COL215 Digital Logic and System Design

Number Representation and Arithmetic Operations

05 September 2017

Values and Representations

- Values : Elements of a domain

  integers

  real numbers

  letters of an alphabet

  characters

  logic values

- Representations : Arrangements of symbols

  bits {0,1} can be used to represent values

  from a variety of domains

Positional representation of numbers

- Consider only non-negative integers at present (unsigned)
- Radix or base = r

Representation : $d_{n-1}\ d_{n-2}\ \ldots\ d_2\ d_1\ d_0$

$0 \le \text{val}(d_i) \le r-1$

Example:

Representation	“1110” , radix 2
Value	14

From representation to value

Representation : $d_{n-1}\ d_{n-2}\ \ldots\ d_2\ d_1\ d_0$

$0 \le \text{val}\,(d_i) \le r-1$

Value : $\sum_{i=0}^{n-1} \text{val}\,(d_i) \,.\, r^i$

or simply $\sum_{i=0}^{n-1} d_i \,.\, r^i$ [if it is clear that d_i denotes value of d_i]

Another way: from representation to value

$((\ldots(d_{n-1} \times r + d_{n-2}) \times r + \ldots + d_1) \times r) + d_0$

From Value to Representation

Given value = V, radix = s

- val (d_0) = V mod s
- val (d_1) = (V div s) mod s
- val (d_2) = ((V div s) div s) mod s
- . . .
- val (d_i) = $(\ldots (V \underbrace{\text{div } s) \ldots}_{i \text{ times}}) \text{ mod } s$
- . . .

Conversion from radix r to s

Do computation in radix r (division method)

- start with value in radix r (same as representation in r)
- compute the digit values with radix = s (from value to representation)
- put together these digits (representations, not values) to form the representation in radix s

Conversion from radix r to s

Do computation in radix s (multiplication method)

- start with representation in radix r
- this gives the digits
- from digits compute the value (from representation to value)
- the value expressed in radix s is the representation in radix s

Example to illustrate the methods

- Conversion from decimal (radix 10) to binary (radix 2) and vice versa
- Use subscript D for decimal and B for binary
- 213_D => $????????_B$
- 11010101_B => $???_D$

Decimal to Binary, division method

- 213_D mod 2_D = 1_D
- 213_D div 2_D = 106_D
- 106 mod 2 = 0
- 106 div 2 = 53
- 53 mod 2 = 1
- 53 div 2 = 26
- 26 mod 2 = 0
- 26 div 2 = 13
- 13 mod 2 = 1
- 13 div 2 = 6
- 6 mod 2 = 0
- 6 div 2 = 3
- 3 mod 2 = 1
- 3 div 2 = 1
- 1 mod 2 = 1
- 1 div 2 = 0

Result = $1\ 1\ 0\ 1\ 0\ 1\ 0\ 1_B$

Decimal to Binary, multiplication method

$2\ 1\ 3_D$ => digit values are 10_B, 1_B, 11_B

radix value is 1010_B

Do this in binary: $(2_D \times 10_D + 1_D) \times 10_D + 3_D$

$(10_B \times 1010_B + 1_B) \times 1010_B + 11_B$

$= (10100 + 1) \times 1010 + 11$

$= 10101 \times 1010 + 11$

$= 10101000 + 101010 + 11$

$= 11010101$

Binary to Decimal, division method

- 11010101_B mod $1010_B = 11_B$
- 11010101_B div $1010_B = 10101_B$
- 10101 mod 1010 = 1
- 10101 div 1010 = 10
- 10 mod 1010 = 10
- 10 div 1010 = 0

$$\text{Result} = 10_B \ 1_B \ 11_B = 2_D \ 1_D \ 3_D$$
$$= 213_D$$

Binary to Decimal, multiplication method

11010101_B => digit values are $1_D,1_D,0_D,1_D,0_D,1_D,0_D,1_D$

radix value is 2_D

Compute (in decimal):

$((((((1x2+ 1)x2+ 0)x2+ 1)x2+ 0)x2+ 1)x2+ 0)x2+ 1$

$= (((((3 x 2 + 0)x 2 + 1)x 2 + 0)x 2 + 1)x 2+0)x 2 + 1$

$= ((((6 x 2 + 1) x 2 + 0) x 2 + 1) x 2 + 0) x 2 + 1$

$= (((13 x 2 + 0) x 2 + 1) x 2 + 0) x 2 + 1$

$= ((26 x 2 + 1) x 2 + 0) x 2 + 1$

Binary to Decimal, multiplication method - continued

= ((26 x 2 + 1) x 2 + 0) x 2 + 1

= (53 x 2 + 0) x 2 + 1

= 106 x 2 + 1

= 213_D

Binary numbers : 4 bits to 64 bits

4 bits :	0 .. 15
nibble	
8 bits :	0 .. 255
byte	
16 bits :	0 .. 65,535
half word	
32 bits :	0 .. 4,294,967,295
word	
64 bits :	0 .. 18,446,744,073,709,551,615
double word	

4 bit binary numbers

Binary	Decimal	Binary	Decimal
0 0 0 0	0 0	1 0 0 0	0 8
0 0 0 1	0 1	1 0 0 1	0 9
0 0 1 0	0 2	1 0 1 0	1 0
0 0 1 1	0 3	1 0 1 1	1 1
0 1 0 0	0 4	1 1 0 0	1 2
0 1 0 1	0 5	1 1 0 1	1 3
0 1 1 0	0 6	1 1 1 0	1 4
0 1 1 1	0 7	1 1 1 1	1 5

Radix 8 (octal) numbers

Binary	Decimal/Octal	Binary	Decimal/Octal
0 0 0 0	0 0 / 0 0	1 0 0 0	0 8 / 1 0
0 0 0 1	0 1 / 0 1	1 0 0 1	0 9 / 1 1
0 0 1 0	0 2 / 0 2	1 0 1 0	1 0 / 1 2
0 0 1 1	0 3 / 0 3	1 0 1 1	1 1 / 1 3
0 1 0 0	0 4 / 0 4	1 1 0 0	1 2 / 1 4
0 1 0 1	0 5 / 0 5	1 1 0 1	1 3 / 1 5
0 1 1 0	0 6 / 0 6	1 1 1 0	1 4 / 1 6
0 1 1 1	0 7 / 0 7	1 1 1 1	1 5 / 1 7

Radix 16 (hexadecimal) numbers

Binary	Decimal/Hex	Binary	Decimal/Hex
0 0 0 0	0 0 / 0	1 0 0 0	0 8 / 8
0 0 0 1	0 1 / 1	1 0 0 1	0 9 / 9
0 0 1 0	0 2 / 2	1 0 1 0	1 0 / A
0 0 1 1	0 3 / 3	1 0 1 1	1 1 / B
0 1 0 0	0 4 / 4	1 1 0 0	1 2 / C
0 1 0 1	0 5 / 5	1 1 0 1	1 3 / D
0 1 1 0	0 6 / 6	1 1 1 0	1 4 / E
0 1 1 1	0 7 / 7	1 1 1 1	1 5 / F

BCD (Binary Coded Decimal) numbers

Binary	Decimal/BCD	Binary	Decimal/BCD
0 0 0 0	0 0 / 0000 0000	1 0 0 0	0 8 / 0000 1000
0 0 0 1	0 1 / 0000 0001	1 0 0 1	0 9 / 0000 1001
0 0 1 0	0 2 / 0000 0010	1 0 1 0	1 0 / 0001 0000
0 0 1 1	0 3 / 0000 0011	1 0 1 1	1 1 / 0001 0001
0 1 0 0	0 4 / 0000 0100	1 1 0 0	1 2 / 0001 0010
0 1 0 1	0 5 / 0000 0101	1 1 0 1	1 3 / 0001 0011
0 1 1 0	0 6 / 0000 0110	1 1 1 0	1 4 / 0001 0100
0 1 1 1	0 7 / 0000 0111	1 1 1 1	1 5 / 0001 0101

Radix 2, 8, 10, 16

$$\begin{tikzcd}
& 1\;1\;0\;1\;0\;1\;0\;1_2 \arrow[dl] \arrow[d] \arrow[dr] & \\
\begin{array}{c} 1\;1,0\;1\;0,1\;0\;1 \\ 3 \quad\;\; 2 \quad\;\; 5_8 \end{array} & 2 \quad 1 \quad 3_{10} & \begin{array}{c} 1\;1\;0\;1,0\;1\;0\;1 \\ D \qquad\quad 5_{16} \end{array}
\end{tikzcd}$$

Binary Addition

Just like in primary school

```
  0 0 1 1 [3]       0 1 0 1  [5]        0 1 1 0  [6]
+ 0 1 1 0 [6]     + 0 1 1 1  [7]      + 1 1 0 1 [13]
--------------------------------------------------------
  1 0 0 1 [9]       1 1 0 0 [12]        0 0 1 1  [3]
```

Overflow will be discussed later.

Binary Subtraction

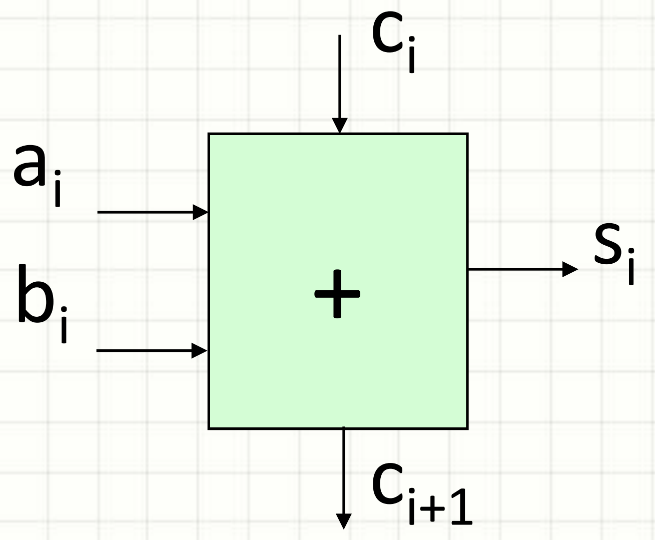
Just like in primary school

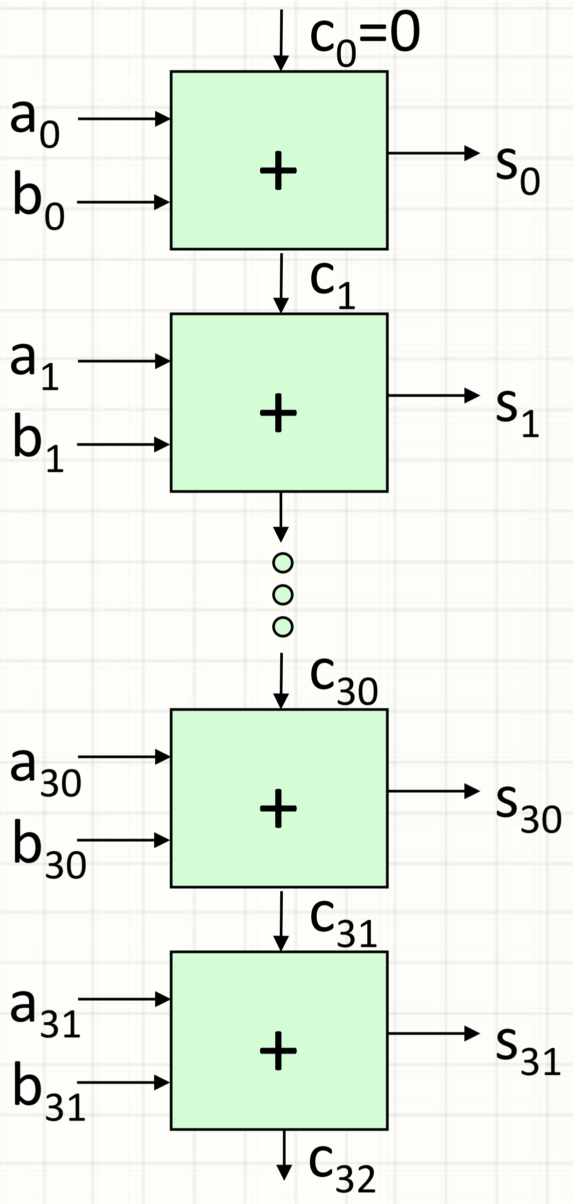
```
   1 1 0 1 [13]      0 1 1 1  [7]          0 0 1 1  [3]
 -  0 1 1 0   [6]   -  0 1 0 1  [5]        -  0 1 1 0  [6]
______________________________________________________
    0 1 1 1  [7]      0 0 1 0   [2]         1 1 0 1 [13]
```

Overflow will be discussed later.

Adder circuit

- Perform bit by bit addition - similar to the "paper and pencil" way





One bit adder truth table

a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Boolean expressions for adder

$s_i = a_i' \, b_i' \, c_i + a_i' \, b_i \, c_i' + a_i \, b_i' \, c_i' + a_i \, b_i \, c_i$

alternatively,

$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i \, b_i + a_i \, c_i + b_i \, c_i$

One bit subtractor truth table

a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Boolean expressions for subtractor

$s_i = a_i' \, b_i' \, c_i + a_i' \, b_i \, c_i' + a_i \, b_i' \, c_i' + a_i \, b_i \, c_i$

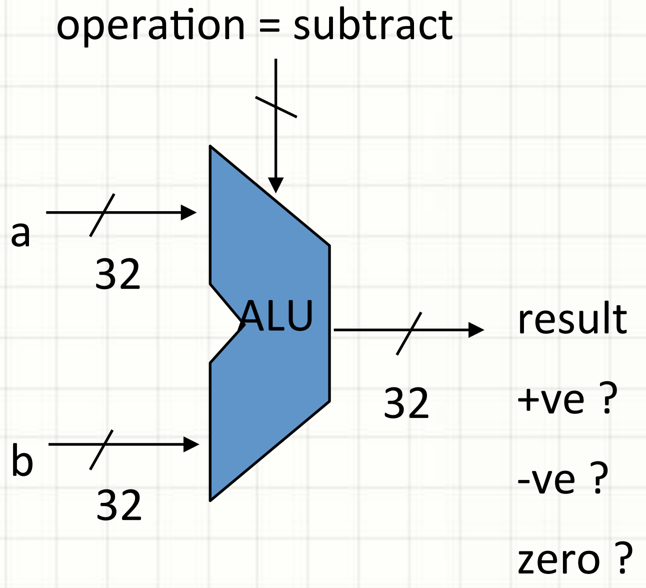
alternatively,

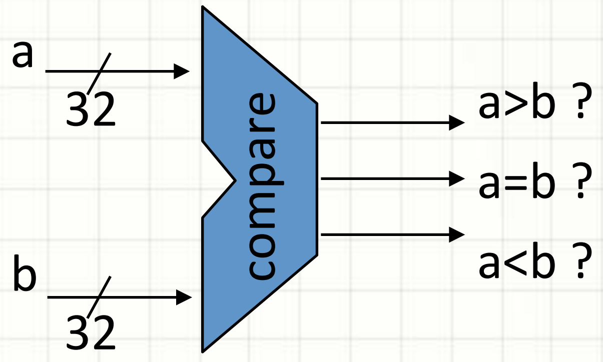
$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i' \, b_i + a_i' \, c_i + b_i \, c_i$

Comparing two integers

- Subtract and check the result
- **Compare directly**





Compare (>) directly (unsigned)

Method 1:

$$a_{0..i} > b_{0..i} \equiv (a_{0..i-1} > b_{0..i-1})(a_i + \overline{b_i}) + a_i \overline{b_i}$$

Method 2:

$$a_{i..31} > b_{i..31} \equiv (a_{i+1..31} > b_{i+1..31}) + (a_{i+1..31} = b_{i+1..31}) a_i \overline{b_i}$$

Compare (=) directly (unsigned)

Method 1:

$$a_{0..i} = b_{0..i} \equiv (a_{0..i-1} = b_{0..i-1})(\overline{a_i \oplus b_i})$$

Method 2:

$$a_{i..31} = b_{i..31} \equiv (a_{i+1..31} = b_{i+1..31})(\overline{a_i \oplus b_i})$$

Method 3:

$$\mathrm{a} = \mathrm{b} \equiv \prod_{\mathrm{i}=0}^{31} (\overline{\mathrm{a_i} \oplus \mathrm{b_i}})$$

Comparator circuit – method 1

$$a_{0..i} > b_{0..i} \equiv (a_{0..i-1} > b_{0..i-1})(a_i + \overline{b_i}) + a_i\overline{b_i}$$

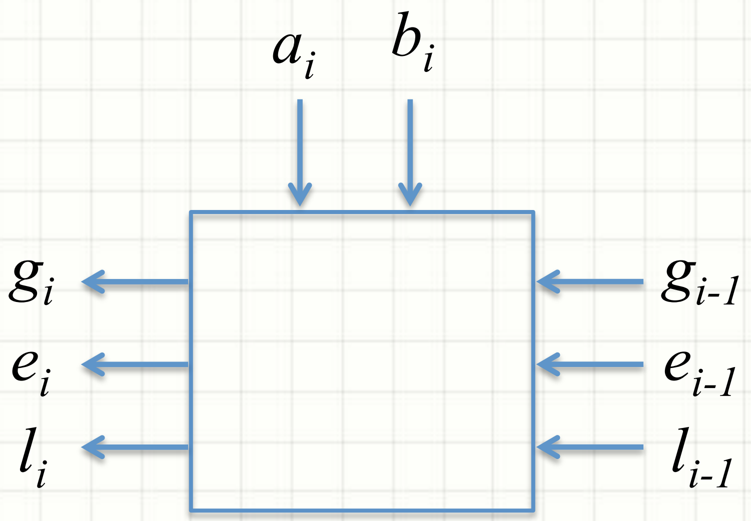
$$a_{0..i} = b_{0..i} \equiv (a_{0..i-1} = b_{0..i-1})(\overline{a_i \oplus b_i})$$

$$a_{0..i} < b_{0..i} \equiv (a_{0..i-1} < b_{0..i-1})(\overline{a_i} + b_i) + \overline{a_i}b_i$$

$$g_i \equiv g_{i-1}(a_i + \overline{b_i}) + a_i\overline{b_i}$$

$$e_i \equiv e_{i-1}(\overline{a_i \oplus b_i})$$

$$l_i \equiv l_{i-1}(\overline{a_i} + b_i) + \overline{a_i}b_i$$



$g_{-1}=0$

$e_{-1}=1$

$l_{-1}=0$

Comparator circuit – method 2

$$a_{i..31} > b_{i..31} \equiv (a_{i+1..31} > b_{i+1..31}) + (a_{i+1..31} = b_{i+1..31})\, a_i \overline{b_i}$$

$$a_{i..31} = b_{i..31} \equiv (a_{i+1..31} = b_{i+1..31})(\overline{a_i \oplus b_i})$$

$$a_{i..31} < b_{i..31} \equiv (a_{i+1..31} < b_{i+1..31}) + (a_{i+1..31} = b_{i+1..31})\, \overline{a_i} b_i$$

$$g_i \equiv g_{i+1} + e_{i+1} a_i \overline{b_i}$$

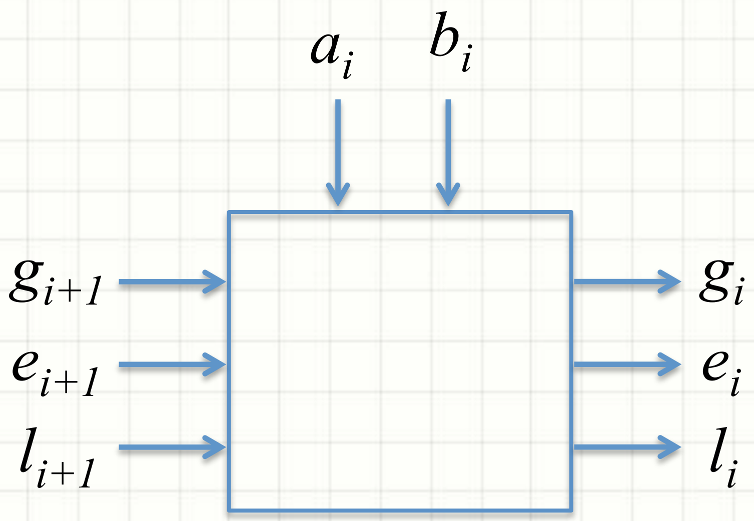
$$e_i \equiv e_{i+1}(\overline{a_i \oplus b_i})$$

$$l_i \equiv l_{i+1} + e_{i+1} \overline{a_i} b_i$$

$g_{32}=0$

$e_{32}=1$

$l_{32}=0$



THANKS