Course Overview and Introduction
(Class 1.1 – 1/15/2013)

CSE 2441 – Introduction to Digital Logic
Spring 2013
Instructor – Bill Carroll, Professor of CSE
Today’s Topics

• Course overview
  – Syllabus
  – Schedule
• Digital systems
  – Analog vs digital
  – Technology evolution
  – Design evolution
• Review of number systems
  – Important bases
  – Arithmetic
  – Base conversion
• Computer codes
  – Numbers
  – Binary codes
Analog Systems vs Digital Systems

• Analog – process continuous time-varying signals

• Digital – process discrete time-varying signals that can be represented by binary digits (bits)

• Digital advantages
  – Functionality/Flexibility/Programmability
  – Advancing technology
  – Economy/costs
  – Design tools
  – Reproducibility of results
Digital Technology Evolution

• Abacus
• Mechanical gears and switches
• Electrical relays
• Vacuum tubes
• Transistors and diodes (discrete components)
• Small scale integrated circuits (DTL, TTL, CMOS)
• Medium scale integrated circuits (TTL, CMOS)
• Large Scale integrated circuits (CMOS)
• Very large scale integrated circuits (CMOS)
• Fixed logic vs programmable logic
• Application Specific Integrated Circuits (ASICs)
Integrated Circuit Complexity Trends
Digital Design Evolution

• Design tools
  – Pencil/paper/templates
  – Simulators
  – Schematic capture
  – Placement/routing
  – Test generation
  – Hardware description languages
  – HDL compilers
  – Integrated design suites, e.g., Quartus II

• Laboratory equipment
  – Parts and a soldering iron
  – Voltmeters and scopes
  – Printed circuit (PC) boards
  – Solder less breadboards
  – Wirewrap technology
  – Custom PC boards
  – Logic analyzers
  – Automated test equipment
  – Development boards
Design Process

1. Vision
2. Develop Specifications
3. Initial Design Capture (Schematic or HDL)
4. Simulate the Design
   - Design Correct?
     - No: Revise the Design
   - Yes: Construct a Prototype
     - Specs Met?
       - No: Revise the Design
       - Yes: Done
Review of Number Systems

- **Decimal** numbers
  - Digits = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}
  - \((26.75)_{10} = 2 \times 10^2 + 6 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}\)

- **Binary** numbers
  - Digits = \{0, 1\}
  - \((11010.11)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}\)
  - \((26.75)_{10}\)

- **Octal** numbers
  - Digits = \{0, 1, 2, 3, 4, 5, 6, 7\}
  - \((127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}\)

- **Hexadecimal** numbers
  - Digits = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}
  - \((B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46,687)_{10}\)
## Important Number Systems

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>
Representation of Numbers

• **Positional Notation**

\[ N = (a_{n-1}a_{n-2} \ldots a_1a_0 \cdot a_{-1}a_{-2} \ldots a_{-m})_r \]

where \( . \) = radix point

\[ r = \text{radix or base} \]
\[ n = \text{number of integer digits to the left of the radix point} \]
\[ m = \text{number of fractional digits to the right of the radix point} \]
\[ a_{n-1} = \text{most significant digit (MSD)} \]
\[ a_{-m} = \text{least significant digit (LSD)} \]

• **Polynomial Notation** (Series Representation)

\[ N = a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \ldots + a_0 \times r^0 + a_{-1} \times r^{-1} \ldots + a_{-m} \times r^{-m} \]

\[ = \sum_{i=-m}^{n-1} a_i r^i \]

• \( N = (251.41)_{10} = 2 \times 10^2 + 5 \times 10^1 + 1 \times 10^0 + 4 \times 10^{-1} + 1 \times 10^{-2} \)
Binary Arithmetic

**Addition**

111011  \textit{Carries}

101011 \textit{Augend}

+ 11001 \textit{Addend}

1000100

---

**Subtraction**

0 1 10 0 10 \textit{Borrows}

1 0 0 1 0 1 \textit{Minuend}

- 1 1 0 1 1 \textit{Subtrahend}

1 0 1 0

---

\[\begin{array}{c|cc}
+ & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 10 \\
\end{array}\]
Binary Arithmetic (2)

**Multiplication**

\[
\begin{array}{c}
11010 \quad \text{Multiplicand} \\
x 1010 \quad \text{Multiplier} \\
00000 \\
11010 \\
00000 \\
11010 \\
\end{array}
\]

**Product**

\[100000100\]

**Division**

\[
\begin{array}{c}
1001 \quad \text{Dividend} \\
11001 \quad \text{Divider} \\
11000 \\
1001 \\
111 \\
\end{array}
\]

**Quotient**

\[110\]

**Remainder**

\[111\]

<table>
<thead>
<tr>
<th>X</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Base Conversion (1)

• **Series Substitution Method**
  
  – Expanded form of polynomial representation:
  \[ N = a_{n-1}r^{n-1} + \ldots + a_0r^0 + a_{-1}r^{-1} + \ldots + a_{-m}r^{-m} \] (1.3)
  
  – Conversation Procedure (base A to base B)
    
    • Represent the number in base A in the format of Eq. 1.3.
    • Evaluate the series using base B arithmetic.
  
  – **Example:**
    
    \[(11010)_2 \rightarrow (\ ? )_{10}\]
    \[
    N = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0
    \]
    \[
    = (16)_{10} + (8)_{10} + 0 + (2)_{10} + 0
    \]
    \[
    = (26)_{10}
    \]
Base Conversion (2)

- **Radix Divide Method**
  - Used to convert the integer in base $A$ to the equivalent base $B$ integer.
  - Underlying theory:
    - $(N_i)_A = b_{n-1}B^{n-1} + ... + b_0B^0$  
      Here, $b_i$'s represents the digits of $(N_i)_B$ in base $A$.
    - $N_i / B = (b_{n-1}B^{n-1} + ... + b_1B^1 + b_0B^0) / B$
      $= (\text{Quotient } Q_i: b_{n-1}B^{n-2} + ... + b_1B^0) + (\text{Remainder } R_0: b_0)$
    - In general, $(b_i)_A$ is the remainder $R_i$ when $Q_i$ is divided by $(B)_A$.

- **Conversion Procedure**
  1. Divide $(N_i)_B$ by $(B)_A$, producing $Q_1$ and $R_0$. $R_0$ is the least significant digit, $d_0$, of the result.
  2. Compute $d_i$, for $i = 1 ... n - 1$, by dividing $Q_i$ by $(B)_A$, producing $Q_{i+1}$ and $R_i$, which represents $d_i$.
  3. Stop when $Q_{i+1} = 0$. 

Base Conversion (3)

Radix divide example

\[(75)_{10} = (?)_{2} = (1001011)_{2}\]

<table>
<thead>
<tr>
<th>Step</th>
<th>Dividends</th>
<th>Divisor</th>
<th>Quotients</th>
<th>Remainders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75</td>
<td>2</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>2</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>2</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Base Conversion (4)

Radix Multiply Method

- Used to convert fractions.
- Underlying theory:
  - \((N_F)_A = b_{-1}B^{-1} + b_{-2}B^{-2} + ... + b_{-m}B^{-m}\)
    Here, \((N_F)_A\) is a fraction in base \(A\) and \(b_i\)'s are the digits of \((N_F)_B\) in base \(A\).
  - \(B \times N_F = B \times (b_{-1}B^{-1} + b_{-2}B^{-2} + ... + b_{-m}B^{-m})\)
    \(= (\text{Integer } I_{-1}: b_{-1}) + (\text{Fraction } F_{-2}: b_{-2}B^{-1} + ... + b_{-m}B^{-(m-1)})\)
  - In general, \((b_i)_A\) is the integer part \(l_{-i}\), of the product of \(F_{-(i+1)} \times (B_A)\).

- Conversion Procedure
  1. Let \(F_{-1} = (N_F)_A\).
  2. Compute digits \((b_{-i})_A\), for \(i = 1 \ldots m\), by multiplying \(F_i\) by \((B)_A\),
     producing integer \(l_{-i}\), which represents \((b_{-i})_A\), and fraction \(F_{-(i+1)}\).
  3. Convert each digits \((b_{-i})_A\) to base \(B\).
Base Conversion (5)

Radix Multiply Example

\((0.479)_{10} = (0.0111...)_{2}\)

\begin{align*}
\text{MSD} & \quad 0.9580 \leftarrow 0.479 \times 2 \\
& \quad 1.9160 \leftarrow 0.9580 \times 2 \\
& \quad 1.8320 \leftarrow 0.9160 \times 2 \\
\text{LSD} & \quad 1.6640 \leftarrow 0.8320 \times 2 \\
& \quad \ldots
\end{align*}
Base Conversion (6)

- **General Conversion Algorithm**
  - **Algorithm 1.1**
    To convert a number $N$ from base $A$ to base $B$, use
    (a) the series substitution method with base $B$ arithmetic, or
    (b) the radix divide or multiply method with base $A$ arithmetic.

  - **Algorithm 1.2**
    To convert a number $N$ from base $A$ to base $B$, use
    (a) the series substitution method with base 10 arithmetic to
    convert $N$ from base $A$ to base 10, and
    (b) the radix divide or multiply method with decimal arithmetic to
    convert $N$ from base 10 to base $B$.

- Algorithm 1.2 is longer, but easier and less error prone.
Base Conversion (7)

• When $B = A^k$

• **Algorithm 1.3**
  (a) To convert a number $N$ from base $A$ to base $B$ when $B = A^k$ and $k$ is a positive integer, group the digits of $N$ in groups of $k$ digits in both directions from the radix point and then replace each group with the equivalent digit in base $B$
  (b) To convert a number $N$ from base $B$ to base $A$ when $B = A^k$ and $k$ is a positive integer, replace each base $B$ digit in $N$ with the equivalent $k$ digits in base $A$.

• **Examples**
  – $(001 010 111.100)_2 = (127.4)_8$ (group bits by 3)
  – $(1011 0110 0101 1111)_2 = (B65F)_{16}$ (group bits by 4)
Test Your Understanding

• Convert \((10011010)_2\) to decimal.

• Convert \((10011010)_2\) to hexadecimal.

• Convert \((175)_{10}\) to binary.

• Convert \((C70A)_{16}\) to binary.
Test Your Understanding – Self-Check

• Convert \((10011010)_2\) to decimal.
  \((128 + 16 + 8 + 2)_{10} = (154)_{10}\)

• Convert \((10011010)_2\) to hexadecimal.
  \((9A)_{16}\)

• Convert \((175)_{10}\) to binary.
  \((10101111)_2\)

• Convert \((C7A0)_{16}\) to binary.
  \(1100011110100000\)
Binary Codes

- Fixed-point numbers
- (Floating-point numbers)
- Binary Coded Decimal (BCD) code
- ASCII code
- Gray code
- Error detection codes
- (Error correction codes)
Fixed-Point Numbers

• Typical formats for integers and fractions

![Magnitude representation with sign and implied binary point](image-a)

![Magnitude representation with sign and implied binary point](image-b)

• Signed-number representation
  • Sign-magnitude
  • 2’s complement
  • 1’s complement
Binary Coded Decimal (BCD)

- Represent decimal digits 0 - 9
- 4 bits are used
- Each bit position has a weight associated with it (weighted code)
- Weights are: 8, 4, 2, and 1 from MSB to LSB (called 8-4-2-1 code)
- BCD Codes
  - 0: 0000  1: 0001  2: 0010  3: 0011  4: 0100
  - 5: 0101  6: 0110  7: 0111  8: 1000  9: 1001
- Uses
  - Encode numbers from keypads
  - Encode numbers for output to numerical displays
  - Represent numbers in processors that perform decimal arithmetic
- **Example**: \((9750)_{10} = (1001011101010000)_{BCD}\)
American Standard Code for Information Interchange (ASCII)

- Most widely used character code.
- See Table 1.11 for 7-bit ASCII code.
- The eighth bit is often used for error detection (parity bit)
- **Example**: ASCII code representation of the word *Digital*

<table>
<thead>
<tr>
<th>Character</th>
<th>Binary Code</th>
<th>Hexadecimal Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1000100</td>
<td>44</td>
</tr>
<tr>
<td>i</td>
<td>1101001</td>
<td>69</td>
</tr>
<tr>
<td>g</td>
<td>1100111</td>
<td>67</td>
</tr>
<tr>
<td>i</td>
<td>1101001</td>
<td>69</td>
</tr>
<tr>
<td>t</td>
<td>1110100</td>
<td>74</td>
</tr>
<tr>
<td>a</td>
<td>1100001</td>
<td>61</td>
</tr>
<tr>
<td>l</td>
<td>1101100</td>
<td>6C</td>
</tr>
</tbody>
</table>
Gray Code

- **Cyclic code**: A circular shifting of a code word produces another code word, e.g., \(101-110-011-101\).
- **Gray code**: A cyclic code with the property that two consecutive code words differ in only 1 bit (the distance between the two code words is 1). A 4-bit Gray code
  
  \[
  0000-0001-0011-0010-0110-0111-0101-0100-
  1100-1101-1111-1110-1010-1011-1001-1000-0000
  \]
Weight, Distance, and Errors

- Weight ($w$) – the number of 1’s in a code word, e.g.,
  $w(101011) = 4$, $w(000000) = 0$.

- Distance ($d$) – the number of bit positions in which two code words differ, e.g.,
  
  $$d(101011, 000000) = 4$$
  $$d(100110, 110010) = 2$$

- Error – an incorrect value in one or more bits
  
  $101011 \rightarrow 101001$ (single error)
  $101011 \rightarrow 100001$ (double error)
Error Detection Codes

• Parity codes
  – Odd-parity \(w(P|I)\) is odd
  – Even-parity \(w(P|I)\) is even
  – Single error detection \((d_{\text{min}} = 2)\)
  – Examples
    \[I = 1000011\]
    \[\text{01000011 (odd)}\]
    \[\text{11000011 (even)}\]
    \[I = 0101101\]
    \[\text{10101101 (odd)}\]
    \[\text{00101101 (even)}\]

• Two-out-of-Five Code (see p. 68 of text)
Test Your Understanding

• What do these strings of 1’s and 0’s represent?
  100001110100111000101
  0010010001000001

• What are the weights of the following two words?
  10110010
  11110100

• What’s the distance between the two words?
Test Your Understanding – Self-Check

• What do these strings of 1’s and 0’s represent?
  
  100001110100111000101
  C          S          E
  0010010001000001
  2        4        4        1

• What are the weights of the following two words?
  \[ w(10110010) = 4 \]
  \[ w(11110100) = 5 \]

• What’s the distance between the two words? \( d = 3 \)