PARALLEL SIZING OF ROBUST ANALOG ICs

Árpád Bűrmen, Janez Puhan, Tadej Tuma

University of Ljubljana, Faculty of Electrical Engineering, Ljubljana, Slovenia

Key words: circuit sizing, analog IC, parametric optimization, parallel optimization, CAD.

Abstract: Automated robust IC design is a time consuming process. The resulting circuit must fulfill all the performance constraints in several different operating conditions and under the influence of different manufacturing process variations (corners). To achieve this, a large number of circuit analyses must be performed for the circuit, considered by the sizing algorithm. By distributing these simulations among multiple computers speedup can be obtained. A major problem is the synchronous nature of the optimization algorithm that in combination with variable duration of a circuit analysis causes the computers in the parallel system to be idle. This results in a reduction of speedup. The paper presents the corner parallel approach to the problem of parallel robust analog IC sizing. The method is tested with a sample opamp sizing problem. The speedups are measured, and an approach is proposed for reducing the idle time of the computers.

Vzporedno dimenzioniranje robustnih analognih integriranih vezij

Kjučne besede: dimenzioniranje vezij, analogna integrirana vezja, parameterska optimizacija, vzporedna optimizacija, računalniško podprto načrtovanje.

Izvleček: Avtomatsko robustno načrtovanje analognih integriranih vezij je časovno zahteven postopek. Cilj je najti vezje, ki zadosti vsem načrtovalskim zahtevam za dano območje pogojev delovanja in variacij parametrov postopka izdelave (vogalnih točk). V ta namen je potrebno narediti veliko število analiz za vsako vezje, ki ga računalnik med postopkom iskanja preizkusi. S porazdelitvijo teh analiz med več računalnikov je mogoče postopek pospešiti. Vendar pa sinhrona narava optimizacijskega postopka v kombinaciji s spremenjlivo dolžino analize povzroči, da so računalniki v vzporednem sistemu del časa brez dela. Rezultat je zmanjšana pospešitev postopka. Članek opisuje pristop k paralelizaciji robustnega načrtovanja vezij preko porazdelitve vogalnih točk med več računalnikov. Podani so rezultati v odvisnosti od števila vzporedno računajočih računalnikov. Predlagan je tudi postopek za skrajšanje časa, med katerim so računalniki brez dela.

1 Introduction

As the competition in the IC design industry is becoming sharper, methods for automating parts of the IC design process are gaining on importance /1/. Up to recently circuit simulation was the tool that gave "the edge" to an IC designer. It may have taken a relatively long time for a circuit simulator to evaluate circuit's performance under the influence of various operating conditions and manufacturing process variations, but that was compensated with designer's knowledge and experience in choosing circuit parameter values. Experienced designers are capable of sizing a circuit in a reasonable amount of time when compared to the time spent for simulation. Various approaches for speeding up the simulation were developed and tested during that time /2/.

The available computing power steadily increased and the simulation time began to decrease until it reached the point where simulation is no more a bottleneck in the design process. This trend is becoming more and more obvious as the techniques (e.g./3/, /4/) and tools (e.g. /5/, /6/) for automating the circuit sizing process begin to appear. These tools put an optimization algorithm (/7/, /8/) in the role of a human designer. The basic idea is to emulate a part of the designer's knowledge and experience by transforming the search for a "better circuit" into the search for a lower cost function (CF) value. An optimization algorithm searches for such circuit parameter (e.g. MOS widths and lengths) values where the CF reaches its lowest value. When using such tools the designer's job is to define

- circuit structure (e.g. the unsized schematic of an opamp),
- element matching /9/ (some circuit element parameters are not independent, e.g. width and length of the two MOS transistors in a differential pair),
- which analyses to perform and what circuit characteristics to extract from them (we consider circuit area as yet another circuit characteristic, although it is not the result of some analysis),
- set of corners for which the circuit is to be analyzed (a corner is a particular combination of operating conditions and manufacturing process variations), and
- performance constraints on circuit characteristics and their relative importance with respect to each other.

The optimizer then handles the rest of the work. Of course if the performance constraints are set to some values that cannot be achieved with the given circuit structure, the resulting circuit doesn't fulfill all performance constraints.

A typical optimization run requires the evaluation of several hundred up to several thousand different circuits with the same structure and different parameter values. As every circuit has to be evaluated in all of the specified cor-

ners, the optimization may take a long time (at the time being, using a state-of-the-art computer it takes typically from a few hours up to several days). Therefore every reduction of this time represents a significant benefit to the designer. There exist different approaches for accelerating the process of optimization by means of parallelization.

This paper presents the so-called corner-parallel approach. The remainder of the paper is subdivided as follows. First the method of constructing the cost function is presented. The various levels at which parallelism can be introduced are briefly discussed. The use of PVM /10/ for implementing the corner-parallel approach is described and runtimes for a sample circuit sizing problem are given. Finally the synchronization penalty is discussed and an approach for its alleviation is proposed.

2 Defining the cost function

2.1 Circuit design and corner points

In order to obtain a robust circuit it must exhibit adequate performance in all corners. A corner is a combination of some process variation (like the worst power and worst speed MOS corner) and M operating conditions (like temperature, supply voltage, etc.). Suppose that we have n_0 possible process variations. Let n_i denote the number of different values for the i-th operating condition that are of interest to the designer. Then the total number of corner points is

$$K = \prod_{i=0}^{M} n_i \tag{1}$$

Of course (1) quickly grows beyond any reasonable value. Therefore the designers usually examine the circuit for a subset of corners (C_S). M_S denotes the number of corners in C_S.

The performance of the circuit can be described by a vector $\underline{y} = [y_1,...,y_N] \in R^N$ of N real valued performance measures (like gain, bandwidth, rise time, etc.). When sizing a circuit the optimizer has to find the values of n circuit parameters represented by a vector $\underline{x} \in R^n$. The circuit itself can be viewed as a transformation (2) that for some combination of n circuit parameters denoted by vector $\underline{x}$ and some corner point denoted by q produces a vector of circuit characteristics $\underline{y}$.

$$D:(\underline{x},q) \mapsto \underline{y} \qquad \underline{x} \in R^n, q \in C, \underline{y} \in R^N \tag{2}$$

$$y(\underline{x},q) = [y_1(\underline{x},q), y_2(\underline{x},q),..., y_N(\underline{x},q)]$$
$$= [D_1(\underline{x},q), D_2(\underline{x},q),..., D_N(\underline{x},q)]$$

The designer's goal is to either minimize or maximize the performance measure. Typically one tries to minimize measures like rise time and fall time and maximize measures like bandwidth and gain. Let h_i be 1 if y_i is to be maximized and 0 if it is to be minimized. Now let b_i denote the worst still acceptable y_i value. A circuit has acceptable performance with respect to performance measure y_i under minimization if $y_i \le b_i$. On the other hand if y_i is being maximized, acceptable circuit performance with respect to y_i requires $y_i \ge b_i$.

Finally the circuit is considered to have acceptable performance if its performance is acceptable with respect to all performance measures for all corners $q \in C_S$.

Let y^{worst} denote the vector of worst performance measure values across all corners.

$$y_i^{worst}(\underline{x}) = \begin{cases} \min\limits_{q \in C_S} y_i(\underline{x},q) & h_i = 1 \\ \max\limits_{q \in C_S} y_i(\underline{x},q) & h_i = 0 \end{cases} \tag{3}$$

2.2 Constructing the cost function

Suppose $g(x)$ is a continuous monotonically increasing function defined for $x \ge 0$. Then

$$f(x) = \begin{cases} 0 & x < 0 \\ g(x) - g(0) & x \ge 0 \end{cases} \tag{4}$$

We know that the optimizer searches for the lowest CF value. Therefore better performing circuits should be assigned a lower CF value. For that purpose a partial CF can be defined using (3) and (4):

$$F_P(\underline{y}) = \sum_{i=1}^{N} \left((1-h_i) f\left(\frac{y_i^{worst} - b_i}{A_i} \right) + h_i f\left(\frac{b_i - y_i^{worst}}{A_i} \right) \right) \tag{5}$$

Finally the CF that is used by the optimizer is the sum of partial CF values across all corners.

$$F(\underline{x}) = \sum_{q \in C_S} F_P(y(\underline{x},q)) \tag{6}$$

By minimizing this function the optimizer actually searches for the circuit (vector of parameters $\underline{x}$) with minimal constraint violation. If a circuit with CF value 0 is found, it means that all the constraints are satisfied and the optimization can be stopped.

The optimizer must consider the fact that the simulator used for evaluating the circuit is not always capable of solving the circuit equations. As a matter of fact there exists a plethora of pathologic circuits for which the simulator fails to find a solution. In such a case we assign either $y_i = L$ (if $h_i = 0$) or $y_i = -L$ (if $h_i = 1$) to all y_i that can't be evaluated due to simulator failure. L is a large positive value. Such strategy produces a very high CF value for pathological circuits and thus forces the optimizer to avoid those parts of the parameter space, where such circuits are found.

3 Parallel circuit sizing

The innermost level (level 1 in fig. 1) at which parallelism can be introduced is the simulator level. If one wants to exploit parallelism at this level a special simulator is required. In the past a lot of effort was invested in developing such

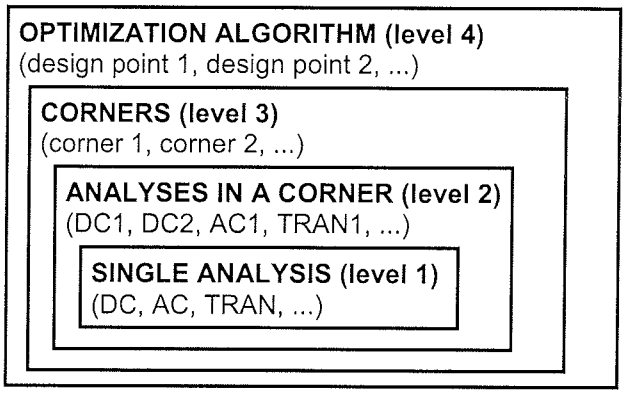

Fig 1: possible levels of parallelism.

simulators, but none of them grew to be as general and widespread as today's circuit simulators (e.g. /12/).

The outermost level is the optimization algorithm level. The amount of speedup that can be gained from parallelism at this level depends on the type of optimization algorithm. Genetic algorithms /13/ (GA) and simulated annealing /14/ (SA) are well suited for such parallel execution. Unfortunately they require many CF evaluations before they reach a solution. So the large speedup that can be gained from parallel execution is lost due to the fact that these algorithms are extremely time consuming per se. On the other hand algorithms like the simplex algorithm need fewer CF evaluations (between one and two orders of magnitude less than SA and GA), but the amount of speedup that can be obtained by introducing parallelism seems to be limited (/15/, /16/).

What remains are the two intermediate levels. As a matter of fact there really is no distinct border between them. The two levels are depicted separately in fig. 1 to emphasize the way the CF is evaluated, i.e. for every corner several different circuit analyses are performed and every one of them is an independent entity that can be simulated in an independent simulator run. One or more performance measures are then extracted from results of every analysis and are used to build a part of the CF according to (5). Introducing parallelism at this level means that the CF evaluation is distributed among several computers, where every computer evaluates the part of the cost function that belongs to a subset of corners and analyses. Effectively this means that one computer runs one or more analyses and extracts performance measures from the obtained analysis results. In the remainder of this paper we focus on corner-level parallelism (level 3).

In the parallel computing community two major tools are available for implementing parallel algorithms: PVM /10/ and MPI /11/. MPI is the official standard. Recently the PVM library gained a wide audience due to its flexibility and the portability of its open source implementation. Today it is recognized as the de-facto standard for parallel processing. Currently the most popular hardware platform for parallel processing is the loosely coupled (ethernet) cluster of ordinary PCs, usually running LINUX.

PVM consists of two parts. The PVM deamon runs as a server on every computer of the cluster. It manages processes and the communication between them. The second part is the PVM library that comprises a large set of C

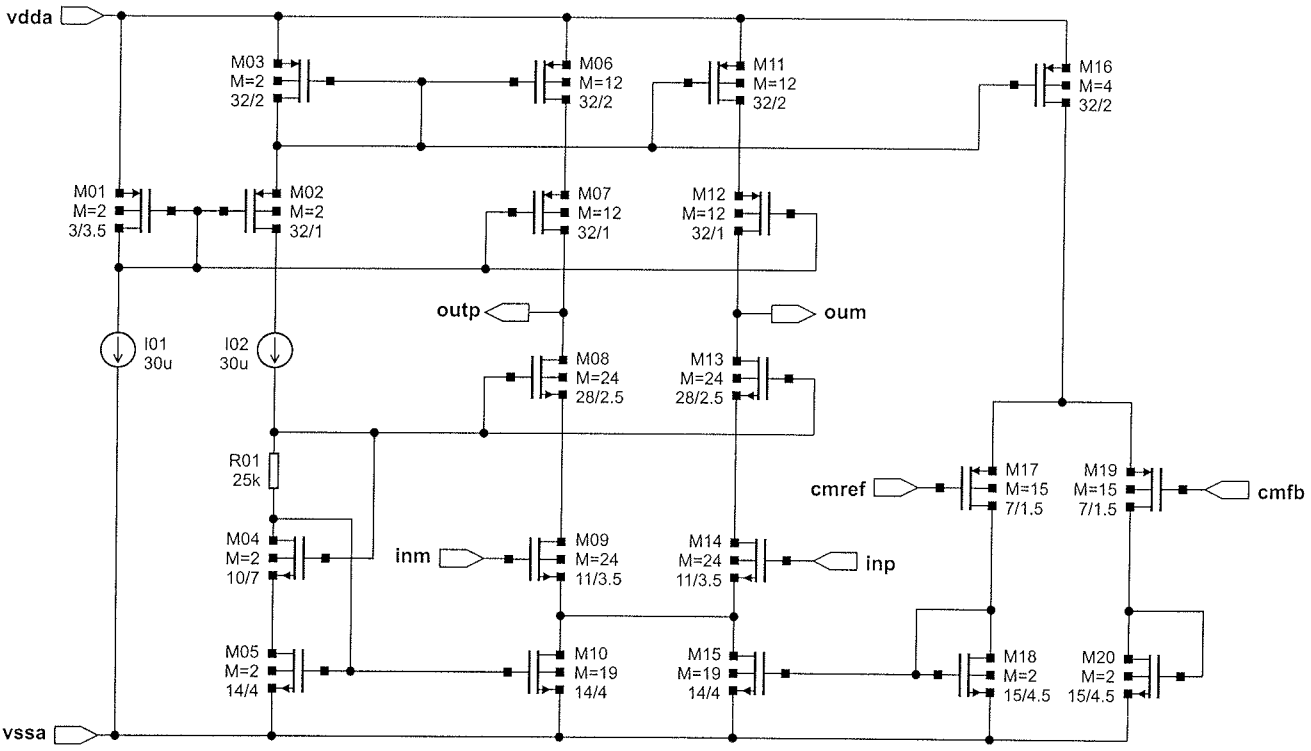

Fig. 2: Sample opamp with M and W/L values. The scale is 0.3μm (all W/L values must be multiplied by 0.3μm to obtain physical dimensions).

functions for process management, and interprocess communication. Typically a PVM application starts as an ordinary process on the master computer. Upon entering PVM the process becomes the master process. Next it spawns worker processes on remote computers in the cluster upon which the parallel algorithm can start executing.

The work is divided among the workers in such a manner that every worker handles one corner (one $q \in C_S$), i.e. one term from the sum in (6). One doesn't have to have M_S computers in the cluster for that. If the number of computers (P) is smaller than M_S then at first only P corners are divided among the workers. The worker that first finishes evaluating its corner obtains a new corner from the master computer. This continues until the master computer collects the partial CF values for all corners. After that the optimization algorithm on the master computer selects a new point ($\underline{x}$) in the parameter space and the parallel CF evaluation is repeated for that point.

The above-proposed parallel approach is synchronous. The evaluation of the CF from collected partial CF values at the master computer represents a synchronization point. The master cannot continue from this point until all partial CF values are collected from the workers. The duration of a partial CF evaluation is not constant since the simulator (SPICE) solves the circuit by means of various iterative techniques. As a consequence some workers finish sooner than the others. If the master doesn't have a partial CF evaluation for them that will keep them busy, those workers remain idle until the master starts evaluating the next trial circuit.

4 Example

To illustrate the proposed approach to parallel circuit sizing, a sample opamp circuit (fig. 2) was used. 9 corners were taken into account representing all possible combinations of 3 Vdd values (3.0V, 3.3V, and 3.6V), 3 operating temperatures (-10°C, 27°C, and 80°C), and 3 MOS corners (TYP, BCS, WCS). W/L/M values in fig. 2 represent a circuit sized by an experienced designer.

The test circuit in fig.3 was used to evaluate the performance of the circuit in fig. 2 during the process of sizing. Several analyses were performed and performance measures were extracted from the results for every corner. Table 4 lists the performance measures, goals, and corresponding penalty coefficients from equation (5).

Although there are 20 MOS transistors in the circuit with 3 adjustable parameters per transistor, the number of optimization parameters is only 18. This number is reduced by grouping transistors like current mirrors and differential pairs in groups /9/. Table 1 lists the optimization parameters, the MOS transistors groups to which these parameters belong and their explicit constraints.

The parallel approach was tried out with different numbers of workers. The workers and the master were AMD ATHLON 2100XP computers running LINUX with PVM. In the first set of tests the optimization was stopped after 160 CF evaluations and the runtime was measured. Table 2 lists the runtimes, the corresponding speedups, and the overall CPU usage. Speedup was obtained by dividing the time spent in a single worker run with the time spent in a N-worker run. The CPU usage was obtained by dividing the

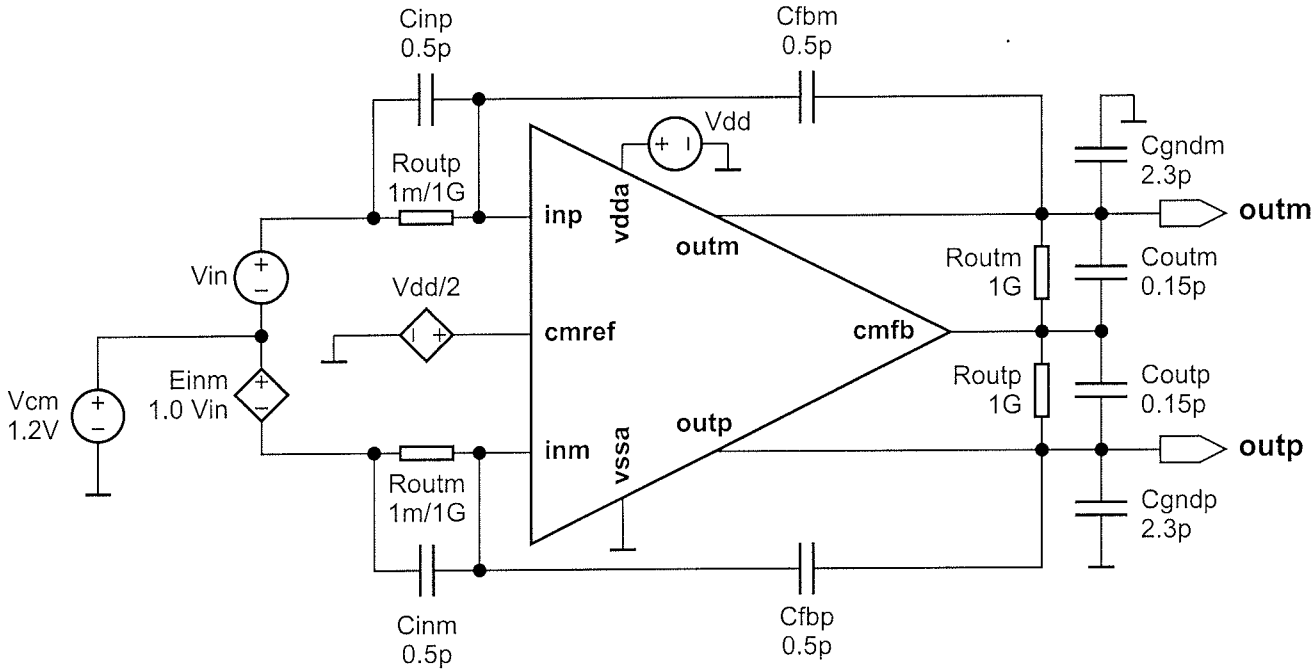

Fig 3: The test circuit. If two values are specified for some element, the first one is valid for all analyses but the transient. For the transient analysis the second value is used.

Table 1: Optimization parameters and matching. The scale is 0.3μm. During optimization widths and length are rounded to the nearest multiple of 0.5. M factor is rounded to the nearest integer.

W=2..50	L=1..10	W=10..50	L=1..10	L =1..10	W=10..50	L=2..10	W=10..50	L=2..10
M1	M1	M2,M3,M6, M7,M11, M12,M16	M2,M7, M12	M6,M6, M11,M16	M4	M4	M5,M10, M15	M5,M10, M15

W=10..50	L=1..10	W=10..50	L=1..10	M=6..24	W=2..10	L=1..10	W=5..50	L=2..10
M8,M13	M8,M13	M9,M14	M9.M14	M10,M15	M17,M19	M17,M19	M18,M20	M18,M20

speedup with the number of workers. From the obtained results it is clear that using 4-5 workers is a reasonable choice. This way speedup values around 2.7 can be obtained.

A large part of the performance degradation (overall CPU usage below 1.0) can be attributed to the variations of the partial CF evaluation time. If the master has no more partial CF evaluations for the current point in the design space, the workers that finish sooner remain idle until the master collects all partial CF values and moves on to the next point in the design space. It is often the case that the designer wants to try several different tradeoffs. Such tradeoffs can be examined by executing multiple optimization runs. If these runs are executed concurrently on the given set of workers one can expect a decrease of worker idle time. With more than one optimization run taking advantage of a worker, the probability that all of the optimization runs will leave the worker idle at some point in time decreases with the number of concurrent runs. Table 2 illustrates this for the case of 2 and 3 concurrent optimization runs.

For multiple concurrent optimization runs the equivalent time is obtained by dividing the total run time with the number of concurrent runs. This figure can be compared directly to the duration of a single run on a single worker (t_1). The quotient of t_1 and equivalent time represents the effective speedup. Finally the overall CPU usage is obtained by dividing the effective speedup with the number of workers. Effective speedup for 5 workers improved to 3.6 (2 concurrent optimization runs) and 4.2 (3 concurrent optimization runs).

With these indications the second set of tests was conducted where several full optimization runs were executed. This time the optimization was stopped when the final solution was found (after 1965 CF evaluations). One run with a single worker and several runs with 5 workers were conducted. Table 3 lists the runtime, equivalent runtime per optimization run, effective speedup, and overall CPU usage. The speedup and overall CPU usage are similar to the ones obtained from truncated runs (stopped after 160 CF evaluations).

Table 3: Performance for different numbers of concurrent parallel optimization runs on 1 and 5 workers.

	1 worker	5 workers		
	1 run	1 run	2 runs	3 runs
Total time [s]	3870.1	1391.3	2170.5	2871.6
Equivalent time [s]	3870.1	1391.3	1085.3	957.2
Effective speedup	1.00	2.78	3.57	4.04
Overall CPU usage	1.00	0.56	0.71	0.81

The final results are presented in table 4. It is worth noting that the computer started from an initial point that wasn't even a feasible opamp. Using 5 workers the computer sized the circuit in 1391s (20 minutes).The resulting circuit had performance comparable to or better than the performance of the human-designed circuit.

Table 2: Time spent for 160 CF evaluations with 1, 2, and 3 optimization runs in parallel. The equivalent time, effective speedup, and overall CPU usage are also listed. Optimization was stopped after 160 CF evaluations.

		Number of workers (N)					
		1	2	3	4	5	6
1 run	Time [s]	617.0	362.1	275.6	243.2	228.4	223.8
	Effective speedup	1.00	1.70	2.24	2.54	2.70	2.76
	Overall CPU usage	1.00	0.85	0.75	0.64	0.54	0.46
2 runs	Time [s]	-	591.4	427.6	358.5	345.2	350.0
	Equiv. time [s]	-	295.7	213.8	179.2	172.6	175.0
	Effective speedup	-	2.09	2.90	3.44	3.57	3.53
	Overall CPU usage	-	1.04	0.97	0.87	0.71	0.59
3 runs	Time [s]	-	-	587.8	484.8	438.4	464.0
	Equiv. time [s]	-	-	195.9	161.6	146.1	154.7
	Effective speedup	-	-	3.14	3.82	4.22	3.99
	Overall CPU usage	-	-	1.06	0.96	0.84	0.67

Table 4: Comparison of performance constraints (b), penalty coefficients (A), human-designed circuit, and computer-designed circuit worst-corner performance. ↑ stands for maximize and ↓ for minimize.

Analysis	Measurement		b	A	Human	Computer
-	Area	↓	1000μm^2	100μm^2	985μm^2	983μm^2
OP	Supply current at V_{in}=0V	↓	0.5mA	0.1mA	0.49mA	0.48mA
DC	Swing (50% gain drop)	↑	2.0V	0.1V	2.09V	2.02V
	Open-loop gain (-1V..1V)	↑	60dB	1dB	57.2dB	56.1dB
AC	Unity gain bandwidth	↑	100MHz	10MHz	83.5MHz	131.2MHz
	Phase margin	↑	70°	1°	73.9°	75.9°
	Gain margin	↑	12dB	1dB	13.0dB	13.8dB
NOISE	RMS output Integ. noise (1kHz..1GHz)	↓	3mV	0.1mV	3.75mV	2.96mV
	Output spot noise (at 10Hz)	↓	15μV/√Hz	1μV/√Hz	16.7μV/√Hz	11.2μV/√Hz
TRAN	Overshoot	↓	0.1%	0.01%	0.11%	0.086%
	Slewrate	↑	20V/μs	1V/μs	13.3 V/μs	21.4 V/μs
	Settling time	↓	10ns	1ns	11.3ns	7.0ns
	Rise time	↓	10ns	1ns	13.5ns	8.4ns
	Fall time	↓	10ns	1ns	13.6ns	8.5ns

5 Conclusion

The corner-parallel approach to robust circuit sizing has been presented. The mathematical formulation of the cost function used in the process of optimization represents the basis for dividing the work among the workers. Every worker evaluates a piece of the cost function that belongs to a particular corner of the circuit. The optimization algorithm needs the cost function value for choosing the next trial point in the design space. Therefore all partial cost function values must be collected first. Duration of the partial cost function evaluation varies significantly since circuit simulation is an iterative procedure. Some workers finish sooner than the others. If the master has no more partial cost function terms for handing over to the workers, these workers remain idle until all partial cost function values are collected by the master. At that point the optimization algorithm running on the master chooses a new point in the design space and provides the workers with new partial CF terms for evaluation.

The consequence of worker idle time is the reduced speedup. For a circuit with M_S corners optimized using M_S workers the speedup should be near M_S. Unfortunately the large variations in partial cost function evaluation time greatly reduce this number. The designer often wants to explore various design tradeoffs resulting from different parallel optimization runs. These optimization runs can be executed concurrently. The experiments show that running 2 or 3 concurrent parallel optimization runs greatly improves CPU usage. This is due to the fact that with the increasing number of optimization processes running on a single worker the probability that all of them will be idle at the same time decreases.

Taking into account Moore's law (CPU power doubles every 18 months) one can expect that in the following 5 years sizing circuits (like the one in fig. 2) will become a matter of minutes, making automated circuit sizing a standard tool for every IC designer. Considering the fact that sizing such a circuit manually can be a matter of days even for experienced designers, manual circuit sizing will become economically unfeasible for many standard circuits (like opamps, linear regulators, etc.). This of course doesn't mean that virtually anyone will be able to do IC design. Setting up an optimization run requires specific designer knowledge in the process of choosing the circuit structure, performance constraints, and element matching. Sensible explicit constraint values on optimization parameters reduce mismatch, but on the other hand increase the minimum achievable circuit area. Choosing these tradeoffs still requires extensive design experience.

6 References

/1/ Gielen, G. G. E., Rutenbar, R. A., *Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits*. Proceedings of the IEEE, vol. 88, no. 12, pp. 1825-1854, 2000.

/2/ Saleh, R.A., Gallivan, K.A., Chang, M.-C., Hajj, I.N., Smart, D., Trick, T.N., *Parallel Circuit Simulation on Supercomputers*, Proceedings of the IEEE, vol. 77, no. 12, pp 1915-1931, 1989.

/3/ Phelps, R., Krasnicki, M., Rutenbar, R. A., Carley, L. R., Hellums, J. R., *Anaconda: Simulation-Based Synthesis of Analog Circuits via Stochastic Pattern Search*, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 19, no. 6, pp. 703-717, 2000.

/4/ Bűrmen, Á., Strle, D., Bratkovič, F., Puhan, J., Fajfar, I., Tuma, T., *Penalty Function Approach to Robust Analog IC design*, Informacije MIDEM-Journal of Microelectronics, Electronic Components, and Materials, vol. 32, no. 3, pp. 149-156, 2002.

/5/ http://www.neolinear.com, January 2004.

/6/ http://www.synopsys.com/products/mixedsignal/hspice/creative_genius_ds.html, January 2004.

/7/ J. Puhan, T. Tuma, *Optimization of analog circuits with SPICE 3f4*. Proceedings of the ECCTD'97, vol. 1, pp. 177 - 180, 1997.

/8/ J. Puhan, T.Tuma, I. Fajfar, *Optimisation Methods in SPICE, a Comparison*. Proceedings of the ECCTD'99, vol. 1, pp. 1279-1282, 1999.

/9/ Gräb, H., Zizala, S., Eckmüller, J., Antreich, K., *The Sizing Rules Method for Analog Integrated Circuit Design*. IEEE/ACM International Conference on Computer-Aided Design, pp. 343-349, 2001.

/10/ Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Mancheck, R., Sunderam, V., *PVM: Parallel Virtual Machine*, MIT Press, 1994.

/11/ Gropp, W., Lusk, E., Skjellum, A., *Using MPI - 2nd Edition*, MIT Press, 1999.

/12/ T. Quarles, A. R. Newton, D. O. Pederson, A. Sangiovanni-Vincentelli, *SPICE3 Version 3f4 User's Manual*, Berkeley, University of California, 1989.

/13/ Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Adisson-Wesley, Reading, 1989.

/14/ Laarhoven, P. J. M. van, *Theoretical and Computational Aspects of Simulated Annealing*, Centrum voor wiskunde en informatica, Amsterdam, 1988.

/15/ J. E. Dennis, Jr., V. Torczon, *Parallel Implementations of the Nelder-Mead Simplex Algorithm for Unconstrained Optimization*. Proceedings of the SPIE, Vol. 880, pp. 187-191, 1988.

/16/ L. Coetzee, E. C. Botha, *The Parallel Downhill Simplex Algorithm for Unconstrained Optimisation*. Concurrency: Practice and Experience, vol. 10, no. 2, pp. 121-137, 1998.

as. dr. Árpád Bűrmen
Univerza v Ljubljani
Fakulteta za elektrotehniko, Tržaška 25, SI-1000 Ljubljana
E-mail: arpadb@fides.fe.uni-lj.si
Telefon: (01) 4768 322

doc. dr. Janez Puhan
Univerza v Ljubljani
Fakulteta za elektrotehniko, Tržaška 25, SI-1000 Ljubljana
E-mail: janez.puhan@fe.uni-lj.si
Telefon: (01) 4768 322

izr. prof. dr. Tadej Tuma
Univerza v Ljubljani
Fakulteta za elektrotehniko, Tržaška 25, SI-1000 Ljubljana
E-mail: tadej.tuma@fe.uni-lj.si
Telefon: (01) 4768 329

Prispelo (Arrived): 16.03.2004 *Sprejeto (Accepted): 20.06.2004*