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

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: https://www.renesas.com/us/en/products/programmable-mixed-signal-asic-ip-products/forgefpga-low-density-fpgas. 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, Renesas Electronics

[2] AN-FG-011 How to implement FIFO in ForgeFPGA using BRAM.ffpga, ForgeFPGA Design File

[3] ForgeFPGA SLG47910, Datasheet, Renesas Electronics

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.

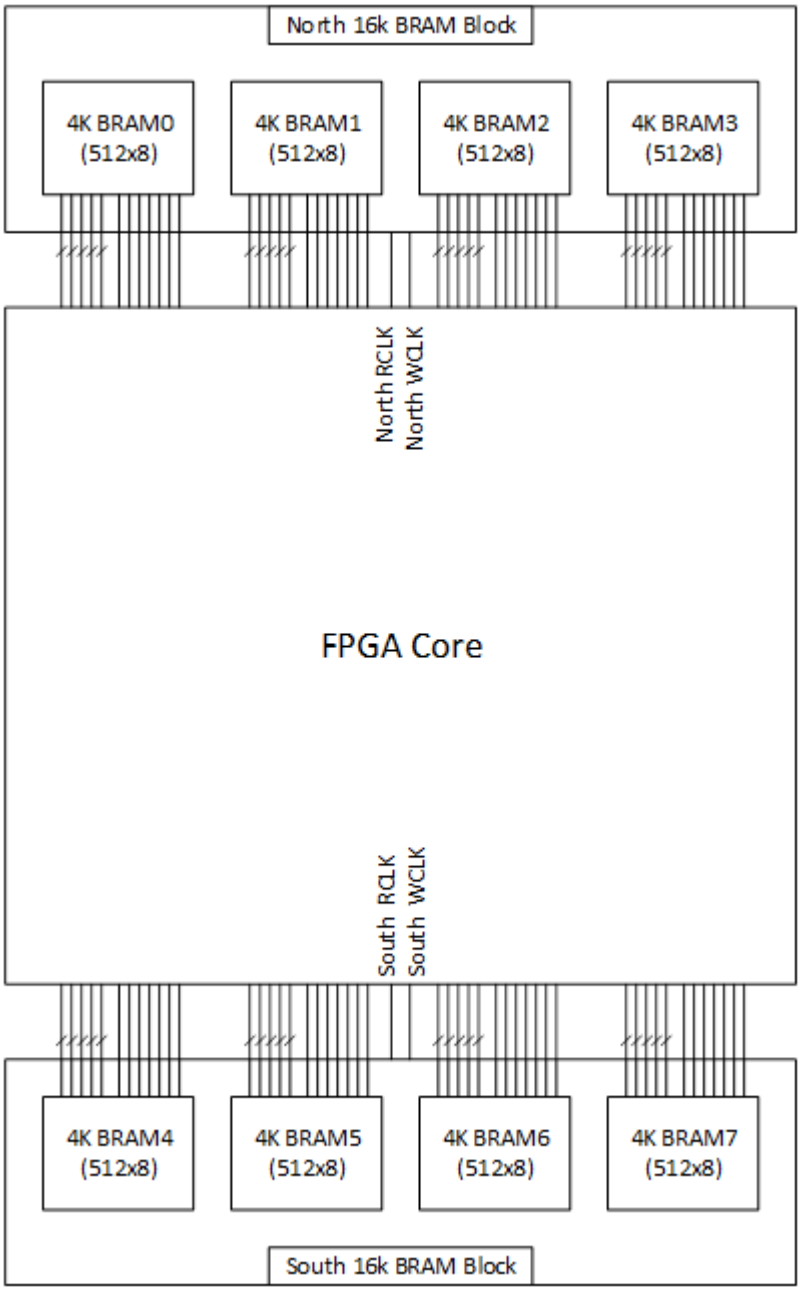

Figure 1: FPGA Core and BRAM Blocks

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.

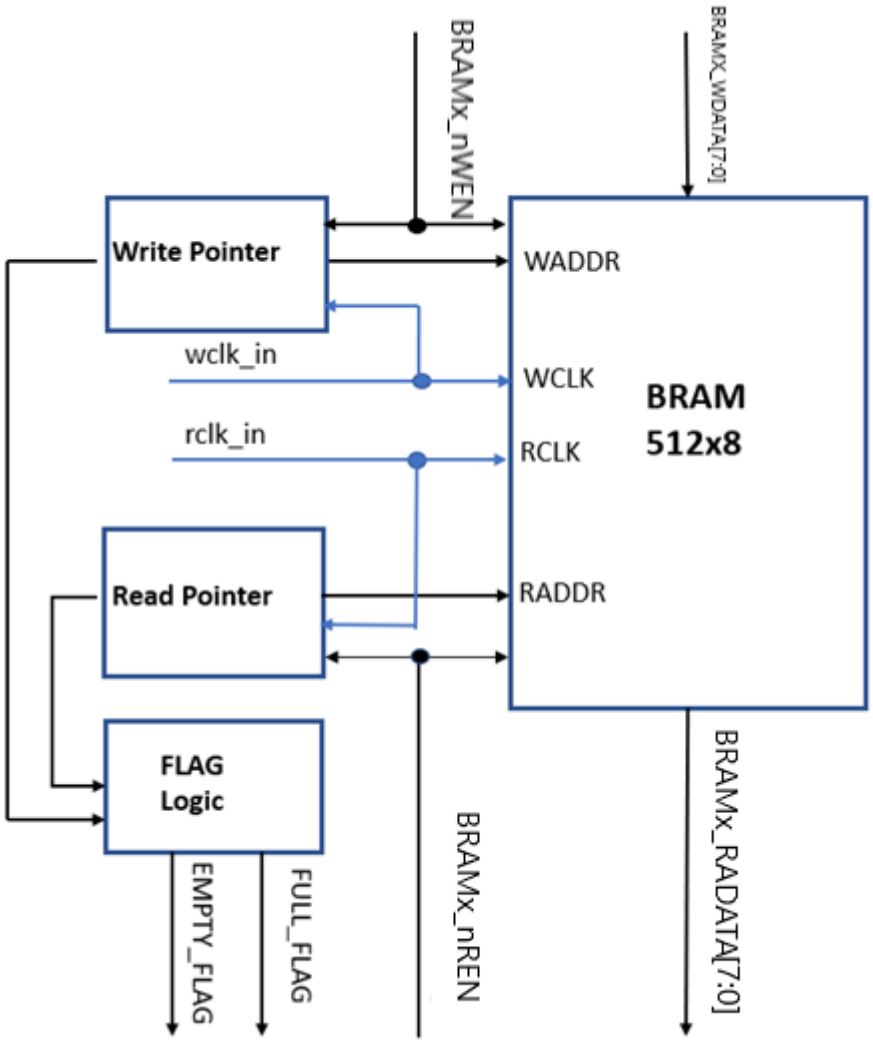

Figure 2: 5x4 FIFO with FULL & EMPTY FLAGs

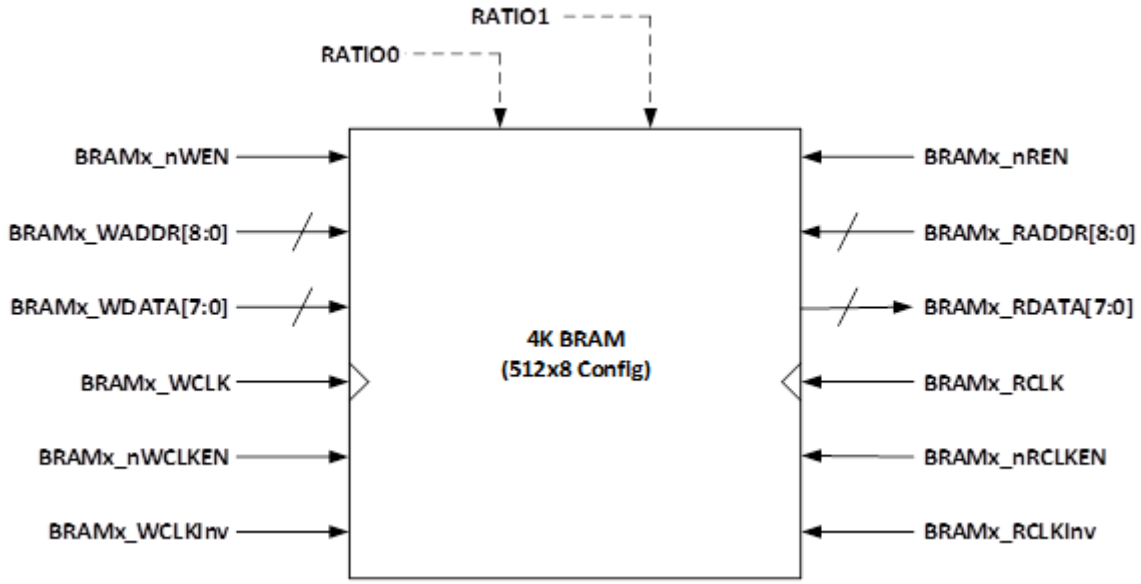

Figure 3: BRAM Slice Structure

Table 1: BRAM Slice Signal Description

Name	Direction	Description
BRAMx_nWEN	Input	Write Enable (active low)
BRAMx_WADDR[8:0]	Input	Write Address Bits; for anything deeper than 512, the unused DIN's can be repurposed as WADDR
BRAMx_WDATA[7:0]	Input	Data Input Bits
BRAMx_nWCLKEN	Input	Write Clock Enable (active low)
BRAMx_WCLKInv	Input	Write Clock Inversion Control
BRAMx_RCLKInv	Input	Read Clock Inversion Control
BRAMx_nRCLKEN	Input	Read Clock Enable (active low)
BRAMx_RDATA[7:0]	Output	Data Output Bits
BRAMx_RADDR[8:0]	Input	Read Address Bits; for anything deeper than 512, the unused DIN's can be repurposed as RADDR
BRAMx_nREN	Input	Read Enable (active low)
BRAMx_WCLK	Input	Write Clock (default Rising Edge, but with falling edge option)
BRAMx_RCLK	Input	Read Clock (default Rising Edge, but with falling edge option)
RATIO [1:0]	Input	Data Width Selection Bits 00: 512 x 8 01: 1024 x 4 10: 2048 x 2 11: 4096 x 1

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 (How to implement FIFO in ForgeFPGA using BRAM.ffpga). It contains the complete FIFO design using the BRAM module and the FPGA Core of the SLG47910 device. The desired BRAM is used by specifying the number BRAM0, BRAM1, etc. 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's input -output ports. The Verilog code for 8x4 FIFO using BRAM can be found in the complete design example. It is available for download (AN-FG-011 How to implement FIFO in ForgeFPGA 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 *) input            rclk_in,
   (* iopad_external_pin *) output           wclk_out,
   (* iopad_external_pin *) output           rclk_out,
   (* iopad_external_pin *) input      [3:0] DIN,
   (* iopad_external_pin *) input            WE,
   (* iopad_external_pin *) input            RE,
   (* iopad_external_pin *) output reg [3:0] DOUT,
   (* iopad_external_pin *) output           DOUT0_oe,
   (* iopad_external_pin *) output           DOUT1_oe,
   (* iopad_external_pin *) output           DOUT2_oe,
   (* iopad_external_pin *) output           DOUT3_oe,
   (* iopad_external_pin *) output reg       FIFO_full,
   (* iopad_external_pin *) output reg       FIFO_empty,
   (* iopad_external_pin *) output           FIFO_full_oe,
   (* iopad_external_pin *) output            FIFO_empty_oe,
   (* iopad_external_pin *) output     [1:0] BRAM0_RATIO,
   (* iopad_external_pin *) output reg [7:0] BRAM0_DATA_IN,
   (* iopad_external_pin *) output reg       BRAM0_WEN,
   (* 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_REN,
   (* iopad_external_pin *) output reg       BRAM0_RCLKEN,
   (* iopad_external_pin *) output reg [8:0] BRAM0_READ_ADDR,
   (* iopad_external_pin *) output           ext_en0, // external clock
   (* iopad_external_pin *) output           ext_en1     // external clock
  );
```

5. Floorplan: CLB Utilization

From the below 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.

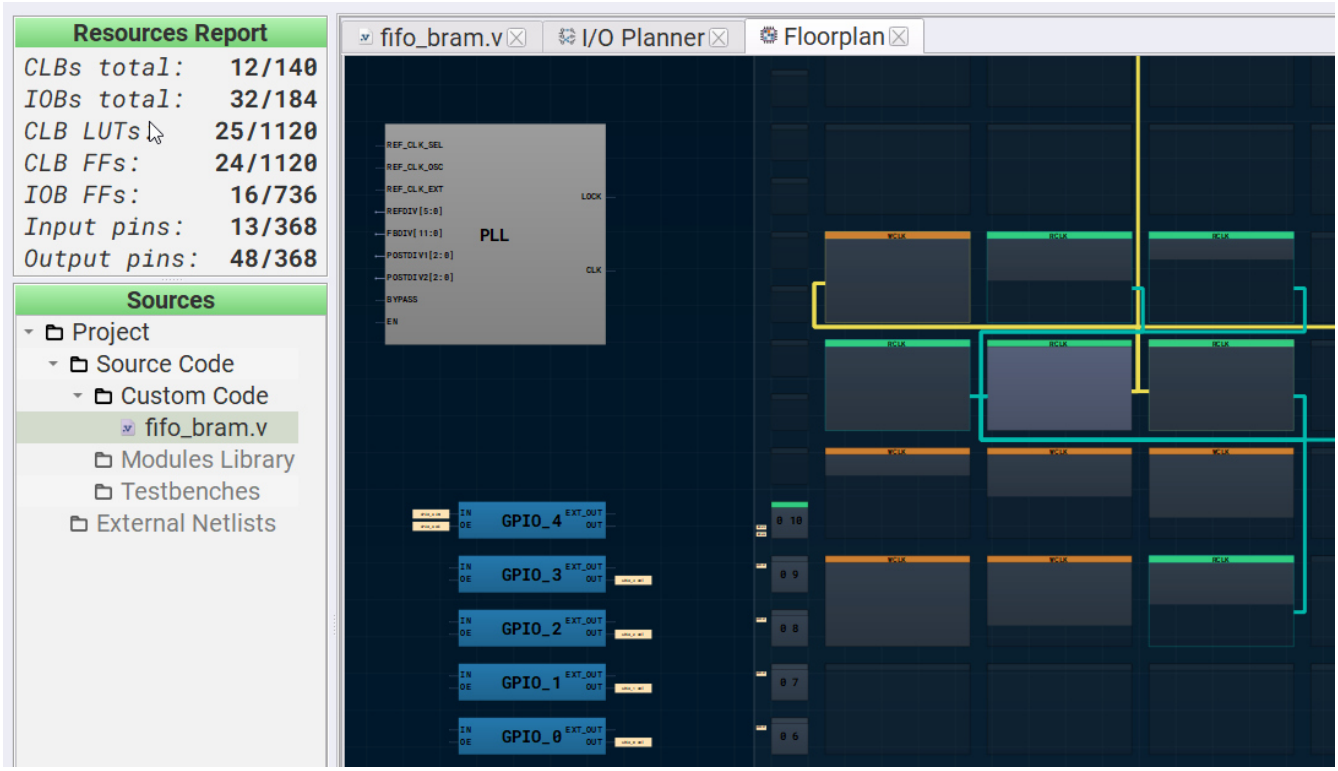

Figure 4 : Floorplan

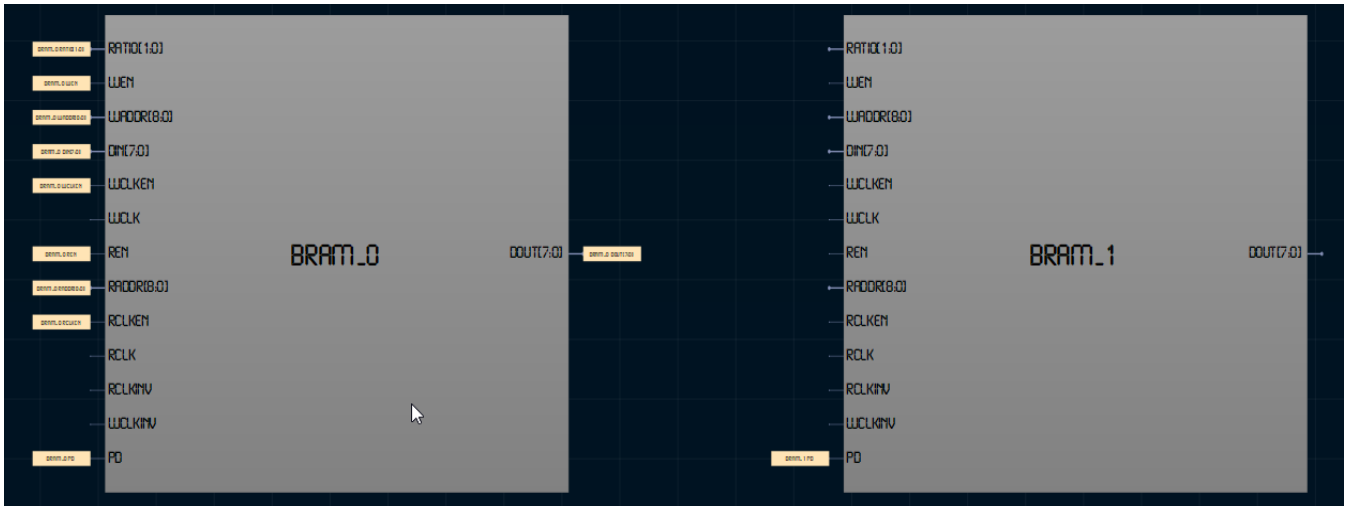

Figure 5: BRAM_0 is being used and BRAM_1 is not being used

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. 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.
6) 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.

FUNCTION	DIRECTION	PORT
[PIN 13] GPIO0_IN	Input	DIN[0]
[PIN 14] GPIO1_IN	Input	DIN[1]
[PIN 15] GPIO2_IN	Input	DIN[2]
[PIN 16] GPIO3_IN	Input	DIN[3]
[PIN 17] GPIO4_OE	Output	DOUT0_oe
[PIN 18] GPIO5_OE	Output	DOUT1_oe
[PIN 19] GPIO6_OE	Output	DOUT2_oe
[PIN 20] GPIO7_OE	Output	DOUT3_oe
[PIN 17] GPIO4_OUT	Output	DOUT[0]
[PIN 18] GPIO5_OUT	Output	DOUT[1]
[PIN 19] GPIO6_OUT	Output	DOUT[2]
[PIN 20] GPIO7_OUT	Output	DOUT[3]
[PIN 24] GPIO9_OUT	Output	FIFO_empty
[PIN 24] GPIO9_OE	Output	FIFO_empty_oe
[PIN 23] GPIO8_OUT	Output	FIFO_full
[PIN 23] GPIO8_OE	Output	FIFO_full_oe
[PIN 2] GPIO11_IN	Input	RE
[PIN 1] GPIO10_IN	Input	WE
LOGIC_AS_CLK0_EN	Output	ext_en0
LOGIC_AS_CLK1_EN	Output	ext_en1
FPGA_CORE_READY	Input	nReset
LOGIC_AS_CLK0	Input	rclk
REF_BRAM(0..3)_READ_CLK	Output	rclk
[PIN 3] GPIO12_IN	Input	rclk_in
REF_LOGIC_AS_CLK0	Output	rclk_out
LOGIC_AS_CLK1	Input	wclk
REF_BRAM(0..3)_WRITE_CLK	Output	wclk
[PIN 4] GPIO13_IN	Input	wclk_in

Figure 6: IO Planner

7) In the ForgeFPGA Workshop window, click on the BRAM block on the bottom. The user will then see the properties tab on the left side of the window. If it doesn't appear, then enable the Properties section by clicking on the Properties icon on the top toolbar. Inside the properties section make sure to enable the BRAMx that is being tested (North BRAM corresponds to BRAM_0, BRAM_1, BRAM_2 and BRAM_3 and similarly, South BRAM corresponds to BRAM_4, BRAM_5, BRAM_6 and BRAM_7). In the example, ensure that the North BRAM is enabled, and South BRAM is disabled. (See Figure 7)

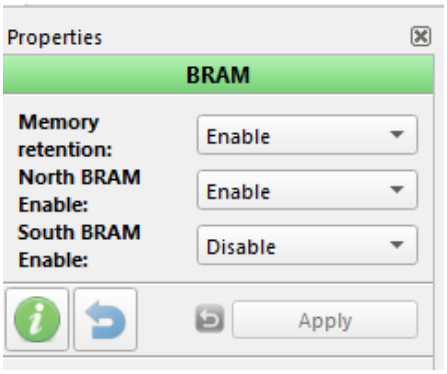

Figure 7. BRAM Properties

8) Once the user is satisfied with the design code, 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 V_{DDIO} pin and set V_{DDIO}= 1.8v.
9) 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.
10) 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.

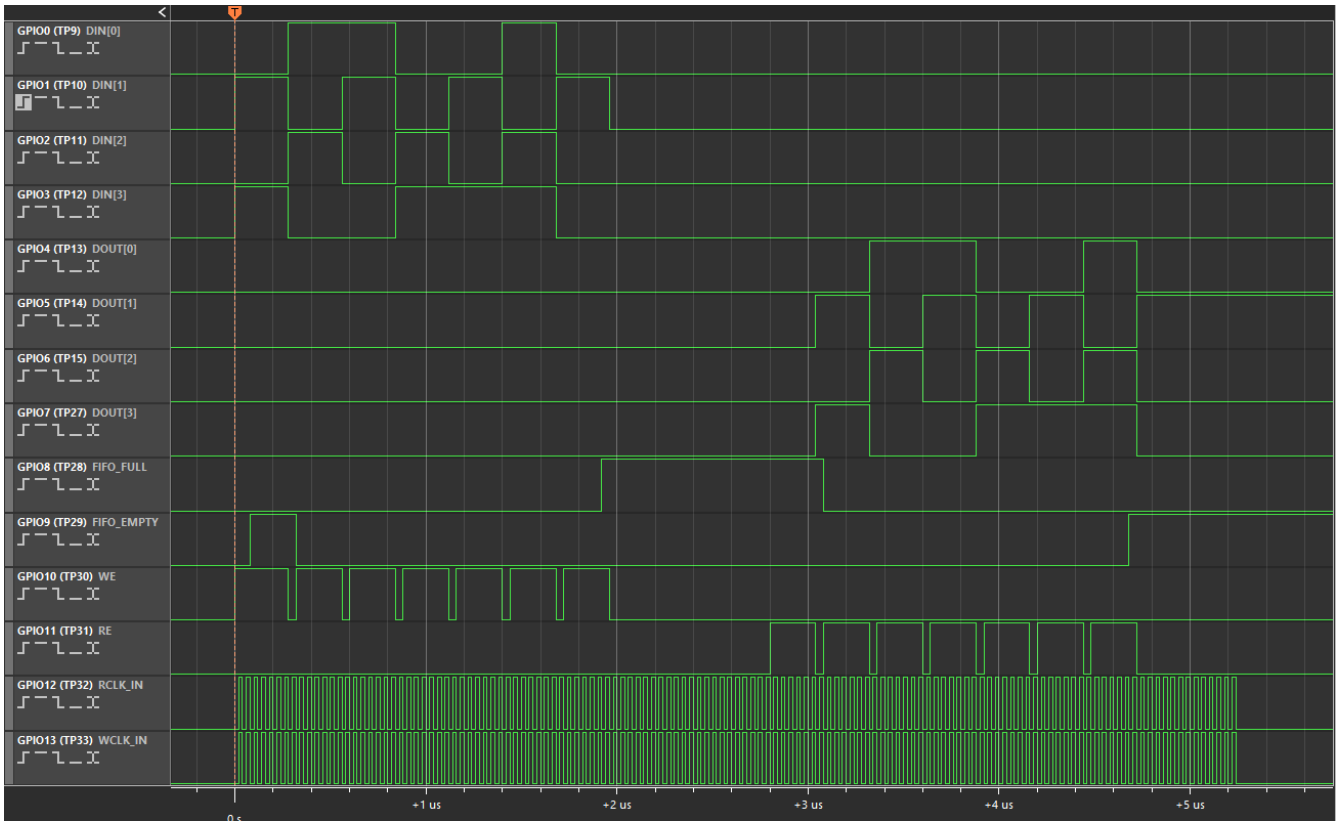

Figure 8: Output on Logic Analyzer

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 the BRAM of different Ratio or use a different BRAM [1-3]. Make changes at the appropriate place to observe the correct results. 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 15, 2023	Initial release.
2.0	Feb 23, 2024	Updated according to K1BB revision
3.0	Jul 24, 2024	Updated as per 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.