

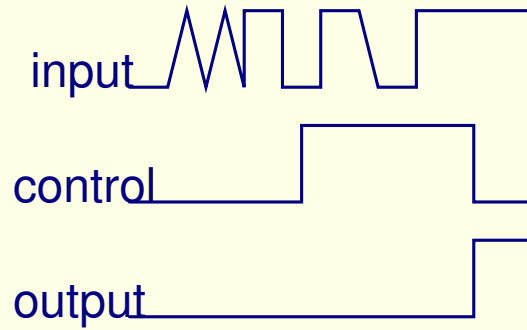
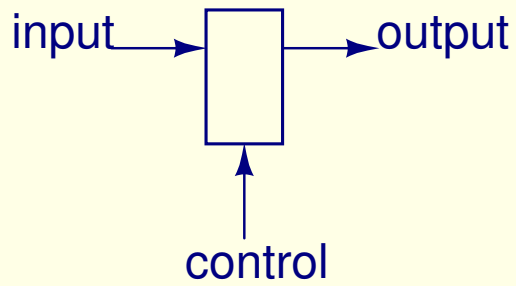
Trade-off between Latch and Flop for Min-Period Sequential Circuit Designs with Crosstalk

Chuan Lin and Hai Zhou

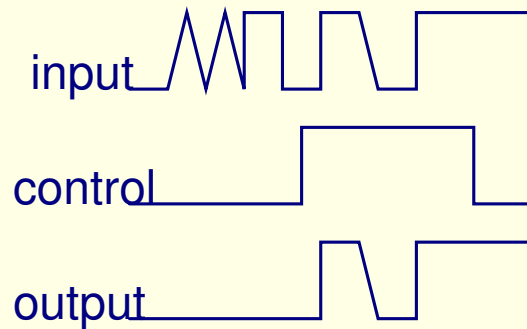
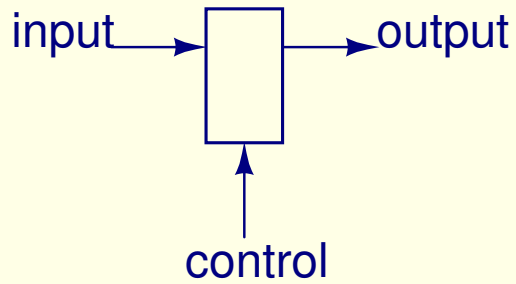
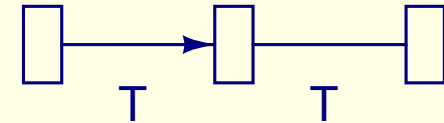
Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

November 8, 2005

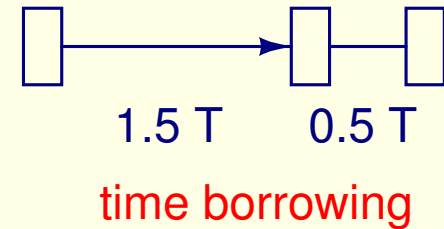
Latch VS Flop



An ideal flop



An ideal latch



Latch VS Flop

- Latch improves clock period by time borrowing

Latch VS Flop

- Latch improves clock period by time borrowing
- Latch has a smaller delay, and occupies less area

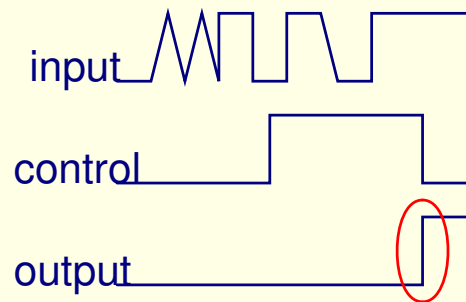
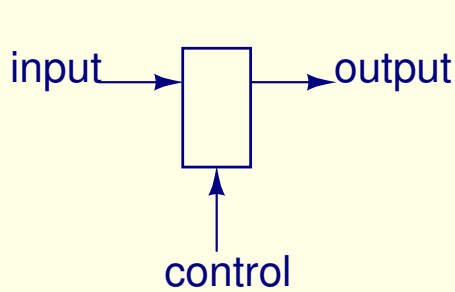
Latch VS Flop

- Latch improves clock period by time borrowing
- Latch has a smaller delay, and occupies less area

Q: Is latch always better ?

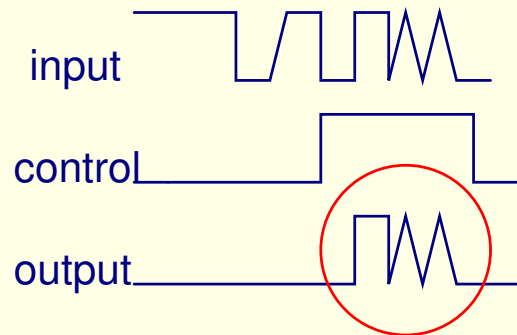
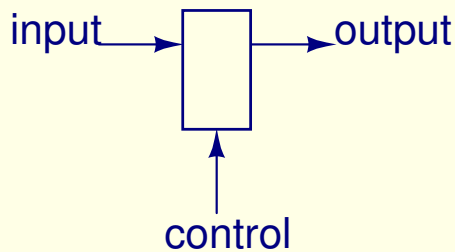
Latch VS Flop

Switching window: all possible switching times



output window is a single point

An ideal flop

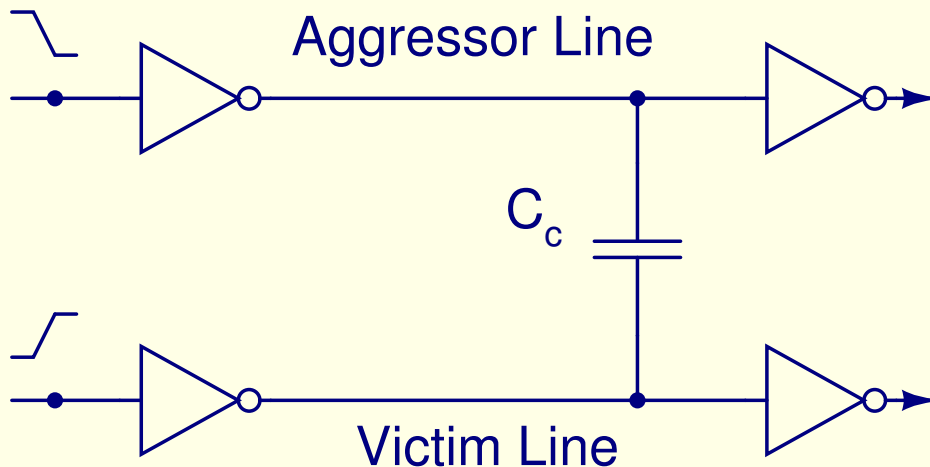


output window may be as wide as active duration

latch cannot filter out noises and glitches

An ideal latch

Crosstalk effect on timing



- Simultaneous switching in **opposite** directions
 - $C_{eff} > C_c$
 - victim slows down
 - set-up violation
- Simultaneous switching in **same** direction
 - $C_{eff} < C_c$
 - victim speeds up
 - hold violation

Latch VS Flop

- Latch improves period by time borrowing
- Latch has a smaller delay, and occupies less area

Q: Is latch always better ?

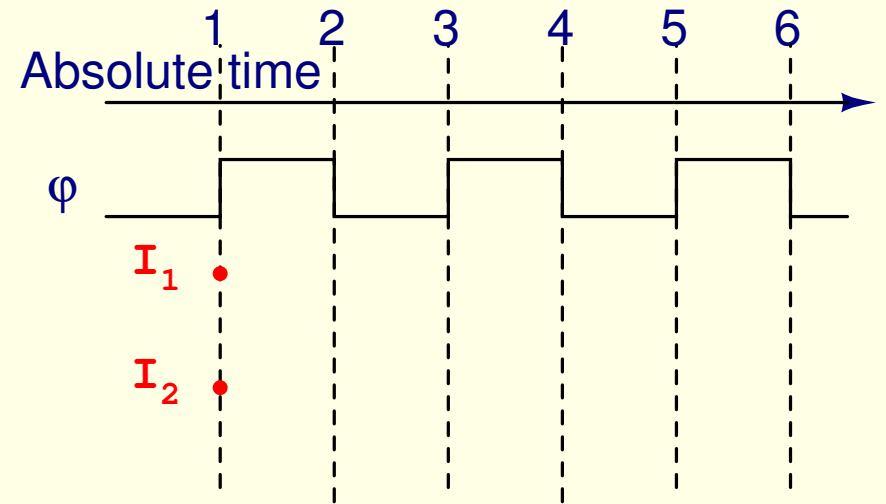
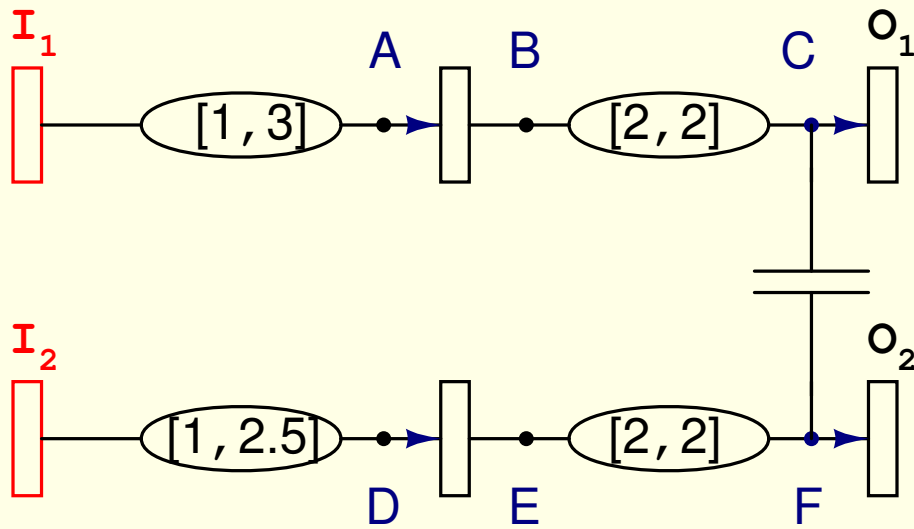
A: Latch could be **worse** with crosstalk !

(more switchings, higher risk of being attacked)

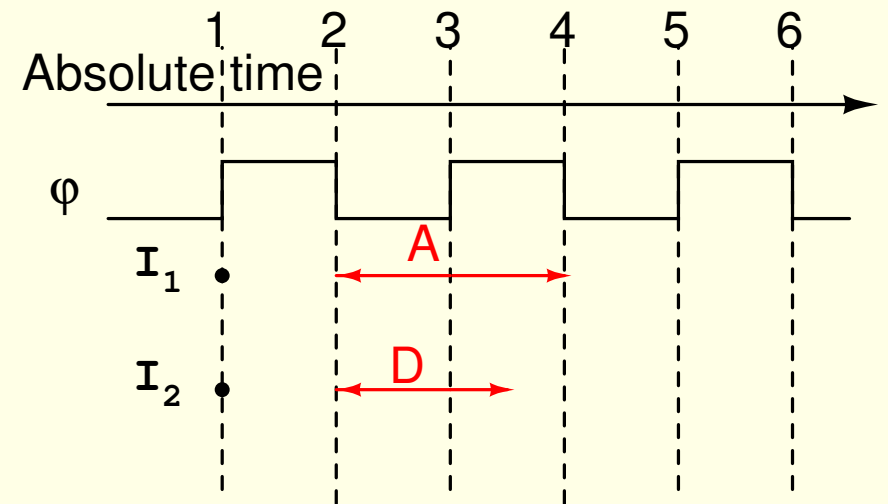
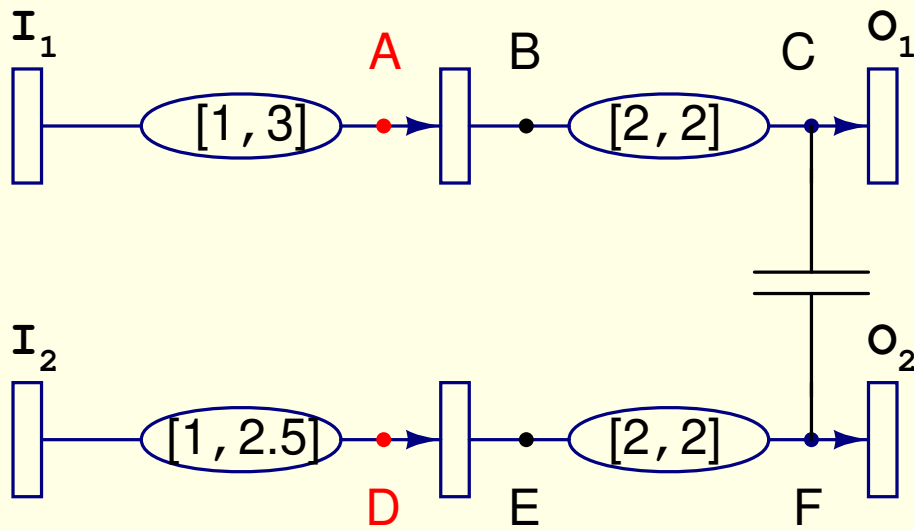
Outline

- Motivation example
- Problem formulation
- Clock verification under circular time representation
- Optimal latch-flop configuration algorithm
- Results and conclusions

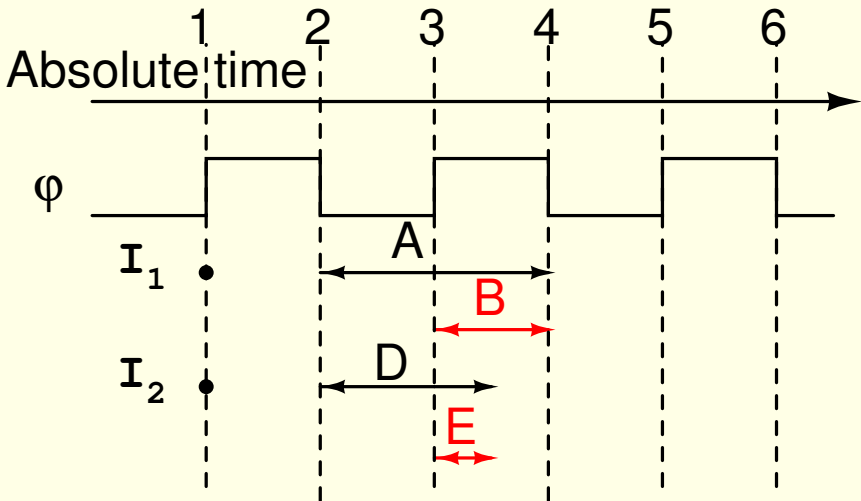
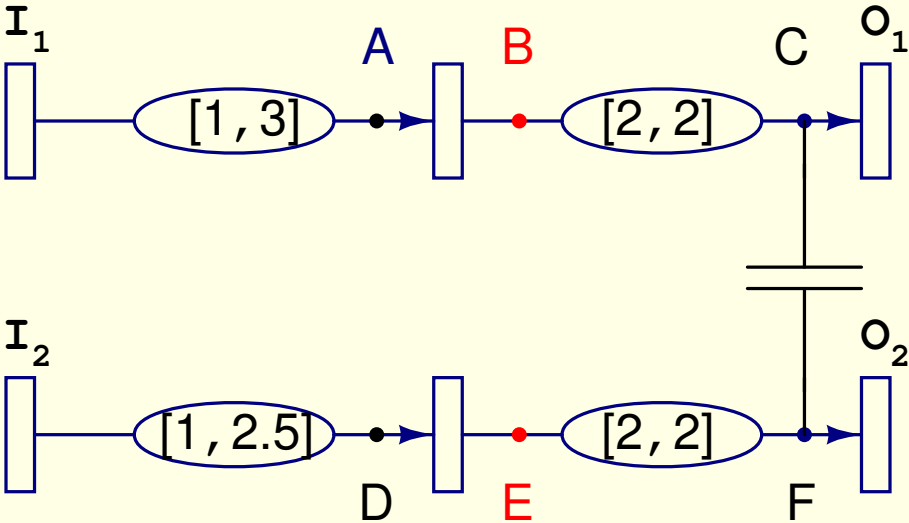
Motivation example



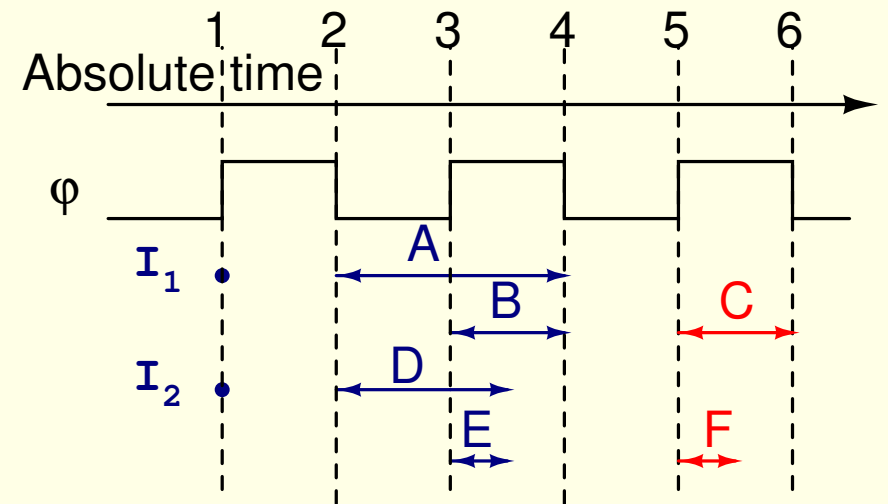
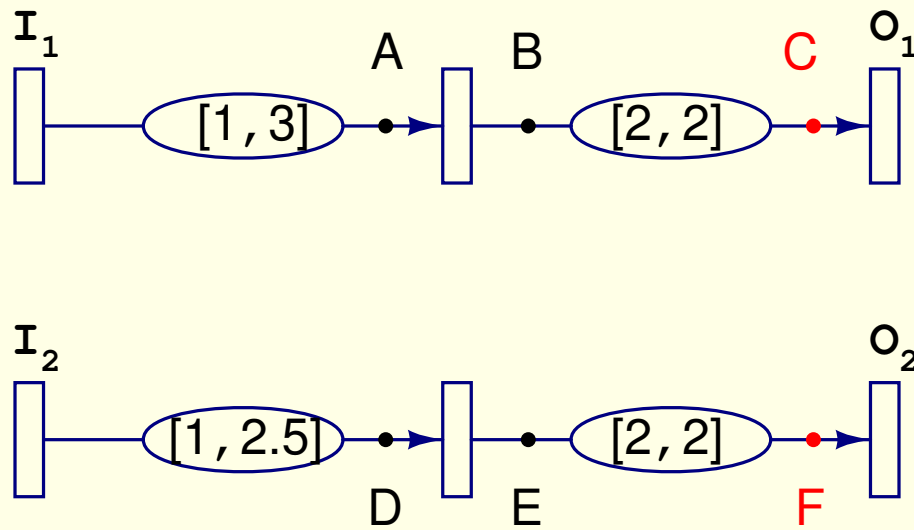
Motivation example



Motivation example

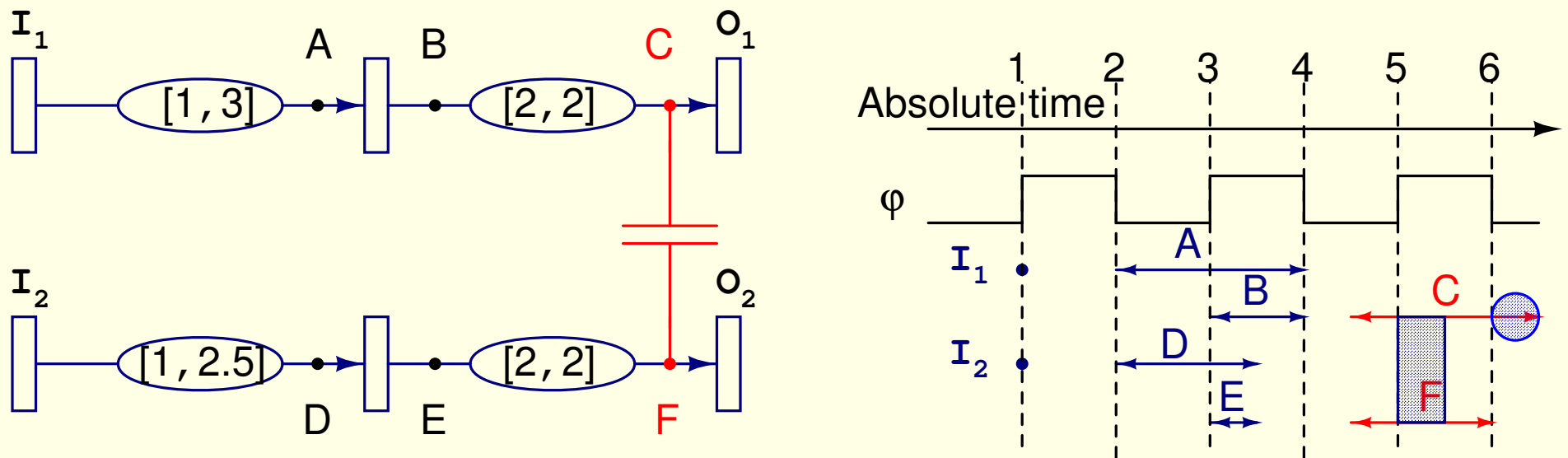


Motivation example



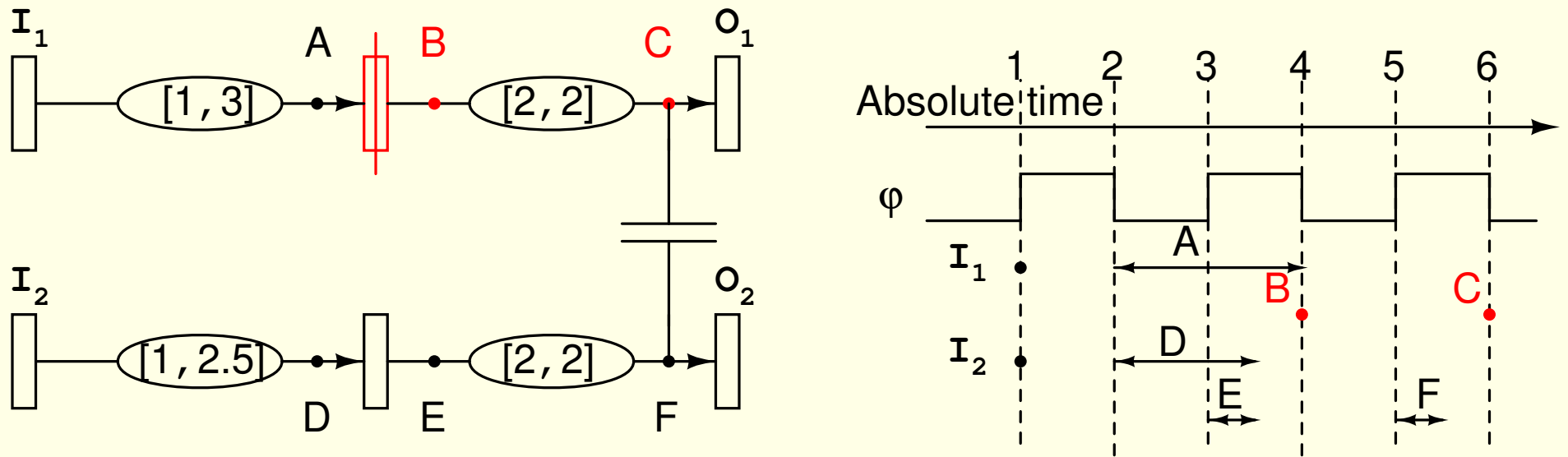
- No clocking violation without crosstalk

Motivation example



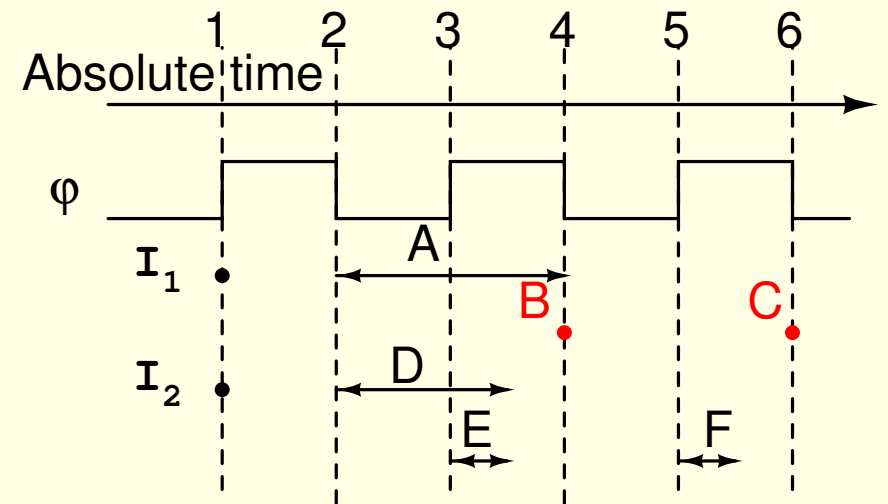
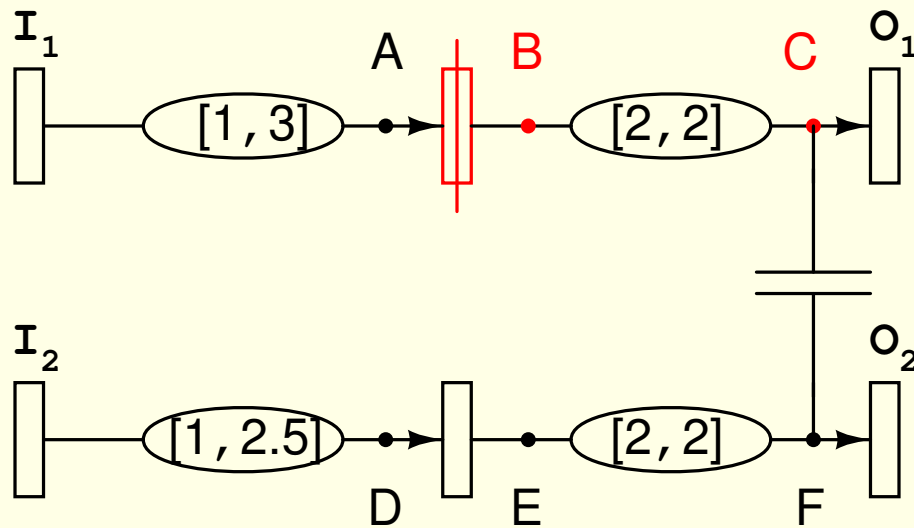
- A setup violation at latch O_1

Motivation example



- No clocking violation with crosstalk

Motivation example



Q: Why using flop is helpful intuitively ?

Motivation example


- Reduces uncertainty of output window
- May result in a delay decrease on a long path
 - Fix a setup violation


Motivation example

- Reduces uncertainty of output window
- May result in a delay decrease on a long path
 - Fix a setup violation
- May result in a delay increase on a short path
 - Fix a hold violation

Minimal period under different configurations


without
crosstalk


 all flops

 all latches



with
crosstalk

 all flops

 all latches

 mixed flops & latches

Problem formulation

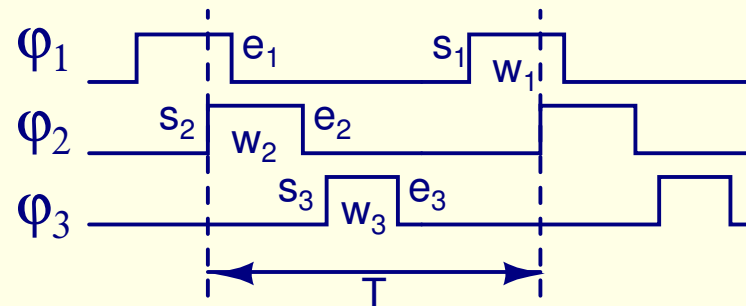
[Optimal Latch-Flop Configuration]

- $G = (V, E)$, $V = \text{gate } V_G \cup \text{memory element } V_L$,
 $E = \text{interconnect } E_I \cup \text{coupling capacitor } E_C$

Problem formulation

[Optimal Latch-Flop Configuration]

- $G = (V, E)$, $V = \text{gate } V_G \cup \text{memory element } V_L$,
 $E = \text{interconnect } E_I \cup \text{coupling capacitor } E_C$
- A clock schedule: ϕ_1, \dots, ϕ_n with a common period T



Problem formulation

[Optimal Latch-Flop Configuration]

- $G = (V, E)$, $V = \text{gate } V_G \cup \text{memory element } V_L$,
 $E = \text{interconnect } E_I \cup \text{coupling capacitor } E_C$
- A clock schedule: ϕ_1, \dots, ϕ_n with a common period T
- Find a configuration of mixed latches and flops
 - No clocking violations
 - Period minimized (clock scales proportionally with T)

Strategy of solving the problem

- **[Fixed Period Latch-Flop Configuration]**
 - Determine if a latch-flop configuration exists under a fixed period T with no clocking violations
 - * Yes $\rightarrow T$ is feasible
 - * No $\rightarrow T$ is infeasible

Strategy of solving the problem

- **[Fixed Period Latch-Flop Configuration]**

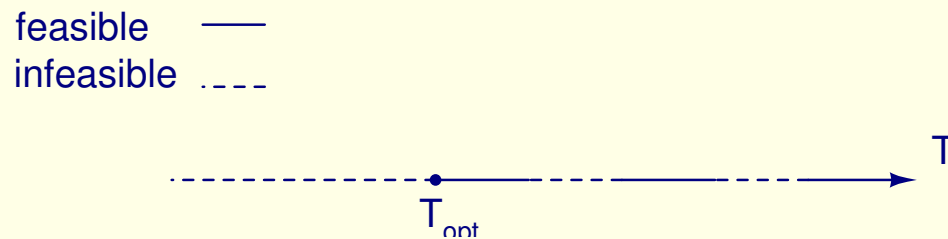
- Determine if a latch-flop configuration exists under a fixed period T with no clocking violations

- * Yes $\rightarrow T$ is feasible

- * No $\rightarrow T$ is infeasible

- **Incremental search for the optimal T**

- Solution space not convex [Hassoun TCAD'03]



Strategy of solving the problem

- **[Fixed Period Latch-Flop Configuration]**
 - Determine if a latch-flop configuration exists under a fixed period T with **no clocking violations**
 - * Yes $\rightarrow T$ is feasible
 - * No $\rightarrow T$ is infeasible
 - **How to verify that a given configuration has no clocking violation ? [Hassoun TCAD'03]**
 - **How to determine that a configuration with no clocking violation exists ? [Open]**

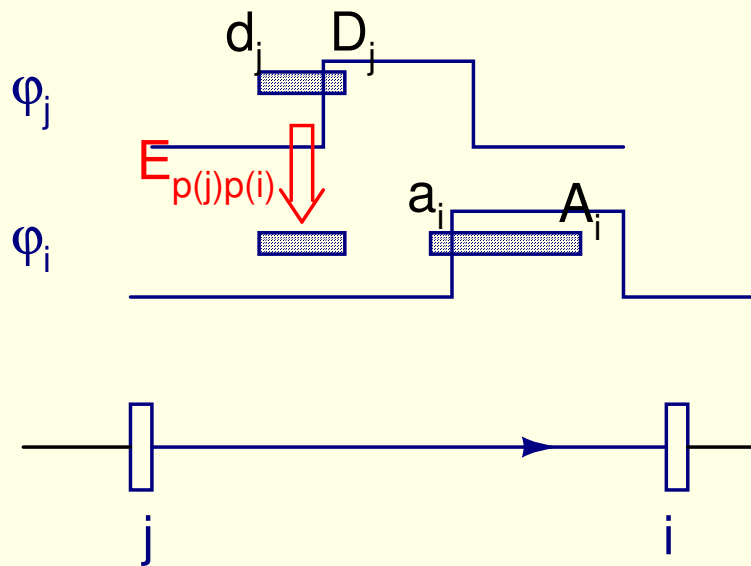
Strategy of solving the problem

- **[Fixed Period Latch-Flop Configuration]**
 - Determine if a latch-flop configuration exists under a fixed period T with **no clocking violations**
 - * Yes $\rightarrow T$ is feasible
 - * No $\rightarrow T$ is infeasible
 - **How to verify that a given configuration has no clocking violation ? [Hassoun TCAD'03]** \leftarrow IMPROVE IT
 - **How to determine that a configuration with no clocking violation exists ? [Open]** \leftarrow GIVE A HEURISTIC

Clock verification

- Compute windows at memory elements [SMO ICCAD'90]

- phase: $p(i)$
- input window: $[a_i, A_i]$ w.r.t. $p(i)$
- output window: $[d_i, D_i]$ w.r.t. $p(i)$



$$D_i = T, \quad \forall \text{ flop } i$$

$$d_i = T, \quad \forall \text{ flop } i$$

$$D_i = \max(A_i, T - w_{p(i)}), \quad \forall \text{ latch } i$$

$$d_i = T - w_{p(i)}, \quad \forall \text{ latch } i$$

$$A_i = \max_{j \rightarrow i} (D_j - E_{p(j)p(i)} + delay_{max})$$

$$a_i = \min_{j \rightarrow i} (d_j - E_{p(j)p(i)} + delay_{min})$$

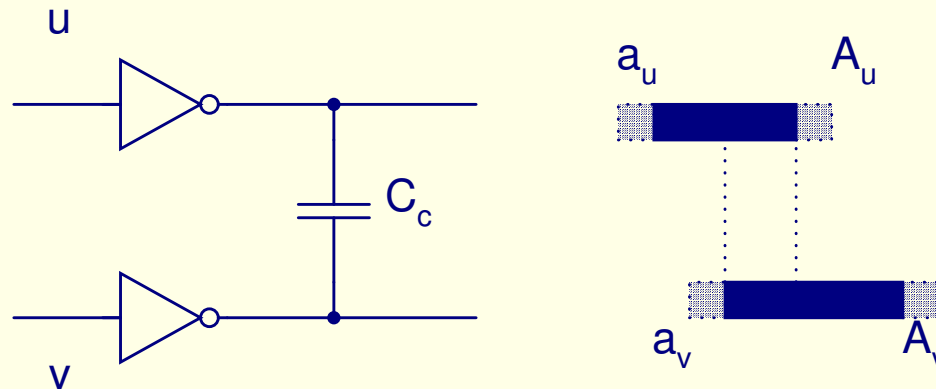
Clock verification

- Compute windows at memory elements [SMO ICCAD'90]
- Compute windows at gates [Hassoun TCAD'03]
 - virtual phases to gates

$$a_v = \begin{cases} \min_{(u,v) \in E_I} (d_u - E_{p(u)p(v)}), & \text{if } p(u) \neq p(v) \\ \min_{(u,v) \in E_I} d_u, & \text{if } p(u) = p(v) \end{cases}$$
$$A_v = \begin{cases} \max_{(u,v) \in E_I} (D_u - E_{p(u)p(v)}), & \text{if } p(u) \neq p(v) \\ \max_{(u,v) \in E_I} D_u, & \text{if } p(u) = p(v) \end{cases}$$
$$d_v = a_v + \delta_v$$
$$D_v = A_v + \Delta_v$$

Clock verification

- Compute windows at memory elements [SMO ICCAD'90]
- Compute windows at gates [Hassoun TCAD'03]
 - virtual phases to gates
- Compute crosstalk effect [Sapatnekar TCAD'00]
 - detect overlap



Clock verification

- **Compute windows at memory elements [SMO ICCAD'90]**
- **Compute windows at gates [Hassoun TCAD'03]**
 - virtual phases to gates
- **Compute crosstalk effect [Sapatnekar TCAD'00]**
 - detect overlap
- **Iterate the above three [Hassoun TCAD'03]**

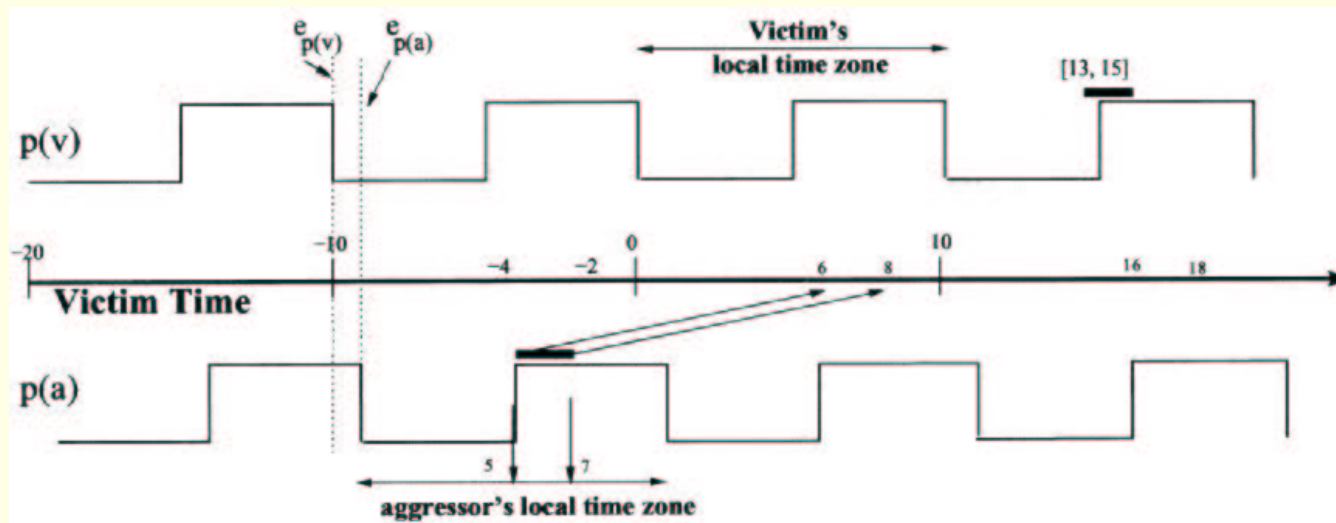
Clock verification

- Compute windows at memory elements [SMO ICCAD'90]
- Compute windows at gates [Hassoun TCAD'03]
 - virtual phases to gates
- Compute crosstalk effect [Sapatnekar TCAD'00]
 - detect overlap
- Iterate the above three [Hassoun TCAD'03]
- Check violations [SMO ICCAD'90]
 - setup violation: $(A_i > T - setup)$
 - hold violation: $(a_i < hold)$

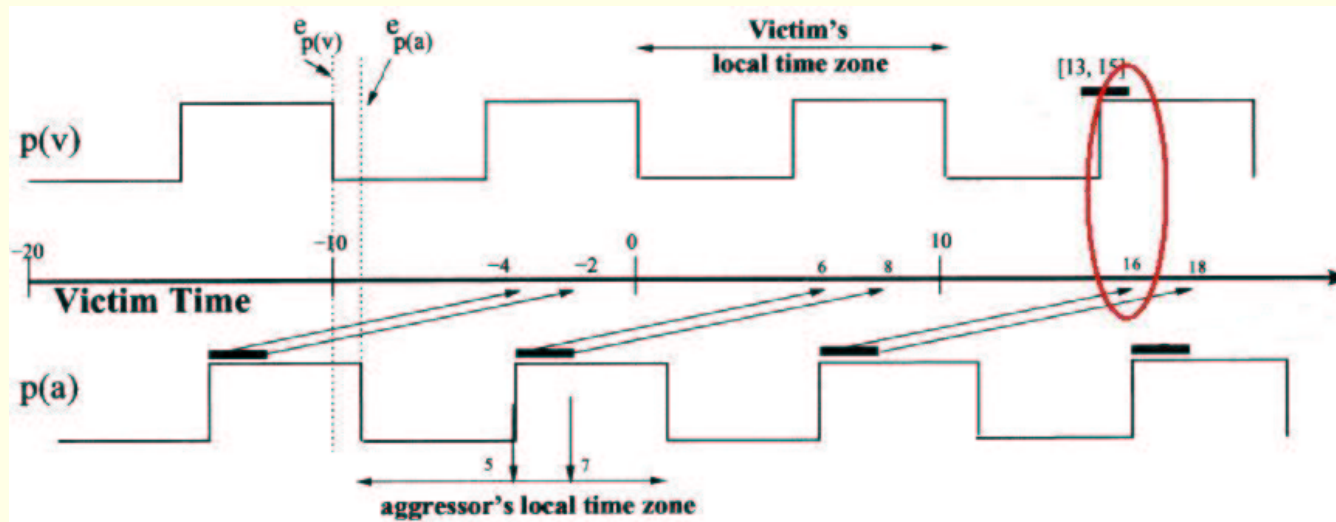
Clock verification

- Compute windows at memory elements [SMO ICCAD'90]
- Compute windows at gates [Hassoun TCAD'03]
 - virtual phases to gates
- Compute crosstalk effect [Sapatnekar TCAD'00]
 - **detect overlap**
- Iterate the above three [Hassoun TCAD'03]
- Check violations [SMO ICCAD'90]
 - setup violation: $(A_i > T - setup)$
 - hold violation: $(a_i < hold)$

Detecting overlap

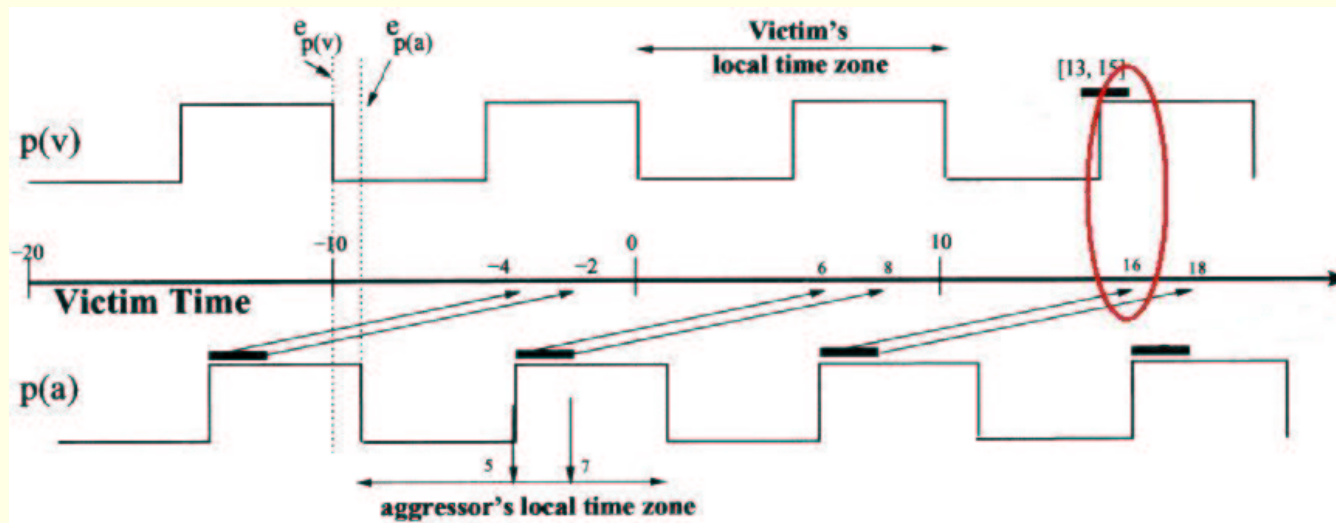


Detecting overlap



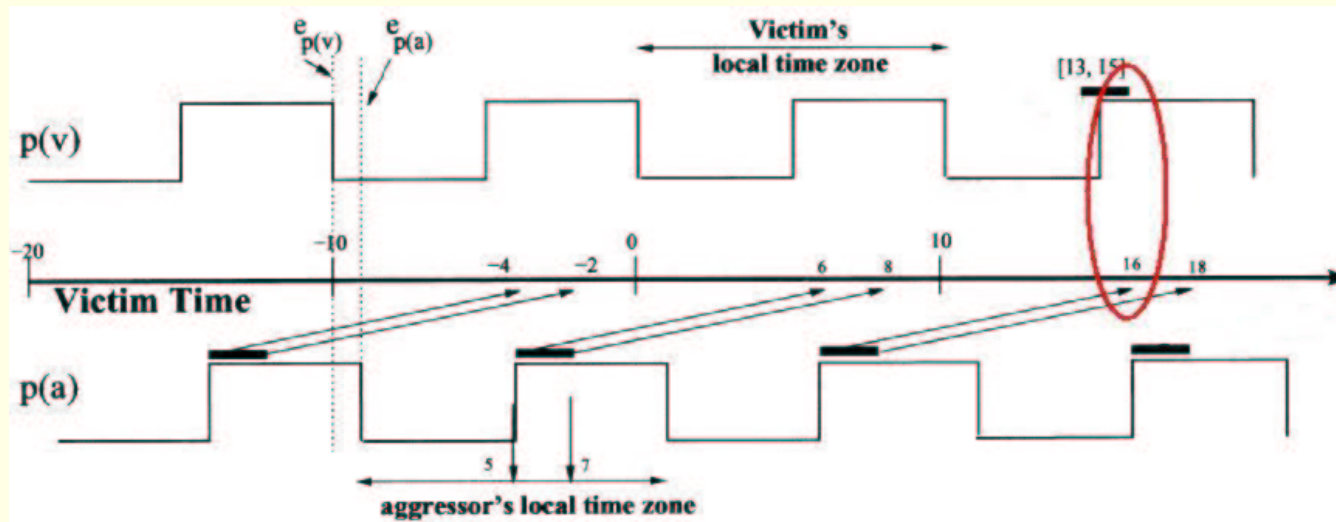
- Translation + Comparing more windows [Hassoun TCAD'03]

Detecting overlap



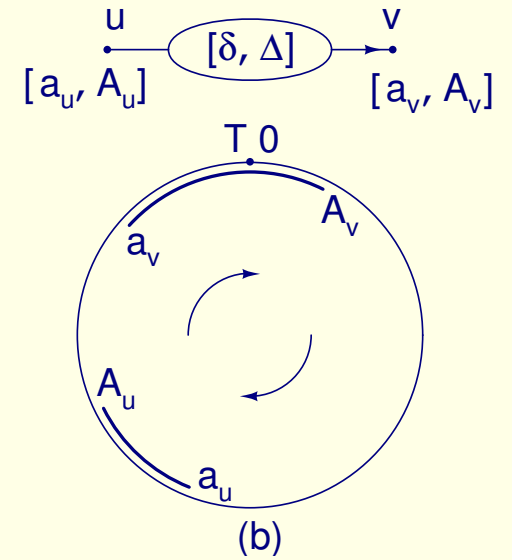
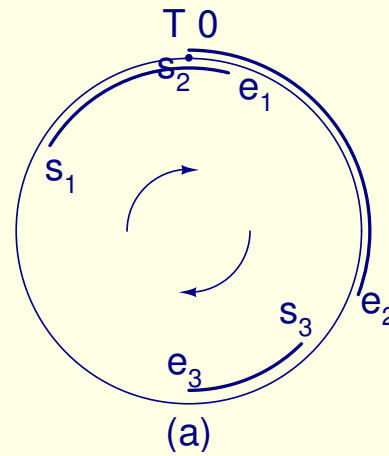
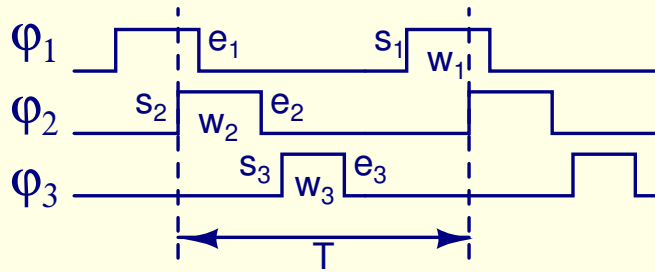
- Translation + Comparing more windows [Hassoun TCAD'03]
- Our idea
 - Take “modulo T ” — **easy detection !**

Detecting overlap

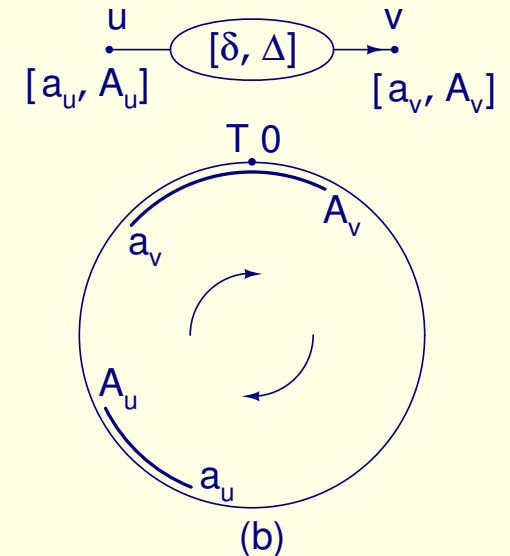
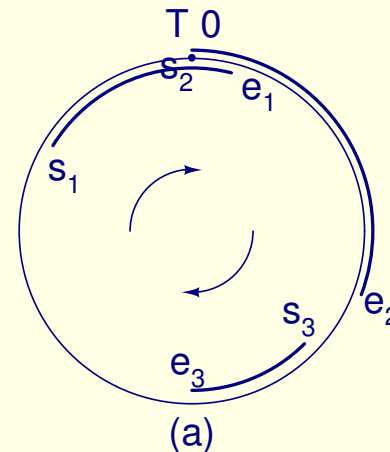
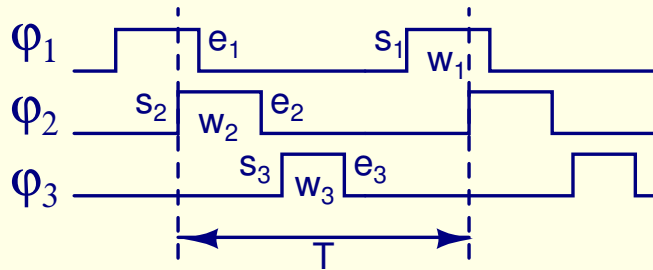


- Translation + Comparing more windows [Hassoun TCAD'03]
- **Our idea**
 - Take “modulo T ” — **easy detection !**
 - Use global time reference — **no phase translation !**

Circular time representation



Circular time representation



Theorem *Two vertices couple if and only if their switching windows (w.r.t. global time reference) overlap within a specified amount of time after being taken modulo T .*

Clock schedule verification with crosstalk under circular time representation

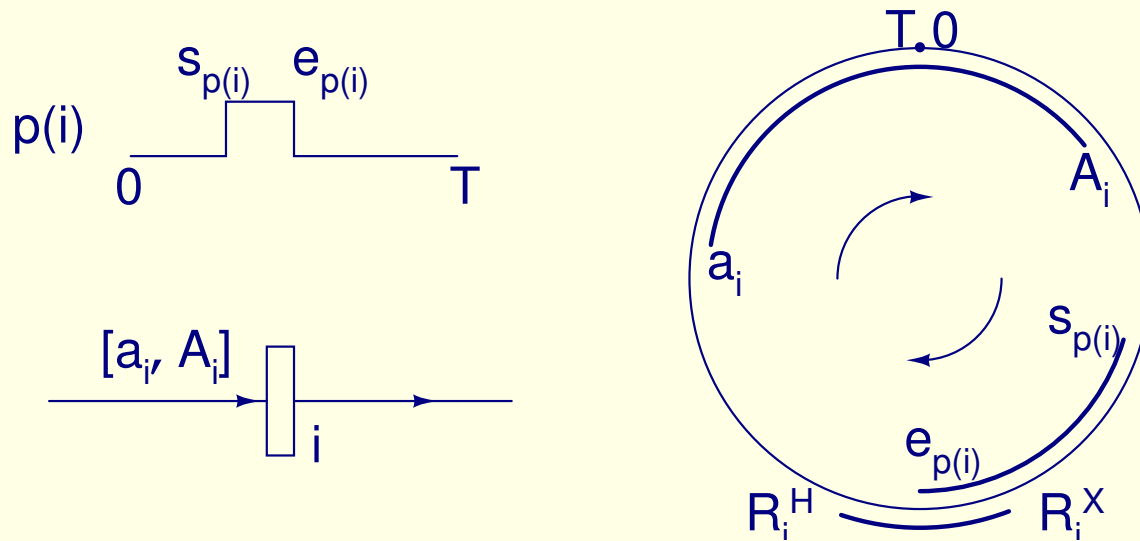
- **Phase 1**

- Verify without crosstalk, terminate if violation exists
- Compute switching windows at gates
- Translate to global time reference
- Take modulo T

- **Phase 2**

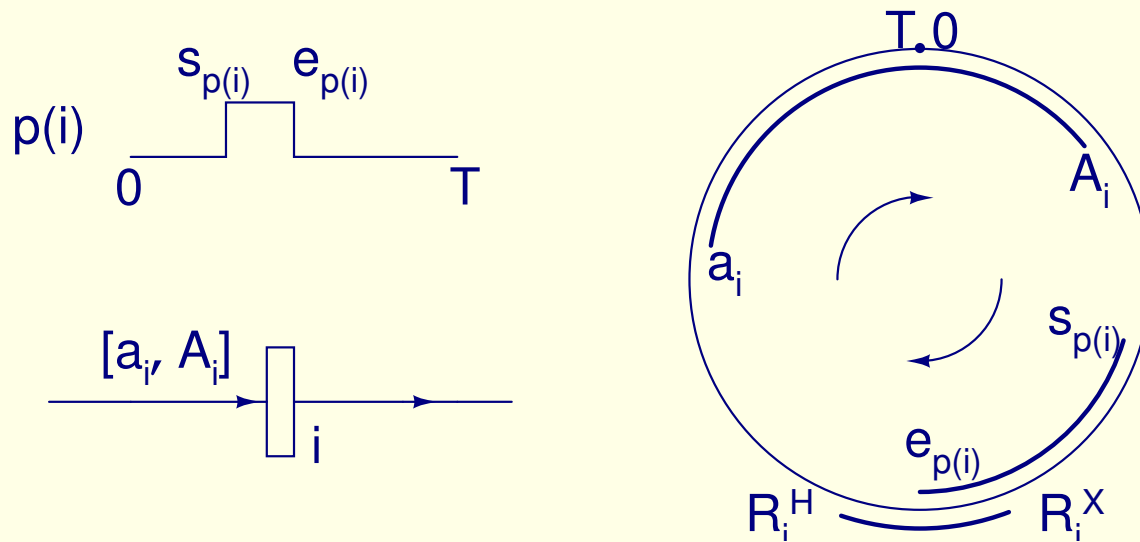
- Detect coupling under circular time representation
- Update switching windows iteratively

Checking violation becomes overlap detection



$$[R_i^X, R_i^H] = [(e_{p(i)} - X_{p(i)}) \bmod T, (e_{p(i)} + H_{p(i)}) \bmod T]$$

Checking violation becomes overlap detection



$$[R_i^X, R_i^H] = [(e_{p(i)} - X_{p(i)}) \bmod T, (e_{p(i)} + H_{p(i)}) \bmod T]$$

Theorem Given a valid schedule without crosstalk, it is also valid with crosstalk if and only if the input switching arc $[a_i, A_i]$ of each memory element $i \in V_L$ does not intersect with $[R_i^X, R_i^H]$ anytime before convergence.

How to determine that a configuration with no violations exists ? [Open] \Leftarrow by construction

Input: A fixed period T

Output: (In)feasibility

Treat all memory elements as latches;

Run clock verification;

While a violation exists

 If replacing a latch with a flop can reduce the violation

 Replace the latch with a flop;

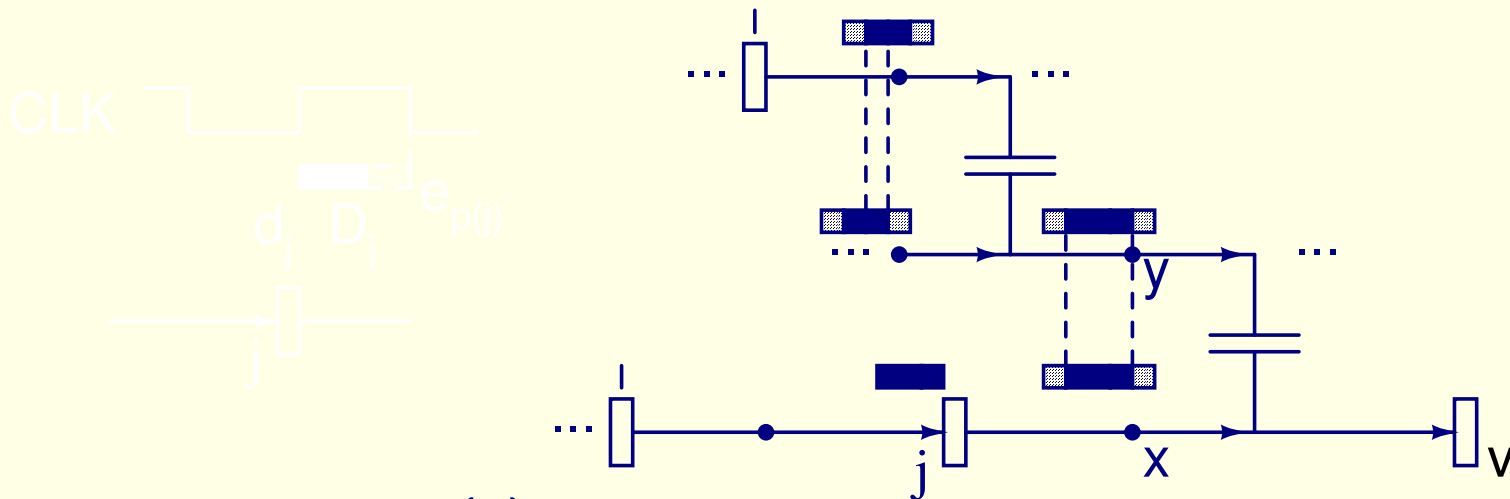
 Run clock verification for the new configuration;

 Else

 Return infeasibility;

Return feasibility;

Identifying the latch for replacement

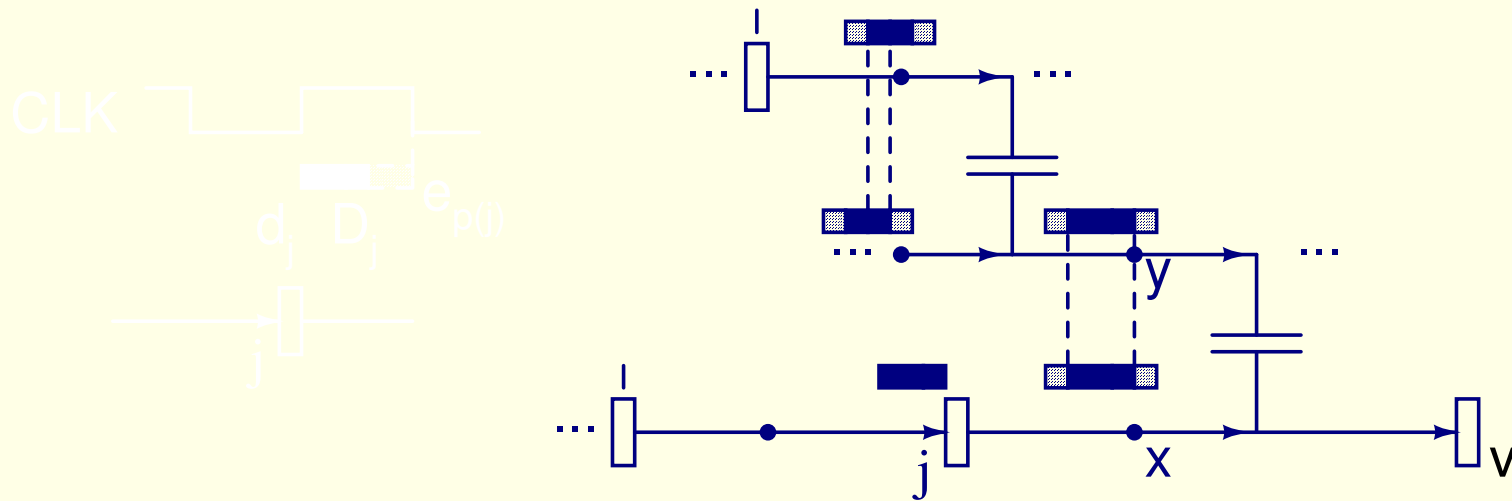


- Source graph $sG(v)$ of $v \in V$

$$sG(v) = \begin{cases} \emptyset, & \text{if } v \in V_L \text{ and } d_v = D_v \\ sG(u) \cup \{u, (u, v)\}, & \text{if } (u, v) \in E_I \text{ or they couple} \end{cases}$$

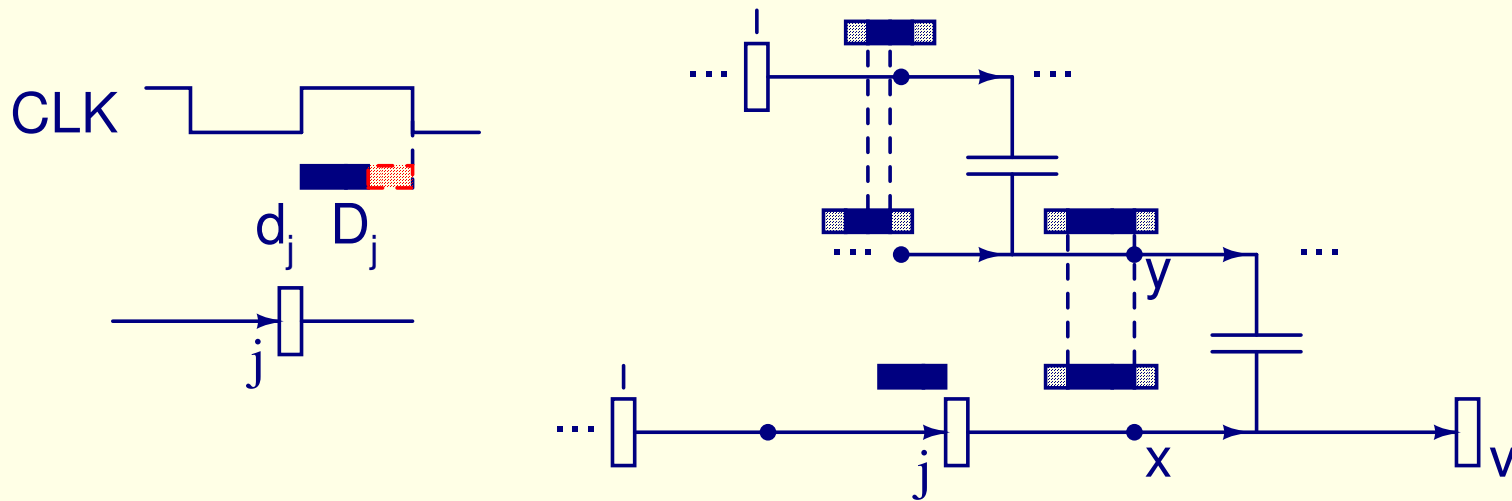
- Gates, interconnects, effective capacitors, memory elements with non-zero-width output windows
- Until memory elements with zero-width output windows

Identifying the latch for replacement



Theorem *At least one latch in the source graph of i has to be replaced with a flop to remove the clocking violation at i .*

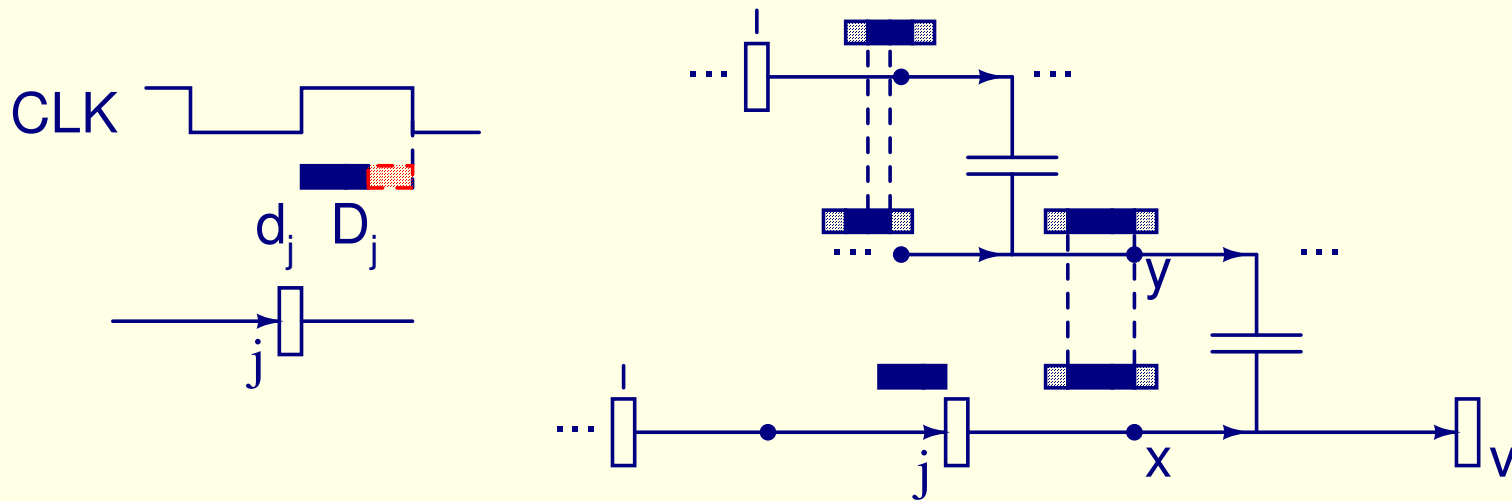
Identifying the latch for replacement



Theorem *At least one latch in the source graph of i has to be replaced with a flop to remove the clocking violation at i .*

- Violation can be removed
 - Choose the one with least change in its output window

Identifying the latch for replacement



Theorem *At least one latch in the source graph of i has to be replaced with a flop to remove the clocking violation at i .*

- Violation can be removed
 - Choose the one with least change in its output window
- Violation cannot be removed
 - Choose the one that gives the minimum violation

Bounds for incremental search

[Optimal Latch-Flop Configuration Problem]

- Fixed Period Latch-Flop Configuration
 - **Solved !**
- Incremental search
 - What are **lower and upper bounds** ?

Bounds for incremental search

- Shenoy and Brayton [ICCAD'92]

$$e_{p(i)} \geq e_{p(j)} - w_{p(j)} + C_{j,i}^q - K_{j,i}^q T + X_{p(i)}, \quad \forall \text{ latch } j \quad (1)$$

$$e_{p(i)} \leq e_{p(j)} + c_{j,i}^q + (1 - K_{j,i}^q)T - H_{p(i)}, \quad \forall \text{ flop } j \quad (2)$$

- $C_{j,i}^q$ ($c_{j,i}^q$) is max (min) delay along path $q : j \rightsquigarrow i$
- $K_{j,i}^q$ counts the number of occurrences that two consecutive memory elements x and y on q have $e_{p(x)} \geq e_{p(y)}$

Bounds for incremental search

- Shenoy and Brayton [ICCAD'92]

$$e_{p(i)} \geq e_{p(j)} - w_{p(j)} + C_{j,i}^q - K_{j,i}^q T + X_{p(i)}, \quad \forall \text{ latch } j \quad (1)$$

$$e_{p(i)} \leq e_{p(j)} + c_{j,i}^q + (1 - K_{j,i}^q)T - H_{p(i)}, \quad \forall \text{ flop } j \quad (2)$$

- $C_{j,i}^q$ ($c_{j,i}^q$) is max (min) delay along path $q : j \rightsquigarrow i$
 - $K_{j,i}^q$ counts the number of occurrences that two consecutive memory elements x and y on q have $e_{p(x)} \geq e_{p(y)}$
- T_l is the minimal period that satisfies (1) for all q .
 T_u is the maximal period that satisfies (2) for all q .

Bounds for incremental search

- Shenoy and Brayton [ICCAD'92]

$$e_{p(i)} \geq e_{p(j)} - w_{p(j)} + C_{j,i}^q - K_{j,i}^q T + X_{p(i)}, \quad \forall \text{ latch } j \quad (1)$$

$$e_{p(i)} \leq e_{p(j)} + c_{j,i}^q + (1 - K_{j,i}^q)T - H_{p(i)}, \quad \forall \text{ flop } j \quad (2)$$

- $C_{j,i}^q$ ($c_{j,i}^q$) is max (min) delay along path $q : j \rightsquigarrow i$
- $K_{j,i}^q$ counts the number of occurrences that two consecutive memory elements x and y on q have $e_{p(x)} \geq e_{p(y)}$

- T_l is the minimal period that satisfies (1) for all q .
 T_u is the maximal period that satisfies (2) for all q .

Theorem *If $0 < T_l \leq T_u < \infty$, then the minimal period that can be possibly achieved with crosstalk is within interval $[T_l, T_u]$.*

Computational complexity

- **Feasibility checking under a given period:**

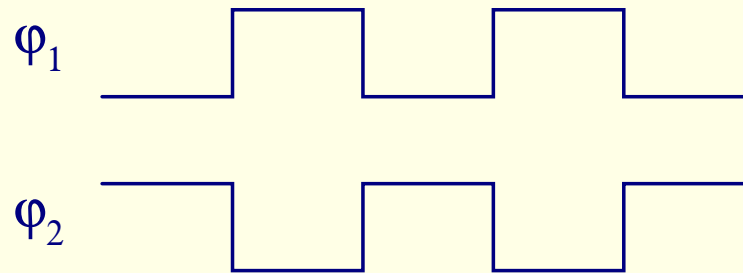
$$O(V_L E_I + V_L^3 + E_C E_I + V_L V_G)$$

- **Entire algorithm:**

$$O\left((V_L E_I + V_L^3 + E_C E_I + V_L V_G) \frac{T_u - T_l}{\epsilon}\right)$$

Experimental setting

- Benchmark: ISCAS-89
 - Gates: $0.5 \leq delay_{max} \leq 2.5$, $delay_{max} - 0.5 \leq delay_{min}$
 - Capacitors: $10\% |V|$, $0 \leq delay \leq 3$
- Convert flops to back-to-back latches
 - Retiming tool ASTRA [Sapatnekar and Deokar TCAD'96]
 - Two-phase symmetric clock schedule



- Search step: $\epsilon = 0.01$

Optimal periods

Circuit	T_0	T_{latch}	T_{mix}	impr%	#veri	t(sec)
s386	21.84	24.08	24.08	0.0%	1228	0.31
s838	34.28	35.78	34.28	100.0%	10	0.00
s1196	53.48	55.63	55.39	11.2%	7275	3.87
s1488	34.07	35.83	35.54	16.5%	398	0.25
s3271	40.31	44.20	43.95	6.4%	3494	8.85
s3330	34.78	36.33	34.78	100.0%	9	0.02
s3384	56.62	58.43	56.62	100.0%	16	0.04
s4863	61.24	63.00	61.60	79.5%	300	1.54
s5378	57.61	61.97	57.61	100.0%	2	0.00
s9234	80.53	84.13	80.53	100.0%	7	0.12
s15850	129.64	133.71	129.64	100.0%	5	0.19
s35932	77.88	84.36	82.23	32.9%	2178	62.04

Optimal periods

Circuit	T_0	T_{latch}	T_{mix}	impr%	#veri	t(sec)
s386	21.84	24.08	24.08	0.0%	1228	0.31
s838	34.28	35.78	34.28	100.0%	10	0.00
s1196	53.48	55.63	55.39	11.2%	7275	3.87
s1488	34.07	35.83	35.54	16.5%	398	0.25
s3271	40.31	44.20	43.95	6.4%	3494	8.85
s3330	34.78	36.33	34.78	100.0%	9	0.02
s3384	56.62	58.43	56.62	100.0%	16	0.04
s4863	61.24	63.00	61.60	79.5%	300	1.54
s5378	57.61	61.97	57.61	100.0%	2	0.00
s9234	80.53	84.13	80.53	100.0%	7	0.12
s15850	129.64	133.71	129.64	100.0%	5	0.19
s35932	77.88	84.36	82.23	32.9%	2178	62.04

- $T_{latch} > T_0$ due to crosstalk: 10.3% for "s386"

Optimal periods

Circuit	T_0	T_{latch}	T_{mix}	impr%	#veri	t(sec)
s386	21.84	24.08	24.08	0.0%	1228	0.31
s838	34.28	35.78	34.28	100.0%	10	0.00
s1196	53.48	55.63	55.39	11.2%	7275	3.87
s1488	34.07	35.83	35.54	16.5%	398	0.25
s3271	40.31	44.20	43.95	6.4%	3494	8.85
s3330	34.78	36.33	34.78	100.0%	9	0.02
s3384	56.62	58.43	56.62	100.0%	16	0.04
s4863	61.24	63.00	61.60	79.5%	300	1.54
s5378	57.61	61.97	57.61	100.0%	2	0.00
s9234	80.53	84.13	80.53	100.0%	7	0.12
s15850	129.64	133.71	129.64	100.0%	5	0.19
s35932	77.88	84.36	82.23	32.9%	2178	62.04

- $T_{latch} > T_0$ due to crosstalk: 10.3% for "s386"
- **Effective:** half of circuits 100% improvement $\left(\frac{T_{latch}-T_{mix}}{T_{latch}-T_0}\right)$

Optimal periods

Circuit	T_0	T_{latch}	T_{mix}	impr%	#veri	t(sec)
s386	21.84	24.08	24.08	0.0%	1228	0.31
s838	34.28	35.78	34.28	100.0%	10	0.00
s1196	53.48	55.63	55.39	11.2%	7275	3.87
s1488	34.07	35.83	35.54	16.5%	398	0.25
s3271	40.31	44.20	43.95	6.4%	3494	8.85
s3330	34.78	36.33	34.78	100.0%	9	0.02
s3384	56.62	58.43	56.62	100.0%	16	0.04
s4863	61.24	63.00	61.60	79.5%	300	1.54
s5378	57.61	61.97	57.61	100.0%	2	0.00
s9234	80.53	84.13	80.53	100.0%	7	0.12
s15850	129.64	133.71	129.64	100.0%	5	0.19
s35932	77.88	84.36	82.23	32.9%	2178	62.04

- $T_{latch} > T_0$ **due to crosstalk:** 10.3% for "s386"
- **Effective:** half of circuits 100% improvement $\left(\frac{T_{latch} - T_{mix}}{T_{latch} - T_0}\right)$
- **Efficient:** 0.03 second/verification for $|V| > 21,000$
30 seconds/verification for $|V| = 4,000$ in [Hassoun TCAD'03]

Change of contributing capacitors and flops

Circuit	c.cap _L	c.cap _{L+F}	latch	flop
s386	13	13	24	0
s838	32	3	210	9
s1196	50	48	156	16
s1488	56	33	72	19
s3271	79	72	544	3
s3330	66	59	557	8
s3384	101	27	689	8
s4863	111	142	512	9
s5378	65	25	844	1
s9234	212	249	990	6
s15850	478	442	1719	4
s35932	523	360	5207	4

- **Marginal overhead:** only a few latches are replaced by flops
($\frac{\#flop}{\#latch} = 0.039$ in average)

Summary

- Coupling detection is easier and more efficient under circular time representation

Summary

- Coupling detection is easier and more efficient under circular time representation
- A heuristic algorithm is presented
 - Finds a latch-flop configuration with minimal period
 - Both effective and efficient

Summary

- Coupling detection is easier and more efficient under circular time representation
- A heuristic algorithm is presented
 - Finds a latch-flop configuration with minimal period
 - Both effective and efficient
- **Extensions**
 - **Other discrete coupling models**
 - **Clock skew uncertainty**
 - **Inductive couplings**

Thank you !