SYNOPSYS®
Silicon to Software™

Timing Driven Partition for Multi-FPGA Systems with TDM Awareness

Sin-Hong Liou,
Sean Liu,
Richard Sun,
Hung-Ming Chen

Outline

- Introduction
- Preliminary
- Problem Formulation
- Methodology
- Experiment results
- Conclusion

Outline

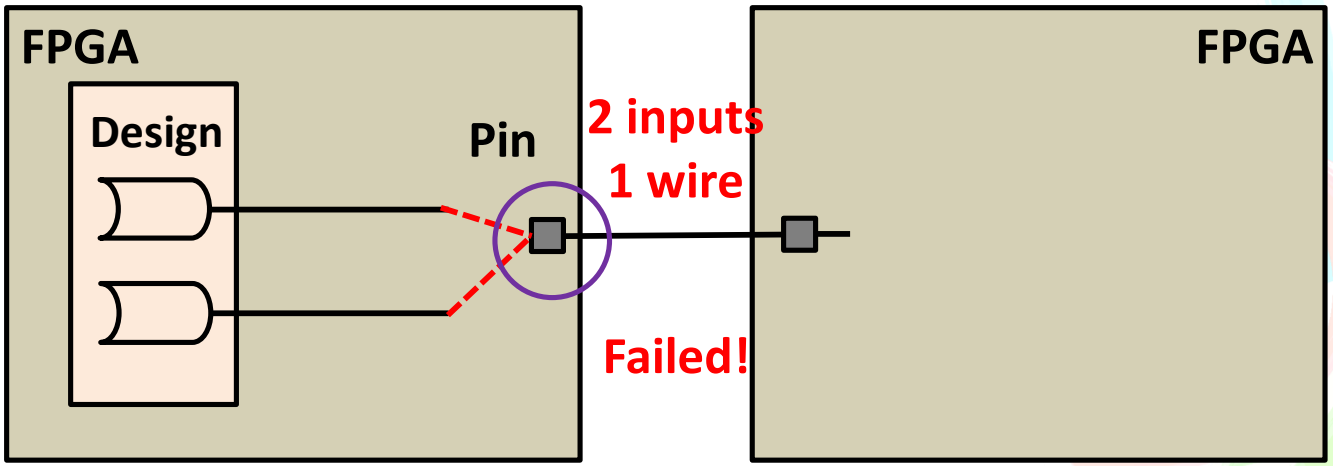
- **Introduction**
- Preliminary
- Problem Formulation
- Methodology
- Experiment results
- Conclusion

Multiple FPGA System

- More capacity than single FPGA
- Multiple FPGA system(**MFS**) is widely used for
  - Logic emulation
  - Fast prototyping of large design
  - Reconfigurable custom computing platform

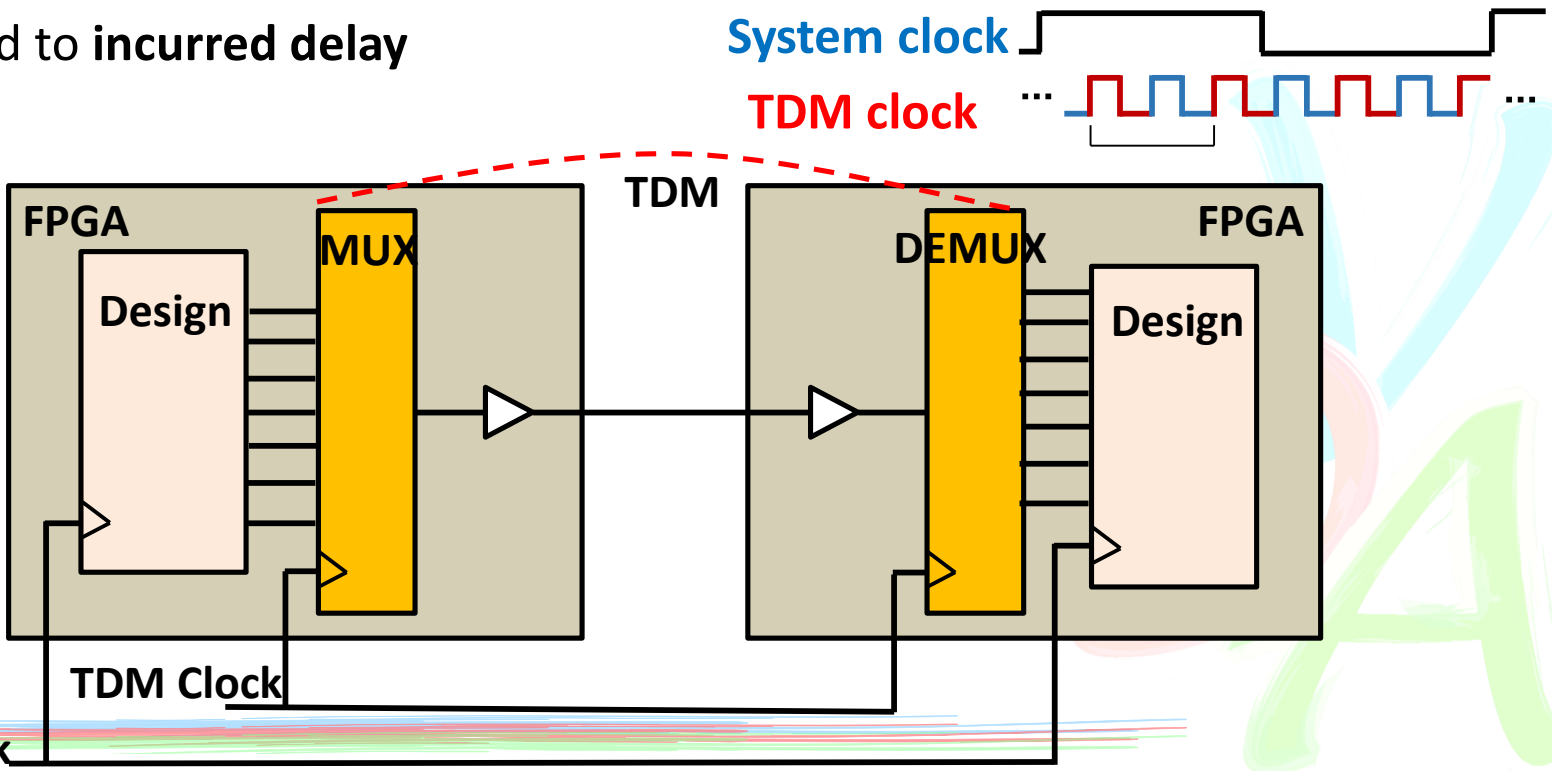
Multiple FPGA System

- MFS connected through **directed hardwired-connections**
- The limitation number on I/O pins between FPGAs will cause
  - Routing failure
  - Timing issue



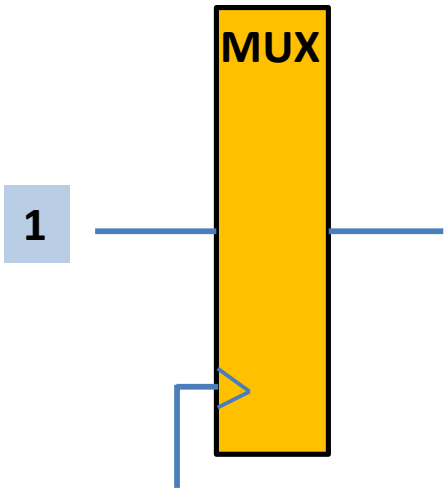
Time-Division Multiplexing (TDM)

- TDM is used to overcome the connection resource constraint
  - Mux two or more signals over a single wire
  - Use a faster asynchronous TDM clock
  - Lead to **incurred delay**

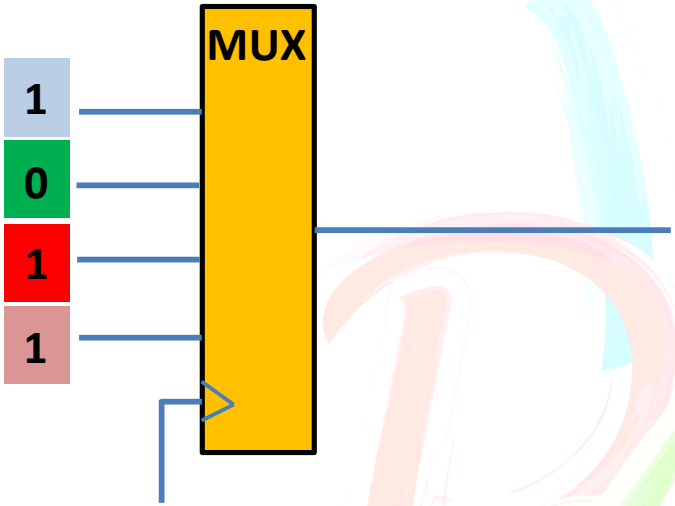


TDM Ratio

- **TDM ratio** is used to evaluate the impact of TDM
  - The number of shared signals using same multiplexer.
  - **Timing criticality** is heavily dependent on TDM ratio

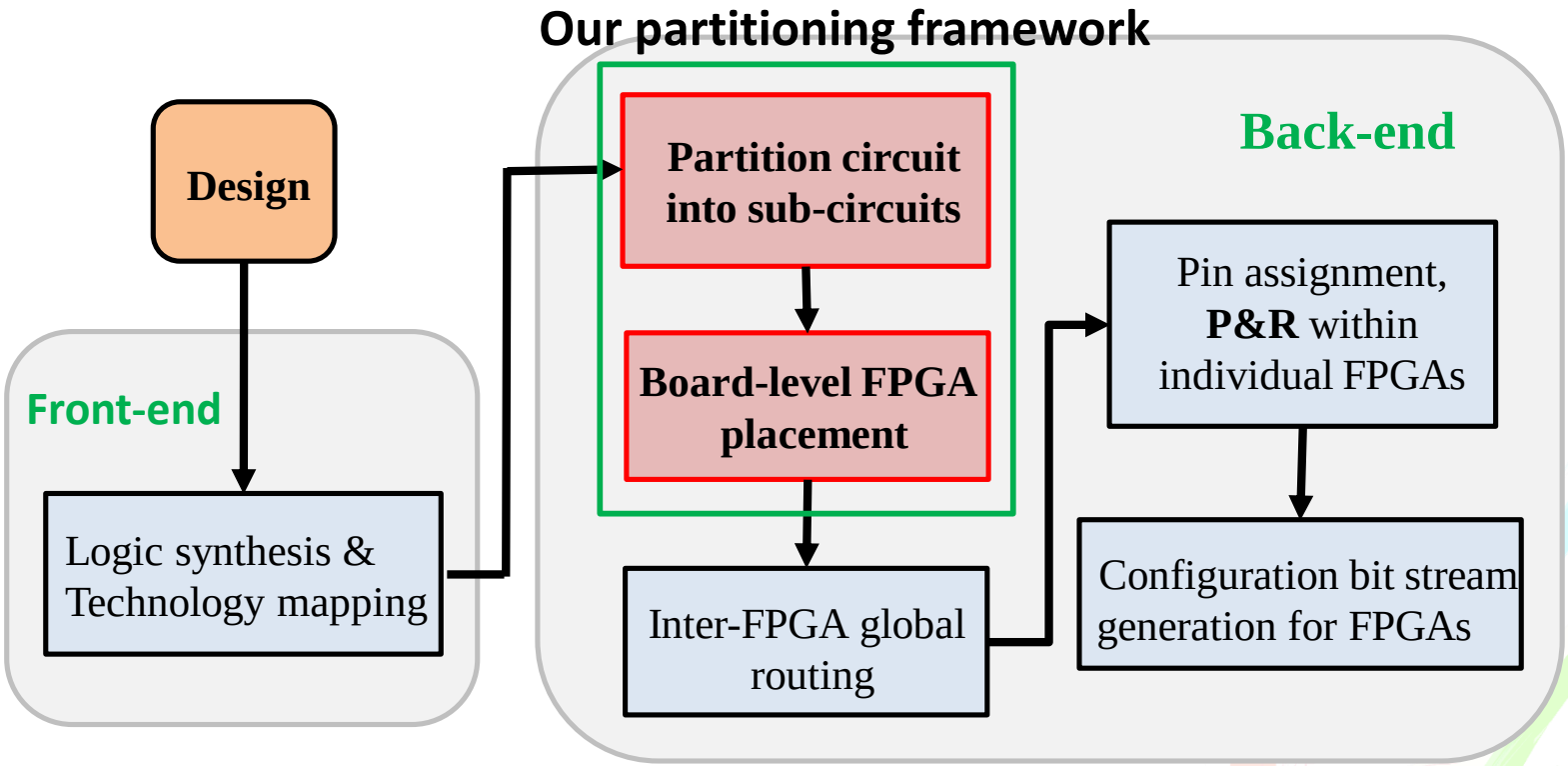


TDM ratio = 1 (Small Delay)



TDM ratio = 4 (Large Delay)

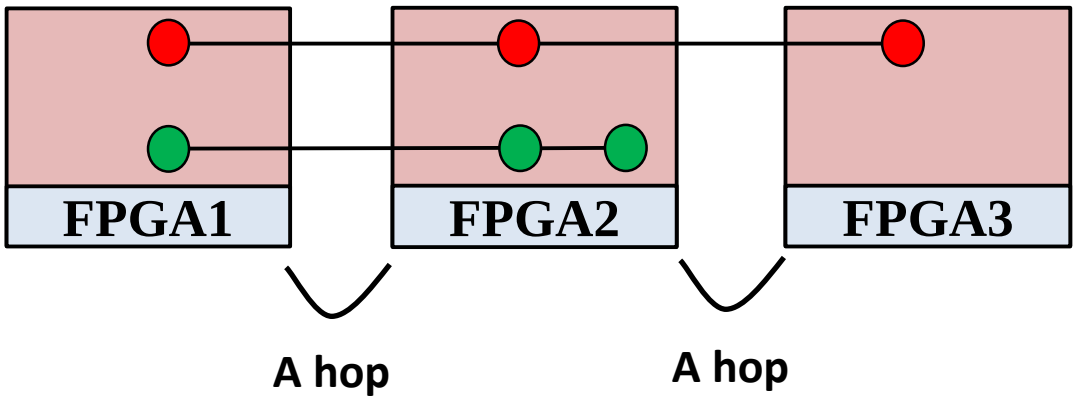
Compilation Flow of MFS



[1] M. A. Khalid, Routing architecture and layout synthesis for multi-FPGA systems. Ph. D. dissertation, Dept. of ECE, Univ. Toronto, 1999.

Motivation

- Performance is dominated by the most critical path
- Two factors that affect performance
  - **FPGA-hop** (crossing between adjacent FPGAs)
  - **Penalty** incurred by **#shared signals** on **FPGA-hop**

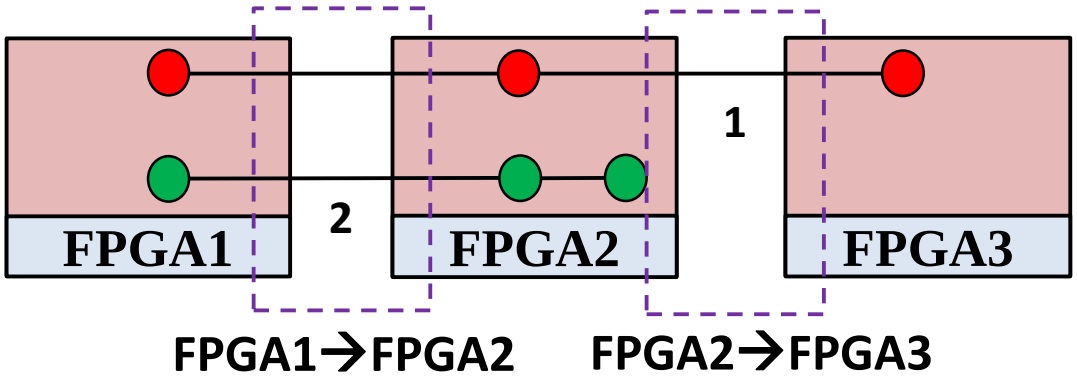


For red path,
#FPGA-hops = 2

For green path,
#FPGA-hops = 1

Motivation

- Performance is dominated by the most critical path
- Two factors that affect performance
  - **FPGA-hop** (crossing between adjacent FPGAs)
  - **Penalty** incurred by **#shared signals** on **FPGA-hop**

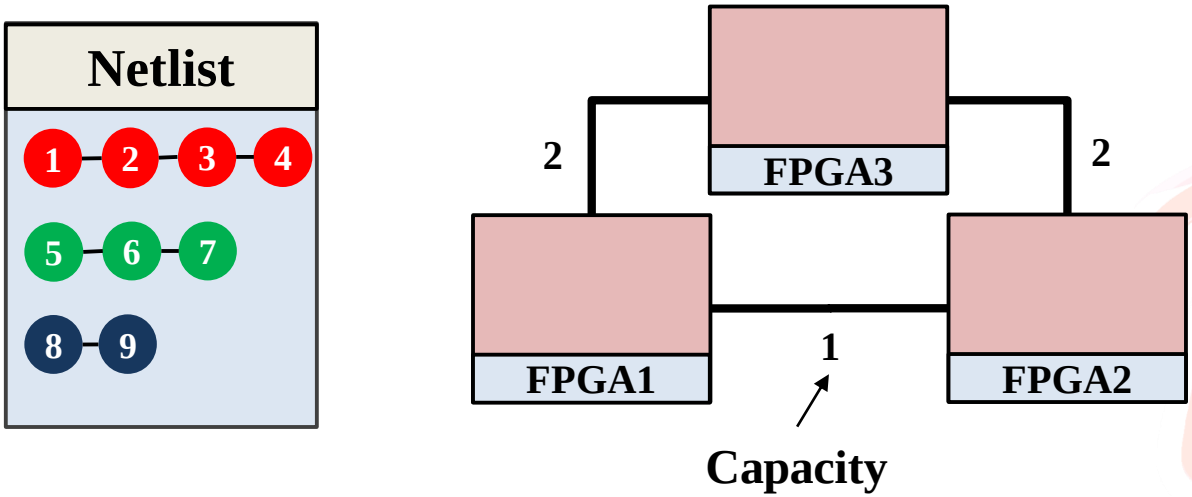


For red path,
#FPGA-hops = 2
Sum of penalty = 2+1 = 3

For green path,
#FPGA-hops = 1
Sum of penalty = 2

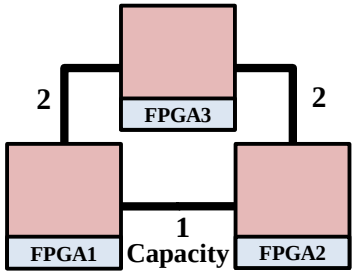
Motivating Example

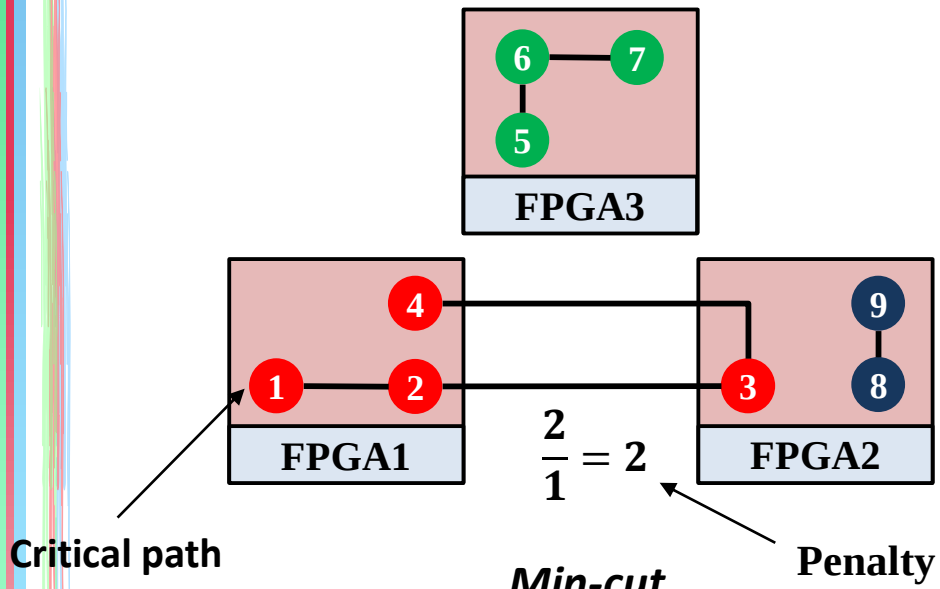
- Consider capacity between FPGA boards
  - **TDM penalty** is defined as $\frac{\#shared\ signals}{capacity}$
  - Example



Partition Is Not Aware of FPGA-Hop

- Example



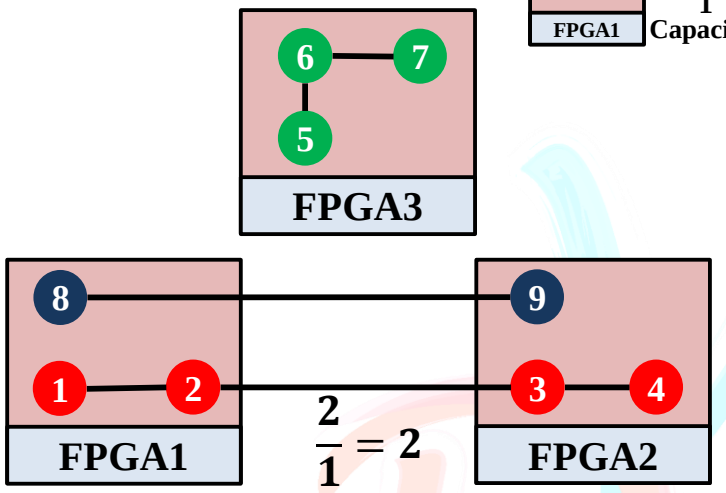


Min-cut

Cut size = 2

Max #FPGA-hop = 2

Max sum of penalty= 4



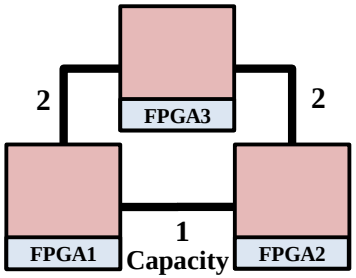
Path-aware

Cut size = 2

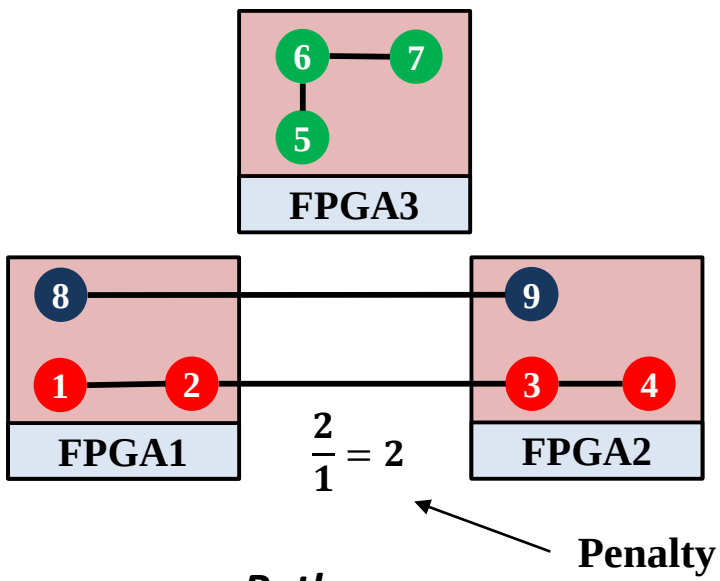
Max #FPGA-hop = 1

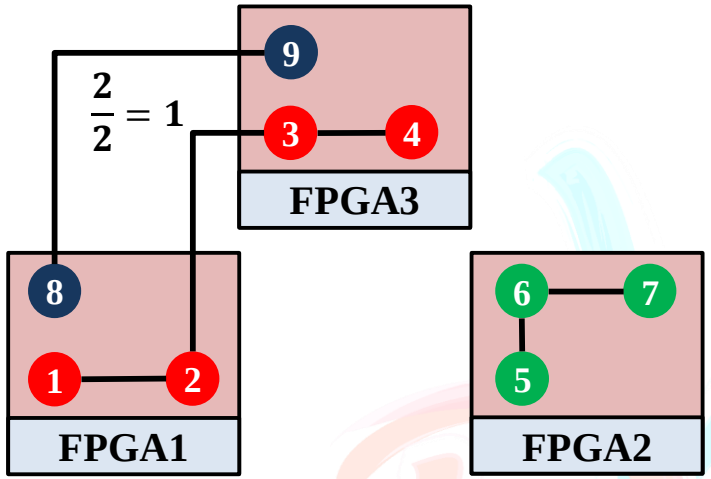
Max sum of penalty = 2

Partition Is Not Aware of TDM Penalty on FPGA-Hop



- Example





Path-aware

Cut size = 2

Max #FPGA-hop = 1

Max sum of penalty = 2

Path-aware + TDM-aware

Cut size = 2

Max #FPGA-hop = 1

Max sum of penalty= 1

Motivation Summary

- Improving cut size does not directly correlate to improve emulation performance
  - Optimize on **FPGA-hop**
  - Optimize on **TDM penalty**

Method	Cut size	#FPGA-hops	Sum of penalty
Min-cut	2	2	4
Path-aware	2	1	2
Path-aware + TDM-aware	2	1	1

Contributions

- Analytical Placement Guided Partitioning
  - Determine net weight by analytical placement
  - Optimize FPGA-hop on a set of path
- TDM optimization for Board-level FPGA placement
  - Consider hardware configuration and TDM impact
  - Estimate TDM penalty
  - Construct look-up table for delay between FPGA boards for efficiency

Outline

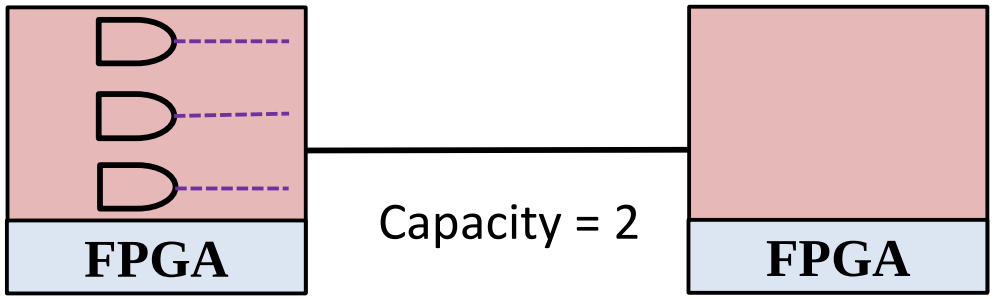
- Introduction
- **Preliminary**
- Problem Formulation
- Methodology
- Experiment results
- Conclusion

Delay Model

- Delay to cross adjacent FPGA boards
  - Given wire demand $f^G(u,v)$ and wire capacity $c^G(u,v)$, wire loading ratio $LR^G(u,v) = \frac{f^G(u,v)}{c^G(u,v)}$
  - Delay between adjacent FPGAs (u,v) :

$$d(u,v) = \begin{cases} \lceil LR^G(u,v) \rceil, & \lceil LR^G(u,v) \rceil \ is \ even \\ \lceil LR^G(u,v) \rceil + 1, & \lceil LR^G(u,v) \rceil \ is \ odd \end{cases}$$

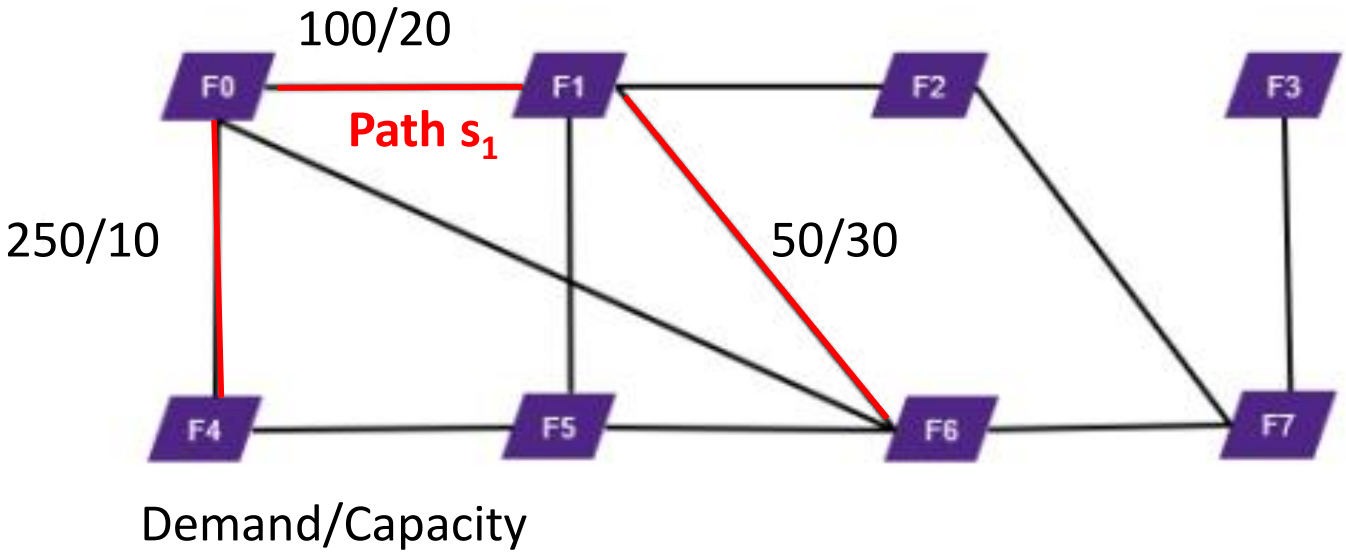
  - Defined as an **even number** due to multiplexing hardware implementation



$$d(u,v) = \left\lceil \frac{3}{2} \right\rceil = 2$$

Delay Model

- Delay function $d(s)$ is the summation of edge delay
  - $d(s_1) = \sum_{e^G(u,v)\ni s} d(u,v) = \left(\left\lceil\frac{250}{10}\right\rceil + 1\right) + \left(\left\lceil\frac{100}{20}\right\rceil + 1\right) + \left\lceil\frac{50}{30}\right\rceil$
    $= 26 + 6 + 2 = 34$



Previous Work

- Modern challenge issues for FPGA-based emulator [5]
  - Indicate that cut size is not an accurate indicator to measure the performance
- Overcome pin limitations in emulator [1][2]
- Multi-FPGA system platform
  - Routing architecture [3]
  - Multi-FGPA prototyping platform generation [4]

[1] J. Babb, R. Tessier, M. Dahl, S. Z. Hanono, D. M. Hoki, and A. Agarwal, "**Logic emulation with virtual wires**," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 16, pp. 609–626, June 1997.

[2] J. Babb, R. Tessier, and A. Agarwal, "**Virtual wires: overcoming pin limitations in fpga-based logic emulators**," in Proceedings IEEE Workshop on FPGAs for Custom
Computing Machines, pp. 142–151, April 1993.

[3] M. A. Khalid, **Routing architecture and layout synthesis for multi-FPGA system**s. Ph. D. dissertation, Dept. of ECE, Univ. Toronto, 1999.

[4] Q. Tang, **Methodology of Multi-FPGA Prototyping Platform Generation**. PhD thesis, Universite Pierre et Marie Curie - Paris VI, 2015.

[5] W. N. Hung and R. Sun, "**Challenges in large fpga-based logic emulation systems**," in Proceedings of the International Symposium on Physical Design, ISPD '18, (New York, NY, USA), pp. 26–33, ACM, 2018.

Previous Work

- Simultaneous partitioning and signals grouping for TDM in a 2.5D FPGA-based systems [11]
  - Consider both partitioning and signal grouping problem on a single multi-die FPGA
- Timing driven partition [8][9][10]
  - Path-based optimization in partition

[8] R. Burra and D. Bhatia, "**Timing driven multi-fpga board partitioning**," in Proceedings Eleventh International Conference on VLSI Design, pp. 234–237, Jan 1998.

[9] S. Ou and M. Pedram, "**Timing-driven partitioning using iterative quadratic programming**," Dept. of EE-Systems, University of Southern California, 2001.

[10] C. Ababei, S. Navaratnasothie, K. Bazargan, and G. Karypis, "**Multi-objective circuit partitioning for cutsize and path-based delay minimization**," in IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 181–185, Nov 2002.

[11] S. Chen, R. Sun, and Y. Chang, "**Simultaneous partitioning and signals grouping for time-division multiplexing in 2.5d fpga-based systems**," in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–7, 2018.

Outline

- Introduction
- Preliminary
- **Problem Formulation**
- Methodology
- Experiment results
- Conclusion

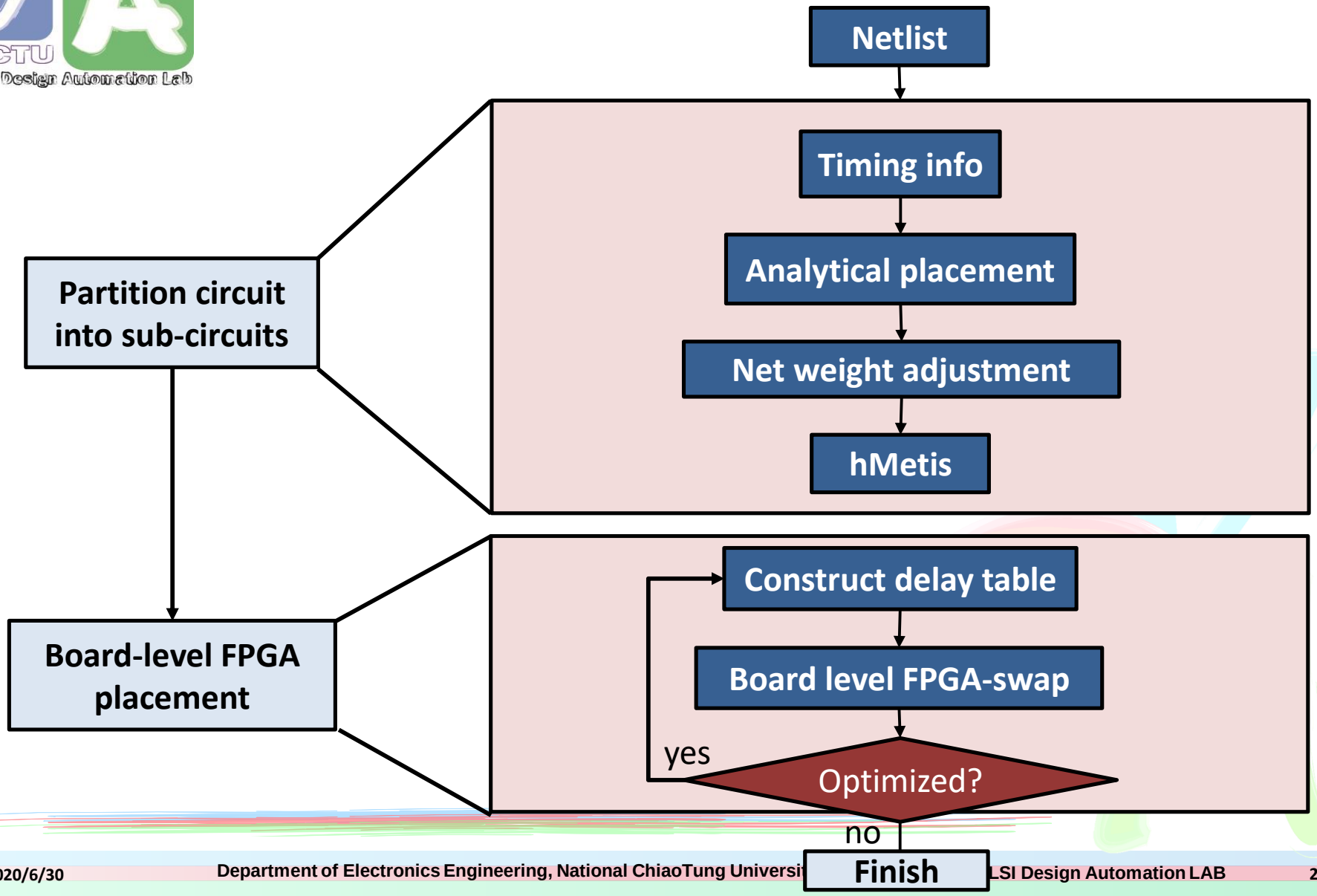
Problem Formulation

- Input
  - A hypergraph $H(V, E)$ to represent the circuit netlist
  - A graph $G(V^G, E^G)$ to represent multi-FPGA system
- Objective
  - Place the netlist on to each individual FPGA $v_i^G \in V^G$
  - Minimize $f_d = \max\{d(s)\}$
- Constraint
  - Each partition p maps to a distinct FPGA v_i^G
  - #logics in each FPGA do not exceed its logic capacity

Outline

- Introduction
- Preliminary
- Problem Formulation
- **Methodology**
- Experiment results
- Conclusion

Overall Flow



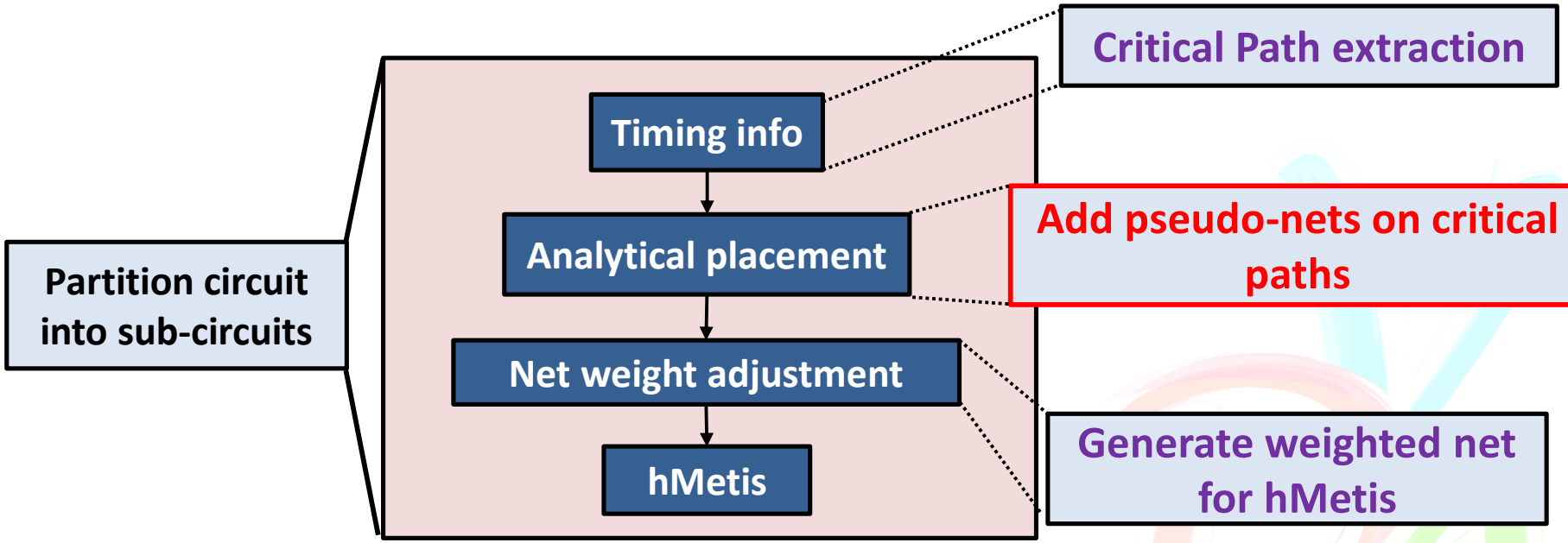
Outline

- Introduction
- Preliminary
- Problem Formulation
- **Methodology**
  - **Net-weight adjustment for partitioning**
  - TDM optimization through board-level FPGA placement
- Experiment results
- Conclusion

Partition Circuit Into Sub-Circuits

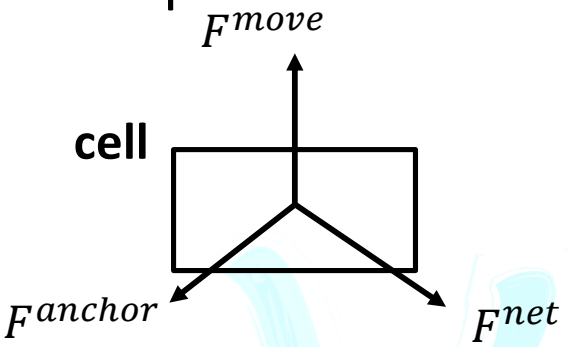
- **Objective:**
  - **How to guide partition to minimize long hop path?**

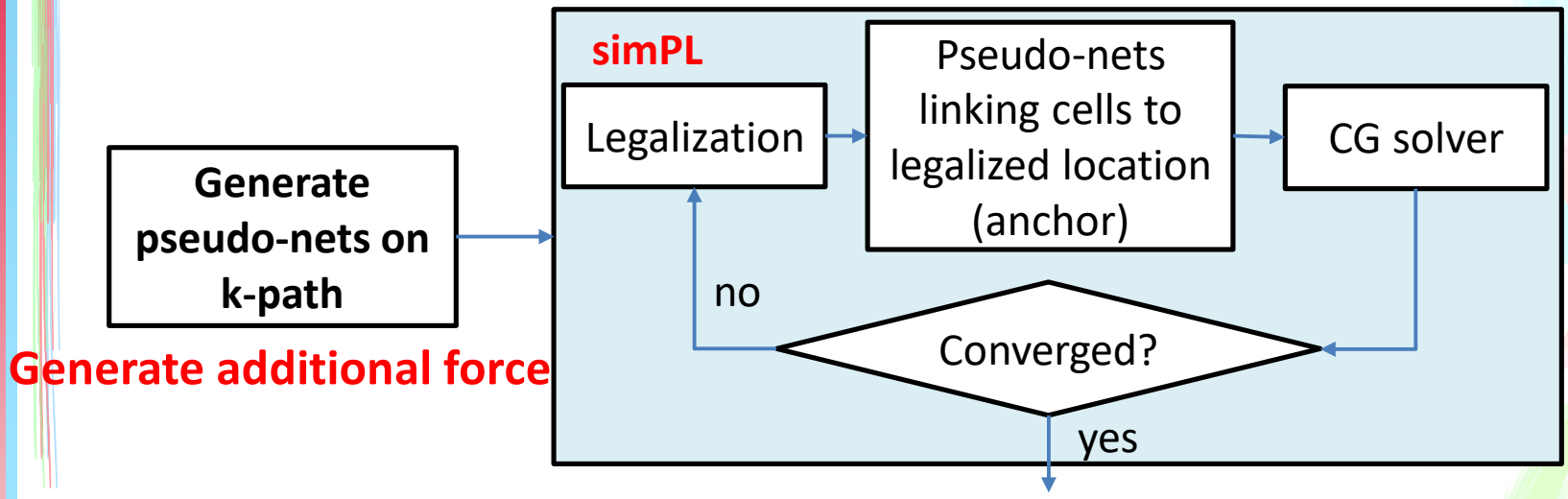
Flow of First Stage



Analytical Placement Penalized on k-path

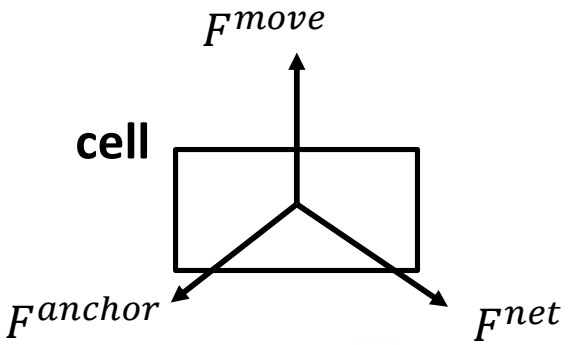
- Benefit of analytical placement over random seed partition
  - Force-directed
  - Offer a global view on the netlist
  - Deterministic

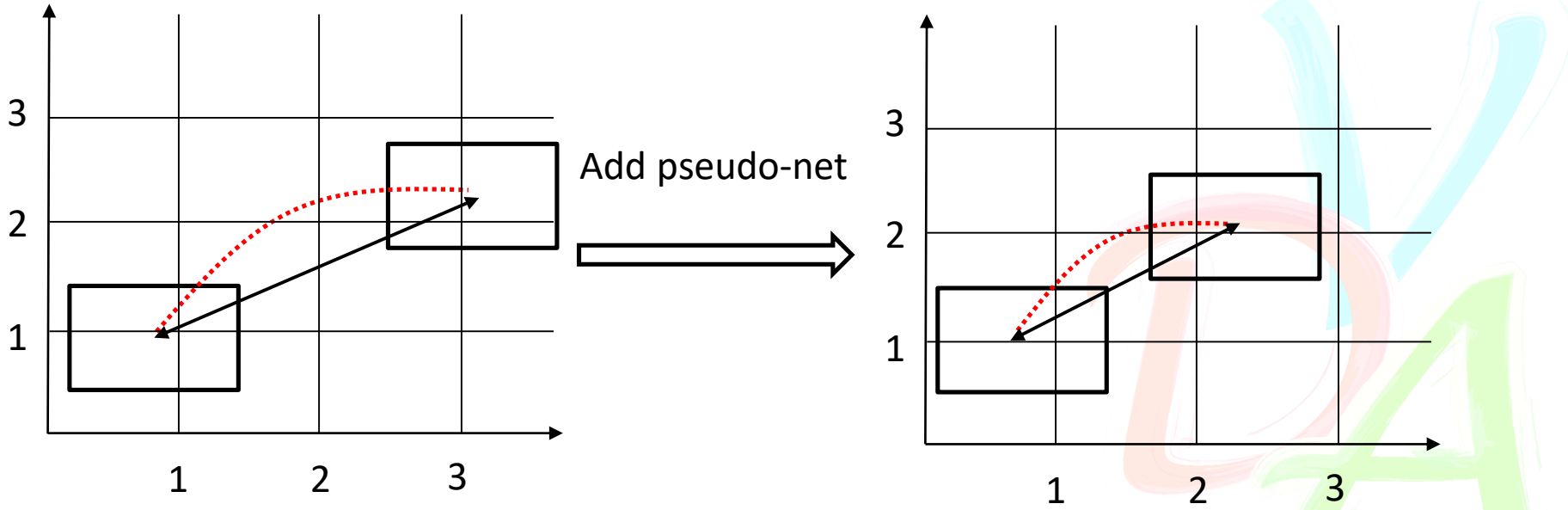




Analytical Placement Penalized on k-path

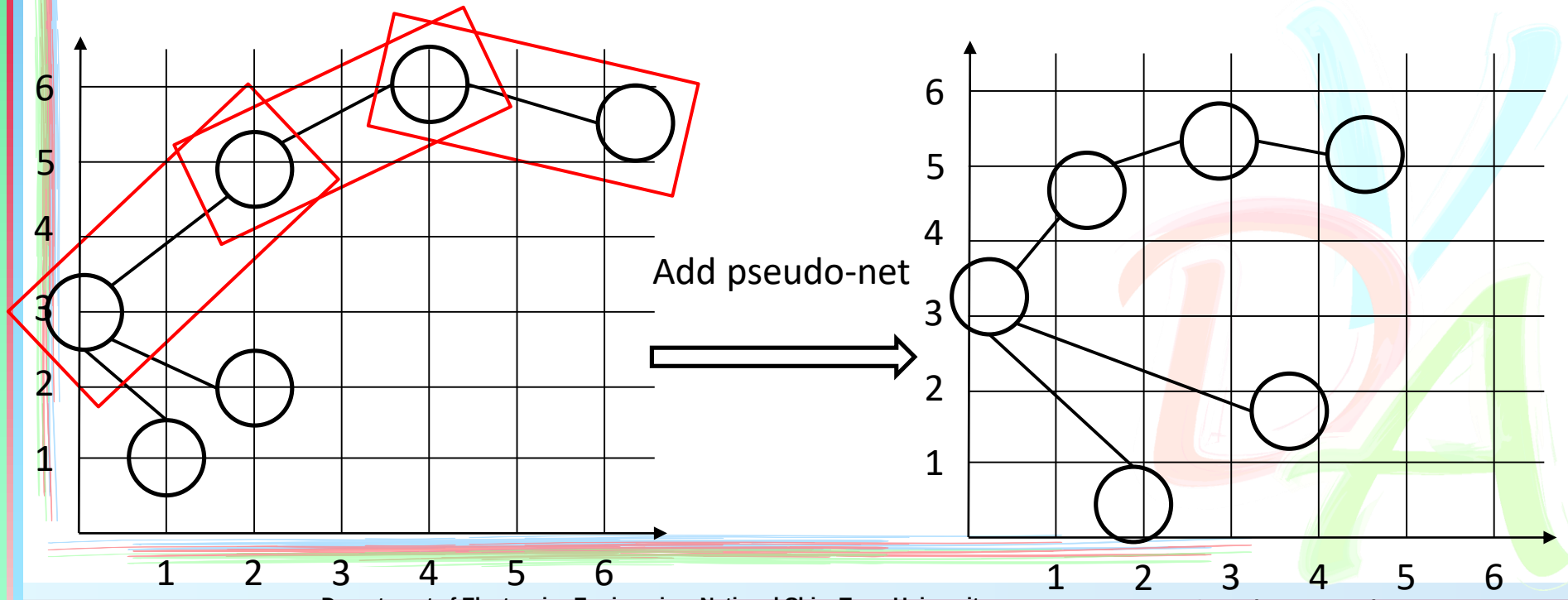
- Force-directed
- Generate pseudo-nets for cells
  - Additional net force to drag cells closer



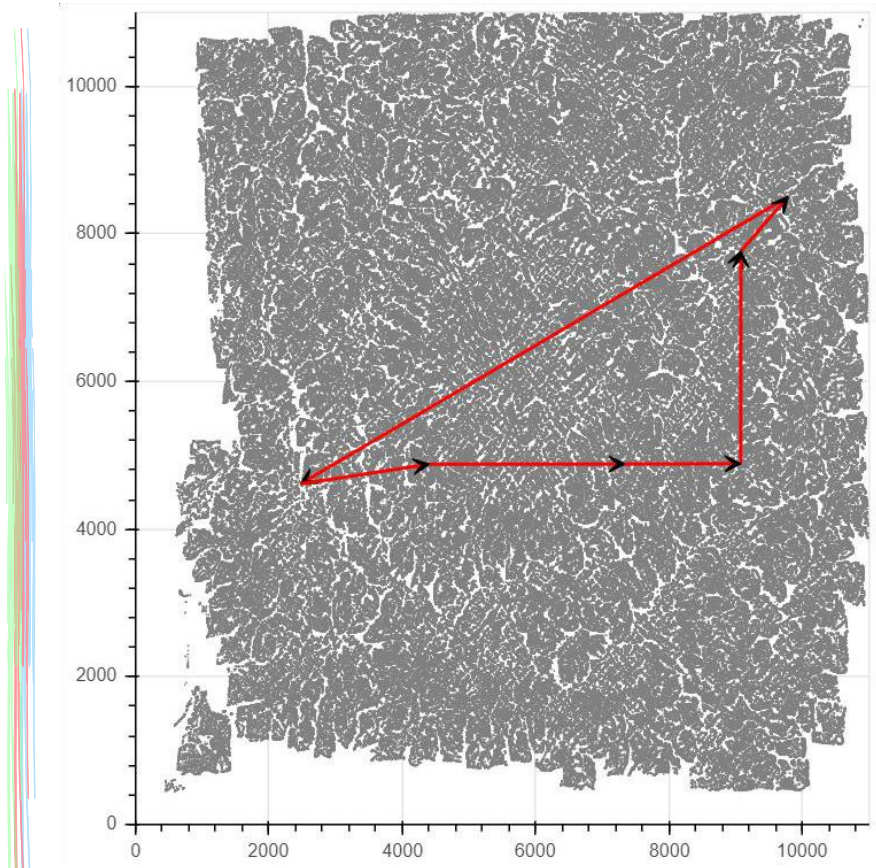


Analytical Placement Penalized on k-path

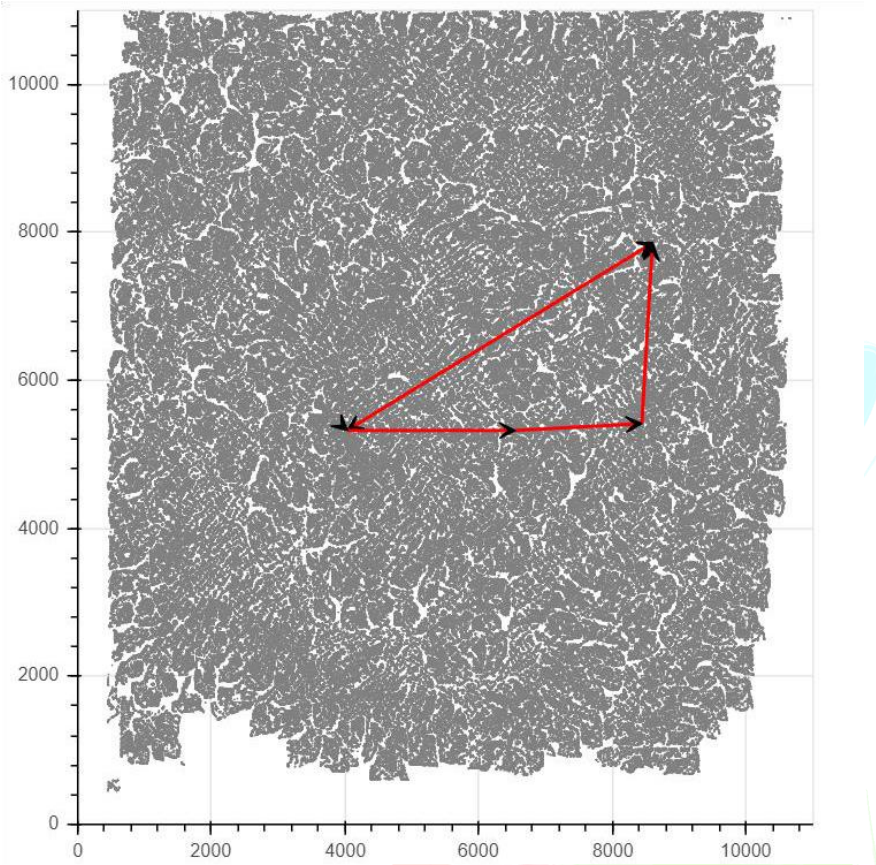
- Additional penalty on critical paths
  - Generate pseudo nets on critical paths
    - Additional net forces to drag cells closer
  - Integrate timing information into placement



Analytical Placement Penalized on k-path

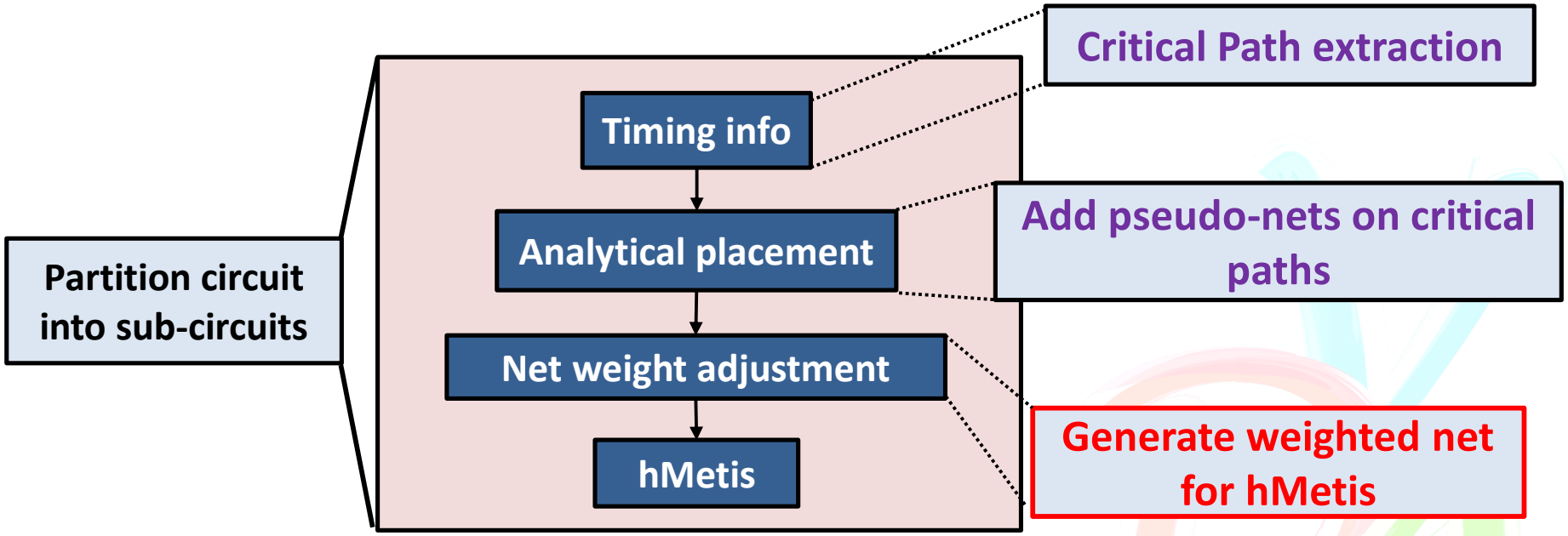


Before adding pseudo-net



After adding pseudo-net

Flow of First Stage



Net Weight Adjustment

- Use placement results to adjust the net-weight
  - Location of each vertex is given
  - Net weight is determined based on the bounding box

- Use **Bound2Bound net model (B2B)** to model connectivity
  - Net weight has an inverse relation with net bounding box

$$w(e) = \frac{1}{\Gamma_x^{HPWL} + \Gamma_y^{HPWL} + 1}$$

Net Weight Adjustment

- Normalization formula
  - Given a upper bound θ_u, weight w_e is normalized between 1 to θ_u

$$w'(e) = \frac{(w(e) - w_{min})(\theta_u - 1)}{w_{max} - w_{min}} + 1$$

    - w_{max} : Maximum weight before normalization
    - w_{min} : Minimum weight before normalization

Net Weight Adjustment

- Normalization formula
  - Given a upper bound θ_u, weight w_e is normalized between 1 to θ_u

$$w'(e) = \frac{(w(e) - w_{min})(\theta_u - 1)}{w_{max} - w_{min}} + 1$$

  - Example
    - $w(e_1) = 0.1$, $w(e_2) = 0.5$, $w(e_3) = 0.7$, $w(e_4) = 0.9$ and $\theta_u = 5$

$$w'(e_1) = \frac{(0.1 - 0.1)(5 - 1)}{0.9 - 0.1} + 1 = 1$$

$$w'(e_2) = \frac{(0.5 - 0.1)(5 - 1)}{0.9 - 0.1} + 1 = 3$$

$$w'(e_3) = \frac{(0.7 - 0.1)(5 - 1)}{0.9 - 0.1} + 1 = 4$$

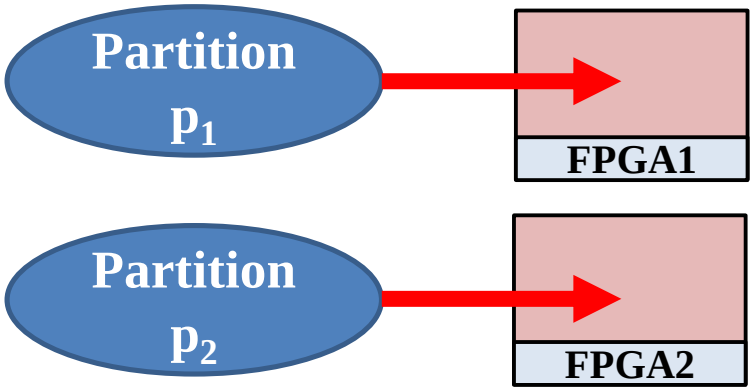
$$w'(e_4) = \frac{(0.9 - 0.1)(5 - 1)}{0.9 - 0.1} + 1 = 5$$

Outline

- Introduction
- Preliminary
- Problem Formulation
- **Methodology**
  - Net-weight adjustment for partitioning
  - **TDM optimization through board-level FPGA placement**
- Experiment results
- Conclusion

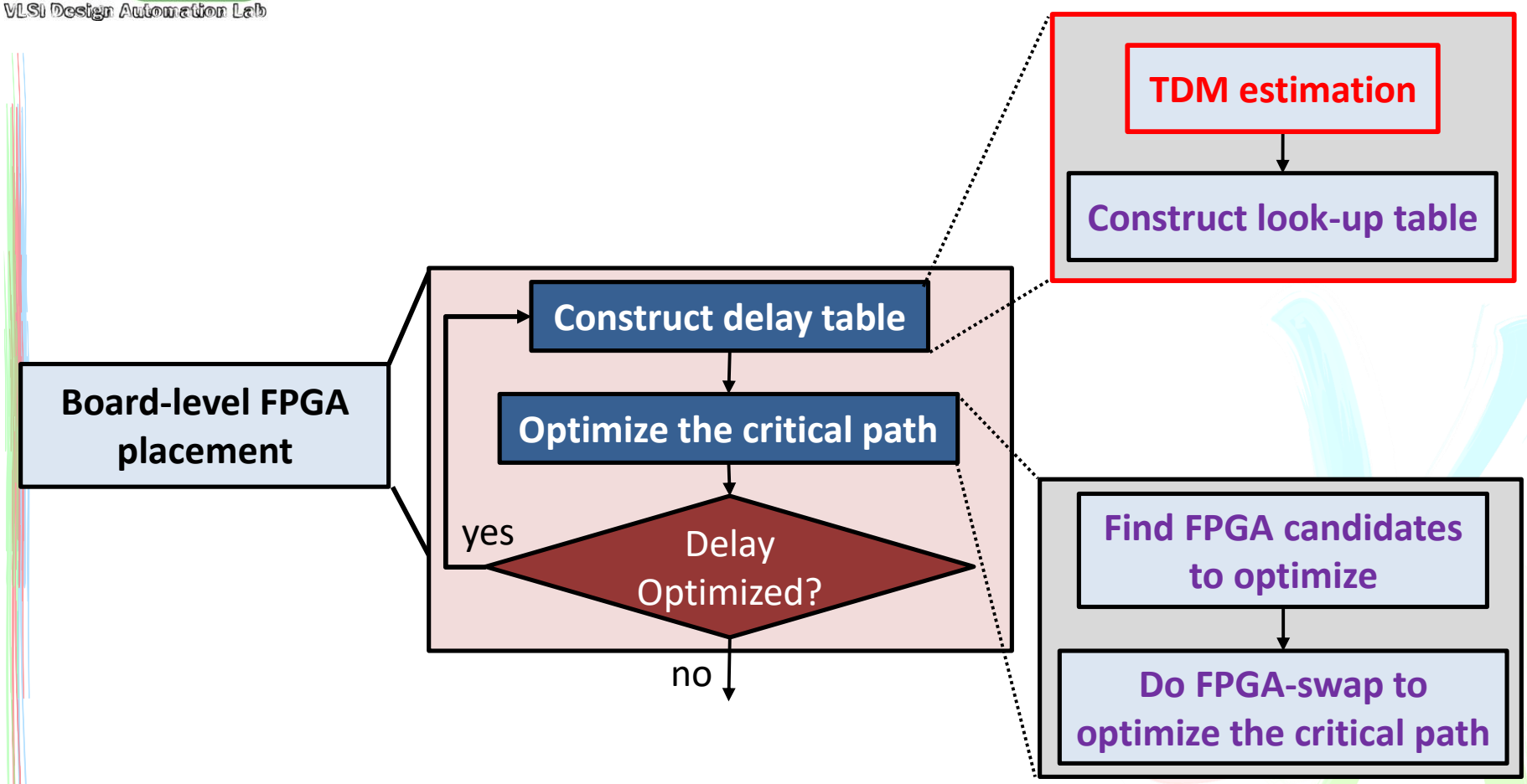
Board-Level FPGA Placement

- **Objective**
  - Put each partition group (sub-circuits) to a distinct physical FPGA

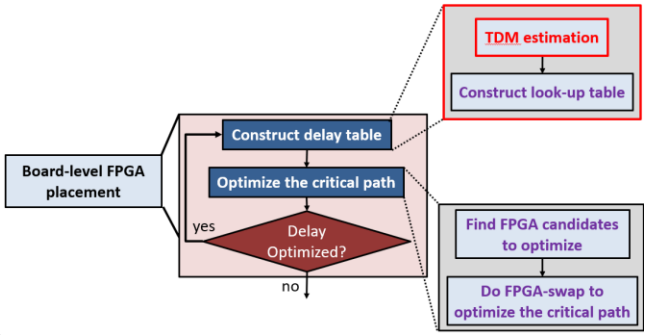


  - **How to optimize TDM delay through board-level FPGA placement?**

Flow of Second Stage



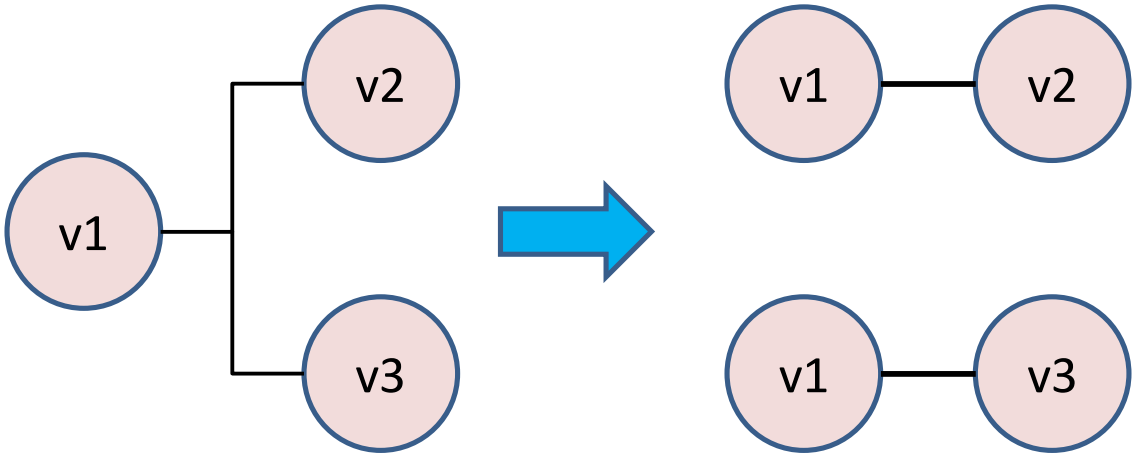
TDM Estimation



- A router to predict wire demand $f^G(u, v)$
- For each net e_i
  - Decompose net e_i into 2-pin nets
  - Route 2-pin nets using shortest path algorithm on multi-FPGA system
  - Remove redundant routed edges
  - Add weight to FPGA edges

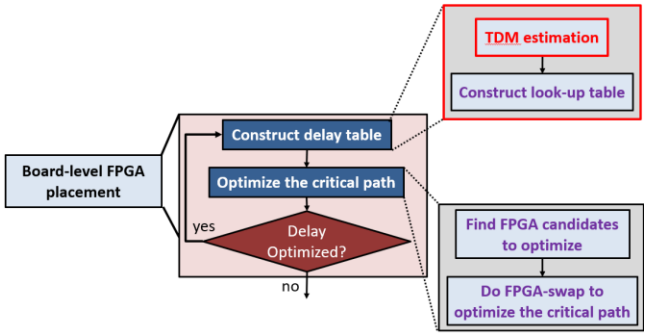
TDM Estimation

- A router to predict wire demand $f^G(u, v)$
- For each net e_i
  - Decompose e_i into 2-pin nets

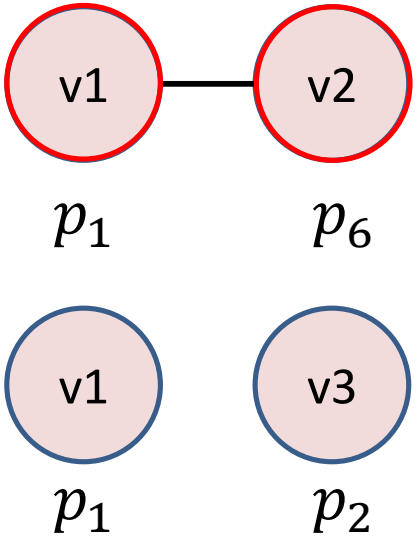


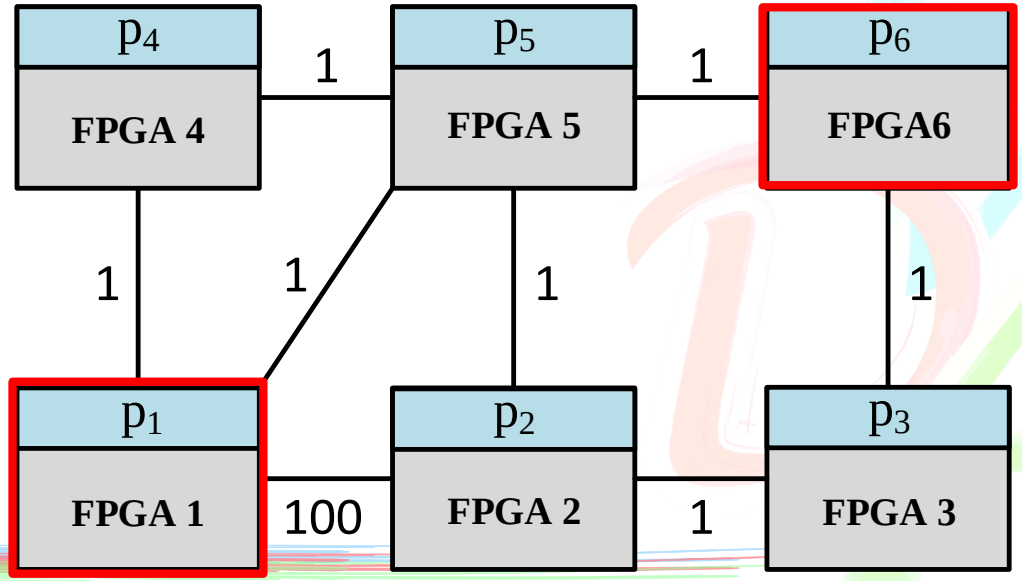
- A router to predict wire demand $f^G(u, v)$
- For each net e_i
  - Decompose e_i into 2-pin nets

TDM Estimation

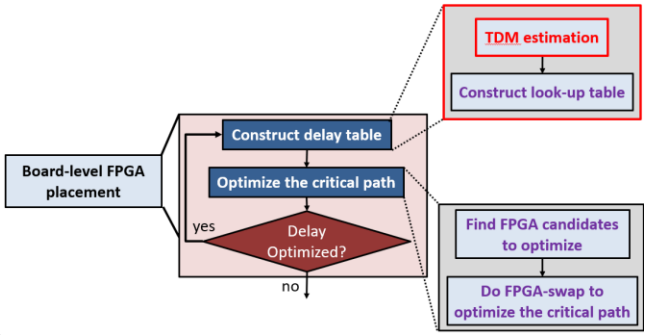


- A router to predict wire demand $f^G(u, v)$
  - Edge weight is wire demand
- For each net e_i
  - Find the corresponding FPGAs by cells' partitions

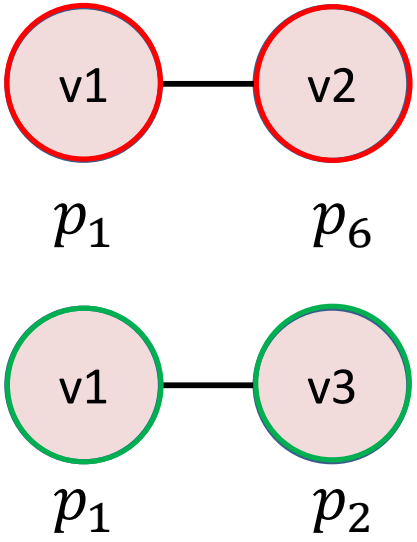


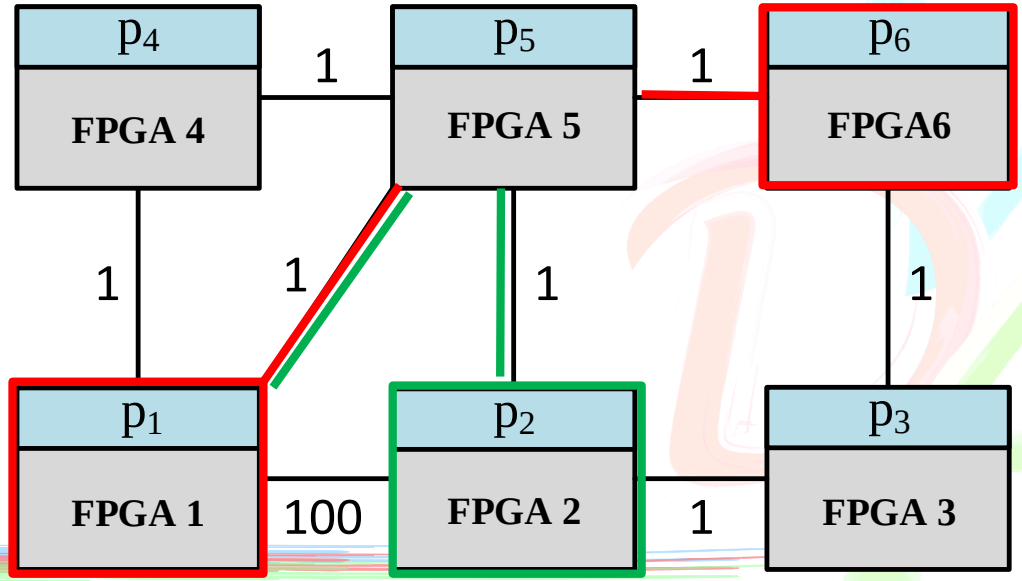


TDM Estimation

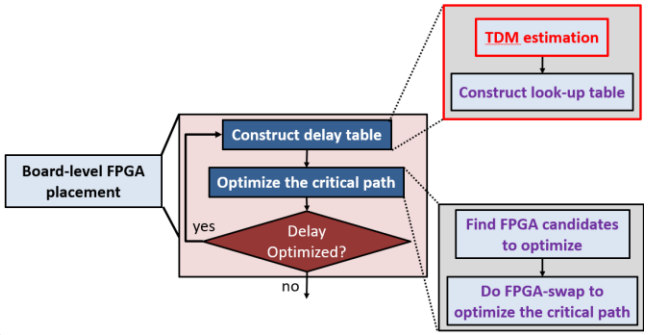


- A router to predict wire demand $f^G(u, v)$
  - Edge weight is wire demand
- For each net e_i
  - Route 2-pin nets using shortest path algorithm

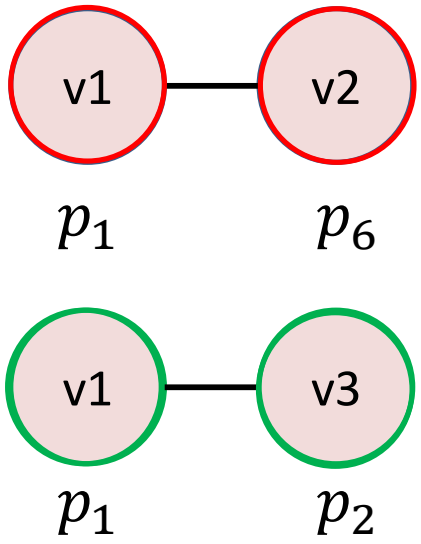


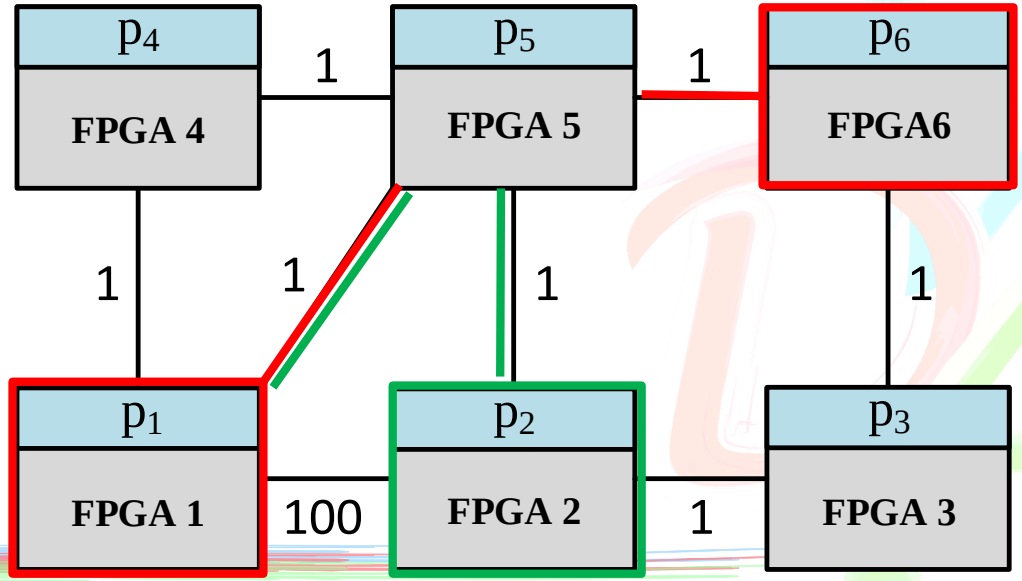


TDM Estimation

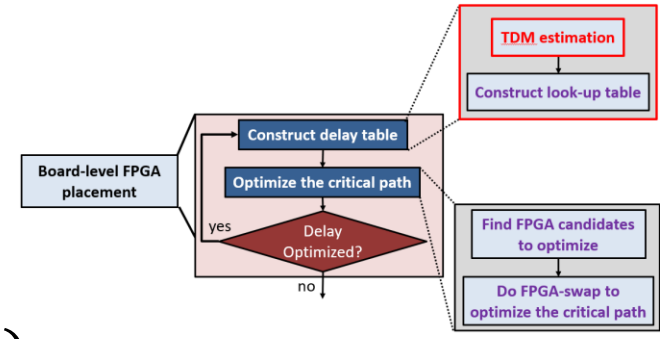


- A router to predict wire demand $f^G(u, v)$
  - Edge weight is wire demand
- For each net e_i
  - Remove redundant routed edges

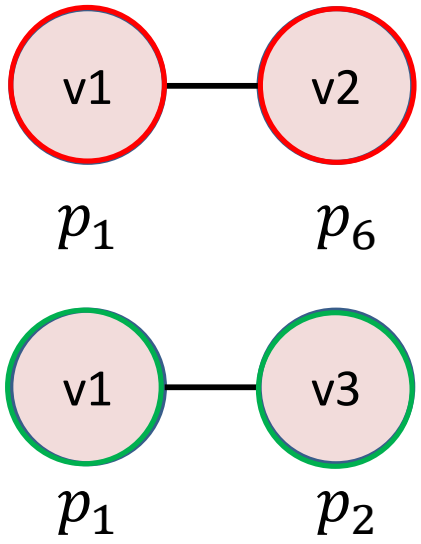


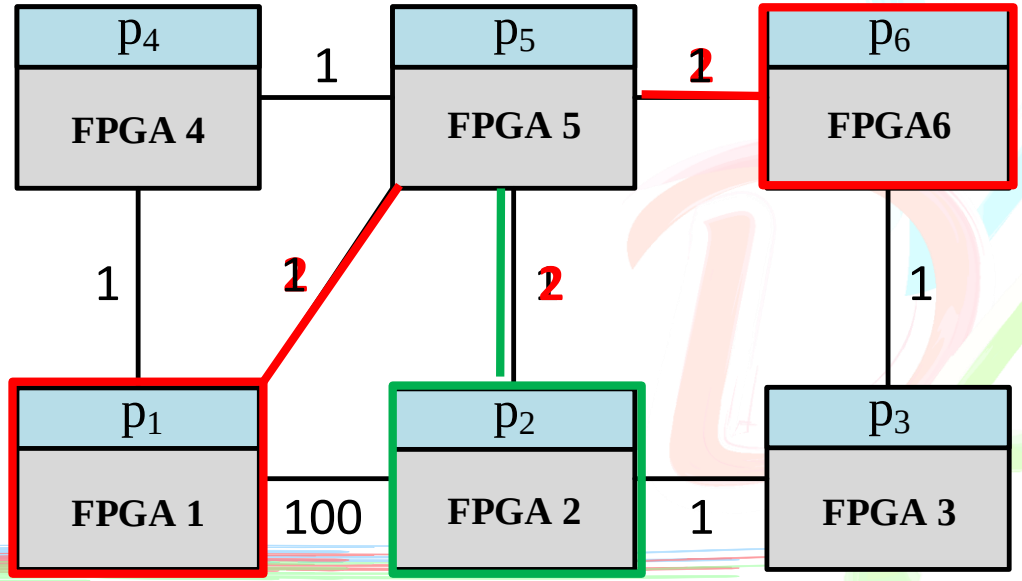


TDM Estimation



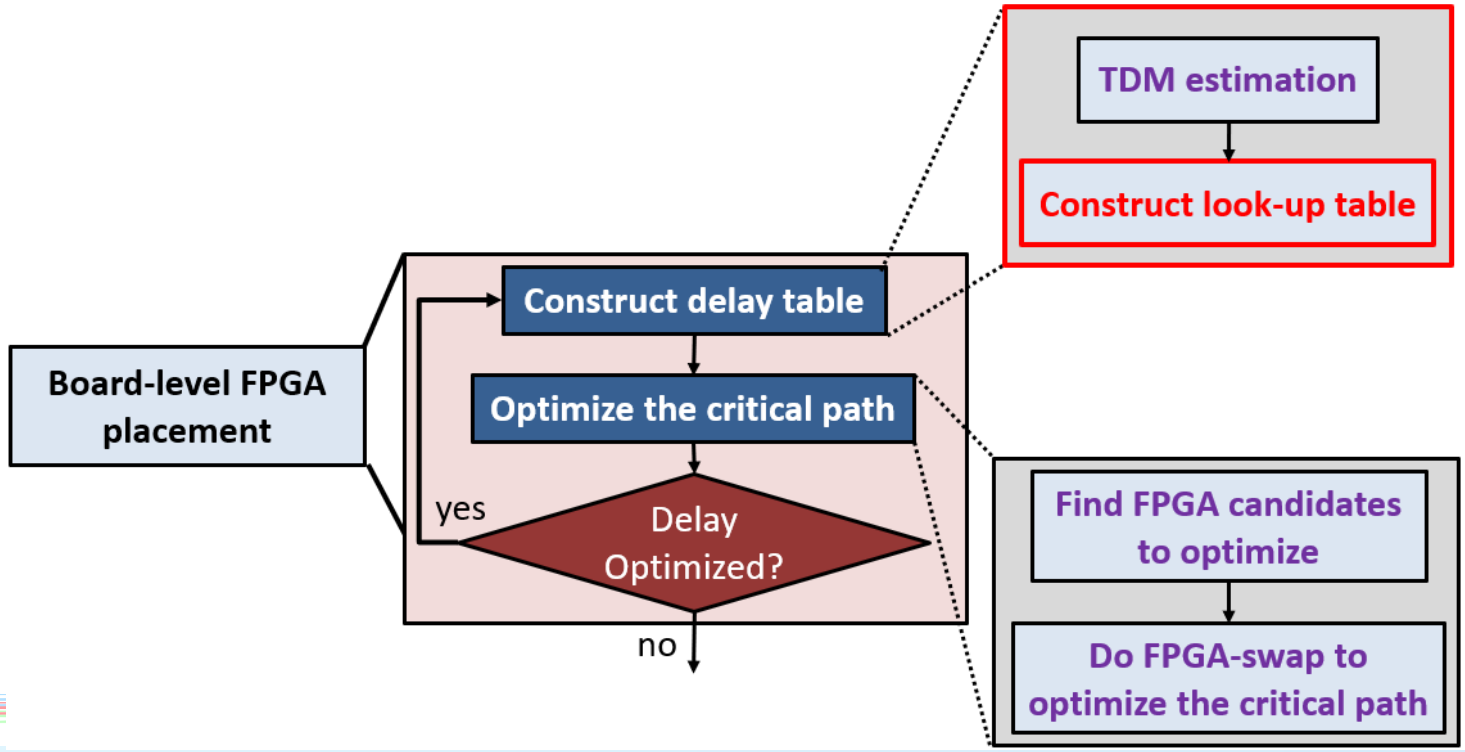
- A router to predict wire demand $f^G(u, v)$
  - Edge weight is wire demand
- For each net e_i
  - Weight of routed edge increases by one





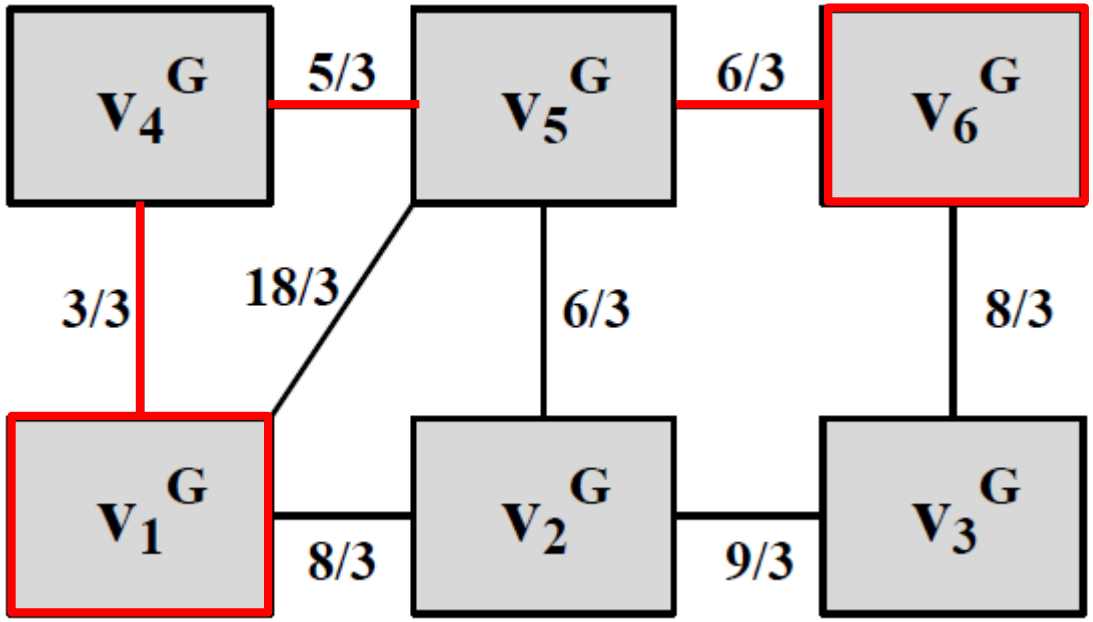
Look-Up Table Construction

- Construct table for delay between each two FPGAs v_i^G and v_j^G
  - Store minimum delay from one FPGA to another
  - Based on shortest path algorithm



Look-Up Table Construction

- A look-up table for delay between FPGA boards
  - Delay between v_1^G and v_6^G is $(\left\lceil \frac{3}{3} \right\rceil + 1) + \left\lceil \frac{5}{3} \right\rceil + \left\lceil \frac{6}{3} \right\rceil = 6$



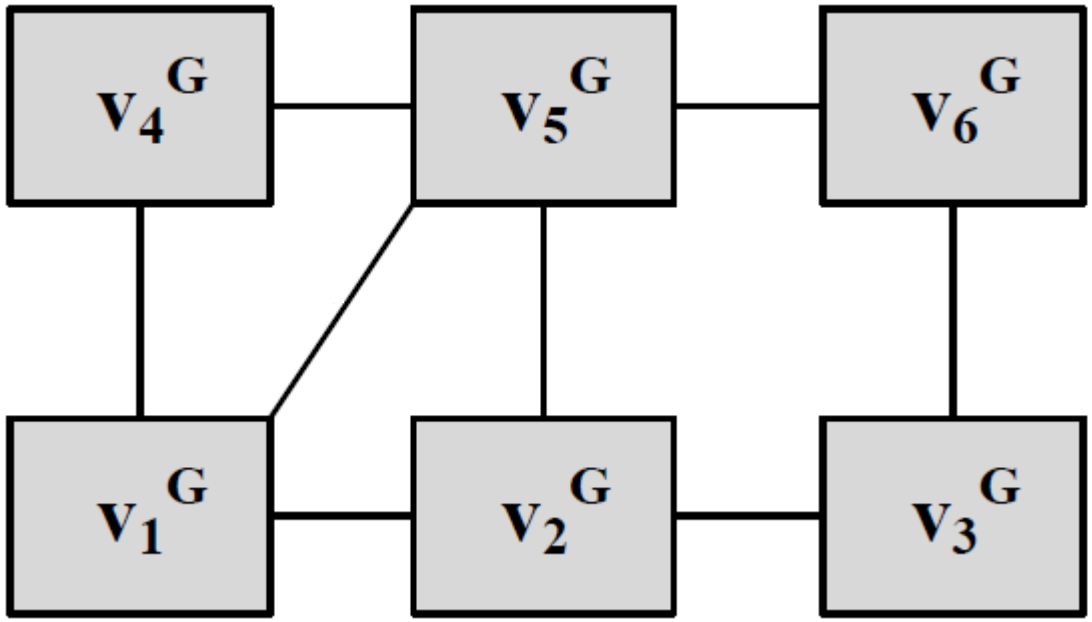
Wire demand/wire capacity

Look-up table	v_1^G	v_2^G	v_3^G	v_4^G	v_5^G	v_6^G
v_1^G	0	4	8	2	4	6
v_2^G	4	0	4	4	2	4
v_3^G	8	4	0	8	6	4
v_4^G	2	4	8	0	2	4
v_5^G	4	2	6	2	0	2
v_6^G	6	4	4	4	2	0

Look-Up Table Construction

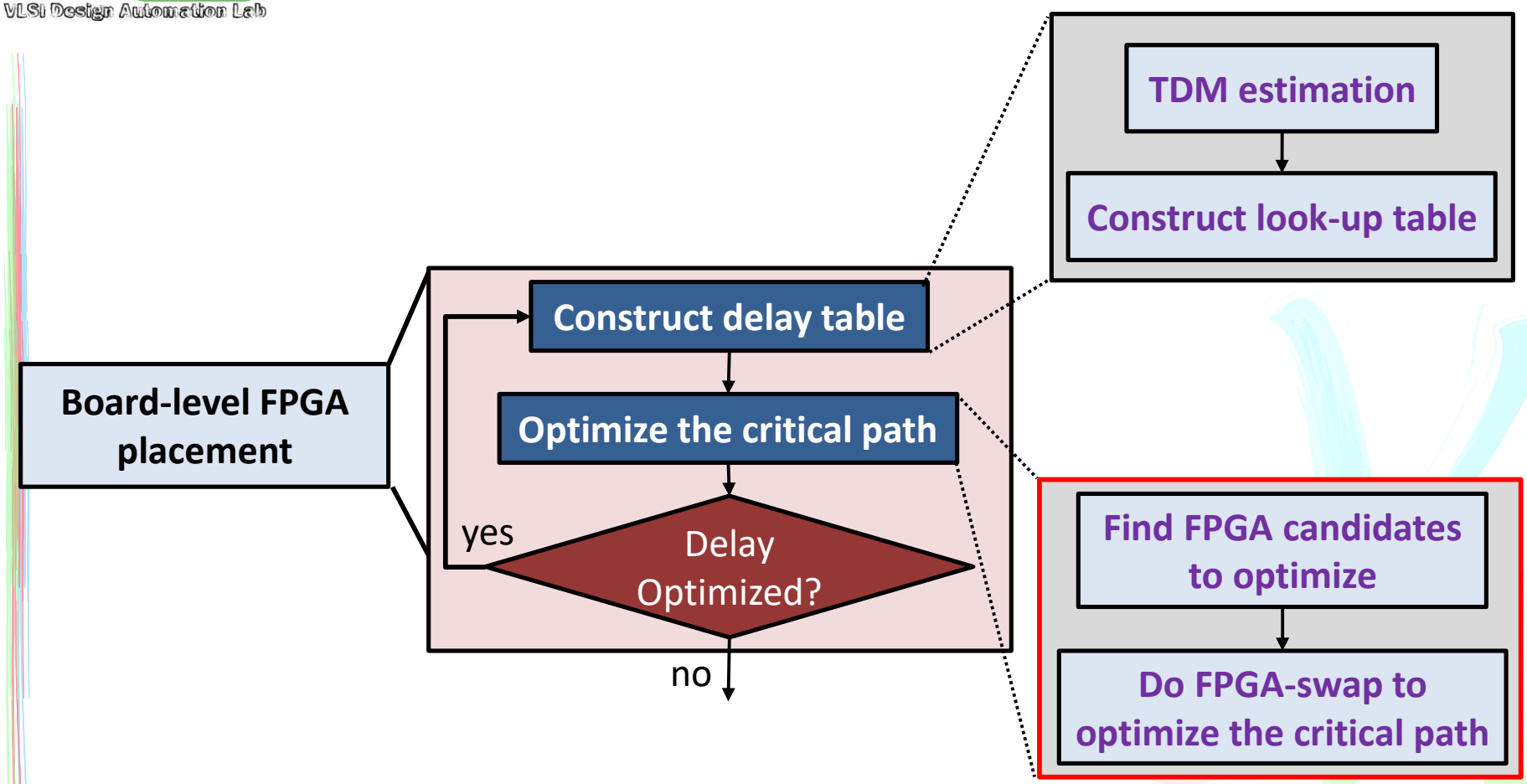
- Path delay calculation
  - Path **s**: $v_1^G \rightarrow v_4^G \rightarrow v_2^G$

$$d(s) = 2 + 4 = 6$$



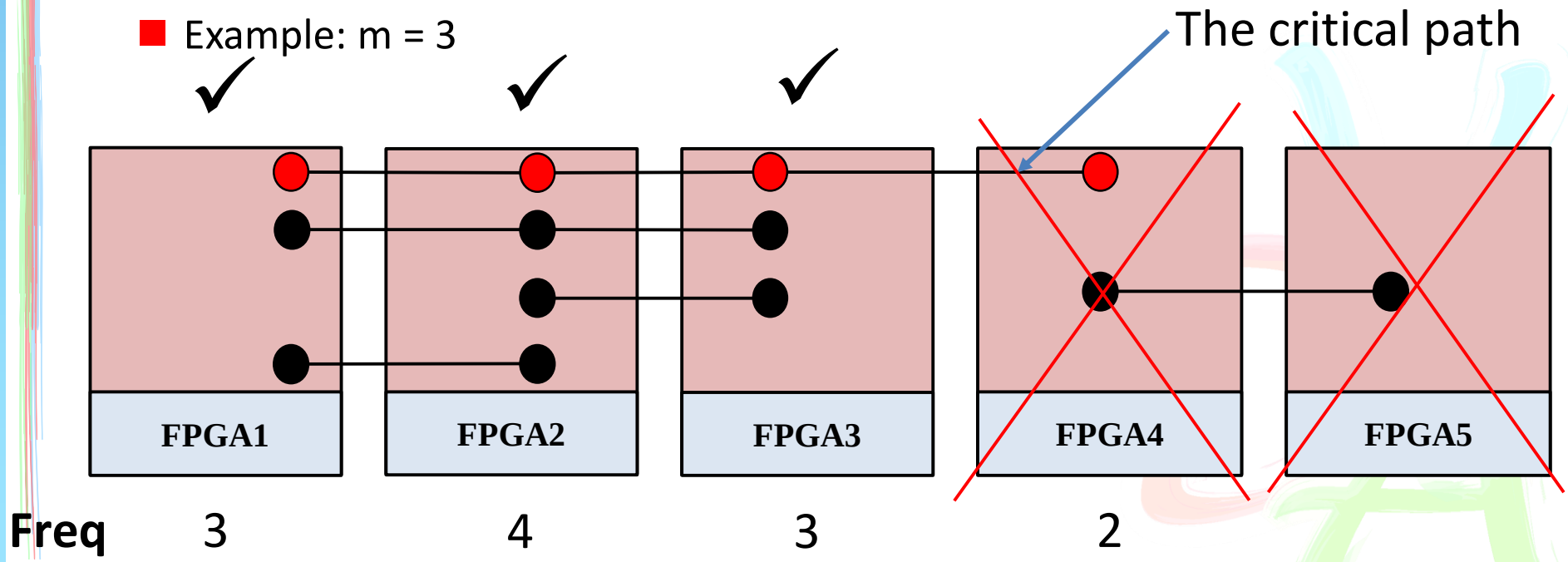
Look-up table						
	v_1^G	v_2^G	v_3^G	v_4^G	v_5^G	v_6^G
v_1^G	0	4	8	2	4	6
v_2^G	4	0	4	4	2	4
v_3^G	8	4	0	8	6	4
v_4^G	2	4	8	0	2	4
v_5^G	4	2	6	2	0	2
v_6^G	6	4	4	4	2	0

Flow of Second Stage



Board-level FPGA swapping

- Find high potential candidates to optimize
  - On the critical path
  - Top m frequency among k-paths
  - Example: m = 3

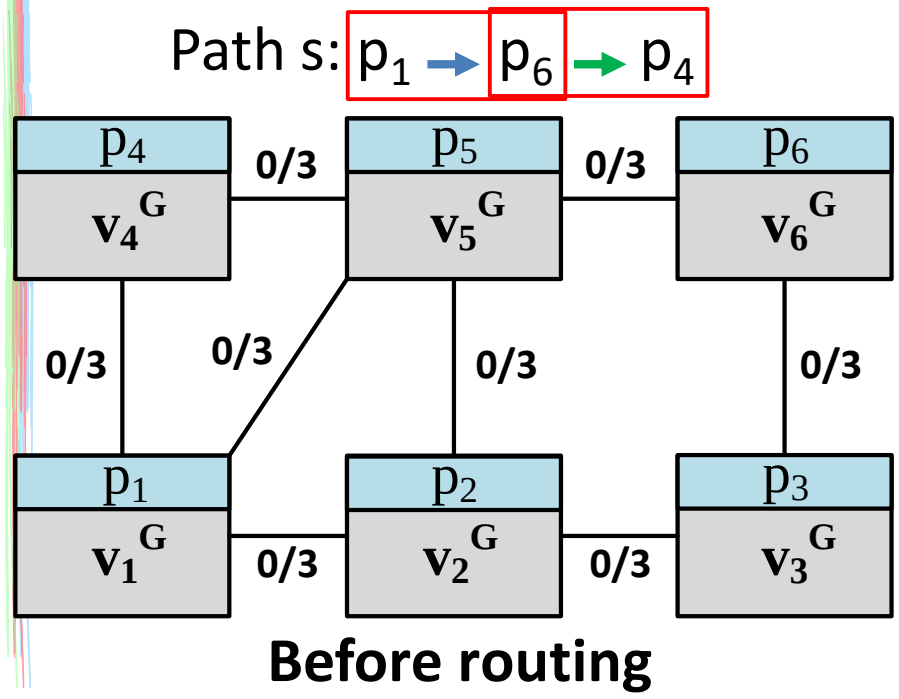


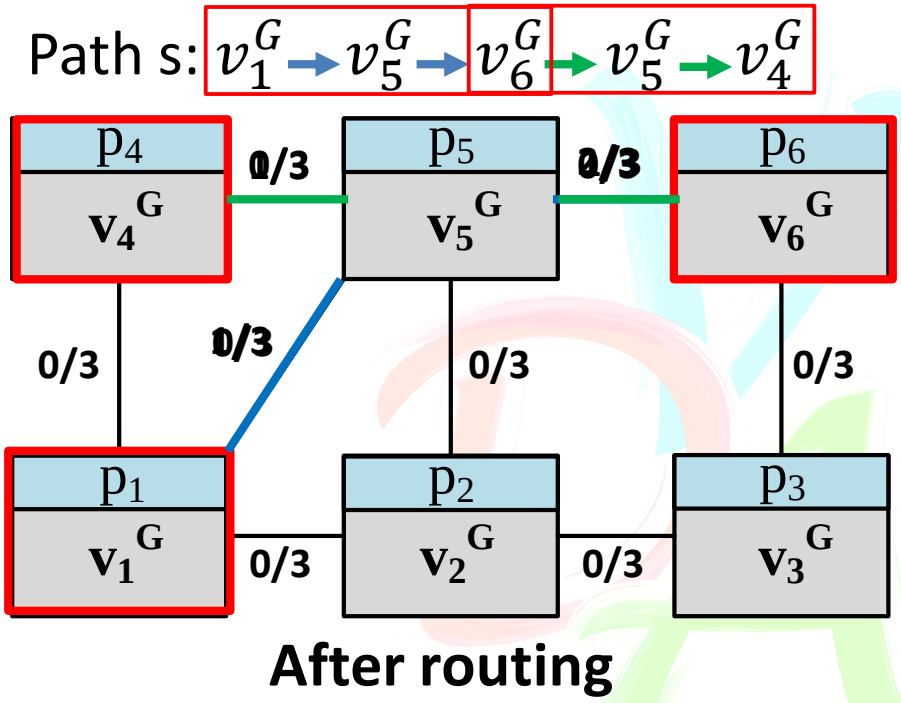
Board-level FPGA swapping

$v_1^G \rightarrow v_5^G$ $v_5^G \rightarrow v_6^G$ $v_6^G \rightarrow v_5^G$ $v_5^G \rightarrow v_4^G$

d(s) =($\left\lceil \frac{1}{3} \right\rceil + 1$ **+** $\left(\left\lceil \frac{2}{3} \right\rceil + 1\right)$ **+** $\left(\left\lceil \frac{2}{3} \right\rceil + 1\right)$ **+** $\left(\left\lceil \frac{1}{3} \right\rceil + 1\right)$ **= 8**

- FPGA swapping
  - Swap the corresponding FPGAs between partitions



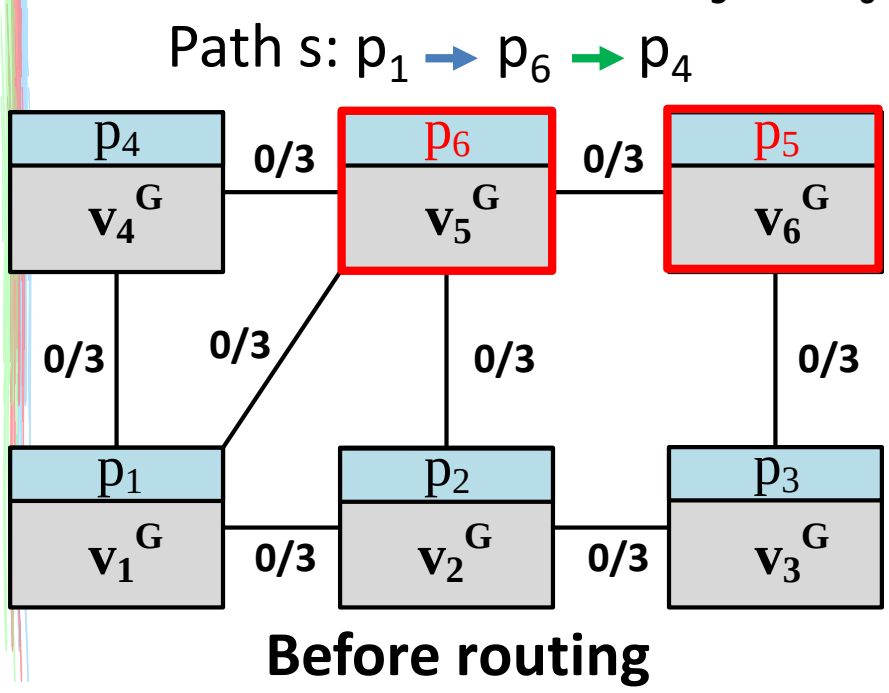


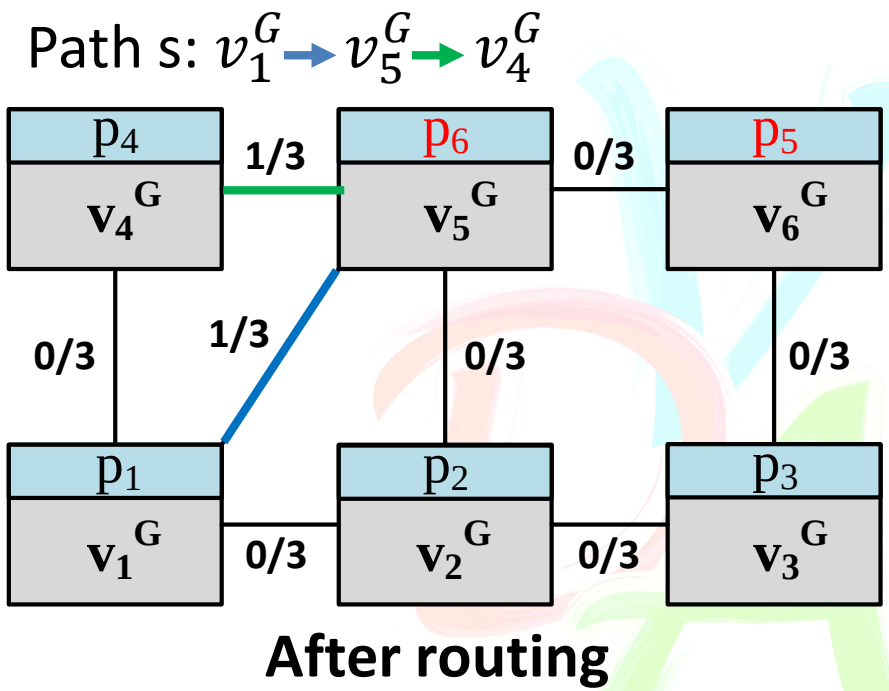
Board-level FPGA swapping

$v_1^G \rightarrow v_5^G \quad v_5^G \rightarrow v_4^G$

d(s) = ($\left\lceil \frac{1}{3} \right\rceil + 1$**) +** $\left\lceil \frac{1}{3} \right\rceil + 1$**) = 4**

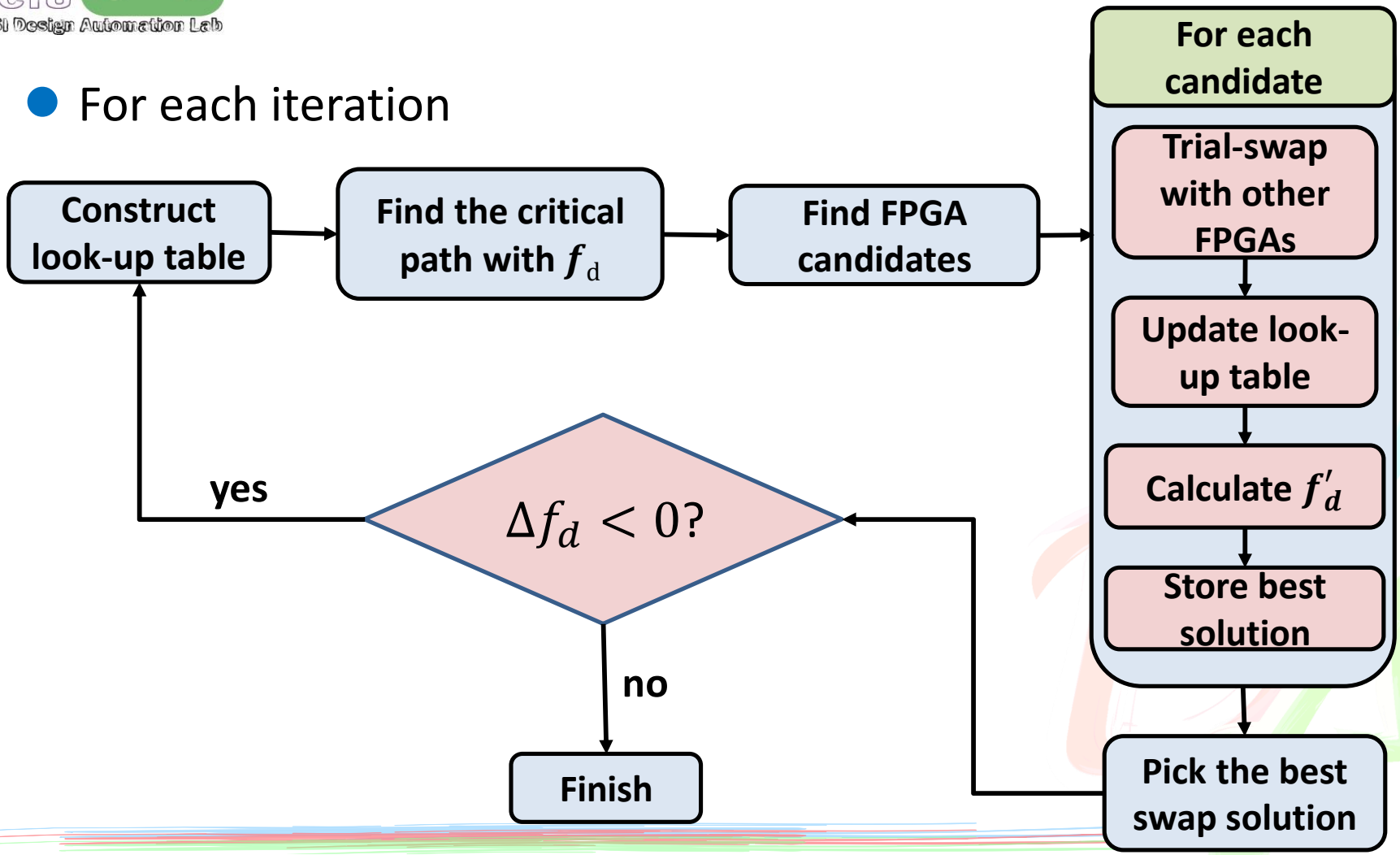
- FPGA swapping
  - Swap the corresponding FPGAs between partitions
  - Swap FPGAs between $\mathbf{p_5}$ and $\mathbf{p_6}$





TDM Optimization Flow

- For each iteration



Summary

- **Objective:**
  - **How to guide partition to minimize long hop path?**
    - Integrate timing information into analytical placement
    - Guide partitioning using analytical placement results
  - **How to optimize TDM delay through board-level FPGA placement?**
    - Estimate TDM delay with a router
    - Construct a delay look-up table
    - Optimize TDM delay by FPGA swap iteratively

Outline

- Introduction
- Preliminary
- Problem Formulation
- Methodology
- **Experiment results**
- Conclusion

Experimental Results

- Platform
  - Intel® Xeon® CPU 2.2 GHz 64-bit Linux workstation with 10 processors
  - C++ language with C++14 compliant compiler
  - 98 GB memory
- Benchmark
  - Titan23 [13]
- Multi-FPGA model
  - “synopsys01” from ICCAD-2019 contest problem B [12]
  - 43 FPGAs
  - 214 connections
  - Capacity for each connection between FPGAs is set as 50

Experimental Results

- Titan23 benchmark

Testcase	#vertices	#edges	#paths
sparcT1_core	91975	92826	49262
Neuron	92289	125304	68039
stereo_vision	94046	126889	55656
Des90	111220	139556	47292
SLAM_spheric	113114	142407	20339
cholesky_mc	113249	144947	84127
Segmentation	138294	179050	13810
bitonic_mesh	192063	235327	82707
Dart	202353	223300	98562
openCV	217452	284107	109285
stap_qrd	240239	290122	177616
Minres	261350	320529	148496
cholesky_bdti	266421	342687	189601
Denoise	275637	256847	22804
sparcT2_core	300108	302662	126539
gsm_switch	493252	507809	331021
mes_noc	547494	577584	274803
LU230	574370	669475	365577
LU_Network	647786	726678	459027
sparcT1_chip2	822254	820217	443159
Directrf	931273	1374740	488086
bitcoin_miner	1089274	1448140	644733

Experimental Results

- Compare with hMetis
  - **Default weighting**
  - **Direct-mapping**
    - Directly assign each partition to a FPGA based on initial constructive order
- **Ours**
  - **Placement guided weighting**
  - **TDM optimization**

Experimental Results

	Default weighting/ direct-mapping				Placement-guided weighting/ TDM optimization (ours)			
Testcase	Cut	NFH	f^d	Time(s)	Cut	NFH	f^d	Time(s)
sparcT1_core	1.02	0.95	1.06	21.9	1.00	1.00	1.00	156.5
neuron	0.99	1.00	1.00	10.5	1.00	1.00	1.00	43.7
stereo_vision	0.90	1.33	1.33	10.7	1.00	1.00	1.00	48.5
des90	1.01	1.50	1.13	22.7	1.00	1.00	1.00	68.9
SLAM_spheric	0.96	1.40	2.00	22.3	1.00	1.00	1.00	117.2
cholesky_mc	1.05	1.60	1.60	16.5	1.00	1.00	1.00	91.4
segmentation	0.88	26.13	26.13	23.4	1.00	1.00	1.00	167.4
bitonic_mesh	0.99	0.67	1.00	39.2	1.00	1.00	1.00	115.4
dart	1.08	1.09	1.23	36.3	1.00	1.00	1.00	201.8
openCV	1.02	1.00	1.20	59.5	1.00	1.00	1.00	220.5
stap_qrd	0.99	1.22	1.15	20.1	1.00	1.00	1.00	125.9
minres	1.02	1.67	1.67	33.3	1.00	1.00	1.00	128.1
cholesky_bdti	1.07	0.83	1.25	40.5	1.00	1.00	1.00	168.8
denoise	0.93	10.45	10.45	54.2	1.00	1.00	1.00	285.4
sparcT2_core	1.01	1.20	1.29	65.0	1.00	1.00	1.00	342.1
gsm_switch	1.04	2.00	1.89	89.3	1.00	1.00	1.00	525.5
mes_noc	0.93	1.00	1.33	97.6	1.00	1.00	1.00	278.9
LU230	0.96	1.60	1.19	124.7	1.00	1.00	1.00	445.3
LU_Network	1.00	1.00	1.50	110.1	1.00	1.00	1.00	379.8
sparcT1_chip2	0.97	1.25	2.14	159.3	1.00	1.00	1.00	692.4
directrf	1.05	1.60	1.78	173.6	1.00	1.00	1.00	598.5
bitcoin_miner	1.02	2.00	2.22	144.0	1.00	1.00	1.00	600.1
Geo. Mean	**0.99**	1.58	**1.76**	44.5	**1.00**	1.00	1.00	198.2

Experimental Results

- Compare to hMetis, our proposed method achieves:
  - **Average 43% better** performance (1.76 -> 1.00)
  - Average cut size remains almost the same (0.99 -> 1.00)

Validation of Individual Technique

- Comparison on 3 weighting schemes
  - **Default weighting**
    - Default hMetis
  - **Uniform weighting**
    - Generate net weights directly from k-paths and weight uniformly
  - **Placement-guided weighting**
    - Implemented in our first stage

Weighting Schemes Comparison

	Default weighting	Uniform weighting		Placement-guided weighting	
Testcase	f^d	f^d	**Improve**	f^d	**Improve**
sparcT1_core	104	136	-0.31	132	-0.27
neuron	8	8	0.00	8	0.00
stereo_vision	8	10	-0.25	8	0.00
des90	18	12	0.33	18	0.00
SLAM_spheric	40	34	0.15	32	0.20
cholesky_mc	16	10	0.38	14	0.13
segmentation	836	266	0.68	54	0.94
bitonic_mesh	12	20	-0.67	12	0.00
dart	32	34	-0.06	46	-0.44
openCV	30	38	-0.27	46	-0.53
stap_qrd	12	12	0.00	12	0.00
minres	10	10	0.00	12	-0.20
cholesky_bdti	20	16	0.20	18	0.10
denoise	230	32	0.86	28	0.88
sparcT2_core	62	66	-0.06	70	-0.13
gsm_switch	68	84	-0.24	60	0.12
mes_noc	24	22	0.08	24	0.00
LU230	50	46	0.08	54	-0.08
LU_Network	24	20	0.17	22	0.08
sparcT1_chip2	90	90	0.00	72	0.20
directrf	32	26	0.19	22	0.31
bitcoin_miner	40	38	0.05	52	-0.30
Geo. Mean	34.23	29.30	**0.14**	27.96	**0.18**

Validation of Individual Technique

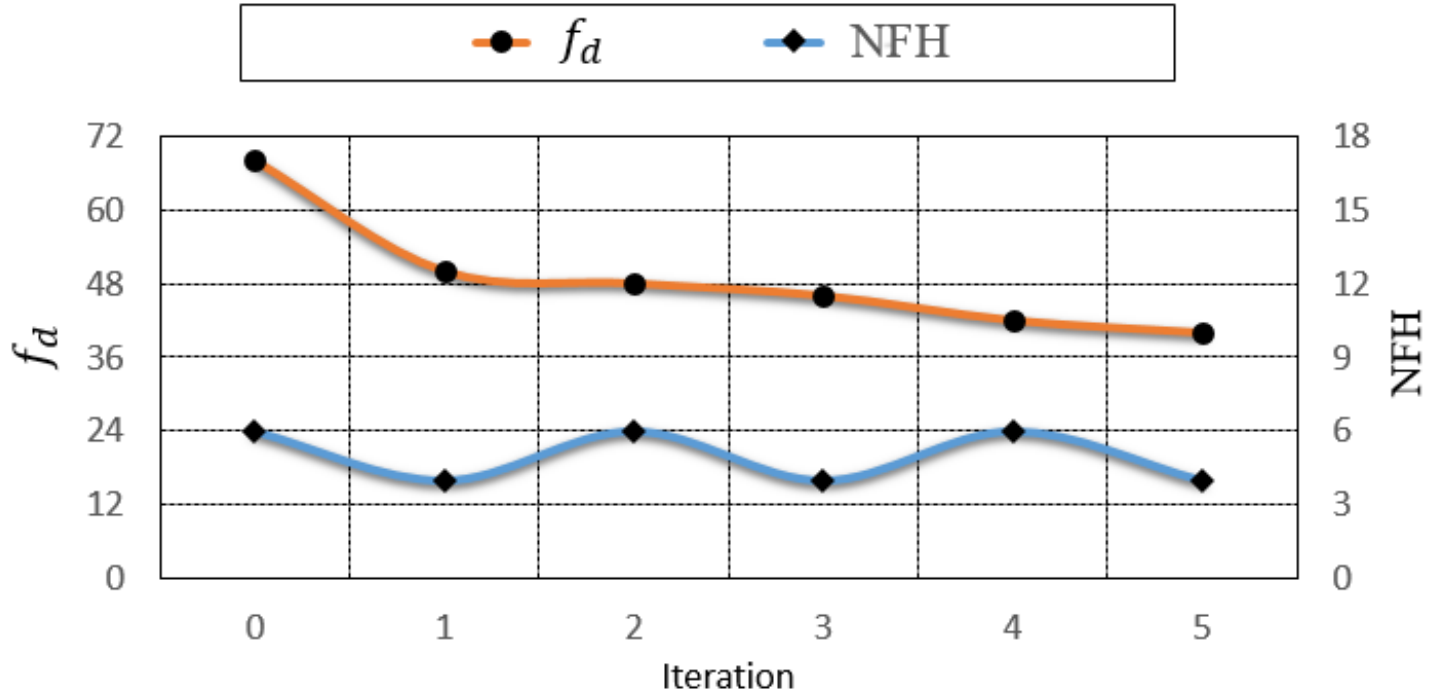
- Comparison on different objectives for board-level FPGA placement
  - **Direct-mapping**
    - Directly assign each partition to a FPGA based on initial order
    - Served as initial solution
  - **Hop optimization**
    - Minimize NFH = maximum #FPGA-hop among critical paths
  - **TDM optimization**
    - Minimize $f_d = \max\{d(s)\}$

Board-Level FPGA Placement

	Direct-mapping	Hop optimization		TDM optimization	
Testcase	f^d	f^d	**Improve**	f^d	**Improve**
sparcT1_core	104	72	0.31	52	0.50
neuron	8	8	0.00	8	0.00
stereo_vision	8	8	0.00	8	0.00
des90	18	16	0.11	16	0.11
SLAM_spheric	40	30	0.25	26	0.35
cholesky_mc	16	18	-0.13	12	0.25
segmentation	836	428	0.49	428	0.49
bitonic_mesh	12	12	0.00	12	0.00
dart	32	30	0.06	24	0.25
openCV	30	30	0.00	22	0.27
stap_qrd	12	12	0.00	12	0.00
minres	10	8	0.20	8	0.20
cholesky_bdti	20	20	0.00	18	0.10
denoise	230	130	0.43	124	0.46
sparcT2_core	62	54	0.13	38	0.39
gsm_switch	68	68	0.00	40	0.41
mes_noc	24	18	0.25	18	0.25
LU230	50	66	-0.32	32	0.36
LU_Network	24	18	0.25	16	0.33
sparcT1_chip2	90	90	0.00	70	0.22
directrf	32	22	0.31	18	0.44
bitcoin_miner	40	20	0.50	20	0.50
Geo. Mean	34.23	28.90	**0.15**	24.43	**0.29**

FPGA-hop versus TDM

- The number of FPGA-hops increases when optimizing f_d
  - Optimization on NFH falls into sub-optimal solution



	Hop optimization		TDM optimization	
Testcase	NFH	f^d	NFH	f^d
sparcT1_core	0.83	0.90	1.00	1.00
neuron	1.00	1.00	1.00	1.00
stereo_vision	1.00	1.00	1.00	1.00
des90	1.00	1.00	1.00	1.00
SLAM_spheric	1.00	1.00	1.00	1.00
cholesky_mc	0.80	1.20	1.00	1.00
segmentation	0.94	1.00	1.00	1.00
bitonic_mesh	0.67	1.33	1.00	1.00
dart	1.00	1.00	1.00	1.00
openCV	1.00	1.20	1.00	1.00
stap_qrd	1.11	1.31	1.00	1.00
minres	1.00	1.00	1.00	1.00
cholesky_bdti	1.17	1.25	1.00	1.00
denoise	1.00	1.00	1.00	1.00
sparcT2_core	1.20	1.67	1.00	1.00
gsm_switch	1.33	2.11	1.00	1.00
mes_noc	0.67	1.00	1.00	1.00
LU230	1.20	1.14	1.00	1.00
LU_Network	0.83	1.00	1.00	1.00
sparcT1_chip2	0.75	1.19	1.00	1.00
directrf	0.80	1.11	1.00	1.00
bitcoin_miner	1.00	1.22	1.00	1.00
Geo. Mean	**0.95**	**1.14**	**1.00**	**1.00**

Outline

- Introduction
- Preliminary
- Problem Formulation
- Methodology
- Experiment results
- **Conclusion**

Conclusion

- We proposed an approach using analytical placement to integrate timing information and a global view on netlist into partitioning
- Improving cut size does not directly correlate to improve emulation performance
- Conventional partition metrics generate sub-optimal solution due to lack in consideration of
  - TDM impact
  - Hardware configuration

Thank you !