# 邏輯系統實驗

Lab 7　Verilog – Sequential Design (2)

2022/04/20

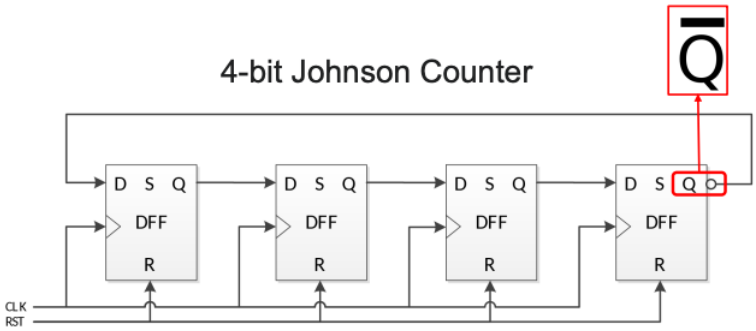

| 第 5 組 | |
|---|---|
| 組員姓名 | 學號 |
| 林珮玉 | E24084096 |
| 廖本恩 | E24102179 |
| 蘇冠誠 | E24084143 |

# 實作題(一): 6-bit Johnson Counter

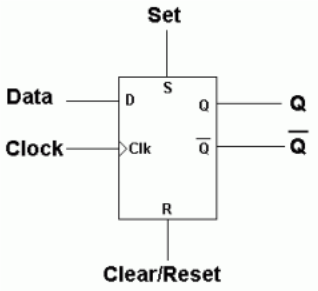
1. 簡述題目：

使用 Use 6 Asynchronous Set/Reset D Flip-Flop 建立 6-bit Johnson Counter。

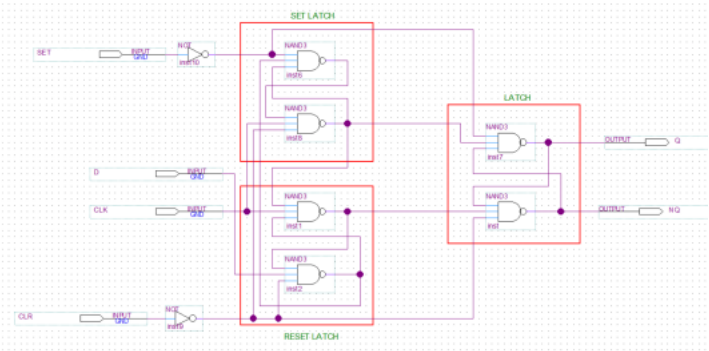
| Johnson counter | | | | |
|---|---|---|---|---|
| State | Q0 | Q1 | Q2 | Q3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| *0* | 0 | 0 | 0 | 0 |




D Flip-Flop Template




Asynchronous Set/Reset D Flip-Flop



2. 實現方式：

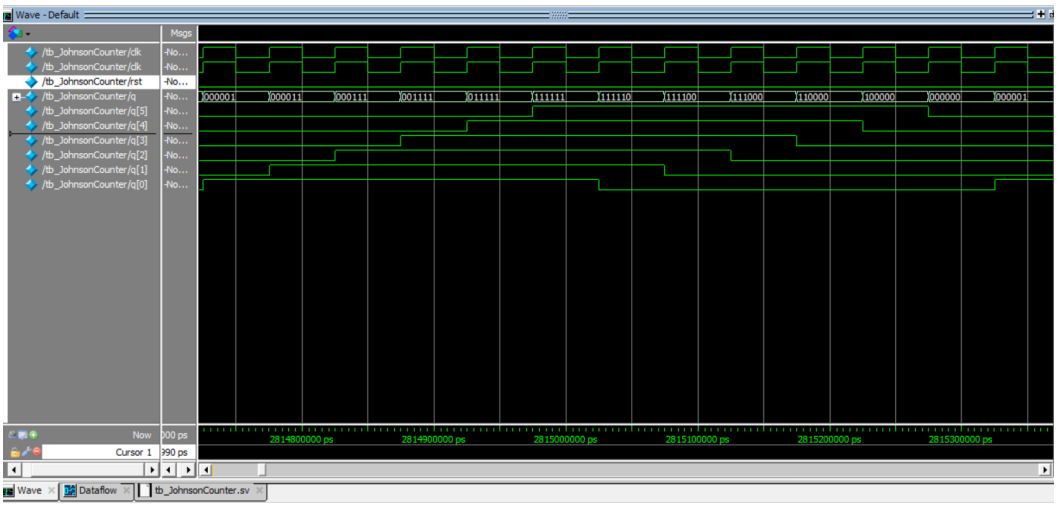
**JohnsonCounter.v**

```
module JohnsonCounter(clk, rst, q);
    input       clk;
    input       rst;
    output [5:0] q;

    reg [5:0] q;

    always @(posedge clk)
    begin
```

```
    if(rst)
        q=6'd0;
    else
        begin
            q[5]<=q[4];
            q[4]<=q[3];
            q[3]<=q[2];
            q[2]<=q[1];
            q[1]<=q[0];
            q[0]<=(~q[5]);
        end
    end
endmodule
```

3. 結果呈現

Waveform of 6-bit Johnson Counter

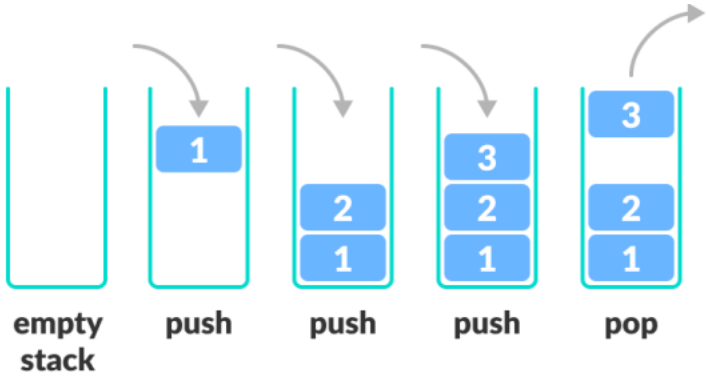



# 實作題(二): Stack with 8 Element

1. 簡述題目：

藉由 LIFO (Last in, First out) 的概念，使用 combination of Vector and Array 建立 8-bit 的 Stack。過程利用 reg[7:0] 或 memory[7:0] 建立 Stack 的記憶元素，並創建一個變數來記錄最新的元素。而計算的部分需使用 combination Block，更新記憶體的部分則是使用 Sequential

Block。



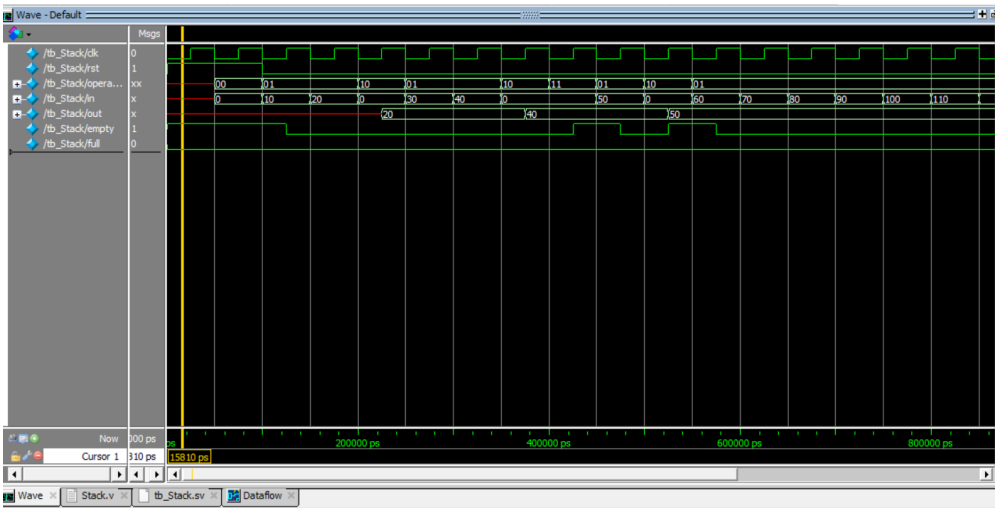

2. 實現方式：

**Stack.v**

```
module Stack(clk, rst, operation, in, out, empty, full);
    input       clk;
    input       rst;
    input [1:0] operation;
    input [7:0] in;
    // Register
    output reg [7:0] out;
    output reg   empty;
    // Wire
    output reg   full;
    integer pointer;
    reg [7:0] memory [7:0];

    always @(rst)
    begin
        pointer = -1;
        empty = 1;
        full = 0;
    end

    always @(posedge clk)
    begin
        case(operation)
            2'b00:;
            2'b01:
                if(full === 1)
```

```
                $display("Stack is full!\n");
            else
                begin
                    pointer = pointer+1;
                    memory[pointer] = in;
                end
        2'b10:
            if(empty === 1)
                $display("Stack is empty!");
            else
                begin
                    out = memory[pointer];
                    pointer = pointer-1;
                end
        2'b11:
            pointer = -1;
        default;
    endcase

    begin
    if(pointer === -1)
        begin
            empty = 1;
            full = 0;
        end
    else if(pointer === 7)
        begin
            full = 1;
        end
    else
        begin
            full = 0;
            empty = 0;
        end
    end
  end
endmodule
```

3. 結果分析：




先看有 Reset 的時候，當 operator 為 01 時會進行 push，把數字 10, 20 推入 stack，接著 10 是 pop 把後面的數字 20 pop 出來，然後一樣 01 繼續將數字 30, 40, 0, 50 送入，遇 10 則把最後面的 50 pop 出來，遵循 LIFO 依此類推。

# 實作題(三)：Queue with 8 Element

1. 簡述題目：

藉由 FIFO (First in, First out) 的概念，使用 combination of Vector and Array 建立 8-bit 的 Queue。過程利用 reg[7:0] 或 memory[7:0] 建立 Stack 的記憶元素，並創建一個變數來記錄最新的元素。而計算的部分需使用 combination Block，更新記憶體的部分則是使用 Sequential Block。

<注> Dequeue 過程永遠是丟去最老的那個，丟掉之後要將所有的元素 shift one 並將 index variable 減一。

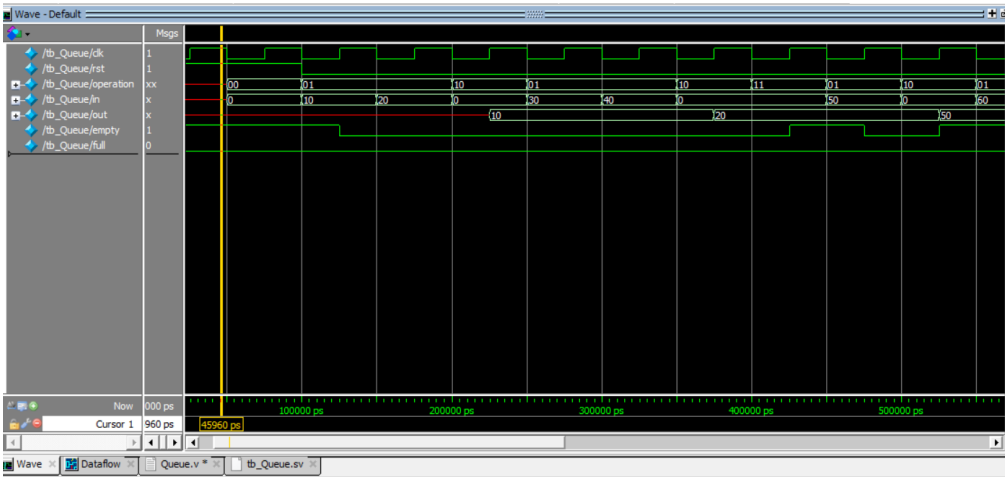

2. 實現方式：

| **Queue.v** |
| --- |

```
module Queue(clk, rst, operation, in, out, empty, full);
    input      clk;
    input      rst;
    input [1:0] operation;
    input [7:0] in;
    // Register
    output reg [7:0] out;
    output reg   empty;
    // Wire
    output reg   full;
    reg [7:0] memory [7:0];
    integer front;
    integer rear;

    always @(rst)
    begin
        front = 0;
        rear = 0;
        empty = 1;
        full = 0;
    end

    always @(posedge clk)
    begin
        case(operation)
            2'b00:;
            2'b01:
                begin
                    if(full === 1)  $display("Queue is full!\n");
                    else
                        begin
                            memory[rear] = in;
                            rear = rear + 1;
                            empty = 0;
                            if(rear === 8)   rear = 0;
                            if(front === rear)  full = 1;
                        end
                end
```

```
            2'b10:
                begin
                    if(empty === 1)  $display("Queue is empty!\n");
                    else
                        begin
                            out = memory[front];
                            front = fronmodule Queue(clk, rst,
operation, in, out, empty, full);
            2'b11:
                begin
                    front = 0;
                    rear = 0;
                    empty = 1;
                    full = 0;
                end
            default:    $display("Error!");
        endcase
    end
endmodule
```

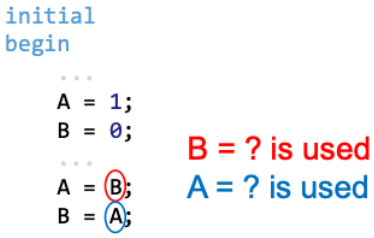
## 3. 結果分析：

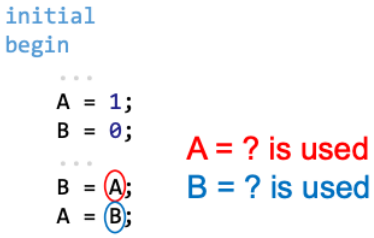



看有 Reset 的時候，當 operator 為 01 時會進行 push，把數字 10, 20 推入 Queue，接著 10 是 pop 把第一個加入的數字 10 pop 出來，然後一樣 01 繼續將數字 30, 40, 0, 50 送入，遇 10 則把首位 20 pop 出來，遵循 FIFO 依此類推。

# Test Yourself's Answer

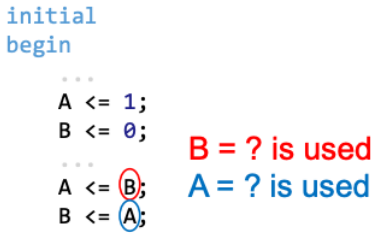
1. Blocking

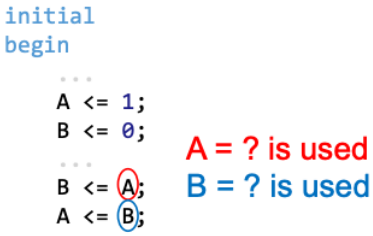



---

2. Non-Blocking




| | Initial Begin | Initial Begin |
|---|---|---|
| **1. Blocking** | **A=0, B=0** | **A=1, B=1** |
| **2. Non-Blocking** | **A=1, B=0** | **A=1, B=0** |

# 心得

## 組員一 林珮玉 E24084096

時隔兩週又是令人期待的 verilog 課程，這週是我們第一次在課堂跑完模擬（終於不用再拿回家跑再用 mail 傳給助教檢查了）也很幸運的成為第一個做完的組別，果然如同我的夥伴所說的，安裝好軟體接下來的實驗就會十分順利。這次課程安排得很不錯，有先幫我們複習之前的 Clock、Latch & Flip-Flop，然後再銜接到 Synchronous & Asynchronous Reset 以及 Blocking 和 Non-blocking 的指派方法。在預習的時候覺得有點挑戰性，原本以為自己大概是要待好待滿才有辦法寫完，不過很感謝我的夥伴，一次把 Stack 和 Queue 解決，而我負責 Johnson Counter。這次這次花了半小時在檢查語法問題，希望藉由這次磨練，下週會更順利！

## 組員二 廖本恩 E24102179

本次實驗實作到 stack 跟 queue 兩個資料結構，其實比較像在寫 C，只是變數改成二進位而已。而本次也因為語法的問題花了蠻多時間，例如大括號及暫存數值要用 register 而不是 wire 等，但是也學習到如何處理 verilog compile 時的 error，相信對之後的實驗很有幫助

# 組員三 蘇冠誠 E24084143

verilog 筆記：

1. module 寫法：
   module_type module_name(i/o);
   module_type(i/o);
2. 多數指令記得要加分號作結

追加：

3. verilog 不(常)用大括號，請用 begin...end 或 case...endcase 等等指令作替換
4. 當 error 出現在同步或非同步替代時，那麼就先去看看輸出是不是還沒設成 output reg，這類問題通常出現在這地方