# How to Create Customized Code (Full Adder)
# SLG47910

This application shows how to combine multiple custom modules and implement it using the SLG47910 FPGA. This application is demonstrated with the help of a Full Adder example. The functionality of the full adder is verified by creating a Testbench and observing the output waveforms on the inbuilt GTKWave Software.

## Contents



## Terms and Definitions

| | |
|---|---|
| CLB | Configuration Logic Block |
| HDL Editor | Workspace where Verilog code is entered |
| FPGA | Field Programmable Gate Array |
| FPGA Editor | Main FPGA design and simulation window |
| Go Configure Software Hub | Main window for device selection |
| ForgeFPGA Window | Main FPGA project window for debug and IO programming |

## References

For related documents and software, please visit:

ForgeFPGA Low-density FPGAs | Renesas

Download our free ForgeFPGA™ Designer software [1] to open the .ffpga design files [1] and view the proposed circuit design.

[1] ForgeFPGA Designer Software, Software Download and User Guide, Renesas Electronics
[2] AN-FG-002 How to Create Customized Code (Full_Adder) ForgeFPGA Design File, Renesas Electronics
[3] SLG47910, Preliminary Datasheet, Renesas Electronics

# 1. Introduction

A Full Adder module is created with the help of 2 Half Adder modules and an OR Gate module mapped together.

Structural modelling can be implemented in ForgeFPGA Workshop by defining the topmost module with the keywork (*top*). This ensures that the software knows the difference between the topmost module and the modules mapped to it using port mapping syntax. The functionality of the Full Adder can be verified by creating a Test Bench and providing simulation within it. The simulation can be visualized using GTKWave Software within the ForgeFPGA software. This Application is designed to understand the different aspects of the software without the need of programming the part to check functionality.

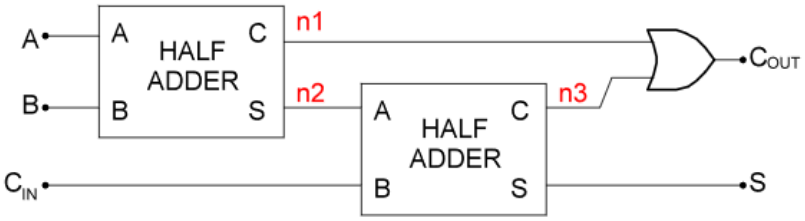



**Figure 1: System Diagram**

# 2. Ingredients

Latest Revision of ForgeFPGA Workshop software

# 3. Verilog Code

```
//OR gate module for Full Adder
module OR_Gate(
    input a,b,
    output c );

assign c = a | b; //OR gate operation

endmodule
```

```
//Half Adder Module for Full Adder
module Half_Adder(input x,y,
    output sum,carry );
 assign sum = x ^ y;
 assign carry = x & y;
endmodule
```

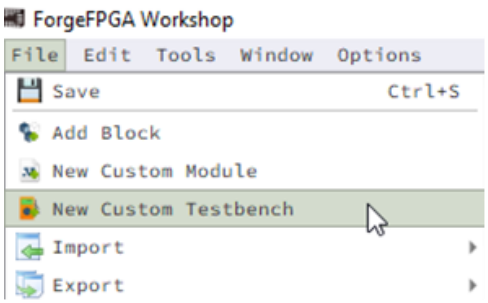
```
//Full Adder Module using Half Adder module& OR Gate Module
(*top*)module Full_Adder(
input A,
input B,
input Cin,
output S,
output Cout
);

wire n1,n2,n3;

//port mapping
Half_Adder   Half_Adder_1 (.x(A), .y(B), .sum(n2) , .carry(n1));;
Half_Adder   Half_Adder_2 (.sum(S), .carry(n3), .x(n2) , .y(Cin));
OR_Gate      OR_Gate_1(.a(n1), .b(n3) ,.c(Cout));
endmodule
```

**Figure 2: Verilog Code**

# 4. Test Bench

The Test Bench is used to provide Stimulus to verify the correctness or working of our DUT, hence we must instantiate our design module . The module name must be given as <module name_tb> to be recognized by the ForgeFPGA Studio as a testbench file (see Figure 3).

The functionality can be verified using the waveforms produced by GTKWave software inbuilt.



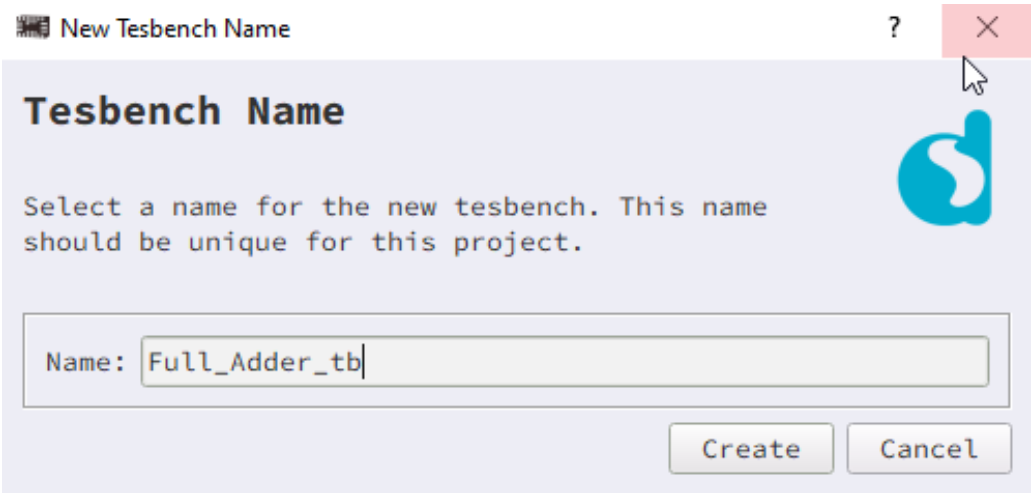



**Figure 3: Steps to Launch Custom Testbench**

```
`timescale 1ns / 1ps
module Full_Adder_tb;   // module declaration
    reg A;  // used to apply a stimulus to the inputs of DUT
    reg  B;
    reg Cin;
    wire  S;    // used to check the output signals from DUT
    wire Cout;
    integer i ;  //for loop

// instantiate the design DUT  and connect to testbench variables
 Full_Adder  fa0( .A(A),  .B(B) , .Cin(Cin), .Cout(Cout), .S(S));

  initial begin
A<=0;   B<=0; Cin<=0;    // starts execution at t=0

    $dumpfile ("Full_Adder_tb.vcd");
    $dumpvars (0, Full_Adder_tb);

$monitor ("A=%b\t B=%b \t Cin=%b\t S=%b\t Cout=%b", A, B, Cin, S, Cout);
// to display values in Logger everytime the input changes

//use for loop to go through all the inputs
for ( i=0; i<5; i=i+1)   begin
 #10 A<= $random ;    //delay by 10ns
     B<= $random;
     Cin<=$random ;
      end
  end
endmodule
```

**Figure 4: Verilog Code**

# 5. Design Steps

1. Open the ForgeFPGA Workshop software in GreenPAK and select the SLG47910 device. From the ForgeFPGA tool bar, select the FPGA Editor Tab (see Figure 5).

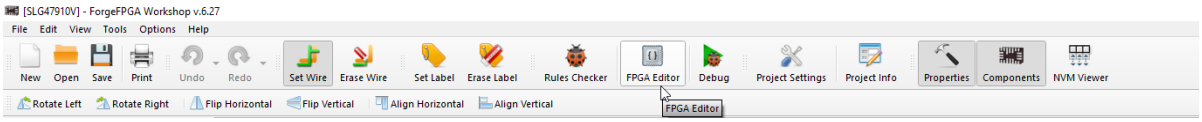



**Figure 5: ForgeFPGA Tool Bar**

3. Create 3 different Custom Modules called OR_Gate , Half_Adder & Full_Adder and enter the respective Verilog Codes in the HDL Editor for each module.

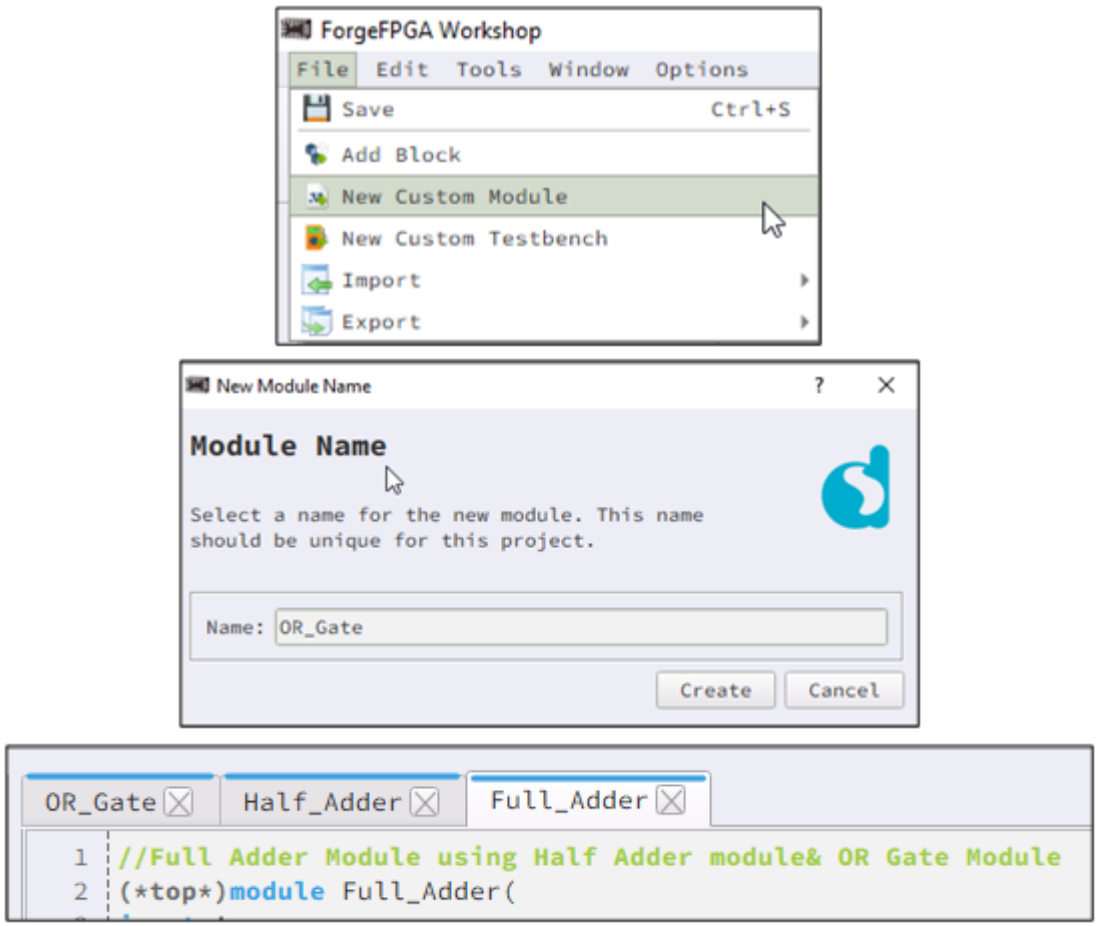



**Figure 6: Steps to Launch Custom Module**

4. Define the Full_Adder module as (*top*) so that it gets recognized as the topmost module and using port mapping, map the OR_Gate & Half_Adder module to it and save the code using the save button in the top left corner of the FPGA Editor

5. Check for any Syntax Errors using the Synthesize Button on the lower left side of the FPGA Editor & Check logger for any issues /warnings

6. Click on Generate Bitstream button. This generates bitstream file and other build files such as Floorplan Mapping, Routing information, Netlist, Timing Analysis, Resource Utilization report used in this design. All these information can be accessed by the different tabs found at the top of the ForgeFPGA Workshop software. Check the logger and issues tab on the bottom to make sure that the bit stream was generated correctly .

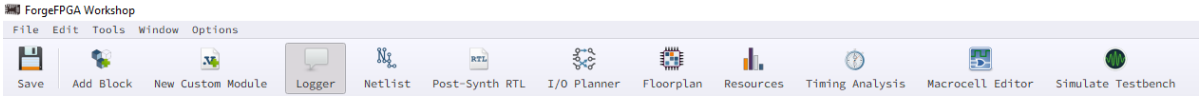



**Figure 7: ForgeFPGA Tool Bar**

7. Create Test Bench module and define the different inputs you want to run the code through to verify functionality at different time intervals. Simulate the testbench through the Simulate TestBench button on the toolbar.

If there are no syntax errors on the TestBench written, then the GTKWave software should pop-up automatically provided that the naming of the module is correct.

8. Load the signals in the GTKWave software the needs to be observed. After this step, you should observe green/ red lines right side of the screen. Click ‘Zoom Fit’ to fir the waveforms in the selected size of the window and then click ‘Reload’ (see Figure 8).

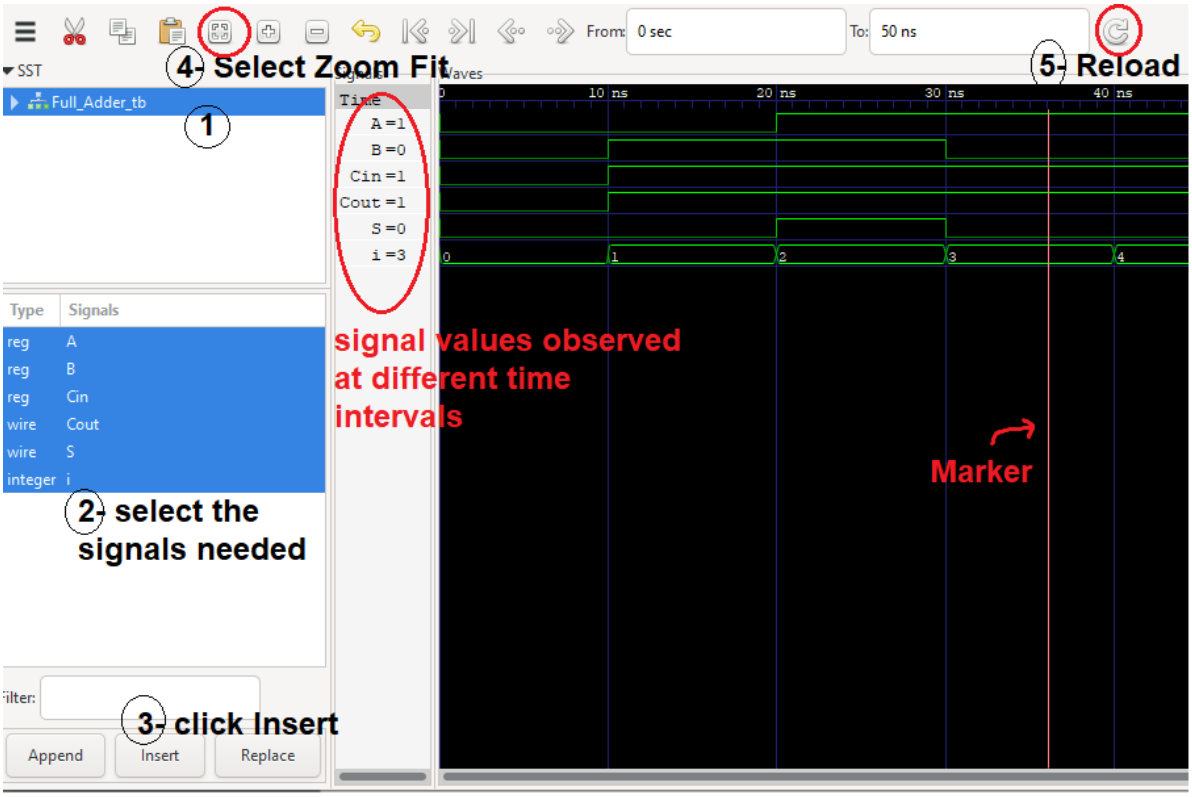



**Figure 8: GTKWave Simulation Waveform**

9. The corresponding waveforms can be observed. Each signal value matches the value assigned in the Test Bench code. With the help of the Marker, the values of the signals can be observed at different times.

# 6. Conclusion

The procedure outlines in this application example can be applied to any circuit by creating the custom modules on the similar lines and then port mapping them to the top module to make it a whole circuit. The usage of defining a module as "top" module has been demonstrate and the functionality has been verified using the inbuild GTKWave software and testbench. This testcase is available for download. If interested, please contact the ForgeFPFA Business Support Team

# 7. Revision History

| Revision | Date | Description |
|---|---|---|
| 1.00 | Mar 3, 2022 | Initial release. |

## IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use o any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.0 Mar 2020)


### Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

### Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

www.renesas.com/contact/


### Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.