ETISB
Embedded Signal Processing

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Indhold

Introduction, number formats and Blackfin

1.1 Lektion 30-01-2018

1. Course introduction
2. Typical embedded system
3. Number formats (fixed- and floating-point)
4. Conversion between different number formats
5. (Blackfin) DSP Architecture
6. Software development flow

- ESP Chapter 1.1 + 1.2
- ESP 5.1 + 5.2.1
- ESP Chapter 6.1.1 (only p.217-p.222) and 6.1.3 - 6.1.5

1.1.1 Typical embedded system

- **Dedicated functions:** Embedded systems usaully execute a specific task repeatedly.
- **Tight constraints:** There are many constraints in designing an embedded system, such as; cost, processing speed, size, power consuption.
- **Reactive and real-time performance:** Many embedded systems must continously react to changes of the system's input.

1.1.2 Number formats (fixed- and floating-point)

- Fixedpoint

- Floatingpoint

- Blockfloatingpoint

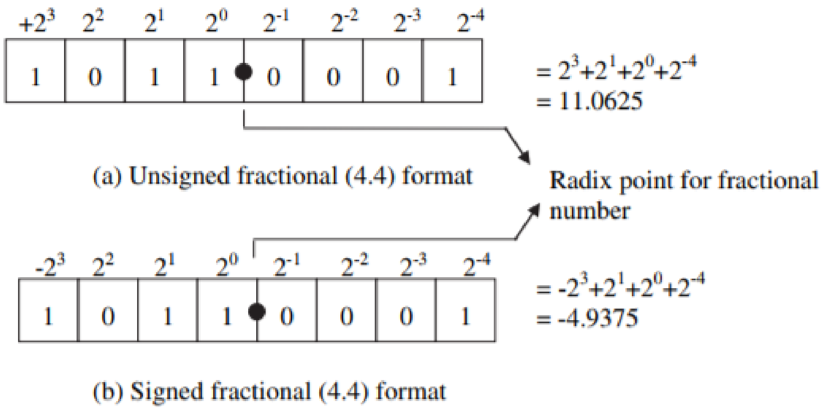
Fixed-point

- Binary data format - signed and unsigned

  - The 2's complement format is the most popular signed number in DSP processors.

  - Most DSP processors support both integer and fractional data formats.

    * In an integer format, the radix point is located to the right of the least significant bit (LSB).
    * In a fractional number format, the radix point is located within the binary number.
      · The number to the right of the radix point assumes a fractional binary bit, with a weighting of 2^{-p} where the lowest fractional increment is 2^4 (or 0.0625) in 1.1.
      · For the number to the left of the radix point, the weighting increases from 2^q. The weighting of the MSB (or sign bit) depends on whether the number is signed or unsigned.
    * (N.M) notation
      · N is the number of bits to the left of the radix point (integer part).
      · M is the number of bits to the right of the radix point (fractional part).
      · The symbol "." represents the radix point.
      · Total number of bits in the data word is B = N + M.



Figur 1.1: Example of 8-bit binary data formats for a fractional number.

- Dynamic Ranges and Precisions
  - The maximum positive number in (1.15) format is 12^{15}(= 0.999969482421875) (0x7FFF).
  - The minimum negative number in (1.15) format is 1 (0x8000).
  - The 1.15 format has a **dynamic range** of [+0.999969482421875 to 1]
    - Numbers exceeding this range cannot be represented in 1.15 format.
  - The smallest increment (or precision) within the (1.15) format is 2^{15}.
- Scaling Factors
  - A number in (N.M) format cannot be represented in the programs because most compilers and assemblers only recognize numbers in integer or (16.0) format.
    - Convert the fractional number in (N.M) format into its integer equivalent.
    - Its radix point must be accounted for by the programmers.
      - To convert a number 0.6 in (1.15) format to its integer representation, multiply it by 2^{15} (or 32 768) and round the product to its nearest integer to become 19 661 (0x4CCD).

In table 1.2 all 16 possible (N.M) formats for 16-bit numbers. Different formats give different dynamic ranges and precisions. There is a trade-off between the dynamic range and precision.

- As the dynamic range increases, precision becomes coarser.

Format ($N.M$)	Largest Positive Value (0x7FFF)	Least Negative Value (0x8000)	Precision (0x0001)
(1.15)	0.999969482421875	−1	0.00003051757813
(2.14)	1.99993896484375	−2	0.00006103515625
(3.13)	3.9998779296875	−4	0.00012207031250
(4.12)	7.999755859375	−8	0.00024414062500
(5.11)	15.99951171875	−16	0.00048828125000
(6.10)	31.9990234375	−32	0.00097656250000
(7.9)	63.998046875	−64	0.00195312500000
(8.8)	127.99609375	−128	0.00390625000000
(9.7)	255.9921875	−256	0.00781250000000
(10.6)	511.984375	−512	0.01562500000000
(11.5)	1,023.96875	−1,024	0.03125000000000
(12.4)	2,047.9375	−2,048	0.06250000000000
(13.3)	4,095.875	−4,096	0.12500000000000
(14.2)	8,191.75	−8,192	0.25000000000000
(15.1)	16,383.5	−16,384	0.50000000000000
(16.0)	32,767	−32,768	1.00000000000000

Figur 1.2: Dynamic ranges and precisions of 16-bit numbers using different formats.

1.1.3 Conversion between different number formats

Fixed-Point Data Types

The Blackfin C compiler supports eight scalar data types and two fractional data types.

Type	Number Representation
`char`	8-bit signed integer
`unsigned char`	8-bit unsigned integer
`short`	16-bit signed integer
`unsigned short`	16-bit unsigned integer
`int`	32-bit signed integer
`unsigned int`	32-bit unsigned integer
`long`	32-bit signed integer
`unsigned long`	32-bit unsigned integer
`fract16`	16-bit signed (1.15) fractional number
`fract32`	32-bit signed (1.31) fractional number

Figur 1.3: Fixed-Point Data Types.

1.1.4 (Blackfin) DSP Architecture

1.1.5 Software development flow

Quantization and fixed-point effects

2.1 Lektion 06-02-2018

1. ADC Quantization
2. Coefficient- and product-quantization
3. Overflow / underflow and coefficient scaling
4. Notch and peak filters as example

- ESP 6.1.1 (only p. 222 - 229)
- ESP 6.2.2, 6.2.3 (p. 240 - 243)
- ESP 3.4.2 + 3.4,3

2.1.1 ADC Quantization

2.1.2 Coefficient- and product-quantization

2.1.3 Overflow / underflow and coefficient scaling

2.1.4 Notch and peak filters as example