ENGG3050: Embedded Reconfigurable Computing

Laboratory Manual

School of Engineering, University of Guelph
Fall 2025

Reconfigurable Computing Systems (RCS) refers to a new class of computer architecture which take advantage of application level-parallelism. This course deals mainly with digital systems implemented on Field Programmable Gate Arrays (FPGA). In this course, we investigate the state-of-the-art in reconfigurable computing and the main factors driving it. Initially, we review the basic concepts of programmable logic in general and FPGA's in particular. Design entry based on Hardware Descriptive Languages and High Level Languages will also be covered. Specific reconfigurable computing systems (i.e architectures) will be examined with emphasis on limitations and future research opportunities.

This lab manual presents the basic tools that we will be using in ENGG3050 in the design of digital circuits and provides a number of methods and procedures suitable for a variety of digital design applications.

1 Laboratory Objectives and Practices

The ENGG3050 Reconfigurable Computing labs are an integral part of the course. The objectives of the laboratory are:

- to help you understand and assimilate the lecture material.
- to give you practical experience with the process of design and implementation of digital circuits.
- to give you hands-on-experience with CAD tools for digital hardware development.

1.1 Laboratory Practices

1. Absolutely no food or drink in the laboratories.
2. Clean up after yourselves. Put paper in the recycling bin and garbage in the trash.
3. Do not leave the door open.

The room will be closed after-hours if these rules cannot be followed.!

1.2 Laboratory Recommendations

- Labs are to be done in groups.
- The labs are to be demonstrated during the lab period on the due date.
- All written lab reports are due the following lab.
- All projects created for labs will be saved in the drop box of course link.
- Students are encouraged to work ahead.

- **Your grades will depend on your preparation of the labs, reports and demo.**
- Please read all tutorials found on the WEB before attempting any lab.

1.3 Lecture & Laboratory Schedule

Lectures:				
Tuesday		08:30 AM - 10:00 AM	MINS 017	S. Areibi
Thursday		08:30 AM - 10:00 AM	MINS 017	S. Areibi
Laboratory:				
Monday	Sec 01	15:30 PM - 17:20 PM	RICH 1532	H. Vahedi

1.4 Laboratory Modules

There will be 6 labs throughout the term. Below are the start and due dates:

Week	Topic	Weight	Report	Due
1	L1: Tutorial on Xilinx Vivado and NEXYS A7 Board	(5%)	Yes	Week 2
2	L2: Design of a Complete Datapath (ALU)	(10%)	Yes	Week 3
3	L3: Design of a Control Unit	(15%)	Yes	Week 4
4	L4: Design for Performance (Arithmetic Units)	(20%)	Yes	Week 6
5	L4: Continue Lab4 ...		Yes	-
6	L5: Design of Custom IP (H/S Co-design)	(25%)	Yes	Week 8
7	L5: Continue Lab5 ...	-	-	-
8	L6: Xilinx High Level Synthesis (HLS)	(25%)	Yes	Week 10
9	L6: Continue Lab6 ...	-	-	-

2 The Digital Design Process

A design always starts with specifications (or "specs") such as "the circuit must be able to add two 4-bit numbers in less than 12 nanoseconds". Specifications describe what the circuit must do, but not how it is done. The specifications are usually set by the customer who wants the final product, but the designer can set specs as well since the customer does not always know what he or she wants.

The digital design flow is shown in Figure 1. Once the designer knows what the circuit must do, he or she can begin to

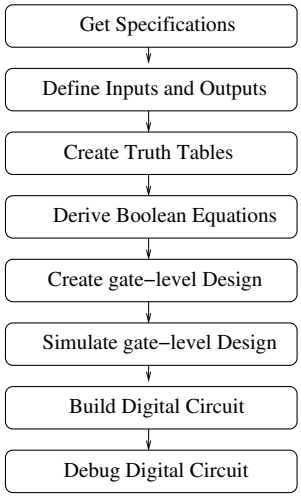

Figure 1: The Flow of Tasks involved in Digital Design

determine how it is done. Defining what the circuit receives as inputs and the outputs it generates is a good first step. Once the inputs and outputs are known, the designer has to create truth tables, which list what values the outputs will have for each possible combination of input values. Once the truth table is written down, the designer has to derive Boolean equations that describe how each binary output can be computed from the binary inputs using the logical operators available. There are a variety of manual and computer-assisted methods to accomplish this step.

Next, the Boolean equations derived in the previous step are transformed into VHDL and the design is synthesized, implemented via the CAD tool.

Before building the circuit, it is a good thing to check and make sure that all the previous steps have been completed correctly. Therefore, the gate-level design of the previous step is simulated to check its operation. In its most primitive form, simulation is performed by the designer, who traces various input.

3 The Xilinx Vivado Design Flow

Xilinx's Vivado Software is an environment for creating programs which describe logic designs (Figure 2).

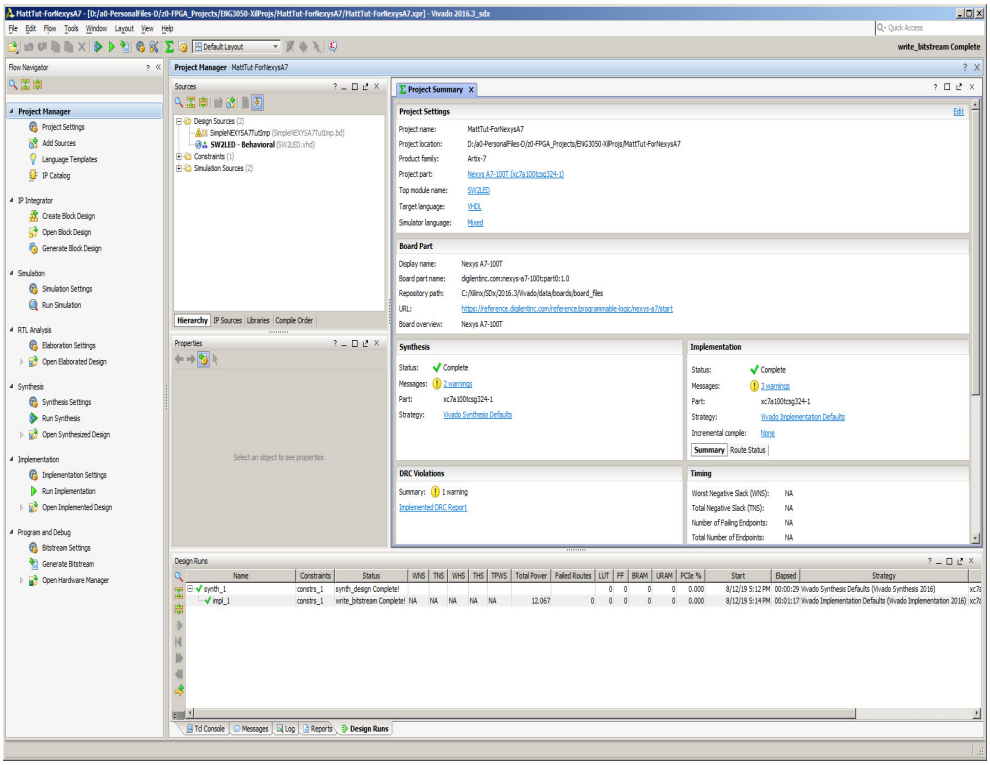

Figure 2: Xilinx Vivado Software Main Window

When using Vivado, your main design flow (shown in Figure 1) progresses as follows:

1. Digital designs are entered using a Hardware Description Language (HDL) by writing Verilog or VHDL programs with the HDL text editor.
2. A functional simulator checks the operation of the compiled design and lets you view the results to see if it is doing what you want. You can go back and edit the Verilog or VHDL file, and/or state machine diagram if any errors are found.
3. The Vivado tool then synthesizes the design, and them implements it by compiling the design using the place and route steps. Finally a bit-stream is created which is used to program the Field Programmable Logic Device (FPLD). It is in this step that a particular device is targeted, such as the Spartan 6 or Virtex device. For Spartan/Virtex devices, the implementation requires mapping the circuit to the FPGA architecture, placing the gates in specific Configurable Logic Blocks (CLBs), and then routing the wires.
4. A timing simulation of your design can be run after the Foundation Implementation tools have determined the gate and routing delays associated with a particular mapping to an FPLD architecture.
5. To download the bit-stream onto the FPGA board you will use the micro USB connection and tool available in Vivado CAD flow.

These steps will be described in the tutorial supplied with the first lab.

Figure 3 shows the changes in the digital design flow (previously shown in Figure 1) when using the Xilinx Vivado CAD flow with the NEXYS A7 FPGA board.

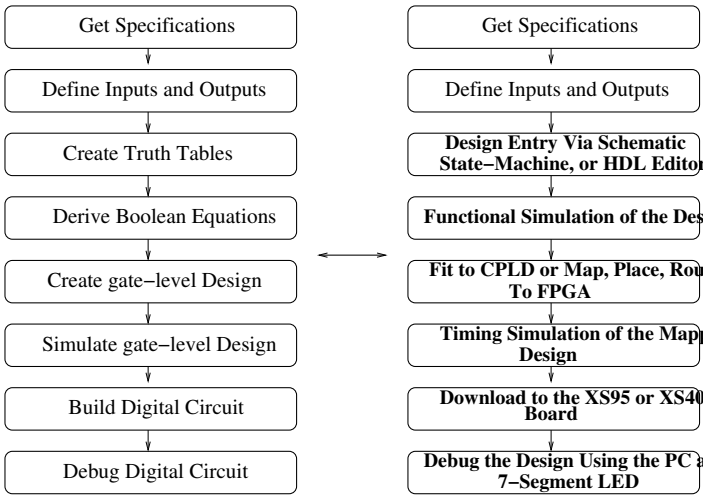

Figure 3: The flow of tasks involved in digital design

4 A Quick Introduction to the Digilent NEXYS A7 FPGA Board

The Digilent Corporation has many products based around FPGAs and CPLDs. The NEXYS A7 is a basic development board (as seen in Figure 4) containing a Xilinx Artix-7 FPGA chip (XC7A100T-1CSG324C).

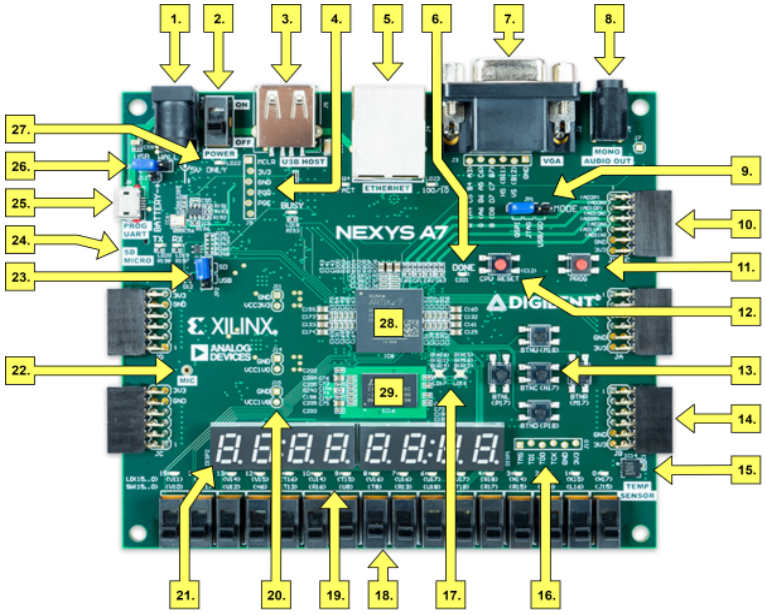

Figure 4: The Digilent NEXYS A7 FPGA board

Table 4 shows the NEXYS A7 feature callout. With its large, high-capacity FPGA, generous external memories, and

Callout	Component Description	Callout	Component Description
1	Power jack	16	JTAG port for external cable
2	Power switch	17	Tri-color RGB LEDS
3	USB host connector	18	Slide switches (16)
4	PIC24 Programming Port	19	LEDs (16)
5	Ethernet Connector	20	Power supply test point(s)
6	FPGA programming done LED	21	Eight digit 7-Seg Display
7	VGA Connector	22	Microphone
8	Audio Connector	23	External Config Jumper (SD/USB)
9	Programming mode Jumper	24	MicroSD card slot
10	Analog Signal Pmod port (XADC)	25	Shared UART/JTAG USB port
11	FPGA Configuration reset button	26	Power Select Jumper and Battery Header
12	CPU reset button (for soft cores)	27	Power-good LED
13	Five Pushbuttons	28	Xilinx Artix-7 FPGA
14	Pmod port(s)	29	DDR2 Memory
15	Temperature Sensor		

Table 1: NEXYS A7 Board Component Description

collection of USB, Ethernet, and other ports, the Nexys A7 can host designs ranging from introductory combinational circuits to powerful embedded processors. Several built-in peripherals, including an accelerometer, temperature sensor, MEMs digital microphone, a speaker amplifier, and several I/O devices allow the Nexys A7 to be used for a wide range of designs without needing any other components.

4.1 Features of NEXYS A7 FPGA Board

The following are some of the features of the NEXYS A7 board:

1. Aritx-7 FPGA Chip:
    - 240 DSP slices
    - 1,188 Kbits of BRam
    - 15,850 Programmable logic slices (63,400 LUTs, 126,800 Flip-Flops)
2. Onboard Memory:
    - 129 MiB DDR2
    - Serial Flash
    - microSD card slot
3. Audio and Video:
    - 12-bit VGA output
    - PWM audio output
    - PDM microphone
4. Sensors:
    - 3-axis accelerometer
    - Temp Sensor.

4.2 Basic I/O on the NEXYS A7 FPGA Board

The NEXYS A7 board inlcudes two tri-color LEDs, sixteen slide switches, six push buttons, sixteen individual LEDs, and an eight-digit 7-Segment display, as shown in Figure 5.

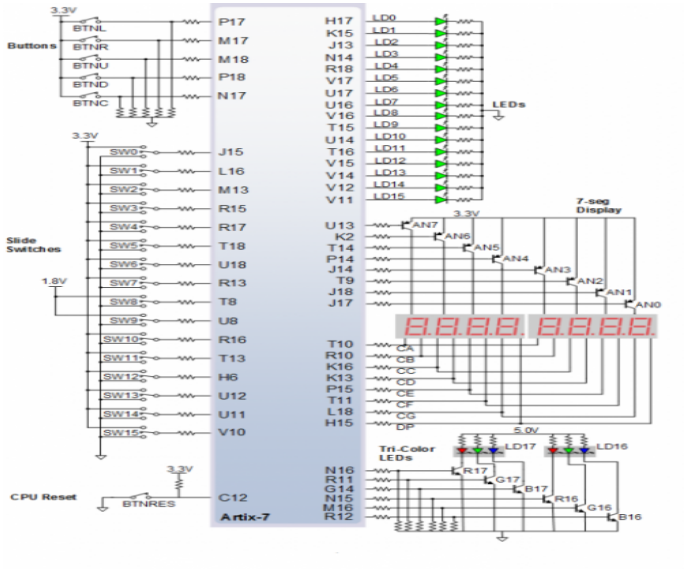

Figure 5: NEXYS A7 Basic I/O Ports with Pin Connections

4.3 Configuring the NEXYS A7 FPGA Board

After power-on, the Artix-7 FPGA must be configured (or programmed) before it can perform any functions. You can configure the FPGA in one of four ways:

1. A PC can use the Digilent USB-JTAG circuitry (portJ6, labeled PROG) to program the FPGA any time the power is on.
2. A file stored in the nonvolatile serial (SPI) flash device can be transferred to the FPGA using the SPI port.
3. A programming file can be transferred to the FPGA from a micro SD card.
4. A programming file can be transferred from a USB memory stick attached to the USB HID port.

Figure 6 shows the different options available for configuring the FPGA. An on-board "mode" jumper (JP1) and a media selection jumper (JP2) select between the programming modes. The FPGA configuration data is stored in files called

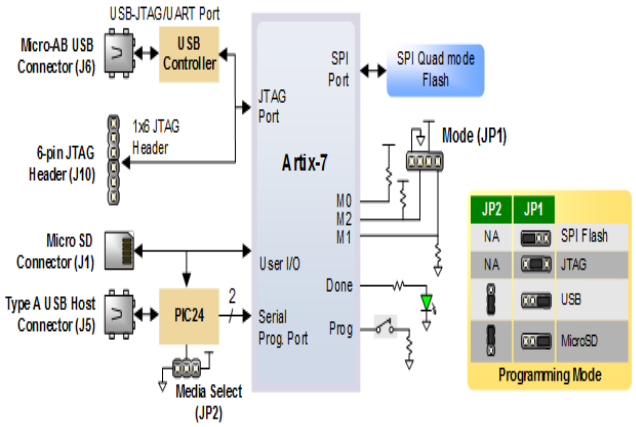

Figure 6: Programming the NEXYS A7 Board with Different Options

bitstreams that have the .bit file extension. The ISE or Vivado software from Xilinx can create bitstreams from VHDL, Verilog, or schematic-based source files (in the ISE toolset, EDK is used for MicroBlaze embedded processor-based designs). Bitstreams are stored in SRAM-based memory cells within the FPGA. This data defines the FPGA's logic functions and circuit connections, and it remains valid until it is erased by removing board power, by pressing the reset button attached to the PROG input, or by writing a new configuration file using the JTAG port.

4.4 More Information on the NEXYS A7 FPGA Board

The Nexys A7-100T is compatible with Xilinx's Vivado Design Suite as well as the ISE toolset, which includes ChipScope and EDK. Xilinx ISE has been discontinued in favor of Vivado Design Suite.

For more information on the the NEXYS A7 board check the following link: NEXYS A7 Reference Manual
The Nexys A7-50T variant is compatible only with Vivado Design Suite. Xilinx offers free WebPACK versions of these toolsets, so designs can be implemented at no additional cost. The Nexys A7 is not supported by the Digilent Adept Utility.

At the "http://www.digilentinc.com" Digilent web-site you can down-load schematics, manuals, tutorials, etc. for the NEXYS A7 board.

5 Xilinx Design Constraints (XDC) and Digilent FPGA Board

In ENG2410 while using the Xilinx ISE Foundation a **U**ser **C**onstraint **F**ile (UCF) was used to assign I/O pins in a design to the actual pins on the FPGA. In Vivado an XDC file is used instead. The XDC file has the following format:

```
#set_property -dict { PACKAGE_PIN R12   IOSTANDARD LVCMOS33 } [get_ports { LED16_B }]
```

Please check the following link for more information: XDC File

5.1 LEDs

The Digilent NEXYS A7 Board provides a series of 16 LEDs (LD0–LD15) for use. All of these LEDs are **Large active high** meaning that an LED segment will glow when a logic-high is applied to it. The following table show the connection from the NEXYS A7 Board to and LEDs expressed as XDC constraints.

```
## LEDs
#set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { LED[0] }];
#set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { LED[1] }];
#set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { LED[2] }];
#set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports { LED[3] }];
#set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { LED[4] }];
#set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { LED[5] }];
#set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports { LED[6] }];
#set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports { LED[7] }];
#set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33 } [get_ports { LED[8] }];
#set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33 } [get_ports { LED[9] }];
#set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { LED[10] }];
#set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33 } [get_ports { LED[11] }];
#set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports { LED[12] }];
#set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33 } [get_ports { LED[13] }];
#set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports { LED[14] }];
#set_property -dict { PACKAGE_PIN V11 IOSTANDARD LVCMOS33 } [get_ports { LED[15] }];
```

5.2 Seven Segment Displays

The Digilent NEXYS A7 Board provides eight multiplexed 7-segment displays for use. The following tables show the connection from the NEXYS A7 Board to the 7-segment displays expressed as XDC constraints.

```
##7 segment display
#set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { CA }];
#set_property -dict { PACKAGE_PIN R10 IOSTANDARD LVCMOS33 } [get_ports { CB }];
#set_property -dict { PACKAGE_PIN K16 IOSTANDARD LVCMOS33 } [get_ports { CC }];
#set_property -dict { PACKAGE_PIN K13 IOSTANDARD LVCMOS33 } [get_ports { CD }];
#set_property -dict { PACKAGE_PIN P15 IOSTANDARD LVCMOS33 } [get_ports { CE }];
#set_property -dict { PACKAGE_PIN T11 IOSTANDARD LVCMOS33 } [get_ports { CF }];
#set_property -dict { PACKAGE_PIN L18 IOSTANDARD LVCMOS33 } [get_ports { CG }];
#set_property -dict { PACKAGE_PIN H15 IOSTANDARD LVCMOS33 } [get_ports { DP }];
#set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { AN[0] }];
#set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { AN[1] }];
#set_property -dict { PACKAGE_PIN T9  IOSTANDARD LVCMOS33 } [get_ports { AN[2] }];
```

```
#set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { AN[3] }];
#set_property -dict { PACKAGE_PIN P14 IOSTANDARD LVCMOS33 } [get_ports { AN[4] }];
#set_property -dict { PACKAGE_PIN T14 IOSTANDARD LVCMOS33 } [get_ports { AN[5] }];
#set_property -dict { PACKAGE_PIN K2  IOSTANDARD LVCMOS33 } [get_ports { AN[6] }];
#set_property -dict { PACKAGE_PIN U13 IOSTANDARD LVCMOS33 } [get_ports { AN[7] }];
```

5.3 Slide Switches

The Digilent NEXYS A7 board has a bank of sixteen slide switches which are accessible by the user.

When closed or ON, each DIP switch pulls the connected pin of the NEXYS A7 Board to ground. When the DIP switch is open or OFF, the pin is pulled high through a $10K\Omega$ resistor.

The table below show the connections from the the Digilent NEXYS A7 Board to the switches expressed as XDC constraints.

```
##Switches
#set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { SW[0] }];
#set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { SW[1] }];
#set_property -dict { PACKAGE_PIN M13 IOSTANDARD LVCMOS33 } [get_ports { SW[2] }];
#set_property -dict { PACKAGE_PIN R15 IOSTANDARD LVCMOS33 } [get_ports { SW[3] }];
#set_property -dict { PACKAGE_PIN R17 IOSTANDARD LVCMOS33 } [get_ports { SW[4] }];
#set_property -dict { PACKAGE_PIN T18 IOSTANDARD LVCMOS33 } [get_ports { SW[5] }];
#set_property -dict { PACKAGE_PIN U18 IOSTANDARD LVCMOS33 } [get_ports { SW[6] }];
#set_property -dict { PACKAGE_PIN R13 IOSTANDARD LVCMOS33 } [get_ports { SW[7] }];
#set_property -dict { PACKAGE_PIN T8  IOSTANDARD LVCMOS18 } [get_ports { SW[8] }];
#set_property -dict { PACKAGE_PIN U8  IOSTANDARD LVCMOS18 } [get_ports { SW[9] }];
#set_property -dict { PACKAGE_PIN R16 IOSTANDARD LVCMOS33 } [get_ports { SW[10] }];
#set_property -dict { PACKAGE_PIN T13 IOSTANDARD LVCMOS33 } [get_ports { SW[11] }];
#set_property -dict { PACKAGE_PIN H6  IOSTANDARD LVCMOS33 } [get_ports { SW[12] }];
#set_property -dict { PACKAGE_PIN U12 IOSTANDARD LVCMOS33 } [get_ports { SW[13] }];
#set_property -dict { PACKAGE_PIN U11 IOSTANDARD LVCMOS33 } [get_ports { SW[14] }];
#set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { SW[15] }];
```

5.4 Push Buttons

The Digilent NEXYS A7 board has five pushbuttons (labeled BTNL, BTNR, BTNU, BTND, BTN) which are accessible by the user.

When pressed, each pushbutton pulls the connected pin of the NEXYS A7 Board to ground. Otherwise, the pin is pulled high through a $10K\Omega$ resistor. The table below show the connections from the the Digilent NEXYS A7 Board to the push buttons expressed as XDC constraints.

```
##Buttons
#set_property -dict { PACKAGE_PIN C12 IOSTANDARD LVCMOS33 } [get_ports { CPU_RESETN }
#set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { BTNC }];
#set_property -dict { PACKAGE_PIN M18 IOSTANDARD LVCMOS33 } [get_ports { BTNU }];
#set_property -dict { PACKAGE_PIN P17 IOSTANDARD LVCMOS33 } [get_ports { BTNL }];
#set_property -dict { PACKAGE_PIN M17 IOSTANDARD LVCMOS33 } [get_ports { BTNR }];
#set_property -dict { PACKAGE_PIN P18 IOSTANDARD LVCMOS33 } [get_ports { BTND }];
```

6 Report and Demonstration

You are expected to submit a report and demo your lab to the lab instructor or TA.

For the demo, be prepared to:

- demonstrate the operation of the system
- explain how your design works
- explain how the components of the system works

6.1 Writeup

- **Problem Statement**
- **Assumptions and Constraints**
- **System Analysis, Design and Justifications of Decisions**
- **Hardware**
    1. used components and their specifications
    2. schematic
    3. brief explanation of operation
    4. any required calculations
    5. timing diagram if appropriate (i.e Simulation: Functional, post synthesis and post place and route).
- **Your analysis should include:**
    1. How you would improve upon your project given more time.
    2. Problems encountered during the lab.
    3. Your approach to debugging the problem.

7 A Quick Introduction to the Avnet/Digilent FPGA ZedBoard

The ZedBoard is an evaluation and development board based on the Xilinx Zynq-7000 Extensible Processing Platform. Combining a dual Corex-A9 Processing System (PS) with 85,000 Series-7 Programmable Logic (PL) cells, the Zynq-7000 EPP can be targeted for broad use in many applications as seen in Figure 7.

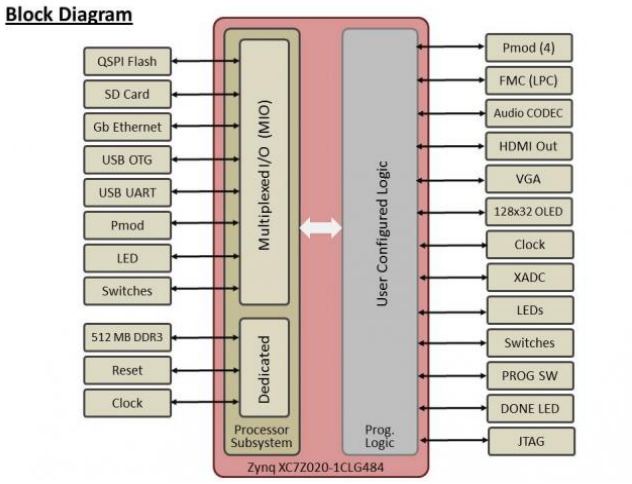

Figure 7: The Avnet ZedBoard BlockDiagram

7.1 ZedBoard Kit Contents

The Zedboard kit consists of the following items (as seen in Figure 8.

- ZedBoard
- 12 Volt/5 Ampere power supply
- USB-A to Micro-USB-B cable
- Micro-USB-B Type A Female adapter cable
- 4 GB SD card
- Documentation (Getting Started Card)

7.2 ZedBoard Features

The ZedBoard's robust mix of on-board peripherals and expansion capabilities make it an ideal platform for both novice and experienced designers. As seen in Figure 9, the features provided by the ZedBoard consist of:

- Memory:
  - 512 MB DDR3
  - 256 MB QSPI Flash
- Interfaces:
  - Two Reset Buttons (1 PS, 1 PL)
  - Eight dip/slide switches (PL)

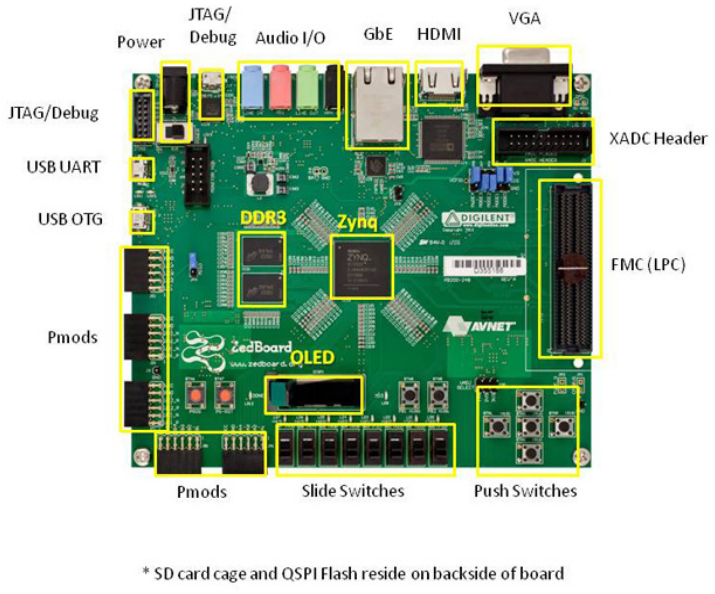

Figure 8: The Avnet ZedBoard

  - Nine USER LEDS (1 PS, 8 PL)
  - SD Card
- Display/Audio:
  - HDMI Output
  - VGA (12-bit Color)
  - Audio Line-in, Line-out, headphone, microphone

7.3 ZedBoard Documentation and Resources

You can find lots of documentation in the following links:

- www.zedboard.org (community website)
- Digilint website
- ENG3050 Course website

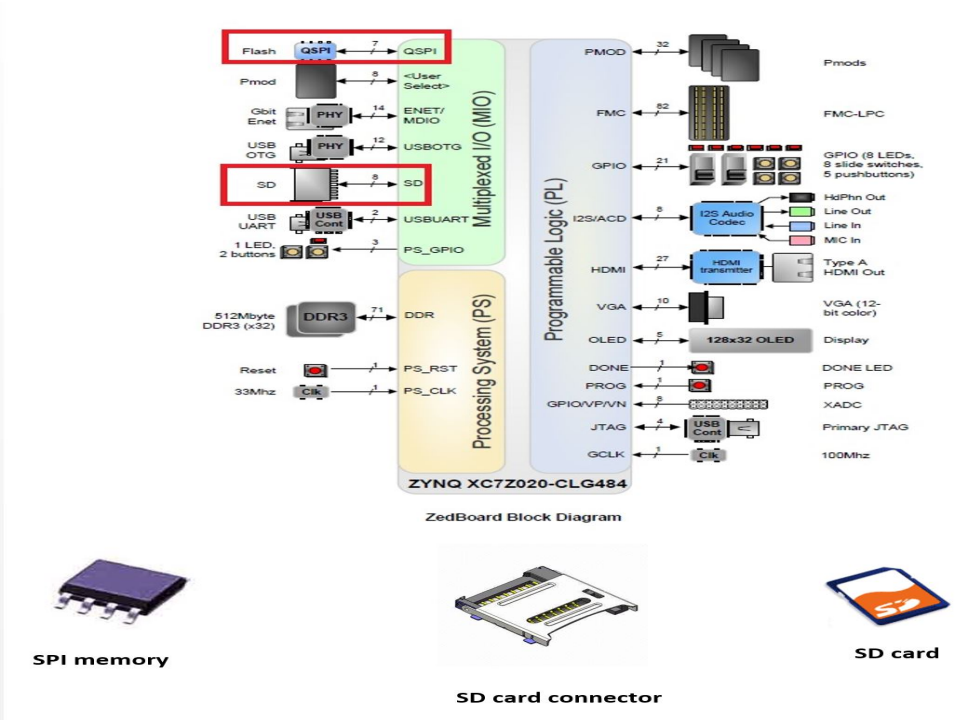

Figure 9: The ZedBoard Block Diagram

7.4 ZedBoard Configuration Modes

The ZedBoard can be used in different configuration modes including JTAG, SD Card e.t.c Figure 10 shows the Zedboard Jumper Map.

1. To boot from the SD card you need to setup the necessary configuration by making the following connections for JP6-JP11 (see Figure 11).
    - JP6 - Shorted
    - JP7 - GND
    - JP8 - GND
    - JP9 - 3V3
    - JP10 - 3V3
    - JP11 - GND
2. To use the Zedboard in JTag mode (normal mode then you need to set JP6-JP11 as follows (See Figure 12)
    - JP6 - Unconnected
    - JP7 - GND
    - JP8 - GND
    - JP9 - GND
    - JP10 - GND
    - JP11 - GND

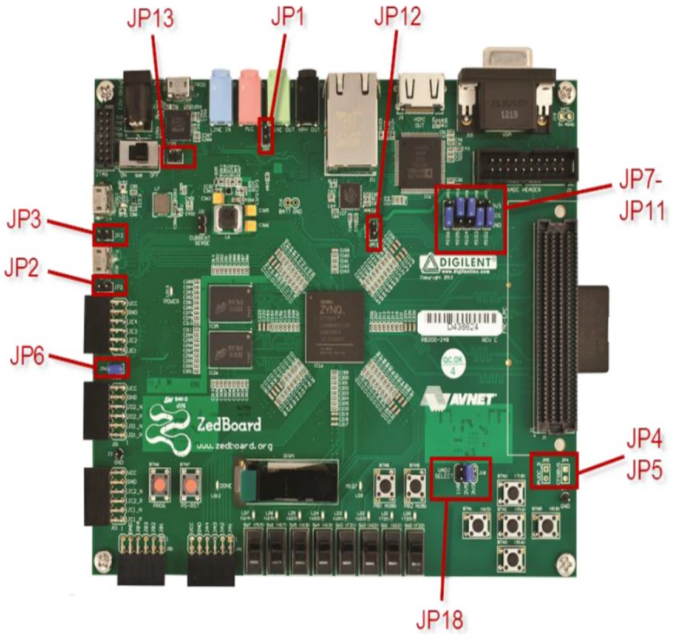

Figure 10: The ZedBoard Jumper Map

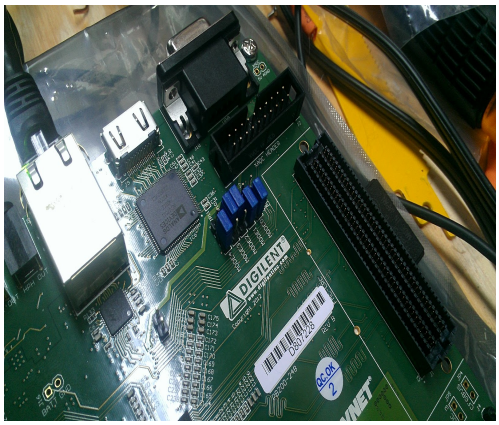

Figure 11: Setup for JP7-JP11 to boot from SD Card

Figure 12: Setup for JP7-JP11 for JTag Mode