

Digital Design & Computer Arch.

Lab 8.2 Supplement: Full System Integration

(Presentation by Aaron Zeller)

Frank K. Gürkaynak

Seyyedmohammad Sadrosadati

ETH Zürich

Spring 2024

[21. May 2024]

Lab 8 Overview

- You will build a whole single-cycle processor and write assembly code that runs on the FPGA board.
- You will learn how a processor is built.
- Learn how the processor communicates with the outside world.
- Implement the MIPS processor and demonstrate a simple “snake” program on the FPGA starter kit.

Lab 8 Sessions

- **Session I:** The Crawling Snake
- **Session II:** Speed Up the Snake

Lab 8 Session II: Speed Up the Snake



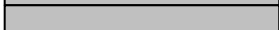
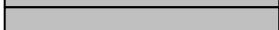
- In lab 8.1 you have made changes to the code template to produce a **crawling snake** on the 7-segment display.
- Lab 8.2 is similar and again requires **careful reading** of the lab manual.

top.v	Top level hierarchy that connects the MIPS processor to the I/O on the FPGA board. <i>You will modify this file for Part 2.</i>
top.xdc	Constraints file of the top level. <i>You will modify this file for Part 2.</i>
MIPS.v	The main processor. <i>For Part 1, you have to add code inside of this file only.</i>
DataMemory.v (datamem_h.txt)	The initial content of the data memory (composed of 64 32-bit words). The datamem_h.txt file contains the data part of the assembly program in a hexadecimal form. This module "loads" the data. <i>You will only have to modify the .txt file if you do the challenges.</i>
InstructionMemory.v (insmem_h.txt)	The ROM (composed of 64 32-bit words) contains the program. The insmem_h.txt file contains the assembly instructions we want to run on the MIPS processor in a hexadecimal form. This module "loads" the instructions. <i>You will modify the .txt file for Part 2.</i>
RegisterFile.v	Register file that creates two instances of reg_half.v as read ports and has one write port. <i>This is the implementation of a register. You do not need to modify it.</i>
reg_half.v reg_half.ngc	Component describing a single port memory and binary description of how it is mapped in the FPGA. <i>These are used to implement the register. You do not need to modify it.</i>
ALU.v	ALU similar to the one from Lab 5. <i>You should not change anything in this file, but if you want, you can use your own implementation (just make sure that it works).</i>
ControlUnit.v	The unit that does the instruction decoding and generates nearly all the control signals. Table 7.5 on page 379 lists most of them and their truth tables (only the ALUOp signal is generated differently in the exercise). <i>This is just a combinational circuit, and it's already given; you don't need to change anything here.</i>
snake_patterns.asm	Assembly program corresponding to the datamem_h.txt and insmem_h.txt dump files that displays a crawling snake on the 7-segment display when all the parts are connected properly. <i>You have to modify this file for Part 2, where you will also learn how to generate the dump files.</i>

Lab 8 Session I: The Crawling Snake

- Let us first see some of the **modifications** made in lab 8.1.
- `IsMemWrite` tells us if an instructions writes to memory or an I / O operation.

```
// Memory Mapped I/O
assign IsI0 = (ALUResult[31:4] == 28'h00007ff) ? 1 : 0; // 1: when datamemory address
// falls into I/O address range



// TODO Part 1
assign IsMemWrite =  // Is 1 when there is a SW instruction on DataMem address
assign IOWriteData =  // This line is connected directly to WriteData
assign IOAddr =  // The LSB 4 bits of the Address is assigned to IOAddr
assign IOWriteEn =  // Is 1 when there is a SW instruction on IO address

assign ReadMemI0 = IsI0 ? IOReadData : ReadData; // Mux selects memory or I/O
// Select either the Data Memory (or IO) output or the ALU Result
assign Result = MemtoReg ? ReadMemI0 : ALUResult; // Slightly modified to include above
```

- `IsMemWrite` hence is 1 if an operation is a memory write operation and not an I / O operation.

Lab 8 Session I: The Crawling Snake





- `IsMemWrite` hence is 1 if an operation is a memory write operation and not an I / O operation.
- This is because there are I / O operations that also write to memory.

 <u>What's the difference between <code>MemWrite</code>, <code>DataMemWrite</code>, and <code>IOWriteEn</code>?</u> 					
			Signal		
			MemWrite	DataMemWrite	IOWriteEn
Instruction	SW instruction	on DataMem address		1	0
		on IO address	1	0	1
	Non-SW instruction	D.C. (don't care)	0	0	0

Lab 8 Session I: The Crawling Snake

- `IsMemWrite` hence is 1 if an operation is a memory write operation and not an I / O operation.
- This is because there are I / O operations that also write to memory.

```
// Memory Mapped I/O
assign IsIO = (ALUResult[31:4] == 28'h00007ff) ? 1 : 0; // 1: when datamemory address
                                                    // falls into I/O address range

// TODO Part 1
assign IsMemWrite =  // Is 1 when there is a SW instruction on DataMem address
assign IOWriteData =  // This line is connected directly to WriteData
assign IOAddr =  // The LSB 4 bits of the Address is assigned to IOAddr
assign IOWriteEn =  // Is 1 when there is a SW instruction on IO address

assign ReadMemIO = IsIO ? IOReadData : ReadData; // Mux selects memory or I/O»
// Select either the Data Memory (or IO) output or the ALU Result»
assign Result = MemtoReg ? ReadMemIO : ALUResult; // Slightly modified to include above
```

Lab 8 Session II: Speed Up the Snake

- Extend the top-level hierarchy:
 - **Modify** the **I/O controller** to **accept the inputs**.
- Understand the provided assembly program and **modify your assembly code** to **accept inputs**.
 - The snake should crawl at **different speeds** for **different inputs**.
 - The **inputs** will be controlled by **switches** on the **FPGA board**.
- *Optionally, you have two challenge tasks to complete.*
 - Change the **direction** of the snake.
 - Change the **pattern** of the snake.

Lab 8 Session II: Speed Up the Snake

- You should proceed similarly to lab 8.1 here.

```
// TODO for Part II of Lab 8
// The speed of the snake must be read as input and sent to the MIPS processor.
// Create the 32 bit IOReadData based on IOAddr value. Remember IOAddr is a 4-bit
// value.

assign IOReadData = ;
```

Address	Direction	Width	Description
0x00007FF0	out	28 bits	Value to be sent to the 7-segment display.
0x00007FF4	in	2 bits	Speed step 0, 1, 2 or 3.

- The ALU is responsible for arithmetic, logic and **memory address computation**.
 - The **opcode** decides then what the result represents.

Lab 8 Session II: Speed Up the Snake

- You should proceed similarly to lab 8.1 here.

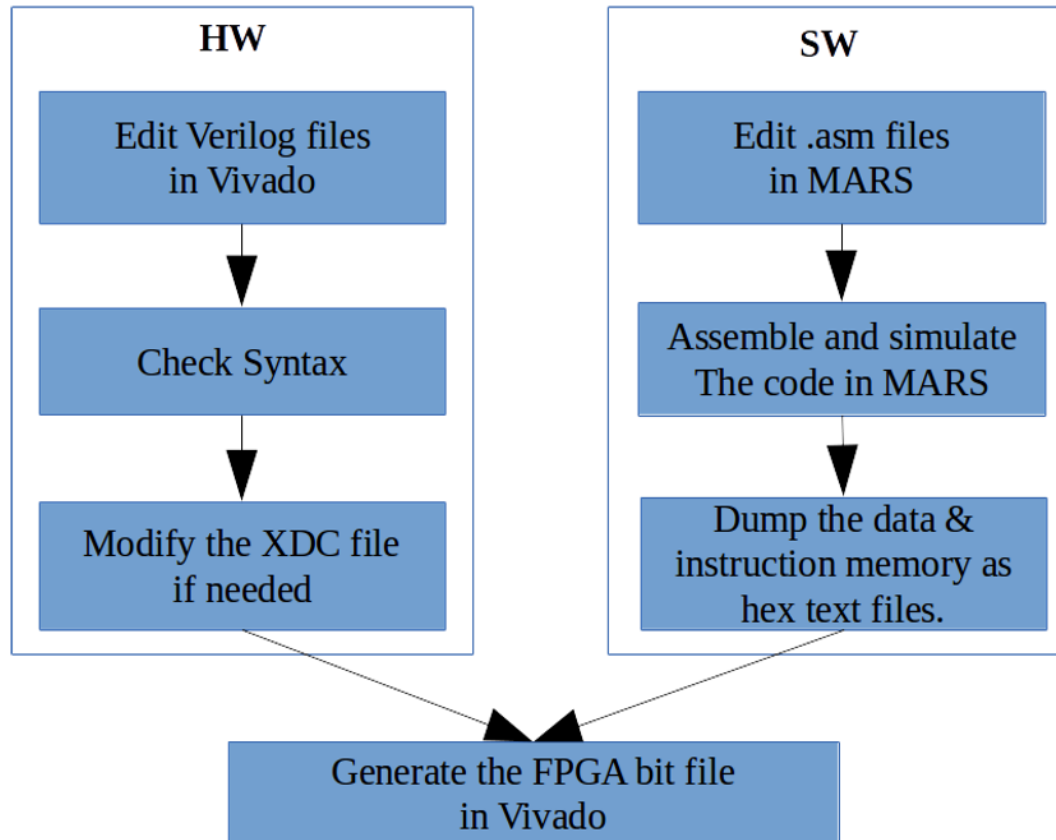
```
// TODO for Part II of Lab 8
// The speed of the snake must be read as input and sent to the MIPS processor.
// Create the 32 bit IOReadData based on IOAddr value. Remember IOAddr is a 4-bit
// value.
```

```
assign IOReadData = (IOAddr == 4'  ?  : 
```

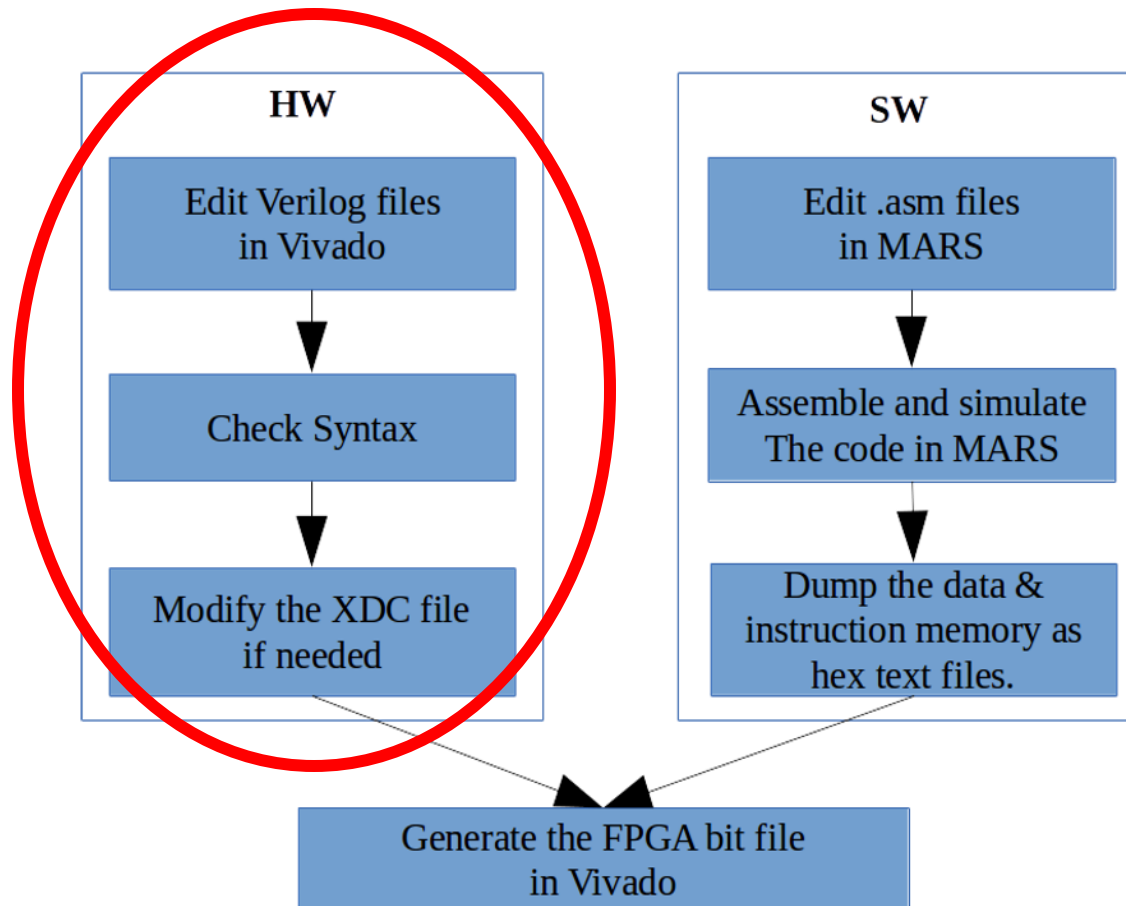
Address	Direction	Width	Description
0x00007FF0	out	28 bits	Value to be sent to the 7-segment display.
0x00007FF4	in	2 bits	Speed step 0, 1, 2 or 3.

Which 4 bits here are important? Remember that 4 bits in binary are one bit in hexadecimal

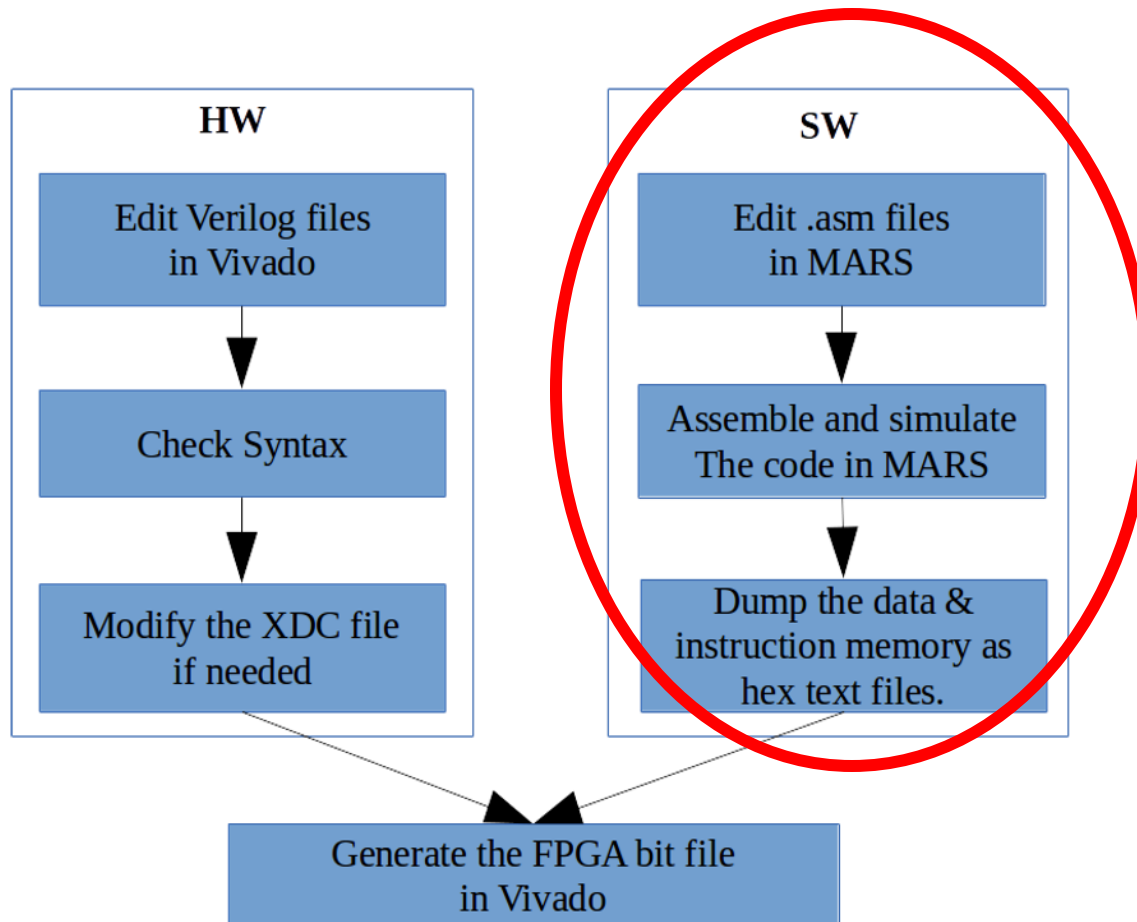
Lab 8 Session II: Summary of the Flow



Lab 8 Session II: Extending I/O



Lab 8 Session II: Modifying the Assembly



Last Words

- You will build a whole single-cycle processor and write assembly code that runs on the FPGA board.
- You will learn how a processor is built.
- Learn how the processor communicates with the outside world.
- Implement the MIPS processor and demonstrate a simple “snake” program on the FPGA starter kit.
- You will have some questions to answer in the report.

Report Deadline

[07. June 2024 23:59]

Digital Design & Computer Arch.

Lab 8.2 Supplement: Full System Integration

Frank K. Gürkaynak
Seyyedmohammad Sadrosadati

ETH Zürich
Spring 2024
[21. May 2024]