Computer Automated Design of Analog Circuits

Dennis Wu - 981073520

Nov 15, 2002

1 Introduction

Transistor feature sizes have shrunk to the point where entire systems can be economically implemented on a monolithic IC. Often these circuits have an analog interface to the natural world. With the increasing availability of chip real estate, designs are increasing in size and complexity. To keep the design times manageable, computer automated design tools have been introduced for the digital portion of these mixed signal designs. The analog component however is still often hand crafted.

Computer automated design tools hope to improve the analog design process in a number of ways. An automation system can shorten design times, which will improve productivity and bring products to market faster and at lower cost. They also free the designer from tedious and error prone tasks. Once errors are worked out of a design automation system, the tool can be used repeatedly to perform the same task correctly. Finally, by hiding design complexities from the user, more circuit designers from novice designers to expert designers, become able to design standard analog circuits.

The techniques used by humans to design analog circuits is complex. Analog circuits often target multiple, conflicting design goals such as gain, speed, linearity, noise, power, input/output impedance, voltage swings and supply voltage. Solutions to certain analog problems usually cannot be easily used to solve other problems. The lack of a systematic approach to building analog circuits makes it difficult to build an automated solution.

The circuit design process usually consists of topology selection, sizing and optimization, simulation and verification, layout, extraction and resimulation. The portion of the design process that I investigate is restricted to topology selection and device sizing.

2 Genetic Algorithm

The genetic algorithm is a heuristic for optimizing a cost function. The basic idea is to evolve a set of solutions towards the optimal through mutation, crossover and selection based on the fitness of the solution. The core algorithm is shown below:

```
chromosome genericGeneticAlgorithm(){

  initializeGeneration();
  bestValue = calculateFitnessOfSolutions();

  while (bestValue < desiredValue){
    Selection();
    Crossover();
    Mutation();
    bestValue = calculateFitnessOfSolutions();
  }

  return( chromosomeOf(bestValue));
}
```

[2]

The inital pool of solutions is usually random. In some cases, known circuits are inserted in the initial pool to speed up convergence, however, this tends to influence the algorithm towards a local (but not global) optimum [2]. To measure the fitness of a solution, the performance parameters are obtained through SPICE simulations and the weighted sums of these parameters are compared.

Selection involves picking solutions that will be continued into the next iterration. Selection aims to keep the fitter solutions (those with lower costs). At the same time, some randomness is introduced in selection to increase the

chances of finding the global minimum. Properties that are considered unfit in one generation may turn out to be advatanges in later generations.

In the genetic algorithm, two methods exist for producing new solutions. Crossover generates new solutions by combining the properties of two existing solutions. Mutation generates new solutions by making random changes to an existing solution. These new solutions are added to the population. Finally, the algorithm terminates once a valid solution is found.

The the flexibility of the genetic algorithm is limited only by how the circuit is enocded. Naturally, only valid circuits should be investigated to avoid wasting computational time. Flexible circuit encodings allow the genetic algorithm to explore a broader range of circuit topologies. This can result in novel designs not normally considered by human designers. In one case, an FPGA was used to contruct a frequency detecting circuit [2].

The genetic algorithm does have drawbacks however. Circuits produced using the genetic algorithm may be difficult to understand making modifications difficult. The great flexibility in circuit posibilities also means a large number of topologies must be considered. Many of these topologies are unpromising and would have been easily ruled out by human designers. Both promising and unpromising topologies must go through the computationaly expensive process of simulation. For a population of 200 chromosomes evolving over 100 generations with each simulation only taking one second, the time spent only on simulations can take 5.5 hour [2].

3 BLADES: Expert Systems

The problem with the genetic algorithm is that it runs too slowly to be practical. It wastes time investigating circuit topologies that circuit designers could pick out immediately as unpromising. Expert systems, use a different approach. Instead of relying on randomness to find a solution, they attempt to mimic the reasoning process used by human designers. This reduces the search space to only those circuit topologies that a human designer would consider.

One of the earliest works on expert systems for analog circuit design is BLADES [3]. Blades contains a knowledge base that stores formal mathematical techniques and intuitive reasoning procedures. I now describe how Blades applies the human reasoning process to design problems.

Given the specficiations for a circuit, Blades determines the type of circuit that is being designed. Based on the circuit type, Blades partitions the system into subsystems and assigns specifications to each of the subsystems. The partitioning process is performed using if-then rules designed for the particular class of circuits. For example, the following rule is used for opamp circuits.

```
(op amp-design-1
if
  (op amp)
then
  (go into partitioning mode)
)
```

The opamp is divided into a differential gain stage, a direct gain stage, and an output stage. Other stages are possible depending on the specification of the opamp. For example, an additional input stage is added if it is a four terminal opamp.

```
(four-terminal-op amp
if
  (number of input terminals equal to 4)
then
  (add another input stage)
)
```

The circuit is recursively partitioned down to function blocks. Function blocks are the most common primatives that designers use. They can be a group of elements or consist of a single transistor. Each function block primative has an associated "subcircuit design expert" that specializes in building that function block. The subcircuit design expert implements the function block using a fixed topology and a set of design equations to calculate the subcircuit parameters.

In situations where several function blocks can be used to meet requirements, the function block with the fewest transistors is selected. A subcircuit design expert is successful if it is able to meets specifications for the function

block. Otherwise, it reports an error and a different function block is attempted. For example, for a current mirror with a given output impedence, BLADES will attempt to use a two transistor circuit. If this circuit does not deliver the required impedance level, then a Wilson current mirror will be tested. If that fails, then a cascode configuration is used.

Expert systems, make two major contributions to analog circuit design. Firstly, the knowledge based approach to designing circuits greatly reduces computational time. BLADES takes 3 to 6 CPU seconds to design an operational amplifier on a fully loaded VAX/785 running the UNIX operating system. Secondly, hierarchical decomposition of the circuit into function blocks allows investment put in designing subcircuit design experts to be reused in other circuits.

The restrictive nature of expert systems does have some drawbacks. They only explore designs that the programmer intends and do not consider the more novel designs explored with the genetic algorithm. Also the setup time for adding or modifying circuit types is time consuming and requires expert designers. Also, the use of simplified mathematical models causes inaccuracies and can result in poor topology or parameter selection. Finally, Blades ignores the strong coupling that exists between different functional blocks during decomposition.

4 ISAID: Sequential Decomposition and SPICE

Hierarchy is not strict in analog circuits because of strong coupling between devices in the circuit. Blades deals with this by collapsing strongly coupled devices into one function block and designing the function block in one shot. The problem with this approach is that larger function blocks are more difficult to implement. Also, large function blocks are too specific too be reusable in other circuits. The reusability benefit of hierarchical design becomes lost.

ISAID [4], another analog IC design automator, deals with this problem by using a technique called sequential decomposition. Instead of using a straight top-down decomposition, as it is done in [3] and [5], where the subblocks are designed independently, sequential decomposition allows information to travel upwards as well as downwards through the hierarchy. This enables a higher level in the hierarchy to monitor the performances of already designed lower-level circuit blocks and to asses the interactions

between them.

ISAID does this in the following way. Say circuit A is decomposed into subcircuits B1, B2 and B3. A subcircuit is selected, say B1, specifications are assigned to it and the subcircuit is synthesised. The performance characteristics and estimated device parasitics for B1 are returned to circuit A. The performance characteristics of B1 can be used to influence the design of subsequent circuit blocks. For example, if subcircuit B1 returns with performances that exceed those originally requested, then the specifications for subcircuits downstream can be relaxed. Performance conflicts may occour when a subblock downstream cannot be made to satisfy the requirements set by previous subblocks. Careful scheduling of the subblocks can reduce such conflicts. Subcircuits that generate a lot of constraints are usually synthesized first.

Another development of ISAID is that it closes the loop by using simulation results to verify and adjust the circuit topology. With Blades, the entire design process is completed using simplified mathematical models. Inaccuracies in these simplified models may cause the designer to make bad choices. ISAID uses a two step design and assess approach where the design is done using level 1 models, and the analysis is done using level 2 models. Differences in the results between the two models are fed back into the system. For example, say the transconductance of a transistor is designed to be g_m but is evaluated using level 2 models to be $g_{m'}$. Let the difference be $\triangle g_m = g_m$ - $g_{m'}$. If $g_{m'}$ is found to be outside the acceptable range, then the design will be repeated with a new transconductance specification of $g_m + \triangle g_m$. The assumption is that the difference $\triangle g_m$ will remain constant so that new the transconductance gain will become $g_{m'} + \triangle g_{m'} = g_m$. This is only valid for differences that are small which is typically the case with ISAID. The number of adjustment interations needed is found to be usually 1 or 2.

5 ANACONDA: Simulation Based Synthesis

Previous computer design automation systems use simplified transistor models to produce designs quickly. ISAID, attempts to correct the simplified models' deficiencies by feeding back the results of the analysis into the system. To produce even better designs, the same industrial-strength simulator used during verification should also be used the design stage. This is the

approach taken by Anaconda [6].

The problem with going to industrial-strength simulation completely is speed. The fact that industrial strength spice simulations take longer to compute than simplified mathematical models is not the problem. The problem is that when using an equation based approach, results are obtained in one shot, whereas with SPICE, the result must be obtained iteratively. This iterrative approach works as follows: A circuit is simulated with set of parameters. If the input mix does not produce satisfactory results, the inputs are adjusted until satisfactory results are obtained. Depending on the algorithm used and the level of optimization desired, a circuit may need to be simulated in the order of 100 000 times before arreiving at a satisfactory result. Commercial circuit simulators are not designed to be invoked 100 000 times in the inner loop of a numerical analyser.

A proven optimization algorithm called simulated annealing is first disscussed. The algorithm is then modified for network level parallesim to decrease computation time.

Simulated annealing attempts to minimize a cost function. The optimization problem for analog circuits is formulated as follows: Given a fixed topology, find devices sizes that minimize the difference between the measured specs and the desired specs. The circuit begins at a random starting point and high temperature. A rating is computed for the circuit using SPICE simulations. The inputs are then adjusted with some randomness. The magnitude of the adjustment decreases as the temperature decreases. The rating for this circuit is computed. If the rating improved, then the new inputs are kept. If the rating does not improve, then the new inputs are kept with probability $e^{-1/T}$ where T is the temperature. The temperature decreases at a rate dictated by an annealing schedule. As the temperature decreases, the inputs settle about some optimal values. The point of adding randomness early in the annealer is to escape local minimums in search of the global minimum.

Anaconda modifies the simulated annealing algorithm into what they call stochastic pattern search (SPS). The modifications aim to distribute the processing time over multiple workstations. SPS begins with a random population of circuits. In each iterration, k circuits are selected randomly for modification. New circuits are generated by applying pattern searches to a small locus around a given solution point. The starting perturbation in the pattern search starts large to skip over local minima and decreases over

time. SPS does not accept uphill movement as in simulated annealing. The justification is that by starting with a random population, there is enough randomness to lead the search to a global minimum. The new circuits are added back into the population and the worst circuits in the population are eliminated.

The pattern search for each of the k candidates can be processed in parallel. And for each of the k candidates, t pattern searches are simulated. Thus allowing the problem to be distributed over kt workstations. Using this distributed approach, 50 000-100 000 circuit candidates can be fully simulated in a few hours.

6 Conclusion

In this paper a number of approaches to the computer automated design problem are investigated. Genetic algorithms produce novel designs but are too slow to be practical. Expert systems, such as Blades, make computation time reasonable by following the human reasoning process and using hierarchical decomposition and simplified equation models. ISAID improves on this by using sequential decomposition to adjust for coupling between subcircuits. It also uses SPICE in a design and verify loop to reduce inaccuracies introduced by the simplified equation models. Finally, Anaconda makes practical the use of an industrial-strength simulator in both the design and verify stages by exploiting parrallism in the stochastic pattern search.

References

[1] Navid Azizi, "Automated analog circuit design using genetic algorithms," 2001.

[2] Fatehy El-Turky and Elizabeth E. Perry, "Blades: An artificial intelligence approach to analog circuit design," *IEEE Transactions of Computer-Aided Design, Vol. 8. NO. 6*, June 1989.

[3] Costas A. Makris Christofer Toumazou, "Analog ic design automation: Part i-automated circuit generation: New concepts and methods," *IEEE*

Transactions of Computer-Aided Design of Integrated Circuits and Systems, VOL. 14. NO. 2, February 1995.

[4] Francky Leyn Koen Lampaert Jan Vandenbussche Georges G. E. Gielen Willy Sansen Petar Veselinovic Geert Van der Plas, Geert Debyser and Domine Leenaerts, "Amgie - a synthesis environment for cmos analog integrated circuits," *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems, VOL. 20. NO. 9*, September 2001.

[5] Rob A. Rutenbar Richard Carley Rodney Phelps, Michael Krasnicki and James R. Hellums, "Anaconda: Simulation-based synthesis of analog circuits via stochastic pattern search," *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems, VOL. 20. NO. 9*, June 2000.