International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-816-823 ISSN: 2249-6645

Optimizing area of local routing network by reconfiguring look up tables (LUTs)

Sathyabhama.B[1] and S.Sudha[2]
[1] *M.E-VLSI Design*
[2]*Dept of ECE*
Easwari engineering college, Chennai-89

***Abstract* - The general way of mapping digital circuits onto field programmable gate arrays (FPGAs) usually consist of two steps. Initially the circuits are mapped into look up tables (LUTs). Then, the LUTs are mapped onto physical resources. This includes the process of reconfiguration. Reconfiguration follows three basic properties, which includes commutative property, duplicate-constant input property, and constant new input equivalence property. Logic blocks are composed of clusters with LUTs and flip flops. In particular for a logic cluster with I inputs and N K- input LUTs a set of N × K (I+N-K+1):1 multiplexers can be used to connect logic cluster input to LUT input. It can increase the flexibility of FPGA routing resources. The flexibility can then be used to reduce the implementation area. This can also reduce the significant amount of fanouts for logic cluster input. Reconfiguration can also be done in correspondence with logical non-equivalency which also tender to give better area efficient result.**

Index terms- Field programmable gate arrays (FPGAs), logical non-equivalency, logic cluster, reconfiguration.

I. INTRODUCTION

Look up tables (LUTs) are connected through two level routing hierarchy in FPGAs. Two level routing hierarchy includes local routing network and global routing network. LUTs are connected to logic clusters through local routing network and the logic clusters are connected to Field Programmable Gate Arrays (FPGAs) through global routing network. Routing hierarchy concentrates on flexibility and minimization of area. Logic block composed of basic logic elements (BLEs) which is connected with fast local interconnect. BLEs are generally indivisible unit with a combination of sequential and combinational logic [1]. In general BLEs consist of flip flops and LUTs. A logic block with one or more number of BLE is said to be a logic cluster. The flexibility of the routing network is increased when the logic cluster are used with logically equivalent inputs and outputs. In logical equivalency the input can enter a logic cluster through any of the input and the output can allow a signal to exit through any of the output pin. Thus, the flexibility of the routers are increased and it leads to the better utilization of routing resources. The logic cluster with logically equivalent input and output allow a signal to enter or exit in a several way. This added connectivity is used in increasing the flexibility of the routers.

This paper is organized as section II briefs about logic clusters with logic equivalency, section III explains LUT structure and properties of LUT, section IV deals with routing flexibility based on the reconfiguration of the LUT, section V comprised of sparse network with the result of the reconfiguration, section VI deals the non-equivalency of the same network which is previously implemented with logical equivalency, section VII gives all the simulation and synthesis results for both the network with logically equivalent and logically nonequivalent.

II. LOGICALLY EQUIVALENT LOGIC CLUSTERS

Logical equivalency of the network can be attained through the fully connected network [2]. Fully connected network is configured as several cluster inputs connected to the number of LUT present in the network through the multiplexers. All the input of the clusters are connected to each and every input of LUT without any merging and coincidence through the multiplexers. Since the previous work describes the fully connected network is not the most area efficient method to attain logical equivalency, the current work goes with the LUT reconfiguration. Sometimes attaining logical equivalency through the fully connected network after implementation results in less routing tracks when compared to the logically non-equivalent. There is a tool available for checking the logical equivalency named LEC (Logic Equivalence Checker). Test patterns are not required for LEC instead it will use Boolean arithmetic technique to prove the equivalency. The network for this work has a logic cluster with two LUTs and the number of cluster input is derived from 2^{k-1}, where k is the number of inputs to the LUT. The cluster size can be varied with varying the number of input to the LUT. The logic cluster input is named as I. Sometimes the output of the LUT is again given to the input of the logic cluster as feedbacks. In this network N represents the feedback given to the logic cluster. The cluster can be initiated by applying inputs, through the multiplexers the function of the network can be changed.

A fully connected local routing network is used to connect the logic cluster inputs to each LUT input in all possible ways it can. The basic fully connected local routing network taken for this work is given below

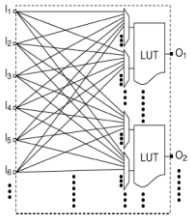


Fig.1. Fully connected local routing network.

A k-input LUT is designed to emulate the operation of a 2^k entry truth table. The LUT is constructed out of a 2^k :1 multiplexer and 2^k bits of configuration memory [3]. The memory is connected to the data inputs of the multiplexer and stores the truth table entries. The LUT inputs are connected to the select inputs of the multiplexer.

III.STRUCTURE AND PROPERTY OF LUT

Three properties of an LUT can be used to determine the minimum area required to implement a logic cluster containing logically equivalent input and outputs. The three main basic property [2] used for this work are

- Commutative property,
- Duplicate-constant input property,
- Constant new input equivalence property.

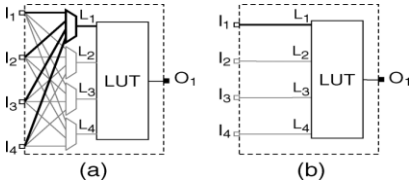


Fig.2. Fully connected network versus LUT reconfiguration.

The LUT inputs are connected to the select inputs of the multiplexer. For a four-input Boolean function such as the one shown in Table.1.1(a), a signal assigned to L_1 can be routed through cluster input I_1. The same function, can be implemented by exchanging the signal assignment of L_1 and L_2 and by reconfiguring the LUT to implement the Boolean function shown in Table.1.1 (b). The signal originally assigned to L_2 now must enter the cluster through logic cluster input I_2. Similarly, the same signal can be made to enter the cluster through logic cluster inputs I_3 and I_4 respectively, by using the LUT configurations shown in Table.1.1 (c) and Table.1.1 (d).

(a) LUT configuration for connection to cluster input 1
(b) LUT configuration for connection to cluster input 2
(c) LUT configuration for connection to cluster input 3
(d) LUT configuration for connection to cluster input 4

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f2
0	0	1	1	f3
0	1	0	0	f4
0	1	0	1	f5
0	1	1	0	f6
0	1	1	1	f7
1	0	0	0	f8
1	0	0	1	f9
1	0	1	0	f10
1	0	1	1	f11
1	1	0	0	f12
1	1	0	1	f13
1	1	1	0	f14
1	1	1	1	f15

(a)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f2
0	0	1	1	f3
0	1	0	0	f8
0	1	0	1	f9
0	1	1	0	f10
0	1	1	1	f11
1	0	0	0	f4
1	0	0	1	f5
1	0	1	0	f6
1	0	1	1	f7
1	1	0	0	f12
1	1	0	1	f13
1	1	1	0	f14
1	1	1	1	f15

(b)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f8
0	0	1	1	f9
0	1	0	0	f4
0	1	0	1	f5
0	1	1	0	f12
0	1	1	1	f13
1	0	0	0	f2
1	0	0	1	f3
1	0	1	0	f10
1	0	1	1	f11
1	1	0	0	f6
1	1	0	1	f7
1	1	1	0	f14
1	1	1	1	f15

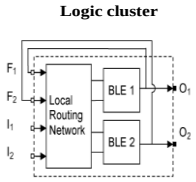
(c)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f8
0	0	1	0	f2
0	0	1	1	f10
0	1	0	0	f4
0	1	0	1	f12
0	1	1	0	f6
0	1	1	1	f14
1	0	0	0	f1
1	0	0	1	f9
1	0	1	0	f3
1	0	1	1	f11
1	1	0	0	f5
1	1	0	1	f13
1	1	1	0	f7
1	1	1	1	f15

(d)

Table.1. LUT configuration for LUT structure in fig.2

The k-input LUT can implement any Boolean function with less than k inputs. Implementing such a function also requires all unused LUT inputs to be connected. Three types of signals can be connected to these inputs. They are the inputs from the Boolean function that is currently being implemented, constant 1's or 0's, and an entirely new set of signals, respectively. If the logic cluster has a set of logically equivalent inputs, each input of can enter the logic cluster through any of the logic cluster inputs. If the logic cluster has a set of logically equivalent outputs, one can implement *f* at any logic cluster output. A feedback signal must also be able to reach *f* from any of the logic cluster outputs.



Logic cluster **S_f for functions implemented at O_1 and O_2**

f(F1,F1), f(F1,F2), f(F1,I1), f(F1,I2)
f(F2,F1), f(F2,F2), f(F2,I1), f(F2,I2)
f(I1,F1), f(I1,F2), f(I1,I1), f(I1,I2)
f(I2,F1), f(I2,F2), f(I2,I1), f(I2,I2)

Fig.3. Logic cluster and set of functions it can be implemented.

The local routing network must be able to generate all functions in S_f for f at each logic cluster output. Conversely, if the local routing network is not flexible enough to generate all functions in S_f at a particular logic cluster output; one must avoid signal assignments that can lead to the un-implementable functions. If these un-impelmentable functions involve logic cluster inputs, then these inputs are no longer logically equivalent to the remaining inputs. Similarly, if the un-implementable functions involve logic cluster feedbacks, then the corresponding logic cluster outputs are no longer logically equivalent to the remaining outputs.

(a) Three input boolean function
(b) Duplicated input implementation
(c) Constant input '0' implementation
(d) New input implementation

L1	L2	L3	L4	O
_	0	0	0	f0
_	0	0	1	f1
_	0	1	0	f2
_	0	1	1	f3
_	1	0	0	f4
_	1	0	1	f5
_	1	1	0	f6
_	1	1	1	f7
_	_	_	_	__
_	_	_	_	__
_	_	_	_	__
_	_	_	_	__
_	_	_	_	__
_	_	_	_	__
_	_	_	_	__

(a)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f2
0	0	1	1	f3
0	1	0	0	X
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	X
1	0	0	1	X
1	0	1	0	X
1	0	1	1	X
1	1	0	0	f4
1	1	0	1	f5
1	1	1	0	f6
1	1	1	1	f7

(b)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f2
0	0	1	1	f3
0	1	0	0	f4
0	1	0	1	f5
0	1	1	0	f6
0	1	1	1	f7
1	0	0	0	X
1	0	0	1	X
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(c)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f2
0	0	1	1	f3
0	1	0	0	f4
0	1	0	1	f5
0	1	1	0	f6
0	1	1	1	f7
1	0	0	0	f0
1	0	0	1	f1
1	0	1	0	f2
1	0	1	1	f3
1	1	0	0	f4
1	1	0	1	f5
1	1	1	0	f6
1	1	1	1	f7

(d)

Table.2. Table describing three properties of LUT

IV FLEXIBILITY OF ROUTING AND RECONFIGURATION

The fully connected local routing network can be configured to any network connection necessary. Hence, in this work certain connections are maintained to achieve the area efficiency. For, attaining certain necessary connection the basic network connection taken is given below.

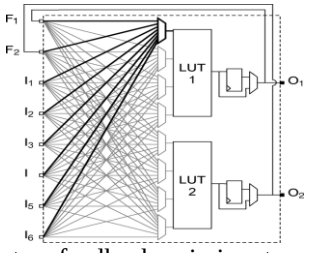


Fig.4 Logic cluster with 2 four-input LUTs, two feedbacks, six inputs, and a fully connected local routing network.

The various connection of the LUT is based on reconfiguration and LUT input rearrangement. Several connection considered in this work are <F_2,I_5,I_1,F_2> configuration, <0,I_5,I_1,F_2> configuration, <F_2,I_1,I_5,0> configuration <F_2,I_1,I_5,I_6> configuration.

(a) <F_2, I_5,I_1, F_2 > configuration
(b) <0,I_5,I_1,F_2> configuration
(c) <F_2,I_1,I_5,0> configuration
(d) <F_2,I_1,I_5,I_6> configuration

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f1
0	0	1	0	f2
0	0	1	1	f3
0	1	0	0	f4
0	1	0	1	f5
0	1	1	0	f6
0	1	1	1	f7
1	0	0	0	f8
1	0	0	1	f9
1	0	1	0	f10
1	0	1	1	f11
1	1	0	0	f12
1	1	0	1	f13
1	1	1	0	f14
1	1	1	1	f15

(a)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f9
0	0	1	0	f2
0	0	1	1	f11
0	1	0	0	f4
0	1	0	1	f13
0	1	1	0	f6
0	1	1	1	f15
1	0	0	0	x
1	0	0	1	x
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

(b)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	x
0	0	1	0	f4
0	0	1	1	x
0	1	0	1	f2
0	1	0	1	x
0	1	1	0	f6
0	1	1	1	x
1	0	0	0	f9
1	0	0	1	x
1	0	1	0	f13
1	0	1	1	x
1	1	0	0	f11
1	1	0	1	x
1	1	1	0	f15
1	1	1	1	x

(c)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f0
0	0	1	0	f4
0	0	1	1	f4
0	1	0	0	f2
0	1	0	1	f2
0	1	1	0	f6
0	1	1	1	f6
1	0	0	0	f9
1	0	0	1	f9
1	0	1	0	f13
1	0	1	1	f13
1	1	0	0	f11
1	1	0	1	f11
1	1	1	0	f15
1	1	1	1	f15

(d)

Table.3. Table describing several configuration of fig.4

In particular, if an LUT input is only connected to a subset of logic cluster inputs and feedbacks, a signal assigned to the LUT input can only enter the cluster through the connected inputs/feedbacks these connected inputs/feedbacks are no longer logically equivalent to the unconnected ones. The fully connected local routing network can be designed as N × K (I+N):1 multiplexers where, N is the number of feedback given to the cluster, K is the number of input given to the LUT, I is the logic cluster input.

V SPARSE NETWORK AFTER RECONFIGURATION

Let A be a k-input LUT implementing a Boolean function f (a_1,a_2,a_3,……,a_k) . Let i_1,i_2,i_3….i_n be the output signals from n LUTs. Let v be a k-bit wide bit vector containing a subset of k signals from{i_1,i_2,i_3….i_n} . If v is in s_i and i_x is the j th element of n-k+j of , then v must be smaller than or equal to j. A local routing network can be used to connect the j th input of a k - input LUT to all signals in the set{i_j, i_{j+1}, …..,i_{n-k+j}} through an (n-k+1):1 multiplexer. Through LUT reconfiguration and function transformations, the LUT can be used to generate all functions in S_f. To generate all functions in S_f without reconfiguration, each input of the k -input LUT must be connected to all signals in {i_1,i_2,i_3….i_n} through an n:1 multiplexer. For example, for the logic cluster shown in fig.2.b there are four logic cluster inputsI_1,I_2, I_3, I_4 and , and no feedbacks. The LUT input L_1 should be connected to all signals in the set {I_1} (for j=1, n=4, and k=4), L2 should be connected to all signals in the set {I_2} (for j=2, n=4 and k=4), L3 should be connected to all signals in the set {I_3} (for j=3, n=4 and k=4, L4 should be connected to all signals in the set {I_4} (for j=4, n=4 and k=4). With reconfiguration, the local routing network is able to generate all functions in set S_f.

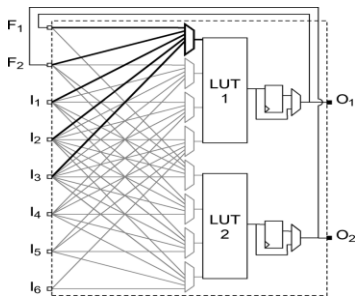


FIG.5 SPARSE LOCAL ROUTING NETWORK.

After reconfiguration the multiplexer size is reduced from 8:1 to 5:1. For N data inputs, the number of control bits should be ceil(log N) For example N = 5, ceil(log (5)) = 3. Thus, there are 3 control bits. Other three inputs should be treated as don't cares.

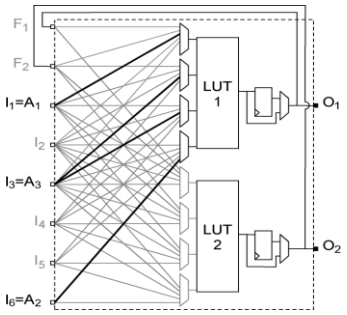


Fig.6 Implementation 1

The fanout of F_1 and I_6 is reduced from 8 to 2. The fanout of F_2 and I_5 is reduced from 8 to 4; the fanout of $I_1$and I_4 is reduced from 8 to 6; and the fanout of I_2 and I_3 remains unchanged at 8. The fanouts of all logic cluster inputs and feedbacks can be reduced to 5 by rearranging the order $\{I_2,I_1,F_2,F_1,I_6,I_5,I_4,I_3\}$ of the logic cluster inputs/feedbacks to when the inputs and feedbacks are connected to LUT 2.

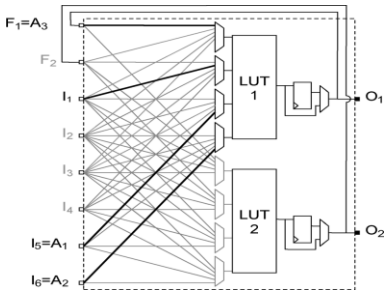


Fig.7 Implementation 2

Both logic cluster designs retain logic equivalency among logic cluster inputs and outputs. For example, consider implementing the three-input Boolean function. In the logic cluster shown in Fig.5, there are 336 unique ways that the three inputs can enter the logic luster. Fig.6 and 7 show two of the possibilities.

(a) A three-input boolean function
(b) LUT configuration for imp. 1.l1=a1, l2=a3, l3=a3,l4=a2
(c) LUT configuration for imp. 2. l1=a3, l2=i1, l3=a1, l4=a2

A1	A2	A3	O
0	0	0	f0
0	0	1	f1
0	1	0	f2
0	1	1	f3
1	0	0	f4
1	0	1	f5
1	1	0	f6
1	1	1	f7
-	-	-	__
-	-	-	__
-	-	-	__
-	-	-	__
-	-	-	__
-	-	-	__
-	-	-	__

(a)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f2
0	0	1	0	x
0	0	1	1	x
0	1	0	0	x
0	1	0	1	x
0	1	1	0	f1
0	1	1	1	f3
1	0	0	0	f4
1	0	0	1	f6
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	f5
1	1	1	1	f7

(b)

L1	L2	L3	L4	O
0	0	0	0	f0
0	0	0	1	f2
0	0	1	0	f4
0	0	1	1	f6
0	1	0	0	f0
0	1	0	1	f2
0	1	1	0	f4
0	1	1	1	f6
1	0	0	0	f1
1	0	0	1	f3
1	0	1	0	f5
1	0	1	1	f7
1	1	0	0	f1
1	1	0	1	f3
1	1	1	0	f5
1	1	1	1	f7

(c)

Table.4 LUT configurations for implementations 1 and 2.

A_1, A_2, and A_3 are assigned to cluster inputs I_1, I_6 and I_3 respectively. A_3 is also duplicated to provide the fourth LUT input. The corresponding LUT configuration is shown in Table.4 (b) Alternatively, in fig.7 a router can assign A_1, A_2, and A_3 to I_5, I_6 and F_1 respectively. Due to the sparse local routing network, none of the three inputs can be expanded into the fourth LUT input. Instead, an arbitrary cluster input I_1 is used as the fourth input. In a directional single-drive architecture, each track is driven by its own buffer. Consequently, it can be connected to any of the routing tracks since the LUT is configured to provide the same output for both and as shown in Table.4 (c). As similarly the further work goes with increasing the LUT input. For, k=5 all the network connections with fully connected network is implemented. The area minimization can be noted in terms of combinational ALUTs (Adaptive LUTs).

VI. LOGICAL EQUIVALENCY

Logically equivalent inputs and outputs allow a signal to enter or exit a logic cluster in several ways. This added connectivity increase the flexibility of the routers and can lead to better utilization of the routing resources. In this work a basic multiplier circuit is implemented in a fully connected network which is been created. The circuit diagram of multiplier which is implemented in this work is given below

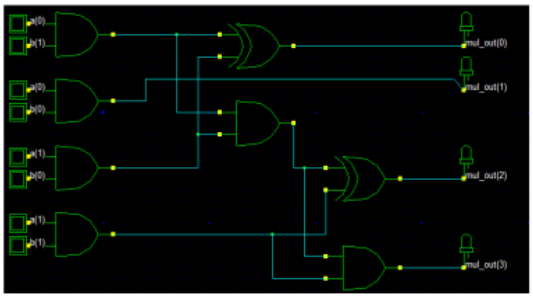


Fig.8 Multiplier circuit with logical equivalency

VII. LOGICAL NON-EQUIVALENCY

Each contains a set of non-equivalent inputs/outputs each input signal must enter the cluster through a dedicated cluster input and each output signal must exit the cluster through a dedicated cluster output [8]. This concept goes with the implementation of multiplier as the enhanced work based on the base paper. In logical equivalency the multiplier circuit is composed of combination of EXOR gate and AND gate. But, in the logical non-equivalence the multiplier circuit is composed of only universal gate. In this paper the universal gate used for multiplier circuit to attain the logical non-equivalency is NAND gate.

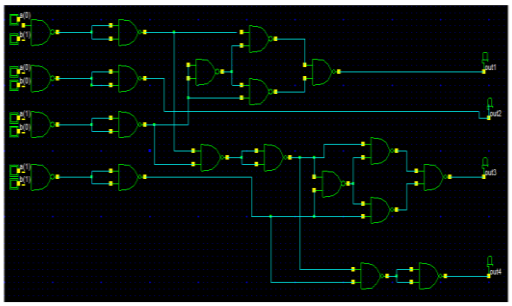


Fig.9 Multiplier circuit with logical non equivalency

VII. RESULTS AND DISCUSSION

This paper work is simulated using the tool Modelsim simulator in VHDL (Very high speed integrated circuit Hardware Descriptive Language) language. The tool used for synthesis process is Quartus. The device used while synthesing is StratixII device. The objective attained in this work is area reduction and fanout reduction. Area reduction is mentioned in terms of combinational ALUTs attained in the synthesis process. Fanout reduction can be mentioned as maximum fanout, total fanout and average fanout. The combinational ALUTs obtained

- Fully connected network is 18,
- Connection <F_2, I_5, I_1, F_2> is 14,
- Connections for <F_2, I_1, I_5, 0> is 11,
- Sparse local routing network is 5,
- Implementation 1 and implementation 2 is 2.

The fanout result for the synthesized networks with the LUT input of 4 can be given as

- Fully connected network average fanout is 2.21,
- Connection <F_2, I_5, I_1, F_2> average fanout is 1.48,
- Connections for <F_2, I_1, I_5, 0> average fanout is 1.18,
- Sparse local routing network average fanout is 0.58,
- Implementation 1 and implementation 2 average fanout is 0.31.

Similarly for 5 input LUT network all results obtained will be similar as 4 input LUT but the net result will increase because of increase in number of inputs. Such as, the whole network can be created with a collection of logic clusters with varying input of the LUT. The RTL (Register Transfer Level) for all the above mentioned network can be attained using synthesis tool Quartus. The constructed network can be implemented using VPR tool (Versatile Place and Route tool) if the implementation is based on the physical end. VPR is an industry based tool. VPR can perform placement and either global routing or combined global and detailed routing.

The implementation of multiplier with the device of stratix II results with the combinational ALUTs of 4 in logically equivalent state. Similarly, the multiplier circuit with logically non-equivalent state composed only of NAND gate also result with the same number of 4 combinational ALUTs.

Alternatively, since some of the logic clusters contain feedbacks, the 8:1 multiplexers can be used to construct logic clusters containing 4 four-input LUTs with eleven logic cluster inputs. Again, this design would require a narrower channel width in order to support the smaller four LUT clusters.

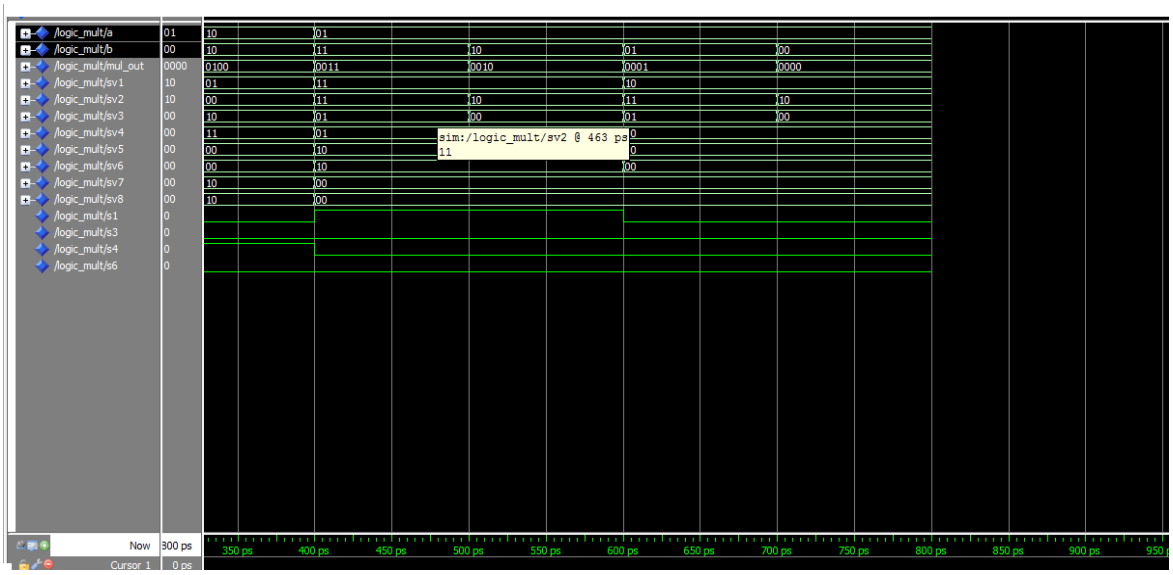


Fig.10. Simulation result of logically equivalent multiplier

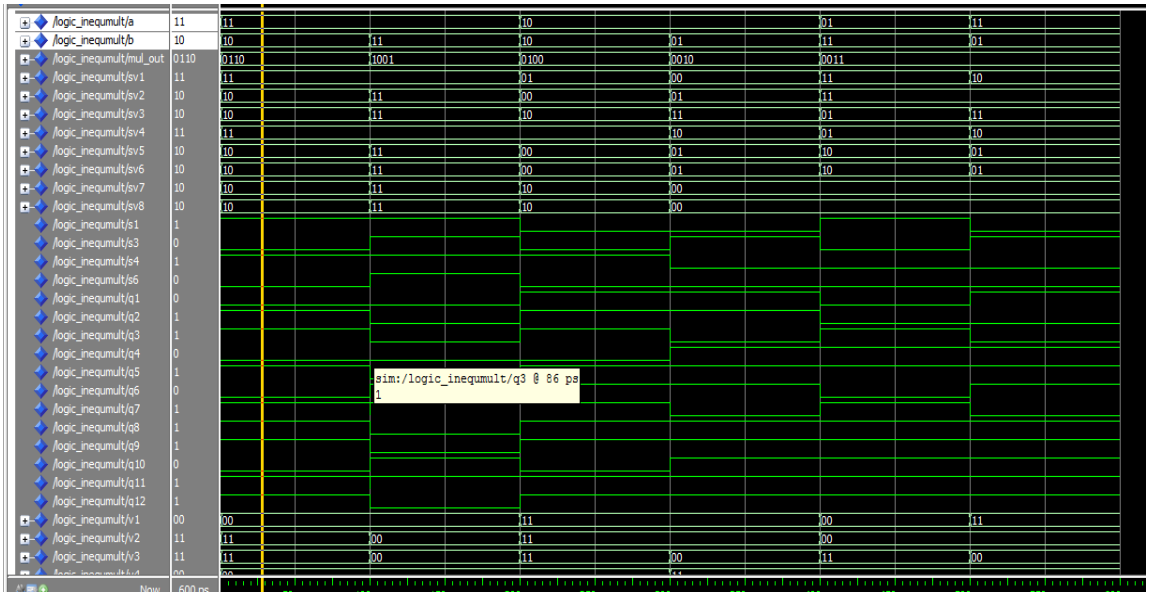


Fig.11 Simulation results of logically non-equivalent multiplier

VIII. CONCLUSION

The examination of the paper reveils the relationship between the logic equivalency of logic cluster input and outputs and LUT reconfiguration for FPGA local routing networks. Also, examined the relationship between the logical equivalency and non-equivalency of the particular network. Since the four LUT design retains logic equivalency among the logic cluster I/Os and has less logic cluster inputs per LUT. This design should also be experimentally evaluated as an extension of future work, along with an examination on the effect of the sparse local routing network design on the power efficiency of FPGAs.

REFERENCES

[1] A. Marquardt, V. Betz, and J. Rose, "Speed and area trade-offs in cluster-based FPGA architectures," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 8, no. 1, pp. 84–93, Feb. 2000.
[2] Andy Gean Ye,"Using the Minimum Set of input combinations to Minimize the Area of Local Routing Networks in Logic Clusters Containing logically Equivalent I/Os", *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 18, no. 1, january 2010.
[3] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deepsubmicron FPGA performance and density," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 3, pp. 288–298, Mar. 2004.
[4] V. Betz and J. Rose, "How much logic should go in an FPGA logic block?," *IEEE Des. Test Comput. Mag.*, vol. 15, no. 1, pp. 10–15, Jan.-Mar. 1998.
[5] V. Betz and J. Rose, "Effect of the prefabricated routing track distribution on FPGA area-efficiency," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 6, no. 3, pp. 445–456, Sep. 1998.
[6] C. E. Shannon, "The synthesis of two-terminal switching circuits," *Bell Syst. Tech. J.*, vol. 28, no. 1, pp. 59–98, Jan. 1949.
[7] G. Lemieux, "Directional and single-driver wires in FPGA interconnect," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, 2004, pp. 41–48.
[8] A. Roopchansingh and J. Rose, "Nearest neighbour interconnect architecture in deep submicron FPGAs," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2002, pp. 59–62.
[9] A. Ye and J. Rose, "Using bus-based connections to improve fieldprogrammable gate array density for implementing datapath circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 462–473, May 2006.

Sathyabhama.B pursuing master's of engineering in VLSI design stream Easwari engineering college Chennai under Anna university of Chennai . This work was guided by Dr.S.Sudha, professor, Assistant HOD ECE department in Easwari engineering college Chennai. Current research interests include field-programmable gate array (FPGA) architectures, computer-aided design (CAD) tools for FPGAs, logic synthesis.