COMPUTER SCIENCE TRIPOS Part IB

Thursday 7 June 2007 1.30 to 4.30

PAPER 6

*Answer **five** questions.*
*No more than **two** questions from any one section are to be answered.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
None

SECTION A

1 ECAD

The following Verilog code describes a one-element FIFO.

```
module FIFO_one(clock, reset, dataInsert, insert, insertComplete,
                dataExtract, extract, extractComplete, full);

parameter nb = 7;  // num. bits: 7..0 bits (i.e. 8 bits) of data will be used
input  clock, reset;       // clock and reset signals
input  [nb:0] dataInsert;  // data to be inserted into the FIFO
input  insert;             // control: high means perform the insert
output insertComplete;     // control: high indicates data has been inserted
output [nb:0] dataExtract; // data extracted from the FIFO
input  extract;            // control: high means please perform an extract
output extractComplete;    // control: high indicates that dataExtract is valid
output full;               // control: high when the FIFO is full
reg    full;
reg    [nb:0] dataStore;
reg    insertComplete, extractComplete;

always @(posedge clock or posedge reset)
  if(reset) begin                          // comment A
    full <= 0;
    insertComplete <= 0;
    extractComplete <= 0;
    dataStore <= 8'bxxxxxxxx;
  end else begin
    full <= (insert || full) && !extract;  // comment B

    if(insert) begin                       // comment C
      if(!full) dataStore <= dataInsert;
      insertComplete <= !full;
    end
    else insertComplete <= 0;

    if(extract) begin                      // comment D
      extractComplete <= full || insert;
    end
    else extractComplete <= 0;
  end

assign dataExtract = dataStore;
endmodule
```

(*a*) What would be suitable comments on the behaviour of the code at points "comment A" to "comment D"? [4 marks]

(*b*) In the synthesised implementation, how will the `reset` and `clock` signals be connected to the D flip-flops that are used to hold the state inside the `always` block? [2 marks]

(CONTINUED ON NEXT PAGE)

(*c*) In Verilog a wire can transmit not only Boolean values `0` and `1`, but also the values `x` and `z`. How is `x` used in simulation and what will it be converted to when synthesised to real hardware? Illustrate your answer with reference to assignments to `dataStore` in `FIFO_one`. [3 marks]

(*d*) What is the state diagram describing the empty/full status of `FIFO_one`? Include the inputs `(insert, extract)` and outputs `(insertComplete, extractComplete)` on the arcs of the state diagram and ignore the data path. [4 marks]

(*e*) Is it possible to insert and extract data on the same clock cycle? [1 mark]

(*f*) How could two instances of `FIFO_one` be joined to produce a two-element FIFO? [4 marks]

(*g*) For your design in part (*f*), how many clock cycles of latency would there be from input to output if data were always extracted as quickly as possible? [2 marks]

2 Computer Design

instruction fetch	decode and register fetch	execute	memory access	write back

With reference to the pipeline above:

(*a*) What is a *control hazard* and how can it be dealt with? [4 marks]

(*b*) What are *data hazards* and how can they be eliminated? [4 marks]

(*c*) A branch could be executed in either the decode or the execute stages. Assuming that branch prediction and branch delay slots are not used, how many bubbles would be introduced into the pipeline in either case? [4 marks]

(*d*) If the memory access results in a cache miss, what happens to the pipeline? [4 marks]

(*e*) For arithmetic operations the result is available after the execute stage. These results could be written directly to the register file during the memory access stage, but what would be the disadvantages of doing so? [4 marks]

3 Digital Communication I

(*a*) Define the term *flow control* as used in communication networks. [4 marks]

(*b*) Describe on–off flow control, window-based flow control, and flow control used in circuit switching. [9 marks]

(*c*) Consider a channel of capacity b and delay τ, over which packets of size p are sent. Compare the performance of window-based flow control protocols having:

(*i*) a window size of one packet;

(*ii*) a window size of two packets; and

(*iii*) a window size of one packet, but with a packet size of $2p$.

[7 marks]

4 Concurrent Systems and Applications

(*a*) For *each* of the following tasks, give a code fragment that achieves as much of the task as is possible using the introspection API of the Java programming language and state which aspect(s) of the requirement is/are impossible in Java.

(*i*) Given a (non-null) object reference, determine whether or not the object has a *public*, *static* method named `myMethod` which takes no arguments. [4 marks]

(*ii*) Invoke the static method `public foo(java.lang.Integer x)` on a class definition named `MyClass` with argument `myInt` when the overloaded, static method `foo(java.lang.Number x)` is also defined on `MyClass`. [4 marks]

(*iii*) Given an object reference to an instance of a class named `Rocket`, set the value of its public field `numberOfEngines` to the (primitive) `int` value `5` and make the method `launch()` into a synchronized method. [4 marks]

(*b*) You are porting the JVM to a new processor that does not have a compare-and-swap (CAS) instruction but does offer test-and-clear (TAC): `tac(addr)` atomically reads the value stored at memory address `addr`, overwrites it with zero, and returns the value that was seen. Construct a Java-style *re-entrant* mutex using TAC. [4 marks]

(*c*) A server daemon has an object of type `Client` for each currently-active connection. Instances of `Client` each contain an object reference to a `java.lang.Socket` which must be closed (by calling `close()`) when the `Client` object is garbage collected. Show, by means of Java code fragments, how Phantom References and Reference Queues can be used to invoke the `close()` method in a timely fashion following an instance of `Client` becoming unreachable. [4 marks]

SECTION B

5 Computer Graphics and Image Processing

(*a*) Explain what a *MIPmap* is, how to create one, why one would want to use one, where one would be used, and how one is used. [8 marks]

(*b*) Describe an algorithm that converts a greyscale image into a black and white image using halftoning. Assume that the black and white image has eight times the resolution of the greyscale image in each dimension. [6 marks]

(*c*) Various types of visual artifact ("aliasing") occur if images are rendered using only one sample per pixel.

(*i*) Describe at least *three* different artifacts that occur. [3 marks]

(*ii*) Describe a straightforward method to ameliorate these artifacts. [3 marks]

6 Compiler Construction

(*a*) Garbage Collection.

(*i*) Explain how it is possible to "leak memory" using a reference counting garbage collector. [3 marks]

(*ii*) Describe any technique that might be used to address this problem. [3 marks]

(*b*) Explain in detail how we might translate code generated for a stack-only machine (such as the JVM) to a register-based machine (such as ARM). [6 marks]

(*c*) Describe in detail how static and dynamic methods are compiled differently for object-oriented languages with single inheritance. [8 marks]

7 Concepts in Programming Languages

(a) Give an overview of the LISP abstract machine (or execution model) and comment on its merits and drawbacks from the viewpoints of programming, compilation, execution, etc. [5 marks]

(b) Define the following parameter-passing mechanisms: pass-by-value, pass-by-reference, pass-by-value/result, and pass-by-name. Briefly comment on their merits and drawbacks. [5 marks]

(c) What is *aliasing* in the context of programming languages? Explain the contexts in which it arises and provide examples of the phenomenon. [5 marks]

(d) Consider the Simula declarations

```
CLASS A; A CLASS B;
```

which have the effect of producing the subtype relation `B<:A`, and

```
REF(A) a; REF(B) b;
```

Recall that Simula uses the semantically incorrect principle that

if `B<:A` then `REF(B)<:REF(A)`

and consider now the following Simula code

```
PROCEDURE ASSIGNa( REF(A) x )
    BEGIN x :- a END ;

ASSIGNa(b);
```

Does it statically type check? If so, will it cause a run-time type error?

Justify your answers. [5 marks]

8 Databases

(*a*) Define the notion of a *functional dependency.* [2 marks]

(*b*) Consider the following "rule" for functional dependencies.

$$\text{if } A \to B \text{ and } B, C \to D, \text{ then } A, C \to D.$$

Either prove this rule is correct, or present a counter-example showing that the rule is false. [4 marks]

(*c*) The *union rule for functional dependencies* states that if $F \models X \to Y$ and $F \models X \to Z$, then $F \models X \to Y \cup Z$ (this can also be written as $F \models X \to Y, Z$).

Prove this rule using only Armstrong's axioms. [4 marks]

(*d*) Suppose that $R(A, B, C)$ is a relational schema. Write a relational algebra query that evaluates to the empty set exactly when the functional dependency $B \to C$ holds on R. [4 marks]

(*e*) The schema $R(A,\ B,\ C,\ D,\ E)$ has the following functional dependencies.

$$\begin{aligned} &A \to B, C \\ &C, D \to E \\ &B \to D \\ &E \to A \end{aligned}$$

Is D, E a candidate key for R? Explain your answer. [6 marks]

SECTION C

9 Logic and Proof

For *each* of the following formulae, state (with justification) whether it is satisfiable, valid or neither:

$$((P \vee Q) \to R) \leftrightarrow (P \to (Q \to R))$$

$$(P \wedge \neg Q) \vee (\neg R \wedge Q) \vee (R \wedge \neg P) \vee (\neg P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge R)$$

$$\Big[\forall x\, \exists y\, (P(x) \to Q(x, y)) \wedge \forall x\, \exists z\, \forall y\, (\neg Q(x, y) \to P(z))\Big] \to \exists x\, \forall y\, Q(x, y)$$

[4+7+9 marks]

10 Semantics of Programming Languages

Concurrent threads can interfere with each other by accessing the same store, so their behaviour can be nondeterministic and hard to reason about. This question develops a simple sufficient condition to rule that out, showing that two threads that do not share any store locations cannot interfere with each other's behaviour.

Consider the following mild variant of $\mathsf{L1}_{|}$, with distinguished grammars of expressions and processes, and corresponding reduction relations $\longrightarrow$ and $\twoheadrightarrow$.

Integers $n \in \mathbb{Z}$ Locations $\ell \in \mathbb{L} = \{l, l_0, l_1, l_2, \ldots\}$
Expressions $e ::= n \mid \mathbf{skip} \mid !\ell \mid \ell := e \mid e_1; e_2$ Processes $p ::= e \mid p_1 \mid p_2$
Stores s, finite partial functions from $\mathbb{L}$ to $\mathbb{Z}$

(e-deref) $\langle !\ell, s\rangle \longrightarrow \langle n, s\rangle$ if $\ell \in \mathrm{dom}(s)$ and $s(\ell) = n$

(e-assign1) $\langle \ell := n, s\rangle \longrightarrow \langle \mathbf{skip}, s + \{\ell \mapsto n\}\rangle$ if $\ell \in \mathrm{dom}(s)$

(e-assign2) $$\frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle \ell := e, s\rangle \longrightarrow \langle \ell := e', s'\rangle}$$

(e-seq1) $\langle \mathbf{skip}; e_2, s\rangle \longrightarrow \langle e_2, s\rangle$

(e-seq2) $$\frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1; e_2, s\rangle \longrightarrow \langle e_1'; e_2, s'\rangle}$$

(p-thread) $$\frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle e, s\rangle \twoheadrightarrow \langle e', s'\rangle}$$

(p-par1) $$\frac{\langle p_1, s\rangle \twoheadrightarrow \langle p_1', s'\rangle}{\langle p_1 | p_2, s\rangle \twoheadrightarrow \langle p_1' | p_2, s'\rangle}$$

(p-par2) $$\frac{\langle p_2, s\rangle \twoheadrightarrow \langle p_2', s'\rangle}{\langle p_1 | p_2, s\rangle \twoheadrightarrow \langle p_1 | p_2', s'\rangle}$$

Write $s \uplus s'$ for the union of two stores that have disjoint domain, and let $\mathrm{loc}(e)$ denote the set of store locations mentioned in e.

(a) Give a counterexample to [One-step determinacy for processes]:
If $\langle p, s\rangle \twoheadrightarrow \langle p_1, s_1\rangle$ and $\langle p, s\rangle \twoheadrightarrow \langle p_2, s_2\rangle$ then $\langle p_1, s_1\rangle = \langle p_2, s_2\rangle$. [1 mark]

(b) Prove [One-step determinacy for expressions]:
If $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ and $\langle e, s\rangle \longrightarrow \langle e_2, s_2\rangle$ then $\langle e_1, s_1\rangle = \langle e_2, s_2\rangle$. [5 marks]

(c) Assume [Irrelevant store can be added]:
If $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ and $\mathrm{dom}(s) \cap \mathrm{dom}(s_0) = \{\}$ then $\langle e, s \uplus s_0\rangle \longrightarrow \langle e_1, s_1 \uplus s_0\rangle$.

(d) Prove [Irrelevant store can be removed]:
If $\langle e, s \uplus s_0\rangle \longrightarrow \langle e_1, s_1\rangle$ and $\mathrm{loc}(e) \subseteq \mathrm{dom}(s)$ then there exists s' such that $\langle e, s\rangle \longrightarrow \langle e_1, s'\rangle$ and $s_1 = s' \uplus s_0$. [8 marks]

(e) Using (b), (c) and (d), prove [One-step confluence for store-disjoint threads]:
If $p = e_1 | e_2$, $\mathrm{loc}(e_1) \cap \mathrm{loc}(e_2) = \{\}$, $\langle p, s\rangle \twoheadrightarrow \langle p', s'\rangle$, and $\langle p, s\rangle \twoheadrightarrow \langle p'', s''\rangle$, then either $\langle p', s'\rangle = \langle p'', s''\rangle$ or $\exists\, p''', s'''.\ \langle p', s'\rangle \twoheadrightarrow \langle p''', s'''\rangle \wedge \langle p'', s''\rangle \twoheadrightarrow \langle p''', s'''\rangle$. [6 marks]

11 Foundations of Functional Programming

(*a*) Define transformations that translate the λ-calculus to and from a language using only combinators and application. [8 marks]

(*b*) Define transformations that translate the λ-calculus to and from continuation-passing style. [8 marks]

(*c*) Indicate the complexity of your given transformations above in terms of size of output term as a function of size of input term. [4 marks]

12 Complexity Theory

Let **Bounded Factor** denote the following decision problem:

> Given positive integers n and k, decide whether n has a proper factor that is less than k.

For each of the following questions, give a detailed justification of your answer. Note that, in some cases, the answer may not be a simple "yes" or "no" but may instead be linked to open problems in complexity theory such as whether P=NP or the existence of one-way functions. In such cases, you should clearly explain the links and state what the consequences might be of both a positive and a negative answer to the question.

(*a*) Is **Bounded Factor** in NP? [4 marks]

(*b*) Is **Bounded Factor** in co-NP? [4 marks]

(*c*) Is **Bounded Factor** NP-hard? [6 marks]

(*d*) Is **Bounded Factor** in P? [6 marks]

END OF PAPER