A DIFFERENTIAL MODEL APPROACH TO ANALOG DESIGN AUTOMATION

D. J. Klein and M. L. Manwaring
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752
509-335-6602

Abstract

A high-level analog design synthesis approach using differential equations as the hardware definition language is presented. From the differential equations, annotated with performance requirements, the design process generates a set of device requirements that can be used as the input requirements to existing low-level analog circuit synthesis tools which build the necessary low level analog devices. An object oriented implementation of the design methodology is presented. A design example is presented that explores the viability of this design approach.

I. Introduction

Digital designers enjoy the use of many CAD tools that aid in the simulation, synthesis, routing, analysis, and verification of their designs. Unfortunately, while analog designers also enjoy the use of CAD tools in areas of simulation and analysis, the area of design synthesis has yet to be completely and successfully addressed. For this reason, the design of an analog system in an analog or combined analog-digital (mixed signal) ASICs takes a considerable amount of time in comparison with a similarly complex, purely digital system.

Fortunately, there has been reasonable success with automated synthesis and generators of low level or "primitive" devices such as operational amplifiers and comparators [1-6]. Also, the development of knowledge base technology that can contain previous design solutions and that allow searches to be made with incomplete search criteria has been completed [7]. What is missing is a method of combining these two elements, the knowledge base and generators, into a system takes a high level description of a design problem and translates the description into a set of low level analog integrated components.

II. Analog CAD Background

As mentioned above, there are several CAD tools that the digital designer has that are not available for the analog designer. Perhaps the most significant of these is the lack of a widely accepted general purpose hardware description language (HDL) which would allow high level description of analog design solutions similar to the digital domain's VHDL. Also missing are standardized methods for complete specification of finished analog designs and their design heuristics.

These tools have proved challenging to develop because of the following problems exhibited by analog design in general: lack of clear hierarchical boundaries, process dependence at almost all levels of the design hierarchy, lack of regularity and repeatability from design to design, strict transistor sizing and matching requirements, and the complex issues of floor planning, routing, modeling and geometry dependence.

Fortunately these problems have not limited the development of tools that automate the synthesis of low-level analog integrated circuits. Some of the tools that deal with such synthesis automation are OPASYN [1], OASYS [2], and IDAC [3]. Also, there are tools dealing
with placement and routing synthesis such as ILAC [8] and ANAGRAM [9].

There have been attempts at higher level analog design automation. These include the use of the algorithmic high-level analog design description language, ADD [10] with silicon compilation of analog functional blocks. Also the automation of switched-capacitor filters [11], which are fairly structured by nature, has been carried out.

The design approach documented here imposes a structure at the top level of its hierarchy by using differential equations (DE) as its HDL. This allows for a functional, hierarchical decomposition of the design into its low-level analog subsystems. Also, the use of DEs allows for abstract functional simulation of design. The required analog subsystems can then be described by a combination of functional and performance. Once the initial translations are completed, the design process works interactively with existing knowledge base or synthesis tools to generate the low-level specification of the design. This presentation assumes the existence of the necessary low-level device synthesis tools and does not specifically address this area of analog CAD tools. The authors are aware that more development is necessary in this area.

### III. A Design Automation Procedure

The analog environment has all the circuit components necessary to build basic functions, such as integrators, summers, scalers, multipliers, and references needed for implementation of the integral form of the DE models of desired analog solutions. Further, by using switched-capacitor realizations of the circuits for these basic functions, it is possible to implement VLSI solutions of the DE models. Realizing this and that many analog systems can be modeled with systems of DE, we chose to use as our HDL the DEs themselves, annotated with the necessary performance and environment requirements. This use of DE systems as our HDL also avoids the complexity and time necessary to define a language similar to digital’s VHDL. A brief description of essential definitions of the DE system grammar used in this design approach is shown in Table I.

The design process consists of three basic steps. These are parsing, conversion of the DE system to integral form, and construction of low-level specifications of the resulting functional blocks. On completion of these steps, assuming the synthesis tools are successful, the resulting low-level device specifications and the interconnection information are passed to automated routing and placement tools.

### TABLE 1: ESSENTIAL GRAMMAR DEFINITIONS FOR DIFFERENTIAL EQUATION SYSTEMS

<table>
<thead>
<tr>
<th>&lt;Differential System&gt;</th>
<th>&lt;Differential Equations List&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Input Description List&gt;</td>
<td>&lt;System Requirements List&gt;</td>
</tr>
<tr>
<td>&lt;Different Equations List&gt;</td>
<td>&lt;Begin DE Statement&gt;</td>
</tr>
<tr>
<td>&lt;Different Equations&gt;</td>
<td>&lt;Begin DE Statement&gt;</td>
</tr>
<tr>
<td>&lt;Different Equations&gt;</td>
<td>&lt;Different Equations&gt;</td>
</tr>
<tr>
<td>&lt;Different Equations&gt;</td>
<td>&lt;Different Equation&gt;</td>
</tr>
<tr>
<td>&lt;Different Equation&gt;</td>
<td>&lt;Forcing Function&gt;</td>
</tr>
<tr>
<td>&lt;Forcing Function&gt;</td>
<td>&lt;Expression&gt;</td>
</tr>
<tr>
<td>&lt;Expression&gt;</td>
<td>;</td>
</tr>
<tr>
<td>&lt;Begin ID Statement&gt;</td>
<td>&lt;Input Description Reference&gt;</td>
</tr>
<tr>
<td>&lt;Input Description Reference&gt;</td>
<td>&lt;Begin ID Statement&gt;</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>1</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>&lt;Input Description&gt;</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>&lt;Input Description Reference&gt;:</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>maxFreq = &lt;Number&gt;</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>minFreq = &lt;Number&gt;</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>maxV = &lt;Number&gt;</td>
</tr>
<tr>
<td>&lt;Input Description&gt;</td>
<td>minV = &lt;Number&gt;</td>
</tr>
<tr>
<td>&lt;Output Description List&gt;</td>
<td>&lt;Begin OD Statement&gt;</td>
</tr>
<tr>
<td>&lt;Output Description List&gt;</td>
<td>&lt;Begin OD Statement&gt;</td>
</tr>
<tr>
<td>&lt;Output Description&gt;</td>
<td>&lt;Output Description&gt;</td>
</tr>
<tr>
<td>&lt;Output Description&gt;</td>
<td>&lt;Output Description&gt;</td>
</tr>
<tr>
<td>&lt;Output Description&gt;</td>
<td>&lt;Output Description&gt;</td>
</tr>
<tr>
<td>&lt;Output Description&gt;</td>
<td>&lt;Output Description&gt;</td>
</tr>
<tr>
<td>&lt;System Requirements List&gt;</td>
<td>&lt;Begin SR Statement&gt;</td>
</tr>
<tr>
<td>&lt;System Requirements List&gt;</td>
<td>&lt;Begin SR Statement&gt;</td>
</tr>
<tr>
<td>&lt;System Requirements&gt;</td>
<td>&lt;System Requirements&gt;</td>
</tr>
<tr>
<td>&lt;System Requirements&gt;</td>
<td>&lt;System Requirements&gt;</td>
</tr>
<tr>
<td>&lt;System Requirements&gt;</td>
<td>&lt;System Requirements&gt;</td>
</tr>
</tbody>
</table>

Now, before describing each of these three steps in detail, examination of these definitions in the grammar will aid in understanding what takes place in the three steps. The definitions of interest are the system requirements and the input and output descriptions. These three definitions define both the operating environment and the performance expected from the
analog design. The input descriptions define the operating ranges and impedances of the signals that will drive the circuit. The system requirements defines the fabrication process parameters, power sources, available clock signals, and the desired error tolerances. The output descriptions define the ranges and impedances of the devices the circuit will drive. These three definitions are key to the conversion of the purely mathematical functional description provided by DEs into realizable and useful integrated circuits.

A. Parsing: In this step, a DE system is checked for grammatical correctness and a graph representing the variable interconnection of the DEs is constructed. If the grammar of the DE system is incorrect, the error is identified and brought to the attention of the user.

B. DE to integral form conversion: Since differentiators are notoriously unstable, the DE system is converted to integral form. This is done using an algorithm that also has three basic steps. First it determines the order of each DE in the DE system. Then, for each DE, a set of integrators, equal in number to the order of the DE, are linked. Finally, the feedback and variable summation connections and scalers are assigned until all the branches of the graph produced in the parsing step have been accounted for. This process is graphically shown in Fig. 1. A brief description of the graphical symbols and their mathematical functions can be found in Table II.

\[
\ddot{x} + a\dot{x} - bx + c = f(t)
\]

(a)

\[
\begin{align*}
\text{Order} & = 2 \\
\sum & = \sum_{i=1}^{n} x_i(t) \\
\text{Scaler} & = K \cdot x(t) \\
\text{Multiplier} & = x(t) \cdot y(t)
\end{align*}
\]

(b)

(c)

Fig. 1. The DE to integral form conversion process.

C. Low-level device definition with performance and scaling assignments: This last step is the most important and the most difficult. In this step, several actions take place. First, the low-level device types that will implement the functions of step B are chosen using design heuristics. Then, the performance requirements of each device are determined from the system performance requirements list. Next the device performance requirements are checked against the limits imposed by the device environment and, if necessary scaling of the device requirements throughout the low-level devices is carried out. Then, the low-level device synthesis tools are interactively called to determine if the devices are realizable. If any of the tools fail, two steps are taken. First, an attempt is made to ease the failed requirement(s) of the failed device(s) by global low-level device requirement re-scaling. If this process fails, an attempt is made of redefining the low-level devices. Thus, step C is an iterative process that continues until either successful generation of the specifications for all of the necessary low-level devices is complete or it is determined that the synthesis tools are unable to produce devices meeting the design performance and environment requirements. Since this could be an NP-complete problem, a limit to the number of iterations step C may cycle through is provided.

At this point, it becomes necessary to explain the process of definition and redefinition of the low-level device types. Unlike its mathematical counterpart, an analog integrator and some configurations of summers and multipliers have gain. Also, an operation-amplifier (op-amp), in proper configuration, can, if connected with a properly configured switched-capacitor network, act as a combination of all the functions detailed in Table 2. Thus, optimization of the design by combination of the functions is possible. This sort of optimization is initially carried out. However, if a synthesis tool fails to meet the requirements or the system design heuristic points to problems with instability, this high level of integration may not be possible and the discrete functions may need to be built.
Thus, the results of this design process are the complete low-level device specifications necessary to drive autoplacement and routing tools. Note that, by keeping track of the iterative process of requirements assignment and low-level synthesis tool behavior in step C, a complete audit trail of the circuit design is possible. This trail can then be used to derive the design's heuristics.

IV. Implementation of The Design Process

The described abstract design approach was implemented in the C++ object oriented environment. This was done to gain the benefits of hierarchical description of the low-level analog circuits and the functions that are built from these low-level circuits. As the high-level, functional objects instantiated in step B are further defined, they must contain more and more information and often become combinations of lower level objects. The object oriented environment, with its encapsulation, inheritance, and polymorphism provides an elegant solution to this problem.

The classes used in implementing the design process are based on the DeviceClass. This class contains information and functions necessary for all levels of abstraction: instance and component names for the device, inputs and outputs of the device, a parent device reference, and the requirements of the device. From this base class, two device classes are derived, the CircuitClass and the FunctionClass. The hierarchy of these classes is shown in Fig. 2.

The FunctionClass encapsulates the characteristics of the connectivity network of low-level devices and the optimization functions. It includes a functional description, child object information, both analytical and the electrical connectivity of the child objects and the I/O ports. Objects belonging to the FunctionClass are instantiated in step B of the design process and can have as child objects both FunctionClass and CircuitClass objects. The top level of the design is contained in a FunctionClass object.

The CircuitClass encapsulates the device definitions and interface translation necessary for the synthesis tools. Objects of this class are instantiated in step C of the design process. The CircuitClass differs from the DeviceClass by adding the functions necessary for interfacing with the low-level synthesis tools. Because of the possibilities of a single circuit carrying out multiple functional tasks, it is possible to have CircuitClass objects that do nothing more than pass messages to another CircuitClass object on what its requirements are.

For optimization and hierarchy to work properly, a message passing scheme and a structure for handling the connectivity between objects and their parents and children is necessary. The connectivity is handled by having directional Port and Pin objects. The Port and Pin differ in that Port objects provide the connectivity information to the parent of a Circuit/FunctionClass object while Pin objects define the connections to the children of the Circuit/FunctionClass objects [12]. The messaging handling functions are built into base DeviceClass. Each object, on receiving a message, determines if the message is either for itself or one of its children. If the message is not for itself or for a child, it passes the message to its parent. Valid messages can be any of the following:

- Child object instantiation notice to parent for optimization purposes.
- Notification of success or failure of synthesis tool. If failure, this message includes the reason(s) for failure.
- Notification of performance requirement change.
- Optimization notice of circuit multi-function requirement.
- Optimization notice that circuit synthesis is not necessary.

For purposes of providing an audit trail, the each FunctionClass object keeps a history of the messages passed to it and its children. At the end of execution, these message histories are compiled by the top object as the audit trail. This audit trail can be used to determine heuristic information about the design.
V. Experiment

The original intent was to completely automate the design system. However, the C++ code for interfacing with the low-level synthesis tools was still under development at the time this document was written. Thus a hand-worked example problem was carried out using the design process described above. Note, this does not effect the potential for automation, since the steps of the design process were followed and in some cases, interaction meant simply invoking the proper tools. Thus, the authors feel confident that the design process can be automated.

The design problem chosen is a switched-capacitor implementation of third order low pass filter. The LC ladder circuit and its DE for this filter can be seen in Fig. 3. Given a desired 3dB frequency of 2.5 kHz with

\[
\frac{1}{RC} \frac{V_o}{V_o} = \frac{1}{RC} \frac{V_i}{V_i}
\]

(1)

Selecting R and C to give a 3 dB frequency of 1 kHz gives an RC component of 0.001. The system requirements were +2.5 and -2.5 volt power with a maximum phase error of 18 degrees with the maximum gain error of 1.66% at two decades past the 3 dB point. The input port of driving source was assumed to have a low impedance while the output port was assumed have infinite impedance. The input maximum voltage was .5 V.

The initial design has two op-amps, the first acting as the summer with a gain of 1000 and the second acting as the integrator. Because this was not possible given the maximum possible input voltage, the input voltage was directly connected to the summer while the feedback \(V_o\) was scaled by 0.001. An output amplifier with a gain of 1000 was used to scale the output. The op-amps used in this design had a folded cascode architecture with compensated outputs. The op-amps has the following characteristics: a 73 dB gain, a 27 MHz gain-bandwidth, a 57 degree phase margin and a slew rate of 100E6 V/s driving a 1 pF load. The timing scheme is two non-overlapping clocks for a half clock delay from input to output of each device. To achieve the accuracy desired, the clocks are run at frequencies of 1 MHz or faster.

The final design exhibited a low-pass frequency response with a 3 dB point of 1017 Hz with a 19.7 dB per decade roll off until the input frequency reached ~100 kHz. Beyond this frequency, a large amount of noise and chaotic behavior were observed in the output signal. Thus, while the desired system specifications were not exactly met, they were close enough that further tuning of the circuit could easily bring them with in range. The circuit could be improved with a better offset cancellation scheme and a method of calculating the switch frequencies that does not assume idea op-amps.

IV. Conclusions

The goal of the experiment was to prove the viability of the design process. While the design process could not be completely automated, the effectiveness of the design process for a general differential problem was demonstrated.

Possible directions for further research on this design process includes the following:

- Completion of methods for interfacing with all low-level synthesis tools. Ideally, this would mean the development of uniform data structures between various synthesis and simulation tools.
- Expansion of the DE grammar to include the more complex mathematical expressions employed in control problems.
- Development of method of accounting for keeping track of system accuracy given the individual components’ accuracy.

As a final note, the authors have explored so some extent the extension of the DEs to include non-linear functions. This has proved to be rather difficult because of problems in implementing non-linear functions in the analog environment. This is an area of research that will require considerable further research and development.

References


design reuse,” Master’s Thesis, Washington State
University 1991.
automation system,” 1990 IEEE International Symposium
system,” Proceedings of the IEEE 1989 Custom Integrated
Circuits Conf., San Diego, CA, pp. 4.3.1-4.3.5.
analog design description to layout: a new approach to
analog silicon compilation,” Proceedings of the IEEE 1989
4.4.1-4.4.4.
[11] Assael, J., P. Senn, and M. S. Tawfi, “A switched-
capacitor filter silicon compiler,” IEEE J. of Solid-State
[12] Hylander, P., personal conversations, unpublished,