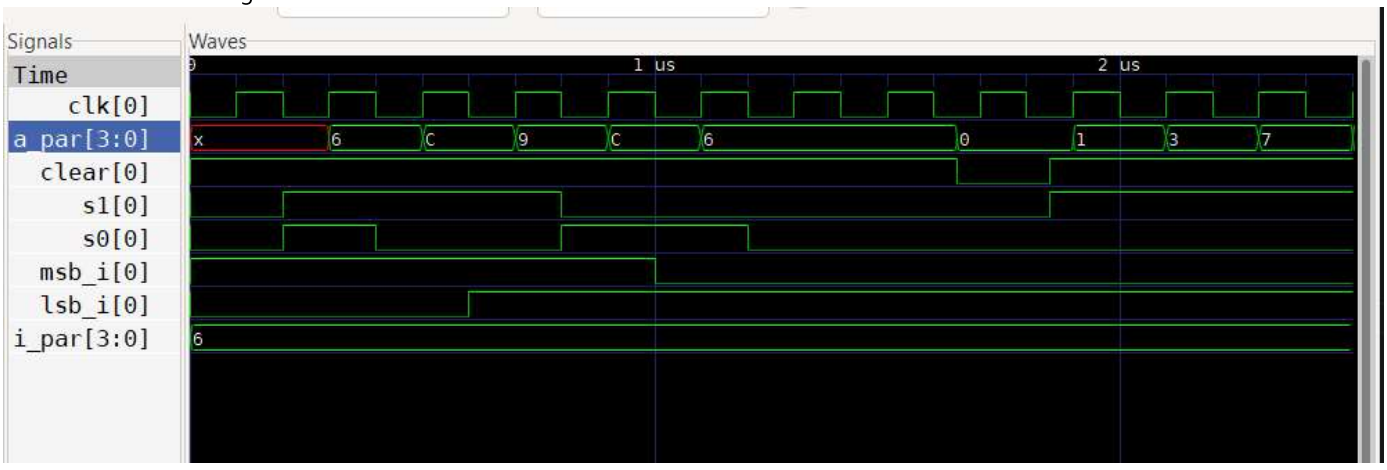
DDLab project 6

```
`timescale 1ns/100ps

module shift_4 (
    output reg [3:0] a_par,
    input wire [3:0] i_par,
    input wire s1,s0,msb_i,clk,clear,lsb_i
);

    always @(posedge clk or negedge clear ) begin
        if(~clear)
            begin
                a_par <= 4'b0;
            end
        else
            begin
                case ({s1,s0})
                    2'b01:
                        begin
                            a_par[2:0] <= a_par[3:1];
                            a_par[3] <= msb_i;
                        end
                    2'b10:
                        begin
                            a_par[3:1] <= a_par[2:0];
                            a_par[0] <= lsb_i;
                        end
                    2'b11:
                        begin
                            a_par <= i_par;
                        end
                    default: a_par <= a_par;
                endcase
            end
    end
endmodule

module t_shift_4 (

);
    reg s1,s0,msb_i,clk,clear,lsb_i;
    reg [3:0] i_par;
    wire [3:0] a_par;

    shift_4 m(a_par,i_par,s1,s0,msb_i,clk,clear,lsb_i);

    //clk
    initial begin
        clk=1'b0;
        repeat(100)
            #100 clk = ~clk;
    end

    initial begin
        //init
        i_par = 4'b0110;
        s1 = 1'b0;
```

```
        s0 = 1'b0;
        msb_i = 1'b1;
        lsb_i = 1'b0;
        clear = 1'b1;

        //load
        #200 s1=1'b1; s0=1'b1;

        //shift left
        #200 s1=1'b1; s0=1'b0;
        #200 lsb_i = ~lsb_i;

        //shift right
        #200 s1=1'b0; s0=1'b1;
        #200 msb_i = ~msb_i;

        //no change
        #200 s1=1'b0; s0=1'b0;
        #200;

        //test clear
        #250 clear = 1'b0;
        #200 clear = 1'b1;
        s1=1'b1; s0=1'b0;

    end

    initial $dumpvars;
    // initial #2500 $finish;

endmodule

module counter_4 (
    output reg [3:0] a_count,
    output reg c_out,
    input wire [3:0] data_in,
    input wire count, load, clear, clk
);
    always @(posedge clk or negedge clear) begin
        if(~clear) begin
            a_count <= 4'b0;
            c_out <= 1'b0;
        end
        else if (load) begin
            a_count <= data_in;
            c_out <= 1'b0;
        end
        else if (count) begin
            c_out <= &a_count;
            a_count <= a_count + 1'b1;
        end
    end
endmodule

module t_counter_4 (

);
```

```
    wire [3:0] a_count;
    wire c_out;
    reg [3:0] data_in;
    reg count, load, clear, clk;

    counter_4 m(a_count,c_out,data_in,count, load, clear, clk);

    //clk
    initial begin
        clk=1'b0;
        repeat(100)
            #100 clk = ~clk;
    end

    initial begin
        data_in = 4'hf;
        count = 1'b0;
        load = 1'b0;
        clear = 1'b1;

        #50 clear = 1'b0;
        #200 clear = 1'b1;
        #200 count = 1'b1;
        #600 load = 1'b1;
        #200 load = 1'b0;
    end

    initial $dumpvars;
    initial #2000 $finish;

endmodule
```

4-bit universal shift register



4-bit binary counter