Repartitioning and Technology Mapping of Electronic Hybrid Systems*

Christoph Grimm and Klaus Waldschmidt
EMail: {grimm|waldsch}@ti.informatik.uni-frankfurt.de
Johann Wolfgang Goethe - University, Frankfurt am Main

Abstract

The systematic top-down design of mixed-signal systems requires an abstract specification of the intended functions. However, hybrid systems are systems whose parts are specified using different time models. Specifications of hybrid systems are not purely functional as they also contain structural information. The structural information is introduced by partitioning the specification into blocks with a homogeneous time model. This often leads to inefficient implementations.

In order to overcome this problem, a homogeneous representation for behavior of hybrid systems – KIR – is introduced. This representation makes it possible to represent behavior in all time models in a common way so that the separation in different modeling styles is no longer necessary. Rules for re-writing the KIR-graph are given which permit the description of the same behaviour in another time model.

1 Introduction

The design of analog and mixed-signal system is still done rather bottom up. A more structured, top-down design of such systems would require an abstract specification of the intended functions. The behaviour of mixed-signal – or more general hybrid – systems is specified using different specification formalisms ([1], figure 1):

In *Differential Equation Specified Systems (DESS)*, signals and states change their values continuously and at all points in time. The behaviour is specified by differential equations. In *Discrete Time Specified Systems (DTS)*, signals and states change their values discontinuously at discrete points in time. The behaviour can be specified for example by automata or difference-equations. In *Discrete-Event Specified Systems (DEVS)*, signals and states change their values discontinuously at any point in time. Behaviour is specified for example by timed automata or communicating processes.

*This work has been sponsored by the DFG (WA 357/9).

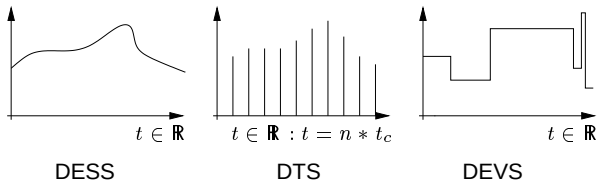

Figure 1: Time models of different specification formalisms in hybrid systems.

Because each specification formalism has different requirements for the behaviour to be described, in hybrid systems the formalisms are separated by, for example, the blocks of a block diagram. Each block then has its own homogeneous specification formalism. Which specification formalism is used for the description of an intended function depends on two different criteria:

First, it is important that the intended function can be described in an efficient way.

Second, a specification formalism has to be chosen depending on the tool and design methodology (register-transfer synthesis, high-level synthesis, module generators for filters, ...) that is intended to be applied on the specification.

In [2] and [3], approaches for systematic top-down design of analog and mixed-signal systems are proposed. Both rely on the intuitive, manual partitioning done *before* specification. Therefore, the specification is not a purely functional description of an intended behaviour – The specification also implies a partitioning of the structure that has to be designed:

1. The choice of the specification formalism determines the domain (RT-Synthesis, High-Level Synthesis, module generators for DESS) of its implementation, although implementation in another domain could be more efficient.

2. The separation of different time models intro-

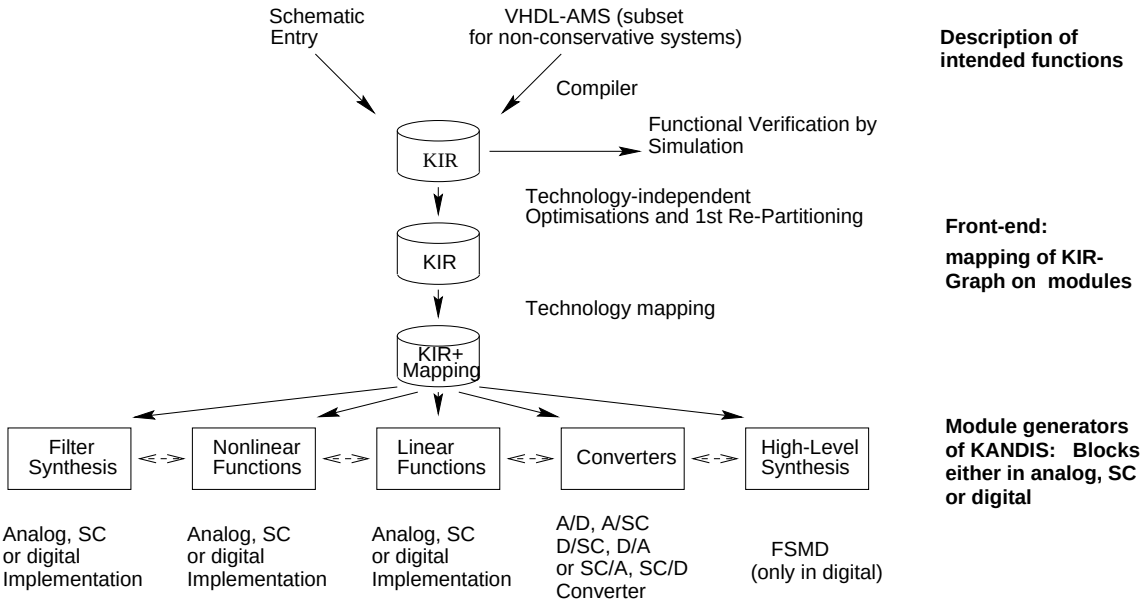

Figure 2: The new front-end of KANDIS.

duces the structure of a block diagram. More efficient structures may exist.

The tool KANDIS[4] supports the choice of an implementation-domain for single blocks of a block diagram by estimations, starting from an initial partitioning into analog and digital blocks. Single blocks, whose behaviour is described functionally can be implemented either in analog, in digital or with switched capacitors (SC). Blocks whose behaviour is described algorithmically are implemented in digital using high-level synthesis. The problem is that the structure of the block-diagram cannot be overcome. For this reason we try to find a good mapping of the specification onto blocks already in the front-end. These blocks will then be designed with the help of the tool KANDIS.

In this paper, we give an overview of the top-down methodology we use in the new front-end of the tool KANDIS (figure 2,[5]). In this frontend, the KANDIS Intermediate Representation (KIR)[6] is used as a homogeneous representation of hybrid systems (section 2). The structuring of the specification in blocks is then no longer necessary (section 3). With the help of graph-rewriting, the time-model of single nodes of a KIR-graph can then be changed (section 4). Finally, the KIR-graph can be mapped onto a block-diagram with a structure that leads to a more efficient implementation (section 5).

In difference to previous approaches, our approach extends the top-down methodology by a more abstract specification, and a methodology which allows to map this specification systematically onto a block-diagram, which is the structure of the implementation.

2 Description of hybrid systems with KIR

Known graph-based models have either a discrete (e. g. Petri-Nets) or a continuous (e. g. Bond-Graphs) implicit time-model. The time-model is given by the semantics of the graph-based model. Therefore, they cannot be used for the description of hybrid systems.

The KIR-Graph permits the combined use of different activation rules and time models in one graph; the semantics of KIR describe the behaviour at the interfaces of different time models (see also [7]).

Edges represent directed signals that transport values from an origin to a destination-node. Edges have one of the types REAL, INTEGER or ENUMERATION. If the origin node of an edge is not active, the edge keeps the last value assigned from its origin node. Nodes describe relations between in- and out-edges. For each node we specify:

- A function f.
- An activation rule a.

The function f can be specified declaratively by giving one of a set of predefined functions (i. e. add, integrate, ...). Alternatively, f can be described operationally by a KIR-Graph describing an algorithm

or the structure of a filter. This makes the definition of a KIR-Graph recursive. The hierarchical structure allows us to express partitioning decisions. The activation rules can be chosen from a set of predefined activation rules:

- a_{DESS}: The activation rule a_{DESS} leads to a continuous activation of its node. Therefore, f describes a functional relation between the signals of the in- and the out-edges. KIR-Graphs whose nodes have the activation rule a_{DESS} can be compared to continuous-time block diagrams known from control-theory.

- a_{DTS}: The activation rule a_{DTS} leads to an execution of f in constant time-steps t_{clk} KIR-Graphs with only the activation rule a_{DTS} can be compared to discrete-time signal-flow graphs. As opposed to discrete-time signal-flow graphs, nonlinear operations are allowed.

- a_{DEVS}: The activation rule a_{DEVS} executes its function when at least one signal of a subset $E_a \subseteq E_{in}$ of the in-edges E_{in} has changed its value. KIR-Graphs with only the activation rule a_{DEVS} can be compared to data-flow graphs.

Since the nodes of a KIR-graph dont have to have the same activation rules, it can occur that a value is required at a point in time when a node is not active. Here, we define that, if the origin of an edge is not active, the edge keeps the last value assigned from its origin-node.

2.1 Simple Example

In figure 3 the block diagram of a signal generator is given. This block diagram is already partitioned in blocks with intended analog and digital implementation. The example given is intentionally partitioned poorly. It consists of three blocks:

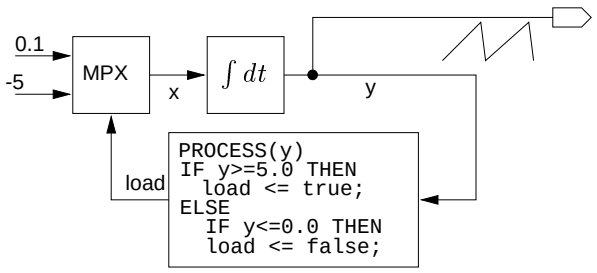

Figure 3: Block diagram of a signal generator.

The most important part of the signal generator is the integrator generating a continuously increasing signal y from a value x on its input. The value of x can be chosen by a switch depending on the boolean value *load*. These two blocks are described using a continuous time model. We suppose an implementation of these blocks in analog, as the integration over the continuous time cannot be implemented precisely with discrete-time digital circuits. The calculation of the value *load* depending on the signal y has been described algorithmically (discrete-event time model). There exists no systematic way to transform algorithmic specifications into analog circuits. Therefore, we suppose this block is to be implemented in the digital domain.

In this simple example, the choice of the specification methodologies has unintentionally determined the domain of implementation. The structure which has been introduced is ineffective, because an A/D converter will be required between the analog integrator and the process which is implemented in digital. As we show below, another representation of the same behavior leads to a more efficient implementation.

In figure 4 the KIR-Graph of the signal generator is given. As opposed to the block-diagram in figure 3, the behaviour of each block is now described in a unique, graph-based way. The block described algorithmically is now divided into a small graph with feedback which represents the discrete state of the process, two comparisons and some nodes producing constants.

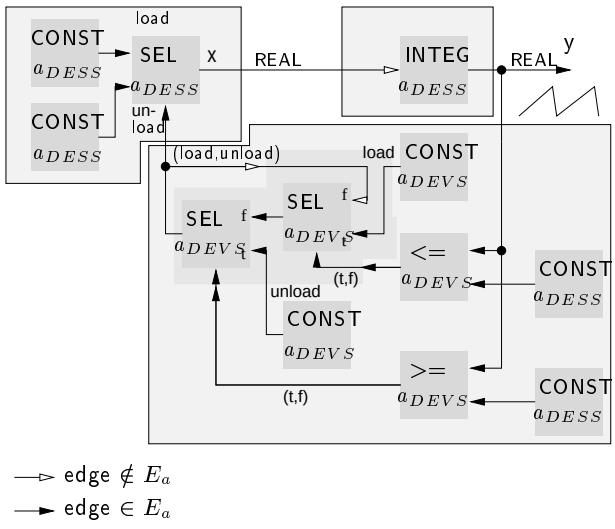

Figure 4: KIR-Graph of the signal generator with structure from block-diagram.

3 Removing the structure of a block diagram

The partitioning of the specification of a hybrid system into blocks with different time models was necessary to separate different modeling methodologies.

Because in KIR each node – and therefore each elementary function or operation – has its own activation rule, this separation is no longer necessary. Therefore, the partitioning in different blocks can be removed. This results in a hierarchically flat graph as in figure 5. As opposed to the original block-diagram, the granularity of the basic blocks/nodes is very fine. The nodes of this KIR-Graph are no longer blocks whose complex functions are described in different modeling methodologies, but nodes with very simple and elementary functions: arithmetical and boolean functions, selection, integration/differentiation over time or store (nodes with a_{DEVS} and $E_a \neq E_{in}$).

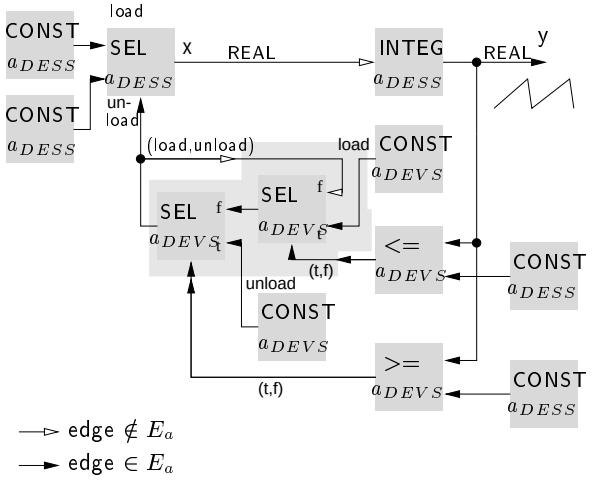

Figure 5: Flattened KIR-Graph of the signal generator.

On the hierarchically flat KIR-Graph many known optimizations are applied that produce descriptions of the same behavior which lead to more efficient implementations:

- Common subexpression elimination is applied on the KIR-Graph. This optimization reduces the number of nodes and later the number of modules generated. We apply common subexpression elimination[8] on descriptions in all (even mixed) time models and not only separately on single blocks which are modeled algorithmically. Furthermore, constant folding removes constant expressions.
- Representations of transfer functions in the frequency domain are transformed into different structures of simple nodes by partial fraction extension or factorization. This is not done in the front-end, but in the back-end of KANDIS[9].
- Rewriting rules that have been previously applied on discrete-time signal flow graphs (e. g. [10]) can also be applied on KIR-Graphs with the activation rules a_{DTS} and a_{DEVS}. They are not implemented in the front-end, because we prefer more functional (continuous-time) descriptions instead of more concrete discrete-time descriptions.

By removing the structure of the initial block-diagram as described above, the structure introduced by separating the different computational models in blocks can be eliminated, and optimization techniques can be applied on the unstructured graph. However, the activation rule still may determine the implementation of partitions of nodes.

4 Changing the time model

In this section, rules are presented that are used in the front-end and back-end of KANDIS to change the time model and implementation domain of a node.

The possibility to approximate continuous-time functions by a discrete-time function is well-known. The Shannon-Theorem[11] requires that the sampling frequency $1/t_{clk}$ is at least twice as much as the bandwidth that has to be transmitted. For getting discrete-time difference equations from linear DAE, various s- to z-transforms can be used[12].

The discrete-time representation of continuous-time functions is not possible in a general form which is valid for all frequencies. Therefore, the transformation from a continuous into a discrete time model determines system-parameters like sampling frequencies and bandwidth. For this reason, this transformation is not really an alternative representation of identical behavior. It can be seen as a part of the design process which is necessary to implement a continuous-time function in digital. For this reason, these transformations are not done in the front-end of KANDIS, but rather during system-level design later.

Re-writing rule 1 (DESS to DTS) *Nodes with the activation rule a_{DESS} and a known bandwidth $f_{max} - f_{min}$ can be replaced by a node with the activation rule a_{DTS} and $f_{clk} > 2 * (f_{max} - f_{min})$. The new function f_{new} must ensure that $f_{new}(n * t_{clk}) = f_{old}(n * t_{clk}) \forall n \in N$*

Aside from the well-known s- to z-transforms and the Shannon theorem there are further methods that

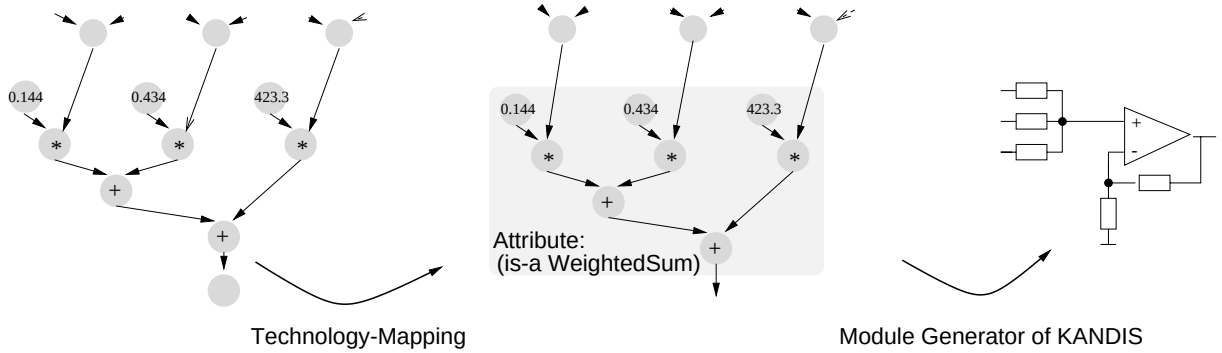

Figure 6: Technology-mapping onto blocks, for which module-generators are provided.

allow us to describe continuous-time functions in a discrete-event way. This is possible if the continuous-time function does not have an internal state which requires continuous activation. In this case, the output is only dependent from the input, and each change of an input leads to a direct change of the output in both time models.

Re-writing rule 2 (DESS to DEVS) *Nodes with the activation rule a_{DESS}, which have no internal state (integration, differentiation have an internal state, for example), whose in-edges are coming from nodes with a discrete time model (a_{DEVS} oder a_{DTS}) or whose value-range is discrete (INTEGER, ENUMERATION) can be replaced by a node with the activation rule a_{DEVS}, identical function f and $E_a = E_{in}$.*

Re-writing rule 3 (DEVS to DESS) *Nodes with the activation rule a_{DEVS} and $E_{in} = E_a$ can be replaced by a node with the same function f and the activation rule a_{DESS}, if they are in a graph with the activation rule a_{DESS}.*

In the front-end of KANDIS, rule 3 is applied when comparisons modeled algorithmically (a_{DEVS}) are dependent only on nodes modeled functionally (a_{DESS}), whose output is not used by other nodes with the activation rule a_{DEVS}. In this case, applying rule 3 on the KIR-Graph replaces an A/D converter with a comparator in the implementation.

Rule 2 can be applied in an interactive way to build up larger partitions with the activation rule a_{DEVS} that can be synthesized by high-level synthesis. At the moment we have no strategy which would allow us to determine if an application of rule 2 results in a more efficient implementation.

5 Technology mapping

The rules described above have removed the structure of a block-diagram separating different time-models. Further, some optimizations strategies have been applied. These optimizations were technology-independent and did not change the behavior specified. Now, it would be possible to find an implementation of the KIR-Graph with KANDIS:

Partitions with the activation rules a_{DTS} and a_{DEVS} can be synthesized with the high-level synthesis of KANDIS. Nodes with a_{DESS} can be implemented with the module generators for filters and non-linear functions. The back-end KANDIS allows us to find analog, SC or sampled digital implementations for these nodes (rule 3). Where necessary, converters are introduced.

However, such a technology-mapping would not be efficient. Often, analog computing devices can calculate more than only an elementary function of a KIR-Node:

- Many additions and multiplications with constants can be mapped onto a single adder.
- Integrators and differentiators allow us also to multiply the signal with a constant value.
- Inverting summers can also integrate.

In figure 6 an example for the technology-mapping of a set of addition- and multiplication-nodes onto a weighted adder is given. As many nodes as possible are clustered to one graph-node, which gets an attribute "is-a". This attribute denotes the name of the module-generator of KANDIS which has to be used for this module. KANDIS has module-generators for filters (analog and digital with rule 1), linear and non-linear operations (also analog and digital with rule 1) and a simple High-Level Synthesis tool for mapping

partitions with the activation rule a_{DEVS} onto digital hardware.

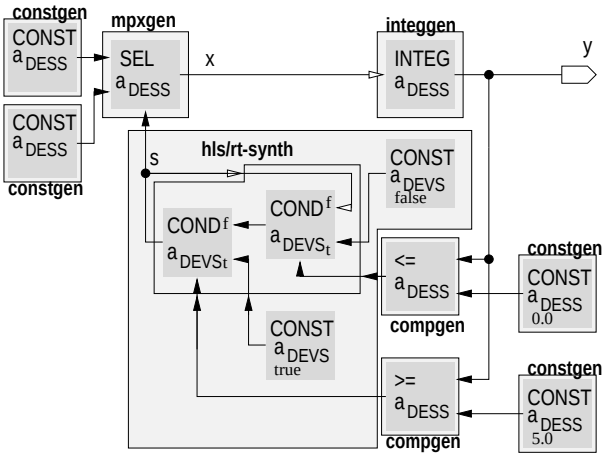

Figure 7: Signal generator after repartitioning and technology-mapping.

Figure 4 shows how the intenionally poor signal generator description from figure 3, will look as a hierarchically flat KIR-Graph. After application of rule 3, the comparisons of y with 5.0 and 0.0 are taken out of the algorithmic description. Technology-mapping results in a KIR-graph as shown in figure 7. In figure 8, the structure of the signal generator on OpAmp-Level is shown. This implementation is more efficient than the one we would have built without re-partitioning, because no A/D converter is required. The only modules required are two comparators, an implementation of a very simple automata, a multiplexer and an integrator.

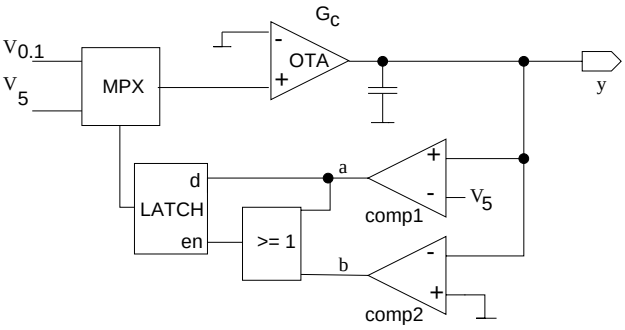

Figure 8: Signal generator on OpAmp-Level.

6 Summary and Future Work

The specification of mixed-signal systems introduces a structure into the specification of hybrid systems. If this structure is not repartitioned, the implementation may be inefficient. In the front-end of KANDIS, a specification of a hybrid system is mapped systematically onto a set of synthesizable or pre-designed modules in three steps:

Translation to KIR and unstructuring: First, the specification given in a VHDL-AMS subset or as a block-diagram is translated into a KIR-Graph by a compiler. The structure introduced by separating different modeling methodologies can be omitted. This results in a hierarchically flat graph without a block-structure.

Technology-independent optimizations: Rewriting-techniques are applied on the KIR-Graph as described in section 3 (Common subexpression elimination, constant folding). The time model of nodes near potential A/D converters is changed, so that as many converters as possible can be saved.

Technology-mapping: Finally, the hierarchically flat KIR-Graph is mapped onto a set of modules, for which generators in KANDIS are provided. The algorithm tries to put as many KIR-Nodes as possible into one module (figure 6).

The front-end described in this paper should not be seen as a synthesis-tool which generates mixed-signal hardware automatically. It is just a first approach to support the systematic top-down design and A/D-partitioning of mixed-signal systems. The design and repartitioning still requires interaction with the user. The module generators also should be used more for estimation purposes, since an optimized analog design can only be done by hand at the moment.

One application is design-space exploration by comparing different partitionings. Here, KANDIS may help to save some design-cycles by providing early estimations. Currently, we are implementing strategies and algorithms to automatically choose the transformations that lead to more efficient implementations.

Another application is the rapid prototyping of embedded systems (see also `http://www.ti.informatik.uni-frankfurt.de/Papers/RapidPro/abstract`). We are going to add a back-end which maps the output of KANDIS on FPGA and FPAA (Field Programmable Analogue Array) from Motorola. It will then be possible to generate a prototype of a mixed-signal system from its VHDL-AMS model.

References

[1] B. P. Zeigler, *Theory of Modelling and Simulation.* New York, Chichester, Brisbane, Toronto: John Wiley, 1976.

[2] S. Donnay, K. Swings, G. Gielen, W. Sansen, W. Kruiskamp, and D. Leenaerts, "A Methodology for Analog Design Automation in Mixed-Signal ASICs," in *The European Design Automation Conference (EURO-DAC)*, (Paris, France), pp. 530–534, Feb. 1994.

[3] H. Chang, A. Sangiovanni-Vincentelli, F. Blarin, and E. C. et al., " A Top-Down, Constraint-Driven Design-Methodology for Analog Integrated Circuits," *Proceedings of the Custom Integrated Circuit Conference*, pp. 841–846, May 1992.

[4] P. Oehler, C. Grimm, and K. Waldschmidt, "KANDIS - A Tool for Construction of Mixed Analog/Digital Systems," in *European Design Automation Conference*, (Brighton, UK), Sept. 1995.

[5] C. Grimm, P. Oehler, I. Kanakis, and K. Waldschmidt, "KANDIS - A Tool for System-Level Specification and Design of Mixed-Signal Systems," in *IEEE/VIUF International Workshop on Behavioural Modeling and Simulation*, (Washington DC, USA), Oct. 1997.

[6] C. Grimm and K. Waldschmidt, "KIR - A graph-based model for description of mixed analog/digital systems," in *European Design Automation Conference*, (Geneva, Switzerland), Sept. 1996.

[7] H. Praehofer and B. P. Zeigler, "Modelling and Simulation of Non-Homogeneous Models," in *Computer Aided Systems Theory — EUROCAST'89 (Serie: Lecture Notes in Computer Science, Vol. 410)* (F. Pichler and R. Moreno-Diaz, eds.), (Las Palmas, Spain), pp. 200–211, Springer-Verlag, February 26 — March 4 1989.

[8] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers - Principles, Techniques, Tools.* Addison-Wesley, 1985.

[9] P. Oehler and K. Waldschmidt, "A Knowledge-Based System-Level Construction Methodology," in *The Second World Conference on Integrated Design & Process Technology*, (Austin, Texas, USA), Dec. 1996.

[10] C. Huijs, "A Graph Rewriting Approach for Transformational Design of Digital Systems," in *EUROMICRO'96*, (Prague), pp. 177–184, Sept. 1996.

[11] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication.* Illinois: University of Illinois Press, 1949.

[12] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, 1979.