Laboratory 12

組合電路與序向電路的差異

Department of Electrical Engineering
National Cheng Kung University

實驗目的

- 了解Blocking與Non-blocking的模擬行為差異
- 了解組合電路與序向電路在實作上的差異

使用器材

- 桌上型電腦
- Xilinx FPGA 板

Behavior model

- 抽象化邏輯閘層次(gate-level)的細節
- 提升整體設計的複雜度

```
module mylogic(A,B,C);
        input A,B;
        output C;
        wire D,E;

        OR    or1(A,B,D);
        AND and1(A,B,E);
        XOR xor1(D,E,C);
endmodule
```

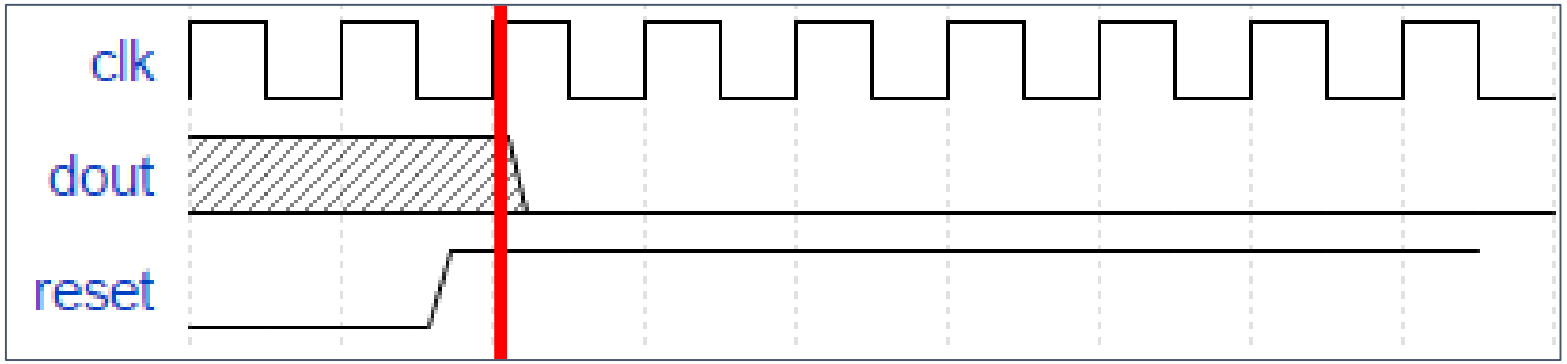
```
module mylogic(A,B,C);
        input A,B;
        output C;

        assign C = (A|B)^(A&B);



endmodule
```

Clock edge (1/3)

- always 中的 sensitive list 分為兩種
  - 正緣觸發 posedge
  - 負緣觸發 negedge
- 所有變數的值會以訊號緣當下的值為標準執行 always 中的運算
- 如果 sensitive list 沒有標明哪種觸發則 list 中的值一改變就會進行運算

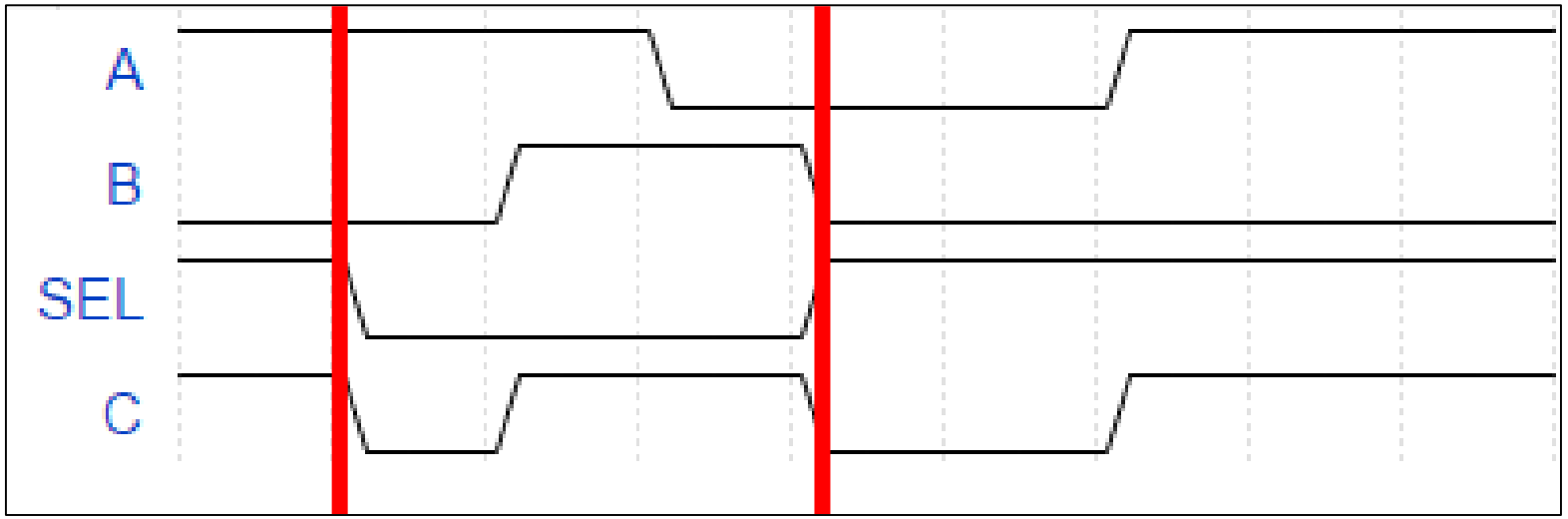
Clock edge (2/3)

```
always@(posedge clk)
begin
        if(reset)
                dout <= 0;
        else
                dout <=din;
end
```



Clock edge (3/3)

```
always@(A,B,SEL)
begin
        if(SEL)
                C <= A;
        else
                C <= B;
end
```



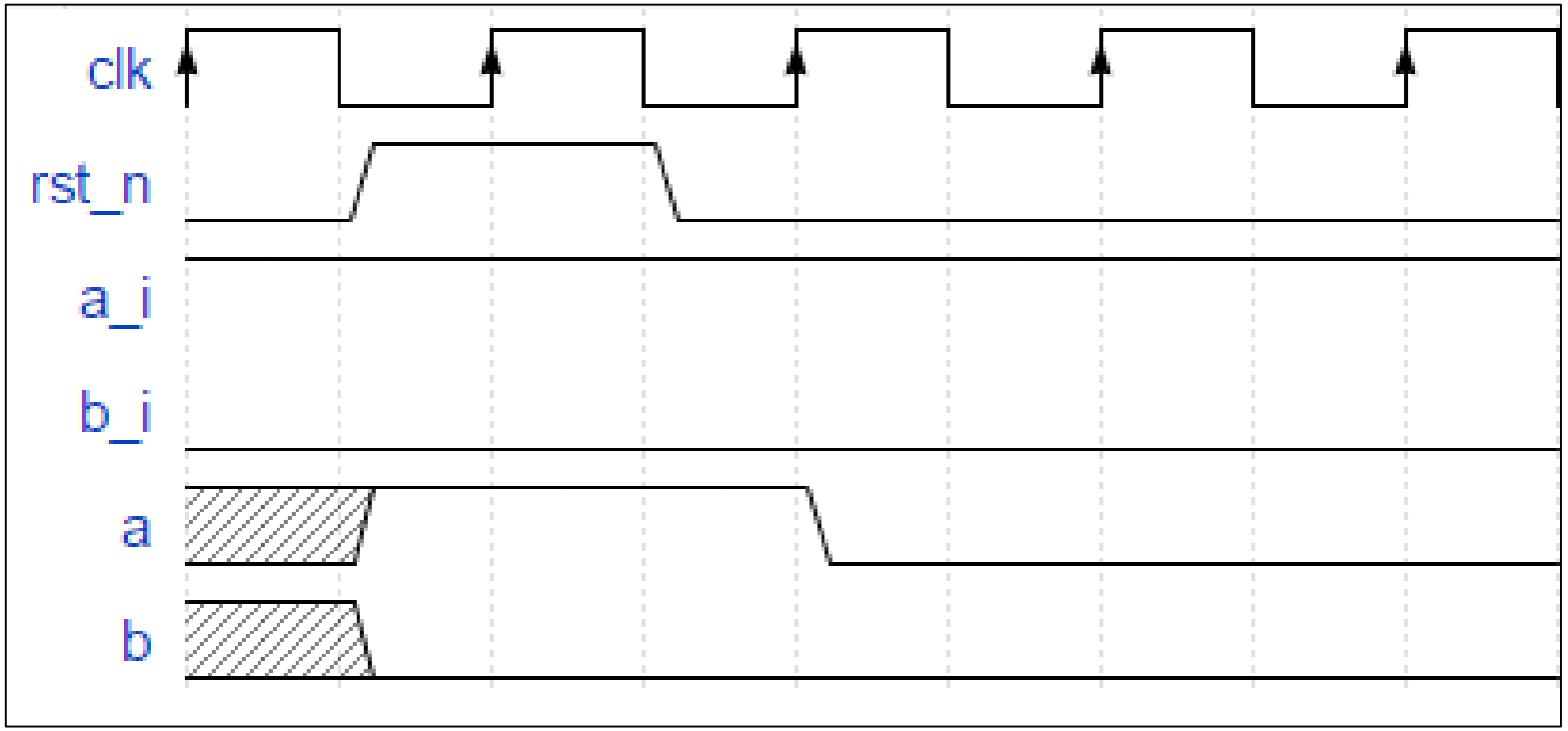
Blocking (1/2)

- 程式碼會依序一行一行執行

```
always@(posedge clk) begin
        if (rst) begin
                a = a_i;
                b = b_i;
        end
        else begin
                a = b;
                b = a;
        end
end
```

Blocking (2/2)

- 程式碼會依序一行一行執行



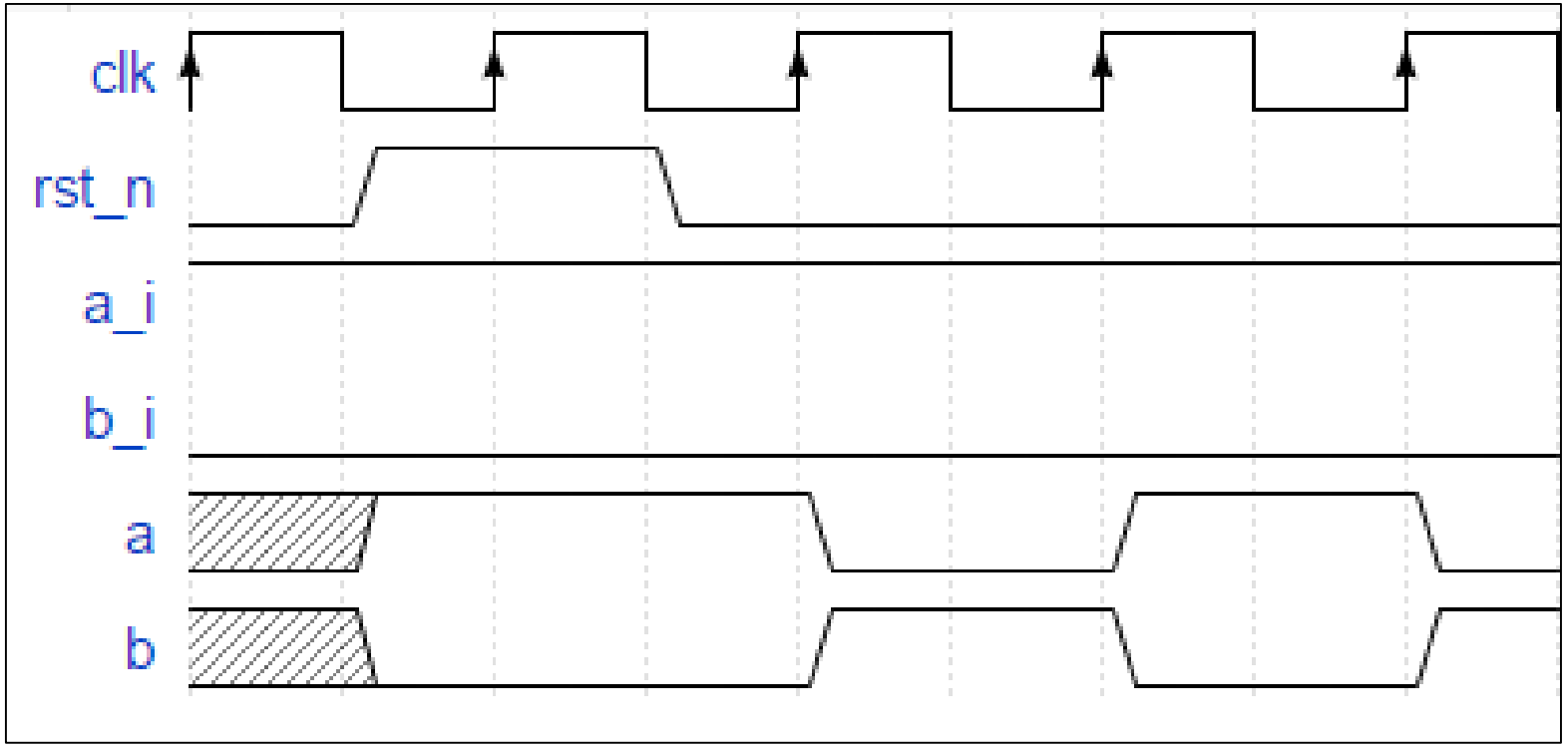
Nonblocking (2/2)

- 用於序向邏輯等需要同步的電路
- 該段程式碼以平行的方式執行

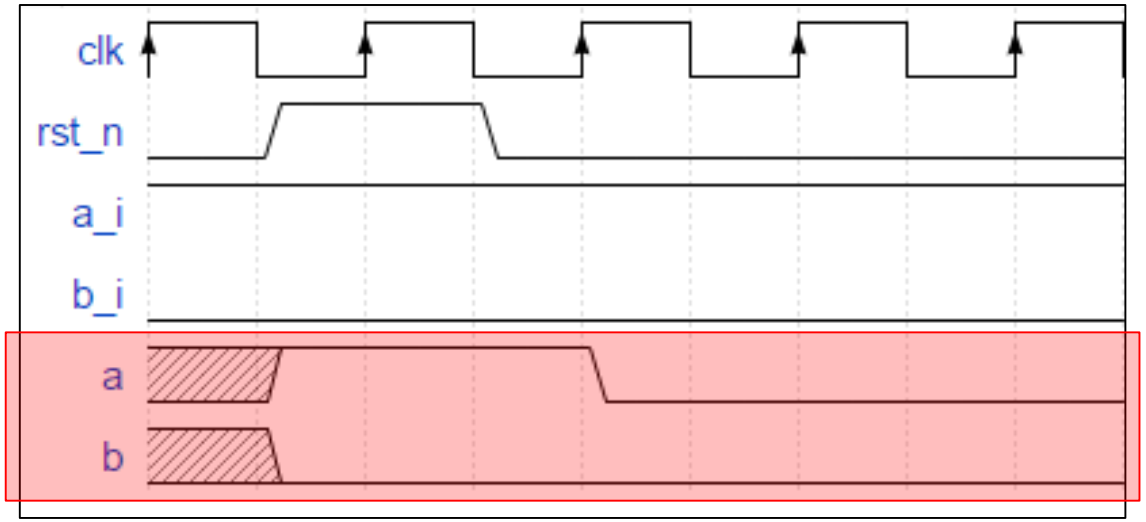
```
always@(posedge clk) begin
        if (rst) begin
                a <= a_i;
                b <= b_i;
        end
        else begin
                a <= b;
                b <= a;
        end
end
```

Nonblocking (2/2)

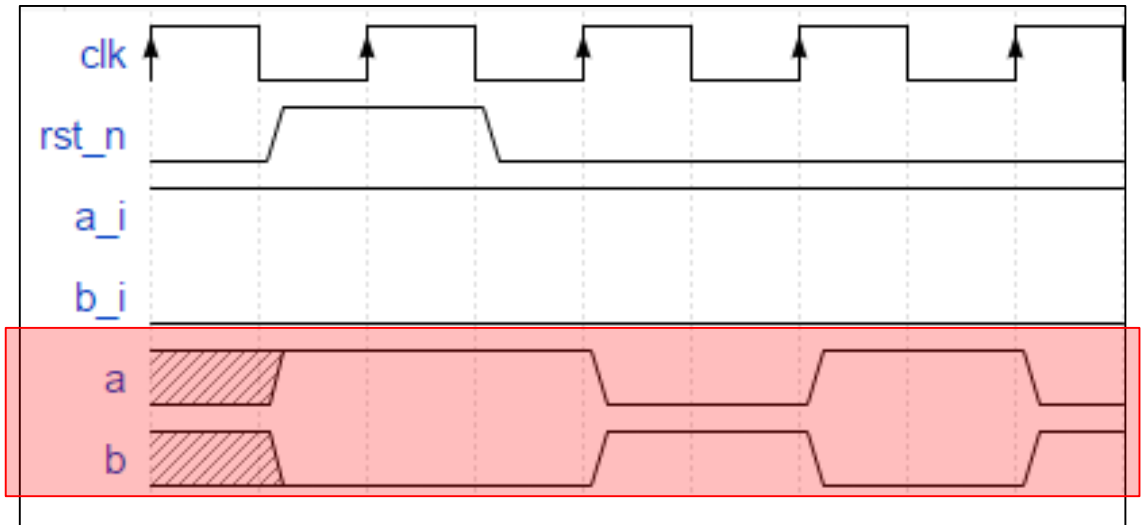
- 用於序向邏輯等需要同步的電路
- 該段程式碼以平行的方式執行



Blocking v.s Nonblocking



Blocking



Nonblocking

Sequential circuit (1/6)

- 稱為序向電路或是循序電路
- 該電路會依賴同步訊號來更新內部變數的值
- 同步訊號通常為系統的時脈(clock)
- 根據重置訊號的有無又分為:
  - 同步重置 (synchronous reset)
  - 非同步重置 (asynchronous reset)
- Latch、Flip Flop 等等皆屬之

Sequential circuit (2/6)

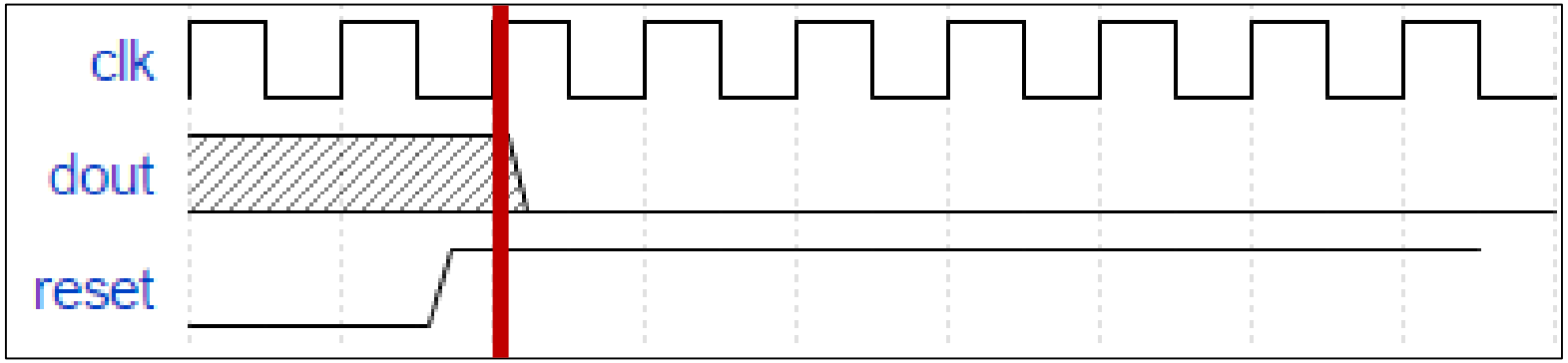
- 上下圖分別為非同步重置以及同步重置

```
always@(posedge clk) //重置訊號同步
begin
        if(reset)
                dout <= 0;
        else
                dout <=din;
end
```

```
always@(posedge clk or posedge reset) //重置訊號沒有同步
begin
        if(reset)
                dout <= 0;
        else
                dout <=din;
end
```

Sequential circuit (3/6)

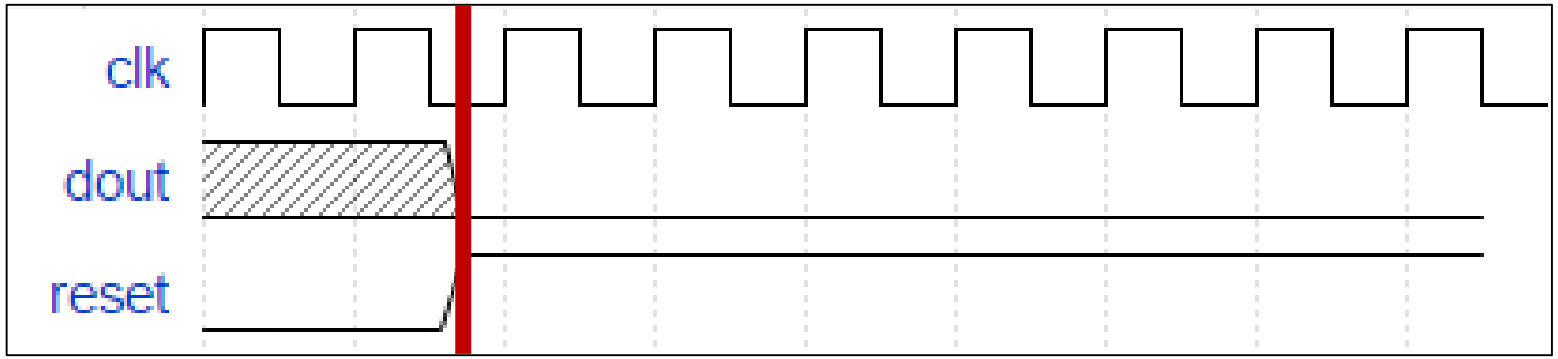
```
always@(posedge clk) //重置訊號同步
begin
        if(reset)
                dout <= 0;
        else
                dout <=din;
end
```



與時脈同步

Sequential circuit (4/6)

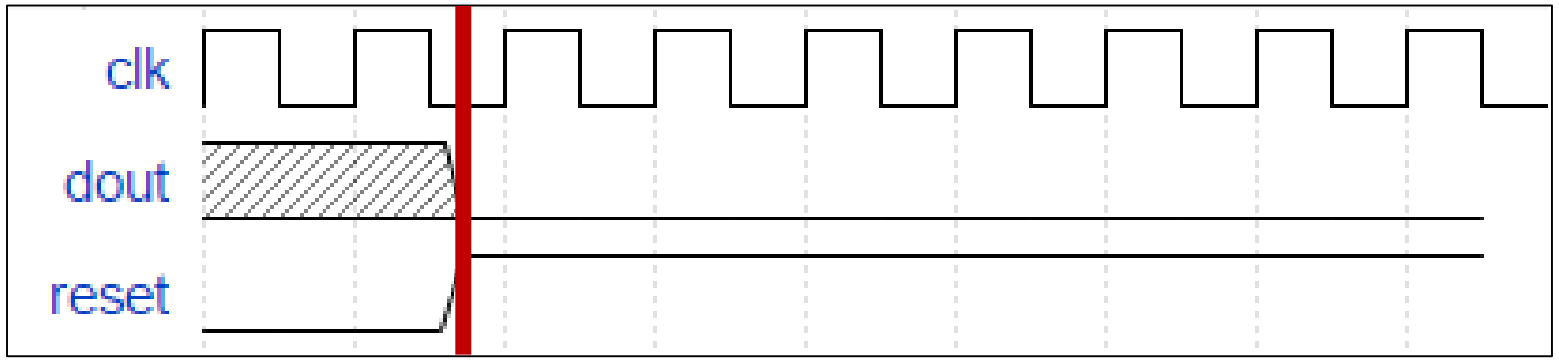
```
always@(posedge clk or posedge reset) //重置訊號沒有同步
begin
        if(reset)
                dout <= 0;
        else
                dout <=din;
end
```



與時脈不同步

Sequential circuit (5/6)

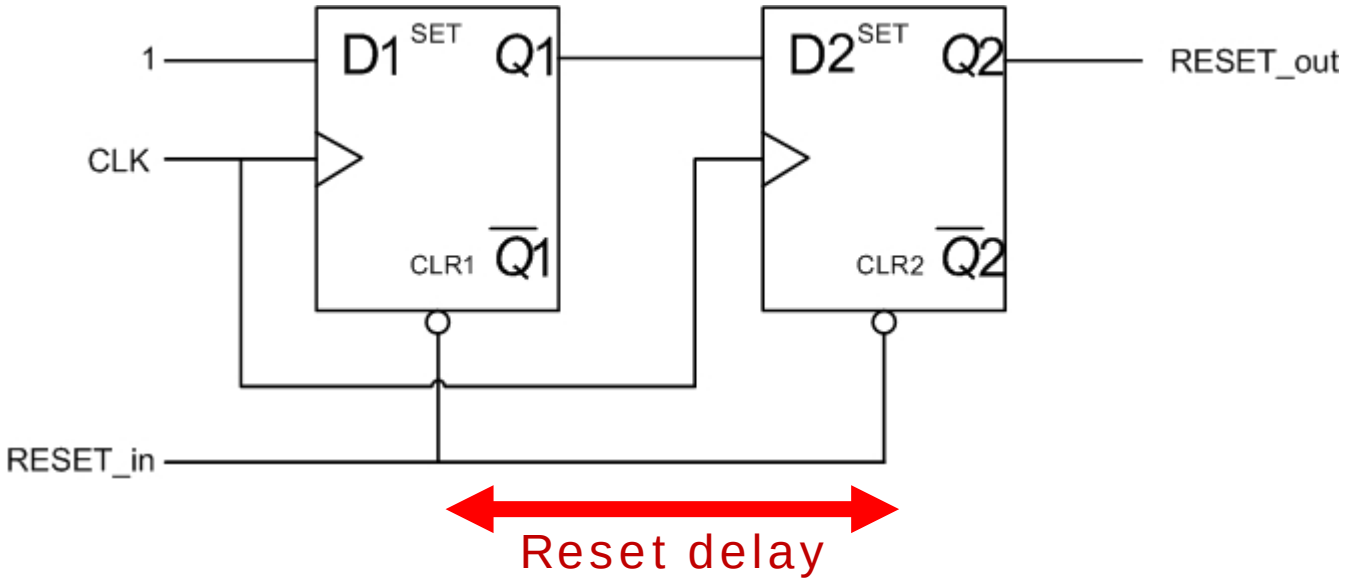
```
always@(posedge clk or posedge reset) //重置訊號沒有同步
begin
        if(reset)
                dout <= 0;
        else
                dout <=din;
end
```



與時脈不同步

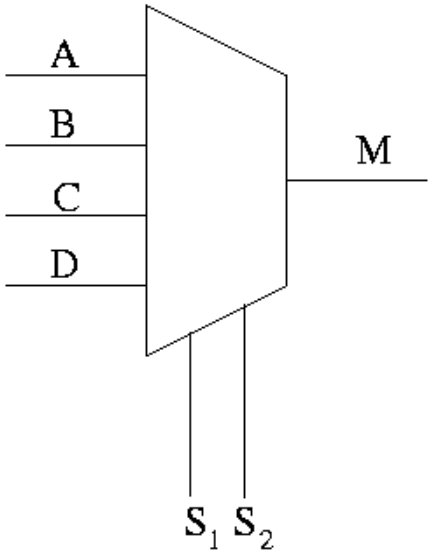
Sequential circuit (6/6)

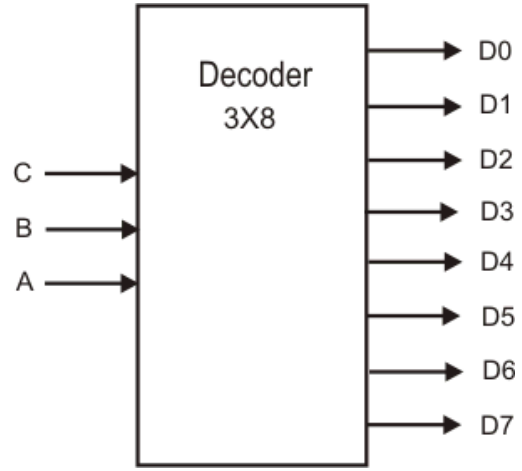
- 非同步重置的電路不須參考時脈,可以直接重置
- 非同步重置的電路在系統中容易因為 reset delay 的關係而造成暫存器間數值的不同步



Combinational circuit (1/2)

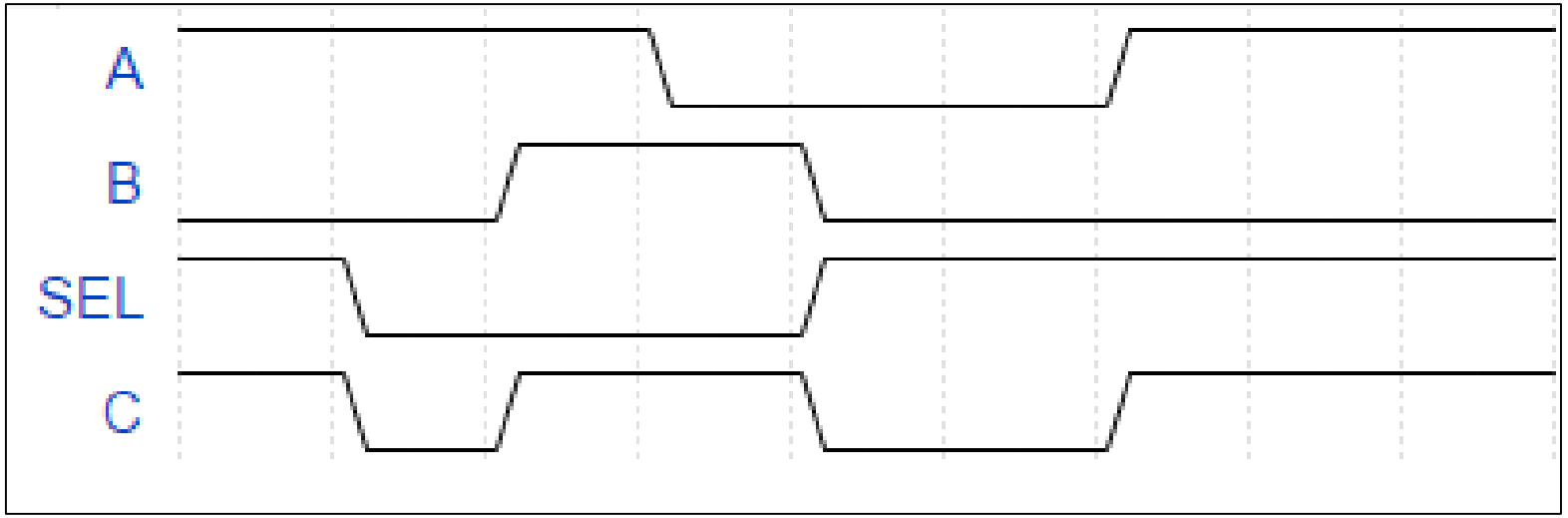
- 稱為組合電路
- 該電路內的任何訊號變動會**直接**反映在其他訊號上
- 沒有時脈進行同步
- 多工器、解碼器皆屬之





Combinational circuit (2/2)

```
always@(A,B,SEL)
begin
        if(SEL)
                C = A;
        else
                C = B;
end
```



Rules

- 以下為 blocking nonblocking 的四個重點
  1. 同一個 always 中不要混用blocking 和nonblocking
  2. 序向電路通常使用 nonblocking
  3. 組合電路通常使用 blocking
  4. assign 請使用 blocking

實作題

本次實作分為2個基本實作與1個結報問題

實作題(一) Blocking (1/4)

- 驗證 Non-blocking 和 blocking 之間的差異。
- 請使用虛擬裝置 (VeriInstrument) 產生兩個指撥開關來代表輸入訊號, 一個按鈕代表 reset 訊號以及兩個 LED 燈代表輸出訊號來驗證 Verilog code。

實作題(一) Blocking (2/4)

步驟

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab12_1
2. 完成下列*.v檔
   - clk_div.v
   - cnt_blk.v
   - lab12_1.v
3. 在虛擬儀表板上新增:
   - **兩個指撥開關**
   - **兩個LED燈**
   - **一個按鈕**

實作題(一) Blocking (3/4)

```
`define FRQ 48000_000
module clock_div(clk_48MHZ, clk_1HZ);

    input clk_48MHZ;
    output clk_1HZ;

    reg [31:0] counter_1HZ;
    reg clk_1HZ;

    always@(posedge clk_48MHZ)begin
        if(counter_1HZ<`FRQ)
            counter_1HZ <=  counter_1HZ+32'd1;
        else
            counter_1HZ <= 32'd1;
    end

    always@(posedge clk_48MHZ)begin
        if(counter_1HZ<=`FRQ/2)
            clk_1HZ <= 32'd1;
        else
            clk_1HZ <= 32'd0;
    end


endmodule
```

clock_div.v

```
module cnt_blk(clk,rst,a_in,b_in,a_out,b_out);

    input clk,rst,a_in,b_in;
    output a_out,b_out;

    reg a,b;

    assign a_out = a;
    assign b_out = b;


    always@(posedge clk or negedge rst)begin
        if(~rst)begin
            a = a_in;
            b = b_in;
        end
        else begin
            a = b;
            b = a;
        end
    end


endmodule
```

cnt_blk.v

實作題(一) Blocking (4/4)

```
`include "clk_div.v"
`include "cnt_blk.v"

module lab12_1(clock,reset,a_in,b_in,a_out,b_out);

    input clock,reset,a_in,b_in;
    output a_out,b_out;

    clock_div clock_div(
        .clk_48MHZ(clock),
        .clk_1HZ  (clk_1HZ)
    );

    cnt_blk counter(
        .clk  (clk_1HZ),
        .rst  (reset),
        .a_in (a_in),
        .b_in (b_in),
        .a_out(a_out),
        .b_out(b_out)
    );


endmodule
```

實作題(二) Nonblocking (1/3)

步驟

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab12_2
2. 完成下列*.v檔
   - clk_div.v
   - cnt_blk.v
   - lab12_2.v
3. 在虛擬儀表板上新增:
   - **兩個指撥開關**
   - **兩個LED燈**
   - **一個按鈕**

實作題(二) Nonblocking (2/3)

```
`define FRQ 48000_000
module clock_div(clk_48MHZ, clk_1HZ);

    input clk_48MHZ;
    output clk_1HZ;

    reg [31:0] counter_1HZ;
    reg clk_1HZ;

    always@(posedge clk_48MHZ)begin
        if(counter_1HZ<`FRQ)
            counter_1HZ <=  counter_1HZ+32'd1;
        else
            counter_1HZ <= 32'd1;
    end

    always@(posedge clk_48MHZ)begin
        if(counter_1HZ<=`FRQ/2)
            clk_1HZ <= 32'd1;
        else
            clk_1HZ <= 32'd0;
    end


endmodule
```

clock_div.v

```
module cnt_nblk(clk,rst,a_in,b_in,a_out,b_out);

    input clk,rst,a_in,b_in;
    output a_out,b_out;

    reg a,b;

    assign a_out = a;
    assign b_out = b;


    always@(posedge clk or negedge rst)begin
        if(~rst)begin
            a <= a_in;
            b <= b_in;
        end
        else begin
            a <= b;
            b <= a;
        end
    end


endmodule
```

cnt_blk.v

實作題(二) Nonblocking (3/3)

```
`include "clk_div.v"
`include "cnt_nblk.v"

module lab12_2(clock,reset,a_in,b_in,a_out,b_out);

    input clock,reset,a_in,b_in;
    output a_out,b_out;

    clock_div clock_div(
        .clk_48MHZ(clock),
        .clk_1HZ  (clk_1HZ)
    );

    cnt_nblk counter(
        .clk  (clk_1HZ),
        .rst  (reset),
        .a_in (a_in),
        .b_in (b_in),
        .a_out(a_out),
        .b_out(b_out)
    );


endmodule
```

結報問題

1. 請解釋 clk_div.v 的作用為何？
2. 請比較 Lab12-1 和 Lab12-2 的結果並解釋軟體上是如何實現 non-blocking 的運算行為？