Timing Issues in Digital Designs

Prof. Chien-Nan Liu
TEL: 03-4227151 ext:34534
Email: jimmy@ee.ncu.edu.tw

Outline

- Timing Models in Digital Circuits
- Timing Analysis
- Timing Optimization
- Retiming for Sequential Circuits

Functionality vs. Performance

- Functionality: system correctly implements specified function
  - Mostly checked by simulation (logic, RTL, behavior)
- Performance: system correctly implements specified function at specified speed
  - Another important consideration in a design
  - Can be checked by simulation or static analysis
- There is always a trade-off between cost (area) and performance
  - Require some estimations at design level

8-3

Delay Model at Logic Level

1. unit delay model
   - Assign a delay of 1 to ***each gate***
2. unit fanout delay model
   - Incorporate an ***additional*** delay for ***each fanout***
3. library delay model
   - Use delay data in the library to provide more accurate delay value
   - May use linear or non-linear (tabular) models
   - Rising delay and falling delay are separated
   - [min, typ, max] model may also be used

8-4

Factors that Affect Timing

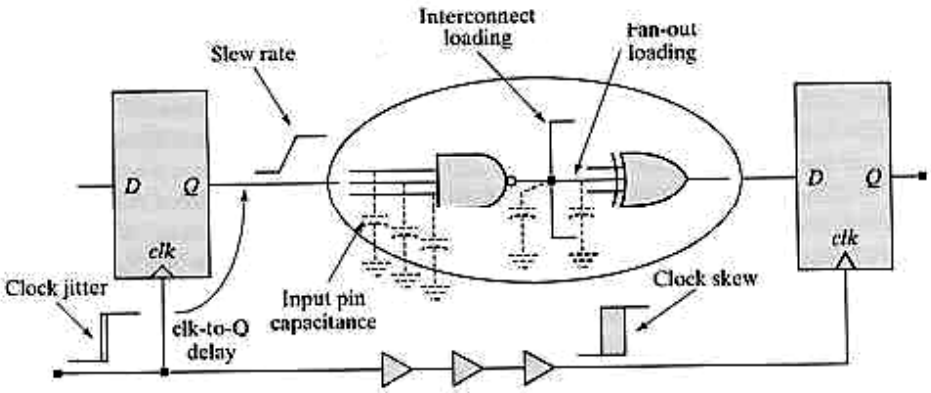


FIGURE 11-9 Factors affecting the timing of a synchronous circuit.

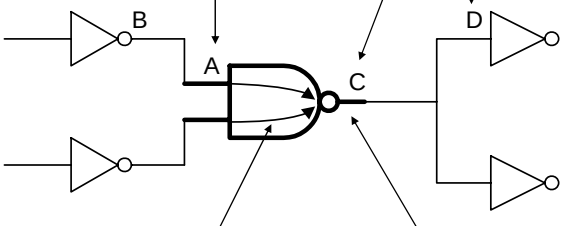
8-5

Linear Delay Model

Delay = Dslope + Dintrinsic + Dtransition + Dwire

Ds : Slope delay : delay at input A caused by the transition delay at B

Dw : Wire delay : time from state transition at C to state transition at D



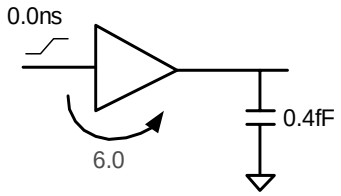
D_I : Intrinsic delay : incurred from cell input to cell output

D_T : Transition delay : output pin loading, output pin drive

8-6

Tabular Delay Model

- Delay values are obtained by a look-up table
  - Two-dimensional table of delays (m by n)
    - with respect to input slope (m) and total output capacitance (n)
  - Each value in the table is obtained by real measurement



Total Output Load (fF)

Input Transition (ns)	0.2	0.3	0.4	0.5
0	3	4.5	6	7
0.1	5	8	10.7	13

Cell Delay (ps)

- Can be more precise than linear delay model
  - table size ↑ ➔ accuracy ↑

8-7

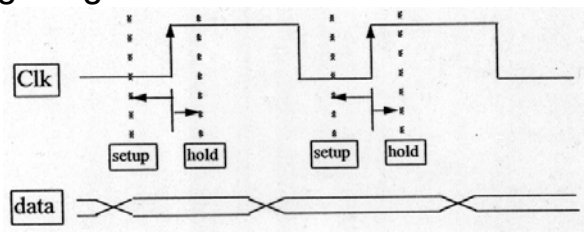
Timing Models of Flip-Flops

- The delay time of a flip-flop is often referred to the **clock-to-Q** delay
  - Operations can start only when clock signal arrives
- The arrival time of clock signal must also be considered
  - Often be referred to the **clock uncertainty** (clock jitter, clock skew, …)
- **Setup Time**: The length of time that data must stabilize before the clock transition
  - The maximum data path is used to check setup time
- **Hold Time**: The length of time that data must remain stable after the active clock transition
  - The minimum data path is used to check hold time

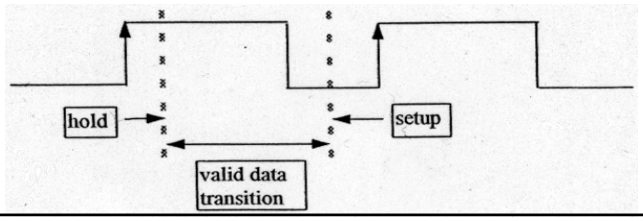
8-8

Setup & Hold Times

- Timing Diagram



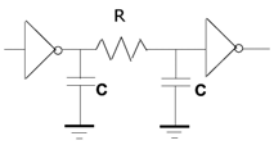
- Valid Data Transition



8-9

Interconnect Delay

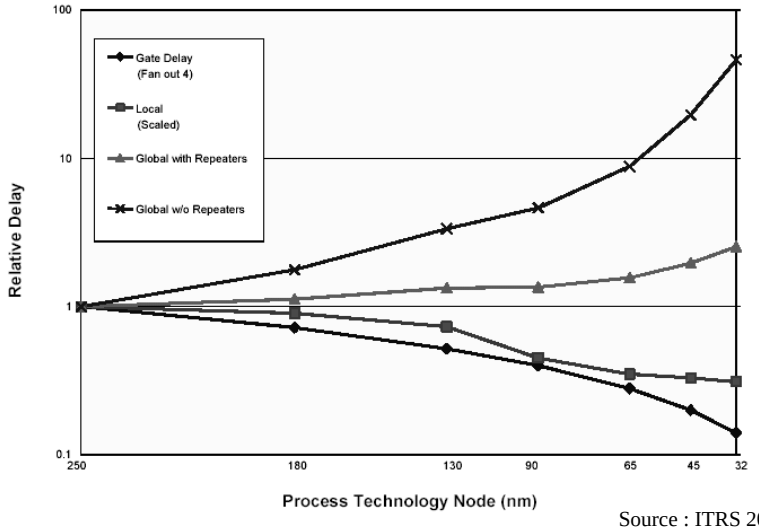
- As taught in digital design course, interconnect delays can be ignored compared to gate delays
  - Is it true in deep-submicron process
- Wire delays start to dominate total delay in DSM era
  - Should be considered carefully
- Several RC delay models for a net are available
  - Ex: π model



- However, only rough wire delay model can be used at design phase
  - Basically, wire delay is determined by wire length
  - Lack of physical information about wire length at early stage
  - Not accurate at all, just an experience value

8-10

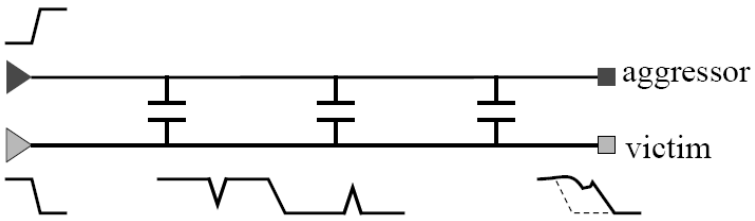
Wiring Delay vs. Feature Size



Source : ITRS 2001

8-11

Crosstalk Noise Between Wires



- Two neighbor wires may be affected by each other
  - Depending on the types of the two signal transitions
- Crosstalk noise may cause
  - Glitch and logical error
  - Extra propagation delay on wires
- Require complex analysis to predict the effects

8-12

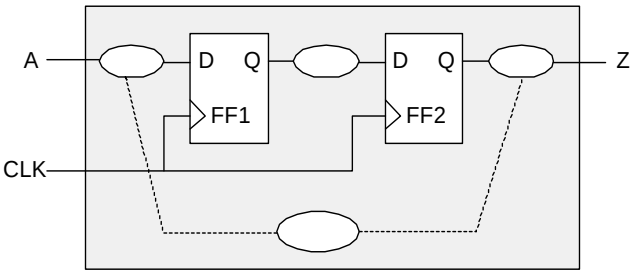
Outline

- Timing Models in Digital Circuits
- Timing Analysis
- Timing Optimization
- Retiming for Sequential Circuits

8-13

What is STA ?

- STA = static timing analysis
- STA is a method for determining if a circuit meets timing constraints without having to simulate

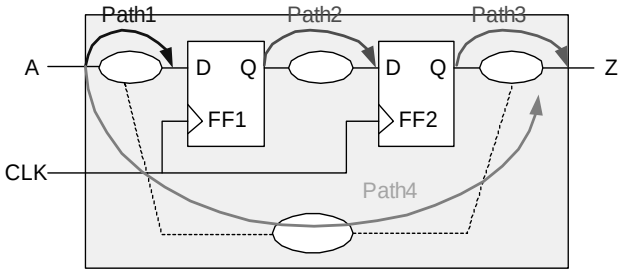


Reference :Synopsys

8-14

Timing Paths

- There are 4 types of timing path
  - Input port to data pin of FF (path 1)
  - Clock pin of FF to data pin of FF (path 2)
  - Clock pin of FF to output port (path 3)
  - Input port to output port (path 4)

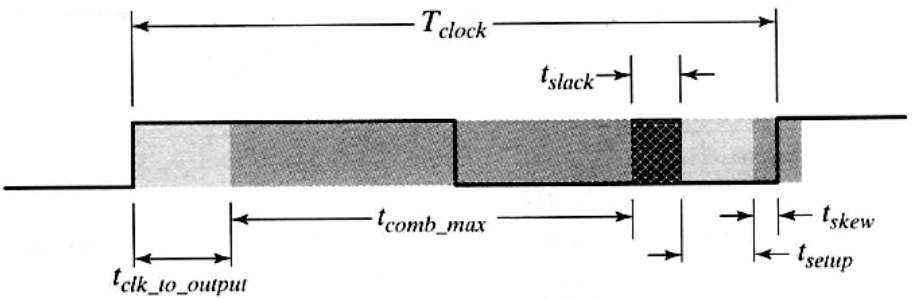


Stat point :
input port
Clock pin of FF
End point
output port
data pin of FF

Reference :Synopsys

Timing Constraint

$$T_{clock} > t_{clk\_to\_output} + T_{comb\_max} + T_{setup} + T_{skew}$$

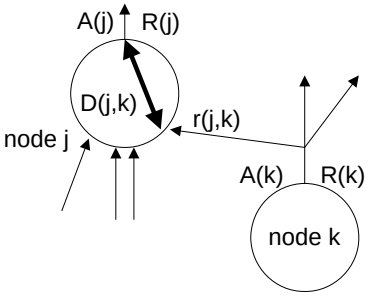


Arrival Time and Required Time

- arrival time : calculated from input to output
- required time : calculated from output to input
- slack = required time - arrival time

A(j): arrival time of signal j
R(k): required time or for signal k
S(k): slack of signal k
D(j,k): delay of node j from input k

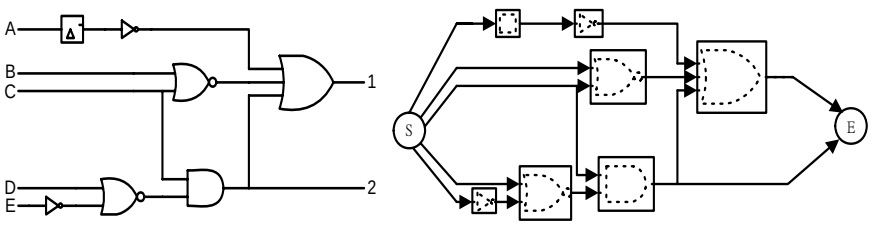
$A(j) = \max_{k \in FI(j)} [A(k) + D(j,k)]$
$r(j,k) = R(j) - D(j,k)$
$R(k) = \min_{j \in FO(k)} [r(j,k)]$
$S(k) = R(k) - A(k)$



8-17

Delay Graph

- Replace logic gates with delay blocks
- Add start (S) and end (E) blocks
- Indicate signal flow with directed arcs



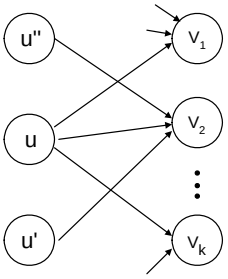
8-18

Longest and Shortest Path

- If we visit vertices in precedence order, the following code will need executing only once for each *u*

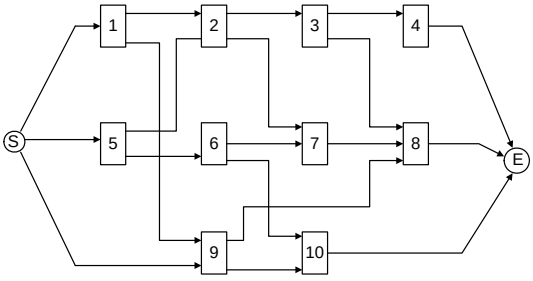
Update Successors[u]

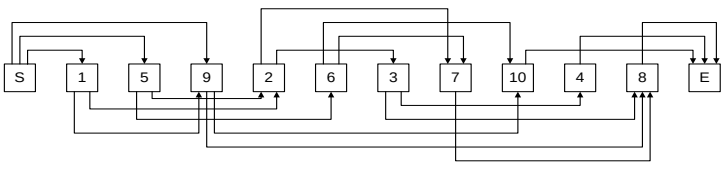
```
1  for each vertex v ∈ Adj[u] do
2      if  A[v] < A[u] + Δ[u]   // longest
3          then A[v] ← A[u] + Δ [u]
4          LP[v] ← u  fi
5      if  a[v] >a[u] + δ[u]   // shortest
6          then a[v] ← a[u] + δ[u]
7          SP[v] ← u  fi
```



8-19

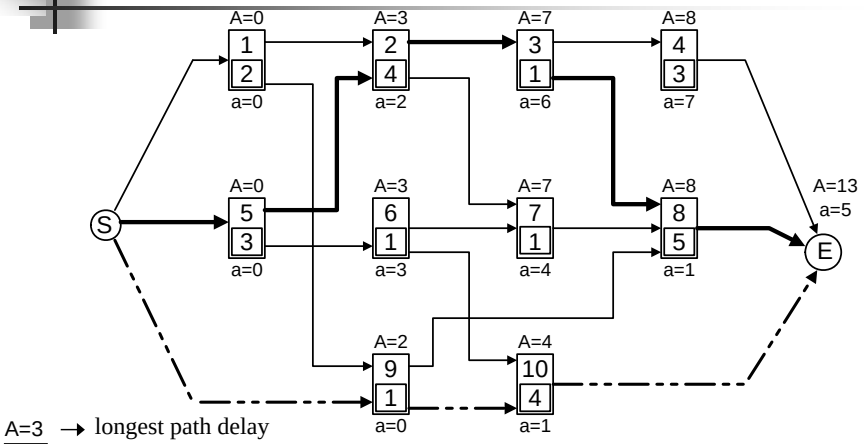
Delay Graph and Topological Sort





8-20

Delay Calculation



A=3 → longest path delay

2 → node number

4 → gate delay

a=2 → shortest path delay

P.S: The longest delay and shortest delay of each gate are assumed to be the same.

8-21

Timing Report

Point	Fanout	Incr	Path
.....			
U19/Z (INB)		0.38	50.76 r
n397 (net)	8	0.00	50.76 r
data_tri[5]/Z (BTD)		0.37	51.13 f
data[5] (net)	8	0.00	51.13 f
data[5] (inout)		0.00	51.13 f
data arrival time			51.13
clock clk (rise edge)		100.00	100.00
clock network delay (ideal)		0.00	100.00
clock uncertainty		-0.50	99.50
output external delay		-20.00	79.50
data required time			79.50
data required time			79.50
data arrival time			-51.13
slack (MET)			28.37

Meet timing

8-22

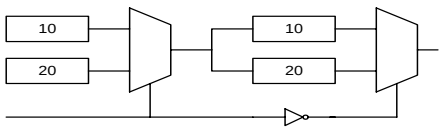
The False Path Problem

- Pattern-independent analysis calculates the delay of logic gates without regard to their function
  - Can result in delays that are unnecessarily conservative
- Some paths may never be sensitized false paths
- Checking for false paths requires the function of the gates to be considered
  - Can approach the complexity of logic simulation or test generation

8-23

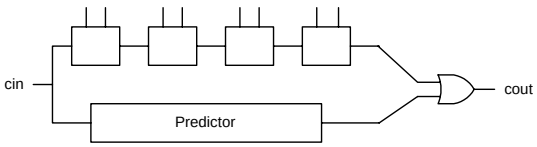
Combinational False Paths

- A false path is a path that is never exercised in operation
- The Classical example :



Longest true path has delay 30

- A carry bypass adder



8-24

Path Sensitization

VALUE / GATE	Controlling value	NonControlling Value
AND	0	1
NAND	0	1
OR	1	0
NOR	1	0
NOT	0 , 1	---

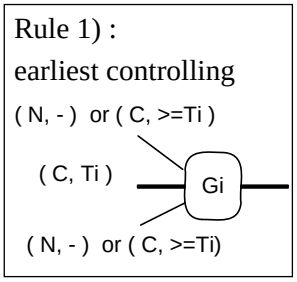
- Path P = (I , G1 , G2 , … , Gn , O) is sensitizable if there is at least one vector which sensitizes the path

8-25

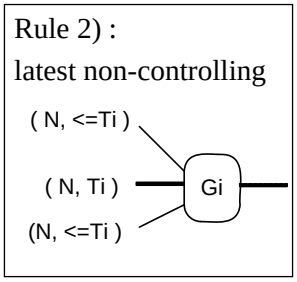
Path Sensitization

Path P = (I , G1 , G2 , …, Gn , O) is sensitizable under input vector V, if each gate meets either of the following rules under V.

Rule 1) :
earliest controlling



Rule 2) :
latest non-controlling

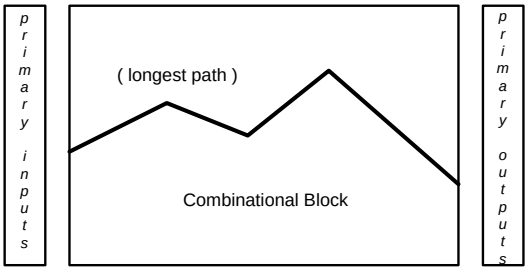


(stable value, stable time)
N : Non - controlling input , C : Controlling input
Ti = delay of partial path (I , G1 , G2 , … , Gi)

8-26

Accurate Performance

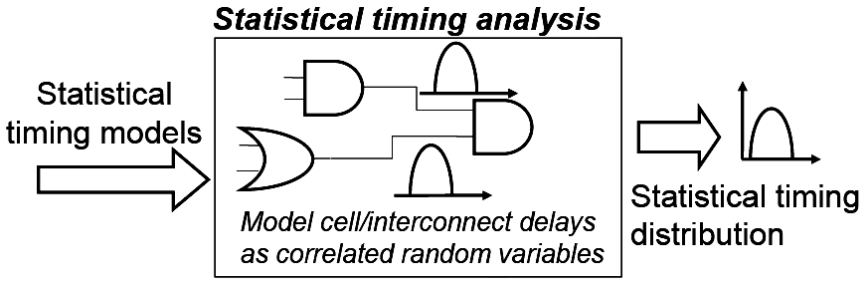
- Actual delay of the circuit: delay of the ***Longest Sensitizable Path***
  - The longest sensitizable path can be much shorter than the longest path in a complex circuit
  - Accuracy increases at the expense of computing efficiency



8-27

Statistical Timing Model

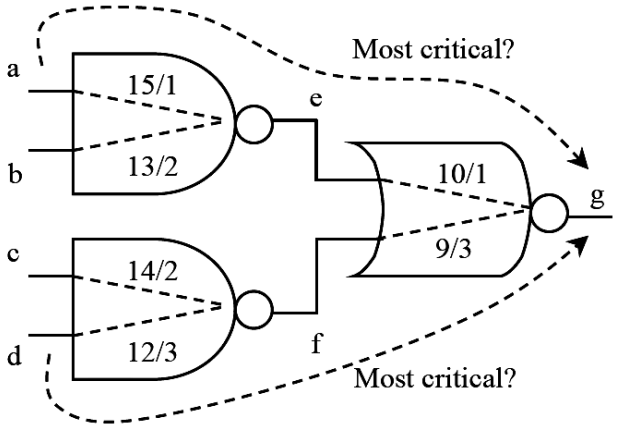
- Circuit delay elements are statistical in nature
  - Process variation
  - Manufacturing defects (open, short, …)
  - Power supply noise
  - Interconnect crosstalk



8-28

Where is the Critical Path?

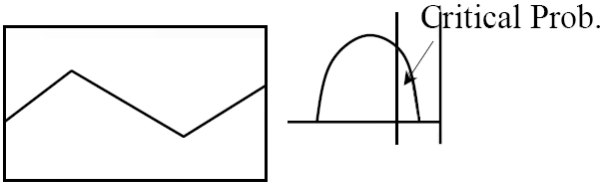
- The most critical path can be different based on which delay model you have in mind



8-29

Statistical Static Timing Analysis

- The delay of each element is a statistical distribution
  - When calculating the total delay, the "sum" of every gates becomes the "convolution" of their PDF curves
- How critical a path is can be defined as the probability of exceeding the clock period T
- Ex: crt. Prob. = 0.1 for a path P
  - Out of 100 chips, 10 of them have path P's delay > clock

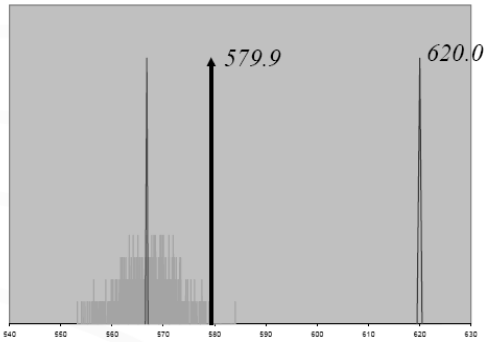


8-30

Statistical Analysis of s38584

Statistical Analysis
Mean *567.0*
Std.dev. *4.29 (0.8%)*
$\mu+3\sigma$ ***579.9***

Traditional static analysis:
Circuit delay:
cell delay is μ_c : ***566.8***
cell delay is $\mu_c+3\sigma_c$: ***620.0***



Delay of traditional worst-case analysis (620.0) is greater than that of the 3σ delay of statistical analysis (579.9)!!

8-31

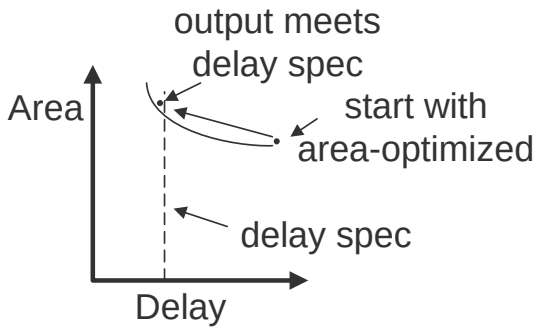
Outline

- Timing Models in Digital Circuits
- Timing Analysis
- Timing Optimization
- Retiming for Sequential Circuits

8-32

Timing Optimization

- There is always a trade-off between area and delay
- Optimize timing to meet delay spec. with minimum area



8-33

Restructuring Algorithm

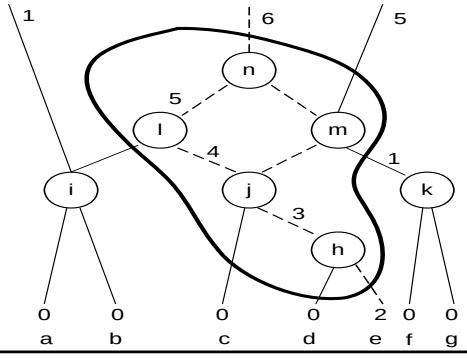
While (circuit timing improves) do
    select regions to transform
    collapse the selected region
    resynthesize for better timing
done

- Which regions to restructure ?
- How to resynthesize to minimize delay ?

8-34

Restructuring Regions

- All nodes with slack within ε of the most critical signal belong to the ε-*network*
- To improve circuit delay, necessary and sufficient to improve delay at nodes on cut-set of ε-*network*



8-35

Find the Cutset

- The weight of each node is W = Wxt + α * Wxa
  - Wxt is potential for speedup
  - Wxa is area penalty for duplication of logic
  - α is decided by various area/delay tradeoff
- Apply the maxflow-mincut algorithm to generate the cutset of the ε-*network*
- ε: Specify the size of the ε-*network*
  - Large ε might waste area without much reduction in critical delay
  - Small ε might slow down the algorithm
- α: Control the tradeoff between area and speed
  - Large α avoids the duplication of logic
  - α = 0 implies a speedup irrespective of the increase in area

8-36

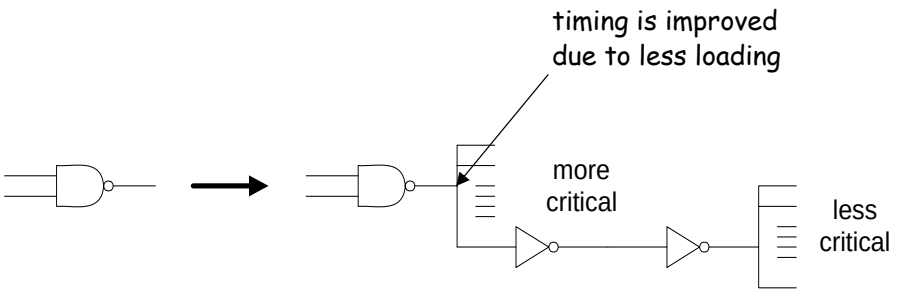
Timing Optimization Techniques (1/8)

- Fanout optimization
  - Buffer insertion
  - Split
- Timing-driven restructuring
  - Critical path collapsing
  - Timing decomposition
- Misc
  - De Morgan
  - Repower
  - Down power
- Most of them will increase area to improve timing
  - Have to make a good trade-off between them

8-37

Timing Optimization Techniques (2/8)

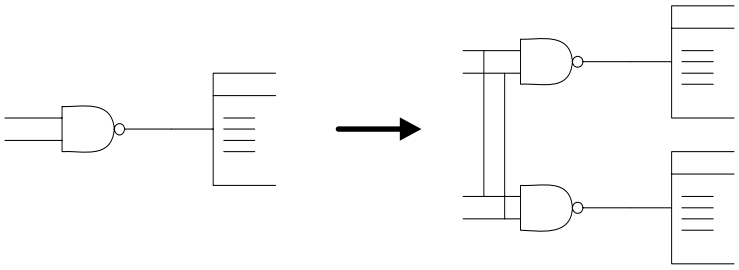
- **Buffer insertion**: divide the fanouts of a gate into critical and non-critical parts and drive the non-critical fanouts with a buffer



8-38

Timing Optimization Techniques (3/8)

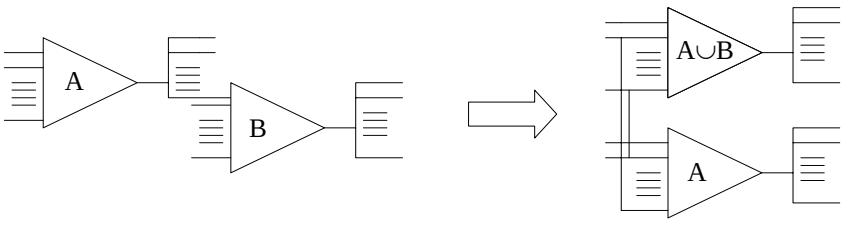
- **Split**: split the fanouts of a gate into several parts. Each part is driven with a copy of the original gate.



8-39

Timing Optimization Techniques (4/8)

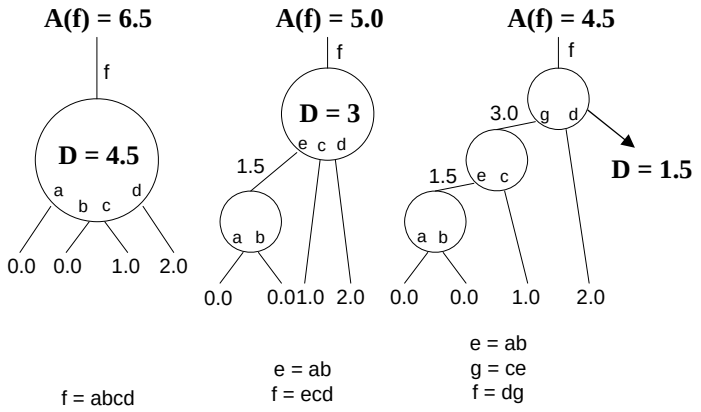
- **Critical path collapsing**: reduce the depth of logic networks



8-40

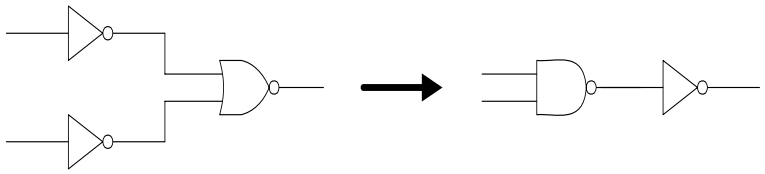
Timing Optimization Techniques (5/8)

- **Timing decomposition**: restructuring the logic networks to minimize the arrival time



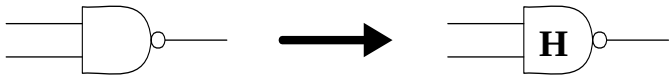
Timing Optimization Techniques (6/8)

- **De Morgan**: replace a gate with its dual, and reverse the polarity of inputs and output
  - NAND gate is typically faster than NOR gate



Timing Optimization Techniques (7/8)

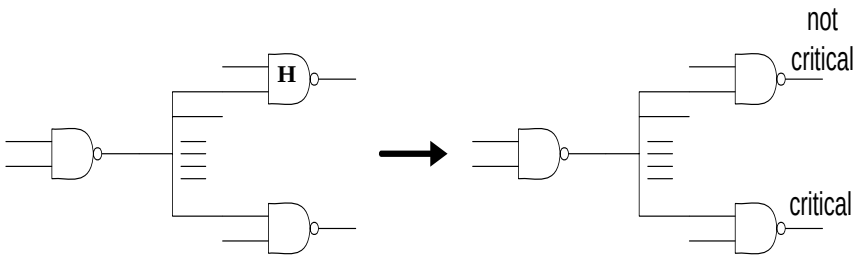
- **Repower**: replace a gate with one of the other gate in its logic class with higher driving capability



8-43

Timing Optimization Techniques (8/8)

- **Down power**: reducing gate size of a non-critical fanout in the critical path



8-44

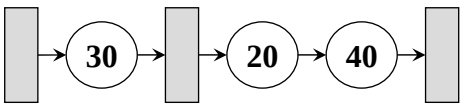
Outline

- Timing Models in Digital Circuits
- Timing Analysis
- Timing Optimization
- Retiming for Sequential Circuits

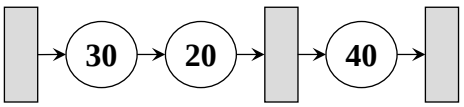
8-45

Retiming

- Exploit the ability to move registers in a circuit
  - To minimize the cycle time
  - To minimize the the number of registers for a given cycle time



cycle time = 60

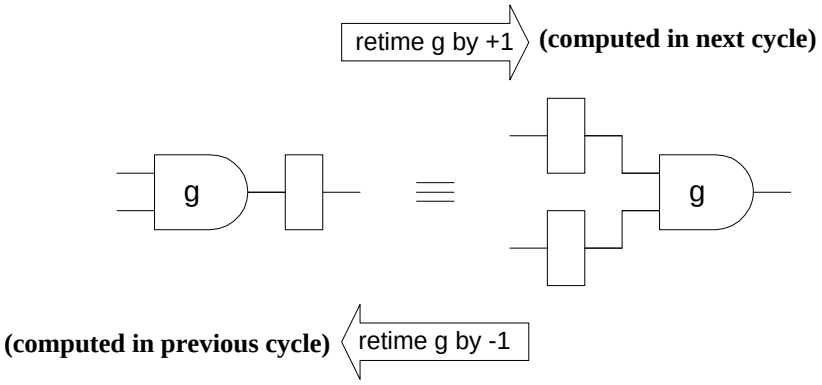


cycle time = 50

8-46

Moving Registers

- Combinational logic are not modified



8-47

Formulation

- Directed graph:
  - Nodes - combinational logic
  - Edges - connections (possible latched) between logic
- Weights
  - Nodes - combinational logic propagation delay
  - Edges - number of registers
- Path delay $d(P)$: sum of node delays along a path
- Path weight $w(P)$: sum of edge weights along a path
- Clock period: $\Phi(G) = \max\{d(p) \mid w(p) = 0\}$

8-48

Some Definitions

- $W(u,v)$ is defined as the minimum number of registers on any path from vertex u to vertex v
- The critical path p is a path from u to v such that $w(p) = W(u,v)$
- $D(u,v)$ is defined as the maximum total propagation delay on any critical path from u to v

8-49

Synchronous Circuits

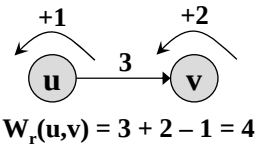
A synchronous circuit must satisfy following rules:

- D1: The propagation delay $d(v)$ is non-negative for each vertex v
  - Infeasible in real cases
- W1: The register count $w(e)$ is non-negative for each edge e
  - Infeasible in real cases
- W2: In any directed cycle, there is some edge with positive register count
  - No combinational loops

8-50

Retiming: Formulation

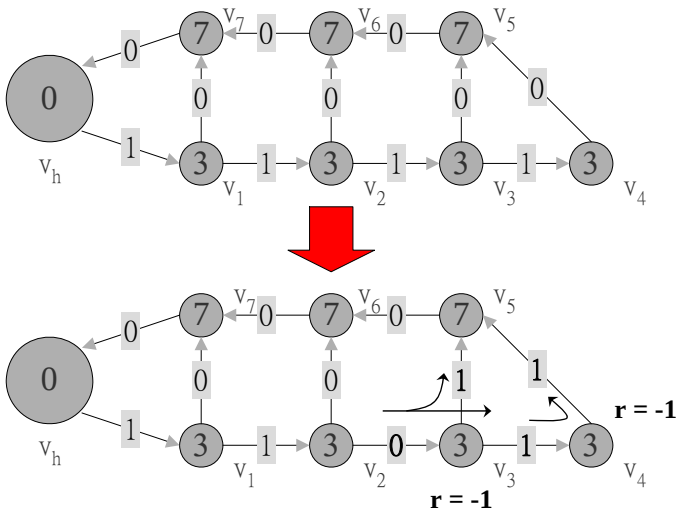
- Assign an integer-valued labeling r to each vertex
- $w_r(u,v) = w(u,v) + r(v) - r(u)$
- $w_r(p) = w(p) + r(v) - r(u)$



- Corollary: For any cycle p, $w_r(p) = w(p)$
- Legal retiming needs only being checked against condition W1: non-negative edge weight
- Corollary: Let G be a synchronous circuit and r be a retiming on G. Then the retimed graph G_r satisfies condition W2

8-51

Relocating Registers



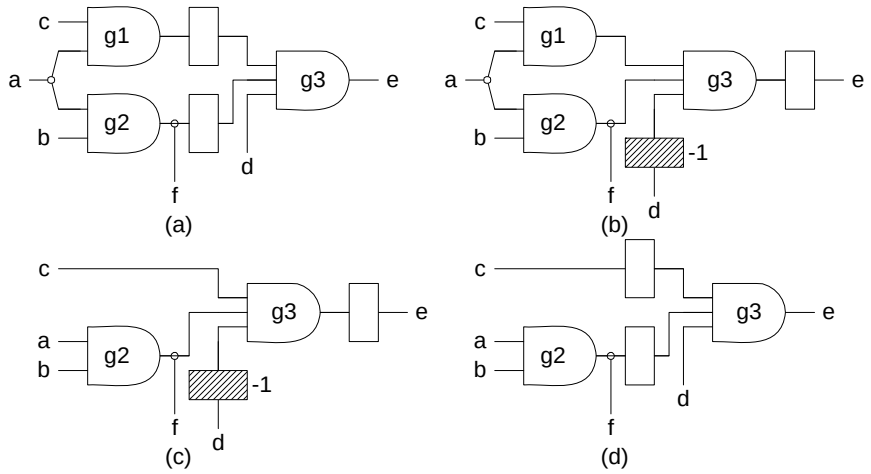
8-52

Retiming and Resynthesis

- Migrate all registers to the periphery of a sub-network
  - Peripheral retiming
- Optimize the sub-network with any combinational technique
  - Resynthesis
- Replace registers back in the sub-network
  - Retiming
- This procedure may further improve the timing across the registers

8-53

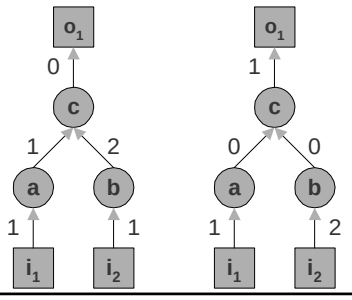
Examples of Resynthesis



8-54

Peripheral Retiming

- A peripheral retiming is a retiming such that
  - $r(v)=0$ where v is an I/O pin
  - $w(u,v)+r(v)-r(u)=0$ where $e(u,v)$ in an internal edge
- Move all registers to the peripheral edges
- Leave a purely combinational logic block between two set of registers
- Example:



8-55

Conditions for Peripheral Retiming

- No two paths between any input i and any output j have different edge weights
- Exist α_i and β_j, $1 \le i \le m$, $1 \le j \le$ n such that $W_{i,j} = \alpha_i + \beta_j$ (m: no. of inputs; n: no. of outputs)
- $W_{i,j} = \sum_{path\ i_i \text{ -> } o_j} w(e)$ if all paths between input i and output j have the same weight
- Complexity $O(e \cdot \min(m,n))$

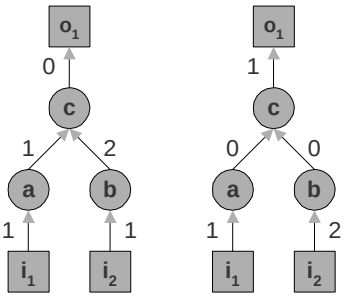
8-56

Examples of Peripheral Retiming

- Example 1:

  $W_{1,1}=2,\ W_{2,1}=3,$

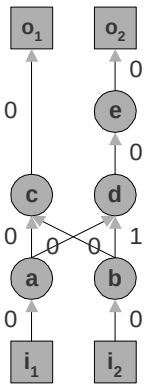
  $\Rightarrow \alpha_1=1,\ \alpha_2=2,\ \beta_1=1$



- Example 2:

  $W_{1,1}=0, W_{1,2}=0, W_{2,1}=0,\ W_{2,2}=1$

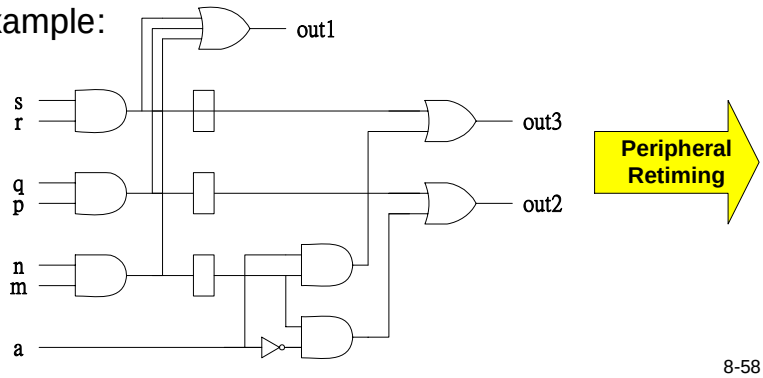
  $\Rightarrow$ no solution



8-57

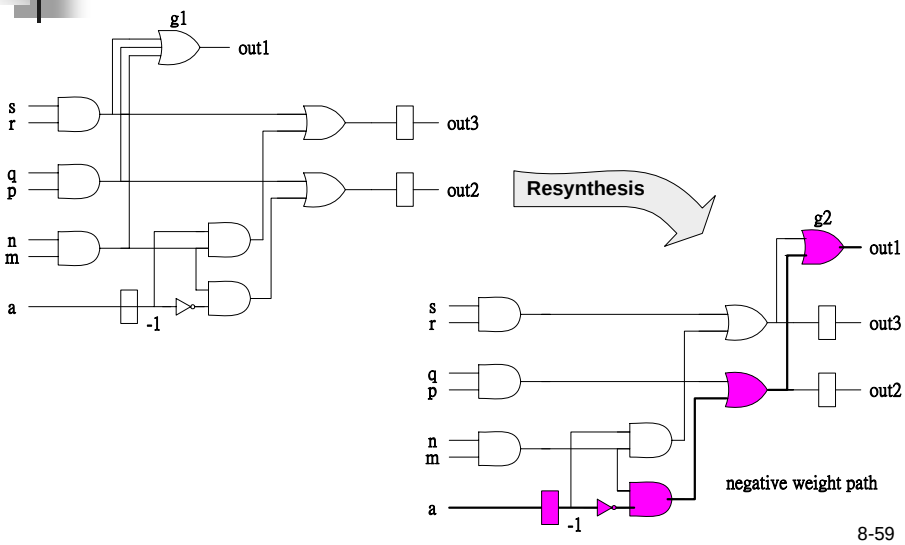
Legal Resynthesis Operations (1/2)

- Any that do not create a path with negative weight
- Resynthesis could create pseudo-dependency between any input and output
- Example:



8-58

Legal Resynthesis Operations (2/2)



8-59

Effects of Retiming and Resynthesis

- Area optimization:
  - No significant improvement
  - Limitation on existing combinational optimization techniques
  - Some circuits (pipelined datapaths) have inherently no potential for further optimization using retiming and resynthesis techniques
- Performance optimization of pipelined circuits:
  - Significant improvements for pipelined arithmetic circuits

8-60