FPGA Routing Architecture

Routing wire length study

Table 1: VPR results with wire length 2 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	56	5958.86	6.7792 ns
s38584	38	4296.83	5.0257 ns
ex1010	64	6688.24	7.0640 ns
elliptic	48	5104.77	7.8397 ns
s38417	34	3738.66	5.4308 ns
frisc	70	7241.68	8.8421 ns
clma	68	7623.89	9.0823 ns
pdc	58	6407.92	7.3492 ns
Tile Area Geo Avg.		5726.39	
Delay Geo Avg.		7.0420 ns	
Area-Delay Product		40325.45	

Table 2: VPR results with wire length 4 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	64	5355.57	6.2793 ns
s38584	48	4205.97	5.1803 ns
ex1010	80	6063.53	6.5166 ns
elliptic	56	4789.48	7.6782 ns
s38417	44	3646.36	5.5176 ns
frisc	80	6316.36	9.3310 ns
clma	80	6549.59	8.6427 ns
pdc	68	5574.69	7.0156 ns
Tile Area Geo Avg.		5218.62	
Delay Geo Avg.		6.8904 ns	
Area-Delay Product		35958.40	

Table 3: VPR results with wire length 8 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	110	6248.52	7.0831 ns
s38584	76	5106.43	6.3174 ns
ex1010	98	6754.48	7.4703 ns
elliptic	86	5468.58	9.2533 ns
s38417	66	4835.78	6.5198 ns
frisc	122	7544.12	11.0306 ns
clma	116	7576.01	9.8859 ns
pdc	104	6092.94	7.9693 ns
Tile Area Geo Avg.		6126.51	
Delay Geo Avg.		8.0437 ns	
Area-Delay Product		49280.03	

Table 4: VPR results with wire length 16 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	392	10797.77	8.6937 ns
s38584	184	10186.31	8.3378 ns
ex1010	214	10178.14	9.2664 ns
elliptic	188	8075.78	10.4068 ns
s38417	144	7840.62	7.6278 ns
frisc	196	9694.95	13.0207 ns
clma	218	10553.05	14.7574 ns
pdc	176	8934.78	10.0391 ns
Tile Area Geo Avg.		9472.08	
Delay Geo Avg.		10.0352 ns	
Area-Delay Product		95054.56	

To sum up, table 2 with wire length 4 has the best area-delay product of 35958.40 and table 4 with wire length 16 has the worst area-delay product of 95054.56. To explain the why wire length 4 has the best results, we need to know the pros and cons of wire length. In general, short wires are good for local connections because they have fewer loads compared to long wires with equivalent lengths. Long wires are good for long connections because they have fewer loads compared to short wires with equivalent lengths. It's impossible to say which wire length is the best without considering the circuits. For circuits where connections are more local, short wires perform better. For circuits where connections are sparse, long wires can take more advantages. In this experiment, wire length 4 performs the best, this implies the 8 benchmark circuits are mostly consist of local connections. if a different set of benchmark circuits with more sparse connections, wire length 8 may be the best one.

Block to routing connectivity study

Table 5: VPR results with Fc_{in}=Fc_{out}=0.15 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	64	5355.57	6.2793 ns
s38584	48	4205.97	5.1803 ns
ex1010	80	6063.53	6.5166 ns
elliptic	56	4789.48	7.6782 ns
s38417	44	3646.36	5.5176 ns
frisc	80	6316.36	9.3310 ns
clma	80	6549.59	8.6427 ns
pdc	68	5574.69	7.0156 ns
Tile Area Geo Avg.		5218.62	
Delay Geo Avg.		6.8904 ns	
Area-Delay Product		35958.40	

Table 6: VPR results with Fc_{in}=Fc_{out}=0.5 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	56	6846.56	6.9147 ns
s38584	44	5869.29	5.4753 ns
ex1010	64	7744.38	7.2833 ns
elliptic	50	6300.10	8.3402 ns
s38417	42	5143.20	5.8247 ns
frisc	70	8337.48	9.7050 ns
clma	74	8822.12	9.3450 ns
pdc	62	7495.34	7.5705 ns
Tile Area Geo Avg.		6968.32	
Delay Geo Avg.		7.4210 ns	
Area-Delay Product		51711.30	

Table 7: VPR results with Fc_{in}=Fc_{out}=1.0 using k6_N10_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	58	9078.29	7.8633 ns
s38584	44	7680.56	6.0022 ns
ex1010	62	10147.60	8.1462 ns
elliptic	50	8010.54	9.1660 ns
s38417	38	6922.01	6.1956 ns
frisc	70	11038.16	10.5242 ns
clma	72	11692.15	10.3230 ns
pdc	60	8590.66	8.5268 ns
Tile Area Geo Avg.		9149.55	
Delay Geo Avg.		8.1892 ns	
Area-Delay Product		74927.29	

Table 8: VPR results with Fc_{in}=Fc_{out}=0.15 using k6_N10_sparse_crossbar_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	76	8362.73	6.5268 ns
s38584	52	5942.46	5.6585 ns
ex1010	82	9257.27	6.6780 ns
elliptic	62	6897.25	8.1941 ns
s38417	46	5378.20	5.5592 ns
frisc	94	9208.08	9.2060 ns
clma	96	10174.30	8.9372 ns
pdc	78	9125.50	7.3902 ms
Tile Area Geo Avg.		7861.49	
Delay Geo Avg.		7.1501 ns	
Area-Delay Product		56210.17	

Table 9: VPR results with Fc_{in}=Fc_{out}=0.5 using k6_N10_sparse_crossbar_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	56	8309.62	7.3990 ns
s38584	44	7160.57	5.8646 ns
ex1010	62	9241.63	7.5887 ns
elliptic	50	7784.78	8.7047 ns
s38417	40	6482.27	5.8277 ns
frisc	70	10146.22	9,7750 ns
clma	70	10782.96	9.8300 ns
pdc	80	10710.95	7.9762 ns
Tile Area Geo Avg.		8689.91	
Delay Geo Avg.		7.7341 ns	
Area-Delay Product		67208.84	

Table 10: VPR results with Fc_{in}=Fc_{out}=1.0 using k6_N10_sparse_crossbar_40nm architecture (Results are averaged with 10 random seeds)

Circuit	Min Width	Low-Stress Tile Area	Low-Stress Critical Path Delay
spla	54	10149.52	8.1931 ns
s38584	44	9015.90	6.5452 ns
ex1010	58	11904.82	8.4745 ns
elliptic	52	9559.81	9.8722 ns
s38417	40	7971.24	6.4567 ns
frisc	66	12480.63	10.8783 ns
clma	68	13452.33	10.9211 ns
pdc	106	14509.28	9.0291 ns
Tile Area Geo Avg.		10920.16	
Delay Geo Avg.		8.6407 ns	
Area-Delay Product		94357.55	

The above tables from 5 to 10 shows the results for Fc=0.15, 0.5, 1.0 of two architectures. By observing the tables, we can conclude a general trend that minimum widths decrease with increasing Fc, whereas tile areas and critical delays increase with increasing Fc. Increasing Fc ratio also increase the number of channels available. Therefore, the minimum channel width decreases as logic blocks can connect to more channels. Although high Fc ratio gives better routability, it also occupies more areas in connection blocks and introduces more loads to wires. These effects directly lead to higher tile area and longer critical delays.

The choice between full crossbar and sparse crossbar does impact the minimum channel width, but it does not impact the choice of Fc. By comparing table 5 and table 8, we can see the minimum width increases for all 8 circuits when sparse crossbar architecture is applied. On the other hand, sparse crossbar shares the same trend when Fc ratio increases. Therefore, both Fc = 0.15 are the best choice for full crossbar and sparse crossbar.

Optimization study

For the custom architecture, I started with best architecture k6_N10_40nm.xml with wire length 4 and Fc=0.15. The first optimization is adding wire segments with different lengths and percentages to find the best area-delay product. In the first step, all the wires use 1 1 1 1... pattern for both switch block and connection block. In addition, resistance and capacitance are also kept unchanged. Table 11 shows the experiments I tested and the results I got. Results without wire length 4 give terrible area-delay product, so they are omitted in the table. From all the values, combination of wire length 2, 4, 8 gives the best outcome.

Table 11: Wire segments combination results

Wire segments and percentages	Area-Delay Product
wire-4 (100%)	35958.40
wire-2 (20%), wire-4 (80%)	35755.01
wire-4 (80%), wire-8 (20%)	36430.87
wire-4 (80%), wire-16 (20%)	38214.97
wire-2 (10%), wire-4 (80%), wire-8(10%)	35633.86
wire-2 (10%), wire-4 (80%), wire-16(10%)	35999.75
wire-4 (80%), wire-8(10%), wire-16(10%)	37249.21

After finding the best combination of wire segments, I did some tests on finding the best percentage of each wire segment. In order to shrink the search space and save time, I bound the length 4 wire percentage to a reasonable range, from 70% to 90%. This is because all benchmark circuits work well with wire length 4. Therefore, wire length 4 should always be the dominant part of all wires. Table 12 shows the results from varying percentage wire segment percentages, where using 12% wire length 2, 80% wire length 4, and 8% wire length 8 gives the best area-delay product.

Next, I started modifying the switch block and connection block pattern of each wire segment.

Table 12: Wire segments percentage results

Wire Segments and Percentages	Area-Delay Product
wire-2 (10%), wire-4 (80%), wire-8(10%)	35999.75
wire-2 (5%), wire-4 (70%), wire-8(25%)	36309.72
wire-2 (5%), wire-4 (80%), wire-8(15%)	36484.87
wire-2 (5%), wire-4 (84%), wire-8(11%)	36354.89
wire-2 (6%), wire-4 (88%), wire-8(6%)	36328.12
wire-2 (8%), wire-4 (80%), wire-8(12%)	35857.22
wire-2 (8%), wire-4 (84%), wire-8(8%)	35741.93
wire-2 (11%), wire-4 (84%), wire-8(5%)	35150.18
wire-2 (12%), wire-4 (80%), wire-8(8%)	35137.87
wire-2 (14%), wire-4 (78%), wire-8(8%)	35458.10
wire-2 (15%), wire-4 (70%), wire-8(15%)	35316.58
wire-2 (15%), wire-4 (80%), wire-8(5%)	35372.85
wire-2 (25%), wire-4 (70%), wire-8(5%)	35521.75

It's unnecessary for long wires to have connections to every switch block and connection block along its path. Therefore, the main focus is on reducing the connections for long wires and maybe a little bit of connections for middle and short wires. The table 13 shows the results and the one with 34750.51 area-delay product is the optimal pattern.

Table 13: Wire pattern results

Wire-2 (12%)	Wire-4 (80%)	Wire-8 (8%)	Area-Delay Product
sb 111 cb 11	sb 11111 cb 1111	sb 111111111 cb 11111111	35137.87
sb 111 cb 11	sb 11111 cb 1111	sb 111000111 cb 11100111	35592.73
sb 111 cb 11	sb 11111 cb 1111	sb 110000011 cb 11000011	35577.53
sb 111 cb 11	sb 11111 cb 1111	sb 100000001 cb 10000001	35906.76
sb 111 cb 11	sb 11111 cb 1111	sb 101010101 cb 10101010	35697.64
sb 111 cb 11	sb 11011 cb 1111	sb 110000011 cb 11000011	34750.51
sb 111 cb 11	sb 10001 cb 1001	sb 110000011 cb 11000011	55131.14
sb 111 cb 11	sb 11011 cb 1011	sb 110000011 cb 11000011	37277.34
sb 101 cb 11	sb 11011 cb 1111	sb 110000011 cb 11000011	34916.58

After confirming the best wire segments, percentages, and patterns, we need to find the best wire resistance and capacitance. There are three methods we can use to modify the wire physical properties: widening wires (reduce R, increase C), narrowing wires (increase R, reduce C), and allocating 15% of wires to a better metal layer (reduce R). The table 14 shows the results of testing different RC parameters. During the experiment, I found a trend that minimizing wire-2 resistance tends to give better outcomes. Therefore, I performed some tests about putting wire-2 on the better metal layer and also modify the percentages.

From the result table 14 we can see the best configuration is putting 15% wire-2 into a better metal layer and widening them (R=101x0.25x0.4=10.1, C=22.5e-15x1.2=27e-15), give 80% to wire-4 and narrowing them (R=101x2=202, C=22.5e-15x0.6=13.5e-15), the rest 5% is occupied by widened wire-8 (R=101x0.4=40.4, C=22.5e-15x1.2=27e-15).

Table 14: Wire resistance and capacitance results

Wire-2 (12%)	Wire-4 (80%)	Wire-8 (8%)	Area-Delay Product
R=101 C=22.5e-15	R=101 C=22.5e-15	R=101 C=22.5e-15	34750.51
R=40.4 C=27e-15	R=40.4 C=27e-15	R=40.4 C=27e-15	34631.91
R=202 C=13.5e-15	R=202 C=13.5e-15	R=202 C=13.5e-15	33968.26
R=40.4 C=27e-15	R=202 C=13.5e-15	R=40.4 C=27e-15	33619.34
R=202 C=13.5e-15	R=40.4 C=27e-15	R=202 C=13.5e-15	34310.85
R=40.4 C=27e-15	R=202 C=13.5e-15	R=101 C=22.5e-15	33840.48
R=101 C=22.5e-15	R=202 C=13.5e-15	R=40.4 C=27e-15	33911.29
R=10.1 C=27e-15	R=202 C=13.5e-15	R=10.1 C=27e-15 (3%) R=40.4 C=27e-15 (5%)	33682.43
R=10.1 C=27e-15 (7%) R=40.4 C=27e-15 (5%)	R=202 C=13.5e-15	R=10.1 C=27e-15	35192.79
R=40.4 C=27e-15	R=50.5 C=13.5e-15 (15%) R=202 C=13.5e-15 (65%)	R=40.4 C=27e-15	33861.58
R=10.1 C=27e-15 (15%)	R=202 C=13.5e-15	R=40.4 C=27e-15 (5%)	33531.65
R=40.4 C=27e-15 (5%)	R=202 C=13.5e-15	R=10.1 C=27e-15 (15%)	34990.96
R=10.1 C=27e-15 (15%)	R=40.4 C=27e-15	R=101 C=22.5e-15	34383.66

The last step is to tune the Fc ratio. In the original architectures, we only tested 0.15, 0.5, and 1.0 Fc values. It's obvious that Fc=0.15 gives the results among the three. In this optimization, we need to explore more values of Fc and double check whether Fc=0.15 is indeed the best ratio. The table 15 shows all the tested Fc values, Fc=0.1 comes out to be best result. This means the benchmarks circuits don't really need Fc=0.15, they can be implemented with less Fc ratio.

Table 15: Wire Fc results

Fc	Area-Delay Product
0.15	33531.65
0.05	36992.13
0.075	33924.49
0.088	33797.64
0.1	32742.00
0.125	33131.24
0.175	34514.97
0.2	35550.24

The final optimal architecture has an area-delay product of 32742.00 with 4996.52 tile area and 6.5530 critical delay. It uses 15% of wire length 2, 80% of wire length 4, and 5% of wire length 8. The wire length 2 uses a switch block pattern of 1 1 1 and a connection block pattern of 1 1. The wire length 4 uses a switch block pattern of 1 1 0 1 1 and a connection block pattern of 1 1 1 1. The wire length 8 uses a switch block pattern of 1 1 0 0 0 0 0 0 1 1 and a connection block pattern of 1 1 0 0 0 0 1 1. The wire length 4 is narrowed to minimize capacitance, whereas wire length 2 and 8 are widened to minimize resistance. The architecture uses Fc=0.1 with a unchanged Wilton switch block of Fs=3.

Appendix A: Architecture File

```
<!--
  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"/>
      <output name="inpad" num_pins="1"/>
      <clock name="clock" num_pins="1"/>
      <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>
    </sub_tile>
  </tile>

  <!-- Define general purpose logic block (CLB) begin -->
        <!-- Area below is for everything inside the
             logic block (LUTs, FFs, intra-cluster
             routing). It's a bit on the low side given the
                large crossbars in this
```

```
                architecture - more appropriate for a lower-
                   cost
                FPGA with smaller transistors and narrower
                   metal.
                -->
    <tile name="clb" area="18000">
      <!-- We can place a clustered block of type clb on a tile
          location
           of type clb.
           -->
      <sub_tile name="clb">
        <equivalent_sites>
          <site pb_type="clb" pin_mapping="direct"/>
        </equivalent_sites>

      <!-- 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="40" equivalent="full"/>
        <output name="O" num_pins="10" equivalent="instance"/>
        <clock name="clk" num_pins="1"/>
        <fc in_type="frac" in_val="0.1" out_type="frac" out_val
           ="0.1"/>
        <pinlocations pattern="spread"/>
      </sub_tile>
    </tile>
  </tiles>
  <!-- Physical tile descriptions end -->

  <!-- Chip layout (in terms of where tiles are) begins -->
  <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>
  <!-- Chip layout ends -->

  <!-- Electrical and inter-cluster (general) routing
     description begins -->
  <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"/>
    <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)
         The Wilton switch block is a sample pattern; you can
            use custom switch blocks for more control -->
    <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.
            We assume 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="0" 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, so wire delay adapts
          automatically if you change the
        length=? value.   -->

    <!-- 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. You can change wirelengths just by changing
          the length="?" values
        and changing the number of 1's (or 0's) in the <sb
          type and <cb type lines to
        match the number of switch blocks and connection
          blocks a wire of that length
        would span. Rmental = 101, Cmetal - 22.5e-15 34351 -->
    <segment freq="0.15000" length="2" type="unidir" Rmetal="
      10.1" Cmetal="27e-15">
      <mux name="0"/>
      <sb type="pattern">1 1 1</sb>
      <cb type="pattern">1 1 </cb>
```

```
    </segment>
    <segment freq="0.80000" length="4" type="unidir" Rmetal="
       202" Cmetal="13.5e-15">
      <mux name="0"/>
      <sb type="pattern">1 1 0 1 1</sb>
      <cb type="pattern">1 1 1 1</cb>
    </segment>
    <segment freq="0.050000" length="8" type="unidir" Rmetal="
       40.4" Cmetal="27e-15">
      <mux name="0"/>
      <sb type="pattern">1 1 0 0 0 0 0 1 1</sb>
      <cb type="pattern">1 1 0 0 0 0 1 1</cb>
    </segment>
  </segmentlist>
  <!-- Electrical and inter-cluster routing description ends --
     >

  <!-- Description of the capabilities (number of BLEs, modes)
     and local interconnect in
       each type of complex (clustered) block (e.g. LBs) begins
    -->
  <complexblocklist>
    <!-- Define I/O pads begin -->
    <!-- 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">
      <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 delays 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>

  <!-- 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">
  <input name="I" num_pins="40" equivalent="full"/>
  <output name="O" num_pins="10" equivalent="instance"/>
  <clock name="clk" num_pins="1"/>
  <!-- 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 -->
    <!-- These are the delay per LUT input on a
      Stratix IV LUT.
        The average is 261 ps, and inputs earlier in
            the mux tree are slower.
       -->
    <delay_matrix type="max" in_port="lut6.in"
      out_port="lut6.out">
       82e-12
       173e-12
       261e-12
       263e-12
       398e-12
       397e-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 -->
          <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>
      </pb_type>
      <!-- Define general purpose logic block (CLB) ends -->
    </complexblocklist>
    <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>
```