COFFE v1.1 User's Manual

1. Overview

COFFE is an automated transistor-level design tool for FPGAs. Automated transistor-level design of FPGA architectures is crucial when evaluating many different architectures with tools such as VPR. Different architectures have different transistor-level requirements. For instance, transistor and wire loading greatly depend on the FPGA's architecture. Thus transistor-level design should be performed for each architecture that is evaluated to ensure a thorough exploration of the design space. COFFE enables this thorough design space exploration by providing area and delay estimates of properly sized FPGA circuitry for a user specified architecture. These estimates can then be fed into VPR to evaluate the architecture through place and route experiments.

The input to COFFE is the architecture and process technology description file detailed in Section 3. COFFE's outputs are area and delay estimates that can be used to create VPR architecture files. For more details about how COFFE works, see [1]. The remainder of this manual is a tutorial that will walk you through an example COFFE run.

2. Setup

COFFE is a Python program built on top of HSPICE. Consequently, you will need Python and a license for HSPICE to use COFFE. COFFE was developed for Python 2.7.3 and HSPICE Version F-2011.09-SP1 32-BIT. However, starting with COFFE v1.1, any version of HSPICE should work just fine.

COFFE uses the 'argparse' Python package. This package is included in Python 2.7. If your version of Python is <2.7, you may need to install the 'argparse' package.

If you have any problems running COFFE with different versions of Python or HSPICE please let us know as we try to make COFFE more stable over different versions.

You can download COFFE here: http://www.eecg.toronto.edu/~vaughn/software.html

Once you've unzipped the file, you should be ready to use COFFE.

3. Creating the input file

In the 'input_files' directory, you will find a file called 'example.txt'. This is an example COFFE input file. It's got lots of comments explaining how things work; feel free to take those out in your own input files. COFFE expects its input file to describe two different kinds of parameters: architecture parameters and process technology parameters.

3.1 Architecture parameters

Figure 1 shows COFFE's supported FPGA architecture and Table 1 describes each architecture parameter. Most architecture parameters are the classic VPR architecture parameters (N, K, etc.) but some parameters (the ones below the double line in Table 1) are new and help COFFE describe a more flexible BLE. See [1] for additional details.

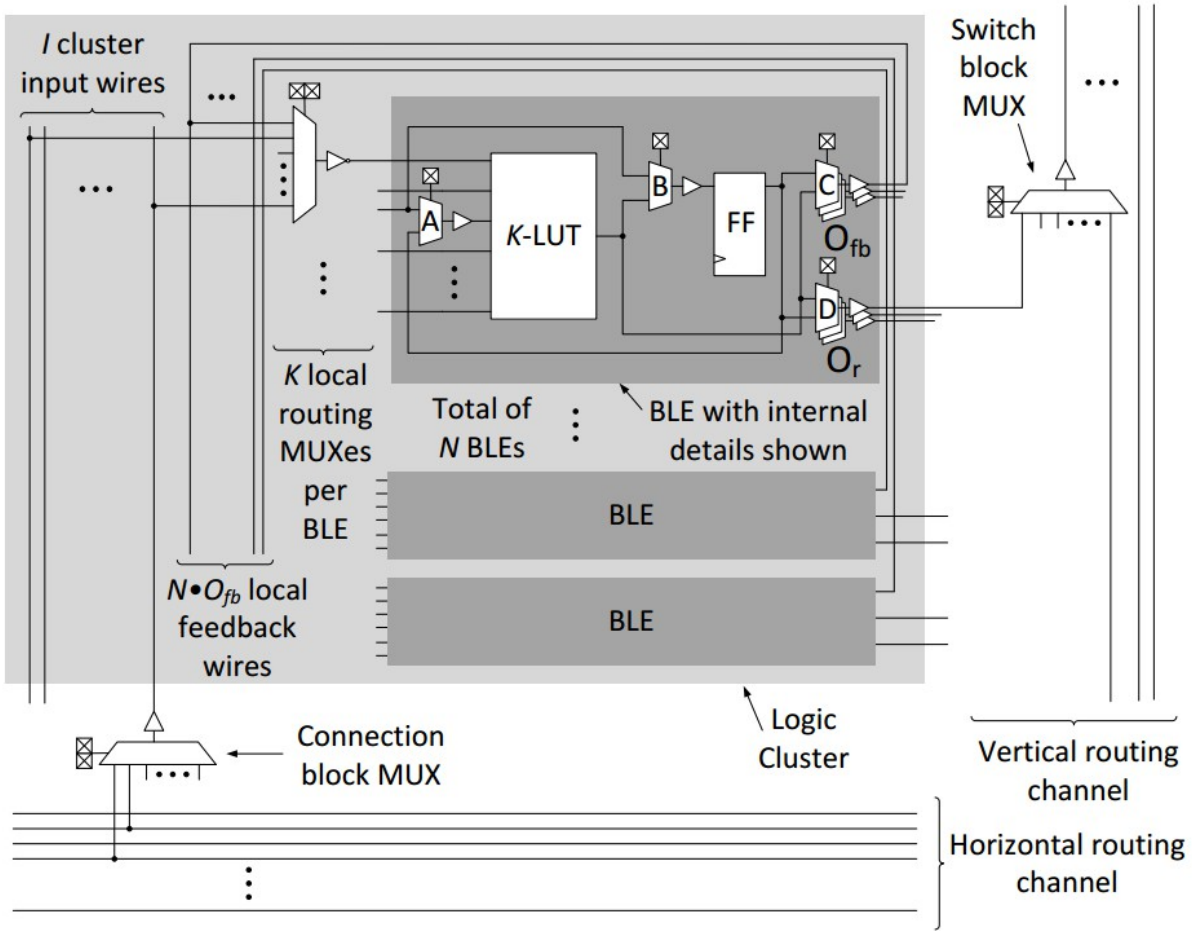

Figure : COFFE's supported architecture

Table : COFFE architecture parameters

Parameter	Legal values	Description
N	Positive int	Number of BLEs in a logic cluster.
K	4, 5, 6	LUT size. COFFE currently only supports three LUT sizes: K = 4, 5 or 6.
W	Positive int	Routing channel width (# routing tracks per channel).
L	Positive int	Routing wire length.
I	Positive int	Number of logic cluster inputs.
Fs	Positive int	Switch block flexibility.
Fcin	0 < Fcin <= 1.0	Logic cluster input connection flexibility.

Fcout	0 < Fcout <= 1.0	Logic cluster output connection flexibility.
Or	Positive int	Number of BLE outputs to general routing.
Ofb	Positive int	Number of BLE outputs to local routing (local feedback)
Fclocal	0 < Fclocal < 1.0	Local routing multiplexer population.
Rsel	a, b, …, z	This parameter defines whether or not the FF can accept its input from both the LUT output and a BLE input[1] (MUX B in Figure 1). Setting Rsel=z tells COFFE that the FF only accepts its input from the LUT output. Setting Rsel to anything else tells COFFE that the FF accepts its input from both the LUT output and a BLE input. For example, Rsel=c tells COFFE that the FF can accept its input from both the LUT output and BLE input 'c'.
Rfb	a, b, …, z ab, ac, bcd, abcdef, etc.	This parameter defines which LUT inputs support register feedback (MUX A in Figure 1). Setting Rfb=z tells COFFE that no LUT inputs support register feedback. Setting Rfb to anything else tells COFFE to put a 2:1 mux on those LUT inputs because they support register feedback. For example, Rfb=a means that LUT input 'a' supports register feedback. COFFE allows more than one LUT input to support register feedback at the same time. Rfb=ab means that both LUT inputs 'a' and 'b' support register feedback (each one would have a 2:1 mux). Similarly, Rfb=abcd means that LUT inputs 'a', 'b', 'c' and 'd' all support register feedback.

3.2 Process technology parameters

Table 2 describes COFFE's process technology parameters. Before you can fill in any of these parameters, you need to choose which process technology you are going to design for. Once this choice has been made, obtaining values for the first 6 parameters of Table 2 is relatively straight forward. However, the 'vsram' and 'vsram_n' parameters might be a little bit trickier to choose. Often, 'vsram' is boosted above 'vdd' to improve pass-transistor performance but it's difficult to tell how much of this boosting is safe. You might be able to get an idea of what 'vsram' voltage to use from [2].

Table : Process technology parameters

Parameter	Description
vdd	Supply voltage in volts.

1Note that by convention, COFFE labels LUT inputs (and corresponding BLE inputs) with letters. LUT input 'a' is the LUT input that controls the first level of pass-transistors along the LUT signal path (i.e. the pass-transistors closest to the SRAM cells).

vsram	Vdd for SRAM cells in volts.
vsram_n	Vss for SRAM cells in volts.
gate_length	Gate length for this process (nm).
min_tran_width	Diffusion width of a minimum sized contactable transistor (nm).
min_width_tran_area	Area of a minimum width transistor (nm^2).
sram_cell_area	Area of an SRAM cell in units of minimum-width transistor areas (i.e. units of min_width_tran_area).
model_path	Path to the file that contains the SPICE models to use. See SPICE models note below.
model_library	Library in the 'model_path' that contains the SPICE models to use. See SPICE models note below.
metal	The 'metal' parameter allows you to add resistance and capacitance data for a metal layer. A 'metal' statement has the following format: metal=R,C where R and C are replaced by the resistance per nm (Ω/nm) and capacitance per nm (fF/nm) of wires implemented in this layer. Each 'metal' statement adds a new metal layer. COFFE will use the first 'metal' statement it sees as the lowest metal layer, which is the layer where most wires are implemented. By default, COFFE requires two metal layers because it places the general routing wires in a separate layer. This allows us to do something like this: metal=0.054825,0.000175 metal=0.007862,0.000215 In this example, the second 'metal' statement defines a metal layer with 2x minimum width and spacing. Implementing the general routing wires in this layer allows them to benefit from the reduced resistance.

Recall that COFFE is built on top of HSPICE. Therefore, you need to provide COFFE with SPICE models for your chosen process technology. To make running a COFFE example easier, we included PTM 22nm HP [3] spice models in the COFFE distribution (you'll find them here: 'spice_models/ptm22hp.l'). If you use these models in your work, **please remember to acknowledge PTM** (http://ptm.asu.edu/).

The 'model_path' parameter of Table 2 tells COFFE which file to go look in for SPICE models. The 'model_library' tells COFFE which library to load from this file.

COFFE assumes that the model name of an NMOS transistor is 'nmos' and the model name of a PMOS transistor is 'pmos' and it builds its netlists as such (see 'spice_models/ptm22hp.l').

Finally, you need to provide COFFE with metal resistance and capacitance for your process technology. This is done with the 'metal' parameter as described in Table 2.

4. Running COFFE

To run COFFE on our 'example.txt' input file, type the following from COFFE's root directory:

```
> python coffe.py input_files/example.txt
```

As it's running, COFFE will print a variety of things to the terminal. Section 5 will explain what some of this data means. On an Intel Xeon E5-1620 3.6GHz processor core, it takes COFFE approximately 4 hours and 45 minutes to complete its transistor sizing work. Therefore, it might be a good idea to take a look at Section 5 right now instead of waiting for COFFE to finish. That way, you can compare what you read in Section 5 to what's happening in the terminal in real time.

COFFE has a few options that allow you to control some parts of its operation. These options are listed in Table 3. To get a list of the options in the terminal, type:

```
> python coffe.py -h
```

Table : COFFE options

Option	Description
-h	Help
-n	Use this option to turn off transistor sizing. If COFFE finds the '-n' option on the command line, it simply generates SPICE netlists, calculates area and delay its current transistor sizes and prints the results to the terminal.
-o	Use this option to specify the optimization type (local or global). Default = global. '-o global' tells COFFE to perform a global optimization. This means that COFFE will choose transistor sizes that optimize total tile area and representative critical path delay. '-o local' tells COFFE to perform a local optimization. This means that COFFE will choose transistor sizes that optimize each subcircuits area and delay.
-m	Use this option to specify the number of top-ranked transistor sizing combinations that COFFE should re-balance rise and fall times on. Default = 1. For example, '-m 5' would tell COFFE to re-balance the rise and fall times on the 5 top ranked transistor sizing combinations before choosing a transistor sizing solution for a subcircuit.
-a	Use this option to specify 'b' in the cost function: cost=area^b * delay^c. Default = 1.
-d	Use this option to specify 'c' in the cost function: cost=area^b * delay^c. Default = 1.
-i	Use this option to change the maximum number of FPGA transistor sizing iterations that COFFE will perform before forcing termination. Default = 6. COFFE will terminate either when the cost stops decreasing or when a maximum number of FPGA transistor sizing iterations have been performed. You can change the maximum number of these FPGA sizing iterations with the '-i' parameter.

	For example, '-i 1' will make COFFE terminate after a single FPGA sizing iteration.

5. Interpreting the output

As it's running, COFFE will print a variety of things to the terminal. If you are only interested in the final results, skip Section 5.1 and go straight to Section 5.2. Section 5.1 will explain what some of the intermediate data means.

5.1 Interpreting the intermediate data

One of the first interesting things that COFFE is going to print to the terminal is 'implementation details'.

```
|-----------------------------------------------------------------------------|
|   FPGA Implementation Details                                               |
|-----------------------------------------------------------------------------|
```

When COFFE creates FPGA circuitry from the input architecture parameters, it figures out a few things about this circuitry. For example, it determines the size (number of inputs) of routing multiplexer, it determines how many multiplexers there are per tile and it determines how many SRAM cells the multiplexer requires. Here is an example for connection blocks.

```
CONNECTION BLOCK DETAILS:
Style: two-level MUX
Required MUX size: 64:1
Implemented MUX size: 64:1
Level 1 size = 8
Level 2 size = 8
Number of unused inputs = 0
Number of MUXes per tile: 40
Number of SRAM cells per MUX: 16
```

COFFE prints implementation details like this for each one of its main subcircuits.

When COFFE starts sizing a new subcircuit, it will print a banner such as this one to tell you which subcircuit it is currently working on (in this case, a switch block multiplexer).

```
|-----------------------------------------------------------------------------|
|    Transistor sizing on subcircuit: sb_mux                                  |
|-----------------------------------------------------------------------------|
```

COFFE will print the transistor size ranges it is exploring for a particular subcircuit like this:

```
Transistor size ranges for sb_mux:
ptran_sb_mux_L2  : [2 -> 6]
ptran_sb_mux_L1  : [1 -> 5]
inv_sb_mux_1     : [2 -> 6]
rest_sb_mux      : [1 -> 1]
inv_sb_mux_2     : [8 -> 12]
```

This is a list of transistors in the 'sb_mux' subcircuit along with the transistor sizing ranges that COFFE is going to explore for each one. For example, 'inv_sb_mux_2' is the inverter that drives the 'sb_mux' output. Recall from [1] that although there are two transistors in an inverter, COFFE sizes the NMOS and PMOS of inverters as a unit based on a pre-computed P/N ratio. This is why there is only one sizing range for 'inv_sv_mux_2'.

Once COFFE is done calculating the area, delay and cost of each transistor sizing combination in these sizing ranges, it will print the 10 best results that it found:

```
TOP 10 BEST COST RESULTS
------------------------
Combo #106   cost=0.160401 area=871.82  delay=1.84e-10  tfall=1.575e-10 trise=1.619e-10
Combo #105   cost=0.160616 area=868.327 delay=1.85e-10  tfall=1.616e-10 trise=1.633e-10
Combo #231   cost=0.160735 area=876.582 delay=1.834e-10 tfall=1.567e-10 trise=1.593e-10
Combo #237   cost=0.160935 area=893.78  delay=1.801e-10 tfall=1.465e-10 trise=1.511e-10
Combo #111   cost=0.160946 area=885.795 delay=1.817e-10 tfall=1.495e-10 trise=1.571e-10
Combo #236   cost=0.161002 area=890.556 delay=1.808e-10 tfall=1.485e-10 trise=1.531e-10
Combo #107   cost=0.161029 area=875.044 delay=1.84e-10  tfall=1.563e-10 trise=1.633e-10
Combo #232   cost=0.161135 area=879.805 delay=1.831e-10 tfall=1.554e-10 trise=1.594e-10
Combo #110   cost=0.16118  area=882.301 delay=1.827e-10 tfall=1.544e-10 trise=1.578e-10
Combo #112   cost=0.161228 area=889.018 delay=1.814e-10 tfall=1.479e-10 trise=1.569e-10
```

The 'area' and 'delay' values that COFFE prints here depend on which type of optimization you chose (global or local, see Table 3). If you chose global optimization, COFFE prints tile area and representative critical path delay. If you chose local optimization, COFFE prints the area and delay of the current subcircuit that it is sizing. 'tfall' and 'trise' are *always* the rise and fall delay for the current subcircuit and they give you an indication of how well the pre-computed P/N ratio is working for this transistor sizing combination.

The next thing that COFFE will do is re-balance the rise and fall delays of the top M transistor sizing combinations where M is an option that can be set by the user (see Table 3).

```
Re-equalizing rise and fall times on combo #106 (ranked #1)

BEST COST RESULTS AFTER RE-BALANCING
------------------------------------
Combo #106   cost=0.158838 area=874.775 delay=1.816e-10 tfall=1.527e-10 trise=1.53e-10
```

In this case, M=1. So, COFFE only re-balances the rise and fall delays of the #1 best transistor sizing combination. Notice how 'tfall' and 'trise' are fairly equal. Compare that to Combo #106 in the 'Top 10 Best Cost Results' above. Notice also how the cost of Combo #106 changed after re-balancing.

Sometimes, when M >1, re-balancing changes the order of the top ranked transistor sizing solutions. Hence, Making M > 1 may give you a better transistor sizing solution but it will increase COFFE's runtime.

Once it has found the best cost transistor sizing combination, COFFE is going to print it out to the terminal. This is shown in the first part of the screenshot below.

```
Sizing results for sb_mux:
inv_sb_mux_2_pmos    : 26.8
ptran_sb_mux_L2_nmos : 3.0
ptran_sb_mux_L1_nmos : 2.0
inv_sb_mux_1_nmos    : 2.0
inv_sb_mux_2_nmos    : 12.0
inv_sb_mux_1_pmos    : 7.9
rest_sb_mux_pmos     : 1.0

Validating transistor sizes
inv_sb_mux_1 is on lower bound
inv_sb_mux_2 is on upper bound
Sizing results not valid, computing new sizing ranges
```

COFFE's next task is to check if these transistor sizing results are valid. The transistor sizes are valid if they are not on the boundaries of the transistor sizing ranges we initially set out to explore. As you can see from the screenshot, COFFE finds that 'inv_sb_mux_1' and 'inv_sb_mux_2' are on one of the bounds of their transistor sizing ranges (*Wait! 'inv_sb_mux_2' has two sizes: 12.0 for the NMOS and 26.8 for the PMOS? Which one is COFFE using?* COFFE always uses the smallest of the two transistor sizes for an inverter. The other one (the larger one) is obtained via the P/N ratio and thus as nothing to do with the transistor size range.).

COFFE found that some transistors were on the boundaries of their transistor sizing range. Therefore, this is not a valid transistor sizing solution. COFFE is going to adjust the transistor sizing ranges around the current solution and try again. COFFE informs you of this with something that looks like this.

```
Transistor size ranges for sb_mux:
ptran_sb_mux_L2  : [2 -> 6]   [2 -> 6]
ptran_sb_mux_L1  : [1 -> 5]   [1 -> 5]
inv_sb_mux_1     : [2 -> 6]   [1 -> 8]
rest_sb_mux      : [1 -> 1]   [1 -> 1]
inv_sb_mux_2     : [8 -> 12]  [10 -> 18]
```

The first transistor sizing ranges are the same ones that we saw in an earlier screenshot. COFFE is not exploring these ranges again; it's just showing you what transistor sizing ranges it's explored so far. The rightmost transistor sizing ranges are the ones that COFFE is going to explore next. Notice how the transistors that have had their sizing range changed are 'inv_sub_mux_1' and 'inv_sb_mux_2', i.e. the transistors whose size was on the boundaries of the first transistor sizing ranges.

COFFE will keep adjusting the transistor sizing ranges like this until it finds transistor sizes that are contained entirely within the transistor sizing ranges (nothing on the boundaries).

Once COFFE is done with sizing 'sb_mux' it will move on to the next subcircuit and perform this same transistor sizing algorithm. Sizing all of the FPGA's subcircuits once is called an *FPGA sizing iteration*. After each FPGA sizing iteration, COFFE prints out its transistor sizing results. Below is a screenshot of what COFFE printed after 3 FPGA sizing iterations.

```
FPGA TRANSISTOR SIZING RESULTS
------------------------------
cb_mux:
inv_cb_mux_1    : 2   2   2
inv_cb_mux_2    : 4   4   3
ptran_cb_mux_L2 : 2   2   2
rest_cb_mux     : 1   1   1
ptran_cb_mux_L1 : 2   2   2

general_ble_output:
ptran_general_ble_output : 10  10  10
inv_general_ble_output_1 : 3   3   3
rest_general_ble_output  : 1   1   1
inv_general_ble_output_2 : 3   3   2

local_ble_output:
inv_local_ble_output_2 : 10  11  10
ptran_local_ble_output : 21  16  16
inv_local_ble_output_1 : 11  8   11
rest_local_ble_output  : 1   1   1
```

Each column shows the sizing results for one FPGA sizing iteration.

COFFE then prints the total cost of for each FPGA sizing iteration. Each row shows the results for one FPGA sizing iteration.

```
TOTALS:
Area            Delay           Cost
912.639 1.2442e-10      0.11355
951.197 1.1361e-10      0.10807
992.036 1.1501e-10      0.11409

Evaluating termination conditions...
Algorithm is terminating: cost has stopped decreasing

FPGA transistor sizing complete!
```

COFFE terminates when the cost stops decreasing (or when a maximum number of FPGA sizing iterations have been performed (see Table 3)). In this case, the cost stopped decreasing after the 2nd iteration. So, COFFE choses the transistor sizing results of the 2nd iteration as its final solution.

5.2 Interpreting the final results

Once COFFE is done sizing the transistors of your FPGA, it will print out the area and delay of each subcircuit in your FPGA.

```
|-------------------------------------------------------------------------------|
|    Area and Delay Report                                                      |
|-------------------------------------------------------------------------------|

  SUBCIRCUIT AREA & DELAY
  -----------------------
  Subcircuit            Area (um^2)  Delay (ps)   trise (ps)   tfall (ps)
  sb_mux                1.219        136.5494     136.5494     135.8969
  cb_mux                2.913        92.4371      91.4944      92.4371
  local_mux             1.016        54.9204      54.9204      54.5299
  local_ble_output      0.99         97.042       97.042       96.7347
  general_ble_output    0.6          32.5677      32.5677      32.5623
  lut                   24.339       185.433      185.433      185.2742
  lut_a_driver          0.247        13.7571      13.7571      13.7372
  lut_a_driver_not      0.317        19.0997      19.0933      19.0997
  lut_b_driver          0.227        11.9112      11.9112      11.8384
  lut_b_driver_not      0.3          17.373       17.3607      17.373
  lut_c_driver          0.535        33.2228      32.8635      33.2228
  lut_c_driver_not      0.219        36.0386      36.0386      35.6731
  lut_d_driver          0.196        10.7715      10.7715      10.7102
  lut_d_driver_not      0.256        17.4311      16.8698      17.4311
  lut_e_driver          0.175        9.5597       9.5443       9.5597
  lut_e_driver_not      0.225        16.6466      16.6466      16.6301
  lut_f_driver          0.151        8.1296       8.1099       8.1296
  lut_f_driver_not      0.208        14.7945      14.7945      14.7862
```

COFFE will also print out the area of important blocks in your FPGA. For example, the ‘Switch block’ area shown below is the total area of all the switch block multiplexers in a tile, including their SRAM cells.

```
  TILE AREA CONTRIBUTIONS
  -----------------------
  Block             Total Area (um^2)   Fraction of total tile area
  Tile              951.197             100%
  LUT               273.956             28.801%
  FF                10.458              1.099%
  BLE output        21.901              2.302%
  Local mux         130.202             13.688%
  Connection block  190.362             20.013%
  Switch block      324.32              34.096%
```

To make it easier for you to create VPR architecture files from its transistor sizing results, COFFE prints out VPR specific area and delay numbers. The names in the screenshot below match those found in VPR architecture files as much as possible. The units are also given such that you can simply copy and paste the results to the right location in a VPR architecture file (i.e. delay in seconds and area in units of minimum-width transistor areas).

```
VPR DELAYS
----------
Path                                          Delay (ps)
Tdel (routing switch)                         1.365494e-10
T_ipin_cblock (connection block mux)          9.24371e-11
CLB input -> BLE input (local CLB routing)    5.49204e-11
LUT output -> BLE input (local feedback)      9.7042e-11
LUT output -> CLB output (logic block output) 3.25677e-11
lut_a                                         2.080497e-10
lut_b                                         2.067423e-10
lut_c                                         2.073079e-10
lut_d                                         1.312244e-10
lut_e                                         1.279759e-10
lut_f                                         1.147417e-10

VPR AREAS
----------
grid_logic_tile_area                          15130.0121668
ipin_mux_trans_size (connection block mux)    1.25595750289
mux_trans_size (routing switch)               1.50823186576
buf_size (routing switch)                     19.6339768892
```

Finally, COFFE prints out the area, delay and cost of its transistor sizing solution. With ‘example.txt’, you should get the results below.

```
SUMMARY
-------
Tile Area                          951.2 um^2
Representative Critical Path Delay 113.64 ps
Cost (area^1 x delay^1)            0.1081
```

References

[1] C.Chiasson and V.Betz. “COFFE: Fully-Automated Transistor Sizing For FPGAs”, in IEEE Int. Conf. on Field-Programmable Technology (FPT), 2013

[2] C. Chiasson and V.Betz. “Should FPGAs Abandon the Pass-Gate?”, in IEEE Int. Conf. on Field-Programmable Logic and Applications (FPL), 2013

[3] Predictive Technology Model (PTM), http://ptm.asu.edu/, 2012.