Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Embedded Systems
Final exam. June 6, 2018

Duration: 2.5 hours. Last revision day: June 27

1 AVR and C: optimization options

1. Comment on the various sorts of optimization that you can use with the compiler `avr-gcc`.
2. What optimization option have you used when compiling the DTMF project? Have you used the right one?

2 AVR and C: bad code that works

The following code seems to work pretty well: it sends an incremented byte through the serial port every N timer0 interruption calls.

```
#define N 100.0

int n=0;
char c='A';

int main(void){
  setup();
  while(1){
  }
  return 0;
}

ISR(TIMER0_COMPA_vect){
  n=n+1;
  if (n==N){
    serial_put(c++);
    n=0;
  }
}
```

1. Enumerate some good practices that help to understand a code.
2. Enumerate some good practices that help to speed up a `C` code in a microcontroller environment.
3. Put in practice these good practices and modify the previous code to obtain a good code that works.

3 FPGA to AVR parallel communication

During the DTMF project, we implemented a parallel communication to send each one of the 16 possible keys of a DTMF keypad from the FPGA to an AVR. Four lines were used to code the key and an additional line was used to *enable* at the rising-edge the reading of the key. Consider the following `C` code that reproduces the parallel communication at the AVR end, but at a lower scale: two lines to code just four keys.

```
# include <avr/io.h>
# include <stdbool.h>
# include "serial.h"

int main(void) {

  static char values[4] = {'A','S','D','F'};
  static bool state = false;

  DDRD &= ~(1<<DDD6); //PIND6 as input, rising edge ==> read PIND7 and PINB1
  DDRD &= ~(1<<DDD7); //PIND7 as input
  DDRB &= ~(1<<DDB1); //PINB1 as input
  serial_init();
  sei();

  while(1) {
    if (PIND & (1<<DDD6)) {
      if (state == false) {
          uint8_t in0 = (bool)(PIND & (1<<DDD7));
          uint8_t in1 = (bool)(PINB & (1<<DDB1));
          char c = values[(in1 << 1) | (in0)];
          serial_put(c);
          state = true;
      }
    }
    else {
      state = false;
    }
   }
   return 0;
}
```

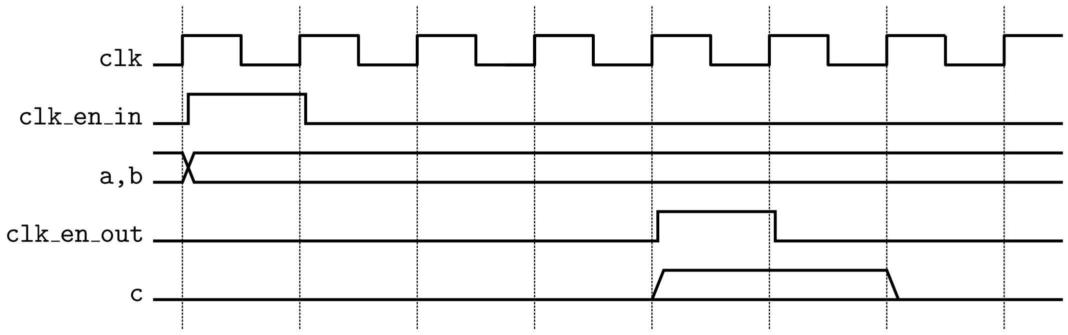
Note that this code uses to explicit cast to `bool` and two implicit cast to `uint8_t`.

1. What character is sent to the serial port for each combination of (PIND7, PINB1)?
2. If we remove `(bool)` from the code, i.e. no explicit cast is made, what character is sent to the serial port for each combination of (PIND7, PINB1)?
3. After knowing that in `C` a number equal to zero is `false` and any other number is `true` we are not sure about the final value of `in0` and `in1` after the explicit and implicit casts. Write an alternative code that guarantees that it will always work well.

4 FPGA and VHDL: minimizing the number of embedded multipliers with a FSM

A signal called `clk_en_in` is high the first rising edge of `clk` after the inputs `a,b` are ready and is low the next rising edge. We want to design an entity in order to compute `c=a*a+b*b+a*b`. We have four `clk` periods to compute the output `c` with the constraint of minimizing the number of embedded multipliers that are used: 132 9-bit multipliers are available in the DE0-Nano/Altera Cyclone IV EP4CE22F17C6N FPGA. A signal called `clk_en_out` must be high the first rising edge of `clk` after `c` is ready and must be low the next rising edge. An additional constraint is that `c` must hold the value only during two `clk` periods. At the other `clk` periods it must be zero. The next waveform shows the previous explanation.

Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA



1. Use the following sketch to write the VHDL code that describes a right design.

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

entity se_exam_2018 is
  port (clk        :  in  std_logic;--10MHz
          clk_en_in  :  in  std_logic;--1MHz
          a       : in  std_logic_vector(8 downto 0);
          b       : in  std_logic_vector(8 downto 0);
          clk_en_out: out std_logic;
          c   : out std_logic_vector(8 downto 0)--c=a*a+b*b+a*b
          );
end entity;

architecture arch_1 of se_exam_2018 is
  signal state : unsigned(2 downto 0):= to_unsigned(0,3);
  --others signals
begin
  --concurrent statements
  process(clk)
  begin
    if rising_edge(clk) then
      if clk_en_in = '1' then
        --do something
      else
        case state is
          when 1 =>
            --do something
          when 2 =>
            --do something
            .
            .
          when others =>
            --do something
        end case;
      end if;
    end if;
  end process;
end architecture;
```

2. Try to compute `c` in less `clk` periods without increasing the number of multipliers.

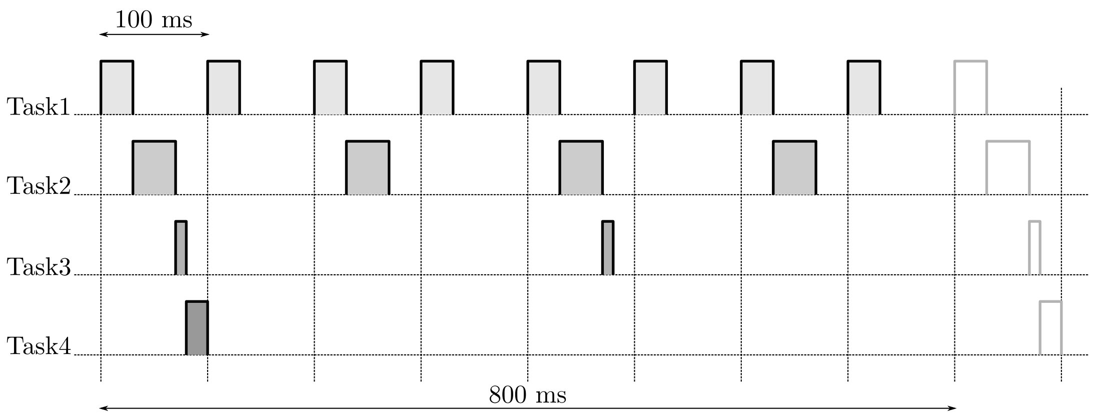
3. Explain the differences in hardware utilization, delay in computation, and maximum frequency of `clk` between a design using

```
c <=a*a+b*b+a*b;
```

and the design of the previous question that uses a FSM.

5 Dealing with schedulers

Consider a Task1 that is executed every 100 ms and lasts 30 ms, a Task2 that is executed every 200 ms and lasts 40 ms, a Task3 that is executed every 400 ms and lasts 10 ms, and, finally, a Task4 that is executed every 800 ms and lasts 20 ms. All tasks start at $t = 0$. The following figure shows the slot of time used by each task.



You can think that these tasks are executed thanks to an interruption called every 100 ms. The time that these tasks are not executed is used by a main process that implements a DTMF decoder in the same way that you have implemented it when using a Raspberry Pi: 8-bit samples are received through a serial connection at 8000 S/s and each window made of 200 samples is processed in 15 ms. To ease the future work you can consider that the process of each window starts reading 200 samples in a negligible time. Consider also that a single buffer of length 4095 B is used to store the received samples in a negligible time.

1. Argue why the DTMF decoder will lose samples.
2. Determine with a resolution of µs at which time instant the buffer will be full for the first time, considering that at $t = 0$ the buffer is empty.
3. What percentage of the samples will be lost when the system is in steady state?
4. Remove the lowest number of tasks of the lowest priority from the scheduler in order to avoid the loss of samples.

6 DTMF project course

1. Build a believable table in which you evaluate development time, hardware cost, power consumption and robustness of each one of the three platforms during the DTMF project: AVR, FPGA and Raspberry Pi.