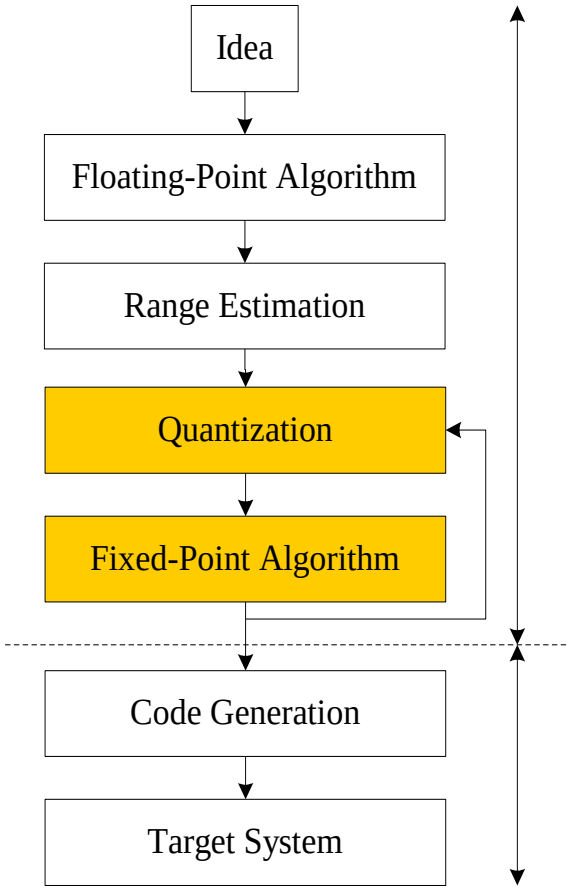
Fixed-point design

Overview

- **Introduction**
- Numeric representation
- Simulation methods for floating to fixed point conversion
- Analytical methods

Fixed-Point Design

- Digital signal processing algorithms
  - Often developed in floating point
  - Later mapped into fixed point for digital hardware realization
- Fixed-point digital hardware
  - Lower area
  - Lower power
  - Lower per unit production cost

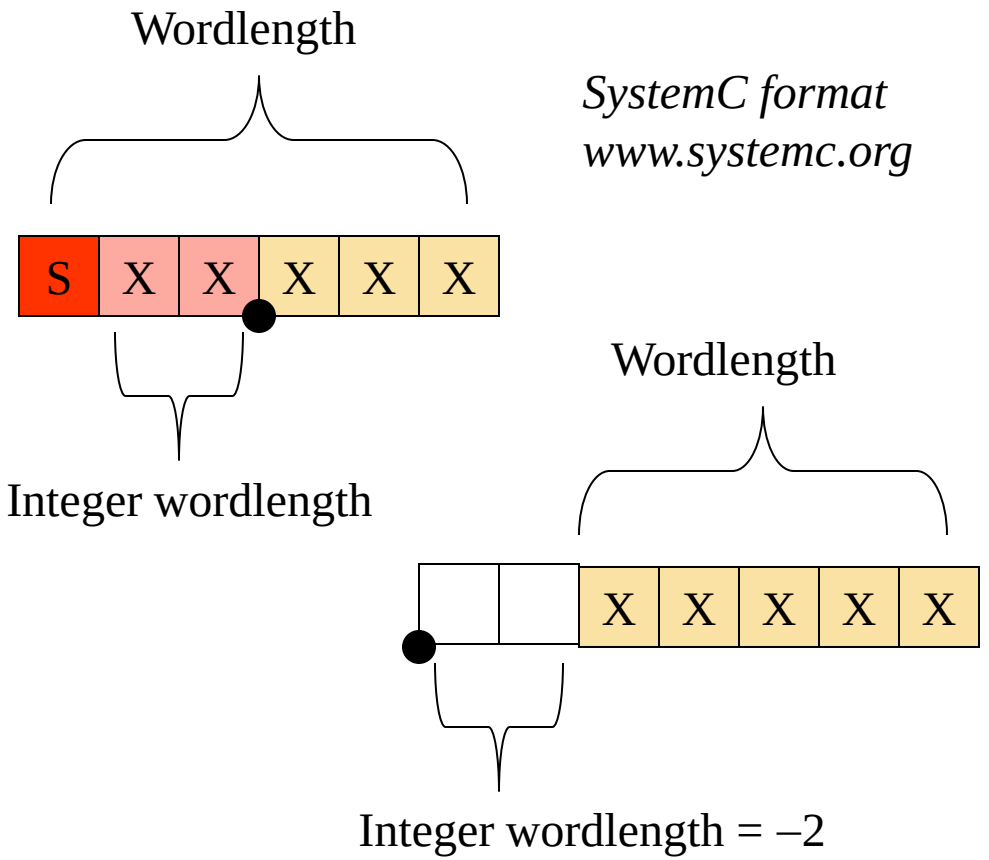


Fixed-Point Design

- Float-to-fixed point conversion required to target
  - ASIC and fixed-point digital signal processor core
  - FPGA and fixed-point microprocessor core
- All variables have to be annotated manually
  - Avoid overflow
  - Minimize quantization effects
  - Find *optimum wordlength*
- Manual process supported by simulation
  - Time-consuming
  - Error prone

Fixed-Point Representation

- Fixed point type
  - Wordlength
  - Integer wordlength
- Quantization modes
  - Round
  - Truncation
- Overflow modes
  - Saturation
  - Saturation to zero
  - Wrap-around



SystemC format
www.systemc.org

Back

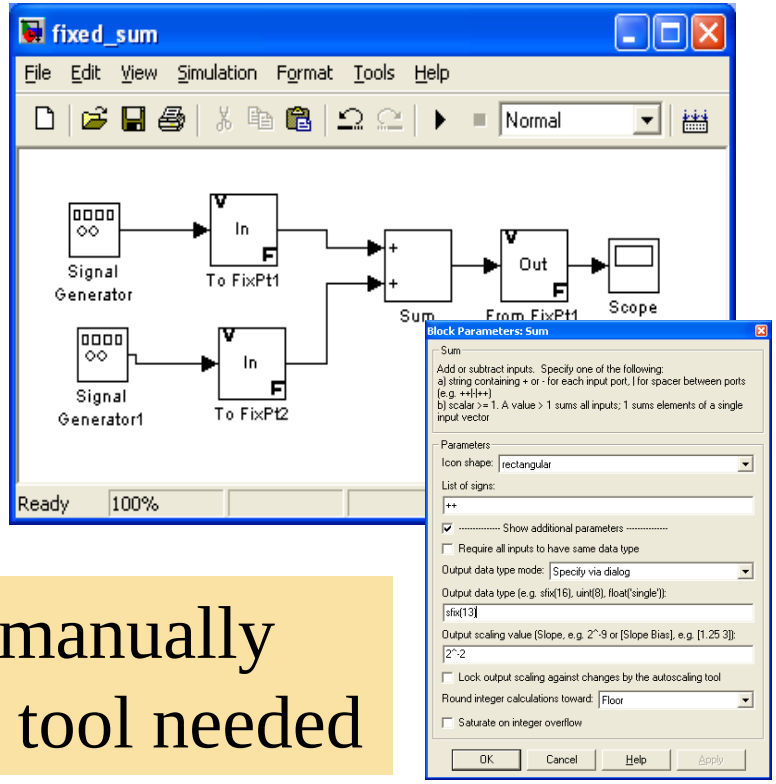
Tools for Fixed-Point Simulation

- gFix (Seoul National University)
  - Using C++, operator overloading
- Simulink (Mathworks)
  - Fixed-point block set 4.0
- SPW (Cadence)
  - Hardware design system
- CoCentric (Synopsys)
  - Fixed-point designer

```
float a;
float b;
float c;
c = a + b;
```

→

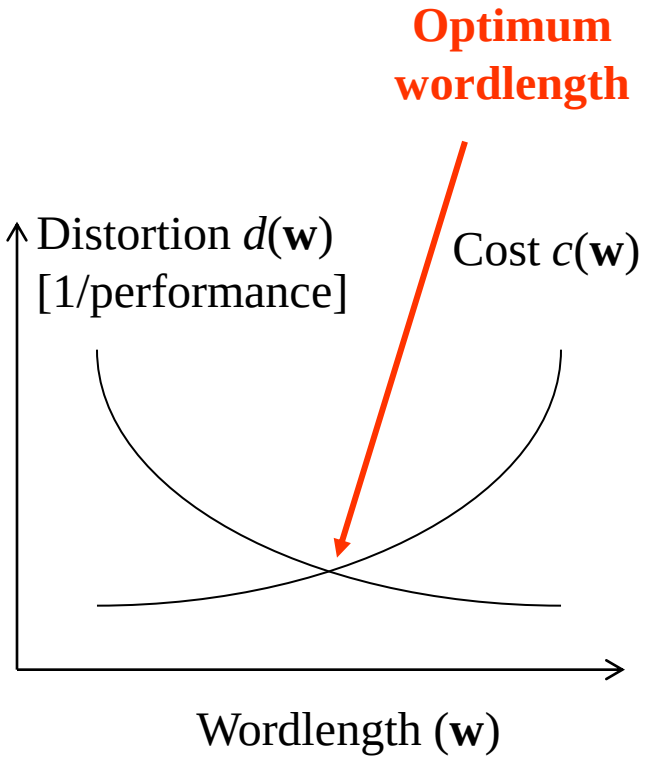
```
gFix a(12,1);
gFix b(12,1);
gFix c(13,2);
c = a + b;
```



Wordlengths determined manually
Wordlength optimization tool needed

Optimum Wordlength

- Longer wordlength
  - May improve application performance
  - Increases hardware cost
- Shorter wordlength
  - May increase quantization errors and overflows
  - Reduces hardware cost
- Optimum wordlength
  - Maximize application performance or minimize quantization error
  - Minimize hardware cost



Wordlength Optimization Approach

- Analytical approach
  - Quantization error model
  - For feedback systems, instability and limit cycles can occur
  - Difficult to develop analytical quantization error model of adaptive or non-linear systems
- Simulation-based approach
  - Wordlengths chosen while observing error criteria
  - Repeated until wordlengths converge
  - Long simulation time

Overview

- Introduction
- **Numeric representation**
- Simulation methods for floating to fixed point conversion
- Analytical methods

Number representation

Matlab examples

- Numeric circle
- fi Basics
- fi Binary Point Scaling

Fi type

- **Integer arithmetic with a fixed number of fractional digits**

```
>> a=fi(pi, true, 8, 5);
>> bin(a)
0   1   1 .  0   0   1   0    1
s   2   1 . 1/2 1/4 1/8 1/16 1/32

>> double(a)
3.15625
```

Fi object

```
>> a = fi(pi)

a =

                3.1416015625

          DataTypeMode: Fixed-point: binary point scaling
                Signed: true
            WordLength: 16
        FractionLength: 13

             RoundMode: nearest
          OverflowMode: saturate
           ProductMode: FullPrecision
  MaxProductWordLength: 128
               SumMode: FullPrecision
      MaxSumWordLength: 128
         CastBeforeSum: true
```

data

numerictype

fimath

Fi Object

- Notation
- Multiplication
- Multiplication with KeepMSB Mode
- Addition
- Addition with KeepLsb Mode
- Numerictype
- fimath

Overview

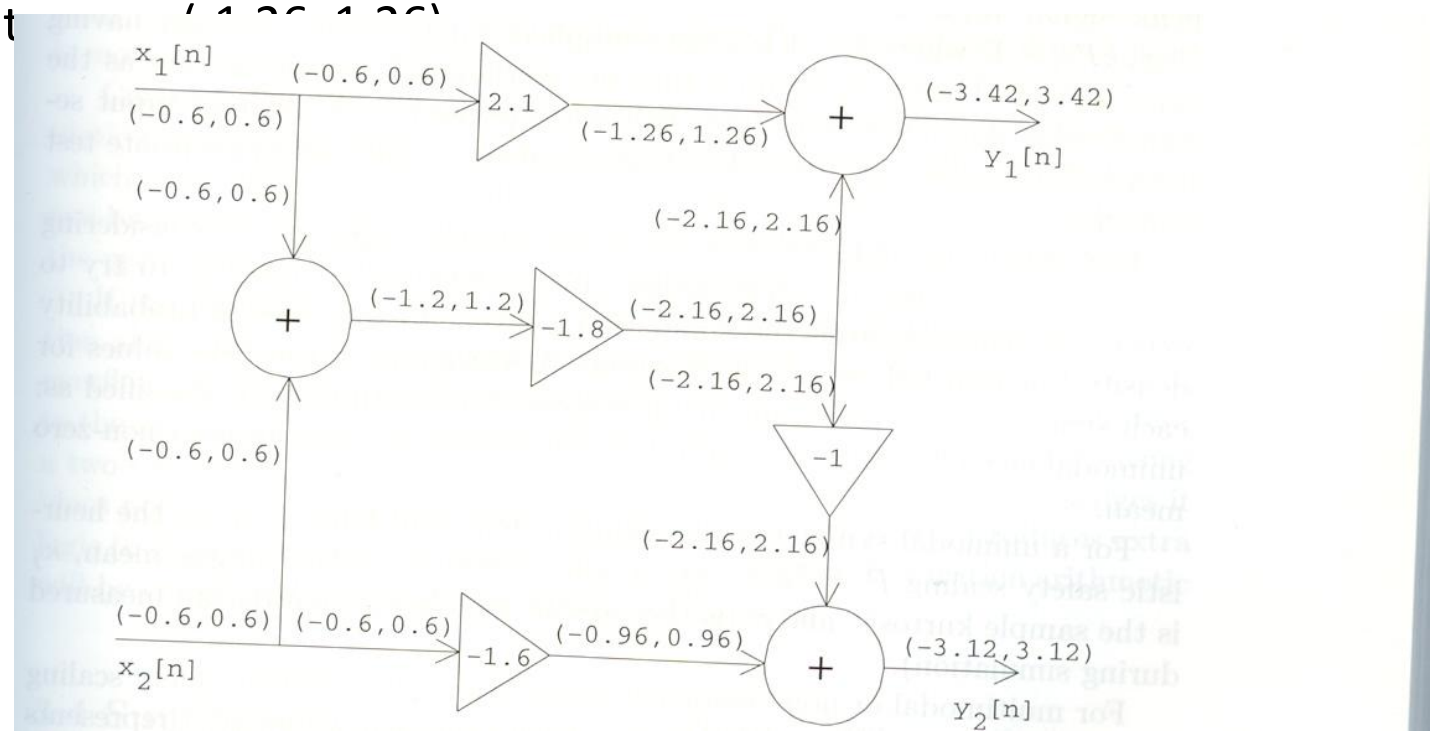
- Introduction
- Numeric representation
- **Simulation methods for floating to fixed point conversion**
- Analytical methods

Data-range propagation

y1=2.1x1-1.8(x1+x2)=0.3x1-1.8x2

Input range: (-0.6 0.6)

Output



Data-range propagation

Disadvantages

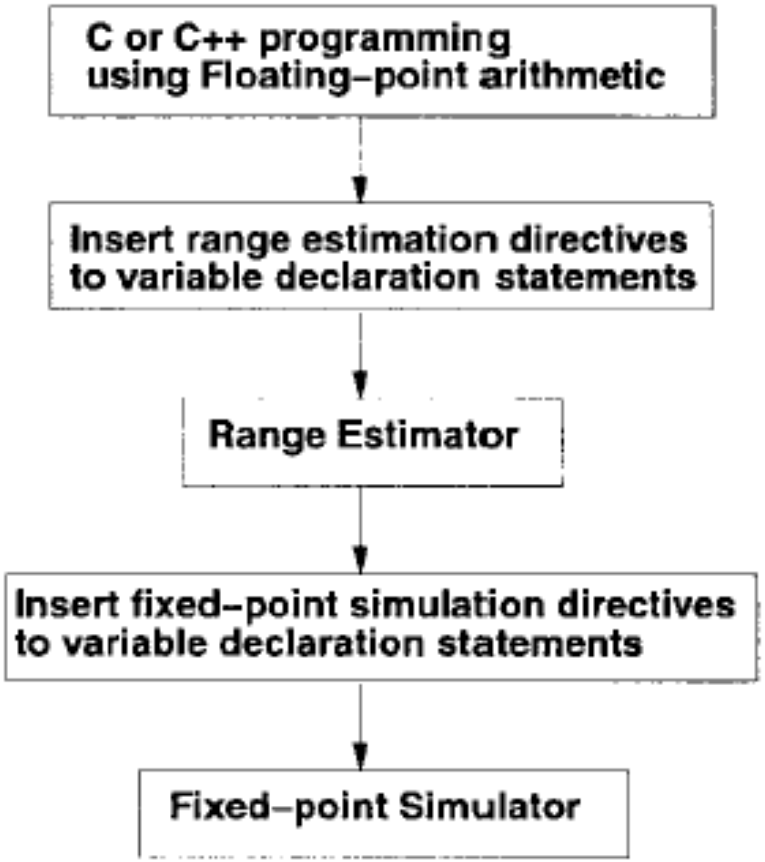
- Provide larger bounds on signal values than necessary

Solution

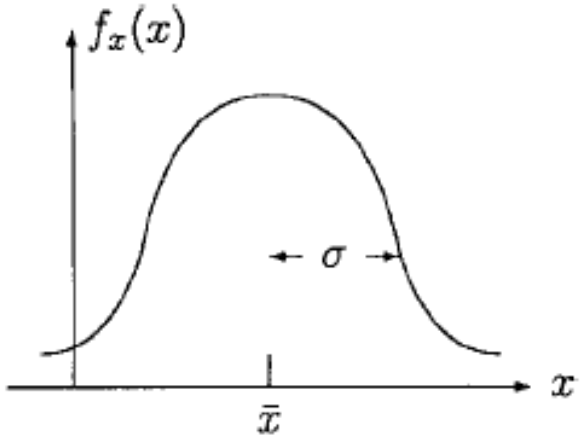
- Simulation-based range estimation

Development of fixed point programs

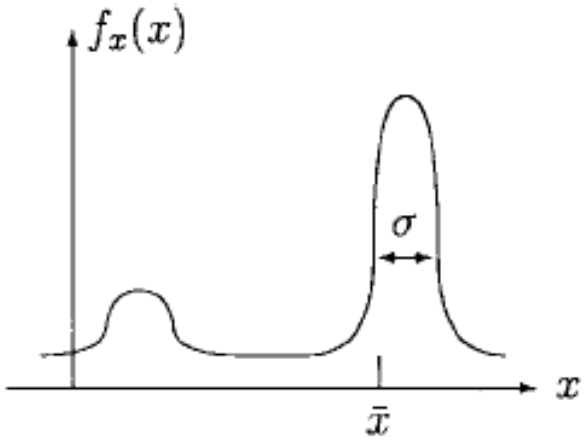
- Toolbox gF



Statistical characteristics of input signals



(a)



$$R = |\mu| + n \times \sigma, \qquad n \propto k.$$

$$R = \hat{R}_{99.9\%} + g$$

Implementation – range estimation

```
void
iir1(short argc, char *argv[])
{
  float Xin;
  float Yout;   // fSig()
  float Ydly;   // fSig()
  float Coeff;

  Coeff = 0.9;
  Ydly = 0.;
  for( i = 0; i < 1000; i++ ) {
    infile >> Xin ;
    Yout = Coeff * Ydly + Xin ;
    Ydly = Yout ;
    outfile << Yout << '\n';
  }
}
```

(a)

```
void
iir1(short argc, char *argv[])
{
  float Xin;
  static fSig   Yout("iir1/Yout");
  static fSig   Ydly("iir1/Ydly");
  float Coeff;

  Coeff = 0.9;
  Ydly = 0.;
  for( i = 0; i < 1000; i++ ) {
    infile >> Xin ;
    Yout = Coeff * Ydly + Xin ;
    Ydly = Yout ;
    outfile << Yout << '\n';
  }
}
```

(b)

Implementation – range estimation

```
class fSig
{
private:
  ......
  double        Data;
  double        Sum;
  double        Sum2;
  double        Sum3;
  double        Sum4;
  double        AMax;
  long          SumC;
  ......

public:

  ......
  fSig& operator = (const fSig&);
  fSig& operator = (double);

  friend double operator + (const fSig&, const fSig&);
  friend double operator + (const fSig&, double);
  friend double operator - (const fSig&, const fSig&);
  friend double operator - (const fSig&, double);
  friend double operator * (const fSig&, const fSig&);
  friend double operator * (const fSig&, double);
  friend double operator / (const fSig&, const fSig&);
  friend double operator / (const fSig&, double);

  friend short operator == (const fSig&, const fSig&);
  friend short operator == (const fSig&, double);
  friend short operator != (const fSig&, const fSig&);
  friend short operator != (const fSig&, double);
  friend short operator > (const fSig&, const fSig&);
  friend short operator > (const fSig&, double);
  friend short operator < (const fSig&, const fSig&);
  friend short operator < (const fSig&, double);
  ......

};
```

References

1. Marc Moonen, Lecture 4 : Filter implementation, lecture slides.
2. Kyungtae Han, ``Fixed-Point Wordlength Optimization and Its Applications to Broadband Wireless Demodulator Design,'' Samsung Advanced Institute of Technology, Korea, Jun 24, 2004