Area-Speed Tradeoffs for Hierarchical Field-Programmable Gate Arrays

Vi Cuong Chan and David M. Lewis
University of Toronto
Department of Electrical and Computer Engineering

Abstract

This paper investigates area-speed trade-offs for Hierarchical FPGA (HFPGA) architectures. Using a set of new CAD tools, we measured the timing performance of HFPGAs and conventional symmetrical FPGAs using data gathered from experiments on a subset of benchmark circuits from the Microelectronics Centre of North Carolina (MCNC). Experiments were also performed to determine the effect of timing optimized placements on routing channel requirements. These experiments demonstrate that HFPGAs can achieve both better area and speed than symmetrical FPGA architectures [2].

1 Introduction

The ability of Field Programmable Gate Arrays (FPGAs) to rapidly prototype and implement complex multilevel digital circuits has made them the subject of many research studies [3 - 9]. This paper explores area-speed trade-offs in Hierarchical FPGAs (HFPGAs.) First proposed by Aggarwal and Lewis [1], an HFPGA consists of a hierarchy of logic blocks connected by partially populated switch blocks and routing channels. Recently, the FLEX series of FPGAs by Altera[21] has been introduced, which shares some features with the HFPGA architecture described here. While the reduced area of HFPGAs compared to other FPGAs has been demonstrated [1], timing issues have so far been ignored. This paper explores timing issues in HFPGAs. In particular, the use of a timing-based placement is investigated, and the increased routing requirements due to a timing-based placement are explored. Further, the area-speed trade-off of different HFPGA architectures is explored. Finally, a comparison to other FPGAs, such as the Xilinx XC3000, is made. The results show an average 30-40% reduction in the routing delay for the critical paths with only small increase in area.

We refer to FPGAs with a symmetrical grid of logic blocks and routing channels on all four sides of the logic block as symmetrical FPGAs. This architecture offers great flexibility and routability. On the other hand, symmetrical FPGAs may be slow due to the large number of programmable switches along a routing path. Structures such as segmented channels, direct interconnect and long lines (as in Xilinx XC3000 [10]) have been used to reduce routing delay.

This work was supported by Micronet

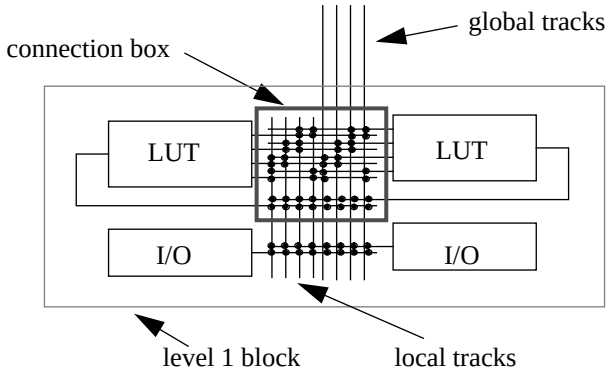

Figure 2.1 Example of level 1 logic block with a (2 + 2) architecture

Since most digital designs contain clusters of circuits with many local connections, an FPGA that is divided into hierarchies of blocks, each comprising a collection of interconnected logic blocks, may be able to provide lower routing delays and more predictable timing behavior. HFPGAs provide segmented routing structures using a hierarchy of logic blocks with short local wires as well as long global wires.

Section 2 of this paper describes the HFPGA architecture. Section 3 briefly describes the CAD tools developed, and section 4 describes the experimental study and results. Section 5 concludes the paper.

2 Architectural Model for HFPGAs

This section provides a brief description of the HFPGA architecture. A more detailed discussion of the architecture can be found in [1][11]. HFPGAs are built using 2 types of primitive blocks: logic block (or level 0 block) and I/O block. This research uses a 4-input lookup table (4-LUT) as the basis for the logic block, although the HPFGA architecture is not dependent on the type of logic block used.

A level 1 block contains a collection of N_0 level 0 logic blocks and M_0 I/O blocks equally placed on both sides of the routing channel as shown in Figure 2.1. Local routing tracks are used to connect cells within the same block while global routing tracks can make connections outside the block. This architecture can be easily extended to arbitrary levels of hierarchy. A level 2 logic block consists of N_1 level 1 logic blocks with local tracks within the block as well as global tracks connecting to routing tracks in a level 3 logic block. The configuration of a *n* level

HFPGA can be described using the expression $N_{n-1} \times \ldots \times N_2 \times N_1 \times (N_0 + M_0)$ An example of a

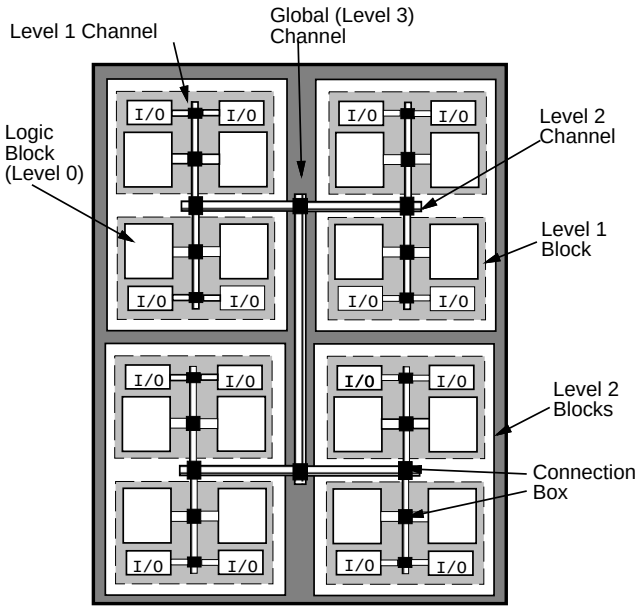

Figure 2.2 Example of a 4 x 2 x (2 + 2) Hierarchical FPGAHierarchical

$4 \times 2 \times (2 + 2)$ architecture is shown in Figure 2.2.

2.1 Switch Patterns

HFPGAs contain connection boxes to connect the pins of a level *i* logic block to another level *i* logic block or to the pins of the level $(i + 1)$ blocks in a higher hierarchy. The connection boxes use a non-uniform switch topology, in which the switches are distributed unevenly amongst the routing tracks using an exponential function. The routing flexibility F_i for level *i* channel is defined as the average number of switches on each level *i* track to which a level $(i - 1)$ track can connect.

2.2 Direct Interconnect

This paper extends the architecture of Aggarwal [1] by exploring the use of direct interconnects on HFPGAs. In order to provide some ability to reduce path delay, fast connections are provided between adjacent logic blocks. Nets that use direct interconnect have reduced interconnect delay and use no general routing resources (local or global tracks). There are two types of direct interconnects: level 1 direct interconnect and level 2 direct interconnect.

- level 1 direct interconnect - the output pin of a 4-LUT is connected to the input pins of its adjacent cells through the use of the optional level 1 direct interconnect to form a linear chain of interconnected 4-LUTs within the same level 1 block. The input pin still has the ability to connect to the general routing resources within the same block as shown in Figure 2.3.
- level 2 direct interconnect - the linear chain of interconnected LUTs in a level 1 block extends to level 2 block by allowing the output pin of the two 4-LUTs at both ends of the linear chain to connect to the input pin of

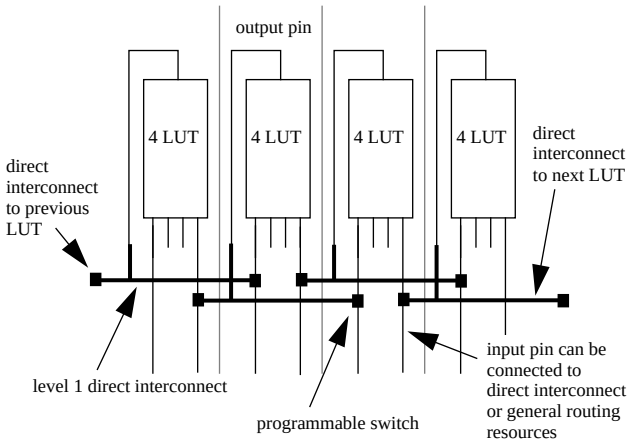

Figure 2.3 Level 1 direct interconnect (linear chain)

the 4-LUT at the end of the adjacent level 1 block through a level 2 direct interconnect channel

The direct interconnect structure can be extended to more than two levels, although this paper only studies the effects of direct interconnect up to two levels. Direct interconnect provides connections between logic blocks, but not to I/O cells. The direct interconnect structure for a $4 \times (6 + 2)$ architecture is shown in Figure 2.4 . To simplify the picture, the direct interconnects between two cells are represented as one wire, even though there are actually two wires, one in each directions, connecting two cells together

Software Tools

This section describes briefly the algorithms used in the software tools developed for timing driven placement of a netlist on HFPGAs. A more detailed discussion of these algorithms can be

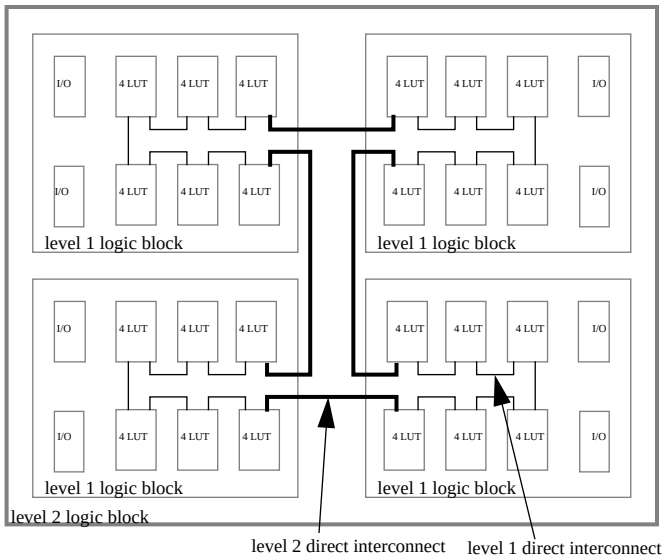

Figure 2.4 Direct interconnect structure of a 4 x (6 + 2) architecture

found in [11].

2.3 Timing Driven Placement (TDPlacement)

Placement is performed using *TDPlacement*, which takes the same approach as the original Fiduccia-Mattheyes min-cut [15] algorithm but incorporates timing in the cost function, with the following modifications:

- A topological timing analysis and path dependency analysis is performed before the placement.
- After each "cell move", the algorithm updates the timing slacks of all edges and re-calculates the gain for each cell.

In topological timing analysis (*TAnnotate*), for each cell *v*, the actual arrival time, $t_a(v)$, and the required arrival time, $t_r(v)$, from the primary input cells to the output pin of the cell are computed. After finding the required arrival time for all cells in the circuit, the path slack for the edge $e = (v_i, v_j)$ can be calculated by subtracting the accumulated delay for a signal to propagate to v_j through the edge *e* from the required arrival time of v_j, i.e.

$$ps(e) = t_r(v_j) - t_a(v_i) - \Delta(v_i, v_j) - \sigma(v_j)$$

where $\sigma(v_i)$ is the intrinsic delay of the logic cells v_i and $\Delta(v_i, v_j)$ is the signal propagation delay from v_i to v_j.

In an HFPGA, an even number of switches are required to connect any two cells together, thus, it is convenient to express the path slack in term of the number of switch pairs. The algorithm computes the timing slack for an edge, *ts*(*e*), by dividing the path slack of the edge with the delay of two switches, i.e.

$$ts(e) = \frac{ps(e)}{2 \times sw}$$

where *sw* is the delay through a programmable switch.

The next step in the placement process is to identify the set of edges whose path slacks are affected by any change in the path slack of e through path dependency analysis. We denote such set Φ_e, the affected edges set of e. In the algorithm, the affected edges set of an edge is used to indicate the importance of a timing slack change for the edge.

The algorithm then uses the timing slack associated with each edge to define the cost of introducing switch pairs to the edge. The cost of the edge is an indicator of the effect of adding a pair of routing switches on the delay of all path through the edge; the higher the cost for an edge, the more effect it has on path delay.

Whenever a cell is moved from one partition to another partition during the placement, the timing slack of each affected edge must be updated to reflect the current timing properties of the circuit. After the update, the cost for all edges and cells are re-calculated to reflect the change in timing slack.

2.4 Direct Interconnect Optimization (*DICOpt*)

The direct interconnect optimizer reads in the placed design after timing driven placement and re-arranges the placement of the logic cells to take advantage of the direct interconnects that exist between adjacent logic cells.

Topological timing analysis and path dependency analysis are performed in the beginning to determine the longest path delay and timing slacks for all edges. Since direct interconnects do not exist for I/O cells, the optimizer will leave the I/O cells placement unchanged while all the placement for logic cells will be treated as "floating" within the same level 1 block. The algorithm uses a greedy approach by first picking the highest cost edge and attempts to place the two cells connected by the edge adjacent to each other; it then proceeds to pick the next highest cost edge until all cells are placed. In addition, after each successful placement, the algorithm will update the timing slack on all the affected edges and the cost for each edge will be re-calculated.

3 Experimental Results

This section describes the experiments and the approach used to study the timing characteristics of HFPGAs. These experiments studied the timing behavior of HFPGAs based on the longest path delay of the test circuits placed using the timing driven placement tools developed. In addition, these experiments also studied the effects of direct interconnects on circuit timing and switch counts. The results were then compared against those obtained for symmetrical FPGAs.

In order to study the timing characteristics of HFPGAs, seventeen circuits from the MCNC test suites were placed on nine different HFPGA architectures. The results of these experiments revealed that circuits using HFPGA can achieve a lower longest path delay than those using symmetrical FPGAs.

3.1 Average Routing Delay

In this paper, we express the routing delay along any path in terms of the average number of programmable switch pairs, denoted by ρ. Let Ψ by a path defined by the sequence $(v_0, \ldots, v_n)$, the delay for the path Ψ can be expressed as

$$D(\Psi) = \sum_{i=0\ldots n} \sigma(v_i) + \sum_{i=0\ldots n-1} \Delta(v_i, v_{i+1})$$

and the routing delay of the path, $\rho(\Psi)$, can be normalized in terms of routing switch delays as

$$\rho(\Psi) = \frac{\sum_{i=0\ldots n-1} \Delta(v_i, v_{i+1})}{2 \times sw}$$

where *sw* is the switch delay.

In the experiments, the same programmable switch delay and direct interconnect delay are assumed for both HFPGAs and symmetrical FPGAs. Since an HFPGA requires a minimum of two switches are required to connect any two cells together, ρ has a lower bound of one. For symmetrical FPGAs, ρ has a lower bound of one if the two cells are adjacent to each other, otherwise, ρ has a lower bound of 1.5 since a minimum of two switches and a connection box are required to connect two cells together. In general, if HFPGAs can achieve a value of ρ less than 1.5, their speed is better than that of symmetrical FPGAs. However, the value of ρ may be less than one for both symmetrical FPGAs and HPFGAs if direct interconnects are utilized. In the experiments, we use ρ to refer to the normalized delay of the critical path.

It should be noted that the use of switch count for delay is limited in its accuracy as it neglects the loading on each line due to other switches' capacitance.

3.2 Routing Resource Requirements

In this paper, the routing resources required for a given architecture are measured in terms of the number of switches required per logic block (or switch counts), denoted by α. The area of the metal tracks is neglected as the area occupied by the routing switches typically exceeds that of the metal tracks. For an HFPGA architecture described by the expression $N_2 \times N_1 \times (N_0 + M_0)$, α is defined as

Table 3.1 Benchmark Circuits

Circuit	#Cells	#I/O	#Nets
9symml	71	10	80
C3540	352	72	402
C432	106	43	142
C880	114	86	174
apex7	78	86	127
dalu	333	91	408
i2	69	202	270
i5	66	199	199
i6	110	205	248
i7	175	266	374
i8	350	214	483
i9	200	151	288
k2	353	88	398
rot	238	242	373
x1	120	86	171
x3	268	234	403
x4	126	165	220

$$\alpha = \frac{tsw}{N_2 \times N_1 \times N_0}$$

where *tsw* is the total number of programmable switches.

In general, better area efficiency is obtained from an FPGA with a smaller value of α and better speed is obtained from an FPGA having a smaller value of ρ.

3.3 Benchmark Circuits

17 circuits from the MCNC test suites [2] were used throughout the experiments presented in this chapter. The size of the circuits are listed in Table 4.1

3.4 Experimental Assumptions

A number of assumptions were made in both the HFPGA model and the symmetrical FPGA model used in the experiments described in this chapter.

i) Only HFPGAs with three levels of hierarchy were used in the experiments, as they have been shown [1] to be a good size for the set of benchmark circuits used in experiments.

ii) A 4-input LUT was used as the basic logic block .

iii) All four input pins LUTare considered to be logically equivalent.

iv) The number of I/O blocks is allowed to "float" such that the ratio of logic blocks to I/O blocks is approximately equal to the ratio of logic cells and I/Os for the specific design. This restriction is made to minimize the number of unused logic blocks.

v) Direct interconnects are implemented as programmable switches for both HFPGAs and symmetrical FPGAs. In addition, the delay through a direct interconnect is considered to be half that of a regular programmable switch.

vi) Routing parameters $F_1 = 2$, $F_2 = 2$, $F_3 = 4$ and $W_x = 3$, determined in [1] to lead to 100% routing, were used. These parameters describe, respectively, the number of routing switches connecting each pin to each track in the channel at levels 1, 2, and 3, and the number of excess tracks allocated beyond the channel density. The routing resource usage of the symmetrical FPGA (non-timing driven) was calculated by CGE [9] during detailed routing. The switch counts for timing driven placement for symmetrical FPGAs was extracted from Xilinx's data sheet.

With the above assumptions, the HFPGA models used in the experiments can be described by choosing specific values of N_0 and N_1, and using a size of HFPGA large enough to contain the circuit, by setting

$$N_2 = 2 \times \left\lceil \frac{\#\text{LUT}}{2 \times N_0 \times N_1} \right\rceil$$

and determining IO requirements as

$$M_0 = 2 \times \left\lceil \frac{N_0 \times \#\text{LUT}}{2 \times \#\text{IO}} \right\rceil$$

where #LUT is the number of LUTs, and #IO is the number of primary input and output cells used in a specific circuit.

3.5 Experimental Procedures

The experiments described in this section used 17 circuits from the MCNC suite to determine the area and speed effects of the architectural parameters, as well as compare HFPGAs to symmetrical FPGAs. The experimental procedures for HFPGAs and symmetrical FPGAs were:

i) Technology independent optimization using *SIS* [12] .

ii) Technology mapping into 4-input LUT logic blocks using *chortle-crf* [13].

iii) Placement: For HFPGA, non-timing driven placement was performed by *HPlacement* [1]. For symmetrical FPGA, non-timing driven placement is performed by *Xaltor* [14].

iv) Timing driven placement using *TDPlacement* HFPGAs, and Xilinx's *apr* [10] for symmetrical FPGAs.

v) Direct interconnect optimization using *DICOpt* for HFPGAs, and Xilinx's *apr* [10] for symmetrical FPGAs.

vi) Timing Analysis using *TAnnotate* for HFPGAs, and Xilinx's *xdelay* [10] for symmetrical FPGAs.

3.6 Timing Performance of different HFPGA architectures

This section presents the results of the experimental studies conducted to understand the timing performance of the different

HFPGA architectures.

Experiments were performed on seventeen (17) MCNC benchmark circuits on nine (9) different combinations of $N_1 \times N_0$, specifically, 4×2, 4×4, 4×8, 4×16, 4×32, 8×4, 8×8, 8×16 and 16×4. The value of N_2 and M_0 are chosen for each circuit so as to provide maximum utilization of logic cells. Thus, a total of 153 independent circuit-architecture combinations are tested and three different kinds of placements are performed for each combination:

- non-timing driven placement,
- timing driven placement, and
- timing driven placement followed by direct interconnect optimization.

The values of α and ρ were computed from each final placement and the conclusions drawn from these results will be discussed.

Figure 3.1 shows a comparison of ρ for the three placements

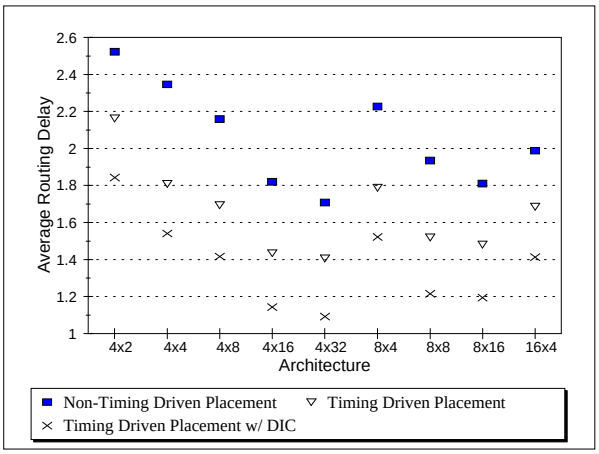

Figure 3.1 Average Routing Delay verses Architecture

and nine different architectures.

Effects of different HFPGA architectures

For the three different placement procedures, the average value of ρ decreases when the value of N_1 and N_0 increase. The lowest value of ρ is achieved with the architecture 4×32. In fact, the average value of ρ is decreased by 32.14%, 34.72% and 40.76% respectively when a 4×32 architecture is used instead of a 4×2 architecture for the three placements. This is because a larger N_0 allows more cells to be connected by using only two switches, and a larger N_1 will reduce the number of connections that require global tracks.

Effects of different placements

From the table, timing driven placement has 19.12% lower delay in the critical path than non-timing driven placement and 33.38% lower if direct interconnect optimization is used.

3.7 Area Performance of different HFPGA architectures

Optimizing placement for timing may have the effect of increasing routing channel requirements. The next set of

Table 3.2 Switch Counts for different HFPGA architectures

Architecture	α unopt	α opt	α opt, DIC
4x2	165.18	169.06	169.35
4x4	137.65	147.29	145.47
4x8	150.29	171.59	164.71
4x16	210.06	236.35	215.35
4x32	272.06	307.47	272.41
8x4	127.06	142.35	140.41
8x8	151.53	164.18	156.18
8x16	205.88	209.82	190.18
16x4	133.35	143.88	142.82
Average	172.56	188.00	177.43

experiments investigate this effect. Figure 3.2 Table 3.2 show a

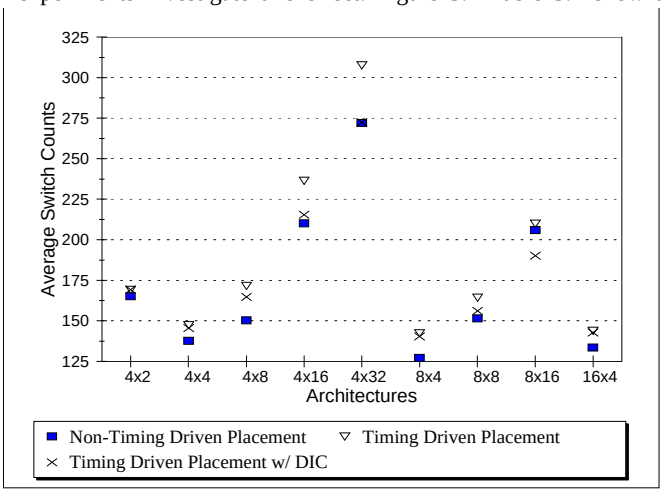

Figure 3.2 Average Switch Counts verses Architectures

comparison of α for the three placements and nine different architectures.

Effects of different HFPGA architecture

For the five architectures with $N_1 = 4$, the smallest value of α is obtained for the architecture 4×4 and increases gradually as the value of N_0 changes. Overall, the architecture 8×4 gives the lowest switch counts among the nine different architectures. Basically, two opposing factors determine the effect of logic block size on switch counts:

i) As the logic block size (N_0) increases, more routing switches are wasted due to routing of a net.

ii) As N_0 decreases, the number of connections using higher level routing channels increases. The penalty for a net passing through every additional higher level channel is two routing switches.

Effects of different placement algorithms

Among the three placement procedures, the minimum value of α is achieved by non-timing driven placement. The results from timing driven placement are on the average 8.81% higher than non-timing driven placement. This is because timing driven placement attempts to place cells along critical paths closer to each others and may increase the number of switches required by non-critical nets. On the other hand, direct interconnects can reduce the amount of general routing resources usage, thus, the switch count is only increased by 3.72%, an average of 5.09% less than that required by timing driven placement.

3.8 Routing Delay verses Switch Counts

Figure 3.3 shows the average routing delay (ρ) against the average switch counts (α) for the nine different architectures on the three different placements. In the graph, results from the non-timing

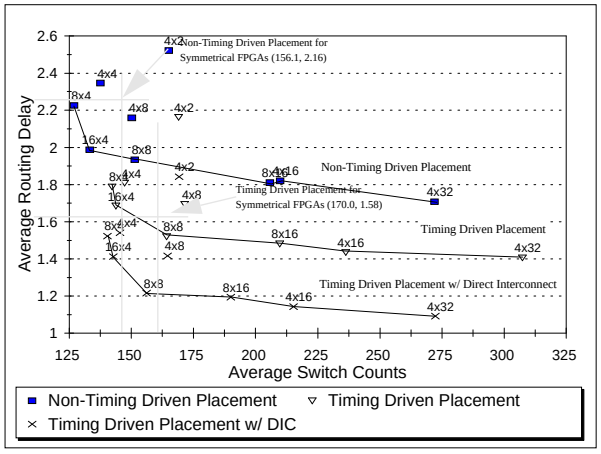

Figure 3.3 Average Routing Delay verses Average Switch Counts

driven placement, timing driven placement and timing driven placement with direct interconnect optimization are displayed; in addition, the Pareto set[1] containing data points in the lower bound for each type of placement is connected together by lines. The architecture 8×4 has the best area while the architecture 4×32 has the best speed for all three different placement procedures. We can also observe that, for the same HFPGA architecture, timing driven placement with direct interconnect optimization offers the greatest improvement in the longest path delay of the circuit with only a small increase in the switch counts. While the Pareto set initially has a steep slope indicating good speed gain for a small area increase, it then tails off indicating that a relatively large increase in the switch counts is required for a small further gain in speed.

1. Given a set of data, the Pareto set is the set of points that are better in at least area or speed than all of the other points.

Table 3.3 ρ for α HFPGAs (subset), and Symm. FPGAs

Circuit	ρ	α
4x2	1.82	163.5
4x4	1.46	136.1
4x8	1.36	147.5
4x16	1.08	191.0
4x32	0.96	232.2
8x4	1.49	129.5
8x8	1.15	144.6
8x16	1.01	178.5
16x4	1.37	135.8
non-timing driven for FPGAs	2.16	156.1
timing driven for FPGAs	1.58	170.0

Section 3.9 investigates the timing performance of symmetrical FPGAs and HFPGAs.

3.9 Comparison of Routing Delay for Symmetrical and Hierarchical FPGAs

In this section, the timing performance of symmetrical FPGAs and HFPGAs are compared. The results show that HFPGAs outperform symmetrical FPGAs in all but the 4×2 architecture.

The experiments compared the timing performance of ten (10) MCNC benchmark circuits on symmetrical FPGAs. Only ten circuits[2] were used because seven out of the original seventeen circuits will not fit into any Xilinx XC3000 parts due to the large number of I/Os on these circuits. For the ten circuits, both non-timing driven and timing driven placement were performed. The delay along the longest path was then calculated by assuming that:

- direct interconnects exist to connect a cell to its four adjacent cells, and
- double-length lines are used to connect CLBs not adjacent to each others.

Table 3.3 shows the routing delay (expressed as ρ) for timing driven placement (with direct interconnect optimization) of HFPGAs with timing and non-timing driven placement of symmetrical FPGAs. It also shows the switch counts (expressed as α) for the timing driven placement with direct interconnect optimization of HFPGAs, as well as the switch counts for timing and non-timing driven placement of symmetrical FPGAs. The switch counts for symmetrical FPGAs and routing delay were extracted from the results of CGE [9] after detailed routing and the switch count for symmetrical FPGAs with timing optimization was estimated from [10].

From the table, the average routing delay for timing driven

2. The 10 benchmarks are: 9symml, C3540, C342, C880, apex7, dalu, i9, k2, x1 and x4.

placement of symmetrical FPGAs has a value of 1.58 even when direct interconnects are used. The performance for symmetrical FPGAs non-timing driven placement is 18.7% slower than that of 4×2, the slowest in all HFPGAs architectures tested. While the performance of timing driven placement for symmetrical FPGAs is 13.19% faster than the 4×2 architecture, it is slower than all other HFPGA architectures (in fact, it is 64.6% slower than 4×32). These results are based on using optimized commercial software for the symmetrical FPGAs, but are subject to the limitations of the simple delay model used for the HFPGAs.

The average switch counts for non-timing driven placement of symmetrical FPGAs has a value of 156.1. The performance for symmetrical FPGAs non-timing driven placement is worse than five of the nine different HFPGA architectures. In fact, it is 20.5% higher than the switch count of the 8×4 architecture, the architecture with the best area performance for HFPGAs.

For timing driven placement of symmetrical FPGAs, the switch counts is larger than six of the nine different HFPGA architectures tested. Overall, it is 31.3% higher than that of the 8×4 architecture.

4 Conclusions

This paper demonstrates that HFPGAs are both faster and denser than symmetrical FPGAs. In addition, the following conclusions on the HFPGA architecture can be drawn.

i) Switch counts for timing driven placement are moderately higher than that required by non-timing driven placement for HFPGAs.

ii) The use of direct interconnects can reduce the routing switch requirement as well as provide a further improvement in timing.

iii) Hierarchical routing architectures with larger lower level blocks are faster than architectures having smaller lower level blocks. Routing switch requirements also increase as the lower level block size is increased

iv) While moderate sized low level blocks offer good speed, beyond this, a relatively large increase in the switch counts is required for a small gain in speed.

5 References

[1] Aditya A. Aggarwal, “Routing Architectures for Hierarchical FPGAs”, M.A.Sc. Thesis, Department of Electrical and Computer Engineering, University of Toronto, 1994.

[2] ISCAS ‘85 Test Generation Benchmark Data obtained from the Microelectronics Centre of North Carolina (MCNC)

[3] S. Brown, R. J. Francis, J. Rose and Z. G. Vranesic, Field-Programmable Gate Arrays, Kluwer Academic Publishers, 1992.

[4] S. Singh, J. Rose, D. Lewis, K. Chung and P. Chow, “Optimization of Field-Programmable Gate Array Logic Block Architecture for Speed”, Proceedings of the 1991 Custom Integrated Circuits Conference (CICC-91), May 1991, pp. 6.1.1-6.1.6.

[5] J. Rose, R. J. Francis, D. Lewis, P. Chow, “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency”, IEEE Journal of Solid State Circuits (JSSC), Vol. 25, No. 5, Oct. 1990, pp. 1217-1225.

[6] S. Singh, “The Effect of Logic Block Architecture on FPGA Performance”, M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto, 1991.

[7] K. Chung, S. Singh, J. Rose and P. Chow, “Using Hierarchical Logic Blocks to Improve the Speed of Field-Programmable Gate Arrays”, Proceedings of the First International Workshop on Field Programmable Logic and Applications, Oxford, Sept. 1991, pp. 103-113.

[8] J. Rose and S. Brown, “Flexibility of Interconnection Structures in Field Programmable Gate Arrays”, IEEE Journal of Solid State Circuits (JSSC), Vol. 26, No. 3, March 1991, pp. 277-282.

[9] S. D. Brown, “Routing Algorithms and Architectures for Field Programmable Gate Arrays”, PhD. Thesis, University of Toronto, 1992.

[10] The Programmable Logic Data Book, Xilinx Inc., 1994.

[11] Vi C. Chan, “Timing Optimization for Hierarchical FPGAs”, M.A.Sc. Thesis, Department of Electrical and Computer Engineering, University of Toronto, 1995.

[12] R. Brayton, R. Rudell, A. Sangiovanni Vincentelli and A. Wang, “MIS: a Multiple-Level Logic Optimization System”, IEEE Transactions on CAD (TCAD), Vol CAD-6, No. 6, Nov. 1987, pp. 1062-1081.

[13] R. J. Francis, J. Rose and Z. Vranesic, “Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs”, Proceedings of the 28th Design Automation Conference (DAC-28), June 1991, pp. 227-233.

[14] J. Rose, Z. Vranesic and W. M. Snelgrove, “ALTOR: An automatic standard cell layout program”, in Proceeding of the Canadian Conference on VLSI, Nov. 1985, pp. 168-173.

[15] Fiduccia, C.M., and Mattheyses, R.M., “A linear-time Heuristics for Improving Network Partitions”, Proceedings of the 19th Design Automation Conference (DAC-19), 1982, pp. 175-181.

[16] B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs”, Bell System Tech. Journal, 49, 2, 1970, pp. 291-308.

[17] Wilm E. Donath, Reini, J. Norman, Bhuwan K. Agrawal, et. al, “Timing Driven Placement Using Complete Path Delays”, 27th ACM/IEEE Design Automation Conference, 1990, pp. 84-89.

[18] P. S. Hauge, R. Nair and E. J. Yoffa, “Circuit Placement for Predictable Performance”, IEEE Proc. of ICCAD, 1987, pp. 88-91.

[19] Michael Burstein, Mary N. Youssef, “Timing Influenced Layout Design”, 22nd Design Automation Conference, 1985, pp. 124-130.

[20] Grant A. Cheston, “Incremental Algorithms in Graph Theory”, Ph. D. Thesis, Department of Computer Science, University of Toronto, 1976.

[21] Altera Corp., “Flex 8000 Handbook”, 1994