

Optimal Wire Retiming Without Binary Search

Chuan Lin and Hai Zhou

Electrical and Computer Engineering
Northwestern University

Outline

- Motivation
- Problem formulation
- Algorithm
- Experimental results
- Conclusions

Motivation

- Industry trend
 - Frequency: 2X/generation, Die size: 1.25X/generation
 - Problem: global signal communication

Motivation

- Industry trend
 - Frequency: 2X/generation, Die size: 1.25X/generation
 - Problem: global signal communication
- Recent research
 - Insert flip-flops on global wires (Intel, IBM, etc.)

Motivation

- Industry trend
 - Frequency: 2X/generation, Die size: 1.25X/generation
 - Problem: global signal communication
- Recent research
 - Insert flip-flops on global wires (Intel, IBM, etc.)
 - How to keep functional correctness

Motivation

- Industry trend
 - Frequency: 2X/generation, Die size: 1.25X/generation
 - Problem: global signal communication
- Recent research
 - Insert flip-flops on global wires (Intel, IBM, etc.)
 - How to keep functional correctness
 - * Nookala and Sapatnekar [DAC'04]
 - Clock period given, Pipelined circuit given

Motivation

- Industry trend
 - Frequency: 2X/generation, Die size: 1.25X/generation
 - Problem: global signal communication
- Recent research
 - Insert flip-flops on global wires (Intel, IBM, etc.)
 - How to keep functional correctness
 - * Nookala and Sapatnekar [DAC'04]
 - Clock period given, Pipelined circuit given
 - Process rate (throughput \times frequency), Area

Motivation

- Retiming can move flip-flops w/o changing functionality

Motivation

- Retiming can move flip-flops w/o changing functionality
- Flip-flops be retimed to pipeline long wire

Motivation

- Retiming can move flip-flops w/o changing functionality
- Flip-flops be retimed to pipeline long wire
 - Chu, Young, Tong and Dechu [ICCAD'03]
 - * Block-level, MILP approach

Motivation

- Retiming can move flip-flops w/o changing functionality
- Flip-flops be retimed to pipeline long wire
 - Chu, Young, Tong and Dechu [ICCAD'03]
 - * Block-level, MILP approach
 - Lin and Zhou [ICCAD'03]
 - * Chip-level, timing macro models

Motivation

- Retiming can move flip-flops w/o changing functionality
- Flip-flops be retimed to pipeline long wire
 - Chu, Young, Tong and Dechu [ICCAD'03]
 - * Block-level, MILP approach
 - Lin and Zhou [ICCAD'03]
 - * Chip-level, timing macro models
 - Both polynomial time algorithm for feasibility checking

Motivation

- Fully Polynomial Time Approximation Schemes
 - Binary search, hence precision, is required

Motivation

- Fully Polynomial Time Approximation Schemes
 - Binary search, hence precision, is required
 - * Previous computation wasted - **NOT** efficient

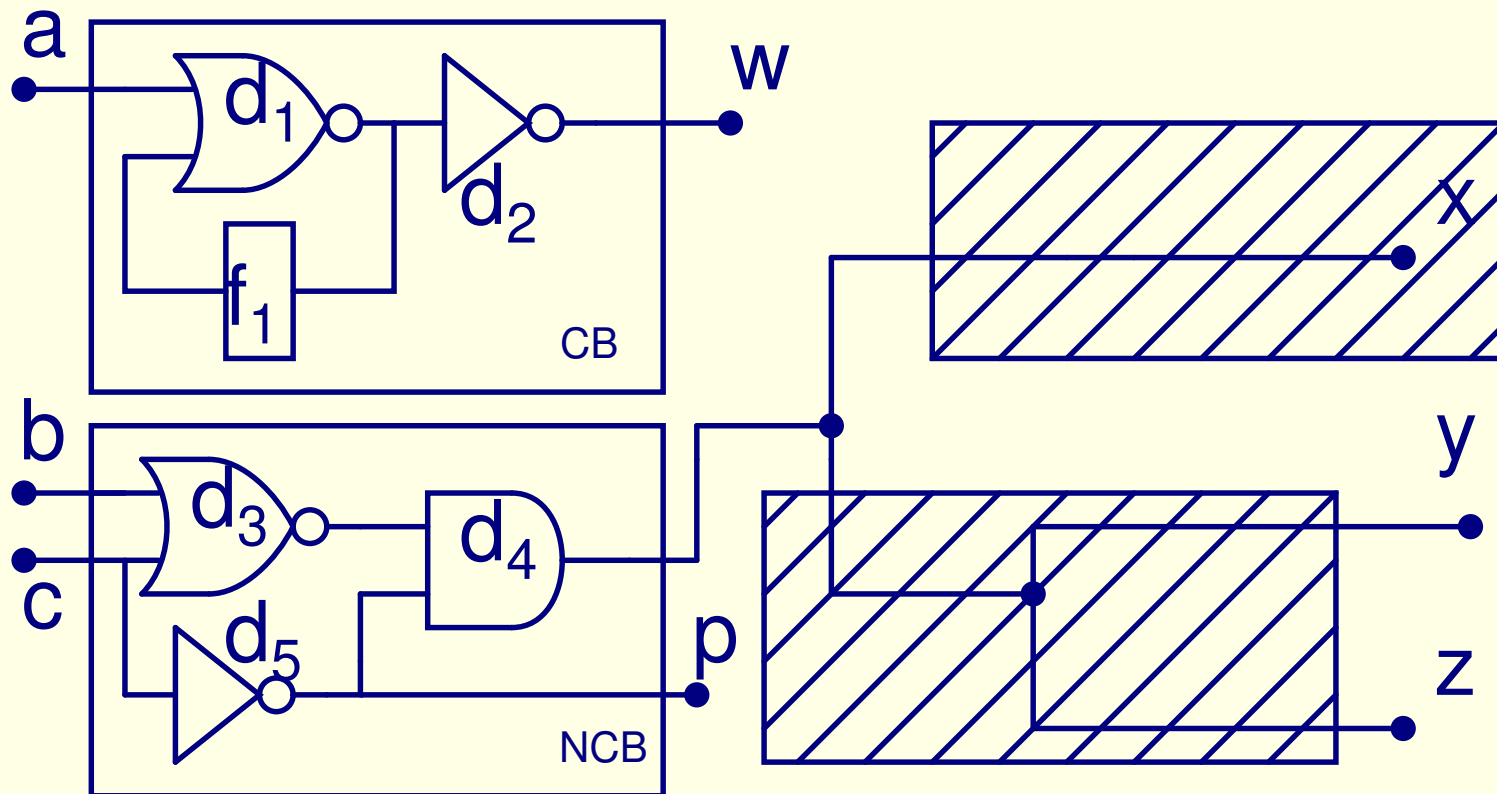
Motivation

- Fully Polynomial Time Approximation Schemes
 - Binary search, hence precision, is required
 - * Previous computation wasted - **NOT** efficient
 - * **NOT** polynomial

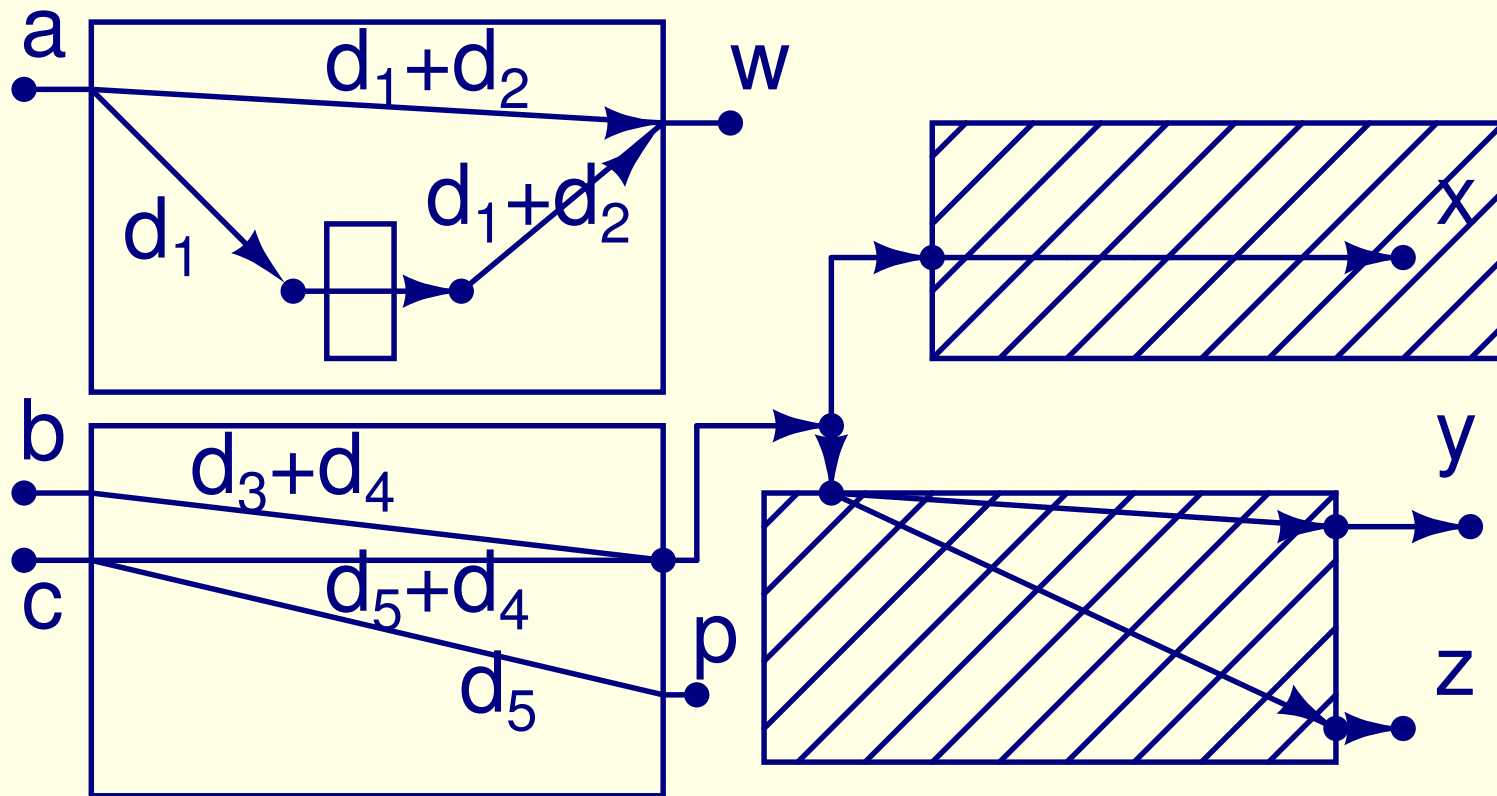
Motivation

- Fully Polynomial Time Approximation Schemes
 - Binary search, hence precision, is required
 - * Previous computation wasted - **NOT** efficient
 - * **NOT** polynomial
- Existence of a polynomial-time approach ?

A specific circuit configuration



The timing model of the circuit



Problem formulation

[Optimal Wire Retiming]

- $G = (V, E)$, $E = E_1 \cup E_2$, $E_1 \cap E_2 = \emptyset$
delay: $d(e)$, #flip-flop: $w(e)$, $\forall e \in E$

Problem formulation

[Optimal Wire Retiming]

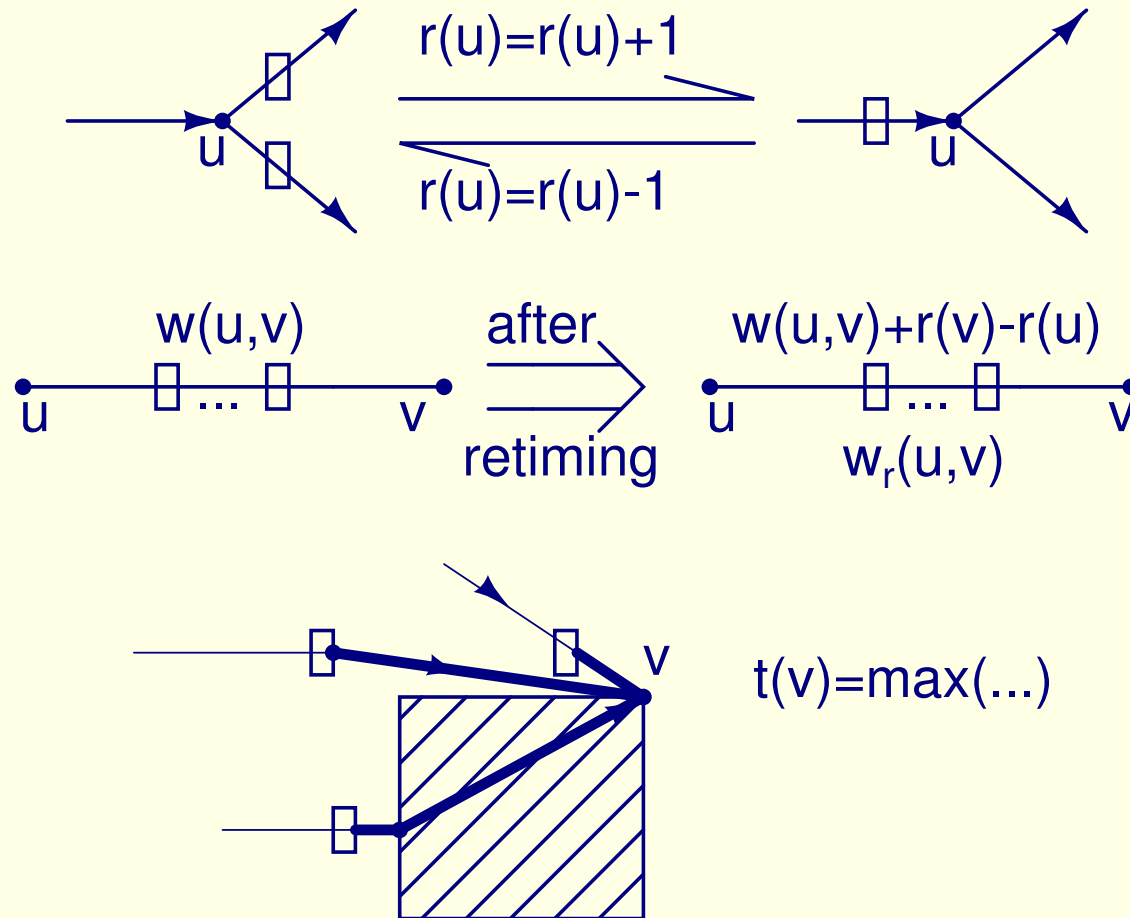
- $G = (V, E)$, $E = E_1 \cup E_2$, $E_1 \cap E_2 = \emptyset$
delay: $d(e)$, #flip-flop: $w(e)$, $\forall e \in E$
- $\forall e \in E_2$, $d(e)$ is proportional to its length
 - Buffers can be inserted to make it linear

Problem formulation

[Optimal Wire Retiming]

- $G = (V, E)$, $E = E_1 \cup E_2$, $E_1 \cap E_2 = \emptyset$
delay: $d(e)$, #flip-flop: $w(e)$, $\forall e \in E$
- $\forall e \in E_2$, $d(e)$ is proportional to its length
 - Buffers can be inserted to make it linear
- Find a relocation of flip-flops
 - No flip-flop change on any $e \in E_1$
 - Minimize the maximum delay between any two consecutive flip-flops(clock period)

Two essential variables r and t



Requirements for r and t

- **Retiming validity**

$$r(u) = r(v), \quad \forall (u, v) \in E_1 \quad (1)$$

$$w_r(u, v) = w(u, v) + r(v) - r(u) \geq 0, \quad \forall (u, v) \in E_2 \quad (2)$$

Requirements for r and t

- **Retiming validity**

$$r(u) = r(v), \quad \forall (u, v) \in E_1 \quad (1)$$

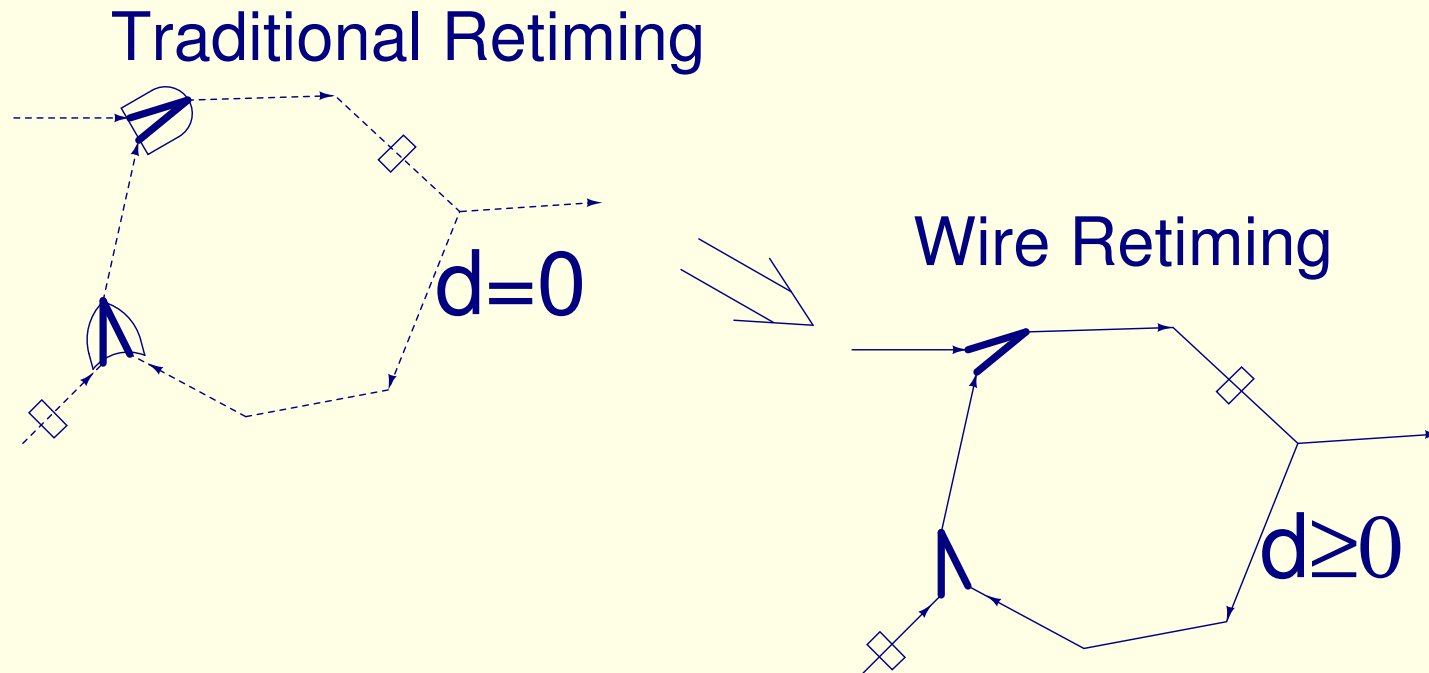
$$w_r(u, v) = w(u, v) + r(v) - r(u) \geq 0, \quad \forall (u, v) \in E_2 \quad (2)$$

- **Timing validity**

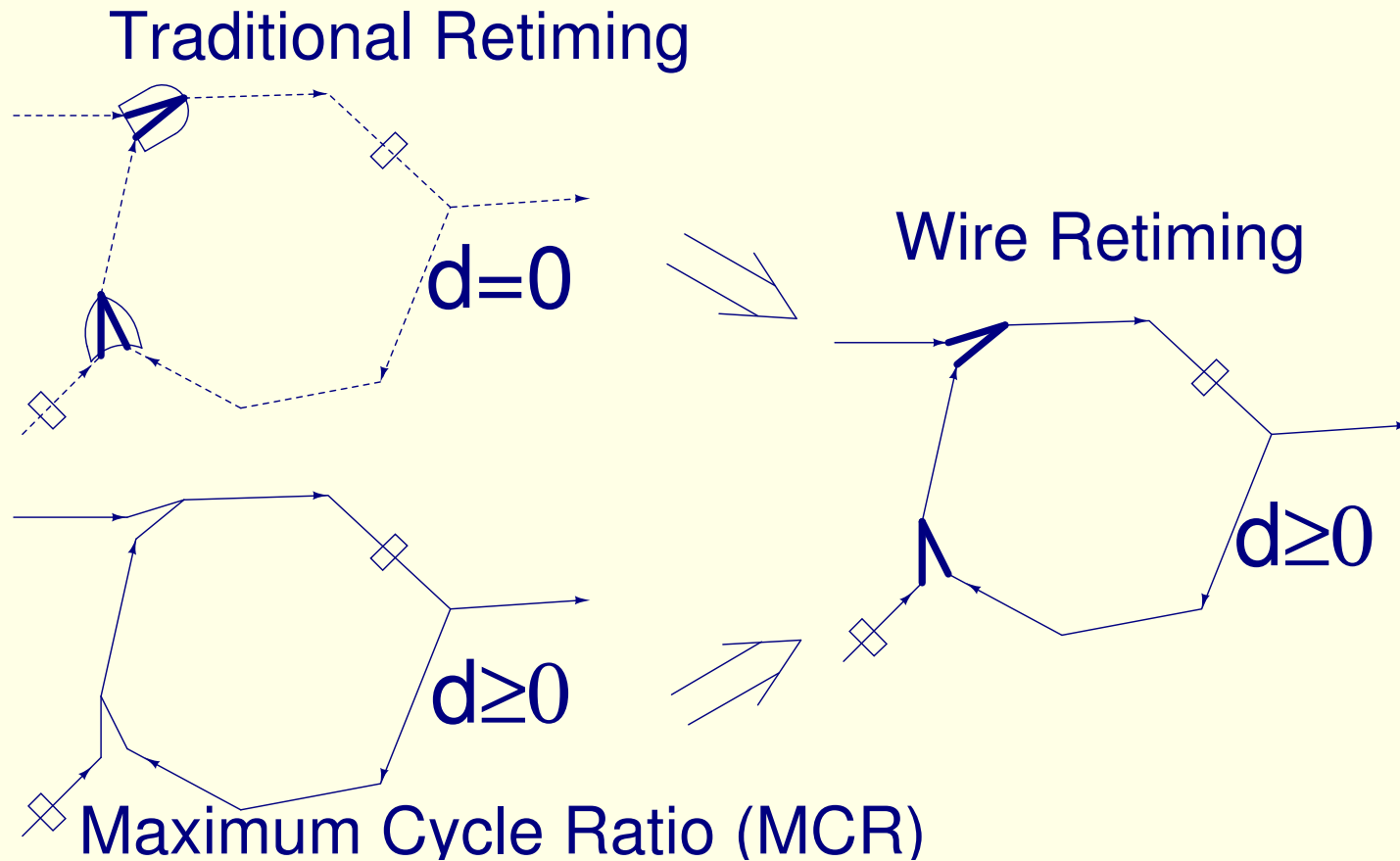
$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

$$t(v) \leq T, \quad \forall v \in V \quad (4)$$

Algorithmic view of the problem



Algorithmic view of the problem



Related algorithms

- Zhou's algorithm [IWLS'04]
 - Solve Traditional Retiming w/o binary search

Related algorithms

- Zhou's algorithm [IWLS'04]
 - Solve Traditional Retiming w/o binary search

$$r(u) = r(v), \quad \forall (u, v) \in E_1 \quad (1)$$

$$w_r(u, v) = w(u, v) + r(v) - r(u) \geq 0, \quad \forall (u, v) \in E_2 \quad (2)$$

$$t(v) \geq t(u) + d(u, v), \quad \forall (u, v) \in \mathbf{E}, w_r(u, v) = 0$$

$$t(v) \leq T, \quad \forall v \in V \quad (4)$$

Related algorithms

- Zhou's algorithm [IWLS'04]
 - Solve Traditional Retiming w/o binary search

$$r(u) = r(v), \quad \forall (u, v) \in E_1 \quad (1)$$

$$w_r(u, v) = w(u, v) + r(v) - r(u) \geq 0, \quad \forall (u, v) \in E_2 \quad (2)$$

$$t(v) \geq t(u) + d(u, v), \quad \forall (u, v) \in \mathbf{E}, w_r(u, v) = 0$$

$$t(v) \leq T, \quad \forall v \in V \quad (4)$$

- Need arrival time info for FFs to tackle Wire Retiming

Related algorithms

- Burns's algorithm [PhD thesis CIT '91]
 - Solve MCR problem in polynomial time

Related algorithms

- Burns's algorithm [PhD thesis CIT '91]
 - Solve MCR problem in polynomial time

$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

Related algorithms

- Burns's algorithm [PhD thesis CIT '91]
 - Solve MCR problem in polynomial time

$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

- Need retiming info to tackle Wire Retiming

Related algorithms

- Burns's algorithm [PhD thesis CIT '91]
 - Solve MCR problem in polynomial time

$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

- Need retiming info to tackle Wire Retiming
- Their combination ?

Strategy of solving the problem

- Start with a feasible T
 - $r(v) = 0, t(v) = 0, \forall v \in V$

Strategy of solving the problem

- Start with a feasible T
 - $r(v) = 0, t(v) = 0, \forall v \in V$
- Gradually push down T with (1)-(4) inviolate

Strategy of solving the problem

- Start with a feasible T
 - $r(v) = 0, t(v) = 0, \forall v \in V$
- Gradually push down T with (1)-(4) inviolate
 - **With r unchanged**
 - * Burns's idea

Strategy of solving the problem

- Start with a feasible T
 - $r(v) = 0, t(v) = 0, \forall v \in V$
- Gradually push down T with (1)-(4) inviolate
 - **With r unchanged**
 - * Burns's idea
 - **Change r (retiming)**
 - * Zhou's idea

Strategy of solving the problem

- Start with a feasible T
 - $r(v) = 0, t(v) = 0, \forall v \in V$
- Gradually push down T with (1)-(4) inviolate
 - **With r unchanged**
 - * Burns's idea
 - **Change r (retiming)**
 - * Zhou's idea
- Certify optimality

Push down T with r unchanged

- Retiming validity ((1) and (2)) kept

Push down T with r unchanged

- **Retiming validity** ((1) and (2)) kept
- Smallest T for **timing validity**

$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

$$t(v) \leq T, \quad \forall v \in V \quad (4)$$

Push down T with r unchanged

- **Retiming validity** ((1) and (2)) kept
- Smallest T for **timing validity**

$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

$$t(v) \leq T, \quad \forall v \in V \quad (4)$$

- Burns's algorithm [PhD thesis CIT '91]
 - Returns minimal T to (3)

Push down T with r unchanged

- **Retiming validity** ((1) and (2)) kept
- Smallest T for **timing validity**

$$t(v) \geq t(u) + d(u, v) - w_r(u, v)T, \quad \forall (u, v) \in E \quad (3)$$

$$t(v) \leq T, \quad \forall v \in V \quad (4)$$

- Burns's algorithm [PhD thesis CIT '91]
 - Returns minimal T to (3)
- Extend Burns's to incorporate (4) as well

Burns's algorithm

While (true)

$E_c \leftarrow \{(u, v) \in E \mid t(v) = t(u) + d(u, v) - w_r(u, v)T\};$

Return T and r , if E_c contains a cycle;

Burns's algorithm

While (true)

$E_c \leftarrow \{(u, v) \in E \mid t(v) = t(u) + d(u, v) - w_r(u, v)T\};$

Return T and r , if E_c contains a cycle;

For $v \in V$ **in topological sort order of** $G_c = (V, E_c)$

do $\Delta(v) \leftarrow 0$, **if** v **is a root in** G_c ;

$\Delta(v) \leftarrow \max_{\forall (u,v) \in E_c} \{\Delta(v), \Delta(u) + w_r(u, v)\};$

Burns's algorithm

While (true)

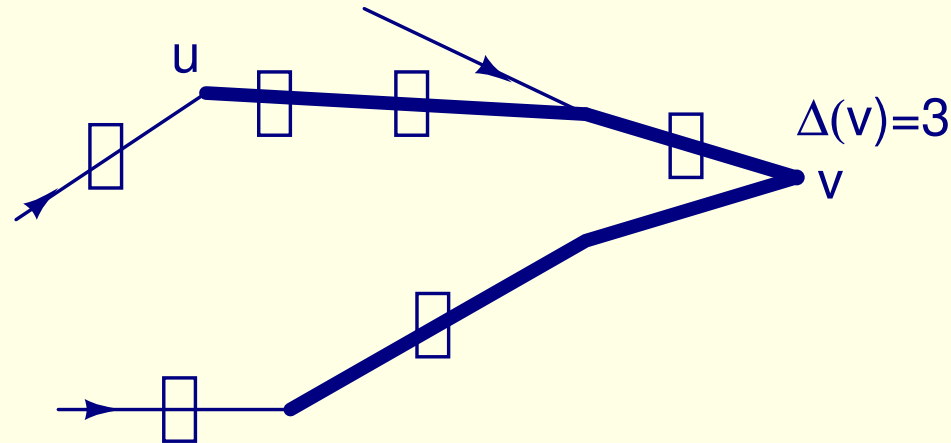
$E_c \leftarrow \{(u, v) \in E \mid t(v) = t(u) + d(u, v) - w_r(u, v)T\};$

Return T and r , if E_c contains a cycle;

For $v \in V$ **in topological sort order of** $G_c = (V, E_c)$

do $\Delta(v) \leftarrow 0$, if v is a root in G_c ;

$\Delta(v) \leftarrow \max_{\forall (u,v) \in E_c} \{\Delta(u) + w_r(u, v)\};$



Burns's algorithm

While (true)

$E_c \leftarrow \{(u, v) \in E \mid t(v) = t(u) + d(u, v) - w_r(u, v)T\}$;

Return T and r , if E_c contains a cycle;

For $v \in V$ in topological sort order of $G_c = (V, E_c)$

do $\Delta(v) \leftarrow 0$, if v is a root in G_c ;

$\Delta(v) \leftarrow \max_{\forall (u, v) \in E_c} \{\Delta(u), \Delta(u) + w_r(u, v)\}$;

$\theta \leftarrow \infty$;

For each $(u, v) \in E$

do **If** $(\Delta(u) + w_r(u, v) > \Delta(v))$ **then**

$\theta \leftarrow \min\{\theta, \frac{t(v) - t(u) - d(u, v) + w_r(u, v)T}{\Delta(u) + w_r(u, v) - \Delta(v)}\}$;

Burns's algorithm

While (true)

$E_c \leftarrow \{(u, v) \in E \mid t(v) = t(u) + d(u, v) - w_r(u, v)T\};$

Return T and r , if E_c contains a cycle;

For $v \in V$ in topological sort order of $G_c = (V, E_c)$

do $\Delta(v) \leftarrow 0$, if v is a root in G_c ;

$\Delta(v) \leftarrow \max_{\forall (u, v) \in E_c} \{\Delta(v), \Delta(u) + w_r(u, v)\};$

$\theta \leftarrow \infty$;

For each $(u, v) \in E$

do If $(\Delta(u) + w_r(u, v) > \Delta(v))$ then

$\theta \leftarrow \min\{\theta, \frac{t(v) - t(u) - d(u, v) + w_r(u, v)T}{\Delta(u) + w_r(u, v) - \Delta(v)}\};$

$T \leftarrow T - \theta$

For each $v \in V$

do $t(v) \leftarrow t(v) + \theta \cdot \Delta(v)$;

An adaption of Burns's algorithm

- Algorithmic modification

.....

For each $v \in V$ **do**

$$\theta \leftarrow \min\left\{\theta, \frac{T-t(v)}{\Delta(v)+1}\right\};$$

$$T \leftarrow T - \theta;$$

.....

An adaption of Burns's algorithm

- Algorithmic modification

.....

For each $v \in V$ do

$$\theta \leftarrow \min\left\{\theta, \frac{T-t(v)}{\Delta(v)+1}\right\};$$

$$T \leftarrow T - \theta;$$

.....

- Theoretical importance

- Push T down to the minimum, with r unchanged

Push down T by changing r

- Condition

- $\exists v, t(v) = T$

Push down T by changing r

- Condition

- $\exists v, t(v) = T$

- Zhou's algorithm

- $r(v) \leftarrow r(v) + 1$

- * Necessary to approach a possible smaller T

Push down T by changing r

- Condition
 - $\exists v, t(v) = T$
- Zhou's algorithm
 - $r(v) \leftarrow r(v) + 1$
 - * Necessary to approach a possible smaller T
 - Keep **retiming validity** ((1) and (2))

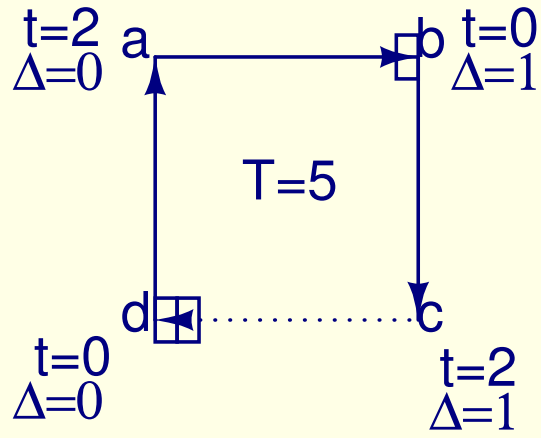
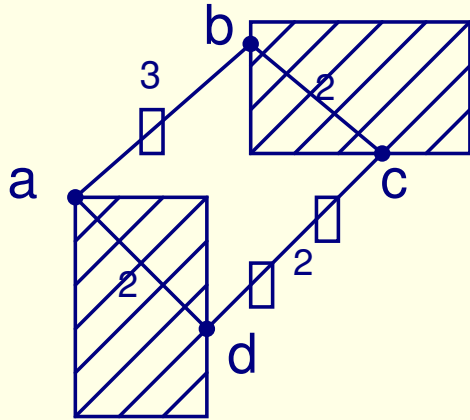
Push down T by changing r

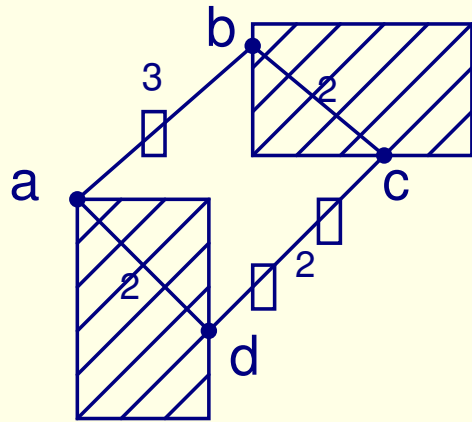
- Condition
 - $\exists v, t(v) = T$
- Zhou's algorithm
 - $r(v) \leftarrow r(v) + 1$
 - * Necessary to approach a possible smaller T
 - Keep **retiming validity** ((1) and (2))
- Run Burns's adaption under new r

Criteria to certify optimality

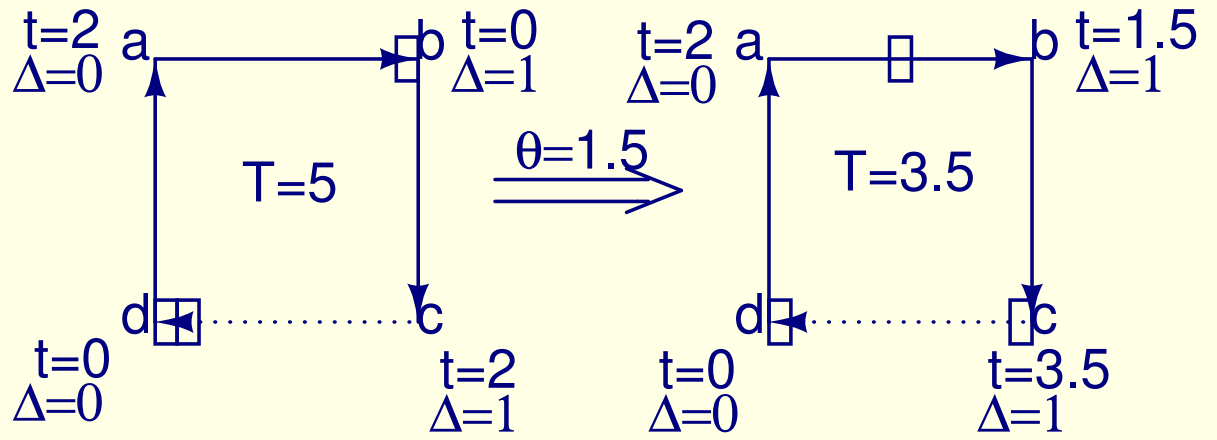
- A critical cycle
- $\exists v \in V, r(v) > N_{\text{ff}}$, the total # of FFs in the circuit

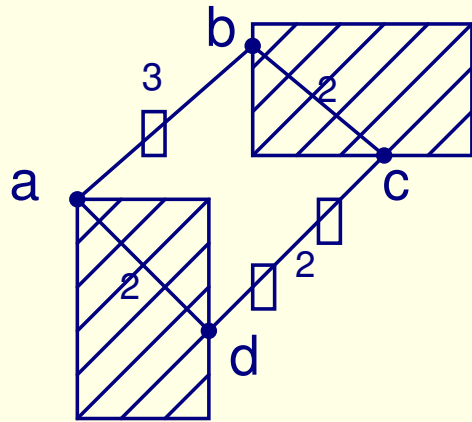
An example



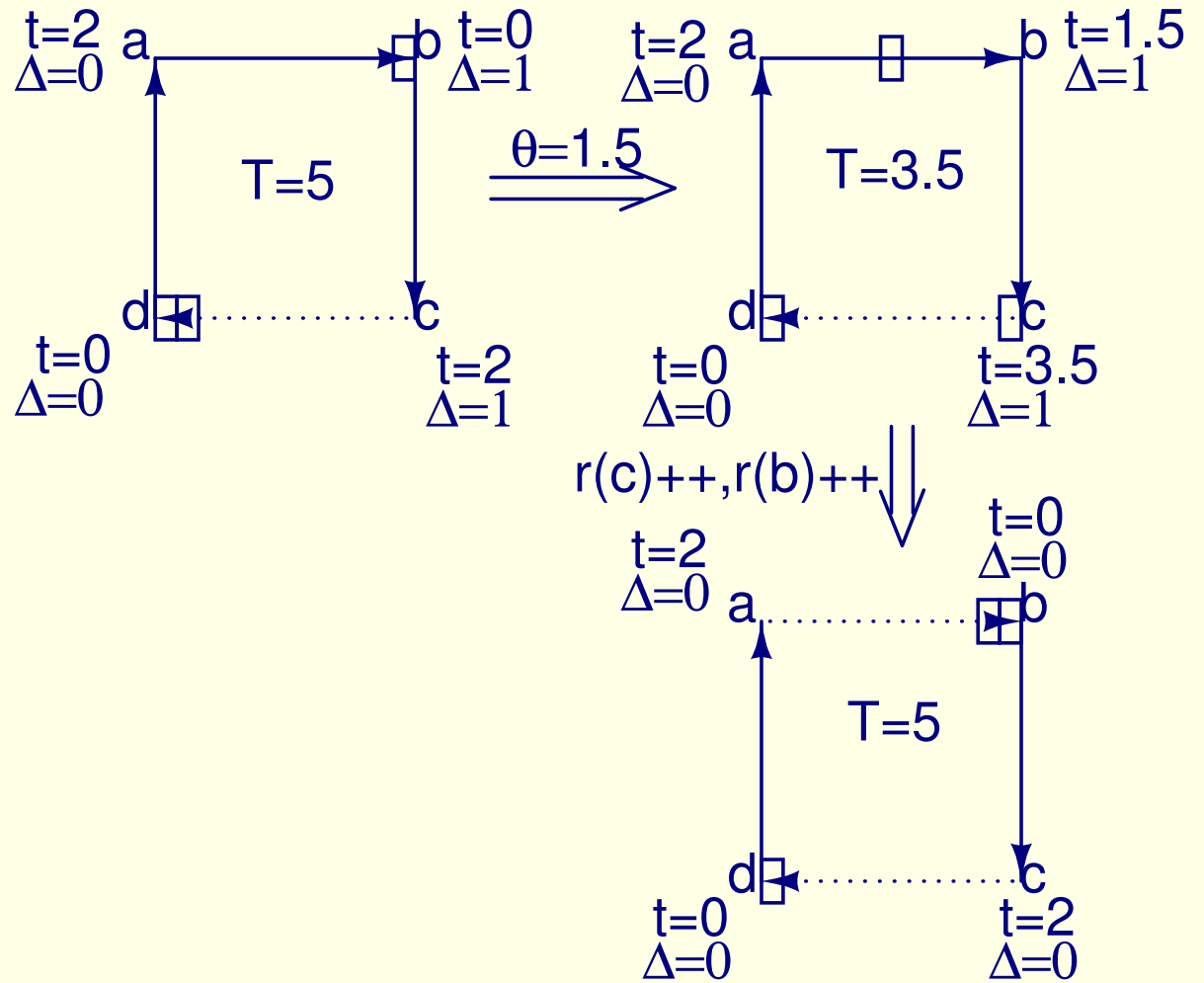


An example

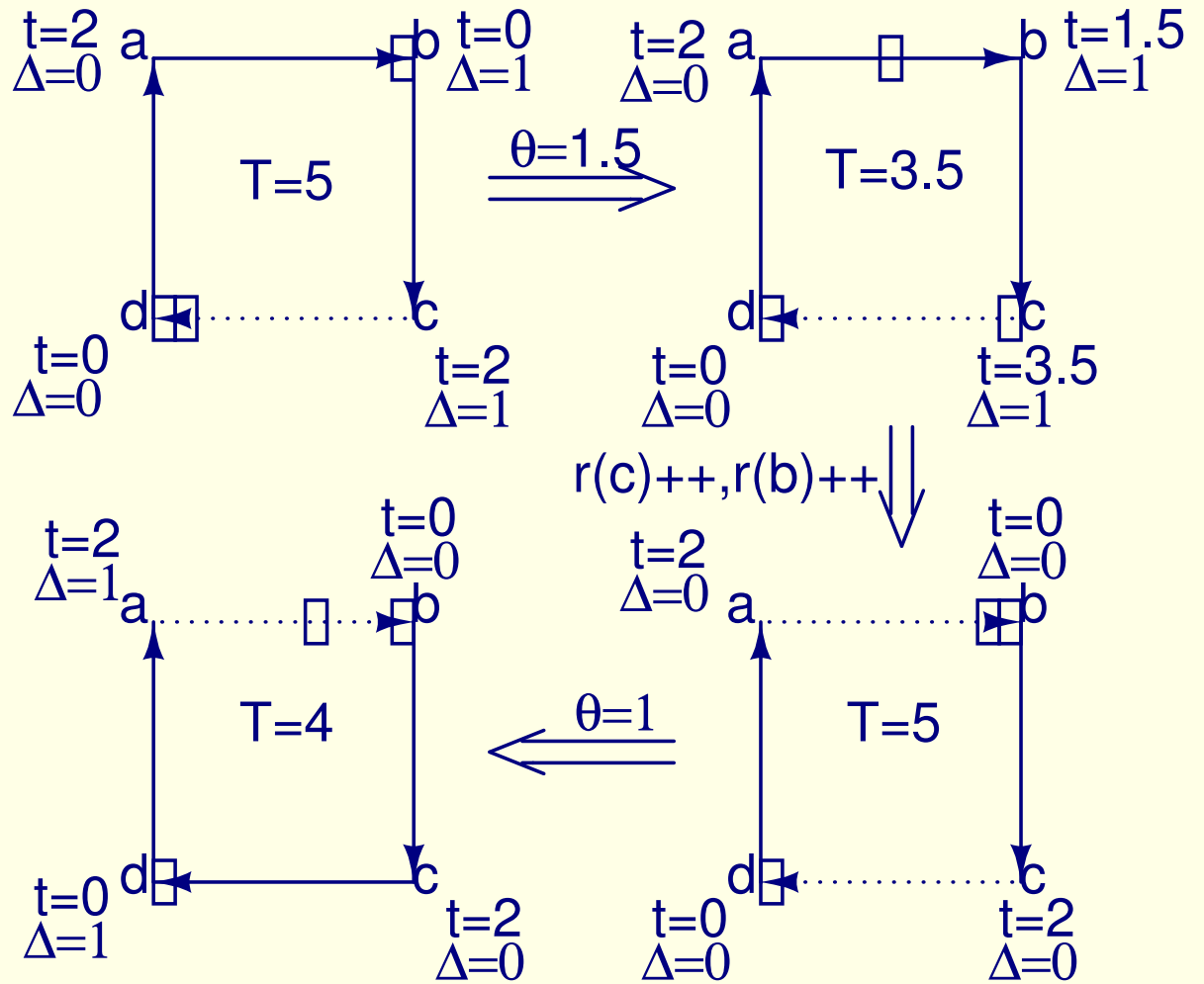
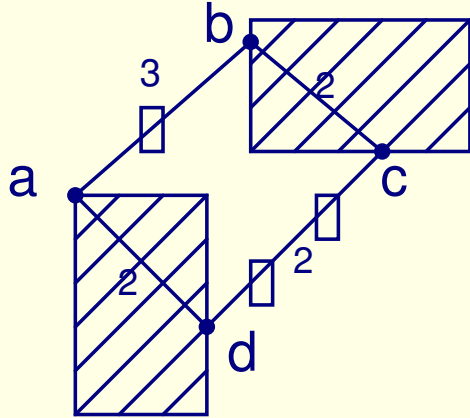




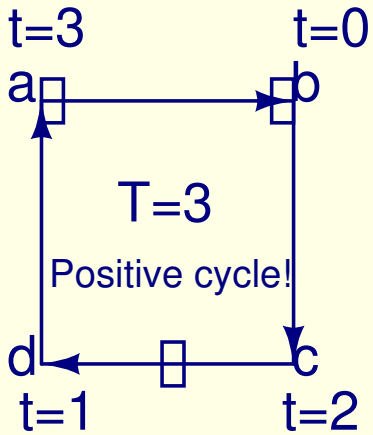
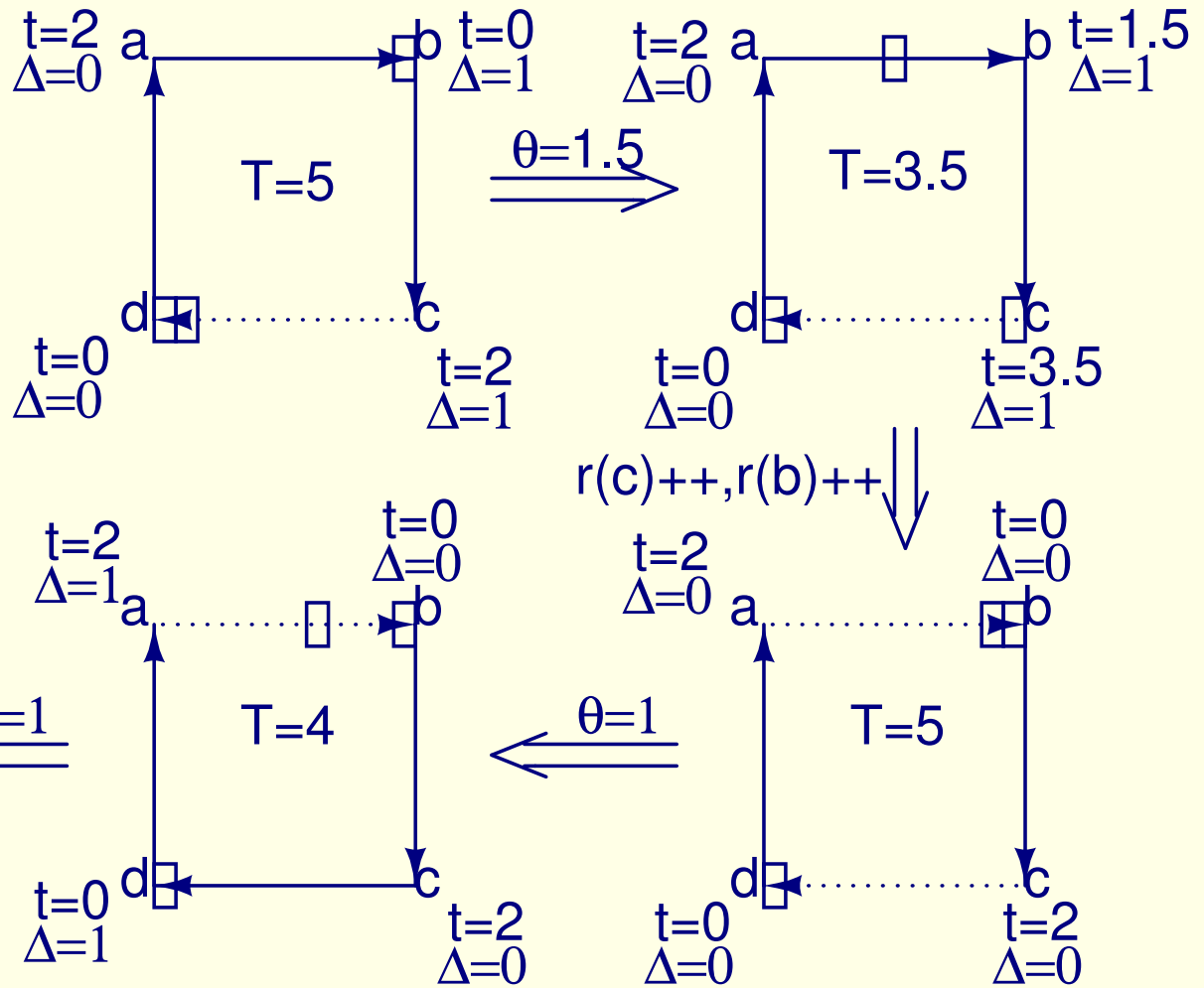
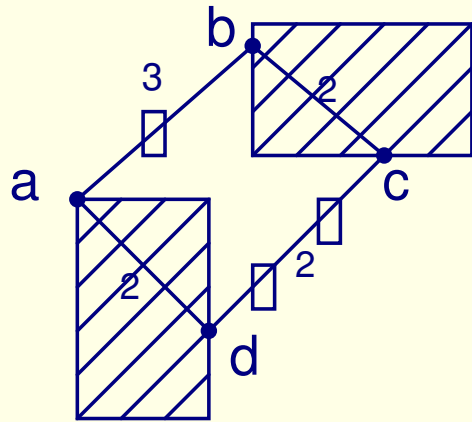
An example



An example



An example



Computation complexity

- Complexity per iteration
 - $O(|V|^2|E|)$

Computation complexity

- Complexity per iteration
 - $O(|V|^2|E|)$
- # of iterations
 - $O(|V| \cdot N_{\text{ff}})$, where N_{ff} is the total # of FFs in the circuit

Computation complexity

- Complexity per iteration
 - $O(|V|^2|E|)$
- # of iterations
 - $O(|V| \cdot N_{\text{ff}})$, where N_{ff} is the total # of FFs in the circuit
- Entire algorithm
 - $O(|V|^3|E| \cdot N_{\text{ff}})$ in the worst case

Computation complexity

- Complexity per iteration
 - $O(|V|^2|E|)$
- # of iterations
 - $O(|V| \cdot N_{\text{ff}})$, where N_{ff} is the total # of FFs in the circuit
- Entire algorithm
 - $O(|V|^3|E| \cdot N_{\text{ff}})$ in the worst case
 - Remarkable efficiency in practice

Experimental results

- Benchmark: ISCAS-89
 - 1st test set: treat gates as blocks
 - 2nd test set: circuits w/ non-complete bipartite blocks
 - * Use hMETIS to partition a circuit into groups
 - * Treat each group as a block

Optimal clock period

| Circuit | $ V $ | $ E $ | N_{ff} | w/o non-CB | | w/ non-CB | | |
|---------------|-------|-------|-----------------|------------|------------------|-----------|-------|------------------|
| | | | | #Step | T^{opt} | #Par | #Step | T^{opt} |
| s386 | 519 | 700 | 6 | 13 | 51.1 | 50 | 1 | 55.0 |
| s400 | 511 | 665 | 21 | 120 | 32.2 | 50 | 1 | 50.6 |
| s444 | 557 | 725 | 21 | 289 | 35.2 | 40 | 1 | 63.2 |
| s838 | 1299 | 1206 | 32 | 2 | 76.0 | 130 | 1 | 84.0 |
| s953 | 1183 | 1515 | 29 | 31 | 60.6 | 110 | 2 | 69.5 |
| s1488 | 2054 | 2780 | 6 | 11 | 70.6 | 200 | 1 | 73.3 |
| s1494 | 2054 | 2792 | 6 | 63 | 76.9 | 160 | 1 | 79.9 |
| s5378 | 7205 | 8603 | 179 | 26 | 111.2 | 500 | 1 | 115.3 |
| s13207 | 19816 | 22999 | 669 | 129 | 239.5 | 1000 | 1 | 292.8 |
| s35932 | 46097 | 58266 | 1728 | 68 | 148.3 | 2000 | 1 | 163.2 |
| s38584 | 53473 | 66964 | 1452 | 126 | 204.0 | 2000 | 1 | 264.0 |

Running time comparison (in seconds)

- t_{bs1} [ICCAD'03], t_{bs2} [DATE'04], precision=0.1

| Circuit | w/o non-CB blocks | | | w/ non-CB blocks | | |
|---------|-------------------|-----------|--------------|------------------|-----------|-------------|
| | t_{bs1} | t_{bs2} | t_{new} | t_{bs1} | t_{bs2} | t_{new} |
| s386 | 1.97 | 0.01 | 0.00 | 3.67 | 0.01 | 0.00 |
| s400 | 1.64 | 0.01 | 0.03 | 3.38 | 0.01 | 0.00 |
| s444 | 2.23 | 0.03 | 0.09 | 4.31 | 0.01 | 0.00 |
| s838 | 8.79 | 0.03 | 0.00 | 33.42 | 0.02 | 0.00 |
| s953 | 9.76 | 0.04 | 0.02 | 17.56 | 0.07 | 0.00 |
| s1488 | 35.17 | 0.08 | 0.08 | 98.88 | 0.05 | 0.00 |
| s1494 | 34.13 | 0.08 | 0.06 | 62.86 | 0.09 | 0.00 |
| s5378 | 684.6 | 0.24 | 0.31 | 1344.74 | 0.29 | 0.00 |
| s13207 | - | 1.07 | 3.46 | - | 206.52 | 0.02 |
| s35932 | - | 18.63 | 7.55 | - | 6.19 | 0.19 |
| s38584 | - | 7.44 | 30.17 | - | 21992.67 | 0.19 |

Summary

- The problem of optimal wire retiming is becoming more and more important in VLSI design

Summary

- The problem of optimal wire retiming is becoming more and more important in VLSI design
- A brand new algorithm is presented
 - Avoids binary search

Summary

- The problem of optimal wire retiming is becoming more and more important in VLSI design
- A brand new algorithm is presented
 - Avoids binary search
 - Polynomial time bounded

Summary

- The problem of optimal wire retiming is becoming more and more important in VLSI design
- A brand new algorithm is presented
 - Avoids binary search
 - Polynomial time bounded
 - Implementation is simple

Summary

- The problem of optimal wire retiming is becoming more and more important in VLSI design
- A brand new algorithm is presented
 - Avoids binary search
 - Polynomial time bounded
 - Implementation is simple
 - Practical efficiency

Summary

- The problem of optimal wire retiming is becoming more and more important in VLSI design
- A brand new algorithm is presented
 - Avoids binary search
 - Polynomial time bounded
 - Implementation is simple
 - Practical efficiency
 - Incremental in nature
 - * Be combined with gate sizing, budgeting, etc.

Thank you !