THE UNIVERSITY OF MELBOURNE

Semester 2, 2020 Exam
November 2020

Melbourne School of Engineering
ELEN90066 Embedded System Design

Writing time: 180 minutes
Reading time: 15 minutes
Scanning and submission time: 30 minutes

This paper has 12 pages

Authorised materials:
This is an open book, hand-written exam and the following materials are permitted:

- Any material loaded onto Canvas as part of the subject content
- Notes (printed, hand-written and digital/electronic)
- Online books and materials
- Language dictionaries
- Calculators (any model), computers, electronic tablets, pens, rulers etc.

Instructions to students:
Attempt **ALL** questions.
All working must be done on your own supplied paper.
All working must be scanned and submitted to Gradescope before the end of the exam time.
The questions carry weight in proportion to the marks in brackets after the question numbers. These marks total **100 marks**.

While you are undertaking this assessment, you must not:

- make use of any messaging or communications technology;
- act in any manner that could be regarded as providing assistance to another student who is undertaking this assessment or will in the future be undertaking this assessment.

The work you submit MUST be based on your OWN knowledge and skills, without assistance from any other person.

Question 1 (18 marks)

Consider the following two fragments of C code, C_1 and C_2 shown in Figure 1:

```
...
x1=0;

while(1) {
  if (i1 == i2) {
    o1 = x1;
    x1 = 1 - x1;
  }
  else {
    o1 = 1 - x1;
    x1 = 0;
  }
}
...
```

(a) Code fragment C_1

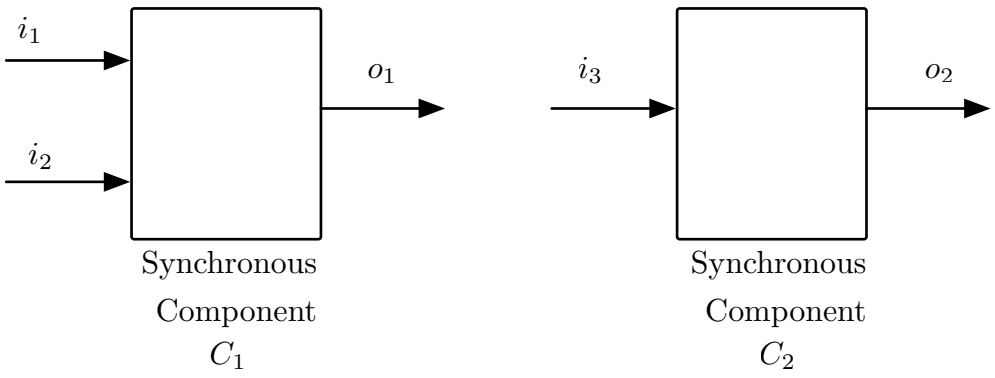
```
...
x2=0;

while(1) {
  x2 = (x2 + i3) mod 3;
  if (x2 == 2) {
    o2 = 1;
  }
  else {
    o2 = 0;
  }
}
...
```

(b) Code fragment C_2

Figure 1: C code fragments

where the integer-valued inputs `i1`, `i2`, `i3` can only take the values $\{0, 1\}$, `x1` and `x2` are integer-valued variables, and `o1` and `o2`, are integer-valued outputs that can only take the values $\{0, 1\}$.

The two fragments of C code above can be represented as two synchronous components, C_1 and C_2, with their block diagrams given below:



Question 1 (continued)

(a) **[4 marks]** Draw a Finite State Machine diagram to represent the behaviour of component C_1.

(b) **[4 marks]** Draw a Finite State Machine diagram to represent the behaviour of component C_2.

(c) **[6 marks]** Construct a Finite State Machine diagram representing the *cascade composition* of C_1 and C_2, where the output of C_1 is fed into the input of C_2, i.e., o_1 is connected to i_3. Which states of the composition are unreachable?

(d) **[2 marks]** What does the term *constructive* mean with respect to the composition of Finite State Machines? Why is constructiveness a useful property?

(e) **[2 marks]** Write down one *safety property* of the composed system from part (c) that is true, and one safety property that is violated by the composed system.

Question 2 (20 marks)

(a) [**2 marks**] Consider a delay component in an embedded system, `Delay`, that can be modelled as an infinitely repeating sequence of two assignment statements in C shown in Figure 2:

```
...
x=0;

while(1) {
  out=x;
  x=in;
}
...
```

Figure 2: Code fragment for `Delay`

The integer-valued variables `in` and `out` can only take the values $\{0, 1\}$. Variables taking such values will be referred to as *Boolean variables* for the rest of this question.

Draw a Finite State Machine diagram to represent the behaviour of this component.

(b) [**4 marks**] Consider a modified version of the `Delay` component, called `OddDelay`, that has a Boolean input variable `in`, a Boolean output variable `out` and two Boolean state variables `x` and `y`.

Both of the state variables are initialised to 0, and the C code modified to that shown in Figure 3, where the `!` operator performs Boolean inversion, i.e., `!0` is `1` and `!1` is `0`.

Question 2 (continued)

```
...
x=0;
y=0;

while(1) {
  if y {
    out = x;
  }
  else {
    out = 0;
  }
    x = in;
    y = !y;
}
...
```

Figure 3: Code fragment for `OddDelay`

Describe in words the behaviour of the component `OddDelay`. List a possible execution trace of the component when supplied with the sequence of inputs 0, 1, 1, 0, 1, 1 for the first six reactions.

(c) [**4 marks**] Describe the component `OddDelay` from part (b) as an Extended Finite State Machine with two states.

(d) [**4 marks**] Consider a counter component, `Counter`, which maintains a non-negative counter using a state variable `x`. The initial value of `x` is 0. The component has two Boolean input variables `inc` and `dec` and operates in the following way:

- when the input `inc` is 1, the counter state is incremented by 1; and
- when the input `dec` is 1, the counter state is decremented by 1.

The output of the component is the updated value of `x`. When both input variables `inc` and `dec` are 1, and when `dec` is 1 with the state `x` equal to 0, the component does not assign any value to the output, and as such there is no corresponding reaction.

The counter does not expect both input variables `inc` and `dec` to be 1 simultaneously, nor does it expect the state to be decremented when the counter value is 0.

Draw a Finite State Machine representing the behaviour of the `Counter` component.

Question 2 (continued)

(e) [**6 marks**] Design a component `CounterTest`, represented as a non-deterministic Finite State Machine, that supplies inputs to the `Counter` component from part (d).

The component `CounterTest` has no inputs, and its outputs are the Boolean variables `inc` and `dec`. It should produce all possible combinations of the outputs as long as the `Counter` component is willing to accept these as inputs: it should never set both `inc` and `dec` to 1 simultaneously, and it should ensure that the number of rounds with `dec` set to 1 never exceeds the number of rounds with `inc` set to 1.

Question 3 (8 marks)

Suppose that a program to compute the total sum of two vectors of floating point numbers of dimension n is executing on a 32-bit processor. For example, given the two vectors of floating point numbers, $[x_0, \ldots, x_{n-1}]$ and $[y_0, \ldots, y_{n-1}]$, the (scalar) total sum S would be calculated as

$$S = \sum_{i=0}^{n-1} x_i + \sum_{i=0}^{n-1} y_i$$

Suppose that the program uses a single `for` loop that, on each iteration, accesses individual elements of x and y to accumulate this sum. Further suppose that the vectors x and y are stored contiguously in memory starting with address 0 and that any caches are initially empty.

The processor has a direct mapped cache and the cache can hold two sets, each of which can hold four floats.

(a) **[2 marks]** What is the role of a *cache* in the memory hierarchy of an embedded system?

(b) **[2 marks]** How many cache misses will occur if $n = 2$? Show all of your working and list any assumptions you have made (if any).

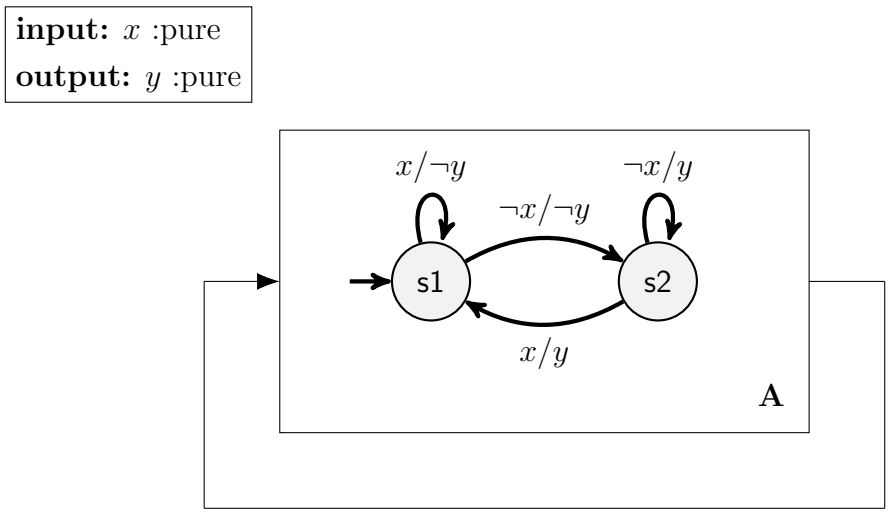
(c) **[2 marks]** How many cache misses will occur if $n = 8$? Show all of your working and list any assumptions you have made (if any).

(d) **[2 marks]** Assume that the vectors x and y are created using the C memory management function `malloc()`. Give one advantage and one disadvantage of this approach for embedded systems. Explain your answer.

Question 4 (14 marks)

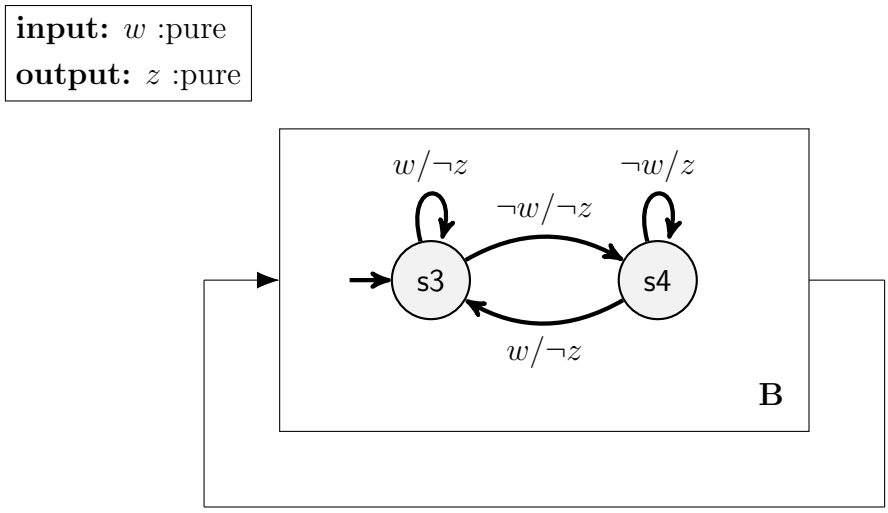
For all compositions in this question, you are to assume a Synchronous Reactive Model of Computation.

(a) **[2 marks]** Consider a system modelled by the following Finite State Machine (FSM), **A**:



Is the feedback composition is well-formed? Explain your answer.

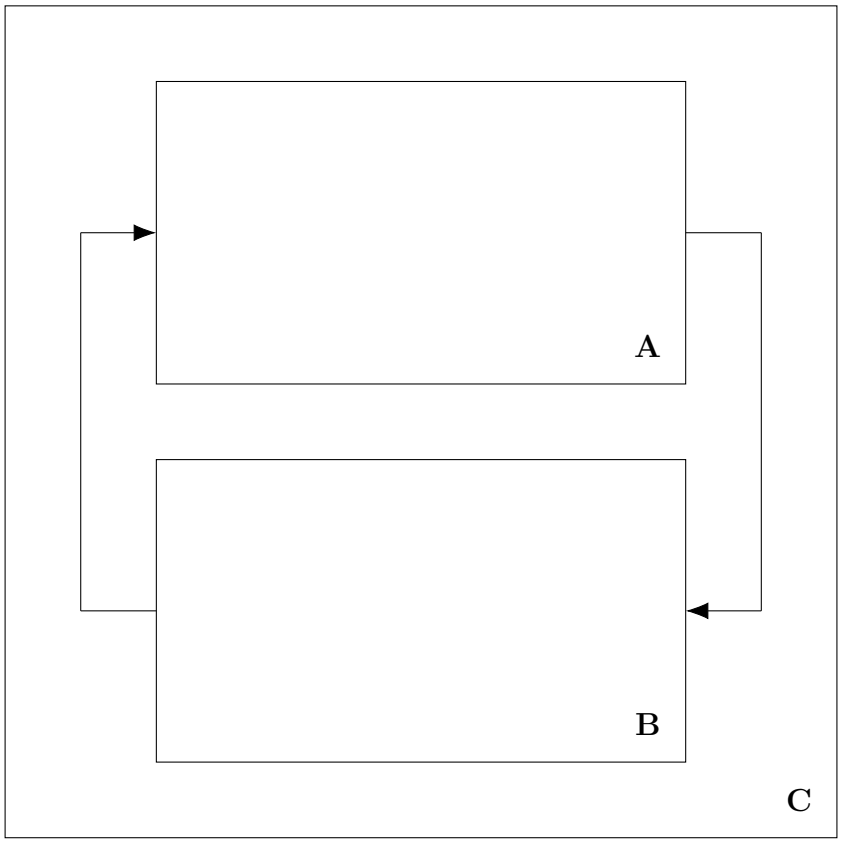
(b) **[2 marks]** Consider a system modelled by the following Finite State Machine (FSM), **B**:



Is the feedback composition is well-formed? Explain your answer.

(c) **[4 marks]** Consider the machine **A** combined with machine **B** in a feedback composition as shown below to form machine **C**. Note that the output port of **B** is drawn on the left and the input port on the right so that the block diagram looks neater.

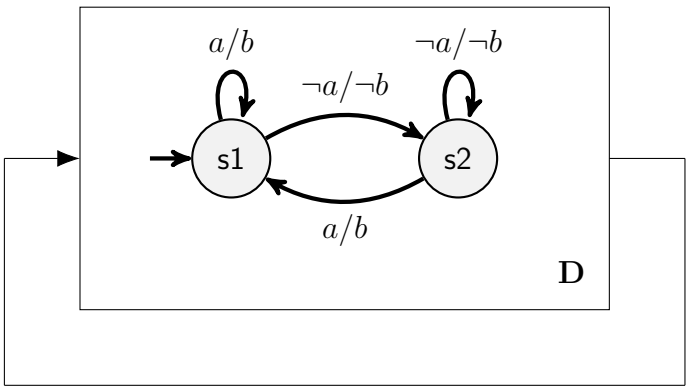
Question 4 (continued)



Show that feedback composition **C** is well-formed, and draw the Finite State Machine model for the composite machine.

(d) **[6 marks]** Consider the machine **D** below

input: a :pure
output: b :pure



Redraw this feedback composition as a non-deterministic state machine so that it is no longer ill-formed.

Give the set theoretic description for the composed machine, that is the 5-tuple:

$$(States, Inputs, Outputs, update, initialState)$$

Question 5 (18 marks)

Consider six tasks τ_1, ..., τ_6 with release times, execution times, and deadlines presented below.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6
release time, r_i	0	2	5	4	1	2
execution time, e_i	3	2	2	3	1	3
deadline, d_i	8	8	13	10	5	14

We say task τ_i precedes task τ_j if the predecessor must complete the execution of τ_i before τ_j can execute. The precedence relationships between tasks τ_1, ...,τ_6 are as the following

- τ_1 precedes τ_3;
- τ_1 precedes τ_4;
- τ_2 precedes τ_4;
- τ_2 precedes τ_5;
- τ_3 precedes τ_6;
- τ_4 precedes τ_6;
- τ_5 precedes τ_4.

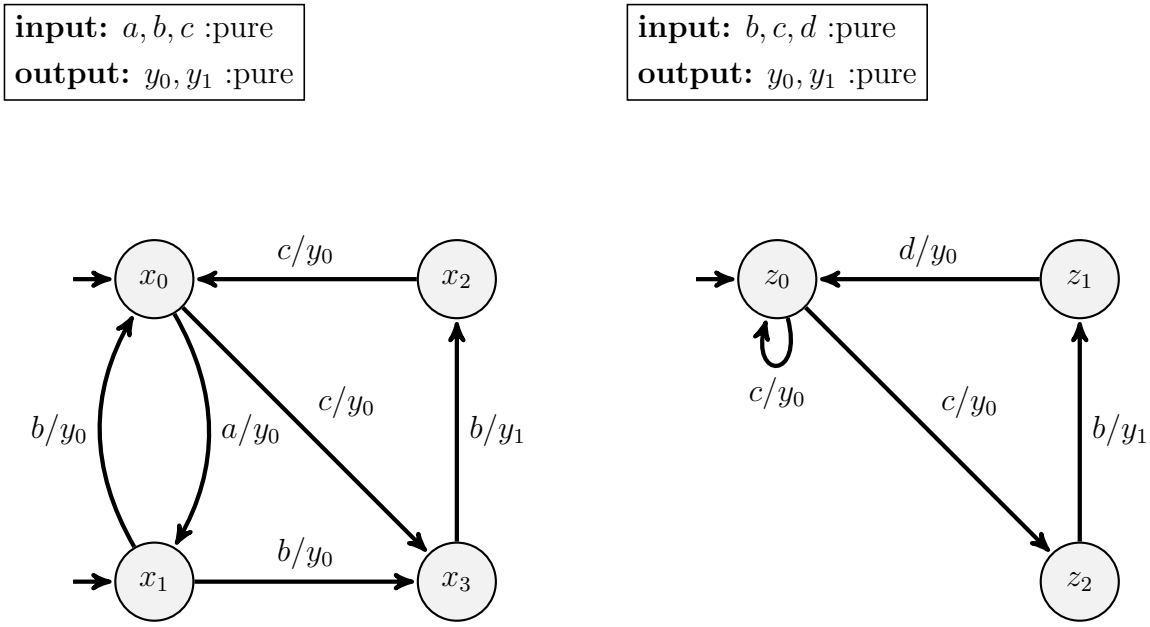
(a) **[4 marks]** Draw the directed acyclic graph representing the precedence of the tasks where a directed edge marks a precedence.

(b) **[4 marks]** Does a scheduling approach based on Latest Deadline First (LDF) yield a feasible deadline? Explain why if the answer is no, and construct the schedule if the answer is yes.

(c) **[10 marks]** Does Earliest Deadline First with Precedence (EDF*) yield a feasible schedule? Explain why if the answer is no, and construct the schedule if the answer is yes.

Question 6 (10 marks)

Consider Finite State Machines Σ_1 and Σ_2 depicted in Figure 4. Show that they are bisimilar.



(a) System Σ_1 (b) System Σ_2

Figure 4: Systems Σ_1 and Σ_2 (Question 6).

Question 7 (12 marks)

Figure 5 depicts the area of operation of a robot. Regions P_1 and P_2 represent two regions of interest, B denotes the base station, D represents a dangerous region, and R_1 and R_2 are two charging stations. Let logical proposition $x \in \{p_1, p_2, b, d, r_1, r_2\}$ correspond to the robot being in region $X \in \{P_1, P_2, B, D, R_1, R_2\}$.

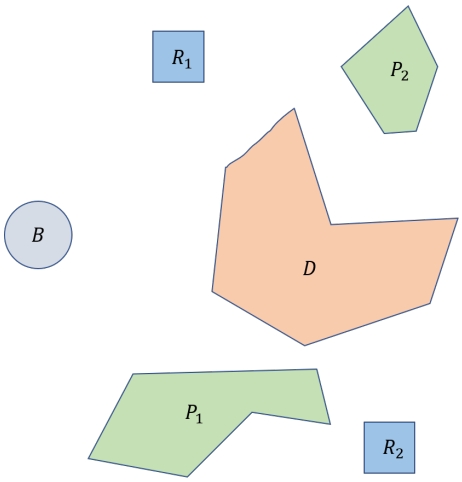


Figure 5: Area of operation of the robot of Question 7.

(a) **[3 marks]** What does the LTL formula ϕ as defined below mean? In addition to a natural language translation of the specification you need to interpret the specification.

$$\phi := \mathbf{GF}b \wedge \mathbf{GF}(p_1 \vee p_2) \wedge \mathbf{GF}(r_1 \vee r_2) \wedge \mathbf{G}\neg d$$

(b) **[5 marks]** A specification might be that the robot should visit either P_1 or P_2 and recharge between any two visits to the base, in addition to the requirement that it should never be in D and it needs to infinitely often visit the base station. Let ψ be the corresponding LTL formula to this specification. What is ψ?

(c) **[4 marks]** Assume that ψ from part (b) holds for a robot's behaviour. Does the robot satisfy $\mathbf{GF}(r_1 \vee r_2)$? Explain.

END OF EXAMINATION