

**Homework # 1**

**Due: September 26**

**Building FPGAs: Design Area versus Functionality**

## 1 Introduction

As we've discussed in class, FPGA design is really about tradeoffs. While the basic architecture of most devices is similar, each vendor device typically has distinguishing features that set it apart from the others. In creating new FPGA architectures designers must explore implementation tradeoffs that lead to effective use of VLSI area. In this exercise, you will examine several of these issues.

This assignment focuses on the island-style FPGA architecture we have been discussing in class. To complete this assignment you will perform a number of experiments using two benchmark circuits commonly used for FPGA research. Several academic CAD tools will be applied to these circuits to help you learn more about the process of translating a text description of a logic circuit into a physical implementation inside an island-style FPGA device. By changing CAD tool parameters, the architecture of the target architecture can easily be modified. Our goal for this exercise is not so much to change the functionality of the tools (this will be explored in PS #2) but rather to examine the effect of architectural issues such as logic block size and wiring network connectivity on the overall size of a target device.

To help you complete the assignment, several reading assignment are noted in the following exercises.

## 2 Island-style FPGAs

Prior to starting this assignment, please read [2], available on the class web site and review [1]. A brief summary of the FPGA model presented in these sources is presented below.

The model for FPGA architecture used in this assignment (shown in Figure 1) models the architecture of several commercial FPGAs from Xilinx and Altera at a high level. The three major architectural parameters in the model are the amount of logic present in each island location labelled  $L$ , the connectivity between the logic elements and the switching network at locations labelled  $C$ , and the connectivity between wires in the switching network at locations labelled  $S$ . In this assignment you will have the opportunity to explore each of these parameters and understand functional tradeoffs.

Each **logic cluster**  $L$  contains collections of one or more look-up tables and flip flops

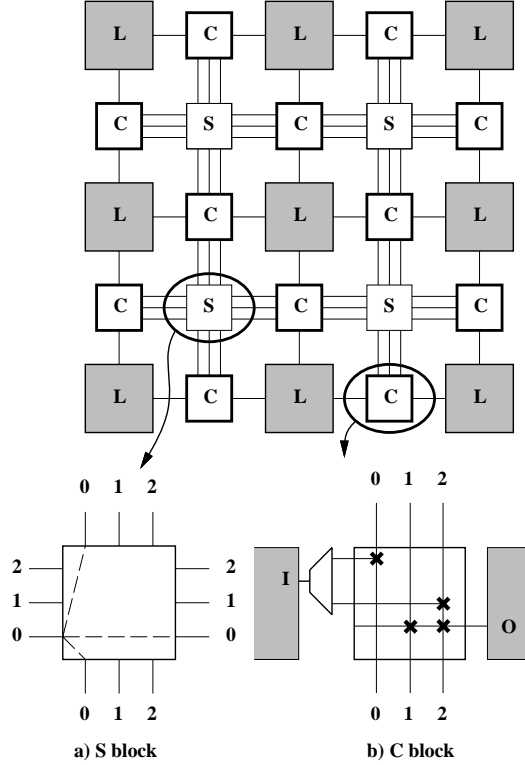


Figure 1: Island-style FPGA model

as described in detail in Section 2 of [2] and shown in Figure 2. The output from each LUT-FF pair or *basic logic element (BLE)* can be either registered or unregistered as needed. Two architectural parameters control the functionality of the logic cluster,  $N$  and  $I$ .  $N$  is the number of BLEs per cluster and  $I$  is the number of cluster inputs. For this assignment, each cluster input can be fanned out to any BLE input through an input multiplexer. Note that all BLE outputs loop back around the BLEs and can be used as BLE inputs.

In Figure 1, routing channels of width  $W$  (in this case 3) are connected to logic clusters through a set of programmable switches, referred to as connection or *C blocks*, at the intersection of logic cluster IO terminals and channel tracks. The flexibility of the C block or  $F_c$  is represented by the fraction of channel tracks that can possibly connect to a logic element. In the example shown in Figure 1b only two of the tracks could possibly attach to either an input or output so  $F_c = 0.66$ . Note that in some literature (e.g. [5])  $F_c$  is defined as the absolute number of tracks connected to the cluster I/O, so  $F_c = 2$  for Figure 1b using this notation.

Wire segments in the routing channels span one logic cluster in the horizontal or vertical dimension. Switchboxes, or S blocks, allow a predefined set of programmable connections between wires at the intersection of horizontal and vertical track channels. Figure 1a shows that each switchbox is sparsely connected so that each horizontal or vertical wire entering the switchbox can connect to only three possible destinations ( $F_s = 3$ ).

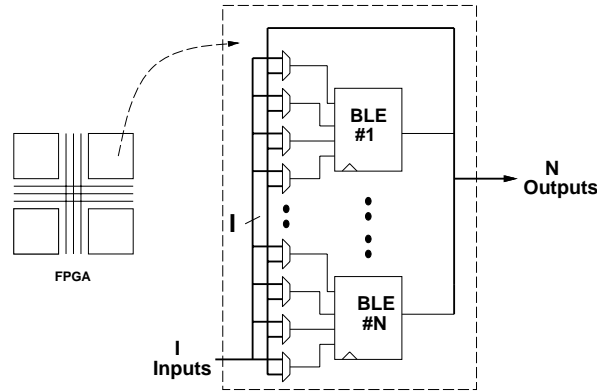
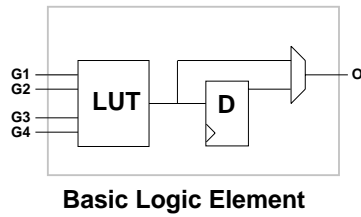


Figure 2: Logic Cluster

### 3 Getting Started

Completion of the assignment will require the use of three academic CAD tools written by Vaughn Betz and Sandy Marquardt at the University of Toronto. These tools are applied to several benchmark logic design circuits from the MCNC FPGA benchmark suite. The source for these tools is well commented and you are encouraged to look over parts of the code to better understand how results are generated. Needed benchmark circuits are located on the course web site in file **ps1.tar.gz**. VPR, version 4.30, can be obtained from the VPR web site, which can be accessed from the course web page. Please follow the directions on the course web site to obtain the necessary tools and configure them on your computer. The standard distribution of VPR works on Sun Solaris. If you use the **makefile** on the course web site, VPR should compile under Linux for U of T ECF computers. VPR has also been successfully compiled for other architectures (HP Unix, MS Windows). You are free to use any type of computer for this assignment.

Two benchmark circuits, **apex2** and **pdcc**, and associated assignment makefiles may be found in subdirectory **tests** in the distribution. These should be used for all exercises. Initially circuits are represented as a collection of lookup tables and flip flops described in *blif*, a common language for describing circuits. These circuits will be translated into logic clusters using **T-VPack**, a logic clustering tool, and placed and routed using **VPR**, an academic placement and routing tool. Following layout, **trans\_count** is used to evaluate

the total number of routing and logic transistors needed to successfully implement the circuit in a minimum-sized island-style FPGA. It is suggested that you look over the VPR and T-VPack User's Manual, manual430.pdf, prior to starting the following experiments. The T-VPack description will help you understand how T-VPack operates and the role of various parameters. You can learn more about trans\_count parameters by going to the trans\_count directory and typing **trans** with no arguments.

## 4 Varying Logic Cluster Size

For the first set of exercises you will determine the relationship of  $I$ , inputs per logic cluster, to  $N$ , BLEs per logic cluster, for several cluster-based island-style devices. We shall determine the appropriate number of inputs per cluster by repeatedly applying T-VPack with different numbers of inputs specified per cluster. A similar set of experiments is described in Section 5 of [2] and Section V.A of [1].

### Ex 1: Logic Clustering Algorithm

Summarize the clustering algorithms described in Section 4 of [2] and in Section 6.2 of [5]. How are the approaches similar? How are they different? Please limit your discussion to somewhere between half and a whole page.

According to the given clustering algorithm, the number of inputs required by BLEs derived from the logic design and assigned to a logic cluster must be less than the number of inputs available,  $I$ , even if some BLEs in the cluster must be left unpopulated. Clearly, minimizing  $I$  in an architecture is beneficial since fewer inputs per cluster means fewer input switches and potentially more device area for a larger logic array. However, if  $I$  is too small many of the BLEs in each cluster may be wasted and the larger logic array may be eaten up by sparsely-filled clusters. The key to determining  $I$  is to find the minimum  $I$  value per cluster that still allows high total BLE utilization in the device. The number of potential values of  $I$  for a cluster range between 4, the number of inputs for one BLE, and  $N \times 4$ , the total number of all BLE inputs in the cluster.

### Ex 2: Comparing $N$ to $I$

For designs **apex2** and **pdic** repeatedly apply T-VPack to evaluate tradeoffs between  $I$ ,  $N$ , and the number of logic clusters needed to implement a design. This analysis can be accomplished with the following example steps:

- Examine the makefile in directory tests/apex2. Note the variables CLUSTER\_SIZE and INPUTS\_PER\_CLUSTER assign  $N = 4$  and  $I = 10$  for this example. Before making changes to the makefile make a copy of it for safekeeping. Change the variable CPATH to point to your local copy of the CAD tools.
- Read over the description of T-VPack in manual430.pdf paying particular attention to the parameters the tool takes as input. Compare these to the parameter setting located in the makefile. Run T-VPack by typing **make apex2.net**. The result of T-VPack is a netlist which can be used for subsequent circuit placement and routing.

Repeat this experiment for  $N = 4$ ,  $I = 4, 8, 9, 10, 11, 12, 13, 14, 16$  and  $N = 8$ ,  $I = 4, 8, 12, 16, 18, 19, 20, 21, 22, 24, 28, 32$  for both **apex2** and **cdc**. To simplify this task you may wish to modify and use the shell script *foreach.script*. Plot results so that the fraction of total available BLEs used per design appears along the vertical axis and  $\frac{I}{4N}$  appears along the horizontal axis. What can you summarize about  $I$  for these experiments? What minimum values of  $I$  achieve 98% BLE utilization for each design and value of  $N$ ?

## 5 Evaluating VLSI Area Costs

An important aspect of FPGA design is understanding how much VLSI area is needed to implement a specific circuit. In the previous section we evaluated tradeoffs in logic block size. Here, we include routing area in the evaluation.

To a first order the area of an FPGA device can be estimated by determining the number of minimum-width transistors needed to construct the device. While a full device layout would be needed to get the most accurate assessment, in this exercise we compare FPGA array sizes by counting the number of transistors needed to implement both interconnect and logic for various device sizes. In general, interconnect transistors include those found in C-blocks and S-blocks and logic transistors include those found in logic clusters. Fringing effects due to IO pad connections are not considered in this analysis. In the following exercises you will take netlists created by T-VPack and apply VPR to first place the circuit in a device with just enough logic clusters to hold it and then route interconnections using the minimum number of tracks per channel needed to route it,  $W_{min}$ . Although the placement and routing processes are controlled by makefile settings that have been preset for the following experiments, you are encouraged to experiment with them to become familiar with the tools.

Following placement and routing, **trans\_count** is used to determine the number of logic and routing transistors needed to implement each circuit. This program takes as input the minimum number of tracks per channel found from routing ( $W_{min}$ ), the number of blocks per cluster ( $N$ ), number of inputs per cluster ( $I$ ),  $F_c$ , and  $F_s$ . All of these parameters are predetermined except for  $W_{min}$  which is found after routing by VPR.

If you wish, you can use routing area numbers reported from VPR rather than **trans\_count** for the following experiments. These numbers are somewhat more accurate than the numbers reported by **trans\_count** as they take more routing factors into account. Note that you will still need to determine logic cluster transistor counts using **trans\_count** if you chose to determine routing area using the VPR reported values.

Perhaps the best way to learn about how these tools work is through an example. Consider the following steps:

- For design apex2, use VPack to create a .net file with  $N = 4$  and  $I = 10$ . Note that the ARCH\_FILE variable in the makefile is 4x4lut\_sanitized.arch to allow for VPR compilation with  $N = 4$  and  $F_c$  in the makefile is 0.5, the connection block

flexibility for  $N = 4$ . Note that these two values should be set to `4lut_sanitized.arch` and  $F_c = 1$ , respectively, for later experiments with  $N = 1$ .

- Create a placement for the design by typing **make apex2.p**. Pay special attention to the size of the array that is targetted. VPR determined that this was the minimum square array size that would hold this circuit.
- Route the new placement by typing **make apex2.r**. Note that the router will try routing at a number of track widths until a minimum value,  $W_{min}$ , is found that will successfully complete routing. **Note: routing for each design will require several minutes.**
- In the makefile, change the variable `WIDTH` to  $W_{min}$  found in the previous step. Now type **make apex2.area** to get the per logic cluster area of the device. This should be scaled by the array size to determine the total number of transistors needed to implement the design.

### **Ex 3: Evaluating Area Costs**

Repeat the above experiment for design **pdcc**,  $N = 4, I = 10$  and for both **apex2** and **pdcc**,  $N = 1, I = 4$ . **Be sure to check  $F_c$  and `WIDTH` each time you run `trans_count` so that you obtain accurate results.** As noted in the makefile  $F_c$  should be set to 1 for  $N = 1$  and 0.5 for  $N = 4$ . Summarize the results of your four experiments in a table with a brief description of what you found. The table should include the number of transistors needed to implement the whole device in addition to the number of transistors per logic cluster. Where are most of the transistors located in the FPGA, in the logic or in the interconnect? Explain why  $F_c$  should be less for larger cluster sizes. Consider using a script (like *foreach.script*) to speed up your experiments. **Optional:** Try other cluster sizes such as  $N = 8$ . Examine file `trans_count/trans_logic.c` to determine exactly how transistors that make up logic blocks are allocated.

## **6 Connection Box Flexibility**

In this section you will evaluate the effect of connection block flexibility on transistor count. This will be accomplished by rerunning the VPR router and `trans_count` with different  $F_c$  settings using the same placements found in the previous exercise.

### **Ex 4: Changing connection block flexibility**

Rerun the above experiments for **pdcc** and **apex2** for  $N = 1, I = 4$ , this time with  $F_c = 0.6$  and  $F_c = 0.8$ . Start from the previously determined `.p` placement files and rerun the VPR router and `trans_count`. **Note: Before doing the experiments make sure  $F_{c\_input}$  and  $F_{c\_output}$  in file `vpr/4lut_sanitized.arch` and  $F_c$  in the local makefile (eg `tests/apex2/Makefile`) have all been set to the correct value (either 0.6 or 0.8).** Summarize your results in a short paragraph and table including values for  $F_c = 1$  found in the previous exercise. The table should include the number of transistors needed to implement the whole device in addition to the number of transistors per logic cluster.

### **Ex 5: Changing connection block flexibility**

Rerun the above experiments for **pdc** and **apex2** for  $N = 4, I = 10$ , this time with  $F_c$  of 0.3 and 0.7. Be sure to follow the same procedure as the previous exercise regarding modifying files except note that `vpr/4x4lut_sanitized.arch` should be used. Summarize your results in a short paragraph and table including values for  $F_c = 0.5$  found previously. The table should include the number of transistors needed to implement the whole device in addition to the number of transistors per logic cluster.

## **7 Switchbox and Intra-Cluster Flexibility**

In this section you have the opportunity to evaluate the effect of switchbox flexibility on FPGA array area for switchboxes that contain both pass gates and tri-state buffers.

### **Ex 6: Transistor counts for switchboxes**

How many transistors are needed to implement a switchbox with 2 inputs per side and with  $F_s = 3$  and  $F_s = 4$  if pass transistors are used for connections. How does the value change if bidirectional tri-state buffers are used? Note Figure 4 in [5] and Figure 9 in [4] for ideas regarding buffer count and transistors per multiplexer. How many transistors are needed to implement a directional switch box with  $F_s = 3$  and two I/Os per side (note Figure 4 in [3]). Create a table to summarize your results and rank the possible implementations for performance.

### **Ex 7: FPGA Intra-cluster Multiplexer size**

How many transistors are needed to implement a LUT input multiplexer similar to the ones shown in Figure 2 as a function of  $N$ ,  $I$ , and  $K$ ?

## **8 FPGA Architecture**

In this section you will comment on the results of Ahmed and Rose [1] in terms of FPGA area and delay.

### **Ex 8: FPGA Area Tradeoffs**

Briefly describe some of the factors which lead to the area curve shapes in Figures 7 and 8 of [1]. Summarize your arguments in a few paragraphs.

### **Ex 9: FPGA Delay Tradeoffs**

Briefly describe some of the factors which lead to the delay curve shapes in Figure 14 of [1]. Summarize your arguments in a few paragraphs.

## References

- [1] E. Ahmed and J. Rose. The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density. *IEEE Transactions on VLSI*, Mar. 2004.
- [2] V. Betz and J. Rose. Cluster-based Logic Block for FPGAs: Area-Efficiency vs. Input Sharing and Size. In *Proceedings, Custom Integrated Circuits Conference*, 1997.
- [3] G. Lemieux, E. Lee, M. Tom, and A. Yu. Directional and Single-driver Wires in FPGA Interconnect. In *IEEE International Conference on Field Programmable Technology*, Brisbane, Australia, Dec. 2004.
- [4] D. Lewis. The Stratix II Logic and Routing Architecture. In *International Symposium on Field Programmable Gate Arrays*, Monterey, Ca., Feb. 2005.
- [5] A. Marquardt, V. Betz, and J. Rose. Using Cluster-Based Logic Blocks and Timing-driven Packing to Improve FPGA Speed and Density. In *International Symposium on Field Programmable Gate Arrays*, Monterey, Ca., Feb. 1999.