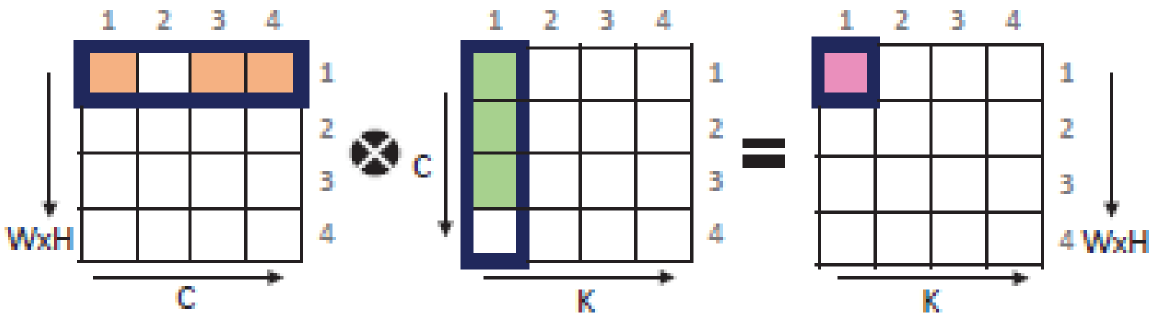
DCS Lab7
Matrix Multiplication

許凱捷

Discrete Cosine Transform

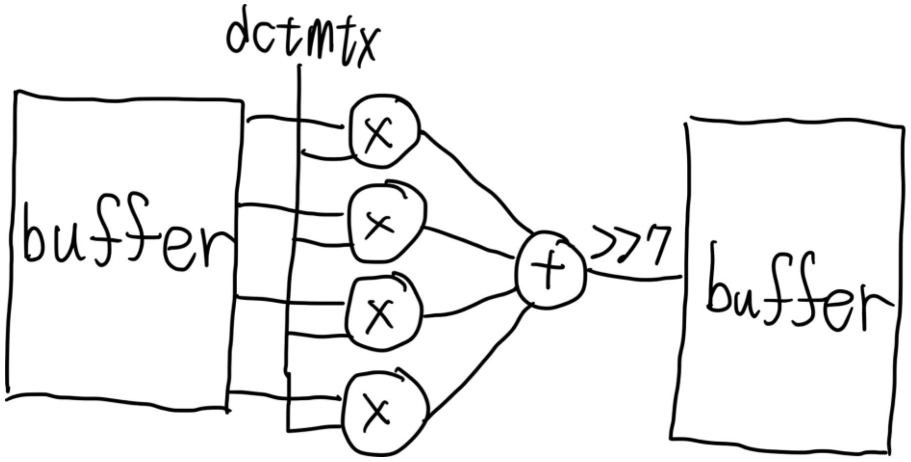
- 本次Lab需要設計計算DCT的電路
- dct = D A D^T
- 矩陣D已附在design內(dctmtx)，為fixed-point格式，由1-bit sign, 0-bit integer, 7-bit fraction組成，因此計算過程與整數相同，只需在每完成一次矩陣乘法後將結果除以128即可

```
>> D =dctmtx(4)

D =

    0.5000    0.5000    0.5000    0.5000
    0.6533    0.2706   -0.2706   -0.6533
    0.5000   -0.5000   -0.5000    0.5000
    0.2706   -0.6533    0.6533   -0.2706
```



Hint

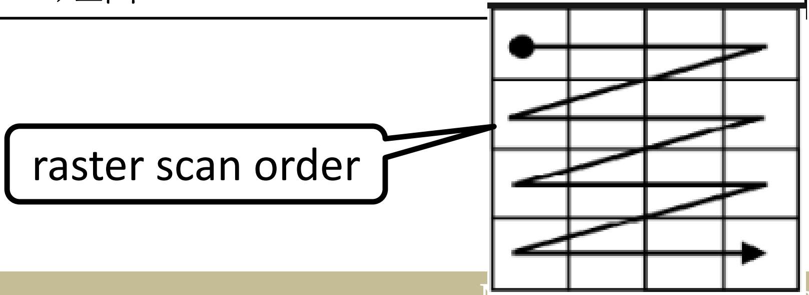
- 依序將input存入暫存器
- 使用4個乘法器，用內積的方式每cycle進行4個乘加法算出一個element，將結果除以128後(因為fixed-point乘法) 存入另一組暫存器，共16 cycle
- 經過兩輪矩陣乘法，將結果輸出



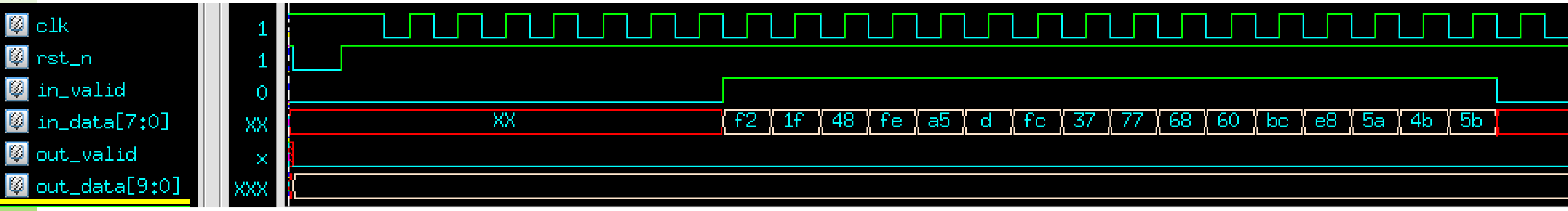
DCT.sv

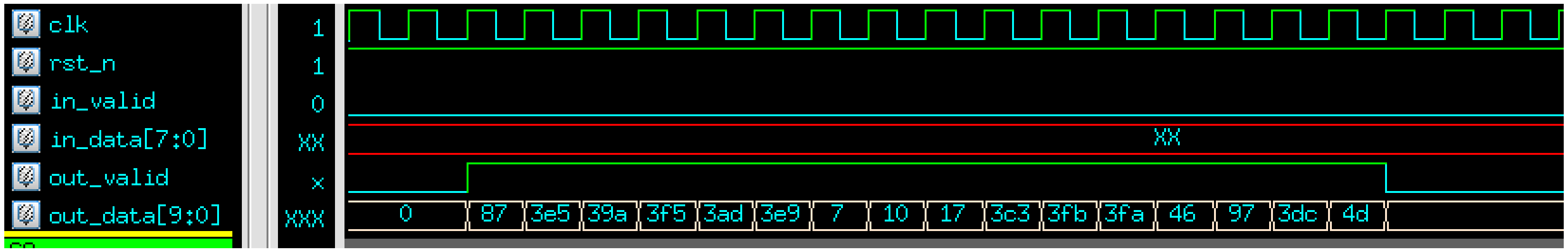
Input Signal	Bit Width	Definition
clk	1	10 ns Clock for 1 cycle
rst_n	1	Asynchronous reset when reset negedge, all output should be zero
in_valid	1	High for 16 cycle
in_data	8	raster scan order 輸入4x4矩陣

Output Signal	Bit Width	Definition
out_valid	1	High for 16 cycle
out_data	10	raster scan order 輸出input矩陣的DCT，為4x4矩陣



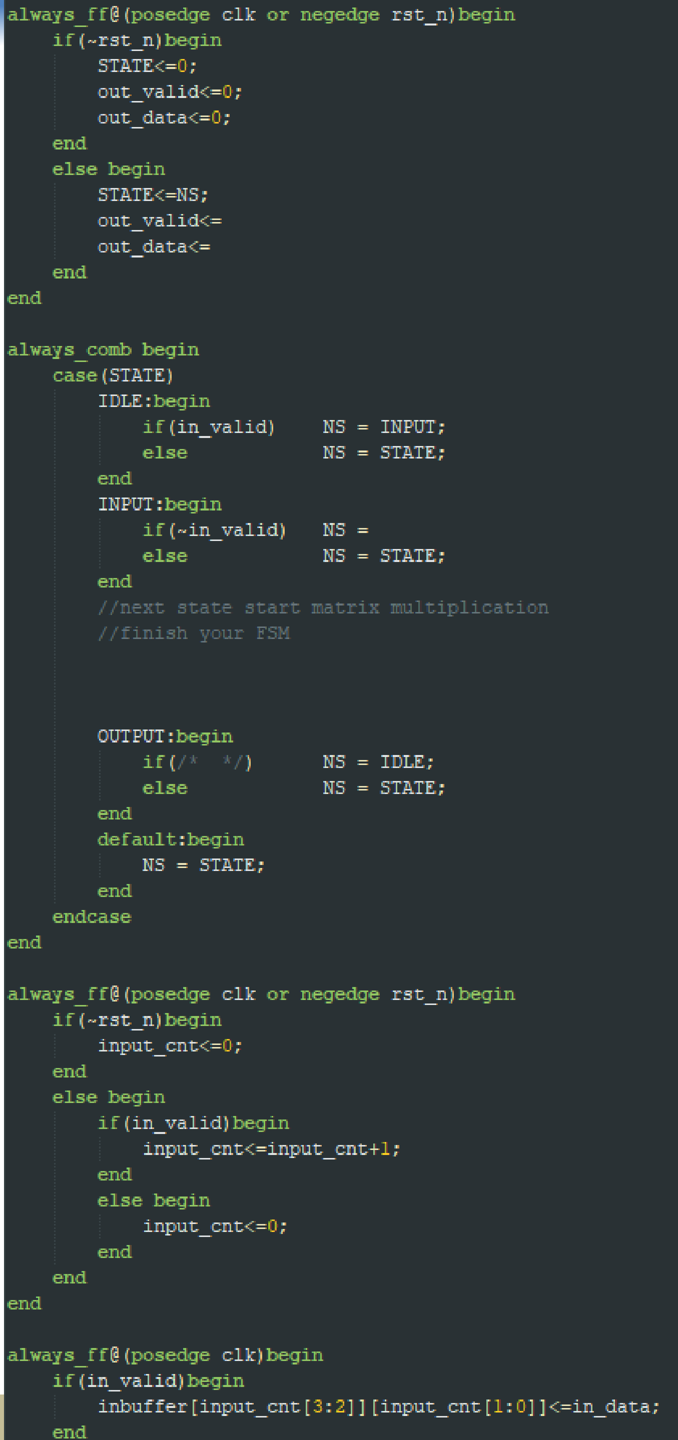
Waveform


clk
rst_n
in_valid
in_data[7:0]
out_valid
out_data[9:0]
1
1
0
XX
x
XXX
XX
f2
1f
48
fe
a5
d
fc
37
77
68
60
bc
e8
5a
4b
5b


clk
rst_n
in_valid
in_data[7:0]
out_valid
out_data[9:0]
1
1
0
XX
x
XXX
XX
0
87
3e5
39a
3f5
3ad
3e9
7
10
17
3c3
3fb
3fa
46
97
3dc
4d

Note

- 請留意乘加法為signed運算
- 注意乘法輸出bit數
- 本次Lab已附上部分FSM和將input存入inbuffer的code，各位同學可基於此繼續完成矩陣乘法部分即可。



```
always_ff@(posedge clk or negedge rst_n)begin
    if(~rst_n)begin
        STATE<=0;
        out_valid<=0;
        out_data<=0;
    end
    else begin
        STATE<=NS;
        out_valid<=
        out_data<=
    end
end

always_comb begin
    case(STATE)
        IDLE:begin
            if(in_valid)    NS = INPUT;
            else            NS = STATE;
        end
        INPUT:begin
            if(~in_valid)   NS =
            else            NS = STATE;
        end
        //next state start matrix multiplication
        //finish your FSM



        OUTPUT:begin
            if(/*  */)      NS = IDLE;
            else            NS = STATE;
        end
        default:begin
            NS = STATE;
        end
    endcase
end

always_ff@(posedge clk or negedge rst_n)begin
    if(~rst_n)begin
        input_cnt<=0;
    end
    else begin
        if(in_valid)begin
            input_cnt<=input_cnt+1;
        end
        else begin
            input_cnt<=0;
        end
    end
end

always_ff@(posedge clk)begin
    if(in_valid)begin
        inbuffer[input_cnt[3:2]][input_cnt[1:0]]<=in_data;
    end
```

Spec

- All outputs must have asynchronous negative-edge reset.
- 01_RTL needs to pass.
- 02_SYN should have no errors or latches.
- 02_SYN's timing slack must be met.
- 03_GATE needs to pass without timing violation.

Command

- tar -xvf ~dcsta01/Lab07.tar
- Upload
  - cd 09_upload
  - ./01_upload
  - ./02_download demoX

Demo1: 4/27(Thursday), 16:20:00

Demo2: 4/27(Thursday), 23:59:59