Lunds Universitet
LTH
Ingenjörshögskolan
IDA IEA
Helsingborg

Tentamensskrivning 14 december 2012

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

Kursansvarig: Bernt-Arne Jönsson Skrivtid 8.00-13.00

Inga hjälpmedel
Obs! Räknare ej tillåten.

Skrivningen omfattar uppgifterna 1-8
Bilaga till skrivningen : Datablad över räknaren 74163

Maximalt antal poäng: 60 poäng

Krav för godkänt: 30 p

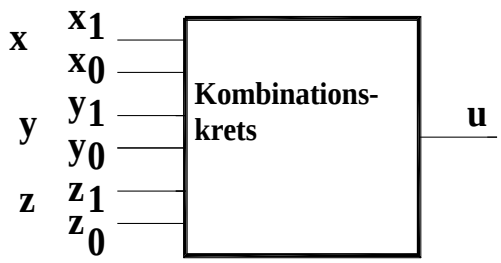
Ordentliga motiveringar skall lämnas.
Alla lösa blad skall vara samlade i omslagsarket.
Inlämnade uppgifter skall vara försedda med uppgiftens nummer.
Lösningarna skrivs in i nummerordning.
Skriv namn på varje ark
Omslagsarket skall vara fullständigt ifyllt med inskrivningsår, namn och personnummer
Kryssa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

1. Konstruera en logisk krets som har fyra insignaler a, b ,c d och där utsignalen f är låg(=0) för kombinationerna i tabellen nedan. I övriga kombinationer är utsignalen 1. Konstruera nätet som ett SP nät. D.v.s. summa av produkter (den formen som avslutas med en eller-grind (5p)

a	b	c	d	f
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	1	0	0
1	1	1	1	0

b) Konstruera nätet med enbart NAND-grindar (3p)

2



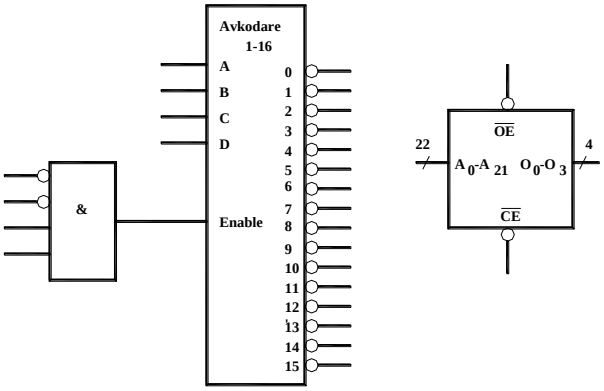
Ett kombinatoriskt nät skall konstrueras, som jämför tre tal x=(x1,x0), y=(y1,y0) och y=(y1,y0) med avseende på relationen 'mindre än' skall konstrueras. För utsignalen u skall gälla att u=1 om och endast om x<y<z (**OBS!** sträng olikhet.). Bestäm minimal SP-form till utsignalen u.(SP summa av produkter).(7p)

3. a) omvandla 93_{10} till basen 2.(1p)
b) omvandla 93_{10} till basen 16.(1p)
c) antag att vi har binära tal med ordlängden 8 bitar, där negativa tal representeras av tvåkomplement.
Skriv tvåkomplementrepresentationen av 93_{10}.(2p)
d) utför operationen $83_{10} - 93_{10}$ i binärkod där negativa tal har tvåkomplementrepresentation. Visa exakt hur du genomfört dina räkningar. Svara sen i decimalkod.(2p)

4. a)En räknare som räknar i BCD-kod skall konstrueras. Du skall använda den redan färdiga modulo16 räknaren 74163. Datablad bifogas. Räknaren skall räkna 0-99. Detta innebär att du behöver 2 st 74163, där varje 74163 räknar 0-9. BCD betyder Binary Coded Decimal, dvs varje decimal siffra har fyra 'bitar'-.Konstruera räknaren. När du ritar ditt kopplingsschema, använd gärna den logiska symbolen (se datablad, fig 2, sid 1)). OBS! lösningen skall givetvis vara helsynkron. (6p)
b) samma som ovan men nu skall räknaren räkna 0 –999, allt i BCD –kod. Då behöver du givetvis 3 st. 74163.(2p)

5. a) Två ingenjörsstudenter tänker programmera ett sekvensnät enligt bifogade VHDL-fil. (Sista sidan)Tyvärr visar det sig när de skall programmera kretsen, så är deras licensfil inte längre giltig. De beslutar sig då att bygga den med traditionell teknik, Och för att det skall bli så enkelt som möjligt beslutar de sig att bygga den med T-vippor. Hjälp dem med detta. Givetvis skall det vara en synkron lösning.(7p)
b) Hjälp dem även att till ovanstående koppling komplettera med Enable, aktivt hög (dvs. då Enable=1, så fungerar den som i uppgift a) men då Enable= 0, så står sekvensnätet i samma tillstånd, oberoende av klockpuls. Även här synkron lösning.(3p)

6



I ett minnessystem ingår bl.a ett läsminne. Minnet skall byggas upp med minneskapslar (4Mx4), se symbolen ovan. Minnessystemet adresseras med 28 adressbitar A0- A27, sålunda omfattande adressområdet 0000000 – FFFFFFF. Läsminnet skall ha en kapacitet om 16 Mord á 8 bitar och vara placerat i adressområdet E000000 –EFFFFFF. Chipenable skall generas av avkodaren., till vilket skall anslutas adressbitarna A22-A27, så att den kan generera chipselect hela adressområdet C000000 – FFFFFFF. Avkodarens Enable är aktivt hög. Rita blockschema för läsminnet och använd symbolerna ovan! Ange på avkodarens samtliga utgångar inom vilket adressområde den ger chipenable. (8p)

7. a)En sekvensdetektor, typ Moore, med insignalerna x och clock samt utsignalen u skall konstrueras. Utsignalen skall förbli 0 i starttillståndet och förbli 0 tills sekvensen 11011 uppträder och då skall utsignalen bli 1 för att bli 0 i nästa klockpulsintervall. Innan utsignalen åter blir 1, måste en ny delsekvens detekteras. Dvs, överlappande sekvenser är inte tillåtna.

x:001101111010110110110000
u:000000010000100000010000
OBS! u är förskjuten åt höger, eftersom det är ett Moore-nät.
Konstruera nätet med D-vippor. Koda tillstånden binärt.I din lösning bör finnas tillståndsdiagram (tillståndsgraf), tillståndstabell, eventuella karnaugh-diagram. Rita sen logikschema, men eftersom det blir ett ganska mycket ritande kan du nöja dig med att enbart rita det kombinatoriska nätet till D-vippen, som ger nästa tillstånd, som du tycker verkar enklast. Till de övrga vipporna kan du nöja dig med principritning. Du får antaga att nätet är i startillstånd. (7p)

b) Konstruera en Reset till nätet ovan. Låt Resetfunktionen vara aktivt hög, dvs då Reset=1, så återgår sekvensnätet i starttillståndet.(2p)

8. Se uppgift 5. Efter det att de båda ingenjörstudenterna framgångsrikt löst uppgiften, realiserad med T-vippor, erhöll de en gällande licensfil till sin VHDL-programvara. Hjälp dem nu med att implementera Enable-funktionen i VHDL-filen, så att de får en VHD- applikation med en Enable, som i uppgift 5 b.(4p)

```
--räknare som skall byggas med T-vippor
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
entity t_counter is
		port(clock, mod_8:in std_logic;
		max, min: out std_logic; q:out std_logic_vector(3 downto 0));
end entity t_counter;
architecture beteende of t_counter is
subtype state_type is integer range 0 to 15;
signal present_state, next_state:state_type;
begin
		process(present_state, mod_8)
		begin
				if present_state=7 and mod_8='1' then
						next_state<=0;
				elsif present_state=15 then
						next_state<=0;
				else
				next_state<=present_state+1;
				end if;
		end process;
		process(present_state)
		begin
				if present_state=15 then
						max<='1';
				else
						max<='0';
				end if
				if present_state=0 then
						min<='1';
				else
						min<='0';
				end if;
		end process;
		q<=conv_std_logic_vector(present_state,4);
		process(clock)
		begin
				if rising_edge(clock) then
						present_state<=next_state;
				end if;
		end process;
end architecture beteende;
```