International Journal of Advances in Scientific Research and Engineering (ijasre)
ISSN: 2454-8006 [Vol. 03, Issue 5, June -2017]
www.ijasre.net

Design and Implementation of 32-Bit Arithmetic Divider and Multiplier using Single Stage Design

Jatinder Thakur[1] and Nishant[2]

PG Scholar, Department of EEE

Assistant Professor, Department of EEE

Arni University, Kathgarh,

Indora, HP (India)

ABSTRACT

Multiplier and divider are the most important parts of any arithmetic unit. Design parameters area, speed and power consumption are main constraints in designing multiplier and divider. In the proposed work we will use single stage design technique to design multiplier and divider. In single stage implementation design the complex logic operations which consist of various multiple numbers of stages are converted into single stage implementation by using single stage design the many short delays are compensated by a single large delay and performance of the design will improve. Xilinx software is use for coding and simulation will be done using Questa Sim simulator and overall design will be implemented on vertex 5 FPGA.

Keywords: Multiplier, Divider, FPGA, Throughput, Single stage, Multiple stages.

I. INTRODUCTION

Day by day IC technology is obtaining additional advanced in terms of style and its performance analysis. A quicker style with lower power consumption and smaller space is implicit to the trendy electronic styles. Unceasing advancement in electronics style technology makes improved use of energy, code knowledge with success, communicate info way more firm, etc. significantly, several of those technologies address low-power consumption to fulfill the necessities of assorted transportable applications. In these application systems, a multiplier and a divider could be a basic arithmetic unit and wide employed in circuits that the multiplication and division methods ought to be optimized properly. Multipliers and Dividers typically have extended latency, huge space and consume substantial quantity of power. Thus low-power number style has become a very important half in VLSI system style. Everyday new approaches square measure being developed to style low-power multipliers at technological, physical, circuit and logic levels. Since multiplier is mostly the slowest component during a system, the system's performance is decided by performance of the multiplier.

2. MULTIPLIER DESIGN

Multiplication is thought of to incorporates 3 basic steps: generation of partial product (PPG), partial product reduction (PPR), and at last at the top addition of carry propagate (CPA).In general we've got combinatory and ordered multiplier factor implementations. Here we have a tendency to area unit taking into thought the combinatory case solely, as a result of the size of integration currently has become large enough to begin accommodating parallel multiplier factor applications in digital VLSI circuits. Completely different multiplication algorithms vary within the approaches of generation and reduction of Partial product and also the addition method. So as to diminish the amount of PPs concerned and so reduce the area/delay of the circuit, one quantity is sometimes recoded into high-radix digit sets.

Types of Multiplier Design

i) Booth's Encoding:- Booth's encryption or Booth's multiplication rule could be a multiplication algorithm which might multiply 2 signed binary numbers during a two's complement notation. Booth's rule has the power to perform fewer additions and subtractions as compared to traditional multiplication rule.

ii) Modified Booth's Algorithm:- One of the various solutions of realizing high speed multipliers is enhancing correspondence that helps in decreasing the quantity of ulterior calculation levels. the initial version of Booth formula (Radix-2) had 2 specific drawbacks.

iii) Array Multiplier:- With an array multiplier two binary numbers will be multiplied by using of an array of half adders and full adders. Simultaneously addition of the different product terms is done in this array. By using an array of AND gates, the partial product terms are formed..Following this an array of AND gates, the adder array is used. The hardware structure for an pxq bit multiplier is described as (pxq) AND gates (p-1)q adders .Here q Half adders and (p-2).q Full adders. Array multiplier doing the multiplication process in traditional way.

iv) Vedic Multiplier:- In Vedic mathematics, two of sixteen sutras are mainly used for multiplication process. One is Urdhva-tiryagbhyam sutra and other is Nikhilam Navatascaramam dasatahs. Urdhva – Tiryagbyam performs the operation of two decimal numbers multiplication. It is applicable to all types of multiplication between two large numbers. It is also referred as "Vertically and crosswise algorithm". This can solve the multiplication of larger number (N X N bits) by breaking it into smaller sizes. Vedic multiplier gives the improved speed than the conventional multiplier and reduces the system memory. Very small area is needed for this multiplier. For binary and decimal number multiplication this multiplier is used exclusively. To solve complex calculations by simple techniques Vedic Mathematics is used.

v) Wallace tree multiplier:- Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers. Wallace trees are irregular structure in that the informal description does not specify a systematic method for the compressor interconnections. But still it is an efficient implementation of adding partial products in parallel. Using this method, a three step process is used to multiply two integer numbers. First step is to multiply each bit of one of the arguments, by each bit of the other, yielding *n2* results. Based on the position of the multiplied bits, the wires carry different weights. The second step is to reduce the number of partial products to two by layers of full and half adders. The third step is to group the wires in two numbers, and then add them with conventional adder

3. DIVIDER DESIGN

In the division, the dividend A is divided by divisor B. The result is called the quotient Q. The remainder R may not be zero if the divisor is not a factor of the dividend.

We may look at the dividend and the divisor. The most significant 4 bits of the divisor are 0. We figure that the first quotient bit that can be 1 is at least 4 bit positions from the left. It is harder for the hardware to figure this one out. The hardware is most adept at the systematic approach. One idea that we can take from the paper and pencil approach is to subtract the dividend by the divisor at the right bit positions. Since it is not easy for the hardware to figure out where the first 1 in the quotient should be, we can start from the very beginning.

4. SINGLE STAGE IMPLEMENTATION DESIGN

In single stage implementation design the complex logic operations which consist of various multiple numbers of stages are converted into single stage implementation as shown in Fig 2. Time taken by data to reach at D_{out} from D_{in} is less in comparison to the multiple stages design shown in Fig 1. In multiple stages design there are many combinational delay and also the same number of flip flop delays but in single stage implementation there is only one combinational delay that is slightly bigger than combinational delay in multiple stage design and also less flip flops is used in single stage implementation so overall throughput is increased in single stage implementation.

***4.1 Design with Divided Smaller Combo Delays* -** This design represents a multiple stage implementation having several small combo delays, propagation delay of in between flip flops and set up delay of in between flip flops. Due to multiple stage implementations it has several delays due to flip flop. The clock time period must be greater than the sum of time period of propagation delay due to input and in between flip flops, time period of set up time of output and in between flip flops and small combinational delay of each component.

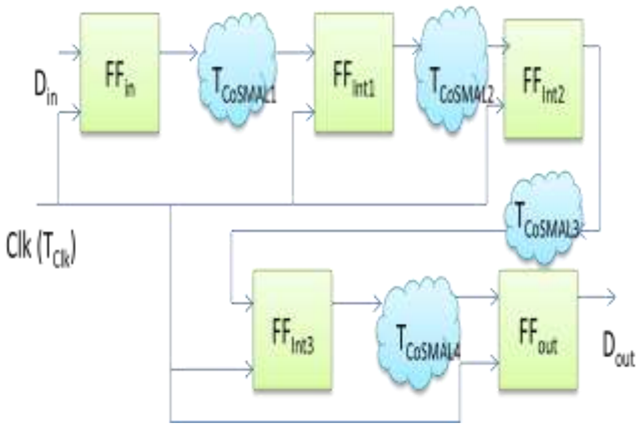

Figure 1: Design with Low Throughput (Divided into Smaller Combo Delays)

$$\mathbf{T_{Clk}} \geq (T_{propFFin} + T_{coSMAL1} + T_{setupFFInt1}) + (T_{propFFInt1} + T_{coSMAL2} + T_{setupFFInt2}) + (T_{propFFInt2} + T_{coSMAL3} + T_{setupFFInt3}) + (T_{propFFInt3} + T_{coSMAL4} + T_{setupFFout})$$

T_{Ckl}= Time period of clock

$T_{propFFin}$=Propagation delay of input flip flop and in between flip flops

$T_{comboSMAL}$=Small Combinational delay due to in between circuits

$T_{setupFFout}$=Setup time of output flip Flop and in between flip flops

***4.2 Design with one Bigger Combo Delay*-** This design represents a single stage implementation having a bigger combo delay, propagation delay of input flip flop and set up delay of output flip flop. Due to single stage implementation it has only two delays due to flip flop. The clock time period must be greater than the sum of time period of propagation delay due to input flip flop, time period of set up time of output flip flop and combinational delay in between circuits.

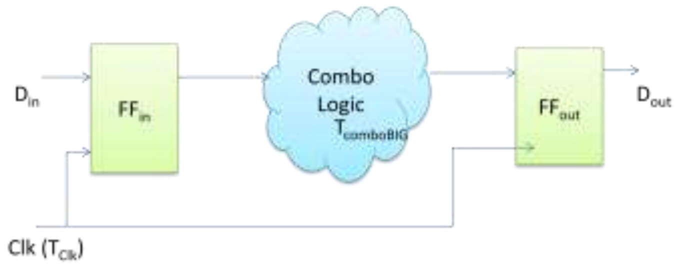

Figure 2: Design with High Throughput (Bigger Combo Delay)

$\mathbf{T_{Clk}} \geq T_{propFFin} + T_{comboBIG} + T_{setupFFout}$

T_{Ckl}= Time period of clock

$T_{propFFin}$=Propagation delay of input flip flop

$T_{comboBIG}$=Combinational delay due to in between circuits

$T_{setupFFout}$=Setup time of output flip Flop

4.3 Proposed Multiplier Design Using Single Stage Implementation

The block diagram of the proposed multiplier arithmetic unit is given in figure 3.1. The unit supports 32 bit multiplication operation. The unit has two 32 bit input operand and one output of 64 bit for the final result and one output for exception handling. The unit also consists of a 32 bit register and a 64 bit product register. So that every time calculated in between outputs will be save in this register. Unit also consists of a control unit which is used for controlling applications.

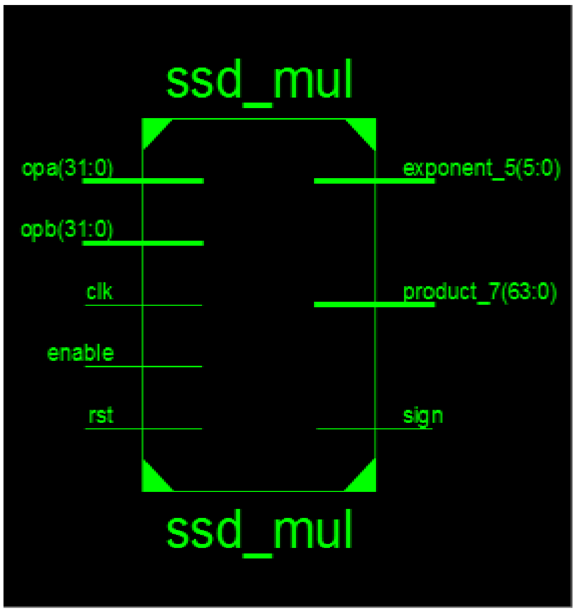

Figure 3: RTL view of Multiplier

The above diagram shows the RTL (Register Transfer Logic) view of multiplier designed with the help of single stage design.

It has following inputs.

i) Two 32 bit input operands (opa and opb):- Inputs are given from this pins. Any data which is to be multiplied is given here.

ii) One clock signal:- This input is used to apply clock signal.

iii) One enable signal:- The unit will only work if enable pin is high. Some times it is required that there is no output. So we can stop working of multiplier unit by using this pin.

iv) One reset signal:- This signal is used to create a reset state. All applying this signal all values beocomes zero.

It has following outputs.

i) One 64 bit product ouput (Product) :- The final output will come here after multiplication of two 32 bit inputs.

ii) Exception output:- If is there any exception in the result this output will show a high signal. Example Overflow, Underflow etc.

4.4 Proposed Divider Design Using Single Stage Implementation

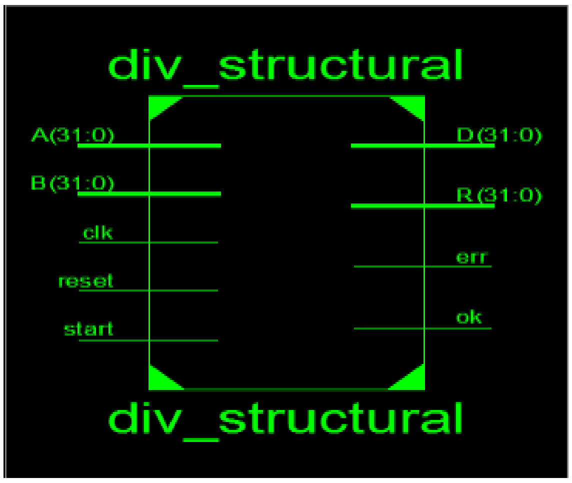

Figure 4: RTL view of Divider

The Divider Unit Consists of Following Inputs:-

i) Two 32 bit input operands (A and B):- Inputs are given from this pins. Any data which is to be divided is given here.

ii) One clock signal:- This input is used to apply clock signal.

iii) One start signal:- Start signal is used to start the design. The unit will not give nay output unless this pin becomes high.

iv) One reset signal:- This signal is used to create a reset state. All applying this signal all values beocomes zero.

It has following outputs.

i) One 32 bit dividend :- The final divident is comes here.

ii) One 32 bit Reemainder :- The final remainder is comes here.

iii) Error output:- If is there any exception in the result this output will show a high signal. Example Overflow, Divide by Zero etc.

5. RESULTS AND DISCUSSIONS

The complete code is synthesis using Verilog, Simulate using Questa Sim Simulator which is a advance version of model sim simulator and implementation is done using Vertex-5 FPGA. The FPGA that is used for the implementation of the design is the Xilinx Vertex-5 (family), XC5VLX30 (Device), FF324 (Package) FPGA device. The working environment/tool for the design is the Xilinx ISE 12.4.1 is used for FPGA design flow of Verilog code.

The Table below showing the Area & Speed Results for multiplier unit implemented on vertex 5 FPGA

TABLE 1. DEVICE UTILIZATION SUMMARY (MULTIPLIER)

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	921	93,120	1%
Number used as Flip Flops	915		
Number used as Latches	6		
Number of Slice LUTs	1,121	46,560	2%
Number used as logic	1,101	46,560	2%
Number of occupied Slices	324	11,640	2%
Number of LUT Flip Flop pairs used	1,169		
Number with an unused Flip Flop	475	1,169	40%
Number with an unused LUT	48	1,169	4%
Number of fully used LUT-FF pairs	646	1,169	55%
Number of bonded IOBs	117	240	48%
Minimum Period	2.273ns		
Maximum Frequency	439.947MHz		

TABLE 2. DEVICE UTILIZATION SUMMARY (DIVIDER)

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	102	93,120	1%
Number used as Flip Flops	102		
Number of Slice LUTs	110	46,560	1%
Number used as logic	110	46,560	1%
Number using O6 output only	77		
Number using O5 and O6	33		
Number of occupied Slices	46	11,640	1%
Number of LUT Flip Flop pairs used	110		
Number with an unused Flip Flop	41	110	37%
Number with an unused LUT	0	110	0%
Number of fully used LUT-FF pairs	69	110	62%
Number of unique control sets	1		
Number of bonded IOBs	133	240	55%
Minimum Period	2.020ns		
Maximum Frequency	494.970MHz		

6. CONCLUSION AND FUTURE WORK

Arithmetic multiplier and divider is being designed and implemented in this work. The unit has been coded in Verilog. Code has been synthesized for the Virtex-5 FPGA board using XILINX ISE and has been implemented and verified on the software successfully. Single stage implementation techniques are utilized in the proposed multiplier and divider unit. Due to single stage design the longer combinational path can be compensated by shorter path delays in the subsequent logic stages and maximum frequency of the design will be increased. Due to single stage implementation the time required to complete the various operations is less in proposed multiplier and divider unit. In the proposed multiplier and divider unit the complex logic operations which consist of various multiple numbers of stages are converted into single stage implementation. The proposed multiplier and divider unit takes less no. of clock cycles in completing their operations. And also due to single stage design the area consumed in proposed multiplier and divider unit is much less. So it has been concluded that due to single stage design the performance of proposed multiplier and divider is better than the multiplier and dividers in the previous design.

The designed arithmetic multiplier and divider units operates on 32-bit operands and implemented on Virtex-5 FPGA it can also be implemented on high performance FPGA like Virtex-6 or Virtex-7 FPGA. When the multiplier and divider unist implements on higher performance FPGA like Virtex-7 both the speed and area of the design will improve but the system becomes more costly. It can also be extended to have more mathematical operations like trigonometric, logarithmic and exponential functions. For high performance the complete units is also designed for 64 nit operations.

REFERENCES

[1] Dempster A.G. and Macleod M.D.: ’Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters’, IEEE Trans. on Circuits and Systems II, Vol. 42, No. 9, 1995, pp. 569-577.

[2] Hägglund R., Löwenborg P., Vesterbacka M.: ‘A polynomial-based division algorithm’, IEEE Trans. on Circuits and Systems III, Vol. 3, 2002, pp. 571-
574.

[3] Soderquist, P. and Leeser, M.: ‘Division and square root choosing the right implementation’, IEEE Micro, Vol. 17, No. 4, 1997, pp. 56-66.

[4] Soma BhanuTej, “Vedic Divider - A High Performance computing Algorithm for VLSI Applications”, IEEE International Conference on Advances in Electronics, Computers and Communications, pp.57–61, 2013.

[5] Surabhi Jain, MukulPancholi, Harsh Garg and Sandeep Saini, “Binary Division Algorithm and High Speed Deconvolution Algorithm”, International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, pp.32-35, 2014.

[6] Miss. AditiS.Tadas andProf. Dinesh Rotake,“Design and Simulation of 64 Bit Divider Using Vedic Mathematics”, International journal of advanced information and communication technology (IJAICT), Vol.1, pp.608-609, 2014.

[7] Ruchi Anchaliya, Chiranjeevi G.N. and Subhash Kulkarni, “Efficient Computing Techniques using Vedic Mathematics Sutras”, International Journal Of Innovative Research In Electrical, Electronics, Instrumentation And Control Engineering(IJIREEICE), Vol. 3, pp.24-27, 2015.

[8] DalalRutwikKishor andV.S.KanchanaBhaaskaran, "Low Power Divider Using Vedic Mathematics", IEEE International Conference on Advances in Computing, Communications and Informatics, pp.575-580, 2014.

[9] Sushma R. Huddar and Sudhir Rao Rupanagudi, Kalpana M., Surabhi Mohan "Novel High Speed Vedic Mathematics Multiplier using Compressors" IEEE ISBN 978-1-4673-5089-1.

[10] Sushma R. Huddar and Sudhir Rao Rupanagudi, Kalpana M., Surabhi Mohan "Novel High Speed Vedic Mathematics Multiplier using Compressors" IEEE ISBN 978-1-4673-5089-1.

[11] A.D.Booth, "A Signed Binary Multiplication Technique," J. mech. and appl. math, vol 4, no.2, pp. 236-240, Oxford University Press, 1951.