A Metal-Only-ECO Solver for Input-Slew and Output-Loading Violations

Chien-Pang Lu, Mango C.-T. Chao, Chen-Hsing Lo, and Chih-Wei Chang

Mstar Semiconductor, ChuPei, Taiwan

Dept. of EE, Nat'l Chiao Tung Univ., Hsinchu, Taiwan

Outline

- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path's delay)
- Experimental result
- Conclusions

Metal-Only ECO

- The increasing pressure of time-to-market has forced IC design houses to improve capability of handling incremental design changes
- Those design changes are often requested after silicon chips are manufactured
  - its photomasks need to be changed
- Solution: metal-only ECO
  - change only the metal layers (for interconnect) while the base layers (for cells) remain the same
  - reduce cost by reusing base-layer photomasks
  - shorten tape-out turn-around time

EDA Tools Needed in Metal-Only ECO

- Allocate spare cells all over a chip
  - EDA vendors already provide effective solutions
- Obtain netlist difference and implement the difference
  - EDA vendors already provide effective solutions
- A router dealing with a lot obstacles
  - EDA vendors already provide effective solutions
- Solve violations of timing-related factors, such as setup time, input slew, and output loading
  - However, vendor's solutions are not effective so far

Outline

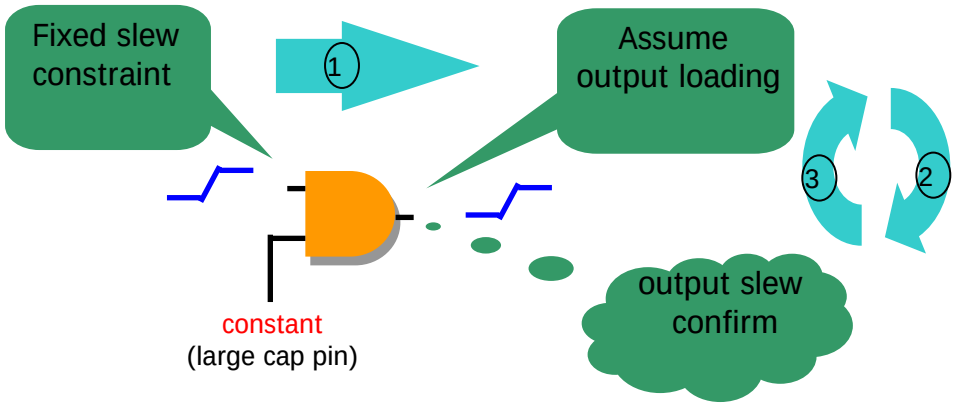
- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path's delay)
- Experimental result
- Conclusions

Problem Formulation of Proposed Work

- Given:
  - Input-slew and output-loading constraints
  - Nets violating the constraints after the design changes are implemented
  - Available spare cells
- Objective
  - Insert fewest spare cells as buffers to eliminate the violations
- Use a commercial APR tool to realize the buffer insertions
- Focus on how to select proper spare cells and estimate the added wire loading when inserting the buffers by the adopted APR tool

Transfer Input-slew Constraint into Equivalent Output-loading Constraint

- OAL_g : Output Available Loading
  - The maximum output loading of gate g which can generate an output slew smaller than the slew constraint assuming that g's input slew is equal to slew constraint
- Obtaining OAL_g for each type of gate
  - Binary search, table look-up
    - Ex: target input slew constraint, 500ps



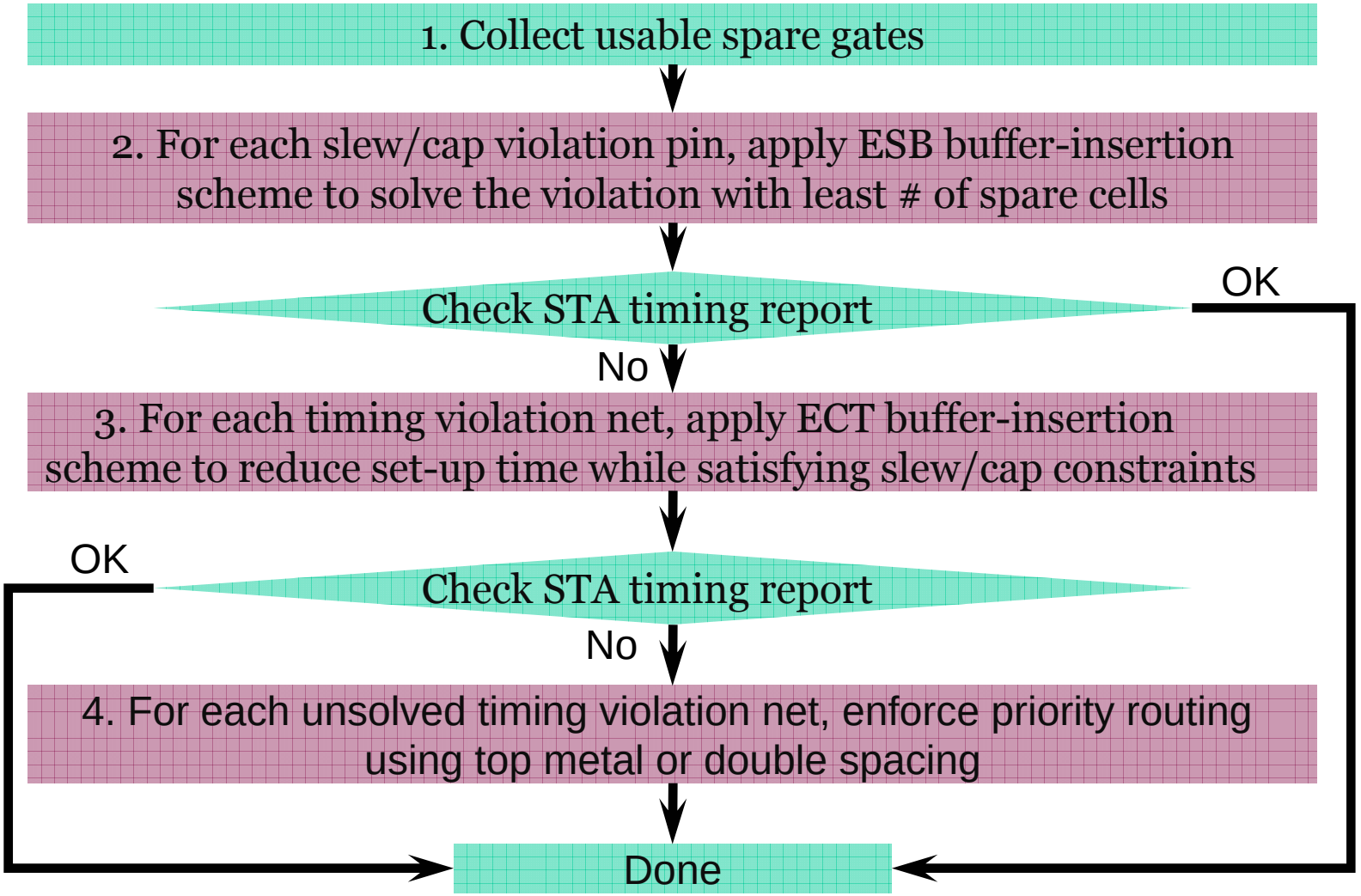
Iteration	input slew	output load	output slew
1	500p	4000ff	2000p
2	500p	1500ff	400p
3	500p	1600ff	520p
4	500p	**1540ff**	**500p**

OAL(1540ff): under 500p input slew

Outline

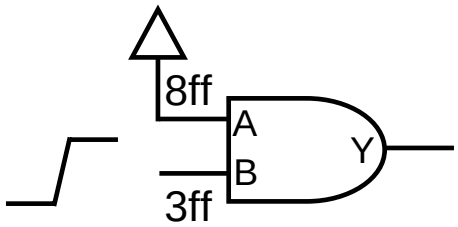
- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path's delay)
- Experimental result
- Conclusions

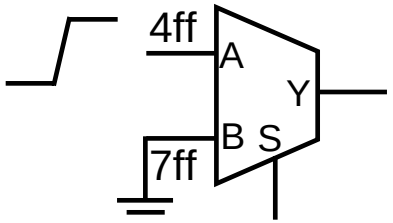
Overall Flow of MOESS



Increase Spare-Buffer Pool

- Recycle of redundant cells
  - APR tools use special tags to identify spare buffers
  - Tags may be lost by engineer's incorrect operation
  - MOESS applies a breadth-first search starting from each floating output to recycle the lost-tag gates
- Function cells as buffers by connecting the other inputs to a constant





Wire-Loading Estimation for a Two-Terminal Net

- Use a net's Manhattan distance (MD) to estimate its wire loading (WL)
- $WL(p1,p2)$

$$= MD_h(p1,p2) * RRMD_h(VD(p1,p2) * K_h + MD_v(p1,p2) * RRMD_v(VD(p1,p2) * K_v$$

Manhattan distance

routing ratio to Manhattan distance

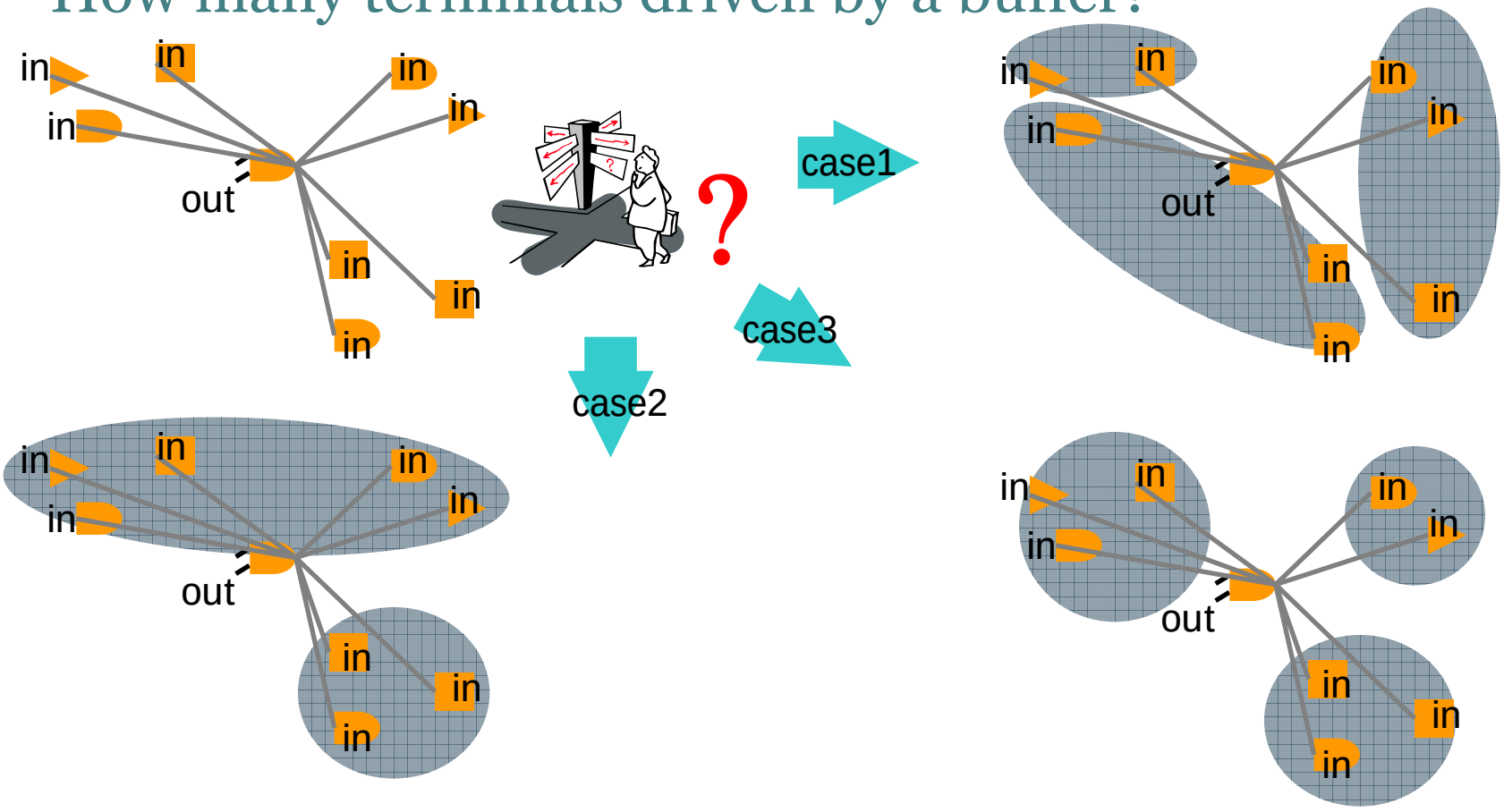
# of vias over rectangle area formed by p1 and p2

Wire loading constant per routing unit

This function is actually the average statistics collected from the past usage of the adopted ARP tool

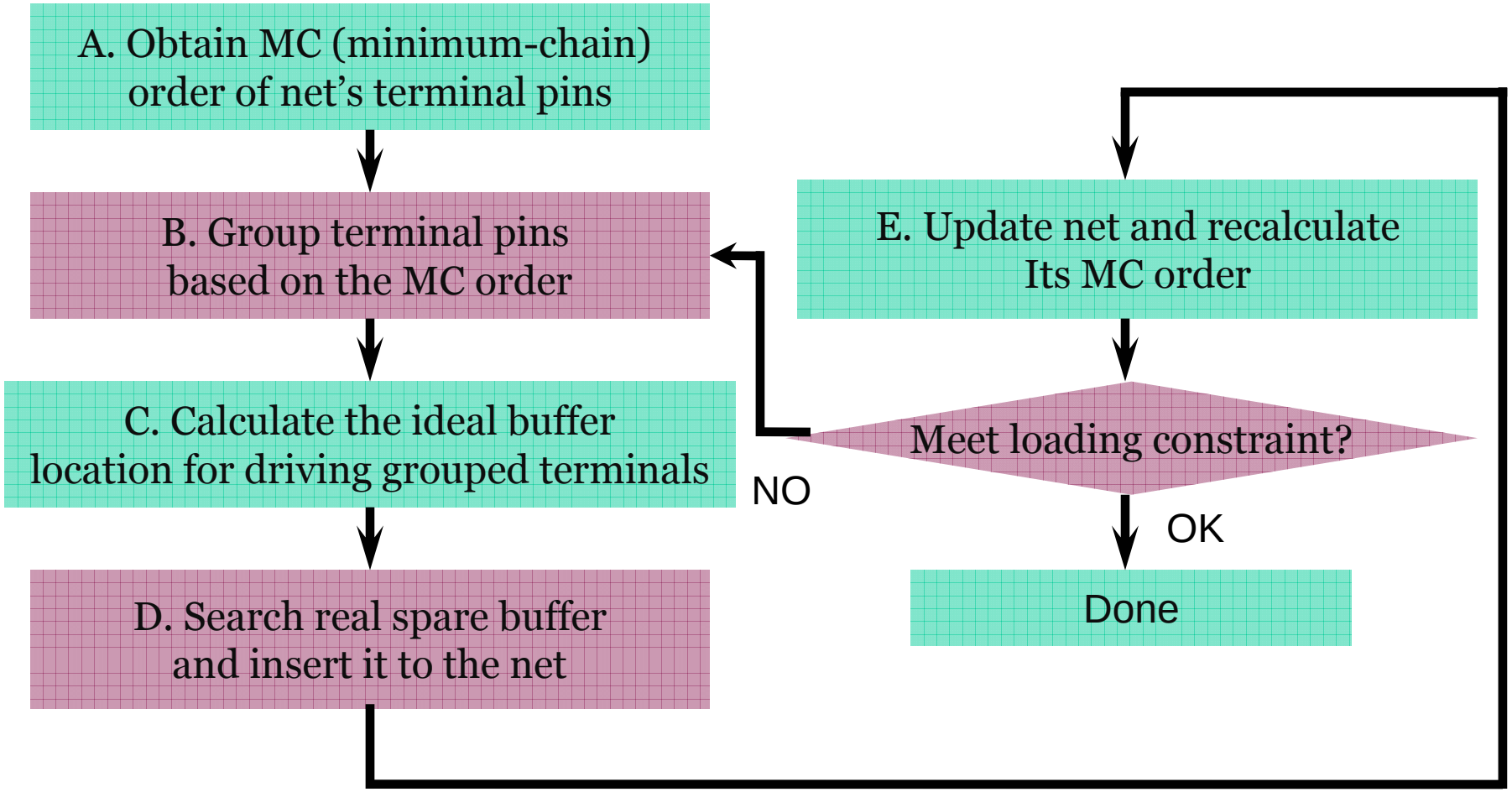
Solving Violation for a High-Fanout Net

- How to use fewest buffers to solve a violation?
  - How many terminals driven by a buffer?



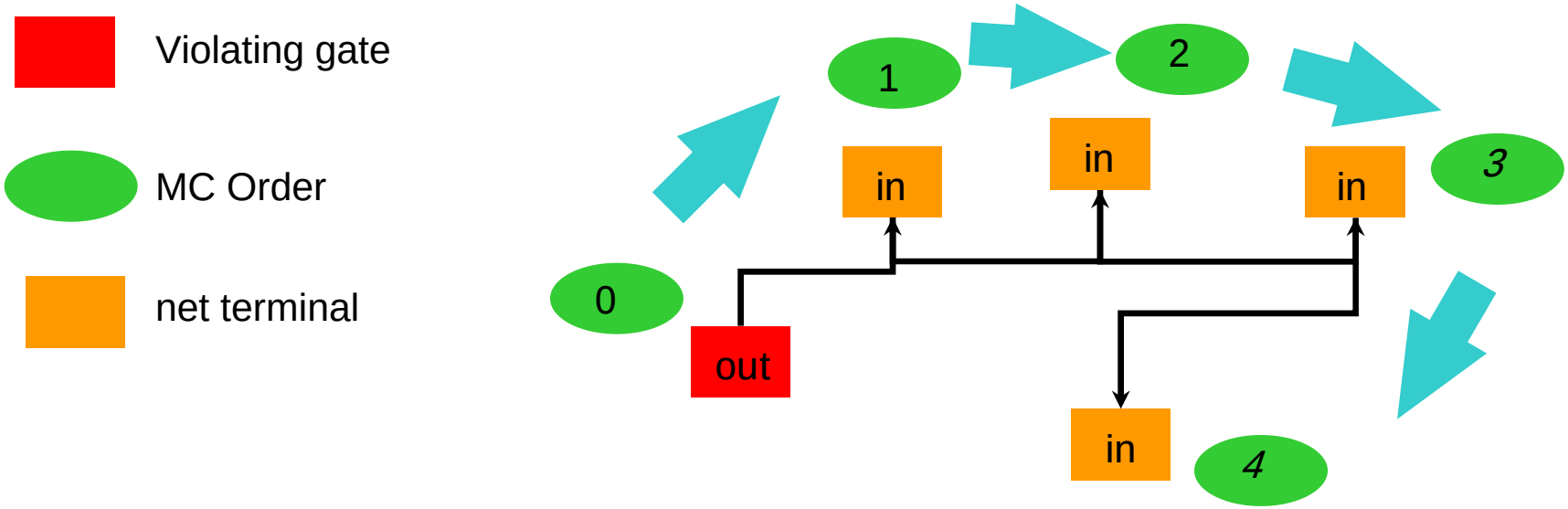
Flow of ESB Buffer-Insertion Scheme

(Use fewest buffers to solve the violation)



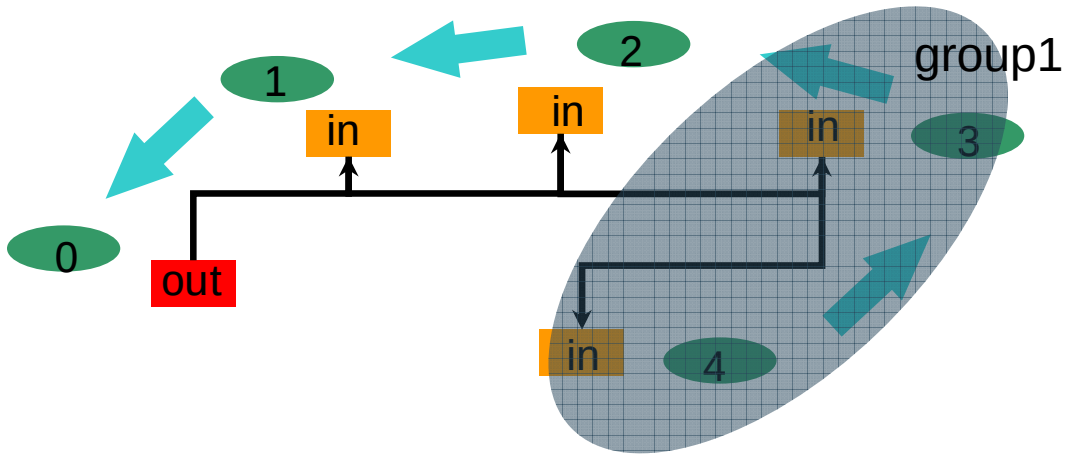
Minimum-Chain Order of a Net's Terminals

- Algorithm:
  - Start from violation gate
  - Select the closest terminal as the next ordered terminal until all terminal are ordered



Group Terminals

- Group the terminals based on the reversed MC order
- Each time add one terminal into the group
- Stop when adding the new terminal would exceed gate's OAL (slew/loading constraint)
  - $\sum_{i=1}^{n}(InC_{p_i} + WL(p_i, p_{i-1})) < OAL_g$
- Use a buffer to drive as many terminals as possible



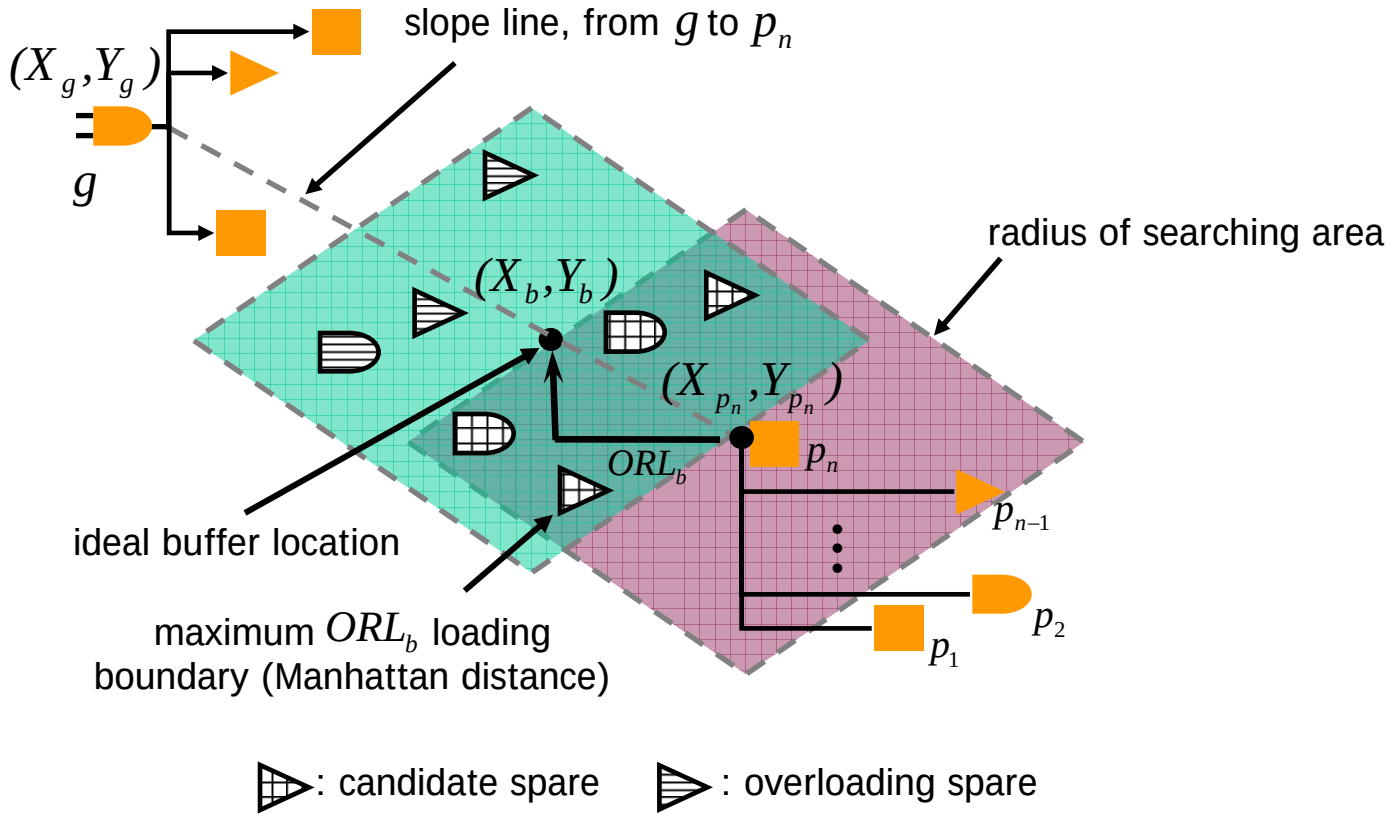
Calculate Ideal Location of Inserted Buffer

- Two rules when deciding ideal buffer's location
  - *R1* : Use all buffer's driving capability under the given constraint
    - $|X_b-X_p|*U_h(b,p_n)+|Y_b-Y_p|*U_v(b,p_n)\leq ORL_b$
    - ORL - Output Remain Loading
      - $ORL_b = OAL_b - \sum_{i=1}^{n}(InC_{p_i} + WL(p_i, p_{i-1}))$
    - U_h and U_v are vertical and horizontal distance per loading unit
  - *R2* : Locate the inserted buffer as close to the violation gate as possible
    - $(Y_b-Y_p)/(X_b-X_p)=(Y_p-Y_g)/(X_p-X_g)$
    - Limit the ideal location between g and p_n

g: output
b: ideal buffer
pn: overloading group terminal closest to g

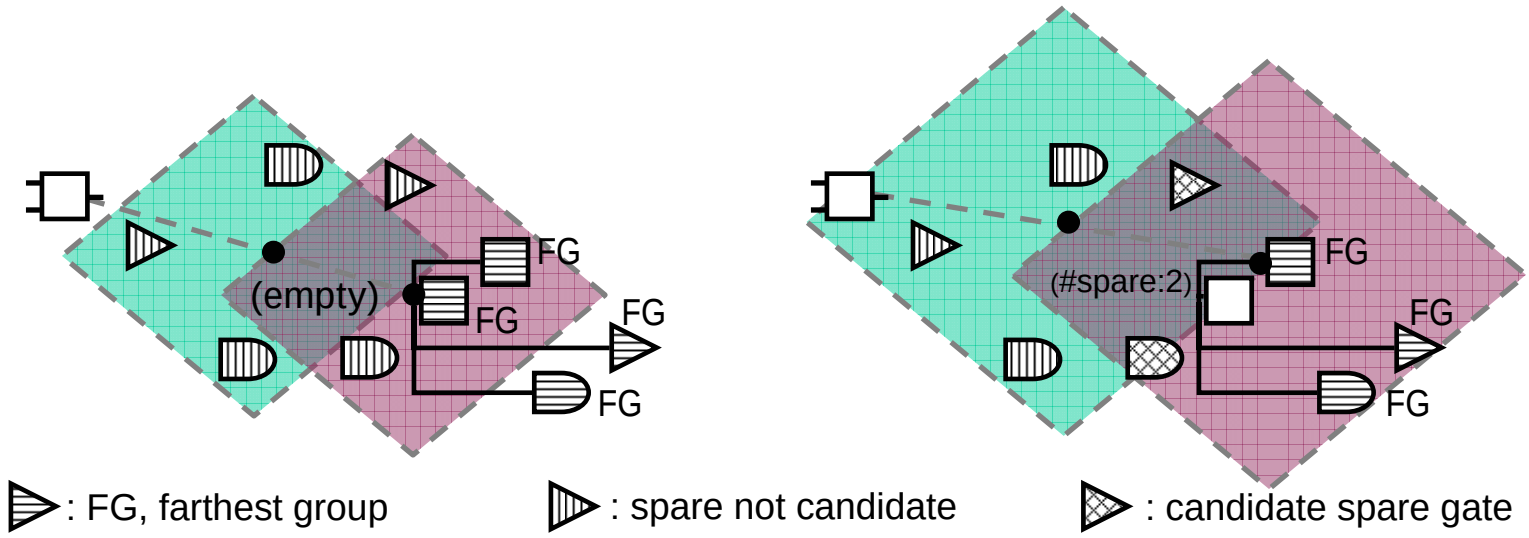
Find Spare Buffer near Ideal Location

- Use ORL_b as the radius to draw the boundary of searching feasible spare buffers

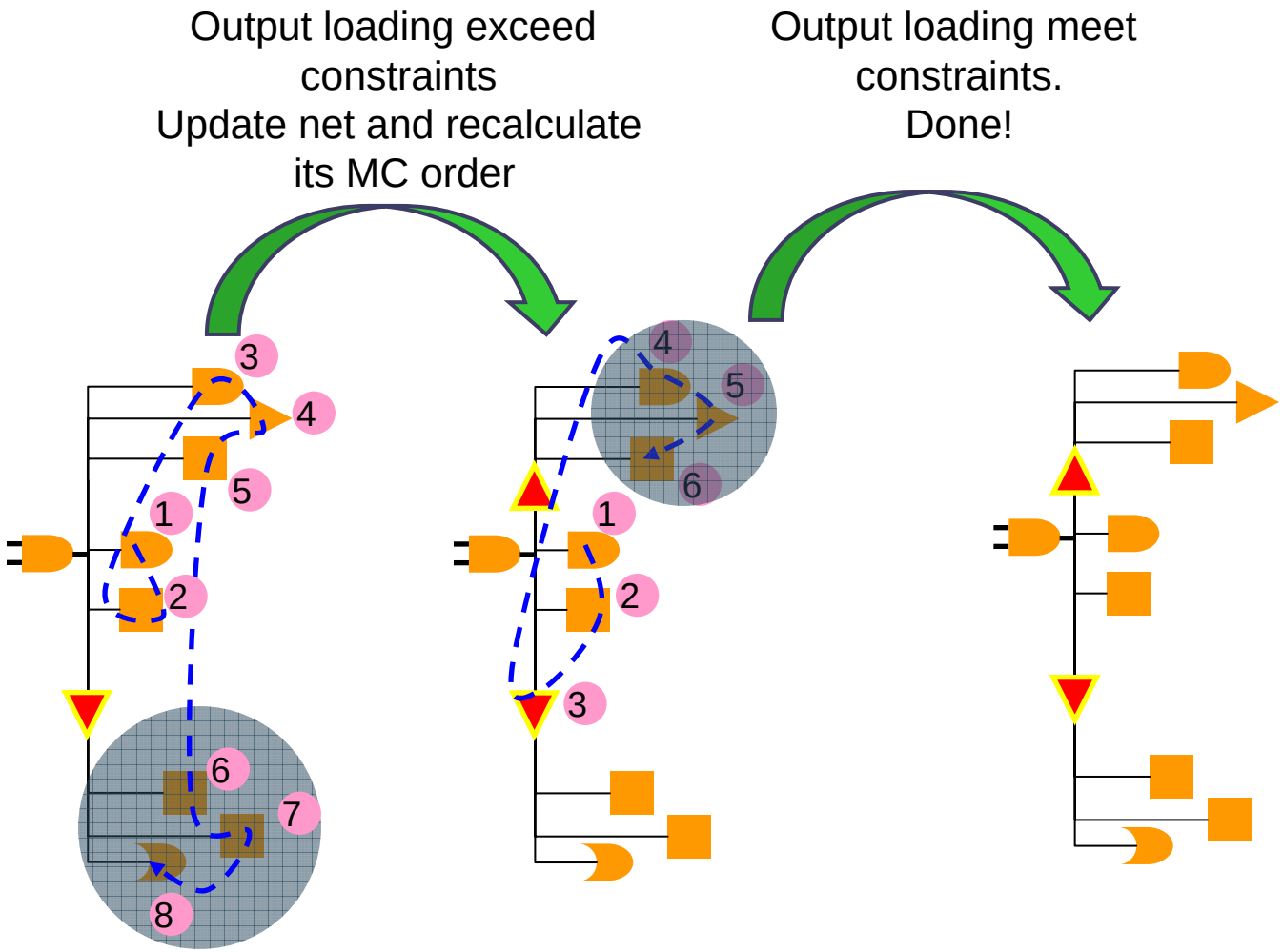


Backward Tolerance: Enlarge Searching Space

- Ungroup the last grouped terminal to increase the *ORL* and in turn the radius of the search space
  - Keep on ungrouping until a spare gate is found

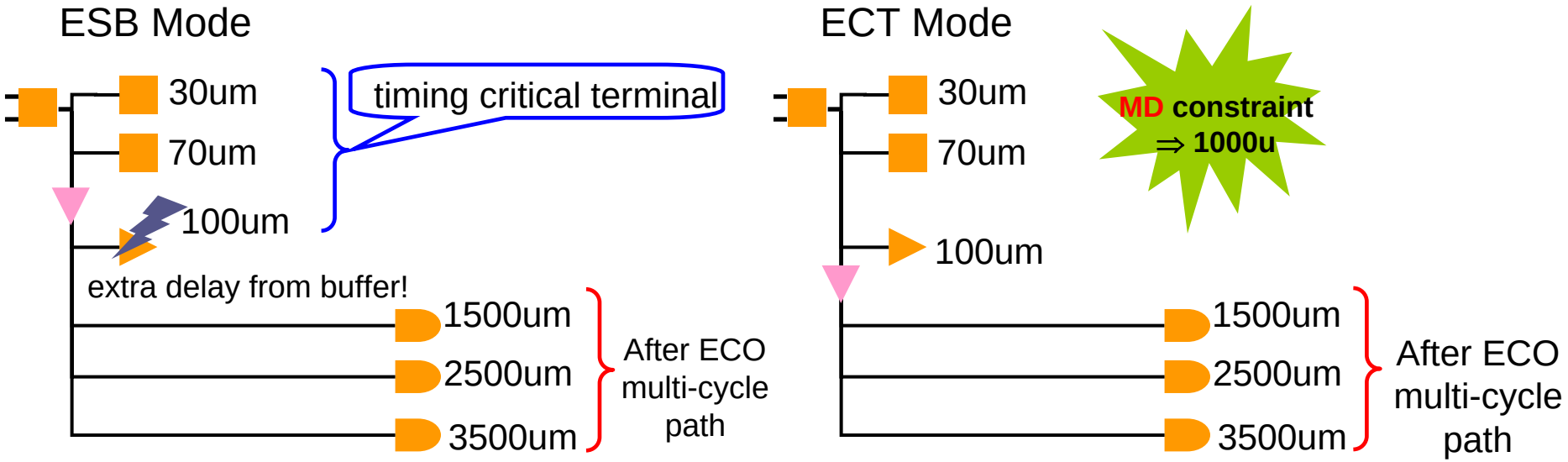


Example


Output loading exceed constraints
Update net and recalculate its MC order
Output loading meet constraints.
Done!
1
2
3
4
5
6
7
8

ECT Mode (reduce critical path's delay)

- Separate the grouping of original terminals from the grouping of new-added terminals far away from the violation output
  - Group the distant, new-added terminals first



Outline

- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path's delay)
- Experimental result
- Conclusions

Design Information

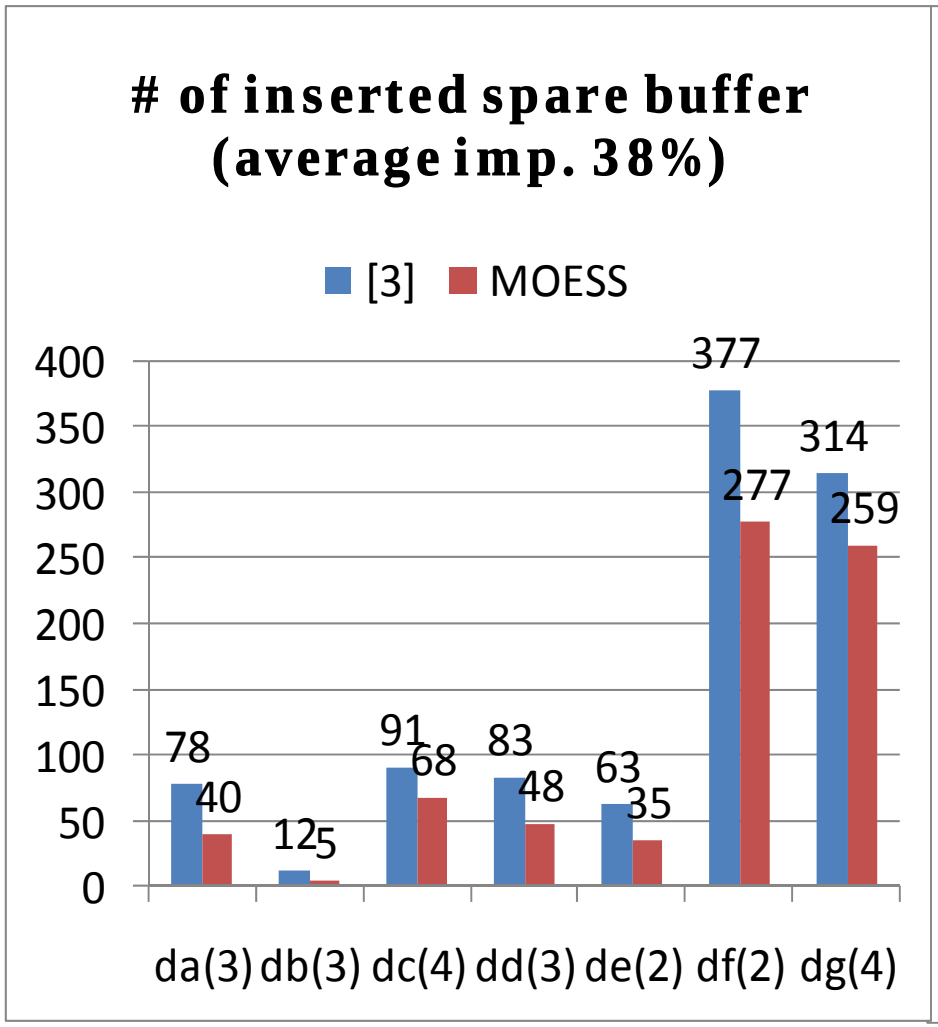
Proj. (ver.)	inst. count	process	spare count	ECO size	violation	
					slew	load
Da(3)	190K	.18	7.6K	142	40	0
Db(3)	210K	.18	9.1K	1030	6	0
Dc(4)	242K	.18	5.5K	507	71	0
Dd(3)	309K	.18	10.4K	1904	47	0
De(2)	871K	.13	62.4K	127	0	35
Df(2)	1.3M	.13	48.8K	1276	15	243
Dg(4)	1.6M	.13	80.5K	1702	166	258

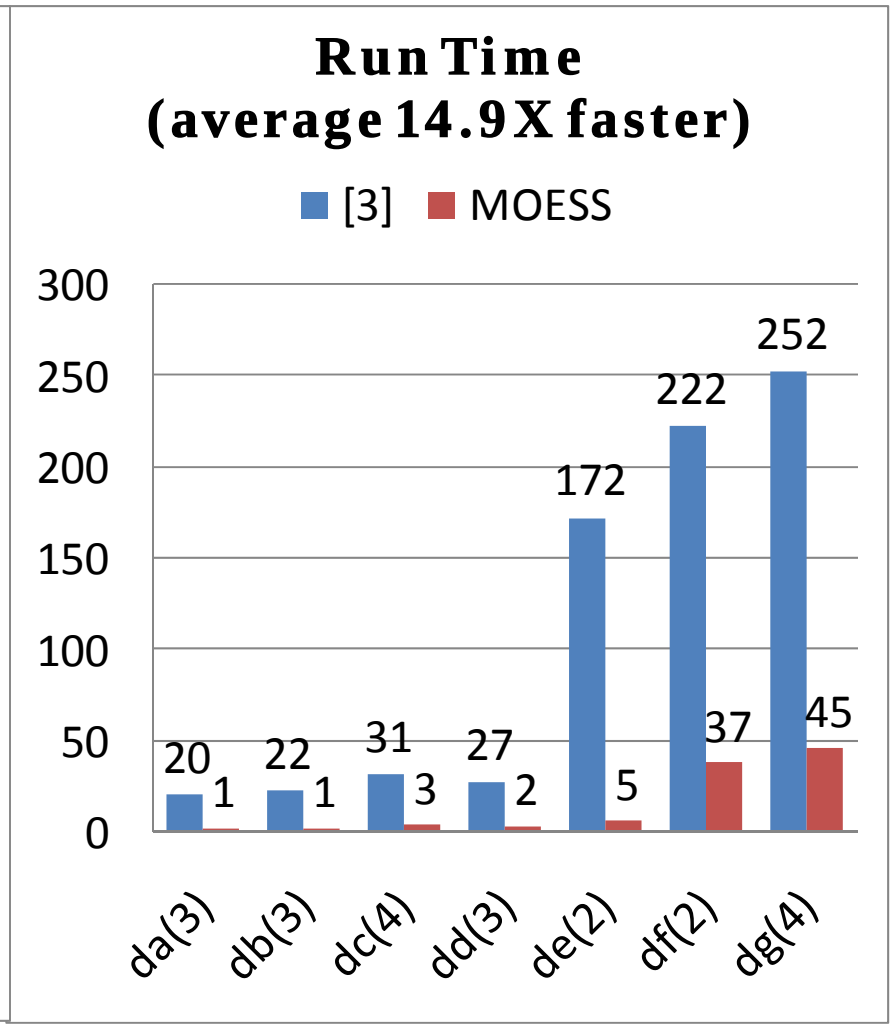
Experiment Result

Proj. (ver.)	worst slew		worst loading		worst slack	
	[3]	MOESS	[3]	MOESS	[3]	MOESS
Da(3)	5.0n	1.9n	<1	<1	-2.2n	>0
Db(3)	1.8n	1.8n	<1	<1	>0	>0
Dc(4)	3.8n	2.0n	<1	<1	-0.3n	>0
Dd(3)	2.1n	2.0n	<1	<1	>0	>0
De(2)	0.9n	0.9n	1.2	1.1	>0	>0
Df(2)	1.3n	0.9n	3.5	1.2	-0.1n	>0
Dg(4)	1.2n	1.0n	4.6	1.2	-0.4n	>0

means the result violates the constraint

Experiment Result





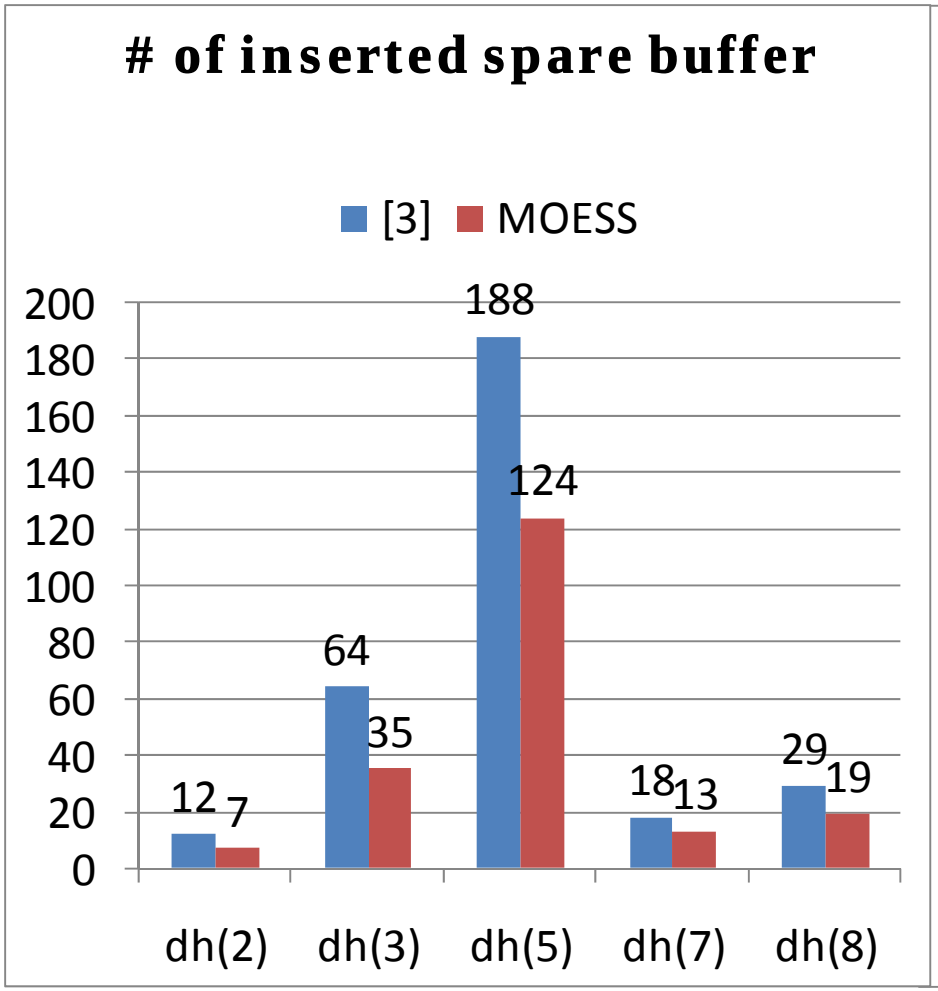
Experiment Result

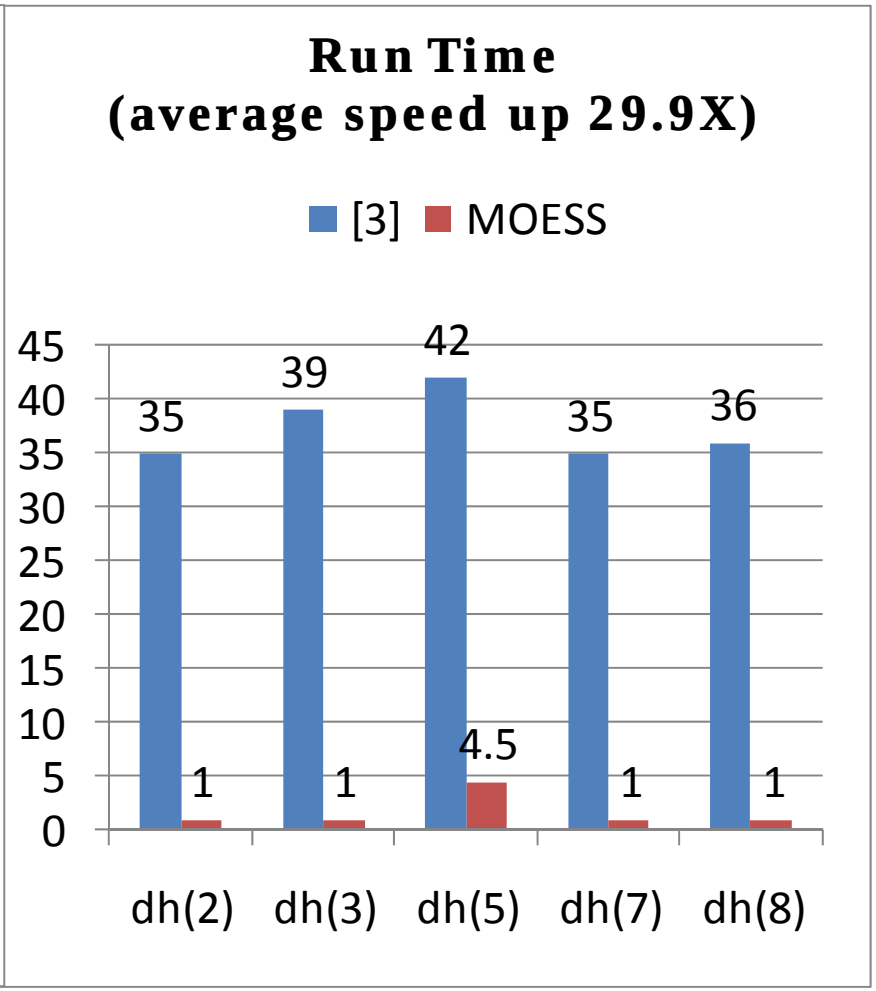
Proj. (ver.)	spare count	ECO size	#violation		worst slew		worst loading		worst slack	
			slew	load	[3]	MOESS	[3]	MOESS	[3]	MOESS
Dh(2)	4.2K	52	7	0	1.8n	1.8n	<1	<1	>0	>0
Dh(3)	4.1K	267	29	2	2.1n	1.8n	<1	<1	>0	>0
Dh(5)	3.7K	1672	118	5	4.6n	2.0n	1.3	<1	-1.4n	>0
Dh(7)	1.9K	43	9	2	2.8n	2.0n	<1	<1	-1.7n	-0.1n
Dh(8)	1.8K	135	17	3	3.5n	2.1n	<1	<1	-0.5n	-0.1n

# of instance count: 352.1K

means the result violates the constraint

Experiment Result





Conclusions

- An effective solver to solve the slew/loading violations generated in metal-only ECO
- Less # of spare gates in use
- Shorter runtime
- The proposed solver can be ported to other APR tools as long as the tools can provide open access to its design database

Thank you!

Solving Slew/Loading Violation in Metal-Only ECO

- Current commercial tools are not aware of the physical locations of spare gates
  - Available spare gates may deviate from its ideal location