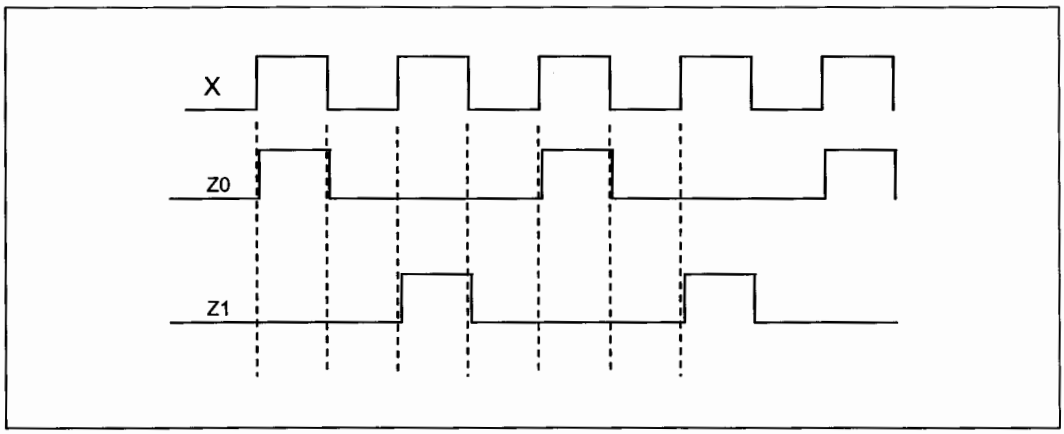
FYSE420 DIGITAL ELECTRONICS

13.05.2011

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.

```
entity comblog is port(
                    sel : in integer range 0 to 3;
                    x,y, z : in std_logic;
                    res : out std_logic);
                end entity comblog;

                architecture behavior_arch of comblog is
                    begin
                    case sel is
                        when 0 => res <= x and y;
                        when 1 => res <= y xor z;
                        when 2 => res <= x nand z;
                        when others => res <= x nor z;
                    end case;
                    end process;
                end architecture behavior_arch;
```

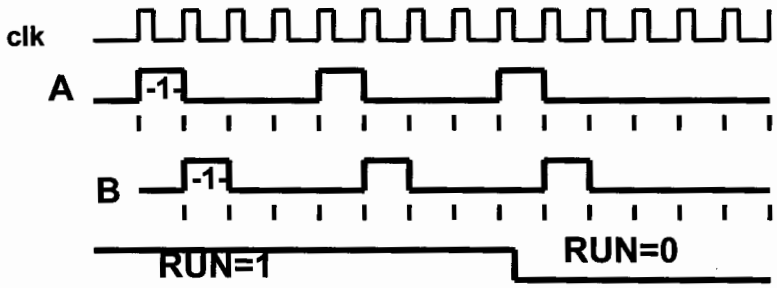
4. Anna kytkentäkaavio alla olevalle VHDL kuvaukselle. Käytössäsi on D-ff, joka on nousevalla reunalla aktiivinen ja sillä on Clock Enable (CE) tulo. CE on ylätila-aktiivinen (level -1-) tulo.

```
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 olevalle VHDL-kuvaukselle. Esitä kytkentä kuvauksen mukaisella rakenteella. 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.



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Comments
--
-------------------------------------------------------------------
--   design  for examination. By KL 03.05.2011  ---------
-------------------------------------------------------------------
-- Comments  ---
--
entity nandgate is
   port (a,b,c : in std_logic;
                y : out std_logic);
end nandgate;

architecture nandgate_arch of nandgate is
begin
   y <= not (a and b and c);
end nandgate_arch;
-----------------------------------------
-- Comments                          --
-----------------------------------------
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity notgate is
   port ( inport : in std_logic;
          outport : out std_logic);
end notgate;

architecture notgate_arch of notgate is
begin
   outport <= not inport;
end notgate_arch;
-----------------------------------------------------
-- Comments.                                      --
-----------------------------------------------------
--
--
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity tentti is
   port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          c : in  STD_LOGIC;
          d0 : out  STD_LOGIC;
          d1 : out  STD_LOGIC;
          d2 : out  STD_LOGIC;
          d3 : out  STD_LOGIC;
          d4 : out  STD_LOGIC;
          d5 : out  STD_LOGIC;
          d6 : out  STD_LOGIC;
          d7 : out  STD_LOGIC);
end tentti;

architecture tentti_arch of tentti is
  component nandgate is
          port (a,b,c : in std_logic;
                   y : out std_logic);
        end component;

        component notgate is
          port (inport : in std_logic;
               outport : out std_logic);
        end component;
-- declare local signals : inva invb and invc  --
  signal inva, invb, invc : std_logic;
begin
-- Comments --------
  U1: notgate port map(a, inva);
  U2: notgate port map(b, invb);
  U3: notgate port map (c, invc);
-- Now we can drive outputs ----
  U4: nandgate port map(inva, invb, invc, d0);
  U5: nandgate port map(a, invb, invc, d1);
  U6: nandgate port map(inva, b, invc, d2);
  U7: nandgate port map(a, b, invc, d3);
  U8: nandgate port map(inva, invb, c, d4);
  U9: nandgate port map(a, invb, c, d5);
  U10: nandgate port map(inva, b, c, d6);
  U11: nandgate port map(a, b, c, d7);
end tentti_arch;
```