ISPD 2012 Contest Evaluation

Last modified: January 29, 2012

Disclaimer

- The information published in this presentation is subject to change.
- It is the contestants responsibility to check the website frequently to check for any updates until the submission deadline.

Benchmarks

- We are planning to use a subset of the benchmarks* posted on the website by January 9, 2012 for final evaluations
  - In this case, the netlists (the .v files) will be reused as posted
  - The constraint files .sdc and .spef will be different from those posted
  - However, we reserve the right to modify the netlists and/or add new benchmarks if needed

- The final cell library file (.lib) that will be used in evaluations will be posted on the website by January 9, 2012

* The exact subset will be decided based on the number of final submissions

Output file

- When your sizer is run, it is expected to produce a <benchmark>.sizes file
  - The .sizes file format is defined in “ISPD_2012_Contest_Details.pdf” presentation on “Sizer Output (.sizes) File” slide
  - Each line is defined as:

    <full-instance-name>   <library-cell-name>
- Logic transformations are not allowed
  - **WARNING: Only cells with the same cell_footprint name can be swapped. For more details about swapping group, refer to slide “contest.lib File Example” in “ISPD_2012_Contest_Details.pdf”**
  - **WARNING: You must NOT use the function field in .lib file to determine which cells can be swapped.**

Contest Evaluation

- Two separate rankings:
  - Primary ranking: Solution quality will be the main metric. Runtime will be used for tie-breaking.
  - Secondary ranking: Both solution quality and runtime will be important. Multi-core implementations are encouraged!

- There will be a hard runtime limit for each benchmark

Violations

- Violations are the primary ranking criteria for both rankings
- Violations are divided into three different types
  - Negative slack (ps)
  - Maximum capacitance (fF)
  - Slew (ps)
- All violations are added together into a single number
- All benchmarks can be sized without any violations

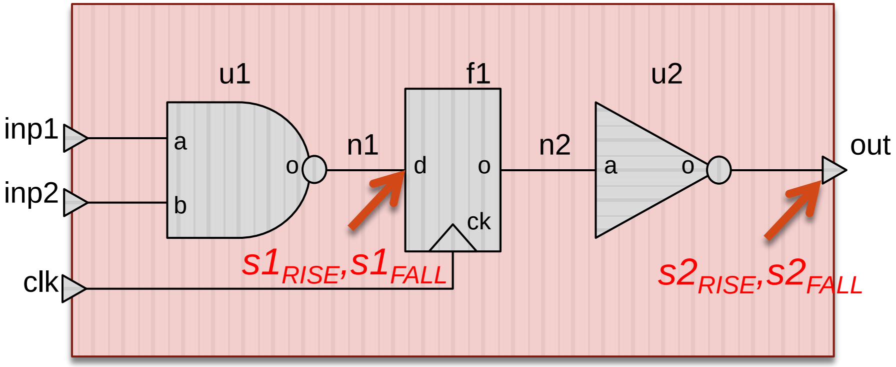
Violations

- Negative slack violations:
  - Measured for both rising and falling transitions at the primary outputs and sequential inputs
  - See contest details slides for slack computation
  - In the simple.v example the slack variables that have to be observed are:

$s1_{RISE}$ = 10ps
$s1_{FALL}$ = -3ps
$s2_{RISE}$ = 1ps
$s2_{FALL}$ = -5ps

Total slack violation = ∑negative slacks
Total slack violation = 3+5 = 8 ps

Simple.v circuit



Violations

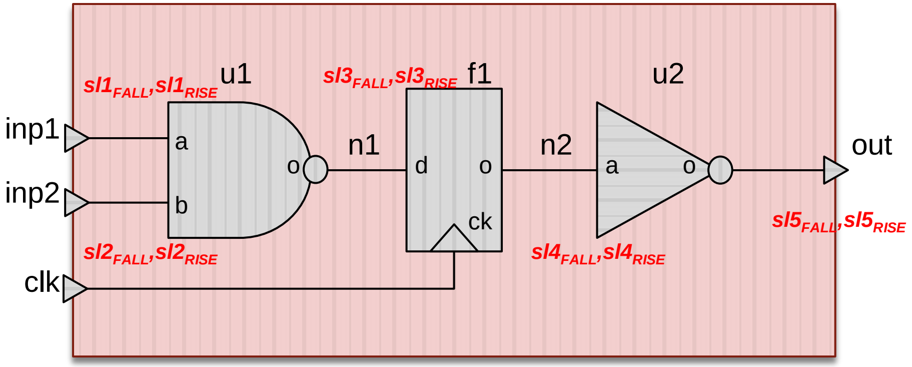
- Slew violation:
  - Measured on input pins for all cell instance and primary outputs (for both rise and fall transitions)
  - Slew limit is defined by the default_max_transition field in the .lib file

Slew violations:
Slew_limit= 100 ps

$sl1_{FALL}$ = 97ps	$sl1_{RISE}$ = 95ps
$sl2_{FALL}$ = 99ps	$sl2_{RISE}$ = 103ps
$sl3_{FALL}$ = 101ps	$sl3_{RISE}$ = 112ps
$sl4_{FALL}$ = 75ps	$sl4_{RISE}$ = 89ps
$sl5_{FALL}$ = 42ps	$sl5_{RISE}$ = 52ps

Total slew violation = 16ps

Simple.v circuit



$sl1_{FALL}$,$sl1_{RISE}$
u1
$sl3_{FALL}$,$sl3_{RISE}$
f1
u2
inp1
a
n1
n2
o
d
o
a
o
out
inp2
b
ck
$sl5_{FALL}$,$sl5_{RISE}$
$sl2_{FALL}$,$sl2_{RISE}$
$sl4_{FALL}$,$sl4_{RISE}$
clk

Violations

- Output capacitance per cell:
  - Will be measured once per cell instance output
    - Includes driving cells for primary inputs as defined in the .sdc file
  - *Violation = max(0,(cap(output_net)+cap(fanout) - max_cap(cell.output_pin)))*
  - The maximum capacitance allowed for the output pin of every cell is defined in the .lib file

Pin and net caps:

cap(n1)=100fF
cap(inp1)=27fF
cap(u1.a)=12fF
cap(f1.d)=26fF
cap(out)=126fF

cap(n2)=53fF
cap(inp2)=20fF
cap(u1.b)=12fF
cap(u2.a)=32fF

Maximum capacitance allowed per cell:

Max_cap(inp1)=60fF
Max_cap(inp2)=60fF
Max_cap(f1.o)=100fF
Max_cap(u2.o)=150fF

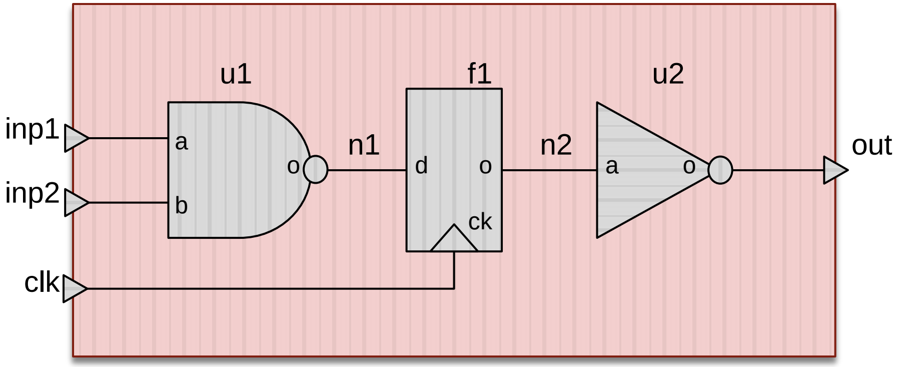
Maximum Capacitance violations:

Cap(inp1) = 20fF+12fF = 32fF
Cap(inp2) = 27fF+12fF = 39fF
Cap(u1.o) = 100fF+26fF = 126fF
cap(f1.o) = 53fF+32fF = 85fF
cap(u2.o) = 126fF

Total max_cap violation = 6fF

Jan-29, 2012

Simple.v circuit

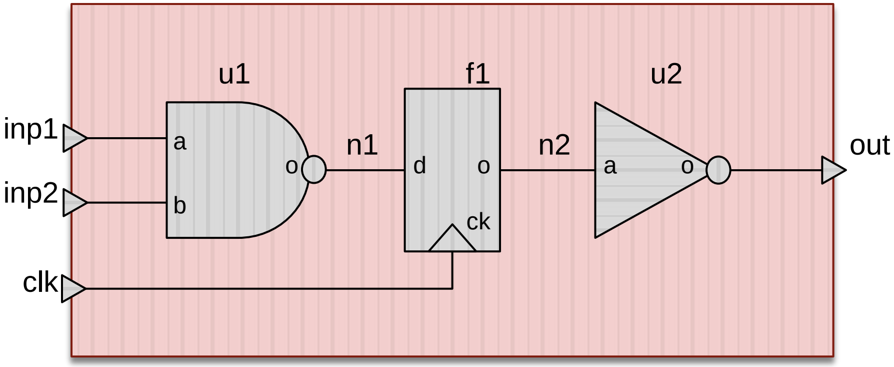


Power

- Only leakage power is considered
- The leakage power value for each cell is given in the .lib file
- Total leakage power value is given by the sum of the leakage power for each cell

Total Power computation
Power(u1)=143uW
Power(f1)=313uW
Power(u2)=50uW
Total Power = 134uW + 313uW + 50uW
Total Power = 497uW

Simple.v circuit



Runtime

- The runtime is computed for each benchmark
- Time to load the design in PrimeTime in the very beginning is not included
  - Circuit is loaded only once in PrimeTime before the runtime starts to be measured
- Runtime is the wall clock time from the beginning to the end of the execution of the submitted binary
  - Runtime includes not only the submission execution time but the time taken to complete any PrimeTime call
  - If no calls are made to PrimeTime runtime will include only the submitted binary execution time
- All jobs running after the runtime limit is reached will be killed

Runtime Limit

- The runtime limit will be defined **per benchmark**
  - Runtime limit is defined based on the number of cells for each benchmark according to the equation:

$$Runtime = 5h + 1h \times Roundup\left(\frac{\#\,gates}{35K}\right)$$

  - Runtime limit is determined by 5 hours plus 1 hour per each 35K gates for each design
  - The number of gates divided by 35K will be rounded up to the next integer number
    - i.e., 2.6 is rounded up to 3, 4.1 is rounded up to 5
  - Runtimes are given in hours

Primary ranking: Quality

- The ranking metric for a benchmark is defined in lexicographic order as:
  - First: ∑violations
  - Second: ∑power (when violations are tied)
  - Third: Runtime (when violations and power are tied)
- Sum of the ranks for each benchmark will define the final score for each team
  - If there is a tie at the end, it is broken using the sum of violations, power and runtime over all benchmarks in the same lexicographic order (example in next slides)

Primary ranking: Quality

- Example: 3 teams, 2 hypothetical benchmarks
- Execution summary for each team on each benchmark
  - benchmark01:

Team	Violations				Total Power	Runtime
	Max Cap	Slew	Negative Slack	Sum		
Team01	0	0	0	0	12.6 mW	31m 42s
Team02	0	0	0	0	11.1 mW	1h 12m 21s
Team03	12fF	130ps	10ps	152	7 mW	10s

  - benchmark02:

Team	Violations				Total Power	Runtime
	Max Cap	Slew	Negative Slack	Sum		
Team01	10fF	0	0	10	5.4 mW	13m 14s
Team02	0	0	0	0	7 mW	15m 55s
Team03	0	10ps	0	10	5.4 mW	1m 49s

Primary ranking: Quality

- Below are the ranking for each benchmark and final ranking, the criteria used to decide on the ranking is highlighted on each case
  - Rankings for each benchmark

benchmark01				
Team	Violations	Power	Runtime	Rank
Team01	0	12.6 mW	31m 42s	#2
Team02	0	11.1 mW	1h 12m 21s	#1
Team03	152	7 mW	10s	#3

benchmark02				
Team	Violations	Power	Runtime	Rank
Team01	10	5.4 mW	13m 14s	#3
Team02	0	7 mW	15m 55s	#1
Team03	10	5.4 mW	1m 49s	#2

  - Final ranking

Final							
Team	benchmark01	benchmark02	Rank Sum	Total Violations	Total Power	Total Runtime	Final Rank
Team01	#2	#3	5	10	18 mW	44m 56s	#2
Team02	#1	#1	2	0	18.1 mW	1h 28m 16s	#1
Team03	#3	#2	5	162	12.4 mW	1m 59s	#3

Secondary ranking: Quality/Runtime

- The secondary ranking evaluates the solution that presents the best quality/runtime trade-off
- Violations are still the primary metric, all the solutions with the same number of violations are ranked by:

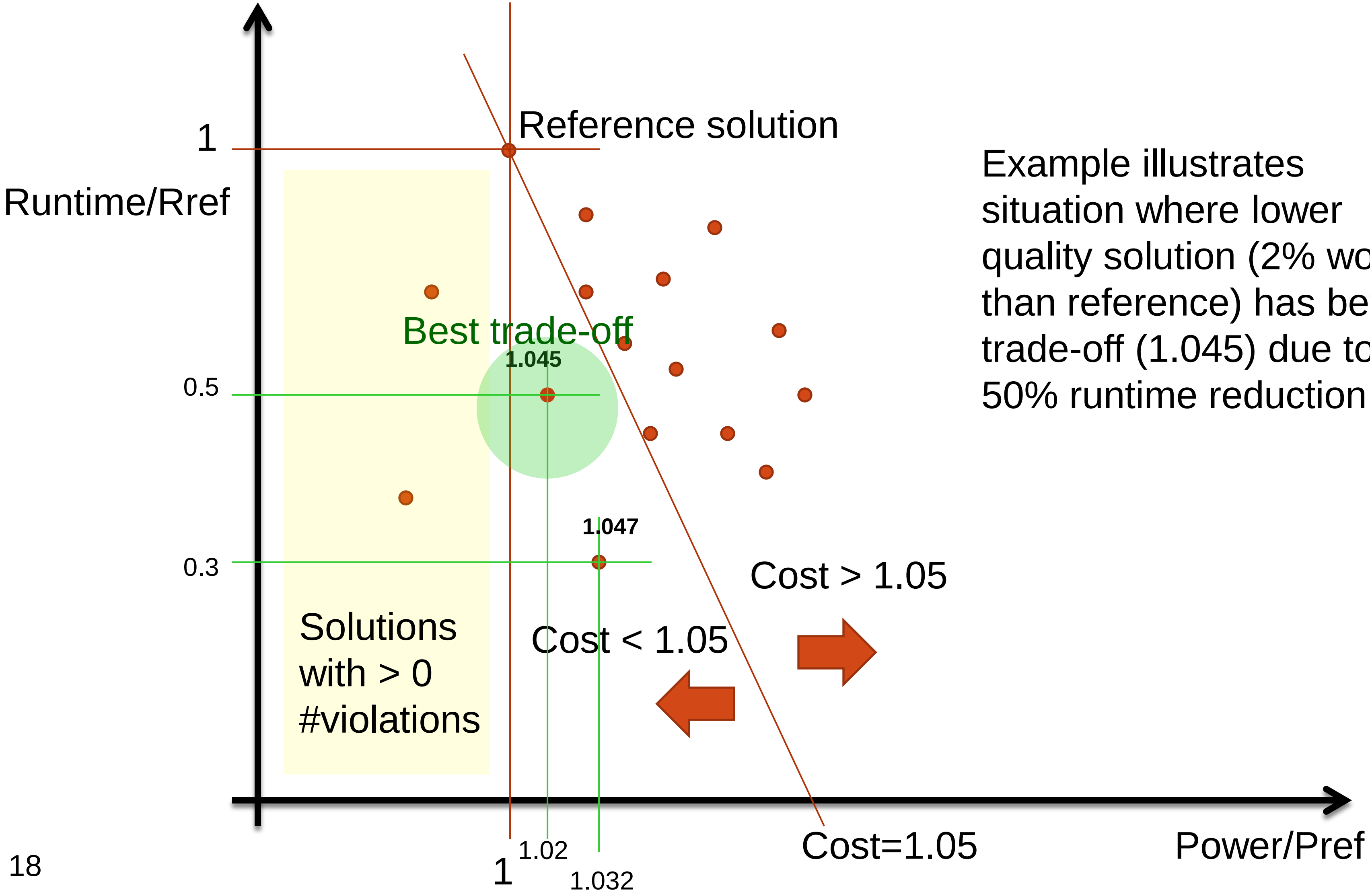
$$\cos t = \frac{Power}{Power_{REF}} + \gamma \frac{Runtime}{Runtime_{REF}}$$

- This metric trades quality by runtime improvement with respect to a reference power and runtime values
- If there are ties those will be broken using the same criteria applying to the Primary metric (Violations, Power, Runtime)

Secondary ranking: Quality/Runtime

- Power$_{REF}$ and Runtime$_{REF}$ are going to be defined by the best quality solution (according to primary ranking metric) for each benchmark
  - If multiple submissions have the same best power value, the smallest runtime among them will be used as reference
- Gamma is 0.05
  - e.g., 1% degradation in the solution quality can be compensated by a 20% runtime reduction w.r.t reference values

Secondary ranking: Quality/Runtime



Example illustrates situation where lower quality solution (2% worse than reference) has better trade-off (1.045) due to 50% runtime reduction

Secondary ranking: Quality/Runtime

- Rankings for each benchmark. Orange cells indicate reference values used for trade-off computation. Yellow cells are the values used to decide ranking positions.

benchmark01					
Team	Violations	Power	Runtime	Trade-off	Rank
Team01	0	12.6 mW	31m 42s	1.16	#2
Team02	0	11.1 mW	1h 12m 21s	1.05	#1
Team03	152	7 mW	10s	0.63	#3

benchmark02					
Team	Violations	Power	Runtime	Trade-off	Rank
Team01	10	5.4 mW	13m 14s	0.81	#3
Team02	0	7 mW	15m 55s	1.05	#1
Team03	10	5.4 mW	1m 49s	0.78	#2

Final							
Team	benchmark01	benchmark02	Rank Sum	Total Violations	Total Power	Total Runtime	Final Rank
Team01	#2	#3	5	10	18 mW	44m 56s	#2
Team02	#1	#1	2	0	18.1 mW	1h 28m 16s	#1
Team03	#3	#2	5	162	12.4 mW	1m 59s	#3

Same ranking

- If 2 teams have the same ranking:
  - Both will have the rank equal to the number of teams that have a better ranking + 1
  - The teams following those will also be ranked according to the number of teams that have a better ranking

Example:
TeamB and TeamC are tied
TeamE, TeamF and TeamG are also tied

All tied teams have the same ranking
The ranking of a team is always equal to the number of teams in front of it in the ranking

Rank	Team
1	TeamA
2	TeamB
2	TeamC
4	TeamD
5	TeamE
5	TeamF
5	TeamG

Error handling

- During timing iterations when using PrimeTime, any invalid cell swapping will be ignored by PrimeTime (i.e. cell size from previous iteration will continue to be used)
- If the .sizes files is missing, the final evaluation will be done using the original cell sizes from verilog file
  - Any missing cell in the .sizes file will have its size unaltered
    - If cell C is not included in the final .sizes file, then the original size of C in the verilog file will be assumed
- For final evaluation, any invalid cell swapping will be ignored and the original cell size from the verilog file will be used
  - Some examples of invalid cell swapping are:
    - An invalid cell size in the .sizes file
    - Swapping cells with different cell_footprint names

Scripts

- Evaluation scripts will be provided.
- Scripts will:
  - Compute #violations
  - Compute power
  - Check solution correctness