An Efficient Methodology for Hierarchical Synthesis of Mixed-Signal Systems with Fully Integrated Building Block Topology Selection

Tom Eeckelaert, Raf Schoofs, Georges Gielen, Michiel Steyaert, Willy Sansen
Katholieke Universiteit Leuven,
Department of Electrical Engineering, ESAT-MICAS
Kasteelpark Arenberg 10, B-3001 Leuven
Tom.Eeckelaert@esat.kuleuven.ac.be

Abstract

An hierarchical synthesis methodology for analog and mixed-signal systems is presented that fully in a novel way integrates topology selection at all levels. A hierarchical system optimizer takes multiple topologies for all the building blocks at each hierarchical abstraction level, and generates optimal topology combinations using multi-objective evolutionary optimization techniques. With the presented methodology, system-level performance trade-offs can be generated where each design point contains valuable information on how the systems performances are influenced by different combinations of lower-level building block topologies. The generated system designs can contain all kinds of topology combinations as long as critical inter-block constraints are met. Different topologies can be assigned to building blocks with the same functional behavior, leading to more optimal hybrid designs than typically obtained in manual designs. In the experimental results, three different integrator topologies are used to generate an optimal system-level exploration trade-off for a complex high-speed $\Delta\Sigma$ A/D modulator.

1. Introduction

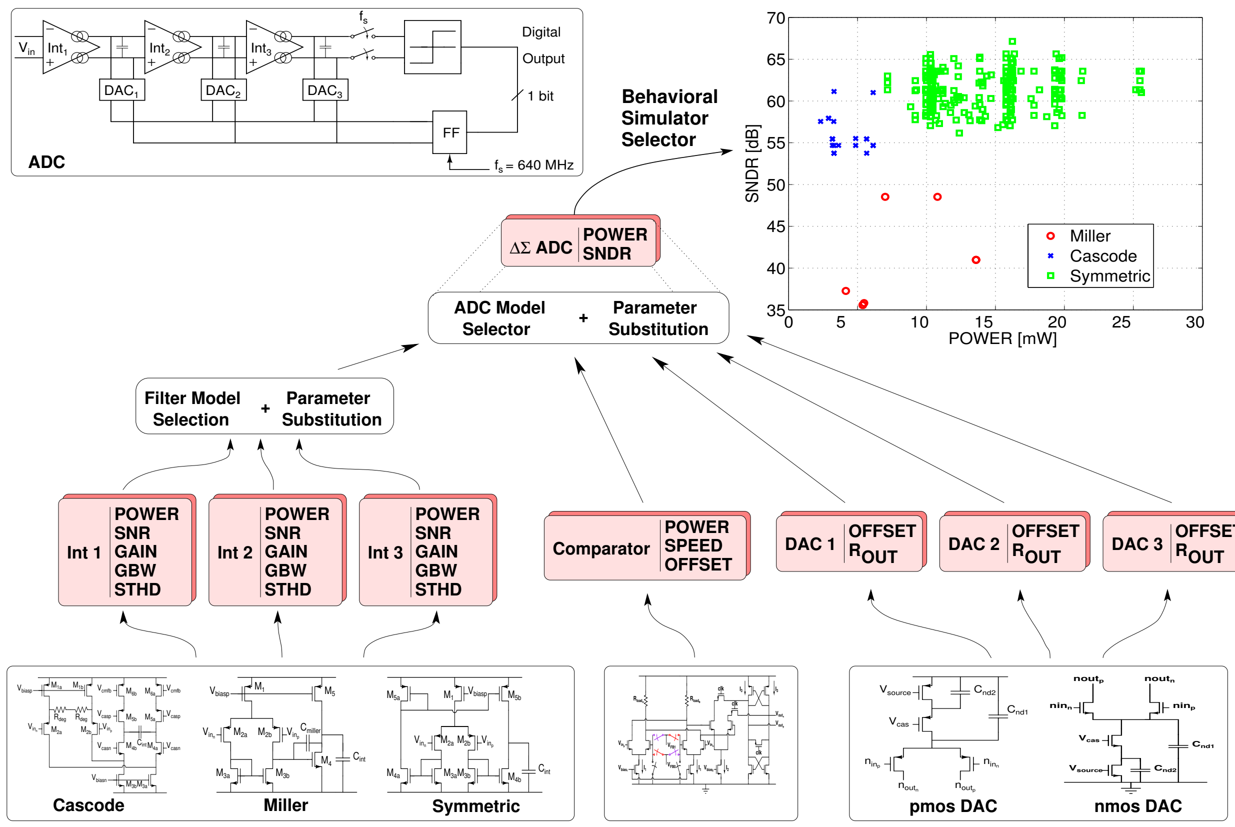
The paper describes a novel analog synthesis methodology for mixed-signal systems that fully integrates topology selection at all abstraction levels. It grants the analog designer access to the relation between the system-level performance behavior and the different building block topologies that can be used, for complex mixed-signal systems.

To categorize the existing analog synthesis tools, first two tasks in the synthesis process have to be distinguished; the *selection of a topology and the sizing* of that selected topology. For those two tasks we can also distinguish between tools that are designed for *circuit-level synthesis* and tools that are designed for *system-level synthesis*. For example, tools like IDAC, OPASYN, OASYS, ANACONDA, AMGIE,... (see [8]) are circuit sizing tools where the topology is selected automatically or interactively, and is sized according to given performance specifications. With tools like SEAS and DARWIN (see [8]) and the work of Koza [6], topologies can be generated or adapted during the sizing optimization process. However, these tools are targeted for circuits with a limited amount of design variables at the cell level e.g. integrators, comparators.

For the design of mixed-signal systems at higher abstraction levels e.g. data converters, the same distinction can be made between tools that select from a set of topologies (e.g. DAISY [4]), and recent methodologies that generate new system architectures using optimization techniques [7, 10]. These synthesis tools use a HDL language to describe the system and to generate new higher-level architectures.

The hierarchical system exploration methodology described in this paper introduces an efficient way to explore different analog system architectures down to the transistor level where topology selection at all hierarchical abstraction levels is fully integrated. With the methodology a set of optimal system-level performance samples are generated, where all the design points in this set contain information on the transistor-level topology combinations that are used to achieve the corresponding performance specifications. It is important to stress that the methodology explores the entire search space of design variables and circuit topologies at all levels. The methodology has the following novel features:

- The designer's insight in the relation between the system performance behavior and the selection of building block topologies at all levels is greatly improved.
- Using evolutionary optimization techniques, hybrid combinations of functionally the same building blocks can be generated, leading to more optimal designs than typically obtained in manual designs where often the same block is reused for reasons of convenience and time pressure.
- Critical connection constraints from higher abstraction levels are efficiently taken into account during the lower-level topology selection process. This makes the methodology practically applicable as opposed to methodologies where sub-blocks are optimized separately and future inter-block constraints are difficult to satisfy.



Vin
Int1
Int2
Int3
DAC1
DAC2
DAC3
fs
Digital Output
1 bit
FF
fs = 640 MHz
ADC
Behavioral Simulator Selector
SNDR [dB]
POWER [mW]
Miller
Cascode
Symmetric
ΔΣ ADC
POWER
SNDR
ADC Model Selector + Parameter Substitution
Filter Model Selection + Parameter Substitution
Int 1 POWER SNR GAIN GBW STHD
Int 2 POWER SNR GAIN GBW STHD
Int 3 POWER SNR GAIN GBW STHD
Comparator POWER SPEED OFFSET
DAC 1 OFFSET ROUT
DAC 2 OFFSET ROUT
DAC 3 OFFSET ROUT
Cascode
Miller
Symmetric
pmos DAC
nmos DAC

Figure 1. Hierarchical decomposition of the high–speed Delta–Sigma A/D converter (top left). Three integrator topologies can be selected at the lowest abstraction level (cascode, miller, symmetric), as well as two DAC topologies.

- The generated system-level performance trade-off samples also determine all the design variables of all building blocks at all abstraction levels in the hierarchically decomposed system. This is the result of the bottom-up way of generating the performance samples as proposed in [3].

This paper is organized as follows. In Section 2, a general description of the presented methodology is given. A $\Delta\Sigma$ modulator (top left in Figure 1) is used as illustrative example. Section 3 describes how a hierarchically decomposed electronic system is mapped onto the data structure of evolutionary algorithms (EA's), and explains how the described features are implemented in the algorithm. Section 4 explains how the experiments were setup, and section 5 presents the results and illustrates the major contributions of the methodology. Section 6 gives some conclusions.

2. Methodology Outline

In this section the general idea of the methodology is described and illustrated in Figure 1 for a single-bit third-order continuous-time $\Delta\Sigma$ modulator (top left architecture).

In order for a synthesis methodology to handle complex mixed-signal systems, first a hierarchical decomposition into smaller, less complex sub-blocks is performed. For our modulator example, this results in a 7-block decomposition (lower shaded blocks): a filter which contains three integrators, a comparator, and three D/A converters belonging to the three integrators respectively.

The proposed methodology is an important extension to the MOBU optimization methodology described in [3]. MOBU provides an efficient way to generate Pareto-optimal performance trade-offs at system level; first the Pareto-optimal performance trade-offs of the systems lower-level building blocks are generated using evolutionary optimization techniques, and then this performance information is propagated bottom-up through the hierarchical tree to generate the optimal trade-off at the system-level. The hierarchical decomposition and the building block topologies are chosen manually in the MOBU methodology, and only one topology per building block could be handle because MOBU can not handle the simultaneous search in multiple design spaces.

With the extension presented here, building blocks at all levels of abstraction can be implemented with different topologies. In his search for optimal system-level performance points, the evolutionary optimizer combines different lower-level designed topologies (selected from pregenerated Pareto fronts). Therefore, the lower-level design spaces change dynamically, as well as the higher-level behavioral models (if different architectures at intermediate levels are selected), and the dedicated simulators corresponding with them. An extra engine was designed to cope with the, at run-time, changing topology combinations and changing inter-block constraints (section 3.4). The bottom row in Figure 1 shows the topology set for each lowest-level building block for our example. At this level, circuit simulators (e.g. Spice, Eldo, Spectre) can be used to generate optimal design information in design and performance spaces. At higher abstraction levels, dedicated behavioral simulators can be used. As illustrated in Figure 1, after the determination of a (hybrid) combination of topologies for the integrators, a filter model has to be selected which complies with the resulting architectural filter structure.

At ADC level again a behavioral model and a dedicated simulator have to be selected, according to the lower-level structural combination of topologies. The resulting designs in the SNDR-POWER space in the example of Figure 1 depict only non-hybrid forms to show the topology comparison strength of the methodology. In Section 5, Figure 5, the optimal intermediate hybrid designs are shown and discussed.

The system-level designs are generated in a bottom-up way using transistor-level Pareto-optimal design samples. So, next to the gain in designer's insight, this methodology also immediately provides access to all design variable values at all levels, as opposed to methodologies where approximate building block performance models are used [2].

3. Automatic Topology Selection

In this section, critical implementation features are discussed: how a hierarchically decomposed system is mapped onto the data type of evolutionary algorithms, a novel multi-dimensional sorting engine is described, how the search towards optimal hybrid topology combinations is improved through a clustering mechanism, and finally how the dynamically changing inter-block constraints are handled.

3.1. Data Type Mapping

An efficient data type mapping is chosen to make the topology selection possible. Each individual of the EA represents a possible design solution. An individual contains a set of chromosomes where one chromosome contains all design variables of the observed top-level abstraction layer, and the other chromosomes contain information on the chosen lower-level sub-block configurations. For example for the ADC from Figure 1, one chromosome contains system-level design variables like signal input amplitude and frequency (V_{IN}, f_{IN}), and oversampling ratio (OSR). The other chromosomes contain information on the selected sub-block configurations. At the lowest abstraction level, there are no sub-blocks, so only the chromosome containing the design variables is present in the individual. As a result of this data type mapping, the evolutionary *cross-over operator* (i.e. interchanging chromosomes between individuals) means switching complete designed block topologies between system configurations. Applying the *mutation operator* on the configuration chromosomes, means stochastically shifting the current sub-block configuration to a nearby configuration in the performance space. Thus, the optimizer needs to have the knowledge about sub-block design points in the neighborhood of the current design. Therefore a sorting algorithm is added to the optimizer.

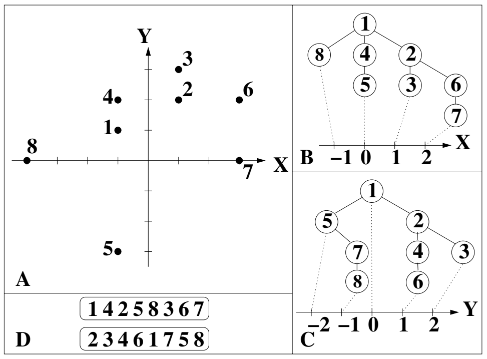


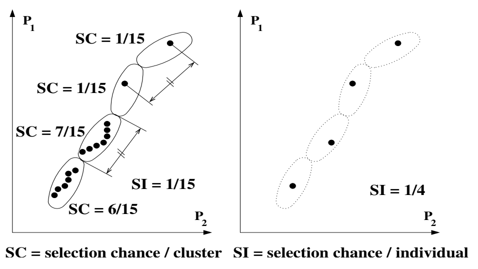
Figure 2. Sorting mechanism: A) sample points, B) X-Tree, C) Y-Tree, D) order with respect to point 1 and 2.

3.2. Sorting of the building block designs

The design space of a system at higher abstraction level is built from the design variables and the set of already designed solutions for each lower-level building block [3]. During optimization, for each individual, a lower-level design is selected for each building block, and during mutation a jump to a *nearby* design point is made. To make the optimizer aware of what the *nearby* points are for a certain point a sorting algorithm was developed, based on techniques for nearest neighbor search in high dimensions [5, 1].

Figure 2 illustrates for a 2D performance space (X,Y). During initialization of the hierarchical system, for each building block, the Pareto-optimal designs are put into sorted trees, one for each dimension (B, C in Figure 2). Each Pareto-optimal design point is then aware of its predecessor and successor for each dimension. During optimization, these single-dimensional orderings are used for the euclidean sorting of points. As opposed to a normal euclidean sorting where the performance coordinates of the points are used, these dimensional mappings induce two important benefits:

- Designs that only differ from the current design in one performance dimension are favored for the nearest neighbor search compared to designs that differ in multiple performances. Observe Figure 2.D, the ordering with respect to point 1: points 5 and 8 are considered 'closer' than point 3, because they differ less from point 1 if separate dimensions are considered.
- During optimization a bias can be induced towards certain building block performance regions, using the knowledge of the separate performance orderings. This benefit can be used to steer the optimizer and speed up the optimization.



P1
SC = 1/15
SC = 1/15
SC = 7/15
SI = 1/15
SC = 6/15
P2
P1
SI = 1/4
P2
SC = selection chance / cluster
SI = selection chance / individual

Figure 3. Clustering mechanism improves the selection chance of new far away individuals.

3.3. Clustering

As can be seen in Figure 1 (top right), there is a clear separation between performance spaces of non-hybrid topology combinations (one integrator topology present). To improve the search capability of the optimizer towards hybrid combinations which give intermediate performance results (Figure 5), clustering is included in the optimizer [11]. When the EA selects individuals for recombination from an unclustered set of individuals (left plot in Figure 3), the chance for selecting from a region where already a lot of points are generated is bigger then selecting points far from each other. This degrades the exploration capabilities. Clustering solves this by representing clusters of individuals by one central point (right plot in Figure 3). The cluster radius is determined by dividing the maximum distance found between two points in the population, by the largest distance between 2 consecutive points.

3.4. Sub-block interconnect constraints

A very important aspect of system design that has to be tackled is the handling of inter-block constraints between connected sub-blocks on a higher hierarchical level. As described in [3], inter-block constraints have to be taken into account when a system-level design tool, using transistor-level performance information, is developed. In [3], a set of inter-block constraints could be defined beforehand, and during optimization lower-level designs are selected keeping in mind these constraints. But when the architectures are selected during optimization, the set of inter-block constraints changes accordingly. Also, the repercussion of the inter-block constraints is different for each topology. A third extension to the MOBU methodology is the implementation of an *extra engine* that works on an intermediate level, invisible for the optimizer. During initialization all the inter-block constraints at each abstraction level are defined to this engine. These constraints are defined for the black box representation of the sub-blocks and are independent of the architecture that was implemented. During optimization, the engine will function as a guard: if the optimizer asks for a design point during selection or mutation, the engine will look at the selected topology and translates the inter-block constraints to constraints on the topology. The engine will then select a design point according to what the optimizer asks, but keeping in mind the constraints. The engine can be seen as a kind of top-down feasibility-bounds translator during the bottom-up traversal of Pareto-optimal performance information.

4. Experimental Setup

This section describes the setup of the experiment that we performed to validate our methodology. The next section presents the results. In the experiment, performance trade-offs are generated for the system topology of Figure 1: a single-bit third-order continuous-time $\Delta\Sigma$ modulator for 802.11a/b/g WLAN in a 0.18 μm standard CMOS technology. The experiment is twofold:

- First the optimizer is used to generate system designs where the topologies of all integrators are chosen to be the same. The optimizer can still select from the set of integrator topologies, but it has to be the same for all integrators. The results prove the exploration capabilities of the method.
- For the second part the optimizer can make all kinds of topology combinations. The result shows a Pareto-optimal trade-off which contains hybrid architectural combinations that represent intermediate Pareto-optimal design points.

For the higher-level optimization, behavioral models are developed for the different topologies of all sub-blocks. At this level, the topology selection of the sub-blocks is driven by the power and accuracy constraints of the overall system.

Three well-known integrator topologies are considered: a folded-cascode opamp with internal source degeneration, a two-stage Miller amplifier and a symmetrical OTA. These circuits are implemented using pmos differential input pairs to lower the DC offset voltages. Furthermore, a single-bit comparator is considered that consists of two preamplifiers, a current stage and a regenerative latch [9]. Finally, current-steering D/A converters are implemented as both nmos and pmos topologies.

4.1. Low-level description of the building-blocks

One of the necessities for a proper setup of the experiment is the knowledge of the Pareto-optimal performance trade-offs of the transistor-level sub-blocks. The generation of these trade-offs does not belong to the contribution of this work, but they are needed to prove the proposed methodology. This section shortly describes the setup for the Eldo simulations of the different topologies of each block.

The **design variables** of the topologies of each sub-block include the gate-overdrive voltages, lengths and finger widths of all transistors. Also the biasing currents, common-mode voltages (V_{cm}), resistances and capacitances are defined. There are **constraints** on the operation mode of most transistors to keep them in saturation. The **performance variables** *for the integrators* are the same for each topology and are extracted during circuit simulations in Eldo.The variables *for*

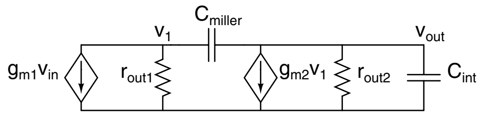


Figure 4. Model of Miller opamp.

the integrators are DC gain (*Gain*), gain-bandwidth (*GBW*), phase margin (*PM*), signal-to-noise-ratio (*SNR*), signal-to-distortion-ratio (*STHD*) and power consumption (*Power*). *For the comparator*, the offset voltage of the input transistors (V_{offset}), the regeneration speed (*Speed*) of the latch and the power consumption are extracted. Finally, the non-dominant poles (f_{nd1} & f_{nd2}) and DC value of the output impedance define the performance *of the D/A converters*.

4.2. System-level description of the ΔΣ modulator

Corresponding with the data type mapping described in subsection 3.1, at system level, a distinction is made between design variables belonging to this level, and the configuration selection variables indicating which sub-block configuration is selected.

Design Variables. Constant values are given for the system-level design variables according to the 802.11 communication standard. The design variables are: the oversampling ratio ($OSR = 32$), the amplitude of the input signal ($A_{in} = 300mV$), and the sampling frequency ($f_s = 640MHz$). The number of integrators in the modulator topology is constant and set to 3.

Sub-block Configuration Variables. The design variables are extended by the configuration variables belonging to each sub-block, indicating which sized topology is chosen from the lower-level trade-off. For the system-level simulations, a behavioral model for the filter is selected and filled in in the model of the system architecture (upper white blocks in Figure 1). A second-order behavioral model is generated for each topology. The parameters are substituted by the transistor-level simulation results corresponding with the selected lower-level design point. As an example, the model of the Miller integrator is shown in Figure 4. Similar models are developed for the other blocks. As a consequence, multiple design spaces are defined at this abstraction level. An extra engine was developed in this methodology that efficiently manages these heterogeneous design spaces (section 3.4).

Top-down feasibility bounds. Resulting from the modulator architecture and the 802.11 communication standard for which it has to be designed, the system-level performance constraints can be translated into requirements on *SNR*, *STHD*, *Gain*, *PM* and *GBW* of the integrators and on the *Speed* and V_{offset} of the comparator and D/A converters.

Next to these *block constraints*, the extra engine we included (subsection 3.4) will handle the *inter-block constraints* between the variables of the different sub-blocks. Among these constraints (Table 1), uniformity between the common-mode voltages of the connected blocks is included. Therefore, mainly D/A converters employing a pmos implementation are selected because of a better tuning with the V_{cm} of the integrators. For stability issues the non-dominant poles of the D/A converters are required to be 5 times higher than the *GBW* of the connected integrators [9]. Some of these constraints can be used during a pre-selection (section 5.1), the others will be watched at run-time by the extra engine.

Table 1. Inter-block constraints.

	Integrator	Comparator	D/A converter
Integrator	$V_{cm,int i}$=$V_{cm,int j}$	$V_{cm,int}$=$V_{cm,comp}$	$V_{cm,int}$=$V_{cm,DAC}$ $i_{int} = i_{DAC}$ $R_{out,int} = R_{out,DAC}$ $f_{nd1,2} > 5 * GBW_{int}$

Note that these constraints sometimes depend on the topology of the selected sub-blocks. For example, the current to be fed back to the folded-cascode circuit is defined by the g_m of its input pair, while for the Miller opamp this current is related to the input voltage as $g_{m1}*g_{out1}/g_{m2}$. In total, there are 20 block constraints and 14 inter-block constraints.

5. Experimental results

In the first experiment, all the integrators have the same topology. The second experiment demonstrates the increased optimized performance due to hybrid topology combinations at lower hierarchical levels.

5.1. Equal Integrator Topologies

Before the system optimization can start, first the Pareto-optimal trade-offs of the integrators, comparator and D/A were generated. For each of the 5 sub-block, a different computer processor was used. It took about 30 hours in total to generate the 5 trade-offs, using more processors in parallel would of course decrease that time. As mentioned in the previous section, with the constant block constraints resulting from the systems specs an initial selection of design points can be made for each sub-block: 347 candidates were selected for the first integrator, 75 for the second integrator and 41 for the third integrator are selected. For the comparator and D/A converters, 1598 and 2541 candidates were found. Taking the connection constraints between the sub-blocks into account, these numbers are lowered to 65, 17 and 7 for the three integrators. For the comparator and three D/A converters 632, 80, 28 and 11 selectable circuits are left. Table 2 shows the amount of circuit designs that are selected for each integrator topology. Note that different numbers of D/A converters are selected for each integrator because of the topology-dependent inter-block constraints. Clearly, the symmetrical OTA provides the most candidates (see Figure 1) because it combines an intrinsic high bandwidth with sufficient *Gain* for low *Power*. On the other hand, the Miller OTA seems to be a bad choice for implementation in a ΔΣ modulator for WLAN applications. This is due to the large small-signal currents in this block, requiring equivalent compensating currents in the D/A converter. Therefore, the transistors of the latter are en-

Table 2. Number of times each circuit is selected as topology in the final trade-off results.

	Symmetrical	Folded-cascode	Miller
Int1	52	9	4
DAC1	69	6	5
Int2	14	3	1
DAC2	21	4	3
Int3	4	2	1
DAC3	6	3	2

larged, which lowers the non-dominant pole frequencies of the output impedance. As a result, most of these integrators do not fulfill the appropriate inter-block constraint. In total, 169 designs were found that meet the specified WLAN accuracy of 9 bits. The power-accuracy trade-off of the generated modulators indicates that the folded-cascode topology is the best choice for a resolution of 9 to 10 bits. When more than 10 bits is required, the symmetrical OTA is preferred. It is clear that the Miller opamp does not meet the design specifications, illustrating the relevance of the inter-block constraints.

5.2. Hybrid Integrator Topologies

Allowing the optimizer to combine different integrator topologies results in new system-level Pareto-optimal architectures as opposed to the solutions of the previous section. Figure 5 shows system-level Pareto-optimal designs employing equal and hybrid integrator topology combinations. The flat parts of the curve prove that the overall accuracy is mainly determined by the performance of the first integrator. Varying the other integrators only slightly improves the *SNDR*. Only when the folded-cascode integrator is replaced by a symmetrical topology, the *SNDR* performance increases significantly. For this configuration, several Pareto-optimal architectures are found where the accuracy is defined by the first-stage symmetrical integrator, while the overall power consumption is reduced by the subsequent folded-cascode or symmetrical integrators. In total, the combination of different integrator topologies resulted in 12 additional Pareto-optimal designs, compared to 6 Pareto-optimal designs when only similar integrators are considered. This shows the huge impact of a hierarchical synthesis methodology that includes fully integrated building block topology selection.

6. Conclusions

In this paper a hierarchical synthesis methodology was presented that efficiently integrates building block topology selection at all levels. The methodology provides a system-level Pareto-optimal performance trade-off in which the samples contain information on the used lower-level topologies for the building blocks. The methodology was applied to a high-speed Delta-Sigma A/D converter designed for the 802.11a/b/g WLAN standard. From the trade-off results it

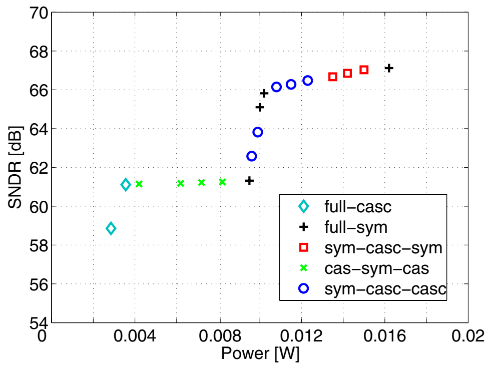


Figure 5. Pareto-optimal trade-off with topology selection.

is easy to examine the impact of different lower-level topologies on the systems performance behavior. The methodology allows hybrid combinations of topologies for the building blocks with similar functionality. It is shown that more optimal designs can be generated this way. As a result of the bottom-up hierarchical generation of the designs, a complete trade-off set of system designs is offered to the designer which all depict completely sized system architectures.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[2] F. De Bernardinis, P. Nuzzo, and A. L. Sangiovanni-Vincentelli. Mixed Signal Design Space Exploration through Analog Platforms. In *Proceedings Design Automation Conference*, pages 875–880, 2005.

[3] T. Eeckelaert, R. Schoofs, G. G. E. Gielen, M. Steyaert, and W. Sansen. Hierarchical Bottom–up Analog Optimization Methodology Validated by a Delta–Sigma A/D Converter Design for the 802.11a/b/g Standard. In *Proceedings Design Automation Conference*, pages 25–30, June 2006.

[4] K. Francken and G. G. E. Gielen. A high–level simulation and synthesis environment for $\Delta\Sigma$ modulators. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 22(8):10049–1061, Aug. 2003.

[5] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proc. the ACM Symposium on Theory of Computing*, Feb. 1997.

[6] J. Koza, F. H. Bennett, D. Andre, M. A. Keane, and D. F. Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. *Transactions on Evolutionary Computation*, 1(2):109–128, 1997.

[7] E. Martens and G. G. E. Gielen. Top-Down Heterogeneous Synthesis of Analog and Mixed-Signal Systems. In *Proceedings Design Automation and Test in Europe Conference*, pages 275–281, 2006.

[8] R. A. Rutenbar, G. G. E. Gielen, and B. A. A. Antao. *Computer–Aided Design of Analog Integrated Circuits and System*. John Wiley & Sons, 2002.

[9] R. Schoofs, M. Steyaert, and W. Sansen. A 7.5mW, 11-bit Continuous-Time Sigma-Delta A/D Converter for WLAN Applications. In *ISCAS*, pages 4419–4422, May 2006.

[10] H. Tang and A. Doboli. High-Level Synthesis of $\Delta\Sigma$ Modulator Topologies Optimized for Complexity, Sensitivity, and Power Consumption. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 25(3):597–607, Mar. 2006.

[11] E. Zitzler and L. Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology, Zurich, Switzerland, May 1998.