I²C Controller

SLG47921V

Abstract

This application note describes how to use an I²C Controller (also known as Master) to communicate with multiple devices using the Renesas ForgeFPGA SLG47920/21 device. For demonstration purposes, it displays temperature and relative humidity using an OLED display controlled by a SSD1306 driver and ATH20 digital temperature and humidity sensor.

This document is supported by design files listed in section 10 References.

Contents

Figures

1. Introduction

The goal of this design is to read environmental data from a AHT20 sensor and display the results on an OLED dot matrix driven by a SSD1306 controller using Verilog and the SLG47921 ForgeFPGA™, demonstrating the use of its finite-state machine for device initialization, data acquisition, and to display updates. This document provides an overview of the I²C protocol, details the operation of the ATH20 and SSD1306 devices, explains the data processing and display logic, and outlines the Verilog modules being used.

2. Requirements

- SLG47920/21V device
- ForgeFPGA™ Deluxe Development Platform with USB cable and power supply
- ForgeFPGA™ Socket Card #3
- Latest Revision of the ForgeFPGA™ Designer software
- ATH20 device
- 0.91 OLED display, SSD1306 device.

3. I²C Overview

I²C is a serial, half-duplex communication interface and protocol that uses two open-drain lines – the serial data line (SDA) and serial clock line (SCL) – to provide controller-target communication. Up to 127 target devices (also known as Slaves) can be connected to one I²C controller. Each I²C target device has a unique 7-bit address. The I²C controller can either read data from the target or write data to it. Standard operations are shown in Figure 1.

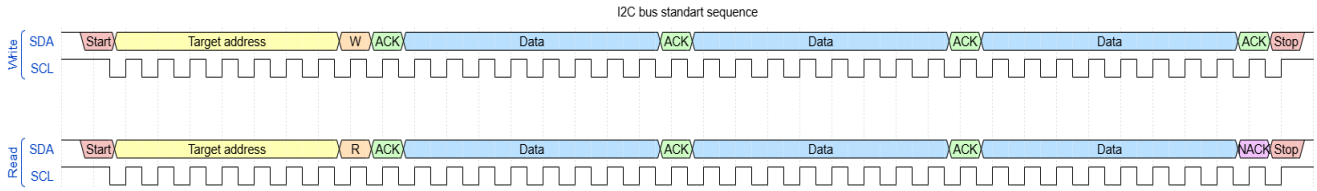

Figure 1. Standard I²C Operations

The controller module generates the serial clock, as well as the start and stop conditions.

The I²C controller initializes the start of the transaction, then sends the target address with which it wants to communicate and a R/W**Error! Bookmark not defined.** bit that indicates whether to read or write. Sending a logic ‘0’ indicates a write operation, otherwise a logic ‘1’ indicates a read operation. If the target is on the line and ready to communicate, it pulls the SDA line to the ground, indicating an Acknowledge (ACK) signal.

If the controller wants to write to the target, it sends data sequentially, byte-by-byte. After each byte is sent, it waits for the target’s response. If the target does not pull the SDA line to ground, it indicates a Not Acknowledge signal (NACK) and controller stops the transaction, otherwise, it continues sending data as needed.

After successfully sending all bytes, a Stop condition is set.

For a read operation, after sending the target address and receiving an ACK signal from the target, the controller starts receiving data. The target sends bytes until the controller no longer wants to receive them. In this case, NACK is either sent when the controller has received all the required data, or when some issue is detected. The SDA line can be changed only when the SCL line is set to a low level, except for a start/stop condition. The Start condition (S) is indicated by pulling SDA low while the SCL line is set high, while a Stop (P) condition is indicated by pulling SDA to a logic ‘1’ while SCL is set high. Data from the target can also be read from a specific subaddress. In this case, the controller sends the target address, write bit (logic ‘0’), and then the subaddress. After that, Start is set. After the start condition is repeated, the controller sends the target address and a read bit (logic ‘1’). This Read operation is shown in Figure 2.

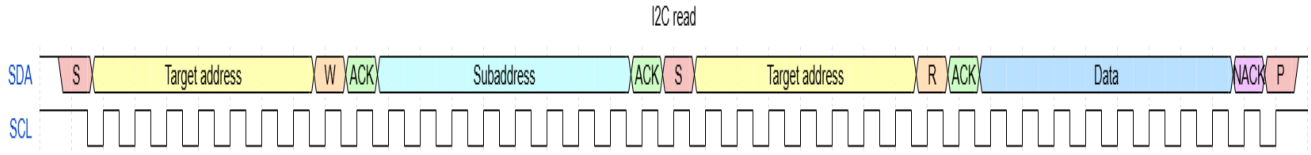

Figure 2. Read Operation

4. ATH20 Overview

The ATH20 is a digital temperature and relative humidity sensor that communicates via I²C. It uses the target address 0x38. To initialize and calibrate the sensor, a 0xBE command is issued. 0xAC is the command that triggers a measurement. Values can be read after an 80 ms wait. The sensor transmits 8 status bits, 20 humidity bits, 20 temperature bits, and 8 optional CRC bits. In this application example, a CRC is not used. Status bits indicate the sensor's state. If the state of the sensor is equal to 0x18, the sensor is calibrated and measurement is completed. The status word is located at subaddress 0x71. ATH20 calibration should be performed once after startup.

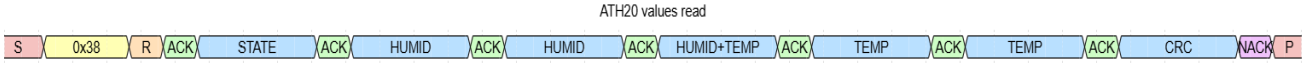

Figure 3. ATH20 Read Format

The sensor requires a 40 ms wait after power-on and a 10 ms wait after the initialization command (0xBE) to fully calibrate. Measurement is done in 80 ms, but to ensure the completion of the measurements, waiting an extra 10 ms (or checking the status register) is required. The ATH20 sends raw data that needs to be calculated and the calculation formulas are provided in section 6 Values Calculation.

5. SSD1306 Overview

The SSD1306 is an OLED/PLED display driver that supports I²C, SPI, and 6800/8080 Parallel interface protocols. For I²C communication, the device uses the 0x3C target address. This driver supports dot matrix panel with a maximum resolution of 128 x 64. For this application, a 128 x 32 OLED dot matrix panel is used. The SSD1306 is controlled by a variety of commands that configure the driver's resolution, addressing mode, contrast, segment re-mapping, display inversion, offsets, etc. To distinguish between command packets and data packets, the I²C controller sends a control byte to identify the type of data that follows. A control byte can indicate either a single control/data byte or a stream of control/data bytes. There are four control bytes:

- 0x00 – Command stream
- 0x80 – Single command byte
- 0x40 – Data stream
- 0xC0 – Single data byte.

When using the 0x80/0xC0 control byte, only one following byte is considered as command/data, so another control byte must be sent if needed. When using the 0x00/0x40 control byte, all following bytes are considered as command/data until the end of the transaction.

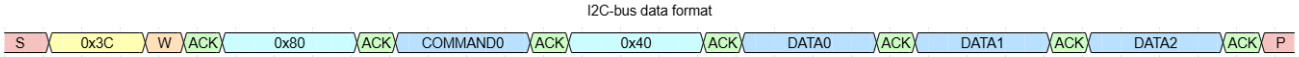

Figure 4. SSD1306 I²C-bus Data Format

The SSD1306 supports three different memory addressing modes: page addressing, horizontal addressing, and vertical addressing. In this application example, horizontal addressing is used, shown in Figure 5.

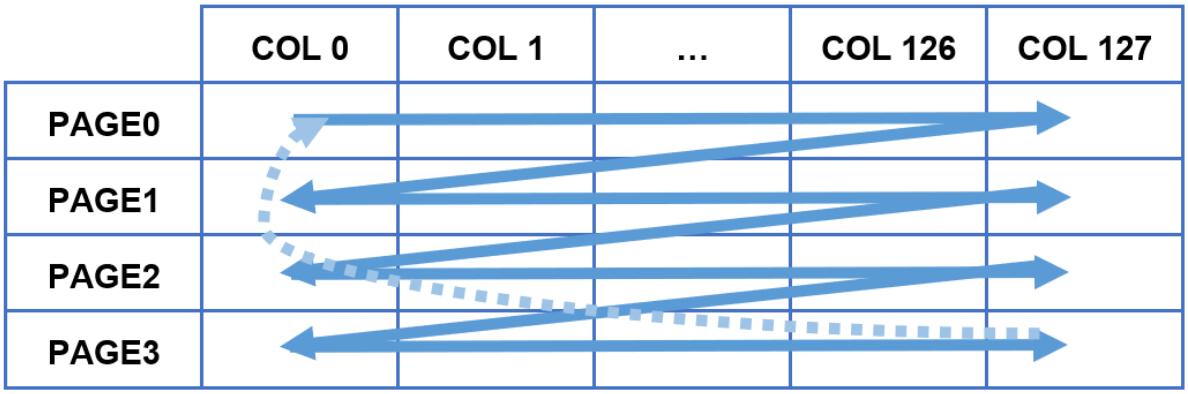

Figure 5. SSD1306 Horizontal Addressing

To initialize an OLED dot panel, a set of configuration commands must be sent. Following that, screen output data must be sent or else the panel will display a random pixel set stored in the display RAM. Data is sent sequentially, byte-by-byte. One byte represents a one page-width column as shown on Figure 6. To fully fill the panel, 512 (128*4) bytes are sent. In this application example, the screen initializes only once through the command stream (0x00), and data is loaded occasionally through the data stream (0x40). For example, to display number 20 as shown on Figure 7, 11 bytes must be sent to the driver: 0x40, 0x00, 0x61, 0x51, 0x49, 0x47, 0x00, 0x3E, 0x41, 0x41, and 0x3E. All other segments will be random if data sending is stopped.

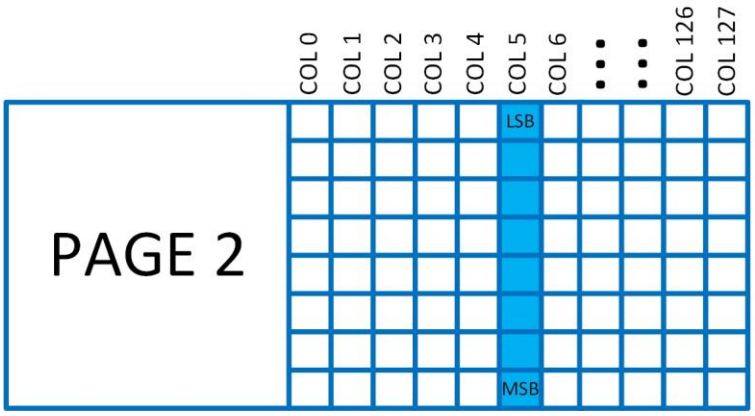

Figure 6. SSD1306 Single Byte Representation

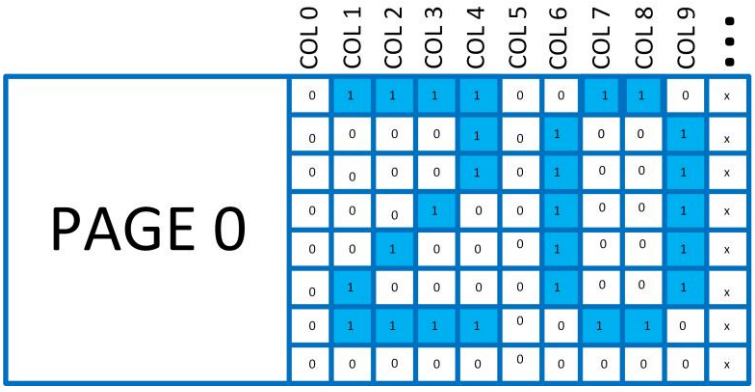

Figure 7. SSD1306 Digits Bitmap

6. Values Calculation

The ATH20 sends raw data that needs to be calculated. The relative humidity and temperature are transformed by two different formulas.

$$RH(\%) = (\frac{S_{RH}}{2^{20}}) \times 100\ \%$$

$$T(°C) = (\frac{S_T}{2^{20}}) \times 200 - 50$$

Both formulas require division and multiplication which is done using shifting to avoid multipliers in the FPGA logic.

6.1 Humidity Calculation

According to the formula, data received should be divided by 2^{20} which is equal to a 20-bit right shift (RSH). The incoming data is a 20-bit value, so this operation will result in all-zeroes. To prevent this, input data should be extended by adding extra zeros at the end of the value. Multiplying by 100x can be done by the shift-add multiplication algorithm. In this case, 100x = 64x + 32x + 4x. Multiplication by 64 is equal to a 6-bit left shift (LSH), multiplication by 32x is equal to a 5-bit LSH, and multiplication by 4x is equal to a 2-bit LSH. After shifting, these values are added together. The sensor has a high resolution that is excessive for this application. Values are stored in fixed-point format, so only the required bits are used.

6.2 Temperature Calculation

Similarly to the humidity calculation formula, data received should be extended by adding extra zeros and divided by 2^{20} . Multiplication by 200x is also done using the shift-add multiplication algorithm with the formula 200x = 128x + 64x + 8x. Multiplication by 128x is equal to a 7-bit LSH, multiplication by 64x is equal to a 6-bit LSH, and multiplication by 8x is equal to a 3-bit LSH. After shifting and summing, subtraction by 50 is done to finally convert the value to Celsius. As in humidity calculation, only the required bits are used.

7. Implementation

Verilog implementation (see Figure 8) controls the I²C module, configures the sensor and display, triggers measurements, reads and calculates values, and displays these values. For these purposes, the implementation consists of a control finite state machine, an I²C controller, a calculation unit, and a decode unit that prepares the values before displaying them. The Verilog code can be downloaded from section 10 References. The full set of configuration commands for both the sensor and display driver are stored in BRAM. This is done using BRAM initialization and can reduce FPGA logic utilization.

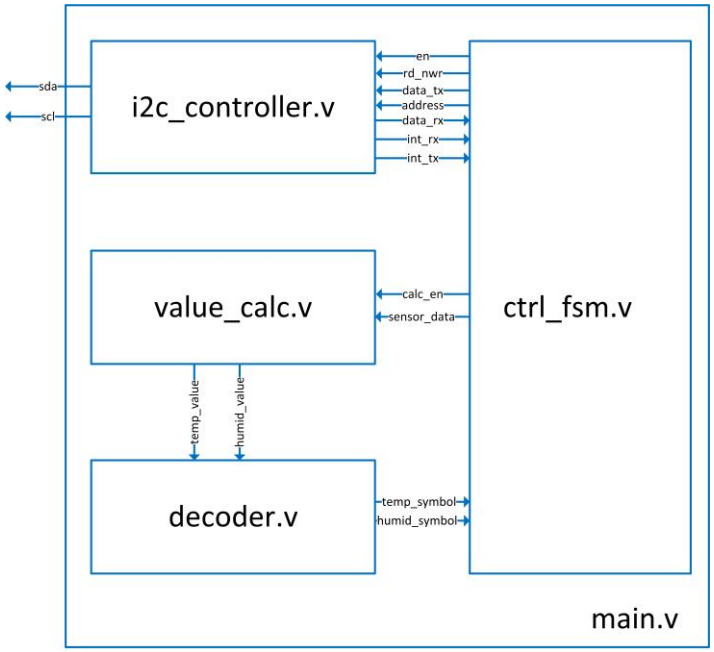

Figure 8. Verilog Implementation

7.1 Control Finite State Machine Module

This module performs all control functions of the design. The FSM controls the I²C address, data to transmit, and read/write signal. It enables the calculation process, sets the data to calculate, as well as the data to display. The State Machine has 11 states:

- POWER_UP
- SENSOR_INIT
- WAIT_10_MS
- SENSOR_TRIG
- WAIT_90_MS
- SENSOR_READ
- WAIT_10_MS_2
- SCREEN_INIT
- SCREEN_DATA_PREP
- SCREEN_DATA_LOAD
- IDLE.

The start state is POWER_UP, and the device remains in this state for 40 ms to completely power up the SSD1306 and the ATH20.

After that, it transitions to the SCREEN_INIT state, where it sends configuration commands to the display driver (0x3C).

After sending 33 configuration bytes (1 control byte + 32 commands), the device moves to the WAIT_10_MS state, where it remains in this state for 10 ms and then transitions to the SENSOR_INIT_SCREEN state, where it sends the sensor (0x38) a calibration command.

After that, the device transitions to the WAIT_10_MS_2 state, where it remains for and additional 10 ms for sensor calibration.

From the WAIT_10_MS_2 state, the device moves to the SENSOR_TRIG state. In this state, the module sends a measurement trigger command to the ATH20.

Next, it transitions to the WAIT_90_MS state where the device must wait for 90 ms.

After waiting 90 ms the device reads sensor data during the SENSOR_READ state and then transitions to the SCREEN_DATA_PREP state, where it calculates the received values and transforms them into a form suitable for display.

The device stays in this state for 10 ms and next transitions to the SCREEN_DATA_LOAD state, where it remains until data is fully sent to the screen.

Following this SCREEN_DATA_LOAD state, the device transitions to the IDLE state where the device remains idle for 10 ms and then finally transitions to the SENSOR_TRIG state to trigger measurements.

The state diagram is shown in Figure 9.

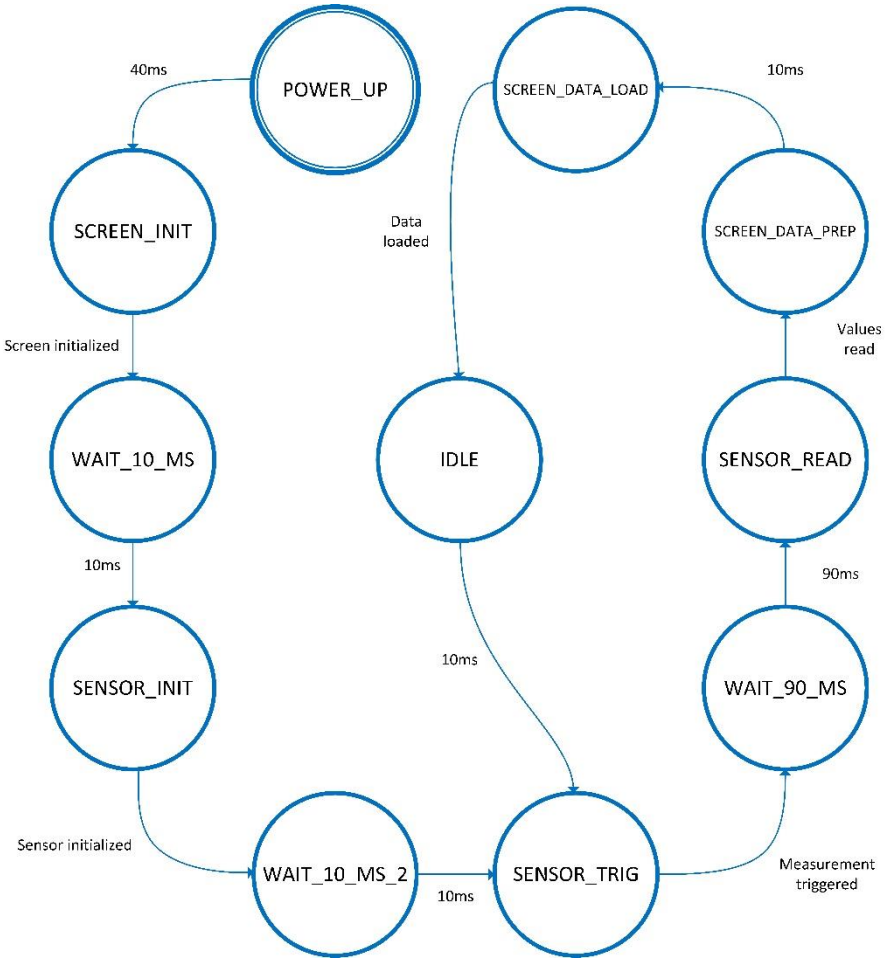

Figure 9. FSM Diagram

The code snippet for the state machine's transitions is shown below.

```
always @* begin
    r_next = r_state;
    case (r_state)
      POWER_UP: begin
        if (r_40_ms)              r_next = SCREEN_INIT;
        else                      r_next = POWER_UP;
      end
      SENSOR_INIT: begin
        if (r_command_cnt == 2)   r_next = WAIT_10_MS_2;
        else                      r_next = SENSOR_INIT;
      end
      WAIT_10_MS: begin
        if (r_10_ms)              r_next = SENSOR_INIT;
        else                      r_next = WAIT_10_MS;
      end
      SENSOR_TRIG: begin
        if (r_command_cnt == 2)   r_next = WAIT_90_MS;
        else                      r_next = SENSOR_TRIG;
      end
      WAIT_90_MS: begin
        if (r_90_ms)              r_next = SENSOR_READ;
        else                      r_next = WAIT_90_MS;
      end
      SENSOR_READ: begin
        if (r_data_cnt == 5)      r_next = SCREEN_DATA_PREP;
        else                      r_next = SENSOR_READ;
      end
      WAIT_10_MS_2: begin
        if (r_10_ms)              r_next = SENSOR_TRIG;
        else                      r_next = WAIT_10_MS_2;
      end
      SCREEN_INIT: begin
        if (r_command_cnt == 33)  r_next = WAIT_10_MS;
        else                      r_next = SCREEN_INIT;
      end
      SCREEN_DATA_PREP: begin
        if (r_10_ms)              r_next = SCREEN_DATA_LOAD;
        else                      r_next = SCREEN_DATA_PREP;
      end
      SCREEN_DATA_LOAD: begin
        if (r_pixel_cnt == 512)   r_next = IDLE;
        else                      r_next = SCREEN_DATA_LOAD;
      end
      IDLE: begin
        if (r_10_ms)              r_next = SENSOR_TRIG;
        else                      r_next = IDLE;
      end
      default: begin
                                  r_next = POWER_UP;
      end
      endcase
  end
```

7.2 Unit Calculation Module

This module is used to calculate both the temperature and humidity values. It has a ‘calculate enable’ input signal and an ‘input data’ signal. Both are set by the FSM. This module outputs the calculated temperature and humidity values to the decode unit. The FSM sends raw sensor data without state and CRC bytes using the calculation described in section 6 Values Calculation. Since division and multiplication are done sequentially and both are done by shifting, a specific twelve bits are chosen, and the other bits are set to zero, depending on the multiplier.

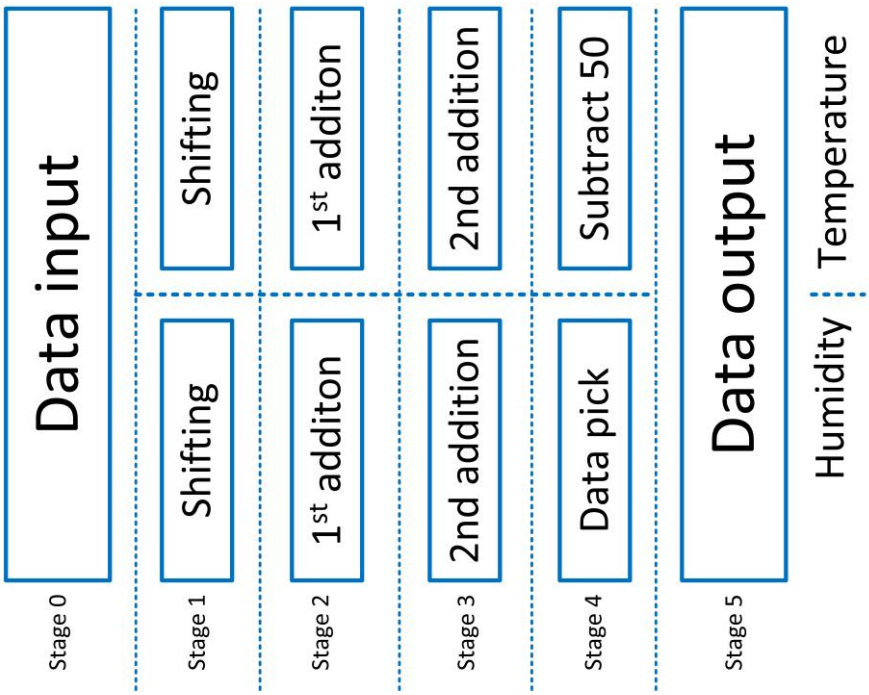

Figure 10. Calculation Steps

The following shows the port definitions for the calculation module.

```
module value_calc(
  // common ports
  input         i_clk,
  input         i_rst,
  // ctrl ports
  input         i_calc_en,
  input  [39:0] i_sensor_data,
  // result ports
  output [7:0] o_temp_value,
  output [7:0] o_humid_value
);
```

7.3 Decode Unit Module

This module is used to decode the calculated data into a form suitable for display. This module has two inputs and two outputs. The decoder receives binary values that should be displayed from the calculation unit and outputs the decoded data for the FSM, transforming digit values into a bitmap.

To display one digit, four bytes are used as shown in Figure 6 and Figure 7. The binary values received are first transformed into binary-coded decimal format (BCD), making decoding easier. Using this method only ten bitmaps for digits (from 0 to 9) are stored. During the SCREEN_DATA_PREP state, the decode unit transforms two numbers (four digits) to bitmaps, and these bitmaps are sent to the FSM.

The following shows the port definitions for the decoder module.

```
module decoder(
  // common porst
  input         i_clk,
  input         i_rst,
  // ctrl ports
  input  [7:0]  i_temp_value,
  input  [7:0]  i_humid_value,
  // result ports
  output [63:0] o_temp_symbol,
  output [63:0] o_humid_symbol
);
```

7.4 Results

The result of this application example is shown in Figure 11. The OLED display shows the environmental data acquired from the sensor with data updated every ~135 ms. The serial clock line is connected to GPIO0, and the serial data line is connected to GPIO1, as shown on Figure 12. Resource utilization is shown in Figure 13 and Figure 14 through Figure 18 show the standard I²C commands used. Figure 19 shows the testbench results of the i2c_controller. v module.

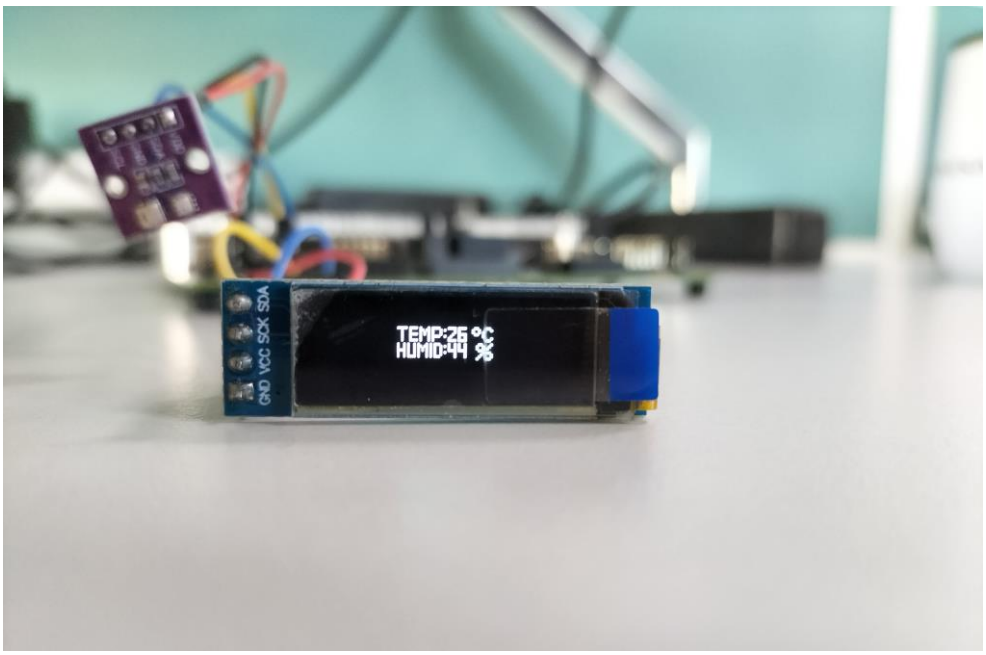

Figure 11. Application Result

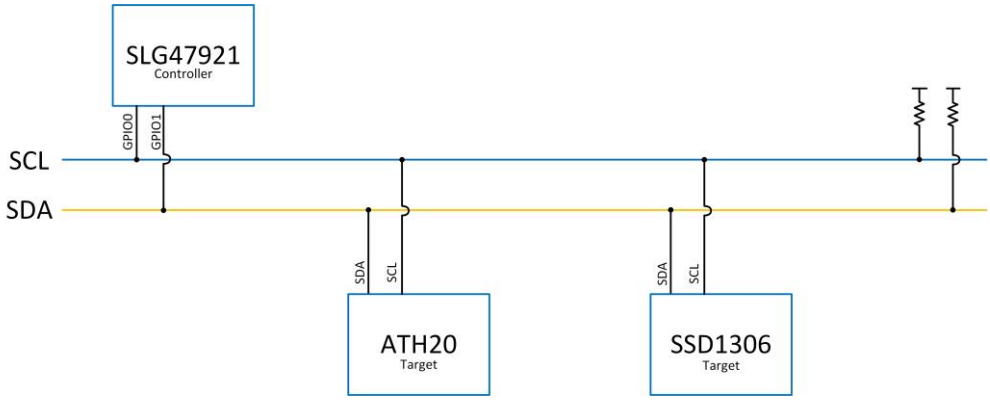

Figure 12. Connection Diagram

Resources Report			
Resource	Utilized	Available	% Utilized
CLB LUT5s	604	2240	26.96
FFs	383		
CLB FFs	373	2240	16.65
IOB FFs	10	1248	0.80
CLBs	111	280	39.64
Tiles	1	2	50.00
RTL Input ports	19		
RTL Output ports	22		
GPIOs	3	40	7.50
PLLs	1	2	50.00
OSCs	1	1	100.00
4k BRAMs	2	16	12.50

Figure 13. Resources Utilization

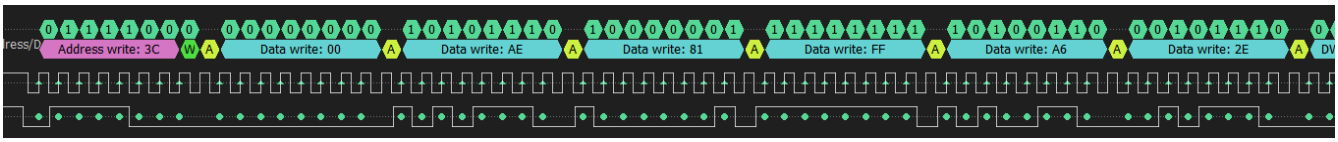

Figure 14. Screen Initialization

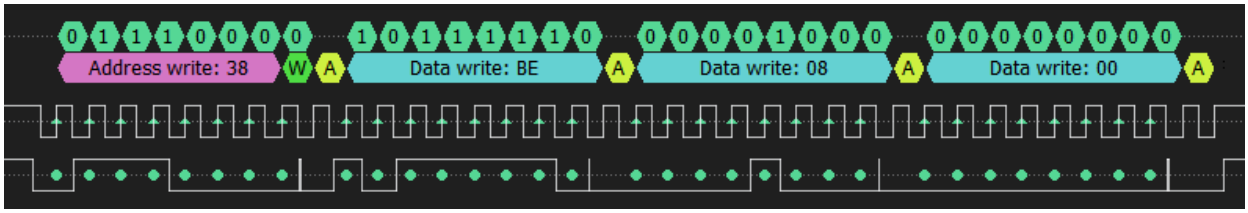

Figure 15. Sensor Calibration

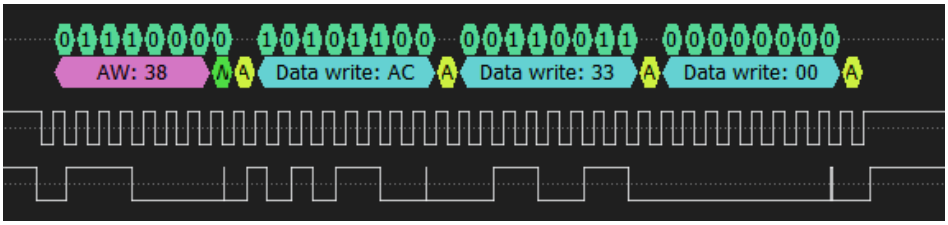

Figure 16. Measurement Trigger

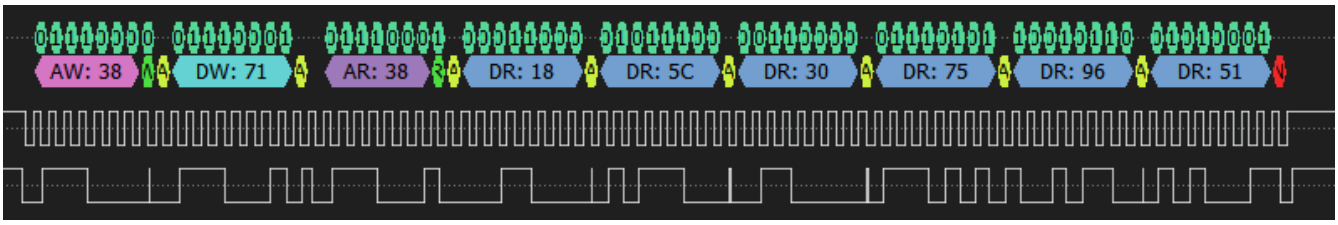

Figure 17. Data Acquisition

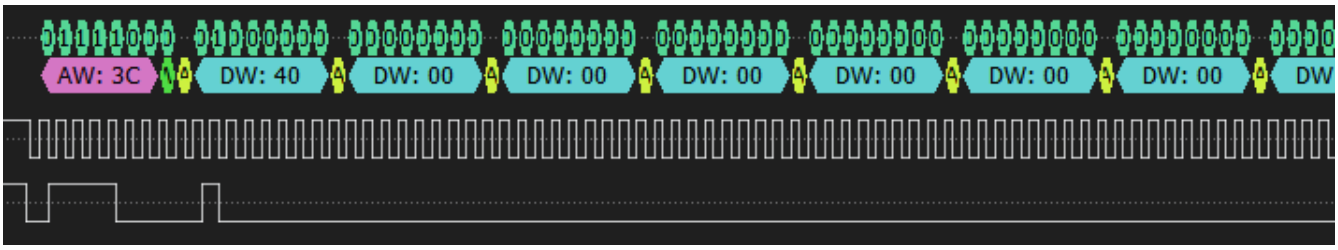

Figure 18. Screen Data Load

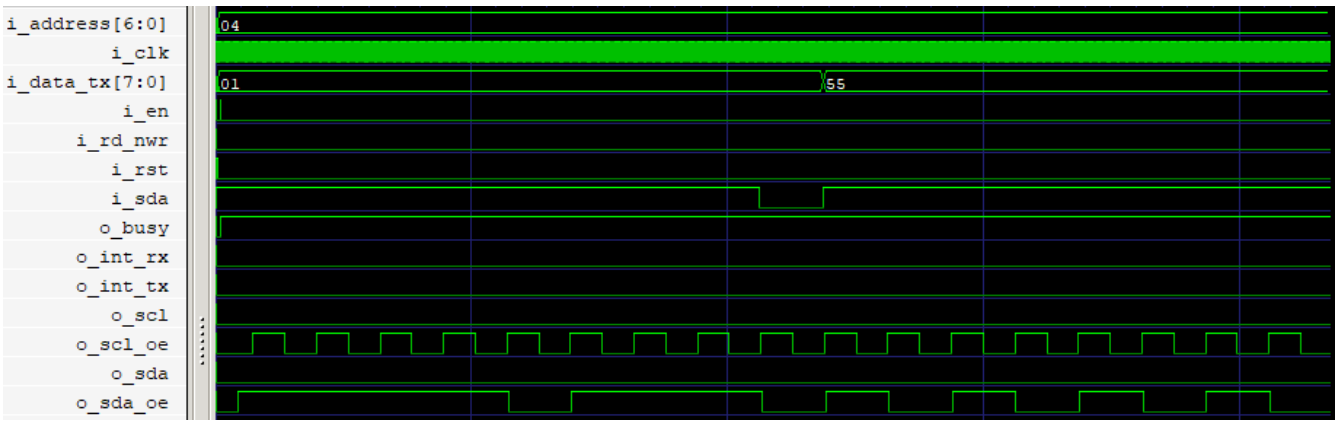

Figure 19. Testbench Results

8. Conclusion

This application note demonstrates how to use the SLG47920/21 ForgeFPGA™ to implement multi-device I²C communication between a temperature and humidity sensor and an OLED display.

It showcases temperature and humidity data acquisition using the ATH20 digital sensor, data processing, and display of this information using the SSD1306 OLED screen driver.

This design file is available for download in section 10 References.

For more information, contact ForgeFPGA Business Support Team.

9. Terms and Definitions

BCD	Binary Coded Decimal
BRAM	Block Random Access Memory
CRC	Cyclic Redundancy Check
FPGA	Field Programmable Gate Array
ForgeFPGA™	Renesas low-density FPGA
FSM	Finite-State Machine
GPIO	General Purpose Input Output
I^2C	Inter-Integrated Circuit
LSH	Left Logical Shift
OLED	Organic Light-Emitting Diode
PLED	Polymer OLED
RAM	Random Access Memory
RSH	Right Logical Shift
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface

10. References

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

Download our free ForgeFPGA™ Designer software [1] to open the .ffpga designs files [2] and view the proposed circuit design

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

[2] AN-FG-032 I2C Controller.ffpga, Design file, Renesas Electronics

[3] SLG47920/21, Datasheet, Renesas Electronics

[4] ForgeFPGA Workshop User Guide, Renesas Electronics

[5] Digital Temperature and Humidity Sensor ATH20 , User Manual

[6] OLED Dot Matrix Controller SSD1306 , User Manual

11. Revision History

Revision	Date	Description
1.00	Jan 20, 2026	Initial release