This user guide will help you navigate the ForgeFPGA Workshop and understand the different features of the software.

Contents

1. Reference ............................................................................................................................................. 4
2. ForgeFPGA Workshop ............................................................................................................................. 4
   2.1 FPGA Editor Main Menu .................................................................................................................... 5
   2.2 Toolbars ............................................................................................................................................. 6
   2.3 HDL Editor ........................................................................................................................................ 7
   2.4 Logger/Issues Panel .............................................................................................................................. 7
   2.5 Control Panel .................................................................................................................................... 8
3. Writing Verilog Code ............................................................................................................................... 10
4. RTL Synthesis ......................................................................................................................................... 11
5. Generating Bitstream ............................................................................................................................... 11
6. Netlist ..................................................................................................................................................... 12
7. I/O Planner ............................................................................................................................................ 12
8. Floorplan ............................................................................................................................................... 14
9. Resources Utilization Report ................................................................................................................ 14
10. Timing Analysis .................................................................................................................................... 15
11. Post-Synth RTL ..................................................................................................................................... 16
12. Simulation ........................................................................................................................................... 17
   12.1 Writing a Testbench ......................................................................................................................... 17
   12.2 Simulating a Testbench ..................................................................................................................... 19
13. Build Folder .......................................................................................................................................... 19
14. IP Block Wizard .................................................................................................................................. 20
15. Macrocell Editor .................................................................................................................................. 21
16. Debug .................................................................................................................................................. 22
   16.1 Type of Hardware Platforms ............................................................................................................ 22
   16.2 Platform Configuration Guide ......................................................................................................... 23
   16.3 Configuring GPIOs ............................................................................................................................ 25
   16.4 Add Test Point Controls .................................................................................................................. 26
     16.4.1 Connection Options .................................................................................................................... 26
   16.5 Logic Analyzer ................................................................................................................................. 33
17. Clock Tree Structure ............................................................................................................................. 34
18. Flowchart ............................................................................................................................................. 37
19. Revision History ................................................................................................................................... 38
Figures

Figure 1. ForgeFPGA main window ................................................................. 4
Figure 2. ForgeFPGA Workshop user interface .................................................. 5
Figure 3. Work area ......................................................................................... 7
Figure 4. Logger and Issues tab ........................................................................ 8
Figure 5. Control panel .................................................................................... 9
Figure 6. Example of working with Verilog ....................................................... 10
Figure 7. EDIF Netlist .................................................................................... 12
Figure 8. Mapping I/O ports .......................................................................... 13
Figure 9. Place-and-Route results ................................................................. 14
Figure 10. Resources utilized report ............................................................... 15
Figure 11. Post-Synth RTL results ................................................................. 16
Figure 12. 16-bit counter testbench ............................................................... 18
Figure 13. Build folder contents .................................................................. 19
Figure 14. IP Block Wizard GUI .................................................................... 20
Figure 15. IP Block Wizard GUI with parameters ......................................... 21
Figure 16. Macrocell Editor ........................................................................... 21
Figure 17. Changing name of input/output pins ........................................... 22
Figure 18. Development Platform Selector ................................................... 23
Figure 19. Platform configuration guide ....................................................... 23
Figure 20. Debugging controls ..................................................................... 24
Figure 21. TP map ......................................................................................... 25
Figure 22. GPIO configuration options .......................................................... 25
Figure 23. Test Point control ........................................................................ 26
Figure 24. N/C (not connected) ................................................................... 26
Figure 25. Set to VDD .................................................................................. 26
Figure 26. Set to GND .................................................................................. 26
Figure 27. Latched button with upper connection as VDD .......................... 27
Figure 28. Unlatched button with upper connection as Hi-Z ...................... 27
Figure 29. Context menu options for configurable button ....................... 27
Figure 30. Default connection VDD/GND can be changed to Hi-Z .......... 28
Figure 31. PWM signal settings .................................................................. 29
Figure 32. Clock signal settings ................................................................... 30
Figure 33. UART signal parameters ............................................................. 30
Figure 34. Raw signal settings ..................................................................... 31
Figure 35. Synchronous Logic Generator .................................................... 31
Figure 36. Signal Wizard for Synchronous Logic Generator ...................... 32
Figure 37. Logic Analyzer .......................................................................... 33
Figure 38. Clock Tree Structure .................................................................. 34
Figure 39. IP planner for external loopback clock ....................................... 35
Figure 40. Phase Locked Loop signals ........................................................... 35
Figure 41. Oscillator signals ......................................................................... 36
Figure 42. IO Planner for Oscillator Clock .................................................... 36
Figure 43. BRAM Clock definitions in IO Planner ........................................ 36
1. Reference

For related documents and software, please visit our website:

Download our free ForgeFPGA Designer software [1] and follow the steps in this user guide. Use the Configuration Document to understand the different FPGA configuration modes [2]. Renesas Electronics provides a complete library of application notes [3] of design examples and an explanation of features and a description of internal FPGA blocks.


2. ForgeFPGA Workshop

When the user opens the Go Configure software and chooses the part SLG47910 and fills in the Project settings, this is the window the user comes across with.

![Figure 1. ForgeFPGA main window](image)
This top view of our 47910 FPGA core hardwired with all the components of it, such as PLL, OSC, BRAM, PWR, EN, and 18 GPIO. To edit the details of the FPGA core and design it, the user needs to enter the FPGA Editor mode by clicking on the FPGA Editor button in the toolbar (see Figure 1) or double-clicking the central FPGA Core. This will open another window which looks similar to Figure 2.

Development software consists of the main menu, toolbar, HDL Editor (main work area), logger panel, and control panel (see Figure 2).

![Figure 2. ForgeFPGA Workshop user interface](image)

2.1 FPGA Editor Main Menu

At the top of the Main Menu are the following tabs, File, Edit, Tools, Window, and Options (see Pink Box in Figure 2) which are described below:

- Selecting File shows the following choices:
  - **Save** – save current project
  - **Save As** - save current project under another name
  - **Add Block** – open IP Blocks Wizard
  - **New Custom Module** – add new module
  - **New Custom Testbench** – add new testbench
  - **Import**
    - Custom Module
    - Custom Testbench
    - Netlist
  - **Export**
    - Custom Module
    - Custom Testbench
    - IP Block
- Selecting **Edit** shows the following editable tabs:
  - Undo
  - Redo
  - Delete
  - Cut
  - Copy
  - Paste
  - Select All
  - Find
- Selecting **Tools** shows the following tabs:
  - Run Synthesis – synthesize Verilog into low-level constructs
  - Generate Bitstream – compilation, and mapping low-level constructs to specific device blocks and generate the Bitstream file
  - Floorplan
    - Load custom PnR
  - Simulation
    - Simulate Testbench
  - I/O Planner
    - Clear data
    - Import I/O Spec
- Selecting **Window** shows the following tabs:
  - Netlist – read-only tab with a description of the connectivity of an electronic circuit
  - Post-Synth RTL (register transfer layer)
  - I/O Planner
  - Floorplan
  - Resources
  - Timing Analysis
  - Macrocell Editor
  - Logger
- Selecting **Options** has the following tab:
  - Settings.

### 2.2 Toolbars

Toolbars provides a quick access to frequently used functions (see Red Box in Figure 2):

- Save
- Add Block
- New Custom Module
- Logger
- Netlist
2.3 HDL Editor

The HDL Editor is the central white portion of the software where the user can type their desired Verilog Code. The work area has a split screen option (see Figure 3) which allows the user to split the screen and view 2 different pages of the software next to each other. The user can choose what they want to view in either of their split screen by selecting their choice from the control panel or from the buttons on the toolbar. The user can also close the split screen when not needed.

![Figure 3. Work area](image)

2.4 Logger/Issues Panel

At the bottom of the window, the user can find the Logger/Issues Panel (see Yellow Box in Figure 2). When the ForgeFPGA Workshop generates messages, they are printed out in the Logger panel. This happens whenever the user synthesizes the Verilog code/Generates Bitstream or runs simulation, all the messages from various tools are displayed in the logger during these procedures. It displays the output from various tools, as well as certain warning and error messages, that were produced processing your design.

User can switch between the Logger and the Issues tab in this Panel. Under the Issues tab, the user can read the warning and the error messages that the software generates automatically as and when necessary.
The user can also clear the contents of the logger/Issues section by pressing the CLEAR tab on the top-right corner of this section (see Figure 4).

![Logger and Issues tab]

2.5 Control Panel

Most of the main controls are concentrated on the Control panel (see Figure 5).

- **Resources Needed** – an extract from the resource usage report that shows what part of available resources was utilized by the design (Orange Tab).

- **Sources** - it has a list of all the Verilog codes we have in this current Project File. It is categorized into 3 main subcategories: Custom Code (the code that you create), IP Blocks (If the user adds/opens an IP Block then it will be listed under this category), and Testbenches (all the testbenches including the IP Block Testbench will be listed in this category). If the user intends to delete a particular source code, then the user can right click on the name on the file from the list and click Delete (Grey Tab).

- **Synthesize and Generate Bitstream** – main controls to run toolchain on your design to produce chip configuration.
Figure 5. Control panel
3. Writing Verilog Code


There are a few special code attributes, those are important to keep in mind while working with the synthesis tools:

i. (* top *) — the main module of your design should be marked with this attribute so the toolchain can successfully recognize which of the modules in the design is the top one.

ii. (* clkbuf_inhibit *) — clocks signals in the input list of the main module should be marked with this attribute to prevent clock buffer insertion by the synthesis tool, which may lead to the distortion of the clock signal name in the resulting netlist.

iii. (*iopad_external_pin*) — all the external pins that are used in any module need to be marked with this attribute.

```
// Example project
(* top *) module counter_15bit #(   
    parameter DUTY = 65535 // Value from 1 to 65535
)   
   // Main inputs 
(*) iopad_external_pin, clkbuf_inhibit *) input clk, 
(*) iopad_external_pin *) input nreset, 
   // OSC config outputs 
(*) iopad_external_pin *) output osc_en, 
(*) iopad_external_pin *) output osc_mode, 
   // Custom IO 
(*) iopad_external_pin *) input clear, 
(*) iopad_external_pin *) output clear_oe, 
(*) iopad_external_pin *) input en, 
(*) iopad_external_pin *) output en_oe, 
(*) iopad_external_pin *) output out 
);   
assign osc_en = 1'b1; 
assign osc_mode = 1'b0; 
assign clear_oe = 1'b1; 
assign en_oe = 1'b1; 
reg [15:0] counter; 
always @(posedge clk) begin 
    if (nreset) 
```

Figure 6. Example of working with Verilog
4. RTL Synthesis

ForgeFPGA Workshop comes with a built-in synthesis tool that takes input design and produces a Netlist. While performing synthesis, the input design is analyzed and converted into gate-level representation.

To run synthesis on your design, you can press the Synthesis button on the bottom of the control panel or from the main menu Tools → Run Synthesis. During the process of synthesis, the software also checks for any Syntax errors in the Verilog code written and points out to the line in error in the logger section (see Figure 4). The Synthesis process will not be completed until the code is free of any syntax errors.

5. Generating Bitstream

To prepare your design to be sent to the device you need to perform the place-and-route procedure, that takes the elements of the synthesized netlist and maps its primitives to FPGA physical resources. You can do this after successfully generating netlist and pressing Generate Bitstream button on the bottom of the control panel or from the main menu Tools → Generate Bitstream.

The user can read the logger to view the background steps once the Generate Bitstream button is clicked. In the background, the software automatically performs steps like the Technology Mapping, Clustering & Floor Planning, Placement & Optimization, Routing and Resource calculation.

If there is an error or warning in any of the steps during the generation of Bitstream, the process will be incomplete, and the issues will be pointed out in the Issues section of the console.

The generated bitstream is a hex file which if generated correctly is sent automatically to the SLG47910 part.
6. Netlist

After successfully performing the Synthesis procedure, a Netlist is generated. It describes the components and connectivity of the source design and is needed to perform the subsequent Place-and-Route procedure. The Netlist can be accessed from the clicking the Netlist button on the toolbar or from the main menu Window → Netlist.

```
1 (edif counter_16bit
2 (edifVersion 2 O O
3 (edifLevel O
4 (keywordMap (keywordlevel O)
5 (comment "Generated by Yosys 0.10"
6 (external LIB
7 (edifLevel 0
8 (technology (numberDefinition)
9 (cell GND
10 (cellType GENERIC
11 (view VIEW_NETLIST
12 (viewType NETLIST
13 (interface (port G (direction OUTPUT))
14 )
15 (cell VCC
16 (cellType GENERIC
17 (view VIEW_NETLIST
18 (viewType NETLIST
19 (interface (port P (direction OUTPUT))
20 )
21 )
22 )
23 (cell INV
24 (cellType GENERIC
25 (view VIEW_NETLIST
26 (viewType NETLIST
27 (interface
28 (port I (direction INPUT)
29 (port O (direction OUTPUT)
30 )
```

Figure 7. EDIF Netlist

7. I/O Planner

Each I/O port, available on the FPGA has a dedicated function, that can be mapped to your design using the I/O Planner tool. A configuration file is generated during synthesis which is used during place-and-route.

The user can access the I/O Planner by click the I/O Planner button on the toolbar or from the main menu Window → I/O Planner

The I/O Planner tool is represented as a table with three columns, each of which has a special meaning (see Figure 8):

- **POSITION** – the coordinate of a certain device I/O port on the FPGA.
- **FUNCTION** – dedicate function, assigned to the port.
- **PORT** – editable column, where you can input ports from your Verilog design, to connect them to the desired functionality. By double-clicking the desired port row, the user can choose from the list of all the ports defined in the Verilog Code.

To make navigation easier, the I/O Planner provides several filter checkboxes, that can show/hide groups of ports according to their functionality.

The user can also clear all the data fed inside the I/O Planner by going to main menu Tools → I/O Planner → Clear data as well as the user can import any I/O Specifications design into the software by going to the main menu Tools → I/O Planner → Import I/O Spec. The user must select the PNR_IO file from the build folder (see Figure 13) of an existing project file. This will populate the I/O Planner with the new I/O Specs from the imported file.
Figure 8. Mapping I/O ports

<table>
<thead>
<tr>
<th>POSITION</th>
<th>FUNCTION</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOB tile[0, 0] coord[31, 31] Output1</td>
<td>BRAM3_READ_ADDR[4]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 31] Output0</td>
<td>BRAM3_READ_ADDR[5]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 30] Output1</td>
<td>BRAM3_READ_ADDR[6]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 30] Output0</td>
<td>BRAM3_READ_ADDR[7]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 29] Output1</td>
<td>BRAM3_READ_ADDR[2]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 29] Output0</td>
<td>BRAM3_READ_ADDR[3]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 28] Output1</td>
<td>BRAM3_READ_ADDR[0]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 28] Output0</td>
<td>BRAM3_READ_ADDR[1]</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 27] Output1</td>
<td>[PIN 23] GPIO8_OE</td>
<td>clear, clear_oe,</td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 27] Output0</td>
<td>[PIN 23] GPIO8_OE</td>
<td>clk, en, en_oe,</td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 27] Input0</td>
<td>[PIN 23] GPIO8_IN</td>
<td>reset, osc_en</td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 26] Output1</td>
<td>[PIN 24] GPIO9_OE</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 26] Output0</td>
<td>[PIN 24] GPIO9_OE</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 26] Input0</td>
<td>[PIN 24] GPIO9_IN</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 25] Output1</td>
<td>[PIN 1] GPIO10_OE</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 25] Output0</td>
<td>[PIN 1] GPIO10_OE</td>
<td></td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[31, 25] Input0</td>
<td>[PIN 1] GPIO10_IN</td>
<td></td>
</tr>
</tbody>
</table>
8. Floorplan

To help visualization your design, the results of the Place-and-Route procedure are shown on the Floorplan. Here you can check, how the primitives from the netlist were placed and interconnected, as well as how I/O ports were mapped to the special internal blocks and GPIOs.

User can launch the Floorplan window by clicking the Floorplan button on the toolbar or from the main menu Windows → Floorplan. User can also load custom PnR settings (.log file) to display their desired connections in the floorplan. To do so, the user can go to Tools → Floorplan → Load custom PnR.

The user must select the PNR_IO file from the build folder (see Figure 13) of an existing Project File. This will populate the floorplan and display the imported floorplan plan on the window.

9. Resources Utilization Report

After successfully performing the Place-and-Route procedure (during Generate Bitstream step), a full report of the resources used is generated (see Figure 10). This report shows how many CLBs, FFs, I/Os, and LUTs are being used for synthesizing the design.

A list of the resources utilized are also shown in the Control Panel (see Figure 5). In Figure 5, if it says that the current Project file used 3/260 Input Pins, it means that there is a total of 360 I/O Pins in the IOBs and 3 of those Input Pins are being used similarly for the Output Pins.

User can launch the Resource Utilization window by clicking the Resources button on the toolbar or from the main menu Windows → Resources.
10. Timing Analysis

Page under construction
11. Post-Synth RTL

At the Register-Transfer Level the design is represented by combinational data paths and registers. RTL synthesis is easy as each circuit node element in the netlist is replaced with an equivalent gate-level circuit. In Post-Synthesis RTL, it takes the synthesized inputs as netlist. This helps in providing information about clock and other clock-related logic in design, which enables additional I/O planning.

The Post-Synth RTL can be accessed after successfully Synthesizing the Verilog Code. The Post-Synth RTL results is a .v file and will also be found in the build folder. User can view it by clicking on the Post-Synth RTL button in the toolbar or from the main menu Window → Post-Synth RTL (see Figure 11). This report contains all the connects made within the module between the LUTs, FDREs and Carry-Chain logics.

Figure 11. Post-Synth RTL results
12. Simulation

The most crucial step in successfully implementing any system is to verify the design and its functionality. Verifying a complex system after implementing the hardware is not a wise choice. It is ineffective in terms of money, time, and resources. Hence, in the case of FPGA, a testbench is used to test the Verilog source code [6].

The ForgeFPGA works in correspondence with another software called GTKWave [7] for simulating the testbench and for verifying the functionality of the source code. The GTKWave software needs to be downloaded separately using the link posted on the website.

12.1 Writing a Testbench

User needs to open a new testbench module to write the testbench in it. The system recognizes any module as testbench only if it is saved as `filename_tb`. To open a testbench module go to File → New Custom Testbench as save the name of the module as `filename_tb`.

To make things easy, the software opens the testbench module with a few lines of code to act as a guideline for writing the testbench. Please refer to the ForgeFPGA Test Bench User Guide [6] for details on how to write a testbench. See Figure 12 for a testbench example.
Figure 12. 16-bit counter testbench

```verilog
// Custom testbench
timescale 1ns / 1ps
module counter_16bit_tb;
    reg clk;
    reg nreset;
    wire osc_en;
    wire osc_mode;
    reg clear;
    wire clear_oe;
    reg en;
    wire en_oe;
    wire out;

// defining the clock
always @ (posedge clk)
    $dumpfile ("counter_16bit_tb.vcd");
    $dumpvars (0, counter_16bit_tb);

// stimulus
    clk <= 0;
    nreset <= 0;
    clear <= 0;
    en <= 0;
    #10
    nreset <= 1;
    clear <= 1;
    en <= 0;
    #20
    nreset <= 1;
    clear <= 0;
    en <= 1;
    #300
    $finish;
endmodule
```

12.2 Simulating a Testbench

ForgeFPGA uses GTKWave software to simulate the testbench. After the testbench has been written, press the Simulate Testbench button on the toolbar to launch GTKWave or from the main menu go to Tools → Simulation.

If the testbench is correct and doesn’t contain any syntax errors, then the GTKWave software will launch automatically, if it does not launch, check the logger for any syntax related issues in the testbench code and make necessary changes. Please refer to the ForgeFPGA GTKWave User Guide [7] for GTKWave usage and features.

13. Build Folder

As soon as the software has successfully run the Synthesis and the Generate Bitstream step, it creates a BUILD folder at the same location as the project file (filename.ffpga). The build folder will have the same name as the project file: build_filename. This folder will be populated with report files generated during this process. Let us discuss some of the important files in the build folder:

- **FPGA_bitstream**: the hex file containing the generated bitstream of the Verilog code. This file automatically gets uploaded onto the part SLG47910.
- **PNR_IO**: this is a .log file which contains all the I/O Specifications manually inserted by the user in the IO Planner. It can also be used to import an externally.
- **Netlist**: this is a .edif file. It describes the components and connectivity within the source design. A netlist can also be externally imported into the software.
- **PNR_ROUTE**: This is a .log file. This has a detail of all the internal connects of the nets during PnR.
- **Post_synth_results**: this is .v file. This has the RTL code generated after successful synthesis.
- **Dump files**: this is a .v file which is a Verilog file. This has the main source code and the testbench code dumped for future reference.
- **Resource-utilization-report**: this file contains the same report as seen in Figure 10.
- **Filename_tb**: this is a .vcd file. This is the dumpfile which gets created along with the testbench. This contains all the changes in the variables mentioned in the testbench.

![Figure 13. Build folder contents](image-url)
14. IP Block Wizard

To help with the development process a library of easy-to-integrate IP Blocks is provided. Here you can find Verilog code for various hardware modules, accompanied by the testbenches to check their functionality.

To launch an IP Block, go to **File → Add Block** → select the desired IP Block and click **Select** → change parameters as needed and click **Select** → write the **Name** of the module → **Create** or click **Add Block** from the toolbar → select the desired IP Block and click **Select** → change parameters as needed and click **Select** → write the **Name** of the module → **Create**.

Inside the IP Block Wizard GUI, the user can find the schematic, resource estimation along with the description of the block selected. The GUI gives a detailed explanation of the all the input and output pins of the block and allows the user to change the parameters as desired. This gives the user the flexibility to create the Verilog code of the IP block needed and its associated testbench with just a few clicks.

![IP Block Wizard GUI](image.png)

*Figure 14. IP Block Wizard GUI*
User can create multiple IP Blocks simultaneously as well. All the created IP Blocks and their respective testbenches can be seen listed under the green tab in the control panel (see Figure 5).

15. Macrocell Editor

The software has another mode called Macrocell mode, which allows the user to draw the desired circuit in a schematic view. To launch the Macrocell mode, the user can click the Macrocell Editor button on the toolbar, or the user can go to main menu Window → Macrocell Editor. The Macrocell Editor screen looks like Figure 16.
The Macrocell Editor has two parts; 1st on the green screen is where the schematic of the circuit is made by using the components, and 2nd on the right side of the screen is the library of all the Sequential & Combinational Components. The user can simply drag and drop the desired component from the library on to the green screen to complete the circuit. Once the user has placed the selected component and its location, the user can press ESC on their keyboard to deselect the component. Two components can be joined by simply joining the ports of the components using a wire which can be produced by clicking the two ports to be joined.

The user can also change the name of their inputs & output pin component by typing the name below it (see Figure 17).

After the user is satisfied with the circuit, the ForgeFPGA software makes it easy for the user to generate the Verilog code of the created design. On the top right corner of the green screen of the Macrocell Editor, there is a button called Generate Verilog. Once the user presses this button, the software will automatically create the Verilog code of the created design in the Macrocell Editor window. The created Verilog code can be seen listed as mcm_generated under the Custom Code (Blue Tab) section of the control panel. The mcm_generated code is a ready to synthesize code and can also be used synchronous with other modules already in the design.

16. Debug

The design is now ready to be tested on the development board. The generated bitstream is automatically sent to the part which is ready to be programmed. The user needs to now switch to the main screen of the ForgeFPGA Software.

The Debug button in the toolbar 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.

16.1 Type of Hardware Platforms

After the Debug button is pressed (make sure to enter the Project Specifications before using Debugging Tool), the Development Platform Selector and the two hardware options will be displayed (see Figure 18). Select the ForgeFPGA Development Platform option.
16.2 Platform Configuration Guide

Recommended Platform Configuration Guide 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 (see Figure 19).
Now let's review the different parts of the Debugging controls (see Figure 20):

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 the user. Also, a user can check the programmed chip using the test mode without emulation. To do this, turn on the test mode. 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 (see Figure 21).

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 (see Figure 20). The Debugging Control Window also indicates the Development Boards (DB) Hardware (HW) and Firmware (FW) version.
16.3 Configuring GPIOs

We have 18 GPIOs on the board and it can also be seen on the software. These GPIOs are hardwired to the FPGA core, however we can configure these GPIO for different Output Modes.

When the user clicks the desired GPIO he wants to configure, a properties panel opens on the left side of the screen. The user has an option of choosing which Output Mode is suitable for their code and the Resistor type (Hi-Z/Pull Up) of the GPIO. See Figure 22 for the various options available to choose from. Once the user has chosen his/her desired configuration setting, they can click Apply to apply the settings to the GPIO.

![Figure 22. GPIO configuration options](image-url)
16.4 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 (see Figure 23).

![Figure 23. Test Point control](image)

16.4.1. Connection Options

1. **NC (not connected)**

![Figure 24. N/C (not connected)](image)

2. **Set to V\text{DD}**

![Figure 25. Set to V\text{DD}](image)

3. **Set to Ground GND**

![Figure 26. Set to GND](image)

4. **Button**

The Configurable Button has 3 aspects to it. The user can configure the button to establish the connection as V\text{DD}, GND or Hi-Z. If the user wants the GPIO to be a fixed connection to V\text{DD}, then user needs to select the LATCH option. The LATCH option will make sure that the GPIO is connected to a particular signal (V\text{DD}/GND/Hi-Z) unless changed. This will enable the button to be LATCHED to V\text{DD}, unless changed (see Figure 27).
The default connection option is V\text{DD}/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 28, 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 30).

![Figure 27. Latched button with upper connection as V_{DD}](image)

![Figure 28. Unlatched button with upper connection as Hi-Z](image)

![Figure 29. Context menu options for configurable button](image)
5. Parametric Generator

Right click on the NC of the desired GPIO and from the context menu select the Parametric Generator option. This option is used for generating 4 different types of signals namely: PWM (Pulse Width Modulation), Clock, UART Transmitter (Universal Asynchronous Receiver/Transmitter) and Raw Signal.

Within each signal type there are different parameters that the user can set to vary the signal according to the application.

**PWM Signal**: within the PWM signal, the user can set the period of the signal ranging from 5 ns to 1 s. The user can also vary the Duty Cycle of the signal and define the number of times he/she wants to repeat the signal. With each repetition added, the period of the signal also adds up. Example, if the user sets the period to 10 ns and repeat to 2, then the signal will reflect a waveform with 20 ns as period (see Figure 31).

The user can also add another PWM signal by clicking on the + button. The user can choose different numbers for the parameters and the signal will append the first signal.

---

**Figure 30. Default connection VDD/GND can be changed to Hi-Z**
**Figure 31. PWM signal settings**

**Clock:** within the Clock signal, the user can set the period of the signal ranging from 10 ns to 1 s whose equivalent frequency is 1 Hz to 100 MHz. The user can also set the number of times he/she wants to repeat the signal. With each repetition added, the period of the signal also adds up (see Figure 32).

The user can also add another CLOCK signal by clicking on the + button. The user can set the parameters as same as the previous signal or different and the 2nd signal with append the 1st signal.

**UART Transmitter:** UART Tx is used to serially transfer data asynchronously. The user can set the Baud Rate specifying at which rate he/she wants to transfer the data. UART transmission happens in the form of packets. Each part of the packet can be modified from the options available in the parametric generator namely, Data Frame, Data (in hexadecimal form), parity bit, stop bit size and define the order bit as MSB or LSB. (see Figure 33).
Figure 32. Clock signal settings

Figure 33. UART signal parameters
RAW: within the RAW signal, the user can define the pattern of the signal he/she wants to generate such as 1100 or 10101 and as the pattern is specified the time duration for each level will get populated. The user can define the time duration for each level. The user can also keep repeating the pattern n number of times by increasing the level count and the signal will appear accordingly (see Figure 34).

![Figure 34. Raw signal settings](image)

6. 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. (see Figure 35).

![Figure 35. Synchronous Logic Generator](image)
Below are the different features in the Signal Wizard for Synchronous Logic Generator (see Figure 36).

i. **Used Points**: Amounts of points which are already used by patterns on all channels. Point indicates a moment when the generator changes its state on the channel.

ii. **Used Bandwidth**: Percentage of used resources needed to successfully execute generator's pattern.

iii. **Pattern**: 0-low/1-high level. We can set the pattern of pulse levels.

iv. **Repeat**: One Shot/Cyclic/Custom.

v. **T / levels**: It sets the duration of each pulse, and the pulse level - High or Low.

vi. **Insert**: To insert pulse before selected position.

vii. **Remove**: Remove pulse from the selected position.

viii. **Invert**: To invert the pattern from 1010 to 0101.

ix. **Level Count**: To insert the total number of pulses to be generated.
16.5 Logic Analyzer
ForgeFPGA Workshop has a built in Logic Analyzer which can be used to observe the waveforms during testing.

![Logic Analyzer](image)

Figure 37. Logic Analyzer

The signals set through the synchronous Logic Generator in Figure 37 can also be observed in the Logic Analyzer window by selecting the GPIOs the user needs to view from the list of GPIOs available in the window.

If the trigger type is internal and not through a hardware button, then select internal as the Trigger type on the right side of the screen and select the GPIO that is used as trigger signal by clicking on the + sign and selecting the needed GPIO from the dropdown list and selecting the trigger edge for the respective GPIO.

After the user is satisfied with the settings of the Logic Analyzer, the user can press START under the generator controls section. This turns on the signals in the Synchronous Logic Generator and then click on Green Play Button on left upper corner called Start to start the Logic Analyzer. The user can select the sample rate of the Analyzer as well ranging from 200 Msps to 500 Msps. The black screen in the center will then light up and the waveforms of can be observed. The user can Zoom-in and Zoom-out by pressing the Ctrl and the mouse wheel as well to view the desired part of the waveform.
17. Clock Tree Structure

We have 4 global clock tree sources in SLG47910, namely external_loopback_clock0, external_loopback_clock1, PLL_Clock and Oscillator_Clock. The part also has a read and write clock dedicated to the BRAMs placed on the North and the South side of the FPGA Tile.

Let us understand how to configure each of these clock sources for optimum and efficient utilization:

1. Ext_loopback_clock

SLG47910 has 4 such external_loopback_clocks divided into groups of two (0/1) and each pair is used as input and output. In situations where we use the external clock, it is not bounded to any particular GPIO, hence the user can use any GPIO to input the clock externally (clk_in). Using the IO Planner in the ForgeFPGA Workshop software, the connections between the internal and external clocks are made. The input & output ports of the design must be assigned to the IOB Block in the IO Planner of the ForgeFPGA Software according to Figure 39. The IOB Input Tile with function as EXT LOOPBACK_CLK0 is assigned to the main loopback_clock signal. On the other hand, the IOB Output Tile with function also as EXT LOOPBACK_CLK0 is assigned to clk_out signal. Hence connecting the loopback_clock signal and clk_out signal internally. See IP Planner below.

See the application note to know more about it.
The Phase Locked Loop (PLL) can be operated via two clock sources, the internal frequency from the oscillator (50 MHz) or the external clock, routed through the GPIO2 pin only. The PLL_CTRL_CLK_SEL input signal is used to select the two clock sources for PLL. When the PLL_CTRL_CLK_SEL signal is LOW, then the clock input to PLL is from the 50 MHz OSC. When the PLL_CTRL_CLK_SEL signal is HIGH, then the clock input to the PLL is from an external clock from GPIO 2. See Figure 40 to understand the connections of PLL.

![Figure 40. Phase Locked Loop signals](image)

See more about how to use PLL through different clock sources in the application note.

3. Oscillator Clock

SLG47910 has a high frequency oscillator on board for use. The default frequency of this oscillator is 50 MHz however it can be changed to a low frequency by pulling the OSC_CTRL_MODE signal low. To enable the Oscillator On-Board, pull the OSC_CTRL_EN signal high. See Figure 41 to understand the structure of the Oscillator.
When using the clock provided by the oscillator, the user needs to specify the clock as the OSC_CLK in the IO Planner along with the OSC_CTRL_EN & OSC_CTRL_MODE signals. See Figure 42 to understand how to write the IP Planner in the ForgeFPGA Workshop.

<table>
<thead>
<tr>
<th>POSITION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOB tile[0, 0] coord[0, 25] Output1</td>
<td>OSC_CTRL_MODE</td>
</tr>
<tr>
<td>IOB tile[0, 0] coord[0, 25] Output0</td>
<td>OSC_CTRL_EN</td>
</tr>
<tr>
<td>CLK tile[0, 0] clk_side=W Input1</td>
<td>PLL_CLK</td>
</tr>
<tr>
<td>CLK tile[0, 0] clk_side=W Input0</td>
<td>OSC_CLK</td>
</tr>
</tbody>
</table>

Figure 42. IO Planner for Oscillator Clock

4. BRAM Clock

We have 2 pairs of BRAM Clock in the North and the South direction of the FPGA Tile. The purpose of these 2 pairs of BRAM clock is to READ and WRITE. The user can specify which BRAM he/she will be using, that is, the BRAMs on the North side or the South side and then specify the clock to those IOB Ports in the IO Planner (see Figure 43).

<table>
<thead>
<tr>
<th>POSITION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK tile[0, 0] clk_side=W Input1</td>
<td>OSC_CLK</td>
</tr>
<tr>
<td>CLK tile[0, 0] clk_side=S Output1</td>
<td>BRAM_SOUTH_WRITE_CLK</td>
</tr>
<tr>
<td>CLK tile[0, 0] clk_side=S Output0</td>
<td>BRAM_SOUTH_READ_CLK</td>
</tr>
<tr>
<td>CLK tile[0, 0] clk_side=N Output1</td>
<td>BRAM_NORTH_WRITE_CLK</td>
</tr>
<tr>
<td>CLK tile[0, 0] clk_side=N Output0</td>
<td>BRAM_NORTH_READ_CLK</td>
</tr>
</tbody>
</table>

Figure 43. BRAM Clock definitions in IO Planner
18. Flowchart

We have created a flowchart to narrow down and follow the basic steps from start to finish. The flowchart shows all the important aspects of using the ForgeFPGA Software.

Figure 44. Software flow
# 19. Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.00</td>
<td>Jan 10, 2023</td>
<td>Updated format&lt;br&gt;Change in software revision, updated document accordingly</td>
</tr>
<tr>
<td>1.00</td>
<td>Mar 3, 2022</td>
<td>Initial release</td>
</tr>
</tbody>
</table>