



Logic Synthesis – Part 2

Technology-Dependent Optimization

Prof. Chien-Nan Liu
Institute of Electronics
National Chiao-Tung Univ.

Tel: (03)5712121 ext:31211
E-mail: jimmyliu@nctu.edu.tw
<http://www.ee.ncu.edu.tw/~jimmy>

Courtesy: Prof. Jing-Yang Jou

NCTU M SEDA Lab.



1

Outline

- Synthesis overview
- RTL synthesis
- Two-level logic optimization
- Multi-level logic optimization
- Technology mapping
- Timing analysis
- Timing optimization
- Synthesis for low power

與製程無關

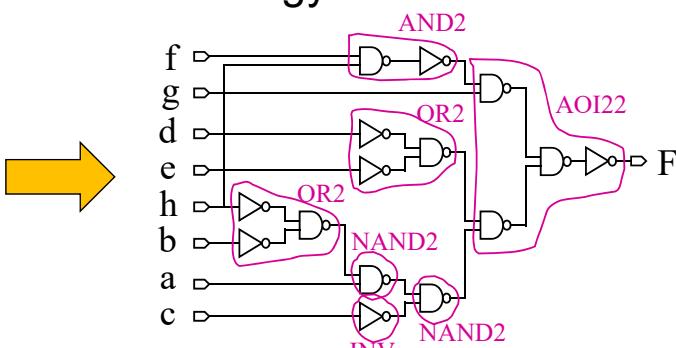


NCTU M SEDA Lab.

2

- Goal: implement an optimized Boolean network using a given library under specific technology

```
t1 = d + e;
t2 = b + h;
t3 = a t2 + c;
t4 = t1 t3 + f g h;
F = t4';
```



- Library cells includes:

- Combinational elements:
 - Single-output functions: AND, OR, AOI
 - Compound cells: adders, decoders
- Sequential elements:
 - Registers, counters

每個紅圈都是cell library 中的一個gate (cell) · 我們需要找到最有效的轉換方式

通常一個cell library 會有約300-400個
cell · 其中很多cell在邏輯上是相同的，
只是具備不同的sizing · 根據所需推力的
不同來調用

NCTU M SEDA Lab.



3

如何挑選適當的standard cell
來implement電路?

Major Approaches

- Rule-based systems: 制定Rule · 根據不同的情況選擇對應的gate
 - Mimic designers' activity
 - Handle all types of cells
- Heuristic algorithms: 使用Heuristic找到最佳近似解
 - Restricted to single-output combinational cells
 - DAGON approach 只能套用在single-output的combinational cell
- Most tools use a combination of both

通常的tool都是兩種方法都採用

NCTU M SEDA Lab.



4

Rule-Based Library Binding

- Binding by stepwise transformations
- Data-base:
 - Set of patterns associated with best implementations
- Rules:
 - Select sub-network to be mapped 使用rule-based應對multi-fanout的gate
 - Handle high-fanout problems
- Advantages:
 - Applicable to all kinds of libraries rule-base只是單純描述轉換方式，因此通常所有的library都通用
- Disadvantages:
 - Large rule data-base
 - Completeness issue 如果rule寫得很完整，可能會造成檔案過大但如果沒寫完整就可能導致某些特別的狀況無法被轉換
 - Data-base updates

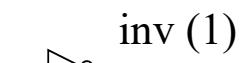


Heuristic Approach

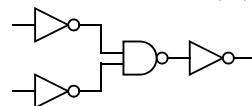
- Find a **partition** of the given network such that each sub-network can be replaced by a library cell
- General approach: 選擇base function，類似元素週期表，使用各種元素就可以組合出我們想要描述的電路
 - Choose base function set for canonical representation
 - Ex: 2-input NAND and Inverter 選擇可以表示所有邏輯的邏輯閘-->NAND
 - Represent optimized network using base functions
 - Subject graph 將想要最佳化的目標電路使用base function表示
--> subject graph
 - Represent library cells using base functions
 - Pattern graph 將cell library中的standard cell使用base function表示
--> Pattern graph
 - Each pattern associated with a cost which is dependent on the optimization criteria 紿予pattern graph中每個pattern一個cost，方便後續對應的時候計算cost
- Goal: 目標: 找到最低cost的對應方式(將subject graph用pattern組合出來)
 - Finding a minimal cost covering of a subject graph using pattern graphs



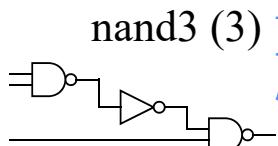
Example Pattern Graph (1/3)



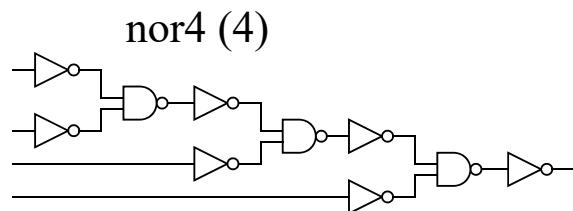
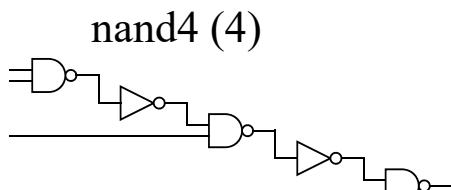
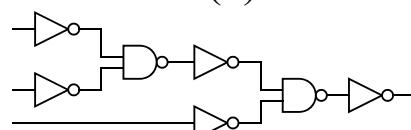
我們有的standard cell
其餘的東西都可以用這兩個cell組合出來
而組合出來的cost則需要對應實際layout大小才知道



一樣先用OR-inv，再用
DeMorgan's Rule換成NAND



可以先用AND與inv組
合出來，再把AND換
成NAND

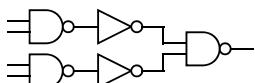


NCTU M SEDA Lab.

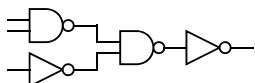


7

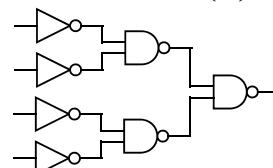
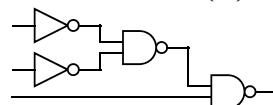
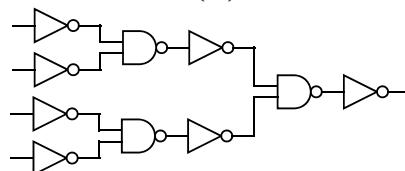
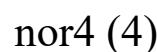
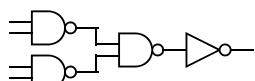
Example Pattern Graph (2/3)



aoi21原本應該是長這樣：



aoi22原本應該是長這樣：



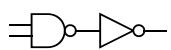
NCTU M SEDA Lab.



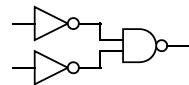
8

Example Pattern Graph (3/3)

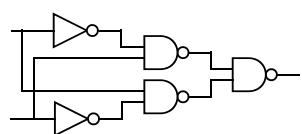
and2 (3)



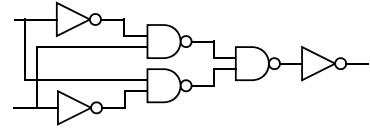
or2 (3)



xor (5)



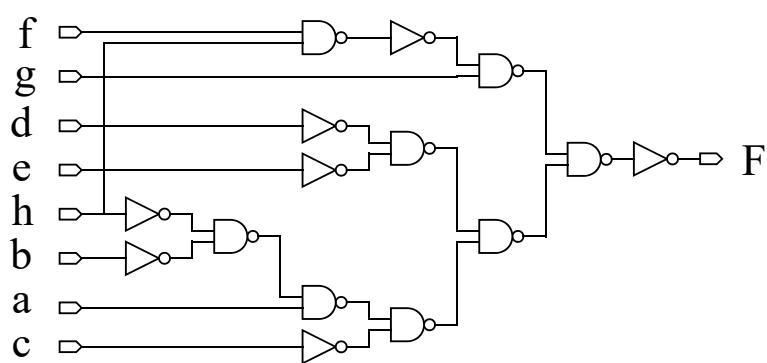
xnor (5)



Example Subject Graph

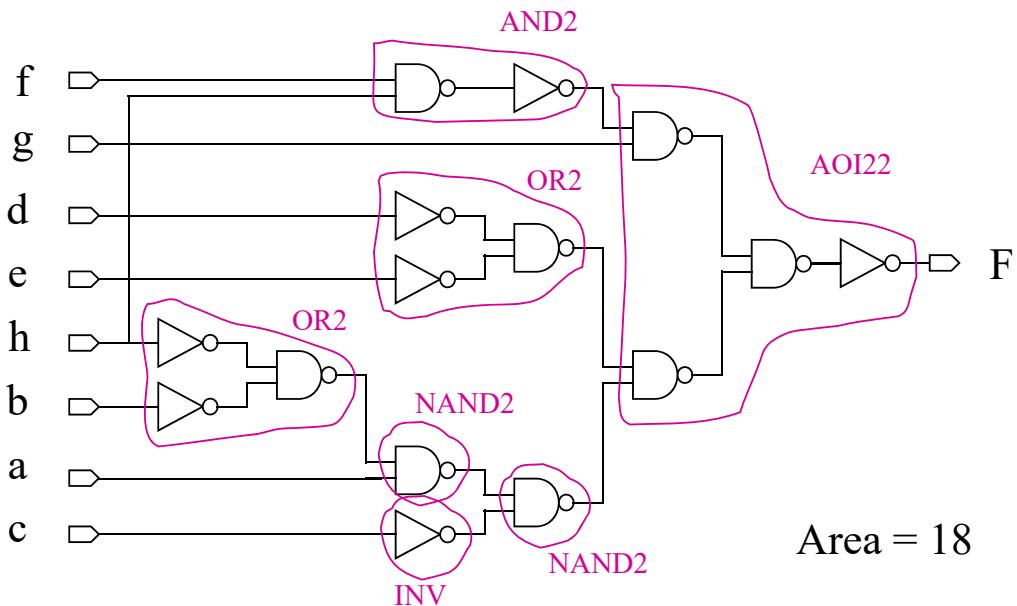
$t_1 = d + e;$
 $t_2 = b + h;$
 $t_3 = a t_2 + c;$
 $t_4 = t_1 t_3 + f g h;$
 $F = t_4';$

先將基本的電路畫出來，
然後全部換成NAND與inv
的組合



Sample Covers (1/2)

然後進行辨識，看那些東西合在一起
進行mapping可以使用最少的cost

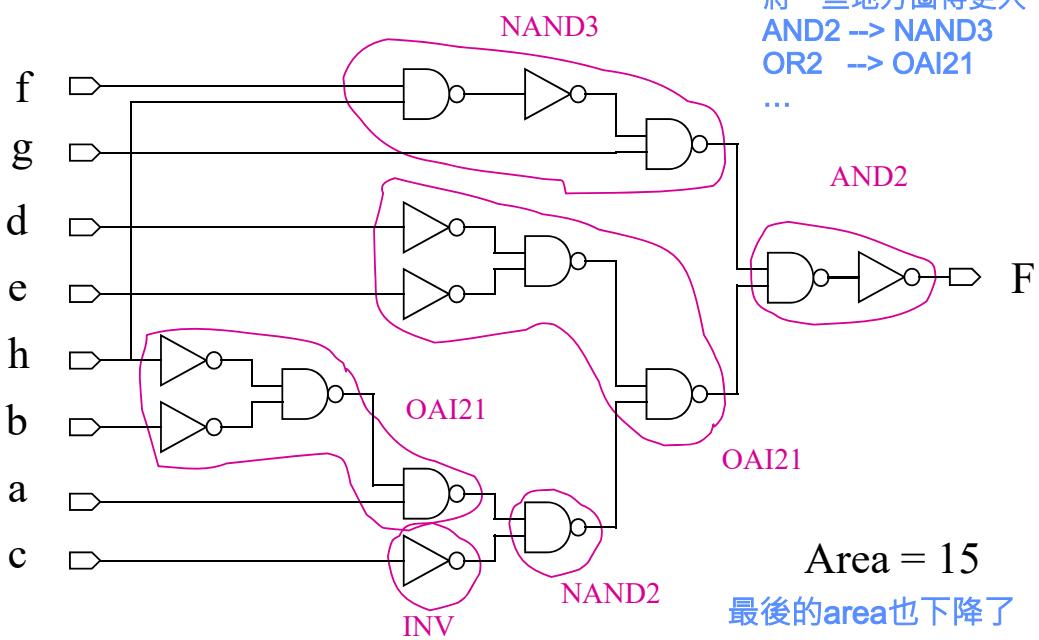


有一些東西無法group起來也沒關係，
因為NAND與inv本身都是standard cell



Sample Covers (2/2)

與上一張圖比較可以發現，這邊
將一些地方圈得更大，原本的
AND2 --> NAND3
OR2 --> OAI21
...



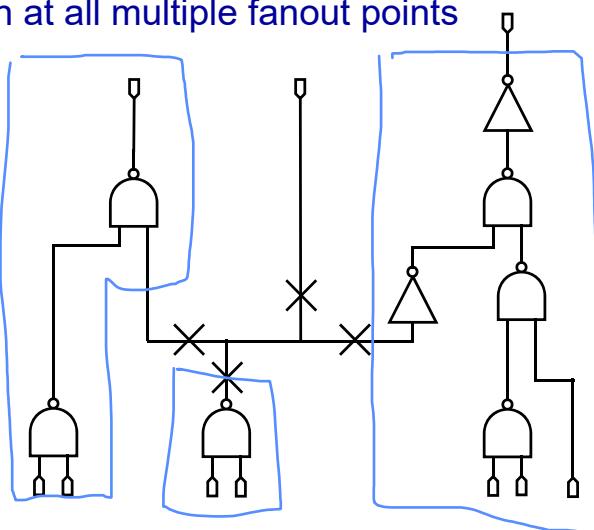
最後的area也下降了



DAGON Approach

- Partition a subject graph into trees
 - Cut the graph at all multiple fanout points

如果電路中有multi-fanout的gate，
DAGON就會將電路進行partition



- Optimally cover each tree using dynamic programming approach
- Piece the tree-covers into a cover for the subject graph

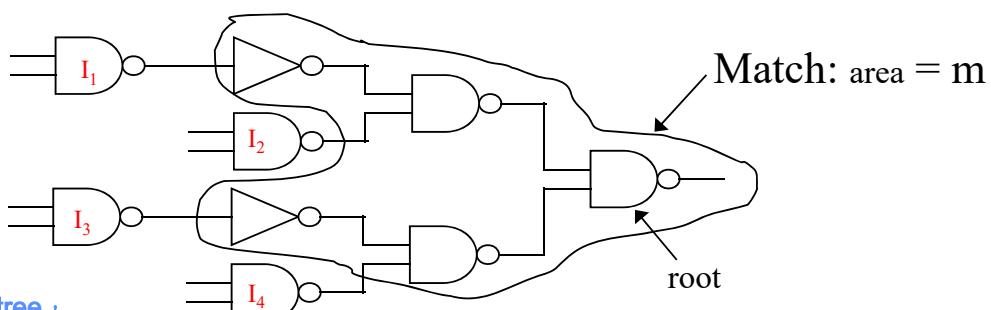
NCTU M SEDA Lab.



13

Dynamic Programming for Minimum Area

- Principle of optimality: optimal cover for the tree consists of a match at the root plus the optimal cover for the sub-tree starting at each input of the match



電路中會有很多切割好的subtree。
我們可以將從root看進去的cost表示
為recursive的形式

$$A(\text{root}) = m + A(I_1) + A(I_2) + A(I_3) + A(I_4)$$

這樣就可以套用DP來解

$$\text{cost of a leaf} = 0$$

NCTU M SEDA Lab.



14

A Library Example

這個例子中所使用的library

Library Element	Canonical Form
INV 2	a'
NAND2 3	$(ab)'$
NAND3 4	$(abc)'$
NAND4 5	$(abcd)'$
AOI21 4	$(ab+c)'$
AOI22 5	$(ab+cd)'$

NCTU M SEDA Lab.

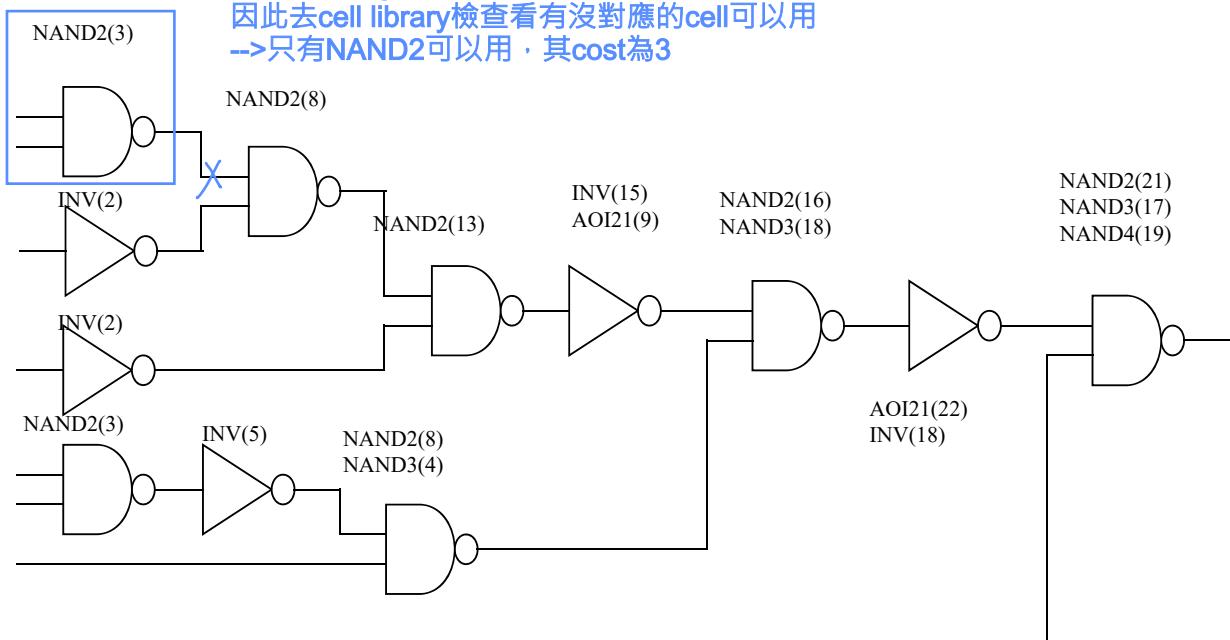


15

Bottom-Up

DAGON in Action

首先從這個gate開始向右走，由X往左看發現沒有東西，因此去cell library檢查看有沒對應的cell可以用
->只有NAND2可以用，其cost為3



NCTU M SEDA Lab.

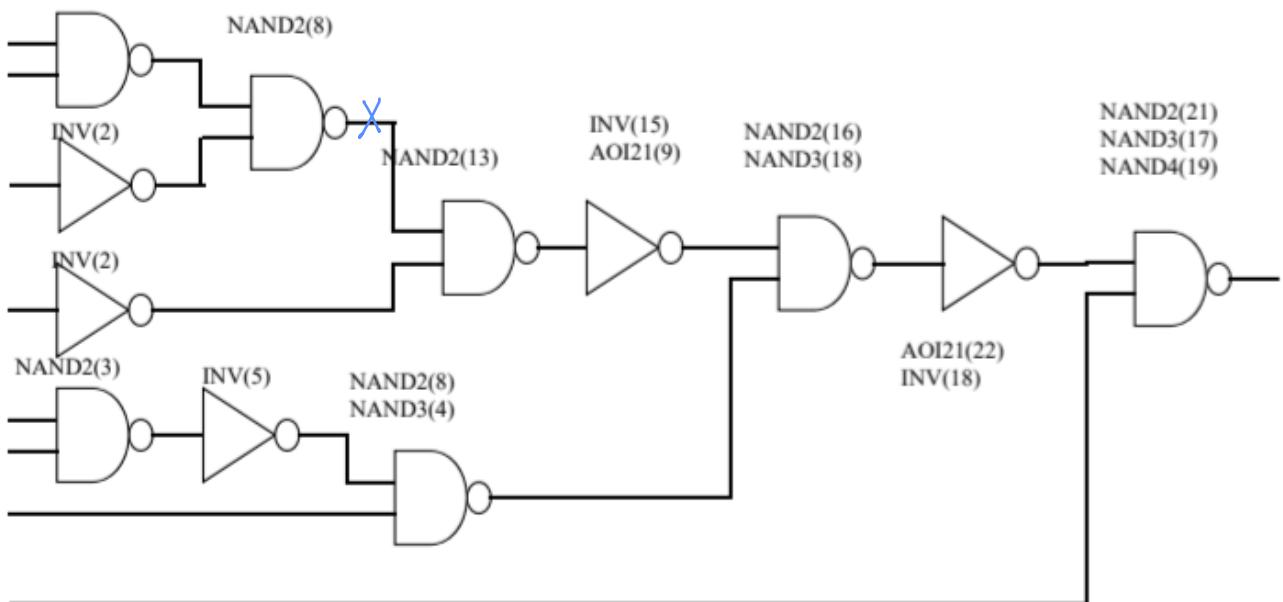


16

DAGON in Action

NAND2(3)

接下來向右走，從X處往左看，看到3個gate，去cell library尋找後發現沒有對應的gate可以將這三個gate合在一起(沒有)，因此也只能夠用NAND2而這邊的cost是X左側的gate所有child與自己的總和，因此是 $3+2+3 = 8$

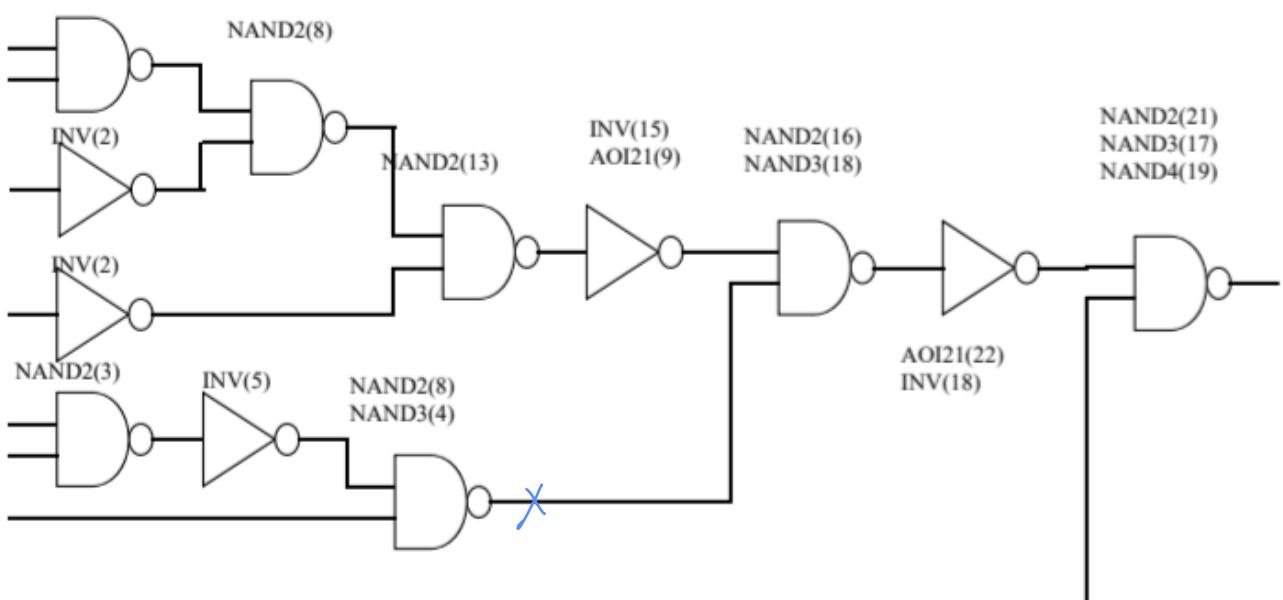


DAGON in Action

NAND2(3)

接下來走下面，由X往左看，發現有3個gate
現在有兩種選擇：

1. 使用NAND2，這樣他的cost需要將他的child(inv與一個input)與自己相加 $\rightarrow 5+0+3 = 8$
2. 使用NAND3，直接將X左側所有gate合在一起，cost為4

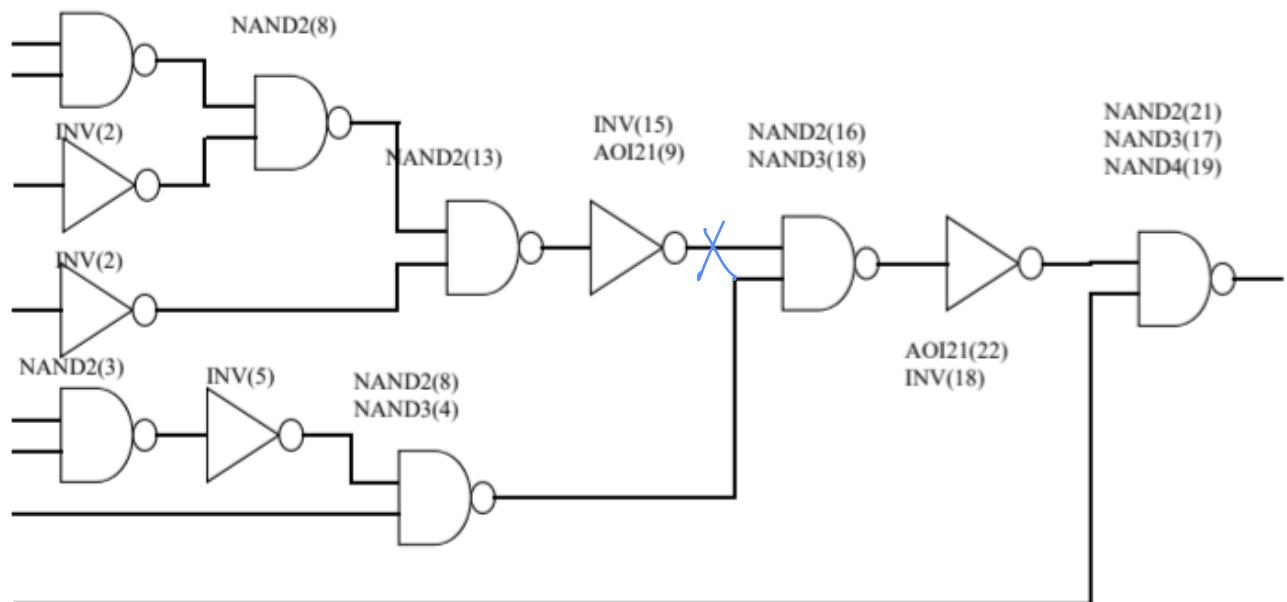


DAGON in Action

由X向左看，也可以發現有兩種組合

1. 自己一個 \rightarrow 使用inv \rightarrow cost = 13+2 = 15
2. 圈比較大圈，使用AOI21 \rightarrow cost = 9

NAND2(3)



NCTU MSED Lab.



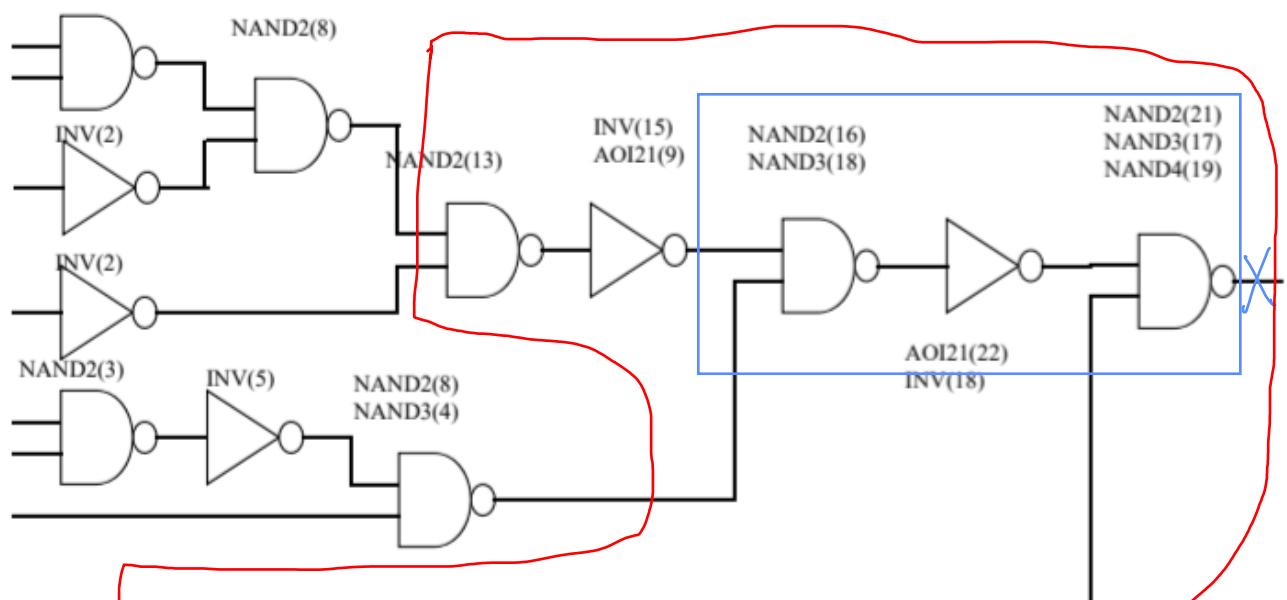
16

DAGON in Action

從output往回看，有三種組合

1. 自己使用NAND2 \rightarrow cost = min(22,18) + 3 = 21
2. 圈大一點(藍色) · 使用NAND3 \rightarrow cost = min(15,9) + min(8,4) + 4 = 17
3. 再圈大一點(紅色) · 使用NAND4 \rightarrow cost = 8+ 2 + min(8,4) + 5 = 19

NAND2(3)



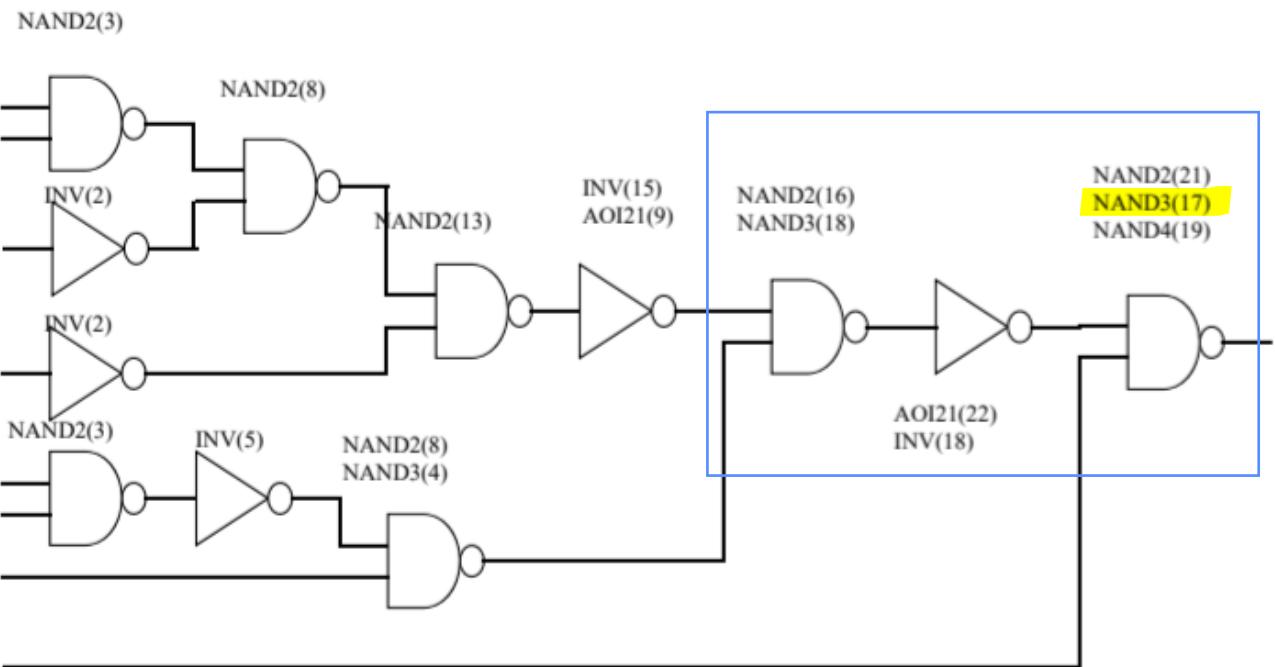
NCTU MSED Lab.



16

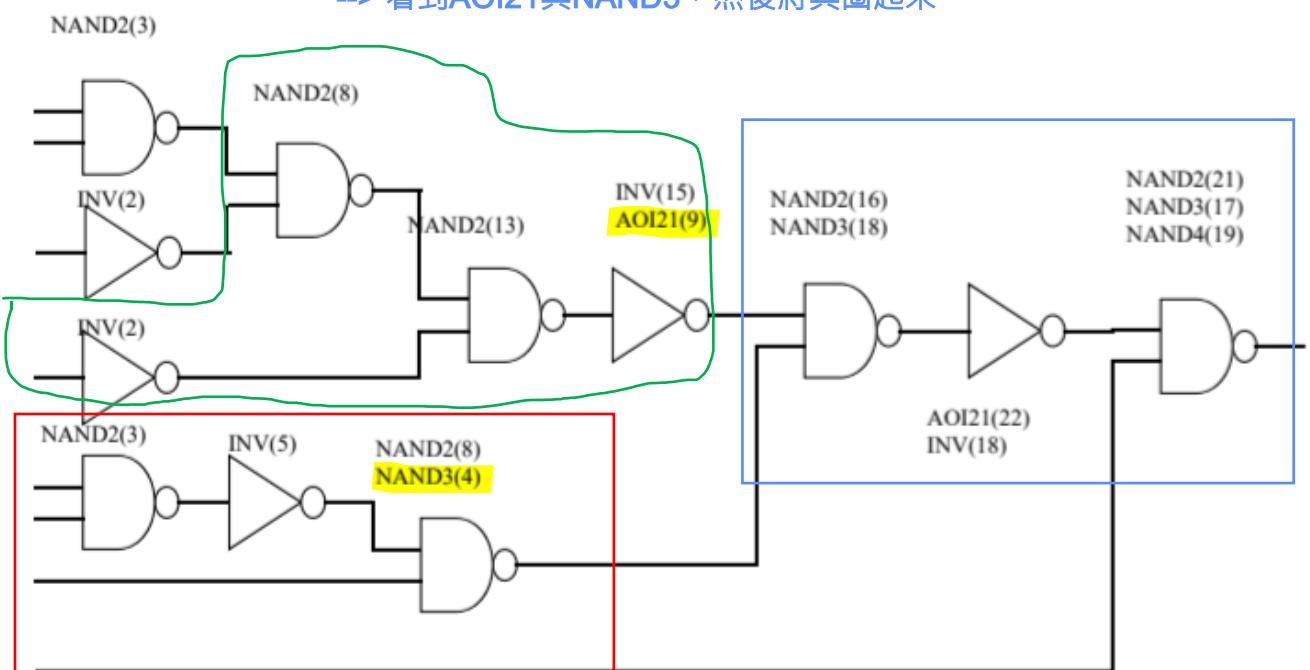
DAGON in Action

由output端往回trace，看到NAND3是最好的結果，就先往回圈出NAND3

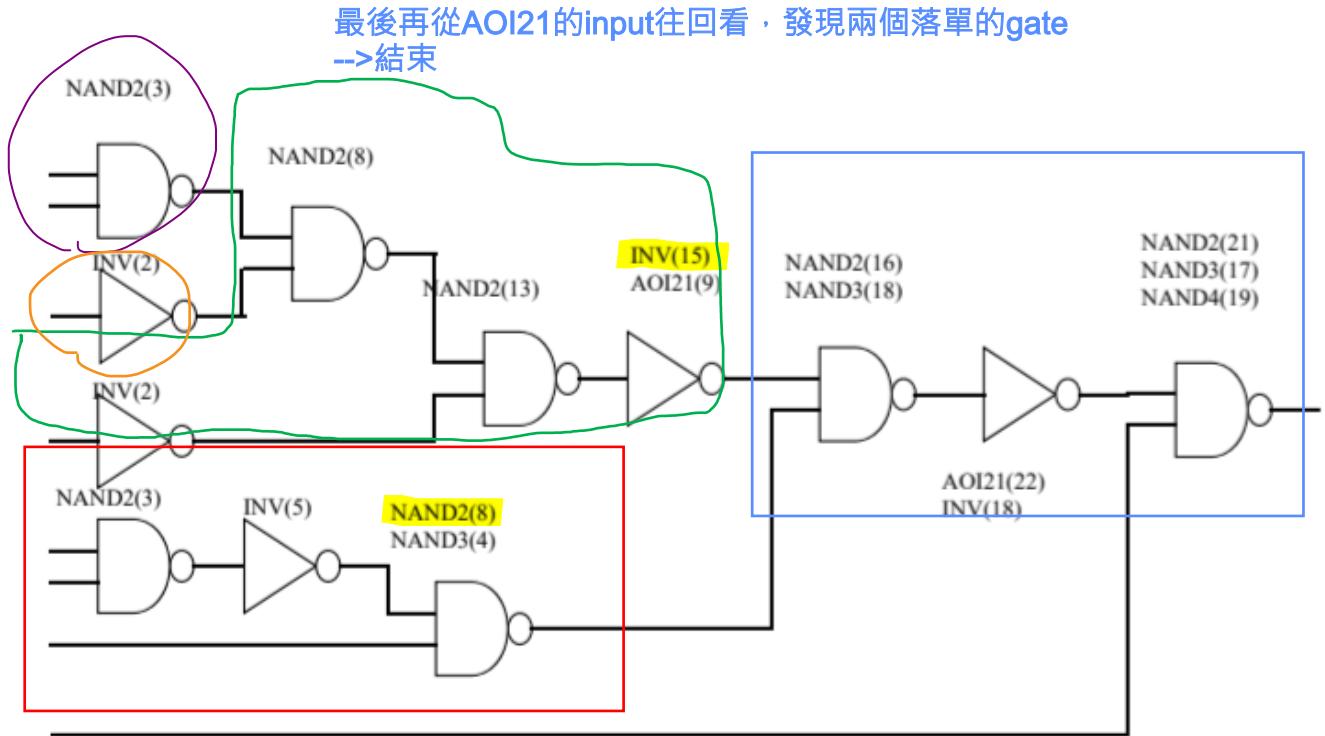


DAGON in Action

再從剛剛圈的NAND3的input往回trace，找最小的cost的組合
--> 看到AOI21與NAND3，然後將其圈起來



DAGON in Action



Features of DAGON

- Pros. of DAGON:

- Strong algorithmic foundation
- Linear time complexity
 - Efficient approximation to graph-covering problem
- Given locally optimal matches in terms of both area and delay cost functions
- Easily “portable” to new technologies 可以輕易地換成其他製程

- Cons. Of DAGON:

- With only a local (to the tree) notion of timing 只能最佳化area · 看不到loading與timing
 - Taking load values into account can improve the results
- Can destroy structures of optimized networks
 - Not desirable for well-structured circuits由於只能套用在single-fanout的電路 ·
因此如果之前有做過resource sharing
的優化(會導致multi-fanout) · 會被
DAGON切掉
- Inability to handle non-tree library elements (XOR/XNOR)
- Poor inverter allocation 如果library中有non-tree的standard cell ·
有時候加一些inverter可以使結果更好 · 但DAGON無法做到 會導致DAGON找不到這些cell

NCTU M SEDA Lab.



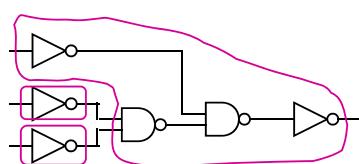
17

可以透過先將那些XOR/XNOR先轉換好 · 再將剩餘的電路使用DAGON即可

Inverter Allocation

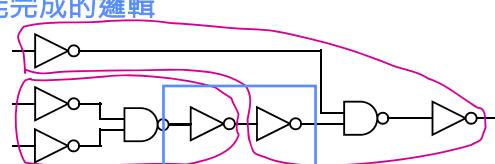
- Add a pair of inverters for each wire in the subject graph
- Add a pattern of a wire that matches two inverters with zero cost
- Effect: may further improve the solution

原本的電路:



2 INV
1 AOI21

在中間加入兩個inverter (相當於插入一個buffer)後 ·
可以發現只需要兩個NOR2就完成原本需要三個gate
才能完成的邏輯



2 NOR2

NCTU M SEDA Lab.



18

Outline

- Synthesis overview
- RTL synthesis
- Two-level logic optimization
- Multi-level logic optimization
- Technology mapping
- Timing analysis
- Timing optimization
- Synthesis for low power

NCTU M S E D A Lab.

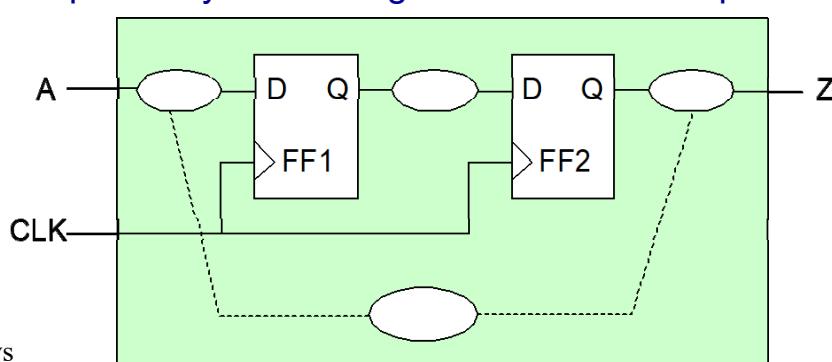


19

Functionality vs. Performance

- Functionality: correctly implements specified function
 - Mostly checked by simulation (Logic, RTL, Behavior)
- Performance: correctly implements specified function at ***specified speed***
 - Can be checked by simulation, but ...
- **Static timing analysis** (STA) is used to determine if a circuit meets timing constraints **without simulation**
 - Also a part of synthesis engine to evaluate the performance

不需要執行simulation。
直接分析電路是否可以
符合timing constraint



Reference :Synopsys

NCTU M S E D A Lab.



20

Performance Verification

- Simulation-based approaches

- Pattern-dependent
- Incomplete coverage
- Slow
- Accurate
- Impractical for large designs

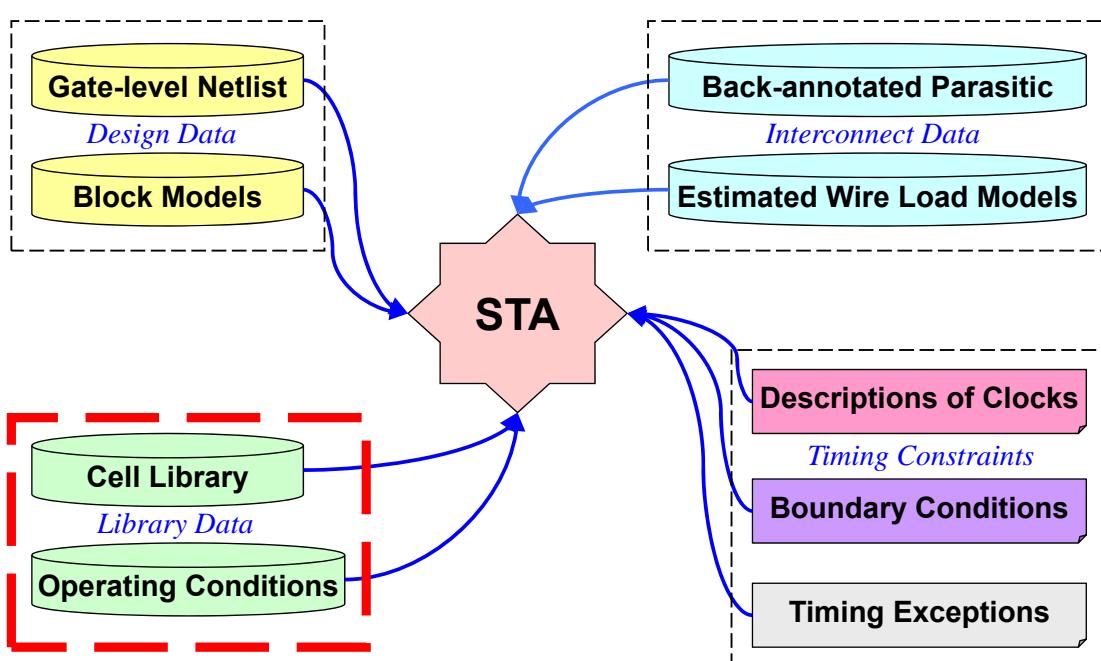
simulation-based的結果會根據input pattern的不同而有所改變，因此我們還需要額外去設計input pattern以提高coverage
對於大型design來說，使用simulation來驗證會花費大量的時間(因為需要各種不同的pattern去hit corner case)
但是simulation-base的結果會較為準確

- Static timing analysis techniques

- Functionality assumed correct STA會去分析電路中所有的timing path
因為不需要跑大量pattern，所以較快
- Pattern-independent timing check 只是STA就無法分析functionality是否正確
- Complete coverage
- Fast
- Maybe inaccurate because of false paths 有些timing path在電路實際運行時其實根本不會發生，但是STA並不能知道這件事

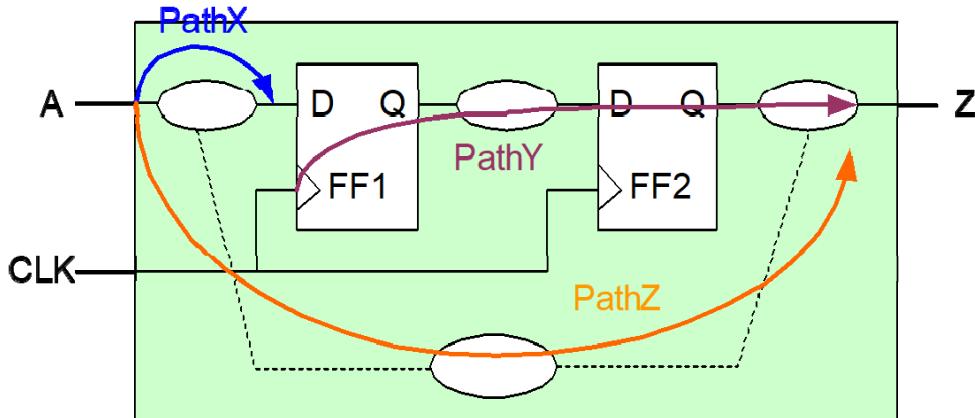


Data Preparation for STA



3 Steps in Static Timing Analysis

1. Design is broken down into sets of timing paths
2. Delay of each path is calculated
STA分析的最小單位就是path
3. Path delay are checked to see if timing constraints have been met
檢查每條path是否都滿足timing constraint
沒通過的path會寫成report



Reference :Synopsys

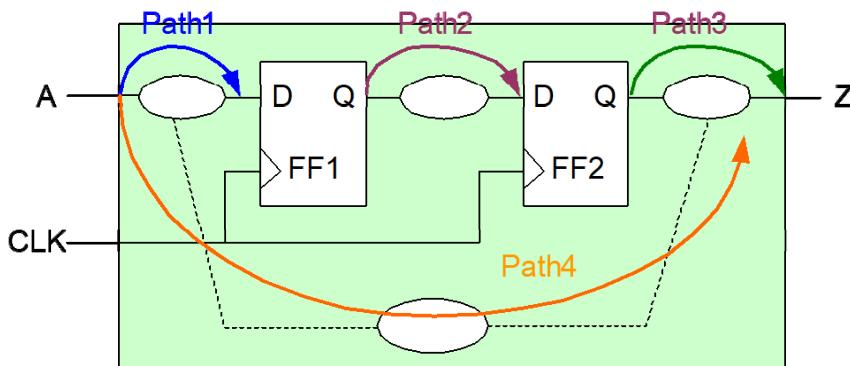
NCTU M SEDA Lab.



23

Timing Paths

- There are 4 types of timing path
 - Input port to data pin of FF (path 1) die input pin連結FF
 - Clock pin of FF to data pin of FF (path 2) tpcq+tpd
 - Clock pin of FF to output port (path 3) FF連結die output pin
 - Input port to output port (path 4) input pin經過combinational logic後直接連到output pin



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

Reference :Synopsys

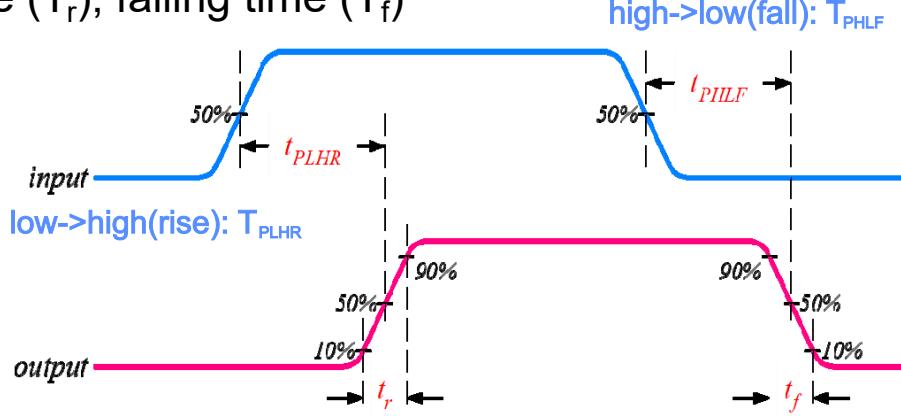
NCTU M SEDA Lab.



24

Definition of Timing Parameters

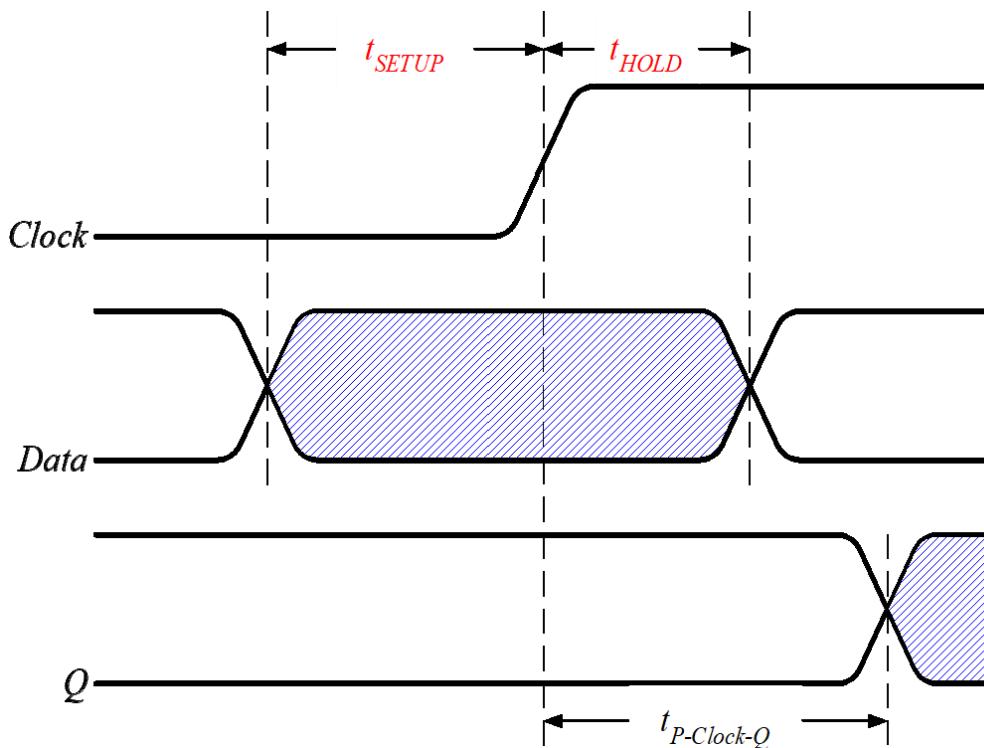
- Combinational circuits: propagation delay (T_{PLH} , T_{PHL}), rising time (T_r), falling time (T_f)



- Flip-Flop:

- **Setup Time**: The length of time that data must stabilize before the clock transition → check maximum delay path
- **Hold Time**: The length of time that data must remain stable at the input pin after the active clock transition → check min-delay path

Setup and Hold Time



Delay Model at Logic Level

1. unit delay model 直接假設每個gate的delay為1
 - Assign a delay of 1 to **each gate**
2. unit fanout delay model 修正unit delay model · 額外增加fanout對於delay的影響
 - Incorporate an **additional** delay for **each fanout**
3. library delay model 現在最常用的 · 由cell library的製造者提供
 - Use delay data in the library to provide more accurate delay value
 - May use linear or non-linear (tabular) models
4. Post layout delay model 在model中加入寄生電容(parasitic capacitance)
 - Calculated based on known net parasitic data
 - Estimate cell delay based on SDF (standard delay format) files

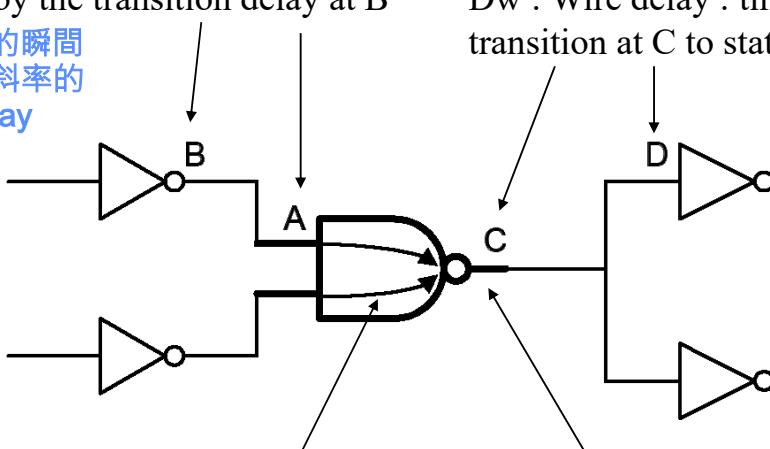


Linear Delay Model

$$\text{Delay} = D_{\text{slope}} + D_{\text{intrinsic}} + D_{\text{transition}} + D_{\text{wire}}$$

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

訊號不是理想的瞬間
從0->1 · 因此斜率的
大小會影響delay



導線的delay

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

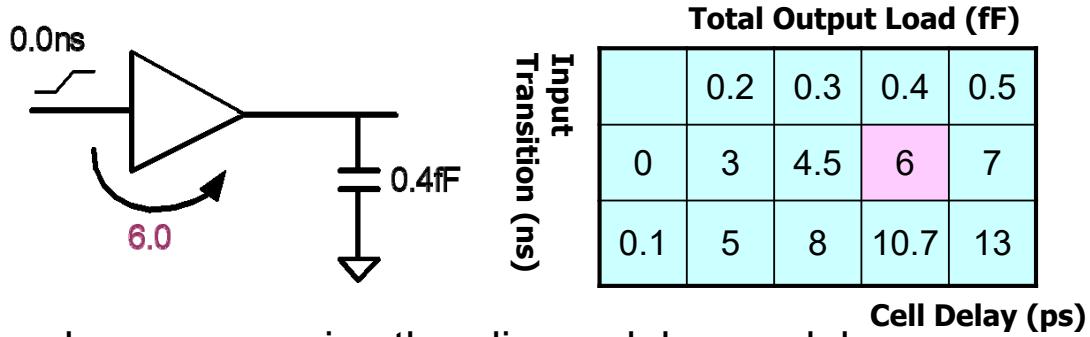
D_I : Intrinsic delay : incurred from cell input to cell output
就是gate本身的delay

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



Tabular Delay Model

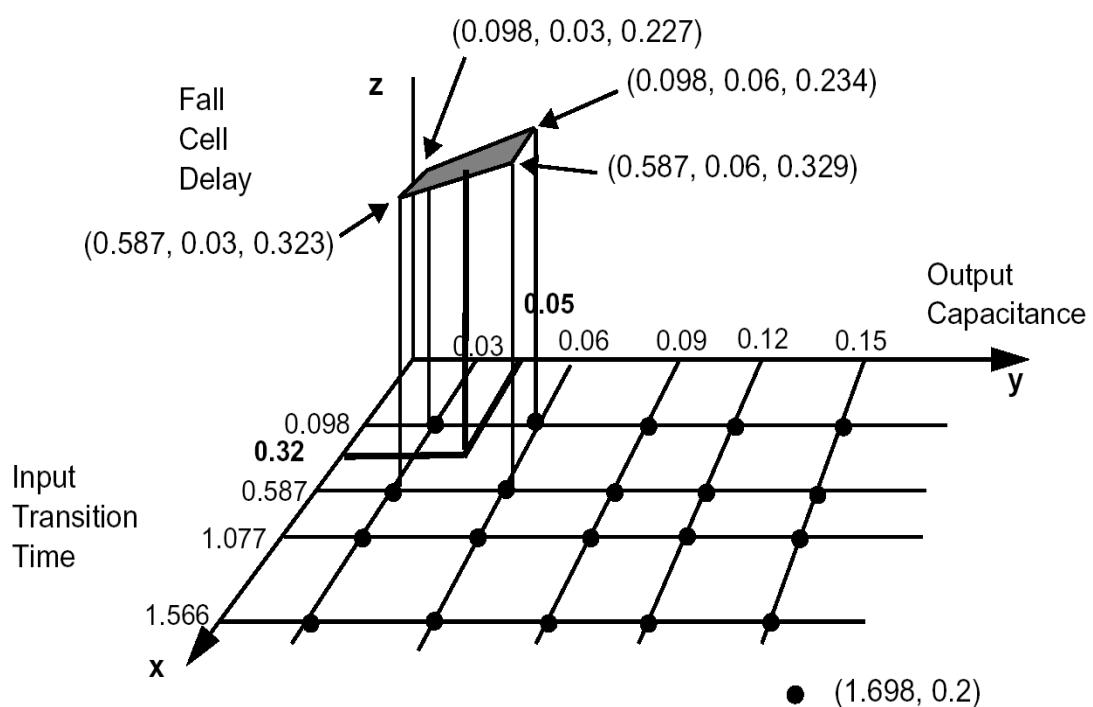
- Delay values are obtained by a look-up table
 - Two-dimensional table of delays (m by n)
 - with respect to input slope (m) and total output capacitance (n)
 - One dimensional table model for output slope (n)
 - with respect to total output capacitance (n)
 - Each value in the table is obtained by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



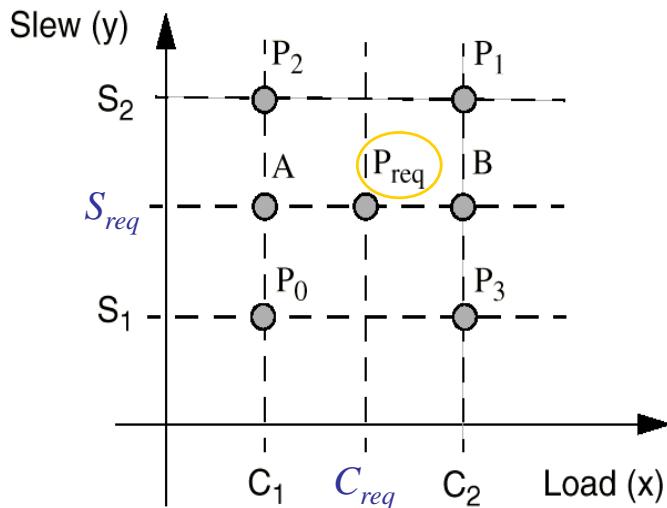
- Can be more precise than linear delay model
 - table size↑ → accuracy ↑ table建得越大，就越能涵蓋各種情況，但是占用的空間也會很大
- Require more space to store the table



Illustration of Delay Calculation



2-D Table Interpolation



$$A = P_0 + \left(\frac{P_2 - P_0}{S_2 - S_1} \right) (S_{req} - S_1)$$

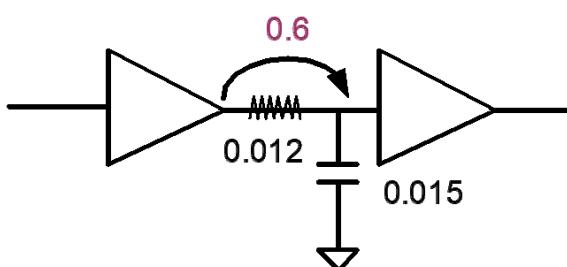
$$B = P_3 + \left(\frac{P_1 - P_3}{S_2 - S_1} \right) (S_{req} - S_1)$$

$$P_{req} = A + \left(\frac{B - A}{C_2 - C_1} \right) (C_{req} - C_1)$$



Net Delay Estimation

- Post-layout net parasitic data is extracted
 - Estimate net delay based on SDF
- Pre-layout net parasitic data **cannot** be accurately calculated 因為不知道實際走線的情況
 - Estimates net parasitic **based on a wireload model**
 - A wireload model is a set of tables
 - net fanout vs load
 - net fanout vs resistance
 - net fanout vs area



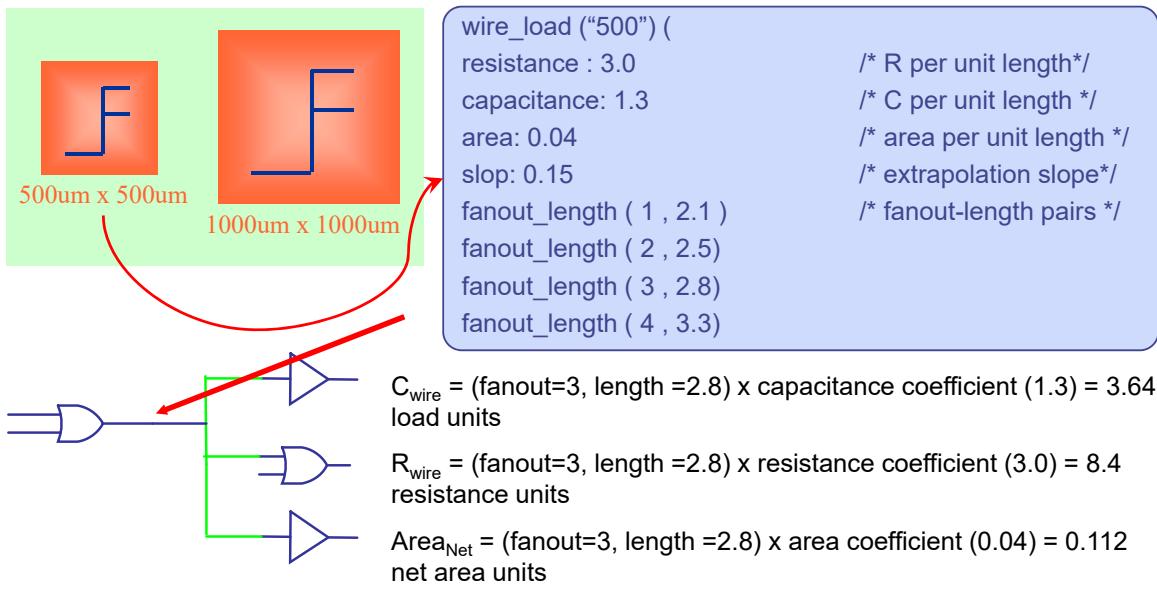
Net Fanout

	Load (fF)	Resistance (mΩ)
1	0.015	0.012
2	0.030	0.016
3	0.045	0.020
4	0.060	0.024



Wireload Model

- Determine wireload based on chip size
→ Very inaccurate!



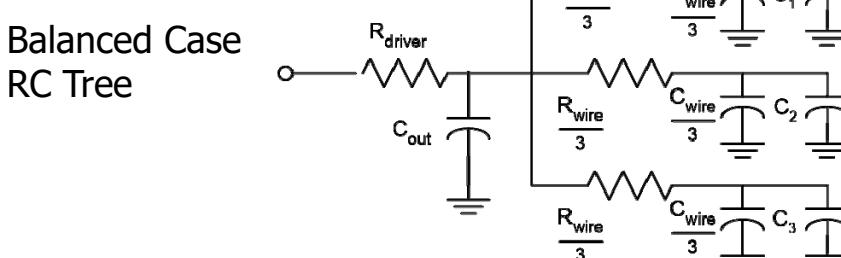
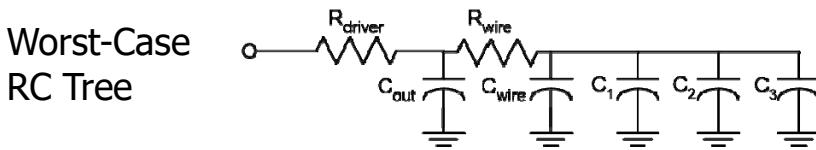
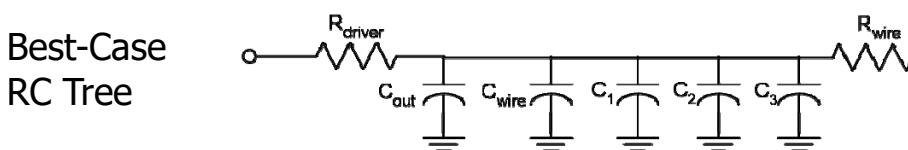
NCTU M S E D A Lab.



33

Interconnect Models

- Accurate net delay depends on the wire length
 - No such information at early design stage



NCTU M S E D A Lab.



34

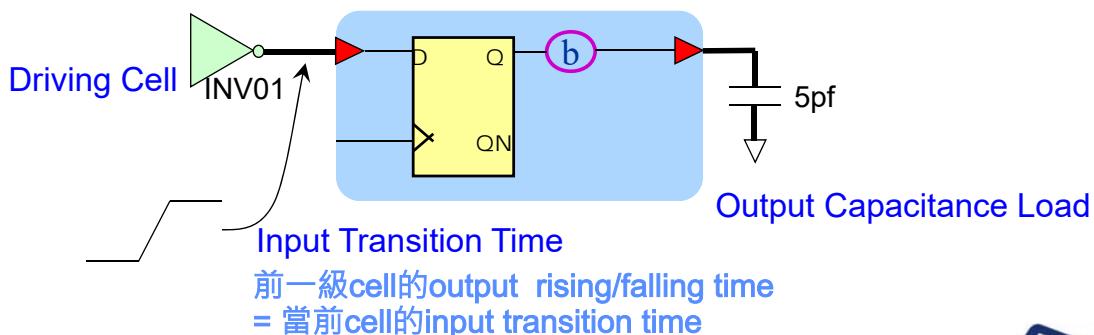
Interconnect Data

- Estimated delay information for nets based on a wire load model is used before P&R
- Back-annotated (Actual) delay information based on the P&R result is often described in the form of
 - SDF (timing information) – Standard Delay Format
 - SDF triplet: (min:typ:max) .sdf中會提供delay的range(min:typ:max)
 - RSPF – Reduced Standard Parasitic Format
 - DSPF – Detailed Standard Parasitic Format
 - SPEF – Standard Parasitic Exchange Format
 - SPEF also has syntax that allows the modeling of capacitance between different nets, so it is used in the crosstalk analysis



Boundary Conditions

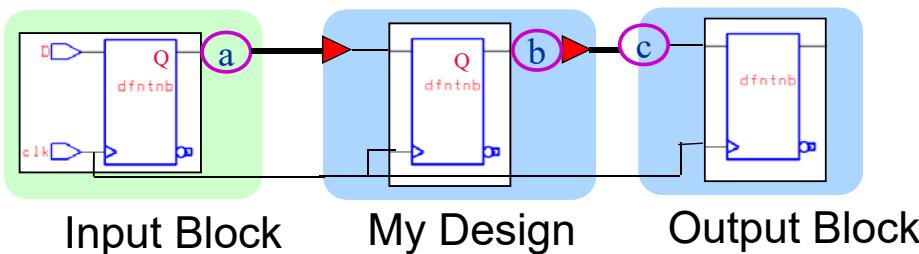
- Input driving cell
- Input transition time
- Output capacitance load
- Input delay (or Arrival Time)
- Output delay (or Required Time)



Input & Output Delay

- An input delay is the specification of an arrival time at an input port relative to a clock edge
- An output delay represents an external timing path from an output or inout port to a register

$$\text{Input delay} = \text{Delay}_{\text{clk-Q}} + a$$



$$\text{Output delay} = c$$

NCTU M SEDA Lab.



37

Arrival Time and Required Time

- arrival time : calculated from input to output 完成的時間
- required time : calculated from output to input deadline
- slack = required time - arrival time slack就是完成的時間距離 deadline還有多久

A(j): arrival time of signal j

R(k): required time or for signal k

S(k): slack of signal k

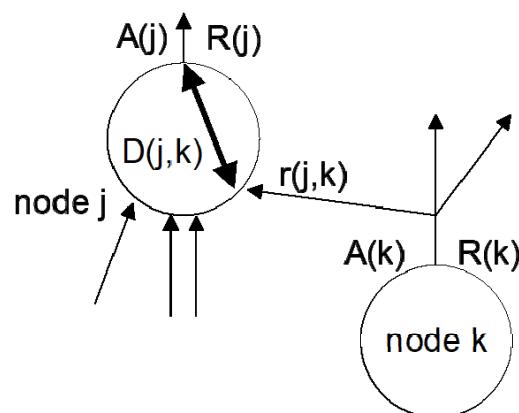
D(j,k): delay of node j from input k

$$A(j) = \max_{k \in FI(j)} [A(k) + D(j,k)]$$

$$r(j,k) = R(j) - D(j,k)$$

$$R(k) = \min_{j \in FO(k)} [r(j,k)]$$

$$S(k) = R(k) - A(k)$$



NCTU M SEDA Lab.

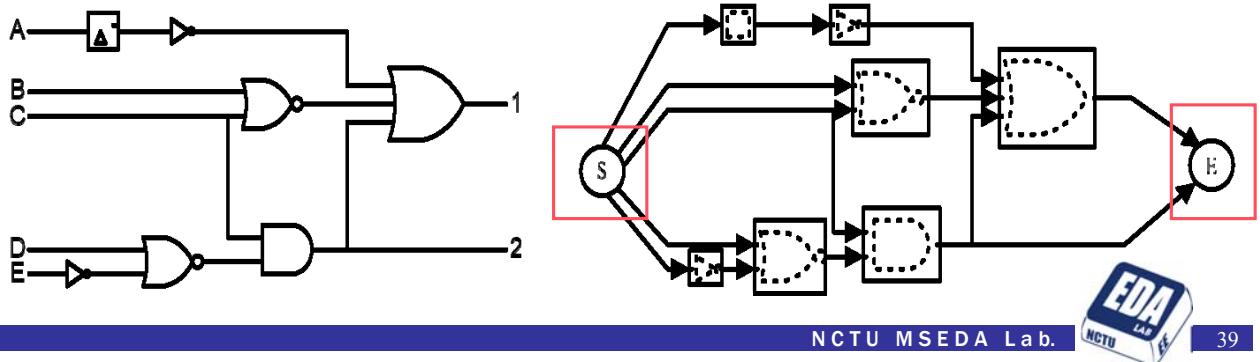


38

Delay Graph

- Replace logic gates with delay blocks
 - Add start (S) and end (E) blocks
- The delay graph is $G = (V, E)$
 - Vertices : $V = \{S, 1, 2, \dots, |V| - 2, E\}$ → logic blocks
 - Directed edges : $E = (u, v) \quad u, v \in V$ → signal flow
- Adjacency list : $Adj[u] = (v_1, \dots, v_k)$
 - $(u, v_i) \in E \quad i = 1, \dots, k$
- Number of input arcs to u : $PredCount[u]$

加入pseudo start/end
point方便計算



39

Longest and Shortest Path

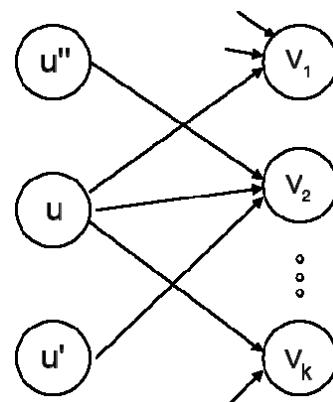
- If we visit vertices in precedence order, the following code will need executing only once for each u

Update Successors[u]

```

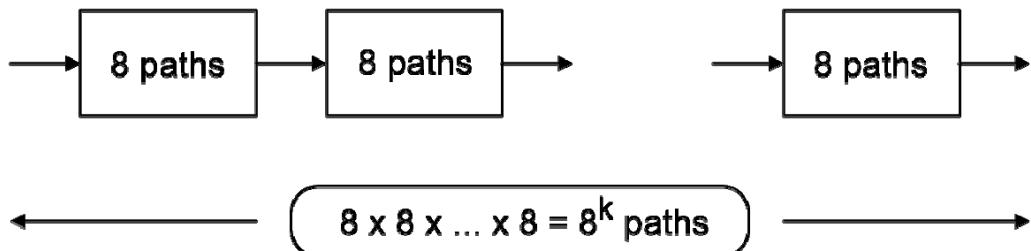
1  for each vertex  $v \in Adj[u]$  do
2      if  $A[v] < A[u] + \Delta[u]$  // longest
3          then  $A[v] \leftarrow A[u] + \Delta[u]$ 
4           $LP[v] \leftarrow u$  fi
5      if  $a[v] > a[u] + \delta[u]$  // shortest
6          then  $a[v] \leftarrow a[u] + \delta[u]$ 
7           $SP[v] \leftarrow u$  fi

```



Path Enumeration

- Exhaustive enumeration can have exponential complexity
- Consider k copies of the example DG :

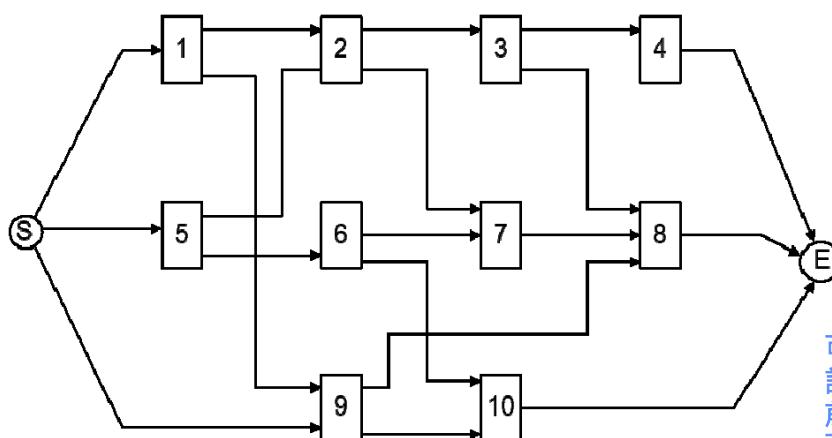


- **Topological sort** is proposed to solve this problem
 - Linear ordering of all the vertices in the DG such that if $(u, v) \in \text{DG}$ the u appear before v in the ordering

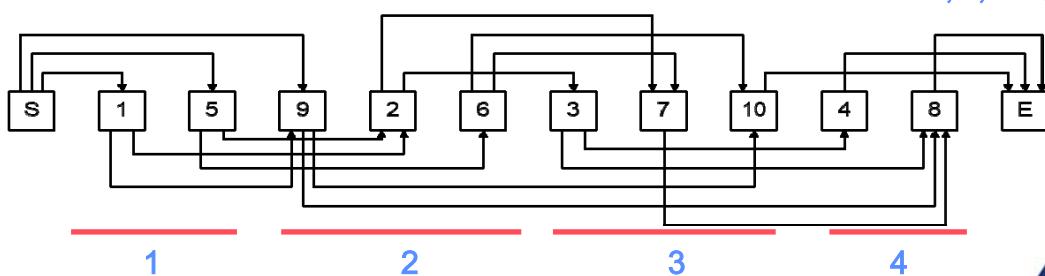
實際上不用全部的組合都試，我們只需去驗證真正具有先後關係的電路就好 \rightarrow topological sort



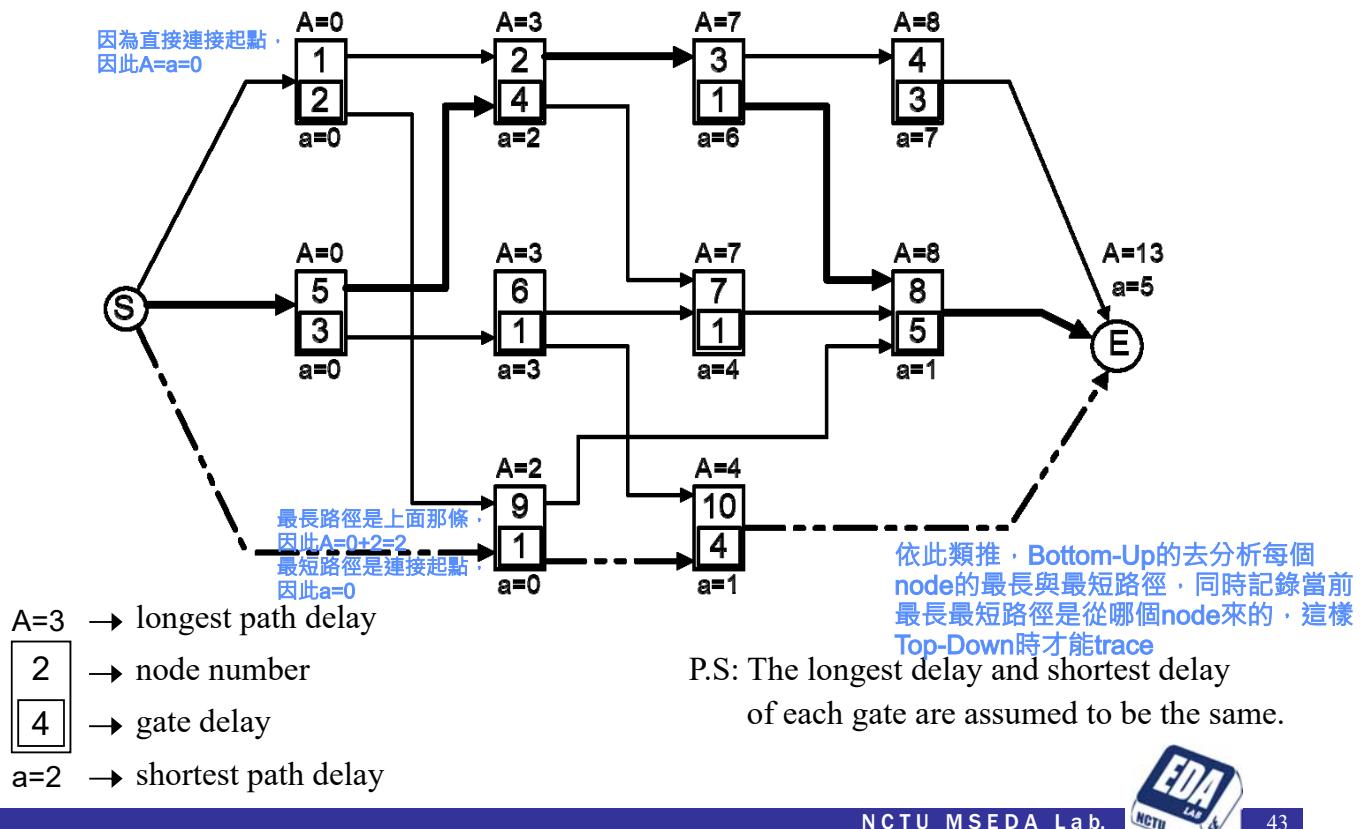
Delay Graph and Topological Sort



可以發現1, 5是最先可以被計算delay的node，因為他們前面沒有需要等待的node而第二批可以被計算的就是9, 2, 6。依此類推



Delay Calculation



NCTU M SEDA Lab.



43

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 !!

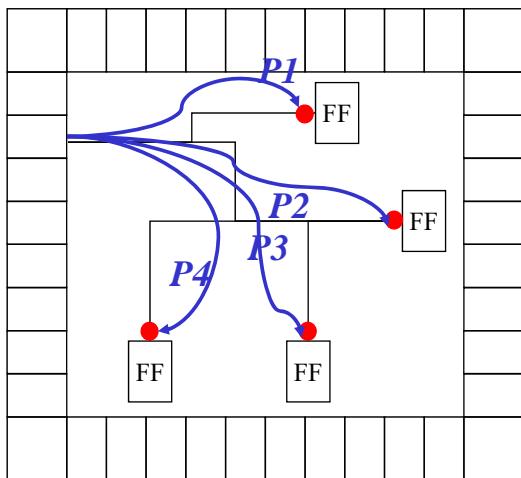


NCTU M SEDA Lab.

44

Clock Uncertainty Definition

- The maximum difference between the arrival of clock signals at sequential cells in one clock domain or between domains



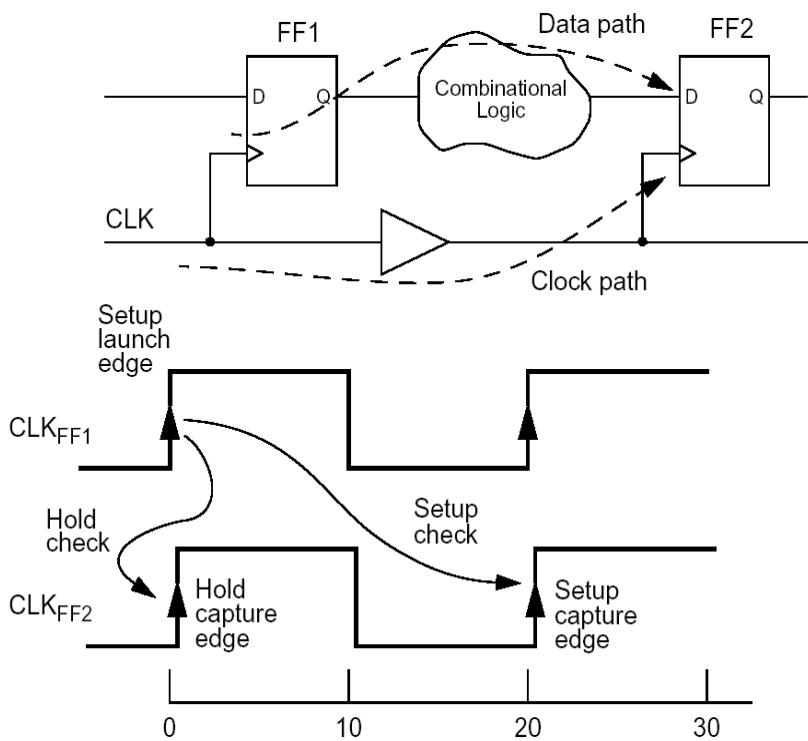
clk傳至各個FF的時間並不是一樣的，
因此會取最快與最慢到達之間的時間差
作為clk uncertainty，確保最慢接收到的
FF一樣能正常運作

Arrival(P1) = 0.5ns 最快
Arrival(P2) = 1ns
Arrival(P3) = 1.2ns
Arrival(P4) = 1.3ns 最慢

$$\begin{aligned} \text{uncertainty} \\ = 1.3 - 0.5 \\ = 0.8\text{ns} \end{aligned}$$

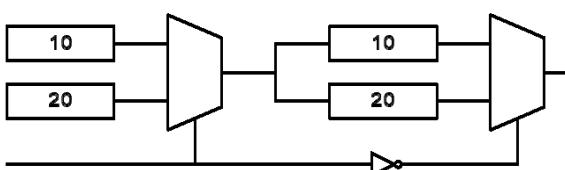


Setup and Hold Checking for FFs

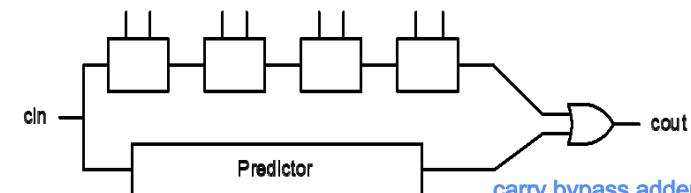


The False Path Problem

- Pattern-independent analysis calculates the delay of logic gates without **regard to their function** 由於STA並沒有實際輸入pattern進行測試，因此有些估計的path其實根本不會發生，導致估計過於保守
 - Can result in delays that are unnecessarily conservative
- Some paths are never sensitized in operation → **false path**
- Checking for false paths requires the function of the gates to be considered
 - Can approach the complexity of logic simulation or test generation
- Classical examples:



* Longest true path
has delay 30
可以看到左邊兩個mux唯二可能
出現的組合只有{0,1}與{1,0}。
因此最長路徑其實只有
 $10+20=30$
但是STA會估計為 $20+20=40$

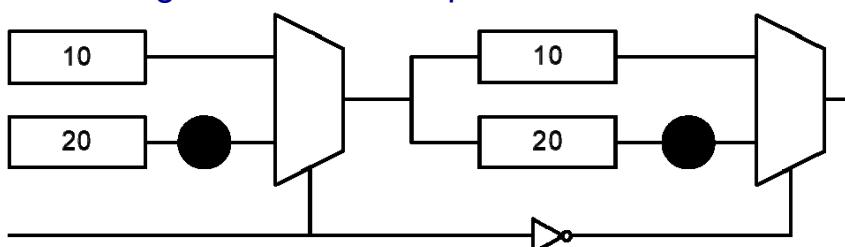


* Carry bypass adder
skip the critical path
carry bypass adder的critical path其實是下面predictor的path，但是STA會估算成上面的



Ad-Hoc Methods

- **Path blocking:** user provides a list of nodes, or pair of nodes, that lie on paths of no interest 人工標記false path
 - Pair blocking works best with path enumeration

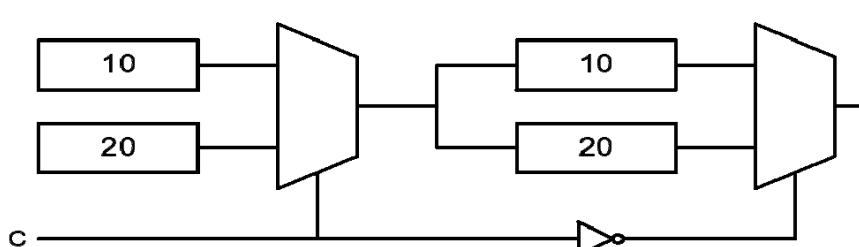


- **Case analysis:** User provides a list of cases (nodes and values) analyze the longest paths for every cases

Two cases:

C = 0

C = 1



打input進去，只需要
少部分的邏輯分析即
可去除一些false path



Path Sensitization

VALUE GATE	Controlling value	NonControlling Value
AND	0	1
NAND	0	1
OR	1	0
NOR	1	0
NOT	0 , 1	0與1都是controlling value

controlling value: 只要有任何input出現x · output就會被決定 · 就會說x是這個gate的controlling value

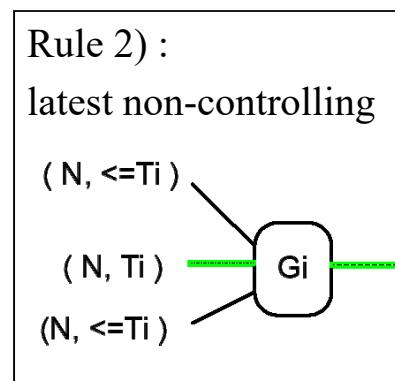
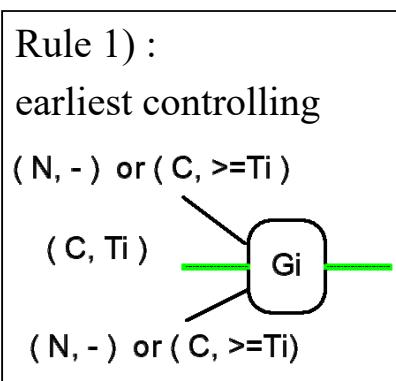
- Path $P = (I, G_1, G_2, \dots, G_n, O)$ is sensitizable if there is at least one vector which sensitizes the path



Path Sensitization

- Path $P = (I, G_1, G_2, \dots, G_n, O)$ is sensitizable under input vector V , if each gate meets either of the following rules.
一條path中只要所有gate都滿足下面其中一個rule · 就是sensitizable(意思就是只要量這條path的delay就好)

一個gate只要他的某個input出現controlling value · 他的output就可以被決定了 · 因此這個gate的delay取決於出現controlling value的那條path



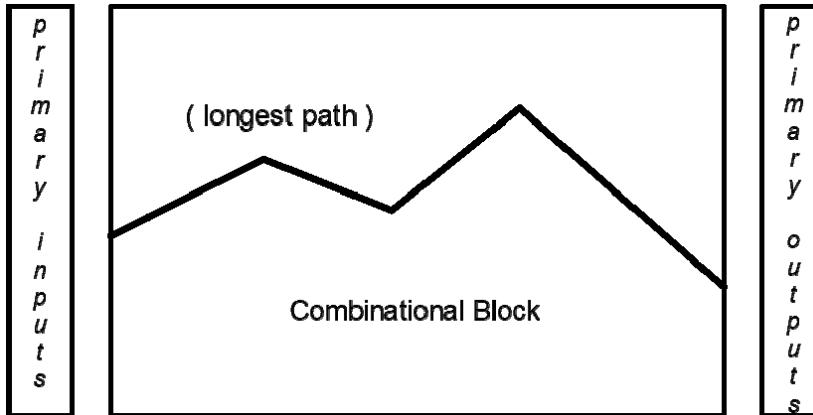
(stable value, stable time)

N : Non - controlling input , C : Controlling input
 T_i = delay of partial path $(I, G_1, G_2, \dots, G_i)$



Accurate Performance

- Actual delay of the circuit: delay of the **Longest Sensitizable Path** 因為只有sensitizable path會影響delay，因此要去量這些path並找出最長的就是電路中真正的delay
 - The longest sensitizable path can be much shorter than the longest path in a complex circuit 通常最長的sensitizable path都會比電路中的最長路徑短
 - Accuracy increases at the expense of computing efficiency 計算Longest Sensitizable Path也不會太費時



NCTU M SEDA Lab.



51

Outline

- Synthesis overview
- RTL synthesis
- Two-level logic optimization
- Multi-level logic optimization
- Technology mapping
- Timing analysis
- Timing optimization**
 - Restructuring
 - Retiming & Resynthesis
- Synthesis for low power

NCTU M SEDA Lab.



52

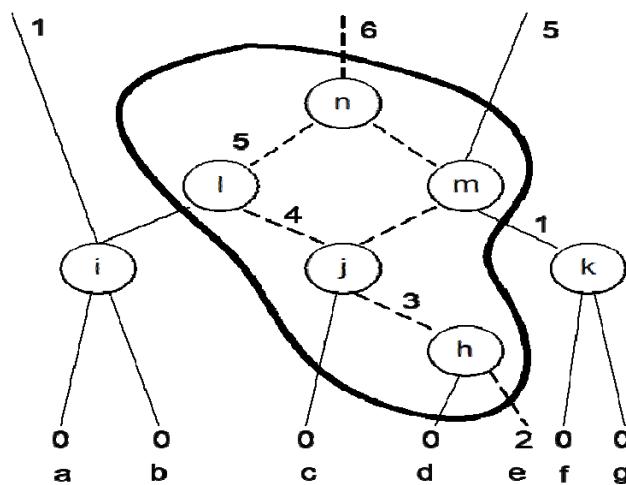
Restructuring Algorithm

While (circuit timing improves) do
 select regions to transform 選擇要重新合成的部分電路
 collapse the selected region
 resynthesize for better timing 重新合成
done

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

Restructuring Regions

- All nodes with slack within ϵ of the most critical signal belong to the ϵ -network 所有與critical signal之間的slack 小於 ϵ 的node，都屬於 ϵ -network
- To improve circuit delay, necessary and sufficient to improve delay at nodes on cut-set of ϵ -network
 - Partially collapse this region and resynthesis for better timing



Find the Cutset

- The weight of each node is $W = W_x^t + \alpha * W_x^a$
 - W_x^t is potential for speedup 這兩個參數在下一页會介紹
 - W_x^a is area penalty for duplication of logic
 - α is decided by various area/delay tradeoff
- ε : Specify the size of the ε -network ε 取太大會導致要重新合成的地方太多，取太小又可能變成沒有取到要修改的點
 - 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
 - $\alpha = 0$ implies a speedup irrespective of the increase in area
- Apply the maxflow-mincut algorithm to generate the cutset of the ε -network

NCTU M SEDA Lab.



55

$N(d)$ 越大，代表這邊類似交通樞紐(經過此處的path較多)。只要把這邊解開就可以得到較大的改善

$M(d)$ 對應的是node，如果把一個node複製，其area就會變大，因此 $M(d)$ 越大area penalty越大

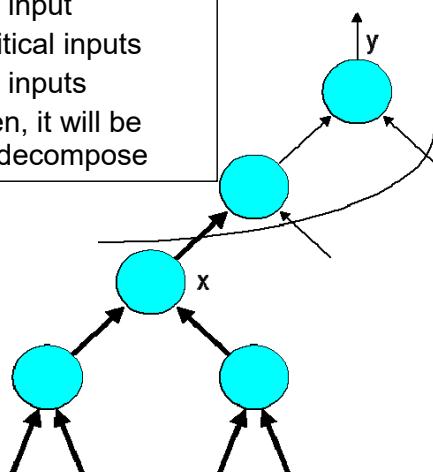
The Weight of Each Node

$$W_x^t(d) = \frac{|\{y \in N(d) | Sy \leq \varepsilon\}|}{|N(d)|}$$

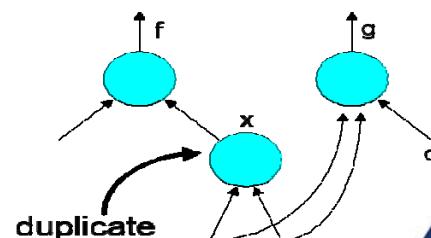
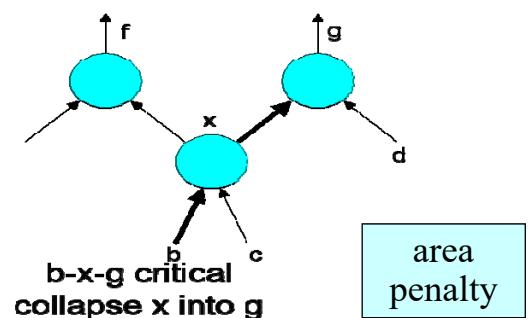
$$W_x^a(d) = \frac{|\{y \in M(d) | y \text{ is shared}\}|}{M(d)}$$

這就只是一個很簡單的指標，讓我們能夠粗略的估計timing、area之間的trade off

- Let $d = 1$ (collapsing depth)
 $y \Rightarrow 1$ critical input
 2 noncritical inputs
 $x \Rightarrow 4$ critical inputs
- If y is chosen, it will be easier to recompute



- $N(d) =$ number of inputs into resynthesis region
- $M(d) =$ number of nodes in the resynthesis region



NCTU M SEDA Lab.

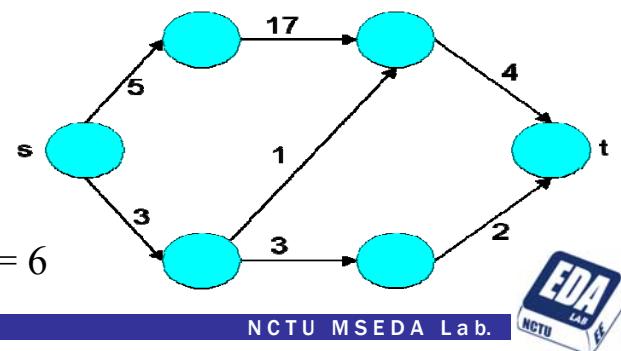


56

Maximum Network Flow

- A network $N=(s, t, V, E, b)$ is a diagram (V, E) together with a source $s \in V$ and a sink $t \in V$ with bound (capacity), $b(u,v) \in \mathbb{Z}^+$ for all edge
- A flow f in N is a vector in $\mathbb{R}^{|E|}$ such that
 1. $0 \leq f(u,v) \leq b(u,v)$ for all $(u,v) \in E$
 2. $\sum_{(u,v) \in E} f(u,v) = \sum_{(v,w) \in E} f(v,w)$ for all $v \in V - \{s,t\}$
- **Maximum network flow problem:** find the maximum allowed flow from s to t with the capacity constraints
 - Ford-Fulkerson algorithm
Complexity = $O(E|f^*|)$
 - Edmonds-Karps algorithm
Complexity = $O(VE^2)$

The value of the max flow $|f| = 6$



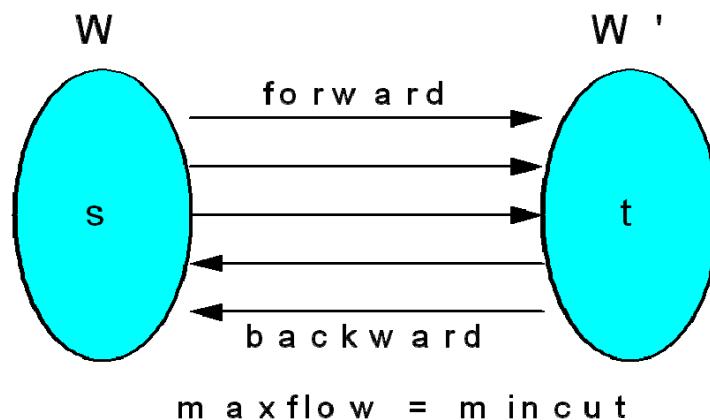
NCTU M SEDA Lab.



57

Max-Flow & Min-Cut 這兩個就是一樣的東西

- An $s-t$ cut is a partition (W, W') of the nodes of V into sets W and W' such that $s \in W$ and $t \in W'$.
- The capacity of an $s-t$ cut
$$c(W, W') = \sum_{(i,j) \in E \text{ such that } i \in W, j \in W'} b(i, j)$$
- Cut minimum nets to allow maximum flow !!



NCTU M SEDA Lab.



58

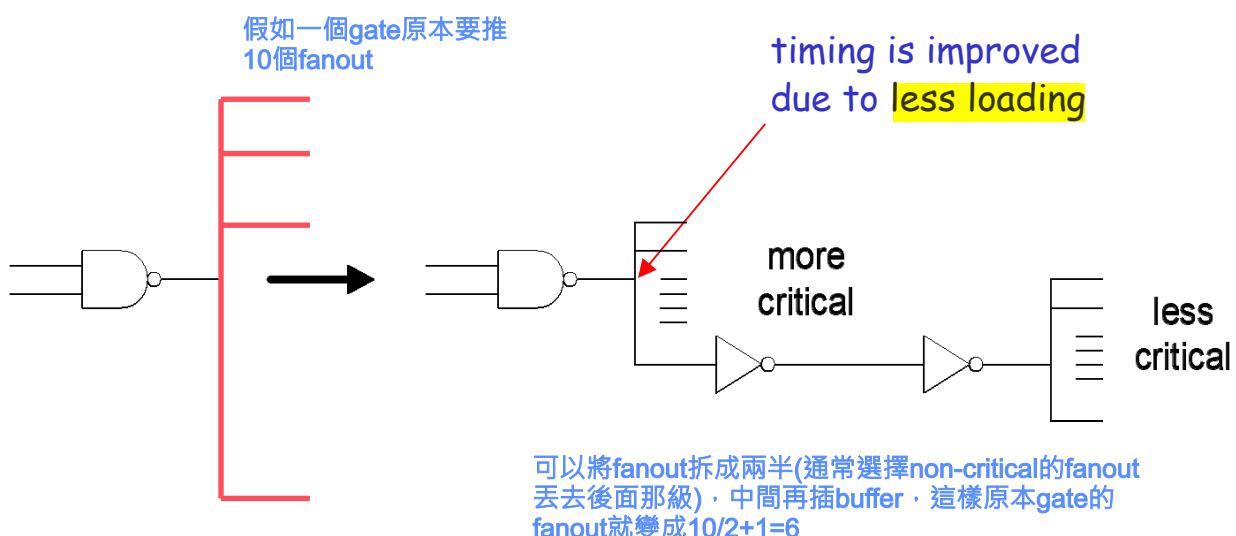
Timing Optimization Techniques (1/8)

- 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



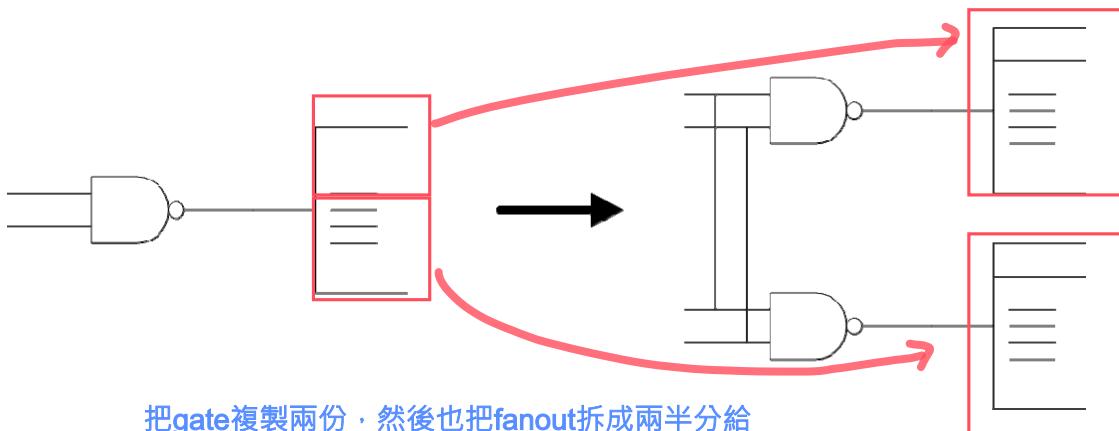
Timing Optimization Techniques (2/8)

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



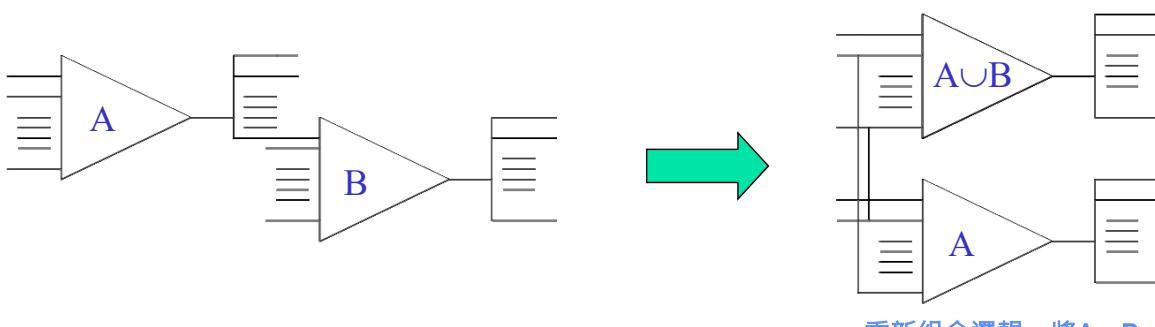
Timing Optimization Techniques (3/8)

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



Timing Optimization Techniques (4/8)

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

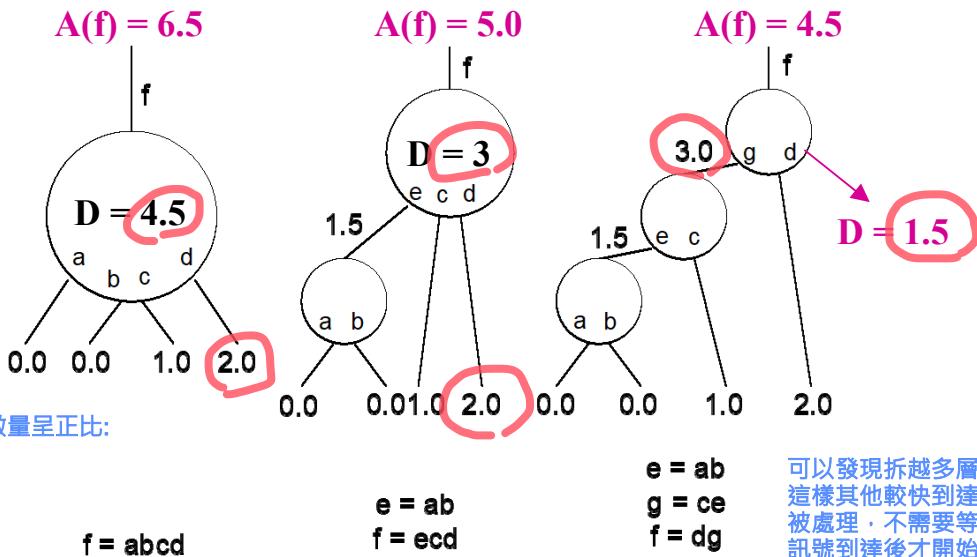


重新組合邏輯，將A、B兩個block做在一起就可以使其平行運作
-> critical path 深度縮短



Timing Optimization Techniques (5/8)

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



NCTU M S E D A Lab.



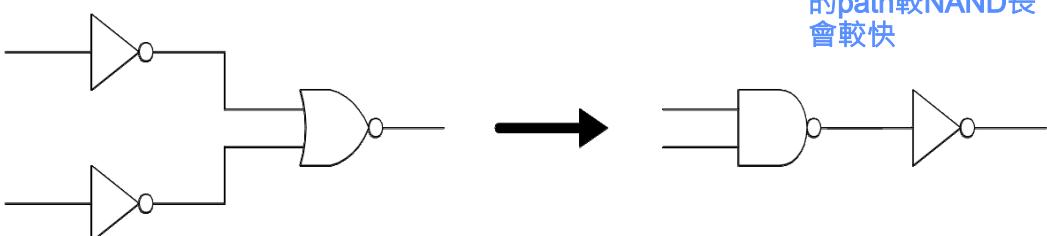
63

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

將NOR gate替換為NAND gate



pmos中majority carrier的移動速度約為nmos中majority carrier的1/2
普通CMOS中又NOR中pmos的path較NAND長，因此NAND會較快

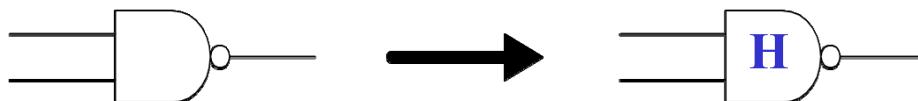
NCTU M S E D A Lab.



64

Timing Optimization Techniques (7/8)

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

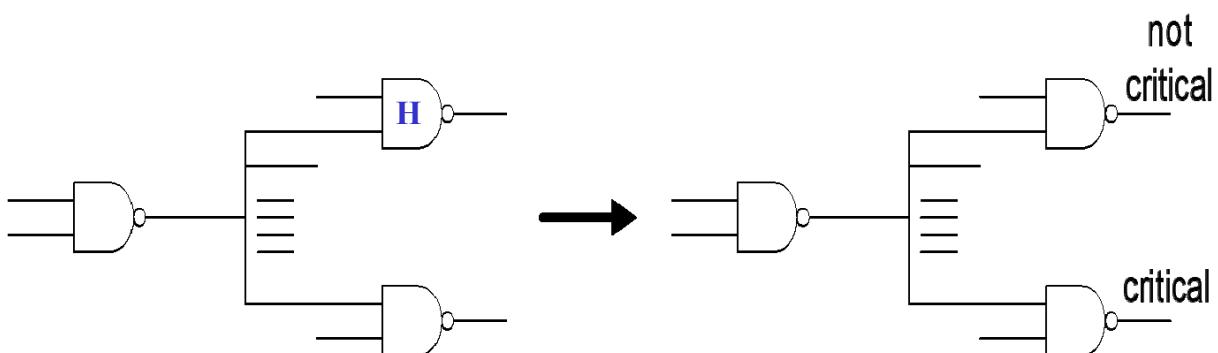


推不動就換一個sizing比較大
(推力較大)的gate

Timing Optimization Techniques (8/8)

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

當timing都met後，可以適度地調降某些non-critical path的sizing



Outline

- Synthesis overview
- RTL synthesis
- Two-level logic optimization
- Multi-level logic optimization
- Technology mapping
- Timing analysis
- Timing optimization
 - Restructuring combinational circuit
 - Retiming & Resynthesis sequential circuit
- Synthesis for low power

NCTU M SEDA Lab.

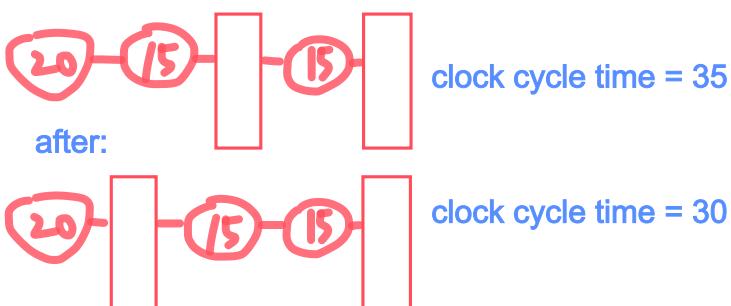


67

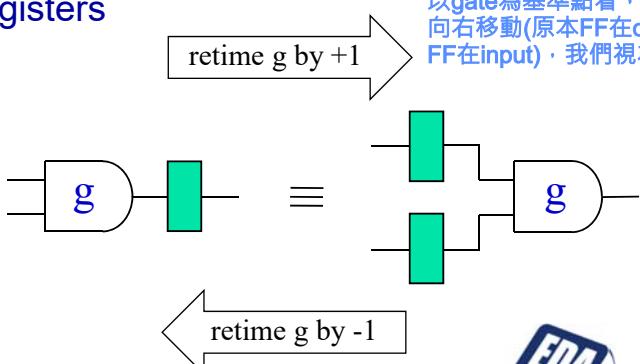
Retiming

- Exploit the ability to move registers in a circuit
 - To minimize the cycle time
 - To minimize the number of registers for a given cycle time
- Combinational logic is not modified
- Graph-Theoretic algorithms with polynomial time complexity
 - Vertex: combinational logic with propagation delay
 - Edge: number of clocked registers
 - $O(|V|^3 \lg |V|)$

before:



以gate為基準點看，如果把gate向右移動(原本FF在output，變成FF在input)，我們視為+1



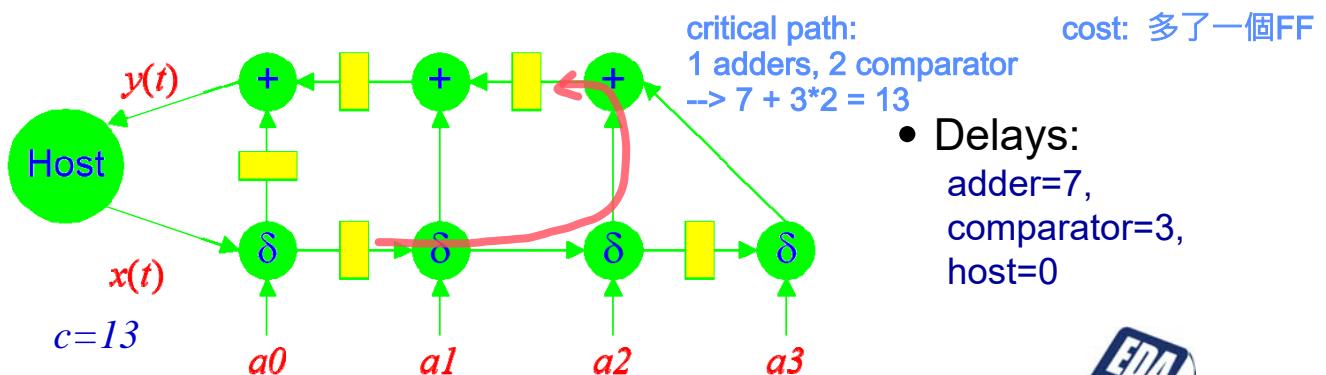
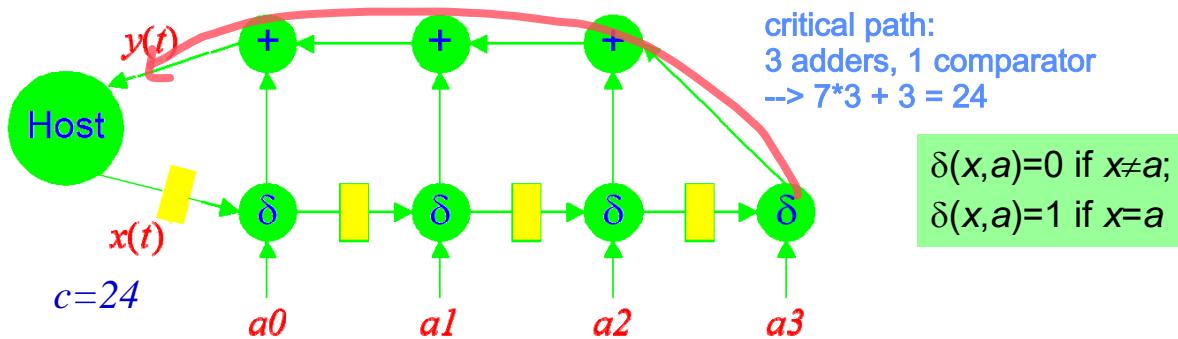
NCTU M SEDA Lab.



68

Example: Digital Correlator

- $y(t) = \delta(x(t), a_0) + \delta(x(t-1), a_1) + \delta(x(t-2), a_2) + \delta(x(t-3), a_3)$



NCTU M SEDA Lab.



69

Formulation

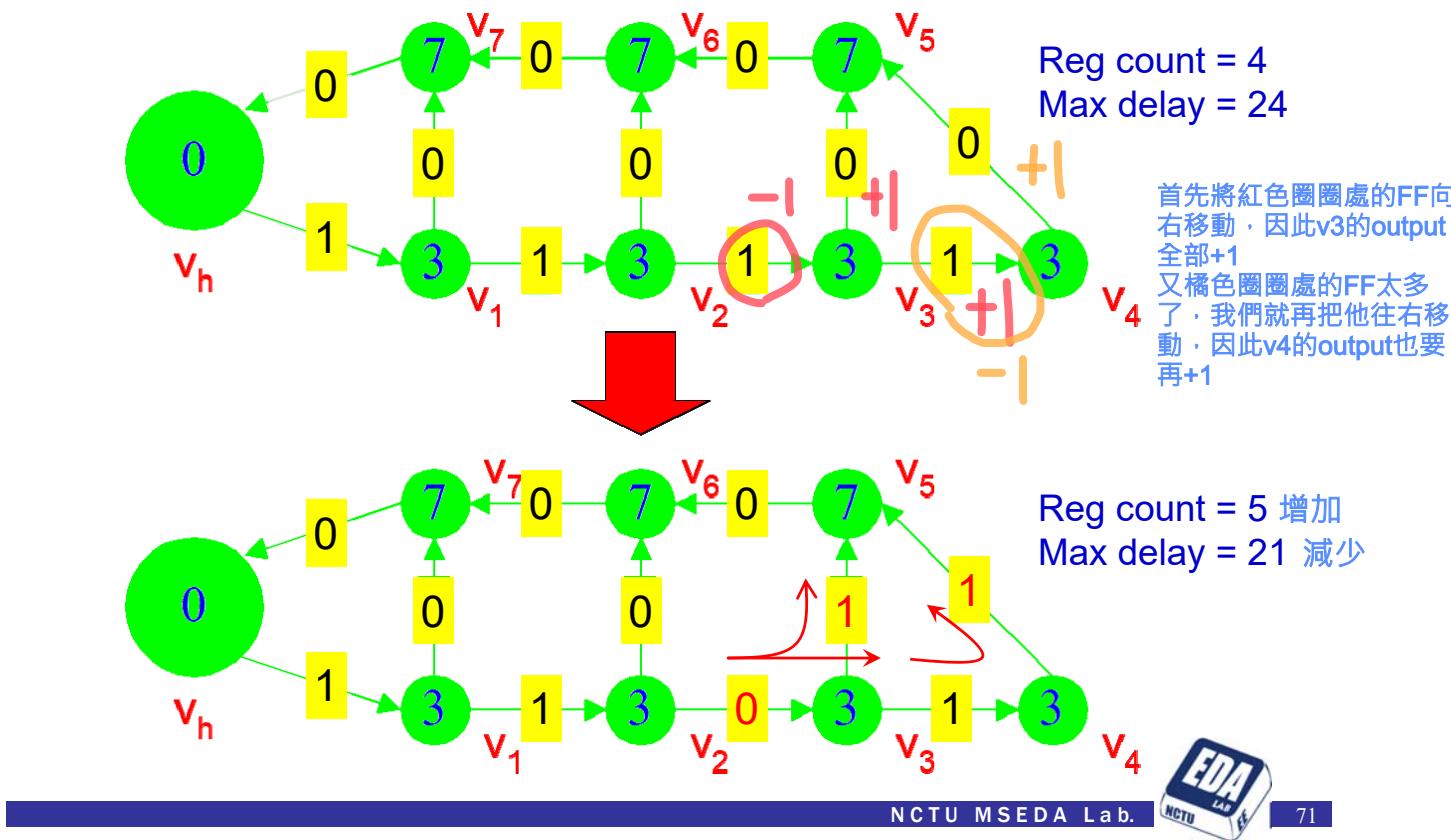
- Directed graph:
 - Nodes - **combinational logic**
 - Edges - **connections (possible latched) between logic**
- Weights
 - Nodes - **combinational logic propagation delay**
 - Edges - **number of registers**
- Path delay $d(P)$: sum of node delays along a path
- Path weight $w(P)$: sum of edge weights along a path
- **D1**: The propagation delay $d(v)$ is **non-negative** for each vertex v delay不可能是負的
- **W1**: The register count $w(e)$ is **non-negative** for each edge e register數量 ≥ 0
- **W2**: In any directed cycle, there is some edge with positive register count \rightarrow **no combinational loops** 有combinational loop
代表有latch · 因此不能有loop

NCTU M SEDA Lab.



70

Example: Relocating Registers



NCTU M SEDA Lab.



71

Problem Definition: Optimal Retiming

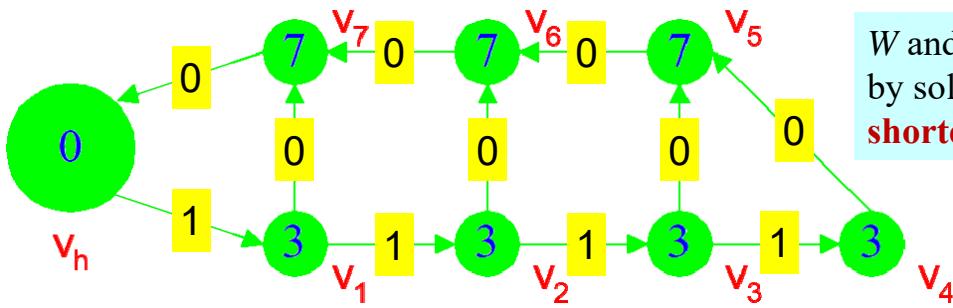
- Given a graph G , find a **legal retiming r** of G such that the **clock period $\Phi(G_r)$** of the retimed circuit G_r is **as small as possible**
- $W(u, v)$ is defined as the **minimum number of registers** on any path from vertex u to vertex v
- The **critical path p** is a path from u to v such 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

NCTU M SEDA Lab.



72

No. Registers (W) & Propagation Delay (D)



W and D can be obtained by solving **all-pair shortest-paths** problem

這個表紀錄兩個 node 之間的最少 register 數量
→ $W(u,v)$

	v_h	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_h	0	1	2	3	4	3	2	1
v_1	0	0	1	2	3	2	1	0
v_2	0	1	0	1	2	1	0	0
v_3	0	1	2	0	1	0	0	0
v_4	0	1	2	3	0	0	0	0
v_5	0	1	2	3	4	0	0	0
v_6	0	1	2	3	4	3	0	0
v_7	0	1	2	3	4	3	2	0

	v_h	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_h	0	3	6	9	12	16	13	10
v_1	10	3	6	9	12	16	13	10
v_2	17	20	3	6	9	13	10	17
v_3	24	27	30	3	6	10	17	24
v_4	24	27	30	33	3	10	17	24
v_5	21	24	27	30	33	7	14	21
v_6	14	17	20	23	26	30	7	14
v_7	7	10	13	16	19	23	20	7

這邊就是把剛剛算 W 的路徑時經過的所有 node 的 weight 加起來

Ex:
 v_h 與 v_5 之間的最少 register 數量為 3，要走的路徑是 v_1, v_2, v_3, v_5 。
因此
 $D = 3+3+3+7 = 16$



Optimal Retiming

- Let G be a synchronous circuit, and let c be the upper bound of clock period (any positive real number)
- Theorem: r is a legal retiming of G such that $\Phi(G_r) \leq c$ if and only if

1. $r(v_h) = 0$ 不會有 register 從起始點的 input 移動到 output
→ $r(v_h) = 0$

2. $r(u) - r(v) \leq w(e)$ for every edge $e(u, v)$

▪ The move-out registers should not exceed the original registers

w(e) 代表的是 register 的數量
u, v 是 e 的 endpoints

因此 $r(v)$ 是移入的 register 數量
 $r(u)$ 是移出的 register 數量
→ $w(e) + r(v) - r(u) \geq 0$

e

$r(x)$ = no. of moved registers across vertex x

把 register 從 x 的 input 移動到 output · $r(x)++$

3. $r(u) - r(v) \leq w(u, v) - 1$ for every vertices u and v if $D(u, v) > c$

▪ Breaking the long paths that exceeds the bound of clock period

- Solve the integer linear programming problem

– Bellman-Ford method in $O(|V|^3)$ 簡稱 ILP
列出方程式後根據 constraint
找出最佳的整數解

- The set of r 's determine new positions of the registers

u-v path 中，如果其 total combinational delay 已經超過我們的目標 clock cycle time (c)，則 u, v 之間一定至少存在一個 register (不然一定無法達成目標)
→ $w(u, v) + r(v) - r(u) \geq 1$



Min-Delay Retiming

- Classical algorithm to find a legal retiming r , such that the cycle time c is minimized
 1. Compute W and D P73的那兩張表
 2. Sort the elements in the range of D 為了執行Binary search
 3. Binary search the minimum achievable clock period by applying Bellman-Ford algorithm 用Binary search找到適合的 c (可以執行的最小clock cycle time)
 4. Derive the $r(v)$ from the minimum achievable clock period found in Step 3
- Complexity = Bellman-Ford ($|V|^3$) x binary search ($\lg|V|$)
= $O(|V|^3 \lg|V|)$ 複雜度很高
- Relaxation algorithm: Leiserson and Saxe [1]
 - Complexity = $O(VE)$ 將條件放寬，複雜度就下降很多

[1] C. E. Leiserson, F. M. Rose, and J. B. Saxe, “Optimizing synchronous circuitry by retiming,” in Third Caltech conference on very large scale integration. Springer, 1983, pp. 87–116.

NCTU M SEDA Lab.



75

Min-Area Retiming

- Minimize the number of registers without delay constraints
 - Minimum-cost flow problem
$$\min : \sum_{\forall e_{uv}} r(u) - r(v) \wedge (\forall e_{uv}, r(u) - r(v) \leq w_{uv})$$
- Goldberg [3]: push-relabel method
 - Complexity = $O(V^2 \log(VC))$
- A. Hurst et al. [2]: maximum network flow problem
 - Complexity = $O(R^2E)$
- Recent approach [4]: minimize area under a specific target clock period 就是將FF的數量降到最低
 - Min-area approach [2] + min-delay retiming

[2] A. P. Hurst, A. Mishchenko, and R. K. Brayton, “Fast minimum-register retiming via binary maximum-flow,” in Formal Methods in Computer Aided Design, IEEE, pp. 181–187, 2007.

[3] A. V. Goldberg, “An efficient implementation of a scaling minimum-cost flow algorithm,” Journal of algorithms, vol. 22, no. 1, pp. 1–29, 1997.

[4] A. Hurst, A. Mishchenko, and R. Brayton, “Scalable min-register retiming under timing and initializability constraints,” in Proc. 45th Design Automation Conference. ACM, pp. 534–539, 2008.

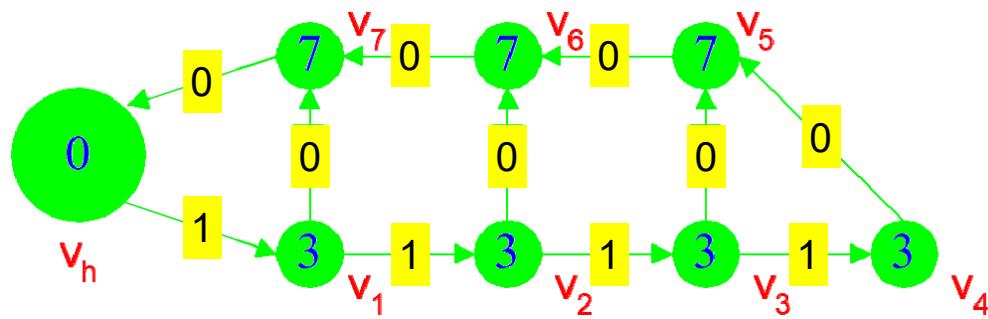
NCTU M SEDA Lab.



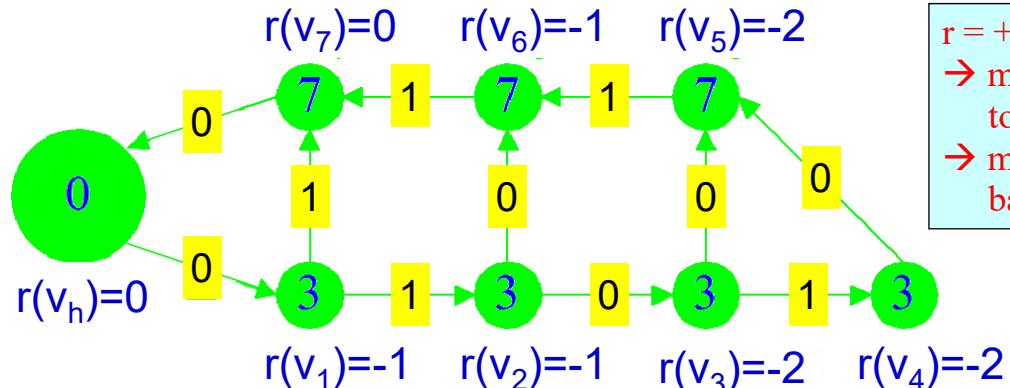
76

Retimed Correlator

Original:

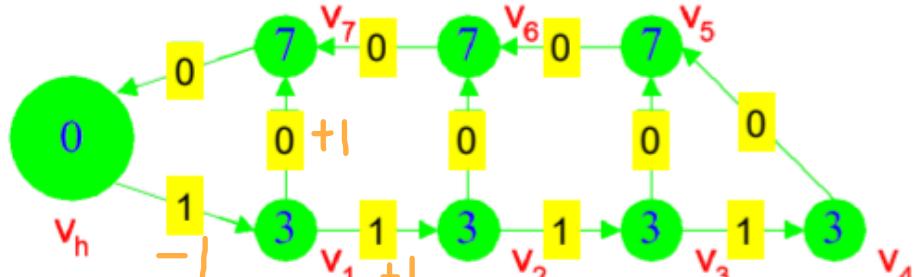


Result:

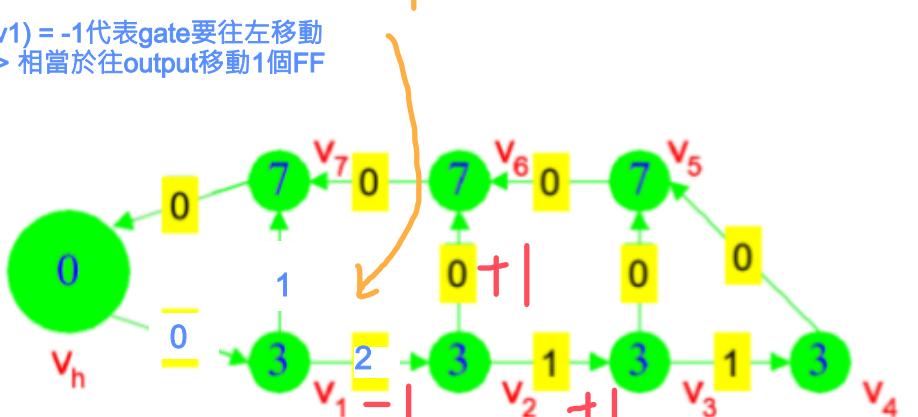


Retimed Correlator

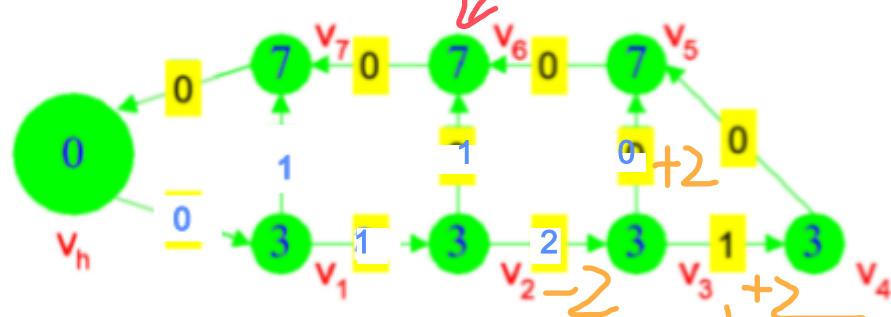
Original:



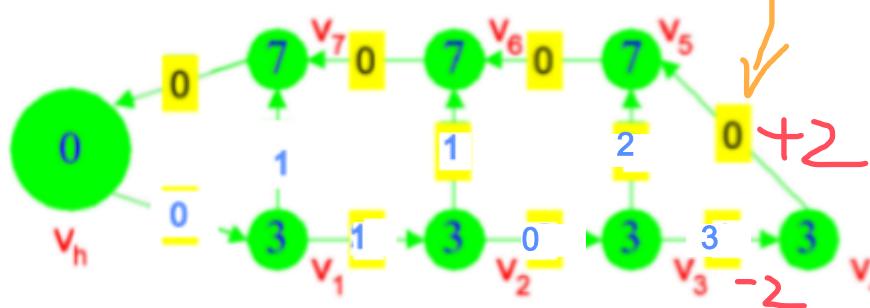
Step 1: $r(v_1) = -1$ 代表 gate 要往左移動
→ 相當於往 output 移動 1 個 FF



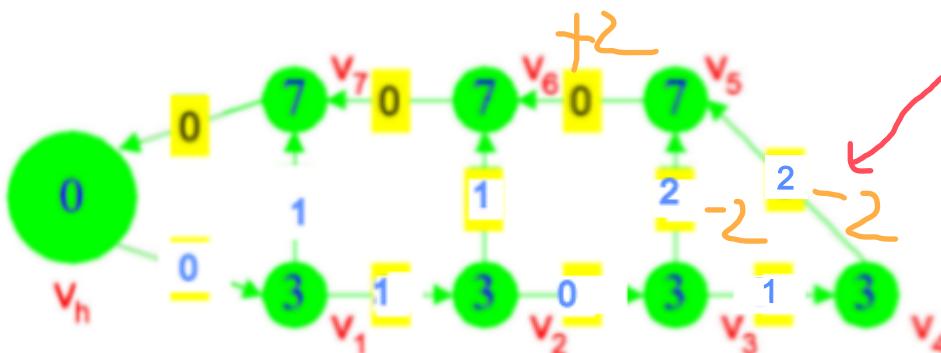
Step 2: $r(v_2) = -1$ 代表 gate 要往左移動
→ 相當於往 output 移動 1 個 FF



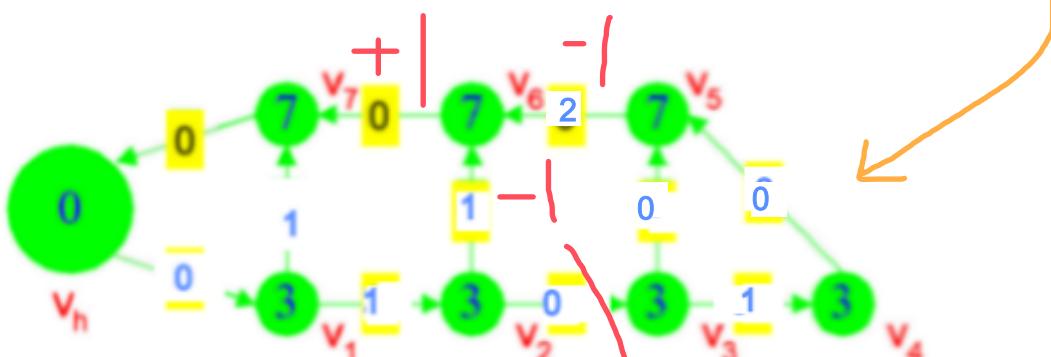
Step 3: $r(v_3) = -2$ 代表 gate 要往左移動
→ 相當於往 output 移動 2 個 FF



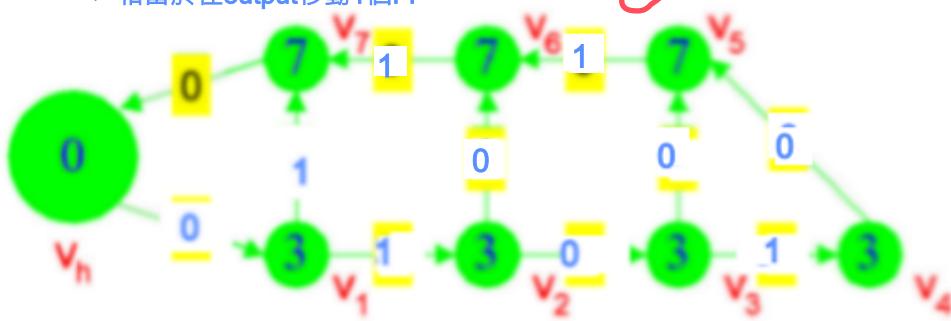
Step 4: $r(v_4) = -2$ 代表 gate 要往左移動
→ 相當於往 output 移動 2 個 FF



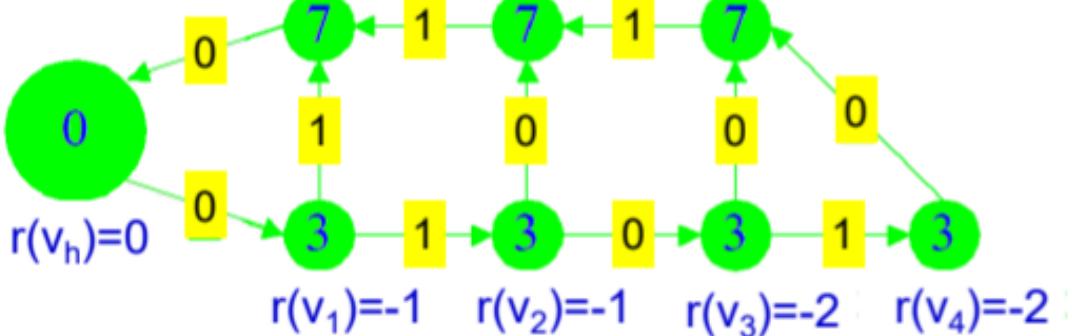
Step 5: $r(v_5) = -2$ 代表 gate 要往左移動
→ 相當於往 output 移動 2 個 FF



Step 6: $r(v_6) = -1$ 代表gate要往左移動
→ 相當於往output移動1個FF



$r(v_7) = 0$ 代表不用移動FF
Result:

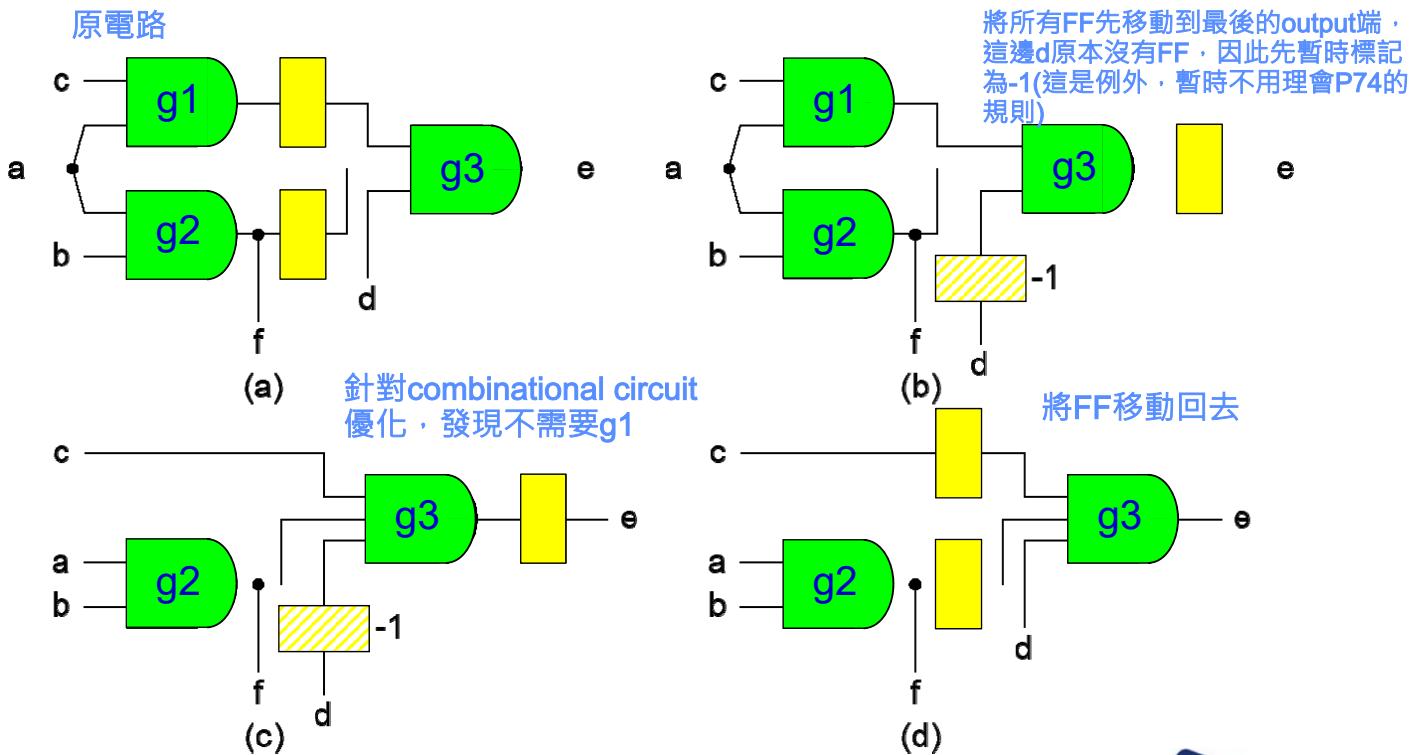


Retiming and Resynthesis

- Retiming offers the opportunity to resynthesize the combinational logic for further reduction [5]
- 3 Steps:
 1. Migrate all registers to the periphery of a sub-network
 - Peripheral retiming 先把所有FF都移到最旁邊去，這樣整個電路就相當於一個超大combinational circuit
 2. Optimize the sub-network with combinational technique
 - Resynthesis 利用優化combinational circuit的演算法優化電路
 3. Replace registers back in the sub-network
 - Retiming 最後再將FF移動回電路中

[5] S. Malik, E. Sentovich, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimization of Sequential Networks with Combinational Techniques," IEEE Transactions on Computer-Aided Design, January 1991.

Example: Retiming + Resynthesis



Peripheral Retiming

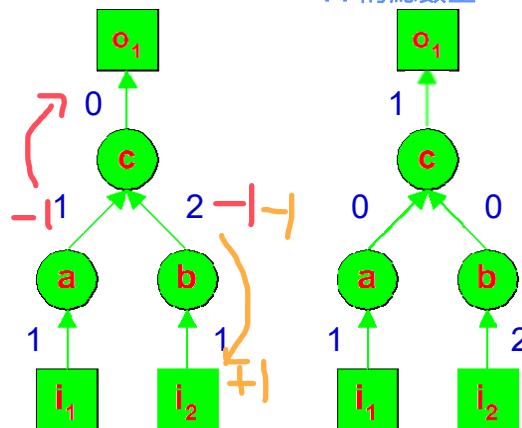
- A peripheral retiming is a retiming such that
 - $r(v)=0$ where v is an I/O pin
 - $w(u,v)+r(v)-r(u)=0$ where $e(u,v)$ is an internal edge
- Move all registers to the peripheral edges
- Leave a purely combinational logic block between two set of registers
- No two paths between any input i and any output j have different edge weights 看下一頁例子
- Exist α_i and β_j , $1 \leq i \leq m$, $1 \leq j \leq n$ such that $W_{i,j} = \alpha_i + \beta_j$
 $W_{i,j} = \sum_{\text{path } i_j \rightarrow o_j} w(e)$ if all paths between input i and output j have the same weight
- Complexity $O(e \cdot \min(m,n))$

Examples of Peripheral Retiming

- Example 1:

$$W_{1,1}=2, W_{2,1}=3, \\ \Rightarrow \alpha_1=1, \alpha_2=2, \beta_1=1$$

把紅色往output端移動 ·
再將橘色往input端移動

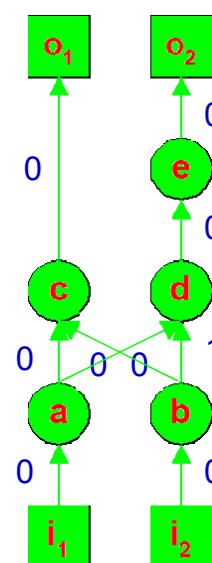


再從每條由input通往
output的path去檢查
FF的總數量

可以看到
 $i_1 \rightarrow o_1: 1+1=2$
 $i_2 \rightarrow o_1: 2+1=3$
皆與原本相同-->correct

- Example 2:

$$W_{1,1}=0, W_{1,2}=0, W_{2,1}=0, W_{2,2}=1 \\ \Rightarrow \text{no solution}$$



這個case如果將
FF移動後 · 發現
中間會出現無法消
除的-1 · 因此這個
電路就不能用這種
方法優化

NCTU M SEDA Lab.

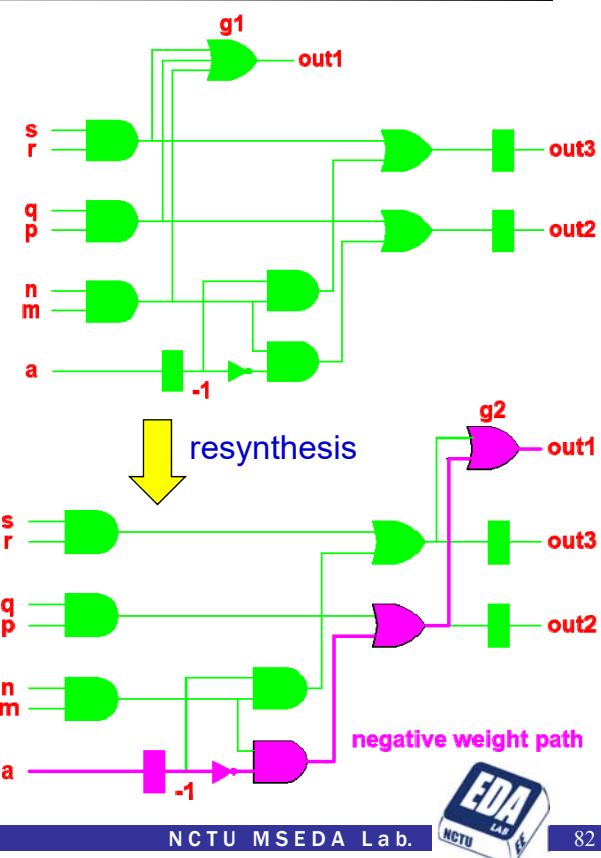
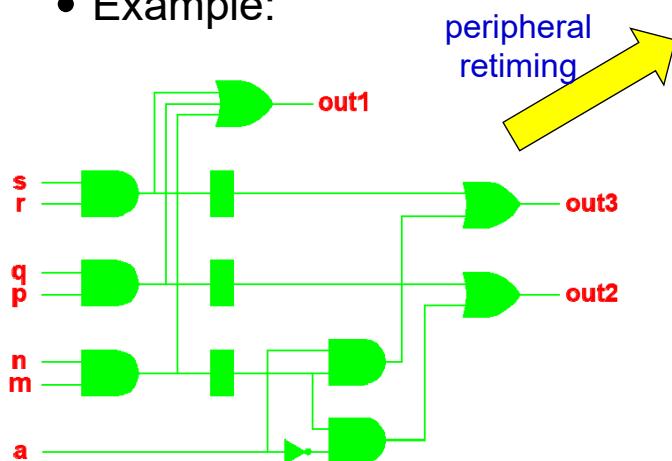


81

Legal Resynthesis Operation

跟上一頁右邊的例子有點像 ·
將FF移動後優化comb ckt · 再
將FF移回來發現粉紅色的path
上有一個-1 --> illegal

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



NCTU M SEDA Lab.



82

Challenges of Retiming Techniques

- Retiming complexity
 - Computing complexity is often too high
- Performance improvements due to retiming
 - Design performance of the physical netlist is not guaranteed due to the lack of placement info
 - To forecast whether the design performance could be improved becomes extremely important
- Side effects on verification
 - Sequential verification becomes difficult retiming可能會導致register
數量改變，讓sequential
verification變得較難驗證
 - Engineering Change Order (ECO)



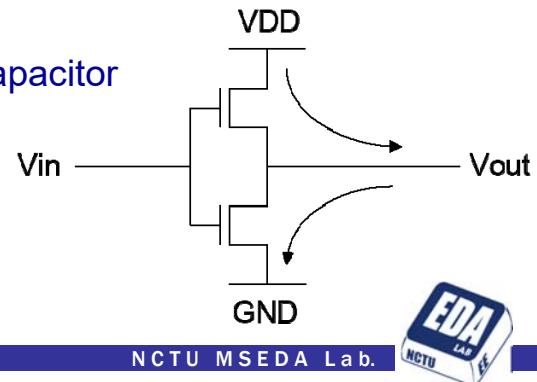
Outline

- Synthesis overview
- RTL synthesis
- Two-level logic optimization
- Multi-level logic optimization
- Technology mapping
- Timing analysis
- Timing optimization
- Synthesis for low power



Power Dissipation

- Leakage power
 - Static dissipation due to leakage current
 - Typically a smaller value compared to other power dissipation
 - Getting larger and larger in deep-submicron process
- Short-circuit power
 - Due to the short-circuit current when both PMOS and NMOS are open during transition
 - Typically a smaller value compared to dynamic power
- Dynamic power
 - Charge and discharge of a load capacitor
 - Usually the major part of total power consumption



Power Dissipation Model

$$P = \frac{1}{2} * C * V_{dd}^2 * D$$

- Typically, **dynamic power** is used to represent total power dissipation

P: the power dissipation for a gate
C: the load capacitance
 V_{dd} : the supply voltage
D: the **transition density** 我們做EDA的能改的就是去盡量減少transition的次數
- To obtain the power dissipation of the circuit, we need
 - The **node capacitance** of each node (obtained from layout)
 - The **transition density** of each node (obtained by computation)



The Signal Probability

- Definition: The signal probability of a signal $x(t)$, denoted by P_x^1 is defined as :

就是 $p(t)$ · 一個gate輸出結果為1的機率
Ex: AND的 $p(t) = 0.25$

$$P_x^1 \equiv \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t) dt$$

where T is a variable about time.

- P_x^0 is defined as the probability of a logic signal $X(t)$ being equal to 0.
- $P_x^0 = 1 - P_x^1$

NCTU M SEDA Lab.



87

Transition Density

- Definition: The transition density D_x of a logic signal $x(t)$, $t \in (-\infty, \infty)$, is defined as

$$D_x \equiv \lim_{T \rightarrow \infty} \frac{n_x(T)}{T \cdot f_c}$$

計算transition數量的時候除以clock freq讓結果比較公平

where f_c is the clock rate or frequency of operation.

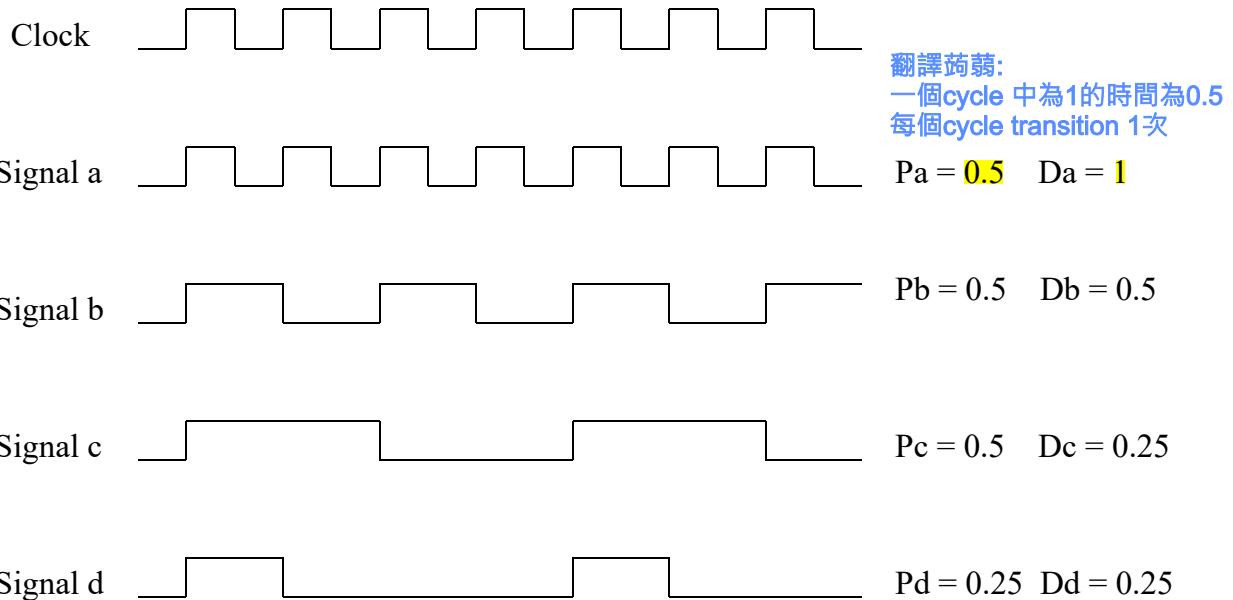
- D_x is the expected number of transitions happened in a clock period.
- A circuit with clock rate 20MHz and 5 MHz transitions per second in a node, transition density of this node is $5M / 20M = 0.4$

NCTU M SEDA Lab.



88

Signal Probability and Transition Density



NCTU M S E D A Lab.



89

The Calculation of Signal Probability

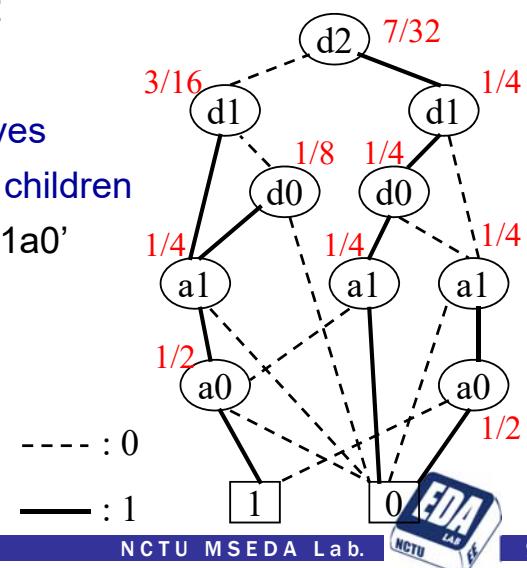
- BDD-based approach is one of the popular way
- Definition
 - $p(F)$: fraction of variable assignments for which $F = 1$
- Recursive Formulation
 - $p(F) = [p(F[x=1]) + p(F[x=0])] / 2$
- Computation
 - Compute bottom-up, starting at leaves
 - At each node, average the value of children
- Ex: $F = d2'(d1+d0)a1a0 + d2(d1'+d0')a1a0' + d2d1d0a1'a0$
 $p(F) = 7/32 = 0.21875$

以最後的output $d2$ 為例子

$d2$ 有 $1/2$ 的機率走到左邊的 $d1 \rightarrow 0.5 * 3/16 = 3/32$

$d2$ 有 $1/2$ 的機率走到右邊的 $d1 \rightarrow 0.5 * 1/4 = 1/8$

$3/32 + 1/8 = 7/32$ · 因此可以用bottom-up的方式求得答案



NCTU M S E D A Lab.

90

The Calculation of Transition Density

- Transition density of cube

- $f = ab$ Da為a改變的機率，但是a如果要影響output，則b一定要是1，因此需要乘上b為1的機率，也就是 P_b
- $D_f = D_a P_b + D_b P_a - 1/2 D_a D_b$ 同理，Db亦要乘上 P_a ，最後再扣掉a、b同時改變的狀況
- $D_a P_b$ means that output will change when $b=1$ and a has changes
- $1/2 D_a D_b$ is the duplicate part when both a and b changes

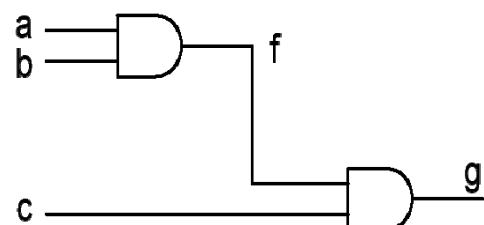
- n-input AND :

- a network of 2 -input AND gate in zero delay model
- 3-input AND gate

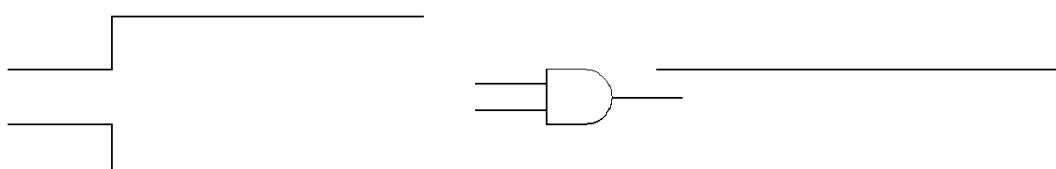
$$D_g = D_f P_c + D_c P_f - 1/2 D_f D_c$$

- Inaccuracy of this simple model :

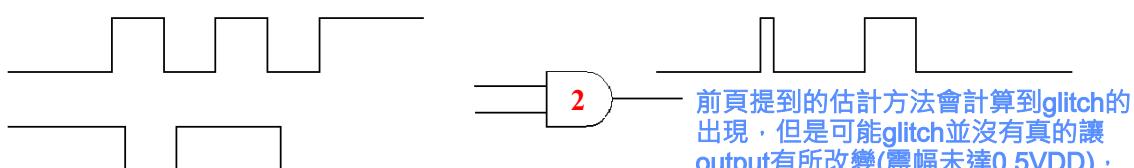
- Temporal relations
- Spatial relations



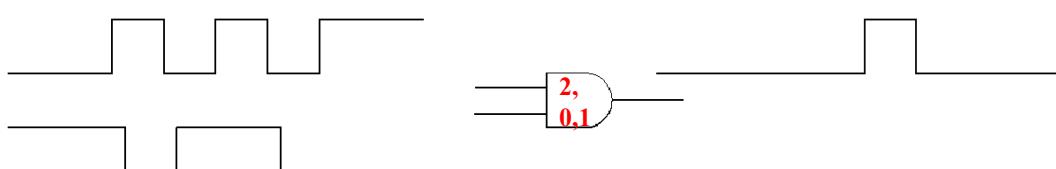
The Problem of Temporal Relations



(1) Without considering the Gate Delay and Inertial Delay



(2) Without considering Inertial Delay



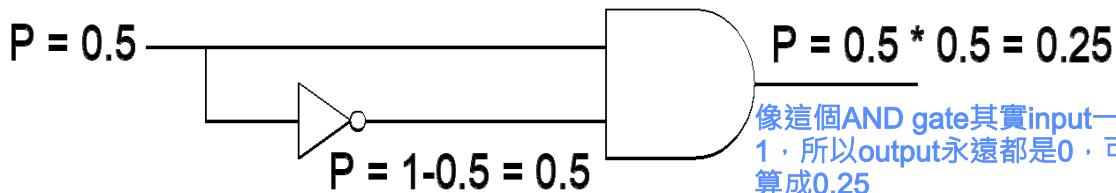
(3) Practical condition

前頁提到的估計方法會計算到glitch的出現，但是可能glitch並沒有真的讓output有所改變(震幅未達0.5VDD)，因此結果會不準

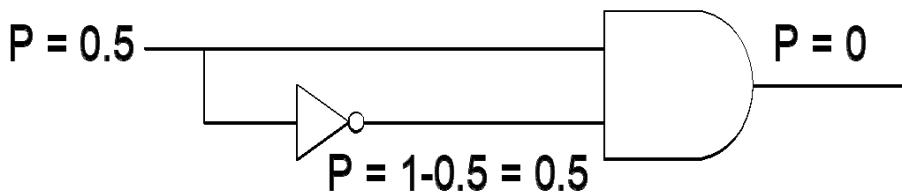


The Problem of Spatial Correlation

如果沒有考慮到兩個input之間的交互關係，
也有可能導致估計錯誤



(a) Without considering Spatial Correlation

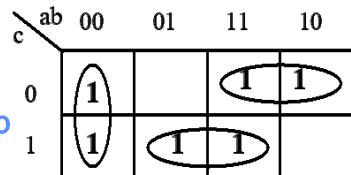
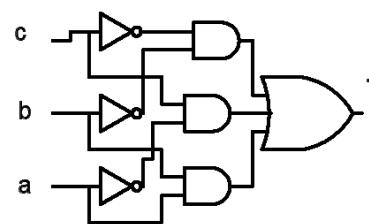
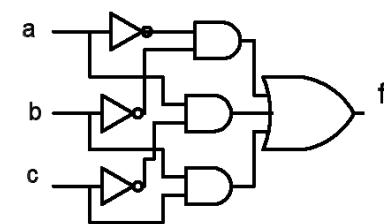


(b) Practical condition



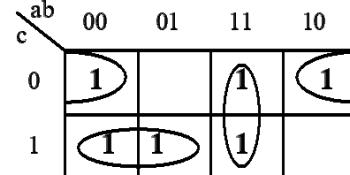
Logic Minimization for Low Power (1/2)

- Consider an example:



$$f = a'b' + ac' + bc$$
$$P = 108.7 \mu W$$

(a)



$$f = b'c' + a'c + ab$$
$$P = 115.5 \mu W$$

(b)

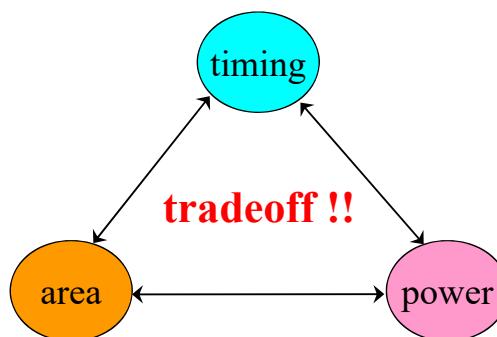
可以將power加入k-map
化簡的cost function
(不同的implicant合成出
的電路power也會不同)

- Different choices of the covers may result in different power consumption



Logic Minimization for Low Power (2/2)

- Typically, the objective of logic minimization is to minimize
 - NPT : the number of product terms of the cover
 - NLI : the number of literals in the input parts of the cover
 - NLO : the number of literals in the output parts of the cover
- For low power synthesis, the power dissipation has to be added into the cost function for best covers

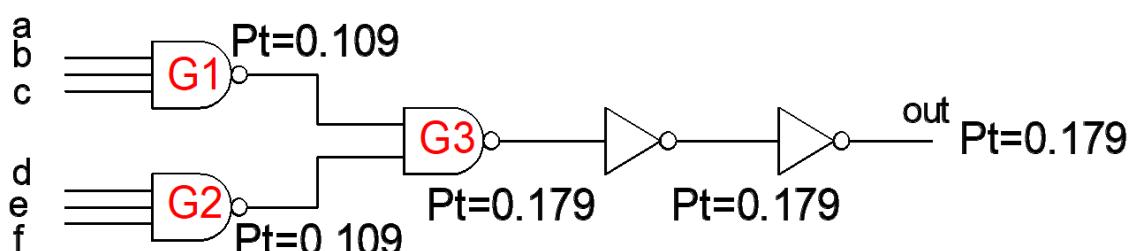


NCTU M SEDA Lab.



95

Technology Mapping for Low Power (1/3)



(a) Circuit to be mapped

有了output loading，我們就可以套用 dynamic power的公式去估計power

Gate Type	Area	Intrinsic Cap. output loading	Input Load
INV	928	0.1029	0.0514
NAND2	1392	0.1421	0.0747
NAND3	1856	0.1768	0.0868
AOI33	3248	0.3526	0.1063

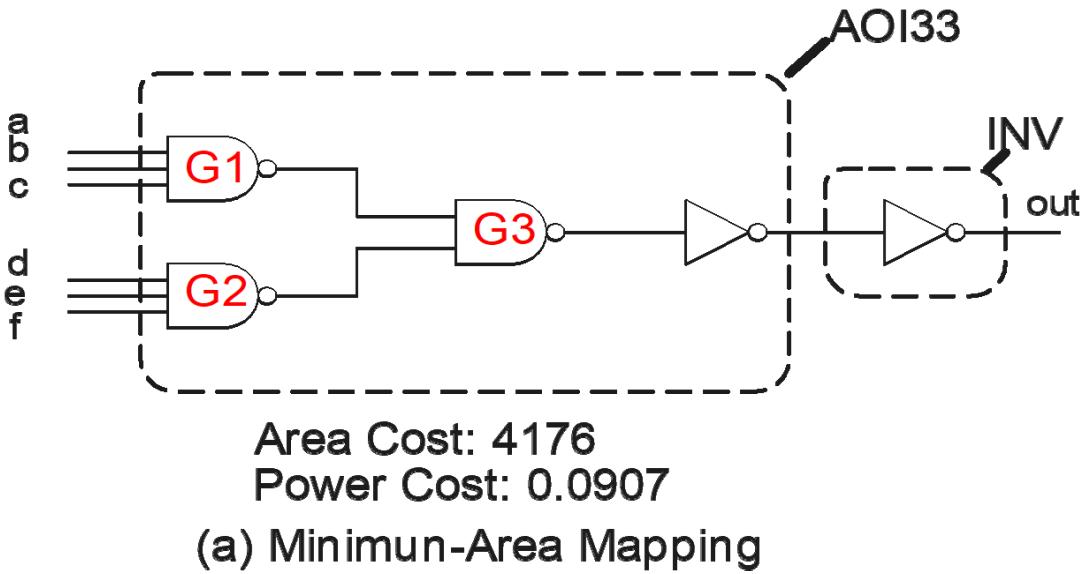
(b) Characteristics of Library

NCTU M SEDA Lab.



96

Technology Mapping for Low Power (2/3)



Technology Mapping for Low Power (3/3)

