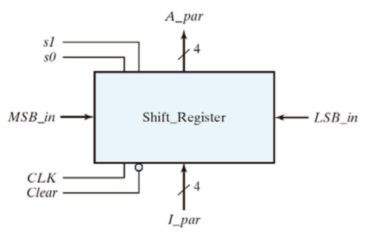
Verilog project4 b10601002 廖品捷

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

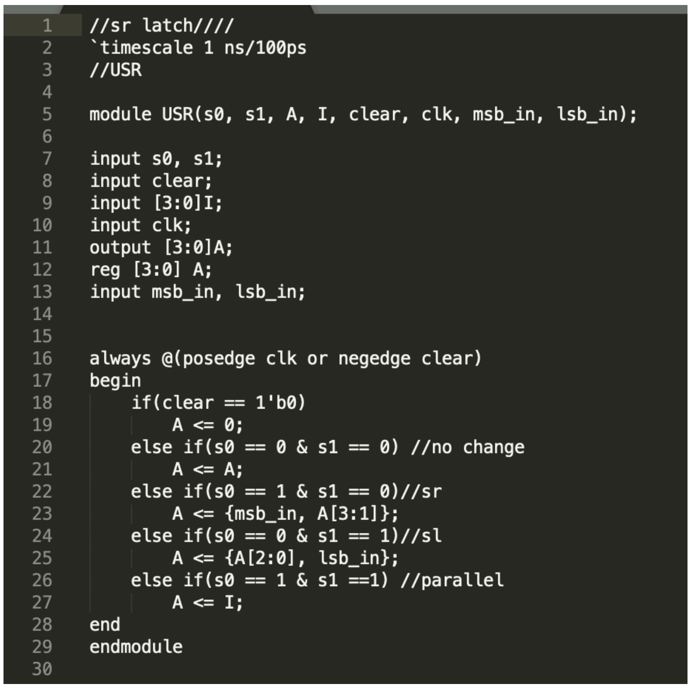
4-bit universal shift register



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

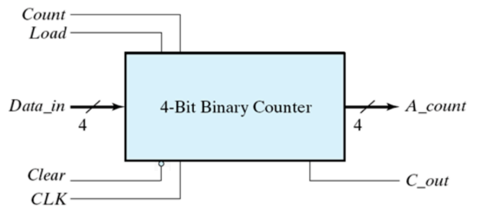
Figure 1. The block diagram and function table of a 4-bit universal shift register.

code:



```verilog
//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

```

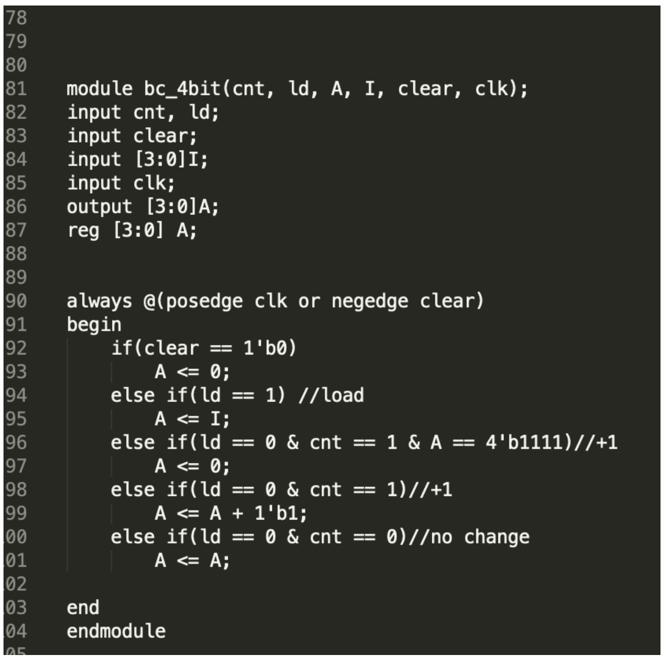
4-bit binary counter



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

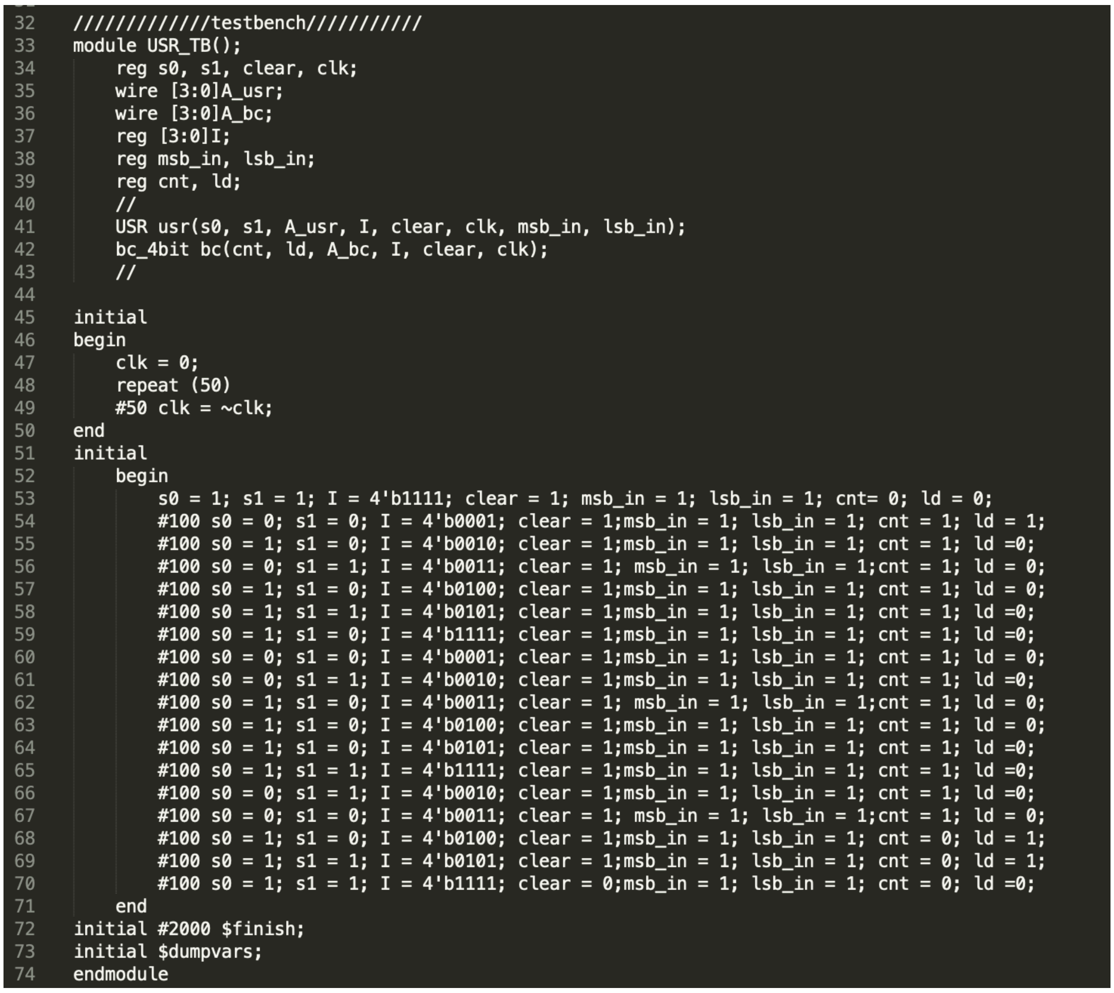
Figure 2. The block diagram and function table of a 4-bit binary counter.

code:



```verilog



module bc_4bit(cnt, ld, A, I, clear, clk);
input cnt, ld;
input clear;
input [3:0]I;
input clk;
output [3:0]A;
reg [3:0] A;


always @(posedge clk or negedge clear)
begin
    if(clear == 1'b0)
        A <= 0;
    else if(ld == 1) //load
        A <= I;
    else if(ld == 0 & cnt == 1 & A == 4'b1111)//+1
        A <= 0;
    else if(ld == 0 & cnt == 1)//+1
        A <= A + 1'b1;
    else if(ld == 0 & cnt == 0)//no change
        A <= A;

end
endmodule
```

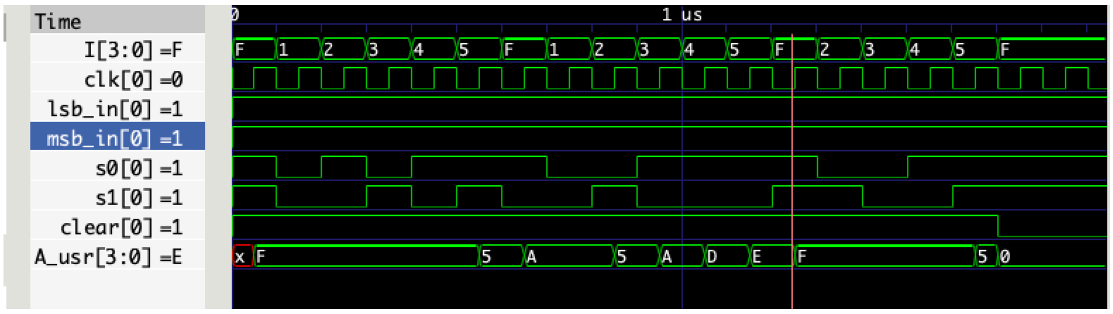
test bench:



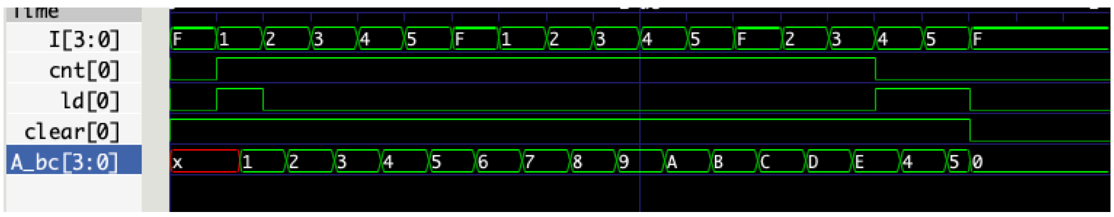
```verilog
/////////////testbench//////////
module USR_TB();
    reg s0, s1, clear, clk;
    wire [3:0]A_usr;
    wire [3:0]A_bc;
    reg [3:0]I;
    reg msb_in, lsb_in;
    reg cnt, ld;
    //
    USR usr(s0, s1, A_usr, I, clear, clk, msb_in, lsb_in);
    bc_4bit bc(cnt, ld, A_bc, I, clear, clk);
    //

initial
begin
    clk = 0;
    repeat (50)
    #50 clk = ~clk;
end
initial
    begin
        s0 = 1; s1 = 1; I = 4'b1111; clear = 1; msb_in = 1; lsb_in = 1; cnt= 0; ld = 0;
        #100 s0 = 0; s1 = 0; I = 4'b0001; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 1;
        #100 s0 = 1; s1 = 0; I = 4'b0010; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 0; s1 = 1; I = 4'b0011; clear = 1; msb_in = 1; lsb_in = 1;cnt = 1; ld = 0;
        #100 s0 = 1; s1 = 0; I = 4'b0100; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 0;
        #100 s0 = 1; s1 = 1; I = 4'b0101; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 1; s1 = 0; I = 4'b1111; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 0; s1 = 0; I = 4'b0001; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 0;
        #100 s0 = 0; s1 = 1; I = 4'b0010; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 1; s1 = 0; I = 4'b0011; clear = 1; msb_in = 1; lsb_in = 1;cnt = 1; ld = 0;
        #100 s0 = 1; s1 = 0; I = 4'b0100; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld = 0;
        #100 s0 = 1; s1 = 0; I = 4'b0101; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 1; s1 = 1; I = 4'b1111; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 0; s1 = 1; I = 4'b0010; clear = 1;msb_in = 1; lsb_in = 1; cnt = 1; ld =0;
        #100 s0 = 0; s1 = 0; I = 4'b0011; clear = 1; msb_in = 1; lsb_in = 1;cnt = 1; ld = 0;
        #100 s0 = 1; s1 = 0; I = 4'b0100; clear = 1;msb_in = 1; lsb_in = 1; cnt = 0; ld = 1;
        #100 s0 = 1; s1 = 1; I = 4'b0101; clear = 1;msb_in = 1; lsb_in = 1; cnt = 0; ld = 1;
        #100 s0 = 1; s1 = 1; I = 4'b1111; clear = 0;msb_in = 1; lsb_in = 1; cnt = 0; ld =0;
    end
initial #2000 $finish;
initial $dumpvars;
endmodule
```

waveform:

1. 4-bit universal shift register



2. 4-bit binary counter


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