Test 2: Lectures 9 to 19

EEE4084F 2018-05-17

Instructions:

- Answer on separate pages.
- Make sure that your student number is on all your answer pages.
- There are **3** questions, each divided into sub-questions. Answer all questions.
- Total time: 45 minutes.
- Total marks: 54.

Question 1: ADCs **[22 Total]**

These question relates to Seminar 6 on ADCs.

(a) Explain what an offset error is for an ADC. Provide an example (or diagram) to help your explanation. [6]

(b) Consider that you have an 8-bit ADC which happens to use SPI with an interface of 4 signals. Answer this: (i) what is the resolution of this ADC [1] and (ii) how many output codes can it provide? [1]

(c) An ENOB is a useful measure for an ADC. (i) Explain ENOB and what the acronym stands for [1]. (ii) Does the ENOB for a given ADC typically decreases or increases as the sampling rate increases? [1] (iii) Explain you answer to (ii). [2]

(d) A Flash ADC (i.e. a parallel or flash architecture ADC) achieves a very high conversion rate. (i) But at what cost? Briefly mention what the cost of a Flash ADC is compared to the more classic and cheaper counter method (i.e. which is closer to how the conceptual model shown in the textbook works). [4 marks] (ii) Sketch a block diagram to show how a Flash ADC is structured and works [6 marks].

Question 2: This concerns lectures 9-12 **[22 Total]**

Answer all following questions…

(a) I know that I said you could skip *most of* OpenCL for this test. But this question is more asking how this sort of tool could be useful to developers, especially considering you recent experience with it. *So answer this:* why do you think a tool (or programming framework) like OpenCL is useful if you are planning to develop applications for heterogeneous systems? Do you think it solves all the programming needs – if not then how does it help programmers develop applications for parallel systems? [5 marks]

(b) Quickly define 'observed speedup' in terms of wall clock time!! [2 marks]

(c) Sometimes parallelism just doesn't work out too well, and your effort might be called a 'stupidly parallel' system. Then there is the official term of 'embarrassingly parallel' – surely that is even worse than 'stupidly parallel'!? Explain why it could be rather nice to have an embarrassingly parallel solution than a stupidly parallel one. [5 marks]

(d) So Von Neumann was a famous guy who wore a suit all the time, even when walking his dogs on the beach. But he was a cleaver guy too, after which the Von Neumann machine was name after. The Harvard architecture has a different design to the Von Neumann. If you were developing a simple embedded application that didn't need to run particularly fancy code, provide a motivation for which architecture (Von Neumann or Harvard) that you would choose and why. Provide a sketch illustrating the architecture you chose and to assist in your explanation[1]. [10]

Question 3: Few other snippets of thoughts **[10 Total]**

(a) The term "Kaizen" has become quite a trendy term in the field of software quality assurance. What does it mean? [2] And what country does it come from? [1]

(b) And here is a tiny bit of Verilog programming that I promised…

So you have a clock *clk* that is running at 10MHz and you need to flash a LED, rather fast, at a 100ms period (i.e. the LED signal line toggles every 100ms). Complete the module below to make this happen. You can add additional registers for counters etc. to slow the clock or to generate blockchain or do other processing you think might need to get the LED to flash at the desired speed[2]. You just need to indicate the code needed to be added after the "put your code here" comment. [7]

```
//-----------------------------------------------------
// Design Name : flashingleds
// File Name   : flashingleds.v
//-----------------------------------------------------
module flashingleds    (
   clk  ,  // clock running at 10MHz
   reset,  // reset the system
   led     // the LED to flash
);
  //------------Input Ports--------------
  input clk, reset;
  //----------Output Ports--------------
  output led;

  //----------put your code here-----------

endmodule
```

Bonus: who made the Stratix FPGA and was it for high or low performance? You can select *one* of the options below (A1 or A2 or L1 … etc):

A1 Altera / Low Performance	A2 Altera / High Performance
L1 Lattice / Low Performance	L2 Lattice / High Performance
T1 Tabula / Low Performance	T2 Tabula / High Performance
X1 Xilinx / Low Performance	X2 Xilinx / High Performance

END OF TEST

[1] The main purpose here is your motivation and sketch, it doesn't so much matter which approach you recommend, although the one will be more difficult to motivate for than the other.

[2] You might notice there is a reset line, you may need to make use of this facility so that the first flash occurs 100ms after the reset line is lowered.

CHEET SHEET

Appendix B: Verilog Cheat sheet

Numbers and constants

Example: 4-bit constant 10 in binary, hex and in decimal: 4’b1010 == 4’ha -- 4’d10
(numbers are unsigned by default)
Concatenation of bits using {}
4’b1011 == {2’b10 , 2’b11}
Constants are declared using parameter:
parameter myparam = 51

Operators

Arithmetic: and (+), subtract (-), multiply (*), divide (/) and modulus (%) all provided.
Shift: left (<<), shift right (>>)
Relational ops: equal (==), not-equal (!=), less-than (<), less-than or equal (<=), greater-than (>), greater-than or equal (>=).
Bitwise ops: and (&), or (|), xor (ˆ), not (˜)
Logical operators: and (&&) or (||) not (!) note that these work as in C, e.g. (2 && 1) == 1
Bit reduction operators: [n] n=bit to extract
Conditional operator: ? to multiplex result
Example: (a==1)? funcif1 : funcif0
The above is equivalent to:
```
 ((a==1) && funcif1)
 || ((a!=1) && funcif0)
```

Registers and wires

Declaring a 4 bit wire with index starting at 0:
wire [3:0] w;
Declaring an 8 bit register:
reg [7:0] r;
Declaring a 32 element memory 8 bits wide:
reg [7:0] mem [0:31]
Bit extract example:
r[5:2] returns 4 bits between pos 2 to 5 inclusive

Assignment

Assignment to wires uses the assign primitive outside an always block, e.g.:
assign mywire = a & b
Registers are assigned to inside an always block which specifies where the clock comes from, e.g.:
```
always@(posedge myclock)
   cnt = cnt + 1;
```

Blocking vs. unblocking assignment <= vs. =

The <= assignment operator is non-blocking (i.e. if use in an always@(posedge) it will be performed on every positive edge. If you have many non-blocking assignments they will all updated in parallel. The <= operator must be used inside an always block – you can’t use it in an assign statement.
The blocking assignment operator = can be used in either an assign block or an always block. But it causes assignments to be performed in sequential order. This tends to result in slower circuits, so avoid using it (especially for synthesized circuits) unless you have to.

Case and if statements

Case and if statements are used inside an always block to conditionally update state. e.g.:
```
always @(posedge clock)
  if (add1 && add2) r <= r+3;
  else if (add2) r <= r+2;
  else if(add1) r <= r+1;
```
Note that we don’t need to specify what happens when add1 and add2 are both false since the default behavior is that r will not be updated.
Equivalent function using a case statement:
```
always @(posedge clock)
  case({add2,add1})
  2’b11  : r <= r+3;
  2’b10  : r <= r+2;
  2’b01  : r <= r+1;
  default: r <= r;
endcase
```

Module declarations

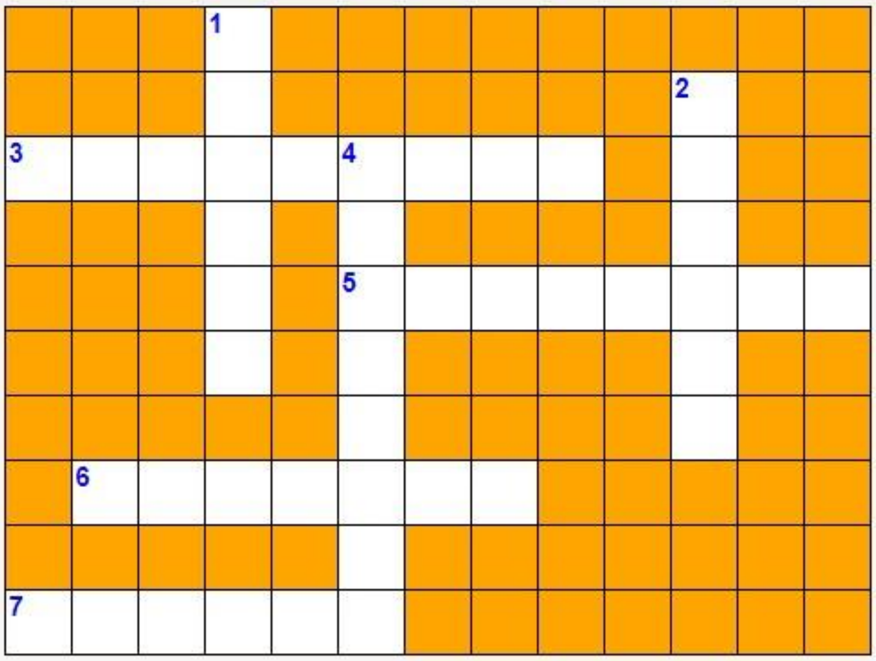
Modules pass inputs, outputs as wires by default.
```
module ModName (
  output reg [3:0] result,  // register output
  input [1:0] bitsin,  input clk, inout bidirectnl  );
 … code …
endmodule
```

Verilog Simulation / ISIM commands

```
$display ("a string to display");
$monitor ("like printf. Vals: %d %b", decv,bitv);
#100   // wait 100ns or simulation moments
$finish  // end simulation
```

For those who are bored or didn't study, here's some lite entertainment… it's seriously easy you should get it in about 2 minutes ☺

For more fun crosswords (where I got this one) and fun ways to improve your English see: http://www.englishmaven.org/Pages/Crossword%20Puzzles.htm



Across:

3. "Hump" day. This day is in the middle of the week.

5. Abbreviated as "Thurs"

6. People say this is the most productive day of the week. Hint: it isn't Monday!

7. The last day of the week. A day to rest.

Down:

1. This is the first day of the week

2. This days starts the weekend! Thank gosh it's ________!

4. Starts with the letter "S". It's not Sunday!