IEEE/ACM ICCAD 2025 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN 44th Edition

DiffCCD: Differentiable Concurrent Clock and Data Optimization

Yuhao Ji[1,3], Yuntao Lu[1], Zuodong Zhang[3], Zizheng Guo[2,3], Yibo Lin[2,3,4], Bei Yu[1],

[1] The Chinese University of Hong Kong

[2] Peking University, Beijing

[3] Institute of Electronic Design Automation

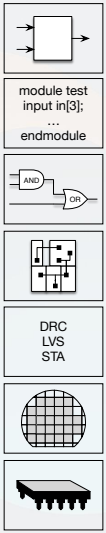
[4] Advanced Innovation Center for Integrated Circuits

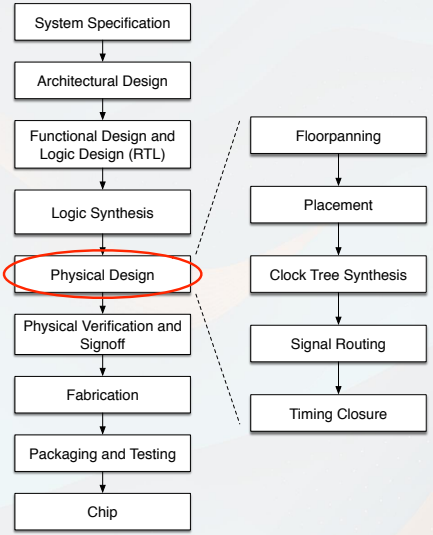
Oct. 27, 2025

Outline

Introduction

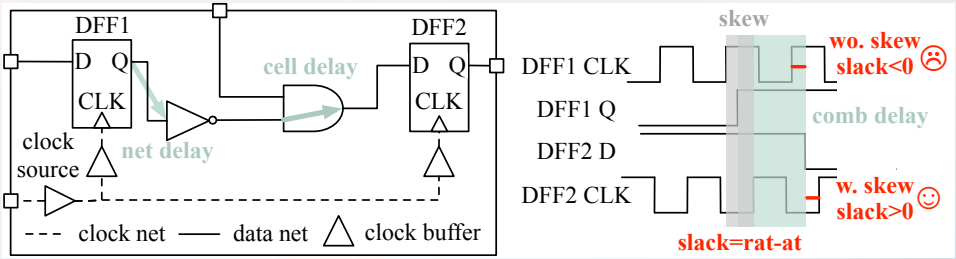
Timing Closure





- The key goal of physical design is to achieve better PPA
- Timing closure determines the performance of the design (how fast the design can run)
- Timing-driven everything...

Useful Skew Optimization (Post-CTS)



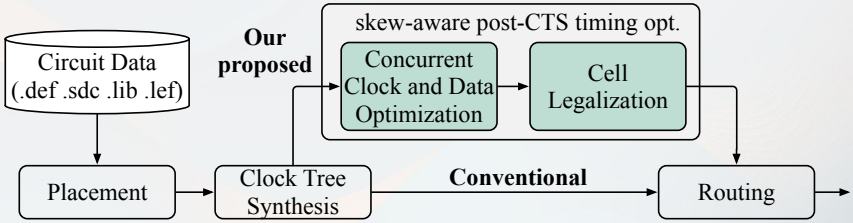
- Clock skew: the variation in clock signal arrival times at different FFs
- Properly controlled clock skew can help to improve timing performance

Existing Works & Challenges

- **Zero Skew Assumption:** Assume (Placement & Routing) and expect (Clock Tree Synthesis) ideal clock net
- **Isolated Step:** Calculate ideal skew without considering physical implementation constraints
- **Optimization Challenges:** Certain FFs must receive accelerated clock signals while others require deliberately delayed clock delivery, indicating non-smooth solution space

DiffCCD

Overall Flow



- Simultaneously optimizes clock skew and logic delay
- Clock buffer sizing and placement refinement in the post-CTS stage

Problem Formulation

$$\min_{\mathcal{P},\boldsymbol{s}_c} \sum_{\text{endpoint } ep} \text{TM}_{ep}(\mathcal{P},\boldsymbol{s}_c) + \sum_{\text{net } e} \text{WL}_e(\mathcal{P},\boldsymbol{s}_c) + \lambda \text{Density}(\mathcal{P},\boldsymbol{s}_c). \tag{1}$$

$$\text{TM}_{ep} = -t_1 \text{WNS}_{ep} - t_2 \text{TNS}_{ep}, \tag{2}$$

where $\mathcal{P}$ represents coordinates of instances, $\boldsymbol{s}_c$ denotes clock buffer sizes, and λ, t_1, t_2 are weighting factors for different objectives.

Logit-based Clock Buffer Sizing Modeling

$\mathcal{S}$: Set of available clock buffer sizes
$\mathcal{N}_c$: Set of clock buffer instances

$$l_b^s \overset{\text{init}}{=} \begin{cases} c, & \text{if } s = s_{b0} \\ 1, & \text{otherwise} \end{cases}, \quad s \in \mathcal{S}, b \in \mathcal{N}_c, \tag{3}$$

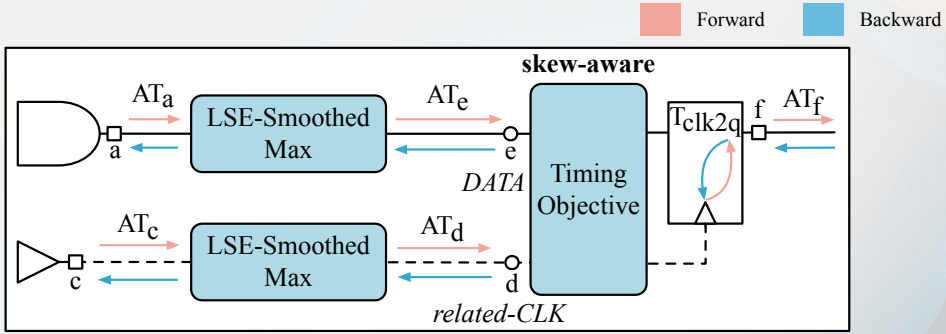
where s_{b0} is the initial buffer size of clock buffer b derived from CTS, c is a constant which is greater than 1, indicating a higher initial preference for the buffer size s_{b0}.

$$\theta_b^s = \frac{\exp((l_b^s + \text{gumbel}(u^s))/\tau)}{\sum_{i \in \mathcal{S}} \exp((l_b^i + \text{gumbel}(u^i))/\tau)}, \quad s \in \mathcal{S}, \tag{4}$$

$$s_b = \underset{s \in \mathcal{S}}{\text{argmax}}(\theta_b^s), \quad b \in \mathcal{N}_c, \tag{5}$$

- Gumbel noise basically provides a randomization effect to prevent premature convergence to suboptimal solutions

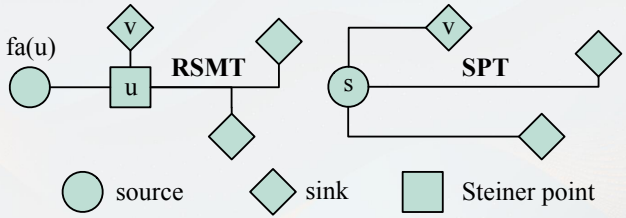
Skew-aware Timing Objective



- Establish *CLK*-to-*OUTPUT* timing arcs with delay T_{clk2q} from each FF's *CLK* port to its *OUTPUT* port

$$Slack = AT_{related\text{-}CLK} + T_{\text{period}} - AT_{DATA} \quad (6)$$

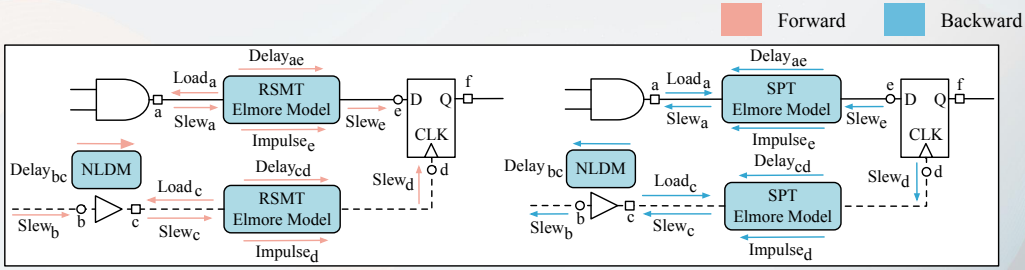
One Finding



Finding & Thinking

- RSMT provides accurate routing approximation but suffers from discontinuous construction. SPT offers continuous construction at the cost of accuracy.
- There is no need to have totally accurate gradients

STE-based framework



- **Straight-Through Estimator (STE):** Maintaining precise computations in forward propagation while approximating the gradients in backward propagation

Elmore Model Gradients Calibration

RSMT-based Elmore model:

$$
\begin{aligned}
Load_R(u) &= cap_u + \sum_{v \in children(u)} Load_R(v), \\
Delay_R(u) &= Delay_R(fa(u)) + res_{fa(u)\to u} \cdot Load_R(u), \\
LDelay_R(u) &= cap_u \cdot Delay_R(u) + \sum_{v \in children(u)} LDelay_R(v), \\
Beta_R(u) &= Beta_R(fa(u)) + res_{fa(u)\to u} \cdot LDelay_R(u), \\
Impulse^2_R(u) &= 2 \cdot Beta_R(u) - Delay^2_R(u)
\end{aligned}
\tag{7}
$$

SPT-based Elmore model:

$$
\begin{aligned}
Load_S(s) &= \sum_{v \in sinks} Load_S(v), \\
Delay_S(v) &= res_{s\to v} \cdot Load_S(v), \\
LDelay_S(v) &= cap_v \cdot Delay_S(v), \\
Beta_S(v) &= res_{s\to v} \cdot LDelay_S(v), \\
Impulse^2_S(v) &= 2 \cdot Beta_S(v) - Delay^2_S(v)
\end{aligned}
\tag{8}
$$

Elmore Model Gradients Calibration (con't)

- RSMT is used for forward propagation
- Calibrated SPT is used for gradient calculation

$$\begin{aligned} Load_R(s) &\simeq \alpha Load_S(s), \\ Delay_R(v) &\simeq \beta Delay_S(v), \\ Impulse_R^2(v) &\simeq \gamma Impulse_S^2(v). \end{aligned} \tag{9}$$

$$\begin{aligned} \nabla Load_R(s) &\simeq \alpha \sum_{v \in sinks} \nabla Load_S(v), \\ \nabla Delay_R(v) &\simeq \beta(\nabla res_{s\to v} \cdot Load_S(v) + res_{s\to v} \cdot \nabla Load_S(v)), \\ \nabla Impulse_R^2(v) &\simeq 2\gamma \cdot (\nabla Beta_S(v) - Delay_S(v) \cdot \nabla Delay_S(v)), \\ \nabla LDelay_S(v) &= \nabla cap(v) \cdot Delay_S(v) + cap(v) \cdot \nabla Delay_S(v), \\ \nabla Beta_S(v) &= \nabla res_{s\to v} \cdot LDelay_S(v) + res_{s\to v} \cdot \nabla LDelay_S(v). \end{aligned} \tag{10}$$

Approximate Clock Buffer Sizing Gradients

Forward propagation:

$$\begin{aligned} cap_i &= cap_i^{s_b}, \\ Delay_{i\to o} &= LUT_{s_b, i\to o} \end{aligned}, \quad arc(i,o) \in \text{buffer } b, \tag{11}$$

where s_b is the clock buffer size, and i and o denote the input and output pins of buffer b respectively.

Approximate Clock Buffer Sizing Gradients (con't)

Backward propagation:

$$\Psi_y(x) = \begin{cases} e^{\frac{2\cdot y - x}{2\cdot y} - \frac{(x-y)^2}{\sigma\cdot y}}, x \geq y, \\ e^{\frac{x}{2\cdot y} - \frac{(x-y)^2}{\sigma\cdot y}} - e^{\frac{1}{2}}, \text{otherwise}, \end{cases}$$

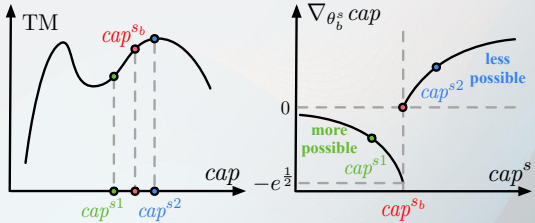
$$\nabla_{\theta_b^s} cap \simeq \begin{cases} \Psi_{cap^{s_b}}(cap^s), \nabla_{cap}\text{TM} \geq 0, \\ -\Psi_{cap^{s_b}}(2cap^{s_b} - cap^s), \text{otherwise}, \end{cases}$$

$$\nabla_{\theta_b^s} Delay \simeq \begin{cases} \Psi_{LUT_{s_b}}(LUT_s), \nabla_{Delay}\text{TM} \geq 0, \\ -\Psi_{LUT_{s_b}}(2LUT_{s_b} - LUT_s), \text{otherwise}, \end{cases}$$

(12)

$$\nabla_{\theta_b^s}\text{TM} = \nabla_{cap}\text{TM} \times \nabla_{\theta_b^s} cap$$

$$\nabla_{cap}\text{TM} \geq 0$$



(a)

- Avoid sudden size transition
  - manually construct gradient function by defining Ψ function to enable incremental upsizing or downsizing

ADMM-Based CCD Optimization

Let $\mathcal{Q} = \mathcal{P}$, the augmented Lagrangian function is:

$$\mathcal{L}(\mathcal{P}, \mathcal{Q}, \mathcal{L}_c, \mathcal{U}) = \sum \mathrm{TM}_{ep}(\mathcal{P}, \mathcal{L}_c) + \sum \mathrm{WL}_e(\mathcal{Q}, \boldsymbol{s}_c) + \lambda \mathrm{Density}(\mathcal{Q}, \boldsymbol{s}_c) + \langle \mathcal{U}, \mathcal{P} - \mathcal{Q} \rangle + \frac{\rho}{2} \|\mathcal{P} - \mathcal{Q}\|_2^2, \tag{13}$$

Update:

$$\begin{aligned}
\mathcal{P}^{k+1} &= \underset{\mathcal{P}}{\operatorname{argmin}} \sum \mathrm{TM}_{ep}(\mathcal{P}, \mathcal{L}_c^k) + \langle \mathcal{U}^k, \mathcal{P} - \mathcal{Q}^k \rangle + \frac{\rho}{2} \left\|\mathcal{P} - \mathcal{Q}^k\right\|_2^2, \\
\mathcal{L}_c^{k+1} &= \underset{\mathcal{L}_c}{\operatorname{argmin}} \sum \mathrm{TM}_{ep}(\mathcal{P}^{k+1}, \mathcal{L}_c), \\
\mathcal{Q}^{k+1} &= \underset{\mathcal{Q}}{\operatorname{argmin}} \sum \mathrm{WL}_e(\mathcal{Q}, \boldsymbol{s}_c^{k+1}) + \lambda \mathrm{Density}(\mathcal{Q}, \boldsymbol{s}_c^{k+1}) \\
&\quad + \langle \mathcal{U}^k, \mathcal{P}^{k+1} - \mathcal{Q} \rangle + \frac{\rho}{2} \left\|\mathcal{P}^{k+1} - \mathcal{Q}\right\|_2^2, \\
\mathcal{U}^{k+1} &= \mathcal{U}^k + \rho(\mathcal{P}^{k+1} - \mathcal{Q}^{k+1}).
\end{aligned} \tag{14}$$

Experimental Results

Experimental Setting

- CCD implementation: C++, CUDA, GPU Flute for Steiner tree generation.
- Benchmark: industrial designs from CircuitNet.

Table: Benchmark statistics.

Benchmark	Tech.	Period	#Cells	#Nets	#Pins	#Macros
VORTEX	14nm	2ns	112478	121142	430699	43
LARGEBOOM	14nm	2ns	787298	805963	3027324	636
OPENC910	14nm	2ns	798166	811653	3149667	32
PULPINO	28nm	2ns	20374	21248	75079	3
RISCY-a	28nm	2ns	56580	57168	220791	3
RISCY-FPU-a	28nm	2ns	78311	79253	300634	3
zero-riscy-a	28nm	2ns	46206	45972	176171	3

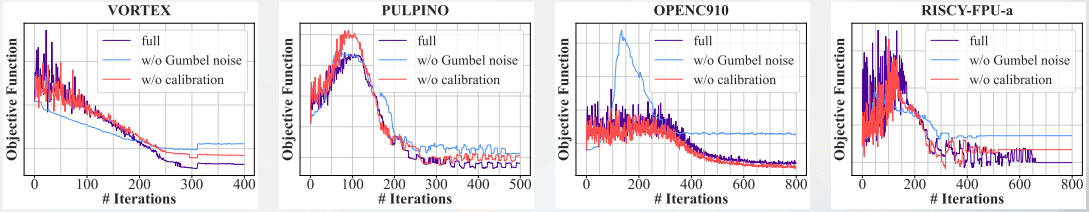
Experimental Results

Table: WNS, TNS and runtime comparisons. The average ratio represents the mean absolute ratio across all benchmarks, with lower values indicating superior performance for all metrics.

Benchmark	OpenROAD CTS		OpenROAD CTS + *repair_timing*			OpenROAD CTS + Our DiffCCD		
	WNS (ns)	TNS (ns)	WNS (ns)	TNS (ns)	Runtime (s)	WNS (ns)	TNS (ns)	Runtime (s)
VORTEX	**-1.034**	-3324.667	-1.529	-3354.724	1221	-1.099	**-1709.71**	**261**
LARGEBOOM	-0.617	-881.285	-0.764	-1354.435	4870	**-0.581**	**-664.672**	**805**
OPENC910	-0.917	-331.766	**-0.498**	-268.719	4802	-0.504	**-152.962**	**1101**
PULPINO	-0.641	-22.708	-0.049	-0.945	10977	**0**	**0**	**133**
RISCY-a	-0.559	-554.164	-0.128	**-4.59**	2187	**-0.097**	-6.424	**264**
RISCY-FPU-a	-1.097	-2515.743	-0.493	-853.758	5443	**-0.475**	**-830.637**	**333**
zero-riscy-a	-0.339	-46.185	-0.165	-2.153	**206**	**-0.114**	**-0.567**	239
Average Ratio	1.000	1.000	0.643	0.542	1.000	**0.499**	**0.298**	**0.106**

- Baseline: *repair_timing* of OpenROAD
- Compared with *repair_timing*:
  - An average improvement of 22.4% in worst negative slack (WNS) and 45.0% in total negative slack (TNS)
  - 9.434× speedup.

Ablation Study



(b) (c) (d) (e)

Loss curves under three configurations, where full method is our complete CCD optimization framework with both Gumbel noise and calibration mechanism.

- Gumbel noise significantly influences the early-stage optimization dynamics
- Empirically, combining both Gumbel noise and the calibration mechanism yields the best performance in most test cases

THANK YOU!