Verilog Review and Fixed Point Arithmetics

CSE4210 Winter 2012

Mokhtar Aboelaze
based on slides by Dr. Shoab A. Khan

YORK UNIVERSITY

Overview

- Floating and Fixed Point Arithmetic
- System Design Flow
  - Requirements and Specifications (R&S)
  - Algorithmic Development in Matlab and Coding Guidelines
- 2's Complement Arithmetic
- Floating Point Format
- Qn.m format for Fixed Point Arithmetic
- Addition, Multiplication and Scaling in Qn.m
- LTI systems and implementation in Qn.m format

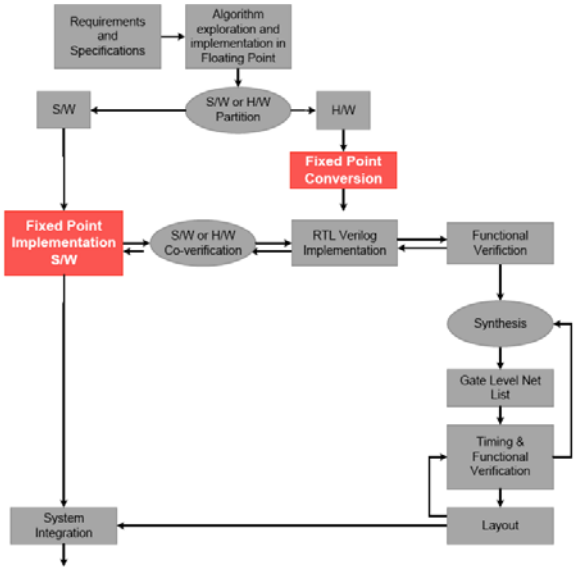
YORK UNIVERSITY

Floating & Fixed Point Arithmetic

- Two Types of arithmetic
  - **Floating Point Arithmetic**
    - After each arithmetic operation numbers are normalized
    - Used where precision and dynamic range are important
    - Most algorithms are developed in FP
      - Ease of coding
    - More Cost (Area, Speed, Power)
  - **Fixed Point Arithmetic**
    - Place of decimal is fixed
    - Simpler HW, low power, less silicon
    - Converting FP simulation to Fixed-point simulation is time
    - consuming
    - Multiplication doubles the number of bits
      - NxN multiplier produces 2N bits
    - The code is less readable, need to worry about overflow and scaling issues

YORK UNIVERSITY

System Design Flow and Fixed



System Design Flow

- The requirements and specifications of the application are captured
- The algorithms are then developed in double precision floating point format
  - Matlab or C/C++
- A signal processing system general consists of hybrid target technologies
  - DSPs, FPGAs, ASICs
- For mapping application developed in double precision is partitioned into
  - hardware & software
- Most of signal processing applications are mapped on Fixed-point Digital Signal Processors or HW in ASICs or FPGAs
- The HW and SW components of the application are **converted** into Fixed Point format for this mapping

Requirements & Specifications

•Gathering R&S is the first step of the system design

•System components and algorithms are then selected that meet the requirements

•Example R&S of a UHF Radio are shown

Characteristics	Specifications
Output Power	2W
Spurious Emission	<60 dB
Harmonic Suppression	>55 dB
Frequency Stability	2ppm or better
Reliability	>10,000 hours MTBF <30 min MTTR
Handheld	12V DC nickel metal hybrid, nickel cadmium or lithium-ion battery pack

R&S of a UHF Radio (cont)

Characteristics	Specifications
Frequency Range	420 MHz to 512 MHz
Data rate	Up to 512 kbps multi-channel non-line of sight
Channel	Multi-path with 15 μs delay spread and 220 km/h relative speed between transmitter and receiver
Modulation	OFDM supporting BPSK, QPSK and QAM
FEC	Turbo codes, convolution, Reed–Solomon
Frequency hopping	> 600 hops/s, frequency hopping on full hopping band
Waveforms	Radio works as SDR and should be capable of accepting additional waveforms

Algorithm Development and Mapping

- The R&S related to digital design are forwarded to algorithm developers and system designers
- Algorithms are coded in behavioral modeling tools like Matlab
- The Matlab code is then translated into a high level language, for example, C/C++
- System is designed based on R&S
  - System usually consists of hybrid technologies consisting of ASICs, DSPs, GPP, and FPGAs
- Partitioning of the application into HW/SW parts is performed
- The SW is then developed for the SW part and architectures are designed and implemented for the HW parts
- Integration and testing is performed throughout the design cycle

Guidelines for Matlab Coding

- Signal processing applications are mostly developed in Matlab
- As the Matlab code is to be mapped in HW and SW so adhering to coding guidelines is critical
  - The code must be designed to work for processing of data in chunks
  - The code should be structured in distinct components
    - Well defined interfaces in terms of input and output arguments and internal data storages
  - All variables and constants should be defined in data structures
    - User defined configurations in one structure
    - System design constants in another structure
    - Internal states for each block in another structure
  - Initialization in the start of simulation

Processing in Chunks

```
% BPSK = 1, QPSK = 2, 8PSK = 3, 16QAM = 4
% All-user defined parameters are set in structure USER_PARAMS
USER_PARAMS.MOD_SCH = 2; %select QPSK for current simulation
USER_PARAMS.CHUNK_SZ = 256; %set buffer size
USER_PARAMS.NO_CHUNKS = 100;% set no of chunks for simulation
% generate raw data for simulation
raw_data = randint(1, USER_PARAMS.NO_CHUNKS*USER_PARAMS.CHUNK_SZ)
% Initialize user defined, system defined parameters and states
PARAMS = MOD_Params_Init(USER_PARAMS);
STATES = MOD_States_Init(PARAMS);
mod_out = [];
% Code should be structured to process data on chunk-by-chunk
   basis
for iter = 0:USER_PARAMS.NO_CHUNKS-1
     in_data = raw_data
     (iter*USER_PARAMS.CHUNK_SZ+1:USER_PARAMS.CHUNK_SZ*(iter+1));
     [out_sig,STATES]= Modulator(in_data,PARAMS,STATES);
     mod_out = [mod_out out_sig];
end
```

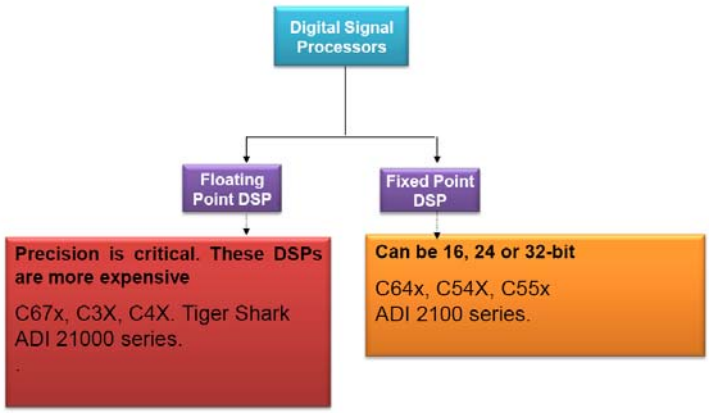
YORK UNIVERSITY

Fixed Point vs. Floating Point HW

- Algorithms are developed in floating point format using tools like Matlab
- Floating point processors and HW are expensive
- Fixed-point processors and HW are used in embedded systems
- After algorithms are designed and tested then they are converted into fixed-point implementation
- The algorithms are ported on Fixed-point processor or application specific hardware

YORK UNIVERSITY

Digital Signal Processors



Number Representation

- In a digital design fixed or floating point numbers are represented in binary format
- Types of Representation
  - one's complement
  - sign magnitude
  - canonic sign digit (CSD)
  - two's complement
- In digital system design for fixed point implementation the canonic sign digit (CSD), and two's complement are normally used

2's Complement

- MSB is the sign bit (has a negative weight)

2^3	2^2	2^1	2^0
1	0	1	1
-8	+0	+2	+1= -5

2^3	2^2	2^1	2^0
0	1	1	0
0	+4	+2	+0 = 6

Floating Point Format

- Floating point arithmetic is appropriate for high precision applications
- Applications that deals with number with wider dynamic range
- A floating point number is represented as

$$x = (-1)^s \times 1 \times m \times 2^{e-b}$$

  - s represents sign of the number
  - m is a fraction number >1 and < 2
  - e is a biased exponent, always positive
  - The bias **b** is subtracted to get the actual exponent

IEEE 754 FP Format (single precision)

Sign

0= +

1= -

8 bit

True exponent=e-b

Bias is 127

23 bit

Mantissa is normalized (MSB=1). There is an implied 1 to make it >1 and <2

$$x = (-1)^s \times 1.m \times 2^{e-127}$$

YORK UNIVERSITY

IEEE 754 FP Format

- SEEEEEEEEMMMMMMMMMMMMMMMMMMMMMMM
- If E=255 and M is nonzero, then V=NaN ("Not a number")
- If E=255 and M is zero and S is 1, then V=- ∞
- If E=255 and M is zero and S is 0, then V= ∞
- If 0<E<255 then V=$(-1)^S$ * $2^{(E-127)}$ * (1.M) where "1.M" is intended to represent the binary number created by prefixing M with an implicit leading 1 and a binary point.
- If E=0 and M is nonzero, then V=(-1)**S * 2 ** (-126) * (0.M) These are "unnormalized" values.
- If E=0 and M is zero and S is 1, then V=-0
- If E=0 and M is zero and S is 0, then V=0

YORK UNIVERSITY

Examples

- 0 00000000 00000000000000000000000 = 0
- 1 00000000 00000000000000000000000 = -0

- 0 11111111 00000000000000000000000 = Infinity
- 1 11111111 00000000000000000000000 = -Infinity

- 0 11111111 00000100000000000000000 = NaN
- 1 11111111 00100010001001010101010 = NaN

- 0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
- 0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
- 1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5
- 0 00000001 00000000000000000000000 = +1 * 2**(1-127) * 1.0 = 2**(-126)
- 0 00000000 10000000000000000000000 = +1 * 2**(-126) * 0.1 = 2**(-127)

- 0 00000000 00000000000000000000001 = +1 * 2**(-126) * 0.00000000000000000000001 = 2**(-149) (Smallest positive value)

IEEE 754 Double Precision

- 64 bit number
- 1 bit for sign
- 11 bits for exponent
- 52 bits for mantissa
- Bias is 1023

FP Addition

- **S0:** Append the implied 1 of the mantissa
- **S1:** Shift the mantissa from S0 with smaller exponent es to the right by $e_l - e_s$, where el is the larger of the two exponents
- **S2:** For negative operand take two's complement and then add the two mantissas.
  - If the result is negative, again takes two's complement of the result for storing in IEEE format
- **S3:** Normalize the sum back to IEEE format by adjusting the mantissa and appropriately changing the value of the exponent e_l
- **S4:** Round or truncate the resultant mantissa to fit in IEEE format

FP Addition -- Example

- Add these 2 numbers (e=4, m=5, bias=7)

0_1010_00101 = $2^{10-7} \times 1.00101 = 1001.01 = 9.25$

0_1001_00101 = $2^{9-7} \times 1.00101 = 1.00101 = 4.625$

Align $1.00101 \times 2^3 \Rightarrow 1.00101 \times 2^3$

$1.00101 \times 2^2 \Rightarrow \underline{0.100101 \times 2^3}$

$1.101111 \times 2^3 = 1.734375 \times 8 = 13.875$

In FP format 0_1010_10111 = $2^{10-7} \times 1.7185 = 13.75$

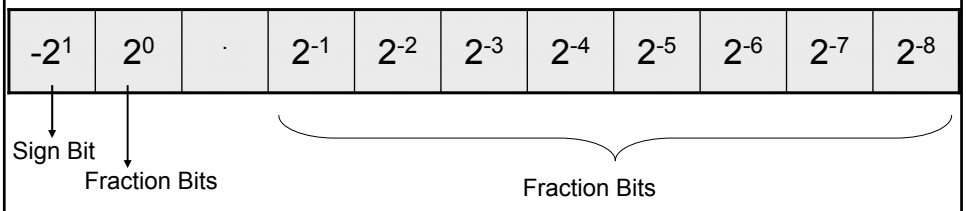
FP Multiplication

- **S0:** Add the two exponents e1 and e2 and subtract the bias once
- **S1:** Multiply the mantissas as unsigned numbers to get the product, and XOR the two sign bits to get the sign of the product
- **S2:** Normalize the product if required
- **S3:** Round or truncate the mantissa

YORK UNIVERSITY

Qn.m Format for Fixed-Point Arithmetic

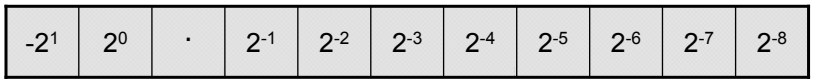
- Qn.m format is a fixed positional number system for representing floating point numbers
- A Qn.m format binary number assumes n bits to the left and m bits to the right of the binary point



Q2.8 Format

YORK UNIVERSITY

Example



01 1101 0000

$1 + 1/2^1 + 1/2^2 + 1/2^4 = 1.8125$

The range of a Q2.8 is -2 (10_000_000) to +1.99609375 (01_1111_1111)

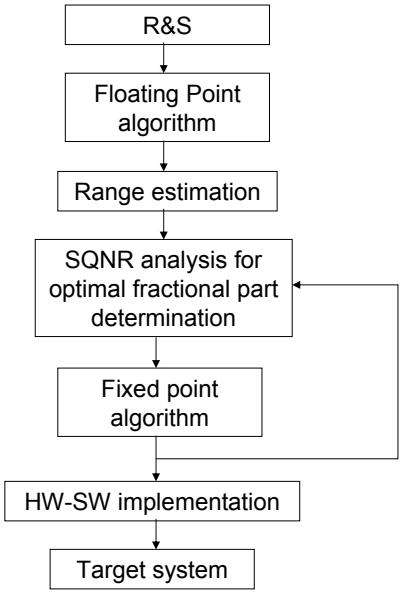
In Qn.m n determines the range of the integer, while m determines the precision of the fractional part

Fixed Point DSPs

- Commercially available off the shelf processors usually have16 bits to represent a number
- In C, a programmer can define 8, 16 or 32 bit numbers as char, short and int/long respectively
- In case a variable requires different precision than what can be defined in C, the number is defined in higher precision
- For example an 18-bit number should be defined as a 32-bits integer
- High precision arithmetic is performed using low precision arithmetic operations
  - 32-bit addition requires two 16-bit addition
  - 32-bit multiplication requires four 16-bit multiplications

FP to Fixed Point Comversion

- Start wit the algorithm (FP)
- Estimate the range
- Determine n
- The fractional part m requires detailed analysis of signal to quantization noise



Addition in Q Format

- Adding two numbers $Qn_1.m_1$ and $Qn_2.m_2$ results in Qn.m where
- $N=\max(n_1,n_2)+1$, $m=\max(m_1,m_2)$
- Note that the decimal point does not exist in H/W, the designer must align the two number properly

YORK UNIVERSITY

Multiplication in Q Format

- Unsigned by unsigned
- Straightforward shift and add, no sign extension in general $Q(n_1+n_2).(m_1+m_2)$

1 1 0 1 = 11.01 in Q2.2 = 3.25

1 0 1 1 = 10.11 in Q2.2 = 2.75

```
  1 1 0 1
1 1 0 1
0 0 0 0
1 1 0 1
```

1 0 0 0 1 1 1 1 = 1000.1111 in Q4.4 format = 8.9375

YORK UNIVERSITY

Multiplication in Q Format

- Signed by Unsigned

Sign extended number

1 1 0 1 = 11.01 in Q2.2 = -0.75 (signed)

0 1 0 1 = 01.01 in Q2.2 = 1.25 (unsigned

```
1 1 1 1 1 1 0 1
0 0 0 0 0 0 0 0
1 1 1 1 0 1
0 0 0 0 0
```

1 1 1 1 1 0 0 0 1 = 1111.0001 in Q4.4 format = -0.9375

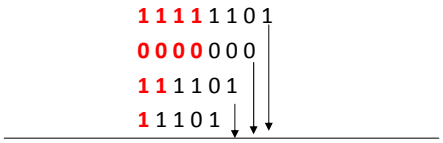
YORK UNIVERSITY

Multiplication in Q Format

- Signed by Unsigned

1 1 0 1 = 11.01 in Q2.2 = -0.75 (signed)

1 1 0 1 = 01.01 in Q2.2 = 3.25 (unsigned



10 1 1 0 1 1 0 0 1 = 1101.1001 in Q4.4 format =

Take 2's complement 0010.0111 =-2.4375

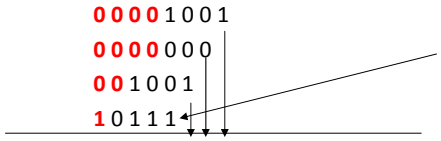
YORK UNIVERSITY

Multiplication in Q Format

- Unsigned by Signed

1 0 0 1 = 11.01 in Q2.2 = 2.25 (unsigned)

1 1 0 1 = 01.01 in Q2.2 = -0.75 (signed)



The 2's complement of the multiplicand 01001=10111

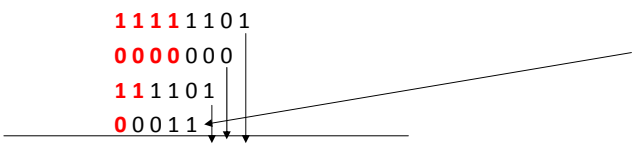
1 1 1 0 0 1 0 1 = 1110.0101 in Q4.4 format take 2's complement it become 0001.1011 = -1.6875

Multiplication in Q Format

- Signed by Signed

1 1 0 1 = 11.01 in Q2.2 = -0.75 (signed)

1 1 0 1 = 1.101 in Q1.3 = -0.375 (signed)



The 2's complement of 11.01 since we are multiplying it by the sign bit

10 0 0 0 0 1 0 0 1 = 00.01001 in Q2.5 format = 0.28125

Note that there is an extra sign bit so we need 2.5 not 3.5 format

YORK UNIVERSITY