INF3430/INF4430 Fasit eksamen Høst 2007 Oppgave 1 – 14

Oppgave	A	B	C	D	E
1	O		O	O	
2		O	O		
3	O	O		O	
4		O		O	
5			O		
6	O			O	
7		O	O		
8	O	O	O		
9		O		O	
10		O	O	O	
11	O			O	
12		O			
13	O	O	O		
14	O	O		O	

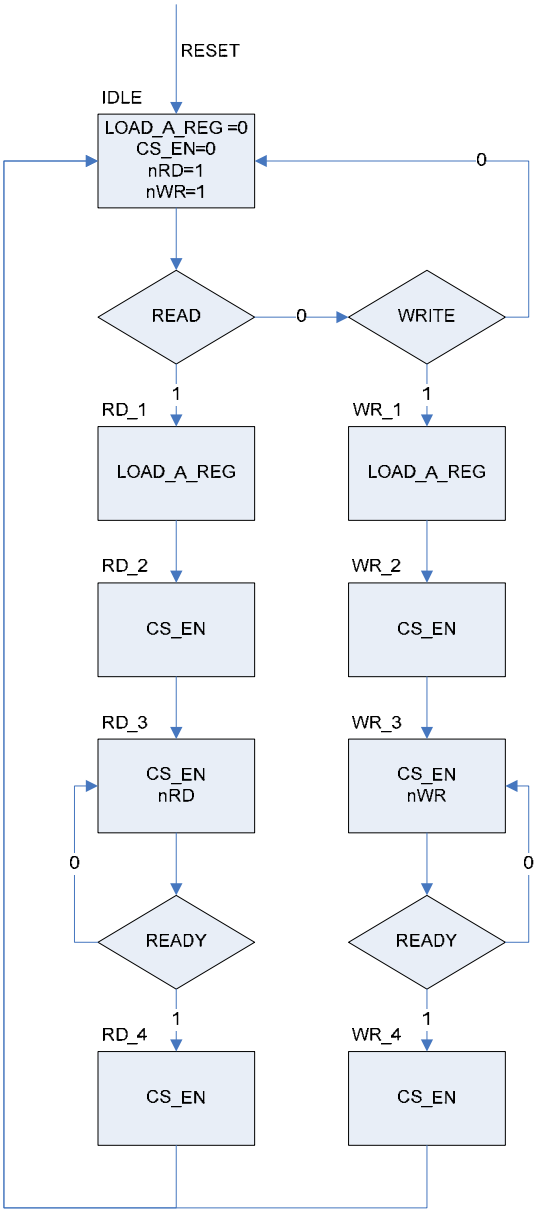
Oppgave 15.

a).

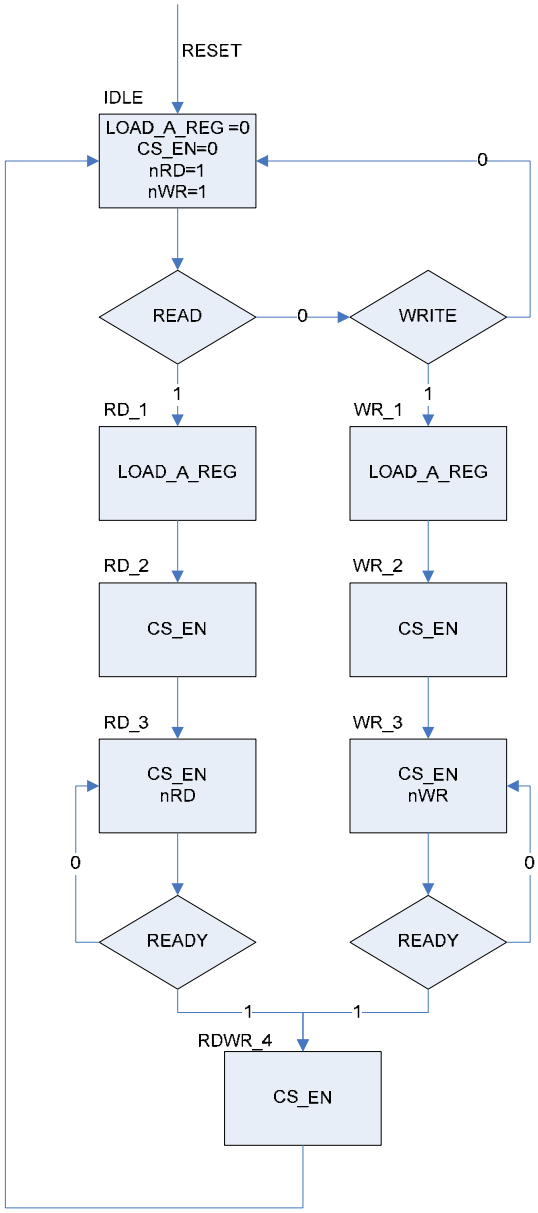
```
--Oppgave 15a).
--==============
AIN <= A(17 downto 16);
--Alt.A
ADDRESS_DECODER:
process(CS_EN,AIN)
begin
  nCS <= "1111";
  if CS_EN = '1' then
    case AIN is
      when "00"   => nCS <= "1110";
      when "01"   => nCS <= "1101";
      when "10"   => nCS <= "1011";
      when others => nCS <= "0111";
    end case;
  end if;
end process;
end;

--Alt.B
ADDRESS_DECODER:
process(CS_EN,AIN)
begin
  nCS <= "1111";
  if CS_EN = '1' then
    if AIN = "00" then
      nCS(0) <= '0';
    elsif AIN = "01" then
      nCS(1) <= '0';
    elsif AIN = "10" then
      nCS(2) <= '0';
    elsif AIN = "11" then
      nCS(3) <= '0';
    end if;
  end if;
end process;
```

b).



Alternativ A



Alternativ B

c).

```
  type IO_CONTROLLER_TYPE is (IDLE, RD_1,RD_2,RD_3,RD_4,
                              WR_1,WR_2,WR_3,WR_4);
--eller
  --type IO_CONTROLLER_TYPE is (IDLE, RD_1,RD_2,RD_3,,
  --                            WR_1,WR_2,WR_3,RDWR_4);
  signal IO_CTRL_ST,IO_CTRL_NEXT_ST   : IO_CONTROLLER_TYPE;

  --Assumes that all signals declared in the entity
begin

IO_CONTROLLER_COMB:
process(READ,WRITE,READY,IO_CTRL_ST)
begin
  nRD     <= '1';
  nWR     <= '1';
  CS_EN   <= '0';
  LOAD_A_REG <= '0';
  IO_CTRL_NEXT_ST <= IDLE;

  case IO_CTRL_ST is
    when  IDLE =>
      if READ = '1' then
        IO_CTRL_NEXT_ST <= RD_1;
      end if;
      if WRITE = '1' then
        IO_CTRL_NEXT_ST <= WR_1;
      end if;
    --Read control
    when RD_1 =>
      LOAD_A_REG <= '1';
      IO_CTRL_NEXT_ST <= RD_2;
    when RD_2 =>
      CS_EN <= '1';
      IO_CTRL_NEXT_ST <= RD_3;
    when RD_3 =>
      CS_EN <= '1';
      nRD <= '0';
      if READY = '0' then
        IO_CTRL_NEXT_ST <= RD_3;
      --Alt A
      else
        IO_CTRL_NEXT_ST <= RD_4;
```

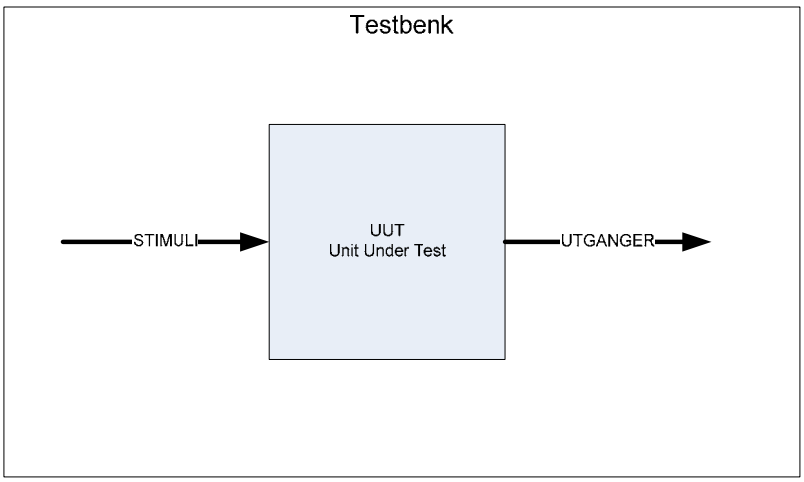
```
      --Alt B
        --IO:CTRL_NEXT_ST <= RDWR_4;
      end if;
    --Alt A
    when RD_4 =>
      CS_EN <= '1';
      IO_CTRL_NEXT_ST <= IDLE;
    --Alt B
    --when RDWR_4 =>
    -- CS_EN <= '1';
    -- IO_CTRL_NEXT_ST <= IDLE;
    --Write control
    when WR_1 =>
      LOAD_A_REG <= '1';
      IO_CTRL_NEXT_ST <= WR_2;
    when WR_2 =>
      CS_EN <= '1';
      IO_CTRL_NEXT_ST <= WR_3;
    when WR_3 =>
      CS_EN <= '1';
      nWR <= '0';
      if READY = '0' then
        IO_CTRL_NEXT_ST <= WR_3;
      else
        --Alt A
        IO_CTRL_NEXT_ST <= WR_4;
        --Alt B
        --IO_CTRL_NEXT_ST <= RDWR_4;
      end if;
    when WR_4 =>
      CS_EN <= '1';
      IO_CTRL_NEXT_ST <= IDLE_ST;

    when others => --Unecessary when using default values
      IO_CTRL_NEXT_ST <= IDLE;
  end case;
end process IO_CONTROLLER_COMB;
```

```
IO_CONTROLLER_STATE_REG:
process (RESET,CLK)
begin
  if RESET = '1' then
    IO_CTRL_ST <= IDLE;
  elsif rising_edge(CLK) then
    IO_CTRL_ST <= IO_CTRL_NEXT_ST;
  end if;
end process IO_CONTROLLER_REG;
```

d).

For å simulere kretsen i oppgave 15c) må man påtrykke inngangene stimuli og sjekke utgangene.



I VHDL gjør man dette ved å lage en testbenk. I sin enkleste er den bygd opp på følgende måte

1. En (vanligvis) tom entitet for selve testbenken. Dvs. testbenken har vanligvis ikke noe interface mot verden utenfor men er ”selfcontained”.
2. Komponentdeklarasjon for UUT (Unit Under Test)
3. Deklarsjon av input stimuli signaler
4. Definisjon av klokke
5. Instantiering av UUT
6. Stimuli process der man lager en sekvens av input signaler
7. Sjekker output i ”Waveform-viewer”

I mer avanserte testbenker kan man istedenfor stimuliprosessen påtrykke inputstimuli ved å benytte simuleringsmodeller av omkringliggende kretser og instantiere disse i testbenken. Selve testbenken kan bli vesentlig enklere på denne måten.

Et annet alternativ er å hente input stimuli fra fil.

En mer avansert måte å sjekke korrekt funksjon er å lage en fasit over forventede verdier på utgangene og sammenligne disse med utgangene av UUT. Fasiten kan man lagre i en egen fil eller inni testbenken.
På denne måten kan testbenken være selvtestende og man kan slippe å studere timingdiagrammer. Man kan bare rapportere om resultatet er OK eller ikke.