數位系統設計實習

Lecture 4

CLA/Sequential Logic

指導老師:陳勇志 教授

實習課助教:黃柏皓

Outline

- Verilog 複習 : Behavior-Level
- 4-bit Carry Look-ahead Adder
- Sequential Logic
- Latch and Flip-flop
- LAB 4-1 ~ 4-3

Chapter

- Verilog 複習 : Behavior-Level
- 4-bit Carry Look-ahead Adder
- Sequential Logic
- Latch and Flip-flop
- LAB 4-1 ~ 4-3

Recap Behavior Modeling

- Gate-level Modeling：

  - 每個指令輸出只能1個位元

  *信號必須是**wire**資料型態

  *例如

  ```
  xor g1(sum,A,B);
  and g2(carry,A,B);
  ```

- Dataflow Modeling：

  - 資料流敘述的前後次序不重要。

  *出現在 = 敘述左邊的信號必須是**wire**資料型態

  *例如 assign out = A + B;

- Behavior Modeling：

  - **always**方塊內敘述的前後次序很重要。

  *出現在 = 敘述左邊的信號必須宣告**reg**資料型態

  *例如 always@ (A,B) //always@(*)

  ```
  out = A + B;
  ```

  *可以用行為來描述電路

  *例如 if 、elseif、case、for loop……等

Recap Behavior Modeling

- Initial Statement:

  -從時間 0 開始

  *只執行一次

  *通常用於初始化、監控、波形

  *多個 Initial block會並發執行

  *必須將多個行為語句分組，通常使用關鍵字 **begin** 和 **end**

- Always Statement：

  -從時間 0 開始

  *以循環方式連續執行always區塊中的語句

  *用於對數位電路中連續重複的活動區塊進行建模

  *必須將多個行為語句分組，通常使用關鍵字 **begin** 和 **end**

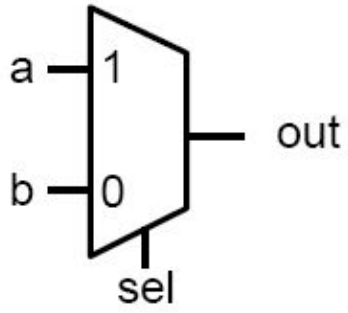
Recap Behavior Modeling

Combinational

```
module combinational(a, b, sel,
                     out);
  input a, b;
  input sel;
  output out;
  reg out;

  always @ (a or b or sel)
  begin
    if (sel) out = a;
    else out = b;
  end

endmodule
```

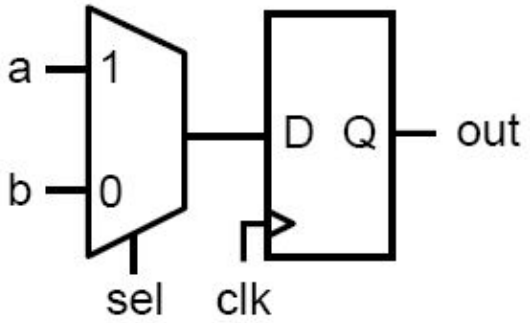


Sequential

```
module sequential(a, b, sel,
                  clk, out);
  input a, b;
  input sel, clk;
  output out;
  reg out;

  always @ (posedge clk)
  begin
    if (sel) out <= a;
    else out <= b;
  end

endmodule
```

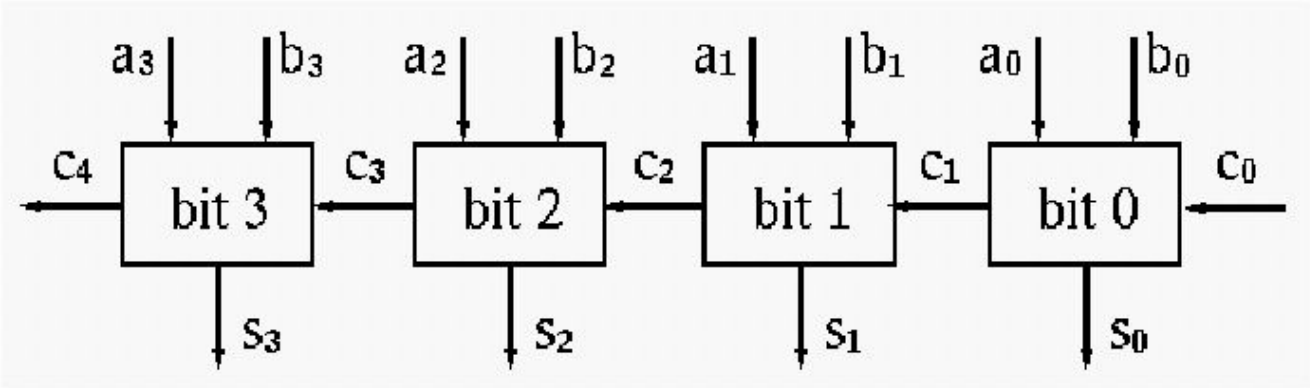


Chapter

- Verilog 複習 : Behavior-Level
- 4-bit Carry Look-ahead Adder
- Sequential Logic
- Latch and Flip-flop
- LAB 4-1 ~ 4-3

4-bit Carry Look-ahead Adder

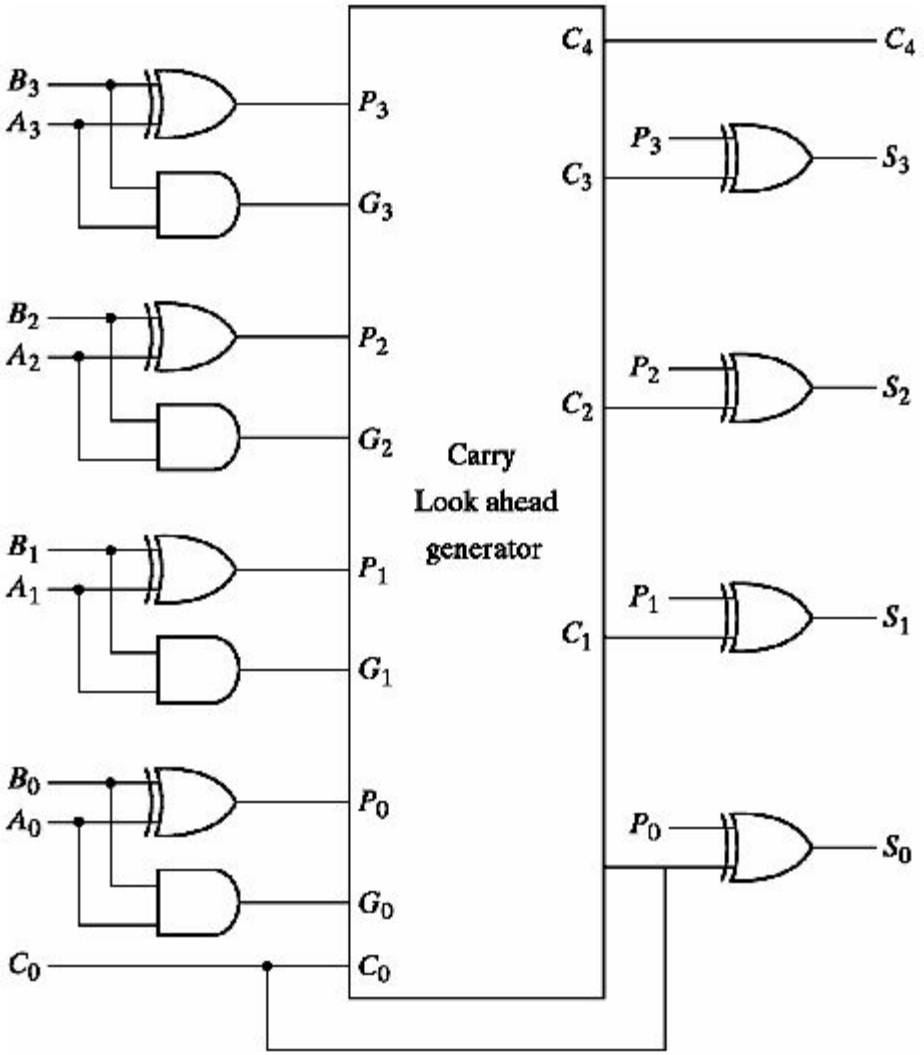
- 傳統的 ripple carry adder 在做加法時須要等上一級的 carry out 產生後才能繼續下去, 對於 16-bit adder, 甚至於 *n*-bits adder 所造成嚴重之 delay。



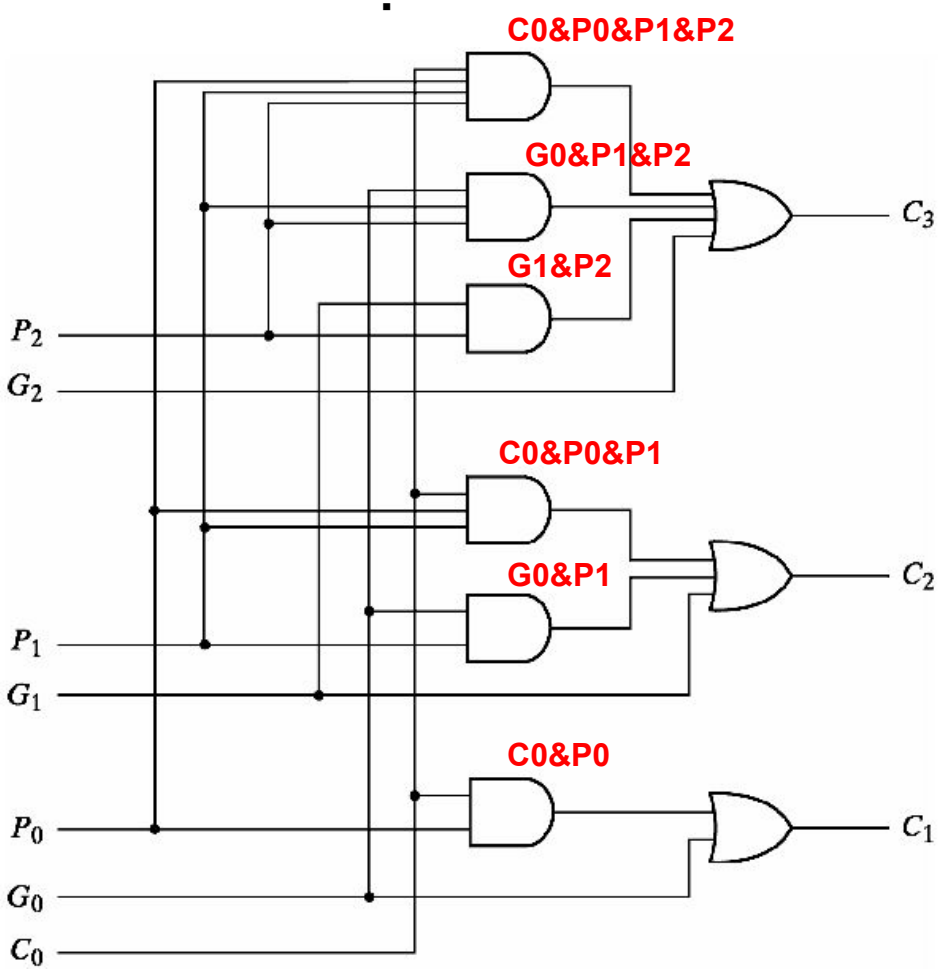
Ripple Carry Adder

- CLA 就是針對改善 delay 而設計出來的架構, 其想法是希望將所有的進位一次運算完成。

4-bit CLA


B_3
A_3
P_3
G_3
B_2
A_2
P_2
G_2
B_1
A_1
P_1
G_1
B_0
A_0
P_0
G_0
C_0
C_0
Carry
Look ahead
generator
C_4
C_3
C_2
C_1
C_4
P_3
S_3
P_2
S_2
P_1
S_1
P_0
S_0

4-bit CLA



Carry Look-ahead Generator

Chapter

- Verilog 複習 : Behavior-Level
- 4-bit Carry Look-ahead Adder
- Sequential Logic
- Latch and Flip-flop
- LAB 4-1 ~ 4-3

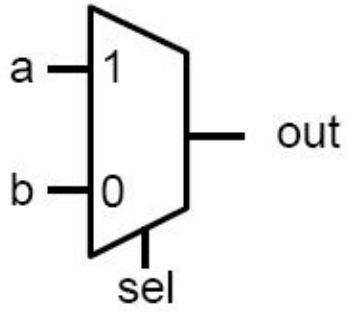
Sequential Logic

Combinational

```
module combinational(a, b, sel,
                     out);
  input a, b;
  input sel;
  output out;
  reg out;

  always @ (a or b or sel)
  begin
    if (sel) out = a;
    else out = b;
  end

endmodule
```

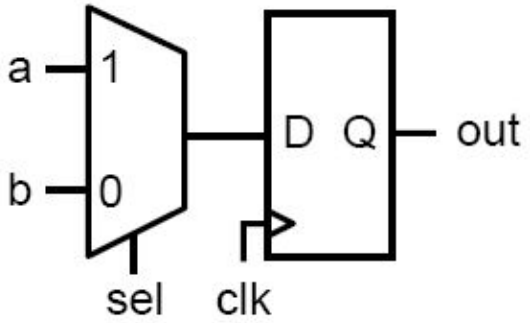


Sequential

```
module sequential(a, b, sel,
                  clk, out);
  input a, b;
  input sel, clk;
  output out;
  reg out;

  always @ (posedge clk)
  begin
    if (sel) out <= a;
    else out <= b;
  end

endmodule
```



Sequential Logic

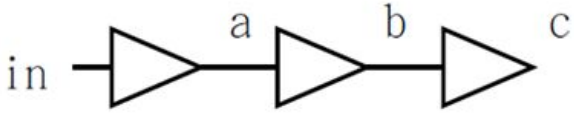
Blocking vs. Nonblocking：

Blocking (=), 具有順序性, 敘述會與先後有關係 (順序處理)

Non-Blocking (<=), 具有同時性, 敘述與先後沒有關係 (平行處理)

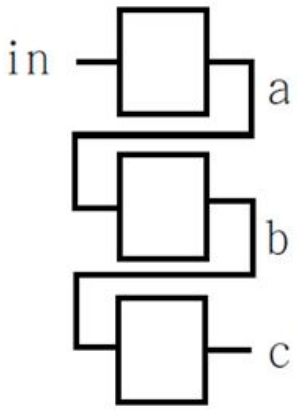
Blocking :

```
a = in;
b = a;
c = b;
//in=a=b=c
```



Nonblocking :

```
a <= in;
b <= a;
c <= b;
```



Sequential Logic

Blocking vs. Nonblocking：

```
input In;
reg [3:0] A, B, C;

always @( posedge CLK ) begin

    /* Blocking */       // 有順序執行，同C概念
    A[0] = In;           // 1CLK後A[0] = In
    A[1] = A[0];         // 1CLK後A[1] = A[0] = In
    A[2] = A[1];         // 1CLK後A[2] = A[1] = A[0] = In
    A[3] = A[2];         // 1CLK後A[3] = A[2] = A[1] = A[0] = In

    /* Non-Blocking */   // 同時執行，此範例有資料平移的效果
    B[0] <= In;          // 1CLK後B[0]存進In, B[1]存進B[0](存In之前的值)
    B[1] <= B[0];        // 2CLK後B[1]存進In
    B[2] <= B[1];        // 3CLK後B[2]存進In
    B[3] <= B[2];        // 4CLK後B[3]存進In
```

Chapter

- Verilog 複習 : Behavior-Level
- 4-bit Carry Look-ahead Adder
- Sequential Logic
- Latch and Flip-flop
- LAB 4-1 ~ 4-3

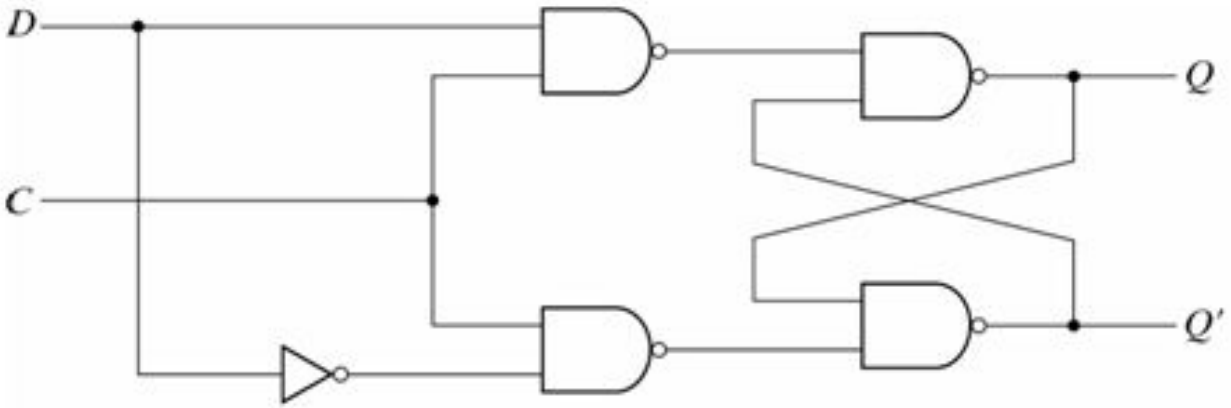
Latch and Flip-flop

- Latch 與 Flip-flop 均是數位電路中可以提供位元狀態儲存的裝置，它可以將邏輯狀態「0」或「1」存放在裝置內直到位元值需要改變或電源被切除。

- 由於有兩個穩定的輸出狀態，所以被稱為雙穩態電路。

Latch and Flip-flop

- D latch :



(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

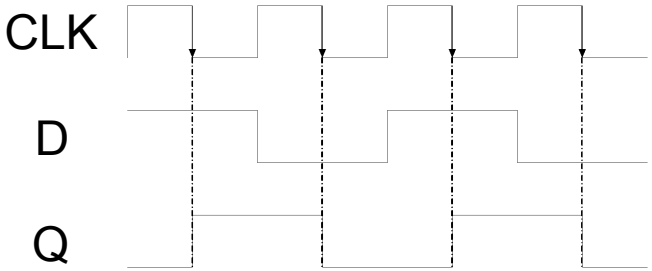
Latch and Flip-flop

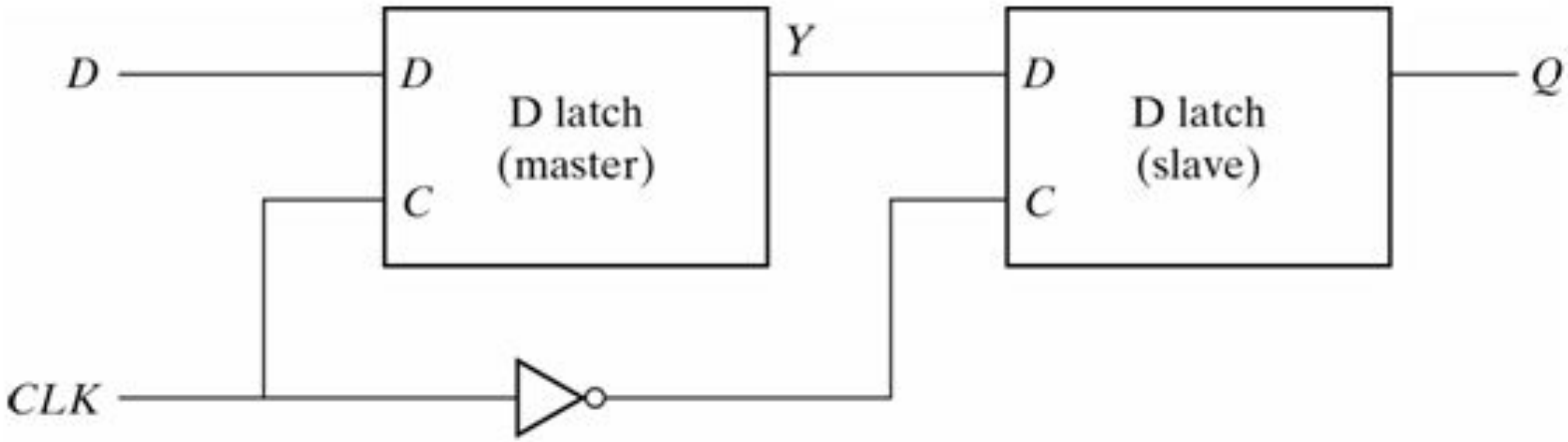
- D latch:

```
module D_latch(D, control, Q);
input D, control;
output Q;  //output reg Q;
reg Q;
always @(control or D)
if (control) Q = D;
endmodule
```

Latch and Flip-flop

- Master-slave D Flip-flop：





Latch and Flip-flop

- Characteristic Table of D Flip-flop ：

D	Q (t + 1)
0	0 (Reset)
1	1 (Set)

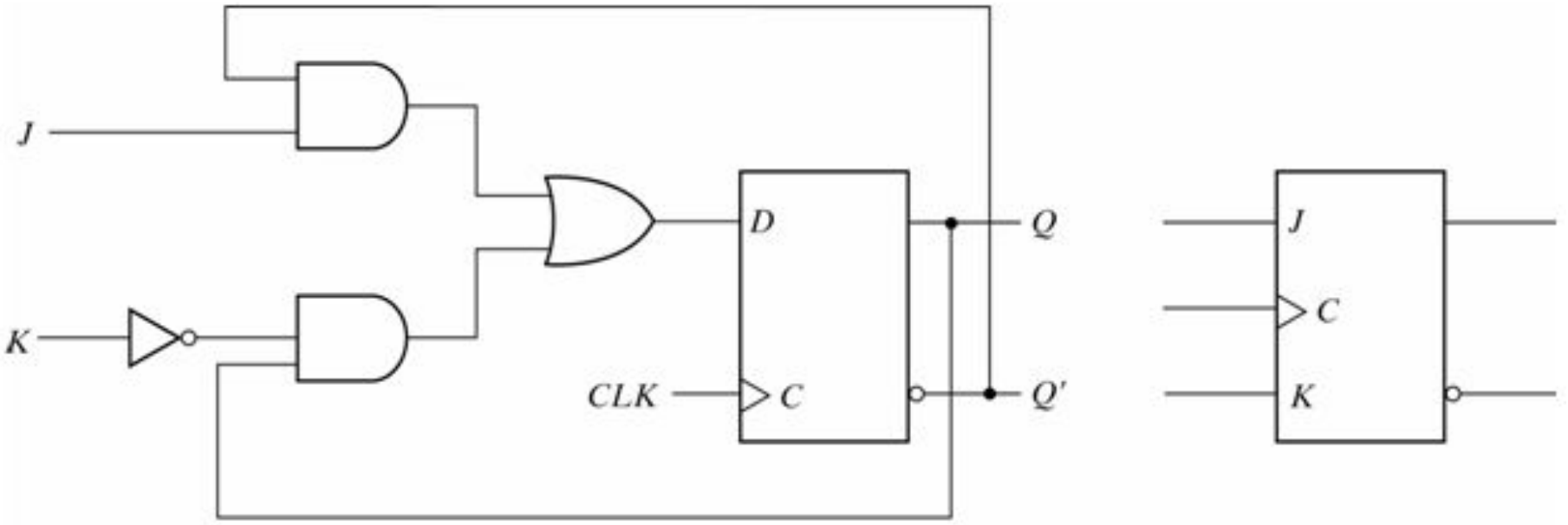
Latch and Flip-flop

- D Flip-flop:

```
module D_FF (D, CLK, Q);
input D, CLK;
output Q; //output reg Q;
reg Q;
always @(negedge CLK)
    Q <= D; //Q = D;
endmodule
```

Latch and Flip-flop

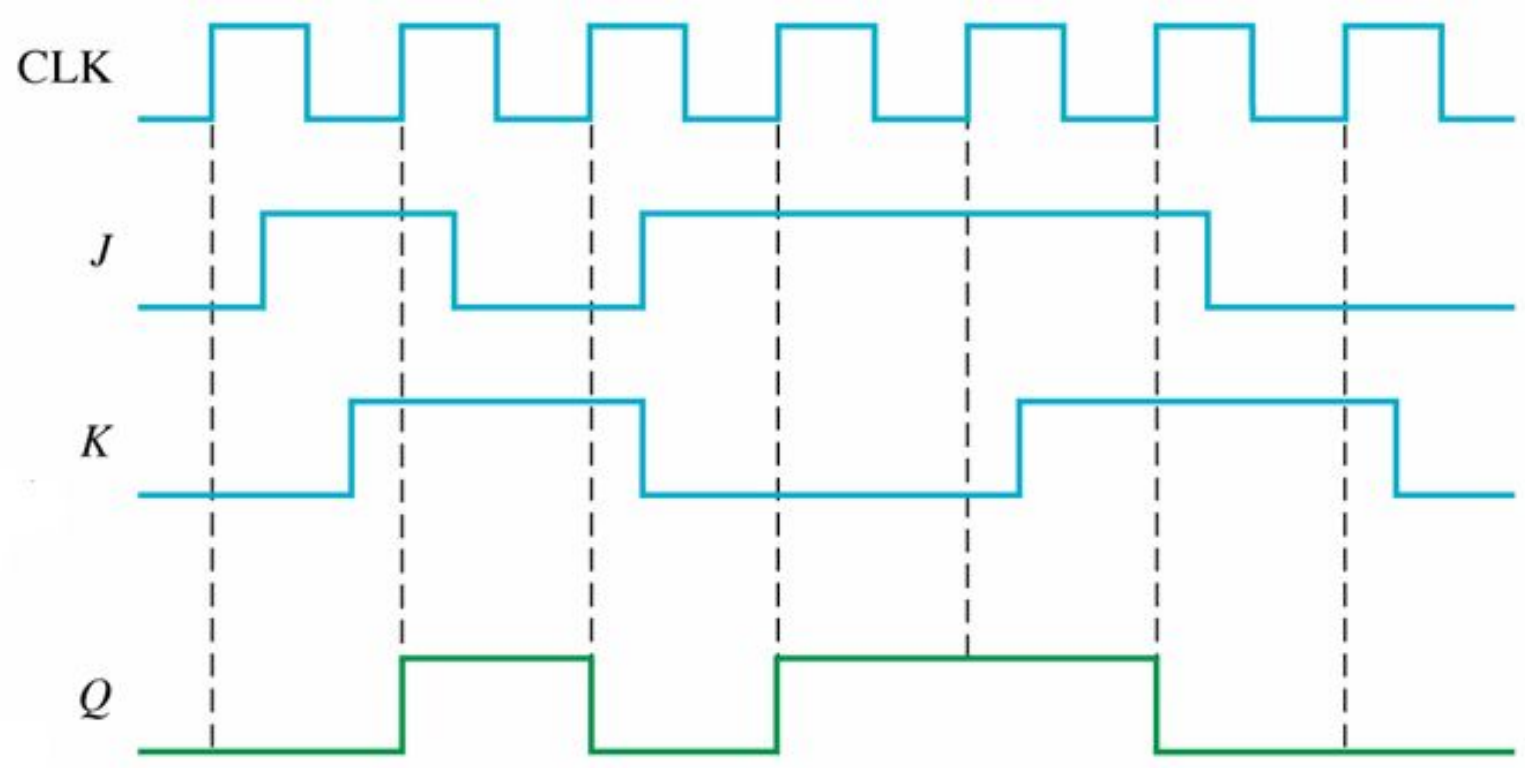
- JK Flip-flop



(a) Circuit diagram

(b) Graphic symbol

Latch and Flip-flop



JK Flip-flop 之時序圖

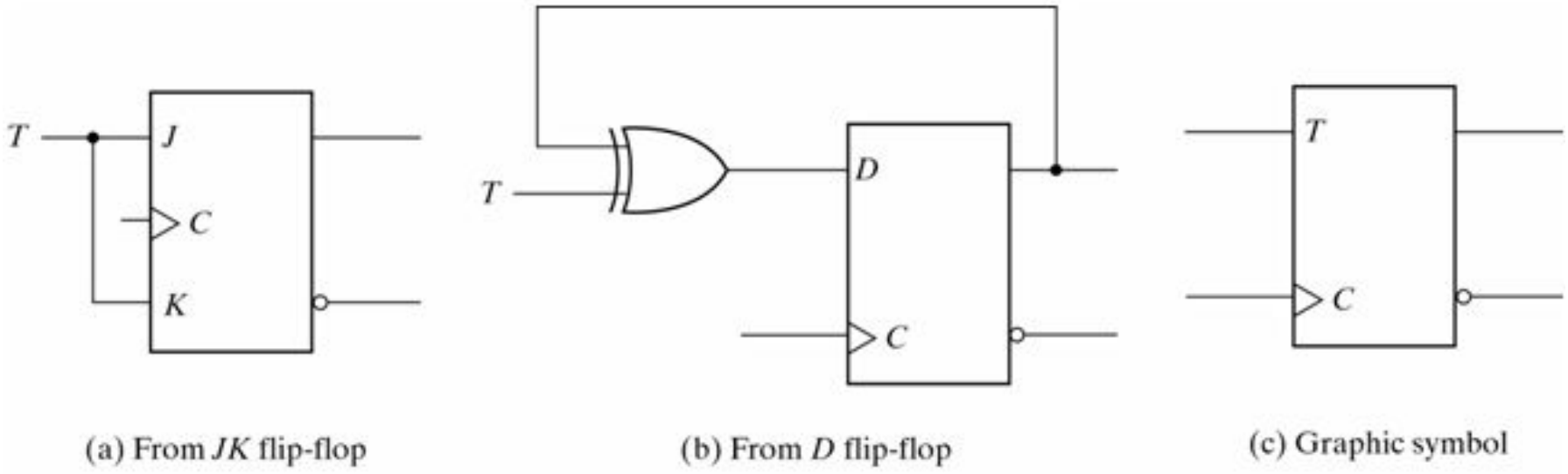
Latch and Flip-flop

- Characteristic Table of JK Flip-flop：

J	K	Q (t + 1)
0	0	Q (t) (No change)
0	1	0 (Reset)
1	0	1 (Set)
1	1	Q'(t) (Complement)

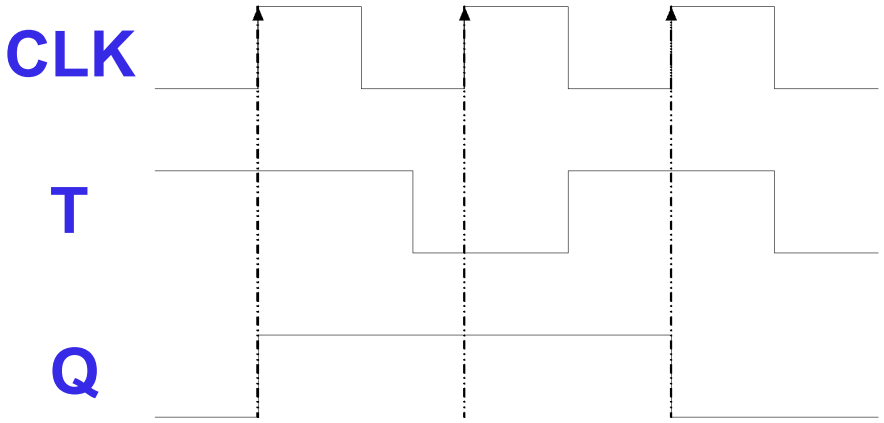
Latch and Flip-flop

- T Flip-flop：



(a) From *JK* flip-flop　　(b) From *D* flip-flop　　(c) Graphic symbol

Latch and Flip-flop



T	Q (t + 1)
0	Q (t) (No change)
1	Q'(t) (Complement)

Characteristic Table of T Flip-flop

Chapter

- Verilog 複習 : Behavior-Level
- 4-bit Carry Look-ahead Adder
- Sequential Logic
- Latch and Flip-flop
- LAB 4-1 ~ 4-3

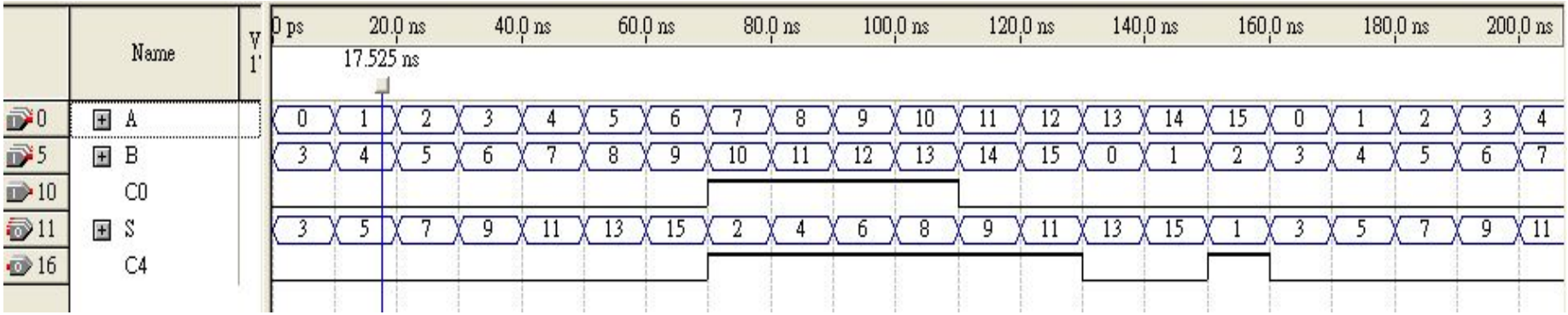
LAB 4-1

- 4-bit Carry Look-ahead Adder，A[3:0]、B[3:0]及C0為輸入，輸出為S[3:0]及C4，請使用Verilog HDL描寫出，並於Quartus II 模擬訊號波型加以驗證結果。

```
input [3:0]A,B;
input C0;
output [3:0]S;
output C4;
```

LAB 4-1

- 課堂檢查

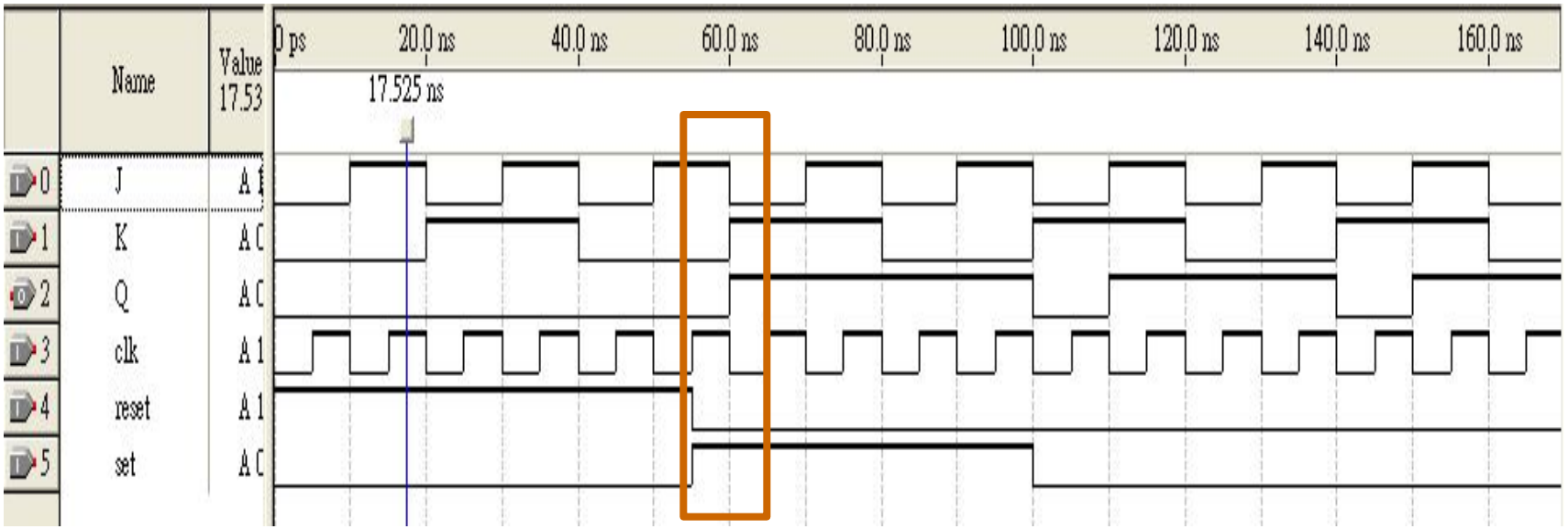

Name
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
160.0 ns
180.0 ns
200.0 ns
17.525 ns
0 A
5 B
10 C0
11 S
16 C4

LAB 4-2

- 請寫出含有set及reset之JK正反器(負緣觸發、以behavior方式描述，set與reset跟隨clk動作，set與reset都是等於1時動作)。

JK正反器

- 課堂檢查

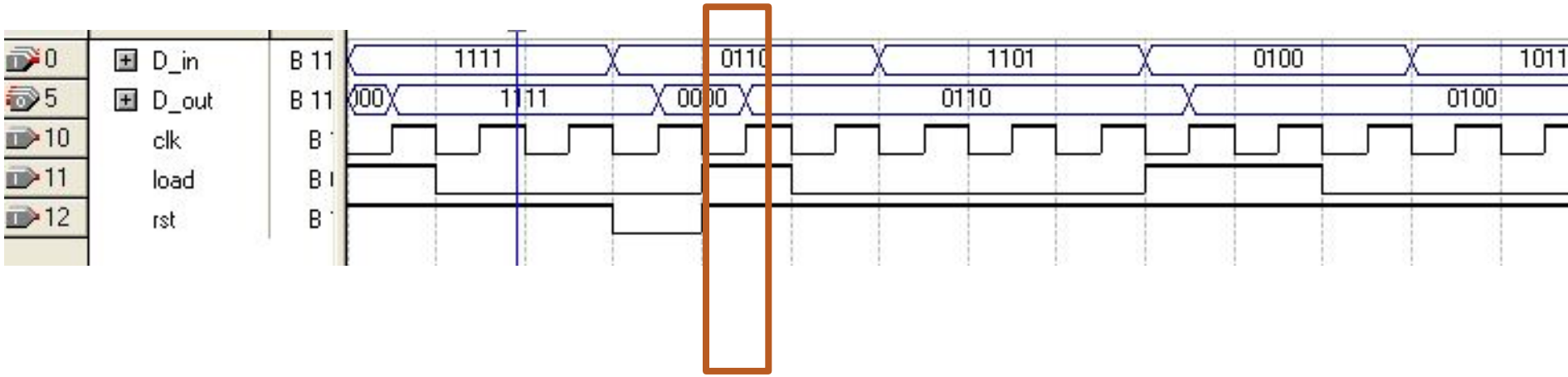

Name
Value
17.53
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
160.0 ns
17.525 ns
0 J
1 K
2 Q
3 clk
4 reset
5 set

LAB 4-3

- 使用Verilog HDL設計一個4位元具有平行輸入之暫存器, 其輸入為D_in[3:0], clk, rst, load,輸出為D_out [3:0]（clk為正緣觸發）。

4位元平行輸入之暫存器

- 課堂檢查



LAB4

下課前將各Lab繳交至moodle：

上傳verilog.v

波形截圖