Timing-Aware Optimization of Die-Level Routing and TDM Assignment for Multi-FPGA Systems

Yijun Chen[1], Haoyuan Li[2], Chunyan Pei[2], Jianwang Zhai[1], Kang Zhao[1], and Wenjian Yu[2]

[1]Beijing University of Posts and Telecommunications

[2]Dept. Computer Science & Tech., BNRist, Tsinghua Univ.

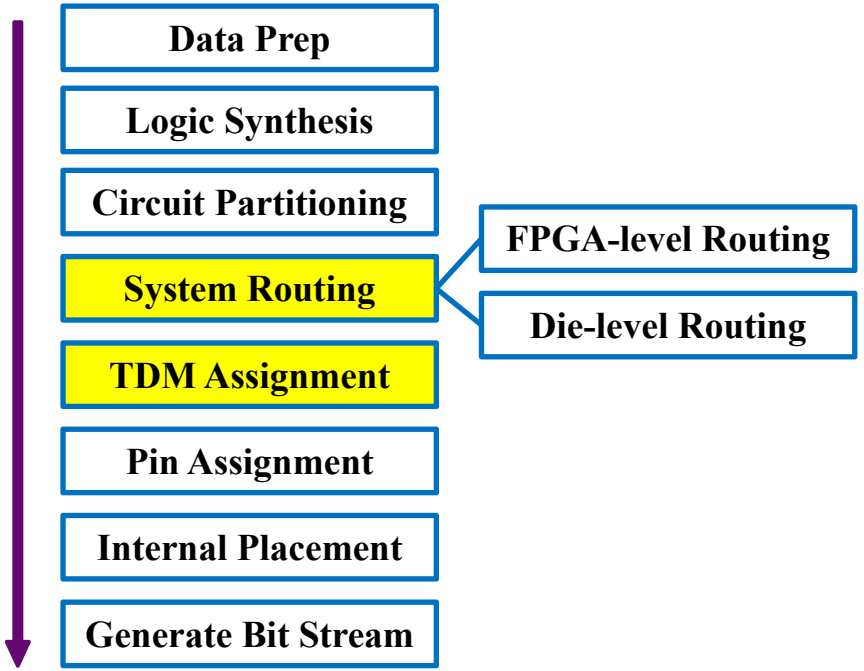
Jan. 22, 2026

Outline

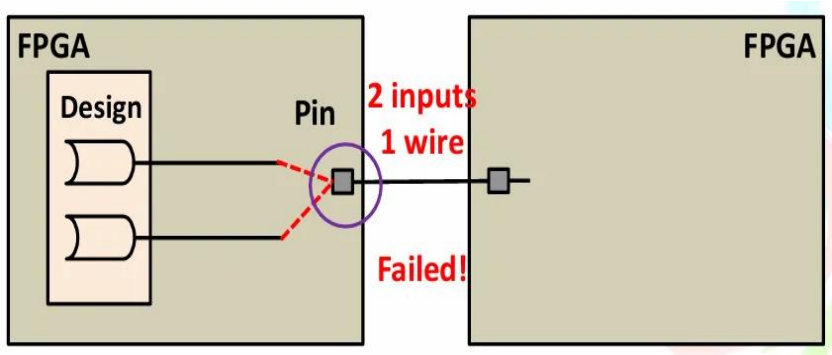
- Introduction
- Preliminaries
- Routing
- TDM Assignment
- Experimental Results
- Conclusion

Introduction

- EDA tools: Design, Verification and Manufacturing
- Logic verification: software verification; prototype verification
- Modern circuits exceed advanced 2.5D FPGAs capacity
- Netlist partitioned⇒ single FPGA⇒ signals cross FPGAs
- Cross-FPGA signals need MFS-based routing
- TDM multiplexes signals as cross-FPGA signals exceed links.



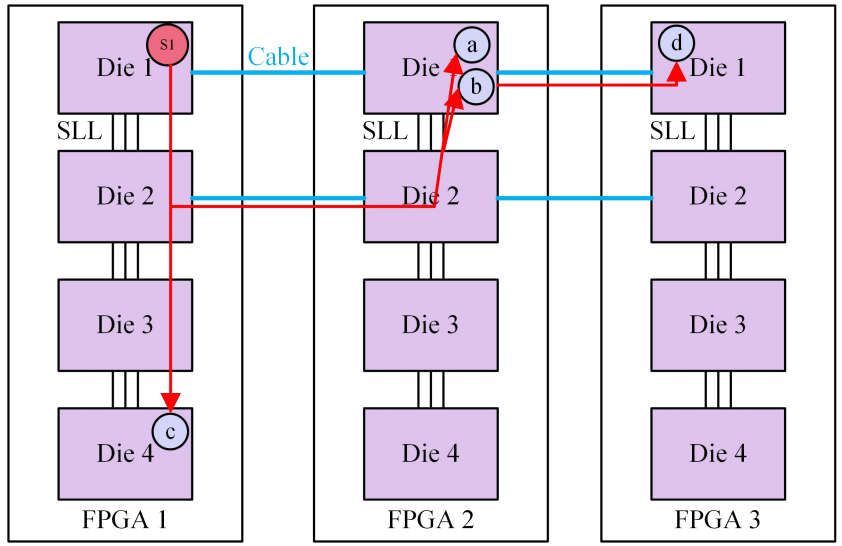
VP1902 FPGA S8-100 Logic System	Virtex™ UltraScale+™ VU19P FPGA
	



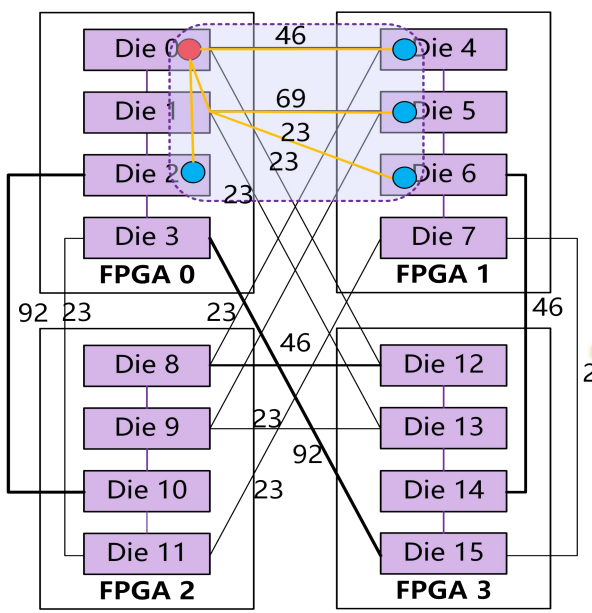
Introduction

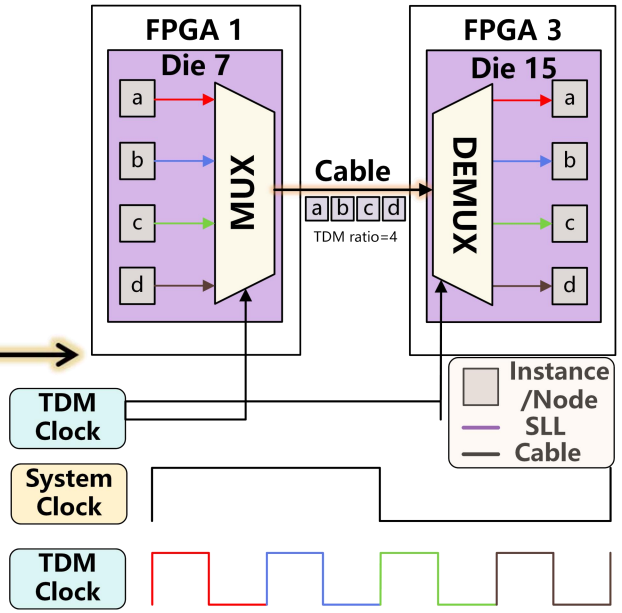
- Super Logic Regions (SLRs/dies) ⟷ high-capacity Super Long Lines (SLLs) ⇒ Die-Level System Routing.
- Limited link resources ⇒ congestion
- TDM increases delay ⇒ degrading system timing performance ↓

⇒ Routing and TDM assignment must be co-optimized to minimize the maximum system delay.



	Intra-FPGA	Inter-FPGA
Connection	Cable	SLL
Direction	one-way	bothway
Capacity	Scarce	Affluent
TDM	√	×
Delay	TDM delay	Die delay

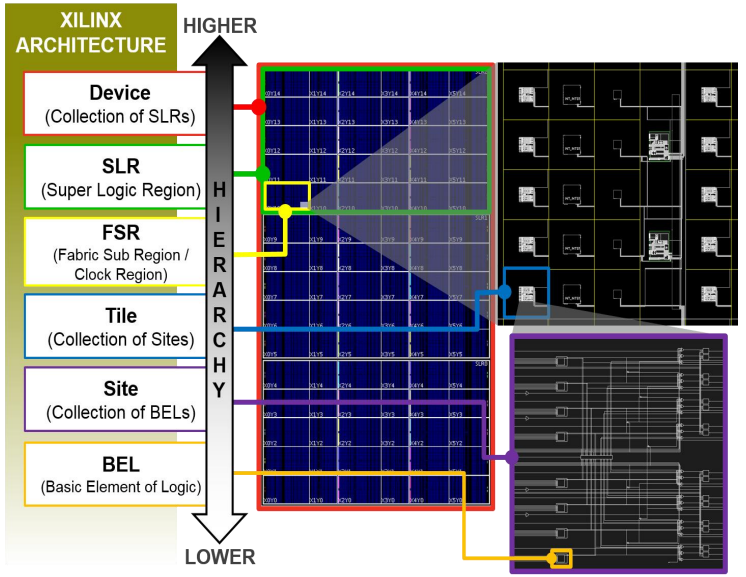




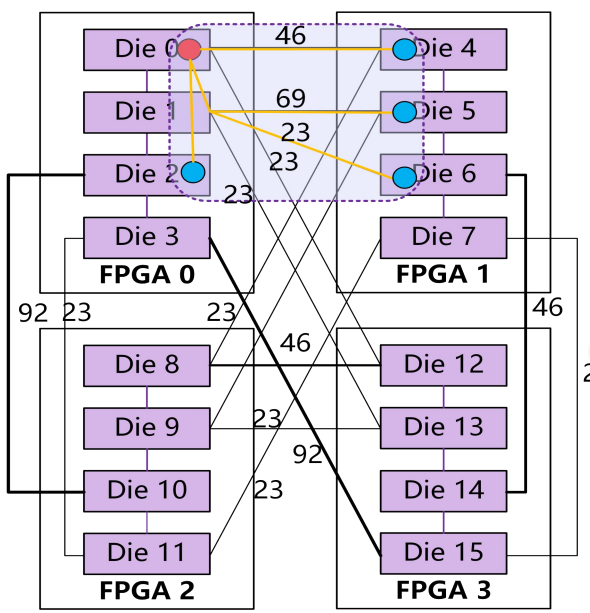
Introduction

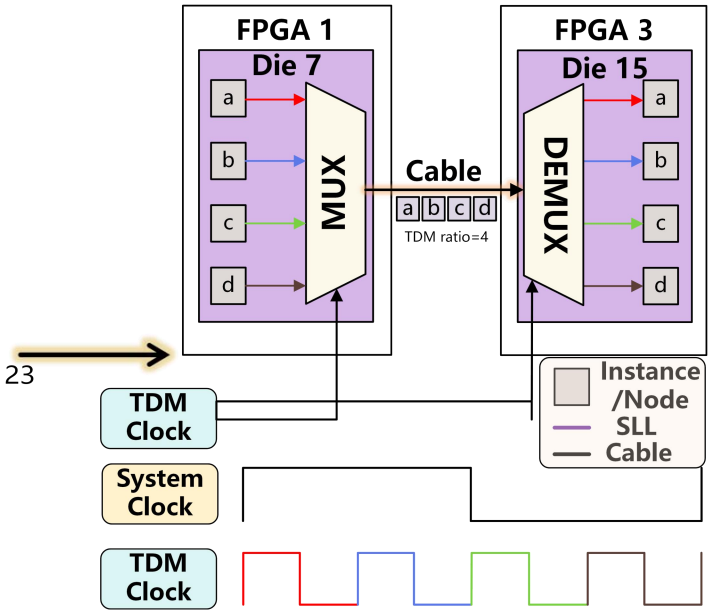
- Super Logic Regions (SLRs/dies) ⟷ high-capacity Super Long Lines (SLLs) ⇒ Die-Level System Routing.
- Limited link resources ⇒ congestion
- TDM increases delay ⇒ degrading system timing performance ↓

⇒ Routing and TDM assignment must be co-optimized to minimize the maximum system delay.



	Intra-FPGA	Inter-FPGA
Connection	Cable	SLL
Direction	one-way	bothway
Capacity	Scarce	Affluent
TDM	√	×
Delay	TDM delay	Die delay





Outline

- Introduction
- Preliminaries
- Routing
- TDM Assignment
- Experimental Results
- Conclusion

Preliminaries

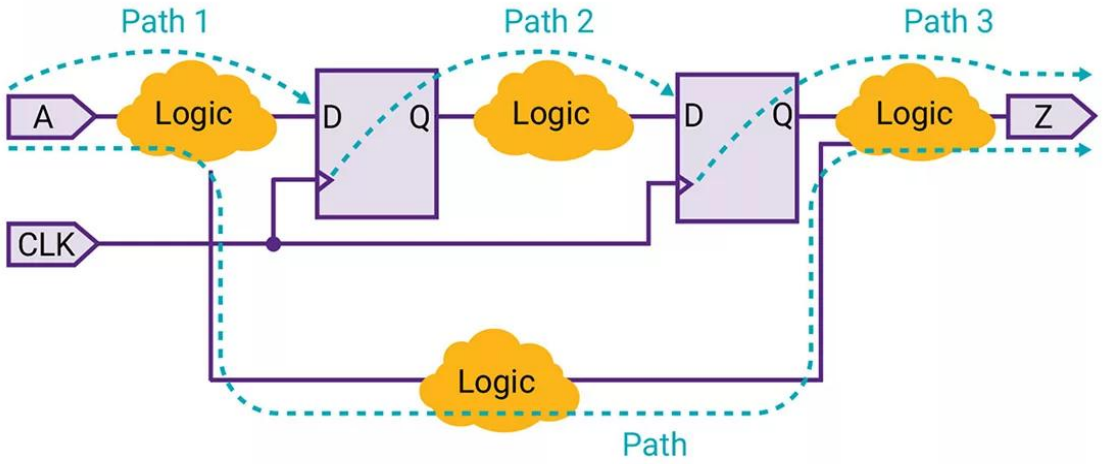
➢ Static Timing Analysis (STA)

- Traditional methods optimize proxy metrics:
  - TDM ratios sum
  - Net max delay
  - Routing cost

  not actual slack ⇒ inaccurate results
- STA:
  - Extract timing path
  - Verifies timing
- Routing and TDM assignment combined with STA
- Accurate timing-path-level delay and slack
- Use STA paths and slack for guidance

SYNOPSYS®

Timing paths



In the example, each logic cloud represents a combinational logic network. Each path starts at a data launch point, passes through some combinational logic, and ends at a data capture point.

Path	Startpoint	Endpoint
Path 1	Input port	Data input of a sequential element
Path 2	Clock pin of a sequential element	Data input of a sequential element
Path 3	Clock pin of a sequential element	Output port
Path 4	Input port	Output port

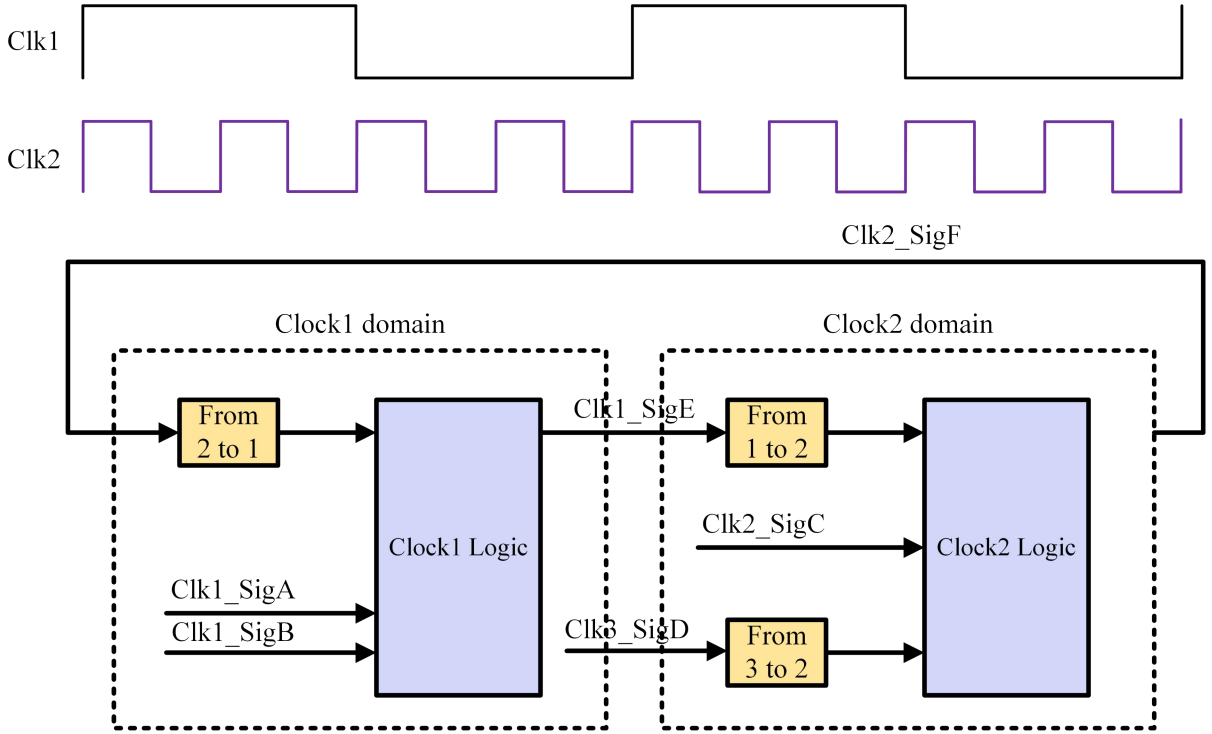
Preliminaries

➢ Static Timing Analysis (STA)

- Timing Path & Slack: Focus on register-to-register paths.

$$slack = clock\,period - delay$$

- Critical path ⇔ minimum slack
- Multi-clock domains: Multi-clock slack cannot be directly compared → use normalized slack



- Normalization formula

$$slack_n = \begin{cases} \dfrac{slack \times T_{\text{clk}}}{slack_{\max} \times T_{\text{clk}}^{\max}}, & slack \geq 0, \\ \dfrac{slack}{|slack_{\min}| \times T_{\text{clk}}}, & slack < 0, \end{cases}$$

Preliminaries

➢ Problem Formulation

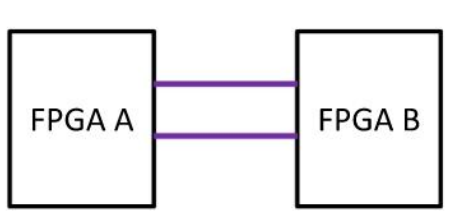
Objective

- Maximize worst timing path slack and minimize the worst timing path delay
- $\min_{\substack{T_n \subseteq E \\ r_e \in R}} \max_{p \in P} [tf_{ixed}(p) + \sum_{e \in p} delay(re)]$

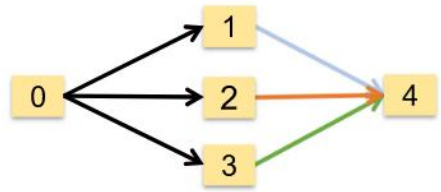
$$\text{s.t.} \quad \sum_{e \in E} \frac{1}{r_e} \leq IO_c, \quad \forall c,$$
$$T_n \text{ spans } S(n) \cup D(n), \quad \forall n,$$
$$r_e \geq \max_{n:e \in T_n} \text{demand}(n, e), \quad \forall e$$

Inputs

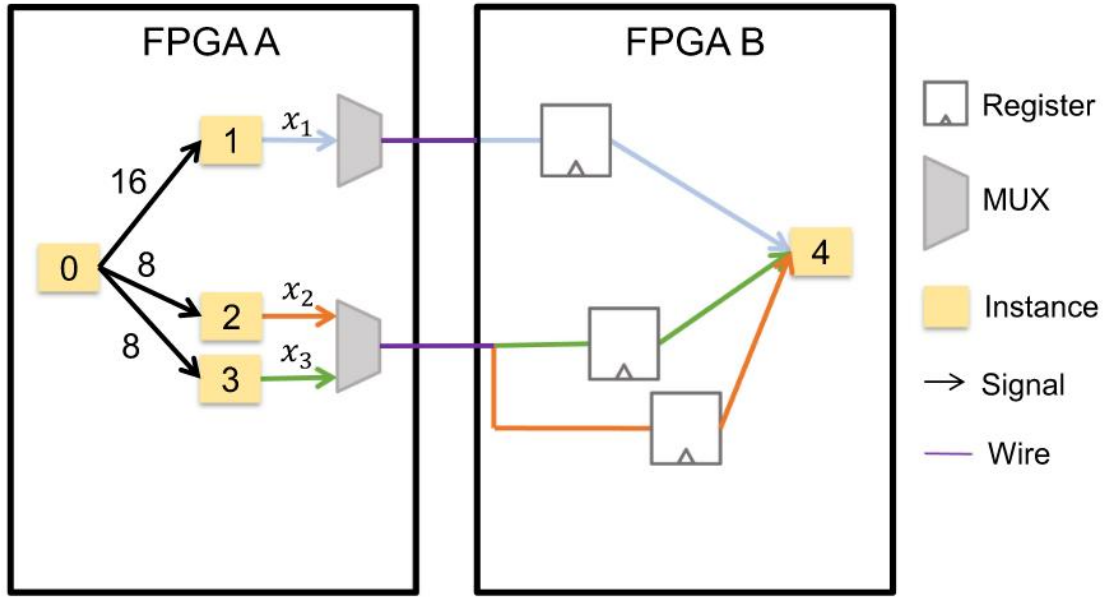
1. Die-level partitioned netlist
2. Multi-die topology
3. STA-reported timing paths information
4. Legal TDM ratio set and channel capacities



(a) Device.



(b) Netlist.



(c) The time-multiplexed I/Os when $\{x_1, x_2, x_3\} = \{1, 8, 8\}$.

Preliminaries

➢ Problem Formulation

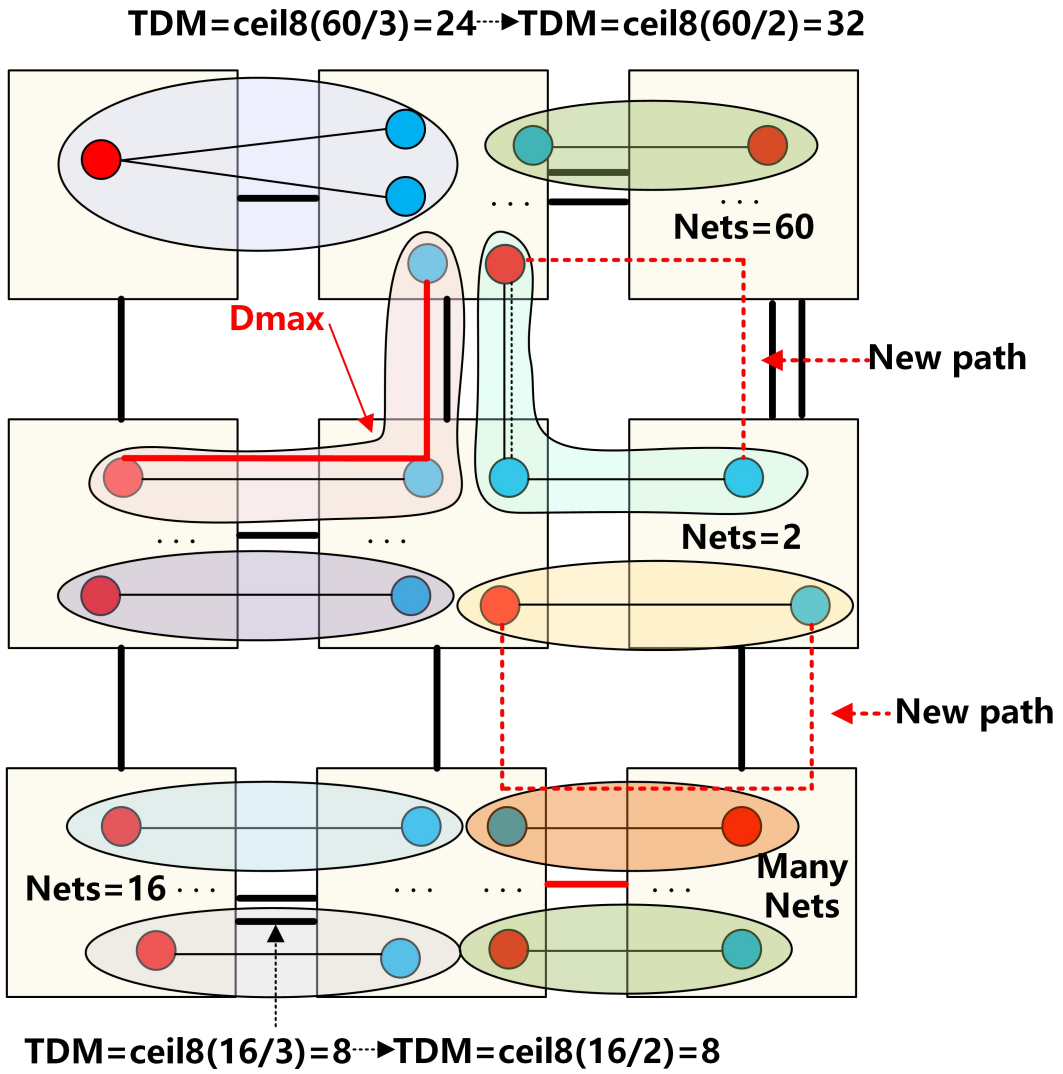
Objective

- Maximize worst timing path slack and minimize the worst timing path delay
- $\min\limits_{\substack{T_n \subseteq E \\ r_e \in R}} \max\limits_{p \in P} [tf_{ixed}(p) + \sum_{e \in p} delay(re)]$

$$\text{s.t.} \quad \sum_{e \in E} \frac{1}{r_e} \leq IO_c, \quad \forall c,$$
$$T_n \text{ spans } S(n) \cup D(n), \quad \forall n,$$
$$r_e \geq \max_{n:e \in T_n} \text{demand}(n, e), \quad \forall e$$

Inputs

1. Die-level partitioned netlist
2. Multi-die topology
3. STA-reported timing paths information
4. Legal TDM ratio set and channel capacities

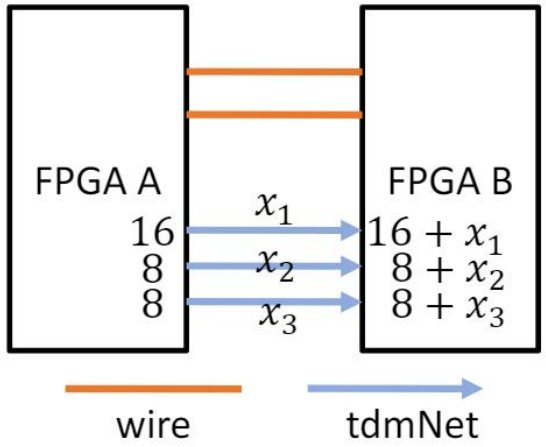


Preliminaries

➢ Challenge

Key Constraints

- Topology constraints
- Direction constraints
- Capacity constraints
- TDM constraints
- Timing constraints:
  - Full path delay = fixed logic delay + inter-die delay + TDM-induced delay
  - Timing path slack = clock period - full path delay

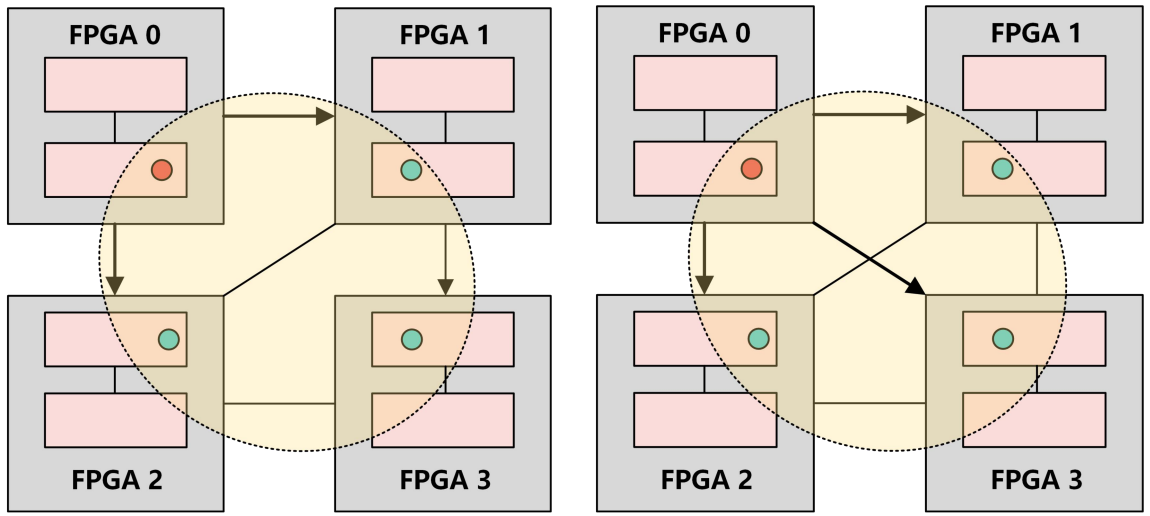


NP-Hard

Core Challenge

Routing and TDM are coupled:

- Routing affects which channels are used
- TDM ratios affect channel delay
- Need a joint optimization to achieve timing closure under limited interconnect resources.



Preliminaries

➢ Related Work

Routing

Originated from the 2019 ICCAD Contest.

Most prior studies focus on FPGA-level routing

- Steiner tree approximation routing
- Hybrid maze routing with MTST

Recent die-level work (2023 EDA Elite Challenge)

- Net-based cost metrics
- Simplified timing models

⇒ Limitation:

- Poor alignment with industrial timing requirements.

TDM Assignment

- Steiner tree timing-driven TDM assignment
- Profiling-guided multiplexing
- ILP-based TDM allocation
- Lagrangian relaxation–based TDM optimization

⇒ Limitation:

- Non-unified optimization objectives
- IgnoreDie-level structure

Outline

- Introduction
- Preliminaries
- Routing
- TDM Assignment
- Experimental Results
- Conclusion

Routing

➢ Overview and Workflow

1. Timing Path Compression

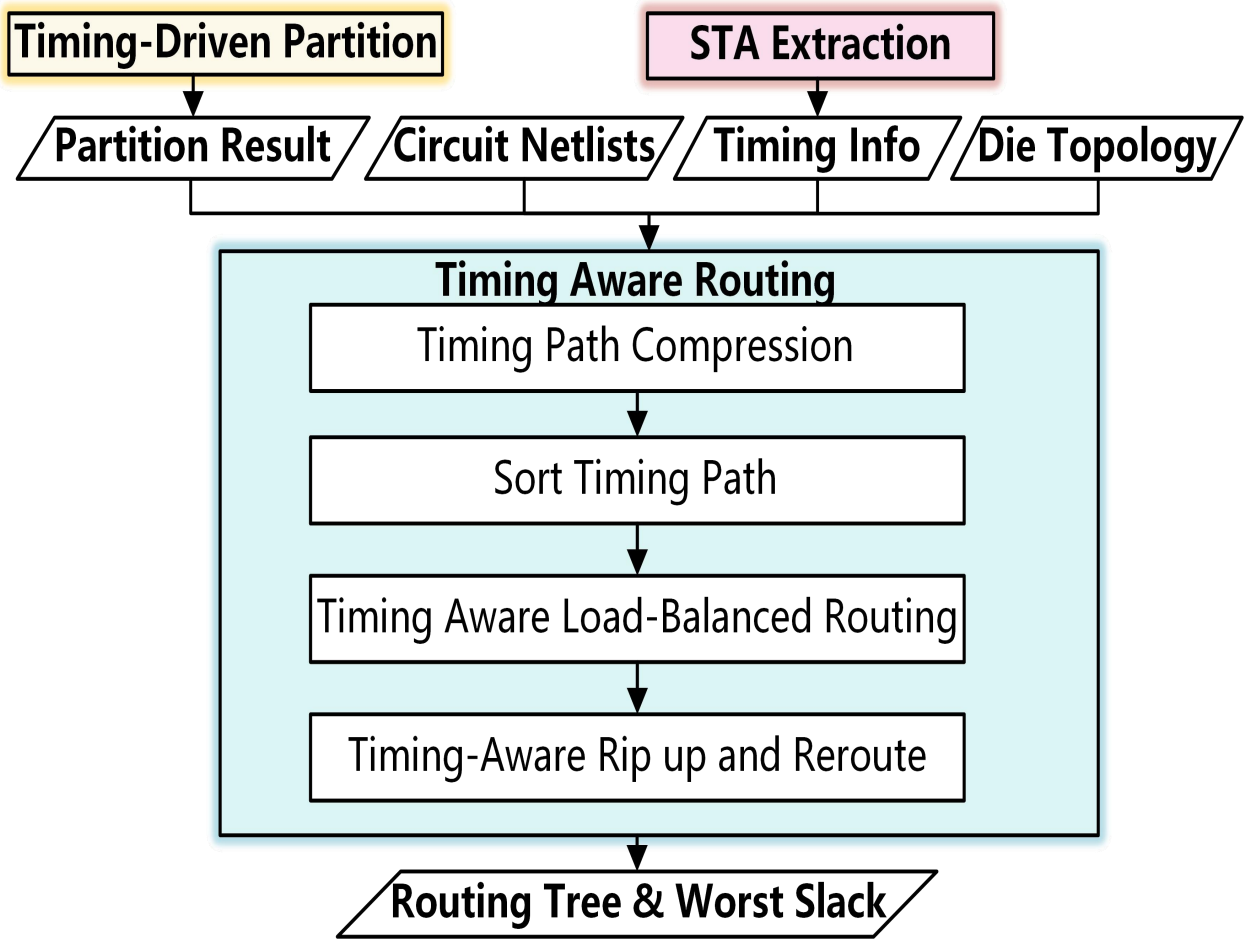
- Merge STA paths with cross-FPGA cut signatures
- Preserve critical timing while reducing path count

2. TLR

- Sort nets by normalized slack and cost
- Dijkstra with timing + congestion-aware weights
- Direct critical nets to low-delay routes

3. TRR

- Identify current worst paths
- Reroute its related nets to improve slack
- Accept changes only if timing or capacity improve



Routing

➢ Timing Path Compression

- STA reports millions of timing paths ⇒ direct timing-driven routing infeasible.
- Paths with same cut signature have identical delay.

$$\sigma(p) = (\widehat{e_1}, \widehat{e_2}, \ldots, \widehat{e_k})$$

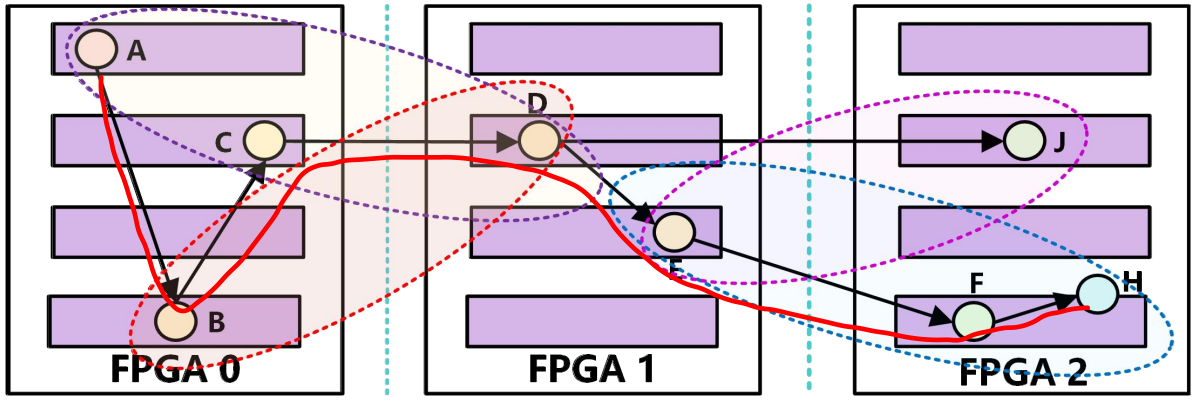
- Groups paths by $\sigma(p)$, keeps one with min slack.

$$P' = \{p^* | p^* = arg \max_{p \in P,\, \sigma(p)=\sigma} t_{fixed}(p), \forall \sigma\}$$

- Reduce timing-path set by > 90%, keeping worst timing path slack intact.

$$a_{max}(P) = amax(P')$$

⇒ Scalable timing-aware routing without losing critical info.



Design	Cut	STA Path	Cut Timing Path	Compression	↓%
Case1	1055	3361	2175	228	89.5%
Case2	6253	442775	225199	3193	98.6%
Case3	8805	2278156	101624	8271	91.9%
Case4	8048	894690	172336	6029	96.5%
Case5	16498	1211204	110522	8323	92.4%

Routing

➢ Timing-Aware Load-Balanced Routing

1. Floyd–Warshall computes all-pairs shortest delays.
   - Flow info determines majority traffic direction on SLLs.
2. Sorted nets by normalized slack; critical paths are routed first.

   $Criticality = a \times delay + b \times Fanout + c \times Span$

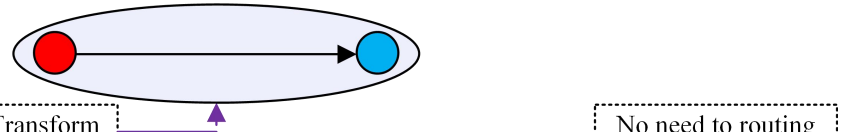
1. Routing uses dynamic weights reflecting delay and load.

$$w_{v_i,v_j} = \begin{cases} \infty, & (v_i, v_j) \notin E, \\ d_{\text{die}}, & (v_i, v_j) \in E_{\text{SLL}}, \\ \text{delay}(r_{v_i,v_j}), & (v_i, v_j) \in E_{\text{Cable}}, \end{cases}$$

   - Dijkstra search picks balanced paths for timing and resource use.

   $Cost = w \times (1 + \lambda_{cong} \times load^{\alpha}) + \lambda_{cong} \times jump$

⇒ Critical nets use low-delay channels, non-critical traffic avoids hotspots.



Algorithm 1 Timing-Aware Load-Balanced Routing (TLR)

Input: Topology weights W (Eq. (6)), nets $\mathcal{N}$, driving die $S(n)$, load dies set $D(n)$ for each net n, die topology graph $G(V, E)$
Output: Routing tree $\mathcal{T}$

```
dist ← ∞, next ← −1 // initialize
for all pair (i, j) with W_ij < ∞ do
    dist[i][j] ← W_ij; next[i][j] ← j
Compute dist[i][j] and next[i][j] for every die pair via a single Floyd–Warshall pass on G(V, E).
Lock SLL direction by majority flow using dist
for all cut timing path p reported by STA do
    slack_n(p) ← normalize Slack(p) // Using Eq. (2)
Sort N by ascending slack_n(p) and predicted_delay from D^(0)
for n ∈ N do // nets sorted by timing path slack
    // Dijkstra on current W
    Initialize a min-priority queue Q and push the source S(n) with distance 0
    Apply Dijkstra's algorithm over current edge weights W to compute shortest paths from S(n)
    Record the predecessor of each node in an array prev
    // back-trace every sink
    for d ∈ D(n) do
        while d ≠ −1 and ¬visited[d] do
            Add (prev[d], d) to T; U_{prev[d],d} ← U_{prev[d],d} + 1
            Update W_{prev[d],d} with Eq. (6)
            visited[d] ← T; d ← prev[d]
```

Routing

➢ Timing-Aware Load-Balanced Routing

1. Floyd–Warshall computes all-pairs shortest delays.
   - Flow info determines majority traffic direction on SLLs.
2. Sorted nets by normalized slack; critical paths are routed first.

   $Criticality = a \times delay + b \times Fanout + c \times Span$

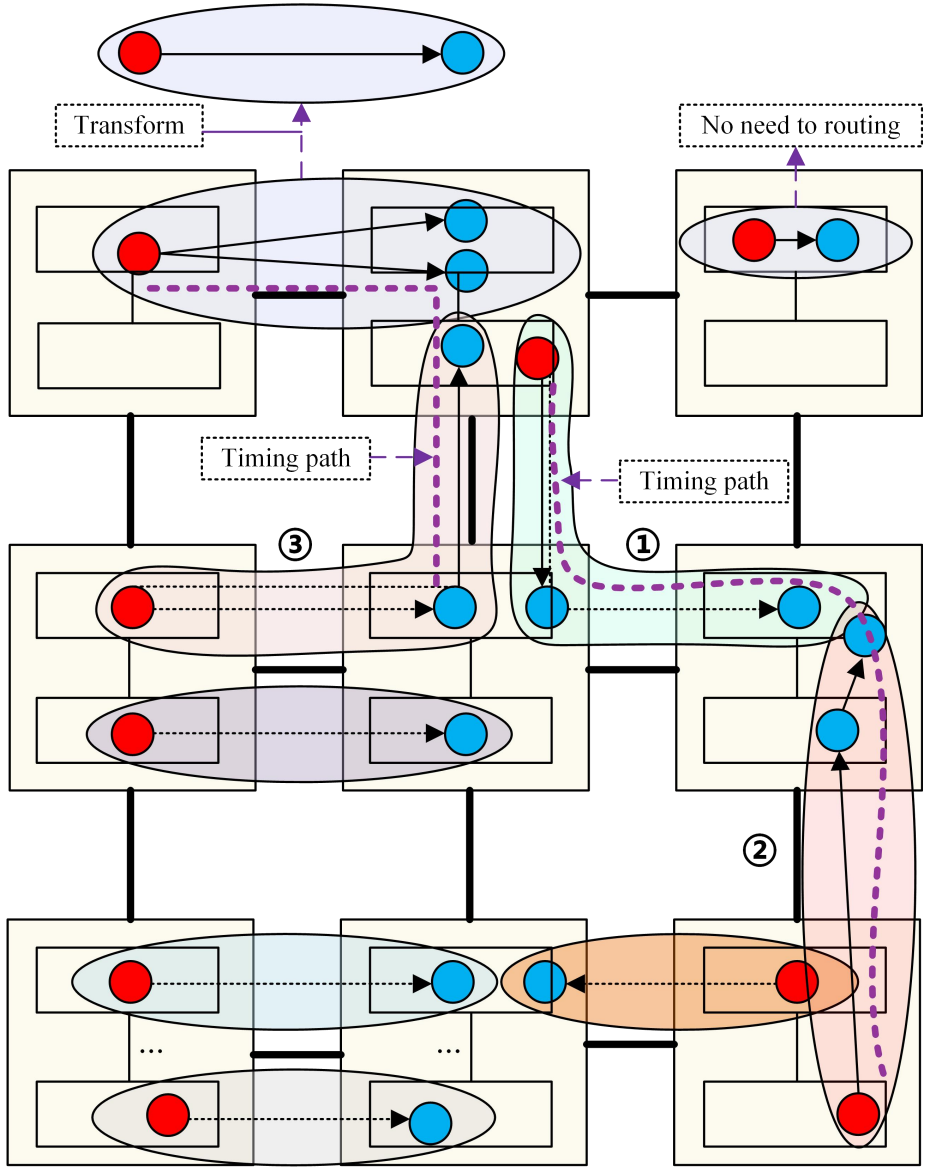
1. Routing uses dynamic weights reflecting delay and load.

$$w_{v_i,v_j} = \begin{cases} \infty, & (v_i, v_j) \notin E, \\ d_{\text{die}}, & (v_i, v_j) \in E_{\text{SLL}}, \\ \text{delay}(r_{v_i,v_j}), & (v_i, v_j) \in E_{\text{Cable}}, \end{cases}$$

   - Dijkstra search picks balanced paths for timing and resource use.

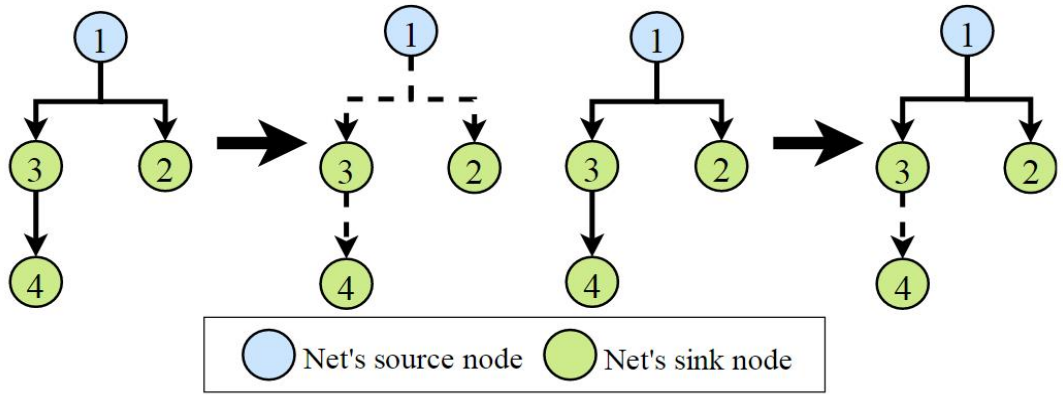
   $Cost = w \times (1 + \lambda_{cong} \times load^{\alpha}) + \lambda_{cong} \times jump$

⇒ Critical nets use low-delay channels, non-critical traffic avoids hotspots.



Routing

➢ Timing-Aware Rip-up and Reroute (TRR)



(a)Net-based rip-up (b)Pin-based rip-up

I. Pick the current worst-slack timing path from STA

II. Rip up the related cut nets

III. Reroute by banning path

IV. Accept only if:

- ➢ net delay and slack is improved
- ➢ global Dmax does not increase

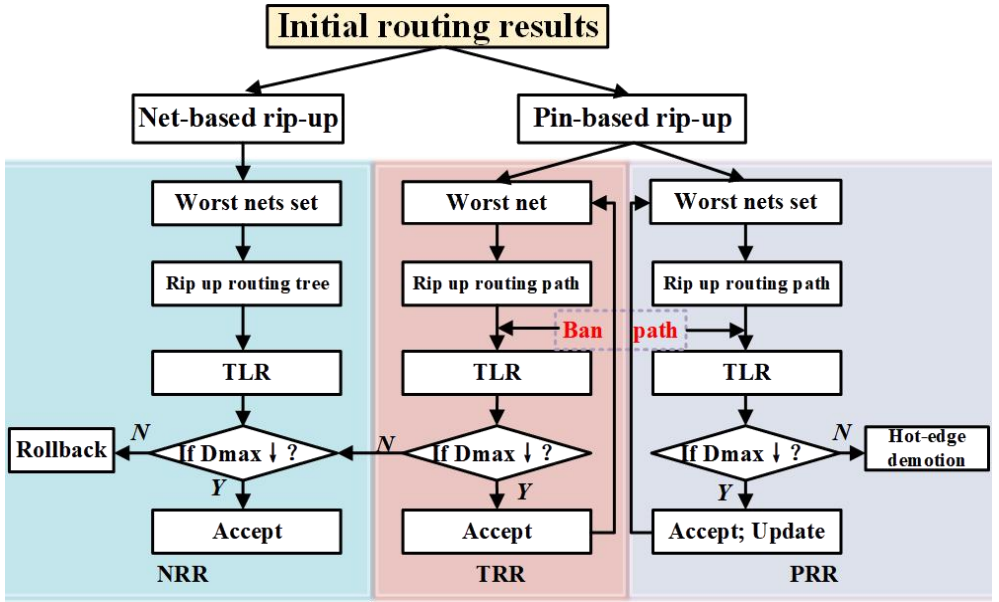
V. Early-stop strategies

Algorithm 2 Timing-Aware Rip-up and Reroute (TRR)

Input: Timing paths $\mathcal{P}$, cut nets $\mathcal{N}$, cost matrix C, capacity matrix U, penalty factors $\lambda_{\text{tdm}}, \lambda_{\text{die}}$, reroute rounds R
Output: Updated routing trees $\mathcal{T}$, path slacks $\{\text{slack}_p\}$

```
Initialize best slack S*, best cut cost (C*_tdm, C*_die)
for r = 1 to R do
    p* ← arg min_{p∈P} slack_p // critical path
    N* ← extract N on p* // optional expansion
    Backup routing state and rip up N* from T, C, U
    for n ∈ N* do
        route n by TLR with current C; update T, C, U
    Recompute {slack_p}; S_new ← min_p slack_p
    if S_new < S* and cut costs not worse then
        Accept update, set S* ← S_new
    else
        Rollback all changes
        if cut cost worsened then
            λ_tdm, λ_die ← α(λ_tdm, λ_die)
        else
            break
```

⇒ TRR refines cut nets with timing paths to improve slack.



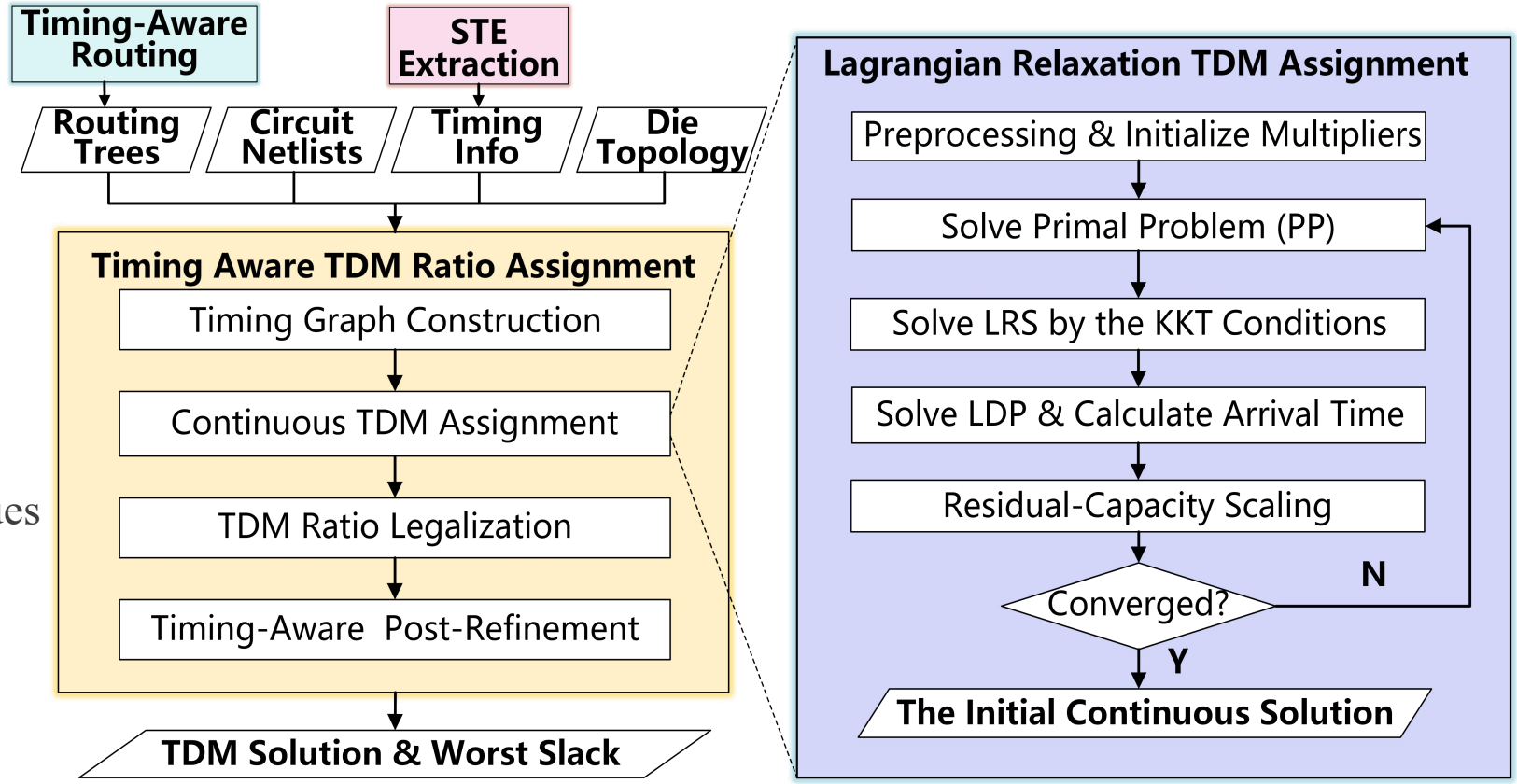
Outline

- Introduction
- Preliminaries
- Routing
- TDM Assignment
- Experimental Results
- Conclusion

TDM Assignment

➢ Overview and Workflow

1. **Timing Graph Construction**
   - Construct a timing DAG
   - Models full path delays
2. **Continuous Optimization**
   - Solve for optimal TDM ratios
   - Balance timing and capacity
3. **Ratio Legalization**
   - Map ratios to discrete vendor values
   - Meet direction and IO constraint
4. **Post-Refinement**
   - Adjust ratios on critical edges
   - Reduce worst timing path delay



TDM Assignment

➢ Timing Graph Construction

- Convert routing results amd timing path to timing DAG:

I. Introduces virtual source and sink nodes and zero-delay virtual edges to unify all timing paths.

II. Other Nodes: dies

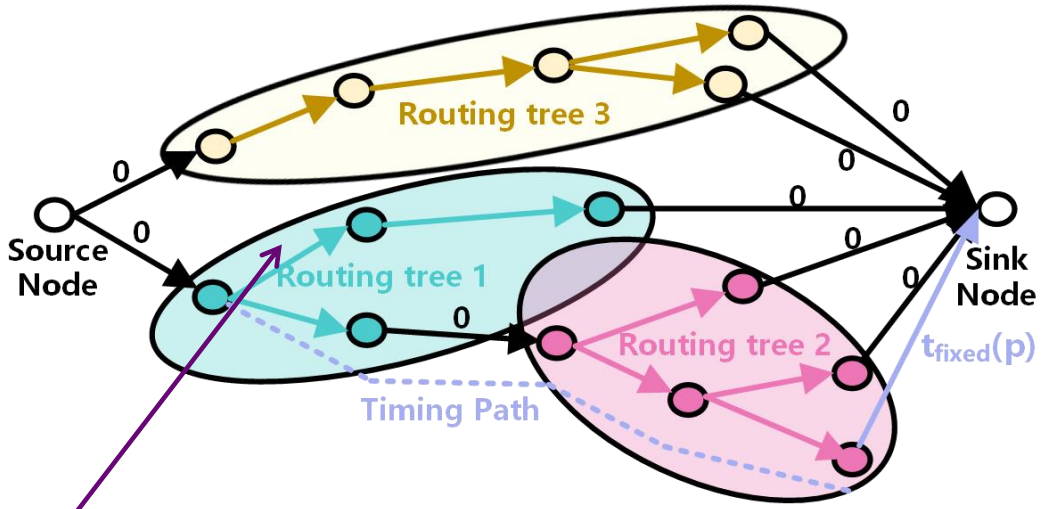
III. SLL edges: die delay

IV. cable edges: TDM-induced delay

V. Compressed paths mapped to DAG paths, preserving delay information.

VI. Routing edges not in STA paths affect performance. Trees added to timing graph via zero-delay edges.

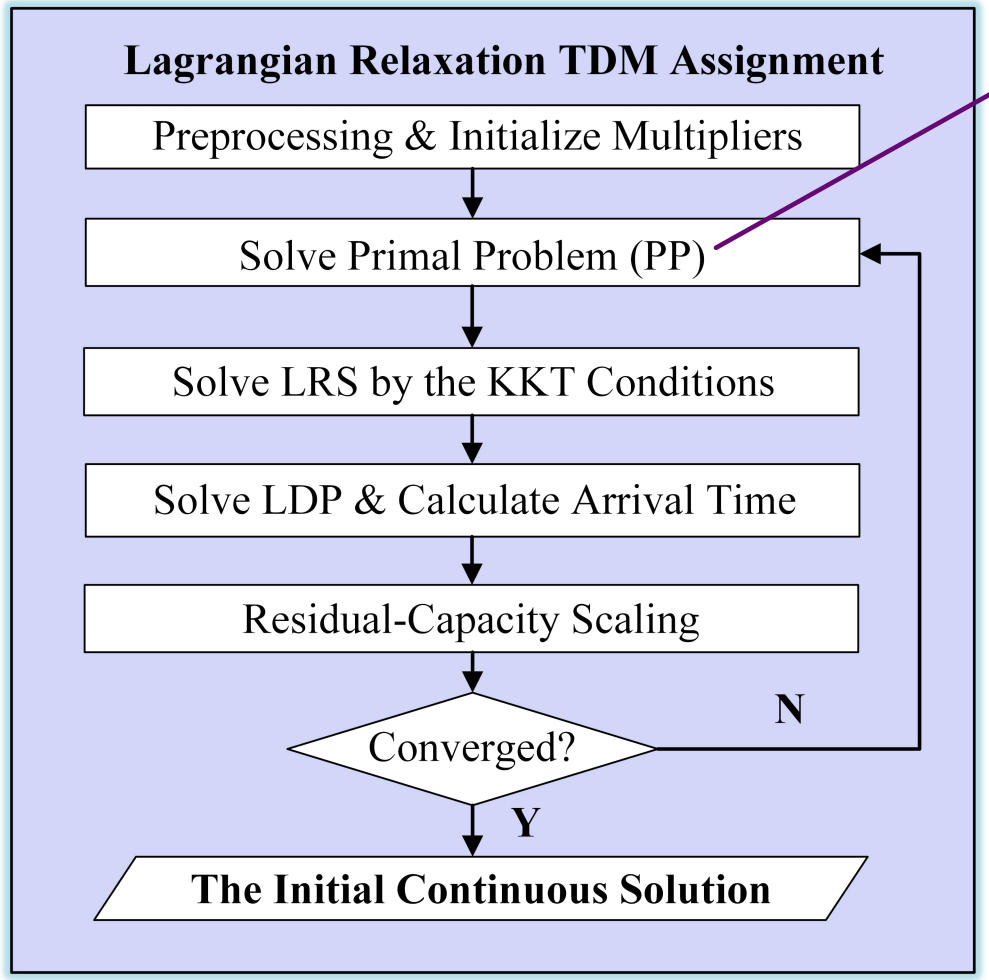
⇒ Enables arrival time computation, basis for TDM optimization.



Timing-directed acyclic graph (DAG).

TDM Assignment

➢ Continuous TDM Assignment



1. Relax discrete ratios → continuous variables

   - Objective: minimize the worst-path arrival A on the timing DAG.

$$\min_{\{r_e\},A} \quad A$$

$$\text{s.t.} \quad a_{\text{sink}(p)} \le A, \quad \forall p \in P',$$

$$a_{\text{sink}(p)} = t_{\text{fixed}}(p) + \sum_{\hat{e}\in\hat{E}_{\text{Cable}}} (\alpha + \beta r_{\hat{e}}),$$

$$\sum_{\hat{e}\in\hat{E}_{\text{Cable}}} \frac{1}{r_{\hat{e}}} \le IO_{\hat{e}}, \quad \forall \hat{e} \in E_{\text{Cable}},$$

$$1 \le r_{\hat{e}} \le r_{\max}, \quad \forall \hat{e} \in E_{\text{Cable}},$$

2. Lagrangian relaxation

   - Multipliers μ for timing constraints and λ for capacity constraints.

$$L_{\mu,\lambda}(r,a,A) = A + \sum_{(i,j)\in\hat{E}_{\text{Cable}}} \mu_{(i,j)}(a_i + \alpha + \beta r_{(i,j)} - a_j)$$

$$+ \sum_{p\in P'} \mu_p(a_{\text{sink}(p)} - A) + \sum_{e\in\hat{e}} \lambda \left(\sum_{\hat{e}\in\hat{E}_{\text{Cable}}} \frac{1}{r_{\hat{e}}} - IO_{\hat{e}} \right).$$

3. KKT conditions → closed-form for r_e

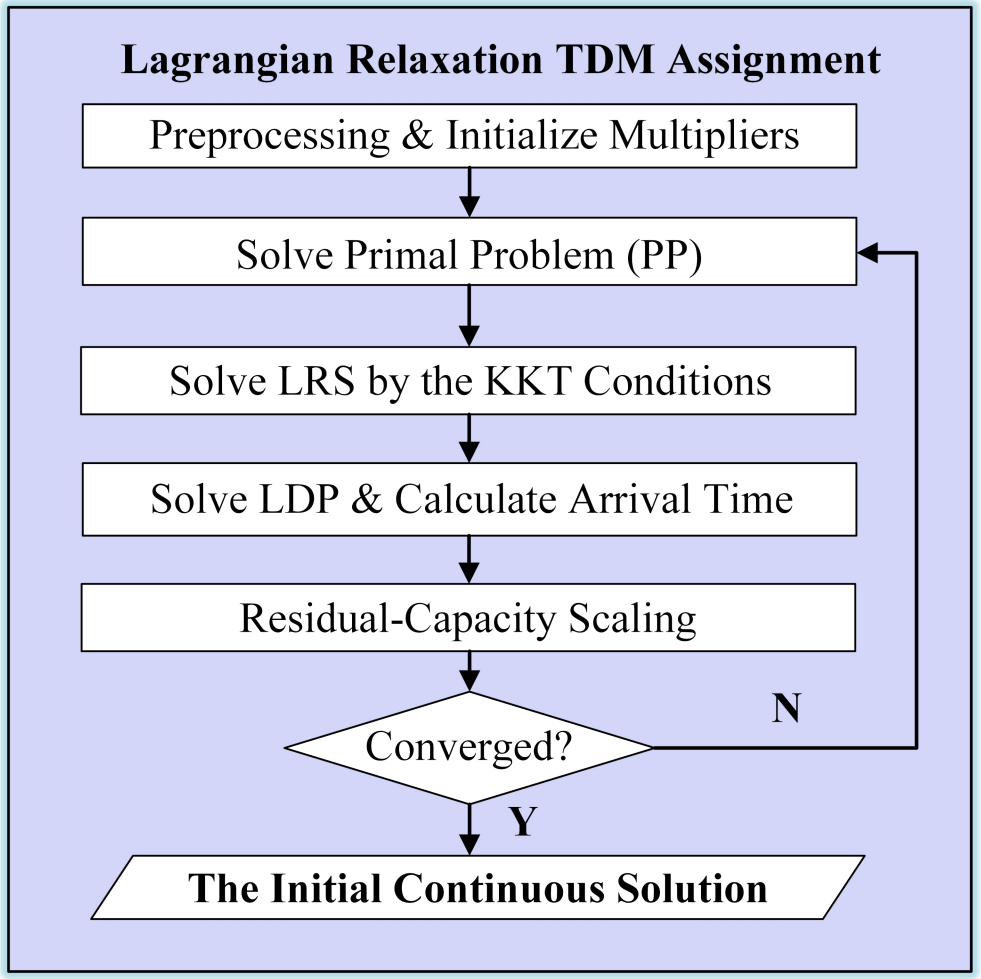
$$\frac{\partial L}{\partial r_{\hat{e}}} = 0 \implies r^{\star}_{\hat{e}} = \text{clip}\left(\sqrt{\frac{\lambda_{(\hat{e})}}{\beta\mu_{\hat{e}}}}, 1, r_{\max} \right),$$

$$\frac{\partial L}{\partial a_i} = 0 \implies \sum_{\text{in-edges}} \mu_{(k,i)} = \sum_{\text{out-edges}} \mu_{(i,j)},$$

$$\frac{\partial L}{\partial A} = 0 \implies \sum_{p\in P'} \mu_p = 1,$$

TDM Assignment

➢ Continuous TDM Assignment



4. Update multipliers

- μ (timing criticality): backward propagate on DAG by edge delay (delay-cost dist).

$$\mu_{(i,j)} = \frac{DC_{(i,j)}}{DC_j} \cdot \sum_{(j,k)\in\widehat{E}} \mu_{(j,k)}$$

- λ (cap pressure): set/update for equality; init $\lambda_{\hat{e}}$ via KKT μ and IO constraints.

$$\lambda_{\hat{e}} = \left(\frac{\sum_{(i,j)\in\widehat{E}} \sqrt{\beta\mu_{(i,j)}}}{IO_{(\hat{e})}}\right)^2$$

5. Update TDM ratio and Scale

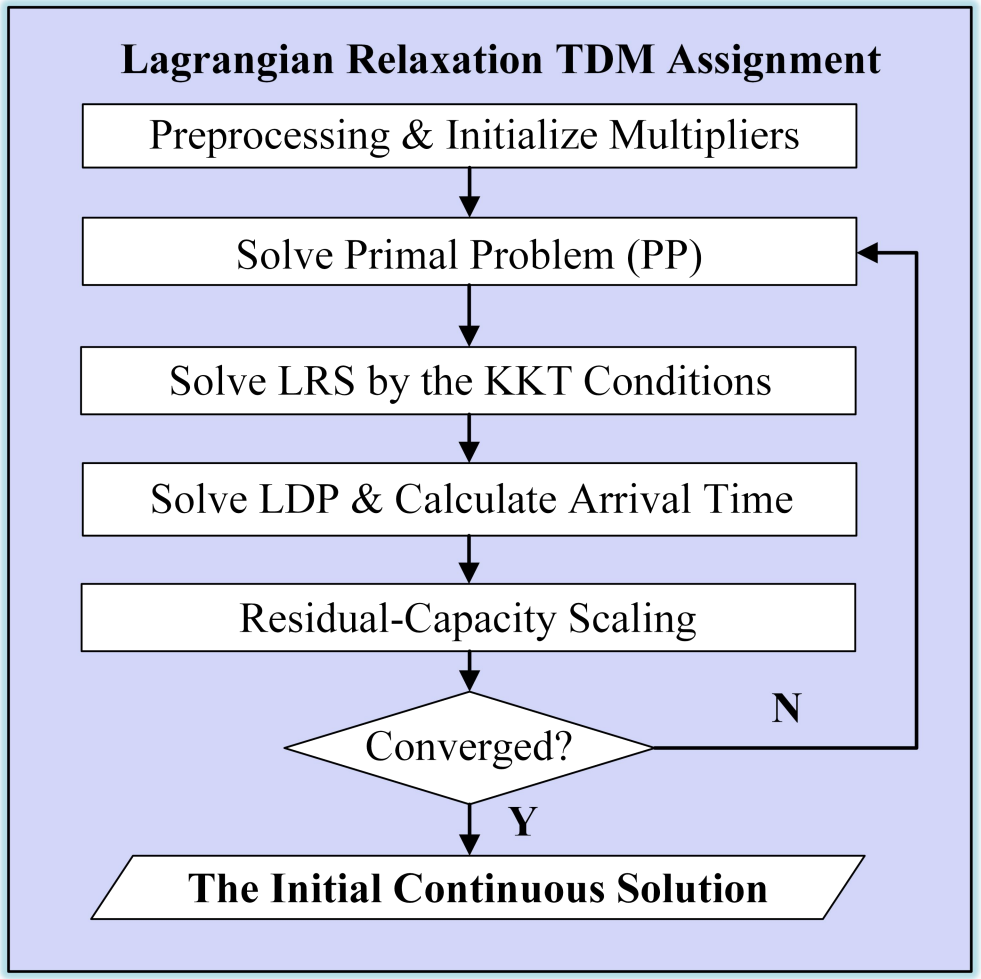
$$\frac{\partial L}{\partial r_{\hat{e}}} = 0 \Rightarrow r^*_{\hat{e}} = clip\left(\sqrt{\frac{\lambda_{\hat{e}}}{\beta\mu_{\hat{e}}}}, 1, r_{max}\right)$$

Algorithm 3 Continuous TDM Ratio Optimization Process

Input: Timing graph $T(\hat{V}, \hat{E})$
Output: Optimal continuous TDM ratio assignment $r^*_{\hat{e}}$
1: Initialize dual variable μ by Eq. (17)
2: **while** not converged and iteration count $<$ max iterations **do**
3: Update λ based on μ using Eq. (19)
4: Update TDM ratio r using Eq. (13)
5: Calculate arrival time a by Eq. (8), store if improved
6: Update μ based on timing slack using Eq. (17)
7: Scale residual capacity by γ using Eq. (20)

TDM Assignment

➢ Continuous TDM Assignment



4. Update multipliers

- μ (timing criticality): backward propagate on DAG by edge delay (delay-cost dist).

$$\mu_{(i,j)} = \frac{DC_{(i,j)}}{DC_j} \cdot \sum_{(j,k)\in\hat{E}} \mu_{(j,k)}$$

- λ (cap pressure): set/update for equality; init $\lambda_{\hat{e}}$ via KKT μ and IO constraints.

$$\lambda_{\hat{e}} = \left(\frac{\sum_{(i,j)\in\hat{E}} \sqrt{\beta\mu_{(i,j)}}}{IO_{(\hat{e})}}\right)^2$$

5. Update TDM ratio and Scale

$$\gamma = \frac{\sum_{r^*_{\hat{e}}>1} \frac{1}{r^*_{\hat{e}}} - n_1}{IO_c - n_1}, r^*_{\hat{e}} \leftarrow \gamma r^*_{\hat{e}}$$

Algorithm 3 Continuous TDM Ratio Optimization Process

Input: Timing graph $T(\hat{V}, \hat{E})$
Output: Optimal continuous TDM ratio assignment $r^*_{\hat{e}}$
1: Initialize dual variable μ by Eq. (17)
2: **while** not converged and iteration count $<$ max iterations **do**
3: Update λ based on μ using Eq. (19)
4: Update TDM ratio r using Eq. (13)
5: Calculate arrival time a by Eq. (8), store if improved
6: Update μ based on timing slack using Eq. (17)
7: Scale residual capacity by γ using Eq. (20)

TDM Assignment

➢ TDM Ratio Legalization

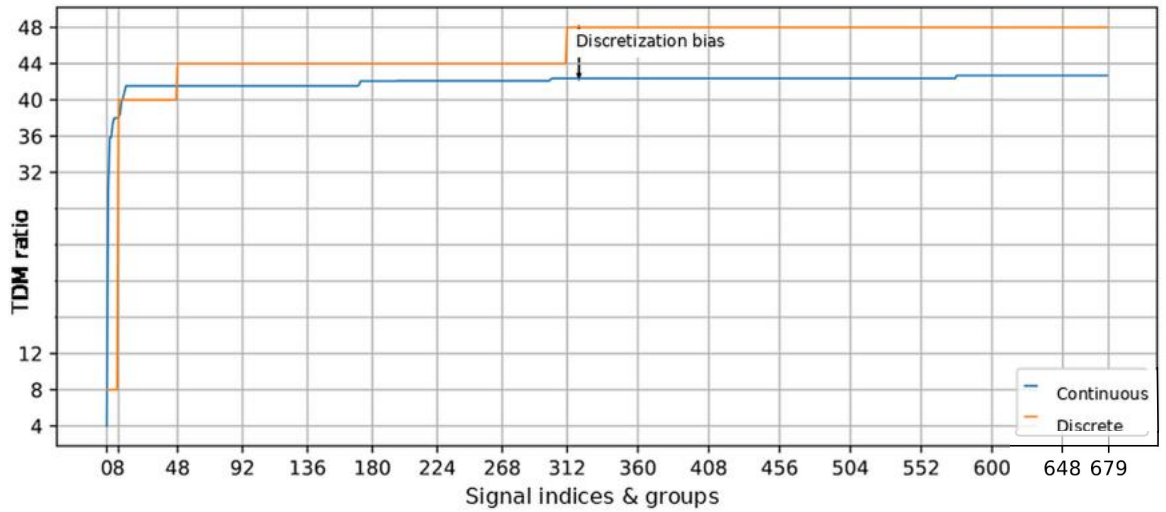
Algorithm 4 TDM Legalization and Post-Refinement

Input: Continuous ratios r_c, grouped TDM signals $T = \{T_d\}$ by direction d, slack bound b
Output: Legal discrete TDM ratios r^d

for each direction d with signals T_d **do**
  Compute total usage: $u_d \leftarrow \sum_{e \in T_d} \frac{1}{r_e^c}$
  Allocate physical channels: $p_d \leftarrow \max(1, \lfloor u_d + 0.5 \rfloor)$
for each direction d **do**
  Sort r_c in T_d as $r_1^*, \ldots, r_{|T_d|}^*$ ascending
  $n \leftarrow 0$, $i \leftarrow 1$, $m \leftarrow |T_d|$
  while $i \leq m$ **do**
    $choice \leftarrow \max\{r \in \mathcal{R} \mid r_i^* \leq r \leq r_i^* + b\}$
    for $j \leftarrow i$ **to** $i + choice - 1$ **do**
      $r_j^d \leftarrow choice$
    $i \leftarrow i + choice$; $n \leftarrow n + 1$
for each direction group T_d **do**
  Initialize lower/upper bounds $l = -b$, $u = b$
  Binary-search $b^* \in [l, u]$: repeatedly bisect the interval, repack T_d with margin m, and shrink the interval until the resulting channel count equals the budget p_d (tolerance ϵ)
  Assign r^d for T_d with $b^* \leftarrow r$ // $r^d \in \mathcal{R}$
repeat
  Compute slacks: $s_{(p,q)} = s_q + a_q - (a_p + \text{delay}_{(p,q)})$
  Trace critical path (where $s_e = 0$)
  for each critical inter-FPGA edge e_i **do**
    for each e_j in same direction with $r_j^d < r_i^d$ **do**
      $\Delta\text{delay} \leftarrow \beta(r_i^d - r_j^d)$
      if $\Delta\text{delay} \leq s_{e_j}$ **then**
        Swap r_i^d, r_j^d, update a_v and slacks
        break
until no critical path improves

- Groups edges by direction, respects IO capacity.
- Min-deviation packing assigns discrete ratios near continuous solutions.
- Uses binary search for min deviation margin meeting capacity constraints.

⇒ Produces legal TDM solution preserving timing.



TDM Assignment

➢ Timing-Aware Post-Refinement

Algorithm 4 TDM Legalization and Post-Refinement

Input: Continuous ratios r_c, grouped TDM signals $T = \{T_d\}$ by direction d, slack bound b
Output: Legal discrete TDM ratios r^d

for each direction d with signals T_d **do**
  Compute total usage: $u_d \leftarrow \sum_{e \in T_d} \frac{1}{r_e^c}$
  Allocate physical channels: $p_d \leftarrow \max(1, \lfloor u_d + 0.5 \rfloor)$
for each direction d **do**
  Sort r_c in T_d as $r_1^*, \ldots, r_{|T_d|}^*$ ascending
  $n \leftarrow 0$, $i \leftarrow 1$, $m \leftarrow |T_d|$
  while $i \leq m$ **do**
    $choice \leftarrow \max\{r \in \mathcal{R} \mid r_i^* \leq r \leq r_i^* + b\}$
    for $j \leftarrow i$ **to** $i + choice - 1$ **do**
      $r_j^d \leftarrow choice$
    $i \leftarrow i + choice$; $n \leftarrow n + 1$
for each direction group T_d **do**
  Initialize lower/upper bounds $l = -b$, $u = b$
  Binary-search $b^* \in [l, u]$: repeatedly bisect the interval, repack T_d with margin m, and shrink the interval until the resulting channel count equals the budget p_d (tolerance ϵ)
  Assign r^d for T_d with $b^* \leftarrow r$ // $r^d \in \mathcal{R}$
repeat
  Compute slacks: $s_{(p,q)} = s_q + a_q - (a_p + \text{delay}_{(p,q)})$
  Trace critical path (where $s_e = 0$)
  for each critical inter-FPGA edge e_i **do**
    for each e_j in same direction with $r_j^d < r_i^d$ **do**
      $\Delta\text{delay} \leftarrow \beta(r_i^d - r_j^d)$
      if $\Delta\text{delay} \leq s_{e_j}$ **then**
        Swap r_i^d, r_j^d, update a_v and slacks
        break
until no critical path improves

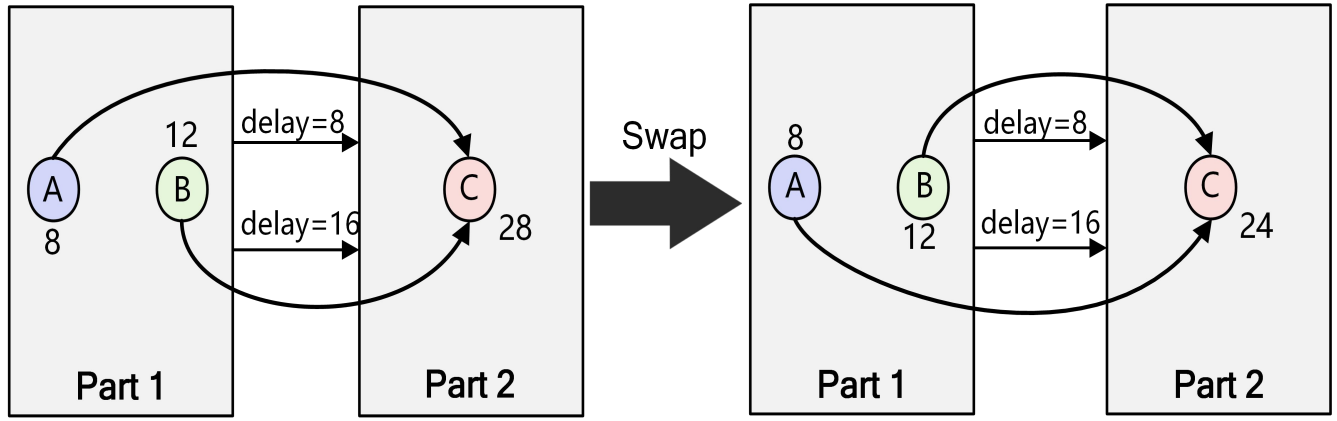
I. Evaluates slack on the legalized TDM solution.

II. Identifies critical paths and limiting edges.

III. Ratio swaps between critical and non-critical edges within the same direction group.

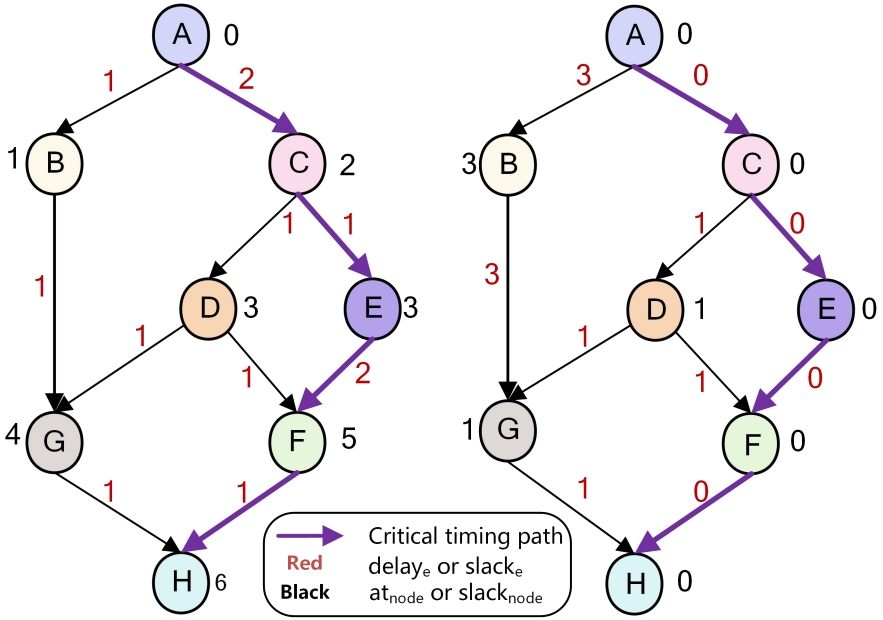
IV. Applies swaps reducing delay, not violating capacity.

⇒ Iteratively improves worst-path slack; keep TDM legal and stable.



TDM Assignment

➢ Timing-Aware Post-Refinement



Edge Delay and Node Arrival Time

Edge Slack and Node Slack

```
repeat
    Compute slacks: s_(p,q) = s_q + a_q − (a_p + delay_(p,q))
    Trace critical path (where s_e = 0)
    for each critical inter-FPGA edge e_i do
        for each e_j in same direction with r_j^d < r_i^d do
            Δdelay ← β(r_i^d − r_j^d)
            if Δdelay ≤ s_{e_j} then
                Swap r_i^d, r_j^d, update a_v and slacks
                break
until no critical path improves
```

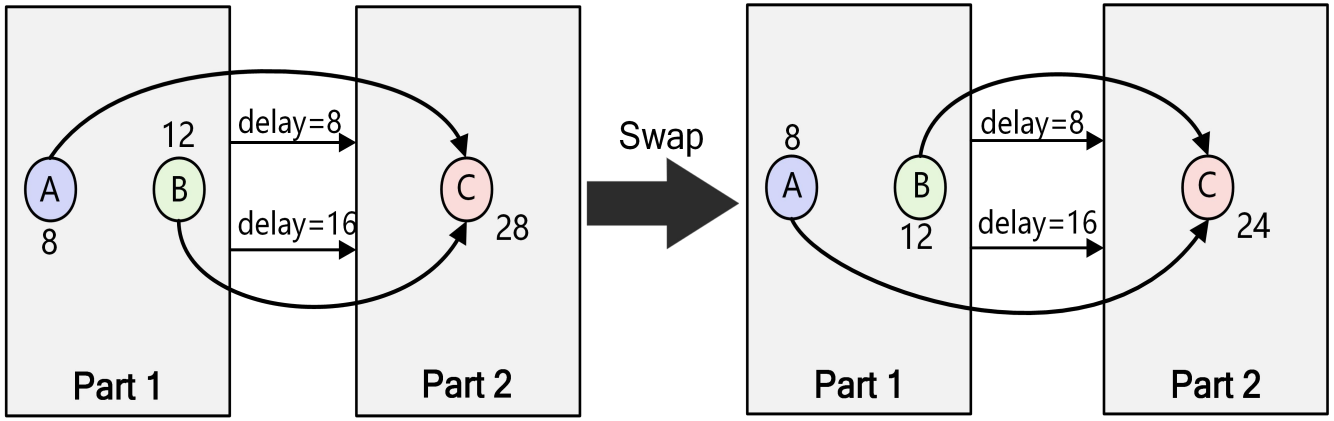
I. Evaluates slack on the legalized TDM solution.

II. Identifies critical paths and limiting edges.

III. Ratio swaps between critical and non-critical edges within the same direction group.

IV. Applies swaps reducing delay, not violating capacity.

⇒ Iteratively improves worst-path slack; keep TDM legal and stable.



Outline

- Introduction
- Preliminaries
- Routing
- TDM Assignment
- Experimental Results
- Conclusion

Experimental Results

➢ Benchmarks

■ Industry

Design	Node	Net	FPGA	Die	Clock Period(ns)
Industry1	6503	6500	4	16	2.00
Industry2	23120	23118	4	16	50.00
Industry3	1221292	1280002	4	16	50.00
Industry4	8653	9396	4	16	41.67,50.00
Industry5	25678	26574	4	16	37.04,62.50, 1000.00,30518.52
Industry6	1921478	2034595	4	16	10.93,20.00,39.92, 100.00
Industry7	3518579	3765663	4	16	50.00,100.00
Industry8	6267124	6480217	4	16	10.00,13.33,20.00, 83.33,500.00

■ Contest

Design	Node	Net	FPGA	Die	Clock Period(ns)
Case1	6	5	2	8	-
Case2	71	86	2	8	-
Case3	69	84	2	8	-
Case4	452	449	2	8	-
Case5	5084	5083	3	12	-
Case6	127100	145660	3	12	-
Case7	549700	76258	4	16	-
Case8	62846	86139	4	16	-
Case9	784814	871588	4	16	-
Case10	3066539	3324963	5	20	-

Experimental Results

➢ Industry Designs & Public Contest Benchmarks

Industry

- Tested on 8 large multi-die industrial circuits, with 6 multi-clock designs vs. Huang[1].
- Proposed method improves worst slack by an average of 98%.
- Up to 88% ↑ in single-clock designs.
- Up to 75% ↑ in the largest multi-clock systems.

Public Contest

- Evaluated on 10 official contest cases.
- Achieves significant delay reduction on large designs up to 140 ns improvement.
- Performs consistently even under simplified timing models.

TABLE III: The Worst Slack of Partitioned Circuits

Clock Domain	Design	Huang [19] (ns)	Ours (ns)	Improve (%)
Single	Industry1	−114.11	−12.68	88.89
	Industry2	33.88	33.88	0.00
	Industry3	−263.07	−82.47	68.64
Multi	Industry4	−262.81	−78.44	70.15
	Industry5	−11.57	55.86	582.65
	Industry6	−379.06	−189.06	50.14
	Industry7	−377.67	−184.12	51.23
	Industry8	−545.36	−131.24	75.94
			Avg.	**98.45**

TABLE IV: The Worst Delay Comparison on Contest Cases

Design	1st (ns)	2nd (ns)	3rd (ns)	Huang [19] (ns)	Ours (ns)	Improve (ns)
Case01	6.5	6.5	6.5	**6.5**	**6.5**	0.0
Case02	7.5	7.5	7.5	**7.5**	**7.5**	0.0
Case03	11.5	14.5	11.5	**11.5**	**11.5**	0.0
Case04	19.5	19.5	18.5	**18.5**	**18.5**	0.0
Case05	135.5	136.0	132.5	**130.0**	132.0	-2.0
Case06	213.0	260.0	287.0	202.0	**171.0**	31.0
Case07	84.5	102.5	**76.0**	77.5	78.0	-0.5
Case08	124.0	145.5	123.0	**111.5**	114.5	-3.0
Case09	150.0	196.5	177.0	147.5	**144.0**	3.5
Case10	4657.5	4745.0	5700.0	4740.5	**4600.5**	**140.0**
Avg.	–	–	–	–	–	**17.5**

1. C. Huang, P. Chu, S. Bi, R. Sun, and H. You, "System routing and tdm assignment optimization in multi-2.5d FPGA-based prototyping systems," in 2024 2nd International Symposium of Electronics Design Automation (ISEDA), 2024.

Experimental Results

➢ Ablation Study on Strategy Components

- **Necessity of Joint Optimization**
  - Routing and TDM are tightly coupled; improving one cannot compensate for the other.
  - Only the full flow (TL + TA) consistently achieves best slack.
- **Robustness Across Partitions**
  - Tested on multiple partition solutions; full flow always performs best.
  - Indicates strong robustness to topology, clock domain and partition variations.
- **Practical Implication**
  - Complete timing-aware framework aligns with real STA-based timing closure requirements.

TABLE II: The Worst Slack of the Partitioned Circuits under Different Strategies of Routing and TDM Assignment (unit: ns)

Strategy	Industry1		Industry2		Industry3		Industry4		Industry5		Industry6		Industry7		Industry8	
	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2
TL+TA (proposed)	**-12.68**	**-12.33**	**33.88**	**38.45**	**-82.47**	**-302.52**	**-78.44**	**-78.44**	**55.86**	**53.86**	**-188.72**	**-188.72**	**-184.12**	**-289.09**	**-131.24**	**-160.75**
TL+AA	**-12.68**	**-12.33**	**33.88**	**38.45**	-219.07	-878.82	-202.79	-154.58	2.43	0.43	-209.06	-246.35	-185.61	-359.09	-203.36	-217.00
SP+TA	**-12.68**	**-12.33**	**33.88**	36.45	-96.47	-597.38	**-78.44**	**-78.44**	**55.86**	**53.86**	-250.35	-250.35	-185.41	-317.20	-313.36	-185.39
SP+AA	**-12.68**	**-12.33**	**33.88**	**38.45**	-119.07	-828.81	-202.79	-208.01	2.43	0.43	-209.06	-290.35	-235.33	-463.20	-237.36	-261.92

TL+TA: timing-aware load-balanced router + timing-aware TDM; TL+AA: timing-aware load-balanced router + average assignment TDM; SP+TA: shortest-path router + timing-aware TDM; SP+AA: shortest-path router + average assignment TDM. P1 and P2 denote different circuit partition results.

Outline

- Introduction
- Preliminaries
- Routing
- TDM Assignment
- Experimental Results
- Conclusion

Conclusion

Main contributions:

- Timing-aware co-optimization framework for die-level routing and TDM assignment:
  - Die-level modeling with direction, capacity, delay, clock domains.
  - STA-guided routing that prioritizes critical timing paths and balances timing criticality and load.
  - Lagrangian-based TDM optimization over the timing graph with effective legalization and post-refinement.
- Improves worst timing path slack on both benchmarks.

THANKS !

Yijun Chen[1], Haoyuan Li[2], Chunyan Pei[2], Jianwang Zhai[1], Kang Zhao[1], and Wenjian Yu[2]

[1]Beijing University of Posts and Telecommunications

[2]Dept. Computer Science & Tech., BNRist, Tsinghua Univ.

Jan. 22, 2026