Scientia Iranica, Vol. 11, No. 3, pp 159–164

SCIENTIA IRANICA

On Routing Architecture for Hybrid FPGA

M. Nadjarbashi*, S.M. Fakhraie[1] and A. Kaviani[2]

In this paper, the routing architecture for an FPGA with hybrid clusters built from a mixture of LUT-based and PLA-like blocks is investigated. The implemented CAD flow that is used to place and route a number of MCNC benchmark circuits in a comparative fashion is discussed. The experimental results demonstrate that cluster sizes of two (2 LUT blocks and 2 PAL blocks) to four (4 LUT blocks and 4 PAL blocks) are appropriate in terms of area and speed. A comparison between hybrid and LUT-based FPGA architectures is also presented, showing that hybrid FPGA has some considerable advantages over a uniform LUT-based architecture.

INTRODUCTION

Hybrid Field Programmable Architecture (HFPA) was introduced in 1996 [1], combining the two common technologies used in programmable logic devices: FPGAs based on Look-Up Tables (LUTs) and CPLDs incorporating PLA-like logic cells. The main idea is to determine what functions are suitable to be implemented on which logic resources. Logic resources include LUTs as well as product-term (PLA-like) logic cells. For small combinational nodes with less than four or five inputs, LUTs are more area efficient than PLAs. On the other hand, high-fanin nodes with more than five inputs could be efficiently implemented in PLA-like blocks, unless they have a large number of Pterms. It has been shown in [2] that a major number of nodes in benchmark circuits are 4-bounded (where the number of inputs is less than, or equal to four) and most of the high-fanin nodes are suited for implementation in PLAs. According to the results presented in [2], hybrid architecture offers both area and depth advantages compared to LUT-based FPGAs. In this paper, the routing parameters of hybrid clusters are investigated with the goal of optimizing the area and performance of the designs in the FPGA.

The next section describes a tentative structure of a hybrid cluster, as proposed in previous research [2]. Then, the CAD flow is explained followed by assumptions about architectural parameters and the employed VLSI technology. After that, the achieved results on hybrid FPGA are presented and discussed. Hybrid FPGA and LUT-based architecture are compared in the final section.

HYBRID CLUSTER

Figure 1 shows a cluster of HFPA, named hybrid cluster. PALB stands for Programmable Array Logic Block, which is a product-term (PLA-like) logic cell. Its structure and architectural specifications have been chosen as investigated and discussed in [2]. It has 16 inputs, an AND-plane, three outputs (where two out of three could be registered) and its structure is developed to accommodate up to three combinational nodes on average. A Look-Up Table Block (LUTB) contains four 4-LUTs (Figure 2), each with an arbitrarily latched output. Block-level tracks in a LUTB are connected to the logic through a fully-populated crossbar, in which

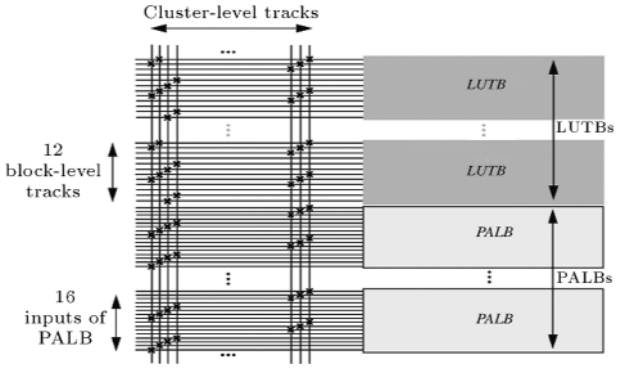

Figure 1. Hybrid cluster.

*. *Corresponding Author, Department of Electrical and Computer Engineering, University of Tehran, P.O. Box 14399, Tehran, I.R. Iran.*

1. *Department of Electrical and Computer Engineering, University of Tehran, P.O. Box 14399, Tehran, I.R. Iran.*
2. *Xilinx Inc. 2100 Logic Drive San Jose, CA 95124, USA.*

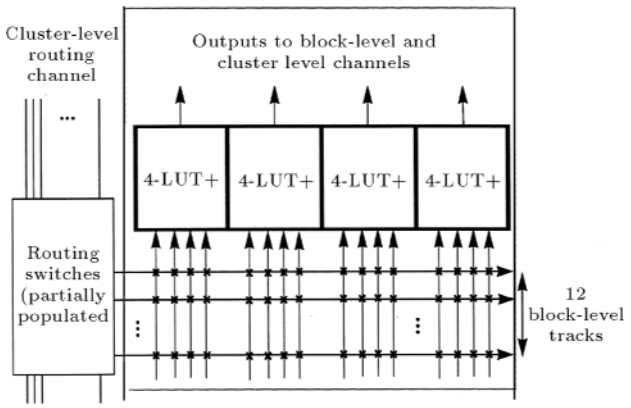

Figure 2. LUTB structure.

each block-level track can be arbitrarily connected to each of 16 4-LUT inputs. Providing full connectivity at lower levels of the hierarchy makes it possible to reduce the number of switches at the higher levels, where transistors drive larger loads and require more chip area. As shown in Figure 1, the cluster-level to block-level crossbar is partially populated.

The ratio between the number of LUTBs and the number of PALBs, which is called Balance Factor (BF), is selected to be one. This selection is based on previous work in [2] showing that benchmark circuits, on average, can get close to their maximum gain of speed and area when implemented on an architecture with $BF = 1$.

While the cluster is similar to that of [2], it is assumed that hybrid clusters are interconnected in an island-style position, like the Xilinx FPGAs [3]. This is different from previous work in [2] that uses a hierarchical routing structure to connect the clusters. This decision was made because the public version of VPR only supports island-style FPGAs.

CAD FLOW

Since there is no manageable analytical model of an FPGA chip, most of the research in this field is accomplished using investigations performed on benchmark circuits. An architectural parameter resides in its optimum point if any other amount for that parameter causes the proposed research criteria (average area and delay of all benchmark circuits) to become worse, while all other independent parameters are kept fixed. It is not possible to simultaneously explore the entire N-dimensional space of all possible routing architectures. Therefore, one or two architectural parameters are varied each time and optimized along a line in the N-dimensional architecture space. One should begin with parameters that do not have a strong interaction with other parameters. These parameters can be fixed at their best values and optimization will carry on along other dimensions.

When the criterion is the delay and area of the implemented benchmarks, the "Implementation Routine" is similar to that which is performed in commercial FPGAs to automatically implement circuits: i.e., technology-independent logic optimization, technology mapping, packing into clusters, placement and routing. However, some of the results were obtained directly from the packing tool (Hpack) [4] and the placement/routing phases were not required.

The technology-independent logic optimization plus technology-mapping phases are conducted as described in [2], leading to circuits mapped on 4-LUT and PALB cells. The mapped circuits could be optimized for area or depth (delay). After this stage, the packing part of the Hpack tool receives the mapped circuits and tries to build LUTBs. Hpack constructs clusters from a network of PALBs and LUTBs by taking advantage of shared signals among various blocks. Furthermore, Hpack is able to do cluster-level to block-level interconnection (internal routing), determining how many switches are required to perform successful routing.

For placement and routing, VPR has been chosen [5], which is a tool for FPGA placement and routing. Unfortunately, VPR accepts architectures with only two routing levels, but the hybrid architecture has three routing levels: Block-level, cluster-level, and chip-level. For this reason, Hpack and front-end represent the circuit and FPGA architecture specification in the format that VPR desires. Since the Hpack and front-end handle all the jobs related to cluster-inside operations, no problem would occur by this representation.

All of the mentioned stages are glued together by the front-end code. The front-end entity carries out another important task: Generating the FPGA architecture file for VPR. VPR requires architectural specifications of FPGA, in addition to the circuit description. VPR does the placement of hybrid clusters, then tries to route them such that a minimum number of chip-level routing tracks are required. Other different scripts are used to extract and prepare the results from raw data at different stages and parts.

ARCHITECTURAL ASSUMPTIONS AND TECHNOLOGY

A wide variety of technological, as well as architectural parameters should be specified in the FPGA architecture file of VPR. The technology issues include parameters such as RC values (time constants) of different wire segments used in routing channels, RC values of different switch types (tri-state buffer, pass-transistor), resistance of base NMOS and PMOS transistors (resistance of channel), delay through different parts of a cluster and so on. For this purpose, the TSMC 0.35μ process has been chosen and all the

required values and parameters [4] have been calculated based on rules and specifications of this process as discussed in [6]. The layouts of PALB and LUTB were drawn and the timing information (such as delay through different combinational and sequential paths) is extracted.

Architectural parameters involve both detailed and global routing parameters, such as the switch box type, the number of tracks to which each cluster input/output pin connects, details of different wire segment types (length, internal population and fraction of routing tracks composed of this type), the fraction of switch types used, routing channel parameters, number of pads and so on. Research on mentioned issues and their effects on FPGA area and delay has been accomplished in [6], and the relevant parameters have been set accordingly with some changes. For example, in [6], a segment length of 4 is considered to be appropriate for a LUT-based cluster with four 4-LUTs (if all segments have the same length). The authors' experiments, however, showed that for larger clusters (similar to the ones investigated), segments of size 2 lead to a lower area and delay. The reason is that in FPGAs with large clusters, large segments result in long metal tracks that require larger buffers to drive and these buffers, in turn, cause delays in signal paths (see also section of Optimum Number of Cluster Blocks and Tracks).

Whether the goal would be area or depth reduction, many of the above parameters may differ. For example, if one were going to reduce the depth, it would be important to use tri-state buffers as much as pass-transistor switches (a roughly equal ratio would be suitable), while reducing the usage of tri-state buffers (as low as 20%-30%) would manage to get more advantage in area gain.

In this research, the drive strength of routing switches has been increased by resizing the transistors as the size of a cluster increases. This allows of reduction in the routing delay. Also, for performance reasons, the cluster-level to block-level crossbar has been implemented using pass-transistors instead of fully encoded multiplexers that occupy less area but have a higher propagation delay.

OPTIMIZATION PROCEDURE AND RESULTS

LUTB inputs

The first cluster parameter that was investigated using Hpack was the number of LUTB inputs. Due to input sharing, the four 4-LUTs often require less than 16 input lines. Figure 3 shows the LUTB logic utilization vs the number of LUTB inputs. This diagram indicates that if one chooses this number to be 11 or more, there

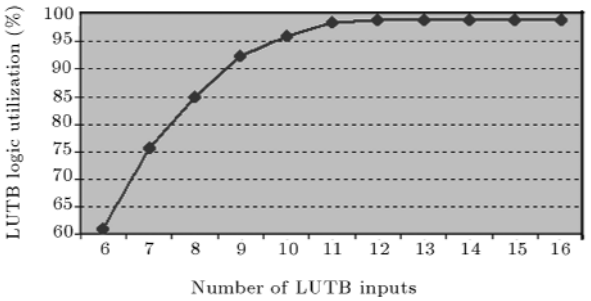

Figure 3. LUTB logic utilization vs number of inputs (obtained over 31 MCNC BMs).

is roughly no logic waste in LUTB. If the number of LUTB inputs is selected less than nine, a noticeable number of circuit nodes should be accommodated in other LUTBs, causing an increment in the area and delay of circuit while wasting logic in most of the LUTBs. The number of LUTB inputs for the next experiments has been chosen as 12.

Internal Routing Switches

Hpack can determine the best switch distribution for cluster-to-block crossbar for each N, where N is the number of PALBs (or LUTBs) in each cluster. After packing LUTBs and PALBs of a circuit into different clusters, the input signals (nets) of each cluster are known. Now, Hpack tries to connect these input lines to appropriate block-level lines by a uniform deployment of switches (see Figure 1). If F_{cb} is defined as the number of switches by which each cluster-level track can be connected to a block (PALB/LUTB), then Hpack is capable of considering different values of F_{cb} for different blocks within each cluster. Hpack starts by using $F_{cb} = 1$ for all of the blocks in a cluster and increases F_{cb} by one for each block until routing is successful. Finally, Hpack stops when every block is connected with the minimum F_{cb}. Thus, Hpack can obtain the suitable F_{cb} value corresponding to each block for a typical cluster, on average. This will be the appropriate switch distribution for all clusters of FPGA. Since manufacturing limitations prohibit the use of various F_{cb} numbers in a cluster, the rounded value of 2 is suggested for a uniform F_{cb} (such as Figure 1: $F_{cb} = 2$), based on the last column of Table 1.

Utilization of Local Interconnections

Another experiment undertaken is calculating the average percentage of local interconnections when N varies. Local interconnections (connections that start and end within the same cluster) are faster than connections that pass through cluster-outside routing switches and tracks. Table 2 shows how this percentage varies as hybrid cluster size increases.

Table 1. Non-uniform cluster-level switch distribution that implies manufacturable $F_{cb} = 2$ (obtained over 31 MCNC BMs).

Cluster Size	Number of Blocks with $F_{cb} = 1$	Number of Blocks with $F_{cb} = 2$	Number of Blocks with $F_{cb} = 3$	Number of Blocks with $F_{cb} = 4$	Effective F_{cb}
$N = 1$	1	1	0	0	1.5
$N = 2$	2	1	1	0	1.75
$N = 4$	4	2	1	1	1.875
$N = 8$	6	3	5	2	2.1875
$N = 16$	12	7	8	5	2.1875

Table 2. Local interconnection percentage vs cluster size (obtained over 31 MCNC BMs).

Cluster Size	$N = 1$	$N = 2$	$N = 4$	$N = 8$	$N = 16$	$N = 32$
Local Interconnection	4.5%	7.0%	11.4%	16.5%	26.4%	41.4%

Optimum Number of Cluster Blocks and Tracks

Now, placement and routing steps are added to measure the area and delay of implemented benchmarks when a cluster parameter changes. The number of blocks within each cluster is fixed in order to see how the average area of implemented circuits varies when the number of cluster-level tracks changes. If this number is too small, then the number of clusters required to implement the circuits will increase and then the total area will increase. If this number is too large, many of these tracks will remain unused and the total area will increase due to wasted routing resources. Therefore, a minimum point is expected, as depicted in Figure 4. This diagram corresponds to $N = 2$ and the area is minimum when the number of cluster inputs (I) is about 46.

If the value of N is too small, then the number of clusters required to implement a typical circuit will increase because of small clusters. But, due to the repetitive supporting circuits existing in each cluster (such as clock distribution and preparation circuits and latch set/reset logic), the area that each cluster and its dependent routing tracks occupy do not decrease in the same ratio with which N decreases. Thus such a small cluster would not be appropriate. In large clusters, two factors make them inappropriate. First, the significant overhead of cluster-level to block-level switch net and second, the huge transistors required to drive long tracks and other large switches existing in a chip-level routing area. Table 3 shows the optimum amount of parameter "I" corresponding to different values of N (benchmarks are optimized for the area). Despite the above, the best cluster in terms of area is achieved when $N = 1$ and $I = 28$. It was found that this is due to the significant overhead of cluster-level to block-level crossbar that, here, overcomes all other factors. The experiments indicate that if one could somehow omit this overhead, then the optimum cluster in terms of area would be achieved with $N = 2$ or $N = 4$. Despite this overhead, when circuits are optimized for delay, the case $N = 2$ has the lowest area.

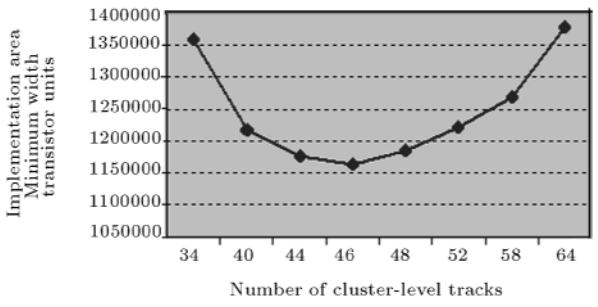

Figure 4. Implementation area vs number of cluster-level tracks, $N = 2$ (obtained over 17 MCNC BMs).

Figure 5 shows the effect of increasing "I" on average critical path delay for case $N = 2$. Since the router algorithm always tries to establish the circuit path through the lowest delay path available, then significant changes in critical path delay is not happening as it was for area. However, for clusters with $N = 1$ to $N = 4$, the delay will decrease if "I" increases. This is because the total number of necessary clusters decreases and a typical connection encompasses less series of logic clusters. Furthermore, the router algorithm performs its job under more low-stressed conditions [6]. As discussed before, the delay criterion should decrease when N increases. This is

Table 3. Implementation area vs cluster size (obtained over 17 MCNC BMs).

Cluster Size	$N = 1$	$N = 2$	$N = 4$	$N = 8$
Minimum Area	1.15 M	1.16 M	1.37 M	2.58 M
Corresponding "*I*"	28	46	70	120

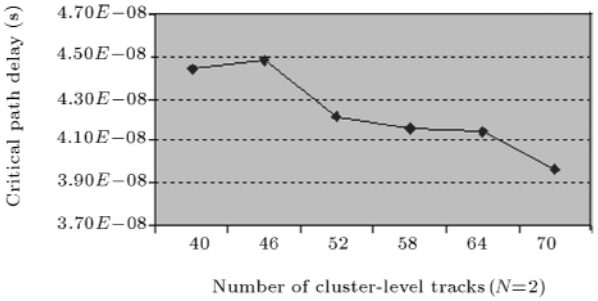

Figure 5. Performance changes vs number of cluster-level tracks (obtained over 17 MCNC BMs).

true for clusters with $N = 1$ to $N = 4$, but it was observed that for $N = 8$ (and above) the delay criterion started to increase. This is due to the enlargement of switch transistors as the size of cluster increases. Too much increase in the size of these transistors causes the effect of self-loading to overcome the drive-strength increment one expects to achieve. A solution to this problem is that, instead of too much increment in cluster size, the number of routing levels are increased such that most of the connections are established locally and through shorter wire segments.

COMPARING HFPA WITH LUT-BASED ARCHITECTURE.

A previous work [2] has shown that HFPA has some advantages over traditional LUT-based architectures in terms of logic resources: Up to 25% in area and up to 65% in depth. That work did not consider routing delays, and the routing architecture of this paper is somewhat different from that of [2], as explained in section of Hybrid Cluster. In this section the area and delay of the benchmark circuits in a traditional LUT-based FPGA are compared with the area and delay of the hybrid FPGA.

For comparison, a reference FPGA with only LUTBs has been considered and the formerly mentioned steps have been accomplished. Two benchmark groups have been produced that are optimized and technology mapped towards area and depth reduction.

The achieved results for LUT-based FPGA are partly similar to those achieved in previous sections (Table 4 and Figures 6 and 7). The best cluster, in terms of area, is case $N = 2$ with 34 inputs. If this area (1.58 M) were compared with the minimum area number obtained for hybrid FPGA (1.15 M), it would be discovered that LUT-based FPGA occupies

Table 4. Comparison of hybrid and LUT-based FPGAs (obtained over 17 MCNC BMs).

	Hybrid FPGA		LUT-Based FPGA	
	Delay(ns)	Area	Delay(ns)	Area
$N = 1$	42.2	1.15 M	45.1	1.61 M
$N = 2$	41.5	1.16 M	44.7	1.58 M
$N = 4$	39.0	1.37 M	44.2	2.32 M
$N = 8$	43.8	2.58 M	47.4	3.93 M

about 37.4% more area for the same logic capacity, while the corresponding delay is about 6% more than hybrid FPGA. If delay-optimized circuits are used and the optimum points regarding the area are calculated, the best cluster for LUT-based architecture will be the case $N = 4$, in which delay criterion is 44.2 ns. Compared to hybrid FPGA performance (39.0 ns in $N = 4$ as the best case), the delay of LUT-based architecture is 13.3% more than hybrid architecture while the respective area is 82.1% more.

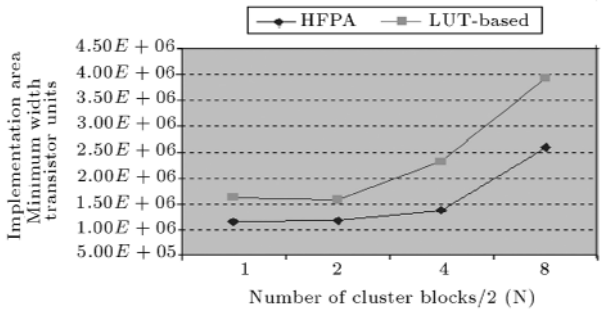

Figure 6. Implementation area vs number of cluster blocks for hybrid FPGA and LUT-based FPGA (obtained over 17 MCNC BMs).

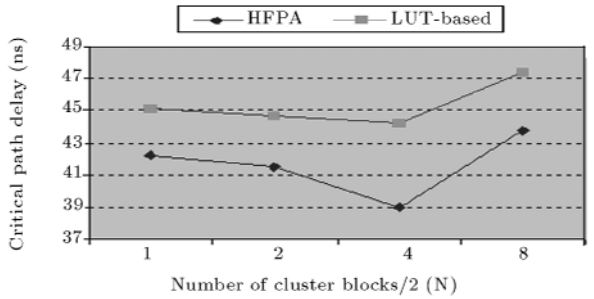

Figure 7. Critical path delay vs number of cluster blocks for hybrid FPGA and LUT-based FPGA (obtained over 17 MCNC BMs).

Table 5. Comparison of HFPA and LUT-based architecture for some selected benchmarks optimized for performance ($N = 4$).

	Hybrid FPGA		LUT-Based FPGA		Delay	Area
Benchmark	Delay (ns)	Area (M)	Delay (ns)	Area (M)	Gain	Gain
alu4	31.7	1.89	46.4	3.56	47.0%	88.3%
ex1010	38.0	2.94	54.2	6.09	42.6%	105.7%
ex4p	24.0	0.37	27.0	0.65	12.5%	75.7%
misex3	37.6	2.01	43.1	3.27	14.6%	62.7%
S1488	20.4	0.41	29.9	0.87	46.6%	112.2%
seq	44.6	2.82	53.1	4.89	19.0%	73.4%
spla	60.1	7.94	67.0	15.12	11.5%	90.4%

Despite area, hybrid FPGA does not have a significant advantage over LUT-based architecture in terms of performance. This is due to the fact that logic resources have a minor role in the delay of the different paths of an implemented circuit. In the architecture presented, the share of the 4-LUTs delay is about 8% of the average delay and the remaining is due to the routing resources (a small share is due to I/O pads). However, when one investigates the benchmark circuits individually, it can be seen that for circuits like "s1488", "seq", "ex4p", "ex1010", "misex3", "spla" and "alu4", the HFPA has a considerable advantage over LUT-based architecture (Table 5). By surveying the type of these circuits, one discovers that most of these benchmarks have a controller (state machine) appearance or are 2-level logic circuits with a high number of inputs. Both of these circuit types are suitable for implementation on product-term logic resources.

CONCLUSIONS

In this paper, some optimum values have been identified for routing and the structural parameters of hybrid field programmable devices. The results are obtained using MCNC benchmark circuits. According to the results, a cluster of size $N = 2$ to $N = 4$ would be appropriate for both area and performance. For $N = 2$, the average delay is 6% higher compared to $N = 4$, which is the best case. The average area for $N = 2$ is 1% higher than that of $N = 1$, which is the best case for area. For all values of N, the values for other cluster-related parameters, such as internal switch population and number of cluster-level tracks have been optimized.

The hybrid FPGA has also been compared with a traditional LUT-based FPGA composed of only LUTBs, which have been optimized with the same methodology. It is shown discovered that average area gain of hybrid FPGA over LUT-based architecture is about 82%, while the performance gain is around 13%. Higher gains for performance, however, are achievable for circuits that have a controller (state machine) nature or are mainly composed of two-level logic with a high number of inputs.

REFERENCES

1. Kaviani, A. and Brown, S. "Hybrid FPGA architecture", *International Symposium on Field-Programmable Gate Arrays (FPGA'96)*, Monterey, CA, USA (1996).
2. Kaviani, A. "Novel architectures and synthesis methods for high capacity field programmable devices", Ph.D. Dissertation, University of Toronto (1999).
3. *The Programmable Logic Data Book*, Xilinx Inc., San Jose, CA, http://www.xilinx.com/partinfo/databook.htm (1999).
4. Nadjarbashi, M. "Optimization on architecture and routing of hybrid field programmable gate arrays", M.Sc. Thesis, University of Tehran (2000).
5. Betz, V., *Vpack and VPR User's Manual v4.22* (1999).
6. Betz, V., *Architecture and CAD for Speed and Area Optimization of FPGA*, Ph.D. Dissertation, University of Toronto (1998).