**INF3430/INF4431 H2018**

**NB: Dette er et løsningsforslag, ekvivalente løsninger vil også bli godtatt.**

## Oppgave 1-3, 6 og 8-9:

| Oppgave | A | B | C | D |
|---|---|---|---|---|
| 1 | X | | | |
| 2 | X | | X | |
| 3 | | X | X | |
| 6 | | | | X |
| 8 | X | | | |
| 9 | | X | | |

## Oppgave 4)

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

entity compute_comb is
  port
    (a      : in  std_logic_vector(15 downto 0);
     b      : in  std_logic_vector(15 downto 0);
     c      : in  std_logic_vector(15 downto 0);
     d      : in  std_logic_vector(15 downto 0);
     result : out std_logic_vector(32 downto 0));
end entity compute_comb;

architecture rtl of compute_comb is
begin

  process (all) is
    variable ab : unsigned(31 downto 0);
    variable cd : unsigned(31 downto 0);
  begin
     ab := unsigned(a) * unsigned(b);
```

```
    cd := unsigned(c) * unsigned(d);
    result <= std_logic_vector(('0' & ab) + ('0' & cd));
  end process;

end architecture rtl;
```

**Alternativ løsning:**

```
Architecture rtl2 of compute_comb is

  signal ab : unsigned(31 downto 0);
  signal cd : unsigned(31 downto 0);

begin

  process (all) is   -- process(a, b, c, d, ab, cd) is also correct.
  begin
    ab <= unsigned(a) * unsigned(b);
    cd <= unsigned(c) * unsigned(d);
    result <= std_logic_vector(('0' & ab) + ('0' & cd));
  end process;

end architecture rtl2;
```

## -- Det er ikke krav om testbenk i besvarelsen

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

entity tb_compute_comb is
  -- empty;
end tb_compute_comb;

architecture beh of tb_compute_comb is

  component compute_comb
    port (
      a      : in  std_logic_vector(15 downto 0);
      b      : in  std_logic_vector(15 downto 0);
      c      : in  std_logic_vector(15 downto 0);
      d      : in  std_logic_vector(15 downto 0);
      result : out std_logic_vector(32 downto 0));
  end component;

  signal a      : std_logic_vector(15 downto 0);
  signal b      : std_logic_vector(15 downto 0);
  signal c      : std_logic_vector(15 downto 0);
  signal d      : std_logic_vector(15 downto 0);
  signal result : std_logic_vector(32 downto 0);
```

```
begin

  compute_comb_0: compute_comb
    port map (
      a      => a,
      b      => b,
      c      => c,
      d      => d,
      result => result);

  P_TEST: process
  begin

    a <= (others => '0');
    b <= (others => '0');
    c <= (others => '0');
    d <= (others => '0');

    wait for 40 ns;

    a <= x"0001";
    b <= x"0002";
    c <= x"0003";
    d <= x"0004";

    wait for 10 ns;

    a <= x"0105";
    b <= x"0206";
    c <= x"0307";
    d <= x"0408";

    wait for 10 ns;

    a <= x"FFFF";
    b <= x"FFFE";
    c <= x"FFFF";
    d <= x"FFFC";

    wait for 10 ns;

    a <= x"FFFF";
    b <= x"FFFF";
    c <= x"FFFF";
    d <= x"FFFF";

    wait for 10 ns;

    wait;

  end process;
end beh;
```

**Oppgave 5)**

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

entity compute_seq is
  port (
    rst     : in  std_logic;
    clk     : in  std_logic;
    a       : in  std_logic_vector(15 downto 0);
    b       : in  std_logic_vector(15 downto 0);
    c       : in  std_logic_vector(15 downto 0);
    d       : in  std_logic_vector(15 downto 0);
    vdata   : in std_logic;
    result  : out std_logic_vector(32 downto 0);
    vresult : out std_logic);
end entity compute_seq;

architecture rtl of compute_seq is

  signal ab       : unsigned(31 downto 0);
  signal cd       : unsigned(31 downto 0);
  signal vdata_d1 : std_logic;

begin

  process (rst, clk) is
  begin
    if rst = '1' then
      ab       <= (others => '0');
      cd       <= (others => '0');
      vdata_d1 <= '0';
      result   <= (others => '0');
      vresult  <= '0';
    elsif rising_edge(clk) then
      ab       <= unsigned(a) * unsigned(b);
      cd       <= unsigned(c) * unsigned(d);
      vdata_d1 <= vdata;
      if vdata_d1='1' then
        result <= std_logic_vector(('0' & ab) + ('0' & cd));
      else
        result <= (others => '0');
      end if;
      vresult  <= vdata_d1;
    end if;
  end process;
```

```
end architecture rtl;
```

**-- Det er ikke krav om testbenk i besvarelsen**

```
entity tb_compute_seq is
  -- empty;
end tb_compute_seq;

architecture beh of tb_compute_seq is

  component compute_seq
    port (
      rst     : in  std_logic;
      clk     : in  std_logic;
      a       : in  std_logic_vector(15 downto 0);
      b       : in  std_logic_vector(15 downto 0);
      c       : in  std_logic_vector(15 downto 0);
      d       : in  std_logic_vector(15 downto 0);
      vdata   : in  std_logic;
      result  : out std_logic_vector(32 downto 0);
      vresult : out std_logic);
  end component;

  signal rst     : std_logic;
  signal clk     : std_logic:= '0';
  signal a       : std_logic_vector(15 downto 0);
  signal b       : std_logic_vector(15 downto 0);
  signal c       : std_logic_vector(15 downto 0);
  signal d       : std_logic_vector(15 downto 0);
  signal vdata   : std_logic;
  signal result  : std_logic_vector(32 downto 0);
  signal vresult : std_logic;


begin

  compute_seq_0: compute_seq
    port map (
      rst     => rst,
      clk     => clk,
      a       => a,
      b       => b,
      c       => c,
      d       => d,
      vdata   => vdata,
      result  => result,
      vresult => vresult);

  -- Clock and reset generation
```

```
clk <= not clk after 10 ns;
rst <= '1', '0' after 20 ns;

P_TEST: process
begin

  a <= (others => '0');
  b <= (others => '0');
  c <= (others => '0');
  d <= (others => '0');
  vdata <= '0';

  wait until falling_edge(rst);

  a <= x"0001";
  b <= x"0002";
  c <= x"0003";
  d <= x"0004";
  vdata <= '1';

  wait for 20 ns;

  a <= x"0105";
  b <= x"0206";
  c <= x"0307";
  d <= x"0408";

  wait for 20 ns;

  a <= x"01FF";
  b <= x"02FE";
  c <= x"03FD";
  d <= x"04FC";
  vdata <= '0';

  wait for 20 ns;

  a <= x"FFFF";
  b <= x"FFFE";
  c <= x"FFFF";
  d <= x"FFFC";
  vdata <= '0';

  wait for 20 ns;

  a <= x"FFFF";
  b <= x"FFFE";
  c <= x"FFFF";
  d <= x"FFFC";
  vdata <= '1';

  wait for 20 ns;
```

```
    a <= x"FFFF";
    b <= x"FFFF";
    c <= x"FFFF";
    d <= x"FFFF";

    wait for 20 ns;

    wait;

  end process;
end beh;
```

## Oppgave 7)

```
package mypack is
 type month_type is (JAN, FEB, MAR, APR, MAY, JUN,
                     JUL, AUG, SEP, OCT, NOV, DEC);
end mypack;


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.mypack.all;

package somesubprograms is

  function numbermonthdays (leap_year : boolean;
                            month     : month_type) return unsigned;

  procedure days (leap_year   : in  boolean;
                  month       : in  month_type;
                  monthdays   : out unsigned(4 downto 0);
                  quarterdays : out unsigned(6 downto 0));

end package;


package body somesubprograms is

  function numbermonthdays (leap_year : boolean;
                            month     : month_type) return unsigned is
    variable monthdays : unsigned(4 downto 0);
  begin

    case month is
      when FEB =>
        if leap_year then
          monthdays := "11101";
        else
          monthdays := "11100";
        end if;
      when JAN | MAR | MAY | JUL |
           AUG | OCT | DEC =>
        monthdays := "11111";
```

```
      when others =>
        monthdays := "11110";
    end case;

    return monthdays;

  end numbermonthdays;


  procedure days (leap_year   : in  boolean;
                  month       : in  month_type;
                  monthdays   : out unsigned(4 downto 0);
                  quarterdays : out unsigned(6 downto 0)) is
  begin

    monthdays := numbermonthdays(leap_year, month);

    case month is
      when JAN | FEB | MAR =>
        quarterdays := ("00" & numbermonthdays(leap_year, JAN)) +
                       ("00" & numbermonthdays(leap_year, FEB)) +
                       ("00" & numbermonthdays(leap_year, MAR));
      when APR | MAY | JUN =>
        quarterdays := ("00" & numbermonthdays(leap_year, APR)) +
                       ("00" & numbermonthdays(leap_year, MAY)) +
                       ("00" & numbermonthdays(leap_year, JUN));
      when JUL | AUG | SEP =>
        quarterdays := ("00" & numbermonthdays(leap_year, JUL)) +
                       ("00" & numbermonthdays(leap_year, AUG)) +
                       ("00" & numbermonthdays(leap_year, SEP));
      when others =>
        quarterdays := ("00" & numbermonthdays(leap_year, OCT)) +
                       ("00" & numbermonthdays(leap_year, NOV)) +
                       ("00" & numbermonthdays(leap_year, DEC));
    end case;

  end days;

end package body;
```

## -- Det er ikke krav om testbenk i besvarelsen

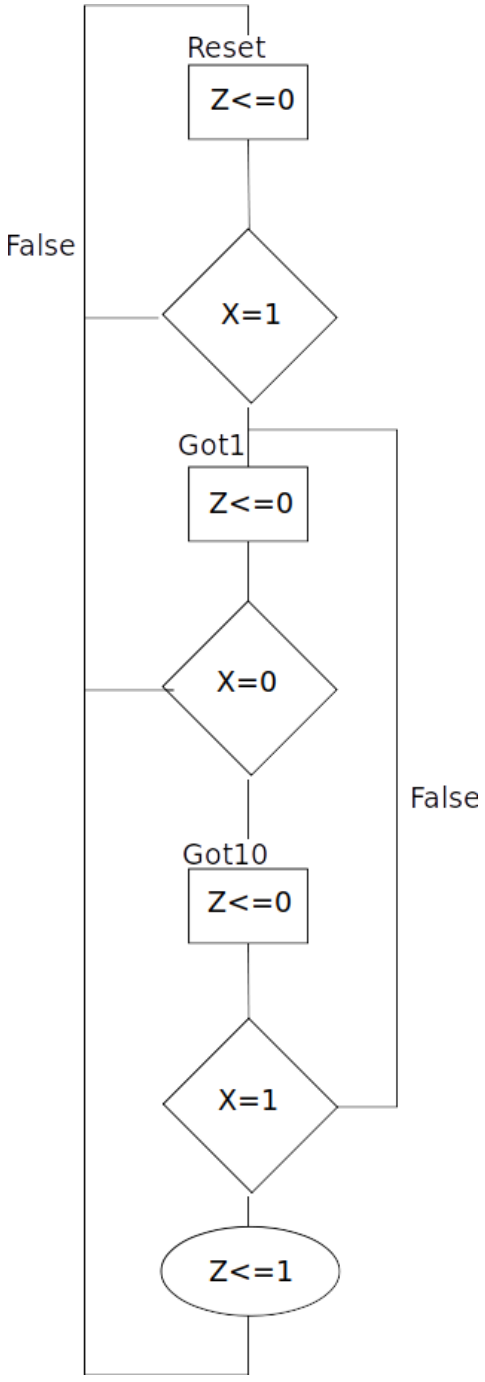
```
use std.textio.all;
use std.env.all; -- Defines finish(0) function etc.

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

library std_developerskit;
use std_developerskit.std_iopak.all;
```

```
use work.mypack.all;
use work.somesubprograms.all;

entity tb_somesubprograms is
end ;


architecture beh of tb_somesubprograms is

begin

  STIM: process
    variable leap_year   : boolean;
    variable month       : month_type;
    variable monthdays   : unsigned(4 downto 0);
    variable quarterdays : unsigned(6 downto 0);
  begin

    leap_year := FALSE;
      month     := FEB;
      monthdays := numbermonthdays(leap_year, month);
      assert (false) report "Number of month days in non leap year february: "
        & to_string(to_integer(monthdays))
      severity note;


      wait for 10 ns;

    leap_year := TRUE;
      month     := FEB;
      days(leap_year, month, monthdays, quarterdays);
      assert (false) report "Number of month days in leap year february: "
        & to_string(to_integer(monthdays))
      severity note;
      assert (false) report "Number of quarter days in leap year
        first quarter: " & to_string(to_integer(quarterdays))
      severity note;

      wait for 10 ns;

    leap_year := FALSE;
      month     := DEC;
      monthdays := numbermonthdays(leap_year, month);
      assert (false) report "Number of month days in december: "
        & to_string(to_integer(monthdays))
      severity note;

      wait for 10 ns;

    leap_year := TRUE;
      month     := DEC;
      days(leap_year, month, monthdays, quarterdays);
      assert (false) report "Number of month days in december: "
        & to_string(to_integer(monthdays))
      severity note;
      assert (false) report "Number of quarter days in fourth quarter: "
        & to_string(to_integer(quarterdays))
```

```
      severity note;
      wait;
  end process;
end architecture beh;
```

**Oppgave 10)**

**ASM Diagram:**



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

entity mealy_detector is
  port (
    rst : in std_logic;
    clk : in std_logic;
    x   : in  std_logic;
    z   : out std_logic);
end mealy_detector;


architecture rtl of mealy_detector is

  type state_type is (RESET, GOT1, GOT10);
  signal current_state, next_state : state_type;

  signal z_current, z_next : std_logic;
  signal x_s1, x_s2: std_logic;

begin
  P_SYNC: process(rst, clk)
  begin

    if rising_edge(clk) then
      if rst = '1' then
       x_s1 <= '0';
       x_s2 <= '0';
      else
       x_s1 <= x;
       x_s2 <= x_s1;
      end if;
   end if;
  end process;


  P_SEQ : process(rst, clk)
  begin

    if rising_edge(clk) then
   if rst = '1' then
      current_state <= reset;
      z_current     <= '0';
   else
      current_state <= next_state;
      z_current     <= z_next;
    end if;
   end if;
  end process;


  P_COMB : process(current_state, x_s2)
  begin

    -- Default values
```

```
    next_state<= current_state;
    z_next <= '0';

    case current_state is
      when RESET =>
        if x_s2 = '1' then
          next_state <= GOT1;
        end if;

      when GOT1 =>
        if x_s2 = '0' then
          next_state <= GOT10;
        end if;

      when GOT10 =>
        if x_s2 = '1' then
          next_state <= GOT1;
          z_next     <= '1';
        else
          next_state <= RESET;
        end if;

      when others =>
        next_state <= RESET;
    end case;

  end process;

  -- Concurrent statements
  z <= z_current;

end rtl;
```

## -- Testbenk:

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

entity tb_mealy_detector is
  -- empty
end tb_mealy_detector;

architecture beh of tb_mealy_detector is

  component mealy_detector
    port (
      rst : in std_logic;
      clk : in std_logic;
      x   : in  std_logic;
      z   : out std_logic);
  end component;
```

```
  signal clk : std_logic := '0';
  signal rst : std_logic;

  signal x : std_logic;
  signal z : std_logic;

  signal data : std_logic_vector(7 downto 0);

begin

  uut : mealy_detector port map(
    rst => rst,
    clk => clk,
    x   => x,
    z   => z);

  clk <= not clk after 10 ns;

  main : process

  begin

    rst <= '1';
    x   <= '0';
    wait for 20 ns;

    rst <= '0';
    wait until falling_edge(clk);

    data <= x"55";
    for i in 0 to 7 loop
      wait until falling_edge(clk);
      x <= data(7-i);
    end loop;  -- i in 7 downto 0

    wait until falling_edge(clk);

    x <= '0';
    wait until falling_edge(clk);


    data <= x"4A";
    for i in 0 to 7 loop
      wait until falling_edge(clk);
      x <= data(7-i);
    end loop;  -- i in 7 downto 0


    x <= '0';
    wait until falling_edge(clk);


    data <= x"6C";
    for i in 0 to 7 loop
      wait until falling_edge(clk);
```

```
      x <= data(7-i);
    end loop;  -- i in 7 downto 0

    wait until falling_edge(clk);
    x <= '0';

    wait until falling_edge(clk);

    wait;
  end process;

end beh;
```