



Digital Systems

EEE4084F



FINAL EXAM

29 May 2012

3 hours

*Examination Prepared by:
Simon Winberg*

Last Modified: 06-May-2014

SOLUTIONS!!

Answers are in **green text**

Section 1: Short Answers [42 marks]

Question 1.1 [11 marks]

This question relates to FPGA-based Reconfigurable Computing (RC) platforms.

- (a) Describe the differences between the temporal and spatial computing paradigms. Motivate which of these paradigms is probably better suited to developing applications that run on RC platforms. [3 marks]

The temporal paradigm represents program solutions in terms of time and sequences of steps. The spatial paradigm represents a solution in terms of spaces and interrelations. The spatial paradigm is better suited to developing solutions for RC platforms as these solutions typically involve both different spaces, as in running on a processor or multiple parts running independently in hardware, as well as work done in parallel with some relation between them (e.g. sharing partial results).

- (b) RC platforms that incorporate FPGAs often include a CPLD as well. The CPLD in such a RC platform is usually used as configuration and glue logic whereas the FPGA(s) are used for computation.

- i. Describe the main differences between a FPGA and a CPLD [2 marks].

The main differences between these are: the number of logic elements (LEs) and the volatility vs. non volatility of the program stored. FPGAs have considerably more LEs but the program is volatile (needs to be programmed each time on start-up); whereas CPLDs have much less LEs but the program is non-volatile.

- ii. Motivate why a CPLD is often chosen to implement this glue logic instead of using a microprocessor processor or simply by clocking the FPGA programming lines directly from the PC. [2 marks]

As the previous question alluded to: the non-volatile nature of the CPLD, as well as its low price, makes it convenient solution for using in the configuration logic for setting up the FPGA. Using a PC (e.g. via JTAG) makes the system less stand-alone; the PC has to be connected during start-up to program the FPGA to make it do something useful – often one wants the FPGA to be programmed immediately on system start-up. Having to put a microprocessor/microcontroller on the FPGA platform make likely make for unnecessary expense and be slower than a CPLD (admittedly a ucontroller could turn out a better solution as it can be programmed in C and likely easier to reprogram).

- (c) The overall design of the Programmable Active Memories (PAM) reconfigurable computing platform is illustrated on the right. As you can see each FPGA has its own SRAM bank, each with dedicated address and data lines. FPGAs connect to one another via a shared bus (not shown). Explain one advantage and one disadvantage of this design in terms of high speed data processing [2 marks]. Briefly argue a type of applications suited to this architecture, and a type not suited to this design. [2 marks].

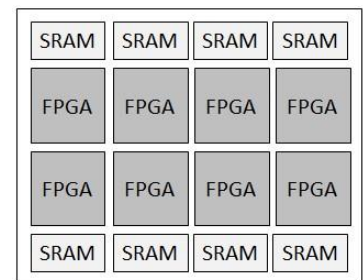


Figure 1: PAM design overview

A clear advantage of this structure is each FPGA having its own SRAM in order to use for 'scratch memory' or perform intermediate calculations without having to worry about sharing address and data lines with other FPGAs that may be attempting to use memory. A disadvantage is in transferring data or results between FPGAs – for example for the first FPGA to get a result generated by the second FPGA, some form of inter-FPGA communication would need to be structured, and it might turn out that such a process of one FPGA 'asking' another FPGA for data might be less efficient than having one big block of shared memory that all the FPGAs content with one another to use.

The effectiveness of this design depends largely on the application. This design would be better for coarse-grained problems, especially embarrassingly parallel algorithms, where the communication between tasks is minimal (i.e. the ratio computation : communication is low, like 100:1). For fine-grained problems this design may be less effective since each task would need to do much communication with others, requiring some form of performance penalties in terms of synchronization and communication between FPGAs.

Question 1.2 [11 marks]

This question relates to general concepts and facts about parallel and high performance computing.

- (a) Name two applications that commonly run on large scale parallel computing systems. [2 marks]

These were discussed in Lecture 2 slide 18, i.e.: Databases, web servers, internet commerce / OLTP (online transaction processing, technical computing, threat analysis, credit card fraud.

- (b) Explain why much of the parallel code in a parallel computer system is often hand crafted? (i.e. much care and effort going into it, writing in a fairly low-level language like C.) [2 marks]

Parallel code is often hand-crafted for two reasons: typically it is just the critical section of a program that is made parallel (to save development costs and because significant performance boosts are unlikely from parallelizing infrequently used parts of code). In addition, there tends to be much complication (especially for fine-grained problems) in terms of getting the synchronization and partial results to be done correctly.

Frequently, automated solvers are more used in a dual programming approach, used as a safety check to verify that the handcrafted parallelized code is producing the same results as the original non-parallel code.

- (c) Consider that in a business meeting you recommend using a parallel language to implement a parallel computing solution, and that automatic parallelization might be worth considering. But a colleague responds: "No way! There aren't parallel computing languages and automatic parallelization is useless". Present an argument to stand by your statement and illustrate examples confirming that parallel languages do indeed exist – also (to be a smart ass) clarify the current situation of automatic parallelization techniques. [5 marks]

Parallel programs do exist: these are essentially programming languages that can be used to implement parallel systems. The definition is quite vague: standard C could be classified as a parallel language (e.g. C using OpenMP). In terms of a spatial language, or visual models, to represent parallel operations, there are many, for example Simulink effectively describes parallel operations without the user having to explain how to go from the parallel model into a parallel implementation. In terms of automatic parallelism, there are systems that work, like Simulink and SystemC compilers but the problem is that the generated code is often difficult to read and to debug.

Additional mark for logical sentence / argument

- (d) Nowadays, server class machine tend to use SMP architectures. Briefly explain: what is a SMP and what does the acronym SMP stand for. [2 marks]

The acronym SMP stands for Symmetric Multi-processor. It is a processor chip that combines more than one (usually a multiple of 2) processor cores for which each core follows the same design (for example, a dual core Pentium chip that is made up of two Pentium processor cores).

Question 1.3 [10 marks]

This question relates to interconnection fabrics and bisection bandwidth.

Consider you want to connect up eight processing nodes. These processing nodes are independent computers all to be networked using crossbar switches as needed. Assume all links have the same maximum speed (1Gbps).

- (a) Contrast a drawback and a benefit between using a simple binary tree network topology compared to a binary fat tree network to connect these eight processor nodes. Determine the bisection bandwidth for each of the two cases. [4 marks]

In this question, I'm hoping the student notices that I mentioned eight processor nodes, to allude to a tree structure that has three levels: processors at the bottom, two crossbars at level 1, one crossbar at level 2.

A benefit of using a binary tree is that the cost of the switches are less, being able to get by with switches that have fewer ports and are thus cheaper. A drawback is that the root level becomes congested as it is a one-link junction between the two halves of the network.

The bisection bandwidth of this case is 1 for the binary tree (it is always 1 for binary trees); whereas for bisection bandwidth of the fat tree scales directly linearly with the number of nodes, so it would be $N/2 = 4$.

- (b) If the processing nodes were instead to be implemented using a VLSI approach, instead of individual computers, motivate why a fat tree approach is probably the better option, especially if there are only 8 processors in the design. [2 marks]

For VLSI (and to a certain extent deployment on a FPGA) the cost aspect of a FAT tree approach may be considerably less – in this case the cost of crossbars are not in the form of network switch but instead interconnections on the silicon chip, which is effectively much cheaper simply being connections. For only 8 nodes it would likely be little difference in cost between doing a routing that makes each node connect to many other nodes, compared to the case where one node only connects to a neighbor and a switch.

- (c) Consider that this 8-processor system works cooperatively on a 128x128 matrix of 32-bit floating point values. Each processor node starts with only part of this matrix: specifically each processor has a sub-matrix of size 128 columns x 16 rows of floats. Processor 1 has the top 16 rows, processor 2 has the next 16 rows, etc. If the sum of all elements of the matrix (i.e. the 32-bit float value $\sum A_{i,j}$ for $i=1:128, j=1:128$) was to be calculated, determine how many bytes would have to be transferred around the network, assuming the result of the final sum is to end up at node 1. Ensure you explain your answer logically. [4 marks]

If you think sensibly it should be clear that this is an embarrassingly parallel problem. Each process node will do a sum of all its matrix elements, producing a single float. Nodes 2 to 8 send a single float to node 1, this is effectively 7 floats sent through the network, so $7 \times 4 \text{ bytes} = 28 \text{ bytes}$.

(Note to marker: student might assume setting up of each node also needs to be done, in which case it would be $4 \times 128 \times 128$ bytes to set up the nodes and another 28 bytes as explained above.)

Question 1.4 [10 marks]

- (a) Define what is understood to be a *reconfigurable computer*, highlighting how to discriminate between a reconfigurable computer and a non-reconfigurable / regular networked computer. [2 marks]

A reconfigurable computer is a computer system that can change its hardware datapaths and control flows by software control. A regular network computer has its network topology hard-wired; it has no control over how the datapaths in the system can be changed (this is generally true even for networks with smart switches: a particular computer can't decide how the switch will route packets).

- (b) The Xilinx Virtex-6 comprises an architecture composed of two types of configuration logic blocks (CLBs), namely SLICEM and SLICEL blocks. Explain what a CLB is – surely it is just the same as a LUT (or is it)? Discuss the differences between SLICEL and SLICEM blocks and how different slides can be beneficial in terms of FPGA manufacture and programming. [4 marks]

The acronym CLB stands for Configurable Logic Block. It is not a LUT: It is a collection of logic elements (LEs), which can be programmed to tell how its LEs are to be interconnected – the LEs within a CLB might well be LUTs. SLICEL is Xilinx's standard slice / CLB (a Xilinx FPGA is made up of multiple 'slides') – the 'L' standard for logic: so it's a block of standard logic (usually LUTs and multiplexers) that has been chosen both as the most basic and most flexible type of slice that can be used to implement pretty much any logic circuit (that fits in the slice). The SLIDEM has 'more' features in it – often memory and DSP elements (e.g. floating point multipliers). Having a variety of different slides can make for better performance implementations, choosing for example a hard DSP element on a SLIDEM that works at much higher speed than the equivalent combinational circuit made from linking together a whole lot of discrete LEs in a SLIDEL. It can also make for faster trace & route and programming (i.e. the DSP block is ready-made, ISE don't need to do T&R for its internal implementation).

- (c) Consider the design below. The system comprises two VHDL entities, imported into a schematic editor (such as the Xilinx ISE block editor). But when the design is synthesized and executed on hardware, weird results occur (see the graphs below), even though the data is streamed into the system at a much slower rate than the speed at which the entries operate. Indeed, much of the output from the hardware looks correct (see right hand graph below) yet doesn't match the golden measure (left hand graph). The correlation coefficient between the datasets is 0.834, verifying that something is wrong. Comment on what may be causing this problem and how it might be solved. [4 marks]

It is a synchronization problem. VHDL entity 2 has no handshaking in place in order to use the correct average (avg output) value before outputting its DeltaN result – but this handshaking can't really be remedied without changing the code of the entities to add additional handshaking lines (this would probably be the recommended approach). But if you don't want to change code for the entities, the a likely solution is to put a delay on the WStrobe, so that Entity 2 'sees' the strobe a few nanoseconds after Entity 1 does. The

DataIn line would be assumed not to change soon after WStrobe is pulsed. (Arguably, if the data was streamed in live, there may be a mismatch in clock domains between the golden measure and the hardware implementation: fixing this would necessitate checking that the entities are clocked at the right speed for the incoming samples – but an assumption was made that the samples were stored and worked on in memory).

Section 2: Multiple Choice [20 marks]

NOTE: Choose only one option (i.e. either a, b, c, d or e) for each question in this section.


ANSWERS ARE INDICATED USING A  arrow

Q2.1 Consider that the following VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity E1 is
    port ( A: in STD_LOGIC; B: in STD_LOGIC; X: out STD_LOGIC );
end E1;
```

Consider that you want this entity to output the result of the C expression $X = A \ \&\& \ !B$ (assuming that all these variables are Boolean or bit values). Which code snippet below will implement this architecture for the entity without syntax errors:

(a)
architecture Elbehavior of E1 is
begin
 X = A and not(B);
end;

(b)  **ANSWER** The other options have syntax errors or not using the right keyword
architecture Elb of E1 is
begin
 X <= A and not(B);
end Elb;

(c)
module Elb of E1
{
 X = and(A,not(B));
};

(d)
process Elb of E1 is
begin
 X := A and not(B);
end;

(e)
architecture E1 :
begin
 X := A and not(B);
end E1;

[4 marks]

Q2.2 What is another term for asynchronous communications?

- (a) Non-blocking communications. ← ANSWER
- (b) Blocking communications.
- (c) Handshaking communications.
- (d) Serialized data transfer.
- (e) Untimed communications

[4 marks]

Q2.3 Consider that the following variable are defined:

Define f = fraction of computation that can be parallelized

Define n = number of processors for parallel case

Then, according to Amdahl's law, the maximum speed-up achievable for the parallel case over a sequential case is (chose one option below):

- (a) Speedup = $f / (f - 1)(n+1)$
- (b) Speedup = $(n + 1) / (1 - f)$
- (c) Speedup = $(1 - f) / ((1 - n \cdot f))$
- (d) Speedup = $1 / ((1 - f) + f/n)$ ← ANSWER
- (e) Speedup = $(1 - f) / (f/n - 1)$

[4 marks]

Q2.4 Which of the following is *not* an acronym describing one of Flynn's classifications for parallel computer architectures.

- (a) SISD
- (b) SIMS ← ANSWER
- (c) MIMD
- (d) SIMD
- (e) MISD.

[4 marks]

Q2.5 Answer **true** or **false** to each question below (each answer is 1 mark).

- (i) A correlation between two identical data sets returns the value 0. ← FALSE
- (ii) iVerilog is a free Verilog HD code compiler and simulator. ← TRUE
- (iii) 8 GFLOPS is the *peak computation rate* of a processor that can complete four FP operations per clock and is being clocked at 2 GHz. ← TRUE
- (iv) The commonly used HPEC acronym "SWAP" means SoftWare And Power matrix. ← FALSE

[4 x 1 mark each = 4 marks]

Section 3: Long Answers [38 marks]

Question 3.1 [14 marks]

Consider the Verilog code provided below and then answer (a) and (b) below.

```
/* Test Program for EEE4084F Final Exam 2012 */
module Test1 ( A, B, CLK, Enable, X, Y );
  input A, B, CLK, Enable;
  output X, Y;
  reg    temp;

  // On every positive edge...
  always@(posedge CLK)
    begin
      temp = A;
    end

  assign X = Enable & (temp & ~B);
  assign Y = Enable & (temp | B);
endmodule
```

- (a) Convert the Verilog code given above into VHDL code. You need not specify any library includes, just focus on the entity and behavioral implementation. [8 marks]

```
LIBRARY ieee;    -- to be correct these libraries should be included
USE ieee.std_logic_1164.all;

-- Define the Entity structure
ENTITY Test1 IS PORT (
  A:    IN  STD_LOGIC;
  B:    IN  STD_LOGIC;
  CLK:  IN  STD_LOGIC;
  C:    IN  STD_LOGIC;
  X:    OUT STD_LOGIC;
  Y:    OUT STD_LOGIC; );
END B1;

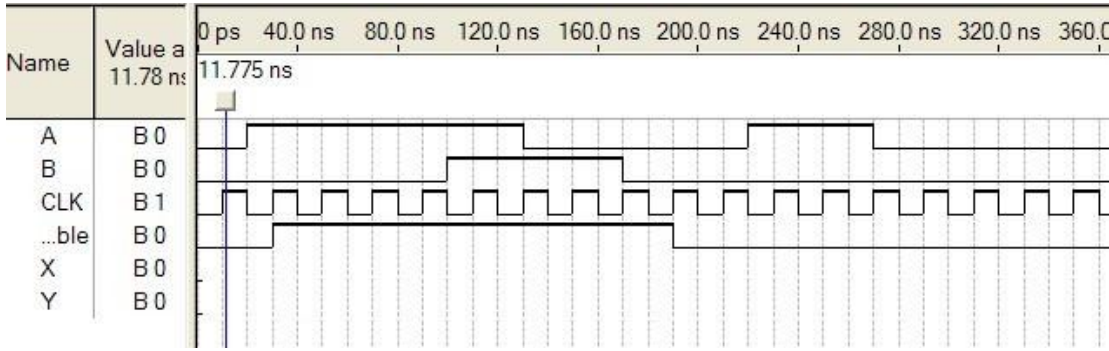
-- Implement the architecture (and give it a suitable name!!)
ARCHITECTURE Test1Bhv OF B1 IS
  SIGNAL temp : STD_LOGIC;
BEGIN
  -- On every positive edge...
  -- The operation on CLK changes can be handled as a process
  PROCESS(CLK)
  BEGIN
    IF (RISING_EDGE(CLK)) THEN
      temp <= A;
    END IF;
  END PROCESS;

  -- These are just wiring descriptions:
  X <= Enable AND (temp AND NOT(B));
  Y <= Enable AND (temp OR B);
END Test1Bhv;
```

Including libraries don't count for any marks. Might use it to reward a student's effort where they might not have got everything else right.

A mark for converting the comment to VHDL form

(b) The timing waveform configuration shown below is to be used to simulate the code given above. The simulation is set to work at 1ps resolution (i.e. much finer than the visualization can shown).



(i) At time 200ns will X be at 1 or 0? [3 marks]

Student gets 3 by just mentioning that X will be 0.

At 200ns, $X = \text{Enable} \& (\text{temp} \& \sim B) = 0 \& (\dots) = 0$, so $X = 0$ (or it should if the clock is slower than the propagation delay for the circuit, considering that Enable will be low at this stage).

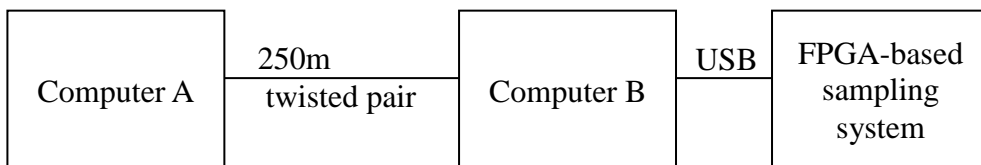
(ii) If X is initially 0, explain when X will first transition to 1. Clearly you can't give a perfectly accurate figure as you haven't been given propagation delay information (but you can assume the maximum propagation time for the circuit is faster than the CLK clock rate). [3 marks]

X will change to 1 soon after the second positive clock edge (considering that changes to A go through the temp register, which is effectively acting as a clocked flip flop). In simulation time, this will be somewhere between 30ns and 50ns.

Student gets the full 3 only if explained decently.

Question 3.2 [14 marks]

This question concerns networked computer systems and calculating effective bandwidth. There are two computer systems in the network, Computer A and Computer B connected via a 250m twisted pair connection. Computer B connects to a FPGA-based sampling system via a short USB cable.



The raw bandwidth for the twisted pair connection is 1Mbps (i.e. bits per second) between computer. The signal propagation in the copper twisted pair is $0.8c$ (i.e. 80% speed of light). The sending overhead, to start transmissions is 200us. The receiving overhead is 100us. Assume each byte is encoded as a 10-bit sequence for both the twisted pair link and for the USB link.

The FPGA based sampling system captures a set of 2Kbyte samples every 500ms. The sample block is send to Computer B from the FPGA via the 400MBps USB link. Computer B captures the whole sample block and sends the data on to Computer A via the twisted pair link.

Complete the following:

(a) What is the difference between *communication latency* and *bandwidth*? Present an example to aid your explanation that draws on the network system scenario given above. [4 marks]

Communication Latency = Time it takes to send a minimal length (e.g., 0 byte) message from one task to another. Usually expressed in time, e.g. factions of a second.

Bandwidth = The amount of data that can be sent per unit of time. Usually expressed as bytes per second or bit per second (Mbps).

(b) What is the *effective bandwidth* for sending the 2K byte block over the twisted pair link that runs between Computer A and B? [5 marks]

Total latency = Sending overhead + Transmission time + time of flight + Receiver overhead

Effective bandwidth = Message_size / total_latency

For this problem:

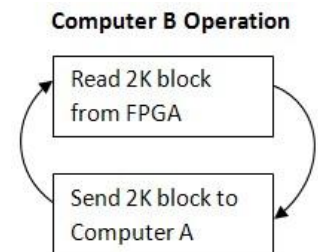
Sending overhead = 200us Receiver overhead = 100us ✓

Transmission time = $10 \times 2\text{K bits} / 1024 \times 1024 \text{ b/s} = 2 \times 10 / 1024 \text{ s} = 0.0195\text{s} = 19.5\text{ms}$ ✓

time of flight = $250\text{m} / (0.8 \times 3 \times 10^8 \text{ m/s}) = 1\text{us}$ ✓

Total latency = $200\text{us} + 19500\text{us} + 1\text{us} + 100\text{us} = 19801\text{us} = \underline{19.8\text{ms}}$ ✓✓

(c) Explain, with a clear argument using paper-based calculations, whether the connections described here will allow Computer B to continuously capture 2Kbyte blocks from the FPGA and send the data blocks to Computer A without missing any samples (i.e. as illustrated by the flowchart on the right). Assume the USB connection has no send and receive overhead (i.e. <10us), and ignore latency and delays in memory access performed for either computer. [5 marks]



Yes, it should work. Each block is transferred from FPGA to Computer B in:

$10 \times 2\text{K bits} / 500\text{Mbps} = 2\text{K} / 50\text{Mbps} = 2 \times 1024 / (50 \times 1024 \times 1024) = 2 / (50 \times 1024) = 39\text{us}$ ✓

The block is transferred from Computer B to Computer A in 19.8 ms, a total of slightly over 19.8ms in total. This is very comfortably within the 500ms window. ✓✓

Question 3.3 [10 marks]

Cloud Computing has become a popular topic in recent year.

(a) Explain clearly what cloud Computing is, using an example to help your explanation. Also briefly indicate how a company can make an income from this. [4 marks]

Cloud computing is a style of computing that uses dynamically scalable and (usually) virtualized computing resources to provide services over the internet. ✓✓

There are a variety of models that can be applied to produce an income from cloud computing, these include: ✓✓

- Utility computing: Rent cycles (e.g. Amazon's EC2, GoGrid, AppNexus)
- Platform as a Service (PaaS): provides / license a user-friendly API to aid implementation
- Software as a Service (SaaS): hire technician / consultant / system resources to 'just run it for me'
- Simple Storage Service (S3): Persistent storage, charge by the GB/month, additional costs for bandwidth

(Note to marker: student can provide any income example above or something that seems logical.)

(b) What is your view on the use of Cloud Computing in the next five years? In your discussion clarify the advantages and disadvantages of Cloud Computing; be particularly clear on how the potential disadvantages may be a hindrance to people using the technology, and how the advantages may help to improve the use of this technology. [6 marks] ✓✓

Marking of the five-year prediction is largely based on having a well reasoned and logical discussion. That said, I would expect students to say cloud computing will expand greatly; considering how popular Gmail and Google Docs have become, for an example, as well as cloud storage solutions, it is highly likely that more companies will start offering more cloud computing services. For standard tasks, like email,

wordprocessing, and games, cloud computing may well overtake standard application use (i.e. software installed on PCs). But in this country the update may likely be slower due to our internet being quite a bit more expensive – and slower – when compared to first world countries. Futuremore, just thinking from my own perspective as an electrical engineer, I think electrical engineers may well be slower in adopting to cloud computing resources, not necessarily for standard tasks like word processing, but rather for development tasks, like programming embedded hardware and having to have low-level program control.

A major hindrance to cloud computing is the internet connectivity: if your internet connecting is not incredibly reliable, people won't make the switch – it will be too much risk in the internet going down as not being able to access important files or loosing information (I for one have already experienced this problem too many time, so I tend to use things like Google Docs more as a file sharing mechanism as opposed to a means for actually editing documents). Similarly not having adequate internet speed is a hindrance – for example depending on where I am, my 3G dongle is sometimes too slow to use web-based storage or word-processing tools. ✓✓

The advantages are not having to install and maintain software on your own machine (besides the obvious things such as browser and internet connection settings). Also not having to perform upgrades – the people providing the service do this without the user needing to do anything (besides perhaps accepting pop-up question to change to a new version). Having access from any internet computer is also a advantage to improving the update of cloud computing – using the same tools and interface from any computer without having to install programs. ✓✓

END OF EXAMINATION