FPGA Switch Block Layout and Evaluation

Herman Schmit
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213
herman@ece.cmu.edu

Vikas Chandra
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213
vchandra@ece.cmu.edu

ABSTRACT

This paper presents abstract layout techniques for a variety of FPGA switch block architectures. We evaluate the relative density of subset, universal, and Wilton switch block architectures. For subset switch blocks of small size, we find the optimal implementations using a simple metric. We also develop a tractable heuristic that returns the optimal results for small switch blocks, and good results for large switch blocks. For switch blocks with general connectivity, we develop a representation and a layout evaluation technique. We use these techniques to compare a variety of small switch blocks. We find that the traditional Xilinx-style, subset switch block is superior to the other proposed architectures. Finally, we have hand-designed some small switch blocks to confirm our results.

Keywords

FPGA Interconnect, VLSI Layout

1. INTRODUCTION

In this paper, we explore the question of how to physically implement FPGA switch blocks. We evaluate numerous switch block connectivities described in the literature. This issue is important for a number of reasons. First, the determination of optimal FPGA architectures requires an evaluation of both utilization and area. For example, in order to compare two architectures it is important to compare the routability of the architecture, as well as the size of the implementation of that architecture. Most published efforts have focussed on evaluating switch block architectures from the perspective of routability. Without an analysis of the area of the design, routability analysis alone may cause researchers to spend time investigating unbuildable interconnect architectures. No published work discusses the layout area required for different switch block architectures. The area of an FPGA is dominated by programmable interconnect, and the switch blocks are the most complex and challenging component in the interconnect of an island-style FPGA. Therefore it makes sense to begin our analysis of FPGA interconnect architectures with these circuits.

Our final motivation for this effort is that future chip design methodologies may incorporate an FPGA fabric as a core on an ASIC, as described in [6]. Ideally, such FPGA cores could vary the mix of interconnect and cell functionality based on the application domain, the length of the wire segments, and other parameters. This methodology requires an automatic layout methodology for the FPGA fabric. Our analysis of switch block design alternatives strongly implies methodologies for automatically generating switch blocks. Automatically generated switch blocks will probably not be as area efficient as the hand-designed switch blocks that are currently used in commercial FPGAs. Our rationale is that optimized parameters will be more important than optimized layout for the FPGA cores in next generation systems.

1.1 Terminology

A switch block, for purposes of this paper, is a circuit in an island-style FPGA that is situated at the intersection of vertical and horizontal channels, and programmably routes nets from their source channel to their destination channel or channels. Wherever possible, we use the terminology used in most of the literature [11, 4] to describe switch blocks. We will address switch blocks with four sides, and W terminals on each side. The *flexibility* of the module is measured by F_S, which is the number of switches connected directly to each terminal. We will only be addressing switch blocks with $F_S = 3$. Previous efforts have shown the switch block flexibility acheived with $F_S > 3$ does not significantly improve routability [11]. All switch blocks with $F_S = 3$ contain a total of $6W$ programmable switches.

1.2 Related Work

The literature on switch block design primarily assesses the routability of different architectures and presents algorithms to perform the routing through sets of switch blocks. Most of these studies compare their results to the the classical "Xilinx-style" switch block [13], which we here call the *subset switch block*.[1] Figure 1 illustrates the "logical" structure of a subset switch block. In this structure, a port at a switch block connects only to its corresponding port on the three other sides of the switch block. The set of terminals in a subset architecture can be split into W disjoint subsets of four terminals each. The complete interconnection between four terminals in a subset are connected by a circuit called a switchpoint, which consists of at least six logical switches and the structures used to store the state of those switches. Connections between terminals in different subsets cannot be created.

Theoretical studies of switch block architectures determine the total number of routable patterns supported by a particular archi-

[1]These switch block architectures are also known as *disjoint*. [8, 12]

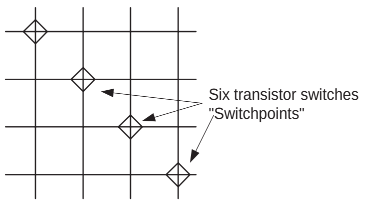

Figure 1: *Logical Architecture of a Subset Switch Block*

W	Optimal Minimum Distance
3	$\sqrt{2}$
4	$\sqrt{5}$
5	$\sqrt{5}$
6	$\sqrt{5}$
7	$\sqrt{8}$
8	$\sqrt{8}$
9	$\sqrt{10}$
10	$\sqrt{10}$
11	$\sqrt{10}$
12	$\sqrt{13}$

Table 1: *Optimal Subset Matrix Distances.*

tecture. The Universal Switch Block [4], is a class of switch blocks that can route any possible two-point routing requirements. This paper [4] proves that subset switch block are not universal and that a universal switch block can be implemented using $6W$ switches. A Hyper Universal Switch Block (HUSB) [5] is an extension of the universal idea to work with nets with more than two points. The Hyper Universal Switch Box requires more than $6W$ switches. They empirically demonstrate an approximate reduction of 10% in the number of routing tracks required to route designs using a HUSB.

Other studies empirically evaluate routability. Wilton [12] presented a switch block architecture which is routable in a fewer number of tracks than the subset architecture. The use of this basic architecture to create FPGAs with wires that span multiple logic blocks is described in [10]. This design is extended in [6] to generate rectangular switch blocks, were the number of vertical and horizonal tracks is not necessarily equal. Other empirical studies include [8, 11].

We are not aware of any published work that focusses on the layout of such switch blocks. In papers such as [11, 12, 10, 6], as well as the VPR tool, layout area is modeled as a linear relationship with the number of switches, which fails to consider the connections between those switches. Our study focusses on the issue of interconnection and placement of configuration and switch resources.

In [1], Betz and Rose describe the generation of FPGA architectures from an abstract architectural description. This research is complementary to ours. Betz and Rose [1] primarily focus on the specification language and the generation of CAD data structures and algorithms that work with a class of specifiable architectures, while this paper describes techniques for automatically generating layout for a critical element of an FPGA architecture from such specifications.

2. LAYOUT OF SUBSET SWITCH BLOCKS

In this section we will present and evaluate a methodology for automatically generating layout for subset switch blocks. The parameter for this generator is the number of terminals W. We assume that the state information for a switch is stored in SRAM cells, and that the switches themselves are implemented with NMOS transistors, although the methodolgy can be applied to a variety of switch and storage technologies.

The logical representation in Figure 1 would not directly produce an efficient layout, because all the switchpoints are situated along the diagonal. Each switchpoint requires a certain amount of active area, A, to implement the six switches and six SRAM cells. Given the diagonal layout, rough dimensions of the switch block would be $W\sqrt{A}$ on a side. The maximum wiring density we could achieve on the switch block would be $1/\sqrt{A}$ wires per unit length, which would be very inefficient use of the metal layer. The active area under the switch block would be only $1/W$ occupied.

There is no reason, however, that logical channel n must be physically adjacent to logical channel $n-1$ or $n+1$. The rows and columns of a switch block can be permuted in order to create a physical representation which is isomorphic to the logical representation. If it was possible to evenly distribute the active area for the switches and configuration RAMs underneath the wires, a wiring density of $1/\sqrt{W \times A}$ wires per unit length could be achieved, which is $\sqrt{W}$ times better than the direct implementation implied by Figure 1.

We have chosen to use a $W \times W$ matrix, P, to represent the possible physical layouts of the subset switch block. The rows of the matrix represent the East and West ports on the switch block, and the columns represent the North and South ports. There is a one in a location (x, y) if and only if there is a switchpoint connecting N_x, S_x, W_y and E_y. When the logical representation is implemented directly, the matrix is an identity matrix. In every case, since $F_S = 3$ there is a single one in each column and each row of the matrix. Therefore P is in general a permutation matrix.

The layout density of the physical layout represented by P can be estimated by determining the minimum distance between any pair of 1's in the P. If this quantity is maximized, there is a high likelihood that the switchpoints are maximally distributed throughout the area under the wires.

We employed a brute force approach to the optimization of switch blocks with small W, by determining the minimum Euclidean distance for every possible $W \times W$ permutation matrix. The results for this experiment are shown in Table 1. Since there are $W!$ possible permutation $W \times W$ matrices, this brute force technique become impossible to use for $W > 12$. Optimal matrices for $W = (4, 8, 9, 12)$, are shown in Figure 2.

2.1 Special Cases

The examples shown in Figure 2 all exhibit some regularity. We will focus on the regularity that is exhibited in two special cases. The first special case is when $W = n^2$ and n is an integer. The second special case occurs when $W = 2n^2$ and n is an integer.

The first special case is demonstrated in the above matrices for P_9 and P_4. Within each such matrix, there is a tilted square mesh of ones like that shown in Figure 3(a). Recall that in this case, $W = n^2$, and $n = 3$. The tilted lines in Figure 3(a) all have slope of $-n$. These lines can be viewed as one line that wraps around the switch block. The mathematical equivalent to this wrapping is the modulus operation. In this particular case, a particular point, (x, y),

$$P_4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad P_8 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P_9 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad P_{12} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2: *Example optimal permuation matrices for* $W = 4, 8, 9, 12$

is a switchpoint in the switch block if it satisfies the equation: [2]

$$y = (6 - 3x) \pmod{10}$$

This equation can be generalized to the following equation for all switch blocks where $W = n^2$.

$$y = (2n - nx) \pmod{n^2 + 1}$$

We will prove this later. In all such designs, the minimum Euclidean distance between any two switchpoints is $\sqrt{n^2 + 1}$.

The second special case occurs when $W = 2n^2$, and is illustrated by the P_8 matrix and in Figure 3(b). In this case, a switchpoint is at point (x, y) if the following equation is satisfied: 2

$$y = (2n + 1)x \pmod{2n^2}$$

The minimum distance between two switchpoints in this layout is $\sqrt{2n^2}$. Since the switchpoint at $(0, 0)$ in Figure 3(b) is on a corner, it could be removed without changing the minimum distance. Therefore this technique also applies to the case where $W = 2n^2 - 1$, which explains why the optimal minimum distance for both P_8 and P_7 is $\sqrt{8}$.

2.1.1 *Layout of Special Cases*

This regularity of these special cases allows structured routing across switch blocks. Figure 4 illustrates how to efficiently implement P_9. In this figure, the assumption is that one metal layer implements the vertical channels and one metal layer implements the horizontal channels. There are nine "switchpoint" components, each consisting of six switches (shown in the figure as NMOS transistors) as well as six configuration storage cells. In this implementation, three sets of wires pass over each switch point in the vertical and horizontal directions. One pair in each direction connects to the switchpoint. The other lines pass over the cell. Because the wires are "shifted" horizontally or vertically for every block, no wire hits more than one switchpoint while passing through the switch block.

[2]In this paper, we will adopt the convention that the operation $x \pmod{y}$ returns a number in the less than y and greater than or equal to zero.

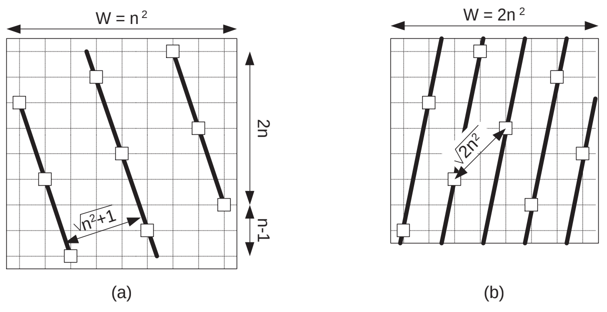

Figure 3: *Two Special Cases of Switch Block Layout: (a) where* $W = 3^2$*, and (b) where* $W = 2 \times 2^2$*.*

2.2 General Case

The proof that these special cases work is based upon the following theorem from basic number theory [9]:

THEOREM 1. *If* $a_1, a_2, ..., a_m$ *comprises a complete residue system modulo* m*, and* k *and* j *are integers, and the greatest common divisor of k and m is one, then the set of numbers* $ka_1 + j, ka_2 + j, ..., ka_m + j$ *is also a complete residue system modulo* m*.*

The trivial case of a complete residue system modulo m is the set of numbers $0, 1, 2, ..., m - 1$. Using this theorem, we can show that the following set of numbers are a complete residue system:

$$(j) \bmod m, (k+j) \bmod m, (2k+j) \bmod m, ..., ((m-1)k+j) \bmod m$$

This set of numbers can be used to specify a permutation matrix P, where $P_{x,y} = 1$ if and only if $y = (kx + j) \bmod m$.

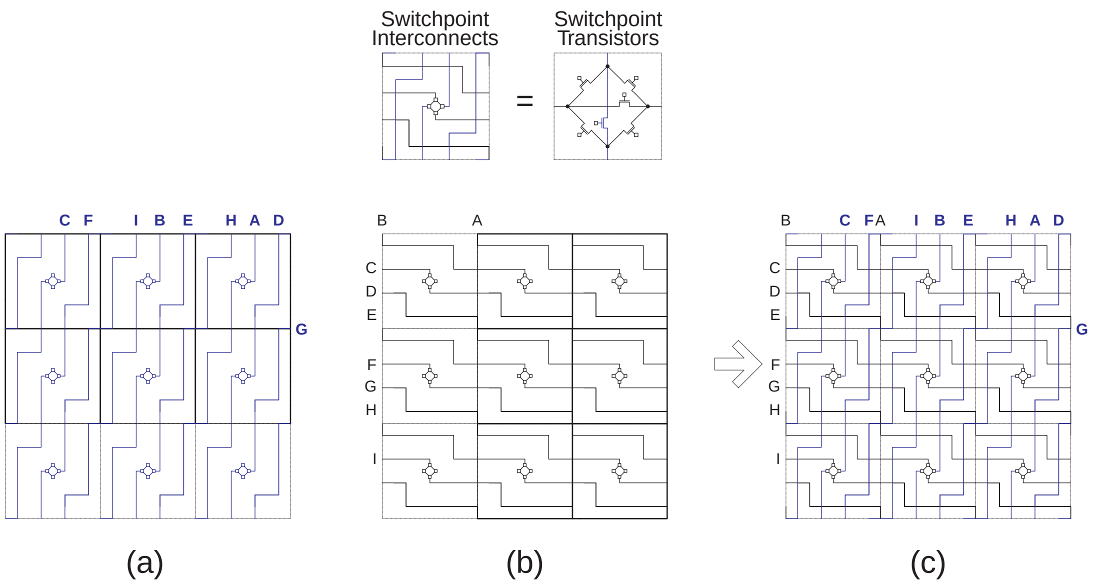

Figure 4: *Layout of P_9. Two individual layers of interconnect are shown in (a) and (b). The overlay of both of these layers is illustrated in (c).*

In the first special case, $k = -n$, $j = 2n$, and $m = n^2 + 1$. For all $n > 1$, $\gcd(-n, n^2 + 1) = 1$. This proves that this congruence defines a permutation matrix, which corresponds to a layout of a subset switch block. In the second case, $k = 2n + 1$, $j = 0$, and $m = 2n^2$. Again, for all $n > 1$, $\gcd(2n + 1, 2n^2) = 1$.

This proof also provides a general methodology for generating regular permutation matrices. We have implemented an algorithm that generates permutation matrices by using the equation:

$$y = (kx + j) \bmod W$$

for all integer values for $0 \le j < W$ and $-W < k < W$ such that $\gcd(k, W) = 1$. For each such equation, we determine the minimum distance of the permutation matrix defined by the equation. The best case for all possible values of k and j is the best case "modulo generated" design for that size switch block.

We also evaluate the matrices defined by the equation:

$$y = (kx + j) \bmod W^*$$

The value of W^* is an element of the set $\{W - 1, W - 2, W - 3, W - 4\}$ and we use all values $0 \le j < W^*$ and $-W^* < k < W^*$ such that $\gcd(k, W^*) = 1$. The smaller matrix defined by this equation is surrounded by different permutations of switchpoints in the corners to generate a permuation matrix of size W. The minimum distance of these matrices are also measured.

The best matrix generated by this routine has the same minimum distance as the proven optimal solutions shown in Table 1. Figure 5 shows the minimum distance of the best matrix for all $W < 100$. The series that corresponds to this approach is labeled **ModGenerated**. We tried a number of other search techniques to increase the minimum distance. None of them worked as well as this technique. This techique has a run time of $O(W^3)$ and takes about one hour to complete all one hundred switch blocks using a 350 MHz Sun Ultra 5 workstation.

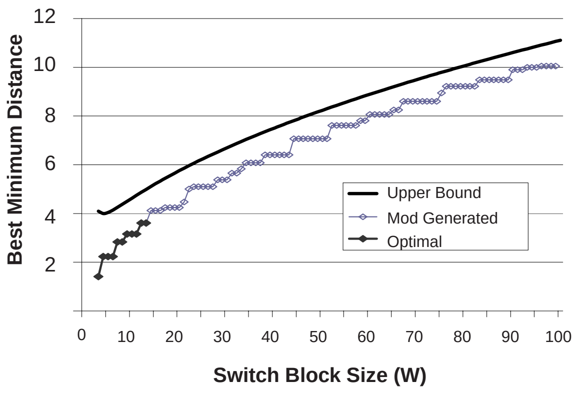

Figure 5: *Best Minimum Distance for Perumuation Matrices.*

2.3 Bounds on layout area

3To determine how good this solution is, we present an infeasible upper bound on the distance as a function of W. If we waive the constraint that each column and row has exactly one 1, then the greatest distance we can achieve between two ones in the matrix occurs if the ones are distributed in a grid which is $\sqrt{W}$ on each side. The maximum distance therefore is $W/(\sqrt{W} - 1)$ because $\sqrt{W}$ divides the W columns into $\sqrt{W} - 1$ difference segments. This upper bound is shown in Figure 5. As illustrated by this graph, the algorithmically obtained matrix tracks this upper bound as W increases.

2.4 Physical Design

Our objective in this symbolic design of switch blocks has been to maximize the distance beween any two switchpoints in the implementation. In reality, the switchpoints will have a fixed area, and the increased symbolic distance between switchpoints translates to increased distance between wiring tracks in the implementation.

We have implemented a SRAM-based switchpoint in a 0.35 micron, four-metal layer process technology. The layout for this design is illustrated in Figure 6. The dimensions of this switchpoint are 18.30 x 22.55 microns. The entire switchpoint is implemented in two layers of metal, which means that the remaining two layers can be used to route the wires over the switchpoint in the horizontal and vertical directions. Currently, the design uses six transistor SRAM cells for configuration storage. The SRAM cells are designed so that, like conventional SRAMs, they share bit, word, power and ground lines. In addition, every other row of SRAM cell are flipped to allow for well sharing. We obtain very close to the conventional SRAM density using this layout. The wires for the SRAM occupy only the first two layers of metal, allowing for signal wires to completely occupy any upper layers.

In Figure 6, all the actual switches are NMOS devices and are two times the minimum size. They are all placed in the channel between two rows of SRAM cells, on the "NMOS side" of the SRAM cells so that they do not cause well spacing problems. All six transistors are aligned so that they can be easily resized. An increase in the width of all six switches will result in an equivalent increase in the switchpoint cell height. A very large increase in transistor width would make this arrangement too tall for our switch block algorithm, which assumes a roughly square switchpoint layout. At that point, a new arrangement of SRAM cells and switches, such as a single row of SRAM cells and a large area for switches would be required.

Using this layout for a switchpoint and assuming minimum width and spacing rules, it is possible to pack 7 wires over each switchpoint. As a result, switch blocks with $W < 49$ will be switchpoint cell-bound, meaning that 100% of the area underneath the switch block will be occupied by switchpoint cells. With $W > 49$ the switch block will be wirebound, meaning that the switch block layout is not fully occupied by switchpoint cells. In a wire-bound switch block design, our methdology will minimize the total area of the design. In the more likely case of a cell-bound design, our methodology will increase the spacing between wires so that:

1. channel wires can have greater width or spacing, which can improve interconnect delay [2];
2. shields can be added between signal wires to reduce coupling and its associated delay; and
3. more wires that travel more than basic layout tile can be routed through the switch block. This will significantly improve routability and delay [7, 3, 10].

2.5 Integration

Figure 7 illustrates the vertical wires in a layout consisting of two switch blocks and a connection block between them. This layout shows how the layout for switch blocks can be used for FPGAs that have wires that span more than one tile. This technique can be extended to allow for wires of arbitrary length and can simultaneously be done in both vertical and horizontal directions.

3. NON-SUBSET SWITCH BLOCKS

In this section we consider the more general layout of switch blocks that may or may not be subset type. All the architectures we evaluate have $F_S = 3$.

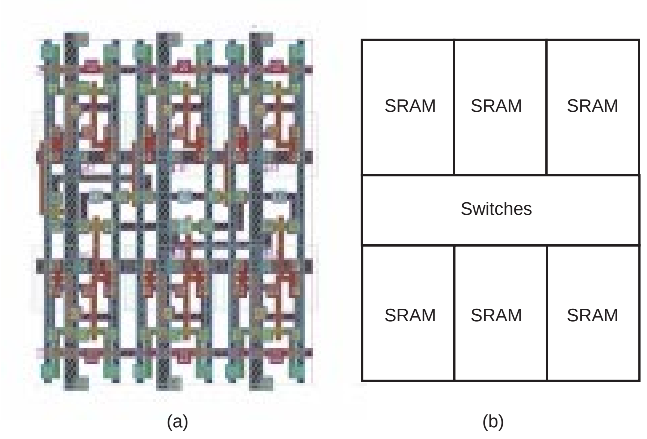

Figure 6: *Layout of a Switchpoint in 0.35 micron technology: (a) shows the actual layout, and (b) shows the boundaries of the SRAM and switch components.*

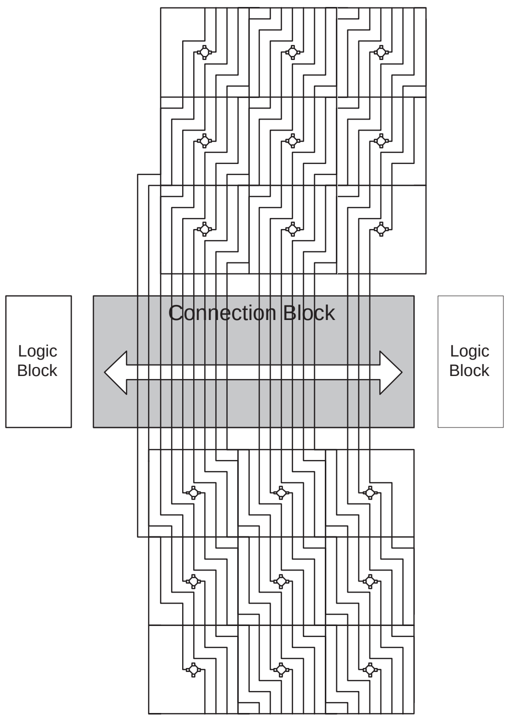

Figure 7: *Layout of one metal layer with channels of 18 wires, with each wire spanning two switch blocks.*

3.1 Representation

The characteristics of the subset switch block made it possible to use a single permutation matrix as a representation of the connectivity. A representation for a more general switch block would have to include the connectivity between the six possible pairs of directions: NS, NE, NW, SE, SW, and WE. To represent the connectivity in each direction pair, we again use a permuation matrix of size W. These six matrices can be used to specify any possible switch block architecture with $F_S = 3$.

Using this representation, a subset switch block could be defined as one in which all six permuation matrices:

$$P_{NS}, P_{NE}, P_{NW}, P_{SE}, P_{SW}, P_{WE}$$

are equivalent.

As in the subset case, we would like to relate this representation to its physical layout. We will do this by transforming the matrix to matrices that are logically isomorphic, and we will evaluate the quality of layout of the isomorphic switch blocks. Notice that in the general case, there is no longer a requirement that N_n be connected to S_n or W_m be connected to E_m. As a result, it is not generally possible to construct a switch block using a mesh of regular wires routed over a set of switchpoints like we did in Figure 4 because the connected terminals on the North and South (and the East and West) do not necessarily line up. As a result we would have to cross pairs of wires running North/South (or East/West) with other wires running in the same direction. This would require additional metal layers and routing tracks for crossing these wires.

While we have no formal proof of this, we believe this is not an effective design style. Instead, we assume that an implementable design will align the connected North/South and West/East terminals, and we achieve this by renumbering the South and East terminals to correspond with the North and South terminals, respectively. Figure 8 shows how to transform an example set of the six matrices to line up the North and South, as well as East and West terminals.

After this transformation of the six matrices is performed, two of the matrices: P_{NS} and P_{WE} are identity matrices. Figure 8 illustrates how the remaining four matrices imply an implementation of this switch block. The wires from the North and South will overlap in some portion of the switch block. The placement of the individual switches that perform routing in the NE, NW, SE, and SW directions is shown. We assume the switches that perform routing the NS and WE directions are somewhere in the overlap area of the North/South and West/East wires.

Given this line up of North/South and West/East, we can determine the placement of switches in the switch block. Figure 9 illustrates the switch placement implied by the representation. A density matrix shown to the right of this figure has a count of the number of switches located at each point in the switch block.

Notice that it is possible to transform this representation and obtain an isomorphic layout where corresponding North/South and West/East pairs of terminals remain connected. This can be done by renaming the terminals in pairs: either North/South pairs or West/East pairs.

3.2 Evaluation Metrics

The problem of evaluating the "layout-ability" of a switch block matrix becomes significantly more difficult in the general case. This is because now we have to consider the placement of individual switches (and their associated configuration state storage), rather than the placement of much larger switchpoints. Using the minimum distance metric is biased towards the subset switch blocks, because only in subset switch blocks is it possible to consolidate all

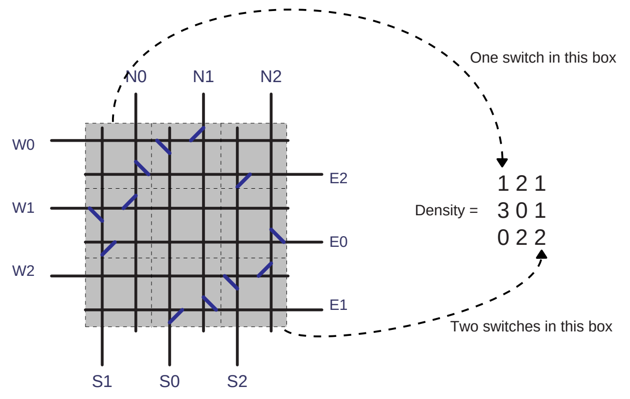

Figure 9: *Switch Placement and Switch Density for Non-subset Switch Block.*

the switches into a set of fully connected switchpoints.

As an alternative metric, we have used an anti-gravitational metaphor. Every point (x, y) in the matrix that has one or more switches is called a *node*. We will denote the set of nodes in the switch as N. All nodes in the matrix exert a repulsive force on all other nodes. This force is proportional to the number of switches at that location, and is inversely proportional to the square of the distance between the two sets of points. Assuming $D(a)$ is the number of switches located at point $a = (x, y)$, then the anti-gravitational force between two nodes, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is computed using the formula:

$$Force(p_1, p_2) = \frac{D(p_1) + D(p_2)}{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The justification for this force is that for each unit of distance between two nodes, there is an area to locate the switches and storage cells that is proportional to the square of the distance between those sets. Using this force, we can gauge how evenly distributed the switches are in the layout, which should indicate how "layout-able" the switch block is.

There are various ways to use this force metric, but no one single way is adequate to completely capture the utility of the design. Through experimentation, we found the most useful way to apply this force metric was to compute the total magnitude of force that each node feels as a result of the force from every other node in the matrix. We call this the *node force* and at a point $a = (x, y)$ it is defined as:

$$NodeForce(a) = \sum_{b \in N, b \neq a} Force(a, b)$$

Taking a weighted average of this node force yields an effective indication of how evenly distributed the forces are. This average is weighted by the number of switches at a node to normalize it:

$$AverageNodeForce = \frac{\sum_{a \in N} [D(a) \times NodeForce(a)]}{\sum_{a \in N} D(a)}$$

Another useful metric is to compute the maximum of all the node forces in the matrix. A third useful metric is to compute the largest single force in the matrix.

We have used these metrics to re-evaluate Subset switch blocks by computing these metrics for every possible switch block. Using

$$P_{NS} = \begin{array}{ccc|c} S0 & S1 & S2 & \\ 0 & 1 & 0 & N0 \\ 1 & 0 & 0 & N1 \\ 0 & 0 & 1 & N2 \end{array} \quad P_{WE} = \begin{array}{ccc|c} E0 & E1 & E2 & \\ 0 & 0 & 1 & W0 \\ 1 & 0 & 0 & W1 \\ 0 & 1 & 0 & W2 \end{array} \quad P_{NE} = \begin{array}{ccc|c} E0 & E1 & E2 & \\ 0 & 0 & 1 & N0 \\ 0 & 1 & 0 & N1 \\ 1 & 0 & 0 & N2 \end{array} \quad P_{NW} = \begin{array}{ccc|c} W0 & W1 & W2 & \\ 0 & 1 & 0 & N0 \\ 1 & 0 & 0 & N1 \\ 0 & 0 & 1 & N2 \end{array} \quad P_{SE} = \begin{array}{ccc|c} E0 & E1 & E2 & \\ 0 & 1 & 0 & S0 \\ 1 & 0 & 0 & S1 \\ 0 & 0 & 1 & S2 \end{array} \quad P_{SW} = \begin{array}{ccc|c} W0 & W1 & W2 & \\ 1 & 0 & 0 & S0 \\ 0 & 1 & 0 & S1 \\ 0 & 0 & 1 & S2 \end{array}$$

$$\boxed{\begin{array}{c} S0 \to S1 \\ S1 \to S0 \\ E2 \to E0 \\ E0 \to E1 \\ E1 \to E2 \end{array}}$$

$$P_{NS} = \begin{array}{ccc|c} S1 & S0 & S2 & \\ 1 & 0 & 0 & N0 \\ 0 & 1 & 0 & N1 \\ 0 & 0 & 1 & N2 \end{array} \quad P_{WE} = \begin{array}{ccc|c} E2 & E0 & E1 & \\ 1 & 0 & 0 & W0 \\ 0 & 1 & 0 & W1 \\ 0 & 0 & 1 & W2 \end{array} \quad P_{NE} = \begin{array}{ccc|c} E2 & E0 & E1 & \\ 1 & 0 & 0 & N0 \\ 0 & 0 & 1 & N1 \\ 0 & 1 & 0 & N2 \end{array} \quad P_{NW} = \begin{array}{ccc|c} W0 & W1 & W2 & \\ 0 & 1 & 0 & N0 \\ 1 & 0 & 0 & N1 \\ 0 & 0 & 1 & N2 \end{array} \quad P_{SE} = \begin{array}{ccc|c} E2 & E0 & E1 & \\ 0 & 1 & 0 & S1 \\ 0 & 0 & 1 & S0 \\ 1 & 0 & 0 & S2 \end{array} \quad P_{SW} = \begin{array}{ccc|c} W0 & W1 & W2 & \\ 0 & 1 & 0 & S1 \\ 1 & 0 & 0 & S0 \\ 0 & 0 & 1 & S2 \end{array}$$

Figure 8: *Representation and Transformation for General Switch Blocks: Top shows six original matrices, bottom shows after alignment of N/S and W/E terminals.*

	Subset			Universal			Wilton			Subset+		
W	AvgNode	MaxNode	MaxForce	AvgNode	MaxNode	MaxForce	AvgNode	MaxNode	MaxForce	AvgNode	MaxNode	MaxForce
4	4.0	4.0	1.6	8.96	9.94	4.0	10.41	11.55	3.0	6.0	6.0	2.4
5	4.15	4.8	1.6	6.73	9.58	2.0	8.80	10.65	2.5	6.23	7.2	2.4
6	4.05	4.59	1.6	7.64	9.84	4.0	8.75	10.97	3.0	6.08	6.88	2.4
7	3.65	4.61	1.0	6.09	7.8	2.0	7.37	9.39	2.0	5.48	6.92	1.5
8	3.61	5.0	1.0	6.56	8.72	4.0	7.41	9.80	3.0	5.41	6.34	1.5
9	3.44	4.23	0.8	5.44	7.05	1.4	6.47	8.09	2.0	5.16	6.35	1.2
10	3.31	4.14	0.8	5.79	7.93	4.0	6.55	8.67	3.0	4.96	6.22	1.2

Table 2: *Different Switch Block Architectures Compared.*

the average node force metric, the brute force optimization technique rediscovered all the optimal layouts for switch blocks with $W < 10$.

Table 2 shows the average node force (**AvgNode**), maximum node force (**MaxNode**), and maximum single force (**MaxForce**) metric for a variety of switch block architectures as a function of size. The value shown in the table is the optimal obtaining using that metric as the goal (i.e. there might be three different designs, each one optimal for the metric.)

Clearly, the subset switch block architecture has a significant advantage over the other two architectures. In all of the metrics, it is nearly twice as efficient as the Wilton and Universal architectures. We expect these results would not change significantly as W increases. Part of this result, we believe, is due to the simplicity and structure of the subset architecture. None of the designs for the non-subset architectures displayed the regularity shown in the subset boxes.

It should be noted that this does not prove that all non-subset switch blocks cannot be efficiently implemented. There might be ways to layout these architectures using a different implementation style, and there could exist non-subset architectures that can be efficiently implemented using this style. For example, we have found some switch block architectures that can be transformed to subset switch blocks matrices by "bundling" pairs of wires together, and implementing a subset switch block using switch points that take four such bundles. This requires a more complex switchpoint layout, but once that is done, the automatic layout is as simple as the traditional subset.

Every non-subet switch block will, however, require some overlap of the north and south wires in each column, as well as the west and east wires in each row. If the objective is to leave as many tracks open in the switch block as possible (for long, double, quad or hex wires for example, or for wire shields), then the subset architecture has an intrinsic advantage.

Finally, we have mentioned that in the non-subset box there is some degree of freedom on the placement of the North-South switch in each column, and the East-West switch in each row. The metrics shown in the first three sets of columns in Table 2 do not include these extra switches. This design freedom does not significantly tip the scales in favor of the non-subset switch blocks, however. In the last set of columns in Table 2, under the label **Subset+** we shown the same experiments and metrics for the subset switch block if the NS and WE switches are placed in the one reasonable location for them. These results still are significantly better than the non-subset switch block metrics. This means a non-subset architecture, even if the extra switches are placed so that their forces were not felt at all by other nodes, would still be worse than an accurate measurement of subset forces.

4. CONCLUSIONS

We presented a style for implementation of subset switch blocks. Using this style, we have determined the optimal design of small subset switch blocks. Our heuristic approach to layout of subset switch blocks consistently produces designs equivalent to the known optimal designs, and quickly produces good results for large switch blocks. These designs have regularity that makes them easy to automatically generate, and they preserve routing tracks for long wires, greater wire spacing and shielding.

We have presented representations and metrics for general switch blocks. We have found that subset switch blocks are consistently superior to other switch block architectures, which implies that the lower routability of subset switch blocks, as measured in the literature, may be more than offset by their efficiency of implementation.

5. FUTURE WORK

As we noted, our work does not conclusively prove that non-subset switch blocks do not have efficient layouts. In fact, recent FPGAs, which do not have subset connectivity, would seem to prove that there exist non-subset switch blocks that can be efficiently implemented. We are investigating ways that we can redefine the switchpoint so that we can create non-subset switch blocks using the subset switch block design technique described in this paper.

In addition, we are developing the techniques for layout generation of connection blocks to provide a complete programmable interconnect design methodology.

Acknowledgements

This research was partially funded by DARPA under the Adaptive Computing Systems program under contract DABT63-96-C-0083, and the Gigascale Silicon Research Center. Herman Schmit is supported in part by a NSF CAREER grant. Sara Stille and Peter Kamarchik did initial layouts for the switch block design.

6. REFERENCES

[1] V. Betz and J. Rose. Automatic generation of FPGA routing architectures from high-level descriptions. In *8th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, (FPGA 2000)*, pages 175–184, Feb. 2000.

[2] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*, chapter 7. Kluwer Academic Publishers, 1999.

[3] S. Brown, M. Khellah, and Z. Vranesic. Minimizing FPGA interconnect delays. *IEEE Design and Test of Computers*, pages 16–23, Dec. 1996.

[4] Y.-W. Chang, D. F. Wong, and C. K. Wong. Universal switch blocks for FPGA design. *ACM Transactions Design Automation of Electronic Systems*, 1(1):80–101, Jan. 1996.

[5] H. Fan, J. Liu, Y.-L. Wu, and C.-C. Cheung. On optimum switch box designs for 2-D FPGAs. In *Proceedings of the 38th ACM/SIGDA Design Automation Conference (DAC)*, pages 201–208, June 2001.

[6] P. Hallschmid and S. Wilton. Detailed routing architectures for embedded programmable logic IP cores. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 69–74, Feb. 2001.

[7] M. Khellah, S. Brown, and Z. Vranesic. Minimizing interconnection delays in array-based FPGAs. In *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference (CICC)*, pages 181–184, May 1994.

[8] G. Lemieux, S. Brown, and D. Vranesic. On two-step routing for FPGAs. In *Proceedings of the 1997 International Symposium on Physical Design*, pages 60–66, Apr. 1997.

[9] W. J. LeVeque. *Elementary Theory of Numbers*. Dover, 1960.

[10] M. I. Masud and S. Wilton. A new switch block for segmented FPGAs. In *International Workshop on Field Programmable Logic and Applications*, Aug. 1999.

[11] J. Rose and S. Brown. Flexibility of interconnection structures for field-programmable gate arrays. *IEEE Journal of Solid State Circuits*, 26(3):277–282, Mar. 1991.

[12] S. Wilton. *Architecture and Algorithms for Field-Programmable Gate Arrays with Embedded Memory*. PhD thesis, University of Toronto, 1997.

[13] Xilinx Inc. *The Programmable Logic Data Book*. 1994.