Design Optimization by Fine-grained Interleaving of Local Netlist Transformations in Lagrangian Relaxation

Apostolos Stefanidis, Dimitrios Mangiras, Giorgos Dimitrakopoulos
Democritus University of Thrace, Greece

Chrystostomos Nicopoulos
University of Cyprus, Cyprus

David Chinnery
Mentor, a Siemens Business, USA

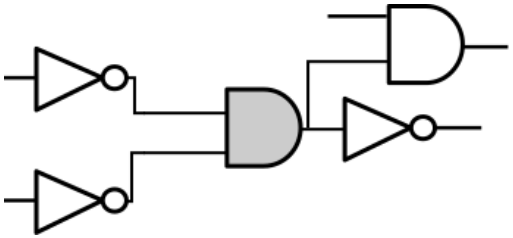
June 30, 2020

Outline

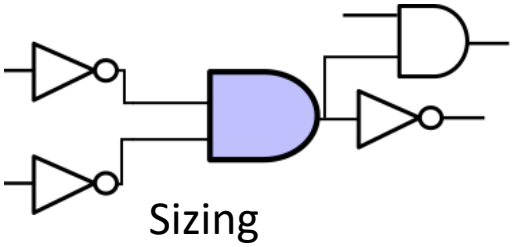
- Design optimization
- Timing-Power optimization using Langrangian relaxation (LR)
- Embedding multiple heuristics inside the same Multi Mode Multi Corner LR optimization loop
  - The criteria that each heuristic should satisfy to be compatible with LR-based optimization
  - Order of applying each heuristic
- Experimental results based on benchmarks of the TAU2019 contest
- Conclusions

Design optimization

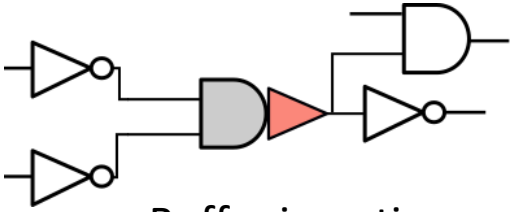
- Gate-level netlist changes to optimize:
  - Timing – fix early/late violations
  - Reduce leakage/dynamic power, area, wire length …
- Can be applied in any physical design step
  - Additional considerations (e.g. SI noise) + need for accuracy increase through the flow
- Examples:



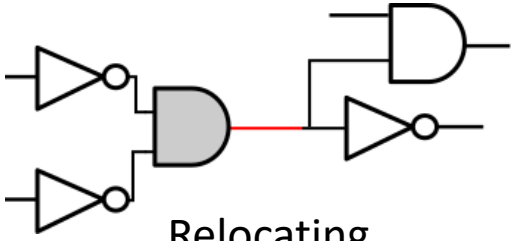
Initial circuit



Sizing



Buffer insertion



Relocating

Design optimization using Lagrangian relaxation

- Relaxes timing constraints into a simplified objective function
  - Lagrangian multipliers (LMs) weigh the constraints to try and ensure that they are met
- Already successfully applied for
  - Combinational gate sizing
  - Clock tree sizing
  - Timing driven incremental placement
- **Our proposal**: embed multiple optimization heuristics in the same Lagrangian relaxation optimization loop

Problem formulation

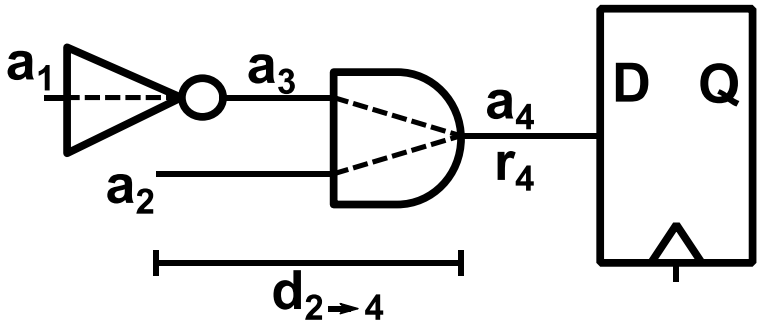
$$\min \sum_{c \in \text{cells}} P(c) + A(c) - \sum_{j \in \text{POs}} slk_j^L - \sum_{j \in \text{POs}} slk_j^E$$

$$\text{s.t.}: slk_j^L \leq 0 \text{ and } slk_j^E \leq 0, \quad \forall j \in \text{POs}$$

$$slk_j^L \leq r_j^L - a_j^L \text{ and } slk_j^E \leq a_j^E - r_j^E, \forall j \in \text{POs}$$

$$a_i^L + d_{i \to j}^L \leq a_j^L \text{ and } a_j^E \leq a_i^E + d_{i \to j}^E, \forall i \to j \in \text{arcs}$$

- **Target:** minimize sum of leakage power, area, and total negative slack (TNS)



$P(c)$: leakage of cell c
A(c): area of cell c
L: late timing information
E: early timing information
slk_j: negative slack of pin j
r_j: required time of pin j
a_j: arrival time of pin j
$d_{i \to j}$: delay of timing arc $i \to j$
arcs: timing arcs of the design
cells: cells of the design
POs: Primary outputs or timing endpoints of the design

Lagrangian relaxation formulation (1)

$$\min \sum_{c \in \text{cells}} P(c) + A(c) - \sum_{j \in \text{POs}} slk_j^L - \sum_{j \in \text{POs}} slk_j^E$$

$$\text{s.t.}: slk_j^L \le 0 \text{ and } slk_j^E \le 0, \quad \forall j \in \text{POs}$$

$$slk_j^L \le r_j^L - a_j^L \text{ and } slk_j^E \le a_j^E - r_j^E, \forall j \in \text{POs}$$

$$a_i^L + d_{i\to j}^L \le a_j^L \text{ and } a_j^E \le a_i^E + d_{i\to j}^E, \forall i \to j \in \text{arcs}$$

Langrangian relaxation

$$\min \sum_{c \in \text{cells}} P(c) + A(c) - \sum_{j \in \text{POs}} slk_j^L - \sum_{j \in \text{POs}} slk_j^E +$$

$$\sum_{j \in \text{POs}} \left(\lambda_{j0}^L slk_j^L + \lambda_{j0}^E slk_j^E\right) +$$

$$\sum_{j \in \text{POs}} \left(\lambda_{j1}^L (slk_j^L - r_j^L + a_j^L) + \lambda_{j1}^E (slk_j^E - a_j^E + r_j^E)\right) +$$

$$\sum_{i \to j \in \text{arcs}} \left(\lambda_{i\to j}^L (a_i^L + d_{i\to j}^L - a_j^L) + \lambda_{i\to j}^E (a_j^E - a_i^E - d_{i\to j}^E)\right)$$

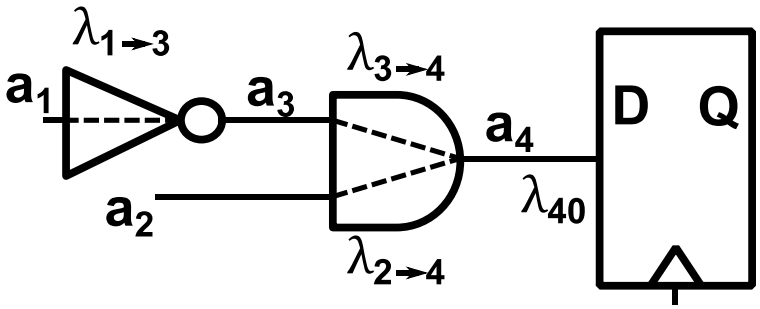
$\lambda_{j0}^L, \lambda_{j1}^L$: Late LMs for slack constraints on endpoints
$\lambda_{i\to j}^L$: Late LMs for early delay constraints on arcs

$\lambda_{j0}^E, \lambda_{j1}^E$: Early LMs for slack constraints on endpoints
$\lambda_{i\to j}^E$: Early LMs for early delay constraints on arcs

Lagrangian relaxation formulation (2)

- λ values represent the criticality of each constraint
- Karush-Kuhn-Tucker (KKT) optimality conditions
  $\sum_{\forall i \in inj} \lambda^L_{i\to j} = \sum_{\forall k \in \text{out}j} \lambda^L_{j\to k}$, $\sum_{\forall i \in inj} \lambda^E_{i\to j} = \sum_{\forall k \in outj} \lambda^E_{j\to k}$
- By applying the KKT conditions and simplifying:

$$\min \sum_{c \in \text{cells}} P(c) + A(c) + \sum_{i\to j \in \text{arcs}} \lambda^L_{i\to j} d^L_{i\to j} - \lambda^E_{i\to j} d^E_{i\to j}$$



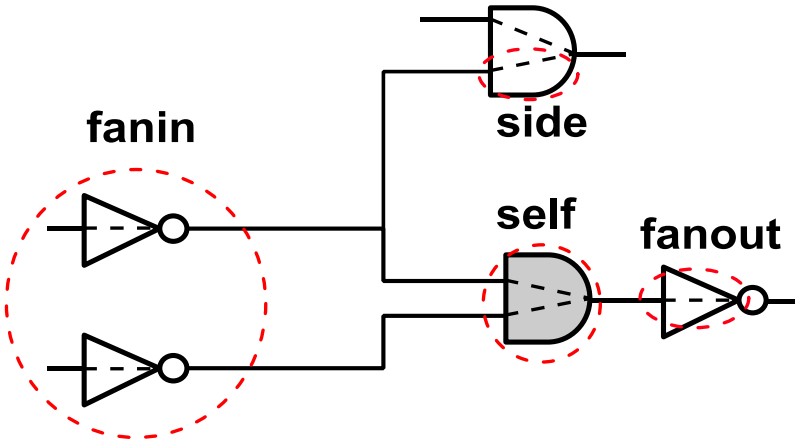
$$\lambda^L_{40} = \lambda^L_{3\to 4} + \lambda^L_{2\to 4} \qquad \lambda^E_{40} = \lambda^E_{3\to 4} + \lambda^E_{2\to 4}$$
$$\lambda^L_{1\to 3} = \lambda^L_{3\to 4} \qquad \lambda^E_{1\to 3} = \lambda^E_{3\to 4}$$

Lagrangian multiplier updates

- Timing arc $i \rightarrow j$ affects cost function by: $\lambda^L_{i\rightarrow j} d^L_{i\rightarrow j} - \lambda^E_{i\rightarrow j} d^E_{i\rightarrow j}$
  - High late LM ⇒ delay should decrease ⇒ late critical arc
  - High early LM ⇒ delay should increase ⇒ early critical arc
- Update method:
  - $\lambda^L_{i\rightarrow j} = \lambda^L_{i\rightarrow j} \frac{(a^L_i + d^L_{i\rightarrow j})}{a^L_j}, \quad \lambda^L_{j0} = \lambda^L_{j0} \frac{a^L_j}{r^L_j},$
  - $\lambda^E_{i\rightarrow j} = \lambda^E_{i\rightarrow j} \frac{a^E_j}{(a^E_i + d^E_{i\rightarrow j})}, \quad \lambda^E_{j0} = \lambda^E_{j0} \frac{r^E_j}{a^E_j},$
- LM values are propagated backwards proportionally to respect KKT optimality conditions

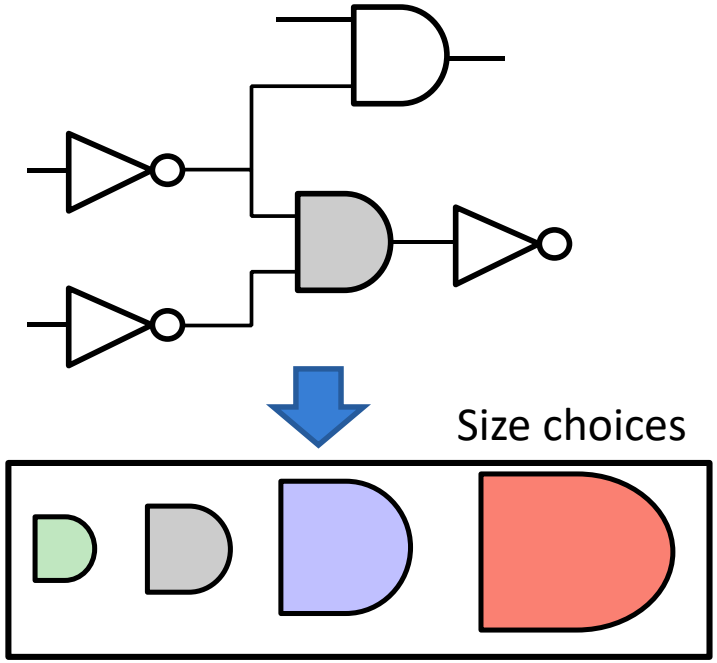
Local LR cost

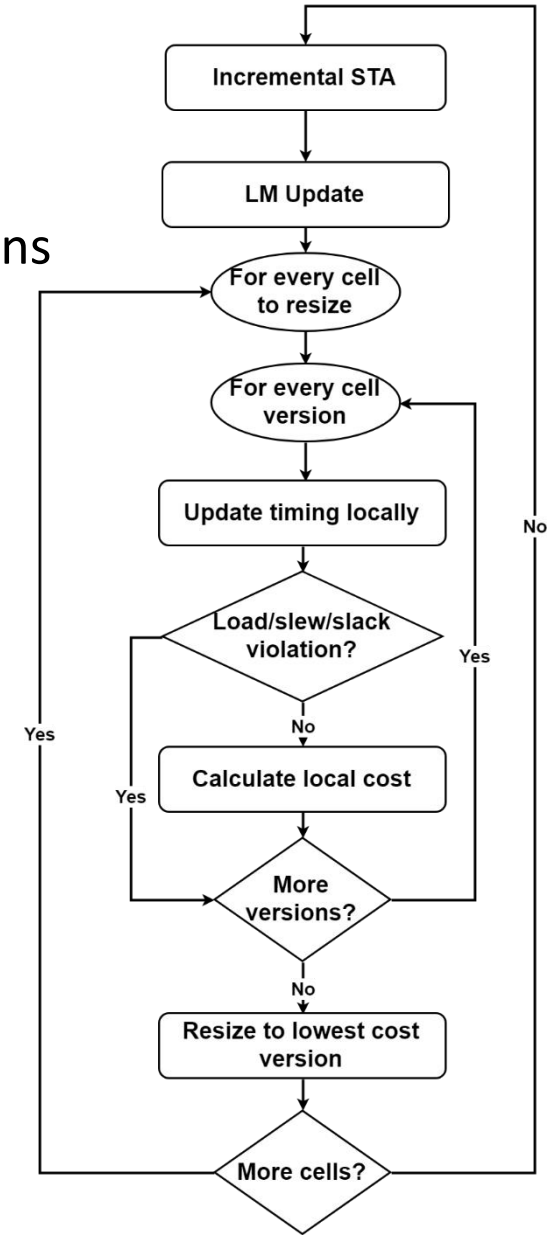
- Recalculate only timing information around the cell's local arcs
  - Calculating the cost function for every timing arc of the design is avoided to save runtime
  - $LC(v) = P(v) + A(v) + \sum_{i \to j \in local\_arcs} \lambda^L_{i \to j} d^L_{i \to j} - \lambda^E_{i \to j} d^E_{i \to j}$



How LR optimization loop works: Gate sizing example

- Make decisions based on local information
  ⇒ timing is updated only on local arcs
- Have discrete choices ⇒ different size / Vt options
- Evaluate each choice using LM values
  ⇒ pick the choice with the lowest local cost





Incorporating design transformations in the LR loop

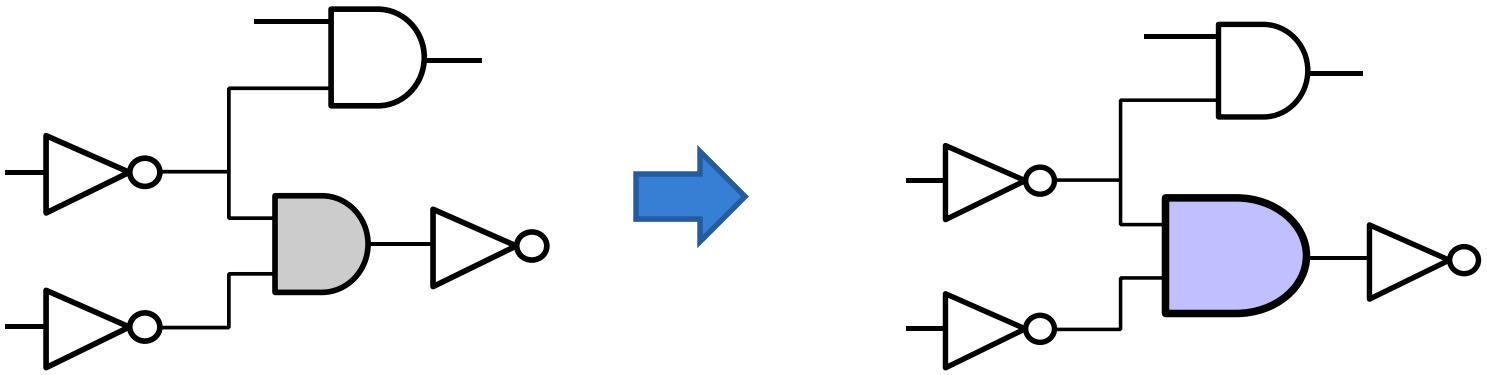
- Any transformation satisfying certain criteria can be applied inside the Lagrangian relaxation
- The method has to:
  - Make decisions based on local information
  - Have discrete choices
  - Evaluate each choice using LM values and the same local cost function
  - Apply small changes each iteration ⇒ allows LR to adapt to the change

LR design transformations in this work

- In this work we apply five transformations inside the LR-based optimization loop:
  - Cell sizing
  - Pin swapping
  - Buffering for early violations
  - Buffering for late violations
  - Clock skew assignment
- All make local decisions based on LM values

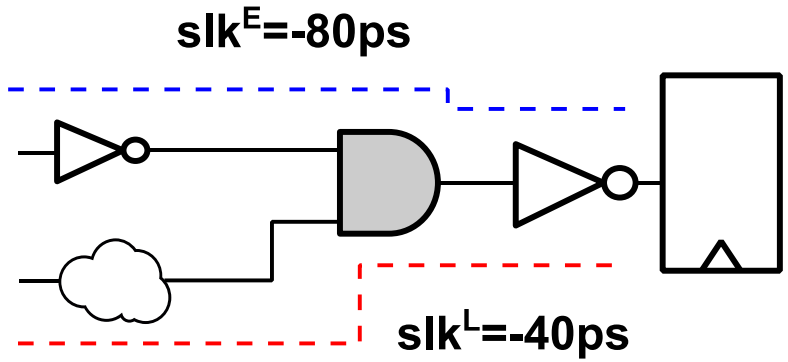
Cell sizing

- Try every option for cell to resize
- Keep the option with the lowest local cost
  - Options that cause load/slew/slack violations are rejected
- Applied on gates and flip flops that are
  - Early or late timing critical
  - Power/area critical

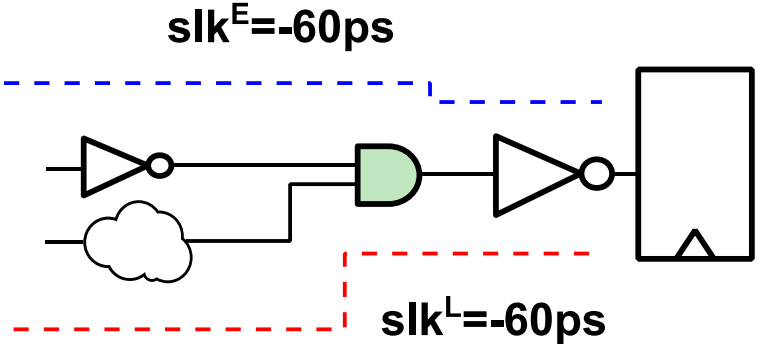


Handling of early/late timing conflicts

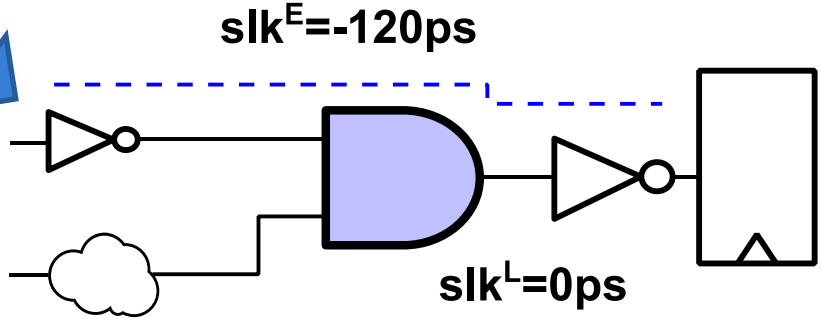
- Refers to cells with **conflicting early/late timing violations**
- LR based resizing will try to **balance the slacks** based on LM values ⇒ **slow convergence**
- Solution: only include **late LMs in the local cost function** of these cells
  - Sizing focuses on late violations
  - Early violations will be solved by other methods (e.g. buffering)



Initial circuit



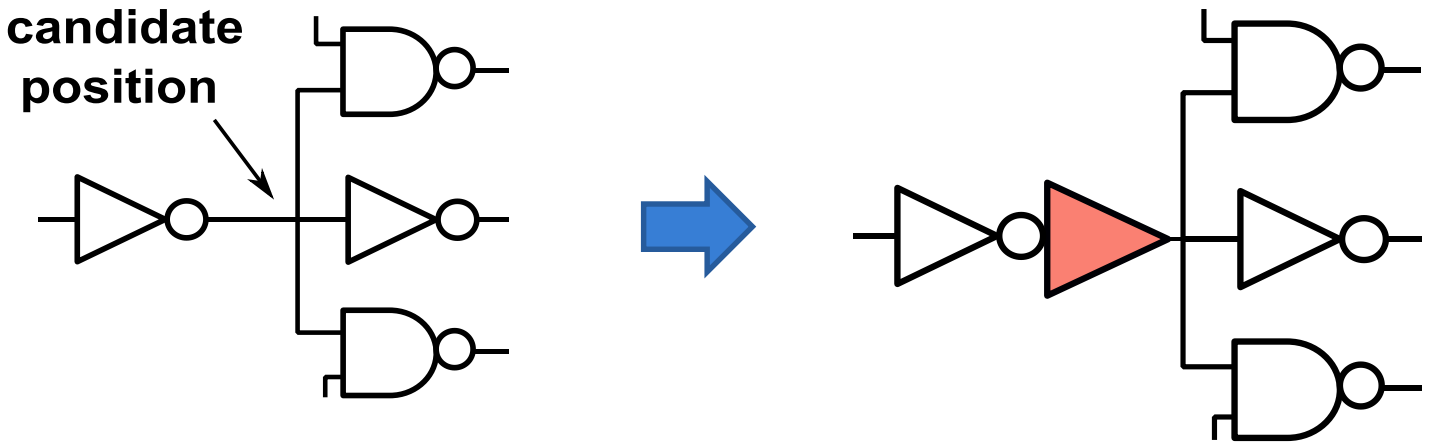
No conflict handling



Conflict handling by late slack focus

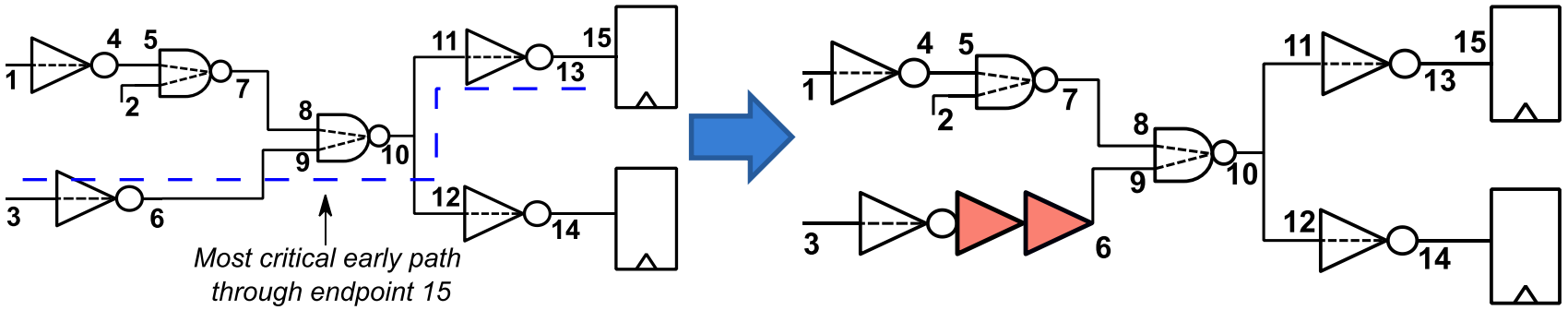
Buffer insertion for fixing late timing violations

- Used for driving large net loads
- Applied on the outputs of cells with high input to output capacitance ratio
- Try every buffer type and keep the lowest local cost option (including adding no buffer as an option)



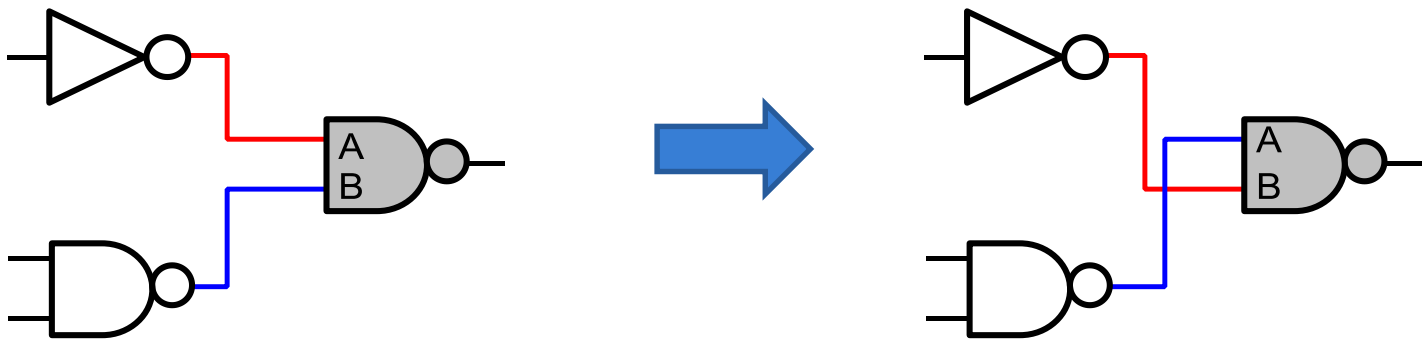
Buffer insertion for fixing hold timing violations

- Increase the delay on early-timing violating paths
- Where to add delay
  - On the most critical path through all early violating endpoints
  - One the pin on the most critical path with the highest late-early LM difference
- How much delay is added?
  - Add that much delay that does not degrade Late negative slack



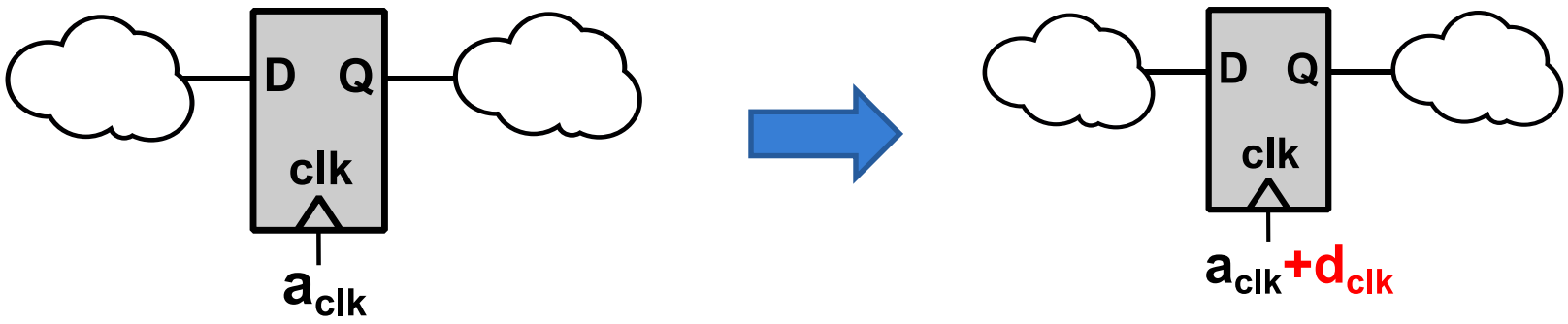
Pin swapping

- Reconnect nets of logically equivalent pins to improve timing
- For each gate that has equivalent pins:
  - Find the most critical input net
  - Try to assign it to each other equivalent pin
  - Keep the option with the lowest local LR cost



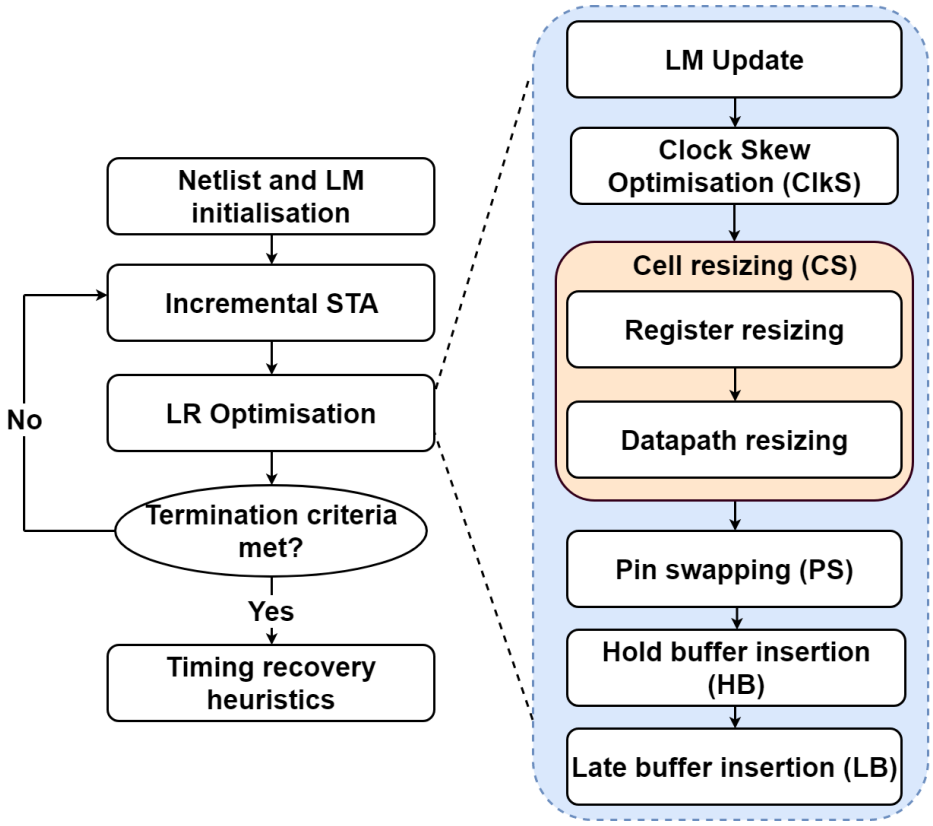
Useful clock skew assignment

- Changes the clock arrival time on registers
- The LMs of the clock pin guide delay addition/removal
  - $\lambda_{clk}^{L} = \lambda_{Q}^{L} + \lambda_{D}^{E}$
  - $\lambda_{clk}^{E} = \lambda_{Q}^{E} + \lambda_{D}^{L}$
- Delay added if $\lambda_{clk}^{E} > \lambda_{clk}^{L}$ else delay is removed



Overall optimization loop

- Starts with initializations
- Every iteration performs LR based transformations
- Timing information derived from the most critical late/early timing corners
- Final recovery steps improve QoR and remove hold violations



Multi-corner approach

- Optimization happens across many different corners
  - Timing closure required across all corners
  - Leakage only measured on the typical corner
- At the start of each iteration we identify the most critical corner for early and late mode ⇒ corner with the worst slack
  - Each corner has different delays, setup/hold times ⇒ most critical corner can change during optimization
  - All transformations and LM updates use timing information from the most critical corners
  - Only the most critical corners gets updated in the local timing updates

Order of applying each heuristic

- The order of applying each heuristic facilitates the propagation of timing info using only local timing updates
  1. Heuristics affecting endpoints – clock skew, register sizing
  2. Heuristics traversing the design in topological order – datapath sizing, pin swapping
  3. Heuristics performed on intermediate levels – early and late buffering
- In this way timing information is carried from the start points to the end points using only local timing updates

Experimental setup

- Applied on the benchmarks of the TAU 2019 Multi-Mode Multi-Corner (MMMC) Design Optimization contest
  - Six benchmarks provided
    - Sizes from 600 to about 800,000 cells
    - SPEF files provided for the three smallest designs
  - Optimization across five timing corners
- Compared against TAU contest winner's executable

Experimental results – best period

- The best period achieved by each method
  - Timing closure across all corners
- Our approach achieves 14% lower clock period (i.e. faster)
  - Also saves 15% leakage and 5% area

Design	Period (ps)		Leakage (uW)		Area (um^2)	
	Ours	Winner	Ours	Winner	Ours	Winner
s1196	1040	918	12.1	12.6	550	569
systemcdes	1777	1788	87	96	3665	3975
usb_funct	2166	2306	419	402	18579	17812
vga_lcd	1871	2826	3215	3106	149033	146257
leon2_iccad	4532	4677	25170	30354	1237510	1312960
leon3mp_iccad	3878	5246	20045	23816	971773	1030860

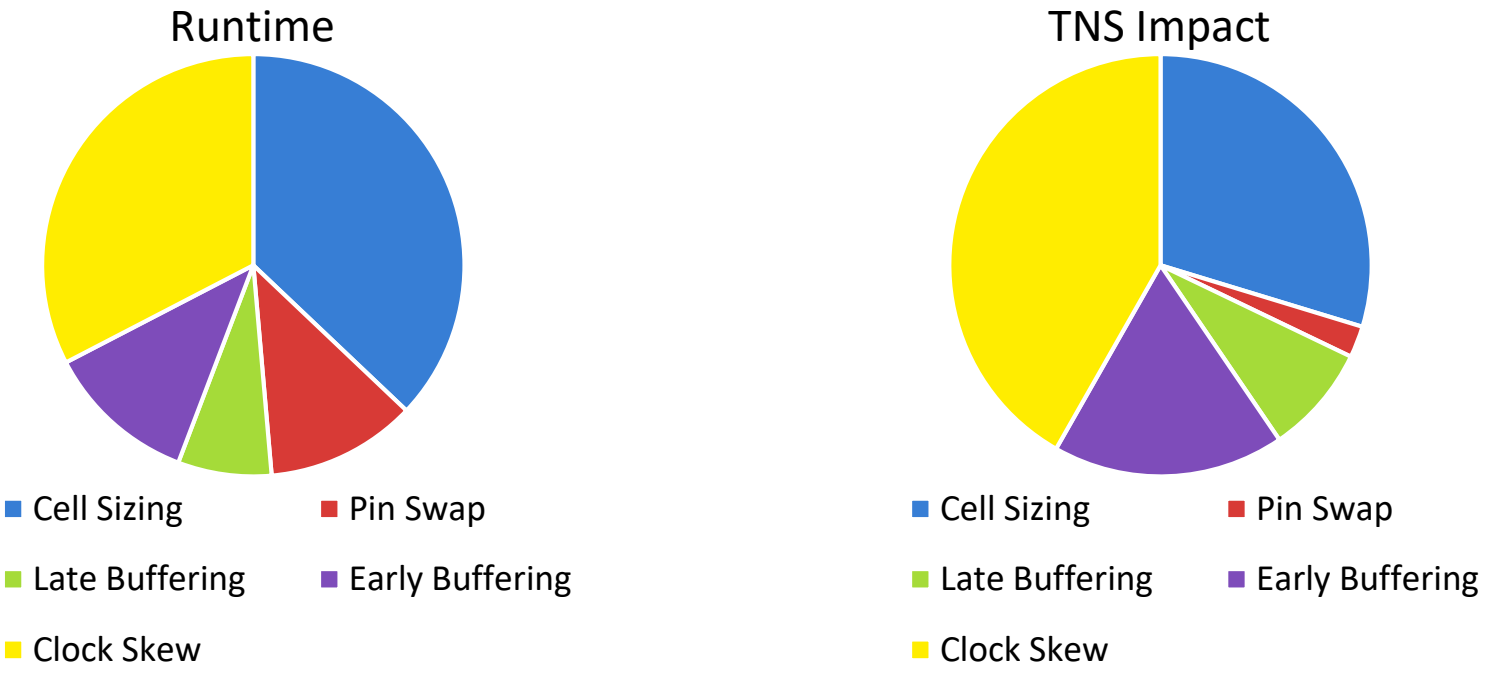
Experimental Results – Common clock period

- Leakage/Area comparison on the clock period where both algorithms close timing
  - Our approach saves 16% more leakage and 6% more area
- Better QoR for higher runtime on larger benchmarks

Design	Period (ps)	Leakage (uW)		Area (um^2)		Runtime (s)	
		Ours	Winner	Ours	Winner	Ours	Winner
s1196	1040	12	13	550	569	2	2
systemcdes	1788	85	96	3603	3975	17	21
usb_funct	2306	397	402	18002	17812	50	52
vga_lcd	2826	3075	3106	145752	146257	455	24
leon2_iccad	4677	24996	30354	1234180	1312960	4471	452
leon3mp_iccad	5246	19632	23816	962764	1030860	3862	362

The contribution of each heuristic

- Most impactful methods:
  - Cell sizing and clock skew assignment
  - But these are also the most runtime expensive



Conclusions

- Presented the simultaneous application of multiple heuristics inside the same LR optimization loop
- Additional specific novel parts
  - The overall formulation and the criteria needed for each heuristic to be embedded in the same optimization loop
  - Novel approach on handling early/late timing conflicts
  - Optimizes both combinational cells and sequential cells (registers), and optimizes the clock arrival time (useful skew)
- Future work:
  - Runtime improvements
  - Better explore the order of applying each heuristic