CS M152A Lab 1

Sequencer

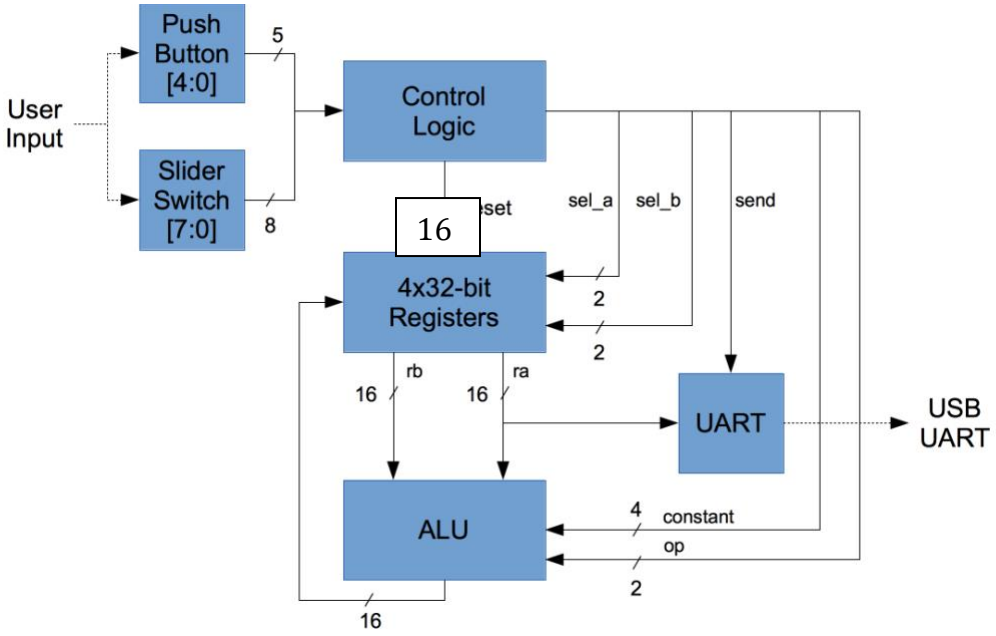
In this lab, you will study a sequencer as an example project and do exercises based on the project.

Introduction

In this lab, you are expected run a small scale FPGA project, understand it and make modification of it. It mainly involves studying the Verilog module and test bench in the example project and answer related questions. The project is a good reference for future design projects, as you can learn the styles, formats, structures of code, etc. from the project. You can also learn useful techniques, like clock dividers and debouncers, from the project. There will be questions based on the code, and there will also be a few tasks that require modifying the original code.

Example Project Design

The example project is an adder/multiplier sequencer. It has four 16-bit general purpose registers. It can perform add or multiply instructions using registers as operands. The results are stored into register files.



Project Diagram

Sequencer Operation

To operate the sequencer, the user enters instructions using the switches and push buttons:

- First 8 switch positions represent a single 8-bit instruction (up: 1, down: 0)

- Push center button to enter and execute the instruction
- The push button BTNR (right button) resets values of all registers to 0

In addition, the First 8 LED indicators shows the number of instructions executed since the last reset, in binary.

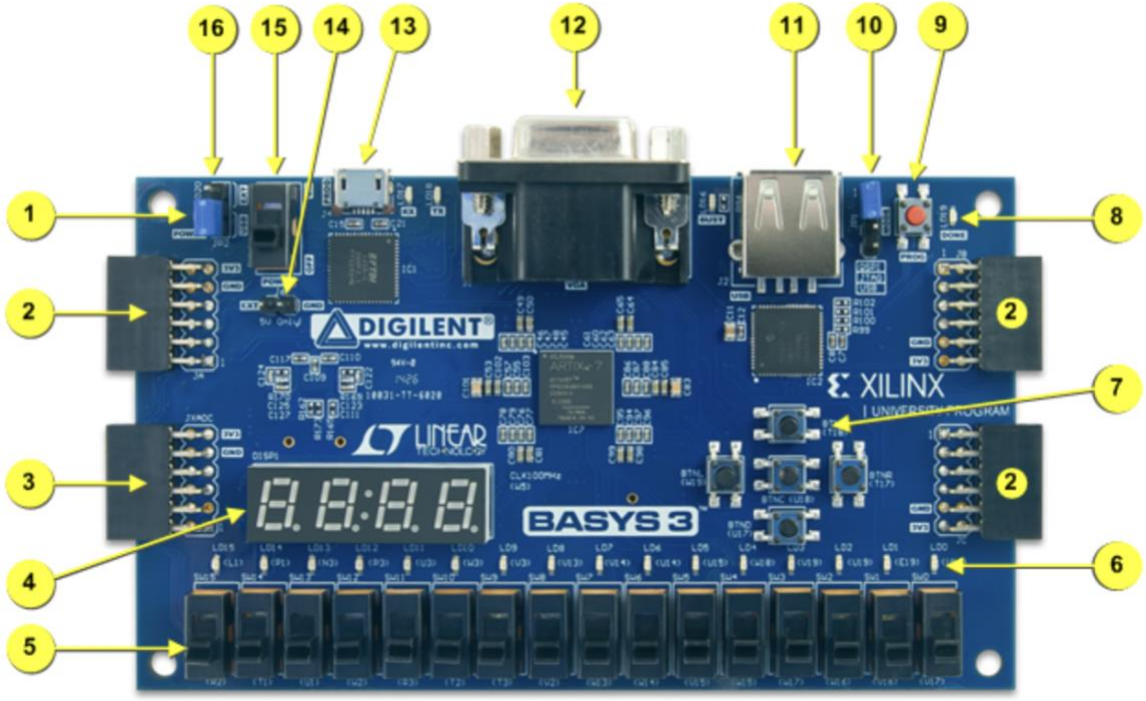


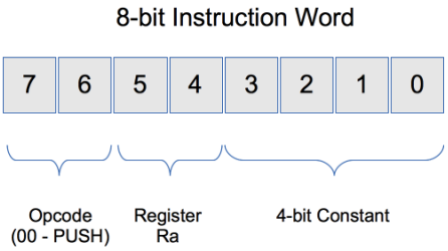
Figure 1. Basys3 board features

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Sequencer Instructions

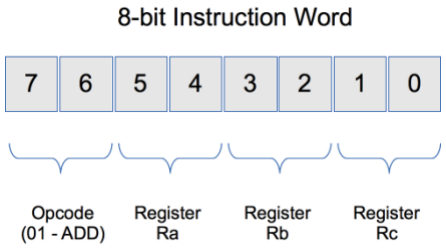
PUSH (00) instruction left-shifts the target register by 4 bits and "pushes" the new constant in. Example:

- R0 starts with value 0x55AA
- PUSH R0 0xB
- Now R0 is 0x5AAB



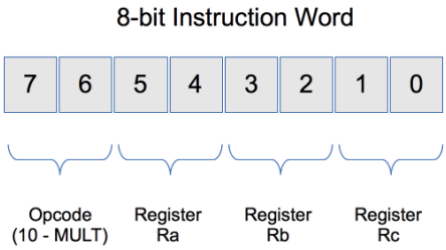
ADD (01) instruction adds Ra and Rb and stores the result to Rc. Addition is performed in unsigned integer arithmetic. Example:

- ADD R0 R1 R3
- R0 + R1 □ R3



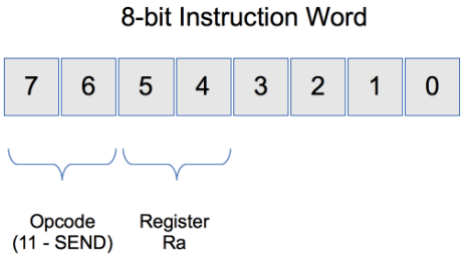
MULT (10) instruction multiplies Ra and Rb and stores the result to Rc. Inputs are assumed to be unsigned integers. Only the lower 16-bit results are retained. Example:

- MULT R3 R0 R2
- R3 * R0 □ R2



SEND (11) instruction sends the content of Rt to the UART for display. The 16-bit value is converted to ASCIIHEX and appended with a newline character. Example:

- R1 has a value of 0x1234
- SEND R1 causes the UART to send "1234\n"



Deliverables

1. (Demo) Build the project using based on the source files in src.zip, and make sure that it works. Translate the following "program" into binary instructions. Demo the UART console output to your TA after you finish.
   - PUSH R0 0x4
   - PUSH R0 0x0
   - PUSH R1 0x3
   - MULT R0 R1 R2
   - ADD R2 R0 R3
   - MULT R1 R3 R3

- SEND R0
- SEND R1
- SEND R2
- SEND R3

2. (Demo) Read the testbench code and finish workshop 2.
3. (Report) Read the implementation code and finish workshop 1. Include answers in the report. Note that the lab report format described in the syllabus does not apply to this lab