

CAD for VLSI Design - I

Quartus II Usage Manual

Quartus Usage Manual

Contents

1	Introduction	3
2	Graphical User Interface Design Flow	4
3	How to Start	4
4	Things that can be done without Board	14
A	4-bit Counter	15
A.1	counter.v	15
A.2	freq_dec.v	17
A.3	sevensseg.v	17
A.4	Pin Assignment	18
B	Counting the number of the ones in a given a 4-bit Vector	19
B.1	countones.v	19
B.2	sevensseg.v	20
B.3	Pin Assignment	21

1 Introduction

The Altera Quartus II design software provides a complete, multi-platform design environment that easily adapts to your specific design needs. It is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes solutions for all phases of FPGA and CPLD design (Figure 1).

In addition, Quartus II software allows you to use the Quartus II graphical user interface and command-line interface for each phase of the design flow. You can use one of these interfaces for the entire flow, or you can use different options at different phases.

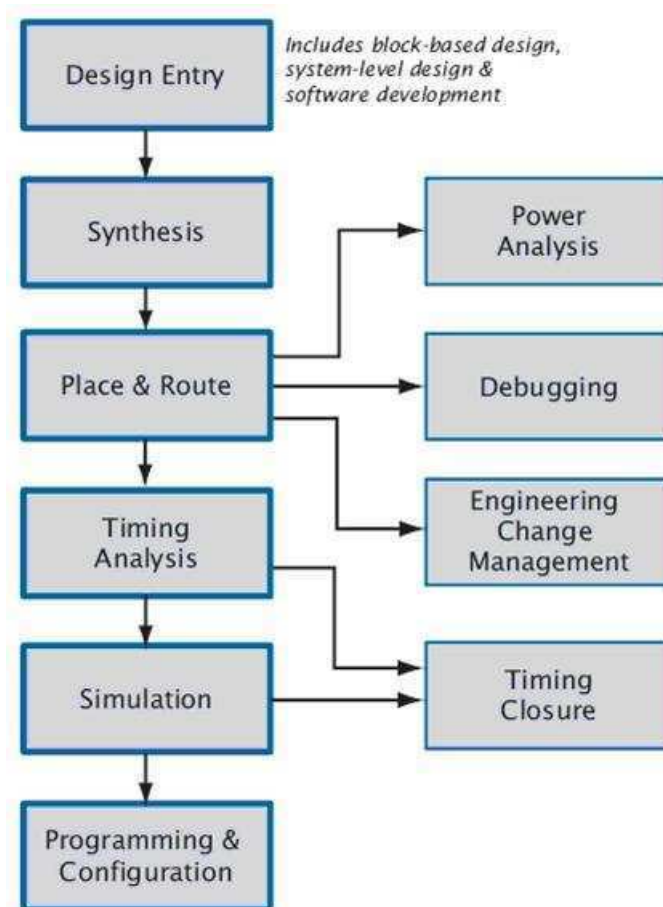


Figure 1: FPGA Design Flow

2 Graphical User Interface Design Flow

You can use Quartus II software to perform all stages of the design flow. The following steps describe the basic design flow for using the Quartus II graphical user interface:

- To create a new project and specify a target device or device family, on the File menu, click New Project Wizard.
- Use the Text Editor to create a Verilog HDL, VHDL, or Altera Hardware Description Language (AHDL) design. Use the Block Editor to create a block diagram with symbols that represent other design files, or to create a schematic.
- Synthesize the design with Analysis & Synthesis.
- (Optional) Generate a functional simulation netlist for your design and perform a functional simulation with the Simulator.
- Place and route the design with the Fitter.
- Perform a power estimation and analysis with the PowerPlay Power Analyzer.
- Use the Simulator to perform timing simulation for the design. Use the TimeQuest Timing Analyzer or the Classic Timing Analyzer to analyze the timing of your design.
- Create programming files for your design with the Assembler and program the device with the Programmer and Altera programming hardware.

3 How to Start

Quartus is installed in all Linux machines of RISE lab. You will be provided accounts in RISE Lab through which you can get access to Quartus. Also, Altera provides the basic software utilities (sans the bit file generator and the downloader) for free as Web Pack. You can download this and run it from your own computers. It is available from their website www.altera.com

1. Create a directory in your working directory and copy the .v files required for creating the project. It is advisable to create a separate project for each design and separate directories for each project. Now you have all the verilog files in a directory, say *dname* to run the project.

Invoke Quartus using the path `$>/tools/altera/quartus/bin/quartus &`

Explore the options available in the main menu such as File, Edit, View, Project, Assignments, Processing and Tools. The *Graphical User Interface Design Flow* helps you. Do not worry if you dont understand all the options fully. You will come to know about these when you create a new project and download the application into the FPGA.

The steps involved in creation of project, compilation, pin assignment and downloading into FPGA are described below with the screen-snapshots.

2. Open a new project using New Project Wizard in File menu. Then a wizard opens up as shown in Figure 2. You have to give your inputs for several options to create a new project using the New Project Wizard.

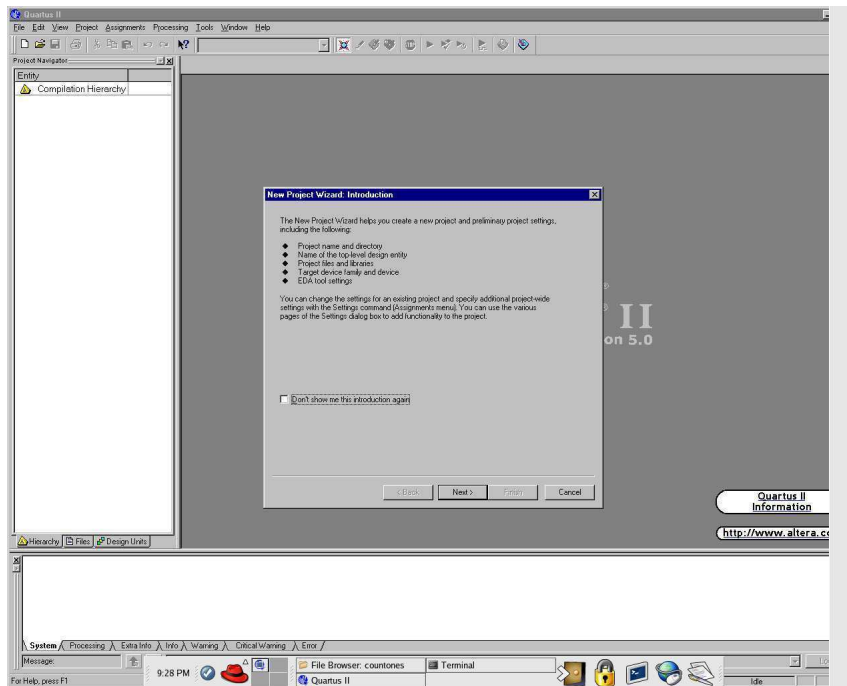


Figure 2: New Project Wizard Introduction

3. The window shown in Figure 3 is for specifying the working directory, project and the top-level design names. Make sure that the project name and the top-level design entry are the same.

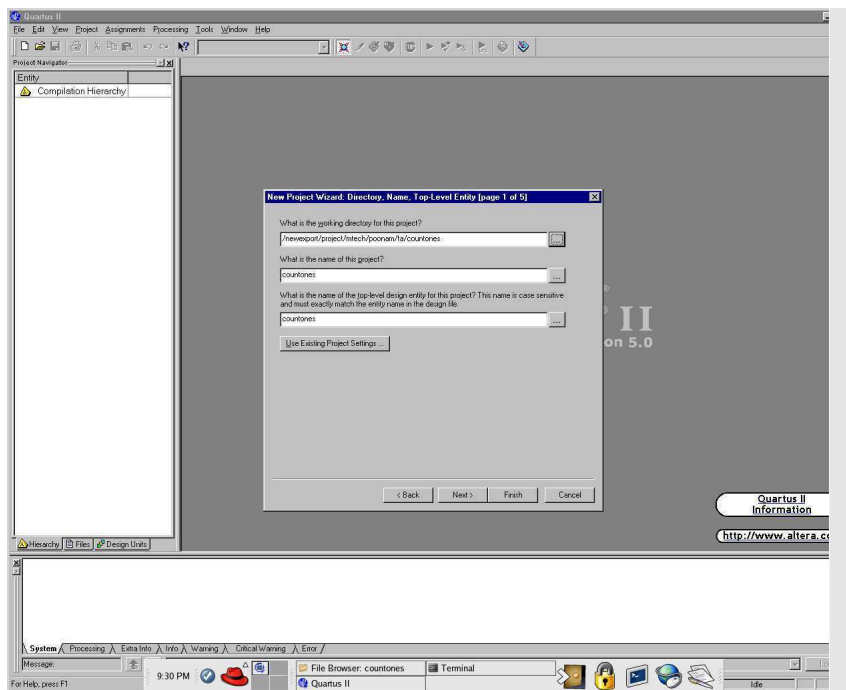


Figure 3: Specify Project name and Directory

Fill in the following:

- Project name and directory
 - Name of the top level design entity. Note that the project name and name of the top level entity should be the same name, i.e. the top-level module in among you Verilog source files.
4. Next you can add the existing files into the project you are creating as shown in Figure 4. You can browse for your folder and add the files and you can add/delete more files even after creating the new project.
 5. Specify the target family and device. This changes for different boards. For the NIOS Development Board that we are using in the class, select the following family and device as shown in Figure 5.

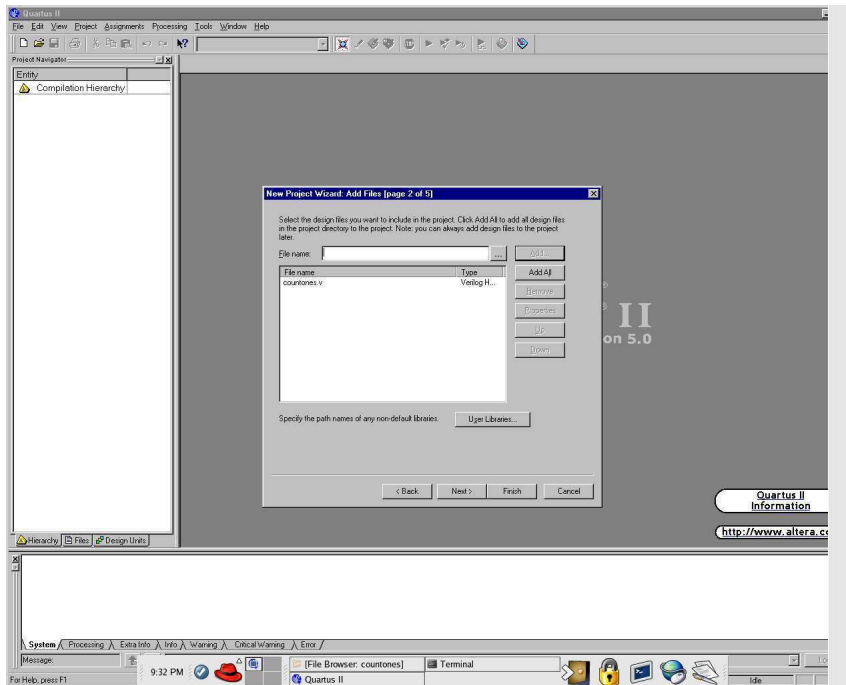


Figure 4: Add files

Family : APEX20KE Device : EP20K200EFC484-2X

After this, click next and finish the project creation by clicking finish. Now the new project is created.

You can actually see the .v files added in the project by going to Files Tab at the bottom as shown in Figure 6.

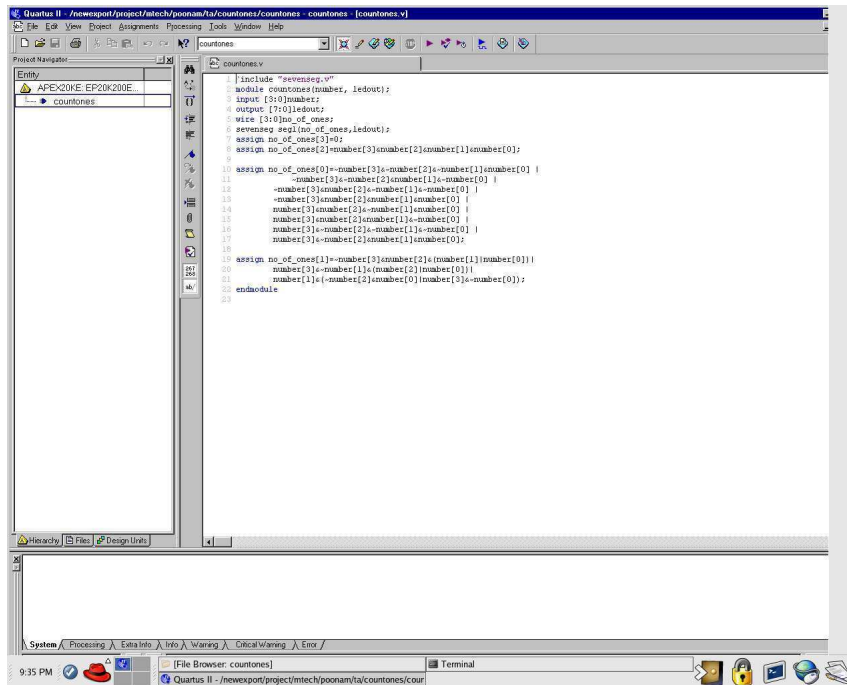


Figure 6: View your project using Quartus

the output pins in the design. Do the pin assignments as shown in Figure 7. Clock is assigned L6 (dedicated clock pin # in the board) and resetbar can be assigned to any of the DIP switches or push-buttons available in the board. Please refer ds_nios_board_apex_20k200e.pdf for mapping the input/output pins to the DIP/push-buttons/seven-segment LEDs.

Save the project after pin assignment and compile again. You should get a message Compilation Successful.

Now the design is ready for downloading onto the FPGA in the nios-board. To download the design into FPGA, open Programmer in Tools menu.

Select the Hardware set-up option and add Byte-blaster in the Programmer dialog-box as shown in Figure 10. Click the Program/Configure option.

Start down-loading the design by clicking on *Start* . You can see the progress-bar indicating that the design is getting downloaded into the FPGA as shown in the Figure 11. You can observe your project running on the board after the downloading is completed.

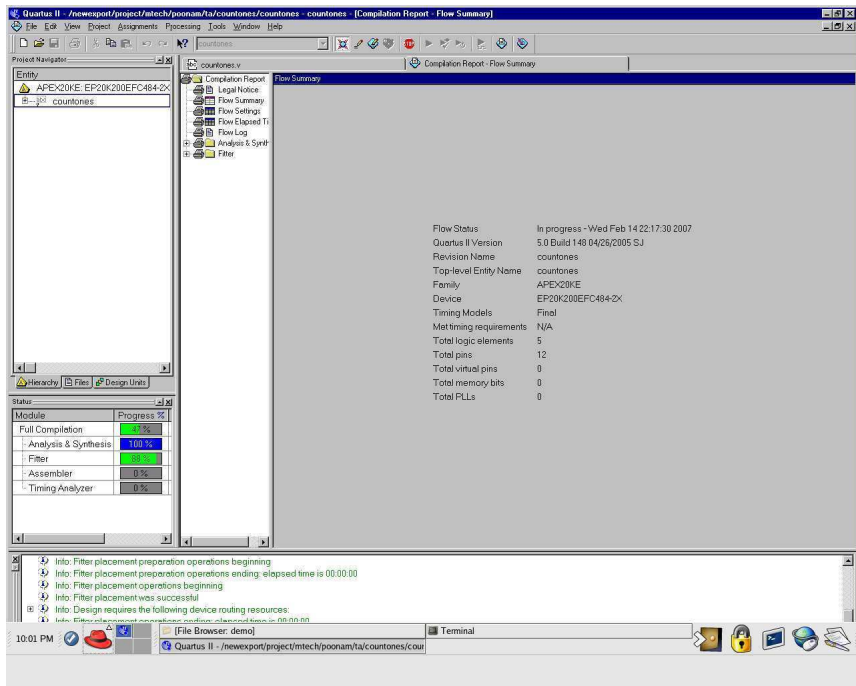


Figure 7: Compilation

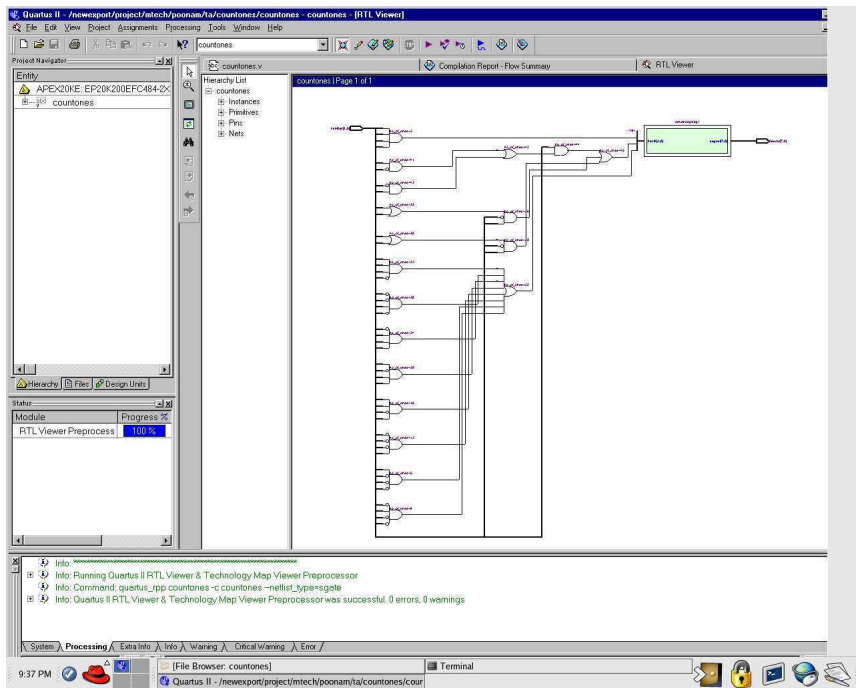


Figure 8: RTL View

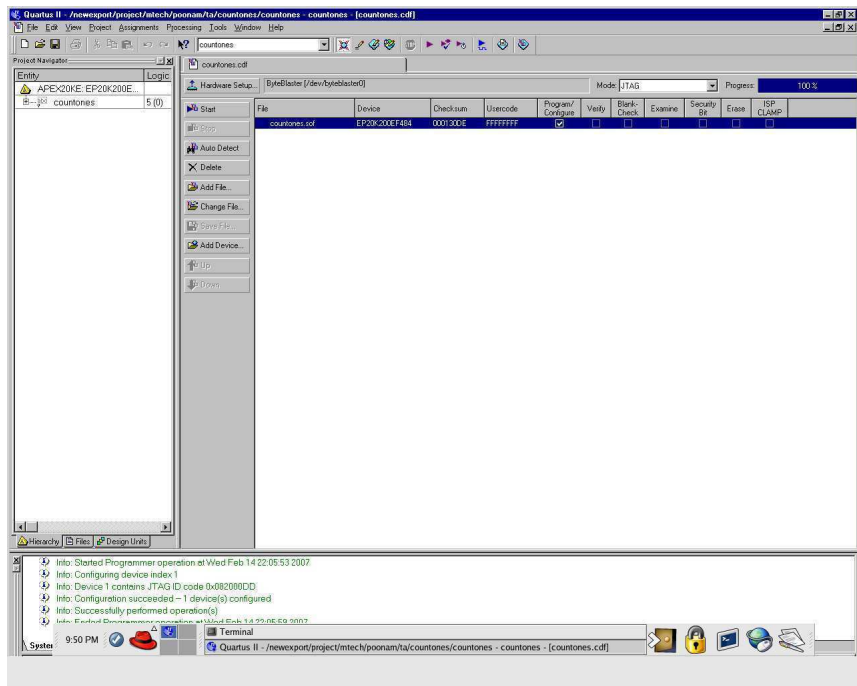


Figure 11: Burn onto the Board

4 Things that can be done without Board

Learning and using Quartus (Compilation and Pin Assignment) can be done without the nios-board. Students are instructed to do all the following steps without nios-board.

- Creating the project
- Settings for the project
- Pin Assignment
- Compilation
- RTL and Technology Map Viewing
- Final Compilation

Students can use the tool and do all the above steps without the board. Nios-board is required only for downloading i.e. burning the application onto the FPGA and see the output on seven segment display or LEDs.

A 4-bit Counter

This program is intended to count the numbers from 0 to 15 and display onto the FPGA board. The main module is counter.v. The Nios board runs at 33 MHz and to view the output of the counter, we bring down the frequency of display to 1 MHz by using a 25 bit counter. freq_dec.v does this. sevenseg.v program is used for displaying the output of the counter on the seven segment display of the FPGA.

A.1 counter.v

```
'include "freq_dec.v"
'include "sevenseg.v"

module counter (output [7:0]ledout, input loadenbar,
               input [3:0]load, input clk,
               input async_resetbar,input pause);

wire loaden,async_reset;
wire [24:0]count;
reg flag,reset;
reg [3:0] tmp;

assign loaden=~loadenbar;
assign async_reset=~async_resetbar;

frqcounter frq( clk, reset, count); // 25 bit counter
sevenseg seg1(tmp,ledout); // for display

always@(posedge clk)
    if(pause==0)
        flag<=~flag;

always @(posedge async_reset or posedge clk)
begin
```

```

//asynchronous reset
if (async_reset == 1'b1)
begin
    tmp <= {4{1'b0}};
    reset<=1'b1;
end

//synchronous load; Note that loaden is not in the
//sensitivity list

else if(loaden == 1'b1)
begin
    tmp <= load;
    reset<=1'b1;
end
else if (count=={25{1'b1}})
begin
    if(flag!=1)
        tmp <= tmp + 4'b001;
    else
        reset<=1'b0;
end
else
begin
    reset<=1'b0;
end

end

endmodule

```

A.2 freq_dec.v

```
module frqcounter(clk, reset, count);
input clk ;
input reset ;
output [24:0]count; //divide 33 MHz by 33M to get 1 Hz display
reg [24:0] tmp;

always @( posedge clk)
if (reset == 1'b1)
    tmp = {25{1'b0}};
else
    tmp = tmp + 25'b0000_0000_0000_0000_0000_00001;

assign count = tmp;

endmodule
```

A.3 sevenseg.v

```
module sevenseg (hexin, segout);
input [3:0] hexin;
output reg [7:0] segout;

always @(hexin)
begin
    case (hexin[3:0])
        4'b0000 : segout = 8'b1100_0000 ; //to display 0
        4'b0001 : segout = 8'b1111_1001 ; //to display 1
        4'b0010 : segout = 8'b1010_0100 ; //to display 2
        4'b0011 : segout = 8'b1011_0000 ; //to display 3
        4'b0100 : segout = 8'b1001_1001 ; //to display 4
        4'b0101 : segout = 8'b1001_0010 ; //to display 5
```

```
4'b0110 : segout = 8'b1000_0010 ; //to display 6
4'b0111 : segout = 8'b1111_1000 ; //to display 7
4'b1000 : segout = 8'b1000_0000 ; //to display 8
4'b1001 : segout = 8'b1001_1000 ; //to display 9
4'b1010 : segout = 8'b1000_1000 ; //to display A
4'b1011 : segout = 8'b1000_0011 ; //to display B
4'b1100 : segout = 8'b1100_0110 ; //to display C
4'b1101 : segout = 8'b1010_0001 ; //to display D
4'b1110 : segout = 8'b1000_0110 ; //to display E
4'b1111 : segout = 8'b1000_1110 ; //to display F
default : segout = 8'b1111_1111 ; //to blank the LED
endcase
end
endmodule
```

A.4 Pin Assignment

The pin assignment for the counter program is as shown in Table 1.

Input/Output	Port	Pin
Input	clk	L6
Input	async_resetbar	W9
Input	loadenbar	T9
Input	pause	T8
Input	load[0]	V9
Input	load[1]	U9
Input	load[2]	T10
Input	load[3]	U10
Output	ledout[0]	R10
Output	ledout[1]	T11
Output	ledout[2]	U8
Output	ledout[3]	W18
Output	ledout[4]	Y18
Output	ledout[5]	U18
Output	ledout[6]	W17
Output	ledout[7]	C18

Table 1: Pin Assignment for Counter

B Counting the number of the ones in a given a 4-bit Vector

countones.v counts the number of ones in a given 4-bit vector. sevenseg.v program is used for displaying the output of the counter on the seven segment display of the FPGA.

B.1 countones.v

```
'include "sevenseg.v"

module countones(number, ledout);
input [3:0]number;
output [7:0]ledout;
```

```

reg [3:0]no_of_ones;
sevenseg seg1(no_of_ones,ledout);

always @(number)
begin
    case(number)
        4'd0 : no_of_ones <= 4'b0000;

        4'd3 : no_of_ones <= 4'b0010;
        4'd5 : no_of_ones <= 4'b0010;
        4'd6 : no_of_ones <= 4'b0010;
        4'd9 : no_of_ones <= 4'b0010;
        4'd10 : no_of_ones <= 4'b0010;
        4'd12 : no_of_ones <= 4'b0010;

        4'd1 : no_of_ones <= 4'b0001;
        4'd2 : no_of_ones <= 4'b0001;
        4'd4 : no_of_ones <= 4'b0001;
        4'd8 : no_of_ones <= 4'b0001;

        4'd7 : no_of_ones <= 4'b0011;
        4'd11 : no_of_ones <= 4'b0011;
        4'd13 : no_of_ones <= 4'b0011;
        4'd14 : no_of_ones <= 4'b0011;

        4'd15 : no_of_ones <= 4'b0100;
        default : no_of_ones <= 4'b0000;
    endcase
end
endmodule

```

B.2 sevenseg.v

The sevenseg.v program is same as the one used in the counter7vim program.

B.3 Pin Assignment

The pin assignment for the counter program is as shown in Table 2.

Input/Output	Port	Pin
Input	number[0]	V9
Input	number[1]	U9
Input	number[2]	T10
Input	number[3]	U10
Output	ledout[0]	R10
Output	ledout[1]	T11
Output	ledout[2]	U8
Output	ledout[3]	W18
Output	ledout[4]	Y18
Output	ledout[5]	U18
Output	ledout[6]	W17
Output	ledout[7]	C18

Table 2: Pin Assignment for Counting the number of 1's