

Two Way Switch

May 26, 2020



Mentors

Prasad T
Aditya Gudla
Simranjeet Singh

Interns

Ajay Chaudhari
Chethan T Bhat
Ritvik Tiwari
Karthik A Shet

Contents

1	Introduction	3
1.1	What is a Two Way Switch ?	3
1.2	X-OR Gate	3
2	Circuit Diagram	4
3	Code in VHDL & Verilog	4
3.1	Verilog Code(RTL Description)	4
3.2	Testbench Code(Verilog)	5
3.3	VHDL Code(RTL Description)	6
3.4	Testbench Code(VHDL)	7
4	Implementing on quartus II	9
5	RTL Circuit of the implemented design	13
6	Pin Assignment	15
7	Downloading the code to DE0 Nano FPGA Board	17
8	Implementing on Modelsim	21
9	Testing the Design	26
9.1	Simulation waveform of the Verilog Design	26
9.2	Simulation waveform of the VHDL Design	26

1 Introduction

1.1 What is a Two Way Switch ?

A two way switch can be explained using a simple example. Consider a room with two switches and one bulb. Now, If both the switches are in OFF state or ON state the bulb will not glow. But, If either of the two switch is ON the bulb will glow. If we have to relate this with digital electronics, the logic can be realised using an X-OR gate.

1.2 X-OR Gate

XOR gate (pronounced as Exclusive OR) is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is odd. An XOR gate implements an exclusive or; a true output results if one, and only one, of the inputs to the gate is true. If both inputs are false (0/LOW) or both are true, a false output results.

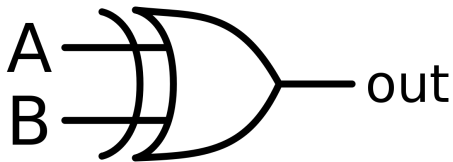


Figure 1: Logic Symbol

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2: Truth Table

2 Circuit Diagram

Note: LED does not conduct in both direction. This circuit is just a representation of the logic

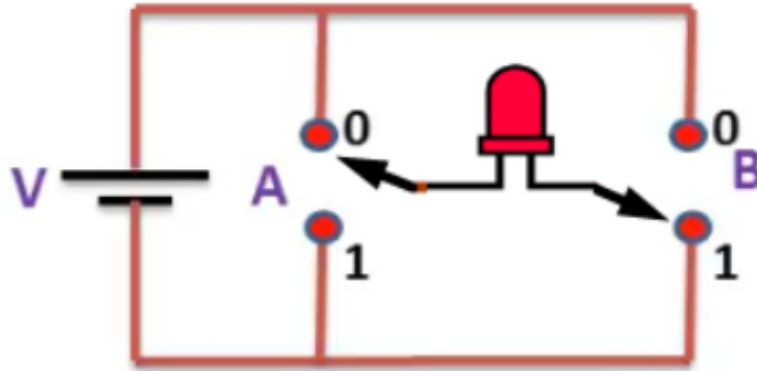


Figure 3: Two way switch Circuit Diagram.

3 Code in VHDL & Verilog

3.1 Verilog Code(RTL Description)

Dataflow Modelling:

```
module TWS_Dataflow(  
    input s1,s2, //Define two Inputs pins  
    output X1); //Define one output pin  
    assign X1 = (s1^s2); //XOR the 2 input pins(s1,s2) and assign  
                        //the output to the output pin(X1)  
endmodule
```

Behavioural Modelling:

```
module TWS_Behavioural(  
    input s1,s2, //Define two inputs(switches)  
    output reg X1 //Define the output pin  
);  
  
always @ (s1,s2) //Every time the input changes, the output  
                //is to be updated  
begin  
    if (s1==s2) //if both switches are in same position,  
                //output is LOW  
        X1 = 0;  
    else  
        X1=1; //if the position of switches are not equal,  
                //output is HIGH  
end  
endmodule
```

3.2 Testbench Code(Verilog)

```
module TWS_tb;  
    reg s1;reg s2; //define input  
    wire X1; //define output  
  
    TWS_Behavioural uut(.s1(s1),.s2(s2),.X1(X1)); //Map testbench ports with  
                                                    //DUT ports  
  
    initial begin  
        s1 = 0; s2 = 0;#100; //different combinations of input  
        s1 = 0; s2 = 1;#100;  
        s1 = 1; s2 = 0;#100;  
        s1 = 1; s2 = 1;#100;  
        #100;  
    end  
endmodule
```

3.3 VHDL Code(RTL Description)

Dataflow Modelling:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity two_way_switch is
    Port ( A :in STD_LOGIC; -- Switch 1
          B :in STD_LOGIC; -- Switch 2
          OP :out STD_LOGIC -- LED output
          );
end two_way_switch;

architecture data_flow of two_way_switch is
begin
    OP <= A xor B; -- XOR logic used for two way switch
end data_flow;
```

Behavioural Modelling:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity two_way_switch is
    Port ( A :in STD_LOGIC; -- Switch 1
          B :in STD_LOGIC; -- Switch 2
          OP :out STD_LOGIC -- LED output
          );
end two_way_switch;

architecture behavioral of two_way_switch is
begin
    process (A, B) --used for behavioral modelling
    begin
        If (A/=B) then --if either one of the switches is ON
            OP <= '1'; --LED will glow
        else --if both the switches are ON or OFF
            OP <= '0'; --LED will not glow
        end if;
    end process;
```

```
end behavioral;
```

3.4 Testbench Code(VHDL)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY tb_two_way_switch IS --defining the entity
END tb_two_way_switch;
ARCHITECTURE Behavioral OF tb_two_way_switch IS --defining the
--architecture
    COMPONENT two_way_switch --two way switch design
    PORT(
        A :IN std_logic; --Switch 1
        B :IN std_logic; --Switch 2
        OP :OUT std_logic --LED Output
    );
    END COMPONENT;

    signal A :std_logic := '0'; --Stimulus signal for switch 1
    signal B :std_logic := '0'; --Stimulus signal for switch 2
    signal OP :std_logic; --Output signal
BEGIN
    uut: two_way_switch PORT MAP ( --defining unit under test i.e
        --two way switch
        A => A,
        B => B,
        OP => OP
    );
    stim_proc: process --begining stimulation process
    begin

        A <= '0'; -Different combination of input to simulate the logic.
        B <= '0';

        wait for 50 ns;
        A <= '0';
        B <= '1';

        wait for 50 ns;
        A <= '1';
```

```
B <= '0';

    wait for 50 ns;
A <= '1';
B <= '1';

    wait for 50 ns;
        wait;
    end process;

END;
```

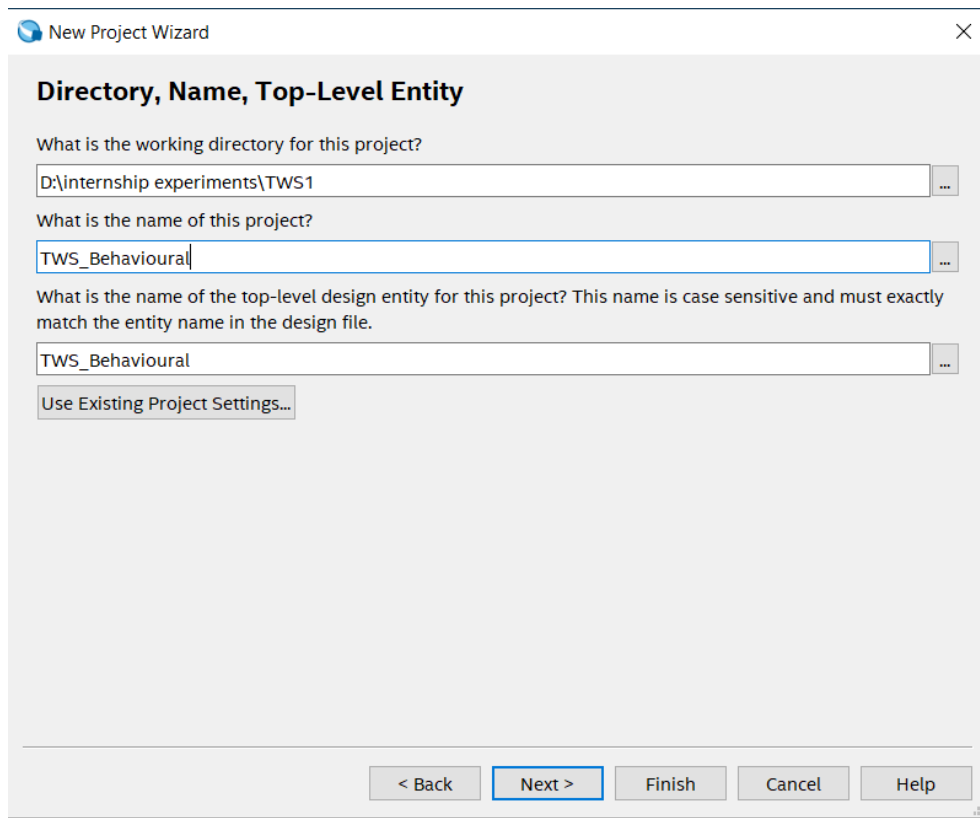

4 Implementing on quartus II

To Implement the design in Quartus Software, follow these steps(For more detailed information on the following steps, refer 'Quick Start Guide to Quartus and Model-Sim Software' Document):

Note: The language used is Verilog HDL. Device is DE0-Nano FPGA Board.

Family:Cyclone IV E; **Chip Name:**EP4CE22F17C6

1. Start a new project in Quartus Lite software



Directory, Name, Top-Level Entity

What is the working directory for this project?

D:\internship experiments\TWS1

What is the name of this project?

TWS_Behavioural

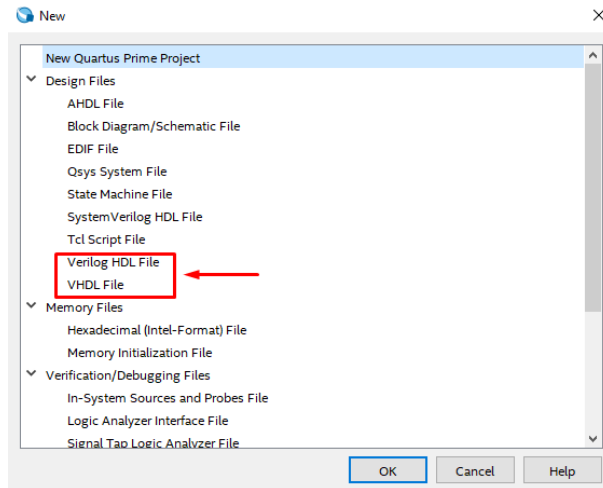
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

TWS_Behavioural

Use Existing Project Settings...

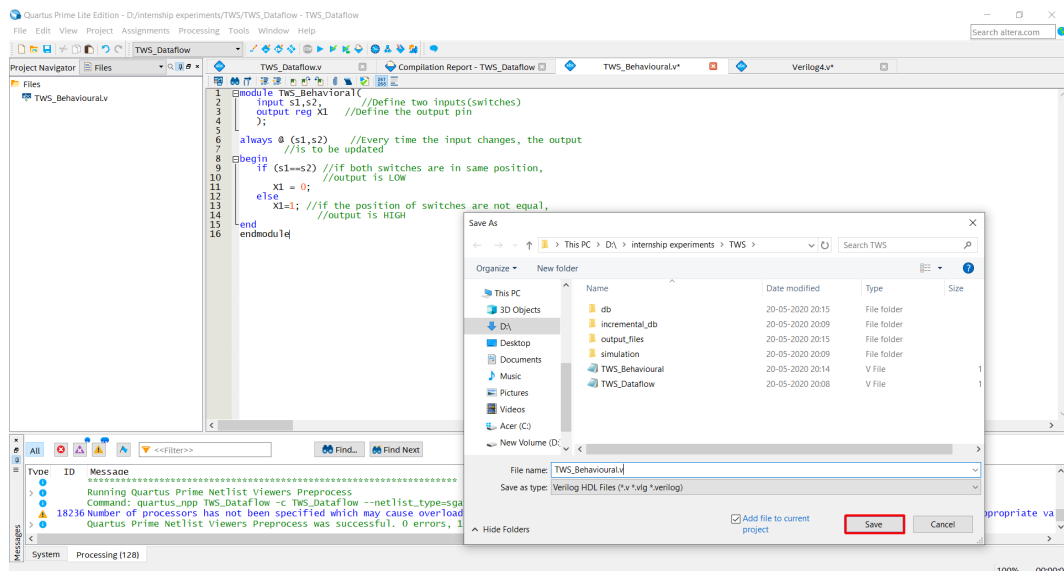
< Back Next > Finish Cancel Help

2. Create a new VHDL or Verilog file. Here we create a Verilog file for demonstration, Same procedure can be followed to VHDL file.

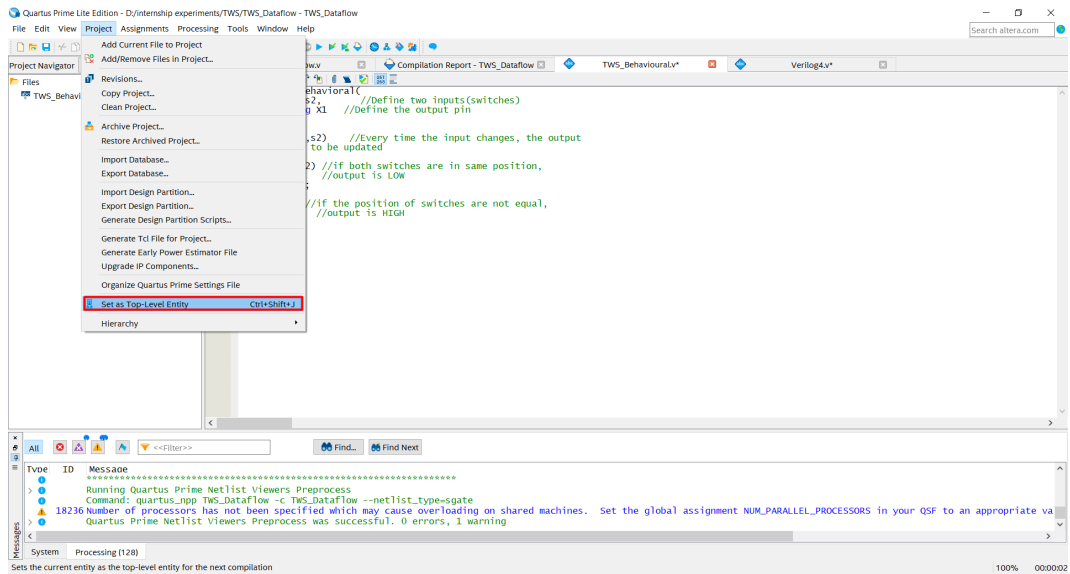


3. Type the above main code in this file and make sure that the Module name and the file name is same.

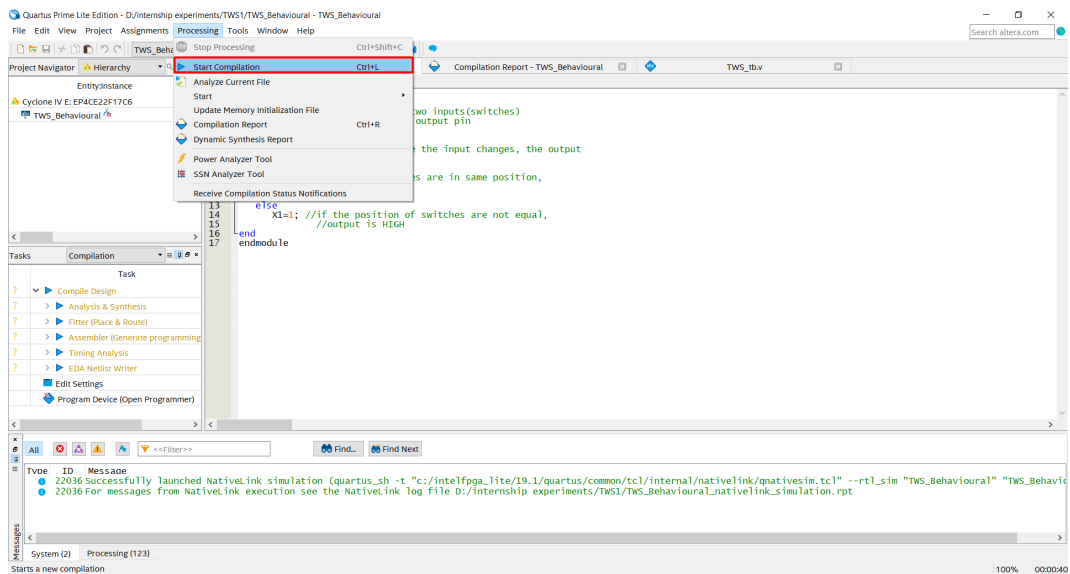
Note: We have used the Behavioural Verilog code for this demonstration



4. Go to 'Project' → 'Set as Top-Level Entity'.



5. Go to 'Processing' → 'Start Compilation'.



6. After successful compilation, you will see these Green check marks.

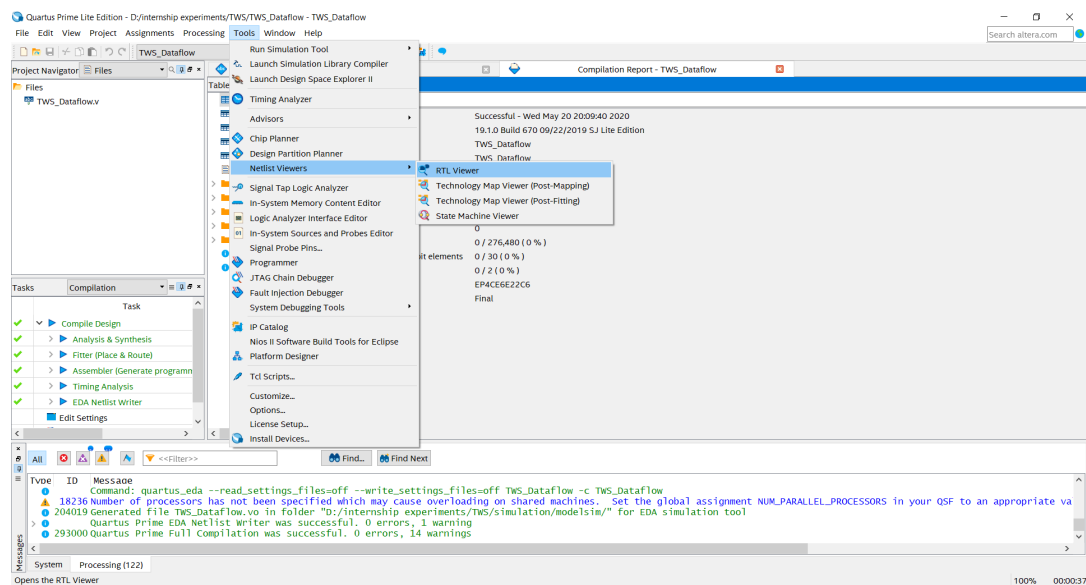
	Task
✓	▼ ▶ Compile Design
✓	> ▶ Analysis & Synthesis
✓	> ▶ Fitter (Place & Route)
✓	> ▶ Assembler (Generate programming)
✓	> ▶ Timing Analysis
✓	> ▶ EDA Netlist Writer
	🔧 Edit Settings
	🔌 Program Device (Open Programmer)

5 RTL Circuit of the implemented design

The below two figures show the RTL Circuit of the two different type of modelling (Dataflow and behavioural). As it can be seen , both types of modelling resulted in the same type of design. Although this is not possible for more complex designs(Behavioural modelling results in a more complex circuit than Dataflow or Structural), intelligent and smart behavioural design can lead to really well optimised circuits.

Steps to get RTL circuit.

1. Go to 'Tools' → 'Netlist Viewers' → 'RTL Viewer'.



2. The below figure shows the equivalent RTL circuit in both Dataflow and Behavioural modelling styles.

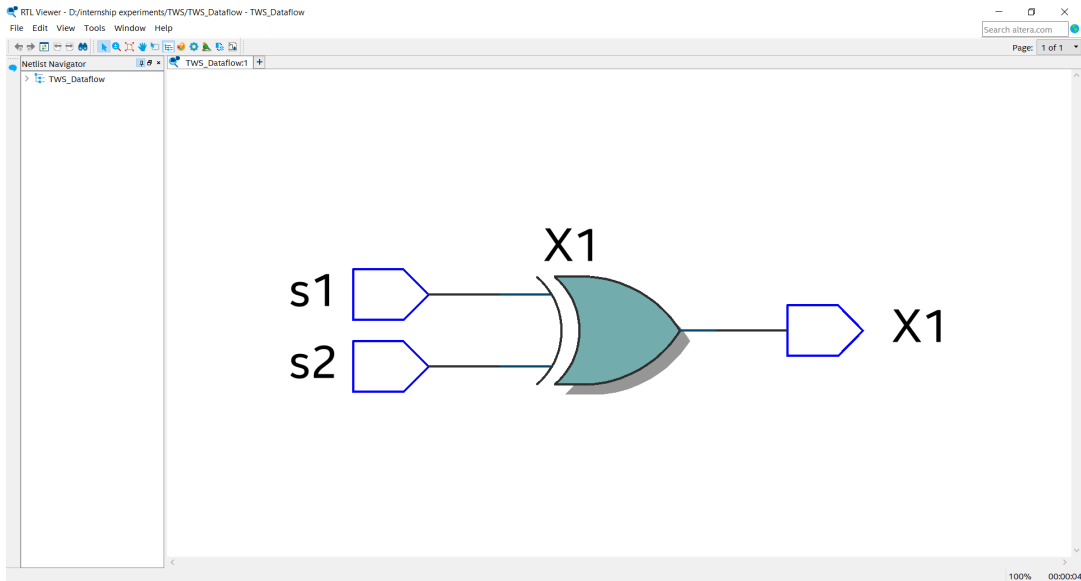


Figure 4: Dataflow modelling

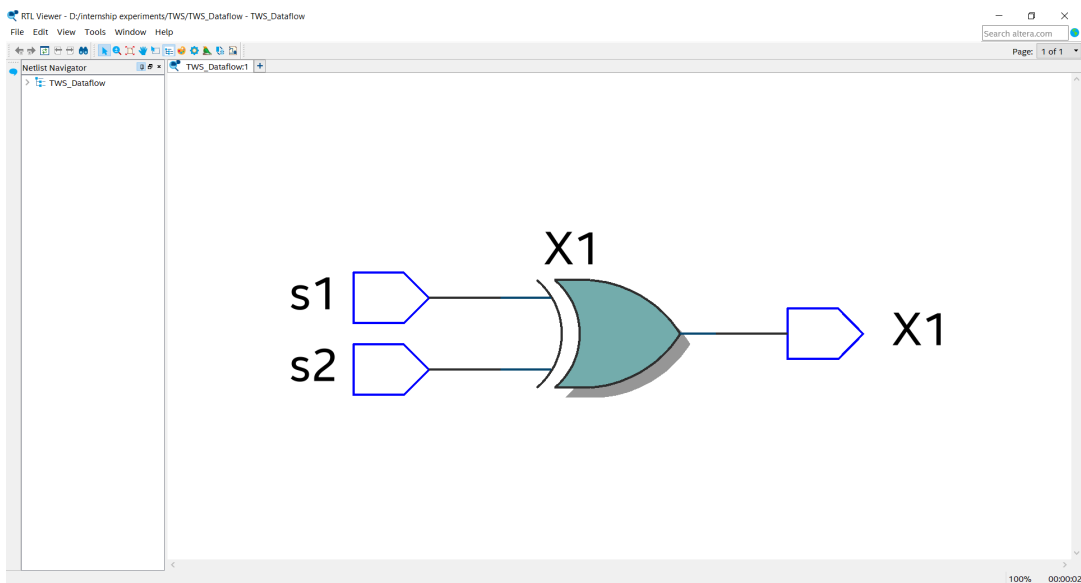
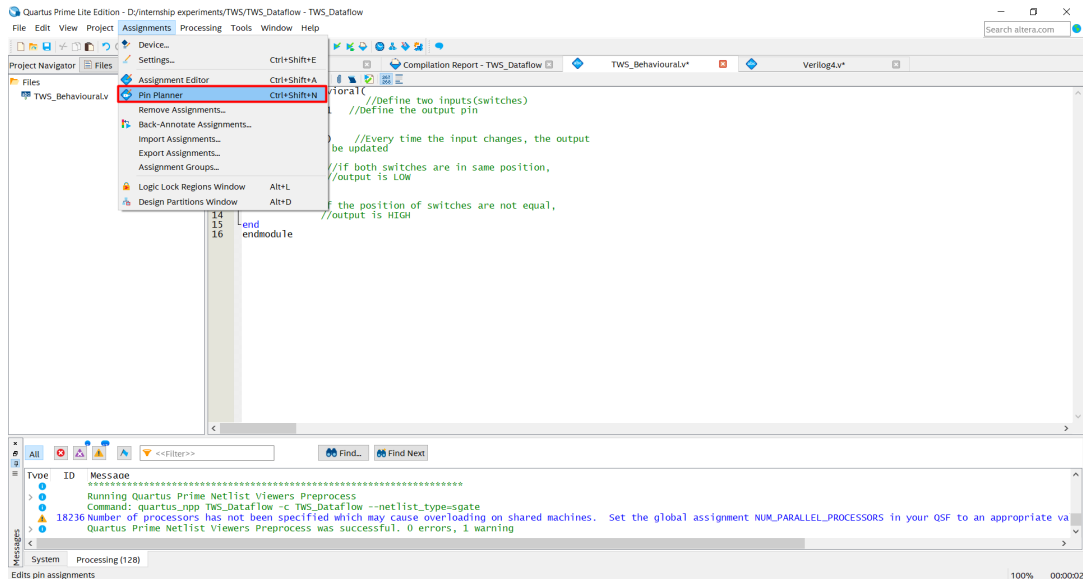


Figure 5: Behavioural modelling

6 Pin Assignment

1. Click on 'Assignments' → 'Pin Planner', This Pin Planner shows the I/O ports that we have created in our design.



2. Now we have to assign each I/O ports with the respective Pin numbers, which can be found on the Device Manual which we are implementing. Here we are referring to the DEO NANO Board .Out of 8 LED present in the DEO NANO Board We will select **LED[0]** as the output 'X1' of the XOR gate ,whose PIN number is **PIN_A15**. And out of 4 DIP Switches present in DEO NANO Board we use two DIP switches **DIP_SWITCH[0]** and **DIP_SWITCH[1]** for input signal 'S1' and 'S2',whose PIN numbers are **PIN_M1** and **PIN_T8** respectively.

Pin Planner - D:\internship experiments\TWS\TWS_Dataflow - TWS_Dataflow

File Edit View Processing Tools Window Help

Groups

Named: *

Node Name Direction

<<new group>>

Tasks

Early Pin Planning

Run I/O Assignment

Export Pin Assignment

Pin Finder...

Highlight Pins

I/O Banks

VREF Groups

Edges

Clock Pins

Clock

Top View - Wire Bond

Cyclone IV E - EP4CE22F17C6

Pin Legend

Symbol Pin Type

User I/O

User assign...

Filter assign...

Unbonded...

Reserved pin

Other confi...

DEV_OE

DEV_CLR

DIFF_n

DIFF_p

DQ

DQS

CLK_n

CLK_p

Other PLL

Other dual...

MSEL0

MSEL1

MSEL2

CONF_DONE

nCE

nCONFIG

TDI

TCK

TMS

TDO

Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	itter Locator	I/O Standard	Reserved	urrent Streng	Slew Rate	fferential Pai	ict Preservati
X1	Output	PIN_A15	7	B7_NO	PIN_D1	2.5 V...fault		8mA (default)	2 (default)		
s1	Input	PIN_M1	2	B2_NO	PIN_G5	2.5 V...fault		8mA (default)			
s2	Input	PIN_TB	3	B3_NO	PIN_F3	2.5 V...fault		8mA (default)			
<<new node>>											

0% 00:00:00

3. Go to 'Processing'→ 'Start I/O Assignment Analysis'.

Pin Planner - D:\internship experiments\TWS\TWS_Dataflow - TWS_Dataflow

File Edit View **Processing** Tools Window Help

Groups

Named: *

Node Name Direction

<<new group>>

Tasks

Early Pin Planning

Run I/O Assignment

Export Pin Assignment

Pin Finder...

Highlight Pins

I/O Banks

VREF Groups

Edges

Clock Pins

Clock

Top View - Wire Bond

Cyclone IV E - EP4CE22F17C6

Pin Legend

Symbol Pin Type

User I/O

User assign...

Filter assign...

Unbonded...

Reserved pin

Other confi...

DEV_OE

DEV_CLR

DIFF_n

DIFF_p

DQ

DQS

CLK_n

CLK_p

Other PLL

Other dual...

MSEL0

MSEL1

MSEL2

CONF_DONE

nCE

nCONFIG

TDI

TCK

TMS

TDO

Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	itter Locator	I/O Standard	Reserved	urrent Streng	Slew Rate	fferential Pai	ict Preservati
X1	Output	PIN_A15	7	B7_NO	PIN_D1	2.5 V...fault		8mA (default)	2 (default)		
s1	Input	PIN_M1	2	B2_NO	PIN_G5	2.5 V...fault		8mA (default)			
s2	Input	PIN_TB	3	B3_NO	PIN_F3	2.5 V...fault		8mA (default)			
<<new node>>											

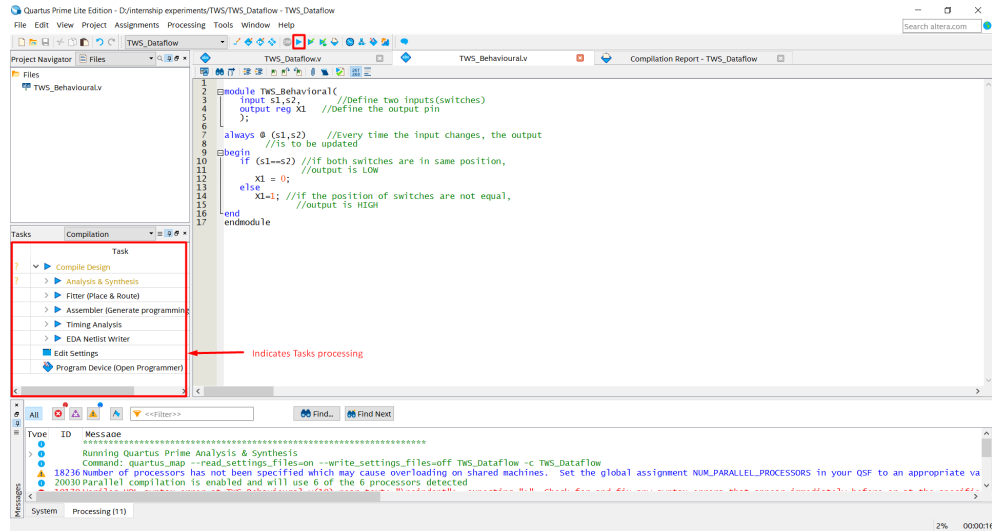
Analyzes I/O assignments for legal placement on the device

0% 00:00:00

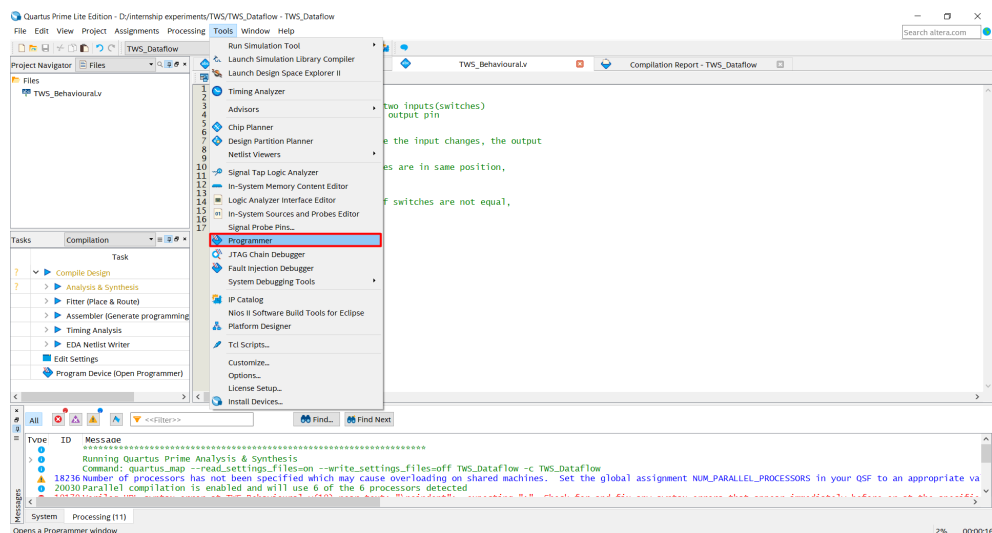
7 Downloading the code to DE0 Nano FPGA Board

Before starting, Make sure the board is powered ON and connected to the computer through an USB Cable

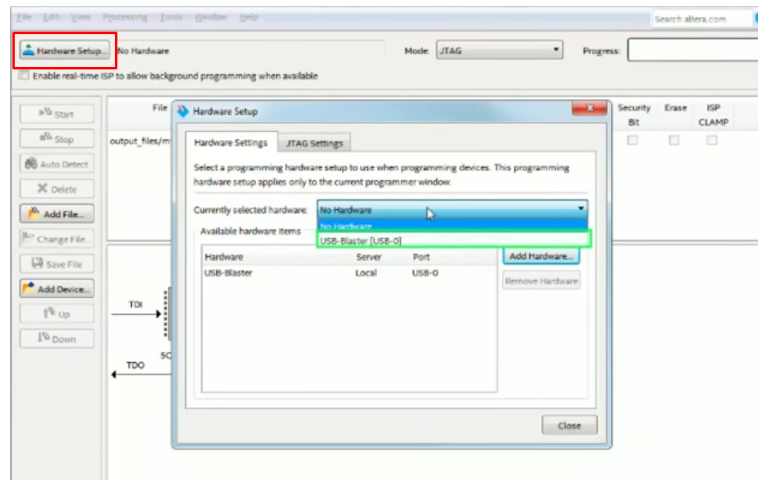
1. **'Compile'** the Project by double clicking on compile or the **'Play'** button. This creates an SRAM object file(.SOF file). This file is used to program the Device



2. Open the programmer by going to **'Tools'→ 'Programmer'**

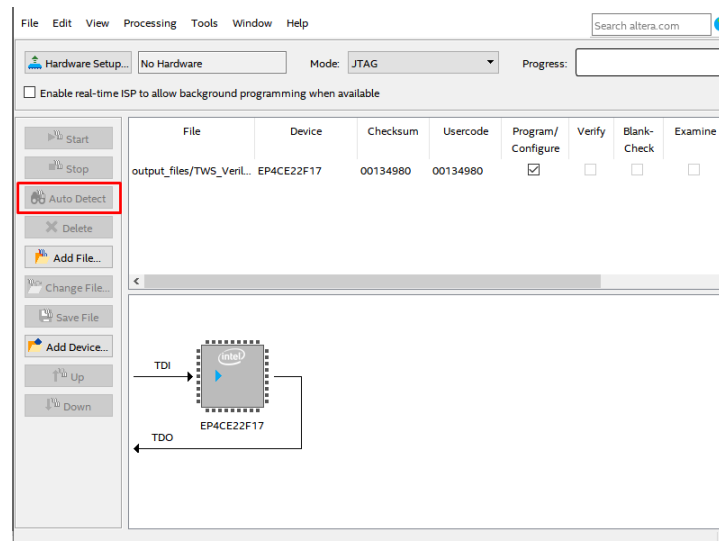


3. Click on '**Hardware Setup**'. The Device required must be listed under "Currently available hardware". If not, check if the device drivers are correctly installed. Choose the hardware from the dropdown menu

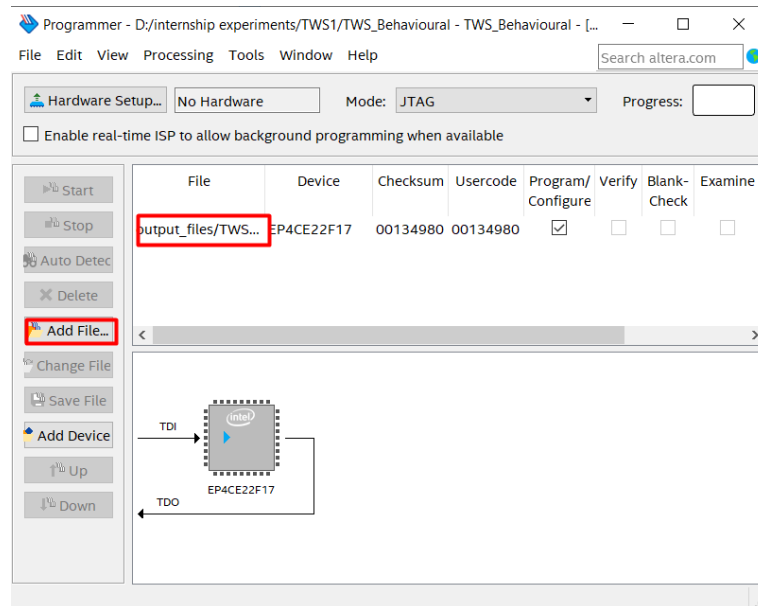


4. In case you didn't find the '**USBBLASTER**' option. Click on '**AUTO DETECT**' to see if the program can find the device. If it still couldn't find it, check if the device drivers are correctly installed. Refer your device's manual for more information.

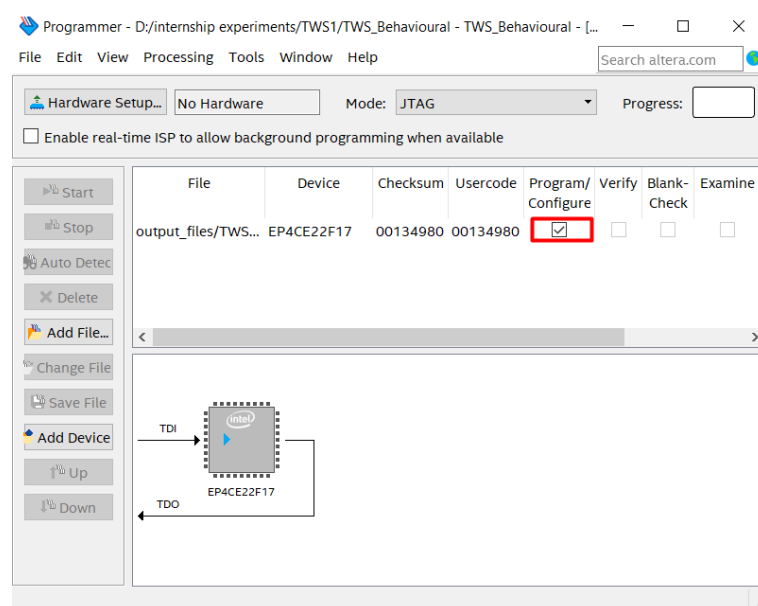
Note:The '**AUTO DETECT**' Button will be highlighted when you connect your board to your computer



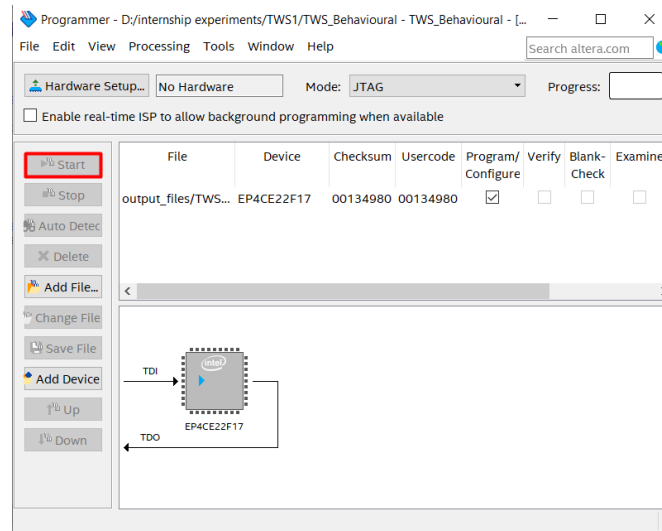
5. If the file is not listed, it can be manually added by clicking on add file. The .SOF file can be found the output directory inside the project directory



6. Make sure the 'Program/Configure' checkbox is ticked



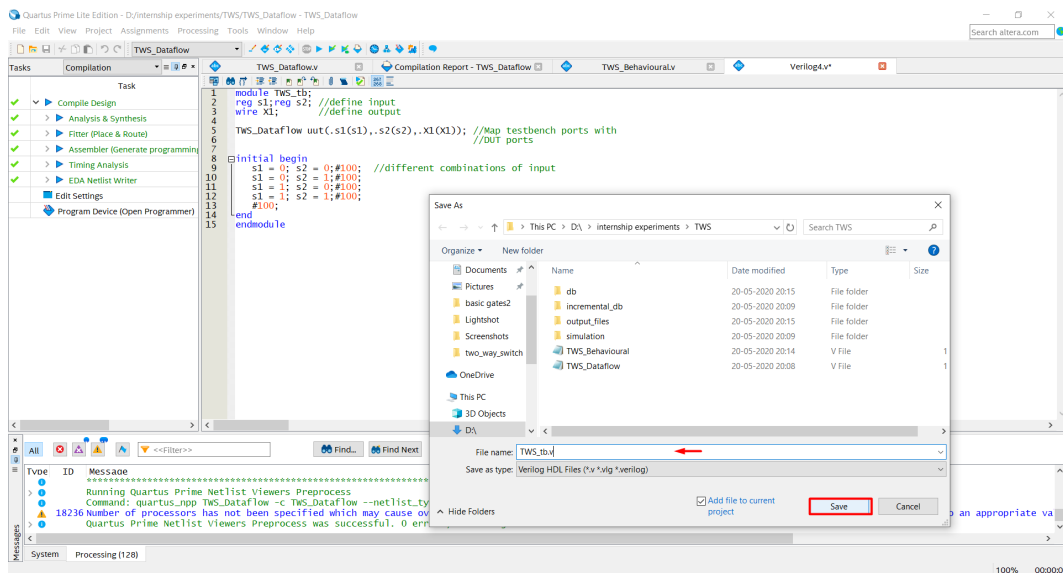
7. When ready, click on '**Start**' to start the programming process. '**Start**' button will be enabled when the 'DEO NANO' board is connected to USB port of your device.



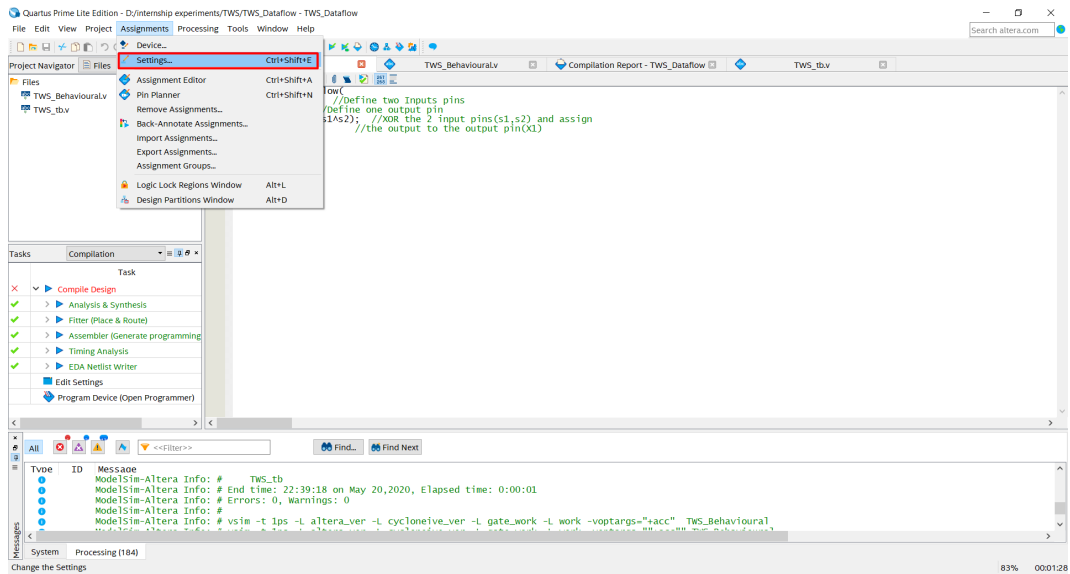
8 Implementing on Modelsim

The TestBench shown here is a Verilog TestBench. For more detailed procedure on using ModelSim, Refer '**Quick Start Guide to Quartus and ModelSim Software**' Document

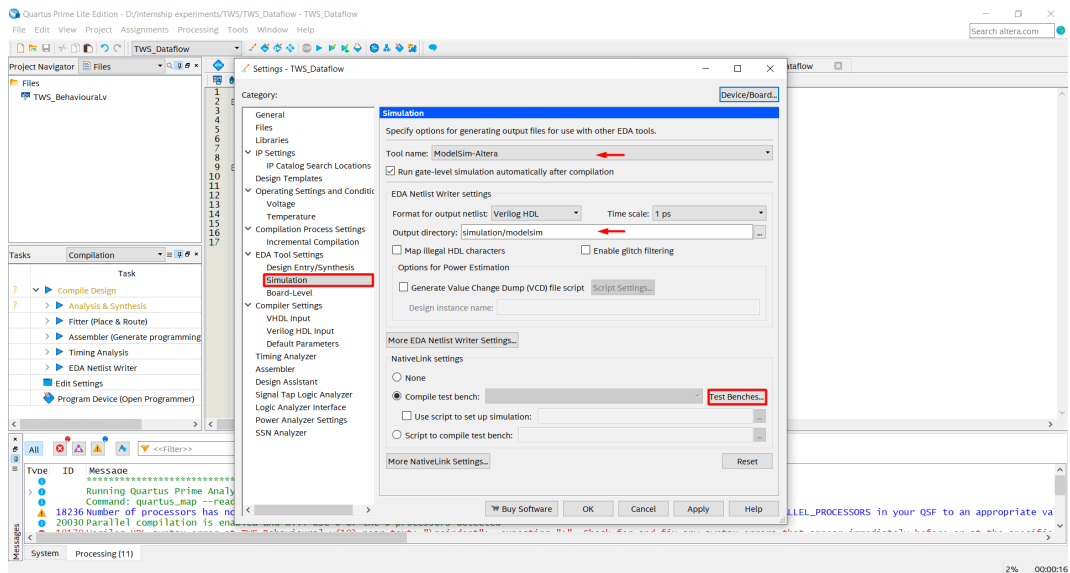
1. Create a **NEW** Verilog file in Quartus Prime.Type in the TestBench code provided in this document and **SAVE** the file with the same name as the module name



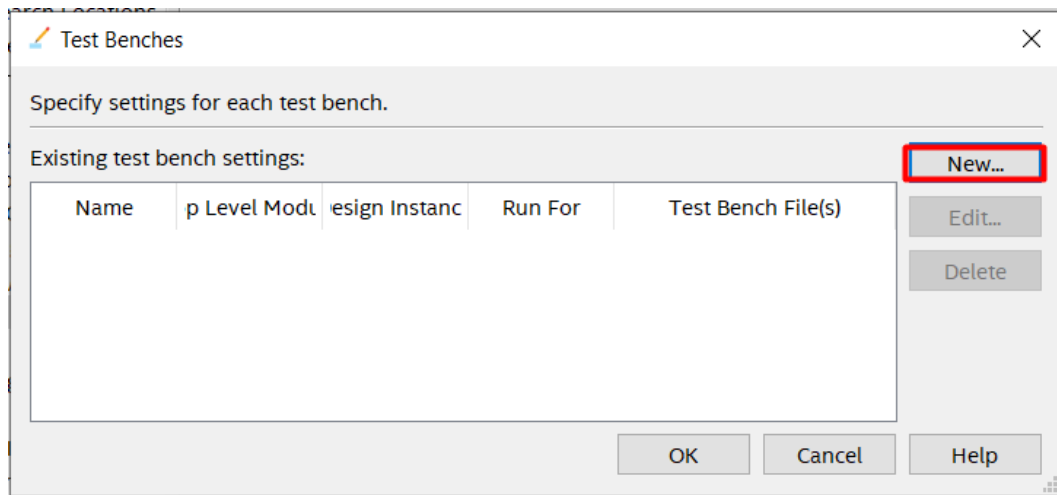
2. Go to 'Assignments'→'Settings'.



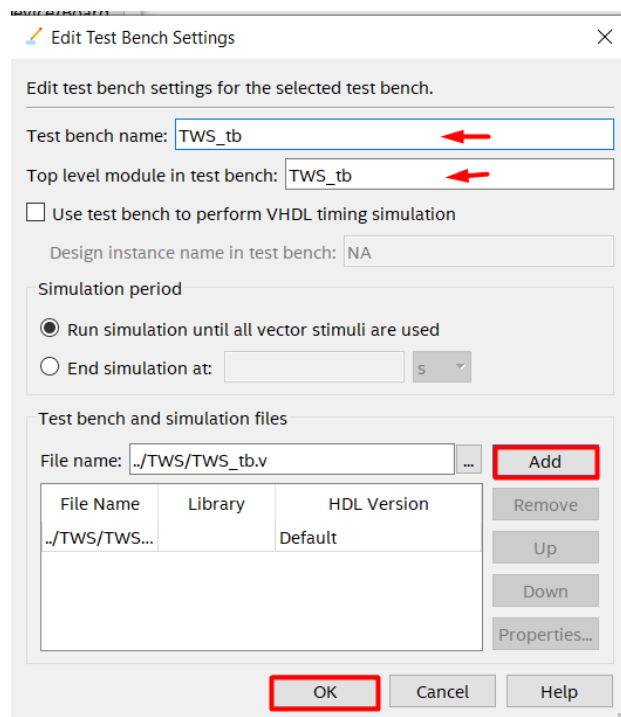
3. Navigate to 'Simulation' under 'EDA Tool Settings'. Set the language as Verilog HDL. Select 'Compile Test Bench' and then click on 'Test Benches'.



4. Click on 'New', this opens another dialogue box.

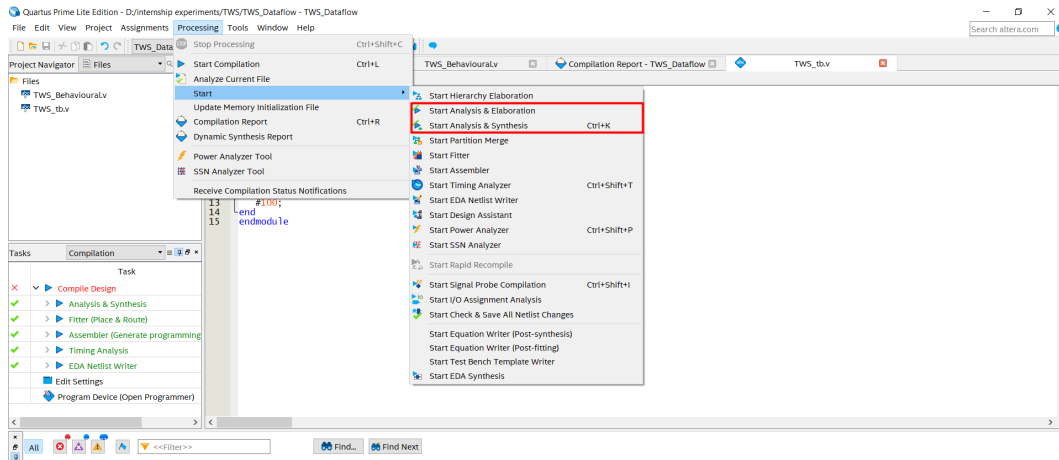


5. Now type in the testbench name(In this design , its **TWS_tb**). Now click on the highlighted browse button. Find the testbench file(it can be found in the project directory) and click on 'Open'. Now click on 'Add',then 'OK'.

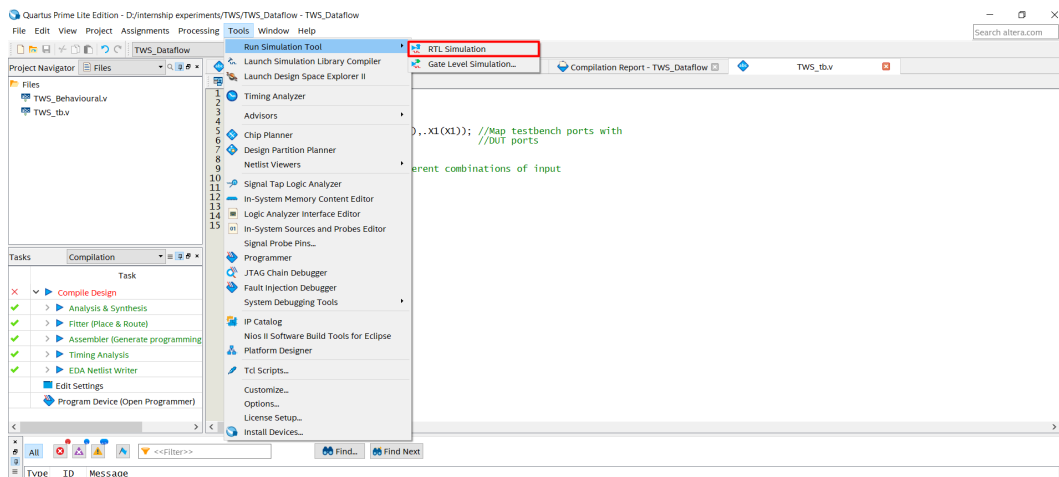


Functional Simulation using NativeLink Feature

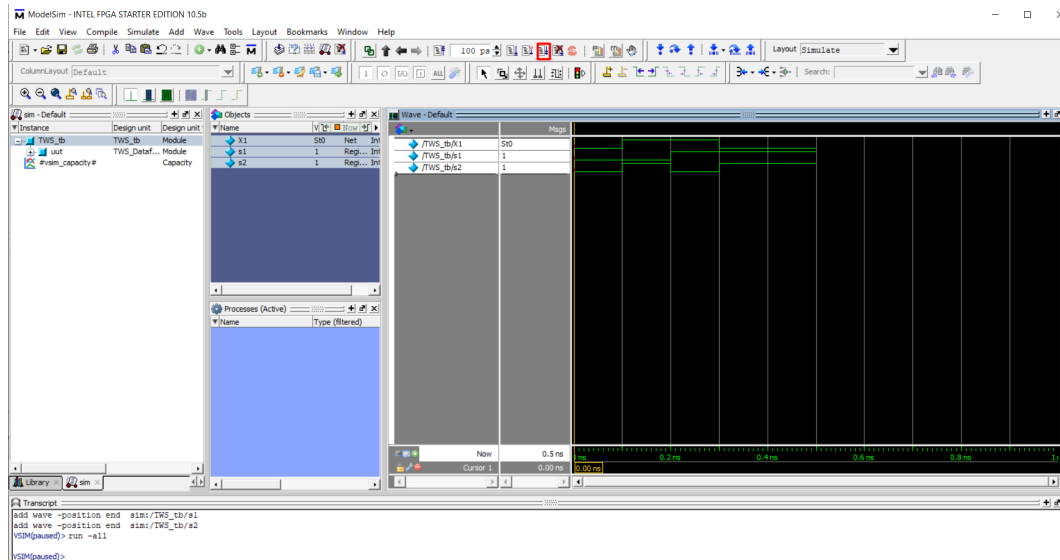
1. Go to menu '**Processing**' → '**Start**' → '**Start Analysis & Elaboration**'. After this click on '**Start Analysis & Synthesis**' on the same drop box.



2. Go to menu '**Tools**' → '**Run Simulation Tool**' → '**RTL Simulation**' to automatically run the EDA simulator (ModelSim-Altera) and to compile all necessary design files.



3. Finally ModelSim-Altera tool opens up with simulated waveform. click on **'Run all'** icon on the tool box to display the waveform.



9 Testing the Design

What you see below is the simulation waveform obtained in ModelSim. Consider the Verilog design, from the waveform, we can see that when ' $s1 = 0$ ' and ' $s2 = 0$ ' the output ' $X1 = 0$ '. When ' $s1 = 0$ ' and ' $s2 = 1$ ' the output ' $X1 = 1$ '. When ' $s1 = 1$ ' and ' $s2 = 0$ ' the output ' $X1 = 1$ '. When ' $s1 = 1$ ' and ' $s2 = 1$ ' the output ' $X1 = 0$ '. This is the logic of a XOR Gate. The same can be observed in the VHDL design. Hence the design has been verified

9.1 Simulation waveform of the Verilog Design



9.2 Simulation waveform of the VHDL Design

