

Nexys4 DDR3 FPGA with Multiple Input Ports

Students:

Thanh Tien Truong (MS in Computer Engineer)

Hai Dang Hoang (MS in Computer Engineer)

Supervisor:

Dr. John Acken

Date:

September 17th, 2017

Table of Contents

Abstract	2
I. Introduction.....	3
II. Nexys4 FPGA Board Features.....	3
III. Block Diagram.....	4
IV. Data Validation.....	5
4.1. PDM microphone	5
4.1.1. General Information.....	5
4.1.2. Pulse Density Modulation.....	6
4.1.3. Validating data	6
4.2. Keyboard.....	7
4.2.1. General Information.....	7
4.2.2. PS2 Interface.....	8
4.2.3. Validating data	9
4.3. OV7670 VGA camera	11
4.3.1. General Information.....	11
4.3.2. SCCB Interface	11
4.3.3. Validating data	12
V. Hardware Setup.....	13
VI. Software Setup.....	15
VII. Appendix	17
7.1. Software Download and Tutorial.....	17
7.2. Bill of Material (BOM)	17
7.3. Additional Documents	17

Abstract

In system security or information security, authentication is the process of confirming a user's identity. For many years, knowledge-based authentication, possession-based authentication and biometric-based authentication are used to be strong authentications. However, those authentication's weaknesses have been exposed causing so many unauthorized accesses, uses, modifications or destructions to protected systems, data and information. This document provides information about the first stage in the process of implementing a new authentication mechanism and the first stage is to make sure that the central processor can receive data from multiple input sources.

I. Introduction

For many years, there have been three different types of authentication. Knowledge-based authentication uses “something you know” such as passwords or PINs, possession-based authentication uses “something you have” such as bank cards, student cards or key cards and biometric-based authentication uses “something you are” such as fingerprint, voice, or DNA. In fact, those types of authentications have weaknesses. PINs and passwords can be guessed or known by using a fast computer. Cards can be lost or stolen. Fingerprint can be cloned, voice can be recorded and DNA information can be retrieved from leaking database. Therefore, we proposed a new authentication mechanism which is implemented in the hardware level. The new authentication mechanism uses knowledge-based data, possession-based data and biometric-based data as primary inputs, and uses a LFSR to generate a unique key number so that the weaknesses of the traditional authentications are covered.

The implementation requires two stages which are making sure that valid data from multiple input sources can be received by a central processor and developing the LFSR to generate a unique key number. This document is all about the first stage. Specifically, section 2 shows the features of the central processor. Section 3 shows the block-diagram of the entire system. Section 4 shows how input data is validated. Section 5 and 6 show how users can setup the demo.

II. Nexys4 FPGA Board Features

A Nexys4 DDR3 FPGA board is used as a central processor that will receive data from other components.

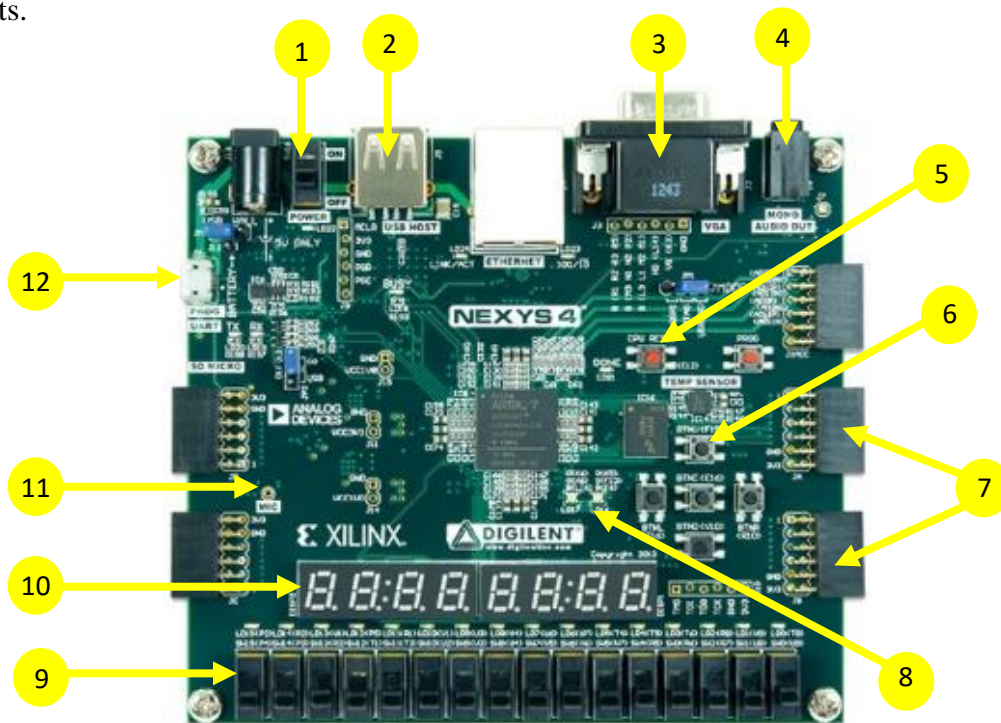


Figure 1: Nexys 4 DDR 3 FPGA Board



Number	Component Description
1	Power switch
2	USB host connector
3	VGA connector
4	Audio connector
5	CPU reset button (for soft cores)
6	Five pushbuttons
7	Pmod port(s)
8	RGB LEDs (2)
9	Slide switches (16)
10	Eight digit 7-seg display
11	Microphone
12	Shared UART/ JTAG USB port

Table 1: Nexys 4 DDR 3 FPGA Feature Descriptions

III. Block Diagram

For the input sources, a keyboard is used for knowledge-based authentication and Pulse Density Modulation (PDM) microphone is used for biometric-based authentication. Instead of using a scanner for possession-based authentication, an OV7670 VGA camera is used for another biometric-based authentication due to hardware limitation.

Each input source requires different way to communicate and retrieve data. In addition, after retrieving the data, the FPGA board should have dedicated memory places to save different type of data.

Input source	Important modules	Description
PDM microphone 	PDM controller	<ul style="list-style-type: none"> - Generating PDM clock signal - Capturing digital audio data
	Deserializer	Putting multiple serial data into an array
	DPRAM	Saving digital audio data
Keyboard 	PS2 controller	<ul style="list-style-type: none"> - Receiving PS2 clock signal - Capturing scan-code from PS2 data signal while a key is pressed
	ASCII converter	Converting scan-code to ASCII value
	DPRAM	Saving ASCII code of the pressed key


OV7670 Camera 	Serial Camera Control Bus (SCCB) controller	Configuring the camera through SCCB interface
	Data capture	Capturing the RGB value of a pixel through Digital Video Port (DVP)
	DPRAM	Saving RGB values of all the pixels

Table 2: Required modules for the input sources

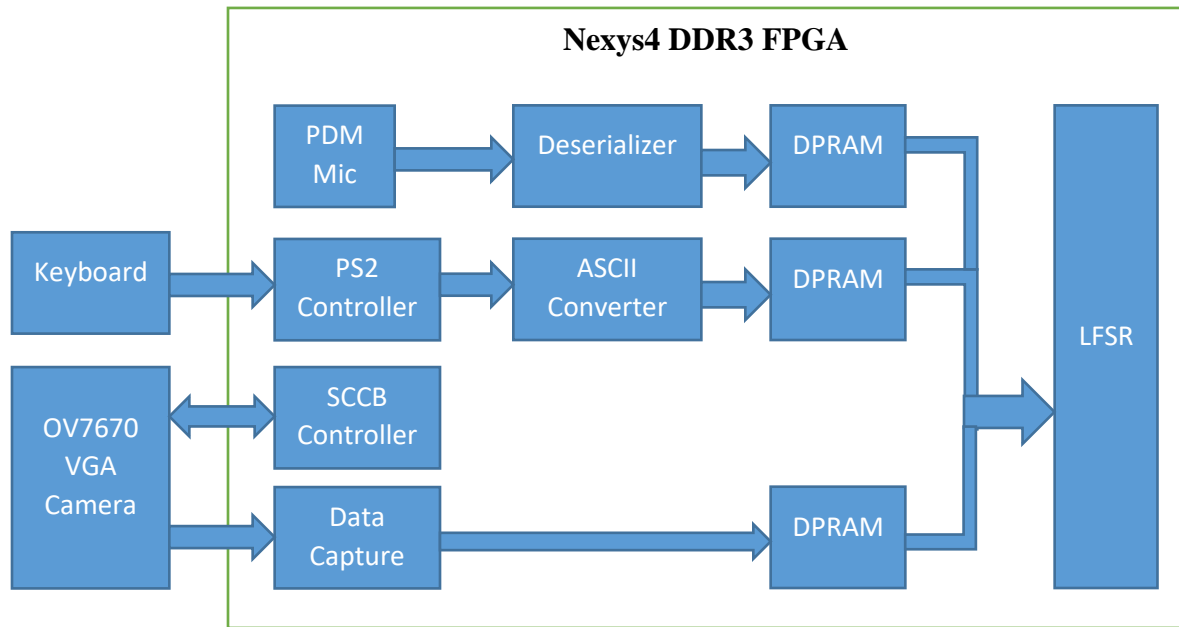


Figure 2: Block diagram of the prototype

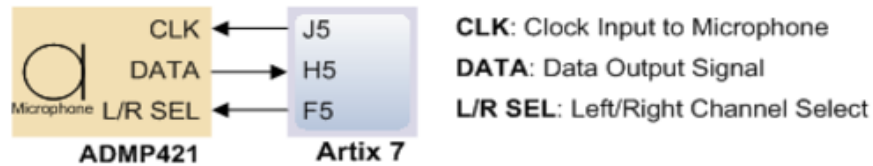
IV. Data Validation

This section describes how the Nexys4 DDR 3 FPGA board communicates with each input source and how the data be validated.

4.1. PDM microphone

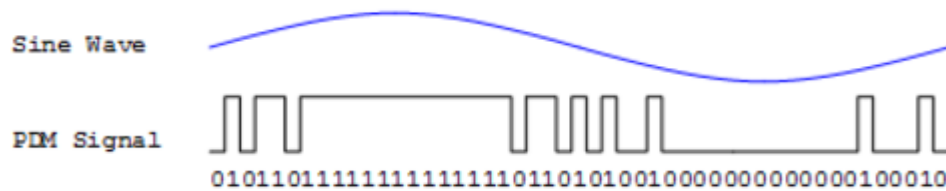
4.1.1. General Information

The Nexys4 DDR board includes an omnidirectional MEMS microphone. The microphone uses an Analog Device ADMP421 chip which has a high signal to noise ratio (SNR) and has a flat frequency response ranging from 100Hz to 15 KHz. The digitized audio is output in the pulse density modulated (PDM) format.



4.1.2. Pulse Density Modulation

PDM use only two wires to transmit data from two channels (left and right). The required frequency of a PDM signal is between 1 MHz to 3 MHz. In a PDM bitstream, a 1 corresponds to a positive pulse and a 0 corresponds to a negative pulse. A run consisting of all '1's would correspond to the maximum positive value and a run of '0's would correspond to the minimum amplitude value.



4.1.3. Validating data

In other to validate the PDM data, an on-board audio jack is used. Basically, the PDM data is considered to be valid if whatever sound that the microphone picks up can be heard through the on-board audio jack.

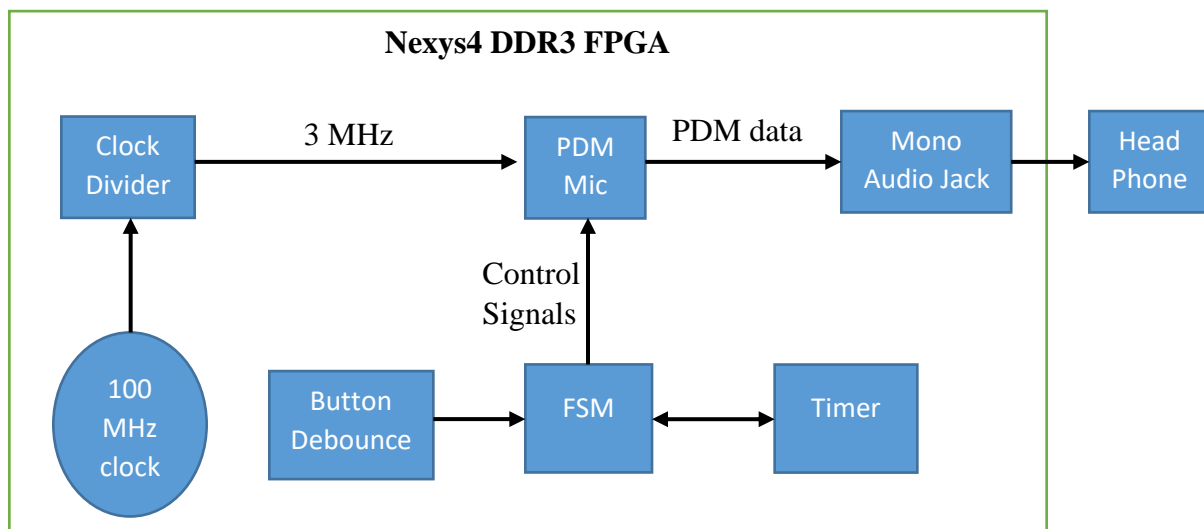


Figure 5: Block diagram for validating PDM data

On the Nexys4 DDR3 FPGA board, there are five pushbuttons. The center pushbutton is used to control the PDM microphone. Specifically, there is a Finite State Machine (FSM) to enable or disable the clock signal to the PDM microphone. The pushbutton will make the FSM go to the “ENABLE” state. In addition, there is also a Timer controlling the FSM. The Timer is activated when the FSM is in the “ENABLE” state. The timer will make sure that after 10 seconds, the FSM will disable the PDM microphone.

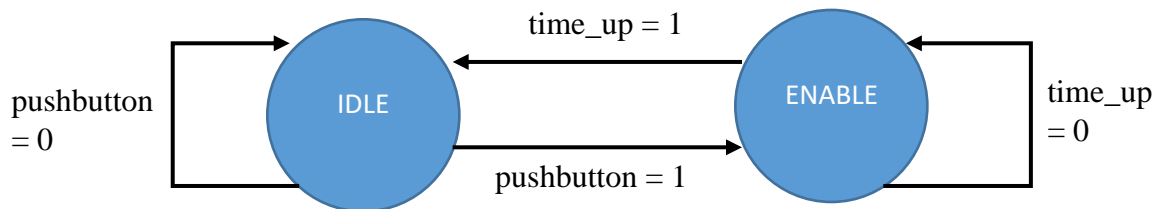


Figure 5: State diagram of FSM controlling PDM microphone

4.2. Keyboard

4.2.1. General Information

Nexys4 DDR3 FPGA has an auxiliary function microcontroller (microchip PIC24FJ128). The PIC24FJ128 provides the FPGA board with USB embedded Human Interface Device (HID) host capability so that the FPGA board can communicate with a keyboard or a mouse through USB port.

In fact, The PIC24FJ128 drives several signals into the FPGA – two are used to implement a standard PS/2 interface for communication with a mouse or keyboard, and the others are connected to the FPGA’s two-wire serial programming port, so the FPGA can be programmed from a file stored on a USB pen drive or microSD card.

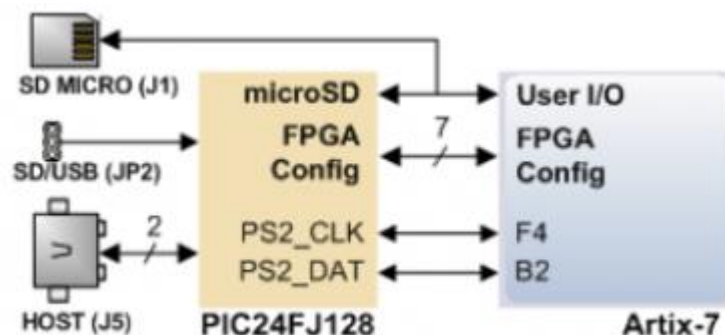


Figure 6: Nexys4 FPGA and PIC24FJ128 connections

4.2.2. PS2 Interface

The PS2 port contains two wires for communication purpose. One wire is for data, which is transmitted in a serial stream. The other wire is for the clock information, which specifies when the data is valid and can be retrieved. The information is transmitted as an 11-bit packet that contains a start bit (0), 8 data bits (Scan-code), an odd parity bit, and a stop bit (1)

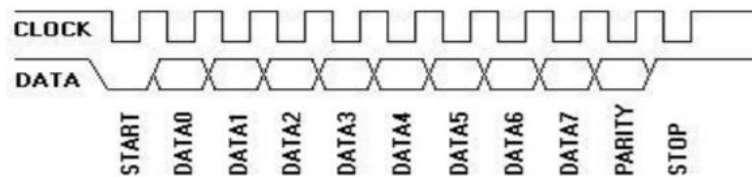


Figure 7: PS2 signals while transmitting an 11-bit packet

PS2 keyboard uses scan-codes to communicate key press data. Each key is assigned a code that is sent whenever the key is pressed. If the key is held down, the scan code will be re-sent repeatedly. When a key is released, an F0 key-up code is sent, followed by the scan-code of the release key. For example, when we press and release the A key, the keyboard first transmits its scan-code, followed by the break code and then its scan-code:

1C → F0 → 1C

If we hold the key down for a while before releasing it, the scan-code will be transmitted multiple times

1C → 1C → 1C → 1C → ... → 1C → 1C → F0 → 1C

If multiple keys can be pressed at the same time (Maximum is 2 keys). For example, we can first press the A key, and then the S key, and release the S key and then release the A key. The transmitted code sequence will be:

1C → 1B → F0 → 1B → F0 → 1C

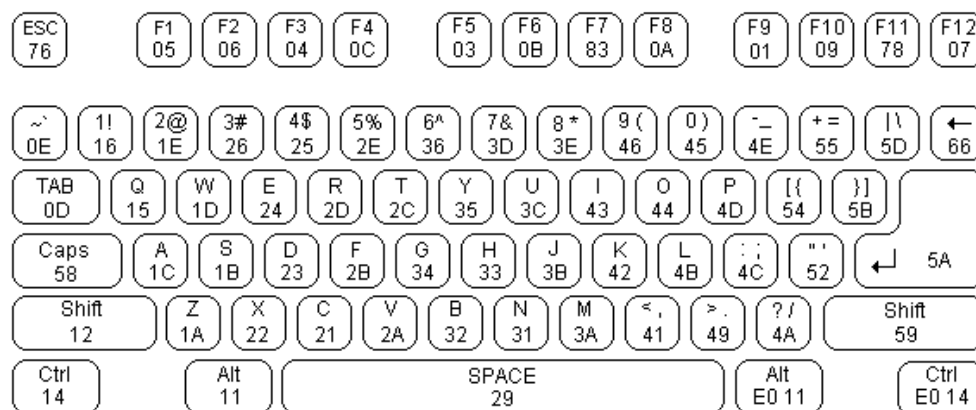
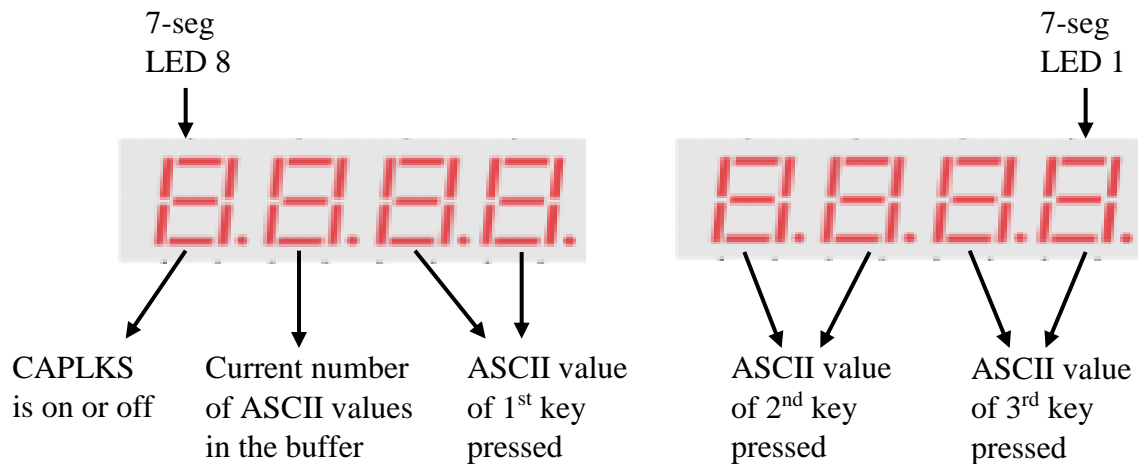


Figure 8: Keyboard scan-codes

4.2.3. Validating data

The 7-segment LEDs and the RGB LEDs will be used to validate the receiving scan-code.

- 7-segment LEDs change displayed values based on the pressed key.
- RGB LEDs change color based on the current stream of ASCII values



Below is the block diagram for validating data from keyboard:

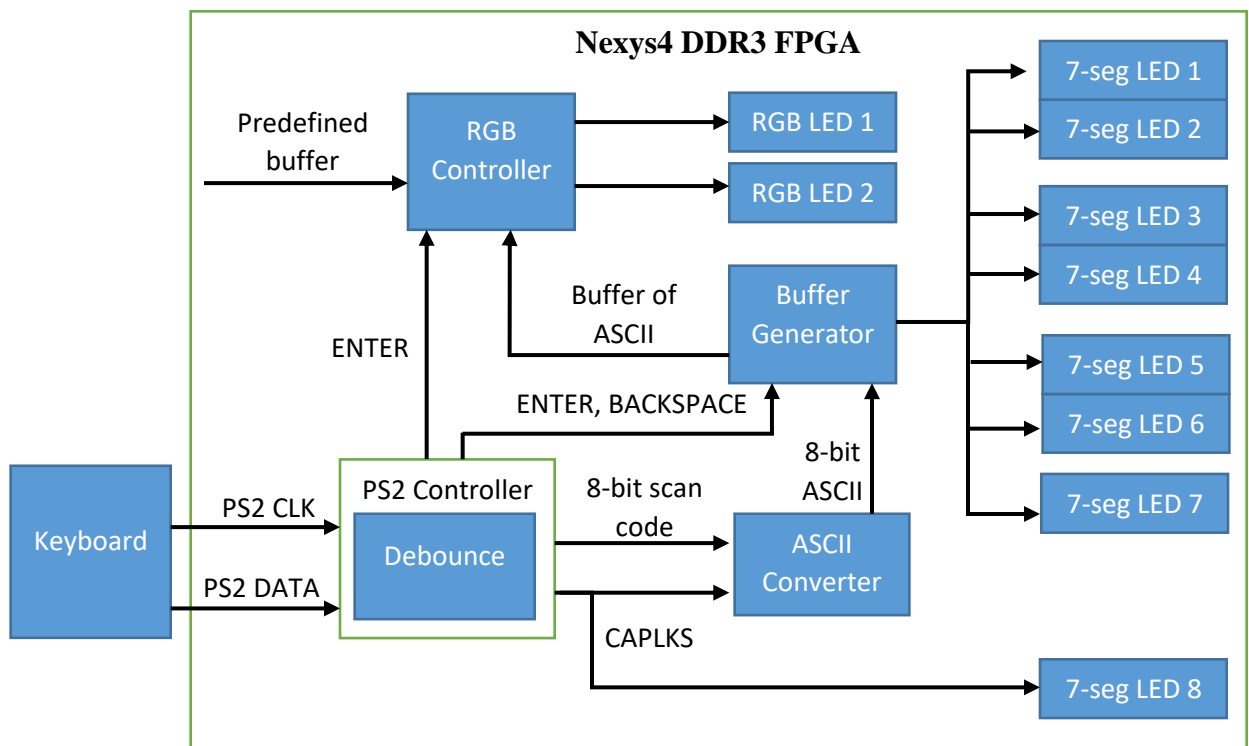


Figure 9: Block diagram for validating keyboard data

Responsibilities of each module:

PS2 Controller:

- Monitoring PS2 CLK and capturing data from PS2 DATA
- Generating scan-code of the pressed key and other control signals to other modules
- Setting the value for the 8th 7-segment LED

ASCII Converter:

- Monitoring the CAPLKS signal
- Converting the scan-code to ASCII value based on the CAPLKS signal

Buffer Generator:

- Monitoring BACKSPACE and ENTER signals
- Updating the buffer based on the BACKSPACE signal, ENTER signal and ASCII value
- Setting value for 7-segment LEDs

RGB Controller:

- Monitoring ENTER signal
- Defining color for 2 RGB LEDs after comparing the generated buffer to the predefined buffer

Note:

The predefined buffer is 0x61, 0x62 and 0x63 which is equivalent to “abc”.

The RGB LEDs will change color right after the ENTER is pressed. If the generated buffer has similar ASCII values to the predefined buffer, the RGB LEDs will turn green. Otherwise, they will turn red. In addition, the color won't change until there is different comparison result, for example, the RGB LEDs will stay red until the generated buffer has similar ASCII values to the predefined buffer.

4.3. OV7670 VGA camera

4.3.1. General Information

The OV7670 camera is a low voltage CMOS image sensor that provides the full functionality of a single-chip VGA camera and image processor in a small footprint package. The OV7670 camera is controlled through the Serial Camera Control Bus (SCCB) interface, supports Digital Video Port (DVP) to improve the speed of transferring data and supports multiple RGB, YUV and image scaling configurations.

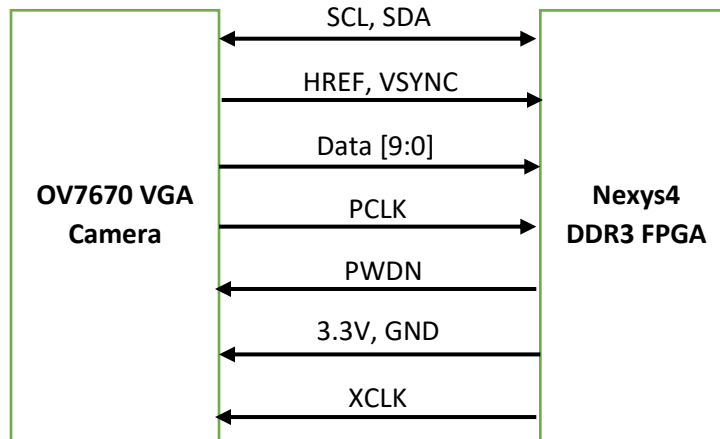


Figure 10: Connections between OV7670 VGA Camera and Nexys4 DDR3 FPGA

4.3.2. SCCB Interface

The OV7670 VGA Camera is configured through the SCCB Interface. The SCCB Interface is similar to I2C protocol. It requires only 2 signals SIOC and SIOD. There are START condition to start transferring data and STOP condition to stop transferring data. The START and STOP conditions are defined by the combination behaviors of SIOC and SIOD.

A START condition is indicated by a transition from tri-state (“Z”) to high (“1”) of SIOD, followed by a transition from high (“1”) to low (“0”) of SIOD. During those 2 transitions of SIOD, SIOC must be high (“1”).

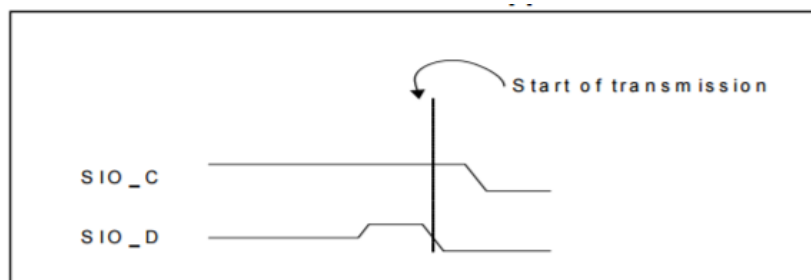


Figure 11: START condition for SCCB Interface

A STOP condition is indicated by a transmission from low (“0”) to high (“1”) of SIOD while SIOC is high

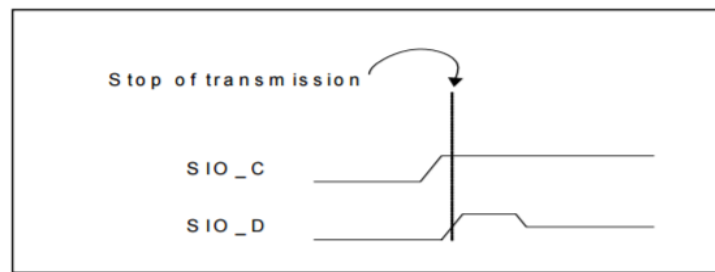


Figure 12: STOP condition for SCCB Interface

4.3.3. Validating data

In order to validate data received from the OV7670 VGA Camera, a VGA monitor is used to display all the captured frames.

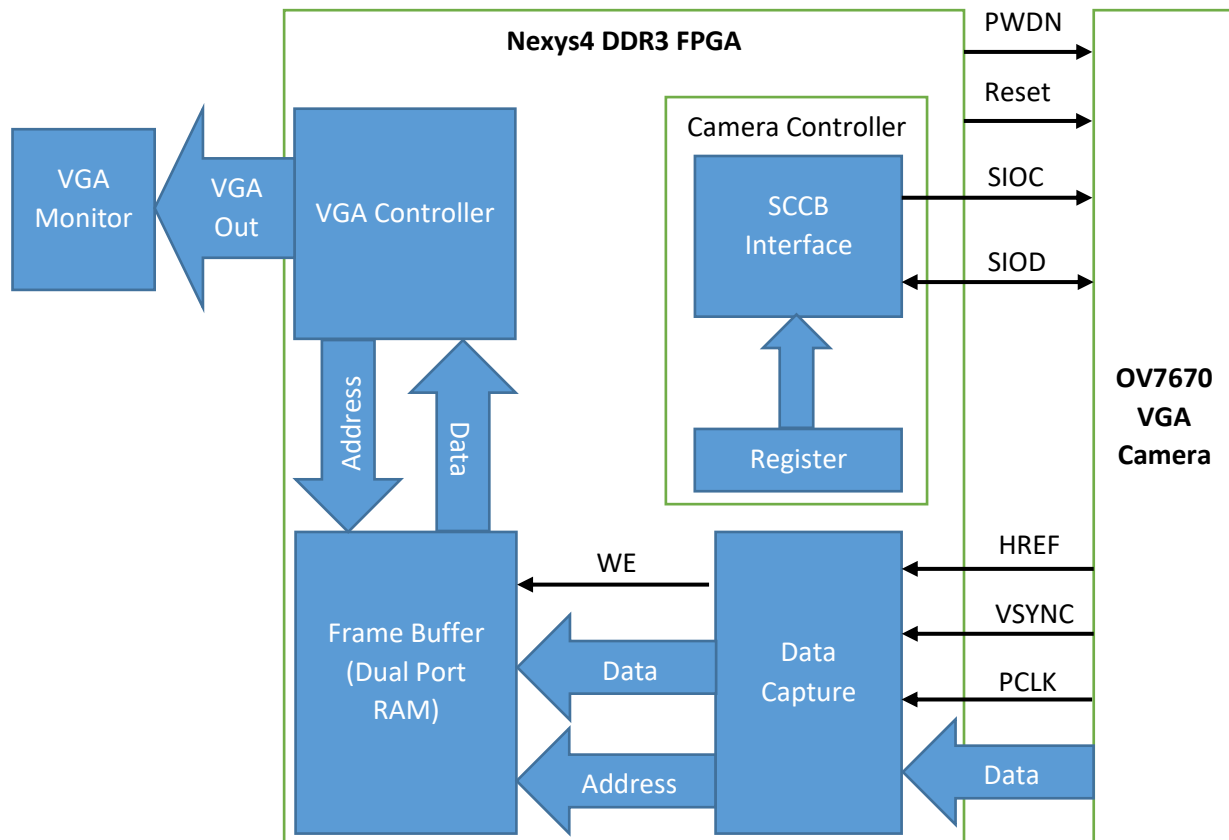


Figure 13: Block Diagram for validating OV7670 VGA Camera data

Responsibilities of each module:

Data Capture:

- Monitoring HREF, VSYCN and PCLK and capturing the data (RGB value) from DVP
- Controlling the WE signal and writing RGB values of a pixel into the memory

Register (Camera Controller):

- Containing data to configure the OV7670 VGA Camera. The data is in the 16-bit format where the 8 MSB is the register value and the 8 LSB is the data to be written into the register.

SCCB Interface (Camera Controller):

- Generating SIOC and SIOD signals based on the SCCB protocol to configure the OV7670 VGA Camera

Frame Buffer:

- Being a Dual Port RAM where RGB values of pixels of a frame are stored and read

VGA Controller:

- Reading RGB values from the Frame Buffer
- Generating HSYNC, VSYNC and RGB signals to VGA monitor (640 x 480 resolution)

V. Hardware Setup

This section shows users who to physically connect the input sources to the Nexys4 DDR3 FPGA board

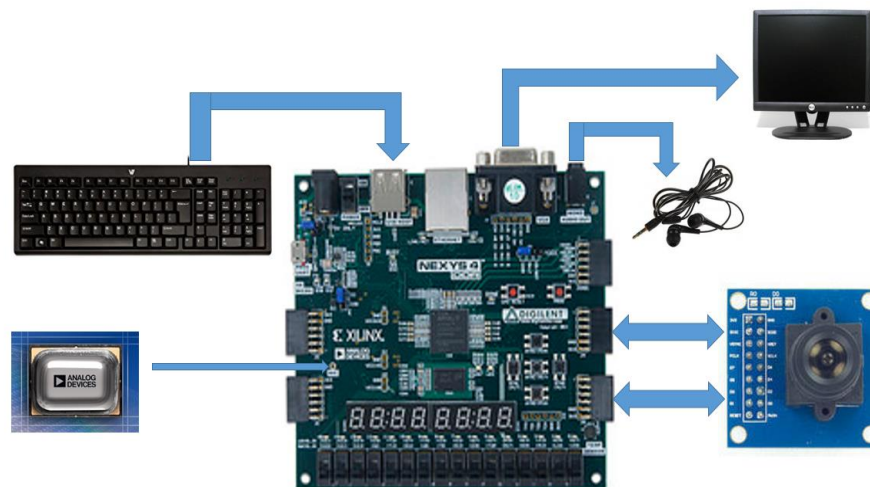


Figure 14: Connections between Nexys4 DDR3 FPGA and other input sources

Below is the PMODs connections for OV7670 VGA Camera.


Camera Header	PMOD Port
RESET	JB 6
D1	JA 2
D3	JA 3
D5	JA 4
GND	JA 5
3v3	JA 6
D7	JA 7
HREF	JA 8
VSYNC	JA 9
SIOC	JA 10
D2	JB 1
D4	JB 2
D6	JB 3
XCLK	JB 4
PWDN	JA 5
D0	JB 8
SIOD	JB 9
PCLK	JB 10

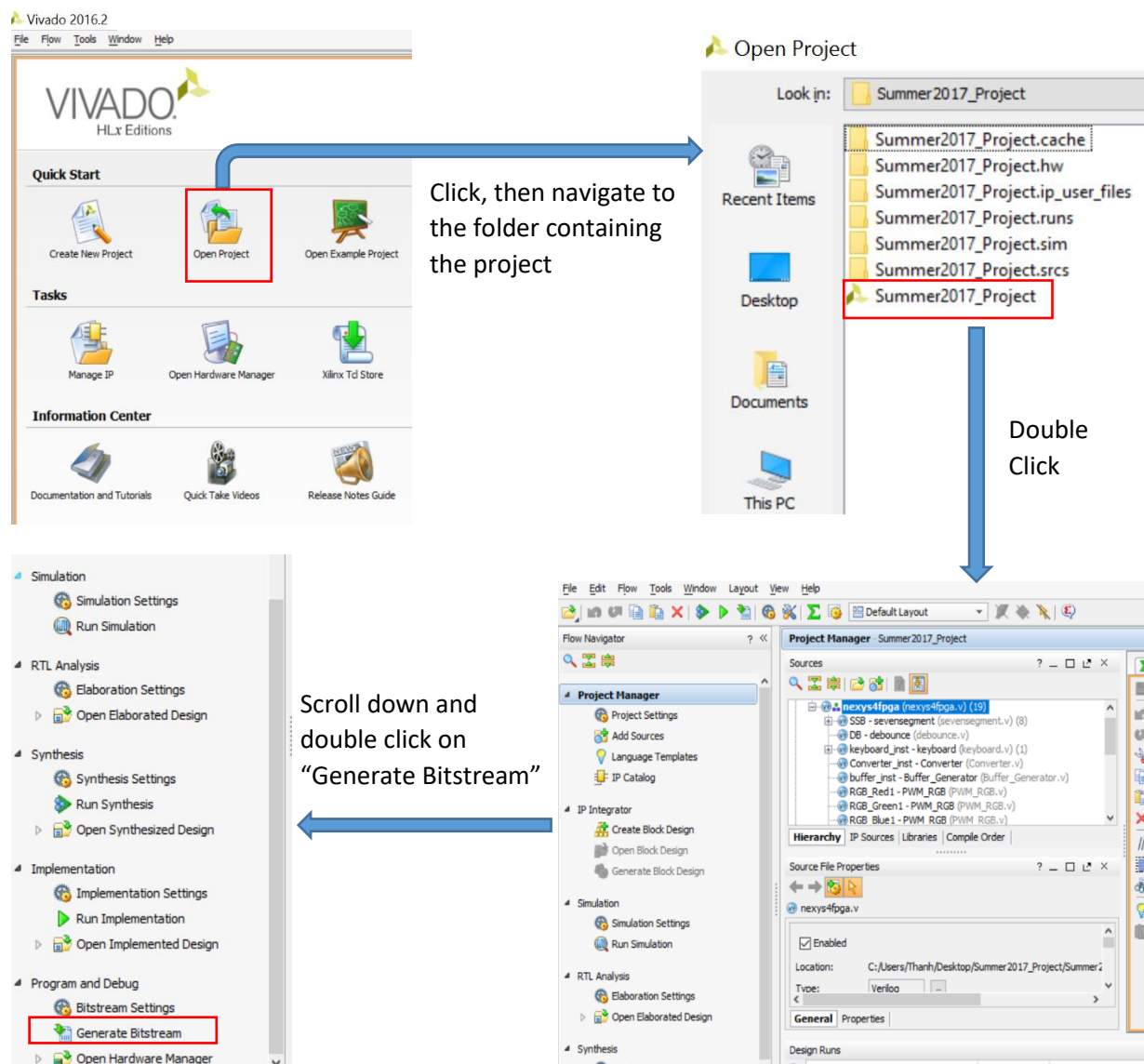
Table 3: PMODs connections for OV7670 VGA Camera

VI. Software Setup

This section shows users how to use Vivado software to open the design file, generate the bitstream file and load the bitstream file into the Nexys4 DDR3 FPGA board. The link for downloading Vivado software will be in the Appendix section.

Note that the current design is developed by using Vivado 2016.2. Using newer or older versions might see some warnings or errors.

After installing the software, look for this icon  and double click on it.

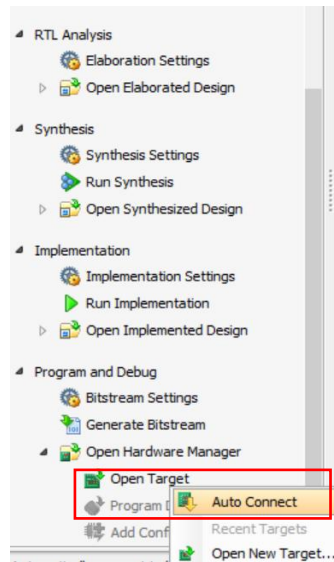


Click, then navigate to the folder containing the project

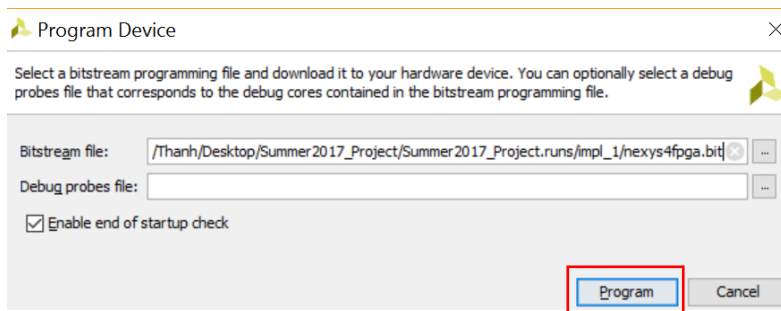
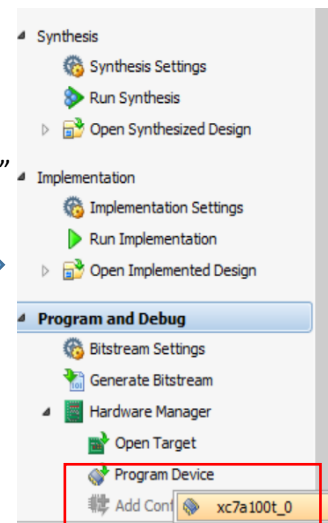
Double Click

Scroll down and double click on "Generate Bitstream"

Connect the Nexys4
DDR3 FPGA with
your laptop, then
click “Open Target”
→ “Auto Connect”



Click “Program Device”
→ “xc7a100t_0”



Check the path of
the bitstream file,
then click “Program”



VII. Appendix

7.1. Software Download and Tutorial

Download Vivado 2016.2 using this link below:

<https://www.xilinx.com/support/download.html>

Learning how to use Vivado using this link below:

<https://www.xilinx.com/support/university/vivado/vivado-teaching-material/hdl-design.html>

7.2. Bill of Material (BOM)

Nexys4 DDR3 FPGA board (Sign up using student account for half price purchase):

<http://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/>

PS2 Keyboard:

<https://www.amazon.com/dp/B017M4IX8W>

Earphone:

https://www.amazon.com/Sony-MDRE9LP-BLK-Ear-Buds/dp/B004RE3YNA/ref=sr_1_4?s=electronics&ie=UTF8&qid=1505609227&sr=1-4&keywords=earphone+sony

OV7670 VGA Camera

https://www.amazon.com/Arducam-Megapixels-OV7670-640x480-Compatible/dp/B013JRXG24/ref=sr_1_1?s=electronics&ie=UTF8&qid=1505609257&sr=1-1&keywords=ov7670+camera+module

7.3. Additional Documents

This section contains all the additional technical documentations on how to implement the communication between the Nexys4 DDR3 FPGA board and each input source and how to retrieve data from each input source.

PDM microphone and Nexys4 DD3 FPGA board: **PDM microphone with FPGA.pdf**

Keyboard and Nexys4 DDR3 FPGA board: **Keyboard with FPGA.pdf**

OV7670 VGA Camera and Nexys4 DDR3 FPGA board: **OV7670 with FPGA.pdf**