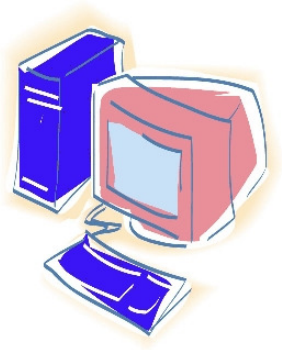
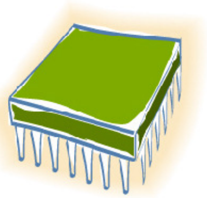
Fixed-Point modeling & analysis

From floating- to fixed-point



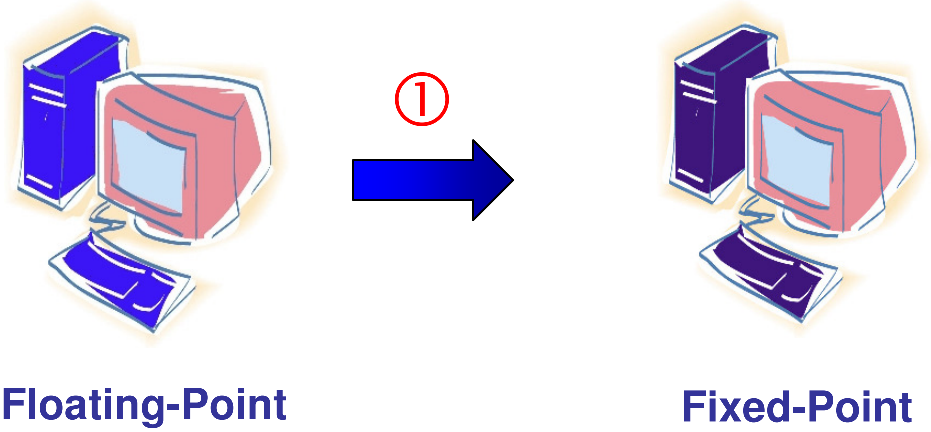
Floating-Point
“unlimited” range



Fixed-Point
limited precision

From floating- to fixed-point

- steps
  - refine the floating point model towards fixed-point precision: model conversion



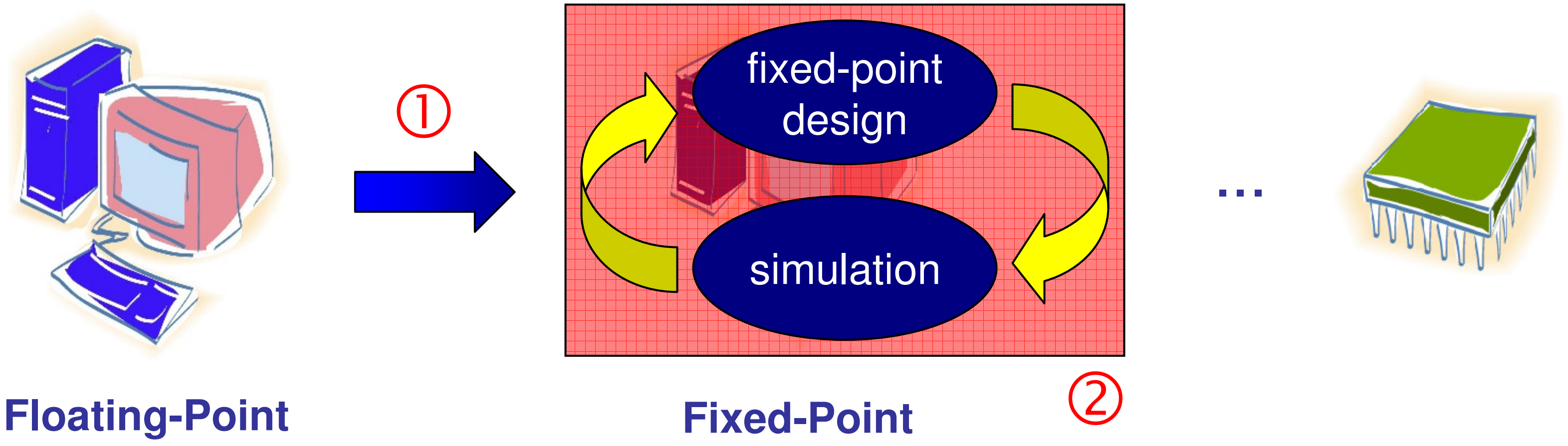
①

Floating-Point

Fixed-Point

From floating- to fixed-point

- steps
  - refine the floating point model towards fixed-point precision: model conversion
  - fixed-point design space exploration
    - scale properly (avoid overflow, minimize quantization error)
    - decide on the minimum required bit widths



Scope

- objectives
  - refine the floating point model towards fixed-point precision: model conversion
  - fixed-point design space exploration

- this requires
  - fixed-point modeling means
  - SQNR constraints

Fixed-point modeling

- C/C++ does not provide fixed-point data types
  - except for bool and char, the bit widths depend on the compiler and the computer architecture
  - but we need bit true data types…

data type	bit width
bool	1
char	8
short	>16
int	>short
long	>32, >int

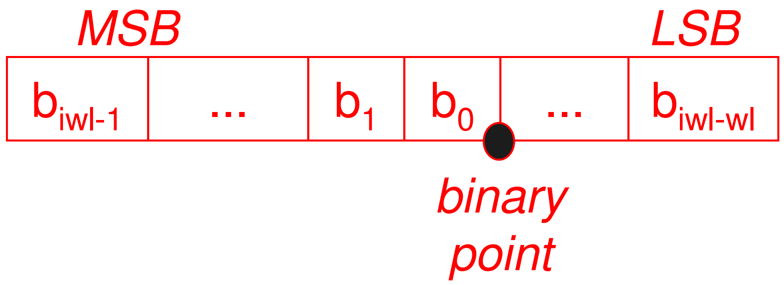
Fixed-point modeling

- SystemC extends C++ and provides support for
  - concurrent behaviors
  - hierarchical decomposition
  - communication
  - time modeling
  - …
  - fixed-point
    - sc_int, sc_uint
    - sc_fixed, sc_ufixed
    - …

SYSTEMC™

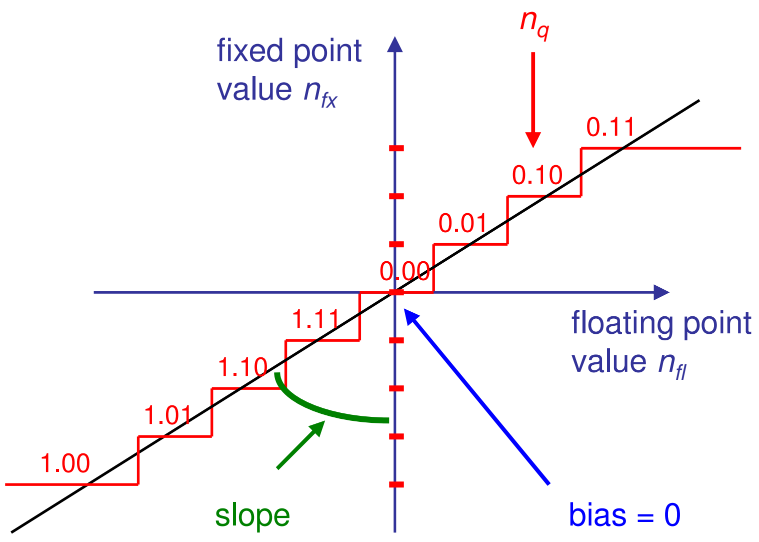
Fixed-point modeling

- fixed-point representation: word length
  - wl: total word length
  - iwl: integer word length



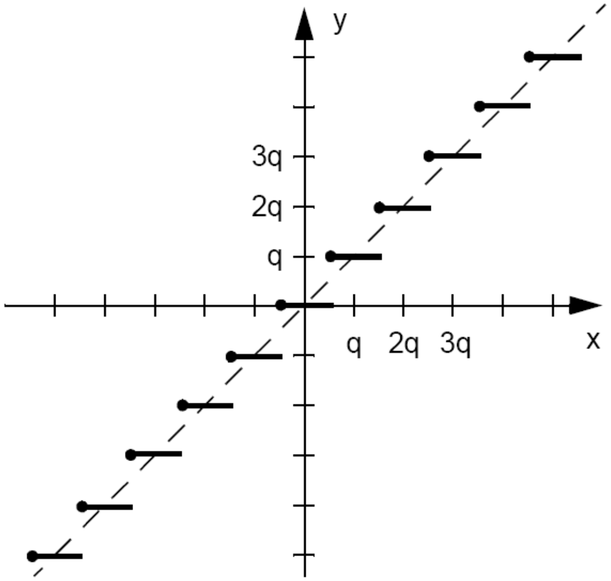
$$n_{fl} \cong n_{fx} = slope \cdot n_q + bias$$

$$\begin{cases} if \;\; unsigned: \;\; n_q = \displaystyle\sum_{i=iwl-wl}^{iwl-1} b_i \, 2^i \\ if \;\; signed: n_q = -b_{iwl-1} \, 2^{iwl-1} + \displaystyle\sum_{i=iwl-wl}^{iwl-2} b_i \, 2^i \end{cases}$$

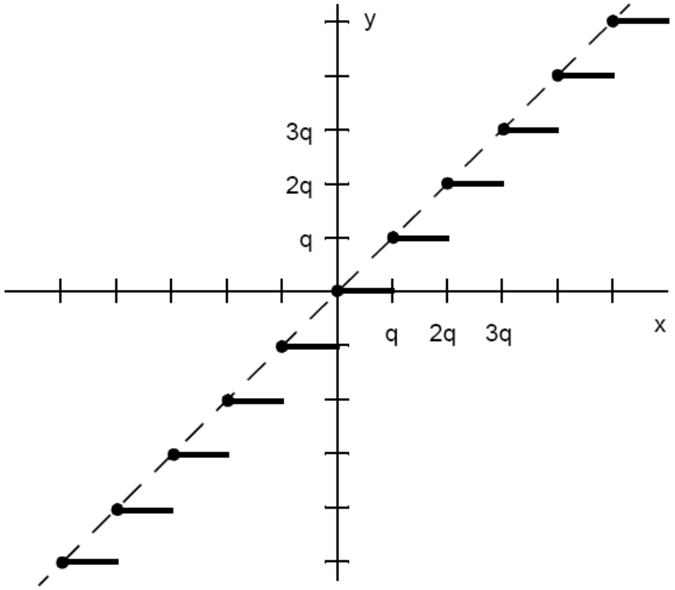


Fixed-point modeling

- fixed-point representation: quantization mode
  - determines the behavior of the fixed point type when the result of an operation generates more precision in the LSBs than is available



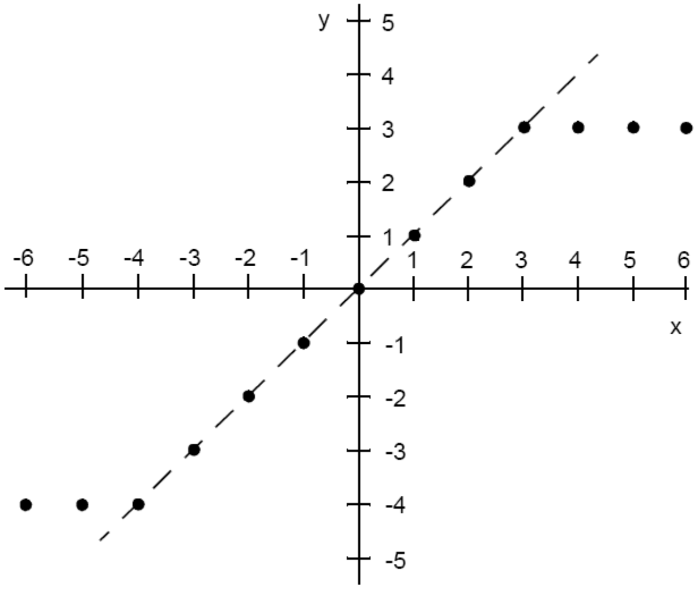
SC_RND



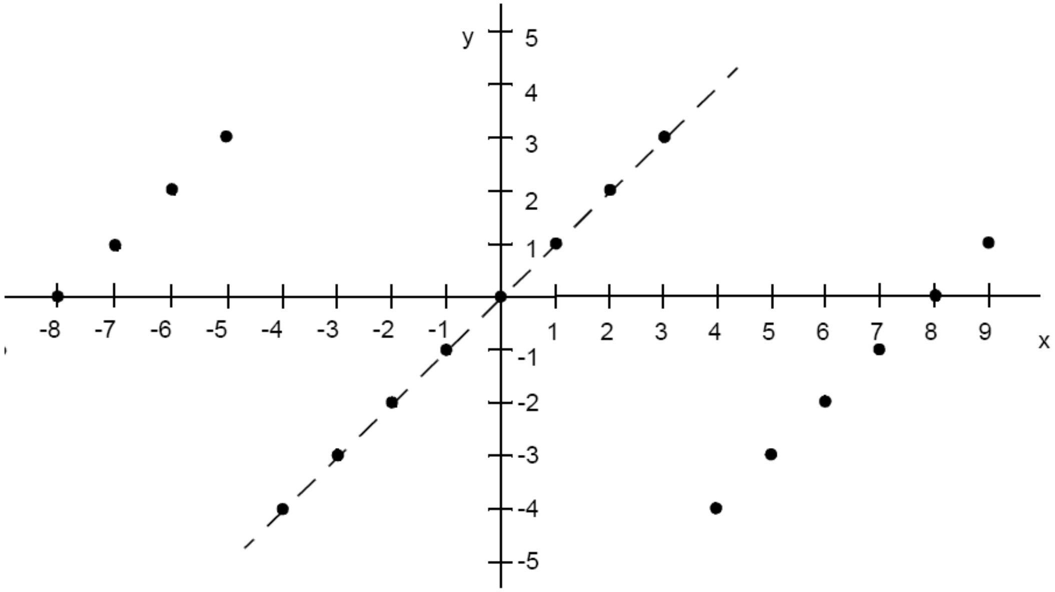
SC_TRN

Fixed-point modeling

- fixed-point representation: overflow mode
  - determines the behavior of the fixed point type when the result of an operation generates more precision in the MSBs than is available



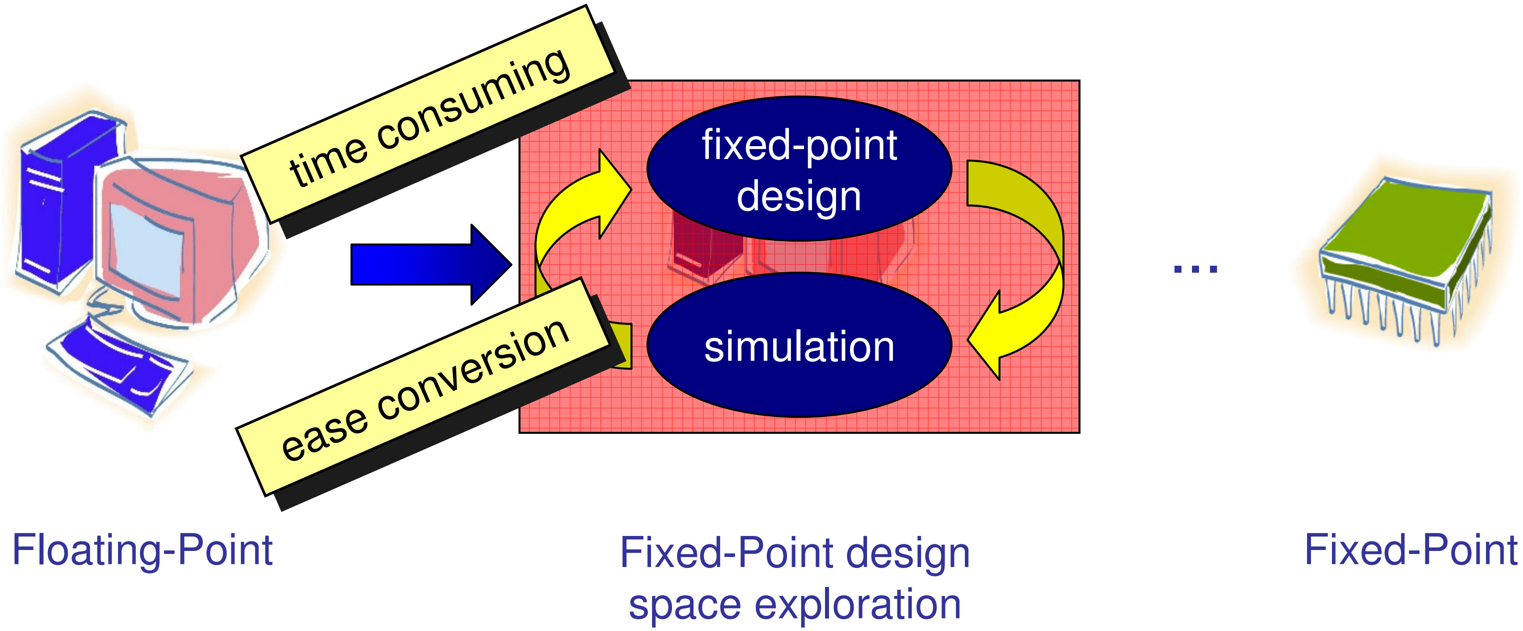
SC_SAT



SC_WRAP

Fixed-point modeling

- more infos: SystemC V2.0 User's Guide, Ch. 7
- still…



- HJ81: executable model supporting floating- <u>and</u> fixed-point precision
  - backward compatible

Fixed-point modeling HJ81

- Support for floating- and fixed-point

```
void rgb2yuv(D_PIXEL r,  D_PIXEL g, D_PIXEL b,
             D_PIXEL &y, D_PIXEL &u, D_PIXEL &v) {

  D_RGBCOEFF coeff[] = { 0.299,   0.587,   0.114,
                        -0.1687, -0.3313,  0.5,
                         0.5,    -0.4187, -0.0813};

  y = coeff[0] * r + coeff[1] * g + coeff[2] * b;
  u = coeff[3] * r + coeff[4] * g + coeff[5] * b + 128;
  v = coeff[6] * r + coeff[7] * g + coeff[8] * b + 128

}
```

specification of the fixed-point precision and behaviour

```
…
// Comment the following line in ord  t     ile in floating-point mode.
// Uncomment the following line in c       compile in fixed-point mode.
#define FINITE


…
#define D_PIXEL       FX_CHAR(SC_TRN,SC_WRAP)
#define D_RGBCOEFF    FX_FLOAT(□,□,□,□)
…
```

my_types.h

Fixed-point modeling HJ81

- Support for floating- and fixed-point

```
void rgb2yuv(D_PIXEL r,  D_PIXEL g, D_PIXEL b,
             D_PIXEL &y, D_PIXEL &u, D_PIXEL &v) {

  D_RGBCOEFF coeff[] = { 0.299,   0.587,   0.114,
                        -0.1687, -0.3313,  0.5,
                         0.5,    -0.4187, -0.0813};

  y = coeff[0] * r + coeff[1] * g + coeff[2] * b;
  u = coeff[3] * r + coeff[4] * g + coeff[5] * b + 128;
  v = coeff[6] * r + coeff[7] * g + coeff[8] * b + 128;

}
```

selection of the precision mode

```
…
// Comment the foll               der to compile in floating-point mode.
// Uncomment the f          ine in order to compile in fixed-point mode.
#define FINITE

…
#define D_PIXEL       FX_CHAR(SC_TRN,SC_WRAP)
#define D_RGBCOEFF    FX_FLOAT(□,□,□,□)
…
```

my_types.h

Fixed-point modeling HJ81

- Support for floating- and fixed-point

```
void rgb2yuv(char r, char g, char b,
             char &y, char &u, char &v) {

 …

}
```

```
…
// Comment the following line in order to compile in floating-point mode.
// Uncomment the following line in order to compile in fixed-point mode.
//#define FINITE

…
#define D_PIXEL      FX_CHAR(SC_TRN,SC_WRAP)
#define D_RGBCOEFF   FX_FLOAT(□,□,□,□)
…
```

my_types.h

Fixed-point modeling HJ81

- Support for floating- and fixed-point

```
void rgb2yuv(sc_fixed<8,1,SC_TRN,SC_WRAP> r,
             sc_fixed<8,1,SC_TRN,SC_WRAP> g,
             sc_fixed<8,1,SC_TRN,SC_WRAP> b,
             sc_fixed<8,1,SC_TRN,SC_WRAP> &y,
             sc_fixed<8,1,SC_TRN,SC_WRAP> &u,
             sc_fixed<8,1,SC_TRN,SC_WRAP> &v) {

 …

}
```

```
…
// Comment the following line in order to compile in floating-point mode.
// Uncomment the following line in order to compile in fixed-point mode.
#define FINITE

…
#define D_PIXEL      FX_CHAR(SC_TRN,SC_WRAP)
#define D_RGBCOEFF   FX_FLOAT(□,□,□,□)
…
```

my_types.h

Fixed-point modeling HJ81

- Support for floating- and fixed-point

```
…
// Comment the following line in order to compile in floating-point mode.
// Uncomment the fol                              e in fixed-point mode.
#define FINITE

…
// Declare ne      data types, which will be replaced by the corresponding
// floating-      ed-point type.
//
// Syntax:
//   FX_DOUBLE(wl, iwl, q_mode, o_mode)  signed   fixed or double
//   UFX_DOUBLE(wl, iwl, q_mode, o_mode) unsigned fixed or double
//   FX_FLOAT(wl, iwl, q_mode, o_mode)   signed   fixed or float
//   UFX_FLOAT(wl, iwl, q_mode, o_mode)  unsigned fixed or float
//   FX_CHAR(q_mode, o_mode)             signed   8-bits fixed or char
//   UFX_CHAR(q_mode, o_mode)            unsigned 8-bits fixed or char
//   FX_INT(iwl, q_mode, o_mode)         signed   fixed or int
//   UFX_INT(iwl, q_mode, o_mode)        unsigned fixed or int
//   FX_SHORT(iwl, q_mode, o_mode)       signed   fixed or short
//   UFX_SHORT(iwl, q_mode, o_mode)      unsigned fixed or short
…
#define D_PIXEL       FX_CHAR(SC_TRN,SC_WRAP)
#define D_RGBCOEFF    FX_FLOAT(□,□,□,□)
…
```

several data types available

my_types.h

Fixed-point modeling HJ81

- Your task
  - define the data types you think are needed
    - specify bit widths, quantization mode, overflow mode
  - change the model
  - verify the conversion is working fine
    - same result as in floating-point mode
    - acceptable degradation in fixed-point mode