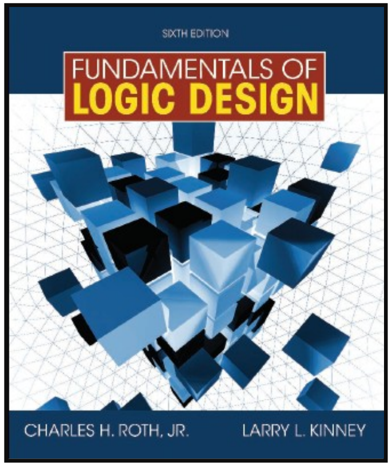
CHAPTER 1

INTRODUCTION
NUMBER SYSTEMS AND CONVERSION



This chapter in the book includes:

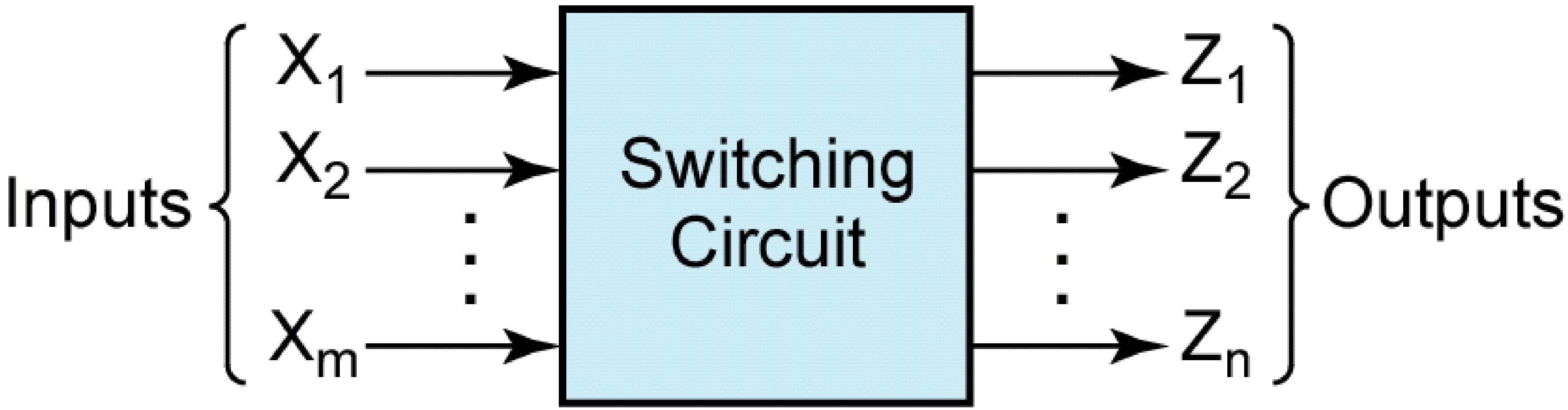


Figure 1-1: **Switching circuit**

Decimal Notation

$$953.78_{10} = 9x10^{2} + 5x10^{1} + 3x10^{0} + 7x10^{-1} + 8x10^{-2}$$

Binary

$$1011.11_{2} = 1x2^{3} + 0x2^{2} + 1x2^{1} + 1x2^{0} + 1x2^{-1} + 1x2^{-2}$$

$$= 8 + 0 + 2 + 1 + {}^{1}/_{2} + {}^{1}/_{4}$$

$$= 11.75_{10}$$

EXAMPLE: Convert 53_{10} to binary.

$$
\begin{array}{lll}
2\,\underline{/53} & & \\
2\,\underline{/26} & \text{rem.} = 1 = a_0 & \\
2\,\underline{/13} & \text{rem.} = 0 = a_1 & \\
2\,\underline{/6} & \text{rem.} = 1 = a_2 & 53_{10} = 110101_2 \\
2\,\underline{/3} & \text{rem.} = 0 = a_3 & \\
2\,\underline{/1} & \text{rem.} = 1 = a_4 & \\
\quad 0 & \text{rem.} = 1 = a_5 &
\end{array}
$$

Conversion (a)

EXAMPLE: **Convert $.625_{10}$ to binary.**

$$F = .625 \quad \times 2 \quad \underline{1.250} \quad (a_{-1} = 1)$$

$$F_1 = .250 \quad \times 2 \quad \underline{0.500} \quad (a_{-2} = 0)$$

$$F_2 = .500 \quad \times 2 \quad \underline{1.000} \quad (a_{-3} = 1)$$

$$.625_{10} = .101_2$$

Conversion (b)

EXAMPLE: Convert 0.7_{10} to binary.

$$\begin{array}{r}
.7 \\
2 \\
\hline
(1).4 \\
2 \\
\hline
(0).8 \\
2 \\
\hline
(1).6 \\
2 \\
\hline
(1).2 \\
2 \\
\hline
(0).4 \\
2 \\
\hline
(0).8
\end{array}$$

$\longleftarrow$ process starts repeating here because 0.4 was previously obtained

$$0.7_{10} = 0.1\ \underline{0110}\ \underline{0110}\ \underline{0110}\ \ldots_2$$

Conversion (c)

EXAMPLE: Convert 231.3_4 to base 7.

$$231.3_4 = 2 \times 16 + 3 \times 4 + 1 + \tfrac{3}{4} = 45.75_{10}$$

$$\begin{array}{rl}
7 \,\underline{/ 45} & \\
7 \,\underline{/ \;\;6} & \text{rem. } 3 \\
0 & \text{rem. } 6
\end{array}
\qquad
\begin{array}{r}
.75 \\
\underline{7} \\
(5)\ .25 \\
\underline{7} \\
(1)\ .75 \\
\underline{7} \\
(5)\ .25 \\
\underline{7} \\
(1)\ .75
\end{array}
\qquad
45.75_{10} = 63.5151\ldots_7$$

Conversion (d)

Binary ⇔ Hexadecimal Conversion

$$1001101.010111_2 = \underbrace{0100}_{4}\ \underbrace{1101}_{D}\,.\,\underbrace{0101}_{5}\ \underbrace{1100}_{C} = 4D.5C_{16}$$

Equation (1-1)

Conversion from binary to hexadecimal (and conversely) can be done by inspection because each hexadecimal digit corresponds to exactly four binary digits (bits).

Add 13_{10} and 11_{10} in binary.

$$
\begin{array}{rr}
 & 1111 \longleftarrow \text{carries} \\
13_{10} = & 1101 \\
11_{10} = & \underline{1011} \\
 & 11000 = 24_{10}
\end{array}
$$

Addition

The subtraction table for binary numbers is

$0 - 0 = 0$

$0 - 1 = 1$ and borrow 1 from the next column

$1 - 0 = 1$

$1 - 1 = 0$

Borrowing 1 from a column
is equivalent to subtracting 1 from that column.

Subtraction (a)

EXAMPLES OF BINARY SUBTRACTION:

(a)
$$\begin{array}{rl} 1\longleftarrow & \text{(indicates} \\ 11101 & \text{a borrrow} \\ -10011 & \text{from the} \\ \hline 1010 & \text{3rd column)} \end{array}$$

(b)
$$\begin{array}{rl} 1111\longleftarrow & \text{borrows} \\ 10000 & \\ -\quad 11 & \\ \hline 1101 & \end{array}$$

(c)
$$\begin{array}{rl} 111\longleftarrow & \text{borrows} \\ 111001 & \\ -\quad 1011 & \\ \hline 101110 & \end{array}$$

Subtraction (b)

A detailed analysis of the borrowing process for this example, indicating first a borrow of 1 from column 1 and then a borrow of 1 from column 2, is as follows:

$$
\begin{aligned}
205 - 18 &= [2 \times 10^2 + 0 \times 10^1 + 5 \times 10^0] \\
&\quad - [\phantom{2 \times 10^2 + {}} 1 \times 10^1 + 8 \times 10^0] \\
&\qquad \text{note borrow from column 1} \\
&= [2 \times 10^2 + (0 - 1) \times 10^1 + (10 + 5) \times 10^0] \\
&\quad - [\phantom{2 \times 10^2 + (0 - {}} 1 \times 10^1 + 8 \times 10^0] \\
&\qquad \text{note borrow from column 2} \\
&= [(2 - 1) \times 10^2 + (10 + 0 - 1) \times 10^1 + 15 \times 10^0] \\
&\quad - [\phantom{(2 - 1) \times 10^2 + (10 + 0 - {}} 1 \times 10^1 + 8 \times 10^0] \\
&= [1 \times 10^2 + 8 \times 10^1 + 7 \times 10^0] = 187
\end{aligned}
$$

Subtraction (c)

The multiplication table for binary numbers is

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Multiplication (a)

The following example illustrates multiplication of 13_{10} by 11_{10} in binary:

$$
\begin{array}{r}
1101 \\
1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
10001111
\end{array}
= 143_{10}
$$

Multiplication (b)

When doing binary multiplication, a common way to avoid carries greater than 1 is to add in the partial products one at a time as illustrated by the following example:

```
     1111     multiplicand
     1101     multiplier
     ----
     1111     1st partial product
    0000      2nd partial product
    -----
   (01111)    sum of first two partial products
   1111       3rd partial product
   ------
  (1001011)   sum after adding 3rd partial product
  1111        4th partial product
  -------
  11000011    final product (sum after adding 4th
                                 partial product)
```

Multiplication (c)

Binary Division

Binary division is similar to decimal division, except it is much easier because the only two possible quotient digits are 0 and 1.

We start division by comparing the divisor with the upper bits of the dividend.

If we cannot subtract without getting a negative result, we move one place to the right and try again.

If we can subtract, we place a 1 for the quotient above the number we subtracted from and append the next dividend bit to the end of the difference and repeat this process with this modified difference until we run out of bits in the dividend.

The following example illustrates division of 145_{10} by 11_{10} in binary:

```
         1101
      ________
1011 |10010001
       1011
       ----
       1110
       1011
       ----
         1101
         1011
         ----
           10
```

The quotient is 1101 with a remainder of 10.

Binary Division

3 Systems for representing negative numbers in binary

Sign & Magnitude: Most significant bit is the sign

Ex: $-5_{10} = 1101_2$

1's Complement: $\overline{N} = (2^n - 1) - N$

Ex: $-5_{10} = (2^4 - 1) - 5 = 16 - 1 - 5 = 10_{10} = 1010_2$

2's Complement: $N^* = 2^n - N$

Ex: $-5_{10} = 2^4 - 5 = 16 - 5 = 11_{10} = 1011_2$

Section 1.4 (p. 16)

Table 1-1: Signed Binary Integers (word length $n = 4$)

$+N$	Positive Integers (all systems)	$-N$	Negative Integers		
			Sign and Magnitude	2's Complement N^*	1's Complement $\bar{N}$
+0	0000	−0	1000	——	1111
+1	0001	−1	1001	1111	1110
+2	0010	−2	1010	1110	1101
+3	0011	−3	1011	1101	1100
+4	0100	−4	1100	1100	1011
+5	0101	−5	1101	1011	1010
+6	0110	−6	1110	1010	1001
+7	0111	−7	1111	1001	1000
		−8	——	1000	——

1. Addition of two positive numbers, sum $< 2^{n-1}$

$$\begin{array}{rr l} +3 & 0011 & \\ +4 & 0100 & \\ \hline +7 & 0111 & \text{(correct answer)} \end{array}$$

2. Addition of two positive numbers, sum $\geq 2^{n-1}$

$$\begin{array}{rr l} +5 & 0101 & \\ +6 & 0110 & \\ \hline & 1011 & \longleftarrow \text{wrong answer because of overflow (+11 requires} \\ & & \text{5 bits including sign)} \end{array}$$

2's Complement Addition (a)

3. Addition of positive and negative numbers (negative number has greater magnitude)

$$
\begin{array}{rr l}
+5 & 0101 & \\
\underline{-6} & \underline{1010} & \\
-1 & 1111 & \text{(correct answer)}
\end{array}
$$

4. Same as case 3 except positive number has greater magnitude

$$
\begin{array}{rr l}
-5 & 1011 & \\
\underline{+6} & \underline{0110} & \\
+1 & (1)0001 & \longleftarrow \text{correct answer when the carry from the sign bit} \\
 & & \quad\;\; \text{is ignored (this is } \textit{not} \text{ an overflow)}
\end{array}
$$

2’s Complement Addition (b)

5. Addition of two negative numbers, $|\text{sum}| \le 2^{n-1}$

$$
\begin{array}{rr}
-3 & 1101 \\
-4 & 1100 \\
\hline
-7 & (1)1001
\end{array}
$$

$\longleftarrow$ correct answer when the last carry is ignored (this is *not* an overflow)

6. Addition of two negative numbers, $|\text{sum}| > 2^{n-1}$

$$
\begin{array}{rr}
-5 & 1011 \\
-6 & 1010 \\
\hline
 & (1)0101
\end{array}
$$

$\longleftarrow$ wrong answer because of overflow (-11 requires 5 bits including sign)

2's Complement Addition (c)

3. Addition of positive and negative numbers (negative number with greater magnitude)

$$\begin{array}{rr} +5 & 0101 \\ \underline{-6} & \underline{1001} \\ -1 & 1110 \end{array} \qquad \text{(correct answer)}$$

4. Same as case 3 except positive number has greater magnitude

$$\begin{array}{rrl} -5 & 1010 & \\ \underline{+6} & \underline{0110} & \\ (1) & 0000 & \\ \hookrightarrow & \underline{1} & \text{(end-around carry)} \\ & 0001 & \text{(correct answer, } \textit{no} \text{ overflow)} \end{array}$$

1’s Complement Addition (b)

5. Addition of two negative numbers, $|\text{sum}| < 2^{n-1}$

$$\begin{array}{rrl} -3 & 1100 & \\ \underline{-4} & \underline{1011} & \\ & (1)\ 0111 & \\ & \hookrightarrow\ \underline{\quad 1} & \text{(end-around carry)} \\ & 1000 & \text{(correct answer, } \textit{no} \text{ overflow)} \end{array}$$

6. Addition of two negative numbers, $|\text{sum}| \geq 2^{n-1}$

$$\begin{array}{rrl} -5 & 1010 & \\ \underline{-6} & \underline{1001} & \\ & (1)\ 0011 & \\ & \hookrightarrow\ \underline{\quad 1} & \text{(end-around carry)} \\ & 0100 & \text{(wrong answer because of overflow)} \end{array}$$

1's Complement Addition (c)

1. Add −11 and −20 in 1's complement.

$+11 = 00001011 \qquad\qquad +20 = 00010100$

taking the bit-by-bit complement,

−11 is represented by 11110100 and −20 by 11101011

$$
\begin{array}{rl}
11110100 & \quad (-11) \\
\underline{11101011} & \quad \underline{+(-20)} \\
(1)\ 11011111 & \\
\longrightarrow 1 & \quad \text{(end-around carry)} \\
\hline
11100000 = -31 &
\end{array}
$$

1's Complement Addition (d)

2. Add −8 and +19 in 2's complement

+ 8 = 00001000

complementing all bits to the left of the first 1,
−8, is represented by 11111000

$$\begin{array}{rr} 11111000 & (-8) \\ \underline{00010011} & \underline{+19} \\ \cancel{(1)}00001011 = & +11 \end{array}$$

(discard last carry)

2's Complement Addition (d)

Binary Codes

Although most large computers work internally with binary numbers, the input-output equipment generally uses decimal numbers. Because most logic circuits only accept two-valued signals, the decimal numbers must be coded in terms of binary signals. In the simplest form of binary code, each decimal digit is replaced by its binary equivalent. For example, 937.25 is represented by:

$$\overbrace{1001}^{9}\quad\overbrace{0011}^{3}\quad\overbrace{0111}^{7}\quad.\quad\overbrace{0010}^{2}\quad\overbrace{0101}^{5}$$

Section 1.5 (p. 21)

Table 1–2. Binary Codes for Decimal Digits

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011

	ASCII Code								ASCII Code						
Character	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Character	A_6	A_5	A_4	A_3	A_2	A_1	A_0
space	0	1	0	0	0	0	0	@	1	0	0	0	0	0	0
!	0	1	0	0	0	0	1	A	1	0	0	0	0	0	1
“	0	1	0	0	0	1	0	B	1	0	0	0	0	1	0
#	0	1	0	0	0	1	1	C	1	0	0	0	0	1	1
$	0	1	0	0	1	0	0	D	1	0	0	0	1	0	0
%	0	1	0	0	1	0	1	E	1	0	0	0	1	0	1
&	0	1	0	0	1	1	0	F	1	0	0	0	1	1	0
′	0	1	0	0	1	1	1	G	1	0	0	0	1	1	1
(	0	1	0	1	0	0	0	H	1	0	0	1	0	0	0
)	0	1	0	1	0	0	1	I	1	0	0	1	0	0	1
*	0	1	0	1	0	1	0	J	1	0	0	1	0	1	0
+	0	1	0	1	0	1	1	K	1	0	0	1	0	1	1
,	0	1	0	1	1	0	0	L	1	0	0	1	1	0	0
–	0	1	0	1	1	0	1	M	1	0	0	1	1	0	1
.	0	1	0	1	1	1	0	N	1	0	0	1	1	1	0
/	0	1	0	1	1	1	1	O	1	0	0	1	1	1	1
0	0	1	1	0	0	0	0	P	1	0	1	0	0	0	0
1	0	1	1	0	0	0	1	Q	1	0	1	0	0	0	1
2	0	1	1	0	0	1	0	R	1	0	1	0	0	1	0
3	0	1	1	0	0	1	1	S	1	0	1	0	0	1	1
4	0	1	1	0	1	0	0	T	1	0	1	0	1	0	0

Table 1-3 ASCII code (incomplete)