Birzeit University

Faculty of Information Technology

Computer Systems Engineering Department

Digital Lab ENCS 211 EXP. No. 8

Verilog: Part II

Objective:

In this experiment we will continue using Quartus II to implement systems in verilog and schematic methods. We will implement a project consisting form several modules. Then we will implement an algorithm using verilog.

Pre-lab:

1-Read the experiment before the lab,

2-Build a 4-bit counter:

The counter have two inputs (clock and reset) and one output (count)

- Create the project counter
- Write the file counter.v
- Process a functional simulation
- Create default symbol

Procedure:

Frequency division example:

Study and simulate the example of the frequency division shown below:

```
1  module freq_div(freq1,rst,freq2);
2    input freq1;
3    input rst;
4    output freq2;
5    wire freq1,rst;
6    reg freq2;
7    reg [2:0] counter;
8    always @(posedge freq1 or negedge rst)
9  begin
10   if(rst==0)
11 begin
12       counter=0;
13       freq2=0;
14   end
15   else
16 begin
17   if (counter==0)
18       begin
19       counter=4;
20       freq2=0;
21       end
22   else
23       begin
24       counter=counter-1;
25       freq2=1;
26       end
27   end
28   end
29   endmodule
```

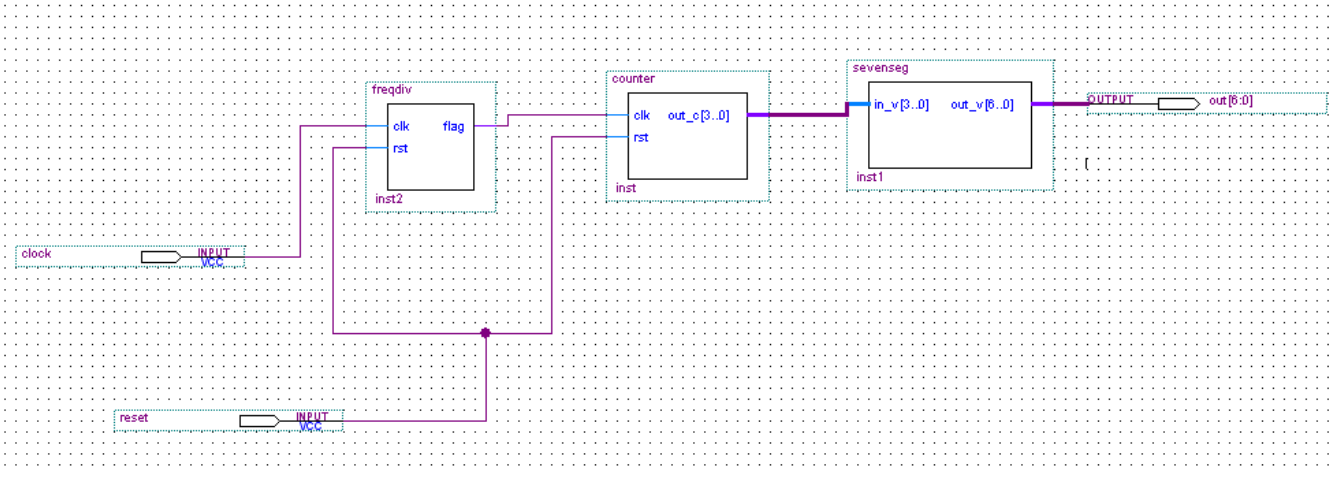
What is the ratio between freq1 and freq2?

In DE1 Kit, there are three clocks (27,24,50 Mhz). Because we want to see the output of the counter, we cant use these clocks directly as input to the counter, therefore modify the above (freqeuncy division) code such that, if the input frequncy is 27 Mhz output freqency is around 2Hz.
Then compile it, and produce a default symbol for it.
Make sure that you have the file divefreq.v, sevenseg.v and counter.v in the same directory you are working on, and it is compiled correctly, and there is a symbol for each one.
(File→create/update→create symbol files for current file)

Project:

Now we are going to connect the three parts above to make the counter count at
Frequency 2Hz and the result appear on 7seg display, to do so :

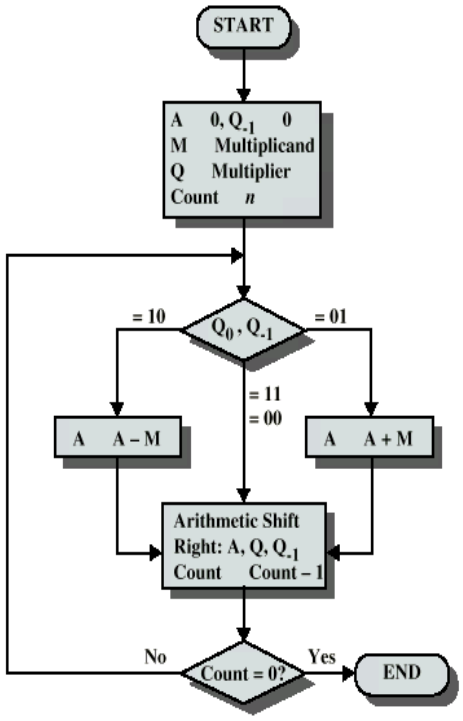
- Create a new project
- Open a new schematic part and put all blocks in this file
- Now connect all parts as shown in the following figure



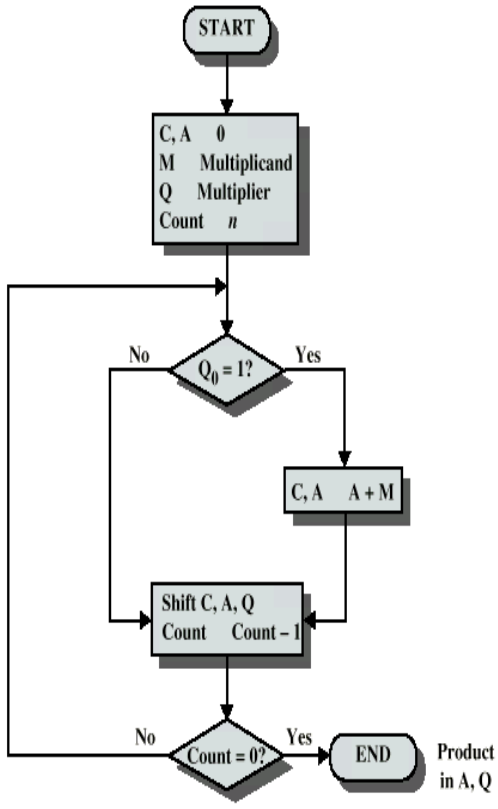
Compile the new file and make sure there are no errors in the file.
Assign the clock, and reset pins, and the output bin to the pins for the clock and
Reset (use the user manual to locate clock and push button), you can assign either by right click the pin then choose locate in assignment editor , or by clicking the assignments icon then assign pins then assign the pins and save.

Unsigned multiplier

In this section we are going to implement an unsigned multiplication algorithm based on shift and add as shown in the flow chart bellow. A verilog implementation of unsigned binary multiplication is shown below. Sudy the program, simulate it. After that implement Booth's algorithm and simulate it.



Booth's Algorithm for two's complement multiplication



unsigned multiplication

```
module mult(product,ready,multiplier,multiplicand,start,clk);
    input [7:0]  multiplier;
    input [7:0]  multiplicand;
    input         start, clk;
    output        product;
    output        ready;

    reg [15:0]    product;

    reg [7:0]     Q;
    reg [7:0]     M;
    reg [8:0]     ACC;
    reg [3:0]      counter;
    wire           ready = !counter;

    initial counter = 0;

    always @( posedge clk )

   if( ready && start ) begin

         counter= 8;
         ACC= 0;
        M = multiplicand ;
         Q = multiplier;
  end else if( counter ) begin

         if( Q[0] == 1'b1 ) ACC = ACC + M;

         Q= Q>> 1;
         Q[7]=ACC[0];
         ACC= ACC>> 1;
         counter = counter - 1;

     if (counter==0) product={ACC[7:0],Q};

      end
endmodule
```