DSYN: A Module Generator for High Speed CMOS Current Output Digital/Analog Converters

Robert R. Neff, Paul R. Gray, and Alberto Sangiovanni-Vincentelli
Electrical Engineering and Computer Sciences
University of California, Berkeley

Abstract

DSYN generates optimized Digital/Analog Converter (DAC) layouts given a set of specifications including performance constraints, a description of the implementation technology, and a set of design parameters. The generation process consists of a synthesis step followed by a layout step. During synthesis a new constrained optimization method is coupled with combination of circuit simulation and DAC design equations. The layout step uses stretching and tiling operations on a set of primitive cells. Prototypes have been demonstrated for an 8-bit, 100-MS/s specification, driving a 37.5-ohm video load, and a static 10-bit specification, driving a 4mA full-scale output current. Both designs use a 5-V supply in a 1.2 μm CMOS process.[1]

Biographies

Robert R. Neff (member) was with the University of California, Berkeley, CA. He is now with Hewlett Packard Company, Palo Alto, CA 94304.

Paul R. Gray (fellow) and Alberto Sangiovanni-Vincentelli (fellow) are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 .

Coorespondence should be directed to:

Dr. Robert Neff
Hewlett Packard Company
3500 Deer Creek Road, MS 26U-4
Palo Alto, CA 94304-0867

Phone: (415)857-6220
Fax: (415)857-3637
E-mail: neff@hpl.hp.com

1. This research was supported by the Semiconductor Research Corporation under grant 94-DC-324.

I. Introduction

In mixed-signal integrated circuits, the analog part of the design often occupies a small portion of the physical die area but requires a disproportionately large amount of the design effort and time. Following a digital domain paradigm, analog circuit synthesis has been proposed to substantially reduce design time [1][2][3][4]. This approach is intended for a wide class of commonly used analog circuits, but as such does not incorporate specific knowledge of the particular device being designed. These methods have proven useful for application to the most common building blocks, such as amplifiers and comparators, which tend to have a relatively small number of large devices, and matching constraints between pairs, but not across large numbers of elements. For other, more complex blocks, the general synthesis approach is a poor match to the problem. A second approach consists of developing several synthesis tools, each optimized for a particular class of circuits, such as switched capacitor filters, opamps, or ADCs [5][6][7][8]. In either case, the goal is to generate blocks that are competitive with manual design in performance and die area. The best test of these analog circuit synthesis methodologies is to follow designs through the complete cycle, including design specific inputs and results from fabricated devices.

Generation of mixed-signal analog blocks can be approached in several ways. In the top-down, hierarchical approach, proposed in [9], constraint values are passed down through layers of the design, with the bottom layer corresponding to the transistors. This approach was initially applied to a current-switched DAC example, and has also been used to build an oversampled ADC [10]. The main goal of the hierarchical approach is to obtain reliably and quickly a design that satisfies all constraints. When in addition to satisfying constraints, a highly optimized circuit with respect to area and speed is required, then a "flat" methodology may prove more effective. In a flat approach many design variables are optimized at once against the complete design specification. When a complex circuit behavior is difficult to separate and evaluate hierarchically, a flat approach is the only option. Of course there are limits to our design optimization capabilities -- as the number of design variables increases, a flat method becomes more difficult.

In this paper a non-hierarchical approach is applied to the current-switched DAC function, allowing simultaneous optimization of architecture and device level design variables, and easy incorporation of all aspects of performance, including layout related parasitics when predicting static and dynamic performance.

II. Module Generation Methodology.

Fig. 1 illustrates the module generation methodology. Input data is separated into information classes, which are used during the optimization step to find the best set of design variables. The design variables are then passed to the layout program, creating the DAC module. Design optimization can be further decomposed into two parts, the optimization algorithm and the design estimation process.

The input data is partitioned into design constants, performance specifications, and design variables. Typically design variables are continuous variables, but in the case of DAC module generation there are several variables which must have *discrete* values. Two examples of this are the number of LSBs per DAC segment, which must be a power of 2, and the number of bias cells, which must be an integer. In addition to these “intrinsic” integer variables, we force device dimensions to fall on the $L_{min}/2$ grid required by our layout tools, so that all design variables are constrained to be integers.

The design estimation step predicts the performance of the circuit, including process variation, device mismatch, and accurate device modeling. In DSYN, the HSPICE circuit simulator performs equation evaluation as well as circuit simulation, where all performance estimates are computed in one multiple-simulation run. Three DC simulations were run to cover worst case bias margins and INL conditions, and two transient simulations were used for settling and glitch energy.

Design optimization for analog circuits is not a new topic [4][7][11], but DSYN’s requirements present some unique problems. The optimization process must work well with the circuit estimation method, in which function evaluations are time consuming and function evaluations in the infeasible region may not produce reliable gradients. The mixed integer formulation creates additional difficulties.

Direct approaches to this Mixed-Integer Non-Linear Programming (MINLP) problem using branch and bound algorithms [4] do not complete in a reasonable time when used with circuit simulation. The branch and bound algorithm formulates a Non-Linear Programming (NLP) problem as a sub-problem, and solves many of these sub-problems in the course of solving the MINLP problem. Using circuit simulation, the NLP sub-problem may take several hours to complete, and the branch-and-bound is exponential in the number of design variables, resulting in an optimization time in the weeks range for the complete MINLP problem.

A faster algorithm was required. A promising technique separates the problem into a Mixed-Integer Linear Programming (MILP) sub-problem, and a separate non-linear programming (NLP) problem. The NLP solution creates linear constraints to form a linear outer approximation to the feasible region [12], and this outer approximation is used by the MILP sub-problem. MILP solutions are tested for feasibility, where infeasible solutions spawn NLP sub-problems to improve the outer approximation. When the MILP solution is found to be feasible the problem is solved. For this work a simpler means of creating the outer approximation was found, using a supporting hyperplane algorithm [13][14]. The key feature of this algorithm is that the search for the mixed integer solution is separated from the explicit use of circuit simulation/analysis. The time required for the MILP problem is short relative to the time spent finding and computing additional linear constraints. In this algorithm the mixed integer design space actually speeds convergence, by limiting the choices of possible solutions at the end of the optimization, and the solution time rivals the time for the equivalent NLP problem. Also, it only requires accurate constraint computation in the region where the constraints are feasible, relaxing requirements on the circuit estimation function. Disadvantages of this algorithm are that it requires an initial feasible point and a convex feasible region. In practice, the initial feasible point was found quickly through either a simple optimization or user guesses. Repeatability tests were made to verify that the convexity assumption was not limiting the optimality of the solutions for DSYN designs.

Because of the regular, cellular nature of current-switched DACs, consisting of many similar segments, most manual DAC layouts are implemented with tightly packed abutting cells so as to minimize the interconnection area. DSYN uses stretching of library cells for customization to design specifications, and tiling of the created cells to generate the DAC module, resulting in similar compact layouts. Note that this layout method is procedural, so accurate parasitic estimation may be done without module instantiation.

III. DAC Module Optimizations and Experimental Results

A current output DAC architecture, suitable for medium resolution video and instrumentation applications was chosen for DSYN. PMOS current sources are used for compatibility with video DAC application requirements. The architecture uses a two dimensional array of segment current sources, with a local row / column digital decode, which allows individual selection of each cell [15]. The cell schematic is shown in Fig. 2, and further details may be found in [14]. The DAC optimization problem, including constraints to ensure both analog circuit performance, and digital signal integrity, comprises 19 design variables and 44 constraints. This is too large an optimization to complete in one step, so the optimization problem is solved with two optimizations run in sequence -- an ‘analog’ optimization with 13 variables run first, setting variables which affect analog circuit performance. This is followed by a ‘digital’ optimization with 6 variables, which sizes circuits in the DAC digital signal path. Table 1 lists the independent design variables, specifies the optimization each is used in, and their optimized values for the two test designs discussed later.

TABLE 1. DSYN Independent Design Variables.

Variable	**Description**	**Optim.**	**8-bit**	**10-bit**
M	Segment Size, in LSB	Analog	4	16
Rows	Number of Module Rows	Analog	4	4
Bias	Number of bias elements	Analog	1	1
WB1, WB2[a]	Widths for Bias FETs	Analog	7.2, 6.6	1.2, 2.4
W1, L1	Size of M1 (Isrc FET)	Analog	15.6, 3.6	4.2, 4.2
W2, L2	Size of M2 (Cascode FET)	Analog	15.6, 1.2	4.2, 1.2
WS1	Width of MOS Switch in Segment	Analog	14.4	2.4
WBUS	Width of Power Bus	Analog	36	19.2
WPINV, WNLAT	Width sets Switch Drivers	Analog	4.8, 3.0	NA[b]
WNLAT2	Cell Latch device	Digital	4.2	NA
WPDEC, WNDEC	Cell Decode	Digital	6.0, 2.4	NA
WC1, WC2, WR	Row and Col Buffers	Digital	9.6, 20.4, 14.4	NA

a. Dimensions in microns, on a 0.6 micron pitch.

b. Minimum size used. Not optimized in this low speed design.

All first and second order static and transient effects were included in these module optimizations. A 1.2 μm digital CMOS process was modeled, and the discrepancies between fitted BSIM 2 models and measured devices, most significantly a factor of 2 error in output conductance, were compensated for in circuit estimation. A Vds margin constraint was included in optimizations to insure operation far enough into saturation that this would bound modelling errors. Worst case design was used to bracket performance predictions. Statistical device mismatch due to random and gradient effects used the model in [16], and mismatch parameters were extrapolated from that source. Static linearity was predicted for 3-σ yield. INL due to I-R drops in supplies was modeled. Lead inductance effects on the output signal were included, but chip level issues such as substrate noise coupling and supply lead inductance were considered beyond the scope of the module optimization. Initial DAC modules were created to an 8-bit, 100 MS/s video specification[1] and a static 10-bit spec. Optimizations required 6 hours compute time for the 8-bit design, and 1 hour compute time for the 10-bit design on a DEC Alpha 3000/300. Circuit areas were 0.71 and 0.38 mm^2. Fig. 3 is the test chip photograph.

Design development and optimization time was 3 months for the initial 8-bit design -- comparable to custom design. Of this, the first half was spent doing the initial design analysis and layout for the cell library, and the second half running optimizations, performing full circuit parasitic extraction and simulation, and then looking for errors in the estimated results. Errors in the cell library were debugged, and assumptions made in analysis were corrected. Multiple optimizations were required on each iteration to determine the optimum power of 2 for segment size. A total of about 20 optimizations were run before tape out.

For the 8-bit DAC, table 2 shows the performance specification, estimated 3-σ worst case performance from optimization, and measured performance. All 12 parts fabricated were functional, and met static specifications. INL and DNL are worst measured for these 12 parts. Linearity and transients are illustrated in [17]. The 10 bit prototypes were all functional, but none met DNL specs. Worst case results are summarized in table 3.

TABLE 2. Performance of 8 bit video DAC (Ifs = 17.6 mA)

Name	**Specified**	**Estimated**	**Measured**	**Units**
INL	< 1.0	0.53	0.64	lsb
DNL	< 0.5	0.16	0.18	lsb
Total Error	< 4	3.6	3	lsb
Tsettle (rise / fall)	< 13.0	7.4 / 4.43	5.5 / 5.25[a]	ns
Tswit (rise / fall)	< 5.0	1.6 / 1.0	1.8 / 1.1	ns
Tdelay	< 5.0	4.75	NM[b]	ns
Glitch	< 1.0	0.77 / 0.62	0.74 / 0.46	lsb*Ts
Rout	> 10	10.5	12.5	kΩ
Clock Rate	> 100	101	225	MS/s
Vds in Satn.[c]	> 0.2	0.2	NM	V

a. Settling to within 2% of final value.

b. NM -- Not Measurable.

c. A separate constraint is used for each device operating in saturation.

1. This was a typical video DAC specification, so it included transient settling and glitch energy, but not SFDR.

TABLE 3. Performance of static 10-bit DAC (Ifs = 4 mA)

Name	Specified	Estimated	Measured	Units
INL	2.0	1.87	2.1	lsb
DNL	0.5	0.16	1.6	lsb

The zero DNL yield on the 10-bit part, and the higher than expected non-linearity results on the 8-bit part are a cause for concern. When the data was further analyzed, two underestimated effects were seen. First, the process gradients were twice the magnitude predicted by extrapolation from [16], and second, edge effects on the array were significant, causing mismatch between edge columns and the center, and between LSB currents and segment currents. Based on these results the parameter for process gradients has been increased, and a separate mismatch source due to these edge effects has been added to the non-linearity model in the module generator.

The reason for implementing a design as part of a module generator methodology is that it allows reuse of elements of elements of the design, when either specifications or technology change. Table 4 summarizes the reuse of elements of the initial 8-bit, 1.2 µm video DAC to both new specifications and new technologies. For the 10-bit design, the layout cell library required modification, but other elements were reused. In the 0.8 µm design the cell library was scaled automatically, and only a new technology file was needed. Preliminary testing of this design indicates linearity within spec. In the last case a new design specification was applied to the existing module generator libraries, resulting in an implementation time of 1 day.

TABLE 4. Development requirements for successive designs. The first three designs include time for exhaustive verification.

Design	Analysis	Technology File	Cell Library	Implementation Time
8-bit, Video, 1.2µm	New	New	New	3 months
10-bit, 1.2µm			Modified	2 weeks
8-bit, Video, 0.8µm		New		1 week
8-bit, 1.2µm				1 day

IV. Conclusions

This work has demonstrated module generation for high speed current switched Digital/Analog Converters, from design input through synthesis, fabrication, and testing. Tight coupling of layout and synthesis resulted in compact layouts with accurate performance prediction. A new MINLP optimization algorithm was developed for design optimizations. Application of the module generator across specifications and different technologies has demonstrated the portability of the design analysis and libraries. Further information on this research may be found on the WWW at http://kabuki.eecs.berkeley.edu/~neff.

References

1. G.G.E. Gielen, H.C.C. Walscharts, and W.M.C. Sansen, “Analog Circuit Design Optimization Based on Symbolic Simulation and Simulated Annealing,”*IEEE J. Solid-State Circuits*, 25(3), pp. 707-713, June, 1990.
2. J.M. Cohn, D.J. Garrod, R.A. Rutenbar, and L.R. Carley, “Techniques for simultaneous placement and routing of custom analog cells in KOAN/ANAGRAM II,” *Proc. International Conference on Computer-Aided Design*, pp. 394-397, Nov. 1991.
3. E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli, “A Constraint Driven Placement Methodology for Analog Integrated Circuits,” *Proc. IEEE Custom Integrated Circuits Conference*, pp. 1711-1714, May 1992.
4. P.C. Maulik, L. R. Carly, and R. A. Rutenbar, “A MINLP Approach to Analog Circuit Synthesis.” *Proc. Design Automation Conference*, pp. 698-703, June, 1992.
5. M.C. Degrauwe, et al. “Towards an Analog System Design Environment,” *IEEE J. Solid-State Circuits*, 24(3), pp. 659-671, June 1989.
6. H. Yaghutiel, “Automatic Synthesis and Layout of Switched-Capacitor Filters.” University of California at Berkeley, UCB/ERL M88/56, August 1988, Ph. D. Thesis.
7. H.Y. Koh, C.H. Sequin, and P.R. Gray, “OPASYN: A Compiler for CMOS Operational Amplifiers,” *IEEE Trans. on Computer-Aided Design*, 9(2): 113-125, Feb. 1990.
8. G. Jusuf, P.R. Gray, and A.Sangiovanni-Vincentelli, “CADICS - Cyclic Analog-to-Digital Converter Synthesis.” *Proc. International Conference on Computer-Aided Design*, pp. 286-289, Nov. 1990.
9. H.C. Chang, E. Liu, R. Neff, et al, “Top-Down, Constraint-Driven design methodology based Generation of Interpolative Current Source D/A Converters,” *Proc. IEEE Custom Integrated Circuits Conference*, pp. 15.5.1-4, May, 1994.
10. H.C. Chang, E. Felt, and A. Sangiovanni-Vincentelli, “Top-down, Constraint-Driven Design Methodology Based Generation of a Second Order ΣΔ-A/D Converter,” *Proc. IEEE Custom Integrated Circuits Conference*, pp. 25.5.1-4, May, 1995.
11. W. Nye et al, “DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits,” *IEEE Trans. on Computer-Aided Design*, 7(4), pp. 501-19, April 1988.
12. M.A. Duran and I.E. Grossmann, “An Outer-Approx. Algorithm for a Class of MINLPs,” *Mathematical Programming*, 36, pp. 307-339, 1986.
13. D.G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Co., 2nd Ed., 1984.
14. R. R. Neff, “Automatic Synthesis of CMOS Digital/Analog Converters,” University of California at Berkeley, UCB/ERL M95/28, April 1995, Ph. D. Thesis.
15. L. Letham, B.K. Ahuja, et al., “A High-Performance CMOS 70-MHz Palette/DAC,” *IEEE J. Solid-State Circuits*, SC-22(6), pp. 1041-47, December 1987.
16. M.J.M. Pelgrom, A. Duinmaijer, and A. Welbers, “Matching Properties of MOS Transistors,” *IEEE J. Solid-State Circuits*, SC-24(5), pp. 1433-40, Oct. 1989.
17. R. R. Neff, P. R. Gray, and A. Sangiovanni-Vincentelli, “A Module Generator for high-speed CMOS current-output D/A Converters,” *Proc. IEEE Custom Integrated Circuits Conference*, pp. 23.7.1-4, May, 1994.

Figure 1. Module Generation Methodology.

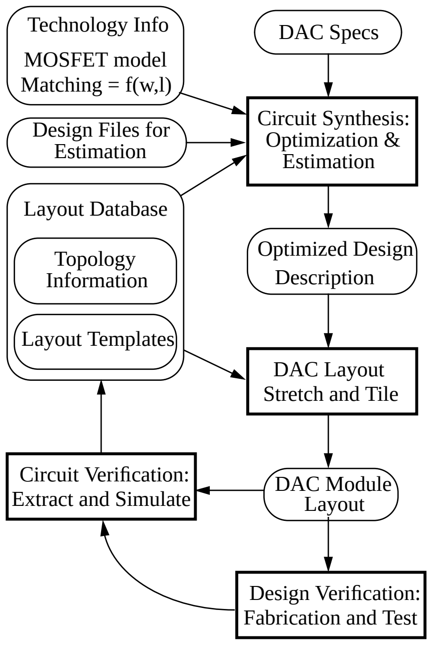
Technology Info
MOSFET model
Matching = f(w,l)
DAC Specs
Design Files for Estimation
Circuit Synthesis: Optimization & Estimation
Layout Database
Topology Information
Layout Templates
Optimized Design Description
DAC Layout Stretch and Tile
Circuit Verification: Extract and Simulate
DAC Module Layout
Design Verification: Fabrication and Test

Figure 2. Switched Current Source DAC segment cell schematic.

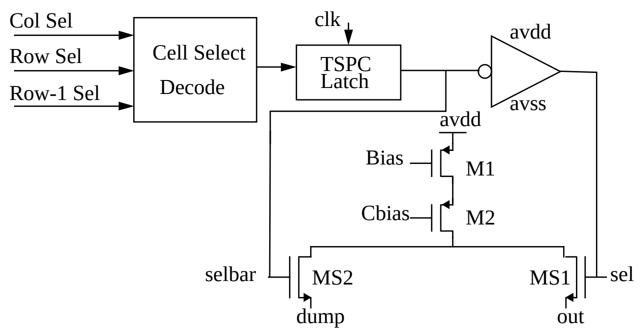

Figure 3. Die photo with 8-bit part in lower left, 10-bit part in lower right