ROUNDING
Rounding

• Rounding is a fundamental method to reduce the size of a word, such as after arithmetic operations
  – For example to maintain the word width for memory storage

• Bits are removed from the LSB end of the word

© B. Baas
Rounding

• Another example: if we multiply two 5-bit words, the product will have 10 bits

  \[ xxxxx \times yyyyy = zzzzzzzzzz \]

  and we likely can not handle or do not want or need all that precision

• More issues are present with signed data

• Issues vary for different formats:
  – unsigned
  – 2’s complement
  – sign magnitude
  – etc.
Rounding

• Rounding modes in IEEE 754 are much more complex than what is commonly needed in digital signal processing systems
• There are four fundamental rounding modes whose matlab function names are:
  1) `round(·)`: towards nearest integer
     • Generally the best rounding algorithm
  2) `fix(·)`: truncates towards zero
  3) `floor(·)`: rounds towards negative infinity
  4) `ceil(·)`: rounds towards positive infinity
1) matlab round()

- Often the best general-purpose rounding mode
- "Unbiased" rounding
- Symmetric rounding for positive and negative numbers
- Max error $\frac{1}{2}$ LSB
2) matlab fix()

- Truncates toward zero
- Numerical performance is poor
- Symmetric rounding for positive and negative numbers
- Very simple hardware for the magnitude of sign magnitude (simple truncation)
  - xxxx in
  - xxxx-- out
- Max error 1 LSB
3) matlab floor()

- Numbers rounded down towards $-\infty$
- Numerical performance is poor
- Very simple hardware for 2’s complement (simple truncation)
  - $\ldots\ldots\ldots\ldots$ in
  - $\ldots\ldots\ldots\ldots$ out
- Max error 1 LSB
4) matlab ceil()

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

A. The easiest hardware method is truncation
   - \( xxx.xxxxx \)
     \( xxx.xx--- \)
   - Simply neglect the truncated bits and remove all hardware which calculates only those bits
   - Maximum rounding error \( \sim 1 \) post-rounded LSB
   - Sign magnitude format numbers (obviously the magnitude portion)
     - Positive and negative numbers both truncate towards zero
     - Same as matlab \( \text{fix}(\cdot) \)
   - 2’s complement format numbers
     - All numbers truncate towards negative infinity
     - Same as matlab \( \text{floor}(\cdot) \)
   - Unsigned format numbers
     - All numbers truncate towards zero (negative infinity)
     - Same as matlab \( \text{fix}(\cdot) \) or \( \text{floor}(\cdot) \)
B. **Method #5. Add ½ LSB (that is, one half of the LSB of the output) and then truncate**

- This does not correspond to any of the matlab rounding functions
- Maximum rounding error ½ of the post-rounded LSB
- Three cases to consider for 2’s complement:
  a. When the input is of the form xxxxx.100 (base 2) in the example above, and positive
     - Rounding is towards positive infinity which is the same as `round(•)`
  b. When the input is of the form xxxxx.100 (base 2) in the example above, and negative
     - Rounding is towards positive infinity which is NOT the same as `round(•)`
  c. Otherwise
     - It performs the same as 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” in a complex datapath if you plan ahead.

\[ \text{keep these bits} \quad \begin{array}{c} 1 \\ \hline \end{array} \quad \text{truncate these bits after adding everything} \]

“1” rounding bit has a weight of \(\frac{1}{2}\) of the post-rounded LSB.
Hardware Rounding:  
B) Add \( \frac{1}{2} \) LSB and Truncate

- It is often not difficult to find a place to add the extra \("1\"\) in a complex datapath if you plan ahead

- "1" rounding bit has a weight of \( \frac{1}{2} \) of the post-rounded LSB

- Keep these bits

- Truncate these bits after adding everything

- Post-rounded LSB position
Hardware Rounding: B) Add $\frac{1}{2}$ LSB and Truncate

- The exact behavior depends on the number format being used:
  - Magnitude portion of Sign magnitude
    - Unbiased rounding for sign magnitude
  - 2’s complement
    - Both positive and negative xxxxx.1000 cases round towards positive infinity as explained previously
    - The behavior requires a little more analysis
B) Add $\frac{1}{2}$ LSB and Truncate

**Unsigned, Sign Magnitude**

- matlab `floor(x+1/2)`
- matlab `fix(x+1/2)`
- Both positive and negative xxxx.1000 cases round away from zero just like `round()`
- Functions the same as matlab `round()` which is the best of our four matlab rounding functions
- Max error $\frac{1}{2}$ LSB

![Graph](image.png)
B) Add $\frac{1}{2}$ LSB and Truncate

2’s Complement

- $\text{matlab } \textit{floor}(x+1/2)$
- The numerical performance is often sufficient

- $1$
  \[
  \begin{array}{c}
  + \underbrace{x\cdots x} \\
  \overbrace{y\cdots y} \\
  \underbrace{y\cdots y}
  \end{array}
  \]
- Biased rounding for 2’s complement
- Max error $\frac{1}{2}$ LSB
B) Add ½ LSB and Truncate 2’s Complement

• The biased rounding in the \( \text{xxx.1000} \) cases when using 2’s complement may be fine in many cases, especially when many bits are being rounded off, but if only a few bits are being rounded off, the case that differs from round() occurs more often.

• Example: \( \text{xxxxxx.x} \) rounded to \( \text{yyyyyyy} \)

<table>
<thead>
<tr>
<th>Pre-rounded value</th>
<th>Rounding action</th>
<th>Net effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+) ( \text{xxxxxx.0} )</td>
<td>No change in value</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(+) ( \text{xxxxxx.1} )</td>
<td>Rounds to integer +0.5</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(−) ( \text{xxxxxx.0} )</td>
<td>No change in value</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(−) ( \text{xxxxxx.1} )</td>
<td>Rounds to integer +0.5</td>
<td>Same as round() +1</td>
</tr>
</tbody>
</table>
B) Add $\frac{1}{2}$ LSB and Truncate

2’s Complement

- Example: positive values of $xxxxxx.xxxxxx$ rounded to $yyyyyy$

<table>
<thead>
<tr>
<th>Pre-rounded value</th>
<th>Rounding action</th>
<th>Net effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+) $xxxxxx.000000$</td>
<td>No change in value</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(+) $xxxxxx.000001$</td>
<td>Rounds down to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(+) $xxxxxx.000010$</td>
<td>Rounds down to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) $xxxxxx.100000$</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) $xxxxxx.111101$</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(+) $xxxxxx.111110$</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(+) $xxxxxx.111111$</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
</tbody>
</table>
### B) Add ½ LSB and Truncate

#### 2’s Complement

- **Example:** negative values of \( \text{xxxxxx.xxxxxx} \) rounded to \( \text{yyyyyy} \)

<table>
<thead>
<tr>
<th>Pre-rounded value</th>
<th>Rounding action</th>
<th>Net effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>((-) xxxxxx.000000)</td>
<td>No change in value</td>
<td>Same as round()</td>
</tr>
<tr>
<td>((-) xxxxxx.000001)</td>
<td>Rounds down to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>((-) xxxxxx.000010)</td>
<td>Rounds down to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>((-) xxxxxx.100000)</td>
<td>Rounds up to integer</td>
<td>Same as round() +1</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>((-) xxxxxx.111101)</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>((-) xxxxxx.111110)</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
<tr>
<td>((-) xxxxxx.111111)</td>
<td>Rounds up to integer</td>
<td>Same as round()</td>
</tr>
</tbody>
</table>
matlab for previous plots

• copy, paste, and try it out