Part II CST: SoC D/M: Quick exercises (examples sheet) (rev 3rd May 2012).

This sheet contains short exercises for quick revision. Please also try some of the longer exercises from the other sheet.

1 LG 1 – Register Transfer Language (RTL)

LG1.1. Give an RTL structural netlist for an RS latch.

LG1.2. Give a fragment of RTL that uses non-blocking assignments (in Verilog the `<=` operator) to swap a pair of bytewide registers on a clock edge when an externally-provided enable signal holds. Sketch the resultant circuit (or gate-level, structural netlist) that would arise when your fragment is compiled to gates.

LG1.3. Give a fragment of RTL that uses blocking assignments (in Verilog the `=` operator) and give the resultant circuit (or structural netlist) showing clearly the difference that would arise if non-blocking assignments were used instead.

LG1.4. The following fragment of RTL produces a repeating pattern on a one-bit output. Assuming it is accepted by some gate synthesis tool, give the resultant circuit (or structural netlist) that it might sensibly generate.

```
    always begin
         @(posedge clk) a<= 1;
         @(posedge clk) a<= 0;
         @(posedge clk) a<= 0;
         end
```

LG1.5. Consider the following fragment in a fictional BlueSPEC-like language. This is to be synthesised into a standard Verilog (or VHDL) RTL for further synthesis to gates using standard RTL tools. Assume that the `write` and `read` methods each add the following wires to the signature of the generated component: a request input, a ready output and an 8-net data transfer input (or output for read). Give the resultant circuit (or structural netlist that the tool might sensibly generate. The handshake convention is that method call takes place on a positive clock edge where both request and ready hold.

```
module InvertingRegister( ... )

reg [7:0] contents; // Basic RTL register (unguarded).

method write(d) { contents <= d; }

method read() { return ~d; }
```

Suppose there is a side condition that read and write cannot occur in the same clock cycle: modify your answer to reflect this. *Hint: the difference would be in the generation of one of the ready outputs.*

2 LG 2 – Simulation

LG2.1. Tabulate the events inserted in an event list when simulating one clock cycle of your net-level hardware design from question LG1.2.. Assume each gate has a delay of 100 picosceonds.

LG2.2. Modify your net-level hardware design from LG1.2. by adding an inverter so that one of the bits is swapped in polarity when moving between one register to the other. What extra events would arise in the net-level simulation. What has happened to the maximum clock frequency of the design?

3 LG 3 – Hazards

LG3.1. Consider a FIFO component that accepts 16-bit writes and provides 8-bit reads. Internally it is implemented with a 128-byte SRAM with organisation 128x8. A revised implementation uses an SRAM with organisation 64x16. Which implementation offers better throughput and which (if any) suffers from structural a hazard ?

LG3.2. The writeback pipeline stage in a simple RISC processor was once described as 'a dummy stage whose only purpose is to avoid a structural hazard'. Is this a fair comment and why? How does data cache cycle time affect your answer?

4 LG 4 – Folding, Retiming & Recoding

LG4.1. The critical path in a clock domain consists of three AND2 gates each with delay 35 ps and starts and ends at flip-flops with the following parameters: set-up time 4 ps, hold time 2ps, clock to Q time 15 ps. What is the maximum clock frequency ?

LG4.2. Time and space can be exchanged by design refactoring. Compare the algorithm on page 20 of the notes to a naive repeated addition approach to multiplication. Which has more time and which uses more space (if either)?

LG4.3. Booth's radix-4 algorithm for long multiplication is embodied in the following ML function:

```
fun booth(x, y, c, carry) = // Inputs are x and y. Other args initially clear.
    if(x=0 andalso carry=0) then c else
let val x' = x div 4
    val y' = y * 4
    val n  = (x mod 4) + carry
    val (carry', c') = case (n) of
      (0) => (0, c)
     |(1) => (0, c+y)
     |(2) => (0, c+2*y)
     |(3) => (1, c-y)
     |(4) => (1, c)
    in booth(x', y', c', carry')
    end
```

How many adders does it require in its natural hardware implementation (no special fold/unfold applied)? Sketch the hardware datapath and list the inputs and outputs to the controlling FSM. How many clock cycles does it use ? *Perhaps this should be on the long exercise sheet.*

5 LG 5 – Protocol and Interface

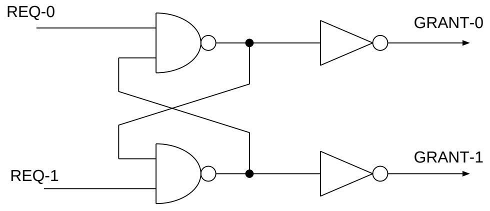
LG5.1. What is the difference between a synchronous and an asynchronous interface?

LG5.2. What is the difference in throughput between the two-phase and the four-phase handshake ?

LG5.3. Why might an interface be 'always ready' ?

LG5.4. How can data from two receiving interfaces be transmitted over a single outgoing interface ? Sketch RTL or a circuit that implements this for a suitable handshaking protocol. Perhaps add an extra feature (such as an extra wire in a parallel bus) where the data is somehow tagged with the originating interface as it conveyed to the output. (NB: This is not a short/quick exercise for those not used to hardware design.)

Hint: An arbiter, will be neeed. For a synchronous design it will be like http://www.cl.cam.ac.uk/teaching/1011/P35/arbiter/zhp711199d4d.html and for asynchronous design it might be based on a violated RS latch as shown: It might be helpful to also think about a companion design that implements the reverse direction (a demultiplexor).



6 LG 6 – SystemC Components

LG6.1. Why does SystemC implement the (non-blocking) signal paradigm where the value written is not immediately readable?

LG6.2. Why does SystemC distinguish between threads and methods and what is the difference ?

LG6.3. Sketch SystemC code for design of question LG5.4.. (If you answered that question using RTL perhaps just list the editor commands needed to make the change!)

7 LG 7 – Basic SoC Components

LG7.1. What is meant by the address space and the address map of a computer ?

LG7.2. Explain why I/O devices and other items in the memory map might appear more than once ?

LG7.3. What problem is encountered with memory-mapped I/O devices when the cache structure of a computer is misconfigured?

LG7.4. Why is it common practice for the interrupt signal to be deasserted by an I/O device when the status register is read?

LG7.5. Why is it necessary to be able to disable transmit interrupts ?

LG7.6. Why must the memory map of a computer be designed taking into account the reset and interrupt vectors hardwired into a processor core ?

LG7.7. A single LED is connected to a single GPIO pin of microcontroller. Sketch C or assembler code to make the LED flash at 1 Hertz.

LG7.8. Why is DMA generally used to service an Ethernet I/O device whereas it is not for a UART connected to an RS232 serial line ?

LG7.9. Why might the processor on a laptop computer adjust its clock own frequency ?

LG7.10. In the big/little approach to processor design, a simple core is used when there is not much processing to do. How might the switch between big and little cores be achieved ? *Note: there are many possible answers. May not be lectured.*

LG7.11. Why do devices have multiple clock domains? What precaution is needed when crossing clock domains? How does this influence the protocol or framing for messages crossing between domains ?

8 LG 8 – Instruction Set Simulator (ISS)

LG8.1. Why might an ISS run faster than the real hardware ?

LG8.2. What difference does it make whether an ISS includes cache models or not ?

9 LG 9 – ESL: Electronic System Level Modelling.

LG9.1. Why not wait for the silicon to be made before developing application software ?

LG9.2. Describe an example where 'functional modelling' of an application to be implemented on a SoC does not reflect the final layout of data in memory (i.e. not memory-accurate modelling).

LG9.3. 'Cycle-accurate modelling is faster than event-level modelling'. If true, what sort of event is being considered ?

LG9.4. To what extent can embedded software and firmware be tested by compiling it with the workstation compiler (e.g. to x86 code) compared with compiling it for the target architecture (e.g. ARM code) ?

LG9.5. What advantage can be made of the fact that embedded firmware is often written in the same language as behavioural models of components (i.e. C++) ?

LG9.6. How can RTL or net-level circuitry for an IP block be imported into an ESL simulation and how might the result differ from using a high-level model of that block?

10 LG 10 – Transactional Level Modelling (TLM).

LG10.1. How can TLM modelling avoid the need to model handshake wires?

LG10.2. Does device driver software have to be changed when using an ESL model ?

LG10.3. Transactor code for four-phase handshake is included in the lecture notes. Sketch equivalent code for two-phase handshake, where data is transferred on both edges of the request signal. Say whether your implementation is synchronous or asynchronous.

LG10.4. What is the advantage of carrying the delay parameter in the transaction signature and what would be the more-simplistic alternative ?

11 LG 11 – ABD - Assertion-Based Design

LG11.1. Classify the following as safety or liveness assertions:

(a) 'Either the inner or outer airlock doors is always closed'

(b) 'The sun never rises in the north.'

(c) 'When the big hand has just passed twelve the little hand is always on a number.'

(d) 'However far you drive along the M25 there will always be more road ahead.'

(e) 'Whichever way you branch you will always come to dead end.'

LG11.2. Give a fragment of RTL that will always eventually do something that no practical simulation run would ever reveal.

LG11.3. Write an assertion for a formal tool to check concerning your answer to LG11.2..

LG11.4. '*Sequential logic equivalence implies combinational logic equivalence*'. Discuss.

LG11.5. A simplistic generator algorithm, for use in directed/constrained random stimulus generation, will run very slowly. Give an example and explain why.

LG11.6. Why are regular expressions a rather limited language for expressing assertions over a hardware subsystem?

12 LG 12 – Network On Chip and Bus Structures.

LG12.1. A 64 bit bus is clocked at 200 MHz. A target always takes 4 clock cycles to respond to acknowledge a write. What is the write througput for a transaction size of one word? How could write posting or larger transactions improve performance?

LG12.2. If there are two general-purpose ARM cores on a SoC, should each have its own bus and should there be two external DRAM banks?

LG12.3. What advantage does the BVCI bus have that enables it to tolerate pipeline delays in the bus structure.

LG12.4. In terms of throughput and number of elements, compare a full cross-bar made of 2x2 elements to interconnect N initiators with N targets with a single ring network. What assumptions have you made about traffic patterns and are they realistic?

LG12.5. Why is DRAM so much slower than SRAM in terms of latency and throughput ? Why then do we use it?

13 LG 13 – SoC Engineering and Associated Tools.

LG13.1. Design a small circuit containing just flip-flops and 2-input multiplexors where a static timing analyser would produce the wrong result since the longest path never carries an event from end to end. (This pattern often arises when an IP block has a test mode controlled by a test input, so perhaps consider an external input called test that feeds your circuit.)

LG13.2. Design a scan path flip-flop (i.e. list its terminals and sketch its internal logic). This is a normal flip-flop that can become part of a shift register during testing. *May not have been lectured.*

LG13.3. Give a list of test vectors that will reveal all stuck-at faults in a direct hardware translation of the following RTL fragment: *May not have been lectured.*

```
assign p = a && b;
assign q = p || r;
```

LG13.4. A BIST unit for a 32Kx4 static RAM operates by writing a pseudo-random pattern to the device and then checks whether 1024 other pseudo-random writes corrupt any location. If there are five such RAMs on a SoC, each with their own BIST unit, and the wafer probe tester supplies a clock of 32 MHz (lower than normal operation), give a lower bound on the time to wafer probe each chip.

14 LG 14 – Architectural Design Exploration.

LG14.1. What is the difference between a co-processor and a peripheral ?

LG14.2. What is the main difference between a standard cell design and a gate array? Which can be mask programmed and which can be field programmed ?

LG14.3. What are the main two benefits of making transistors smaller in VLSI and what disadvantage is now becoming predominant below 45nm feature size ?

LG14.4. A 'respin' occurs when a chip has a design fault that means new masks are needed before it can go into production. Estimate (or Google for) the cost of a respin to the nearest million dollars!

LG14.5. The company V-Tech makes children's toys which normally consist of a battery, a loudspeaker, one or more motors, LEDs, switches, a number of pretty plastic moldings, the odd bit of artificial fur and a small circuit board. The company is very successful, with production runs of a given toy often exceeding 300,000 units. What technology should they use on their circuit board?

LG14.6. Give two major reasons why chips should increasingly be made with the intention that most/much of their logic is switched off or not in use much/most of the time.

15 LG 15 – Silicon Power and Technology

LG15.1. A logic gate with intrinsic delay of 190 picoseconds feeds four other logic gates using a total net length of 0.05 mm on metal layer 2 of a chip. The capacitance per unit length on layer 2 is 0.3 pF per mm. The logic gates being driven each have an input capacitance of 4×10^{-15} F. The gate derates as 10 picoseconds per load unit where a load unit is also 4×10^{-15}. What gate delay should be modelled?

LG15.2. The clock network within a subsystem has total length 3 mm. Its frequency is 200 MHz. The supply voltage is 1.2 Volts. The wiring capacitance is 0.3 pF/mm. How much power is dissipated by this clock net when in operation?

LG15.3. How can dynamic voltage scaling lead to a cubic relationship between clock frequency and power dissipation?

LG15.4. Why might the power consumption be different for different programs run on the same core even if they are all highly CPU-intensive ?

LG15.5. Why can dynamic clock gating respond more quickly than dynamic power gating ?

LG15.6. There are numerous potential mechanisms for frequency, power and clock gating control, including:

(a) a dedicated input pad (or pads) on the chip,

(b) additional logic inserted by the Verilog compiler as it generates a netlist from RTL,

(c) programmed I/O by a core writing to control registers.

Suggest a use case, such as an IP block, subsystem, application or device that might use each of these three techniques.