Towards Automatic Accuracy Validation and Optimization of Fixed-Point Hardware Descriptions in SystemC

Arnaud Tisserand

CNRS, IRISA laboratory, CAIRN research team

SCAN 2010, September 27–30, Lyon, France

Outline

- Motivations
- Used tools
  - SystemC fixed-point support overview
  - Gappa software for error bounds determination/verification
- Developed library
  - Overall description
  - Validation mode (certified error bounds)
  - Optimization mode (operands width reduction under certified accuracy constraint)
  - Experimental results
- Conclusion & future prospects

Motivations

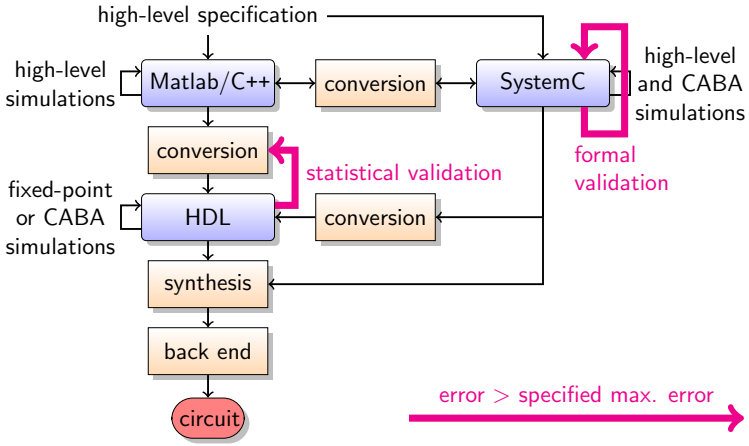
In hardware, fixed-point arithmetic is prefered to floating-point:

- smaller operators (circuit cost and static power consumption)
- faster operators (important speedup for addition)
- lower dynamic power consumption
- limited accuracy requirements in most of embedded applications (8 to 24 bits)
- but dynamic range limitation (tradeoff between the datapath width and the number of operand scaling operations)

Accuracy validation:

- Average error analysis (widely used in signal processing) assumption: errors $\simeq$ noise sources [5]
- Maximum error analysis (this work)

Typical and Proposed Design Method



HDL: hardware description language (VHDL or Verilog)

CABA: cycle accurate bit accurate (functional validation for logic elements)

SystemC

- set of C++ classes, macros and library used for
  - system-level modeling in hardware and software (co-design)
  - functional verification
  - event-driven simulation of complete systems
  - architectural exploration
  - performance modeling
  - high-level synthesis
- definition, library, documentation and links:
  - defined and promoted by OSCI: the Open SystemC Initiative
  - `http://www.systemc.org/`
  - IEEE Standard 1666-2005: Open SystemC Language Reference Manual
- arithmetic data types (warning: limited synthesis capabilities):
  - integers: $\leq$64 bits words or "arbitrary" precision (512 bits max.)
  - floating-point: IEEE-754 32 bits machine words (`float`)
  - fixed-point

SystemC Fixed-Point Support

Fixed-point representations:

- signed numbers (2's complement): `sc_fixed` or `sc_fix`
- unsigned numbers (radix 2): `sc_ufixed` or `sc_ufix`

Format for `sc_[u]fixed<`*wl*`,` *iwl*`,` *q_mode*`,` *o_mode*`,` *n_bits*`>`:

- *wl* total word length
- *iwl* integer word length
- *q_mode* quantization/rounding mode
- *o_mode* overflow/saturation mode
- *n_bits* number of saturation bits (0 as default)

Template parameters definition time:

static: `sc_fixed` and `sc_ufixed` at compile time

dynamic: `sc_fix` and `sc_ufix` at run time

SystemC Fixed-Point Rounding and Overflow Modes

Quantization/rounding modes (*q_mode* parameter):

- `SC_RND` rounding to $+\infty$
- `SC_RND_ZERO` rounding to 0
- `SC_RND_MIN_INF` rounding to $-\infty$
- `SC_RND_INF` rounding to ∞
- `SC_RND_CONV` convergent rounding
- `SC_TRN` truncation, default mode
- `SC_TRN_ZERO` truncation to 0

Overflow/saturation modes (*o_mode* parameter):

- `SC_SAT` saturation
- `SC_SAT_ZERO` saturation to 0
- `SC_SAT_SYM` symmetrical saturation
- `SC_WRAP` wrap-around, default mode
- `SC_WRAP_SM` sign magnitude wrap-around

Gappa Overview

- developed by Guillaume Melquiond
- goal: formal verification of the correctness of numerical programs:
  - software and hardware
  - integer, floating-point and fixed-point arithmetic ($\pm$, $\times$, $\div$, $\sqrt{}$)
- uses multiple-precision interval arithmetic, forward error analysis and expression rewriting to bound mathematical expressions (rounded and exact operators)
- generates a theorem and its proof which can be automatically checked using a proof assistant (e.g. Coq or HOL Light)
- reports tight error bounds for given expressions in a given domain
- C++ code and free software licence (CeCILL)
- publication: ACM Transactions on Mathematical Software [2]
- source code and documentation: `http://gappa.gforge.inria.fr/`

Gappa Example

Degree-2 polynomial approximation to e^x over $[1/2, 1]$ and format 1Q9:

```
p0 = 571/512;     p1 = 275/512;     p2 = 545/512;

x = fixed<-9,dn>(Mx);

y1 fixed<-9,dn>= p2 * x + p1;
p  fixed<-9,dn>= y1 * x + p0;

Mp = (p2 * Mx + p1) * Mx + p0;

{
    Mx in [0.5,1]    /\    |Mp-Mf| in [0,0.001385]
->
    |p-Mf| in [0,0.011]
}
```

Gappa-0.12.0 result ($[a, b]$, $x\{(\approx x)_{10}, \log_2 x\}$, $x$by$y = x2^y$):

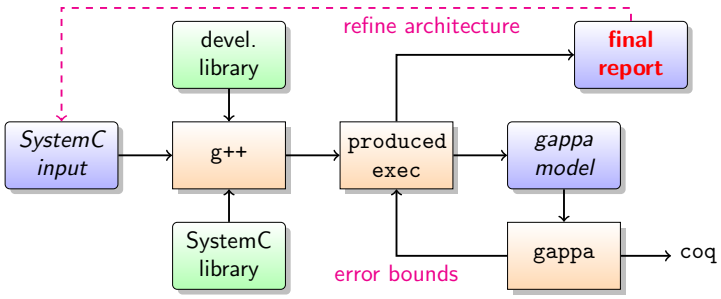
```
Results for Mx in [0.5, 1] and |Mp - Mf| in [0, 0.001385]:
|p - Mf| in [0, 81165739243220271b-66 {0.011, 2^(-6.50636)}]
```


Description of the Developed Library

SystemC (C++ with restricted template usage) library

- extension (templates) of fixed-point data types:
  - static types `sc_fixed` and `sc_ufixed` for validation mode
  - dynamic types `sc_fix` and `sc_ufix` for optimization mode
- overloading of fixed-point operations:
  - conversion to gappa model
  - arithmetic operations binary: $\pm, \times, \div, \sqrt{}$, unary: $-$
  - shifts: <<, >> (useful for scaling operations)
- add extra informations:
  - operand domain and format
  - approximation information
- specific functions:
  - generate complete gappa model
  - gappa external call
  - read gappa output
  - generate error bounding final report
  - optimization algorithms

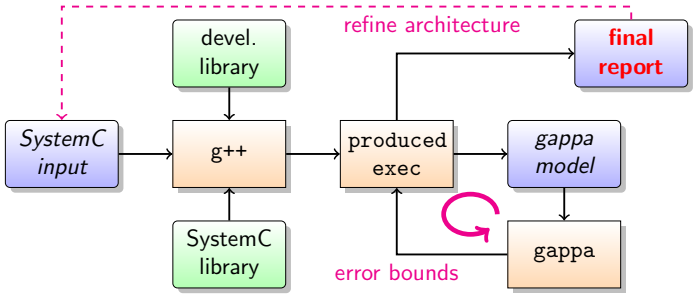
Validation Mode



Restricted to `sc_fixed` and `sc_ufixed` types

- format specification for all values translated into gappa description
- gappa computes error bounds
- verification

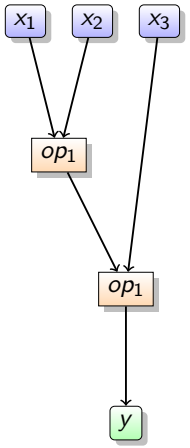
Optimization Mode



Restricted to `sc_fix` and `sc_ufix` types

- no format specification
- optimization process using multiple gappa calls

Optimization Heuristics



Initial formats assignment:

- propagate output accuracy backwards
- propagate input formats forwards (+ guard bits)

Optimization steps:

- exhaustive search
  limited to a few operations
- priority lists:
  1. large operators ($\times$)
  2. small operators ($\times$ by cst, $\pm$)
  3. almost free operators (shifts)
- accuracy "sensibility" analysis
  (locally reduce bitwidth and reevaluate accuracy)

Preliminary Experimental Results

- signal processing, multimedia and telecommunication in-house benchmarks
- FPGA implementation (Xilinx small Spartan 3 XC3S400A device, ISE 12.1 tools, medium efforts, area optimization target)
- power estimation based on simulations 10 000 random vectors (high activity)
- relative reduction compared to a standard manual optimization
- validation and optimization process time between 1 s and 2 minutes

circuit	delay	area	power
16b FIR	-5 %	-6 %	-9 %
24b FIR	-7 %	-15 %	-18 %
8p 16b IDCT	-3 %	-24 %	-31 %
LDPC decoder	-12 %	-21 %	-42 %

Conclusion & Future Prospects

Current status:

- SystemC library for fixed-point arithmetic with
  - computation of tight error bounds (for numerical validation)
  - operands width optimization under accuracy constraint (for area and power consumption reduction)
  - formal validation of the produced error bounds
- Limitations:
  - Only absolute error
  - No good optimization heuristics for $\div$ and $\sqrt{}$
  - No link with resource scheduling (several operations mapped on the same operator)

Future work:

- add advanced optimization schemes
- add power consumption constraints in the optimization process
- extension to other tools than gappa (e.g. gappa++)

References

J.-M. Chesneaux, L.-S. Didier, F. Jézéquel, J.-L. Lamotte, and F. Rico.
CADNA: Control of accuracy and debugging for numerical applications.
http://www-pequan.lip6.fr/cadna/.
LIP6–UPMC.

M. Daumas and G. Melquiond.
Certification of bounds on expressions involving rounded operators.
ACM Transactions on Mathematical Software, 37(1):2:1–20, January 2010.

M. D. Ercegovac and T. Lang.
Digital Arithmetic.
Morgan Kaufmann, 2003.

E. Goubault, M. Martel, and S. Putot.
FLUCTUAT: Static analysis for numerical precision.
http://www-list.cea.fr/labos/fr/LSL/fluctuat/index.html.
CEA–LIST.

D. Ménard and O. Sentieys.
Automatic evaluation of the accuracy of fixed-point algorithms.
In *Proc. Design, Automation and Test in Europe (DATE)*, pages 529–537, March 2002.

The Coq Development Team.
The Coq proof assistant.
http://coq.inria.fr/.
INRIA.

The end, some questions ?

Contact:

- mailto:arnaud.tisserand@irisa.fr
- http://www.irisa.fr/prive/Arnaud.Tisserand/
- CAIRN Group http://www.irisa.fr/cairn/
- IRISA Laboratory, CNRS–INRIA–Univ. Rennes 1
  6 rue Kérampont, BP 80518, F-22305 Lannion cedex, France

Thank you