DOI 10.29042/2018-3133-3140

Helix Vol. 8(2): 3133-3140

Design of High Performance Low Leakage FPGA Slices using Control Logic

[*1] Esther Rani Thuraka, [2] Raghava Katreepalli

[1] Department of ECE, CVR College of Engineering, Hyderabad, India, Department of Electrical Engineering, Navajo Technical University, Crown point, NM, USA 87313

Email: *estherlawrenc@gmail.com, rkatreepalli@navajotech.edu*

Received: 10th December 2017, Accepted: 17th January 2018, Published: 28th February 2018

Abstract

Field Programmable Gate Arrays (FPGAs) are significant in digital designs, Scientific Computing, Digital Signal Processing and Tele-communications. To use re-configurable technology in the portable applications, the devices that consume significantly less power and programmable are essential. In FPGAs, leakage power has become a major component as the technology evolves to deep sub-micron. An effective way to reduce leakage power is to parallelly employ both Power Gating and MTCMOS techniques in the designs. The elementary impression is design of Re-Configurable Look-Up Tables (LUT) in which logic gates are controlled by sleep transistors to reduce the leakage power. For turning off logic gates, the input of the sleep transistor is controlled by a control signal. A 4-bit Arithmetic Logic Unit (ALU) is designed with these LUTs and found that power dissipation is reduced by 40%. The control signal is given in such a way that the non-used blocks of the design are completely turned off by using Power gating technique and the leakage power is reduced by using MTCMOS technique. LUTs are designed in CADENCE Virtuoso Schematic Editor and simulated for verification in Analog Design Environment.

Keywords

FPGA, LUT, Control Unit, Leakage Power, MTCMOS.

Introduction

FPGA is a field programmable Integrated Circuit (IC) that can be programmed after fabricating the device. As per the Hardware Description Language (HDL) code the gates involved in the design are connected. Devices that consume very less power and reconfigurability are most important in new programmable devices. FPGAs cannot be used in mobiles as they have limitations because of their high-power consumption. Compared to Application-Specific Integrated Circuit (ASIC) implementations, FPGA implementations have more power overhead. A very high speed I/O and data buses are used in FPGA designs therefore verification of setup time and hold time is a complex task. During floor planning the resources allocation within FPGA take place and time constraints are met. FPGAs can be used to implement any logical function that an ASIC could perform. FPGAs hold considerable promise as fast to market replacement for ASICs in many applications. Even after applying low power design techniques, the power consumption of FPGAs remains unaffordable. Earlier techniques to reduce the power dissipation of FPGAs, focused on both dynamic and static power of these devices [1]. In hand-held devices like mobile phones the leakage power will be even more significant, since these devices are often used in an 'always on' state, remaining idle excluding for short bursts of activity [2].

ICs reduced system complexity, manufacturing cost and improve performance. They are relatively very expensive to develop and time to market is more. FPGAs were introduced as substitute to custom ICs for implementing entire system on a single chip and to provide the convenience of re-programmability to the designer and user. FPGAs can offer high density designs compared to discrete small and medium scale components. FPGAs take small amount of time for implementation with the help of Computer Aided Design (CAD) tools. Time taken for Physical design and mask preparation steps can be saved in case of FPGAs.

In general, three types of manufacturing Full-Custom, Semi-Custom and ASIC are considered. In Semi-Custom design, only gates effecting the overcritical of the circuit are pre-established and remaining gates can be re-configured as per the customer's choice. Full-Custom and Semi-Custom designs provide logic blocks for re-configuration. Logic blocks contains Look-Up Tables (LUTs) for implementing required logic. Figure 1 shows design flow used in FPGAs.

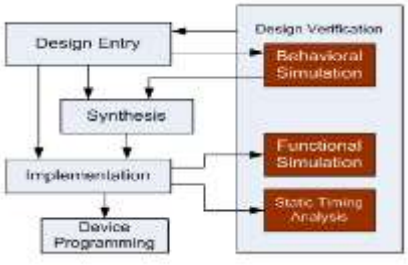


Figure 1: FPGA design flow

The objective of this work is to design configurable LUTs with minimum leakage power in FPGAs by introducing control logic for re-configurable LUTs. Re-configuration increases the

performance of FPGA in-terms of power and speed. An Arithmetic Logic Unit (ALU) is designed and verified using these re-configured LUTs.

II. FIELD PROGRAMMABLE GATE ARRAY

A. Architecture of Field-Programmable Gate Array

FPGA contains a sea of gates and switch matrix. When configured, the internal logic blocks of FPGA are connected in a way that creates a hardware implementation of the HDL code written. As in Processors, FPGAs do not have an operating system. Pre-existing fabricated hardware present in FPGA is routed as per the logic to connect the gates. Different processing operations of FPGA use resources and do not have to share any of the resources. Hence, the performance of one module of the application is not affected when additional processing is added. FPGA can run multiple control loops on a single device at different speeds. FPGA-based control systems can impose critical interlock logic. Hard-wired Printed Circuit Board (PCB) designs have fixed hardware resources; whereas FPGA-based systems can accurately rewire their internal blocks to allow reconfiguration after the control system is arranged to the field. FPGA devices assure the performance and reliability of dedicated hardware circuitry.

A single FPGA can replace thousands of discrete components by incorporating millions of logic gates in a single Integrated Circuit (IC). The internal resources of an FPGA chip consist of a matrix of Configurable Logic Blocks (CLBs) surrounded by a periphery of I/O blocks as shown in figure2. Inputs and interconnects are routed within the FPGA matrix by programmable interconnect switches. In an FPGA logic blocks are implemented using multiple level low fan-in gates, which gives it a more compact design compared to an implementation with two-level AND-OR logic. FPGA provides its user a way to configure, the intersection between the logic blocks and the function of each logic block. Logic blocks present in an FPGA can be configured such that it can provide functionality of a simple transistor to a complex microprocessor.

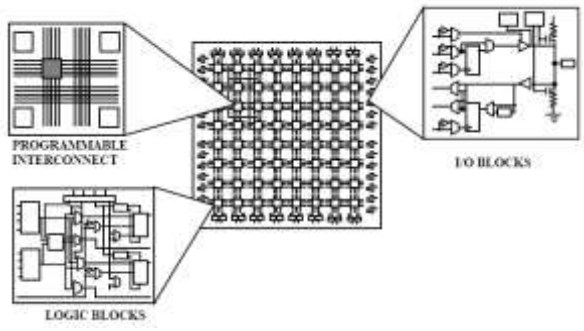


Figure 2: Architecture of FPGA.

Implementation of various combinational and sequential logic functions are conceivable on FPGAs. The Configurable Logic Blocks of an FPGA can be implemented as, transistor pairs, combinational Logic functions realised using basic NAND or XOR gates, multiplexers, N-input look-up tables and large fan-in AND-OR structures.

Wire segments of various lengths present in FPGAs are used for routing. These are interconnected through electrically programmable switches of switch box. Length and number of wire segments used for routing decides the density of logic block used in an FPGA. Number of segments utilised for interconnection classically is a trade-off between density of logic blocks utilised and amount of area utilised for routing.

b. Xilinx Logic Block.

Xilinx was the first company to introduce FPGAs. Look-Up Tables are used in Xilinx logic block to implement various functionalities. A k-input LUT based logic block can be implemented in several ways with trade-off between performance and logic density.

SPARTAN series: SPARTAN series is the huge production FPGA by Xilinx. SPARTAN series consists of hundreds to thousands of logic blocks and much more in new series. 300 logic blocks only were present in the initial SPARTAN series. Later series of SPARTAN 3A, 3 and 6 has 1.5K, 1.7K and 3.8K logic blocks respectively. Implementation of combinational and sequential circuits utilises the CLBs which are important logic resources, Switch matrix connects each logic block for an access to the general routing matrix. A CLB of SPARTAN6 consists of a pair of slices. These two slices do not have direct connections to each other and each slice is organized as a column. In every CLB, the slice in the bottom of the CLB is named as SLICE(0), and the slice in the top of the CLB is named as SLICE(1) as shown below in figure 3.

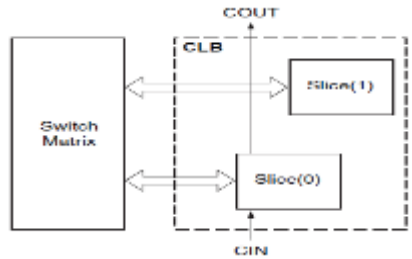


Figure 3: Slices is SPARTAN6 FPGA.

Four logic-function generators and eight storage elements are present in every slice. These elements are used by all slices to provide logic and Read Only Memory (ROM) functions. SLICE-X is the basic slice. SLICE-Ls contain an arithmetic carry structure that can be connected vertically up through the slice column and wide function multiplexers. The SLICE-M contains the carry structure and

multiplexers. SLICE-M enhances the capability to use the LUTs as 64-bit distributed RAM and as variable-length shift registers (maximum 32-bit). Each column of CLBs contains two slice columns. One column is a SLICE-X column, the other column alternates between SLICE-L and SLICE-Ms. Thus, approximately 50% of the existing slices are of type SLICE-X, while 25% each are SLICE-L and SLICE-Ms. SLICE-X, SLICE-L and SLICE-M shown in figure 4 (a),(b) and (c).[2]

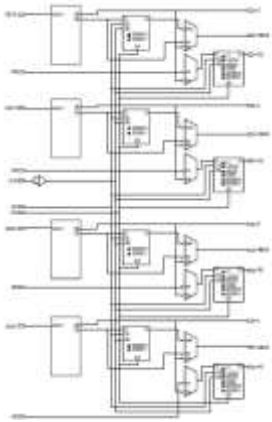


Figure 4(a): SLICE-X

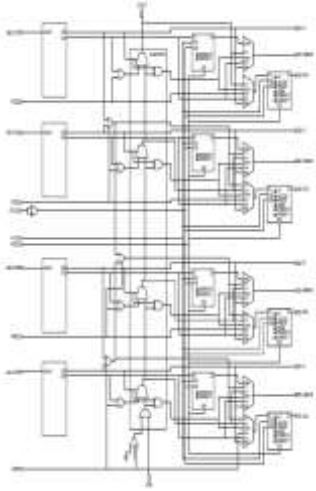


Figure 4 (b): SLICE-L

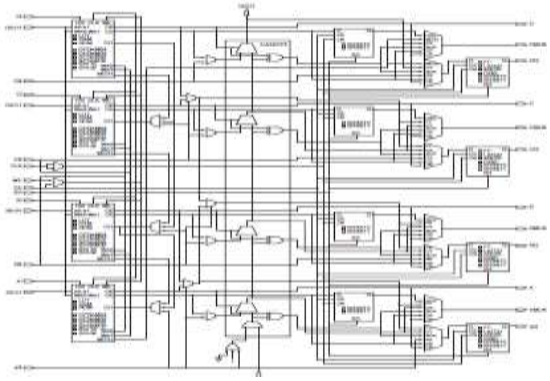


Figure 4(c): SLICE-M of SPARTAN-6.

III. DESIGN OF MINIMUM LEAKAGE RE-CONFIGURABLE LOOK-UP TABLES FOR FPGA

a. Look-Up Table (LUT): The function generators in SPARTAN-6 FPGAs are implemented as six-input Look-Up Tables (LUTs). There are six independent inputs (A inputs - A1 to A6) and two independent outputs (O5 and O6) for each of the four function generators in a slice (A, B, C, and D).

The function generators can implement any arbitrarily defined six-input Boolean function. Each function generator can also implement two arbitrarily defined five-input Boolean functions, if these two functions share common inputs. Only the O6 output of the function generator is used when a six-input function is implemented. Both O5 and O6 are used for each of the five-input function generators.

A logic block commonly used in FPGA devices is the look-Up Table (LUT). An LUT contains storage cells that are used to implement small logic functions. Each cell can store a single logic value (0 or 1). Multiplexers are used to select one of the storage cells for output. Essentially, the cells store the truth table for a function and the multiplexers select a cell for output based on a set of select (control) inputs. Look-up table is simply an initialized array that contains pre-calculated information. They are typically used to avoid performing complex (and hence time consuming) calculations.

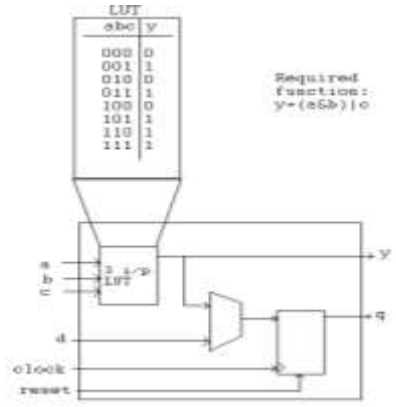
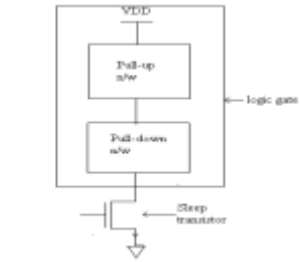


Figure 5: Look-up table

A look-up for a required function is based on the truth table of the function. In Figure 5, the function required has three inputs, so that there are eight possible outputs. Hence, for this function, eight storage cells are required and an 8X1 multiplexer is used for selecting the outputs.



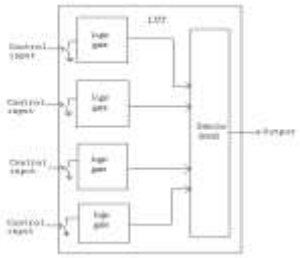


Figure 6: Modified logic gate **Figure 7:** Design of re-configured LUT

The re-programmability of FPGAs is their major advantage. FPGAs are also not suitable for applications such as mobiles, PDAs etc., for their high power dissipation. In FPGAs power dissipations is due to LUTs and their routing architecture. So, the traditional structure of a basic CMOS logic gate and LUT are re-configured and are shown in figures 6 and 7. These re-configured LUTs contain logic gates implemented in CMOS and the transistor level pull-down network is connected to a pin instead of a ground terminal. A sleep transistor is used between the pin and the ground terminal as shown in figure 6. The double benefit of the sleep transistor is that it enables ON and OFF of logic gate whenever necessary and reduces the leakage power during stand-by mode. The LUT contains logic gates and a multiplexer (MUX) to select one of the outputs from logic gates. The selection lines of the MUX are used to select the logic gate outputs[3-5].

IV. Design of ALU with Re-Configurable Look-Up Tables

The Arithmetic Logic Unit (ALU) is one of the main components of any microprocessor. It is responsible for performing arithmetic and logic operations such as addition, subtraction, increment, decrement, logical AND, logical OR, logical XOR and logical XNOR as shown in table 1. ALU is designed based on Full-adder structure. For processing one single bit in ALU, a full-adder along with three multiplexers is used as shown in figure 8. A 4-bit ALU is implemented with and without re-configurable LUTs and power consumption is calculated[6].

Table 1: Operations of ALU.

Selection	Operation
000	Increment
001	Subtraction
010	Addition
011	Decrement
100	AND
101	XNOR
110	XOR
111	OR

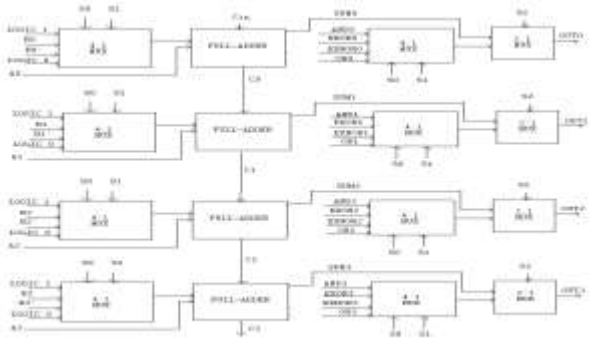


Figure 8: Block diagram of a 4-bit ALU

A 1-bit ALU has two 4X1 multiplexers, one full-adder and one 2X1 MUX. All these individual blocks are extracted from LUTs. Normally, while implementing ALU using FPGA, each gate from LUTs is taken and logic is performed. But, in this design, Re-Configurable LUTs (RC LUTs) are used with control inputs only to switch on required logic gate and remaining logic gates are maintained in the off state as shown in figure 9. Individual logic gates are turned OFF using a control transistor (sleep transistor) at the ground terminal. The gate of the control transistor is considered as control input. The control input will be 1 or 0. If the control input is 1, as the transistor is NMOS, the logic gate is connected to the ground terminal and it functions normally. If the control input is 0, the logic gate is disconnected from the ground terminal and logic gate stops its functioning.

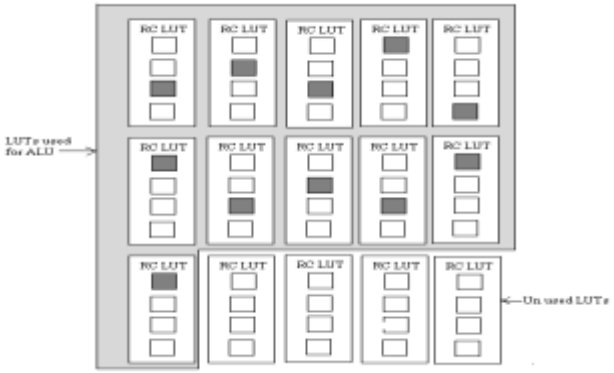


Figure 9: Block diagram of LUTs used in ALU

2 input and 5 input LUTs are designed with and without re-configuration. A 4-bit ALU which performs arithmetic operations of addition, subtraction, increment, decrement and logical operations of AND, OR, XOR, XNOR designed using normal LUTs and with RC LUTs. The implementations are done in CADENCE Virtuoso Schematic Editor (Version IC 6.1.4). Simulations of schematics are performed in CADENCE Virtuoso ADE (Analog Design Environment) and libraries used are gpdk45 and analoglib. Operating system required is LINUX.

a)Two Input Look-Up Tables: Look-up tables are type of memory units which store the pre-calculated values of a particular function. LUTs greatly reduce the time needed for calculation and increase the operation speed. A 2 input LUT which can perform the functions of XOR, NAND, AND and OR is designed. All these gates are connected to a 4X1 multiplexer. The selection lines of MUX are used to select one of these 4 functions as shown in figures 10(a) and the with minimum leakage design with sleep transistor is shown in figure 10(b). This ground pin is connected to NMOS transistor for controlling. The gate of NMOS acts as control input (C) as shown in figure 10(b).

Figure 10(a): 2 input LUT (b) 2 input LUT with control

b) Design of functions using re-configurable LUTs: Many LUTs combine to perform a single operation in FPGA. Two functions, F1 (half-adder: XOR & AND gates) and F2 (general function: NAND & or) are implemented. With normal LUT operation, when any one of these functions is selected, all the gates of LUT are turned ON. While using control logic, only one logic gate is turned ON as shown in figures 11(a) and (b). A MUX is used to select between the two functions F1 and F2.

Figure 11(a): Function F1 (b)**:** Function F2 with new LUT

When F1 is selected, the output of MUX will be 1. This logic 1 will turn ON the AND gate. The output of AND gate is connected is directly connected to c1 of LUT1 and c3 of LUT2. So, when inputs are given, only XOR gate of LUT1 and gate of LUT2 are turned on and remaining gates are turned off. When MUX selects the function F2, XOR gate is turned on and similarly, NAND gate in LUT1 and or gate in LUT2 are turned on. As, only required gates are ON, the power consumed is greatly reduced.

c) 5 Input Look-Up Tables: LUTs with multiple functions with eight complex functions using 5-inputs is implemented. A 8X1 MUX is used to select one of these 8 different functions. A 5-input LUT of eight different functions is shown in figure 12 (a) & (b). When 5inpus are given, all the eight functions start working, even-though the required function is only one. This leads to unnecessary power consumption.

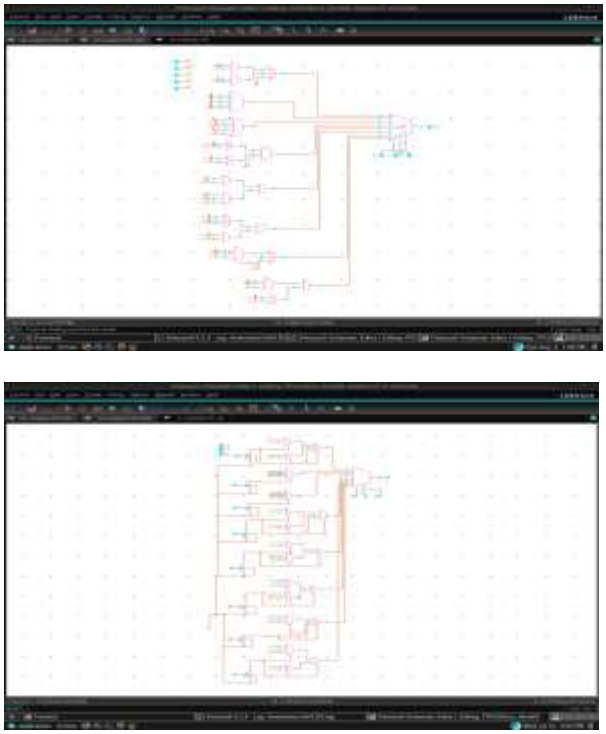


Figure 12(a): 5 input LUT **(b):** 5input RE-configurable LUTs

d) Arithmetic Logic Unit

Full-adder is a combination of two half-adders. A general full-adder contains two XOR gates, two AND gates and an OR gate. Full-adder shown in figure13(a) contains an extra OR gate and NOT gate. All the logical operations are performed in this full-adder only. Depending upon the inputs even arithmetic operations are performed in full-adder. The outputs are taken at appropriate gates. Figure 13(b) shows Full-adder with control inputs.

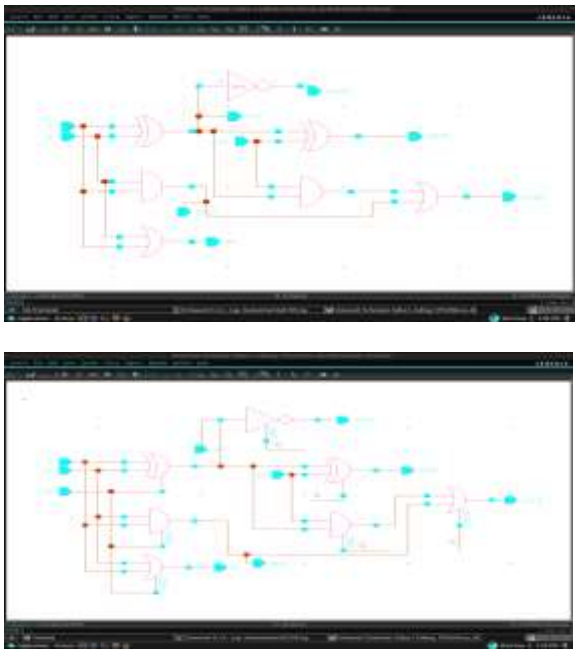


Figure 13(a): Full-adder **(b):** Full-adder with control inputs.

The selection lines used s0,s1 and s2 which are used in the design, determines the operation performed by ALU. The carry in one stage is cascaded to another stage. The 4-bit ALU consist of eight 4X1 MUX, four 2X1 MUX and four full-adders. For increment and decrement operations logic '1' and logic '0' are applied inputs respectively. The complement of B is used for subtraction operation. The full-adder performs the subtraction operation by two's complement method. An increment operation is analyzed as adding '1' to addend and decrement is a subtraction operation. The outputs from the full-adder are SUM, XOR, XNOR, AND & OR. Based on condition of select signals, the MUX before full-adder will selects the appropriate inputs and gives it to full-adder. The full-adder computes the results. The MUX after the full-adder selects the appropriate output and sends it out.

When an n-bit ALU is used for (n-x) bit operation (where 0<x<n), all the modules in ALU will start working even though the inputs to certain modules is zero. This leads to excess power consumption. When a 4-bit ALU is used for 2-bit applications, the power consumption is same for both 4-bit and 2-bit operations. In such case ALU with control inputs will reduce the power consumption to greater extent. In ALU with control inputs, for 2-bit operation, only two modules are ON and remaining modules are OFF using control inputs. Schematics of 4 bit ALU with control logic is shown in figure 14(a) & (b).

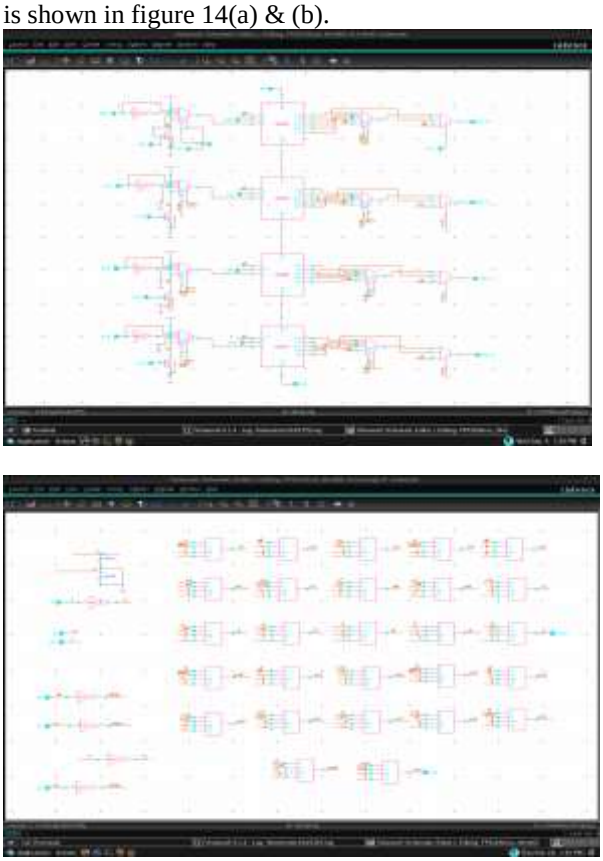
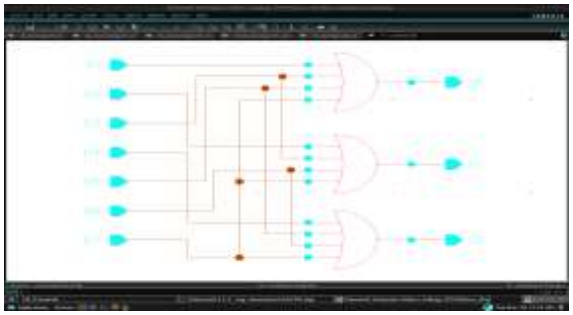


Figure 14(a): 4-bit ALU (b): ALU using normal LUTs

LUTs are basic blocks of every FPGA. In every LUT a single gate is used. Group of LUTs form a single block or a single module. For example, a 2X1 MUX require three NAND gates and a NOT gate, a 4X1 MUX require nine NAND gates and three NOT gates. Similarly, a full-adder require two XOR gates, two AND gates, two OR gates and a NOT gate. All these gates are taken from LUTs. Four such blocks are combined to get a 4-bit ALU.



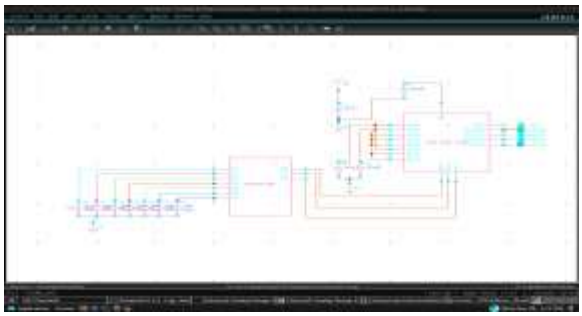


Figure 15(a): Control unit **(b):** ALU with control unit.

In above schematics, the selection lines for functioning of ALU are given manually. A control unit is designed such that it takes eight inputs and gives the required 3-bit output. The eight inputs are used to select the four arithmetic and four logical operations. The schematic of control unit is shown in figure 15(a). The schematic of ALU using re-configurable LUTs is shown in figure15(b). The inputs are a, b, c, d, e. Let f1, f2, f3 and f4 be four functions. The function, f1 requires only third function of LUT to be performed. So, only third function in LUT is made ON and remaining functions are made OFF by using control input 'c3=1'. Similarly, c4=1 for f2, c1=1 for f3 and c6=1 for f4. For an instant at which s2s1s0 are 010 respectively, then fifth input of 8X1 is selected i.e., function [(a+b)(c+d+e)] is selected. For an instant at which s2s1s0 of first and second LUTs are 001 and 100 respectively, then second input and third input of 8X1 MUX are selected i.e., functions [a+b+c+d+e] and [abcde] are selected. Hence, in a single module, there will be 29 LUTs. For a 4-bit ALU four such modules are needed. Total of 116 LUTs are required. The remaining operation is same as that of normal ALU. The power consumption of the circuit is 4.239µW.

Results and Discussion

All the blocks are designed using re-configurable LUTs. The control inputs for re-configurable LUTs are given directly based on which gate is necessary. The simulation result is shown in figure 5.16. Total of 116 LUTs are required. The remaining operation is same as that of ALU using control logic. The power consumption of the circuit is 2.065µW.

Figure 5.16: Simulation result of ALU using re-configurable LUTs and control unit.

Control Logic for 4-bit ALU using re-configurable LUTs prove that power is saved by power gating and sleep transistor for leakage control. All these designs are simulated and verified in Analog Design Environment.

Table 2: Comparison of % of Power Savings

Design Modules	Power without control logic	Power with control logic	% Power savings
2 Input LUT	513.2nW	379.5nW	26.05
2 i/p LUTs	790.3nW	532.9nW	32.5
5 i/p LUT	667.0nW	392.4nW	41.16
5 i/p LUTs	840.5nW	309.9nW	63.12
4-bit ALU	343.3nW	148.8nW	56.65
ALU using LUTs	4.239µW	2.065µW	51.28
ALU using control unit	2.065µW	1.201µW	41.84

Conclusion

The power consumption which is the key factor in the design of FPGA is greatly reduced by using re-configurable LUTs driven by control logic. Both dynamic and leakage power is reduced by using re-configurable LUTs. Among the pool of LUTs present in FPGA, when re-configurable LUTs are used along with control logic, only the required LUTs are ON and remaining LUTs are maintained in OFF position by using control inputs. This reduces the dynamic power consumption. During standby mode of the design, the LUTs remain off by a sleep transistor which is present between pull-

down circuit and the ground terminal resulting the further reduction of the leakage power. Presence of sleep transistor for every logic gate will increase the area of the circuit at same time reducing power consumption by more than 40%. This type of logic where only required logic gates are used while others remain in unused state, is very useful in systems where power consumption is the dominating factor. The control input for sleep transistor may be applied by scripting tools with supporting algorithms which can ease the control input to the LUTs by programmed logic to the FPGA.

Acknowledgement
The authors would like to express their deep sense of gratitude to the UGC for providing funds and the management of the college for the infrastructural facilities and laboratory.

References

[1] Naresh Grover and Dr. M.K.Soni, " Reduction of Power Consumption in FPGAs - An Overview" published in I.J. Information Engineering and Electronic Business, 2012.

[2]Assem A. M. Bsoul and Steven J. E. Wilton, "An FPGA architecture supporting dynamically controlled power gating" published in Field-Programmable Technology (FPT), December, 2010 International Conference.

[3] Sujata Prajapati, Prof. M. Zahid Alam and Dr. Rita Jain, "New Approach to Low-Power & Leakage Current Reduction Technique for CMOS Circuit Design" published in ISSN : 2248-9622, Vol. 4, Issue 2(Version 1), February 2014, pp.612-617.

[4] T. Esther Rani and Dr. Rameshwar rao, "Design of minimum leakage multipliers using MTCMOS Technique" published in 2011 3rd International Conference on Electronics Computer Technology (ICECT 2011).

[5] Velicheti Swetha1, S Rajeswari, "Design and Power Optimization of MTCMOS circuits using Power Gating Techniques" published in International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering (An ISO 3297: 2007 Certified Organization) Vol. 2, Issue 8, August 2013.

[6] T. Esther Rani, M. Asha Rani and Dr. Rameshwar rao, " Area optimized low power arithmetic and logic unit" published in 2011 3rd International Conference on Electronics Computer Technology (ICECT 2011).

