Tutorial 1 - FPGA Architecture

Philip Leong

June 22, 2014

1. INTRODUCTION

Verilog to routing (VTR) [2] is a tool which allows the effect of different FPGA architectural choices to be quantified. In this laboratory exercise, we will study the effect of such choices on performance.

2. LABORATORY QUESTIONS

In answering these questions, marks will be awarded not only for correctness but also understandability and elegance of the solution. Brief instructions on using VTR are available from https://code.google.com/p/vtr-verilog-to-routing/wiki/TOC.

1. Familiarisation (40%). From the ‘‘`vtr_release/vtr_flow/tasks`’’ directory, enter the following command:

   ```
   ../scripts/run_vtr_task.pl basic_flow/
   ../scripts/parse_vtr_task.pl basic_flow/
   ```

   This will use the `k6_N10_memDepth16384_memData64_40nm_timing.xml` architecture and `ch_intrinsics.v` circuit. Results will be written to `basic_flow/run[#]/parse_results.txt`.

   Make a copy of the `basic_flow` directory and call it tut1. Modify the `config.txt` file within so that it runs on the *ch_intrinsics* and *sha* examples.

2. Impact of fracturable lookup table (LUT) (30%). The architecture file [1] describes a *fracturable LUT (FLUT)* based on the Altera Stratix IV. Create an architecture file for a configurable logic block (CLB) containing 20× 5-LUTs (which is similar to 10× FLUTs). Keep the timing of the registers and LUT the same as for the fracturable LUT. Compare the number of CLBs×N (which is roughly area) and critical path day (delay) between 5-LUTs and FLUTs over the geometric mean of the two benchmark results. Which gives more favourable results? Explain why.

   Information regarding how to define a 4-LUT in VPR is provided in Appendix A and the hard_fpu_arch_timing.xml file in the `arch` directory can be used directly in place of fracturable LUT. Note that some patience is required to create a working architecture file as error reporting is sometimes difficult to interpret. All outputs are logged to the `tut1/run[#]` directory.

3. Optimising delay× area (30%). Assuming K=5, how does changing N from 10-30 in steps of 2 $(10, 12, \ldots, 30)$ affect the performance? Make plots of delay×area vs N. Does the empirical relationship $I = \frac{K}{2}(N+1)$ hold for these examples?

4. Area model (bonus 20%). One shortcoming of this experiment was that a very crude area model was used. Develop an improved model which takes into account the additional area required when N is changed. Incorporate this into the architecture files and see how this affect the results.

A. Describing a CLB in VPR

This information is reproduced from the VPR manual[1].

A.1. Soft Logic Block

Figure 1 shows an example of a classical soft logic block found in academic FPGA literature. This block consists of N Basic Logic Elements (BLEs). The BLE inputs can come from either the inputs to the logic block or from other BLEs within the logic block via a full crossbar. The logic block in this figure has I general inputs, one clock input, and N outputs (where each output corresponds to a BLE). A BLE can implement three configurations: a K-input look-up table (K-LUT), a flip-flop, or a K-LUT followed by a flip-flop. The structure of a classical soft logic block results in a property known as logical equivalence for certain groupings of input/output pins. Logically equivalent pins means that connections to those pins can be swapped without changing functionality. For example, the input to AND gates are logically equivalent while the inputs to a 4-bit adders are not logically equivalent. In the case of a classical soft logic block, all input pins are logically equivalent (due to the fully populated crossbar) and all output pins are logically equivalent (because one can swap any two BLEs without changing functionality). Logical equivalence is important because it enables the CAD tools to make optimizations especially during routing. We describe a classical soft logic block with N = 10, I = 22, and K = 4 below.
First, a complex block pb_type called CLB is declared with appropriate input, output and clock ports. Logical equivalence is labelled at ports where it applies:

```
<pb_type name="clb">
  <input name="I" num_pins="22" equivalent="true"/>
  <output name="O" num_pins="10" equivalent="true"/>
  <clock name="clk" equivalent="false"/>
```

A CLB contains 10 BLEs. Each BLE has 4 inputs, one output, and one clock. A BLE block and its inputs and outputs are specified as follows:

```
  <pb_type name="ble" num_pb="10">
    <input name="in" num_pins="4"/>
```

[1]Available at http://www.eecg.utoronto.ca/vpr/utfal_ex1.html.

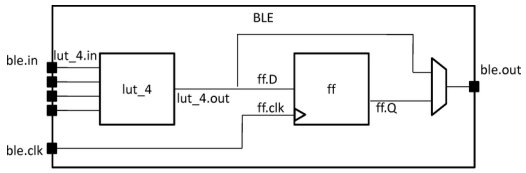

Figure 1: Model of a classical FPGA soft logic cluster.

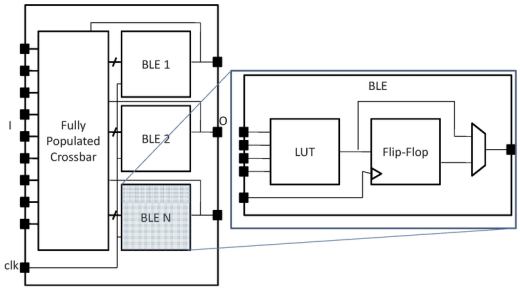

Figure 2: Internal names of BLE.

```
<output name="out" num_pins="1"/>
<clock name="clk"/>
```

A BLE consists of one LUT and one flip-flop (FF). Both of these are primitives. Recall that primitive physical blocks must have a blif_model attribute that matches with the model name in the BLIF input netlist. For the LUT, the model is ".names" in BLIF. For the FF, the model is ".latch" in BLIF. The class construct denotes that these are special (common) primitives. The primitives contained in the BLE are specified as:

```
<pb_type name="lut_4" blif_model=".names" num_pb="1" class="lut">
  <input name="in" num_pins="4" port_class="lut_in"/>
  <output name="out" num_pins="1" port_class="lut_out"/>
</pb_type>
<pb_type name="ff" blif_model=".latch" num_pb="1" class="flipflop">
  <input name="D" num_pins="1" port_class="D"/>
  <output name="Q" num_pins="1" port_class="Q"/>
  <clock name="clk" port_class="clock"/>
</pb_type>
```

Figure 2 shows the ports of the BLE with the input and output pin sets. The inputs to the LUT and flip-flop are direct connections. The multiplexer allows the BLE output to be either the LUT output or the flip-flop output. The code to specify the interconnect is:

```
  <interconnect>
    <direct input="lut_4.out" output="ff.D"/>
    <direct input="ble.in" output="lut_4.in"/>
    <mux input="ff.Q lut_4.out" output="ble.out"/>
    <direct input="ble.clk" output="ff.clk"/>
  </interconnect>
</pb_type>
```

The CLB interconnect is then modeled (see Figure 1). The inputs to the 10 BLEs (ble[9:0].in) can be connected to any of the CLB inputs (clb.I) or any of the BLE outputs (ble[9:0].out) by

using a full crossbar. The clock of the CLB is wired to multiple BLE clocks, and is modeled as a full crossbar. The outputs of the BLEs have direct wired connections to the outputs of the CLB and this is specified using one direct tag. The CLB interconnect specification is:

```
  <interconnect>
    <complete input="{clb.I ble[9:0].out}" output="ble[9:0].in"/>
    <complete input="clb.clk" output="ble[9:0].clk"/>
    <direct input="ble[9:0].out" output="clb.O"/>
  </interconnect>
```

Finally, we model the connectivity between the CLB and the general FPGA fabric (recall that a CLB communicates with other CLBs and I/Os using general-purpose interconnect). The ratio of tracks that a particular input/output pin of the CLB connects to is defined by fc_in/fc_out. In this example, a fc_in of 0.15 means that each input pin connects to 15

```
  <!-- Describe complex block relation with FPGA -->

  <fc_in type="frac">0.150000</fc_in>
  <fc_out type="frac">0.125000</fc_out>

  <pinlocations pattern="spread"/>
  <gridlocations>
    <loc type="fill" priority="1"/>
  </gridlocations>
</pb_type>
```

A.2. Basic Soft Logic Block Complete Example

```
<!--
Example of a classical FPGA soft logic block with
N = 10, K = 4, I = 22, O = 10
BLEs consisting of a single LUT followed by a flip-flop that can be bypassed
-->

<pb_type name="clb">
  <input name="I" num_pins="22" equivalent="true"/>
  <output name="O" num_pins="10" equivalent="true"/>
  <clock name="clk" equivalent="false"/>

  <pb_type name="ble" num_pb="10">
    <input name="in" num_pins="4"/>
    <output name="out" num_pins="1"/>
    <clock name="clk"/>

    <pb_type name="lut_4" blif_model=".names" num_pb="1" class="lut">
```

```
      <input name="in" num_pins="4" port_class="lut_in"/>
      <output name="out" num_pins="1" port_class="lut_out"/>
    </pb_type>
    <pb_type name="ff" blif_model=".latch" num_pb="1" class="flipflop">
      <input name="D" num_pins="1" port_class="D"/>
      <output name="Q" num_pins="1" port_class="Q"/>
      <clock name="clk" port_class="clock"/>
    </pb_type>

    <interconnect>
      <direct input="lut_4.out" output="ff.D"/>
      <direct input="ble.in" output="lut_4.in"/>
      <mux input="ff.Q lut_4.out" output="ble.out"/>
      <direct input="ble.clk" output="ff.clk"/>
    </interconnect>
  </pb_type>

  <interconnect>
    <complete input="{clb.I ble[9:0].out}" output="ble[9:0].in"/>
    <complete input="clb.clk" output="ble[9:0].clk"/>
    <direct input="ble[9:0].out" output="clb.O"/>
  </interconnect>

  <!-- Describe complex block relation with FPGA -->

  <fc_in type="frac">0.150000</fc_in>
  <fc_out type="frac">0.125000</fc_out>

  <pinlocations pattern="spread"/>
  <gridlocations>
    <loc type="fill" priority="1"/>
  </gridlocations>
</pb_type>
```

REFERENCES

[1] Jason Luu, Jason Helge Anderson, and Jonathan Rose. Architecture description and packing for logic blocks with hierarchy, modes and complex interconnect. In John Wawrzynek and Katherine Compton, editors, *FPGA*, pages 227–236. ACM, 2011.

[2] Jonathan Rose, Jason Luu, Chi Wai Yu, Opal Densmore, Jeffrey Goeders, Andrew Somerville, Kenneth B. Kent, Peter Jamieson, and Jason Helge Anderson. The vtr project: architecture and cad for fpgas from verilog to routing. In Katherine Compton and Brad L. Hutchings, editors, *FPGA*, pages 77–86. ACM, 2012.