Lunds Universitet
LTH
Ingenjörshögskolan
IDA IEA
Helsingborg

Tentamensskrivning 13 januari 2015

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

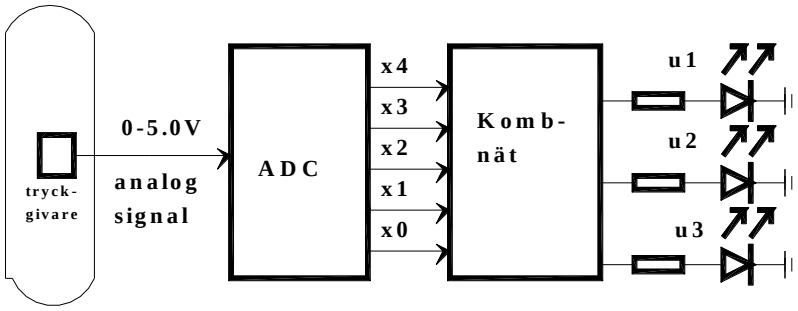
Maximalt antal poäng: 60 poäng

Krav för godkänt: 30 p

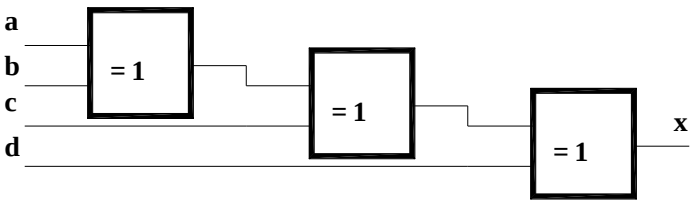
Ordentliga motiveringar skall lämnas.
Alla lösa blad skall vara samlade i omslagsarket.
Inlämnade uppgifter skall vara försedda med uppgiftens nummer.
Lösningarna skrivs in i nummerordning.
Skriv namn på varje ark
Omslagsarket skall vara fullständigt ifyllt med inskrivningsår, namn och personnummer
Kryssa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

1. Ett kombinatoriskt nät har fyra insignaler: a,b,c och d och en utsignal x. x skall bli ett då insignalerna består av minst 3 st. nollor annars skall x bli noll.
a) konstruera nätet 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 nätet med enbart nand-grindar. Här skall du rita upp nätet! (tot 7p)

2. En trycktank för ånga är utrustad med en analog tryckgivare. Spänningsområdet är 0V – 5.0V, där 0V innebär lågt tryck och 5.0V innebär högt tryck. Tryckgivaren skall anslutas till en dator via en 5-bitars analog-digitalomvandlare. Man vill också kunna få en uppfattning om vilket tryck som tanken har genom att ett kombinatoriskt nät tänder olika lysdioder. Utsignalen från analog-digitalomvandlare (x=x4,x3,x2,x1,x0) omfattar således området 0- 31 (binärkod), där 0 är lågt tryck och 31 är högt tryck.
a) För lågt tryck anses föreligga då värdet från analog-digitalomvandlare är 5 eller lägre och då skall u1 vara hög. Tag fram de logiska villkoren, som styr u1.(gul lysdiod)(2p)
b) För högt tryck anses föreligga om värdet från analog-digitalomvandlare är 28 eller högre och då skall u2 vara hög. Tag fram de logiska villkoren som styr u2.(röd lysdiod).(2p)
c) lagom tryck anses föreligga om $6 \leq x \leq 27$. Tag fram de logiska villkoren som styr lysdiod u3 (grön lysdiod)
Obs! Enkla lösningar premieras.(4p)



3.



a,b,c och d är digitala insignaler till det kombinatoriska nätet ovan och x är utsignalen. Ange sanningstabellen.(6p)

4. a) omvandla 93_{10} till basen 2.(1p)
b) omvandla 93_{10} till basen 16.(1p)
c) omvandla 93_{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 93_{10}.(1p)
e) utför operationen 80-93 i binärkod, där negativa tal har tvåkomplementrepresentation. Visa exakt hur du genomfört dina räkningar. Svara sen i decimalkod.(2p)

5. En enkel räknare, som räknar i binärkod skall konstrueras. Den har styrsignalen *upp*. Om *upp*=1 så är sekvensen 0,1,2,3,4,0 …osv. Då upp=0 så blir naturligtvis sekvensen 4, 3,2,1,0,4…osv.
a) Realisera räknaren med d-vippor. (räknarens funktion framgår kanske tydligare med den VHDL-fil som finns efter tentamensuppgifterna.). Du skall således ta fram de kombinatoriska nät som styr de olika vipporna. Redovisa din lösningsmetod (t.ex tabeller, karnaughdiagram)
Obs! Det kan bli ett ganska omfattande nät att rita hela nätet. Du kan därför nöja dig med att rita upp nätet fullständigt för en d-vippa (den som t.ex. har enklast nät).
b) Komplettera din lösning i uppgift a med en Enable funktion och set_4 funktion, båda aktivt höga. D.v.s. då Enable= 1 och då set_4=0. Fungerar den räknaren som i uppgift a. Då Enable=1 och set_4=1 blir nästa tillstånd 4. Enable dominerar över set_4, vilket innebär att om Enable= 0 och set_4=1, så blir räknarens tillstånd oförändrad.
Du skall använda dig av standardgindar (and, nand, or, nor, not, xor) för att implementera Enable och set_4.(tot. 9p)

6. 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 5. Hjälp dem att komplettera VHDL-filen!(5p)

7. En sekvensdetektor Sdet10111 av typ Moore med insignalerna x och clock samt utsignalen u skall konstrueras. Sekvensdetektorn skall detektera förekomst av delsekvenser 10111 i en godtyckligt lång insekvens. Utsignalen skall vara 0 i

starttillståndet och förbli 0 tills sekvensen 10111 då utsignalen skall bli ett, för att sedan bli 0 i nästa klocksignal. Överlappande sekvenser är inte tillåtna. Det innebär att delar av den sekvens som gett en 1:a inte räknas in i den nya sekvensen.

ex:

x: 000001011101110000010111100

u: 000000000100000000000000100

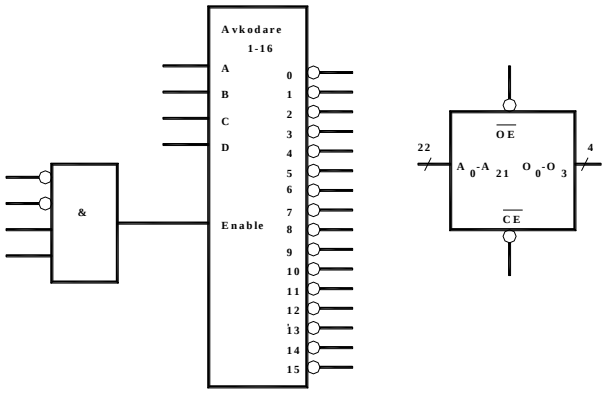
utsignalen är förskjuten ett steg åt höger (Moore-nät), enligt lärobokens sätt att ange sekvenser.

a) Rita tillståndsdiagram (tillståndsgraf).(3p)

b) Koda tillstånden binärt. Realisera sedan kretsen med d-vippor.

Obs! Du kan nöja dig med att rita det kombinatoriska nätet, som styr en vippa fullständigt och enbart skissa nätet till övriga vippor.(6p)

8.



I ett minnessystem ingår bl.a ett läsminne. Minnet skall byggas upp med minneskapslar (4Mx4), se symbolen ovan. Minnessystemet adresseras med 30 adressbitar A0- A29, sålunda omfattande adressområdet 00000000 – 3FFFFFFF. Läsminnet skall ha en kapacitet om 14 Mord á 8 bitar och vara placerat med lägsta adressen 27000000_{16}. Chipenable skall generas av avkodaren. Avkodarens Enable är aktivt hög. Rita blockschema för läsminnet och använd symbolerna ovan! Givetvis skall de mest signifikanta adressbitarna anslutas till och-grinden, som är ansluten till Enable. Ange på avkodarens samtliga utgångar inom vilket adressområde den ger chipenable. Du skall också ange hur de 30 adressbitarna skall anslutas.(10p)

VHDL-kod till uppgift 5a.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.all;

entity counter_5 is
    Port ( upp : in  STD_LOGIC;
           clock : in  STD_LOGIC;
           q : out  STD_LOGIC_VECTOR (2 downto 0));
end counter_5;

architecture Behavioral of counter_5 is
subtype state_type is integer range 0 to 4;
signal present_state, next_state:state_type;
begin
   process(present_state,upp)
   begin
      if upp='1' then
         if present_state=4 then
            next_state<=0;
         else
            next_state<= present_state+1;
         end if;
      else
         if present_state=0 then
            next_state<=4;
         else
            next_state<=present_state-1;
         end if;
      end if;
   end process;
   q<=conv_std_logic_vector(present_state,3);

   process(clock)
   begin
      if rising_edge(clock) then
         present_state<=next_state;
      end if;
   end process;
end Behavioral;
```