Lei Liu
[redacted]

Assignment 4

ECE 1756
December 11, 2023

1. Routing wire length study

length	minimum width	area per tile	critical path delay
1	52.25	7741.35	8.35
2	52.94	5749.35	7.01
4	62.02	5213.78	6.88
8	94.00	5970.90	8.14
16	182.08	8537.97	10.18

Table 1: geometric average of 5 architectures with different lengths

From Table 1, we can observe that table will form a "U-shape" like graph. The combination of area and delay will drop to the lowest when wire length equals to 4, and then start to increase.

For longer wires, although it provides a "highway" for long connections, long wires also introduce higher capacitance and resistance. It increases the delay for transferring signals. Since addional delay is introduced along the wire, we may need to ensure buffer or other resources that help to deal with clock skews/signal propogation to ensure timing correctness. Besides, addional routing resources are added for signals that rely on short connections/local connections. Overall, it increases the area needed per tile on the FPGA for longer wires.
For shorter wires, a signal need to be derived to multiple swith boxes to reach its destination, which would also create additional delay along the path.

As a result, it is found that wire = 4 architecture reaches a good balance to provide the best combination of area and delay.

2. Block to routing connectivity study

Fcin & Fcout (CLB)	minimum width	area per tile	critical path delay
0.15	62.02	5213.78	6.88
0.5	56.50	6879.30	7.38
1.0	55.78	9009.31	8.26

Table 2: geometric average of 3 architectures with different connectivity

As table 2 shows the result of different architecture with respect to the Connection Block Flexibility. It is found that minimum width would decrease when Fc increases. Since we increase the flexibility of the input/output driven in the CLB, there are more wires available in connection to support signal transfers. Therefore, we could maintain a lower minimum

width. However, increased connection flexibility requires increased connection block, such as addional muxes for input connections, addtional pass gates for output connections. This results in increased area and delay induced by the additional connection resources.
According to the experiment shown in Table 2, we can conclude that additional flexibility in connection impact more on dragging area and delay. Fc = 0.15 maintain the best combination of area and delay.

Fcin & Fcout (CLB)	minimum width	area per tile	critical path delay
0.15	108.25	8443.43	6.96
0.5	77.67	9003.79	7.51
1.0	63.33	10013.22	8.10

Table 3: geometric average of 3 architectures(sparse crossbars)with different connectivity

Architecture with sparse local routing will be more suitable when routing signals all the way through the local routing to the inputs of LUTs instead of to the cluster inputs. Thus, additional resource is added to deal with the routing from cluster inputs to LUTs input, which result in mimunum width, area and delay all increased.

5. Optimization study

Given the best result come from architecture with wire length = 4, Fcin = Fcout = 0.15 for CLB, and with full crossbar. We use this architecture as the base.
We first adjust the combination of Rmetal and Cmetal.

Architecture	minimum width	area per tile	critical path delay	area $\times delay$
base	62.02	5213.78	6.88	35870.81
0.4 Rmetal, 1.2 Cmetal	61.97	5159.83	6.89	35551.23
2 Rmetal, 0.6 Cmetal	61.90	5143.30	6.73	34614.41

Table 4: geometric average of 3 architectures with different Rmetal/Cmetal

From Table 4, we can conclude that by doubling the Rmetail, and reduce 40% of Cmetal, the wire is narrower which provide extra space to be compatible sharing area with other routing resources. Besides, frequency got increased caused by increased RC time constant, which further reduce path delay. We will now use 2Rmetal, 0.6Cmetal as the base.

We then adjust 15% of the wires on a higher metal layer.
According to Table 5, it is found that by putting 15% of wires on a higher metal layer, we could achieve a better combination of delay and area.
We then use the result found in Table 5 as the new base. After the consideration of electrical

Architecture	minimum width	area per tile	critical path delay	area $\times delay$
base	61.90	5143.30	6.73	34614.41
15% on higher metal layer	60.66	5075.06	6.56	33292.39

Table 5: geometric average of 2 architectures on different metal layer

parameters, we now change wire lengths and switch pattern.
From Table 1, it is found that the result of length = 2 is also competitive, so in Table 6, we tried to make 15% of the wire length to 2. And it turns out, from Table 6, Adding the wire length diversity does not contribute to deducting area and delay. Therefore, we will keep the base as the best result so far.

Architecture	minimum width	area per tile	critical path delay	area $\times delay$
base	60.66	5075.06	6.56	33292.39
15% in length = 2	58.24	5114.06	6.57	33599.37

Table 6: geometric average of 2 architectures on different lengths

Finally, we examine different combinations of switch pattern in the 85% of the wires.

Architecture	minimum width	area per tile	critical path delay	area $\times delay$
base	60.66	5075.06	6.56	33292.39
11011	63.78	5051.41	6.61	33389.82
10101	66.33	4963.84	6.65	33009.54

Table 7: geometric average of 2 architectures on different switch pattern

From Table 7, it is concluded that reducing the routing switches affects delay since lower switch-ability reduce the flexibility to transfer signals. However, it helps deduct area to a greater extent since switch resources(transistors/buffers) are reduced. From experiments, it is found that switch pattern: 10101 achieves the lowest area-delay product.
As a conclusion, the optimized architecture would be: the routing architecture with full crossbar, connectivity Fcin = Fcout = 0.15 for CLB,
85% of wires length in 4, R = 202, C = 13.5e-15, switch pattern = 1 0 1 0 1
15% of wires length in 4, R = 50, C = 13.5e-15, switch pattern = 1 1 1 1 1

Appendix

Appendix I: Optimized 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.
  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: Jason Luu, Jeff Goeders, Vaughn Betz
-->
<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 -->

  <!-- Descritions of the physical tiles that exist on the die begins -->
  <tiles>
    <tile name="io" area="0">
      <sub_tile name="io" capacity="8">
        <equivalent_sites>
          <site pb_type="io" pin_mapping="direct"/>
        </equivalent_sites>
        <input name="outpad" num_pins="1"/>
```

```
46          <output name="inpad" num_pins="1" />
47          <clock name="clock" num_pins="1" />
48          <fc in_type="frac" in_val="0.15" out_type="frac" out_val="0.15" />
49        <!-- IOs go on the periphery of the FPGA in this
50             architecture. Since I don't want to define four
51             different physical I/Os for the left, right, top,
52             and bottom sides just say each pin of the I/O
53             block is accessible from all four sides so we can
54             reach routing channels on some side of the block
55             no matter which side of the chip we're on.
56          -->
57          <pinlocations pattern="custom">
58            <loc side="left">io.outpad io.inpad io.clock</loc>
59            <loc side="top">io.outpad io.inpad io.clock</loc>
60            <loc side="right">io.outpad io.inpad io.clock</loc>
61            <loc side="bottom">io.outpad io.inpad io.clock</loc>
62          </pinlocations>
63        </sub_tile>
64      </tile>
65
66      <!-- Define general purpose logic block (CLB) begin -->
67            <!-- Area below is for everything inside the
68                 logic block (LUTs, FFs, intra-cluster
69                 routing). It's a bit on the low side given the large
                       crossbars in this
70                 architecture - more appropriate for a lower-cost
71                FPGA with smaller transistors and narrower metal.
72                -->
73      <tile name="clb" area="18000">
74        <!-- We can place a clustered block of type clb on a tile location
75             of type clb.
76            -->
77        <sub_tile name="clb">
78          <equivalent_sites>
79            <site pb_type="clb" pin_mapping="direct"/>
80          </equivalent_sites>
81
82          <!-- We have a full crossbar between the cluster inputs and the
83               LUT inputs, so the router can route to *any* input or from
84               *any* output on the logic block.  Hence mark the logic block
85               inputs as fully logically equivalent (swappable by the router)
                     and also the
86               logic block outputs as logically equivalent, which means
87               they can also be swapped by the router.
88              -->
89
90          <input name="I" num_pins="40" equivalent="full"/>
91          <output name="O" num_pins="10" equivalent="instance"/>
92          <clock name="clk" num_pins="1"/>
93          <fc in_type="frac" in_val="0.15" out_type="frac" out_val="0.15"/>
```

```
94          <pinlocations pattern="spread"/>
95        </sub_tile>
96      </tile>
97    </tiles>
98    <!-- Physical tile descriptions end -->
99
100   <!-- Chip layout (in terms of where tiles are) begins -->
101   <layout>
102     <auto_layout aspect_ratio="1.0">
103       <!--Perimeter of 'io' blocks with 'EMPTY' blocks at corners-->
104       <perimeter type="io" priority="100"/>
105       <corners type="EMPTY" priority="101"/>
106       <!--Fill with 'clb'-->
107       <fill type="clb" priority="10"/>
108     </auto_layout>
109   </layout>
110   <!-- Chip layout ends -->
111
112   <!-- Electrical and inter-cluster (general) routing description begins
        -->
113   <device>
114     <!-- Some area and timing parameters -->
115     <sizing R_minW_nmos="8926" R_minW_pmos="16067"/>
116     <!-- The grid_logic_tile_area below will be used for all blocks that do
          not explicitly set their own (non-routing)
117       area; set to 0 since we explicitly set the area of all blocks
            currently in this architecture file.
118      -->
119     <area grid_logic_tile_area="0"/>
120     <chan_width_distr>
121       <x distr="uniform" peak="1.000000"/>
122       <y distr="uniform" peak="1.000000"/>
123     </chan_width_distr>
124
125     <!-- Define the switch block pattern (pattern of switches between inter
          -tile routing wires)
126         The Wilton switch block is a sample pattern; you can use custom
              switch blocks for more control -->
127     <switch_block type="wilton" fs="3"/>
128
129     <!-- Set which switch to use for input connection blocks. Only affects
          timing and area, not connectivity -->
130     <connection_block input_switch_name="ipin_cblock"/>
131   </device>
132   <switchlist>
133     <!-- VB: the mux_trans_size and buf_size data below is in minimum width
          transistor *areas*, assuming the purple
134         book area formula. This means the mux transistors are about 5x
              minimum drive strength.
```

```
135           We assume the first stage of the buffer is 3x min drive strength
                  to be reasonable given the large
136           mux transistors, and this gives a reasonable stage ratio of a bit
                  over 5x to the second stage.
137           -->
138     <switch type="mux" name="0" R="551" Cin=".77e-15" Cout="4e-15" Tdel="58
          e-12" mux_trans_size="2.630740" buf_size="27.645901"/>
139     <!--switch ipin_cblock resistance set to yeild for 4x minimum drive
          strength buffer-->
140     <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"/>
141   </switchlist>
142   <segmentlist>
143     <!--- VB & JL: using ITRS metal stack data, 96 nm half pitch wires,
          which are intermediate metal width/space.
144           Wires of this pitch will fit over a 90 nm
145           high logic tile (which is about the height of a Stratix IV logic
                  tile).
146           I'm using a tile length of 90 nm, corresponding to the length of
                  a Stratix IV tile if it were square.
147           length below is in units of logic blocks, and Rmetal and Cmetal
                  are
148           per logic block passed, so wire delay adapts automatically if you
                  change the
149           length=? value.   -->
150
151     <!-- Currently only one type of routing wire, which
152          is of length 4 and has switches to every connection
153          box (4 of them) and switch box (5 of them)
154          it passes. You can change wirelengths just by changing the length=
                 "?" values
155          and changing the number of 1's (or 0's) in the <sb type and <cb
                 type lines to
156          match the number of switch blocks and connection blocks a wire of
                 that length
157          would span. -->
158     <segment freq="0.85" length="4" type="unidir" Rmetal="202" Cmetal="13.5
          e-15">
159       <mux name="0"/>
160       <sb type="pattern">1 0 1 0 1</sb>
161       <cb type="pattern">1 1 1 1</cb>
162     </segment>
163     <segment freq="0.15" length="4" type="unidir" Rmetal="50" Cmetal="13.5e
          -15">
164       <mux name="0"/>
165       <sb type="pattern">1 1 1 1 1</sb>
166       <cb type="pattern">1 1 1 1</cb>
167     </segment>
168   </segmentlist>
169   <!-- Electrical and inter-cluster routing description ends -->
```

```
170
171   <!-- Description of the capabilities (number of BLEs, modes) and local interconnect in
172         each type of complex (clustered) block (e.g. LBs) begins
173     -->
174   <complexblocklist>
175     <!-- Define I/O pads begin -->
176     <!-- 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.
177       -->
178     <pb_type name="io">
179       <input name="outpad" num_pins="1" />
180       <output name="inpad" num_pins="1" />
181       <clock name="clock" num_pins="1" />
182       <!-- IOs can operate as either inputs or outputs.
183         The delays below are to and from registers in the I/O (and generally I/Os are registered
184         today).
185         -->
186       <mode name="inpad">
187         <pb_type name="inpad" blif_model=".input" num_pb="1">
188           <output name="inpad" num_pins="1" />
189         </pb_type>
190         <interconnect>
191           <direct name="inpad" input="inpad.inpad" output="io.inpad">
192             <delay_constant max="4.243e-11" in_port="inpad.inpad" out_port="io.inpad" />
193           </direct>
194         </interconnect>
195       </mode>
196       <mode name="outpad">
197         <pb_type name="outpad" blif_model=".output" num_pb="1">
198           <input name="outpad" num_pins="1" />
199         </pb_type>
200         <interconnect>
201           <direct name="outpad" input="io.outpad" output="outpad.outpad">
202             <delay_constant max="1.394e-11" in_port="io.outpad" out_port="outpad.outpad" />
203           </direct>
204         </interconnect>
205       </mode>
206
207       <!-- Not modeling I/O power for now -->
208       <power method="ignore" />
209     </pb_type>
210     <!-- Define I/O pads ends -->
211
212     <!-- Define general purpose logic block (CLB) begin -->
213           <!-- Area below is for everything inside the
                 logic block (LUTs, FFs, intra-cluster
```

```
214                 routing).
215             -->
216       <pb_type name="clb">
217         <input name="I" num_pins="40" equivalent="full"/>
218         <output name="O" num_pins="10" equivalent="instance"/>
219         <clock name="clk" num_pins="1"/>
220         <!-- Describe basic logic element.
221                 Each basic logic element has a 6-LUT that can be optionally
                        registered
222           -->
223         <pb_type name="fle" num_pb="10">
224           <input name="in" num_pins="6"/>
225           <output name="out" num_pins="1"/>
226           <clock name="clk" num_pins="1"/>
227           <!-- 6-LUT mode definition begin -->
228           <mode name="n1_lut6">
229             <!-- Define 6-LUT mode -->
230             <pb_type name="ble6" num_pb="1">
231               <input name="in" num_pins="6"/>
232               <output name="out" num_pins="1"/>
233               <clock name="clk" num_pins="1"/>
234               <!-- Define LUT -->
235               <pb_type name="lut6" blif_model=".names" num_pb="1" class="lut"
                      >
236                 <input name="in" num_pins="6" port_class="lut_in"/>
237                 <output name="out" num_pins="1" port_class="lut_out"/>
238                 <!-- LUT timing using delay matrix -->
239                 <!-- These are the delay per LUT input on a Stratix IV LUT.
240                       The average is 261 ps, and inputs earlier in the mux
                            tree are slower.
241                   -->
242                 <delay_matrix type="max" in_port="lut6.in" out_port="lut6.out
                      ">
243                   82e-12
244                   173e-12
245                   261e-12
246                   263e-12
247                   398e-12
248                   397e-12
249                 </delay_matrix>
250               </pb_type>
251               <!-- Define flip-flop -->
252               <pb_type name="ff" blif_model=".latch" num_pb="1" class="
                      flipflop">
253                 <input name="D" num_pins="1" port_class="D"/>
254                 <output name="Q" num_pins="1" port_class="Q"/>
255                 <clock name="clk" num_pins="1" port_class="clock"/>
256                 <T_setup value="66e-12" port="ff.D" clock="clk"/>
257                 <T_clock_to_Q max="124e-12" port="ff.Q" clock="clk"/>
258               </pb_type>
```

```
259
260             <!-- many lines below to describe the interconnect
261                  wires, muxes and crossbars inside a cluster.
262               -->
263             <interconnect>
264               <direct name="direct1" input="ble6.in" output="lut6[0:0].in"/
                    >
265               <direct name="direct2" input="lut6.out" output="ff.D">
266                 <!-- Advanced user option that tells CAD tool to find LUT+
                        FF pairs in netlist -->
267                 <pack_pattern name="ble6" in_port="lut6.out" out_port="ff.D
                      "/>
268               </direct>
269               <direct name="direct3" input="ble6.clk" output="ff.clk"/>
270               <mux name="mux1" input="ff.Q lut6.out" output="ble6.out">
271                 <!-- LUT to output is faster than FF to output on a Stratix
                        IV -->
272                 <delay_constant max="25e-12" in_port="lut6.out" out_port="
                      ble6.out"/>
273                 <delay_constant max="45e-12" in_port="ff.Q" out_port="ble6.
                      out"/>
274               </mux>
275             </interconnect>
276           </pb_type>
277           <interconnect>
278             <direct name="direct1" input="fle.in" output="ble6.in"/>
279             <direct name="direct2" input="ble6.out" output="fle.out[0:0]"/>
280             <direct name="direct3" input="fle.clk" output="ble6.clk"/>
281           </interconnect>
282         </mode>
283         <!-- 6-LUT mode definition end -->
284       </pb_type>
285       <interconnect>
286         <!-- We use a full crossbar to get logical equivalence at inputs of
                CLB
287           The delays below come from Stratix IV. the delay through a
                connection block
288           input mux + the crossbar in Stratix IV is 167 ps. We already
                have a 72 ps
289           delay on the connection block input mux (modeled by Ian Kuon),
                so the remaining
290           delay within the crossbar is 95 ps.
291           The delays of cluster feedbacks in Stratix IV is 100 ps, when
                driven by a LUT.
292           Since all our outputs LUT outputs go to a BLE output, and have a
                 delay of
293           25 ps to do so, we subtract 25 ps from the 100 ps delay of a
                feedback
294           to get the part that should be marked on the crossbar.  -->
```

```
295         <complete name="crossbar" input="clb.I fle[9:0].out" output="fle[9
                :0].in">
296           <delay_constant max="95e-12" in_port="clb.I" out_port="fle[9:0].
                in" />
297           <delay_constant max="75e-12" in_port="fle[9:0].out" out_port="fle
                [9:0].in" />
298         </complete>
299         <complete name="clks" input="clb.clk" output="fle[9:0].clk">
300         </complete>
301
302         <!-- The BLE outputs are directly connected to the
303              CLB (cluster) outputs.
304           -->
305         <direct name="clbouts1" input="fle[9:0].out" output="clb.O" />
306       </interconnect>
307     </pb_type>
308     <!-- Define general purpose logic block (CLB) ends -->
309   </complexblocklist>
310   <power>
311     <local_interconnect C_wire="2.5e-10" />
312     <mux_transistor_size mux_transistor_size="3" />
313     <FF_size FF_size="4" />
314     <LUT_transistor_size LUT_transistor_size="4" />
315   </power>
316   <clocks>
317     <clock buffer_size="auto" C_wire="2.5e-10" />
318   </clocks>
319 </architecture>
```