Number Systems: Algebraic

Algebraic Number

e.g. $a = \pi + b$


- High level abstraction
- Infinite precision
- Often easier to understand
- Good for theory/algorithm development
- Hard to implement
Number Systems: Floating Point

- Widely used in CPUs
- Floating precision
- Good for algorithm study and validation

Value = \((-1)^{\text{Sign}} \times \text{Fraction} \times 2^{(\text{Exponent} - \text{Bias})}\)

<table>
<thead>
<tr>
<th>IEEE 754 standard</th>
<th>Sign</th>
<th>Exponent</th>
<th>Fraction</th>
<th>Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double precision</td>
<td>1</td>
<td>11 [62:52]</td>
<td>52 [51:00]</td>
<td>1023</td>
</tr>
</tbody>
</table>

A short floating-point number

\[
\pi = 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1
\]

\[
\pi = (-1)^0 \times (1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 0\times2^{-4} + 1\times2^{-5} + 0\times2^{-6})
\times 2^{1\times2^2 + 0\times2^1 + 1\times2^0 - 3} = 3.125
\]

Number Systems: Fixed Point

- Economical implementation
- \(W_{\text{int}}\) and \(W_{\text{fr}}\) suitable for predictable range
- o-mode (saturation, wrap-around)
- q-mode (rounding, truncation)
- Economic for implementation
- Useful built-in MATLAB functions: e.g. fix, round, ceil, floor, dec2bin, bin2dec, etc.

\[
\pi = -0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 0\times2^{-5} + 1\times2^{-6}
= 3.140625
\]

\[
\pi = 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1
\]

\(\Rightarrow\) In MATLAB:
\[
\text{dec2bin(round(pi*2^6),10)}
\text{bin2dec(above)*2^{-6}}
\]

\(\Rightarrow\) Simulink SynDSP and SysGen
Motivation for Floating-to-Fixed Point Conversion

Algorithms designed in algebraic arithmetic, verified in floating-point or very large fixed-point arithmetic

\[ a = \pi + b \]

VLSI Implementation in fixed-point arithmetic

\[ \pi = \begin{array}{ccc} S & W_{\text{int}} & W_{\text{Fr}} \end{array} \]

\[ \begin{array}{ccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \]

\[ \text{Overflow-mode} \quad \text{Quant.-mode} \]

- Time consuming
- Error prone

Hardware mapping

> 1 month

Floating-pt algorithm

OK?

No

Quantization

Fixed-pt algorithm

OK?

No

Yes

Idea

Algorithm designed in algebraic arithmetic, verified in floating-point or very large fixed-point arithmetic

\[ a = \pi + b \]

Optimization Techniques: FRIDGE

[2]

Set of test vectors for inputs

Pre-assigned \( W_{\text{Fr}} \) at all inputs

Range-detection through simulation

\[ W_{\text{int}} \]

\[ W_{\text{int}} \text{ in all internal nodes} \]

- Conservative but good for avoiding overflow

Deterministic propagation

\[ W_{\text{Fr}} \]

\[ W_{\text{Fr}} \text{ in all internal nodes} \]

- Unjustified input \( W_{\text{Fr}} \)
- Overly conservative

Optimization Techniques: Robust Ad Hoc

- **Ad hoc search** [3] or procedural [4]
  - Long bit-true simulation, large number of iterations [5]
  - Impractical for large systems

---

**Problem Formulation: Optimization**

Minimize hardware cost:

\[ f(W_{int,1}, W_{fr,1}; W_{int,2}, W_{fr,2}; ...; o-q-modes) \]

Subject to quantization-error specifications:

\[ S_j(W_{int,1}, W_{fr,1}; W_{int,2}, W_{fr,2}; ...; o-q-modes) < \text{spec}, \forall j \]

Feasibility:

\[ \exists N \in \mathbb{Z}^+, \text{s.t. } S_j(N, N; ...; \text{any mode}) < \text{spec}, \forall j \]

Stopping criteria:

\[ f < (1 + \alpha)f_{opt} \text{ where } \alpha > 0. \]

From now on, concentrate on \( W_{fr} \) [1]

---

**Perturbation Theory On MSE** [6]

Output MSE Specs:

\[
\text{MSE} = \mathbb{E}[(\text{Infinite-precision-output} - \text{Fixed-point-output})^2]
\]

\[
\mu = \mu^T B \mu + \sum_{i=1}^p c_i 2^{-W_{i,j}} \text{ for a datapath of } p, \ W_L \ B \in \mathbb{S}^p_+, C \in \mathbb{R}^p_+ \\
\mu_i = \left\{ \begin{array}{l}
\frac{1}{2} q_i w_{-W_{i,j}}, \text{ datapath} \\
\text{fix-pt}(c_i, 2^{-W_{i,j}}) - c_i, \text{ const } c_i
\end{array} \right\}
\]

\[
q_i = \left\{ \begin{array}{l}
0, \text{ round-off} \\
1, \text{ truncation}
\end{array} \right\}
\]


---

**Actual vs. Computed MSE**

**11-tap LMS Adaptive Filter**

Check MSE-cost Fitting Behavior

- Further improvement can be made considering correlation

\[
\text{MSE} \cong \sum_{i, j, m, T} \mathbb{E}[b_i, b_j] |\mathbb{E}[c_i, c_j, c_m, T]| = \mu^T B \mu + \sigma^T C \sigma
\]

with \( B, C \in \mathbb{S}^p_+ \), and \( \sigma_i = 2^{-W_{i,j}} \)

- More simulations required
- Usually not necessary

**SVD U-Sigma**

Check MSE-cost Fitting Behavior

12.10
FPGA Hardware Resource Estimation

Design Mapping

- **Accurate**
- Sometimes unnecessarily accurate
- Slow (minutes to hours)
- Excessive exposure to low-end tools
- No direct way to estimate subsystem
- Hard to realize for incomplete design

Fast and flexible resource estimation is important for FFC! Tool needs to be orders of magnitude faster

Model-based Resource Estimation

Individual MATLAB function created for each type of logic
- MATLAB function estimates each logic-block area based on design parameters (input/output WL, o, q, # of inputs, etc...)
- Area accumulates for each logic block

```
63 - area = area + register_area(0,0,0,0,0,0,0,0);
64 - area = area + addbit_area(0,0,0,0,0,0,0,0);
65 - area = area + accum_area(0,0,0,0,0,0,0,0);
66 - area = area + addsub_area(0,0,0,0,0,0,0,0);
67 - area = area + counter_area(0,0,0,0,0,0,0,0);
68 - area = area + counter_area(0,0,0,0,0,0,0,0);
```

Total area accumulated from individual area functions (register_area, accum_area, etc...)
- Xilinx area functions are proprietary, but ASIC area functions can be constructed through synthesis characterizations

[*] by C. Shi and Xilinx Inc. (© Xilinx)
ASIC Area Estimation

ASIC logic block area is a multi-dimensional function of its input/output WL and speed, constructed based on synthesis

- Each WL setting characterized for LP, MP, and HP
- Perform curve-fitting to fit data unto a quadratic function

![Graphs showing ASIC area estimation](image)

Analytical Hardware-Cost Function: FPGA

- If all design parameters (latency, o, q, etc.) and all $W_{\text{int}}$’s are fixed, then the FPGA area is roughly quadratic to $W_{Fr}$

$$f(W) \approx W^T H_1 W + H_2 W + h_3,$$ where $W = (W_{Fr,1}, W_{Fr,2}, \ldots)^T$

![Graphs showing comparison of actual vs. modeled hardware cost](image)
Wordlength Optimization Flow

Simulink Design in XSG or SynDSP → Initial Setup (10.16) → WL Analysis & Range Detection (10.18) → Optimal WL


Create Cost-function for ASIC (10.12) → Create cost-function for FPGA (10.12) → MSE-specification Analysis (10.22)

Data-fit to Create HW Cost Function (10.21) → Data-fit to Create MSE Cost Function (10.22)

Wordlength Optimization → Optimization Refinement (10.23) → Optimal W_F

Note: See the book website for tool download.

Initial Setup

- Insert a FFC setup block from the library – see notes
- Insert a “Spec Marker” for every output requiring MSE analysis
  – Generally every output needs one
**Wordlength Reader**

- Captures the WL information of each block
  - If user specifies WL, store the value
  - If no specified WL, back-trace the source block until a specified WL is found
    - If source is the input-port of a block, find source of its parent

**Wordlength Analyzer**

- Determines the integer WL of every block
  - Inserts a “Range Detector” at every active/non-constant node
  - Each detector stores signal range and other statistical info
  - Runs 1 simulation, unless specified multiple test vectors
Wordlength Connectivity

- Connect wordlength information through WL-passive blocks
  - Back-trace until a WL-active block is reached
  - Essentially “flattens” the design hierarchy
  - First step toward reducing # of independent WLs

Wordlength Grouping

- Deterministic
  - Fixed WL (mux select, enable, reset, address, constant, etc)
  - Same WL as driver (register, shift reg, up/down-sampler, etc)
- Heuristic (WL rules)
  - Multi-input blocks have the same input WL (adder, mux, etc)
  - Tradeoff between design optimality and simulation complexity
Resource-Estimation Function, Analyze HW Cost

- Creates a function call for each block

```
area=area+wux_area('u', 'u', 'u', 'u', 'u', 0+3wux_fatp0_w0); 44
area=area+polux_area('u', 0+3wux_fatp0_w0); ...
```

Slide 12.12, 12.14

- HW cost is analyzed as a function of WL
  - One or two WL group is toggled with other groups fixed
    - Quadratic iterations for small # of WLs
    - Linear iterations for large # of WLs

\[
\text{Resource-Estimation Function, Analyze HW Cost}
\]

Analyze Specifications, Analyze Optimization

- Computes MSE’s sensitivity to each WL group
  - First simulate with all WL at maximum precision
  - WL of each group is reduced individually

- Once MSE function and HW cost function are computed, user may enter the MSE requirement
  - Specify 1 MSE for each Spec Marker

- Optimization algorithm summary
  1) Find the minimum \( W_{fr} \) for a given group (others high)
  2) Based on all the minimum \( W_{fr} \)'s, increase all WL to meet spec
  3) Temporarily decrease each \( W_{fr} \) separately by one bit, only keep the one with greatest HW reduction and still meet spec
  4) Repeat 3) until \( W_{fr} \) cannot be reduced anymore

Slide 12.9, 12.10

12.21

11/5/2012
Optimization Refinement and Result

- The result is then examined by user for suitability
  - Re-optimize if necessary, only takes seconds

**Example: 1/sqrt() on an FPGA**

- (14,9)
- (24,16) (13,8)
- (24,16) (11,6)
- (24,16) (10,6)
- (16,11) (10,7)
- (12,9) (10,7)

About 50% area reduction

Legend:
- red = WL optimal
- black = fixed WL

- About 50% area reduction
- 409 slices
- 877 slices

ASIC Example: FIR Filter

**Original Design**
- Area = 48916 μm²

**Optimized for MSE = 10^-6**
- Area = 18356 μm²

Example: Jitter Compensation Filter

![Diagram of Jitter Compensation Filter]

Example: Jitter Compensation Filter

\[29.4 \text{ dB}\]

\[30.8 \text{ dB}\]


Tradeoff: MSE vs. Hardware-Cost

![Tradeoff Graph]

Acceptable MSE

WL-Optimal Design

ACPR (MSE=6x10^{-3})

46 dB

ACPR (MSE=7x10^{-3})
Summary

- Wordlength minimization is important in the implementation of fixed-point systems in order to reduce area and power
  - Integer wordlength can be simply found by using range detection, based on input data
  - Fractional wordlengths require more elaborate perturbation theory to minimize hardware cost subject to MSE error due to quantization
- Design-specific information can be used
  - Wordlength grouping (e.g. in multiplexers)
  - Hierarchical optimization (with fixed input/output WLs)
  - WL optimizer for recursive systems takes longer due to the time require for algorithm convergence
- FPGA/ASIC hardware resource estimation results are used to minimize WLs for FPGA/ASIC implementations

References (1/3)

References (2/3)

- See book supplement website for tool download.
  – Also see: [http://bwrc.eecs.berkeley.edu/people/grad_students/ccshi/research/FFC/documentation.htm](http://bwrc.eecs.berkeley.edu/people/grad_students/ccshi/research/FFC/documentation.htm)

References (3/3)

Course Wiki – CAD Tutorials

WL Optimization Tool

- Source code
- Tested with Matlab 2006b and SynDSP 3.6