International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-5, March 2019

Optimal Design Approach to Multiplier Unit Using Adaptive Logical Counters

Mohammad Abdul Naveed, R.P. Singh

Abstract: ***Multipliers are the primal constituents of a processing unit. The process of multiplication is carried out using different suggested architectures. Among the developments the common factor observed is the use of recursive register interface in multiplication operation. multiple registers were used in count to buffer the temporary data and gives a new result on addition of these register values. It is needed to minimize the resource requirement to enhance the objective of optimal multiplication operation, in this paper a new low resource adaptive counter based multiplier design is proposed, which minimizes the register requirement by deriving a sub counter logic in multiplier design.***

Key Words: Multiplier design, adaptive logical count, low resource overhead.

I. INTRODUCTION

To achieve the objective of increasing demand of high level services on compact devices, new devices were developed and, are in further development to meet the required demand. The raise in these demand has lead to developing of high processing units, performing arithmetic and logic operation at a very high speed to cope the processing demand. As new advanced interfacing units were developed, the processing need to be synchronized to give better operational efficiency. In the design of processing unit, the core processing block operating on input data is the arithmetic and logical unit (ALU). These units are responsible for performing all type of arithmetic and logical instructions, on the input data and provide the derived results. In this ALU unit, the constituent operations are arithmetic addition, subtraction, division and multiplication. The adder and subtractor unit perform the addition and subtraction operation, whereas a shifter and adder realizes a multiplier unit. Among these operational units, multiplier units are observed to be recurrent processing and requires large resources for temporary storage and addition process. This large demand of resource results in higher area coverage and power dissipation. This effect in operation efficiency of the processing unit. Digital design has given a big advantage in various real-time applications wrt. the accuracy and speed of operation. The need of high-speed operation and increasing design complexity is extending the design tools, and methodology to outer extent. Silicon Technology's are developing towards large gate density and high clock processing in digital system design. Design Engineers have used these advancements to synchronize more complex functionality in system-on-chip designs.

Manuscript published on 30 March 2019.
*Correspondence Author(s)
Mohammad Abdul Naveed, Research Scholar, Sri Satya Sai University of Technology and Medical Science, Bhopal, Madhya Pradesh
Dr. R. P. Singh, Research Guide, Sri Satya Sai University of Technology and Medical Science, Bhopal, Madhya Pradesh

In digital processing of real time applications, adding digital signal processor (DSP), adequate operations in different digital systems such as architecture, microprocessor, microcontroller and data process unit got added [1]. In the operation of digital computing, multiplier are the most commonly used unit. Multipliers are used as a main functional block in most of the digital signal processing applications. In many a applications researchers have used different approach s of addition [2-4] in optimizing the multiplication operation. various approaches of multiplication is developed wrt. binary adder architecture. A ripple adder and the carry-look-ahead adder were used as a conditional sum adder in multiplication operation. A VHDL definition for the design and comparative studies of multiplier is outlined in [5]. The design works on the area and delay of a unit-gate model. A full accuracy in bit-serial multiplier was outlined in [6]. A precision scheme for the 2's complement is presented in [7]. This n-bit functionality requires 2n clocks and 2n of five input adder to perform a multiplication operation. A serial multiplier, and a non-latency cycle was outlined in [8]. A programs including serial square and serial / serial multiplier is presented in [9]. A two symbol multiplication was applied which is applicable to both signs [10]. This process defines the multiplication of sign number in both primary and sign notation by uniform processing. Multiplication has two basic operations of generating partial product and its accumulation. The baugh-wooley multiplier [11,12] is used for multiplication of unsigned and signed numbers. The serial accumulation of signed binary numbers by high speed 1's counters has been suggested in [13]. The multiplicity process consists of three stages, production of partial products (PPs), reducing the PPs, and a final carry flow addition [14]. Unlike traditional architectures, in [15] the only critical path in a multiplier is derived through a set of logical AND gates.

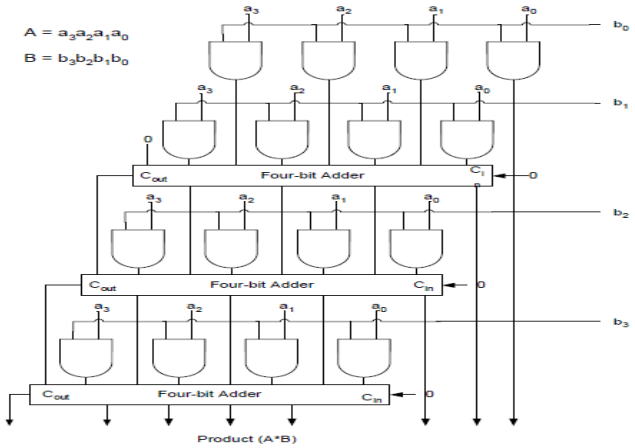


Fig. 1: A conventional multiplier unit design [18]

The required processing 2n clock can create partial product lines in n clock cycles instead of 2n cycles, which reduces the delays. Column compression techniques are used to reduce the height of the partial product tree. Thus, the number of computational cycles are decreased. In addition, counters only change the input '1', which leads to low switching power [16,17]. Adder units are the main constituting unit of a signal processing unit. In this unit for a fast arithmetic processing, various design approaches were developed. in [18,19] an adder design unit for signal processing applications is proposed. in [20,21] a Kogge-stone adder design by parallel processing is suggested. These approaches were using recursive register logic to buffer temporary results and perform a successive addition of these registers in a shift manner to generate the output. These register overhead could be minimized by the usage of counter logic in performing multiplication operation. this paper defines the resource optimized register logic for developing multiplication operation using counter operation.

II. MULTIPLIER DESIGN

With the operational function of a arithmetic operation unit in digital signal processing, multiplication and additions are the main functions of the processing system. The input value and past output values are to be multiplied by some iterations, and then these products are summed up together to get final output. In addition, the values of the addition need to be designed to process negative two compliment values, each of which should be designed to generate an output. The operational functionality of the model is to be developed using a VHDL definition in target to a FPGA compatibility. To perform this operation, ripple carry adder and carry look ahead adder were defined in VHDL interfacing. The adder will have a disruptive delay and a carrying look ahead adder has a 2n+2 logical delays. These type of adder requires more logical block, but the speed enhancement is considerably important in this case. For example, the results of a 16-bit ripple adder result a 34 logical gate delay, and a carry look-ahead adder has only 10 gate delay. Different adder designs were compared to describes how the compiler generates based on the delay. Showing results, created for similar hardware requirement gives different delay performance. An overflow protection is included to improve the stability and accuracy of the adder. If an overflow is encountered, the processor generates an incorrect result. Due to the complexity of the carry-look-ahead adder, a simple ripple-carry adder has been used in its place. Carry Look-Ahead Adder use a large chip area not only for utilizing the logical resource but introduces delay. The cellular approach was used in the design of the Ripple carry adder. Originally designed by a simple one-bit adder. For a multi bit multiplier, this design included multiple inputs, two adding bits, and carry-in bit. VHDL based design were developed with several multilevel testing to determine the best ratio of speed and area coverage. Two types of multiplier, serial and parallel were used in the designing. A serial multiplier generates a little area and reduces, the speed for a n-bit multiplier. A parallel multiplier has the significance of speed, and for optimal utilization such design were used if the area is of no concern. Digital signal processor (DSP) use signed numbers, hence for such application, a signed number multiplier is designed.

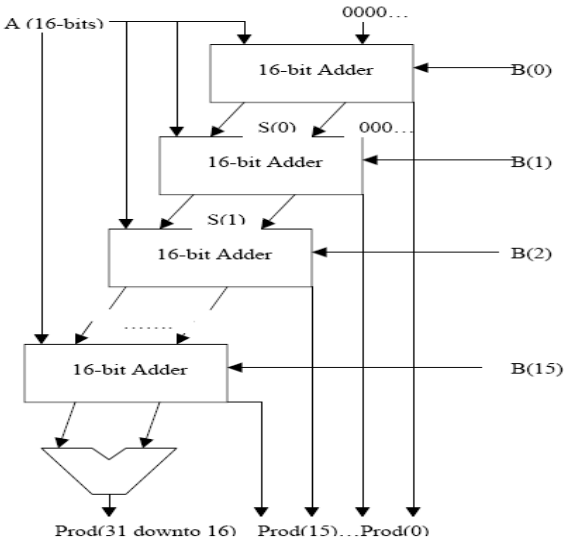


Fig. 2: A parallel signed 2's complement multiplier

A parallel multiplier were used as it is confined to a chip area. More modified booth algorithms were used to speed up the operation. The booth algorithm accelerates by minimizing the number of iterations to half for required additions. A modified Multiplier of a 16-bit booth modeling experiences a delay of less than 60 ns. However, an issue of extra sign bit is generated in this operation. This problem occurs only when there are two same numbers. This contradiction is corrected by adjusting the product by examining the specific bit positioning. In comparisons to adder design, multipliers are more complicated in modeling. A 4 bit multiplier cellular multiplier is more optimal in design than a 16bit booth multiplier, though it has a better speed factor.

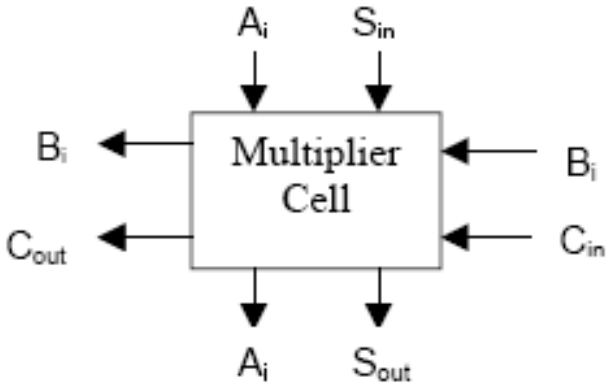


Fig. 3: Cell block unit for a 8-bit multiplier

In the design of a multiplier unit, small functional blocks of a multiplier cells are inter connected to form a n-bit multiplier. A basic cell unit is shown in figure 3. The unit takes the input as A and B generating a product result. The 2 temporary results are passed as input to the next interconnect to generate a temporary result. These results are passed to the lowest level to generate the output. To validate the result, this multiplier unit is tested for different values. These multiplication operation, generate a result which is validated by a calculator operation.

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

Compared to the answers from a calculator, the products were compared to simulation; The observations from these results show that the multiplier works properly. However, this multiplier only operates on positive numbers only, so circuits need to be added to input and output steps, so that it calculate negative numbers correctly. Input blocks take the number of 2s complement and convert it to a positive number before sending it to the multiplier. It identifies the related product range, and then the product based on positive or negative inputs is adjusted for the output value accordingly. For this, a ripple-carry adder and 2-to-1 multiplexer is been added to the input and output steps of the multiplier. To converts to a positive number, two complementary negative numbers is bit wise negation. Sign bit is used to select the appropriate value passing to the multiplier. The symbol bits of two inputs are used for output and the ripple carry adder most important bit is used to convert two complement numbers.

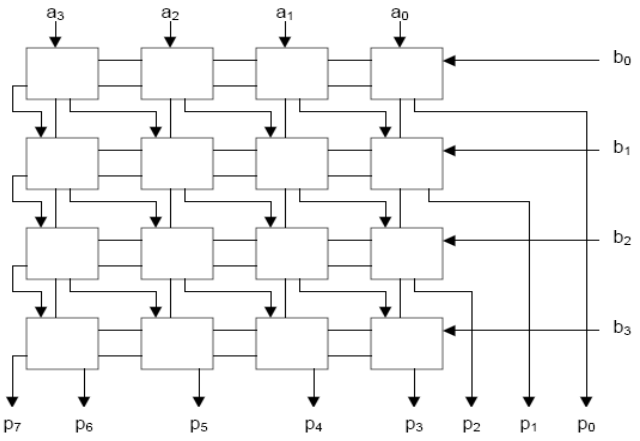


Fig. 4: A 4 –by-4 bit multiplier using multiplier cell

By configuring these signals, the port is configured to multiplier mode. Similarly, the output adjusted circuitry contains a circuit that selects the appropriate value for multiplication. However, the selected bit in this step looks like the two bits of inputs from the two major bit counters of the two ripple carry adder. The appropriate values are selected to enable the multiplexer multiplier based on these three signals. There were two separate cases to be operated. This happened when the input or output in signed and unsigned. To adjust these values, another factor of multiplexer was added to the configured circuitry for these cases.

III. ADAPTIVE LOGICAL COUNT MULTIPLIER

In the design process of a multiplier unit, the successive recursion leads to a large resource overhead and delay in resulting product. This multiplier operation has a differential multiplication and additive functionality. Multiple accumulation of N successive inputs are made to N continuous clocks. For a N bit operation, the process overhead and power consumption are high. This processing overhead leads to processing delay to reach the timing need for real time need. Therefore, to overcome this problem a new advanced adaptive logical count (ALC) design is proposed. This proposed approach is defined in two stages. The first stage of a counter component is a C3, consisting of eight 3-bit counters. The second stage implements an adder component that contains a shift-add (SA) links, as shown in the figure below.

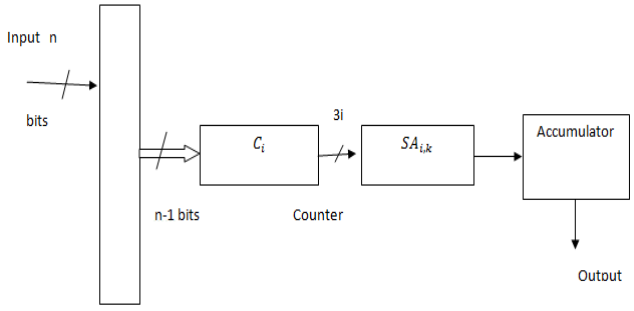


Fig. 5: Proposed adaptive count multiplier

Here,
n= counts of parallel bit to process
i = total bit counters
K = total shift add register
The time set for the clock that is used for the counter module is the worst delay in a propagation of a T flip-flop is used. At initial state, the flip-flops of all counters are reset and each counter is provided with a input words to get input from the register. For a k-parallel bit multiplier, after k-clock cycles, the condition of the flip-flops in the counter represents intermediate operator, the number of the input words as '1'. Outputs of counters in the SA-tree are left to four SA-1 cells. Each SA-1 vertical input is moved to one location towards left and its right input adds the left-handed word.
The second row of the SA tree is two left shifted and latched in SA -2. Left-input adds to its right-input.
The third stage of the SA-tree consists of a SA-4 unit, which makes it to the left to four locations and adds it to the right-input. SA-Tree Duration of Combined Path is given by, $T_1 = T_{A3} + T_{A5} + T_{A7}$ bit output adds counter balance module in a non-pipeline form. Here 3-bit, 5-bit, 7-bit addition time delays are represented by T_{A3}, T_{A5} and T_{A7} respectively. The total time of the structure of a non-pipeline implementation is the $T_c + T_1$, , where $T_c = N \cdot T_{TF}$, is the counter component, operated in counter module. Pipeline period is the time for the Counter Module and the Adder component that performs operation in two different pipeline steps with a $max\{T_c, T_1\}$. Usually the length of T_c depends on the number of input words, and T_1 depends on N and the word-length. The high values of L make the SA-tree become larger and the length of the words grows when they descend in SA Tree. In this case the module can increase the output rate using a pipeline SA tree. Similarly, when the size is larger than N, the bit-level processing becomes a large processing overhead for the multiplication operation. This introduces a large value of delay in the computation. For most values, it must prioritize the collection of specified storage in multiple pipeline steps. In any case, the clock period of the adder module is a full integer of the counter-model's clock period in the derivation of a operational clock period.

IV. SIMULATION RESULT

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

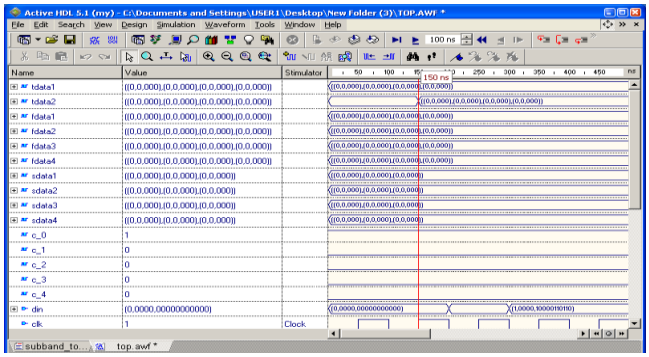


Fig. 6: Timing result for the Multiplier design

For the multiplication process, the input data is passed over the ‘din’ line and buffered into the temporary data line ‘tdata’. The temporary result are buffered on ‘tdata’ lines where each of this temporary result is passed to the next operational level. the ‘fdata’ line stores the next level shifted output from each of the shift adder (SA) unit. This data is individual passed over ‘sdata’ line. Five counter values c_0 to c_4 is used to design the counter operation. A system clock pulse for the operations passed through the ‘clk’ signal.

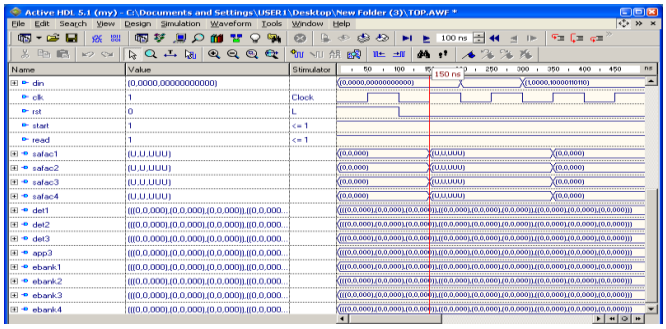


Fig. 7: Result for the multiplication operation

The simulation result illustrate above is reset by the global signal 'rst'. The design initiate the activation of the system at a zero value of the reset signal as active low operational. At the initialization of the system, the system is reset by clearing all signal and port lines at the first clock pulse. The ‘start’ signal initializes the activation operation of the designed multiplier. Control signal ‘read’ is used to initiate the read operation of the data stored in input-buffer. The content of the input buffer is read and passed to the processing unit on the high (‘1’) value of the read signal.

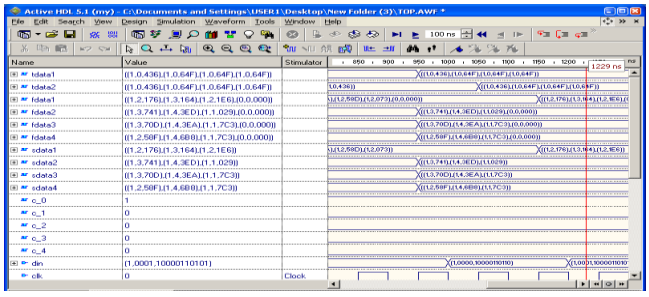


Fig. 8: Operation of multiplier in buffering of temporary data

The simulation results show the operation of design implemented for multiplication operation. The signal values read from the input buffer are shown on signal ‘tdata’ line. The design unit operates the input in parallel and separates these signals into distinct counters, where the count value each illustrated the number of active high values in a line. Temporary signals are used to transfer the data from the input line to the SA unit.

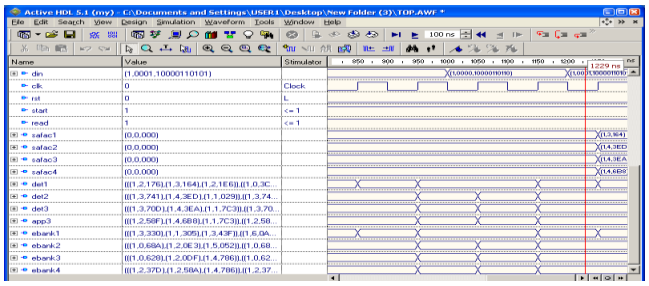
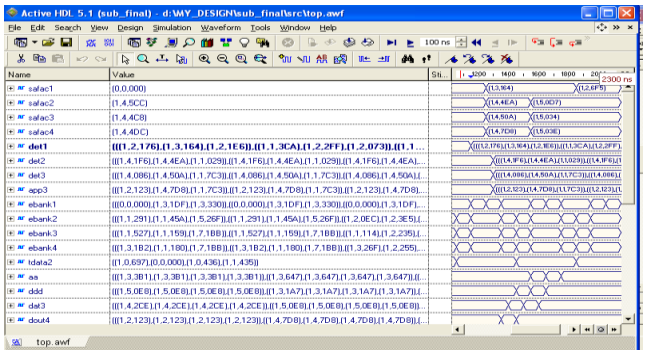
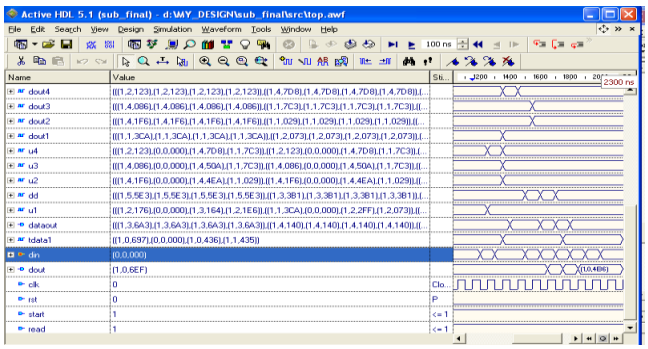


Fig. 9: Simulation result illustrating the data flow from temporary register to SA unit

Figure 9 shows the input values given to the multiplier unit. The system operates on the master clock pulse using a 100Mhz operational frequency, under the reset signal active low. The 'read' signal enable the system read of input data from the buffer unit. The ‘ebank’ buffers the shifter accumulator output to the output register.



(a)



(b)

Fig. 10: Comparative results for the developed unit (a) Input values and (b) Output values

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

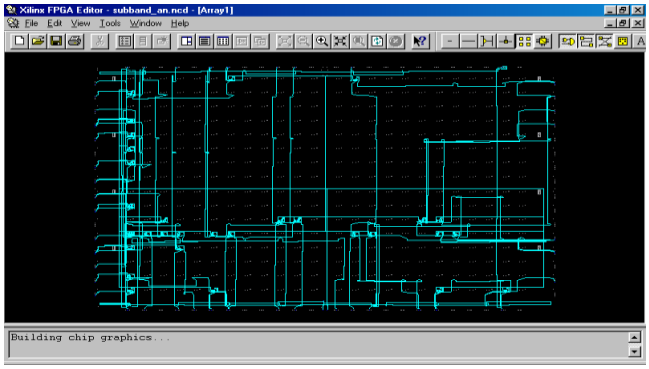


Fig. 11: Logical routing with CLB interconnect on a Xilinx FPGA device (Xc2s50e-ft256-7)

The logical placement of the implement unit on a targeted FPGA is illustrated in figure 11. The implementation illustrates the CLB configuration and routing of CLB interconnect in the FPGA device. The I/O interface and the signal transition are observed in the interconnection of FPGA logic.

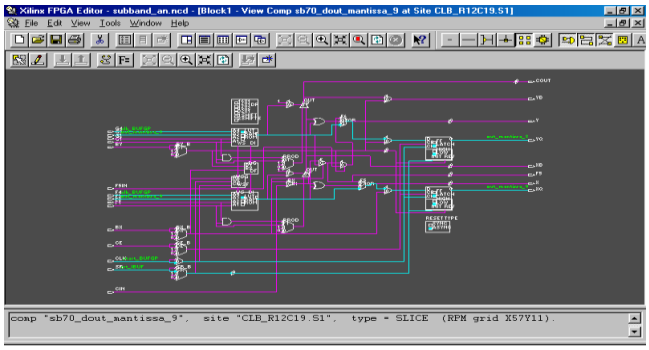


Fig. 12: logical placement into the CLB of FPGA device (xc2s50e-ft256-7)

The logical placement and interconnect of FPGA CLB units into the targeted device is illustrated. The CLB consist if logical blocks, mux, LUT and I/O buffers are observed. Each if the input signal is passed though buffer element and the local unit to process the temporary output. This operation cascaded to the successive CLB units for the operation.

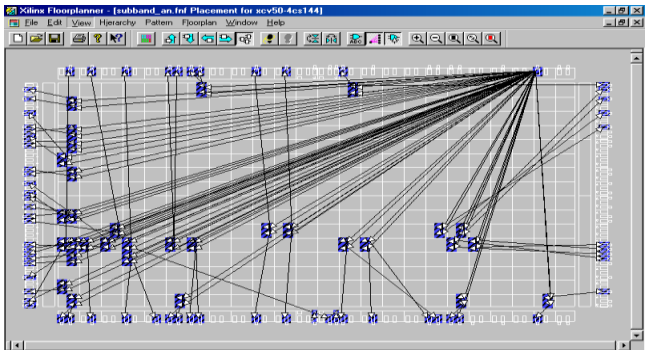


Fig. 13: Floor planning on to the targeted FPGA (Xc2s50e-ft256-7)

The floor planning for the develop approach is illustrated in figure 13. The floor plan of the implemented design onto the targeted FPGA device is seen. The interconnects are observed, and each of the net is monitored for logical CLB and input output interconnects. The floor plan control the net interconnects for optimization.

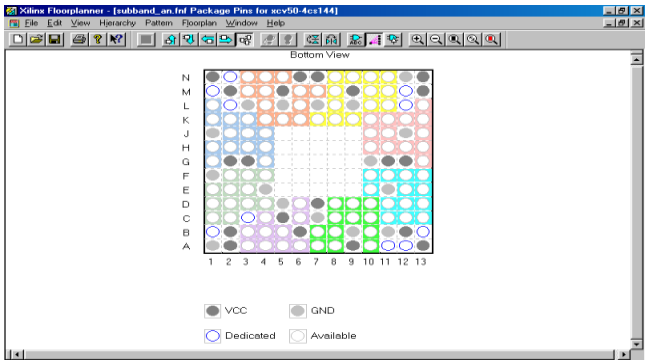


Fig. 14: Package View (Xc2s50e-ft256-7)

The package pin configuration of the develop unit is illustrated in figure 14. The pin configuration of global lines, VCC and Ground interconnections are illustrated. This unit illustrates an occupied IO's of 105 lines which are dedicated for input output flow.

Observation:

Parameter	Parallel Multiplier [1]	ALC multiplier (proposed)
Time period	3.148ns	2.133ns
Maximum operational Frequency	231.311MHz	532.34MHz
Number of Input-output lines	105	105
Number of BELs	1023	976

V. CONCLUSION

This paper presented a design approach for multiplier designing in accordance to block counting in multiplier operation. A enhanced algorithm of multiplication using multiple accumulations, where N inputs of L-size are transformed into bit counts and used for modifying the required accumulated numbers for shift-add operation. The proposed algorithm has low energy consumption and less operational complexity when compared to conventional methods. The specified multiplier design consumes a lower computation time. This design provides a solution for multiplication processing with low-resource overhead systems

REFERENCES

1. Amin Malekpour, Alireza Ejlali, Saeid Gorgin, "A comparative study of energy/power consumption in parallel decimal multipliers, Microelectron. J , Elsevier, 2014.
2. B. Parhami, Computer Arithmetic, Algorithm and Hardware Design, Oxford University Press, New York, pp. 91-119, 2000.
3. Stephen Brown and Zvonko Vranesic, Fundamentals of Digital Logic with VHDL Design. 2nd Edn. McGraw-Hill Higher Education, 2005.
4. Wakerly, J.F., 2006. Digital Design-Principles and Practices. 4th Edn. Pearson Prentice Hall.
5. A. Sertbas and R.S. Özbey, "A performance analysis of classified binary adder architectures and the VHDL simulations", J. Elect. Electron. Eng.,Istanbul, Turkey, vol. 4, pp. 1025-1030, 2004.

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

6. N. R. Strader and V. T. Rhyne, "A canonical bit-sequential multiplier," IEEE Trans. Comput., vol. C-31, no. 8, pp. 791–795, August 1982.
7. Nam Su Changa,, Tae Hyun Kim, Chang Han Kim, Dong-Guk Han, Jongin Lim, "A New Bit-Serial Multiplier Over Gf(P^m) Using Irreducible Trinomials," Computers and Mathematics with Applications , 60 (2010),pp-355-361.
8. Deepak Bordiya and Lalit Bandil, "Comparative Analysis Of Multipliers Serial And Parallel With Radix Based On Booth Algorithm," International Journal of Engineering Research & Technology, Vol. 2 Issue 9, September – 2013.
9. Arash Reyhani-Masoleh and M. Anwar Hasan, "On Low Complexity Bit Parallel Polynomial Basis Multipliers," CHES 2003, LNCS 2779, pp. 189–202, Springer 2003.
10. Nishat Bano, "VLSI Design of Low Power Booth Multiplier," International Journal of Scientific & Engineering Research, Volume 3, Issue 2, February -2012.
11. Vignesh Kumar. R and Kamala. J, "High accuracy Fixed Width Multipliers Using Modified Booth Algorithm," International Conference on Modelling Optimisation and Computing, 38 (2012) 2491 – 2498, 2012.
12. A.Bellaouar and M.Elmasry, "Low Power Digital VLSI Design: Circuits and Systems", Kluwer Academic Publishers. 1995.
13. Jin-HaoTu and Lan-Da Van, "Power-Efficient Pipelined Reconfigurable Fixed-Width Baugh-Wooley Multipliers" IEEE Transactions on computers, vol. 58, No. 10, October 2009.
14. ManasRanjanMeher, Ching-ChuenJong, Chip-Hong Chang, "High-Speed and Low-Power Serial Accumulator for Serial/Parallel Multiplier" IEEE, 2010.
15. B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs. New York: Oxford Univ. Press, 2009.
16. Manas Ranjan Meher, Ching Chuen Jong, Chip-Hong Chang, "A High Bit Rate Serial-Serial Multiplier with On-the-Fly Accumulation by Asynchronous Counters," IEEE transactions on Very Large Scale Integration (VLSI) systems, 2011.
17. http://en.wikipedia.org/wiki/Kogge Stone_adder.
18. J.Rabaey, "Digital Integrated Circuits: A Design Perspective", Prentice Hill, Second Edition, 2003.
19. Ragini Parte and Jitendra Jain, "Analysis of Effects of using Exponent Adders in IEEE- 754 Multiplier by VHDL," International Conference on Circuit, Power and Computing Technologies, ICCPCT, 2015.
20. Sarita Singh and Sachin Mittal, "VHDL Design and Implementation for Optimum Delay & Area for Multiplier & Accumulator Unit by 32-Bit Sequential Multiplier," International Journal of Engineering Trends and Technology- Volume 3 Issue 5- 2012.
21. Wasil Raseen Ahmed, "FPGA Implementation of Vedic Multiplier Using VHDL," International Journal of Emerging Technology and Advanced Engineering, Volume 4, Special Issue 2, April 2014

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