**DIGITAL SYSTEMS: Course Objectives and Lecture Plan**

**Aim**: At the end of the course the student will be able to analyze, design, and evaluate digital circuits, of medium complexity, that are based on SSIs, MSIs, and programmable logic devices.

**Module 1: Number Systems and Codes (3)**

Number systems: Binary, octal, and hexa-decimal number systems, binary arithmetic. Codes: Binary code, excess-3 code, gray code, and error detection and correction codes.

**Module 2: Boolean Algebra and Logic Functions (5)**

Boolean algebra: Postulates and theorems. Logic functions, minimization of Boolean functions using algebraic, Karnaugh map and Quine – McClausky methods. Realization using logic gates

**Module 3: Logic Families (4)**

Logic families: Characteristics of logic families. TTL, CMOS, and ECL families. **Module 4: Combinational Functions (8)**

Realizing logical expressions using different logic gates and comparing their performance. Hardware aspects logic gates and combinational ICs: delays and hazards. Design of combinational circuits using combinational ICs: Combinational functions: code conversion, decoding, comparison, multiplexing, demultiplexing, addition, and subtraction.

**Module 5: Analysis of Sequential Circuits (5)**

Structure of sequential circuits: Moore and Melay machines. Flip-flops, excitation tables, conversions, practical clocking aspects concerning flip-flops, timing and triggering considerations. Analysis of sequential circuits: State tables, state diagrams and timing diagrams.

**Module 6: Designing with Sequential MSIs (6)**

Realization of sequential functions using sequential MSIs: counting, shifting, sequence generation, and sequence detection.

**Module 7: PLDs (3)**

Programmable Logic Devices: Architecture and characteristics of PLDs, **Module 8: Design of Digital Systems (6)**

State diagrams and their features. Design flow: functional partitioning, timing relationships, state assignment, output racing. Examples of design of digital systems using PLDs

**Lecture Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Modules** | **Learning Units** | **Hours**  **per**  **topic** | **Total**  **Hours** |
| 1. Number  Systems and  Codes | 1. Binary, octal and hexadecimal number systems, and conversion of number with one radix to another | 1.5 | 3 |
| 2. Different binary codes | 1.5 |
| 2. Logic  Functions | 3. Boolean algebra and Boolean operators | 1.5 | 5 |
| 4. Logic Functions | 1 |
| 5. Minimization of logic functions using Karnaugh -map | 1.5 |
| 6. Quine-McClausky method of minimization of logic functions | 1 |
| 3.Logic Families | 7. Introduction to Logic families | 0.5 | 4 |
| 8. TTL family | 1 |
| 9. CMOS family | 1.5 |
| 10. Electrical characteristics of logic families | 1 |
| 4. Combinational Circuits | 11. Introduction to combinational circuits, logic convention, and realization of simple  combinational functions using gates | 2 | 8 |
| 12. Implications of delay and hazard | 1 |
| 13. Realization of adders and subtractors | 2 |
| 14. Design of code converters, comparators, and decoders | 2 |
| 15. Design of multiplexers, demultiplexers, | 1 |
| 5. Analysis of  Sequential  Circuits | 16. Introduction to sequential circuits: Moore and Mealy machines | 1 | 5 |
| 17. Introduction to flip-flops like SR, JK, D & T with truth tables, logic diagrams, and timing relationships | 1 |
| 18. Conversion of Flip-Flops, Excitation table | 1 |
| 19. State tables, and realization of state stables | 2 |
| 6. Design with Sequential MSIs | 20. Design of shift registers and counters | 2 | 6 |
| 21. Design of counters | 2 |
| 22. Design of sequence generators and detectors | 2 |
| 7. PLDs | 23. Introduction to Programmable Devices | 1 | 3 |
| 24. Architecture of PLDs | 2 |
| 8. Design of  Digital Systems | 25. State diagrams and their features | 2 | 6 |
| 26. Design flow | 1 |
| 27. Design of digital systems using PLDs | 3 |

**Learning Objectives of the Course**

**1. Recall**

1.1 List different criteria that could be used for optimization of a digital circuit.

1.2 List and describe different problems of digital circuits introduced by the hardware limitations.

**2. Comprehension**

2.1 Describe the significance of different criteria for design of digital circuits.

2.2 Describe the significance of different hardware related problems encountered in digital circuits.

2.3 Draw the timing diagrams for identified signals in a digital circuit. **3. Application**

3.1 Determine the output and performance of given combinational and sequential circuits.

3.2 Determine the performance of a given digital circuit with regard to an identified optimization criterion.

**4. Analysis**

4.1 Compare the performances of combinational and sequential circuits implemented with SSIs/MSIs and PLDs.

4.2 Determine the function and performance of a given digital circuit.

4.3 Identify the faults in a given circuit and determine the consequences of the same on the circuit performance.

4.4 Draw conclusions on the behavior of a given digital circuit with regard to hazards, asynchronous inputs, and output races.

4.5 Determine the appropriateness of the choice of the ICs used in a given digital circuit.

4.6 Determine the transition sequence of a given state in a state diagram for a given input sequence.

**5. Synthesis**

5.1 Generate multiple digital solutions to a verbally described problem. 5.2 Modify a given digital circuit to change its performance as per specifications. **6. Evaluation**

6.1 Evaluate the performance of a given digital circuit.

6.2 Assess the performance of a given digital circuit with Moore and Melay configurations.

6.3 Compare the performance of given digital circuits with respect to their speed, power consumption, number of ICs, and cost.

**Digital Systems: Motivation**

A digital circuit is one that is built with devices with two well-defined states. Such circuits can process information represented in binary form. Systems based on digital circuits touch all aspects our present day lives. The present day home products including electronic games and appliances, communication and office automation products, computers with a wide range of capabilities, and industrial instrumentation and control systems, electro

medical equipment, and defence and aerospace systems are heavily dependent on digital circuits. Many fields that emerged later to digital electronics have peaked and levelled off, but the application of digital concepts appears to be still growing exponentially. This unprecedented growth is powered by the semiconductor technology, which enables the introduction of more and complex integrated circuits. The complexity of an integrated circuit is measured in terms of the number of transistors that can be integrated into a single unit. The number of transistors in a single integrated circuit has been doubling every eighteen months (Moore’ Law) for several decades and reached the figure of almost one billion transistors per chip. This allowed the circuit designers to provide more and more complex functions in a single unit.

The introduction of programmable integrated circuits in the form of microprocessors in 70s completely transformed every facet of electronics. While fixed function integrated circuits and microprocessors coexisted for considerable time, the need to make the equipment smaller and portable lead to replacement of fixed function devices with programmable devices. With the all pervasive presence of the microprocessor and the increasing usage of other programmable circuits like PLDs (Programmable Logic devices), FPGAs (Field Programmable Gate Arrays) and ASICs (Application Specific Integrated Circuits), the very nature of digital systems is continuously changing.

The central role of digital circuits in all our professional and personal lives makes it imperative that every electrical and electronics engineer acquire good knowledge of relevant basic concepts and ability to work with digital circuits.

At present many of the undergraduate programmes offer two to four courses in the area of digital systems, with at least two of them being core courses. The course under consideration constitutes the first course in the area of digital systems. The rate of obsolescence of knowledge, design methods, and design tools is uncomfortably high. Even the first level course in digital electronics is not exempt from this obsolescence.

Any course in electronics should enable the students to design circuits to meet some stated requirements as encountered in real life situations. However, the design approaches should be based on a sound understanding of the underlying principles. The basic feature of all design problems is that all of them admit multiple solutions. The selection of the final solution depends on a variety of criteria that could include the size and cost of the substrate on which the components are assembled, the cost of components, manufacturability,

reliability, speed etc.

The course contents are designed to enable the students to design digital circuits of medium level of complexity taking the functional and hardware aspects in an integrated manner within the context of commercial and manufacturing constraints. However, no compromises are made with regard to theoretical aspects of the subject.

**Learning Objectives**

**Module 1: Number Systems and Codes (3)**

Number systems: Binary, octal, and hexa-decimal number systems, binary arithmetic. Codes: Binary code, excess-3 code, gray code, error detection and correction codes.

**Recall**

1. Describe the format of numbers of different radices?

2. What is parity of a given number?

**Comprehension**

1. Explain how a number with one radix is converted into a number with another radix.

2. Summarize the advantages of using different number systems. 3. Interpret the arithmetic operations of binary numbers.

4. Explain the usefulness of different coding schemes.

5. Explain how errors are detected and/or corrected using different codes. **Application**

1. Convert a given number from one system to an equivalent number in another system.

2. Illustrate the construction of a weighted code.

**Analysis: Nil**

**Synthesis: Nil**

**Evaluation: Nil**

****

Digital Electronics

Module 1: Number Systems and Codes - Number Systems

N.J. Rao

Indian Institute of Science

Numbers 

We use numbers

ñ to communicate

ñ to perform tasks

ñ to quantify

ñ to measure

ï Numbers have become symbols of the present era ï Many consider what is not expressible in terms of numbers is not worth knowing

December 2006 N.J. Rao M1L1 2

Number Systems in use 

Symbolic number system

ï uses Roman numerals (I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 and M = 1000)

ï still used in some watches

Weighted position system

ï Decimal system is the most commonly used ï Decimal numbers are based on Indian numerals ï Radix used is 10

December 2006 N.J. Rao M1L1 3

Other weighted position systems 

ï Advent of electronic devices with two states created a possibility of working with binary numbers

ï Binary numbers are most extensively used ï Binary system uses radix 2

ï Octal system uses radix 8

ï Hexa-decimal system uses radix 16

December 2006 N.J. Rao M1L1 4

Weighted Position Number System

ï Value associated with a digit is dependent on its positionï The value of a number is weighted sum of its digits 2357 = 2 x 10

3 + 3 x 10

2 + 5 x 10

1 + 7 x 10

0

ï Decimal point allows negative and positive powers of 10526.47 = 5 x 10

2 +2 x 10

1 + 6 x 10

0 + 4 x 10

-1

+ 7 x 10

-2

ï 10 is called the base or radix of the number system

December 2006 N.J. Rao M1L1 5

General positional number system

ï Any integer > 2 can serve as the radix ï Digit position ëií has weight ri.

ï The general form of a number is

dp-1 dp-2, .... d1, d0. d-1d-2.... d-n

p digits to the left of the point (radix point) and n digits tothe right of the point

December 2006 N.J. Rao M1L1 6

General positional number system(2)

ï The value of the number is

∑−

*p* 1

*ii d r*

D =

*i n*

=−

ï Leading and trailing zeros have no values ï The values dis can take are limited by the radix value ï A number like (357)5is incorrect

December 2006 N.J. Rao M1L1 7

Binary Number System

ï Uses 2 as its radix

ï Has only two numerals, 0 and 1 Example:

(N)2 = (11100110)2

ï It is an eight digit binary number ï The binary digits are also known as bits ï (N)2is an 8-bit number

December 2006 N.J. Rao M1L1 8

Binary numbers to Decimal Number 

(N)2 = (11100110)2

Its decimal value is given by,

(N)2 = 1 x 2

7 + 1 x 2

6 + 1 x 2

5 + 0 x 2

4 + 0 x 2

3

+ 1 x 2

2 + 1 x 2

1 + 0 x 2

0

= 128 + 64 + 32 + 0 + 0 + 4 + 2 + 0 = (230)10

December 2006 N.J. Rao M1L1 9

1 

8

Binary fractional number to Decimal number

ï A binary fractional number (N)2 = 101.101 ï Its decimal value is given by

(N)2 = 1 x 2

2 + 0 x 2

1 + 1 x 2

0

+ 1 x 2

-1 + 0 x 2

-2 + 1 x 2

-3

1

1

= 4 + ~~0 + 1 + + 0 +~~

2

8

= 5 + 0.5 + 0.125 = (5.625)10

December 2006 N.J. Rao M1L1 10

Some features of Binary Numbers

ï Require very long strings of 1s and 0s

ï Some simplification can be done through grouping ï 3-bit groupings: Octal (radix 8) groups three binary digitsDigits will have one of the eight values 0, 1, 2, 3, 4, 5, 6 and 7

ï 4-digit groupings: Hexa-decimal (radix 16)

Digits will have one of the sixteen values 0 through 15. Decimal values from 10 to 15 are designated as A (=10), B (=11), C (=12), D (=13), E (=14) and F (=15)

December 2006 N.J. Rao M1L1 11

Conversion of binary numbers 

Conversion to an octal number

ñ Group the binary digits into groups of three ñ (11011001)2 = (011) (011) (001) = (331)8 ï Conversion to an hexa-decimal number

ñ Group the binary digits into groups of four ñ (11011001)2 = (1101) (1001) = (D9)16

December 2006 N.J. Rao M1L1 12

Changing the radix of numbers 

ï Conversion requires, sometimes, arithmetic operations ï The decimal equivalent value of a number in any radix ∑−

*p* 1

*ii d r*

D =

*i n*

= −

Examples

(331)8 = 3 x 8

2 + 3 x 8

1 + 1 x 8

0 = 192 + 24 + 1 = (217)10

(D9)16 = 13 x 16

1 + 9 x 16

0 = 208 + 9 = (217)10

(33.56)8 = 3 x 8

1 + 3 x 8

0 + 5 x 8

-1 + 6 x 8

-2 = (27.69875)10

(E5.A)16 = 14 x 16

1 + 5 x 16

0 + 10 x 16

-1 = (304.625)10

December 2006 N.J. Rao M1L1 13

Conversion of decimal numbers tonumbers with radix r 

Represent a number with radix r as

D = ((... ((dn-1).r + dn-2) r + ....).r + d1).r + d0 To convert a number with radix r to a decimal number ▪ Divide the right hand side by r

▪ Remainder: d0

▪ Quotient: Q = ((... ((dn-1).r + dn-2) r + ....).r + d1 ▪ Division of Q by r gives d1 as the remainder ▪ so on

December 2006 N.J. Rao M1L1 14

Example of Conversion 

Quotient Remainder

156 ˜ 2 78 0

78 ˜ 2 39 0

39 ˜ 2 19 1

19 ˜ 2 9 1

9 ˜ 2 4 1

4 ˜ 2 2 0

2 ˜ 2 1 0

1 ˜ 2 0 1

(156)10 = (10011100)2

December 2006 N.J. Rao M1L1 15

Example of Conversion 

Quotient Remainder

678 ˜ 8 84 6

84 ˜ 8 10 4

10 ˜ 8 1 2

1 ˜ 8 0 1

(678)10 = (1246)8

Quotient Remainder

678 ˜ 16 42 6

42 ˜ 16 2 A

2 ˜ 16 0 2

(678)10 = (2A6)16

December 2006 N.J. Rao M1L1 16

Negative Numbers 

Sign-Magnitude representation

▪ ì+î sign before a number indicates it as a positive number

▪ ì-î sign before a number indicates it as a negative number

▪ Not very convenient on computers

ï Replace ì+î sign by ì0î and ì-î by ì1î

(+1100101)2 🡪 (01100101)2

(+101.001)2 🡪 (0101.001)2

(-10010)2 🡪 (110010)2

(-110.101)2--. (1110.101)2

December 2006 N.J. Rao M1L1 17

Representing signed numbers 

ï Diminished Radix Complement (DRC) or (r-1) - complement

ï Radix Complement (RXC) or r-complement Binary numbers

ï DRC is known as ìoneís-complementî

ï RXC is known as ìtwoís-complementî

Decimal numbers

ï DRC is known as 9ís-complement

ï RXC is known as 10ís-complement

December 2006 N.J. Rao M1L1 18

Oneís Complement Representation

The most significant bit (MSD) represents the sign If MSD is a ì0î

▪ The number is positive

▪ Remaining (n-1) bits directly indicate the magnitude If the MSD is ì1î

▪ The number is negative

▪ Complement of all the remaining (n-1) bits gives the magnitude

December 2006 N.J. Rao M1L1 19

Example: Oneís complement 

1111001🡪 (1)(111001)

ï First (sign) bit is 1: The number is negative ï Onesí Complement of 111001 🡪 000110 🡪 (6)10

December 2006 N.J. Rao M1L1 20

Range of n-bit numbers 

Oneís complement numbers:

0111111 + 63

0000110 --> + 6

0000000 --> + 0

1111111 --> + 0

1111001 --> - 6

1000000 --> - 63

ïì0î is represented by 000.....0 and 111.....1 ï 7- bit number covers the range from +63 to -63. ï n-bit number has a range from +(2

n-1- 1) to -(2

n-1- 1)

December 2006 N.J. Rao M1L1 21

Oneís complement of a number 

Complement all the digits

ï If A is an integer in oneís complement form, then oneís complement of A = -A

ï This applies to fractions as well.

A = 0.101 (+0.625)10

Oneís complement of A = 1.010, (-0.625)10 Mixed number

B = 010011.0101 (+19.3125)10

Oneís complement of B = 101100.1010 (- 19.3125)10

December 2006 N.J. Rao M1L1 22

Twoís Complement Representation

If MSD is a ì0î

▪ The number is positive

▪ Remaining (n-1) bits directly indicate the magnitude If the MSD is ì1î

▪ The number is negative

▪ Magnitude is obtained by complementing all the remaining (n-1) bits and adding a 1

December 2006 N.J. Rao M1L1 23

Example: Twoís complement 

1111010🡪 (1)(111010)

ï First (sign) bit is 1: The number is negative ï Complement 111010 and add 1🡪 000101 + 1 = 000110 = (6)10

December 2006 N.J. Rao M1L1 24

Range of n-bit numbers 

Twoís complement numbers:

0111111 + 63

0000110 + 6

0000000 + 0

1111010 - 6

1000001 - 63

1000000 - 64

ïì0î is represented by 000.....0

ï 7- bit number covers the range from +63 to -64. ï n-bit number has a range from +(2

n-1- 1) to -(2

n-1)

December 2006 N.J. Rao M1L1 25

Twoís complement of a number 

Complement all the digits and add *ë*1*í* to the LSB If A is an integer in oneís complement form, then ▪ Twoís complement of A = -A

This applies to fractions as well

▪ A = 0.101 (+0.625)10

▪ Twoís complement of A = 1.011🡪 (-0.625)10 Mixed number

▪ B = 010011.0101 (+19.3125)10

▪ Twoís complement of B = 101100.1011🡪 (- 9.3125)10

December 2006 N.J. Rao M1L1 26

**Number Systems**

We all use numbers to communicate and perform several tasks in our daily lives. Our present day world is characterized by measurements and numbers associated with everything. In fact, many consider if we cannot express something in terms of numbers is not worth knowing. While this is an extreme view that is difficult to justify, there is no doubt that quantification and measurement, and consequently usage of numbers, are desirable whenever possible. Manipulation of numbers is one of the early skills that the present day child is trained to acquire. The present day technology and the way of life require the usage of several number systems. Usage of decimal numbers starts very early in one’s life. Therefore, when one is confronted with number systems other than decimal, some time during the high-school years, it calls for a fundamental change in one’s framework of thinking.

There have been two types of numbering systems in use through out the world.

One type is symbolic in nature. Most important example of this symbolic numbering system is the one based on Roman numerals

I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 and M = 1000

IIMVII - 2007

While this system was in use for several centuries in Europe it is completely superseded by the weighted-position system based on Indian numerals. The Roman number system is still used in some places like watches and release dates of movies.

The weighted-positional system based on the use of radix 10 is the most commonly used numbering system in most of the transactions and activities of today’s world. However, the advent of computers and the convenience of using devices that have two well defined states brought the binary system, using the radix 2, into extensive use. The use of binary number system in the field of computers and electronics also lead to the use of octal (based on radix 8) and hex-decimal system (based on radix 16). The usage of binary numbers at various levels has become so essential that it is also necessary to have a good understanding of all the binary arithmetic operations.

Here we explore the weighted-position number systems and conversion from one system to the other.

**Weighted-Position Number System**

In a weighted-position numbering system using Indian numerals the value associated with a digit is dependent on its position. The value of a number is weighted sum of its digits.

Consider the decimal number 2357. It can be expressed as

2357 = 2 x 103 + 3 x 102 + 5 x 101 + 7 x 100

Each weight is a power of 10 corresponding to the digit’s position. A decimal point allows negative as well as positive powers of 10 to be used;

526.47 = 5 x 102 +2 x 101 + 6 x 100 + 4 x 10-1 + 7 x 10-2

Here, 10 is called the *base* or *radix* of the number system. In a general positional number system, the *radix* may be any integer *r* > 2, and a digit position i has weight *r*i. The general form of a number in such a system is

dp-1 dp-2, .... d1, d0 . d-1d-2 .... d-n

where there are *p* digits to the left of the point (called *radix point*) and *n* digits to the right of the point. The value of the number is the sum of each digit multiplied by the corresponding power of the *radix*.

D = ∑−

*p* 1

*i*

*d r*

*i*

*i n*

= −

Except for possible leading and trailing zeros, the representation of a number in positional system is unique (00256.230 is the same as 256.23). Obviously the values di’s can take are limited by the radix value. For example a number like (356)5, where the suffix 5 represents the radix will be incorrect, as there can not be a digit like 5 or 6 in a weighted position number system with radix 5.

If the radix point is not shown in the number, then it is assumed to be located near the last right digit to its immediate right. The symbol used for the radix point is a point (.). However, a comma is used in some countries. For example 7,6 is used, instead of 7.6, to represent a number having seven as its integer component and six as its fractional.

As much of the present day electronic hardware is dependent on devices that work reliably in two well defined states, a numbering system using 2 as its radix has become necessary and popular. With the radix value of 2, the binary number system

will have only two numerals, namely 0 and 1.

Consider the number (N)2 = (11100110)2.

It is an eight digit binary number. The binary digits are also known as *bits*. Consequently the above number would be referred to as an 8-bit number. Its decimal value is given by

(N)2 = 1 x 27 + 1 x 26 + 1 x 25 + 0 x 24 + 0 x 23 + 1 x 22 + 1 x 21 + 0 x 20 = 128 + 64 + 32 + 0 + 0 + 4 + 2 + 0 = (230)10

Consider a binary fractional number (N)2 = 101.101.

Its decimal value is given by

(N)2 = 1 x 22 + 0 x 21 + 1 x 20 + 1 x 2-1 + 0 x 2-2 + 1 x 2-3 = 4 + 0 + 1 + 12 + 0 + 18

= 5 + 0.5 + 0.125 = (5.625)10

From here on we consider any number without its radix specifically mentioned, as a decimal number.

With the radix value of 2, the binary number system requires very long strings of 1s and 0s to represent a given number. Some of the problems associated with handling large strings of binary digits may be eased by grouping them into three digits or four digits. We can use the following groupings.

③ Octal (radix 8 to group three binary digits)

③ Hexadecimal (radix 16 to group four binary digits)

In the octal number system the digits will have one of the following eight values 0, 1, 2, 3, 4, 5, 6 and 7.

In the hexadecimal system we have one of the sixteen values 0 through 15. However, the decimal values from 10 to 15 will be represented by alphabet A (=10), B (=11), C (=12), D (=13), E (=14) and F (=15).

Conversion of a binary number to an octal number or a hexadecimal number is very simple, as it requires simple grouping of the binary digits into groups of three or four. Consider the binary number 11011011. It may be converted into octal or hexadecimal numbers as

(11011001)2 = (011) (011) (001) = (331)8

= (1101) (1001) = (D9)16

Note that adding a leading zero does not alter the value of the number. Similarly for grouping the digits in the fractional part of a binary number, trailing zeros may be added without changing the value of the number.

**Number System Conversions**

In general, conversion between numbers with different radices cannot be done by simple substitutions. Such conversions would involve arithmetic operations. Let us work out procedures for converting a number in any radix to radix 10, and vice versa. The decimal equivalent value of a number in any radix is given by the formula

D = ∑−

*p* 1

*i i d r*

*i n*

= −

where *r* is the radix of the number and there are *p* digits to the left of the radix point and n digits to the right. Decimal value of the number is determined by converting each digit of the number to its radix-10 equivalent and expanding the formula using radix-10 arithmetic.

Some examples are:

(331)8 = 3 x 82 + 3 x 81 + 1 x 80 = 192 + 24 + 1 = (217)10 (D9)16 = 13 x 161 + 9 x 160 = 208 + 9 = (217)10 (33.56)8 = 3 x 81 + 3 x 80 + 5 x 8-1 + 6 x 8-2 = (27.69875)10 (E5.A)16 = 14 x 161 + 5 x 160 + 10 x 16-1 = (304.625)10 The conversion formula can be rewritten as

D = ((... ((dn-1).r + dn-2) r + ....).r + d1).r + d0

This forms the basis for converting a decimal number D to a number with radix r. If we divide the right hand side of the above formula by r, the remainder will be d0, and the quotient will be

Q = ((... ((dn-1).r + dn-2) r + ....).r + d1

Thus, d0 can be computed as the remainder of the long division of D by the radix r. As the quotient Q has the same form as D, another long division by r will give d1 as the remainder. This process can continue to produce all the digits of the number with radix r. Consider the following examples.

Quotient Remainder

156 ⎟ 2 78 0

78 ⎟ 2 39 0

39 ⎟ 2 19 1 19 ⎟ 2 9 1 9 ⎟ 2 4 1 4 ⎟ 2 2 0 2 ⎟ 2 1 0 1 ⎟ 2 0 1

(156)10 = (10011100)2

Quotient Remainder 678 ⎟ 8 84 6 84 ⎟ 8 10 4 10 ⎟ 8 1 2 1 ⎟ 8 0 1 (678)10 = (1246)8

Quotient Remainder 678 ⎟ 16 42 6 42 ⎟ 16 2 A 2 ⎟ 16 0 2 (678)10 = (2A6)16

**Representation of Negative Numbers**

In our traditional arithmetic we use the “+” sign before a number to indicate it as a positive number and a “-” sign to indicate it as a negative number. We usually omit the sign before the number if it is positive. This method of representation of numbers is called “sign-magnitude” representation. But using “+” and “-” signs on a computer is not convenient, and it becomes necessary to have some other convention to represent the signed numbers. We replace “+” sign with “0” and “-” with “1”. These two symbols already exist in the binary system. Consider the following examples:

(+1100101)2 (01100101)2

(+101.001)2 (0101.001)2

(-10010)2 (110010)2

(-110.101)2 (1110.101)2

In the sign-magnitude representation of binary numbers the first digit is always treated separately. Therefore, in working with the signed binary numbers in sign-magnitude form the leading zeros should not be ignored. However, the leading zeros can be ignored after the sign bit is separated. For example,

1000101.11 = - 101.11

While the sign-magnitude representation of signed numbers appears to be natural extension of the traditional arithmetic, the arithmetic operations with signed numbers in this form are not that very convenient, either for implementation on the computer or for hardware implementation. There are two other methods of representing signed numbers.

③ Diminished Radix Complement (DRC) or (r-1)-complement

③ Radix Complement (RX) or r-complement

When the numbers are in binary form

③ Diminished Radix Complement will be known as “one’s-complement” ③ Radix complement will be known as “two’s-complement”.

If this representation is extended to the decimal numbers they will be known as 9’s complement and 10’s-complement respectively.

**One’s Complement Representation**

Let A be an n-bit signed binary number in one’s complement form.

The most significant bit represents the sign. If it is a “0” the number is positive and if it is a “1” the number is negative.

The remaining (n-1) bits represent the magnitude, but not necessarily as a simple weighted number.

Consider the following one’s complement numbers and their decimal equivalents:

0111111 + 63

0000110 --> + 6

0000000 --> + 0

1111111 --> + 0

1111001 --> - 6

1000000 --> - 63

There are two representations of “0”, namely 000.....0 and 111.....1. From these illustrations we observe

③ If the most significant bit (MSD) is zero the remaining (n-1) bits directly indicate the magnitude.

③ If the MSD is 1, the magnitude of the number is obtained by taking the complement of all the remaining (n-1) bits.

For example consider one’s complement representation of -6 as given above.

③ Leaving the first bit ‘1’ for the sign, the remaining bits 111001 do not directly represent the magnitude of the number -6.

③ Take the complement of 111001, which becomes 000110 to determine the magnitude.

In the example shown above a 7-bit number can cover the range from +63 to -63. In general an n-bit number has a range from +(2n-1 - 1) to -(2n-1 - 1) with two representations for zero.

The representation also suggests that if A is an integer in one’s complement form, then one’s complement of A = -A

*One’s complement of a number is obtained by merely complementing all the digits*. This relationship can be extended to fractions as well.

For example if A = 0.101 (+0.625)10, then the one’s complement of A is 1.010, which is one’s complement representation of (-0.625)10. Similarly consider the case of a mixed number.

A = 010011.0101 (+19.3125)10

One’s complement of A = 101100.1010 (- 19.3125)10

This relationship can be used to determine one’s complement representation of negative decimal numbers.

**Example 1**: What is one’s complement binary representation of decimal number -75?

Decimal number 75 requires 7 bits to represent its magnitude in the binary form. One additional bit is needed to represent the sign. Therefore,

one’s complement representation of 75 = 01001011

one’s complement representation of -75 = 10110100

**Two’s Complement Representation**

Let A be an n-bit signed binary number in two’s complement form.

③ The most significant bit represents the sign. If it is a “0”, the number is positive, and if it is “1” the number is negative.

③ The remaining (n-1) bits represent the magnitude, but not as a simple weighted number.

Consider the following two’s complement numbers and their decimal equivalents: 0111111 + 63

0000110 + 6

0000000 + 0

1111010 - 6

1000001 - 63

1000000 - 64

There is only one representation of “0”, namely 000....0.

From these illustrations we observe

If most significant bit (MSD) is zero the remaining (n-1) bits directly indicate the magnitude.

If the MSD is 1, the magnitude of the number is obtained by taking the complement of all the remaining (n-1) bits and adding a 1.

Consider the two’s complement representation of -6.

③ We assume we are representing it as a 7-bit number.

③ Leave the sign bit.

③ The remaining bits are 111010. These have to be complemented (that is 000101) and a 1 has to be added (that is 000101 + 1 = 000110 = 6).

In the example shown above a 7-bit number can cover the range from +63 to -64. In general an n-bit number has a range from + (2n-1 - 1) to - (2n-1) with one representation for zero.

The representation also suggests that if A is an integer in two’s complement form, then Two’s complement of A = -A

*Two’s complement of a number is obtained by complementing all the digits and adding ‘1’ to the LSB*.

This relationship can be extended to fractions as well.

If A = 0.101 (+0.625)10, then the two’s complement of A is 1.011, which is two’s complement representation of (-0.625)10.

Similarly consider the case of a mixed number.

A = 010011.0101 (+19.3125)10

Two’s complement of A = 101100.1011 (- 19.3125)10

This relationship can be used to determine two’s complement representation of negative decimal numbers.

**Example 2**: What is two’s complement binary representation of decimal number -75?

Decimal number 75 requires 7 bits to represent its magnitude in the binary form. One additional bit is needed to represent the sign. Therefore,

Two’s complement representation of 75 = 01001011

Two’s complement representation of -75 = 10110101

**M1L1: Number Systems**

Multiple Choice Questions

1. Which number system is understood easily by the computer? (a) Binary (b) Decimal (c) Octal (d) Hexadecimal 2. How many symbols are used in the decimal number system? (a) 2 (b) 8 (c) 10 (d) 16

3. How are number systems generally classified?

a. Conditional or non conditional

b. Positional or non positional

c. Real or imaginary

d. Literal or numerical

4. What does (10)16 represent in decimal number system?

(a) 10 (b) 0A (c) 16 (d) 15

5. How many bits have to be grouped together to convert the binary number to its corresponding octal number?

(a) 2 (b) 3 (c) 4 (d) 5

6. Which bit represents the sign bit in a signed number system?

a. Left most bit

b. Right most bit

c. Left centre

d. Right centre

7. The ones complement of 1010 is

(a) 1100 (b) 0101 (c) 0111 (d) 1011

8. How many bits are required to cover the numbers from +63 to -63 in one’s complement representation?

(a) 6 (b) 7 (c) 8 (d) 9

**M1L1: Number Systems**

**Problems**

1. Perform the following number system conversions:

(a) 101101112 = ?10 (b) 567410 = ?2

(c) 100111002 = ?8 (d) 24538 = ?2

(e) 1111000102 = ?16 (f) 6893410 = ?2

(g) 10101.0012 = ?10 (h) 6FAB716 = ?10

(i) 11101.1012 = ?8 (j) 5623816 = ?2

2. Convert the following hexadecimal numbers into binary and octal numbers (a) 78AD (b) DA643 (c) EDC8

(d) 3245 (e) 68912 (f) AF4D

3. Convert the following octal numbers into binary and hexadecimal numbers (a) 7643 (b) 2643 (c) 1034

(d) 3245 (e) 6712 (f) 7512

4. Convert the following numbers into binary:

(a) 123610 (b) 234910 (c) 345.27510

(d) 45678 (e) 45.658 (f) 145.238

(g) ADF516 (h) AD.F316 (i) 12.DA16

5. What is the range of unsigned decimal values that can be represented by 8 bits? 6. What is the range of signed decimal values that can be represented by 8 bits? 7. How many bits are required to represent decimal values ranging from 75 to -75?

8. Represent each of the following values as a 6-bit signed binary number in one’s complement and two’s complement forms.

(a) 28 (b) -21 (c) -5 (d) -13

9. Determine the decimal equivalent of two’s complement numbers given below: (a) 1010101 (b) 0111011 (c) 11100010



Digital Electronics

Module 1:Number Systems and Codes - Codes

N.J. Rao

Indian Institute of Science

Need for Coding 

Information sent over a noisy channel is likely to be distorted

Information is coded to facilitate

▪ Efficient transmission

▪ Error detection

▪ Error correction

December 2006 N.J. Rao M1L2 2

Coding 

ï Coding is the process of altering the characteristics of information to make it more suitable for intended application

ï Coding schemes depend on

▪ Security requirements

▪ Complexity of the medium of transmission ▪ Levels of error tolerated

▪ Need for standardization

December 2006 N.J. Rao M1L2 3

Decoding 

ï Decoding is the process of reconstructing source information from the received encoded information ï Decoding can be more complex than coding if there is noprior knowledge of coding schemes

December 2006 N.J. Rao M1L2 4

Bit combinations 

Bit - a binary digit 0 or 1

Nibble - a group of four bits

Byte - a group of eight bits

Word - a group of sixteen bits;

(Sometimes used to designate 32 bit or 64 bit groups of bits)

December 2006 N.J. Rao M1L2 5

Binary coding 

Assign each item of information a unique combination of 1sand 0s

▪ n is the number of bits in the code word ▪ x be the number of unique words

If n = 1, then x = 2 (0, 1)

n = 2, then x = 4 (00, 01, 10, 11)

n = 3, then x = 8 (000,001,010 ...111)

n = j, then x = 2

j

December 2006 N.J. Rao M1L2 6

Number of bits in a code word 

x: number of elements to be coded binary coded format x < 2

j

or j > log2x

> 3.32 log10x

j is the number of bits in a code word.

December 2006 N.J. Rao M1L2 7

Example: Coding of alphanumericinformation 

ï Alphanumeric information: 26 alphabetic characters + 10decimals digits = 36 elements

j > 3.32 log1036

j > 5.16 bits

ï Number of bits required for coding = 6 ï Only 36 code words are used out of the 64 possible codewords

December 2006 N.J. Rao M1L2 8

Some codes for consideration 

ï Binary coded decimal codes

ï Unit distance codes

ï Error detection codes

ï Alphanumeric codes

December 2006 N.J. Rao M1L2 9

Binary coded decimal codes 

Simple Scheme

ï Convert decimal number inputs into binary form ï Manipulate these binary numbers

ï Convert resultant binary numbers back into decimal numbers

However, it

ï requires more hardware

ï slows down the system

December 2006 N.J. Rao M1L2 10

Binary coded decimal codes 

ï Encode each decimal symbol in a unique string of 0s and 1s

ï Ten symbols require at least four bits to encode ï There are sixteen four-bit groups to select ten groups. ï There can be 30 x 10

10(16C10.10!) possible codes

ï Most of these codes will not have any special properties

December 2006 N.J. Rao M1L2 11

Example of a BCD code 

ï Natural Binary Coded Decimal code (NBCD) ï Consider the number (16.85)10

(16.85)10 = (0001 0110 . 1000 0101) NBCD 1 6 8 5

ï NBCD code is used in calculators

December 2006 N.J. Rao M1L2 12

How do we select a coding scheme?

It should have some desirable properties ï ease of coding

ï ease in arithmetic operations

ï minimum use of hardware

ï error detection property

ï ability to prevent wrong output during transitions

December 2006 N.J. Rao M1L2 13

Weighted Binary Coding 

Decimal number (A)10

Encoded in the binary form as ëa3 a2 a1 a0í

w3, w2, w1 and w0 are the weights selected for a given code

(A)10 = w3a3 + w2a2 + w1a1 +w0a0

The more popularly used codes have these weights as w3 w2 w1 w0

8 4 2 1

2 4 2 1

8 4 -2 -1

December 2006 N.J. Rao M1L2 14

Binary codes for decimal numbers

|  |  |  |
| --- | --- | --- |
| Decimal digit | Weight  8 4 2 1 | Weights  2 4 2 1 |
| 0 | 0 0 0 0 | 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 1 |
| 2 | 0 0 1 0 | 0 0 1 0 |
| 3 | 0 0 1 1 | 0 0 1 1 |
| 4 | 0 1 0 0 | 0 1 0 0 |
| 5 | 0 1 0 1 | 1 0 1 1 |
| 6 | 0 1 1 0 | 1 1 0 0 |
| 7 | 0 1 1 1 | 1 1 0 1 |
| 8 | 1 0 0 0 | 1 1 1 0 |

Weights

8 4 -2 -1

0 0 0 0

0 1 1 1

0 1 1 0

0 1 0 1

0 1 0 0

1 0 1 1

1 0 1 0

1 0 0 1

1 0 0 0

9 1 0 0 1 1 1 1 1 1 1 1 1 December 2006 N.J. Rao M1L2 15

Binary coded decimal numbers 

ï The unused six combinations are illegal

ï They may be utilised for error detection purposes. ï Choice of weights in a BCD codes

1. Self-complementing codes

2. Reflective codes

December 2006 N.J. Rao M1L2 16

Self complementing codes 

Logical complement of a coded number is also its arithmetic complement

Example: 2421 code

Nineís complement of (4)10 = (5)10

2421 code of (4)10 = 0100

Complement 0f 0100 = 1011 = 2421 code for (5)10 = (9 - 4)10.

A necessary condition: Sum of its weights should be 9.

December 2006 N.J. Rao M1L2 17

Other self complementing codes 

Excess-3 code (not weighted)

Add 0011 (3) to all the 8421 coded numbers Another example is 631-1 weighted code

December 2006 N.J. Rao M1L2 18

Examples of self-complementary codes

|  |  |
| --- | --- |
| Decimal  Digit | Excess-3  Code |
| 0 | 0011 |
| 1 | 0100 |
| 2 | 0101 |
| 3 | 0110 |
| 4 | 0111 |
| 5 | 1000 |
| 6 | 1001 |
| 7 | 1010 |
| 8 | 1011 |
| 9 | 1100 |

631-1 Code

2421 Code

0011 0000 

0010 0001

0101 0010

0111 0011

0110 0100

1001 1011

1000 1100

1010 1101

1101 1110

1100 1111

December 2006 N.J. Rao M1L2 19

Reflective code 

ï Imaged about the centre entries with one bit changed Example

ï 9ís complement of a reflected BCD code word is formedby changing only one of its bits

December 2006 N.J. Rao M1L2 20

Examples of reflective BCD codes

|  |
| --- |
| Decimal  Digit |
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |

Code-A Code-B

0000 0100 0001 1010 0010 1000 0011 1110 0100 0000 1100 0001 1011 1111 1010 1001

8 1001 1011 9 1000 0101

December 2006 N.J. Rao M1L2 21

Unit Distance Codes 

Adjacent codes differ only in one bit ïìGray codeî is the most popular example ï Some of the Gray codes have also the reflectiveproperties

December 2006 N.J. Rao M1L2 22

3-bit and 4-bit Gray codes

|  |  |  |
| --- | --- | --- |
| Decimal  Digit | 3-bit GrayCode | 4-bit GrayCode |
| 0 | 000 | 0000 |
| 1 | 001 | 0001 |
| 2 | 011 | 0011 |
| 3 | 010 | 0010 |
| 4 | 110 | 0110 |
| 5 | 111 | 0111 |
| 6 | 101 | 0101 |
| 7 | 100 | 0100 |
| 8 | - | 1100 |

|  |
| --- |
| DecimalDigit |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |

3-bit Gray

4-bit Gray

Code

Code

9 - 1101

- 1111 - 1110 - 1010 - 1011 - 1001 - 1000

December 2006 N.J. Rao M1L2 23

More examples of Unit Distance Codes

|  |  |
| --- | --- |
| Decimal  Digit | UDC-1 |
| 0 | 0000 |
| 1 | 0100 |
| 2 | 1100 |
| 3 | 1000 |
| 4 | 1001 |
| 5 | 1011 |
| 6 | 1111 |
| 7 | 0111 |
| 8 | 0011 |

UDC-2 UDC-3

0000 0000 0001 1000 0011 1001 0010 0001 0110 0011 1110 0111 1111 1111 1101 1011 1100 1010

9 0001 0100 0010

December 2006 N.J. Rao M1L2 24

3-bit simple binary coded shaft encoder111 000

110 101

001 

0 0 1

010

100 011

Can lead to errors (001 🡪 011 🡪 010)

December 2006 N.J. Rao M1L2 25

Shaft encoder disk using 3-bit Gray code

100 000



101 

001



0 0 1

111 

011

110 010 

December 2006 N.J. Rao M1L2 26

Constructing Gray Code 

ï The bits of Gray code words are numbered from right toleft, from 0 to n-1.

ï Bit i is 0 if bits i and i+1 of the corresponding binary codeword are the same, else bit i is 1

ï When i+1 = n, bit n of the binary code word is consideredto be 0

Example: Consider the decimal number 68. (68)10 = (1000100)2

Binary code : 1 0 0 0 1 0 0

Gray code : 1 1 0 0 1 1 0

December 2006 N.J. Rao M1L2 27

Convert a Gray coded number to astraight binary number 

ï Scan the Gray code word from left to right

ï All the bits of the binary code are the same as those of the Gray code until the first 1 is encountered, including the first 1

ï 1ís are written until the next 1 is encountered, in which case a 0 is written.

ï 0ís are written until the next 1 is encountered, in which case a 1 is written.

Examples

Gray code : 1 1 0 1 1 0

Binary code: 1 0 0 1 0 0

Gray code : 1 0 0 0 1 0 1 1

Binary code: 1 1 1 1 0 0 1 0

December 2006 N.J. Rao M1L2 28

Alphanumeric Code (ASCII)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| b4 | b3 | b2 | b1 |  |  |  | b7 | b6 b5 |
|  |  |  |  | 000 | 001 | 010 | 011 | 100 101 110 |
| 0 | 0 | 0 | 0 | NUL | DLE | SP | 0 | @ P ë |
| 0 | 0 | 0 | 1 | SOH | DC1 | ! | 1 | A Q a |
| 0 | 0 | 1 | 0 | STX | DC2 | ì | 2 | B R b |
| 0 | 0 | 1 | 1 | ETX | DC3 | # | 3 | C S c |
| 0 | 1 | 0 | 0 | EOT | DC4 | $ | 4 | D T d |
| 0 | 1 | 0 | 1 | ENQ | NAK | % | 5 | E U e |
| 0 | 1 | 1 | 0 | ACK | SYN | & | 6 | F V f |
| 0 | 1 | 1 | 1 | BEL | ETB | , | 7 | G W g |
| 1 | 0 | 0 | 0 | BS | CAN | ( | 8 | H X h |
| 1 | 0 | 0 | 1 | HT | EM | ) | 9 | I Y i |
| 1 | 0 | 1 | 0 | LF | SUB | \* | : | J Z j |
| 1 | 0 | 1 | 1 | VT | ESC | + | ; | K [ k |
| 1 | 1 | 0 | 0 | FF | FS | , | < | L \ l |
| 1 | 1 | 0 | 1 | CR | GS | - | = | M ] m |

111

p q r

s

t

u

v

w

x

y

z

{

|

}

1 1 1 0 SO RS . > N Λ n ~ 1 1 1 1 SI US / ? O - o DEL

December 2006 N.J. Rao M1L2 29

Other alphanumeric codes 

ï EBCDIC (Extended Binary Coded Decimal Interchange Code)

ï 12-bit Hollerith code

are in use for some applications

December 2006 N.J. Rao M1L2 30

Error Detection and Correction

ï Error rate cannot be reduced to zero

ï We need a mechanism of correcting the errors that occur ï It is not always possible or may prove to be expensive ï It is necessary to know if an error occurred ï If an occurrence of error is known, data may be retransmitted

ï Data integrity is improved by encoding ï Encoding may be done for error correction or merely for error detection.

December 2006 N.J. Rao M1L2 31

Encoding for data integrity 

ï Add a special code bit to a data word ï It is called the ëParity Bitî

ï Parity bit can be added on an ëoddí or ëevení basis

December 2006 N.J. Rao M1L2 32

Parity 

Odd Parity

ï The number of 1ís, including the parity bit, should be oddExample: S in ASCII code is

(S) = (1010011)ASCII

S, when coded for odd parity, would be shown as (S) = (11010011)ASCII with odd parity

Even Parity

ï The number of 1ís, including the parity bit, should be evenWhen S is encoded for even parity

(S) = (01010011) ASCII with even parity

December 2006 N.J. Rao M1L2 33

Error detection with parity bits 

ï If odd number of 1ís occur in the received data word coded for even parity then an error occurred ï Single or odd number bit errors can be detected ï Two or even number bit errors will not be detected

December 2006 N.J. Rao M1L2 34

Error Correction 

ï Parity bit allows us only to detect the presence of one bit error in a group of bits

ï It does not enable us to exactly locate the bit that changed

ï Parity bit scheme can be extended to locate the faulty bit in a block of information

December 2006 N.J. Rao M1L2 35

Single error detecting and single error correcting coding scheme 

The bits are conceptually arranged in a two-dimensional array, and parity bits are provided to check both the rows and the columns

Row

Parity

Information bits Column parity bits

bits

December 2006 N.J. Rao M1L2 36