IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 21, 2015, accepted February 25, 2016, date of publication March 7, 2016, date of current version June 24, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2537983

Intrinsic Evolution of Truncated Puiseux Series on a Mixed-Signal Field-Programmable SoC

VIGNESH THANGAVEL[1], ZI-XIA SONG[2], AND RONALD F. DEMARA[1], (Senior Member, IEEE)
[1]Computer Architecture Laboratory, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816-2362, USA
[2]Department of Mathematics, University of Central Florida, Orlando, FL 32816-1364, USA

Corresponding author: R. F. DeMara (ronald.demara@ucf.edu)

ABSTRACT Mixed-signal system-on-chip (SoC) devices offer single-chip solutions, but face challenges of hardware-software co-design optimization, device signal range constraints, and limited precision. These issues are addressed by developing a multi-level evolutionary approach to realize complex computational circuits called Embedded-Cascaded Hierarchically Evolved Logic Output Networks (ECHELON). The ECHELON technique utilizes analog evolved building blocks and refines their output using digital fabric to compose power series expansions of transcendental functions which are all routed under intrinsic control on a field-programmable SoC (PSoC). The result for the evolution of seven different powers of the independent variable is a reduction of 31.24% in the overall error as compared to the analog circuits that produce the raw inputs to a differential digital correction phase. Computation blocks developed on a Cypress PSoC-5LP mixed-signal SoC reduced error in the final mathematical approximation to the range of 40–150 mV. In doing so, speedups of roughly 1.4-fold to 6.6-fold with an average of 2.72-fold reduction in function execution times were attained. In particular, this approach achieved a 41.7-fold reduction in error with respect to the largest power of the independent variable used as an input to compute an *erf(x)* function.

INDEX TERMS Programmable system on chip (PSoC), power series, genetic algorithm, programmable logic device.

I. INTRODUCTION

Cooperative analog-digital signal processing techniques outlined in [1] have recently gained recognition for their ability to address problems in both domains by drawing out their complementary computational characteristics. As depicted in Fig.1, the advent of the SoC era is characterized by embedded computing architectures with support to perform a wide range of computations, autonomously under real-time constraints. SoC architectures addressed in this paper utilize a heterogeneous fabric of mixed-signal based circuitry along with dedicated Computing Elements (CEs) realized from the more generic analog and digital subsystems. SoCs face challenges of sparser reconfigurable fabrics, limited memory capacities, and need for area-efficient design. On the other hand, they offer the mixed-signal advantage where analog computations manipulate data in continuous ranges very cost-effectively and interface well with real world data, while digital computations offer great scalability in performing accurate computations. These CEs are operationally distinct from the more specially designed accelerators and coprocessors that perform special operations such as error detection and correction, cryptographic functions, audio/video codecs, which require ASIC design methodology and cannot be modified readily at runtime. The CEs developed herein utilize reconfigurable analog and digital fabric components only and are driven by the rationale of relying on the underlying adaptive algorithm developed in this work to perform various mathematically intensive computations while minimizing the additional hardware required. The specific objectives herein, are to autonomously organize these heterogeneous resources to attain custom CEs for a range of transcendental mathematical functions with tunable accuracies.

Contemporary high performance embedded computing applications rely on heterogeneous multiprocessor SoC (MPSoCs) utilizing specialized units and accelerators for realization of different mathematical functions. In response, Tabkhi et al. [2] recognize the need for simpler mechanisms to avoid instruction bandwidth and memory bottlenecks resulting from Hardware Accelerators and Coarse Grain Reconfigurable Architectures. They propose the addition of Function-Level Processor (FLP) instead, requiring a separate Functional Set Architecture and extra buffers/cache

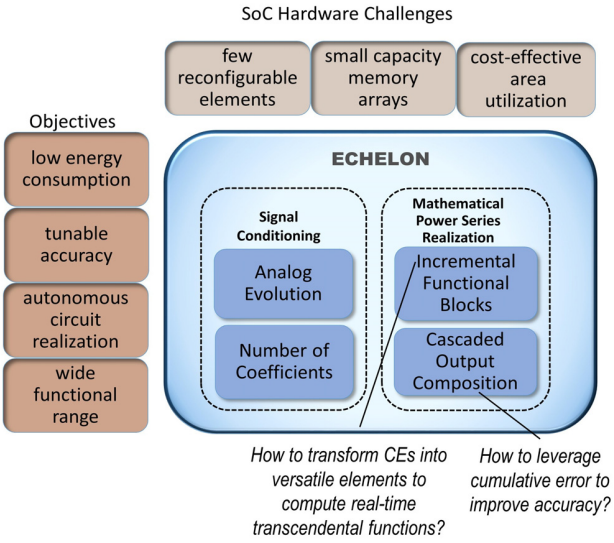

FIGURE 1. Meeting mixed-signal challenges with ECHELON.

per stage. Such specialized FLPs realize complex computations through "functional wiring" of elements designed in hardware. However, such approaches can add overheads to design time effort and execution speed, which ECHELON seeks to avoid.

II. ECHELON AND RELATED WORK

A time and area sparing approach called ECHELON or *Embedded-Cascaded Hierarchically-Evolved Logic Output Network,* is developed herein, which computes special mathematical functions by cascading incremental functionality of reconfigurable analog and digital blocks. ECHELON relies on evolutionary algorithms to search the design space based on the intrinsic switching behavior of the target circuit. This method to realize CEs extends the work done in [3] and [4] to exploit the interaction between general purpose analog and digital reconfigurable fabrics under the control of an embedded ARM core. The analog computation engine evolves analog reconfigurable fabric composed of switched capacitor op-amp blocks which may be configured to realize various amplifier, mixer and modulator topologies. This circuit produces analog signals roughly approximating the mathematical function desired and performs intelligent operations with self-scaling genetic algorithm (SSGA) to compress outputs beyond the device ADC range. It realizes a more sophisticated elaboration of hybrid analog and digital computation identified in [3] whereby an unrefined analog GA evolves a coarse solution in a narrow voltage range. Next, the output is scaled to allow a more computationally-tractable range. The entire set of operations performed up-to this stage is referred to here as *analog pre-processing*. This output is then converted to digital signals which are then adaptively refined by PLD-based digital fabric and combined in cascaded stages with digitally-evolved weights, using techniques for spatial self-adaptation of digital fabrics [5], [6], [9].

TABLE 1. Motivation for ECHELON and comparison.

Researcher/Name of Work Done	EA Type	Major Contributing Idea	Incorporation in Current Work
Ando 2003 [10]	Extrinsic	Multi-stage evolution with increasing evolutionary pressure	Cascaded stages for evolution of powers and pressure on coefficient prediction
Kazadi 2001 [9]	Extrinsic	Piece-wise evolution of simple parts	Evolution of mathematical building blocks
Haddow 2011 [11]	Extrinsic	Divide-and-conquer strategy to simplify fitness evaluation	Separate fitness functions for each component
Mitchell 1992 [12]	Extrinsic	Smallest evolving blocks preserve genetic diversity	Each power and its associated coefficient evolved independently
ECHELON	***Intrinsic***	***Range Adaptive Evolution***	***N/A***

Most recent embedded architectures rely on specialized hardware based computation units and a combination of lookup tables and series expansions to improve accuracy. These techniques typically consume extra power and may increase computation time to produce an arbitrary level of accuracy determined at design time for the hardware. Work done in [7] analyzes Worst Case Execution Time (WCET) and resource demands for various iterative and lookup table based methods, concluding that iterative schemes based on power series can provide a better choice for resource constrained hardware, but naturally reduce accuracy. On the other hand, pure software-based implementations of iterative techniques such as CORDIC require minimal hardware but large computation times as elaborated in [8]. Table 1 indicates some major research whose key ideas were useful in formulating the approach developed here. ECHELON extends the idea of decomposing a single complex problem into viable and simple connected parts as outlined by Kazadi *et al.* [9], and others. It uses multi-stage evolution popularized by Ando in [10] with increased pressure on the last stage and deploys the divide-and-conquer strategy outlined by Haddow in [11] to simplify fitness evaluations in each stage. Finally, it combines them at the finest granularity as suggested in work done by Mitchell in [12] on Royal Road functions.

Herein, the realization of specialized functions as a mathematical power series occurs by evolving the constituent powers with analog pre-processing followed by digital refinement to serve as universal building blocks. As depicted in the lower right corner of Fig. 1, these are combined to leverage cumulative error to improve accuracy.

Research contributions of this work include:

1. optimization of digital reconfigurable logic fabrics capable of tuning analog circuit outputs for mathematical computations,
2. adaptive prediction of power-series coefficients via dynamic allocation of programmable logic resources, and
3. an approach to partially decouple the computational complexity from sophistication of the mathematical function being expressed.

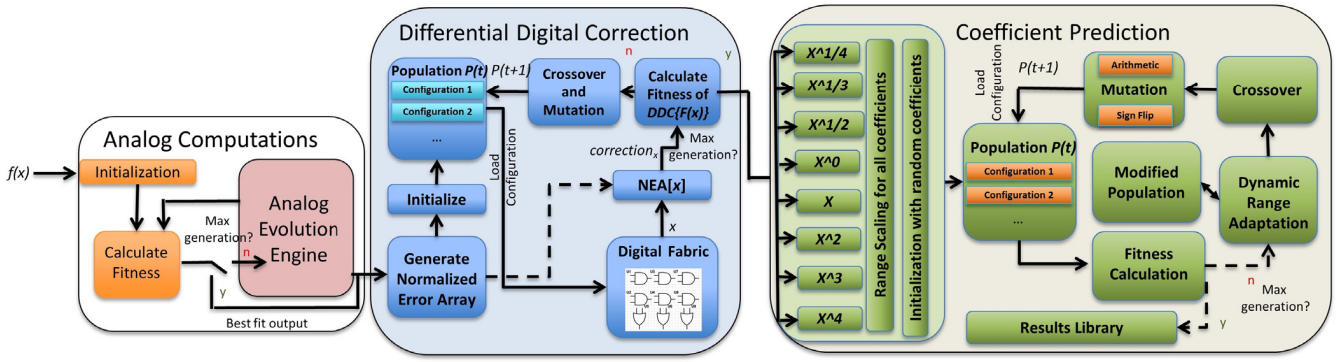

FIGURE 2. Schematic of ECHELON technique. The two stages of ECHELON are shown in blue and green colors, receiving analog pre-processed inputs.

III. PUISEUX SERIES AND COMPUTATIONAL COMPLEXITY

Power series based approaches are frequently used to approximate and compute functions to arbitrary accuracies around a particular point. They usually involve positive, integral powers of the independent variable. Puiseux or Newton-Puiseux series are a generalization of power series allowing for fractional (and negative) powers of the independent variable [13]. Mathematically speaking, it is well-known that any polynomial equation $f(x,y) = 0$ has $deg_y(f)$ zeros being Puiseux series in x, i.e., $y(x) = \sum_{k=k0}^{\infty} (y_k \times x^{k/n})$, for suitable integers $n \geq 1$, k_0, and the coefficients y_k from an algebraically closed field, where $deg_y(f)$ is the highest degree of y in f. Usually, Puiseux series expansions are used to describe the behavior of a curve near singularities with mathematically elaborate techniques to compute the corresponding coefficients. Another popular approach to approximating functions to better accuracies, albeit at the cost of greater computation time and added complexity, is known to be that of the computation of Padé approximants, which are rational functions whose power series agree with the Taylor series of the function being approximated up to a certain number of terms. The different approaches to mathematically approximate a curve at a given point, are elaborated in [14].

While power series are not new to numerical computational techniques, the use of Puiseux series has rarely been attempted in embedded computations due to the need to determine coefficients and associated complexities. Typically transcendental functions such as trigonometric, exponential and hyperbolic functions and their combinations are computationally more expensive, requiring significantly more clock cycles than simpler arithmetic operations, and are implemented with a combination of lookup tables and interpolation technique. Composition of such functions are prohibitively costly requiring even more time and resources in real-time systems. Such interpolation techniques achieve high accuracies at each point of interest, but require several iterations and hence more computation to achieve the same for a large number of points over a range while also requiring several lookup table memory accesses. They may increase computation time to achieve the desired accuracy or better in many computations.

In this work, a Puiseux series inspired approach is extended to quickly approximate a function to an acceptable error tolerance over an entire input range as determined by the device characteristics. The number of powers used in the approximation usually depends on the accuracy desired at a given point of interest, ''near'' which the function is being approximated. Usually, increasing the number of powers involved can improve the approximation that is obtained. However, calculation of greater number of powers takes more computational resources and/or time. When the same is extended to several points on a relatively wide range, the computation time required increases rapidly. In order to compute an expansion, the powers used are to be computed accurately followed by storage and retrieval of predetermined coefficients. However, if the powers computed are subject to several errors, the coefficients no longer serve their purpose and determination of coefficients becomes a non-trivial concern. Intuitively, a combination of convex and concave curves is expected to better balance the approximation process and hence eight powers namely $x^{(1/4)}$, $x^{(1/3)}$, $x^{(1/2)}$, x^0, x, x^2, x^3 and x^4 have been used largely in this work. Evolution of $x^{(1/5)}$, $x^{(1/6)}$ have also been attempted and $x^{(1/5)}$ was used to replace x^4 in some cases. This selection of powers is unconventional and lacks a set of pre-determined coefficients to guide the process. Herein, we consider a running example of evaluation of $sin(x)$ over the desired range approximated using truncated Puiseux series using a hybrid analog-digital reconfigurable fabric.

IV. ECHELON TECHNIQUE ON PSoC

Cypress Semiconductors PSoC devices versions 3 and higher contain a fabric of Configurable Analog Blocks (CABs), Universal Digital Blocks (UDBs), ARM Cortex-M cores, flexible GPIOs and serial communication blocks and peripherals for interfacing as outlined in [15] and [16]. The PSoC 5LP is a low power model of Cypress' most recent design with 32-bit ARM Cortex-M3, 256 KB flash and 64 KB SRAM as used in developing and testing ECHELON. Different processing stages of ECHELON are shown in Fig. 2. The analog

evolution engine detailed in [3] produces approximations that serve as pre-processed inputs to the digital fabric, but exhibit low accuracy and precision across the input range. The device ADC has a range of 0-4.08V. Starting from 0V, there are 256 data points or values of the independent variable x, increasing in steps of 0.016V. Differential Digital Correction (DDC) evolves PLDs to approximate various powers of x. The Coefficient Prediction (CP) stage evolves coefficients of the powers of x indicated to approximate the function sought to the desired accuracy. In this work, a tunable tradeoff of execution time versus accuracy is sought. Accuracy is considered to be valuable over the entire device range. Thus, the metric for accuracy selected is the summation of absolute error of the computed output using increments of 16 mV over the operating range, which is denoted herein as the *Total Error*. A design goal of 10 percent Total Error is sought and up to 15 percent Total Error is considered acceptable.

For instance, to approximate *sin(x)*, 8 different powers are evolved first and then the corresponding coefficients are determined corresponding to each power from fourth root to fourth power. To evolve the mixed signal circuit for each power, analog evolution is first performed in order to obtain a rough approximation, where the output may saturate if it is beyond the device limit of 4.08V. This is followed by scaled analog evolution to handle and compress outputs beyond device limit. Consider the evolution of x^3. As shown in Fig. 3, uncompensated analog evolution produces unsaturated outputs for x where $f(x) \leq 4.08$V. The outputs for the values of x larger than this saturate at 4.08V as indicated by the horizontal line in Fig. 3. Scaled Analog Evolution then proceeds to optimize saturated outputs beyond device range to produce a better approximation of x^3. The maximum deviation of this approximation from the oracle is extracted. This is then followed by Differential Digital Correction which uses fractions of this maximum deviation to refine the output over the device range for x^3 as in Fig. 4 and then proceeds to evolve other powers likewise. The frequencies of correction factors of different magnitudes produced on-demand for x^3 in different ranges are indicated in blue in Fig. 4. The impact of these stages are as indicated in Fig. 3. This then proceeds to the Coefficient Prediction stage in order to evolve coefficients of the powers evolved, to approximate *sin(x)*.

The CP stage first considers issues of data overflow and hence implements range scaling to ensure that these coefficients are randomly initialized within appropriate value ranges. Once initialized, CP Genetic Algorithm (CPGA) performs fitness evaluation over 256 data points starting from 0 and incremented in steps of 0.016 for the independent variable x corresponding to the native range for analog pre-processed inputs. ECHELON can refine analog preprocessed data for any arbitrary range and then use the same to compute functions sought for those ranges. Evolution proceeds till the termination limit of 1,000 generations and may also be configured to proceed until the desired accuracy is reached. Evolution starts with creation of a population of 80 random

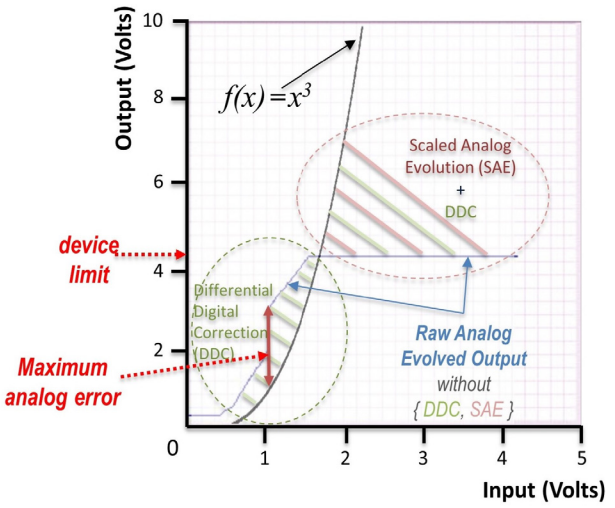

FIGURE 3. Example of evolution of cube of independent variable and how analog and digital correction improve accuracy in respective regions of operation [3].

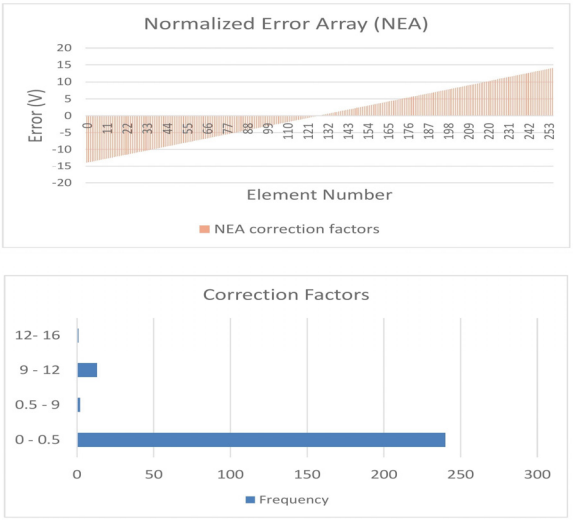

FIGURE 4. Elements of the NEA or normalized differences appearing as fractions of the maximum analog error (above) and correction factors evolved (below) for each point on x^3 indexed by D[i] by DDC.

individuals, fitness evaluation and identification of elites. Binary tournament selection, crossover between one individual from the fitter half and another randomly selected from the population are then performed. The number of individuals replaced is set at 60, but can be varied. Mutation is performed on these individuals and the steps starting from fitness evaluation are repeated again. The best-performing circuit configuration in each generation is retained without undergoing mutation. Once the termination condition is attained, the coefficients of the best fit individual are recorded and the function computed by this individual is evaluated against the originally-computed oracle to determine Total error. The corresponding *execution time* is recorded.

After receiving the analog pre-processed inputs over the native data range from the analog evolution phase,

digital refinement starts by application of DDC to these inputs. DDC constructs a Normalized Error Array (NEA), which contains fractions of the maximum error produced in the analog computation data. These elements are both positively and negatively valued to accommodate all possible errors and provide to correct them. The NEA containing correction factor elements called normalized differences, is indexed by *D[i]* at the *i-th* point, to realize the 8-bit output mapped to one of the 256 values of correction factors for the corresponding data input. The DDC fitness is evaluated as:

$$for\ i := 0\ to\ 255$$
$$fitness\ +\!= |oracle\,[i] - |\left(out_{analog}\,[i] - NEA\,[D[i]]\right)||;$$

This calculates Total Error across all data points for each individual and evolution proceeds to minimize the same.

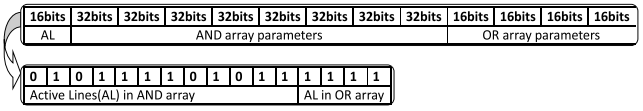

FIGURE 5. Chromosome used in DDC. Each chromosome controls a pair of PLDs in a UDB and two such chromosomes are evolved in parallel [3].

A. DDCGA GENOME AND OPERATION

Each pair of PLDs belonging to the same PSoC Universal Digital Block (UDB) is encoded with the genome indicated in Fig. 5. The bits indicate which product terms are asserted in PLD's AND arrays and OR arrays respectively as elaborated in [3]. The *Differential Digital Correction GA (DDCGA)* evolves the 4 PLDs or two UDBs in parallel to produce more accurate computations of different powers of the independent variable, x, through optimized use of correction factors to minimize Total Error as discussed above. Evolution proceeds in a cascade of stages, producing the output for each power of x desired. In this work, powers of x evolved include $\{x^{1/4}, x^{1/3}, x^{1/2}, x^0, x, x^2, x^3, x^4\}$ which were used in further computations. Other powers evolved include $x^{1/5}$ and $x^{1/6}$ from a different set of pre-processed inputs and some computations were attempted with them too. DDCGA is characterized by single-point crossover, binary tournament selection with a variable replacement rate typically around 60, elitism of two, bit mutation at 0.1% rate. Variable threshold parameters are used to ascertain stasis based on the difference between the average and best fitness. To mitigate stasis, 0.01% increment in mutation rate and hyper-mutation at variable intervals are implemented.

B. CPGA GENOME AND OPERATION

Following evolution of powers of x in cascaded stages, *Coefficient Prediction (CP)* algorithm evolves coefficients for these powers to form a truncated Puiseux series of the function sought, for instance, *sin(x)*. The genome for *sin(x)* is indicated in Fig. 6. Coefficients *C0* through *C7* are floating-point values. However, each is multiplied by the corresponding power of x to calculate the function at every evaluation point.

x^1/4	x^1/3	x^1/2	1	x	x^2	x^3	x^4
C0	C1	C2	C3	C4	C5	C6	C7

FIGURE 6. Chromosome used in CP. Each coefficient C0 through C7 is a 32-bit floating point value corresponding to the power of x indicated above it. The zeroth power term has a stabilizing effect on the computations [4].

Coefficient prediction involves an eight dimensional search space, and it is necessary to determine the largest allowable coefficient or range limit to prevent overflow in subsequent computations.

To illustrate, consider the calculation of *sin(x)* where each power x^k has been evaluated at every x:

$$sin(x) = \boldsymbol{C0}\cdot x^{(1/4)} + \boldsymbol{C1}\cdot x^{(1/3)} + \boldsymbol{C2}\cdot x^{(1/2)} + \boldsymbol{C3} + \boldsymbol{C4}\cdot x + \boldsymbol{C5}\cdot x^2 + \boldsymbol{C6}\cdot x^3 + \boldsymbol{C7}\cdot x^4$$

For the above calculation, the sum of products at each x must be a value within the range of a 32-bit floating-point number. *Range Scaling* is performed in order to ensure that all coefficients have the same scaled range and that overflow does not occur in the sum of products computed. Coefficients are then initialized randomly within this scaled range.

Algorithm 1 defines the CGPA used to evolve coefficients to minimize the Total Error in the functional approximation constructed using the imperfect building blocks produced. *Dynamic Range Adaptation* technique reduces the range for random assignment of coefficients, which otherwise become intractable to conventional GA operators such as crossover and mutation. *Crossover* is implemented as exchange of coefficients about a point chosen randomly on the chromosome and entails binary tournament selection with a replacement rate of 60 per generation which could be varied as a parameter. *Mutation* is implemented in two versions denoted **Q** and **S** with the **Mutation_Q** performed at a rate of 0.1% as addition of a number between −4 and 4, while **Mutation_S** induces a sign inversion at the same rate randomly for 10 out of 80 individuals. To deal with potential stasis conditions, the mutation rate for **Mutation_Q** increases to 0.01% rate. *Hypermutation* replaces half of the population with new random individuals every 100 iterations to mitigate stasis.

The use of rational approximations or Padé approximants to compute elementary functions in a numerical coprocessor has been described in [17]. In this work, Padé approximant computations with $x^{1/4}, x^{1/3}, x^{1/2}, x^0, x, x^2, x^3, x^4$ have also been attempted for different functions and are provided as a runtime selection. They have identical chromosome structure with one chromosome each for the numerator and denominator (represented with D in the end) evolved together with the quotient as the result. For instance for sin(x) we have:

$$\begin{aligned} sin(x) &= (\boldsymbol{C0}\cdot x^{(1/4)} + \boldsymbol{C1}\cdot x^{(1/3)} + \boldsymbol{C2}\cdot x^{(1/2)} + \boldsymbol{C3} + \boldsymbol{C4}\cdot x \\ &\quad + \boldsymbol{C5}\cdot x^2 + \boldsymbol{C6}\cdot x^3 + \boldsymbol{C7}\cdot x^4)/(\boldsymbol{C0D}\cdot x^{(1/4)} \\ &\quad + \boldsymbol{C1D}\cdot x^{(1/3)} + \boldsymbol{C2D}\cdot x^{(1/2)} + \boldsymbol{C3D} + \boldsymbol{C4D}\cdot x \\ &\quad + \boldsymbol{C5D}\cdot x^2 + \boldsymbol{C6D}\cdot x^3 + \boldsymbol{C7D}\cdot x^4) \end{aligned}$$

Algorithm 1 Coefficient Prediction (CP) GA

Input: DDC evolved powers of $x\forall\omega|\Omega$ = {set of intrinsic *analog − evolved inputs*}

Output: Coefficients of truncated Puiseux Series for function being approximated

Range Scaling:
$A\_largest$ = largest value in data range
$C\_largest = range(32\_bit\_float)/(A\_largest * 8)$
Initialization:
for $p\epsilon P$ where P denotes population of coefficients **do**
  for $i := 0$ **to** 7 **do**
    Ci = random real number r where 0 < r < C_limit
  end for
end for
while *iterations* < *max_iterations* **do**
  for $p\epsilon P$ **do**
    `Fitness Evaluations`();
    `Compute Stasis Condition`();
    `Elitism`();
  end for
  Dynamic Range Adaptation:
  if *max_fitness* > *accuracy_bound* **then**
    $C\_largest\_new = C\_largest/2$
    Reinitialize selected individuals with {C0, C1, C2, C3, C4, C5, C6, C7} < $C\_largest\_new$
  end if
  for $p\epsilon P$ **do**
    `Crossover`();
    `Assess Stasis`();
    `MutationQ`(Adjust *parameters* based on stasis);
    `Hypermutation`(Switch on/off based on stasis);
    `MutationS`();
  end for
end while
Store winner's chromosome in Results Library

V. EXPERIMENTAL RESULTS AND DISCUSSION

Following analog-processing, DDC was used to focus the effort during evolution on the powers $\{x^{1/4}, x^{1/3}, x^{1/2}, x^0, x, x^2, x^3, x^4\}$ one-by-one in a cascaded fashion to serve as building blocks for the next hierarchy of evolution. The results of their evolution have been indicated in Table 3. The outputs generated exhibit significantly increased accuracy having Total Error well below 10%. Analog evolution produced approximations of each power, which were then refined using DDC as indicated. The Total Error for each case has been indicated with and without use of the hyper-mutation operator. Hyper-mutation helps improve accuracy for larger/integral powers, whereas for fractional powers evolution without hyper-mutation performs better, since only minor refinements with DDC are required for very good results with the evolution of fractional powers as against larger powers where bigger error corrections are needed to achieve desired accuracy.

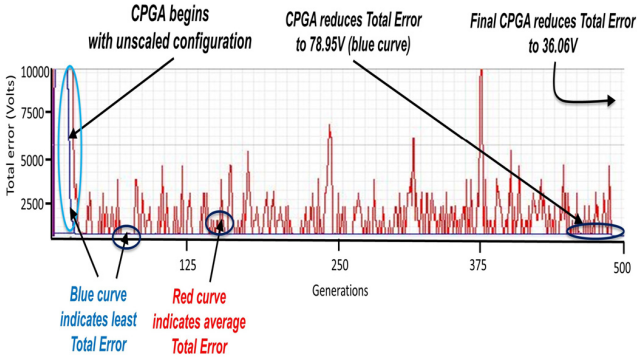

FIGURE 7. Evolution of coefficients approximating *sin(x)* as a Puiseux series in $x^{(1/4)}$, $x^{(1/3)}$, $x^{(1/2)}$, x^0, x, x^2, x^3 and $x^{(1/5)}$.

The average error is reported as the Total Error divided by 256, i.e., the total number of data points. The best case average errors for each power appear at the bottom of Table 2. Apart from these, evolution of $x^{1/5}$ and $x^{1/6}$ were also attempted from a series of ones to serve as input instead of actual analog pre-processed data, to demonstrate and evaluate the ability of DDC to evolve them independent of the analog stage. The results were less favorable than expected, but indicated accuracies well within the 10% limit. This approach proves to be viable for smaller fractional powers since their curves asymptotically approach the $y = 1$ line over the device range. It is also to be noted that as integral powers get larger, their curves approach the $x = 1$ line asymptotically. Since it is not possible to model the behavior of the latter, evolution of larger powers can't be attempted independent of the analog stage. Their evolution with analog stage produced results with much error and hence they were avoided from being used in CPGA.

Powers evolved in cascaded DDC stages were used to approximate transcendental functions as a truncated Puiseux series of powers of x. The result for evolution of *sin(x)* is indicated in Fig. 7. The red line indicates the average fitness and the faint blue line at the very bottom indicates the Total Error for the best fit individual, which is the only circuit retained in each generation and thus maximizes the accuracy of the operational circuit. Initially, the powers used were symmetrical from the fourth root to fourth power and is indicated as the Puiseux 1 Scheme (P1) in Table 2. This resulted in a fitness of 63.232V in total or 247 mV on average for *sin(x)*. The Puiseux 2 Scheme (P2) in Table 2, involved replacing fourth power with fifth root evolved, thereby skewing the symmetry towards concave (fractional) powers and eliminating the power with most error (fourth power) in it. Interestingly, this reduced the Total Error to 36.064V and the average error to 140mV. Thus, starting out from a practically unbounded error, CPGA reduces the average error to about 140 mV in 1,000 iterations.

Several different types of functions were evolved and the corresponding coefficients are indicated in Table 2. Work done in [18] estimates resource and time requirements for computation of elementary functions using a combination of

TABLE 2. Evolution of coefficients.

Circuit Evolved		Coefficients Evolved									Error			Speed up (fold)
		$x^{(1/4)}$	$x^{(1/3)}$	$x^{(1/2)}$	x^0	x	x^2	x^3	x^4	$x^{(1/5)}$	10 % (V)	Total (V)	Avg (mV)	
$e^{(x)}$(Pa)	Num	2.415	0.025	-0.041	0.227	0.150	0.063	0.161	0.152	NA	366.4	195.37	763	0.938
	Den	0	0	0	1	0	0	0	-2.260	NA				
sin(x)(P2)		0.157	0.123	0.189	0.393	0.184	0.073	-0.063	NA	0.068	15.09	36.064	140	1.398
log(x+1)(P2)		0.316	-0.222	0.405	-0.057	0.175	0	0	NA	0.041	26.18	10.485	40	1.670
arctan(x)(P1)		0.183	-0.028	0.613	-0.218	0	0	0	0	NA	25.02	15.616	61	1.692
sinh(x) (Pa)	Num	0.436	0.199	0.561	-0.608	0.197	-0.168	0.128	0.072	180.1	180.1	92.536	361	1.614
	Den	12205.30	20902.78	-9184.51	50845.21	12771.27	9145.81	35283.54	18047.67	NA				
arcsinh(x)(P2)		-2.023	3.120	0	-0.182	0	0	0	NA	-0.087	34.00	28.207	110	2.289
erf(x)(P1)		0.263	1.180	-0.852	0.073	-0.046	0.160	-0.035	0.001	NA	22.02	27.444	107	2.926
sinh(cos(sin(x))) (P2)		0.443	0.187	0	-0.055	0.129	0	0	NA	-0.168	22.74	37.480	146	3.021
sinh(cos(sin(x))) + $sin^4(x)$ (P2)		2.476	0	-1.578	0.992	-0.034	0.183	-0.035	NA	0.081	30.72	40.024	156	6.635
Hyper-mutation rate		300 and slower	300 or faster	200 or slower	NA	200 or faster	200 or faster	150-200	100-150	100	-	-	-	-
Total (V)		4.059	5.247	6.703	NA	18.155	26.57	161.156	1146.01	12.71	-	-	-	-
10 % (V)		29.05	30.63	34.44	NA	52.22	142.32	436.37	1427.11	28.20	-	-	-	-
Average (mV)		15.9	20.4	26.1	NA	70.9	103.8	629.5	4476.60	49.6	-	-	-	-

(P1): Puiseux 1 scheme [$x^{1/4}, x^{1/3}, x^{1/2}, x^0, x, x^2, x^3, x^4$]; (P2):Puiseux 2 scheme [$x^{1/4}, x^{1/3}, x^{1/2}, x^0, x, x^2, x^3, x^{1/5}$]; (Pa): Pade scheme [(P1) *Num* / (P1) *Den*]

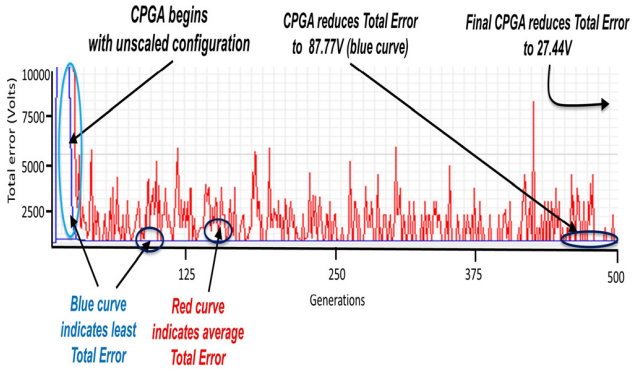

FIGURE 8. Evolution of coefficients approximating *erf(x)* as a Puiseux series in $x^{(1/4)}, x^{(1/3)}, x^{(1/2)}, x^0, x, x^2, x^3$ and x^4.

TABLE 3. Evolution of powers.

Comp. Circuit Evolved (Power of x)	Analog Error (V)	DDC Error		Percent Improvement	
		Without Hyp. (V)	With Hyp. (V)	Without Hyp. (%)	With Hyp. (%)
Fourth Root	6.734	4.649	4.059	31	39
Cube Root	5.984	5.582	5.247	6.7	12.3
Square Root	8.137	6.703	7.475	17.6	8.1
First Power	180.614	18.155	-	-	89.9
Square	35.233	-	26.574	-	24.6
Cube	296.306	196.851	166.953	33.5	43.6
Fourth Power	1696.815	1552.985	1299.830	8.4	23.3

lookup tables and dedicated adder stages for a similar range of x. In contrast, ECHELON completely avoids the need for individual lookup tables for each function and encapsulates the required functionality in a much smaller chromosome in an actual implementation. As can be seen, CPGA leverages errors in the powers to produce functions that have much less Total Error than the constituting powers do. In this context, it is to be noted that the coefficients evolved are applicable to the powers evolved only with their native errors and within the device range for which evolution was performed. On computing the function with the same coefficients while using error free powers, it was observed that the Total Error was larger. Hence ECHELON can adaptively compose building blocks using a set of hardware resources to compute corresponding coefficients which best optimize the output.

Fig. 8 indicates the coefficients evolved for the sigmoid error function or *erf(x)* which has an ARM core execution time of 124.835ms, while ECHELON achieves the same computation in 42.661ms. The execution time here refers to the time taken by the function to obtain and the return the result, while evolution time refers to the time spent in evolving the circuit that evaluates the function being called. The execution time data for different functions evolved have been tabulated in Table 4. The improvement in execution time due to ECHELON over those for standard library based implementations ranges from speedups of 1.4-fold to 6.6-fold and on average, is of the order of 2.72-fold for the functions evolved. Also it can be observed that the computation times for functions over all ranges of complexity are essentially the same in ECHELON as the only instructions involved are simple multiplication, addition and subtraction operations. The only differentiating feature required to accommodate higher complexity functions is the utilization of adequate chromosomes to denote the coefficients, as a viable alternative to lookup tables.

The main advantage of utilizing ECHELON is seen during computation of more complex composite functions as greater

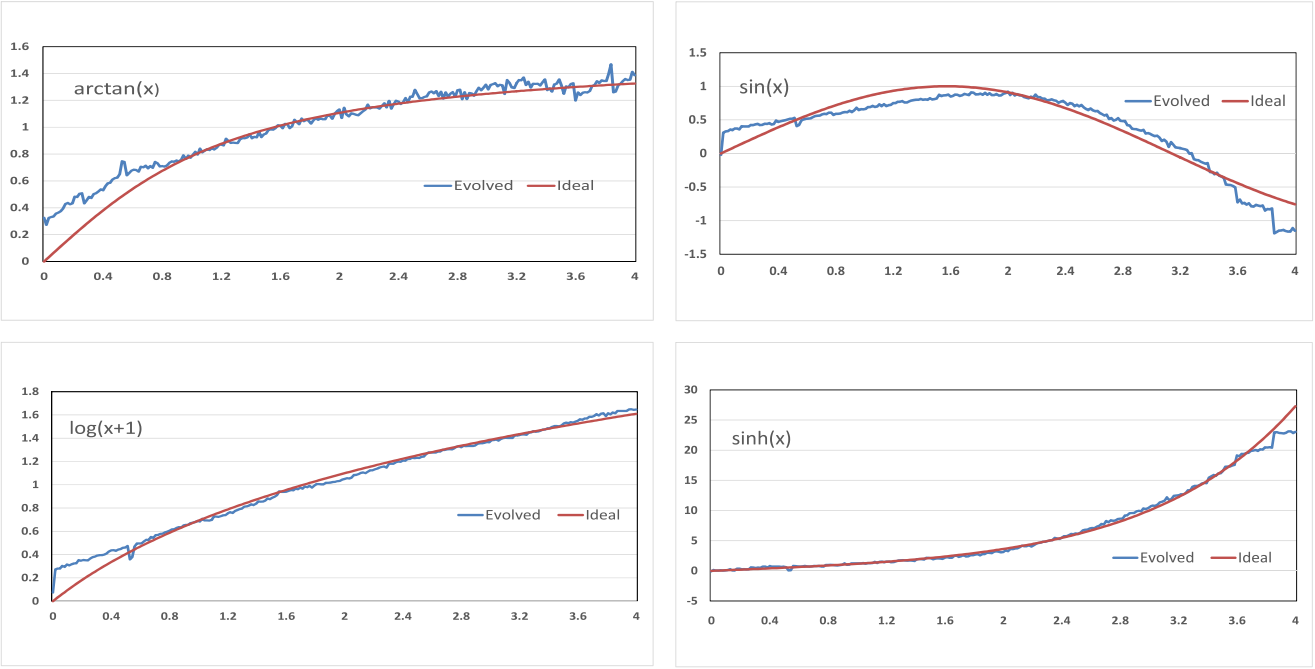

FIGURE 9. ECHELON evolved circuits (blue) approximating *arctan(x), sin(x), log(1 + x) and sinh(x)* as a Puiseux series in $x^{(1/4)}$, $x^{(1/3)}$, $x^{(1/2)}$, x^0, x, x^2, x^3 $x^4/x^{(1/5)}$ or the Pade approximant with the corresponding ideal curves (red) indicating relative performance of ECHELON.

TABLE 4. Execution time data.

Times / Circuit Evolved	*Total Core time (ms)*	*Core time per point(μs)*	*Total ECHELON time (ms)*	*ECHELON time per point (μs)*	*Speedup (fold)*
sin(x)	59.327	231.74	42.409	165.66	1.398
log(x+1)	70.040	273.59	41.916	163.73	1.670
arctan(x)	70.106	273.85	42.452	165.83	1.692
sinh(x)	87.022	339.92	40.873	159.66	2.129
arcsinh(x)	93.583	365.55	40.869	159.64	2.289
erf(x)	124.835	487.63	42.661	166.64	2.926
sinh(cos(sin (x)))	138.314	540.28	41.318	161.40	3.021
sinh(sin(cos (x))) + sin^4(x)	278.830	1089.17	42.023	164.15	6.635
Average	115.25	450.21	41.815	163.33	2.72

speedups are achieved. The ARM core is expected to be operating at the bus clock frequency of 24 MHz. However, it is to be noted that the large computation times reported here are owing to several factors, including the threads active in it while executing the code, memory access times for stored results and peripheral data transfer speed, which determines the actual number of clock cycles needed. The computation times were measured using in-built PSoC timers with 24-bit resolution counting down, set to count to a maximum range for measurement of up to one second. Execution times are a relative measure of performance of the two approaches to perform the same computation.

Figure 9 indicates some circuits evolved using ECHELON within each category of transcendental functions targeted. For *arctan(x)* and *log(x + 1)* functions, CPGA performed much better than others, reducing average error to 61mV and 40mV respectively over the device range. Work done in [19] compares prevailing formulae and implementations to compute the *arctan(x)* function with a lookup table based approach and argues in favor of the latter for an ARM based platform. Comparison of ECHELON with the same indicates that ECHELON achieves good results for a larger number of points at a much slower frequency and achieves greater speedups against the reference platform without the need for a lookup table. Initially, all functions were computed with the (P1) scheme of constituting powers and it was noted that the Total Errors in final circuits were larger than the 15% error limit for most functions. With evolution of the fifth root, the same was used to replace fourth power and much better performance was observed. Most of the functions attempted stayed within the 15% error limit and quite a few stayed within the 10% limit targeted, three of which are indicated in Fig. 9. Unfortunately, *sin(x)* exceeded the 15% limit for itself, but this is because the output range of *sin(x)* is relatively small and hence is more challenging to evolve to a high degree of accuracy.

For rapidly increasing functions such as *sinh(x)* and $e^{(x)}$, (P1) and (P2) schemes failed to produce results within 10% error, but still managed to stay close to or within 15%. However, Padé approximant scheme (Pa) produced a much better approximation, at the cost of increased computation time owing to division operations and computation with a greater number of coefficients. Execution time data for *sinh(x)* indicates performance for approximation calculated with (P2) scheme that produced a Total Error of 101V versus 92V as indicated in Table 2. The computation time with (Pa)

scheme was found to be 20-25% slower than with (P2) or (P1) schemes in general.

Functions evolved spanned three categories ranging from concave (slowly increasing), convex (rapidly increasing) and oscillatory (increasing and decreasing). Deeply concave functions benefit more with (P2) scheme, meanwhile shallow concave functions perform well using the (P1) scheme and convex functions may perform best from (Pa) scheme. Oscillatory functions were hardest to evolve to within 10% error and they can benefit from balanced schemes with very small coefficients. Illustratively, moderately concave *arctan(x)* function performed best with (P1) scheme to attain a Total Error of 15.616V, as against (P2) and (Pa) schemes resulting in Total Errors of 21.507V and 58.378V respectively. Excessive use of very small (concave) powers of x doesn't necessarily improve the accuracy of concave functions. When evolution for *arctan(x)* was attempted with $\{x^{1/4}, x^{1/3}, x^{1/2}, x^0, x, x^2, x^{1/5}, x^{1/6}\}$ the Total Error was 126V approximately. The zeroth power term proved to be of significance in evolution, adding or subtracting a required bias adjustment.

VI. CONCLUSION AND FUTURE WORK

ECHELON is developed as a multi-level embedded strategy to evolve transcendental functions as a truncated fractional power series of the independent variable. Development of this approach on a highly-constrained reconfigurable fabric recognizes the need to achieve powerful computational capabilities while constraining storage requirements compared to a look-up-table approach to correlate (input, output) pairs.

ECHELON was instrumental in approximating functions to the targeted 10% error limit for most of the functions evolved. Further improvements in accuracy for existing schemes and powers may be achieved using evolution with a larger number of generations, bigger populations, and improved mutation schemes. It is also expected that with a larger chromosome for CPGA and hence more power terms in the calculation, greater accuracies may be realizable albeit at the cost of some computation time while still attaining speedup for complex functions over other implementations.

The results library by ECHELON may be used as a configuration store with minimal memory overhead for chromosomes of different computational circuits evolved to perform rapid computations of functions with increasing levels of complexity. Significant speedup in performance is achieved using ECHELON in comparison with standard approaches to perform similar computations, as shown in Table 4, and a directly proportional relationship between speedup and complexity makes this approach more promising for computation and analysis of complex datasets. Memoization techniques similar to [20] may be used for specific data ranges which require further speedups. Within the device limits, ECHELON realizes a promising mixed-signal approach towards decoupling mathematical complexity from computation running time.

ACKNOWLEDGEMENTS

The authors wish to thank the reviewers for constructive suggestions and Sindhu Muttineni for editorial assistance.

REFERENCES

[1] P. Hasler and D. V. Anderson, "Cooperative analog-digital signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 4. May 2002, pp. IV-3972–IV-3975.

[2] H. Tabkhi, R. Bushey, and G. Schirner, "Function-level processor (FLP): A high performance, minimal bandwidth, low power architecture for market-oriented MPSoCs," *IEEE Embedded Syst. Lett.*, vol. 6, no. 4, pp. 65–68, Dec. 2014.

[3] S. D. Pyle, V. Thangavel, S. M. Williams, and R. F. DeMara, "Self-scaling evolution of analog computation circuits with digital accuracy refinement," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Jun. 2015, pp. 1–8.

[4] V. Thangavel, "Cascaded digital refinement for intrinsic evolvable hardware," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, USA, 2015.

[5] N. Imran, R. F. DeMara, J. Lee, and J. Huang, "Self-adapting resource escalation for resilient signal processing architectures," *J. Signal Process. Syst.*, vol. 77, no. 3, pp. 257–280, Jul. 2013.

[6] R. S. Oreifej, R. N. Al-Haddad, H. Tan, and R. F. DeMara, "Layered approach to intrinsic evolvable hardware using direct bitstream manipulation of Virtex II pro devices," in *Proc. 17th Int. Conf. Field Program. Logic Appl. (FPL)*, Amsterdam, The Netherlands, Aug. 2007, pp. 299–304.

[7] R. Kirner, M. Grössing, and P. Puschner, "Comparing WCET and resource demands of trigonometric functions implemented as iterative calculations vs. table-lookup," *OASIcs-OpenAccess Series in Informatics*, vol. 4. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.

[8] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.

[9] S. Kazadi *et al.*, "Insufficiency of piecewise evolution," in *Proc. 3rd NASA/DoD Workshop Evol. Hardw.*, Los Alamitos, CA, USA, Jul. 2001, pp. 223–231.

[10] S. Ando, M. Ishizuka, and H. Iba, "Evolving analog circuits by variable length chromosomes," in *Advances in Evolutionary Computing*. Berlin, Germany: Springer, 2003, pp. 643–662.

[11] P. C. Haddow and A. M. Tyrrell, "Challenges of evolvable hardware: Past, present and the path to a promising future," *Genet. Program. Evol. Mach.*, vol. 12, no. 3, pp. 183–215, 2011.

[12] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," in *Proc. 1st Eur. Conf. Artif. Life*, Paris, France, Dec. 1991, pp. 245–254.

[13] A. Poteaux and M. Rybowicz, "Improving complexity bounds for the computation of Puiseux series over finite fields," in *Proc. ACM Int. Symp. Symbolic Algebraic Comput. (ISSAC)*, Bath, U.K., Jul. 2015, pp. 299–306.

[14] R. J. Walker, *Algebraic Curves*. Princeton, NJ, USA: Princeton Univ. Press, 1950.

[15] *PSoC-5LP Architecture TRM (Technical Reference Manual) Version D*, document 001-78426, Cypress Semiconductor, Oct. 2015.

[16] *PSoC-5LP Registers TRM (Technical Reference Manual) Rev. D*, document 001-82120, Cypress Semiconductor, Jun. 2015.

[17] I. Koren and O. Zinaty, "Evaluating elementary functions in a numerical coprocessor based on rational approximations," *IEEE Trans. Comput.*, vol. 39, no. 8, pp. 1030–1037, Aug. 1990.

[18] W. F. Wong and E. Goto, "Fast evaluation of the elementary functions in single precision," *IEEE Trans. Comput.*, vol. 44, no. 3, pp. 453–457, Mar. 1995.

[19] A. Ukil, V. H. Shah, and B. Deck, "Fast computation of arctangent functions for embedded applications: A comparative analysis," in *Proc. IEEE Int. Symp. Ind. Electron. (ISIE)*, Jun. 2011, pp. 1206–1211.

[20] A. Suresh, B. N. Swamy, E. Rohou, and A. Seznec, "Intercepting functions for memoization: A case study using transcendental functions," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 2, Jun. 2015, Art. no. 18.

VIGNESH THANGAVEL received the B.E. (Hons.) degree in electrical and electronics engineering from the Birla Institute of Technology and Sciences, Pilani, India, in 2013, and the M.S. degree in electrical engineering from the University of Central Florida, Orlando, FL, USA, in 2015. His research interests include adaptive computer architectures and evolvable hardware, embedded systems, and FPGA design. His co-authored research work on hybrid analog and digital adaptive design received the Best Design Paper Award at the 2015 NASA/ESA Conference on Adaptive Hardware and Systems.

ZI-XIA SONG received the Ph.D. degree from the Georgia Institute of Technology in Algorithms, Combinatorics and Optimization in 2004, under the supervision of R. Thomas. She then spent one year with The Ohio State University as a Zassenhaus Assistant Professor. Since 2005, she has been a full-time Faculty Member with the University of Central Florida. Her research interests are in graph theory, combinatorics, optimization, and algorithms. Her current interest is mainly in structural graph theory, well-quasiordering, graph colorings, and degree sequences of graphs.

RONALD F. DEMARA (S'87–M'93–SM'04) received the Ph.D. degree in computer engineering from the University of Southern California, in 1992. Since 1993, he has been a full-time Faculty Member with the University of Central Florida, where he is currently a Professor and Computer Engineering Program Coordinator. His research interests are in computer architecture with emphasis on reconfigurable logic, evolvable hardware, and emerging computing devices, in which he has published 175 articles. He received the Joseph M. Bidenbach Outstanding Engineering Educator Award from the IEEE in 2008. He has served on the Editorial Boards of the IEEE TRANSACTIONS ON VLSI SYSTEMS, the *Journal of Circuits, Systems, and Computers*, the *Journal Microprocessors*, and *Microsystems*, and an Associate Guest Editor of the *ACM Transactions on Embedded Systems*, on various conference program committees. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS.