Fall 1998 CS150 - Problem Set 1: Solutions David Chinnery

Question 1.

There is a simple way of changing from binary to octal or hexadecimal. Octal represents sets of three binary bits with a number from 0 to 7; hexadecimal represents sets of four binary bits with a number from 0 to F. Splitting the binary number left of the binary point into sets of three or four bits respectively, makes conversion easier. Similarly for binary fractions.

For example, converting 10100111.0111 to octal:

- Left of the binary point in groups of 3, we have 10 100 and 111. Which correspond in octal to 2 4 and 7 (247_8).
- Right of the binary point, in groups of 3, we have 011 and 100 [expand this to the full three digits, otherwise we would convert it to 1]. Which correspond to 3 and 4 ($.34_8$).
- So 10100111.0111 in octal is 247.34_8

	(a)	(b)	(c)	(d)	(e)
binary	11010101	110.111	10010110	10101000.1111	1001000001
octal	325	6.7	226	250.74	1101
decimal	213	6.875	150	168.9375	577
hexadecimal	$D5$	$6.E$	96	$A8.F$	241

Table 1: Solutions to Question 1.

Question 2.

	180_{10}	171_{10}	223_4	103_4
binary	10110100	10101011	101011	10011
decimal	180	171	43	19
hexadecimal	B4	AB	2B	13

Table 2: Conversions from base 10 and base 4

Base 2

		1	0	1	1	0	1	0	0
+		1	0	1	0	1	0	1	1
carry	1	0	1	0	0	0	0	0	0
	1	0	1	0	1	1	1	1	1

Base 10

	1	8	0
+	1	7	1
carry	1	0	0
	3	5	1

Base 16

		B	4
+		A	B
carry	1	0	0
	1	5	F

	Base 2										
						1	0	1	0	1	1
$\times$							1	0	0	1	1
						1	0	1	0	1	1
					1	0	1	0	1	1	
									0		
								0			
	1	0	1	0	1	1					
carry		1	1	1	1	1	1	1	1	0	
	1	1	0	0	1	1	0	0	0	1	

	Base 10		
		4	3
$\times$		1	9
	3	8	7
	4	3	0
carry	1	0	0
	8	1	7

	Base 16		
		2	B
$\times$		1	3
		8	1
	2	B	
carry	1	0	0
	3	3	1

Question 3.

(a) $105_{10} \bmod 8 = 1_{10}$ (8 divides 104)

(b) $105_{10} \bmod 4 = 1_{10}$ (4 divides 104)

(c) Not in general. If $(x \bmod 16) + (y \bmod 16) < 8$ then yes; a counter-example is $x = y = 7$, $(7 \bmod 16) + (7 \bmod 16) = 7 + 7 = 14$, but $14 \bmod 8 = 6$.

Question 4.

The theorems and laws, referred to by number, are on page 51 of the text.

(a)
$$\begin{aligned} F &= AB(C+\overline{C}) + A\overline{B} \\ &= AB + A\overline{B} && \text{(5. theorem of complementarity)} \\ &= A(B+\overline{B}) && \text{(8. AND is distributive)} \\ &= A && \text{(5. theorem of complementarity, and 1D.)} \end{aligned}$$

(b)
$$\begin{aligned} F &= AB + BC\overline{D} + \overline{A}C\overline{D} \\ &= AB + (A+\overline{A})BC\overline{D} + \overline{A}C\overline{D} && \text{(5. theorem of complementarity, and 1D.)} \\ &= AB + ABC\overline{D} + \overline{A}BC\overline{D} + \overline{A}C\overline{D} && \text{(6D. and 8., AND is distributive and commutative)} \\ &= AB(1 + C\overline{D}) + \overline{A}C\overline{D}(B+1) && \text{(1D., 6D. and 8.)} \\ &= AB + \overline{A}C\overline{D} && \text{(2. } 1 + x = 1\text{, where } x\epsilon\{0,1\}\text{)} \end{aligned}$$

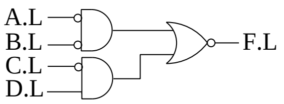
(c)
$$\begin{aligned} F &= \overline{A\overline{(B+C)}D} \\ &= \overline{A\overline{B} \cdot \overline{C}D} && \text{(12. DeMorgan's Theorem)} \\ &= \overline{A} + B + C + \overline{D} && \text{(12D. DeMorgan's Theorem)} \end{aligned}$$

(d)
$$\begin{aligned} F &= \overline{A\overline{B} \cdot \overline{C}} \\ &= \overline{A} + B + C && \text{(12D. DeMorgan's Theorem)} \end{aligned}$$

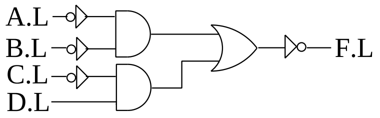
$$
\begin{aligned}
\text{(e)}\quad F &= (A+B)(\overline{A}+B+C)(\overline{A}+B+\overline{C}) & \\
&= (A+B)(\overline{A}\cdot\overline{A}+\overline{A}B+\overline{A}\cdot\overline{C}+BB+B\overline{C}+C\overline{C}) & \text{(8. AND is distributive)}\\
&= (A+B)(\overline{A}+\overline{A}B+\overline{A}\cdot\overline{C}+B+B\overline{C}) & \text{(3D. and 5D.)}\\
&= (A+B)(\overline{A}(1+B+\overline{C})+B(1+\overline{C}) & \text{(1D. and 8., AND is distributive)}\\
&= (A+B)(\overline{A}+B) & \text{(2.)}\\
&= A\overline{A}+AB+B\overline{A}+BB & \text{(8. AND is distributive)}\\
&= AB+B\overline{A}+B & \text{(3D. and 5D.)}\\
&= B(A+\overline{A}+1) & \text{(8. AND is distributive)}\\
&= B & \text{(2.)}
\end{aligned}
$$

Question 5.

In implementing negative logic (when inputs and outputs are asserted low), a simple and effective approach is to proceed as we would with postive logic, but invert the inputs and outputs (this approach is suggested by Katz for use with mixed positive and negative logic). Inversion is indicated by a small circle in the diagrams below - bubble matching with inputs and output.

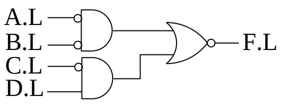


We do not need to invert D.L, as we want the negation in implementing AB+C$\overline{D}$.

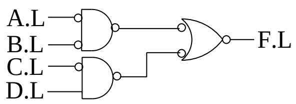


The resulting implementation, showing bubble matching.

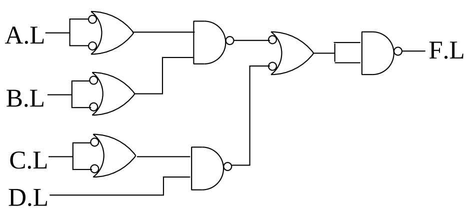
(a) implementation with AND, OR and INVERTER gates



From here we switch to NAND gates.

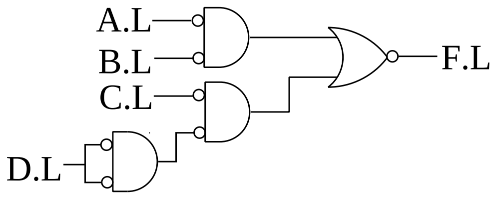


Which gives this.



And to get inversion, we feed an input into both ports of a NAND gate.

(b) implementation with NAND gates



The NOR gate implementation follows readily from the first diagram in 5(a). We require an inverter on the D.L input, so the D.L signal goes to both ports of a NOR gate.

(c) implementation with NOR gates

Question 6.

Implementation and truth tables are for $A - B$. BorrowIn is the input from the lower significant bit, and when 1 (true) indicates that we must subtract 1 from the current bit (as well as subtracting B). BorrowOut subtracts 1 from the next higher bit, which adds 2 (multiplying by the base as we go down a digit) to the current bit - necessary when we have $0 - 1$ (or $0 - 2$, or $1 - 2$ if both BorrowIn and B are 1). D is the subtractor output at the current bit.

A	B	BorrowIn	D	BorrowOut
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 3: (a) Truth table for one-bit full subtractor.

(b) The logic equations are (note $\oplus$ designates XOR):

$$\begin{aligned} D &= \overline{A} \cdot \overline{B} \cdot BorrowIn + \overline{A} \cdot B \cdot \overline{BorrowIn} + A \cdot \overline{B} \cdot \overline{BorrowIn} + A \cdot B \cdot BorrowIn \\ &= A \oplus B \oplus BorrowIn \end{aligned}$$

$$\begin{aligned} BorrowOut &= \overline{A} \cdot \overline{B} \cdot BorrowIn + \overline{A} \cdot B \cdot \overline{BorrowIn} + \overline{A} \cdot B \cdot BorrowIn + A \cdot B \cdot BorrowIn \\ &= \overline{A}(B + BorrowIn) + A \cdot B \cdot BorrowIn \end{aligned}$$

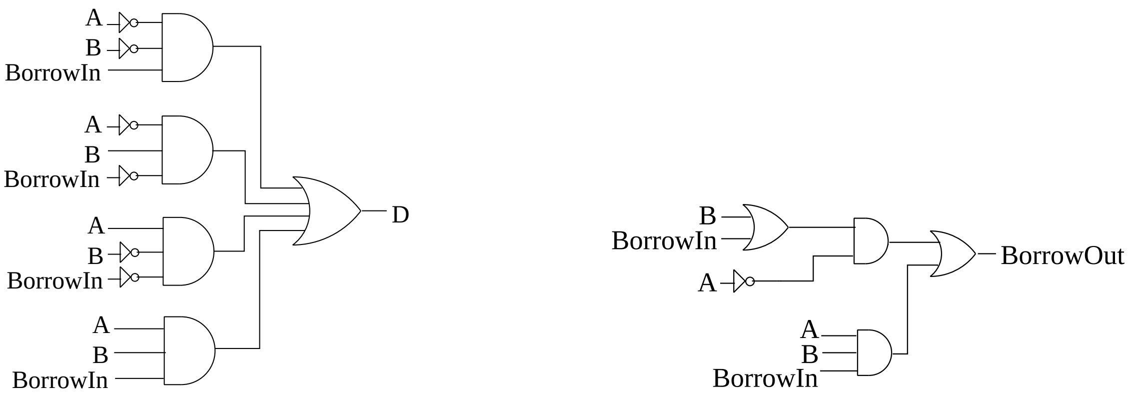


Table 4: (c) Logic for D and BorrowOut with AND, OR and INVERTER gates.

Question 7.
These timing diagrams assume A was zero prior to $t = 0$, otherwise the behaviour of $A1$ is not known until $t = 1$ and hence the behaviour of Y is not known until $t = 2$.

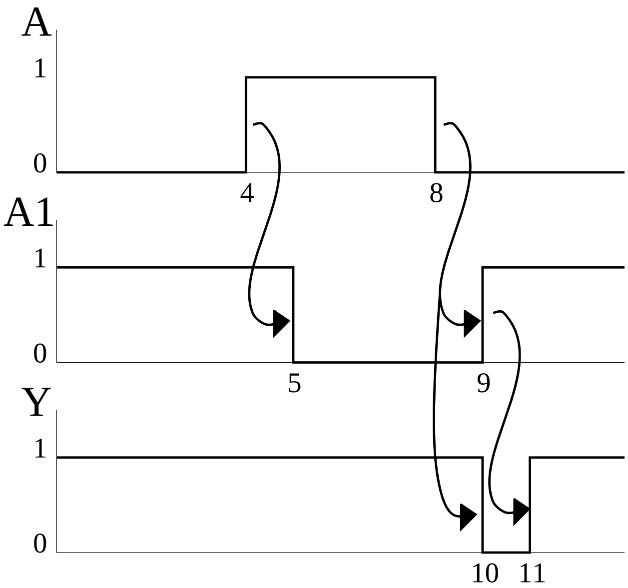


Table 5: Timing diagram of A, A1 and Y.