KANDIS - A Tool for Construction of Mixed Analog/Digital Systems*

Peter Oehler, Christoph Grimm, Klaus Waldschmidt

Professur fuer Technische Informatik
Johann Wolfgang Goethe-Universitaet
D-60054 Frankfurt am Main, Robert-Mayer-Strasse 11-15

Abstract
The synthesis of electronic circuits on system level offers the possibility to find better locations of the A/D interfaces and to determine parameters like clock rates and bit widths. KANDIS, a tool for (K)construction of mixed A/ANa/analogue/Digital Systems, uses a structure-oriented expert system for these tasks. The main idea is the successive construction on functional and algorithmic blocks with different kinds of tools and the permanent exchange of resource limitations with constraint propagation.

1 Introduction
Approaches to the synthesis of electronic systems are usually separated into the analog domain, where efforts are limited to the design of modules (operational amplifiers, for example), and into the digital domain, the high-level synthesis. System-level abstraction offers additional degrees of freedom and divide the synthesis into three different tasks:

1. The partitioning of mixed analog/digital systems: The search for efficient locations of A/D boundaries, considering the fact that some components can be realized either in analog or digital (filters and mathematical operators, for example).

2. The determination of the parameters of individual components, such as sampling rate, bit widths and the reference voltages of sampling systems.

3. The translation of behavioral descriptions into a structure (to select the kind of A/D and D/A converters to be used for a concrete realization, for example).

Task 3 is necessary because the structure (of a chosen realization) has a great influence on quantities like size, performance and power consumption, which are needed for system evaluation.

For the handling of these tasks our approach uses a knowledge-based expert system for construction tasks ([4]). We have to distinguish between construction and synthesis. Per definition, synthesis means the translation from behavior to structure. In this context, construction shall be taken for working user guided, and synthesis means working in an automated manner.

Figure 1: Overview of KANDIS

Figure 1 shows an overview of KANDIS, a knowledge-based construction tool for mixed analog/digital systems. This paper begins with some aspects on modeling of mixed analog/digital systems on a high level of abstraction. The purpose heterogeneous model cannot be expressed within VHDL. That’s the reason why we use VHDL-hybrid, an extented VHDL, for system specification. In the remainder of this paper functional (algorithmical) block means a block described by a function (an algorithm). VHDL-hybrid is the input language for KANDIS which will be compiled into an intermediate representation (KIR, section 3). In section 4 the concept of KANDIS is presented which is based on PLAKON, a knowledge-based expert system for construction tasks ([3]).

*This work is supported by DFG under WA 357/9.
construction tool uses different approaches for functional and for algorithmical blocks. Section 5 describes the interaction between these both approaches, whereas section 6 and 7 show the principles of both approaches. The first one works on PLAKON, the second one uses high-level synthesis tools.

2 Modeling Aspects

Because of the different time models, a homogeneous model of a heterogeneous system usually needs the translation of the continuous-time domain into the discrete-time domain, using mainly numerical integration methods. For simulation purposes it seems to be more accurate to use a mixed-mode simulator. Such heterogeneous simulators use different modeling formalisms: discrete event system (DEVS), discrete time system (DTS) and differential equation specified system (DESS) ([5]). The reason to prefer a mixed-mode simulator is that the translation of continuous-time systems to a discrete-time model can only use simple forms of numerical integration, pulse invariant simulation or backward Euler, for example. The methods for numerical integration in modern continuous-time simulators are much more sophisticated.

For synthesis purposes we need the possibility to specify components in a short and meaningful way. Continuous-time filters can be specified by transfer functions. The synthesis tool has to choose a filter structure and to compute the values of the used components taking the coefficients respectively the poles and zeroes of the transfer function, for example. A discrete-time model of a continuous-time filter is not suited for that purpose. Instead of this, a heterogeneous synthesis tool needs a common model for parameters like area, power consumption, delays and accuracy.

Thus, an inhomogeneous modeling seems more suitable for synthesis respectively construction of heterogeneous systems (multi-paradigmatic approach, [6]). A block diagram can be used as a model of the system at the highest level (directed signal-flow). The individual blocks can be modeled by domain-specific behavioral descriptions, such as transfer functions in s and z domain, nonlinear functions, or in an algorithmic language.

3 System specification with an extended VHDL

VHDL permits modeling and simulation of DTS and DEVS. VHDL-A ([1]) allows in addition modeling and simulation of DESS using the equation-environment and dlt/integ-functions.

We want to use the following definitions of specification and modeling ([7]):

*Specification* means a description of an intended system using modeling techniques. Therefore, behavior has not to be determined completely. *Modeling* is the task of describing an (at least in mind) existing system. All behavior is determined\(^1\). Therefore, models can be simulated easily. They can also be used for synthesis, because they describe an existing system. Therefore, a solution (at least the simulator) exists. Specifications containing nondeterminism can be used for finding more efficient implementations.

3.1 Differences to VHDL

VHDL-hybrid has been defined for behavioral specification of hybrid systems using the well-known syntax of VHDL. The semantic ("modeling-style") of VHDL-hybrid in contrast to VHDL reflects the difference between simulation and synthesis. The specification does not model value representation (analog/digital, accuracy), time - domain (continuous/discrete/logic) or an implementation, although each of them can be specified by attributes.

`concurrent statements` ([2]) are used to specify a netlist of functional blocks. These can be specified by component instantiations or by operators. Concurrency means concurrent in continuous, physical time. Therefore, we do not need an additional environment for differential equations as in VHDL-A, where two types of simulation are provided.

Processes with activation on events of signals are used to specify DEVS as in VHDL. Processes with `wait for time` - statement (used by many HLS-Systems to specify time-constraints) are used to specify repeatedly activated blocks (e. g. digital filters).

`REAL` - numbers are used to specify the most abstract signal. Its implementation can be analog or digital with undetermined accuracy. `INTEGER` - numbers specify digital signals or variables with undetermined accuracy. `BIT` - Vectors and enumerations specify digital signals or variables used for control - purposes.

The implementation (analog, digital or with switched capacitors) of blocks/operators in concurrent statements is undetermined.

Value representation, time-model and implementations, which are by default undetermined can be constrained by attributes. An initial partitioning can be given by inserting converter-blocks (A/D, D/A, ...).

\(^1\)Undefined Values in VHDL are treated as a known value "U". Therefore, you can write "IF a='U' THEN ...", which can be simulated, but makes no sense when you are going to synthesize it.
In order to allow specification of analog circuits as nonconservative systems, VHDL-hybrid provides features to express integration and differentiation of a signal over time. They are defined as functions \texttt{ddt} and \texttt{integ} in the package \texttt{hybrid}. They can be used only in \texttt{concurrent\_statements}. To permit specification in the frequency-domain, the components \texttt{s\_polynomial}, \texttt{z\_polynomial}, \texttt{s\_product}, \texttt{z\_product}, \texttt{s\_partial} and \texttt{z\_partial} are defined in the same package.

### 3.2 The Frontend of KANDIS

The Frontend translates VHDL-hybrid into an intermediate representation (KIR). KIR is a graph of edges and nodes. Each node can contain a subgraph with its own semantic - Data-Flow Graph, Control-Flow Graph, Netlist, Signal-Flow Graph - defined by a time model and an activation model.

- \texttt{sequential\_statements} with activation on events are translated into a graph with activation on events (Control- and Data-Flow Graphs), usually implemented in discrete time.
- \texttt{sequential\_statements} with activation after a delay \(i\) are translated into a graph with permanent activation in discrete time (digital signal-flow graph).
- \texttt{concurrent\_statements} become a graph which is always active in continuous time. The underlying semantic is a netlist of functional blocks (figure 2).

As example, we describe a circuit for compressing the dynamic-range of signals:

```vhdl
USE work.hybrid.all;
ENTITY comp IS
  PORT(u1: IN REAL; u2: OUT REAL);
END ENTITY comp;
ARCHITECTURE multiparadigmatic OF comp IS
  SIGNAL u3, u4: REAL;
  ATTRIBUTE fs_min OF hp1: LABEL IS 44.1 kHz;
  ATTRIBUTE fs_min OF alg: LABEL IS 44.1 kHz;
  ATTRIBUTE fs_min OF div: LABEL IS 44.1 kHz;
BEGIN
  -- frequency-domain: \(H(s) = \frac{0+s}{1+s}\)
  hp1: s\_polynomial
    GENERIC MAP((0.0, 1.0), (1.0, 1.0));
    PORT MAP(u1, u3);
  -- Algorithmic description
  alg: PROCESS(u3)
    VARIABLE max: REAL;
    BEGIN
      IF abs(u3) < 0.25 THEN
        max := 0.25;
      ELSE
        max := abs(u3);
      END IF;
      u4 <= 0.01*max*0.99+u3;
      z := max;
    END PROCESS;
  -- Continuous-time
  div: u2 <= u1 / u4;
END multiparadigmatic;
```

The VHDL-hybrid frontend translates the VHDL-hybrid description into KIR. All statements in the process \texttt{alg} are translated into a CDFG. The filter \texttt{hp1} and the divider \texttt{div1} are instances, whose implementation can be either digital or analog.
Depending on the semantic of a node, its subnodes are passed to different tools: nodes, whose subgraphs can be interpreted as netlists are passed to the expert-system (section 4), where they are constructed (section 6) or transformed into discrete-time graphs (digital SFG, CDFG).

CDFG and digital SFG are passed to the high-level synthesis and estimation tool (section 7).

4 Concept of KANDIS

Electronic systems of high complexity are normally designed using a hierarchy with several levels of abstraction (composition). During the design of electronic systems, alternatives exist on different levels relative to various possible realizations (specialization). Both together can be well described in a conception-hierarchy like the one defined in PLAKON ([3]): on a conceptual hierarchy (isa relations) lies a compositional hierarchy (has-parts relations). Moreover, attributes of concepts are inherited to their specializations. In the remainder of this paper, the expression conception-hierarchy always means the combination of a conceptual hierarchy with a compositional hierarchy. Figure 3 shows a detail of a conception-hierarchy of an analog/digital converter, for example.

Because a large solution space exists where search techniques as well as heuristics can be used and the fact that the knowledge of electronic systems is well representable within a conception-hierarchy lead to the conclusion that a knowledge-based expert system for construction tasks is well suited for this construction tool.

The main structure of a construction process is as follows. The compiler generates for each block one instance. The underlying conception-hierarchies are taken from a library. The functions which express the behavior of the functional blocks are stored as attributes of the corresponding instances. The behavior of the algorithmic blocks is represented in data-flow graphs which are associated with the corresponding instances.

The construction starts with a set of instances which represent a determined location of A/D interfaces. The following construction steps are possible:

- to split objects, that means to instantiate sub components (top-down, using has-parts-relations);
- to integrate components (bottom-up);
- to specialize objects (using is-a-relations);
- to parameterize objects, that means to determine attributes of objects.

The structure of a system at block diagram level is expressed by the insertion of concepts which represent input and output signals and input/output relationships between them and block concepts. Slots of block concepts hold attributes like area, power consumption and clock rate, which will be determined during the construction process. Slots of signal concepts hold attributes like bit width (accuracy), data rate and signal type.
Constraint nets are used for the representation and evaluation of dependencies between construction objects or their attributes. They are expressed as conceptual constraints on the conception-hierarchy. Not only instances are generated during the construction process but also the corresponding constraint nets. Constraint propagation is done for consistency checks and for determination respectively reducing of attribute values.

After the completion of a construction which is valid for one determined location of A/D interfaces, the possibility exists to move blocks from one domain into another. Domains are digital (discrete-time, discrete-value), analog (continuous-time, continuous-value) and a realization with switched capacitors (discrete-time, continuous-value). The construction process will rerun with these new locations of A/D interfaces.

The handling of the data-flow graphs needs high-level synthesis tools different from the knowledge-based construction process. These tools are described in section 7. The principles of interaction between PLAKON and high-level synthesis tools are shown in the next section.

5 Interaction between the expert system and the high-level synthesis tool

Resources used by a system are limited or shall be optimized during the design process. Such resources are size of a realization (area), power consumption, delays of signals and signal noise (distortion) ratios. Bit widths and clock and sampling frequencies are resources, too. Resource limitations are handled in PLAKON with constraint nets. For the delay constraints we use a simple timing model on system level. A noise analysis investigates the influence of inaccuracies at different points in the system to the system outputs (accuracy constraints). The attributes of functional blocks are determined resp. reduced by computing functions. Attributes of algorithmic blocks are located inside the conception-hierarchy but get their values stepwise by high-level synthesis tools which work on the associated data-flow graphs and result in design estimations (figure 4).

For that functional blocks which are inside the digital domain (caused by the actual partitioning in analog and digital domains) KIR-graphs are automatically generated during the construction process. Such a KIR-graph represents the behavior of a digital realization of the underlying function \( H(z) \) or nonlinear and is handled by the high-level synthesis tool like an algorithmic block.

6 Construction of functional blocks

The construction of functional blocks uses the concept of PLAKON. Conception-hierarchies exist for A/D and D/A converters, linear and nonlinear functions, and filters. Because of the many possibilities of filter realizations, only some basic methodologies are implemented, like simple \( s \) to \( z \) transformations, cascade approach only with some first and second order blocks, simple passive network generation, simulation methods (en bloc approach) and some common structures for digital filters.

The representation of filter structures are done with signal-flow graphs. Such graphs use only linear operations: addition of signals, multiplication with constants and \( s^{-1} \) operation (integration) for continuous-time filters respectively \( z^{-1} \) operation (register) for discrete-time filters. The signal-flow graph of a digital filter will be translated into a data-flow graph, as mentioned in the last section.

The refinement is done top-down to basic blocks, like operational amplifiers, capacitors, registers, etc. Estimations for area, power, delays and inaccuracies (noise of the operational amplifiers, for example) for these basic blocks are taken from a technology map. These estimations will be propagated bottom-up by the constraint net.

These construction methods don’t generate structures which will be accurate enough for implementation. The reason is that analog design is much more complicated. But these methods are sufficient to generate design parameters for estimation purposes.

7 Synthesis of digital blocks

As high-level synthesis (HLS) is part of system-level synthesis of heterogenous systems, it must provide interfaces to allow communication between HLS and coordinating strategy in PLAKON on system level. As
the coordinating strategy knows only about resource consumption, but not about implementation details, it is sufficient to provide an interface for estimations and constraints. Estimations must be very fast, because system-level synthesis will try to evaluate a lot of different configurations with different accuracy to find a good solution on system-level.

With system-level synthesis choosing many different blocks (digital filters, algorithmic blocks, ...) to be implemented digitally, a flexible target architecture is needed. The design-space exploration in HLS should be done automatically, because the coordinating strategy may try different solutions, and the user may not want to optimize them all by hand.

The input of HLS, which can be several KIR-nodes with the underlying semantics of a DFG, CFG or digital SFG is distributed onto a number of FSMD or - if a fast digital filter is needed - implemented as pure hardware. For this task, we are implementing estimators for design-curves. Nonlinear optimization is used for initial partitioning of digital subsystems. After initial partitioning, the accuracy of the digital system is determined, the graphs are mapped onto a set of FSMD or specialized datapath elements, e.g. shift-registers for digital filter implementation, and their constraints are set.

Then, a clock frequency is determined which will not lead to conflicts with the delay-constraints and is a multiple of the sampling-frequencies used. A library of components in discrete time is generated from concepts in continuous time, optimized for the chosen clock. At last, scheduling is performed.

Each node in behavioural representation in KIR has a reference to a hierarchy of possible instances which implement its behaviour. Each implementation has estimators for area, delay and power.

As the partially implemented HLS is based on design-specialization instead on design-transformation (figure 5), upper and lower bound estimations can be calculated even for incomplete constructions. HLS is not done in "one step". The refinement of ranges and attributes is done successively. This allows parallel construction of analog and digital subsystems.

Implementation of HLS is subject of current work.

8 Conclusions

This paper has presented KANDIS, a knowledge-based construction tool for the design of mixed analog/digital systems. The implementation is still in progress. Our current work is to build up the libraries for the functional blocks and to implement the procedures for the high-level synthesis tools. A first version of KANDIS works purely interactive. Our future work will be the development of strategies for the construction with PLAKON and strategies for the optimization of the partitioning into analog and digital domains.

References


