Digital Systems
Final examination. June 15, 2021

Time limit: 120 minutes.

1 A Tumble dryer (40%)

Consider a tumble dryer with the following simplified operation. When this tumble dryer is powered on, it waits for a user to make a coin payment. When the coin payment is validated, the drum of the machine starts moving to the right, a fan moves air into the drum through a heater and the heater is turned on. The fun and the drum never stop moving, although each minute the drum changes the direction of rotation. The heater is turned off when the temperature of the air is higher than 80ºC and it is turned on again when the temperature is lower than 70ºC. After 10 minutes of operation the machine stops and a new payment can be made.

In order to synthesize a hardware that controls the operation of the machine, read the next explanation, which explicitly consider input and output signals.

- When the machine is powered on, it waits for a coin payment.The machine can not start its operation until a coin payment of 1 euro is made. When a valid coin payment is made, the coin validator generates the signal `s_coin` that is high during one `clock`. This payment allows the machine to operate during 10 minutes. No more coin payments are permitted until the machine finishes its operation.
- The temperature is read from the signal `s_temp` that is coded as an unsigned number that can take values in the range 0-100ºC. You can decide the length of this signal.
- The fan is turned on when the signal `s_fan` is high.
- The heater is turned on when the signal `s_heater` is high.
- The drum moves when the signal `s_drum` is high. It moves to the right when the signal `s_right` is high and to the left otherwise.

a) Draw the graph of a synchronous (clock-driven) FSM in which each output depends only on the state. I suggest to use a single FSM, although it is tempting to use independent FSM to control the heater and the drum. Use two counters driven by a 1Hz clock: one to measure 1 minute and the other 10 minutes.

b) Describe the previous FSM in `VHDL` code. Help yourself to the following `VHDL` sketch. Use one `process` statement to describe the sequential part, and one or two more `process` statements, as you like, to describe the combinational part.

```
library ieee;
use ieee.std_logic_1164.all, ieee.numeric_std.all;

entity fsm is
  port( clk      : in std_logic;
        ...      : in ...);
        ...
        ...      : out std_logic);
end;
```

```
architecture arch of fsm is
  type multiplier_state is (...);
  signal current_state : multiplier_state := ...;
  signal next_state : multiplier_state;
begin
  process(clk) is
  begin
  ...
  end process;
  ...
```

2 Registering or not registering (20 %)

a) Describe the block diagram of *Figure* 1 in `VHDL`. Be careful when describing the different combinational and sequential parts.

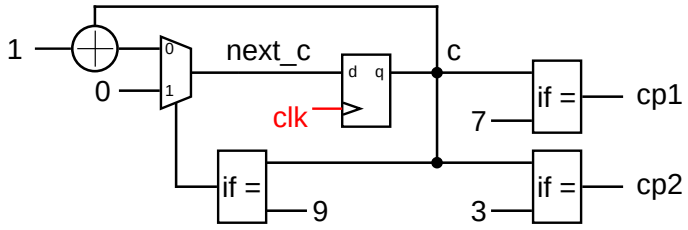

Figure 1: Block diagram of design 1. The signal `c` can take integer values from 0 to 15. The signals `cp1` and `cp2` can take logical values '0' or '1'. All numbers in the figure are coded in decimal. The block with the `clk` signal is a register.

b) Draw a waveform with the first 10 `clocks` after *reseting* `c` to zero [1]. Draw the value of:

- `c`
- `next_c`
- `cp1`
- `cp2`

c) Next describe the block diagram of *Figure* 2 in `VHDL`, in such a way that the waveforms of the signals `c`, `cp1` and `cp2` of both designs are the same [2].

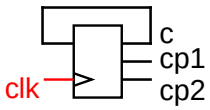

Figure 2: Block diagram of design 2. The block with the `clk` signal must be described with a single `process(clk)`.

d) Even if both designs have the same functionality, point out any difference that can make one design more suitable than the other in terms of, for example, the use of resources in an FPGA.

[1] You don't have to consider the `reset` signal in the previous or next sections.
[2] Consider that the delay of the combinational blocks are insignificant.

3 Mini AVR: Branch (15 %)

The `Mini AVR` architecture of *Figure* 3 has in its ROM the following code:

```
0.   LDI r20,xFF
1.   LDI r21,x02
2.   ADC r20,r21
3.   BREQ +2
4.   LDI r20,xFD
5.   RJMP -4
6.   RJMP -1
```

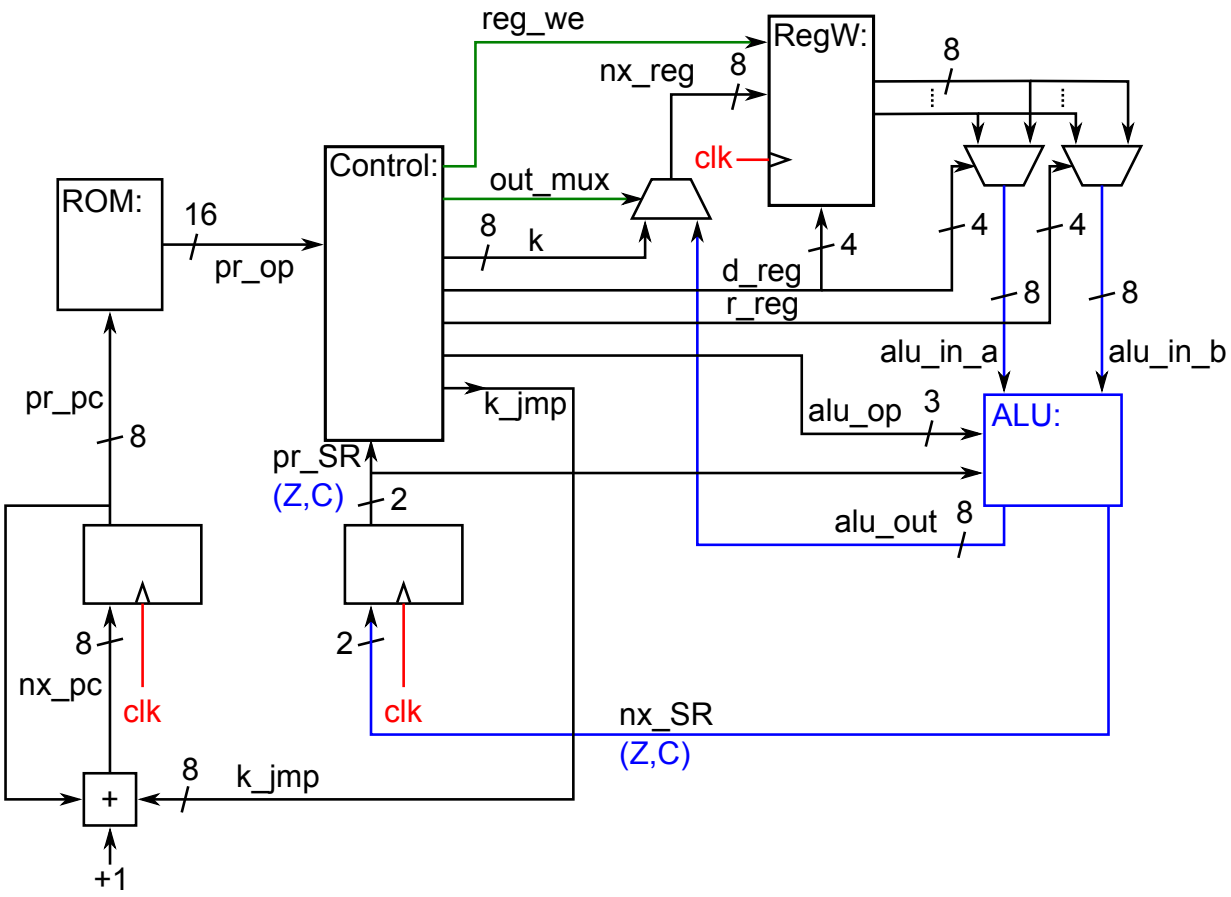

Figure 3: `Mini AVR` architecture with relative jumps.

a) Draw a waveform with the first 10 `clocks` after a general `reset` [3]. Draw the value of:

- `program counter: pr_pc`
- `r20` register
- `C` flag
- `Z` flag

b) Describe the value of all the significant signals in *Figure* 3 just before the rising edge of the seventh `clock` of the previous waveform.

[3] This general `reset` is not shown in *Figure* 3.

4 Mini AVR: RAM (15 %)

In the last class about the `Mini AVR` we added 1kB of RAM. To do this we introduced the concept of *indirect addressing*.

a) Explain how this concept is implemented in the `Mini AVR`.

b) Compare with some other solutions.

c) Justify the need of the solutions that have explained in the two previous questions in terms of the length of the `opcode` [4] and the size of the RAM.

5 The importance of the sensitivity list (10 %)

Write a simple `process` to explain the importance of the sensitivity list. Consider different sensitivity lists that for a given set of inputs generate different output or outputs. Draw an illustrative waveform.

[4] Six bits of the `opcode` are used to indicate the operation.