Week3 Boolean Arithmetic

Numbers (Binary Counting)

Binary to Decimal

略

Decimal to Binary

略

Binary Addition

略

Representing Negative Numbers

For binary numbers – define leftmost bit to be the sign

> two zero:1000000 and 00000000

One’s Complement

A negative binary number is the bitwise NOT applied to it — the "complement" of its positive counterpart

Excess-n

A value is represented by the unsigned number which is n greater than the intended value

Two’s Complement (补码)

- To get the negative version of a number
  - Invert the bits
  - Add 1
- So, if we want -29
  - 29 = 0001 1101
  - Invert 1110 0010
  - Add 1 1110 0011
- Try -30

Example of 4-Bit Signed Encodings

Sign and Mag.		Ones' Comp.		Excess-3		Two's Comp.	
1111	−7	1000	−7	0000	−3	1000	−8
1110	−6	1001	−6	0001	−2	1001	−7
1101	−5	1010	−5	0010	−1	1010	−6
1100	−4	1011	−4	0011	0	1011	−5
1011	−3	1100	−3	0100	+1	1100	−4
1010	−2	1101	−2	0101	+2	1101	−3
1001	−1	1110	−1	0110	+3	1110	−2
1000	−0	1111	−0	0111	+4	1111	−1
0000	+0	0000	+0	1000	+5	0000	0
0001	+1	0001	+1	1001	+6	0001	+1
0010	+2	0010	+2	1010	+7	0010	+2
0011	+3	0011	+3	1011	+8	0011	+3
0100	+4	0100	+4	1100	+9	0100	+4
0101	+5	0101	+5	1101	+10	0101	+5
0110	+6	0110	+6	1110	+11	0110	+6
0111	+7	0111	+7	1111	+12	0111	+7

Signed Extension

- For example, if six bits are used to represent the number "00 1010" (decimal +10) and the sign extend operation increases the word length to 16 bits, then the new representation is simply "0000 0000 0000 1010“ – padding the left side with 0s

- If ten bits are used to represent the value "11 1111 0001" (decimal -15) using two's complement, and this is sign extended to 16 bits, the new representation is "1111 1111 1111 0001 – padding the left side with 1s

Overflow

略

Binary Multiplication by Base

```
1100 * 10 = 11000 //Binary
```

Shift Operations

- For unsigned numbers if no bit disappears, shift left corresponds to multiplication by 2
  - E.g. 0011 (3) shifted left is 0110 (6)
- For unsigned numbers, shift right corresponds to division by 2, ignoring the remainder
  - E.g. 0101 (5) shifted right is 0010 (2)
- Arithmetical shift right performs sign extension when shifting
  - E.g. 1100 (-4) shifted right arithmetical results in 1110 (-2)

Adder

• Half adder: adds two bits

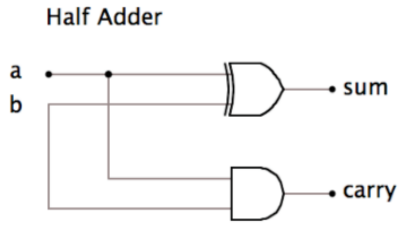
• Full adder: adds three bits

• Adder: adds two integers

Half Adder

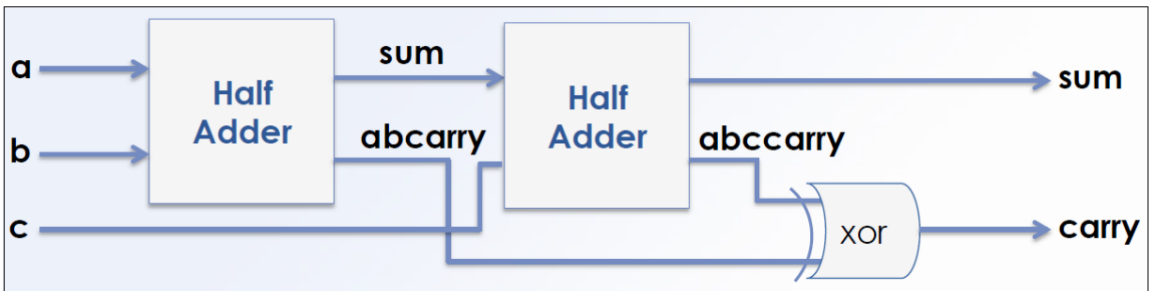
a	b	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

• The common representation uses a XOR and a AND gate



Full Adder

a	b	Carry(in)	Carry(out)	Sum
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

> Full Adder: Implementation
>
> • Use two half adders to build a full adder
>
> 
>