Switching Theory and Digital Design

Deptt. Of CS&E, IIT Madras

Digital  vs  Analog
Digital Systems – More Accuracy and Reliability

Analog Systems – Errors introduced by Noise

Digital Systems

System Design

Logic Design

Circuit Design
Digital Systems

- **System Design**
  - Issues in breaking the overall system into sub-systems and specifying the characteristics of each sub-system (e.g. Memory, CPU, I/O Bus control).

- **Logic Design**
  - Determine how to interconnect basic logic building blocks to perform a specific function. E.g. Connect logic gates and Flip-Flops

- **Circuit Design**
  - Specify the interconnection of specific components. E.g. Resistors, Diodes, Transistors etc. to form a gate and other logic building blocks.
Combinational and Sequential

**Combinational Circuits.**
- Operation depends on the current state of the input or *Present State* but not the past state

**Sequential Circuits**
- Output depends on the previous and the present stages. Hence it has some memory (not required in Combinational ).
Combinational Circuits

- Design of Combinational Network involves the interconnection of logic gates

- Output input relationship can be described mathematically using the Boolean algebra

- CLN (Combinational Logic n/w) be designed as
  - Derive a table or algebraic logic Equation
  - Simplify using the K-Maps or Q-M procedures
  - Use different gates to realize the reduced logic equation
Sequential Circuits

- Use memory elements – Flip-Flops (F/F)
- F/F are interconnected with gates to form counters and Registers
- General Sequential Circuits designed using *Timing Diagrams*.
- Combinational and Sequential circuit design technique are used to build ADDER, SUB, MULT and DIV
- *Asynchronous Seq. N/w are most difficult.*
Binary Arithmetic

- I/p and o/p of switching devices assume only different and distinct values

- So Binary Number Systems is used in all digital systems

- Decimal System
  \[(938.75)_{10} = 9 \times 10^2 + 3 \times 10^1 + 8 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}\]

- Binary System
  \[(1101.01)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}\]
For any number system, Radix or base \( R \) is a +ve integer. (Here \( R = 10 \) for decimal, 2 for binary)

For base \( R \), \((R-1)\) digits are used for representing the number

\[
M = (a_n a_{n-1} \ldots a_0. a_{-1} a_{-2} \ldots a_{-m})_R
\]

\[
= a_n R^n + a_{n-1} R^{n-1} + \ldots + a_0 R^0 + a_{-1} R^{-1} + a_{-2} R^{-2} \ldots + a_{-m} R^{-m}
\]

where, \( 0 \leq a_i \leq (R-1) \)

If \( R > 10 \)

\( 0,1,\ldots,9, A, B, C, D, E, F \)

E.g. \((A2F)_{16} = 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0 = (2607)_{10}\)
Any base to decimal:

<table>
<thead>
<tr>
<th>Base</th>
<th>Digit1</th>
<th>Digit2</th>
<th>Digit3</th>
<th>Digit4</th>
<th>Digit5</th>
<th>Digit6</th>
<th>Digit7</th>
<th>Digit8</th>
<th>Digit9</th>
<th>Digit10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Octal</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$3 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1536</td>
<td>448</td>
<td>16</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>7</td>
<td>D</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$7 \times 16^2 + 13 \times 16^1 + 1 \times 16^0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1792</td>
<td>208</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Read about conversion from any base to another.
\[ M = (a_n a_{n-1} \ldots a_0 . a_{-1} a_{-2} \ldots a_{-m})_R \]
\[ = a_n R^n + a_{n-1} R^{n-1} + \ldots + a_0 R^0 + a_{-1} R^{-1} + a_{-2} R^{-2} \ldots + a_{-m} R^{-m} \]
where, \( 0 \leq a_i \leq (R-1) \)

**Steps to convert from any base to another:**
- First Convert to decimal
- Convert decimal output to new base

**Solve:**
\((231.3)_4 = (??)_7\)

\((231.3)_4 = (45.75)_{10}\)

\((45.75)_{10} = \)
- DIV(45,7) \(\Rightarrow\) Q1 = 6; Rem1 = 3;
- DIV(Q1,7) \(\Rightarrow\) Q2 = 0; Rem2 = 6;
- Mult(.75 * 7) \(\Rightarrow\) INT1 = 5; Frac1 = 0.25 ; \textbf{ANS:} \(63 . 51 51 \ldots \)_7
- Mult(.25 * 7) \(\Rightarrow\) INT1 = 1; Frac1 = 0.75 ;
- Mult(.75 * 7) \(\Rightarrow\) INT1 = 5; Frac1 = 0.25 ;
- Mult(.25 * 7) \(\Rightarrow\) INT1 = 1; Frac1 = 0.75 ; and so on
A few special cases of base conversion:

Binary to Octal: \((11\ 010\ 111\ 110\ .\ 001\ 1)_2\)
\((3\ 2\ 7\ 6\ .\ 1\ 4)_8\)

Binary to Hexadecimal: \((100\ 1101\ .\ 0101\ 11)_2\)
\((4\ D\ .\ 5\ C)_{16}\)

What do you think of the floating point number representation:

\[ M \beta^E \]
Convert a Base 3 number into a Base 5 number in one step?

Each subsequent digit starting from the right will be the remainder when you divide your current working total by \((12)_3 = (5 \text{ in base } 3)\).

The only tricky parts here are keeping track of what base you're in and remembering to convert to base 5 for each digit at the end.

Let's demonstrate with an example:

Convert 12210 base 3 (156 in dec) to base 5:

All math is in base 3:
Another example:

Convert 10211 base 3 (103 in dec) to base 5:

10211 = 12 * 202 + 10
202 = 12 * 11 + 0
11 = 12 * 0 + 11

Taking these digits in order: 11, 0, 10

4, 0, 3
403 base 5
Binary Codes

- Number representation Internal to the computer – *Binary*
- For human beings – Decimal
- So, any I/O interface must *Convert* from Decimal to Binary
- Finally the Binary bits are transmitted in terms of binary signals
<table>
<thead>
<tr>
<th>Binary</th>
<th>Unsigned</th>
<th>-N Signed Mag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>-1</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-2</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>-3</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>-4</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>-5</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>-6</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-7</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-7</td>
</tr>
</tbody>
</table>
### Binary Codes

<table>
<thead>
<tr>
<th>N decimal</th>
<th>N binary</th>
<th>(-N) signed mag.</th>
<th>(-N) 1's compl.</th>
<th>(-N) 2's compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000001</td>
<td>10000001</td>
<td>11111110</td>
<td>11111111</td>
</tr>
<tr>
<td>2</td>
<td>00000010</td>
<td>10000010</td>
<td>11111101</td>
<td>11111110</td>
</tr>
<tr>
<td>3</td>
<td>00000011</td>
<td>10000011</td>
<td>11111100</td>
<td>11111101</td>
</tr>
<tr>
<td>4</td>
<td>00000100</td>
<td>10001000</td>
<td>11101111</td>
<td>11111100</td>
</tr>
<tr>
<td>5</td>
<td>00000101</td>
<td>10001010</td>
<td>11101101</td>
<td>11111101</td>
</tr>
<tr>
<td>6</td>
<td>00000110</td>
<td>10001100</td>
<td>11101100</td>
<td>11111110</td>
</tr>
<tr>
<td>7</td>
<td>00000111</td>
<td>10001111</td>
<td>11100001</td>
<td>11111101</td>
</tr>
<tr>
<td>8</td>
<td>00001000</td>
<td>10100100</td>
<td>11011011</td>
<td>11111100</td>
</tr>
<tr>
<td>9</td>
<td>00001001</td>
<td>10100101</td>
<td>11011010</td>
<td>11111101</td>
</tr>
<tr>
<td>10</td>
<td>00001010</td>
<td>10101000</td>
<td>11011011</td>
<td>11111110</td>
</tr>
<tr>
<td>20</td>
<td>00010100</td>
<td>10101010</td>
<td>11011001</td>
<td>11111110</td>
</tr>
<tr>
<td>30</td>
<td>00011110</td>
<td>10111110</td>
<td>11000001</td>
<td>11111110</td>
</tr>
<tr>
<td>40</td>
<td>00101000</td>
<td>11010000</td>
<td>10111011</td>
<td>11111110</td>
</tr>
<tr>
<td>50</td>
<td>00110010</td>
<td>11010010</td>
<td>10111010</td>
<td>11111110</td>
</tr>
<tr>
<td>60</td>
<td>00111100</td>
<td>11011100</td>
<td>10111001</td>
<td>11111110</td>
</tr>
<tr>
<td>70</td>
<td>01000110</td>
<td>11100010</td>
<td>10111101</td>
<td>11111110</td>
</tr>
<tr>
<td>80</td>
<td>01010000</td>
<td>11101000</td>
<td>10111111</td>
<td>11111110</td>
</tr>
<tr>
<td>90</td>
<td>01011010</td>
<td>11101100</td>
<td>10100101</td>
<td>11111110</td>
</tr>
<tr>
<td>100</td>
<td>01100100</td>
<td>11101101</td>
<td>10011011</td>
<td>11111110</td>
</tr>
<tr>
<td>127</td>
<td>01111111</td>
<td>11111111</td>
<td>10000000</td>
<td>11111110</td>
</tr>
<tr>
<td>128</td>
<td>Nonexistent</td>
<td>Nonexistent</td>
<td>Nonexistent</td>
<td>10000000</td>
</tr>
</tbody>
</table>
BCD

- BCD format needs 4 bits for each decimal digit
- Is a way to represent binary numbers in decimal format
- BCD is not the same as binary representation!

\[ 1234_{10} = 1 \quad 2 \quad 3 \quad 4_{10} = \]

\[
= 0001 \quad 0010 \quad 0011 \quad 0101 \quad \text{BCD} = \\
= 1001000110101 \quad \text{BCD} = \\
= 10011010010 \quad 2 = \\
= 4D2_{16}
\]
Excess-3 code

- Excess-3 code: Given a decimal digit \( n \), its corresponding excess-3 codeword is binary code \((n+3)_2\)

- Example:
  
  \[
  \begin{align*}
  n = 5 & \rightarrow n+3 = 8 \rightarrow 1000_{\text{excess-3}} \\
  n = 0 & \rightarrow n+3 = 3 \rightarrow 0011_{\text{excess-3}}
  \end{align*}
  \]
Grey-Code

- Grey-code is one where only one bit changes at a time.
- Codes of successive decimal digits differ in exactly one bit.

The following tables show the difference between three-bit Binary numbers and Gray-coded numbers.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Gray-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Gray-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>111</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>100</td>
</tr>
</tbody>
</table>

Problem of uniqueness in representation
2-out-of-5 code

Exactly 2 out of the 5 bits are 1 for every valid code combination.

Good error checking properties.

The Gray code and 2-out-of-5 code are non-weighted codes. The decimal values of a coded digit cannot be computed by a simple formula, in case of non-weighted codes.
## Table of Binary Codes

<table>
<thead>
<tr>
<th>Decimal</th>
<th>BCD</th>
<th>6-3-1-1</th>
<th>Excess 3</th>
<th>2-out of 5</th>
<th>Gray code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
<td>11</td>
<td>00011</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
<td>100</td>
<td>00101</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>11</td>
<td>101</td>
<td>00110</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>100</td>
<td>110</td>
<td>01001</td>
<td>0010</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>101</td>
<td>111</td>
<td>01010</td>
<td>0110</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>111</td>
<td>1000</td>
<td>01100</td>
<td>1110</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>1000</td>
<td>1001</td>
<td>10001</td>
<td>1010</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>1001</td>
<td>1010</td>
<td>10010</td>
<td>1011</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1011</td>
<td>1011</td>
<td>10100</td>
<td>1001</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1100</td>
<td>1100</td>
<td>11000</td>
<td>1000</td>
</tr>
</tbody>
</table>
Binary Addition

Adding binary numbers:
1 + 0 = 0 + 1 = 1; 0 + 0 = 0; 1 + 1 = 0, with carry 1

<table>
<thead>
<tr>
<th>Addend</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Augend</td>
<td>+0</td>
<td>+1</td>
<td>+0</td>
<td>+1</td>
</tr>
<tr>
<td>Sum</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Carry</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Add 109_{10} to 136_{10}:
01101101 + 10001000
= 11110101

= 235_{10}
Binary Subtraction

- Unsigned numbers: minus sign is not explicitly represented.
- Given 2 binary numbers M and N, find M-N:
  - Case I: $M \geq N$, thus, MSB of Borrow is 0
    - B000110
    - M11110
    - N-10011
    - Dif01011
    - Result is Correct

  - Case II: $N > M$, thus MSB of Borrow is 1
    - B111000
    - M10011
    - N-11110
    - Dif10101
    - Result requires correction!
### 1-bit Adder unit:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### 1-bit Subtractor unit:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

C – Carry to next column;

B – Borrow from next column
Some more work out examples:

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
- & 1 & 0 & 0 \\
1 & 0 & 1 & 0
\end{array}
\]

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 \\
- & 1 & 0 & 1 \\
1 & 0 & 1 & 1
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
- & 1 & 1 & 1 \\
1 & 1 & 0 & 1
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0
\end{array}
\]
In general, if $N > M$, $\text{Dif} = M-N+2^n$, where, $n = \# \text{ bits}$.

In Case II of the previous example,

$$\text{Dif} = 19 - 30 + 2^5 = 21.$$ 

To correct the magnitude of $\text{Dif}$, which should be $N-M$, calculate $2^n - (M-N+2^n)$.

$$= 32 - 21 = 11 \text{ (check out binary)}$$

This is known as the 2's complement of $\text{Dif}$. 
To subtract two n-bit numbers, M-N, in base 2:

- Find M-N.
- If MSB of Borrow is 0, then M ≥ N. Result is positive and correct.
- If MSB of Borrow is 1, then N > M. Result is negative and its magnitude must be corrected by subtracting it from $2^n$ (find its 2’s complement).
Given \( M = 01100100 \) and \( N = 10010110 \), find \( M - N \).

\[
\begin{array}{c|c|c}
\text{B} & 1 & 0 0 1 1 1 1 0 0 \\
\text{M} & 0 & 1 1 0 0 1 0 0 \\
\text{N} & -1 & 0 0 1 0 1 1 0 \\
\text{Dif} & 1 & 1 0 0 1 1 1 0
\end{array}
\]

Check the 2’s complement relationship of the two results.
**Binary Multiplication**

- **Example:**
  - Multiplier $A = A_1A_0$ and multiplicand $B = B_1B_0$
  - Find $C = AxB$:

  $\begin{array}{cccc}
  & B_1 & B_0 \\
  x & A_1 & A_0 \\
  \hline
  & A_0B_1 & A_0B_0 & \\
  + & A_1B_1 & A_1B_0 & \\
  \hline
  C_3 & C_2 & C_1 & C_0
  \end{array}$

- **Multiplication rules**
  - $0 \times 0 = 0$
  - $0 \times 1 = 0$
  - $1 \times 0 = 0$
  - $1 \times 1 = 1$
Example

Multiply \((5)_{10}\) by \((3)_{10}\)  

\[
\begin{array}{c}
101 \\
\times 11 \\
\
101 \\
101x \\
1111
\end{array}
\]

Result \((15)_{10}\) or \((1111)_{2}\)

Multiply \((13)_{10}\) \(\times\) \((10)_{10}\)

\[
\begin{array}{c}
1101 \\
\times 1010 \\
0000 \\
1101x \\
0000xx \\
1101xxx \\
10000010
\end{array}
\]

Result \((130)_{10}\) or \((10000010)_{2}\)
Binary Division

Divide \((59)_{10}\) by \((3)_{10}\)

\[
11)111011(\quad 1\quad 0\quad 0\quad 1\quad 1
\]

\[
11
\]

\[
-----------
101
\]

\[
101
11
\]

\[
-----------
101
\]

\[
101
11
\]

\[
-----------
10
\]

Quotient – \((10011)_2 / (19)_{10}\)

Remainder – \((10)_2 / (2)_{10}\)
REFERENCES


