Lab 2 Report

Group 7
Daniel Chris Daniil - 504636684
Junhong Wang - 504941113

1. Introduction and requirements

Background Information
There are four registers. We have an 8-bit instruction word. Assuming MSB to be the leftmost bit, 7th and 6th bits represent the opcode. The interpretation of the rest bits depends on the instruction encoded in the opcode.

PUSH (00)
3rd, 2nd, 1st, and 0th bits represent what hex value we want to push into a register in binary. 5th and 4th bits represent which register we want to push value into.

ADD (01)
5th and 4th bits represent the register number of Ra. 3rd and 2nd bits represent the register number of Rb. 1st and 0th bits represent the register number of Rc. Finally, it will perform Rc=Ra+Rb.

MULT (10)
Ra, Rb, and Rc are assigned as above. Finally, it will perform Rc=Ra*Rb.

SEND (11)
It will display the content of Ra. Rb and Rc are ignored.

Design requirements

1. We want to perform add and multiplication on registers.
2. We want an execution button that triggers sequencer module to read inputs from switches. We also want to have a separate button solely for sending the content of Ra for display.
3. We want to output the content of ra if opcode is 11. We also want it to also display the register number along with its value (ex. R0:0003).
4. We want to write a test bench that reads a file containing instructions.
5. We want to write some instructions that display first 10 numbers of the Fibonacci series.

2. Design description

Design Description

First of all, the button pushes need to be down sampled so that one push will not be recognized as multiple pushes. Then we need to create a module that read opcode and operand from the switches, which could do some operations on four registers we created. If the opcode is send, we will trigger UART module, which handles outputting the content of a register onto the screen.

Modular Architecture

- Nexys3
    - Seq
        - Seq_rf
        - Seq_alu
            - Seq_add
            - Seq_mult
    - Uart_top
        - Uart_fifo
        - Uart

Interaction among the Modules

First the inputs are passed into Nexys 3 module and down sampled. From there, they are passed into the sequencer module. The sequencer will read the opcode and instantiate the ALU module, which eventually performs the requested operation. The sequencer also instantiates the register file module, to hold the values of the registers. Finally the Nexys 3 module passes a flag and the content of register A into the UART module. If the opcode was "send" instruction, this flag will be set 1. UART will print the content of register A if the flag is 1.

Interface of Each Major Module

Nexys3 module:

The top module that includes seq module and uart_top module.

Seq module:

A sequencer module that has register file module and ALU module.

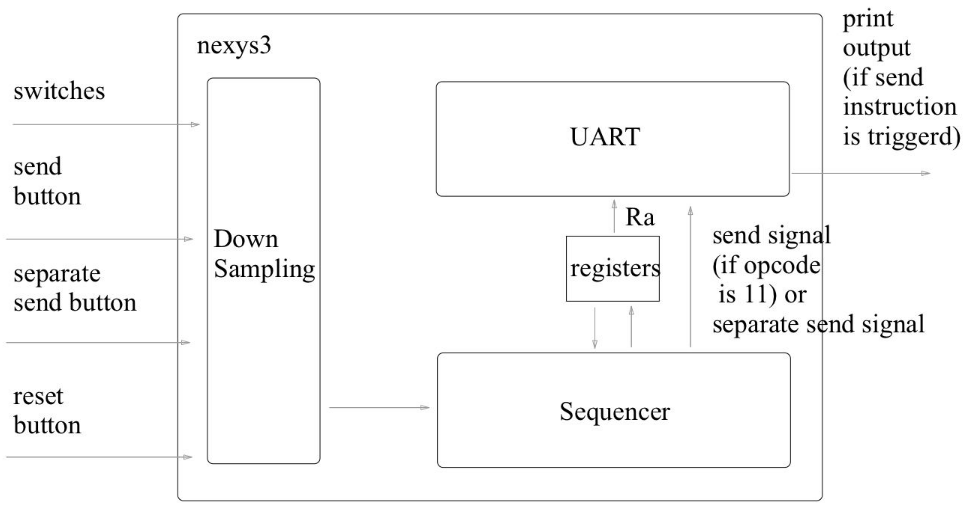
RF module:

A module that contains the information of the four registers we want to use.

ALU module:
An Arithmetic Logic Unit module that performs addition and multiplication, which are handled by add module and mult module.

Uart_top module:
The top module of uart related modules including FIFO module and Uart module.

Schematics for System Architecture


nexys3
switches
send button
separate send button
reset button
Down Sampling
UART
Ra
registers
Sequencer
send signal (if opcode is 11) or separate send signal
print output (if send instruction is triggerd)

3. Simulation documentation

Requirement Tested / Test Cases

Other than the performed demos, we created our own test cases to ensure the correctness of our components. These test cases are presented below:

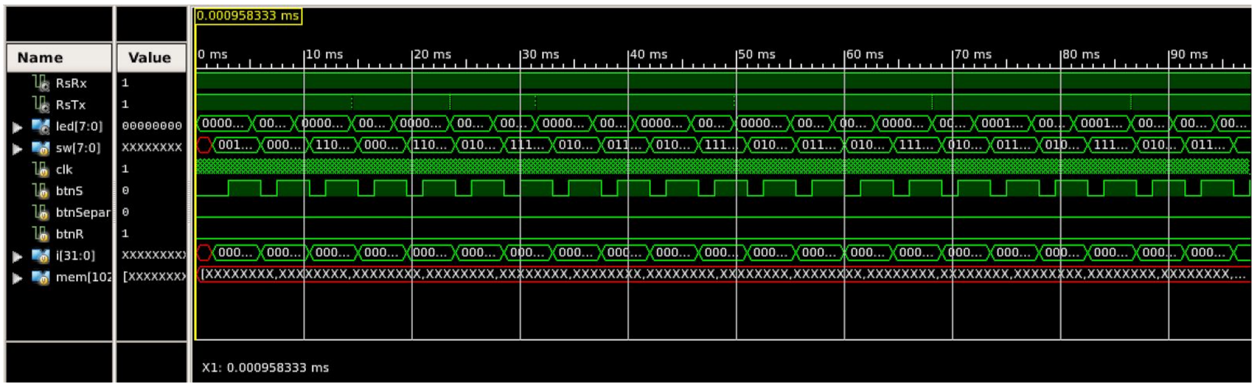
Requirement Tested	Sequence	Expected State	Result State

Multiplication	PUSH R0 0x4
	PUSH R1 0x3
	MULT R0 R1 R2
		R0 = 0004
		R1 = 0003
		R2 = 000C
			R0 = 0004
			R1 = 0003
			R2 = 000C

Seperate Button	PUSH R0 0x1 #15 btnSeperateS=1 #30 btnSeperateS=0	Must see: 0001 as output	We saw 0001 as output
Register Name	PUSH R0 0x1 #15 btnSeperateS=1 #30 btnSeperateS=0	Must see: R0:0001 as output	We saw: R0:0001 as output
Instruction file automation	Create file with: PUSH R0 0x4 SEND R0	Must see: R0:0004	We saw: R0:0004
Fibonacci Number output	Create file to output the fibonacci sequence	Must see the fibonacci sequence output	We saw the fibonacci sequence output

Bugs
We did not observe any bugs.

Simulation Waveforms
(Simulating Fibonacci series)



(Console output)

```
R0: 0000
R1: 0001
R2: 0001
R2: 0002
R2: 0003
R2: 0005
R2: 0008
R2: 000D
R2: 0015
R2: 0022
```

4. Conclusion

Summary of the Design

Essentially, there are two main module: sequencer and uart. We let sequencer interpret the 8 bit instructions and perform arithmetic operations. The sequencer will notify the uart module if the opcode is “send” (11). If the separate send button is pressed, the instruction is interpreted as “send” instruction regardless of the actual value of opcode.

Difficulties

The largest difficulty we were faced with was understand the initial code we were given. The reasons were poor documentation and our inexperience with Verilog. It was our first time being presented with such complicated interactions between Verilog modules.

Suggestions

Please provide documentation (comments) on the code because some variable names are unintuitive.