國立交通大學電子工程學系

INTEGRA: Fast Multi-Bit Flip-Flop Clustering for Clock Power Saving Based on Interval Graphs

IRIS HUI-RU JIANG

CHIH-LONG CHANG

YU-MING YANG

EVAN YU-WEN TSAI

LANCER SHENG-FONG CHEN

NCTU

FARADAY

IRIS Lab

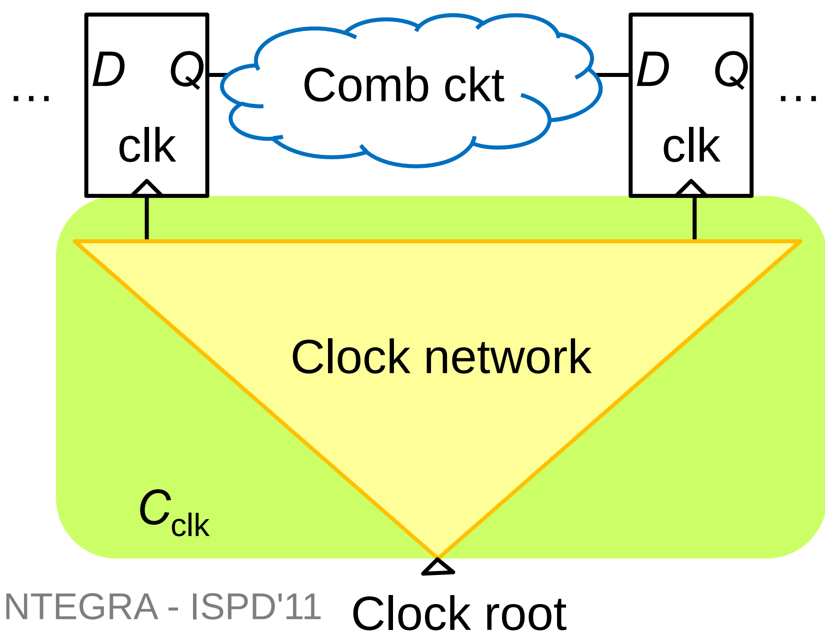
Nat'l Chiao Tung Univ. / Faraday Tech Corp.

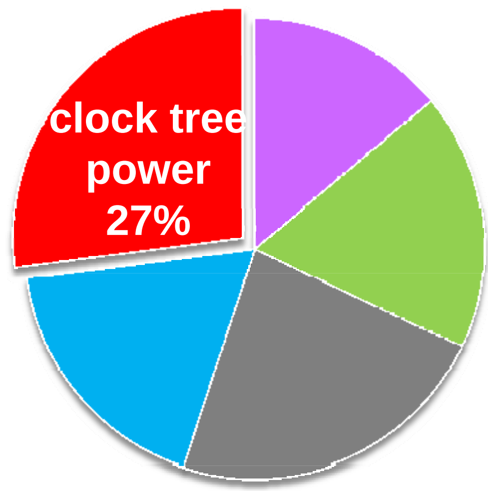
Outline

- **Introduction**
- **Problem & properties**
- **Algorithm - INTEGRA**
- **Experimental results**
- **Conclusion**

Clock Power Dominates!

- **Power has become one bottleneck for circuit implementation**
- **Clock power is the major dynamic power source**
  - The clock signal toggles in each cycle ⇒ High switching activity
- **Clock power model: dynamic power**
  - $P_{clk} = C_{clk}V_{dd}^2 f_{clk}$
  - C_{clk}: switching capacitance charged/discharged by clock





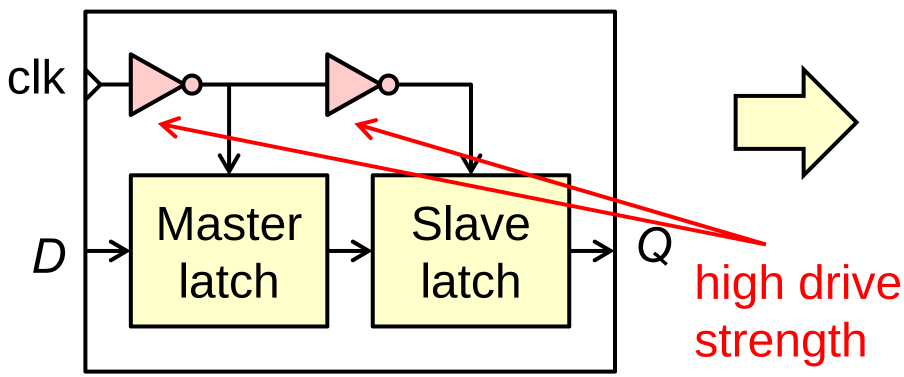
Power breakdown of an ASIC

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

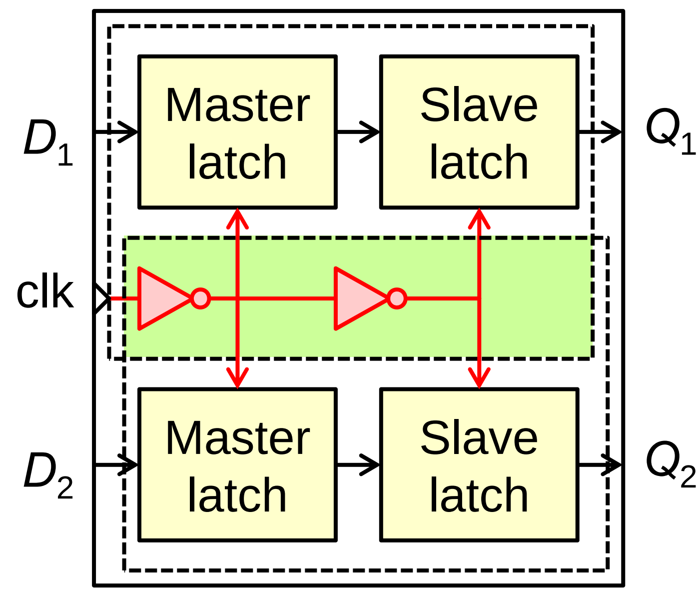
Multi-Bit Flip-Flops

- **A multi-bit flip-flop (MBFF)**
  - Cluster several single-bit flip-flops (share the drive strength)

Single-bit flip-flop



Dual-bit flip-flop



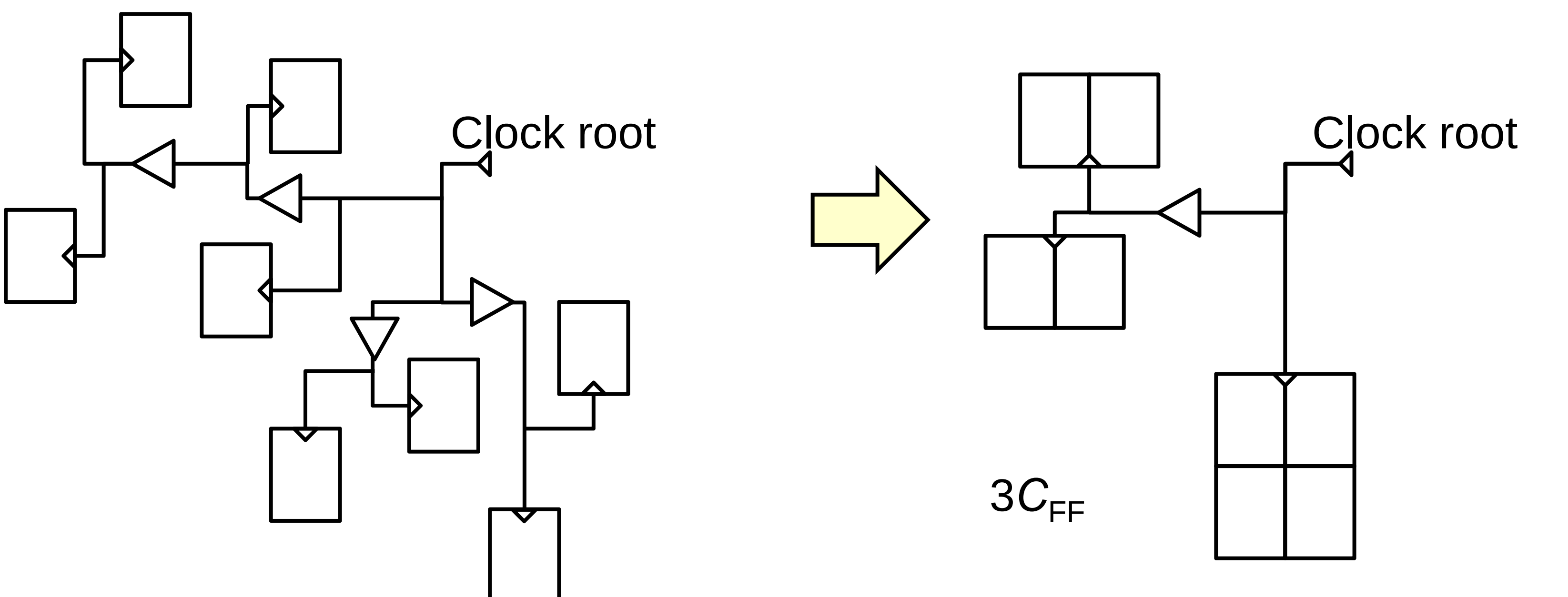
  - Save flip-flop power and area

Bit number	1	2	4
Normalized power per bit	1.000	0.860	0.780
Normalized area per bit	1.000	0.960	0.713

Clock Power Saving using MBFFs (1/2)

- **Reduce switching capacitance charged/discharged by clock**

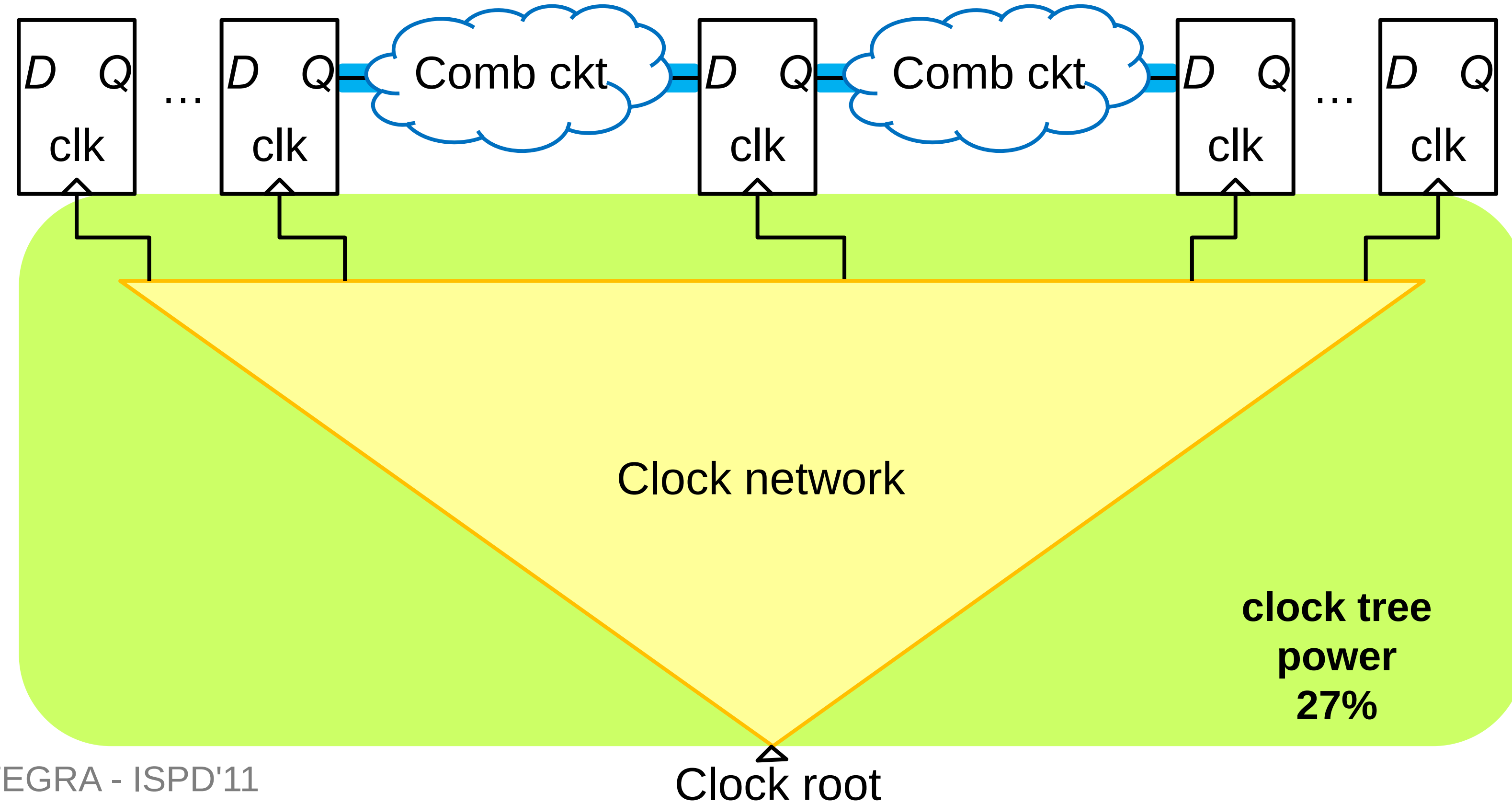
Switching capacitance	Clock power saving	Other benefits
Clock sinks (Flip-flops)	**Small FF capacitance:** Share C into FF clock pins	**Small area:** Share the inverter chain
Clock network (wires, clock buffers)	**Small wire/buf capacitance:** #leaf ↓ ⇒ depth ↓ #buffer ↓	**Regular topology** and **easy skew control**



Pokala *et al.* Physical synthesis for performance optimization. *ASIC*, 1992.

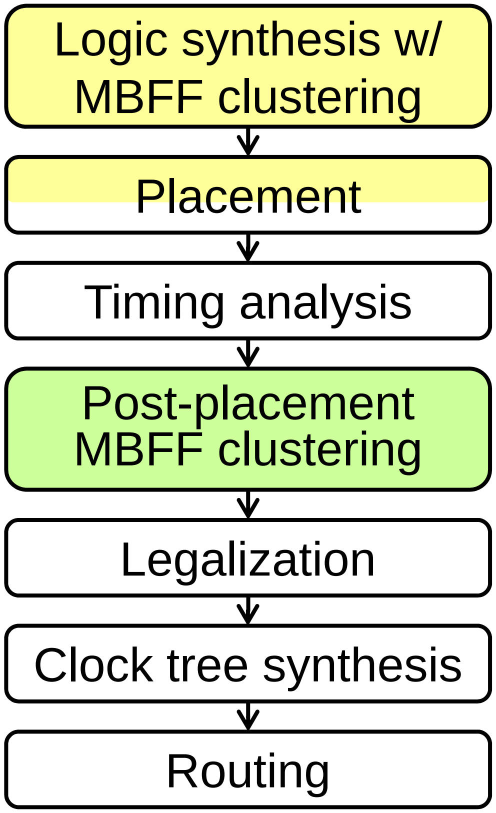
Clock Power Saving using MBFFs (2/2)

- **Clock power reduction can be significant**
  - FF clock pins, clock buffers/inverters, wires in clock network
- **Wire power overhead on data pins is small**
  - Wirelength on data pins << total wirelength



Prior Works on MBFF Clustering

- **Logic synthesis**
  - [Chen *et al.*, SNUG-10]
- **Early physical synthesis**
  - [Hou *et al.*, ISQED-09]
- **Post-placement: timing and routing**
  - [Yan and Chen, ICGCS-10]
    - Minimum clique paritioning
    - Greedy clustering
    - Contiguous and infinite MBFF library
  - [Chang *et al*., ICCAD-10]
    - Window-based clustering
    - Maximum independent set
    - Discrete and finite MBFF library



INTEGRA

- **Since post-placement MBFF clustering is NP-hard, our goal is to solve it effectively and efficiently instead of optimally.**
  - Do not enumerate all possible combinations (maximal cliques)
  - Do not relate to the number of layout grids/bins
  - Do not manipulate on a general graph

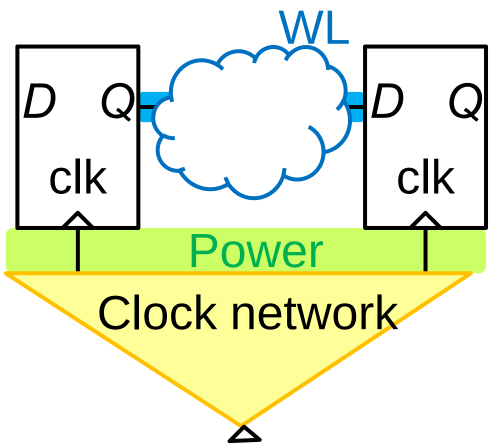
- **Features:**
  - Efficient representation: a pair of linear-size sequences
  - Fast operations: coordinate transformation
  - Few decision points: #decision points << #flip-flops
    - We cluster flip-flops at only decision points thus leading to an efficient clustering scheme.
  - Global relationships among flip-flops: cross bin boundaries

Outline

- Introduction
- **Problem & properties**
- Algorithm - INTEGRA
- Experimental results
- Conclusion

The Multi-Bit Flip-Flop Clustering Problem

- **Clock power saving using multi-bit flip flops**
- **Given**
  - MBFF library
  - Nelist & Placement
  - Timing slack constraints (in terms of wirelength)
  - Placement density constraint
- **Find**
  - MBFF clustering to
  - Minimize
    - Clock dynamic power
    - Wirelength
  - Subject to
    - Timing slack constraints (in terms of wirelength)
    - Placement density constraints



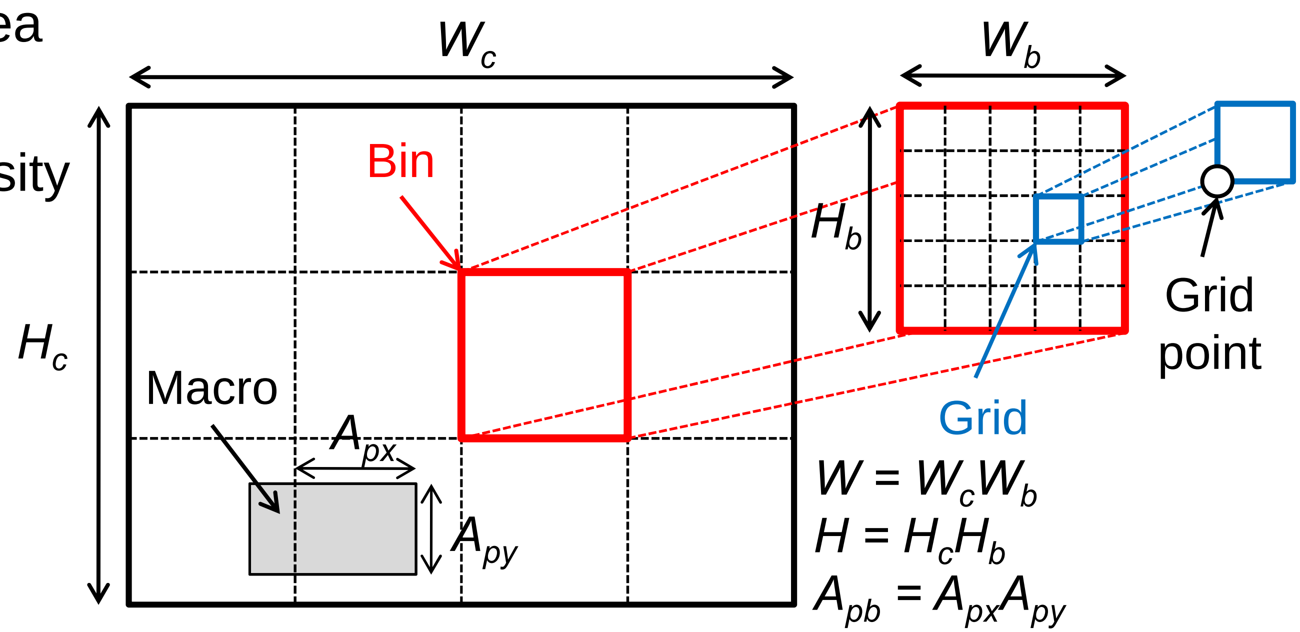
MBFF Library

- **MBFF library**
  - Lexicographical order: <1,100,100>, <2,172,192>, <4,312,285>

Bit number	Power	Area	Normalized power per bit	Normalized area per bit
1	100	100	1.00	1.00
2	172	192	0.86	0.96
4	312	285	0.78	0.71

Placement

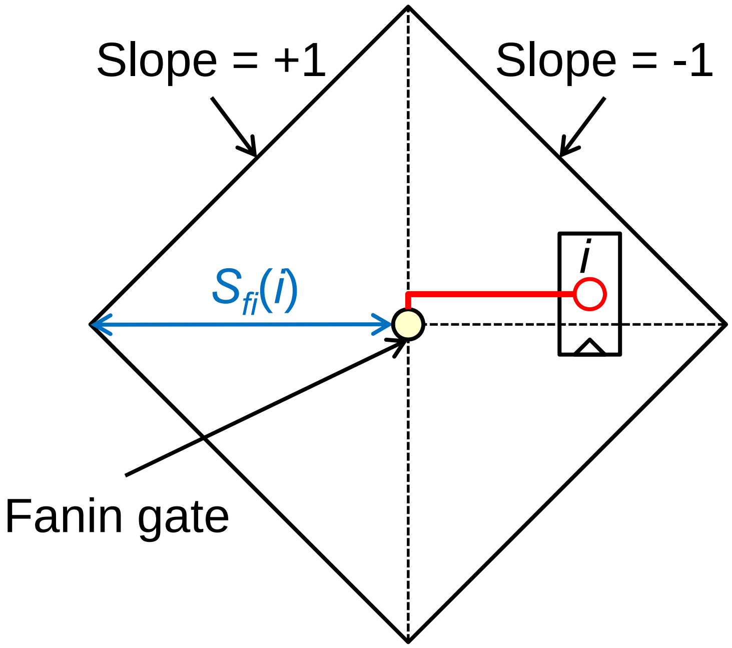
- **Chip area** = $W_c H_c$ bins = WH grids
- **Flip-flops should be placed on grid (left-bottom corner)**
- **Placement density constraint for bin *b*:**
  - $A_{fb} \leq T_b(W_b H_b A_g - A_{pb}) - A_{cb}$
  - A_{fb}: FF area
  - A_{cb}: Combinational logic area
  - A_{pb}: macro area
  - A_g: grid area
  - T_b: target density



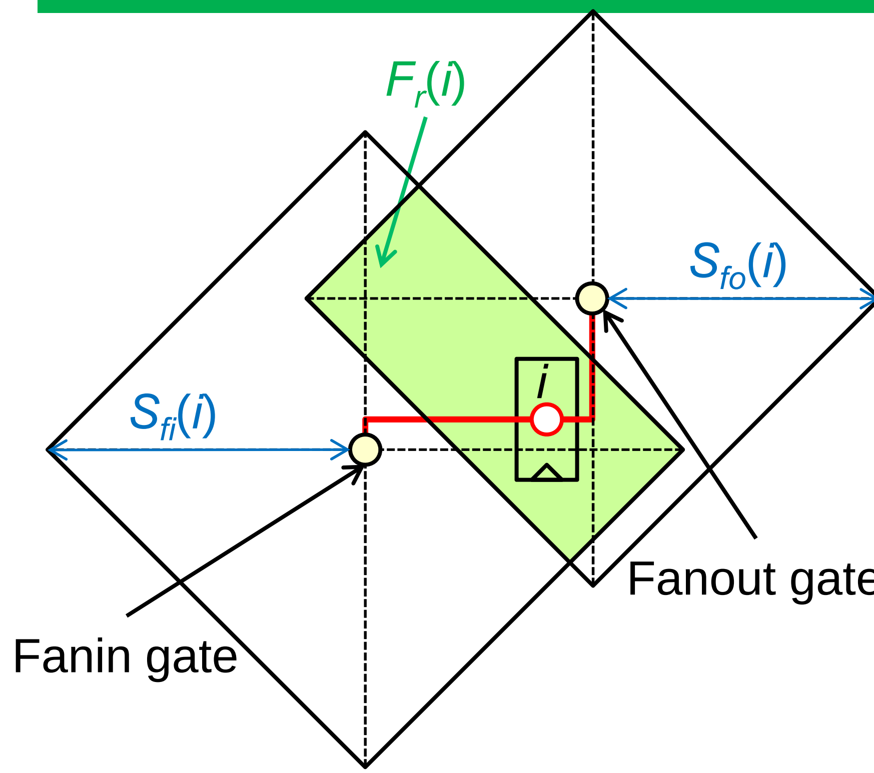
Timing Slack and Feasible Region

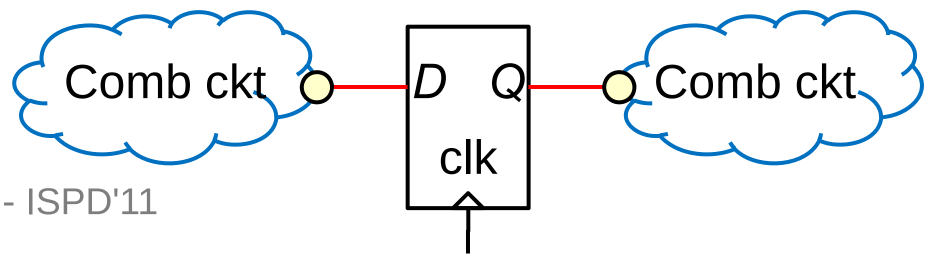
Input slack

- **Slack ⇔ wirelength**



Feasible region

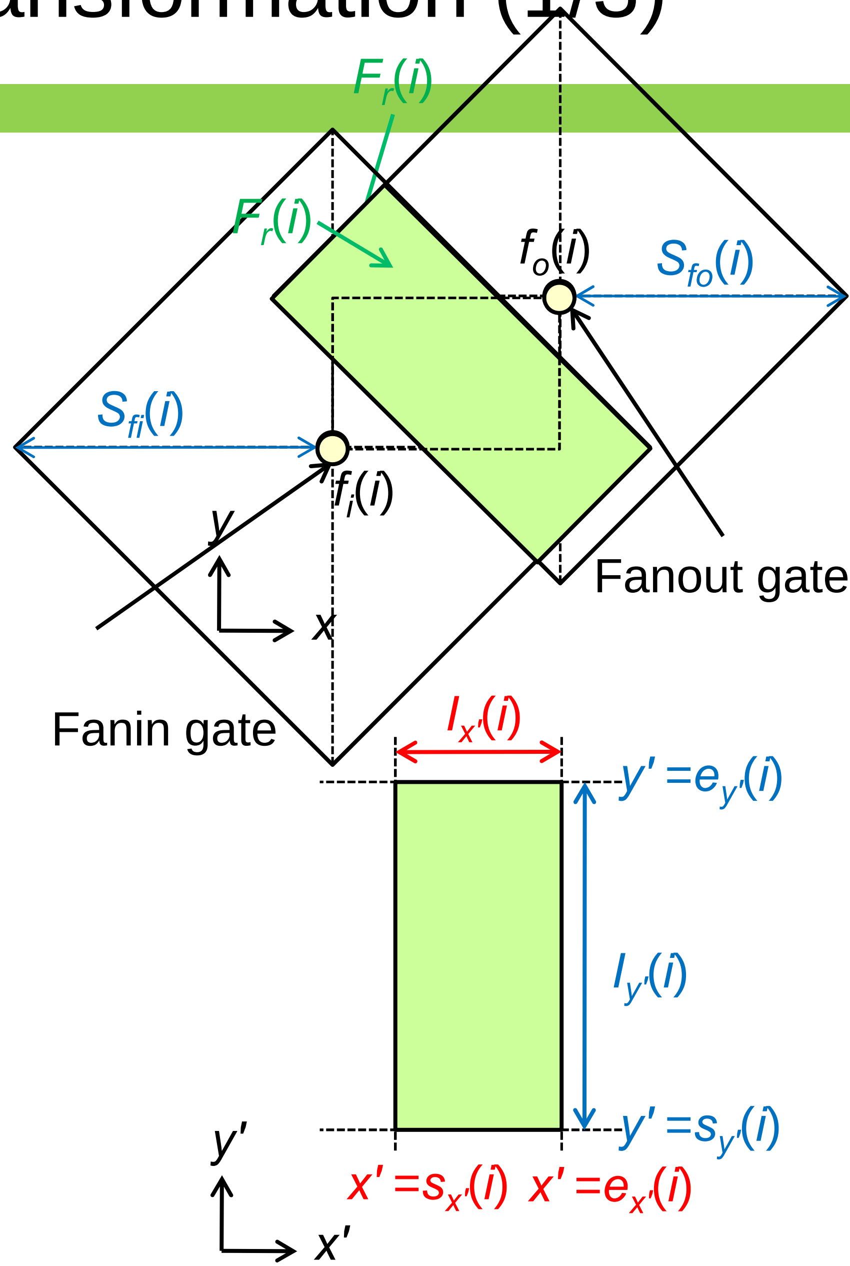




Multiple-fanout:
multiple fanout diamonds

Coordinate Transformation (1/3)

- **It's hard to determine if a grid point is located inside or outside the feasible region**
- **Rotate 45° clockwise; we have rectangles instead**
  - Easy checking!

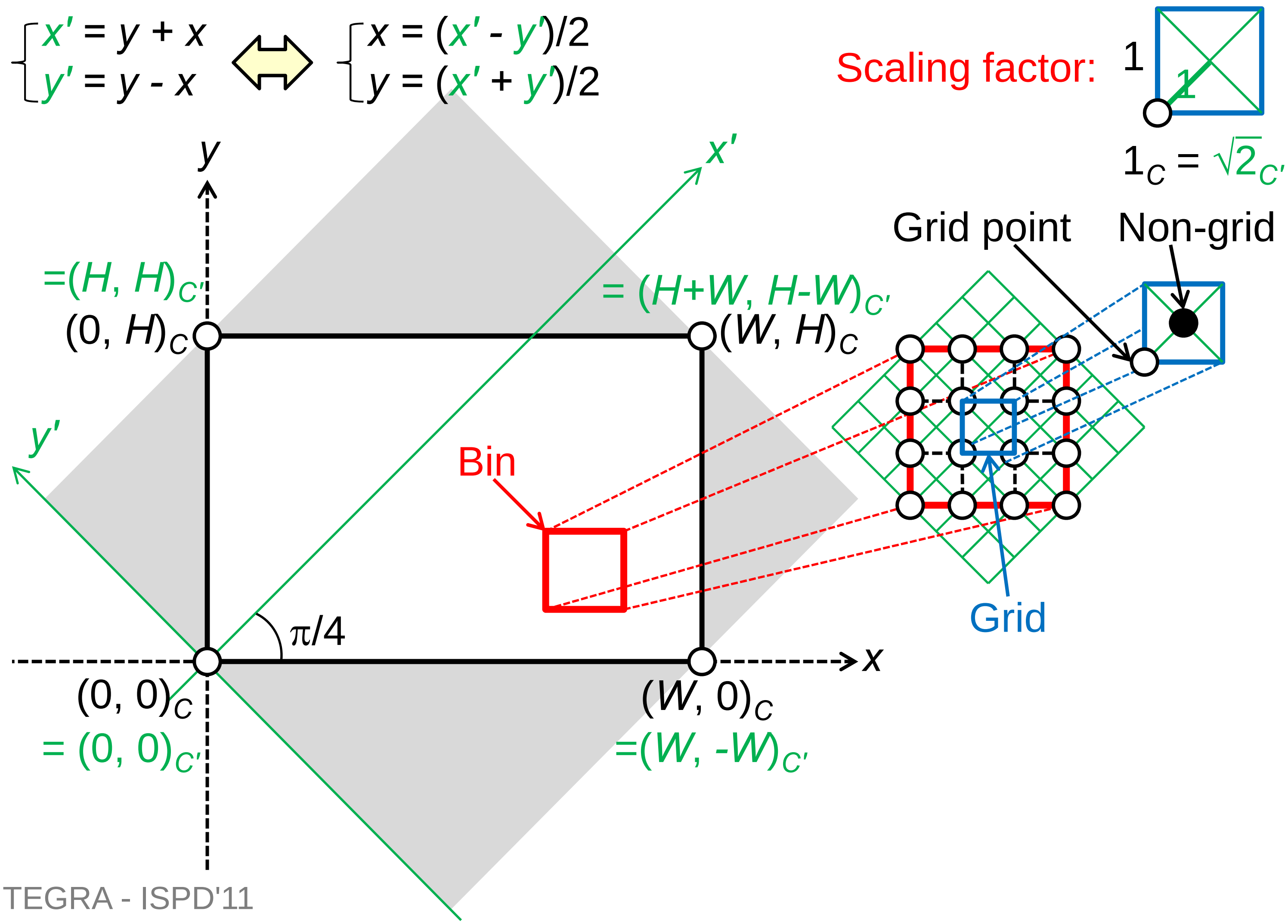


Coordinate Transformation (2/3)

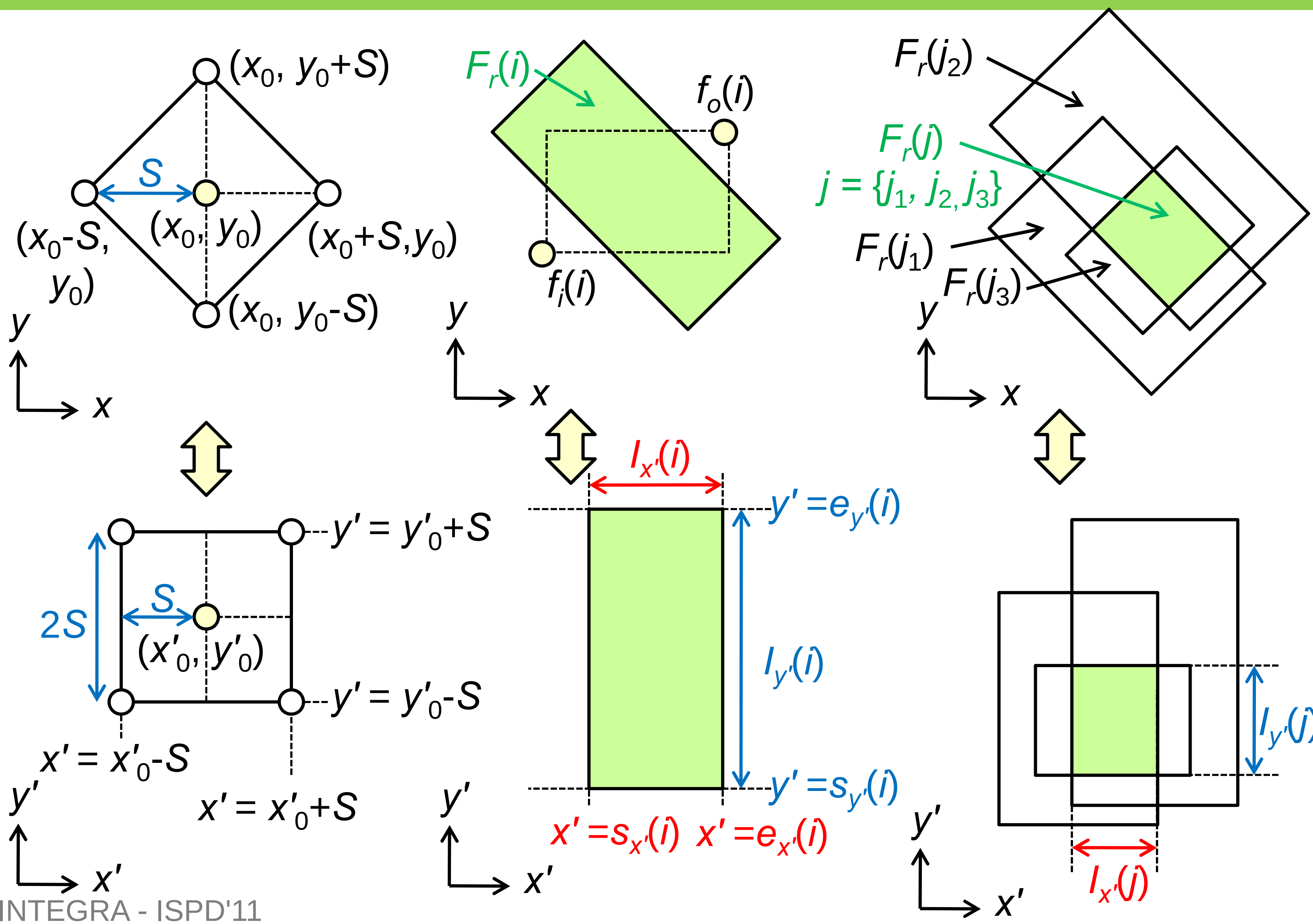
- **Coordinate transformation is done by integer operations**

$$\begin{cases} x' = y + x \\ y' = y - x \end{cases} \Leftrightarrow \begin{cases} x = (x' - y')/2 \\ y = (x' + y')/2 \end{cases}$$

Scaling factor: $1_C = \sqrt{2}_{C'}$

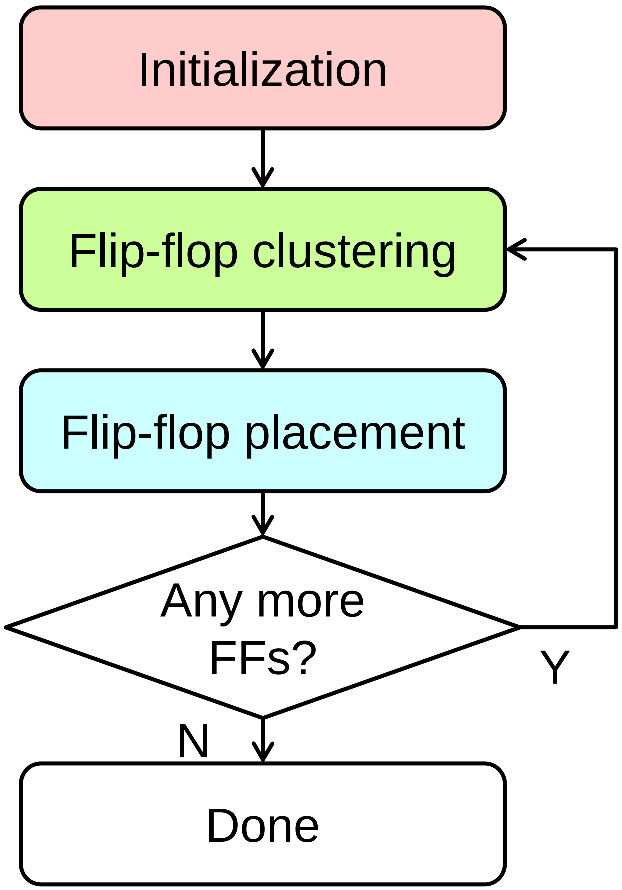


Coordinate Transformation (3/3)



Outline

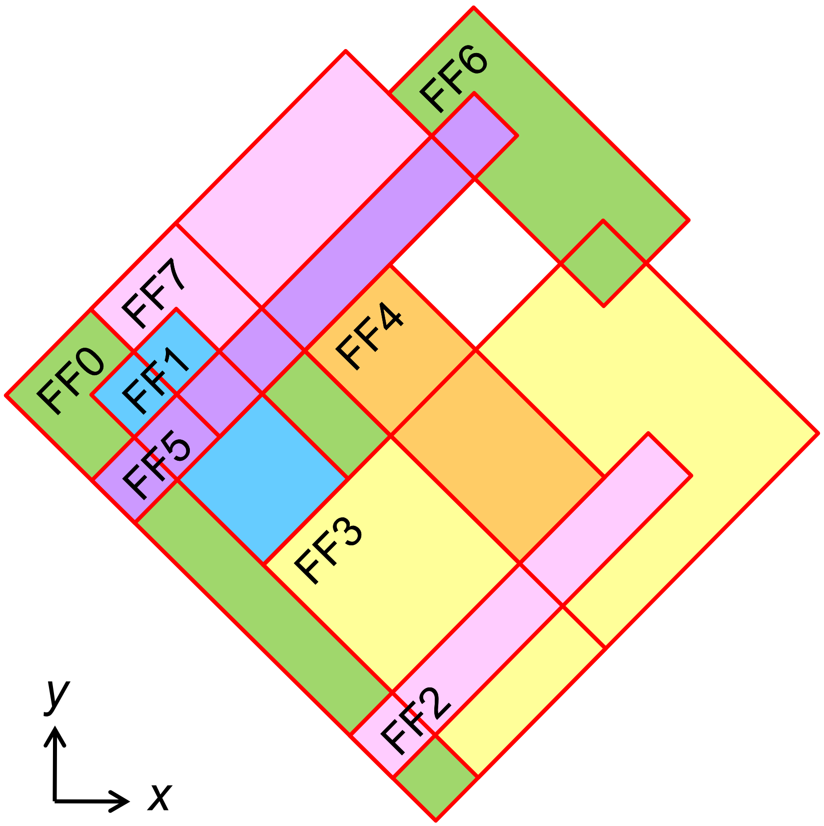
Overview of INTEGRA



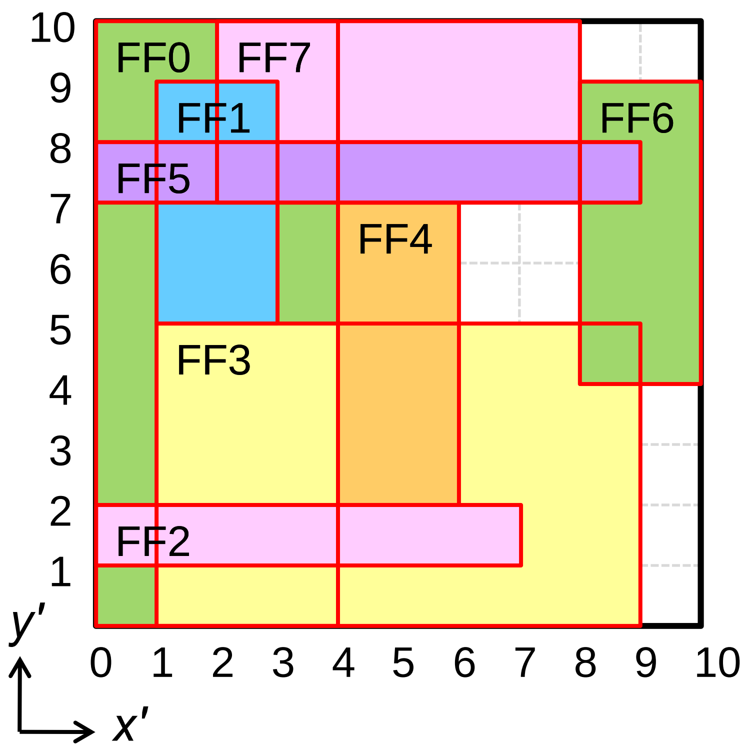
1. Analyzes the design intent
2. Finds a decision point in *X'* and extracts the essential flip-flops and their related flip-flops
3. Finds the maximal clique in the partial *Y'* for each essential flip-flop
4. Clusters each essential flip-flop
5. Places the clustered flip-flop at a legal location with routing cost and density consideration
6. Repeats steps 2–5 until all flip-flops are investigated

Example (1/5)

Initial

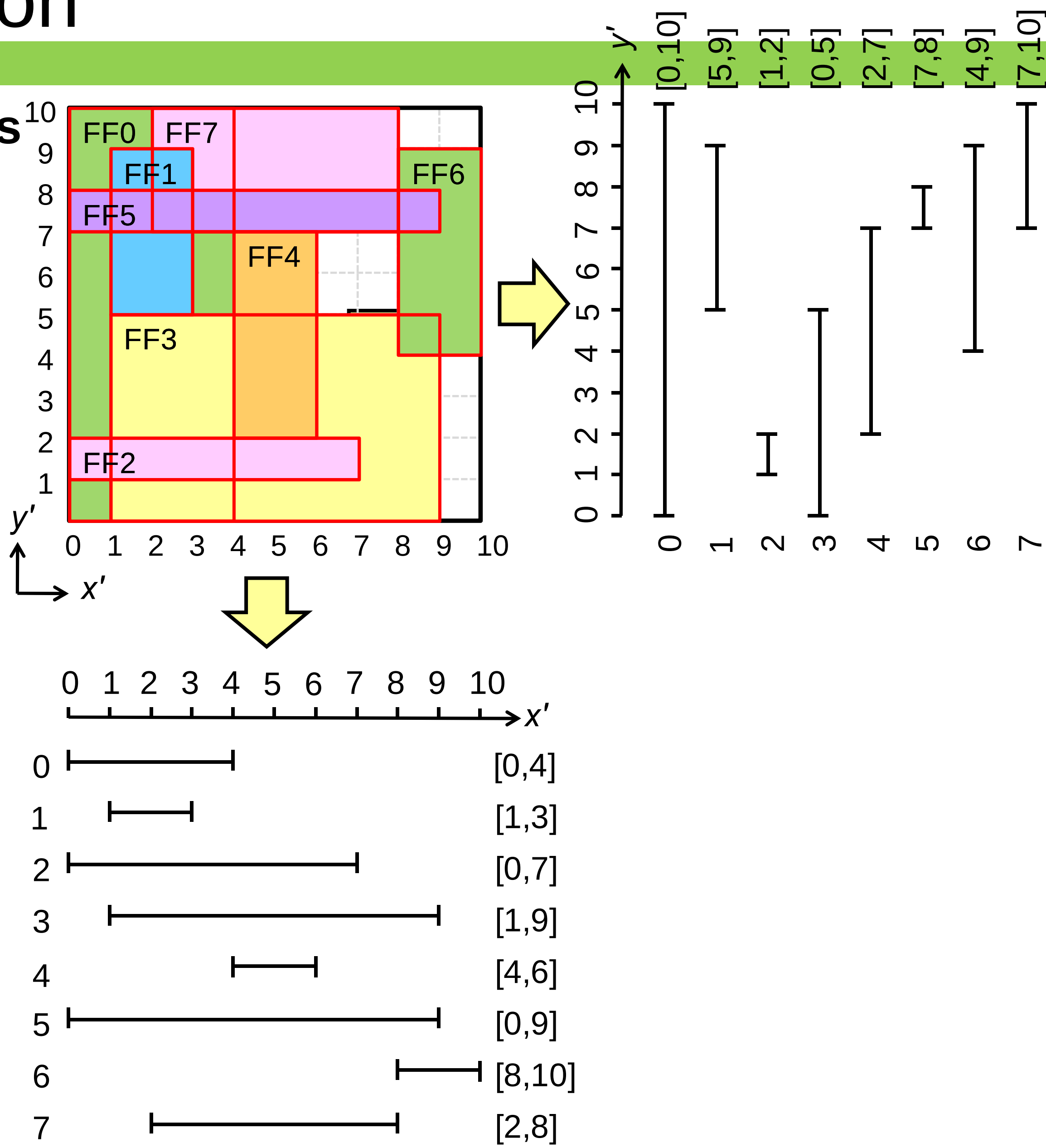


Transformed



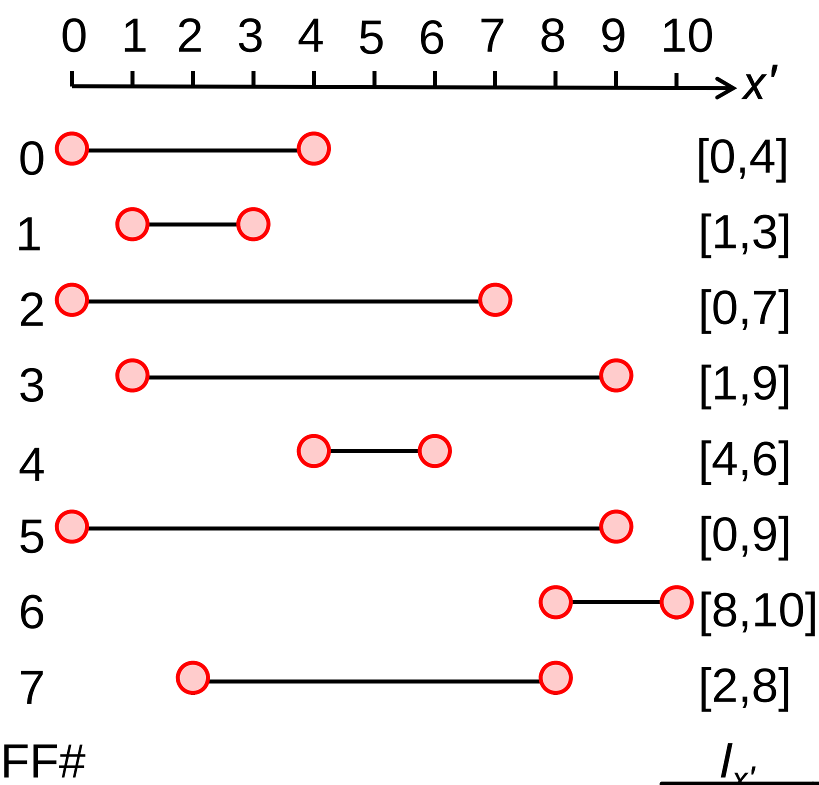
Example (2/5)
- Representation

- **Two interval graphs**

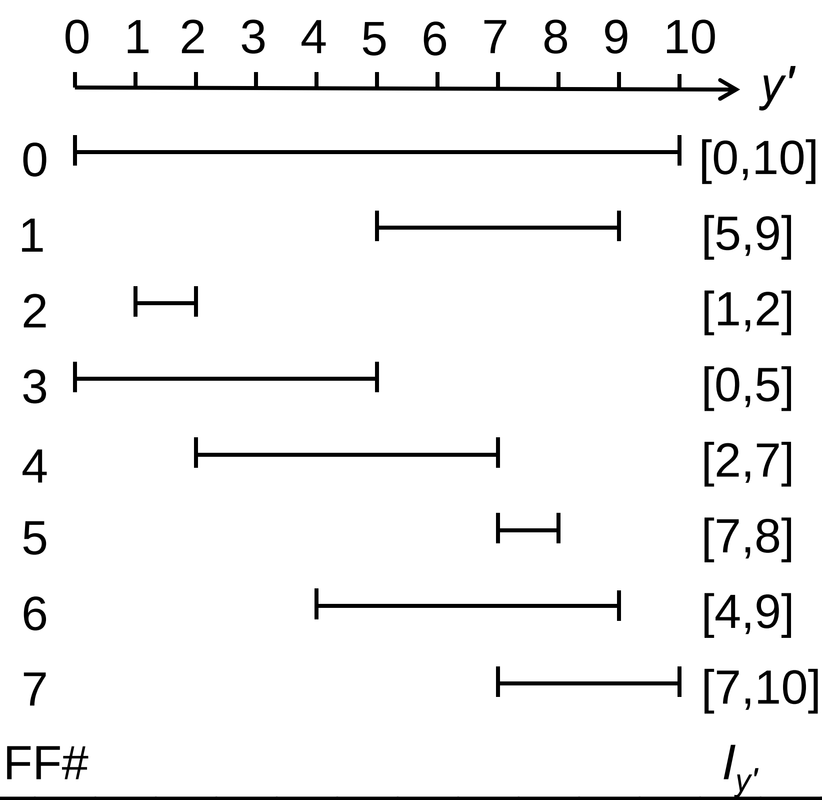


Example (2/5)
- Representation

X'



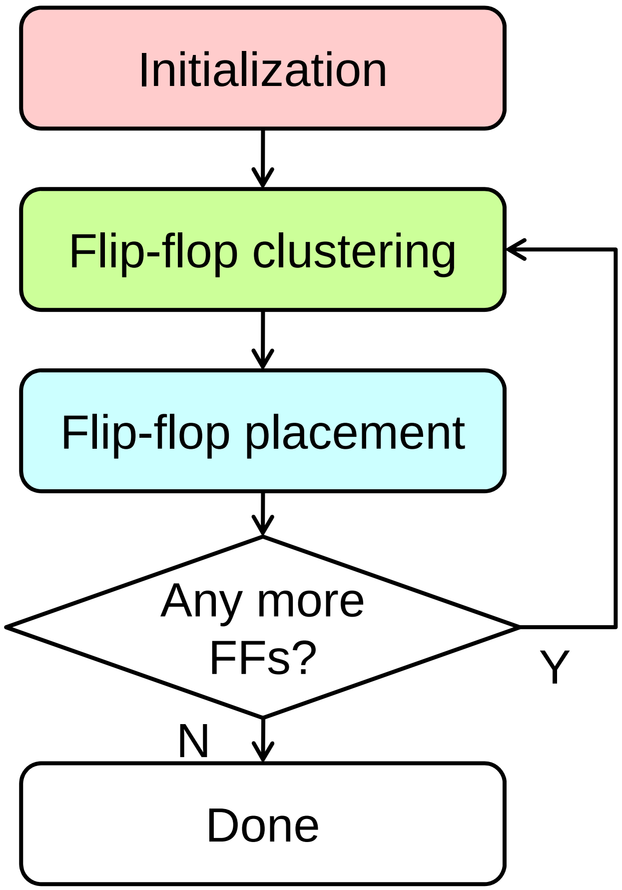
Y'



X' :

Type	s	s	s	s	s	s	e	s	e	e	e	s	e	e	e	e
FF#	0	2	5	1	3	7	1	4	0	4	2	6	7	3	5	6
x'	0	0	0	1	1	2	3	4	4	6	7	8	8	9	9	10

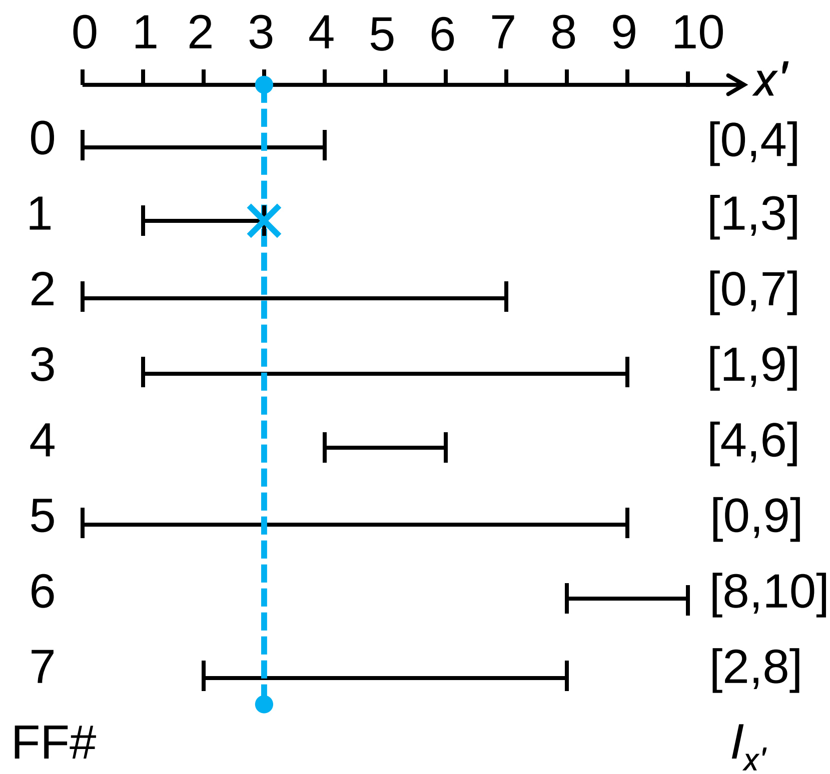
Overview of INTEGRA

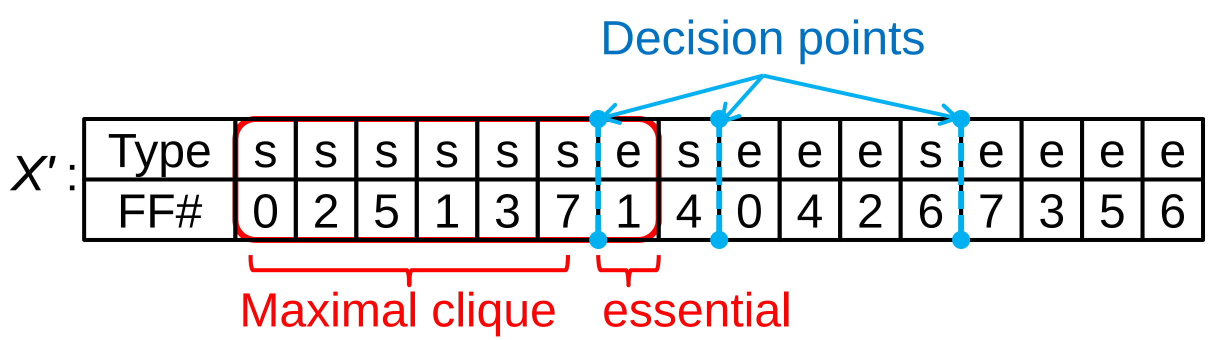


1. Analyzes the design intent
2. Finds a decision point in *X'* and extracts the essential flip-flops and their related flip-flops
3. Finds the maximal clique in the partial *Y'* for each essential flip-flop
4. Clusters each essential flip-flop
5. Places the clustered flip-flop at a legal location with routing cost and density consideration
6. Repeats steps 2–5 until all flip-flops are investigated

Decision Points and Essential Flip-Flops

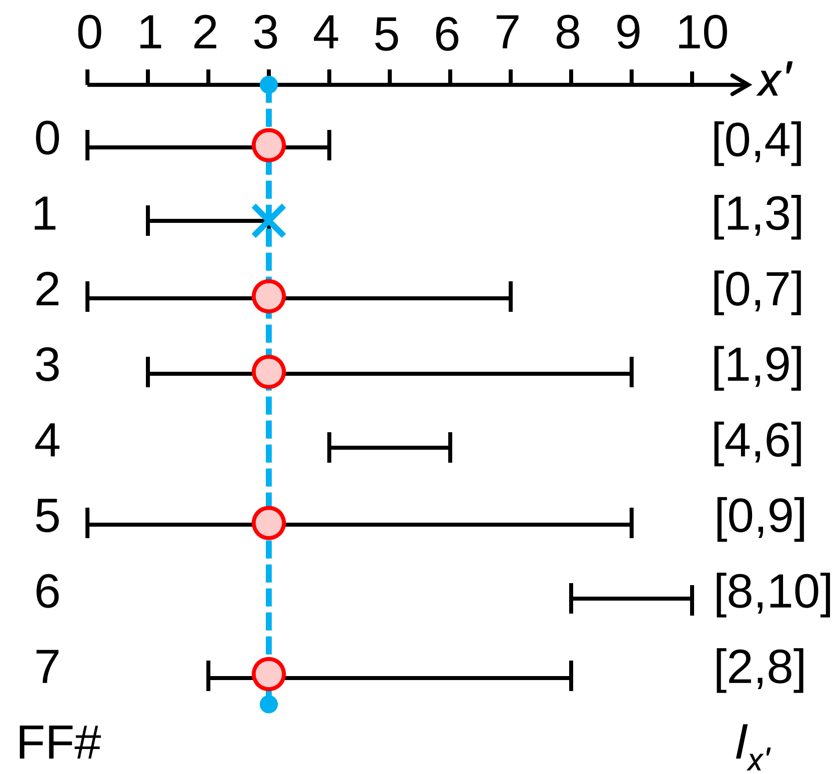
- **Definition:** If there exist two consecutive points x_k' and x_{k+1}' in X', where $x_k' = s_{x'}(i)$, $x_{k+1}' = e_{x'}(j)$, $1 \leq i, j \leq n$, a decision point is the coordinate of x_{k+1}', i.e., $e_{x'}(j)$.
- **Definition:** The essential flip-flops with respect to a decision point are the flip-flops whose end points ordered from this decision point to the next decision point or to the end of X' for the last decision point.

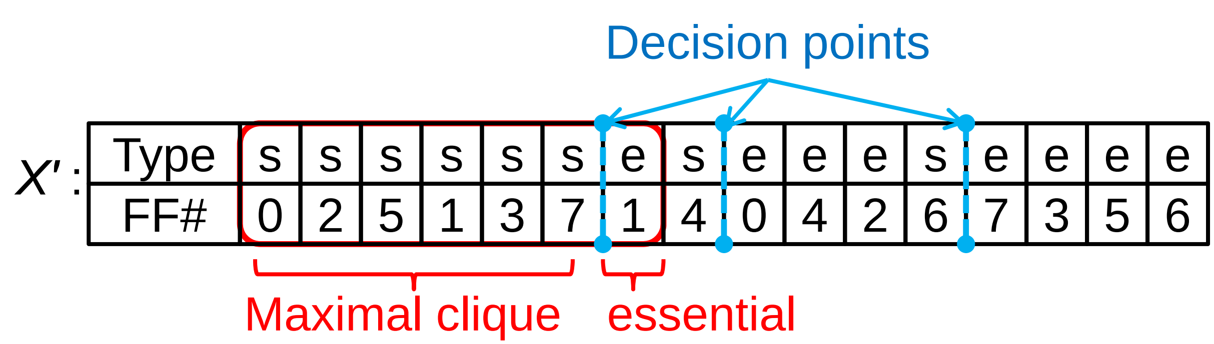




Decision Points and Essential Flip-Flops

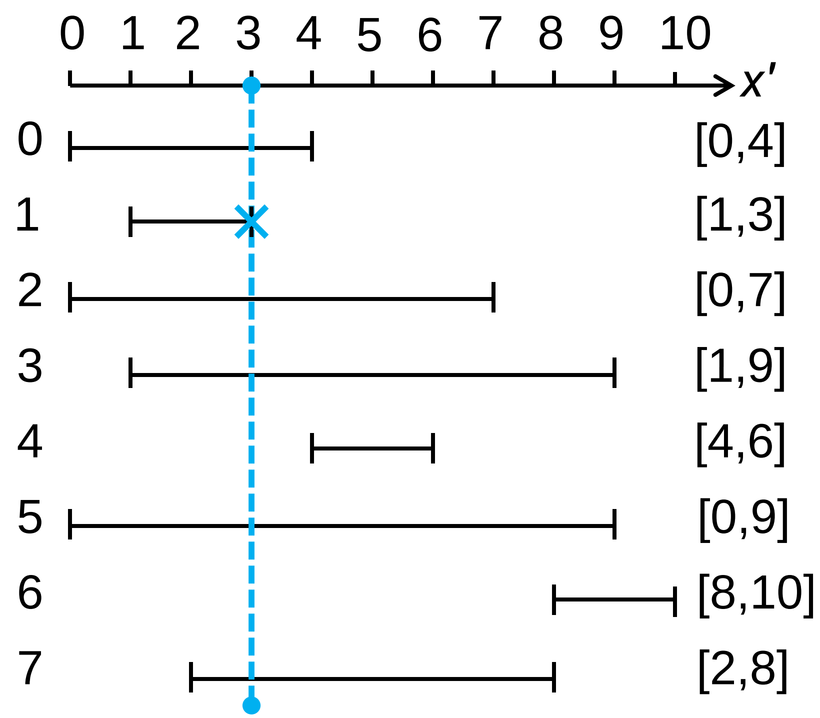
- **Theorem:** Consider X', a decision point, and the corresponding essential flip-flops. The maximal clique containing the essential flip-flops in x' interval graph can be found at this decision point.
- **Corollary:** A decision point corresponds to at least one essential flip-flop. Hence, the number of decision points is less than or equal to the number of flip-flops.

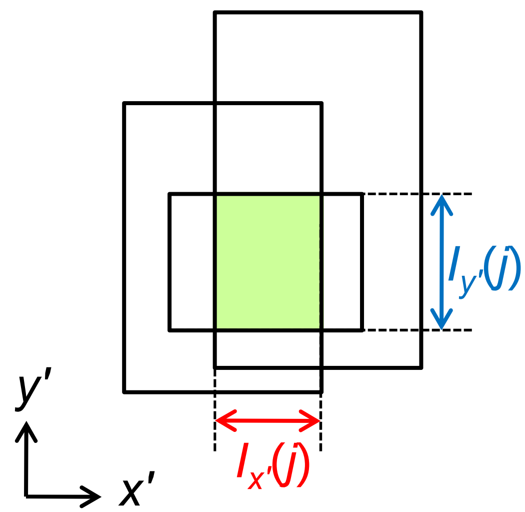




Example (3/5)
- Flip-Flop Clustering

X′: Find candidates

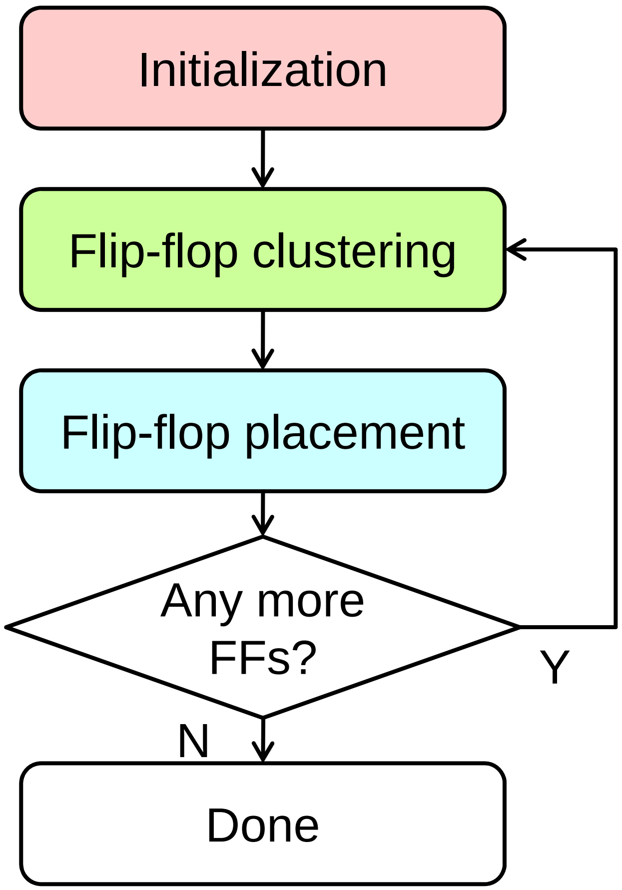




X′ :

Type	s	s	s	s	s	s	e	s	e	e	e	s	e	e	e	e
FF#	0	2	5	1	3	7	1	4	0	4	2	6	7	3	5	6
x′	0	0	0	1	1	2	3	4	4	6	7	8	8	9	9	10

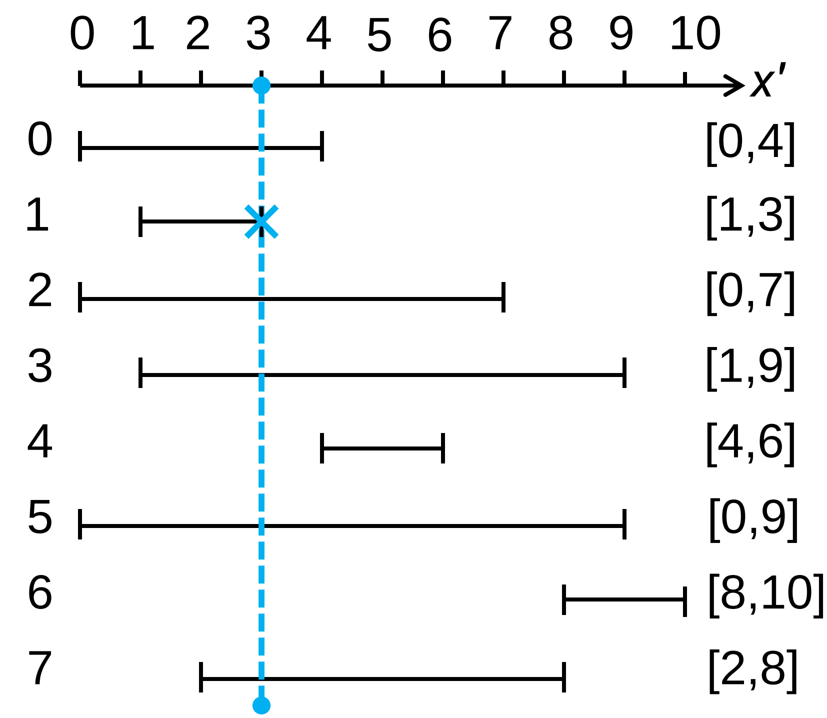
Overview of INTEGRA



1. Analyzes the design intent
2. Finds a decision point in *X'* and extracts the essential flip-flops and their related flip-flops
3. Finds the maximal clique in the partial *Y'* for each essential flip-flop
4. Clusters each essential flip-flop
5. Places the clustered flip-flop at a legal location with routing cost and density consideration
6. Repeats steps 2–5 until all flip-flops are investigated

Example (3/5)
- Flip-Flop Clustering

X': Find candidates

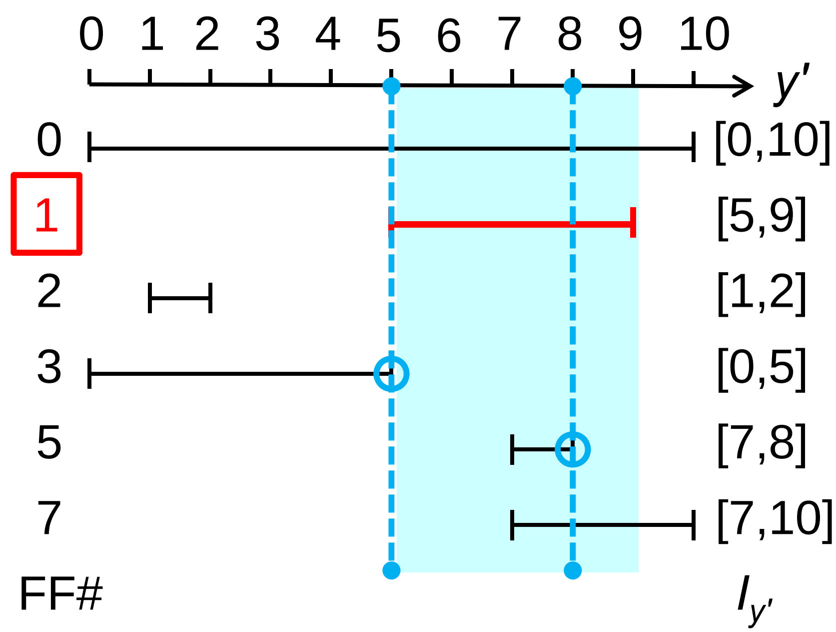


$I_{x'}$

X' :

Type	s	s	s	s	s	s	e	s	e	e	e	s	e	e	e	e
FF#	0	2	5	1	3	7	1	4	0	4	2	6	7	3	5	6
x'	0	0	0	1	1	2	3	4	4	6	7	8	8	9	9	10

Y': Verify and cluster MBFF



$I_{y'}$

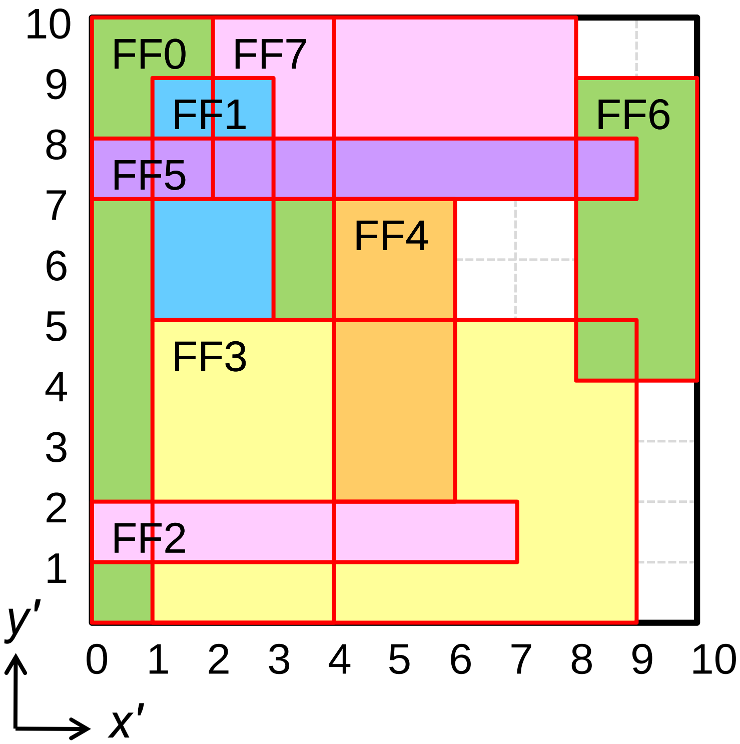
Y' :

Type	s	s	s	e	s	e	s	s	e	e
FF#	0	3	2	2	1	3	5	7	5	1

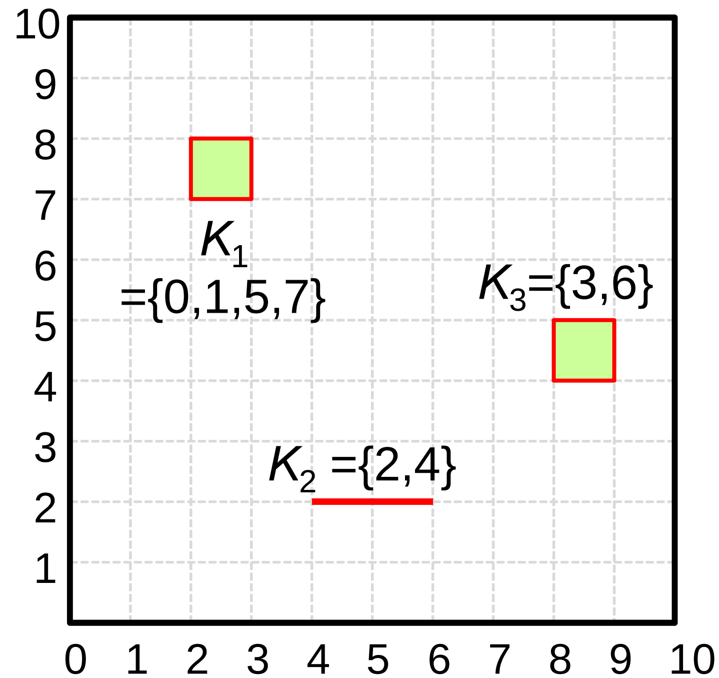
K_1: {0,1,5,7}

Example (4/5)
- Flip-Flop Clustering

Initial

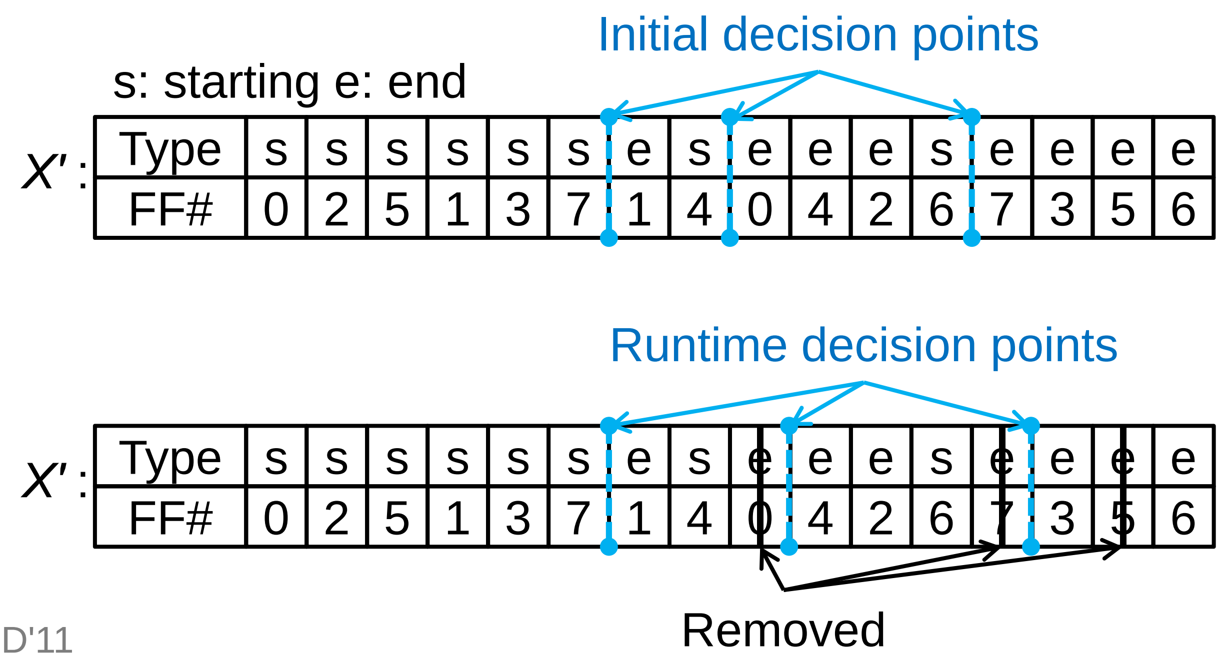


MBFFs & their feasible regions

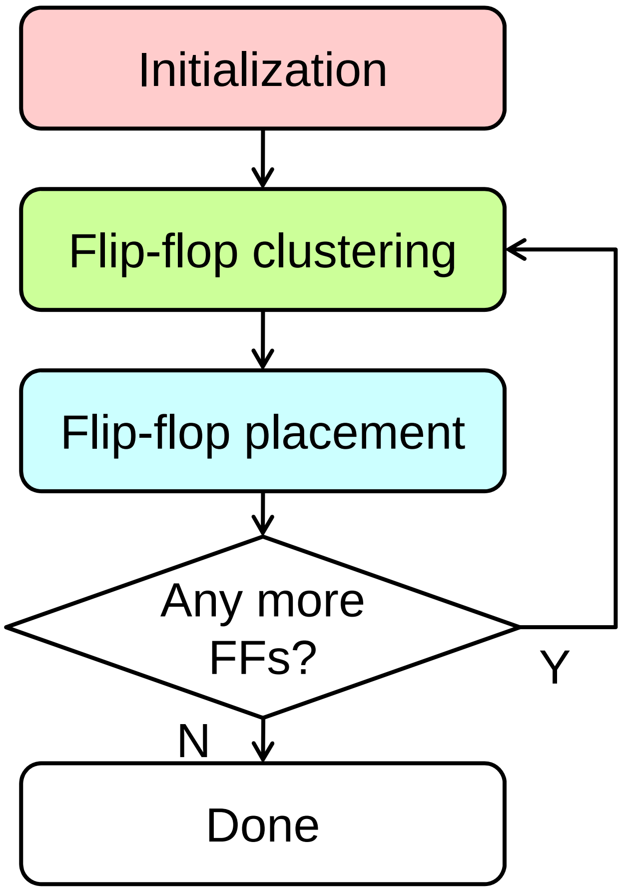


Runtime Decision Points Are Few!

- **Corollary:** A decision point corresponds to at least one essential flip-flop. Hence, the number of decision points is less than or equal to the number of flip-flops.
- **Runtime decision points ≤ initial decision points**
  - Runtime decision points are shifted because of removed flip-flops.



Overview of INTEGRA



1. Analyzes the design intent
2. Finds a decision point in *X'* and extracts the essential flip-flops and their related flip-flops
3. Finds the maximal clique in the partial *Y'* for each essential flip-flop
4. Clusters each essential flip-flop
5. Places the clustered flip-flop at a legal location with routing cost and density consideration
6. Repeats steps 2–5 until all flip-flops are investigated

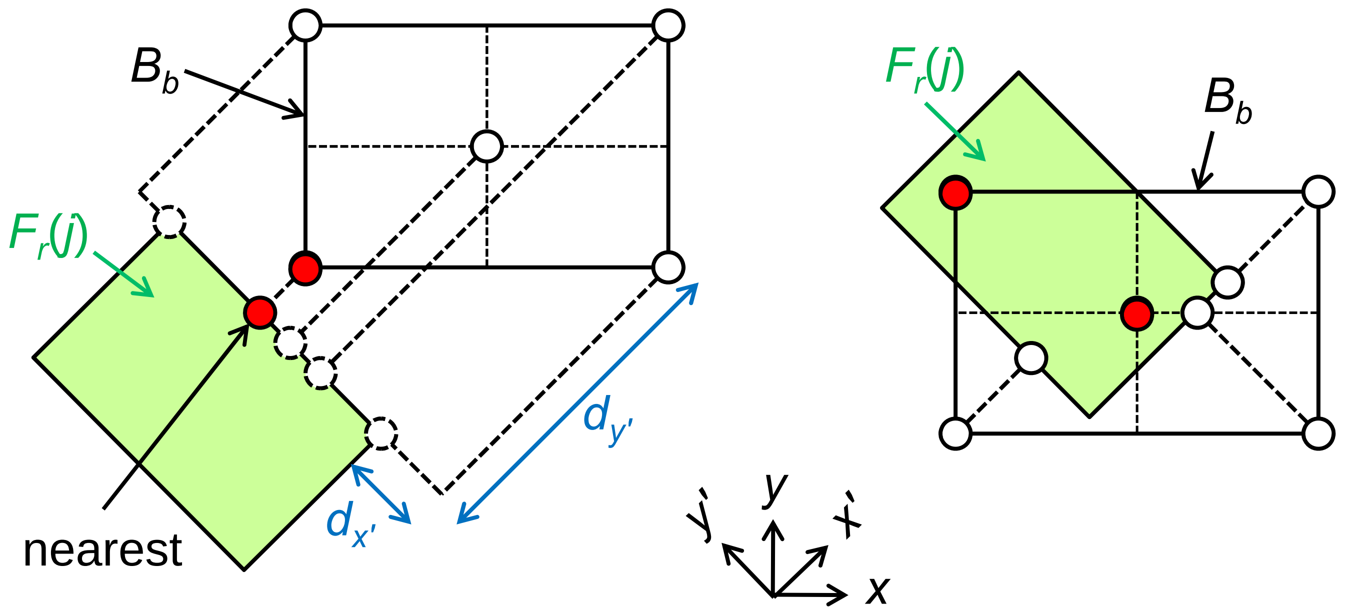
Legal Grid Points

- **Place MBFFs at legal grid points.**

- **A legal grid point satisfies the following conditions:**
  - It is a grid point.
  - It is not occupied by other gates or flip-flops.
  - It is density-safe.

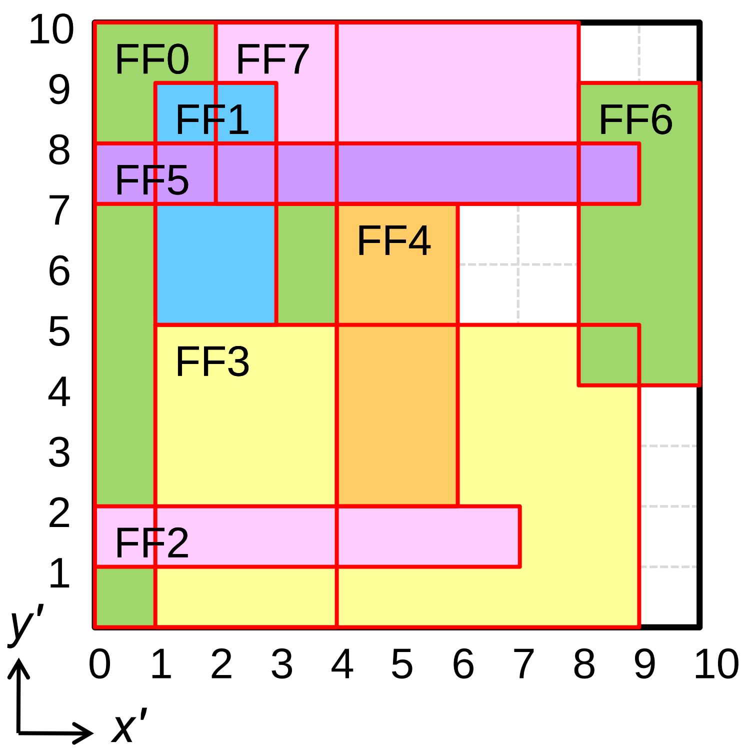
Flip-Flop Placement

- **Goal: Find a legal placement with wirelength consideration**
  - Optimal location: Within the bounding box of median coordinates of fanin and fanout gates

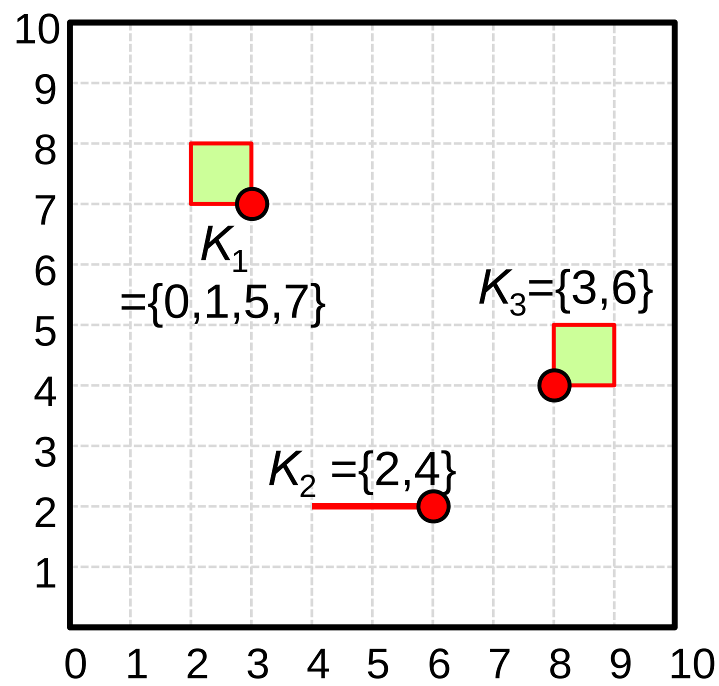


Example (5/5)
- Flip-Flop Placement

Initial



Placed MBFFs



Procedure of INTEGRA

Algorithm INTEGRA

// Initialization

1. lexicographically sort the MBFF library
2. collapse MBFFs
3. $X' \leftarrow$ sort $\{s_{x'}(i), e_{x'}(i): i = 1..n\}$, $j \leftarrow 1$, $Q \leftarrow \varnothing$

// Main body

4. **while** (X' is not empty) **do**
5. find a decision point in X'
6. $Q \leftarrow Q$ + essential flip-flops and related flip-flops
7. $Y' \leftarrow$ sort $\{s_{y'}(i), e_{y'}(i): i \in Q\}$
8. **foreach** essential flip-flop k **do**

   // Flip-flop clustering

9. $K_{max} \leftarrow$ max_clique(Y', k)
10. find the appropriate MBFF cell of bit number B for $|K_{max}|$
11. $K_{max} \leftarrow$ sort $\{e_{x'}(i): i \in K_{max} - \{k\}\}$
12. $K_j \leftarrow$ flip-flop k and the first (B-1) flip-flops in K_{max}

    // Flip-flop placement

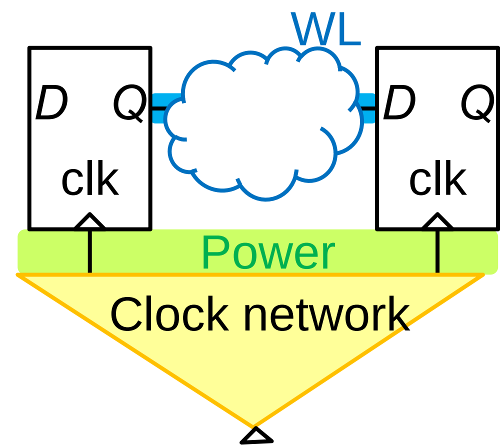
13. find bounding box B_b for K_j
14. project B_b's corner and center points to $F_r(K_j)$
15. find the projected point with min distance between B_b and $F_r(K_j)$
16. legalize this point and assign it to MBFF K_j
17. **if** legalization fails **then** go to line 9
18. $Q \leftarrow Q - K_j$, $X' \leftarrow X' - K_j$
19. j++

Outline

- Introduction
- Problem & properties
- Algorithm - INTEGRA
- **Experimental results**
- **Conclusion**

Comparison
- Post-Placement MBFF Clustering

Circuit	#FFs	Chip size (#Grids)	Initial	
			Power	Wirelength
C1	120	600×600	11,384	89,425
C2	480	1,200×1,200	46,404	348,920
C3	1,920	2,400×2,400	185,616	1,395,680
C4	5,880	4,200×4,200	566,972	4,290,655
C5	12,000	6,000×6,000	1,160,100	8,723,000
C6	192,000	24,000×24,000	18,561,600	139,568,000



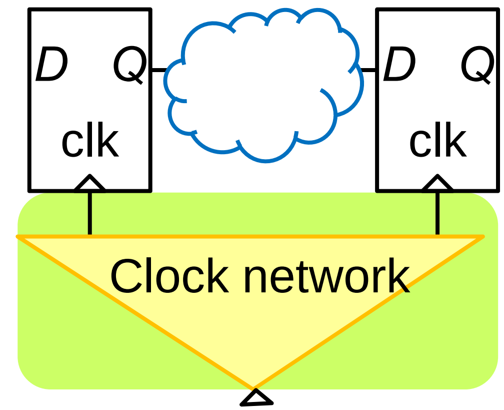
FF library cells (Bit-number, power, area): (1,100,100), (2,172,192), (4,312,285)

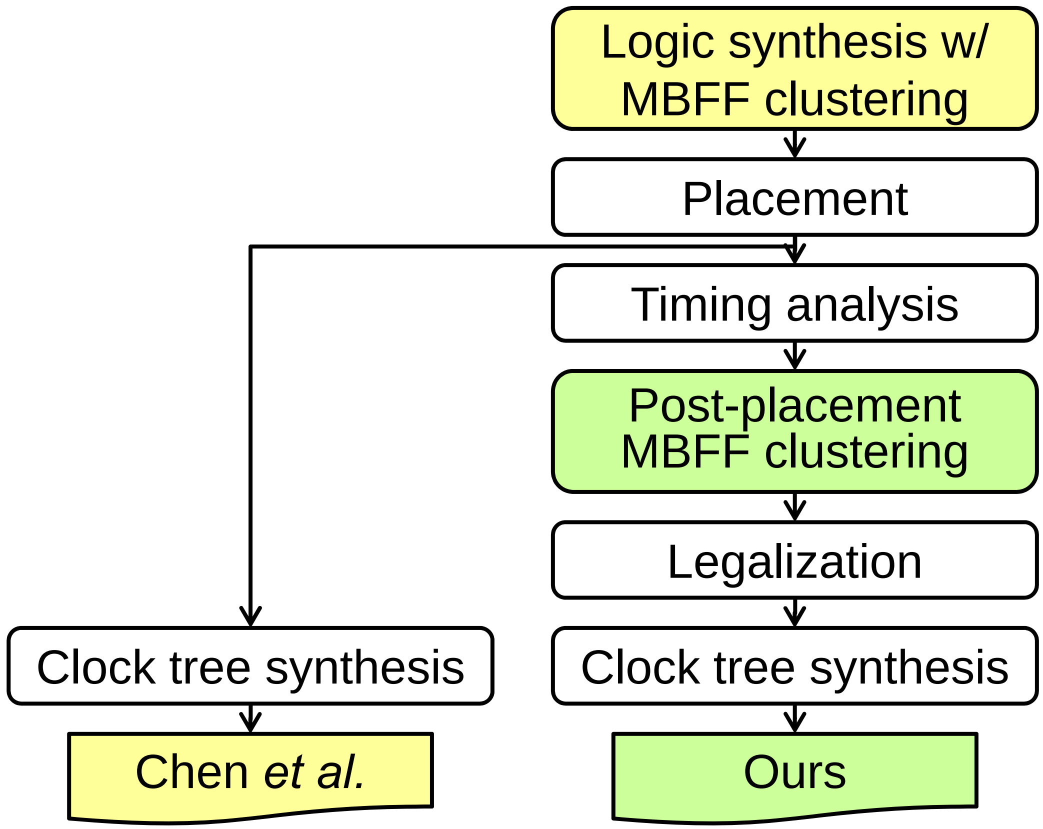
Circuit	Lower bound		Modified Yan&Chen			Chang *et al.*			INTEGRA			
	Power ratio	WL ratio	Power ratio	WL ratio	Time (s)	Power ratio	WL ratio	Time (s)	Power ratio	WL ratio	#Dec	Time (s)
C1	82.2%	48.7%	82.8%	123.0%	0.03	85.2%	91.7%	< 0.01	82.8%	96.4%	28	< 0.01
C2	80.7%	49.9%	81.2%	124.8%	0.11	83.1%	94.7%	0.02	80.9%	102.0%	90	< 0.01
C3	80.7%	49.9%	81.3%	125.2%	0.53	82.9%	94.8%	0.07	80.8%	103.6%	229	< 0.01
C4	80.9%	49.7%	81.5%	124.7%	2.55	83.2%	94.5%	0.23	81.0%	104.1%	458	0.02
C5	80.7%	49.9%	81.3%	124.2%	8.01	82.9%	94.9%	0.52	80.7%	104.8%	690	0.05
C6	80.7%	49.9%	81.3%	124.4%	1994.61	82.8%	94.9%	76.94	80.7%	105.3%	3,007	1.11
Avg. ratio	+0.00%		+0.60%		358.61	+2.36%		16.87	+0.17%		12%	1.00

Chang *et al*. Post-placement power optimization with multi-bit flip-flops. *ICCAD*, 2010.

Yan and Chen. Construction of constrained multi-bit flip-flops for clock power reduction. *ICGCS*, 2010.

Comparison
- MBFF Clustering at Logic Synthesis





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

Comparison
- MBFF Clustering at Logic Synthesis

RISC32 CPU	Chen *et al.*	Ours
# Single-bit FFs	3,689	75
# Dual-bit FFs	2,155	3,962
FF replacement rate	53.88%	99.06%
# Clock tree leaves	5,844	4,037
Clock tree synthesis report		
Normalized dynamic power for combinational ckt	1.000	1.009
Normalized dynamic power for clock buffers	1.000	0.789
Normalized dynamic power for FFs	1.000	0.933
# Clock subtrees	157	150
# Clock buffers	165	110
Depth of clock tree	5	5

1. RISC32 CPU: gate count 120k, 7999 flip-flops.
2. 55nm process; power supply voltage is 0.9 V; the target clock skew is 300 ps.
3. MBFF library: 1-bit FF, 2-bit FF

Conclusion

- **INTEGRA is a fast post-placement multi-bit flip-flop clustering algorithm for clock power saving.**
  - Based on coordinate transformation and interval graphs, we adopt a pair of linear-size sequences as the representation.
  - The concept of decision points helps us significantly reduce the times of clustering applied.
- **Compared with prior work applying MBFF clustering at post-placement and early design stages, our results show the superior efficiency and effectiveness of our algorithm.**

Thank You!

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

Backup Slides

Timing Issue

- **Timing slack setting:**
  - Timing budgeting avoids dynamic interference among multi-bit flip-flops.
  - Update the feasible regions of timing related FF's once an MBFF is formed
    - Scanning sequence X' from left to right
- **Timing safety**
  - STA approval.
  - For the Synopsys Liberty library, the delay of a gate, lumped with its output wire delay, is dominated by its output loading.

$$C(i) = C_W(i) + C_O(i) + \sum_{g_j \in FO(g_i)} C_I(j),$$

  - Since the placement of combinational elements is unchanged during post-placement MBFF clustering, the timing slack between a flip-flop and its fanin/fanout gate depends on only the wire loading, i.e., the Manhattan distance between them.

Placement Issue

- **Placement density constraint**
  - MBFF consume less area
  - Density constraint becomes looser and looser during MBFF clustering

- **Legalization?**
  - Easy and doable

Maximal Clique in *Y'*

- **Find maximal cliques in some region in *Y'***
  - Find decision points
  - Compare their cardinalities
- **Scan *Y'* from the starting point of the essential flip-flop found in *X'* to its end point.**
- **Count the size**
  - *s*: +1
  - *e*: -1
  - Largest partial sum