ELEC2204

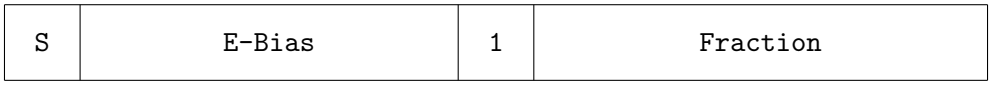
Computer Engineering

Updated: February 5, 2017

Contents

1 Arithmetic

1.1 Floating Point Numbers



1.2 Converting Decimal to Floating Point

- Convert integer part to binary.
- Convert fractional part by multiplying by 2 each time and shifting the decimal point right one until there is no fractional part left.
- Append 2^0 to the end of the number, this has no effect.
- Normalise the number by moving your binary point to one bit from the left (be sure to adjust exponent accordingly).
- The fractional field is then the bits to the right of the binary point. Pad the remaining bits with 0's, so for IEEE 32-bit floats the fractional field has 23 bits.
- Add an exponent bias of $2^{k-1} - 1$ where k is the number of bits in the exponent field. For IEEE 32-bit float, $k = 8$, so the bias is $2^{8-1} - 1 = 127$.
- Set the sign bit as `1` if the number is negative, `0` otherwies.

Convert 2.625 to our 8-bit floating point format.

Integer part, $2_{10} = 10_2$.
Fraction part,

$$0.625 \times 2 = 1.25 \quad \boxed{1}$$
$$0.25 \times 2 = 0.5 \quad \boxed{0}$$
$$0.5 \times 2 = 1.0 \quad \boxed{1}$$

So, $0.625_{10} = 0.101_2$ and $2.625_{10} = 10.101_2$.

$$\text{Add exponent part, } 10.101_2 = 10.101_2 \times 2^0$$
$$\text{Normalise, } 10.101_2 \times 2^0 = 1.0101_2 \times 2^1$$
$$\text{Fractional field, } 0101_2$$
$$\text{Exponent, } 1 + 127 = 128_{10} = 10000000_2$$
$$\text{Sign bit, } 0$$

$\therefore 2.625_{10} =$ | 0 | 10000000 | 01010000000000000000000 |

1.3 Adders

1.3.1 Ripple Adder

An n-bit ripple adder is constructed by cascading n full adder cells. The size of this is proportional to the number of cells you have. However since cell $n + 1$ takes the carry bit from the previous addition the delay will also be proportional to n.

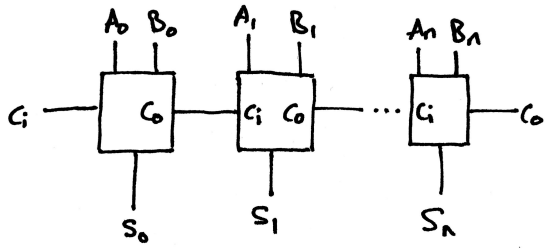


Figure 1: Cascading full adders.

1.3.2 Carry–Lookahead Adder

A carry-lookahead adder works by calculating the carry of each adding stage separately. This makes the delay constant regardless of the number of bits but also low (only chip propagation delays). However each stage is geometrically larger than the previous and it is said that size $\propto n^2$.

1.4 Multicycling Addition

1.5 Floating Point Addition/Subtraction

1.6 Floating Point Multiplication/Division

1.7 Microcoded Operations

Microcode allows us to add simple functionality that isn't already present in hardware. Below is an example of a 32-bit unsigned square root in C code.

```
int fn sqrt(int val)
{
    int mask = 0x00008000;
    int best = 0x00000000;
    if (val <= 0) return 0;
    while (mask != 0) {
        if ((best+mask^2) <= val) best |= mask;
        mask >>= 1;
    }
    return (int)best;
}
```