國立交通大學電子工程學系

Novel Pulsed-Latch Replacement Based on Time Borrowing and Spiral Clustering

Chih-Long Chang
Iris Hui-Ru Jiang
Yu-Ming Yang
Evan Yu-Wen Tsai
Aki Sheng-Hua Chen

NCTU
FARADAY

IRIS Lab

National Chiao Tung University

Outline

Introduction

Preliminaries

Feasible region

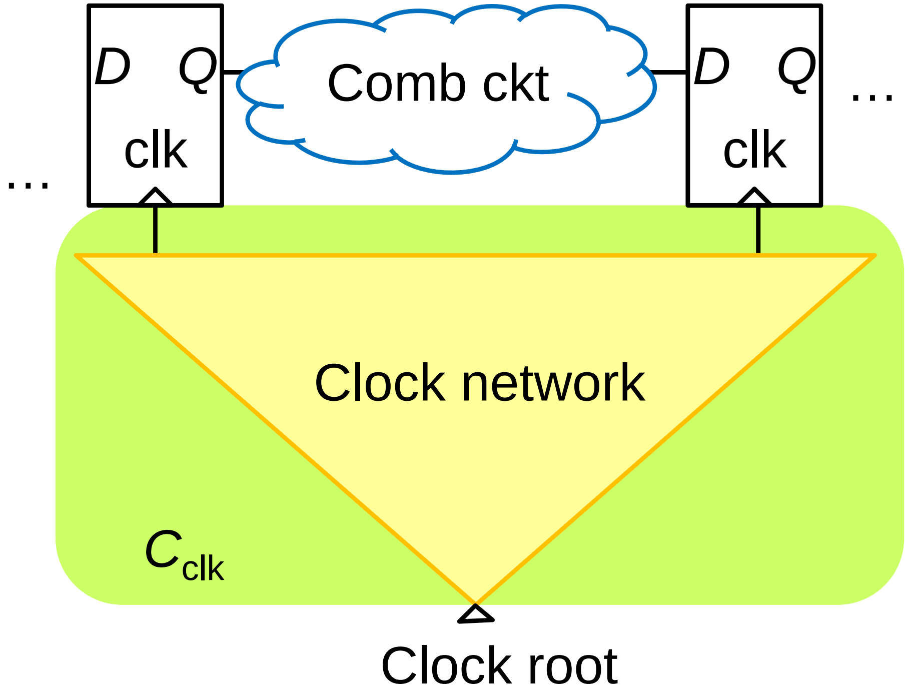
Algorithm

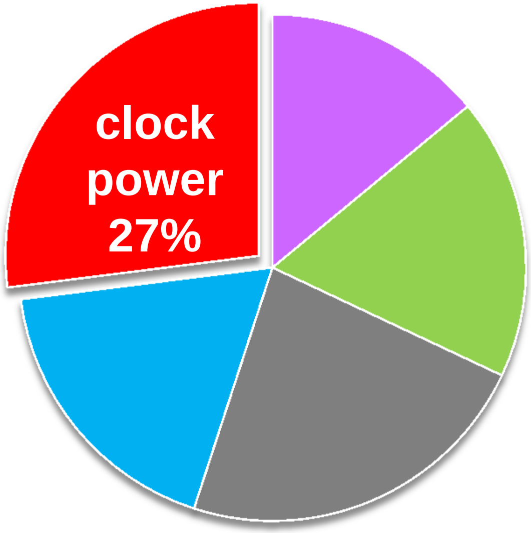
Experimental results

Conclusion

Clock Power Dominates!

- □ **Clock power is the major contributor of total chip power consumption**
- □ **Large portion of it is consumed by sequencing elements**
- □ **Minimize the sequencing overhead!**



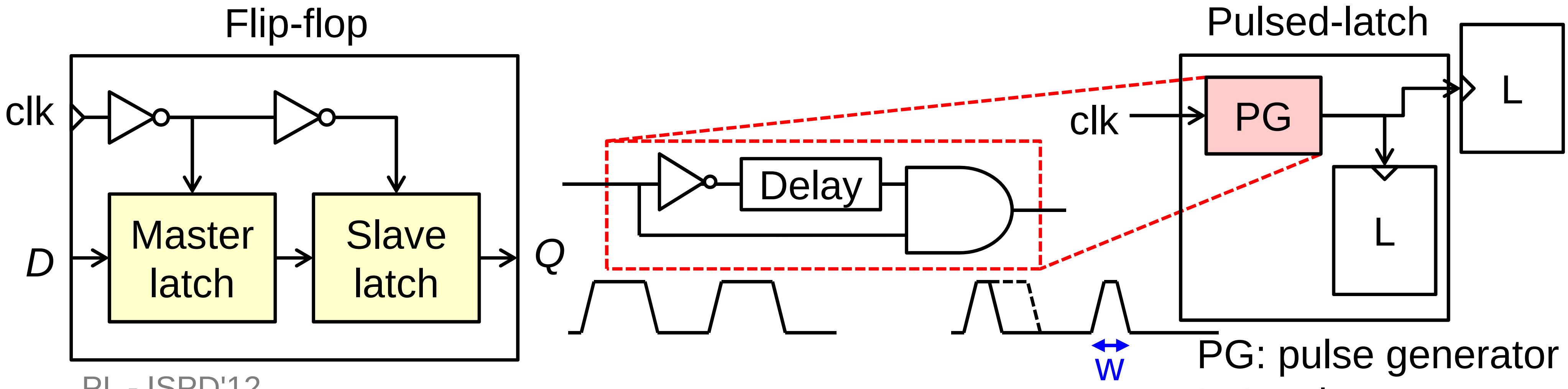


Power breakdown of an ASIC

Chen *et al*. Using multi-bit flip-flop for clock power saving by DesignCompiler. *SNUG*, 2010.

Flip-Flops vs. Pulsed-Latches

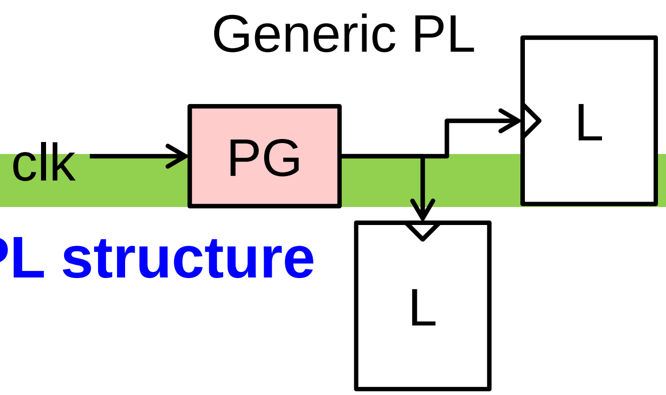
- **Flip-flop (FF)**
  - The most common form of sequencing elements
  - Two cascaded latches triggered by a clock signal
  - High sequencing overhead in terms of delay, power, area
- **Pulsed-latch (PL)**
  - A latch synchronized by a pulse clock
  - A PL can be approximated as a fast, low-power, and small FF
  - Promising to reduce power for high performance circuits
- **Migrate from a FF-based design to a PL-based counterpart to reduce the sequencing overhead**



PL - ISPD'12

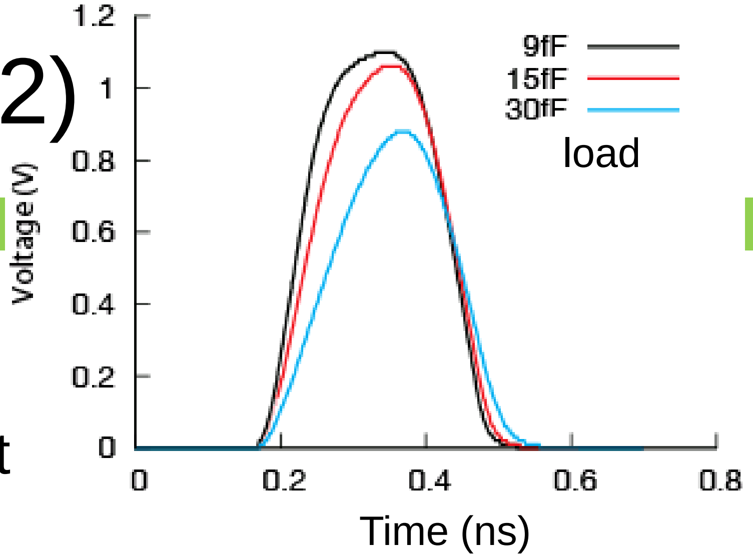
PG: pulse generator
L: Latch

Prior Work

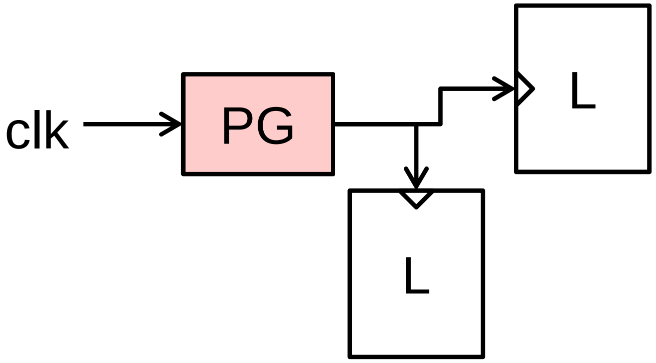


- □ **Most of previous works adopt the generic PL structure and flip-flop-like timing analysis**
- □ **Pulse distortion**
  1. Chuang *et al.* [DAC'10] propose a PL-aware analytical placer, controlling pulse distortion by limiting the # of PLs and total WL driven by each PG (no timing consideration)
- □ **Timing**
  2. Lee *et al.* [ICCAD'08], Lee *et al.* [ICCAD'09] and Paik *et al.* [ASPDAC'10] apply aggressive time borrowing techniques (clock skew scheduling, pulse width allocation, retiming)
- □ **Power**
  3. Shibatani and Li [EETimes'06] propose a methodology
  4. Kim *et al.* [ASPDAC'11] generate clock gating functions of PGs
  5. Lin *et al.* [ISLPED'11] minimize # of PGs without considering clock gating
  6. Chuang *et al.* [ICCAD'11] perform placement and clock network co-synthesis (based on 1 and 5)

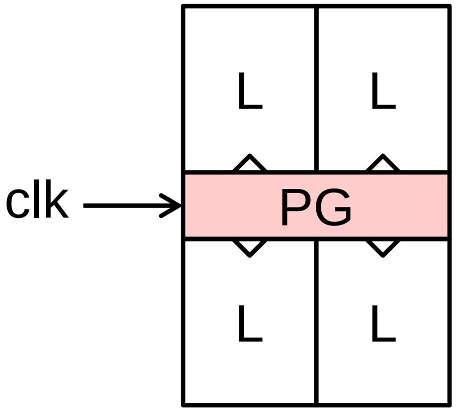
Multi-bit Pulsed-Latches (1/2)



- □ **The generic PL structure**
  - Pulses can easily be distorted since the PG and latches are placed apart
- □ **Multi-bit pulsed-latches**
  - The PG and latches are placed and hard-wired together in a compact and symmetric form
  - The pulse distortion and clock skew can be well controlled



Generic pulsed latch:
pulse generator (PG) and latches (L)



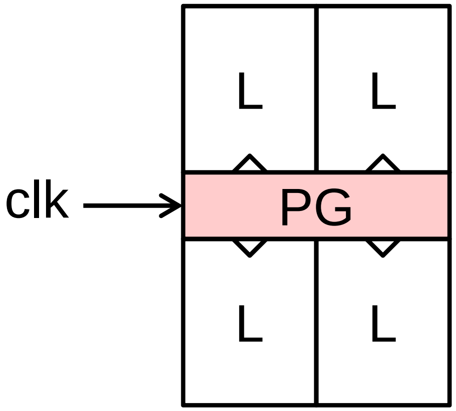
Multi-bit pulsed latch:
hardwired PG and L together

Chuang et al. Pulsed-latch-aware placement for timing-integrity optimization. DAC-10.
Farmer, et al. Pipeline array. US patent 6856270 B1, 2005.
Venkatraman et al., “A robust, fast pulsed flip-flop design,” GLSVLSI-08.

Multi-bit Pulsed-Latches (2/2)

- **Multi-bit pulsed-latches are more power efficient than single-bit pulsed latch.**

Bit Number	Normalized power per bit
1	1.000
2	0.740
4	0.613
8	0.575



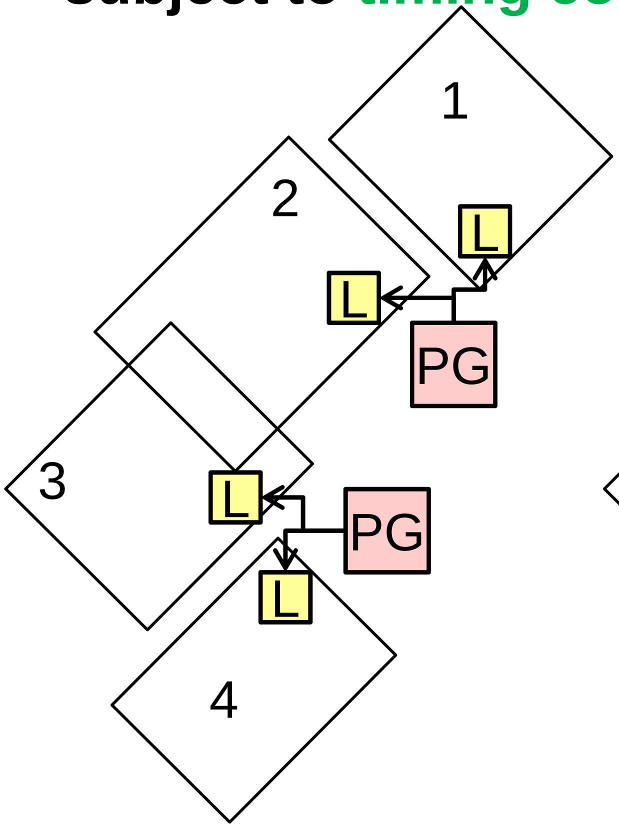
Multi-bit pulsed latch:
hardwired PG and L together

Do We Need Aggressive Time Borrowing?

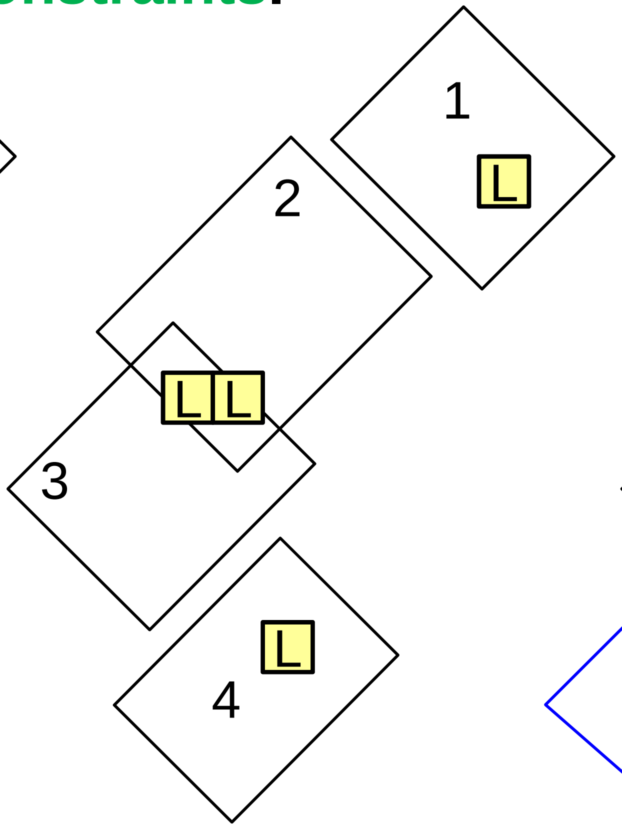
- **Under flip-flop-like timing analysis, prior works use aggressive time borrowing techniques**
  - Various pulse widths, clock skew scheduling, and retiming may induce some difficulties on timing closure and functional verification
- **Latches have the time borrowing property**
  - STA tools are mature to handle time borrowing
  - The amount of time borrowing offered by the pulse width is significant for high performance circuits
- **We can utilize only the intrinsic time borrowing of latches to provide flexibility to relocate pulsed-latches**

How About MBPL Replacement?

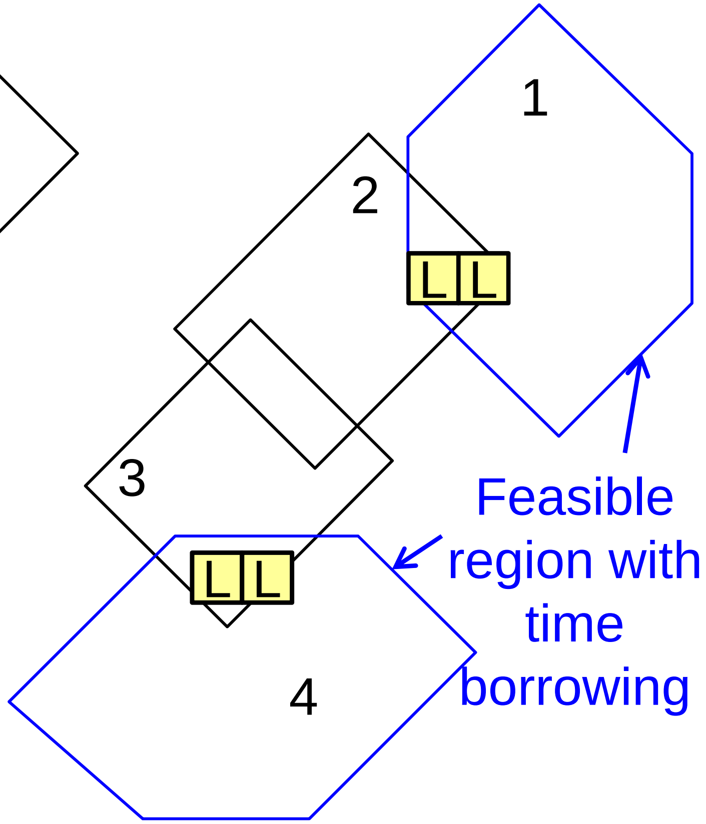
- **Based on the multi-bit pulsed-latch structure and time borrowing offered by the pulse width, we apply post-placement pulsed-latch replacement to minimize power consumption subject to timing constraints.**



Generic pulsed latches without time borrowing may incur pulse distortion

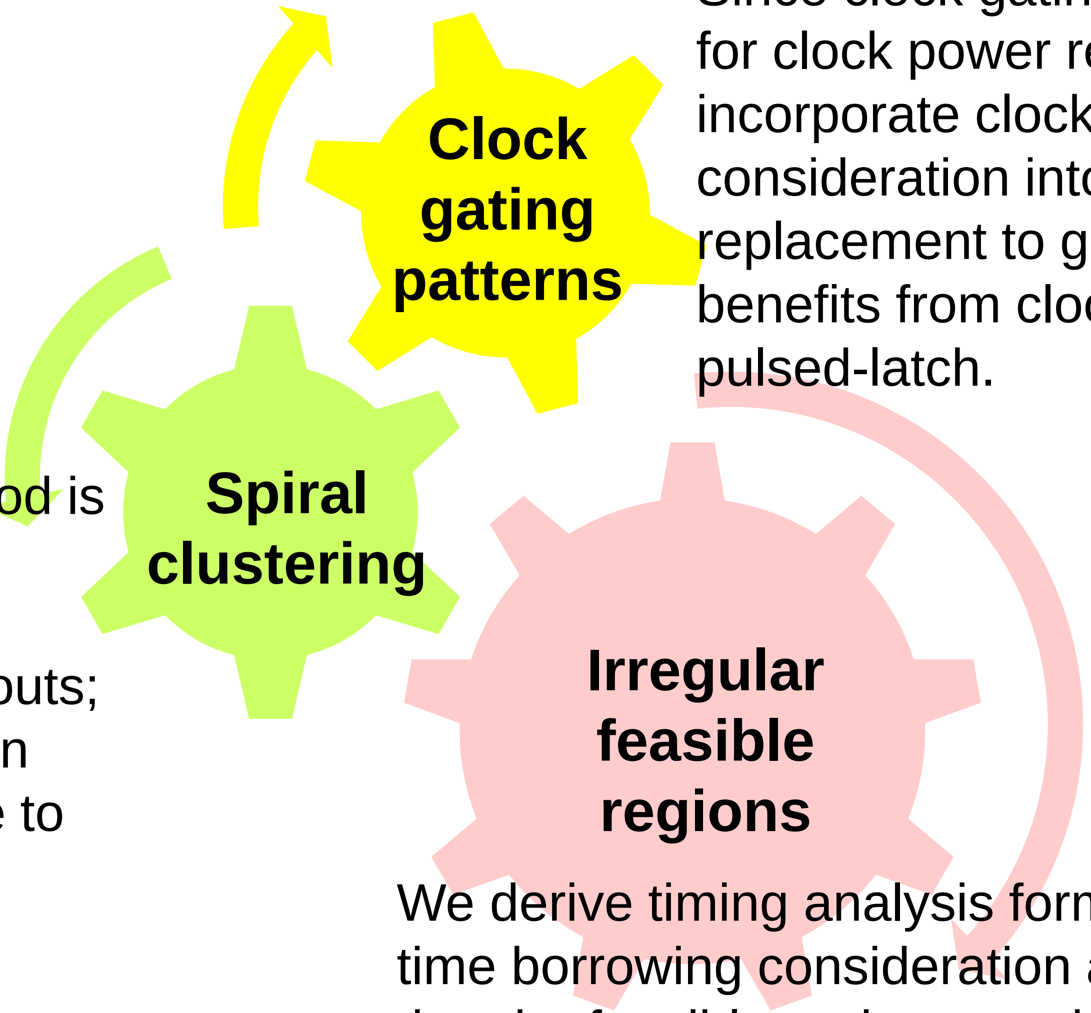


MBPL without time borrowing



MBPL with time borrowing

Our Contributions



Clock gating patterns

Spiral clustering

Irregular feasible regions

Since clock gating is widely used for clock power reduction, we incorporate clock gating consideration into pulsed-latch replacement to gain double benefits from clock gating and pulsed-latch.

Spiral clustering method is suitable for not only rectangular but also rectilinear shaped layouts; the latter are popular in modern IC design due to macros.

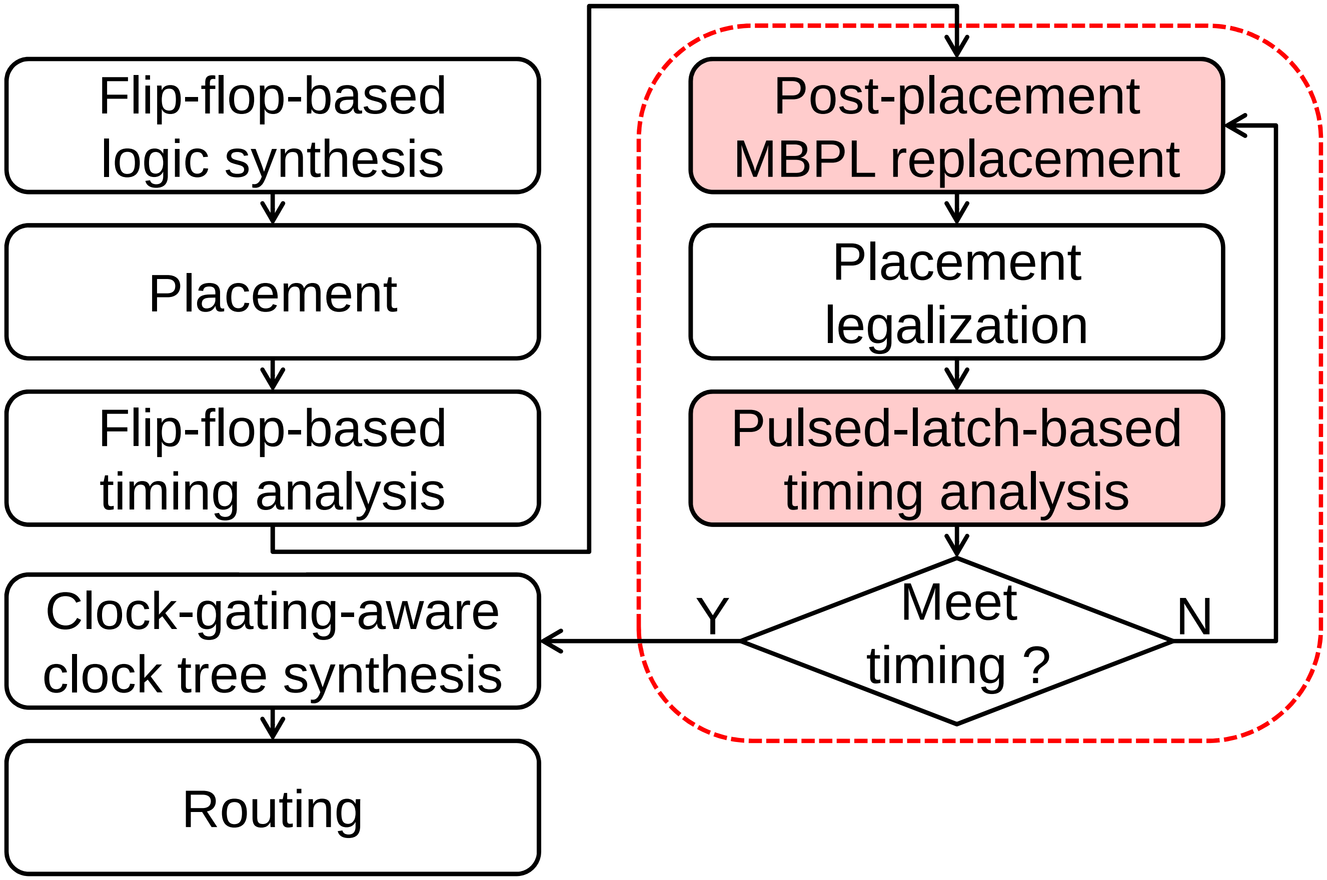
We derive timing analysis formulae with time borrowing consideration and reveal that the feasible regions can be very irregular. We adopt an efficient representation to manipulate them.

Outline

- Introduction
- **Preliminaries**
- Feasible region
- Algorithm
- Experimental results
- Conclusion

The Pulsed-Latch Migration Flow

- **We replace flip-flops by multi-bit pulsed-latches based on their timing slacks and the available amount of time borrowing.**


Flip-flop-based logic synthesis
Placement
Flip-flop-based timing analysis
Clock-gating-aware clock tree synthesis
Routing
Post-placement MBPL replacement
Placement legalization
Pulsed-latch-based timing analysis
Meet timing ?
Y
N

Problem Formulation

- **The Multi-Bit Pulsed-Latch Replacement problem:**
- **Given**
  - A multi-bit pulsed-latch library
  - Nelist & placement of a design
  - The timing slacks
  - Clock gating patterns of flip-flops
- **Goal**
  - Replace flip-flops by multi-bit pulsed-latches with time borrowing
  - Minimize power on pulsed-latches
  - Subject to timing slack and placement density constraints

Outline

Introduction

Preliminaries

Feasible region

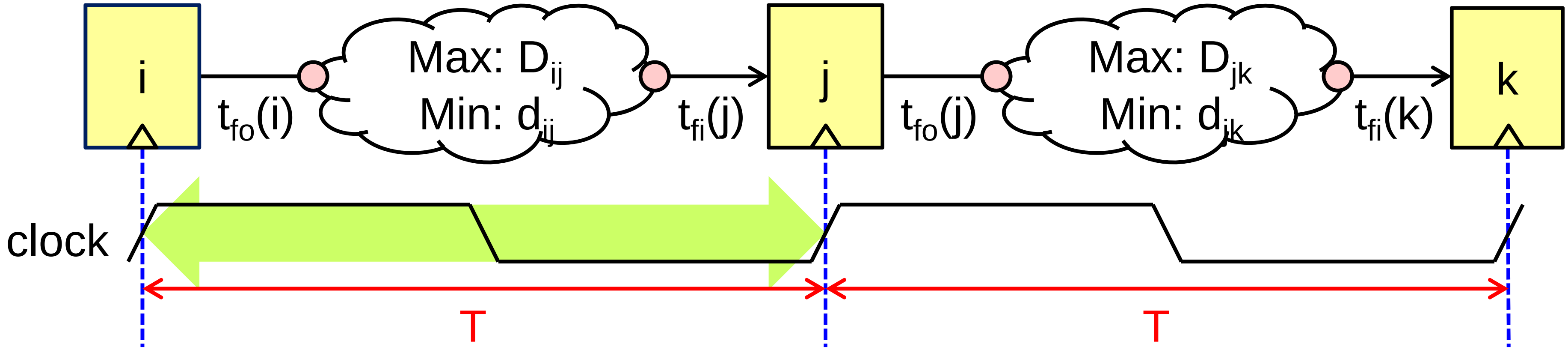
Algorithm

Experimental results

Conclusion

Timing Analysis – Flip-flops

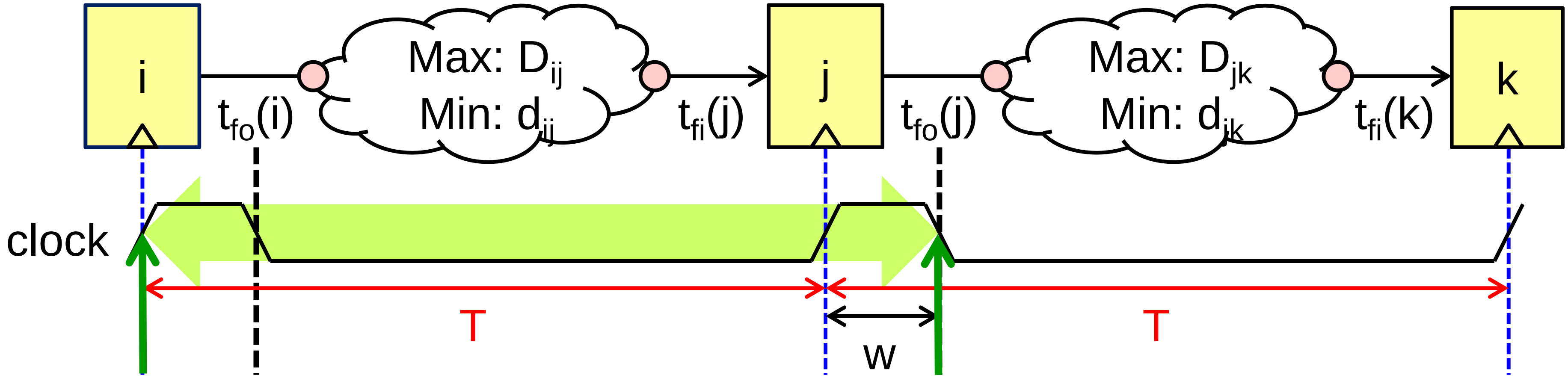
- **Flip-flop**



- Setup $t_{cq} + t_{fo}(i) + D_{ij} + t_{fi}(j) + t_{su} \leq T$
- Hold $t_{cq} + t_{fo}(i) + d_{ij} + t_{fi}(j) - t_{hd} \geq M$

Timing Analysis – Pulsed-latches (1/2)

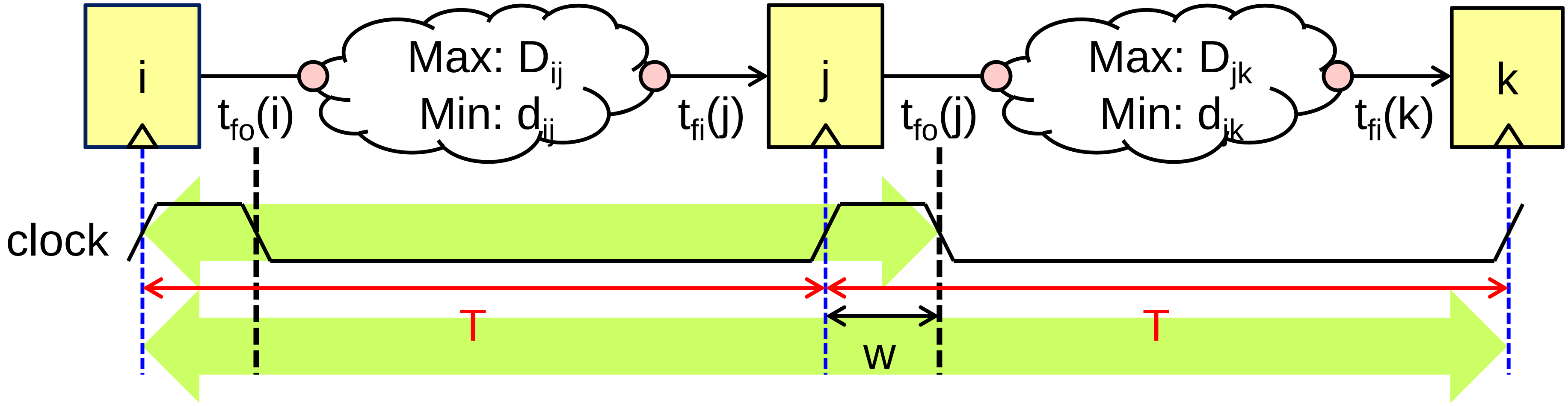
- **Pulsed-latch**



- When we replace flip-flops with pulsed-latches, the data can depart the launching latch on the rising edge of the clock, but does not have to set up until the falling edge of the clock on the receiving latch.
- If the maximum delay from i to j exceeds a cycle period, it can borrow time from the delay from j to k.

Timing Analysis – Pulsed-latches (2/2)

- **Pulsed-latch**

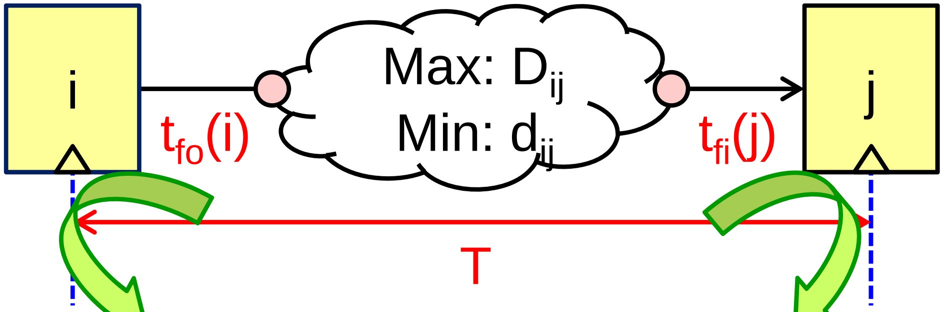


- Setup
$$t'_{cq} + t_{fo}(i) + D_{ij} + t_{fi}(j) + t'_{su} \leq T + w$$
$$t'_{cq} + t_{fo}(i) + D_{ij} + t_{fi}(j) + t_{fo}(j) + D_{jk} + t_{fi}(k) + t'_{su} \leq 2T$$
- Hold
$$t'_{cq} + t_{fo}(i) + d_{ij} + t_{fi}(j) - t'_{hd} \geq \boxed{M = M_0 + w}$$

- To guarantee successful time borrowing, in this paper, time borrowing is allowed between two adjacent timing windows

Timing Slack Conversion

- □ **Flip-flop-based synthesis and placement have considered the extra hold time margin *w* ⇒ we focus on setup slacks**



$$S_{fo}(i) = \frac{T-(D_{ij}+t_{cq}+t_{su})+\epsilon_1}{2} = S_{fi}(j)$$

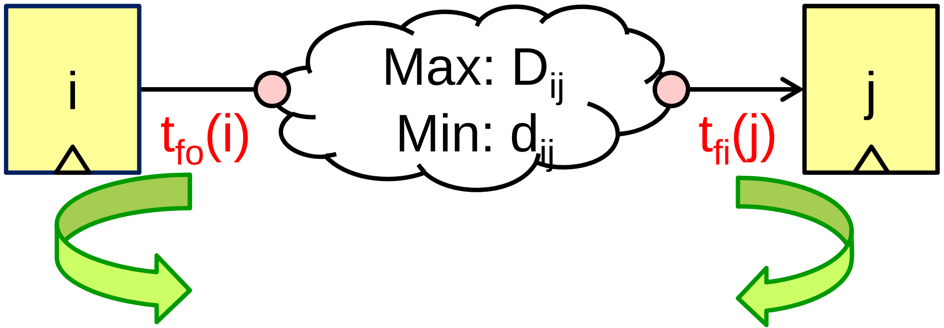
$$\epsilon_1 = (t_{cq}+t_{su}) - (t'_{cq}+t'_{su}),\; D'_{ij} = D_{ij}+t_{cq}+t_{su}-\epsilon_1,$$

$$\boxed{S_{fo}(i) = S_{fi}(j) = \frac{T-D'_{ij}}{2}}$$

- □ **Convert the timing slacks for $t_{fo}(i)$ and $t_{fi}(j)$ obtained by flip-flop-based timing analysis into pulsed-latch-based slacks without time borrowing**
- □ **We equally distribute the whole setup slacks to the latches' fanin and fanout parts**

Slack vs. Wirelength

- **Based on Synopsys' Liberty library, wire delays $t_{fo}(i)$ and $t_{fi}(j)$ can be approximated by piece-wise linear functions with the Manhattan distances $L_{fo}(i)$ and $L_{fi}(j)$**

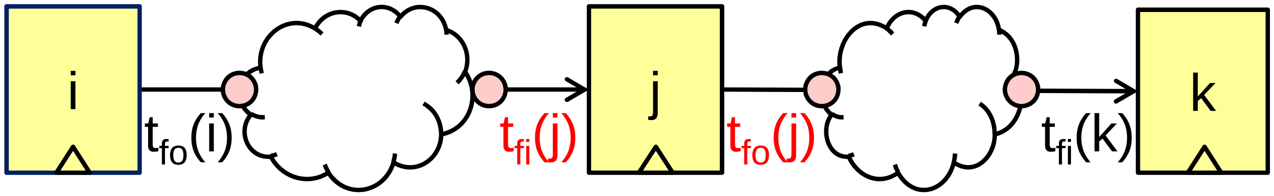


$$t_{fo}(i) = \tau L_{fo}(i) \qquad t_{fi}(j) = \tau L_{fi}(j)$$

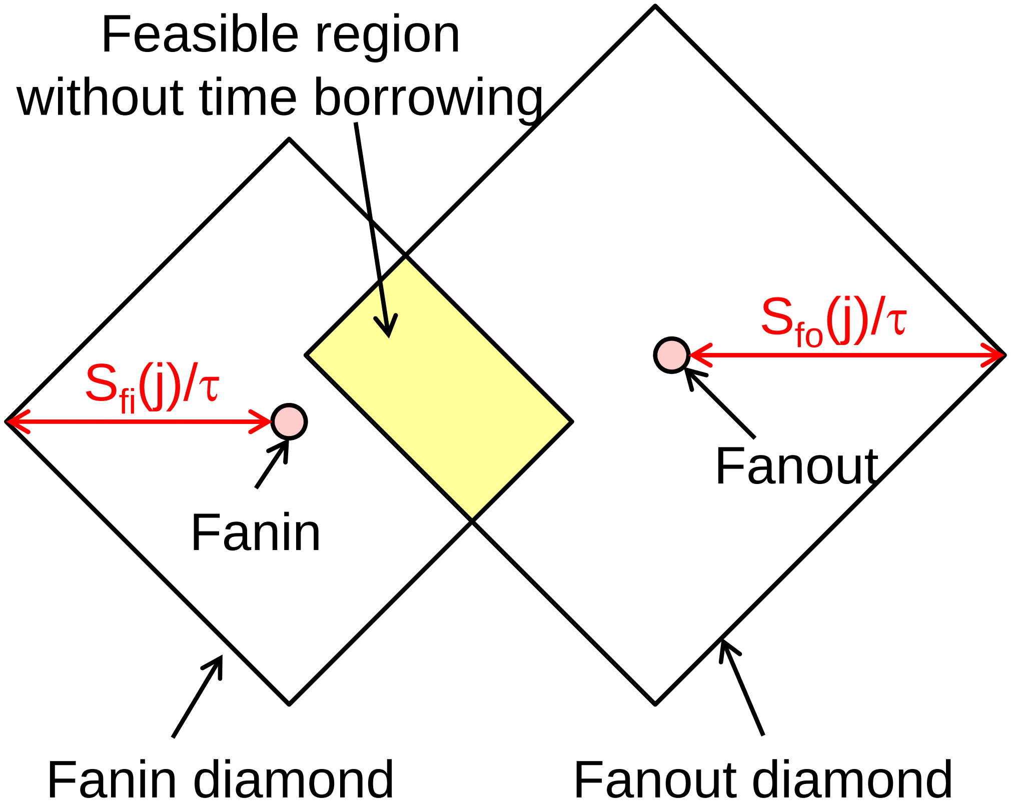
  - τ is calibrated by the delay table of the pulsed-latch library

- **We incorporate time borrowing into the slack value to derive feasible regions**

Feasible Region with Time Borrowing (1/3)



$$t_{fi}(j) = \tau L_{fi}(j) \le S_{fi}(j) \qquad t_{fo}(j) = \tau L_{fo}(j) \le S_{fo}(j)$$

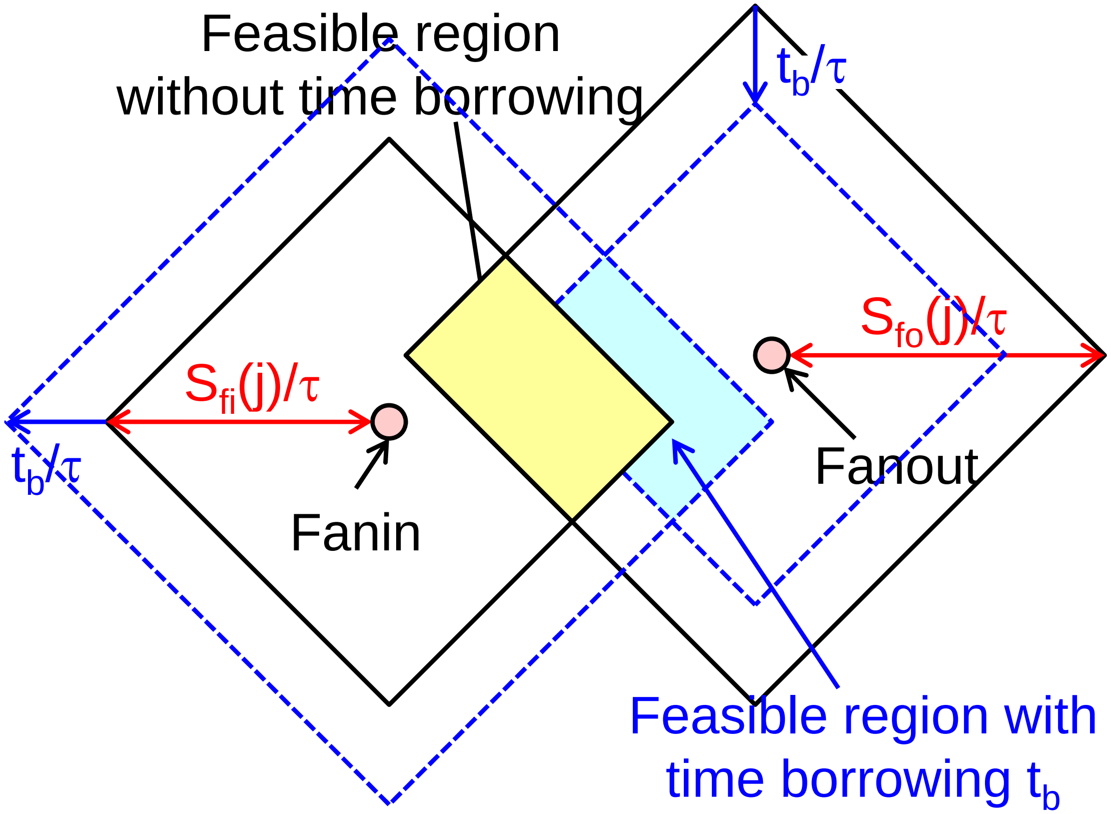


The fanin and fanout setup time slacks define two diamonds centered at the fanin and fanout gates of pulsed-latch *j*.
The overlap area is the initial feasible region without time borrowing.

Feasible Region with Time Borrowing (2/3)

- ***t_b*: the amount of time borrowed from the timing window *j-k* to window *i-j*, $t_b \le w$**

$$t_{fi}(j) = \tau L_{fi}(j) \le S_{fi}(j) \boxed{+ t_b} \qquad t_{fo}(j) = \tau L_{fo}(j) \le S_{fo}(j) \boxed{- t_b}$$



Feasible region without time borrowing

t_b/τ

$S_{fi}(j)/\tau$

t_b/τ

Fanin

$S_{fo}(j)/\tau$

Fanout

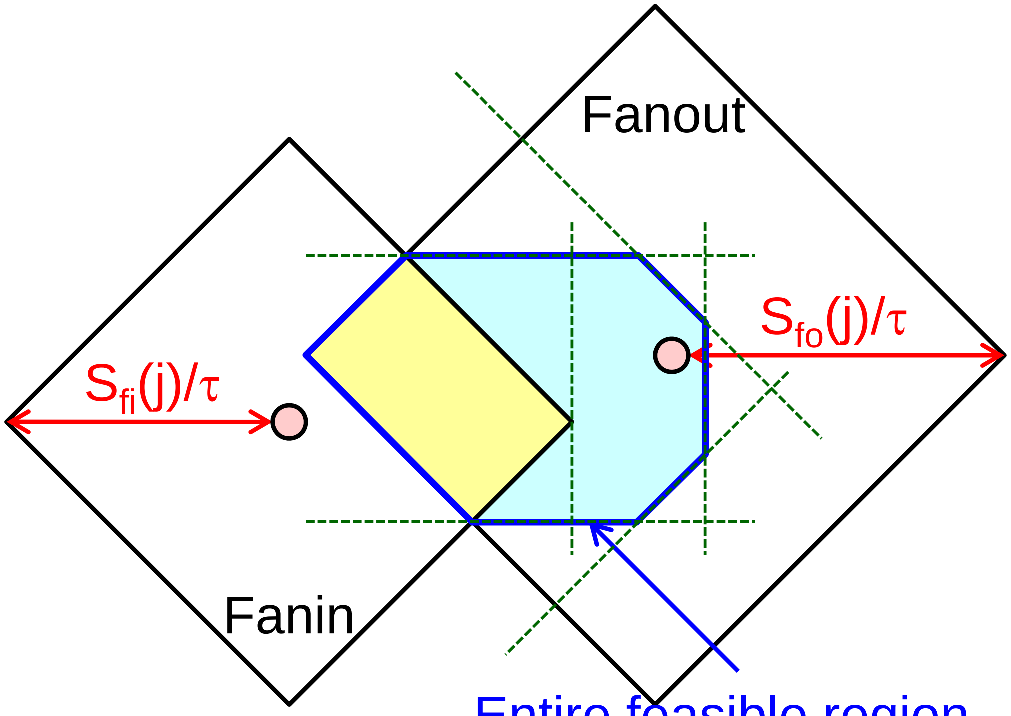
Feasible region with time borrowing t_b

When we borrow some time t_b, the fanin diamond is expanded by t_b/τ, while the fanout diamond is shrunk by t_b/τ.

The overlap area slides horizontally or vertically.

Feasible Region with Time Borrowing (3/3)

- ***t_b*: the amount of time borrowed from the timing window *j-k* to window *i-j*, *$t_b \leq w$***



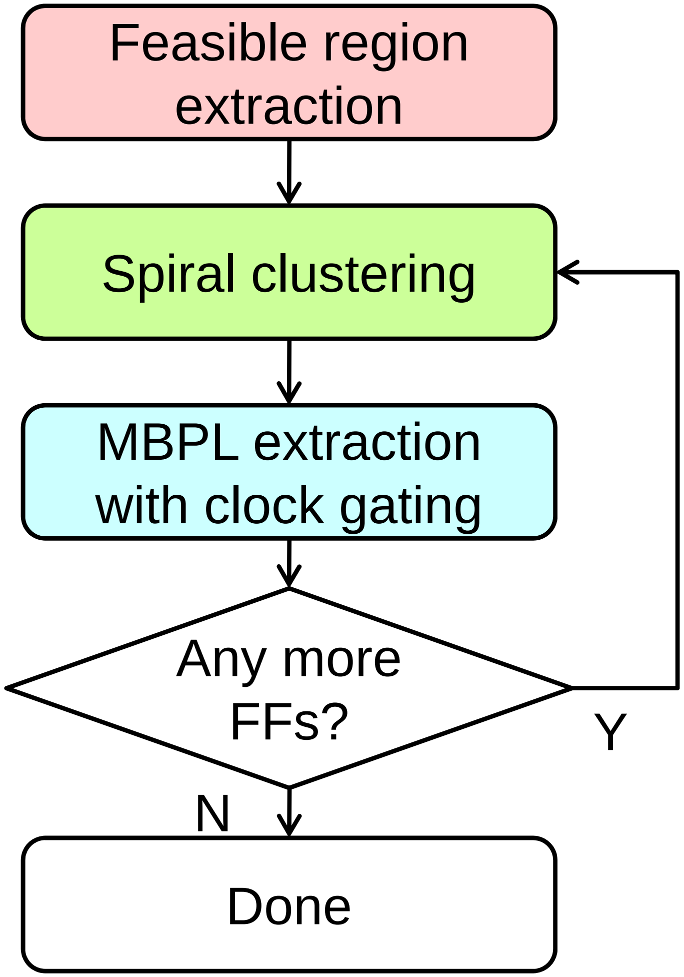
Entire feasible region with time borrowing

When we keep borrowing, the fanin or fanout diamond would reach the middle lines of the boundaries of fanin/fanout diamonds, and the overlap area are truncated.
The entire feasible region is irregular.
In the worst case, the feasible region could be an octagon.

Outline

- Introduction
- Preliminaries
- Feasible region
- **Algorithm**
- Experimental results
- Conclusion

Post-Placement Pulsed-Latch Replacement

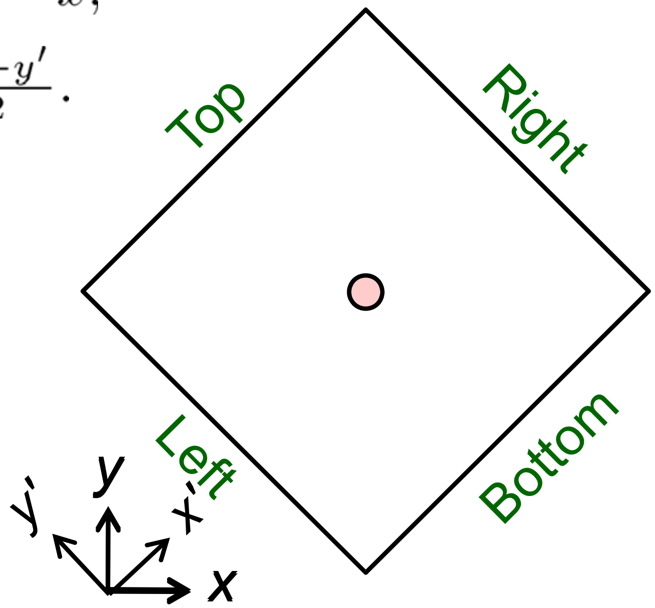


1. Extract feasible regions and represent them by four interval graphs
2. Use spiral clustering to form multi-bit pulsed-latches
3. Meanwhile, consider clock gating during MBPL extraction
4. Relocate the newly formed multi-bit pulsed-latches
5. Repeat steps 2–4 until all latches are investigated

Coordinate Transformation

- **To facilitate our feasible region extraction, we adopt a simple and fast coordinate transformation**
  - The fanin/fanout diamonds in Cartesian coordinate system *C* become squares in *C*′, obtained by rotating by 45-degree.
  - $x' = y + x,\ y' = y - x;$

    $x = \frac{x'-y'}{2},\ y = \frac{x'+y'}{2}.$

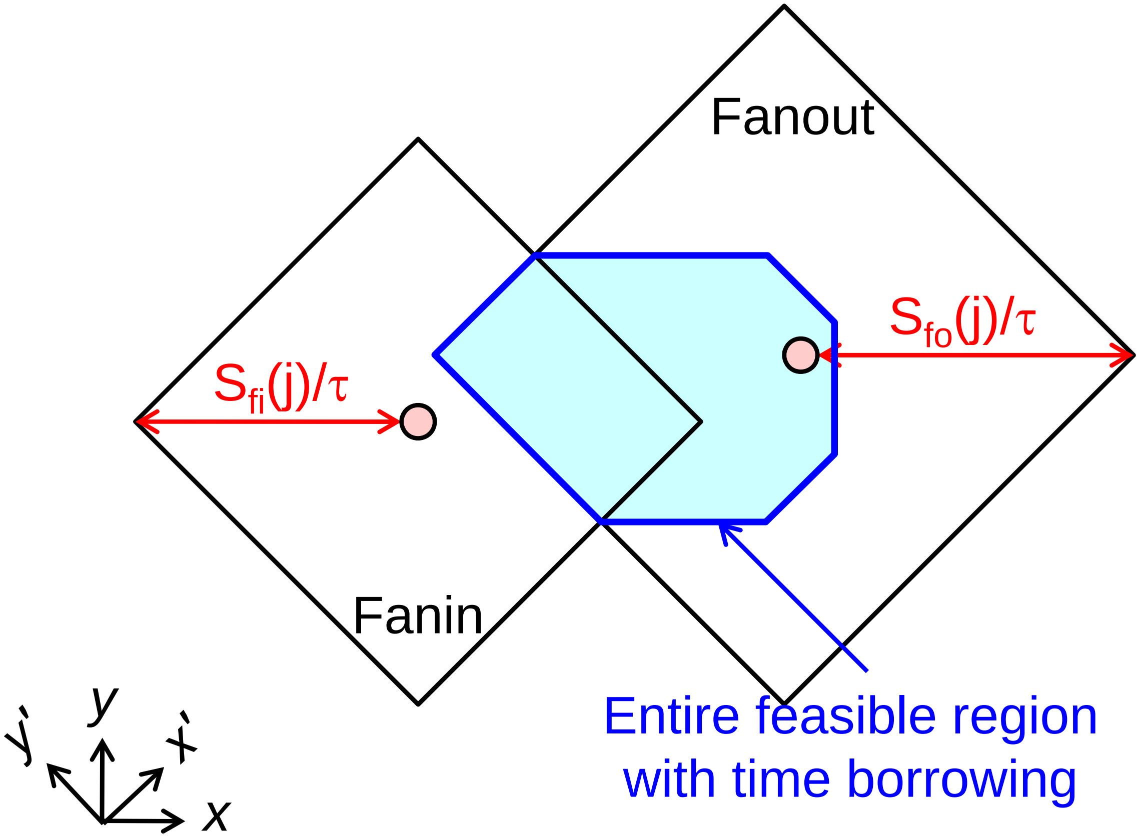


- **Define the four boundaries of a fanin/fanout diamond as right, bottom, left, and top boundaries.**

Chang, et al. INTEGRA: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs. ISPD -11

Feasible Region Extraction

- **The fanin diamond expands, while the fanout diamond shrinks with time borrowing**
- **The entire feasible region is irregular. In the worst case, the feasible region could be an octagon**

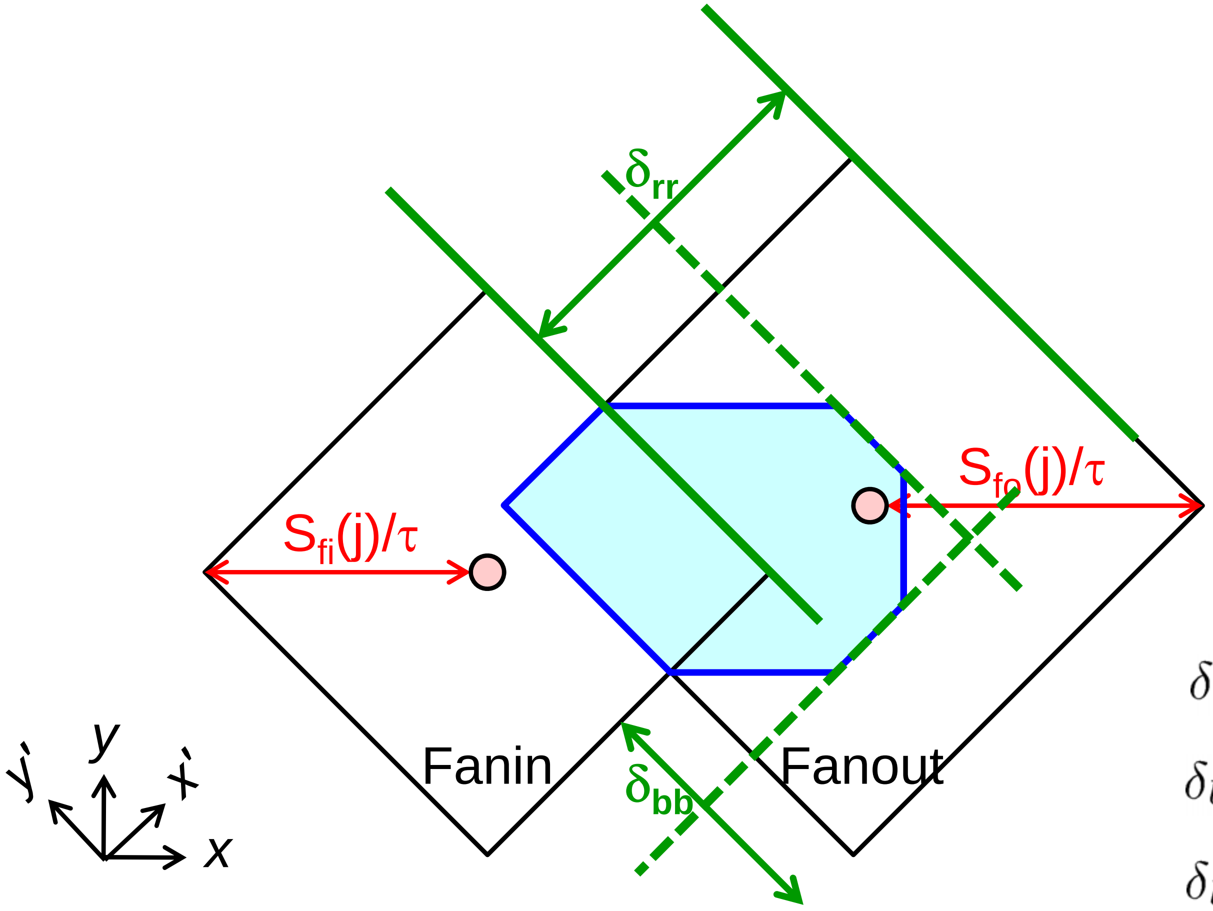


Entire feasible region with time borrowing

How to extract the feasible region?

Fence Finding (1/2)

- If some fanout boundary is outer of the corresponding fanin one, there is a fence constraining the feasible region sliding



$$\delta_{tt} = \delta_{ll} = 0$$

$$\delta_{rr}, \delta_{bb} > 0$$

$$\delta_{rr}(j) = \max\left(0, x'_{r,fo}(j) - x'_{r,fi}(j)\right),$$

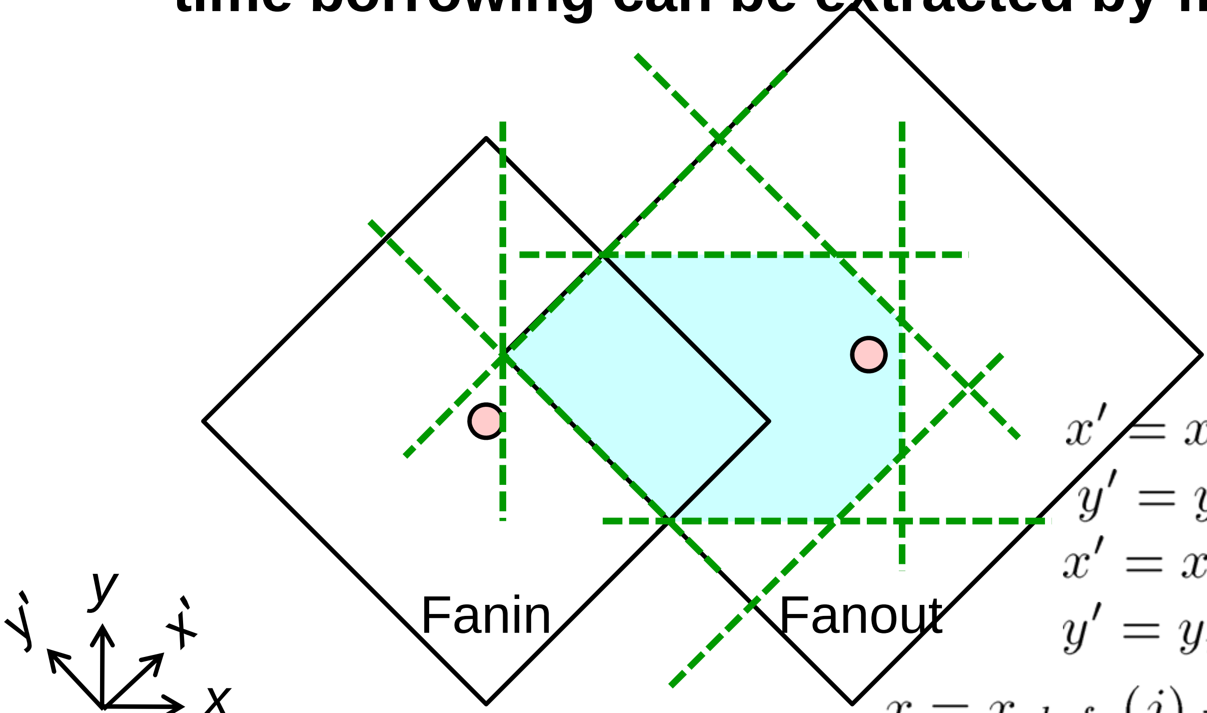
$$\delta_{bb}(j) = \max\left(0, y'_{b,fi}(j) - y'_{b,fo}(j)\right),$$

$$\delta_{ll}(j) = \max\left(0, x'_{l,fi}(j) - x'_{l,fo}(j)\right),$$

$$\delta_{tt}(j) = \max\left(0, y'_{t,fo}(j) - y'_{t,fi}(j)\right).$$

Fence Finding (2/2)

- **The fences are determined by**
  - The pulse width
  - The differences between boundaries of fanin/fanout diamonds
- **Given the initial feasible region, the entire feasible region with time borrowing can be extracted by finding eight fences.**

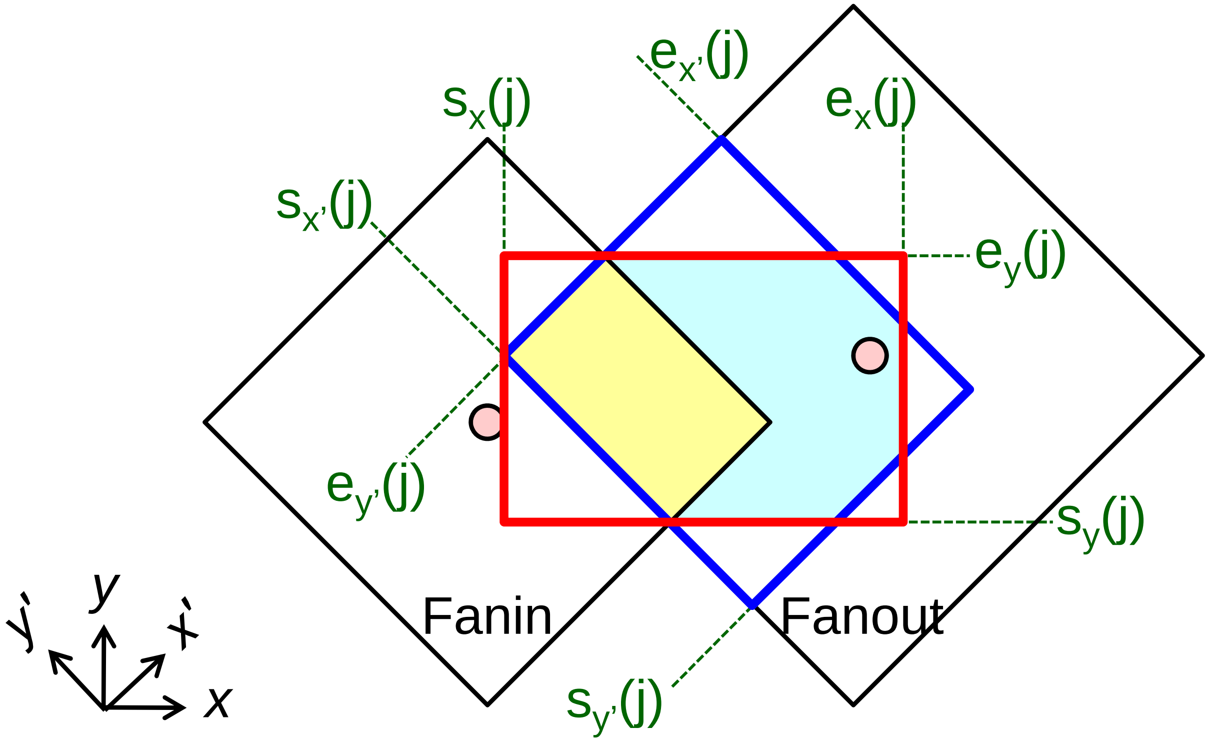


$$\begin{aligned}
x' &= x'_{r,fi}(j) + \min(w, \delta_{rr}(j)/2) = e_{x'}(j),\\
y' &= y'_{b,fi}(j) - \min(w, \delta_{bb}(j)/2) = s_{y'}(j),\\
x' &= x'_{l,fi}(j) - \min(w, \delta_{ll}(j)/2) = s_{x'}(j),\\
y' &= y'_{t,fi}(j) + \min(w, \delta_{tt}(j)/2) = e_{y'}(j).
\end{aligned}$$

$$\begin{aligned}
x &= x_{rb,fr}(j) + \min(w, \delta_{rr}(j)/2, \delta_{bb}(j)/2) = e_{x}(j),\\
y &= y_{lb,fr}(j) - \min(w, \delta_{ll}(j)/2, \delta_{bb}(j)/2) = s_{y}(j),\\
x &= x_{lt,fr}(j) - \min(w, \delta_{ll}(j)/2, \delta_{tt}(j)/2) = s_{x}(j),\\
y &= y_{rt,fr}(j) + \min(w, \delta_{rr}(j)/2, \delta_{tt}(j)/2) = e_{y}(j).
\end{aligned}$$

Four Interval Graphs

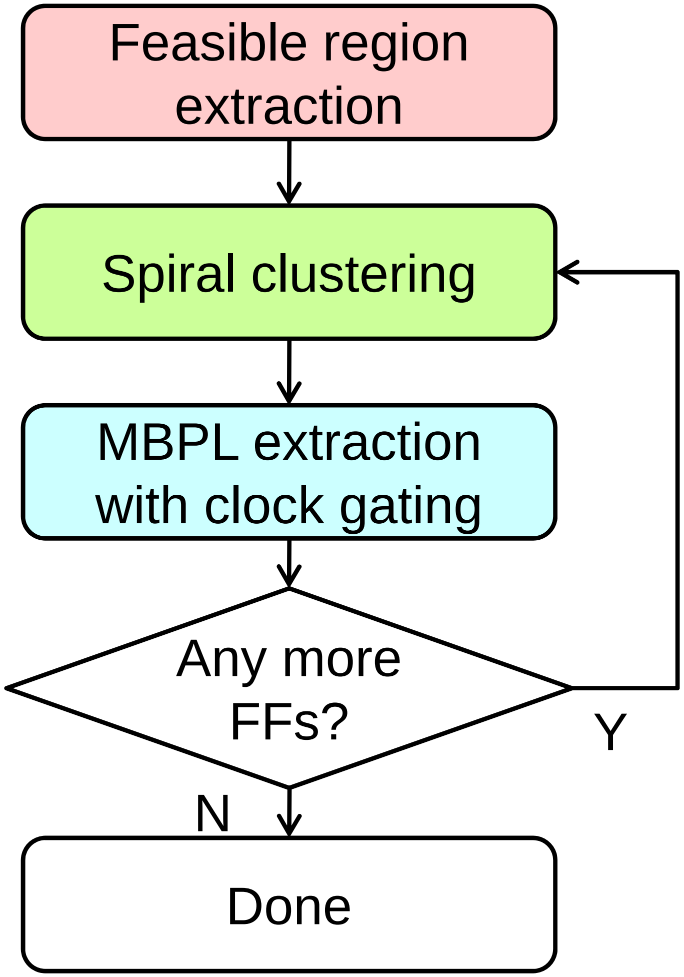
- □ **Using these eight fences, we can handle any irregular feasible region.**
- □ **The projection of all feasible regions to *x′*-, *y′*-, *x*-, and *y*-axes form four interval graphs.**



Sequences *X′*, *Y′*, *X*, *Y* to record the starting and ending coordinates of *x′*, *y′*, *x*, and *y* intervals in ascending order.

The feasible regions of 2 pulsed-latches overlap iff their feasible regions overlap on these four interval graphs.

Post-Placement Pulsed-Latch Replacement



1. Extract feasible regions and represent them by four interval graphs.
2. Use spiral clustering to form multi-bit pulsed-latches
3. Meanwhile, consider clock gating during MBPL extraction
4. Relocate the newly formed multi-bit pulsed-latches.
5. Repeat steps 2–4 until all flip-flops are investigated

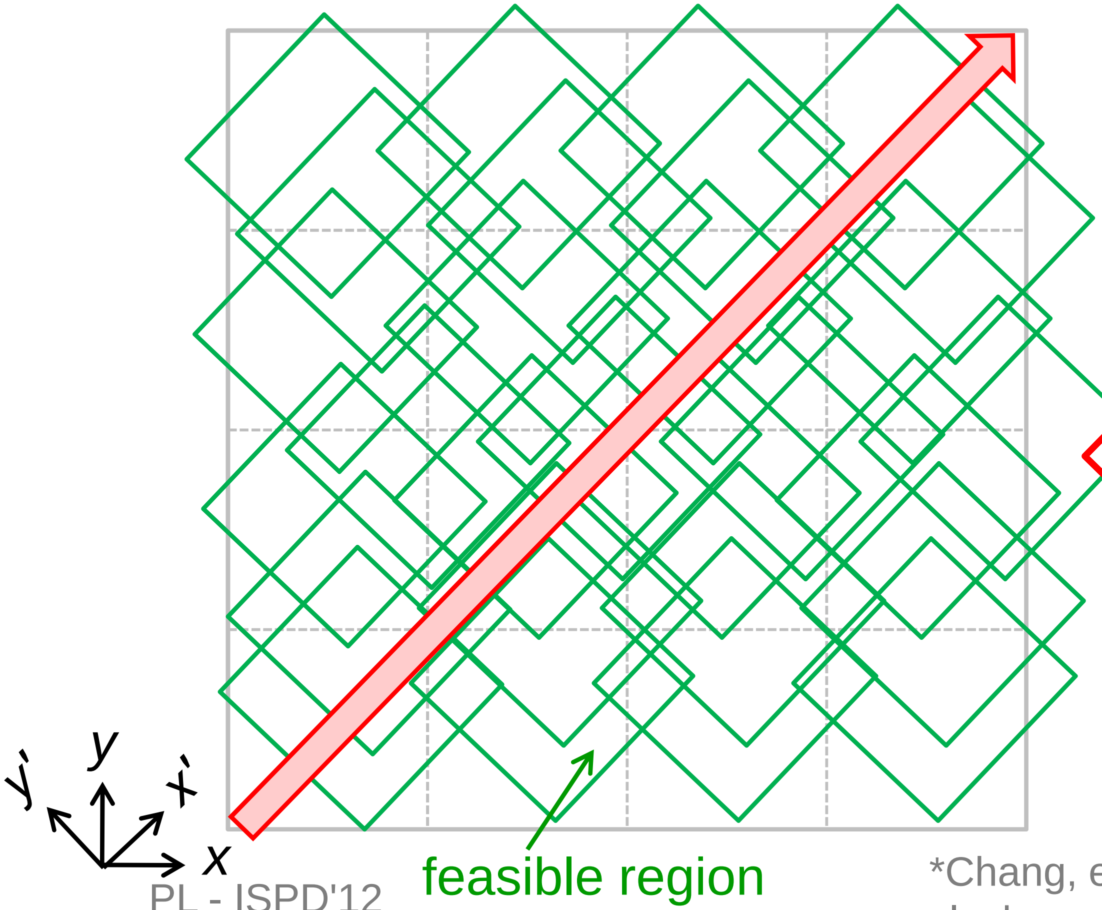
Spiral Clustering and MBPL Extraction

- **Spiral clustering**
  - Find maximal cliques in the intersection graph of all feasible regions
  - In physical perspective
- **MBPL extraction with clock gating**
  - Extract subset with similar clock gating patterns from the found maximal clique to form a multi-bit pulsed latch
  - In logical perspective

One Way Clustering vs. Spiral Clustering

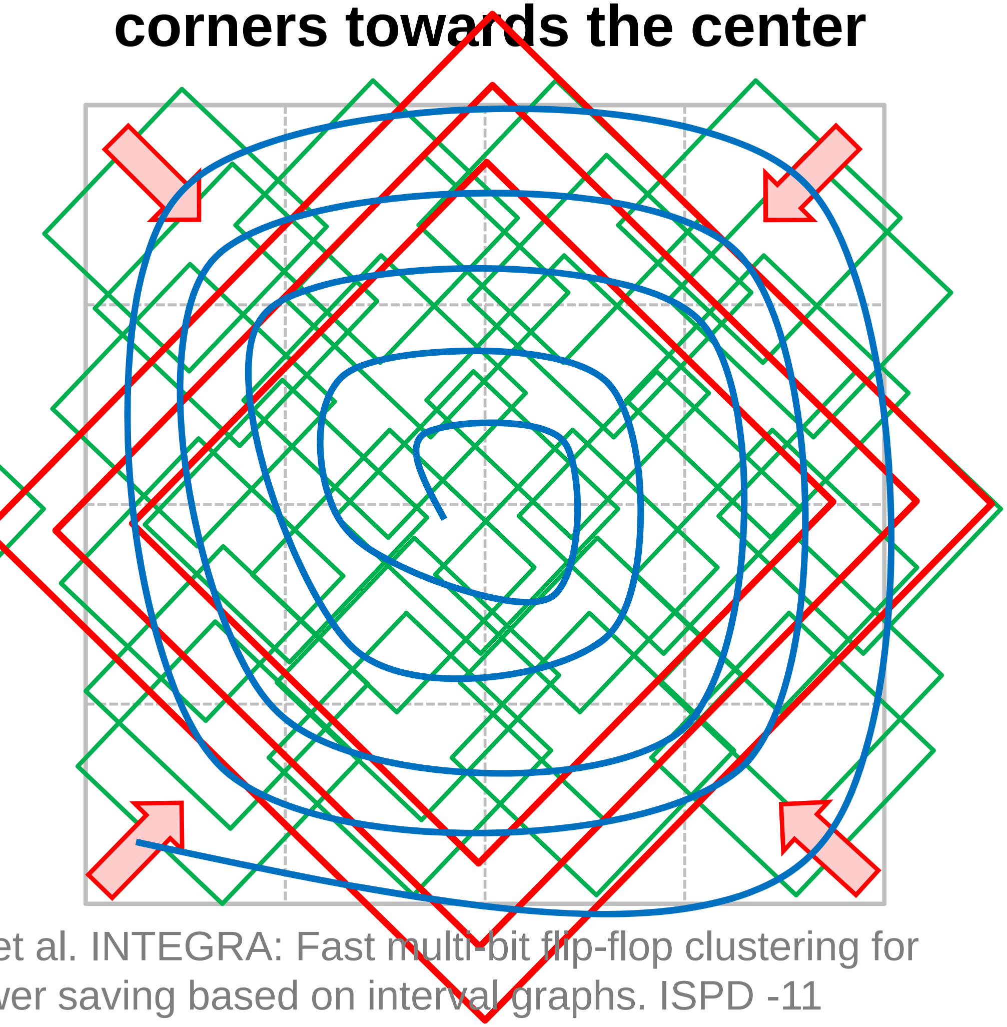
One way clustering*

- □ **Cluster along *x′* axis**
- □ **Orphans around the end of *X′***



Spiral clustering

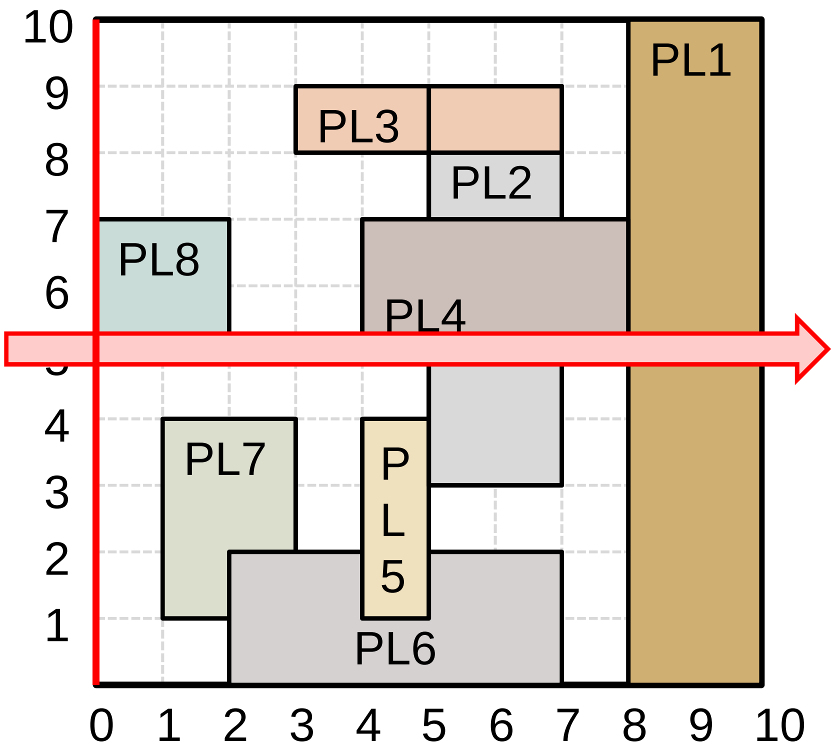
- □ **Find cliques from four corners towards the center**

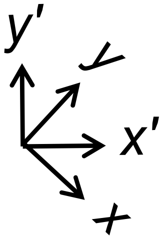


*Chang, et al. INTEGRA: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs. ISPD -11

One Way Clustering vs. Spiral Clustering

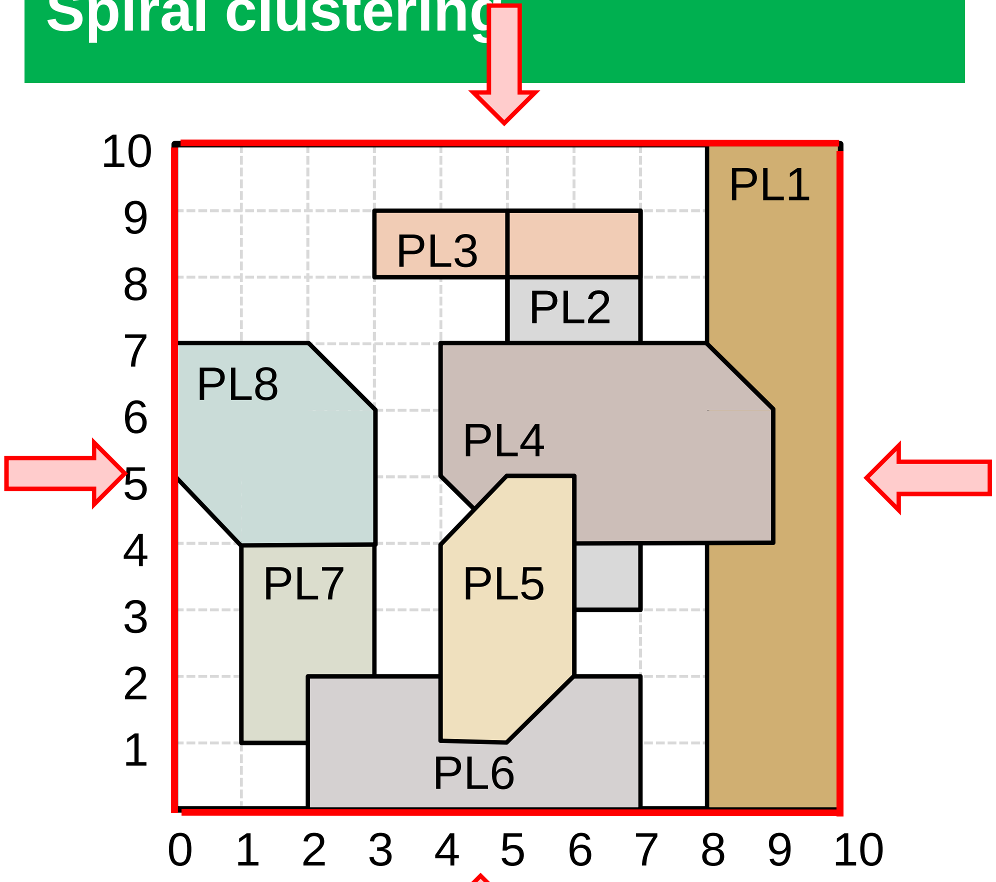
One way clustering*





{8} {6, 7} {2, 5} {3} {1, 4}

Spiral clustering

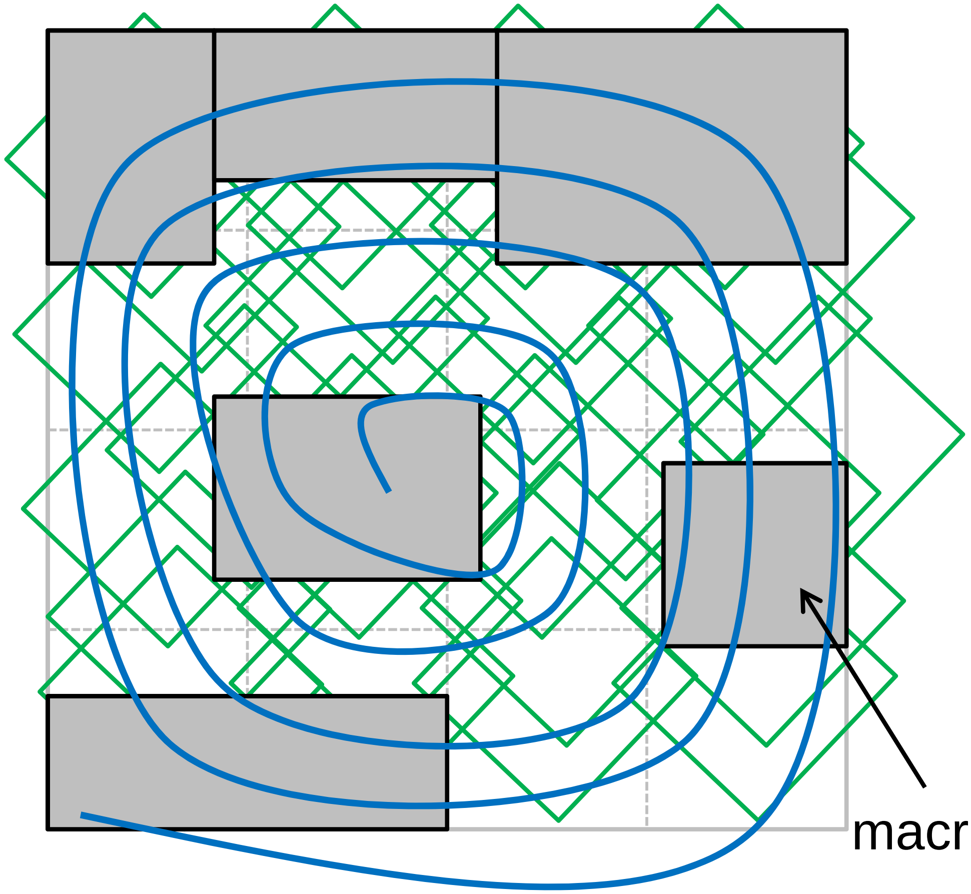


{7, 8} {5, 6} {1, 4} {2, 3}

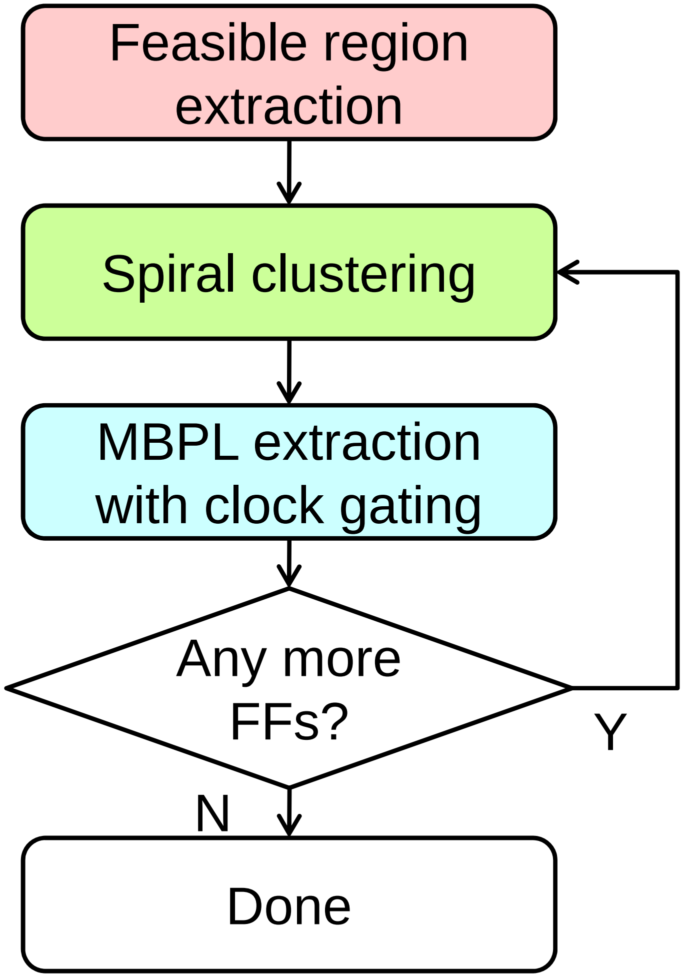
*Chang, et al. INTEGRA: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs. ISPD -11

Rectilinear Layout

- **Spiral clustering groups from corners**
  - Suitable for rectilinearly shaped layout with many macros



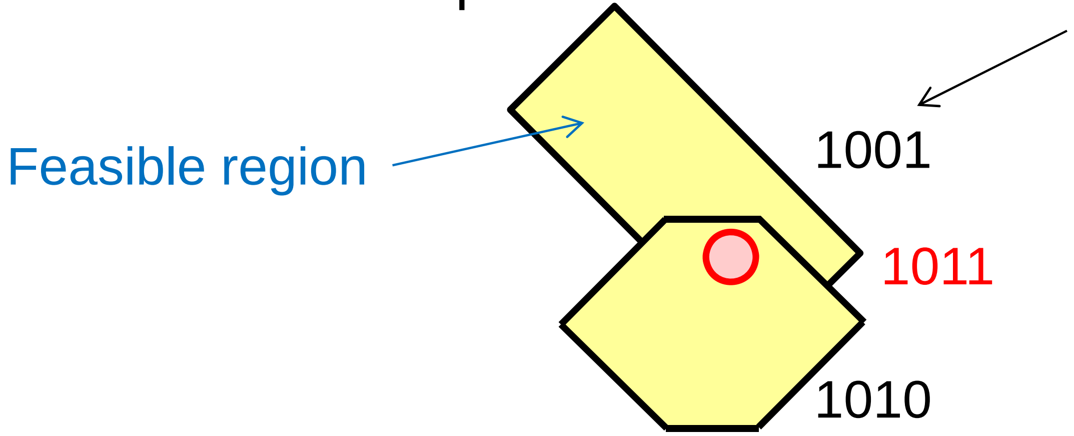
Post-Placement Pulsed-Latch Replacement



1. Extract feasible regions and represent them by four interval graphs.
2. Use spiral clustering to form multi-bit pulsed-latches
3. Meanwhile, consider clock gating during MBPL extraction
4. Relocate the newly formed multi-bit pulsed-latches.
5. Repeat steps 2–4 until all flip-flops are investigated

Clock Gating Is Important!

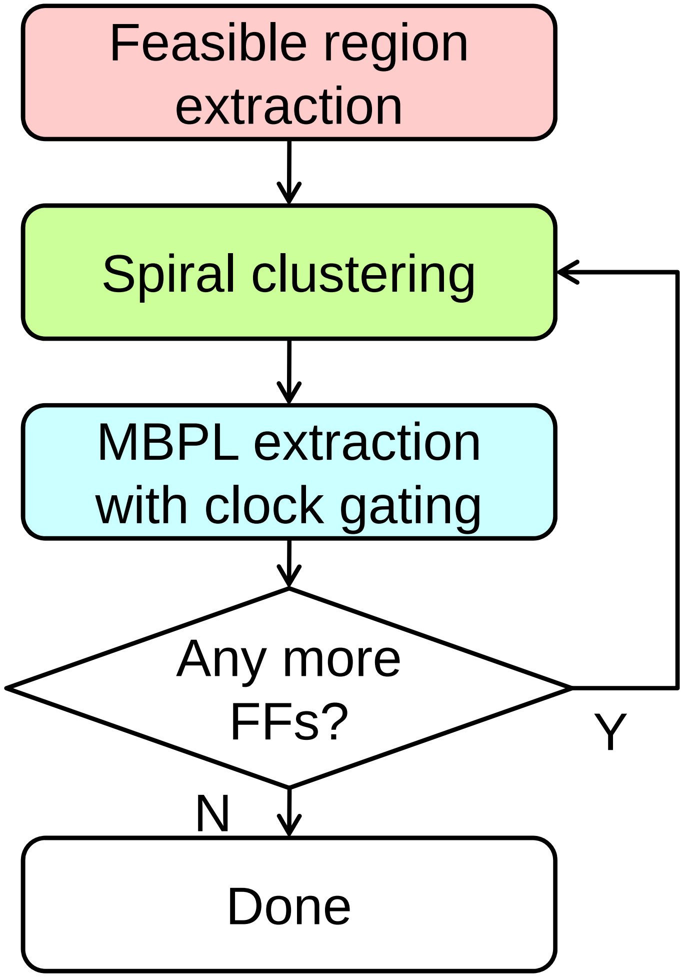
- **Since the latches inside one MBPL cell share the pulse clock, their clock gating functions are logic ORed together.**
- **If we merge pulsed-latches with very different clock gating patterns, we may not reduce power consumption.**
  - Effective power ratio = library * pattern
  - E.g., library: 0.74, pattern: 1.5 => effective power ratio = 1.11
  - Worse than separate PLs



Bit Number	**Normalized power**
1	1.00
2	1.48

- **To reduce power, our strategy is to extract a subset of feasible bit number and with minimum effective power ratio from a found maximal clique.**

Post-Placement Pulsed-Latch Replacement

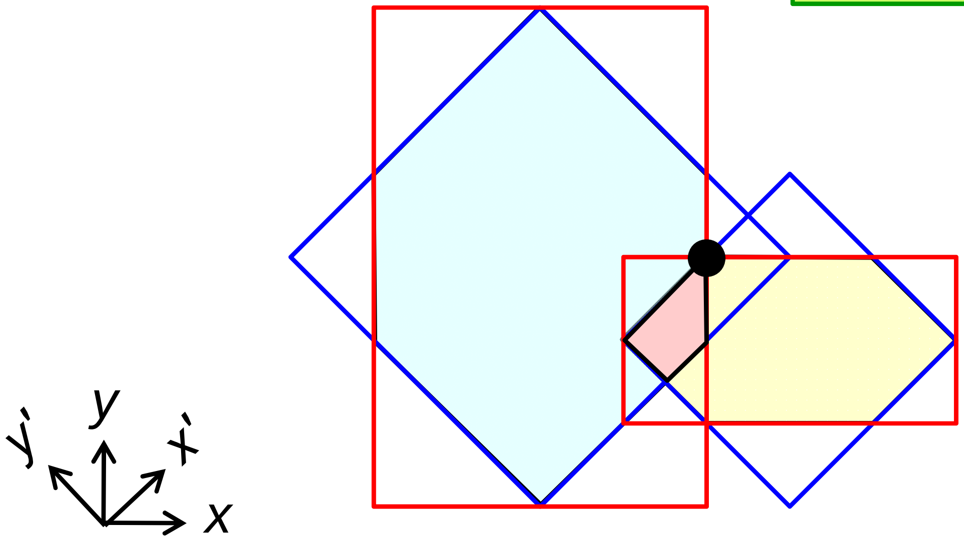


1. Extract feasible regions and represent them by four interval graphs.
2. Use spiral clustering to form multi-bit pulsed-latches
3. Meanwhile, consider clock gating during MBPL extraction
4. Relocate the newly formed multi-bit pulsed-latches.
5. Repeat steps 2–4 until all flip-flops are investigated

MBPL Relocation

1. **For a formed multi-bit pulsed latch, find the point in the feasible region with minimum wirelength**
2. **Legalize it**

Minimum wirelength region



Outline

Introduction

Preliminaries

Feasible region

Algorithm

Experimental results

Conclusion

Settings

- **We implemented our algorithm in the C programming language and executed the program on a platform with an Intel Xeon 3.8 GHz CPU and with 16 GB memory under Ubuntu 10.04 OS.**
- **1-/2-/4-/8-bit MBPL cells based on 55-nm technology**
  - *w* = 100 ps

Bit Number	Normalized power	Normalized area
1	1.00	1.00
2	1.48	1.92
4	2.45	3.85
8	4.60	7.58

- **Benchmark**

Circuit	#FFs	#Bins	#Grids	Avg. activity
Industry1	120	6×6	600×600	0.25
Industry2	120	6×6	600×600	0.13
Industry3	60,000	100×300	2,000×3,000	0.69
Industry4	5,524	100×200	2,000×2,000	0.44
Industry5	953	30×160	600×1,600	0.25

  - avg. activity is the average active rate of clock gating functions.

One Way Clustering vs. Spiral Clustering

- **Focus on power reduction contributed from the MBPL library during spiral clustering**

Circuit	One Way Clustering*				Spiral Clustering with Time Borrowing w=100ps w/o Clock Gating			
	Power Ratio	Pattern-Aware Power Ratio	#Sinks (1/2/4/8-bit PLs)	Runtime (s)	Power Ratio	Pattern-Aware Power Ratio	#Sinks (1/2/4/8-bit PLs)	Runtime (s)
Industry1	74.93%	130.67%	62 (18/37/7/0)	< 0.01	69.34%	140.38%	49 (4/32/13/0)	< 0.01
Industry2	75.78%	101.22%	64 (20/38/6/0)	< 0.01	72.36%	104.30%	56 (14/31/11/0)	< 0.01
Industry3	57.54%	79.53%	7,558 (10/35/46/7,467)	3.36	57.50%	79.49%	7,500 (0/0/0/7,500)	3.07
Industry4	62.98%	96.61%	1,520 (52/432/920/116)	0.41	60.84%	99.33%	1,233 (16/182/784/251)	0.39
Industry5	65.36%	113.79%	311 (27/123/152/9)	0.04	62.33%	121.02%	246 (9/62/145/30)	0.05
Avg.	67.32%	104.36%	35.55%	-	64.47%	108.90%	29.63%	-

*Chang, et al., "INTEGRA: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs," *ISPD* 2011

w = 150 ps vs. w = 200 ps

- **If the pulse width increases, the power saving can be further improved.**

Circuit	Spiral Clustering with Time Borrowing w = 150 ps w/o Clock Gating				Spiral Clustering with Time Borrowing w = 200 ps w/o Clock Gating			
	Power Ratio	Pattern-Aware Power Ratio	#Sinks (1/2/4/8-bit PLs)	Runtime (s)	Power Ratio	Pattern-Aware Power Ratio	#Sinks (1/2/4/8-bit PLs)	Runtime (s)
Industry1	68.07%	142.54%	46 (4/26/16/0)	< 0.01	67.64%	144.35%	45 (4/24/17/0)	< 0.01
Industry2	70.22%	101.35%	51 (10/27/14/0)	< 0.01	69.79%	103.56%	50 (10/25/15/0)	< 0.01
Industry3	57.50%	79.53%	7,500 (0/0/0/7,500)	3.20	57.50%	79.47%	7,500 (0/0/0/7,500)	3.23
Industry4	60.52%	99.68%	1,184 (14/157/727/286)	0.41	60.46%	99.95%	1,170 (14/163/690/303)	0.40
Industry5	62.00%	121.95%	239 (7/55/145/32)	0.05	62.12%	122.86%	240 (7/63/135/35)	0.04
Avg.	63.66%	109.01%	27.97%	-	63.50%	110.04%	27.61%	-

Without vs. With Clock Gating (w=100ps)

- Consider clock gating during spiral clustering

Circuit	Spiral Clustering with Time Borrowing w = 100 ps w/o Clock Gating				Spiral Clustering with Time Borrowing w = 100ps w/ Clock Gating			
	Power Ratio	Pattern-Aware Power Ratio	#Sinks (1/2/4/8-bit PLs)	Runtime (s)	Power Ratio	Pattern-Aware Power Ratio	#Sinks (1/2/4/8-bit PLs)	Runtime (s)
Industry1	69.34%	140.38%	49 (4/32/13/0)	< 0.01	95.68%	95.68%	110 (104/4/2/0)	< 0.01
Industry2	72.36%	104.30%	56 (14/31/11/0)	< 0.01	78.38%	78.38%	70 (32/32/6/0)	< 0.01
Industry3	57.50%	79.49%	7,500 (0/0/0/7,500)	3.07	63.59%	68.78%	15,033 (8,578/25/17/6,413)	5.20
Industry4	60.84%	99.33%	1,233 (16/182/784/251)	0.39	73.33%	73.99%	2,633 (1,584/328/621/100)	0.45
Industry5	62.33%	121.02%	246 (9/62/145/30)	0.05	77.46%	77.59%	535 (337/102/89/7)	0.05
Avg.	64.47%	108.90%	29.63%	-	77.69%	78.88%	55.77%	-

Outline

- Introduction
- Preliminaries
- Feasible region
- Algorithm
- Experimental results
- **Conclusion**

Conclusion

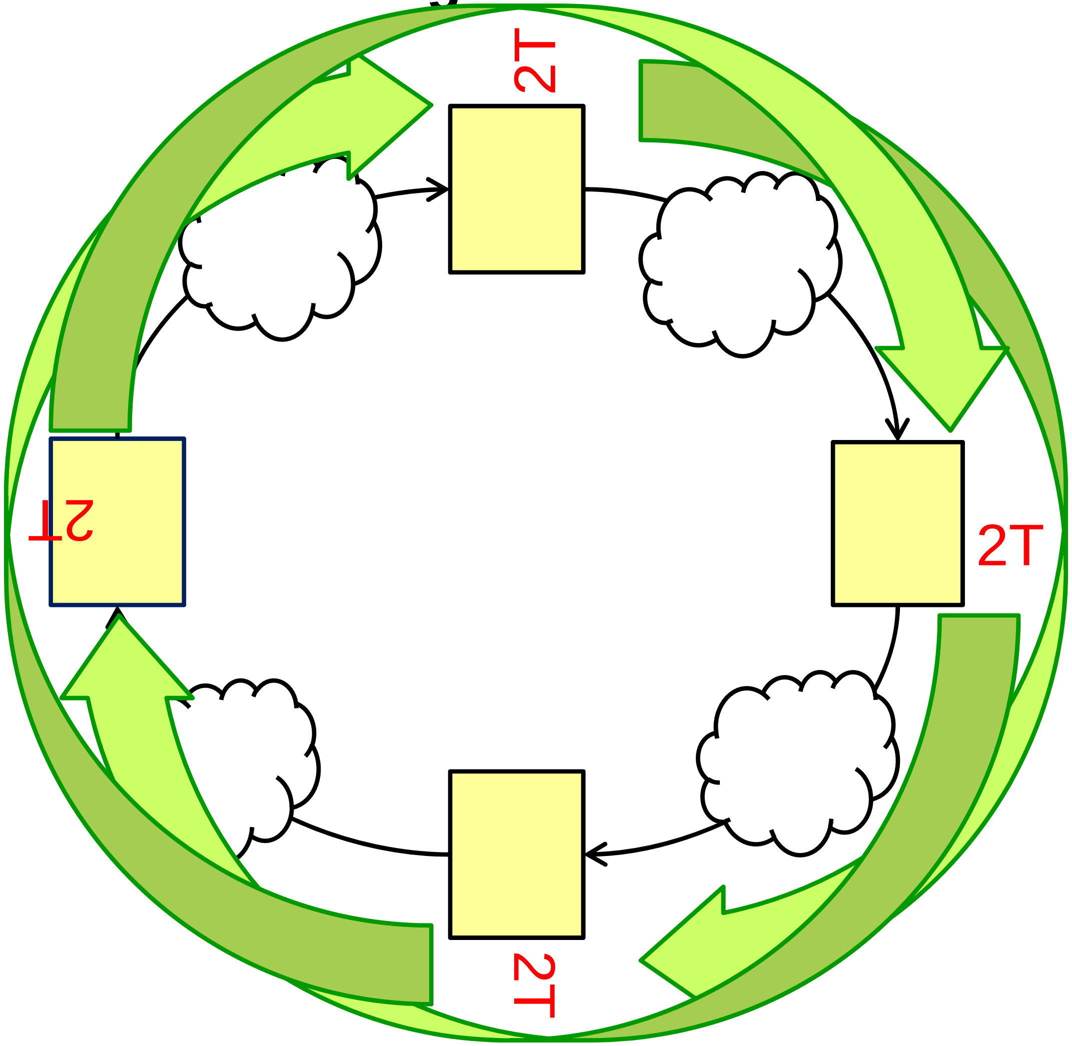
- **Derive timing properties**
  - Setup/hold time constraints with time borrowing
  - Use intrinsic time borrowing: safer than skew scheduling, pulse width allocation and retiming
- **Reveal irregular feasible regions**
  - Maybe an octagon
  - New representation: two pairs of interval graphs
- **Propose spiral clustering**
  - Better clustering results than one way clustering
  - Suitable for rectilinearly shaped layout
- **Consider clock gating**
  - Effective power reduction
- **Our results show that with time borrowing, spiral clustering, and clock gating consideration, we can achieve very power efficient results**

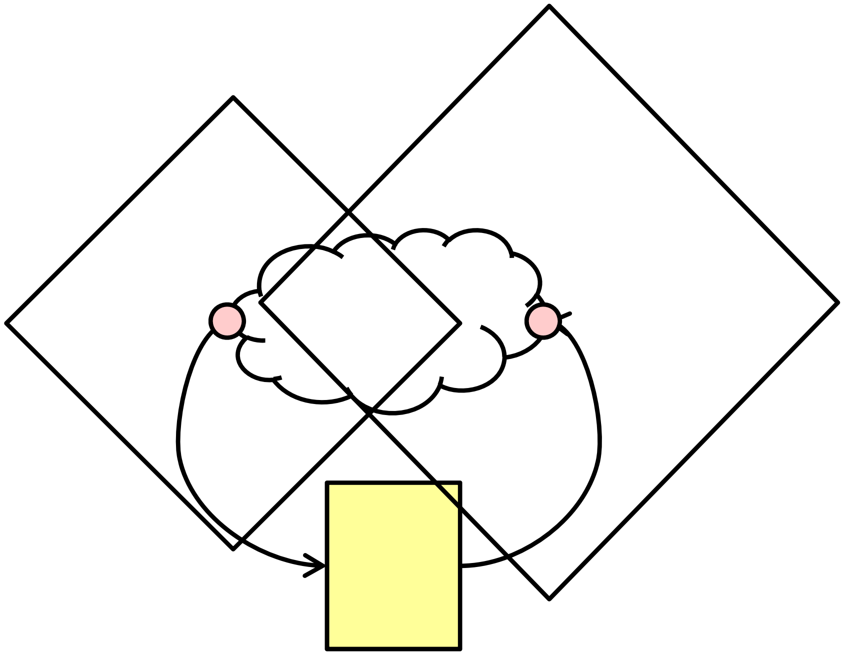
Thank You!

Contact info:
Iris Hui-Ru Jiang
huiru.jiang@gmail.com

How about Loops?

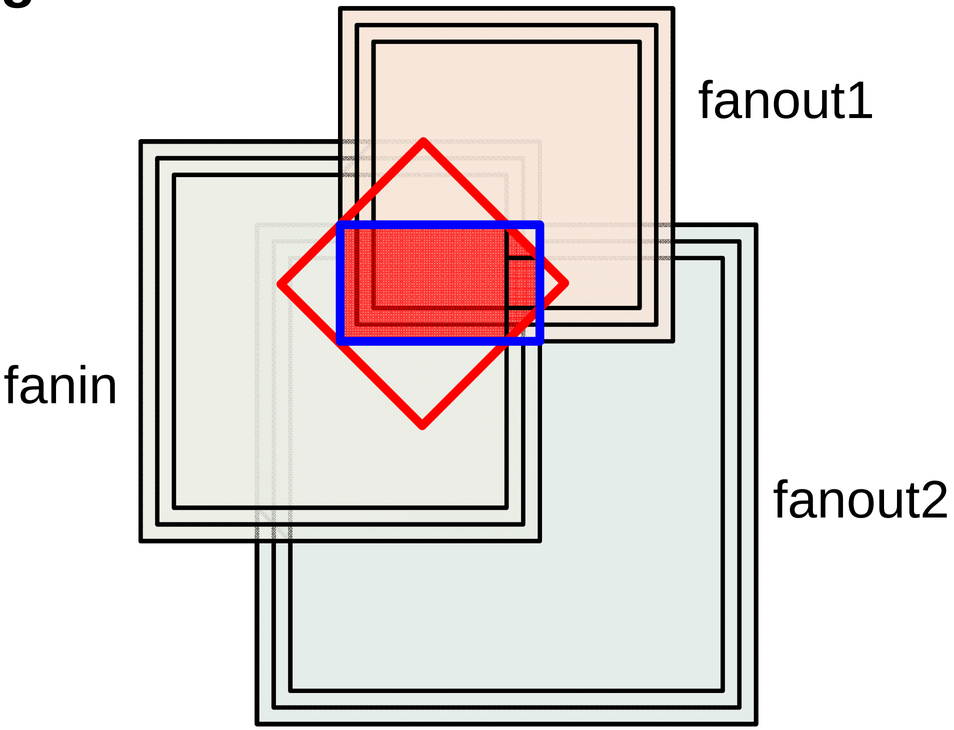
- To guarantee successful time borrowing, in this paper, time borrowing is allowed between two adjacent timing windows





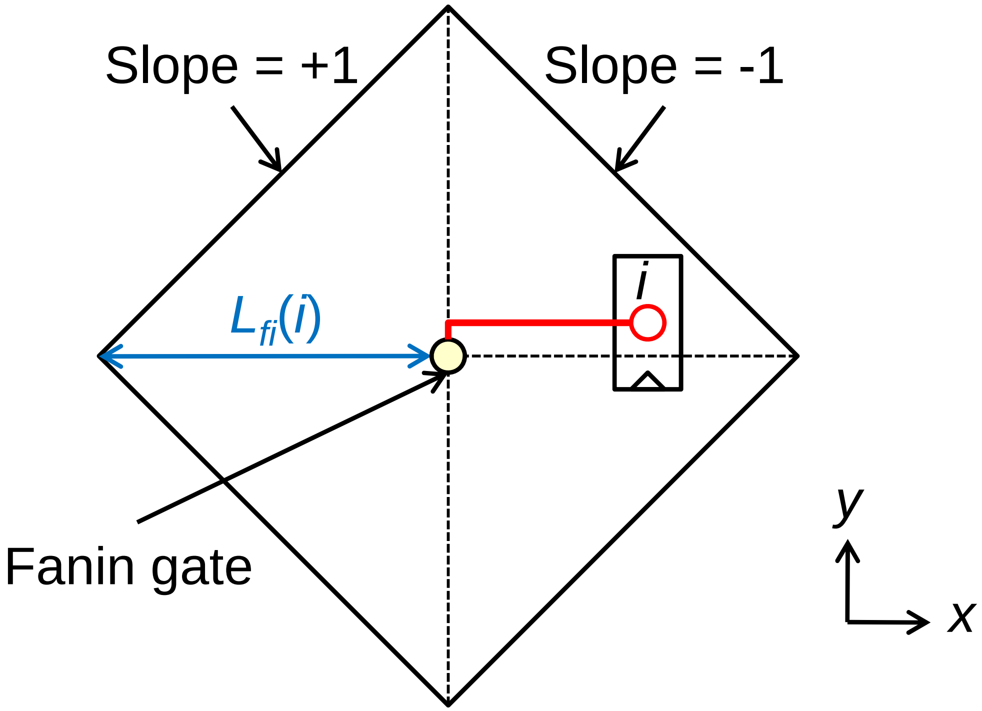
How about Multiple Fanouts?

- Consider individually
- Combine together

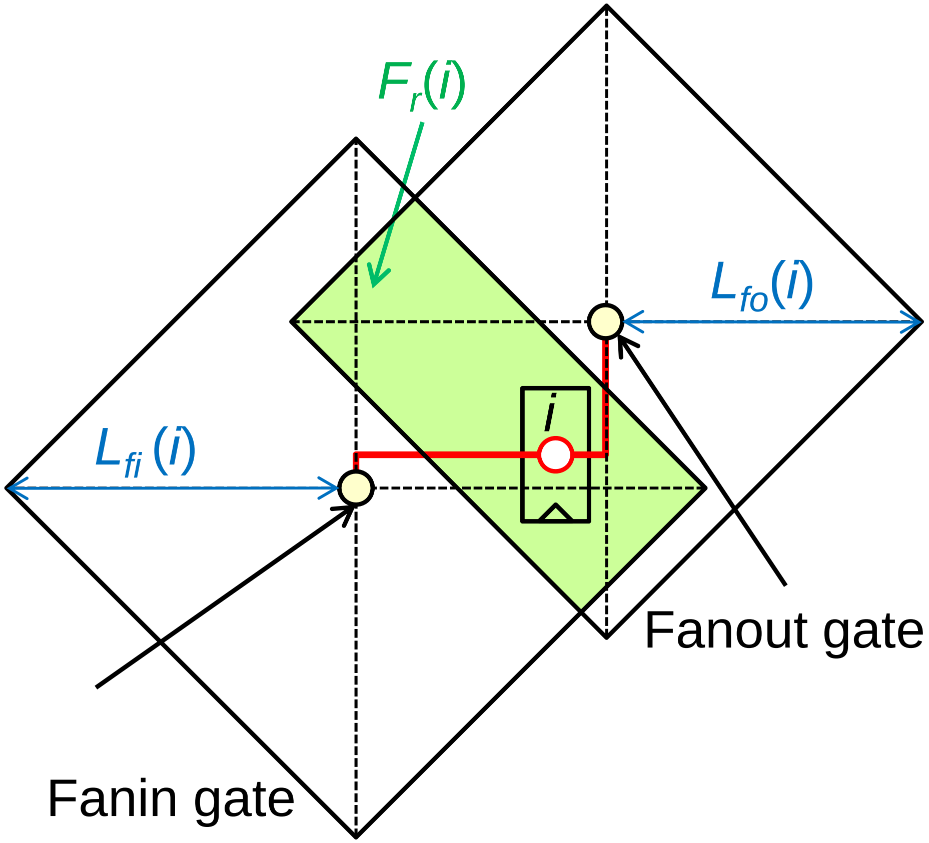


What We Have Already

Fain slack



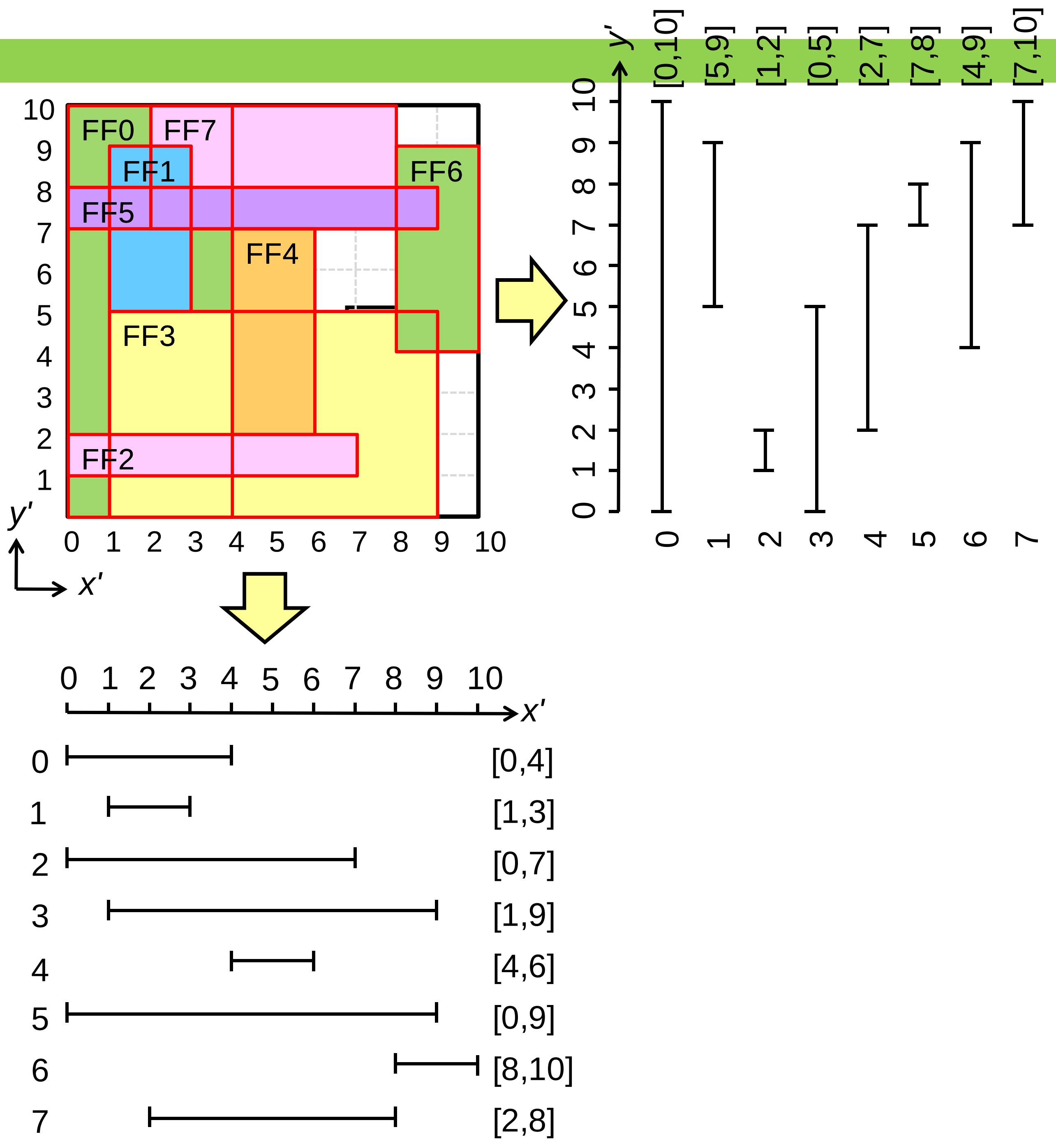
Feasible region



Efficient transformation

Representation

- □ **Interval graphs**
- □ **Sequences**



Efficient data structure