Applied Mechanics and Materials Vols. 241-244 (2013) pp 2548-2554 *Online: 2012-12-13*
© (2013) Trans Tech Publications, Switzerland
doi:10.4028/www.scientific.net/AMM.241-244.2548

Efficient utilization of FPGA using LUT-6 Architecture

Razia Zia[a], Muzaffar Rao[b], Arshad Aziz [c], Parvez Akhtar[d]

National University of Sciences and Technology, Pakistan

[a]raziamaroof@yahoo.com, [b]rao_muzaffar@yahoo.com, [c]arshad@nust.edu.pk ,
[d]pervez@pnec.edu.pk

Keywords: FPGA, LUT-6 Architecture

Abstract. Field Programmable gate array (FPGA) technology is continuously gaining market share and becoming essential part of the today's modern embedded systems. The most common FPGA architecture consists of an array of logic blocks called Configurable Logic Block (CLB), I/O pads, and routing channels. In general, a logic block (CLB) consists of logical cells called Slices and other dedicated resources. A typical cell consists of LUTs (Look up table). In modern FPGAs, there are 6-input LUTs instead of 4-input LUTs. In this paper we present the use of 6-input LUT architecture for some Boolean functions (Mux8, Mux16, Mux32, Mux64, SOP64, OR40 and AND40).we show our results in terms of LUTs and Slices and these results are much better as compare to previously reported results that based on 4-input LUTs.

Introduction

FPGAs are reprogrammable silicon chips. Using prebuilt logic blocks and programmable routing resources, these chips can be configured to implement custom hardware functionality. The number of gates and other resources are increasing dramatically to compete with capabilities that have normally been offered through ASIC devices only. Modern FPGA devices have progressed both in terms of resources and performance. Now latest devices have come to provide "platform" solutions that are easily customized for system connectivity, DSP, and/or data processing applications [1].FPGAs have traditionally been configured by hardware engineers using a Hardware Design Language (HDL). The two principal languages used are Verilog HDL and Very High Speed Integrated Circuits (VHSIC) HDL (VHDL) which allows designers to design at various levels of abstraction. The growing use of reconfigurable devices makes it very important to develop techniques to effectively and efficiently utilize the internal resources of these devices [2].4-input Look up table (LUT) based architecture is used by conventional FPGAs to implement logic. The disadvantage of using this approach is the utilization of more area on the chip.

An n-bit LUT can encode any n-bit Boolean function by modeling such functions as truth tables. This is an efficient way of encoding Boolean logic functions. Modern FPGAs (Spartan-6, Virtex-6 and Virtex-7) used 6-input look-up tables (LUTs) to implement logic.

In this paper we proposed LUT-6 architecture to further optimize our previously reported results [9].The architecture that we used is LUT-6 architecture and the results are much better in terms of Slices and LUTs.The remainder of this paper is organized as follows. We briefly give an overview about architecture of FPGA in Section II. Section III gives brief description about LUT-4 based architecture. In Section IV we present the optimized technique using LUT-6 based architecture. In section V we give the results of our work and in section VI compare it with previous implementation. Finally, we provide some conclusion in Section VII.

Architecture of Xilinx FPGA

Xilinx FPGAs as shown in figure 1 contain a large number of programmable logic blocks or configurable logic blocks called CLB. By means of appropriate SRAM programming cells each logic block in the device could be configured to perform a different function. In addition to LUT, MUX and register each logic block contains a smattering of other elements including some special fast carry logic for use in arithmetic operations[4].

A CLB is a basic building block containing logic cells. As CLBs are connected through Switch matrices (SMs) which are programmable interconnects of wire segments. The CLBs are organized in a grid array to implement different type of logic designs i.e. combinational and synchronous.

Spartan-3 and Virtex-4 FPGAs includes 4-input LUTs. Now modern FPGAs like Spartan-6 Virtex-5,6 and 7 also includes 6-input LUTs. These 6-input LUTs can be used to optimize the designs. These 6-input LUTs with other resources available in each CLB can be used to improve the performance, density and size of wide logic that can be implemented in each CLB. There are two slices in each CLB and each slice is organized in a column as shown in figure 2. There are three types of slices SLICEX, SLICEL & SLICEM. Slices called SLICELs also contain an arithmetic carry structure that can be concatenated vertically up through the slice column, and wide function multiplexers. The SLICEMs contain the carry structure and multiplexers, and provide the ability to use the LUTs as 64-bit distributed RAM and as variable-length shift registers (maximum 32-bit).

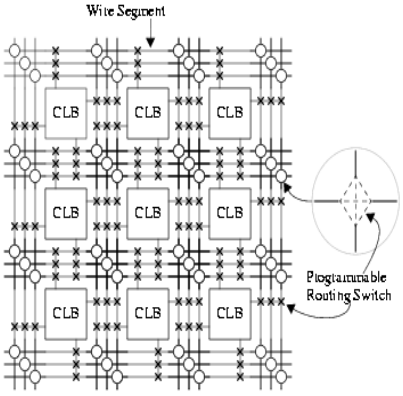

Figure 1: General block diagram of FPGA

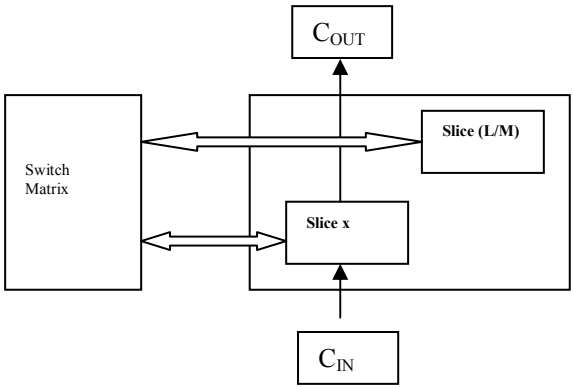

Figure 2: Arrangement of Slices within CLB

There are two slice columns in each CLB. One column is a SLICEX column, the other column alternates between SLICEL and SLICEMs. Thus 50% of the available slices are of type SLICEX , 25% are of SLICEL and 25% are of SLICEMs. In modern FPGAs every slice contains four logic-function generators or look-up tables (LUTs) and eight storage elements. These elements are used by all slices to provide logic and ROM functions [5].

LUT-4 based architecture

In conventional design approach LUT-4 based architecture is used which results in consumption of bigger chip area and longer input to output path delays. So, the design becomes bigger and runs at slower clock rates.

Consider an example of 16-input XOR gate .Simply code it using HDL instructions. If ‘out’ is the output of XOR gate and ‘a’ is the input variable then we can implement this example in verilog HDL using following command.

```
Assign out = a[0]^a[1]^a[2]^a[3]^a[4]^a[5]^a[6]^a[7]^a[8]^a[9]^a[10]^a[11]^a[12]^a[13]^a[14]^a[15];
```

The above instruction performs the logical XOR of the 16 input variable ‘a’ and the output goes to ‘out’. Five 4-input LUTs are used to implement this logic as shown in Figure 3. The first four LUTs perform the XOR operation on three 4-bit groups of input and then resulting three bits will be XORed using a fifth 4-input LUT. In next section we will show that how these number of LUTs can be minimized using 6 input LUT architecture.

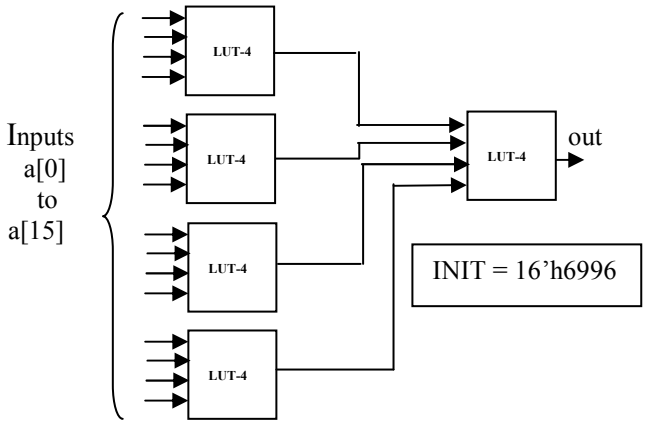

Figure 3: 16-input XOR function using LUT-4

Optimized approach (LUT-6 based architecture)

In 16 input XOR gate we require five 4 -input LUTs. In modern FPGAs (Spartan 6, Virtex 5, Virtex 6 and Virtex 7) we have the option of 6-input LUTs. By using these LUTs we can improve our results in term of LUTs and Slices. Following is an example code fragment using LUT-6 in place of LUT-4. Here we have implemented 16-input XOR gate using only three LUT6 thus saving two LUTs. Now we will discuss the proposed architecture implementation for wide input Boolean functions.

```
 LUT6 #(                          LUT6 #                    LUT6 #
.INIT(64'h699696696696996)       .INIT(64'h699696696696996)  .INIT(64'h699696696696996)
  LUT6_inst_1 (                   LUT6_inst_1 (               LUT6_inst_1 (
     .O(O1),                      .O(O2),                     .O(O2),
     .I0(a[0]), // LUT input      .I0(a[6]), // LUT input      .I0(a[6]), // LUT input
     .I1(a[1]), // LUT input      .I1(a[7]), // LUT input      .I1(a[7]), // LUT input
     .I2(a[2]), // LUT input      .I2(a[8]), // LUT input      .I2(a[8]), // LUT input
     .I3(a[3]), // LUT input      .I3(a[9]), // LUT input      .I3(a[9]), // LUT input
     .I4(a[4]), // LUT input      .I4(a[10]), // LUT input     .I4(a[10]), // LUT input
     .I5(a[5])  // LUT input      .I5(a[11])  // LUT input     .I5(a[11])  // LUT input
   );                              );                          );
```

A. Wide input AND & OR operation

As in 16-bit AND gate using 4-input LUT we require five 4-input LUTs. Using LUT-6 we require just three LUTs for the same task hence saving 2 LUTs. Similar if we implement Boolean function of 40-input AND Gate used only 8-LUTs when implemented using LUT-6 architecture as shown in figure 4. Same architecture can be used to implement 40-input OR Gate.

```
Assign out  =  a[0]^a[1]^a[2]^a[3]^a[4]^a[5]^a[6]^a[7]^a[8]^ a[9]^a[10]^…….a[37]^a[38]^a[39];
```

B. Wide input multiplexers

Modern Xilinx FPGAs also contain MUXFX multiplexers dedicated for the design of wide input multiplexers. In addition to the basic LUTs, SLICEL and SLICEM contain three multiplexers (F7AMUX, F7BMUX, and F8MUX). These multiplexers are used to combine up to four function generators to provide any function of seven or eight inputs in a slice. F7AMUX and F7BMUX are used to generate seven input functions from a slice while F8MUX is used to combine all slices to generate eight input functions. With more than eight inputs can be implemented using multiple slices. As a LUT4 can support a maximum of 2:1 MUX as shown in figure 5 .With LUT6 we can implement 4:1 MUX shown in figure 6.

We can implement 8 to 1 multiplexer using two 6-input LUTs and F7 Mux as shown in figure 7 while a slice can implement 16:1 Mux as shown in figure 8. In modern FPGAs (Spartan 6, Virtex 5, Virtex 6) each column of CLBs contain two slice columns. One column is a SLICEX column, the other column alternates between SLICEL and SLICEMs.

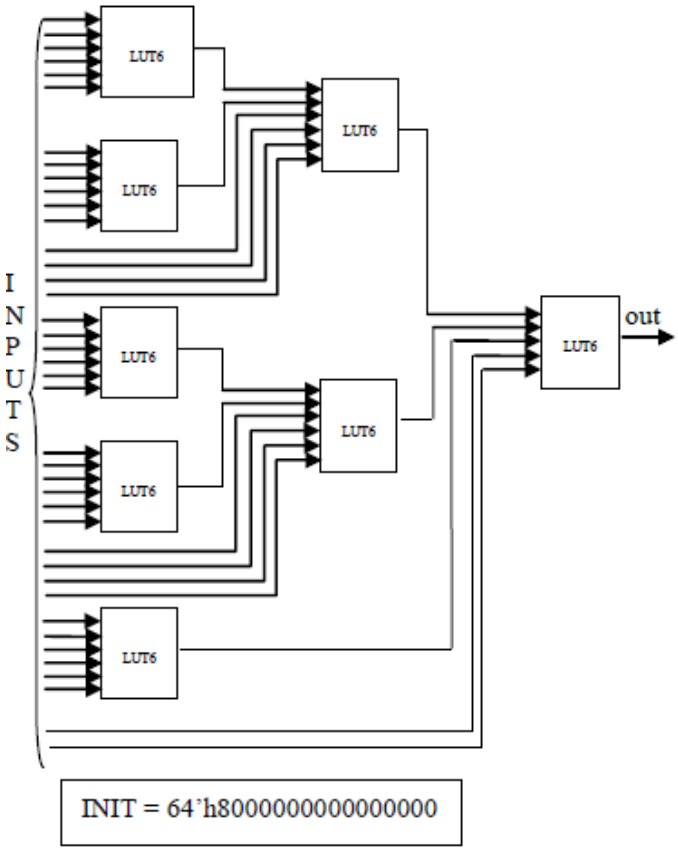

Figure 4: 40- input AND Gate using LUT-6

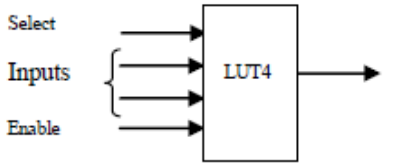

Figure 5: 4-input LUT as 2:1 Mux

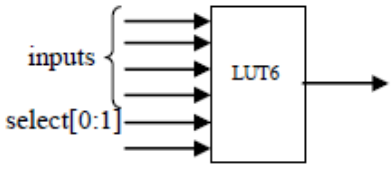

Figure 6: 6- input LUT as 4:1 Mux

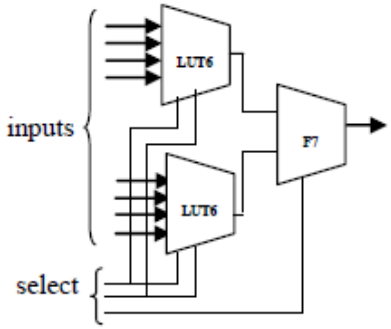

Figure 7: MUX 8-1 using LUT-6

C. Sum of Product Function (SOP64)

Figure 9 provides the function of 64-bit sum of product using 10 LUTs of 6-input LUT architecture. First and second stage of LUTs implement AND logic and the last LUT implement OR logic.

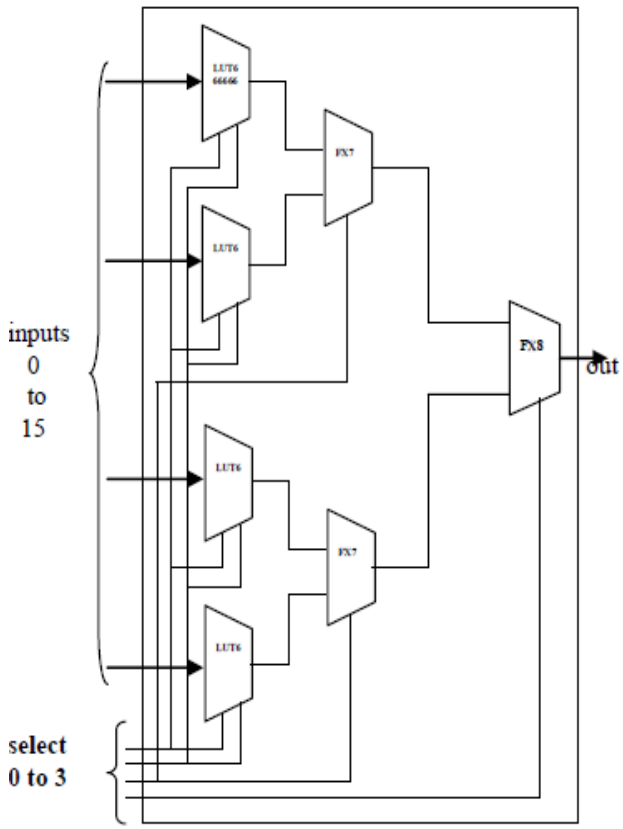

Figure 8: MUX 16-1 using LUT-6

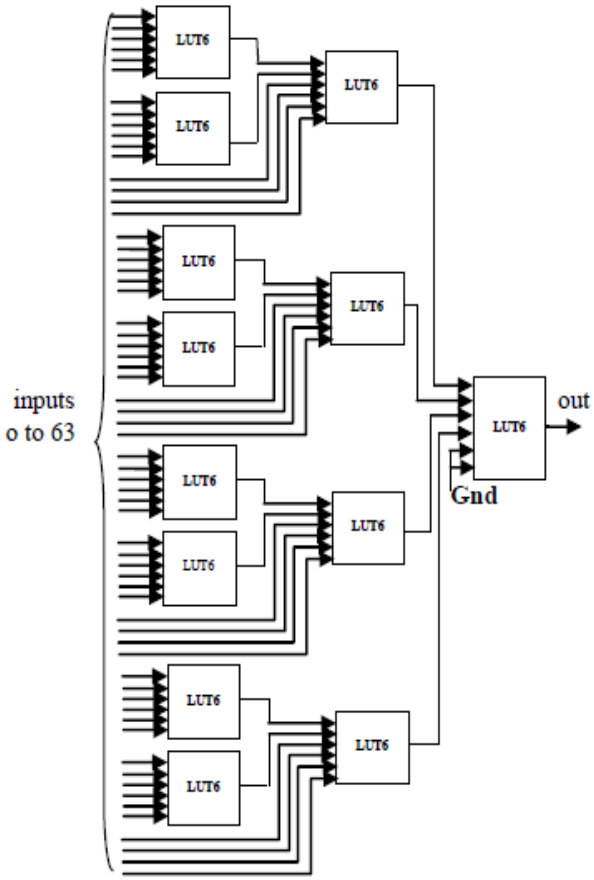

Figure 9: SOP64 using LUT-6

Implementation Results

The proposed architecture has been implemented on Xilinx spartan 6. Detailed device specification is: Virtex 6 LX75T, speed grade 3, package FF784 (6vlx75tff784-3). The hardware resources are given in terms of LUTs and slices. Table 1 shows the implementation results of some Boolean functions. We can observe that when Boolean functions are implemented using LUT-6 based architecture, there is a significant reduction in number of LUTs and Slices.

Comparison with previous work

Table 2 shows the comparison of results with previously reported implementations in terms of LUTs and Slices. Previously reported result is based on LUT-4 architecture. Kashif et. al. [9] discussed and reported their results for various architectures. For performance comparison, we considered some results of architecture to compare with our work. Graphical representation is also shown in figure 11.

Table: 01 Implementation results for different Boolean function architectures.

Architecture	Parameter	LUT4 design	LUT6 design	Improvement (%)
MUX8	LUTs	7	2	71.42
	Slices	4	1	75
MUX16	LUTs	15	4	73.333
	Slices	9	1	88.88
MUX32	LUTs	31	8	74.19
	Slices	18	2	88.88
MUX64	LUTs	63	17	73.01
	Slices	36	5	86.11
SOP64	LUTs	23	13	43.47
	Slices	13	9	30.76
AND40	LUTS	13	8	38.46
	Slices	7	2	71.42
OR40	LUTs	13	8	38.46
	Slices	7	2	71.42

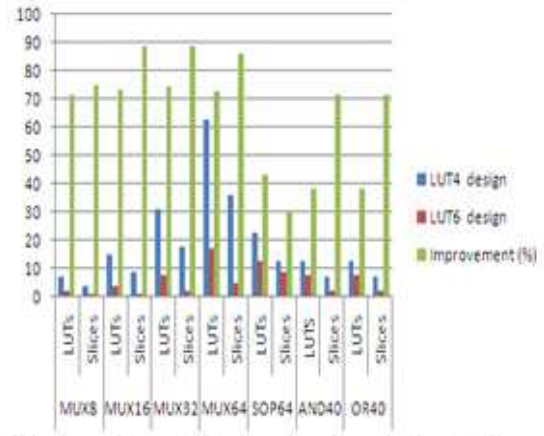

Figure 10: Comparison of Boolean Function implementation using LUT-6 and LUT-4 Architecture

Table: 02 Comparison with previously reported results

Author(s)	MUX8		MUX16		MUX32		MUX64		SOP64		AND40		OR40	
	L	S	L	S	L	S	L	S	L	S	L	S	L	S
ThisWork	2	1	4	1	8	2	17	5	13	9	8	2	8	2
Kshif Latif et al. [9]	7	4	15	9	31	18	63	36	23	13	13	7	13	7

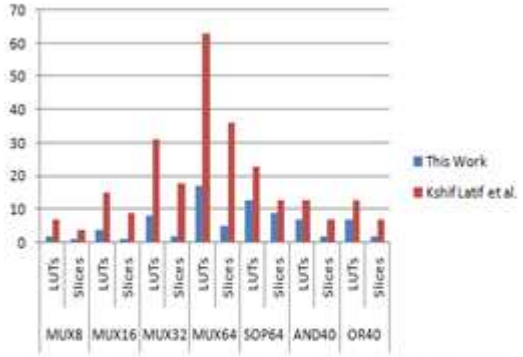

Figure 11: Comparison of Boolean Function implementation with previously reported results.

Conclusion

In this work we have presented efficient utilization of modern FPGAs using LUT-6 Architecture. We showed the implementation of some Boolean functions on Xilinx Spartan 6.We reported our implementation results in terms of area and compared it with previously reported implementation results that is based on LUT-4 Architecture. Our implementation architecture is much efficient and used minimum LUTs and Slices as compare to previously reported results.

REFERENCES

[1] National Instruments, Introduction to FPGA technology: Top Five Benefits, April 2012. P. 1-2. http://www.ni.com/white-paper/6984/en

[2] Advantages of FPGA design methodologies, EE Times: july 2004. http://www.design-reuse.com/articles/8404/advantages-of-fpga-design-methodologies.html

[3] BDTI Focus Report: FPGA for DSP. 2nd ed. BDTI Benchmarking; 2006.

[4] Latif Kashif, Aziz Arshad, Mahboob. Efficient resource utilization of FPGA. In:FIT'09 proceedings of sixth international conference on Frontiers of information Technology, ISBN:978-1-60558-642-7. New York, USA: ACM;2009. P.1-5.

[5] Xilinx, Spartan-6 FPGA CLB User Guide UG384 V1.1 February 23 2010. http://www.xilinx.com/.

[6] Xilinx, Project Navigator, View/Edit Routed Design (FPGA Editor), Xilinx ISE Design Suite 13.3.

[7] Clive "Max" Maxfield,The Design warrior's Guide to FPGA.

[8] Xilinx, Vertix-5 FPGA: Complete data sheet DS100 (v5.0) February 6, 2009 www.xilinx.com.

[9] Latif Kashif, Aziz Arshad, Mahboob. Optimal Utilization of available reconfigurable hardware resources. Computer and Electrical Engineering 37(2011). P. 1043-1057.

Industrial Instrumentation and Control Systems
10.4028/www.scientific.net/AMM.241-244

Efficient Utilization of FPGA Using LUT-6 Architecture
10.4028/www.scientific.net/AMM.241-244.2548