HIGH-LEVEL DESIGN OF INTEGRATED MICROSYSTEMS - ARITHMETIC PERSPECTIVE

Zeljko Zilic

McGill University, Dept. ECE
Montréal, Québec, Canada

Boris Karajica

STMicroelectronics
Boston, Massachusets, USA

ABSTRACT

Integration of physical sensing and actuating interfaces into embedded systems presents a new set of high-level design and synthesis challenges. An important aspect of such integration is the explicit use of numerical values at the interfaces, so we propose the system design method focusing on the efficient and correct arithmetic-centric synthesis. Integrated precision and range analysis and optimization are detailed, together with the iterative arithmetic processing.

Index Terms— cyber-physical systems, microsystems, arithmetic optimization, static arithmetic analysis

I. INTRODUCTION

Cyber-physical systems are characterized by tighter integration of embedded computer systems and the physical sensing and actuating devices. A significant part of the appeal of cyber-physical systems is in the integration of heterogeneous subsystems, be it electronics, MEMS, microfluidics, and at the same time the functionality provided in increasingly sophisticated software. Such integrated microsystems will need to be designed and optimized following the design techniques that are suited for complete systems. For this reason, high-level design tools and methodologies will need to be developed that account for the new issues in interactions among the subsystems.

For instance, consider a robot including a MEMS-based accelerometer with a given precision tolerance, temperature dependence and frequency characteristics. There is a need to optimize interfaces and the embedded control, and to tailor them to compensate for the imperfections in physical sensors. In this scenario, we ought to optimize the overall system design, rather than apply a series of isolated steps that either insert the pessimism (and hence are inefficient) or are not adequate for the whole range of inputs and states of the systems (and hence are incorrect under some conditions).

In trying to undertake the high-level design and optimization of heterogeneous microsystems, we are necessarily bound to deal with the continuous-valued variables in the system, such as the analog outputs of the sensors. The issue we are facing here is that the physical processes are continuous in nature, both in time and the state-space, while the computing part is discrete-valued and executed in discrete time steps. Therefore, the required integration has to address salient features of interfaces between the discrete and continuous domains in providing optimization, verification, guaranteeing fidelity in dealing with physical processes as well as the overall quality measures in integrated system design. [1].

I-A. Arithmetic-centric Microsystem Synthesis

The methodologies that we explore deal with the efficient and correct handling of the numerical quantities used in integrated microsystems. The ”control plane” is assumed to be largely equivalent to that of the systems on chip. The explicit measures are undertaken to provide the end-to-end functionality guarantees for:

- Sufficient dynamic *range* of all quantities of interest, such that when digitized we employ the bit-widths that are both sufficient to avoid overflows and also efficiently use the area, energy and time for discretized computation. *Calibration* will also deal with range.
- Adequate *precision* of the results of computation across the integrated microsystem, given by some acceptable measures of the distance to the specified output, or, in other words, the fidelity of the system operation.
- Suitable *response in the frequency domain*, taking into account the behavior of the mechanical, chemical, electronic and the discrete-time control subsystems. Frequency response includes the requirements for the range and precision above.
- Interaction and *fusion* of multiple sources of data, including noise filtering. As the sampled data is bound to be noisy and might appear inconsistent, solid techniques need to be applied at the system level.

II. INTEGRATED MICROSYSTEM SYNTHESIS

An increasing range of system designs, from embedded systems to consumer electronics to FPGA-based prototypes now require tight integration of physical sensors and actuators. The design methodologies for such systems will need to be established, with the clear aim of achieving an overarching integrated system-level design. With the tight integration into physical processes, the specification of such systems is necessarily given in terms of the system dynamics

in continuous space and time, as well as in frequency domain. We illustrate the issues in such a system design using the example.

Example 1: Avionics tilt-compensated compass. Consider a system including a 3D accelerometer, magnetometer and gyroscope that is specified to compute the *tilt* angles α, β and γ in three dimensions, with the latter being the exact position of magnetic north, known as *heading*. The heading calculation should operate under normal and free-fall conditions, in which case the interrupt service routine (ISR) is required to calculate the heading direction, using an alternative algorithm that does not rely on the accelerometer. The system should be self-calibrated, and tilt angles calculated with the precision of 2 degrees. The acceleration and the magnetic signal noise in all three dimensions should be filtered above f_a and f_H cutoff frequencies, respectively. The three readings of acceleration a_x, a_y and a_z are used to obtain the Euler angles, relative to the three coordinate planes as:

$$\alpha = \arctan\left(\frac{a_x}{\sqrt{(a_y^2 + a_z^2)}}\right) \tag{1}$$

$$\beta = \arctan\left(\frac{a_y}{\sqrt{(a_x^2 + a_z^2)}}\right) \tag{2}$$

Since the obtained accelerometer readings do not suffice for obtaining the third angle, which is heading, as that information is invariant relative to the gravity force. For this angle, the magnetometer reading is used to obtain compensations in X and Y directions and the heading angle γ.

$$\begin{aligned} X' = &\quad X\cos(\alpha) - Y\sin(\beta)\sin(\alpha) + Z\cos(\beta)\sin(\alpha) \\ Y' = &\quad Y\cos(\beta) + Z\sin(\alpha) \\ \gamma = &\quad 90 - \arctan(\frac{X'}{Y'}), when \quad Y' \geq 0 \end{aligned}$$

We consider the integrated design of the whole system, and note that the MEMS sensors can already be easily added to the CMOS substrate performing arbitrary computing and communication functions. Companies such as ST Microelectronics routinely perform 3D integration of such systems including multiple discrete MEMS sensors, hence the system-level synthesis is a clear possibility.

Numerous discrete MEMS sensors are in existence that sense all quantities in Example 1. For example, the acceleration sensor measurements in the immobile state corresponds to the three projections of the gravity force to the coordinate axes. The magnetic module similarly measures the three projections of the magnetic field *H*. Figure 1 shows the details of the integrated geomagnetic module by ST Microelectronics, with 3 accelerometers and 3 magnetometers as well as the interface circuitry [2].

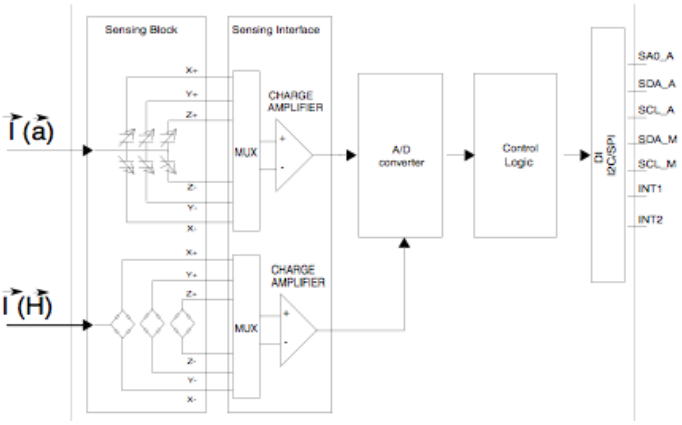

Fig. 1. Example of integrated geomagnetic sensor module in 3 dimensions: LSM303DLM by ST Microelectronics [2]

II-A. Correct and Precise Arithmetic Computation

We focus on the integration of numerical quantities in the synthesis of integrated microsystems. In Example 1, the main component of the system is that of calculating the tilt angles within given precision constraints, which entails the selection of the calculation algorithm, optimization of the word-lengths and avoidance of the incorrect cases, be it due to the range (overflow), corner cases or the noise.

To obtain the required precision, the system design needs to account for all the causes of imprecision across the mechanical parts ($e_{readout}$, $temp_{comp}$), analog electronics (e_{conv}) and the finite word-length digital processing parts.

$$Error_\alpha = e_{readout} + e_{conv} + \Delta t * temp_{comp} + e_{arithmetic}$$

The synthesis task then entails finding suitable word-lengths and possibly approximation algorithms for the last part $e_{arithmetic}$, such that the overall error bound is not exceeded:

$$Error_\alpha \leq \epsilon. \tag{3}$$

While doing this optimization, we also need to ensure the arithmetic correctness, including the avoidance of arithmetic overflows, conditions when denominator approaches zero and similar, which exceed the range of possible arithmetic representations.

II-B. Noise, Self-calibration and Data Fusion

The noise exhibits significant presence in the sensor data, and handling noisy data is critical to all applications of microsystems, ranging from the usable user interfaces in consumer electronics employing motion sensors, to the emerging cyber-physical systems. Considering Example 1, with specified cutoff frequencies for the accelerometer and magnetometer measurements, one can easily appreciate that any mechanical vibrations introduce unwanted noise in all three dimensions, hence the requirement to eliminate all

the sampled accelerometer signal above given frequency f_a. Similarly, magnetometers are constantly impacted by numerous artificially produced magnetic fields present everywhere, and the data readouts would appear wrong without proper filtering.

Often, sensor modules such as the one from Figure 1 already come equipped with digital filters. Further, the frequency characteristics of the sensor system might itself play a significant role, or a set of filters could be available as IP cores for synthesis. For all such scenarios, in order to integrate and filter the data, the system design would require the *fusion synthesis*, to cascade the newly synthesized modules with the existing transfer characteristics, to obtain the required frequency response, Figure 2. The problem could involve a solution in hardware, software or a combination, calling for the solution space exploration.

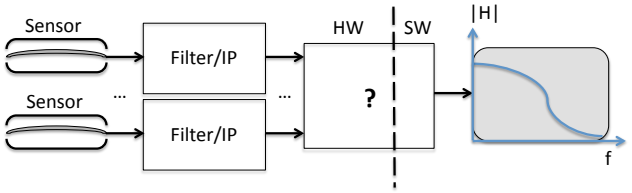

Fig. 2. Fusion synthesis: fitting frequency-domain response

The utility of multiple sensor systems is impossible without the noise-free integration of all sensed data, referred to as data fusion, whereas solutions such as Kalman filters are just one possibility, and in their implementation, the same word-length issues (precision, range) remain, but now are additionally applied to the iterative and non-linear computations. System validation, including the runtime correctness now involves the effects due to the physical device effects in sensors, and the self-calibration, temperature compensation etc. are all must in interfacing the sensors.

III. ARITHMETIC-CENTERED MICROSYSTEM SYNTHESIS

The arithmetic issues in high-level synthesis are long-standing [3], [4], [5], with the noticeable renewal in interest due to the proliferation of FPGAs as ASIC replacements, which force the use of word-length limited fixed-point representations.

Suitable results in arithmetic-driven high-level synthesis and verification include the methods for word optimizations under precision constraints [6] that employs Arithmetic Transform (AT) [7], [8]. For range analysis and optimization, there is the satisfiability modulo theory (SMT) approach[9].

III-A. Arithmetic Verification and Optimization with AT

AT is defined as a polynomial over pseudo-Boolean functions, $f : B^n \mapsto w$, where the output w is a word-level quantity, over which we can perform addition and subtraction [10].

Table I. AT Encoding for Common Arithmetic Functions

Word	Integer	Fixed-Point
Unsign.	$\sum_{i=0}^{n-1} x_i 2^i$	$\sum_{i=1}^{n-1} x_i 2^{i-m}$
Sign-ext.	$(1-2x_{n-1}) \sum_{i=0}^{n-1} x_i 2^i$	$(1-2x_0) \sum_{i=1}^{n-1} x_i 2^{i-m}$
2 Compl.	$\sum_{i=0}^{n-2} x_i 2^i - x_{n-1} 2^{n-1}$	$-x_0 2^{m-n} - \sum_{i=1}^{n-2} x_i 2^{i-m}$

$$f = \sum_{i_0=0}^{1} \sum_{i_1=0}^{1} \cdots \sum_{i_{n-1}=0}^{1} c_{i_0 i_1 \ldots i_{n-1}} x_0^{i_0} x_1^{i_1} \ldots x_{n-1}^{i_{n-1}}. \quad (4)$$

Hence, AT expresses a function using the set of linearly independent functions defined as: $x_0^{i_0} x_1^{i_1} \ldots x_{n-1}^{i_{n-1}}$, where

$$x_j^{i_j} = \begin{cases} x_j, & \text{if } i_j = 1 \\ 1, & \text{if } i_j = 0 \end{cases}, j = 0, \ldots, n-1.$$

We say that the *arithmetic spectrum* is a set of coefficients $c_{i_0 i_1 \ldots i_{n-1}}$, each of which multiplies an orthogonal basis function $x_0^{i_0} x_1^{i_1} \ldots x_{n-1}^{i_{n-1}}$. In the case of AT, the transform can be simply treated as a polynomial.

An important propertiy of AT is that it allows the outputs to be grouped into the word-level quantities, which allows us to directly reason about the arithmetic, including determining statically the precision and range values. Then, the *encoding* of word-level quantities determines the exact way the calculation is done, i.e., the meaning of the operations "+" and "−", to provide the "numerical" value . The common encodings are summarized in Table I. The same encoding can be easily expanded to describe more complex functions as shown next.

III-B. Relation to Real-valued Specifications

Arithmetic circuits commonly implement real-valued specifications of functions that are in general multivariate. For a number of real-valued specifications, including polynomials and Taylor series, there is an efficient construction of AT.

For function f over interval I around point X_0, the infinite Taylor series converging over I are:

$$f(X) = \sum_{i=1}^{\infty} \frac{f^{(i)}(X_0)}{i!} * (X - X_0)^i. \quad (5)$$

In practical implementations, the series become truncated to the first $n+1$ terms, in which case the remainder is provably bounded by an $(n+1^{st})$ order derivative:

$$R_n(X) = \frac{f^{(n+1)}(\xi)}{(n+1)!} * (X - X_0)^{n+1} \quad (6)$$

where ξ is a point in the given interval I. The finite Taylor expansion can be converted efficiently to an AT [11]. For an instance of m-bit fixed-point presentation the arithmetic

rewrite replaces all instances of the real-valued X with the encoding function over a m-tuple of bits. For instance, with unsigned encoding, $X_0 = 0$ and $f_0 = f(X_0)$, the AT for the $(n+1)$-term Taylor polynomial is:

$$f_0 + f_0^{'}\left(\sum_{i=0}^{m-1} x_j 2^j\right) + \cdots + \frac{f_0^{(n)}}{n!} * \left(\sum_{j=0}^{m-1} x_j 2^j\right)^n$$

The single most challenging issue in the conversion to AT is that of the intermediate polynomial swell during the transformation. The techniques for the efficient rewriting are well elaborated in, for instance, symbolic computing area.

Having the efficient construction of AT from real-valued specification, we proceed to the precision analysis, which will allow us to efficiently verify whether a fixed-point implementation achieves the required precision.

III-C. AT and Arithmetic Precision and Range

Implementations of real-valued functions are bound to be imprecise. First, there are the *approximations* applied to the real-valued specification, and the *imprecision* is based on a finite-word implementation.

The *imprecision error* is a distance between the specification and implementation, most commonly measured as the maximal absolute difference between the two, i.e., the uniform norm L_∞. The error is commonly expressed in the terms of the *units in the last place* (ULP), i.e., relative to the length of the fixed-point implementation.

The precision analysis cannot be dealt with at the bit-level, and the explicit representation of output word-level values is needed. Consider an example of computation where all output bits are incorrect, while the imprecision can be made arbitrarily small by extending the wordlength. For example, if the exact n-bit fractional result is $1.00\ldots0$,, while the approximation is $0.11\ldots1$., then all bits are incorrect, but the error can be made arbitrarily small by increasing n.

Since AT represents the function output at a word level, it can be used for reasoning directly about the imprecision. For a given reference implementation f_{ref}, the imprecision due to the n-bit wordlength is then directly expressed as

$$AT(f_{ref}) - AT(f_n).$$

Hence, the imprecision to the finite wordlength n is expressed in terms of maximum absolute distance in L_∞ as

$$||AT(f_{ref}) - AT(f_n)||_\infty = ||AT(f_{ref} - f_n)||_\infty. \quad (7)$$

Then, it suffices to search for a maximum absolute value that the AT of a difference between the reference and the implementation functions take via an branch-and-bound search [12]. Furthermore, the other distance functions, like the sum of squares could be equally applied as well.

To verify an implementation consisting of arithmetic values passed by multiple sensors, we construct the AT and evaluate whether the imprecision from Eqn. 7 is acceptable, i.e., smaller than a given bound ε. The problem is given as:

Problem 1 PRECISION VERIFICATION
Require: $f_{ref}, f_{impl}, \varepsilon$
Ensure: TRUE *iff* $||AT(f_{ref}) - AT(f_{impl})||_\infty < \varepsilon$

This formulation is also practical if a reference f_{ref} by itself is imprecise within a bound, say δ. For instance, the transcendental functions can only have the approximate reference implementations, such as with Taylor series. Then, we can apply the triangle inequality

$$||AT(f_{abs}) - AT(f_{impl})||_\infty \leq ||AT(f_{abs}) - AT(f_{ref})||_\infty + ||AT(f_{ref}) - AT(f_{impl})||_\infty$$

among the precise f_{abs}, the reference and the implementation, to guarantee that $|\varepsilon + \delta|$ is an acceptable imprecision.

Hence, AT facilitates a completely *static verification* scheme precision and range constraints. Towards that goal, a branch-and-bound search in [12] finds whether the worst case imprecision of a fixed-point implementation is smaller than the allowed imprecision. The exact search for the maximal imprecision is reduced to a search for a maximal value that an AT polynomial takes, as per Eqn. 7.

For range analysis, it suffices to find the minimal and maximal value of the AT, which is directly achieved by the same search algorithm applied to the implementation polynomial.

III-D. Component Matching and Fusion Synthesis

Another task in synthesis is that of finding the most suitable library components for imprecise arithmetic, including the transductor synthesis, Figure II-B. The component matching has surfaced as a problem in conjunction with IP library reuse. The work such as [13] has shown the need for multivariate polynomials real-valued polynomials for the matching of real-valued functions. With AT, the transition to multivariate real-valued polynomial specifications is straightforward. Further, the search for a suitable library component from a given library lib is reduced to the problem PRECISION VERIFICATION, applied to all the elements of lib.

III-E. Optimization of Fixed-point Arithmetic

We show next how to optimize the fixed-point implementations of integrated microsystems. Starting from a polynomial or Taylor series description, the algorithm automatically selects the multiple approximation and imprecision parameters by a scheme that is essentially branch-and-bound, with a number of optimizations specific to this case.

While this approach could be applied to a variety of real-valued representations, we present the main ideas applied to the case of finite Taylor expansion. The optimization problem is defined as that of finding the parameters such

as the number of Taylor terms, n, and the wordlength m of the input representation, such that the cost is minimal.

Problem 2 TAYLOR PRECISION OPTIMIZATION
Require: f_{ref}, ε
Ensure: $||AT(f_{ref}) - AT(f_{n,m})||_\infty < \varepsilon$
Goal: $m,\ n\ s.t.\ min\ terms(AT(f_{n,m}))$

Consider now the number of AT terms of an expanded nth order Taylor polynomial over m-bit fixed point numbers as a cost function in the optimization procedure. By enumeration, it can be shown that the number of AT terms is then equal to

$$cost = terms(AT(f_{n,m})) = \sum_{i=1}^{n} \binom{m}{i}$$

In this case, the cost function exhibits the same monotonicity in the two variables. By using this cost function, we will effectively be searching for a minimal size AT polynomial implementing a function within a given precision.

Search guidance - sensitivity Similar to the previous instance of branch-and-bound, providing a search heuristic is a significant way to speed up the algorithm. In this case, we can apply the domain-specific knowledge about the *sensitivity* of a function relative to the change in either n or m.

Sensitivity of function f with respect to variable x is defined as a derivative of a function $\frac{df}{dx}$. Here, variables n and m are discrete, and need to be mapped to continuous variables that they present.

In the case of m, the wordlength, an increase of one bit results in the added shift of half an ulp: $\Delta X = 2^{-(m+1)}$. Then, the impact of the wordlength change to the function is

$$\Delta f = \frac{df}{dX}\Delta X = \frac{df}{dX}2^{-(m+1)}.$$

Furthermore, by recalling that the first Taylor term is a derivative at the expansion point X_0, the sensitivity is calculated as an $m+1$-fold shift of the first coefficient. Since this is true only around X_0, a more accurate sensitivity is obtained by differentiating the function. AT can be used here as well, to easily differentiate the given function. By finding the maximal value over the interval, as in the precision search, the worst case influence of the variable change is obtained.

In the case of sensitivity to n, we apply the error bound from Eqn. 6. When a new n is to be selected in the search, the difference between the two remainder functions, bounding approximation error, can be easily computed. Further, when going, say, from n to $n+1$ terms, the function difference is

$$\Delta f = Taylor_{n+1}(f) - Taylor_n(f) + R_n(f) - R_{n+1}(f)$$

The algorithm takes into account in an unified way all the approximation and imprecision sources for real-valued specifications such as Taylor series, where only the parameters n and m are to be chosen. The pseudo-code is given in Algorithm 3. It starts with the number of terms given by Eqn. 6 and then explores m and n according to the sensitivity.

Algorithm 3 OPTIMIZE (f, ε)

```
1:  lb = MINTAYLORTERMS
2:  CurrentBest = lb; Current = ub;
3:  MIN_AREA (AT(f), Current)
4:  {
5:  if UnexploredPosibilities then
6:     direction = FINDMOSTSENSITIVEVAR
7:     if (direction ==m) then
8:        Current = MIN_AREA (AT(f_m), Current)
9:     else
10:       Current = MIN_AREA (AT(f_n), Current)
11:    end if
12: else
13:    Current = AT;
14:    CurrentBest = |min(Current, CurrentBest)|
15:    return CurrentBest
16: end if
17: }
```

This algorithm allows us to perform design space exploration of microsystems, which are arithmetic-intensive by finding the precise implementation within given approximation schemes (Taylor, polynomial, lookup table, etc.).

IV. EXPERIMENTAL RESULTS

We have undertaken the design space exploration and arithmetic optimization of the system outlined in Example 1, as carried out in the project [14]. For each of its subsystems, multiple designs were completed and the results were compared towards finding the most suitable solution targeting the iNEMO board [2]. Because of the presence of the ARM Cortex M3 processor running at 72MHz, we showcase for comparison here three obvious candidates suited for the software implementations, leaving out the CORDIC and other schemes suited more for hardware.

First, the implementations of the arctan function and the whole block for computing the three angles in Eqn. 1 were explored for ARM Cortex software implementation. The times it takes to execute the function for the best and worst case, as reported by Keil ARM development system, are reported in Table II. The code developed by the existing C library is included for comparison. The table lookup method requires 400 bytes of on-chip FLASH memory and it is precise within 1.15^o. The approximation property, Eqn. 3 for achieving the overall specified precision was verified using AT static analysis from [6] in negligible time.

Table II. Exploration of Sufficiently Precise arctan function

Input Case	Pol. Appr.	LUT	C library
Best	84.4 μs	14.6 μs	305.3 μs
Worst	103 μs	22.8 μs	311.6 μs

Table III. Filters/transductors Meeting Spec

Max Rate	1^{st} Order/50	3^{rd} Order/100	1^{st} Recur/50
Sample	2.4kHz	1.96 kHz	2.4 kHz
Data	62.3 Hz	124.6 Hz	58 Hz
Output	31.1 Hz	31.1 Hz	31.1 Hz

Next, the design exploration of the fusion/filter was conducted, taking into account that the sensors already come equipped with a lowpass (and highpass, which is in this case turned off) filter. In this case, the lowest cutoff rate for the lowpass filter is 37Hz, so the frequency-domain fusion/transductor needs to be inserted to merge and filter further to f_a =12 Hz. Among the range of solutions explored within the project, we show here non-recursive and recursive first order mean filter and third order mean filter, sampled at 50 or 100 Hz. Its performance in the ARM Cortex code for the target output rate at 31 Hz is given in Table III.

The overall performance is obtained for the selection of the LUT-based arctan calculation and the third order filter, as reported in Table IV. Here, the specification of the sampling rates were given (except that the free-fall interrupts were processed asynchronously), the actual processing time includes the data transfer through all hardware interfaces (such as I^2C for accelerometer and magnetometer) and the code performing the readouts, filtering, fusion and overall processing.

V. CONCLUSIONS AND FUTURE WORK

We demonstrate that the AT allows static analysis and optimization for realizing full-scale synthesis of multi-sensor systems with the emphasis on the numerical values passed between sensors and the rest of the system, and their correct and efficient processing. We have identified the key steps for the static analysis for precision, range and iterative arithmetic computation based on AT. We will consider fault-tolerant design [15] of multisensor systems.

Table IV. System Performance

Operation	Time [μs]	Rate [Hz]
Accel. Readout	2761	100
Magnet. Readout	1250	100
Filter accel.	35	25
Filter magn.	75	25
Free-fall ISR	280	async
Free-fall filter	452	25

VI. REFERENCES

[1] S. Bourduas, J.S. Chenard, and Z. Zilic, "A quality-driven design approach for NoCs," *IEEE Design & Test of Computers*, pp. 416–428, 2008.

[2] ST Microelectronics, "iNEMO: Inertial module board," *www.st.com/inemo*, 2010.

[3] C. Shi and R. Brodersen, "Automated fixed-point data type optimization tool for signal processing and communication systems," *Proc. Design Automation Conference*, pp. 478–483, June 2004.

[4] S. Kim and W. Sung, "Combined word-length optimization and high level synthesis of digital signal processing systems," *IEEE Transactions on CAD*, vol. 20, no. 8, pp. 921–930, August 2001.

[5] C. Fang, R. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," *Proc. ICCAD*, pp. 275–282, November 2003.

[6] Y. Pang, K. Radecka, and Z. Zilic, "Optimization of imprecise circuits represented by Taylor series and real-valued polynomials," *IEEE Transactions on CAD*, vol. 29, pp. 1179–1190, August 2010.

[7] K. Radecka and Z. Zilic, "Using arithmetic transform for verification of datapath circuits via error modeling," in *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*. IEEE, 2000, pp. 271–277.

[8] Z. Zilic and Z.G. Vranesic, "A multiple-valued Reed-Muller transform for incompletely specified functions," *Computers, IEEE Transactions on*, vol. 44, no. 8, pp. 1012–1020, 1995.

[9] A. Kinsman and N. Nicolici, "Bit-width allocation for hardware accelerators for scientific computing using sat-modulo theory," *IEEE Transactions on CAD*, vol. 29, pp. 405–413, March 2010.

[10] W. G. Schneeweiss, "On the polynomial form of Boolean functions: Derivations and applications," *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 217–221, February 1998.

[11] K. Radecka and Z. Zilic, "Arithmetic transforms for compositions of sequential and imprecise datapaths," *IEEE Transactions on CAD*, vol. 25, no. 7, pp. 1382–1391, July 2006.

[12] K. Radecka and A. Zilic, "Specifying and verifying imprecise sequential datapaths by arithmetic transforms," *Proc. ICCAD*, pp. 128–131, November 2002.

[13] K. Radecka and Z. Zilic, "Design verification by test vectors and arithmetic transform universal test set," *Computers, IEEE Transactions on*, vol. 53, no. 5, pp. 628–640, 2004.

[14] McGill University, "ECSE 420 - microprocessor systems," *Class Project*, 2011.

[15] Z. Zilic and K. Radecka, "Fault tolerant glucose sensor readout and recalibration," *Proceedings of Wireless Health, WH 2011*, 2011.