Icarus Verilog code implementation for Trustworthy for Event Detection (Fire detection and Smart Irrigation System).

This document is based upon free Icarus Verilog compiler, that works very well for windows as well as Linux. We use Intel Core i7, 16 GB RAM, Win 10, 64 Bit OS. We utilised the latest version of the Icarus Verilog (hardware description language version 10.3) for PLD programming, and GTKWave (version 3.3.104). This is a very small footprint software, especially if we wish to eventually port the code in a real FPGA and see the things working in real - and not just in simulator.

In order to run our project code, first we need to know about the Icarus Verilog fundamental.

Introduction

Hardware Description Language (HDL) is used to model digital circuits using codes. Verilog is one such code (VHDL is another type). Icarus Verilog is a free compiler implementation for the IEEE-1364 Verilog hardware description language. Icarus is maintained by Stephen Williams and it is released under the GNU GPL license.

What Is Icarus Verilog?

Icarus Verilog is a Verilog simulation and synthesis tool. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format. For batch simulation, the compiler can generate an intermediate form called vvp assembly. This intermediate form is executed by the ``vvp'' command. For synthesis, the compiler generates netlists in the desired format. The compiler proper is intended to parse and elaborate design descriptions written to the IEEE standard IEEE Std 1364-2005. This is a fairly large and complex standard, so it will take some time to fill all the dark alleys of the standard, but that's the goal.

A Test Suite?

There is also a test suite available. The test suite is also accessible as the ivtest github.com project, available here:

<https://github.com/steveicarus/ivtest>.

Icarus Verilog for Windows

You can find Icarus Verilog sources and binaries for most platforms at the Icarus site FTP. The sources available here:

http://bleyer.org/icarus/

How to run your first Verilog code/program?

Below is a complete tutorial:

Link: http://www.referencedesigner.com/tutorials/verilog/verilog_01.php

The C:/iverilog/bin subdirectory contains the executable file verilog.exe that is used to run simulator.

Now create a new file called hello.v in the directory C:/iverilog/bin and edit it with notepad or any other text editor(I prefer free notepad++). Enter the following lines of code in hello.v

```
1. module main;
2.   initial
3.     begin
4.       $display("Learning Verilog is easy with referencedesigner.com
   tutorial");
5.       $finish ;
6.     end
7. endmodule
```

- Go to your DOS prompt (Start - > cmd) and navigate to the directory C:/iverilog/bin

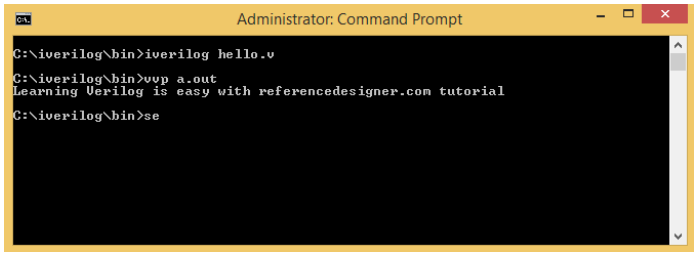
```
C:\> cd \iverilog\bin
```

Run iverilog using the command

```
C:\>iverilog\bin > iverilog hello.v
```

If you have done everything right, it should generate a file called a.out which you can run using command

```
C:\>iverilog\bin > vvp a.out
```



Every Verilog implementation goes through extensive verification. At this point the code just gives out the output saying "Learning Verilog is easy with referencedesigner.com tutorial". As shown in the above picture. It just verifies that your set up is ready for running Verilog code. Now we have understood the concept we will compile the code and run it in Icarus. In the next page we will write an actual real-life Verilog code two different IoT scenarios, i.e., for fire detection and smart irrigation system. Copy the following source codes in the directory C:\iverilog\bin.

****

Our Project Code and modules:

myproject.v

```
// Example of comparator using UDP Table; For event detection (Fire).
Case 5: Figure 9 (b) GTKWave waveform for Trustworthy Event Detection using
PLD/User Defined Primitive (UDP) (Logic Object Domain - I, Logic Object
Domain - II, and Logic Object Domain- III) for Smart Irrigation System.
//Here we are using three different variables for three different sensors
and two outputs one for the actual output for an event and one for the flag
(i.e., Gray Code). We use Intel Core i7, 16 GB RAM, Win 10, 64 Bit //OS. We
utilised the latest version of the Icarus Verilog iverilog-0.9.7 // (latest
stable release) for PLD programming, and GTKWave (version 3.3.104).
```

```
1. `timescale 1ns / 1ps
2. module comparator(
3.       input Temperature,
4.       input Smoke,
5.       input Humidity,
6.       output Output,
7.       output [2:0] Flag
8. );
9. compare c0(Output, Temperature, Smoke, Humidity);
10.      compare2 c20(Flag, Temperature, Smoke, Humidity);
11.      endmodule
12.      ///The below logic table is used for the logic gate design
13.      primitive compare(Output, in1, in2, in3);
14.            output Output;
15.            input in1,in2,in3;
16.      table
17.      // in1  in2  in3 : out
18.          0    0    0  :  0;
19.          0    0    1  :  0;
20.          0    1    0  :  0;
```

```
21.          0    1    1  :  1;
22.          1    0    0  :  0;
23.          1    0    1  :  1;
24.          1    1    0  :  1;
25.          1    1    1  :  1;
26.
27.      endtable
28.      endprimitive
29.      // The below logic table is used for the Flag (i.e., Gray Code), it
   //also transfer the status of all sensors such as which sensor or variables
   //participate in the output
30.      primitive compare2(Flag, in1, in2, in3);
31.            output Flag;
32.            input in1,in2,in3;
33.      table
34.      // in1  in2  in3 : out2
35.          0    0    0  :  0;
36.          0    0    1  :  1;
37.          0    1    0  :  1;
38.          0    1    1  :  1;
39.          1    0    0  :  1;
40.          1    0    1  :  1;
41.          1    1    0  :  1;
42.          1    1    1  :  1;
43.
44.      endtable
45.      endprimitive
```

CaseOne_tb.v

Note: CaseOne_tb.v is a testbench for myproject.v module; we have 5 different Test Cases for fire detections and smart irrigation such as shown in our main manuscript Figure 9 GTKWave waveform for Trustworthy Event Detection using PLD/User Defined Primitive (UDP) (Logic Object Domain - I, Logic Object Domain - II, and Logic Object Domain- III) for fire detection and smart irrigation system.

Case 1: Figure 9 (a) GTKWave waveform for Trustworthy Event Detection using PLD/User Defined Primitive (UDP) (Logic Object Domain - I, Logic Object Domain - II, and Logic Object Domain- III) for fire detection.

```
//CaseOne TestBench for myproject.v module
The simulation runs for 250 seconds According to the logical object module
for fre detection, four of the eight output values are sent towards the
gateway for the reliable event detection, denoted in the logic table by the
'1' output column. The reliable event information will be sent towards the
```

upstream node, if and only if, at least two sensors or all sensors sense an event simultaneously, i.e., temperature, smoke, and humidity sensor exceed the threshold. For trustworthy fire detection, we can observe two of the three reliable events are detected at 60 and 180 seconds, respectively, over 250 seconds of simulation time.

```
// testbench for myproject.v module
`timescale 1s / 1ps
module stimulus;
	// Inputs
	reg Temperature;
	reg Smoke;
	reg Humidity;
	// Outputs
	wire Output;
		wire [2:0] Flag;
		// Instantiate the Unit Under Test (UUT)
		comparator uut (
			.Temperature(Temperature),
			.Smoke(Smoke),
			.Humidity(Humidity),
			.Output(Output),
			.Flag(Flag)
		);
		initial begin
			// Initialize Inputs
		$dumpfile("test.vcd");
	    $dumpvars(0,stimulus);
		Temperature = 0;
		Smoke = 0;
		Humidity = 0;
	    // an event occures and all sensors observed the event
		#60 Temperature = 1;
		#0  Smoke = 1;
		#0  Humidity = 1;
		#20 Temperature = 0;
		#0  Smoke = 0;
		#0  Humidity = 0;
		////////////only one sensor due to change environment effects
		#40 Temperature = 1;
		#0  Smoke = 0;
		#0  Humidity = 0;
		#20 Temperature = 0;
		#0  Smoke = 0;
		#0  Humidity = 0;
		////////////only one sensor changes due to environment effects
		#10 Temperature = 0;
		#0   Smoke = 0;
		#0   Humidity = 1;
```

```
44.             #20 Temperature = 0;
45.             #0   Smoke = 0;
46.             #0   Humidity = 0;
47.             // an event occurs and all sensors observed the event
48.             #10 Temperature = 1;
49.             #0   Smoke = 1;
50.             #0   Humidity = 1;
51.             #20 Temperature = 0;
52.             #0   Smoke = 0;
53.             #0   Humidity = 0;
54.             #50;
55.           end
56.             initial begin
57.             $monitor("Time=%3d, Temperature=%d,  Smoke=%d,  Humidity=%d,
    Output=%d,  Flag (8421)=%b\n",$time, Temperature, Smoke, Humidity, Output,
    Flag);
58.             end
59.       Endmodule
60.       // C:\iverilog\bin> iverilog -o CaseOne myproject.v CaseOne_tb.v
61.       //C:\iverilog\bin>vvp CaseOne
62.       //C:\iverilog\bin>gtkwave test.vcd &
```

Now go to the dos windows (Start -> cmd) navigate to the iverilog\bin directory

C:\> cd iverilog\bin

Compile the program using

C:\iverilog\bin>iverilog -o CaseOne myproject.v CaseOne_tb.v

If everything goes right, it will not produce any output. If there are any syntax errors it will show some errors.

To see the output of the stimulus, you may like to give the following command

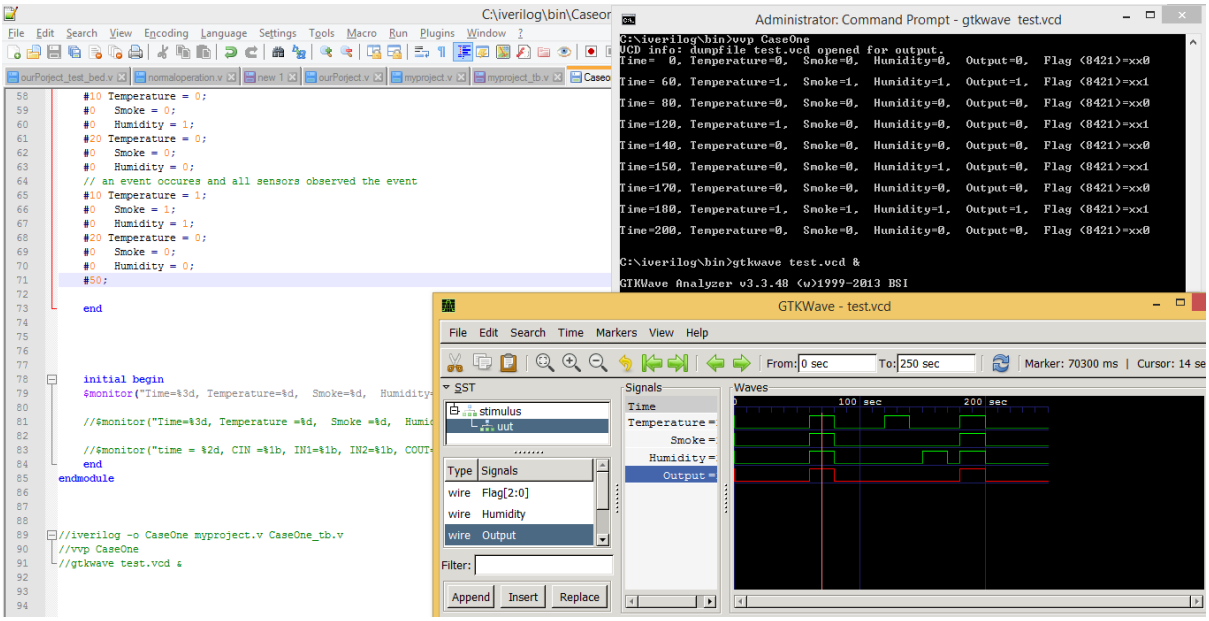
C:\iverilog\bin>vvp CaseOne

Note: If you want to see the output in the graphic form i.e., GTKWave then used below command

C:\iverilog\bin>gtkwave test.vcd &

This shows the following outputs.

Note: The above commands can be used for Case 2, Case 3, and Case 4 according to TestBench names.



CaseTwo_tb.v

Case 2: Figure 10. (a) GTKWave waveform for Trustworthy Event Detection using PLD/User Defined Primitive (UDP) (Logic Object Domain - I) with some noise/faulty signals.

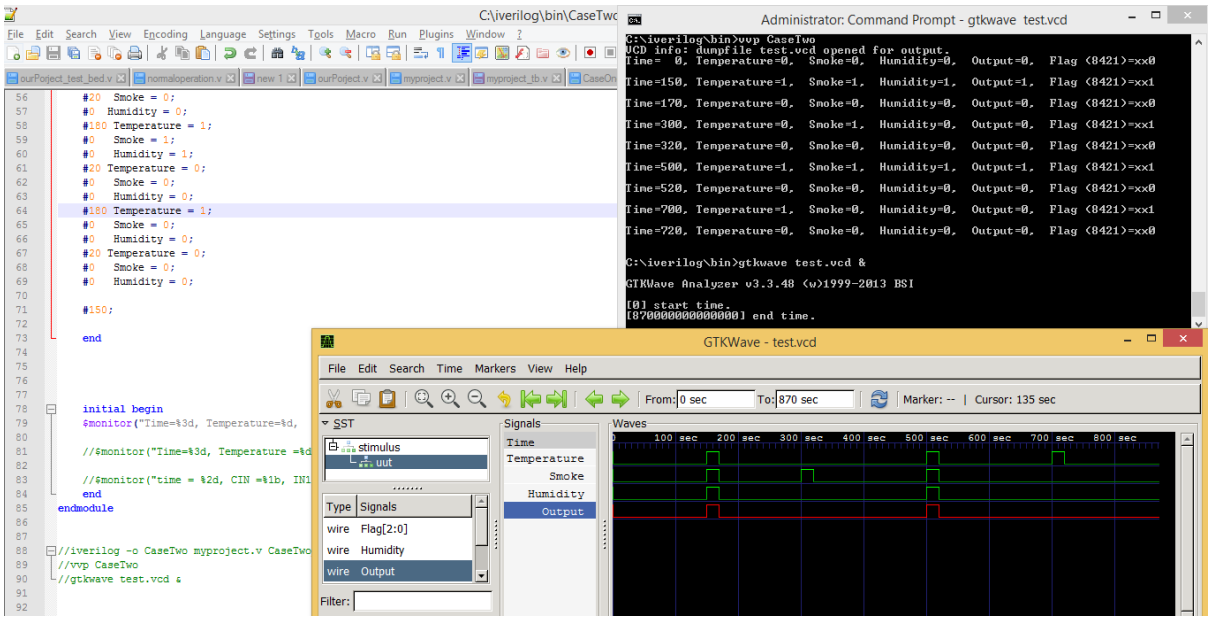
```
//The simulation runs for 870 seconds According to the logical object module for fire detection, four of the eight output values are sent towards the gateway for the reliable event detection, denoted in the logic table by the '1' output column. An output event is produced if and only if at least two of the three or all sensors observe the event simultaneously. For trustworthy fire detection, we can observe two out of four reliable events are detected at 150 and 500 seconds, respectively, over 870 seconds of simulation time. Furthermore, in this simulation the temperature and a smoke sensor having defective or somewhat compromise signals at 700 and 300 seconds, respectively.
```

```
1.
2. // testbench for myproject.v module
3. `timescale 1s / 1ps
4. module stimulus;
5.         // Inputs
6.         reg Temperature;
7.         reg Smoke;
8.         reg Humidity;
9.         // Outputs
10.                wire Output;
11.                wire [2:0] Flag;
12.                // Instantiate the Unit Under Test (UUT)
13.                comparator uut (
14.                        .Temperature(Temperature),
15.                        .Smoke(Smoke),
```

```
16.                         .Humidity(Humidity),
17.                         .Output(Output),
18.                         .Flag(Flag)
19.                 );
20.                 initial begin
21.                         // Initialize Inputs
22.                 $dumpfile("test.vcd");
23.             $dumpvars(0,stimulus);
24.                 Temperature = 0;
25.                 Smoke = 0;
26.                 Humidity = 0;
27.                 // an event occures and all sensors observed the event
28.                 #150 Temperature = 1;
29.                 #0  Smoke = 1;
30.                 #0  Humidity = 1;
31.                 #20 Temperature = 0;
32.                 #0  Smoke = 0;
33.                 #0  Humidity = 0;
34.                 ////////////only one sensor change due to enviroment effects
35.                 #0 Temperature = 0;
36.                 #130  Smoke = 1;
37.                 #0  Humidity = 0;
38.                 #0 Temperature = 0;
39.                 #20  Smoke = 0;
40.                 #0  Humidity = 0;
41.             // an event occures and all sensors observed the event
42.                 #180 Temperature = 1;
43.                 #0   Smoke = 1;
44.                 #0   Humidity = 1;
45.                 #20 Temperature = 0;
46.                 #0   Smoke = 0;
47.                 #0   Humidity = 0;
48.                 ////////////only one sensor change due to enviroment effects
49.                 #180 Temperature = 1;
50.                 #0   Smoke = 0;
51.                 #0   Humidity = 0;
52.                 #20 Temperature = 0;
53.                 #0   Smoke = 0;
54.                 #0   Humidity = 0;
55.                 #150;
56.             end
57.
58.                 initial begin
59.                 $monitor("Time=%3d, Temperature=%d,  Smoke=%d,  Humidity=%d,
    Output=%d,  Flag (8421)=%b\n",$time, Temperature, Smoke, Humidity, Output,
    Flag);
60.         Endmodule
61.
```

```
// C:\iverilog\bin> iverilog -o CaseTwo myproject.v CaseTwo_tb.v
//C:\iverilog\bin>vvp CaseTwo
//C:\iverilog\bin>gtkwave test.vcd &
```

Output:



CaseThree_tb.v

Case 3: Figure 10. (b) GTKWave waveform for Trustworthy Event Detection using PLD/User Defned Primitive (UDP) (Logic Object Domain - II) without noise/faulty signals.

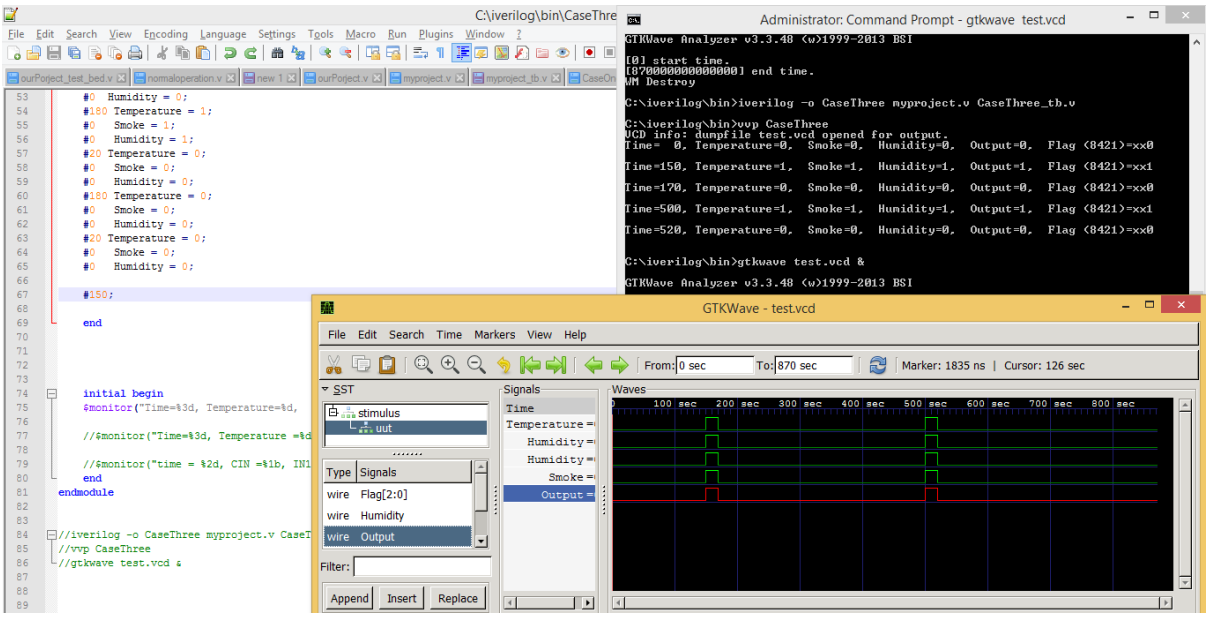
```
// CaseThree_tb.v Case 3: Figure 10. (b)
For trustworthy fire detection, we can observetwo out of four reliable events are detected at 150 and 500 seconds, respectively, over 870 seconds of simulation time. Furthermore, in this simulation we have no defective or somewhat compromise signals, it means that the obtained event is trustworthy, and the corresponding output signals are reliable.
```

```
// testbench for myproject.v module
`timescale 1s / 1ps
module stimulus;
        // Inputs
        reg Temperature;
        reg Smoke;
        reg Humidity;
        // Outputs
        wire Output;
                wire [2:0] Flag;
                // Instantiate the Unit Under Test (UUT)
                comparator uut (
                        .Temperature(Temperature),
```

```
14.                         .Smoke(Smoke),
15.                         .Humidity(Humidity),
16.                         .Output(Output),
17.                         .Flag(Flag)
18.                 );
19.                 initial begin
20.                         // Initialize Inputs
21.                 $dumpfile("test.vcd");
22.             $dumpvars(0,stimulus);
23.                 Temperature = 0;
24.                 Smoke = 0;
25.                 Humidity = 0;
26.                 //an event occures and all sensors observed the event
27.                 #150 Temperature = 1;
28.                 #0  Smoke = 1;
29.                 #0  Humidity = 1;
30.                 #20 Temperature = 0;
31.                 #0  Smoke = 0;
32.                 #0  Humidity = 0;
33.                 #0 Temperature = 0;
34.                 #130  Smoke = 0;
35.                 #0  Humidity = 0;
36.                 #0 Temperature = 0;
37.                 #20  Smoke = 0;
38.                 #0  Humidity = 0;
39.                 //an event occures and all sensors observed the event
40.                 #180 Temperature = 1;
41.                 #0   Smoke = 1;
42.                 #0   Humidity = 1;
43.                 #20 Temperature = 0;
44.                 #0   Smoke = 0;
45.                 #0   Humidity = 0;
46.                 #180 Temperature = 0;
47.                 #0   Smoke = 0;
48.                 #0   Humidity = 0;
49.                 #20 Temperature = 0;
50.                 #0   Smoke = 0;
51.                 #0   Humidity = 0;
52.
53.                 #150;
54.
55.             end
56.                 initial begin
57.                 $monitor("Time=%3d, Temperature=%d,  Smoke=%d,  Humidity=%d,
    Output=%d,  Flag (8421)=%b\n",$time, Temperature, Smoke, Humidity, Output,
    Flag);
58.                 end
59.         endmodule
```

```
60.
61.
62.         // C:\iverilog\bin> iverilog -o CaseThree myproject.v CaseThree_tb.v
63.         // C:\iverilog\bin> vvp CaseThree
64.         // C:\iverilog\bin> gtkwave test.vcd &
```

Output:



CaseFour_tb.v

Case 4: Figure 10. (c) GTKWave waveform for Trustworthy Event Detection using PLD/User Defned Primitive (UDP) (Logic Object Domain - III) with some noise/faulty signals.

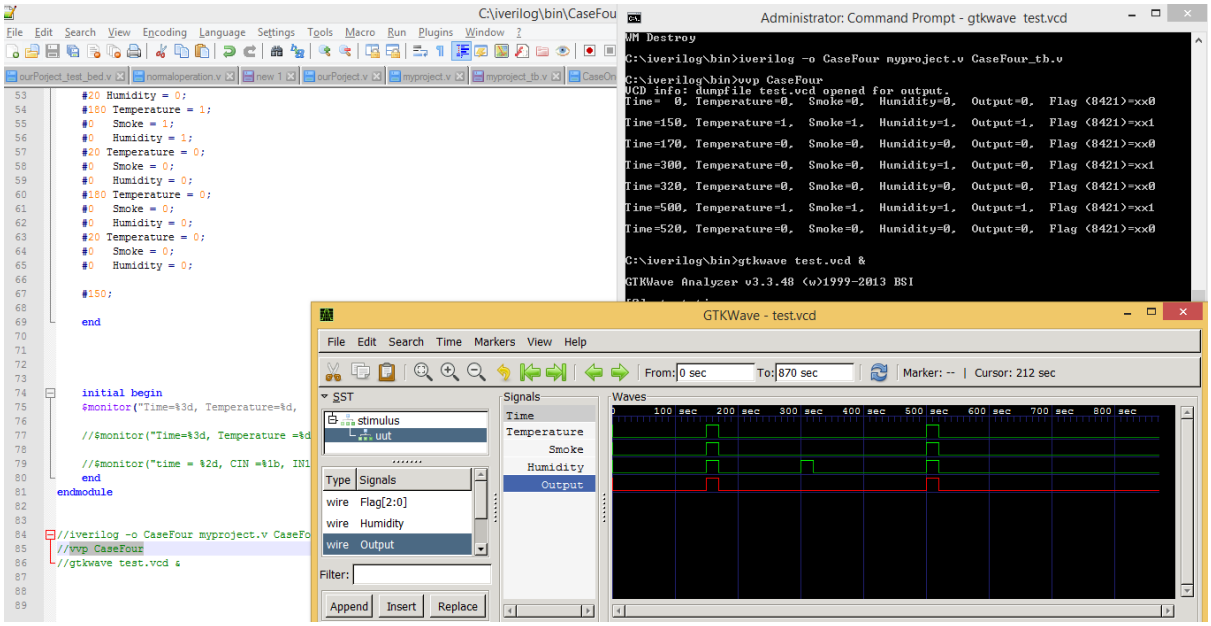
```
// CaseFour_tb.v; For trustworthy fre detection, we can observe
// two out of four reliable events are detected at 150 and 500 seconds,
respectively, over 870 seconds of simulation time. In this simulation, only
one faulty signal is fed into the humidity sensor, and the associated event
information is captured.
```

```
1. // testbench for myproject.v module
2. `timescale 1s / 1ps
3. module stimulus;
4.         // Inputs
5.         reg Temperature;
6.         reg Smoke;
7.         reg Humidity;
8.         // Outputs
9.         wire Output;
10.                wire [2:0] Flag;
```

```
11.                 // Instantiate the Unit Under Test (UUT)
12.                 comparator uut (
13.                         .Temperature(Temperature),
14.                         .Smoke(Smoke),
15.                         .Humidity(Humidity),
16.                         .Output(Output),
17.                         .Flag(Flag)
18.                 );
19.
20.                 initial begin
21.                         // Initialize Inputs
22.                 $dumpfile("test.vcd");
23.             $dumpvars(0,stimulus);
24.                 Temperature = 0;
25.                 Smoke = 0;
26.                 Humidity = 0;
27.                 //an event occures and all sensors observed the event
28.                 #150 Temperature = 1;
29.                 #0  Smoke = 1;
30.                 #0  Humidity = 1;
31.                 #20 Temperature = 0;
32.                 #0  Smoke = 0;
33.                 #0  Humidity = 0;
34.                 ////////////only one sensor change due to enviroment effects
35.                 #0 Temperature = 0;
36.                 #0  Smoke = 0;
37.                 #130  Humidity = 1;
38.                 #0 Temperature = 0;
39.                 #0  Smoke = 0;
40.                 #20 Humidity = 0;
41.                 //an event occures and all sensors observed the event
42.                 #180 Temperature = 1;
43.                 #0   Smoke = 1;
44.                 #0   Humidity = 1;
45.                 #20 Temperature = 0;
46.                 #0   Smoke = 0;
47.                 #0   Humidity = 0;
48.                 #180 Temperature = 0;
49.                 #0   Smoke = 0;
50.                 #0   Humidity = 0;
51.                 #20 Temperature = 0;
52.                 #0   Smoke = 0;
53.                 #0   Humidity = 0;
54.
55.                 #150;
56.
57.             end
58.
```

```
59.                 initial begin
60.                 $monitor("Time=%3d, Temperature=%d,  Smoke=%d,  Humidity=%d,
   Output=%d,  Flag (8421)=%b\n",$time, Temperature, Smoke, Humidity, Output,
   Flag);
61.
62.                 end
63.         endmodule
64.
65.         // C:\iverilog\bin> iverilog -o CaseFour myproject.v CaseFour_tb.v
66.         // C:\iverilog\bin> vvp CaseFour
67.         // C:\iverilog\bin> gtkwave test.vcd &
```

Output:



****

OurProjectSAS.V

Case 5: Figure 9 (b) GTKWave waveform for Trustworthy Event Detection using PLD/User Defined Primitive (UDP) (Logic Object Domain - I, Logic Object Domain - II, and Logic Object Domain- III) for Smart Irrigation System.

```
// OurProjectSAS.V
//Example of comparator using UDP Table for smart irrigation system
(trustworthy event detevtion)
Case 5: Figure 9 (b) GTKWave waveform for Trustworthy Event Detection using
PLD/User Defined Primitive (UDP) (Logic Object Domain - I, Logic Object
Domain - II, and Logic Object Domain- III) for Smart Irrigation System.
1.
```

```
2. `timescale 1ns / 1ps
3. module comparator(
4.         input AirTemperature,
5.         input SoilTemprature,
6.         input AirHumidity,
7.         input SoilMoisture,
8.         output Output,
9.         output [2:0] Flag
10.
11.        );
12.        compare  c0(Output,  AirTemperature,  SoilTemprature,  AirHumidity,
   SoilMoisture);
13.
14.        compare2  c20(Flag,  AirTemperature,  SoilTemprature,  AirHumidity,
   SoilMoisture);
15.
16.        endmodule
17.
18.        ///The below logic table is used for the logic gate design
19.
20.        primitive compare(Output, in1, in2, in3, in4);
21.                output Output;
22.                input in1,in2,in3, in4;
23.        table
24.        // in1  in2  in3  in4: out
25.            0    0    0    0    :  0;
26.            0    0    0    1    :  0;
27.            0    0    1    0    :  0;
28.            0    0    1    1    :  1;
29.            0    1    0    0    :  0;
30.            0    1    0    1    :  1;
31.            0    1    1    0    :  1;
32.            0    1    1    1    :  1;
33.            1    0    0    0    :  0;
34.            1    0    0    1    :  1;
35.            1    0    1    0    :  1;
36.            1    0    1    1    :  1;
37.            1    1    0    0    :  1;
38.            1    1    0    1    :  1;
39.            1    1    1    0    :  1;
40.            1    1    1    1    :  1;
41.
42.        endtable
43.        endprimitive
44.
45.        /// The below logic table is used for the Flag (i.e, Gray Code), it
   also transfers the status of all sensors such as which sensor or variables
   participate in the output
```

```
46.
47.         primitive compare2(Flag, in1, in2, in3, in4);
48.                 output Flag;
49.                 input in1,in2,in3, in4;
50.         table
51.         // in1   in2   in3   in4: out
52.             0     0     0     0     :  0;
53.             0     0     0     1     :  1;
54.             0     0     1     0     :  1;
55.             0     0     1     1     :  1;
56.             0     1     0     0     :  1;
57.             0     1     0     1     :  1;
58.             0     1     1     0     :  1;
59.             0     1     1     1     :  1;
60.             1     0     0     0     :  1;
61.             1     0     0     1     :  1;
62.             1     0     1     0     :  1;
63.             1     0     1     1     :  1;
64.             1     1     0     0     :  1;
65.             1     1     0     1     :  1;
66.             1     1     1     0     :  1;
67.             1     1     1     1     :  1;
68.
69.         endtable
70.         endprimitive
```

CaseFive_tb.v

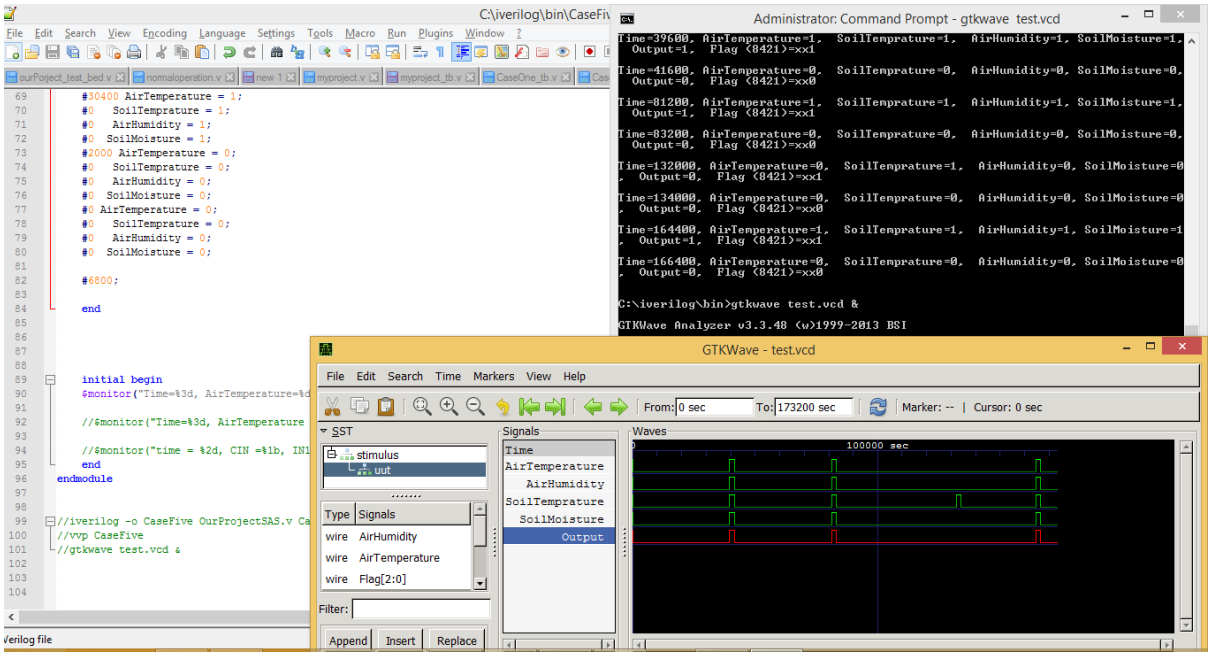
```
//Case 5: TestBench for OurProjectSAS.v module
For trustworthy event detection (smart irrigation), we can observe three out of four reliable events are detected at 39600, 81200, and 16400 seconds, respectively, over 173200 seconds (48.1 Hours) of simulation time.
In this simulation, only one faulty signal is fed into the Soil Temperature sensor, and the associated event information is captured too at 132000 Seconds.
```

```
1.
2. // testbench for comparator module
3. `timescale 1s / 1ps
4. module stimulus;
5.         // Inputs
6.         reg AirTemperature;
7.         reg SoilTemprature;
8.         reg AirHumidity;
9.         reg SoilMoisture;
10.                // Outputs
11.                wire Output;
12.                wire [2:0] Flag;
```

```
13.
14.                 // Instantiate the Unit Under Test (UUT)
15.                 comparator uut (
16.                         .AirTemperature(AirTemperature),
17.                         .SoilTemprature(SoilTemprature),
18.                         .AirHumidity(AirHumidity),
19.                         .SoilMoisture(SoilMoisture),
20.                         .Output(Output),
21.                         .Flag(Flag)
22.                 );
23.
24.                 initial begin
25.                         // Initialize Inputs
26.                 $dumpfile("test.vcd");
27.             $dumpvars(0,stimulus);
28.
29.                 AirTemperature = 0;
30.                 SoilTemprature = 0;
31.                 AirHumidity = 0;
32.                 SoilMoisture = 0;
33.                 // an event occures and all sensors observed the event
34.                 #39600 AirTemperature = 1;
35.                 #0  SoilTemprature = 1;
36.                 #0  AirHumidity = 1;
37.                 #0  SoilMoisture = 1;
38.                 #2000 AirTemperature = 0;
39.                 #0  SoilTemprature = 0;
40.                 #0  AirHumidity = 0;
41.                 #0  SoilMoisture = 0;
42.                 // an event occures and all sensors observed the event
43.                 #39600 AirTemperature = 1;
44.                 #0  SoilTemprature = 1;
45.                 #0  AirHumidity = 1;
46.                 #0  SoilMoisture = 1;
47.                 #2000 AirTemperature = 0;
48.                 #0  SoilTemprature = 0;
49.                 #0  SoilMoisture = 0;
50.                 #0 AirHumidity = 0;
51.                 ////////////only one sensor change due to enviroment effects
52.                 #0 AirTemperature = 0;
53.                 #48800  SoilTemprature = 1;
54.                 #0  SoilMoisture = 0;
55.                 #0 AirHumidity = 0;
56.                 #0 AirTemperature = 0;
57.                 #2000  SoilTemprature = 0;
58.                 #0  SoilMoisture = 0;
59.                 #0 AirHumidity = 0;
60.                 // an event occures and all sensors observed the event
```

```
61.                 #30400 AirTemperature = 1;
62.                 #0   SoilTemprature = 1;
63.                 #0   AirHumidity = 1;
64.                 #0  SoilMoisture = 1;
65.                 #2000 AirTemperature = 0;
66.                 #0   SoilTemprature = 0;
67.                 #0   AirHumidity = 0;
68.                 #0  SoilMoisture = 0;
69.                 #0 AirTemperature = 0;
70.                 #0   SoilTemprature = 0;
71.                 #0   AirHumidity = 0;
72.                 #0  SoilMoisture = 0;
73.
74.                 #6800;
75.
76.             end
77.
78.
79.                 initial begin
80.                 $monitor("Time=%3d,  AirTemperature=%d,    SoilTemprature=%d,
   AirHumidity=%d,  SoilMoisture=%d,    Output=%d,    Flag  (8421)=%b\n",$time,
   AirTemperature, SoilTemprature, AirHumidity,  SoilMoisture, Output, Flag);
81.
82.                 end
83.         endmodule
84.
85.
86.         // C:\iverilog\bin> iverilog -o CaseFive OurProjectSAS.v CaseFive_tb.v
87.         // C:\iverilog\bin> vvp CaseFive
88.         // C:\iverilog\bin> gtkwave test.vcd &
```

Output:



****

If you need more information or help you can contact me.

Thanks

Regards:
Hafiz ur Rahman
www.imhafiz.com