ModelSim Verilog Tutorial

Terasic DE2-115 Board

Document Version 1.00 April 13, 2010 by Terasic

INTRODUCTION **1**

1.1 Before You Begin 1

1.2 DESIGN FLOW 2

1.3 GETTING HELP 4

GET STARTED **5**

2.1 CREATING A NEW PROJECT 6

2.2 ADDING ITEMS TO THE PROJECT 9

2.3 COMPILING THE FILES 18

SIMULATING A DESIGN **19**

3.1 STARTING SIMULATION 19

3.2 SIMULATION SETTING 22

CHAPTER

Introduction

This document introduces how to use Modelsim to simulate Verilog HDL designs, to improve your understanding; we will take the “LEDs sequencer” as an example to illustrate the whole process.

1.1 Before You Begin

Before you read this tutorial, setup your system to use the Modelsim ase 6.5 software as follows:

a) Install the following software.
   - Altera Quartus II 9.1
   - ModelSim Altera Starter Edition 6.5b

b) Verify that your system is properly configured. Consult the release notes and installation notes that came with your software package for more information.

Assumptions

We assume that you are familiar with the use of your operating system. You should also be familiar with the window management functions of your graphic interface: Microsoft Windows XP.

We also assume that you have a working knowledge of the language in which your design and/or testbench is written (i.e., VHDL, Verilog, etc.). Although ModelSim ase is an excellent tool to use while learning HDL concepts and practices, this tutorial is not written to achieve that goal.

1.2 Design Flow

Basic Simulation Flow

The diagram below illustrates the basic steps for simulating a design using ModelSim.

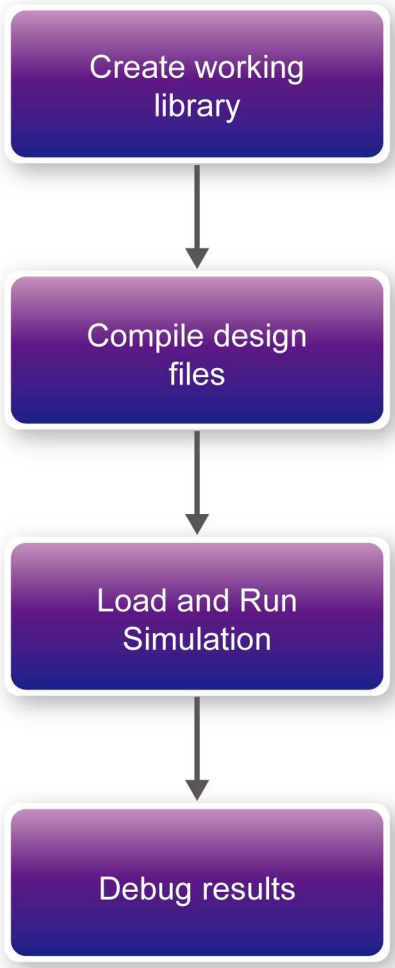

Figure 1.1 Simulation Flow

a. Creating the Working Library

In ModelSim, all designs are compiled into a library. You typically start a new simulation using ModelSim by creating a working library called "work". "Work" is the library name used by the compiler as the default destination for compiled design units.

b. Compiling Your Design

After creating the working library, you compile your design units into it. The ModelSim library format is compatible for all supported platforms. You can simulate your design on any platform without having to recompile your design.

c. Loading the Simulator with Your Design and Running the Simulation With the design compiled, you load

the simulator with your design by invoking the simulator on a top-level module (Verilog) or a configuration or entity/architecture pair (VHDL). Assuming the design loaded successfully, the simulation time is set to zero, and you enter a run command to begin simulation.

d. Debugging Your Results

If you don't get the results you expect, you can use ModelSim's robust debugging environment to track down the cause of the problems.

Project Flow

A project is a collection mechanism for an HDL design under specification or test. Even though you don't have to use projects in ModelSim, they may ease interaction with the tool and are useful for organizing files and specifying simulation settings.

The diagram below illustrates the basic steps for simulating a design within a ModelSim project.

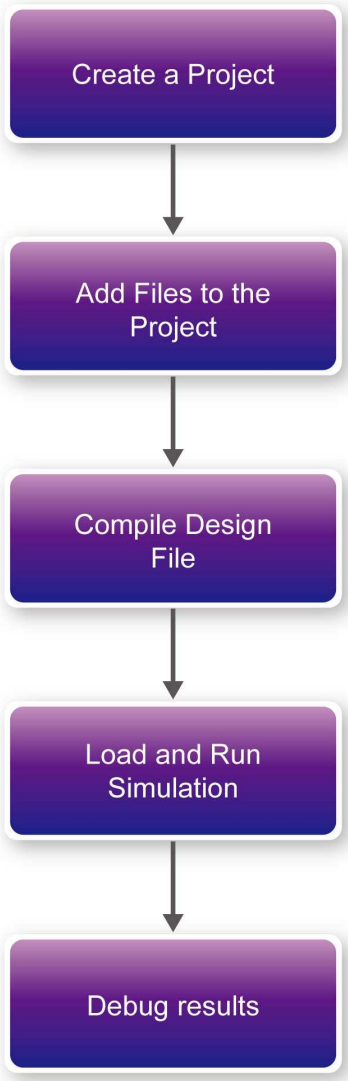

Figure 1.2 Project Flow

1.3 Getting Help

The following are a list of ways to contact us if you encounter any problems:

- ✓ E-mail :support@terasic.com
- ✓ Taiwan & China: +886-3-550-8800
- ✓ Korea : +82-2-512-7661
- ✓ Japan: +81-428-77-7000

CHAPTER

2. Get Started

This chapter describes the three basic steps to work with a project.

- **Step 1 — Creating a New Project**
- **Step 2 — Adding Items to the Project**

  Projects can refer to or contain source files, folders for organization, simulations and any other files you want to associate with them. You can also duplicate files into the project directory or simply create mappings to files in other locations.
- **Step 3 — Compiling the Files**

  This step checks syntax and semantics and creates the pseudo machine code which ModelSim needs for simulation.

2.1 Creating a New Project

Run the ModelSim software. The ModelSim Interface in Figure 2.1 will appear.

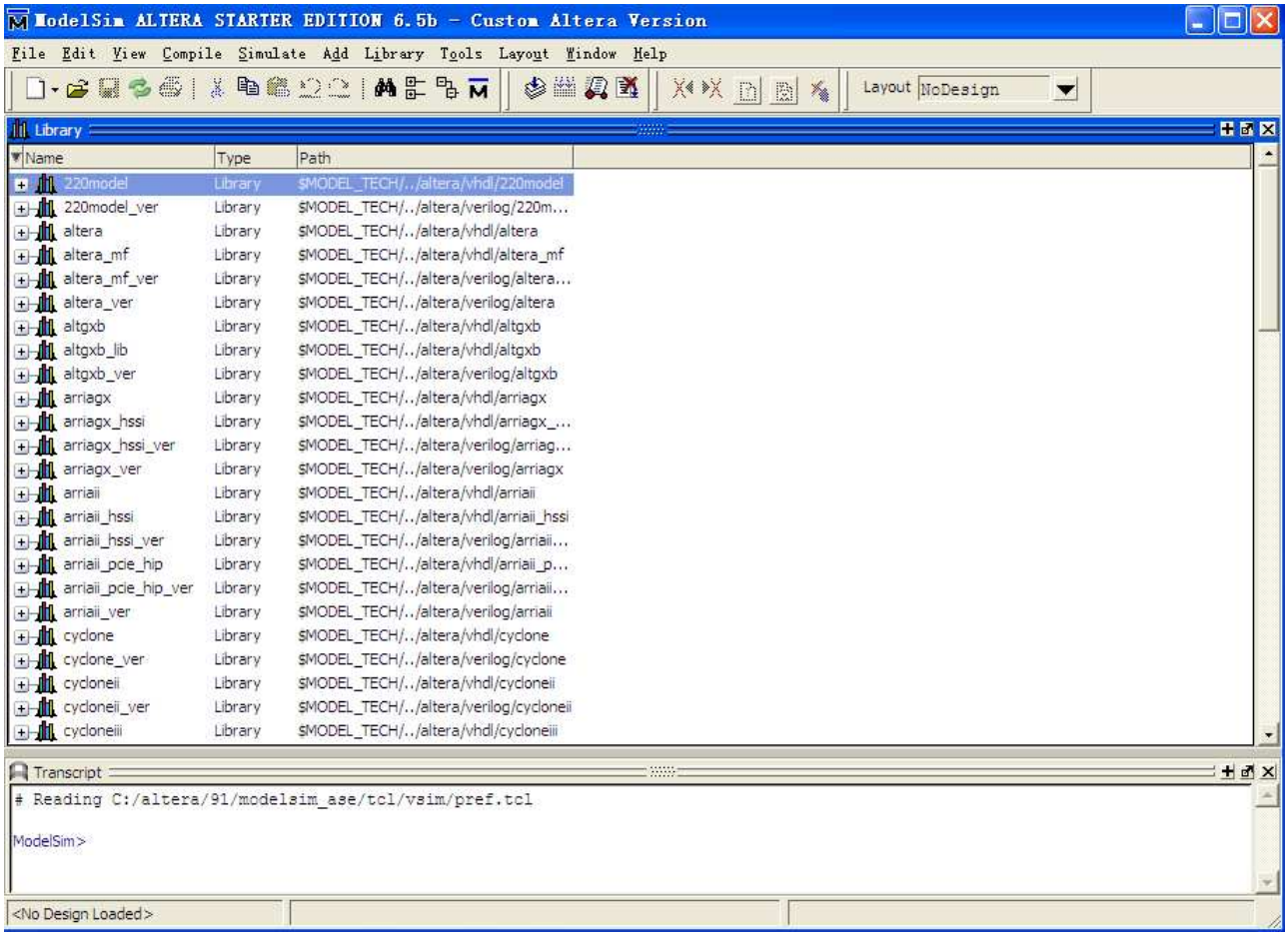

Figure 2.1 ModelSim Interface

1. Select **File > New > Project** to create a new project. This opens the **Create Project** dialog window where you can specify name, location, and default library name of a project. You can generally leave the **Default Library Name** set to "work." The name you specify will be used to create a working library subdirectory within the Project Location. This dialog window, shown in Figure 2.2, also allows you to refer library settings from a selected **.ini** file or copy them directly into the project.

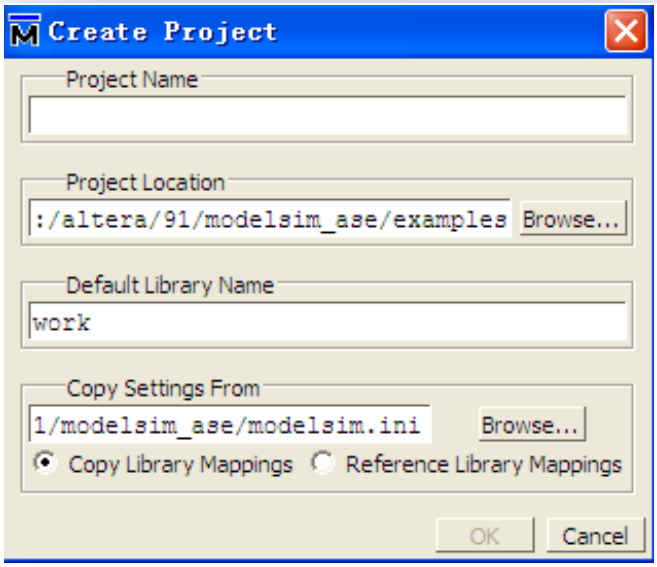

Figure 2.2 Create Project Dialog

1. Enter the following information about your project, shown in Figure 2.3:
   - **What is the name of this project?** Type `LED_FLOW`.
   - **What is the project location?** For example, C:/altera/91/modelsim_ase/examples.

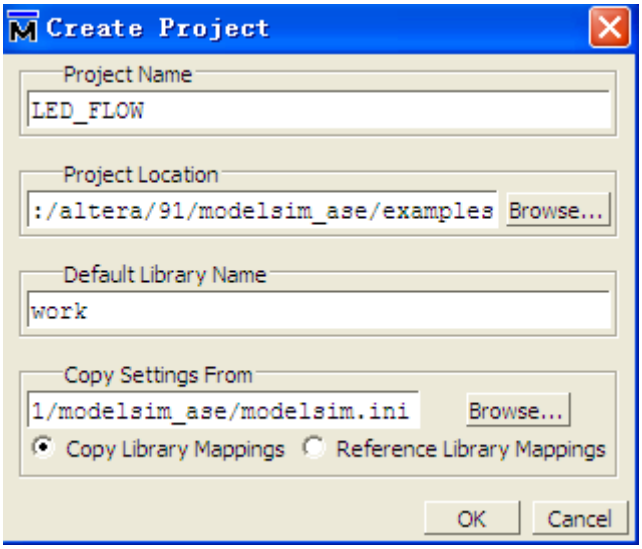

Figure 2.3 Input project information

After clicking the **OK** button, you will see a blank Project tab in the Workspace pane of the Main window, which shown in Figure 2.4.

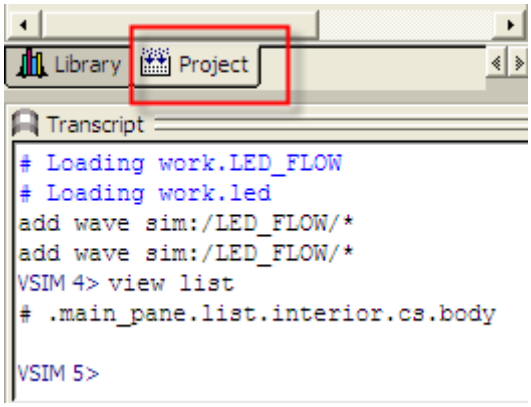

Figure 2.4 Project Tab in Workspace Pane

2. After that, you will observe the **Add Items to the Project** dialog window, shown in Figure 2.5.

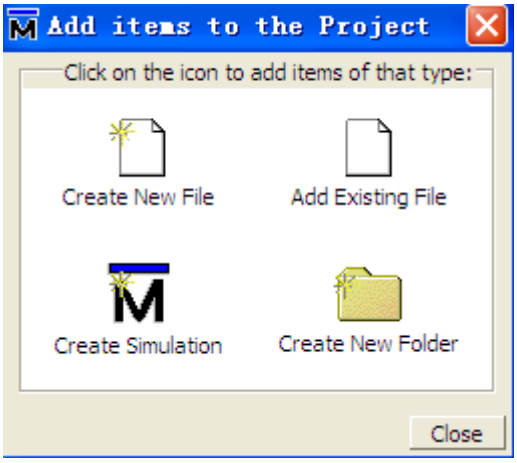

Figure 2.5 Add items to the Project

2.2 Adding Items to the Project

The **Add Items to the Project** dialog window includes these options:

- **Create New File** — Create a new VHDL, Verilog, Tcl, or text file using the Source editor.
- **Add Existing File** — Add an existing file.
- **Create Simulation**—Create a Simulation Configuration that specifies source files and simulator options.
- **Create New Folder** — Create an organization folder.

1. Click "**Create New File**". The window in Figure 2.6 will appear.

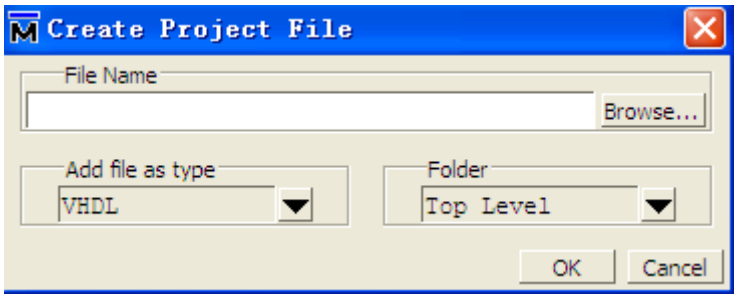

Figure 2.6 Create Project File

2. Input File Name: **LED_FLOW**, then Add file as type we choose **Verilog**.

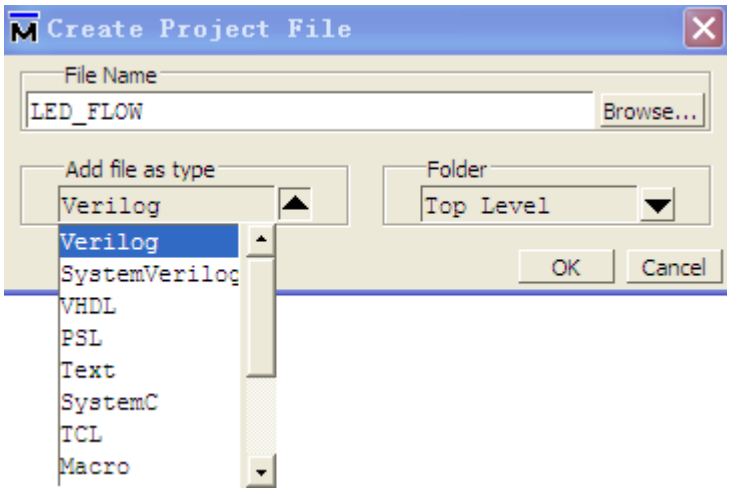

Figure 2.7 Input project file information

3. Click on the **OK** button, and then close the dialog. The new design file will appear in the project window. Click **Close** button to close **Add items to the Project** window.

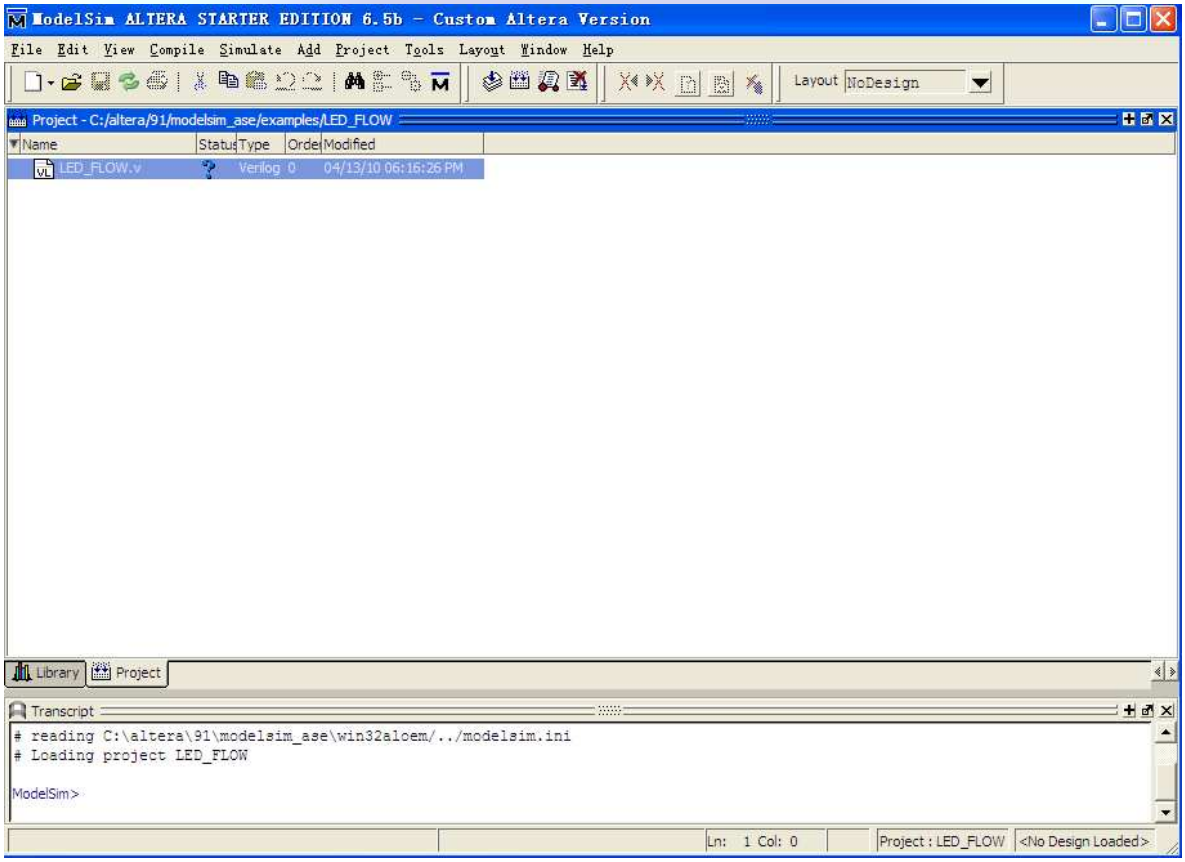

Figure 2.8 The new design file LED_FLOW.v

4. Double click **LED_FLOW.v** file.

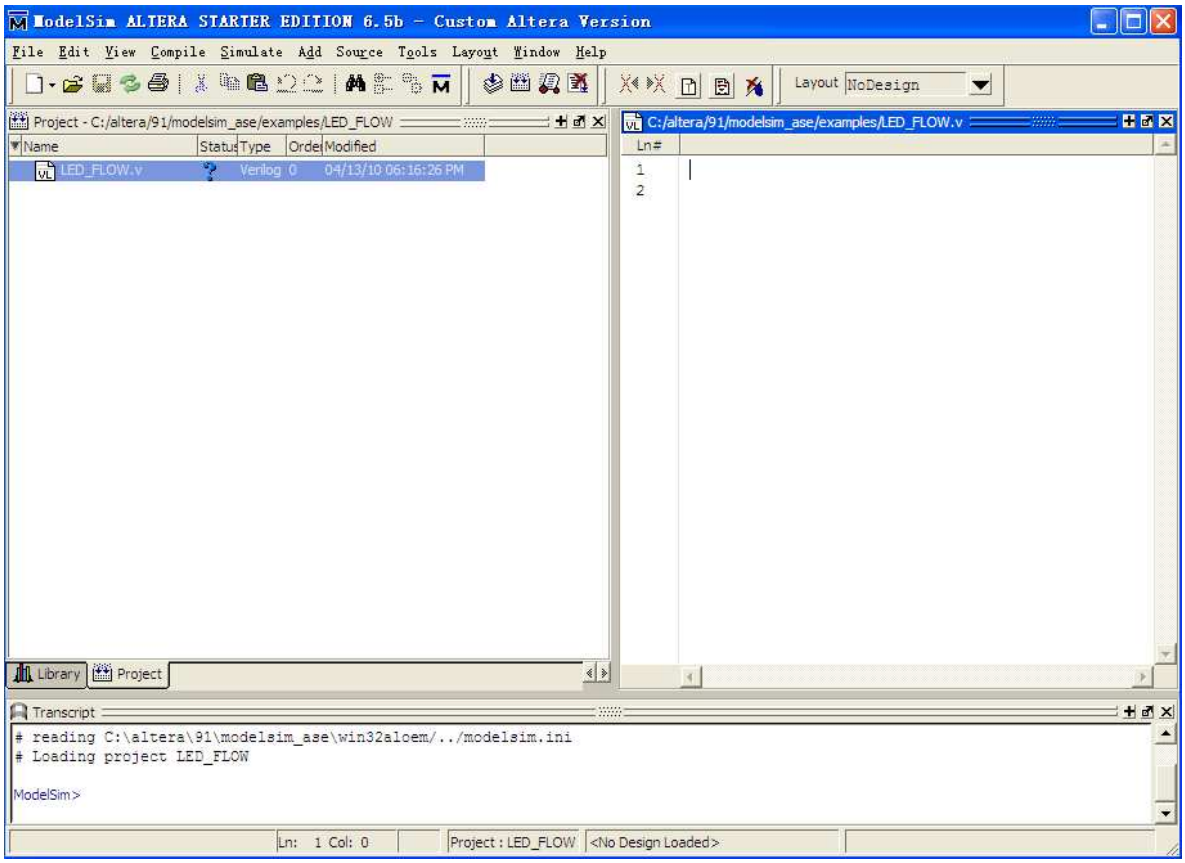

Figure 2.9 LED_FLOW.v input code window

In the LED_FLOW.v input code window, please input the testbench codes as follows:

```
`timescale 1ns/1ns
module LED_FLOW;
   reg              clk_50M;
   reg              reset;
   wire   [9:0]   led;
   led u1 (
                clk_50M,
                reset,
                led
              );
   initial
      begin
         clk_50M = 0;
         while(1)
            #10 clk_50M = ~clk_50M;
      end

   initial
      begin
         reset = 0;
         while(1)
         #10 reset = 1;
      end

   initial
         begin
               $display($time,,,"clk_50M=%d reset=%d led=%d",clk_50M,reset,led);
         end

endmodule
```

Inputting all the codes above to reach the window in Figure 2.10. Click **Save**.

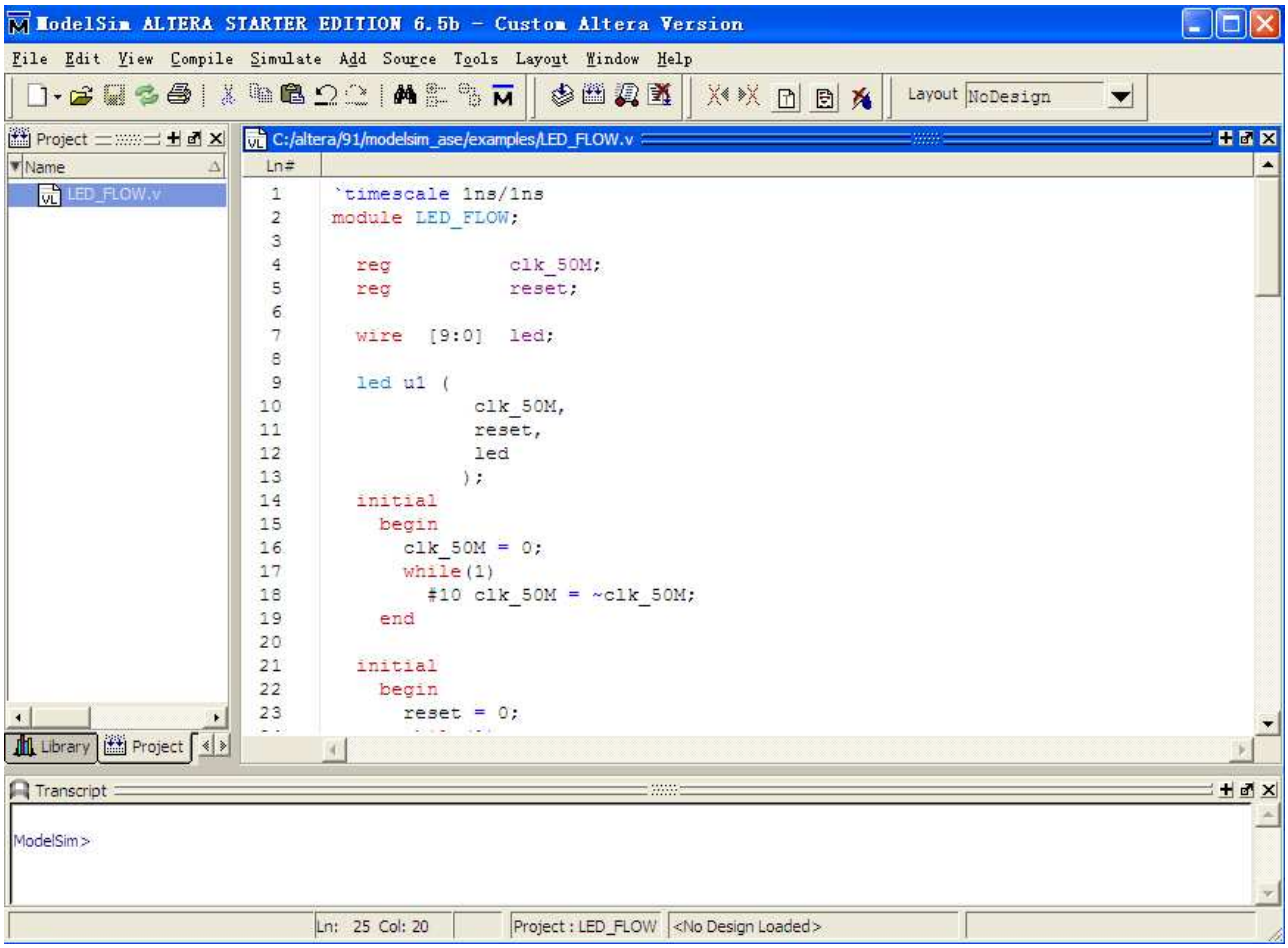

Figure 2.10 Input Testbench code

5. Select **File > New > Source > Verilog**, to create a new Verilog file, The window in Figure 2.11 will appear.

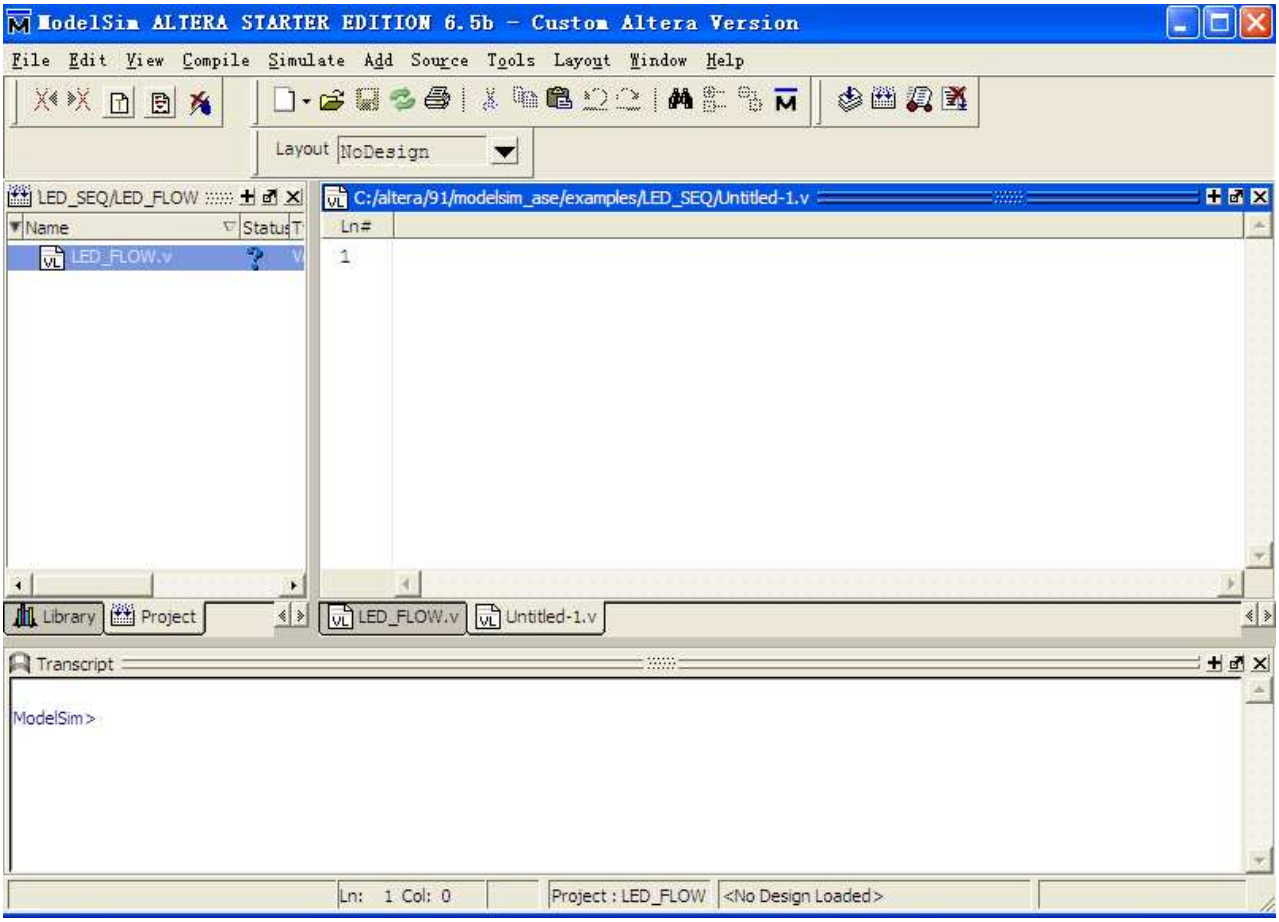

Figure 2.11 create a new Verilog file

6. In the new Verilog file window, please input the Verilog codes as show Figure 2.12.

```
`timescale 1ns/1ns
module led (
                clk_50M, // System clock 50MHz
                reset,     // System reset
                led         // led
               );


input               clk_50M;
input               reset;

output      [9:0]  led = 0;

// If you want to implement it on the DE2-115 board, please change counter[13:0] to counter[23:0]
reg         [13:0]   counter = 0;
reg         [3:0]    state = 0;
reg         [9:0]    led = 0;
```

```
always@(posedge clk_50M or negedge reset)
 begin
        if(!reset)
                begin
                        counter <= 0;
                end
        else begin
                        counter <= counter +1'b1;
                 end
end

// If you want to implement it on the DE2-115 board, please change counter[13] to counter[23]
always@(posedge counter[13])

 begin
        if(!reset)
                begin
                        state <= 0;
                end
        else begin
                        if(state == 4'b1001)
                                begin
                                        state <= 0;
                                end
                        else begin
                                        state <= state + 1'b1;
                                 end
                 end
 end

always@(posedge clk_50M or negedge reset)
 begin
        if(!reset)
                begin
                        led <= 0;
                end
        else begin
                        case(state)
                        4'b0000 : led <= 10'b0000000001;
                        4'b0001 : led <= 10'b0000000010;
```

```
                4'b0010 : led <= 10'b0000000100;
                4'b0011 : led <= 10'b0000001000;
                4'b0100 : led <= 10'b0000010000;
                4'b0101 : led <= 10'b0000100000;
                4'b0110 : led <= 10'b0001000000;
                4'b0111 : led <= 10'b0010000000;
                4'b1000 : led <= 10'b0100000000;
                4'b1001 : led <= 10'b1000000000;
                default : led <= 10'b0000000001;
                endcase
            end
    end

endmodule
```

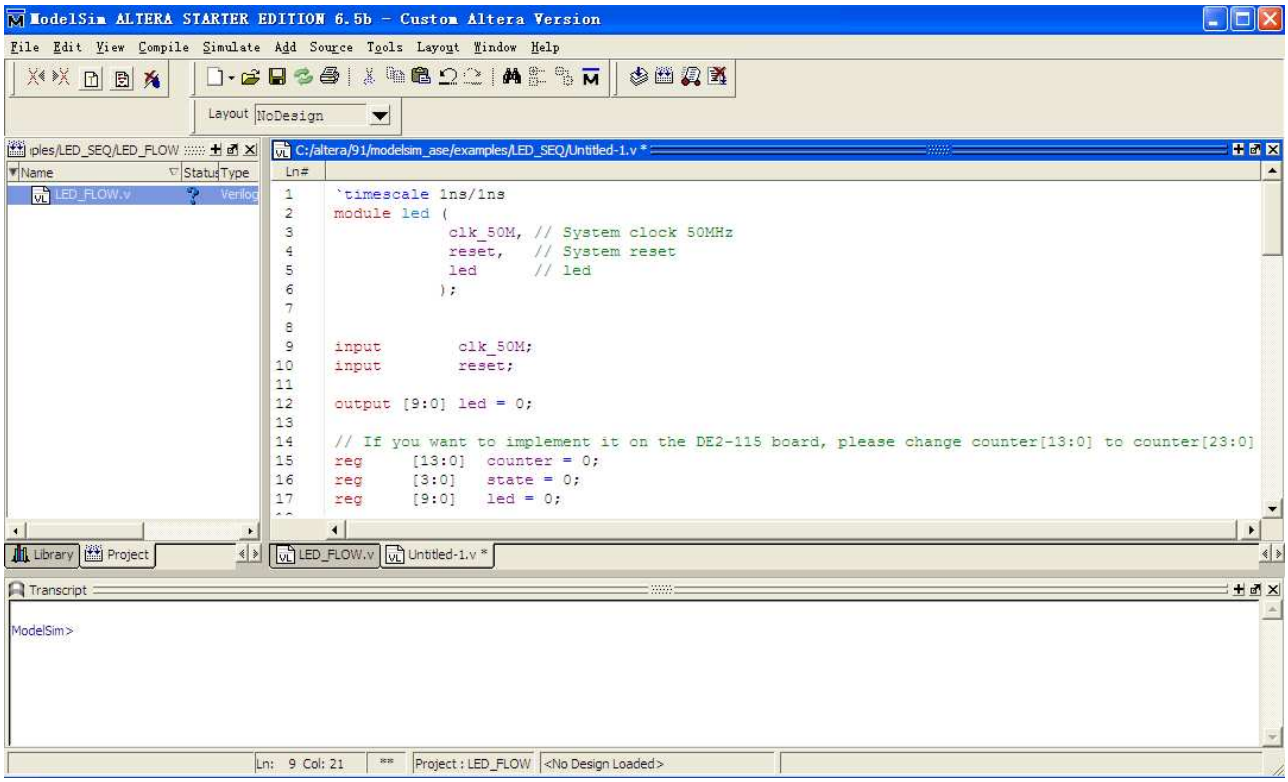

Figure 2.12 input the Verilog codes

7. Select **File > Save**, input the file name: led.v. Click **Save**. shown in Figure 2.13.

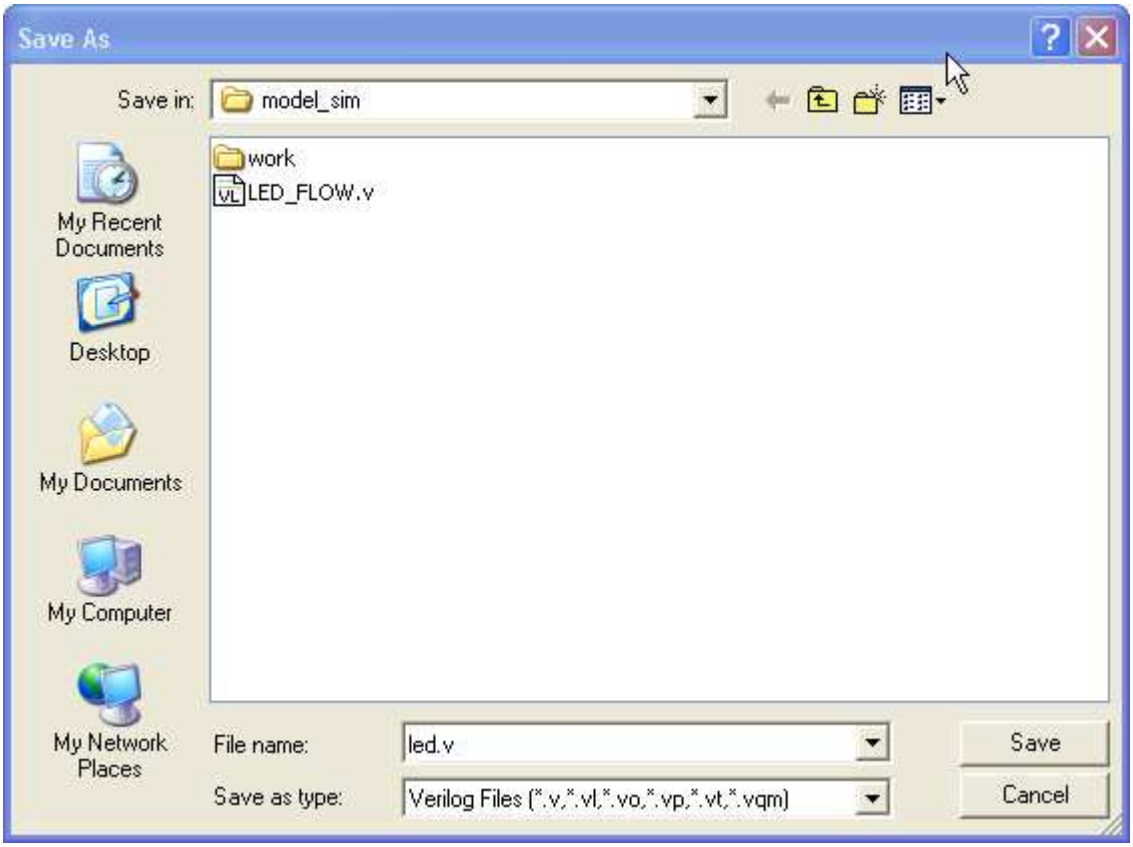

Figure 2.13. Save the led.v

8. Select **Project > Add to Project > Existing File**, shown in Figure 2.14.

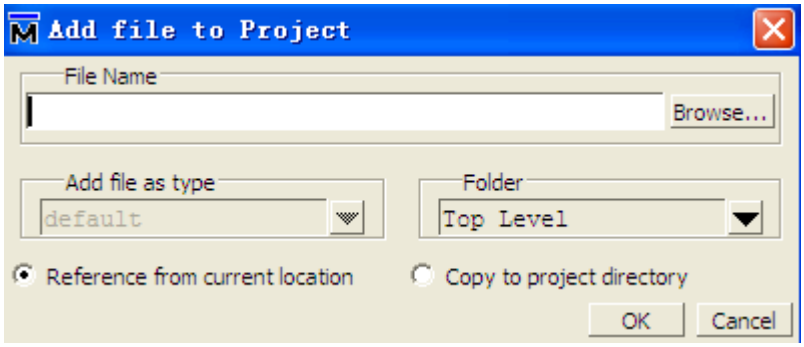

Figure 2.14 Add file to project

9. Click **Browse**, select the led.v, shown in Figure 2.15.

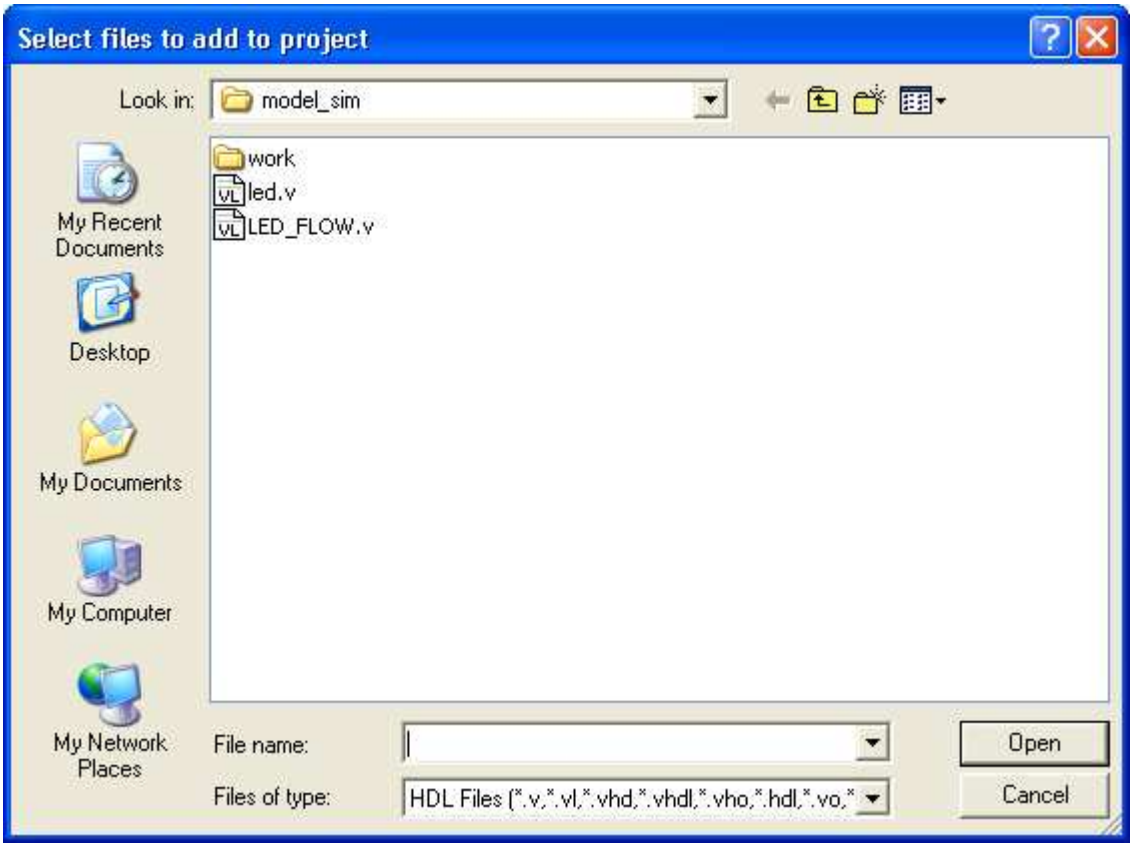

Figure 2.15 Select files to add to project

10. Click **Open**, on the **Add file to project** window click **OK**.

2.3 Compiling the Files

The question mark in the Status column in the Project tab means the files haven't been compiled into the project or the source has changed since the last compile. To compile the files, select **Compile > Compile All** or right click in the Project tab and select **Compile > Compile All**

1. If there is no error, the compile successful information (pointed by the red arrow), shown in Figure 2.16, will appear in the Transcript window.

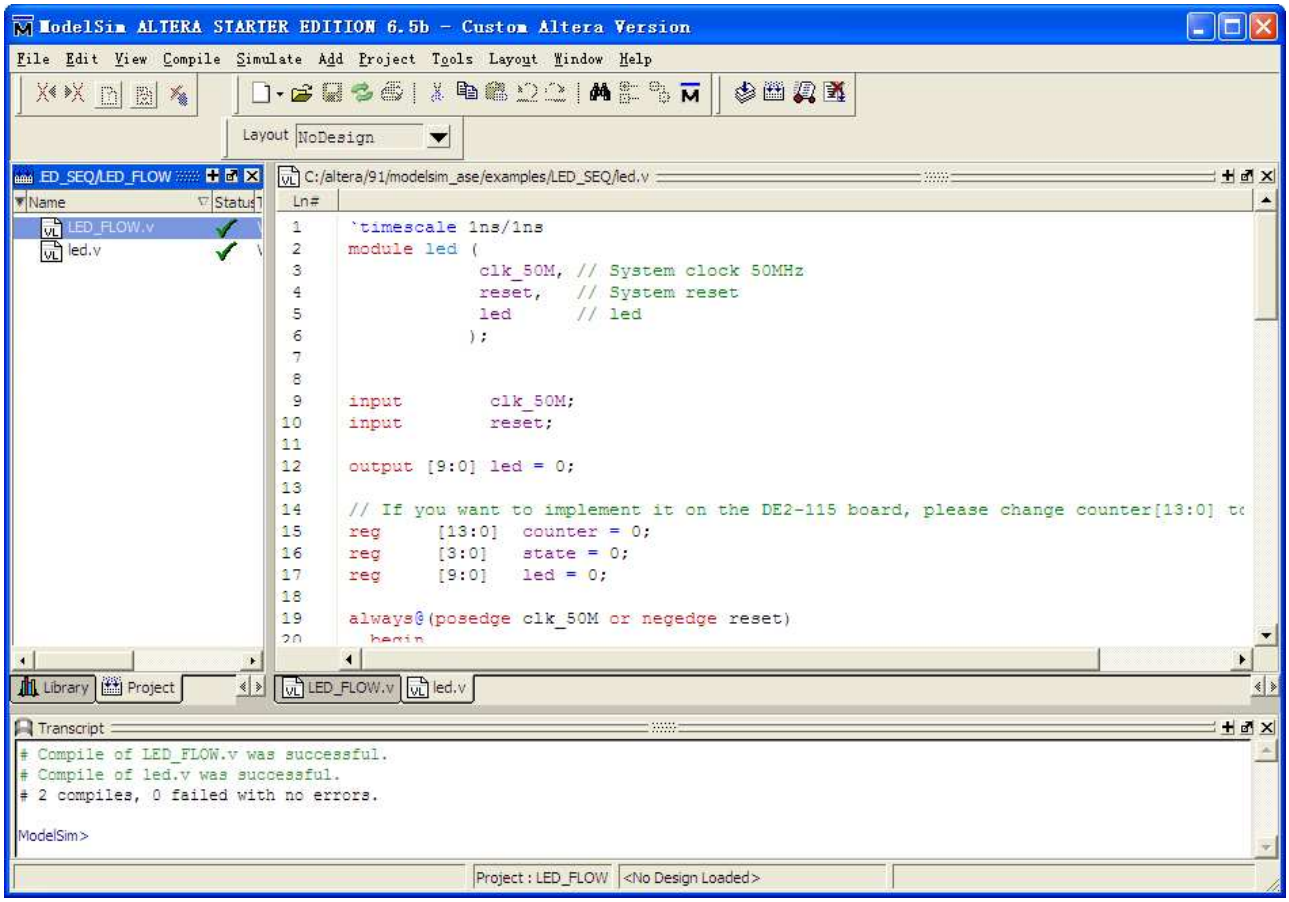

Figure 2.16 Compile successful

After compiling successfully, we begin to check the process of simulation.

3.1 Starting Simulation

1. Click **library** icon, select **work**, and click **+** to expand the options, then select **LED_FLOW**, click the right mouse button, finally select simulation, shown in Figure 3.1.

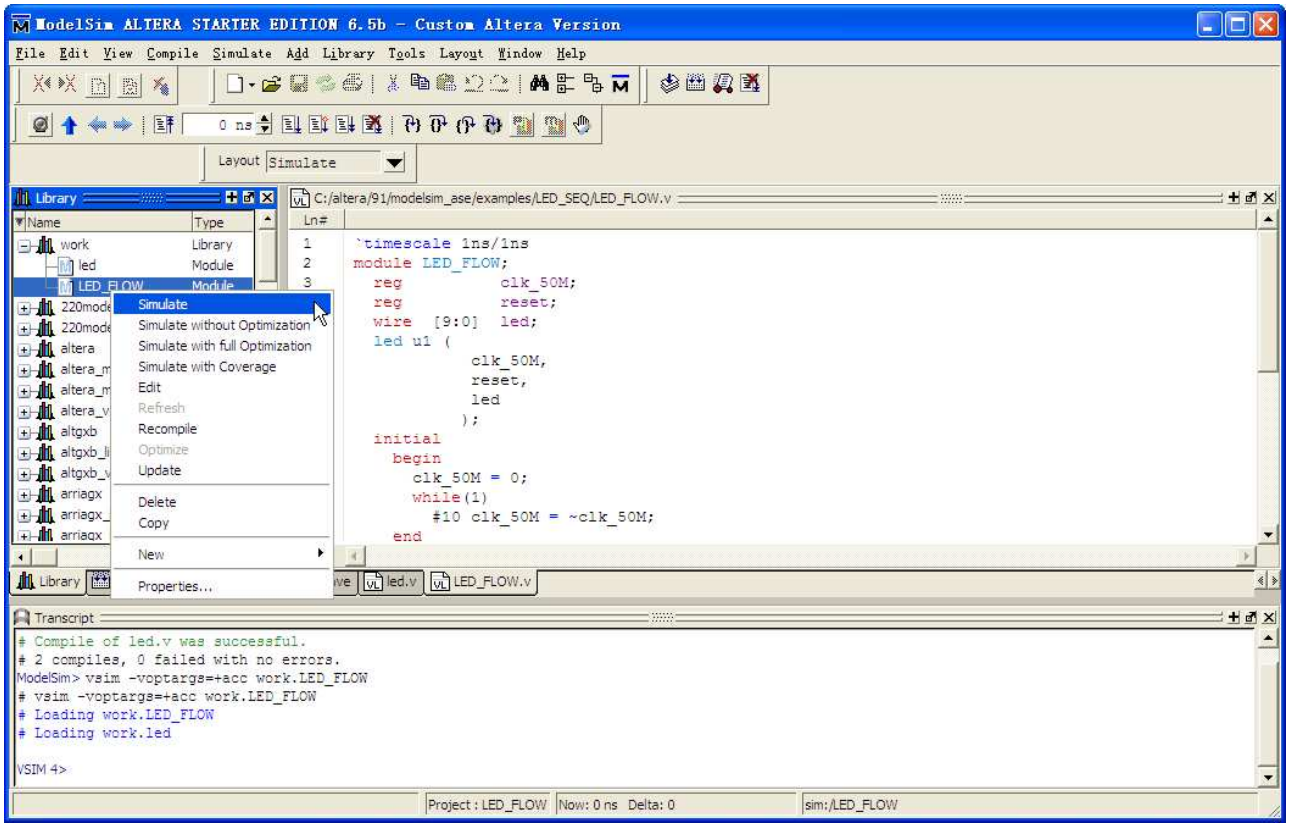

Figure 3.1 Click Simulation

3. Click **Simulation** icon to reach the window shown in Figure 3.2.

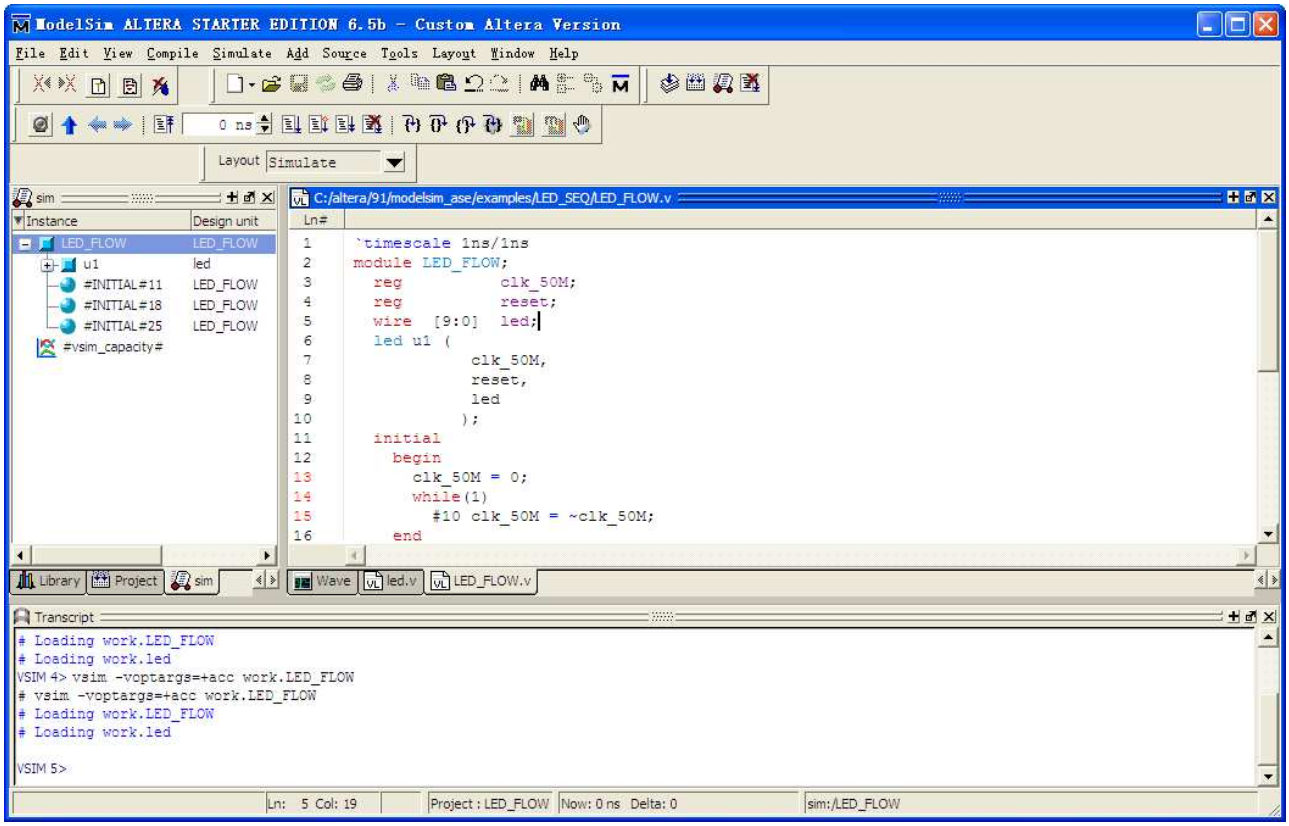

Figure 3.2 Simulation window

4. In Figure 3.3, select **LED_FLOW**, click the right mouse button, then select **Add > To Wave > All items in region**, then click the left mouse button. Figure 3.4 will appear.

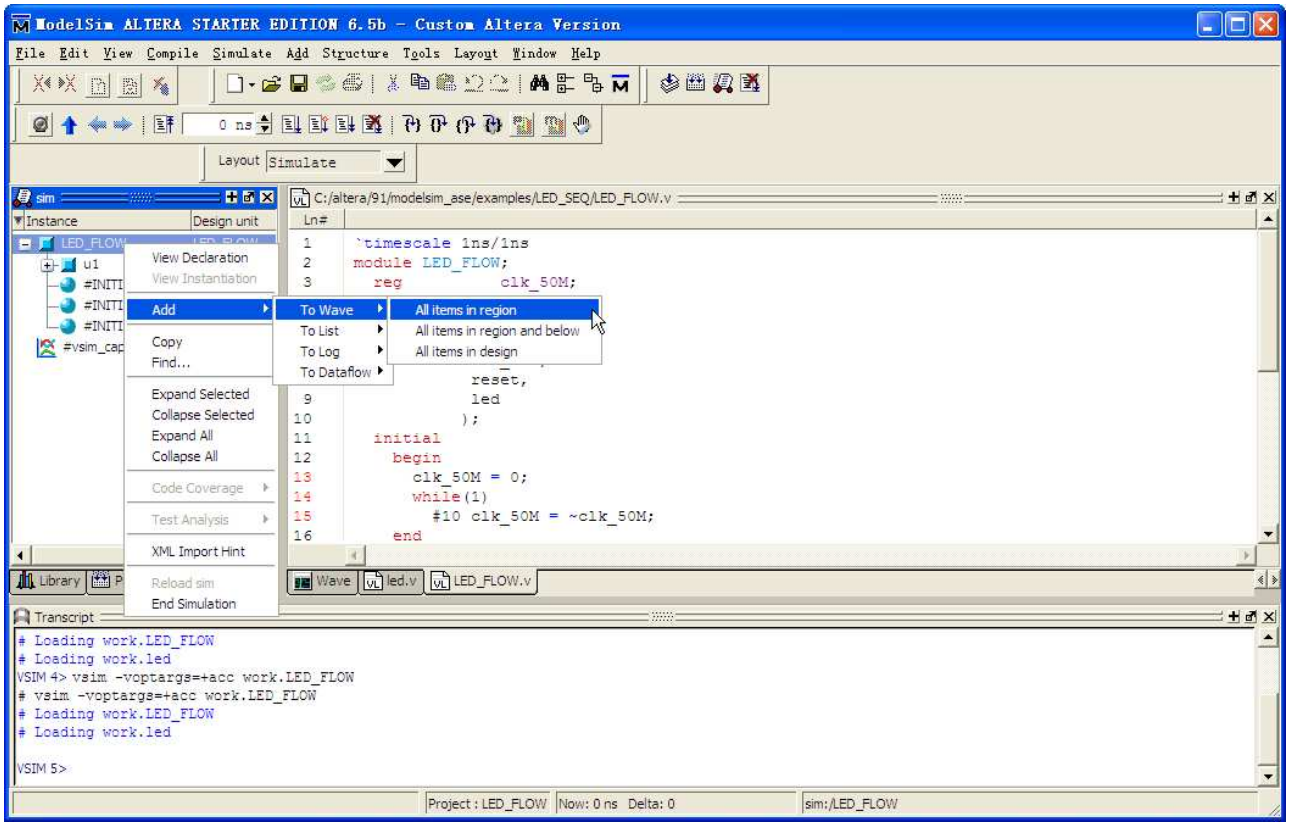

Figure 3.3 Add To Wave

5. After finishing the last step above, the wave window will appear, shown in Figure 3.4.

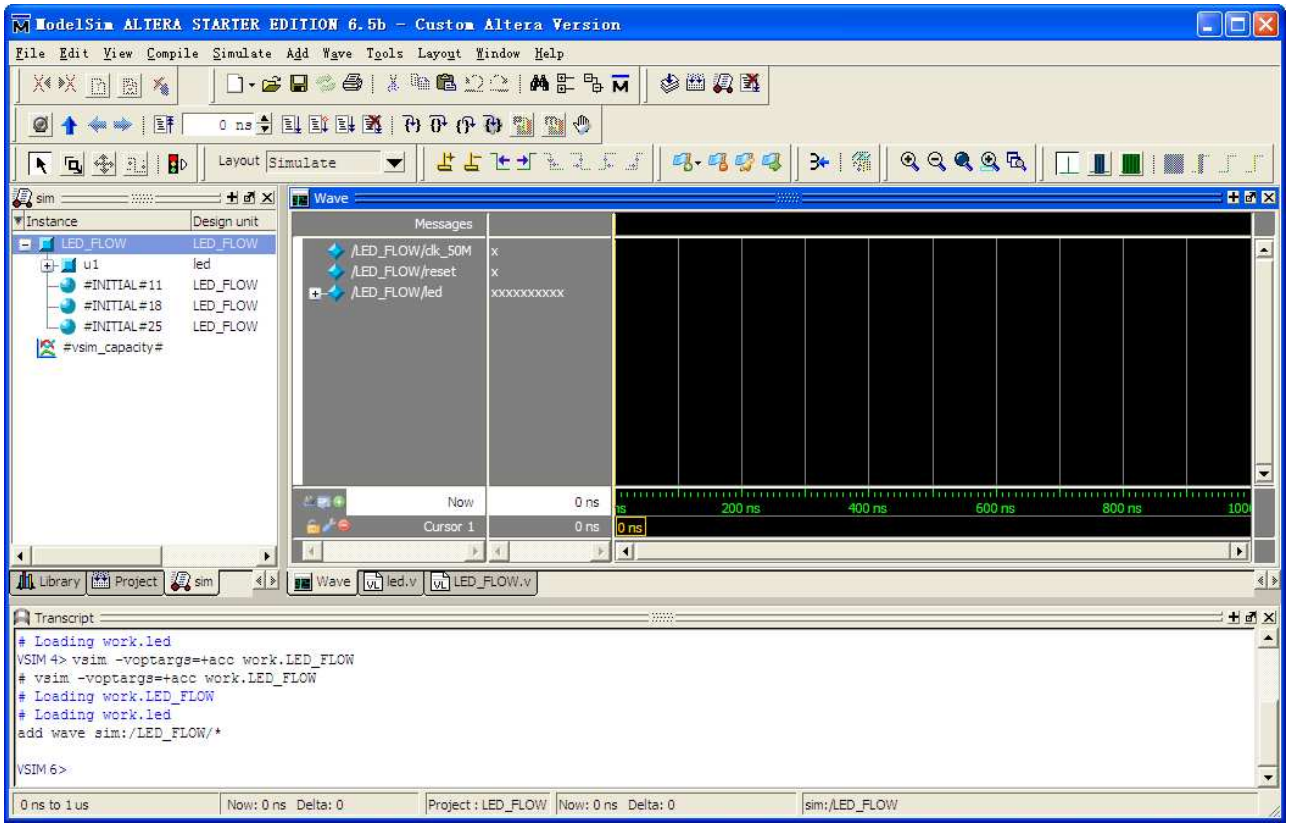

Figure 3.4 Wave window

6. Input simulation time length “10ms” in the Run Length column，shown in Figure 3.5.

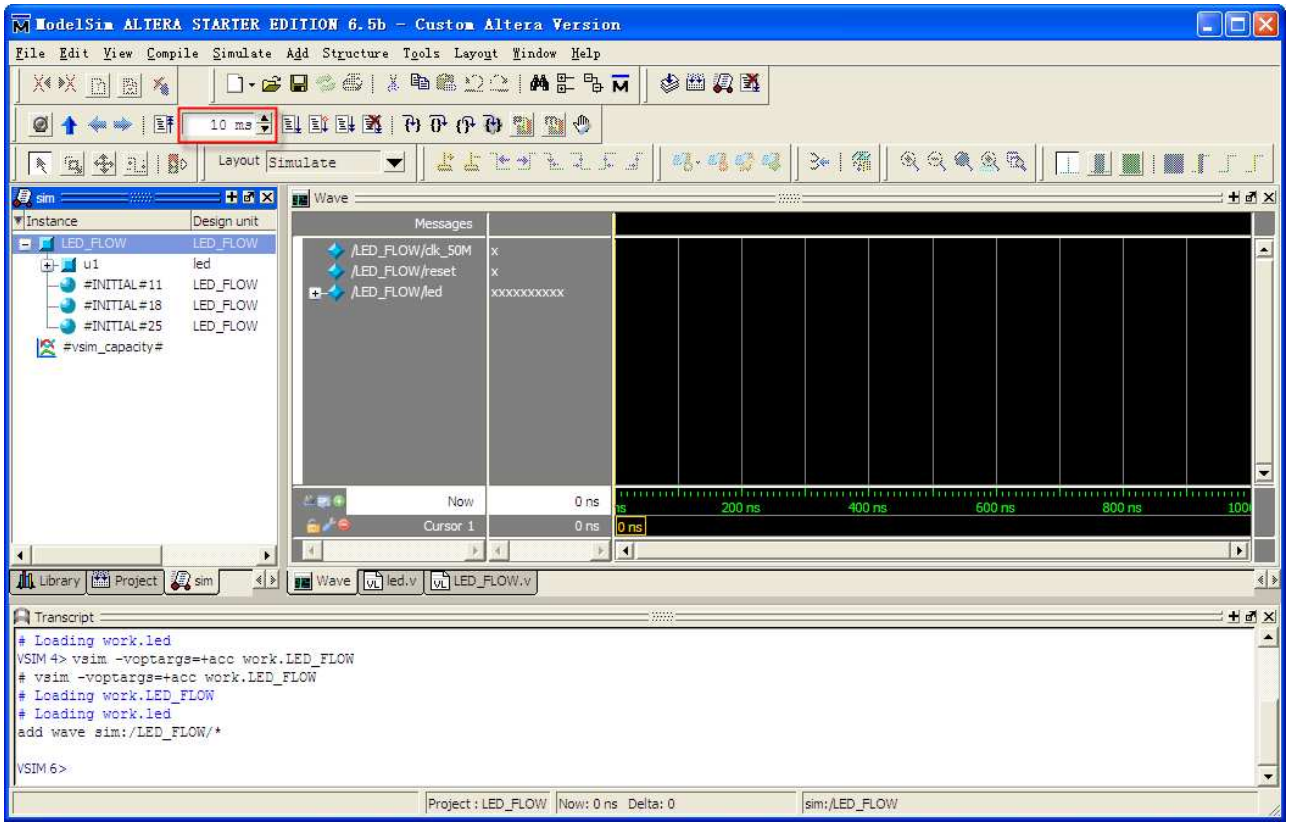

Figure 3.5 Setting Run Length

7. Click on the **Run** button, which is pointed by a red arrow, shown in Figure 3.6.

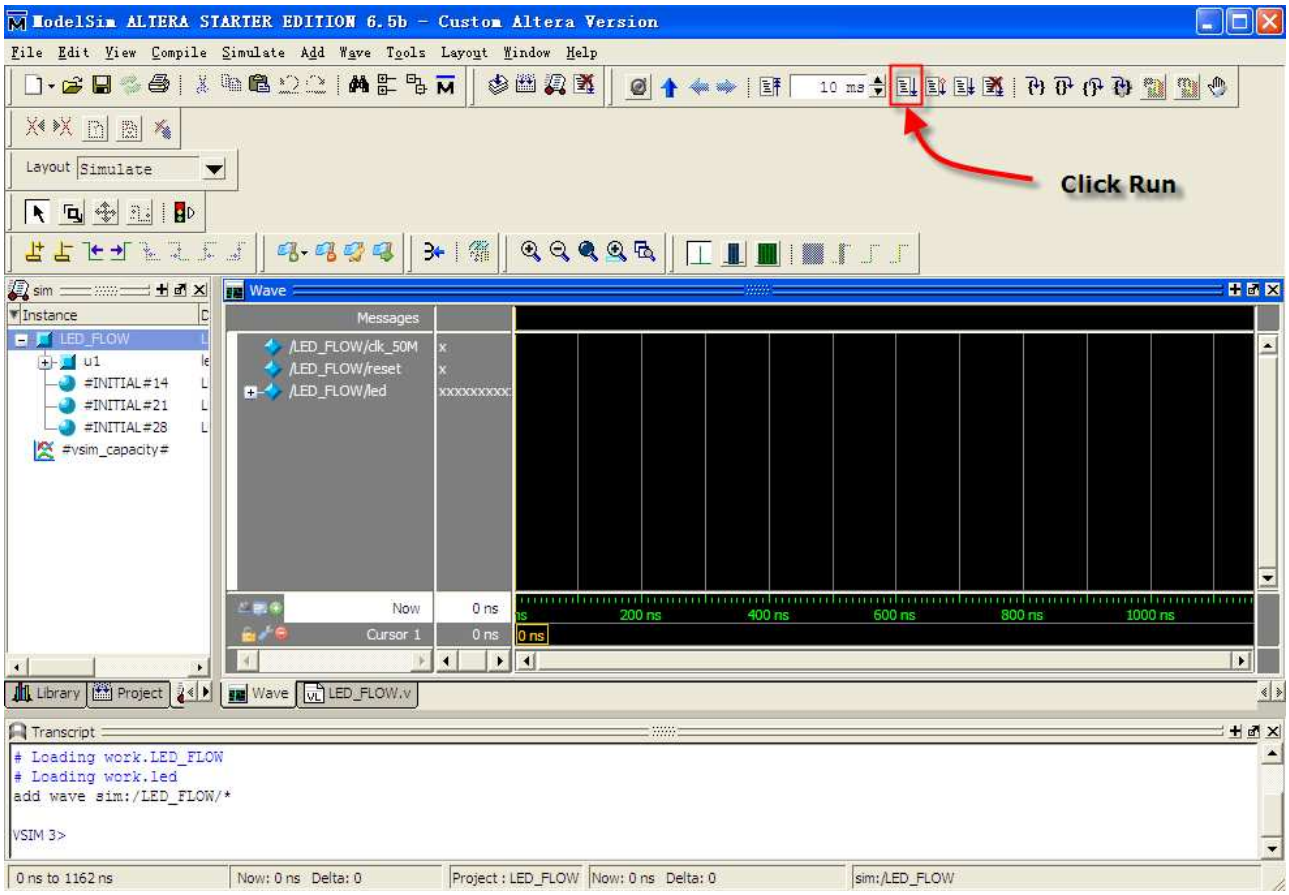

Figure 3.6 Run the Simulation

8. Running for about 3 seconds，the Simulation Result shown in Figure 3.7 will appear.

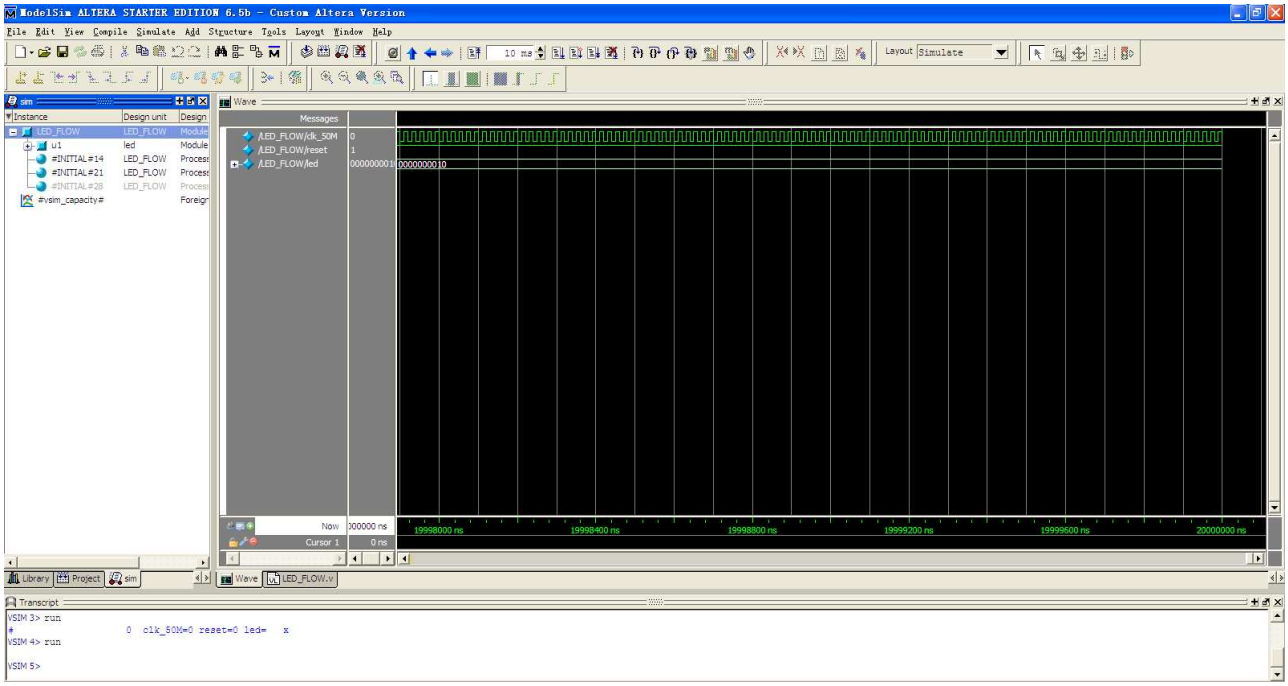

Figure 3.7 Display the Simulation Result

9. Click **Zoom Out** icon, you can observe the whole simulation wave, shown in Figure 3.8.

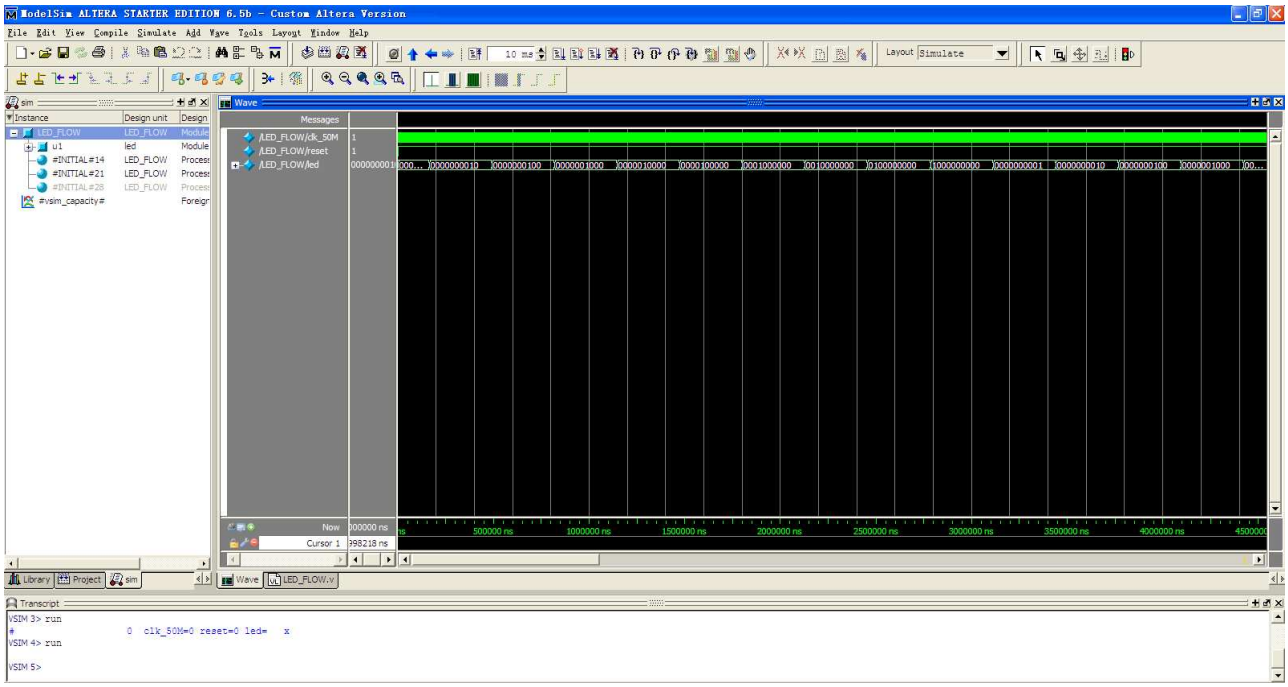

Figure 3.8 Wave window

10. Click on the **+** button pointed by a red arrow to expand the **LED_FLOW** wave, shown in Figure 3.9.

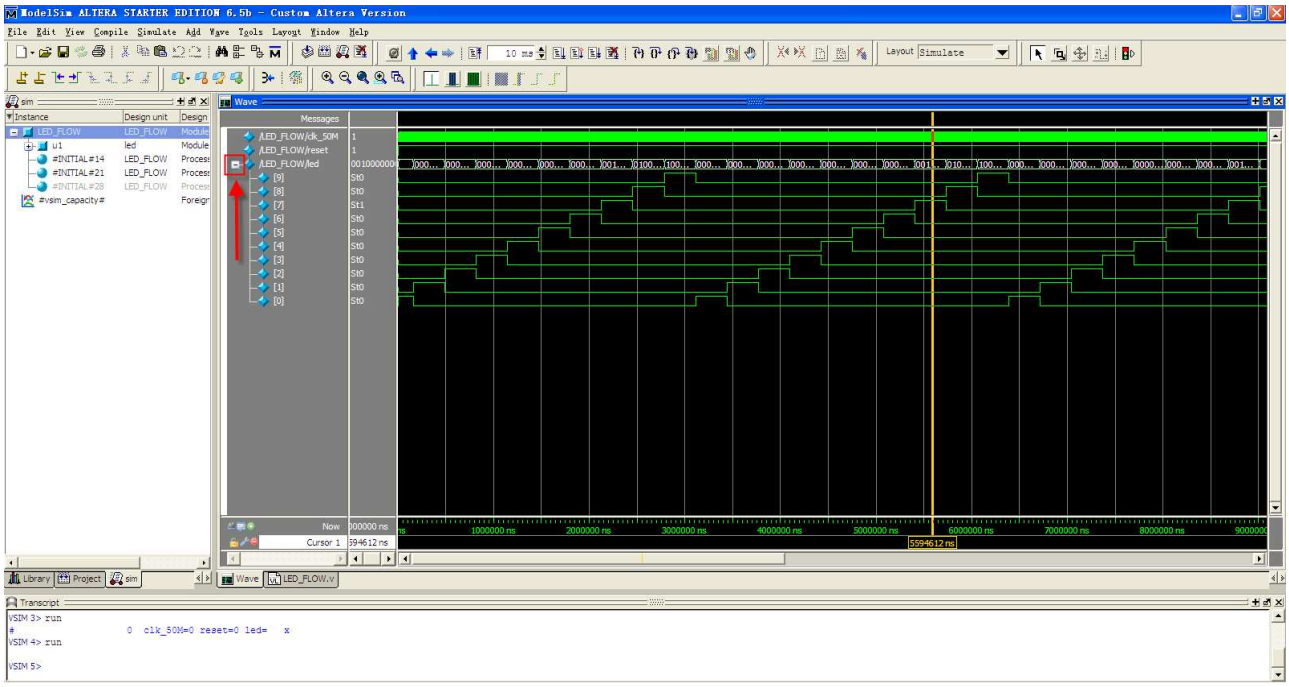

Figure 3.9 Expand the wave

By zooming in /out the wave, we can observe that the values of LEDs keep changing, which means we finally achieve the effects of LEDs sequencer. When transplanting it to Quartus II, you need to change the bit width of counter to 23 bit （download to DE2-115 Board）, here in order to simulate more conveniently, we make it smaller. At this point, the overall process of simulation is finished.