Solutions for EEE4084F Exam June 2013

Section 1: Short answers

Q1.1.

(a) Hardware solution / Highly specialized hardware: such as using ASIC solutions. Gives maximum speed but long fabrication times and high expense, especially if ASICs need to be refabricated to fix mistakes.
Software solution: more flexible, possibly quicker to develop (esp. if the application is a complex one), provides options of remedying bugs by changing the software instead of refabricating the hardware. Advantage of software upgrades being able to keep the physical product operational and useful for longer. Drawback is in parallelizability – if only one CPU this may significantly limit the speed and scope of developing parallel solutions using the platform.
Reconfigurable solution: offers advantages both of speed from custom hardware, flexibility of reconfigurability (and possibility of further prolonging the lifetime of the system), and use of software for part of the solution being software based and also provision of upgrades and bugfixing without refabing. But disadvantage is the lengthy learning curve and variety of tools needed. [6]

(b) Answer: a reconfigurable computing platform is a computer system that can change part of its hardware according to the computation it is to performed. Microprocessor reconfiguration happens more at the macro scale, connecting processing nodes, whereas FPGA-based reconfigurable is more at the micro scale of gates and constructing combinational logic circuits. [4]

Q1.2.

This question is largely marked based on the student's description and meaningful argument. Suitable example(s) of what we many expect of cloud computing in the years to come will count for a few marks. I would expect disadvantages to include issues such as reliance on network connectivity, issues related to risk and security (such as trusting someone else to keep your work secure). Other disadvantages may be limits on the software available, such as not being able to install your own applications as is so easy to do when you have your own computer running something like Windows. Advantages of cloud computing are listed in the slides, including things that were highlighted in the video clip about cloud computing such as needing a smaller in-house IT maintenance team and less need for frequent updates and upgrades of individuals' office computers – which could save both costs and the environment.

Q1.3.

(a) 1950

(b)

Clutter mitigation (M)	Back-end
Pulse compression (P)	Front-end
Sampling ADCs (S)	Front-end
Convolvers or FIR filtering (C)	Front-end
Database processing (D)	Back-end

(c)

A measure for complexity is measuring the lines of code that is developed for a HPEC system; the usual unit is SLOC (source lines of code). Other measures of complexity are the number of software modules. The size of a FPGA bit image. The number of interconnects between hardware modules, or between software modules. [4]
Measures of performance include Dhrystone, which describes how fast a processor can loops though a specific set of signal processing operations; the unit of measure is the DMIPS (Dhrystone MIPS). Whetstone is another performance measure like Dhrystone. This question has been left fairly open because the student should be able to think of many ways to measure performance and complexity as these issues were discussed in detail in the lectures and seminars.

Q1.4.

(a) Answer: Temporal computation is sequential, doing one thing at a time. Spatial computation is expressed in as a space, dependencies and linkages between parts of the computation, which is not related to a time-based sequence of steps. Spatial computation algorithms are well suited for implementation as on parallel processor or as hardware because multiple tasks could overlap in time.

(b) Answer: (only one of the differences listed below is needed):

- A CPLD contains considerably less programmable logic elements (Les) in comparison to an FPGA. CPLDs contain a few tens of 1000s LEs; whereas FGPAs typically comprise millions of LEs.
- CPLDs are cheaper than FPGAs, and in terms of configuration arch there is not as much need for so many LEs as per used in application-specific hardware the FPGAs are programmed with (i.e. no need to use an expensive FPGA for config arch where a cheaper CPLD would do).

CPLDs are generally used as configuration glue logic instead of FPGAs because the configuration architecture is generally much simpler, taking considerably less LEs than does the application-specific hardware implemented in the FPGA(s) (which the configuration architecture programs). Accordingly, to save cost and complexity of the RC platform, CPLDs are used for the configuration architecture.

(c) Answer: ABI stands for Application Binary Interface, and is a means by which two applications (or software programs) connect with one another – usually ABI refers to a connection between an application program and an operating system running on a computer platform. ABI are generally formulated according to a specification, rather than purely hardware constraints, for specified methods of passing information from one program to another. Thus a platform may support operating system A which has an ABI that explains that parameters are passed via the stack; and the same platform may be supported by operating system B that passes parameters via a particular CPU register.

Section 2: Multiple choice answers

Q2.1 (d)

Q2.2 (b)

Q2.3 (c)

Q2.4 (b)

Q2.5 (c)

Q2.6

(i)	False
(ii)	True
(iii)	False
(iv)	False

Section 3: Long answers

3.1 Solution for up_down_counter [14 marks]

```
module up_down_counter (
       count     ,  // 8-bit output for the counter module
       up_down   ,  // up_down - 1 for up, 0 for down
       clk       ,  // clock input
       reset        // reset input (keeps count set to 0)
);
//----------Output Ports--------------
reg output [7:0] count;
//------------Input Ports--------------
input up_down, clk, reset;

//-------------Code Starts Here-------
always @(posedge clk)
if (reset) begin // active high reset
  count <= 8'b0 ;
end else if (up_down) begin
  count <= count + 1;
end else begin
  count <= count - 1;
end

endmodule
```

Comments for interface

Other comments for interface

Comments for reset

(i.e. 3 marks for comments)

3.2 Solutions [21 marks]

(a)

```
    // MiniMicro code to calculate and output X:
    IN 0xF0
    SWP B
    IN 0xF1
    OR B
    SWP G   // G = inB OR inC
```

// added code as required for the answer:

```
    IN 0xF2
    SWP D
    IN 0xF3
    AND D
    CPY A,H // H = inD AND inE
    AND G   // A = H AND G
    OUT 0xF4 // output result to X
    // the values of B,C,D,E are already in registers
```

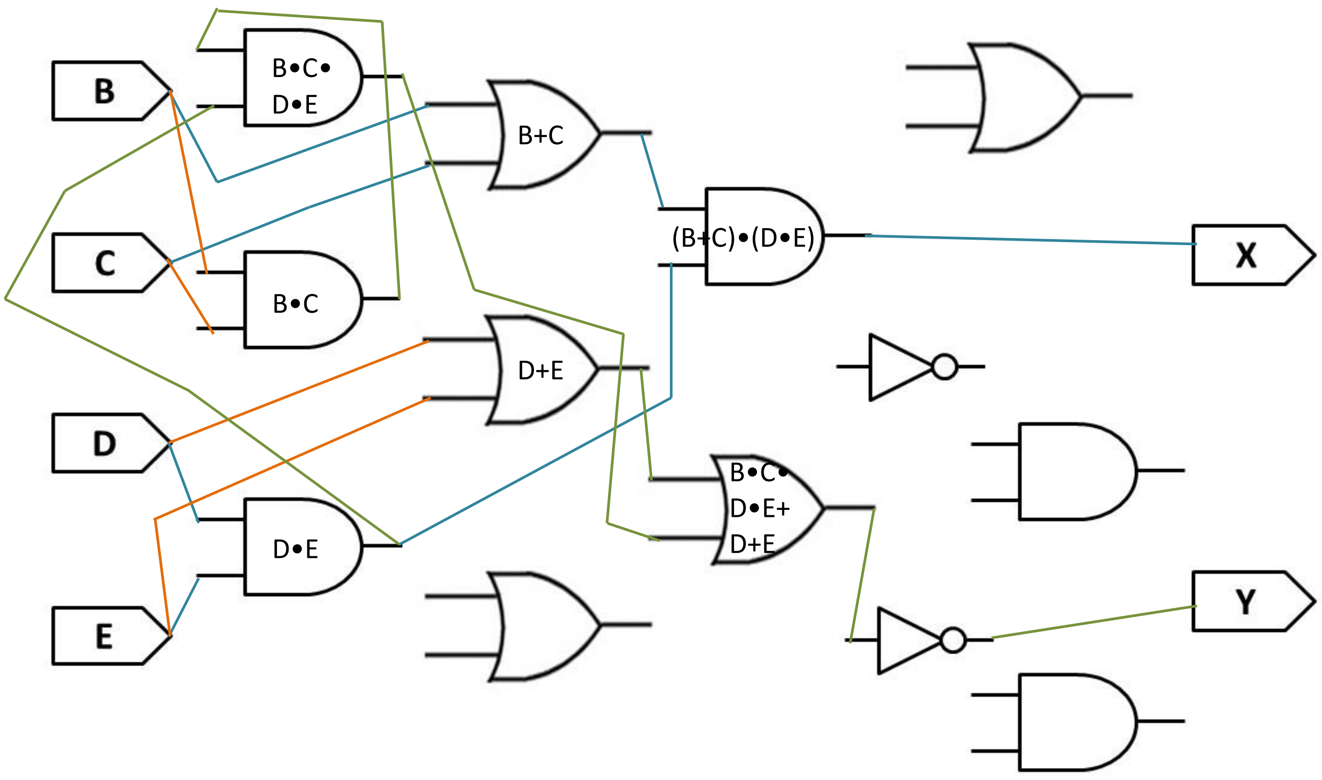
```
// Calculating Y:
CPY B,A
AND C
AND H
CPY A,H  // H = A and C and D and E
CPY D,A
OR  E
OR  H    // H = (A and C and D and E) OR (D OR E)
NOT
OUT 0xF5  // output result to Y
```

[7 marks]

(b)

There are 21 instructions above. Each instruction takes 1/100,000,000 s to run = 10ns. A total of 21 instructions = 210ns.

(c) Routing the non-optimized expression :



Boolean expressions to be implemented in this CLB:

X = AND(OR(B,C), AND(D,E))   Y = NOR(AND(B,C,D,E), OR(D,E))

Clearly the expressions for Y can be optimized

Y = NOR(AND(B,C,D,E), OR(D,E))

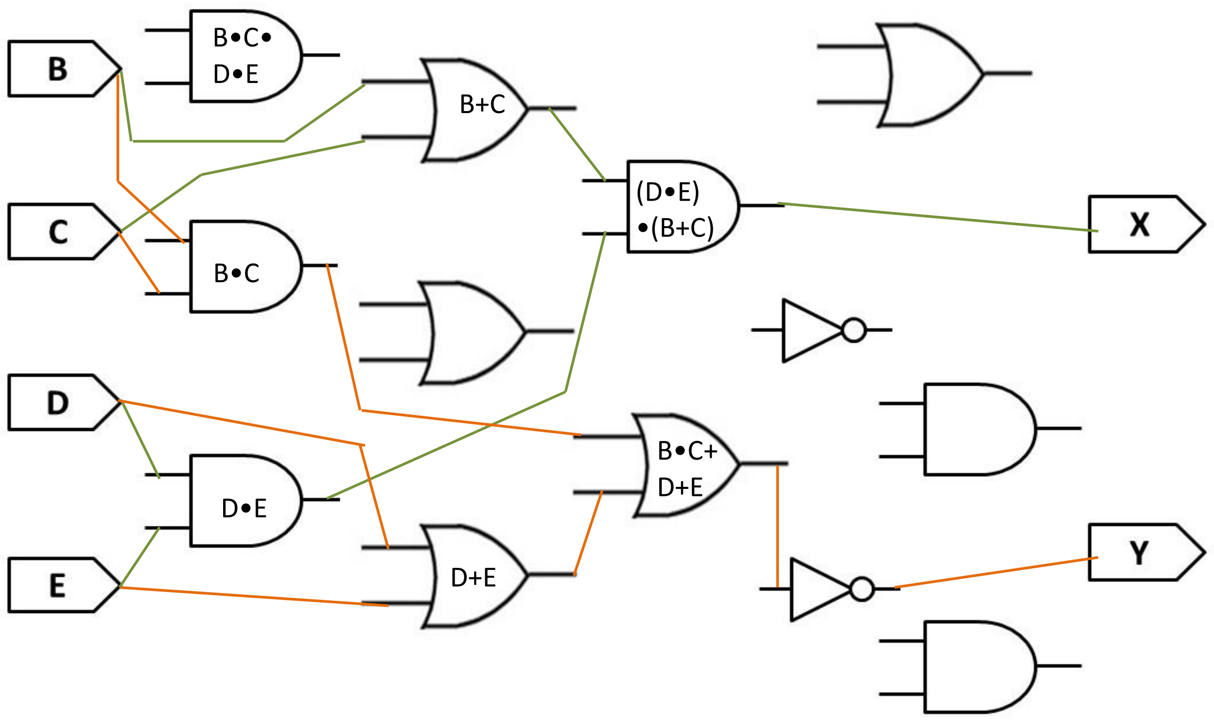
Y = NOT(BCDE+D+E)

Y = NOT (BC+D+E)

Whereas X can stay as is:

X = DE(B+C)

The optimized diagram is shown below:



Boolean expressions to be implemented in this CLB:

X = AND(OR(B,C), AND(D,E))

Y = NOR(AND(B,C,D,E), OR(D,E))

$\boxed{\text{Y = NOT (BC+D+E)}}$

(d) Propagation delay:

Non-optimized:

Longest path is NOT -> OR -> AND -> AND

Gates	# Gates in path	Delay/Gate	Delay
NOT	1	15ns	15ns
OR	1	20ns	20ns
AND	2	22ns	44
			79ns

The propagation delay for the non-optimized version is 79ns

For the optimized version, there is one less AND in the path, so it is just 57ns

Speedup of hardware over software:

210ns / 79ns = 2.66 for non-optimal solution, or

3.64 for optimal solution.

End of Solutions