VLSI-2 Course Project (Phase II)

Project: Multicycle ARM Processor control Unit

Project Delivery: 1 Mordad 96

Multi-cycle Processor Completion

In this lab, you will complete your own multicycle ARM processor! Please refer to Harris Book for an overview of the processor. In phase I, you created the control unit. In this phase you will design the datapath and test your completed ARM multicycle processor.

This phase uses the same testbench and generic parts as in workshop for single cycle processor. Copy them from arm_single_xx.sv. Copy your `alu_xx.sv` file to your project directory. Also copy your `arm_multi_xx.sv` file containing your controller from phase I.

Test Program

Use the same test program and `memfile.dat` from the single cycle (given in Figure 7.60 of the book). Copy it to your project directory.

As in single cycle workshop, it is very helpful to first predict the results of a test program before running the program so that you know what to expect and can discover and track down discrepancies. Table 1, which is partially completed, lists the expected instruction trace while running the test program. Complete the remainder of the table. Do this before you run simulations so you have a set of expectations to check your results against; otherwise, it is easy to fool yourself into believing that erroneous simulations are correct.

Notice that the instruction (`Instr`) is fetched during the FETCH state and therefore not updated until the Decode state of each instruction.

When `ALUResult` will not be used (e.g. in the Writeback state of any instruction), you may indicate an 'x' for don't care rather than predicting the useless value that the processor will actually compute.

Datapath Design

Refer to Figure 1 in Lab 10 for the hardware modules you need to set up your datapath. Design the `datapath` unit in SystemVerilog.

Remember that you may reuse hardware from earlier labs (such as the ALU, multiplexers, registers, sign-extension hardware modules, register file, etc.) wherever possible.

All of your registers should take a *Reset* input to reset the initial value to a known state (0). The Instruction Register (IR) and PC also require enable inputs. Pay careful attention to bus connections when you instantiate a module; they are an easy place to make mistakes.

Simulate your processor using the testbench from Lab 9. The Reset signal is set high at first. Display, at a minimum, *clk*, *reset*, *PC*, *Instr*, *state*, *SrcA*, *SrcB*, and *ALUResult*. You will likely want to add other signals to help debug. Check that your results match the expectations from Table 1. If there are any mismatches, debug your design and fix the errors.

When you are finished – congratulations! You have built a microprocessor by yourself and have proven your mastery of microarchitecture, SystemVerilog, FSMs, and logic design!

Debugging

Hopefully your lab will have at least one error so you will get to hone your debugging skills! Here are some hints:

- Be sure you thoroughly understand how the ARM multicycle processor is supposed to work. This system is too complex to debug by trial and error. You should be able to predict what value every signal should be at every point in time while debugging.
- In general, trace problems by finding the first point in a simulation where a signal has an incorrect value. Don't worry about later problems because they could have been caused by the first error. Identify which circuit element is producing the bad output and add all its inputs to the simulation. Repeat until you have isolated the problem.

What to Turn In

Include the following elements **in the following order** in your final submission. Clearly label each part by number. Poorly organized submissions will lose points.

1. **Please indicate how many hours you spent on this lab.** This will not affect your grade (unless completely omitted), but will be helpful for calibrating the workload for next semester's labs.
2. A completed copy of Table 1 indicating the expected outcome of running the test program.
3. SystemVerilog code of the `datapath` module.
4. Simulation waveforms of the processor showing *clk*, *reset*, *PC*, *Instr*, *state*, and *ALUResult* in this order while running the test program. As always, output the values in hexadecimal and make sure they are readable. Do the results match your expectations? Does the program indicate Simulation Succeeded?

Cycle	Reset	PC	Instr	(FSM) state	SrcA	SrcB	ALUResult
1	1	00	0	FETCH	0	4	4
2	0	04	SUB E04F000F	DECODE	4	4	8
3	0	04		EXECUTER	8	8	0
4	0	04		ALUWB	x	x	x
5	0	04		FETCH	4	4	8
6	0	08	ADD E2802005	DECODE	8	4	C
7	0	08		EXECUTEI	0	5	5
8	0	08		ALUWB	x	x	x
9	0	08		FETCH	8	4	C
10	0						
11	0						
12	0						
13	0						
14	0						
15	0						
16	0						
17	0						
18	0						
19	0						
20	0						
21	0						
22	0						
23	0						
24	0						
25	0	18		FETCH	18	4	1C
26	0	1C	ADD E0855004	DECODE	1C	4	20
27	0	1C		EXECUTER	4	7	B
28	0	1C		ALUWB	x	x	x
29	0						
30	0						
31	0						
32	0						
33	0						
34	0						
35	0						
36	0						
37	0						
38	0						
39	0						
40	0						
41	0						
42	0						
43	0						
44	0						
45	0						
46	0						
47	0						
48	0						
49	0						
50	0						
51	0						
52	0						
53	0						
54	0						
55	0						
56	0						
57	0						
58	0						
59	0						
60	0						
61	0						
62	0						
63	0						
64	0						
65	0						
66	0						
67	0						
68	0						
69	0						
70	0						
71	0						
72	0						

73	0						
74	0						

Table 1. Expected Instruction Trace

توضیحات ضروری

در این فاز از پروژه Data path پردازنده چند سیکل تکمیل می شود. با مطالعه فایل arm_multi_xx.sv بخش هایی از مسیر داده که باید کامل شوند، مشخص می شود. شما می توانید از کد مربوط به پردازنده تک سیکل arm_single.sv نیز کمک بگیرید.

قبل از شبیه سازی باید جدول بالا که نشانگر مقادیر مورد انتظار پارامتر های مسیر داده در سیکل های مختلف هر دستور است تکمیل شود. این جدول برای debug کردن خطاهای احتمالی می تواند مفید باشد. سپس ضروری است،

برای شبیه سازی از فایل **ALU** خودتان و نیز فایل **memfile.dat** پردازنده تک سیکل نیز استفاده کنید. همینطور با کمک **test bench** پردازنده تک سیکل خودتان **test bench** پردازنده چند سیکل را در یک فایل جدا تهیه کنید. که از آن قبل و بعد از سنتز (در فاز ۳) نیز بتوانید استفاده کنید.

بنابراین در فاز دوم باید موارد زیر را در یک فایل **zip** شده تحویل دهید.

۱- جدول کامل شده فوق

۲- کلیه فایل های مورد نیاز تکمیل شده برای شبیه سازی که در یک فولدر به نام **arm_multi** قرار دارند.

۳- گزارش نهایی شبیه سازی که **Simulation successful** در آن مشاهده می شود. می توانید از **printscreen** استفاده کنید.

۴- شکل موج سیگنال های شبیه سازی شده **clk, reset, PC, Instr, state, and ALUResult** به همین ترتیب

مهم: در متن ایمیلی که پروژه را تحویل می دهید لطف کنید به صورت تقریبا دقیق مدت زمانی که صرف انجام فاز اول و دوم کرده اید را بنویسید، بدون شک اطلاع از این مدت زمان ما را در ارائه مفیدتر پروژه ی نهایی در ترم های بعدی کمک خواهد کرد و مطمئن باشید هیچ گونه تاثیری در نمره‌ی شما نخواهد داشت.