FPGA Design
EECE 277

Number Representation and Adders

Dr. William H. Robinson
February 21, 2005

http://eecs.vanderbilt.edu/courses/eece277/

Topics

“There are 10 kinds of people in the world, those that
understand binary and those that don’t.”

– Unknown

• Administrative stuff
  – Laboratory Assignment #2 (due Monday, February 28)
  – PLEASE do the tutorials in Appendix B, C, and D of the textbook

• Number systems

• Adder circuits

Laboratory Assignment #2

• Quartus II Software
  – Refer to Appendices A, B, C, and D

• UP2 Design Laboratory Kits
  – Cables
  – Problem #5

• Laboratory Report
  – Follow guidelines from handout

• In-Class Lab Session
  – Wednesday, February 23, 9am – 11am

Class Exercise

• 1101010001102 = ____________16
• 11011110112 = ____________8
• 110110012 = ____________10
• DEF16 = ____________8
• 111110 = ____________2

D46
15738
21710
67578
100010101112
Positional Number Representation

- Decimal
  \[
  \begin{array}{cccccc}
  \quad & 10^4 & 10^3 & 10^2 & 10^1 & 10^0 \\
  \hline
  \end{array}
  \]

- Hexadecimal
  \[
  \begin{array}{cccccc}
  \quad & 16^4 & 16^3 & 16^2 & 16^1 & 16^0 \\
  \hline
  \end{array}
  \]

- Octal
  \[
  \begin{array}{cccccc}
  \quad & 8^4 & 8^3 & 8^2 & 8^1 & 8^0 \\
  \hline
  \end{array}
  \]

- Binary
  \[
  \begin{array}{cccccc}
  \quad & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
  \hline
  \end{array}
  \]

Numbers in Different Systems

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>03</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>05</td>
<td>05</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>07</td>
<td>07</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>10</td>
<td>08</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>11</td>
<td>09</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>12</td>
<td>0A</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>13</td>
<td>0B</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>0C</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>15</td>
<td>0D</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>16</td>
<td>0E</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>17</td>
<td>0F</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>101</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>18</td>
<td>102</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>103</td>
<td>23</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>104</td>
<td>24</td>
<td>14</td>
</tr>
</tbody>
</table>

Conversion from Decimal to Binary

Convert \(857_{10}\)

<table>
<thead>
<tr>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>857 ÷ 2  = 428</td>
</tr>
<tr>
<td>428 ÷ 2  = 214</td>
</tr>
<tr>
<td>214 ÷ 2  = 107</td>
</tr>
<tr>
<td>107 ÷ 2  = 53</td>
</tr>
<tr>
<td>53 ÷ 2   = 26</td>
</tr>
<tr>
<td>26 ÷ 2   = 13</td>
</tr>
<tr>
<td>13 ÷ 2   = 6</td>
</tr>
<tr>
<td>6 ÷ 2    = 3</td>
</tr>
<tr>
<td>3 ÷ 2    = 1</td>
</tr>
<tr>
<td>1 ÷ 2    = 0</td>
</tr>
</tbody>
</table>

Result is \(1010101011\).

Units of Memory/Storage

- Binary digit = bit (little “b”)
- Byte = 8 bits (capital “B”)
- KB = 2^10 bytes
- MB = 2^20 bytes
- GB = 2^30 bytes
- TB = 2^40 bytes
**Half-Adder**

- **Sum**
  - 0
  - 1
  - 1
  - 0
- **Carry**
  - 0
  - 0
  - 1
  - 1

(a) The four possible cases

(b) Truth table

(c) Circuit

(d) Graphical symbol

---

**Decomposed Full-Adder**

- **C0**
- **s**
- **c**
- **s_{i+1}
- **c_{i+1}

(a) Block diagram

(b) Detailed diagram

(c) Symbol

---

**VHDL Code for Full Adder**

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fulladd IS
  PORT ( Cin, a, b : IN STD_LOGIC;
         s, Cout : OUT STD_LOGIC );
END fulladd;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
  s <= a XOR b XOR Cin ;
  Cout <= (a AND b) OR (Cin AND a) OR (Cin AND b) ;
END LogicFunc;
```

- VHDL can be used to describe the structure as well

---

**Ripple-Carry Adder**

- Starts from the LSB to the MSB
- Delay equals \( n \Delta t \) where \( \Delta t \) is the delay through one full-adder circuit
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY adder4 IS
PORT ( Cin : IN STD_LOGIC;
a3, a2, a1, a0 : IN STD_LOGIC;
b3, b2, b1, b0 : IN STD_LOGIC;
Cout : OUT STD_LOGIC);
END adder4;

ARCHITECTURE Structure OF adder4 IS
BEGIN
stage0: fulladd PORT MAP ( Cin, a0, b0, s0, c1 ) ;
stage1: fulladd PORT MAP ( c1, a1, b1, s1, c2 ) ;
stage2: fulladd PORT MAP ( c2, a2, b2, s2, c3 ) ;
stage3: fulladd PORT MAP ( Cin => c3, Cout => Cout, a => a3, b => b3, s => s3 ) ;
END Structure ;

Negative Binary Numbers

• Signed magnitude
  – Leftmost bit is sign, remaining bits are magnitude
  – Two representations for zero

• One’s complement (obsolete)
  – Invert all bits
  – Two representations for zero

• Two’s complement
  – Invert all bits and add “1”
  – Only one representation for zero

Integer Representation

![Integer Representation Diagram]

Interpretation of 4-Bit Signed Integers

<table>
<thead>
<tr>
<th>Signed magnitude</th>
<th>1’s complement</th>
<th>2’s complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>+7</td>
<td>+7</td>
<td>+7</td>
</tr>
<tr>
<td>+6</td>
<td>+6</td>
<td>+7</td>
</tr>
<tr>
<td>+5</td>
<td>+5</td>
<td>+6</td>
</tr>
<tr>
<td>+4</td>
<td>+4</td>
<td>+5</td>
</tr>
<tr>
<td>+3</td>
<td>+3</td>
<td>+4</td>
</tr>
<tr>
<td>+2</td>
<td>+2</td>
<td>+3</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>+2</td>
</tr>
<tr>
<td>-0</td>
<td>-0</td>
<td>-0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-0</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>-3</td>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>-4</td>
<td>-4</td>
<td>-3</td>
</tr>
<tr>
<td>-5</td>
<td>-5</td>
<td>-4</td>
</tr>
<tr>
<td>-6</td>
<td>-6</td>
<td>-5</td>
</tr>
<tr>
<td>-7</td>
<td>-7</td>
<td>-6</td>
</tr>
</tbody>
</table>
Finite-Precision Numbers

- Computers have fixed number of digits to represent values (creates set of valid values)
- Computation can lead to “errors”
  - Overflow: result is larger than largest number in set
  - Underflow: result is smaller than smallest number in set
- Computation could result in a number that cannot be represented (non-integer result on integer machine)

2's Complement Number Wheel

```
1111 1 1 1 1
1110 0 1 1 1
1101 0 1 0 0
1000 0 0 0 0
1001 0 0 0 1
1010 0 0 1 0
1011 0 0 1 1
1100 0 1 0 0
1101 0 1 0 1
1110 0 1 1 0
```

Determining Overflow Flag

\[
\begin{array}{c|c|c|c}
& (+ 7) & (-7) & 0 0 0 0 \\
\hline
(+ 2) & 0 1 1 0 & + 0 0 1 0 & (+ 2) + 0 0 1 0 \\
(+ 9) & 1 0 0 1 & (-5) & 1 0 1 1 \\
\hline
& c_4 = 0 & c_4 = 1 & c_5 = 0 \\
& c_3 = 1 & c_3 = 1 & c_5 = 0 \\
\end{array}
\]

```
c_4 = 0 
c_3 = 1 
c_5 = 0
```

Adder/Subtractor Megafuction

- Carry into MSB must equal carry out of MSB (XOR gate)
Adder/Subtractor Circuit

Representation of Numbers in VHDL

- Use a bus declaration
  - SIGNAL C: std_logic_vector(2 downto 0) describes a 3-bit bus where C(2) is most significant bit and C(0) is least significant bit.
  - SIGNAL C: std_logic_vector(0 to 2) is also a 3-bit bus, but C(0) is MSB, C(2) is LSB.
  - ALWAYS use 'downto' in this class.

- A signal assignment can be done in many ways:
  - C <= "110"; assigns all three bits.
  - C(2) <= '1'; assigns only bit #2.
  - C(1 downto 0) <= "10"; assigns two bits of the bus.

4-Bit Adder with Multi-Bit Signals

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.fulladd_package.all;

ENTITY adder4 IS
  PORT ( Cin : IN STD_LOGIC;
          X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          S : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          Cout : OUT STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
  SIGNAL C : STD_LOGIC_VECTOR(3 DOWNTO 1) ;
  BEGIN
    stage0: fulladd PORT MAP ( Cin, X(0), Y(0), S(0), C(1) ) ;
    stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
    stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
    stage3: fulladd PORT MAP ( C(3), X(3), Y(3), S(3), Cout ) ;
    END Structure ;
END Structure ;

VHDL Code for 16-Bit Adder

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;

ENTITY adder16 IS
  PORT ( X, Y : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
          S : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) ) ;
END adder16 ;

ARCHITECTURE Behavior OF adder16 IS
  BEGIN
    S <= X + Y ;
    END Behavior ;

- Uses the "signed" package to add signals.
Using the Arithmetic Package

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY adder16 IS
  PORT ( Cin : IN STD_LOGIC;
         X, Y : IN SIGNED(15 DOWNTO 0);
         S : OUT SIGNED(15 DOWNTO 0);
         Cout, Overflow : OUT STD_LOGIC);
END adder16;

ARCHITECTURE Behavior OF adder16 IS
  SIGNAL Sum : SIGNED(16 DOWNTO 0);
BEGIN
  Sum <= ('0' & X) + Y + Cin;
  S <= Sum(15 DOWNTO 0);
  Cout <= Sum(16);
  Overflow <= Sum(16) XOR X(15) XOR Y(15) XOR Sum(15);
END Behavior;

Using Integer Signals

ENTITY adder16 IS
  PORT ( X, Y : IN INTEGER RANGE -32768 TO 32767;
         S : OUT INTEGER RANGE -32768 TO 32767);
END adder16;

ARCHITECTURE Behavior OF adder16 IS
BEGIN
  S <= X + Y;
END Behavior;

Summary

• Calculators do an excellent job of converting values among decimal, binary, octal, and hexadecimal systems.

• 2’s complement is used because it only has one representation for zero.

• The carry chain is the critical path for an adder circuit.