Roni Das ESE 218: L03

ID:108378223 Prelab 12

Objective to approximate analog signal by concept of Direct Digital Synthesis using the help of an 8-bit DAC was carried out in this laboratory experiment. Following the data sheet of 8-bit DAC MCP4801, we denote all the necessary inputs required for the DAC to approximate an analog saw-wave.

DAC requirements:

SCK : 16 period clock

Config bits : 0111 (7)

SDI : Serial data input from a magnitude counter.

CSbar : Active low to initiate operation.

Following laboratory manual 12, SPI master was obtained with control logic, two 8-bit up counter and Parallel in serial Out Shift resister. Following diagram depicts data path and controller units.

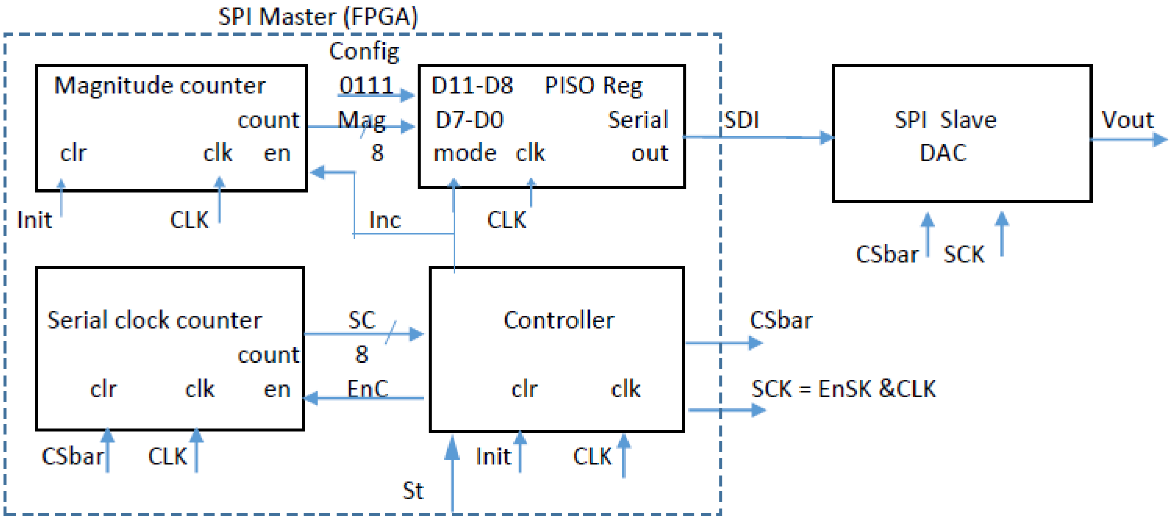


Figure 1: The block diagram of the system.

Following the ASM Chart provide by the laboratory manual, control unit was designed and implemented in active HDL.

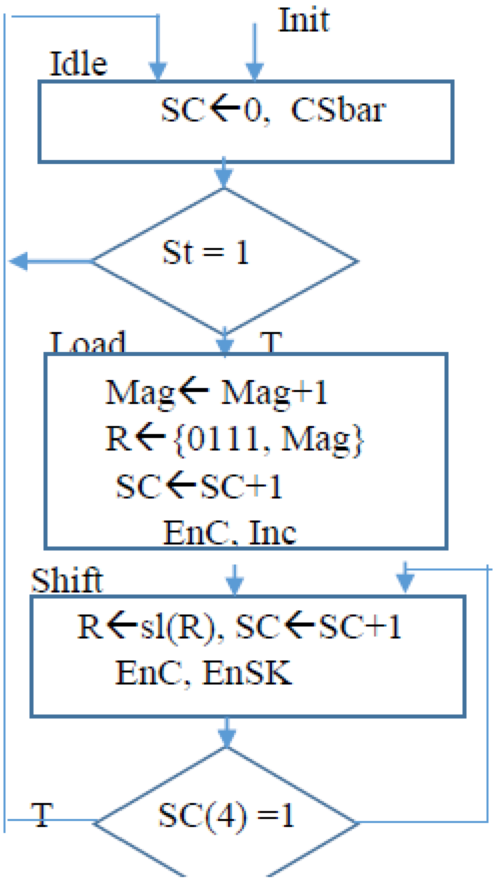
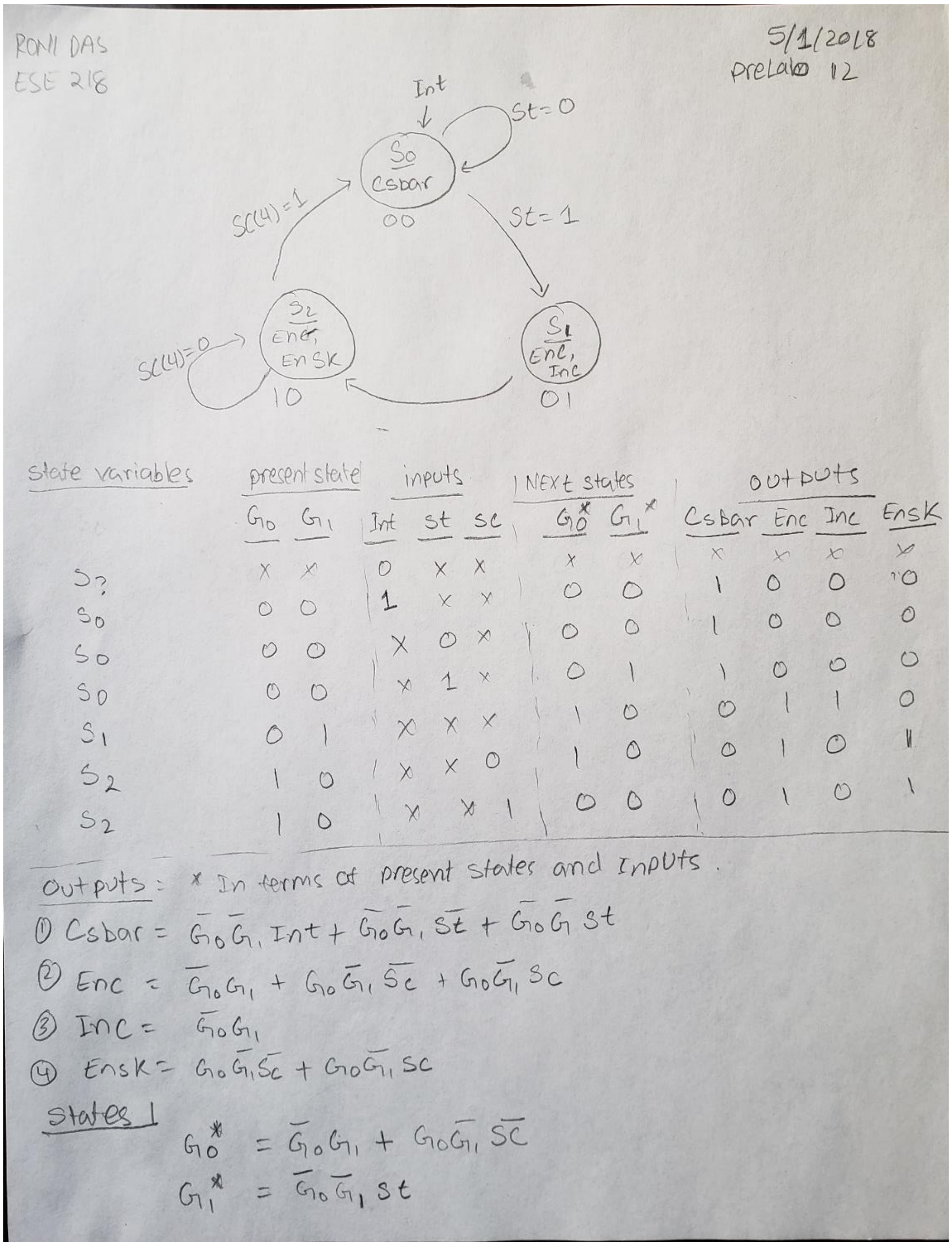


Figure 2: The ASM chart for control unit.

In obtaining control logic all states of ASM chart was identified along with all decisions box (inputs) to system. State diagram was obtained from the chart.


RONI DAS
ESE 218

5/1/2018
preLab 12

Int
St=0
S0
Csbar
00
St=1
SC(4)=1
S2
Enc,
EnSK
10
SC(4)=0
S1
Enc,
Inc
01

state variables	present state		inputs			NEXt states		outputs			
	G_0	G_1	Int	st	sc	G_0^*	G_1^*	Csbar	Enc	Inc	EnsK
S?	x	x	0	x	x	x	x	x	x	x	x
S0	0	0	1	x	x	0	0	1	0	0	0
S0	0	0	X	0	x	0	0	1	0	0	0
S0	0	0	x	1	x	0	1	1	0	0	0
S1	0	1	X	X	X	1	0	0	1	1	0
S2	1	0	x	x	0	1	0	0	1	0	1
S2	1	0	x	x	1	0	0	0	1	0	1

Outputs: * In terms of present states and InpUts.

① Csbar = $\bar{G_0}\bar{G_1}\,Int + \bar{G_0}\bar{G_1}\,\bar{St} + \bar{G_0}\bar{G}\,St$

② Enc = $\bar{G_0}G_1 + G_0\bar{G_1}\,\bar{Sc} + G_0\bar{G_1}\,Sc$

③ Inc = $\bar{G_0}G_1$

④ Ensk = $G_0\bar{G_1}\bar{Sc} + G_0\bar{G_1}\,Sc$

States |

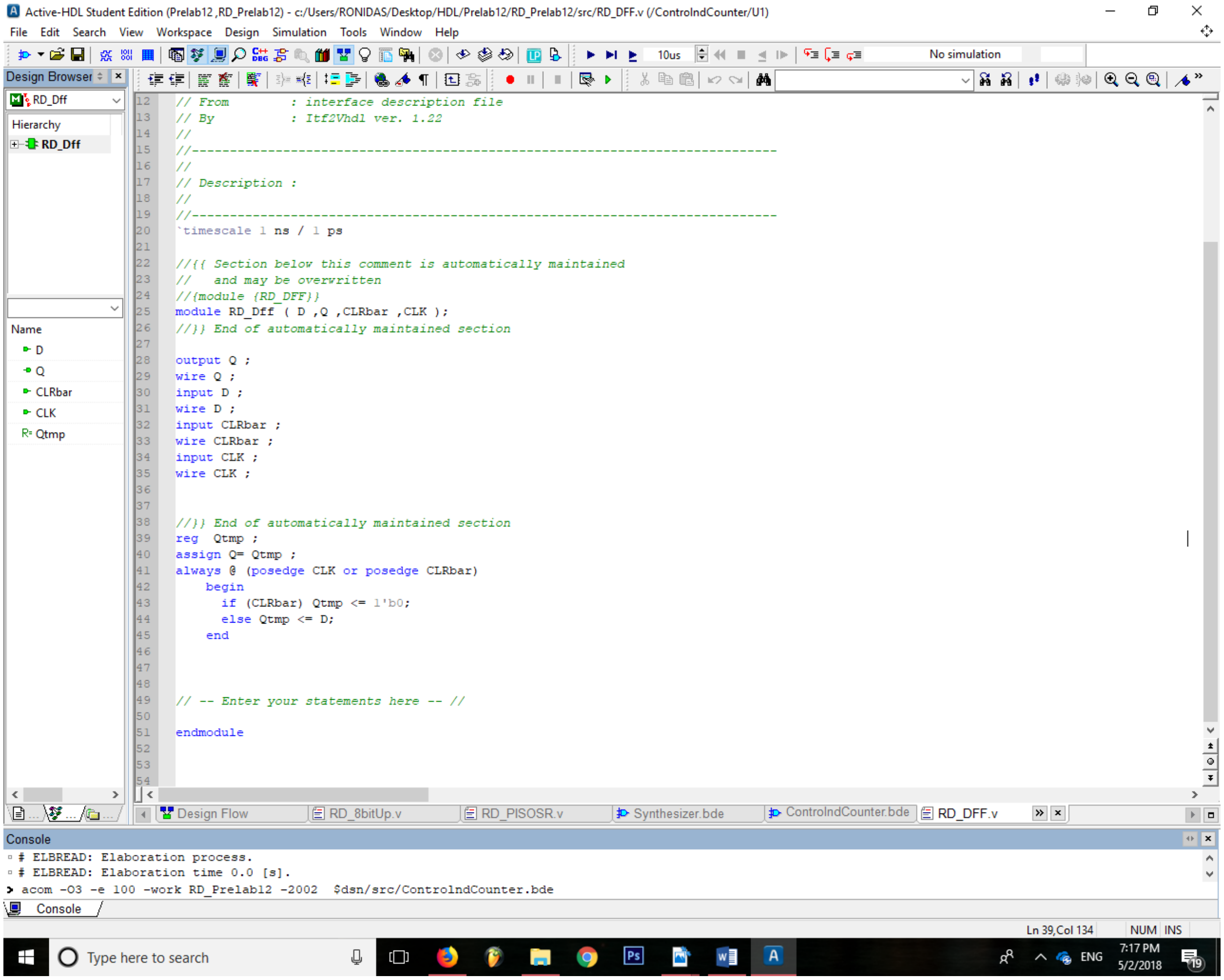
$G_0^* = \bar{G_0}G_1 + G_0\bar{G_1}\,\bar{SC}$

$G_1^* = \bar{G_0}\bar{G_1}\,St$

Figure 3: State diagram and State Table for Controller.

As shown above, State diagram contains all the information derived from ASM chart. Following the diagram State Table was obtained, for further dissection of control logic. Finally, equations for Outputs and next states were obtained from state table. For three states sequence, we require 2 D- flip flops. Outputs will be decoded from flow of the signals.

For accurate designing of SPI master, operations of each component were verified, and results presented below.

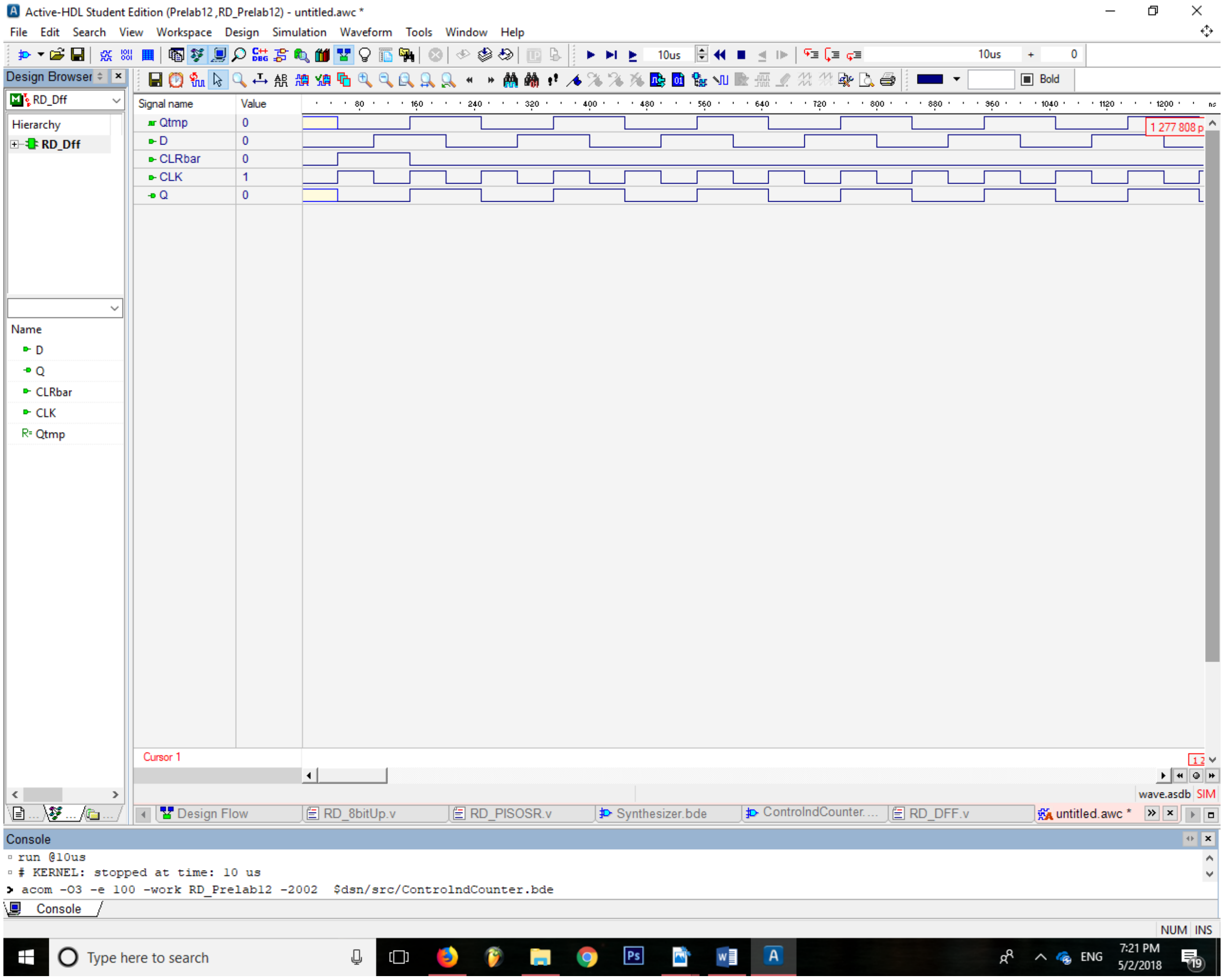


```
// From         : interface description file
// By           : Itf2Vhdl ver. 1.22
//
//-----------------------------------------------------------------------------
//
// Description :
//
//-----------------------------------------------------------------------------
`timescale 1 ns / 1 ps

//{{ Section below this comment is automatically maintained
//   and may be overwritten
//{module {RD_DFF}}
module RD_Dff ( D ,Q ,CLRbar ,CLK );
//}} End of automatically maintained section

output Q ;
wire Q ;
input D ;
wire D ;
input CLRbar ;
wire CLRbar ;
input CLK ;
wire CLK ;


//}} End of automatically maintained section
reg  Qtmp ;
assign Q= Qtmp ;
always @ (posedge CLK or posedge CLRbar)
    begin
      if (CLRbar) Qtmp <= 1'b0;
      else Qtmp <= D;
    end



// -- Enter your statements here -- //

endmodule
```

```
# ELBREAD: Elaboration process.
# ELBREAD: Elaboration time 0.0 [s].
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/ControlndCounter.bde
```

Figure 4: Verilog model for D-flip flops.

Verilog model for D-flip flops was obtained in Active HDL. Here, CLRbar simply clears the flip to initiate state transitions and is active high.

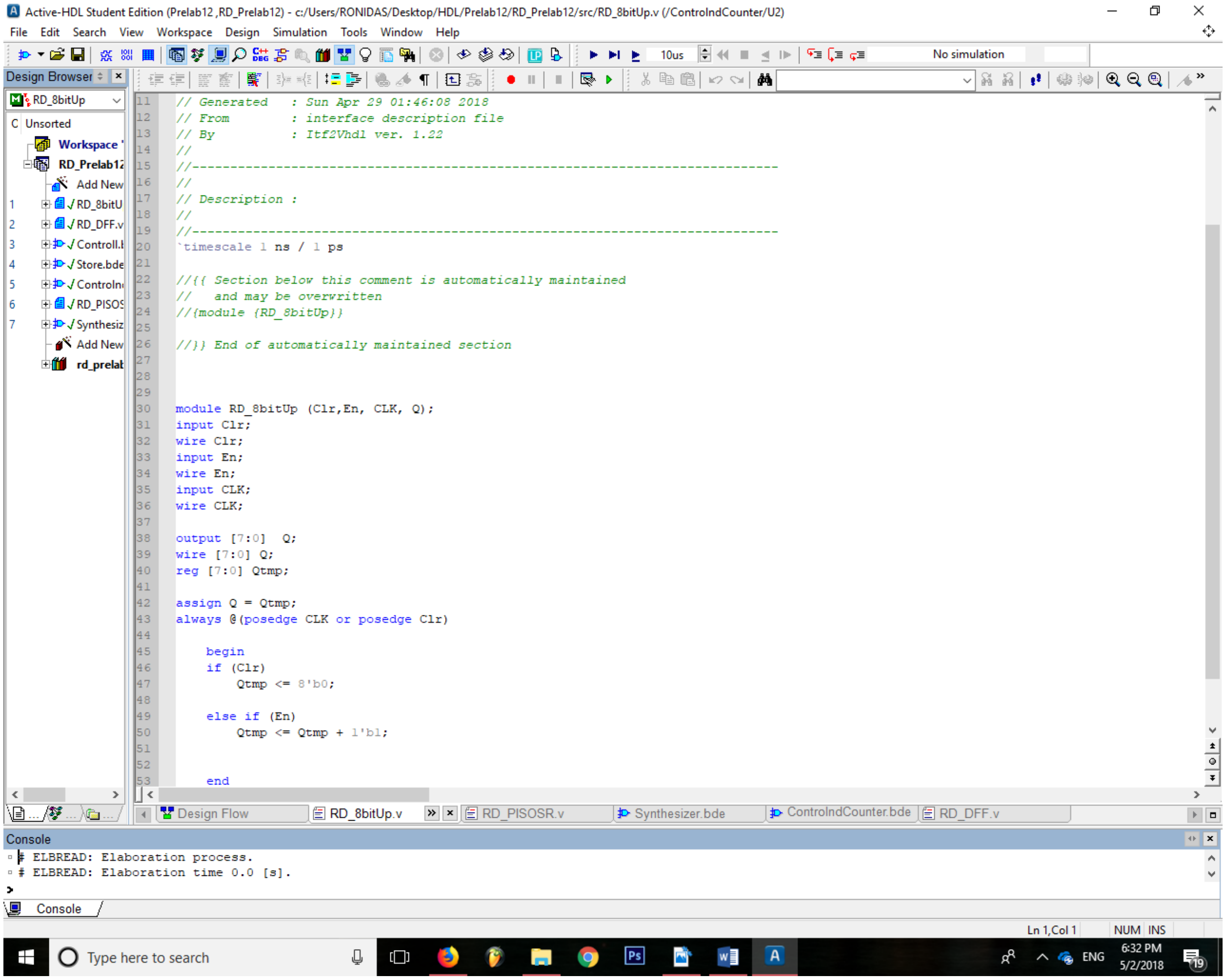


Signal name	Value
Qtmp	0
D	0
CLRbar	0
CLK	1
Q	0

```
▫ run @10us
▫ # KERNEL: stopped at time: 10 us
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/ControlndCounter.bde
```

Figure 5: Operation of D-flip flops verified.

We can decode states Q (output) from the waveforms presented above. Initially CLRbar clears the D-ff, followed by positive clock edge D (input) transfers to Q . On the negative edge of the clock state of Q is unchanged regardless of the inputs values. The model operates as expected and cleared for implantation in control modules.



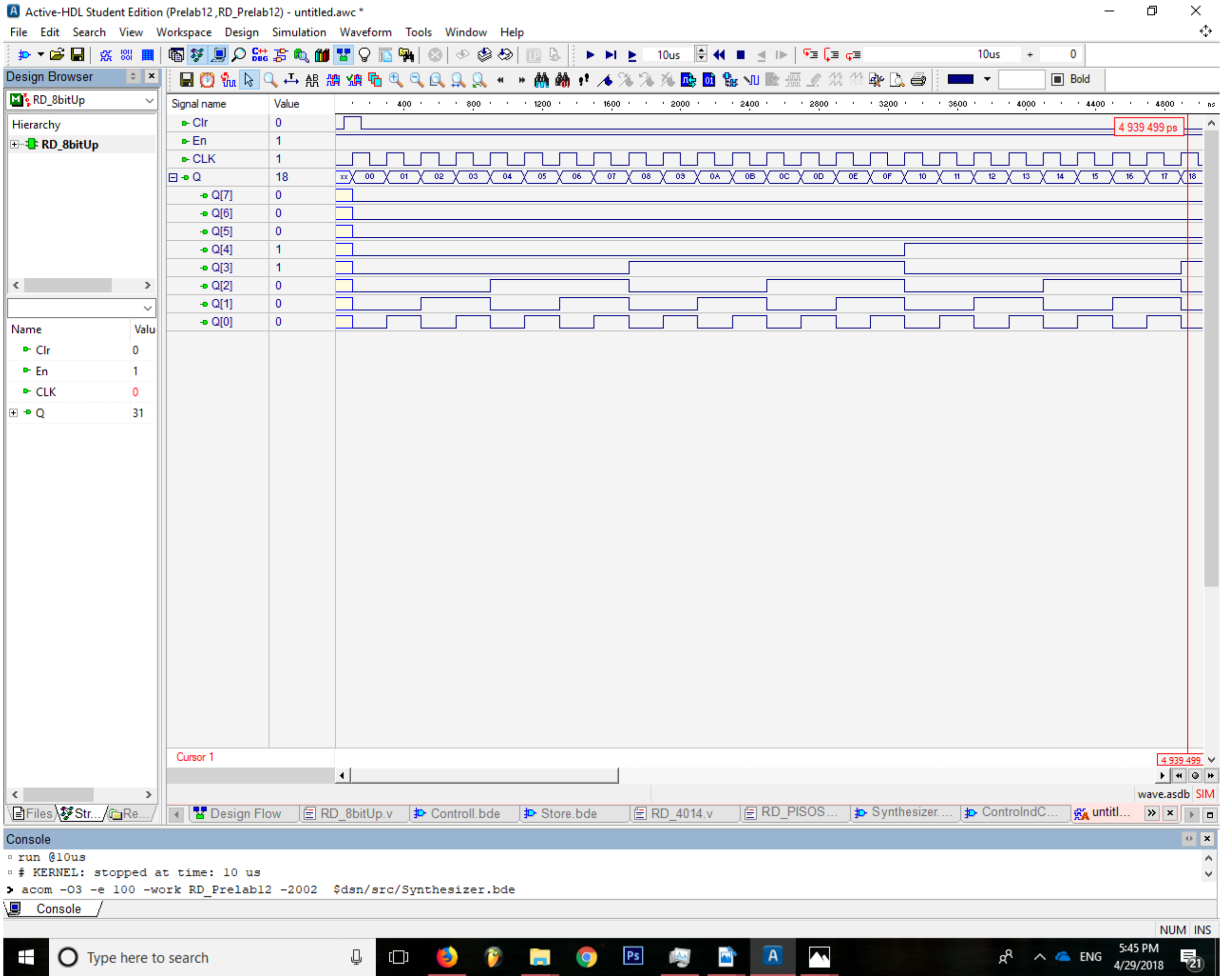
Active-HDL Student Edition (Prelab12 ,RD_Prelab12) - c:/Users/RONIDAS/Desktop/HDL/Prelab12/RD_Prelab12/src/RD_8bitUp.v (/ControlndCounter/U2)

```
// Generated   : Sun Apr 29 01:46:08 2018
// From        : interface description file
// By          : Itf2Vhdl ver. 1.22
//
//-----------------------------------------------------------------------------
//
// Description :
//
//-----------------------------------------------------------------------------
`timescale 1 ns / 1 ps

//{{ Section below this comment is automatically maintained
//   and may be overwritten
//{module {RD_8bitUp}}

//}} End of automatically maintained section



module RD_8bitUp (Clr,En, CLK, Q);
input Clr;
wire Clr;
input En;
wire En;
input CLK;
wire CLK;

output [7:0]  Q;
wire [7:0] Q;
reg [7:0] Qtmp;

assign Q = Qtmp;
always @(posedge CLK or posedge Clr)

    begin
    if (Clr)
        Qtmp <= 8'b0;

    else if (En)
        Qtmp <= Qtmp + 1'b1;


    end
```

```
# ELBREAD: Elaboration process.
# ELBREAD: Elaboration time 0.0 [s].
>
```

Figure 6: Verilog model for 8-bit Up Counter.

An 8-bit up counter was described using Verilog in Active HDL. From the code *if and else* , we denote the CLR has the highest priority followed by En. When clear is High, counter resets, and En = Enable is on, system starts counting from 0 .



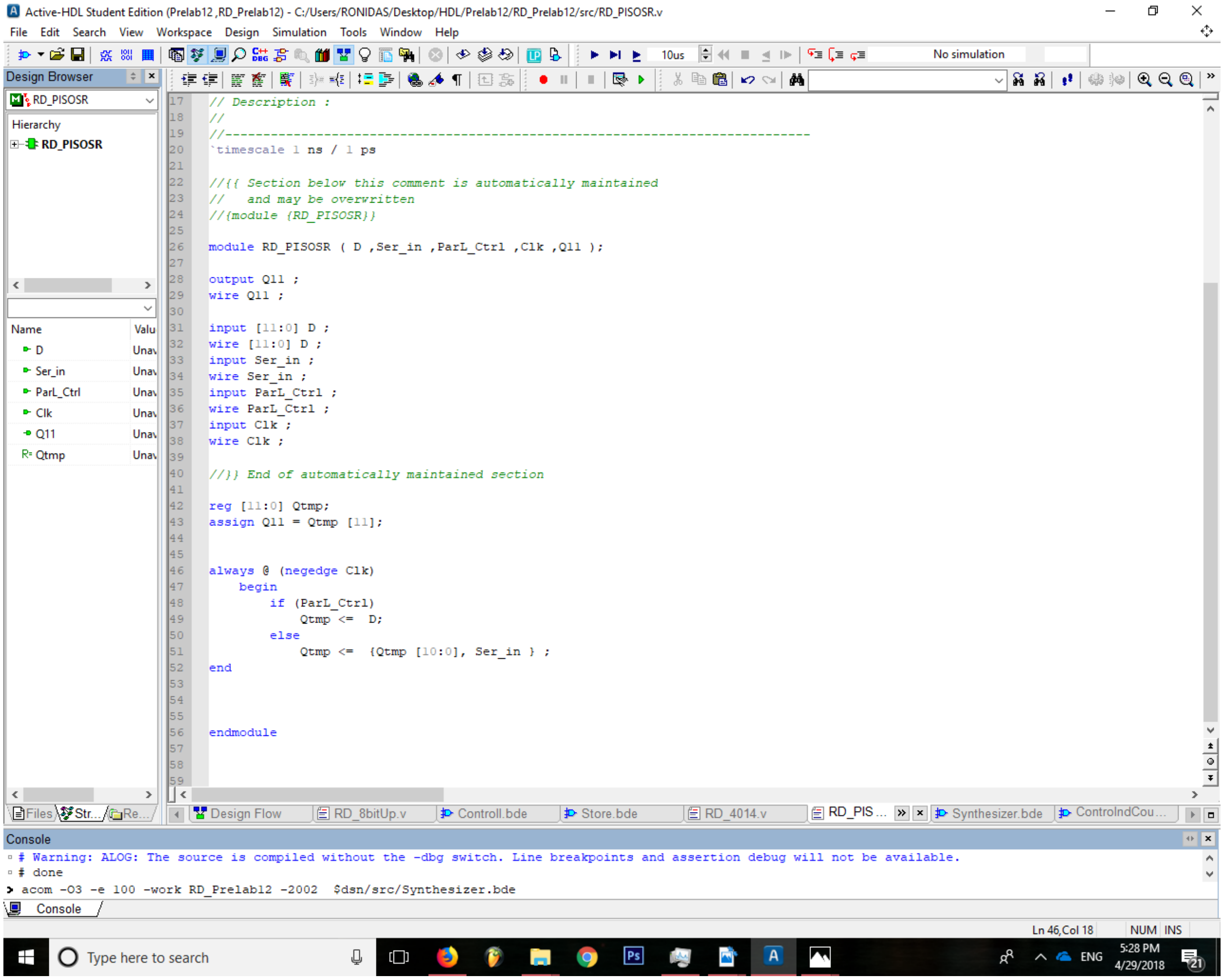
Signal name	Value
Clr	0
En	1
CLK	1
Q	18
Q[7]	0
Q[6]	0
Q[5]	0
Q[4]	1
Q[3]	1
Q[2]	0
Q[1]	0
Q[0]	0

Name	Valu
Clr	0
En	1
CLK	0
Q	31

```
▫ run @10us
▫ # KERNEL: stopped at time: 10 us
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/Synthesizer.bde
```

Figure 7: Operations of 8-bit Up counter verified using simulation.

From the waveforms above, we can denote that the counters works properly and given a pass to be implemented in SPI master.



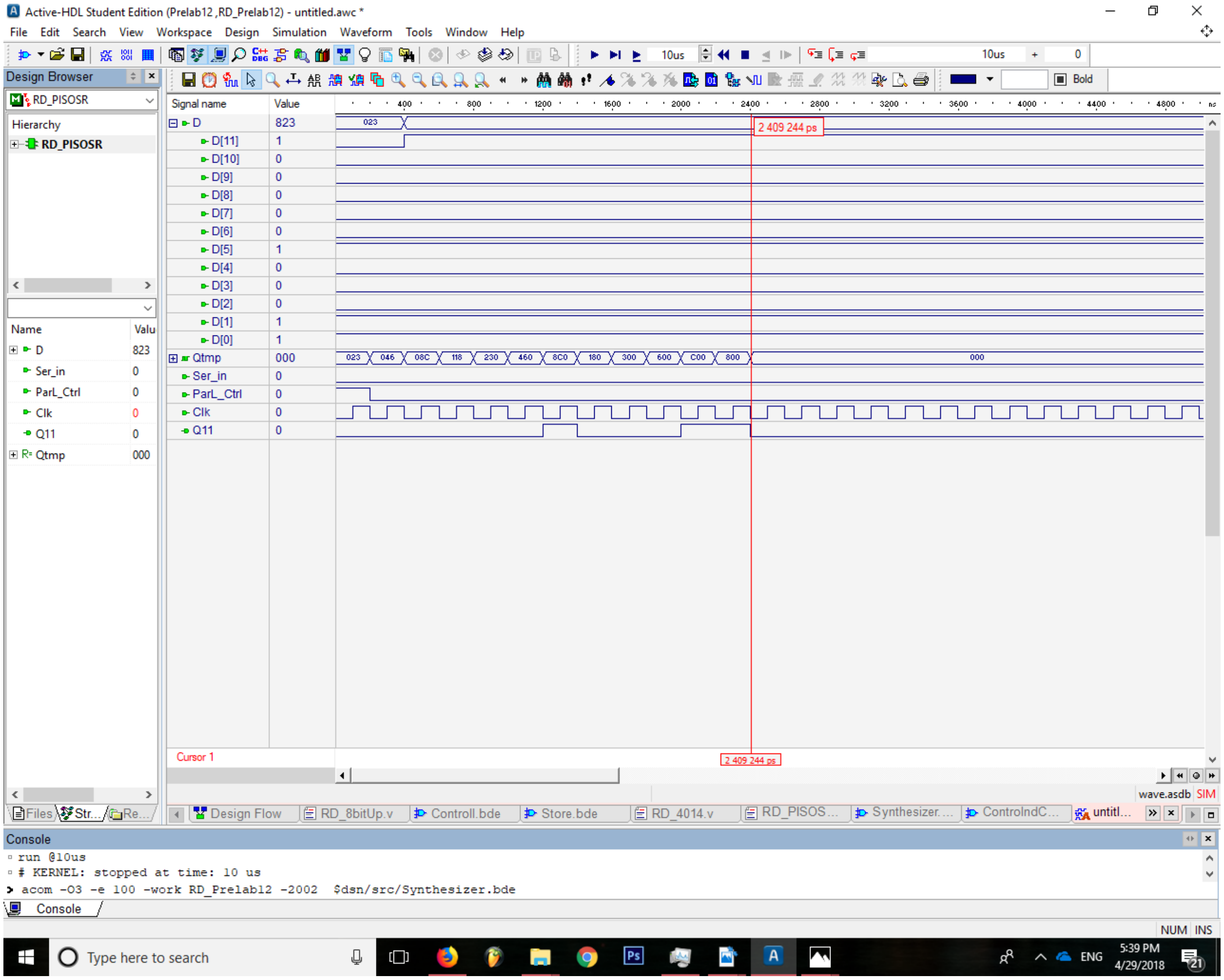
Active-HDL Student Edition (Prelab12 ,RD_Prelab12) - C:/Users/RONIDAS/Desktop/HDL/Prelab12/RD_Prelab12/src/RD_PISOSR.v

Name	Valu
D	Unav
Ser_in	Unav
ParL_Ctrl	Unav
Clk	Unav
Q11	Unav
Qtmp	Unav

```
// Description :
//
//-----------------------------------------------------------------------------
`timescale 1 ns / 1 ps

//{{ Section below this comment is automatically maintained
//   and may be overwritten
//{module {RD_PISOSR}}

module RD_PISOSR ( D ,Ser_in ,ParL_Ctrl ,Clk ,Q11 );

output Q11 ;
wire Q11 ;

input [11:0] D ;
wire [11:0] D ;
input Ser_in ;
wire Ser_in ;
input ParL_Ctrl ;
wire ParL_Ctrl ;
input Clk ;
wire Clk ;

//}} End of automatically maintained section

reg [11:0] Qtmp;
assign Q11 = Qtmp [11];


always @ (negedge Clk)
    begin
        if (ParL_Ctrl)
            Qtmp <=  D;
        else
            Qtmp <=  {Qtmp [10:0], Ser_in } ;
end



endmodule
```

```
# Warning: ALOG: The source is compiled without the -dbg switch. Line breakpoints and assertion debug will not be available.
# done
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/Synthesizer.bde
```

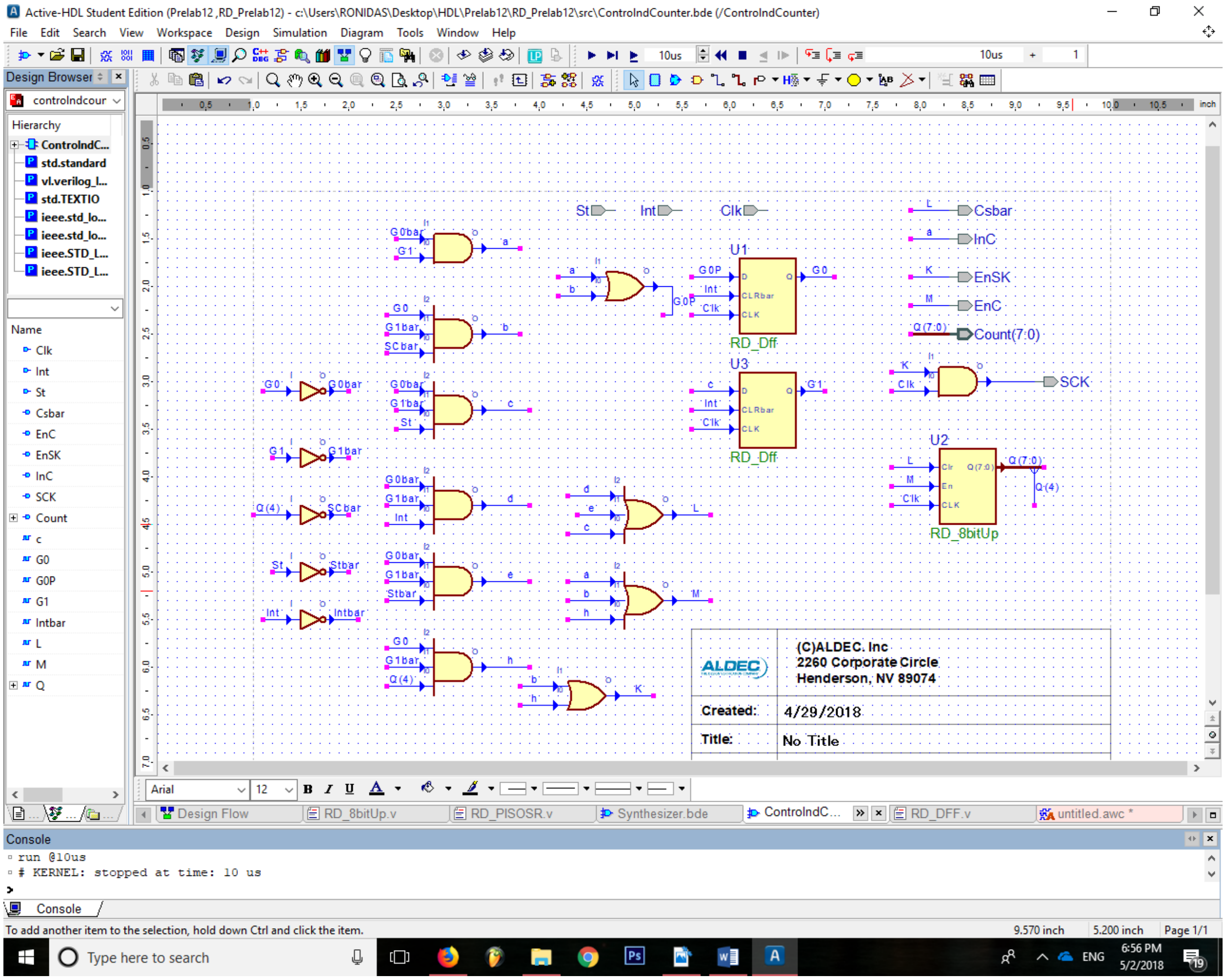
Figure 8: Verilog model for 12-bit PISO SR.



```
run @10us
# KERNEL: stopped at time: 10 us
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/Synthesizer.bde
```

Figure 9: 12-bit PISO SR operation verified.

Q11 = Serial output of 12-bit PISO SR. By moving the cursor from left to right, one can count digit 0 , 2 , 3 from Q11. ID digit 0, 2 , 3 was indeed used as data input. Using parallel control Data in was processed to Q11. Note the serial input is grounded.



Active-HDL Student Edition (Prelab12 ,RD_Prelab12) - c:\Users\RONIDAS\Desktop\HDL\Prelab12\RD_Prelab12\src\ControlndCounter.bde (/ControlndCounter)

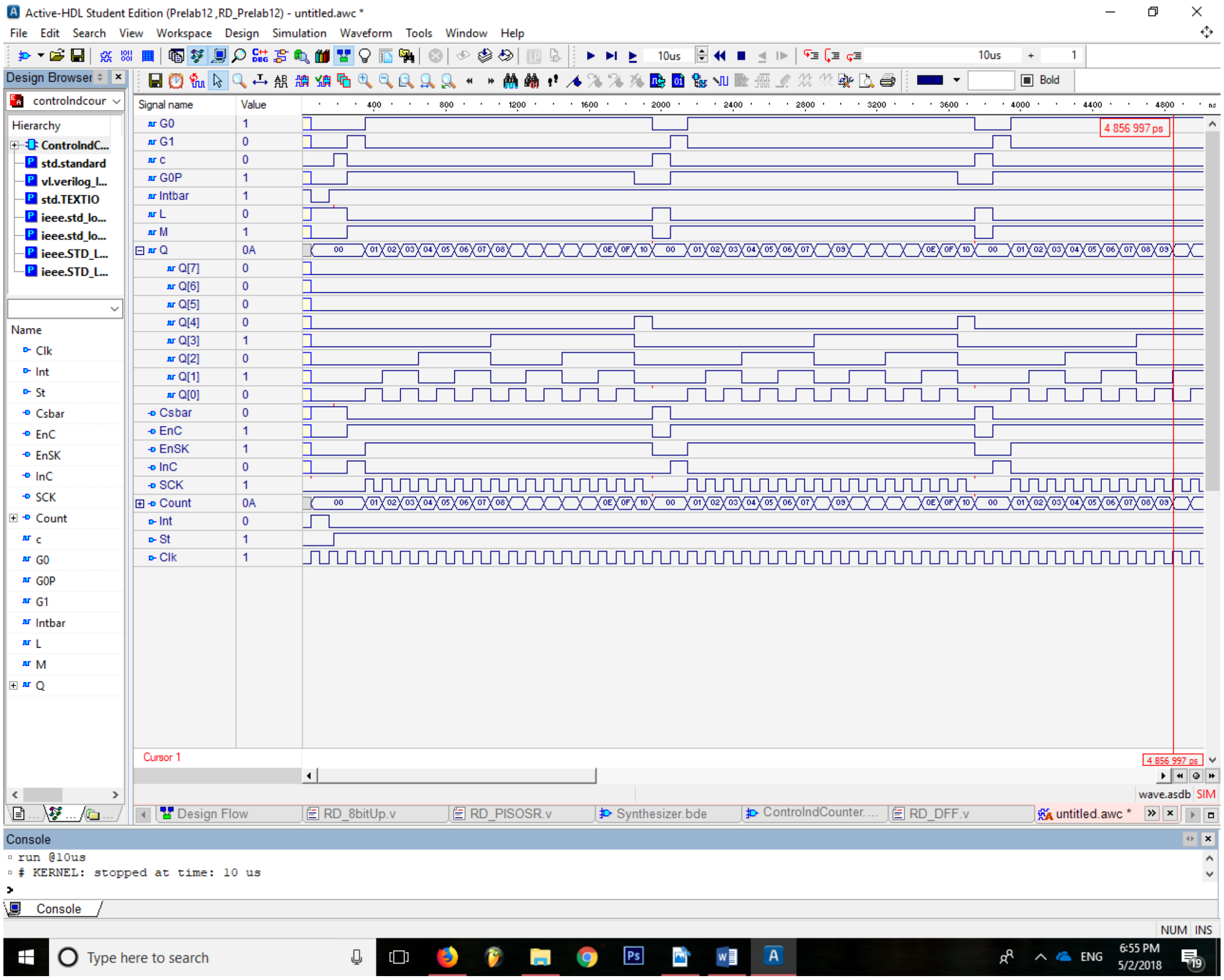
```
run @10us
# KERNEL: stopped at time: 10 us
>
```

To add another item to the selection, hold down Ctrl and click the item.

Figure 10: Controller schematic.

Following state equations and output equations controller logic was obtained and implemented in Active HDL. Controller counts with the help of 8-bit up counter.

SC = Q(4) decodes 16 from 8-bit up counter’s output. Schematic converted into fub for later implementation.

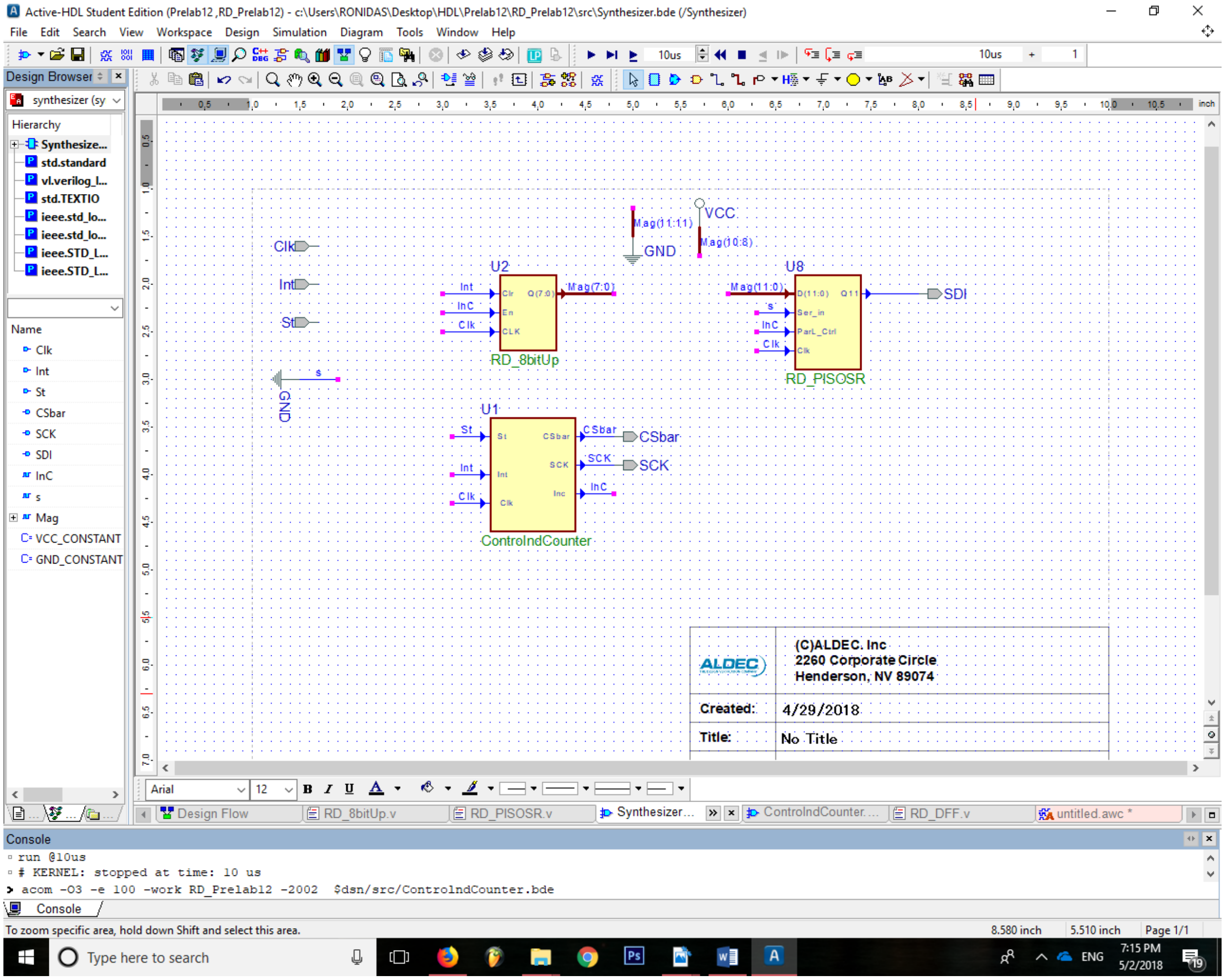


Signal name	Value
G0	1
G1	0
c	0
G0P	1
Intbar	1
L	0
M	1
Q	0A
Q[7]	0
Q[6]	0
Q[5]	0
Q[4]	0
Q[3]	1
Q[2]	0
Q[1]	1
Q[0]	0
Csbar	0
EnC	1
EnSK	1
InC	0
SCK	1
Count	0A
Int	0
St	1
Clk	1

```
run @10us
# KERNEL: stopped at time: 10 us
>
```

Figure 11: Controller's logic in Waveforms.

As depicted above, G0 and G1 = states. Outputs of counter is shown as bus Outputs = Q(7:0) . SC= Q(4) = 1 cuts off counter and cycles back to idle state . Csbar = 1 , at beginning of each cycle. SCK provides 16 clock periods needed by the DAC .

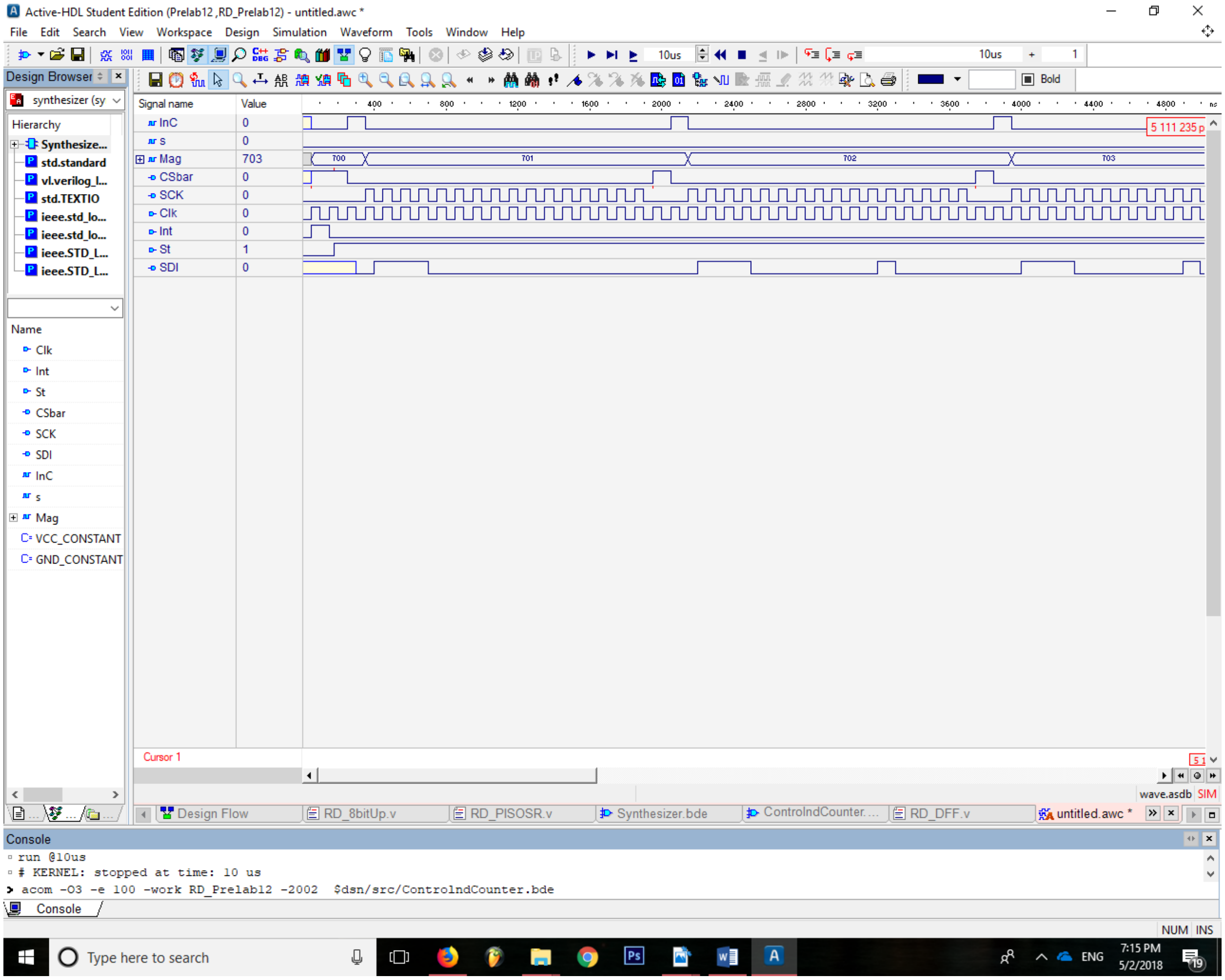


```
▫ run @10us
▫ # KERNEL: stopped at time: 10 us
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/ControlndCounter.bde
```

To zoom specific area, hold down Shift and select this area.

Figure 12 : SPI Master implementation in Active HDL.

Control unit and a 8-bit Counter was converted to fub (single box) ; only necessary outputs were obtained from controller for further simplification in circuit design. An 8-bit Counter(aptly named Magnitude counter) was implemented in data path (shown above). 12-bit PSIO SR , takes inputs from Magnitude counter for 8 bit data D(7:0) and 4-config Bit = 0111 D(11:8). SDI is the final Output of the system (readied for DAC).



Signal name	Value
InC	0
s	0
Mag	703
CSbar	0
SCK	0
Clk	0
Int	0
St	1
SDI	0

```
run @10us
# KERNEL: stopped at time: 10 us
> acom -O3 -e 100 -work RD_Prelab12 -2002  $dsn/src/ControlndCounter.bde
```

Figure 13: SPI Master operation verified in Waveform.

As shown in the waveform, SPI Master is operating correctly as expected. SCK provided 16 clock periods for DAC. CSbar activates at beginning of each cycle, (puts DAC in high impedance mode after each SCK cycle) . SDI produces all data needed for generation of an analog saw-wave signal.