Design and Simulation of 64 Bit FPGA Based Arithmetic Logic Unit

Nuray Sağlam Bedir, Fırat Kaçar

Department of Electrical and Electronics Engineering, İstanbul University-Cerrahpaşa School of Medicine, İstanbul, Turkey

Cite this article as: Sağlam Bedir N, Kaçar F. Design and Simulation of 64 Bit FPGA Based Arithmetic Logic Unit. *Electrica*, 2019; 19(2): 158-165.

ABSTRACT

Arithmetic Logic Unit (ALU) is the essential part of the Central Processing Unit (CPU) core which performs arithmetical operations such as addition, subtraction, division, multiplication etc., logical operations such as and, or, xor etc. and shift-rotate operations. The CPU performance is directly related to the performance of ALU. In this study, the 64-bit ALU has been designed by using the Very High Speed Integrated Circuits Hardware Description Language (VHDL) and Altera Field Programmable Gate Array (FPGA) families, synthesized and simulated with the help of Altera Quartus II (Intel, Santa Clara, CA, USA) v13.0sp1 and Modelsim-Altera v10.1d (Intel, Santa Clara, CA, USA) software. Many different studies are given about ALU Design and Implementation with the use of FPGA architecture and VHDL language. The difference of this study from recent studies is that the proposed design allows the processing of the signed numbers. Also, Conditional Sum Adder (COSA) is used in addition operation instead of Carry Ripple Adder (CRA) or Carry Look-ahead Adder (CLA) because of its benefit in fast addition and less propagation delay of Carry Chain.

Keywords: FPGA, VHDL, arithmetic logic unit design

Introduction

In a computer system, Arithmetic Logic Unit (ALU) is the vital structure of the Central Processing Unit (CPU). Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, today's CPUs have a control unit (CU) which operates the ALU through the control signals. These signals tell to ALU which operations will be performed and the ALU stores results of these operations in output registers. Also, the CU moves the data between these registers, the ALU, and memory through the control signals.

Very High Speed Integrated Circuits Hardware Description Language (VHDL) is one of the most popular languages of an industry for the modeling, description, and synthesis of digital circuits and systems. It is a high-level language that is difficult to learn, and suitable for the design of complex systems. Also, this language allows users to create complex data types. Design units, also called library units, are the main components of the VHDL language. It consists of 4 different notifications which are package, entity, architecture and configuration-component. Also, new library creation and library management which are two of the most valuable features is allowed in VHDL language [1, 2].

The main reason of using VHDL in this study is that the language allows us to manage the library and create a special structure such as package, block, function, procedure, and component etc. This feature made the design simpler and easy to manage. Also, using these structures provide portable and reusable designs.

A basic field programmable gate array (FPGA) is an integrated circuit with logic blocks which are arranged in a matrix. In addition to logic blocks, modern FPGAs have multiplier blocks and memory blocks inside them. In an FPGA structure, logic blocks and interconnections between them, input/output elements (IOE) etc. are configurable. When an FPGA is configured, the internal circuitry is connected in a way that creates a hardware implementation of our design. If

Corresponding Author:
Fırat Kaçar

E-mail:
fkacar@istanbul.edu.tr

Received: 25.12.2018

Accepted: 07.03.2019

DOI: 10.26650/electrica.2019.18052

the FPGA needs to be reprogrammed, the configuration data in it can be deleted and programmed again and again, and this is the most important reason why FPGA is so valuable in the market. Unlike CPUs, FPGAs do not have a fixed hardware design. In contrast, they can be programmed from scratch according to user applications. The functions that logical cells perform and the connections between these functions are determined by the user. Some of other important features of an FPGA have made its use popular include parallel processing capability, design testing and validation, and the ability to embed a processor in an FPGA as discussed by [3, 4].

Many different studies are given about ALU Design and Implementation with the use of FPGA architecture and VHDL language. In these studies carry ripple adder (CRA) or carry look-ahead adder (CLA) is used in addition operation and unsigned numbers are processed in ALU. Also, Xilinx's software and hardware are used as a development environment as discussed elsewhere [5-10].

Adder cell is the primary unit of an ALU. Power, speed and area requirements need to be satisfied by the adder cell. The delay in the adder circuit originates from the carry bit calculation. Previous studies used CRA structure in their adder unit [7-10].

Carry ripple adder is the simplest but the slowest adder structure and constructed by cascading full adders (FA) blocks in series. In the CRA technique, a sequential addition is performed from the less significant bit (LSB) to the most significant bit (MSB) using the FA structures. In this addition, the initial carry bit input is taken as zero during the sum of the LSB bit. While performing the sum of the next two bits, the new carry bit which is calculated from the sum of the previous two bits is used here as a carry bit input. This process runs until the sum of the MSB bits is reached, and the carry bit inputs are calculated from the sum of the two previous bits each time. Thus, in order to generate the carry bit input to be used in the sum of the MSB bits, all carry bits, from the LSB bit, have to be calculated. In this carry bit calculation process, a delay occurs and as the size of the processed data grows, this delay also increases in parallel. The worst-case delay of the CRA is approximated by equation 1.

$$t = (n - 1)\, t_c + t_s \qquad \text{Eq (1)}$$

t_c stands for the delay through the carry stage of a FA, t_s stands for the delay to compute the sum of the last stage and n stands for the number of bits. The delay of CRA is linearly proportional to the number of bits; therefore the performance of the CRA is limited when the number of bits increases [11].

Other conventional types of adders are carry-look-ahead adder, carry-skip adder, carry select adder, conditional sum adder (COSA), and Manchester Carry Chain adder. In these conventional types of adders including CRA, COSA is the extension of Carry Select Adder and has less delay in its operation because of the carry bit computation method.

The basic idea in the COSA is to calculate two different outputs for a given group of input bits, for instance, n bits. In each calculation, n bits addition result and a carry bit out are obtained. In one of the addition operations, the carry input is taken as zero and the other is taken as one. By using a multiplexer structure, the correct carry input is selected and the correct addition result is obtained through this carry bit selection. Thus, the result is calculated without waiting for the calculation of the carry bit. The addition is much faster as higher bit addition operations [12]. Block diagram of 4 Bit COSA is given in Figure 1 [13].

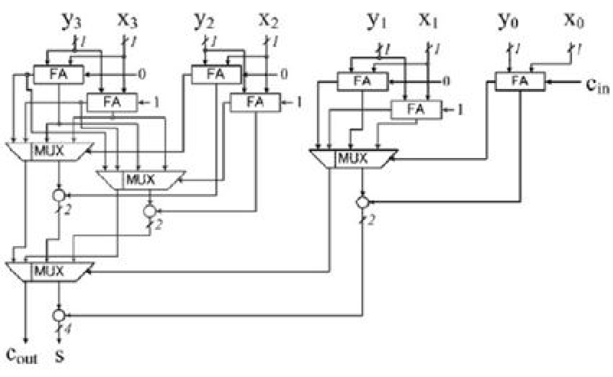


Figure 1. 4 Bit Conditional Sum Adder (COSA)

Material and Method

In the study, the main ALU structure is divided into three subunits as an arithmetic unit, a logic unit, and a shift-rotate unit. Each subunit is introduced as a component to the system. Each component consists of different components.

One of the Altera's FPGA families, Cyclone II EP2C70F896C6 (Intel, Santa Clara, CA, USA) device is targeted in the design platform namely Quartus II (Intel, Santa Clara, CA, USA) and Modelsim-Altera (Intel, Santa Clara, CA, USA) is used as a simulation tool. In EP2C70F896C6 device, 2C means that this device belongs to Cyclone II family, 70F means that this device has approximately 70 thousand logic elements (LEs), 896C refers to the total number of pins both dedicated pins and user pins and last number in the device code refers to the speed grade of this device. In addition to these features, EP2C70F896C6 device includes 250 M4K random access memory (RAM) blocks, approximately 1.1Mbit RAM bits, 150 18x18 embedded multiplier and 4 phase lock loop (PLL) structures [14].

Altera Quartus II software which is developed by ALTERA is the most comprehensive environment available for a programmable chip design. It allows users to design both at gate level and register transfer level (RTL), also allows you to use mega function and intellectual property (IP) and synthesis their HDL files and also you can use this software to implement timing analysis of your design. After the synthesis and analysis of the design with the Quartus II software, the functional and temporal simulation is realized with the use of Modelsim-Altera software.

Simulations can be performed manually and directly via Modelsim-Altera or indirectly by using the testbench file written in Quartus II [15].

In this study we used RTL design which is done with use of VHDL language and wrote a testbench VHDL file for simulation process of the proposed design via Modelsim-Altera.

Design diagram of proposed ALU is given in Figure 2.

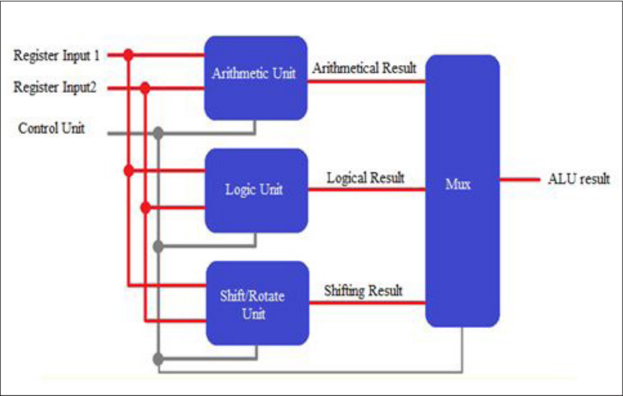


Figure 2. Arithmetic Logic Unit Design Diagram

The source code of design is written with VHDL hardware description language and behavioural and structural modelling is adopted as a modelling method. The design flow is selected as bottom to top.

Arithmetic Unit of proposed design contains signed addition, signed subtraction, signed multiplication, signed division and signed mod operations.

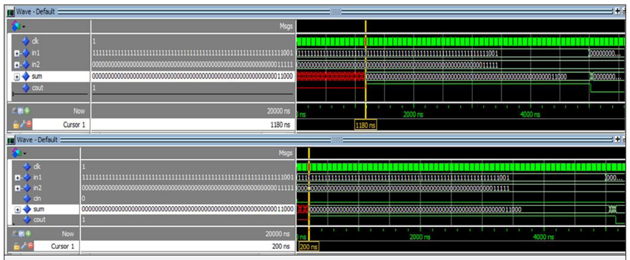
Logical Unit of proposed design contains and, or, xor, not, nand, nor and xnor operations.

Shift/Rotate Unit of proposed design contains arithmetic left and right shifting, logical left and right shifting and right and left rotation operations.

Operations and select lines of proposed ALU is given in Table 1. These 19 operations in the Table were defined as a component in the specific unit where the operation belongs. But each operation could be defined as a procedure, block or function. Each of this structure has different syntax and usage.

After the target device was selected as Cyclone II EP-2C70F896C6, as a first part, arithmetic unit is designed. The arithmetic unit has 4 inputs, namely inpu1, input 2, clk (clock) and ALU control (select input from the control unit of CPU) and 3 outputs namely arithmetic unit out, error (for division errors such as "0/0" and "data/0", overflow errors and size out errors of the multiplication operation) and flag register out (parity, overflow, sign and zero flags). Data inputs and arithmetic unit out are 64 bits long, clock input (in simulation 50MHz clock frequency is used), arithmetic unit control(selection lines) is 3 bits long (first 3 bits of ALU control input) and error output are 1 bit long and flag register output is 4 bits long.

Conditional sum adder which is used in addition operation is the most important part of the arithmetic unit. After the comparison of CRA and COSA structures, COSA is selected as main structure of the addition operation. CRA and COSA simulation result is given in Figure 3 respectively, and delay and LE usage comparison of these two adders is given in Figure 4.


```
Wave - Default
                                                                              Msgs
clk    1
in1    1111111111111111111111111111111111111111111111111111111111111001
in2    0000000000000000000000000000000000000000000000000000000000011111
sum    0000000000000000000000000000000000000000000000000000000000011000
cout   1
Now       20000 ns
Cursor 1  1180 ns

Wave - Default
                                                                              Msgs
clk    1
in1    1111111111111111111111111111111111111111111111111111111111111001
in2    0000000000000000000000000000000000000000000000000000000000011111
cin    0
sum    0000000000000000000000000000000000000000000000000000000000011000
cout   1
Now       20000 ns
Cursor 1  200 ns
```

Figure 3. 64 Bit Carry Ripple Adder (CRA) and Conditional Sum Adder (COSA) Simulation Results Respectively

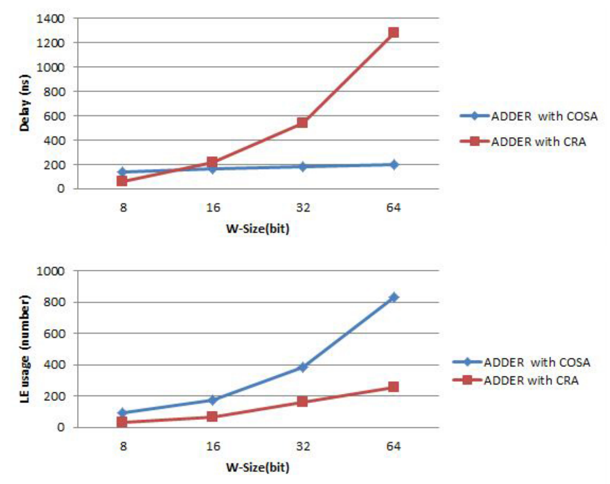


Figure 4. Delay and LE Usage Comparison of CRA and COSA

N bits COSA type addition device is composed of a two-bit COSA consisting of essentially three full adders and a multiplexer structure. One of the full adders collects the first bits of the input data with the "0" carry in. Other two full adders collect the last bits of the input data. One of them with "0" carry in and the other one uses "1" carry in. The selection of the carry-bit which is going to be used in the sum of the last bits is selected by looking at the carry bit result from the sum of the first bits with the usage of multiplexer structure. 64-bit COSA is designed as an extension of two-bit COSA structure. RTL schema of 64 Bit COSA with Overflow Detection component is given in Figure 5.

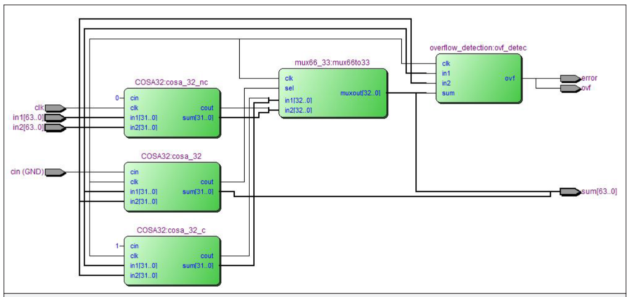


Figure 5. RTL Schema of 64 Bit COSA with Overflow Detection

Table 1. Operations and Select Lines

Function	Operation	Select Lines				
Signed ADD	(±A)+(±B)	0	0	0	0	0
Signed SUB	(±A)-(±B)	0	0	0	0	1
Signed MULT	(±A)*(±B)	0	0	0	1	0
Signed DIV	(±A)/(±B)	0	0	0	1	1
Signed MOD	(±A)mod(±B)	0	0	1	0	0
AND	A and B	0	1	0	0	0
OR	A or B	0	1	0	0	1
XOR	A xor B	0	1	0	1	0
NOT A	not A	0	1	0	1	1
NOT B	not B	0	1	1	0	0
NAND	A nand B	0	1	1	0	1
NOR	A nor B	0	1	1	1	0
XNOR	A xnor B	0	1	1	1	1
Logic left shift	±A sll ±B	1	0	0	0	0
Logic right shift	±A srl ±B	1	0	0	0	1
Arith. left shift	±A sla ±B	1	0	0	1	0
Arith right shift	±A sra ±B	1	0	0	1	1
Left Rotate	±A rotl ±B	1	0	1	0	0
Right Rotate	±A rotr ±B	1	0	1	0	1

A, B: data inputs; ADD: addition; SUB: subtraction; MULT: multiplication; DIV: division; MOD: modular, sll: shift left logical; srl: shift right logical; sla: shift left arithmetical; sra: shift right arithmetical; rotl: rotation to left; rotr: rotation to right

Subtraction component is designed with the usage of COSA.

Another important operation is multiplication operation. The Booth's Algorithm which provides great convenience in two signed number multiplication is used for multiplier component.

In the Table 1, data A and B are both 64 bits long data. In the multiplication operation, first 32 bits of data A and B are used. The multiplication operation in proposed design allows multiplying two 32 bits long data. Also, several functions has been used for checking the data if it is greater than the number which is the biggest number in 32 bit long data or not. These function checks the second 32 bits of 64 bit data. When the data is negative and if there are '0' bits in the second 32 bits of data, these functions tell us that we can't multiply this data due to the size out in the multiplication result, and this process works in positive numbers differently.

A Similar algorithm to the multiplication algorithm is used in division component design.

The last component of the arithmetic unit is a modular component and it is designed with the usage of the division component. The only difference is that the remainder of the division operation is taken as an output of the component.

After components of all operations are defined, these components are called under the arithmetic unit. The output of the arithmetic unit is taken as the output of the multiplexer structure according to the value of the selection input which is related to the ALU control input port. After the source code of the arithmetic unit was written, the design is analysed and synthesized with the help of the Altera Quartus II v13.0sp1 software. RTL schema of 64 Bit Arithmetic Unit is given in Figure 6.

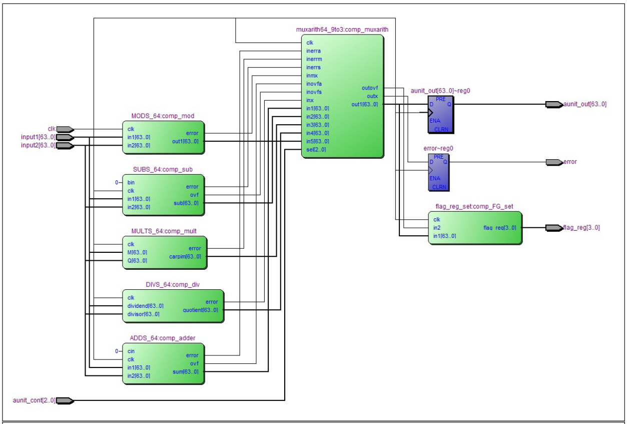


Figure 6. RTL Schema of 64 Bit Arithmetic Unit

64 Bit Arithmetic unit's simulation is implemented with the help of Modelsim- Altera v10.1d software. The simulation results of this unit are shown Figure 7 and Figure 8.

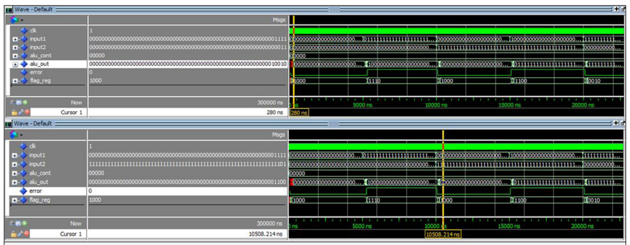


Figure 7. Positive Simulation Results of the 64 Bit Arithmetic Unit

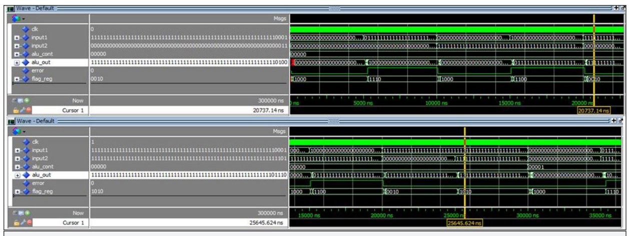


Figure 8. Negative Simulation Results of the 64 Bit Arithmetic Unit

Logic Unit and Rotate/Shift Unit which are second and third parts of ALU are designed respectively. RTL schema of Logic Unit is given in Figure 9. After the source code of the logic unit was written, the design is analysed and synthesized. The simulation result of Logic Unit is shown in Figure 10. 64 Bit Rotate/Shift Unit's RTL scheme is shown in Figure 11. After the source code of the Rotate/Shift unit was written, the design is analysed and synthesized. The simulation result of this unit is shown in Figure 12.

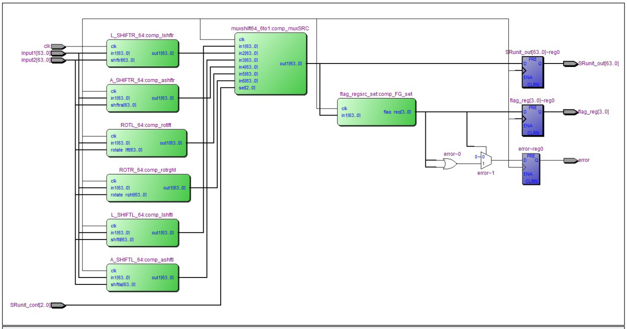


Figure 9. RTL Schema of 64 Bit Logic Unit

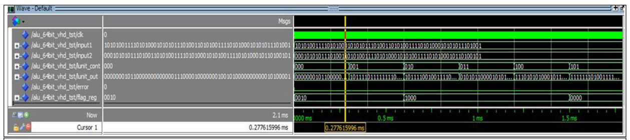


Figure 10. Simulation Result of the 64 Bit Logic Unit

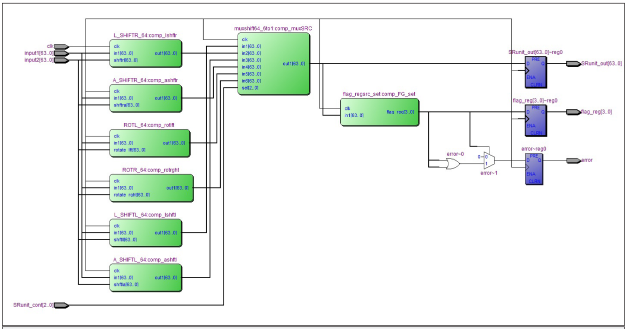


Figure 11. RTL Schema of 64 Bit Rotate/Shift Unit

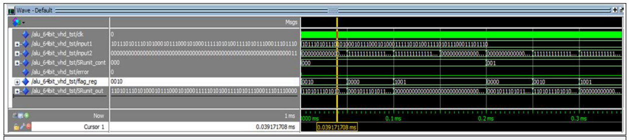


Figure 12. Simulation Results of 64 Bit Rotate/Shift Unit

These three subunits are defined as a component and called under the Arithmetic Logic Unit. The outputs of these subunits supply a multiplexer structure defined in the top-level design. Last 2 bits of ALU control input port is used as a select line and this select line supply the multiplexer selection inputs. These two bits should be "00" when we want to activate Arithmetic Unit, "01" when we want to activate Logic Unit and "10" when we want to activate Shift/rotate unit.

Arithmetic Logic Unit Output is taken from the output of the multiplexer structure according to the value of the selection input of the top-level multiplexer structure which is related to the last two bits of ALU control input port. Flag register output from MSB bit to LSB bit defines respectively parity flag, an overflow flag, sign flag, and Zero flag.

In the last case, the Arithmetic Logic Unit is defined as a single component in the most-level design and can be called and executed as a component in any design. Top-level design of 64 Bit Arithmetic Logic Unit's RTL scheme is shown in Figure 13.

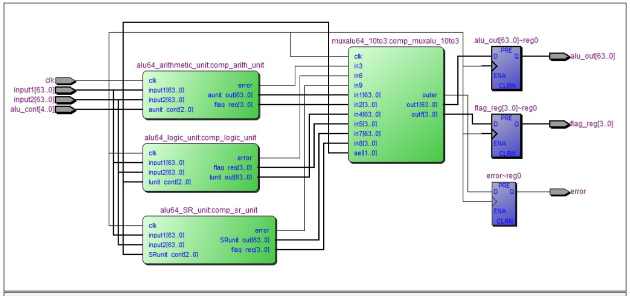
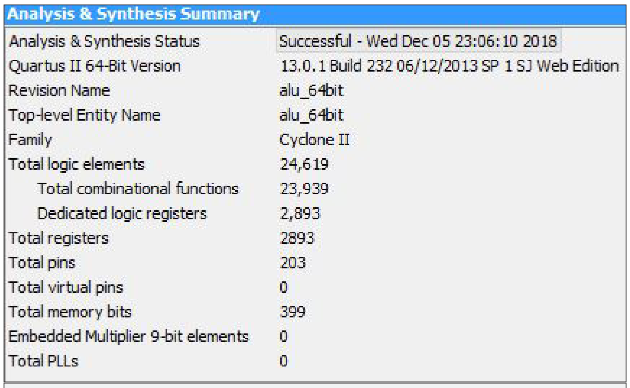


Figure 13. RTL Schema of 64 Bit Arithmetic Logic Unit

The top-level, Arithmetic Logic Unit, module is analysed and synthesized and then simulated with the help of Modelsim-Altera software. 64 Bit Arithmetic Logic Unit's compilation report is given in Figure 14.



Analysis & Synthesis Summary	
Analysis & Synthesis Status	Successful - Wed Dec 05 23:06:10 2018
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	alu_64bit
Top-level Entity Name	alu_64bit
Family	Cyclone II
Total logic elements	24,619
Total combinational functions	23,939
Dedicated logic registers	2,893
Total registers	2893
Total pins	203
Total virtual pins	0
Total memory bits	399
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Figure 14. Compilation Report of 64 Bit ALU

Results

The 64-bit arithmetic logic unit for signed numbers processing is designed using VHDL language and synthesized using Altera Quartus II v13.0sp1 platform. The ALU is implemented us-

Table 2. Comparison between previous studies and proposed design

Design	Operation (number)	FPGA Family	Data Type	Clock Rate (Hz)	Adder Type	Delay on Adder (ns)
Previous studies	Max. 16	Xilinx FPGAs	Unsigned	Max. 10 MHz	CRA or CLA	Min. 5900
Proposed Design	19	Altera Cyclone II	Signed	50 MHz	COSA	280

FPGA: Field Programmable Gate Arrays; Hz: hertz; MHz: megahertz; ns: nanosecond; CRA: Carry Ripple Adder; CLA: Carry Look-ahead Adder; COSA: Conditional Sum Adder

ing parallel implementation of three different subunits which perform various functions such as arithmetic, logical, shift and rotate operations. At first step components of subunits are designed to handle unsigned numbers. Then with the use of process structures these components are arranged to handle signed numbers. After this arrangement, the overflow control circuit is designed.

Data are received from the data input ports and these data fed to the three different subunits. These subunits calculate their result in each clock cycle and send them into top-level multiplexer structure. This multiplexer structure generates an output according to the value of the signal from the control unit which is defined as an input port named ALU control. This generated outputs from the multiplexer fed to the output ports of ALU. The synthesized most-level module is targeted to Cyclone II device.

The most level design has 4 input ports as two data input, a clock input and select input which is assumed to come from the control unit of CPU and has 3 output ports as ALU operation output, error output, and flag register output.

50 MHz clock is used as clock input in testbench simulation. Clock frequency can be increased up to 250 MHz with the PLL structure usage. Error output port shows errors which occurs in signed addition and subtraction when overflow occurs and incorrect result calculated, in multiplication of two numbers which are exceeding the maximum value expressed in 32 bits and in division operation when data inputs are selected as both zero or second input is selected as zero (these selections causes "0/0" uncertainty and "number/0" undefined state). Flag register output is 4 bits long and shows from MSB bit to LSB bit respectively parity flag status, overflow flag status, sign flag status, and zero flag status.

After testbench simulation, the first result of The ALU is taken after 280 ns. This operation is signed addition operation. Some of the simulation results of the 64 Bit Arithmetic Logic Unit are shown in Figure 15. After the simulation was completed, the results obtained from the Altera Quartus II Design Suit were confirmed by the theoretical results for all the operations with signed numbers and we found that they matched the theoretical values.

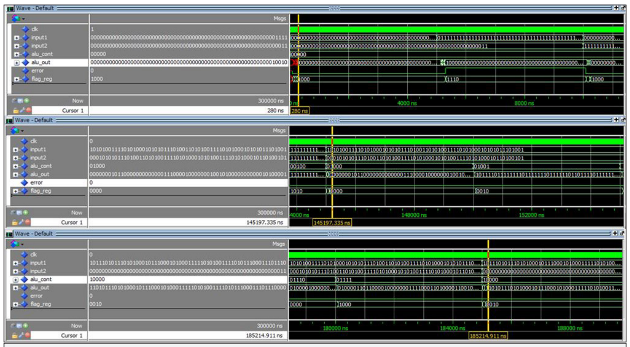


Figure 15. Simulation Results of 64 Bit ALU

Conclusion

In the addition operation, instead of CRA or CLA type of adder, the usage of COSA structure reduced the delay which consists of carry bit calculation. As a result, the output of the addition operation was obtained in a shorter time (adder with COSA is approximately 6 times faster than adder with CRA). In spite of this positive development - the increase of the multiplexer structures used in the addition circuit caused the number of logic elements used in the FPGA increase.

The 64-bit arithmetic logic unit which is designed for the processing of signed numbers has been successful in all operations regardless of whether the number is positive or negative. In case of overflow in the addition and subtraction operations, indefinite and undefined situations occurring in the division process and the result of multiplication operation exceeding 64 bits, errors are shown successfully with the error output as a result of the simulation.

The design is shaped around the "component" structure in the VHDL language. The usage of this structure has enabled us to manage our design easily and make the design simpler. In addition, all "components" can be easily used in other designs because of the special benefit of the component structure in the VHDL language. The most important disadvantage of the usage of these structures is the calculation of all operation results regardless of the process priority and this causes extended simulation time. The design can be implemented using "procedure" structures with a different approach. But with the usage of this structure, the size of the source code will increase and the readability will decrease and the design will be complicated.

Therefore, the structure to be used should be chosen according to the purpose. The main reason for us to use the "component" structure in our design is to eliminate some of the complexity resulting from the use of signed numbers which caused the addition of new structures to the design, the number of operation and the large size of the processed data. Comparison between previous studies and proposed design is given in Table 2. If the clock rate of proposed design is lowered to 10 MHz, delay on the adder unit will be 1400ns, and this delay is also lower than delay on the adder with CRA or CLA structure in previous studies. Also, when we use the PLL structure in the FPGA, clock frequency can increase up to 250 MHz and delay in the adder operation can decrease down to 60 ns.

Peer-review: Externally peer-reviewed.

Conflict of Interest: The authors have no conflicts of interest to declare.

Financial Disclosure: The authors declared that the study has received no financial support.

References

1. B. Mealy, F. Tappero, "Free Range VHDL", 2012. Available from: URL: http://www.gstitt.ece.ufl.edu/courses/eel4712/labs/free_range_vhdl.pdf.
2. E. Sarıtaş, S. Karataş, "Her Yönüyle FPGA ve VHDL", Palme Yayıncılık, Ankara, Türkiye, 2015.
3. R. Sahani, S. Gupta, N. Kishore, "FPGA In Embedded System And Its Application", IJIRT, vol. 1, no. 12, 2015.
4. Module Design of Embedded Processors. Available from: URL: https://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Embedded%20systems/Pdf/Lesson-20.pdf. Dec, 2018.
5. P. Vanjare, P. Pandey, February 2016, "Design and Implementation of 64-bit ALU using VHDL", International Journal of Engineering and Management Research, vol. 6, no. 1, pp. 500-502, 2016.
6. R. M. Rewatkar, A. V. Khode, A. S. Kalinkar, P. S. Bangde, S. D. Potey, " Design and Simulation of High Speed, less area 64-Bit ALU using Efficient Technique", International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, no. 4, pp. 326-330, 2016.
7. R. Chetia, K. C. D. Sarma, G. Baruah, "Behavioural Design and Synthesis of 64 BIT ALU using Xilinx ISE", IOSR-JECE, vol. 7, no. 4, pp. 37-41, Sept-Oct, 2013. [CrossRef]
8. P. Bhanusree, G. B. Sai, Y. A. Kumar, K. S. Kumar, "VHDL Implementation Of 64-bit ALU", IOSR-JECE, vol. 7, no. 4, pp. 14-17, Sept-Oct, 2013. [CrossRef]
9. M. P. Mahajan, P. G. Salunke, Y. M. Gaikwad, V. P. Jagtap, "Design and Simulation of 64 bit ALU", IJARECE, vol. 4, no. 4, pp. 1049-1051, Jan, 2015.
10. R. Prabhakar, C. Rekha, "VLSI Design and Implementation of Arithmetic and Logic Unit Using VHDL", International Journal of Engineering and Science, vol. 3, no. 10, pp. 62-67, Oct, 2013.
11. R. Uma, V. Vijayan, M. Mohanapriya, S. Paul, "Area, Delay and Power Comparison of Adder Topologies", International Journal of VLSI design & Communication Systems, vol. 3, no. 1, pp. 153-168, Febr, 2012. [CrossRef]
13. Hardware algorithms for arithmetic modules. Available from: URL: http://www.aoki.ecei.tohoku.ac.jp/arith/mg/algorithm.html#fsa_csa. Dec, 2018.
14. Altera® Corparation University Program, "DE2-70 Development and Education Board User Manual". Available from: URL: https://www.terasic.com.tw/attachment/archive/226/DE2_70_User_manual_v105.pdf.
15. Altera® Corparation, "Introduction to the Quartus® II Software version 10.0", Available from: URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/intro_to_quartus2.pdf. 2012.

Fırat Kaçar received his B.Sc., M.Sc. and Ph.D. degrees from Istanbul University, all in Electrical and Electronics Engineering in 1998, 2001 and 2005 respectively. He is currently an Assistant Professor at the Electrical and Electronics Engineering Department of Istanbul University. His current research interests include analog circuits, active filters, synthetic inductors, CMOS based circuits electronic device modeling and hot-carrier effect on MOS transistor. He is the author or co-author of about 100 papers published in scientific journals or conference proceedings.

Nuray Sağlam Bedir was born in Tokat, Turkey, in 1990. She is M.Sc. student. She received the B.Sc. degree from Istanbul University in Electrical and Electronics Engineering in 2014. Her main research interests are FPGA prototyping, design and implementation of digital circuits and embedded systems.