Lunds Universitet
LTH
Ingenjörshögskolan
IDA IEA
Helsingborg

Tentamensskrivning 22 april 2014

EDI 610 Digitala system 15 poäng, varav tentamen 4,5 p

Kursansvarig: Bernt-Arne Jönsson Skrivtid 14.00-19.00

Inga hjälpmedel
Obs! Räknare ej tillåten.

Skrivningen omfattar uppgifterna 1-8
Bilaga till skrivningen : Syntaxhjälp till VHDL

Maximalt antal poäng: 60 poäng

Krav för godkänt: 30 p

Ordentliga motiveringar skall lämnas! Det skall gå att följa din tankegång.
Alla lösa blad skall vara samlade i omslagsarket.
Inlämnade uppgifter skall vara försedda med uppgiftens nummer.
Lösningarna skrivs in i nummerordning.
Skriv namn på varje ark
Omslagsarket skall vara fullständigt ifyllt med inskrivningsår, namn och personnummer
Kryssa för lösta uppgifter och ange antalet inlämnade blad.

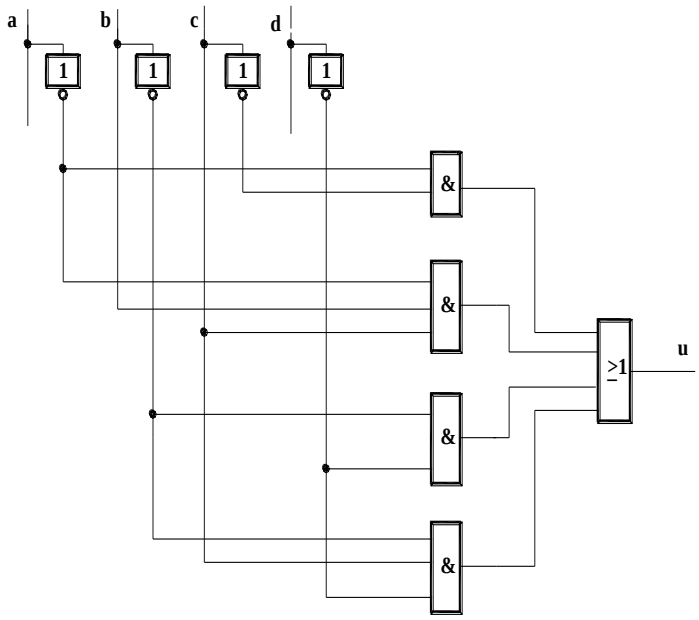
1. Betrakta nedanstående sanningstabell:
x_3,x_2,x_1 och x_0 är logiska insignaler och f är utsignelen

x_3	x_2	x_1	x_0	f
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	1	0	0
1	1	0	0	0

I övriga kombinationer är f=1

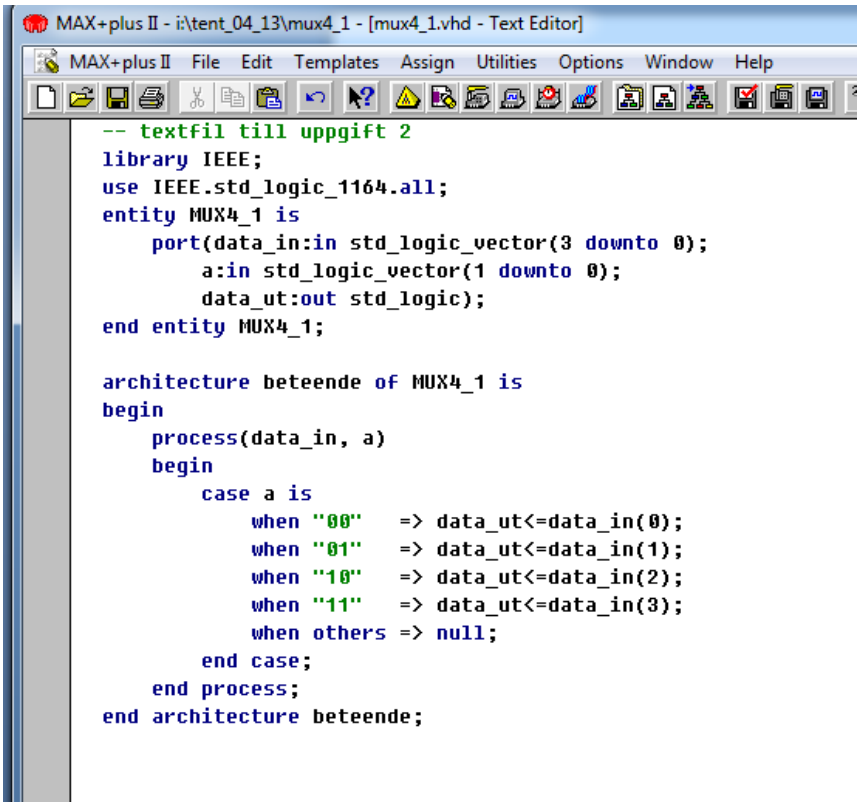
a) skriv upp det förenklade booleska uttrycket för f på sp-form (den form där det logiska uttrycket avslutas med en eller-funktion. Du behöver inte rita upp nätet.
b) skriv upp det förenklade booleska uttrycket för f på ps-form (den form där det logiska uttrycket avslutas med en och-funktion) Du behöver inte rita nätet. (8p)

2.



Ovan ser du ett kombinatoriskt nät med fyra ingångar (a,b,c och d) samt utgången u.
a) Skriv upp det booleska uttrycket för nätet **innan** du förenklat det.
b) förenkla uttrycket. (8p)

3. Konstruera en logisk krets som realiserar nedanstående textfil. Du får använda standardgrindar (AND,NAND,OR, NOR, NOT, XOR)




```
-- textfil till uppgift 2
library IEEE;
use IEEE.std_logic_1164.all;
entity MUX4_1 is
    port(data_in:in std_logic_vector(3 downto 0);
        a:in std_logic_vector(1 downto 0);
        data_ut:out std_logic);
end entity MUX4_1;

architecture beteende of MUX4_1 is
begin
    process(data_in, a)
    begin
        case a is
            when "00"   => data_ut<=data_in(0);
            when "01"   => data_ut<=data_in(1);
            when "10"   => data_ut<=data_in(2);
            when "11"   => data_ut<=data_in(3);
            when others => null;
        end case;
    end process;
end architecture beteende;
```

(7p)

4. a) omvandla 86_{10} till basen 2.(1p)
b) omvandla 86_{10} till basen 16.(1p)
c) omvandla 86_{10} till basen 8.(1p)
d) antag att vi har binära tal med ordlängden 8 bitar, där negativa tal representeras av tvåkomplement.
Skriv tvåkomplementrepresentationen av 86_{10}. (dvs du skall skriva -86, som det representeras med tvåkomplement.(1p)
e) utför operationen 76_{10}-86_{10} i binärkod där negativa tal har tvåkomplementrepresentation. Visa exakt hur du genomfört dina räkningar. Svara sen i decimalkod.(2p)

5. Längre bak i skrivningen ser du en VHDL-fil, som beskriver ett sekvensnät.
a) Rita tillståndsgraf för sekvensnätet.
b) Realisera det med d-vippor. Enkel lösning premieras. Koda tillstånden i vanlig binärkod. Lös först uppgiften utan resetfunktion. Lägg sedan till den på lämpligt vis.(8p)

6. a) Konstruera en n räknare, som räknar i graykod dvs: 000, 001, 011, 010, 110, 111, 101, 100,000…. Använd d-vippor.
b) Inför efter det att du konstruerat räknare en styrsignal set_6, som på en logisk etta gör att nästa tillstånd blir 110. (8p)

7.

komp.vhd - Text Editor

```
-- textfil till uppgift 7
library IEEE;
use IEEE.std_logic_1164.all;

entity komp is
    port(a,b:in std_logic_vector(3 downto 0); u:out std_logic);
end entity komp;
architecture beteende of komp is
begin
    process(a,b)
    begin
        if a=b then
            u<='1';
        else
            u<='0';
        end if;
    end process;
end architecture beteende;
```

Ovan ser du ett VHDL-beskrivningen av kombinatoriskt nät. Realisera nätet med standardkomponenter. Du får även gärna använda XOR- grindar.(8p)

8. Det finns ett antal förkortningar som rör halvledarminne. Här nedan finns några sådana förkortningar. Skriv först ut vad det är förkortning för (dvs. på engelska) och förklara sedan kortfattat vad det innebär.

a) ROM
b) PROM
c) EPROM
d) EEPROM
e) RAM
f) SRAM
g) DRAM
(tot 7p)

VHDL-kod till uppgift 5.

uppg5.vhd - Text Editor

```
use IEEE.std_logic_1164.all;

entity uppg5 is
    port(reset, x, clock:in std_logic;
        u:out std_logic);
end entity uppg5;

architecture beteende of uppg5 is
type state_type is( s0,s1,s2,s3,s4);
signal present_state, next_state:state_type;
begin
    process(x,reset)
    begin
        if reset='1' then
            next_state<=s0;
        else

            case present_state is
                when s0     =>  if x='0' then
                                    next_state<=s0;
                                else
                                    next_state<=s1;
                                end if;
                when s1     =>  if x='0' then
                                    next_state<=s0;
                                else
                                    next_state<=s2;
                                end if;
                when s2     =>  if x='0' then
                                    next_state<=s0;
                                else
                                    next_state<=s3;
                                end if;
                when s3     => if x='1' then
                                    next_state<=s3;
                                else
                                    next_state<=s4;
                                end if;
                when s4     => if x='0' then
                                       next_state<=s0;
                                   else
                                       next_state<=s1;
                                   end if;
            end case;
        end if;
    end process;

    process(present_state)
    begin
        if present_state=s4 then
                if x='0' then
                    u<='1';
                else
                    u<='0';
                end if;
        else
            u<='0';
        end if;
    end process;
    process(clock)
    begin
        if rising_edge(clock)  then
            present_state<=next_state;
        end if;
    end process;
end architecture beteende;
```