Remote Triggered FPGA based Automation System

Table of Contents:

Chapter One:

Introduction

Need & Inspiration for building such a system

Simulation does not give the feel of running the designs on an actual FPGA system. So if a system is built such that the user can run his design and check his output for various inputs on the actual system without actually having the system with him i.e. he accesses the FPGA based system remotely, then most of the users' work will be simplified.

But the design cannot be implemented until the designer compiles, synthesizes then assigns pins to the board for which he needs to know the board details. All this becomes tedious. This can be simplified if the user is just asked to provide with his design file and the test vectors and rest of the part is automated.

Thus bringing us to the Remote Triggered FPGA based Automation system. Such kind of system is developed under WEL Virtual labs with the help of Altera DE2-70 board.

Block diagram:

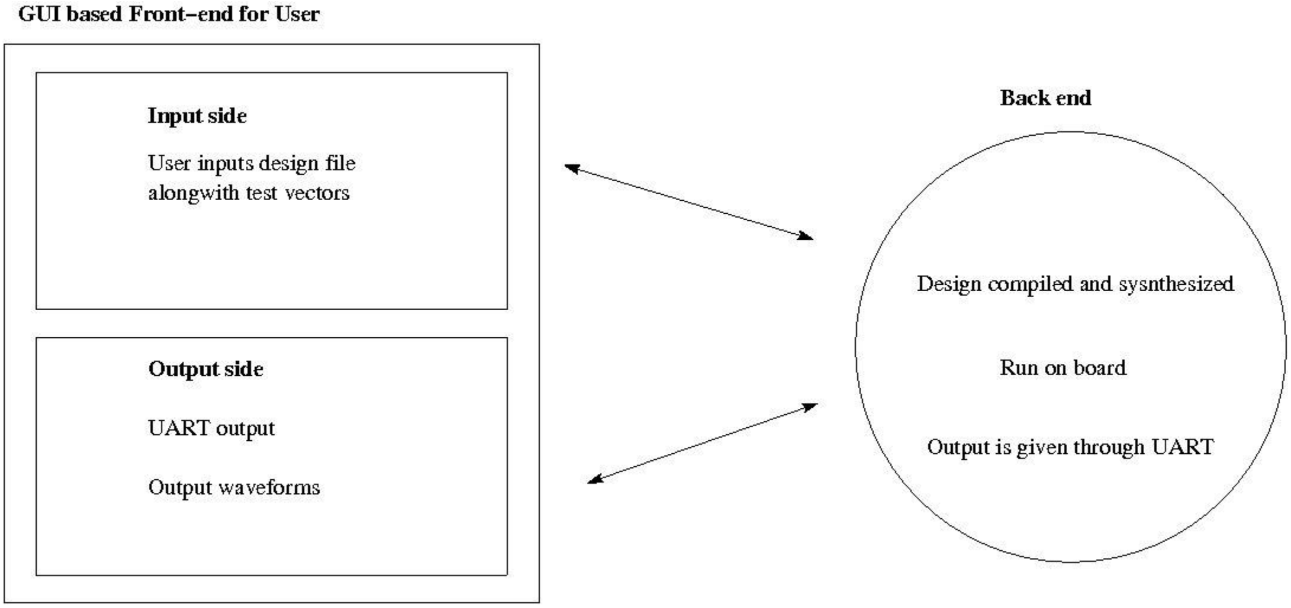


Fig. 1.1 – Basic block diagram of Remote triggered FPGA based automation system

Basic working of this system

- Remote user uploads the design file (eg.: user_design.v) in specified format along with test vectors.
- An FSM system is invoked which will automate the process.
- This FSM will take the user design along with the test vectors and generate the output and send it through UART from which the output waveforms can be generated.
- This FSM will be responsible to produce output corresponding to each test vector input.

The following gives a better idea of this FSM system:

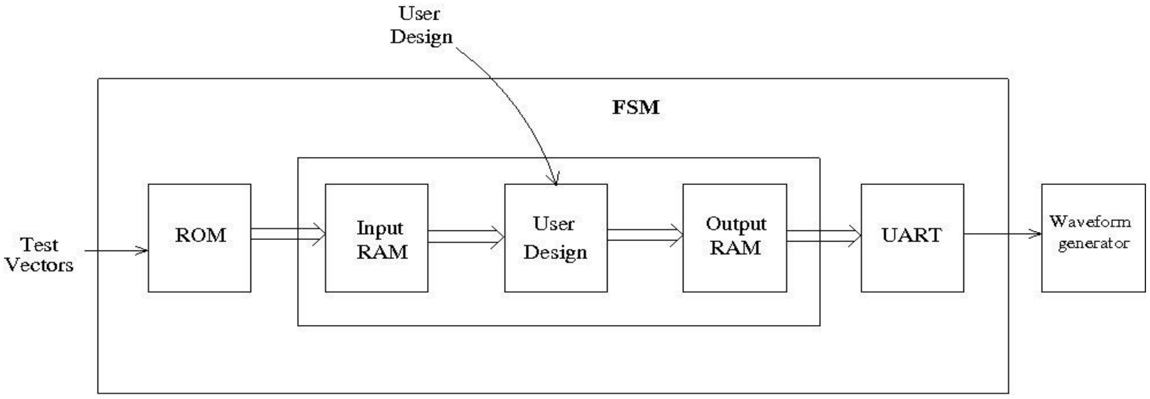


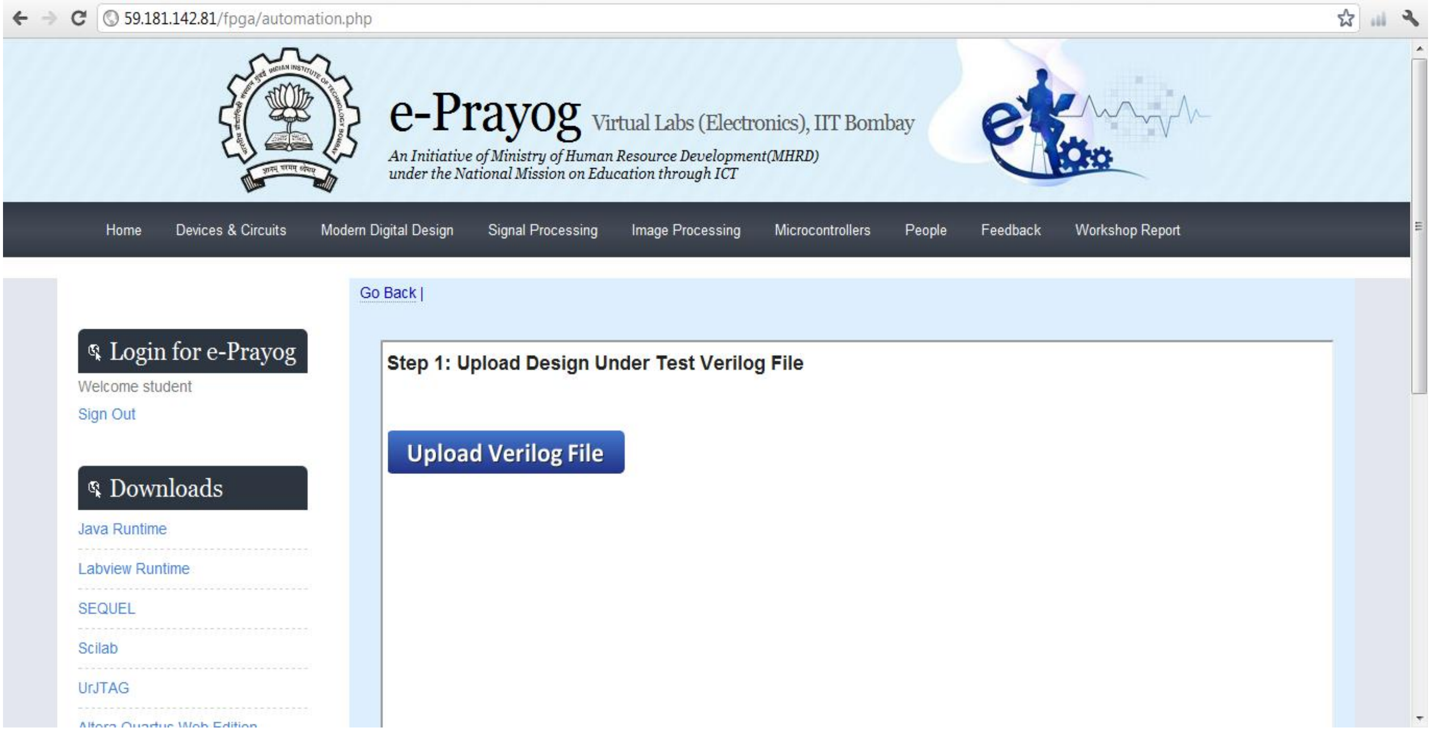
Fig. 1.2 – FSM system

- This Automation FSM system is designed using Verilog HDL and will thus have the Verilog modules ROM.v, INPUT_RAM.v, USER_DESIGN_WRAPPED.v, OUTPUT_RAM.v, UART.v and FSM.v. Named as per the blocks shown in fig. 1.2.
- FSM.v will act as a top-level module which instantiates all other Verilog modules depending on the FSM design.

Chapter Two:

Usage Scenario

- User will access the website of "Remote triggered FPGA based automation system" (http://59.181.142.81/fpga/automation.php)



59.181.142.81/fpga/automation.php

e-Prayog Virtual Labs (Electronics), IIT Bombay

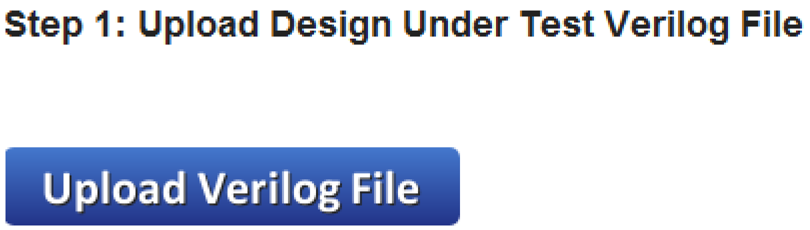
An Initiative of Ministry of Human Resource Development(MHRD) under the National Mission on Education through ICT

Welcome student

Step 1: Upload Design Under Test Verilog File

Fig. 2.1 – Webpage of "Remote triggered FPGA based automation system"

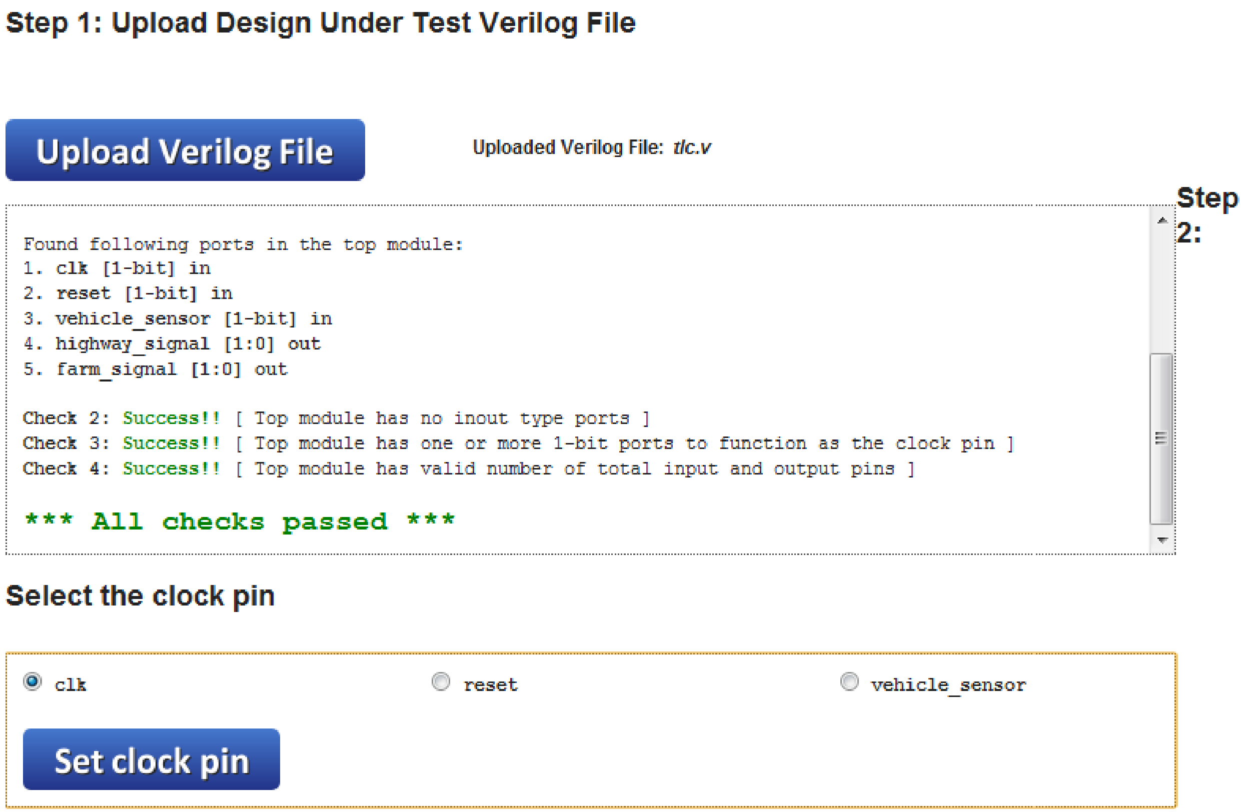
- The user will upload his design by clicking on "Upload Verilog File"



Step 1: Upload Design Under Test Verilog File

Fig. 2.2 – Upload Design file (.v)

- After uploading the design file, the system checks for 1-bit input ports and lists options for the user to select the clock pin.



Step 1: Upload Design Under Test Verilog File

Uploaded Verilog File: *tlc.v*

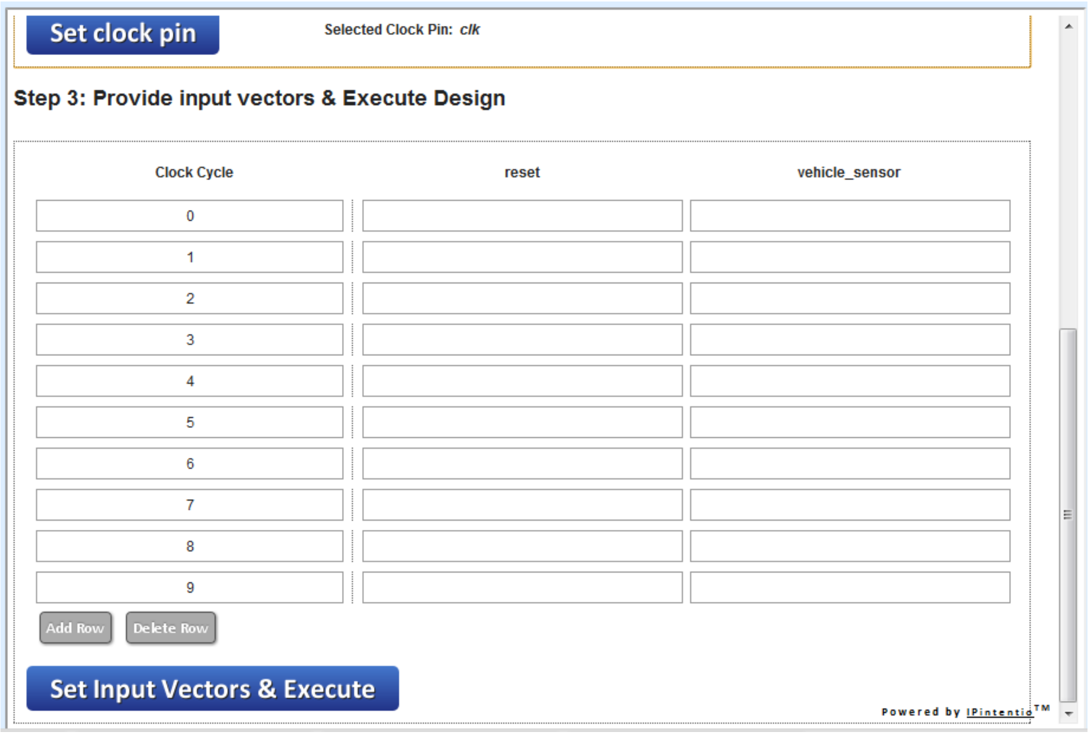
```
Found following ports in the top module:
1. clk [1-bit] in
2. reset [1-bit] in
3. vehicle_sensor [1-bit] in
4. highway_signal [1:0] out
5. farm_signal [1:0] out

Check 2: Success!! [ Top module has no inout type ports ]
Check 3: Success!! [ Top module has one or more 1-bit ports to function as the clock pin ]
Check 4: Success!! [ Top module has valid number of total input and output pins ]

*** All checks passed ***
```

Select the clock pin

● clk ○ reset ○ vehicle_sensor

Fig. 2.3 – Select the appropriate clock pin and click on “Set clock pin” button

- The user has to give appropriate test vector input for the corresponding clock.
- User can provide the input for maximum 256 clock cycles.



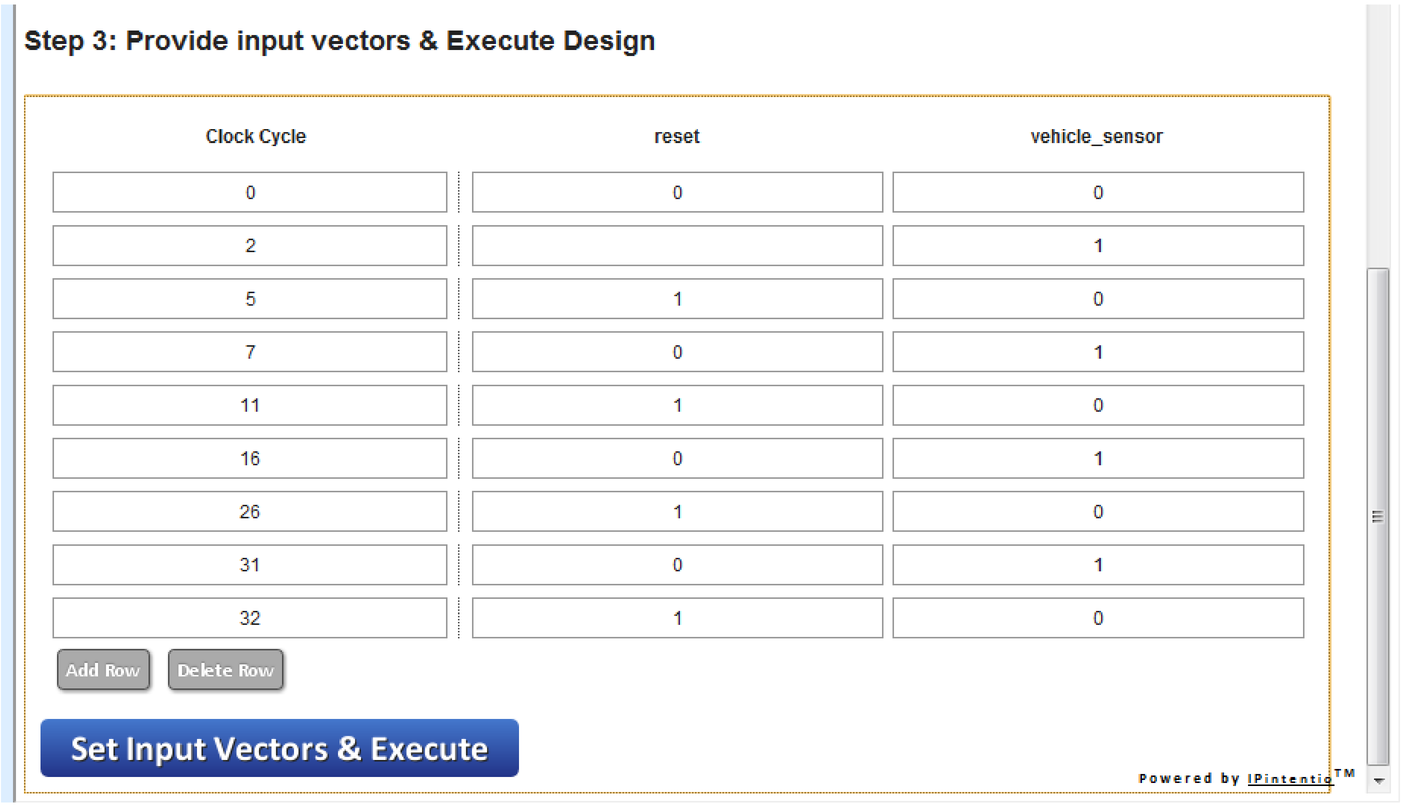
Selected Clock Pin: *clk*

Step 3: Provide input vectors & Execute Design

Clock Cycle	reset	vehicle_sensor
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

Fig. 2.4 – Step appearing after pressing “Set clock pin”

- The user may opt out from giving an input if the input is just repeating for some number of clock cycles where the system will take the value as the value of previous clock cycle but user will have to give the input where the input is changing.
- User can add and delete rows, if required, and can extend upto maximum a total of 256 rows.

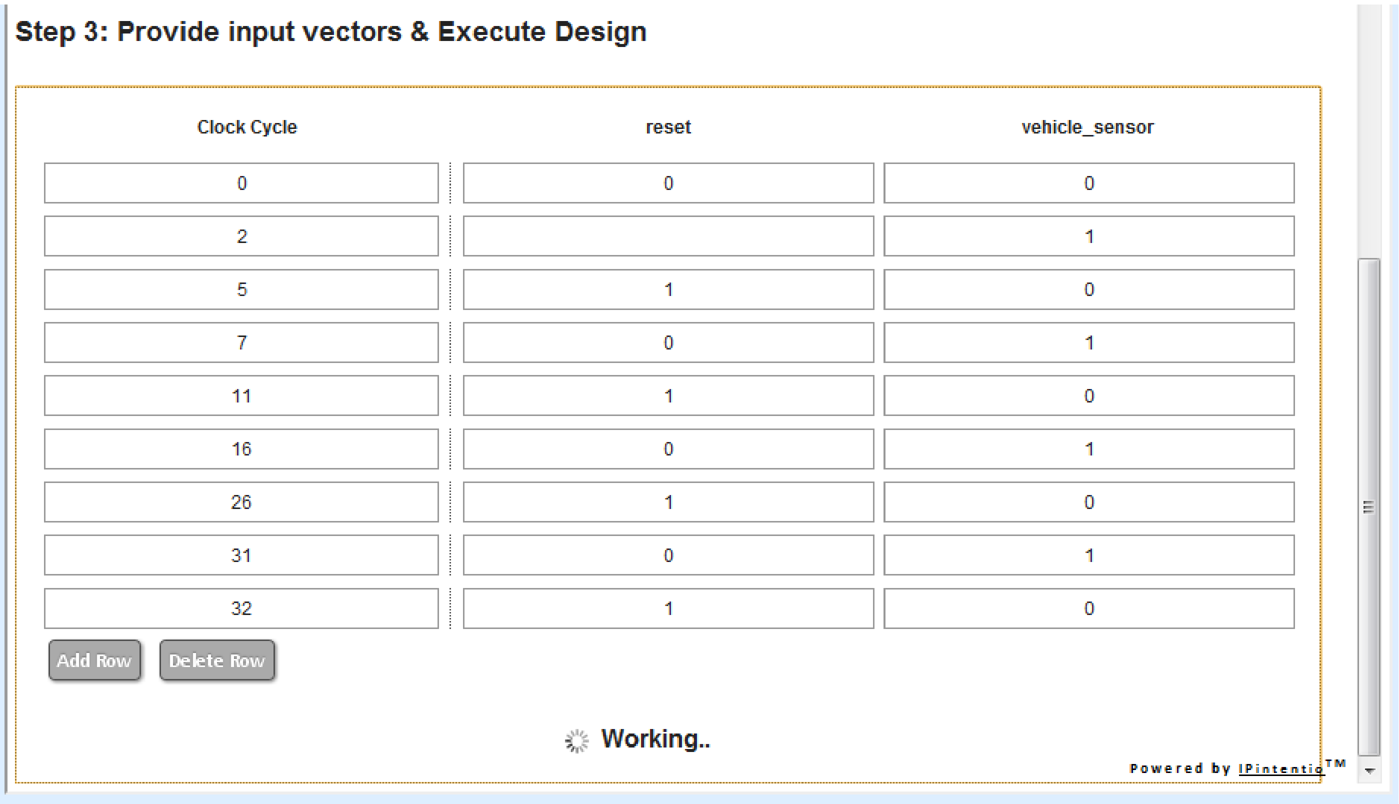


Step 3: Provide input vectors & Execute Design

Clock Cycle	reset	vehicle_sensor
0	0	0
2		1
5	1	0
7	0	1
11	1	0
16	0	1
26	1	0
31	0	1
32	1	0

Fig. 2.5 – Input the appropriate test vectors by giving appropriate value of clock cycle and press “Set Input Vectors & Execute” button

- These inputs along with the clock pin are combined together to form a 16-bit where the rest of the bits are zeros.
- The files INPUT_RAM.c, OUTPUT_RAM.c, ROM.c USER_DESIGN_WRAPPER.c, FSM.c are called the configuration files.
- The input_width and the no. of inputs are supplied to file INPUT_RAM.c by the user to generate INPUT_RAM.v.
- The output_width and no. of outputs (same as no. of inputs) are supplied by the user to file OUTPUT_RAM.c to generate OUTPUT_RAM.v.
- The input_width, output_width and no. of inputs are supplied by the user to file OUTPUT_RAM.c to generate FSM.v
- The test_vectors supplied by the user (as described in fig. 2.5) are read by the ROM.c along with the input_width and no. of inputs to generate ROM.v file .
- The user_design.v file is read by the USER_DESIGN_WRAPPER.c file along with input_width and output_width to generate USER_DESIGN_WRAPPED.v file. This file instantiates module of user_design.v.



Step 3: Provide input vectors & Execute Design

Clock Cycle	reset	vehicle_sensor
0	0	0
2		1
5	1	0
7	0	1
11	1	0
16	0	1
26	1	0
31	0	1
32	1	0

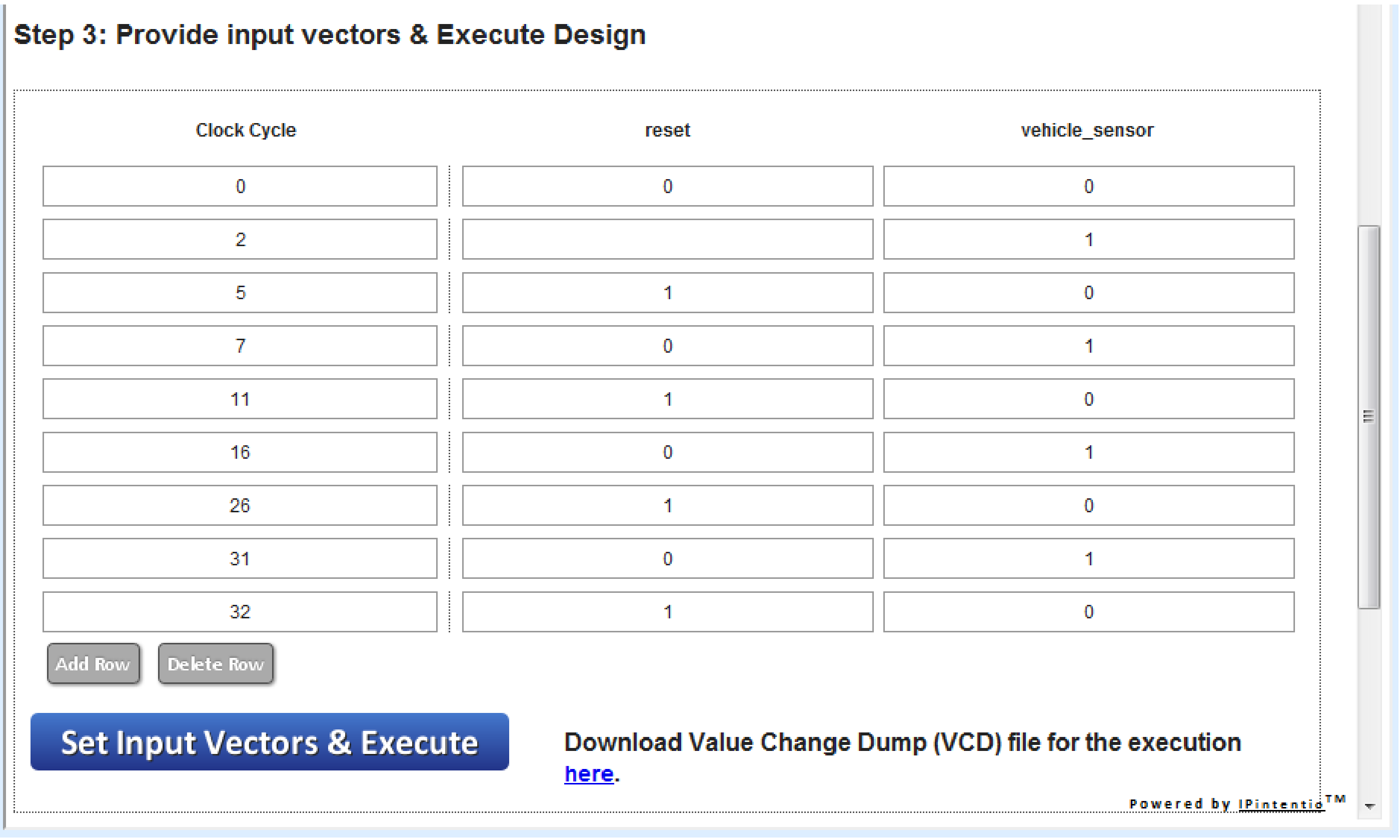
Fig. 2.6 - User clicks on “Set Input Vectors & Execute”

- This will generate board files which will take-in various inputs and convert all .c files to .v files (mentioned above).
- After all the .v files are generated, all the modules are compiled, synthesized and then pin assignment is done according to the device family of the Altera DE2-70 board in the Quartus II software in shell mode.
- To simplify this procedure, TCL script has been written which will create follow all the procedure of:
  1. Creating a project
  2. Selecting device family
  3. Defining top-level module and adding all automated FSM design
  4. Assigning pins of input, output, clk and reset to the Altera DE2-70 board
  5. Compile, synthesize and execute (execute means running the design on board)
- The TCL script is shown below:

TCL_script.tcl :

```
load_package flow

package require ::quartus::flow
project_new counter1 -overwrite

# Assign family, device, and top-level file
set_global_assignment -name FAMILY CycloneII
set_global_assignment -name DEVICE EP2C70F896C6
set_global_assignment -name VERILOG_FILE final.v
set_global_assignment -name VERILOG_FILE input_ram_dual.v
set_global_assignment -name VERILOG_FILE output_ram_dual.v
set_global_assignment -name VERILOG_FILE up_counter.v
set_global_assignment -name VERILOG_FILE baud_gen.v
set_global_assignment -name VERILOG_FILE rom3.v
set_global_assignment -name VERILOG_FILE uart_rx.v
set_global_assignment -name VERILOG_FILE uart_top.v
set_global_assignment -name VERILOG_FILE uart_tx.v
set_global_assignment -name TOP_LEVEL_ENTITY final

# Assign pins
set_location_assignment -to clk Pin_AD15
set_location_assignment -to glbl_rst Pin_AA23
set_location_assignment -to start Pin_AB26
set_location_assignment PIN_E21 -to ser_out

#compile the project
execute_flow -compile
qexec quartus_pgm -l
exec quartus_pgm --mode=jtag --cable=USB-Blaster --operation=p\;counter1.sof

#close project
project_close
```

- The system is configured such that the output is given to the UART.
- A script is written which will take-in this UART output from the DE2-70 board and interpret it in an 8-bit format. So the rest of the bits other than the output buts in the 8-bits are assigned zero.
- This 8-bit output is then converted in VCD format and is made available to the user for download.
- A log is also displayed which is actually the compilation report of quartus.



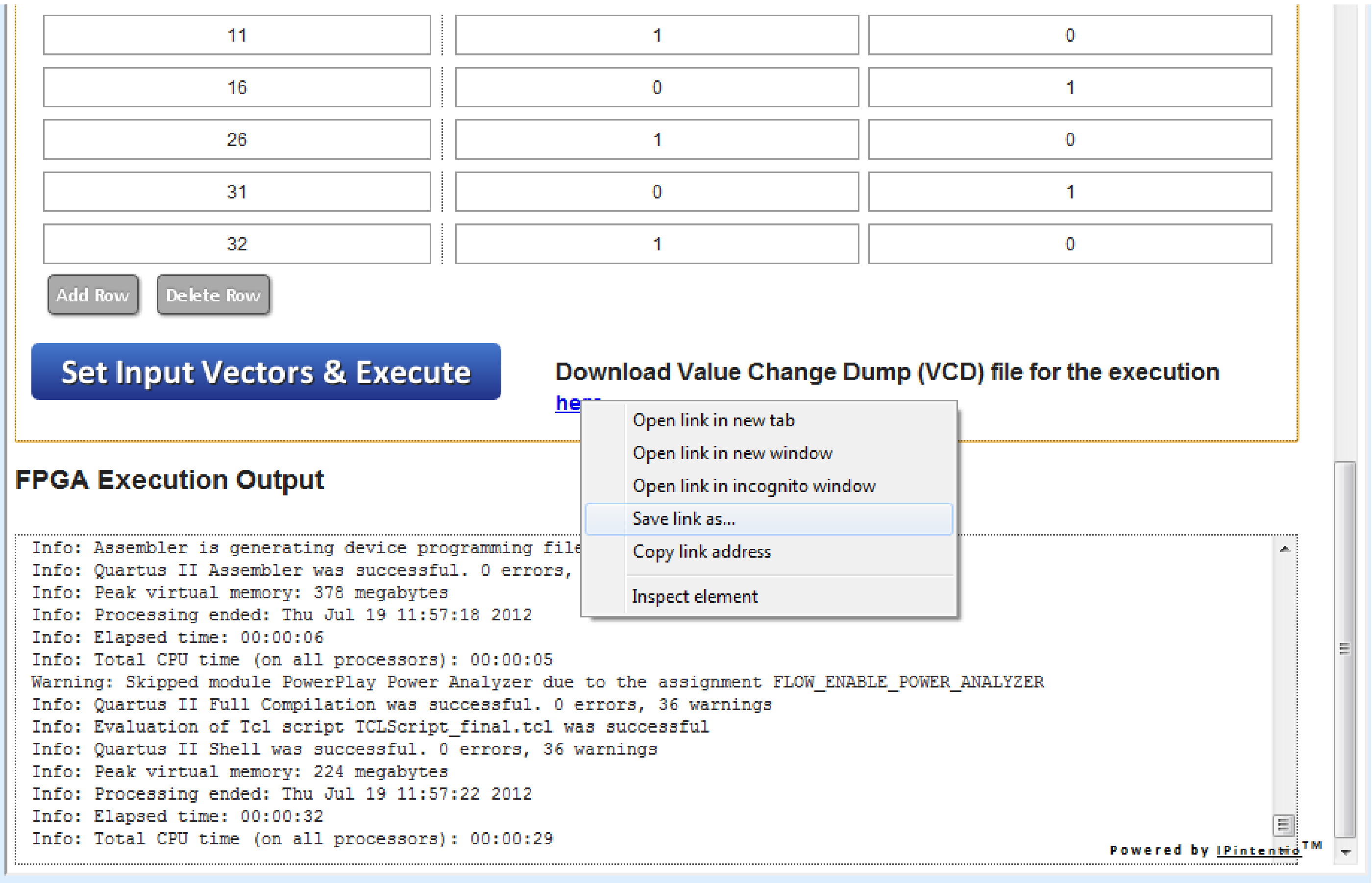
Step 3: Provide input vectors & Execute Design

Clock Cycle	reset	vehicle_sensor
0	0	0
2		1
5	1	0
7	0	1
11	1	0
16	0	1
26	1	0
31	0	1
32	1	0

Download Value Change Dump (VCD) file for the execution here.

Fig. 2.7 – VCD file ready for download

- Please check at the end of the log for possible error.
- If no error then download the VCD file.



11	1	0
16	0	1
26	1	0
31	0	1
32	1	0

Download Value Change Dump (VCD) file for the execution here

FPGA Execution Output

```
Info: Assembler is generating device programming file
Info: Quartus II Assembler was successful. 0 errors,
Info: Peak virtual memory: 378 megabytes
Info: Processing ended: Thu Jul 19 11:57:18 2012
Info: Elapsed time: 00:00:06
Info: Total CPU time (on all processors): 00:00:05
Warning: Skipped module PowerPlay Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
Info: Quartus II Full Compilation was successful. 0 errors, 36 warnings
Info: Evaluation of Tcl script TCLScript_final.tcl was successful
Info: Quartus II Shell was successful. 0 errors, 36 warnings
Info: Peak virtual memory: 224 megabytes
Info: Processing ended: Thu Jul 19 11:57:22 2012
Info: Elapsed time: 00:00:32
Info: Total CPU time (on all processors): 00:00:29
```

Fig. 2.8 – Download VCD file if no error

- This VCD file can then be viewed on the GTKWave waveform viewer and is illustrated below.
- Goto File>Open New Tab

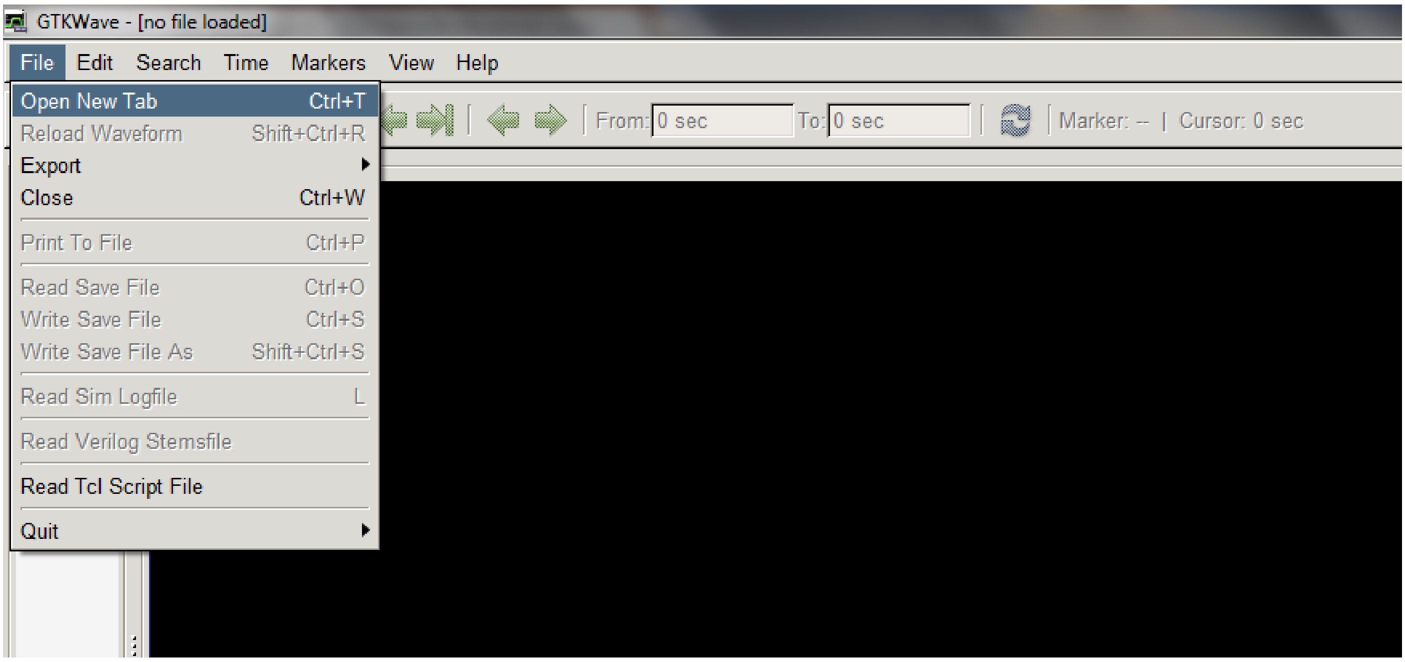
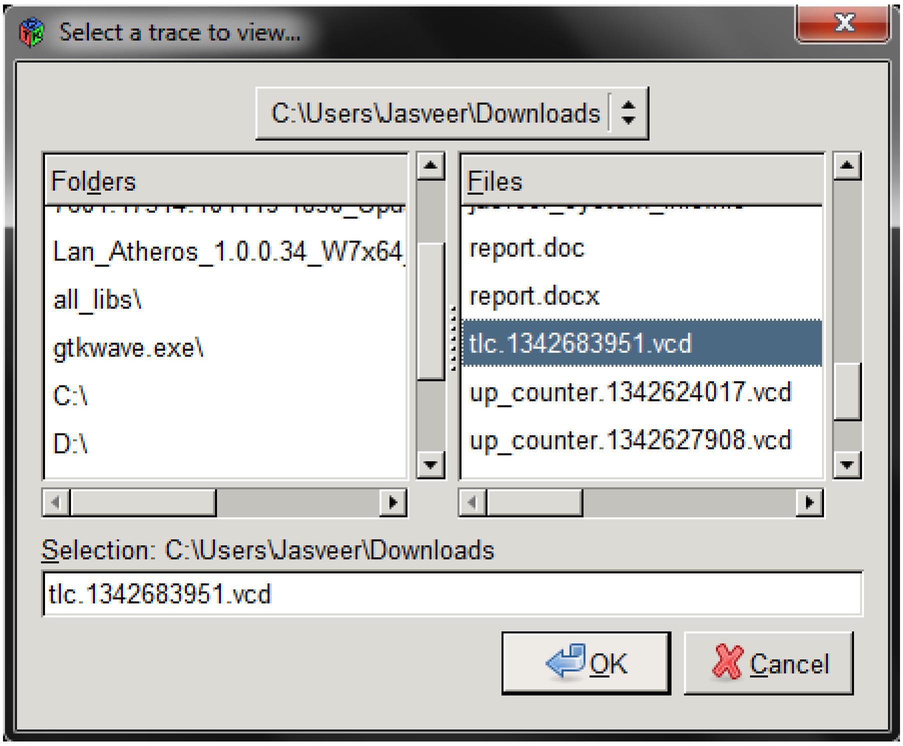


Fig. 2.8 – Open GTKWave waveform viewer (refer to GTKWave installation manual provided)

- Select the file from the directory in which .vcd file is saved.


Select a trace to view...
C:\Users\Jasveer\Downloads
Folders
Lan_Atheros_1.0.0.34_W7x64
all_libs\
gtkwave.exe\
C:\
D:\
Files
report.doc
report.docx
tlc.1342683951.vcd
up_counter.1342624017.vcd
up_counter.1342627908.vcd
OK
Cancel

Selection: C:\Users\Jasveer\Downloads

tlc.1342683951.vcd

Fig. 2.9 – Select vcd file to be viewed

- The top level module name is now displayed on the top left pane of the GTKWave window.

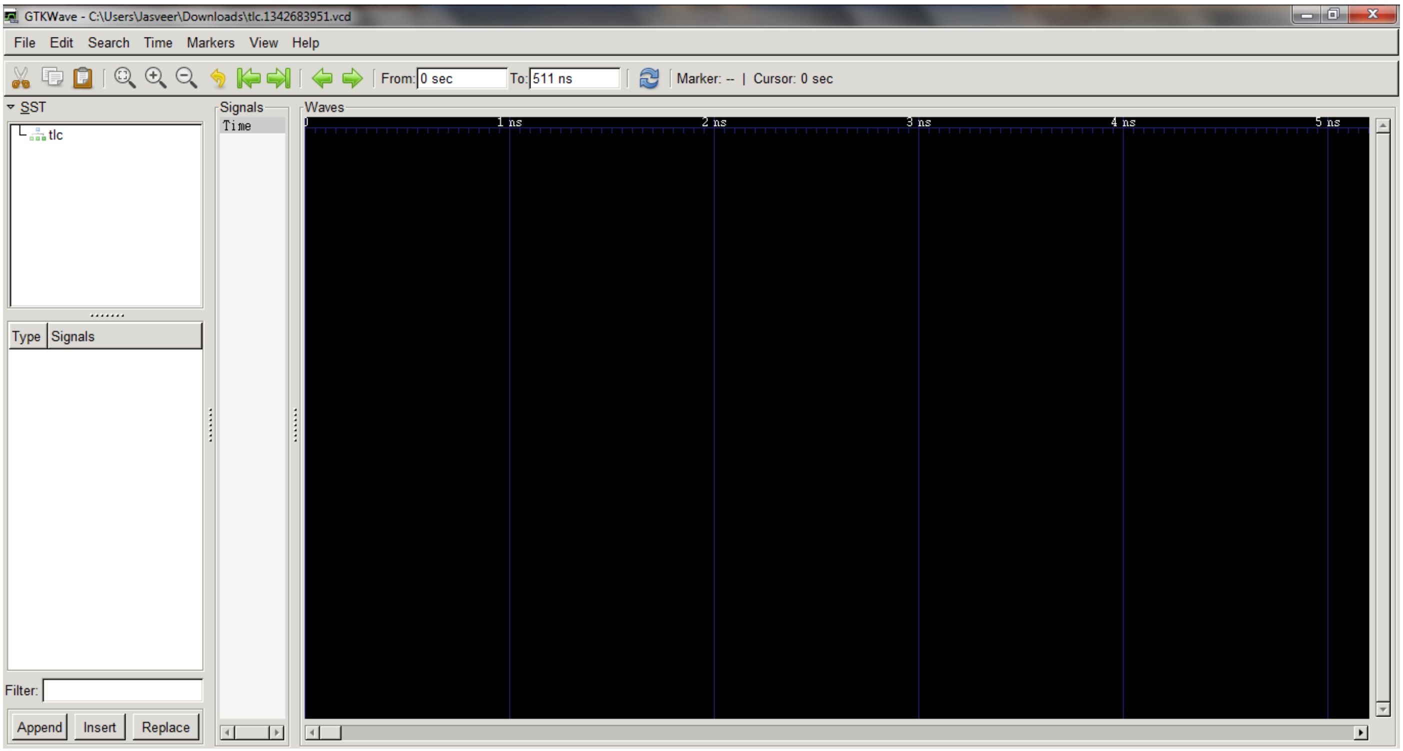

GTKWave - C:\Users\Jasveer\Downloads\tlc.1342683951.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 511 ns
Marker: -- | Cursor: 0 sec
SST
tlc
Type Signals
Filter:
Append Insert Replace
Signals
Time
Waves
1 ns
2 ns
3 ns
4 ns
5 ns

Fig. 2.10 – Select the top level module name displayed on the top left pane of the GTKWave window

- After selecting the top level module name displayed on the top left pane of the GTKWave window, the signals can be seen on the bottom left pane of the GTKWave window.

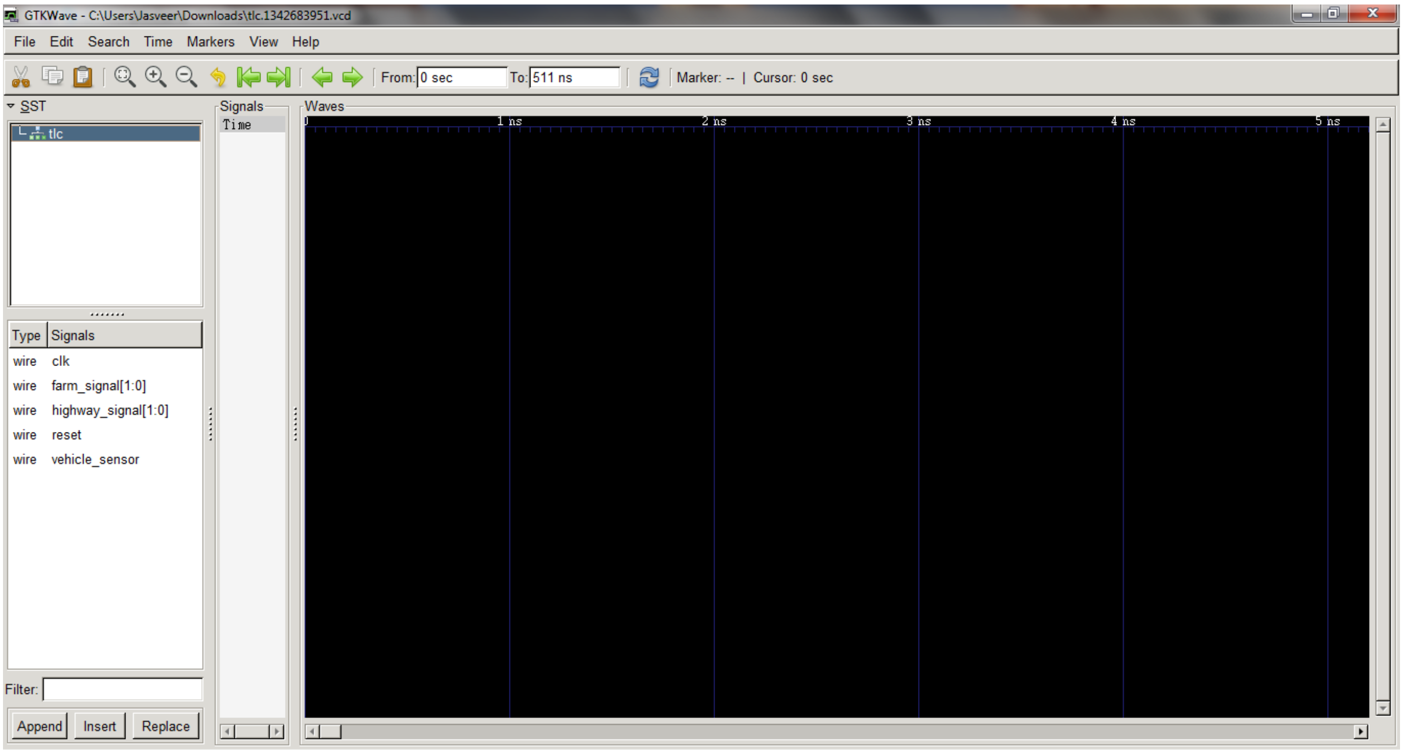


Fig. 2.11 – Open GTKWave waveform viewer (refer to GTKWave installation manual provided)

- Select all of the signals and press the button Append.

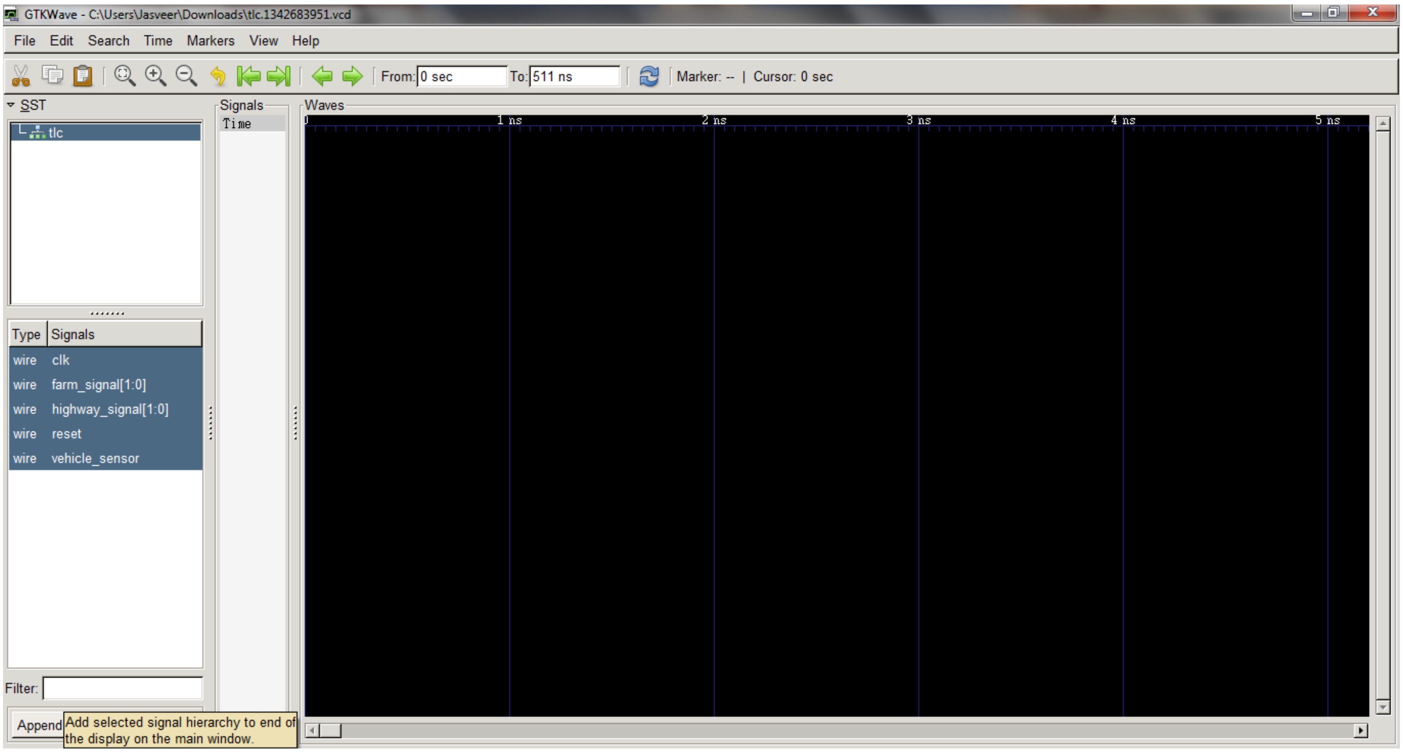
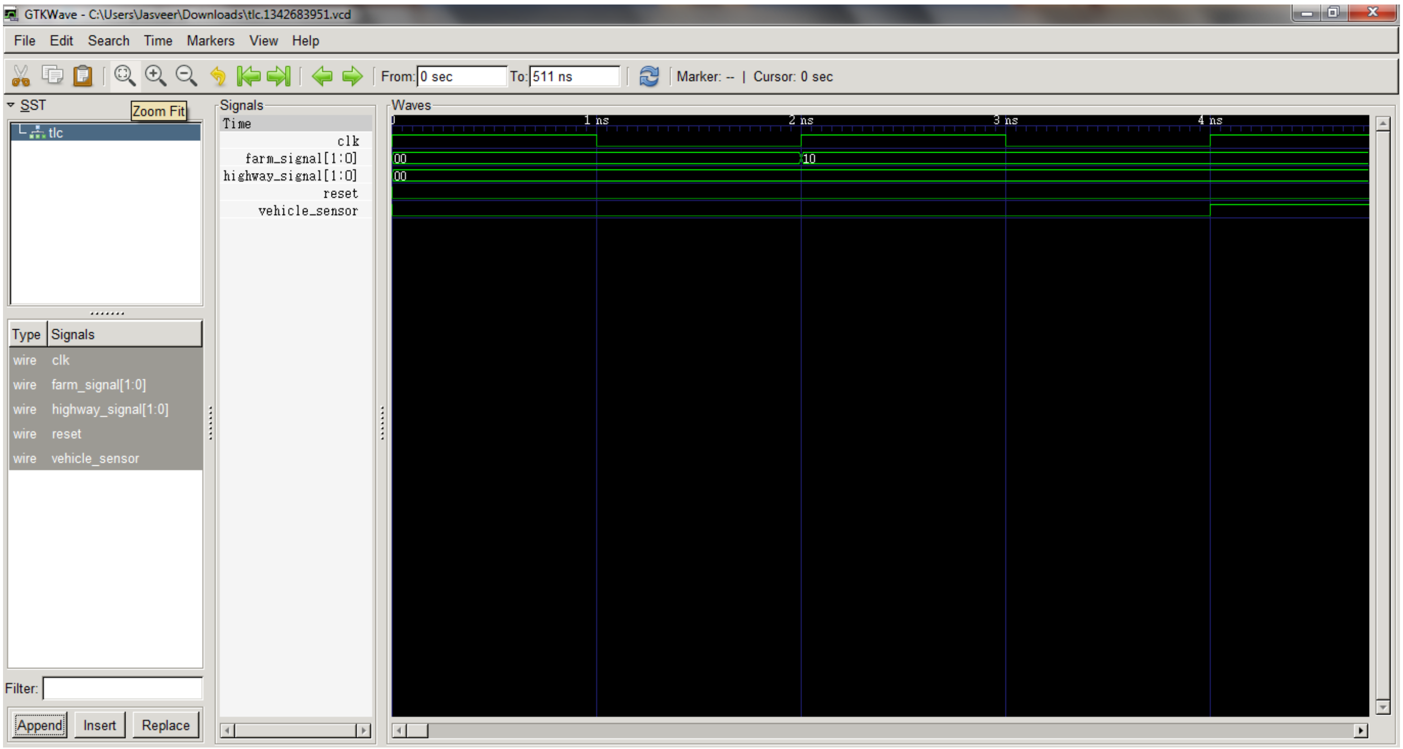

GTKWave - C:\Users\Jasveer\Downloads\tlc.1342683951.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 511 ns
Marker: -- | Cursor: 0 sec
SST
tlc
Type Signals
wire clk
wire farm_signal[1:0]
wire highway_signal[1:0]
wire reset
wire vehicle_sensor
Filter:
Append
Add selected signal hierarchy to end of the display on the main window.
Signals
Time
Waves
1 ns
2 ns
3 ns
4 ns
5 ns

Fig. 2.12 – Select all of them and press append



GTKWave - C:\Users\Jasveer\Downloads\tlc.1342683951.vcd

From: 0 sec To: 511 ns Marker: -- | Cursor: 0 sec

Fig. 2.12 – The waveform can be seen now

- Use the button, 🔍 called Zoom-fit to see the whole 256 clock cycles on your screen.

- Use the button, 🔍 called Zoom-in to see larger view i.e. less no. of cycles on your screen.

- Use the button, 🔍 called Zoom-out to see the large no. of cycles on your screen.

Constraints:

- The user can see maximum upto 256 clock cycles.
- Total of all input pins cannot be more than 16.
- Total of all the output pins cannot be more than 8.