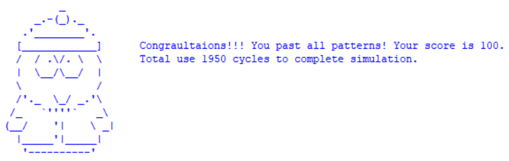
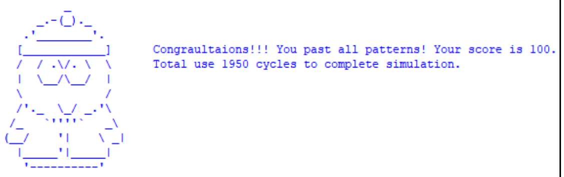
2023 Digital IC Design Homework 3

NAME	劉兆軒
Student ID	N26112437

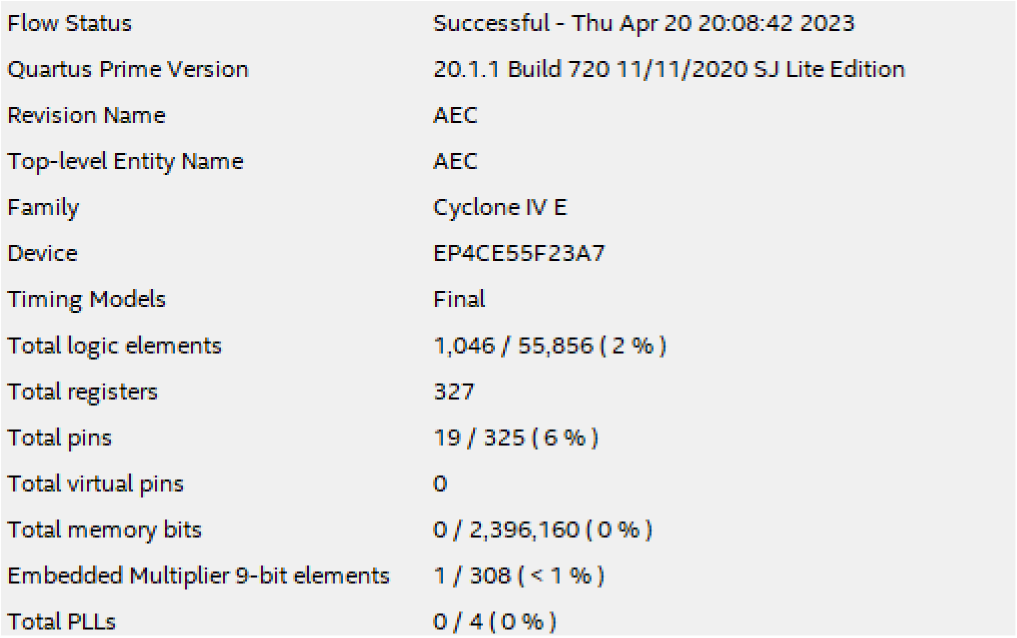
Simulation Result

Functional simulation	100	Gate-level simulation	100

 Congraultaions!!! You past all patterns! Your score is 100. Total use 1950 cycles to complete simulation.	 Congraultaions!!! You past all patterns! Your score is 100. Total use 1950 cycles to complete simulation.

Synthesis Result

Total logic elements	1046
Total memory bits	0
Embedded multiplier 9-bit elements	1
Total cycle used	1950
Clock width	20



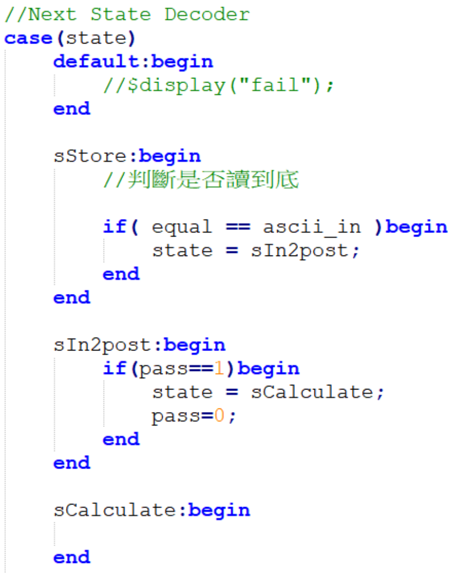
Flow Status	Successful - Thu Apr 20 20:08:42 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	AEC
Top-level Entity Name	AEC
Family	Cyclone IV E
Device	EP4CE55F23A7
Timing Models	Final
Total logic elements	1,046 / 55,856 (2 %)
Total registers	327
Total pins	19 / 325 (6 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	1 / 308 (< 1 %)
Total PLLs	0 / 4 (0 %)

分成三個 state：sStore >> sIn2post >> sCalculate

剛開始 store 階段先存每個 ascii_in 在 vec，之後再將該 vec 內的值轉成 postfix，轉完後再利用 stack 的方式將 postifx 計算出來。

Description of your design

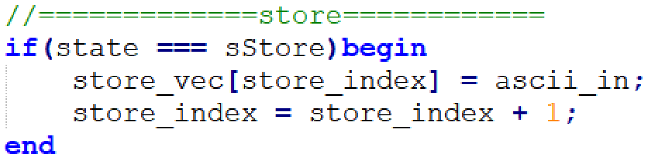
狀態轉換



```
//Next State Decoder
case(state)
    default:begin
        //$display("fail");
    end

    sStore:begin
        //判斷是否讀到底

        if( equal == ascii_in )begin
            state = sIn2post;
        end
    end

    sIn2post:begin
        if(pass==1)begin
            state = sCalculate;
            pass=0;
        end
    end

    sCalculate:begin

    end
```

sStore

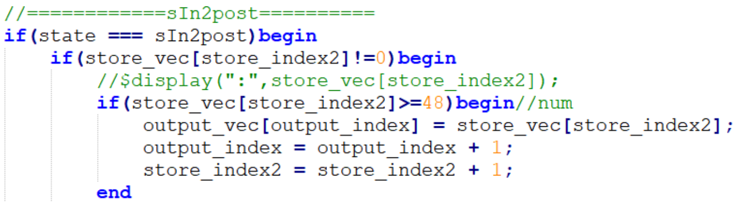
將值存到 store_vec 內



```
//============store============
if(state === sStore)begin
    store_vec[store_index] = ascii_in;
    store_index = store_index + 1;
end
```

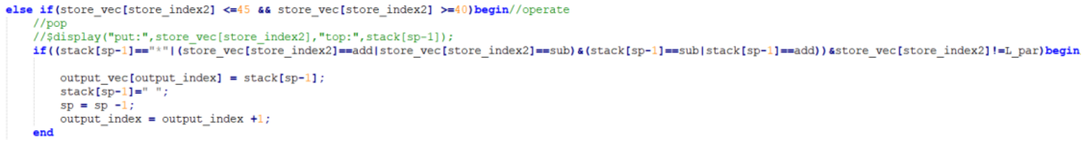
sIn2post

分成 2 部分，判斷現在指到的 store_vec 內容為何，屬於 num 則存到 vec，否則將 op push 到 stack 內，其中屬於 op 的話有幾點要判斷。



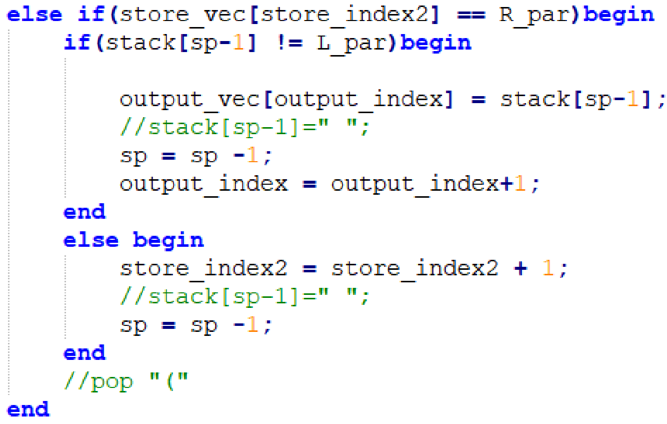
```
//============sIn2post==========
if(state === sIn2post)begin
    if(store_vec[store_index2]!=0)begin
        //$display(":",store_vec[store_index2]);
        if(store_vec[store_index2]>=48)begin//num
            output_vec[output_index] = store_vec[store_index2];
            output_index = output_index + 1;
            store_index2 = store_index2 + 1;
        end
```

1.要 push 之前，需檢查 stack 頂部 op 的 precedence 是否大於或等於要 psuh 進去的 op，若是的話則先將 stack 頂部的 op 不斷 pop 出來，直到頂部 op 的 precedence 小於要 push 進去的 op。



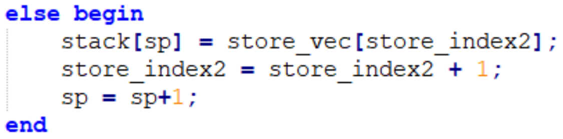
```
else if(store_vec[store_index2] <=45 && store_vec[store_index2] >=40)begin//operate
    //pop
    //$display("put:",store_vec[store_index2],"top:",stack[sp-1]);
    if((stack[sp-1]=="*"|(store_vec[store_index2]==add|store_vec[store_index2]==sub)&(stack[sp-1]==sub|stack[sp-1]==add))&store_vec[store_index2]!=L_par)begin

        output_vec[output_index] = stack[sp-1];
        stack[sp-1]=" ";
        sp = sp -1;
        output_index = output_index +1;
    end
```

2.若遇到要 push 進去的 op 是”)”，則將 stack pop 至出現”(“為止。



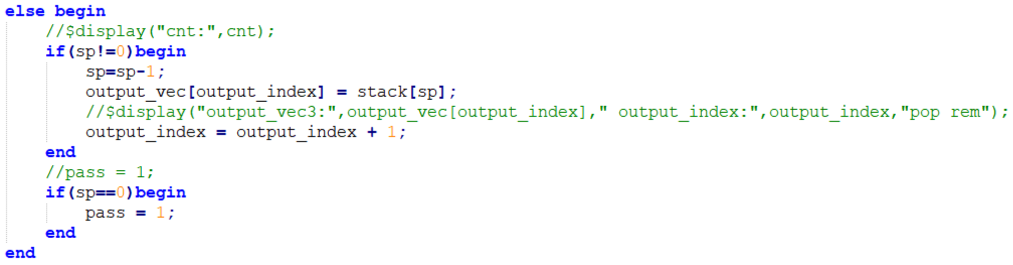
```
else if(store_vec[store_index2] == R_par)begin
    if(stack[sp-1] != L_par)begin

        output_vec[output_index] = stack[sp-1];
        //stack[sp-1]=" ";
        sp = sp -1;
        output_index = output_index+1;
    end
    else begin
        store_index2 = store_index2 + 1;
        //stack[sp-1]=" ";
        sp = sp -1;
    end
    //pop "("
end
```

3.其餘狀況 op 可以直接 push 進去 stack 內。



```
else begin
    stack[sp] = store_vec[store_index2];
    store_index2 = store_index2 + 1;
    sp = sp+1;
end
```

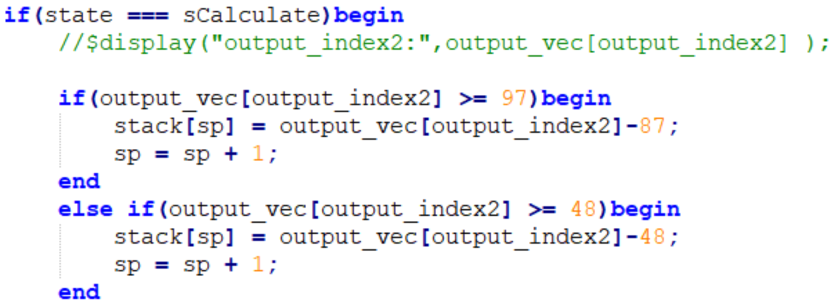
當 store_index2 走訪完 store_vec，則將 stack 內剩餘的 op，依序 push 至 vec 內，並轉換狀態進入下個階段。



```
else begin
    //$display("cnt:",cnt);
    if(sp!=0)begin
        sp=sp-1;
        output_vec[output_index] = stack[sp];
        //$display("output_vec3:",output_vec[output_index]," output_index:",output_index,"pop rem");
        output_index = output_index + 1;
    end
    //pass = 1;
    if(sp==0)begin
        pass = 1;
    end
end
```

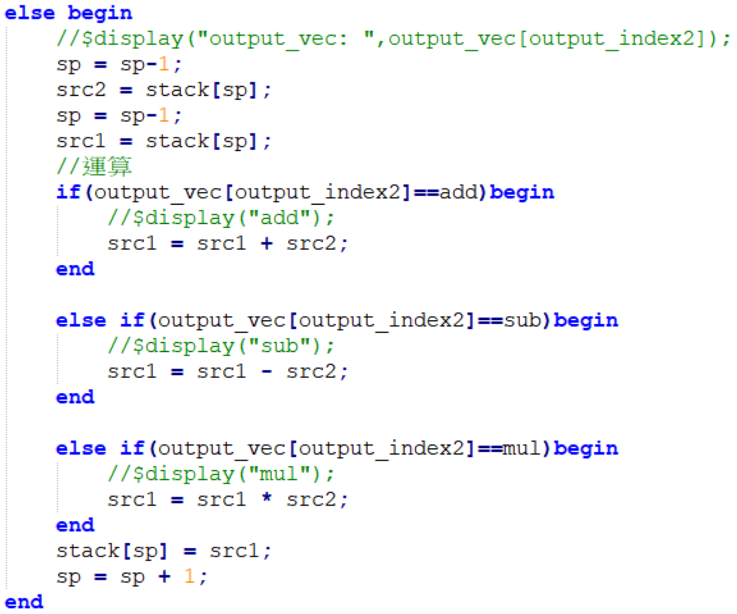
sCalculate

Step1:一樣先對 vec 做處理，若是數字則放到 stack 內



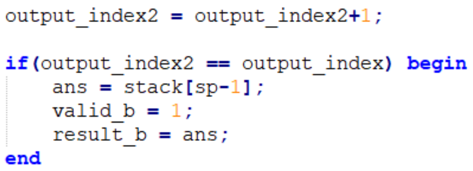
```
if(state === sCalculate)begin
    //$display("output_index2:",output_vec[output_index2] );

    if(output_vec[output_index2] >= 97)begin
        stack[sp] = output_vec[output_index2]-87;
        sp = sp + 1;
    end
    else if(output_vec[output_index2] >= 48)begin
        stack[sp] = output_vec[output_index2]-48;
        sp = sp + 1;
    end
```

Step2:Vec 目前的值屬於 op 的話則將 stack 內頂部存的 2 個 num pop 出來做計算，再 push 計算結果回去，不斷重複 step1~2。



```
else begin
    //$display("output_vec: ",output_vec[output_index2]);
    sp = sp-1;
    src2 = stack[sp];
    sp = sp-1;
    src1 = stack[sp];
    //運算
    if(output_vec[output_index2]==add)begin
        //$display("add");
        src1 = src1 + src2;
    end

    else if(output_vec[output_index2]==sub)begin
        //$display("sub");
        src1 = src1 - src2;
    end

    else if(output_vec[output_index2]==mul)begin
        //$display("mul");
        src1 = src1 * src2;
    end
    stack[sp] = src1;
    sp = sp + 1;
end
```

直到走訪完所有 vec 內的值，最後留在 stack 內的值便是最後運算的結果。



```
output_index2 = output_index2+1;

if(output_index2 == output_index) begin
    ans = stack[sp-1];
    valid_b = 1;
    result_b = ans;
end
```

*Scoring = Area cost * Timing cost*

*Area cost = Total logic elements + Total memory bits + 9*Embedded multipliers 9-bit elements*

*Timing cost = Total cycle used * Clock width*

*** Total logic elements must not exceed 1500.**