Nearest Neighbour Interconnect Architecture in Deep Submicron FPGAs

Ajay Roopchansingh and Jonathan Rose

Edward S. Rogers Sr. Dept. of Electrical & Computer Engineering, University of Toronto
Toronto, Ontario, Canada, M5S 3G4
{ajaytr, jayar}@eecg.utoronto.ca

Abstract

Several commercial FPGA architectures provide fast connections between adjacent logic blocks that decrease the *best-case* delay between circuit elements with the goal of increasing overall performance. This paper explores the architecture of these *Nearest Neighbour (NN) interconnects* to determine topologies, quantities and distances that are best for performance and area. We show that certain architectures can achieve a 7.4% performance improvement at the cost of a 6.3% increase in total FPGA area when fully populated. We also show that a 6.4% improvement can be achieved for a more modest cost of 3.8% increase in area.

1 Introduction

Performance is a key issue in the design and use of FPGAs. Depending on the architecture, 60% to 80% of the FPGA critical path delay is due to the routing between logic blocks [1][7], thus motivating our research into new and faster routing architectures.

NN interconnects are direct connections between a BLE (Basic Logic Element) output and the input multiplexer of another logic block, bypassing all the intermediate routing switches as illustrated in Figure 1.

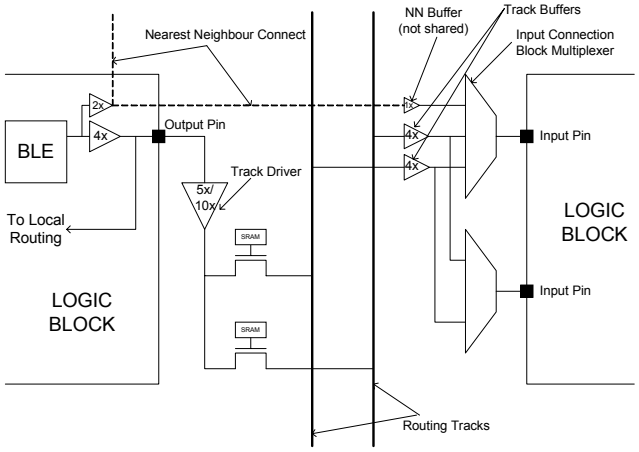

Figure 1 – Nearest Neighbour Interconnect

Although several commercial architectures [11-13] have employed NN interconnects, there are no published studies that investigate NN interconnect quantity or patterns and their speed and area tradeoffs. We are interested in the following properties of NN interconnect architectures:

1. *Topology*: This describes the pattern which dictates which neighbouring logic blocks a given block can connect to.
2. *Distance/Radius*: This describes the distance of the neighbouring logic blocks to which NN interconnects from a given block can extend to.
3. *Quantity*: This is the number of NN interconnects present in a given topology at a given radius.

NN interconnects have been used in several commercial programmable logic architectures. The Algotronix CAL1024 [11] used NN interconnects as the entire routing fabric. Function units could *only* connect to their immediate North, South, East and West (which we refer to as *Manhattan* directional) neighbours. The Xilinx 3000 [13], Xilinx XC6200 [13] and Atmel AT6000 [12] use heterogeneous routing schemes in which NN connections are used in tandem with other types of routing resources. The NN interconnects in these FPGAs employ a pattern similar to the CAL1024, connecting only to their immediate Manhattan neighbours. The more recent Atmel AT40K [12] and Xilinx Virtex II architectures [13], employ a slightly different pattern where they connect to the immediate diagonal logic blocks as well as the Manhattan. This paper explores a range of these architectures, including those used in present-day FPGAs.

This paper is organized as follows: Section 2 outlines the basic FPGA architecture and parameterizes the NN interconnect architectural space that we seek to explore. Section 3 presents the experimental procedure and results of our experiments. Section 4 gives our conclusions.

2 Architecture

Here we describe the basic FPGA architecture that we explore and how it is modified to include NN interconnects.

The basic FPGA uses a symmetric *island-style* architecture [1]. Each logic block or logic cluster is made up of 4 basic logic elements (BLEs) and is fed by 10 cluster inputs. Each BLE consists of a 4-input lookup table (LUT) and a register. A 2-input mux is used to provide either a registered or unregistered BLE output. Figure 2 shows the general schematic for a logic cluster and BLE. We further assume that the cluster is "fully connected", which means that all 10

cluster inputs and 4 BLE outputs can connect to each of the 4 inputs on every LUT. Each of the cluster inputs can connect to 60% of the tracks in the channel adjacent to it and each of the 4 cluster outputs connects to 25% of the tracks in the channel adjacent to them.

The routing channel uses metal wires that span 4 clusters (length 4 tracks). Of these tracks, 50% are buffered and 50% are switched by pass transistors. The routing channel switch block uses a *disjoint* topology, meaning that once signals are routed onto pass-transistor-switched tracks, they can connect *only* to other pass-transistor-switched tracks. The same topology is used for switching buffered tracks. Finally, at a switch-block, each track can be switched onto 3 other tracks.

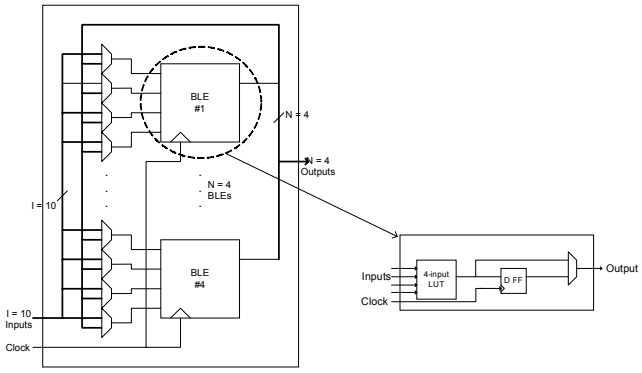

Figure 2 - Logic Cluster & BLE

The architecture is modelled in a 0.18um CMOS process. Further architectural details can be found in [2].

2.1 Circuit Design of NN Connects

Figure 1 illustrates the circuit design of an NN interconnect. These are faster than connections made through the usual routing fabric for 3 reasons:

1. The buffer driving an NN interconnect is smaller than the driver for the output pin since it drives roughly half the fan-out. The output pin driver is sized as a 4x buffer since it must drive four LUTs with 4 inputs each for a total of 16 LUT inputs (feedback to the cluster's local routing). This has been shown to produce good area and delay results [1]. In the NN interconnect architectures we explore, the maximum fan-out of an NN interconnect is 8, and so its buffer only needs to be ½ the size. Furthermore, since NN interconnects travel only a short distance, a track driver is not needed. These two differences account for a significant portion of the delay improvement of an NN interconnect. The detailed circuit showing buffer sizing is shown in Figure 1.
2. The conventional routing fabric has a higher capacitive load than an NN interconnect wire because there are many routing switches connected to each general routing track.
3. The NN buffer is also smaller since a track buffer must fan out to any mux that connects that track to an input pin (which can be up to 4 in this basic architecture), whereas the NN buffer is dedicated as shown in Figure 1.

2.2 Parameterization of NN Architecture

The *NN Topology* describes the fan-out pattern of the NN interconnects. In a *Manhattan* topology, NN interconnects go to neighbours that are North, South, East and West of a given logic cluster. In a *Cross* topology, they go to neighbours that are NE, SE, SW and NW of a given cluster. In a *Full* topology, they go to all of these surrounding neighbours. Figure 3 illustrates these 3 basic topologies.

The *NN Radius* parameter dictates the distance to the neighbour that the NN interconnect connects to. Figure 3 illustrates Radius 1 connections for the three basic topologies since they connect only to immediate neighbours. Notice that neighbours in the NE, SE, SW and NW directions are still considered radius 1. The same principle is applied when extending to larger radii.

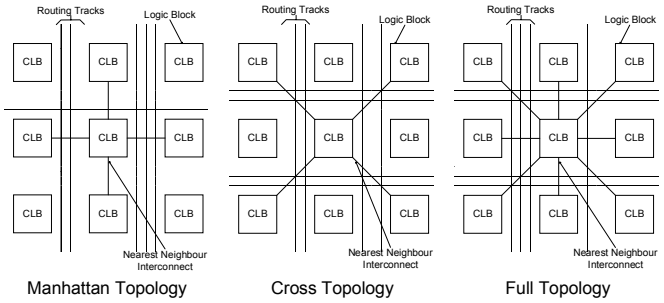

Figure 3 – Various NN Topologies

Finally, we must also specify the *quantity* of NN interconnects that exist in a specified topology at a given radius. Note that each topology, at a given radius, will have a limit to the number of NN interconnects that can be present. For example, since our architecture has only 4 output pins per logic cluster, a Manhattan Radius 1 topology can have at most 16 NN interconnects (4 outputs each fanning out in 4 directions). The same is true for the Cross topology. Note also that the Manhattan and Cross topologies will have the same maximum number of NN interconnects for each radius. In a Full Radius 1 topology however, we have 4 outputs each fanning out in 8 directions for a total of 32 possible NN interconnects. For a Full radius 2, we have 4 outputs each fanning out in 16 directions = 64 possible NN interconnects.

These 3 parameters, *Topology*, *Radius* and *Quantity* define the architectural space that we explore in this research.

3 Results

To evaluate new FPGA architectures, we synthesize real circuits, using the CAD flow described in [1][7], into the architecture and then measure the resulting area and delay. The flow consists of logic synthesis [8], packing [5][9] and placement and routing using VPR [1]. In our experiments we

use the 20 largest MCNC [10] circuits and 8 new circuits created at the University of Toronto. These circuits range in sizes from 800 BLEs to 10,000 BLEs.

In this section we present experimental results obtained using the flow described above. The area metric is the sum of the total logic area plus the routing area for the smallest possible FPGA required to fit the circuit. It is measured in equivalent number of minimum-width transistor areas required to implement all elements of the FPGA in a 0.18um CMOS process.

The speed of an FPGA architecture is measured by first placing and routing the circuit to find the smallest possible FPGA required, then re-routing with an FPGA which has 20% more tracks. This models a real-life scenario since designers rarely push the limits of an FPGA but rather choose ones that are slightly larger than the design requires in order to create a *low-stress* routing environment [1]. Note that we present the average of five different runs for each circuit to reduce the experimental noise in the results.

For each architecture when the number of NN interconnects is greater than zero, we use a placement that is aware of their presence. This was shown to increase NN interconnect utilization and produces better results than placements that had no knowledge of the presence of NN interconnects [2].

3.1 Performance Results

In this section we show the effect of the various NN interconnect parameters on delay. Figure 4a shows how the average critical path delay across the 28 circuits varies as we increase the number of NN interconnects present in Manhattan and Cross Radius 1 architectures. Observe that increasing the number of NN interconnects decreases delay. For Manhattan and Cross architectures with 16 NN interconnects, the critical path delay is reduced by 5.3% and 4.8% respectively compared to an architecture with no NN interconnects. The increase in total area cost for these two architectures is 3.5% and 3.3% respectively (see [2] for more information on area).

Figure 4b shows results for the larger Manhattan and Cross Radius 1 & 2 and Full Radius 1 architectures. The fully populated versions of these architectures, (in which the number of NNs = 32), decreases the average critical path delay by 6.9%, 6.4% and 7.4% at the expense of a 6.8%, 6.4% and 6.8% increase in total area respectively. Table 1 summarizes the results.

Observing Figures 4a and 4b, we see that the majority of the delay reductions can be achieved without the expense of a fully populated architecture. Using fewer NN interconnects reduces the area penalty. For example, Figure 4a shows that, for the Manhattan Radius 1 architecture, most of the gain is achieved using only 10 of the maximum 16 NN connects. Figure 4b shows that using 20 NN interconnects for the Manhattan Radius 1 & 2 architecture and 18 NN interconnects for the Full Radius 1 architecture nets the majority of the delay reduction. Table 2 presents a selection of good delay and area results obtained by depopulating the NN interconnects of all architectures.

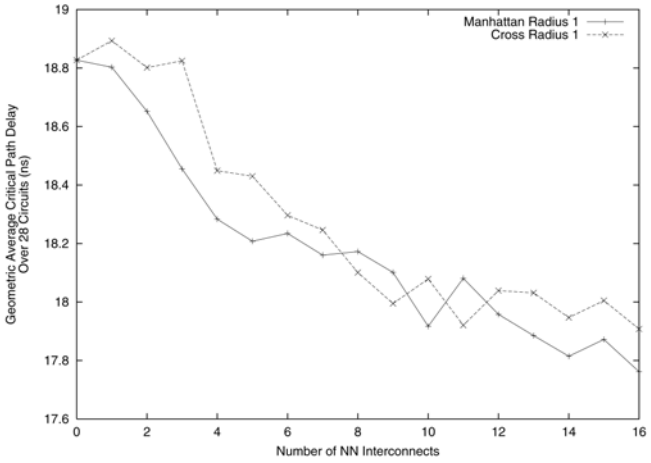

Figure 4a - Delay vs. # of NN Interconnects for Manhattan/Cross Radius 1

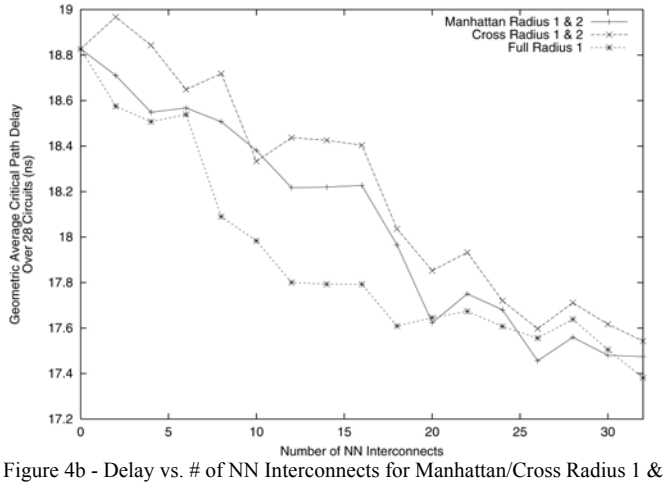

Figure 4b - Delay vs. # of NN Interconnects for Manhattan/Cross Radius 1 & 2 and Full Radius 1

Topology (Radius)	# of NNs	Avg. Delay (ns)	Change In Delay	Change In Total Area
Manhattan (1)	16	17.8	- 5.3 %	+ 3.5 %
Cross (1)	16	17.9	- 4.8 %	+ 3.3 %
Manhattan (1 & 2)	32	17.5	- 6.9 %	+ 6.8 %
Cross (1 & 2)	32	17.6	- 6.4 %	+ 6.4 %
Full (1)	32	17.4	- 7.4 %	+ 6.8 %

Table 1 – Delay and Area Results for Fully Populated Architectures

Topology (Radius)	# of NNs	Avg. Delay (ns)	Change In Delay	Change In Total Area
Manhattan (1)	10	17.9	- 4.8 %	+ 2.1 %
Cross (1)	9	18.0	- 4.3 %	+ 1.8 %
Manhattan (1 & 2)	20	17.6	- 6.4 %	+ 4.3 %
Cross (1 & 2)	26	17.6	- 6.4 %	+ 5.3 %
Full (1)	18	17.6	- 6.4 %	+ 3.8 %

Table 2 - Area & Delay Results for Partially Populated Architectures

3.2 Area vs. NN Interconnects

In the previous section's experiments, the number of tracks was held constant while the number of NN interconnects was increased. It is interesting to determine if NN interconnects can be used to reduce total area by permitting a reduction in the number of tracks required to route circuits. To determine this, we measured the minimum number of routing tracks per channel required to achieve routability while varying the number of NN interconnects. Note that this is a different scenario from the one above in Section 3.1, where all experiments are run using 20% more than the minimum track count required without NN interconnects.

Figure 5a shows how the area varies as we increase the number of NN interconnects for 3 architectures. Observe that the curves are flat for the first 9 to 12 NN interconnects. Here the area required to add NN interconnects is being balanced by the area saved in reducing track count. Figure 5b shows how the track count varies as we increase the number of NN interconnects in the various architectures.

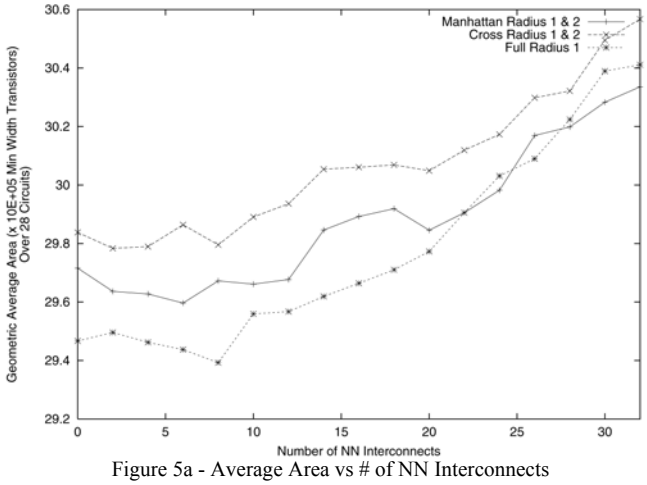

Figure 5a - Average Area vs # of NN Interconnects

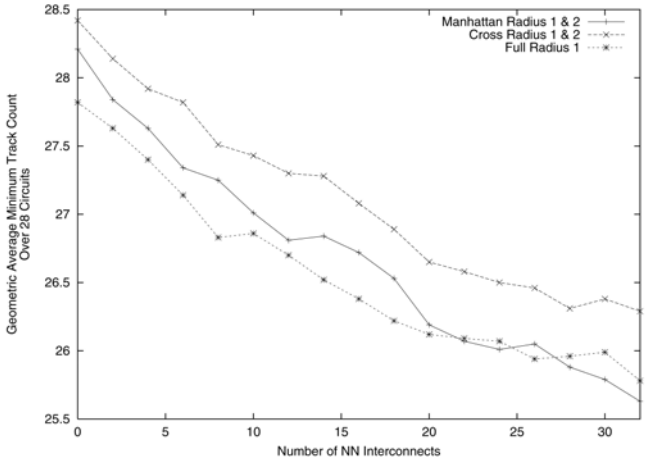

Figure 5b - Track Count vs # of NN Interconnects

It can be seen from Figure 5a that, for each architecture, a particular number of NN interconnects can be added for zero change in total area. Table 3 lists the number of NN interconnects that can be added for no change in total area and also shows the track-count reduction at that point.

Topology	# NNs That Achieve Same Area	Track Count Reduction
Manhattan Radius 1 & 2	12	5.0 %
Cross Radius 1 & 2	9	3.3 %
Full Radius 1	9	3.5 %

Table 3 – Area-Neutral Architectures that Contain NN Interconnects

4 Conclusions

We have introduced a set of Nearest Neighbour architectures and shown that they are able to achieve useful performance benefits for little cost in area. The best result was a Full Radius 1 architecture that achieved a 6.4% reduction in critical path delay at a cost of 3.8% increase in area.

We also showed that NN interconnects can be included in an architecture at zero total area cost. The best result shows that 12 NN interconnects can be included in a Manhattan topology in this manner.

Overall, we conclude that the Full Radius 1 architecture is best in terms of delay while the Manhattan Radius 1 & 2 architecture is best in terms of area.

References

(1) V. Betz, J. Rose, A. Marquardt, "Architecture & CAD For Deep-Submicron FPGAs", Kluwer Academic Publishers, 1999.

(2) A. Roopchansingh, "M.A.Sc. Thesis in progress: Nearest Neighbour Interconnect Architecture in Deep Submicron FPGAs", University of Toronto, 2002.

(3) J. Gray, T. Kean, "Configurable Hardware: A New Paradigm for Computation", March 1989.

(4) A. Marquardt, V. Betz, J. Rose, "Timing Driven Placement for FPGAs", Right Track CAD, 1999.

(5) A. Marquardt, "M.A.Sc. Thesis: Cluster-Based Architecture, Timing Driven Packing and Timing Driven Placement for FPGAs", University of Toronto, 1999.

(6) A. Marquardt, V.Betz, J.Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density", ACM/SIGDA FPGA 99, 1999.

(7) M. Sheng, J.Rose, "Mixing Buffers and Pass Transistor in FPGA Routing Architectures", 2000

(8) E.M. Sentovich et al, "SIS: A System for Sequential Circuit Analysis", Tech. Report No. UCB/ERL M92/41, University of California, Berkeley, 1990.

(9) J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", IEEE Trans. on CAD, Jan. 1994, pp.1-12.

(10) S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," Tech. Report, Microelectronics Centre of North Carolina, 1991

(11) Algotronix Ltd., "CAL1024 Datasheet", 1988.

(12) Atmel Datasheets, http://www.atmel.com/atmel/products/prod99.htm

(13) Xilinx Datasheets, http://www.xilinx.com/partinfo/databook.htm