FIXED-POINT DESIGN

• Central issue: how to perform a desired computation with as few bits per operand as possible
• Some material based on:
• Thanks to Jeroen de Zoeten, for some material reused from his M.Sc. graduation presentation (2004).

TOPICS

• Fixed-point data types
• SystemC
• Peak-value estimation
• Word-length optimization

FIXED-POINT DATA TYPES

• A specific interpretation of a logic vector
  – Binary point
  – Integer and fractional part: iwl and fwl (integer and fractional word length)
  – Signed or unsigned

EXAMPLES OF FIXED-POINT NUMBERS

• Example pattern: 1101
  – With iwl = 2 and unsigned → 13/4
  – With iwl = 2 and signed → -3/4
  – With iwl = 6 and unsigned → 52
  – With iwl = 6 and signed → -12
  – With iwl = -1 and unsigned → 13/32
  – With iwl = -1 and signed → -3/32

© Sabih H. Gerez, University of Twente, The Netherlands
**FIXED-POINT ADDITION/SUBTRACTION**

- Integer adder can be used after:
  - Alignment of binary point
  - Sign extension

<table>
<thead>
<tr>
<th>A: Signed 2,4</th>
<th>B: Signed 4,2</th>
<th>Y: Signed 3,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,4)</td>
<td>(4,2)</td>
<td>(5,4)</td>
</tr>
<tr>
<td>S S S</td>
<td>S 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>

\[ Y = A + B \]

**FIXED-POINT MULTIPLICATION**

- Integer multiplier can directly be used.
- One only needs to figure out the location of the binary point.

<table>
<thead>
<tr>
<th>A: Signed 2,4</th>
<th>B: Signed 4,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,4)</td>
<td>(4,2)</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

**QUANTIZATION: TRUNCATION**

- If the target provides less accuracy than the value to assign:
  - Truncation \(\rightarrow\) no hardware

**QUANTIZATION: ROUNDELING**

- If the target provides less accuracy than the value to assign:
  - Rounding (various modes) \(\rightarrow\) extra hardware
OVERFLOW: WRAP AROUND
• If the value to assign is outside the range of target:
  – Wrap around → no hardware

OVERFLOW: SATURATION
• If the value to assign is outside the range of target:
  – Saturation (various modes) → extra hardware

SystemC
• Open source standard for system-level modeling, based on C++ class libraries and a simulation kernel.
• Provides modeling from system level down to (mainly) register-transfer level (RTL).
• For more details, see the Accellera web site (non-profit organization for system-level design):
  http://www.accellera.org/

SystemC FIXED-POINT DATA TYPES
• Declaration (signed and unsigned version):
  \[
  \text{sc\_fixed<wl, iwl, q\_mode, o\_mode, n\_bits> x;}
  \text{sc\_ufixed<wl, iwl, q\_mode, o\_mode, n\_bits> x;}
  \]
• \(wl\): word length, \(iwl + fwl\)
• \(iwl\): integer word length
• \(q\_mode\): (optional) quantization mode, default is truncation
• \(o\_mode\): (optional) overflow mode, default is wrap around
• \(n\_bits\): (optional) number of bits for overflow (\(n\_bits\) are saturated, the others are wrapped around)
• \(\text{sc\_fix/\_ufix}\) data types can be resized at run time
### SystemC FIXED-POINT CODE EXAMPLE

```cpp
sc_fixed<6, 2> a;
sc_fixed<6, 4> b;
sc_fixed<3, 2, SC_RND, SC_SAT> c;
c = a + b;
```

**Implementation:**
- Calculate sum at full precision
- Perform quantization processing
- Perform overflow processing

### THE FIXED-POINT DESIGN PROBLEM (1)

- Mathematical descriptions of DSP algorithms often assume infinite precision in the signal representation.
- The closest approximation of infinite precision in computers is the floating-point number representation.
- Floating-point hardware is expensive and is avoided if possible.
- Implementations therefore use fixed-point hardware.

- **Problem:** *which fixed-point formats should be used to obtain the cheapest implementation of the original algorithm?*

### THE FIXED-POINT DESIGN PROBLEM (2)

- One should look at:
  - The dynamic range: avoid *overflow* and therefore know *peak values*.
  - The accuracy: *quantization* levels.

### TOPICS OF FIXED-POINT DESIGN, PART 2

- Fixed-point design problem
- Peak-value estimation
- Word-length optimization
BOUGANIS FIXED-POINT FORMAT

Considers signed numbers only; sign bit is not counted in size.

PEAK-VALUE ESTIMATION

- Related to the fact that signal magnitude may grow due to addition or multiplication
- In a stable system, the signal cannot grow indefinitely
- Question is: what is the maximal value encountered for each signal in the system?
- Issue is not directly related to accuracy, the number of bits used for each signal.

PEAK-VALUE ESTIMATION METHODS

- Analytic:
  - examine transfer functions
- Data-range propagation:
  - Interval analysis
  - Compute result interval from input intervals
  - Tends to overestimate requirements
- Simulation-driven analysis:
  - Monitor values produced during a representative simulation and record extremes (possible in System Studio)
  - Use a safety factor > 1

ANALYTIC PEAK-VALUE ESTIMATION

- Consider an FIR filter:
  \[ y[n] = \sum_{k=0}^{N} h[k] \cdot x[n-k] \]
- Then, an upper bound for the output value is found by:
  \[ y_{\text{peak}} = x_{\text{peak}} \sum_{k=0}^{N} |h[k]| \]
- For recursive filters, a similar approach can be followed, starting from a state-space representation.
INTERVAL ANALYSIS (1)

- Represent each value $x$ as an interval: $\tilde{x} = [x^-, x^+]$
- For each arithmetic operation, one can calculate the result interval from the operand intervals. For example:

\[
\tilde{x} + \tilde{y} = [x^- + y^-, x^+ + y^+]
\]
\[
\tilde{x} \cdot \tilde{y} = [\min(x^-, y^-, x^-, y^+, x^+, y^-, x^+, y^+), 
\max(x^-, y^-, x^-, y^+, x^+, y^-, x^+, y^+)]
\]

INTERVAL ANALYSIS (2)

Beware: this is no FIR filter, but a phantasy design.

WORD-LENGTH PROPAGATION

<table>
<thead>
<tr>
<th>Type</th>
<th>Propagation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAIN</td>
<td>For input $(n_a, p_a)$ and coefficient $(n_b, p_b)$: $p_j = p_a + p_b$ $n_j^u = n_a + n_b$</td>
</tr>
<tr>
<td>ADD</td>
<td>For inputs $(n_a, p_a)$ and $(n_b, p_b)$: $p_j = \max(p_a, p_b) + 1$ $n_j^u = \max(n_a, n_b + p_a - p_b) - \min(0, p_a - p_b) + 1$ (for $n_a &gt; p_a - p_b$ or $n_b &gt; p_b - p_a$)</td>
</tr>
<tr>
<td>DELAY or FORK</td>
<td>For input $(n_a, p_a)$: $p_j = p_a$ $n_j^u = n_a$</td>
</tr>
</tbody>
</table>

QUANTIZATION: NOISE MODELING (1)

- Suppose signal with fixed-point format $(n, 0)$ is multiplied with another signal with fixed-point format $(n, 0)$ and the result is truncated to $n$ bits.

  - Error ranges from 0 to $2^{-2n} - 2^{-n} \approx -2^{-n}$
  - Uniform distribution of error: $p(e) = 2^n$, $e \in [-2^{-n}, 0]$
NOISE MODELING (2)

- Average error is: $-2^{-(n+1)}$
- Variance:

$$\sigma^2 = \int_{-2^{-n}}^{0} 2^n [e + 2^{-(n+1)}]^2 \, de = \frac{1}{12} 2^{-2n}$$

NOISE PROPAGATION

- In linear time-invariant (LTI) systems, one can analytically calculate the effect of quantization in input or intermediate nodes to noise on the output.
- In case of non-linear systems, one could linearize the system by means of Taylor expansion (a similar approach as a small-signal model used in electronics).
- Noise propagation methods have the advantage of reduced computational complexity with respect to a simulations-only approach.

FIXED-POINT OPTIMIZATION PROBLEM

- Define a performance measure. Examples:
  - SNR at the output of a filter
  - Bit-error rate in a communication system
- Define a cost measure, such as the area of the circuit.
- Goal is to satisfy a performance requirement at minimal cost by optimally choosing a fixed-point format for each signal in the system.
- The most practical approach is to start with a floating-point model and gradually replace the data types by fixed-point types while monitoring performance by simulations.

SCHEDULING, ETC.

- Sharing of resources across multiple clock cycles puts additional constraints on the fixed-point format of signals.
NON-MONOTONIC BEHAVIOR

• One would expect that larger word lengths always improve the performance measure.
• It is possible, however, to construct systems where performance is non-monotonic, see:
• Such systems have *forks* that use different fixed-point formats at each end and reconverge.