LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Tentamen
EDAA05 – Datorer i system

2012–10–22, 8.00–13.00

Tillåtna hjälpmedel: *bifogad formel- och symbolsamling*.

För godkänt betyg på tentamen krävs minst 20 poäng av totalt 30 möjliga.

Börja med att fylla i personuppgifter och övrig om kurs/datum på det utdelade tentamensomslaget. Ha legitimation i beredskap för identitetskontroll. Påbörja därefter tentamen. Skriv dina svar på separat papper. Det går bra att skriva lösningarna till flera uppgifter på samma papper. Märk varje papper med dina initialer i överkanten. När du är klar lämnar du in dina lösningar i det utdelade tentamensomslaget. Behåll gärna detta papper med uppgifterna om du vill.

Lycka till!

---

1. Moores lag

   Moores lag har i många år beskrivit hur antalet transistorer på ett chip har ökat efter hand som kretsarna har gjorts mindre. Detta har i sin tur även lett till att datorernas snabbhet ökat i motsvarande grad. Nu börjar vi dock närma oss de fysikaliska gränserna för hur små enskilda kretsar kan göras och datoringenjörerna står inför ett dilemma hur de ska kunna fortsätta konstruera allt snabbare datorer. Nämn en vanlig strategi som används för att åstadkomma snabbare datorer och som inte bygger på en fortsatt miniatyrisering av kretsarna.

   (1p)

2. Olika talbaser och aritmetik.

   a) Man kan ibland tala om ett binärt tals *minst signifikanta* respektive *mest signifikanta* siffra. Förklara dessa båda begrepp kortfattat, t.ex. genom ett exempel.

   (2p)

   b) Skriv det decimala talet 187 som ett binärt tal och därefter som ett hexadecimalt tal.

   (2p)

   c) Skriv det decimala talet -35 som ett 8-bitars binärt tal i tvåkomplementsform.

   (2p)

   d) Skriv det hexadecimala talet $5D$ i decimalform.

   (1p)

   e) Utför *subtraktionen* $10101101 - 1110010$. Svara genom att ställa upp talen på formen

   ```
     xxxx
   - yyyy
   ------
   ```

   och utför subtraktionen så att svaret framgår tillsammans med vilka lån som gjorts.

   (2p)

---

3. Teckenkodning

   a) Vilka av nedanstående fyra påståenden är sanna?

      1) Om vi vet att en textfil är kodad enligt UTF-8 vet vi också att radsluten är representerade med ett enkelt tecken med teckenkoden 10 (hexadecimalt 0A, ””, ”linefeed”).
      2) UTF-8 stämmer överens med 7-bitars amerikansk ASCII för teckenkoder lägre än 128.
      3) Det finns alltid en s.k. BOM (Byte Order Mark) i början på alla textfiler kodade i enlighet med Unicode-standarden.
      4) ISO 8859 är en internationell familj av standarder som bygger på den amerikanska ASCII-standarden, men använder den åttonde biten för att kunna representera ett antal nationella tecken - olika för olika regioner.

   (2p)

   b) Inom Unicode-familjen hittar vi teckenkodningar såsom UTF-8, UTF-16 och UTF-32. Förklara kortfattat, med några få meningar, hur dessa skiljer sig åt samt vilka fördelar/nackdelar de olika varianterna har.

   (2p)

4. Boolesk algebra

   a) Använd sanningstabeller för att avgöra om nedanstående likhet gäller. Svara genom att redovisa dina sanningstabeller samt motivera ditt svar utgående från dem.
   $x \wedge (y \vee z) = x \wedge \neg y \vee y \wedge \neg z$

   (1p)

   b) Förenkla nedanstående booleska uttryck så långt det går med hjälp av räknelagarna i den bifogade formelsamlingen. Redovisa varje delsteg i förenklingen så att det framgår vilken räknelag som använts (du behöver *inte* skriva ut namnen på lagarna).
   $x \vee \neg(y \wedge x)$

   (2p)

   c) Logik-Lars har gjort följande förenkling av uttrycket $x \wedge y \vee x \wedge (y \vee z)$:

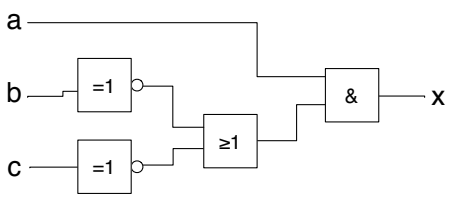
      1) $x \wedge y \vee x \wedge (y \vee z)$
      2) $x \wedge y \vee x \wedge y \vee z$
      3) $x \wedge y \vee z$

      Tyvärr har ett fel smugit sig in i Lars förenkling. Beskriv vad Lars har gjort för misstag samt vad resultatet av förenklingen egentligen borde blivit.

   (2p)

5. Booleska uttryck och digitala grindar

   a) Skriv ett booleskt uttryck för signalen x i figuren till höger. Svara med en funktion av insignalerna a, b och c. Uttrycket ska ej förenklas.

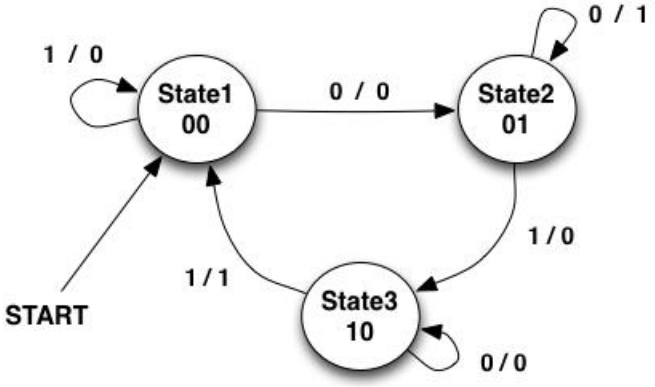


   (2p)

   b) Rita med hjälp av grindsymboler (se den bifogade grindsymbolsamlingen) upp ett kombinatoriskt nät som realiserar det booleska uttrycket $(a \vee b) \wedge c \vee \neg d$ (förenkla inte uttrycket).

   (2p)

6. Realisering av tillståndsmaskin
   Roger och Jonas har designat en tillståndsmaskin med tre tillstånd enligt figuren nedan. Tillståndsmaskinen har en insignal, *i* och en utsignal, *o*. Roger och Jonas har dessutom numrerat tillstånden från 00 till 10 (binärt). Det finns inget tillstånd kallat 11, men om den färdiga maskinen av någon händelse ändå skulle råka hamna i detta tillstånd (t.ex. när strömmen slås på) vill vi att maskinen genast går över till tillstånd 00 (med utsignalen 0).



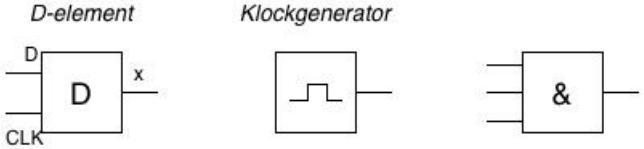
   Nedanstående sanningstabell beskriver hur utsignalen och tillståndsövergångarna ska ske i tillståndsmaskinen ovan. Givet insignalen, *i*, och det nuvarande tillståndet, representerat av *s1* och *s2*, visas vad utsignalen, *o* och det nya tillståndet (*s1'* och *s2'*) ska bli.

i	s1	s2	o	s1′	s2′
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

   a) Hjälp Roger och Jonas att ta fram booleska uttryck för utsignalen, *o*, samt de nya tillstånden, *s1'* och *s2'*. Dvs, skriv booleska funktioner (uttryckta i *i*, *s1* och *s2*) enligt:
   $o = ...$
   $s1' = ...$
   $s2' = ...$

   (3p)

   b) Hjälp Roger och Jonas att rita upp hur den färdiga tillståndsmaskinen ska realiseras med hjäp av digitala grindar. Förutom grindsymbolerna i formelsamlingen kan symbolerna för D-element och klockgenerator vara användbara (se vänster/mitten nedan). Du kan även anta att AND respektive OR-grindar kan ha mer än två ingångar (se exemplet till höger nedan).



   (4p)

Slut!

Formelsamling och grindsymbolförteckning

$x \vee 0 = x$
$x \wedge 1 = x$

$x \vee 1 = 1$
$x \wedge 0 = 0$

$x \wedge \neg x = 0$
$x \vee \neg x = 1$

$x \vee y = y \vee x$
$x \wedge y = y \wedge x$

$x \vee (y \vee z) = (x \vee y) \vee z$
$x \wedge (y \wedge z) = (x \wedge y) \wedge z$

$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

$\neg(x \vee y) = \neg x \wedge \neg y$
$\neg(x \wedge y) = \neg x \vee \neg y$