How to Use Distributed Memory as a RAM

SLG47910V

This Application shows how to design a RAM using distributed memory resources in SLG47910. Simulation waveforms generated by GTKWave software and Logic Analyzer in FPGA workshop can be used to verify the functionality of the design.

This application note comes complete with design files which can be found in the Reference section.

Contents

1. 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

2. References

For related documents and software, please visit: ForgeFPGA Low-density FPGAs | Renesas

Download our free Go Configure Software Hub [1] to open the. ffpga design files [2] and view the proposed circuit design.

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

[2] AN-FG-018 How to use distributed memory as a RAM.ffpga, ForgeFPGA Design File

[3] ForgeFPGA SLG47910, Datasheet, Renesas Electronics

3. Introduction

This application shows how to use the SLG47910's distributed memory to design a RAM, this RAM is a simple dual port RAM, the RAM block is shown in Figure 1. In such cases users can define the width and depth of the RAM and define the RAM operation mode when read and write of the same address occur.

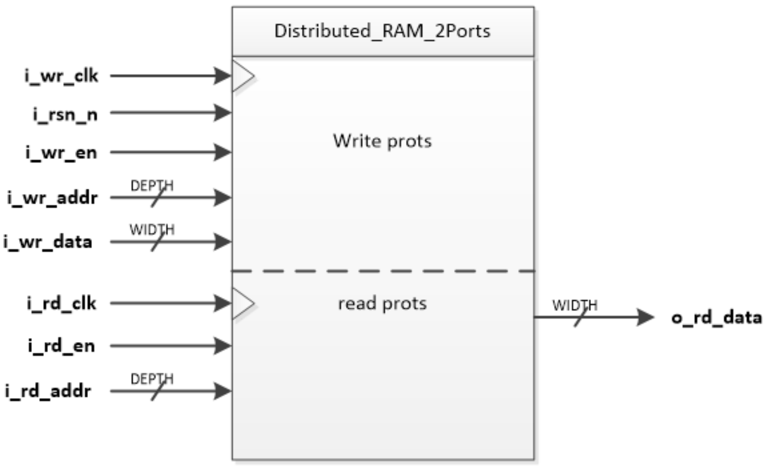

Figure 1: Simple Dual Port RAM

The following signal names are the PINs that are used in the design.

- i_wr_clk – input write clock
- i_rst_n – input negative reset signal
- i_wr_en – input write enable signal
- i_wr_addr – input write address signal
- i_wr_addr – input write data signal
- i_rd_clk – input read clock
- i_rd_en – input read enable signal
- i_rd_addr – input read address signal
- i_rd_data – input read data signal

Using the ForgeFPGA Workshop software, the Verilog code was synthesized, and the bit stream was loaded on to the SLG47910V device. The functional waveforms below (see Figure 2) show the data write to address 0~7 and the read-out data from address 0 to 7.

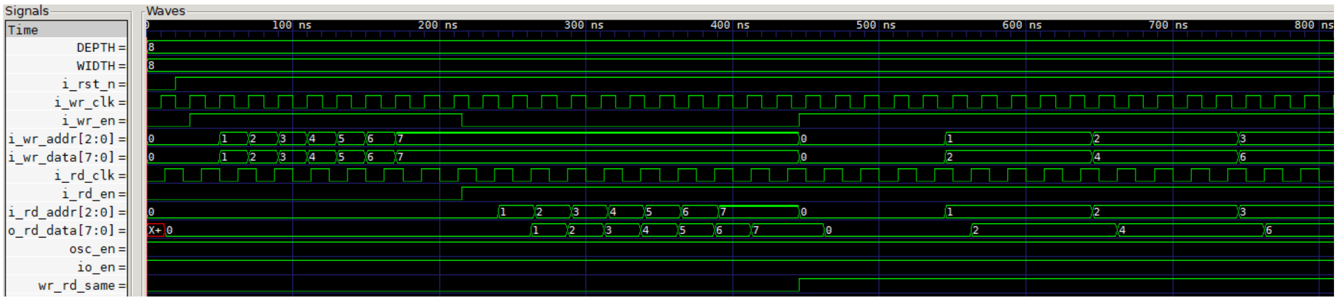

Figure 2: Functional Waveform in GTKWave

4. RAM Description

Distributed memory in an FPGA is implemented using LUTs, the fundamental building blocks of logic in the FPGA fabric. While primarily used for combinational logic, LUTs can also be configured as small RAM blocks for temporary data storage. This makes them a valuable alternative to Block RAM when high-speed, low-latency memory is required. Distributed memory is particularly well-suited for applications that demand rapid access, high parallelism, and flexible memory structures, providing an efficient solution for small-scale storage needs in FPGA designs.

5. Distributed RAM Verilog Code

The Distributed RAM design is available for download ([AN-FG-018 How to use distributed memory as a RAM.ffpga]). It contains the complete distributed RAM design using the LUT module and the FPGA Core of the SLG47910V device.

Shown below is the (*top*) module named dist_ram_2ports.

```
(* top *) module dist_ram_2ports
#(
    parameter DEPTH = 4,   // The depth of RAM, means how many data could be stored in RAM
    parameter WIDTH = 4,   // The port data with for the RAM
    parameter OPRA_MODE = "WRITE_FIRST" // operation mode when read and write the same address occur
                                        // could be "WRITE_FIRST" or "READ_FIRST"
)
(
  (* iopad_external_pin, clkbuf_inhibit *) input          i_wr_clk,
  (* iopad_external_pin, clkbuf_inhibit *) input          i_rd_clk,
  (* iopad_external_pin *)   input                        i_rst_n,
  (* iopad_external_pin *)   output                       osc_en,
```

```
(* iopad_external_pin *)  input                        i_wr_en,
(* iopad_external_pin *)  input [$clog2(DEPTH)-1:0]    i_wr_addr,
(* iopad_external_pin *)  output reg [WIDTH-1:0]       i_wr_data,

(* iopad_external_pin *)  input                        i_rd_en,
(* iopad_external_pin *)  input [$clog2(DEPTH)-1:0]    i_rd_addr,
(* iopad_external_pin *)  output reg [WIDTH-1:0]       o_rd_data,
(* iopad_external_pin *)  output                       io_en

(* iopad_external_pin *)  output                       pll_en,
(* iopad_external_pin *)  output  [5:0]                pll_refdiv,
(* iopad_external_pin *)  output  [11:0]               pll_fbdiv,
(* iopad_external_pin *)  output  [2:0]                pll_postdiv1,
(* iopad_external_pin *)  output  [2:0]                pll_postdiv2,
(* iopad_external_pin *)  output                       pll_bypass,
(* iopad_external_pin *)  output                       pll_clk_selection
);
  assign osc_en = 1'b1;
  assign io_en = 1'b1;

  assign pll_en = 1'b1;
  assign pll_refdiv = 6'b00_0001;             // Equivalent value in decimal form 6'd1,
  assign pll_fbdiv = 12'b0000_0001_1000;      // Equivalent value in decimal form 12'd24,
  assign pll_postdiv1 = 3'b110;               // Equivalent value in decimal form 3'd6,
  assign pll_postdiv2 = 3'b101;               // Equivalent value in decimal form 3'd5,
  assign pll_bypass = 1'b0;
  assign pll_clk_selection = 1'b0;

reg [WIDTH-1:0] mem_ram [DEPTH-1:0];

// write data to RAM
always @(posedge i_wr_clk) begin
  if(i_wr_en)
        mem_ram[i_wr_addr] <= i_wr_data;
end
// define a read and write operations occur simultaneously
```

```
  wire wr_rd_same;
  assign wr_rd_same = (i_wr_en&&i_rd_en&&(i_wr_addr == i_rd_addr))?  1'd1 : 1'd0;

// rade data from RAM
always @(posedge i_rd_clk) begin
  if(!i_rst_n)
                o_rd_data <= 'h0;
  else if(i_rd_en) begin
`ifdef  OPRA_MODE == "WRITE_FIRST"
        if(wr_rd_same)
                o_rd_data <= i_wr_data;
`else
        if(wr_rd_same)
                o_rd_data <= mem_ram[i_rd_addr];
`endif
  else
                o_rd_data <= mem_ram[i_rd_addr];
  end
end
endmodule
```

Note:

$clog2() is a Verilog function which returns the ceiling of the logarithm to base 2. In cases where the answer is a decimal number; the function rounds it to the next integer value. The argument can be an integer or an arbitrary sized vector value. The argument shall be treated as an unsigned value, and an argument value of 0 shall produce a result of 0.

Example:

$clog2(8) = 3

$clog2(15) = 3.907 ≈ 4

$clog2(4095) = 12

Hence, with the help of $clog2() function we can determine the maximum number of bits needed for the input read address.

6. Floorplan: CLB Utilization

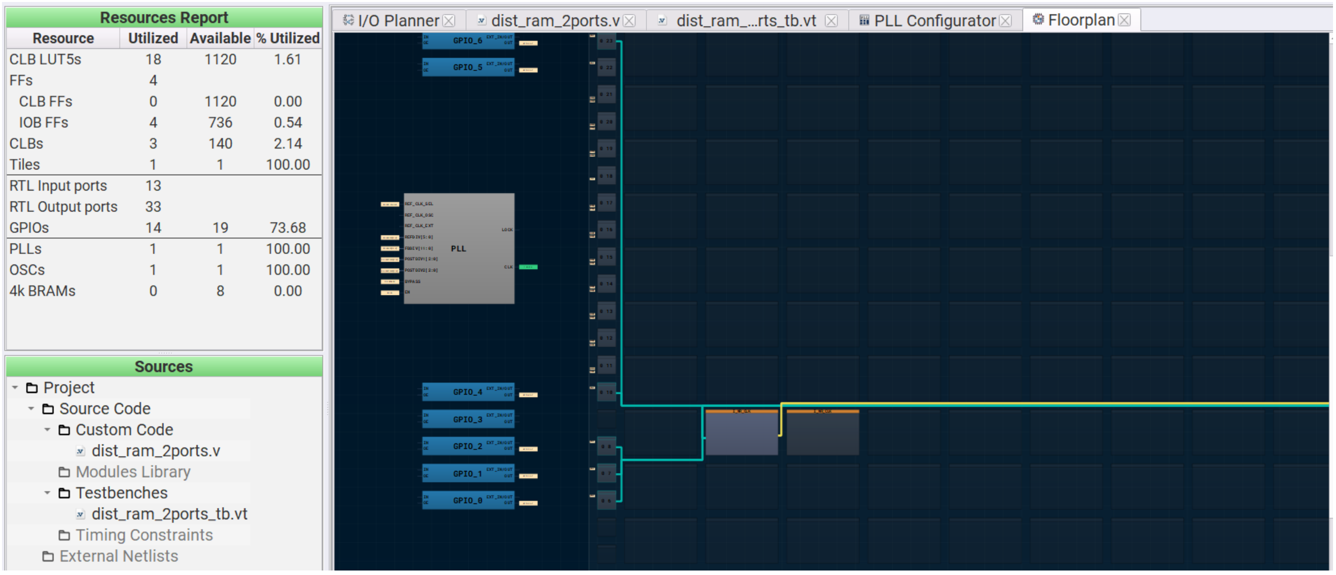

Figure 3: Distributed RAM CLB Utilization

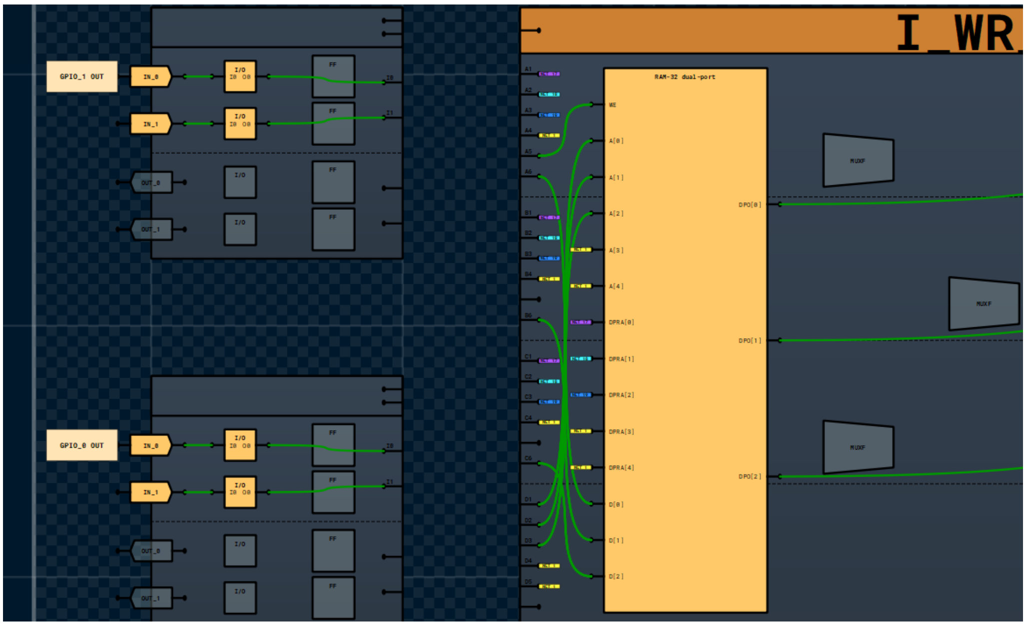

Figure 4: RAM-32 Dual-port Shown in CLB

The Floor planner tab in the FPGA Editor shows the placement of CLBs and FFs (Figure 3). The resource utilization is shown in the top left corner. And we can also zoom in the CLB to check the LUTs in it are used as a RAM-32 dual-port (Figure 4).

7. Logic Analyzer: RAM Data Write and Read Out

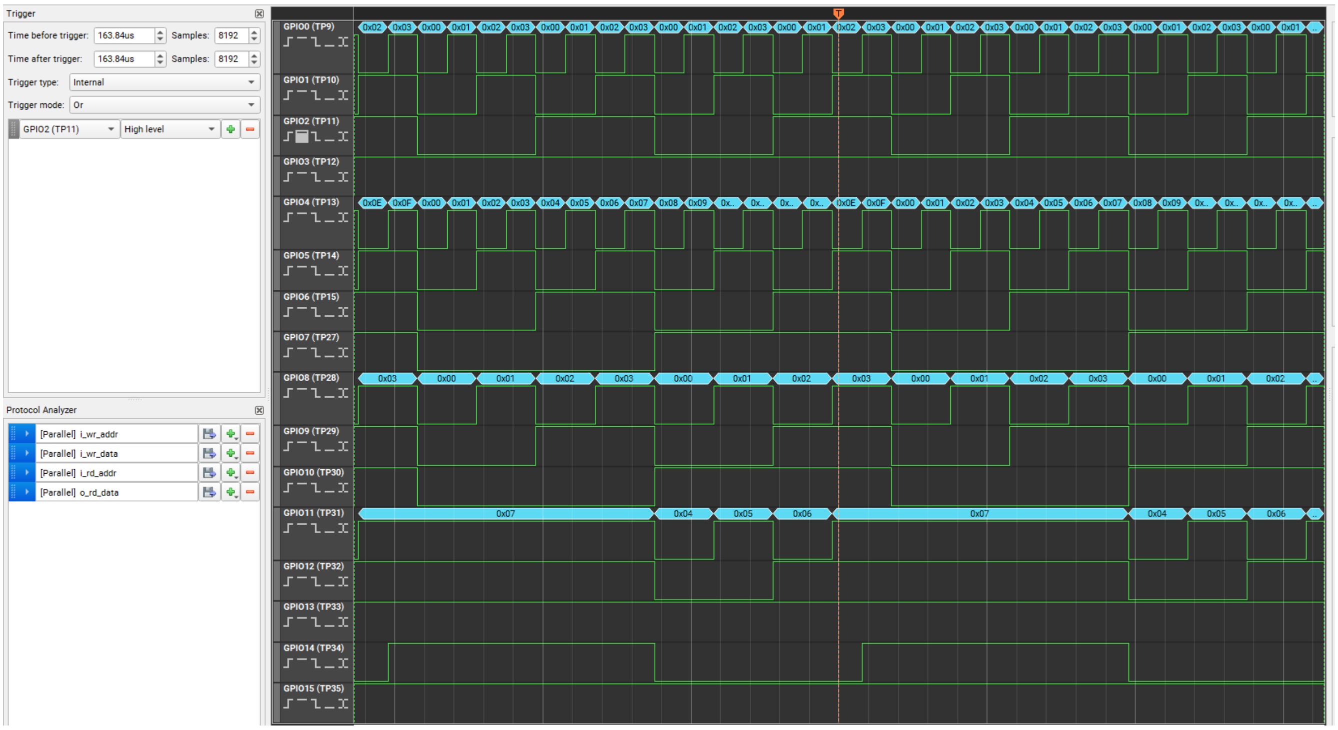

Figure 5: Distributed RAM Data Write and Read Out

The Logic Analyzer in the FPGA Editor shows the RAM data write and read out (Figure 5). We can set the GPIO0, GPIO1 as RAM write address and use Synchronous Logic generator, to make sure the address input comes from 0~3, and GPIO2 as write enable, also use Synchronous Logic generator, to make sure when we set address from 0~3, we can see the write enable asserted; We can set the GPIO4, GPIO5,GPIO6,GPIO7 as RAM write data and use Synchronous Logic generator, to make sure the write data comes from 0~15; We can set the GPIO8, GPIO9 as RAM read address and use Synchronous Logic generator, to make sure the address input comes from 0~3, and GPIO10 as read enable, in the Logic Analyzer we can set the GPIO11, GPIO12, GPIO13, GPIO14 as the RAM data output. We set all those as respective parallel buses, so that we can see the data write into RAM and the data read out from RAM in Logic Analyzer.

8. Design Steps

1. Launch the latest version of the Go Configure Software Hub. Select SLG47910V device and the ForgeFPGA Workshop software will load.
2. Download the design example How to use distributed memory as a RAM.ffpga. If you are not familiar with the ForgeFPGA Workshop software, review the How to use Distributed memory as a RAM application notes that cover the basic design steps.
3. Open the How to use distributed memory as a RAM.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. 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 bitstream was generated correctly.

7. Now click on the Floorplan tab and see the CLB utilization (Figure 3). Press the Ctrl and the mouse wheel to zoom-in. Confirm that the IOs selected in the IO Planner are shown in the floorplan.
8. Now click on the Simulate Testbench button at the upper top. 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 (Figure 2).
10. You can 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.

9. Conclusions

This application note shows how the SLG47910 can be used to implement a RAM, and the RAM width and depth could be modified by users according to the project need, the input read address can be determined using $clog2() function accurately. This testcase is available for download (AN-FG-018 How to use distributed memory as a RAM.ffpga). If interested, please contact the ForgeFPGA Business Support Team.

10. Revision History

Revision	Date	Description
1.00	Mar 24, 2025	Initial release.

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-data version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.