CECS 461/561 Spring 2017 Midterm 2 Study Guide

Study Materials

Lectures 5 - 8, reference materials listed at the end of each lecture, Labs 3 – 4.

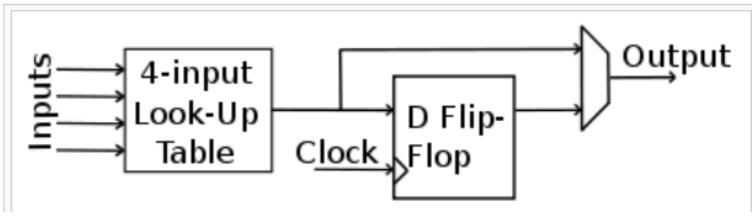
Test Format

Close book, close notes for written test. Please bring an 882E scantran form. Open book and notes for hands-on test.

Review Topics

Lecture 5 – FPGA Basic Architecture

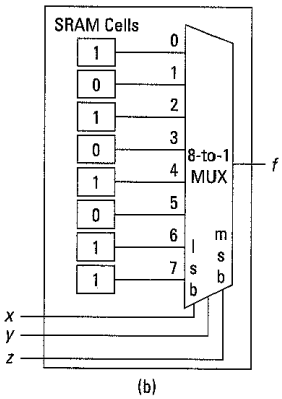
1. Three basic elements in a modern FPGA:
   a. **CLBs:** an array of configurable logic blocks
   b. **I/O pads:** Along the perimeter of the FPGA are special logic blocks connected to external package I/O pins
   c. **Routing channels:** routing resources implemented in CMOS technology
2. What is a logic cell in an FPGA? Explain how a logic cell can be used to implement a digital circuit.
   A logic cell is the combination of a look-up table and a D flip-flop. It is the low-level building block upon which FPGA designs are built. Ex:
   A classic FPGA logic cell consists of a 4-input lookup table (**LUT**), and a flip-flop, as shown below.



3. What is a function generator or LUT? Explain how to use a LUT to implement a logic function. A LUT is a multiplexer with information input connected to a configurable SRAM.
   Ex: f(x,t,z) = xy + z' can be implemented using a 3-input function generator: create the 8-line truth table. Each of the function's output bit is stored into an individual static memory (SRAM) cell and connected as inputs to an 8x1 MUX. The 3 inputs will be the select lines for the MUX:

x	y	z	xy + z'
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

(a)



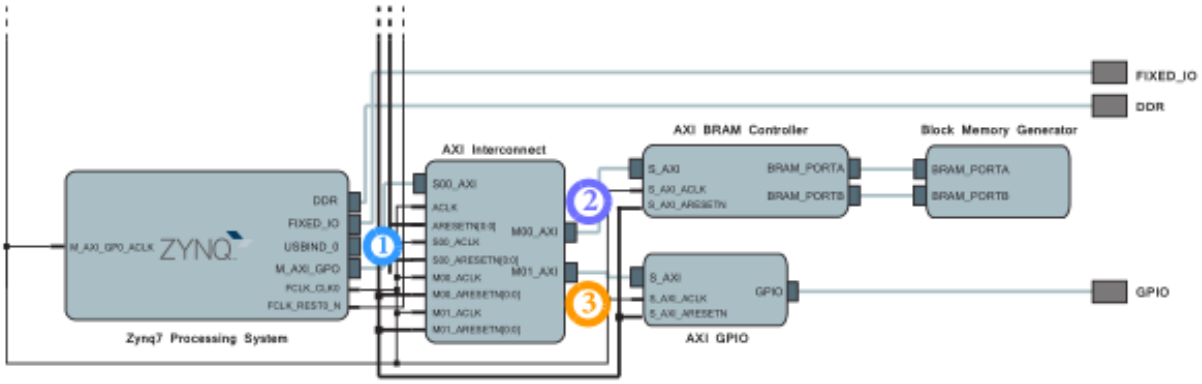
(b)

4. Memories in the FPGA:
   a. Two different types of memory in FPGA: configuration memory and BRAM
   b. Which one of the following two memory blocks in FPGA is used to hold hardware configuration: Configuration memory or BRAM? Configuration memory.
   c. What memory type is used in FPGA for configuration memory? Static RAM.
   d. In Zynq, BRAM is available in the FPGA PL as a softcore. We need to create a BRAM controller soft core to control the access of the BRAM soft core.
5. Where does software executable stored on a Xilinx Zynq FPGA?
   Two memory locations can be used to store our code:
   1) External DDR3 memory located outside of FPGA, controlled by DDR3 memory controller hard core located in PS.
   2) Internal BRAM located in PL.

Lecture 6 – Creating and Adding Custom IP

1. Understand AXI interconnect: Slide #4
   1) AXI is a protocol belonging to the ARM AMBA family of microcontroller buses.
   2) The AMBA protocol is an open standard on-chip interconnect specification, allowing the connection and management of many controllers and peripherals in a multi-master design.
   3) The AXI protocol is optimized for FPGA implementation through coordinated development with Xilinx and is used as a means of communication between IP cores of an FPGA design.
2. What are the 5 basic AXI transaction channels for AXI-Lite Slave? Slide #5 Read address channel, Read data channel, Write address channel, Write data channel, and Write response channel.
3. All AXI Channels Use A Basic "VALID/READY" Handshake, describe a typical handshake activity:
   1) *SOURCE* asserts and holds VALID when DATA is available, *DESTINATION* asserts READY if able to accept DATA
   2) DATA transferred when VALID and READY = 1
   3) *SOURCE* sends next DATA (if an actual data channel) or deasserts VALID
   4) *DESTINATION* deasserts READY if no longer able to accept DATA
4. In Lab 3&4, in the three HDL file: led_ip_v1_0.v, led_ip_v1_0_S_AXI.v, lab3_user_logic.v, which one takes care of the AXI-Lite slave handshake signals, which one implements the custom IP logic?
5. How does Xilinx Vivado provide support for IP reuse? Vivado supports reuse of IP is by providing users a means to capture their individual IP or IP subsystems and package that IP, and then make the IP available to other users through the Vivado IP catalog.
6. What are the IP resources supported by Vivado? Xilinx IP, 3rd party IP, and user IP
7. What files are supported in IP packager?
   Source code, Constraints, Test Benches (simulation files), documentation.
8. When creating custom IP using Vivado Create and Package IP Wizard, what are the major services available to support the process?
   Generates HDL **template for AXI** Lite/Full/Stream Slave/Master, and optionally generates **software drivers** (only for AXI Lite and Full slave interface) and **Test Software Application**.
9. Explain how user specified registers implemented in custom IP is used.
10. They can be used to send/receive information to/from external device controlled by the custom IP.

11. What device driver functions will be generated for custom IP by the tool? Are they Level 0, Level 1, or Level 2 drivers?
    Functions to read and write user specified registers. They are Level 0 drivers.
12. Explain all master-slave relationships presented in the following block diagram.



1. The AXI master signal from the Zynq processing system connects to the AXI slave port of the AXI Interconnect block

2. An AXI master signal from the AXI Interconnect connects to the AXI slave port of the BRAM controller

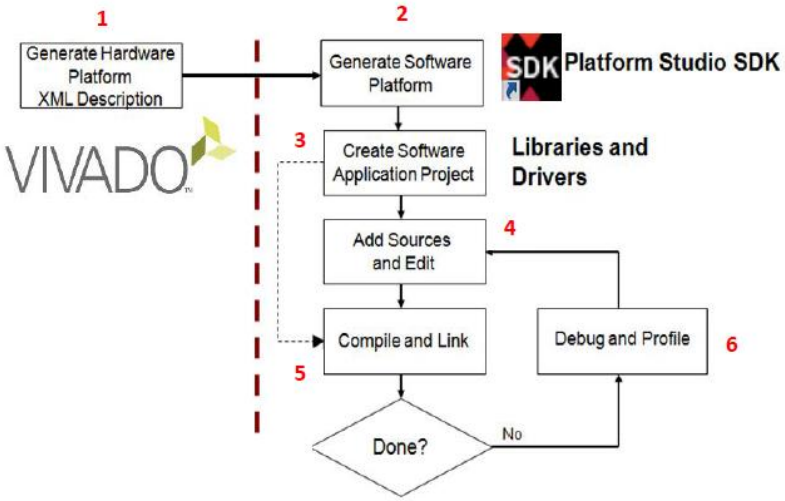
3. An AXI master signal from the AXI Interconnect connects to the AXI slave port of the GPIO instance

The AXI interconnect is a slave to the master processing system, and is a master routing signals from the processor to various instances of slave AXI devices. In this system, the interconnect is connected to two slave devices; AXI GPIO (gpio_1) and an AXI BRAM controller (bram_ctrl_1).

Lecture 7 - Software Development Environment

1. List at least three differences between software development for desktop applications and embedded systems.
2. Explain what a Board Support Package is.
   The Board Support Package provides software services based on the processor and peripherals that make up the processor system. It can be automatically created when creating Application project and it also can be created standalone. The BSP must be attached to a Hardware Platform.
3. Describe the relationships among Hardware Platform, BSP, Software Application Project in FPGA-based embedded system design using Vivado.
   A BSP is attached to a Hardware Platform, a software project is attached to a BSP project. Multiple software projects can be attached to one BSP project and multiple BSP projects can be attached to a Hardware Platform.
4. List at least two ways a linker script can control the linking process.
   1) Map the code and data to a specified memory space
   2) Set the entry point to the executable
   3) Reserve space for the stack and heap

5. Memory address space to store data and instructions can be either internal block memory or external memory.
6. Linker script is required when the software segments do not reside in a contiguous memory space.
7. SDK development flow:



Answer the following questions:

1) Which step generates libraries and drivers? 2
2) Which step generates xparameters.h? 2
3) Which step generates MSS file? 2
4) What design tasks are done moving from Step 1 to Step 2?
   Hardware Platform information is passed from Vivado to SDK, System Wrapper Hardware Platform project is created. BSP project is created.
5) After Step 2, how many projects are created in SDK? What are they?
   Two projects are created: System Wrapper Hardware Platform Project and BSP project
6) After Step 3, how many projects are created in SDK? What are they?
   Three projects are created: System Wrapper Hardware Platform Project, BSP project, and software application project.
7) In which project can we find MSS file? In which project, can we find HDF file?
   BSP project: mss file, System Wrapper Hardware Platform project: hdf file.
8) What are the purposes for performing step 6?
   Debug for function verification and performance evaluation
9) What does the dotted line from step 3 to step 5 represent?
   It shows the object code coming from BSP, including libraries and drivers used in software application project.

8. From the following memory map and memory dump in Lab 4 manual, specify what is the memory allocation for Code and data, stack and heap in Lab 4: first run and second run.

Start Address	Size	Description
0x0000_0000	1GB	External DDR RAM
0x4000_0000	2GB	Custom Peripherals (Programmable Logic including PCIe)
0xE000_0000	256MB	PS I/O Peripherals
0xF800_0000	32MB	Fixed Internal Peripherals (Timers, Watchdog, DMA, Interconnect)
0xFC00_0000	64MB	Flash Memory
0xFFFC_0000	256KB	On-Chip Memory

First run: Code and data, stack and heap all in external DDR RAM 1GB starting address: 0x00000000.
Second run: Code and data in Custom Peripheral (PL) 2GB starting address: 0x400000000, stack and heap are in Custom Peripheral (PL) 2GB: 0x400000000

Lecture 8 – Xilinx Device Driver Structure

1. What are the major objectives for the Xilinx device driver design?
   a) Provide maximum portability
   b) Support FPGA configurability
   c) Support simple and complex use cases
   d) Ease of use and maintenance
2. The layered device driver architecture support which one of the above four objectives?
   Support c) simple and complex use cases
3. Describe Xilinx layered device driver architecture. Slide #5
   - ❖ Level 2: RTOS application layer
   - ❖ Level 1: High-level device drivers that are full-featured and portable across operating systems and processors
   - ❖ Level 0: Low-level drivers for simple use cases
4. What are the characteristics of drivers in Level 0 and Level 1? Slide #5, #6
5. Specify which one(s) of the following GPIO drivers are Level 0 driver functions, which one(s) are Level 1 driver functions. Check function prototype/definition for detail.
   XGpio_WriteReg, XGpio_ReadReg, XGpio_Initialize – Level 0
   XGpio_GetDataDirection, XGpio_DiscreteRead, XGpio_DiscreteWrite – Level 1
6. Which one(s) of the above driver function are implemented with Macro, which ones are implemented as function? Why?