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
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...

<table>
<thead>
<tr>
<th>data type</th>
<th>bit width</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>8</td>
</tr>
<tr>
<td>short</td>
<td>&gt;16</td>
</tr>
<tr>
<td>int</td>
<td>&gt;short</td>
</tr>
<tr>
<td>long</td>
<td>&gt;32, &gt;int</td>
</tr>
</tbody>
</table>
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
    - …
Fixed-point modeling

- fixed-point representation: word length
  - **wl**: total word length
  - **iw**: integer word length

**binary point**

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

if unsigned: $$n_q = \sum_{i=iw-l}^{iw-1} b_i 2^i$$

if signed: $$n_q = -b_{iw-l} 2^{iw-l} + \sum_{i=iw-w}^{iw-2} b_i 2^i$$
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
Fixed-point modeling

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

- HJ81: executable model supporting floating- and fixed-point precision
  - backward compatible
Fixed-point modeling HJ81

- Support for floating- and fixed-point

```c
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;
}
```

...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(8,1,SC_TRN,SC_WRAP)
...
Fixed-point modeling HJ81

- Support for floating- and fixed-point

```c
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;
}
```

... // 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(8,1,SC_TRN,SC_WRAP) ...

... // 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(8,1,SC_TRN,SC_WRAP) ...

my_types.h
Fixed-point modeling HJ81

- Support for floating- and fixed-point

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

```c
...// Comment the following line in order to compile in floating-point mode.
// Uncomment the following line in order to compile in fixed-point mode.
#ifndef FINITE
#define D_PIXEL      FX_CHAR(SC_TRN,SC_WRAP) #define D_RGBCOEFF   FX_FLOAT(8,1,SC_TRN,SC_WRAP) ...
#endif
```

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(8,1,SC_TRN,SC_WRAP)
...
Fixed-point modeling HJ81

- Support for floating- and fixed-point

```c
... // 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
...
// Declare next your data types, which will be replaced by the corresponding floating- or fixed-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(8,1,SC_TRN,SC_WRAP) ...
```

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