1.0 Introduction

The following document serves as the lab report for assignment 4 of ECE 1756 (Fall 2021). This report is written by Sheng Zhao (1003273913).

Section 2.0 describes results for the routing wire length study. Section 3.0 describes results for the block to routing connectivity study. Section 4.0 describes results for the optimization study. All studies conducted in this assignment was done across 5 different seeds. The results presented are the geomean values of all 8 test benches across the 5 different seeds.

2.0 Routing Wire Length Study

Architecture (Length)	**Min Channel Width (Geomean)**	**Area per Tile (Geomean)**	**Crit. Path Delay (Geomean)**	**Area Delay Product**
1	50.65599	7284.84	7.47673	54466.78
2	50.05616	5227.42	6.09080	31839.16
4	58.38442	4654.36	5.89037	27415.91
8	91.82980	5457.34	6.86053	37440.23
16	192.58040	8357.36	8.99196	75149.08

Table 1: Geomean results for routing wire length study

Table 1 shows the results from the routing wire length study. As the length of the wires increase, there is a clear upwards trend in the minimum channel width, area and delay. This is due to the increasing inflexibility that comes with a longer wire, which as a result causes more wires to be needed in the channel to allow for all the necessary connections. This is also likely the reason why area per tile is so high for an architecture with a wire length of 16. In addition, while longer wires are more efficient at carrying signals across more logic blocks, they are slower for connections that are closer since the additional length translates to more resistance and capacitance in the path.

The minima seen in the area per tile and critical path delay metrics for length of 4 is likely due to the correct balance of routability and delay cost savings. As mentioned before, having wires of longer length reduces the delay since the additional delay caused by the length increase is lower than the delay cost of going through mutliple switch block muxes. However, the length are also not too long as to cause the channel width to blow up like in the case of length 16. As a result, we arrive at the lowest area-delay product at length 4.

3.0 Block to Routing Connectivity Study

CLB Fcin/Fcout	Min Channel Width (Geomean)	Area per Tile (Geomean)	Crit. Path Delay (Geomean)	Area Delay Product
0.15	58.38441	4654.36	5.89037	27415.91
0.5	52.84038	5980.22	6.28279	37572.45
1.0	52.13359	7648.77	7.00239	53559.70

Table 2: Geomean results for block and routing connectivity study (CLB)

With respect to Table 2, we can see that as Fc increases for the inputs and outputs of CLBs, the minimum channel width decreases. This is due to the increased level of connectivity between CLBs (and therefore the wider FPGA architecture). The higher level of routability meant that less wires are needed in the channel for signals to be carried to where they need to be. However, despite the decrease in channel width and the area gains as a result, it is not enough to overcome the increase in area required for the additional circuitry needed to implement these additional connections. This can be seen from the increase in area per tile required. The additional circuitry also causes increase load on the wires, leading to an increase in delay. Overall, these two trends lead to an increase in the area-delay product.

IO Fcin/Fcout	Min Channel Width (Geomean)	Area per Tile (Geomean)	Crit. Path Delay (Geomean)	Area Delay Product
0.15	58.38442	4654.36	5.89037	27415.91
0.5	58.13474	4742.71	5.87096	27844.25
1.0	58.20492	4847.89	5.87689	28490.55

Table 3: Geomean results for block and routing connectivity study (IO)

Moving on to the Fc of IO blocks, it would seem like the effects of changing the Fc is very minor compared to changing that of CLBs. This is likely due to the lower usage of IO blocks when compared to CLBs. All circuit logic will have to be implemented mainly on CLBs and thus routability of CLBs have a much greater impact on the overall performance.

CLB Fcin/Fcout	Min Channel Width (Geomean)	Area per Tile (Geomean)	Crit. Path Delay (Geomean)	Area Delay Product
0.15	130.70798	9535.70	7.39891	70553.71
0.5	100.377302	10377.35	8.09355	83989.60
1.0	81.09910	11105.02	8.80138	97739.48

Table 4: Geomean results for block and routing connectivity study with no local crossbar

Without the full local crossbar, additional restrictions will be placed on the routing. As a result, the original Fc of 0.15 is now too limiting and the minimum channel width has to be increased to make up for the decreased flexibility within the CLBs. This is supported by the decreasing trend of minimum channel width as Fc increases. Area per tile sees a upward trend with Fc value due to the increasing number of muxes and circuitry required to implement the Fc, delay also increases as a result of the increased load on the wires.

4.0 Optimization Study

To arrive at the most optimal architecture, multiple smaller studies were conducted.

4.1 Wire properties

Properties R/C	Min Channel Width (Geomean)	Area per Tile (Geomean)	Crit. Path Delay (Geomean)	Area Delay Product
101/22.5e-15 (Base)	58.38442	4654.36	5.89037	27415.91
41/27e-15 (0.4R/1.2C)	58.46167	4663.83	5.80307	27064.54
202/13.5e-15 (2R/0.6C)	58.17173	4634.35	5.65714	26217.17

Table 5: Geomean results for wire properties study

The length 4 architecture is used as the base and the changes are made on top of that. With this, tests were run and the geomean results are presented in Table 5. Based on the table, the decrease in capacitance by 40% while having double the resistance seems to be the most effective at lowering the area-delay product, suggesting that capacitance pays a greater role in reducing the delay.

Length A / Length B	Min Channel Width (Geomean)	Area per Tile (Geomean)	Crit. Path Delay (Geomean)	Area Delay Product
2 / 4	52.26888	4774.56	5.75658	27485.11
2 / 8	55.95871	4732.81	6.08804	28813.53
4 / 8	68.32751	4783.99	6.03582	28875.33

Table 6: Geomean results for duo length study

Next, a study was conducted on having two different lengths within the architecture. These wires connect to all the switches and CLBs present along their path and the results suggests that having duo lengths does not seem to improve the area-delay product.

```
        it passes. -->
    <segment freq="1.000000" length="2" type
      <mux name="routing_switch"/>
      <sb type="pattern">1 0 1</sb>
      <cb type="pattern">1 1</cb>
    </segment>
    <segment freq="1.000000" length="4" type
      <mux name="routing_switch"/>
      <sb type="pattern">1 0 1 0 1</sb>
      <cb type="pattern">1 0 1 1</cb>
    </segment>
  </segmentlist>
```

Figure 1: Alternating pattern

```
        it passes. -->
    <segment freq="1.000000" length="2" type=
      <mux name="routing_switch"/>
      <sb type="pattern">1 0 1</sb>
      <cb type="pattern">1 1</cb>
    </segment>
    <segment freq="1.000000" length="4" type=
      <mux name="routing_switch"/>
      <sb type="pattern">1 1 0 1 1</sb>
      <cb type="pattern">1 1 1 1</cb>
    </segment>
  </segmentlist>
```

Figure 2: "Leapfrogging" pattern

With the most efficient duo length architecture, the switch block and connection block patterns are varied to see if doing so could further improve the area-delay product. The motivation for doing so is due to the fact that there are two different wire lengths and having the shorter length 2 wires focus on connecting the local CLBs while the length 4 is to be used for larger distances might improve the efficiency of routing. To do so, two different patterns are considered, the alternating pattern (Fig. 1) and "leapfrogging" pattern (Fig. 2).

SB/CB Pattern	**Min Channel Width (Geomean)**	**Area per Tile (Geomean)**	**Crit. Path Delay (Geomean)**	**Area Delay Product**
Full	52.26888	4774.56	5.75658	27485.11
Alternating	58.37674	4661.02	5.84631	27249.78
"Leapfrogging"	54.19312	4566.43	5.80761	26520.04

Table 7: Geomean results for 2/4 length, SB/CB pattern study

With those patterns, the results in Table 7 are obtained, showing that the "leapfrogging" pattern can yield tangible improvements on the area-delay product. The reasons for which is likely as mentioned in the previous paragraph.

Properties R/C	**Min Channel Width (Geomean)**	**Area per Tile (Geomean)**	**Crit. Path Delay (Geomean)**	**Area Delay Product**
101/22.5e-15 (Base)	54.19312	4566.43	5.80761	26520.04
41/27e-15 (0.4R/1.2C)	54.00696	4551.12	5.72004	26032.62
202/13.5e-15 (2R/0.6C)	54.38988	4575.81	5.61855	25709.40

Table 8: Geomean results for wire properties study with 2/4 len and "leapfrogging" pattern

Combining the most optimal "leapfrogging" pattern with better electrical properties (Table **??**), we arrive at a final area-delay product of 25709.40. The architecture file can be found in Appendix A.

5.0 Future Work

Due to how the different factors that affect the efficiency of the architecture (Fc, SB/CB pattern, wire lengths) play into each another, it is possible that more efficient architecture still exists since not all combinations are explored in the study. Splitting the wires across different metal layers was also not experimented with and could also be a potential source of improvement since having infrequent, longer wires on higher layers could benefit the design with the lower resistance wires for longer distance signals.

5.0 Appendix

Appendix A: .xml file for final architecture

```
<!--
  Architecture with no fracturable LUTs

  - 40 nm technology
  - General purpose logic block:
    K = 6, N = 10
  - Routing architecture: L = 4, fc_in = 0.15, Fc_out = 0.15
  - Unidirectional (mux-based) routing

  Details on Modelling:

  Based on flagship k6_frac_N10_mem32K_40nm.xml architecture.  This
↪  architecture has no fracturable LUTs nor any heterogeneous blocks so it
↪  is simpler.
  The delays and areas are based on a mix of values from commercial 40 nm
  FPGAs with a comparable architecture and 40 nm interconnect and
  transistor models.


  Authors: Vaughn Betz, Jason Luu, Jeff Goeders
--><architecture>

  <!--
       ODIN II specific config begins
       This part of the architecture file describes the "primitives"
       that exist in a device to the synthesis tool used to "elaborate"
       verilog into these primitives (which is called ODIN-II).
       Basic LUTs, I/Os and FFs are built into the language used by this
       flow (blif keywords .names, .input, .output and .latch), so they
       don't have to be described here.

       For this lab you are also given the benchmark netlists after
       synthesis is complete (in the blif directory), so you don't need
       to run ODIN II.
  -->
  <models>
  </models>
  <!-- ODIN II specific config ends -->
```

```
  <!-- Physical descriptions begin -->
  <layout>
    <auto_layout aspect_ratio="1.0">
        <!--Perimeter of 'io' blocks with 'EMPTY' blocks at corners-->
        <perimeter type="io" priority="100"/>
        <corners type="EMPTY" priority="101"/>
        <!--Fill with 'clb'-->
        <fill type="clb" priority="10"/>
        </auto_layout>
</layout>

  <device>
    <!-- Some area and timing parameters -->
    <sizing R_minW_nmos="8926" R_minW_pmos="16067"/>
    <!-- The grid_logic_tile_area below will be used for all blocks that do
    ↪  not explicitly set their own (non-routing)
              area; set to 0 since we explicitly set the area of all
↪  blocks currently in this architecture file.
         -->
        <area grid_logic_tile_area="0"/>

    <!-- All routing channels have the same width -->
    <chan_width_distr>
      <x distr="uniform" peak="1.000000"/>
      <y distr="uniform" peak="1.000000"/>
    </chan_width_distr>

    <!-- Define the switch block pattern (pattern of switches between
    ↪  inter-tile routing wires) -->
    <switch_block type="wilton" fs="3"/>

    <!-- Set which switch to use for input connection blocks. Only affects
    ↪  timing and area, not connectivity -->
    <connection_block input_switch_name="ipin_cblock"/>
  </device>

  <switchlist>
          <!-- VB: the mux_trans_size and buf_size data below is in minimum
          ↪  width transistor *areas*, assuming the purple
               book area formula. This means the mux transistors are about
↪  5x minimum drive strength.
```

```
                The first stage of the buffer is 3x min drive strength to
↪  be reasonable given the large
                mux transistors, and this gives a reasonable stage ratio of
↪  a bit over 5x to the second stage.  -->
    <switch type="mux" name="routing_switch" R="551" Cin=".77e-15"
    ↪  Cout="4e-15" Tdel="58e-12" mux_trans_size="2.630740"
    ↪  buf_size="27.645901"/>
  <!--switch ipin_cblock resistance set to yeild for 4x minimum drive
  ↪  strength buffer-->
        <switch type="mux" name="ipin_cblock" R="2231.5" Cout="0."
        ↪  Cin="1.47e-15" Tdel="7.247000e-11" mux_trans_size="1.222260"
        ↪  buf_size="auto"/>
        </switchlist>

  <segmentlist>
    <!-- VB & JL: using ITRS metal stack data, 96 nm half pitch wires,
    ↪  which are intermediate metal width/space.  Wires of this pitch will
    ↪  fit over a 90 nm
    high logic tile (which is about the height of a Stratix IV logic
↪   tile).
    I'm using a tile length of 90 nm, corresponding to the length of a
↪   Stratix IV tile if it were square.
    length below is in units of logic blocks, and Rmetal and Cmetal are
    per logic block passed.   -->

    <!-- Currently only one type of routing wire, which
         is of length 4 and has switches to every connection
         box (4 of them) and switch box (5 of them)
         it passes. -->
    <segment freq="1.000000" length="2" type="unidir" Rmetal="202"
    ↪  Cmetal="13.5e-15">
      <mux name="routing_switch"/>
      <sb type="pattern">1 0 1</sb>
      <cb type="pattern">1 1</cb>
    </segment>
    <segment freq="1.000000" length="4" type="unidir" Rmetal="202"
    ↪  Cmetal="13.5e-15">
      <mux name="routing_switch"/>
      <sb type="pattern">1 1 0 1 1</sb>
      <cb type="pattern">1 1 1 1</cb>
    </segment>
  </segmentlist>
```

```
  <complexblocklist>

    <!-- Define I/O pads begin -->
    <!-- Capacity is a unique property of I/Os, it is the maximum number of
    ↪  I/Os that can be placed at the same (X,Y) location on the FPGA -->
          <!-- Not sure of the area of an I/O (varies widely), and it's not
          ↪  relevant to the design of the FPGA core, so we're setting it
          ↪  to 0. -->
    <pb_type name="io" capacity="8" area="0">
      <input name="outpad" num_pins="1"/>
      <output name="inpad" num_pins="1"/>
      <clock name="clock" num_pins="1"/>

      <!-- IOs can operate as either inputs or outputs.
             The delay below are to and from registers in the I/O (and
↪  generally I/Os are registered today).
            -->
      <mode name="inpad">
        <pb_type name="inpad" blif_model=".input" num_pb="1">
          <output name="inpad" num_pins="1"/>
        </pb_type>
        <interconnect>
          <direct name="inpad" input="inpad.inpad" output="io.inpad">
            <delay_constant max="4.243e-11" in_port="inpad.inpad"
            ↪  out_port="io.inpad"/>
          </direct>
        </interconnect>

      </mode>
      <mode name="outpad">
        <pb_type name="outpad" blif_model=".output" num_pb="1">
          <input name="outpad" num_pins="1"/>
        </pb_type>
        <interconnect>
          <direct name="outpad" input="io.outpad" output="outpad.outpad">
            <delay_constant max="1.394e-11" in_port="io.outpad"
            ↪  out_port="outpad.outpad"/>
          </direct>
        </interconnect>
      </mode>
```

```
      <!-- Every input pin is driven by 15% of the tracks in a channel,
      ↪  every output pin drives 15% of the tracks in a channel -->
      <fc in_type="frac" in_val="0.15" out_type="frac" out_val="0.15"/>

      <!-- IOs go on the periphery of the FPGA in this
           architecture. Since I don't want to define four
           different physical I/Os for the left, right, top,
           and bottom sides just say each pin of the I/O
           block is accessible from all four sides so we can
           reach routing channels on some side of the block
           no matter which side of the chip we're on.
        -->
      <pinlocations pattern="custom">
        <loc side="left">io.outpad io.inpad io.clock</loc>
        <loc side="top">io.outpad io.inpad io.clock</loc>
        <loc side="right">io.outpad io.inpad io.clock</loc>
        <loc side="bottom">io.outpad io.inpad io.clock</loc>
      </pinlocations>

      <!-- Place I/Os on the sides of the FPGA -->
      <!-- Not modeling I/O power for now -->
      <power method="ignore"/>
    </pb_type>
    <!-- Define I/O pads ends -->

    <!-- Define general purpose logic block (CLB) begin -->
          <!-- Area below is for everything inside the
                logic block (LUTs, FFs, intra-cluster
                routing).
           -->
    <pb_type name="clb" area="15000">
      <!-- We have a full crossbar between the cluster inputs and the
           LUT inputs, so the router can route to *any* input or from
           *any* output on the logic block.  Hence mark the logic block
           inputs as fully logically equivalent (swappable by the router)
    ↪  and also the
           logic block outputs as logically equivalent, which means
           they can also be swapped by the router.
           -->
      <input name="I" num_pins="33" equivalent="full"/>
      <output name="O" num_pins="10" equivalent="full"/>
      <clock name="clk" num_pins="1"/>
```

```
      <!-- The logic block pins can connect to 15% of the wires passing by
      ↪
           in the adjacent channel, and the pins are evenly spread
           across all 4 sides of the logic block. Each pin appears on one
↪  side. -->

      <fc in_type="frac" in_val="0.15" out_type="frac" out_val="0.15"/>
      <pinlocations pattern="spread"/>


      <!-- Describe basic logic element.
             Each basic logic element has a 6-LUT that can be optionally
↪  registered
        -->
      <pb_type name="fle" num_pb="10">
        <input name="in" num_pins="6"/>
        <output name="out" num_pins="1"/>
        <clock name="clk" num_pins="1"/>
        <!-- 6-LUT mode definition begin -->
        <mode name="n1_lut6">
          <!-- Define 6-LUT mode -->
          <pb_type name="ble6" num_pb="1">
            <input name="in" num_pins="6"/>
            <output name="out" num_pins="1"/>
            <clock name="clk" num_pins="1"/>

            <!-- Define LUT -->
            <pb_type name="lut6" blif_model=".names" num_pb="1" class="lut">
              <input name="in" num_pins="6" port_class="lut_in"/>
              <output name="out" num_pins="1" port_class="lut_out"/>
              <!-- LUT timing using delay matrix -->
              <!-- Real LUTs have different delays per input
                   but since VPR's router does not exploit
                   that by changing which signal goes to which
                   LUT input we'll make all the LUT
                   delays the same to reduce CAD noise.
                  -->
              <delay_matrix type="max" in_port="lut6.in"
              ↪  out_port="lut6.out">
                200e-12
                200e-12
```

```
            200e-12
            200e-12
            200e-12
            200e-12
          </delay_matrix>
        </pb_type>

        <!-- Define flip-flop -->
        <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" num_pins="1" port_class="clock"/>
          <T_setup value="66e-12" port="ff.D" clock="clk"/>
          <T_clock_to_Q max="124e-12" port="ff.Q" clock="clk"/>
        </pb_type>

        <!-- many lines below to describe the interconnect
             wires, muxes and crossbars inside a cluster.
          -->
        <interconnect>
          <direct name="direct1" input="ble6.in" output="lut6[0:0].in"/>
          <direct name="direct2" input="lut6.out" output="ff.D">
            <!-- Advanced user option that tells CAD tool to find
            ↪  LUT+FF pairs in netlist
                 and make sure it packs them together -->
            <pack_pattern name="ble6" in_port="lut6.out"
            ↪  out_port="ff.D"/>
          </direct>
          <direct name="direct3" input="ble6.clk" output="ff.clk"/>
          <mux name="mux1" input="ff.Q lut6.out" output="ble6.out">
            <!-- LUT to output is faster than FF to output on a Stratix
            ↪  IV -->
            <delay_constant max="25e-12" in_port="lut6.out"
            ↪  out_port="ble6.out"/>
            <delay_constant max="45e-12" in_port="ff.Q"
            ↪  out_port="ble6.out"/>
          </mux>
        </interconnect>
      </pb_type>
      <interconnect>
        <direct name="direct1" input="fle.in" output="ble6.in"/>
```

```
            <direct name="direct2" input="ble6.out" output="fle.out[0:0]"/>
            <direct name="direct3" input="fle.clk" output="ble6.clk"/>
          </interconnect>
        </mode>
        <!-- 6-LUT mode definition end -->
      </pb_type>
      <interconnect>
        <!-- We use a full crossbar to get logical equivalence at inputs of
        ↪  CLB
                    The delays below come from Stratix IV. the delay
↪  through a connection block
                    input mux + the crossbar in Stratix IV is 167 ps. We
↪  already have a 72 ps
                    delay on the connection block input mux (modeled by
↪  Ian Kuon), so the remaining
                    delay within the crossbar is 95 ps.
                    The delays of cluster feedbacks in Stratix IV is 100
↪  ps, when driven by a LUT.
                    Since all our outputs LUT outputs go to a BLE output,
↪  and have a delay of
                    25 ps to do so, we subtract 25 ps from the 100 ps
↪  delay of a feedback
                    to get the part that should be marked on the
↪  crossbar.          -->
        <complete name="crossbar" input="clb.I fle[9:0].out"
        ↪  output="fle[9:0].in">
          <delay_constant max="95e-12" in_port="clb.I"
          ↪  out_port="fle[9:0].in"/>
          <delay_constant max="75e-12" in_port="fle[9:0].out"
          ↪  out_port="fle[9:0].in"/>
        </complete>
        <complete name="clks" input="clb.clk" output="fle[9:0].clk">
        </complete>

        <!-- The BLE outputs are directly connected to the
             CLB (cluster) outputs.
          -->
        <direct name="clbouts1" input="fle[9:0].out" output="clb.O"/>
      </interconnect>
```

```
      <!-- Place this general purpose logic block in any unspecified column
      ↪  -->
      </pb_type>
    <!-- Define general purpose logic block (CLB) ends -->

  </complexblocklist>

  <!-- data below gives extra information about the logic
       block and clock network electrical properties needed
       for power analysis.
   -->
  <power>
    <local_interconnect C_wire="2.5e-10"/>
    <mux_transistor_size mux_transistor_size="3"/>
    <FF_size FF_size="4"/>
    <LUT_transistor_size LUT_transistor_size="4"/>
  </power>
  <clocks>
    <clock buffer_size="auto" C_wire="2.5e-10"/>
  </clocks>
</architecture>
```