SoC D/M Exercise Sheet(s) 6, 2016/2017

This sheet contains exercises of various lengths, but the very short ones are in a separate PDF file (called something like socdamquick.pdf). 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.

Supervisors are recommended to mainly just set the exercises marked with the ♡ symbol, leaving other parts of the sheets for discussion or for self-study revision purposes. Example answers are available to supervisors.

The course has been rejigged a few times in the last ten years and, despite my attempts, it is likely that one or two of the questions here are now no-longer-relevant or else under the wrong heading. SystemC is no longer being lectured for net-level modelling: it is just presented as a basis for TLM modelling, so where a question suggests giving a SystemC descriptons instead of RTL, that route will not be available unless you have separately studied low-level SystemC coding.

SP6: ESL (Electronic System Level) and TLM Exercises

6.1 (SystemC Net-Level Modelling - Full Questions)

Note: In 2015/16 and 16/17 - SystemC was almost entirely removed from the course. SystemC was used in the concrete examples of TLM ESL Modelling, Net-level modelling in SystemC was only briefly mentioned. So the following (SYSC) questions are not appropriate this year and marked with a †.

SYSC1. † Describe the principle features of SystemC. [5 Marks]

SYSC2. † With what user syntax and how internally is an RTL-style non-blocking assignment achieved in SystemC? [5 Marks]

SYSC3. † How is design module heirarchy expressed in SystemC?

SYSC4. † ♡ Why adapt a general-purpose language like C++ for hardware use when special hardware languages exist? [2 Marks]

SYSC5. † ♡ To what level of detail can a gate-level design be modelled using SystemC, would one ever want to do this and what simulation performance might be achieved? [5 Marks]

SYSC6. † Give a fragment of SystemC or RTL that relies on its kernel scheduler to correctly implement non-blocking updates (avoiding shoot-through) and then give an equivalent fragment of pure C that has the same behaviour but which does not need support from a scheduler or other library. [10 Marks]

SYSC7. † How does SystemC help model registers that have widths not native to the C language? [4 Marks]

SYSC8. † Give synthesisable SystemC for a five-bit synchronous counter that counts up or down dependent on an input signal. *You should sketch C++ code that looks roughly like RTL rather than worrying about a precise definition of synthesisable for SystemC.* [5 Marks]

SYSC9. † Define suitable nets for a simplex interface that transfers packets of 3 bytes over an asynchronous eight-bit bus with a protocol that is based on the four phase-handshake. Describe the protocol. *Answer this part using RTL, timing diagrams or natural language.* [5 Marks]

SYSC10. † Sketch SystemC RTL-like code for a synthetic data generator that creates three byte packets and delivers them over the four-phase interface of SYSC9.. *Precise syntax and operational details are unimportant, but a sensible answer would be a Verilog module that puts a counting sequence in the packet payloads.* [5 Marks]

Practical Exercise: Using the RTL-style blocks provided in the 'toy classes' with the nominal processor, please experiment with various configurations and understand how you would make a more-complex system using more of the addr_decode and busmux components to address the components.

6.2 ESL (Electronic System Level) and TLM Exercises - Exercises

The dagger † symbol marks SystemC net-level exercises that are no-longer examinable.

ESLm1. Define a transaction in Computer Science. How does the ESL use of this term differ ? [5 Marks]

ESLm2. Sketch a TLM model as a generic (templated) class in an OO language for a basic FIFO with capacity of 8 items. [8 Marks]

ESLm3. Sketch code that will instantiate two such TLM FIFOs together and connect them together to make a longer TLM FIFO model. [5 Marks]

ESLm4. † Sketch synthesisable SystemC or RTL-like code for such a FIFO (using either a circular buffer in a RAM or else based on a multi-stage structure). *This is rather straightforward exercise, but it is useful preparation for the next one!* [5 Marks]

ESLm5. † Sketch code for a transactor (one of several possible) that enables interworking between the TLM and Synthesisable FIFOs of ESLm2. and ESLm4.. [5 Marks]

ESLm6. What is the difference between a blocking and non-blocking transaction in terms of implementation, efficiency and callability? [6 Marks]

ESLm7. ♡ Sketch SystemC code for a shim function that converts a transactional port from blocking to non-blocking, or vice versa. (*n.b. One direction is harder than the other*). [5 Marks]

ESLm8. ♡ Rather like the net-level model you may have given for SYSC8., give a TLM model in an OO language of your choice, for a five-bit synchronous counter that counts up or down dependent on an input signal. Give a schematic symbol or RTL signature for the net-level implementation. Add a further transactional entry point to the module that allows a remote client to perform a synchronous parallel load, that sets the counter to a specific value. Roughly how many gates might this component use in reality? Make observations about the scale of subsystem that is suitable for TLM modelling. What happens to the clock input in your TLM model? The difference between a synchronous and an synchronous parallel load is quite significant in an RTL model or real model. What difference does it make with TLM? [10 Marks]

ESLm9. † Assume TLM calling is not synthesisable, but basic RTL-style SystemC can be converted to gates. Restructure your answer of ESLm8. so that the five-bit counter has a net-level parallel load and so remains synthesisable. Then illustrate how to use a transactor to provide the TLM parallel load entry point into the now-supported, net-level parallel load. (You may ignore contention with other, simultaneous net-level operations on the counter.) [7 Marks]

ESLm10. Here is some simple code for a net-level data generator consisting of a behavioural model of the data generator core and a transactor that exercises the hardware-level nets:

```
void write4p(unsigned char d)            // Transactor
{
  do (wait(clk.posedge_event()) while (ack.read())
  data = d;
  req = 1;
  do (wait(clk.posedge_event()) while (!ack.read())
  req = 0;
}

unsigned char x;
while(1) instance.write4p(x++);   //And here is a client for it.
```

Sketch code for a further part of a baroque system where there is another transactor that owns its own thread and is a client for this net-level interface which makes an upcall to a user-provided function for each byte received. I call this baroque because, ideally, we normally do not want to first convert between modelling styles and then convert straight back. [4 Marks]

ESLm11. ♡ *a)* Why might embedded firmware be cross-compiled to native code for a workstation ? Would this be automatic or manual? [5 Marks]

b) Give two or more ways hardware device access can be modelled when firmware including device drivers is cross-compiled for the modelling platform. [5 Marks]

ESLm12. Describe at least two ways that caches might be modelled for a SoC and say how accurate they will be. [5 Marks]

ESLm13. ♡ Sketch a TLM model of a bus bridge and say what arbitration, queuing and address translation policies it implements. *Hint: a high-level model will likely lead to the shortest answer. It can be about six lines of code per direction. Syntax details are unimportant and, as always, pseudocode is acceptable.* [8 Marks]

ESLm14. a) Sketch a block diagram for a SoC containing at least two identical processor cores, a DRAM controller and some amount of on-chip SRAM. Mark each end of each connection with a suitable port style to be used as part of a TLM model (eg. blocking, non-blocking, initiator, target). Does your design allow simultaneous access to the DRAM by one core while the other is using the SRAM? [10 Marks]

b) What instrumentation does the TLM model need for us to find the system performance bottleneck? [5 Marks]

c) How could we rapidly explore the best positioning and dimensioning of caches and bus widths?

ESLm15. Roughly estimate (order of magnitude) how many workstation instructions are used when modelling each access to the DRAM of ESLm14.. [5 Marks]

6.3 ESL Performance Modelling - Exercises

ESLt1. ♡ Define each type of modelling: cycle-accurate, approximately-timed, loosely-timed, untimed. [8 Marks]

ESLt2. ♡ What is the main advantage of loose timing? [3 Marks]

ESLt3. How can the degree of re-ordering of simulated transactions be constrained? When must re-ordering be avoided and how? [3 Marks]

ESLt4. Consider what simulation performance an ISS might give and can it ever be faster than real time ? [5 Marks]

ESLt5. ♡ How can contention for a resource be modelled with and without actual queuing of the transactions? [5 Marks]

6.4 ESL Power Prediction - Exercises, Tripos-like

For the questions in this section, think widely about what factors are really likely to influence energy use in the design and then consider whether an ESL model that embodies these factors is ultimately any more use than a carefully-crafted spreadsheet.

ESLp1. A ceiling-mounted smoke sensor with wireless backlink is specified to be reliable provided it has a new battery installed every ten years.

(a) Assuming a time-to-market to develop the new design of ten plus years is unacceptable, what should be the principle design approach for the energy budget? [10 Marks]

(b) List the assumptions or figures you will need to make for this design approach. How will you evaluate your sensitivity to these assumptions? [6 Marks]

(c) You may have used both an executable model and a spreadsheet in your approach: explain whether you could manage with only one of these. [3+3 Marks]

ESLp2. Two teams are attempting to evaluate convolutional neural networks in hardware with low energy. One is using FPGA and the other ASIC. How much of a common model can they use to predict performance and energy and how will this model work? What will they compare it with? (If you are unfamiliar with CNNs, then answer for an L/U decomposition of a constant 1000x1000 matrix where a near-infinite stream of right-hand-sides need to be solved for.)

ESLp3. Pollack's Rule states that microprocessor 'performance increase due to micro-architecture advances is roughly proportional to the square root of the increase in complexity'. Hence, where a program can easily be parallelised, it is always better to have multiple small cores executing it compared with fewer, higher-performance cores. Leg Microsystem's new core can run in two modes: simple in-order and complex out-of-order. The instructions per clock (PIC), on average, in each mode, is 1.5 and 3.5. The core also supports DVFS where

voltage scaling is applied between the 300 to 800 MHz clock rates of 0.95 to 2.0 volts. It almost goes without saying, but there is also the extended bottom clock range, where cores can be operated between 0 and 300 MHz at 0.95 volts.

(a) What, by Pollack's rule, is the expected amount of logic turned on within the core in the simple mode? [3 Marks]

(b) What is the expected performance and power ratio between the 300 MHz simple mode and the 800 MHz complex mode? [3 Marks]

(c) How might an ESL model of the core encompass the various modes? [5 Marks]

(d) For an embarrassingly-parallel workload (like Mandlebrot), what is the best number and style of cores, without considering uncore, for a throughput of 5 GIPS (giga instructions per second). Note 'uncore' is a colloquial term referring to all of a SoC except for the cores and L1 caches. [5 Marks]

(e) Would a TLM model be helpful in working out whether the uncore costs affect the design decision of the previous part? [This is a little open-ended but good for supervision discussion.] [5 Marks]

6.5 ESL Example Tripos Questions

ESLt1. a) Give the programming model for a simple DMA (direct memory access) controller, like the one in the lecture notes, with one control/status register and three operand registers for block length and source and destination addresses. *The DMA 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. You may find this is similar to the dma_controller.h example in the TOY-ESL practical materials.* [4 Marks]

b) Augment your system diagram of ESLm14. to include one or more instances of your DMA controller. [4 Marks]

c) Sketch a full TLM 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 using any high-level language syntax or object-oriented pseudocode, You may use RTL if and where needed or preferred, but this will be longer.* [7 Marks]

ESLt2. The Dalvik compiler and VM takes Java bytecode and converts it to virtual register machine code. But register machine code is generally considered[1] to be poor for software interpreting owing to the parallel extraction of the various bit fields inside the instruction being serialised compared with the parallel execution on a hard processor. On the other hand, simple bytecodes are considered efficient to implement in software using an indexed jump to handler code for each instruction. Threaded code is even better but not as dense. Hence dynamic expansion to threaded code, on-the-fly, is potentially a good idea. [2].

(a) Assuming you had C++ implementations and testsuites for all 3 approaches readily available, how would you use an ESL model to compare their relative performance? [5 Marks]

(b) What features might your ESL model expose about their relative performance that are not likely to be obvious just from running the C++ implementations on your workstation? [5 Marks]

(c) A new implementation approach now emerges, where a custom co-processor is used to accelerate certain VM operations. Your ESL model has been extended to wth the co-processor and you are confident it will give an accurate performance prediction of the real ASIC when made. Suggest two ways you can start to add power annotations to the co-processor. [10 Marks]

7 HLS: High-Level Synthesis

HLS was lectured and examinable for the first time this year. IP-XACT is not examinable this year and we did not get to the topics at the end, so there are quite a few questions marked with a † as unexaminable this year (16/17).

[1] Some patent attorneys are paid to disagree on this.

[2] Threaded code. `https://en.wikipedia.org/wiki/Threaded_code` Each basic block of the user's code is compiled to a list of subroutine calls but the PC is not virtualised. We would not ask about threaded code in a real Tripos question but it's worth private study. It is not the same as multithreaded-code!

HLS1. † IP-XACT

a) What is the purpose of the IP-XACT specification ? [5 Marks]

b) How can device driver register definitions be kept in step with RTL implementations ? [5 Marks]

c) What alternatives to IP-XACT might be considered for structural netlists ? [5 Marks]

d) How might IP-XACT be used in conjunction with transactor synthesis ? [5 Marks]

HLS2. Consider the following multiplier code:

```
fun booth(int32 x, int32 y, int32 c, int32 carry) =
    if x=0 && carry=0 then c else
       let (x', n) = (x div 4, x mod 4 + carry)
       let y' = y * 4
       let (carry', c') = case n of
            | 0  => (0, c)
            | 1  => (0, c+y)
            | 2  => (0, c+2*y)
            | 3  => (1, c-y)
            | 4  => (1, c)
        booth(x', y', c', carry')
```

a) Speculate on whether the code in this form is likely to be synthesisable by today's HLS tools in its current form.

b) Does a 32-bit multiplier return a 32 or 64-bit result? What does the example return and what, if any, changes might be needed for a 64-bit result ?

c) Manually convert the code into a non-recursive form.

d) What external handshaking nets would be needed for a hardware implementation that made one iteration per clock cycle?

e) Looking at your non-recursive form, collate all of the expressions ever stored in each register, hence forming a custom datapath for executing this function. You may wish to rename the input formal parameters as `x_in` and `y_in` to distinguish their argument busses from the internal state registers with those names.

f) Draw the state-transition diagram for a sequencer that implements the external handshake and controls multiplexors and/or any other resources in your custom datapath.

g) Estimate the area use in gate-equivalent units. Count a ripple-carry adder stage as 4 gates, a multiplexor as 3 gates per bit and a D-type flip-flop as 6 gates. h) Estimate the critical path of your design hand hence the minimum clock period in units of average gate delay.

i) For the really keen: Manually unwind your inner loop by a factor of two so that the processing latency should be nearly halved: give both the revised imperative code and the new custom datapath.

j) Statically determining the needed maximum width for each data path connection is important for synthesiser tools to trim logic and save energy and area. For the really keen: Can you give a small set of width-inference rules that would be used in practice to statically determine the necessary width of the two loop-carried carry/accumulator values (sic)?

HLS3. You are given a library of instantiatable RTL blocks that perform double-precision floating point addition, subtraction, multiplication and division. Each is fully pipelined and they have latencies of 4, 4, 5 and 12 cycles respectively. The signature of each block is simply a clock input, two data inputs and one data output, where the data are all 64-bit wide busses. You also have combinational blocks (or Verilog functions if you prefer) that convert from small integers to double-precision representations.

You are to evaluate the sine function for an input known to be in the range $\pm\sqrt{2}$ using Taylor's expansion as follows:

```
fun sine x =
   let sq = - x * x
   let sine1 n sofar diff =
       let diff = (diff * sq / (n * (n+1))
       let x = diff + ans
       in if diff < margin then return x else sine1 (n+2) x diff
   in sine1 2 x x
```

a) Draw a block diagram of a custom datapath for this problem that uses as many block instances as you like.

b) Draw a detailed static schedule or timing diagram for your solution.

c) Assuming you could easily work out the maximum number of iterations ever needed (which is an exercise for another course), what is the overhead of making a fully-pipelined implementation instead?

HLS4. A well-known algorithm has the following code:

```
procedure wikipedia_bubbleSort( A : list of sortable items )
   n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do  /* if this pair is out of order */
       if A[i-1] > A[i] then  /* swap them and remember something changed */
         swap( A[i-1], A[i] )
         swapped = true
       end if
     end for
   until not swapped
end procedure
```

A design is needed that implements this algorithm for 32768 32-bit integers where the items are held in a single-ported, synchronous static RAM with the following signature:

```
module SRAM_32768_32_FL1(
   input clk,
   input [14:0] addr,
   input wen,
   input [31:0] din,
   output [31:0] dout);
```

On any clock positive edge where the `wen` signal holds, the RAM will write the value on `din` to the address on `addr` and the old contents of that address will also come out on `dout` fairly early in the following clock interval and remain there until just after the next clock positive edge. On cycles where `wen` is not asserted, the RAM is unchanged but readout is the same. In other words, the RAM has a fixed latency of one cycle.

a) Is this algorithm amenable to loop forwarding ?

b) Determine a custom datapath that can be used to implement this algorithm and describe the purpose of any holding registers needed.

c) Compute the worst-case run time in clock cycles.

HLS5. † Bluespec System Verilog.

a) What is a Bluespec rule (guarded atomic transaction) ? [5 Marks]

b) How is the programmer's mental model of parallel programming expressed in Bluespec Verilog compared with conventional RTL ? [5 Marks]

c) What is the code explosion problem in high-level synthesis and how does Bluespec avoid it ? [5 Marks]

d) How does higher-level design expression potentially help with timing closure ? [5 Marks]

HLS6. Kiwi Project (C#-to-gates via .net).

a) What is a `generate` statement, as found in VHDL and Verilog, and how is the same effect achieved in Kiwi ? [5 Marks]

b) What do parallel programming and hardware design have in common ? [5 Marks]

c) How have the designers of the Kiwi system exploited this ? [5 Marks]

d) What might be the interpretation of a thread fork and join in general C-to-gates flows and in Kiwi in particular ? [5 Marks]

HLS7. † UML For VLSI Design (Marte Project).

a) What are the basic roles of a graphical cockpit, such as Eclipse or another GUI, in SoC design ? [5 Marks]

b) Give an overview of the Marte Project and explain the potential roles of UML in SoC design ? [5 Marks]

c) How does UML differ from IP-XACT as used in SoC design ? [5 Marks]

d) List several visualisation tools could usefully be offered in an integrated development environment for SoC design, debugging and evaluation ? [5 Marks]

HLS8. † Glue Logic Synthesis.

a) What is the data conservation principle in interface design ? [2 Marks]

b) List the major steps in the product method for glue logic synthesis. [5 Marks]

c) Give four (or more) commonly appearing component joining paradigms. [8 Marks]

d) Using example fragments of RTL or SystemC-like glue code that joins a pair of interfaces, explain what the user needs to define and what might be synthesised. (*You might choose, as your example, a duplex mailbox that offers blocking target ports on both sides.*) [5 Marks]

HLS9. † Transactor Synthesis.

a) What is a transactor, as used when mixing different modelling styles in ESL ? [5 Marks]

b) Explain why it might be useful for a common protocol specification to be used both to synthesise bus monitors and to synthesise transactors. [5 Marks]

c) Name three typical transactor configurations (and explain why the obvious fourth is potentially useless). [5 Marks]

d) Can glue logic be synthesised from transactor definitions ? [5 Marks]

 END OF DOCUMENT.