ROUNDING
Rounding

- Rounding operations eliminate lower-weight bits; in other words, bits on the LSB end of the word
- It is commonly necessary to reduce the number of bits due to word growth
  - For example, if we multiply two 5-bit words, the product will have 10 bits
    \[ \text{xxxxx} \times \text{yyyyy} = \text{zzzzzzzzzzzz} \]
    and we likely can not handle or do not want or need all that precision
  - In some cases such as when processed data must be written back into its source memory, a reduction in word size is clearly required
Rounding

- Rounding modes in IEEE 754 are much more complex than what is commonly needed in digital signal processing systems

- Matlab rounding
  1) `round(·)`: towards nearest integer
     - Pos. and neg. numbers are rounded symmetrically about zero
     - Generally the best possible rounding algorithm
  2) `fix(·)`: truncates towards zero
     - Pos. and neg. numbers are rounded symmetrically about zero
  3) `floor(·)`: rounds towards negative infinity
  4) `ceil(·)`: rounds towards positive infinity
1) matlab round()

- One of the best general-purpose rounding modes
- “Unbiased” rounding
- Symmetric rounding for positive and negative numbers
- Max error $\frac{1}{2}$ LSB
2) matlab fix()

- Truncates toward zero
- Numerical performance poor
- Symmetric rounding for positive and negative numbers
- Max error 1 LSB
3) Truncation, or matlab floor()

- Numbers rounded down towards –infinity
- Numerical performance poor
- Very simple hardware
- in -- out
- Max error 1 LSB
4) matlab ceil()

- Numbers rounded up towards +infinity
- Numerical performance poor
- Max error 1 LSB
Hardware Rounding: A) Truncation

A. The easiest is truncation

- \texttt{xxx.xxxxx}
  \texttt{xxx.xx---}

- Maximum rounding error \~1 post-rounded LSB

- Signed magnitude format numbers
  - Positive and negative numbers both truncate towards zero
  - Same as matlab \texttt{fix(\cdot)}

- \texttt{2\textquotesingle s complement} and unsigned format numbers
  - All numbers truncate towards negative infinity
  - Same as matlab \texttt{floor(\cdot)}
Hardware Rounding: B) Add ½ LSB and Truncate

B. A better rounding numerically is to add ½ LSB (that is, one half of the LSB of the output) and then truncate

\[
\begin{array}{c}
1 \\
+ \ xxx.xxxxx \\
\hline
YYY \cdot YYYYYx \\
\hline
YYY \cdot YY---
\end{array}
\]

- Our 5th rounding method; it does not correspond to any of the matlab rounding functions
- Maximum rounding error ½ post-rounded LSB
- Two cases:
  a. When the input is xxxx.5000 (base 10) (or xxx.xx100 (base 2) in the example above)
      - Rounding is towards positive infinity (for both positive and negative numbers)
      - Same as matlab ceil(•)
  b. Otherwise
      - It performs the best rounding: matlab round(•)
Hardware Rounding:
B) Add $\frac{1}{2}$ LSB and Truncate

- It is often not difficult to find a place to add the extra “1” if you plan ahead.
Hardware Rounding: B) Add $\frac{1}{2}$ LSB and Truncate

- But there is a biased rounding in the $\text{xxx}.1\text{000}$ cases
  - The bias may be fine in many cases, especially when many bits are being rounded off (then the $\text{xxxx}.1\text{0000}$ case is less frequent as shown in this example where five LSB bits are being rounded off)
  - The exact behavior depends on the number format being used:
    - Signed magnitude
      - Both positive and negative $\text{xxxx}.1\text{000}$ cases round away from zero
    - 2’s complement and unsigned
      - Both positive and negative $\text{xxxx}.1\text{000}$ cases round towards positive infinity
Add \( \frac{1}{2} \) LSB and Truncate 2’s Complement

- matlab `floor(x+1/2)`
- The numerical performance is often sufficient
- Biased rounding for 2’s complement
- Max error \( \frac{1}{2} \) LSB
Add $\frac{1}{2}$ LSB and Truncate

Signed Magnitude

- matlab \texttt{floor(x+1/2)}
- Functions same as \texttt{matlab round()} which is the best of our four \texttt{matlab} rounding functions
- Unbiased rounding for signed magnitude
- Max error $\frac{1}{2}$ LSB