EASTERN MEDITERRANEAN UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

CMPE224 DIGITAL LOGIC SYSTEMS

EXPERIMENTAL WORK #4

COUNTERS IN VeriLog HDL

OBJECTIVES:

This laboratory work aims to introduce a practical work on the design of Counters from architectural and behavioral descriptions. The architectural description covers both the schematic and the software implementation of circuits designed through the conventional design procedure. The behavioral descriptions cover the implementation using state transition diagrams.

Important Note: *For each of the following experimental tasks (in each phase), open a new project to avoid compilation errors due to multiple use of components within the same project's files.*

Phase 1: Schematic-Entry

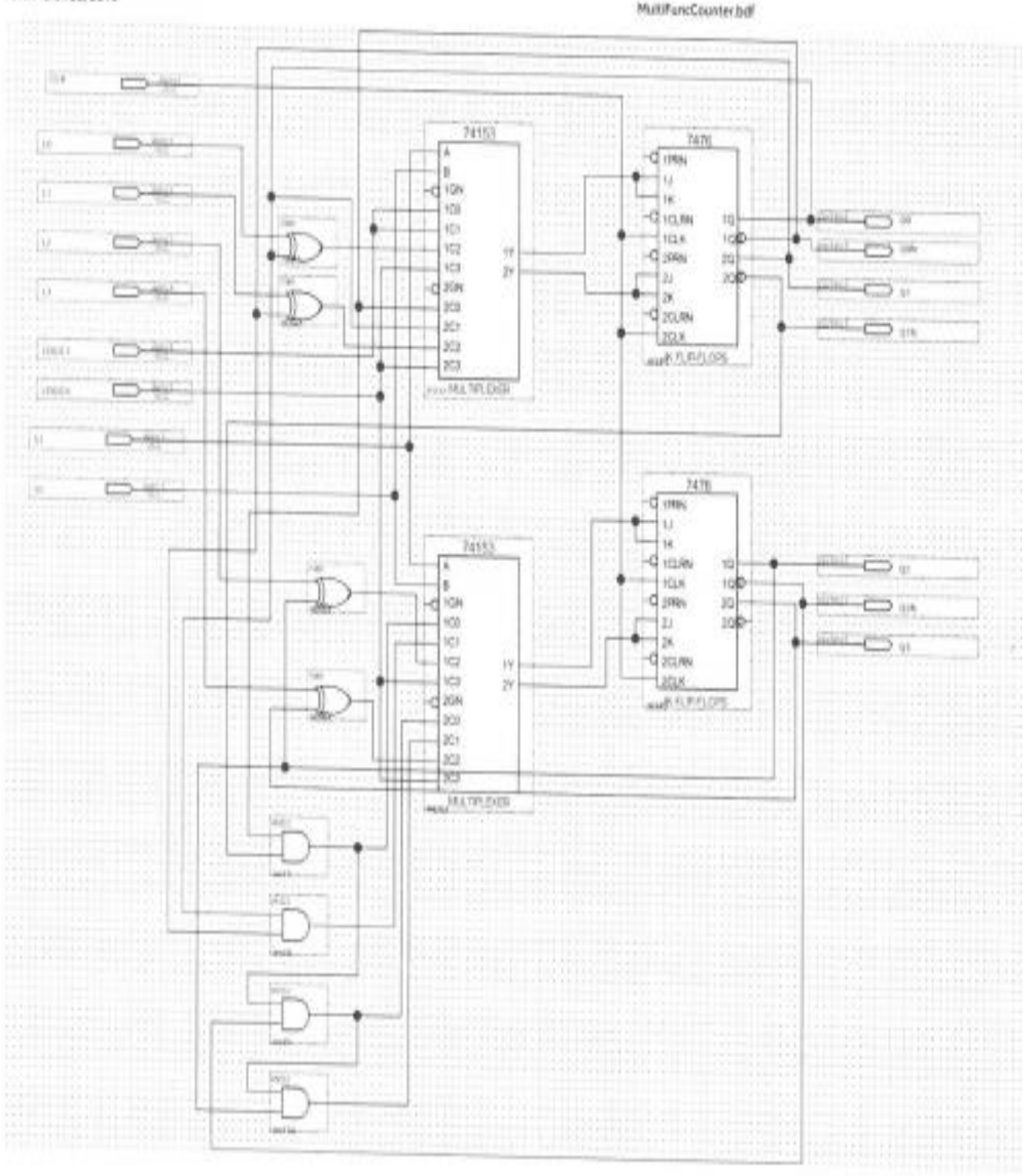
Assume that we want to design the following multi-function counter that is controlled by two control inputs S1 and S0 as follows:

Mode Control		**Register Operation**
S1	***S0***	
0	0	*Count Up*
0	1	*Count Down*
1	0	*Parallel Load*
1	1	*No Change*

The schematic circuit corresponding to this multi-function counter is given below:

1.1. Draw the circuit of this multifunction counter, compile and simulate it in VeriLog HDL environment and verify its mode of operations. Adjust and apply appropriate waveforms to observe different function modes easily.

MultiFuncCounter.bdf



Phase 2: Implementing the Architectural Design of Multi-Function Counter in Verilog HDL

Enter the following architectural VeriLog code of multifunction counter design into Quartus Lite development suite. Compile and simulate your code to verify its correctness.

```
/* Multifunction counter controlled by two control inputs S1 and S0 as follows:
        S1              S0              Operation Mode
        0               0               Count up
        0               1               Count down
        1               0               Prallel load
        1               1               No change
*/

module MultiFuncCounter_Arch(CLK,Clear,S,PL,Q);
        input CLK, Clear;
        input [1:0] S;
        input [3:0] PL;             // Parallel load
        output reg [3:0]Q;          // Counter outputs

        wire [3:0] W;
        wire [3:0] QT;

        MUX_4_1 m1(W[0],S[1],S[0],I31,I21,I11,I01);
        MUX_4_1 m2(W[1],S[1],S[0],I32,I22,I12,I02);
        MUX_4_1 m3(W[2],S[1],S[0],I33,I23,I13,I03);
        MUX_4_1 m4(W[3],S[1],S[0],I34,I24,I14,I04);

        T_FF t1(QT[0],W[0],CLK,Clear);
        T_FF t2(QT[1],W[1],CLK,Clear);
        T_FF t3(QT[2],W[2],CLK,Clear);
        T_FF t4(QT[3],W[3],CLK,Clear);

        assign

                I31=1'b0,
                I21=QT[0]^PL[0],
                I11=1'b1,
                I01=1'b1,

                I32=1'b0,
                I22=QT[1]^PL[1],
                I12=QT[0],
                I02=~QT[0],

                I33=1'b0,
                I23=QT[2]^PL[2],
                I13=QT[1]&QT[0],
                I03=~QT[1] & ~QT[0],

                I34=1'b0,
                I24=QT[3]^PL[3],
                I14=QT[2]&QT[1]&QT[0],
```

```
		I04=~QT[2] & ~QT[1] & ~QT[0];

	always
		Q <= QT;

endmodule


module MUX_4_1(Y,S1,S0,I3,I2,I1,I0);
	input S1,S0,I3,I2,I1,I0;
	output reg Y;

	always @(S1,S0,I3,I2,I1,I0)
	begin
		if (S1==0 & S0==0)
			Y=I0;
		else if (S1==0 & S0==1)
			Y=I1;
		else if (S1==1 & S0==0)
			Y=I2;
		else if (S1==1 & S0==1)
			Y=I3;
	end
endmodule

module T_FF(QT,T,CLK,CLR);
	input T,CLK,CLR;
	output reg QT;

	always @(posedge CLK)
		if (CLR == 1'b1)
			QT<= 1'b0;
		else
			QT<= T^QT;
endmodule
```

HOMEWORK #4 : (Behavioral Description of a Multi-function Counter)

Behavioral VeriLog code of the above described multifunction counter is given below:

```
/* Behavioral description of a multifunction counter in veriLog HDL
        s1 s0 =00 Count up
        S1 S0 =01 Count down
        S1 S0 =10 Parallel load
        S1 S0 =11 No change
*/

module MultiFunctCounter_Behav(CLK,Clear,S,PL,Q);
        input CLK, Clear;
        input [1:0] S;
        input [3:0] PL;             // Parallel load
        output reg [3:0]Q;          // Counter outputs

        reg[3:0] QT;

always @(posedge CLK)
begin
        if (Clear == 1)
                QT <= 4'b0000;
        else if (S[1]==0 & S[0]==0)   // Count up
                QT <= QT+1;
        else if (S[1]==0 & S[0]==1)   // Count down
                QT <= QT-1;
        else if (S[1]==1 & S[0]==0)          // Parallel load
        begin
                QT[3] <= PL[3]; QT[2] <= PL[2];
                QT[1] <= PL[1]; QT[0] <= PL[0];
        end
        else if (S[1]==1 & S[0]==1)
        begin
                QT=QT;
        end
end

always
        Q <= QT;

Endmodule
```

Modify the above-given behavioral code to design a 4-bit multifunction counter that operates as follows:

Enable	**S1**	**S0**	**Operation Mode**
0	*x*	*x*	*No change*
1	*0*	*0*	*Complement*
1	*0*	*1*	*Count up by two*
1	*1*	*0*	*Count down by two*
1	*1*	*1*	*Shift left*

Submit your homework at the beginning of the 5-th experimental work.

Prepared by Assoc. Prof. Dr. Adnan ACAN