Hierarchical Bottom–up Analog Optimization Methodology Validated by a Delta–Sigma A/D Converter Design for the 802.11a/b/g Standard

Tom Eeckelaert, Raf Schoofs, Georges Gielen, Michiel Steyaert, Willy Sansen
Katholieke Universiteit Leuven, ESAT–MICAS
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
tom.eeckelaert@esat.kuleuven.ac.be

ABSTRACT

This paper describes key points and experimental validation in the development of a bottom–up hierarchical, multi–objective evolutionary design methodology for analog blocks. The methodology is applied to a continuous–time $\Delta\Sigma$ A/D converter for WLAN applications, to generate a set of Pareto–optimal design solutions. The generated performance tradeoff offers the designer access to a set of optimal design solutions, from which the designer can choose a satisfactory design point according to the performance specifications. The presented method takes advantage of the Pareto–optimal performance solutions of the hierarchical lower–level sub–blocks to generate the overall Pareto–optimal set at system level. The way the lower–level performance tradeoffs are combined and propagated to higher hierarchical levels, is one of the major key points in the bottom–up methodology. The experimental results validate the methodology for a 7–block hierarchical decomposition of a complex high–speed $\Delta\Sigma$ A/D modulator for a WLAN 802.11a/b/g standard.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: Computer-aided design; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms

Algorithms, Design, Performance

Keywords

Hierarchical Synthesis

1. INTRODUCTION

The continuous growth of the electronics application field over the past years has led to a large demand for complex mixed–signal electronic systems. The major bottleneck in the design of these systems is the design time of the analog parts [6]. As a result, automated analog sizing has become an unavoidable solution. The research on efficient analog design automation methodologies has progressed to a vast collection of commercially available design tools. However, these tools are targeted for circuits with a limited amount of design variables at the cell level e.g. amplifiers. For the automated design of larger systems with many more design variables e.g. data converters, a generally applicable design methodolgy is not yet implemented in an industrial tool. This is largely because all the pieces of the system–level design puzzle are not fully in place. The challenge is to have a methodology that makes the complexity of the problem tractable.

According to the manner in which the design space is organized and traversed, different methodologies can be distinguished [5]. The design space of a complex system can be handled as a whole, where all the design variables can be taken into account during propagation of the performance space in search of a suitable design (also called a "flat" design). Or, it can be organized hierarchically into subspaces by hierarchically decomposing the complex system in smaller less complex building blocks. Different hierarchical methodologies exist [9, 1, 3, 13, 2, 7, 12, 4] depending on how the different subspaces are combined in search of a suitable design.

This paper elaborates on the implementation and experimental validation of the multi–objective hierarchical design methodology described in [4]. In this methodology, first, Pareto–optimal performance tradeoff samples are generated for the lowest–level hierarchical sub–blocks, using evolutionary optimization techniques. Then, these performance samples are propagated upwards in the hierarchy using the same evolutionary optimization techniques until a set of designs for the highest hierarchical level is generated, which is Pareto–optimal in the systems performance space. This paper describes the key implementation details of this Multi–Objective, Bottom–Up methodology (MOBU). Such as the sorting of the lower–level Pareto–optimal designs needed by the evolutionary optimizer to efficiently search the design space. And the handling of the block interconnection constraints when they are combined in a higher hierarchical block. The methodology is implemented and used to generate Pareto–optimal performance tradeoff samples for a

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

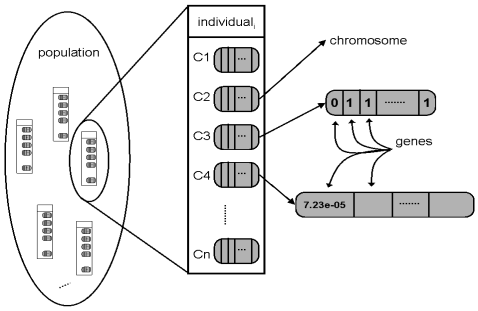

Figure 1: A population in the decision space contains individuals, each representing a solution in the decision space. The individuals are composed out of chromosomes, which are vectors of genes.

third–order single–loop continuous–time Delta–Sigma modulator. Out of the generated design set, one modulator is selected and compared with a transistor–level ("flat") manual design to meet the WLAN 802.11a/b/g communication standard.

This paper is organized as follows. First a general discription of the evolutionary optimization algorithm is given, and how the hierarchically decomposed system is fit in the data structure of the algorithm. Next, in Section 3, the upward propagation of the lower–level designs is handled and key implementation aspects are discussed. Section 4 describes the results of the multi–objective bottom–up design of a 3th–order single–loop continuous–time $\Delta\Sigma$ modulator for the WLAN 802.11a/b/g communication standard. Finally, some conclusions are drawn in Section 5.

2. EVOLUTIONARY OPTIMIZATION

Evolutionary algorithms (EA) are stochastic optimization methods designed by analogy with natural evolution [14]. Starting from a set (population) of solutions (individuals), subsequent sets are composed by means of selection (survival of the fittest) and variation (mating and mutating). The typical data structure for these algorithms is shown in figure 1: Each individual contains different chromosomes which carry the values for the decision variables as their genes.

As proposed by the authors in [4], we integrated the existing Strength Pareto Evolutionary Algorithm (SPEA) [15] in the implementation of our methodology. The typical algorithmic flow of such an EA looks like:

Initialization The decision space is sampled by random individuals in the initializing population.

Generation loop:

- **update Pareto set** An external set of Pareto–optimal solutions is updated by comparison with the newly found solutions. This way, Pareto–optimal solutions don't get lost. In the original SPEA algorithm the size of the external set is kept constant by means of a clustering algorithm.
- **fitness assignment** Related to the quality of the individual with respect to the targets, a real–valued number (fitness) is assigned to each individual. In SPEA the concept of Pareto dominance [4] is used to assign the fitness. Only the externally stored Pareto set determines the fitness of an individual. Domination by individuals in the population is irrelevant. In this step, each of the individuals in the population has to be compared with the individuals in the external Pareto set, for all objectives.
- **selection** Individuals are selected randomly to form a new set (mating pool). A good fitness value means favored for selection, but all can be selected!
- **recombination** From the mating pool, different pairs of individuals (parents) are used to recombine (exchanging chromosomes) into 2 new individuals (children). Recombination is an operator that can bring large diversity between the children and the parents. This induces global search capability.
- **mutation** New genes are introduced into the population by stochastically changing their values. Mutation is an operator that brings small changes to the genes. This induces local search capability.

Remark that the size of the externally stored Pareto–optimal set is kept constant in the original algorithm [15]. For our methodology it is better to keep all Pareto–optimal solutions, so clustering is not used in our implementation and the external set grows in size.

The data structure of the general evolutionary optimization algorithm is adapted to the way the set of sub–blocks of the hierarchical decomposed system is traversed: If a certain sub–block is not an element of the lowest hierarchical level, then it is depending on some lower–level sub–blocks. So, to represent this block, an individual consists typically out of a set of chromosomes which each hold a gene representing a sub–block design configuration, and one chromosome which holds a set of genes, one for each design variable.

Recombination of 2 individuals now consists of recombination of the 'config' chromosomes, and the recombination of the genes between the 'design variables' chromosome. Mutation of an individual consists of mutating the values of the genes in the 'design variables' chromosome, and mutation of the 'config' chromosomes. The latter means mutating to a design point 'near' the current design of the sub–block. To be able to jump from a point in the Pareto–optimal set to a 'nearby' point can only be done when a certain order is applied to the Pareto–optimal samples. This will be addressed in section 3.1.

3. ALGORITHM IMPLEMENTATION ASPECTS

With the multi–objective bottom–up (MOBU) methodology proposed in [4], Pareto–optimal sets in the performance space of complex mixed–signal electronic systems can be generated. The methodology uses hierarchical decomposition to divide the system in less complex sub–blocks. After the decomposition a Pareto–optimal design set can be generated for each lowest–level block using a multi–objective optimization algorithm in combination with a dedicated simulator. These cell–level Pareto–optimal sets can then directly be exploited for system–level design in a bottom–up way. Figure 2 shows the hierarchical decomposition of the $\Delta\Sigma$ system used in Section 4 to validate the methodology.

The design space for the next level up is the 'selection' of a design for each of the sub–blocks. A 'selected' sub–block design is actually pointing to a specific design from the lower–level Pareto–optimal tradeoff of that sub–block.

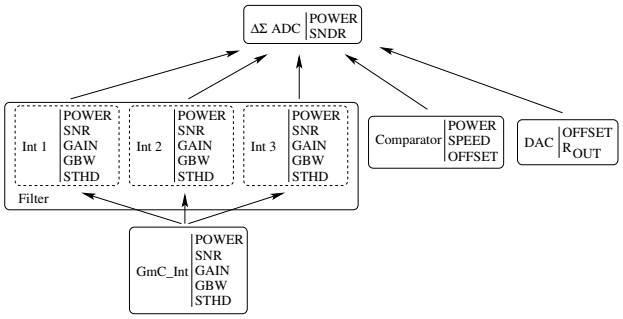

Figure 2: Hierarchical decomposition of the high–speed Delta–Sigma A/D converter from figure 3, used by the Multi–Objective Bottom–Up Methodology to propagate the generated performance tradeoffs upwards to combine into the Pareto–optimal performance tradeoff at the system level.

The hierarchy traversal proceeds in an upwards fashion, in the end providing an optimal system–level tradeoff. In the next section (3.1), sorting of the Pareto–optimal sub–block designs is discussed. This is an important aspect in the evolutionary search for optimal high–level designs.

3.1 Sorting of Pareto–optimal samples

In order for the evolutionary optimization algorithm not to lose its 'sense of direction', it should be able to find nearby points of a certain current design configuration. Therefore we implemented an efficient multidimensional sorting algorithm based on the algorithm in [8].

The sorting is based on building a tree of all the Pareto points for each performance dimension:

- Each Pareto point is represented by a vertex in the tree.
- Each vertex in the tree consists of 3 branches:
  1. one branch connected with a vertex representing a Pareto point that has a smaller value for that performance dimension.
  2. one branch connected with a vertex representing a Pareto point that has a larger value for that performance dimension.
  3. one branch connected with a vertex representing a Pareto point that has an equal value for that performance dimension.

It is now easy to find for each point the point that is closest, for each dimension. Ordering this set of points according to their euclidean distance from the current point, gives a local ordering of points for each point. For propagations (mutation steps) larger than the amount of locally ordered points, the tree comes into play to propagate further away.

This ordering of Pareto points of each lower–level sub–block can be done during the initialisation step, when the system with all its hierarchically decomposed sublocks is described to the optimizer. The amount of Pareto solutions of a sub–block is constant during the whole optimization process.

Ordering the Pareto–optimal designs of the sub–blocks is essential for the optimization because the evolutionary optimizer loses track when the local search of the mutation process results in very large changes (natural evolution would be messed up when the mating of 2 human beings resulted in a primate offspring).

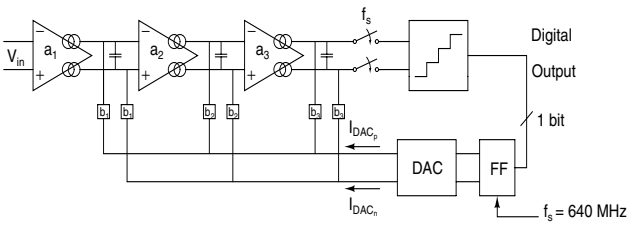

Figure 3: Third-order single-bit ΔΣ modulator.

3.2 Interblock Constraints

In the experiment from [4], no interactions between the sub–blocks was taken into account at higher hierarchical level. For a real design however, this assumption does not hold. In our ΔΣ application in section 4 for example, a mutual common–mode voltage is defined between sub–blocks. So, the selection of a Pareto–optimal design for each sub–block has to be biased towards a common–mode voltage set by the already selected sub–blocks.

So, when certain variables of sub–blocks have boundaries depending on other sub–blocks, this has to be taken into account during the lower–level optimizations. First of all, these mutual variables have to be used as design variables during the lower–level optimization. Also the sampling of those variables has to be dense enough to be able to find at least one solution at the higher–level.

4. EXPERIMENTAL VALIDATION

In this section we will illustrate and experimentally validate the presented methodology by generating the Pareto–optimal performance tradeoff of a 1–bit third–order continuous–time ΔΣ modulator for the WLAN 802.11a/b/g standard (Figure 3). All sub–blocks of the modulator followed a hierarchical decomposition scheme. This resulted in different models that describe the behaviour of each individual sub–block. The model parameters were extracted during transistor–level simulations of the circuits using Eldo. The generated models were used in high–level simulations of the sigma–delta modulator. The extracted optimal modulator is compared with a transistor–level design for the same WLAN communication standard. The objective was to reach a conversion accuracy of 60 dB within a signal bandwidth of 10 MHz. Therefore, these modulators can be implemented in a quadrature I/Q receiver architecture. Both modulators are designed in a 0.18 μm standard CMOS technology. The sampling speed is 640 MHz, resulting in an oversampling–ratio of 32. The performance of both circuits is determined by an fft of the transient digital output signal. This was done in MATLAB^{TM}, using 17600 time steps for an input frequency of 830 kHz.

The different analog building blocks are described next in detail. The discription at transistor–level and the model extraction is explained for the integrators, 1–bit comparator and D/A converters. Most attention was paid to the integrators of the single–loop low–pass filter because they determine mainly the overall accuracy and power consumption of the modulator.

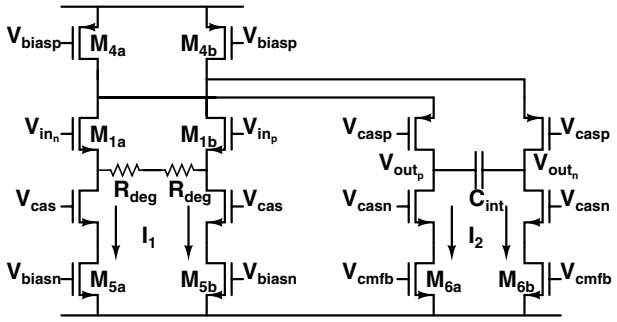

Figure 4: GmC integrator.

4.1 Integrator

A GmC integrator is chosen because it consumes less power than a RC integrator at high sampling speeds for the same accuracy, as stated in [10]. The architecture of the integrator is shown in Figure 4. Internal source degeneration is applied to improve the linearity. A folded–cascode structure is selected because a high DC gain is required to suppress the quantization noise in the 10 MHz signal band.

- The performance variables for this circuit are power consumption ($Power$), DC gain ($Gain$), gain–bandwidth (GBW), phase margin (PM), signal–to–noise–ratio (SNR) and signal–to–distortion–ratio ($STHD$). They are all extracted from Eldo transistor–level simulations.

- The design variables include the biasing currents in both branches (I_1 & I_2), the gate overdrive voltages of all transistors (V_{gstx}), the lenghts of all transistors (L_x), the common–mode voltage (V_{cm}), the finger width (W_{finger}), the degeneration resistance (R_{deg}) and the integration capacitance (C_{int}). The output V_{cm} is chosen equal to the input V_{cm}. The supply voltages are fixed to $V_{dd} = 1.8$ V and $V_{ss} = 0$ V. They determine the upper and lower boundaries in the calculation of the applied voltages (V_{cm} & V_{gstx}). The values for the other variables cover a wide range of at least two orders of magnitude.

- The constraint variables are the drain–source voltages of all transistors (V_{dsx}). They are measured to check if all transistors operate in the saturation region.

Analysis of the performance and design variables shows an expected trade–off between the R_{deg}, SNR and $STHD$ (Figure 5). The degeneration resistor enhances the linearity of the input transconductance, but also generates thermal noise. There is a third–order relationship between the $STHD$ and R_{deg}, while the SNR varies linearly with it. It can be seen that there is a value where the SNR and $STHD$ curves cross. This resistance results in the best performance of the integrator. It will be selected by the tool when the Pareto–optimal performance tradeoff design set is created.

The second–order model of the integrator is shown in Figure 6. All model parameters are extracted or calculated from transistor–level simulations. The obtained GBW, $Gain$ and PM define the model parameters g_m, r_{out}, C_{int}, r_{in} and C_{in}. Simulations of the model demonstrate identical small–signal behavior as the corresponding circuit.

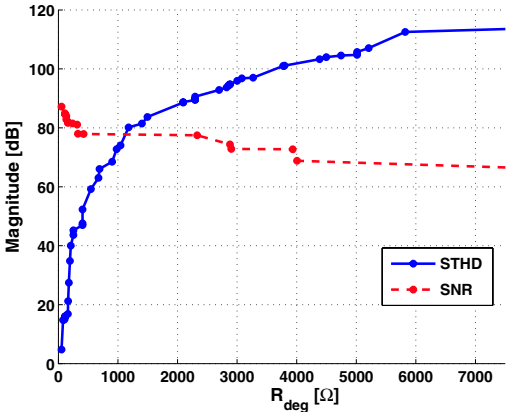

Figure 5: Trade–off between SNR and STHD.

4.2 Comparator

The comparator in Figure 7 is built of two preamplifiers, a current stage and a regenerative latch. When the clock is high, the differential input voltage is amplified and a small offset voltage is created between the output nodes of the latch. During the regeneration phase, i.e. when the clock is low, the output nodes are pulled towards the supply or ground voltage. The switches between the current stage and latch prevent the preamplifier from kick–back effects.

- The performance variables for the comparator are the mismatch offset voltage of the input transistors (V_{offset}), the regeneration speed of the latch ($Speed$) and the power consumption ($Power$). They are extracted during transient simulations of the circuit. Here, both small and large differential input signals are used to calculate the regeneration speed.

- The design variables are the biasing currents (I_1 & I_2), the latch current (I_3), the gate overdrive voltages of all transistors (V_{gstx}), the lengths of the transistors (L_x), the finger width (W_{finger}), the common–mode voltage (V_{cm}) and the load resistance of the preamplifiers (R_{loadx}). There are no critical limits for these variables since the aim is to characterize comparators for different kinds of systems, requiring various speeds and offset voltages.

- As in the case of the integrator, also here the constraint is that the transistors of the preamplifiers operate in saturation mode. Therefore, the drain–source voltages (V_{dsx}) are checked.

Based on the performance variables a model is implemented that has the same transient behaviour as the transistor–level circuit. Therefore, it can be used at the high–level analysis of the sigma–delta modulator while it reduces the simulation time.

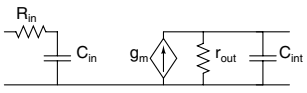

Figure 6: Integrator model.

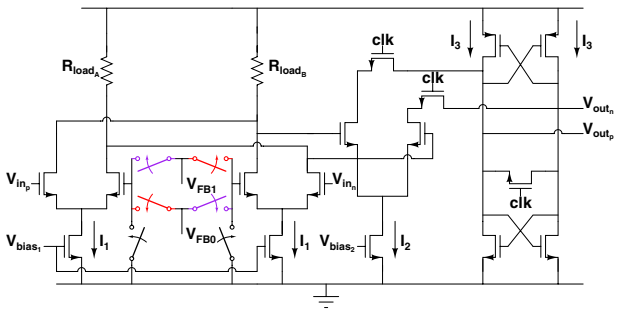

Figure 7: Comparator topology.

4.3 D/A converter

A switched current source topology is used, as shown in Figure 8. According to [11], care has to be taken with the location of the non–dominant poles of the output resistance of the converter. The pole frequencies need to be 5 times higher than the GBW of the corresponding integrator in order to prevent the modulator from instable behaviour.

- The performance variables are the location of these non–dominant poles (f_{nd1} & f_{nd2}) and the DC output impedance. The first pole is defined by the impedance at the drain node of the cascode transistor and the parasitic capacitance on this node. The second one by the impedance and parasitic capacitance at the drain node of the current source.
- The current, the finger width, the common–mode voltage, the gate overdrive voltages and the lenghts of the transistors are the chosen design variables. Values for the boundaries of these variables are determined from design experience. Again, a broad spectrum of different converters is covered.
- The current source and the cascode transistor have to operate in saturation mode in order to achieve a high output impedance. At the modulator–level, a check will be done on the ratio between the pole frequencies and the GBW of the integrator.

A model is extracted that behaves within 1% accuracy of the extracted performances. This model is used for each of the three converters during the high–level simulations.

4.4 $\Delta\Sigma$ modulator

The models derived from the building blocks are used in the high–level ELdo simulations of the modulator. Therefore, simulation time is reduced. Each set of model parameters defines an accurate representation of the corresponding

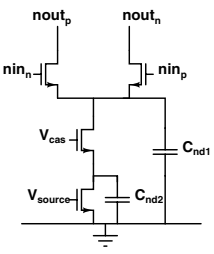

Figure 8: D/A converter.

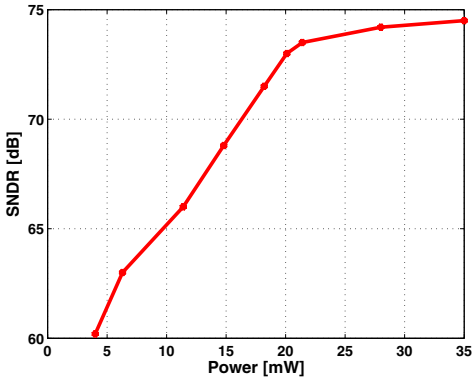

Figure 9: SNDR and power consumption of optimized modulators.

circuit. The information in the description file of the modulator is listed below:

- The performance variables chosen for the exploration are power consumption ($Power$) and signal--to--noise--and--distortion--ratio ($SNDR$).
- There are 24 design variables at the modulator level. They include the $Gain$, GBW, PM and V_{cm} of the different integrators and the V_{offset} and $Speed$ of the comparator and D/A converter.
- There are constraints for the variables of each block, but also between the variables of the different blocks. Among the first ones, requirements are put on SNR, $STHD$, $Gain$, PM and GBW of the integrators and on the $Speed$ and V_{offset} of the comparator and D/A converter. The latter ones include a constraint on the V_{cm} of the consecutive sub–blocks. Because the output common voltage of one block defines the input common–mode voltage of the next block, an offset voltage of maximal 2 percent is allowed. The locations of the non–dominant poles of the D/A converters are also constrainted. As mentioned before, these poles has to be at least 5 times higher than the GBW of the related integrators. In total, there are 22 constraints.

After examination of the constrainst set on the sub–blocks, 238 candidates for the first integrator, 159 for the second integrator and 136 for the third integrator are selected. For the comparator and D/A converter, 898 and 1656 candidates were found. Next, the interconnection constraints between the sub–blocks are evaluated, resulting in 9 designs were found that meet the specified requirements.

The selected $\Delta\Sigma$ modulator was optimized towards the required $SNDR$ of 60 dB. Therfore, the power consumption was minimized to 4 mW. A trade–off between the accuracy and power consumption of the valid modulators is shown in Figure 9. It shows that the accuracy saturates for high power consumptions. This is because it is limited by the quantization noise, which is independent of the power. A comparison between the selected modulator and the manually designed A/D converter is presented in Table 1. They both meet the specifications for the WLAN 802.11a/b/g standards. The optimized modulator achieves a power saving of 25% compared to the manually designed modulator. Notice that most power is saved in the first integrator, which

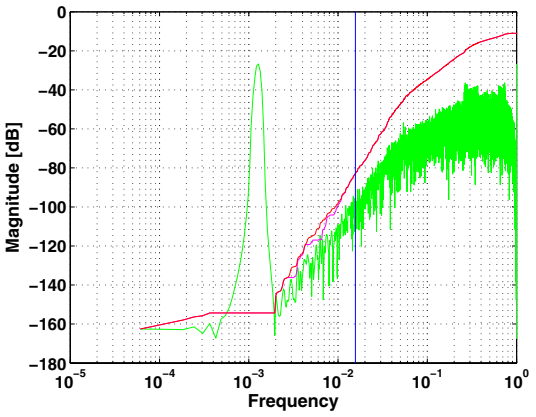

Figure 10: Output power spectrum density.

determines the overall performance of the modulator. During the three–level hierarchical decomposition process 19215 circuits were evaluated in 4 days, while it took 5 weeks to implement the reference design at transistor–level. The Matlab output spectrum with the integrated noise and harmonic distortion suppression of the final selected design is shown in Figure 10.

5. CONCLUSIONS

In this paper the Multi–Objective Bottom–Up (MOBU) design methodology from [4] was validated by generating a system–level Pareto–optimal performance tradeoff set for a high–speed Delta–Sigma A/D converter designed for the 802.11a/b/g WLAN standard. A multi–dimensional sorting algorithm was designed to handle the 'smart' selection of lower–level hierarchical sub–block designs during the evolutionary optimization process. Another key implementation detail that was handled, is the block interconnection constraints when they are combined in a higher hierarchical block. Experimental results validate the methodololgy for the 7–block hierarchical decomposition of the complex $\Delta\Sigma$ A/D modulator, by comparing the optimizer results to a manually designed modulator.

Table 1: Comparison and results of optimized Sigma–Delta modulator.

		Optimized	Reference
Int_1	Gain	49 dB	50 dB
	GBW	21.8 MHz	26 MHz
	PM	81.7°	80°
	Power	1.55 mW	2.5 mW
Int_2	Gain	45 dB	50 dB
	BW	39.6 MHz	32 MHz
	PM	85.5°	92°
	Power	0.95 mW	1.1 mW
Int_3	Gain	51.7 dB	50 dB
	GBW	50 MHz	53 MHz
	PM	105°	95°
	Power	1.2 mW	1.45 mW
Comparator	Speed	3 GHz	3 GHz
	V_{offset}	0.6 mV	1 mV
	Power	0.3 mW	0.35 mW
D/A converters	V_{offset}	< 9 mV	< 10 mV
Modulator	SNDR	60.2 dB	60.5 dB
	Power	4 mW	5.4 mW

6. REFERENCES

[1] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, A. Malvasi, A. L. Sangiovanni-Vincentelli, and I. Vassiliou. *A Top–Down, Constraint–Driven Design Methodology for Analog Integrated Circuits*. Kluwer Academic Publishers, 1997.

[2] A. Doboli, N. Dhanwada, A. Nunez-Aldana, and R. Vemuri. A Two–Layer Library–Based Approach to Synthesis of Analog Systems from VHDL–AMS Specifications. *ACM Transactions on Design Automation of Electronic System*, 9(2):238–271, Apr. 2004.

[3] S. Donnay, G. G. E. Gielen, W. Sansen, W. Kruiskamp, D. Leenaerts, S. Buytaert, K. Marent, M. Buckens, and C. Das. Using Top–Down CAD Tools for Mixed Analog/Digital ASIC's: a Practical Design Case. *Journal of Analog Integrated Circuits and Signal Processing*, 10:101–117, 1996.

[4] T. Eeckelaert, T. McConaghy, and G. G. E. Gielen. Efficient Multiobjective Synthesis of Analog Circuits using Hierarchical Pareto–optimal Performance Hypersurfaces. In *Proceedings Design Automation and Test in Europe Conference*, pages 1070–1075, 2005.

[5] G. G. E. Gielen, T. McConaghy, and T. Eeckelaert. Performance Space Modeling for Hierarchical Synthesis of Analog Integrated Circuits. In *Proceedings Design Automation Conference*, pages 881–886, June 2005.

[6] G. G. E. Gielen and R. A. Rutenbar. Computer–aided design of analog and mixed–signal integrated circuits. *Proceedings of the IEEE*, 88(12):1825–1852, Dec. 2000.

[7] R. Harjani and J. Shao. Feasibility and Performance Region Modeling of Analog an Digital Circuits. *Journal of Analog Integrated Circuits and Signal Processing*, 10(1):23–43, June 1996.

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

[9] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, and J. R. Hellums. A Case Study of Synthesis for Industrial–Scale Analog IP: Redesign of the Equalizer/Filter Frontend for an ADSL CODEC. In *Proc. Design Automation Conference*, pages 1–6, 2000.

[10] R. Schoofs, M. Steyaert, and W. Sansen. A GmC filter design methodology for high-speed continuous-time A/D converters in a deep sub-micron technology. In *VLSI Circuits and Systems II Conf., Sevilla*, May 2005.

[11] R. Schoofs, M. Steyaert, and W. Sansen. Current-Steering DAC Design for High-Speed Continuous-Time Sigma-Delta A/D Converters. In *DCIS'05, Lisbon*, Nov. 2005.

[12] G. Stehr, H. Graeb, and K. Antreich. Feasibility Regions and their Significance to the Hierarchical Optimization of Analog and Mixed–Signal Systems. *International Series of Numerical Mathematics*, 146:167–184, 2003.

[13] G. Van der Plas, G. Debyser, F. Leyn, K. Lampaert, J. Vandenbussche, G. G. E. Gielen, W. Sansen, P. Veselinovic, and D. Leenaerts. AMGIE — A Synthesis Environment for CMOS Analog Integrated Circuits. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 20(9):1037–1058, Sept. 2001.

[14] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, 1999.

[15] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *Transactions on Evolutionary Computation*, 3(4):257–271, Nov. 1999.