

FP Addition

- Assume
 - A represented as bit pattern $[S_A, E_A, F_A]$
 - B represented as bit pattern $[S_B, E_B, F_B]$
- What is the bit pattern for $A+B$ $[S_{A+B}, E_{A+B}, F_{A+B}]$?
 - $[S_A+S_B, E_A+E_B, F_A+F_B]$? Nope!
 - So what is it then?

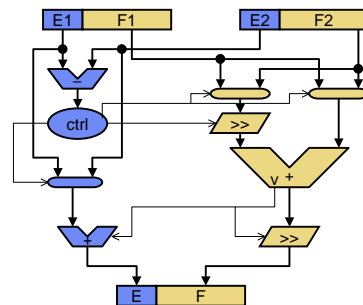
FP Addition Decimal Example

- Let's look at a decimal example first: $99.5 + 0.8$
 - $9.95 * 10^1 + 8.0 * 10^{-1}$
- Step I: **align exponents (if necessary)**
 - Temporarily** de-normalize operand with smaller exponent
 - Add 2 to its exponent → must shift significand right by 2
 - $8.0 * 10^{-1} \rightarrow 0.08 * 10^1$
- Step II: **add significands**
 - $9.95 * 10^1 + 0.08 * 10^1 \rightarrow 10.03 * 10^1$
- Step III: **normalize result**
 - Shift significand right by 1 and then add 1 to exponent
 - $10.03 * 10^1 \rightarrow 1.003 * 10^2$

FP Addition (Quarter Precision) Example

- Now a binary "quarter" example: $7.5 + 0.5$
 - $7.5 = 1.875 * 2^2 = 0\ 101\ 11110$ (the **1** is the implicit leading 1)
 - $1.875 = 1 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2} + 1 * 2^{-3}$
 - $0.5 = 1 * 2^{-1} = 0\ 010\ 10000$
- Step I: **align exponents (if necessary)**
 - $0\ 010\ 10000 \rightarrow 0\ 101\ 00010$
 - Add 3 to exponent → shift significand right by 3
- Step II: **add significands**
 - $0\ 101\ 11110 + 0\ 101\ 00010 = 0\ 101\ 10000$
- Step III: **normalize result**
 - $0\ 101\ 10000 \rightarrow 0\ 110\ 10000$
 - Shift significand right by 1 → add 1 to exponent

FP Addition Hardware



What About FP Subtraction?

- Or addition of negative quantities for that matter
 - How to subtract significands that are not in TC form?
 - Can we still use an adder?
- Trick: internally and temporarily convert to TC
 - Add "phantom" -2 in front ($-1 * 2^1$)
 - Use standard negation trick
 - Add as usual
 - If phantom -2 bit is 1, result is negative
 - Negate it using standard trick again, flip result sign bit
 - Then ignore "phantom" bit (which is now 0 anyway)
 - You'll want to try this at home!