Exploiting Shortcuts in Fixed Wire Programmable Network

Helia Naeimi
Dept. of CS, 256-80
California Institute of Technology
Pasadena, CA 91125
helia@cs.caltech.edu

ABSTRACT

Exploiting shortcuts in HSRA structure is considered during routing in previous jobs. It is shown in this report that, while keeping partitioning of the design fixed, by changing the placement of the design so that shortcuts are more efficiently used the path delay can be reduced. For solving this problem a greedy local algorithm and a quadrisection-based windowing method are used [1].

1. INTRODUCTION

There are number of fixed wired programmable networks for designing a circuit on a programmable device. The network used here is HSRA, a fat-Pyramid structure having advantages of both hierarchical structure of a tree and a mesh network [2]. The mesh network makes the shortcut connections of HSRA. More details on HSRA structure is given in section 2.

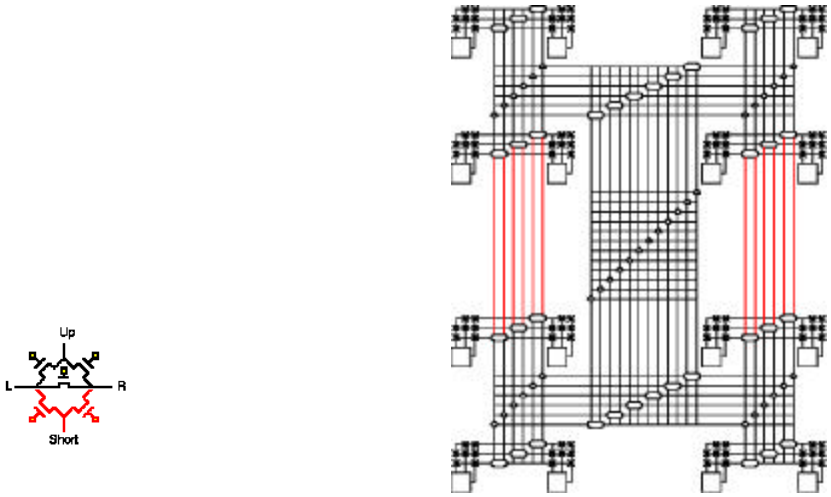
In this project I try to show the advantage of using shortcut connections in time efficiency. Shortcuts were exploited in routing in past works. Here I pulled up using shortcuts to one level upper in circuit designing, placement phase. I also show the benefit of considering shortcuts in upper levels (section 3).

Once the designed is partitioned, it is time to place it on the device. This is the part I exploit shortcuts. The point is that placing logic elements, while considering shortcuts is no more straightforward. They are not placed one after another in the order appeared in partitioning step. Now during placement I can arrange each sub-tree's children configuration so that the largest number of shortcuts is used. To do that, first I divide the design into windows, then locally try to find the best configuration of each window based on a certain cost function (section 4).

2. FIXED WIRE PROGRAMMABLE NETWORK STRUCTURE

There are numbers of fixed wire programmable network structures, like mesh structure, and different kind of tree structures, etc. The structure used in this project is HSRA, Hierarchical Synchronous Reconfigurable Array [3]. It is a fat-pyramid structure. Fat-pyramid has both fat-tree and mesh structures. It combines the advantages of both of them [2].

The logic elements are located at the base level of the tree, on the leaves. The edges of the tree are wires, and the internal nodes are switch boxes. The mesh structure adds shortcuts to this structure. Looking at switch boxes may help to understand HSRA structure better. Fig. 1a shows a switch box, located at internal node, and fig. 1b shows a 4-level HSRA structure.



(a) (b)

Fig. 1 a) A switch box structure. The wires in left and right comes from root of the children sub-trees and the wire in top goes to the upper level, the wire in the bottom comes from an adjacent sub-tree. The shortcut wire can only be connected to the left and right children and not to the upper level.
b) A 4-level HSRA structure. The black wires are tree edges and red wires are shortcuts.

3. WHY EXPLOITING SHORTCUTS

Considering shortcuts in placement step, can affect time efficiency.

Now, how can exploiting shortcuts reduce path delays?

There is especial characteristic in HSRA structure that leads to reducing path delays by using shortcut connections. It is possible that two adjacent sub-trees may not be sibling and their first common ancestor be numbers of levels upper. The common ancestor of two adjacent sub-trees may even be the root of the tree (Fig.2).

Now consider routing case, two adjacent sub-trees have connection, and only tree edges are allowed to be used. For routing, the wires go up from the source sub-tree to their common ancestor, which may be as up as root of the tree, and then com down to the sink sub-tree. Now if using shortcuts is allowed, then the path between these two sub-trees can

be routed considerably shorter, only using a single shortcut wire, not through the long wires and switch elements. In this way using shortcut leads to smaller path delays.

Now, imagine those two sub-trees were not adjacent, so connecting wires must be the tree edges. But there are cases that little change in placement can bring those two sub-trees adjacent, and again exploiting shortcuts reduces the path delays.

4. ALGORITHM OVERVIEW

4.1 Windowing

HSRA structure is highly connected by mesh and tree edges; so considering all the possible configurations of the luts at the base level of the tree is too complicated. Besides, for large designs, the number of luts is really huge, and working with this number of luts and finding the best configuration of them, would have time complexity about O(N!)! So in this case I forget about the best answer to the problem and use a greedy local algorithm.

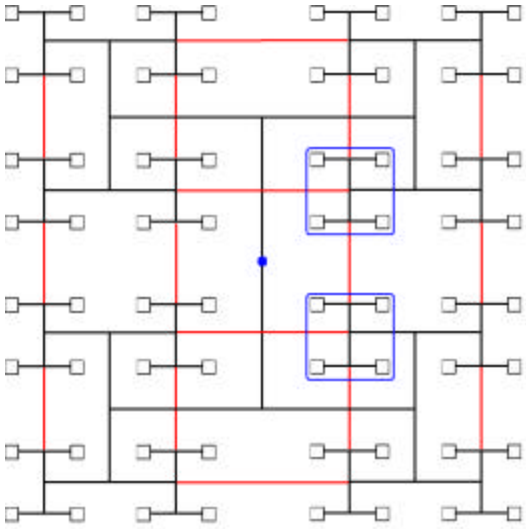


Fig.2 The two marked sub-trees, are adjacent, have shortcut connection, but their first common ancestor is the root of the tree.

First instead of solving the problem at the bottom level of the tree, I look at the upper levels of the tree, and even in each upper level, I try to focus on a small part and then go through the rest of the part in that level. Each of this smaller part is called a window. Each window is a sub-tree of the partitioned design.

For example a partitioned design would make a 7-level HSRA structure. This design can be divided into 2 windows, 1-level lower sub-trees. Each window is divided to four 3-level lower sub-trees called sub-windows [1].

The core of the algorithm is finding the best configuration of sub-windows in each window. The best configuration is evaluated by a cost function that will be defined later in this section. This process is called relaxing a window. It is shown in more details later.

Now consider the case, the algorithm is on level L of the tree. It starts relaxing the windows; it picks up each window, relaxes it and moves to the next window [1]. Once all of the windows are relaxed, the algorithm moves to the first window and repeats this process until the cost function remains unchanged within a certain range.

After relaxing the level L completely, the algorithm moves to two level lower, L-2, and partition each window to 4 new windows, or better say the sub-windows of upper level is now the windows of this level, and goes through the whole relaxing process the same way.

4.2 Priority of the tree edges and shortcuts

Shortcuts that connect sub-trees are located in different levels, that means two sub-tree are highly connected by shortcuts from different levels of the tree. Finding a configuration that uses all of these shortcuts efficiently is not possible, so I give priority to the shortcuts. This priority comes from the priority I set to the edges of tree because each shortcut is used to bypass a tree edge. The algorithm tries to use shortcuts that bypass a highest priority edge available.

Priority of the tree edges are defined based on the level of each edge, the higher level an edge is located the higher priority it gains. The motivation for this definition is that the higher level edges are the common ancestor of a larger number of luts, which means that they would be used by larger number of luts, and also are longer in length than the lower level edges [3].

When the windows are the sub-trees at level L, the most important tree edges are those that connect the sub-trees at level L and L+1, i.e. those edges that connects the windows to each other. Consequently, shortcuts that bypass these edges have the highest priority.

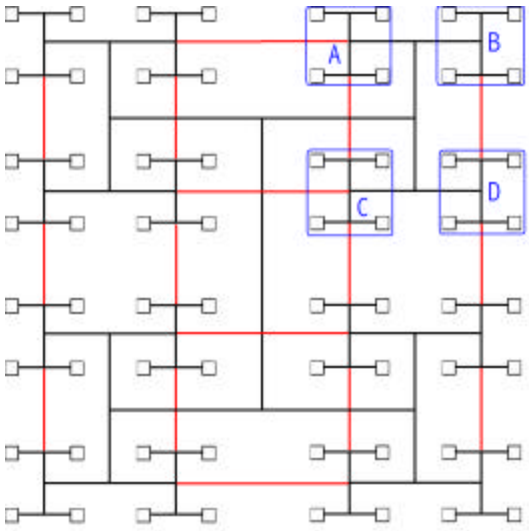


Fig.3 Sub-tree A and B are siblings, so they can exchange their places, and partitioning does not change. Sub-tree A and C belong to different sub-trees so exchanging their places, changes the partitioning.

4.3 Relaxing

Now I should introduce a method to evaluate the sub-windows configuration in each window. To evaluate the sub-window configuration, I calculate a cost function. This cost

function counts the number of potentially used shortcuts. So in this case the goal is to maximize this cost function during relaxation process.

How to find the configuration with the highest cost function?

Each window has four sub-window and they would have 24(=4!) different configurations, but not all of them keeps the partition as it was. Once we have partitioned the design to map to the HSRA, it must remain the same in the next designing steps. For example, consider the window in fig. 3, if we change the place of sub-window A with sub-window B the partitioning remains unchanged, but if we change exchange sub-window A and C then it has changed the partitioning. This means that only flipping the position of two sibling sub-trees is allowed for making a new configuration.

For making a possible configuration, we put joint in each of the internal nodes in a window. By twisting a joint, which means flipping two sub-trees a new configuration is made, and the partitioning remains the same. In fig. 4 all the 8 possible configuration of a window is shown.

How to compute the cost function?

Consider the configuration in fig. 5, for sub-window A all connections between sub-window A and sub-window 0 can be routed through shortcuts, and so can be connection between sub-window A and sub-window 1, sub-window A and sub-window 7. As you can see, sub-window A can have shortcut connections to both sub-windows of upper neighbor, but only shortcut connection to one of the sub-windows of the left neighbor. This happens because of the HSRA structure. If I define (x, y) to be the number of connections between sub-windows x and y, the cost function of the window at fig. 5 is:

Cost_function= (A,0) + (A,1) + (A,7) + (B,0) + (B,1) + (B,2) + (C,4) + (C,5) + (C,6) + (D,3) + (D,4) + (D,5).

For relaxing a window, the algorithm, calculate the cost function for all the possible configuration, and rearrange the window's sub-windows configuration to the one with highest cost function.

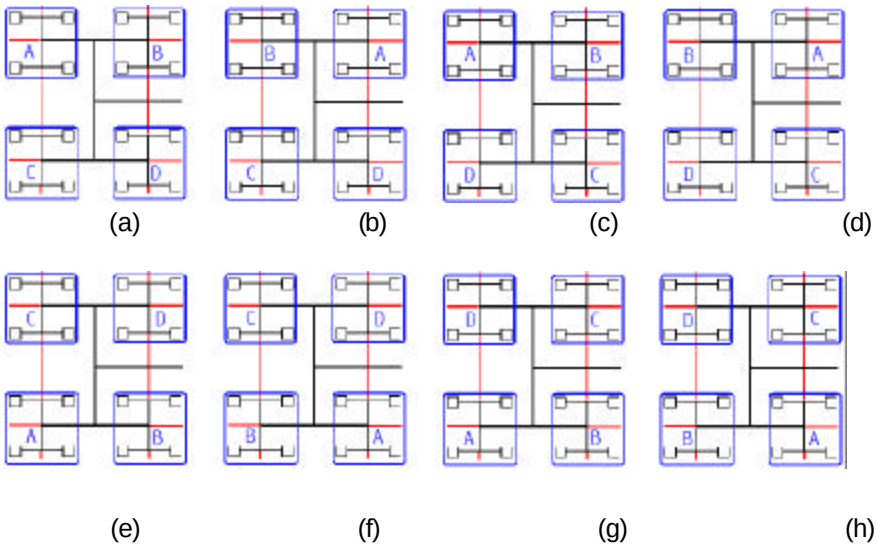


Fig. 4 Eight possible configurations for sub-windows in a window.

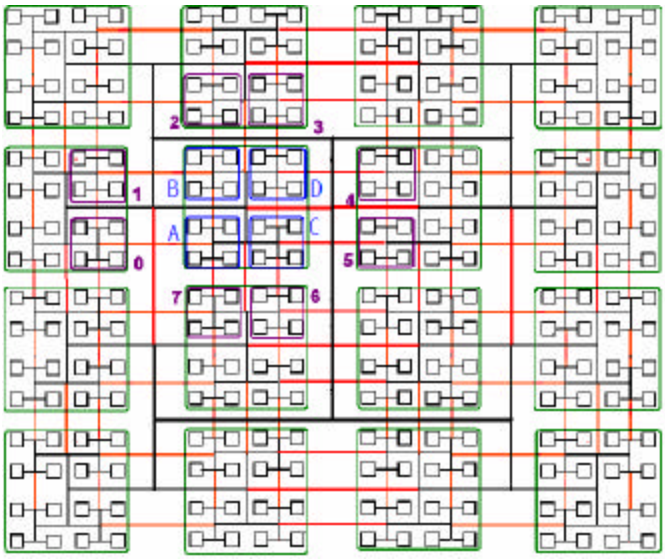


Fig. 5 Computing the cost function.

REFERENCES

[1] D.J.H. Huang, A.B. Kahng, Partitioning-based standard-cell global placement with an exact object. Proceedings of the 1997 international symposium on Physical design 1997 , Napa Valley, California, United States

[2] R.I. Greenburg. The fat-pyramid: A robust network for parallel computation. W.J. Dally, editor, Advanced Research in VLSI: Proceedings of the Sixth MIT Conference, pages, 195-213. MIT Press, 1990.

[3] William Tsu, Kip Macy, Atul Joshi, Randy Huang, Norman Walker, Tony Tung, Omid Rowhani, Varghese George, John Wawrzynek, André DeHon. HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array. Proceedings of the International Symposium on Field Programmable Gate Arrays, pages 69--78, February 1999.