.

[20] R. Rathod, P. Ramesh, P. S. Zele and A. K. Y, "Implementation of 32-Bit Complex Floating Point Multiplier Using Vedic Multiplier, Array Multiplier and Combined integer and floating point Multiplier (CIFM)," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bangluru, India, 2020, pp. 1-5, doi: 10.1109/INOCON50539.2020.9298363.

[21] A. Sai Kumar, U. Siddhesh, N. Sai kiran and K. Bhavitha, "Design of High Speed 8-bit Vedic Multiplier using Brent Kung Adders," 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2022, pp. 1-5, doi: 10.1109/ICCCNT54827.2022.9984591.

[22] A. Awasthy, "VLSI Implementation of Multiplier and Adder Circuits with Vedic Algorithm Computation," 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichy, India, 2022, pp. 1-4, doi: 10.1109/ICEEICT53079.2022.9768534.