SoC D/M Exercises 10/11

These exercises are allocated marks at Tripos examination level, with 20 marks making a full exam question. Example answers are available to supervisors. There is some repetition of material between the exercises, so a suitable target is to solve approximately half of them.

1 LG1-5 (RTL, Simulation, Hazards, Folding) Exercises

RTL1. Give a brief definition of RTL and Synthesisable RTL. Name two example languages. [4 Marks]

RTL2. Explain Verilog's blocking and non-blocking assignment statements. Show how to exchange the contents of two registers using non-blocking assignment. Show the same using blocking assignment. [6 Marks]

RTL3. Convert the following behavioural RTL into an unordered list of (pure) RTL non-blocking assignments: [5 Marks]

```
always @(posedge clk) begin
   foo = bar + 22;
   if (foo > 17) foo = 17;
   foo_final = foo;
   foo = 0;
   end
```

RTL4. Convert the following behavioural RTL into an unordered list of (pure) RTL non-blocking assignments: [8 Marks]

```
always begin
   @(posedge clk) foo = 2;
   @(posedge clk) foo = 3;
   @(posedge clk) foo = 4;
   end
```

RTL5. Synthesisable RTL standards require that a variable is updated by at most one thread: give an example of a variable being updated in two `always` blocks and an equivalent combined always block or circuit. [8 Marks]

RTL6. Explain the terms 'structural hazard' and 'non-fully pipelined'. [4 Marks]

RTL7. Give a fragment of RTL that implements a counter that wraps after seven clock ticks. [3 Marks]

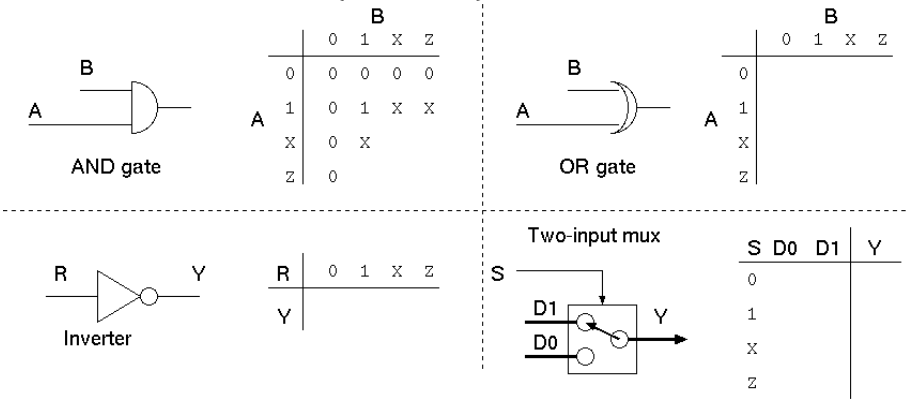
RTL8. Give a fragment of RTL that uses two multiply operators but where only one multiplier is needed in the generated hardware. Sketch the output circuit. [3 Marks]

RTL9. Show an example piece of synchronous RTL before and after inserting an additional pipeline stage. [4 Marks]

RTL10. Give an RTL design for a component that accepts a five-bit input, a clock and a reset and gives a single-bit output that holds when the running sum of the five bit input exceeds 511. [6 Marks]

RTL11. Give a schematic (circuit) diagram for the design of RTL10. Use adders and/or ALU blocks rather than giving full circuits for an such components. [7 Marks]

RTL12. Complete the truth tables for the 2-input AND gate (using symmetry) and the other three functions shown in this grid where the inputs range over { 0, 1, X, Z }.



RTL13. Identify the structural hazards in the following fragment of Verilog. What holding registers are simplistically needed if the main array (called daza) is a single-ported RAM? What if it is dual ported ? Can all the hazards be removed be recoding the algorithm ?

```
module FIBON(clk, reset);
   input clk, reset;
   reg [15:0] daza [32767:0];
   integer pos;
   reg [3:0] state;
   always @(posedge clk) begin
      if (reset) begin
         state <= 0;
         pos <= 2;
         end
      else case (state)
             0: begin
                daza[0] <= 1;
                daza[1] = daza[0];
                state <= 1;
             end
             1: begin
                daza[pos] <= daza[pos-1]+daza[pos-2];
                if (pos == 32767) state <= 2; else pos <= pos + 1;
             end
           endcase // case (state)
      end
endmodule
```

RTL14. Summarise the main differences between synthesisable RTL and general multi-threaded software in terms of programming style and paradigms. [10 Marks].

RTL15. In Verilog-like RTL, write out the complete design, including sequencer, for the datapath and controlling sequencer for the Booth long multiplier following the style of the long multiplier given in the lecture notes. (later we will do it in SystemC RTL-style and SystemC TLM-style).

```
// Call this function with c=0 and carry=0 to multiply x by y.
fun booth(x, y, c, carry) =
    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
```

It should start as follows:

```
module LONGMULT8b8(clk, reset, C, Ready, A, B, Start);

   input clk, reset, Start;
   output Ready;
   input [7:0] A, B;
   output [15:0] C;
```

(Details of language syntax are unimportant) [15 Marks]

non-both multiplier.

RTL16. Optional exercise: (no further examinable ground covered): Repeat the previous exercise for for long division (using either the shift-left till greater then shift right method or using Goldschmidt's method (repeatedly multiply denominator by $1 - 2d$)). [20 Marks]

RTL17. Modify (fold in space) the following RTL code so that it uses half-as-many ALUs and twice-as-many clock cycles to achieve the same functionality as the following component:

```
module TOFOLD(clk, reset, start, pp, qq, gg, yy, ready);
   input clk, reset, start;
   input [7:0] pp, qq, gg;
   output reg [7:0] yy;
   output reg ready;
   integer state;
   always @(posedge clk) begin
      if (reset) begin
         state <= 0;
         ready <= 0;
         end
      else case (state)
             0: if (start) state <= 1;
             1: begin
                yy <= (pp*gg +  qq*(255-gg)) / 256;
                ready <= 1;
                state <= 2;
             end

             2: if (!start) begin
                ready <= 0;
                state <= 0;
                end
           endcase // case (state)
      end
endmodule
```

[10 Marks]

2 LG6: SystemC Components Exercises

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 and what sorts of '*channels*' are supported between modules ? [8 Marks]

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. Give the SystemC synthesisable equivalent design for the design of RTL10. *You should sketch C++ code that looks roughly like RTL rather than worrying about a precise definition of synthesisable.* [7 Marks]

SYSC10. 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]

SYSC11. Sketch SystemC RTL-like code for a synthetic data generator that creates three byte packets and delivers them over the four-phase interface of SYSC10.. *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.

`ssh linux.pwf.cam.ac.uk: /ux/clteach/SOCDAM/thisyears-toyclasses`

(C) 2008-11 DJ GREAVES. END OF DOCUMENT.