Fixed Point Arithmetic

Lecture 3

Number Representations

- Unsigned integers
  - Positive values only
  - All non-zero values are >1
- N-bit unsigned binary number B:
  - $B = b_{N-1} b_{N-2} \dots b_1 b_0$
- Decimal value is calculated as:

$$B = b_{N-1} 2^{\mathrm{N}-1} + \mathrm{b}_{\mathrm{N}-2} 2^{\mathrm{N}-2} + \cdots \mathrm{b}_1 2^1 + \mathrm{b}_0 2^0$$

$$B = \sum_{0}^{N-1} b_i 2_i^i$$

- Hardware for handling arithmetic is very simple
  - In practise, we want to deal with signed and fractional numbers

Negative Numbers

- Signed numbers
  - In hardware, sign denoted by left-most bit
  - MSB is now sign bit
  - “N-1 magnitude bits”
  - Maximum amplitude is halved
- Possible representations:
  - Sign-and-magnitude
  - 1’s complement
  - 2’s complement
  - Offset binary

Poll

- Which signed number representation is de-facto standard for digital hardware?
  1. Sign-and-magnitude
  2. 1’s complement
  3. [✓ 2’s complement]
  4. Offset binary

Negative Numbers

- Sign-and-magnitude
  - Sign bit 0/1 for positive/negative
  - Not useful for hardware arithmetic
- 1’s complement
  - Obtained by inverting all bits including sign bit
  - Drawbacks in arithmetic; end around carry/borrow
  - Two zeros

Negative Numbers

- 2’s complement
  - Invert all bits and add 1
  - Makes subtraction as easy as adding in hardware
  - Single representation of zero
  - Sign bit has weight of -2^{N-1}
- Offset Binary
  - Shift unsigned binary so that 0 now represents some negative number
  - Typically 0 represents -2^{N-1}
  - Also called “excess-k”, where 0 code has value -k
  - Seen in ADCs

Short Question

- We are receiving data from an offset binary ADC, how can we quickly convert this to 2’s complement for our DSP chain?
  1. Flip all bits, add 1
  2. Add 2^{N-1}
  3. Flip all bits
  4. [✓ Flip sign bit]

Two’s Complement Binary Weights

- N-bit **signed** binary number B:
  - $B = b_{N-1} b_{N-2} \ldots b_1 b_0$
- Decimal value is calculated as:

$$B = \boldsymbol{-b_{N-1} 2^{N-1}} + \mathrm{b}_{\mathrm{N-2}} 2^{\mathrm{N-2}} + \cdots \mathrm{b}_1 2^1 + \mathrm{b}_0 2^0$$

$$B = \sum_{0}^{N-1} b_i 2_i^i$$

Number Wheels

- Example of 4-bit number wheels:

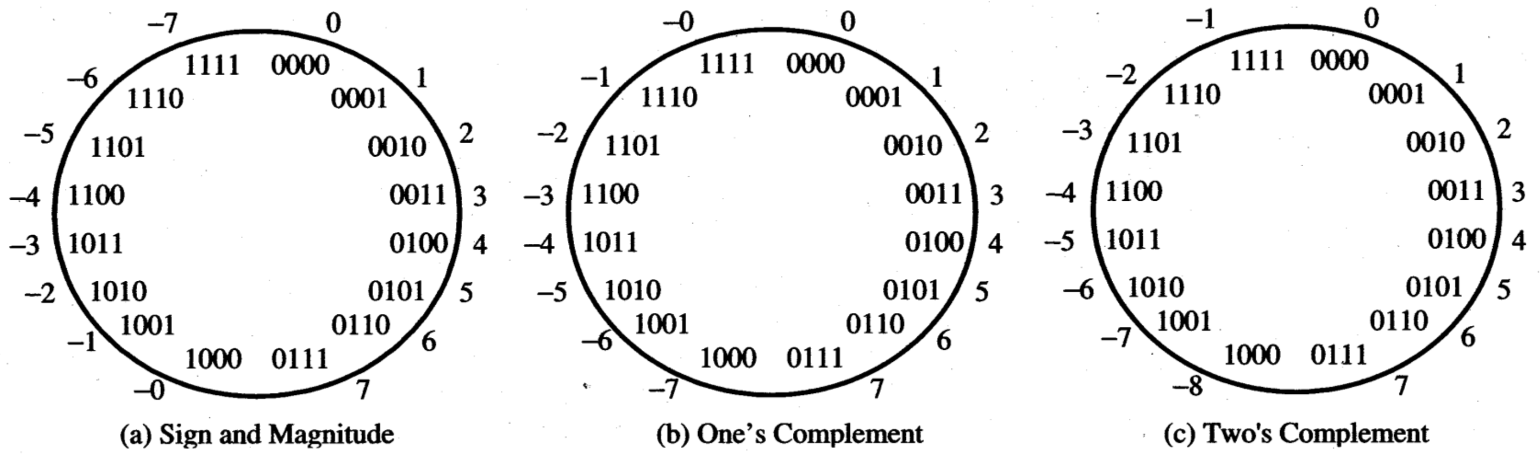


Figure 3.1 Number wheel representation of 4-bit number

- Should be able to quickly draw a 3-bit number wheel
- Should be familiar with range of representable values for N bits

Sign Extension

- How do we increase N bits without modifying the value?
- In unsigned binary, 3’d7 = 3’b111 = 4’b0111 = 4’d7
  - What about signed binary?
  - -3’sd1 = 3’sb111…
  - -4’sd1 = ?
- **POLL: What is -4’sd1:**

1. [✓ 1111]
2. 0001
3. 1110

Sign Extension

- How do we increase N bits without modifying the value?
- In unsigned binary, 3’d7 = 3’b111 = 4’b0111 = 4’d7
  - What about signed binary?
  - -3’sd1 = 3’sb111…
  - -4’sd1 = ?
- Will Verilog take care of this for us?
  - Yes, provided we code it very, very… **VERY** carefully
  - Need to be familiar with how Verilog handles signed types

Signed Arithmetic in Verilog

- `signed` keyword for declaring two's complement wire or reg
- Arithmetic operation result is only signed if ALL operands are signed
- `$signed`, `$unsigned` casting operators
- Concatenation and part-select result in unsigned
- Constants must use s; 3'sd3 not 3'd3
- Bit-shift for signed type, >>>
- Incorrect usage won't cause an error…

What About Fractions?

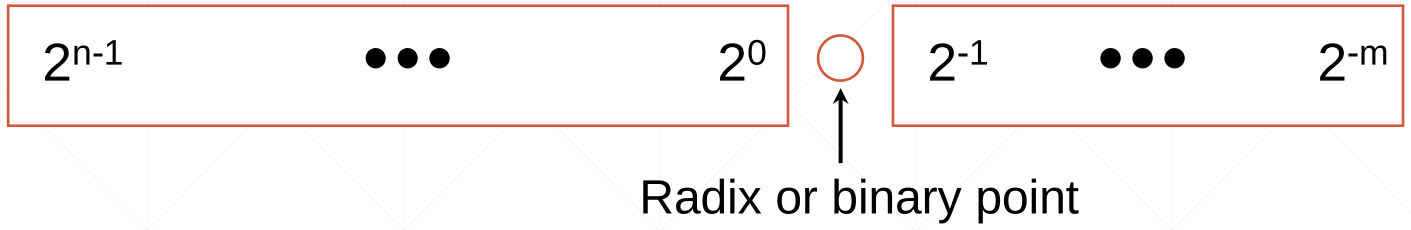
- Unsigned N-bit number, represent values from 0 to 2^{N-1}
- More general case, where $N = n + m$
  - $B = b_{n-1}b_{n-2} \dots b_1 b_0 b_{-1} \dots b_{-m+1} b_{-m}$
- Where value is:

$$B = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_{-m+1}2^{-m+1} + b_{-m}2^{-m}$$

$$B = \sum_{i=-m}^{n-1} b_i 2^i$$

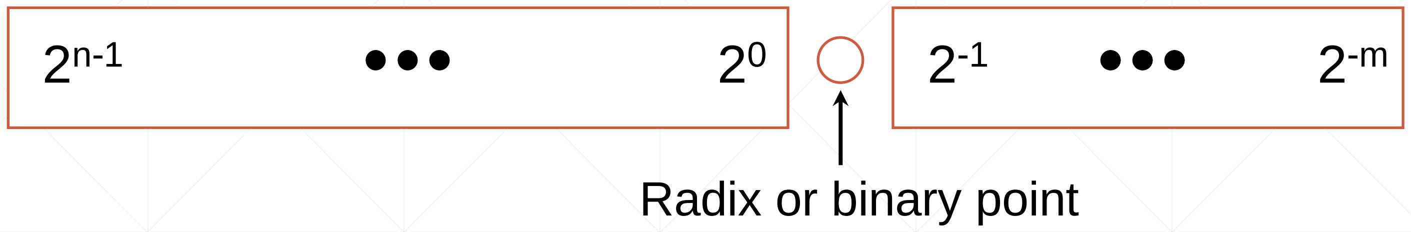
- Fractional parts 2^{-1}, 2^{-2} etc or ½ , ¼

Fractions



- Binary point separates magnitude bits with weight < 1
- Fixed point vs floating point
  - “Fixed” aspect of fixed point?
- How do we handle this in hardware?

Fractions



- Many different notations used…
- UQp.q or Qp.q gives number of **magnitude** bits to left (p) and right (q) of fixed point
  - Implicit sign bit (for Qp.q, N = p+q+1)
  - Terminology separates magnitude & sign bits, but remember sign bit still has weight
- u(N.m) or s(N.m) gives word length (N) and fractional length (m), explicit sign bit
- Here N=p+q unsigned or N = p+q+1 signed, m=q

Fractions

Format1	Format2	Min	Max	Resolution	Magnitude bits ≥1, p	Magnitude bits <1, q	N (total bits)
Q15.16	s(32,16)	-2^{15}	$2^{15} - 2^{-16}$	2^{-16}	15	16	32
UQ15.16	u(31,16)	0	$2^{15} - 2^{-16}$	2^{-16}	15	16	31

Dealing with Fractions

- Binary point separates integer part from fractional part
- User has to keep track of where the radix is
  - No Verilog or Hardware concept of fixed point
- Be aware that it can move around after arithmetic operations!
- Aim to maintain the best overall dynamic range
- Deal appropriately with overflow
  - Additional circuitry, must be detected
  - Rounding & Saturation, must understand impact on quality of data

Fixed Point Operations

- Addition
  - Add two M-bit numbers, sum can be M+1 bits
- Multiplication
  - Multiply N-bit number by an M-bit number, result can be N+M bits
- Must allow for expansion in bit-width
  - Choose word widths to avoid overflow
- Can’t keep growing…
  - Round results
  - Reduces word width without losing data

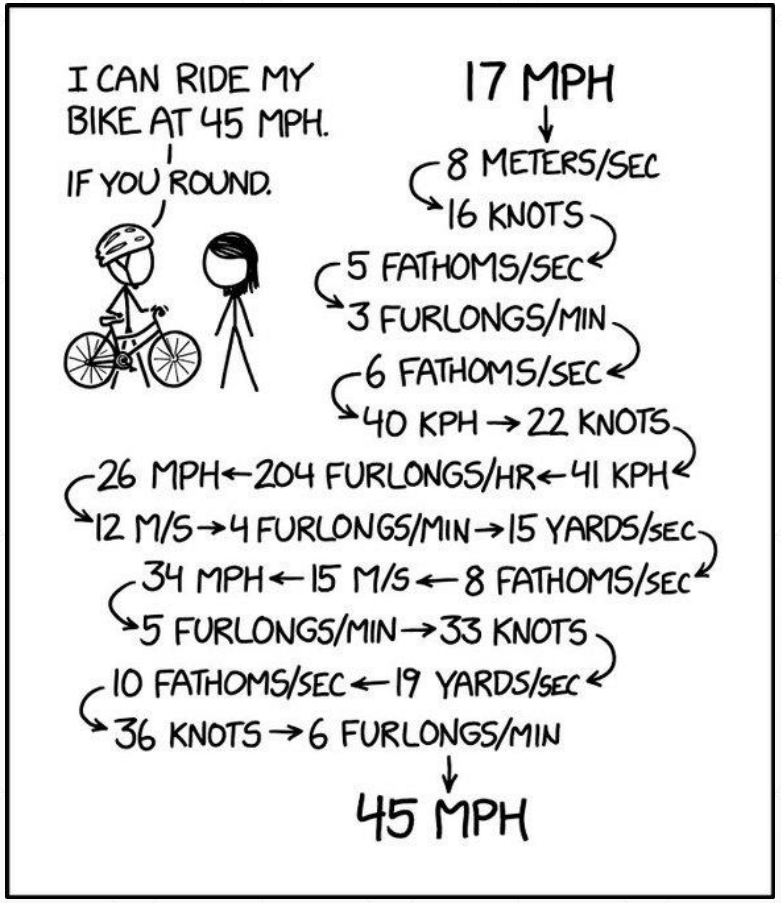
Impact on Two’s Complement System

- Truncation
  - Hardware required?
  - Impact on data?

Impact on Two’s Complement System

- Rounding
  - Hardware required?
  - Impact on data using “normal” rounding?
  - Impact on data using convergent rounding?
    - e.g. half-even

The problem with rounding


I CAN RIDE MY BIKE AT 45 MPH.
IF YOU ROUND.
17 MPH
8 METERS/SEC
16 KNOTS
5 FATHOMS/SEC
3 FURLONGS/MIN
6 FATHOMS/SEC
40 KPH → 22 KNOTS
26 MPH ← 204 FURLONGS/HR ← 41 KPH
12 M/S → 4 FURLONGS/MIN → 15 YARDS/SEC
34 MPH ← 15 M/S ← 8 FATHOMS/SEC
5 FURLONGS/MIN → 33 KNOTS
10 FATHOMS/SEC ← 19 YARDS/SEC
36 KNOTS → 6 FURLONGS/MIN
45 MPH

Impact on Two’s Complement System

- Saturation or Wrap
  - Hardware required?
  - Impact on data?

Practise time…

- Convert
  - 1001 to decimal
  - 1001 to decimal if format is UQ(2.2)
  - 1010 to decimal if format is u(4,2)
  - 1110 to decimal if format is u(4,3)
  - 101.001 to decimal
- Convert these decimal numbers to suitable 8-bit binary representation:
  - 0.43359375
  - 1.43359375
  - -1.43359375

Practise time…

- Represent negative numbers with fractional part in two’s complement format
- Work through multiplication examples and observe the bit growth figures are correct
- In various arithmetic examples, observe the movement of the fixed point. This must be considered in hardware