Bit Operations

CM0506 – Small Embedded Systems

Dr Alun Moon

Department of Computer and Information Science

Lecture 4a

Boolean Algebra

The mathematics of logic, true/false, 1/0

Values are :

True	1	High
False	0	Low

Boolean Algebra

The mathematics of logic, true/false, 1/0

Values are :

True	1	High
False	0	Low

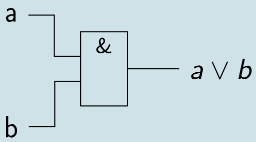
Operations are:

And	$a.b$	$a \vee b$
Or	$a + b$	$a \wedge b$
Not	$\bar{a}$	$\neg a$
Exclusive Or	$a \oplus b$	

And $a \vee b$

Truth Table

a	b	$a \vee b$
0	0	0
0	1	0
1	0	0
1	1	1



Identities

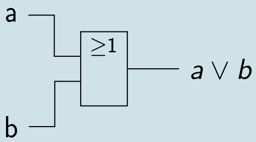
$$a \vee 0 = 0$$

$$a \vee 1 = a$$

Or $a \wedge b$

Truth Table

a	b	$a \wedge b$
0	0	0
0	1	1
1	0	1
1	1	1



Identities

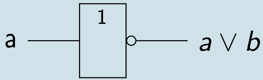
$$a \wedge 0 = a$$

$$a \wedge 1 = 1$$

Not $\neg a$

Truth Table

a	$\neg a$
0	0
1	1



Identities

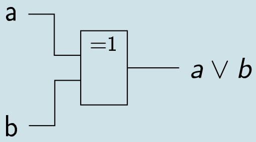
$$a \wedge 0 = a$$

$$a \wedge 1 = 1$$

Exclusive Or $a \oplus b$

Truth Table

a	b	$a \wedge b$
0	0	0
0	1	1
1	0	1
1	1	0



Identities

$$a \oplus 0 = a$$

$$a \oplus 1 = \neg a$$

C Operators

Bitwise logic operators

And $\quad$ a & b

0	0	0	1	0	1	1	0
1	0	0	1	1	0	0	1
0	0	0	1	0	0	0	0

Or $\quad$ a | b

0	0	0	1	0	1	1	0
1	0	0	1	1	0	0	1
1	0	0	1	1	0	1	1

Operating on bits

Often we want to operate on a single bit (or group of bits) within a register.

- IO configuration
- Direction bits
- Output bits

Operating on bits

Often we want to operate on a single bit (or group of bits) within a register.

- IO configuration
- Direction bits
- Output bits

We can use a combination of masks and bitwise operators to

- Set
- Clear
- Test

the state of individual bits.

Masks and notations

Often a bitmask is a pattern of zeros and ones marking a bit-of-interest.

Example

For an 8 bit register, we are interested in bit 5

7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	0

- The *mask* is 00100000
- in C this can be created using shift operators

```
(1<<5)
```

- Usually given a name describing the bit it corresponds to

```
enum { led=(1<<5) };
```

Setting bits

- use the or identities $a \wedge 1 = 1$ and $a \wedge 0 = a$
- Or the register with the mask, writing the result back into the register

```
dir = dir | led;
```

or

```
dir |= led;
```

Example

- Register is $\boxed{1\,1\,0\,0\,0\,0\,1\,1}$
- Applying the mask led as above
- gives $\boxed{1\,1\,1\,0\,0\,0\,1\,1}$

Clearing bits

- use the and identities $a \vee 0 = 0$ and $a \vee 1 = a$
- Here we use and with not

```
dir = dir & ~but;
```

or

```
dir &= ~but;
```

Example

- Register is $\boxed{1\,1\,1\,0\,0\,0\,1\,1}$
- Applying a mask defined as

```
enum { but=(1<<6) };
```

- gives $\boxed{1\,0\,1\,0\,0\,0\,1\,1}$

Toggling bits

- use the exclusive or identities $a \oplus 1 = \neg a$ and $a \oplus 0 = a$
- Here we use xor

```
dir = dir ^ func;
```

or

```
dir ^= func;
```

Example

- Register is $\boxed{1\,0\,1\,1\,0\,0\,1\,1}$
- Applying a mask defined as

```
enum { func=(3<<1) };
```

- gives $\boxed{1\,0\,1\,1\,0\,0\,1\,0\,1}$

Testing bits

- use the properties of and with the behaviour of C
- and the mask with the register

```
pin & but;
```

- if the result is
  - 0 the bit is 0
  - not-0 the bit is 1

Testing bits

Example

- with the pin Register as $\boxed{1\,0\,1\,1\,0\,0\,1\,1}$
- and a mask defined as

```
enum { but=(1<<6) };
```

```
pin & but
```

- gives $\boxed{0\,0\,0\,0\,0\,0\,0\,0}$
- C tests this as false

```
        if( pin & but) {
          button = pressed;
        }
```

Testing bits

Example

- with the pin Register as $\boxed{1\,1\,1\,1\,0\,0\,1\,1}$
- and a mask defined as

```c
enum { but=(1<<6) };
```

```c
pin & but
```

- gives $\boxed{0\,1\,0\,0\,0\,0\,0\,0}$
- C tests this as true $(2^6 = 64)$

```c
        if( pin & but) {
          button = open;
        }
```