Shift Register (SRL)

SLG47921V

This Application shows how to use shift register with SLG47920/21. Simulation waveforms generated by GTKWave software and Logic Analyzer in FPGA workshop can be used to verify the functionality of the design.

This document is supported by design files listed in the Reference section.

Contents

1. Introduction

This document shows how to use the SLG47920/21's configurable logic to design a shift register. See the SLG47921 datasheet section 5.3.2.1 Configurable Logic Block for Memory Used in Shift Register Mode (Figure 1) for more information. When using the Configurable Logic Blocks for Memory in Shift Register Mode (SRM), there is a selectable choice in terms of the width of the shift registers that are implemented. The three choices are to have four independent 16-bit shift registers (SRL16E), two independent 32-bit shift registers (SRL32E) or one independent 64-bit shift register (SRL64E). This configuration is a part of the internal 5 kb distributed memory.

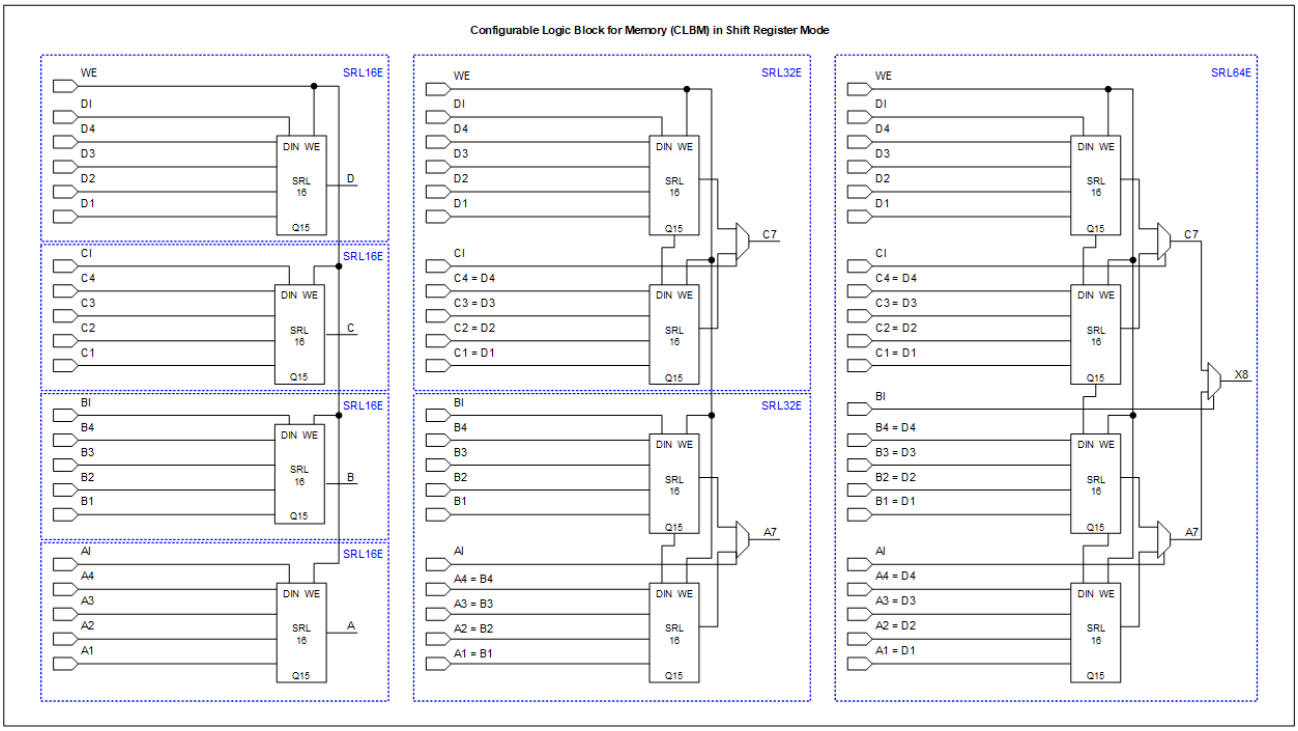

Figure 1. Configurable Logic Blocks for Memory used in Shift Register Mode

Configurable Logic Blocks (CLBs) are the fundamental building blocks for implementing custom digital logic in an FPGA. While primarily used for constructing combinational and sequential circuits, the Look-Up Tables (LUTs) within CLBs can also be configured as an addressable shift register (See Figure 2) for data delay or serial shifting operations. This makes them an efficient alternative to traditional flip-flop chains, particularly when minimal latency and highly flexible data paths are required. Shift registers built from CLBs are well-suited for applications involving high-speed data streams, lightweight filtering, or configurable-depth pipelining, offering an optimized solution for small-scale, high-performance sequential data processing in FPGA designs.

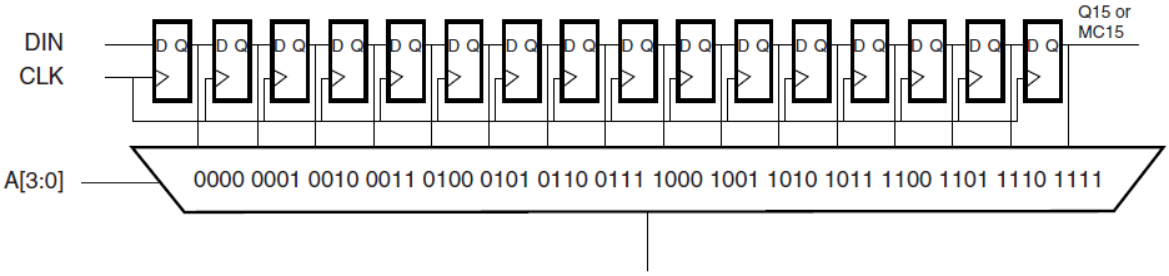

Figure 2. LUT Configured as an Addressable Shift Register

Using the ForgeFPGA Workshop software, the Verilog code was synthesized, and the bit stream was loaded on to the SLG47921V device. The following functional waveforms show the simulation results for different bits shift register and the dynamic shift register output.

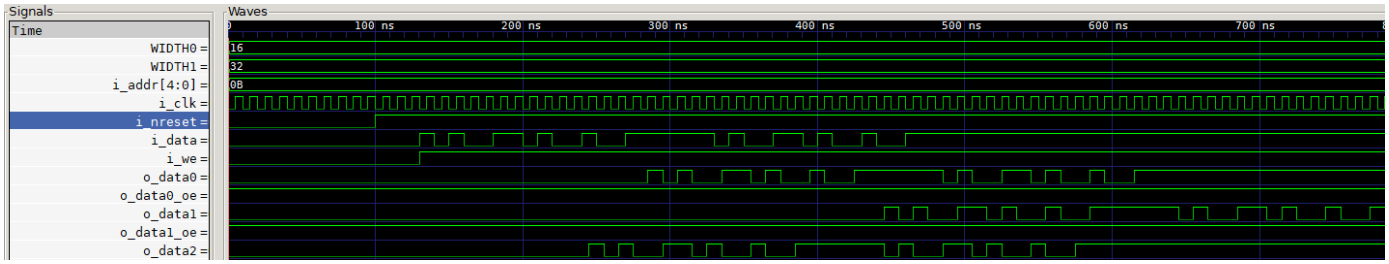

Figure 3. Functional Waveform in GTKWave

2. Ingredients

- SLG47920/21 Device
- ForgeFPGA Deluxe Development Board and power cables
- ForgeFPGA Socket Adaptor Board
- Latest Revision of the ForgeFPGA Workshop software

3. Shift Register Verilog Code

The Shift Register design is available for download (AN-FG-023 Shift Register (SRL).ffpga). It contains the complete Shift Register design using the LUT within the FPGA Core of the SLG47921V device. There are two different types of shift registers, the static shift registers and dynamic shift registers.

Shown below is the (*top*) module named shift_registers. It is available for downloading (AN-FG-023 Shift Register (SRL).ffpga)

```
(*top*) module shift_registers (
  (* iopad_external_pin, clkbuf_inhibit *) input i_clk,
  (* iopad_external_pin *) input            i_nreset,
  (* iopad_external_pin *) input            i_data,
  (* iopad_external_pin *) input            i_we,
  (* iopad_external_pin *) input      [4:0] i_addr,
  (* iopad_external_pin *) output           o_data0,
  (* iopad_external_pin *) output           o_data1,
  (* iopad_external_pin *) output           o_data2,

  (* iopad_external_pin *) output           o_data0_oe,
  (* iopad_external_pin *) output           o_data1_oe,
  (* iopad_external_pin *) output           o_data2_oe,

  (* iopad_external_pin *) output           o_osc_ctrl_en,
  (* iopad_external_pin *) output           o_osc_ctrl_mode,
  (* iopad_external_pin *) output           o_postdiv0_ctrl_en,
  (* iopad_external_pin *) output     [2:0] o_postdiv0_ctrl_sel
);

  localparam WIDTH0 = 16;
  localparam WIDTH1 = 32;

  reg [WIDTH0-1:0] shift_reg0 = 0;
  reg [WIDTH0-1:0] shift_reg1 = 0;
  reg [WIDTH1-1:0] shift_reg2 = 0;

  assign o_data0_oe = 1'b1;

  assign o_data1_oe = 1'b1;

  assign o_data2_oe = 1'b1;


  assign o_osc_ctrl_en = 1'b1;

  assign o_osc_ctrl_mode = 1'b1;

  assign o_postdiv0_ctrl_en = 1'b1;

  assign o_postdiv0_ctrl_sel = 3'b000;

  //Shift register 16 bits static

  always @(posedge i_clk) begin

    if(i_we)

      shift_reg0 <= {shift_reg0[WIDTH0-2:0],i_data};

  end

  assign o_data0 = shift_reg0[WIDTH0-1];

  //Shift register 32 bits static

  always @(posedge i_clk) begin

    if(i_we)

      shift_reg1 <= {shift_reg1[WIDTH0-2:0],o_data0};

  end

  assign o_data1 = shift_reg1[WIDTH0-1];
```

```
    //Shift register 32 bits Dynamic
    always @(posedge i_clk) begin
        if(i_we)
            shift_reg2 <= {shift_reg2[WIDTH1-2:0],i_data};
    end
    assign o_data2 = shift_reg2[i_addr];
endmodule
```

4. Floorplan: CLB Utilization

From the Floorplan window, we can see a part of the floorplan (Figure 4) for this document. Also, on the left corner of the figure, the user can see a mini resource needed list. From the Verilog code the user can observe that two 16-bit shift registers have been used to implement a 16-stage static shift register and a 32-stage static shift register, one 32-bit shift register has been used to implement a dynamic shift register. This can also be observed under the floorplan tab in the software (Figure 5).

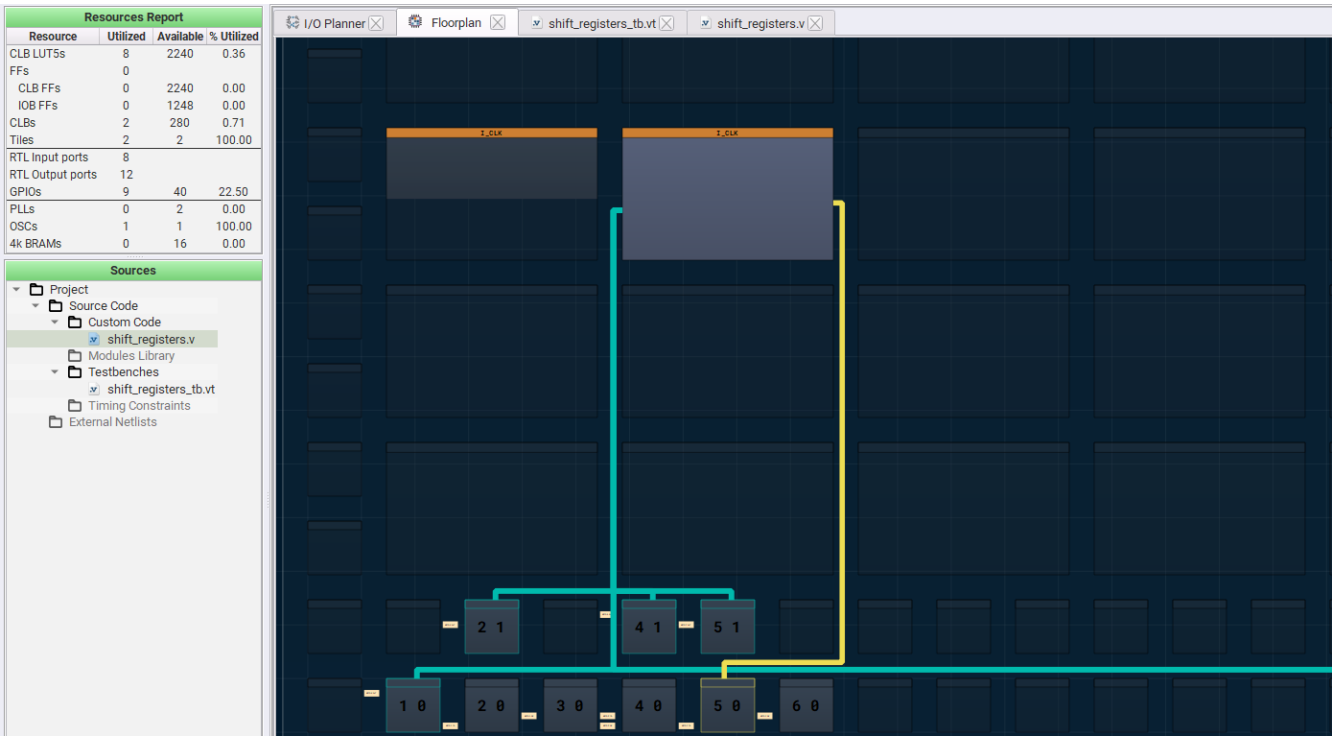

Figure 4. Distributed RAM CLB Utilization

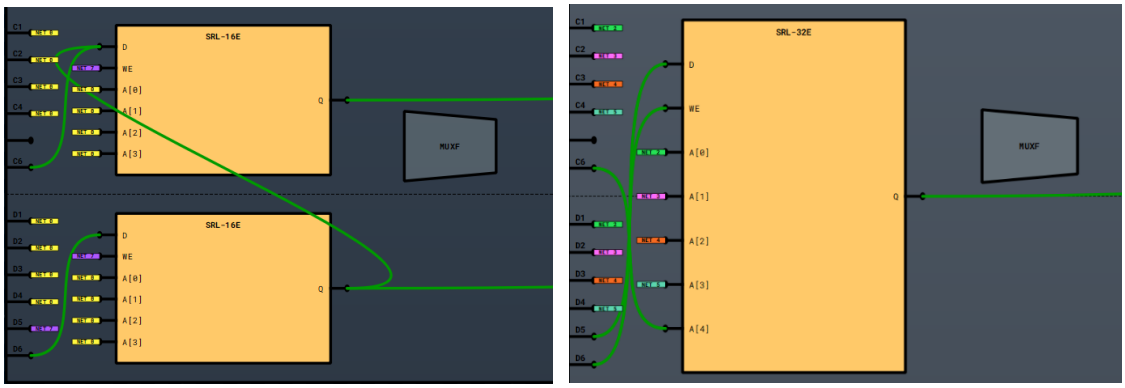

Figure 5. SRLs shown in CLB

5. Logic Analyzer: RAM Data Write and Read Out

The Logic Analyzer in the FPGA Editor shows the i_data, i_we, i_addr and different Shift Registers outputs (See Figure 6). We can set the GPIO0 as i_data, GPIO1 as i_we , GPIO2 as o_data0, GPIO3 as o_data1, GPIO4 as o_data2. GPIO5~GPIO10 as i_addr for the dynamic shift register address. Set the i_addr parallel buses to a constant 0x0B, so that we can see the data feed into shift registers and data output from shift registers in Logic Analyzer.

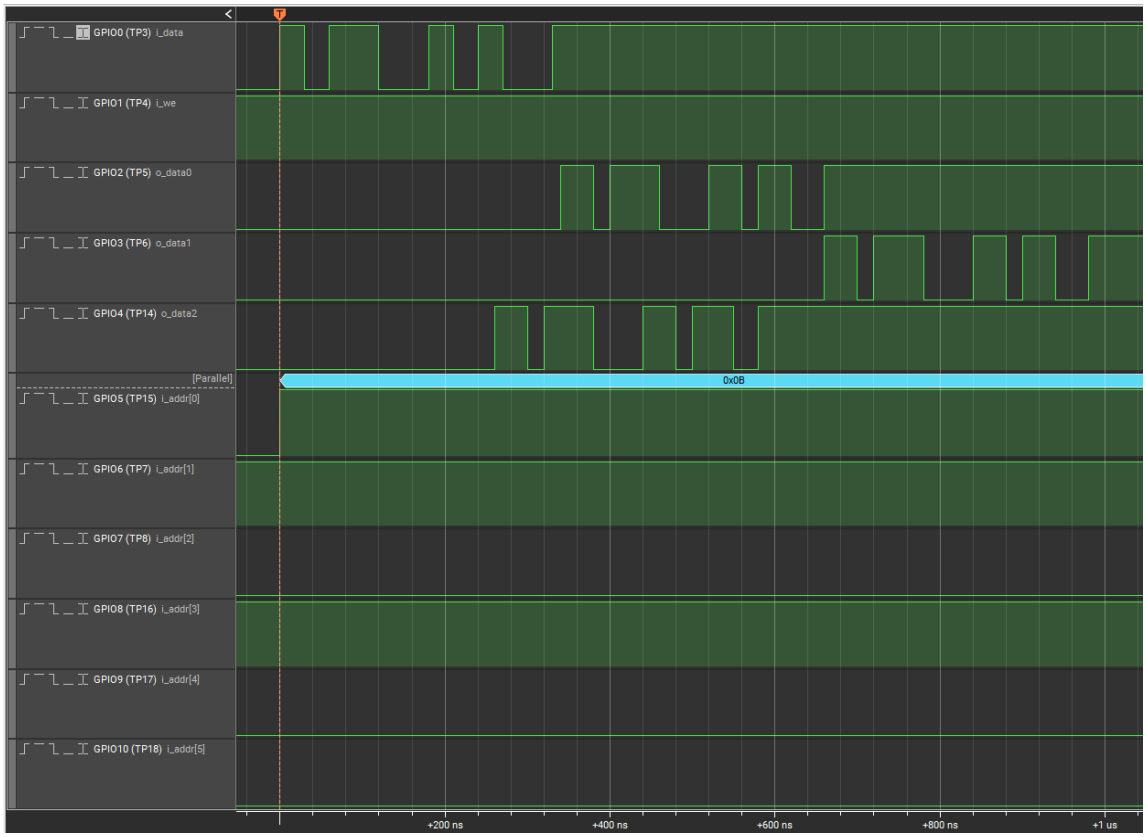

Figure 6. Shift Registers Input and Output Data Waveform

6. Design Steps

1. Launch the latest version of the Go Configure Software Hub. Select SLG47921V device and the ForgeFPGA Workshop software will load.
2. Download the design AN-FG-023 Shift Register (SRL).ffpga. If you are not familiar with the ForgeFPGA Workshop software, review the Shift Register (SRL) application notes that cover the basic design steps.
3. Open the AN-FG-023 Shift Register (SRL).ffpga file after downloading.
4. Open the FPGA editor and review the Verilog code and the testbench code. Change the Width and Depth of RAM you would like to use and define the value in initial block.
5. Open the IO planner tab on the FPGA editor and review the pin assignment.
6. 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 bitstream was generated correctly.

7. Click on the Floorplan tab and see the CLB utilization. Press the Ctrl and the mouse wheel to zoom-in (Figure 4). Confirm that the IOs in the IO Planner (Figure 7).

FUNCTION	DIRECTION	PORT	DESCRIPTION
GPIO5_IN [PIN 7]	Input	i_addr[0]	GPIO5 Input Value
GPIO6_IN [PIN 8]	Input	i_addr[1]	GPIO6 Input Value
GPIO7_IN [PIN 9]	Input	i_addr[2]	GPIO7 Input Value
GPIO8_IN [PIN 16]	Input	i_addr[3]	GPIO8 Input Value
GPIO9_IN [PIN 17]	Input	i_addr[4]	GPIO9 Input Value
GPIO10_IN [PIN 18]	Input	i_addr[5]	GPIO10 Input Value
GLOBAL_CLK_MUX_OUT[0]	Input	i_clk	Tile00 Clock Input 0 from Global Clock Network
GPIO0_IN [PIN 2]	Input	i_data	GPIO0 Input Value
FPGA_CONFIG_RDY	Input	i_nreset	FPGA Configuration Ready signal
GPIO1_IN [PIN 3]	Input	i_we	GPIO1 Input Value
GPIO37_OUT [PIN 38]	Output	o_clk	GPIO37 Output Value
GPIO37_OE [PIN 38]	Output	o_clk_oe	GPIO37 Output Enable
GPIO2_OUT [PIN 4]	Output	o_data0	GPIO2 Output Value
GPIO2_OE [PIN 4]	Output	o_data0_oe	GPIO2 Output Enable
GPIO3_OUT [PIN 5]	Output	o_data1	GPIO3 Output Value
GPIO3_OE [PIN 5]	Output	o_data1_oe	GPIO3 Output Enable
GPIO4_OUT [PIN 6]	Output	o_data2	GPIO4 Output Value
GPIO4_OE [PIN 6]	Output	o_data2_oe	GPIO4 Output Enable
OSC_EN	Output	o_osc_ctrl_en	Oscillator Enable signal
OSC_MODE	Output	o_osc_ctrl_mode	Oscillator Mode Select signal
POSTDIV_OUT0_EN	Output	o_postdiv0_ctrl_en	Oscillator Postdivider Out0 Enable signal
POSTDIV_OUT0_DIV_SEL[0]	Output	o_postdiv0_ctrl_sel[0]	Oscillator Postdivider Out0 divider select (bit 0)
POSTDIV_OUT0_DIV_SEL[1]	Output	o_postdiv0_ctrl_sel[1]	Oscillator Postdivider Out0 divider select (bit 1)
POSTDIV_OUT0_DIV_SEL[2]	Output	o_postdiv0_ctrl_sel[2]	Oscillator Postdivider Out0 divider select (bit 2)

Figure 7. IO Planner

8. Click on the Simulation button at the upper top then select “Run RTL Simulation”. The GTKWave will automatically open if there are no Syntax errors in the testbench. Check logger for errors.
9. In the GTKWave software, select the signals you want to view, and Click Insert on the left corner to insert the signals in the wave window. Once the desired signals are selected, click on Reload (See Figure 3).
10. Observe the RAM data write and read out in the waveform displayed in the GTKWave software. The same results can also be observed in the Logic Analyzer of the ForgeFPGA Workshop software.

7. Conclusions

This application note shows how SLG47921 can be used to implement a shift register, this design file is available for download (AN-FG-023 Shift Register (SRL).ffpga). For more information, contact the ForgeFPGA Business Support Team.

8. Terms and Definitions

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

9. References

For related documents and software, please visit ForgeFPGA Download our free ForgeFPGA™ Designer software [1] to open the. ffpga design files [2] and view the proposed circuit design.

[1] Go Configure Software Hub, Software Download and User Guide, Renesas Electronics

[2] AN-FG-023 Shift Register (SRL).ffpga, ForgeFPGA Design File

[3] ForgeFPGA SLG47920/SLG47921, Datasheet, Renesas Electronics

[4] ForgeFPGA Workshop User Guide, Renesas Electronics

10. Revision History

Revision	Date	Description
1.00	Jan 06,2026	Initial release.