CHAPTER 1

INTRODUCTION

1.1 EXERCISES

Section 1.2: The World of Digital Systems

1.1. What is a digital signal and how does it differ from an analog signal? Give two everyday examples of digital phenomena (e.g., a window can be open or closed) and two everyday examples of analog phenomena.

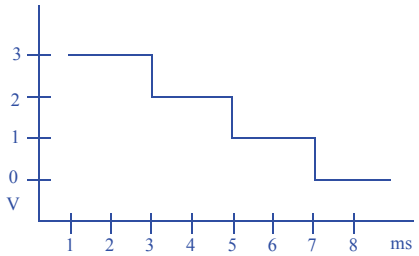
A digital signal at any time takes on one of a finite number of possible values, whereas an analog signal can take on one of infinite possible values. Examples of digital phenomena include a traffic light that is either be red, yellow, or green; a television that is on channel 1, 2, 3, ..., or 99; a book that is open to page 1, 2, ..., or 200; or a clothes hangar that either has something hanging from it or doesn't. Examples of analog phenomena include the temperature of a room, the speed of a car, the distance separating two objects, or the volume of a television set (of course, each analog phenomena could be digitized into a finite number of possible values, with some accompanying loss of information).

1.2 Suppose an analog audio signal comes in over a wire, and the voltage on the wire can range from 0 Volts (V) to 3 V. You want to convert the analog signal to a digital signal. You decide to encode each sample using two bits, such that 0 V would be encoded as `00`, 1 V as `01`, 2 V as `10`, and 3 V as `11`. You sample the signal every 1 millisecond and detect the following sequence of voltages: 0V 0V 1V 2V 3V 2V 1V. Show the signal converted to digital as a stream of `0`s and `1`s.

00 00 01 10 11 10 01

1.3 Assume that 0 V is encoded as `00`, 1 V as `01`, 2 V as `10`, and 3 V as `11`. You are given a digital encoding of an audio signal as follows: `1111101001010000`. Plot

the re-created signal with time on the x-axis and voltage on the y-axis. Assume that each encoding's corresponding voltage should be output for 1 millisecond.



1.4 Assume that a signal is encoded using 12 bits. Assume that many of the encodings turn out to be either `000000000000`, `000000000001`, or `111111111111`. We thus decide to create compressed encodings by representing `000000000000` as `00`, `000000000001` as `01`, and `111111111111` as `10`. `11` means that an uncompressed encoding follows. Using this encoding scheme, decompress the following encoded stream:

`00 00 01 10 11 010101010101 00 00 10 10`

000000000000 000000000000 000000000001 111111111111 010101010101 000000000000 000000000000 111111111111 111111111111

1.5 Using the same encoding scheme as in Exercise 1.4, compress the following unencoded stream:

`000000000000 000000000001 100000000000 111111111111`

00 01 11 100000000000 10

1.6 Encode the following words into bits using the ASCII encoding table in Figure 1.9.

a. LET
b. RESET!
c. HELLO $1

a) 1001100 1000101 1010100
b) 1010010 1000101 1010011 1000101 1010100 0100001
c) 1001000 1000101 1001100 1001100 1001111 0100000 0100100 0110001 (don't forget the encoding 0100000 for the space between the O and the $).

1.7 Suppose your are building a keybad that has the buttons A through G. A three-bit output should indicate which button is currently being pressed. 000 represents no button being pressed. Decide on a 3-bit encoding to represent each button being pressed.

One possible set of encodings is: A=001, B=010, C=011, D=100, E=101, F=110, and G=111. Another possible set is: A=001, B=010, C=100, D=101, E=110, F=111, G=011. Many other sets of encodings are possible; any set of encodings is fine as long as each encoding is unique.

1.8 Convert the following binary numbers to decimal numbers:

a. 100

b. 1011
c. 0000000000001
d. 111111
e. 101010

a) 4
b) 11
c) 1
d) 63
e) 42

1.9 Convert the following binary numbers to decimal numbers:

a. 1010
b. 1000000
c. 11001100
d. 11111
e. 10111011001

a) 10
b) 64
c) 204
d) 31
e) 1497

1.10 Convert the following binary numbers to decimal numbers:

a. 000011
b. 1111
c. 11110
d. 111100
e. 0011010

a) 3
b) 15
c) 30
d) 60
e) 26

1.11 Convert the following decimal numbers to binary numbers using the addition method:

a. 9
b. 15
c. 32
d. 140

a) 1001
b) 1111
c) 100000
d) 10001100

1.12 Convert the following decimal numbers to binary numbers using the addition method:

a. 19
b. 30
c. 64
d. 128

a) 10011
b) 11110
c) 1000000
d) 10000000

1.13 Convert the following decimal numbers to binary numbers using the addition method:

a. 3
b. 65
c. 90
d. 100

a) 11
b) 1000001
c) 1011010
d) 1100100

1.14 Convert the following decimal numbers to binary numbers using the divide-by-2 method:

a. 9
b. 15
c. 32
d. 140

a) 1001
b) 1111
c) 100000
d) 10001100

1.15 Convert the following decimal numbers to binary numbers using the divide-by-2 method:

a. 19
b. 30
c. 64
d. 128

a) 10011
b) 11110
c) 1000000
d) 10000000

1.16 Convert the following decimal numbers to binary numbers using the divide-by-2 method:

a. 3
b. 65
c. 90
d. 100

a) 11
b) 1000001
c) 1011010
d) 1100100

1.17 Convert the following decimal numbers to binary numbers using the divide-by-2 method:

a. 23
b. 87
c. 123
d. 101

a) 10111
b) 1010111
c) 1111011
d) 1100101

1.18 Convert the following binary numbers to hexadecimal:

a. 11110000
b. 11111111
c. 01011010
d. 1001101101101

a) F0
b) FF
c) 5A
d) 136D

1.19 Convert the following binary numbers to hexadecimal:

a. 11001101
b. 10100101
c. 11110001
d. 1101101111100

a) CD
b) A5
c) F1
d) 1B7C

1.20 Convert the following binary numbers to hexadecimal:

a. 11100111
b. 11001000

c. 10100100
d. 011001101101101

a) E7
b) C8
c) A4
d) 336D

1.21 Convert the following hexadecimal numbers to binary:

a. FF
b. F0A2
c. 0F100
d. 100

a) 1111 1111
b) 1111 0000 1010 0010
c) 0000 1111 0001 0000 0000
d) 0001 0000 0000

1.22 Convert the following hexadecimal numbers to binary:

a. 4F5E
b. 3FAD
c. 3E2A
d. DEED

a) 0100 1111 0101 1110
b) 0011 1111 1010 1101
c) 0011 1110 0010 1010
d) 1101 1110 1110 1101

1.23 Convert the following hexadecimal numbers to binary:

a. B0C4
b. 1EF03
c. F002
d. BEEF

a) 1011 0000 1100 0100
b) 0001 1110 1111 0000 0011
c) 1111 0000 0000 0010
d) 1011 1110 1110 1111

1.24 Convert the following hexadecimal numbers to decimal:

a. FF
b. F0A2
c. 0F100
d. 100

a) 255
b) 61602
c) 61696

d) 256

1.25 Convert the following hexadecimal numbers to decimal:

a. 10
b. 4E3
c. FF0
d. 200

a) 16
b) 1251
c) 4080
d) 512

1.26 Convert the decimal number 128 to the following number systems:

a. binary
b. hexadecimal
c. base three
d. base five
e. base fifteen

a) 10000000
b) 80
c) 11202
d) 1003
e) 88

1.27 Compare the number of digits necessary to represent the following decimal numbers in binary, octal, decimal, and hexadecimal representations. You need not determine the actual representations -- just the number of required digits. For example, representing the decimal number 12 requires four digits in binary (1100 is the actual representation), two digits in octal (14), two digits in decimal (12), and one digit in hexadecimal (C).

a. 8
b. 60
c. 300
d. 1000
e. 999,999

a) 4 digits in binary, 2 digits in octal, 1 digit in decimal, 1 digit in hexadecimal
b) 6 digits in binary, 2 digits in octal, 2 digits in decimal, 2 digits in hexadecimal
c) 9 digits in binary, 3 digits in octal, 3 digits in decimal, 3 digits in hexadecimal
d) 10 digits in binary, 4 digits in octal, 4 digits in decimal, 3 digits in hexadecimal
e) 20 digits in binary, 7 digits in octal, 6 digits in decimal, 5 digits in hexadecimal

1.28 Determine the decimal number ranges that can be represented in binary, octal, decimal, and hexadecimal using the following numbers of digits. For example, 2 digits can represent decimal number range 0 through 3 in binary (00 through 11), 0 through 63 in octal (00 through 77), 0 through 99 in decimal (00 through 99), and 0 through 255 in hexadecimal (00 through FF).

a. 1
b. 3
c. 6
d. 8

a) 0-1 in binary, 0-7 in octal, 0-9 in decimal, 0-15 in hexadecimal
b) 0-7 in binary, 0-511 in octal, 0-999 in decimal, 0-4,095 in hexadecimal
c) 0-63 in binary, 0-262,143 in octal, 0-999,999 in decimal, 0-16,777,215 in hexadecimal
d) 0-255 in binary, 0-16,777,215, 0-99,999,999 in decimal, 0-4,294,967,295 in hexadecimal

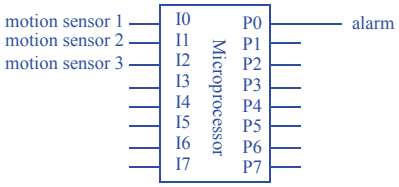
1.29 Rewrite the following bit quantities as byte quantities, using the most appropriate metric prefix, e.g., 16,000 bits (2,000 bytes) would be rewritten as 2 Kbytes.

a. 8,000,000
b. 32,000,000,000
c. 1,000,000,000

a) 8,000,000 bits * (1 byte/ 8 bits) = 1,000,000 bytes = 1 Mbyte
b) 32,000,000,000 bits / 8 = 4,000,000,000 = 4 Gbytes
c) 1,000,000,000 bits / 8 = 125,000,000 bytes = 125 Mbytes

Section 1.3: Implementing Digital Systems: Programming Microprocessors versus Designing Digital Circuits

1.30 Use a microprocessor like that in Figure 1.23 to implement a system that sounds an alarm whenever there is motion detected at the same time in three different rooms. Each room's motion sensor output comes to us on a wire as a bit, `1` meaning motion, `0` meaning no motion. We sound the alarm by setting an output wire "alarm" to `1`. Show the connections to and from the microprocessor, and the C code to execute on the microprocessor.




```
void main() {
  while (1) {
    P0 = I0 && I1 && I2;
  }
}
```

1.31 A security camera company wishes to add a face recognition feature to their cameras such that the camera only broadcasts video when a human face is detected in the video. The camera records 30 video frames per second. For each frame, the camera would execute a face recognition application. The application implemented on a

microprocessor requires 50 ms. The application implemented as a custom digital circuit requires 1 ms. Compute the maximum number of frames per second that each implementation supports, and indicate which implementation is sufficient for 30 frames per second.

50 ms/frame means 1 frame / 50 ms = 1 frame / 0.05 s = 20 frames / s.
1 ms/frame means 1 frame / 1 ms = 1 frame / 0.001 s = 1000 frames / s.
Thus, the digital circuit implementation would suffice, but the microprocessor implementation is too slow.

1.32 Suppose a particular banking system supports encrypted transactions, and that decrypting each transaction consists of three sub-tasks A, B, and C. The execution times of each task on a microprocessor versus a custom digital circuit are 50 ms versus 1 ms for A, 20 ms versus 2 ms for B, and 20 ms versus 1 ms for C. Partition the tasks among the microprocessor and custom digital circuitry, such that you minimize the amount of custom digital circuitry, while meeting the constraint of decrypting at least 40 transactions per second. Assume each task requires the same amount of digital circuitry.

40 transactions / second means that decryption should occur at a rate of 1 second / 40 transactions = 0.025 seconds / transaction, or 25ms/transaction. Implementing all three tasks on the microprocessor would result in 50+20+20 = 90 ms/transaction, which is too slow. Implementing any one task as a digital circuit is still too slow. Implementing A as a digital circuit would reduce the time to 1+20+20 = 41 ms. Implementing A and B as a digital circuit would reduce the time to 1+2+20 = 23 ms. Implementing A and C as a digital circuit would reduce the time to 1+20+1 = 22 ms. Thus, either solution suffices. Implementing B and C as a digital circuit would not suffice, as the time would be 50+2+1 = 53 ms. Implementing all three as a digital circuit would result in 1+2+1 = 4 ms/transaction, which is plenty fast but uses extra digital circuitry. Thus, one solution is A and B as digital circuits, C on the microprocessor. Another solution is A and C as digital circuits, B on the microprocessor.

1.33 How many possible partitionings are there of a set of N tasks where each task can be implemented either on the microprocessor or as a custom digital circuit? How many possible partitionings are there of a set of 20 tasks (expressed as a number without any exponents)?

2^n
For 20 tasks, there are 2^{20} or 1,048,576 (over 1 million) possible partitionings.