A Software Tool for Designing Fixed-Point Implementations of Computational Data Paths for Embedded and Reconfigurable Computational Environments

David M. Buehler Gregory W. Donohoe

University of Idaho
Center for Advanced Microelectronics and Biomolecular Research

Pen-Shu Yeh

NASA GSFC

Outline

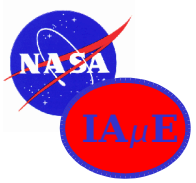
- Problem Overview
  - Floating-point vs. fixed-point computations
  - Terminology: “Computational Data Path”, “Fixed-point Implementation”
- Methodology for determining a fixed-point implementation of a computation
  - Information about how the run-time integral values are partitioned
- SIFOpt – a design tool which implements this methodology

Problem Overview

- Desire: “paper and pencil” description of an implementation of a computation:
  - Variables, unary/binary mathematical operators, constants.
- Floating-point implementations provide this, but circuitry is expensive.
- Fixed-point implementations require the designer to deal with design-time scaling factors.

Comparing Floating and Fixed-Point Representations of "Real" values

Feature	Floating-Point	Fixed-Point
Scaling	Run-time. Rules fixed in hardware.	Design (compile)-time. (Hardware sees only Integers)
Ease of Software Design	Designer friendly: hardware takes care of scaling.	Difficult: Designer in charge of scaling operations.
Dynamic Range	Fail-soft: adjust granularity	Fail-hard: overflow or saturate
Computational Error	Minimizes relative error, unknown absolute error.	Designer knows bounds on absolute error.
Value	Explicit in representation	Interpreted by the application designer
Speed (of computa-	Slow, data dependent.	Fast, known at design time.
Hardware Complexity	Complex, BIG, power-hungry.	Simple (relative to floating-point.)



Computational Data Path

- A sequence of arithmetic operations which perform some computation.
- Variables, constants and mathematical operators.
- Can be expressed as a computation tree with no cycles.
- Inputs can be distinguished from computed values (leaves of the computation tree.)

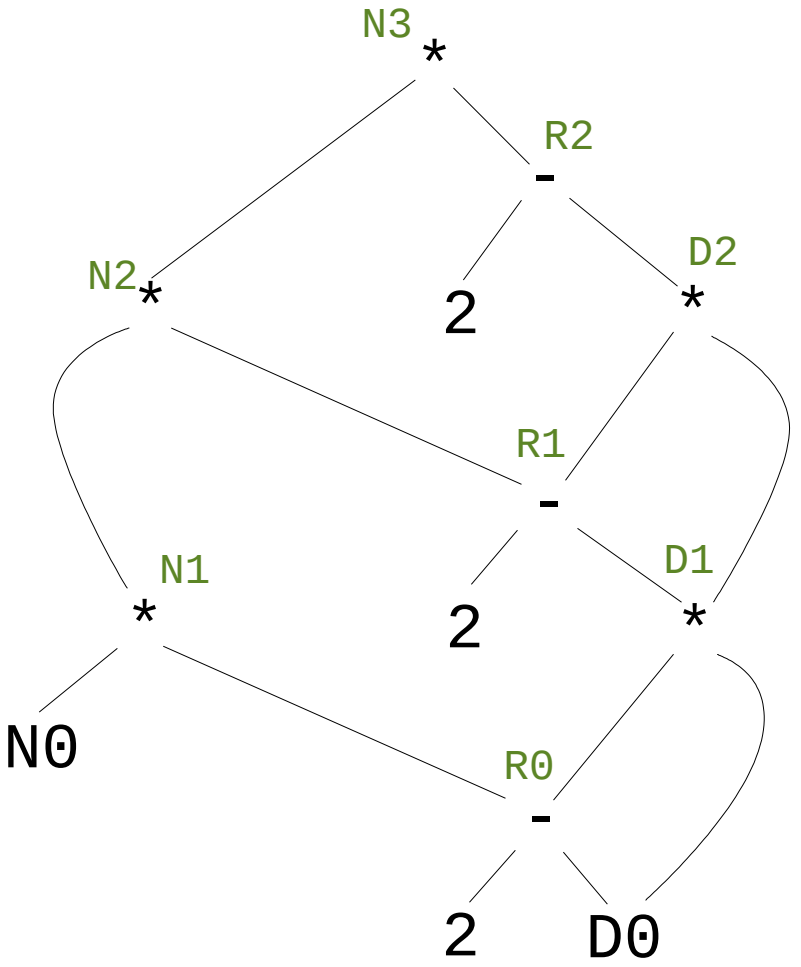
Computational Data Path

```
R0 = 2 - D0;

N1 = N0 * R0;
D1 = D0 * R0;
R1 = 2 - D1;

N2 = N1 * R1;
D2 = D1 * R1;
R2 = 2 - D2;

N3 = N2 * R2;
```



Fixed-Point Implementation

- Runtime: values and operations are exclusively integral.
- Design-time: scaling factors (2^n) are associated with every value in the computation. Each value in the computation can have a unique scaling factor associated with it.
- Real values: run-time integer multiplied by the scaling factor.

Fixed-Point Implementation

- Challenges for the designer:
  - Addends must have equal scaling factors.
  - Scaling factors must be computed/ tracked.
  - Multiplicands must be pre-scaled or results of multiplication post-scaled to limit growth of the data path width.
  - Constant values must be represented as integer scaling factor pairs.

Fixed-Point Implementation

```
R0 = 2 - D0;

N1 = N0 * R0;
D1 = D0 * R0;
R1 = 2 - D1;

N2 = N1 * R1;
D2 = D1 * R1;
R2 = 2 - D2;

N3 = N2 * R2;
```

```
int R0 = 512 – D0;

int N1 = N0 * R0;
int D1 = D0 * R0;
int R1 = 131072 - D1;

int N2 = (N1 >> 1) * (R1 >> 1);
int D2 = D1 * R1;
int R2 = 0x80000000 - (D2 >> 2);

int N3 = (N2 >> 16) * (R2 >> 15);
```

Methodology for Designing a Fixed-Point Implementation...

- Partitioning of integer values, scaling factors
- Static analysis of computational tree
  - Addition
  - Multiplication
  - Constants
- Estimate absolute error

Partitioning the Integral (run-time) Values

- Bits in the runtime integral values have three possible semantics:
  - Sign bits
  - Mantissa bits
  - Right-padding (unused) bits

SSSSSSSMMMMMMMMM

SSSSMMMMMMMMM000

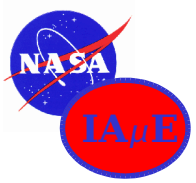
(fixed wordlength computational environments)

Partitioning the Integral Values

SIF Partitioning

(7/4/5)	SSSSSSSIIIIFFFFF
(4/4/5)	SSSSIIIIFFFFF000
(+6/5/5)	000000IIIIIFFFFF
(6/0/10)^-3	SSSSSSFFFFFFFFFF

SIF + Wordlength → Scaling Factor



Static Determination of a Fixed-Point Implementation

- Post-order traversal of the computation tree.
  - Variable leaves (input/argument variables) must be assigned SIF partitionings.
- Each value will get assigned:
  - An SIF partitioning (scaling factor).
  - An integer value range.
- Alignment and scaling operations will be inserted into the tree.

Addition

- Insert shift operations to align binary points.
- Minimize truncation error by utilizing knowledge of which bits are mantissa bits (SIF partitioning.)
- Tracking integer value range provides logarithmic growth of mantissa due to carry-out assumptions.

Multiplication

- In computational environments where the result is not twice as wide as the multiplicands, insert pre-scaling operations.
- Tracking integer value range allows determination of cases where no carry out will occur.

Constants

- Integer / scaling-factor representations of constants are determined optimally.

Tracking Absolute Error

Value before right-shift operation:

$$v_R = I \cdot f$$

Value after right-shifting by 4 bit positions:

$$v'_R = I' \cdot f'$$

$$\varepsilon = 0.375$$

$$v_R = I' \cdot f' + \varepsilon$$

SIFOpt

- A software tool which implements the methodology which has been outlined
- Designer can override some automated decisions
- How SIFOpt performs on some “real world” examples

Example

SIFOpt input file

```
// For 16-bit words...
var (9/0/7) in0;
var (9/0/7) in1;
var (9/0/7) in2;
var (9/0/7) in3;
var out = 0.15*in0 + 0.35*in1 + 0.35*in2 + 0.15*in3;
```

C output

```
short in0;
short in1;
short in2;
short in3;
short out = (((((154 * in0) >> 1) + (179 * in1)) >> 1) +
((179 * in2) >> 1)) + ((154 * in3) >> 2);
```

Giving the Designer Control

Correlations between values (both static (design-time) and dynamic (runtime)) are not taken into account.

- Designer can specify that a specific mathematical operation will not cause a carry out – thus limiting value range.
- Designer can explicitly specify value ranges of computed values.

Results

- Computations:
  - Convolution.
  - Division by repeated multiplications (image pixel nonuniformity correction.)
  - Goertzel's Discrete Fourier Transform algorithm.
- Future work
  - Representing / specifying data dependencies.

Experimental Results 8-weight convolution

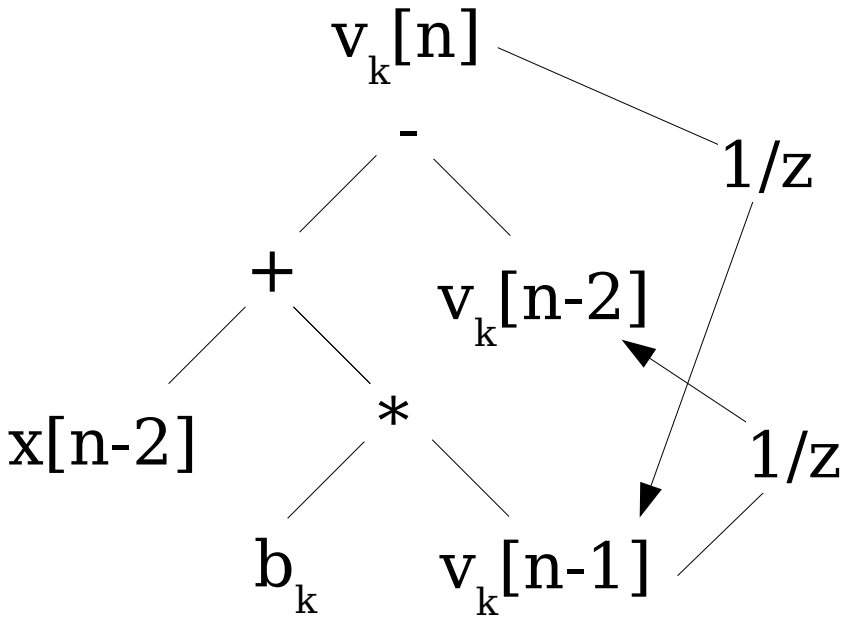
Variable	Runtime Maximum	Runtime Minimum	Maximum Runtime Error	SIFOpt Truncation Error Estimate
con8a0	0x51eadc29	0xae1523d7	1.90729E-008	1.90729E-008
con8a1	0x479471ec	0xb86b8e12	2.03331E-007	2.09875E-007
con8a2	0x4c8d699d	0xb3729662	2.55122E-007	2.86283E-007
con8a3	0x5418ddb9	0xabe72245	3.98783E-007	5.91566E-007
con8a4	0x502c0337	0xafd3fcc8	3.44589E-006	3.64346E-006
con8a5	0x6be29a4a	0x941d65b4	4.66071E-006	4.86413E-006
con8a6	0x78706a3a	0x878f95c3	5.03520E-006	5.24599E-006
con8a7	0x7e90455b	0x816fbaa1	5.21792E-006	5.43716E-006

Experimental Results
Division by Repeated Multiplications

Variable	Runtime Maximum	Runtime Absolute Error	SIFOpt Maximum Error Value
N2	0xfd818000	0.00006074	0.00006080
R2	0x7f017f01	0.00000000	0.00000000
N3	0xfb87f902	0.00032146	0.00036430
D3	0xffffffff	0.00002947	0.00004572
R3	0x7e06f508	-0.00002947	0.00004572
N4	0xf7a5a9db	0.00104036	0.00157476
D4	0xffffffff	0.00007526	0.00018269
R4	0x7c1d92bd	-0.00007526	0.00018269
N5	0xf020eb07	0.00229392	0.00700224
D5	0xffffffff	0.00012556	0.00059309
N7	0xc8f5e2b6	-0.01013790	0.18792800
D7	0xffffffff	0.00016305	0.00551102
R7	0x66c9ddd1	-0.00016305	0.00551102
N8	0xa15f95af	-0.01951070	1.07691000

Goertzel's Fourier Transform Algorithm

$$v_k[n]=x[n-2]+b_k \cdot v_k[n-1]-v_k[n-2]$$



Determine magnitude of largest value on vk[n] using sample data. This gives us the required range for the leaves, which we use to set an SIF partitioning for the feedback “inputs”.

Conclusions

- A methodology for designing fixed-point implementations of computations has been outlined.
- This methodology has been implemented in the design tool SIFOpt, which has been used to create fixed-point implementations of several algorithms.
- Methods for determining data dependent optimizations need to be explored further.