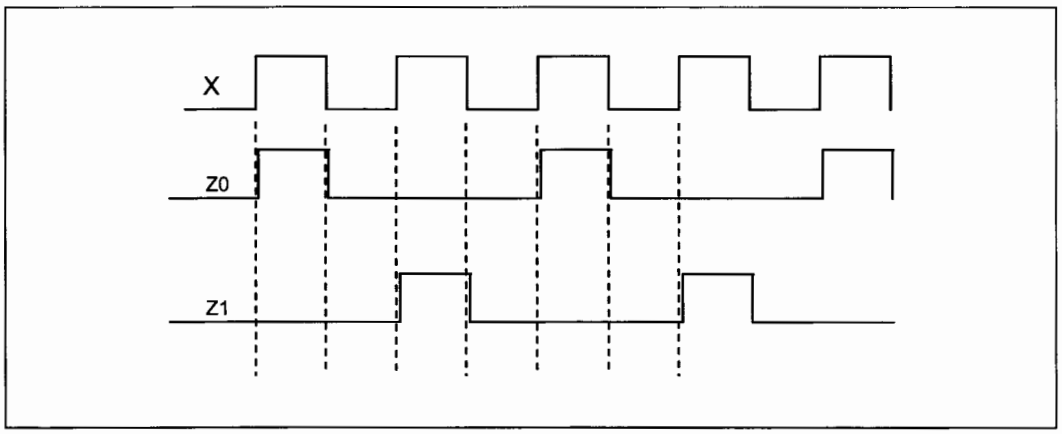
FYSE420 DIGITAL ELECTRONICS

3.06.2011

K.Loberg

1. Toteuta alla esitetyn sekvenssin tuottava asynkroninen piiri. Anna herätefunktiot, siirtotaulukko ja kokonaistilataulukko (**exitation functions, transition table** and **total state table) kyseiselle piirille** . Alleviivaa siirtotaulukon stabiilit tilat. Vältä hasardeja ja muista oikea tilakoodaus. X olkoon tulo, Z0 ja Z1 ovat antoja. Käytä vain logiikka portteja.



2. Selitä lyhyesti

(a) Mealy tilakone. Esitä rakenne lohkokaaviolla.
(b) Moore tilakone. Esitä rakenne lohkokaaviolla.
(c) One-Hot Encoded state machine.
(d) Registered-Output Finite State Machine
(e) Esitä tekijöitä, jotka vaikuttavat digitaalisen piirin tehon kulutukseen.

3. Anna kytkentäkaavio alla olevalle VHDL kuvaukselle.

```
library ieee ;
use ieee.std_logic_1164.all;
entity circuit is
port (
      a, b, c : in std_logic ;
      y : out std_logic_vector (7 downto 0) ) ;
end circuit ;

architecture circuit_arc of circuit is
signal abc : std_logic_vector (2 downto 0) ;
begin
 abc <= a & b & c ;
 with abc select y <=
 "00000001" when "000",
 "00000010" when "001",
 "00000100" when "010",
 "00001000" when "011",
 "00010000" when "100",
 "00100000" when "101",
 "01000000" when "110",
 "10000000" when others ;
end circuit_arc ;
```

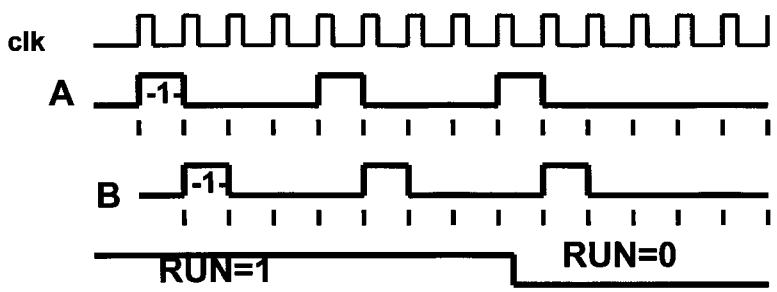
4. Anna kytkentäkaavio alla olevalle VHDL kuvaukselle. Käytössäsi on D-ff, joka on nousevalla reunalla aktiivinen ja sillä on Clock Enable (CE) tulo. CE on ylätila-aktiivinen (level -1-) tulo. Käytä VHDL-kuvauksen mukaisia symboleja.

```
library IEEE;
use ieee.std_logic_1164.all;
entity reg is
port (
        ina : in std_logic_vector(7 downto 0);
        enable : in std_logic;
        clk,start: in std_logic;
        com : out std_logic_vector(7 downto 0)
    );
end reg;

architecture reg_arch of reg is
        signal reg_out : std_logic_vector(7 downto 0);
begin
process (clk) is
        begin
        if (clk'event and clk = '1') then
        if enable ='1' then
                reg_out <= ina;
        end if;
```

```
        end if;
end process;

process(reg_out,start) is
        variable varcom : std_logic_vector(7 downto 0);
begin
        if start = '0' then
                varcom := "00000000";
        else
                varcom := reg_out;
        end if;
        com <= varcom;
end process;
end architecture reg_arch ;
```

5. Anna kytkentäkaavio **liitteessä 1** olevalle VHDL-kuvaukselle. Esitä kytkentä kuvauksen mukaisella rakenteella. Käytä kuvauksen mukaisia symboleja. Mikä kytkentä on kyseessä?

6. Design the synchronous **binary encoded** finite state machine (FSM). The external control input is ***RUN***. If ***RUN*** = ***-1-***, the FSM enters to the continuous sequence described below. When ***RUN*** goes to the ***-0-*** level, the **FSM stop to the state where *A* = *B* = *-0-*. If outputs A or B are -1- when RUN goes to the level : RUN=0, the FSM sequence runs normally until *A* = *B* = *-0-* at the same time**. Let the first state be ***a***, when outputs ***A*** = ***-0-*** and ***B*** = ***-0-*** . Draw only clock connections between DFFs and give the boolean functions of the exitationfunctions of DFFs. Outputs A and B has own DFFs. The solution is Registered Output FSM, which has binary-encoded states.



Tässä on VHDL-tiedosto Dffpavkage.vhd liite 1

```
-----------------------------------------------
-- Package of Dflip-flops for hierarchical ----
--  example1 by Kari Loberg 07.03.2011    ----
-----------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
--
-----------------------------------------------
-- Positive edge triggered D fip-flop --
-----------------------------------------------
entity Dflipflop is
port ( D: in std_logic;
       CLK: in std_logic;
       Q: out std_logic
     );
end entity Dflipflop;
--
-----------------------------------------------
-- Architecture of Dflipflop entity -----
-----------------------------------------------
architecture Dflipflop_arch of Dflipflop is
begin
Dffwait: process is
 begin
   wait until (CLK'event and CLK='1');
   Q <= D;
 end process Dffwait;
end architecture Dflipflop_arch;
--
-----------------------------------------------
--              My package              ---
-----------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
--
package flipflops is
-- components ----------------
component Dflipflop is
 port ( D : in std_logic;
        CLK : in std_logic;
        Q : out std_logic
        );
end component Dflipflop;
end flipflops;
--
-----------------------------------------------
--              End My package          ----
-----------------------------------------------
```

Tässä on VHDL-tiedosto, jossa on kuvattu itse kytkentä liite1

```
------------------------------------------------------------------
---  Hierarchical VHDL example with package      ---
---              flipflops                       ---
---       by K. Loberg 07.03.2011                ---
------------------------------------------------------------------
--
library ieee;
use ieee.std_logic_1164.all;
use work.flipflops.all;
--
-------------------------------------------
--       koetehtävän kytkentä          ---
-------------------------------------------
--
entity circuit is
  port ( d3,d2,d1,d0: in std_logic;
         clock: in std_logic;
         q3,q2,q1,q0: out std_logic
        );
end entity circuit;
--
-----------------------------------------------
--         Architecture for circuit        ---
-----------------------------------------------
--
architecture circuit_arch of  circuit is
-- declarations for local signals (connection wires) ---
signal din_0,din_1,din_2,din_3,clk_in : std_logic;
signal qout_0, qout_1, qout_2,qout_3 : std_logic;
--
begin
u0: Dflipflop port map(D=>din_0, CLK=>clk_in,Q=>qout_0);
u1: Dflipflop port map(D=>din_1, CLK=>clk_in,Q=>qout_1);
u2: Dflipflop port map(D=>din_2, CLK=>clk_in,Q=>qout_2);
u3: Dflipflop port map(D=>din_3, CLK=>clk_in,Q=>qout_3);
--
din_0<=d0;
din_1<=d1;
din_2<=d2;
din_3<=d3;
--
q0<=qout_0;
q1<=qout_1;
q2<=qout_2;
q3<=qout_3;
--
clk_in<=clock;
end architecture circuit_arch;
```