SoC D/M Exercise Sheet, 2014/2015

This sheet contains exercises of various lengths. Many exercises are nominally allocated marks at Tripos examination level (i.e. with 20 marks making a full exam question).

There is some repetition of material between the exercises, so a suitable target is to solve approximately one third of them. Exercises marked with a ♡ form a recommended core. Some sections contain additional preference instructions.

Example answers are available to supervisors.

SP1: SoC Components and Bus Structure Exercises

Slide Pack SP1: This sheet should be tackled for the first supervision. Supervisors are recommended to set just the few exercises marked with the ♡ symbol, leaving the second half of this sheet for brief discussion or for self-study revision purposes.

This year SystemC is being lectured at the end, so please leave the SystemC parts of this sheet for now and answer again after that has been lectured.

SoC-SP1.1 What is meant by polled I/O and how does it compare with interrupt driven I/O ? [4 Marks]

SoC-SP1.2 **Done as an example in lectures.** Swapping bit order in a word requires an expensive sequence of instructions on many processors. However, it can be done instantly in wiring. It is proposed to provide an I/O device with three registers respectively of width 8, 16, 32 bits. Any word written to such a register can then be immediately read back with its bits swapped.

a) Give the programming model for such an I/O device and sketch RTL or the circuit diagram for the actual device. [4 Marks]

b) How should such a device be connected to the processor such that it actually saves time or energy compared with the sequence of instructions approach? Discuss various wiring possibilities and their pros and cons [6 Marks].

For your reference, here is some C code (from the ARM ISS), one quarter of this meets one of third of our specification:

```
switch(size_code)
  {
  case 0:  //Reverse byte order in a word.
    d = ((d>>24) & 0xFF) | ((d>>8) & 0xFF00) | ((d<<8) & 0xFF0000) | ((d<<24) & 0xFF000000);
    break;
  case 1: // Reverse byte order in each halfword independently.
    d = ((d>>8) & 0x00FF00FF) | ((d<<8) & 0xFF00FF00);
    break;
  case 2://   RBIT - (a full butterfly) reverse the bit order in a 32-bit word.
    d = ((d>>16) & 0x0000FFFF) | ((d<<16) & 0xFFFF0000);
    d = ((d>>8) & 0x00FF00FF)  | ((d<<8)  & 0xFF00FF00);
    d = ((d>>4) & 0x0F0F0F0F0) | ((d<<4)  & 0xF0F0F0F0);
    d = ((d>>2) & 0x333333333) | ((d<<2)  & 0xCCCCCCCC);
    d = ((d>>1) & 0x555555555) | ((d<<1)  & 0xAAAAAAAA);
    break;
  case 3: //REVSH      Reverse byte order in the bottom halfword, and sign extend to 32 bits.
    d = ((d >> 8) & 0xFF) | ((d<<8) & 0xFF00);
    if (d & 0x8000) d |= 0xFFFF0000;
    break;
  }
```

SoC-SP1.3 Sketch a set of typical macro definitions in C suitable for making low-level hardware access to the device of the previous question or else to a UART or similar device that contains status, control and data registers. [4 Marks]

SoC-SP1.4 ♡ Show how to wire up a push button to a GPIO pin and sketch out the code for a device driver that sets up the GPIO device and returns how many times it has so far been pressed. Sketch polled code and then write a few sentences about how the interrupt driven solution would be implemented and when it should be used instead. (No credit is allocated for proper debouncing so feel free to neglect that aspect). [10 Marks]

SoC-SP1.5 ♡ A bus arbiter decides which initiator next makes access to the set of shared targets connected to that bus. In simple systems these have fixed behaviour and so do form part of the programming model. Here we consider an arbiter with programmable policy.

a) Sketch the circuit, RTL or SystemC code for a programmable bus arbiter that manages two initiators. Start by specifying the behaviour and then defining a suitable programming model. Then list the net-level connections to the component to draw a schematic symbol before going on to implementation details.

b) Should the programmed I/O access to the bus arbiter itself be subject to arbitration and if this were to be a problem, in what ways could higher-priority access to the arbiter's register file be granted?

c) What are the advantages and disadvantages of having an arbiter programmable? What method might a network-on-chip used to get much richer programmable arbitration?

SoC-SP1.6 Sketch the RTL or SystemC code for an interrupt controller that stores eight vectors with individual interrupt enable flags. The controller monitors eight interrupt inputs and presents the highest-priority, non-masked interrupt vector to the processor when the processor asserts an interrupt acknowledge signal or otherwise reads the device. [Start by defining a suitable programming model and then list the net-level connections to the component to draw a schematic symbol.] *Fine details will vary from answer to answer. As always, syntactic accuracy is not required in SoC D/M examination answers.* [15 Marks]

SoC-SP1.7 How does the processor set up the interrupt distributor device of SoC-SP1.6 and what must it do after servicing an interrupt ? [4 Marks]

SoC-SP1.8 How would you make an interrupt distributor that shares work over two CPUs? Is dynamic distribuition of interrupts always a good idea ? [6 Marks]

SoC-SP1.9 Give a programming model for a simple DMA controller with one control/status register and three operand registers for block length and source and destination addresses. *The DMA (direct memory access) controller, when active, becomes a bus master and copies a block of data from one area to another, generating an interrupt on completion. n.b. This is very similar to the dma_controller.h example in the TOY-ESL online practical classes.* [4 Marks]

b) Sketch a full implementation of such a DMA controller that includes provision for slave access to the programmable registers, active bus mastership and interrupt generation. Memory access should use a high-level modelling style that ignores bus arbitration. *Answer preferably using SystemC syntax, or pseudocode at the same level of abstraction. Use RTL if and where needed or preferred.* [7 Marks]

SoC-SP1.10 Bus Bridge.

a) What is the function of a bus bridge in a SoC ? [2 Marks]

b) What typical address translation semantics might a bus bridge implement ? [4 Marks]

c) How might internal queue structure vary between bus bridge designs ? [3 Marks]

d) How might arbitration policy vary between bus bridge designs ? [3 Marks]

SoC-SP1.11 Input and Output to a Network Controller

a) Sketch the structural schematic symbol for a generic network block that is bus target only, giving full details and descriptions of the signals used to connect to a typical system bus. *The network type or internal structure does not matter, it could be Ethernet, USB, Firewire etc..* [6 Marks]

b) What advantages are there to giving the network block the capability of being a bus master? [2 Marks]

c) Describe the additional signals needed to make the network block a bus master. [6 Marks]

d) Assuming the device can be a bus master, sketch the code for a typical device driver. [6 Marks]

SoC-SP1.12 Define a feasible, net-level, serial interface used between a sound controller device (That does DMA and so on) to an audio output DAC (digital-to-analog convertor). The interface conveys a pair of stereo channels of 16 bit precision at 44.1 ksps. *Hint: Three nets are normally used.* [4 Marks]

SoC-SP1.13 Sketch the block diagram or RTL for a simple audio output controller that uses DMA to send a serial audio data-stream to a DAC. Include the full programmers' model. [12 Marks]

SoC-SP1.14 Clock Domain Crossing.

a) List basic principles used in the design of a reliable clock-domain crossing bridge to avoid metastability problems and achieve reliable transfer of data ? [6 Marks]

b) Sketch the RTL or block diagram for a simplex clock crossing bridge that internally uses one parallel data bus and four-phase handshake ? *If giving RTL, only the receiving side logic is needed.* [6 Marks]

c) What constraints exist for simplex protocols that cross clock domains ? [6 Marks]

d) What constraints exists for duplex protocols that span clock domains ? [2 Marks]

SoC-SP1.15 Exercise: sketch RTL code for a non-preemptive version of the 3-input arbiter given in the handout. Alternatively, provide RTL code for a round-robin, non-preemptive version of the 3-input arbiter. *An asynchronous implementation is quite tricky unless you are experienced at logic designing with transparent latches and other level-sensitive latches, so feel free to present a synchronous design, which is just a finite-state machine.*

 END OF DOCUMENT.