Lecture 2
Review on Digital Logic (Part 1)

Xuan ‘Silvia’ Zhang
Washington University in St. Louis

http://classes.engineering.wustl.edu/ese461/
Grading

- Engagement  5%
- Review Quiz  10%
- Homework    10%
- Labs        40%
- Final Project 35%

Policy:
- 90% or above    A
- 80% - 89%       B
- 65% - 79%       C
- 45% - 64%       D
- 44% or below    F
Number Systems

- **Decimal** (radix $r=10$)
  - digits 0-9

- **Binary** (radix $r=2$)
  - $(1011.01)_2 = (\ . \ )_{10}$

- **Octal**
  - radix = 8

- **Hexadecimal**
  - each HEX digit can represent 4 bits
  - $(10011110.0101)_2 = (\ . \ )_{16}$
  - $(11110010.0001)_2 = (\ . \ )_{16}$
  - $(11010100.1111)_2 = (\ . \ )_{16}$
Base Conversion

- Convert the integer part
- Convert the fraction part
- Join the two results with a radix point

Example: \((325.64)_{10}\) to \((.\quad)_{5}\)

- \(2 \times 5^3 + 3 \times 5^2 = 325\)
- \(3 \times 5^{-1} + 1 \times 5^{-2} = 0.64\)

\[
\sum A_i r_1^i = \sum B_i r_2^i
\]
Range of Numbers

• Integer (n-bit Number)
  - $2^n$ different numbers
  - Min: 0
  - Max: $2^n-1$

• Fraction (m-bit Number)
  - Min: 0
  - Max: $(2^m-1)/2^m$
Complements

- **Diminished Radix Complement of N**
  - defined as \((r^n-1)-N\), with \(n =\) number of digits or bits
  - 1’s complement for binary (radix = 2)

- **Radix Complement**
  - defined as \(r^n-N\)
  - 2’s complement for binary

- **Why**
  - subtraction as addition of complement
  - if negative?
Binary Complement

• 1’s complement
  - complement each individual bit (bitwise NOT)

• 2’s complement
  - 1’s complement plus 1
  - alternative
  - start from the least significant bit (LSB)
  - copy all least significant 0’s
  - copy the first 1
  - complement all bits thereafter
Subtraction with 2’s Complement

- For n-digit unsigned numbers $M$ and $N$
- $M - N = ?$
  - add 2’s complement of $N$ to $M$
  - $M + (2^n - N) = M - N + 2^n$

- Example
  - carry 1
  - carry 0
Signed Binary Numbers

- To represent a sign (+ or -)
  - need one more bit
  - sign + magnitude
  - signed-complements

- Positive numbers unchanged
- Negative numbers use one of the two methods

<table>
<thead>
<tr>
<th>#</th>
<th>sign+</th>
<th>1’s</th>
<th>2’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>010</td>
<td>010</td>
<td>010</td>
</tr>
<tr>
<td>-2</td>
<td>110</td>
<td>101</td>
<td>110</td>
</tr>
<tr>
<td>+3</td>
<td>011</td>
<td>011</td>
<td>011</td>
</tr>
<tr>
<td>-3</td>
<td>111</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>+0</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>-0</td>
<td>100</td>
<td>111</td>
<td>000</td>
</tr>
</tbody>
</table>
2’s Complement Arithmetic

- **Addition**
  - represent negative number by its 2’s complement
  - add the number including the sign bits
  - discard a carry out of the sign bits
  - e.g. \( M + (-N) \rightarrow M + (2^n-N) \)

- **Subtraction**
  - \( M - N \rightarrow M + (2^n-N) \)
  - form the complement of the subtrahend
  - follow the rules for addition
Overflow

• Error occurs when out of range
  - \([-2^{n-1}, 2^{n-1}-1]\]
  - carry-in into the sign bit different from the carry-out

-39 + 92 = 53:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
+ & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Carryout without overflow. Sum is correct.

-19 + -7 = -26:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 \\
+ & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 \\
\end{array}
\]

Carryout without overflow. Sum is correct.

44 + 45 = 89:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
+ & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

No overflow nor carryout.

104 + 45 = 149:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
+ & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Overflow, no carryout. Sum is not correct.

-75 + 59 = -16:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
+ & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

No overflow nor carryout.

-103 + -69 = -172:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Overflow, with incidental carryout. Sum is not correct.

10 + -3 = 7:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
+ & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

Carryout without overflow. Sum is correct.

127 + 1 = 128:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Overflow, no carryout. Sum is not correct.

-1 + 1 = 0:

\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Carryout without overflow. Sum is correct.
Outline

Number Representation

Boolean Logic and Gates

Combinational Logic
Binary Logic and Gates

- **Basic logic operations**
  - AND: $X \cdot Y$
  - OR: $X + Y$
  - NOT: $\overline{X}$ or $/X$

- **Truth table**
  - all possible input combinations

- **CMOS as a switch**
**Boolean Algebra**

- **Basic identities**
  1. \( X + 0 = X \)
  2. \( X \cdot 1 = X \)
  3. \( X + 1 = 1 \)
  4. \( X \cdot 0 = 0 \)
  5. \( X + X = X \)
  6. \( X \cdot X = X \)
  7. \( X + \overline{X} = 1 \)
  8. \( X \cdot \overline{X} = 0 \)
  9. \( \overline{X} = X \) Involution
  10. \( X + Y = Y + X \) Commutative
  11. \( XY = YX \) Associative
  12. \( (X + Y) + Z = X + (Y + Z) \)
  13. \( (XY) Z = X(Y Z) \)
  14. \( X (Y + Z) = XY + XZ \) Distributive
  15. \( X + YZ = (X + Y) (X + Z) \) DeMorgan’s
  16. \( \overline{X + Y} = \overline{X} \cdot \overline{Y} \)

- **Multiple Variables**

**Dual**

Replace “+” by “.”, “.” by +, “0” by “1” and “1” by “0”
Boolean Algebra

• Other useful theorems

\[ XY + \overline{XY} = Y \quad \text{Minimization} \]
\[ X + XY = X \quad \text{Absorption} \]
\[ X + \overline{XY} = X + Y \quad \text{Simplification} \]
\[ XY + \overline{XZ} + YZ = XY + X\overline{Z} \quad \text{Consensus} \]
\[ (X + Y)(\overline{X} + Z)(Y + Z) = (X + Y)(\overline{X} + Z) \quad \text{Dual} \]
Standard (Canonical) Forms

- For comparison of equality
- Correspondence to the truth table
- Sum of Products (SOP)
  - sum of minterms
- Product of Sum (POS)
  - product of maxterms

<table>
<thead>
<tr>
<th>Index</th>
<th>Minterm</th>
<th>Maxterm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (00)</td>
<td>/x/ y</td>
<td>x + y</td>
</tr>
<tr>
<td>1 (01)</td>
<td>/x y</td>
<td>x + /y</td>
</tr>
<tr>
<td>2 (10)</td>
<td>x /y</td>
<td>/x + y</td>
</tr>
<tr>
<td>3 (11)</td>
<td>x y</td>
<td>/x + /y</td>
</tr>
</tbody>
</table>

- Relationship between min and MAX term

\[ M_i = \overline{m_i} \quad m_i = \overline{M_i} \]
Circuit Optimization

- Simplest implementation
- Cost criterion
  - literal cost (L)
  - gate input cost (G)
  - gate input cost with NOTs (GN)
- Examples (all the same function):
  - \( F = BD + AB'C + AC'D' \)
    \( L = \)
  - \( F = BD + AB'C + AB'D' + ABC' \)
    \( L = \)
  - \( F = (A + B)(A + D)(B + C + D')( B' + C' + D) \)
    \( L = \)
  - Which solution is best?
Karnaugh Maps (K-map)

- Simplify Boolean algebra
- Transfer truth table to 2D grid
  - ordered in Gray code
- Human’s pattern recognition capability
  - don’t cares
  - race hazards
Other Gate Types

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>BUF</th>
<th>NAND</th>
<th>NOR</th>
<th>XOR</th>
<th>XNOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Tri-State Buffer

Truth Table

<table>
<thead>
<tr>
<th>EN</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>Hi-Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
CMOS NAND and NOR Gates

(a)

VDD

on

on

F = 1

B = 0
A = 0

off

off

VDD

off

on

F = 1

B = 1
A = 0

on

off

VDD

on

off

F = 1

B = 0
A = 1

off

on

VDD

off

on

F = 0

B = 1
A = 1

on

on

F = NAND(A, B)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

p-channel

n-channel
NAND Mapping Algorithm

- Replace ANDs and ORs

- Repeat until there is at most one inverter between
  - a circuit input or driving NAND gate output
  - the attached NAND gate inputs
NOR Mapping Algorithm

- Replace ANDs and ORs

- Repeat until there is at most one inverter between
  - a circuit input or driving NOR gate output
  - the attached NOR gate inputs
Construct AOI and OAI Gates

- AND-OR-INVERT (AOI)
- OR-AND-INVERT (OAI)
Outline

Number Representation

Boolean Logic and Gates

Combinational Logic
Decoding

- n-bit to represent up to $m=2^n$ elements
- convert n-bit input to m-bit output code
  - $n \leq m \leq 2^n$

- Outputs correspond to minterms: $D_i = m_i$
- Divide into smaller decoders
Arbitrary Combinational Logic

- Decoder and OR gates
  - implement m functions of n variables
  - SOP expressions
  - one n-to-2^n line decoder
  - m OR gates, one for each output

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Indicate MSB, LSB

\[ F = \sum m(3,5,6,7) \]
Encoding

- Convert one-hot code to its position
  - assume exactly one bit is 1
  - need to know its position

- Priority
  - more than one input value is 1
Multiplexing

- Select data
  - a set of \( n \) information inputs to select from
  - a set of \( k \) control (select) lines to make selection
  - a single output

\[ n \leq 2^k \text{ inputs} \]
MUX Realization

- SOP Expression
  \[ \text{OUT} = \sum_{i=0}^{2^{k}-1} m_i l_i \]

- Transmission gates
Questions?

Comments?

Discussion?
Homework #1

- Download problem sets from class website
- Due 09/07 (Wednesday) in class
- No grace period