Computer Organization 2022

HOMEWORK 3

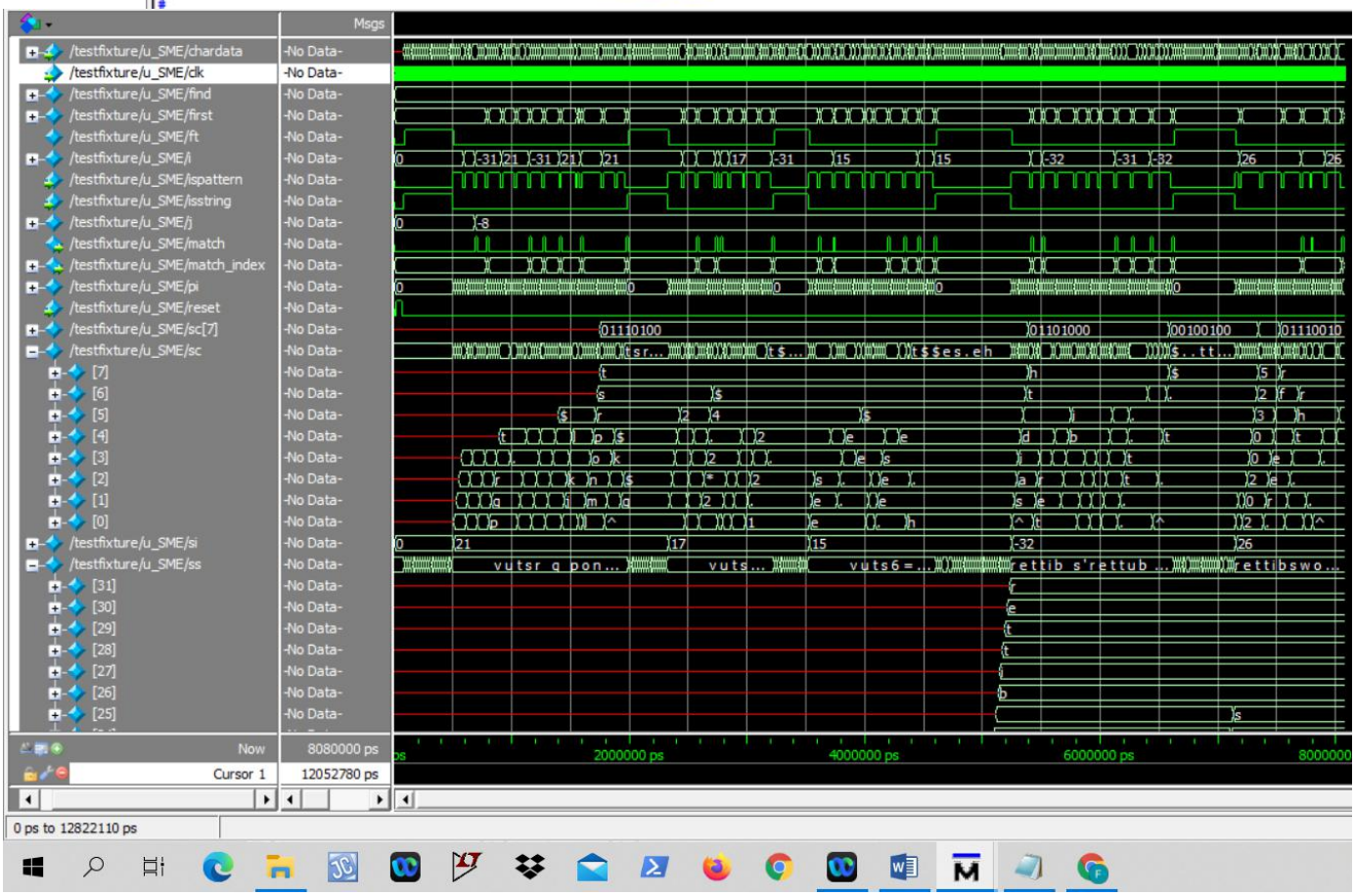
系級: 資訊工程系　　學號: F74109016　　姓名: 葉惟欣

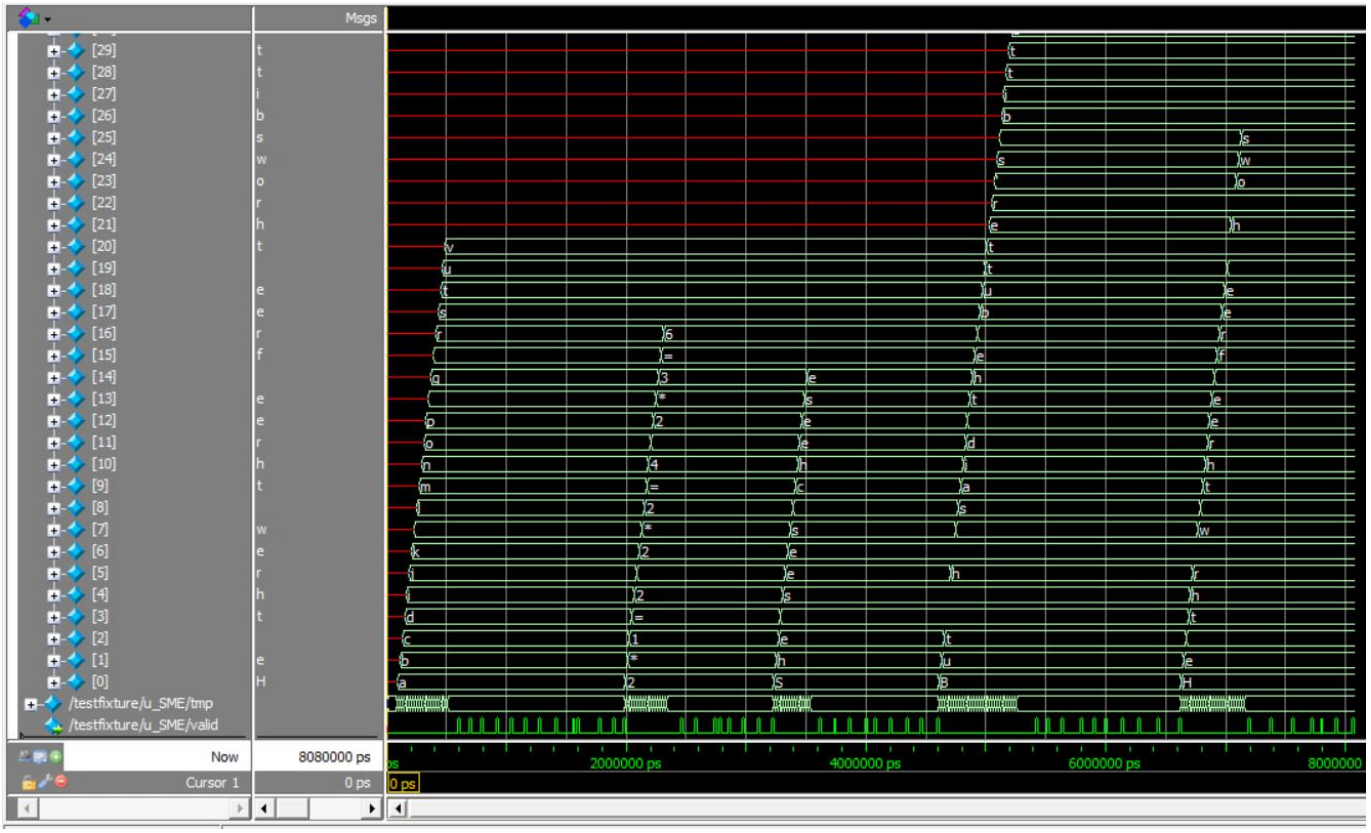
實驗結果圖:

(波形圖及模擬完成截圖)

模擬完成截圖

```
Transcript
#    -- Pattern 3  ".ree  fr"
#         cycle 17a, expect(0,--) , get(0,--) >> Pass
#    -- Pattern 4  "re. thr"
#         cycle 182, expect(1,05) , get(1,05) >> Pass
#    -- Pattern 5  "..."
#         cycle 186, expect(1,00) , get(1,00) >> Pass
#    -- Pattern 6  "^...$"
#         cycle 18c, expect(0,--) , get(0,--) >> Pass
#    -- Pattern 7  "^....$"
#         cycle 193, expect(1,0f) , get(1,0f) >> Pass
# ------------------------------------------------------------
# ---------------- Simulation finish, ALL PASS  ----------------
# -- cycle =404 , Score =100       --
#
#
#          ****************************
#          **                        **       /|__/|
#          **  Congratulations !!    **      / O,O  |
#          **                        **    /_____   |
#          **  Simulation PASS!!     **   /^ ^ ^ \  |
#          **                        **  |^ ^ ^ ^ |w|
#          ************** ************   \m___m__|_|
#
```





程式運作流程:(簡單說明波形變化的意義)

兩個組合電路做比對的工作，循序電路則餵 chardata。

```
if(ispattern == 1'b1) //讀輸入patter
begin
        sc[pi] = chardata;
        pi = pi + 5'b00001;
        if(ft == 1'b1)
        begin
                si = tmp;
                tmp = 6'b000000;
                ft = 1'b0;
        end
end

else if(isstring == 1'b1) //讀輸入string
begin
        ss[tmp] = chardata;
        tmp = tmp + 6'b000001;
        if(ft == 1'b0)
        begin
                ft = 1'b1;
                pi = 5'b00000;
        end
end
```

循序電路

在每次clk來的時候，判斷現在isstring或 ispattern是不是為正緣。如果是正緣就讀進去。而 **isstring有多少個字元存在就存在si裡面**，tmp 是一個暫存字串有幾個字元的變數，**一旦 ispattern 為正緣的時候，tmp 就把值賦值給 si，而 tmp 則變為 0，讓下一次讀入 string，可以從陣列 index = 0 的時候開始讀**。而 ft 變數是判斷現在是不是第一個輸入的 pattern。

接下來的 else if 是判斷現在 clk 來的時候，**不是 isstring 也不是 ispattern 且 pi 跟 si 也不是 0，則此時就是要比對 string 跟 pattern。先判斷是否為 head (^)。**

特別額外比較此特殊字元的原因 : 因為比較的方式是看是否 pattern(有 n 個字元都符合)，每次符合一筆，變數 find 就會加 1。當 find 的數量等於 pattern 的數量就代表比對相同成功。而 pattern 如果第一個是 head 則是一個條件，代表為 string 的開頭或是 string 內單字的開頭，而這些都不會被比對相同成功，但仍要算進 n 個字元的其中之一中。

```
//如果第一個字一樣 或 第一個字可為任意字元 才繼續比下一個
if((sc[4'b0001] == ss[i] || sc[4'b0001] == ".") && (i== 6'b000000 ||ss[i-4'b0001] == " "))
```

第一個組合電路 compare_head 為正緣

當比較是 head 後就開始比較裡面的字元。And 後面的條件是確保^是開頭或是一個單字最前面那個(意同於前一個字元為空白的)

而如果 find 為 0，就是在這之前都還沒有跟 pattern 其他字元相符時，就將目前的 i 紀錄到 first 裡面，之後當 pattern 跟 string 比對相同成功時 first 再存到 match_index 裡面去。

```
if(find == 4'b0000)
begin
        first = i;                  //紀錄起始位址
        find = find + 4'b0001;      //比對相同多一個find就加一
end
```

之後就是當首自元通過審合後的比對了。此時的就從當前的 index i 開始往後比(加上 pattern 的 index j)。之所以第二個迴圈 index j 是從 1 開始比，是因為 pattern 的首自元為^。且已經在 find == 4’b0000 的 if 判斷式 body 中 find + 1 了。所以就從 pattern 的第二個字元比較即可。如右圖。

此判斷式是判斷字元 sc[j] 是否個字串中的字元 ss[i+j-4’b0001]相同或是字元 sc[j] 是否是任意字元，如果是的話就 find 加一代表又找到一個相同的字元了。

```
if(sc[j] == ss[i+j-4'b0001] || sc[j] == ".")
```

如果 pattern 比對成功代表現在找到相同字元的數量等於 pattern 的數量了。(如右圖)match 拉成正緣，特別寫的是 i 與 j 都直接設超過兩個 for 迴圈的值，這是讓兩層 for 迴圈都能提早完成判斷。

```
//字串相符
if(find == pi)
begin
        match = 1'b1;
        match_index = first;
        j =  4'b0111;       //讓第二層迴圈停止
        i = 6'b100000;      //讓第一層迴圈停止
end
```

如果不是相同字元則在判斷是否為$ 代表最後一個字元。是最後一個字元且對應到 string 的空格，或是對應到 string 的 index 已經超過 string 大小也可，也就是 pattern 對應到 string 的末端。因為比到這裡代表已將整個 pattern 比較完了，所以直接將 match 提高，且兩層迴圈提早結束。(如右圖)

```
if(sc[j] == "$" && (ss[i+j-4'b0001] == " " || i+j-4'b0001 == si))
begin
        match = 1'b1;
        match_index = first;
        j =  4'b0111;       //讓第二層迴圈停止
        i = 6'b100000;      //讓第一層迴圈停止
end
```

如果比對過程有字元不相同，代表 pattern 的開頭無法對應到現在的 string 的 first 變數中存的 index。所以將第二層迴圈結束，且讓 i 回到原本的 first 值，第一層迴圈加一，繼續下一個回全，也就是 string 的下一個字元與 patter 的第二個字元做比較(第一個字元為^)

```
else
begin
        j =  4'b0111;       //讓第二層迴圈停止
        i = first;          //讓i回到原本的值
end
```

```
//如果第一個字元一樣才繼續比下一個 或是 第一個字元是任意字元
if(sc[0] == ss[i] || sc[0] == ".")
```

第二個組合電路: compare_normal 為正緣

如果不是開頭 pattern 的首字元就可能是.或是其他字元。判斷跟前面類似。但計數時不一樣，pattern 的字元樹樣不用額外處理。

完整的程式碼: (2 C 1 S)

SME.v - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
module SME(clk,reset,chardata,isstring,ispattern,valid,match,match_index);
        input clk;
        input reset;
        input [7:0] chardata;
        input isstring;
        input ispattern;
        output match;
        output [4:0] match_index;
        output valid;
        reg match;
        reg [4:0] match_index;
        reg [4:0] first;
        reg valid;
        reg [7:0] sc [7:0];
        reg [7:0] ss [31:0];
        reg [5:0] i;
        reg [3:0] j;
        reg [3:0] find;
        reg [4:0] pi;
        reg [5:0] si;
        reg [5:0] tmp;
        reg ft,compare_head,compare_normal;

        initial
        begin
                first <= 5'b00000;
                i <= 6'b000000;
                j <= 4'b0000;
                find <= 4'b0000;
                match <= 1'b0;
                match_index <= 5'b00000;
                valid <= 1'b0;
                pi <= 5'b00000;
                si <= 6'b000000;
                tmp <= 6'b000000;
                ft <= 1'b0;
                compare_head <=1'b0;
                compare_normal <= 1'b0;
        end
```

循序電路

```
always@(posedge clk)                        //sequential circuit
begin
        first = 5'b00000;
        match_index = 5'b00000;
        valid = 1'b0;
        match = 1'b0;
        compare_head =1'b0;
        compare_normal = 1'b0;
        if(reset == 1'b1)
        begin
                first = 5'b00000;
                i = 6'b000000;
                j = 4'b0000;
                find = 4'b0000;
                match = 5'b00000;
                match_index = 5'b00000;
                valid = 1'b0;
                pi = 5'b00000;
                compare_head =1'b0;
                compare_normal = 1'b0;
        end

        else
        begin

                if(ispattern == 1'b1) //讀輸入patter
                begin
                        sc[pi] = chardata;
                        pi = pi + 5'b00001;
                        if(ft == 1'b1)
                        begin
                                si = tmp;
                                tmp = 6'b000000;
                                ft = 1'b0;
                        end
                end

                else if(isstring == 1'b1) //讀輸入string
                begin
                        ss[tmp] = chardata;
                        tmp = tmp + 6'b000001;
                        if(ft == 1'b0)
                        begin
                                ft = 1'b1;
                                pi = 5'b00000;
                        end
                end

                else if(pi!=5'd0 && si!= 6'd0) //輸入完pattern 與 string
                begin
                        if(sc[0] == "^")
                                compare_head = 1'b1;
                        else
                                compare_normal = 1'b1;
                end
        end
end
```

組合電路

```
always@(posedge compare_head)                                          //combinational circuit
begin
        for(i = 6'b000000;i < si;i=i+6'b000001) //0-31 最32個bit
        begin

                //如果第一個字一樣 或 第一個字可為任意字元 才繼續比下一個
                if((sc[4'b0001] == ss[i] || sc[4'b0001] == ".") && (i== 6'b000000 ||ss[i-4'b0001] == " "))
                begin
                        if(find == 4'b0000)
                        begin
                                first = i;                //紀錄起始位址
                                find = find + 4'b0001;    //比對相同多一個find就加一
                        end
                        for(j = 4'b0001;j < pi;j=j+4'b0001)
                        begin
                                if(sc[j] == ss[i+j-4'b0001] || sc[j] == ".")            //如果相同
                                begin
                                        find = find + 4'b0001;
                                        //字串相符
                                        if(find == pi)
                                        begin
                                                match = 1'b1;
                                                match_index = first;
                                                j =  4'b0111;       //讓第二層迴圈停止
                                                i = 6'b100000;      //讓第一層迴圈停止
                                        end
                                end
                                else                                //如果不相同
                                begin
                                        if(sc[j] == "$" && (ss[i+j-4'b0001] == " " || i+j-4'b0001 == si))
                                        begin
                                                match = 1'b1;
                                                match_index = first;
                                                j =  4'b0111;       //讓第二層迴圈停止
                                                i = 6'b100000;      //讓第一層迴圈停止
                                        end
                                        else
                                        begin
                                                j =  4'b0111;       //讓第二層迴圈停止
                                                i = first;          //讓i回到原本的值
                                        end
                                end
                        end
                end
                find = 4'b0000;
        end
        valid = 1'b1;
        compare_head = 1'b0;
        pi = 5'b00000;
end
```

組合電路

```
        always@(posedge compare_normal)                     //combinational circuit
        begin

                for(i = 6'b000000;i < si;i=i+6'b000001) //0-31
                begin
                        //如果第一個字元一樣才繼續比下一個 或是 第一個字元是任意字元
                        if(sc[0] == ss[i] || sc[0] == ".")
                        begin
                                if(find == 4'b0000)
                                        first = i; //紀錄起始位址

                                for(j = 4'b0000;j < pi;j=j+4'b0001)
                                begin
                                        if(sc[j] == ss[i+j] || sc[j] == ".")          //如果相同
                                        begin
                                                find = find + 4'b0001;
                                                if(find == pi)      //字串相符
                                                begin
                                                        match = 1'b1;
                                                        match_index = first;
                                                        j =  4'b0111; //讓第二層迴圈停止
                                                        i = 6'b100000;//讓第一層迴圈停止
                                                end
                                                if(find == pi-5'b00001 && (sc[j+4'b0001] == "$" && i+j+4'b0001 == si))
                                                begin
                                                        match = 1'b1;
                                                        match_index = first;
                                                        j =  4'b0111; //讓第二層迴圈停止
                                                        i = 6'b100000;//讓第一層迴圈停止
                                                end
                                        end
                                        else                                    //如果不相同
                                        begin
                                                if(sc[j] == "$" && ss[i+j] == " ")
                                                begin
                                                        match = 1'b1;
                                                        match_index = first;
                                                        j =  4'b0111; //讓第二層迴圈停止
                                                        i = 6'b100000;//讓第一層迴圈停止
                                                end
                                                else
                                                begin
                                                        j =  4'b0111; //讓第二層迴圈停止
                                                        i = first;    //讓i回到原本的值
                                                end
                                        end
                                end
                        end
                        find = 4'b0000;
                end
                valid = 1'b1;
                pi = 5'b00000;
                compare_normal = 1'b0;
        end
endmodule
```

心得

(請寫下完成本次作業的心得、學到哪些東西、困難點的部分。)

這次作業中我在一開始沒有完全理解作業的內容,也因此剛開始寫的程式碼為組合電路,當 isstring 與 ispattern 為正緣的時候,就直接輸入 chardata 進二維陣列中。也因此測資一直跑到一半就停了。且顯示此錯誤訊息。

```
$display("-- Failed waiting valid signal, Simulation STOP --");
```

後來我才想通原來 isstring 與 ispattern 是當 clk 為正緣時才在裡面判斷,改了這部分後,此錯誤訊息就沒了,而是會正常顯示目前錯誤的比對有哪些。

在 debug 上述問題後,我的二維陣列在每次比對 pattern 後都忘記重設 index 為 0,讓下一次讀入後先放在 index 0 的位置,且還有陣列界線的問題,常常會不寫新寫超出陣列邊界,而造成 run 的時候 run 到一半就停下來了。一直想說是不是比對的哪個條件設錯了,後來才發現原來是界線問題。

這次作業困難點的部分我覺得是上述的 bug 最花時間找到,第一個問題是我還是不夠了解題目內容而直接反映到循序電路(clk 為正緣來時才動作)。第二個邊界問題也困擾很久。因為此次作業比對的條件太多了,每當發生 bug 我都第一時間認為設條件設錯,但其實上面兩個問題解決後,所有的測資都跑得出來,部會再 run 到一半就停止。而測資跑出來都會顯示是哪個比對發生問題,因此 debug 變得容易許多,雖然還是有許多條件需要思考如何設定,很多加一減一的問題,但每次修改完都可以直接看到哪個比對成功與失敗是很好的反饋。蠻喜歡這次 debug 的過程,我都會在紙上實際跑一次整個過程,讓各式 pattern 通過檢測!

而這次作業是正規表示式 Regular Expression 的作業,在電腦科學中很常進行搜索,因此完成此次作業後,我覺得這個作業寫出來還蠻有實用性的,即便過程中遇到許多困難,仍不失趣味性。