Clock-Aware UltraScale FPGA Placement with Machine Learning Routability Prediction

Chak-Wa Pui, Gengjie Chen, Yuzhe Ma, Evangeline F. Y. Young, Bei Yu

CSE Department, Chinese University of Hong Kong, Hong Kong

Speaker: Jordan, Chak-Wa Pui

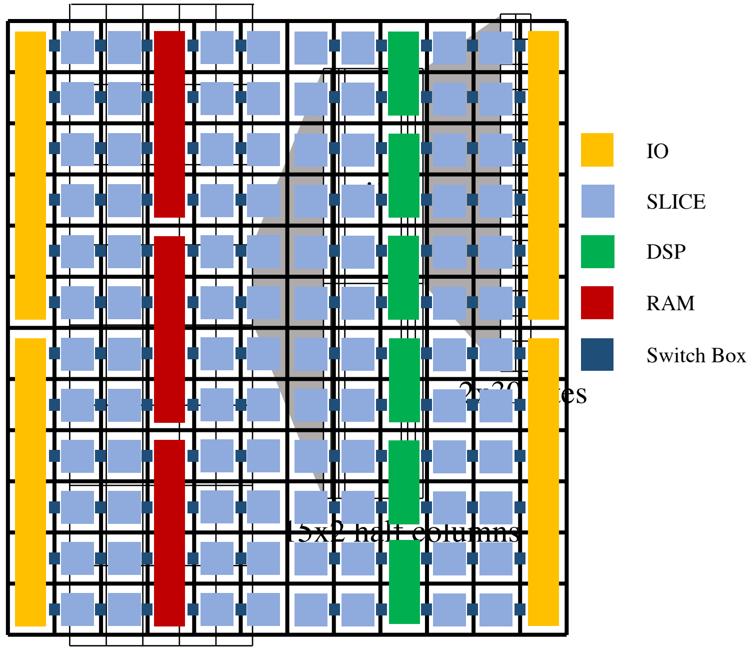
香 港 中 文 大 學
The Chinese University of Hong Kong

Outline

- Background
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

Introduction

- The architecture of heterogeneous FPGAs yields more sophisticated placement techniques
- The gap between FPGA and ASIC placement becomes smaller
  - Clock tree routing
  - Scale
  - Placement techniques
  - etc.
- As the scale of FPGA grows rapidly
  - routability becomes a major problem in placement



An illustration of Xilinx UltraScale architecture

An illustration of clock architecture of UltraScale

Previous Works

- Routablility-driven placement for UltraScale FPGAs
  - RippleFPGA[1]
  - UTPlaceF[2]
  - GPlace[3]
- Congestion estimation methods in FPGAs
  - Probabilistic model[1][4]
  - Global router[2]

[1] RippleFPGA: A routability driven placement for large-scale heterogeneous FPGAs. ICCAD2016
[2] UTPlaceF: A routability-driven FPGA placer with physical and congestion aware packing. ICCAD2016
[3] GPlace: A congestion-aware placement tool for UltraScale FPGAs. ICCAD2016
[4] A congestion driven placementalgorithm for fpga synthesis. FPL2006

Contributions

- Several placement techniques for UltraScale FPGAs to meet the challenges of clock constraints, routability, wirelength
    - A two-step displacement-driven legalization is introduced to remove all clock constraint violations
    - Chain move is proposed as a general framework to optimize placement
    - We study the performance of different routability prediction methods in FPGAs
- All the above techniques are incorporated into our FPGA placer

Problem Formulation

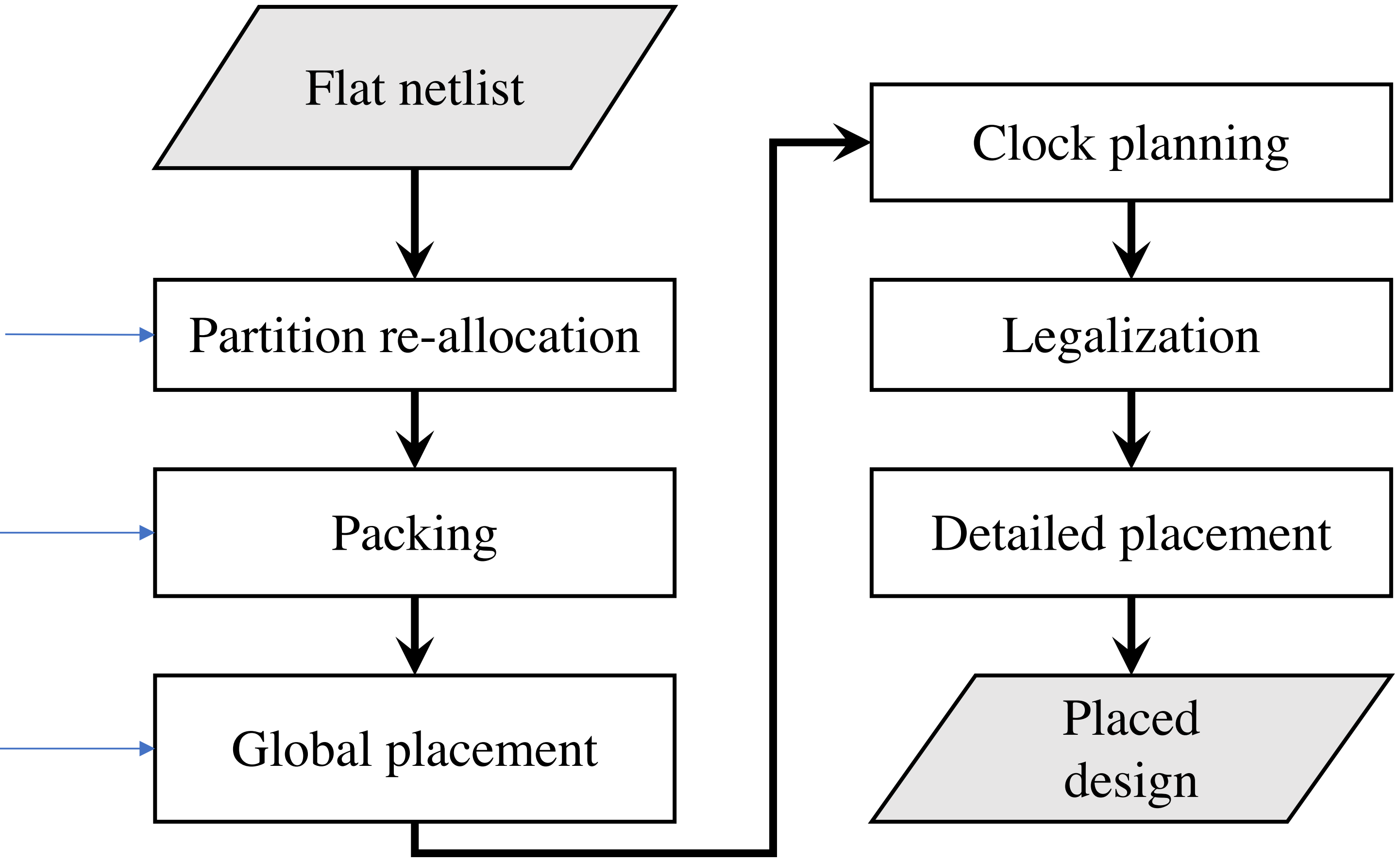
- Clock-Aware Routability-driven FPGA placement
  - Given the netlist and architecture of an FPGA
  - Minimize: routed wirelength measured by VIVADO
  - Subject to: each logic element has no overlap, no violation to the architecture specific legalization rules (basic rules and clock rules)

Overview of Our Framework

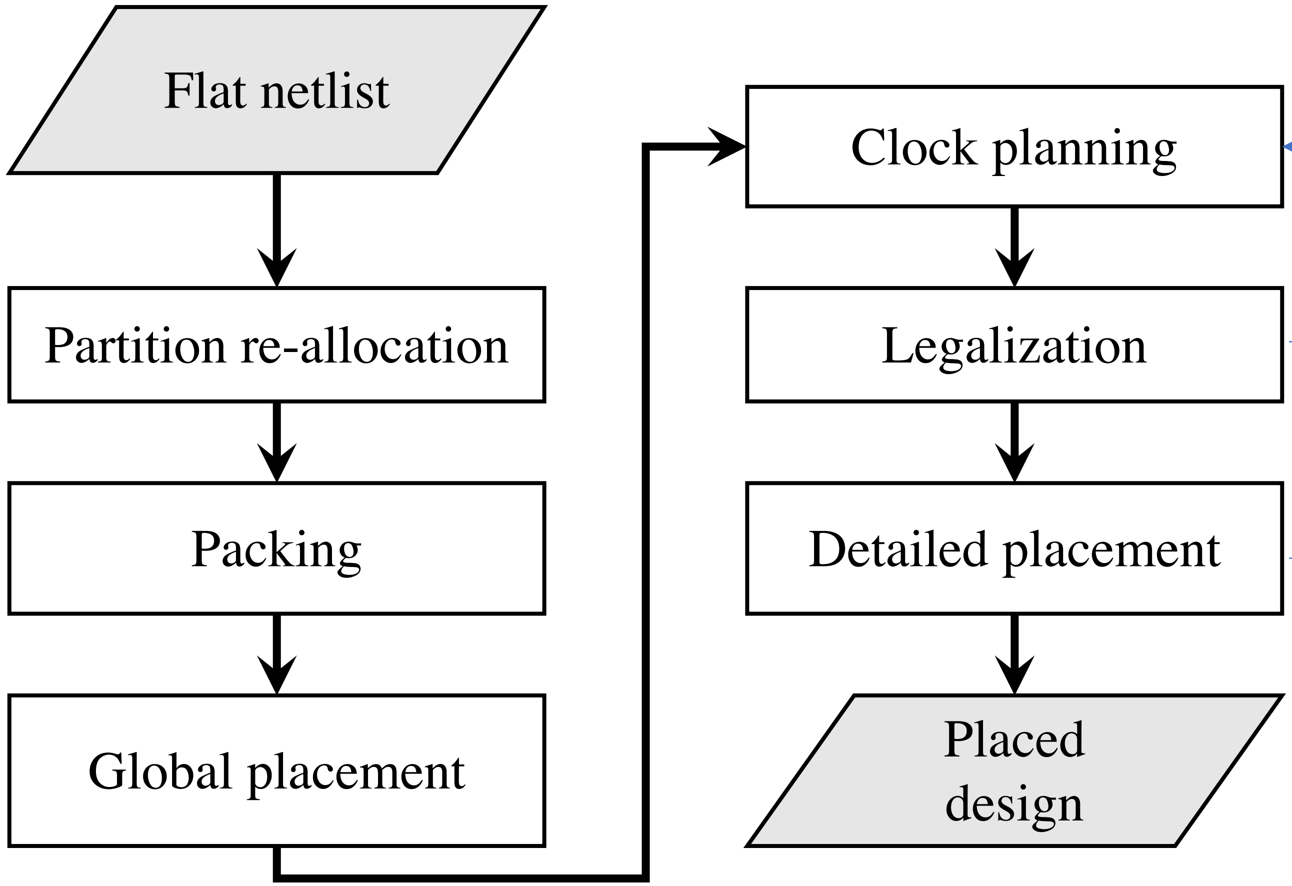
Reduce congestion caused by unbalanced routing supply in the horizontal and vertical directions

LUTs and FFs are packed into basic logic elements (BLEs) to reduce the inter-connections between sites in routing

Machine learning method is used to predict the routing congestion



Overview of Our Framework



Violations of the clock region constraint in global placement will be removed

- The placement is first legalized such that no violations regarding to rules in ISPD2016.
- Then violations of the half column constraint will be removed by half column legalization

Chain move is used to improve wirelength and displacement

Overview of Our Methods

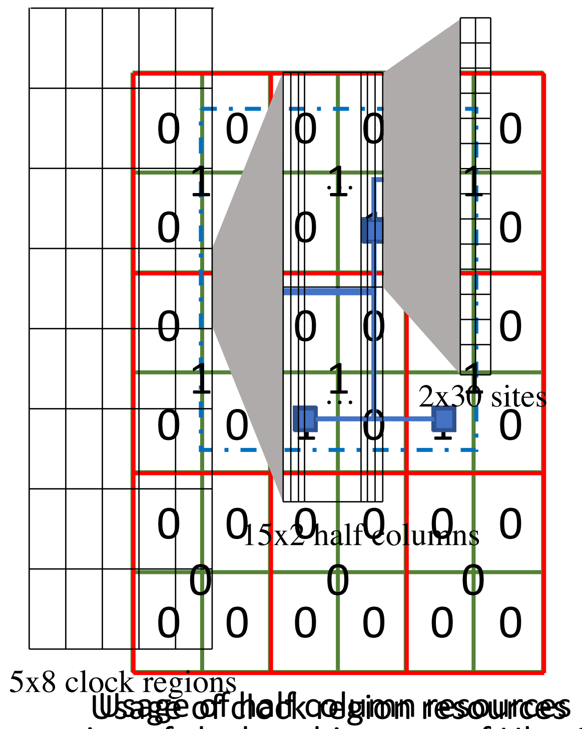
- Two-Step Clock Constraints Legalization
- Chain Move
- Machine Learning-Based Congestion Estimation

Overview of Our Methods

- Two-Step Clock Constraints Legalization
    - Clock Region Planning
    - Half Column Legalization
- Chain Move
- Machine Learning-Based Congestion Estimation

Two-Step Clock Constraints Legalization

- Clock constraints of UltraScale FPGAs
  - Clock region constraints
    - Bound box of the clock net
    - Violation: #clock is larger than 32
  - Half column constraints
    - Loads of the clock net
    - Violation: #clock is larger than 16
- Displacement-driven two-step legalization
  - Clock region planning
    - Remove all the clock region violations after global placement
  - Half Column Legalization
    - Remove all the half column violations after legalization

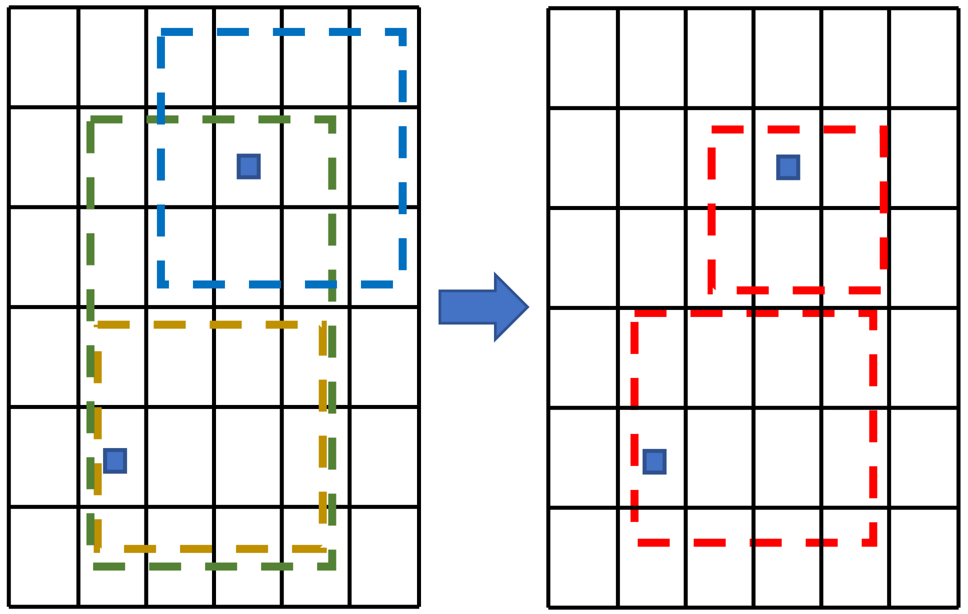


2x30 sites
15x2 half columns
5x8 clock regions
Usage of half column resources
Usage of clock region resources

An illustration of clock architecture of UltraScale

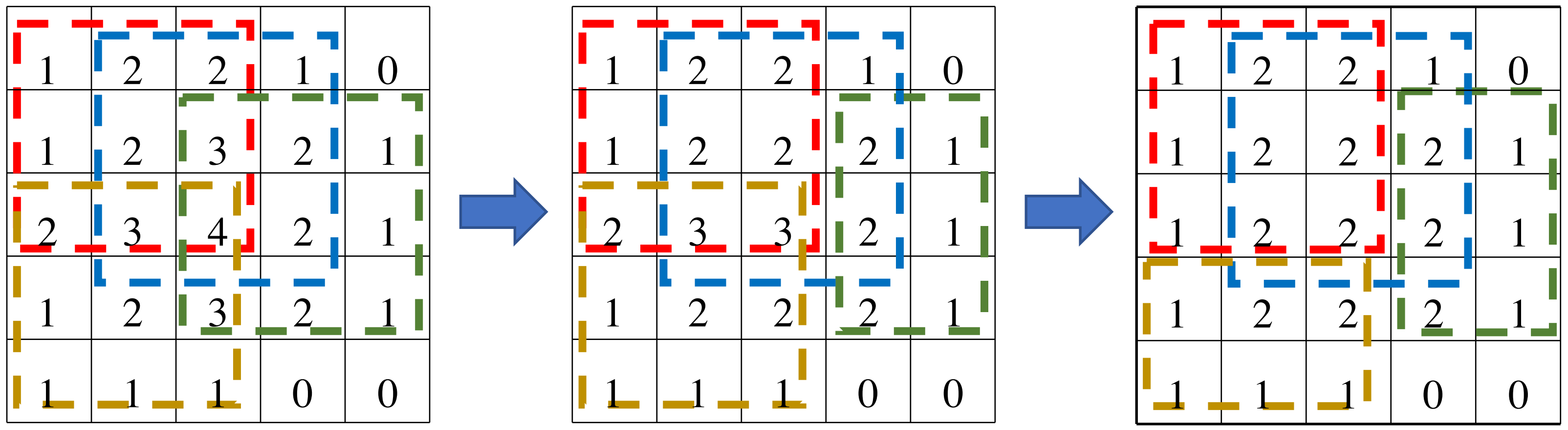
Two-Step Clock Constraints Legalization

- Two-Stage Clock region planning
  - Assign a bounding box to each cell such that there will be no violation if they stay in the box
  - Shrink Stage
  - Expand Stage



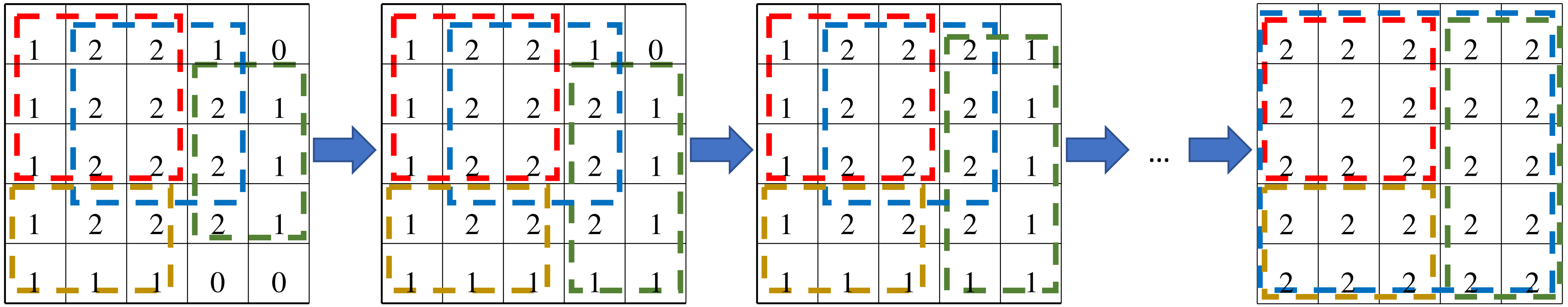
Two-Step Clock Constraints Legalization

- Two-Stage Clock region planning
  - Shrink Stage
    - iteratively shrink the bounding box of each clock
    - shrink the BB of the clock in the most overflowed clock region such that it induces smallest displacement. Move the corresponding cells to the boundary.
  - Expand Stage



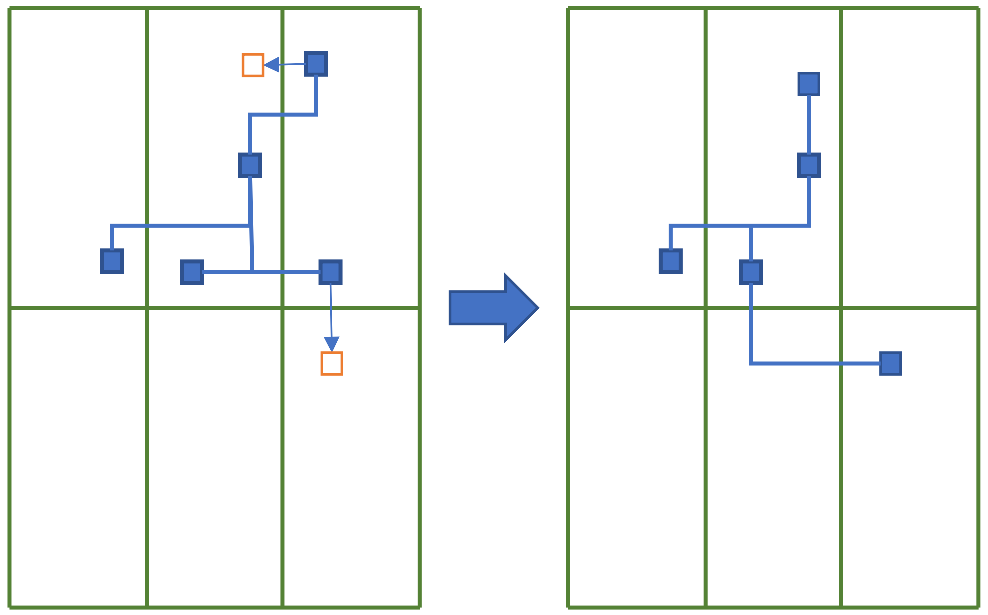
Two-Step Clock Constraints Legalization

- Two-Stage Clock region planning
  - Shrink Stage
  - Expand Stage
    - iteratively expand the bounding box of each clock
    - increase the width/height of the clock BB with highest cell density by 1 unit. Direction is determined such that the cell density of resulted BB is smallest



Two-Step Clock Constraints Legalization

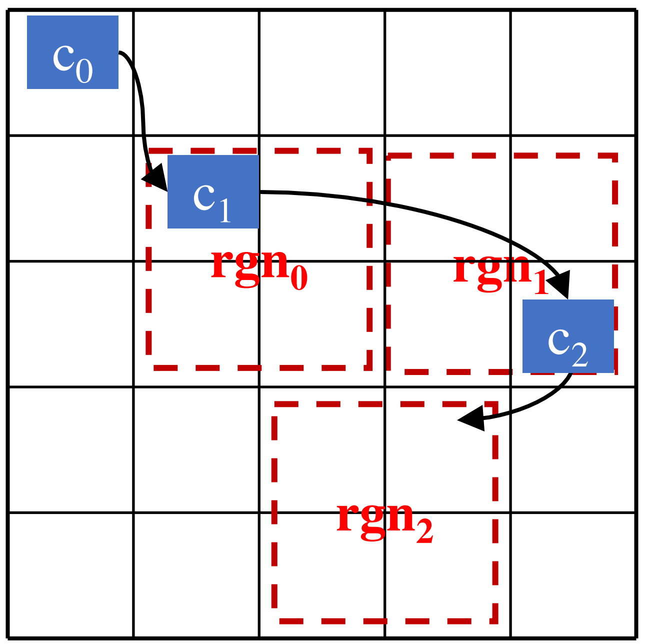
- Half Column Legalization
  - All the future movement cannot induce any new half column violation
  - Iteratively select the most overflow column and remove the clock such that the smallest displacement is induced
  - Each load will be moved to its nearest site in another half column



Overview of Our Methods

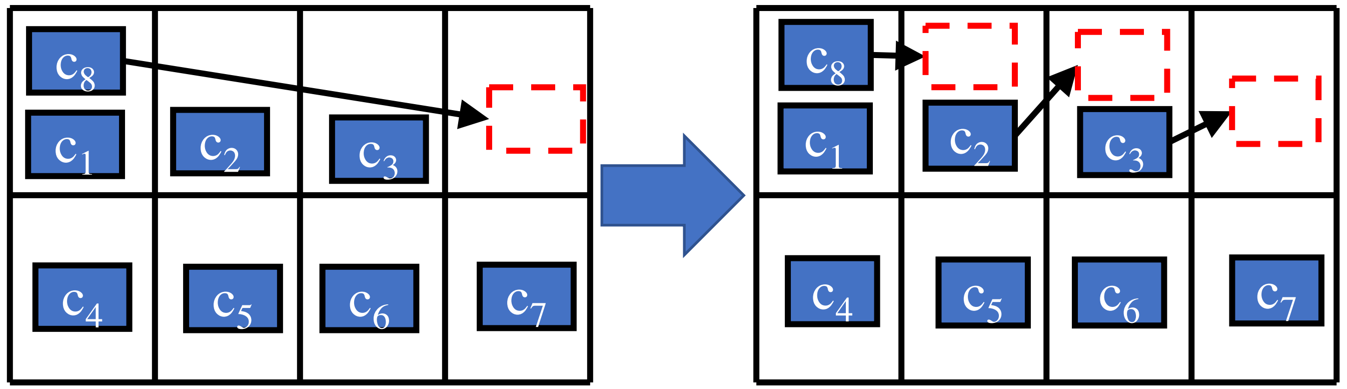
- Two-Step Clock Constraints Legalization
- Chain Move
- Machine Learning-Based Congestion Estimation

Chain Move



- Motivation
  - Reduce the quality loss due to sequential placement
- Generate a sequence of cell moves such that
  - all of cells involved are legal after the move
  - the objective is improved
- DFS-based
  - Limit the number of trials of each cell and the length of the chain
- General framework, easy to modify
  - The objective is optimized by selecting the candidate sites of each cell

Chain Move



- Applications
  - Reduce Max. and Total Displacement in Legalization
    - Max. Displacement Mode
      - Invoked when the displacement of c_i is larger than D_{max}
      - The resulted chain move should satisfy:
        - The total displacement should be no larger than the original
        - The displacement of each moved cell should be no larger than the original displacement of the first cell
    - Total Displacement Mode
  - Reduce the distance to optimal region in detailed placement

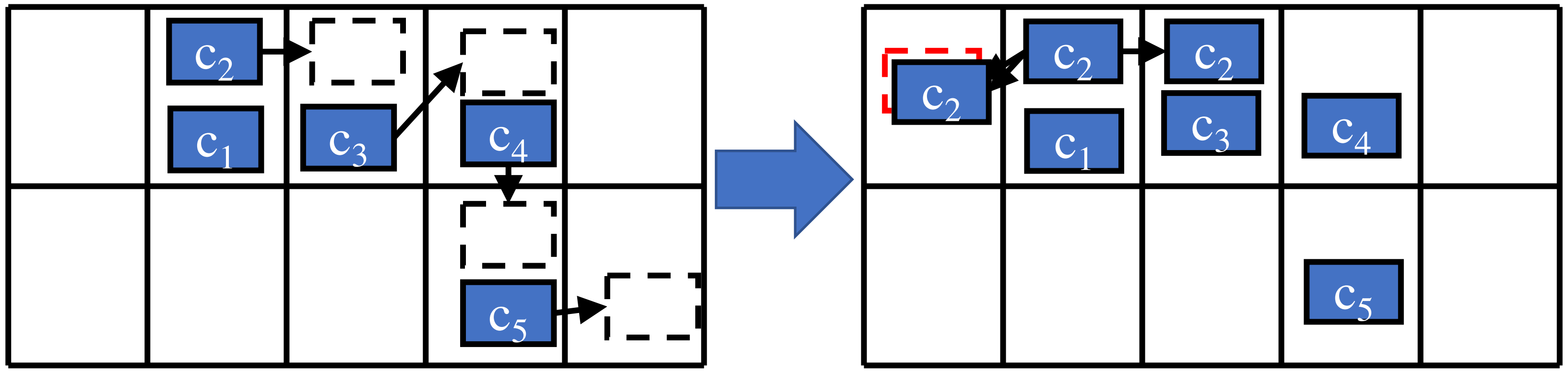
Chain Move

- Applications
  - Reduce Max. and Total Displacement in Legalization
    - Max. Displacement Mode
    - Total Displacement Mode
      - Invoked c_i cannot be legalized with displacement d
      - The displacement of any cell c_j in the chain should satisfy,

$$dist(s, c_j) \begin{cases} = d, & \text{if } c_j \text{ is the first cell in the chain,} \\ \leq dist(s_j, c_j), & \text{otherwise,} \end{cases}$$

  - Reduce the distance to optimal region in detailed placement

Chain Move

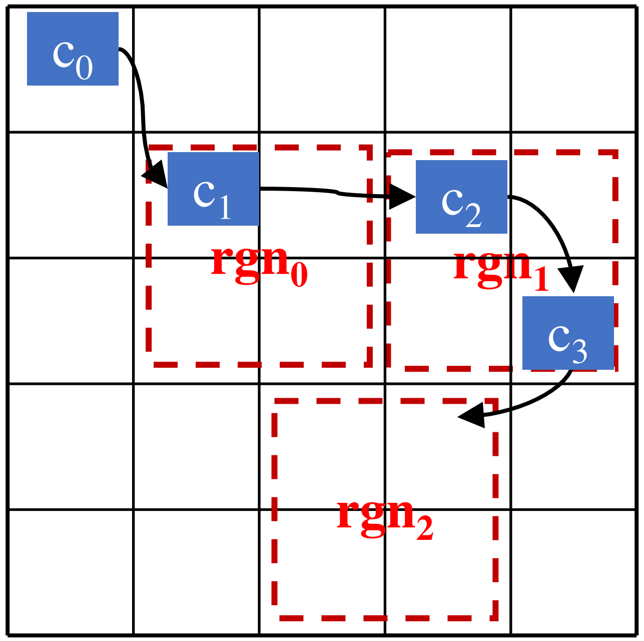


- Applications
  - Reduce Max. and Total Displacement in Legalization
    - Max. Displacement Mode
    - Total Displacement Mode
      - Invoked c_i cannot be legalized with displacement d
      - The displacement of any cell c_j in the chain should satisfy,

$$dist(s, c_j) \begin{cases} = d, & \text{if } c_j \text{ is the first cell in the chain,} \\ \leq dist(s_j, c_j), & \text{otherwise,} \end{cases}$$

  - Reduce the distance to optimal region in detailed placement

Chain Move



- Applications
  - Reduce Max. and Total Displacement in Legalization
    - Max. Displacement Mode
    - Total Displacement Mode
  - Reduce the distance to optimal region in detailed placement
    - The candidate cells of each cell are those that are in its optimal region

Overview of Our Methods

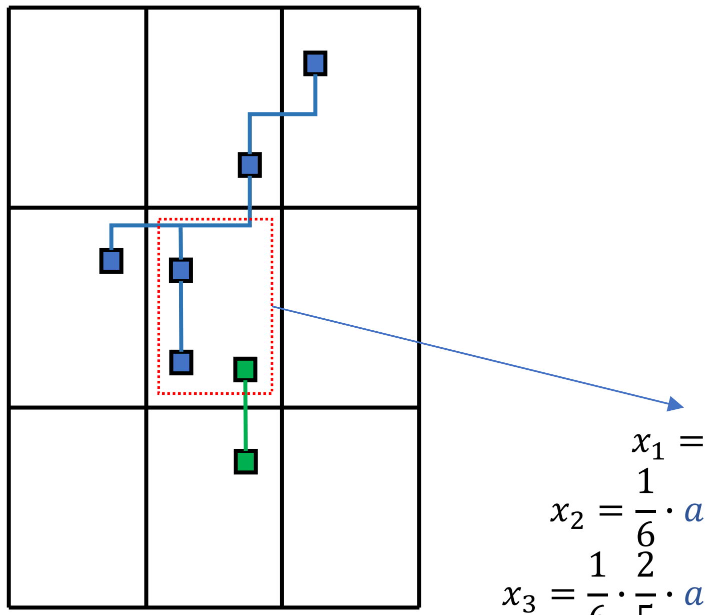
- Two-Step Clock Constraints Legalization
- Chain Move
- Machine Learning-Based Congestion Estimation

ML-Based Congestion Estimation

- Motivation:
  - More accurate and less parameter tunings
    - Previously used congestion estimation methods in FPGAs
      - Global routers for ASICs
      - Probabilistic models
    - Limitations:
      - Not tailored for FPGAs
      - A lot of parameters to set
- Goals of our methods
  - Try to mimic the behavior of congestion estimation of design tools from the device company
    - Assume the congestion estimation from the tool can guide the placement well
  - Study how to leverage machine learning to build a congestion model on FPGA

ML-Based Congestion Estimation

- Congestion Model
  - G-Cells based, each corresponds to a switchbox
- Three Features for each G-Cell
  - Total number of pins of the net covering it
    - $x_1 = \sum_{m \in N_i} \#pins\ of\ net\ m$
  - A weighted sum of BB box covering it
    - $x_2 = \sum_{m \in N_i} \frac{w_m \cdot HPWL_m}{\#gcell_m}$
  - Combining the two
    - $x_3 = \sum_{m \in N_i} \frac{p_{m,i}}{\#pins\ of\ net\ m} \cdot \frac{w_m \cdot HPWL_m}{\#gcell_m}$



$x_1 = 7$

$x_2 = \frac{1}{6} \cdot a + \frac{1}{2} \cdot b$

$x_3 = \frac{1}{6} \cdot \frac{2}{5} \cdot a + \frac{1}{2} \cdot \frac{1}{2} \cdot b$

(a, b are the weighted wirelength of the two nets)

ML-Based Congestion Estimation

- Learning Models
  - Local Linear Model
    - Only consider the current site
    - $y = f_{llm}(X) = \sum_{i=1}^{3} w_i x_i$
  - Hierarchical Hybrid Model
    - Two-Layer
    - Use the value of the local linear model as the first layer result $y_{hm1} = \sum_{i=1}^{3} w_i x_i$
    - The second layer use the SVM as the machine learning model with the y_{hm1} value of the site and its neighboring 8 sites as features
      - $y_{hm2} = f_{SVM}\ (y_{hm1}^{1}, \dots, y_{hm1}^{9})$
  - Global Linear Model
    - Consider the current site and its neighboring 8 sites
    - $y = f_{glm}(X) = \sum_{i=1}^{27} w_i x_i$

ML-Based Congestion Estimation

- Training Methods
  - Unified model
    - One model for all design
    - Pros: generalize well
  - Independent model
    - Different model for different design
    - Pros: capture the unique characteristics of different design
  - Ensemble model
    - Different model for different known design
    - Ensemble all the known models to generate a model for new designs
    - $y = \sum_{i=1}^{N} \frac{1}{N} \cdot f_{glm,i}(x)$

ML-Based Congestion Estimation

- Experiments setting
  - Unified and Independent
    - 70% training and 30% testing per design
  - Unified+ and Ensemble
    - 12 design for training, others for testing
- Result Analysis
  - Training Method
    - Unified is better than independent in our test
      - Why? Similar designs
  - Model
    - Global models are better than local model
    - Global linear model is best, SVM perform worse
      - Why? Features are linear to the golden results
    - Both unified and ensemble model can generalize well to other designs

Model	Unified		Independent	
	Avg. r^2	Avg. MPE	Avg. r^2	Avg. MPE
Local Linear Model	0.891	16.1	0.878	17.6
Hierachical Hybrid Model	-	-	0.833	16.3
Global Linear Model	0.943	11.5	0.933	12.8

Model	Unified+		Ensemble	
	Avg. r^2	Avg. MPE	Avg. r^2	Avg. MPE
Global Linear Model	0.914	17.2	0.926	16.3

ML-Based Congestion Estimation

- Comparison
  - Global routers for ASICs
    - Cons: hard to set the routing capacity
  - Probabilistic models
    - Cons: only good correlation with the relative congestion level
  - Machine Learning-Based
    - Good correlation with the congestion level
    - Give a better sense of congestion level
    - Less parameter tuning

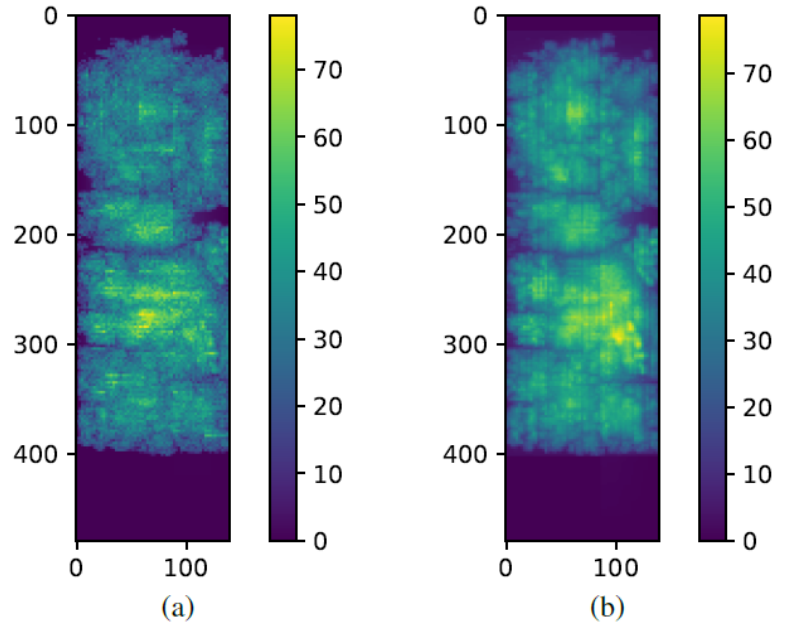


Fig. 7: Congestion Maps of CLK-FPGA13 in ISPD2017 contest: (a) routing congestion predicted by Vivado; (b) our estimation.

Experimental Result

	This Work				1st Place				2nd Place				3rd Place			
Design	WL	**ratio**	Time	**ratio**	WL	**ratio**	Time	**ratio**	WL	ratio	Time	ratio	WL	ratio	Time	ratio
CLK-FPGA01	2011452	**1**	288	**1**	2208170	**1.098**	530	**1.84**	2209328	1.098	2686	9.326	2268532	1.128	2686	9.326
CLK-FPGA02	2167861	**1**	266	**1**	2279171	**1.051**	521	**1.959**	2273729	1.049	2788	10.481	2504444	1.155	2788	10.481
CLK-FPGA03	5265206	**1**	583	**1**	5353071	**1.017**	1038	**1.78**	6229292	1.183	3740	6.415	5803110	1.102	3740	6.415
CLK-FPGA04	3606567	**1**	380	**1**	3697950	**1.025**	725	**1.908**	3817377	1.058	2850	7.5	4085670	1.133	2850	7.5
CLK-FPGA05	4660136	**1**	569	**1**	4692356	**1.007**	943	**1.657**	4995177	1.072	3164	5.561	5180916	1.112	3164	5.561
CLK-FPGA06	5736998	**1**	591	**1**	5588507	**0.974**	1075	**1.819**	5605573	0.977	3570	6.041	6216898	1.084	3570	6.041
CLK-FPGA07	2325787	**1**	304	**1**	2444359	**1.051**	585	**1.924**	2504544	1.077	3698	12.164	2676088	1.151	3698	12.164
CLK-FPGA08	1778292	**1**	247	**1**	1885632	**1.06**	482	**1.951**	1989632	1.119	2504	10.138	2057117	1.157	2504	10.138
CLK-FPGA09	2530105	**1**	327	**1**	2601161	**1.028**	600	**1.835**	2583442	1.021	3158	9.657	2813538	1.112	3158	9.657
CLK-FPGA10	4495500	**1**	512	**1**	4464341	**0.993**	868	**1.695**	4770168	1.061	2971	5.803	4839765	1.077	2971	5.803
CLK-FPGA11	4189622	**1**	455	**1**	4182726	**0.998**	768	**1.688**	4207699	1.004	2535	5.571	4777177	1.14	2535	5.571
CLK-FPGA12	3387586	**1**	409	**1**	3368698	**0.994**	744	**1.819**	3376930	0.997	3007	7.352	3739517	1.104	3007	7.352
CLK-FPGA13	3833106	**1**	441	**1**	3815718	**0.995**	822	**1.864**	3920965	1.023	3155	7.154	4320345	1.127	3155	7.154
Average		**1**		**1**		**1.03**		**1.84**		1.073		7.933		1.126		7.933

Routed wirelength and running time (s) comparison with the ISPD 2017 contest winners

Experimental Result

	w/ CCL				w/o CCL			
Design	HPWL	ratio	Time	ratio	HPWL	ratio	Time	ratio
CLK-FPGA01	1582915	1	288	1	1582917	1.000	276	0.958
CLK-FPGA02	1577051	1	266	1	1577175	1.000	254	0.955
CLK-FPGA03	4059162	1	583	1	4060708	1.000	558	0.957
CLK-FPGA04	2716961	1	380	1	2717722	1.000	367	0.966
CLK-FPGA05	3532759	1	569	1	3533407	1.000	534	0.938
CLK-FPGA06	4485498	1	591	1	4486401	1.000	572	0.968
CLK-FPGA07	1708920	1	304	1	1708954	1.000	293	0.964
CLK-FPGA08	1355308	1	247	1	1354247	0.999	244	0.988
CLK-FPGA09	1946225	1	327	1	1945948	1.000	313	0.957
CLK-FPGA10	3505733	1	512	1	3506732	1.000	499	0.975
CLK-FPGA11	3270338	1	455	1	3270689	1.000	440	0.967
CLK-FPGA12	2592324	1	409	1	2593721	1.001	395	0.966
CLK-FPGA13	2927103	1	441	1	2926786	1.000	420	0.952
Average		1.000		1.000		**1.000**		**0.962**

Comparison of HPWL and running time (s) before and after applying the two-step clock constraint legalization (CCL)

Experimental Result

Design	RippleFPGA[1]		This work	
	WL	ratio	WL	ratio
FPGA01	350060	1	350802	1.002
FPGA02	635044	1	634700	0.999
FPGA03	3251264	1	3251721	1.000
FPGA04	5492214	1	5411107	0.985
FPGA05	9909270	1	9911182	1.000
FPGA06	6144522	1	6143973	1.000
FPGA07	9593240	1	9520252	0.992
FPGA08	8087931	1	8036647	0.994
FPGA09	12062928	1	12123865	1.005
FPGA10	6972278	1	7020054	1.007
FPGA11	10918250	1	10462601	0.958
FPGA12	7239553	1	7605996	1.051
Average		1		**0.999**

Routed wirelength comparison between different routing congestion estimation models.

[1] RippleFPGA: A routability driven placement for large-scale heterogeneous FPGAs. ICCAD2016

Conclusion

- A two-step displacement-driven legalization is introduced to remove all clock constraint violations with almost neglectable overhead in practice
- Chain move is proposed as a general framework to optimize placement
- We study the performance of different routability prediction methods in FPGAs which save time in congestion-driven global placement and ease the burden of parameter tuning
- All of the above techniques together can achieve 3% shorter wirelength and about 2X runtime compared to ISPD2017 contest winner

Thanks

香 港 中 文 大 學
The Chinese University of Hong Kong