BINARY NUMBER FORMATS

Binary Number Formats

- Read Dally textbook
  - Chapter 10 – Binary numbers, add, subtract, multiply, divide
  - Chapter 11 – Floating point
  - ~~Chapter 12 – Fast arithmetic~~, Skip for EEC 180
  - Chapter 13 – Arithmetic examples
- Binary numbers
  - All number systems considered in EEC 180:
    - are n-digit — $a_{n-1}, a_{n-2}, \ldots, a_1, a_0$
    - are "binary" — the base $b = 2$
    - LSB — Least Significant Bit (Digit)
    - MSB — Most Significant Bit (Digit)
  - Accuracy or Precision: "the maximum error over a number's input range" [Chapter 11]

Binary Number Formats

Consider for each: positional weights, range, and zero(s)

1) Unsigned — range of $[0, 2^n - 1]$

$$value = \sum_{i=0}^{n-1} a_i b^i$$

2) Sign Magnitude

S	unsigned

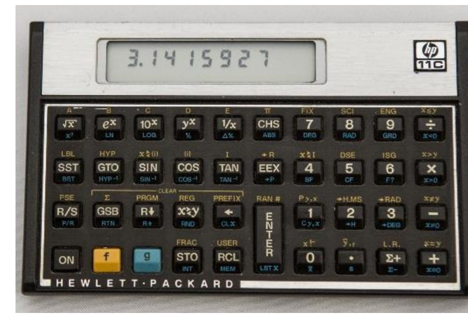
3) Signed 2's complement — The positional weight of the MSB is negative.
range of $[-2^{(n-1)}, +2^{(n-1)} - 1]$

4) Signed 1's complement — Not used for hardware

5) BCD — Binary-Coded Decimal
   – Each base-10 digit is coded with 4 binary bits

Motivation for using the BCD format

- By necessity:
  - For example, displaying a number on a display in base 10
  - For example, inputting a number from a 10-key keypad from a user
  - High-accuracy financial calculations
- In some cases, processing is done in a “normal” binary format and so input/output must be converted from/to BCD
- In some cases, processing may be done in BCD format directly. Most likely for applications that perform simple operations on data that is input and/or output in BCD format.



Common Binary Number Formats

- Binary numbers
  - Ex: 0000_0101

Ex: 0000_0101	= 5 (base 10)	unsigned
	= +5 (base 10)	sign-magnitude
	= +5 (base 10)	signed 2's complement
	= 0 5	BCD
Ex: 1000_0011	= 131 (base 10)	unsigned
	= –3 (base 10)	sign-magnitude
	= –125 (base 10)	signed 2's complement
	= 8 3	BCD

  - A) Integer
  - B) Fractional
    - Where f is the number of fractional bits
    - Format can be unsigned, sign-magnitude, 2's complement

$$\sum_{i=0}^{n-1} a_i b^{i-f}$$

Common Binary Number Formats

- Binary numbers
  - B) Fractional
    - Ex: Positional weights for 2's complement 5.3 format:
      –16 8 4 2 1 . 1/2 1/4 1/8
    - Ex: Positional weights for unsigned 5.3 format:
      16, 8, 4, 2, 1 . 1/2, 1/4, 1/8
    - Ex: 1010_0.001 5.3 in different formats:

= 20 1/8	unsigned 5.3 format
= –4 1/8	sign-magnitude 5.3 format
= –11 7/8	2's complement 5.3 format

    - "There is no decimal point in the hardware"
    - The hardware for an 8.0 format adder is the same as for 7.1, 5.3, etc.
  - C) Full fractional
    - This is really a special case of (B) Fractional with no bits for the whole number portion of the number
    - Ex: 0.16 format

Converting BCD → Unsigned Binary

- Converting BCD format to unsigned binary is not difficult
- To convert a 3-digit BCD input to unsigned format, add the following values:
  - 100 × *Hundreds-digit*
  - 10 × *Tens-digit*
  - *Ones-digit*
- For example, 135 (BCD) converted to unsigned:

–	100 (base 10)	0110_0100
	30 (base 10)	0001_1110
	5 (base 10)	0000_0101
	sum	1000_0111 = 128 + 4 + 2 + 1 = 135 check

Converting Unsigned Binary → BCD

- There is no super-simple way to convert an unsigned binary number to BCD format
- As an example, take a long look at the 6-bit binary number and the corresponding 2-digit BCD number where values are ten or greater
- Conversions will generally require the following steps (for two BCD digits):
  - Find the tens position, for example by testing various tens ranges, e.g.,

```
if (in >= 60 && in < 70) begin  tens = 6;  end   // verilog pseudo-code
// There are simpler ways to implement this but this works.
```

  - Calculate the remainder with something like:

```
rem = in - (10 * tens);                           // verilog pseudo-code
```

Unsigned binary	Base 10	BCD
000000	0	0000 0000
000001	1	0000 0001
000010	2	0000 0010
000011	3	0000 0011
000100	4	0000 0100
…	…	…
111100	60	0110 0000
111101	61	0110 0001
111110	62	0110 0010
111111	63	0110 0011