How to use Lock Bit on ForgeFPGA

SLG47910V

This application shows how to protect the intellectual property by means of locking NVM using the SLG47910V FPGA. This application note explains how to design a pseudo-random number generator (PRNG) and its functionality is verified by creating a Testbench and observing the output waveforms on the Icarus Verilog and GTKWave Software. 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
4. Ingredients 2
5. Verilog code 2
6. Test Bench 3
7. Design Steps 4
8. Conclusion 8
9. Revision History 9

1. Terms and Definitions

CLB	Configuration Logic Block
HDL	Editor Workspace where Verilog code is entered
PRNG	Pseudo-random number generator
LFSR	Linear feedback shift register
FPGA	Field Programmable Gate Array
FPGA Editor	Main FPGA design and simulation window
ForgeFPGA Workshop	FPGA Project window for debug and IO programming

2. References

To find more information about ForgeFPGA™ please visit the website: ForgeFPGA Low-density FPGAs | Renesas

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-014 How to use Lock bit on ForgeFPGA.ffpga, ForgeFPGA Design File

[3] ForgeFPGA SLG47910, Datasheet, Renesas Electronics

3. Introduction

A pseudo-random number generator (PRNG) uses a linear feedback shift register (LFSR) to generate a sequence of numbers approximating the properties of random numbers. The LFSR is designed using DFFs (Figure 1).

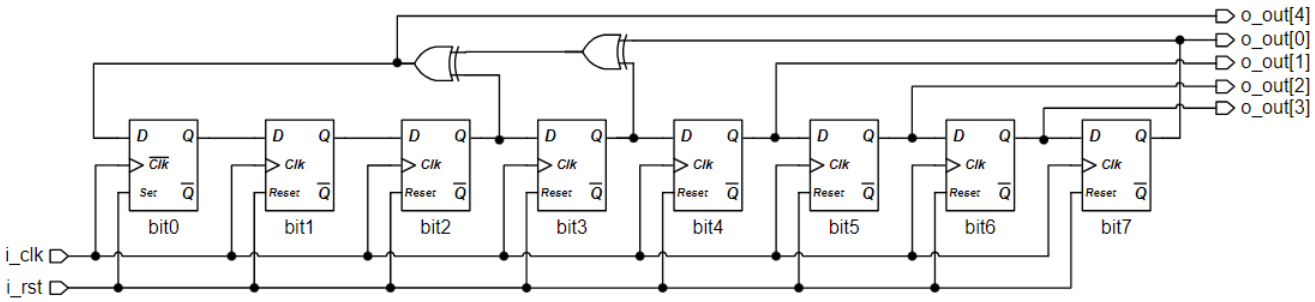


Figure 1: PRNG Circuit

PRNGs are suitable for applications where many random numbers are required and where it is useful that the same sequence can be replayed easily.

4. Ingredients

- Latest Revision of ForgeFPGA Workshop software

5. Verilog Code

Shown below is the top module of pseudo-random generator.

```
module PRNG (
   (* iopad_external_pin, clkbuf_inhibit *) input i_clk,// clock signal
   (* iopad_external_pin *) input          i_rst,        // reset signal
   (* iopad_external_pin *) output [4:0]   o_out,        // 4-bits output of PRNG
   (* iopad_external_pin *) output [4:0]   o_out_oe,     // output enable for outputs
   (* iopad_external_pin *) output         o_osc_en,     // 50MHz oscilator enable
   // PLL configuration
   (* iopad_external_pin *) output [5:0]   o_pll_ctrl_refdiv,
   (* iopad_external_pin *) output [2:0]   o_pll_ctrl_postdiv1,
   (* iopad_external_pin *) output [2:0]   o_pll_ctrl_postdiv2,
   (* iopad_external_pin *) output         o_pll_en,
```

```
  (* iopad_external_pin *) output [11:0] o_pll_fbdiv,
  (* iopad_external_pin *) output        o_pll_ctrl_bypass,
  (* iopad_external_pin *) output        o_pll_ctrl_clk_selection
);

  reg [7:0] r_lfsr;                       // LFSR (Linear Feedback Shift Register)
  wire      w_feedback;                   // Feedback signal of FFSR
  wire      w_rst;                        // global reset

 //OSC configuration
 assign o_osc_en                = 1'b1;
 //PLL configuration
 assign o_pll_ctrl_refdiv       = 6'd5;
 assign o_pll_ctrl_postdiv1     = 3'd7;
 assign o_pll_ctrl_postdiv2     = 3'd7;
 assign o_pll_en                = 1'b1;
 assign o_pll_fbdiv             = 12'd50;
 assign o_pll_ctrl_bypass       = 1'b0;
 assign o_pll_ctrl_clk_selection = 1'b0;
 assign w_rst            = ~i_rst;
 assign o_out_oe = 5'b1_1111;

  always @(posedge i_clk) begin
    if (w_rst) begin
        r_lfsr <= 8'b0000_0001;               // Reset to init state
    end else begin
        r_lfsr <= {r_lfsr[6:0], w_feedback};  // Cyclical shift right
    end
  end

  assign w_feedback = (r_lfsr[7] ^ r_lfsr[3] ^ r_lfsr[2]);
  assign o_out      = {w_feedback,r_lfsr[7:4]};

endmodule
```

6. Test Bench

A Test Bench is used to provide stimulus to verify the correctness or working of our DUT, hence we should instantiate our design module. The module name must be given as to be recognized by the ForgeFPGA Workshop as testbench file (PRNG_tb)

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

```
`timescale 1ns / 1ps

module PRNG_tb();
  reg i_clk;
  reg i_rst;
  wire [4:0] o_out;
PRNG dut (
    .i_clk ( i_clk ),
    .i_rst ( i_rst ),
    .o_out ( o_out )
);

// Generating clock signal
  always #10 i_clk = ~i_clk; // Period 20 ns
```

```
initial begin
    //$dumpfile ("PRNG_tb.vcd");
    //$dumpvars (0, PRNG_tb);
    i_clk = 0;
    i_rst = 0;
    #20
    i_rst = 1;

    #2000
    $finish;
end

always @(posedge i_clk) begin
    if (i_rst) begin
        $display("Time: %t, Random Number: %d", $time, o_out);
    end
end

endmodule
```

7. 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. Download the design example AN-FG-014 How to use Lock Bit on ForgeFPGA.ffpga. If you are not familiar with the ForgeFPGA Workshop software, review the Four-Bit Counter application notes that covers the basic design steps.
3. Open the AN-FG-014 How to use Lock Bit on ForgeFPGA.ffpga file after downloading.
4. Open the FPGA editor and review the Verilog code. There is a main code with the module name PRNG, which is the top module defining the whole design. PLL_CLK is used to provide clock to the design.

5. Open the IO planner tab on the FPGA editor and review the pin assignment Figure 2

FUNCTION	DIRECTION	PORT
PLL_CLK	Input	i_clk
FPGA_CORE_READY	Input	i_rst
OSC_EN	Output	o_osc_en
[PIN 13] GPIO0_OUT	Output	o_out[0]
[PIN 14] GPIO1_OUT	Output	o_out[1]
[PIN 15] GPIO2_OUT	Output	o_out[2]
[PIN 16] GPIO3_OUT	Output	o_out[3]
[PIN 17] GPIO4_OUT	Output	o_out[4]
[PIN 13] GPIO0_OE	Output	o_out_oe[0]
[PIN 14] GPIO1_OE	Output	o_out_oe[1]
[PIN 15] GPIO2_OE	Output	o_out_oe[2]
[PIN 16] GPIO3_OE	Output	o_out_oe[3]
[PIN 17] GPIO4_OE	Output	o_out_oe[4]
PLL_BYPASS	Output	o_pll_ctrl_bypass
PLL_REF_CLK_SEL	Output	o_pll_ctrl_clk_selection
PLL_POSTDIV1[0]	Output	o_pll_ctrl_postdiv1[0]
PLL_POSTDIV1[1]	Output	o_pll_ctrl_postdiv1[1]
PLL_POSTDIV1[2]	Output	o_pll_ctrl_postdiv1[2]
PLL_POSTDIV2[0]	Output	o_pll_ctrl_postdiv2[0]
PLL_POSTDIV2[1]	Output	o_pll_ctrl_postdiv2[1]
PLL_POSTDIV2[2]	Output	o_pll_ctrl_postdiv2[2]
PLL_REFDIV[0]	Output	o_pll_ctrl_refdiv[0]

Figure 2: IO Planner

6. Next select the Synthesize button on the lower left side of the FPGA editor. 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 steam was generated correctly.
7. Now click on the Floorplan tab and see the CLB utilization. Press the Ctrl and the mouse wheel to zoom-in.

8. User can test the functionality of the design by running the GTKWave and using the Testbench to provide Stimulus. See logger to observe the Randomly Generated Numbers Figure 3

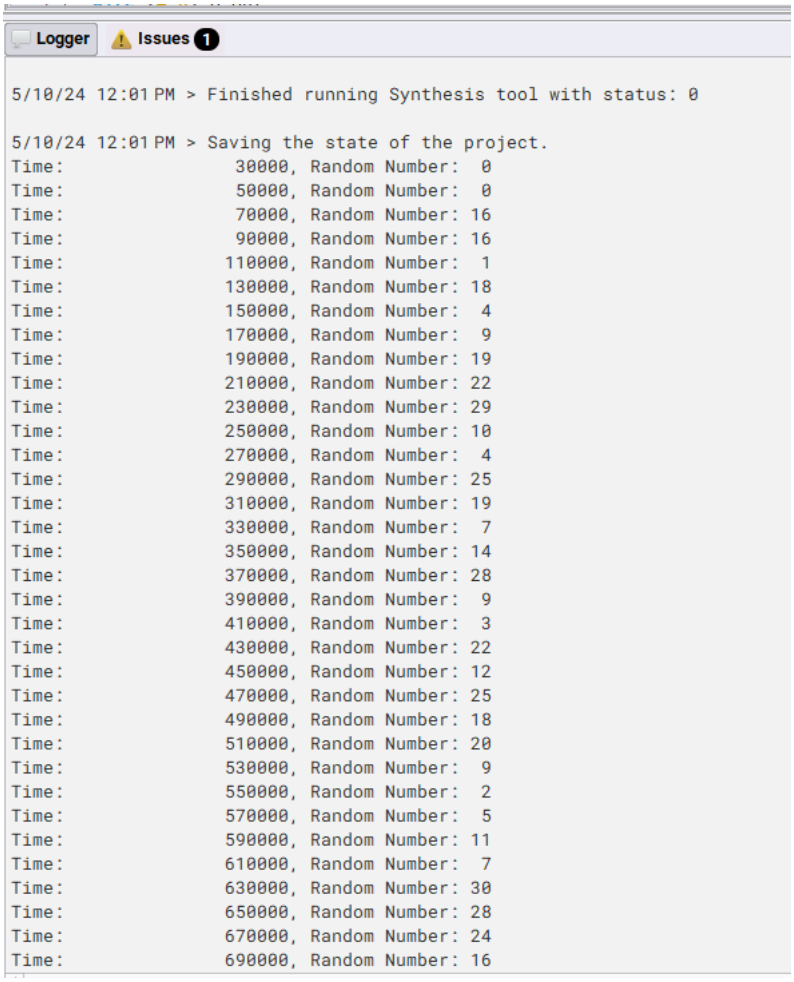
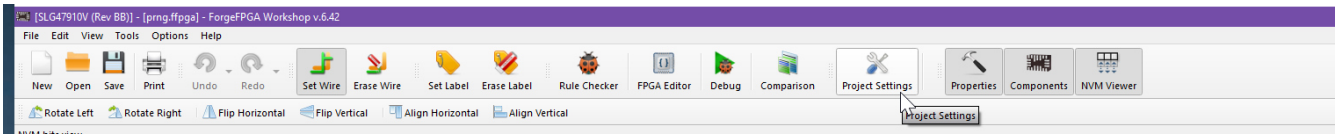


Figure 3: Randomly Generated Numbers

9. After the user is satisfied with the results, the user can go ahead program the part. Before programming the SLG47910V device, user can protect the design by choosing one of the four options found in the Project Settings window. Under it, user needs to open the Security tab and choose one option. Let's choose the first option called "Unlocked" (See Figure 4).



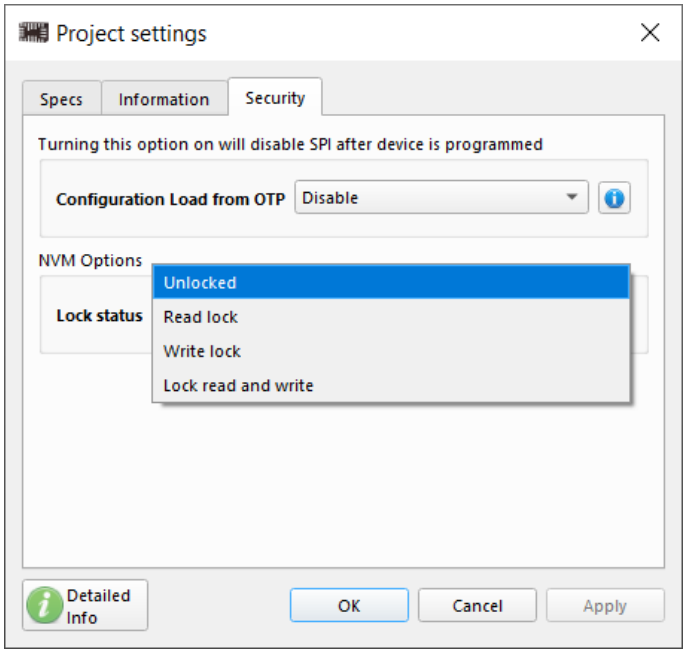


Figure 4. Project Settings- Lock Status

3. After programing the NVM, user has possibility to read NVM data by clicking “Read” button on Debug section (Figure 5).

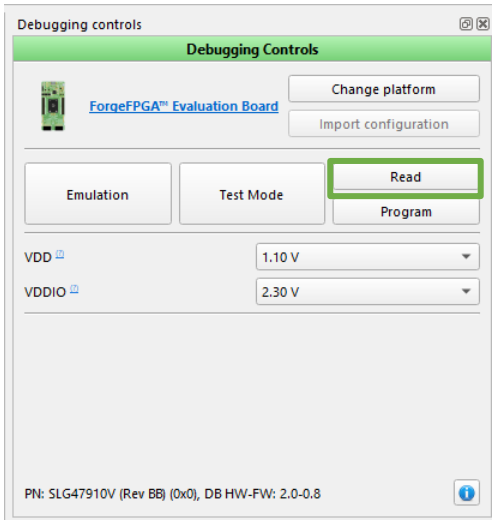


Figure 5. Reading the Bitstream

4. The ForgeFPGA Workshop shows the notification that Chip was successfully read (Figure 6). A user has possibility to see the data from NVM by clicking the “Yes” button.

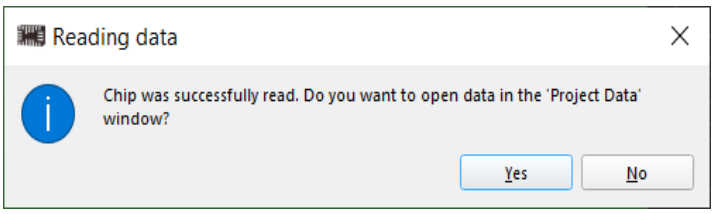


Figure 6. Read Data

5. Figure 7 presents data that was read from the NVM and the user can export the data and program another SLG47910V device with the same project.

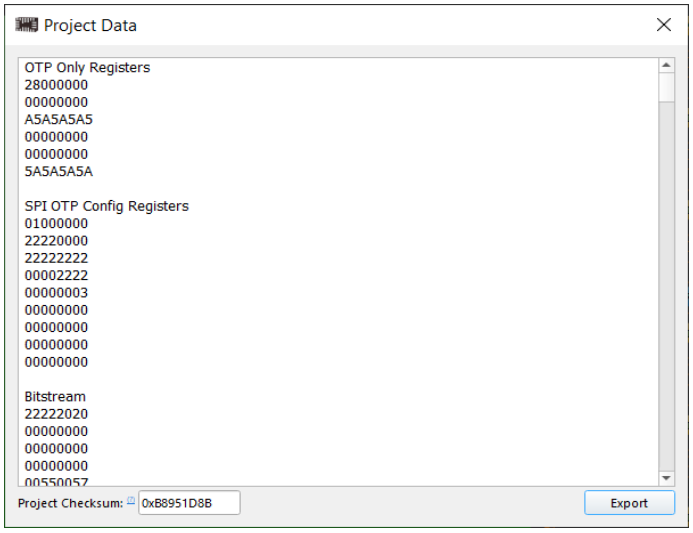


Figure 7. Project Data

6.If user choose the "**Read Lock**" or "**Lock read and write**" in the security tab (Figure 4) before programing the part, and then the user tries to read the NVM content from the device after programing it. The user will see the notification that he does not have possibility to read the data by ForgeFPGA software or other methods as the OTP is locked (Figure 8)

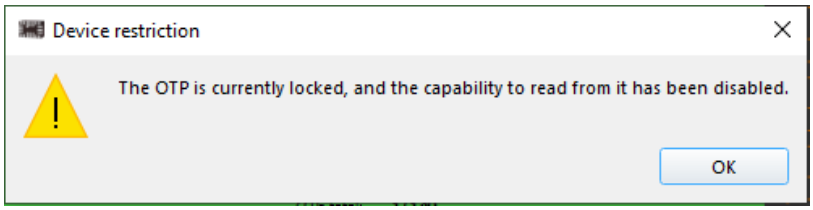


Figure 8. Device Restriction

7. If user choose "Write lock" or "Lock read and write" in the security tab (Figure 4) before programing the part. It loses the ability to program bits with "HIGH" value into NVM.

8. Conclusion

This application note shows how the lock status allows the locking of NVM code in the programmed chip, preventing a subsequent read/write of the chip's NVM sequence and which sequence the user should use in the ForgeFPGA Workshop software to lock read/write NVM. Once a design is read locked, failure analysis will be very limited.

9. Revision History

Revision	Date	Description
1.00	May 10, 2023	Initial Release with 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)

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-us/.

Trademarks

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