UG Exam Performance Feedback
Second Year
2017/2018 Semester 1

COMP22111 Processor Microarchitecture

Paul Nutter
Dirk Koch

Comments Please see the attached report.

COMP22111 Exam Feedback 2017/18

Q1 - PWN

This question was relatively straightforward. A minimal amount of revision should have enabled a student to have performed well. However, the quality of the answers was disappointing; the main issue being evidence from the answers that the questions had not been read properly before answering. It is important that you read and understand a question before you attempt to answer it.

The average mark for the question was 60%.

a)

The aim of this question was to discuss two design constraints (and not area, as this was given) – such as performance/speed, power, cost, area etc. – comparing the effect of these constraints with respect to designs produced using full custom or programmable logic devices. For example, in the case of clock speed, designs produced using programmable logic devices will invariably be slow because the hardware is designed and provided for you, and there is little opportunity to "tweak" the hardware design to improve the speed of operation. However, for full custom, you can optimize the design to maximise clock speed. Another example is cost. If a single device is required then the use of full custom design would be prohibitively expensive due to the design effort etc., but relatively cheap using a programmable logic device. However, if large quantities are required then full custom could then be more cost effective compared to the use of programmable logic devices. I was generous when marking awarding ½ mark each for identifying the constraints, and then a mark for comparing full custom and programmable logic devices with respect to the constraint. Issues were:

- The aim was to compare full custom and programmable logic devices with respect to the constraints, not just to explain what these are, or the difference.
- Just discussing the constraints and not relating the discussion to full custom and programmable logic devices! (A prime of example of not reading the question before answering.)

b)

The aim of the question was for you to explain situations where the three testing approaches are used. For example:

- Unit testing is often associated with the exhaustive testing of a single, simple design.
- Integration testing is conducted when individual units are brought together to form a more complicated design, and the testing focuses on checking the connectivity is correct.
- Regression testing is used when testing a design to ensure that further changes do not change the operation of the design.

As the 32-bit adder is composed of two 16-bit adder (which we assume have been unit tested) then integration testing should be performed to test the connectivity of the 32-bit adder design. Issues were:

- Not discussing the three approaches adequately.
- Discussions limited in detail.

c)

All that was required to answer this question was a discussion of the two addressing modes and provide a Stump instruction example.

- PC relative is only used in branch instruction, where a literal value, provided in the instruction is added to the PC to give a new address, i.e. BAL #7
- Indexed addressing is used in LD/ST instructions where an address is formed from a base address specified in a register in the instruction, to which an offset is added that can be found in another register specified in the instruction, i.e. LD R4, [R3, R2]

Issues were:

- Not providing a detailed enough discussion, or
- Not providing a Stump example (again, read the question)
- Stating a literal offset to a register value in the case of indexed addressing
- Not recognising that PC-relative only applies to branch instructions, any old instruction specifying the PC as one of the registers is not the same thing!
- Invalid Stump instruction examples (there is no excuse for this!)

d)

The answer required the discussion of the way the simulator uses active and non-blocking event queues with respect to the evaluation of the Verilog code given. The discussion was required to include the following points:

- Events are placed on the queue on the rising edge of the clock – time stops
- Only the RHS evaluation of blocking statements are placed on the active events queue – LHS assignment placed on the non-blocking queue
- Once the active events have been evaluated and the queue is empty, the assignments in the non-blocking queue are performed
- The change in Q results in the assign statement (Qbar) being added to the active events queue and evaluated.

Issues were:

- Presenting state transition diagrams, or timing diagrams – why? When have we done this when discussing the operation of the simulator?
- Limited discussion – just presenting a badly drawn diagram is not enough – details were required in the answer
- Not recognising the dependency of Qbar on Q and how the events queue is populated as a result of Q changing
- Not recognising that the event queue is populated on the rising edge of the clock
- Incoherent rambling discussions. It is important when writing a descriptive answer that you produce a coherent, structured discussion. If you simply write down your random thoughts, then you will probably end up losing

marks as it makes it extremely difficult for the marker to pick out the salient points.

e)

This question was an easy 2 marks. A mark was awarded for recognising that R0 is hard wired to zero, and ½ mark each for STUMP code examples for implementing MOV and CMP:

- MOV: ADD R3, R2, R0 – R3 => R2+0, so the contents of R2 are “moved” to R3
- CMP: SUBS R0, R2, R1 – perform R2-R1, update flags, but no result is written, as the result is written to R0 (which is always 0).

Issues were:

- Not stating that R0 is hard wired to zero and is always 0 (again, read the question)
- CMP being implemented as ADD
- Missing the S from SUBS – this is required for the flag update to occur
- Not setting the destination for the SUBS as R0, as CMP should not update any registers.

Q2 - DK

This question was on scaling and process technology which is covered more than once throughout the course unit.

Overall this question was well answered and some students got full marks. However, quite a number of students failed entirely on this question. This is a bit surprising as just analytical thinking would allow to collect several marks. For example, the question on "how many flash transistors are needed for a 64 GB memory card" required only the knowledge what a "giga" means and that for memory applications we have to think in bits rather than bytes. Also the question on the impact of CMOS shrinking could have been solved by simply drawing boxes and comparing different feature sizes with each other (two students solved the question this way). Unfortunately, about a third of the students thought that the impact of shrinking has linear impact on the number of transistors that can be fit which is wrong as technology shrinks in more than one dimension.

Q3 - PWN

On the whole, this question was answered relatively well. The question focused on testing your understanding of the Stump datapath, and it was clear that most students did. The second half of the question focuses on writing Verilog code to describe the behaviour of the Stump sign extender and also test its operation. Considering the amount of Verilog coding performed in the lab, I was very much surprised about the amount of misunderstanding here.

The average mark for the question was 64%.

a)

This question required you to briefly (and briefly is key) describe the three instruction formats used in Stump. I was looking for the following points:

- Type 1: operands provided by two registers specified in the instruction, with the result being written to a register that is also specified (1 mark). The key feature of a Type 1 instruction is that a shift operation can be applied to the operand from srcA (1 mark).
- Type 2: one operand is provided by a register specified in the instruction and the other is a 5-bit literal value also supplied in the instruction (½ mark for stating this is 5 bits). The result is written back to a register specified in the instruction (½ mark).
- Type 3: conditional branch instruction (½ mark) where an offset to the PC is provided as an 8 bit value in the instruction (½ mark).

Generally, this was answered well. Some minor issues were

- Not stating where the result goes.
- Not specifying the number of bits for the immediate (Type 2) and offset (Type 3) values specified in the instructions.

b)

This question tested your understanding of the Stump datapath and its operation by asking you to identify errors in the signal usage charts provided for the four cases given. In each case you were required to identify the errors and, if appropriate, discuss why it is an error and how it should be fixed. There were two marks available for each question. The errors were:

a. Here there were three errors, you are only required to identify two errors for the two marks. This is a Type 2 instruction so the second operand should not come from the register bank, but from the IR, via the sign extender, so the appropriate path needs to be identified. The address bus is shaded indicating that a value is placed on the address bus – this isn't the case and so should be removed. The data writeback path mux1 to register bank should be shaded, as the result should be written back to the register bank.
b. The PC is not being updated (+1) when it should be in the execute phase. As such, paths from srcA -> shifter -> ALU, '1' into ALU from mux3, and result back into the register bank via mux1 should be shaded - I gave the full two marks for a complete description. In addition, the address line should be shaded, since the PC needs to go out on the address bus.

c. This is the memory operation for a store instruction, however, the signal usage chart implies a load operation. Consequently, the shaded path data_in -> mux1 and IR should be removed (1 mark) and instead srcA should go out to memory via data_out (1 mark).
d. Here the writeback path where the result of adding 36 to the current value of the PC is not shown to be written back to the register bank. Hence, alu_out->mux1->register bank should be shaded (1 mark). Also, no address goes out to memory, so the address bus should not be shaded (1 mark).

On the whole, the question was answered very well, and I was pleased that overall students had a good grasp of the operation of the Stump datapath. Some minor issues were:

- Not identifying that the address line is not used for a) and d). Some answers stated that the value shouldn't be loaded in to the address register, this doesn't matter. Key is that shading the address bus indicates that a memory operation should be performed, when it shouldn't.
- Not recognising any errors in c).
- Saying data_out should be shaded in b) –the address from the PC only goes out on the address bus.
- You should not discuss the status of control signals – such as the address register shouldn't be enabled etc – the exercise simply requires you to identify the correct path usage for each case.

c)

In this question, you are required to produce the missing Verilog code for the sign extender module. It was a mixed bag in terms of the "quality" of code produced. However, on the whole almost everyone did well on this question. The easiest way of implementing this is to use concatenation, i.e.

```
always @ (*)
if (select)
  immed = {{8{ir[7]}}, ir[7:0]};
else
  immed = {{11{ir[4]}}, ir[4:0]};
```

Marks were awarded for the correct operation, valid Verilog code, and identifying the always block as being combinatorial. In some cases a set of { and } brackets were missing, however, I didn't knock marks off for this. Issues were:

- Identifying the wrong bit as the sign bit.
- Only appending zeros and not copying the sign bit.
- Providing incomplete code.
- Invalid Verilog code.

d)

Here, you were required to produce a Verilog testbench for the sign extender of part c). You are required to define the signals used, instantiate the design-under-test and then use sensible test data to test the design. An example would be:

```
reg [7:0]      ir;
reg            select;
wire [15:0]    immed;

sign_extender stump_se(.ir(ir),
                       .select(select),
                       .immed(immed));

initial
begin

  // set inputs
  // delays between

  $stop
end
```

Marks were awarded for correct definition of signals, correct instantiation of the DUT, having an initial block with a $stop in it, and for using sensible test values. Issues were:

- Implementing code for a clock – where is a clock mentioned in the question (again, read the question), or even used in a combinatorial design?
- Not defining the signals properly.
- Not instantiating the DUT.
- Not providing sensible test data – you need to test that sign extension of bit 11 and bit 4 work depending on the value of select. In some cases, what appeared as random data, or all 0s or 1s, were used, which might not necessarily given you an indication that the sign extender actually works.
- Invalid Verilog code.
- Wasting time using for loops – no need here, the testing is relatively straightforward, exhaustive testing is not necessary.

Q4 - DK

This question is about pipelining and multithreading. More than half of the questions are pretty standard and are basically assessing knowledge about fundamental principles. About half the students succeeded well on this. The comparison of pipelining versus multithreading assess beyond this as it, for example, requires to predict how a specific workload would perform on the different variants. About a quarter of the students performed well on this. However, about a quarter of the students have not really tried this question at all. This is a bit surprising as it was clearly communicated that all exam questions are mandatory this time and because pipelining is a rather frequent exam topic.