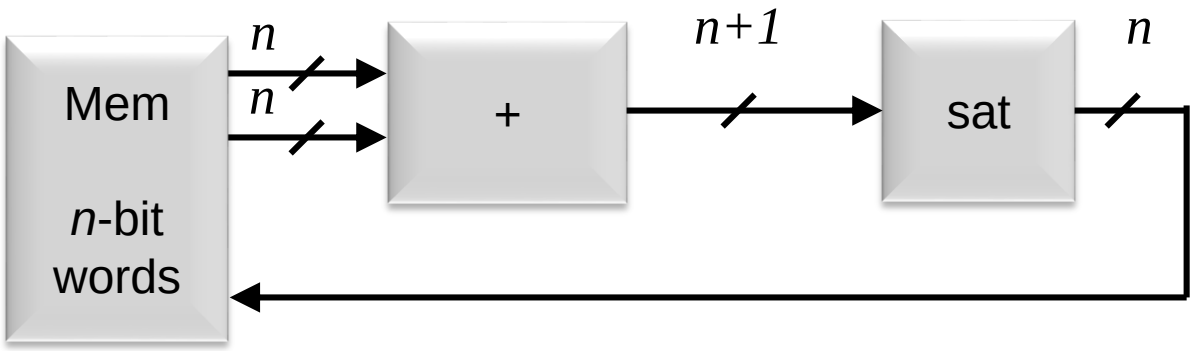
ROUNDING

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

```
XXXXXXXX
 YYYYY
```

Rounding

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

  `xxxxx × yyyyy = zzzzzzzzzz`

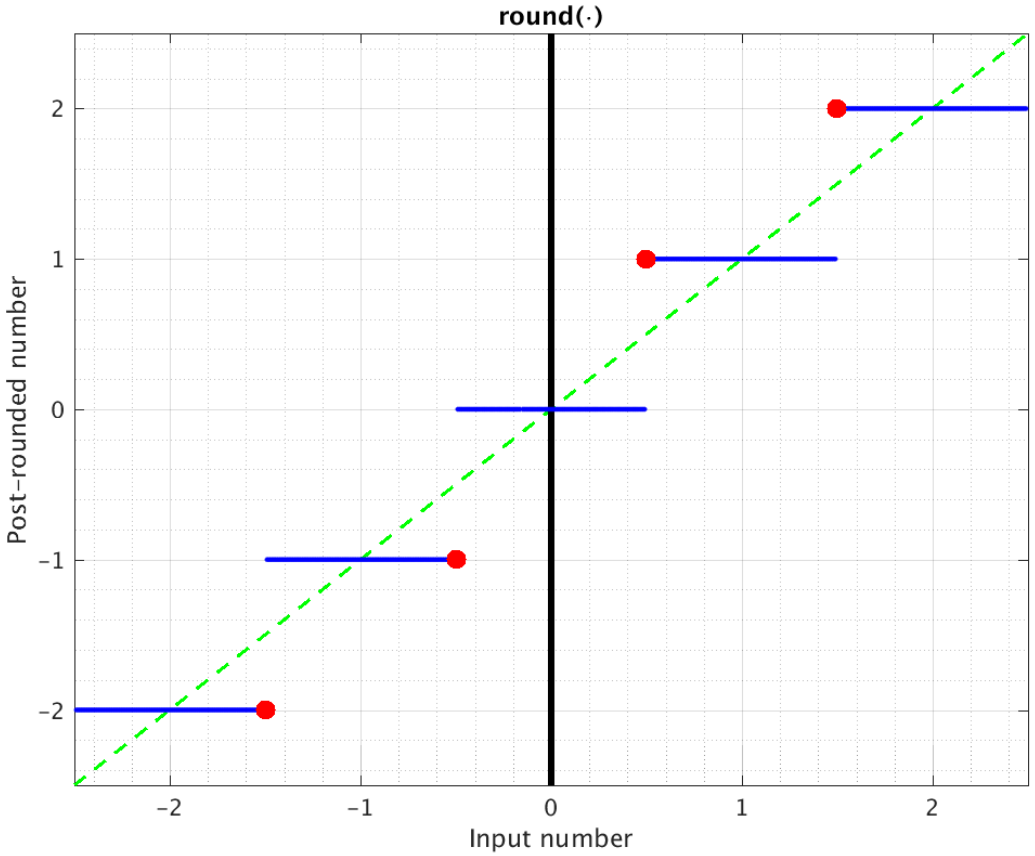
  and we likely can not handle or do not want or need all that precision
- More issues are present with signed data
- Issues vary for different formats:
  - unsigned
  - 2's complement
  - sign magnitude
  - etc.

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

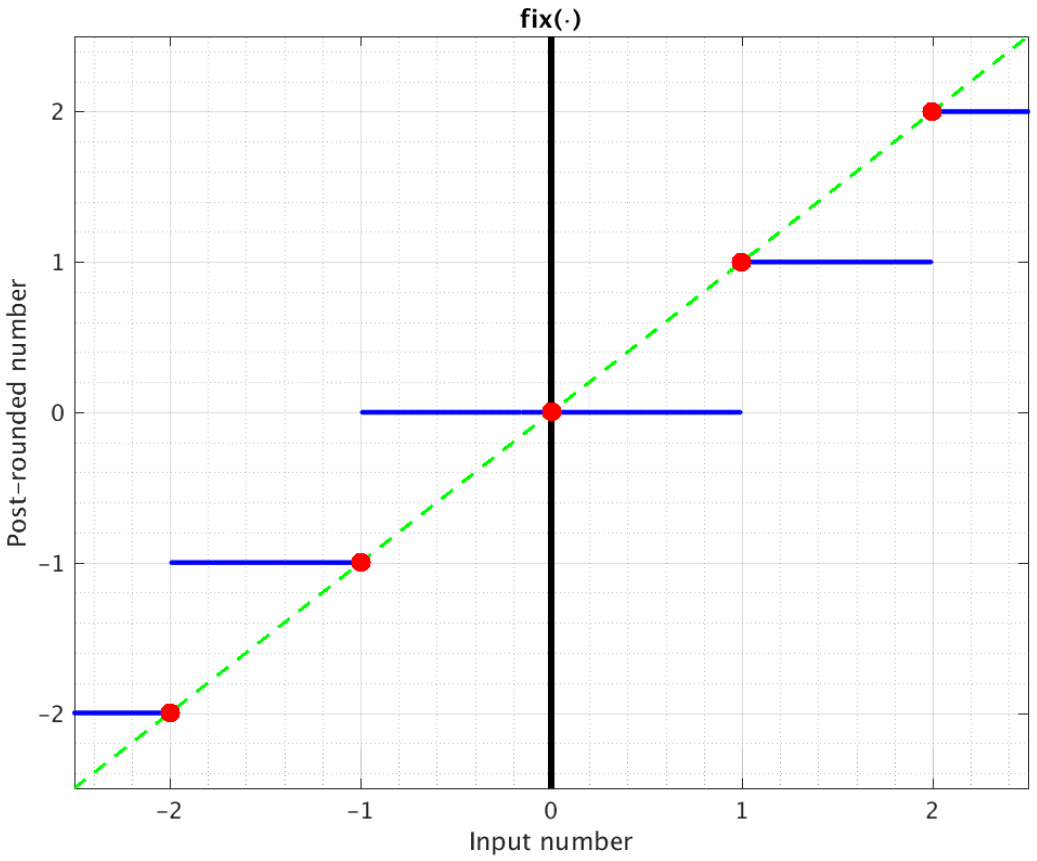
1) matlab round()

- Often the best general-purpose rounding mode
- "Unbiased" rounding
- Symmetric rounding for positive and negative numbers
- Max error ½ LSB



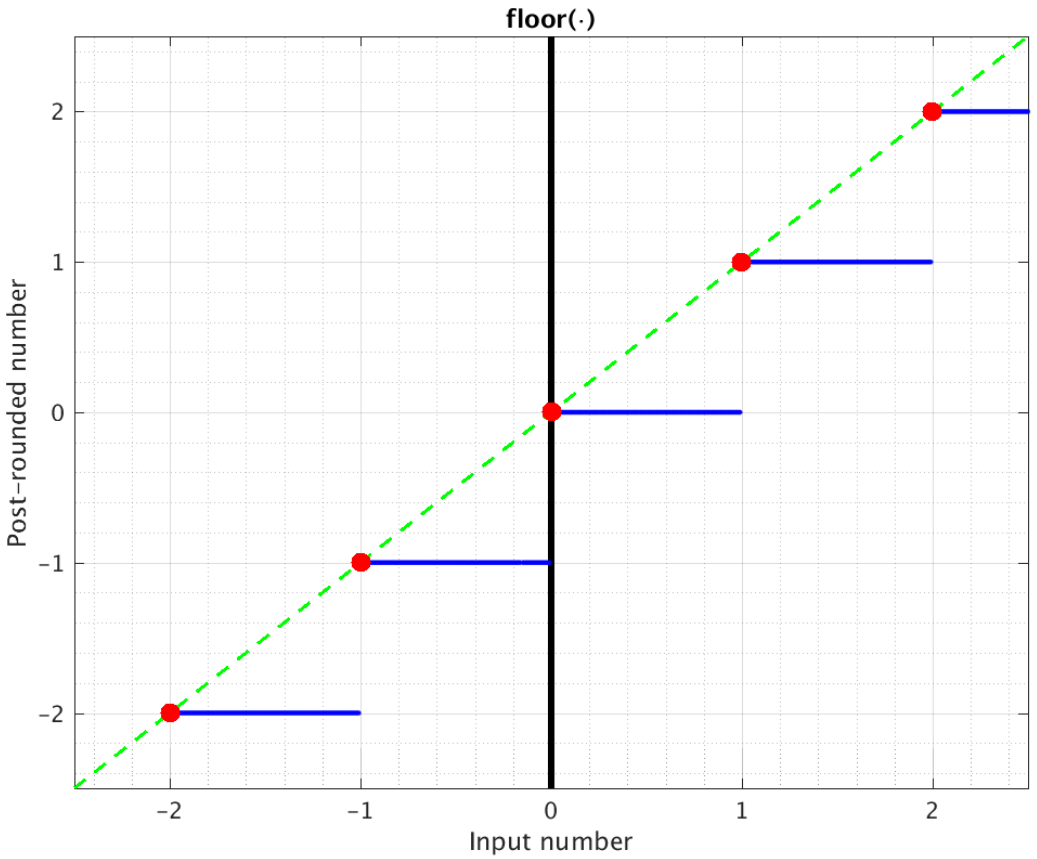
2) matlab fix()

- 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)
  - `xxxxxx` in
    `xxxx--` out
- Max error 1 LSB



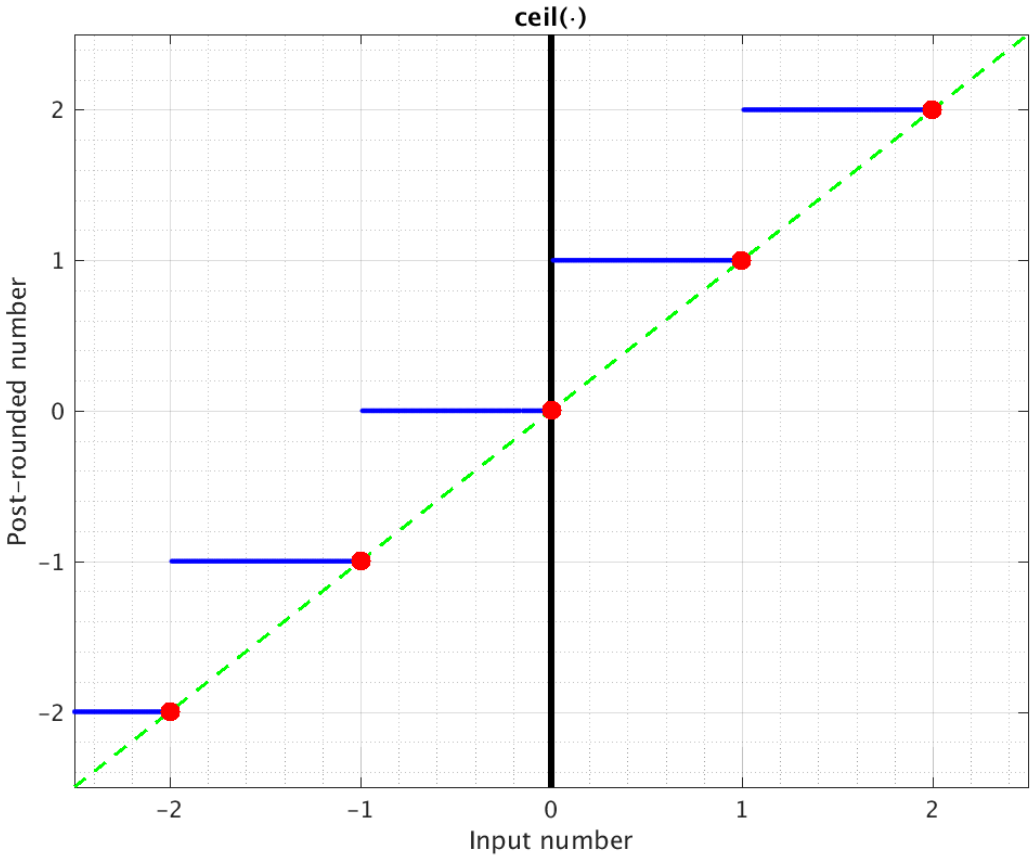
3) matlab floor()

- Numbers rounded down towards –infinity
- Numerical performance is poor
- Very simple hardware for 2's complement (simple truncation)
  - `xxxxxx` in
    `xxxx--` out
- Max error 1 LSB



4) matlab ceil()

- Numbers rounded up toward +infinity
- Numerical performance is poor
- Max error 1 LSB



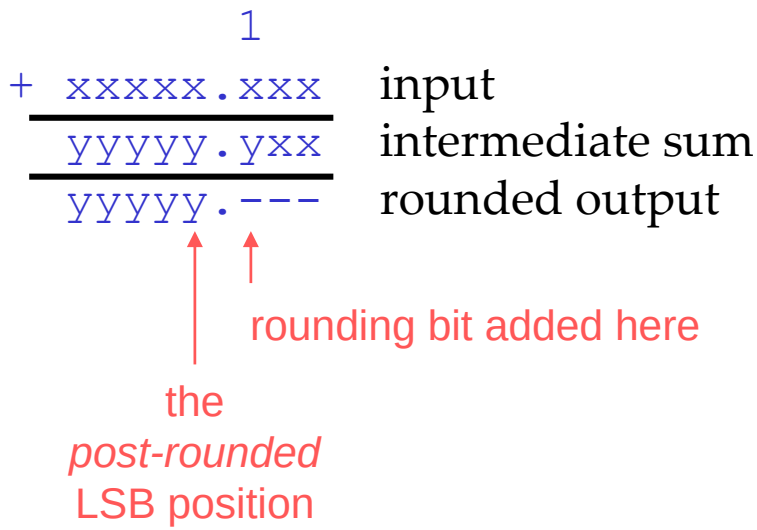
Hardware Rounding: A) Truncation

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 ~1 post-rounded LSB
   - Sign magnitude format numbers (obviously the magnitude portion)
     - Positive and negative numbers both truncate towards zero
     - Same as matlab `fix(•)`
   - 2's complement format numbers
     - All numbers truncate towards negative infinity
     - Same as matlab `floor(•)`
   - Unsigned format numbers
     - All numbers truncate towards zero (negative infinity)
     - Same as matlab `fix(•)` or `floor(•)`

Hardware Rounding: B) Add ½ LSB and Truncate

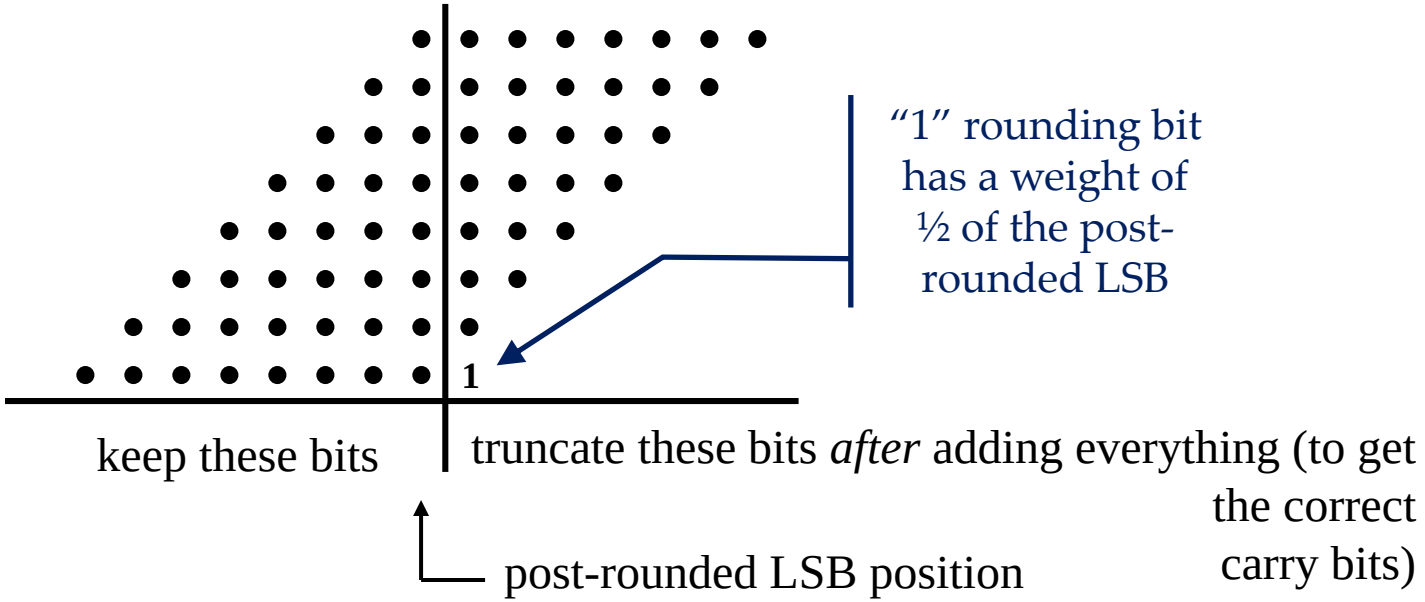
B. Method #5. Add ½ LSB (that is, one half of the LSB of the *output*) and then truncate

- This does not correspond to any of the matlab rounding functions for all binary formats
- Maximum rounding error ½ of the post-rounded LSB



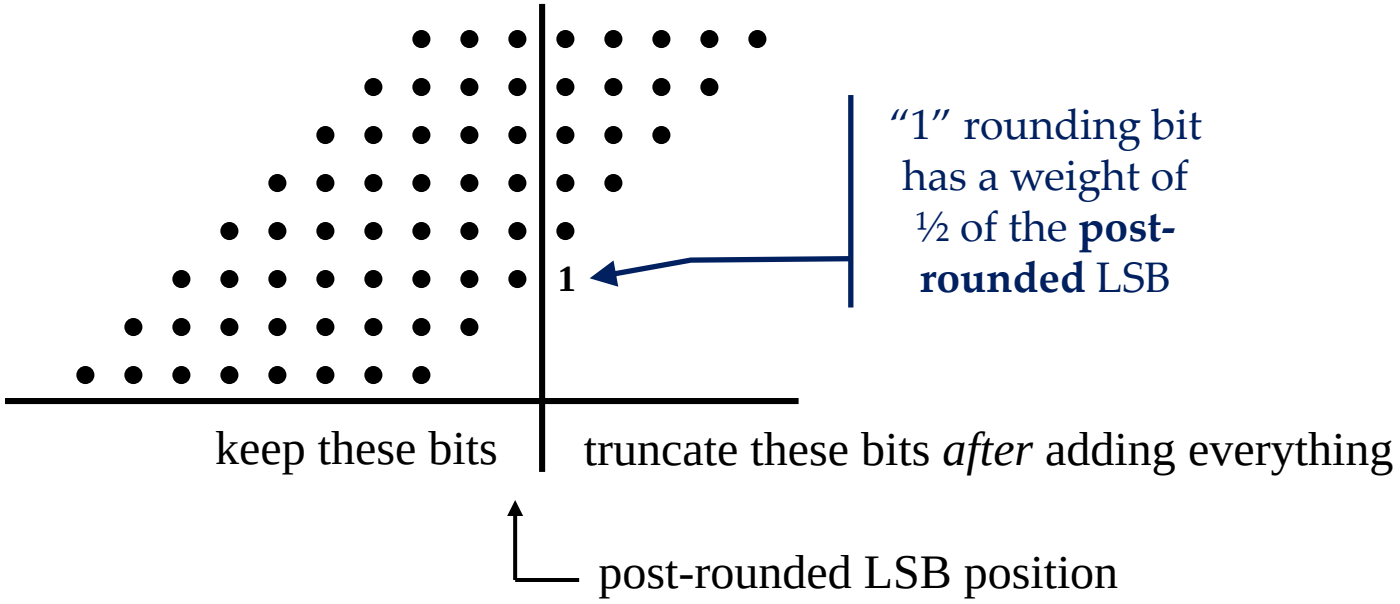
Hardware Rounding: B) Add ½ LSB and Truncate

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



Hardware Rounding: B) Add ½ 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



Hardware Rounding: B) Add ½ LSB and Truncate

- The exact behavior depends on the number format being used:
  - Unsigned
    - Unbiased rounding
  - Magnitude portion of Sign magnitude
    - Unbiased rounding
  - 2's complement
    - Both positive and negative `xxxx.1000` cases round towards positive infinity as explained previously
    - The behavior requires a little more analysis