今天的實驗為實作 Latch 及 Flip-Flop

第一個實驗為實作 D Latch

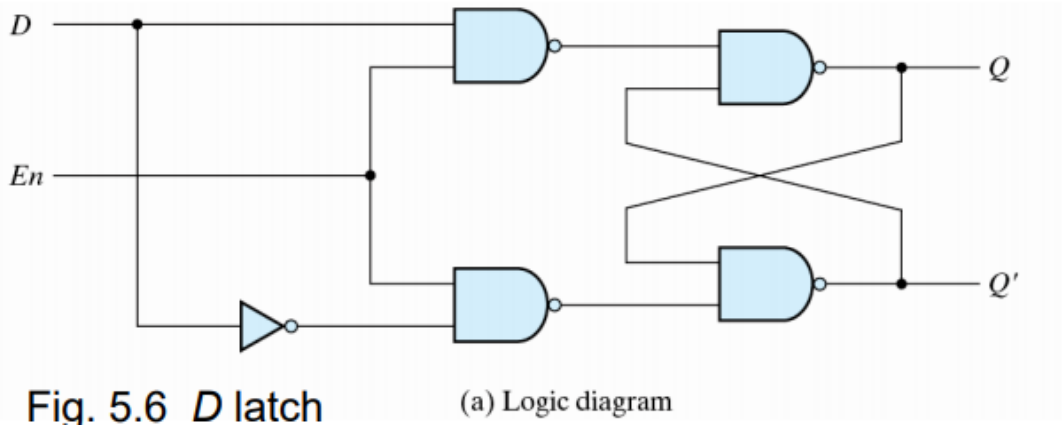


Fig. 5.6 *D* latch (a) Logic diagram

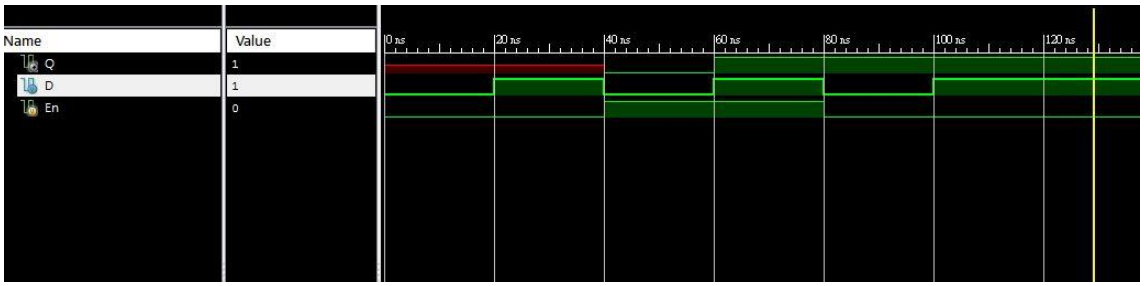
```
20  ////////////////////////////////////////////////
21  module mod1(output Q , input D , input En);
22      wire not_D;
23      wire For_Q;
24      wire For_notQ;
25      wire not_Q;
26      not(not_D,D);
27      nand(For_Q,D,En);
28      nand(Q,For_Q,not_Q);
29      nand(For_notQ,not_D,En);
30      nand(not_Q,For_notQ,Q);
31  endmodule
32
```

為一個很簡單的實作

比較會擔心的是:

nand 何時會讀取回朔訊號(Q’ (t)產出 Q(t+1)或 Q(t)產出 Q’ (t+1))

Ans:考慮訊號改變的前一刻 Q(t)、Q’ (t)狀況

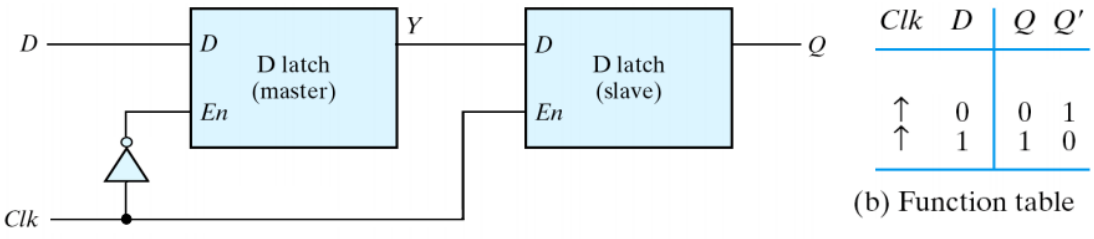


可以看到當 enable 為 1 時 Q 會跟 D 相等 enable 為 0 時保持不變

在 40ps 之前，因為 enable 一直沒有開啟，所以也無法成功 assign 值，Q 為 unknown

第二個實驗為 Master-Slave D Flip-Flop

此 Flip-Flop 為正源觸發(postedge)



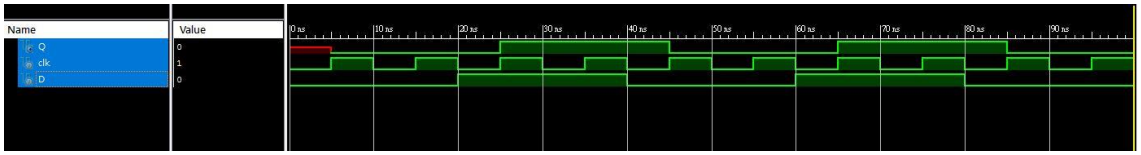
Clk	D	Q	Q′
↑	0	0	1
↑	1	1	0

(b) Function table

主要就直接拿兩個實驗一的 D Latch 接在一起

要注意的是 clk 必須兩個相反，不然就等同於原本的 D Latch

```
21  module mod1(output Q , input D , input En);
22     wire not_D;
23     wire For_Q;
24     wire For_notQ;
25     wire not_Q;
26     not(not_D,D);
27     nand(For_Q,D,En);
28     nand(Q,For_Q,not_Q);
29     nand(For_notQ,not_D,En);
30     nand(not_Q,For_notQ,Q);
31  endmodule
32
33  module mod2(output Q , input clk , input D);
34     wire not_clk;
35     wire Y;
36     not(not_clk,clk);
37     mod1 master(Y,D,not_clk);
38     mod1 slave(Q,Y,clk);
39  endmodule
40
```



可以看到訊號在 20ps~25ps，雖然 D=1，但 clk 不是 1，所以無法啟動 slave

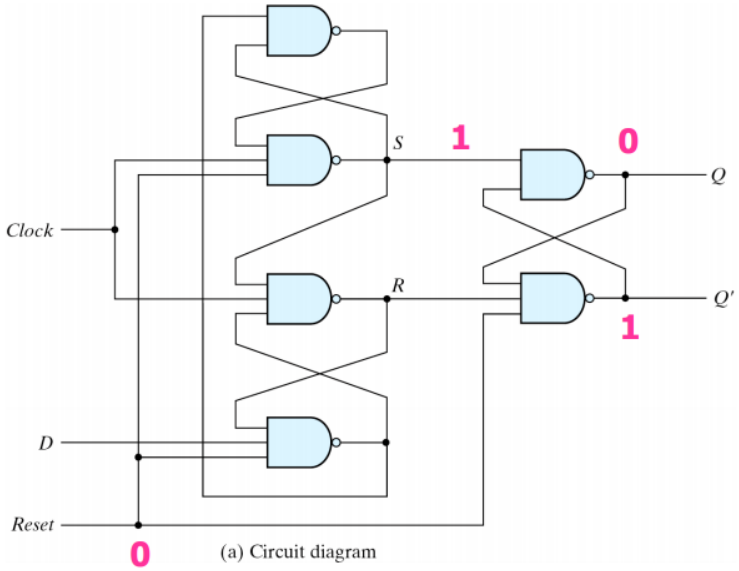
25ps~30ps，clk=1，slave 讀取 master 的 Y output，因此 Q 為 1

40ps~45ps，D=0，但 clk 不是 1，目前改到的只有 master

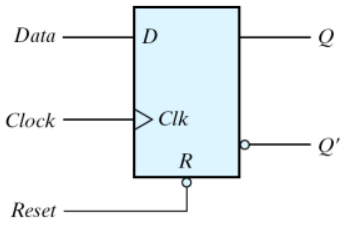
45ps~50ps，clk=1，slave 讀取 master 的 Y output，因此 Q 為 0

第三個實驗為 D flip-flop with asynchronous reset

注意!! Reset 為 0 時才會進行 reset，為 1 時不影響原本 Flip-Flop



(a) Circuit diagram

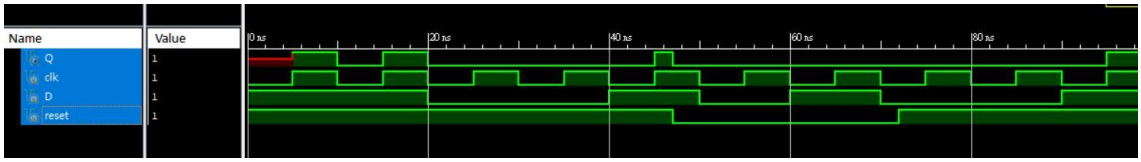


(b) Graphic symbol

R	Clk	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table

```
21  module mod3(output Q , input clk , input D , input reset);
22      wire out_A;
23      wire out_B;
24      wire out_C;
25      wire out_D;
26      wire not_Q;
27
28      nand(out_A,out_D,out_B);
29      nand(out_B,out_A,clk,reset);
30      nand(out_C,out_B,clk,out_D);
31      nand(out_D,out_C,D,reset);
32
33      nand(Q,out_B,not_Q);
34      nand(not_Q,out_B,Q,reset);
35
36  endmodule
37
```



0ps~47ps reset = 1 ，行為如同 D Flip-Flop

47ps reset =0，一下去後直接馬上將 Q reset 成 0

47ps~72ps reset = 0，Q = 0

72ps~100ps reset = 1，行為如同 D Flip-Flop

Note:

```
43    initial #100 $finish;
44    initial begin clk = 0; forever #5 clk = ~clk; end
45    initial fork
46       D = 1;
47       reset = 1;
48       #20 D = 0;
49       #40 D = 1;
50       #50 D = 0;
51       #60 D = 1;
52       #70 D = 0;
53       #90 D = 1;
54        #47 reset = 0;
55        #72 reset = 1;
56    join
57 endmodule
58
```

新的 testbench 寫法

Line43: 表示執行 100ps 後停止模擬

Line44: 表示 clk 一開始為 0，每 5ps 將 clk 反向

Line45~56: fork 用法 ，利用 “# number” 指定在 number 時間時執行後面

指令，最後用 join，完成 fork 完整語法