Designing Inexact Systems Efficiently using Elimination Heuristics

Shyamsundar Venkataraman, Akash Kumar
Department of Electrical and
Computer Engineering
National University of Singapore, Singapore
Email: {akash,shyam}@nus.edu.sg

Jeremy Schlachter, Christian Enz
Integrated Circuits Laboratory
EPFL, Neuchâtel 2
Switzerland
Email: {jeremy.schlachter, christian.enz}@epfl.ch

***Abstract*—There are a wide variety of applications that are able to tolerate small errors in the values of the outputs, provided they are within the application-specific thresholds. For such applications, there have been many efforts to study the trade-off involved in the accuracy of the output and the energy/area requirement. However, most of the efforts have been at the level of individual components. In this article, we present a design flow to study the inexactness at the level of system and provide heuristics to quickly explore the design-space under given inexactness and area/energy constraints. The approach is applied to various digital signal processing filters and an ECG application of QRS detection. In both cases, orders of magnitude speed-ups are obtained in the design-flow process. Area savings of 21.61% and power savings of 22.79% were observed for a low-pass filter having a relative error of just 8E-5%.**

I. Introduction

Reliable operation of a system has been a major concern with decreasing transistor feature sizes. Many research works are therefore focused on how reliable results can be achieved from a system built on unreliable components [1]–[5]. However, the need for reliable operations at any cost is being scrutinized, especially considering that a slightly incorrect operation at the hardware level may consume 1-2 orders of magnitude less power and energy than its error-prone counterpart [6], [7]. This also helps in minimizing the power and energy hurdle that is emerging limiting further shrinking of transistors and the number of components that can be simultaneously used (also known as dark-silicon era). It is also interesting to note that there has been an increase in the number of applications that make sense of real-world data that is inherently noisy and therefore the output can never be perfect. Big-data analytics, machine learning, speech recognition are examples of such applications [8], [9].

Many research directions have spawned to exploit such application and architecture characteristics. Some works have been done on probabilistic computing where the general idea is to decrease the voltage of operation significantly to reduce the power consumption albeit at the cost of reliable circuit operation [2], [10]. Trade-off exists between the amount of energy the designers wants to save and the probability of the correct operation. There is another branch of research which focuses on approximate or inexact logic circuits, where the number of transistors is reduced in order to save energy [6], [7]. One main difference in this approach is that the result, although inaccurate, is still deterministic unlike the probabilistic computing approaches. Further, with such techniques silicon area is lower since fewer transistors are used than their exact counterparts. Such works also allow a trade-off between the accuracy of the result and energy/area savings that can be achieved.

Unfortunately, designing such components is largely a manual affair and there are hardly any tools to assist architects in designing large systems. Tools such as [11], [12] provide a framework to design inexact circuits by simplifying the exact designs. However, these tools do not focus on building bigger designs using smaller inexact components. Another problem that limits the size of the components to a simple adder or a multiplier is the long simulation time to compute the inexactness of a module. Simulating a 32-bit high-frequency multiplier at the gate-level takes about 8 minutes on a quad-core Intel i7 processor with 16 GB of memory. Needless to say that as the system becomes larger, the simulation time increases significantly. Further, when each module in the system has multiple options of inexactness to choose from, exploring all design points to identify the ideal combination is simply infeasible through simulation.

In this paper, an approach is presented to estimate the properties of the combined system that is composed of individual components, given their area, power, energy and inexactness. This estimation takes in the order of microseconds depending on the number of composing components. Nevertheless, when each component has multiple options to choose from depending on the inexactness property, for large systems evaluating all options exhaustively soon becomes intractable, as shown in our experiments. In order to solve this problem, a heuristic is presented that reduces this execution time by discarding the combinations that are guaranteed to not give any new Pareto-optimal solutions.

Contributions: Following are the key contributions:

- An algorithm for quickly estimating the inexactness of the entire system.
- A design-flow that uses the above algorithm to design the entire system under the area and power constraints, given the inexact components and their properties.
- A heuristic to reduce the design-space exploration time by eliminating only the non-distinct points. This ensures that no Pareto-optimal points are eliminated.
- Results of the design-flow applied to an ECG application of QRS detection.

978-3-9815370-4-8/DATE15/©2015 EDAA

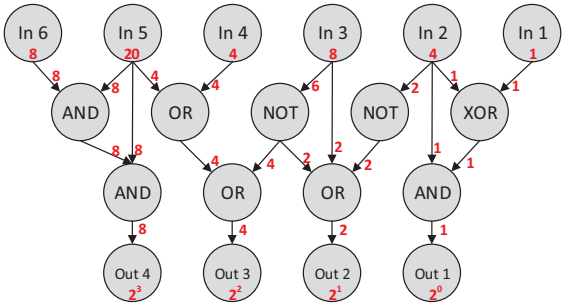

Fig. 1: Directed acyclic graph representation of a gate level netlist

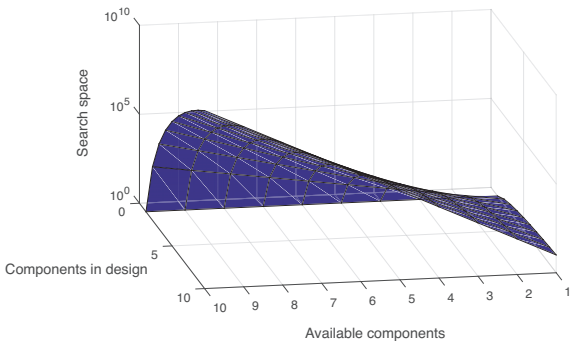

Fig. 2: Size of search space

The approach has been applied for digital signal processing filters of various sizes ranging from 1 to 30 taps. We observe that while the actual value of the estimates often differ from the simulated result, the estimated values follow the same trend as the simulated results for most designs. Therefore, the Pareto front computed using the estimates is almost identical to the one generated using simulation, albeit with a speed-up of several orders of magnitude. The heuristic to prune away duplicate Pareto points before exploration further reduces the exploration time by up to 4 orders of magnitude. The hardware circuit for an ECG algorithm, namely QRS detection, has also been designed using our approach.

The remainder of the paper is organized as follows. Section II introduces a brief background of the concepts used and the motivation behind the work. Section III presents the proposed work and the overall design flow. Section IV presents the results and then studies a specific case study in detail. Finally, Section V concludes the paper with future directions.

II. Background and Motivation

Inexact or approximate adders and multipliers presented in this paper are generated using the *probabilistic pruning* technique [6]. In this approach, some circuit parts or elements are literally removed in exchange for significant reduction of silicon area, power consumption and critical path delay, at the cost of some occasional errors. A circuit can be represented as a directed acyclic graph shown in Figure 1, where the edges are wires, and the nodes are elements such as gates, or at coarser granularity, full adder blocks. The decision of pruning a node is guided by two parameters: the switching activity, which is extracted from hardware simulations, and the significance, which is a structural parameter based on the weight of each primary output. Nodes with the lowest significance-activity product are pruned first. The resulting amount of error is proportional to the number of pruned nodes.

Figure 2 shows the increase in the search space with the increase in the design size and the number of components available. The X-axis represents the number of available components while the Y-axis represents the number of components in the design. As explained in Section III-B, the search space increases exponentially when either the number of components in the design or the number of available components is increased. This trend can also be seen in the surface plot shown in the figure. It is to be noted that the Z-axis in the figure (search space size) is a log plot and hence the search space increases to over a billion possibilities for a very small design with 10 components. This shows the need for the heuristic search proposed in Section III-C.

III. Design Flow

A. Estimating Inexact Components

In this paper, 2 inexact components (adders and multipliers) have been considered. As explained in Section II, the inexact components are built by pruning away those parts of the design that are not used frequently, thereby leading to savings in both area and power. Two configuration types — series and parallel — the two fundamental digital signal processing architectures, were considered for the components as shown in Figure 3. Each node corresponds to a single inexact component with its different parameters (as enumerated in Tables II and III). Any complex system built with these smaller components can be represented as a graph $G = (V, E)$, where V is the set of vertices corresponding to the inexact component and E is the set of edges connecting the vertices. A path P in the graph G represents a set of vertices starting from the input vertices and ending at the root vertex. It is represented by $P_i = \langle v_1, ..., v_n \rangle$ where i is the i^{th} path of the graph, v_1 is the input vertex and v_n is the root vertex. It is to be noted here that a path P_i can only contain components in a serial fashion. Relative error of a path P_i is defined as

$$R_{P_i} = \prod_{k=1}^{n} 1 - R_{v_k} \tag{1}$$

where R_{v_k} is the relative error of vertex k. The overall relative error of the design is computed as

$$R_G = \max(R_{P_i}) \forall P_i \in G \tag{2}$$

Similarly, the delay of the entire design is computed as the maximum delay of any of the paths in the design. It can be represented as

$$D_G = \max(D_{P_i}) \forall P_i \in G \tag{3}$$

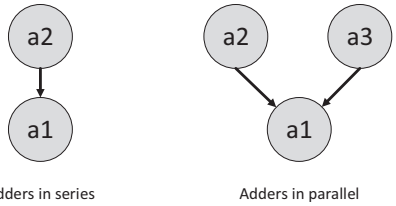

Fig. 3: Possible configurations of components

where D_{P_i} is the delay of path i. The total area and power required by the design is estimated as the summation of each of the individual components' area and power i.e. $A_G = \sum_{k=1}^{n} A(v_k)$ and $P_G = \sum_{k=1}^{n} P(v_k)$, where n is the total number of vertices in G.

B. Exhaustive Search

Given a list of inexact components to choose from, as shown in Tables II and III, an exhaustive search is performed to consider all designs that are possible. For each design, relative error, power and area are estimated. Once all the possible designs for the given inexact components are explored, the designer can then choose a specific design based on a specific parameter constraint.

Given n_a and n_m different possibilities of inexact adders and multipliers to choose from respectively, the size of the search space for exhaustive search can be given by $n_a^{|V_a|} \times n_m^{|V_m|}$, where V_a and V_m denote the set of inexact adders and multipliers required in the design respectively. This shows that the exhaustive search has an exponential complexity as both the number of vertices and the possibilities of available components increase. For example, with a total of 4 vertices (2 adders and 2 multipliers) and 5 available components each for adders and multipliers, the total number of possible designs in the search space would be 625. However, if the number of vertices in the design increase to 10 (5 adders and 5 multipliers), the total number of possible designs increases to about 9.7 million possibilities. This shows that the time taken to estimate parameters for every possible design point becomes intractable very fast. Hence, a heuristic search is proposed in the following section to reduce the search space.

C. Heuristic Search

As can be seen from Equation 2, the different arrangements of the same components in a serial fashion does not change the overall relative error of the design. This means that the overall search space can be minimized by eliminating those designs that result in the same estimates. The heuristic search only considers those points that would result in distinct points on the Pareto plot. Designs that result in the same points are not taken into consideration and hence the time taken to estimate the overall parameters reduces. Mathematically, the number of points for a given design using the heuristic search is given by the recursive function $f(|V_a|, n_a) \times f(|V_m|, n_m)$ where $f(x, y) = f(x-1, y) + f(x, y-1)$ and the base cases are $f(x, 1) = 1$ and $f(1, y) = y$ with x being the number of components in the design and y being the number of inexact components available. It can be seen that this equation is of polynomial time while the exhaustive search is of exponential time. For a design with 14 vertices and 2 available components, the number of possible designs in the heuristic search space is only 64, whereas for exhaustive search it would be $16,384$.

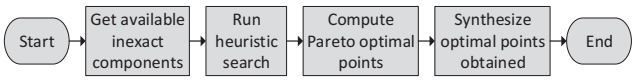

Fig. 4: Overall design flow

D. Synthesizing Final Design

After the heuristic search is completed, the set of Pareto-optimal points is obtained. This set of points is then synthesized to obtain the accurate values for every parameter considered. Since only a few points of the entire search space are synthesized, this leads to a much faster design flow as compared to synthesizing every possible design.

An overall flow of the design is shown in Figure 4. The first step is to retrieve the available components to be used for the design. The design to be built is then passed along with the available components to the tool, which runs the heuristic search determining the distinct points in the search space. In the next step, the Pareto-optimal points are then obtained from these distinct points. Finally, the Pareto-optimal points are synthesized to obtain the actual values of the parameters.

IV. Results and Case Study

A. Accuracy of Estimation

Figure 5 shows the accuracy of the actual synthesized (simulated) design and the estimated design. Designs $1-10$ contain a 2-component series adder, $11-20$ contain a 3-series adder and $21-30$ contain a 3-component parallel adder. For each set of designs, one of the components were varied while keeping the others constant. As can be seen from the result, the estimated result is always of the same order as the simulated one. Moreover, the estimation generally follows the same trend as the simulated design. These results show that the estimation technique can give a quick estimate of the overall design parameters without having to synthesize the designs.

B. Case Study Background

The electrocardiogram (ECG) is a recording of the electrical activity of the heart, typically used for the diagnosis of heart abnormalities. The detection of the electrical signals produced by the heart requires various hardware consisting of multiple Digital Signal Processing (DSP) components/filters. Hence, such an application has been chosen as a case study for this paper, illustrating how the proposed approximate solution can show savings in both area and power aspects.

For this case study, we consider the detection of QRS signals, one of the most important features of the ECG signal. *QRS complex* (see Figure 6) reflects the rapid depolarization of the right and left ventricles of the heart. It generally lasts $0.06-0.10$ seconds and abnormalities in it are used

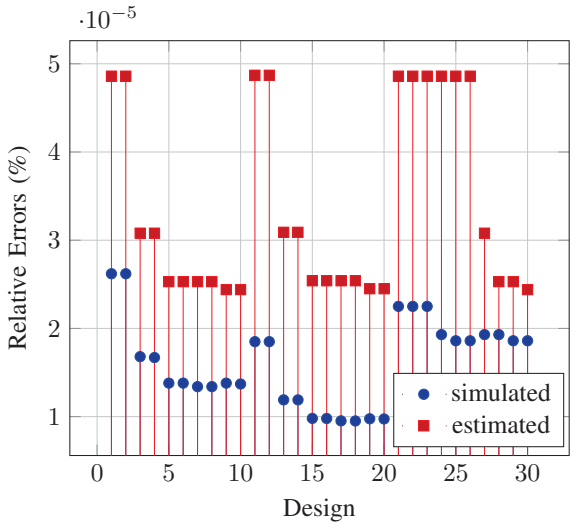

Fig. 5: Simulated versus estimated relative error

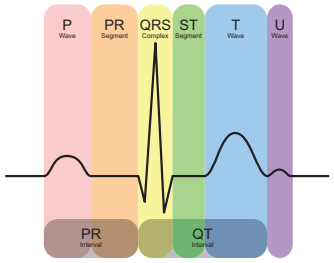

Fig. 6: Various features of the ECG signal

as indications for *hyperkalemia*, a condition in which the concentration of Potassium in the blood is elevated.

Pan and Tompkins proposed an algorithm for the detection of QRS signals in their seminal paper in 1985 [13]. Figure 7 summarizes the various steps of the algorithm. In order to attenuate the noise, the signal is first passed through a band-pass filter consisting of a cascaded low-pass and high-pass filter. Subsequently, the signal is then differentiated, squared and time averaged. For the purpose of the case study, the parameters for the filters and components were chosen as given by Pan and Tompkins [13]. Table I outlines the different parameters used for the filters.

TABLE I: Parameters used for QRS algorithm components

Component	Cutoff f	Gain	Delay (samples)	Transfer function
Low-pass filter	11 Hz	36	6	$H(z) = \frac{(1-z^{-6})^2}{(1-z^{-1})^2}$
High-pass filter	5 Hz	32	16	$H(z) = \frac{(-1+32z^{-16}+z^{-32})}{(1+z^{-1})}$
Derivative	-	-	2	-
Squaring function	-	-	1	-
Integration	-	-	30	-

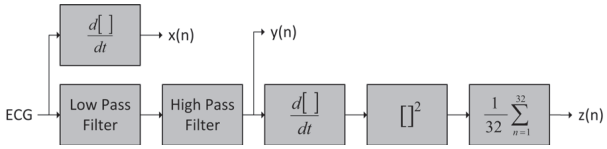

Fig. 7: QRS flow

TABLE II: Available inexact adders

Pruned Nodes	Rel. err. (%)	Delay (ns)	Power (mW)	Area (μm^2)	EDAP
0/417	0	0.29	0.41	2416	291
10/417	8.85E-08	0.3	0.31	1823	171
20/417	5.64E-07	0.31	0.26	1501	122
30/417	1.03E-06	0.3	0.35	2045	218
40/417	4.73E-06	0.27	0.42	2428	279
50/417	2.43E-05	0.28	0.35	2067	207

C. Inexact QRS Design

In order to build an inexact QRS design, inexact components as listed in Table I were first built. Since using components of different inexactness results in different relative errors, powers, and area usage, an optimal set of Pareto points were plotted for the different set of available adders and multipliers. The list of available adders and multipliers is shown in Tables II and III respectively. The first column denotes the number of nodes that have been pruned from the adder/multiplier. The other columns denote the value of each different parameter of the inexact component with the final column showing the Energy-Delay-Area Product (EDAP). The general trend of the design is that the area, power and energy decreases exponentially with a higher number of nodes pruned. As expected, the relative errors increase as the number of nodes pruned increases.

D. Pareto-Optimal Points

Choosing the most optimal components for a given area, energy or power requires physical synthesis and simulation of each and every possible inexact component. However, as shown in the previous section III-B, synthesizing every possible solution for the entire search space takes a very long time (in the order of over 10^9 hours). Through our proposed solution, we reduce the time taken to less than 2 hours (for typical designs considered). The sections below compare the solutions obtained for different components using a brute force approach and the proposed heuristic approach.

1) Low Pass Filter: Figure 8 shows the different points obtained for the low-pass filter using an exhaustive search and

TABLE III: Available inexact multipliers

Pruned Nodes	Rel. err. (%)	Delay (ns)	Power (mW)	Area (μm^2)	EDAP
0/1543	0	0.83	2.45	6196	12578
10/1543	7.87E-05	0.87	1.92	5055	8442
20/1543	4.90E-04	0.84	2.44	6033	12385
30/1543	6.87E-03	0.84	2.20	5767	10657
40/1543	9.68E-02	0.84	2.25	5869	11092
50/1543	1.92E-01	0.82	2.14	5820	10200

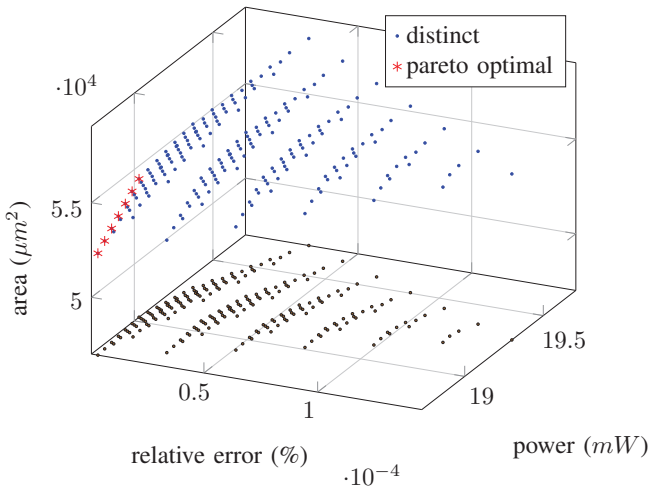

Fig. 8: Low-pass filter heuristic search

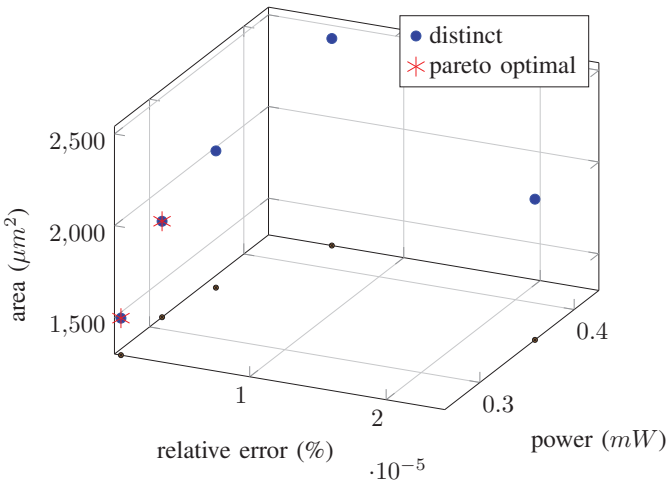

Fig. 9: Integration heuristic search

a heuristic search. For the exhaustive approach, all possible combinations of the available inexact components were estimated, hence leading to a huge search space. Overall, there were over 6.1 billion combinations for the low pass filter containing just 13 components. The total time taken to run the exhaustive search was 2.93×10^{12} ns. Clearly, such an approach is not scalable. The red stars in the figure correspond to the Pareto optimal points from the initial estimation. The same figure plots the points through the proposed heuristic. As can be seen from the graph, the points plotted for the exhaustive and the heuristic search are exactly the same, though the heuristic search uses over 4 orders fewer points and is also 3.2×10^4 times faster than the former. Moreover, the red stars plotted (Pareto-optimal points) are the same as obtained in the exhaustive approach.

2) Integration: Figure 9 shows the points plotted for the integration component of the QRS algorithm. Only 5 distinct points can be obtained from over 4.65×10^{21} possible combinations since the design of the integration unit contains a chain of 29 cascaded adders. Though the search space is huge, it can be noted from Equation 2 that the order of the inexact components does not matter to the overall relative error obtained. Hence, multiple combinations of the inexact modules only give a few distinct points on the graph. .

3) Other modules: The plots for the high-pass filter show similar results as the low-pass filter and have hence been omitted in view of the space limitations. Similarly, the differentiation and squaring modules are quite trivial to plot as they have very few individual components. Table IV shows the number of options explored for exhaustive and the heuristic solutions. It is to be noted that since the high-pass filter and integration module had a large number of vertices, only 2 approximate components were considered for them since exhaustive search for 5 approximate components was intractable.

E. Space and Time Complexity

Figure 10 shows the number of points explored by both exhaustive and heuristic search for a varying number of inexact components. For the inexact components, it was assumed that an equal number of adders and multipliers were available. It can be observed that the time taken as well as the number of points explored for heuristic search is on average 2 orders of magnitude smaller than the exhaustive search as can be seen from Figure 11.

TABLE IV: Pareto-space exploration

QRS Module	Total vertices	Time taken (ns)		Points explored	
		Exhaustive	Distinct	Exhaustive	Distinct
Low-pass filter	13	1.1×10^{11}	2.7×10^7	6.1×10^9	1.0×10^5
High-pass filter	33	1.48×10^{12}	9.4×10^5	1.7×10^{10}	3.2×10^2
Integration	30	7.1×10^{18}	9.2×10^8	2.1×10^9	5.0×10^2
Squaring	1	3.6×10^4	2.5×10^4	5	5

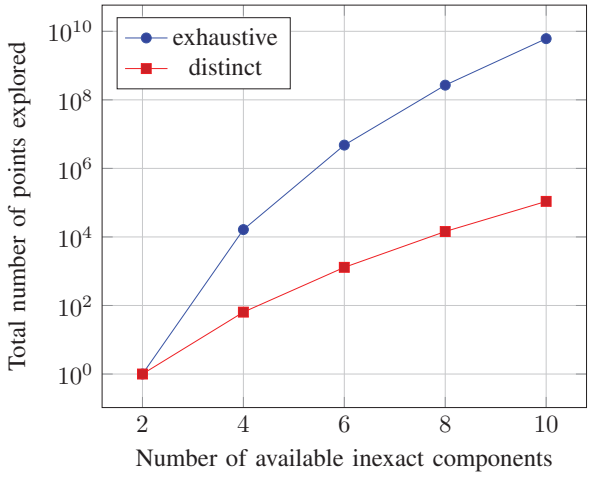

Fig. 10: Number of points plotted for exhaustive and heuristic search

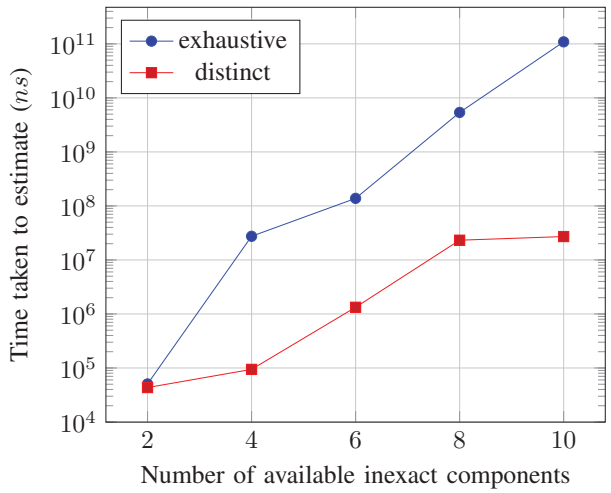

Fig. 11: Time taken for exhaustive and heuristic search

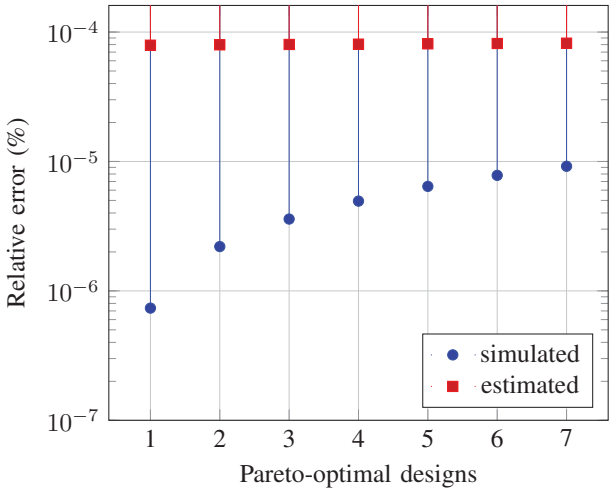

Fig. 12: Estimation versus synthesis of the LPF

F. Estimation versus Synthesis

Figure 12 plots the relative errors of the Pareto-optimal points obtained in Section IV-D and the relative errors of the designs corresponding to the Pareto-optimal points. Note that the Y-axis is in log scale. It can be observed that though the estimated values are not accurate, they are always conservative — they are always more inexact than the synthesized designs — and follow the same trend as the synthesized designs. This result shows that the proposed technique is indeed effective in estimating the optimal inexact designs. The average relative error of these optimal designs was 8.07×10^{-5} with average power savings of 22.79% and area savings of 21.62%.

V. Conclusion

In this paper, an overall design flow is presented to study the inexactness at a system level. An algorithm is proposed to quickly estimate the inexactness of a group of inexact components and also that of the entire system. A heuristic is also proposed to reduce the design-space exploration by pruning away the non-distinct points. The overall tool helps quickly estimate the Pareto optimal points which can then be physically synthesized to realize the actual inexact circuit. Experimental results show that the proposed tool achieves a speed-up of up to 4 orders of magnitude faster than the exhaustive approach. These results and design flow have then been applied to a real-life ECG application of QRS detection illustrating its use.

Acknowledgments

We would like to thank Prof. Krishna Palem from Rice University for stimulating this research, bringing the collaborators together and providing regular guidance on the matter.

References

[1] K. V. Palem, "Energy Aware Algorithm Design via Probabilistic Computing: From Algorithms and Models to Moore's Law and Novel (Semiconductor) Devices," in *Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2003. [Online]. Available: http://doi.acm.org/10.1145/951710.951712

[2] J. Kim and S. Tiwari, "Inexact Computing Using Probabilistic Circuits: Ultra Low-power Digital Processing," *J. Emerg. Technol. Comput. Syst.*, 2014. [Online]. Available: http://doi.acm.org/10.1145/2564925

[3] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2013.

[4] S. Cheemalavagu, P. Korkmaz, and K. V. Palem, "Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship," in *Proc. of The 2004 International Conference on Solid State Devices and Materials*, 2004.

[5] P. Korkmaz, B. E. Akgul, K. V. Palem, and L. N. Chakrapani, "Advocating Noise as an Agent for Ultra-Low Energy Computing: Probabilistic Complementary Metal–Oxide–Semiconductor Devices and Their Characteristics," *Japanese journal of applied physics*, 2006.

[6] A. Lingamneni, C. Enz, J. L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2011.

[7] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, "Synthesizing Parsimonious Inexact Circuits Through Probabilistic Design Techniques," *ACM Trans. Embed. Comput. Syst.*, 2013. [Online]. Available: http://doi.acm.org/10.1145/2465787.2465795

[8] A. Acero, L. Deng, T. T. Kristjansson, and J. Zhang, "HMM adaptation using vector taylor series for noisy speech recognition." in *INTERSPEECH*, 2000.

[9] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000.

[10] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem, "Probabilistic Arithmetic and Energy Efficient Embedded Signal Processing," in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2006. [Online]. Available: http://doi.acm.org/10.1145/1176760.1176781

[11] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic Logic Synthesis of Approximate Circuits," in *Proceedings of the 49th Annual Design Automation Conference*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2228360.2228504

[12] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: A Unified Design Paradigm for Approximate and Quality Configurable Circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2485288.2485615

[13] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *Biomedical Engineering, IEEE Transactions on*, 1985.