Measuring and utilising the correlation between signal connectivity and signal positioning for FPGAs containing multi-bit building blocks

A. Ye and J. Rose

Abstract: As the logic capacity of field-programmable gate arrays (FPGAs) increases, there has been a corresponding increase in the variety of FPGA building blocks. From a mere collection of conventional logic blocks, FPGAs can now include digital signal processors, multipliers, multi-bit addressable memory cells and even processor cores. One of the common characteristics of these new building blocks is their multi-bit design, where each block is designed specifically to process several bits of data at a time. This multi-bit processing paradigm is significantly different from the single-bit processing design of the conventional FPGA logic blocks, as it creates differentiation in signals through its bussed structures. Consequently, the correlation between the positions of the signals in buses and the connectivity of these signals is examined. On the basis of correlation measurements, a multi-bit routing architecture is then proposed along with its routing tool. It is experimentally shown that, compared with the conventional routing architectures, the multi-bit architecture requires 6–12% less area to implement. In particular, it needs 27% fewer routing switches to connect its multi-bit blocks to their routing tracks and 18% less configuration memory to store the configuration information.

1 Introduction

Over the years, there has been a dramatic increase in the variety of field-programmable gate array (FPGA) building blocks. Evolving from a mere combination of simple logic blocks and bit-addressable memory cells, FPGAs can now include digital signal processing blocks, multipliers, multi-bit addressable memory and even processor cores. One significant difference between these new building blocks and the classical logic block design is in the way that they process data. On the one hand, the classical logic blocks are designed to process one bit of data at a time; on the other, the new building blocks are designed to process multiple bits of data simultaneously. Consequently, the use of these multi-bit processing elements presents new opportunities for exploiting datapath regularity.

In particular, although the conventional FPGA logic blocks are connected to FPGA routing through individual input and output signals, the inputs and outputs of the multi-bit building blocks can be grouped into buses. The positions of signals in these buses often strongly correlate to the way that they connect. This strong correlation between the connectivity and the physical positioning of signals can create opportunities for FPGA architects to selectively remove routing switches from the routing fabric of an FPGA and to share configuration memory; all, at the same time, maintaining the original routability of the architecture. As routing switches often consume a significant amount of FPGA area, reducing the total number of routing switches and their configuration memory in an FPGA can significantly increase its area efficiency, especially for implementing large arithmetic-intensive datapath circuits, including computer graphics, multimedia, digital signal processing and Internet routing applications.

Several FPGAs containing only multi-bit building blocks have been proposed over the years [1–12]. Although they come in a wide variety of routing architectures, in this work, we focus on the study of FPGAs containing segmented-style routing resources [13]. In particular, we empirically measure the correlation between signal connectivity and the position of these signals in buses for several datapath circuits. The correlation is then used to remove routing switches and to share configuration memory in order to increase the overall area efficiency of the routing fabric. Note that the primary reason for the choice of segmented-style routing resources is due to the fact that these are the building blocks of many state-of-the-art commercial FPGAs (including the Altera Flex, Stratix and Cyclone series [14, 15] and Xilinx 5200, Virtex and Spartan families [16] of FPGAs). With their ever-increasing logic capacity, commercial FPGAs are being increasingly used to implement large datapath-intensive applications.

For any new FPGA architecture, it is essential to have a set of automated design tools that can make effective use of its new architectural features. In particular, the sharing of configuration memory places new demands on computer-aided design (CAD) tools. As a result, a set of datapath-oriented CAD tools, including synthesis [17],

IEE Proceedings online no. 20050178

doi:10.1049/ip-cdt:20050178

Paper first received 1st November 2005 and in revised form 11th January 2006

A. Ye is with the Department of Electrical and Computer Engineering, Ryerson University, 350 Victoria Street, Toronto, Ontario, Canada M5B 2K3

J. Rose is with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 3G4

E-mail: aye@ee.ryerson.ca

packing [18], placement [19] and routing tools, have been developed at the University of Toronto. In this paper, we focus on the particular problem of automated routing. Routing for architectures containing configuration-memory sharing switches is more difficult than classical routing [13, 20–23], because the router has to properly model the new architectural features. Additionally, in order to effectively utilise these added features, the router has to preserve the regularity of datapath circuits throughout the routing process, while, at the same time, attempting to achieve the conventional routing objectives of minimising congestion and critical path delay. As a result, this paper presents a new routing algorithm that is an evolution of the classical negotiated-congestion (NC) routing algorithm [21]. The algorithm leverages many existing features of the classical router while incorporating several new metrics, which are designed to measure the regularity of datapath circuits, into the traditional cost functions of congestion and critical path delay.

While two previous papers [24, 25] have examined the area efficiency of datapath-oriented FPGAs and the work of Lemieux *et al.* [26] examined the issue of generating sparse crossbars for general purpose FPGAs without considering datapath regularity, this work advances the research in two fundamental ways. First, it is the first to statistically quantify the routing demand of datapath circuits – resulting in statistics that are valuable both for designing specialised datapath-oriented FPGAs and for improving the connectivity of multi-bit building blocks in conventional FPGAs. Secondly, this paper proposes a new routing algorithm for configuration memory sharing resources (which can be used to take advantage of the statistics in order to improve area efficiency). The present paper enhances a previous version [27] with an extended analysis on the architectural design space of the switch patterns that connect FPGA building blocks to their routing resources. This analysis formally defines the uniform distribution-nature of the previous switch pattern designs and exposes the vastness of the design space, which has been frequently overlooked in previous works.

2 Signal connectivity against signal positioning

As shown in Fig. 1, the input and output signals of a multi-bit block can be grouped into buses. Each signal in a bus can then be associated with a unique integer number indicating the bit position of the signal in the bus. Note that this bussed structure arises from the regularity of datapath circuits, where a datapath is created by duplicating a single sub-design, called a bit-slice, multiple times. For circuits implemented using conventional logic blocks, a majority of this regularity is often destroyed by the CAD tools during the optimisation process; for circuits implemented using the multi-bit blocks, in contrast, these bit-slices are routinely preserved for the purpose of multi-bit processing. As the primary purpose of FPGA routing is to provide connectivity between the input and output signals of various FPGA building blocks, it is important to examine the relationship between the connectivity and the bit positions of these signals.

As multi-bit blocks come in many different physical forms, this work uses a model that contains M logic clusters [13] to capture their common characteristics of bussed interface. The basic structure of a logic cluster is shown in Fig. 2. In the figure, each cluster contains a group of N four-input look-up tables (LUT) and N D-type flip-flops (DFF), where the LUTs and flip-flops are grouped into N basic logic elements (BLEs) as shown in Fig. 2*a*. In Fig. 2*b*, each logic cluster is shown to contain I input signals and N output signals. These signals are connected to the LUT inputs through a set of $I + N$ to 1 multiplexers. Finally, each logic cluster output is directly connected to a unique BLE output. Note that, in this work, N is always set to be 4 and I is always set to be 10, as these values have been previously shown to be among the most efficient [28, 29].

M logic clusters are then used to create a single multi-bit building block, where M is called the granularity of the block. Each cluster-level input/output signal is assigned to a unique M-bit wide input/output bus at the block level. Each of these buses, as shown in Fig. 3, is created by taking one unique input/output signal from each logic cluster. Consequently, for each multi-bit building block, there are I input buses and N output buses.

To map a datapath circuit onto the multi-bit blocks, the datapath-oriented packing algorithm as described in the work of Ye and Rose [18] is used. The algorithm first groups logic in each bit-slice into a series of logic clusters. It then groups M identical logic clusters from the neighbouring bit-slices into a multi-bit block. (Note that by preserving datapath regularity, carry chains in a datapath are also routinely preserved as inter-logic-cluster connections by the packing algorithm. Furthermore, logic that generates a set of carry connections is often grouped into the same multi-bit logic blocks.)

Once a circuit is mapped into multi-bit blocks, the connectivity between the multi-bit block input and output signals can be measured using two-terminal connections, where each connection consists of a connected pair of a multi-bit block input (called the sink of the connection)

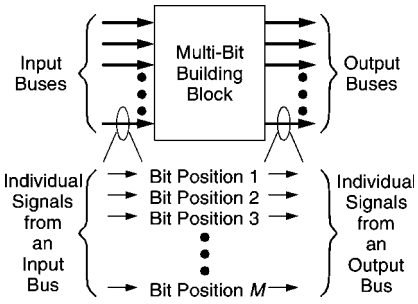

Fig. 1 *Multi-bit building block interface*

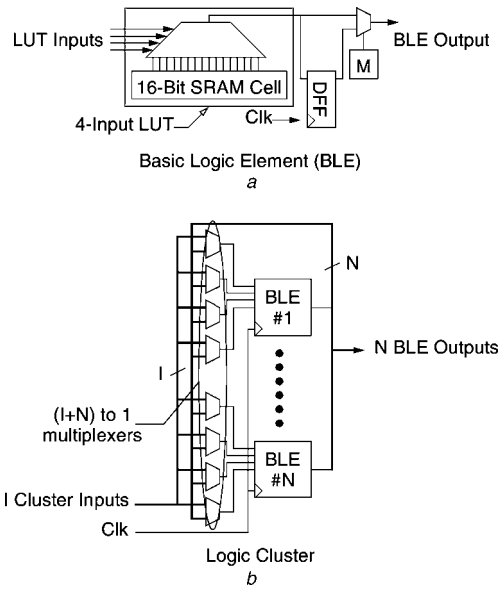

Fig. 2 *Logic cluster structure*

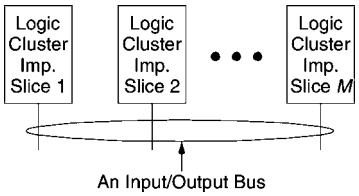

Fig. 3 *Modelling the bussed interface*

and a multi-bit block output signal (called the source of the connection). We then classify these two-terminal connections on the basis of the bit positions of their sources and sinks.

In particular, for the granularity value of two, all connections can be classified into four types as shown in Table 1: namely connections with both sources and sinks from bit position 1, connections with both sources and sinks from bit position 2, connections with sources from bit position 1 and sinks from bit position 2 and connections with sources from bit position 2 and sinks from bit position 1. The table then shows the number of connections in each type as a percentage of the total number of two-terminal connections for 15 benchmark circuits from the Pico-Java processor [30] in columns 2–5, respectively; column 6 shows the total number of two-terminal connections in each circuit. As shown, a majority (70%) of the connections in these benchmark circuits have the same source and sink bit positions. Connections with different source and sink bit positions, in contrast, consist of only 30% of the total number of two-terminal connections.

This strong correlation between the bit position difference and the signal connectivity is observed across all granularity values. Fig. 4 plots the difference against the number of two-terminal connections (as a percentage of the total number of two-terminal connections) for the granularity of 4, 8, 12, and 16. As shown, connections with the same source and sink bit positions (where the difference value is 0) consist of from over 30% (for $M = 12$ and $M = 16$) to over 70% (for $M = 2$) of the total

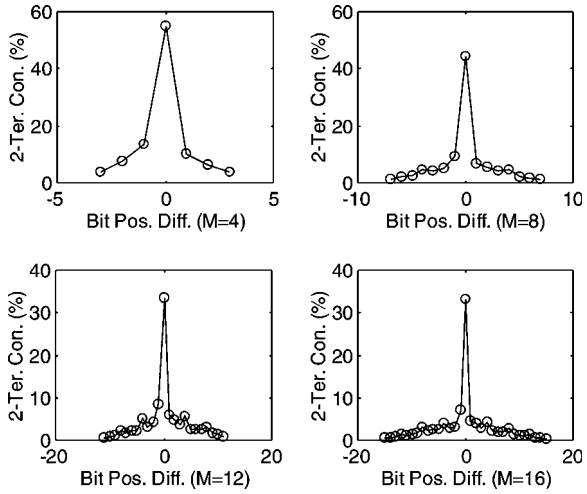

Fig. 4 *Correlation between bit-position difference and signal connectivity*

number of two-terminal connections. These percentage values are significantly greater than all other percentage values.

The correlation still exists at the level of individual bit positions. In particular, Fig. 5 shows the connectivity of multi-bit block output signals at bit position 1 in terms of the number of input signals that the outputs are connected to. As shown, for all granularity values, the output signals are connected to a significantly higher amount of input signals that are also from bit position 1 than inputs from any other bit positions. The same trend is observed for outputs from other bit positions (figures not shown because of space limitations).

The major architectural conclusion that can be drawn from these observations is that, for multi-bit blocks, a significant amount of connectivity should exist between the inputs and outputs that are from the same bit positions, and input and output signals from distinct bit positions, on the contrary, would require much less connectivity. Note that this observation is contrary to the connectivity

Table 1: Two-terminal connections for $M = 2$

Circuit name	Two-terminal connections				
	Source = 1, sink = 1, %	Source = 1, sink = 2, %	Source = 2, sink = 1, %	Source = 2, sink = 2, %	Per circuit total
code_seq_dp	40	22	12	26	873
dcu_dpath	39	11	11	39	2247
ex_dpath	38	13	12	37	7127
exponent_dp	30	19	20	30	1399
icu_dpath	39	15	11	35	8160
imdr_dpath	34	14	16	36	3030
incmod	31	25	19	26	2248
mantissa_dp	36	17	15	33	2554
multmod_dp	26	25	24	25	3645
pipe_dpath	40	16	10	34	1134
prils_dp	31	23	20	27	983
rsadd_dp	38	16	13	34	740
smu_dpath	35	17	16	33	1211
ucode_dat	39	12	11	38	3304
ucode_reg	47	14	4	36	191
Total	36	16	14	34	38 846

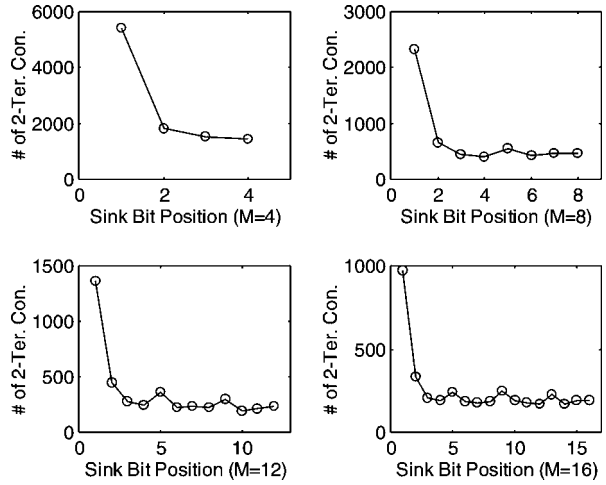

Fig. 5 *Number of inputs that outputs at bit position 1 are connected to*

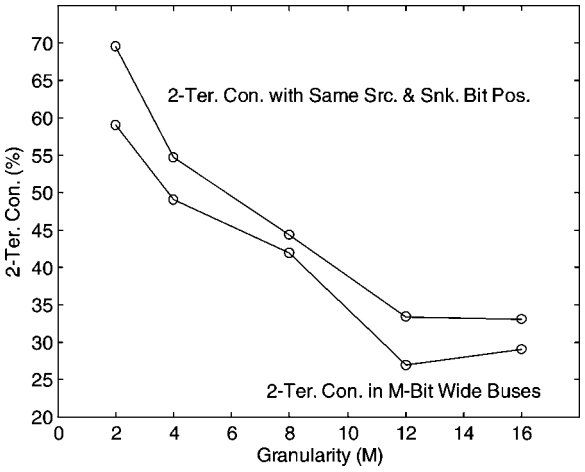

Fig. 7 *Two-terminal connections in M-bit wide buses*

requirements of the conventional FPGA logic blocks, where architects strive to uniformly distribute the connections of each logic block output signal to all available logic block input signals [13].

For two-terminal connections that have the same source and sink bit positions, some of these signals can be further grouped into M-bit wide buses, as shown in Fig. 6, where each signal in a bus has a distinct source/sink bit position, and all the signals in the bus originate from a common multi-bit block and terminate at another. The number of signals that exist in these buses is shown in Fig. 7 for the benchmark circuits over a range of granularity values (M). Here, there are two lines in the figure, where the top line indicates the percentage of two-terminal connections that have the same source and sink bit positions, and the bottom line shows the percentage of two-terminal connections that can be grouped into M-bit wide buses. As shown, the number of signals that can be grouped into M-bit wide buses consists of a significant proportion (from 25 to 60%) of the total number of connections. As all signals in each bus share a single source and a single sink block, these signals can be a potential source of redundant information in FPGA routing configuration. This redundancy, in turn, can be exploited by FPGA architects to improve the area efficiency of FPGAs, where a single configuration memory bit can be used in place of multiple bits of memory to store the identical configurations.

Having observed the correlation, one question that naturally arises is how exactly this correlation can be turned into area savings. One of the best ways to address the question is through a set of empirical studies, where an FPGA architecture with varying switch patterns is first defined and then used to implement a set of benchmark

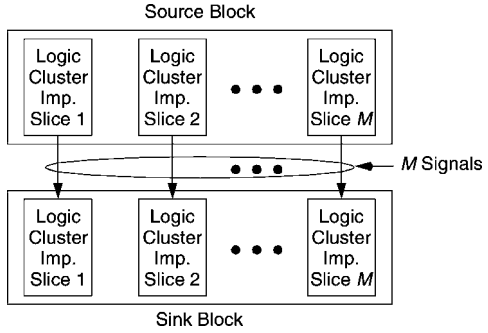

Fig. 6 *Grouping signals into M-bit wide buses*

circuits. Such an approach is used in this study. The architecture used in the study is discussed in what follows.

3 Multi-bit routing architecture

Each FPGA building block is connected to its neighbouring routing channels through a set of programmable routing switches. The distribution of these switches (called switch patterns) can be partially characterised by five variables, consisting of N, I, W, $F_{c\,out}$ and $F_{c\,in}$. Here N is the number of output pins that a block has, I is the number of input pins of the block and W is the number of routing tracks in a routing channel. $F_{c\,out}$ and $F_{c\,in}$, respectively, are the percentage of these tracks that an output pin and an input pin connect to. For any given set of quintuplet, (N, I, W, $F_{c\,out}$, $F_{c\,in}$), the total number of all possible switch pattern variations can be as large as

$$D = \begin{pmatrix} W \\ \lfloor F_{c\,out} \times W \rfloor \end{pmatrix}^{N} \times \begin{pmatrix} W \\ \lfloor F_{c\,in} \times W \rfloor \end{pmatrix}^{I} \quad (1)$$

and this design space can be somewhat reduced to

$$D_E = \left(\begin{pmatrix} W \\ \lfloor F_{c\,out} \times W \rfloor \end{pmatrix} \right)_{N} \times \left(\begin{pmatrix} W \\ \lfloor F_{c\,in} \times W \rfloor \end{pmatrix} \right)_{I} \quad (2)$$

if all input pins of a block are functionally identical (called a set of logically equivalent [13] input pins) and all output pins of the block are also functionally identical (called a set of logically equivalent output pins).

D_E, however, can still be a quite large number for the typical values of N, I, W, $F_{c\,out}$ and $F_{c\,in}$ found in modern FPGAs. For example, for $N = 4$, $I = 10$, $W = 20$, $F_{c\,out} = 0.25$ and $F_{c\,in} = 0.5$, which are the typical values for connecting a conventional FPGA logic block to its neighbouring routing tracks, D and D_E are equal to 2.7×10^{69} and 3.1×10^{61}, respectively. For multi-bit building blocks, which typically contain many more input and output pins, the design space is even larger.

This enormous design space has never been extensively exploited before, especially with the aid of the connectivity information such as those described in Section 2. Most automated FPGA architecture generation tools usually are designed for FPGAs containing conventional logic blocks, and usually assume that each of these blocks contains a set of logically equivalent input/output pins. Typically, these tools only explore a very limited number of switch

patterns in the entire architectural design space. For example, one of the most popularly used tools, as described in the work of Betz *et al.* [13], allows the user to generate various switch patterns through the variation of the five architectural parameters, N, I, W, $F_{c\,out}$ and $F_{c\,in}$, as described earlier. For any given set of parameter values, however, the tool only selects one design point in the entire design space on the basis of following five design criteria as outlined in Betz *et al.* [13].

1. 'Ensure that each of the W tracks in a channel can be connected to roughly the same number of input pins, and roughly the same number of output pins'.
2. 'Ensure that each pin can connect to a mix of different wire types ...'.
3. 'Ensure that pins that appear on multiple sides of the logic block connect to different tracks on each side ...'.
4. 'Ensure that logically equivalent pins connect to different tracks ...'.
5. 'Ensure that pathological switch topologies in which it is impossible to route from certain output pins to certain input pins do not occur'.

Note that, in combination, criteria 1, 2, 4 and 5 attempt to distribute, as uniformly as possible, the output connections to all available inputs. This uniform-nature of the distribution is more precisely defined by the actual software implementation of the above criteria [31] as shown in Fig. 8.

In the figure, P and F_c are assigned to be I and $F_{c\,in}$, respectively, for generating switch patterns for input pins. For generating switch patterns for output pins, in contrast, they are assigned to be N and $F_{c\,out}$, respectively. The resulting matrix, switch_pattern, contains the position of each routing switch. Here each entry of the matrix, (i, j), contains the index of a routing track (numbered from 0 to $W-1$) that should be connected to the ith input/output pin (where the input pins are numbered from 0 to $I-1$ and the output pins are numbered from 0 to $N-1$) through a programmable routing switch. Note that for any given value of i, the switches that connect to the ith input/output pin are indexed by j in each column of the matrix.

An example of the switch patterns generated by the above algorithm is shown in Fig. 9. In the figure, a programmable routing switch is denoted by an X. For clarity, all input and output connections are connected to one routing channel instead of being evenly distributed to the four routing channels surrounding the building block as assumed by criterion 3.

The uniform distribution-nature of the switch patterns shown by the example strives to equalise the number of

```
step = (W/P)/⌊F_c × W⌋;

increment = W/⌊F_c × W⌋;

for (i = 0; i < P; i++) {

    for (j = 0; j < ⌊F_c × W⌋; j++) {

        switch_pattern[i][j] = step × i + increment × j;

    }

}
```

Fig. 8 *Algorithm for uniform switch pattern generation [31]*

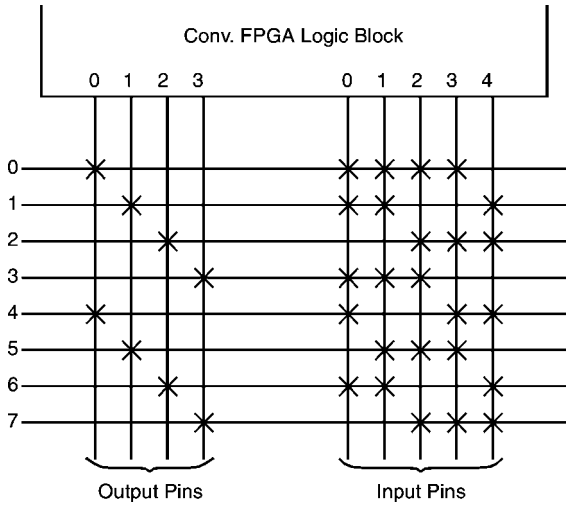

Fig. 9 *Switch patterns for $W = 8$, $N = 4$, $I = 5$, $F_{c\,out} = 0.25$ and $F_{c\,in} = 0.7$*

routing paths from any output pin to any of the input pins. In particular, in the figure, the routing paths are evenly distributed such that there is at least one routing path connecting one output pin to each of the five input pins. For example, output pin 0 can be connected to input pins 0, 1, 2 and 3 through track 0 and pin 4 through track 4; output pin 1 can be connected to input pins 0, 1 and 4 through track 1 and input pins 2 and 3 through track 5; similar connections exist for output pins 2 and 3, respectively.

Additionally, there are five extra routing paths and the distribution of these routing paths ensures that each input/output pin receives at least one extra path. In particular, output pins 1, 2 and 3, each have two routing paths for connecting to pin 1, 4 and 2, respectively, whereas output pin 0 has one extra routing path for connecting to pin 0 and another for connecting to pin 3.

Mathematically, this uniform connection pattern can be made more apparent by an $N \times I$ matrix, which will be called the routing path distribution matrix in this paper, where each entry (i, j) indicates the total number of routing paths available for connecting output pin i (numbered from 0 to $N-1$) to input pin j (numbered from 0 to $I-1$). The distribution matrix for the switch patterns shown in Fig. 9 is listed below. Note that all entries of the matrix are either n (where $n = 1$ in this case) or $n+1$ and all $n+1$ values are carefully staggered to ensure the even distribution of all the extra routing paths

$$\boldsymbol{R}_{\text{path}} = \begin{bmatrix} 2 & 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 & 1 \end{bmatrix} \tag{3}$$

Fig. 10 shows a denser switch pattern for the input pins. This pattern nearly guarantees two routing paths for almost all of the 20 pairs of output to input pin connections with only five exceptions. These exceptions include the connections from output pin 0 to input pin 2, output pin 1 to input pin 4, output pin 2 to input pin 1, output pin 3 to input pin 0 and output pin 3 to input pin 3, where only one routing path exists in each case. Again these single paths are carefully distributed to ensure that each input/output pin is connected to at least one of these paths as shown by the distribution

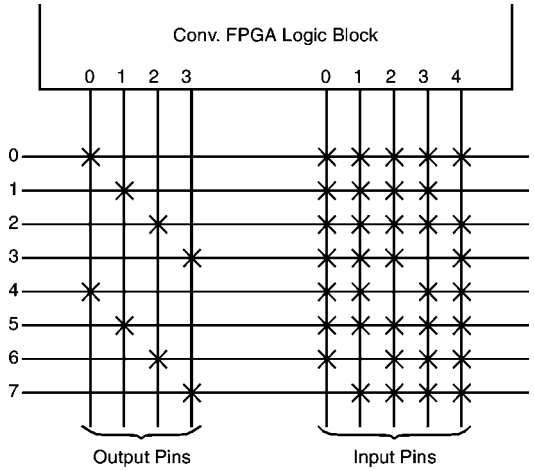

Fig. 10 *Switch patterns for $W = 8$, $N = 4$, $I = 5$, $F_{c\ out} = 0.25$ and $F_{c\ in} = 0.7$*

matrix below

$$\boldsymbol{R}_{\text{path}} = \begin{bmatrix} 2 & 2 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 1 \\ 2 & 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 1 & 2 \end{bmatrix} \quad (4)$$

(Note that occasionally, the algorithm shown in Fig. 8 will create a set of switch patterns that violate criterion 5, where an output pin can only be connected to a limited number of input pins. One example of such an occurrence appears when both the step and the increment variables for generating the output switch pattern are even multiples of their counterparts for generating the input switch pattern. In these cases, one usually, as the work of Betz *et al.* [31] does, adjusts the values of $F_{\text{c in}}$ and $F_{\text{c out}}$ according to a set of pre-prescribed rules such that the new switch patterns would conform to all the design criteria.)

The uniformly distributed switch patterns described earlier, however, cannot take advantage of the non-uniform nature of the correlation statistics observed in Section 2. Turning our observation in signal connectivity into actual area savings would require the design of another concrete routing architecture. Such an architecture (called the multi-bit routing architecture) is used in this work. As shown in Fig. 11, the architecture contains two types of routing tracks, the single-bit tracks and the multi-bit tracks. Each track is connected to several multi-bit block input and output signals through a set of routing switches (denoted by an X in the figure).

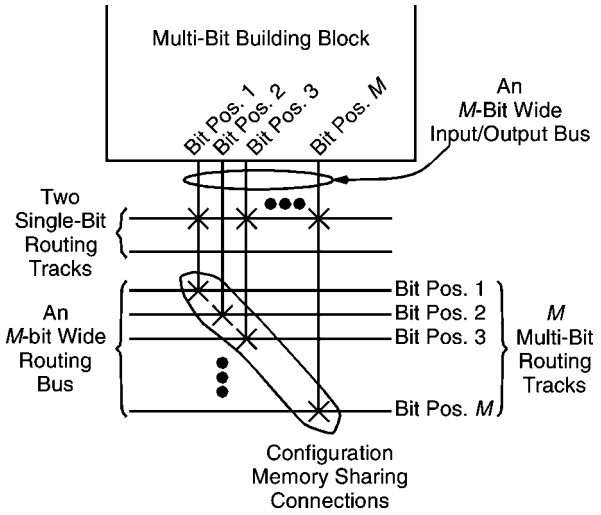

Fig. 11 *Multi-bit routing*

The single-bit tracks are similar, in structure, to the conventional FPGA routing tracks. Their switch patterns are designed to uniformly distribute multi-bit block output connections to all available multi-bit block inputs [13]. The multi-bit tracks, in contrast, are organised into M-bit wide buses. In a bus, each track is assigned a unique bit position number, ranging from 1 to M. These numbers are then used to match the bit positions of the signals that the tracks are connected to. For example, a track at bit position 1 can only be connected to multi-bit block input and output signals that are also from bit position 1. In general, a track at bit position x can only be connected to input and output signals also from bit position x. Consequently, increasing the number of multi-bit tracks in the architecture only increases the availability of one type of two-terminal connections, namely two-terminal connections with the same source and sink bit positions. Increasing the number of single-bit tracks, on the contrary, uniformly increases the availability of all types of two-terminal connections. (Note that within each set of multi-bit routing tracks that have the same bit position number, the routing switches are evenly distributed across these tracks on the basis of an algorithm that is similar to the one outlined in Fig. 8.)

Furthermore, the routing switches in each routing bus (which consists of a group of M multi-bit routing tracks) are further grouped into M-bit wide groups in order to exploit the observed redundant configuration information. Each group, as shown in Fig. 11, shares a single set of configuration memory. The sharing of configuration memory, however, complicates the design of the routing tools. In the next section, a new NC-based routing algorithm is introduced. The algorithm accommodates configuration memory sharing. Both the architecture and the algorithm are then used to empirically measure the overall effect of this correlation between signal connectivity and signal positioning on the area efficiency of FPGAs.

4 Routing on the configuration-memory sharing routing resources

As an NC-based routing algorithm, routing is performed in multiple routing iterations. Each iteration is controlled by a set of cost functions that measure the delay and the congestion of each route. These cost functions consist of a collection of cost metrics. Each metric is updated at the end of each routing iteration on the basis of the current and the historical routing results.

During a routing iteration, two-terminal connections that can be grouped into M-bit wide buses (as defined in Section 2) are first routed. In particular, all signals in the bus are routed through the multi-bit routing tracks (which are connected by the configuration-memory sharing switches) as a single group. Then a bit chosen at random from the bus is routed through the single-bit tracks as an individual signal. These two routing solutions, one consists of a series of multi-bit routing tracks and the other consists of a series of single-bit routing tracks, are then compared on the basis of their congestion and delay metrics. If the single-bit tracks provide a better solution (because of much lower delay or congestion), the multi-bit track solution is abandoned in favour of routing all the signals in the bus through the single-bit tracks. Otherwise, the multi-bit track solution is used to route the bus. Note that the main purpose of routing a single bit of each bus through the

single-bit routing tracks is to test the congestion of these tracks against the congestion of the multi-bit routing tracks. In the case that the multi-bit tracks are heavily congested, whereas the single-bit tracks are not, the routing of the test bit will incur a significantly lower cost (based the cost functions described subsequently). Consequently, all signals in the bus will be routed through the less congested single-bit tracks to avoid the congestion of the multi-bit routing tracks.

For two-terminal connections that cannot be grouped into buses, they are routed as individual signals. Here each signal is routed through both the single-bit and the multi-bit tracks in order to obtain the best routing solutions (which means routing solutions with the best possible combination of delay and congestion).

4.1 Expansion topologies

In conventional FPGAs, each architecture can be represented by a graph called the routing resource graph, where nodes represent the routing resources, and the edges represent the routing switches. To route a signal in this graph, one uses a series of wave-front expansions, where the initial wave front consists of only the source node of the signal. At each stage of the expansion, the node with the lowest possible cost is selected from the front. The selected node is then substituted by its neighbouring nodes to expand the current front. The expansion continues until all sinks of the signal are reached.

The multi-bit routing architecture can also be represented by a routing resource graph. Here each node in the graph represents either a multi-bit block input pin, a multi-bit block output pin, a continuous wire segment between two routing switches on a single-bit routing track or a similar wire segment on a multi-bit routing track. Each edge of the graph represents a routing switch that connects these nodes together. Similarly signals can also be routed through the graph using the technique of wave-front expansion. In particular, when routing an individual signal through the single-bit routing tracks, the expansion appears exactly the same as the topology of the conventional wave-front expansions. An example of such an expansion is shown in Fig. 12*a*, where the wave front is expanded from a single node denoted by n', to five neighbouring nodes (one of which is denoted by n in the figure).

Routing either a bus or an individual signal through a routing bus (which consists of M multi-bit routing tracks), however, would require the simultaneous expansion of M wave fronts, where one wave front is used to keep track of a bit in the bus. The multiple fronts are needed in order to fully account for the expansion costs because groups of M switches in each routing bus are collectively controlled by a single set of configuration memory. Consequently, no individual connections can be made in isolation. Instead, wire segments must be connected together in M-bit wide groups.

An example of such an expansion topology is shown in Fig. 12*b*, where a set of wave fronts, each containing one node, denoted by n'_1 through n'_M in the figure, is expanded into two sets of nodes, where one of the sets is denoted by n_1 through n_M. This added expansion topology requires the design of a set of expansion cost functions so the single-bit expansions (such as the one shown in Fig. 12*a*) can be fairly compared with the multi-bit expansions (such as the one shown in Fig. 12*b*).

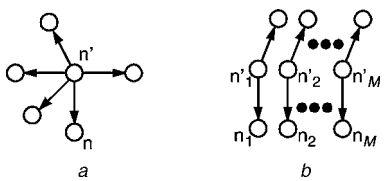

Fig. 12 *Expansion topologies*
a Single-bit expansions
b Multi-bit expansions

4.2 Expansion costs

As the single-bit expansion topology is identical to the conventional expansion topology, the conventional cost function as defined [13] is used in this study for these expansions. In particular, the following equation is used

$$\begin{aligned}\text{expansion\_cost}(n) &= [1 - \text{criticality}] \times C(n) \\ &\quad + \text{criticality} \times D(n) \\ &\quad + \text{future\_expansion\_cost}(n)\end{aligned} \tag{5}$$

Note that a detailed description of (5) is presented in the work of Betz [13]. Here, we briefly review the major features of the equation for completeness. In the equation, $C(n)$ is defined to be the accumulated congestion cost of all nodes in an expansion path that connects the source of the signal all the way to node n. It is calculated on the basis of following formula

$$C(n) = \text{congestion\_cost}(n) + C(n') \tag{6}$$

which states that the accumulated congestion cost of any node, n, is equal to the sum of the accumulated congestion cost of n', the node immediately preceding n on the expansion path, and the congestion cost of n. This congestion cost is a function of the capacity (which is equal to the maximum number of times that a node can be legally used) of node n and the number of times that n is actually being used [13].

Also, in (5), $D(n)$ is defined to be the delay cost of the expansion path that connects the source of the signal to node n. Both $C(n)$ and $D(n)$ are scaled by the criticality of the signal (which is a fraction between 0 and 1). A high criticality value means that when compared with other signals, the signal that is being routed has a higher delay value; therefore a larger proportion of the expansion cost should be equal to the delay cost. Otherwise, the signal has a lower delay value and the accumulated congestion cost should be the larger proportion of the expansion cost. Finally, the final term in the equation represents the future expansion cost, which is an estimation on the additional delay and accumulated congestion cost that can incur in the path that connects node n to the sink of the signal.

When routing a bus of two-terminal connections through the multi-bit expansion topology, one has to deal with the delay and accumulated congestion of M signals and M nodes instead of one. In this case, (5) is used to calculate the expansion cost for each signal and its corresponding node. The maximum of these costs is then used as the overall cost of the expansion. More formally, the expansion cost in this case is defined to be

$$\begin{aligned}&\text{expansion\_cost}'(n_1, n_2, \ldots, n_M) \\ &= \max(\text{expansion\_cost}(n_1), \text{expansion\_cost}(n_2), \ldots, \\ &\quad \text{expansion\_cost}(n_M))\end{aligned} \tag{7}$$

When routing a single bit of a signal through the same expansion topology, however, there will be only one delay cost. This cost reflects the delay of the signal that is

being routed. There are, however, M different accumulated congestion costs, one for each node of the expansion. As a result, (7) no longer applies. To accommodate, the accumulated congestion cost is redefined to be the maximum of all accumulated congestion costs as follows

$$C'(n_1, n_2, \ldots, n_M) = \max(\text{congestion\_cost}(n_1), \text{congestion\_cost}(n_2), \ldots, \text{congestion\_cost}(n_M)) + C'(n'_1, n'_2, \ldots, n'_M) \quad (8)$$

This accumulated congestion cost is then used in place of $C(n)$ in (5) to calculate the overall expansion cost of the topology.

5 Results

The router is then used with a set of datapath-oriented CAD tools to map the 15 benchmark circuits (presented in Section 2) onto several variants of the multi-bit routing architecture presented in Section 3. These variants primarily differ in the composition of their routing tracks where each architecture contains a different amount of single-bit and multi-bit tracks. From these architectures, the architecture with the best area is identified for each circuit. It is then compared with a base architecture that contains only single-bit routing tracks (tracks that strictly distribute the connections of each multi-bit block output uniformly across all available multi-bit block inputs) for the number of routing switches, the amount of configuration memory (SRAM) and the area usage of routing resources.

For the study, it is assumed that each circuit is placed onto a square FPGA that contains just enough multi-bit building blocks to accommodate the given circuit. It is also assumed that each block contains four clusters, which is one of the most widely used granularity values for several existing datapath-oriented FPGA architectures [4, 9]. As discussed in Section 2, it is assumed that each cluster contains four four-input LUTs and four DFFs. The number of input signals per multi-bit block is assumed to be 40 ($I = 10$). The number of output signals per multi-bit block is assumed to be 16 ($N = 4$). These signals are grouped into ten 4-bit wide input buses and four 4-bit wide output buses, respectively. The routing switches that connect the wire segments together are assumed to have a disjoint switch block topology [32] (for both the single-bit and the multi-bit tracks). Using data from conventional FPGAs [13], it is further assumed that each multi-bit block input/output pin is connected to 40/25% of the single-bit tracks in a routing channel, and each input bus/output bus is connected to 40/25% of the 4-bit wide routing buses. Finally, each wire segment is assumed to continuously expand two multi-bit building blocks before being interrupted by a switch-block routing switch for both the single-bit and the multi-bit tracks. Again, length-two wire segments have been shown to have good performance for conventional FPGAs [13]. Note that all transistors are properly sized in each architecture to ensure good performance and efficient area usage.

Table 2 shows the number of single-bit tracks and the number of multi-bit tracks per channel, which are required to implement each benchmark circuit in columns 2 and 3, respectively. It also shows the number of tracks that is needed to implement the same circuit, using purely single-bit tracks in column 4. (Shown in parentheses in column 2 is the value of column 2 as a percentage of the value presented in column 4.) As shown, the multi-bit tracks can be used to significantly reduce the number of single-bit tracks in an architecture. For a vast majority (ten) of the circuits shown in Table 2, the total number of single-bit tracks in each is reduced by over 30% when compared with the full single-bit track implementations. The average number of single-bit tracks per channel is reduced by over 37%.

Table 2: Track count per channel

Circuit	Single-bit + multi-bit		Single-bit only
	Number of single-bit tracks, n (%)	Number of multi-bit tracks	
code_seq_dp	37 (90)	8	41
dcu_dpath	27 (48)	36	56
ex_dpath	39 (45)	52	86
exponent_dp	41 (63)	36	65
icu_dpath	39 (45)	60	86
imdr_dpath	47 (66)	32	71
incmod	37 (69)	32	54
mantissa_dp	42 (55)	52	77
multmod_dp	57 (92)	8	62
pipe_dpath	32 (94)	8	34
prils_dp	33 (85)	20	39
rsadd_dp	22 (59)	32	37
smu_dpath	34 (79)	16	43
ucode_dat	29 (49)	44	59
ucode_reg	13 (46)	36	28
Average	35 (63)	31	56

The direct benefit of reduction in single-bit tracks is in the reduction of routing switches that connect the multi-bit blocks to their routing tracks. Table 3 shows the reduction in columns 2–4. Here column 2 lists the total number of switches (both for connecting the multi-bit block input and output signals) required for architectures containing both the single-bit and the multi-bit tracks. Column 3 lists the same number for architectures containing only single-bit tracks. The percentage reduction is then listed in column 4. As shown, one can achieve an overall routing switch reduction of over 27% through the use of multi-bit routing tracks.

Reducing the number of single-bit tracks also reduces the number of SRAM bits required to configure the routing resources because of configuration memory sharing. Columns 5–7 of Table 3 show the SRAM reduction figures. As shown, architectures containing both the single-bit and the multi-bit tracks consume about 18% fewer SRAM bits than architectures containing only single-bit routing tracks.

The area savings that can be achieved through the reduction of routing switches and SRAM bits, however, are offset by the larger switch block size in architectures that use multi-bit routing tracks. This increase in switch block size is primarily due to the increase in the total number of routing tracks. As shown in Table 2, the average track count per channel, including both the single-bit tracks and the multi-bit tracks, actually increases from the 56 tracks of the full single-bit track implementations to the 66 tracks of the combined single-bit and multi-bit implementations. This increase in track count also increases the number of routing switches required in each switch block (which should be differentiated from the routing switches that connect multi-bit blocks to the routing tracks), and consequently reduces the overall area savings.

Taking into consideration the increase in switch block size, Table 4 summarises the actual area savings/increases

Table 3: Switch count and SRAM bit count

Circuit	Switches in routing			SRAM bits in routing		
	Single-bit + multi-bit	Single-bit only	% Reduction	Single-bit + multi-bit	Single-bit only	% Reduction
code_seq_dp	18 768	19 688	4.7	20 700	21 206	2.4
dcu_dpath	47 880	72 072	34	53 739	68 922	22
ex_dpath	194 304	308 352	37	178 464	245 872	27
exponent_dp	34 048	41 984	19	32 128	37 312	14
icu_dpath	239 568	380 184	37	222 425	303 147	27
imdr_dpath	92 664	117 288	21	95 013	99 225	4.2
incmod	54 264	62 928	14	55 062	61 845	11
mantissa_dp	74 240	99 840	26	66 752	81 152	18
multmod_dp	127 680	131 880	3.2	116 550	121 065	3.7
pipe-dpath	20 416	20 416	0.0	24 969	25 375	1.6
prils_dp	21 216	20 800	−2.0	22 802	23 738	3.9
rsadd_dp	13 608	15 960	15	17 031	18 648	8.7
smu_dpath	28 800	32 256	11	32 184	35 532	9.4
ucode_dat	71 048	99 600	29	72 459	93 209	22
ucode_reg	3072	3552	14	4356	4896	11
Total	1 041 576	1 426 800	27	1 014 634	1 241 144	18

for each benchmark circuit. Here the area is measured using the equivalent minimum-width transistor area as described in the work of Betz *et al.* [13], where the total area required to implement an FPGA is normalised against the area that is required to implement a minimum-width transistor. As shown, ten of the 15 circuits require less area to implement when using multi-bit routing tracks. These circuits consist of 82% of the total area of all benchmark circuits. Finally, the overall area reduction for these benchmark circuits is over 12%. In particular, the two largest circuits (ex_dpath and icu_dpath) did particularly well, and achieve an area reduction of 22 and 18% each.

Table 5 shows that the use of multi-bit routing tracks has little impact on the overall delay of a circuit. In particular, column 2 of the table shows the critical path delay for architectures containing both single-bit and multi-bit routing tracks. Column 3 shows the delay for architectures containing only single-bit routing tracks. Finally, the change in critical path delay is listed in column 4. Overall, there is a slight reduction of 0.8% in critical path delay for all benchmark circuits. In the worst case, one circuit (multmod_dp) has a 32% increase in critical path delay; but another circuit (dcu_dpath), however, has a 32% reduction in critical path delay. (Note that both variations are caused by the large number of near-critical path signals that these circuits contain).

Finally, Table 6 shows the area results obtained by fixing the total number of the multi-bit routing tracks to 32 (one track more than the average number of multi-bit tracks shown in Table 2) and adding just enough single-bit routing tracks to make the circuits routable. In the table, column 1 shows the circuit names; column 2 shows the

Table 4: Routing area

Circuit	Routing area (10×10^5)		
	Single-bit + multi-bit	Single-bit only	% Reduction
code_seq_dp	2.69	2.66	−1.1
dcu_dpath	7.24	8.81	18
ex_dpath	26.9	34.6	22
exponent_dp	5.04	5.40	6.7
icu_dpath	33.5	40.7	18
imdr_dpath	11.7	12.9	9.3
incmod	7.23	7.27	0.55
mantissa_dp	10.5	11.6	9.5
multmod_dp	14.8	14.7	−0.68
pipe_dpath	2.82	2.71	−4.1
prils_dp	2.80	2.54	−10
rsadd_dp	2.14	2.09	−2.4
smu_dpath	4.02	4.16	3.4
ucode_dat	10.2	11.6	12
ucode_reg	0.536	0.565	5.1
Total	142	162	12

Table 5: Routing delay

Circuit	Routing delay (ns)		
	Single-bit + multi-bit	Single-bit only	Change, %
code_seq_dp	6.47	6.51	−0.61
dcu_dpath	6.67	9.83	−32
ex_dpath	20.1	20.1	0.0
exponent_dp	8.96	9.15	−2.1
icu_dpath	12.0	11.6	+3.5
imdr_dpath	19.1	18.2	+4.9
incmod	19.1	19.6	−2.6
mantissa_dp	8.14	7.86	+3.6
multmod_dp	15.6	11.8	+32
pipe_dpath	6.65	7.01	−5.1
prils_dp	15.1	13.9	+8.6
rsadd_dp	13.3	12.8	+3.9
smu_dpath	9.92	10.5	−5.5
ucode_dat	7.80	8.86	−12
ucode_reg	1.99	1.96	+1.5
Geo. average	9.92	10.0	−0.8

Table 6: Area results for architectures containing 32 multi-bit routing tracks

Circuit	Tracks (single-bit + multi-bit) per channel	Routing area (10×10^5) Area	% Reduction	Total area (10×10^5) Area	% Reduction
code_seq_dp	63	2.96	−11	4.51	−6.9
dcu_dpath	65	7.84	11	12.1	7.6
ex_dpath	97	32.5	6.1	44.4	4.5
exponent_dp	77	5.16	4.4	7.52	3.1
icu_dpath	89	35.1	14	49.8	10
imdr_dpath	79	11.7	9.3	17.2	6.0
incmod	69	7.23	0.55	11.1	0.0
mantissa_dp	90	11.3	2.6	15.6	2.5
multmod_dp	85	16.3	−11	23.3	−6.9
pipe-dpath	53	2.89	−6.6	5.05	−3.7
prils_dp	61	2.91	−15	4.67	−8.9
rsadd_dp	54	2.14	−2.4	3.56	−1.4
smu_dpath	61	4.26	−2.4	6.69	−1.4
ucode_dat	66	10.2	12	15.8	8.1
ucode_reg	53	0.698	−24	1.10	−13
Total/average	71	153	5.6	222	4.0

total number of tracks per channel. The routing area results are shown in columns 3 and 4, respectively. The total area results (logic area + routing area) are shown in columns 5 and 6. As shown, in this case, the overall routing area reduction is 5.6%. The overall total area savings is 4.0%.

6 Conclusions

This paper examines the correlation between signal connectivity and signal positioning for multi-bit building blocks in FPGAs. It is shown that, for their bussed structures, multi-bit block input and output signals with the same bit positions are much more likely to connect together. On the basis of the observation, a routing architecture is designed to use dedicated routing tracks to enrich the connectivity between these identical bit positions. Configuration-memory sharing is then used to reduce the amount of memory required to configure these tracks.

Using the proposed architecture and a specialised routing algorithm, it is empirically shown that the correlation can be directly translated into a 27% reduction in the number of routing switches, which connect the multi-bit blocks to their tracks. Furthermore, 18% fewer configuration-memory bits are needed to control the routing resources. Overall, one can achieve a routing area saving of between 6 and 12%.

7 Future work

This work has quantified the routing demand of datapath circuits implemented using multi-bit building blocks. Datapath circuits, however, often are controlled by a set of signals externally generated by the control logics of an application, which typically are not datapath in nature. In most cases, the sizes of control logics are much smaller than the sizes of the datapaths. Consequently, control circuits usually have much less impact on the overall routing demand of applications. It is still, however, useful to quantify the routing characteristics of these circuits relative to the routing demand of the actual datapaths. This work will be carried out in future.

In future, we will also examine the routing characteristics of entire applications, which typically consist of control logics and several interconnected datapath circuits. These applications will be implemented using a variety of multi-bit building blocks in conjunction with the conventional FPGA logic blocks to simulate the actual implementation platforms provided by the current commercial FPGAs. The detailed architectural questions that can be addressed through these more extensive studies include the effect of larger logic cluster sizes, more realistic channel widths (at around 100 tracks per channel) and the effect of RAM blocks with configurable datapath widths on the overall area of efficiency of multi-bit routing tracks.

8 Acknowledgment

The majority of this work was carried out at the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto.

9 References

1 Chen, D., and Rabaey, J.: 'A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths', *IEEE J. Solid-State Circuits*, 1992, **27**, (12), pp. 1895–1904
2 Yeung, A., and Rabaey, J.: 'A reconfigurable data driven multi-processor architecture for rapid prototyping of high throughput DSP algorithms'. Proc. Hawaii Int. Conf. on System Science, January 1993, pp. 169–178
3 Bittner, R., and Athanas, P.: 'Wormhole run-time reconfiguration'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 1997, pp. 79–85
4 Cherepacha, D., and Lewis, D.: 'DP-FPGA: An FPGA architecture optimized for datapaths', *VLSI Des.*, 1996, **4**, (4), pp. 329–343
5 Ebeling, C., Cronquist, D., Franklin, P., and Fisher, C.: 'RaPiD – Reconfigurable pipelined datapath'. Proc. Int. Workshop on Field-Programmable Logic and Applications, August 1996, pp. 237–241
6 Hauser, J., and Wawrzynek, J.: 'Garp: a MIPS processor with a reconfigurable coprocessor'. Proc. IEEE Symp. on Field-Programmable Custom Computing Machines, April 1997, pp. 24–33
7 Waingold, E., *et al.*: 'Baring it all to software: raw machines', *Computer*, 1997, **30**, (9), pp. 86–93
8 Miyamori, T., and Olukotun, K.: 'A quantitative analysis of reconfigurable coprocessors for multimedia applications'. Proc. IEEE Symp. on Field-Programmable Custom Computing Machines, April 1998, pp. 2–11

9 Marshall, A., Stansfield, T., Kostarnov, I., Vuillemin, J., and Hutchings, B.: 'A reconfigurable arithmetic array for multimedia applications'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 1999, pp. 135–143
10 Alsolaim, A., Starzyk, J., Becker, J., and Glesner, M.: 'Architecture, and application of a dynamically reconfigurable hardware array for future mobile communication systems'. Proc. IEEE Symp. on Field-Programmable Custom Computing Machines, April 2000, pp. 205–214
11 Goldstein, S., Schmidt, H., Budiu, S., Cadambi, S., Moe, M., and Taylor, R.: 'PipeRench: a reconfigurable architecture, and compiler', *Computer*, 2000, **33**, (4), pp. 70–77
12 Leijten-Nowak, K., and van Meerbergen, J.: 'An FPGA architecture with enhanced datapath functionality'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 2003, pp. 195–204
13 Betz, V., Rose, J., and Marquardt, A.: 'Architecture, and CAD for deep-submicron FPGAs' (Kluwer Academic Publishers., 1999)
14 Altera Corporation, 'Altera product literature', http://www.altera.com
15 Lewis, D., *et al.*: 'The Stratix-II logic, and routing architecture'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 2005, pp. 12–20
16 Xilinx Inc.: 'Xilinx technical documentation', http://www.xilinx.com
17 Ye, A., Rose, J., and Lewis, D.: 'Synthesizing datapath circuits for FPGAs with emphasis on area minimization'. Proc. Int. Conf. on Field-Programmable Technology, December 2002, pp. 219–227
18 Ye, A., and Rose, J.: 'Using multi-bit logic blocks and automated packing to improve field-programmable gate array density for implementing datapath circuits'. Proc. Int. Conf. on Field-Programmable Technology, December 2004, pp. 129–136
19 Ye, A.: 'Field-programmable gate array architectures and algorithms optimized for implementing datapath circuits'. Ph.D. Thesis, University of Toronto, 2004, http://www.eecg.toronto.edu/~jayar/pubs/theses/Ye/AndyYe.pdf
20 Lee, C.: 'An algorithm for path connections and its applications', *IRE Trans. Electron. Comput.*, 1961, **10**, pp. 346–365
21 Ebeling, C., McMurchie, L., Hauck, S., and Burns, S.: 'Placement and routing tools for the triptych FPGA', *IEEE Trans Very Large Scale Integr. (VLSI) Syst.*, 1995, **3**, (4), pp. 473–482
22 Swartz, J., Betz, V., and Rose, J.: 'A fast routability-driven router for FPGAs'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 1998, pp. 140–149
23 Chan, P., and Schlag, M.: 'New parallelization and convergence results for NC: a negotiation-based FPGA router'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 2000, pp. 165–174
24 Ye, A., Rose, J., and Lewis, D.: 'Architecture of datapath-oriented coarse-grain logic and routing for FPGAs'. Proc. IEEE Custom Integrated Circuits Conf., September 2003, pp. 61–64
25 Ye, A., and Rose, J.: 'Using bus-based connections to improve field-programmable gate array density for implementing datapath circuits'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 2005, pp. 3–13
26 Lemieux, G., Leventis, P., and Lewis, D.: 'Generating highly-routable sparse crossbars for PLDs'. Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February 2000, pp. 155–164
27 Ye, A., and Rose, J.: 'Measuring and utilizing the correlation between signal connectivity and signal positioning for FPGAs containing multi-bit building blocks'. Proc. Int. Workshop on Field-Programmable Logic and Applications, August 2005, pp. 159–166
28 Betz, V., and Rose, J.: 'How much logic should go in an FPGA logic block?', *IEEE Des. Test Comput.*, 1998, **15**, (1), pp. 10–15
29 Ahmed, E., and Rose, J.: 'The effect of LUT and cluster size on deep-submicron FPGA performance and density', *IEEE Trans Very Large Scale Integr. (VLSI) Syst.*, 2004, **12**, (3), pp. 288–298
30 Sun Microsystems: 'Pico-Java processor design documentation', 1999
31 Betz, V., Rose, J., and Marquardt, A.: 'VPR: A placement and routing tool for FPGA research', http://www.eecg.utoronto.ca/~vaughn/vpr/vpr.html
32 Hseih, H., *et al.*: 'Third-generation architecture boosts speed and density of field-programmable gate arrays'. Proc. IEEE Custom Integrated Circuits Conf., March 1990, pp. 31.2.1–31.2.7