Tentamen
EDAA05 Datorer i system
Lösningsförslag

2013–10–21, 08.00–13.00

---

1. Moores lag säger att antalet transistorer på ett chip fördubblas ungefär vartannat år. Varianter av lagen förekommer, t.ex. anges fördubblingstakten ibland till var 18:e månad.

2. a) 111010 och 3A

   b) 10100111

   c) 169

   d)
   ```
      11111
      10101101
   +  1110010
   --------------
    100100011
   ```

   e)
   ```
     x   xx
     11011100
   -   100011
   --------------
     10111001
   ```

   ”x” markerar positioner från vilka lån gjorts.

3. a) Påstående *iii)* och *iv)* är sanna.

   b) Eftersom svenska tecken representeras som en enda byte och denna byte har ett värde över 127 (8-bitarskodning) bör det vara fråga om ISO8859-1.

   c) Radslut representeras med två byte - carriage return (0d hexadecimalt) följt av linefeed (0a hexadecimalt).

   d) Windows.

4. a) Sanningstabellerna blir...

x	y	z	$x \wedge y \vee x \wedge (y \vee z)$	$x \wedge z \vee x \wedge \neg z \wedge y$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Eftersom sanningstabellerna för de två uttrycken är lika gäller likheten.

b) $x \wedge \neg y \vee \neg(z \vee \neg y) \vee z \wedge y =$ *(De Morgans lag)*
$x \wedge \neg y \vee \neg z \wedge y \vee z \wedge y =$ *(kommutativa lagen)*
$x \wedge \neg y \vee y \wedge \neg z \vee y \wedge z =$ *(distributiva lagen)*
$x \wedge \neg y \vee y \wedge (\neg z \vee z) =$ *(inverslagen)*
$x \wedge \neg y \vee y \wedge 1 =$ *(identitetslagen)*
$x \wedge \neg y \vee y$

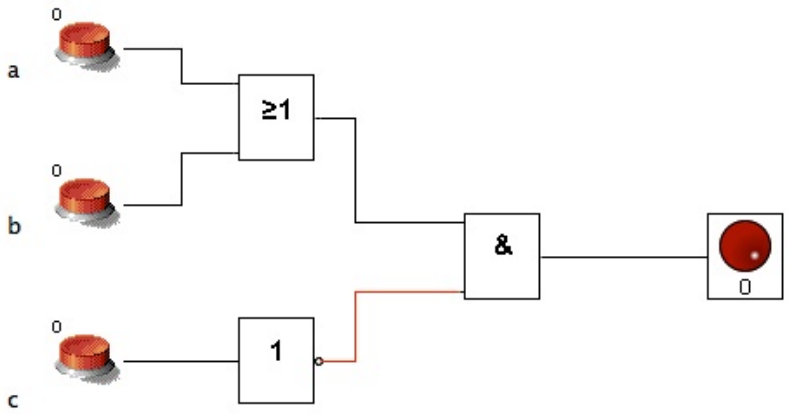
Detta räcker för godkänt svar. Det går att gå vidare till $x \vee y$:

$x \wedge \neg y \vee y =$ *(distributiva lagen)*
$(x \vee y) \wedge (\neg y \vee y) =$ *(inverslagen)*
$(x \vee y) \wedge 1 =$ *(identitetslagen)*
$x \vee y$

5. a) $x = a \wedge \neg(\neg b \vee c \wedge d)$

Se upp med beräkningsordningen och operationernas prioriteter!

b) Uttrycket kan t.ex. realiseras på följande sätt (ritat i LogicSim):



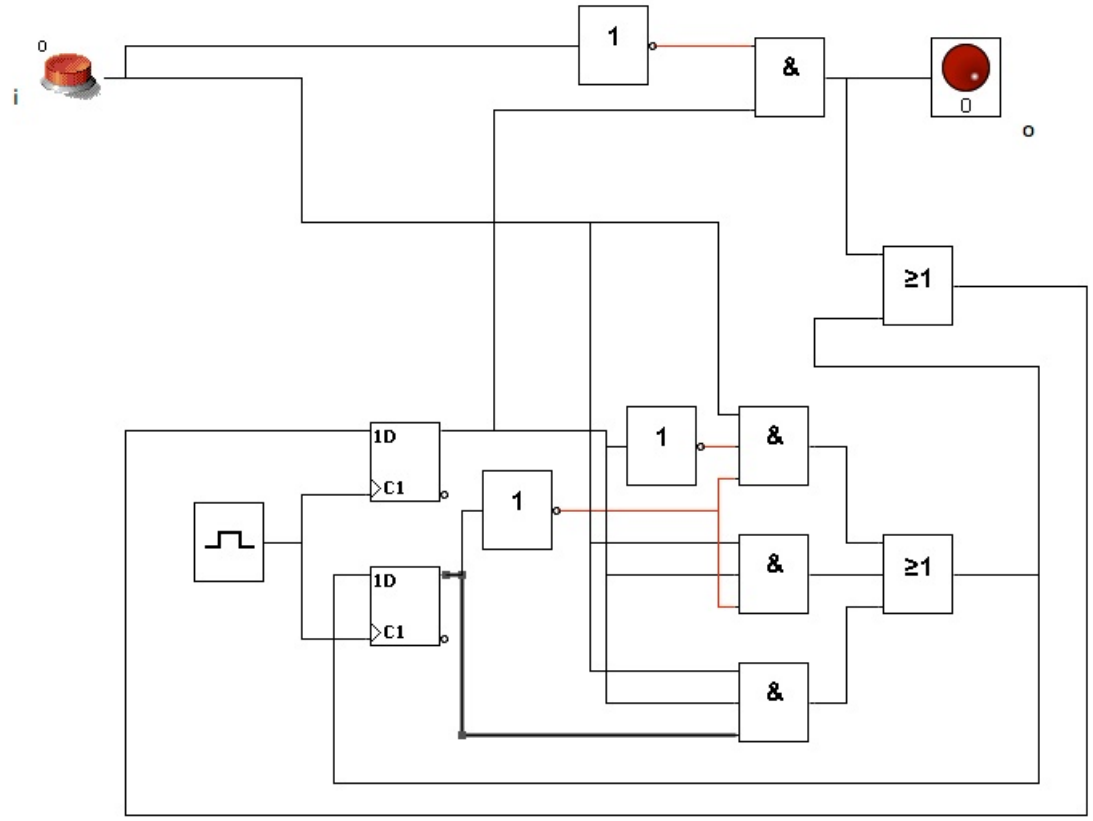
6. a) $o = (\neg i \wedge s1 \wedge \neg s2) \vee (\neg i \wedge s1 \wedge s2)$

   b) $(\neg i \wedge s1 \wedge \neg s2) \vee (\neg i \wedge s1 \wedge s2) =$
   $\neg i \wedge (s1 \wedge \neg s2) \vee \neg i \wedge (s1 \wedge s2) =$
   $\neg i \wedge ((s1 \wedge \neg s2) \vee (s1 \wedge s2)) =$
   $\neg i \wedge (s1 \wedge (\neg s2 \vee s2)) =$
   $\neg i \wedge (s1 \wedge 1) =$
   $\neg i \wedge s1$

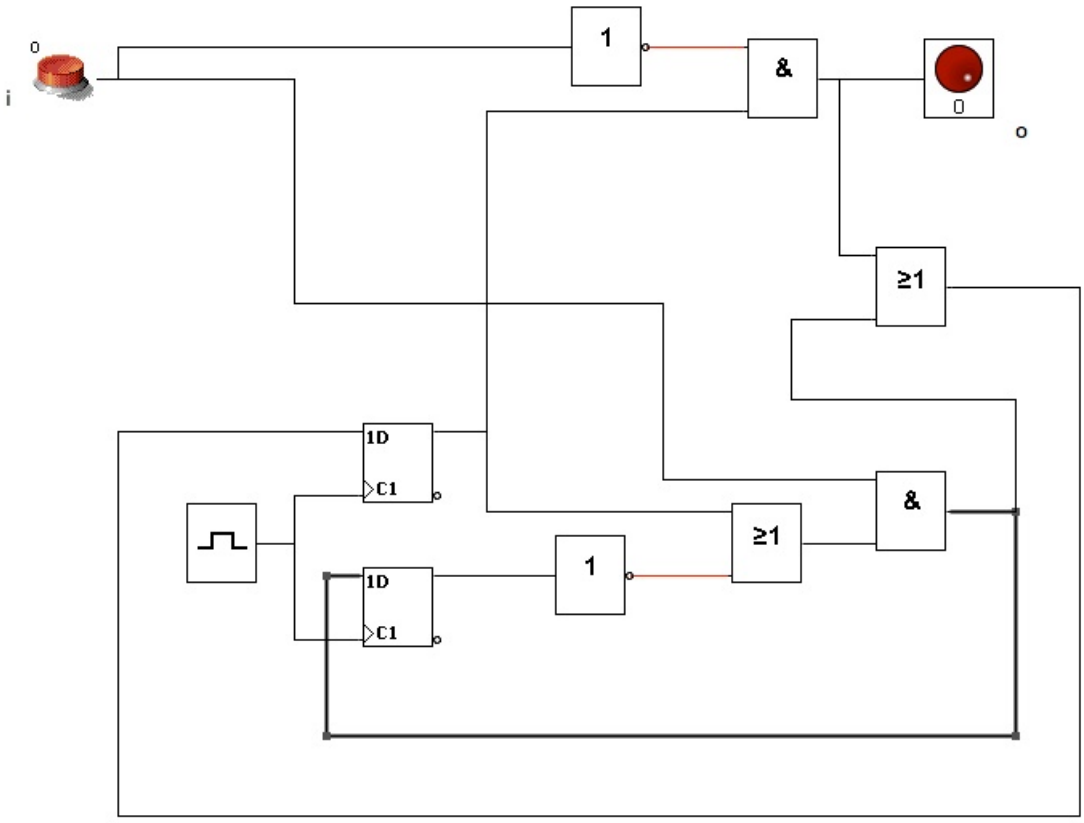
   c) $s2' = (i \wedge \neg s1 \wedge \neg s2) \vee (i \wedge s1 \wedge \neg s2) \vee (i \wedge s1 \wedge s2)$
   Det kan observeras att även detta uttryck kan förenklas med varierande möda om så önskas, t.ex. till
   $s2' = (i \wedge \neg s1 \wedge \neg s2) \vee (i \wedge s1) = i \wedge (\neg s1 \wedge \neg s2 \vee s1) = i \wedge (\neg s2 \vee s1)$

   d) $s1' = o \vee s2'$ (eller $s1' = o \oplus s2'$)

   e) Den färdiga maskinen kan t.ex. realiseras på nedanstående sätt (ritat i LogicSim). Mindre och enklare realiseringar är möjliga, men så här blir det om man direkt utgår från uttrycken ovan och inte använder den möjliga ytterligare förenklingen av s2′.



   På nästa sida visas hur en realisering som använder det förenklade uttrycket för s2′ kan se ut.


0
i
1
&
0
o
≥1
1D
C1
&
≥1
1
1D
C1