

# System-On-Chip Design And Modelling Exercises kg1-kg3 Feb 2018

Exercise Sheet TWO - Feb/Mar 2018.

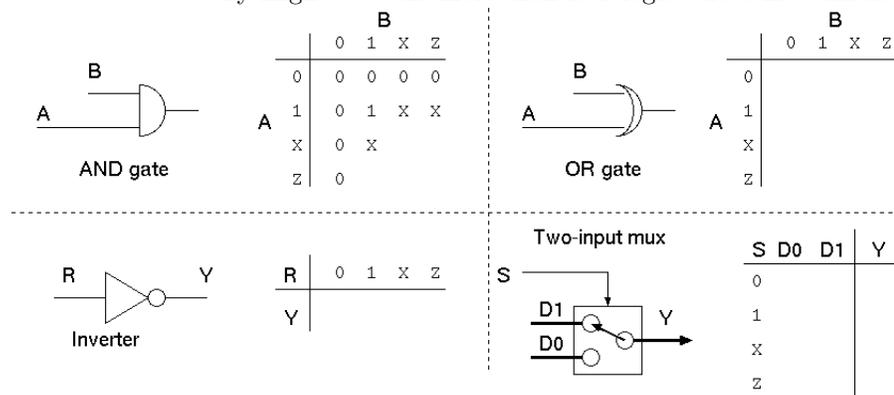
## 4 RTL, Simulation, Hazards, Folding - Full Questions

RTL1. 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 that is valid/correct. [8 Marks]

RTL2. Give a schematic (circuit) diagram for the design from the quick sheet <sup>1</sup> that checks whether running sum of the five-bit input exceeds 511. Use adders and/or ALU blocks rather than giving full circuits for any such components. [7 Marks]

RTL3. 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 }. (X denotes don't know and Z denotes high-impedance.)

*For supervision discussion:* Why might we want more than four logic values in a simulation?



RTL4. ♡ 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 by 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
end
endmodule

```

RTL5. Summarise the main differences between synthesisable RTL and general multi-threaded software in terms of programming style and paradigms. [10 Marks].

<sup>1</sup>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.

RTL6. 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.

```
// 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]

RTL7. 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]

RTL8. Bus Bridge.

- What is the function of a bus bridge in a SoC ? [2 Marks]
- What typical and possible address translation semantics might a bus bridge implement ? [4 Marks]
- How might internal queue structure vary between bus bridge designs ? [3 Marks]
- How might arbitration policy vary between bus bridge designs ? [3 Marks]

RTL9. a) What does it mean to say that an on-chip bus protocols is amenable to pipelining?

- Why is it desirable to use on-chip protocols that are amenable to pipelining?
- Why is flow control necessary in on-chip busses? Is flow control needed for access to simple registers?

RTL10. a) Give a circuit that would 'fool' a static timing analyser. This is one where truth-tables show that a long logic path cannot ever propagate an event, but the timing analyser ignores truth tables and sees the path dominating.

## KG5: HLS

- HLS1. *a)* Why does HLS claim to deliver faster computation than Von Neumann (for some applications at least)?  
*b)* Why does HLS claim to deliver lower energy computation than Von Neumann in general?  
*c)* Which of the following applications is likely to be improved by HLS to FPGA: word processing, code cracking, MPEG compression, high-frequency trading, facial recognition, database lookup?  
*d)* Compare automatic inference of parallelism with explicit expression using parallel programming constructs.  
*e)* Can static schedules be used with tasks whose execution time is unpredictable?

HLS2. 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 nsofar 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?

HLS3. 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.

HLS4. (*This question mainly for discussion in supervisions.*)

- a) What factors should influence the choice of layout of data in RAM memories?
- b) Why is task-level parallelism always easy to support and when do other forms of parallelism get tricky?
- c) What optimisations become possible when loop bounds and array subscripts are predictable at compile time?
- d) The FFT uses a shuffle data path and so seems to be ideally suited to a fixed wiring pattern between butterfly units on an FPGA. Discuss. Why is it always said that the FFT is hard to layout?
- e) Assuming mapped to FPGA, give the principle defining features of the data paths in Systolic Arrays, Kahn Process Networks and Classical HLS. Should an HLS tool be free to make its own choice about what style of execution to deploy?

HLS5. (*Kahn Networks and Systolic Arrays not examinable in 2017/18.*)

- a) It is argued that the systolic array approach is superior to a CSP/Kahn-like network when there will be no deviations from a static schedule. Consider a matrix multiplication or CNN application: is the FIFO between nodes needed for CNN accelerators?
- b) Can a CSP/Kahn-like network where the read operator removes from two or more queues at once (chordal read) always be emulated by one that can only do simple queue reads?
- c) Can a CSP/Kahn-like network where the read operator peeks in a receive queue be emulated by one that does not have this capability?

## **KG6: ESL**

- ESL1. a) missing ... in this draft ...
- b)
  - c)