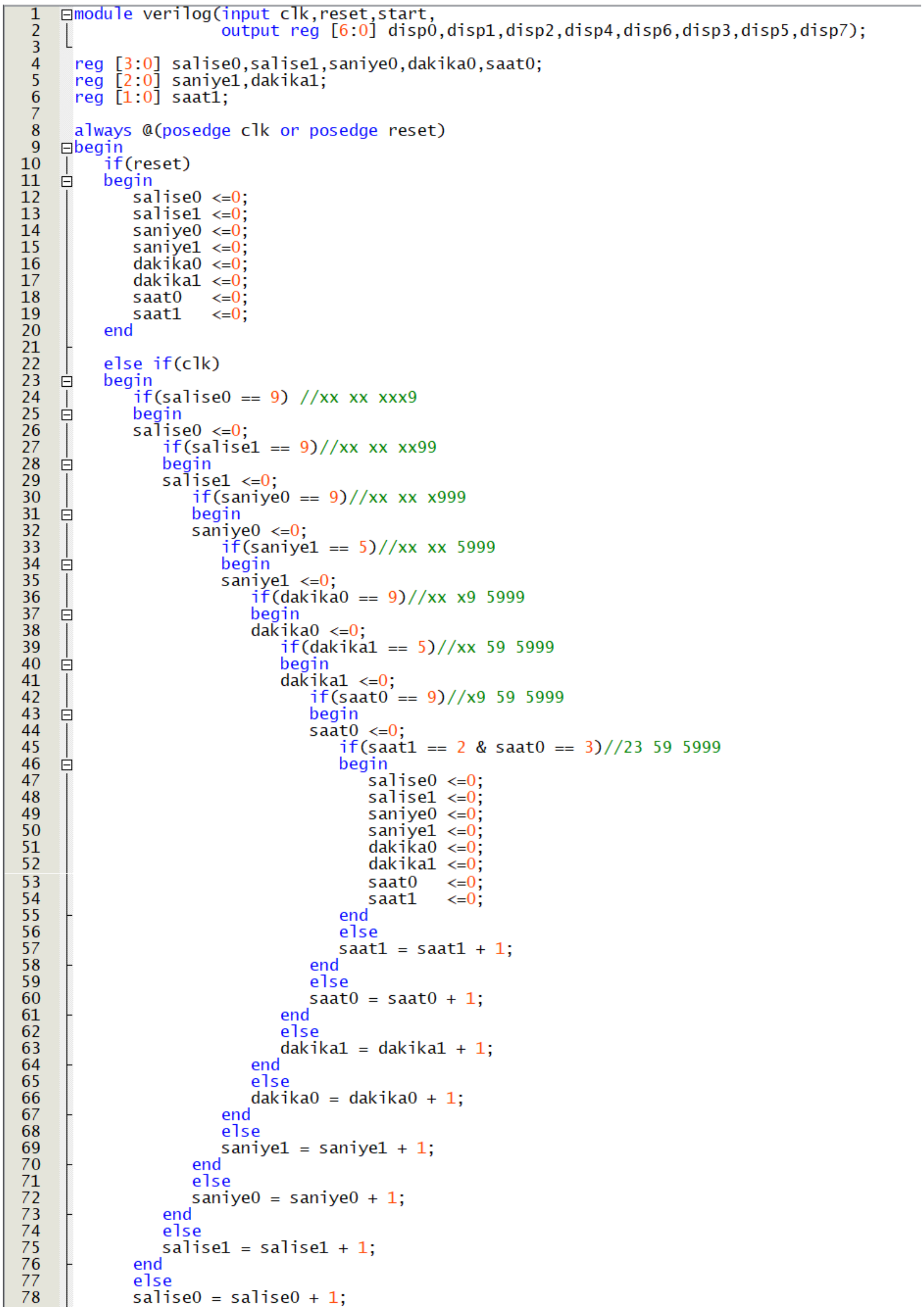
Workshop on Programmable Logic Devices

Łukasz Sajewski

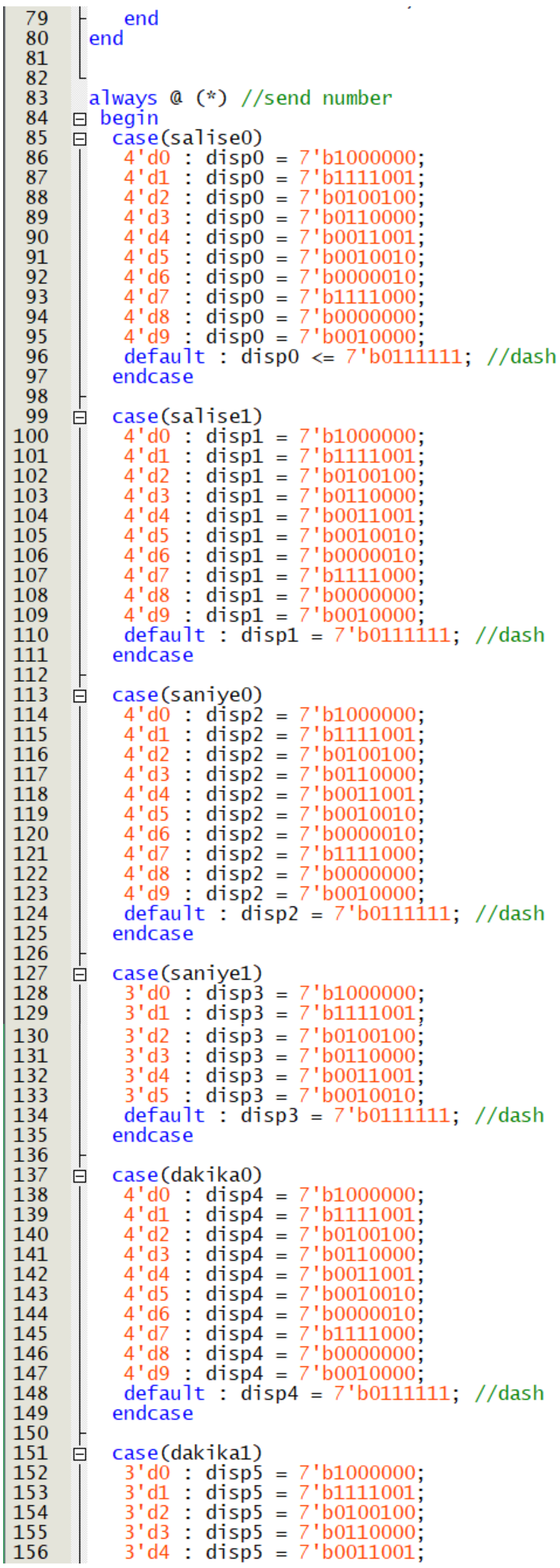
Tasks Part-3

Burak ELHAMAN

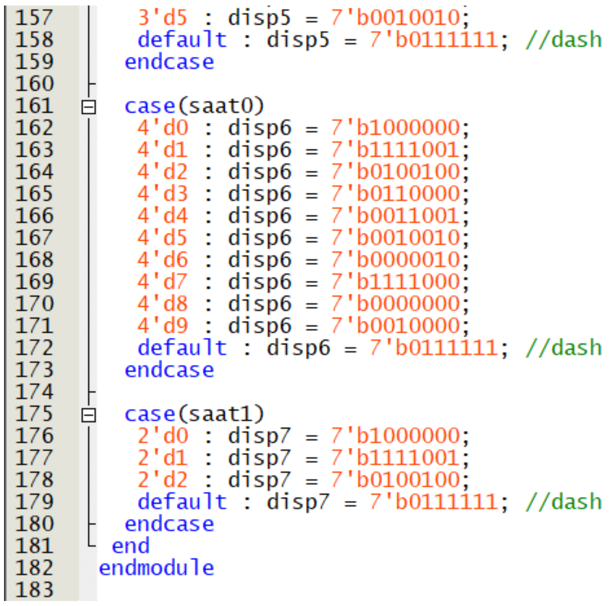
Verilog Code



```verilog
module verilog(input clk,reset,start,
               output reg [6:0] disp0,disp1,disp2,disp4,disp6,disp3,disp5,disp7);

reg [3:0] salise0,salise1,saniye0,dakika0,saat0;
reg [2:0] saniye1,dakika1;
reg [1:0] saat1;

always @(posedge clk or posedge reset)
begin
   if(reset)
   begin
      salise0 <=0;
      salise1 <=0;
      saniye0 <=0;
      saniye1 <=0;
      dakika0 <=0;
      dakika1 <=0;
      saat0   <=0;
      saat1   <=0;
   end

   else if(clk)
   begin
      if(salise0 == 9) //xx xx xxx9
      begin
      salise0 <=0;
         if(salise1 == 9)//xx xx xx99
         begin
         salise1 <=0;
            if(saniye0 == 9)//xx xx x999
            begin
            saniye0 <=0;
               if(saniye1 == 5)//xx xx 5999
               begin
               saniye1 <=0;
                  if(dakika0 == 9)//xx x9 5999
                  begin
                  dakika0 <=0;
                     if(dakika1 == 5)//xx 59 5999
                     begin
                     dakika1 <=0;
                        if(saat0 == 9)//x9 59 5999
                        begin
                        saat0 <=0;
                           if(saat1 == 2 & saat0 == 3)//23 59 5999
                           begin
                              salise0 <=0;
                              salise1 <=0;
                              saniye0 <=0;
                              saniye1 <=0;
                              dakika0 <=0;
                              dakika1 <=0;
                              saat0   <=0;
                              saat1   <=0;
                           end
                           else
                           saat1 = saat1 + 1;
                        end
                        else
                        saat0 = saat0 + 1;
                     end
                     else
                     dakika1 = dakika1 + 1;
                  end
                  else
                  dakika0 = dakika0 + 1;
               end
               else
               saniye1 = saniye1 + 1;
            end
            else
            saniye0 = saniye0 + 1;
         end
         else
         salise1 = salise1 + 1;
      end
      else
      salise0 = salise0 + 1;
```



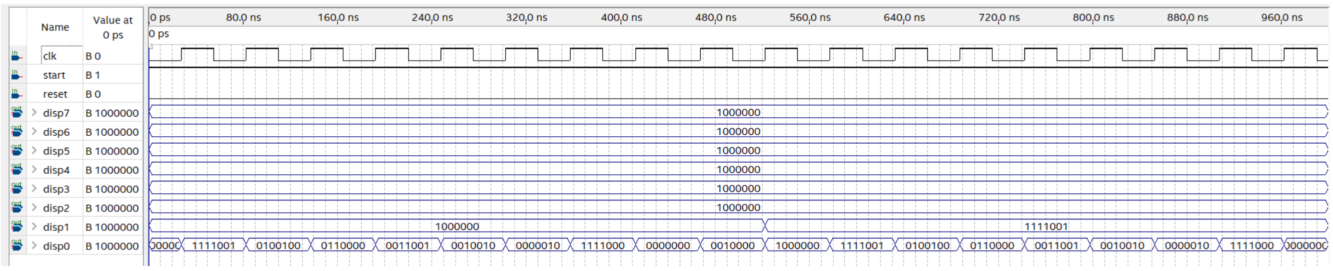
```verilog
   end
end


always @ (*) //send number
 begin
  case(salise0)
   4'd0 : disp0 = 7'b1000000;
   4'd1 : disp0 = 7'b1111001;
   4'd2 : disp0 = 7'b0100100;
   4'd3 : disp0 = 7'b0110000;
   4'd4 : disp0 = 7'b0011001;
   4'd5 : disp0 = 7'b0010010;
   4'd6 : disp0 = 7'b0000010;
   4'd7 : disp0 = 7'b1111000;
   4'd8 : disp0 = 7'b0000000;
   4'd9 : disp0 = 7'b0010000;
   default : disp0 <= 7'b0111111; //dash
  endcase

  case(salise1)
   4'd0 : disp1 = 7'b1000000;
   4'd1 : disp1 = 7'b1111001;
   4'd2 : disp1 = 7'b0100100;
   4'd3 : disp1 = 7'b0110000;
   4'd4 : disp1 = 7'b0011001;
   4'd5 : disp1 = 7'b0010010;
   4'd6 : disp1 = 7'b0000010;
   4'd7 : disp1 = 7'b1111000;
   4'd8 : disp1 = 7'b0000000;
   4'd9 : disp1 = 7'b0010000;
   default : disp1 = 7'b0111111; //dash
  endcase

  case(saniye0)
   4'd0 : disp2 = 7'b1000000;
   4'd1 : disp2 = 7'b1111001;
   4'd2 : disp2 = 7'b0100100;
   4'd3 : disp2 = 7'b0110000;
   4'd4 : disp2 = 7'b0011001;
   4'd5 : disp2 = 7'b0010010;
   4'd6 : disp2 = 7'b0000010;
   4'd7 : disp2 = 7'b1111000;
   4'd8 : disp2 = 7'b0000000;
   4'd9 : disp2 = 7'b0010000;
   default : disp2 = 7'b0111111; //dash
  endcase

  case(saniye1)
   3'd0 : disp3 = 7'b1000000;
   3'd1 : disp3 = 7'b1111001;
   3'd2 : disp3 = 7'b0100100;
   3'd3 : disp3 = 7'b0110000;
   3'd4 : disp3 = 7'b0011001;
   3'd5 : disp3 = 7'b0010010;
   default : disp3 = 7'b0111111; //dash
  endcase

  case(dakika0)
   4'd0 : disp4 = 7'b1000000;
   4'd1 : disp4 = 7'b1111001;
   4'd2 : disp4 = 7'b0100100;
   4'd3 : disp4 = 7'b0110000;
   4'd4 : disp4 = 7'b0011001;
   4'd5 : disp4 = 7'b0010010;
   4'd6 : disp4 = 7'b0000010;
   4'd7 : disp4 = 7'b1111000;
   4'd8 : disp4 = 7'b0000000;
   4'd9 : disp4 = 7'b0010000;
   default : disp4 = 7'b0111111; //dash
  endcase

  case(dakika1)
   3'd0 : disp5 = 7'b1000000;
   3'd1 : disp5 = 7'b1111001;
   3'd2 : disp5 = 7'b0100100;
   3'd3 : disp5 = 7'b0110000;
   3'd4 : disp5 = 7'b0011001;
```



```verilog
     3'd5  : disp5 = 7'b0010010;
     default : disp5 = 7'b0111111; //dash
    endcase

    case(saat0)
     4'd0 : disp6 = 7'b1000000;
     4'd1 : disp6 = 7'b1111001;
     4'd2 : disp6 = 7'b0100100;
     4'd3 : disp6 = 7'b0110000;
     4'd4 : disp6 = 7'b0011001;
     4'd5 : disp6 = 7'b0010010;
     4'd6 : disp6 = 7'b0000010;
     4'd7 : disp6 = 7'b1111000;
     4'd8 : disp6 = 7'b0000000;
     4'd9 : disp6 = 7'b0010000;
     default : disp6 = 7'b0111111; //dash
    endcase

    case(saat1)
     2'd0 : disp7 = 7'b1000000;
     2'd1 : disp7 = 7'b1111001;
     2'd2 : disp7 = 7'b0100100;
     default : disp7 = 7'b0111111; //dash
    endcase
   end
endmodule
```

Waveform

We can see milisecond count in this picture.



Introduction

In this project I did clock and this clock has hour, minute, second, milisecond. Clocks has a big importance in our life. I used two different methods for doing this project. I learned these methods in FPGA course. First method is frequence converting, second method is Code to Block diagram converting.

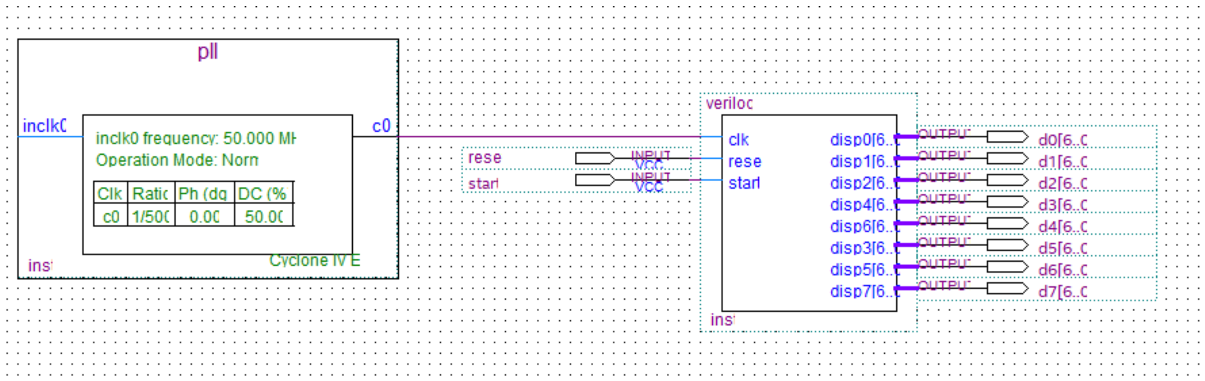
1. Frequency Converting

Our normal clock input is 50 MHz. With this method we can get the clock output we want. Block Diagram clock input is 100 Hz. This is equal milisecond count. Ve can see stages in Fig-1 and Fig-2 and make settings. If we want, we can add poly output of the converter.

2. Code To Block Diagram Converting

First we should write code after that we can convert. Our code is ready, we choose create symbol files for current file. Then we wait for creating after that its ready in symbol files. We can see stages in Fig-3 and Fig-4.

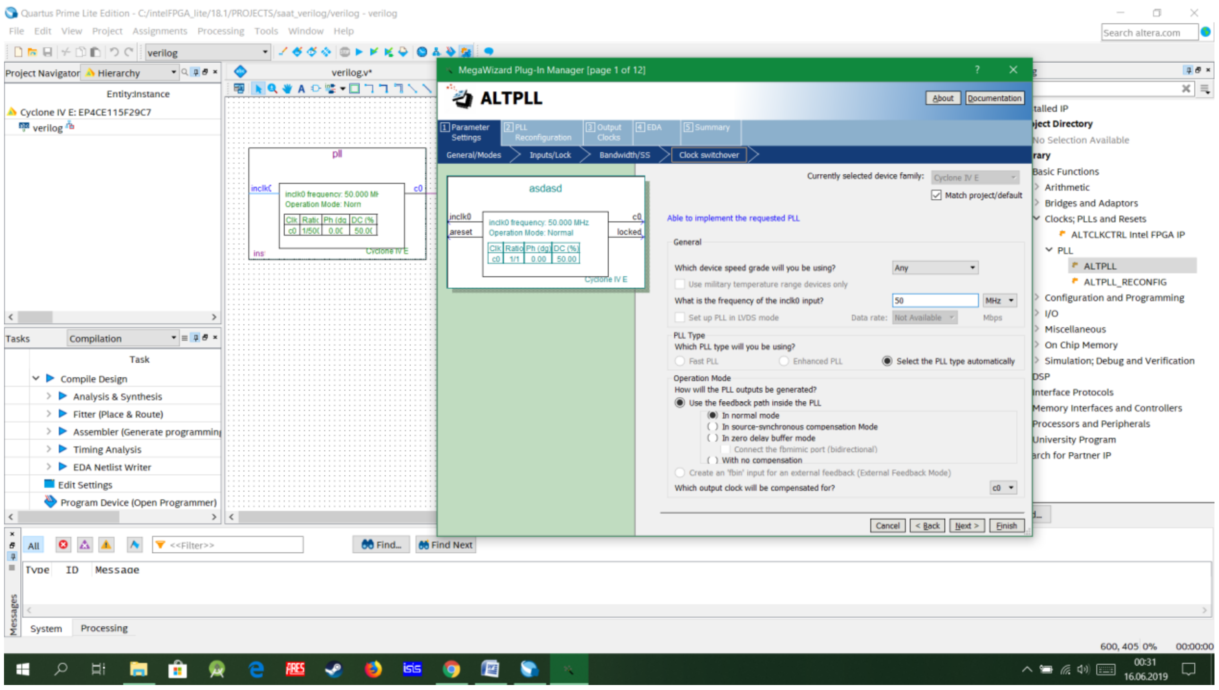
Block Diagram



Important Things about Verilog Language

We should add input and output in module brackets. Then we use reg for registration. We use always for main menu, if we put something in brackets it will depend on that or we put (*) this symbol always running always. And every always running at the same time. If and case comments are similar with C language.

Stages



MegaWizard Plug-In Manager [page 1 of 12]

ALTPLL

Currently selected device family: Cyclone IV E
☑ Match project/default

Able to implement the requested PLL

General
Which device speed grade will you be using? Any
☐ Use military temperature range devices only
What is the frequency of the inclk0 input? 50 MHz
☐ Set up PLL in LVDS mode Data rate: Not Available Mbps

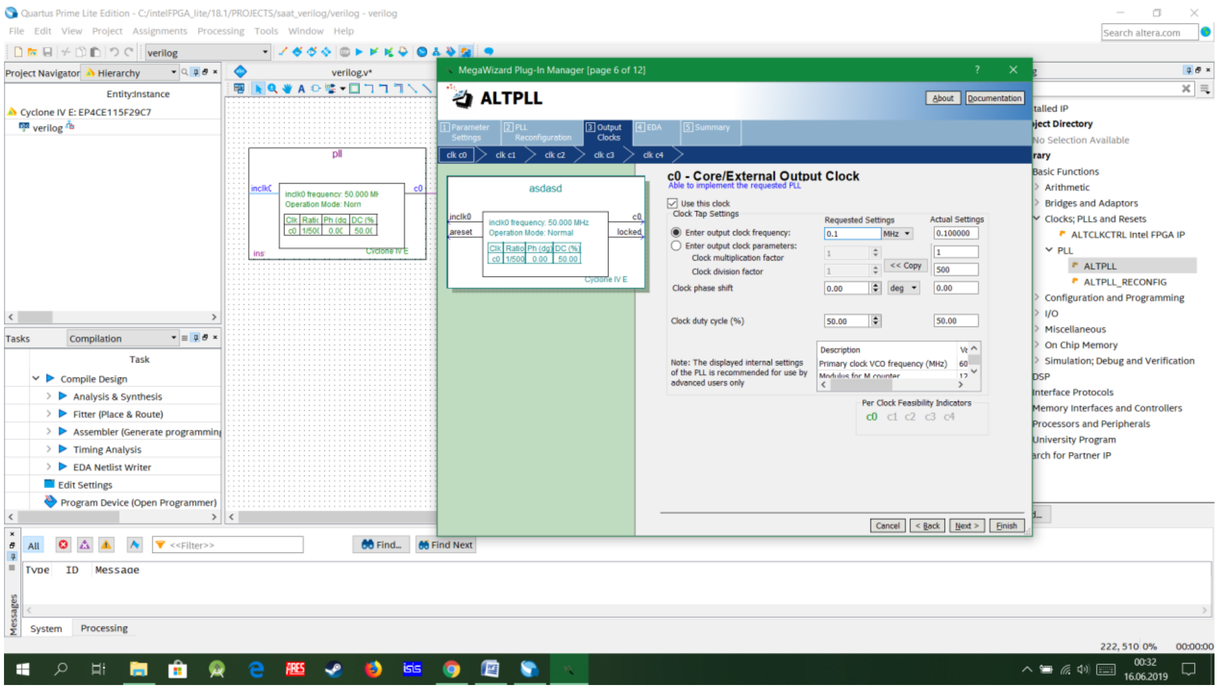
PLL Type
Which PLL type will you be using?
○ Fast PLL ○ Enhanced PLL ● Select the PLL type automatically

Operation Mode
How will the PLL outputs be generated?
● Use the feedback path inside the PLL
● In normal mode
( ) In source-synchronous compensation Mode
( ) In zero delay buffer mode
☐ Connect the fbmimic port (bidirectional)
( ) With no compensation
○ Create an 'fbin' input for an external feedback (External Feedback Mode)
Which output clock will be compensated for? c0

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Fig-1 Frequance converting 50 MHz to 100 Hz



MegaWizard Plug-In Manager [page 6 of 12]

ALTPLL

c0 - Core/External Output Clock
Able to implement the requested PLL

☑ Use this clock
Clock Tap Settings

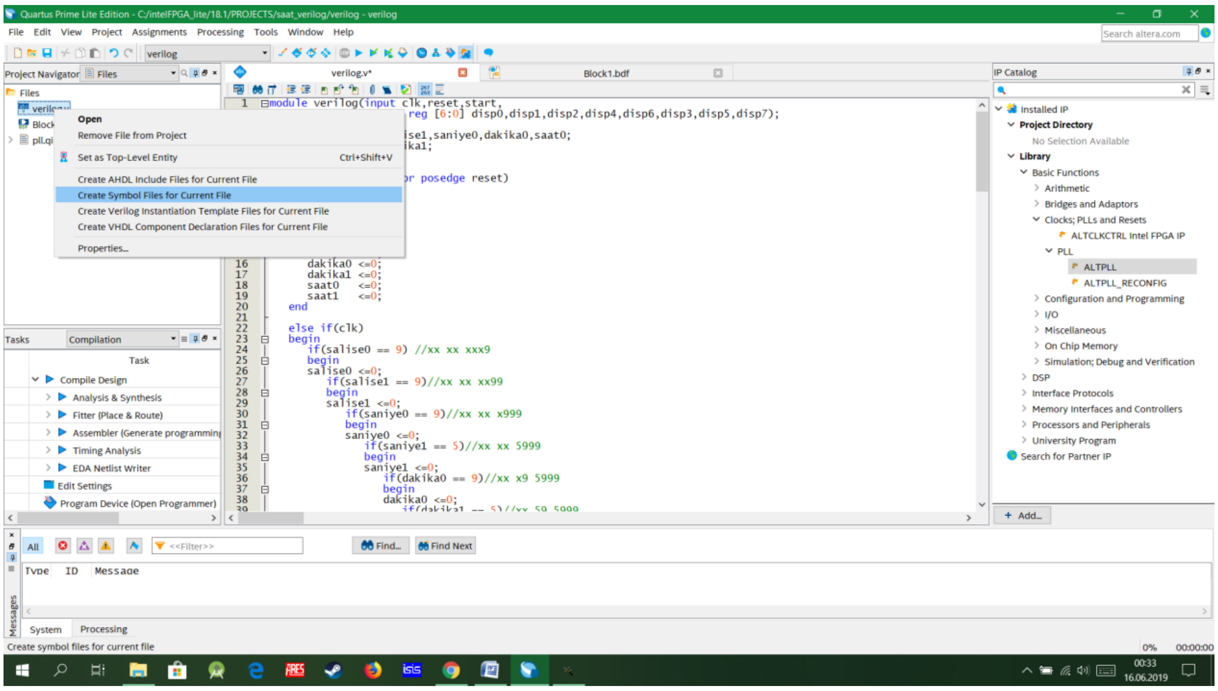
	Requested Settings	Actual Settings
● Enter output clock frequency:	0.1 MHz	0.100000
○ Enter output clock parameters:		
Clock multiplication factor	1	1
Clock division factor	1	500
Clock phase shift	0.00 deg	0.00
Clock duty cycle (%)	50.00	50.00

Note: The displayed internal settings of the PLL is recommended for use by advanced users only

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/500	0.00	50.00

Fig-2



```
module verilog(input clk,reset,start,
                    reg [6:0] disp0,disp1,disp2,disp4,disp6,disp3,disp5,disp7);

                    isel,saniye0,dakika0,saat0;
                    ika1;

                    or posedge reset)

        dakika0 <=0;
        dakika1 <=0;
        saat0   <=0;
        saat1   <=0;
    end

    else if(clk)
    begin
        if(salise0 == 9) //xx xx xxx9
        begin
        salise0 <=0;
            if(salise1 == 9)//xx xx xx99
            begin
            salise1 <=0;
                if(saniye0 == 9)//xx xx x999
                begin
                saniye0 <=0;
                    if(saniye1 == 5)//xx xx 5999
                    begin
                    saniye1 <=0;
                        if(dakika0 == 9)//xx x9 5999
                        begin
                        dakika0 <=0;
```

```
Type  ID  Message
```

Fig-3 Converting Verilog Code to Block Diagram

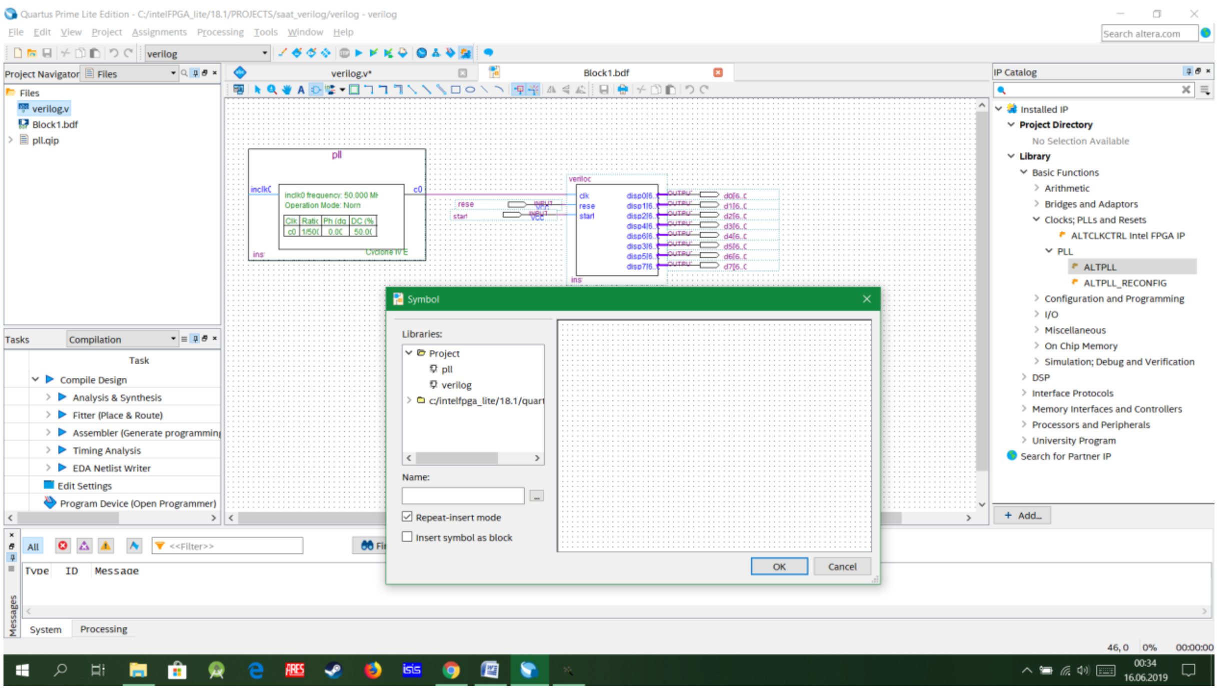


Fig-4 Find that Block Diagram in symbol tools