Tentamen i
Digitala system - EITA15 15hp
varav denna tentamen 4,5hp

Institutionen för elektro- och informationsteknik
Campus Helsingborg, LTH

2018-01-09 8.00 - 13.00 (förlängd 14.00)

Uppgifterna i tentamen ger totalt 60 poäng. Uppgifterna är inte ordnade på något speciellt sätt. Läs därför igenom alla uppgifter innan du börjar lösa dem. Några uppgifter är uppdelade i deluppgifter. Av totalt 60 möjliga poäng fordras minst 30 för godkänt.

Betygsgränser:

- 30p - 39p ger betyg 3
- 40p - 49p ger betyg 4
- 50p - 60p ger betyg 5

Inga hjälpmedel är tillåtna

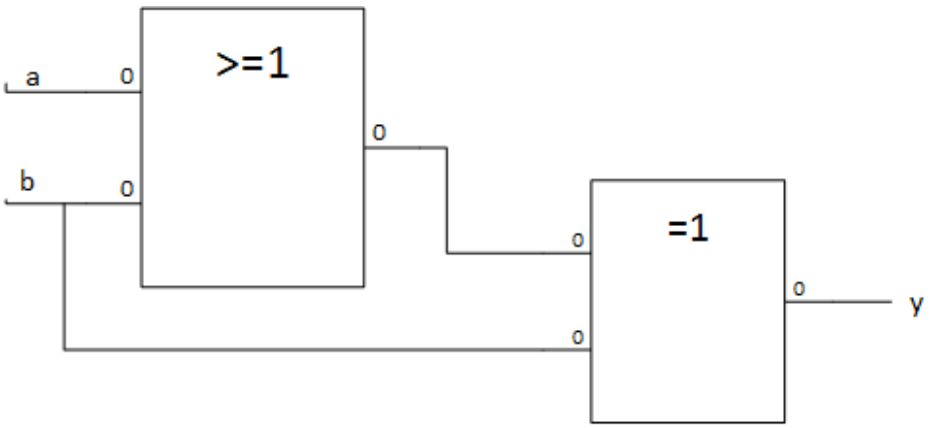
Observera!

- För att rättning av lösning skall komma i fråga fordras att den är läslig samt klart och tydligt uppställd.
- Glöm inte att skriva personlig identifierare på varje blad.
- Alla lösa blad ska vara samlade i omslaget.
- Lösningarna ska vara numrerade och ordnade i nummerföljd.
- Påbörja ny uppgift på nytt papper.

Lycka till!

1. Denna uppgift består av fem påstående. Besvara varje påstående om det är sant eller falskt. Varje rätt påstående ger +2 poäng medan fel svar ger -1 poäng. Maximala antalet poäng på denna uppgift är 10 och minsta poäng 0. (inga minuspoäng totalt!)

   (a) nedanstående sanningstabell gäller för nedanstående grindnät.



a	b	y
0	0	0
0	1	1
1	0	0
1	1	0

Figur 1: grindnät = tabell

   (b) $\mathtt{f}_1 = \mathtt{a}'\mathtt{c}' + \mathtt{c}'\mathtt{d} + \mathtt{a}\mathtt{b}'\mathtt{d}$ och $\mathtt{f}_2 = (\mathtt{a} + \mathtt{c}')(\mathtt{b}' + \mathtt{c}')(\mathtt{a}' + \mathtt{d})$ är lika.

   (c) Det binära talet är ett åtta-bitars tal representerat i två-komplement. Gäller likheten? $10101011_2 = -43_{10}$

   (d) Med kodningen 'one-hot' ändras endast en tillståndsbit åt gången.

   (e) asynkron reset är en resetsignal som är oberoende av klocksignalen.

2. Ett nät är beskrivet i VHDL-kod i slutet av tentamensskrivningen.

   (a) Rita tillståndsgrafen för sekvensnätet. (5 p)

   (b) Skriv sekvensmaskinen på minimal SP-form. (5 p)

3. En modulo 6 räknare med en insignal x vars beteende är, x=0 räkna +3 och om x=1 räkna -2.
   Realisera en tillståndsmaskin på SP-form för ovanstående räknare. Ingående funktioner skall vara realiserade på minimal form. Uppgiften skall vara tydligt dokumenterad med alla stegen väl dokumenterade för att få full poäng. (10 p)

4. Ascii är en teckenkodning som används för att representera bokstäver och tecken i datorn. Tabell 1 längs bak är en tabell över en 7 bitars internationell standard. Exempelvis kan man ur denna tabell avläsa att koden för **A** är 41 hexadecimalt (65 decimalt). Dessa 7 bitar är insignaler till ett nät, som har till uppgift att avgöra om en siffra (0 till 9) dyker upp. Då detta inträffar ska en lysdiod tändas med hjälp av en logisk etta.
   Olle får i uppdrag att konstruera detta nät som tänder en diod då insignalerna till nätet är en siffra. Olle får det goda rådet att angripa problemet med att, på lämpligt sätt, dela upp nätet i två olika delnät , då 7 insignaler blir för mastigt att hantera.
   Hur ser hela nätet ut (i SP-form) som Olle kom fram till? Ingående funktioner skall vara realiserade på minimal form. Uppgiften skall vara tydligt dokumenterad med alla stegen väl dokumenterade för att få full poäng. (10 p)

5. En programmerare har skrivit koden: A=B+C och med hjälp av en kompilator har följande Assembly-instruktioner genererats:

LOAD	R1, B	//R1 ← B
LOAD	R2, C	//R2 ← C
ADD	R1, R2	//R1 ← R1 + R2
STORE	A, R1	// A ← R1

Där A, B, och C är minnesplatser för lagring av A, B och C.

(a) För Assemblyprogrammet ovan, beräkna exekveringstid om exekveringen görs på en icke-pipelinad processor där minnesläsning/skrivning tar 100 ns, registerläsning/skrivning tar 1 ns och ALU operationer tar 1 ns. (3 p)

(b) Om komiplatorn i exemplet använt direkt adressering, beräkna exekveringstid om exekveringen görs på en icke-pipelinad processor där minnesläsning/skrivning tar 100 ns, registerläsning/skrivning tar 1 ns och ALU operationer tar 1 ns. (3 p)

(c) En programmerare har skrivit koden: A=(B+C)-(D+E). Visa hur Assembly koden skulle kunna se ut om all data (A-E) ligger på minnesplatserna A-E och processorn använder ett 2-adress format. (4 p)

6. (a) Antag ett datorsystem med ett primärminne som består av 64 bytes. Hur många bitar behövs för att specificera primärminnets adressrymd om minnet är byte-adressbart? (2 p)

(b) Vad bestämmer/påverkar lokalitet av referenser (locality of references)? (2 p)

(c) Vilka två olika lokaliteter finns? (2 p)

(d) Förklara de två lokaliteterna? (2 p)

(e) Vad är skillnaden mellan write-through och write-back? (2 p)

VHDL-kod till uppgift 2

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity uppgift is
    Port ( reset,x,clk : in STD_LOGIC;
           u : out STD_LOGIC);
end uppgift;

architecture beteende of uppgift is
constant s0 : std_logic_vector(2 downto 0) :="000";
constant s1 : std_logic_vector(2 downto 0) :="001";
constant s2 : std_logic_vector(2 downto 0) :="011";
constant s3 : std_logic_vector(2 downto 0) :="010";
constant s4 : std_logic_vector(2 downto 0) :="110";
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;
      --    cont next page
```

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

end beteende;
```

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Tabell 1: Ascii tabell