UNIVERSITY OF CAMBRIDGE

COMPUTER SCIENCE TRIPOS Part IB

Wednesday 1 June 2016 1.30 to 4.30

COMPUTER SCIENCE Paper 5

Answer ***five*** *questions.*

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
Approved calculator permitted

1 Computer Design

Below is a functionally and syntactically correct mysterious module written in SystemVerilog.

```
typedef enum { opNone, opIn, opOut } operationT;

module mystery
  # (parameter depth, parameter width)
   (
   input  clk,
   input  rst,
   input  operationT op,
   input  logic [width-1:0] dataIn,
   output logic [width-1:0] dataOut,
   output logic empty,
   output logic full,
   output logic error);

   logic [width-1:0] mem[depth-1:0];
   reg [$clog2(depth-1)+1:0] head;
   // where $clog2(x) = ceiling(log_base_2(x))

   always_comb
     begin
        full    = head>=depth;
        empty   = head==0;
        error   = ((op==opIn) && full) || ((op==opOut) && empty);
        dataOut = empty ? -1 : mem[head-1];
     end

   always @(posedge clk)
     if(rst)
       head <= 0;
     else
       if(!error)
         case(op)
           opIn: begin head <= head+1; mem[head] <= dataIn; end
           opOut:      head <= head-1;
         endcase // case (op)
endmodule
```

(*a*) What is the function of the `mystery` module? Include in your answer the behaviour when the module is full (`full==1`) or empty (`empty==1`). What does `dataOut` output? What does input `op` do? [8 marks]

(*b*) What is production test and how does it differ from functional test? [2 marks]

(*c*) What are the key challenges in functionally testing the `mystery` module? [5 marks]

(*d*) What are the challenges in undertaking a production test of the `mystery` module and how do these challenges compare with those for functional test? [5 marks]

2 Computer Design

(*a*) Why is there a risk of data and control hazards in a pipelined processor? [4 marks]

(*b*) For modern RISC instruction set architectures like RISC-V, why are control hazards not exposed in the programming model? [4 marks]

(*c*) How might data hazards be mitigated in each of the three pipelines below? [6 marks]

Pipeline A:

instruction fetch	decode	register fetch execute memory access write-back

Pipeline B:

instruction fetch	decode register fetch	execute memory access write-back

Pipeline C:

instruction fetch	decode register fetch	execute	memory access	write-back

(*d*) Do exceptions introduce a control hazard? Give justification for your answer. [3 marks]

(*e*) Do interrupts introduce a control hazard? Give justification for your answer. [3 marks]

3 Computer Design

(*a*) (*i*) Describe the four classes in Flynn's taxonomy of computing systems. [4 marks]

(*ii*) Describe Amdahl's law and use it to calculate the maximum speedup achievable by running a program, P, on a multicore system, S, where 80% of P is parallel and S contains 16 cores. [4 marks]

(*b*) Consider the following pseudo-code that is run on a SIMD processor with 8 lanes, where `i` gives the lane number.

```
r1 = load X[i]
r2 = load Y[i]
if (i%2 == 0)
  if (i%8 == 0)
    r1 = r1 * 2
    r1 = r1 + r2
  endif
else
  r1 = r1 - r2
endif
store r1, X[i]
```

(*i*) Describe how the processor can support branch divergence between the different lanes. [4 marks]

(*ii*) With the aid of a diagram, show the utilisation of the SIMD lanes for each pseudo-code operation, hence calculate the code's efficiency (overall utilisation of the SIMD lanes). [6 marks]

(*iii*) What architectural technique do GPUs employ to allow them to perform useful work even though the loads from `X` and `Y` often cause stalls? [2 marks]

4 Computer Networking

Two objects are being retrieved from B by A using HTTP over a network where the TCP-style transport protocol has no maximum segment size and the network experiences no loss. Each object is 125 kByte (i.e., one Mbit) in size. We assume all connection setup packets and HTTP request packets are negligible in size; we ignore the connection tear-down.

(*a*) Suppose the Round-Trip-Time (RTT) between A and B is 10 milliseconds and the bandwidth between the sites is 10 Mbit/s. Illustrate with a simple diagram in each case how long it takes for A to retrieve both files under the following circumstances: [2 marks each]

(*i*) Sequential (one-at-a-time) requests with non-persistent connections.

(*ii*) Concurrent requests with non-persistent connections.

(*iii*) Sequential requests within a single persistent connection.

(*iv*) Pipelined requests within a single persistent connection.

(*b*) Suppose A is connected to a cache C by a link with 10 Gbit/s bandwidth and negligible RTT. C connects to B with the link originally specified in part (*a*): 10 Mbit/s and 10 millisecond RTT.

The cache operates as follows:

- If the object is not in the cache, the request is forwarded to B which responds with the object, which the cache stores with an object timeout and then forwards to A.
- If the object is in the cache, and the cache entry has not timed out (i.e., the cache Time To Live (TTL) has not expired), the object is returned to the client.
- If the object is in the cache, but the cache entry has timed out, the cache issues a conditional GET to the original server, asking if the object has changed since this object was cached. If the original server responds that it has not, the cache returns the cached object, otherwise the original server responds with the updated object which the cache forwards to the client.

Showing your working, how long does it take for A to retrieve one file under the following circumstances: [3 marks each]

(*i*) The file is not in the cache

(*ii*) The file is in the cache and the TTL has not expired

(*iii*) The file is in the cache, the TTL has expired, but the file has not changed

(*iv*) The file is in the cache, the TTL has expired, and the file has changed

5 Computer Networking

(*a*) This question concerns IPv6 notation. [2 marks]

(*i*) Rewrite the following IPv6 address in the most compact form.
3ffe:1944:0000:0000:0010:0000:0000:d00b

(*ii*) In IPv6 notation, what do ::/0 and ::1/128 represent?

(*b*) Describe five differences between IPv4 and IPv6. [5 marks]

(*c*) Describe, giving reasons, how an Internet Service Provider (ISP) may use their allocated IPv6 addresses to provide address blocks to their (residential) customers. [3 marks]

(*d*) What are the implications of IPv6 for the Network Address Translation (NAT) gateways and firewalls of residential customers? [4 marks]

(*e*) Give one advantage and one disadvantage that Stateless Address Autoconfiguration (SLAAC) has compared to DHCPv4. [2 marks]

(*f*) Suggest two features that would be desirable extensions of IPv6. Justify your answer. [4 marks]

6 Computer Networking

(*a*) Consider packet switching and circuit switching. In simple terms, packet switching may allow more users to use the network.

We have N users sharing a 1 Mbit/s link. Each user consumes 100 kbit/s when transmitting but only transmits for 10% of the time, on average.

(*i*) How many users would a circuit-switching system support? State your assumptions. [1 mark]

(*ii*) For $N = 40$ users, the probability that more than 10 users are active at the same time is approximately 0.0015.

Show how to compute this probability. You are not required to calculate the actual figures. [6 marks]

(*iii*) State two assumptions that you made in Part (*a*)(*ii*) about the network users. In each case describe why the assumption may fail for current Internet traffic. [4 marks]

(*b*) The formulae below are used in TCP implementations to compute a value for the retransmission time-out T. R is an estimate of the round-trip time (RTT), D is an estimate of variance, M is the most recently measured round-trip measurement, $\alpha = 0.875$ and $h = 0.25$.

$$\begin{aligned} D &\leftarrow D + h(|M - R| - D) \\ R &\leftarrow \alpha R + (1 - \alpha)M \\ T &= R + 4D \end{aligned}$$

(*i*) Give an example of how the retransmission time-out T is used within TCP. [1 mark]

(*ii*) Describe why the computation of the retransmission time-out T incorporates a correction for deviation in the estimate of the RTT. [2 marks]

(*iii*) For each assumption you stated in Part (*a*)(*iii*), describe the impact on the estimate of the retransmission time-out T. [3 marks each]

7 Concurrent and Distributed Systems

This question is with respect to an operating system that supports multi-threaded processes using the POSIX threads (pthreads) API. Assume that each call to printf prints its output atomically, that thread scheduling is non-deterministic, and that threads are allocated unique and sequential integer IDs starting with 0.

(*a*) Some program state is per-process, and some is per-thread. How many instances of each of the following will a 2-thread process have: *virtual address space*, *executable program*, *register file*, *scheduling state* (e.g., RUN, SLEEP), and *stack*? [5 marks]

(*b*) A programmer adds printfs to a concurrent program to debug a race condition, but the symptoms vanish. Explain why this might have happened. [2 marks]

(*c*) thrprint accepts as arguments the current thread's unique ID and a debug message to print. If each thread calls thrprint exactly once on start, how many possible interleavings are there with n threads? [2 marks]

```
void thrprint(int threadid, char *message) {
    printf("Thread %d: %s\n", threadid, message);
}
```

(*d*) ordered_thrprint attempts to print debug messages ordered by thread ID. Describe three ways in which the synchronisation in this implementation is incorrect, and provide a corrected pseudocode implementation. [6 marks]

```
int             next_thread_id = 0;  // Next ID to print
pthread_mtx_t   ordering_mtx;        // Lock protecting next ID
pthread_cond_t  ordering_cv;         // next_thread_id has changed

void ordered_thrprint(int thread_id, char *message) {
    pthread_mtx_lock(ordering_mtx);
    if (thread_id != next_thread_id) {
        pthread_cond_wait(ordering_cv, ordering_mtx);
    }
    next_thread_id = next_thread_id + 1;
    pthread_mtx_unlock(ordering_mtx);
    printf("Thread %d: %s\n", thread_id, message);
}
```

(*e*) This approach to implementing ordered_thrprint suffers a substantial performance problem: if lower-numbered threads are slow in starting, then higher-numbered threads will also be delayed. Describe an alternative strategy, paying particular attention to synchronisation, that maintains ordered output while allowing greater concurrency. [5 marks]

8 Concurrent and Distributed Systems

History graphs record dependencies between individual atomic operations within sequences of events associated with specific schedules of more complex *transactions*.

(*a*) (*i*) What do *edges* in a history graph represent? [1 mark]

(*ii*) What graph property holds if a *bad schedule* is present? [1 mark]

(*iii*) Which ACID properties may be violated by a bad schedule? [2 marks]

(*iv*) Define *serial* and *serialisable* executions. Explain whether (and if so, how) one is a superset of the other. [3 marks]

(*b*) Two transactions, **T1** and **T2**, consist of operations on two objects, **A** and **B**:

```
T1: {                                T2 (v): {
    a = A.getbalance();                  A.debit(v);
    b = B.getbalance();                  B.credit(v);
    return (a + b);                  }
}
```

(*i*) Explain how a *dirty read* might be experienced through concurrent executions of **T1** and **T2**. [2 marks]

(*ii*) Draw and label a history graph illustrating this bad schedule. [2 marks]

(*c*) A programmer designs a transaction system that uses history graphs to detect bad schedules. After an operation is performed, and before its containing transaction is allowed to commit, the history graph is updated and a graph analysis is run. If a bad schedule is detected, affected transactions will be aborted and rolled back.

(*i*) Will this scheme always make progress? Explain your answer. [2 marks]

(*ii*) *Time Stamp Ordering (TSO)* will sometimes reject good schedules, which could lead to unnecessary transaction aborts. Does the scheme described here accept or reject more schedules than TSO? Explain why. [3 marks]

(*iii*) Explain one way in which this scheme may perform better than TSO. Explain one way in which it may perform worse. [4 marks]

9 Concurrent and Distributed Systems

Remote Procedure Call (RPC) allows procedures (functions, methods) to be forwarded over the network, and is a fundamental building block of distributed systems such as the Network File System (NFS).

(*a*) Explain, with respect to a client, what it means for an RPC call to be:

(*i*) *Synchronous* [1 mark]

(*ii*) *Asynchronous* [1 mark]

(*iii*) *Idempotent* [1 mark]

(*b*) Distributed-filesystem clients utilize different tradeoffs between performance and consistency for operations on the directory namespace (e.g., file or directory creation) versus those on file data itself. The following program creates and opens a file `foo`, writes some data to it, and closes it, via NFSv3:

```
fd = open("foo", O_CREAT | O_RDWR, 0755);
// POINT A
write(fd, data, sizeof(data));
// POINT B
close(fd);
// POINT C
```

For each of points *A*, *B*, and *C* in the program, discuss whether or not another NFSv3 client is guaranteed to be able to see the results of each of `open` and (if it has been called) `write`. [5 marks]

(*c*) The developers of a new distributed filesystem use a *persistent log* to ensure *all-or-nothing* retry semantics for filesystem operations. However, this comes at a substantial performance cost. Describe two circumstances under which filesystem RPCs can be excluded from the log while maintaining correctness, and give an example of each. [4 marks]

(*d*) The distributed filesystem's persistent RPC log is maintained in a *write-ahead log* stored in local disk blocks. Explain why large RPCs may require special care in the server-side log implementation. [2 marks]

(*e*) RPC protocols and write-ahead logging both rely on unique ID numbers. Explain why reusing client-generated RPC IDs for server log transaction IDs could harm each of *correctness*, *scalability*, and *security.* [6 marks]

END OF PAPER