```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity dff is
	port(	D: in STD_LOGIC;
		Q: out STD_LOGIC;
		Clock: in STD_LOGIC;
		Clear: in STD_LOGIC);
end dff;
architecture Behavioral of dff is
begin
	process(Clock) --eğer clock değişirse alttaki begin-end process i çalıştır --
	begin
		if(rising_edge(Clock)) then
			if(Clear='0') then
				Q <= '0';
				--Qn <= '1';
			else
				Q <= D;
				--Qn <= not(D);
			end if;
		end if;
	end process;
end Behavioral;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux4x1 is
		port( A,B,C,D : in STD_LOGIC;
			S0,S1: in STD_LOGIC;
			Z: out STD_LOGIC );
```

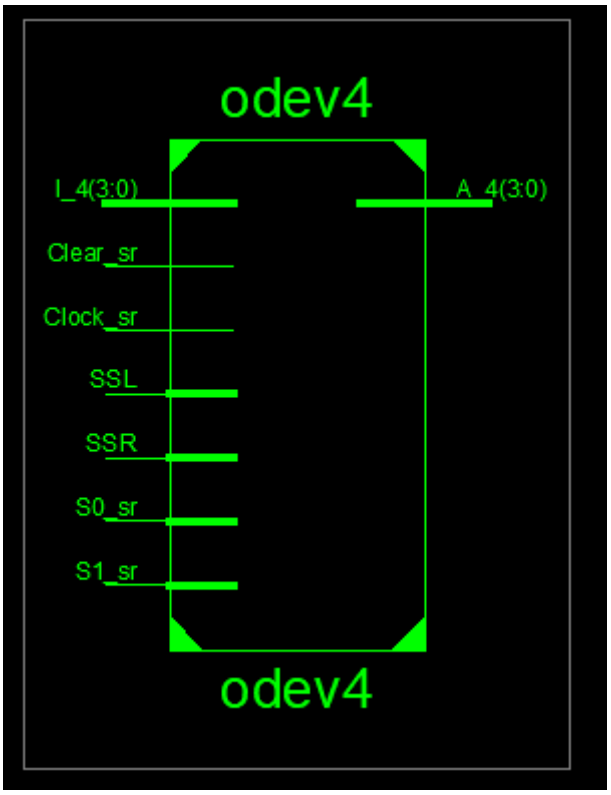
```
end mux4x1;

architecture Behavioral of mux4x1 is

begin
process(A,B,C,D,S0,S1) is
        begin
                if(S1='0' and S0='0') then
                        Z<=A;
                elsif(S1='0' and S0='1') then
                        Z<=B;
                elsif(S1='1' and S0='0') then
                        Z<=C;
                else
                        Z<=D;
                end if;
end process;
end Behavioral;
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dff_and_mux is
                port(   dff_mux_4: in STD_LOGIC_VECTOR (3 downto 0);
                        serial_shift_2: in STD_LOGIC_VECTOR (1 downto 0);
                        Clk: in STD_LOGIC;
                        Clr: in STD_LOGIC;
                        Q_out: out STD_LOGIC);
end dff_and_mux;

architecture Behavioral of dff_and_mux is
```

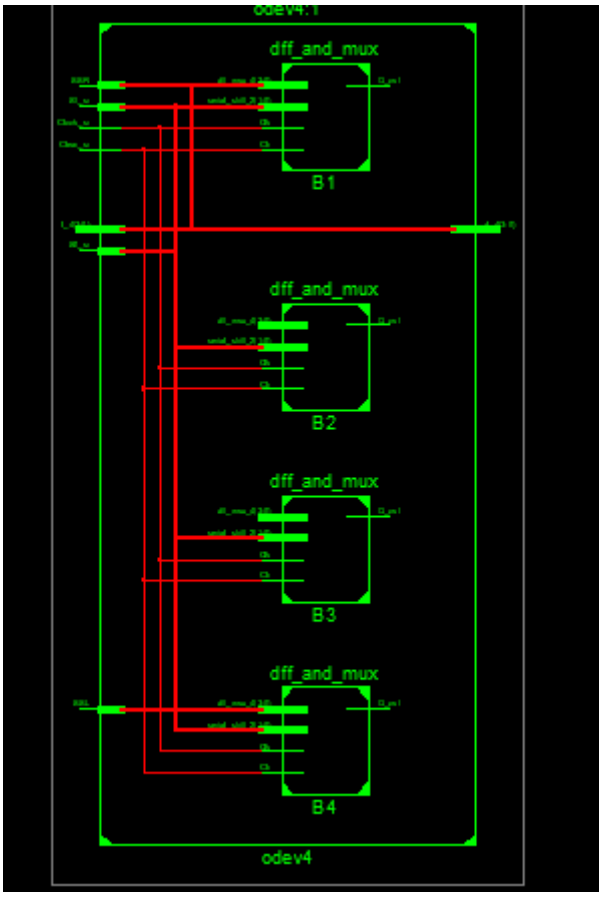
```
        component mux4x1 is
                port( A,B,C,D : in STD_LOGIC;
                        S0,S1: in STD_LOGIC;
                        Z: out STD_LOGIC );
        end component;

        component dff is
                port(   Q: out STD_LOGIC;
                        --Qn: out STD_LOGIC;
                        Clock: in STD_LOGIC;
                        Clear: in STD_LOGIC;
                        D: in STD_LOGIC);
        end component;

        signal mux_to_dff: STD_LOGIC;
begin
                B1: mux4x1 port map( A=>dff_mux_4(0),
                                     B=>dff_mux_4(1),
                                     C=>dff_mux_4(2),
                                     D=>dff_mux_4(3),
                                     S0=>serial_shift_2(0),
                                     S1=>serial_shift_2(1),
                                     Z=>mux_to_dff);

                B2: dff port map( Q=>Q_out,
                                  Clock=>Clk,
                                  Clear=>Clr,
                                  D=>mux_to_dff);
end Behavioral;
--------------------------------------------------------------------------------
```

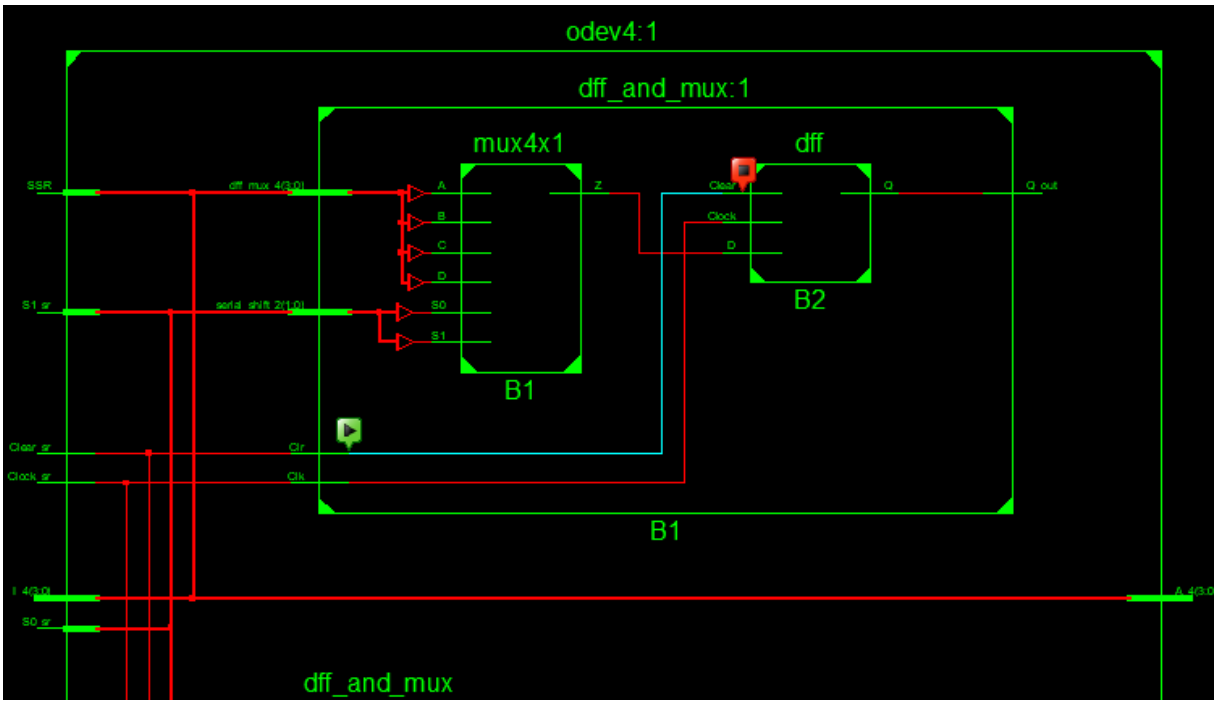
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity odev4 is
	port( I_4: in STD_LOGIC_VECTOR (3 downto 0);
		SSR: in STD_LOGIC; --serial input for shift right
		SSL: in STD_LOGIC; --serial input for shift left
		S1_sr: in STD_LOGIC;
		S0_sr: in STD_LOGIC;
		Clock_sr: in STD_LOGIC;
		Clear_sr: in STD_LOGIC;
		A_4: out STD_LOGIC_VECTOR (3 downto 0));
end odev4;

architecture Behavioral of odev4 is

	component dff_and_mux is
		port(	dff_mux_4: in STD_LOGIC_VECTOR (3 downto 0);
			serial_shift_2: in STD_LOGIC_VECTOR (1 downto 0);
			Clk: in STD_LOGIC;
			Clr: in STD_LOGIC;
			Q_out: out STD_LOGIC);
	end component;

	signal store_4: STD_LOGIC_VECTOR (3 downto 0);
	signal shift_right: STD_LOGIC_VECTOR (3 downto 0);
	signal shift_left: STD_LOGIC_VECTOR (2 downto 0);

begin

	B1: dff_and_mux port map(dff_mux_4(0)=>store_4(3),
```

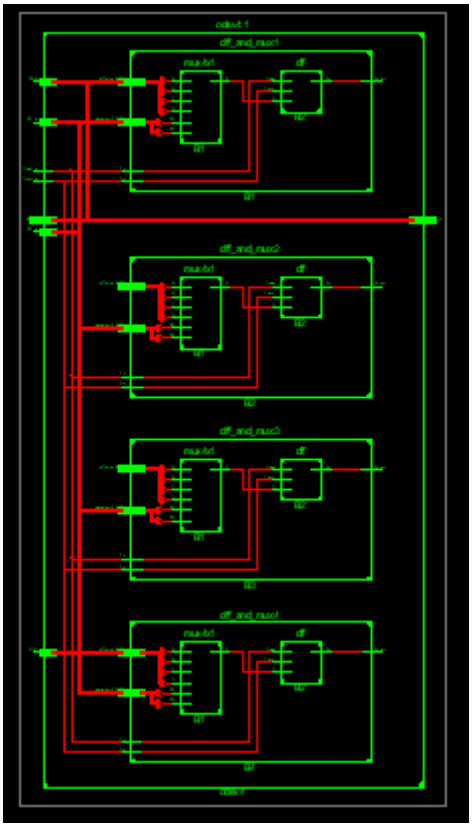
```
                                    dff_mux_4(1)=>SSR,
                                    dff_mux_4(2)=>shift_left(2),
                                    dff_mux_4(3)=>I_4(3),
                                    serial_shift_2(0)=>S0_sr,
                                    serial_shift_2(0)=>S1_sr,
                                    Clk=>Clock_sr,
                                    Clr=>Clear_sr,
                                    Q_out=>shift_right(3));

B2: dff_and_mux port map(dff_mux_4(0)=>store_4(2),
                                    dff_mux_4(1)=>shift_right(3),
                                    dff_mux_4(2)=>shift_left(1),
                                    dff_mux_4(3)=>I_4(2),
                                    serial_shift_2(0)=>S0_sr,
                                    serial_shift_2(1)=>S1_sr,
                                    Clk=>Clock_sr,
                                    Clr=>Clear_sr,
                                    Q_out=>shift_right(2));

B3: dff_and_mux port map(dff_mux_4(0)=>store_4(1),
                                    dff_mux_4(1)=>shift_right(2),
                                    dff_mux_4(2)=>shift_left(0),
                                    dff_mux_4(3)=>I_4(1),
                                    serial_shift_2(0)=>S0_sr,
                                    serial_shift_2(1)=>S1_sr,
                                    Clk=>Clock_sr,
                                    Clr=>Clear_sr,
                                    Q_out=>shift_right(1));
```

```
B4: dff_and_mux port map(dff_mux_4(0)=>store_4(0),
                         dff_mux_4(1)=>shift_right(1),
                         dff_mux_4(2)=>SSL,
                         dff_mux_4(3)=>I_4(0),
                         serial_shift_2(0)=>S0_sr,
                         serial_shift_2(1)=>S1_sr,
                         Clk=>Clock_sr,
                         Clr=>Clear_sr,
                         Q_out=>shift_right(0));

A_4(3)<=shift_right(3);
store_4(3)<=shift_right(3);

A_4(2)<=shift_right(2);
store_4(2)<=shift_right(2);
shift_left(2)<=shift_right(2);

A_4(1)<=shift_right(1);
store_4(1)<=shift_right(1);
shift_left(1)<=shift_right(1);

A_4(0)<=shift_right(0);
store_4(0)<=shift_right(0);
shift_left(0)<=shift_right(0);


end Behavioral;
```

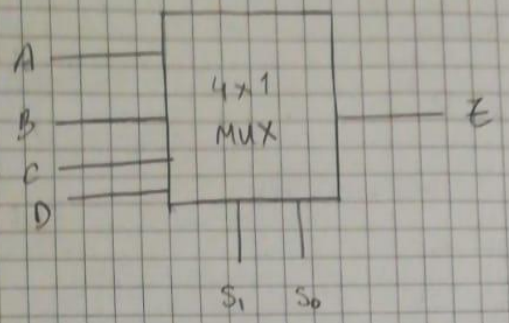

odev4
I_4(3:0)
A_4(3:0)
Clear_sr
Clock_sr
SSL
SSR
S0_sr
S1_sr
odev4


dff_and_mux
B1
dff_and_mux
B2
dff_and_mux
B3
dff_and_mux
B4
odev4


odev4:1
dff_and_mux:1
mux4x1
dff
SSR
dff_mux_4(3:0)
A
B
C
D
Z
Clear
Clock
D
Q
Q_out
S1_sr
serial_shift_2(1:0)
S0
S1
B1
B2
Clear_sr
Clock_sr
Clr
Clk
B1
I_4(3:0)
A_4(3:0)
S0_sr
dff_and_mux



shift right

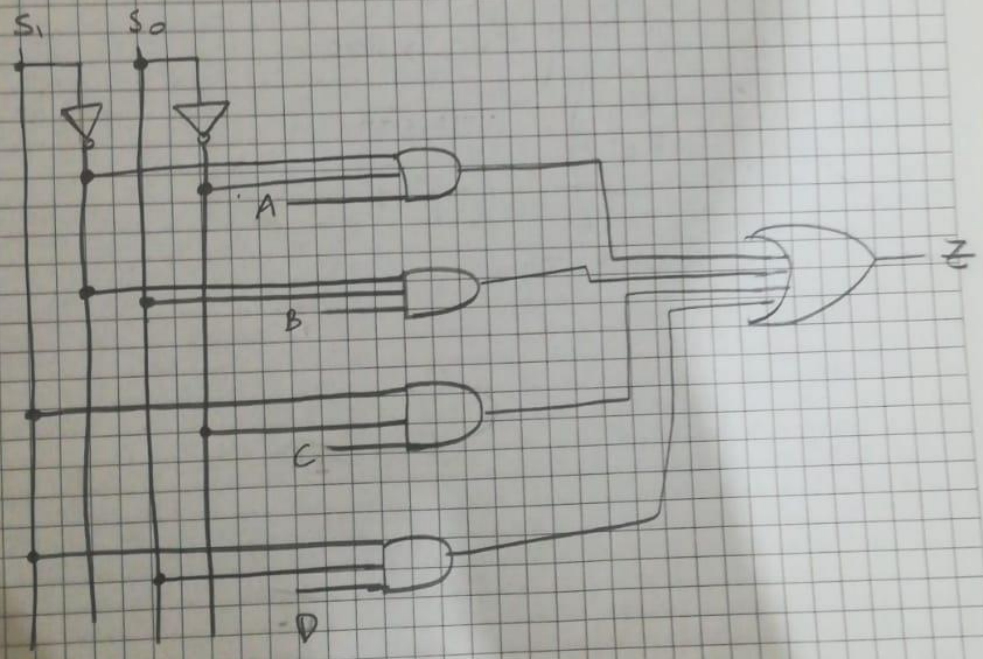


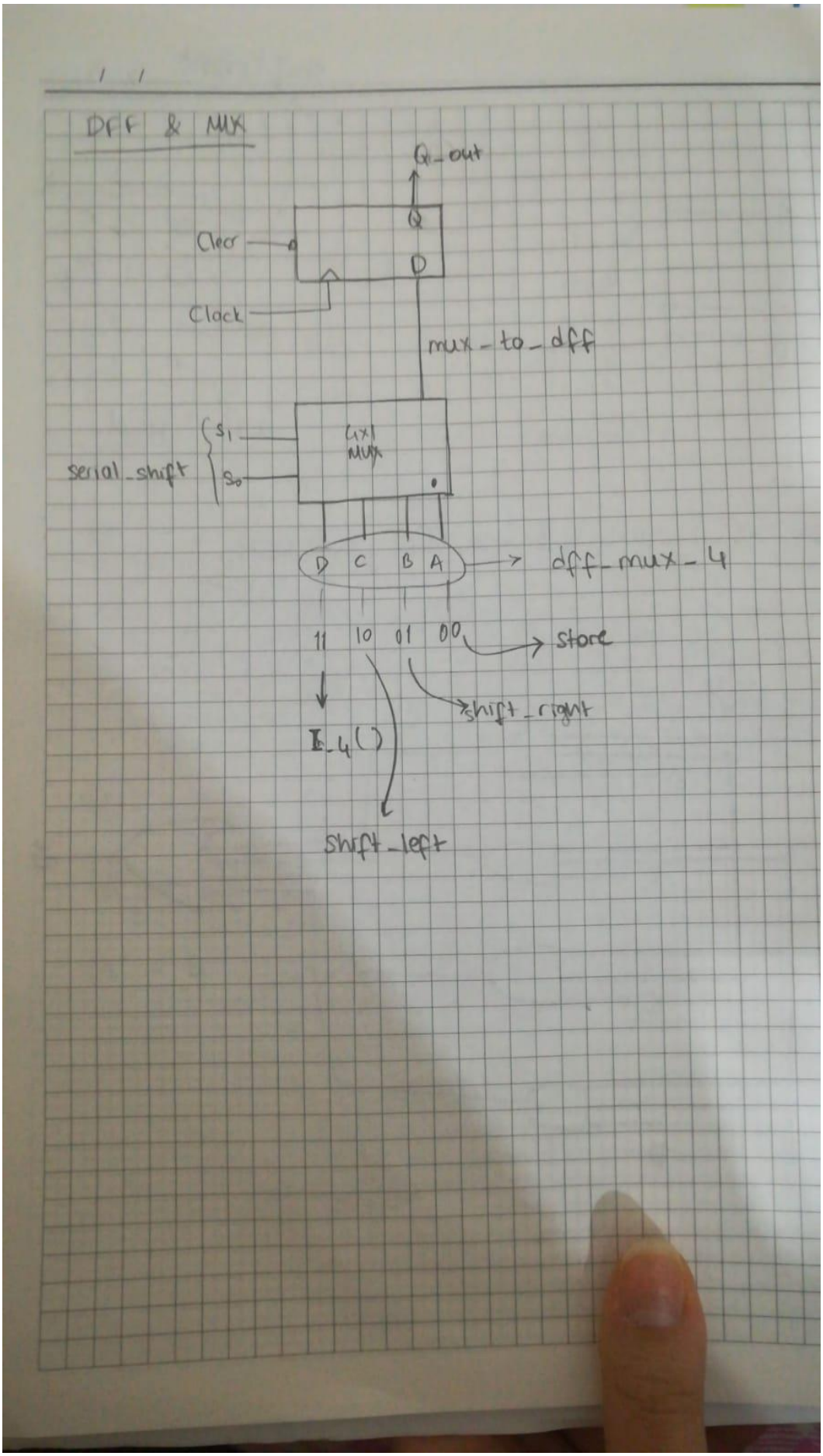
S_1	S_0	Z
0	0	A
0	1	B
1	0	C
1	1	D

shift left

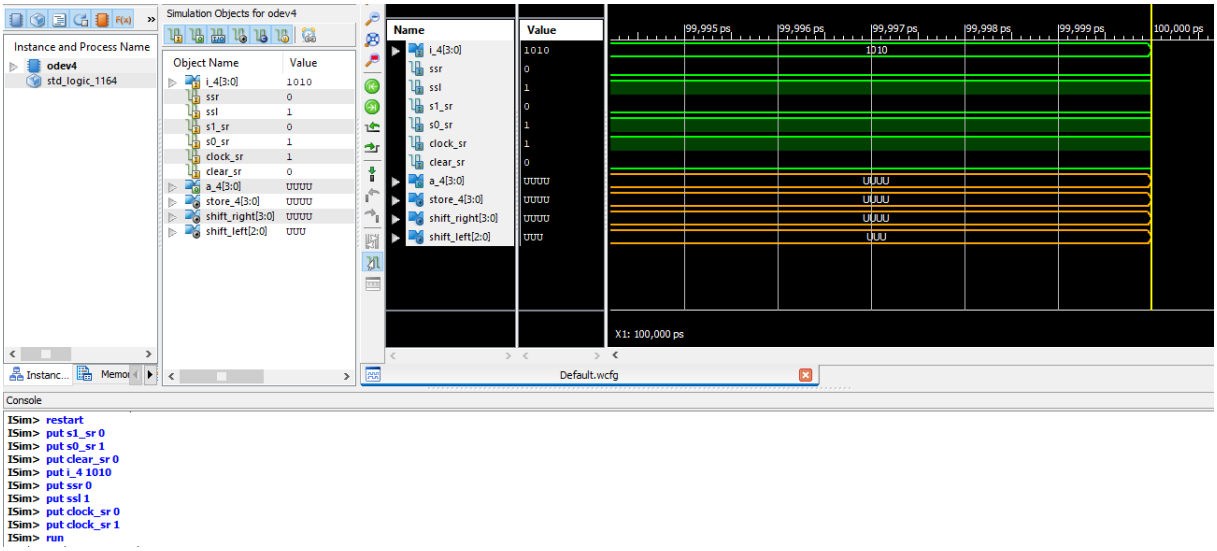
parallel load

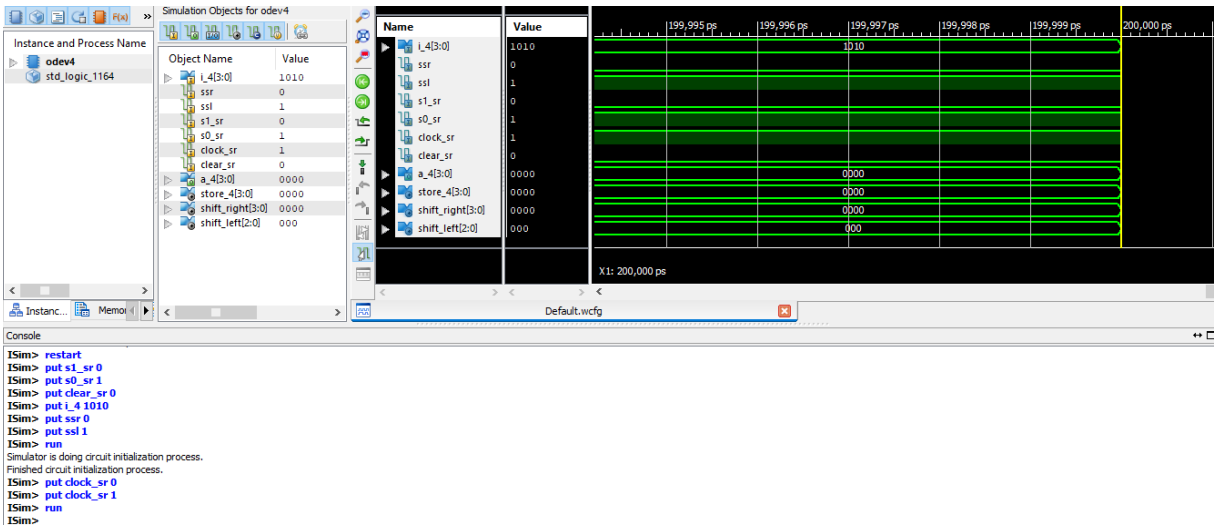
$$Z = \bar{S_1}\bar{S_0}A + \bar{S_1}S_0B + S_1\bar{S_0}C + S_1S_0D$$



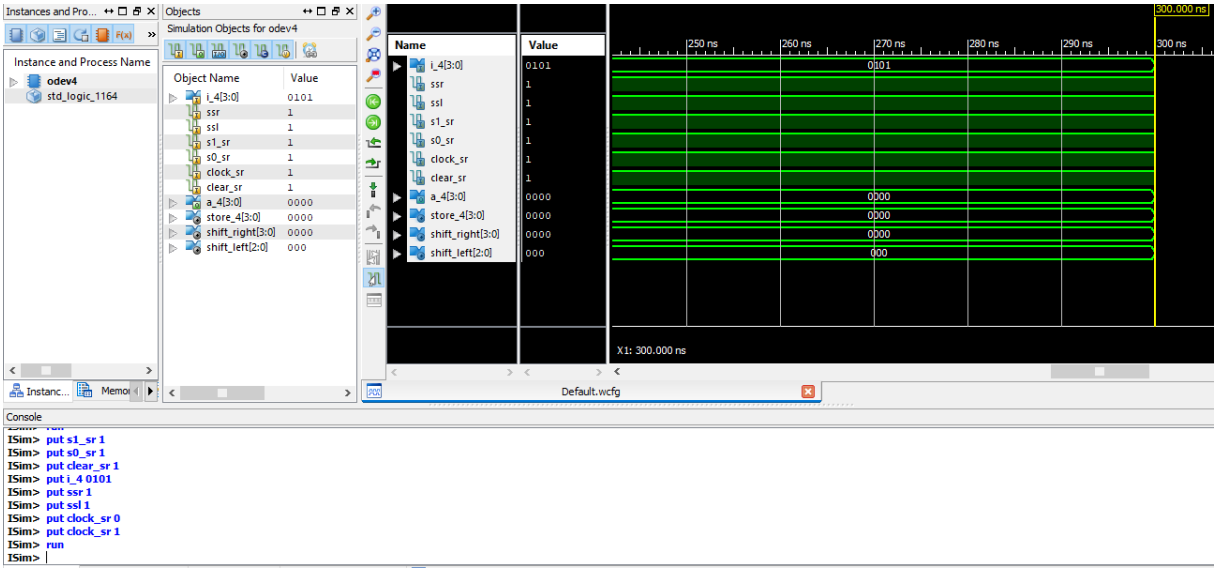

DFF & MUX
Q_out
Q
Clear
D
Clock
mux_to_dff
4x1 MUX
S1
S0
serial_shift
D C B A
dff_mux_4
11 10 01 00
store
shift_right
I_4( )
shift_left

S	CLEAR	I	MSB	LSB
01	0	1010	0	1
11	1	0101	1	1
11	1	0110	1	0
10	1	1101	0	1
01	1	1111	1	1

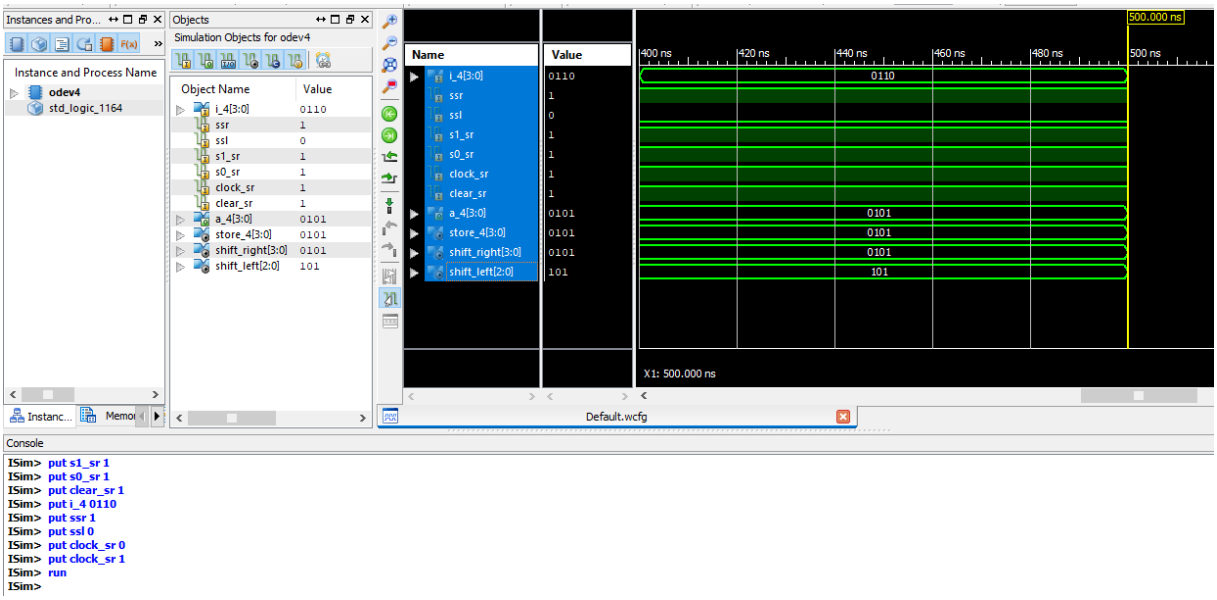




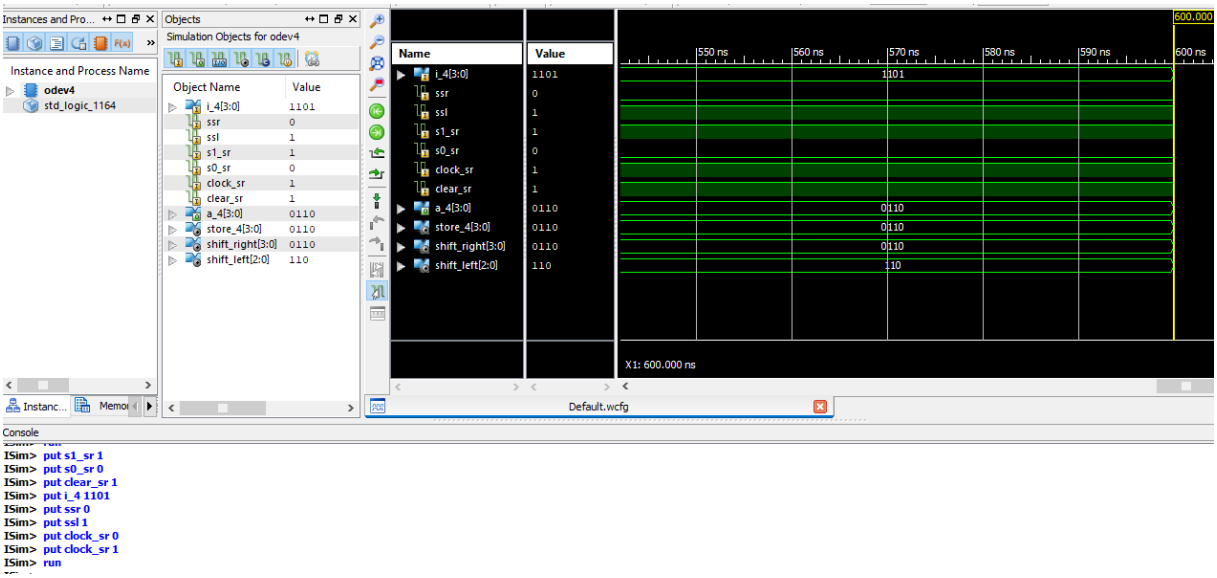
S	CLEAR	I	MSB	LSB
01	0	1010	0	1
11	1	0101	1	1
11	1	0110	1	0
10	1	1101	0	1
01	1	1111	1	1



S	CLEAR	I	MSB	LSB
01	0	1010	0	1
11	1	0101	1	1
11	1	0110	1	0
10	1	1101	0	1
01	1	1111	1	1



S	CLEAR	I	MSB	LSB
01	0	1010	0	1
11	1	0101	1	1
11	1	0110	1	0
10	1	1101	0	1
01	1	1111	1	1



S	CLEAR	I	MSB	LSB
01	0	1010	0	1
11	1	0101	1	1
11	1	0110	1	0
10	1	1101	0	1
01	1	1111	1	1