



Academic Year: 2007 - 2008
Examination Period: Summer
Module Leader: Nigel Gunton
New Module No: UFEEHH-30-2
Old Module No:
Title of Module: Architecture of CPUs & VHDL
Examination Date:
Examination Start time: 09:00
Duration of Examination: 3 hours Hour(s) 00 Minutes

Instructions to Students: Attempt Question One and any 1 other question.

Materials supplied to the student will be:

Number of Examination Booklets (+ any continuation booklets as required) per Examination	1
Number of Pre-printed OMR (Multiple Choice Answer Sheet)	0
Number of sheets of Graph Paper size G3 (Normal)	0

Additional Instructions to Invigilators:

Calculators may be used subject to University regulations	Yes
Students allowed to keep Examination Question Paper	No
Material supplied by student allowed :	None
Additional Specialised Material:	

Treasury tags & adhesive triangles will be supplied as standard

ANSWERS

ANSWERS

ANSWERS

This page blank

ANSWERS

ANSWERS

1.a) 10

See attached diagram. Expect blocks for the timers and the controller, labelled control signals. Should show inputs from buttons and outputs to the sets of lights as well. Explanation of the component parts. Give credit for an overall description of the system.

1.b) 25

See diagram in appendix. Answers must show state_name at top left of state box, output signals listed in state box.

Decision boxes must follow state boxes and contain the name of the input signal within the box. Branches must clearly indicate decision value for each branch.

Conditional output boxes, where used, must follow decision boxes and show the conditional output value.

Basic correct diagram, poor labelling, some of which matches entity ≈ 10

Correct diagram, most labels correct & match entity ≈ 18

All correct and matching, clear & well laid out ≈ 25

1.c) 10

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity traffic_controller is
port (
    clk, reset, vss, vdd      : in std_logic;
    button                    : in std_logic;
    done_a                    : in std_logic;
    done_5                    : in std_logic;
    done_3                    : in std_logic;
    done_mf                   : in std_logic;
    done_f_fl                 : in std_logic;
    start_a                   : out std_logic;
    start_5                   : out std_logic;
    start_3                   : out std_logic;
    start_mf                  : out std_logic;
    start_f_fl                : out std_logic;
    traffic_lights            : out std_logic_vector(5 downto 0)
);
end traffic_controller;

-- traffic_lights are MR MA MG FR FA FG

```

expect correct syntax, use of std_logic, power, reset and clock inputs, meaningful names.

1.c) 15

Expect a two process state machine design providing code fragments similar to the example below. Other design approaches should be considered on their own merits. Expect justification based on clear mapping from ASM to a two process state machine in VHDL, ease of verification from design to code, speed and clarity of coding style, permits synthesis tool greater flexibility in synthesis for a given target architecture, other plausible points (up to 5 marks) . 10 marks for the coding; Deduct marks for the following errors or omissions in the architecture.

Failure to declare state_type. -2

Missing signals in the sensitivity lists for the processes. -1 per missing signal

Failure to initialise the state machine correctly. -2

Failure to ensure default state assignment in each state or missing 'when others' clause, -2

This could be implemented as a hardwired controller or microsequencer which would be more difficult to code from the asm but has other advantages.

```
architecture asm of treaffic_controller is
  type state_type is(R_R, G_R, MGF_0, MGF_1, MA, RR_2, R_G, FGF_0, FGF_1, R_A);

  -- Pragma CLOCK clk
  -- Pragma CURRENT_STATE current_state
  -- Pragma NEXT_STATE next_state

  -- This design assumes a 0.5hz clock

  signal current_state, next_state : state_type;
begin
  -- purpose: state comb
  -- type : combinational
  -- inputs : button, all the done_*, init
  -- outputs: lights, start_*
  comb: process (current_state, init, button, done_a,done_5,done_mf,
                done_f_fl,done_3)
  begin -- process comb
    if init = '0' then -- initialise the state machine
      next_state <= R_R;
      lights <= "100100";
      start_a <= '1';
    else
      case current_state is

        when R_R => lights <= "100100"; -- red/red
                   if done_a = '1' then
                     next_state <= G_R;
                     start_5 <= '1';
                   else
                     next_state <= R_R;
                   end if;
      end if;
    end if;
  end process;
end;
```

```
when G_R => lights <= "001100" -- green/red
    if done_5 = '1' and button = '1' then
        next_state <= MGF_0;
        start_m_fl <= '1';
    else
        next_state <= G_R;
    end if;

when MGF_0 => lights <= "000100";
    next_state <= MGF_1;

when MGF_1 => lights <= "001100"
    if done_mf = '1' then
        next_state <= MA ;
        start_a <= '1';
    else
        next_state <= MGF_0;
    end if;

when MA => lights <= "010100";
    if done_a = '1' then
        next_state <= RR_2;
        start_a <= '1';
    else
        next_state <= MA;
    end if;
-- remaining states not shown

when others => lights <= "100100";
    start_a <= '1';
    next_state <= R_R;

end case;
end if;
end process comb;
-- purpose: update
-- type : sequential
-- inputs : clk
-- outputs:
ticker: process (clk)
begin -- process ticker
    if clk'event and clk = '1' then -- rising clock edge
        current_state <= next_state;
    end if;
end process ticker;
end asm;
```

2.a) 8

Define the problem domain for the processor 1.
Develop informal description of the processor 1 and its instructions 1,
Write formal description of programmers level (abstract RTL) 1 and programmers register set 1,
specify high level data path architecture to fulfil requirements of abstract RTL 2,
write concrete RTL sequences for each instruction 1, must use only register transfers allowed from earlier step 1,
design logic circuits for the data path 1 and identify the control signals required 1.
write the control signals for each concrete RTL step 1 to produce a complete set for all instructions 1.
Design the sequencer and control unit 2, driven by master clock 1, to generate the control sequences 1.

2.b) 5)

2 marks for naming conditional jumps eg. carry set & carry clear, overflow set & clear and absolute jumps eg jump positive(N = 0), jump negative(N = 1), 1 mark for set carry flag, clear carry flag,
2 marks for for examples of multi-byte arithmetic (addition and subtraction). give marks for abstract RTL in the absence of other examples.

2.c) 12)

Expect discussion of carry and overflow generation when adding/subtracting using 2's complement.
examples of addition showing carry from bit 6 -> bit 7 and carry out, (carry from 6 to 7 xor carry out) and addition = overflow.
for subtraction using $a - b \equiv a + (-b)$ then carry out of 0 = overflow.
V flag generated from (not add and not carryout) or (add and (carryout xor carry6-7))
expect discussion of need to modify carry_in to ALU to be from carry flag not from control unit. need to set/clear carry flag for correct arithmetic. So must have carry_set and carry_clear instructions. Need to bring the carry_into_bit_7 signal out of the ALU into the flag generation hardware. Negative flag is just a copy of bit 7. Z is calculated as before.

ANSWERS

3.a 10)

- i) expect and gate, because all combinations covered no need to preserve value, clk & a in sensitivity list so evaluated whenever either changes.
 - ii) transparent latch, as incomplete statement so infers that value must be preserved, latch because it is level sensitive and transparent as input is transferred to output when clk is 1
- 2 marks per correct diagram, 6 marks for description and explanation. Trade diagram marks for clear written explanations.

3.b 15)

- i) Errors include :-

Missing init & ready from sensitivity list of state_comb.

```
state_comb: process( current_state, read_write, init,
ready)
```

Missing init condition for initialisation of next_state.

```
if ( init = '0') then next_state <= idle;
else case current_state is
```

Missing default assignment in idle state.

```
when idle ...
```

```
if (ready = '1') ... else next_state <= idle; end if;
```

1 mark per error + 2 marks+ per correct code fragment.

- ii) If the init 'if' clause is added without init/reset being in sensitivity list then compiler warning. Missing 'if init may lead to random starting conditions. Missing default assignment may give compiler warnings(not on Alliance or older Quartus) certainly will give odd behaviour of statemachine.

ANSWERS

ANSWERS

4.a) 10

Expect a 1 to 4 demux. Marks for labelling signals and gate instances as well as for correct schematic. 2-4 decoder + 4 and gates.

4.b 15)

They're supplied with a 2 input nand so give marks for creating and & not subcomponents (up to 4 marks). 5 marks for overall correct approach to netlists in VHDL. Expect entity with all inputs and outputs. Penalise (-1 for each) for including clk and/or reset as it's combinational logic. Architecture should show signals and components, signal names should amtch schematic in a), body of arch should have instance names matching the schematic also.

ANSWERS

ANSWERS

ANSWERS

ANSWERS