Overview

- ALUs - arithmetic Logic Units
- Shifters
- Number systems

HW #4 LSN 24-27 due LSN 29

Arithmetic Logic Unit (ALU):

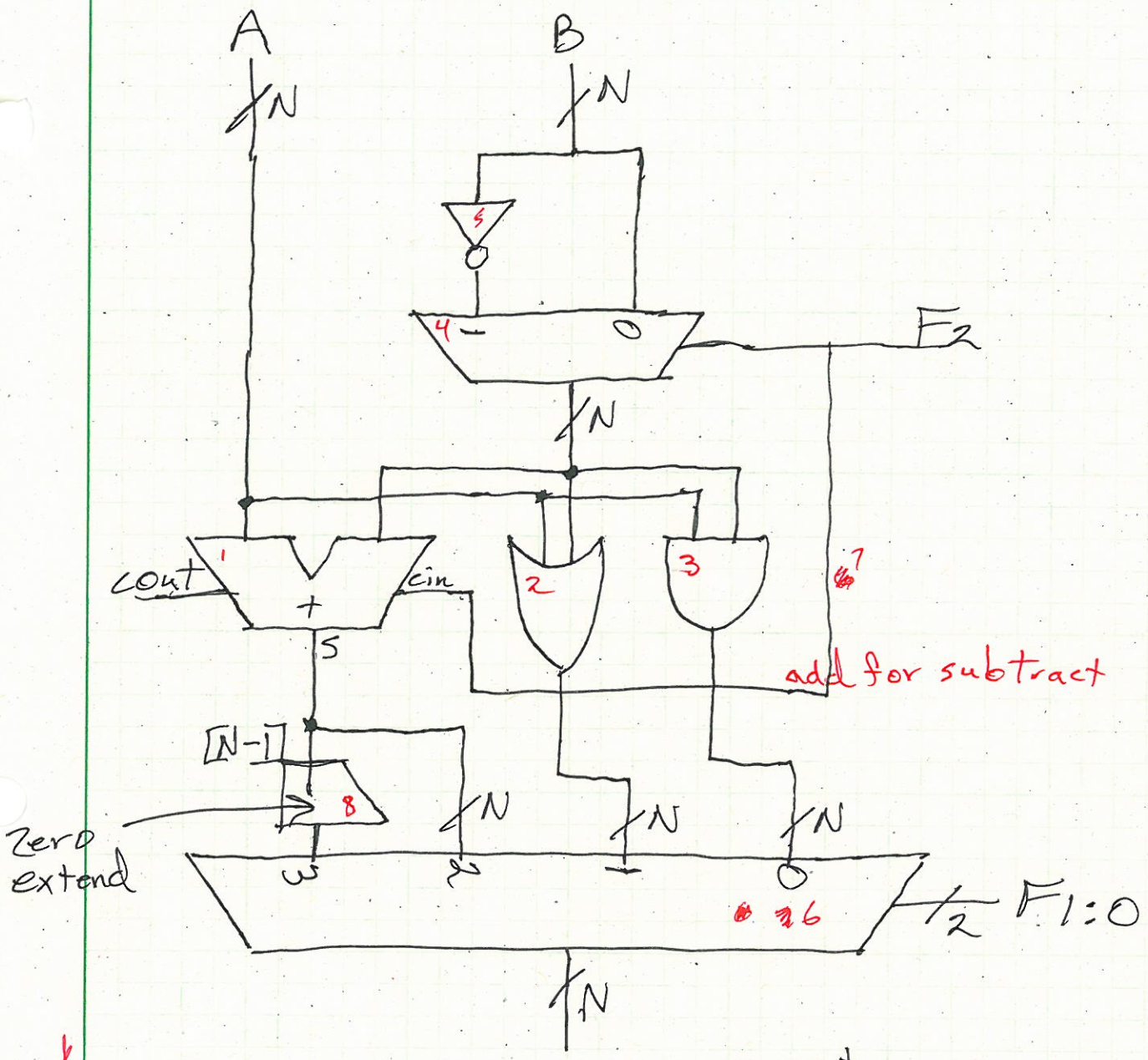
- combines math and Logic
- heart of most comparators

Does 7 things:

F_2 F_1 F_0	operation
0 0 0	A and B
0 0 1	A or B
0 1 0	A + B
0 1 1	not used
1 0 0	A and $\overline{B}$
1 0 1	A or $\overline{B}$
1 1 0	A - B
1 1 1	SLT

ALU Built using what? Logic Functions?

- adder
- inverter
- and
- mux
- or
- comparator



Magnitude comparison: Compute A − B and looking at the sign (most significant bit) of the result. If the result is negative (i.e. sign bit is 1), then A is < B. Otherwise A is greater than or equal to B

How dowe multiply in Binary

```
 101   =5
x011   x3
 101
1010
0000
----
1111 => 15
```

Shifter:

- Logical → empty bits <=0 (LSL or LSR)
- arithmetic → on R shifts, keep MSB useful for mult/Div signed Numbers
- ASL is same as LSL!

⇒ Shamt = shift amount

100110 ⇒ shift R ⇒ 010011

32 + +4+2 = 38 16 + 2+1 = 19

Shift R divide by 2

100110 ⇒ shift L ⇒ 001100 = 12 ⇒ 76?

We ran out of Bits and lost a Bit

6-Bit system only goes up to 63!

Rotator:

- Bits that fall off the end move to the other end.

Fixed Point:

- implies binary point btwn integer & fractional bits
- You keep track of radix!

Q 2.6

XX.XXXXX

What is this number in Decimal?

01101.10101 (given in 2's comp)

$16+8+4+1 = 29$

$0.5 = 2^{-1}$

$0.25 = 2^{-2}$

$0.125 = 2^{-3}$

$0.03125 = 2^{-5}$

$\Rightarrow 29.65625$

101000.00100 (2's comp)

Flip 010111.11011 (add 1 to LSB)

$+1$

010111.11100

-23 $\quad$ 0.875

101000 Flip 010111 $+1$ $= 011000 = 24$

00100 Flip 11011 $+1$ $= 11100$

$0.5 + 0.25 + 0.125 \Rightarrow .875$

$\Rightarrow -23.875$

Fixed point easier in HW

Floating Point:

- similar to scientific notation

2.3×10^{6}

sign, mantissa, Base, exponent

1	8 Bits	23 bits

sign

163_{10}

1010 0011

1.01000011×2^{7}

⇒ [0] [00000111] [10100011 00.... 0]

sign exponent Mantissa

Finally standardized w/ IEEE 754 – floating point ~~notation~~ standard

⇒ add 127 to exponent

⇒ do not include Leading 1

Implicit Leading One
Leading 1 is always 1 therefore need not be stored

~~This is~~ 0 00000111 10100011 00.... 0

→ IEEE 0 10000110 01000000000.....00

For 32 Bit Float

7 + (Bias) 127 = 134 ← Biased exponent for + & − exponents

adds another bit for useful data

This is a double = 64 Bits

11 exponent

52 fraction

★ Look at example in book of floating point addition!

★ Figure 5.29 – Floating Point Addition

Floating Point Addition:

1) extract exponent & Fraction Bits
2) add Leading 1 back to make mantissa
3) compare exponents
4) shift smaller ~~amount~~ exponent's mantissa
5) add mantissas
6) shift mantissas and adjust mantissas if necessary
7) Round if necessary
8) turn back into Legit (IEEE 754) Floating Point