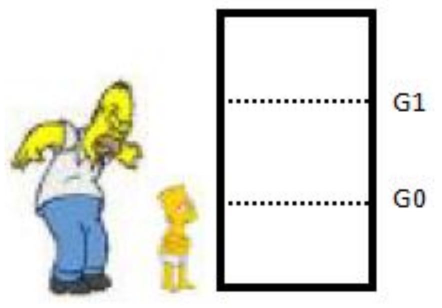
SIMPSONS SENSOR

Case Statement

BY: BANGON KALI (NOTE: SAYUP CGURO KAAU NI!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ^_^)

Marge wants to install an alarm that triggers as soon as somebody enters the kitchen. The alarm should have several alert levels.

LEVEL 0: neither Homer nor Bart is in the Kitchen
LEVEL 1: Bart but not Homer is in the Kitchen
LEVEL 2: Homer but not Bart is in the Kitchen
LEVEL 3: Homer and Bart are in the Kitchen



To detect who enters or leaves the Kitchen 2 sensors G1 and G0 are installed in the door frame as depicted. The sensors emit a '1' as soon as their reflection is interrupted. If Bart enters the Kitchen only G0 will emit a '1'. Homer is always leaning forward when he is entering the kitchen, and, thus, G1 will always be interrupted before G0. Once they have decided to go into the Kitchen they'll go through the door. However, if they are in the Kitchen they always can leave, e.g., LEVEL 3 changes to LEVEL 2. The size of homers hips and belly prevent them from entering the kitchen simultaneously. The clock frequency is 1 MHz. (More info here)

Expected Behaviour

The following are the expected simulation results. These are the same outputs produced by a simulation of the Verilog code that follows.

```
Bart Enters: LEVEL = 1
Bart Leaves: LEVEL = 0
Homer Enters: LEVEL = 2
Homer Leaves: LEVEL = 0
Bart Enters: LEVEL = 1
Homer Enters: LEVEL = 3
Homer Leaves: LEVEL = 1
Homer Enters: LEVEL = 3
Bart Leaves: LEVEL = 2
Bart Enters: LEVEL = 3
Homer Leaves: LEVEL = 1
Bart Leaves: LEVEL = 0
```

Solution

The following conventions are assumed and a state diagram is developed based on them.

```
FORMAT
[INPUT]/[OUTPUT] // The output is in Binary, it's decimal form would be the LEVEL

INPUT DEFINITION
[INPUT] = [G1][G0]
        BART ENTERS => [G0] = 1
        HOMER ENTERS => [G1] = 1 THEN G0 = 1

OUTPUT DEFINITION
[OUTPUT] =>
        00 = LEVEL 0 => NONE  => // neither Homer nor Bart is in the Kitchen
        01 = LEVEL 1 => BART  => // Bart but not Homer is in the Kitchen
        10 = LEVEL 2 => HOMER => // Homer but not Bart is in the Kitchen
        11 = LEVEL 3 => BOTH  => // Homer and Bart are in the Kitchen
```

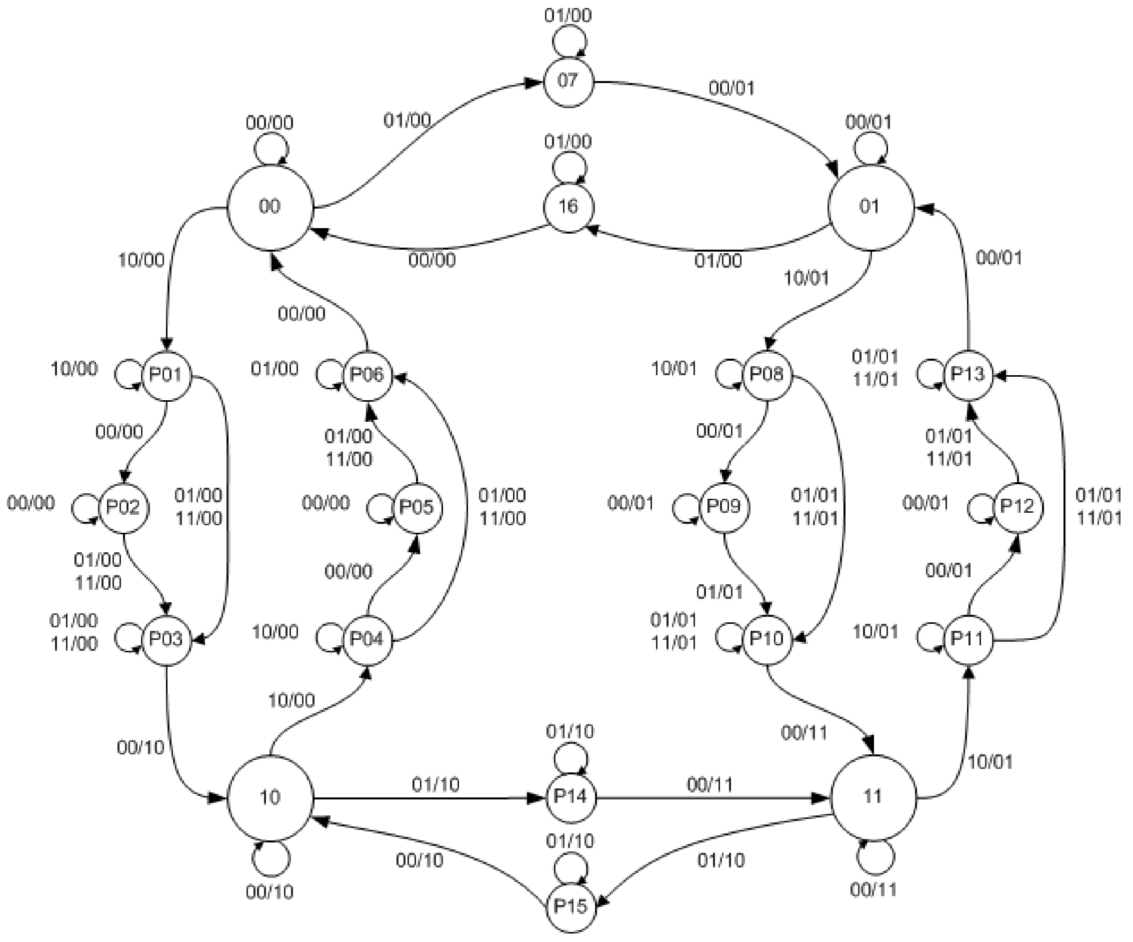


Figure 1 The state diagram is based on the conventions stated above it. The Microsoft Visio file is available <u>here</u>. The small circles are mini states used as transition states among the four big circles which are the main states. Note this is a Mealy Machine. ☺

Discussion

A simple trace of the state diagram can go as follows.

1.) Initially starting at state 00 (the big circle with two zeroes) if the sensor inputs are 01 or 0 from G1 and 1 from G0, then it means that Bart at the door about to enter the room. The next state is P07. He has not entered yet. The moment where he is still entering is taken care by the loop at P07.

2.) If the sensor indicates a signal of 00 or 0 from G1 and G0 is 0, then it means he has entered already. Then it will proceed to the main state 01, while outputting 01 which indicates a LEVEL 1 in binary.

3.) Inside the room, if for example nobody’s at the door, then the loop at Main State 01 will make sure the output remains at 01 even though the inputs by the sensor are 00, because at the moment Bart entered the room, the level should be 01 or 1.

4.) If for example Homer enters the room then the sensor would input a reading of 10 initially then 01 afterwards. You can follow line from Main State 01 to P08. The loop at P08 makes sure Homer’s head can still pass the door. After his head, either the sensors will output 00 for a moment or 10 immediately, it will proceed to P10. At P10, if the sensor inputs are 00, then this means no one’s at the door and Homer has finally entered the room. The state will then be 11 indicating an output Level of 3.

5.) If Bart leaves at Main State 11, then the inputs from the sensors would be 01 indicating 0 for G1, and 1 for G0. The new state would be at P15, after finishing receiving inputs of 01, 00 will follow indicating no one is at the door, which means Bart is already out. The new state would be 10 or an output Level of 2.

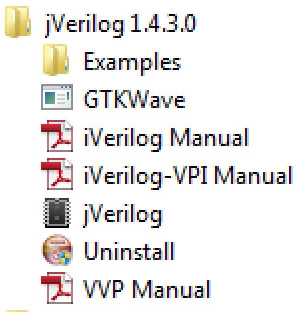
6.) You should be able to follow the rest of the situation by yourself now. =P

jVerilog Simulation

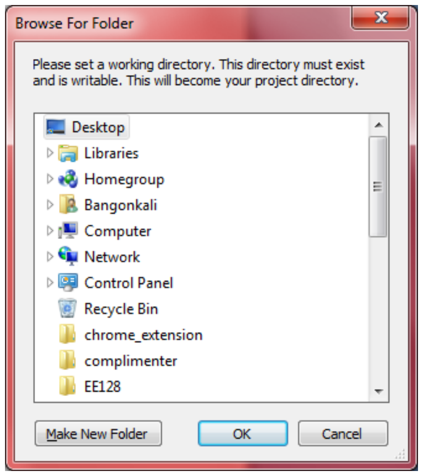
A Verilog code is written depicting the behaviour described by the state machine above. The code is very long; therefore it would be hopeless posting it here with formatting. The Xilinx Compatible code is available along with its test bench from here.

The code was written with the Xilinx ISE Project Navigator 13.2. Compilation and synthesis would probably be very easy if one has experience on the use of Xilinx. However, without Xilinx, one can still observe the codes behaviour by using other tools. One such is jVerilog which you can download from here. The following discussions will be on how to view the waveforms produced by the code using jVerilog. (**NOTE: If you have iVerilog installed in your PC, uninstall it first before installing jVerilog!**)

1.) Download the jVerilog compatible code from here. Extract it to your desired folder.
2.) Download and install jVerilog from here.
3.) After installing jVerilog, you can find it from the start menu.

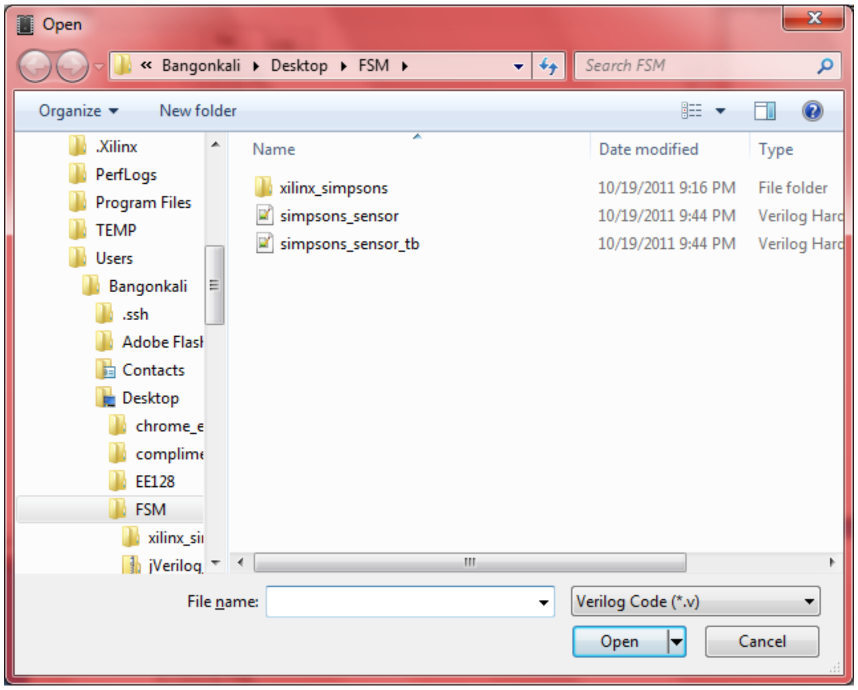


4.) You'll find this Window after running jVerilog. It will ask you to specify a working directory or a project folder. Find the jVerilog folder where you extracted the jVerilog compatible code.


Browse For Folder

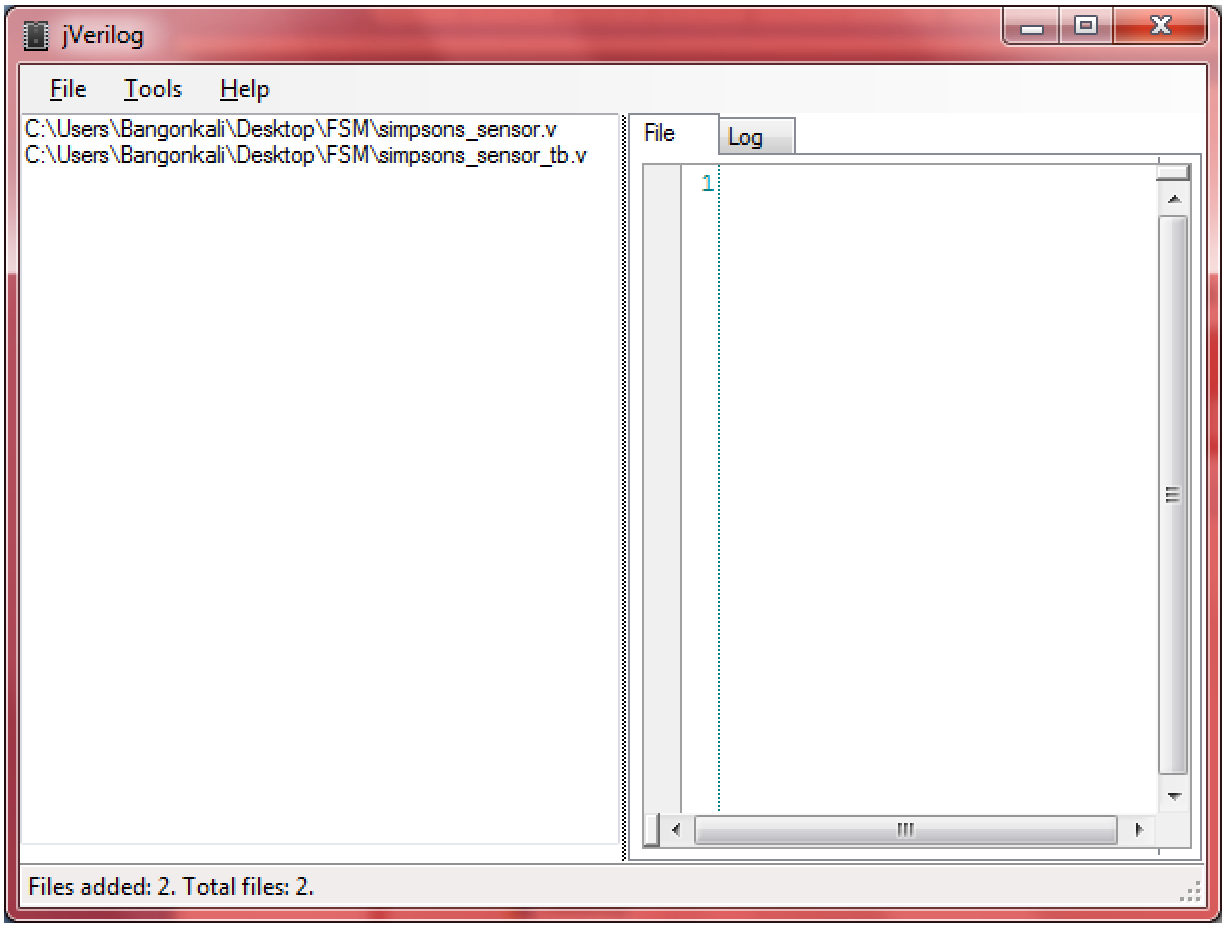
Please set a working directory. This directory must exist and is writable. This will become your project directory.

5.) Go to the File Menu, and then add the two files inside the folder you just extracted.



Name	Date modified	Type
xilinx_simpsons	10/19/2011 9:16 PM	File folder
simpsons_sensor	10/19/2011 9:44 PM	Verilog Hard
simpsons_sensor_tb	10/19/2011 9:44 PM	Verilog Hard

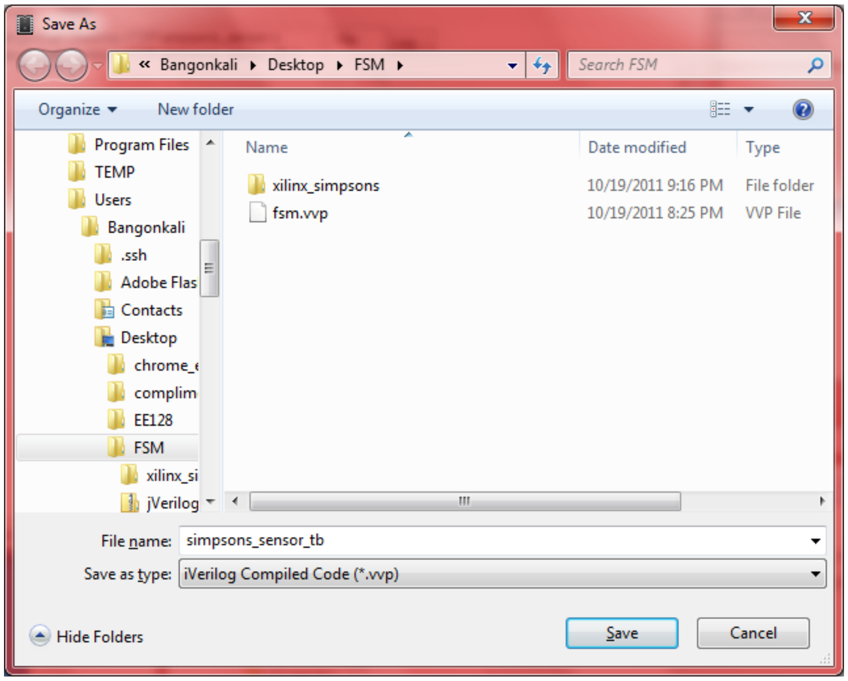
6.) You should now have the two files in your jVerilog window.



```
C:\Users\Bangonkali\Desktop\FSM\simpsons_sensor.v
C:\Users\Bangonkali\Desktop\FSM\simpsons_sensor_tb.v
```

Files added: 2. Total files: 2.

7.) Now, go to Tools Menu, then click compile. In the filename, put simpsons_sensor_tb.

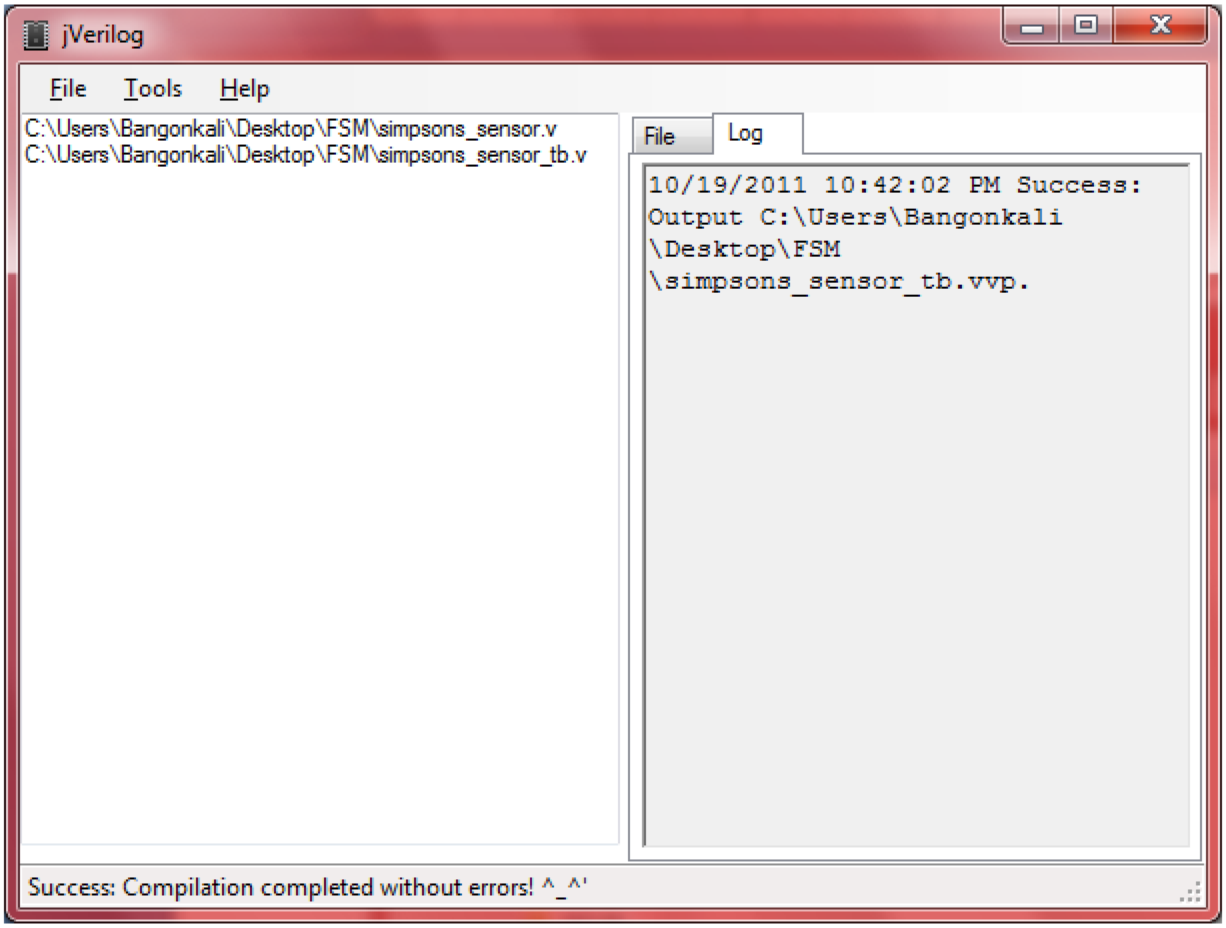


Name	Date modified	Type
xilinx_simpsons	10/19/2011 9:16 PM	File folder
fsm.vvp	10/19/2011 8:25 PM	VVP File

File name: simpsons_sensor_tb

Save as type: iVerilog Compiled Code (*.vvp)

8. jVerilog should respond with the following log. This indicates a successful compile.

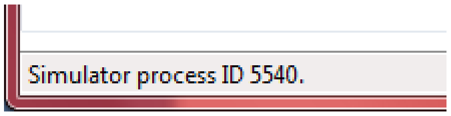


C:\Users\Bangonkali\Desktop\FSM\simpsons_sensor.v
C:\Users\Bangonkali\Desktop\FSM\simpsons_sensor_tb.v

```
10/19/2011 10:42:02 PM Success:
Output C:\Users\Bangonkali
\Desktop\FSM
\simpsons_sensor_tb.vvp.
```

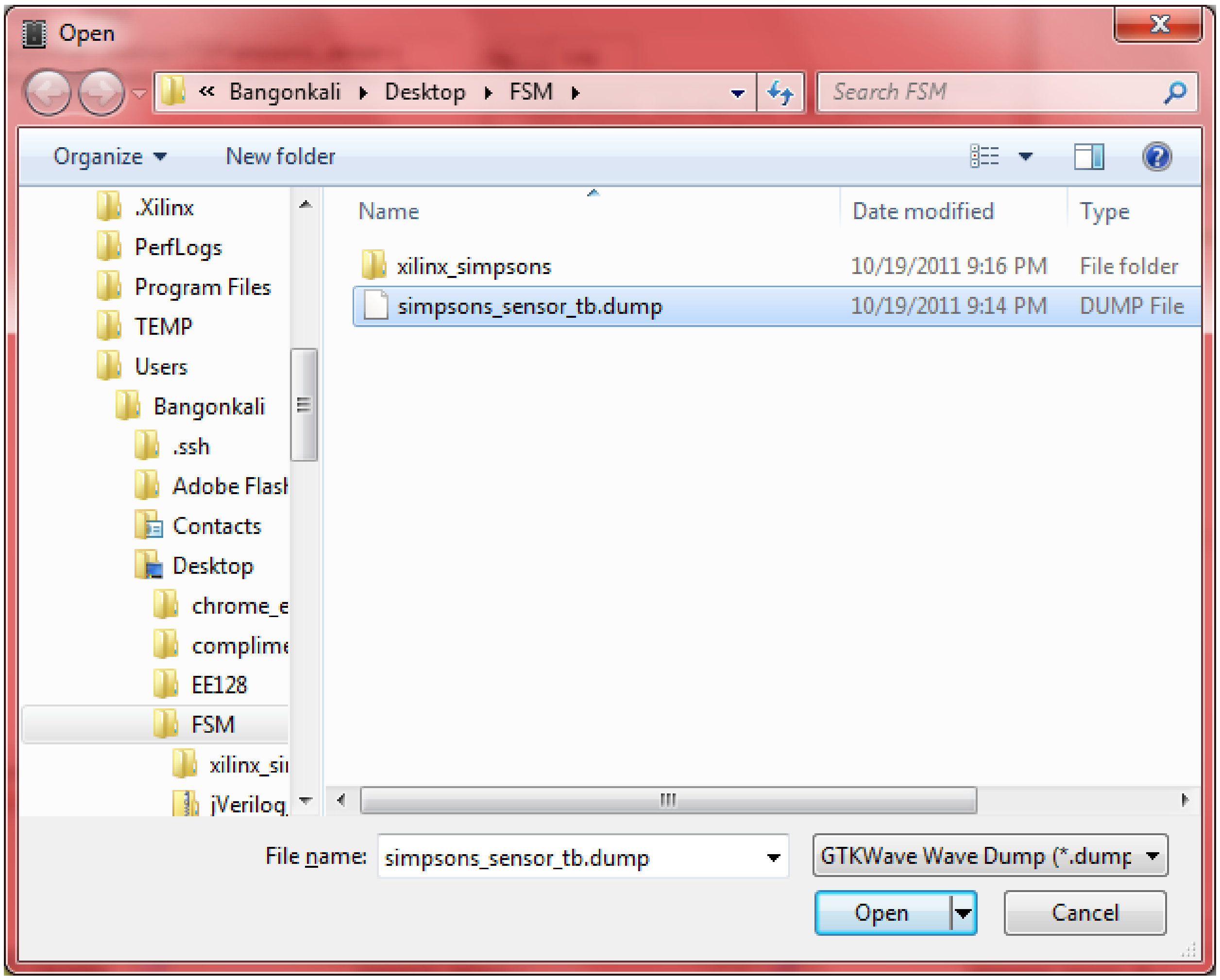
Success: Compilation completed without errors! ^_^'

9.) Go back to the Tools menu, and then click Simulate. The VVP simulator should simulate for a while, and then automatically close. You should see something like this in the status bar after a successful simulation.



Simulator process ID 5540.

10.) After simulation, go back to the Tools Menu to show the wave form produced by the code. Find the file named simpsons_sensor_tb.dump (which was automatically produced after simulation as indicated in the source code of simpsons_sensor_tb.v).

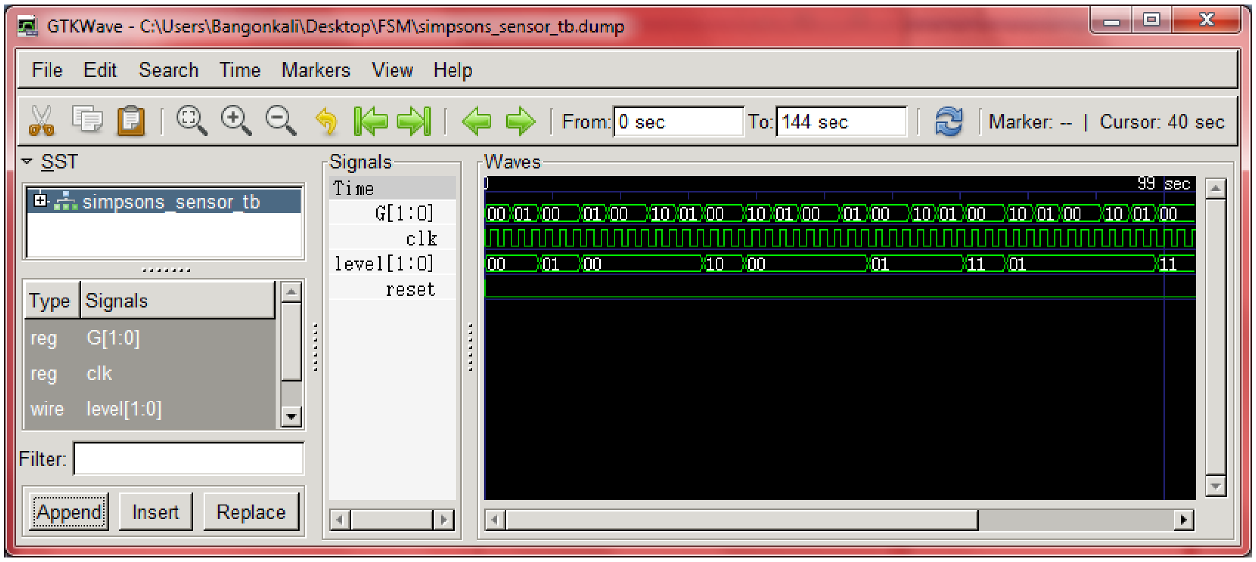


Name	Date modified	Type
xilinx_simpsons	10/19/2011 9:16 PM	File folder
simpsons_sensor_tb.dump	10/19/2011 9:14 PM	DUMP File

File name: simpsons_sensor_tb.dump

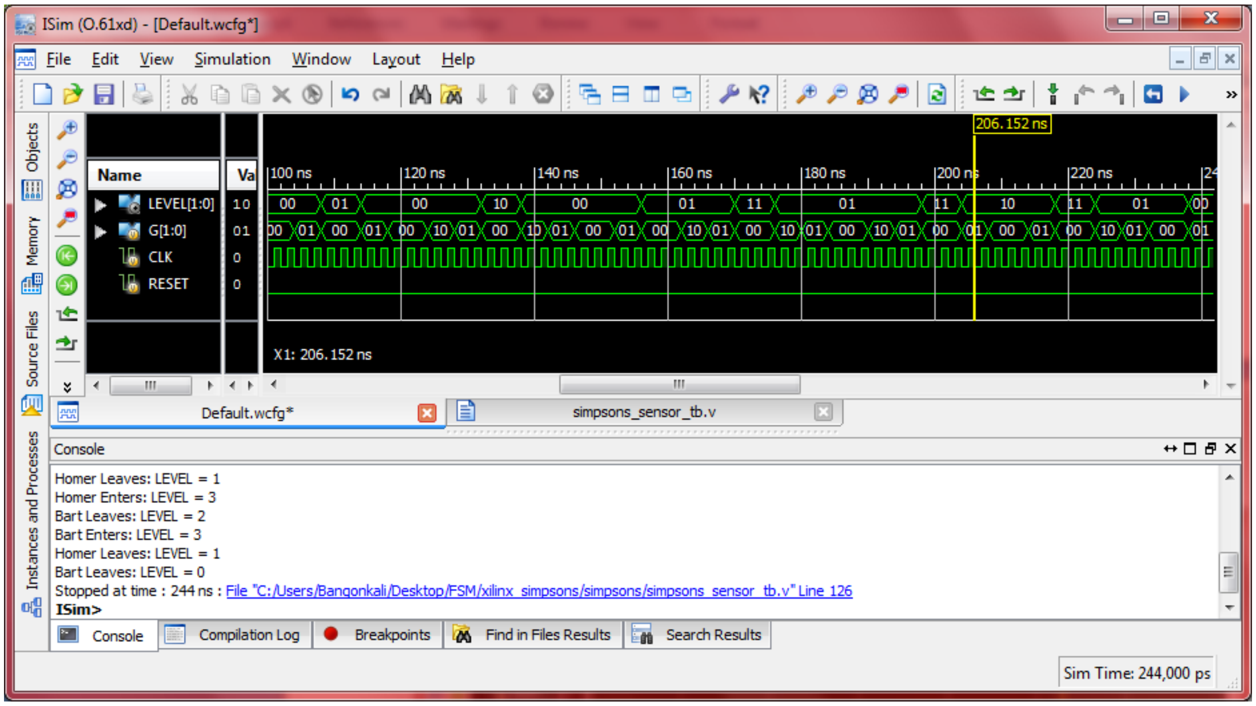
GTKWave Wave Dump (*.dump

11.) After clicking that file, GTKWave will open and you can then use it to inspect the wave form generated by the source code.


GTKWave - C:\Users\Bangonkali\Desktop\FSM\simpsons_sensor_tb.dump
File Edit Search Time Markers View Help
From: 0 sec To: 144 sec Marker: -- | Cursor: 40 sec
SST
simpsons_sensor_tb
Type Signals
reg G[1:0]
reg clk
wire level[1:0]
Filter:
Append Insert Replace
Signals
Time
G[1:0]
clk
level[1:0]
reset
Waves
99 sec

XILINX

This is how the wave forms would look like using the Xilinx ISim Simulator.



```
Homer Leaves: LEVEL = 1
Homer Enters: LEVEL = 3
Bart Leaves: LEVEL = 2
Bart Enters: LEVEL = 3
Homer Leaves: LEVEL = 1
Bart Leaves: LEVEL = 0
Stopped at time : 244 ns : File "C:/Users/Bangonkali/Desktop/FSM/xilinx_simpsons/simpsons/simpsons_sensor_tb.v" Line 126
ISim>
```

License