**1 Introduction**

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. To complete this assignment you will perform a number of experiments based on the benchmark circuits used in homework-1. 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. Our goal for this exercise is not so much to change the functionality of the tools (this will be explored later) 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.

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. A set of experiments for determining the relationship of I, inputs per logic cluster, to N, BLEs per logic cluster, are described in Section 5 of [2] and Section V.A of [1]. We have also discussed the relationship between N, K and I with post-routing area and delay in detail based on [1]. In this assignment we will extend this analysis to wirelength, critical path delay and power. You will answer the following fundamental questions:

* What is the impact of logic block and routing architecture configuration choices on post-routing wirelength, delay and power metrics?
* Is there any correlation between wirelength, delay and power metrics?

**2 Island-style FPGAs**

Prior to starting this assignment, please read [2], and review [1] and [6]. 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 labeled L, the connectivity between the logic elements and the switching network at locations labeled C, and the connectivity between wires in the switching network at locations labeled S. In this assignment you will have the opportunity to explore each of these parameters and understand functional tradeoﬀs.

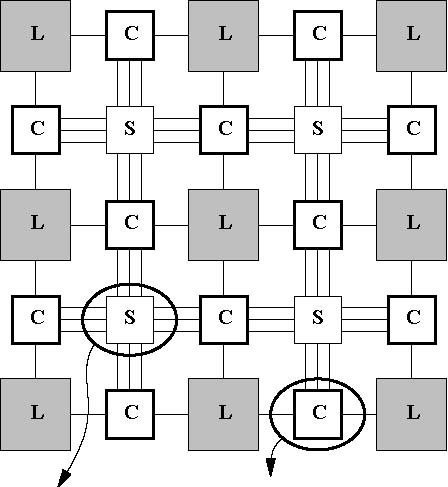
****

Figure 1: Island-style FPGA model

Each logic cluster L contains collections of one or more look-up tables and ﬂip ﬂops 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. 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 Fc 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 Fc =0.66. Note that in some literature (e.g. [5]) Fc is deﬁned as the absolute number of tracks connected to the cluster I/O, so Fc = 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 (Fs =3).

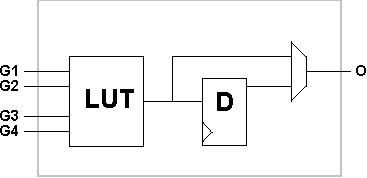
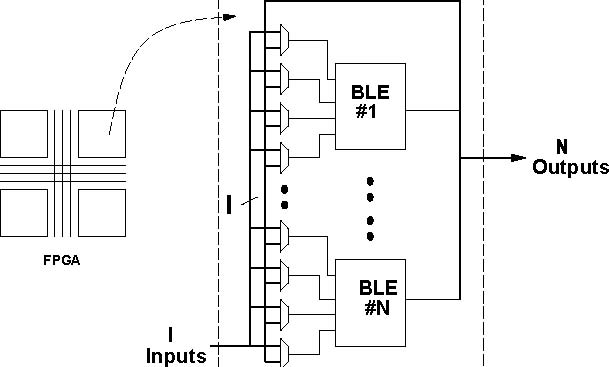
 

Figure 2: Basic Logic Element and Logic Cluster

**3 VersaPower**

VersaPower is a power model incorporated with VTR to model the power usage of many different FPGA architectures. Prior to starting this assignment, please read [7].

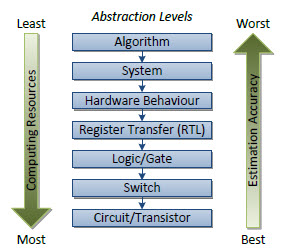
In the past, FPGA power models have been developed; however they were designed to target single, simple architectures with known circuitry. VersaPower incorporates a new technique estimating power usage for many different user-created architectures. Even though the user specifies the functionality of the FPGA architecture, the physical circuitry is not specified. VersaPower’s new technique translates these functional descriptions into physical circuits. From this translation, standard methods can be used to estimate power dissipation.

Figure 3: Circuit abstraction levels used for power estimation, and impacts on accuracy and resources

When compared to past models, VersaPower provides substantially more detailed static power estimations, which are increasingly relevant as CMOS is scaled to smaller technologies. Therefore the model can incorporate and operate with modern CMOS technologies, which in turn have been validated against SPICE using 22 nm, 45 nm, and 130 nm technologies.

Since Versapower has such an enlarged architecture support, the model provides support for modern FPGA features such as fracturable look-up tables and hard blocks (which will not be used for the experiments defined below). Results show that for common architectures, nearly 73% of power consumption is due to the routing fabric, 21% from logic blocks and 3% from the clock network. For architectures supporting fracturable look-up tables, roughly 3.5-14% more power is required, as each logic element has additional I/O pins, increasing both local and global routing resources.

Below in figure 4, you’ll see how the model is integrated into the VTR CAD flow. Notice the details in the architecture generator – granted this module is the largest piece of VersaPower’s power model. The architecture generator generates the entire FPGA circuitry based on the user-described architecture.

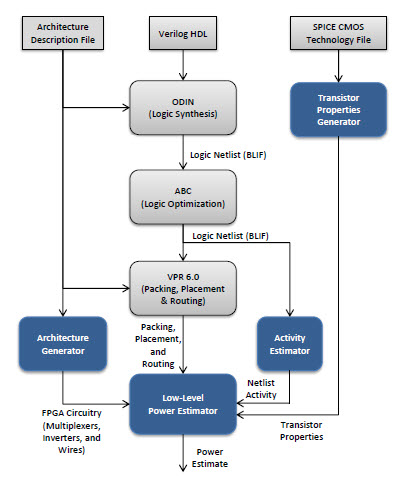
**

Figure 4: Modifications to the VTR flow for power estimations

Please refer to:

<https://vtr.readthedocs.io/en/latest/>

<https://vtr.readthedocs.io/en/latest/vtr/power_estimation/>

The VersaPower source code is located:

/vtr\_release/vpr/SRC/power/

To run VTR flow with VersaPower:

$VTR\_ROOT/vtr\_flow/scripts/run\_vtr\_flow.pl  $VTR\_ROOT/vtr\_flow/benchmarks/vtr\_benchmarks\_blif/sha.blif $VTR\_ROOT/vtr\_flow/arch/timing/k6\_frac\_N10\_mem32K\_40nm.xml -power -cmos\_tech $VTR\_ROOT/vtr\_flow/tech/PTM\_130nm/130nm.xml  -starting\_stage abc -keep\_intermediate\_files -temp\_dir sha\_temp

**/vtr\_flow/scripts/run\_vtr\_flow.pl <verilog> <architecture> -power –cmos\_tech <tech\_xml>**

Power information will be output in the same intermediate circuit folder as vpr.out, thus:

/sha\_temp/sha.power

For the given experiments below, we will be using tasks, thus similar to running VTR flow with VersaPower, the config file must be updated accordingly.

The first config file below will allow the minimum channel width to be determined for each benchmark. Note: Update your architecture paths accordingly with respect to where the architecture is placed (currently the config file shows it is placed in ‘arch/power’) they are in VTR. Please note the changes in yellow highlight (default architecture file, power flag, and cmos tech behavior)

##############################################

# Sample Configuration files for running

# experiments

##############################################

# Path to directory of circuits to use

circuits\_dir=benchmarks/vtr\_benchmarks\_blif

# Path to directory of architectures to use

**archs\_dir=arch/timing**

# Add circuits to list to sweep. This is just a

# sample. you need to modify this list to include

# your assigned benchmarks.

circuit\_list\_add=stereovision0.blif

circuit\_list\_add=diffeq1.blif

circuit\_list\_add=sha.blif

circuit\_list\_add=blob\_merge.blif

# Add architectures to list to sweep

**arch\_list\_add=k6\_frac\_N10\_mem32K\_40nm.xml**

# Parse info and how to parse

parse\_file=vpr\_standard.txt

# Parameters for vtr flow

script\_params= -no\_mem -starting\_stage abc -check\_equivalent -keep\_intermediate\_files **-power**

# Pass requirements

pass\_requirements\_file=pass\_requirements.txt

# Spice technology input parameters

**cmos\_tech\_behavior=PTM\_130nm/130nm.xml**

**Ex1:** The baseline architecture file is included in the hw4 package **(k4-n10-i22-fs3-fcin0.25-l4.xml)**. This architecture file is different from hw3. Therefore, you need to first generate the minimum channel for each benchmark and fill in the table below.

|  |  |
| --- | --- |
| Benchmarks | Minimum channel width |
| Apex2 |  |
| diffeq |  |
| elliptic |  |
| ex1010 |  |
| ex5p |  |
| frisc |  |
| s298 |  |
| s38417 |  |
| spla |  |
| tseng |  |
| sha |  |
| blob\_merge |  |
| stereovision0 |  |

**Please use the run\_vtr\_task.pl given in the VTR. Commands are given below:**

**Run Task:**

path: <vtr>/vtr\_flow/scripts/run\_vtr\_task.pl

**Parse Task:**

path: <vtr>/vtr\_flow/scripts/parse\_vtr\_task.pl

parse\_vtr\_task.pl <path/task\_name>

The channel width obtained above should be relaxed by 30%. Next we update the config.txt accordingly to incorporate a fixed channel width for all benchmarks. It is highly recommended to generate a script, or modify the example script which is also packaged with hw4, when sweeping a given parameter. Since several architecture files will be produced, update the arch\_list accordingly for the architecture files you sweep.

The second configuration file below will allow a fixed channel width to be defined for all circuit and architecture files specified. Notice the parse\_file, script\_params, and pass\_requirements are each updated.

##############################################

# Configuration files for running experiments

##############################################

# Path to directory of circuits to use

circuits\_dir=benchmarks/blif

# Path to directory of architectures to use

archs\_dir=arch/power

# Add circuits to list to sweep. This is just a

# sample. You need to modify this list to include

# all benchmarks and use proper relaxed channel

# width amount. Channel width values given below

# are random numbers. Do not use them in your

# experiments.

circuit\_list\_add=stereovision0.blif

channel\_width\_list\_add=58

circuit\_list\_add=diffeq1.blif

channel\_width\_list\_add=78

circuit\_list\_add=sha.blif

channel\_width\_list\_add=45

circuit\_list\_add=blob\_merge.blif

channel\_width\_list\_add=38

# Replace the architecture file

# here. You can list all architecture files here

# when sweeping a parameter. However, if an

# architecture file causes trouble, it may become

# hard to figure out which one is failing. We

# recommend you to use single architecture file at a

# time over the list of assigned benchmarks. arch\_list\_add=k4\_N10\_I22\_Fi6\_L1\_frac0\_ff1\_45nm.xml

# Parse info and how to parse

**parse\_file=vpr\_wirelength.txt**

# Pass requirements

**pass\_requirements\_file=pass\_requirements.txt**

# Parameters for vtr flow

script\_params= -no\_mem -starting\_stage abc -check\_equivalent -keep\_intermediate\_files -power

# note that if you wanted to keep the same channel

# width for all benchmarks, you could have used

# the following command. This is not recommended for

# this assignment.

# vpr\_route\_chan\_width <insertyourchannelwidthhere>

# Spice technology input parameters

cmos\_tech\_behavior=PTM\_130nm/130nm.xml

**For the above config file that specifies channel width for each benchmark, you must use “ece506\_run\_vtr\_task.pl” provided in the hw4 folder.**

ece506\_run\_vtr\_task.pl <path/task\_name>

**Ex2:** Table below summarizes the architecture parameters and their sweeping range. Execute the following experiments using the same relaxed channel width.

|  |  |  |
| --- | --- | --- |
| Architecture Parameter | Sweeping Range | Default Value |
| LUT size,  **K** | 3-7  (increment of 1) | * 4 |
| Cluster size,  **N** | 2-20  (increment of 2) | * 10 |
| Inputs per cluster,  **I** | 4-40  (increment of 2) | * I-Limited architecture when sweeping logic architecture parameter, * 22 when sweeping routing architecture parameters |
| Switchbox Flexibility, **Fs** | 3-21  (increment of 3) | * 3 |
| Input connection box Flexibility, **FCin** | 0.1-1.0  (increment of 0.1) | * 0.25 when sweeping logic architecture parameters * 0.1 when sweeping routing architecture parameters |
| Output connection box Flexibility, **FCout** | 0.1-1.0  (increment of 0.1) | * 0.25 when sweeping logic architecture parameters * 1.0 when sweeping routing architecture parameters |
| Segment length,  **L** | 1-6  (increment of 1) | * 4 when sweeping logic architecture parameters * 4 when sweeping routing architecture parameters |

**For all configurations, the architecture files are provided in the “arch” folder of this assignment.**

**All benchmarks (.blif) are located in the “benchmarks” folder.**

**Experimental Flow**

**Step1:**

For each **benchmark** {your assigned benchmarks, see table below}

For each **architecture** **parameter** {K, N, I, Fs, Fcin, Fcout, L}

For each **sweeping value** of architecture parameter under evaluation

1. **Fix** all other parameters to their default values
2. **Run** VTR
3. **Collect** the following metrics: {average wirelength, critical path delay, total power consumed by routing resources, and total power consumed by logic block resources}. Wirelength and delay metrics will be found in the vpr.out and the power metrics will be in the <circuit\_name>.power files.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Group | | | | | | | | | | | | |
| Benchmarks | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 13 | 14 |  |
| Apex2 |  | x | |  | x | x |  | x | x | x |  |  |  |  |  |
| diffeq |  | x | |  | x | x | x |  | x | x |  |  |  |  |  |
| elliptic |  |  | | x | x | x |  |  |  | x | x |  |  |  |  |
| ex1010 |  |  | | x | x | x |  |  |  | x | x |  |  |  |  |
| ex5p |  | x | | x | x | x |  | x |  |  | x | x |  |  |  |
| frisc | x | x | | x | x | x |  |  |  |  | x | x |  |  |  |
| s298 | x | x | | x |  | x | x |  |  |  |  | x | x |  |  |
| s38417 | x | x | | x |  | x |  |  |  |  |  | x | x |  |  |
| spla | x |  | |  |  | x |  | x |  |  |  |  | x | x |  |
| tseng | x |  | |  |  | x | x |  |  |  |  |  | x | x |  |
| sha | x |  | |  |  | x | x | x |  |  |  |  |  | x |  |
| blob\_merge | x | x | | x | x | x |  |  | x |  |  |  |  | x |  |
| stereovision0 | x | x | | x | x | x |  |  | x |  |  |  |  |  |  |

**Important Notes:**

1. The experimental flow described above assumes an N-limited architecture. Please note that for the N-limited FPGA, the number of logic inputs is calculated based on the following expression.

However, when sweeping the I value, you will fix the K and N to 4 and 10 respectively. Therefore, the sweeping range of I will allow you to evaluate both I-limited and N-limited architectures.

1. A sample python script to sweep Fs is included in hw “vtr\_fs\_sweep.py”. You can run it as shown below. Please note that the script has been tested using the rcl servers, however may require additional packages when run on other machines (i.e. python-numpy).

Usage: python vtr\_fs\_sweep.py arch\_file range increment

Example: python vtr\_fs\_sweep.py k4-n10-i22-fs3-fcin0.25-l4.xml 3-15 3

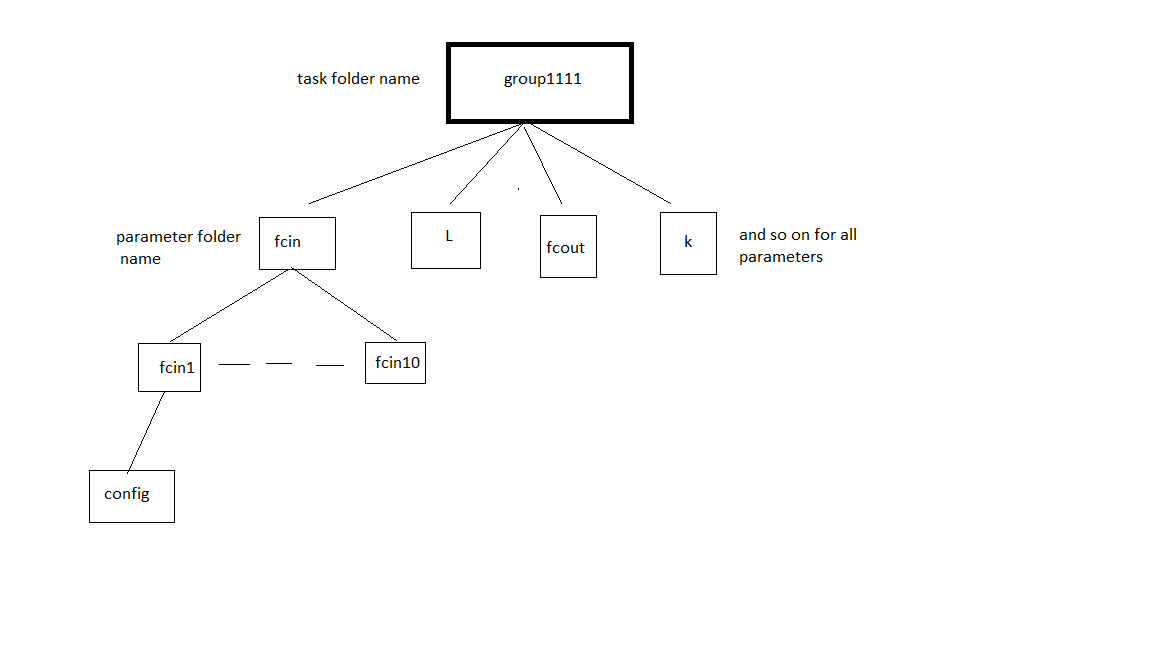
The script also accounts for the increment amount (last command line parameter above). The increment value can be int or float (sweeping by 0.1 for fcin).

1. Another script is included in hw4 package for extracting the 4 metrics (2 from vpr.out and 2 from <circuit\_file>.power), as well as the minimum or fixed channel width (found in vpr.out). The output is an excel file. Simply specify the folder name including both files for any given benchmark, and it will iterate for however many specified folders.

Usage: python read\_vpr\_power.py [directories]

**Organization of Data**

Create your task folders with your group number as the root folder. Maintain a hierarchy of your results when sweeping architecture parameters as shown for fcin below. Follow the same naming convention for all other architecture parameters.



**Submission instructions:**

**Part-1 Due March 16, @10pm on D2L (VTR\_Data\_Collection)**

1. Submit your entire task folder, create a tar file, for ex “group1.tar.gz”
   1. There should be only be one "run" folder for a particular task from which you have collected data. Please delete all other "run" folders
2. Fill in the excel file (data) given in the assignment folder. Please note the “wirelength”, “delay”, “logic power” and “routing power” worksheets.
   1. Rename data.xlsx as “data\_group#.xlsx”
   2. Submit along with your task folder

3) Submit **source data**, **excel file** and **this word document** with table for ex1 filled on d2L.

**Part-2 Due March 23, @ 10pm on D2L (VTR\_Data\_Analysis)**

1. Generate the average of each metric over the assigned benchmarks.
2. Plot the line trend of each metric (4 metrics) with respect to the sweeping range of each architecture parameter (7 parameters). An example chart is given below for your reference. You should use the same format and label the x-axis and y-axis appropriately for each chart. Note that each data point in this chart should be extracted by taking the average over your assigned benchmarks. Your results may show different pattern.
3. Discuss the trends observed in each chart. For example in the context of the given example chart, you should first summarize your **observations on relationship between K and average wirelength. You should then explain the root cause(s) for this relationship.**

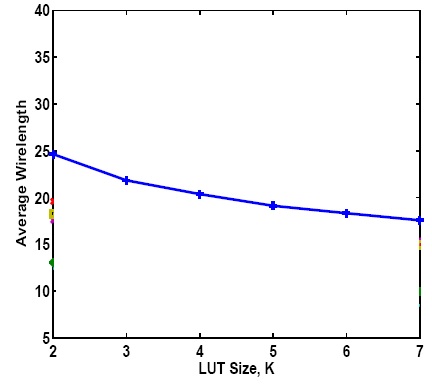
****

Figure 3: Average wirelength versus LUT size

1. Expose any correlation that may exist between performance metrics.

Hint: Please refer to the evaluations of Ahmed and Rose on area as a function of N and K. This will allow you to see the correlation between N and K with respect to wirelength. You should consider generating similar overlapped charts for other metrics using logic and routing architecture parameters.

**Analysis of Results**

**Include your analysis here and submit as .docx/pdf on D2L.**

**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 StratixII 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.

[6] J. Luu, J. Anderson, and J. Rose. Architecture Description and Packing for Logic Blocks with Hierarchy, Modes and Complex Interconnect, In International Symposium on Field Programmable Gate Arrays, Monterey, Ca., Feb. 2011.

[7] Jeffery Goeders and Steven Wilton. VersaPower: Power Estimation for FPGA Architectures. In International Conference on Field Programmable Technology, Vancouver, British Columbia, Oct. 2012.