

4-bit Counter

SLG47910

Abstract

This application shows how to build a 4-bit counter and observe the simulation waveforms on GTKWave and explains the Emulation feature on the ForgeFPGA Software. The internal oscillator is used as clock for the counter. The counter outputs are programmed to GPIOs.

This application note comes complete with design files which can be found in the References Section.

Contents

1. Terms and Definitions	1
2. References	2
3. Introduction	2
3.1 Debug	2
3.2 Type of hardware platforms	2
3.3 Platform Configuration Guide	3
3.4 Add Test Point Controls	4
3.4.1. Types of areas	5
4. Logic Analyzer	7
5. Ingredients	8
6. Verilog Code	8
7. Testbench	8
8. Floorplan: CLB Utilization	9
9. Design Steps	10
10. Conclusion	12
11. Revision History	13

1. Terms and Definitions

FPGA	Field Programmable Gate Array
FPGA Editor	Main FPGA design and simulation window
Go Configure Software Hub	Main window for device selection
ForgeFPGA workshop	Main FPGA project window for debug and IO programming
CLB	Configuration Logic Block
TP	Test Point

2. References

Download our free ForgeFPGA™ Designer software [1] to open the .ffpga design files [2] and view the proposed circuit design.

- [1] ForgeFPGA Designer Software, Software Download and User Guide
- [2] [AN-FG-001 4-Bit Counter.ffpga](#), ForgeFPGA Design File
- [3] SLG47910, Datasheet, Renesas Electronics
- [4] SLG47910 Software User Guide, Renesas Electronics

3. Introduction

This application shows how to build a 4-bit Counter using SLG47910 and how to simulate the testbench to observe the waveform in GTKWave. Emulation function allows the user to see how the part will behave after it has been programmed. Since the SLG47910 has One-Time-Program capability (OTP), it is important to verify the functionality of the design you designed by emulating it.

3.1 Debug

The Debug button starts the Debug tool in ForgeFPGA Workshop Window. The Debug tool enables electronic circuit emulation and chip programming, which uses specific hardware platform to replicate the behavior of chip components designed. Before starting the emulation process, add test point (TP) controls to configure the emulation process. The Test Points controls allows the user to configure the GPIOs in different options (Section 3.4)

3.2 Type of hardware platforms

After the Debug button is pressed, the Development Platform Selector and the two hardware options will be displayed (Figure 1). Select the *ForgeFPGA Development Platform* option.

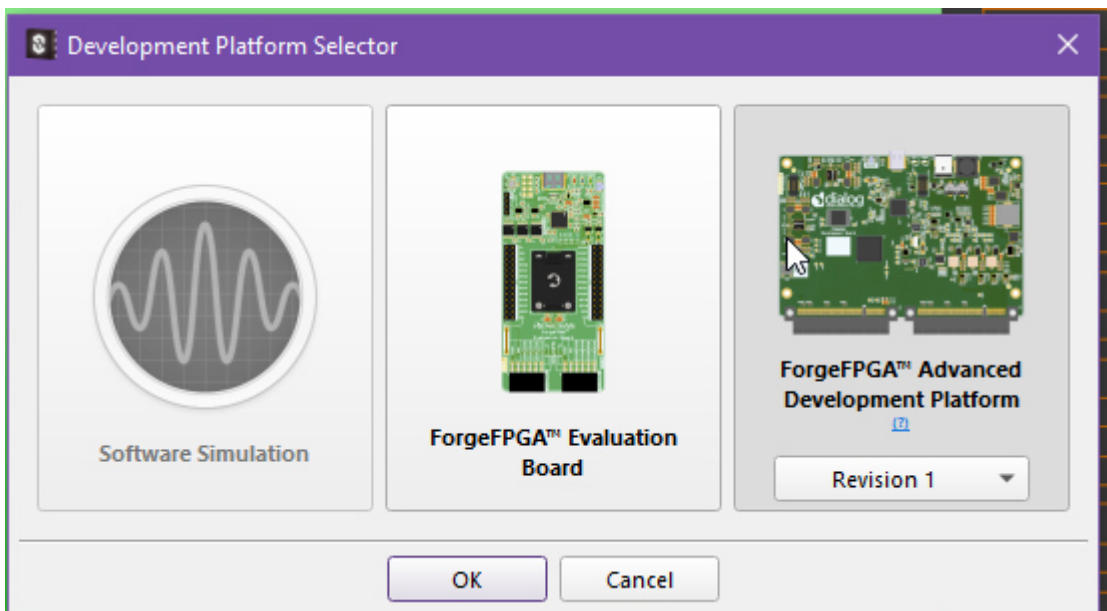


Figure 1: Development Platform Selector

3.3 Platform Configuration Guide

Recommended Platform Configuration contains information about suitable sockets, adapters, and boards for specific chip. The user can pop up the guide by clicking on platform's name into Debugging controls panel. (Figure 2)

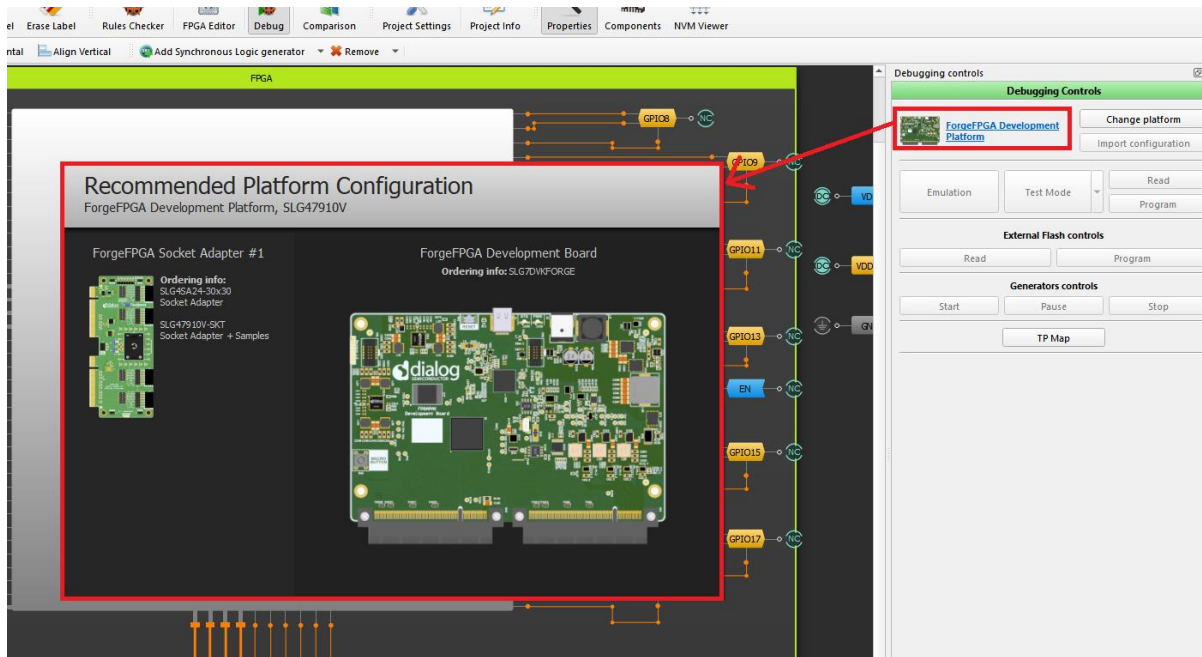


Figure 2: Platform Configuration Guide

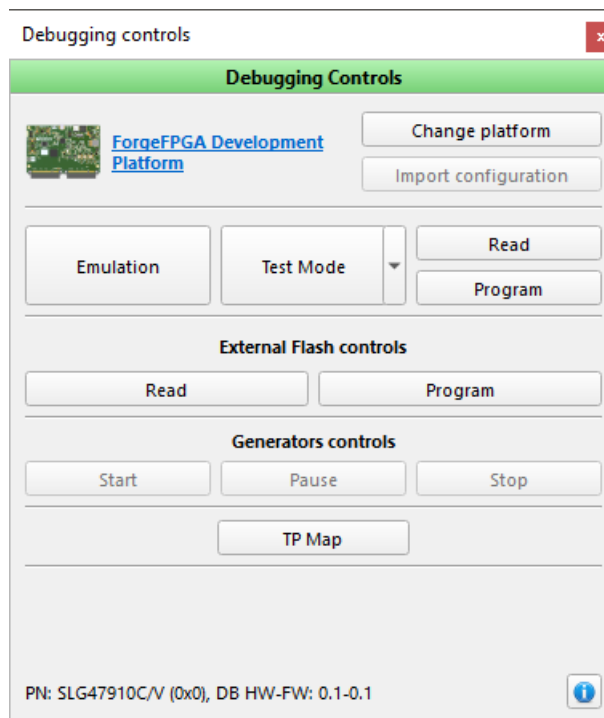


Figure 3: Debugging controls

Now let's review the different parts of the Debugging controls (Figure 3)

- a. **Change platform** - select type of hardware platform with supported features.
- b. **Import configuration** - Allows user import configuration of test points from another platforms.
- c. **Emulation** - the current project will be loaded to the chip (but not programmed) and will be ready for test on the hardware board.

d. Test Mode - Test mode is used for connecting or disconnecting the chip's I/O pads to TP controls, configured by user. Also, a user can check the programmed chip using the test mode without emulation. To do this, turn on the test mode and internal VDD button. The test mode can work without power on the chip. User will control the power manually. Another feature of the Test Mode is that it can test with the conditions from Flash Memory.

e. Read - Read chip using hardware board.

f. Program - Program chip with the current project. For some chip models user can configure programming process by clicking Programming options at Program button. As SLG47910 is OTP, it can be programmed only once.

g. External Flash Controls -The data can also be read from External Flash. The read data can be accessed from the Project Data window after the data has been read.

The current bitstream can also be loaded onto the external Flash by pressing the Program button under this category.

h. Generator Controls - During Emulation, you can start all the test points together, or pause them or even Stop them from running altogether as well.

i. TP Map - The Test Points of each pin will be shown next to the respective pin. (Figure 4)

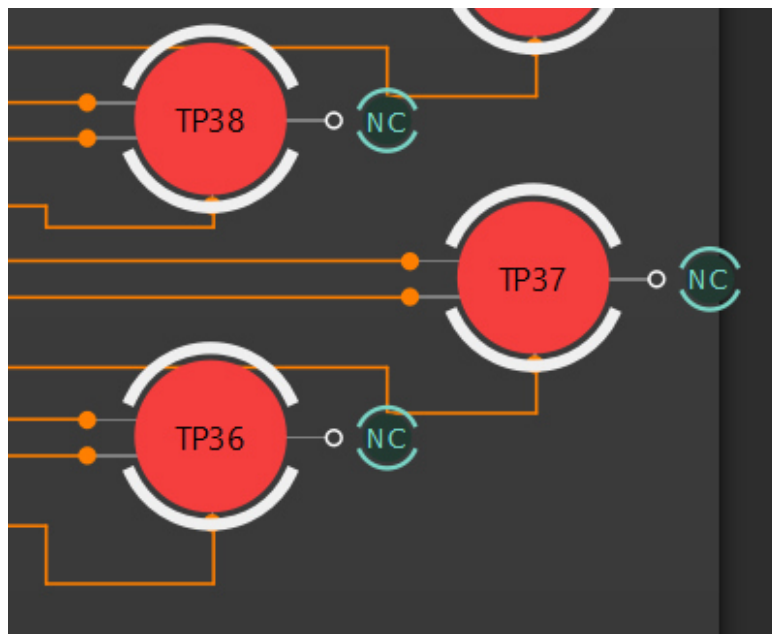


Figure 4: TP Map

j. PN: SLG47910 C/V(0x0), DB HW-FW: 1.2-0.3

After the Development board with the Chip in the Socket Adapter Board has been connected, we can see the PN (Part Number) SLG47910 being displayed at the bottom of the Debugging Controls Dialog Box (Figure 3). The Debugging Control Window also indicates the Development Boards (DB) Hardware (HW) and Firmware (FW) version.

3.4 Add Test Point Controls

Debugging tool controls are used to configure input signals on external inputs of chip. There are many ways in which we can manage the chip input signals.

Use the context menu on the GPIOs with a right click to see the options of connectivity (Figure 5)

3.4.1. Types of areas

1. NC (not connected)



Figure 5: N/C (Not connected)

2. Set to VDD



Figure 6: Set to VDD

3. Set to Ground GND



Figure 7: Set to GND

4. Configurable Button:

The Configurable Button has 3 aspects to it. The user can configure the button to establish the connection as VDD, GND or Hi-Z. If the user wants the GPIO to be a fixed connection to VDD, then user needs to select the LATCH option. The LATCH option will make sure that the GPIO is connected to a particular signal (VDD/GND/Hi-Z) unless changed. This will enable the button to be LATCHED to VDD, unless changed. (See Figure 8a-b)

The default connection option is VDD/GND, but it can be changed to Hi-Z by selecting Hi-Z option from the Upper Connection or the Lower Connection option as per the need from the context menu. In Figure 8b, you can see that the Upper Connection "U" corresponds to Hi-Z, and the Bottom Connection "B" corresponds to GND.

Say, you configure the button as UNLATCHED and set the default connection as Upper Connection "U" which equals to Hi-Z and the Bottom Connection "B" to GND. Whenever the button is pressed, there will be toggle in the waveform between Hi-Z and GND at that very moment with the default waveform being at Hi-Z. (see Figure 8b)

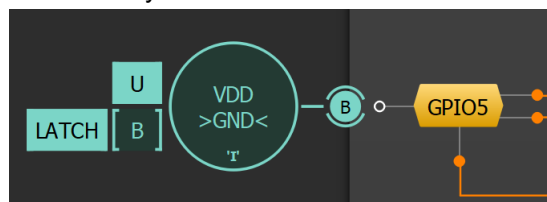


Figure 8a: Latched Button with Upper Connection as VDD

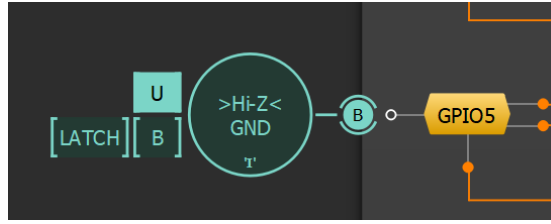
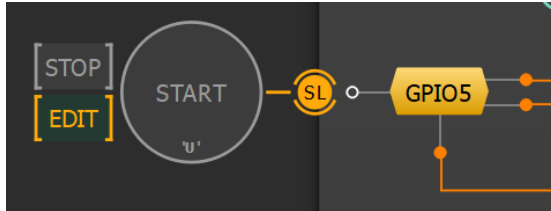


Figure 8b: Unlatched Button with Upper Connection as Hi-Z



5. Synchronous Logic Generator

Figure 9: Synchronous Logic Generator

Right click on the NC of the desired GPIO and from the context menu select the Synchronous Logic Generator option. The synchronous Logic Generator is used for generating the logic pulses and waveforms for each GPIOs. The 'Edit' Button allows the configuring of the signal. (Figure 9-10)

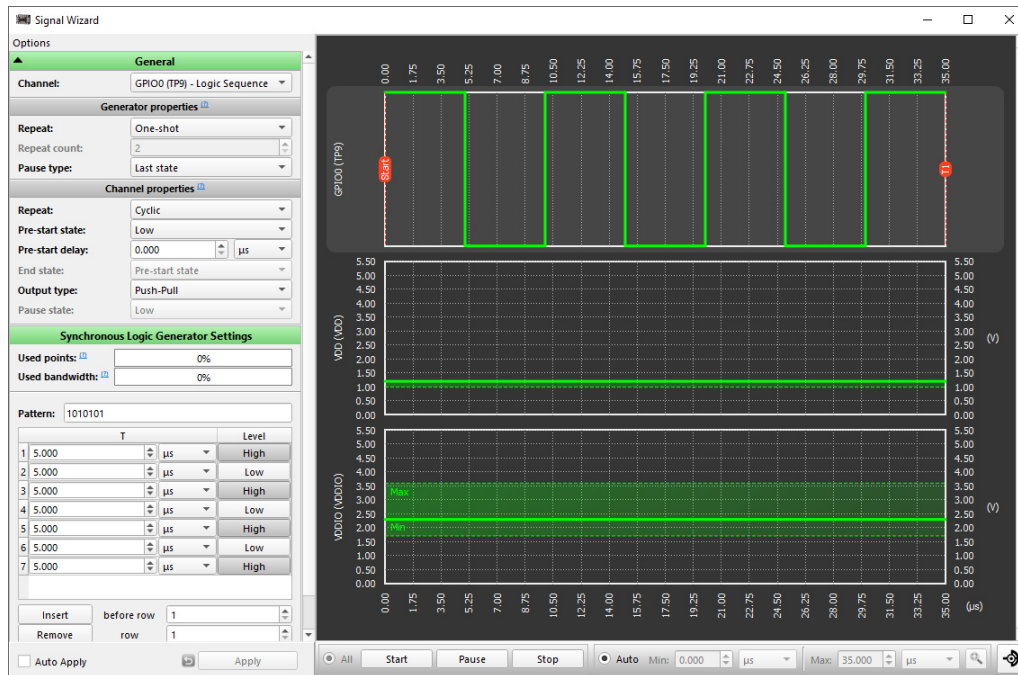


Figure 10: Signal Wizard for Synchronous Logic Generator

Below are the different features in the Signal Wizard for Synchronous Logic Generator (Figure 12)

1. **Used Points:** Amounts of points which are already used by patterns on all channels. Point indicates a moment when generator changes a state on at least on channel.
2. **Used Bandwidth:** Percentage of used resources needed to successfully execute generator's pattern.
3. **Pattern:** 0-low/ 1-high level. We can set the pattern of pulse levels.
4. **Repeat:** One Shot / Cyclic / Custom
5. **T / levels:** it sets the duration of each pulse
6. **Insert:** To insert pulse before selected position
7. **Remove:** remove pulse from the selected position

4-bit Counter

8. **Invert:** to invert the pattern from 1010 to 0101
9. **Level Count:** to insert the total number of pulses to be generated

4. Logic Analyzer

The logic analyzer allows the user to test how the design behaves on the ForgeFPGA silicon during emulation, whether in testing mode without external tools. The user can pop up the logic analyzer (Figure 12) by clicking the Logic analyzer button on the ForgeFPGA tool bar (Figure 11).

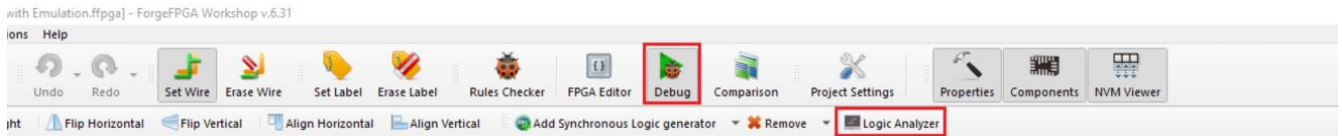


Figure 11: ForgeFPGA tool bar

To run Logic Analyzer, you need to press the Start button during Emulation or Test mode. Below are the different features in the Logic analyzer:

- *Mode* - Trigger Mode – Single, Auto, Normal
- *Samples per second* – data reading frequency.
- *Trigger* – opens Trigger configurations.
- *View options* – opens GPIO view options tab.
- *Debugging Controls* – To select between Emulation or Test Mode
- *Measurements* – Measures the width of the signal at each GPIO
- *Protocol Analyzer* – Used to analyze signal through I2C, SPI or UART Protocols
- *Start* – to start the Logic Analyzer

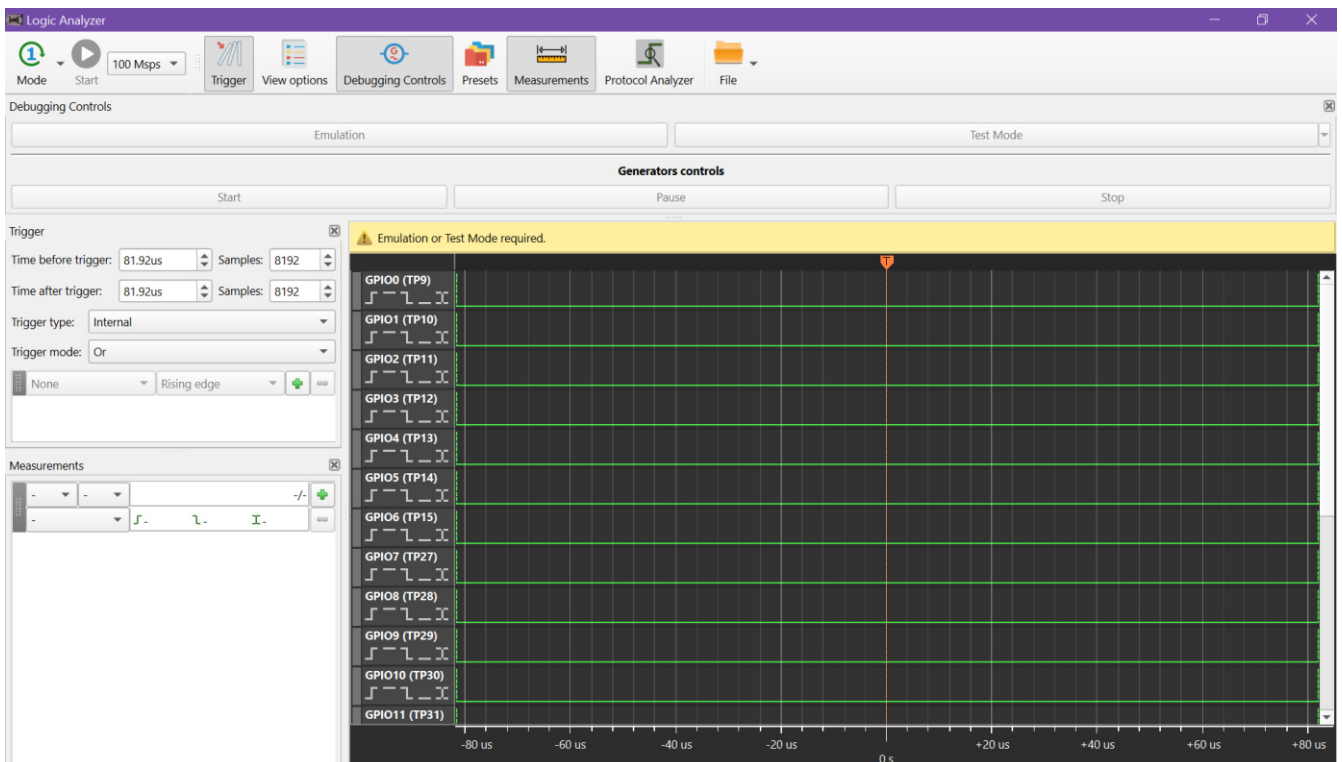


Figure 12: Logic Analyzer window

5. Ingredients

- ForgeFPGA Device SLG47910V
- FPGAPAK Development Board with USB cable and power supply
- FPGAPAK Socket Adaptor Board
- Latest Revision of ForgeFPGA Workshop software

6. Verilog Code

Shown below is the (*top*) module named counter. It's written in Verilog language and it describes how to design a 4-bit Up & Down Counter. It is available for download (AN_FG_001 4-bit Counter.ffpga).

```
(* top *) module counter(  
(* iopad_external_pin, clkbuf_inhibit *) input clk,  
  (* iopad_external_pin *) input nreset,  
  (* iopad_external_pin *) input up_down,  
  (* iopad_external_pin *) output [3:0]out_oe,  
  (* iopad_external_pin *) output osc_en,  
  (* iopad_external_pin *) output[3:0] counter  
);  
  
  reg [3:0] counter_up_down;  
  //Output enable need to be specified only for outputs  
  assign out_oe = 1;  
  
  //OSC  
  assign osc_en = 1'b1;  
  
  always @(posedge clk or negedge nreset) begin  
    if (!nreset)  
      counter_up_down <= 4'h0;  
    else if (up_down)  
      counter_up_down <= counter_up_down + 4'd1; // if up_down = 1  
    else  
      counter_up_down <= counter_up_down - 4'd1; // if up_down = 0  
  end  
  
  assign counter = counter_up_down;  
  
endmodule
```

7. Testbench

A testbench is written to test the functionality of the design without going through the hassle of programming or emulating the part. For simulation, GoConfigure is configured to work with GTKWave to display the waveforms of simulation. Below is the testbench for testing the stimulus written to test the counter. The testbench is written under the module name as counter_tb for it be recognized as a testbench by the software.

```
`timescale 1ns / 1ps  
  
module counter_tb;
```


4-bit Counter

```
reg clk;
reg nreset;
reg up_down;
wire [3:0] counter;

counter cnt1(.clk(clk), .nreset(nreset), .up_down(up_down), .counter(counter));

initial begin

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

forever #10 clk = ~clk;
end
initial begin

assign up_down = 1'b1;
assign nreset = 1'b0;

clk = 1'b0;

#30;
assign nreset = 1'b1;
#20;
#100;

assign up_down = 1'b0;
#30;
#50
$finish;
end

endmodule
```

8. Floorplan: CLB Utilization

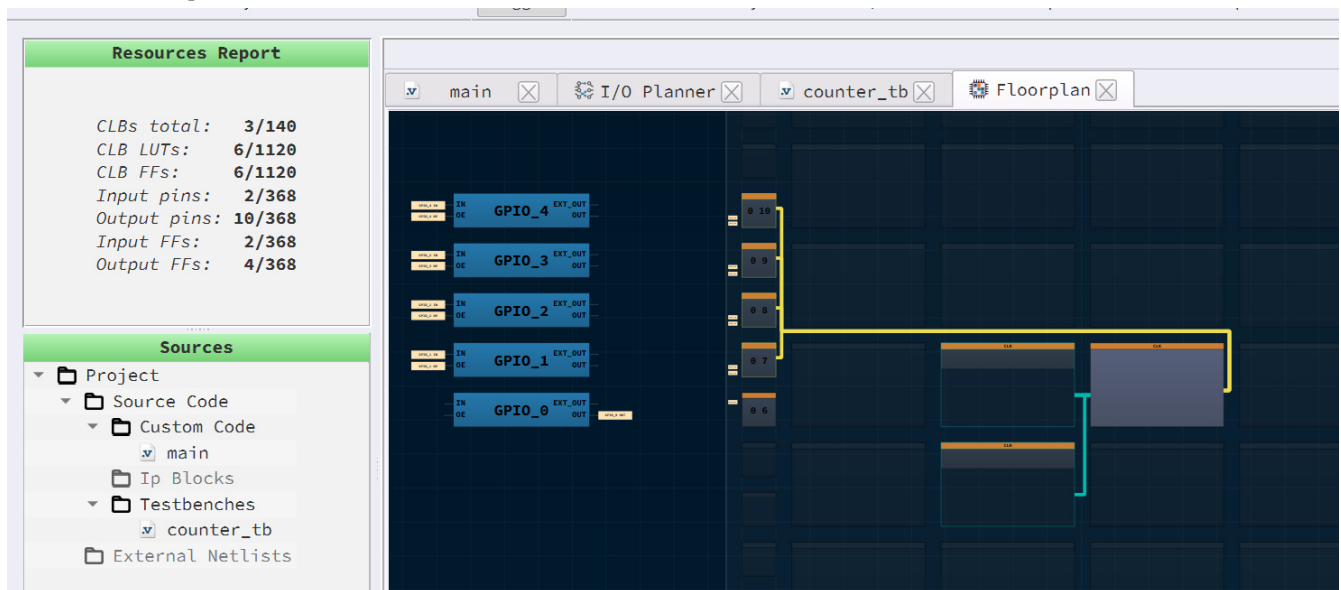
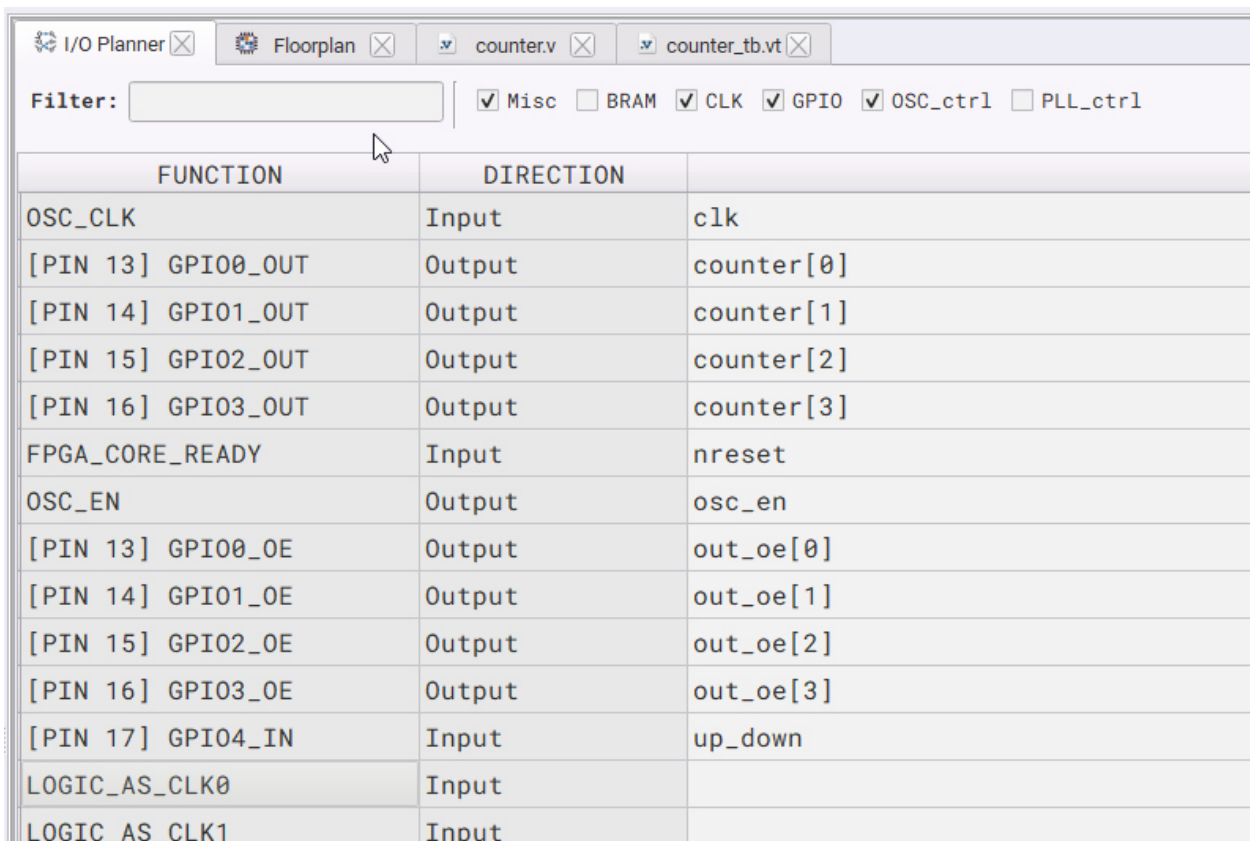


Figure 13: Floorplan

9. Design Steps

1. Launch the latest version of the Go Configure Software Hub. Select the SLG47910V device and the ForgeFPGA Workshop software will load.
2. From the ForgeFPGA tool bar (Figure 12) select the FPGA Editor tab.
3. Enter the Verilog code into the HDL editor and save the code using the save button on the top left corner of the FPGA Editor.
4. Open the IO planner tab on the FPGA editor. Assign the IOs that are in the Verilog code to GPIO pins on the device and save.
5. Next select the Synthesize button on the lower left side of the FPGA editor.
6. Select the Generate Bitstream button on the lower left side of the FPGA editor. Check the Logger and Issues tabs to make sure that the bit stream was generated correctly. Now click on the Floorplan tab and see the CLB utilization Press the Ctrl and the mouse wheel to zoom-in. Confirm that the IOs selected in the IO Planner (Figure 14). are shown in the floorplan (Figure 13). The IO Planner has been set in such a way that the board uses an external clock instead of the OSC clock on-board.



FUNCTION	DIRECTION	
OSC_CLK	Input	clk
[PIN 13] GPIO0_OUT	Output	counter[0]
[PIN 14] GPIO1_OUT	Output	counter[1]
[PIN 15] GPIO2_OUT	Output	counter[2]
[PIN 16] GPIO3_OUT	Output	counter[3]
FPGA_CORE_READY	Input	nreset
OSC_EN	Output	osc_en
[PIN 13] GPIO0_OE	Output	out_oe[0]
[PIN 14] GPIO1_OE	Output	out_oe[1]
[PIN 15] GPIO2_OE	Output	out_oe[2]
[PIN 16] GPIO3_OE	Output	out_oe[3]
[PIN 17] GPIO4_IN	Input	up_down
LOGIC_AS_CLK0	Input	
LOGIC_AS_CLK1	Input	

Figure 14: IO Planner

7. To simulate the testbench, make sure that the software is linked to the GTKWave software in the software settings.[4]. To run simulation, user can click on the respective button on the toolbar to initiate the simulation process. If there are no errors, then the GTKWave software should open automatically.
8. Users can observe the waveforms in GTKWave. See Figure 15

4-bit Counter

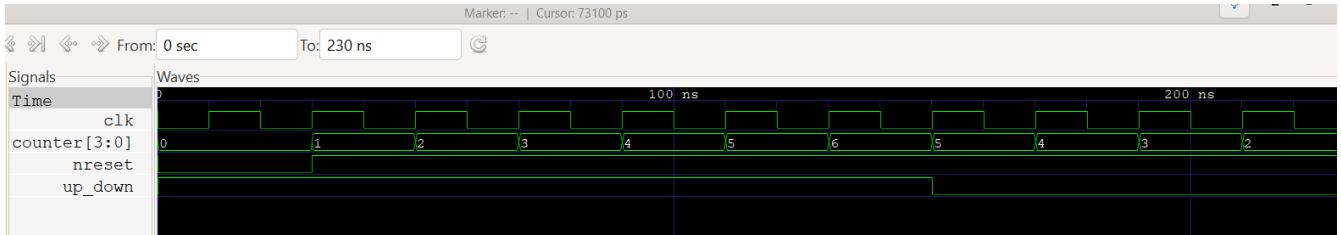


Figure 15. Simulated waveforms in GTKWave

- The counter design is now ready to load onto the FPGA using the development board. Connect the ForgeFPGA Development Board attached with the Adapter Board and the SLG47910 part placed in the socket mounted on to the board.
- Close the FPGA Editor and go to the ForgeFPGA widow (Figure 16). Selecting the Debug tab will enable the debug controls. Double click on the VDD pin and set VDD= 1.2v. Then double click on VDDIO pin and set VDDIO= 1.8v.
- In the ForgeFPGA Workshop window, select Change platform on the Debugging Controls tab. Choose the ForgeFPGA Development Platform then select Emulation. The Emulation button will toggle the design on and off.
- The IO planner has GPIO0 configured as an input. In the ForgeFPGA window, left click on the blue circle of GPIO0 and select Button. Now the input up_down can be toggled to count-up or count-down.

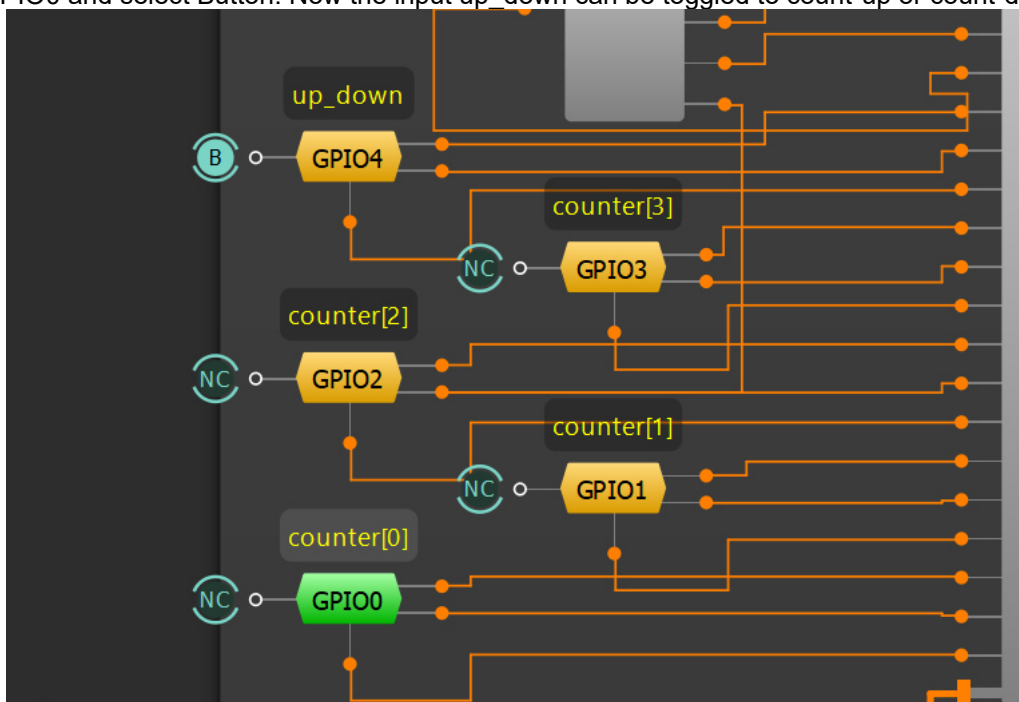


Figure 16: Forge FPGA Window

4-bit Counter

- Now the counter is loaded onto the FPGA device. GPIO [4:1] will have the outputs of the counter on them. Apply oscilloscope probes to GPIO [4:1] to view the counter output (Figure 17).

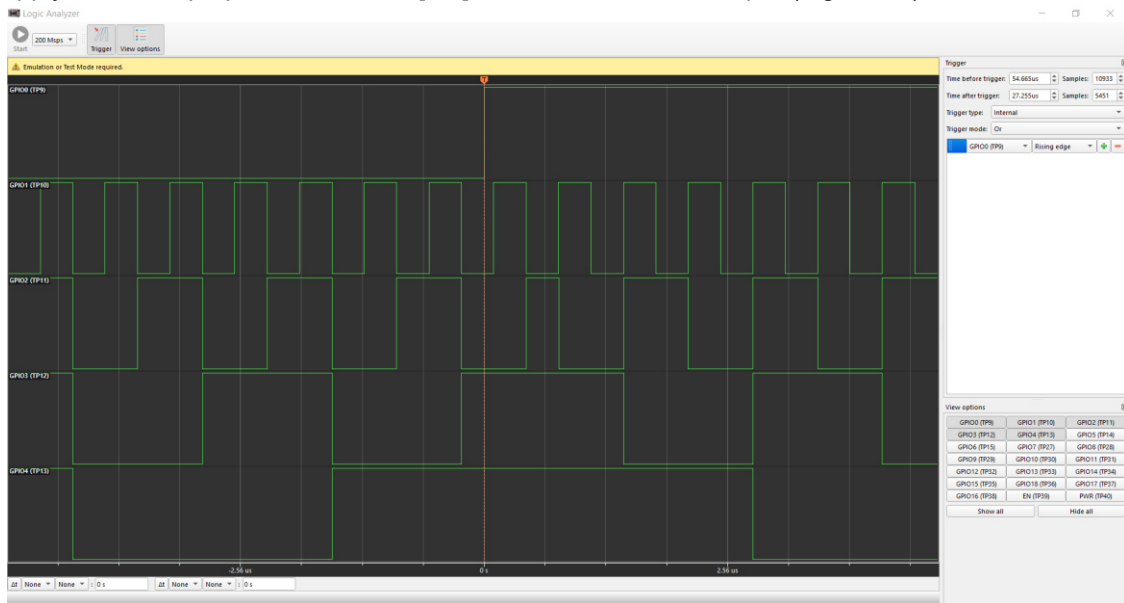


Figure 17: Waveforms of 4-bit counter after Emulation

10. Conclusion

The procedure outlined in this application example can be applied to any circuit design that is implemented in Verilog Code. The Emulation and the Test Points can be manipulated as per the application and all the output can be observed with the help of ForgeFPGA Development Board. The testbench is written to test out the functionality of the design with waveforms on GTKWave. The 4- Bit Counter.fpga design file can be downloaded as an example of the emulation features discussed in this application note. If interested, please contact the ForgeFPGA business Support Team.

11. Revision History

Revision	Date	Description
1.00	July 29, 2022	Initial release.
2.00	Feb 15, 2024	Updated as per BB revision
3.00	July 07,2024	Updated as per ForgeFPGA Workshop v6.43

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 who are 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 to develop 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 from 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 of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

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-us/.