Design Automation Tasks Scheduling for Enhanced Parallel Execution of a State-of-the-Art Layout-Aware Sizing Approach

David Neves, Ricardo Martins, Nuno Lourenço and Nuno Horta
Instituto de Telecomunicações/Instituto Superior Técnico – ULisbon
Lisbon, Portugal

Abstract — **This paper presents an innovative methodology to efficiently schedule design automation tasks during the execution of an analog IC layout-aware sizing process. The referred synthesis process includes several sub-tasks such as DC simulation, floorplanning, placement, global routing, parasitic extraction, and circuit simulations in multiple worst case corners. The schedule of the design tasks is here optimized taking into account standard multi-core architectures, tasks dependencies, accurate time estimations for each task and a limited number of licenses for using commercial tools, e.g., number of simulator licenses. The proposed methodology, first, considers a directed acyclic graph for representing the design flow and task dependencies, then, an evolutionary kernel is used to implement a single-objective multi-constraint optimization. The efficiency and impact of the proposed approach is validated by using a state-of-the-art Analog IC design automation environment.**

Keywords— Analog Integrated Circuits, Electronic Design Automation, Schedulling, Parallel Processing.

I. Introduction

Analog IC design automation is being dominated by optimization-based approaches using accurate electrical simulators as evaluation engines [1, 2, 3], however, this leads to a great overhead in terms of time consumption. Moreover, when layout effects and multiple simulations are considered to account for process variability and worst case corners [4] in the evaluation procedure, the execution time increases tenfold. A common approach to decrease the execution time of the optimizers kernels is to use parallel simulations [5], however in real world applications and environments, the software licenses for the tools are much more scarce than the available processing elements, hence the efficient use of parallelism is of the utmost importance to promote an efficient use of the available system resources.

This paper presents an innovative methodology to efficiently schedule the evaluation tasks needed for the automatic synthesis of analog ICs targeting the efficient usage of the standard and widespread multi-core CPU architectures, available in even the most modest workstations. The paper is organized as follows. In section II, the related work is briefly reviewed, section III describes in detail the employed methodology. Section IV presents the achieved results using benchmark circuits and compares the results with competing alternatives. Finally, in section V the conclusions are drawn.

II. Related Work

An overview on analog IC sizing tools and approaches shows that only a few explore a parallel implementation but practically none explore an optimal resource allocation. The analog synthesis tool PAGE [6] uses an equation-based optimizer and employs a coarse-grained distributed multi-objective genetic algorithm for parallel fitness evaluation of individuals using a ring topology with migration of individuals between isolated populations. PyCO [7] uses logistic functions to evaluate the fitness of individuals and a circuit simulator to optimize the global solutions found by the genetic algorithm. The framework was developed using the Python language and the optimization kernel was coded using PyEvolve library which is an API for genetic algorithms. Though not mentioned in the paper what was the parallel model implemented, the library allows for parallel fitness evaluation. Santos-Tavares [5] implemented a framework for time-domain optimization of multi-stage amplifiers using a parallel genetic algorithm that uses an open-source circuit simulator (NGSPICE) for individual fitness evaluations. ANACONDA [8] uses a pool of workstations with dynamic scheduling of simulation tasks (bin-packing heuristic) combined with a Parallel Recombinative Simulated Annealing (PRSA) approach. The evaluation master serves as an intermediary by mapping SPICE simulations to the slaves, regulating results from slaves and returning the obtained results to the optimization kernel.

The reason for the lack of analog IC sizing tools that use parallelism is the inherent extra complexity associated with parallel algorithms. However, due to the growing availability of multicore processors and PC-based clusters, this approach is becoming more common since the extra effort spent in a parallel implementation payoff, due to the reduced execution time. From the best of our knowledge, previous approaches present improvements in execution time when compared to a sequential execution by means of parallel fitness evaluation, however, they do not promote an efficient resource usage, when resources in real design centers, such as software licenses for commercial circuit evaluators, are limited, moreover, they also do not account for the parallel flow that emerges from the Layout-Aware evaluation approaches. This work contributes to fill in this gap, by using a scheduling mechanism to distribute the evaluation tasks efficiently using minimal resources (i.e. software licenses).

III. Proposed Method

The proposed method for design-tasks scheduling will be here described, considering a simple example, and following the layout-aware sizing approach implemented in the AIDAsoft's design environment [9, 10, 11].

A. Problem Formulation

The problem consists of finding the optimal scheduling of the tasks of the layout-aware sizing procedure illustrated in Figure 1, where the tasks are identified by the boxes in blue and red. The boxes in blue are circuit simulations and the boxes in red are layout generation tasks. The approach implemented in AIDA is optimization-based with electrical simulators as evaluation engines, supported on evolutionary techniques. Therefore, a population with several potential and independent solutions must be evaluated during each optimization cycle. This leads to huge potential for parallelization, however, the number of available CPU cores and the number of software licenses for electrical simulation is limited, representing critical constraints for optimizing the task scheduling.

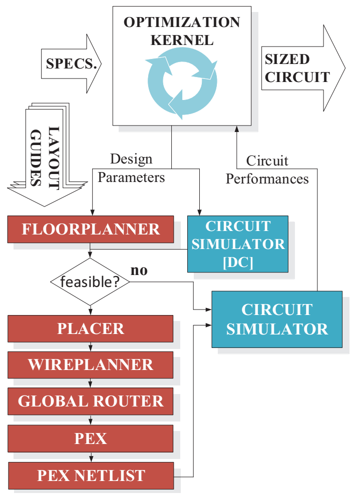

Figure 1 – Layout-aware sizing flow implemented in AIDA.

B. Methodology Overview

The proposed approach for achieving the optimal parallel execution starts by considering the following assumptions:

- Task preemption is not allowed, since that would interrupt the normal execution of the evaluation procedure;
- Expected execution times for each task are obtained through benchmarking in the first iteration of the optimization.

After benchmarking, the next step is to create the representation of the parallel flow using a directed acyclic graph (DAG) for the design flow. Then, the task schedule is optimized. Finally, once the optimal schedule is found, the tasks are created and then are distributed to a private queue in each thread for execution. The method proposed in this paper is not restricted to AIDA and it is applicable to any environment with multiple heterogeneous evaluation tasks and licensing restrictions.

C. DAG Representation

A parallel program can be represented as a DAG, $G = \{V, E\}$ where each node belonging to the set $V = \{T_1, T_2, \dots, T_n\}$ in the DAG represents a task. The main goal is to find the optimal mapping of each task in the task set V to the set of available machines $M = \{m_0, m_1, \dots, m_p\}$ to minimize the total completion time of the evaluation procedure. The set of edges $E = \{e_1, e_2, \dots, e_m\}$ represent the precedence constraints of tasks meaning that task T_j cannot be executed before all of its direct predecessors T_i finished execution. Usually each edge as a weight w_j associated and it represents the data communication costs between each adjacent node, but since the target platform is Shared Memory Multi-Processor (SMP) system, it is considered unit weight in this work and is not represented in the DAG. In Figure 2 is illustrated the graph of AIDA's layout aware evaluation flow, using a population of size 4, and each individual has 2 floorplans and 2 corners simulations.

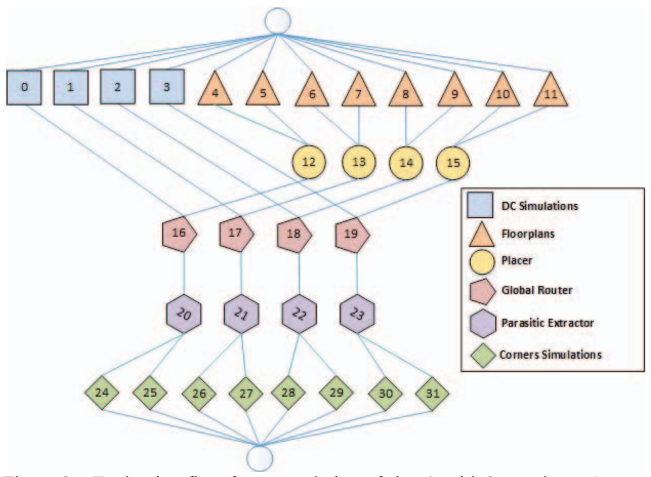

Figure 2 – Evaluation flow for a population of size 4, with 2 templates, 1 test bench and 2 corners per circuit showing the dependencies between tasks.

The above graph can be easily represented as an adjacency matrix (each edge on the graph is represented either by 1 or 0 if there exists an edge between tasks) and serves as an input to the scheduler, where the initial schedules are constructed according to the precedence constraints described by the DAG.

D. Task Schedule Optimization

Scheduling tasks in multiprocessors systems can be described as finding a schedule for a general DAG so that the completion time or "makespan" is minimized. This problem is NP-Hard and has the following formulation:

$$\min \quad FT_i\,, \qquad i \in S_i \tag{1}$$

where FT_i is the finishing time of the ith schedule belonging to the set of schedules S. In this work, the implemented scheduling algorithm was a Heuristic GA-based Static Scheduler, which finds an optimal solution to the multiprocessor scheduling problem. The GA-based scheduler allows the implementation of the mentioned parallel model in a simple away, i.e., without complicated mathematical formulations such as those found in Branch & Bound-based

[13] and ILP-based schedulers [12]. Moreover, the GA-based scheduler is more suitable when the number of tasks increases, e.g., in the layout-Aware evaluation strategy, the populations used in the circuit optimizer usually has 128 elements or more and if each element has 8 floorplans and 8 PVT corners simulations, the total number of tasks in the DAG is $128 + 128 \times 8 + 128 \times 8 = 2176$, and gives 2175! combinations which mathematical-based solvers would struggle to efficiently process that number of tasks [13]). The developed GA-based scheduler, works by applying the genetic operators of crossover and mutation in order to find an optimal schedule, in the schedule space S, that minimizes the total completion time of a given parallel application.

1) *Chromossome Representation*

The chromosome is a sequence of gene pairs of integer values (t_i, p_j) where p_j stands for the j^{th} processor executing the i^{th} task. The order and permutations of the tasks is generated respecting the adjacency matrix. Figure 3 shows a valid chromosome for the layout-aware sizing process illustrated in Figure 2. A population of chromosomes is initialized using a random process, but always generating valid gene sequences by following the precedence constraints encoded in the adjacency matrix.

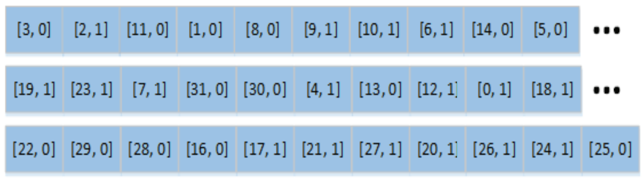

Figure 3 – Permutation of a chromosome.

2) *Crossover and Mutation*

The genetic operators, crossover and mutation, are applied to generate new offspring at each new generation. As mentioned earlier, the GA implements a permutation-based chromosome with the use of integers, so the genetic operators have to be suitable to this data type. The Partially Mapped Crossover (PMX) in combination with a heuristic mutation were chosen. The PMX, considers two crossover points, selected randomly, from the parent's genes to produce the new offspring, as illustrated in Figure 4.

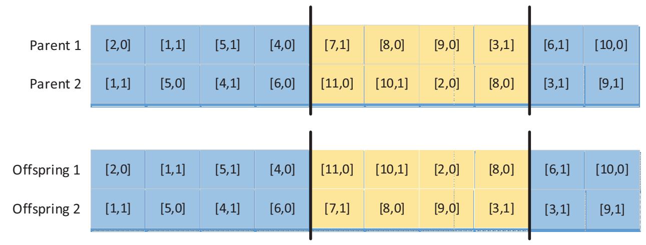
Parent 1 [2,0] [1,1] [5,1] [4,0] [7,1] [8,0] [9,0] [3,1] [6,1] [10,0]
Parent 2 [1,1] [5,0] [4,1] [6,0] [11,0] [10,1] [2,0] [8,0] [3,1] [9,1]
Offspring 1 [2,0] [1,1] [5,1] [4,0] [11,0] [10,1] [2,0] [8,0] [6,1] [10,0]
Offspring 2 [1,1] [5,0] [4,1] [6,0] [7,1] [8,0] [9,0] [3,1] [3,1] [9,1]

Figure 4 – PMX crossover.

Then, in the heuristic mutation, λ genes are chosen randomly and all the possible permutations for the chromosome are generated from those genes, and evaluated. Finally, the best chromosome is chosen from those permutations, as illustrated in Figure 5 for $\lambda = 3$.

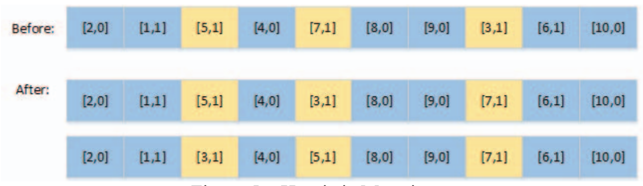
Before: [2,0] [1,1] [5,1] [4,0] [7,1] [8,0] [9,0] [3,1] [6,1] [10,0]
After: [2,0] [1,1] [5,1] [4,0] [3,1] [8,0] [9,0] [7,1] [6,1] [10,0]
[2,0] [1,1] [3,1] [4,0] [5,1] [8,0] [9,0] [7,1] [6,1] [10,0]

Figure 5 – Heuristic Mutation.

3) *Licenses and Constraints*

One of main goals of this work, is the efficient use of simulator licenses by the simulation-based optimizer. A schedule is considered feasible if, at each time interval t, the tasks that are active consuming a given resource do not exceed the total available resources of that type. Figure 6 shows an example of a schedule with tasks that consume software licenses.

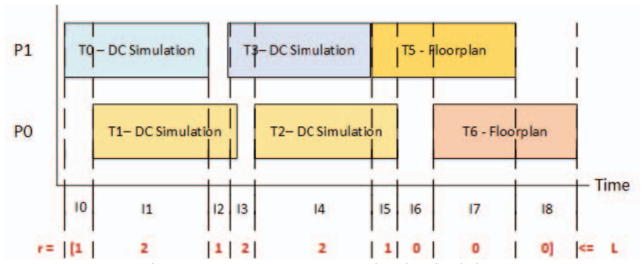

Figure 6 – Resource Constrained Schedule.

IV. RESULTS

The proposed methodology was integrated in an established design automation tool, AIDA [3], and validated using real IC design examples, particularly, a two-stage Miller amplifier for the UMC 130 nm CMOS design process, loaded with a 10 MΩ resistor in parallel with 1 pF capacitor and biased with a current of 10 µA. The two objectives chosen, for the design optimization, were to maximize the DC gain and minimize the circuit area. The circuit schematic is illustrated in Figure 7 and the specifications are shown in Table 1.

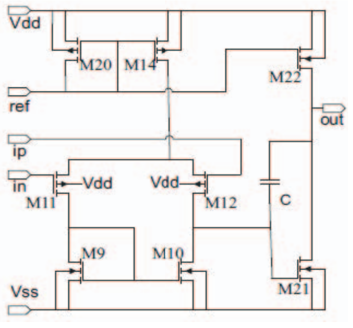

Figure 7 – Schematic of a Two-stage Miller amplifier.

The design of the circuit was tested for a typical sizing strategy (no corners), using 64 elements in the NSGA II optimization kernel, for 100 iterations. The proposed approach was tested for 4 different scenarios: (1) sequential execution; (2) 2 threads and 2 licenses, where the population is split in half between the 2 threads and each thread executes sequentially the Layout-Aware evaluation procedure; (3) 4 threads and 2 licenses for the Scheduled Layout-Aware approach; and (4) 6 threads and 2 licenses for the Scheduled Layout-Aware approach.

Table 1 - Two-stage Miller amplifier specifications

Specifications	Measure	Target	Units	Description
Objectives	area	Minimize	µm	Area
	A_0	Maximize	dB	DC Gain
Constraints	IDD	≤ 80	µA	Current Consumption
	GDC	≥ 50	dB	Low-Frequency Gain
	GBW	≥ 1	MHz	Unity Gain Frequency
	PM	≥ 55	º	Phase Margin
	PSRR	≥ 55	dB	Power Supply Rejection Ratio
	SR	≥ 0.8	V/µs	Slew Rate
	Voff	≤ 1	mV	Offset Voltage
	No	≤ 400	µVrms	Noise RMS
	Sn	≤ 100	nV/√Hz	Noise Density
	OVP^n	≥ 100	mV	VTH-VGS (PMOS)
	OVN^n	≥ 100	mV	VGS-VTH (NMOS)
	DP^n	≥ 50	mV	VDSat-VDS (PMOS)
	DN^n	≥ 50	mV	VDS-VDSat (NMOS)

Figure 8 illustrates the resulting Pareto Front of valid design solutions that are obtained, in any of the scenarios.

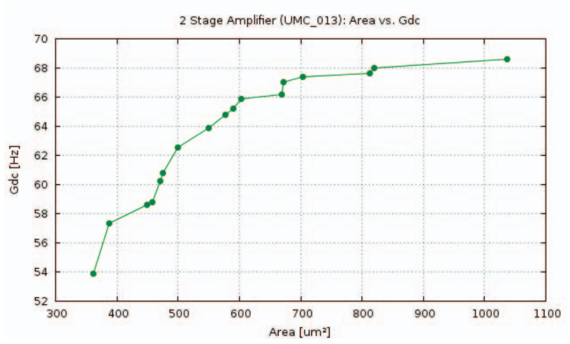

Figure 8 – Obtained Pareto Front.

The execution time of the several scenarios is summarized in Table 2. The proposed approach achieves a reduction of about 18% and 22% of the execution time when using 4 and 6 threads respectively while using the same number of licenses, 2. The results show effective gains when using the proposed approach for a typical analog circuit structure, which anticipates extended efficiency for more complex circuit structures and/or analysis which require large computation time per evaluation.

Table 2 – Summary of the results.

# of Threads	# of Licenses	Execution Time [h:m:s]
1	1	1:44:59
2 (trivial)	2	0:51:19
4	2	0:42:05
6	2	0:40:07

V. Conclusions

The proposed methodology consists of an innovative scheme for optimal task scheduling in a state-of-the-art layout-aware sizing approach. The proposed scheduling approach models the design flow using a DAG and considers a GA optimizer to obtain the optimal schedule solution under operational constraints such as a limited number of software licenses. The proposed approach was integrated in a design automation environment and validated using benchmark circuits. The results show a clear efficiency gain of 18% and 22% when both compared to the traditional parallelization schema for the same licensing cost.

VI. Acknowledgments

This work was supported in part by the Instituto de Telecomunicações (Research project OPERA PEST-OE/EEI/LA0008/2013) and by the Fundação para a Ciência e Tecnologia (Research project DISRUPTIVE EXCL/EEI-ELC/0261/2012, Research project UID/EEA/50008/2013 and Grant FCT-SFRH/BPD/104648/2014).

References

[1] E. Roca, M. Velasco-Jiménez, R. Castro-López and F. V. Fernández, “Context-dependent transformation of Pareto-optimal performance fronts of operational amplifiers”, Analog Integrated Circuits and Signal Processing, vol. 73, no. 1, pp. 65-76, 2012.

[2] H. Gupta, and B. Ghosh, “Analog Circuits Design Using Ant Colony Optimization”, International Journal of Electronics, Computer & Communications Technologies, vol. 2, no. 3, pp. 9-21, 2012.

[3] R. Martins, N. Lourenço, A. Canelas, R. Póvoa and, N. Horta, “AIDA: Robust Layout-Aware Synthesis of Analog ICs including Sizing and Layout Generation”, International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2015.

[4] R. Castro-Lopez, O. Guerra, E. Roca, and F. V. Fernandez, “An Integrated Layout-Synthesis Approach for Analog ICs”, IEEE Transactions on Computer-Aided of Integrated Circuits and Systems, vol. 27, no. 7, pp. 1179-1189, 2008.

[5] R. Santos-Tavares, N. Paulino, J. Higino, J. Goes, and J. P. Oliveira, “Optimization of Multi-Stage Amplifiers in Deep-Submicron CMOS Using a Distributed/Parallel Genetic Algorithm”, IEEE International Symposium on Circuits and System (ISCAS), pp. 724 - 727, 2008.

[6] P. Pan, H. Cheng, and C. Lin, “PAGE: Parallel Agile Genetic Exploration towards Utmost Performance for Analog Circuit Design”, Design Automation & Test in Europe Conference & Exhibition (DATE), pp. 1849 – 1854, 2013.

[7] T. G. Rabuske, R. B. Pinheiro, J. Fernandes, and C. R. Rodrigues, “PyCO: A Parallel Genetic Algorithm Optimization Tool for Analog Circuits”, IEEE International Symposium on Circuits and System (ISCAS), pp. 3266-3269, 2012.

[8] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, and J. R. Hellums, “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.

[9] N. Lourenço, R. Martins, and N. Horta, “Layout-Aware Sizing of Analog ICs using floorplan & routing estimates for parasitic extraction”, Design Automation & Test in Europe Conference & Exhibition (DATE), pp. 1156-1161, 2015.

[10] R. Martins, N. Lourenço, A. Canelas, and N. Horta, “Electromigration-Aware Analog Router with Multilayer Multiport Terminal Structures”, Integration, The VLSI Journal, Elsevier, vol. 47, no. 1, pp. 532-547, 2014.

[11] N. Lourenço, A. Canelas, R. Póvoa, R. Martins, and N. Horta, “Floorplan-Aware Analog IC Sizing and Optimization based on Topological Constraints”, Integration, The VLSI Journal, Elsevier, vol. 48, no. 1, pp. 183-197, 2015.

[12] Y. Yi, W. Han, X. Zhao, A. T. Erdogan, and T. Arslan, “An ILP Formulation for Task Mapping and Scheduling on Multi-core Architectures”, Design Automation & Test in Europe Conference & Exhibition (DATE), pp. 33-38, 2009.

[13] I. Chakroun, and N. Melab, “Operator-level GPU-accelerated Branch and Bound algorithms”, International Conference on Computational Science, ICCS, pp. 280-289, 2013.