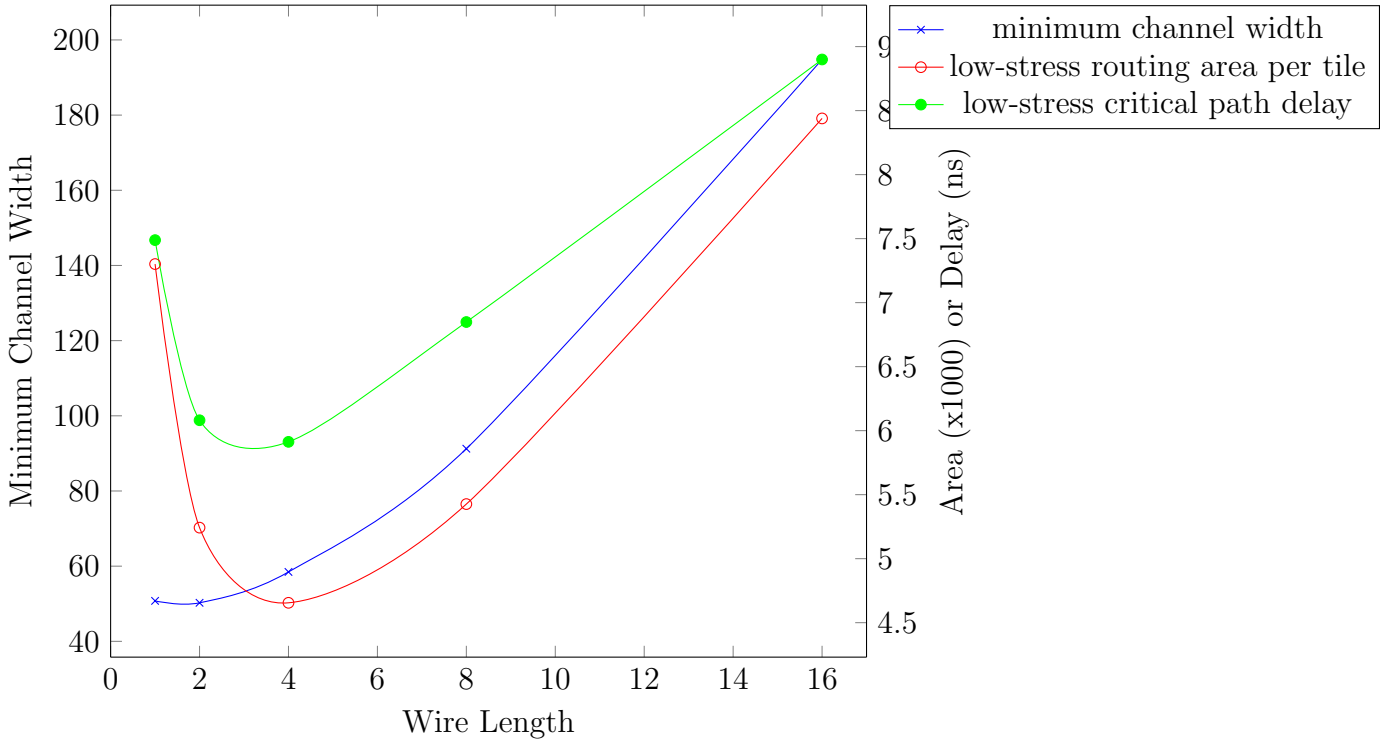
Lichen Liu
1001721498

Assignment 4

ECE 1756H
December 8, 2022

1 Routing Wire Length Study

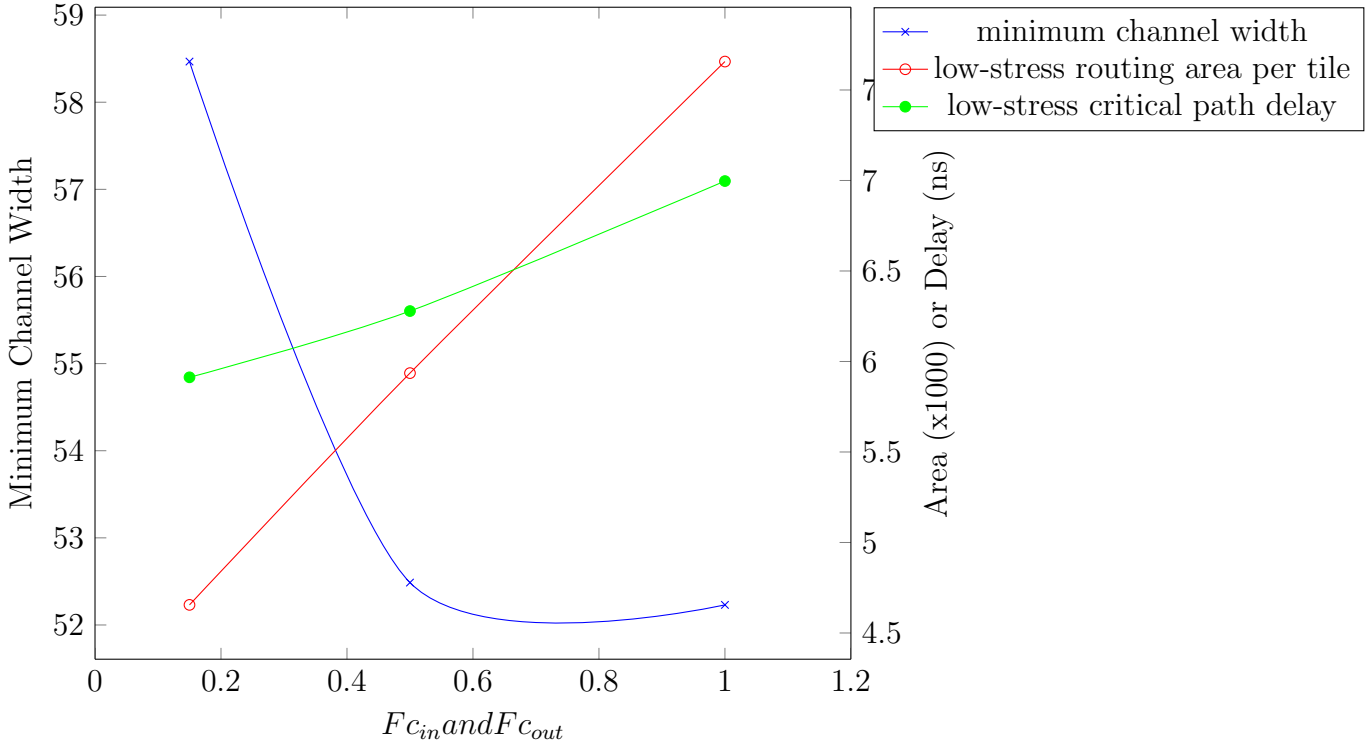


The wire length is adjusted by modifying the segment length, while preserving the routing switch patterns so that there are switches at each switch box and connection box. The graph for minimum channel width, low-stress routing area per tile, and low-stress critical path delay all have a u-shape pattern. The u-shape of minimum channel width is not very obvious, and the minimum point is at wire length 2, then it increases as wire length increases. This is because the wires still occupy the channel width even after the signals are already tapped off at earlier switches. A single wire can only be driven by a single source at a time, so a new wire has to be used to fill the demand for a new wire. The longer the wire, the more routing channel width will be wasting. The minimum point for low-stress routing area per tile and critical path delay is at wire length 4. While at shorter wire length, delay and area is worse because of the frequent switches; while at longer wire length, the increase in delay is caused by the electrical characteristics of long wires and the increase in area is due to the increase in channel width.

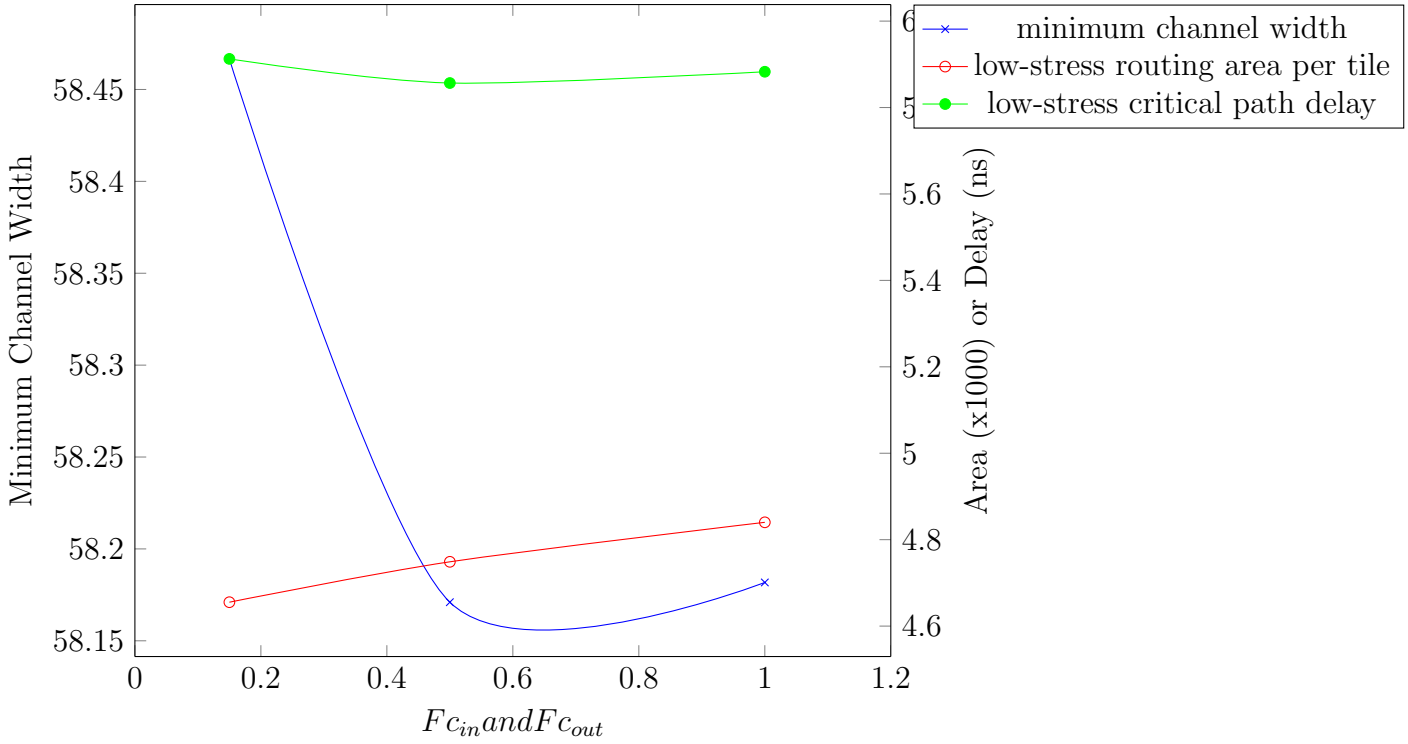
According to the area-delay metrics, wire length 4 is the best choice.

2 Block to Routing Connectivity Study

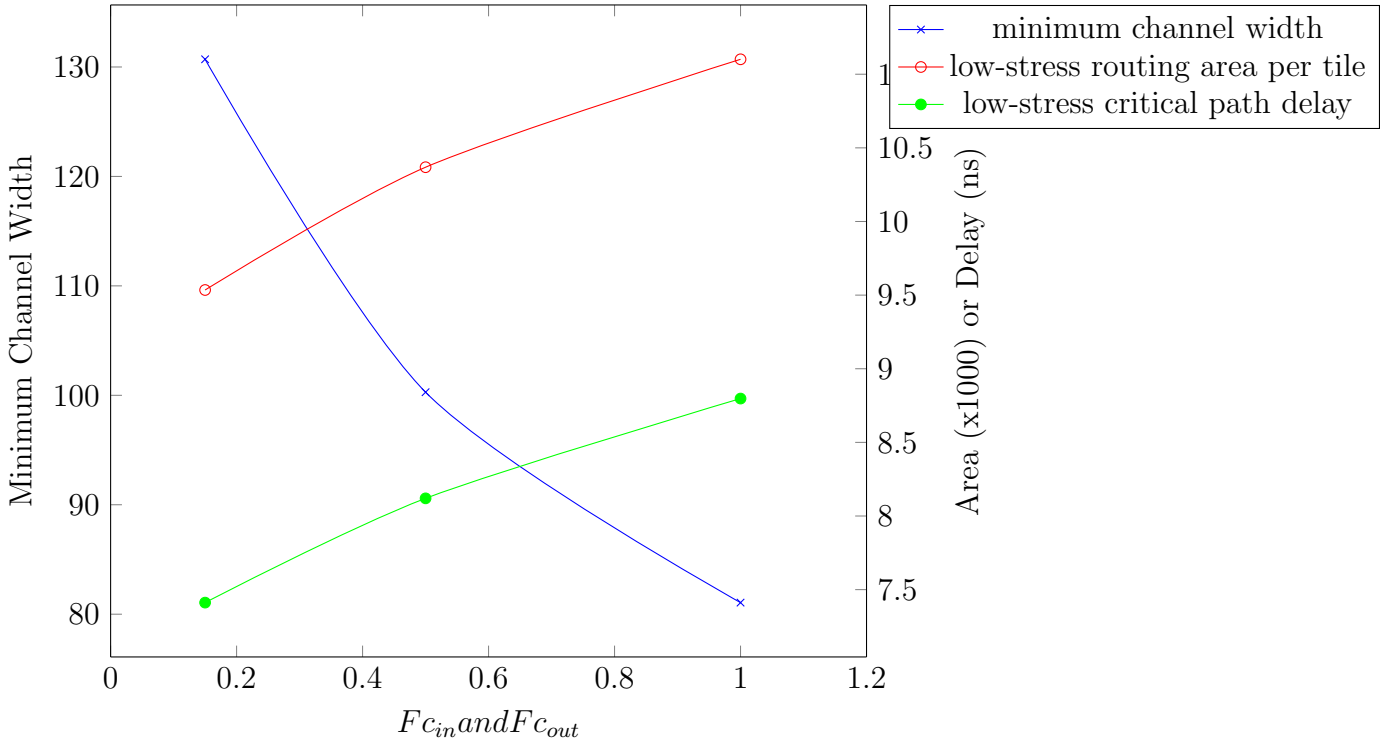
2.1 Varying $F_{c_{in}}$ and $F_{c_{out}}$ for CLBs



2.2 Varying $F_{c_{in}}$ and $F_{c_{out}}$ for IOs



2.3 Varying $F_{c_{in}}$ and $F_{c_{out}}$ for CLBs with No Logical Equivalence



2.4 Discussions

The minimum channel width decreases as F_c increases in general. In all three cases, there is an obvious drop in minimum channel width when F_c is increased to 0.5 from 0.15. This shows that an inflexible logic pin to routing wire topology can greatly increase the minimum channel width. However, F_c is less important to the minimum channel width when it approaches 1 for the logical equivalence cases. On the other hand, it is still very important for the case when CLB pins are not logically equivalent, that an increase of F_c to 1.0 further reduces the minimum channel width by a lot. This shows that a flexible routing topology only benefits the minimum channel width to a certain degree, and the marginal benefit gradually diminishes as the topology becomes more flexible.

The low-stress critical path delay shows an increase when F_c is increased for CLBs, which can be explained by the electrical effect caused by the presence of more switches in the routing fabric. On the other hand, the trend for low-stress critical path delay is pretty flat when F_c is increased for IOs. This might be due to the fact that the critical path of the benchmarking circuits are not IO-related.

The low-stress routing area per tile shows a very similar pattern as that of the low-stress critical path delay. The area increases as F_c increases simply due to the increase in number

of switches in the connection box. This increase is not very obvious for the IO case possibly due to the fact that there are much fewer IO blocks than CLB blocks.

In all three scenarios, an F_c value of 0.15 gives the best area-delay metric.

3 Optimization Study

3.1 Experiments

			Low-stress		
ID	**Description**	**Minimum Channel Width**	**Routing Area Per Tile**	**Critical Path Delay (ns)**	**Area-Delay**
0	**Best from Previous**	58.4666	4655.2961	5.9126	27524.7706
1	**0 + IO F_c to 0.1**	58.3500	4609.5524	5.9052	27220.4599
2	**1 + Convert 15% to upper metal layer**	58.1854	4605.0165	5.7453	26457.2846
3	**2 + Convert 25% to length 2 wire**	55.7238	4785.3687	5.7220	27382.07816
4	**3 + Reduce length 4 wire sb pattern**	58.7281	4570.5540	5.6600	25869.1904
5	**4 + Set switch block type to subset**	59.0016	4549.3782	5.6815	25847.4652
6	**4 + Set switch block type to universal**	*FAIL*	*FAIL*	*FAIL*	*FAIL*
7	**4 + 40% R_{metal} 1.2x C_{metal} for length 4 wire**	58.5280	4558.1312	5.6475	25741.9911
8	**4 + 2x R_{metal} 60% C_{metal} for length 4 wire**	58.1455	4536.0576	5.6685	25712.4674
9	**4 + 2x R_{metal} 60% C_{metal} for all wires, remove top metal layer**	56.9904	4416.1154	5.6465	24935.4690

The table shows the list of architecture exploration experiments being performed and their corresponding QoR results, and a lot more results are omitted from the table. Different architectures are compared by taking the area-delay metric from the 5-seeded 8-circuits benchmark runs.

3.2 Discussions

Comparing to the base architecture which has an area-delay of 27524, the best architecture has only 24935. The architecture reduces the F_c from 0.15 to 0.1 for IO blocks. The reduction is aimed at reducing the number of switches required in the connection box for IO blocks. Reducing the connectivity does not seem to affect the minimum channel width probably because IO blocks are less congested with signals. The second architecture change is by redistributing 25% of length 4 wires to length 2 wires, while also reducing the switch box pattern for length 4 wires to "1 0 1 0 1". The area saving in switches used by the switch boxes outweighs the additional routing wires needed for detouring, which can also be relieved by the new shorter length 2 wires. The last architecture change is related to the electrical parameter, such that C_{metal} is 40% less at the cost of a doubled R_{metal} (by making the wire narrower and using the extra room to increase the spacing between wires). This boosted another few hundreds of area-delay metric. This electrical parameter was determined based from experiment sweeps. However, it is not clear why this actually boosts the area-delay.

4 Appendix

```
<!--
  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>

  <!--
```

```
22        ODIN II specific config begins
23        This part of the architecture file describes the "primitives"
24        that exist in a device to the synthesis tool used to "elaborate"
25        verilog into these primitives (which is called ODIN-II).
26        Basic LUTs, I/Os and FFs are built into the language used by this
27        flow (blif keywords .names, .input, .output and .latch), so they
28        don't have to be described here.
29
30        For this lab you are also given the benchmark netlists after
31        synthesis is complete (in the blif directory), so you don't need
32        to run ODIN II.
33    -->
34    <models>
35    </models>
36    <!-- ODIN II specific config ends -->
37
38    <!-- Physical descriptions begin -->
39    <layout>
40      <auto_layout aspect_ratio="1.0">
41          <!--Perimeter of 'io' blocks with 'EMPTY' blocks at corners-->
42          <perimeter type="io" priority="100"/>
43          <corners type="EMPTY" priority="101"/>
44          <!--Fill with 'clb'-->
45          <fill type="clb" priority="10"/>
46          </auto_layout>
47  </layout>
48
49    <device>
50      <!-- Some area and timing parameters -->
51      <sizing R_minW_nmos="8926" R_minW_pmos="16067"/>
52      <!-- The grid_logic_tile_area below will be used for all blocks that do
        ↪  not explicitly set their own (non-routing)
53                area; set to 0 since we explicitly set the area of all
    ↪  blocks currently in this architecture file.
54            -->
55          <area grid_logic_tile_area="0"/>
56
57      <!-- All routing channels have the same width -->
58      <chan_width_distr>
59        <x distr="uniform" peak="1.000000"/>
60        <y distr="uniform" peak="1.000000"/>
61      </chan_width_distr>
```

```
62
63      <!-- Define the switch block pattern (pattern of switches between
        ↪  inter-tile routing wires) -->
64      <switch_block type="wilton" fs="3"/>
65
66      <!-- Set which switch to use for input connection blocks. Only affects
        ↪  timing and area, not connectivity -->
67      <connection_block input_switch_name="ipin_cblock"/>
68    </device>
69
70    <switchlist>
71            <!-- VB: the mux_trans_size and buf_size data below is in minimum
              ↪  width transistor *areas*, assuming the purple
72                 book area formula. This means the mux transistors are about
    ↪  5x minimum drive strength.
73                 The first stage of the buffer is 3x min drive strength to
    ↪  be reasonable given the large
74                 mux transistors, and this gives a reasonable stage ratio of
    ↪  a bit over 5x to the second stage.  -->
75      <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"/>
76    <!--switch ipin_cblock resistance set to yeild for 4x minimum drive
      ↪  strength buffer-->
77          <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"/>
78          </switchlist>
79
80    <segmentlist>
81      <!-- 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
82      high logic tile (which is about the height of a Stratix IV logic
    ↪  tile).
83      I'm using a tile length of 90 nm, corresponding to the length of a
    ↪  Stratix IV tile if it were square.
84      length below is in units of logic blocks, and Rmetal and Cmetal are
85      per logic block passed.   -->
86
87      <!-- Currently only one type of routing wire, which
88           is of length 4 and has switches to every connection
```

```
89          box (4 of them) and switch box (5 of them)
90          it passes. -->
91
92      <!-- Base: Rmetal="101" Cmetal="22.5e-15" -->
93
94      <!-- Wires with double the Rmetal (2x), but 40% (0.6x) less Cmetal (by
        ↪  making the wire narrower and using the extra room to increase the
        ↪  spacing between wires) -->
95      <segment freq="0.75" length="4" type="unidir" Rmetal="202"
        ↪  Cmetal="1.35e-14">
96        <mux name="routing_switch"/>
97        <sb type="pattern">1 0 1 0 1</sb>
98        <cb type="pattern">1 1 1 1</cb>
99      </segment>
100
101     <!-- Wires with double the Rmetal (2x), but 40% (0.6x) less Cmetal (by
        ↪  making the wire narrower and using the extra room to increase the
        ↪  spacing between wires) -->
102     <segment freq="0.25" length="2" type="unidir" Rmetal="202"
        ↪  Cmetal="1.35e-14">
103       <mux name="routing_switch"/>
104       <sb type="pattern">1 1 1</sb>
105       <cb type="pattern">1 1</cb>
106     </segment>
107   </segmentlist>
108
109   <complexblocklist>
110
111     <!-- Define I/O pads begin -->
112     <!-- 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 -->
113           <!-- 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. -->
114     <pb_type name="io" capacity="8" area="0">
115       <input name="outpad" num_pins="1"/>
116       <output name="inpad" num_pins="1"/>
117       <clock name="clock" num_pins="1"/>
118
119       <!-- IOs can operate as either inputs or outputs.
120              The delay below are to and from registers in the I/O (and
    ↪  generally I/Os are registered today).
```

```
121             -->
122         <mode name="inpad">
123           <pb_type name="inpad" blif_model=".input" num_pb="1">
124             <output name="inpad" num_pins="1"/>
125           </pb_type>
126           <interconnect>
127             <direct name="inpad" input="inpad.inpad" output="io.inpad">
128               <delay_constant max="4.243e-11" in_port="inpad.inpad"
                  ↪ out_port="io.inpad"/>
129             </direct>
130           </interconnect>
131
132         </mode>
133         <mode name="outpad">
134           <pb_type name="outpad" blif_model=".output" num_pb="1">
135             <input name="outpad" num_pins="1"/>
136           </pb_type>
137           <interconnect>
138             <direct name="outpad" input="io.outpad" output="outpad.outpad">
139               <delay_constant max="1.394e-11" in_port="io.outpad"
                  ↪ out_port="outpad.outpad"/>
140             </direct>
141           </interconnect>
142         </mode>
143
144         <!-- Every input pin is driven by 15% of the tracks in a channel,
            ↪ every output pin drives 15% of the tracks in a channel -->
145         <fc in_type="frac" in_val="0.1" out_type="frac" out_val="0.1"/>
146
147         <!-- IOs go on the periphery of the FPGA in this
148              architecture. Since I don't want to define four
149              different physical I/Os for the left, right, top,
150              and bottom sides just say each pin of the I/O
151              block is accessible from all four sides so we can
152              reach routing channels on some side of the block
153              no matter which side of the chip we're on.
154           -->
155         <pinlocations pattern="custom">
156           <loc side="left">io.outpad io.inpad io.clock</loc>
157           <loc side="top">io.outpad io.inpad io.clock</loc>
158           <loc side="right">io.outpad io.inpad io.clock</loc>
159           <loc side="bottom">io.outpad io.inpad io.clock</loc>
```

```
160       </pinlocations>
161
162       <!-- Place I/Os on the sides of the FPGA -->
163       <!-- Not modeling I/O power for now -->
164       <power method="ignore"/>
165     </pb_type>
166     <!-- Define I/O pads ends -->
167
168     <!-- Define general purpose logic block (CLB) begin -->
169           <!-- Area below is for everything inside the
170                 logic block (LUTs, FFs, intra-cluster
171                 routing).
172            -->
173     <pb_type name="clb" area="15000">
174       <!-- We have a full crossbar between the cluster inputs and the
175            LUT inputs, so the router can route to *any* input or from
176            *any* output on the logic block.  Hence mark the logic block
177            inputs as fully logically equivalent (swappable by the router)
    ↪  and also the
178            logic block outputs as logically equivalent, which means
179            they can also be swapped by the router.
180            -->
181       <input name="I" num_pins="33" equivalent="full"/>
182       <output name="O" num_pins="10" equivalent="full"/>
183       <clock name="clk" num_pins="1"/>
184
185       <!-- The logic block pins can connect to 15% of the wires passing by
          ↪
186            in the adjacent channel, and the pins are evenly spread
187            across all 4 sides of the logic block. Each pin appears on one
    ↪  side. -->
188
189       <fc in_type="frac" in_val="0.15" out_type="frac" out_val="0.15"/>
190       <pinlocations pattern="spread"/>
191
192
193       <!-- Describe basic logic element.
194              Each basic logic element has a 6-LUT that can be optionally
    ↪  registered
195          -->
196       <pb_type name="fle" num_pb="10">
197         <input name="in" num_pins="6"/>
```

```
198         <output name="out" num_pins="1"/>
199         <clock name="clk" num_pins="1"/>
200         <!-- 6-LUT mode definition begin -->
201         <mode name="n1_lut6">
202           <!-- Define 6-LUT mode -->
203           <pb_type name="ble6" num_pb="1">
204             <input name="in" num_pins="6"/>
205             <output name="out" num_pins="1"/>
206             <clock name="clk" num_pins="1"/>
207
208             <!-- Define LUT -->
209             <pb_type name="lut6" blif_model=".names" num_pb="1" class="lut">
210               <input name="in" num_pins="6" port_class="lut_in"/>
211               <output name="out" num_pins="1" port_class="lut_out"/>
212               <!-- LUT timing using delay matrix -->
213               <!-- Real LUTs have different delays per input
214                    but since VPR's router does not exploit
215                    that by changing which signal goes to which
216                    LUT input we'll make all the LUT
217                    delays the same to reduce CAD noise.
218                   -->
219               <delay_matrix type="max" in_port="lut6.in"
                  ↪ out_port="lut6.out">
220                 200e-12
221                 200e-12
222                 200e-12
223                 200e-12
224                 200e-12
225                 200e-12
226               </delay_matrix>
227             </pb_type>
228
229             <!-- Define flip-flop -->
230             <pb_type name="ff" blif_model=".latch" num_pb="1"
                  ↪ class="flipflop">
231               <input name="D" num_pins="1" port_class="D"/>
232               <output name="Q" num_pins="1" port_class="Q"/>
233               <clock name="clk" num_pins="1" port_class="clock"/>
234               <T_setup value="66e-12" port="ff.D" clock="clk"/>
235               <T_clock_to_Q max="124e-12" port="ff.Q" clock="clk"/>
236             </pb_type>
237
```

```
238             <!-- many lines below to describe the interconnect
239                  wires, muxes and crossbars inside a cluster.
240               -->
241             <interconnect>
242               <direct name="direct1" input="ble6.in" output="lut6[0:0].in"/>
243               <direct name="direct2" input="lut6.out" output="ff.D">
244                 <!-- Advanced user option that tells CAD tool to find
                      ↪ LUT+FF pairs in netlist
245                       and make sure it packs them together -->
246                 <pack_pattern name="ble6" in_port="lut6.out"
                      ↪ out_port="ff.D"/>
247               </direct>
248               <direct name="direct3" input="ble6.clk" output="ff.clk"/>
249               <mux name="mux1" input="ff.Q lut6.out" output="ble6.out">
250                 <!-- LUT to output is faster than FF to output on a Stratix
                      ↪ IV -->
251                 <delay_constant max="25e-12" in_port="lut6.out"
                      ↪ out_port="ble6.out"/>
252                 <delay_constant max="45e-12" in_port="ff.Q"
                      ↪ out_port="ble6.out"/>
253               </mux>
254             </interconnect>
255           </pb_type>
256           <interconnect>
257             <direct name="direct1" input="fle.in" output="ble6.in"/>
258             <direct name="direct2" input="ble6.out" output="fle.out[0:0]"/>
259             <direct name="direct3" input="fle.clk" output="ble6.clk"/>
260           </interconnect>
261         </mode>
262         <!-- 6-LUT mode definition end -->
263       </pb_type>
264       <interconnect>
265         <!-- We use a full crossbar to get logical equivalence at inputs of
            ↪ CLB
266                      The delays below come from Stratix IV. the delay
    ↪ through a connection block
267                      input mux + the crossbar in Stratix IV is 167 ps. We
    ↪ already have a 72 ps
268                      delay on the connection block input mux (modeled by
    ↪ Ian Kuon), so the remaining
269                      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>
```

```
304   <clocks>
305     <clock buffer_size="auto" C_wire="2.5e-10"/>
306   </clocks>
307 </architecture>
```