CSE471—Computer Design and Organization

Prof. Steven Burns *Autumn 1994*

Lecture Notes: Ninth Day

Floating Point

- IEEE Floating point standard.
- Sign, Exponent, Significand (Mantissa)
- Single Precision

1	8	23
s	$yyyy\ yyyy$	$xxx\ xxxx\ xxxx\ xxxx\ xxxx\ xxxx$
sign	exponent	significand

- Double Precision

1	11	20
s	$yy\ yyyy\ yyyy$	$xxxx\ xxxx\ xxxx\ xxxx\ xxxx$
sign	exponent	significand

32
$xxxx\ xxxx\ xxxx\ xxxx\ xxxx\ xxxx\ xxxx\ xxxx$
significand

Floating Point

- The bit patterns in IEEE floating point numbers mean:

$$(-1)^s \times 1.xxx\cdots xx \times 2^{yyy\cdots yy-\text{bias}}$$

  unless $yyy\cdots yy$ is all zeroes or all ones.
- The bias value is 127 for single precision and 1023 for double precision.
- Why did they choose to do this in such a weird way?
- An integer (sign-magnitude) comparison can be used to order floating point numbers.

Floating Point Addition

- How do we do a floating point add?
- We must first line up the significands to account for different exponents.
- Then we perform the addition, and renormalized the result to scientific notation.
- Example: Base 10, Four digits of precision

$$
\begin{array}{rcl}
9.999 & \times & 10^{1} \\
1.610 & \times & 10^{-1} \\
\hline
9.999 & \times & 10^{1} \\
0.016 & \times & 10^{1} \\
\hline
10.015 & \times & 10^{1} \\
\hline
1.002 & \times & 10^{2}
\end{array}
$$

Rounding Modes

- How do you round if the last digit (bit) is halfway between 0 and the radix? toward $+\infty$, toward $-\infty$, toward 0, toward the nearest even (default and most complex).
- The IEEE standard says that the result of an FP operation should be the *same* as if it were computed exactly and then rounded according to the current mode.
- Rounding occurs for two reasons: aligning the significands, and converting the result back to normalized form.

Implementing Exact Rounding

- On the surface, it looks expensive to do exact rounding. Consider:

$$
\begin{array}{l}
4.5674 \times 10^{0} \\
2.5001 \times 10^{-4} \\
\hline
4.56740000 \times 10^{0} \\
0.00025001 \times 10^{0} \\
\hline
4.56765001 \\
\hline
4.5677
\end{array}
$$

- It is conceivable that we need $2p$ bits to do exact addition (p is number of bits in the significand.) Fortunately, we can get away with fewer bits.

Rounding Examples

- Here is an example where we need two extra digits:

$$
\begin{array}{rl}
 & 1.01 \times 10^2 \\
- & 3.76 \times 10^0 \\
\hline
 & 1.0100 \times 10^2 \\
- & 0.0376 \times 10^2 \\
\hline
 & 0.9724 \times 10^2 \\
\hline
 & 9.724 \times 10^1
\end{array}
\qquad\qquad
\begin{array}{rl}
 & 1.010 \times 10^2 \\
- & 0.037 \times 10^2 \\
\hline
 & 0.973 \times 10^2 \\
\hline
 & 9.730 \times 10^2
\end{array}
$$

- It is possible to prove that all you need are two bits (for all the modes except the round to even.) These bits are called *guard* and *round*.

Verilog

- Implementation of (positive) FP addition

```
reg [63:0] a, b, c;
reg [12:0] expdiff;
reg [53:0] am, bm, cm;
reg guard, round;
   c[63] = 0;
   expdiff = a[62:52] - b[62:52];
   if ( a[62:52] < b[62:52]) begin
      {am,guard,round} = {2'b01,a[51:0],2'b00} >> (-expdiff);
      bm = {2'b01,b[51:0]};
      c[62:52] = b[62:52];
   end else begin
      am = {2'b01,a[51:0],2'b00};
      {bm,guard,round} = {2'b01,b[51:0],2'b00} >> expdiff;
      c[62:52] = a[62:52];
   end
   cm = am + bm;
```

```
if ( cm[53] == 1) begin
   {cm,guard,round} = {cm,guard,round}>>1;
   c[62:52] = c[62:52] + 1;
end
if ( guard == 1 && round == 1) // to Zero (-inf)
   cm = cm + 1;
if ( cm[53] == 1) begin
   cm = cm>>1;
   c[62:52] = c[62:52] + 1;
end
c[51:0] = cm[51:0];
```

What About Round to Even?

- You need an extra bit, called the *sticky* bit, which is set to one if the bits shifted off in the alignment phase are not all zero.

```
...
if ( a[62:52] < b[62:52]) begin
   sticky = ((1 << (-expdiff)) - 1) & {2'b01,a[51:0],2'b00};
   {am,guard,round} = {2'b01,a[51:0],2'b00} >> (-expdiff);
   bm = {2'b01,b[51:0]};
   c[62:52] = b[62:52];
end else
...
```

- At the rounding point, we round up if: guard is one and round is one, or guard is one and round is zero and (sticky is true or the last bit is one).

FP Multiply

- Conceptually simpler than addition.
- Multiply the significands and add the exponents (bias notation means you must subtract the bias after you perform the sum.)
- You must normalize the result and round as in the addition case.