Fasit INF3430/4431 eksamen 2017

Oppgave 11a)

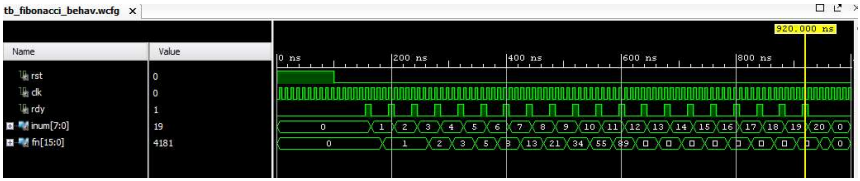
```
-- Fasit oppgave 11 a)
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fibonacci is
  generic
    (
      FIBWIDTH : natural := 32 --antall bit i generert Fibonaccitall
      );
  port
    (
      rst       : in  std_logic; --asynkron reset
      clk       : in  std_logic; --clock
      run       : in  std_logic; --en klokkepuls starter funksjonsgenratoren
      funcsel   : in  std_logic_vector(2 downto 0); --funksjonsvalg, "001"
                                                    --velger Fibonacci
      nmax      : in  std_logic_vector(7 downto 0); --angir max antall Fibonacci
                                                    --tall som skal genereres
      inum      : out std_logic_vector(7 downto 0); --angir Fibonaccitall index n
      rdy       : out std_logic; --en positiv puls med en clk-periodes
                                 --varighet for å angi et gyldig
      fn        : out std_logic_vector (FIBWIDTH-1 downto 0) --Fibonaccitall n
      );
end fibonacci;

architecture RTL_fibonacci of fibonacci is
  signal f0, next_f0 : unsigned(FIBWIDTH-1 downto 0);
  signal f1, next_f1 : unsigned(FIBWIDTH-1 downto 0);
  signal f2, next_f2 : unsigned(FIBWIDTH-1 downto 0);
  signal n, next_n   : unsigned(7 downto 0);

  type fib_st is (idle_st, calc_st, fn_st, f1_st, f0_st, update_st, rdy_st);
  signal curr_st, next_st : fib_st;
begin

  state_reg :
  process (clk, rst)
  begin
    if rst = '1' then
      curr_st <= idle_st;
      f0      <= (others => '0');
      f1      <= ((0)    => '1', others => '0');
      f2      <= (others => '0');
      n       <= (others => '0');
    elsif rising_edge(clk) then
      curr_st <= next_st;
      f0      <= next_f0;
      f1      <= next_f1;
      f2      <= next_f2;
      n       <= next_n;
    end if;
  end process;


  next_state_comb :
  process(run,funcsel,nmax,curr_st,f0,f1,f2,next_n,n)
  ---process (all) --vhdl 2008
  begin
    next_st   <= curr_st;
```

```
    next_n    <= n;
    rdy       <= '0';
    next_f0 <= f0;
    next_f1 <= f1;
    next_f2 <= f2;

    case curr_st is
      when idle_st =>
        next_f0 <= (others => '0');
        next_f1 <= (others => '0');
        next_f2 <= (others => '0');
        next_n    <= (others => '0');
        if run = '1' then
          if funcsel = "001" then
            next_st <= calc_st;
          end if;
        end if;

      when calc_st =>
        if n >= 2 then
          next_st <= fn_st;
        elsif N = 1 then
          next_st <= f1_st;
        else
          next_st <= f0_st;
        end if;

      when f0_st =>
        next_f2 <= (others => '0');
        next_st   <= update_st;

      when f1_st =>
        next_f2 <= ((0) => '1', others => '0');
        next_st   <= update_st;

      when fn_st =>
        next_f2 <= f0 + f1;
        next_st   <= update_st;

      when update_st =>
        next_f0 <= f1;
        next_f1 <= f2;
        if n = unsigned(nmax) then
          next_st <= idle_st;
        else
          next_st <= rdy_st;
        end if;

      when rdy_st =>
        rdy     <= '1';
        next_n  <= n + 1;
        next_st <= calc_st;

    end case;
  end process;

  inum <= std_logic_vector(n);
  fn   <= std_logic_vector(f2);

end RTL_fibonacci;
```

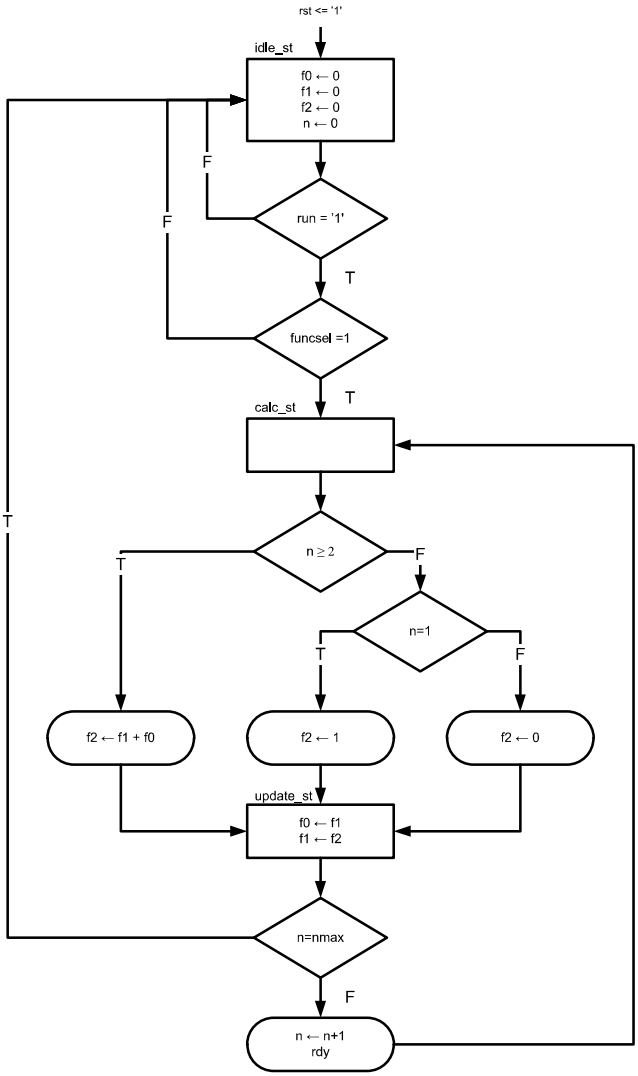
Oppgave 11b)

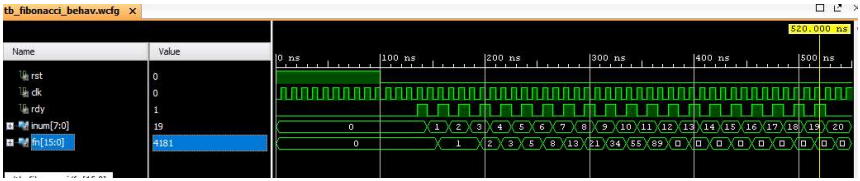
```
--Fasit oppgave 11b)
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_fibonacci is
end tb_fibonacci;

architecture testbench of tb_fibonacci is

constant MYNMAX  : natural:= 20;
type fibfasit is array (0 to MYNMAX-1) of natural;
signal myfasit : fibfasit := (0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,
                              987,1597,2584,4181);
constant MYWIDTH : natural:= 16;

component fibonacci is
  generic
    (
      FIBWIDTH : natural := 32 --antall bit i generert Fibonaccitall
      );
  port
    (
      rst       : in  std_logic; --asynkron reset
      clk       : in  std_logic; --clock
      run       : in  std_logic; --en klokkepuls starter funksjonsgenratoren
      funcsel   : in  std_logic_vector(2 downto 0); --funksjonsvalg, "001"
                                                    --velger Fibonacci
      nmax      : in  std_logic_vector(7 downto 0); --angir max antall Fibonacci
                                                    --tall som skal genereres
      inum      : out std_logic_vector(7 downto 0); --angir Fibonaccitall index n
      rdy       : out std_logic; --en positiv puls med en clk-periodes
                                 --varighet for å angi et gyldig
      fn        : out std_logic_vector (FIBWIDTH-1 downto 0) --Fibonaccitall n
      );
end component fibonacci;

signal rst     : std_logic := '0';
signal clk     : std_logic := '0';
signal run     : std_logic := '0';
signal funcsel : std_logic_vector(2 downto 0) := "001";
signal nmax    : std_logic_vector(7 downto 0) :=
std_logic_vector(to_unsigned(MYNMAX,8));
signal inum    : std_logic_vector(7 downto 0);
signal rdy     : std_logic;
signal fn      : std_logic_vector (MYWIDTH-1 downto 0);

begin

  UUT : fibonacci
    generic map
    (
      MYWIDTH
    )
    port map
    (
      rst     => rst,
      clk     => clk,
      run     => run,
      funcsel => funcsel,
      nmax    => nmax,
      inum    => inum,
      rdy     => rdy,
      fn      => fn
    );
```

```
  clk <= not clk after 5 ns;

  stimuli :
  process
    variable errorcnt : integer := 0;
  begin
    rst   <= '1', '0' after 100 ns;
    wait for 120 ns;
    run <= '1', '0' after 20 ns;
    funcsel <= "001";
    for i in 0 to MYNMAX-1 loop
        wait until rising_edge(rdy);
        wait for 5 ns;
        if (unsigned(fn) /= to_unsigned(myfasit(i),mywidth)) then
            errorcnt := errorcnt +1;
            --assert (unsigned(fn) = to_unsigned(myfasit(i),mywidth))
            report "Fibonacci number " & integer'image(i) & " incorrect @ time: "
& time'image(now)  &
            " Expected " & natural'image(myfasit(i)) & " Actual " &
natural'image(to_integer(unsigned(fn)))
            severity error;
        end if;
    end loop;

    assert not(errorcnt = 0)
    report "Passed"
    severity note;

    assert (errorcnt = 0)
    report "Failed with " & integer'image(errorcnt) & " errors"
    severity error;
    wait;
  end process;

end testbench;
```

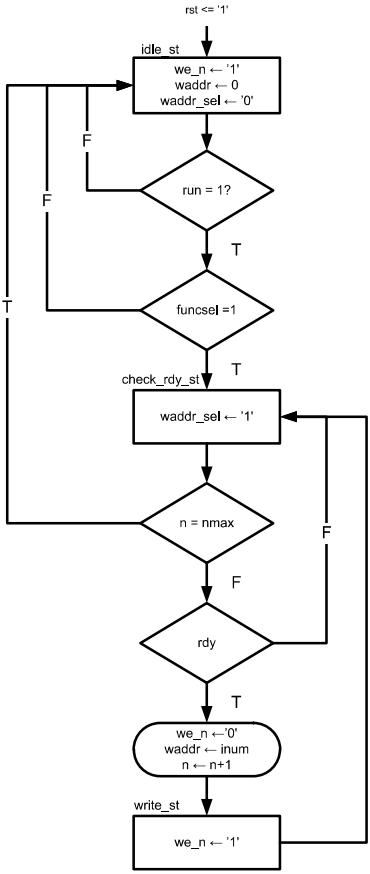


Oppgave 11c)


rst <= '1'
idle_st
f0 ← 0
f1 ← 0
f2 ← 0
n ← 0
F
F
run = '1'
T
funcsel =1
T
calc_st
T
n ≥ 2
T
F
n=1
T
F
f2 ← f1 + f0
f2 ← 1
f2 ← 0
update_st
f0 ← f1
f1 ← f2
n=nmax
F
n ← n+1
rdy


tb_fibonacci_behav.wcfg
Name
Value
rst
clk
rdy
inum[7:0]
fn[15:0]
0
0
1
19
4181
520.000 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns

Oppgave 11d)


rst <= '1'
idle_st
we_n ← '1'
waddr ← 0
waddr_sel ← '0'
F
run = 1?
F
T
T
funcsel =1
T
check_rdy_st
waddr_sel ← '1'
n = nmax
F
F
rdy
T
we_n ←'0'
waddr ← inum
n ← n+1
write_st
we_n ← '1'

Oppgave 11e)

```
tristate_buffer:
process (nwe,wdata)
begin
  if nwe = '0' then
    data <= wdata;
  else
    data <= (others => 'Z');
  end if;
end process;

addressmultiplexer:
process (waddr,raddr,waddr_sel)
begin
  addr <= raddr;
  if waddr_sel = '1' then
    addr <=waddr;
  end if;
end process;
```