Lunds Universitet
LTH
Ingenjörshögskolan
IDA IEA
Helsingborg

Tentamensskrivning 4 maj 2015

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

Kursansvarig: Bernt-Arne Jönsson **Skrivtid 08.00-13.00**

Inga hjälpmedel
Obs! Räknare ej tillåten.

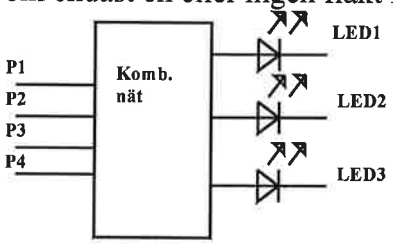
Skrivningen omfattar uppgifterna 1-8
Bilaga till skrivningen: Syntaxhjälp till VHDL

Maximalt antal poäng: 60 poäng

Krav för godkänt: 30 p

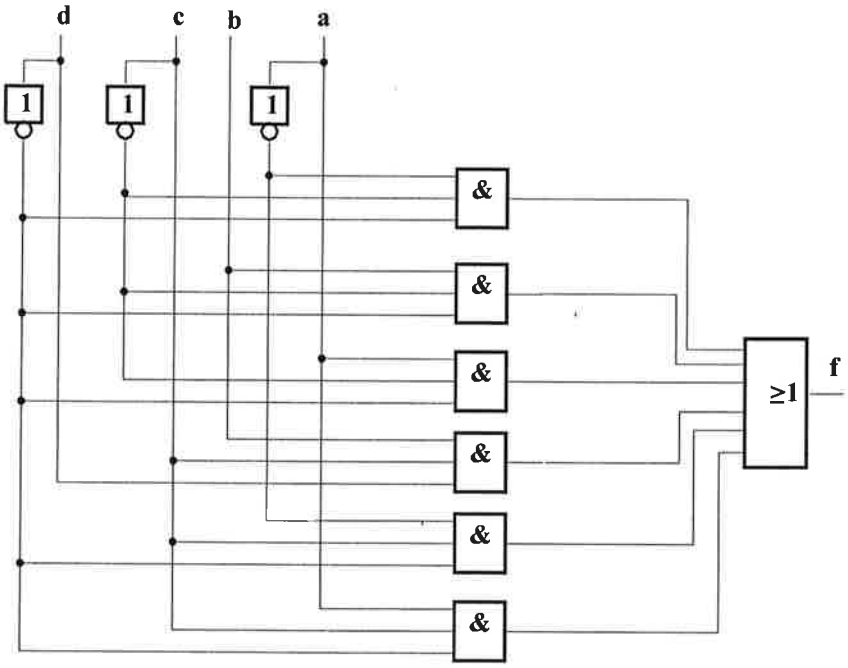
Ordentliga motiveringar skall lämnas.
Alla lösa blad skall vara samlade i omslagsarket.
Inlämnade uppgifter skall vara försedda med uppgiftens nummer.
Lösningarna skrivs in i nummerordning.
Skriv namn på varje ark
Omslagsarket skall vara fullständigt ifyllt med inskrivningsår, namn och personnummer
Kryssa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

1. Fyra stycken fläktar evakuerar luften i en industrilokal. Varje fläkt är utrustad med en givare. Om fläkten fungerar tillfredställande ger givaren en logisk etta annars en nolla. Om fyra eller tre fläktar fungerar tillfredande skall ett kombinatoriskt nät tända en grön lysdiod (LED1). Om däremot endast 2 fläktar fungerar, så skall en gul lysdiod tändas (LED2) och den gröna lysdioden släckas. Slutligen skall en röd lysdiod (LED3) tändas om endast en eller ingen fläkt fungerar.



a) konstruera den del av nätet som tänder LED1 med standardgrindar (inv, and, or, nand nor eller xor.). Nätet skall förenklas.
Obs! du behöver inte rita upp nätet utan bara ange det logiska uttrycket för det.
b) konstruera den del av nätet som tänder LED2 med samma typ av grindar som i a.
c) konstruera den del av nätet som tänder LED3 med samma typ av grindar som i a (tot 9p)

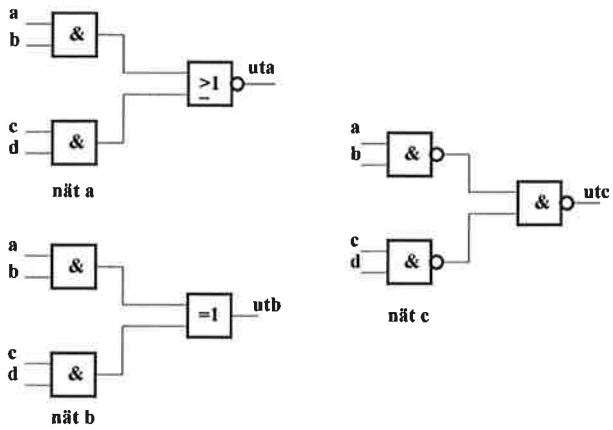
2.



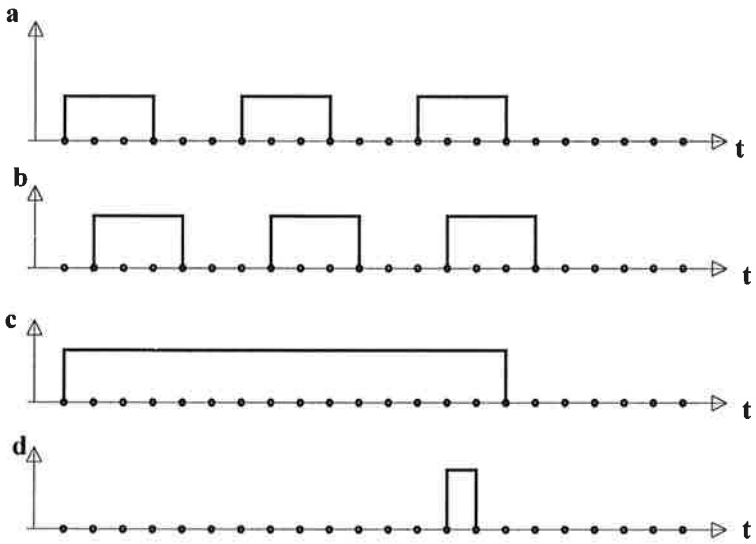
a)Ett enkelt kombinatoriskt nät skulle kopplas upp på ett kopplingsdäck . Det kombinatoriska nätet har 4 insignaler (a, b, c, d) samt utsignalen f. För att minska kopplingsarbetet beslöt man sig för att försöka förenkla kopplingen. Hjälp dem med detta! Rita sedan upp grindnätet för det förenklade nätet (SP-form, dvs ett nät som avslutas med en eller-grind).
b) samma nät som ovan men det skall realiseras som ett nät på PS-form (ett nät som avlsutas med en och-grind) (tot 8p)

3. a) omvandla 92_{10} till basen 2.(1p)
b) omvandla 92_{10} till basen 16.(1p)
c) omvandla 92_{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 92_{10}.(1p)
e) utför operationen 80-92 i binärkod, där negativa tal har tvåkomplementrepresentation. Visa exakt hur du genomfört dina räkningar. Svara sen i decimalkod.(2p)
f) Antag att variablerna a, b och c är åttabitars heltal utan tecken (i programspråket C motsvarar detta unsigned char). Antag vidare att a= 10 och b= 13 och att operationen c =a-b utföres. Vilket värde bör c ha? (tot 8p)

4.



Ovan ser du tre kombinatoriskt nät, där uta, utb och utc är utsignalerna. a, b, c och d är insignalerna, som är samma till de tre näten. Insignalerna visas i form av diagram där inspänningen är en funktion av tiden. Rita de olika utsignaler i ett tidsdiagram!(6p)



tidsdiagram till uppgift 4.

5. Betrakta VHDL-filen i slutet på tentamensuppgifterna.
a) Rita tillståndsgraf som beskriver sekvensnätet.
b) Realisera sekvensnätet med d-vippor. Redovisa din lösningsmetodik (t.ex tabeller och karnaughdiagram). Rita schema. Om det blir mycket att rita kan du nöja dig med att rita kopplingen fullständigt för en vippa.(8p)

6. Komplettera räknaren i uppg. 5 med två funktioner: en enable-funktion och en set_s4-funktion. Enable-funktionen skall vara aktivt hög och även set_s4-funktionen skall vara aktivt hög. set_s4 skall dominera över Enable. Detta innebär att om Enable=0 och set_s4=1, så skall räknaren tillstånd bli s4. set_s4 dominerar även över upp.
OBS! Endast synkrona lösningar.(8p)

7. Räknaren i uppgift 5 blev ganska omfattande att realisera med d-vippor. Speciellt omfattande blev kopplingen när Enable och set_4 skulle implementeras. De beslöt sig därför att komplettera den VHDL-kod som finns i slutet på tentamen med Enable och set_4 funktionerna. Samma villkor skall gälla Enable och set_4, som i uppgift 6.
Hjälp dem att komplettera VHDL-filen!(5p)

8. I dynamiska minne finns två signaler CAS och RAS. Vad är de förkortning för och hur fungerar de?(8p)

```
1   -- enkel modula 6 räknare med upp/nedfunktion
2   library IEEE;
3   use IEEE.std_logic_1164.all;
4
5   entity mod6 is
6     port(upp, clock:in std_logic;
7          q: out std_logic_vector(2 downto 0));
8   end entity mod6;
9
10  architecture beteende of mod6 is
11  type state_type is (s0,s1,s2,s3 ,s4, s5);
12  signal present_state, next_state:state_type;
13  begin
14     process(present_state, upp)
15     begin
16         case present_state is
17              when s0 =>  if upp='1' then
18                          next_state<=s1;
19                     else
20                          next_state<=s5;
21                     end if;
22           when s1  => if upp='1' then
23                          next_state<=s2;
24                     else
25                          next_state<=s0;
26                     end if;
27           when s2  => if upp='1' then
28                          next_state<=s3;
29                     else
30                          next_state<=s1;
31                     end if;
32           when s3  => if upp='1' then
33                          next_state<=s4;
34                     else
35                          next_state<=s2;
36                     end if;
37           when s4  => if upp='1'  then
38                          next_state<=s5;
39                     else
40                          next_state<=s3;
41                     end if;
42           when s5  => if upp='1' then
43                          next_state<=s0;
44                     else
45                          next_state<=s4;
46                     end if;
47        end case;
48     end process;
49
50     process(present_state)
51     begin
52        case (present_state) is
53           when s0  => q<="000";
54           when s1  => q<="001";
55           when s2  => q<="010";
56           when s3  => q<="011";
```

```
57             when s4  => q<="100";
58             when s5  => q<="101";
59          end case;
60       end process;
61
62       process(clock)
63       begin
64          if rising_edge(clock) then
65             present_state<=next_state;
66          end if;
67       end process;
68    end architecture beteende;
69
70
```