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
SECTION A

1 Computer Design

(a) In SystemVerilog, what is the difference between:

(i) The ternary operator ? and if...then...else statements? [2 marks]

(ii) always_ff and always_comb? [2 marks]

(iii) Blocking, non-blocking and continuous assignment? [3 marks]

(iv) Logic values 0, 1, x and z and how these values propagate through Boolean logic gates? [3 marks]

(v) The way that synchronous and asynchronous reset are declared in an always_ff statement? [2 marks]

(b) The following module attempts to implement a reset control circuit that should have the following behaviour: when the user-controlled reset button (which needs to be debounced) is pressed the asyncButton signal is high and should result in the rst going high and remaining high for a minimum of $10^6$ clock cycles. rst should be generated immediately after the rising clock edge to allow time for it to propagate.

```verilog
module timeResetBad(input logic clk, 
                    input logic asyncButton, 
                    output logic rst);
logic ctr [18:0];
logic ctrAtMax;
always_comb begin
    ctrAtMax = &ctr;
    rst = !ctrAtMax;
end
always_ff @(posedge clk)
    ctr <= asyncButton ? 0 : !ctrAtMax ? ctr+1 : ctr;
endmodule
```

(i) What is wrong with the timeResetBad module? [4 marks]

(ii) Write a corrected version timeResetBad that makes minimal changes and adds no new modules. [4 marks]
2 Computer Design

(a) Describe each of the four models defined by OpenCL’s specification. [4 marks]

(b) Describe the different types of memory available to OpenCL kernels. [4 marks]

(c) Contrast how calls to a kernel, e.g. DAXPY, are invoked and grouped for execution in OpenCL compared with CUDA. [4 marks]

(d) Describe, with the aid of a diagram, how a GPU executes data-parallel kernels efficiently, including the two main pieces of hardware support. [4 marks]

(e) Describe the trade-offs between using a GPU or a specialised accelerator for tasks containing data-level parallelism. [4 marks]
3 Concurrent and Distributed Systems

(a) In the Network Time Protocol (NTP), a client (C) and a server (S) exchange (request, reply) messages to compute corrections to the time at C. Assume the time at S is always correct, and that C is synchronised to S at 13:30:00.

(i) Thirty days later the time at S is again 13:30:00 but C now believes the time to be 13:28:30. Define and compute skew and drift for C. [2 marks]

(ii) NTP estimates the offset and delay using four timestamps \(T_0, T_1, T_2, T_3\) from a request-reply message exchange. Two such exchanges occur between C and S, producing timestamps \((310.000, 400.100, 400.102, 310.202)\) and \((311.000, 401.150, 401.160, 311.410)\) respectively, denoting all timestamps as seconds since a common fixed point. Show on a diagram the point in the message exchange at which each timestamp \(T_0 \ldots T_3\) is taken. Give definitions for offset and delay, and compute both for each set of timestamps. Which of the two offsets you have computed would you prefer to use to adjust the time at C, and why? [5 marks]

(iii) What happens to your estimates of offset and delay if network delays are no longer symmetric? [2 marks]

(b) It is often necessary to agree only on the ordering of events, not their times.

(i) \(x \rightarrow y\) indicates event \(x\) happens-before event \(y\). Define happens-before. Explain why it provides only a partial order on events. [2 marks]

(ii) Vector clocks can be used to implement happens-before. Give the vector clock values at each event, \(a \ldots g\), and explain whether each of the following relations is true or false: \(b \rightarrow c, c \rightarrow e, c \rightarrow f, d \rightarrow g\). If false, give the relation that does hold between the given pair of events. [8 marks]

(iii) An earlier approach used Lamport Clocks, defining \(L(x)\) such that, for two events \(x\) and \(y\), \(x \rightarrow y \Rightarrow L(x) < L(y)\) but \(L(x) < L(y) \nRightarrow x \rightarrow y\). Explain how vector clocks resolve this issue and ensure \(L(x) < L(y) \Rightarrow x \rightarrow y\). [1 mark]
4 Concurrent and Distributed Systems

(a) Programs with concurrency are vulnerable to classes of problems that are not exhibited in single-threaded programs.

(i) Explain the concepts of deadlock and livelock in a multithreaded program. [2 marks]

(ii) Explain the four conditions required for deadlock. A sentence explaining each condition is sufficient. [4 marks]

(b) Modern instruction set architectures provide instructions for performing atomic operations over memory locations.

(i) One class of instructions are generically referred to as Compare And Swap (CAS). Describe how the CAS instruction on the x86 architectures is used to perform an atomic operation. Briefly explain how the instruction uses the instruction operands when executed. [3 marks]

(ii) Databases often utilise a technique known as write-ahead logging to provide durability guarantees. Describe how a disk-based transaction log might be implemented, and what the atomic operation used in this technique is. [3 marks]

(c) The below snippet of C code uses pthreads for concurrent execution. It uses a mutex M and a condition variable C to ensure that the run_critical_code function only executes when the condition boolean is true.

L1:   pthread_mutex_lock(&M);
L2:   run_critical_code ();
L3:   if (!condition)
L4:   pthread_cond_wait(&C, &M);
L5:   if (!condition)
L6:   pthread_cond_broadcast(&C);
L7:   pthread_mutex_lock(&M);

Unfortunately, there are four bugs in the code that prevent it from working correctly. List each of the four bugs and describe how each bug affects programme execution. Write down a new version of the code snippet with the four bugs corrected. [8 marks]
SECTION C

5 Digital Electronics

(a) (i) Show that

\[ A + B.C = (A + B).(A + C) \]  

[2 marks]

(ii) Using the distributive law in Part (a)(i), express the following equation in product of sums form with 4 product terms, each with a sum of 3 variables:

\[ F = H + I.J + K.L \]  

[4 marks]

(b) (i) Write down the truth table for the logic unit (LU) defined in the following table that can execute one of 4 logical operations at a time on 2 data inputs \((Y_1, Y_0)\), to yield output \((Z)\). The LU is under the control of inputs \((I_1, I_0)\) and the logical operations are encoded to 2-bit instruction codes as follows:

<table>
<thead>
<tr>
<th>Operation</th>
<th>OR</th>
<th>AND</th>
<th>XOR</th>
<th>NOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction code ((I_1 I_0))</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

Note that operations OR, AND and XOR have their usual meanings and that the execution of NOP implies \(Z\) can take any binary value.

[3 marks]

(ii) Use a Karnaugh Map to determine a simplified expression for \(Z\) in Part (b)(i).

[2 marks]

(c) (i) Show using a circuit diagram how \(W\) can be implemented in 2-level sum of products form using AND gates followed by OR gates. Remember to include any NOT gates required since only uncomplemented input variables are available:

\[ W = \overline{B} \overline{C} + \overline{A}B.C + A.C.\overline{D} \]  

[2 marks]

(ii) Consider the implementation in Part (c)(i). Assume that the gates have finite propagation delay. Describe what happens at \(W\) when inputs \(\{A, B, C, D\}\) change from \(\{1, 0, 1, 0\}\) to \(\{1, 0, 0, 0\}\).

[3 marks]

(iii) Determine the other single input variable change that will give a similar problem to that observed in Part (c)(ii).

[2 marks]
(iv) Determine a modified sum of products expression for $W$ that will eliminate the problems observed in Part (c)(ii) and Part (c)(iii).

[2 marks]
6 Digital Electronics

(a) (i) Briefly describe the ways in which sequential logic differs from combinational logic.

(ii) Describe the main features that differentiate synchronous from asynchronous sequential logic.

(b) A synchronous 3-bit counter implemented using D-type Flip-Flops has a mode control input $M$. When $M = 0$, the counter output sequence represented in decimal form is 0, 1, 2, 3, 4, 5, 6, 7, and repeat. When $M = 1$, the counter output sequence represented in decimal form is 0, 1, 3, 2, 6, 7, 5, 4, and repeat. The Flip-Flop outputs are $\{Z_2Z_1Z_0\}$ where $Z_0$ represents the least significant bit of the counter output.

(i) Draw a state diagram that describes this counter.

(ii) Write down the state transition table corresponding with the state diagram in Part (b)(i).

(iii) Determine the excitation combinational logic in sum of products form for D-type Flip-Flop input $D_0$, i.e., the input of the Flip-Flop that represents the least significant bit of the counter. Show that the required combinational logic can be implemented using a 2-input XNOR gate plus some other combinational logic gates.

(c) Use row matching to reduce the number of states required to represent the single input $(X)$, single output $(Z)$, Mealy finite state machine described in the following state transition table:

<table>
<thead>
<tr>
<th>Current State</th>
<th>Next State $X = 0$</th>
<th>Next State $X = 1$</th>
<th>Output $(Z)$ $X = 0$</th>
<th>Output $(Z)$ $X = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>E</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>B</td>
<td>C</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Draw the resulting state diagram. [5 marks]
SECTION D

7 Programming in C and C++

(a) Find at least 2 sources of undefined behaviour in the following program, and write a corrected version of this function. [5 marks]

```c
int main(void) {
    char *s = "abcde"; int len = strlen(s);
    for (int i = 0; i <= len; i++)
        s[i] += 1;
    return printf("'%s' is %d characters long\n", ++s, strlen(s));
}
```

(b) Restructure the program below to be more cache-efficient, giving the code and explaining your changes. [5 marks]

```c
typedef struct point { double x, y, z; } Point;

int find_max_x_argument(int n, Point *elems) {
    double max = 0; int max_index = 0;
    for (int i = 0; i < n; i++)
        if (max < elems[i].x) { max_index = i; max = elems[i].x; }
    return max_index;
}
```

(c) The following definition forms part of a legal C++ program:

```cpp
int foo() {
    MyClass x(1,2);
    MyClass y = C(3,4);
    MyClass z = x;
    MyClass t;
    z = x;
    z.f = x.f;
    return z.f;
}
```

(i) Give a declaration of MyClass which enables foo to compile and run, noting any methods or constructors in MyClass which are invoked when foo is called. [Note: Precise C++ syntax is not necessary to obtain full marks.] [4 marks]

(ii) Having seen your declaration of MyClass, a colleague points out some of the lines of foo may be redundant. Which are these? [2 marks]

(iii) Your boss now replaces your declaration of MyClass. Not having access to the new declaration, explain, giving reasons, which if any lines of foo are now redundant. [4 marks]
8 Programming in C

(a) The following function is specified to return the quotient of two integers, returning zero when the answer is undefined.

```c
#include <stdint.h>
#include <limits.h>

int64_t divide(int64_t x, int64_t y) {
    return x / y;
}
```

(i) Identify two bugs in this program.

(ii) Write a correct version of this program.

[6 marks]

(b) The `strlen` function takes a valid C string as an argument, and returns the length of the string up to and not including the first null character. An (erroneous) implementation is given below:

```c
#include <stddef.h>

size_t strlen(const char *s) {
    size_t i;
    while (s[i] >= 0)
        i++;
    return i;
}
```

(i) Find two errors in this program.

(ii) Give a correct implementation of this function.

[6 marks]

(c) Write a function with the prototype

```c
void rotate(int len, int *array, int k)
```

which rotates its input k elements to the right. E.g., if the input `array` is the array `[0, 1, 2, 3, 4, 5]`, then the call `rotate(6, array, 2)` should result in `array` being modified to `[4, 5, 0, 1, 2, 3]`. Assume the array length is passed in the `len` argument and $0 \leq k < len$.

[8 marks]