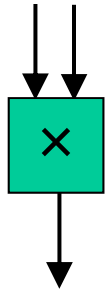
MULTIPLICATION SCALING

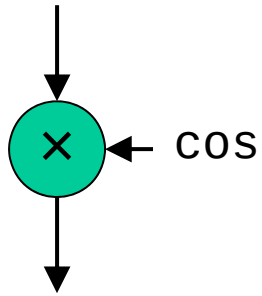
Multiplication Scaling

1. Multiplication of two 2's complement numbers
   - For two *N*-bit inputs, the output word width must be 2*N* bits to avoid overflow or underflow
   - Example:
     - Inputs: 4-bit x 4-bit; each input range `[-8, +7]`
     - Output: product: range `[-56, +64]`
       8-bit: range `[-128, +127]`
       7-bit: range `[-64, +63]`
     - Notice however that the MSB almost completely wasted—there is only case that uses it: `-8 x -8 = +64`
       Or for fractional `1.x` notation: `-1 x -1 = +1.0`
     - There is no way around this unless we can somehow guarantee this one case will never occur
     - Normally it is unacceptable to ignore the MSB and let this case overflow

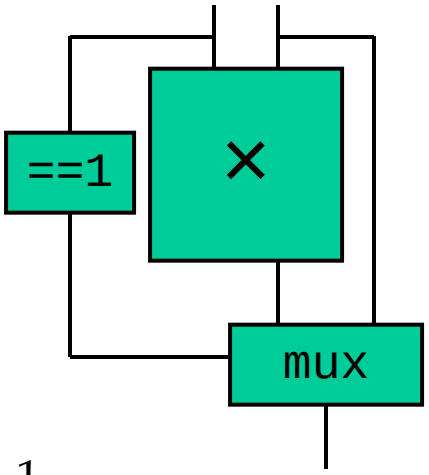


Multiplication by [–1, +1]

2. Assuming fractional numbers, we often want to multiply by a number that ranges from –1.0 to +1.0 What is the best way to encode that input?

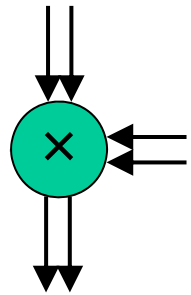


   - 1) Normally, the only way would be to encode it in:
     `2.x` (range `[-2, +1.99]`) format because
     `1.x` (range `[-1, +0.99]`) would overflow
     - Effectively lose almost a bit of precision for the same hardware
   - 2) The only case that causes a problem is `+1.0`
     - This is the most trivial multiplication!
     - Perform `× +1.0` with a mux!
     - Select mux bypass when the one multiplier input is equal to 1.0
     - Input can then be coded in `1.x` format
     - Achieve *N*-1 fractional bits for an *N*-bit multiply
     - Note there may still be a separate issue with –1 × –1

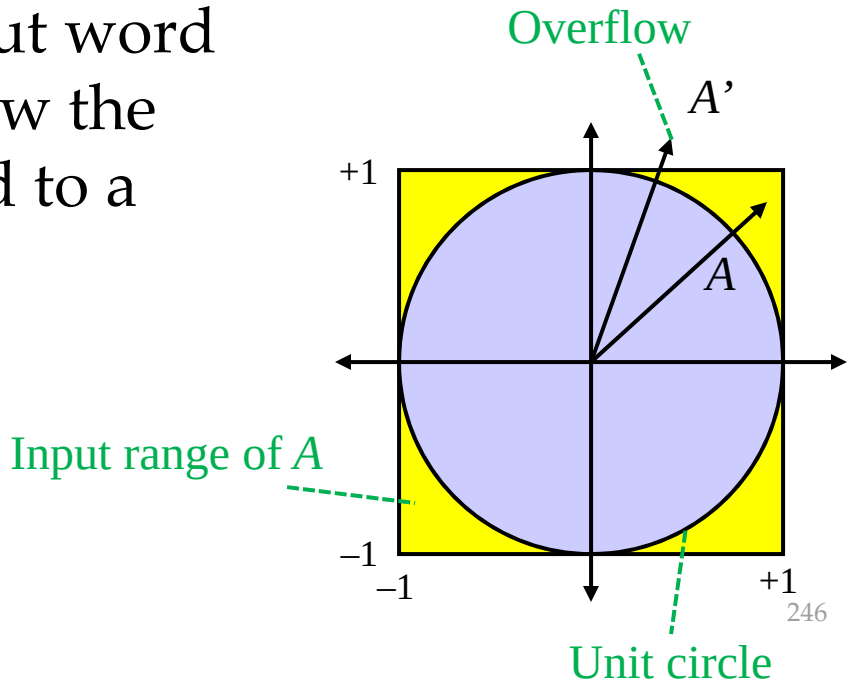


Complex Rotation

3. (complex rectangular-form values) Assuming fractional numbers, we often want to multiply by a complex number with a magnitude of +1.0

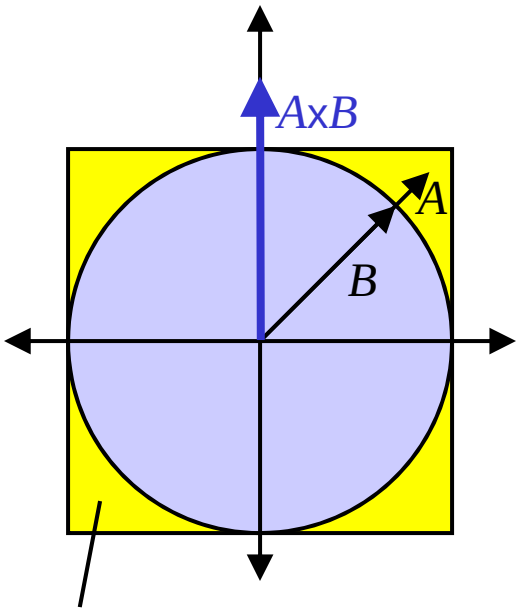


- Even though we multiply by $|B| = 1.0$, we must increase the output word size by 1 bit unless we know the input word (A) is restricted to a magnitude less than 1.0



Complex Multiplication Scaling

```
prawn_57> matlab -nodesktop

                          < M A T L A B (R) >
                Copyright 1984-2009 The MathWorks, Inc.
               Version 7.8.0.347 (R2009a) 32-bit (glnx86)
                           February 12, 2009

>> a = 0.9 + j*0.9
a =
   0.9000 + 0.9000i

>> b= 1/sqrt(2) + j*1/sqrt(2)
b =
   0.7071 + 0.7071i

>> abs(b)       % abs() gives the magnitude of a complex number
ans =
    1.0000

>> a*b
ans =
        0 + 1.2728i

>>
```



Input range of *A*

Complex Multiplication by a Value Whose Components Range [–1, +1]

4) (complex rectangular-form values) Handling multiplication by an input which ranges up to +1.0
   - Similar to non-complex case
   - Two special cases:
     - Multiply by `(+1.0, 0)`
     - Multiply by `(0, +1.0)`
     - `(-1.0, 0)` and `(0, -1.0)` present no special encoding challenges
   - Solutions:
     1) Code (fractional) inputs in `2.x` format
        Effectively wastes almost an entire bit
     2) Use muxes to bypass complex multiplier for the two cases
        Modest extra hardware