Computer Architecture and System Laboratory

Lab 6
Verilog – Sequential Design(1)

Department of Electrical Engineering
National Cheng Kung University

Outline

1. Combinational logic V.S. Sequential logic
2. Latch 介紹
3. 實作題(一) D Latch
4. Level trigger V.S. Edge trigger
5. Flip-Flop 介紹
6. 實作題(二) D Flip-Flop using Edge Detector
7. 實作題(三) D Flip-Flop using Master Slave D Latch
8. 課間檢查與結報內容
9. 參考資料

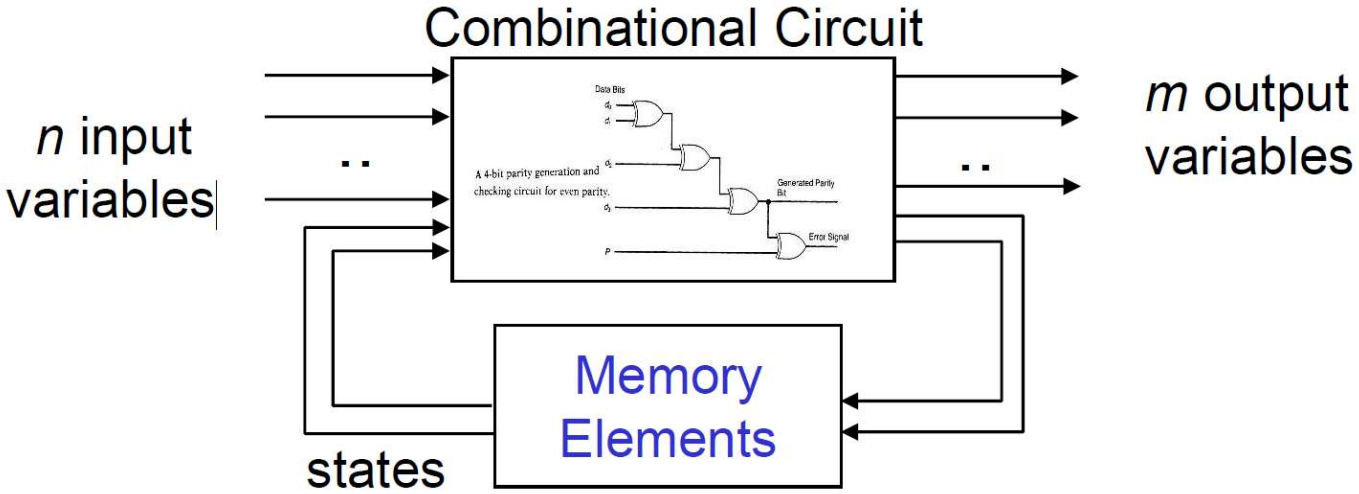
Combinational circuit V.S. Sequential circuit

Combinational Circuit

- 電路中任一時刻的 output，只取決於當時的 input，與該時刻以前的 input 無關。
- 電路中無 memory element
- E.g. Lab3的 MUX / DEMUX 及 Encoder / Decoder

Sequential Circuit

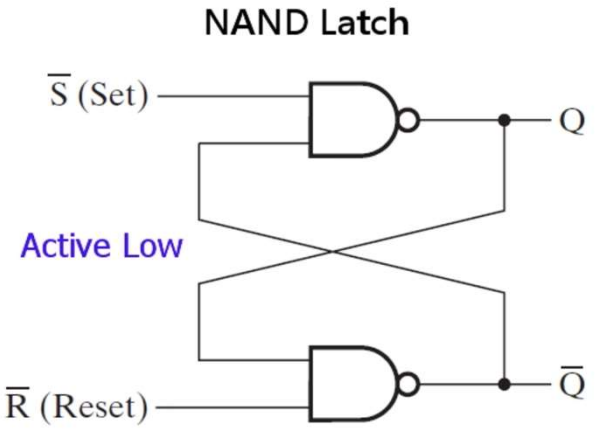
- 電路中任一時刻的 output 和當時的 input 及前一個時刻 input 產生的 output 有關.
- 電路中有 memory element
- E.g. 本次實驗中的Latch及Flip-Flop



Latch

SR Latch(NAND)

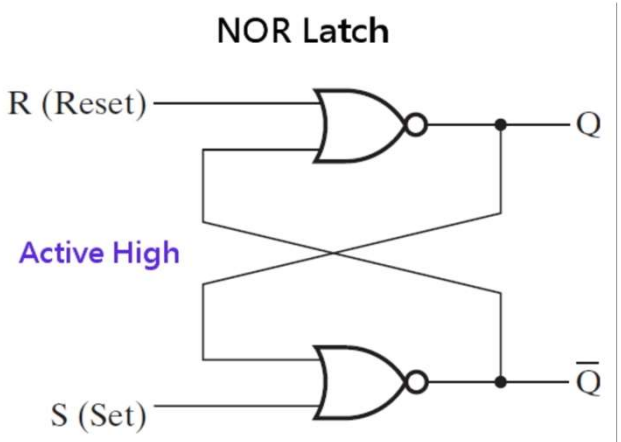
- 使用兩個 NAND 閘來建構, 稱為 NAND Latch
- 此電路為 active low (訊號為 low 時啟動), 因此兩個輸入 S 與 R 上面有個 "bar" 的符號
- 需注意兩個輸入不可同時為 0



Inputs		Outputs		Remarks
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
0	0	1	1	Not allowed
0	1	1	0	Latch SET
1	0	0	1	Latch RESET
1	1	NC	NC	No change

SR Latch(NOR)

- 使用兩個 NOR 閘來建構, 稱為 NOR Latch
- 此電路為 active high （訊號為high時啟動），因此兩個輸入 S 與 R 上面沒有 ”bar” 的符號
- 需注意兩個輸入不可同時為 1

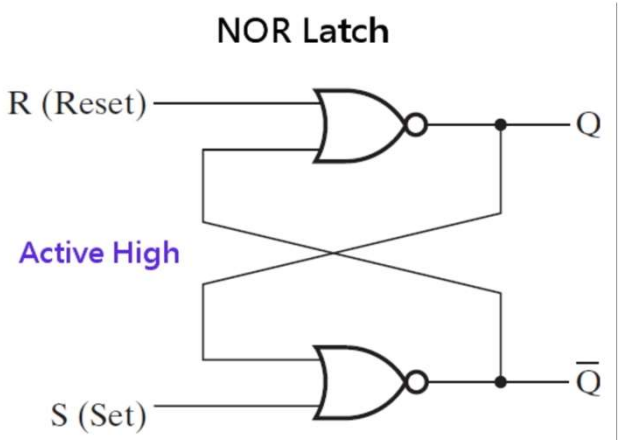


Inputs		Outputs		Remarks
S	*R*	*Q*	$\overline{Q}$	
0	0	NC	NC	No change
0	1	0	1	Latch RESET
1	0	1	0	Latch SET
1	1	0	0	Not allowed

SR Latch – Not allowed input

以 NOR Latch 為例，假設兩個 input 皆為 1 時

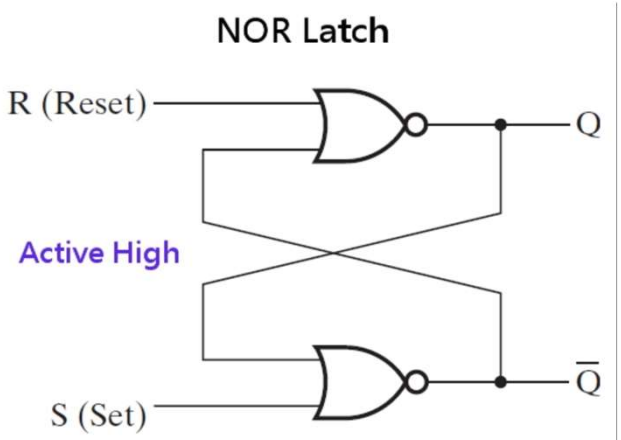
- output Q 及 Q_bar 皆為 0，Q 與 Q_bar 將失去反相對比的功能
- 在離開此狀態時，將造成 Racing condition



Inputs		Outputs		Remarks
S	*R*	*Q*	$\overline{Q}$	
0	0	NC	NC	No change
0	1	0	1	Latch RESET
1	0	1	0	Latch SET
1	1	0	0	Not allowed

SR Latch – Racing condition

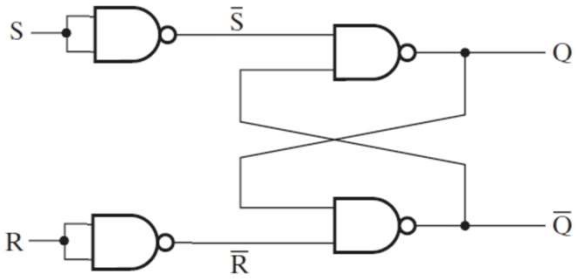
假設目前已進入 Not allowed 狀態，而下一個 input 為 S = 0、R = 0，output 應該為 Q = 0、Q_bar = 0，但在實際電路中，反應較快的 NOR 閘 output 會先變1，進而使另一個 NOR 閘的 output 變為0，造成 output不可預測。因為兩個 NOR 閘就像在競賽一樣，因此稱為Racing condition



Inputs		Outputs		Remarks
S	*R*	*Q*	$\overline{Q}$	
0	0	NC	NC	No change
0	1	0	1	Latch RESET
1	0	1	0	Latch SET
1	1	0	0	Not allowed

Avoid Racing condition – Gated SR Latch

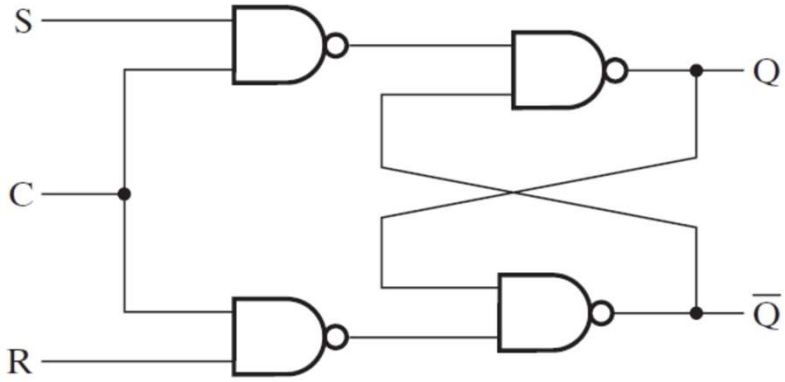
- 目的：讓 input 只能在特定時刻輸入至 Latch 中，而非隨時皆可改變 output
- 步驟1：在 NAND Latch 的 input 加入 NAND 閘，其行為將變為 Active High

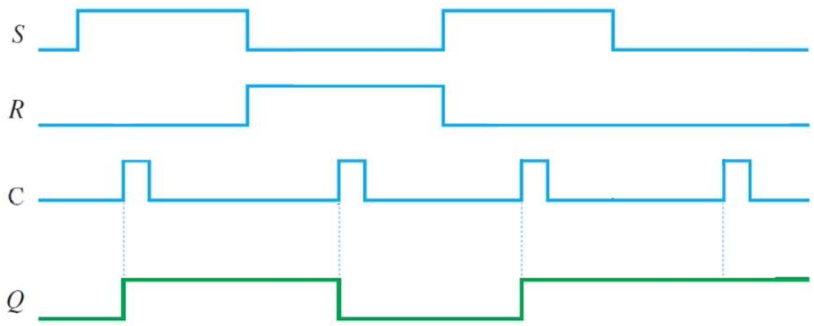


Inputs		Outputs		Remarks
S	*R*	*Q*	$\overline{Q}$	
0	0	NC	NC	No change
0	1	0	1	Latch RESET
1	0	1	0	Latch SET
1	1	0	0	Not allowed

Avoid Racing condition – Gated SR Latch (Cont'd)

- 步驟二：加入 control 訊號C，讓 input 只在 C = 1 時才可進入 Latch
- C=1、S=1、R=1仍為Not allowed

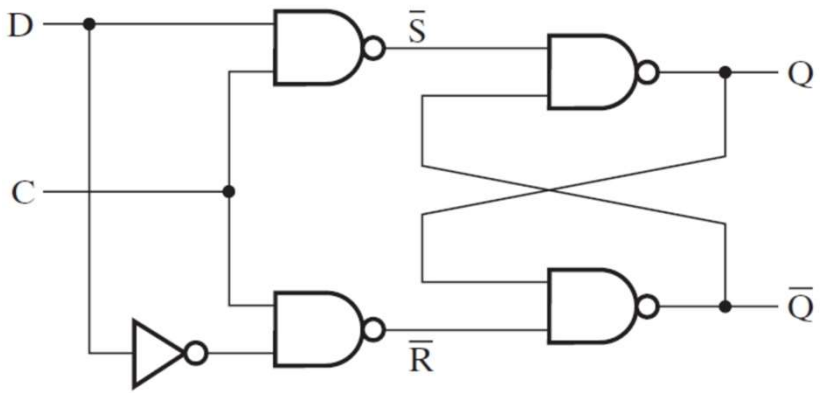


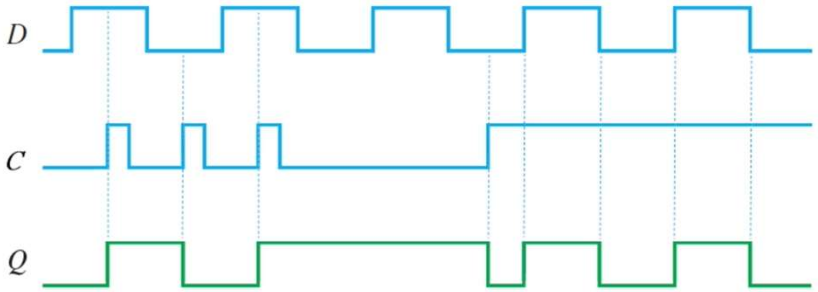


Inputs			Outputs		Remarks
C	*S*	*R*	*Q*	$\overline{Q}$	
0	X	X	NC	NC	No change
1	0	0	NC	NC	No change
1	0	1	0	1	Latch RESET
1	1	0	1	0	Latch SET
1	1	1	0	0	Not allowed

Avoid Racing condition – Gated D Latch

- 使用 D 及 D_bar 取代 S 及 R：讓進入第一級 NAND 閘的輸入永遠不會同時為 1
- D Latch 可徹底解決 Racing condition





Inputs		Outputs		Remarks
C	*D*	*Q*	$\overline{Q}$	
0	X	NC	NC	No change
1	0	0	1	Latch RESET
1	1	1	0	Latch SET

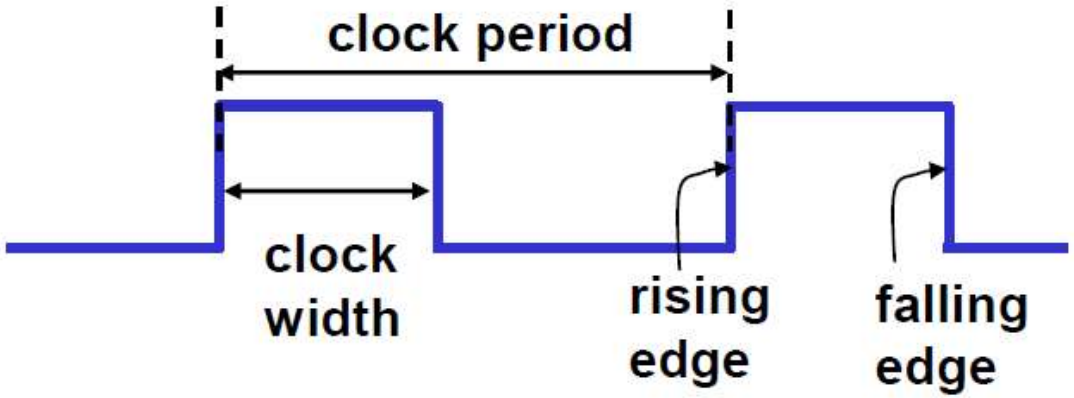
實作題(一) Gated D Latch

- 在實作題(一)裡，你們需要完成
  - Gated D Latch，需要使用 Gate Level 的方式去撰寫。
  - 在 prob1 資料夾中的 prob1.v 裡面，使用 Gate Level的方式去寫自己的 D Latch 模組，並使用 tb_prob1.v 模擬。
  - 查看波形圖確認行為是否正確

Level trigger
V.S.
Edge trigger

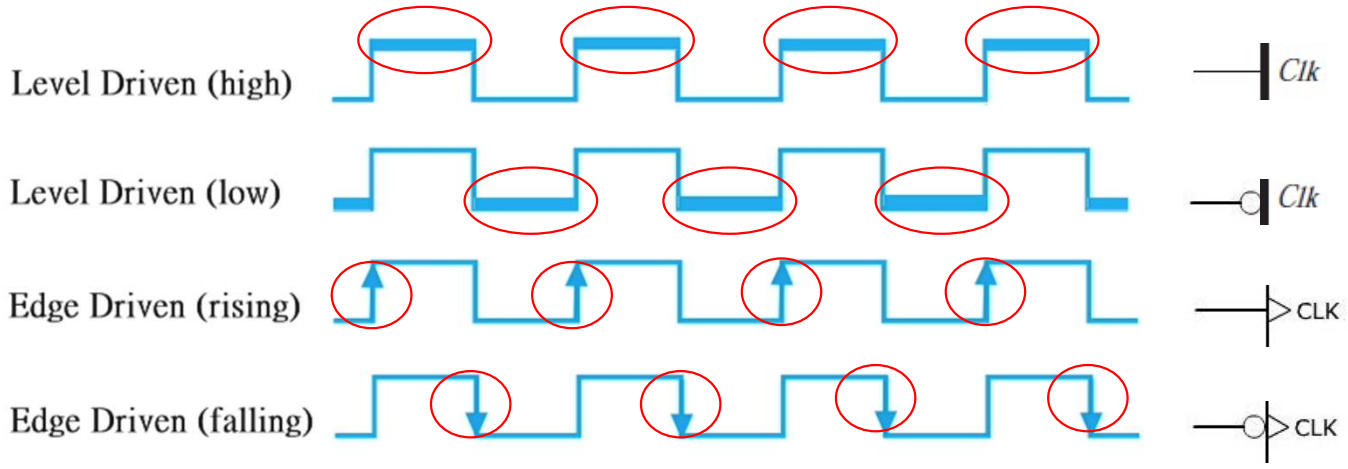
Clock

- Clock period: 一個 cycle 的時間(second/cycle)
- Clock frequency: clock period 的倒數. (cycle/second)
- Clock width: clock訊號為1的時間長度.
- Duty cycle: clock width/clock period，即一個 clock period 內 clock 訊號為1的比例
- Rising edge：clock由0變1的時刻
- Falling edge：clock由1變0的時刻



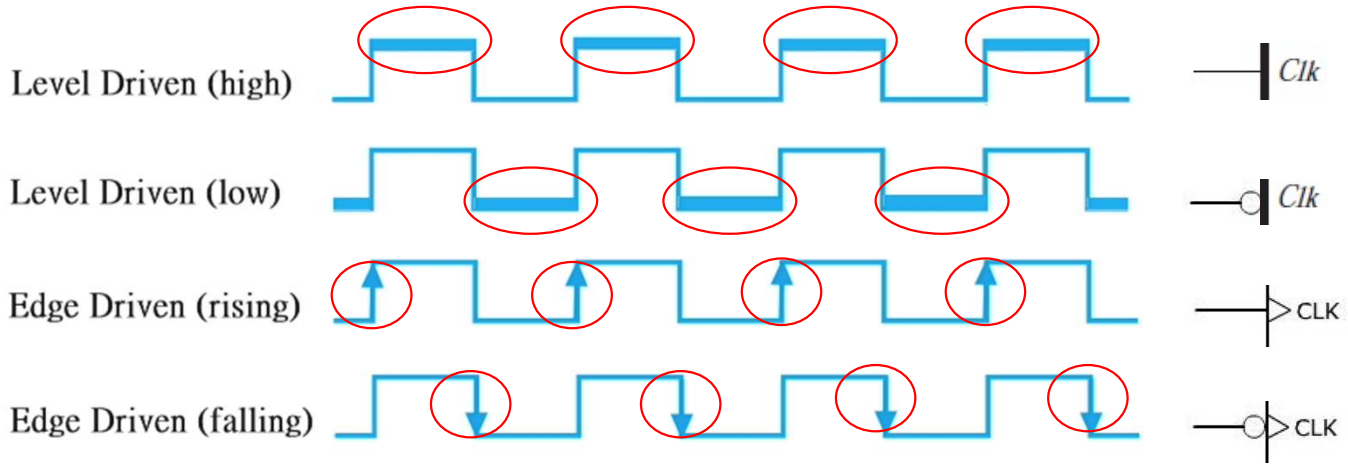
Level trigger V.S. Edge trigger

- Level trigger：input 可在 clock 為 0 或者 1 時改變 output
- Edge trigger：input 只可在 Rising Edge 或者 Falling Edge 時改變output



Level trigger V.S. Edge trigger

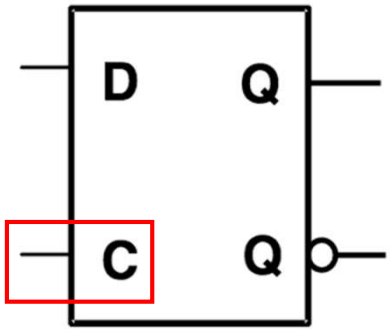
- Level trigger：如 Latch，input 可改變 output 的時間較長，在此期間電路雜訊可能會影響 output
- Edge trigger：如 Flip-Flop，input 只可在 Rising / Falling Edge 改變 output，可避免雜訊影響 output



Flip-Flop

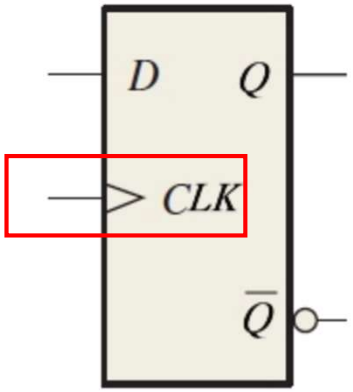
D Latch V.S. D Flip-Flop (Rising edge)

D Latch



Inputs		Outputs		Remarks
C	*D*	*Q*	$\overline{Q}$	
0	X	NC	NC	No change
1	0	0	1	Latch RESET
1	1	1	0	Latch SET

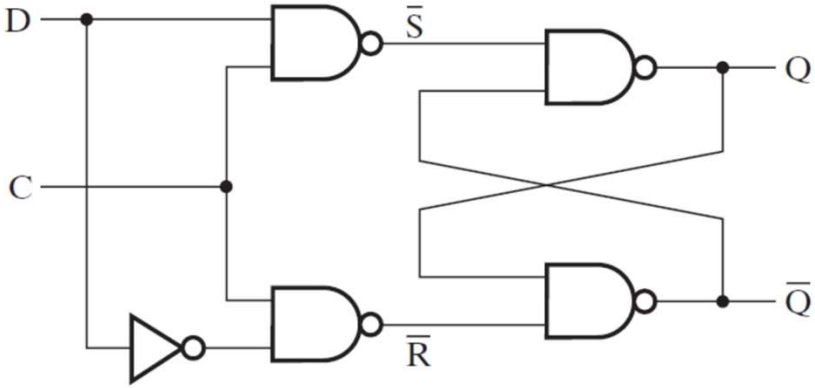
D Flip-Flop



Inputs		Outputs		Remarks
CLK	*D*	*Q*	$\overline{Q}$	
0 1 ↓	X	NC	NC	No change
↑	0	0	1	RESET
↑	1	1	0	SET

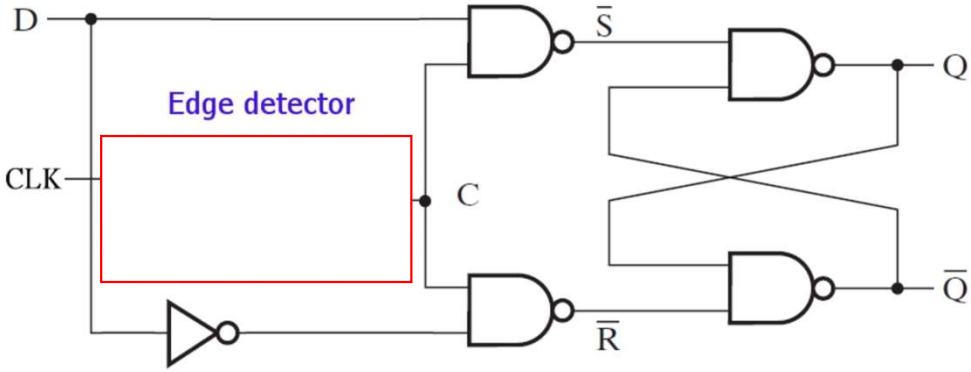
D Flip-Flop implementation – using Edge Detector

D Latch



Inputs		Outputs		Remarks
C	*D*	*Q*	$\overline{Q}$	
0	X	NC	NC	No change
1	0	0	1	Latch RESET
1	1	1	0	Latch SET

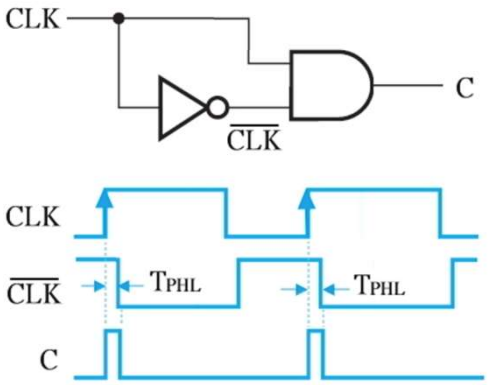
D Flip-Flop



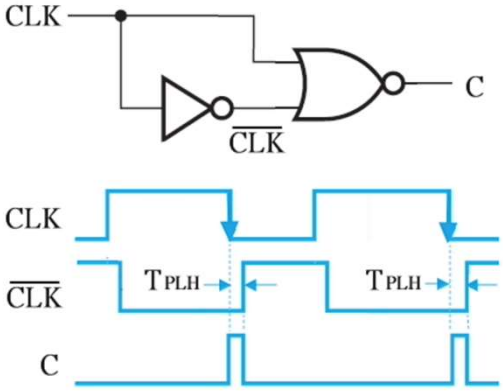
Inputs		Outputs		Remarks
CLK	*D*	*Q*	$\overline{Q}$	
0 1 ↓	X	NC	NC	No change
↑	0	0	1	RESET
↑	1	1	0	SET

Edge Detector

- 以 Rising Edge Detector 為例，當 CLK 由 0 變為 1 時，CLK_bar 需要經過一個 NOT 閘，而 CLK 則不需要，因此 CLK_bar 會較晚從 1 變為 0，這段 delay 的時間記為 T_{PHL} (Propagation High to Low)，在 T_{PHL} 這段時間 CLK_bar 與 CLK 皆為 1，因此 AND 閘的 output C 會為短暫為 1。利用 C 作為 D Latch 的 control 即可在 Rising Edge 時將input D 送入 Latch 中，達到 Falling Edge trigger 的效果。



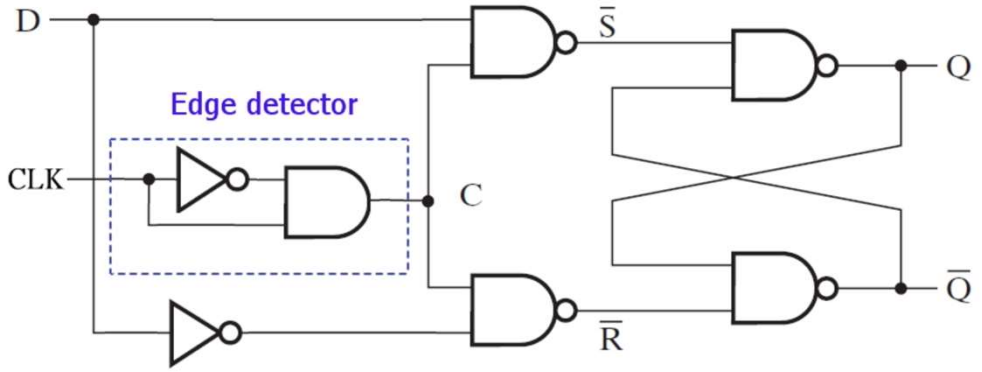
Rising edge detector

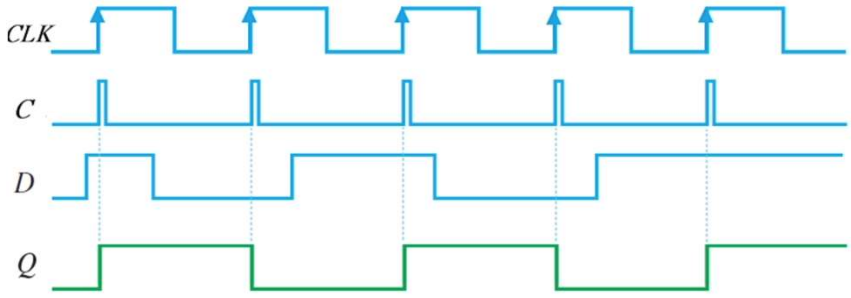


Falling edge detector

D Flip-Flop implementation – using Edge Detector

- 將 CLK 訊號經過 Edge detector 後和 D Latch 的 control 連接即為 D Flip-Flop



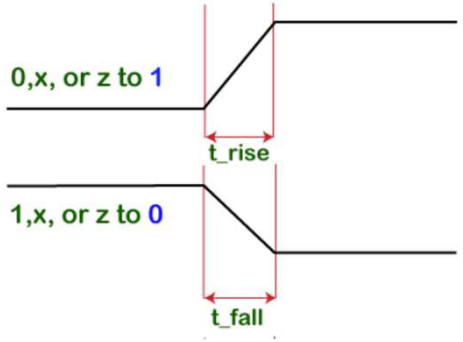


Inputs		Outputs		Remarks
CLK	*D*	*Q*	$\overline{Q}$	
0 1 ↓	X	NC	NC	No change
↑	0	0	1	RESET
↑	1	1	0	SET

Verilog 補充

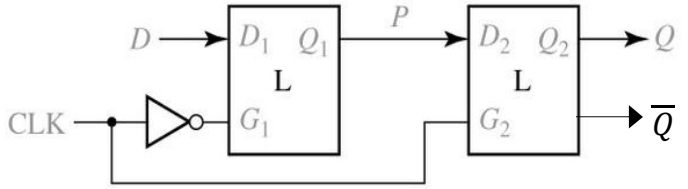
- Rise delay：邏輯閘 output 由 low / x / z 轉 high 的延遲時間
- Fall delay：邏輯閘 output 由 high / x / z 轉 low 的延遲時間

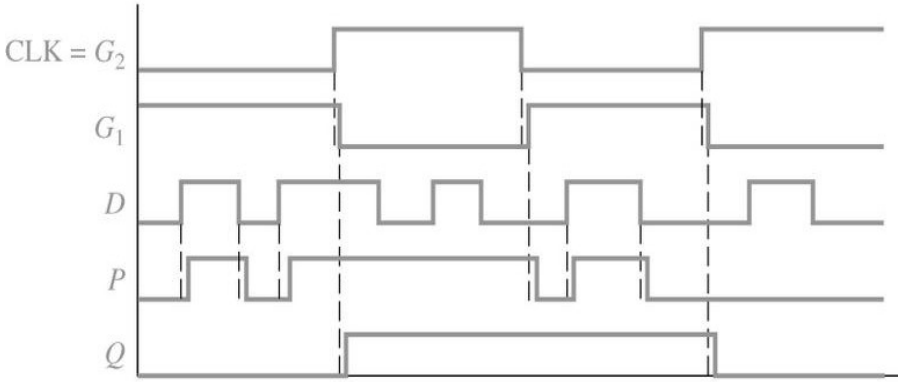
```
// Two delays specified - used for Rise and Fall transitions
or #(<rise>, <fall>) o1 (out, a, b);
```



D Flip-Flop implementation – using Master Slave D Latch

- 利用兩個 D Latch 分別接上 CLK 及 CLK_bar，第一個 Latch 在 CLK 輸入為0 時，將資料存入 Q_1，當 CLK 輸入轉為 1 時 (Rising Edge)，Q_1 會進入第二個 Latch 並改變 output Q。
- 左側的 D Latch 負責在 Rising Edge 將資料傳入右側的 D Latch，因此稱為 Master，而右側的 D Latch負責接收資料並輸出，稱為 Slave



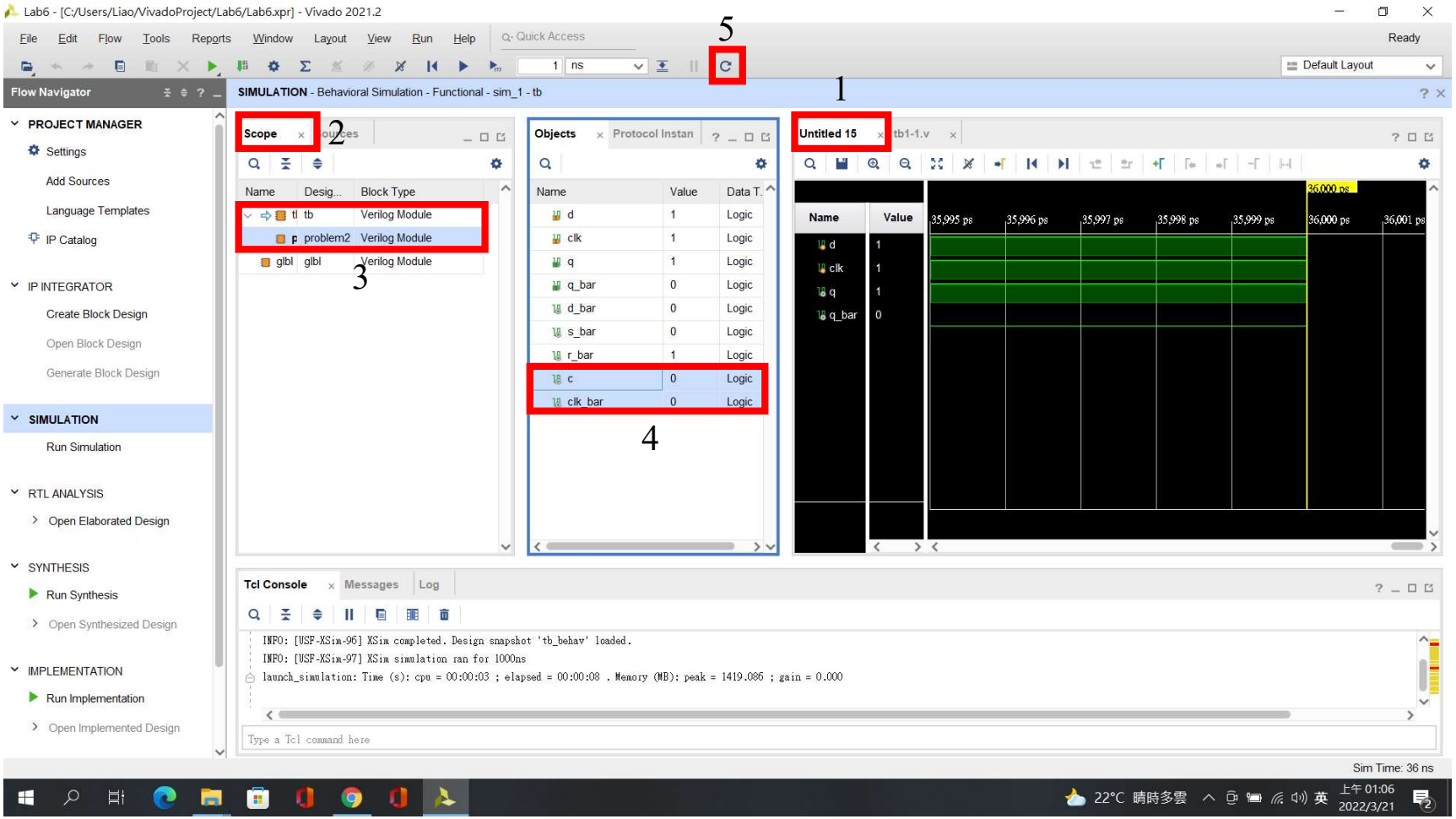


Inputs		Outputs		Remarks
CLK	*D*	*Q*	$\overline{Q}$	
0 1 ↓	X	NC	NC	No change
↑	0	0	1	RESET
↑	1	1	0	SET

實作題(二) D Flip-Flop – using Edge Detector

- 在實作題(二)裡，你們需要完成
  - 使用 Edge Detector 的方式實作 Rising Edge trigger 的 D Flip-Flop，需要使用 Gate Level 的方式去撰寫。
  - Not gate 的其中一個 delay 設定為 1，另一個設定為 0，請思考是 rise 還是 fall 需要 delay。
  - 在 prob2 資料夾中的 prob2.v 裡面，使用 Gate Level 的方式去寫自己的 D Flip-Flop 模組，並使用 tb_prob2.v 模擬。
  - 查看波形圖確認行為是否正確，請注意查看 edge detector 的 input 及 output 波形變化

Vivado Simulation補充



Vivado simulation補充

要查看 module 底下特定 wire 的波形圖時，在開啟 simulation 後可以依照上一頁的步驟

1. 開啟波形圖
2. 點擊 scope
3. 將 tb 展開，找到要查看的 module
4. 將需要的 wire 拖移至波形圖左側
5. 點擊上方 relaunch

實作題(三) D Flip-Flop – using Master Slave D Latch

- 在實作題(三)裡，你們需要完成
  - 使用Master Slave D Latch 實作 Rising Edge trigger 的 D Flip-Flop，需要使用 Gate Level 的方式去撰寫。
  - 在 prob3 資料夾中的 prob3.v 裡面，使用 Gate Level 的方式去寫自己的 D Flip-Flop 模組，並使用 tb_prob3.v 模擬。
  - 請使用實作題一做出來的 Latch 程式碼，並用 module instantiate 的方式實作
  - 查看波形圖確認行為是否正確

Verilog 補充

- 上次Lab時有發現幾個同學常見錯誤

1. 在 always block 內 instantiate module ：always 和 module instantiate 及 assign 是不同層級的描述方式，不可以在一個層級底下使用另一層級的描述方式，詳細資料可參考以下網址
   https://hackmd.io/@dppa1008/BJWS5_B_G?type=view#Dataflow-Description--Dataflow-Modeling-

Verilog 補充

2. 在連接兩個module時，沒有宣告wire，直接將module A的output port 接到 module B 的 input port ：這是錯誤的連線方式，compiler 並不會報錯，但是在之後的合成會有問題。正確方式應該是使用一條 wire 分別連接至 module A 的output port 和 module B 的 input port

下圖程式碼一部份省略，僅保留重點部分

```
1  module example(...)
2      wire x;
3      module A(...,.outputOfA(x));
4      module B(.inputOfB(x),...);
5  endmodule
```

課間檢查與結報內容

課間檢查與結報內容

- 課間檢查（此次 Testbench 皆無自動驗證功能，請同學自行確認波形無誤）
  - 實作（一）：波形圖與程式碼
  - 實作（二）：波形圖（包含 Edge Detector 的 input/output）與程式碼
  - 實作（三）：波形圖與程式碼（需使用 module instantiate 方式重複使用實作一的程式碼）
- 結報內容
  - 三個實作分別對波形進行說明
  - 思考如何使用兩個 D Latch 實作 Falling Edge trigger 的 D Flip-Flop並畫出電路圖（如投影片24頁的圖）及解釋其原理
  - 實驗心得

參考資料

- http://yhhuang1966.blogspot.com/2019/06/latch-flip-flop.html
- https://hackmd.io/@dppa1008/BJWS5_B_G?type=view#Dataflow-Description--Dataflow-Modeling-