

Designing an I²C Target (Slave) Module using the SLG47910 ForgeFPGA SLG47910V

This application note describes how to use an I²C Target (also known as Slave) module in the SLG47910 ForgeFPGA. A test bench was used to verify its functionality by observing the waveforms on an oscilloscope. 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. Description	3
4.1 Write Operation to Memory	3
4.2 Read Operation from Memory	3
5. Emulation.....	4
5.1 Write/Read operation on a Word	4
5.2 Write/Read operation on a Sequence of Words.....	5
5.3 Read Word operation on the Current Address	6
6. Requirements	7
7. Verilog Code.....	7
8. Test Bench.....	8
9. Simulation Results	9
10. Step by Step Design	12
11. Conclusion	13
12. Revision History	14

1. Terms and Definitions

I ² C	Inter-Integrated Circuit
SDA	Serial Data
SCL	Serial Clock
FPGA	Field Programmable Gate Array
FPGA Editor	Main FPGA design and simulation window
Go Configure Software Hub	Main window for device selection
ForgeFPGA	Window Main FPGA project window for debug and IO programming

2. References

For related documents and software, please visit [ForgeFPGA Low-density FPGA: Small-scale Programmable Logic Device | Renesas](#)

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] [How to design I2C Slave Module.ffpga](#), Design file, Renesas Electronics

[3] [SLG47910](#), Datasheet, Renesas Electronics

3. Introduction

The I²C bus is a standard bidirectional interface that utilizes a serial data line (SDA) and a serial clock line (SCL). As the I²C Target module can only respond to requests from a controller (master), an external controller is also required to demonstrate the operating principle of this module.

To capture the received data sent from the controller, a distributed memory of 8 words (8 bits equal to one word) is used, and an oscilloscope is used to provide additional visualization of the transmitted data.

The block diagram for our implementation is shown in [Figure 1](#):

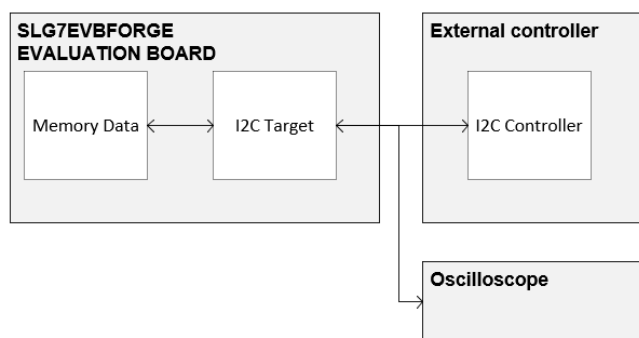


Figure 1. Block Diagram

[Table 1](#) shows the word structure of Write/Read data that will be written to memory.

Table 1. Write/Read data

#	0	1	2	3	4	5	6	7
Word	A1	B2	C3	D4	E5	F6	A7	B8

4. Description

The I²C Target module is activated by an internal enable signal that must be active to begin operation. The enable signal is only needed to start the module when the start condition happens. If the signal is reset during operation, the module will complete its current read/write operation. The device address is specified in the I²C Target module through a configurable parameter before implementation of the design.

The target device requires an 8-bit address word once the start condition is given. The first seven bits are reserved for the address, and the eighth bit is used to select the write/read operation. A read operation is initiated if this bit is high, and a write operation is initiated if this bit is low. If the address transmitted by the controller does not match the address set through the parameter, the module will go into an idle state.

When the controller writes a word to the target, the data is stored in the o_data_rx buffer and the ready signal o_int_rx is set to one period of the internal frequency. When the controller reads a word from the target, the data on the i_data_tx line must be already set before the start condition occurs. The o_int_tx pulse signifies the end of the transfer, after which it is possible to change the data on the i_data_tx line.

The o_busy signal will be set HIGH from when the start condition occurs to until the stop condition occurs.

4.1 Write Operation to Memory

To write a word to a specified address of memory, the controller must transmit the device address in the first word, the address of the memory cell in the second word, and the data in the third word (see Figure 2).

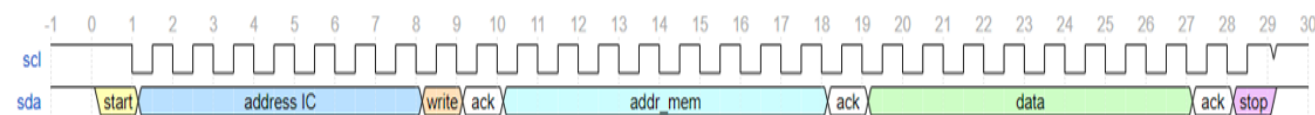


Figure 2. Write a word to memory at a specified address

To write a sequence of words to memory, the device address must be specified in the first word, the address of the memory cell to which writing will begin must be specified with the second word, followed by the words of the data to be transmitted. The address in memory will be incremented automatically (see Figure 3).



Figure 3. Write a sequence to memory

4.2 Read Operation from Memory

To read data from memory at a specific address, the device address is specified in the first word followed by the memory address. Then a read operation to read from the specified memory address is given (see Figure 4).

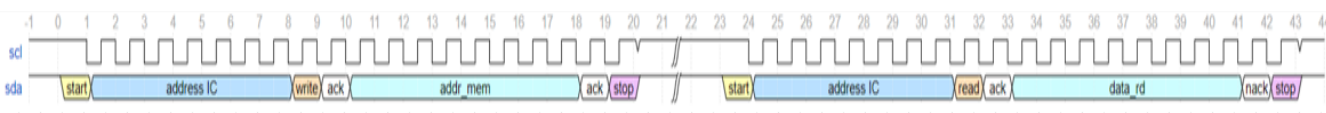


Figure 4. Read a word from memory at the specified address

To read a sequence of words from memory, we first set the memory address from which reading will begin, as in the previous example. After setting the address, a read operation is performed on the specified address. The address in memory is incremented automatically (see Figure 5).



Figure 5. Read a sequence of words in memory

The address counter stores the last address accessed during the last read or write operation, incremented by one. To read data from the currently specified memory address, a read operation must be performed (see Figure 6).

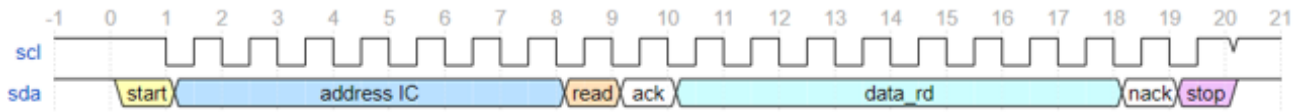


Figure 6. Read a word from the current memory address

5. Emulation

A project with an I²C Target module and memory is loaded into the SLG7EVBFORGE Evaluation board with a SLG47910 FPGA device. The External controller is an I²C Ccontroller. The data transmitted and received by the I²C controller is monitored by an oscilloscope.

The schematic for the implementation of our test setup is shown in Figure 7.

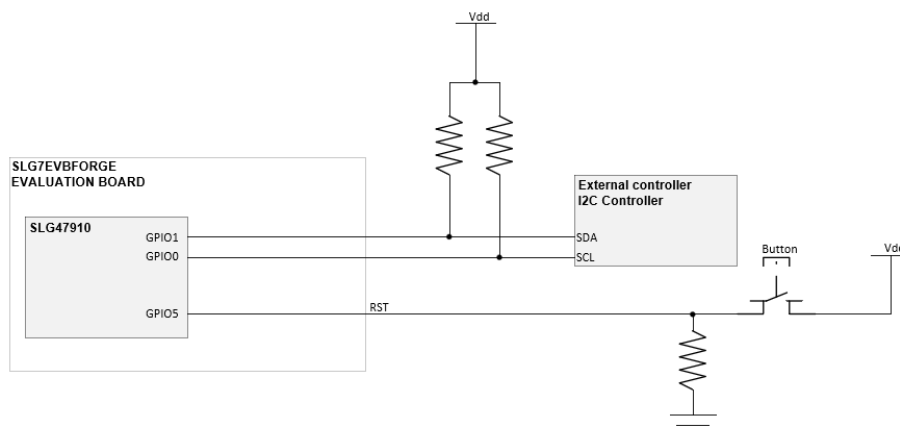


Figure 7. Test Setup Schematic

The reset signal is active high. When the button is pressed, the reset signal is set HIGH, and the registers of the project are reset. In turn, the memory is cleared only after a complete power reset. The circuit uses two standard pull-up resistors for the I²C bus SDA and SCL lines.

The design has the following features:

- 50 MHz Internal frequency
- 100 KHz I²C frequency
- 4.7 kΩ Pull up resistors
- GPIO0 (SCL) and GPIO1 (SDA) are configured to operate at 1x open drain output mode;
- The I²C Target address specified in the parameter is 0x08.

Next, let's look at how writing/reading data into memory is done.

5.1 Write/Read operation on a Word

Figure 8 shows the oscilloscope results when writing a single word (0xA1) to a memory address (0x3). Addressing starts from 0x0.

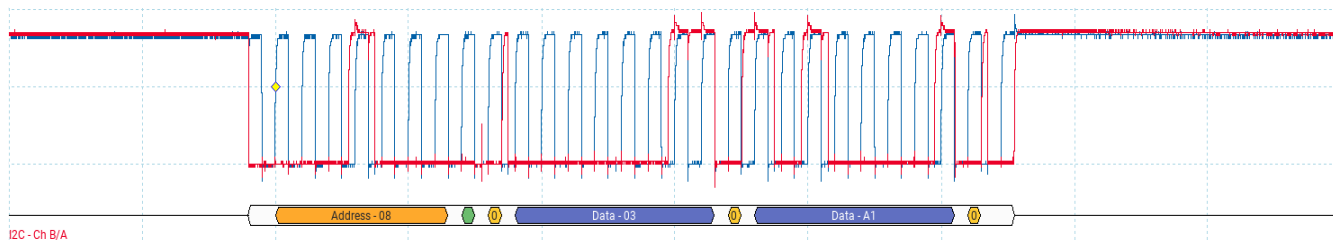


Figure 8. Results of writing Word 0xA1 to Address 0x3 from an oscilloscope

An address equal to 0x3 is set from which data will be read. After setting the address (Figure 9), a read operation is performed (Figure 10).

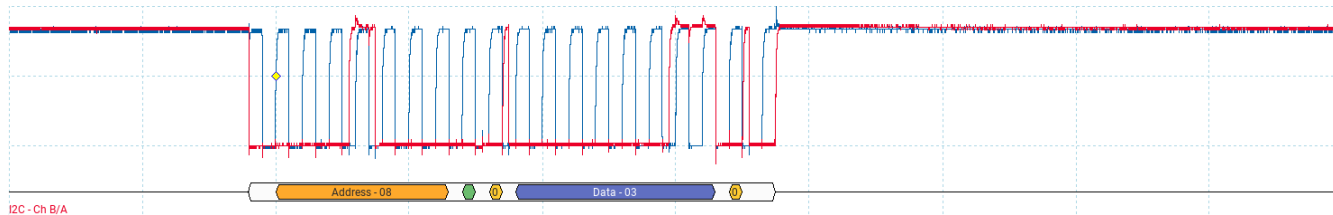


Figure 9. Setting the address (write operation)

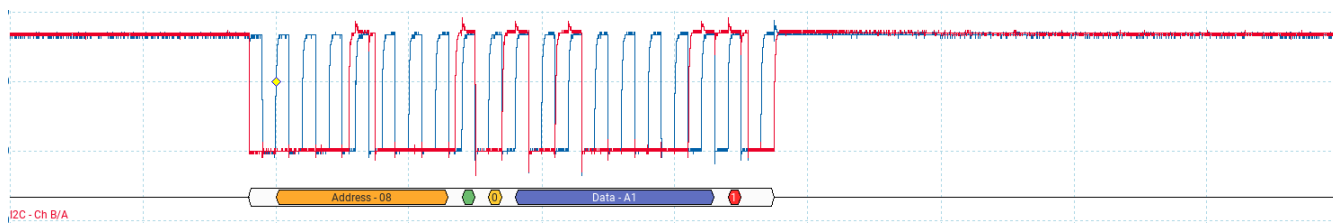


Figure 10. Results of writing Word 0xA1 to Address 0x3 from an oscilloscope

5.2 Write/Read operation on a Sequence of Words

Figure 11 shows the oscilloscope results of writing a sequence of 8 words starting from address 0x0 with the sequence of values taken from Table 1.

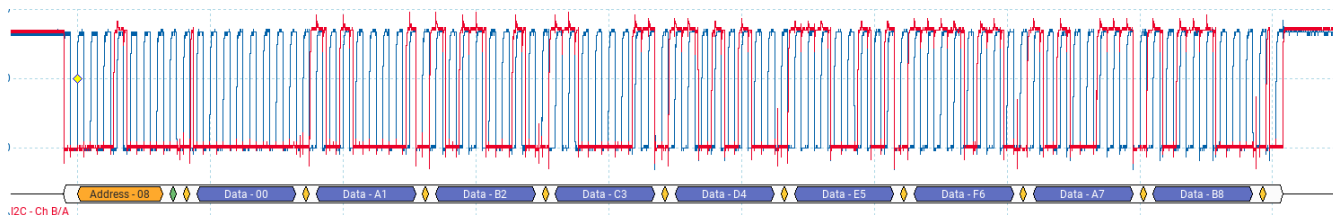


Figure 11. Result of the writing a sequence of word from an oscilloscope

An address of 0x0 is set from which the sequence of words will be read. After setting the address (Figure 12), a read operation is performed (Figure 13).

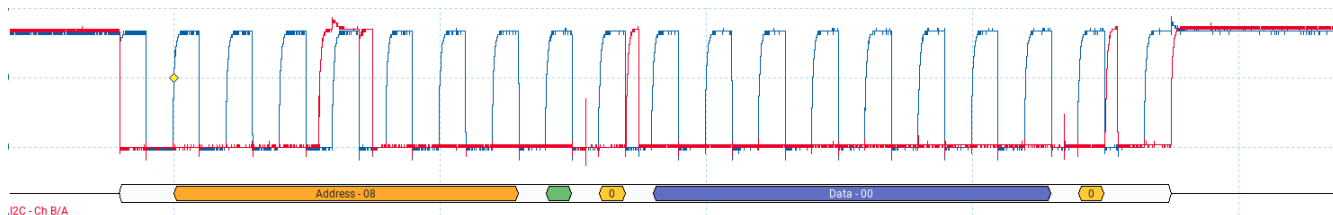


Figure 12. Setting the address for a write operation.

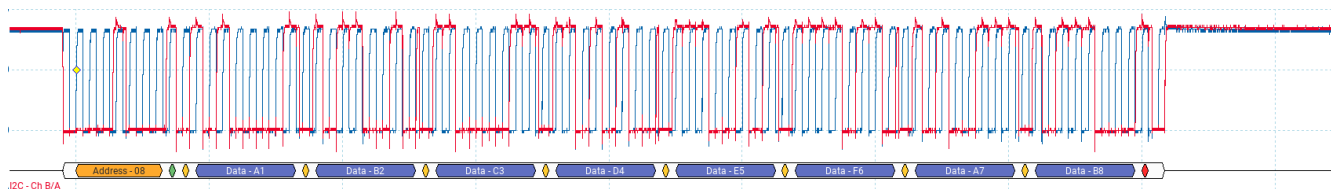


Figure 13. Result of the reading a sequence of words from an oscilloscope

5.3 Read Word operation on the Current Address

For this measurement, the starting address is set to 0x2 (Figure 14), after which a read operation on the current address is performed consecutively (Figure 15 to Figure 17).

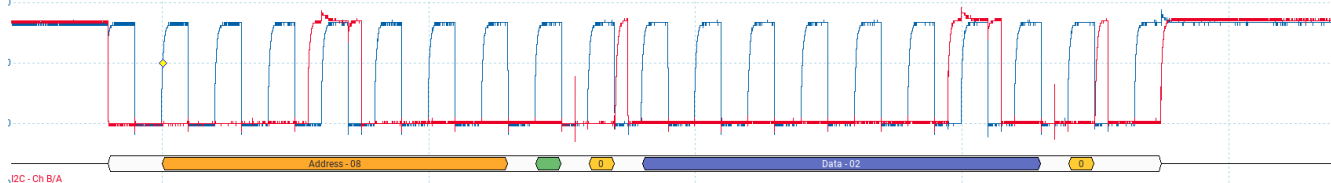


Figure 14. Setting the initialization address 0x2 (write operation)

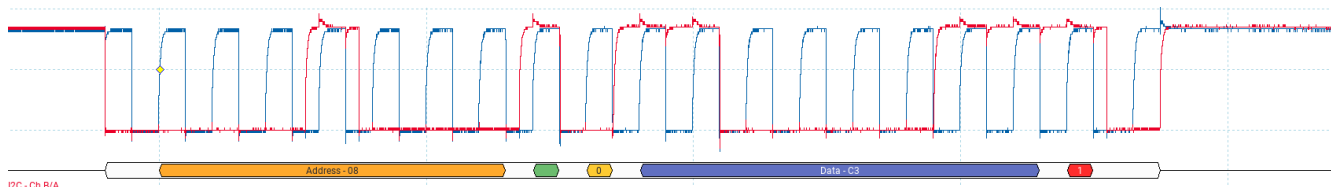


Figure 15. Reading data at address 0x2

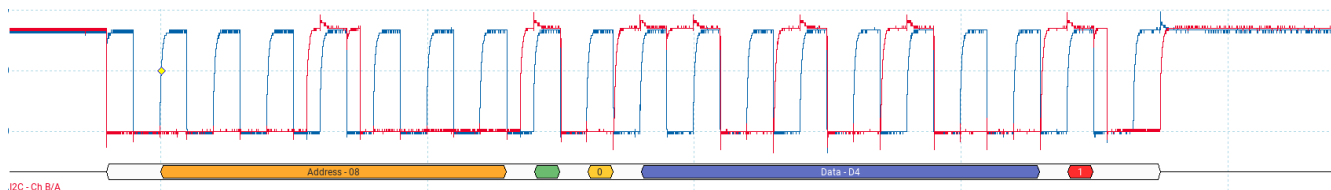


Figure 16. Reading data at address 0x3

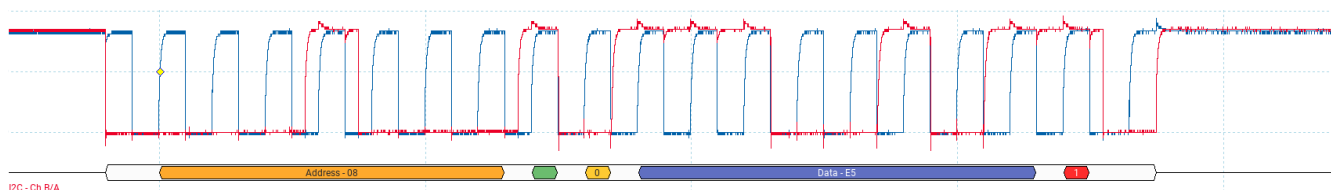


Figure 17. Reading data at address 0x4

As can be seen from the measurement results, the address is automatically incremented after each read operation. The data corresponds to the values from Table 1, since this sequence was previously written to memory.

6. Requirements

- Latest Revision of ForgeFPGA Workshop software

7. Verilog Code

The i2c_slave design is available for download ([AN-FG-020 How to design I2C Slave Module](#)). The top module (top_i2c_slave.v) contains two submodules (mem_data.v for memory and i2c_slave.v which is the I²C target from the Module Library). The Verilog code demonstrates how to connect different modules under one top module. The following shows the port definitions for the top module.

```
// Top module
(* top *) module top_i2c_slave #(
  parameter DATA_WIDTH = 8,
  parameter RAM_WIDTH = 8,
  parameter I2C_SLAVE_ADR = 7'h08 // Slave address, default value 7'h08 (Type - Hex, Default value = 7'h08,
                                   // Min value = 7'h01, Max value = 7'h7F)
) (
  (* iopad_external_pin, clkbuf_inhibit *) input wire i_clk, // input clock signal
  (* iopad_external_pin *) input wire i_rst,                // input reset signal
  (* iopad_external_pin *) input wire i_scl,                // input serial clock signal
  (* iopad_external_pin *) input wire i_sda,                // input serial data signal
  (* iopad_external_pin *) output wire o_sda,               // output serial data signal
  (* iopad_external_pin *) output wire o_sda_oe,            // output enable signal for serial data output signal
  (* iopad_external_pin *) output wire o_osc_en,            // output signal for enable OSC
  (* iopad_external_pin *) output wire o_gpio0_oe,          // setting up the gpio0 port
  (* iopad_external_pin *) output wire o_gpio5_oe           // setting up the gpio5 port
);

// -----
// Declaration signals
// -----
wire busy; // signal which indicates the operation state
wire [DATA_WIDTH-1:0] data_tx; // data inputs bus
wire [DATA_WIDTH-1:0] data_rx; // data outputs bus
wire int_tx; // signal which indicates that data_tx was sent
wire int_rx; // signal which indicates that data_rx was updated
wire en; // signal which start operation
reg [3:0] count_rst = 4'h0; // counter to reset after initialization
wire rst_comm; // common reset
wire rst_soft; // soft reset
```

8. Test Bench

The tb_top_i2c_slave.v test bench is used to verify that the DUT is working correctly. The module name must be given so that it can be recognized by the ForgeFPGA Workshop as a test bench file.

```
`timescale 1ns / 1ps
```

```
module tb_top_i2c_slave ();
// -----
// Localparam
// -----
localparam DATA_WIDTH = 8;
localparam RAM_WIDTH = 8;
localparam I2C_SLAVE_ADR = 7'h8; // Slave address, default value 7'h08 (Type - Hex, Default value = 7'h08,
                                   // Min value = 7'h01, Max value = 7'h7F)
localparam IN_CLK_PERIOD = 20;    // 50 MHz
localparam SCL_CLK = 10000;      // 100 KHz
// -----
// Declaration signals
// -----
reg i_clk;           // input clock signal
reg i_rst;           // input reset signal
reg i_scl = 1'b1;    // input serial clock signal
reg i_sda = 1'b1;    // input serial data signal
wire o_sda;          // output serial data signal
wire o_sda_oe;       // output enable signal for serial data output signal
reg i_scl_delay = 1'b0;
reg start_scl = 1'b0;
reg pause_scl = 1'b0;
wire [6:0] addr_ic;
reg [DATA_WIDTH-1:0] data_r_array [RAM_WIDTH-1:0];
// -----
// Clock
// -----
always begin
  i_clk = 1'b0;
  #(IN_CLK_PERIOD/2) i_clk = 1'b1;
  #(IN_CLK_PERIOD/2);
end
```

This functionality can be verified using the waveforms produced by the built-in GTKWave software.

Note: The complete code is available for download from the design file [\[2\]](#)

9. Simulation Results

The simulation results show screenshots of the main signals for the mem_data and i2c_slave modules.

Consider the situation of writing a sequence of 8 words from address 0x0. The sequence value is taken from Table 1.

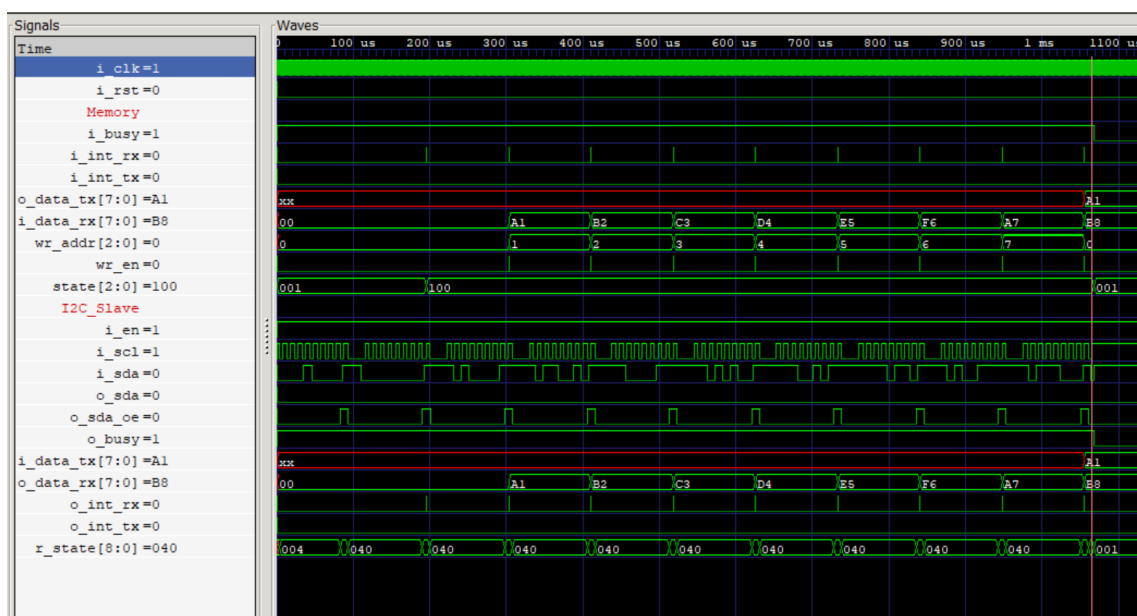


Figure 18. Result of the writing a sequence of words

Figure 18 shows that the data comes via the I²C bus, is converted into the i2c_slave module (o_data_rx signal), and then is transferred to the mem_data module. The o_int_rx signal indicates that 8 bits have been received. In the mem_data module, this signal is used to enable the writing of data to memory. The memory address (signal wr_addr) is incremented automatically with the arrival of a new i_int_rx signal. Also, the first 8 bits in the data transmission sets the current memory address.

The timing diagram results of writing a single word 0xAA at address 0x2 into memory are shown in Figure 19.

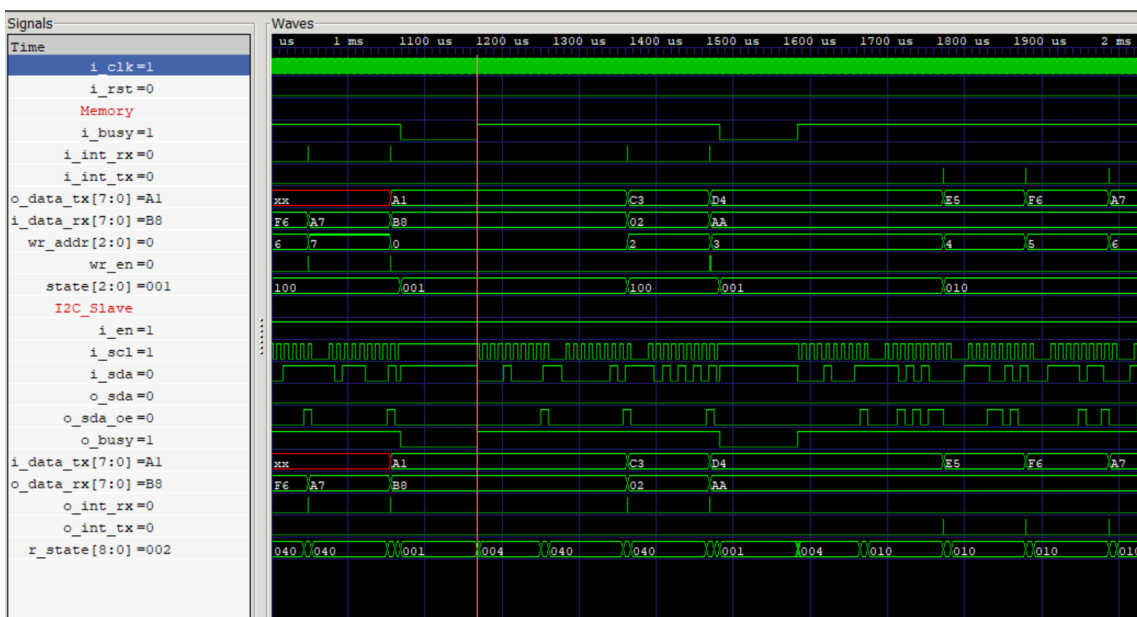


Figure 19. Results of the write word operation: Address - 0x2, Word - 0xAA

Upon receipt of the first pulse of the i_int_rx signal, the address for writing is set as 0x2 (i_data_rx signal). The second pulse of the i_int_rx signal writes the 0xAA (i_data_rx signal) data to the mem_data module.

Figure 20 shows the timing when reading a data sequence from the current address equal to 0x3. The marker marks the beginning of the sequence.



Figure 20. Reading a data sequence starting from address 0x3

The memory address (wr_addr signal) is incremented automatically after each arrival of the i_int_tx signal pulse in the mem_data module.

Figure 21 shows the timing when reading data one word at a time from the current address. The data for transmission is specified before sending to the i2c_slave module (i_data_tx signal).

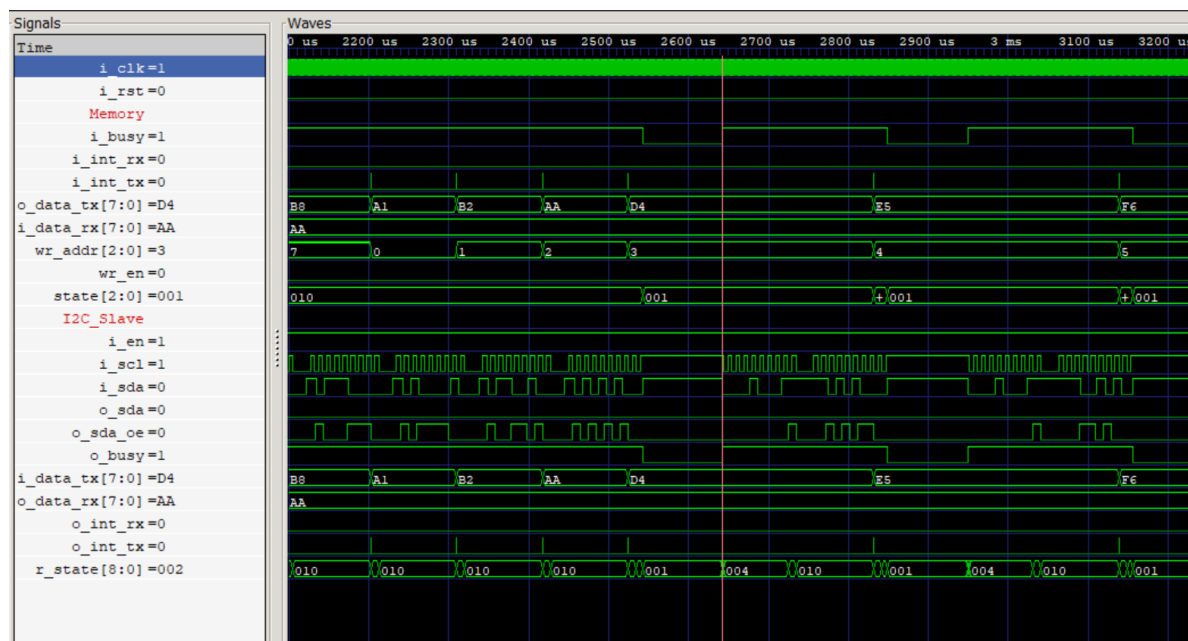


Figure 21. Reading data at address 0x3 and 0x4

Figure 22 shows the timing when changing the current memory address to 0x3 and then reading data from addresses 0x3 and 0x4 in one sequence.

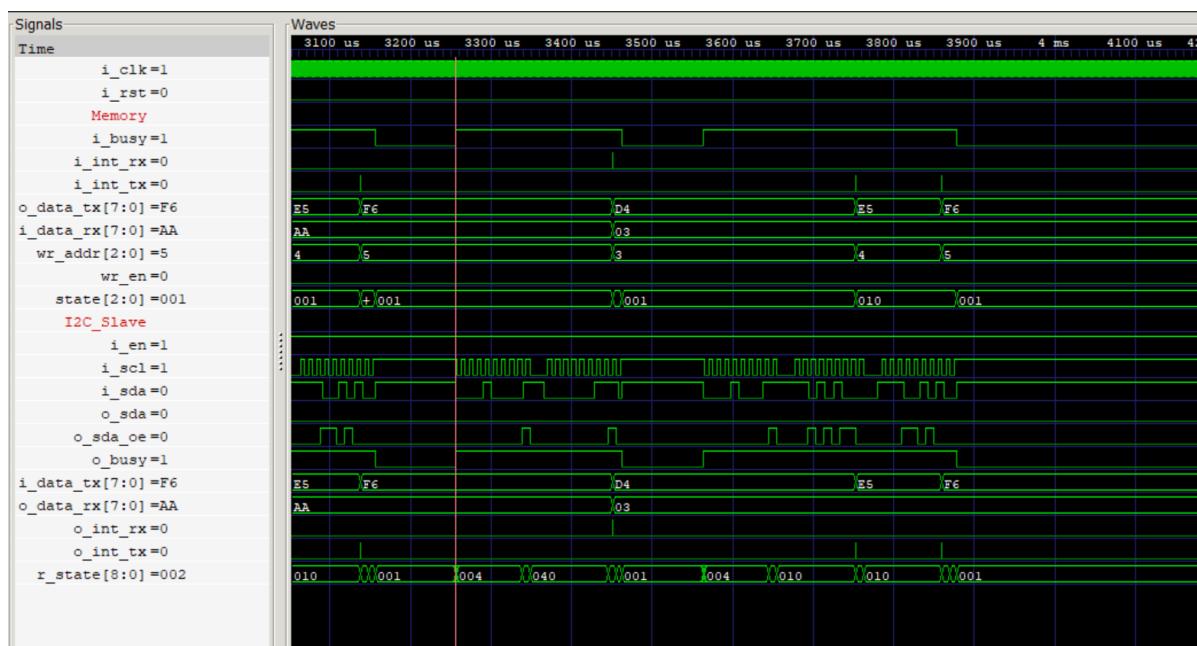


Figure 22. Reading data by changing the current memory address to 0x3

The first sequence on the I²C interface changes the current address, and the second sequence reads data with data incremented automatically.

10. Step by Step Design

1. Open the ForgeFPGA Workshop software in the Go Configure Software Hub and select the SLG47910 device. Create a project and select the FPGA Editor Tab.
2. Select *Module Library Tool* → *Communication* → *I2C* → *I2C Slave*. After the module parameters are configured, the testbench and module name are selected.

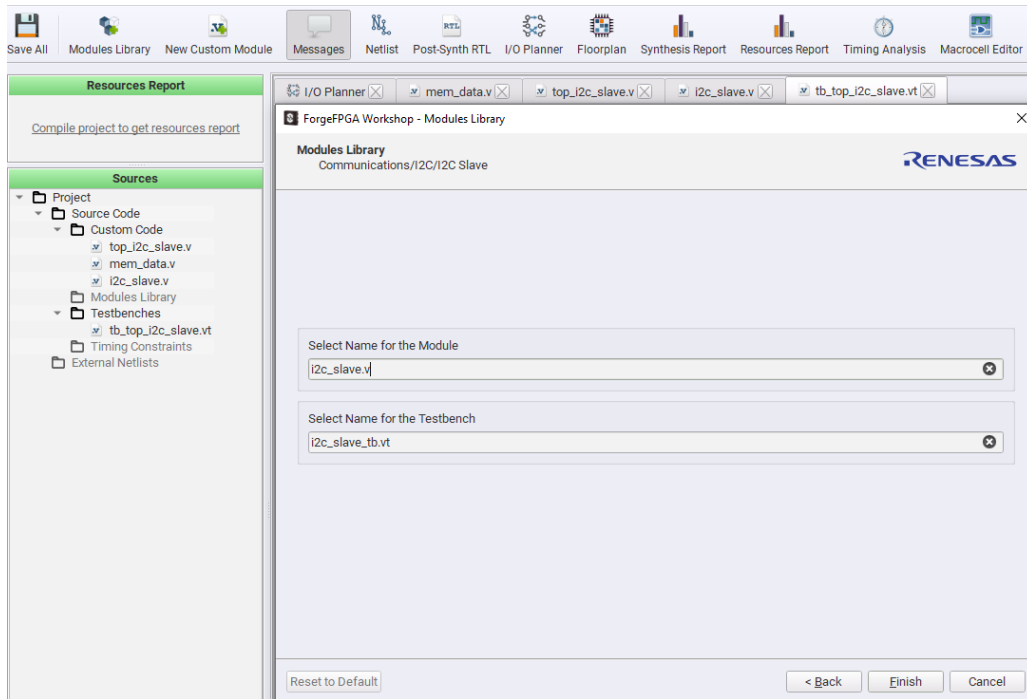
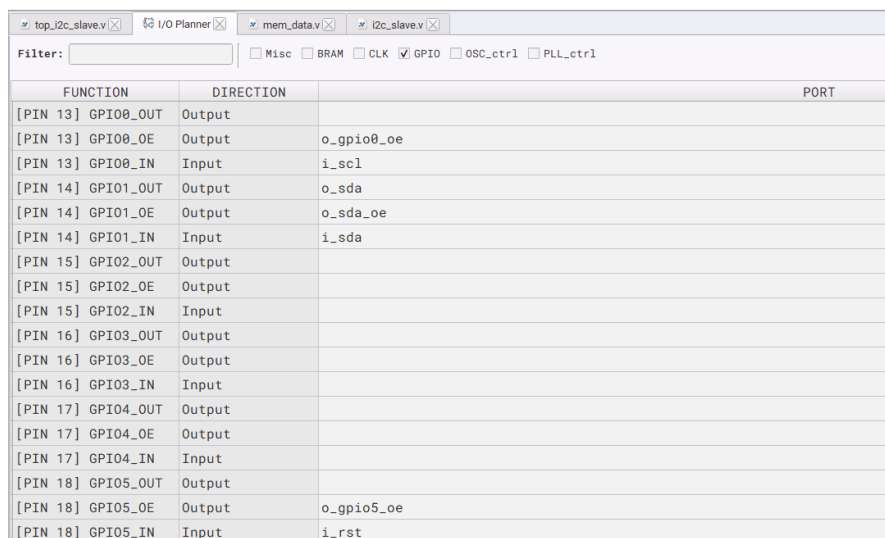


Figure 23. Module Library Tool

3. Create and add the *top_i2c_slave.v* and *mem_data.v* modules to the project. The complete code can be downloaded from the Renesas website: [AN-FG-020 How to design I2C Slave Module](#)
4. Open the I/O Planner in the FPGA editor and review the pin assignments.

Filter:	☐ Misc ☐ BRAM ☒ CLK ☐ GPIO ☒ OSC_ctrl1 ☐ PLL_ctrl1	
FUNCTION	DIRECTION	PORT
LOGIC_AS_CLK0	Input	
LOGIC_AS_CLK1	Input	
OSC_CLK	Input	i_clk
PLL_CLK	Input	
REF_BRAM(0..3)_REA...	Output	
REF_BRAM(0..3)_WRI...	Output	
REF_BRAM(4..7)_REA...	Output	
REF_BRAM(4..7)_WRI...	Output	
OSC_EN	Output	o_osc_en
OSC_READY	Input	

Figure 24. I/O Planner CLK and OSC_CTRL pin assignments



FUNCTION	DIRECTION	PORT
[PIN 13] GPIO0_OUT	Output	
[PIN 13] GPIO0_OE	Output	o_gpio0_oe
[PIN 13] GPIO0_IN	Input	i_scl
[PIN 14] GPIO1_OUT	Output	o_sda
[PIN 14] GPIO1_OE	Output	o_sda_oe
[PIN 14] GPIO1_IN	Input	i_sda
[PIN 15] GPIO2_OUT	Output	
[PIN 15] GPIO2_OE	Output	
[PIN 15] GPIO2_IN	Input	
[PIN 16] GPIO3_OUT	Output	
[PIN 16] GPIO3_OE	Output	
[PIN 16] GPIO3_IN	Input	
[PIN 17] GPIO4_OUT	Output	
[PIN 17] GPIO4_OE	Output	
[PIN 17] GPIO4_IN	Input	
[PIN 18] GPIO5_OUT	Output	
[PIN 18] GPIO5_OE	Output	o_gpio5_oe
[PIN 18] GPIO5_IN	Input	i_rst

Figure 25. I/O Planner GPIO

- Next click the Synthesize button on the lower left side of the FPGA editor. Click 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.
- Proceed to simulate the testbench using the built-in GTKWave software. As a part of the design file, the user can find a test bench file attached as well named tb_top_i2c_slave.v. This can be simulated by clicking the *Simulate Testbench* button on the toolbar at the top. This will open the GTKWave software automatically.
- Select the signals that need to be observed by simulation and reload it to view the waveform.

11. Conclusion

This application note shows how the I²C Target module operates from the Modules Library using a SLG47910. The operation of this module is shown using examples of writing/reading data to/from memory. There are also examples on how to work with sequential words. This test case is available for download ([AN-FG-020 How to design I2C Slave Module](#)). If interested, please contact the [ForgeFPGA Business Support Team](#).

12. Revision History

Revision	Date	Description
1.00	June 27, 2025	Initial release.