8x4 FIFO Using BRAM
SLG47910

This application shows how to design 8x4 FIFO using the onboard Block RAM(BRAM). Simulation waveforms generated by GTKWave software 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

Terms and Definitions 1

References 1

1. **Introduction** 2
2. **FIFO Description** 2
3. **Ingredients** 4
4. **FIFO Verilog Code** 4
5. **Floorplan : CLB Utilization** 5
6. **Design Steps** 6
7. **Conclusion** 8
8. **Revision History** 9

Terms and Definitions

FPGA	Field Programmable Gate Array
FIFO	First-In-First-Out Memory
BRAM	Block RAM
FPGA Core	Circuit Block that contains the digital array macro cells
ForgeFPGA Workshop	Top level FPGA display and control window
FPGA Editor	Main FPGA design and simulation window
CLB	Configuration Logic Block

References

For related documents and software, please visit:

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-013 BRAM FIFO Design .ffpga, ForgeFPGA Design File

[3] SLG47910, Preliminary Datasheet

1. Introduction

This application shows how to use the SLG47910's embedded Block RAM (BRAM) to design an 8x4 FIFO. The 8x4 FIFO is clocked by two external clocks (wclk_in and rclk_in). CLBs are used to implement the EMPTY_FLAG and FULL_FLAG.

There are eight BRAM slices on the SLG47910 device with configurable depths and widths. The top-level BRAM placements are shown in Figure 1.This FIFO implementation uses BRAM_0 slice that is configured as a 512x8 SRAM but only four data bits (x4) are used.

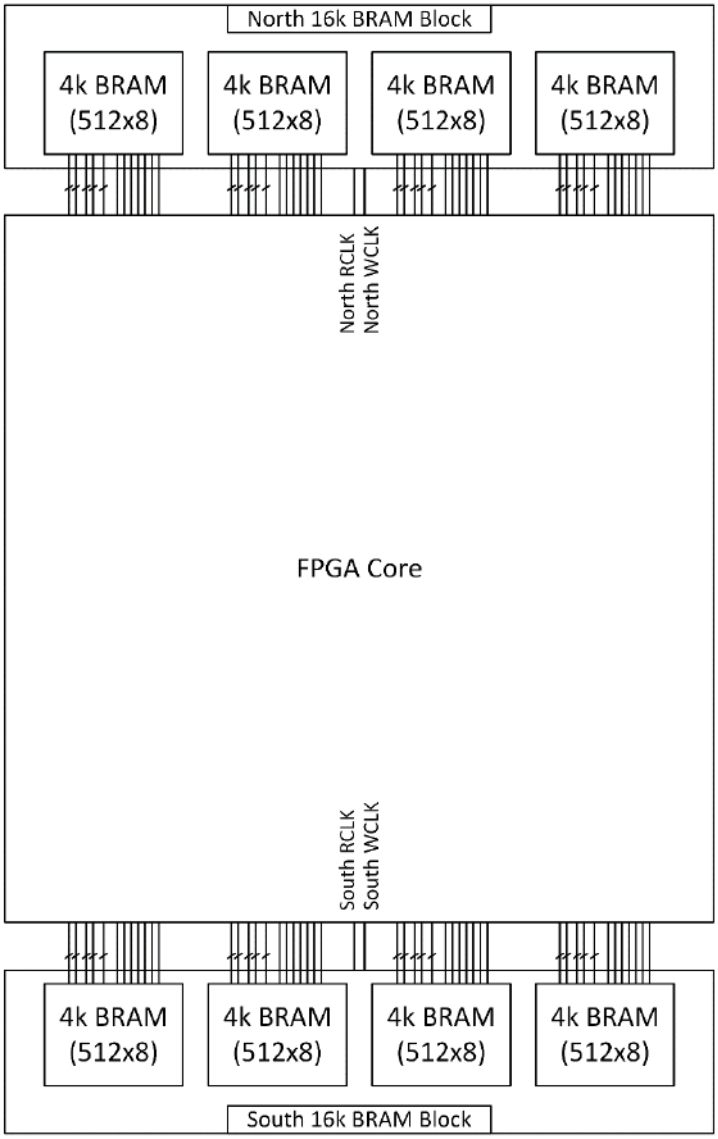

Figure 1: FPGA Core and BRAM Blocks

2. FIFO Description

In this design example the BRAM is configured as 512x8 by setting RATIO[1:0] = 2'b00. The address space has a depth of eight so only a small portion of the BRAM address space is used. Only DIN[3:0] and DOUT[3:0] are used during write and read cycles. The BRAM inputs and its description are shown in Figure 3 and Table 1. The BRAM clocks, WCLK and RCLK are driven by GPIOs. The FIFO block diagram is shown in Figure 2. The Write Pointer and Read Pointer generate the addresses for the BRAM. The Flag Logic keeps track of when the FIFO is full or empty by comparing the difference between the write and read addresses.

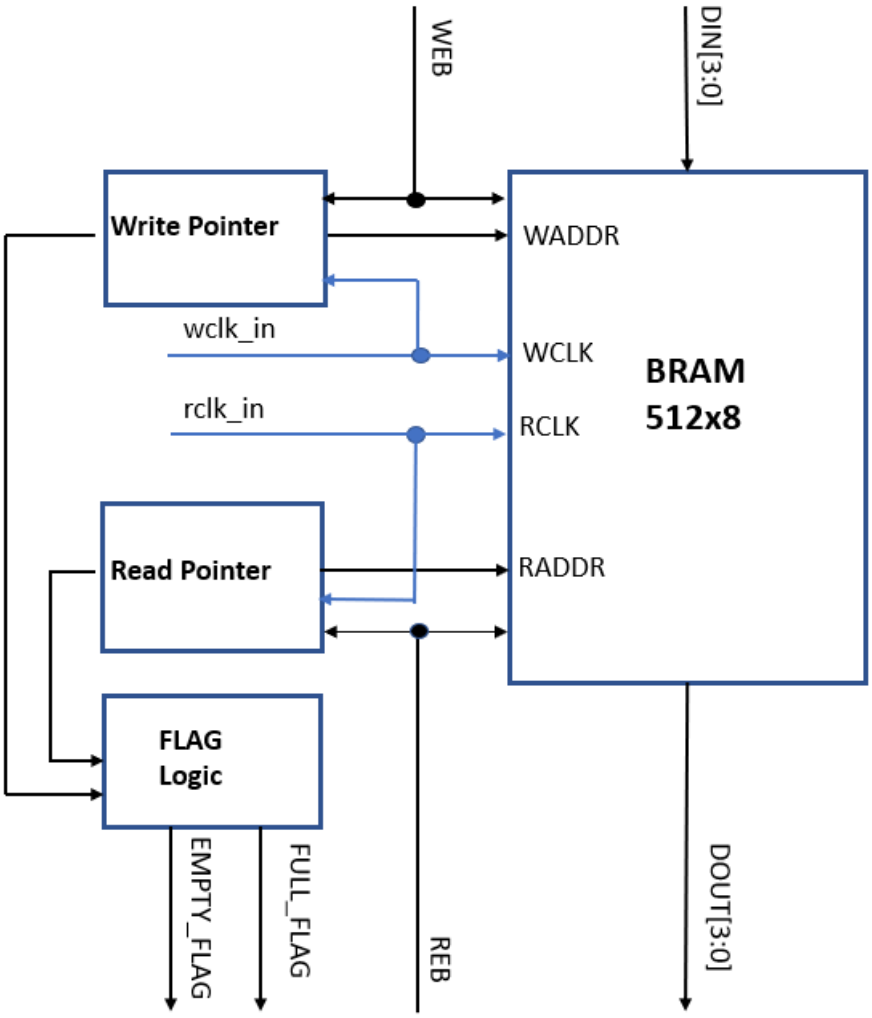

Figure 2: 5x4 FIFO with FULL & EMPTY FLAGs

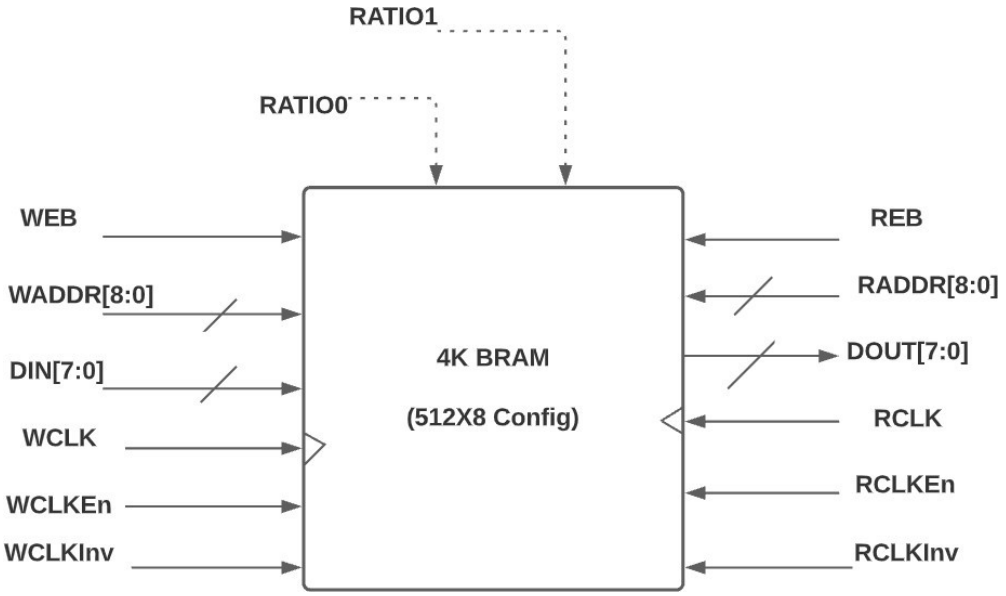

Figure 3: BRAM Slice Structure

Table 1: BRAM Slice Signal Description

Signal Name	Signal Direction	Signal Description
WEB	Input	Write Enable (active low)
WADDR[8:0]	Input	Write Address Bits; for anything deeper than 512, the unused DIN's can be repurposed as WADDR
DIN[7:0]	Input	Data Input Bits
WCLKEN	Input	Write Clock Enable (active low)
WCLKInv	Input	Write Clock Inversion Control
RCLKInv	Input	Read Clock Inversion Control
RCLKEN	Input	Read Clock Enable (active low)
DOUT[7:0]	Output	Data Output Bits
RADDR[8:0]	Input	Read Address Bits; for anything deeper than 512, the unused DIN's can be repurposed as RADDR
REB	Input	Read Enable (active low)
PD	Input	Power Down (active high)
WCLK	Input	Write Clock (default Rising Edge, but with falling edge option)
RCLK	Input	Read Clock (default Rising Edge, but with falling edge option)
RATIO[1:0]	Input	Data Width Selection Bits 00: 512x8 01: 1024x4 10: 2048x2 11: 4096x1

3. Ingredients

- SLG47910 Device
- ForgeFPGA Development Board and power cables
- ForgeFPGA Socket Adaptor Board
- Latest Revision of the ForgeFPGA Workshop software

4. FIFO Verilog Code

The BRAM_FIFO design is available for download (FIFO using BRAM.ffpga). It contains the complete FIFO design using the BRAM module and the Digital array of the SLG47910 device. Only BRAM0 on the North side of the die is used (`assign BRAM0_PD = 1'b0`) the other BRAMs are powered down `(assign BRAM_PD = 1'b1)`. The nReset input resets the addresses and flag logic. The address space of the FIFO can be modified by adjusting the DEPTH parameter.

Shown below is the (*top*) module named FIFO_BRAM. The Verilog code for 8x4 FIFO using BRAM can be found in the complete design example. It is available for download (AN-FG-013 FIFO using BRAM.ffpga)

Multiple `always` block in the Verilog code allows the user to configure the read and the write clock of the BRAM according to their use.

```
(*top*) module FIFO_BRAM #(
parameter DEPTH = 3
)(
(* iopad_external_pin *)input nReset,
(* iopad_external_pin, clkbuf_inhibit *)input wclk,
(* iopad_external_pin, clkbuf_inhibit *) input rclk,
(* iopad_external_pin *) input wclk_in,
```

```
(* iopad_external_pin *) output wclk_OE,
(* iopad_external_pin *) input rclk_in,
(* iopad_external_pin *) output rclk_OE,
(* iopad_external_pin *) output wclk_out,
(* iopad_external_pin *) output rclk_out,
(* iopad_external_pin *) input [3:0] DIN,
(* iopad_external_pin *) input [3:0] DIN_OE,
(* iopad_external_pin *) input WE,
(* iopad_external_pin *) input WE_OE,
(* iopad_external_pin *) input RE_OE,
(* iopad_external_pin *) input RE,
(* iopad_external_pin *) output reg [3:0] DOUT,
(* iopad_external_pin *) output reg FIFO_full,
(* iopad_external_pin *) output reg FIFO_empty,
(* iopad_external_pin *) output [1:0] BRAM0_RATIO,
(* iopad_external_pin *) output BRAM0_PD,
(* iopad_external_pin *) output BRAM1_PD,
(* iopad_external_pin *) output BRAM2_PD,
(* iopad_external_pin *) output BRAM3_PD,
(* iopad_external_pin *) output reg [7:0] BRAM0_DATA_IN,
(* iopad_external_pin *) output reg BRAM0_WEB,
(* iopad_external_pin *) output reg BRAM0_WCLKEN,
(* iopad_external_pin *) output reg [8:0] BRAM0_WRITE_ADDR,
(* iopad_external_pin *) input [3:0] BRAM0_DATA_OUT,
(* iopad_external_pin *) output reg BRAM0_REB,
(* iopad_external_pin *) output reg BRAM0_RCLKEN,
(* iopad_external_pin *) output reg [8:0] BRAM0_READ_ADDR
);
```

5. Floorplan : CLB Utilization

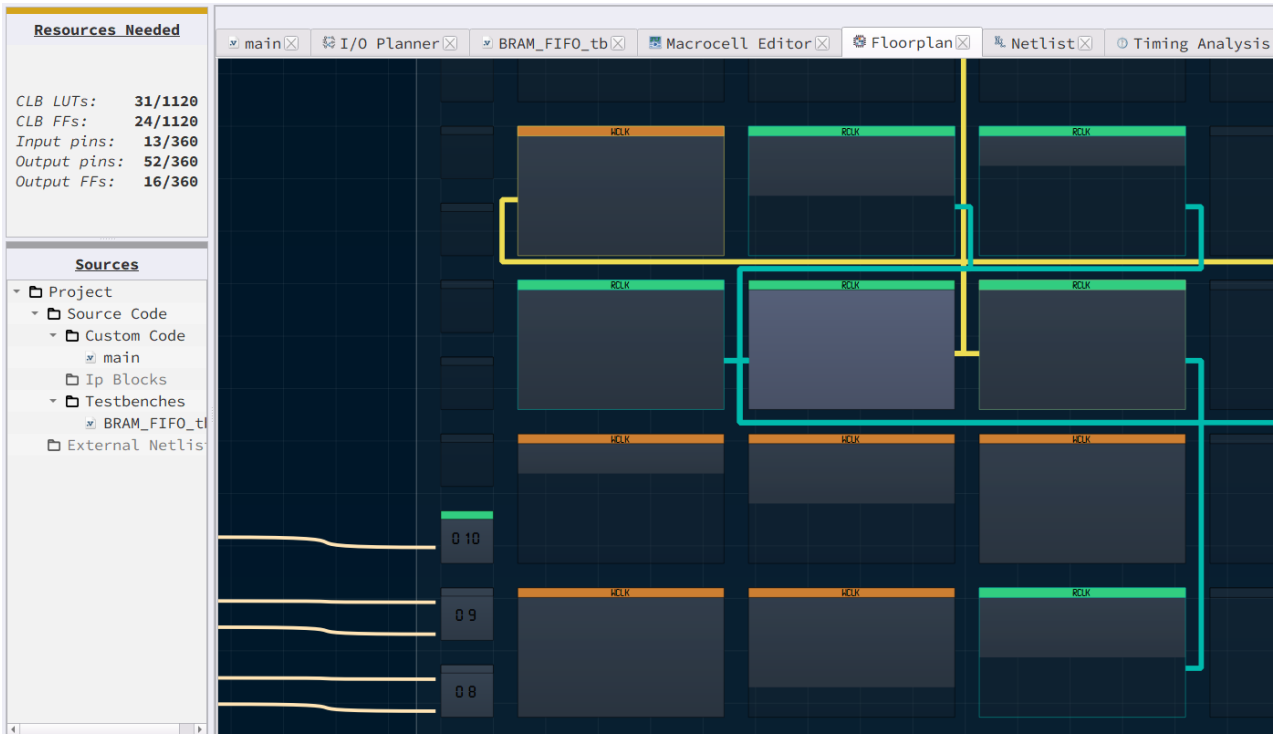

Figure 4 : Floorplan

From the above Figure 4, we can see a part of the floorplan for this application note. Also, on the left corner of the figure we can see a mini resource needed list.

From the Verilog code the user can observe that only BRAM_0 has been used for this application and all the other BRAM_[1:7] has been disabled. This can also be observed under the floorplan tab in the software and in Figure 5.

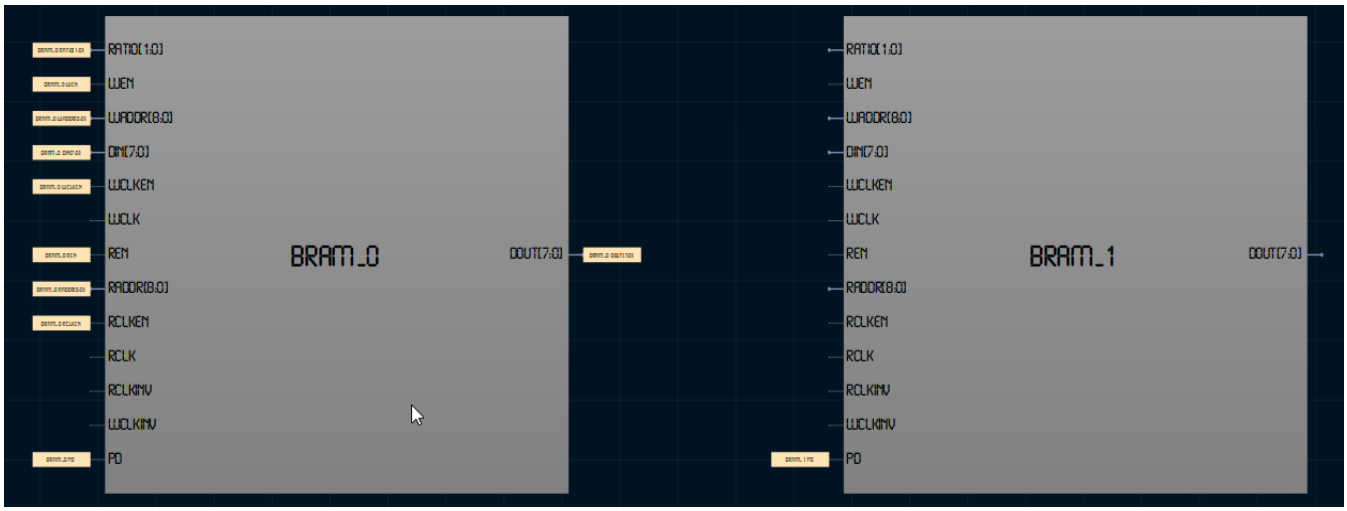

Figure 5 : BRAM_0 is being used and BRAM_1 is not being used

6. 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, 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. (Figure 6)

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. (Figure 4). Confirm that the IOs selected in the IO Planner (Figure 6) are shown in the floorplan. The IO Planner has been set in such a way that the board uses an external clock instead of the OSC clock on-board also that we use only BRAM_0 for this application and disable the other BRAMs on board.

7. This design file also comes with a testbench to verify the working of FIFO. The simulation waveforms can be verified by using the inbuilt GTKWave software. BRAM_FIFO_tb is the testbench for this design file. The user can launch the GTKWave software by clicking on the Simulate Testbench icon on the toolbar. This will automatically launch the GTKWave software provided there are no errors in your testbench. The errors/ issues can be observed from the console at the bottom of window.

8.In the GTKWave software, the user can insert the desired signal in the window from the list of all the signals in the code and observe the waveform. Through the generated waveform the user can verify the functionality of FIFO without connecting the development board. (Figure 7).

POSITION	FUNCTION	
IOB tile[0, 0] coord[1, 31] Output1	BRAM0_DATA_IN[0]	BRAM0_DATA_IN[0]
IOB tile[0, 0] coord[1, 31] Output0	BRAM0_DATA_IN[1]	BRAM0_DATA_IN[1]
IOB tile[0, 0] coord[1, 30] Output1	BRAM0_DATA_IN[2]	BRAM0_DATA_IN[2]
IOB tile[0, 0] coord[1, 30] Output0	BRAM0_DATA_IN[3]	BRAM0_DATA_IN[3]
IOB tile[0, 0] coord[0, 31] Output1	BRAM0_DATA_IN[4]	BRAM0_DATA_IN[4]
IOB tile[0, 0] coord[0, 31] Output0	BRAM0_DATA_IN[5]	BRAM0_DATA_IN[5]
IOB tile[0, 0] coord[0, 30] Output1	BRAM0_DATA_IN[6]	BRAM0_DATA_IN[6]
IOB tile[0, 0] coord[0, 30] Output0	BRAM0_DATA_IN[7]	BRAM0_DATA_IN[7]
IOB tile[0, 0] coord[1, 31] Input1	BRAM0_DATA_OUT[0]	BRAM0_DATA_OUT[0]
IOB tile[0, 0] coord[1, 31] Input0	BRAM0_DATA_OUT[1]	BRAM0_DATA_OUT[1]
IOB tile[0, 0] coord[1, 30] Input1	BRAM0_DATA_OUT[2]	BRAM0_DATA_OUT[2]
IOB tile[0, 0] coord[1, 30] Input0	BRAM0_DATA_OUT[3]	BRAM0_DATA_OUT[3]
IOB tile[0, 0] coord[6, 31] Output0	BRAM0_PD	BRAM0_PD
IOB tile[0, 0] coord[0, 28] Output0	BRAM0_RATIO[0]	BRAM0_RATIO[0]
IOB tile[0, 0] coord[0, 28] Output1	BRAM0_RATIO[1]	BRAM0_RATIO[1]
IOB tile[0, 0] coord[0, 27] Output1	BRAM0_RCLKEN	BRAM0_RCLKEN
IOB tile[0, 0] coord[6, 30] Output1	BRAM0_READ_ADDR[0]	BRAM0_READ_ADDR[0]
IOB tile[0, 0] coord[6, 30] Output0	BRAM0_READ_ADDR[1]	BRAM0_READ_ADDR[1]
IOB tile[0, 0] coord[5, 31] Output1	BRAM0_READ_ADDR[2]	BRAM0_READ_ADDR[2]
IOB tile[0, 0] coord[5, 31] Output0	BRAM0_READ_ADDR[3]	BRAM0_READ_ADDR[3]
IOB tile[0, 0] coord[5, 30] Output1	BRAM0_READ_ADDR[4]	BRAM0_READ_ADDR[4]
IOB tile[0, 0] coord[5, 30] Output0	BRAM0_READ_ADDR[5]	BRAM0_READ_ADDR[5]

Figure 6 :IO Planner

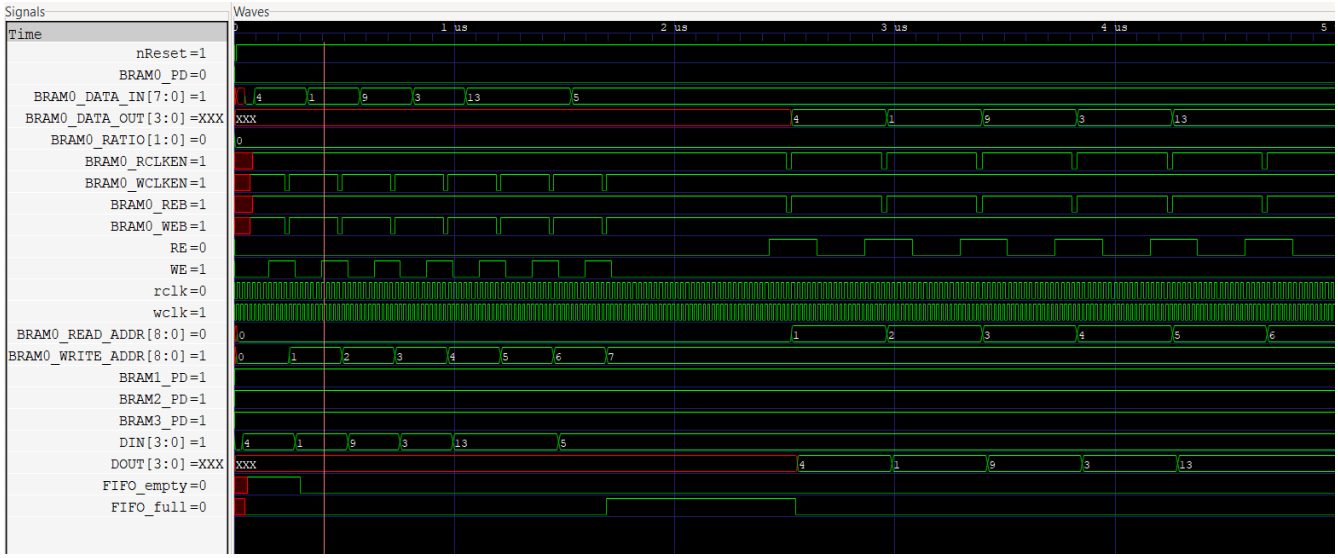

Figure 7: GTKWave simulation

9. Once the user is satisfied with the waveforms, the user can Debug the design file. Close the FPGA Editor and go to the ForgeFPGA widow . 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.

10. 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.

11. Apply the desired inputs from GPIO[0-3] and observed the DOUT through GPIO[4-7]. GPIO8 will indicate if the FIFO is full and GPIO9 will indicate if the FIFO pipeline is empty. The user can observe waveforms through the inbuilt Logic Analyzer (Figure 8), or the user can connect the desired GPIOs to an oscilloscope to observe the waveforms.

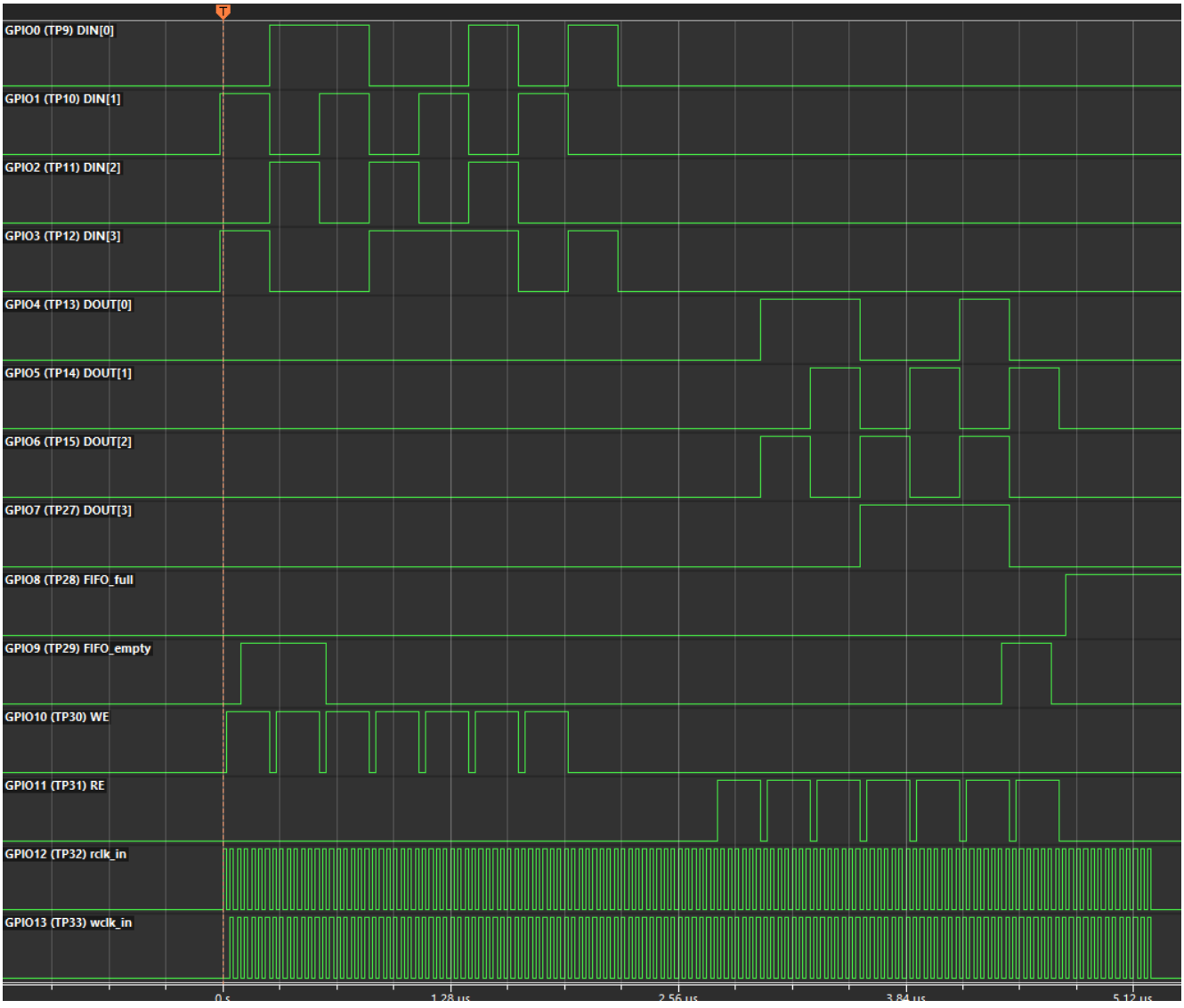

Figure 8 : Output on Logic Analyzer

7. Conclusion

The procedure outlined in this application note can applied to any BRAM the user wants to use to implement this design. Similar procedure needs to be followed to implement anything using BRAM. The FIFO using BRAM.ffpga design file is ready for download.

If interested, please contact the ForgeFPGA Business Support Team.

8. Revision History

Revision	Date	Description
1.00	Feb 24, 2023	Initial release.

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 skilled in the art 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 for development of 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 out of 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 o any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.0 Mar 2020)

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/

Trademarks

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