UNIVERSITY OF OSLO

Faculty of Mathematics and Natural Sciences

Exam in: **FYS4220**
Day of exam: **16 December 2014**
Exam hours: **09.00 – 13.00**
This examination paper consists of 5 pages.
Appendices: **none**
Permitted materials: **none**

Make sure that your copy of this examination paper is complete before answering.

Read the text carefully before you start. All questions are weighted equally within a problem unless otherwise stated.

Problem 1 (25%):

a) (5%) What is the purpose of the sensitivity list in a VHDL process statement?

Answer P1 a)
The sensitivity list is used to specify which signal should cause the process to re-evaluate (activate, trigger). Whenever any event occurs on one of the signals in the sensitivity list, the process is re-evaluated.

b) (5%)Write the VHDL code for the 4-bit tristate-buffer shown in Figure 1. The enable signal ***ENA_n*** is active low.

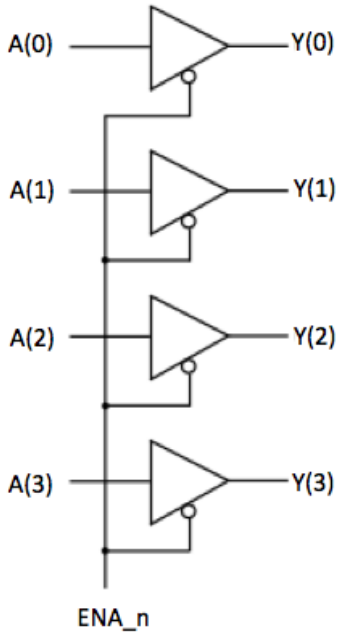


Figure 1

Answer P1 b)
Multiple solution possible. Two options shown below.

```
Y <= A when ENA_n = '0' else (others => 'z' );


process(ENA_n, A)
begin
  if ENA_n = '0' then
    Y <= A;
  else
    Y <= (others => 'Z');
  end if;
end process;
```

c) (5%) The combinational VHDL process in **Error! Reference source not found.** describes the behaviour of a 3-input multiplexer with two select pins. The VHDL code is incomplete as shown in the figure. Which modifications are needed to correctly complete the code? Explain your answer.

```
process( )
  if sel = "00" then
    Y <= A;
  elsif sel = "01" then
    Y <= B;
  elsif sel = "10" then
    Y <= C;
  end if;
end process;
```

Figure 2

Answer P1 c)

```
process(sel,A,B,C)
begin
  if sel = "00" then
    Y <= A;
  elsif sel = "01" then
    Y <= B;
  elsif sel = "10" then
    Y <= C;
  else
    Y <= A;
  end if;
end process;
```

- Signals sel,A,B,C needs to be added to the sensitivity list as this is a combinational process and thus all signals which are read in the process should be included in the sensitivity list.
- An else statement needs to be added to be added in order to avoid an unintentional latch. The assignment under the else statement is not specified and any of the options Y<= {A,B,C,'0' } is accepted

d) (10 %) Figure 3 shows a state diagram of a Moore type state machine with an input signal ***A***, an output signal ***Z***, and an asynchronous active low reset signal ***areset_n***. Write the corresponding VHDL architecture construct to implement this state machine. You do not need to write the entity construct.

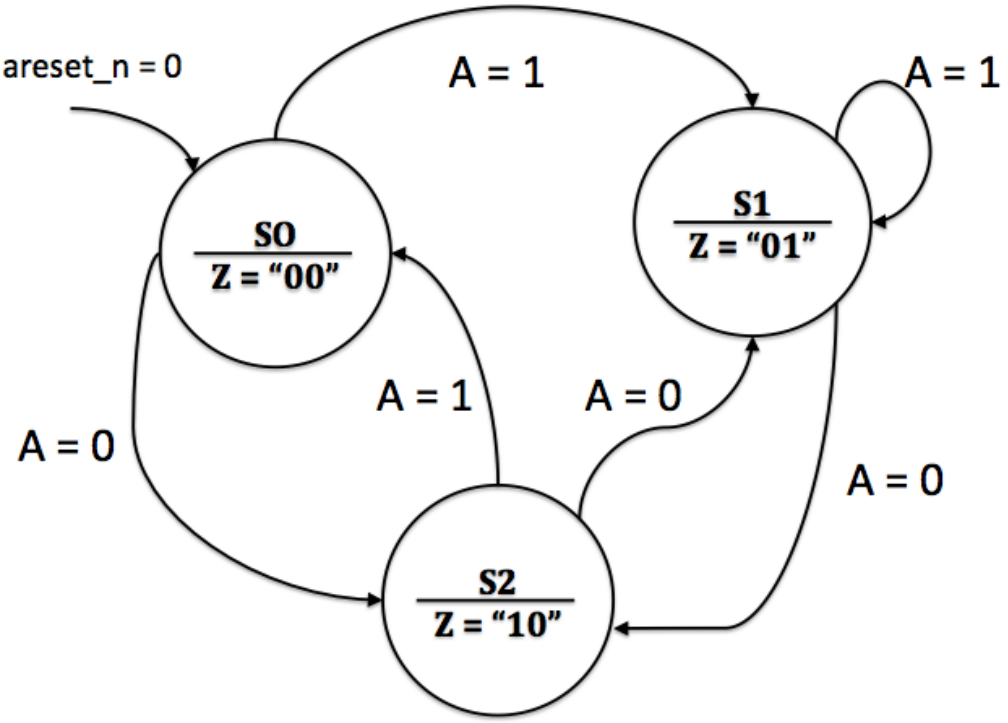


Figure 3

Answer P1 d)

```
architecture statemachine of problem1d is

  --Can also assume that areset_n, A and Z are
  --declared in the entity declaration.
  signal areset_n : std_logic;
  signal A        : std_logic;
  signal Z        : std_logic_vector(1 downto 0);

  type statetype is (S0, S1, S2);
  signal state : statetype;


begin

  process(clk, areset_n)
  begin
    if areset_n = '0' then
      state <= S0;
    elsif rising_edge(clk) then
      case state is
        when S0 =>
          Z <= "00";
          if A = '0' then
            state <= S2;
          else
            state <= S1;
          end if;

        when S1 =>
          Z <= "01";
          if A = '0' then
            state <= S2;
          end if;

        when S2 =>
          Z <= "10";
          if A = '0' then
            state <= S1;
          else
            state <= S0;
          end if;

      end case;
    end if;
  end process;
end architecture;
```

Problem 2 (25%):

Figure 4 shows the top-level entity for a VHDL module that can be used to measure the speed of a passing object. The asynchronous input signals ***start_n*** and ***stop_n*** are connected to two optical sensors and used to respectively start and stop a 32-bit counter. The counter has a synchronous active low reset signal ***reset_n***. The optical sensor gives out a high signal value ('1') when exposed to light, and a low signal value ('0') when a passing object blocks the light. Your task is to implement the VHDL description for this module. The expected functional behaviour of the module is shown in Figure 5.

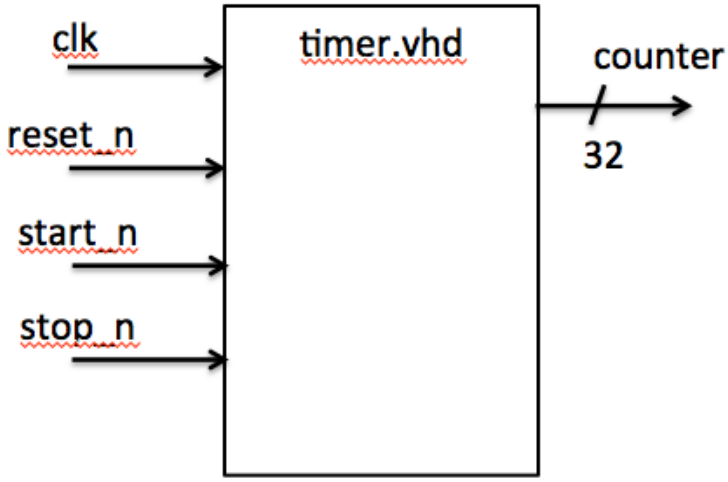


Figure 4

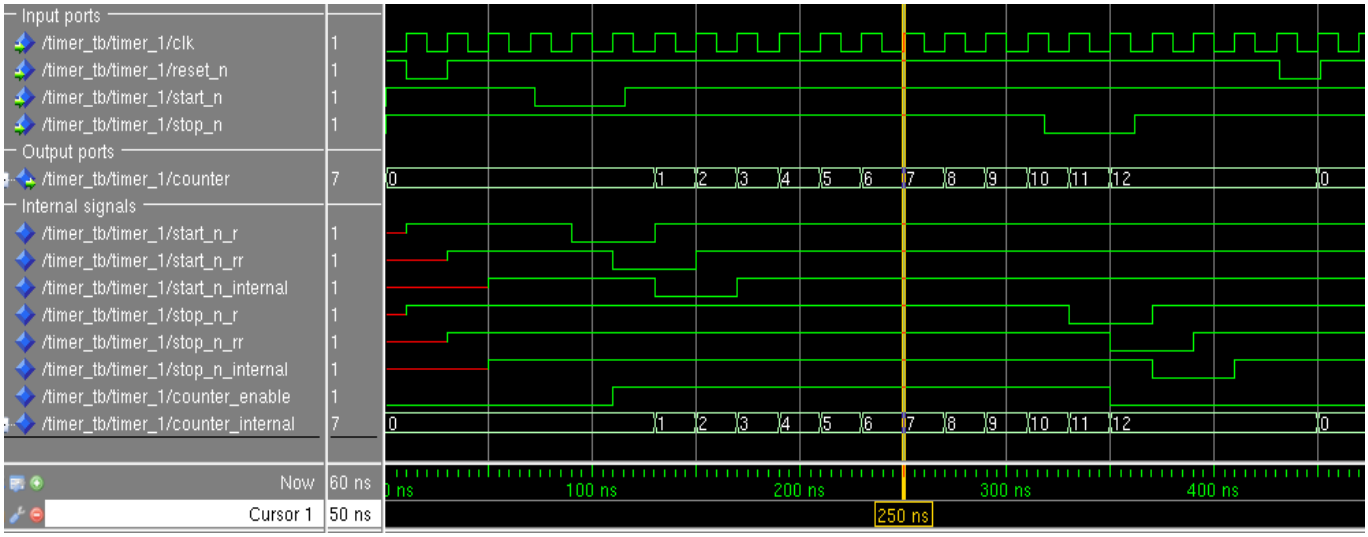


Figure 5

The module must have the following internal signals:

- *signal counter_internal : unsigned(31 downto 0)*
- *signal counter_enable : std_logic*
- *signal start_n_r : std_logic*
- *signal start_n_rr : std_logic*
- *signal stop_n_r : std_logic*
- *signal stop_n_rr : std_logic*
- *signal start_n_internal : std_logic*
- *signal stop_n_internal : std_logic*

a) Write the VHDL code for the ***entity*** of the module ***timer*** shown in Figure 4.

Answer P2 a)

```
entity timer is
  port (
    reset_n : in  std_logic;
    clk     : in  std_logic;
    start_n : in  std_logic;
    stop_n  : in  std_logic;
    counter : out std_logic_vector(31 downto 0));
end timer;
```

b) The input signals ***start_n*** and ***stop_n*** are asynchronous and therefore need to be synchronized to the system clock by routing them through two registers. Write the process statement that synchronizes these signals with the clock signal ***clk***. Use the internal signals with the extensions ***_r*** and ***_rr*** to make the synchronization registers.

Answer P2 b)

```
proc_1 : process(clk)
begin
  if rising_edge(clk) then
    stop_n_r   <= stop_n;
    stop_n_rr  <= stop_n_r;
    start_n_r  <= start_n;
    start_n_rr <= start_n_r;
  end if;
end process;
```

c) It has been decided that the counter will be activated and deactivated when a falling edge is detected on the synchronized input signals ***start_n_rr*** and ***stop_n_rr*** respectively. Write the VHDL code that makes use of the signals ***start_n_rr, start_n_internal, stop_n_rr, stop_n_internal*** to control the ***counter_enable*** signal.

Answer P2 c)

```
proc_3 : process(clk)
begin
  if rising_edge(clk) then
    if reset_n = '0' then
      counter_enable <= '0';
    else
      start_n_internal <= start_n_rr;
      stop_n_internal  <= stop_n_rr;

      if start_n_internal = '1' and start_n_rr = '0' then
        counter_enable <= '1';
      elsif stop_n_internal = '1' and stop_n_rr = '0' then
        counter_enable <= '0';
      end if;
    end if;
  end if;
end process;
```

d) Write the VHDL process statement for the 32-bit counter with a synchronous active low reset signal. The counter is activated when the enable signal ***counter_enable*** is high (‘1’).

Answer P2 d)

```
proc_2 : process(clk)
begin
  if rising_edge(clk) then
    if reset_n = '0' then
      counter_internal <= (others => '0');
    elsif counter_enable = '1' then
      counter_internal <= counter_internal + 1;
    end if;
  end if;
end process;

counter <= std_logic_vector(counter_internal);
```

e) What is the frequency of the clock signal ***clk***, and how can you calculate the maximum time value this module can measure?

Answer P2 e)

- From Figure 5 you can count 5 clk cycles between 100 ns and 200 ns. The clock frequency is then 1 / (100ns/5) = 50 MHz
- With a 32-bit counter one can measure a maximum time of 2^32 * 20 ns = ~ 85 s

f) What is meant by *Static Timing Analysis (STA),* and why is it important?

Answer P2 f)

- STA is a method of computing the expected timing of a digital circuit without requiring simulation. It considers timing of paths from e.g. register to register, input port to register, register to output port, purely combinational paths.
- Performing STA analysis is important in order to make sure that the implemented design will meet the given timing requirements. (E.g. by providing information about the system clock frequency, the STA will e.g. calculate and identify the most critical path.)

Problem 3 (20%):

a) Can you mention at least two advantages of a soft core processor compared to a hard core processor?

Answer P3 a)

Some advantages:

- Include processor core only when needed.
- Include only necessary features of the soft core processor system
- Can implement multiple processor cores if needed
- Can easily reuse processor core system for newer FPGAs

Some disadvantages:

- May be slower than a hard core processor
- Less area efficient compared to hard core processor

Other advantages and disadvantage suggested by the candidate may also be accepted.

b) The Nios-II architecture provides memory mapped I/O access through the Avalon interface. Can you briefly explain what memory mapped I/O access is?

Answer P3 b)
Memory mapped I/O access means that all modules/peripherals connected to the Avalon bus are mapped into the address space of the master (cpu). These modules can thus be accessed by writing and reading data to the respective address location in the master address space.

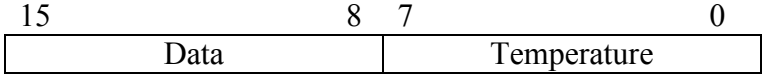
c) What is the purpose of the Board Support Package (BSP) when developing a Nios-II software project?

Answer P3 c)
- A BSP provides a software runtime environment customized for the processor system defined in the Qsys builder system.
- The NIOSII BSP is a specialized library containing system-specific support code. E.g. it contains the Hardware Abstraction Layer (HAL) that allows a computer operating system to interact with a hardware device at a general or abstract level rather than at a detailed hardware level. The BSP also contains device drivers, the Newlib C standard library, and an optional real-time operating system.

d) What is the difference between interrupt-driven and polled-based I/O strategies?

Answer P3 d)
- For a polling based I/O strategy the master requests/reads/checks the content of a register or module at regular intervals. That is, the master controls when to read information.
- For an interrupt based I/O strategy, the master only reads/checks the value of a register or module when it receives a signal (interrupt, event) informing the master that information/data is available.

e) A function *alt_u16 read_data(void)* is used to read data from a sensor over a serial interface. It returns a 16-bit unsigned value where the 8 least significant bits hold the temperature of the sensor, and the 8 most significant bits hold an 8-bit address value as shown below.



The temperature is stored as a signed two's complement value. The resolution of the

temperature sensor is 0.25°C / LSB. Write another c-function to print the address value and temperature value to the standard output device. The address value should be printed as a hexadecimal value and the temperature as a float value.

Answer P3 e)

```
void print_data(void)
{
  alt_u16 result = read_data();
  printf("results 0x%x\n",result);
  //filter 8 LSB and convert to signed 8 bit type alt_8
  alt_8 temp_signed = (alt_8)(result & 0xff);
  //convert to float and multiply by resolution of 0.25C/LSB
  float temp_float = ((float)temp_signed)*0.25;

  ////shift address bits from 8 MSB to 8 LSB. Filter the 8 LSB
  ////and convert to alt_u8. The latter convertion is strictly not necessary
  ////as the value will be printed as hexadecimal and it is not a signed value.
  alt_u8 address = (alt_u8)((result >> 8) & 0xff);

  ////print the respective data
  printf("Temperature: %f\n",temp_float);
  printf("Address    : 0x%x\n",address);
}
```

f) Assuming that the temperature sensor has no offset and its range is symmetric around 0°C, what is the possible temperature range of the sensor? Explain your answer.

Answer P3 f)

- A total of 8 bits is used for the temperature sensor. With 1 bit defining positive or negative value this leaves 7 bits to contain the value. The range is then from -128 (-32C) to 127 (+31.25C). Values close to this is accepted as long as it is followed by an explanation.

Problem 4 (30%):

a) Briefly explain what a real-time operating system is.

Answer P4 a)

- A real-time operating system is specially designed to run applications with very precise timing and high degree of reliability.
- The correctness of the system depends not only on the logical results of computation, but also on the time at which the results are generated.
-

b) Is a real-time operating system expected to be mainly deterministic or non-deterministic? Explain your answer.

Answer P4 b)

- Deterministic
- A real-time system is expected to have a well defined time for how long it takes to perform a task and thus when the results of the task will be delivered.
- In other words, it does what you expect it to do, when you expect that it will do this.

c) What does it mean that a real-time operating system is a priority-based pre-emptive system?

Answer P4 c)

- This means that the scheduler of the RTOS ensures that at a given time the processor executes the highest priority task of all task which are currently ready to be executed.
- That is, if a task of a higher priority than the currently running task is ready to be executed, the scheduler interrupts the currently running tasks and the makes the processor execute the higher priority task. When the higher priority task is completed, the execution of the lower priority task is resumed.

d) Can you briefly explain the term jitter and why it is an important performance measure for a real-time operating system?

Answer P4 d)

- Jitter is the amount of error in the timing of a task over subsequent iterations of a program or loop.
- Jitter is an important performance measure since it gives information about the punctuality/accuracy of the system. Low jitter means that a task will take very close to the same amount of time to execute each time it is run.

e) What is a context switch and why is this an essential feature of a real-time operating system?

Answer P4 e)

- A context switch takes place when the system suspends the currently running task in order to execute another higher priority task. In this case the RTOS must save all the information needed to eventually resume the task being suspended.
- This is an essential part of an RTOS since is allows a system to multitask. The system operation can be divided into several smaller tasks which then are executed at a given time controlled by a privileged task or part of the system known as the scheduler.

f) What is the main difference between a mailbox and a message queue in uC/OS-II?

Answer P4 f)
A mailbox can contain one message only while a message queue can contain several messages.

g) Figure 6 shows an example of a task that prints a message to the console (i.e. standard output). How often will the message be printed?

```
void task1(void* pdata)
{
  while (1)
  {
    printf("Hello from task1\n");
    OSTimeDlyHMSM(0, 0, 0, 5);
  }
}
```

Figure 6

Answer P4 g)
The message will be printed every 5 ms on average.

h) It has been decided to add another task, ***task2,*** to the system. This new task will be used to control how often ***task1*** will be executed. A semaphore will be used to synchronize the two tasks so that the message of ***task1*** is printed every second. Make the needed modifications to ***task1*** and write the c-code for the new function ***task2***. You can assume that the semaphore ***sync_sem*** already has been declared and created/initialized.

Answer P4 h)

```
void task1(void* pdata)
{
  INT8U error_code = OS_NO_ERR;
  while (1)
  {
    OSSemPend(sync_sem,0,&error_code);
    printf("Hello from task1\n");
  }
}

void task2(void* pdata)
{
  while (1)
  {
    OSSemPost(sync_sem);
    OSTimeDlyHMSM(0, 0, 1, 0);
  }
}
```