LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Tentamen
EDAA05 Datorer i system
Lösningsförslag

2010–10–18, 08.00–13.00

---

1. a) 100010111 och 117

   b) 10010100

   c) 74

   d) 1110100100001100

   e)
   ```
      11  1
    --------
     1001001
   +   11101
   ---------
     1100110
   ```

   f)
   ```
     xxxx
     1001001
   -   11101
   ---------
      101100
   ```

2. a) 7-bitars ASCII har bara plats för 128 olika tecken samt saknar nationella tecken (det finns dock nationella varianter av 7-bitars ASCII). ISO 8859-1 har plats för 256 tecken och innehåller nationella teckenvarianter för flera västeuropeiska språk.

   b) BOM förekommer när någon teckenkodningsstandard som bygger på Unicode används, t.ex. UTF-8. En BOM anger exakt vilken Unicode-variant som använts, UTF-8, UTF-16 eller UTF-32. Den anger också i förekommande fall i vilken ordning byten som bygger upp ett tecken kommer (big-endian eller little-endian).

   c) Unicode är en förteckning över hur olika tecken motsvaras av en teckenkod. Den säger dock ingenting om hur denna teckenkod exakt lagras i datorns minne. UTF-8 är ett exempel på en standard som talar om hur ett utvalt tecken i Unicode-tabellen ska lagras i form av en sekvens av bytes (eller ettor och nollor).

3. a) Sanningstabellerna blir:

p	q	r	$p \wedge q \vee \neg p \wedge r$	$r \wedge \neg(p \wedge q \wedge r)$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

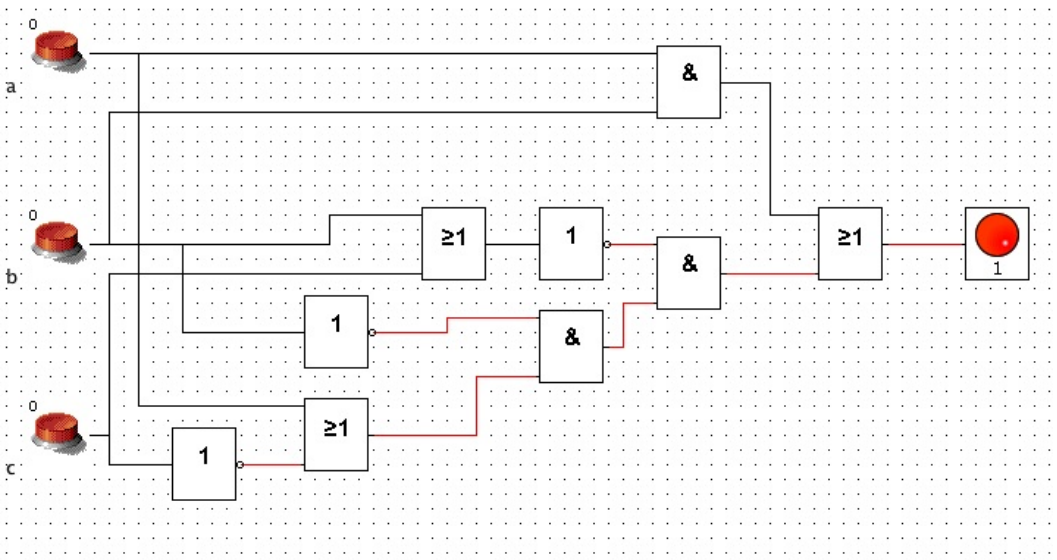
   Eftersom tabellerna för de två uttrycken skiljer sig åt är de *inte* ekvivalenta.

---

b) $p \wedge \neg q \vee \neg(r \vee \neg q) \vee r \wedge q = p \wedge \neg q \vee \neg r \wedge q \vee r \wedge q = p \wedge \neg q \vee q \wedge (\neg r \vee r) =$
$p \wedge \neg q \vee q = (p \vee q) \wedge (\neg \vee q) = p \vee q$

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

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



5. a) $o = i \wedge s1 \wedge \neg s2$
$s1' = \neg i \wedge \neg s1 \wedge s2 \vee \neg i \wedge s1 \wedge \neg s2$ (eller: $s1' = \neg i \wedge (s1 \oplus s2)$)
$s2' = \neg i \wedge \neg s1 \wedge \neg s2$

b) Den färdiga maskinen kan realiseras på följande (ritat i LogicSim):