INF3430/INF4431 H2017.

Fasit oppgavene 1-5 og 9-10:

Oppgave	A	B	C	D	E
1			X		
2		X	X	X	X
3		X		X	
4				X	
5			X	X	X
9			X		
10	X*				X*

Kommentar til oppgave 10:

Vi har valgt å la de som har oppgitt enten alternativ A (dvs. 51) eller alternativ E (dvs. 69) få poeng i oppgave 10. De som har oppgitt feil eller unnlatt å besvare oppgaven 10 vil IKKE få minus poeng. Se fasit for oppgave 8 for begrunnelse.

Oppgave 6)

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

entity arbitrary_sequence_gen is
  port
    (
      rst   : in  std_logic;
      clk   : in  std_logic;
      run   : in  std_logic;
      q     : out std_logic_vector(2 downto 0)
    );
```

```
end entity arbitrary_sequence_gen;

architecture rtl of arbitrary_sequence_gen is

  type   state_type is (VALUE1, VALUE2, VALUE3, VALUE4, VALUE5, VALUE6,
VALUE7);
  signal present_state, next_state : state_type;

begin

  --stateregister
  STATE_REG :
  process (rst, clk) is
  begin
    if rst = '1' then
      present_state <= VALUE1;
    elsif rising_edge(clk) then
      present_state <= next_state;
    end if;
  end process state_reg;

  --Nextstate logic and output logic
  COMB :
  process (present_state, run) is
  begin

    -- Set default next state to current state
    next_state <= present_state;

    case present_state is
      when VALUE1 =>
        q <= "100";   -- i.e. 4
        if run='1' then
          next_state <= VALUE2;
        end if;
      when VALUE2 =>
        q <= "010";   -- i.e. 2
        if run='1' then
          next_state <= VALUE3;
        end if;
      when VALUE3 =>
        q <= "101";   -- i.e. 5
        if run='1' then
          next_state <= VALUE4;
        end if;
      when VALUE4 =>
        q <= "110";   -- i.e. 6
        if run='1' then
          next_state <= VALUE5;
        end if;
     when VALUE5 =>
        q <= "111";   -- i.e. 7
```

```
        if run='1' then
          next_state <= VALUE6;
        end if;
      when VALUE6 =>
        q <= "011";   -- i.e. 3
        if run='1' then
          next_state <= VALUE7;
        end if;
      when VALUE7 =>
        q <= "001";   -- i.e. 1
        if run='1' then
          next_state <= VALUE1;
        end if;
    end case;
  end process COMB;

end architecture rtl;
```

-- Det er ikke krav om testbenk i besvarelsen

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

entity tb_arbitrary_sequence_gen is
  -- empty;
end tb_arbitrary_sequence_gen;

architecture beh of tb_arbitrary_sequence_gen is

  component arbitrary_sequence_gen
    port (
      rst : in  std_logic;
      clk : in  std_logic;
      run : in  std_logic;
      q   : out std_logic_vector(2 downto 0));
  end component;

  signal rst : std_logic;
  signal clk : std_logic:= '0';
  signal run : std_logic;
  signal q   : std_logic_vector(2 downto 0);

begin

  arbitrary_sequence_gen_0: arbitrary_sequence_gen
    port map (
      rst => rst,
      clk => clk,
      run => run,
      q   => q);
```

```
  -- Clock and reset generation
  clk <= not clk after 10 ns;
  rst <= '1', '0' after 20 ns;

  P_TEST: process
  begin

    run <= '0';

    wait until falling_edge(rst);

    wait for 40 ns;

    run <= '1';

    wait for 200 ns;

    run <= '0';

    wait for 40 ns;

    run <= '1';

    wait for 200 ns;

    run <= '0';

    wait for 20 ns;

    wait;

  end process;

end beh;
```

Oppgave 7)

I oppgave 7 var det ved en feil ikke blitt oppgitt funksjonen til signalet run. Det vil derfor ikke bli trukket noen poeng hvis run ikke har blitt brukt i løsningen.
Load av ny seed kan enten gjøres som vist i fasiten eller alternativt brukes med en gang uten «else» statementet ettersom det heller ikke var oppgitt i oppgaven.

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

```
entity lfsr is
  port
    (rst   : in  std_logic;
     clk   : in  std_logic;
     load  : in  std_logic;
     seed  : in  std_logic_vector(2 downto 0);
     run   : in  std_logic;
     q     : out std_logic_vector(2 downto 0);
     err   : out std_logic);
end entity lfsr;

architecture rtl of lfsr is

  -- Shift register
  signal q_i : std_logic_vector(2 downto 0);

begin

  process (rst, clk) is
    variable feedback : std_logic;
  begin
    if rst = '1' then
      q_i <= "100";
      err <= '0';
    elsif rising_edge(clk) then
      if load='1' then
        if seed="000" then
          err <= '1';
        else
          err <= '0';
        end if;
        q_i <= seed;
      else
        if run='1' then
          feedback := q_i(0) xor q_i(1);
          q_i <= feedback & q_i(2 downto 1);
        end if;
      end if;
    end if;
  end process;

  -- Concurrent output statements
  q <= q_i;

end architecture rtl;
```

-- Det er ikke krav om testbenk i besvarelsen

```
library ieee;
```

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

entity tb_lfsr is
  -- empty;
end tb_lfsr;

architecture beh of tb_lfsr is

  component lfsr
    port (
      rst  : in  std_logic;
      clk  : in  std_logic;
      load : in  std_logic;
      seed : in  std_logic_vector(2 downto 0);
      run  : in  std_logic;
      q    : out std_logic_vector(2 downto 0);
      err  : out std_logic);
  end component;

  signal rst  : std_logic;
  signal clk  : std_logic:= '0';
  signal load : std_logic;
  signal seed : std_logic_vector(2 downto 0);
  signal run  : std_logic;
  signal q    : std_logic_vector(2 downto 0);
  signal err  : std_logic;

begin

  lfsr_0: lfsr
    port map (
      rst  => rst,
      clk  => clk,
      load => load,
      seed => seed,
      run  => run,
      q    => q,
      err  => err);

  -- Clock and reset generation
  clk <= not clk after 10 ns;
  rst <= '1', '0' after 20 ns;

  P_TEST: process
  begin

    load <= '0';
    seed <= (others => '0');
    run  <= '0';

    wait until falling_edge(rst);
```

```
    wait for 40 ns;
    load <= '1';
    seed <= "100";

    wait for 20 ns;
    load <= '0';
    seed <= "000";

    wait for  20 ns;
    run <= '1';

    wait for 200 ns;
    run <= '0';

    wait for 40 ns;
    run <= '1';

    wait for 200 ns;
    run <= '0';

    wait for 20 ns;
    load <= '1';
    seed <= "000";

    wait until rising_edge(err);

    wait for 20 ns;
    load <= '0';
    seed <= "000";

    wait for 40 ns;
    load <= '1';
    seed <= "100";

    wait for 20 ns;
    load <= '0';
    seed <= "000";

    wait for  20 ns;
    run <= '1';

    wait for 100 ns;
    run <= '0';

    wait for 40 ns;

    wait;

  end process;

end beh;
```

Oppgave 8)

I oppgaven kom antall pipelinetrinn for dårlig frem. Derfor blir det flere løsningen hvor det enten kan utføres 2 addisjoner i første pipelinetrinn og deretter en addisjon og en større enn operasjon i neste pipelinetrinn, eller det er også mulig å tolke oppgaven slik at i første pipelinetrinn utføres 2 addisjoner i parallell, deretter en addisjon i neste pipelinetrinn og i tredje pipelinetrinn utføres større enn operasjonen.

Oppgaven kan enten løses med 1 prosess eller med 3 prosesser.

Det er mulig å la alle pipeline registrene i VHDL koden være på 18 bit (dvs. 17 downto 0). Da vil syntese verktøyet til Vivado fjerne/optimalisere bort unødvendige registre (dvs. 2) slik at antall registere/flip-flop i oppgave 10 blir 51 (pga. 17+17+16+1=51) for 2 pipeline løsningen og 69 (pga. 17+17+18+16+1=69) for 3 pipeline løsningen.

Vi har valgt å la de som har oppgitt enten 51 eller 69 få poeng i oppgave 10. De som har oppgitt feil eller unnlatt å besvare oppgaven 10 vil <u>IKKE</u> få minus poeng.

Løsning med 3 pipelinetrinn:

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

entity compute 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);
     result : out std_logic_vector(15 downto 0);
     max    : out std_logic);
end entity compute;


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

```
use ieee.numeric_std.all;

architecture pipelined_rtl of compute is

  signal ab_tmp   : unsigned(16 downto 0);
  signal cd_tmp   : unsigned(16 downto 0);
  signal result_i : unsigned(17 downto 0);

begin

  process (rst, clk) is
  begin
    if rst = '1' then
      ab_tmp   <= (others => '0');
      cd_tmp   <= (others => '0');
      result_i <= (others => '0');
      result   <= (others => '0');
      max      <= '0';
    elsif rising_edge(clk) then
      ab_tmp   <= unsigned('0' & a) + unsigned('0' & b);
      cd_tmp   <= unsigned('0' & c) + unsigned('0' & d);
      result_i <= ('0' & ab_tmp) + ('0' & cd_tmp);
      if result_i>"0011111111111111" then
        result <= (others => '1');
        max    <= '1';
      else
        result <= std_logic_vector(result_i(15 downto 0));
        max    <= '0';
      end if;
    end if;
  end process;

end architecture pipelined_rtl;
```

Alternativ løsning med 3 prosesser og 3 pipelinetrinn:

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

architecture pipelined2_rtl of compute is

  signal ab_tmp   : unsigned(16 downto 0);
  signal cd_tmp   : unsigned(16 downto 0);
  signal result_i : unsigned(17 downto 0);

begin

  process (rst, clk) is
```

```
  begin
    if rst = '1' then
      ab_tmp <= (others => '0');
      cd_tmp <= (others => '0');
    elsif rising_edge(clk) then
      ab_tmp <= unsigned('0' & a) + unsigned('0' & b);
      cd_tmp <= unsigned('0' & c) + unsigned('0' & d);
   end if;
  end process;

  process (rst, clk) is
  begin
    if rst = '1' then
      result_i <= (others => '0');
    elsif rising_edge(clk) then
      result_i <= ('0' & ab_tmp) + ('0' & cd_tmp);
    end if;
  end process;

  process (rst, clk) is
  begin
    if rst = '1' then
      result <= (others => '0');
      max    <= '0';
    elsif rising_edge(clk) then
      if result_i>"0011111111111111" then
        result <= (others => '1');
        max    <= '1';
      else
        result <= std_logic_vector(result_i(15 downto 0));
        max    <= '0';
      end if;
    end if;
  end process;

end architecture pipelined2_rtl;
```

Alternativ løsning med 3 pipelinetrinn:

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

architecture pipelined3_rtl of compute is

  signal ab_tmp   : unsigned(17 downto 0);
  signal cd_tmp   : unsigned(17 downto 0);
  signal result_i : unsigned(17 downto 0);
```

```
begin

  process (rst, clk) is
  begin
    if rst = '1' then
      ab_tmp   <= (others => '0');
      cd_tmp   <= (others => '0');
      result_i <= (others => '0');
      result   <= (others => '0');
      max      <= '0';
    elsif rising_edge(clk) then
      ab_tmp   <= unsigned("00" & a) + unsigned("00" & b);
      cd_tmp   <= unsigned("00" & c) + unsigned("00" & d);
      result_i <= ab_tmp + cd_tmp;
      if result_i>"0011111111111111" then
        result <= (others => '1');
        max    <= '1';
      else
        result <= std_logic_vector(result_i(15 downto 0));
        max    <= '0';
      end if;
    end if;
  end process;

end architecture pipelined3_rtl;
```

Alternativ løsning med 2 pipelinetrinn:

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

architecture pipelined4_rtl of compute is

  signal ab_tmp   : unsigned(16 downto 0);
  signal cd_tmp   : unsigned(16 downto 0);

begin

  process (rst, clk) is
    variable result_i : unsigned(17 downto 0);
  begin
    if rst = '1' then
      ab_tmp   <= (others => '0');
      cd_tmp   <= (others => '0');
      result   <= (others => '0');
      max      <= '0';
    elsif rising_edge(clk) then
```

```
      ab_tmp   <= unsigned('0' & a) + unsigned('0' & b);
      cd_tmp   <= unsigned('0' & c) + unsigned('0' & d);
      result_i := ('0' & ab_tmp) + ('0' & cd_tmp);
      if result_i>"0011111111111111" then
        result <= (others => '1');
        max    <= '1';
      else
        result <= std_logic_vector(result_i(15 downto 0));
        max    <= '0';
      end if;
    end if;
  end process;

end architecture pipelined4_rtl;
```

Alternativ løsning med 2 pipelinetrinn:

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

architecture pipelined5_rtl of compute is

  signal ab_tmp   : unsigned(17 downto 0);
  signal cd_tmp   : unsigned(17 downto 0);

begin

  process (rst, clk) is
    variable result_i : unsigned(17 downto 0);
  begin
    if rst = '1' then
      ab_tmp   <= (others => '0');
      cd_tmp   <= (others => '0');
      result   <= (others => '0');
      max      <= '0';
    elsif rising_edge(clk) then
      ab_tmp   <= unsigned("00" & a) + unsigned("00" & b);
      cd_tmp   <= unsigned("00" & c) + unsigned("00" & d);
      result_i := ab_tmp + cd_tmp;
      if result_i>"0011111111111111" then
        result <= (others => '1');
        max    <= '1';
      else
        result <= std_logic_vector(result_i(15 downto 0));
        max    <= '0';
      end if;
    end if;
  end process;
```

```
end architecture pipelined5_rtl;
```

-- Det er ikke krav om testbenk i besvarelsen

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

entity tb_compute is
  -- empty;
end tb_compute;

architecture beh of tb_compute is

  component compute
    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);
      result : out std_logic_vector(15 downto 0);
      max    : 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 result : std_logic_vector(15 downto 0);
  signal max    : std_logic;

begin

  compute_0: compute
    port map (
      rst    => rst,
      clk    => clk,
      a      => a,
      b      => b,
      c      => c,
      d      => d,
      result => result,
      max    => max);

  -- 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');

  wait until falling_edge(rst);

  wait for 40 ns;

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

  wait for 100 ns;

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

  wait for 100 ns;

  a <= x"FFFF";
  b <= x"0000";
  c <= x"0001";
  d <= x"0000";

  wait until rising_edge(max);
  wait for 90 ns;

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

  wait for 100 ns;

  a <= x"0111";
  b <= x"0222";
  c <= x"0333";
  d <= x"0444";

  wait for 100 ns;

  wait;
```

```
  end process;

end beh;
```