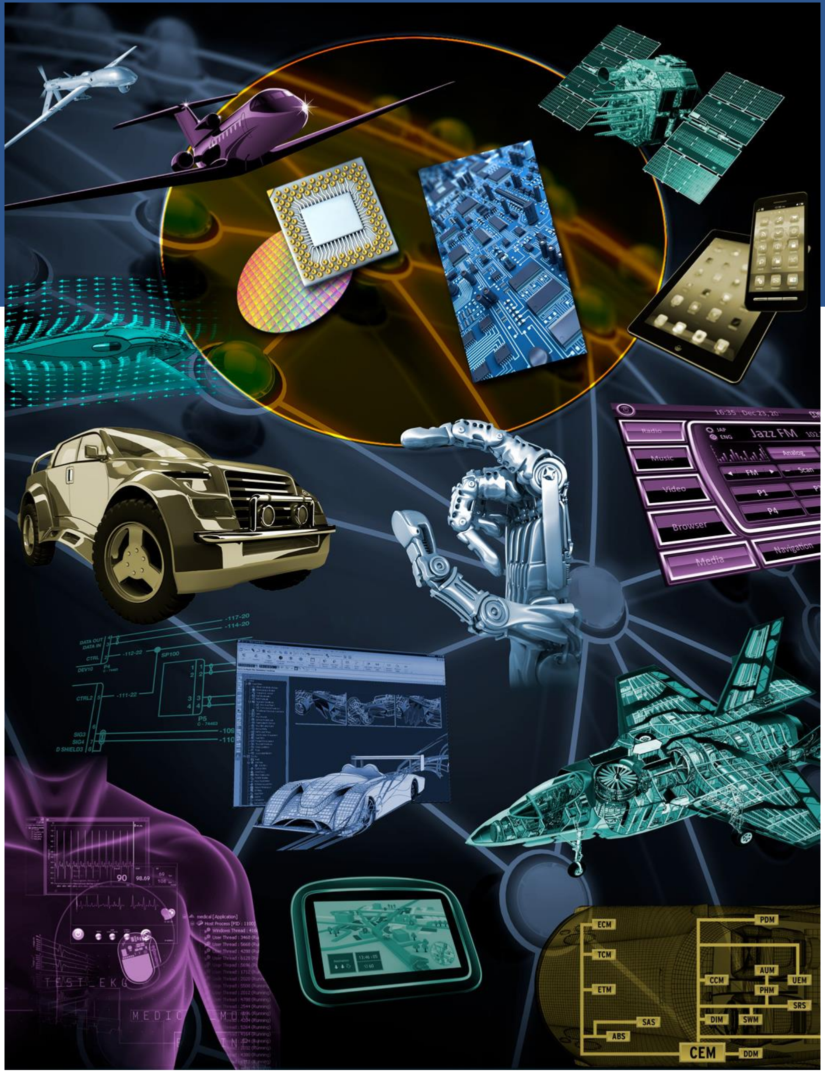


Lagrangian Relaxation Based Gate Sizing with Clock Skew Scheduling – A Fast and Effective Approach

Ankur Sharma, David Chinnery
Mentor, a Siemens Business

Chris Chu
Iowa State University, Computer Engineering

Mentor®
A Siemens Business

Outline

- Motivation – Previous work – Contribution

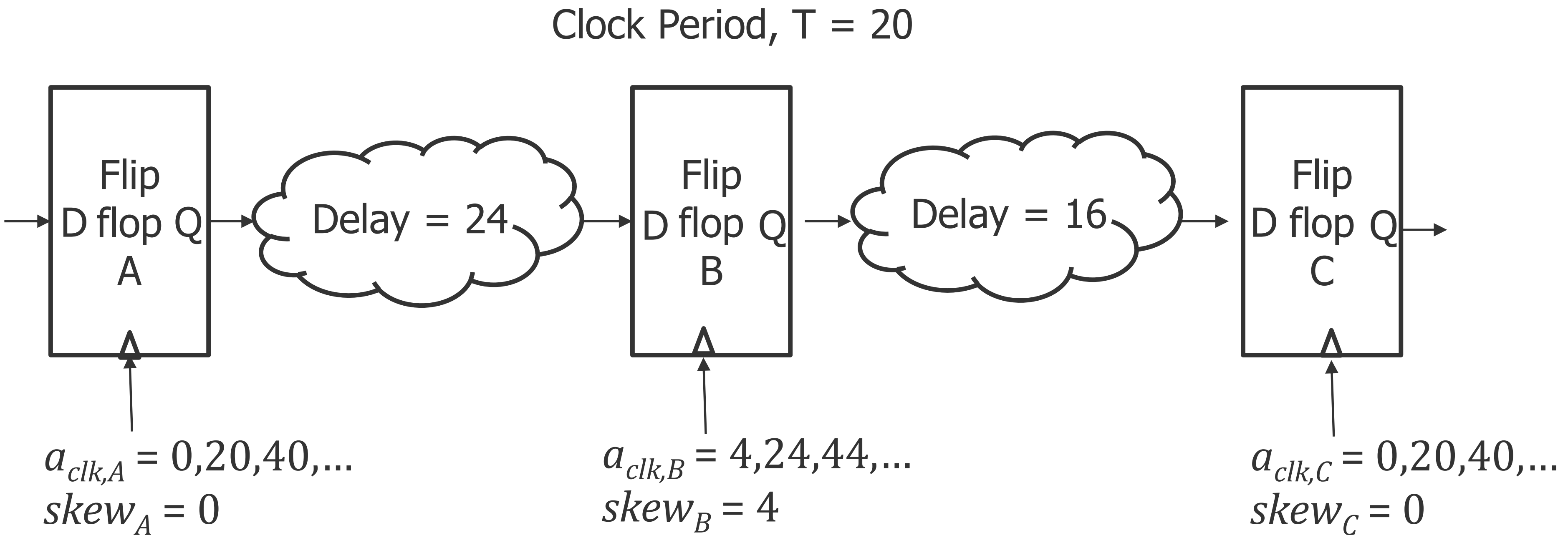
- Problem statement
- Previous approach
- Our proposed approach

- Experimental results
- Conclusion

Motivation

- Gate sizing is a key circuit optimization technique
  - Can trade off area, delay, and power
  - Delay-constrained leakage power minimization
- Skewing the clock arrival allows time borrowing between sequential stages. This is known as **useful skew**.
- Timing borrowing can be used for:
  - Increasing performance or satisfying delay constraints
  - Timing slack to reduce area or power

Simultaneous Gate Sizing with Skew Scheduling



- Signal is required to travel within one clock cycle
- Clock skew alters the required and arrival times

Previous Work

- [Chuang’95] formulated the primal problem as a linear program.
  - Piece-wise linear approximation of convex delays
- [Roy’07] formulated a Lagrangian dual problem (LDP). Solved the Lagrangian sub-problem simultaneously over size and skew.
  - Assumed continuous sizes and convex delays
- **[Wang’09]** transformed the primal problem to eliminate skew variables. Formulated an LDP and maximized the dual.
  - Used network flow solver to update Lagrange multipliers
  - Optimal for continuous sizes and convex delays
- [Shklover’12] formulated an LDP with discrete sizes and skews.
  - Focus on clock tree optimization via dynamic programming

Our Contributions

- Integration of clock skew scheduler inside LR gate sizer (**EGSS**).
  - Our LR formulation preserves the acyclic structure of the timing graph.
  - Modify Lagrange multiplier update to account for skew
  - A new strategy for solving the Lagrangian sub-problem with skew variables
- For comparison, we extended the dual maximization strategy from [Wang’09] to apply to discrete sizes and non-convex delay (**NetFlow**).
- We identify and empirically demonstrate several limitations of realizing primal optimality via dual maximization.

[Wang’09] J. Wang, D. Das, and H. Zhou. Gate sizing by Lagrangian relaxation revisited. IEEE TCAD 28(7):1071–1084, 2009.

Primal Problem Formulation

$$\underset{\boldsymbol{x},\boldsymbol{a},\boldsymbol{w}}{\text{minimize}} \quad p(\boldsymbol{x},\boldsymbol{w})$$

Minimize total leakage power

subject to

$$\boxed{\begin{aligned} &a_i + d_{ij}(\boldsymbol{x}) \le a_j, && \forall (i,j) \in E \\ &a_{d_k} \le T - setup_k + w_k, && \forall k \in FF \\ &w_k + d_{clk,q_k} \le a_{q_k}, && \forall k \in FF \end{aligned}}$$

Timing constraints

$$\boxed{w_{min} \le w_k \le w_{max}, \qquad \forall k \in FF}$$

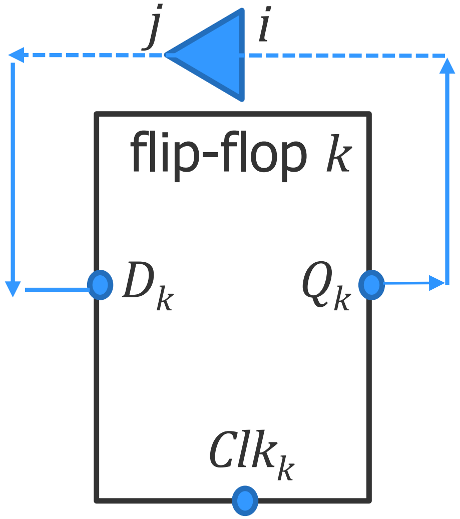
Skew bounds

T : target clock period
x : cell sizes
a_i : arrival time at node i
(i,j): timing arc from node i to node j
E : set of all timing arcs
d_{ij}: delay of timing arc from node i to node j
w_k: skew at flip-flop k
FF: set of flip-flops

Timing Graph

- Graphical representation of timing constraints

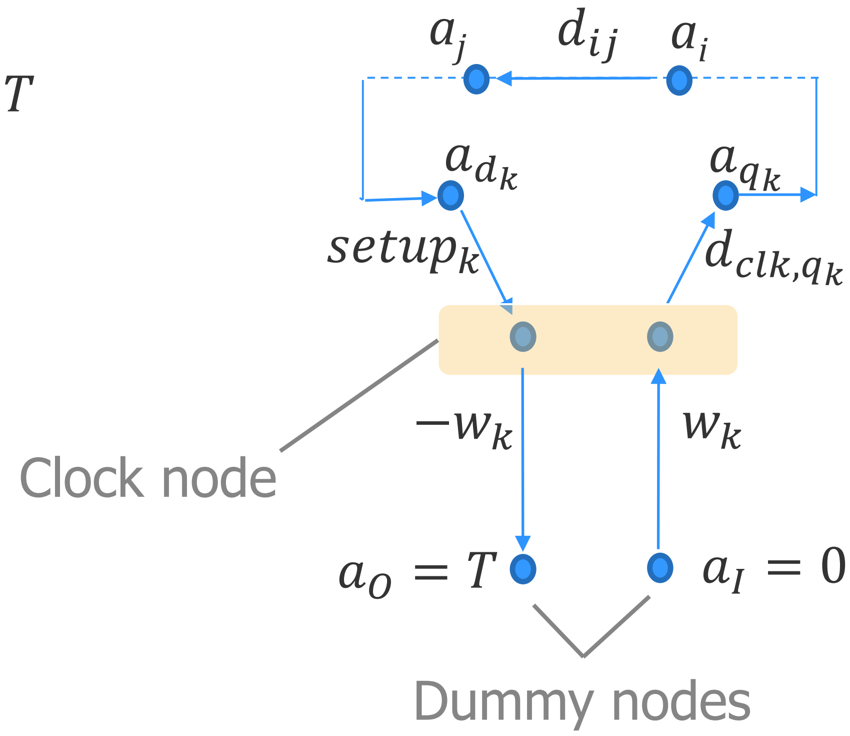
Circuit



Timing constraints

$$a_i + d_{ij}(\boldsymbol{x}) \leq a_j$$
$$a_{d_k} + setup_k - w_k \leq T$$
$$w_k + d_{clk,q_k} \leq a_{q_k}$$

Timing graph



NetFlow – Skew Elimination

- Due to [Wang'09]. We refer to it as **NetFlow**.

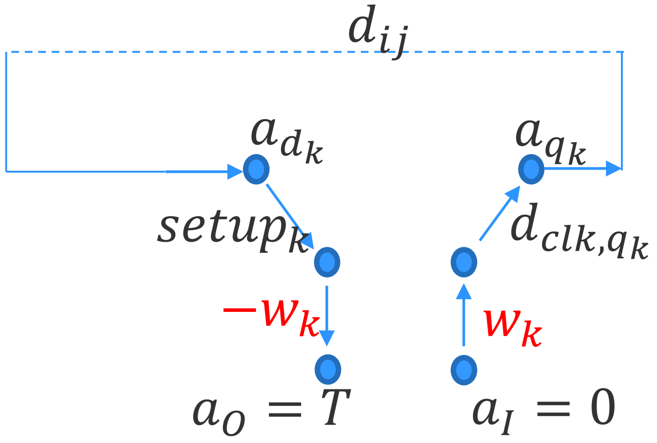
$$a_{d_k} + setup_k - w_k \leq T$$
$$w_k + d_{clk,q_k} \leq a_{q_k}$$
$$w_{min} \leq w_k \leq w_{max}$$

⬇

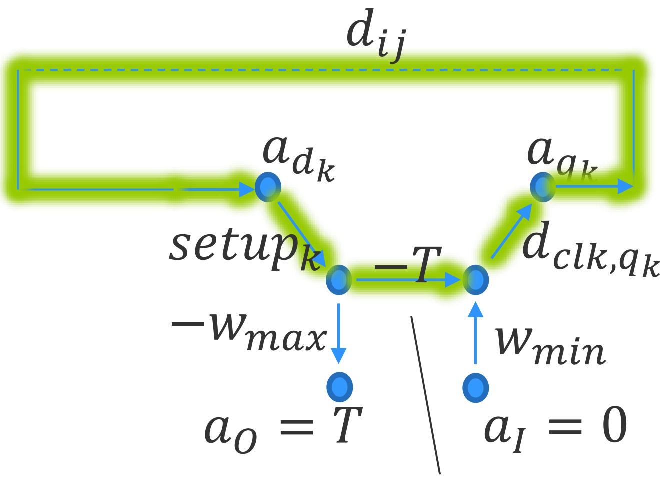
$$a_{d_k} + setup_k - T \leq w_k \leq a_{q_k} - d_{clk,q_k}$$
$$w_{min} \leq w_k \leq w_{max}$$

⬇

$$a_{d_k} + setup_k - T \leq w_{max}$$
$$w_{min} \leq a_{q_k} - d_{clk,q_k}$$
$$a_{d_k} + setup_k - T \leq a_{q_k} - d_{clk,q_k}$$



O and I are dummy nodes.

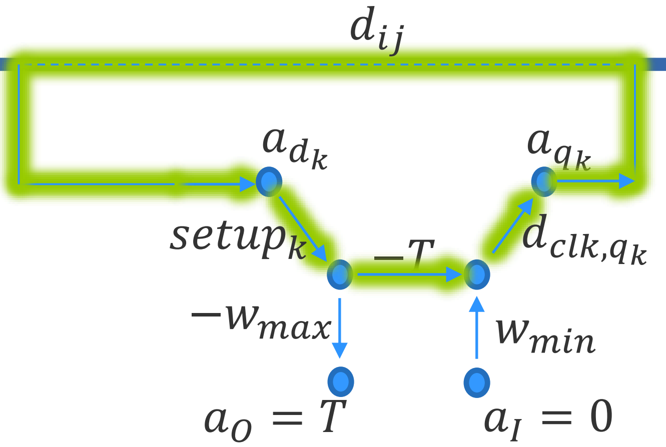


New arc

No skews, but there are loops in the timing graph.

[Wang'09] J. Wang, D. Das, and H. Zhou. Gate sizing by Lagrangian relaxation revisited. IEEE TCAD 28(7):1071–1084, 2009.

NetFlow – Lagrangian Relaxation Formulation



Primal problem:

$$\underset{\boldsymbol{x},\boldsymbol{a}}{\text{minimize}}\, p(\boldsymbol{x})$$

subject to

$$\begin{array}{ll} a_i + d_{ij}(\boldsymbol{x}) \le a_j, & \forall (i,j) \in E \\ a_{d_k} + setup_k - T \le w_{max}, & \forall k \in FF \\ w_{min} \le a_{q_k} - d_{clk,q_k}, & \forall k \in FF \\ a_{d_k} + setup_k - T \le a_{q_k} - d_{clk,q_k}, & \forall k \in FF \\ x_g \in X_g, & \forall g \in G \end{array}$$

Lagrangian relaxation sub-problem (LRS$_\lambda$)

$$g(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}}(L_{\boldsymbol{\lambda}}(\boldsymbol{x}))$$

Lagrangian function:

$$L_{\boldsymbol{\lambda}}(\boldsymbol{x}) = p(\boldsymbol{x}) + \sum_{(i,j)\in E} \lambda_{ij} \times cost_{ij}(\boldsymbol{x})$$

$cost_{ij}$ is the cost of arc (i,j), i.e. d_{ij}, $setup_k$, etc.
λ_{ij} is the Lagrange multiplier for timing arc (i,j).

Lagrangian dual problem (LDP):

$$\underset{\boldsymbol{\lambda}\ge\boldsymbol{0}}{\text{maximize}}\, g(\boldsymbol{\lambda})$$

subject to

$$\boldsymbol{\lambda} \in \Omega = \left\{\boldsymbol{\lambda} \middle| \textstyle\sum_{i|(i,u)\in E} \lambda_{iu} = \sum_{i|(u,i)\in E} \lambda_{ui}, \forall u \in N\right\}$$

flow conservation

where N is the set of all nodes in the timing graph.

Network flow solver to update λ.

[Wang'09] J. Wang, D. Das, and H. Zhou. Gate sizing by Lagrangian relaxation revisited. IEEE TCAD 28(7):1071–1084, 2009.

NetFlow – Dual Maximization

Lagrangian dual problem (LDP):

$$\underset{\boldsymbol{\lambda}\geq\mathbf{0}}{\text{maximize}} \quad g(\boldsymbol{\lambda})$$

subject to

flow conservation constraints on $\boldsymbol{\lambda}$

LRS$_\lambda$:

$$g(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}}\left(p(\boldsymbol{x}) + \sum_{(i,j)\in E} \lambda_{ij} \times cost_{ij}(\boldsymbol{x})\right)$$

Iteratively,

- Update $\boldsymbol{\lambda}$, for given $\boldsymbol{x}$ subject to flow constraints
  - Formulated as a min-cost network flow problem. Run time expensive
- Update $\boldsymbol{x}$, for given $\boldsymbol{\lambda}$
  - Heuristically solve LRS – a discrete combinatorial optimization problem.

Focus is dual maximization rather than primal feasibility

[Wang'09] J. Wang, D. Das, and H. Zhou. Gate sizing by Lagrangian relaxation revisited. IEEE TCAD 28(7):1071–1084, 2009.

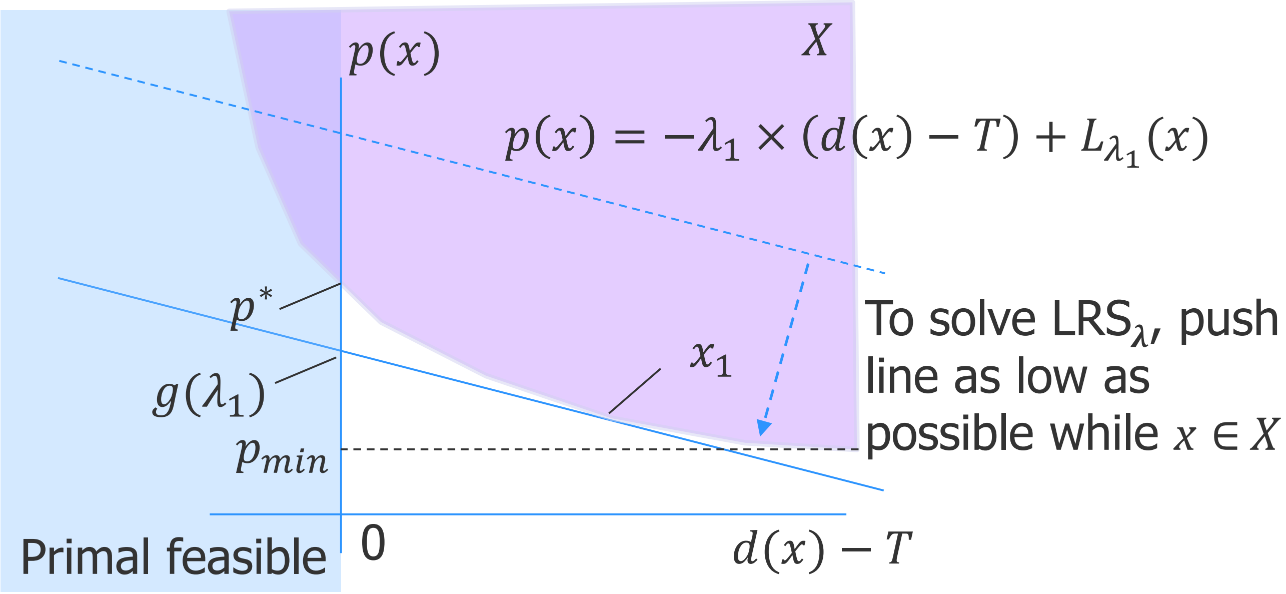
NetFlow – Visualizing Dual Maximization

For a single gate circuit:

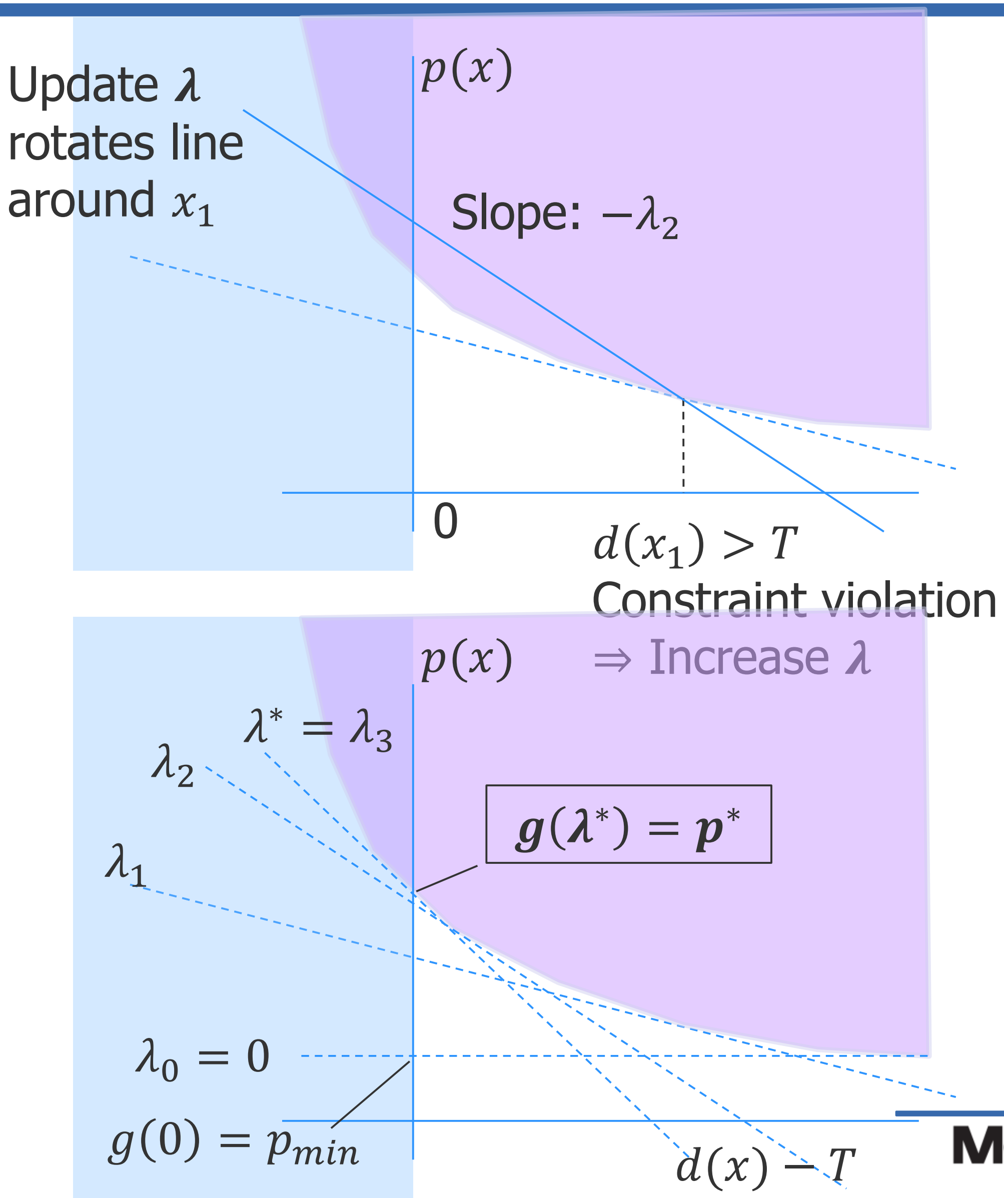
$$L_{\lambda}(x) = p(x) + \lambda \times (d(x) - T)$$

Equation of line on $p(x)$ vs. $d(x)$–T plane:

- The slope is –$\boldsymbol{\lambda}$.
- $L_{\lambda}(x)$ is the intercept on the $p(x)$ axis



To solve LRS$_{\lambda}$, push line as low as possible while $x \in X$



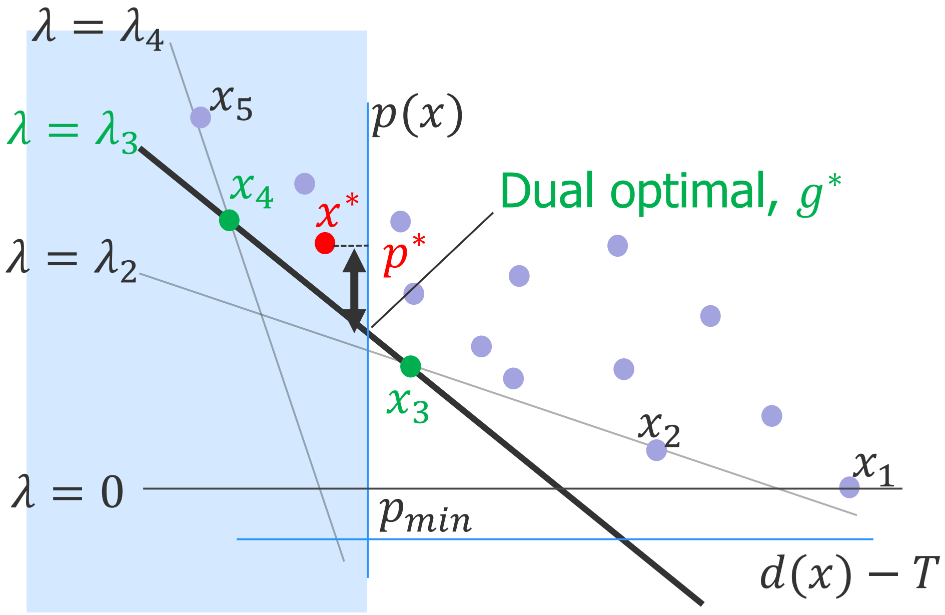
Update $\boldsymbol{\lambda}$ rotates line around x_1

Constraint violation

⇒ Increase λ

NetFlow: Dual Maximization Limitations with Discrete Sizes

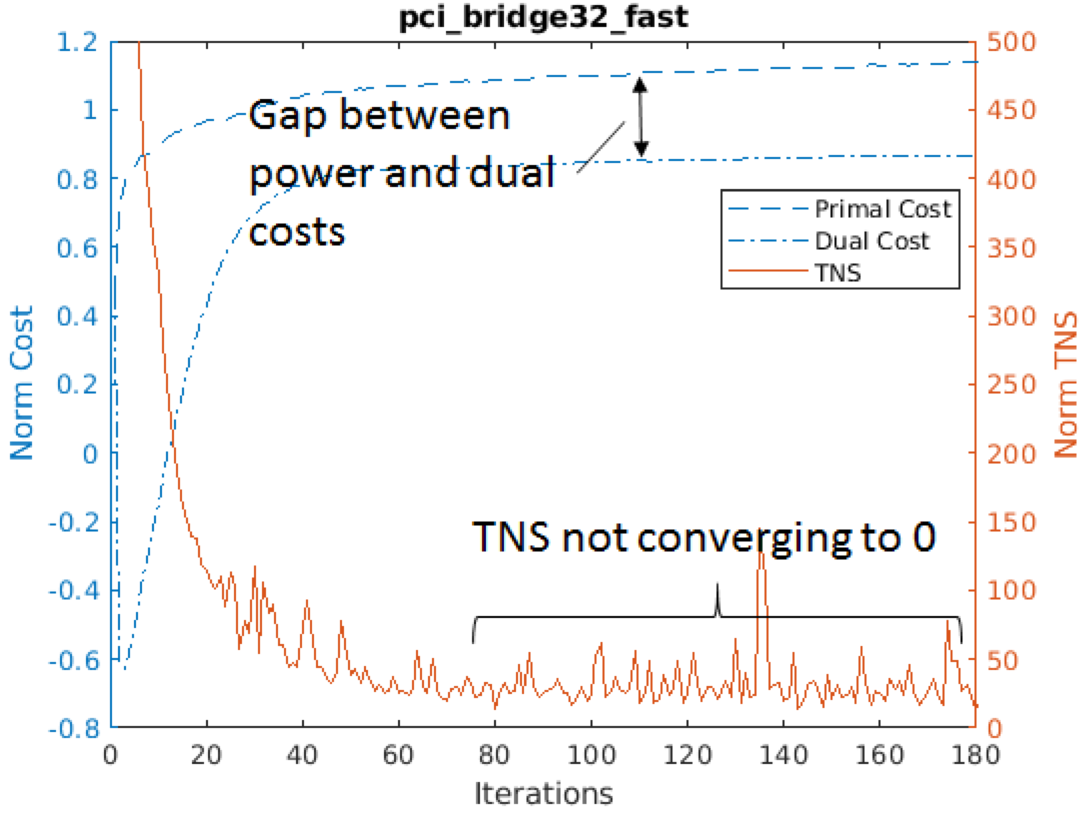
- **Duality gap:** Dual optimum may not be equal to primal optimum, $g^* < p^*$
- **Primal feasibility:** At dual optimum, multiple sizing solutions are possible & some don't satisfy timing constraints.
  - The dual optimal $g(\lambda_3)$ is realized at x_3 as well as x_4, but only x_4 is primal feasible.
- Dual optimality is not guaranteed, as LRS solver is no longer optimal



Each dot ● denotes a distinct sizing solution.

NetFlow: Dual Maximization Limitations with Discrete Sizes

- Three profiles are shown:
  - Primal cost (blue dash)
  - Dual cost (blue dash-dot)
  - Total negative slack (TNS) (red solid)
- Dual cost is less than primal cost.
  - Gap is roughly 20% wide; may partly be due to the duality gap.
- TNS does not converge to zero.
  - Oscillations prevent convergence
- Due to discreteness and non-convexity, dual maximization does not guarantee primal feasibility



Effective Gate Sizer and Skew Scheduler (EGSS)

- Seamlessly integrates with state-of-the-art discrete LR gate sizer
- Re-use LRS solver from discrete LR gate sizer
  - Focus on primal feasibility rather than exact computation of dual function
  - Extend the LRS solver to iteratively size gates and schedule skews
- Explicitly update skews rather than deducing them implicitly
- Modify and apply projection based Lagrange multiplier update
  - Compared to min-cost flow solver based multiplier update
    - Linear runtime complexity, more than a order of magnitude faster
    - Much better convergence
  - Requires the timing graph to be loop-free

EGSS Lagrangian Relaxation Formulation

Primal Problem:

$$\underset{\boldsymbol{x},\boldsymbol{a},\boldsymbol{w}}{\text{minimize}}\, p(\boldsymbol{x},\boldsymbol{w})$$

subject to

$$a_i + d_{ij}(\boldsymbol{x}) \leq a_j, \qquad \forall (i,j) \in E$$

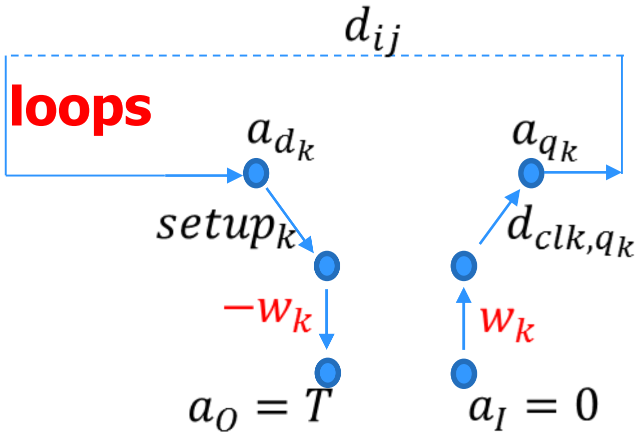
$$a_{d_k} \leq T - setup_k + w_k, \forall k \in FF$$

$$w_k + d_{clk,q_k} \leq a_{q_k}, \qquad \forall k \in FF$$

$$x_g \in X_g, \qquad \forall g \in G$$

$$w_{min} \leq w_k \leq w_{max}, \qquad \forall k \in FF$$

Skews but no loops



LRS_{λ}:

$$g(\boldsymbol{\lambda}) = \min_{\boldsymbol{x},\boldsymbol{w}} \left(p(\boldsymbol{x},\boldsymbol{w}) + \sum_{(i,j)\in E} \lambda_{ij} d_{ij}(\boldsymbol{x}) + \sum_{k\in FF} (\lambda_{q_k} - \lambda_{d_k}) w_k \right) - \sum_{k\in FF} \lambda_k T$$

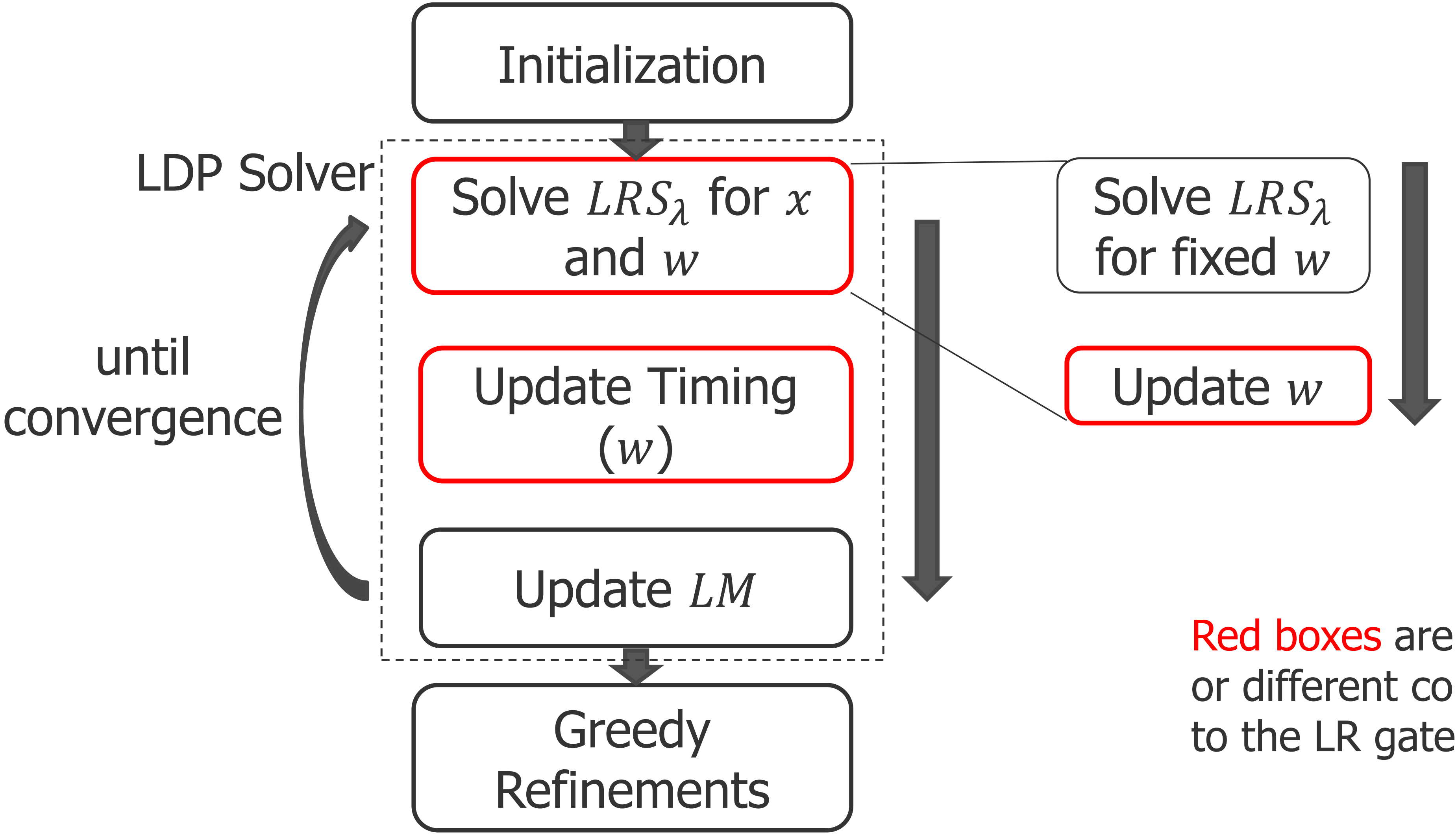
subject to

$x_g \in X_g$

$w_{min} \leq w_k \leq w_{max}, \forall k \in FF$, where FF is the set of flip-flops

LDP: $$\max_{\substack{\boldsymbol{\lambda}\in\Omega \\ \boldsymbol{\lambda}\geq 0}} g(\boldsymbol{\lambda})$$

EGSS – Overall Flow



Red boxes are new or different compare to the LR gate sizer.

EGSS – Skew Update

- From the LRS$_\lambda$ objective
  $$\min_{\boldsymbol{x},\boldsymbol{w}}\left(p(\boldsymbol{x},\boldsymbol{w}) + \sum_{(i,j)\in E}\lambda_{ij}d_{ij}(\boldsymbol{x}) + \sum_{k\in FF}\left(\lambda_{q_k}-\lambda_{d_k}\right)w_k\right)$$
- Extract out the skew terms:
  $$h(\boldsymbol{w}) = skew\_power(\boldsymbol{w}) + \sum_{k\in FF}\left(\lambda_{q_k}-\lambda_{d_k}\right)w_k$$
- Ignore skew power; minimize $h(\boldsymbol{w})$
- If $\lambda_{q_k} \geq \lambda_{d_k}$, $w_k = w_{min}$; else $w_k = w_{max}$
  - Causes oscillation
- We propose to use:
  $$\Delta w_k = \frac{slack_{q_k} - slack_{d_k}}{2}$$
  $$w_k = \max\{w_{min}, \min\{w_{max}, w_k + \Delta w_k\}\}$$

EGSS – Modified Lagrange Multiplier Update

- Skew alters the tightness of the timing constraint

Projection idea:

- Traverse design in reverse topological order
- Distribute the sum of the outgoing multipliers to incoming multipliers in proportion to their existing values.

```
// Our new LM update heuristic
for each $k \in FF$
    $\lambda_{d_k} = \lambda_{d_k} \times \left(1 + \frac{a_{d_k} - T - \mathbf{w_k}}{T}\right)^K$
for each timing arc $(i,j)$
    $\lambda_{ij} = \lambda_{ij} \times \left(1 + \frac{a_i + d_{ij} - q_j}{T}\right)^K$

Project $\lambda$ to feasible space
```

Experimental Setup

- We implemented NetFlow and EGSS in C++
  - Used Gurobi's linear program solver for solving MCNF
- ISPD2012 and ISPD2013 gate sizing contest benchmark suite
- We compare results from:
  - EGSS without skew (sizing only baseline)
  - NetFlow with w_{max} = 165ps, w_{min} = 0ps
  - EGSS with w_{max} = 165ps, w_{min} = 0ps
- All of them use 8 threads

ISPD 2012 Designs – Power Reduction

			Power (W)			Power Reduction	
Design	**# Gates**	**Clock (ps)**	**Baseline**	**NetFlow**	**EGSS**	**vs. Baseline**	**vs. NetFlow**
DMA_slow	23109	900	0.135	0.111	0.104	23.1%	6.6%
pci_bridge32_slow	29844	720	0.098	0.073	0.072	26.9%	2.2%
des_perf_slow	102427	900	0.583	0.420	0.404	30.6%	3.7%
vga_lcd_slow	147812	700	0.329	0.310	0.310	5.9%	0.2%
b19_slow	212674	2500	0.569	0.577	0.556	2.2%	3.7%
leon3mp_slow	540352	1800	1.335	1.326	1.321	1.0%	0.4%
netcard_slow	860949	1900	1.763	1.762	1.762	0.1%	0.0%
DMA_fast	23109	770	0.245	0.173	0.137	44.3%	20.8%
pci_bridge32_fast	29844	660	0.141	0.083	0.078	44.7%	6.2%
des_perf_fast	102427	735	1.436	0.686	0.615	57.2%	10.3%
vga_lcd_fast	147812	610	0.417	0.318	0.316	24.3%	0.8%
b19_fast	212674	2100	0.729	0.823	0.682	6.5%	17.1%
leon3mp_fast	540352	1500	1.449	1.393	1.360	6.1%	2.4%
netcard_fast	860949	1200	1.846	1.804	1.800	2.5%	0.2%
		Average	0.791	0.704	0.680	**19.7%**	**5.3%**

Loose target

Tighter target, more savings

ISPD 2012 Designs – Run Time

	Run Time (min)			Speedup	
Design	**Baseline**	**NetFlow**	**EGSS**	**vs. Baseline**	**vs. NetFlow**
DMA_slow	0.07	7.90	0.08	0.86x	94.0x
pci_bridge32_slow	0.09	8.70	0.10	0.91x	88.0x
des_perf_slow	0.32	21.09	0.30	1.07x	69.2x
vga_lcd_slow	0.44	28.35	0.44	0.98x	64.0x
b19_slow	0.83	45.75	1.24	0.67x	37.0x
leon3mp_slow	2.52	194.90	2.91	0.87x	67.0x
netcard_slow	2.35	343.90	2.82	0.83x	122.0x
DMA_fast	0.08	9.20	0.10	0.85x	92.0x
pci_bridge32_fast	0.10	9.20	0.11	0.95x	84.0x
des_perf_fast	0.40	23.39	0.34	1.18x	69.1x
vga_lcd_fast	0.56	27.90	0.50	1.10x	55.3x
b19_fast	1.13	19.58	1.61	0.70x	12.2x
leon3mp_fast	3.13	233.10	3.56	0.88x	65.5x
netcard_fast	3.33	237.05	3.98	0.83x	59.5x
Average	1.10	86.43	1.29	0.91x	**69.9x**

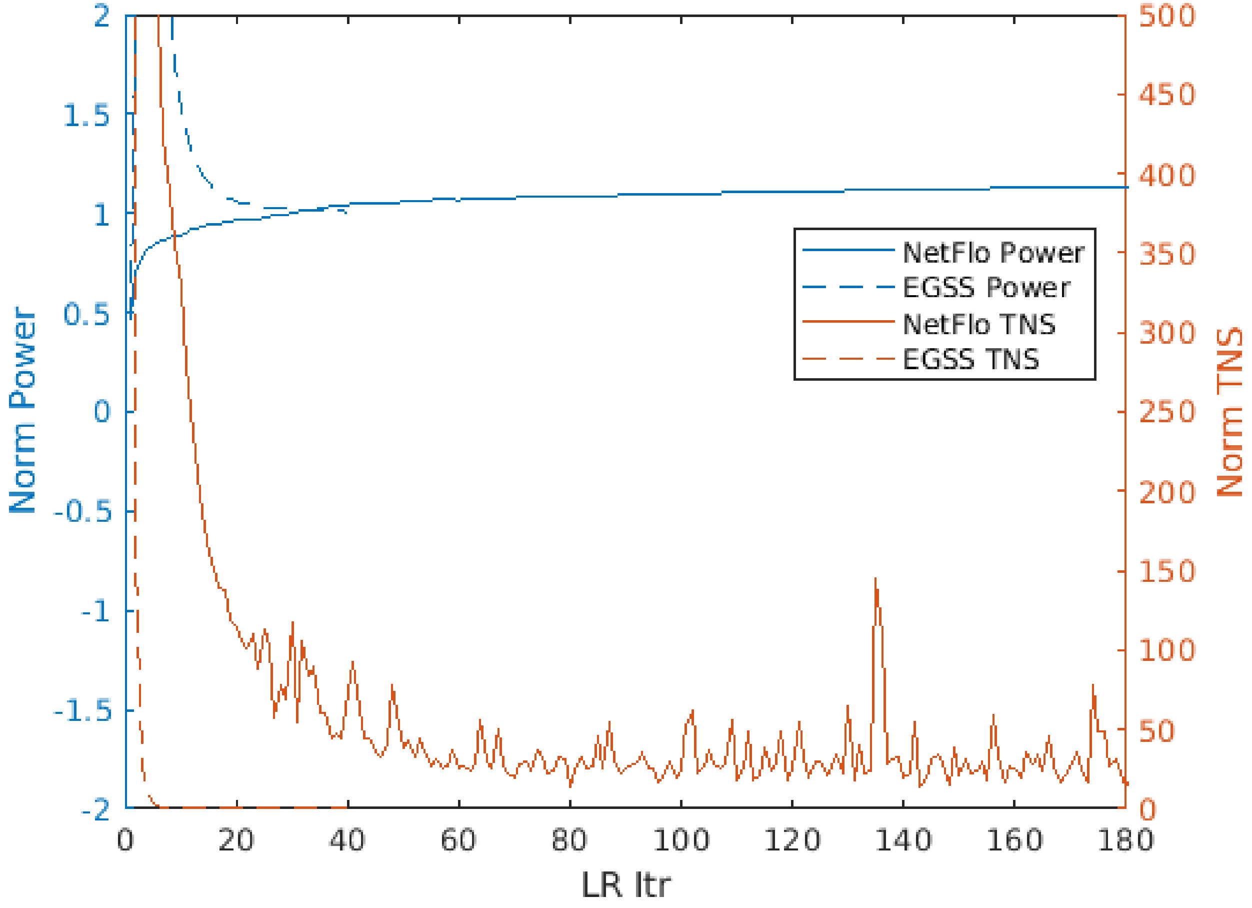
- Only 10% slower than the gate sizing only baseline; 70x faster than NetFlow!
- On a million gate design, netcard_fast, EGSS takes only 4min

Comparing NetFlow and EGSS

- Compare **TNS** and **power** profiles for NetFlow (solid lines) and EGSS (dash lines)
- Exit criterion:
  - TNS is below a threshold, or
  - Maximum (200) iterations are reached

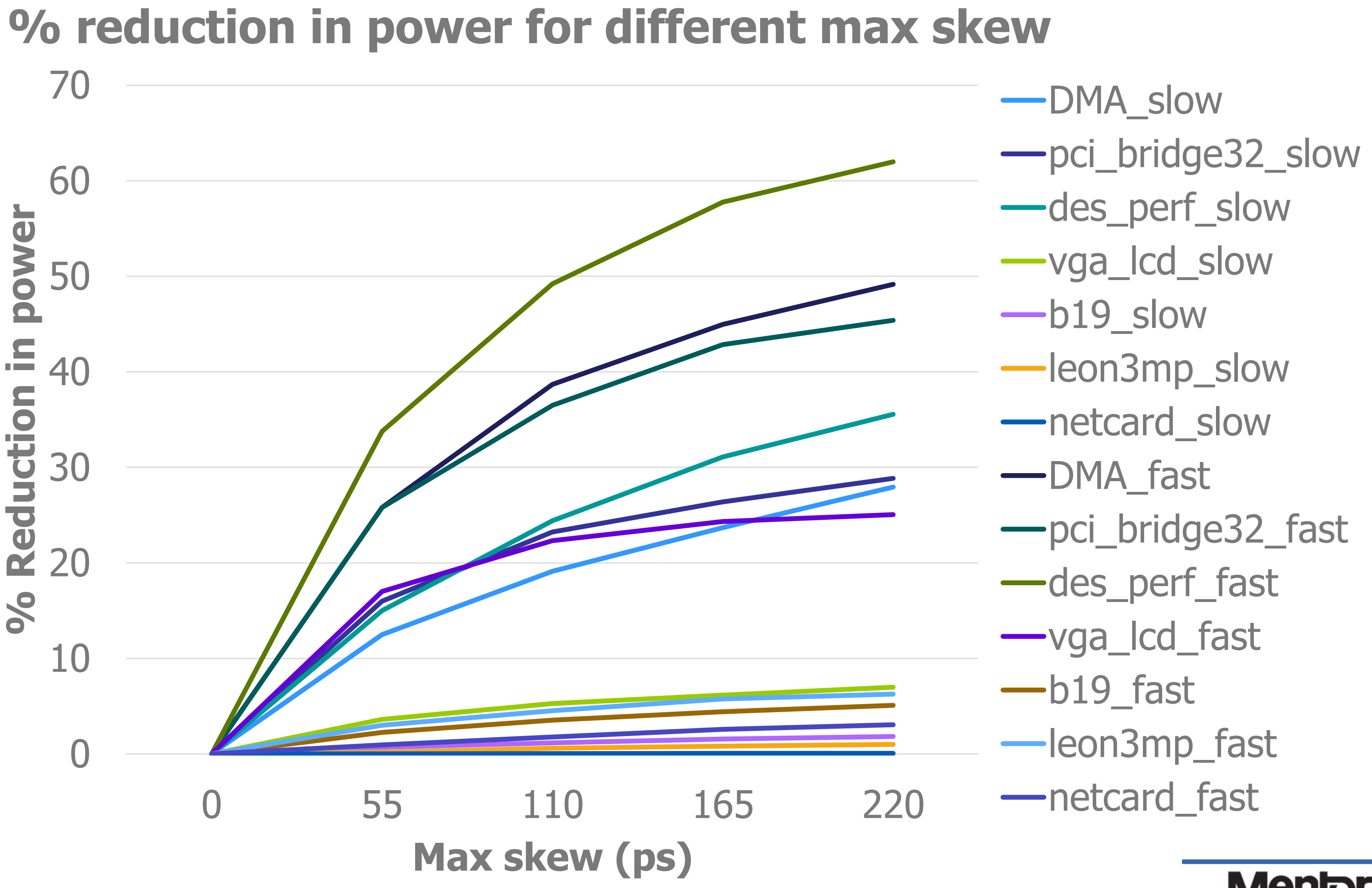
Why better convergence with EGSS?

- Focus on primal feasibility
  - TNS is quickly brought close to zero, and degradation is not allowed thereafter.
- Projection based multiplier update.



Power Saved With EGSS vs. Max Skew Bound

Sequential cycles in the circuits limit the benefit derived from useful skew optimization. Hence, power savings saturate.



Conclusion

- Gate sizing potential can be enhanced by allowing variable skew
- Previously, Lagrange dual maximization has been used to realize a primal optimal solution. But with discrete sizes, dual maximization has limitations, leading to a duality gap and sub-optimal results.
- We proposed an effective gate sizing & skew scheduling algorithm, seamlessly integrated into a state-of-the-art discrete LR gate sizer.
  - Proposed a modified LRS solver flow to solve for sizes as well as skews
  - Modified existing projection based Lagrange multiplier update
- Our tool saved 20% more power with only 10% extra runtime vs. only gate sizing on ISPD 2012 gate sizing contest benchmarks.

Mentor®

A Siemens Business

www.mentor.com

BACKUP

NetFlow – Lagrange Multiplier Update

- [Wang'09] Formulated dual maximization as min-cost network flow ($MCNF_\lambda$) in the neighborhood of current λ
- Objective of $MCNF_\lambda$ is linear first-order approximation of $g(\boldsymbol{\lambda})$
- Neighborhood is heuristically defined
- Min-cost flow solver to compute $\Delta\boldsymbol{\lambda}$
- Line search along $\Delta\lambda$ to maximize dual
- **Line search and min-cost solver are severe runtime bottlenecks.**

$MCNF_\lambda$:

$$\underset{\Delta\boldsymbol{\lambda}}{\text{minimize}}(-\nabla g(\boldsymbol{\lambda}) \times \Delta\boldsymbol{\lambda})$$

subject to

$$\Delta\boldsymbol{\lambda} \in \Omega$$

$$\max\{-\lambda_{ij}, -U\} \leq \Delta\lambda_{ij} \leq U$$

ISPD 2013 Designs – Power Reduction

Design	# Gates	Clock (ps)	[Flach'14] Power (W)	EGSS Power (W)	% Power Reduction	TNS (ps)
usb_phy_slow	510	450	0.001	0.001	2.4	0
pci_bridge32_slow	27244	1000	0.057	0.055	2.6	0
fft_slow	30782	1800	0.087	0.081	6.9	-4
cordic_slow	41673	3000	0.271	0.227	16.2	-58
des_perf_slow	104310	1300	0.330	0.273	17.4	-28
edit_dist_slow	121004	3600	0.425	0.429	-0.8	-222
matrix_mult_slow	153542	2800	0.444	0.409	7.9	-63
netcard_slow	884427	2400	5.155	5.167	-0.2	-13
usb_phy_fast	510	300	0.002	0.001	14.6	0
pci_bridge32_fast	27244	750	0.085	0.062	27.9	-4
fft_fast	30782	1400	0.194	0.120	38.1	-10
cordic_fast	41673	2626	1.001	0.634	36.7	-228
des_perf_fast	104310	1140	0.649	0.357	44.9	-71
edit_dist_fast	121004	3000	0.540	0.501	7.2	-300
matrix_mult_fast	153542	2200	1.611	0.847	47.4	-341
netcard_fast	884427	2000	5.200	5.180	0.4	-65
		Average	**1.003**	**0.897**	**16.8**	**-88**

- 6.5% less power at slow constraints
- 27.2% less power at fast constraints
- We trade timing accuracy for speed, so there are a few timing violations (TNS).

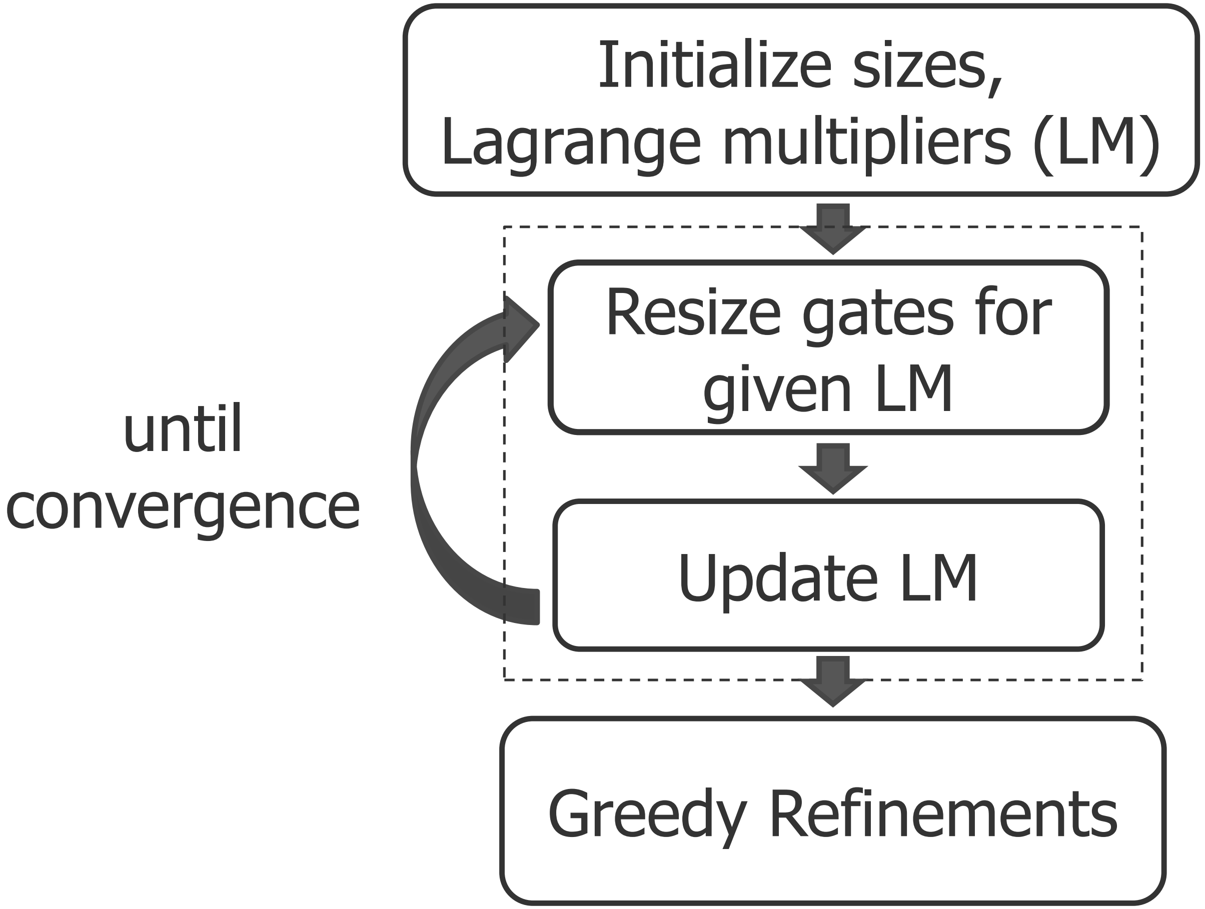
ISPD 2013 Designs – Run Time

Design	[Flach '14] Run time (min)	EGSS run time (min)	Speedup
usb_phy_slow	0.5	0.2	2.1x
pci_bridge32_slow	10.5	0.9	11.2x
fft_slow	25.7	1.2	20.6x
cordic_slow	69.0	2.1	33.2x
des_perf_slow	132.3	5.2	25.6x
edit_dist_slow	123.9	4.6	26.7x
matrix_mult_slow	226.1	7.4	30.4x
netcard_slow	483.6	24.5	19.7x
usb_phy_fast	0.4	0.2	1.8x
pci_bridge32_fast	22.6	1.0	22.7x
fft_fast	40.4	1.5	27.3x
cordic_fast	117.1	3.5	33.8x
des_perf_fast	347.9	9.5	36.5x
edit_dist_fast	353.0	6.2	56.6x
matrix_mult_fast	396.0	12.5	31.8x
netcard_fast	400.9	28.4	14.1x
Average	**171.9**	**6.8**	**24.6x**

[Flach’14] LR gate sizer is single-threaded

Generic flow

- Integration of clock



Initialize sizes, Lagrange multipliers (LM)

Resize gates for given LM

Update LM

until convergence

Greedy Refinements

Results Summary

- Average across all designs

	Gate sizing only	NetFlow	EGSS
Average power (W)	0.791	0.704	0.680
Average Run Time (min)	1.10	86.43	1.29