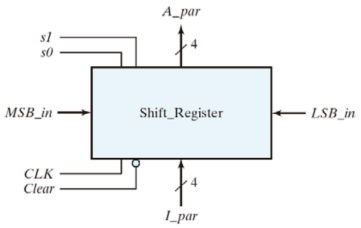
Verilog project5　　b10601002 廖品捷

Github: https://github.com/JamesLiao714/DLD-verilog

4-bit universal shift register

根據 S1 和 S2 的值來決定 output 的 function

當 clear 為 0 時將 A 清為 0



Clear	S1	S0	A3	A2	A1	A0	Operations
0	×	×	0	0	0	0	Clear
1	0	0	A3	A2	A1	A0	No change
1	0	1	MSBi	A3	A2	A1	Shift right
1	1	0	A2	A1	A0	LSBi	Shift left
1	1	1	I3	I2	I1	I0	Parallel load

Figure 1. The block diagram and function table of a 4-bit universal shift register.

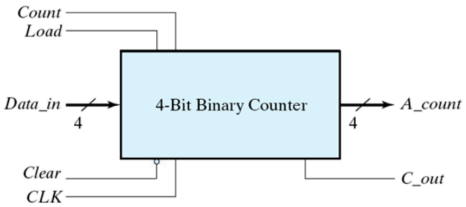
code:

```
//sr latch////
`timescale 1 ns/100ps
//USR

module USR(s0, s1, A, I, clear, clk, msb_in, lsb_in);

input s0, s1;
input clear;
input [3:0]I;
input clk;
output [3:0]A;
reg [3:0] A;
input msb_in, lsb_in;


always @(posedge clk or negedge clear)
begin
    if(clear == 1'b0)
        A <= 0;
    else if(s0 == 0 & s1 == 0) //no change
        A <= A;
    else if(s0 == 1 & s1 == 0)//sr
        A <= {msb_in, A[3:1]};
    else if(s0 == 0 & s1 == 1)//sl
        A <= {A[2:0], lsb_in};
    else if(s0 == 1 & s1 ==1) //parallel
        A <= I;
end
endmodule

```

4-bit binary counter

根據 load 和 count 的值來決定 output 的 function

當 clear 為 0 時 講 output 清為 0



Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Figure 2. The block diagram and function table of a 4-bit binary counter.

code:

```
78
79
80
81  module bc_4bit(cnt, ld, A, I, clear, clk);
82  input cnt, ld;
83  input clear;
84  input [3:0]I;
85  input clk;
86  output [3:0]A;
87  reg [3:0] A;
88
89
90  always @(posedge clk or negedge clear)
91  begin
92      if(clear == 1'b0)
93          A <= 0;
94      else if(ld == 1) //load
95          A <= I;
96      else if(ld == 0 & cnt == 1 & A == 4'b1111)//+1
97          A <= 0;
98      else if(ld == 0 & cnt == 1)//+1
99          A <= A + 1'b1;
00      else if(ld == 0 & cnt == 0)//no change
01          A <= A;
02
03  end
04  endmodule
```

test bench:

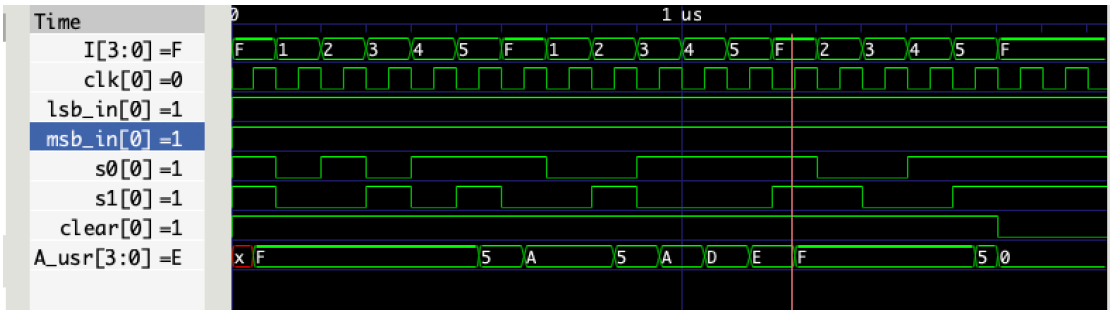
```
32  ////////////testbench///////////
33  module USR_TB();
34      reg s0, s1, clear, clk;
35      wire [3:0]A_usr;
36      wire [3:0]A_bc;
37      reg [3:0]I;
38      reg msb_in, lsb_in;
39      reg cnt, ld;
40      //
41      USR usr(s0, s1, A_usr, I, clear, clk, msb_in, lsb_in);
42      bc_4bit bc(cnt, ld, A_bc, I, clear, clk);
43      //
44
45  initial
46  begin
47      clk = 0;
48      repeat (50)
49      #50 clk = ~clk;
50  end
51  initial
52      begin
53          s0 = 1; s1 = 1; I = 4'b1111; clear = 1; msb_in = 1; lsb_in = 1; cnt= 0; ld = 0;
54          #100 s0 = 0; s1 = 0; I = 4'b0001; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 1;
55          #100 s0 = 1; s1 = 0; I = 4'b0010; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
56          #100 s0 = 0; s1 = 1; I = 4'b0011; clear = 1; msb_in = 1; lsb_in = 1;cnt = 1; ld = 0;
57          #100 s0 = 1; s1 = 0; I = 4'b0100; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 0;
58          #100 s0 = 1; s1 = 1; I = 4'b0101; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
59          #100 s0 = 1; s1 = 0; I = 4'b1111; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
60          #100 s0 = 0; s1 = 0; I = 4'b0001; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 0;
61          #100 s0 = 0; s1 = 1; I = 4'b0010; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
62          #100 s0 = 1; s1 = 0; I = 4'b0011; clear = 1; msb_in = 1; lsb_in = 1;cnt = 1; ld = 0;
63          #100 s0 = 1; s1 = 0; I = 4'b0100; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 0;
64          #100 s0 = 1; s1 = 0; I = 4'b0101; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
65          #100 s0 = 1; s1 = 1; I = 4'b1111; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
66          #100 s0 = 0; s1 = 1; I = 4'b0010; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
67          #100 s0 = 0; s1 = 0; I = 4'b0011; clear = 1; msb_in = 1; lsb_in = 1;cnt = 1; ld = 0;
68          #100 s0 = 1; s1 = 0; I = 4'b0100; clear = 1;msb_in = 1; lsb_in = 1; cnt = 0; ld = 1;
69          #100 s0 = 1; s1 = 1; I = 4'b0101; clear = 1;msb_in = 1; lsb_in = 1; cnt = 0; ld = 1;
70          #100 s0 = 1; s1 = 1; I = 4'b1111; clear = 0;msb_in = 1; lsb_in = 1; cnt = 0; ld =0;
71      end
72  initial #2000 $finish;
73  initial $dumpvars;
74  endmodule
```

waveform:

實作模擬如下圖

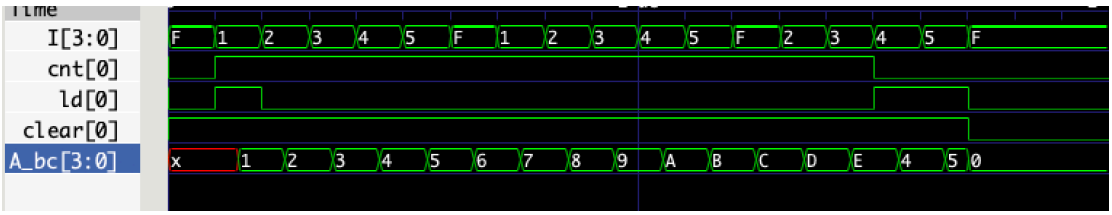
1. 4-bit universal shift register

   A_usr 為根據 s0, s1 所產生的 output(shift…)



2. 4-bit binary counter

   A_bc 為根據 ld 和 cnt 所產生的 ouput(計數, load l)


Time
I[3:0]
cnt[0]
ld[0]
clear[0]
A_bc[3:0]
F 1 2 3 4 5 F 1 2 3 4 5 F 2 3 4 5 F
x 1 2 3 4 5 6 7 8 9 A B C D E 4 5 0