KIR – A graph-based model for description of mixed analog/digital systems*

Christoph Grimm, Klaus Waldschmidt
EMail: {grimm|waldsch}@ti.informatik.uni-frankfurt.de
Technische Informatik, Professur Prof. Dr. K. Waldschmidt
Johann Wolfgang Goethe - University, Frankfurt am Main

Abstract

Systems can be described in different time models and on various levels of abstraction. We can distinguish between models in discrete-event, discrete and continuous time.

Graph-based, formal models allow us to use either discrete event/discrete time models or continuous time models. The combined use of all three time models in one system is the main problem when modeling mixed analog/digital systems.

In this paper, a graph-based model is presented that supports the use of all three time models in different parts of a graph. This allows digital, discrete-time systems to be modeled together with their analog, physical environment.

1. Introduction

For computer aided design of electronic systems, a formal specification of the functions and features is necessary. We can distinguish between:

- text-based, formal specification using modeling languages, such as VHDL[1];
- graph-based, formal specification using graphs, such as petri-nets [5] or data-flow graphs [8], [13].

Mixed analog/digital systems consist of sections modeled in three different time models [12] (see figure 1). Sections that work in the continuous-time domain are specified using differential equations (Differential Equation Specified Systems, DESS). Sections working in the discrete-time domain (Discrete Time Systems, DTS) are described using automata models or difference equations. Discrete-Event Systems (DEVS) can be described by petri-nets or data-flow graphs.

*This work is supported by the DFG, grant no. WA 357/9.

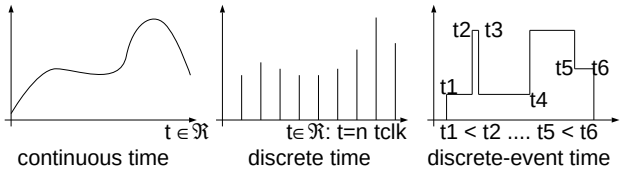

Figure 1. Time models of hybrid systems

Text-based modeling such as VHDL-A [3] permits the modeling of some sections of a system in discrete/discrete-event time and the modeling of other sections of the same system in continuous time.

However, design automation algorithms don't work on text-based models. The formulation of algorithms and methods requires graph-based models. In order to get a homogeneous, graph-based representation of a hybrid system, Brielmann and Kleinjohann[6] translate continuous-time systems into discrete-time systems using the z-transform. The resulting discrete-time system can be represented by petri-nets (ETPN). This results in a lower precision in simulation than direct simulation of DESS. Furthermore, the application of known methods of synthesis for the analog parts of that homogeneous model seems to be an unsolved problem.

Hybrid automata can be viewed as a generalization of timed automata, in which the behavior of variables is governed in each state by a set of differential equations [4]. They allow modeling of interactions of physical environment with reactive programs. Hybrid automata represent reactive behavior by a state transition graph. The differential equations of each state are represented by their text-based representation.

None of the known representations allows the common graph-based modeling of differential equation specified systems together with discrete-time or discrete-event systems [9]. The aim of this paper is to propose such a graph-based model of hybrid systems.

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00

In section 2, we describe the structure and semantics of the KIR-graph (KIR = **K**andis **I**ntermediate **R**epresentation). How this graph-based model can be used to specify or model behavior of mixed analog/digital systems is shown in section 3. Existing and potential applications are described in section 4.

2. The KIR - Graph

Petri-nets and data-flow graphs do not allow the representation of continuous-time functions. This is due to their event-driven semantics. This semantic does not permit the representation of continuous-time signals that are always valid. These signals are needed to calculate for example differentiation and integration of signals over time. The event-driven semantics of these graphs is determined by their "firing rule". This rule defines when the tokens on the in-edges of the nodes are consumed and when new tokens on the out-edges are produced.

Modeling of continuous-time systems is done using signal-flow graphs or block diagrams. These can represent differential equations in a graph-based form. They do not have "firing rules" that are needed to describe reactive, event-driven behavior. Therefore, they cannot be used for modeling of reactive, event-driven behavior.

The problem using existing graphs is that event-driven systems need "firing rules" in some form, but continuous-time systems must be always active. When specifying in a homogeneous time model, only the operation of a node is described (e. g. "multiply operands"). The condition that leads to the execution of an operation is described by the semantics of the graph. It is not a part of the specification given by the "user". However, the existence of different time models in a mixed analog/digital system requires different "firing rules".

The KIR-graph tries to avoid these problems using the following ideas:

- In addition to the operation f, the condition that leads to the execution of f has to be specified. This means, that the "firing rule" is not a constant feature for the whole graph. It can be different for each node.
- The edges of the graph represent signals $s(t)$. They are a function of time in the corresponding time model of the graph or node.

Definition 1 (KIR-Edge) *Edges represent directed signals that connect nodes. An edge e is a tuple consisting of a reference (this means i. e. only its name) to one origin node $v_{o,r}$, a set of references to destination nodes $V_{d,r}$ and a signal $s(t)$:*

$$e = (v_{o,r}, V_{d,r}, s(t))$$

$s(t)$ remains constant, until v_o assigns a new value to it.

The signal $s(t)$ can be of three different types: REAL, INT or ENUM. The type REAL is used to model continuous-value signals. The type INT is used to model discrete-value approximation of continuous-value signals. The type ENUM allows to model discrete-value signals.

Definition 2 (KIR-Node) *Nodes represent operations performed on signals. A node v is a tuple consisting of a set of references to edges $E_{in,r}$, a set of references to edges $E_{out,r}$, a set $E_{a,r} \subseteq E_{in,r}$ of references to in-edges, that can activate the firing rule, the operation f and a firing rule a (see definition 4).*

$$v = (E_{in,r}, E_{out,r}, E_{a,r}, f, a)$$

The operation f of a node can be specified in the following ways:

- In a declarative way by giving the abstract function f of all output-signals $S_{e_{out}}(t)$ of the input-signals $S_{e_{in}}(t)$.
- In an operational way by describing a method or a structure. The method or the structure are described by a graph consisting of KIR-nodes and -edges (see definition 3).

Definition 3 (KIR-Graph) *A KIR-graph G is a KIR-node, whose function f is described in an operational way by a graph (V, E) of KIR-nodes V and KIR-edges E.*

$$G = (E_{in,r}, E_{out,r}, E_{a,r}, (V, E), a)$$

Definition 4 (Firing rule of KIR-nodes) *A "firing rule" a determines when the operation f of a node has to be executed. The firing rule consists in two conditions c_1, c_2:*

$c_1 : T \rightarrow \{true, false\}$ *assigns each time t from the set of all physical time T one of the values $true$ or $false$.*

$c_2(s_1(t), \ldots, s_n(t))$
is a function of the signals $s_1(t), \ldots, s_n(t)$ from the edges $E_{a,r}$ (a subset of the in-edges of a node, see definition 2), with the value-range $\{true, false\}$.

If both conditions are $true$, the operation f of the node is performed.

Definition 3 allows hierarchical ordering of graphs. To ensure that no node of a graph is firing while the graph itself is not firing, we restrict the use of firing rules as follows:

Rule 1 (Allowed firing rules) *For a firing rule a of a KIR-graph and the firing rules a_i of all its sub-nodes/graphs $v_i \in V$:*

$$a = true \quad \Leftarrow \quad \exists a_i \text{ with: } a_i = true$$

Infinitesimal small changes of continuous value signals could always activate a node sensitive on an edge. To avoid this:

Rule 2 (Type of $E_{a,r}$) *The type of the signals of $E_{a,r}$ of a node must not be REAL*

To give the hierarchical graph a clear structure, we restrict the access of edges and nodes to those which are locally defined or imported als in- or out-edges.

Rule 3 (Visibility of edges)
The in(out)-edges $E_{in,v}$, $(E_{out,v})$ of a node v of a graph G are in(out)-edges $E_{in,G}, (E_{out,G})$ or edges E_G of G:

$$(E_{in,v} \cup E_{out,v}) \subseteq (E_G \cup E_{in,G} \cup E_{out,G}) \quad \forall v \in G$$
$$(v_{o,r} \cup V_{d,r}) \subseteq V_G \qquad \forall e \in G$$

3. Modeling hybrid systems with KIR

In the previous section, we have described the structure of the KIR-graph. To represent behavior of hybrid systems, we need a semantic. The semantic is described by firing rules a. To describe these firing rules, an underlying time model is required. To simplify the graph-based representation of VHDL- or VHDL-A descriptions, we use a similar time model. The time is a pair of a value of physical time and a value of a causal time model:

$$t = (t_{phys} \in \Re, t_\delta \in N)$$

We define the relation "$<$" as follows:

$$t_1 < t_2 \quad \Leftrightarrow \quad t_{1,phys} < t_{2,phys}$$
$$\text{or} \quad t_{1,\delta} < t_{2,\delta} \wedge t_{1,phys} = t_{2,phys}$$

This leads to two orthogonal ranges of time (figure 2).

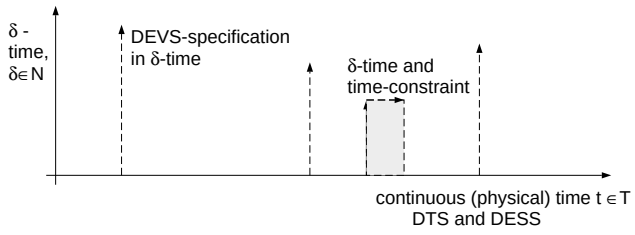

Figure 2. Time-model of the KIR-graph

3.1. Modeling in the discrete-event time model

In hybrid systems, parts of the system are described in an algorithmic way. An algorithm can be described by a partial order of a set of operations to be performed. This partial order can be described for example by a data-flow graph. In the context of a system, it is important, *when* the algorithm is executed. Nodes of a data-flow graph are active when the values of all in-edges are known. These semantics could not be used in KIR, as the edges are signals, whose values are always valid and known. The semantics of VHDL use an event-driven method to activate a process. In KIR, we use a similar method to describe activation of a node: A node with $a = a_{DEVS}$ is activated/fires, when the value of a signal $s_{e_a}(t)$ of an edge in E_a has changed at time t_1.

1. The behavior of discrete-event systems can be modeled using a KIR-graph with the following conditions c_1, c_2 in its firing rule a_{DEVS}:

$$c_1 = true \qquad \forall t \in \Re$$
$$c_2 = true \qquad \Leftarrow s_{e_a}(t_1) \neq s_{e_a}(t_2), t_1 < t_2$$

2. To satisfy rule 1, all nodes in a graph-node with $a = a_{DEVS}$ must also have the firing rule a_{DEVS}.

The functions $cond(G_{if}, G_{then}, G_{else})$ and $iterate(G_{cond}, G_{iter})$ can be used for modeling conditions and iterations. This is similar to the use of branch and merge-nodes in data-flow graphs. These functions can only be used inside nodes with $a = a_{DEVS}$.

3.2. Modeling in discrete time

Often, functions are evaluated independently from external events. Instead of waiting for external events, these functions are executed in regular time steps t_{clk}.

1. For describing the regularly repeated execution of a function, the conditions c_1, c_2 of a_{DTS} are defined as follows:

$$c_1 = true \qquad \forall t = n * t_{clk}, n \in N, t_{clk} \in \Re$$
$$c_2 = true$$

2. The condition of rule 1 is true, if for all nodes $v_i \in V$ $a_i = a_{DEVS}$ or $a_i = a_{DTS}$ with $t_{clk,a_i} = n * t_{clk}, n \in N$.

If all operations are linear, the KIR-graph corresponds to a discrete-time signal-flow graph, where all edges E have the semantics of a multiplication with z^{-1} (delay of one time step).

As an example, in figure 3 the representation of the direct form I of $H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}}$ by a KIR-graph is given.

The graph with $a = a_{DTS}$ models behavior in the discrete time model. The subgraph with $a = a_{DEVS}$ describes the calculation of new values in the discrete-event time model.

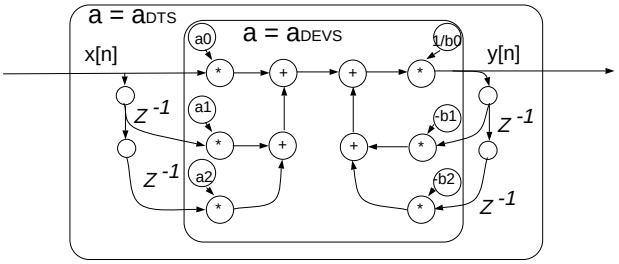

Figure 3. KIR-graph with a_{DTS}

3.3. Modeling in continuous time

Behavior in continuous time can be described by differential equations or transfer functions. Graph-based models for differential equations include signal-flow graphs or block diagrams. To describe behavior in continuous time, a_{DESS} is defined as follows:

1. For describing continuous-time behavior in a graph-based model, the semantics of a block diagram can be used. Therefore, we define the conditions c_1, c_2 of a_{DESS}:

   $$c_1 = true \qquad \forall t \in \Re$$
   $$c_2 = true$$

2. As $a_{DESS} = true \quad \forall t \in \Re$, rule 1 allows the use of all a for all nodes of the graph.

In figure 4 a differential equation is represented by a continuous-time signal-flow graph, a block-diagram and a KIR-graph with $a = a_{DESS}$.

3.4. Modeling hybrid systems

A system is more than only the sum of its parts. Therefore, it is not possible to use signal-flow graphs, data-flow graphs and block-diagrams to describe behavior of a hybrid system. These graphs have a well defined semantics describing their behavior, but the interaction of the graphs of different semantics is not defined.

Hybrid systems make use of all three time models. Behavior in all three time models can be described by one KIR-graph with nodes (subgraphs) with different firing rules. As hybrid systems are modeled in *all* three time-models, the KIR-graph representing system-level behavior can only be

Firing rule of graph-node	firing rule of nodes
continuous-time (a_{DESS})	$a_{DEVS}, a_{DTS}, a_{DESS}$
discrete-time (a_{DTS})	a_{DEVS}, a_{DTS}
discrete-event (a_{DEVS})	a_{DEVS}

Table 1. Allowed firing rules

a KIR-graph with $a = a_{DESS}$. Due to rule 1, only the continuous time-model allows us to have subgraphs with all other firing rules a! The nodes of the KIR-graph representing system-level behavior can then have the firing rules $a = a_{DEVS}$, $a = a_{DTS}$ or $a = a_{DESS}$.

Graph-nodes with discrete-time activation ($a = a_{DTS}$) cannot – due to rule 1 – contain nodes with continuous-time activation ($a = a_{DESS}$). Sub-nodes with $a = a_{DEVS}$ are possible.

Graph-nodes with discrete-event activation ($a = a_{DEVS}$) can contain only nodes with $a = a_{DEVS}$.

Table 1 gives an overview of the firing rules that are allowed in nodes of a graph with given firing rule.

As a graph can consist of nodes with different firing rules, nodes can also be activated by events on in-edges of a graph with another firing rule (see table 1). The following combinations can occur:

- DESS $\rightarrow$ DTS: A node with $a = a_{DTS}$ is part of a graph-node with $a = a_{DESS}$. In this case, there are no problems since a_{DTS} is independent from the in-edges. a_{DESS} and rule 1 ensure that the signals of the in-edges are always valid and up-to-date, when a_{DTS} activates the function of its node.

- DESS $\rightarrow$ DEVS: A node with $a = a_{DEVS}$ is part of a graph-node with $a = a_{DESS}$. In this case, the discrete-event node could be constantly activated by infinitesimally small changes in the signals of the in-edges. In this case, the node would describe a continuous-time behavior. *To avoid this, the signals of the edges E_a that are used in condition c_2 of the firing rule must be value-discrete!* Value-discrete signals can be obtained by comparing two continuous-value signals.

- DTS $\rightarrow$ DEVS: A node with $a = a_{DEVS}$ is part of a graph-node with $a = a_{DTS}$. a_{DEVS} ensures that all necessary calculations are performed at once (in δ time, without delay in physical time).

It can also occur that a graph (e. g. a DESS) uses signals on the out-edges of a node that is not always active (e. g. DTS, DEVS). In this case, the edge keeps its value while the node is inactive. In Definition 1 (KIR-edge) we have defined that

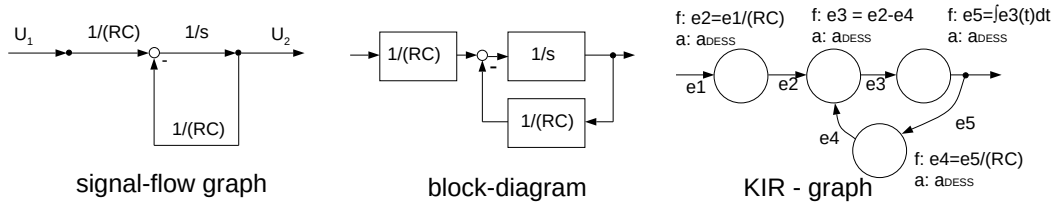

Figure 4. Representation of a transfer function by graphs.

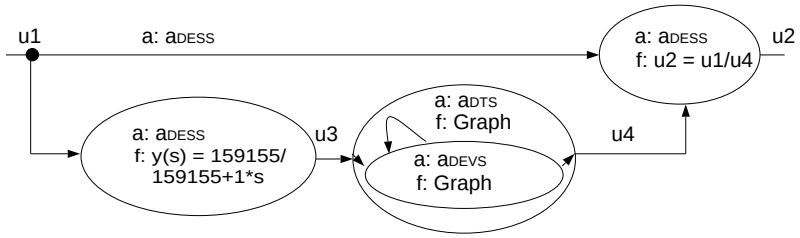

Figure 5. KIR-graph of the dynamic compressor specified in VHDL in figure 6 (simplified)

an edge keeps its value until the node becomes active again and the node assigns a new value to the signal.

The KIR-graph is a formal, behavioral model of hybrid systems on a high level of abstraction. Since all signals/edges are directed, there is no *direct* way to model networks with across- and through-values (Kirchhoff-laws). These networks would not specify a behavior or a function, but model the structure of an existing system. As the KIR-graph contains only directed, signal-flow oriented models it is a good specification of a hybrid system.

4. Application

In the project "Specification of Hybrid Systems", we have developed methods for evaluation and partitioning of hybrid systems. A system is specified in a subset of VHDL-A ("VHDL-hybrid", e. g. figure 6) on behavioral and functional level as a nonconservative system. A compiler translates this text-based specification into a KIR-graph (figure 5). In the example from figures 5 and 6, the concurrent statements hp1, alg and div are subgraphs of a graph with a_{DESS}. The process alg, which is activated every $100ns$, is translated into a subgraph with a_{DTS}. Sequential statements in a process are translated into nodes of the subgraph alg with a_{DEVS}. The differential equation represented by the transfer function in hp1 can be represented either by its abstract function or – after first steps of synthesis – by a block diagram. The resulting KIR-graph is written into a file with a syntax very similar to EDIF or ASCIS-DFG[8].

```
ENTITY comp IS
  PORT(u1: IN REAL; u2: OUT REAL);
END comp;

ARCHITECTURE multiparadigmatic OF comp IS
  SIGNAL u3: REAL;
  SIGNAL 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
  hp1: ENTITY s_polynomial (default)
          GENERIC MAP((0.0,159155.000),
                      (1.0,159155.000))
          PORT MAP(u1,u3);
  alg: PROCESS (u3)
          VARIABLE max:  REAL;
          VARIABLE z:    REAL;
       BEGIN
          IF abs(u3) < 50 THEN
             max := 50;
          ELSE
             max := abs(u3);
          END IF;
          u4 <= 0.99*u4+0.01*max;
          z := max;
          WAIT FOR 100 ns;
       END PROCESS;
  div: u2 <= u1 / u4;
END multiparadigmatic;
```

Figure 6. Dynamic compressor

There exists a hybrid implementation for each KIR-graph: DTS and DEVS can be synthesized with known methods of high-level synthesis. Transfer functions in DESS can be implemented with analog computing devices. The resulting system would be a correct implementation of the specification. Nevertheless, it is not optimal concerning area, delay and power consumption. The tool KANDIS[11]

is being developed to find a better partitioning. In this tool, the KIR-graph is used as a starting point for:

- Methods for estimation of area, power and delay.
- Transformations:
  continuous-time $\rightarrow$ discrete-time with numerical integration methods, Shannon theorem.
  discrete-time/discrete-event $\rightarrow$ continuous-time.
- Construction and synthesis of analog or digital structures[11],[10].
- Communication between analog construction and digital synthesis[7].

5. Summary

We have presented a graph-based, formal model for mixed analog/digital systems. It can serve as intermediate representation or as the basis of an information model of VHDL-A. This graph-based model allows to use formal, graph-based methods in system design or synthesis. Possible applications are:

- The simulation of hybrid systems. [12] describe the idea of abstract simulation machines. Each KIR-node could be such a simulation machine.
- The design of hybrid systems (see section 4). The formal, graph-based representation allows to formulate graph-based methods for analysis and partitioning into analog and digital parts.
- Formal verification requires a formal specification. We would have to prove the equivalence of the formal specification (KIR-graph) and a model of its implementation.

In our current work, we use KIR for documenting and proving transformations that allow us to re-partition the structure of a KIR-graph representing the behavior of a mixed analog/digital system.

References

[1] *IEEE Standard VHDL Language Reference Manual (IEEE Std 1076-1993).* 1994.

[2] *7. E.I.S.-Workshop*, Technische Universität Chemnitz, Nov. 1995.

[3] *IEEE VHDL subPAR 1076.1: Analog Extensions to VHDL. Design Objective Document (DOD). Version 2.2.* oct 1995.

[4] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems . In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems (Serie: Lecture Notes in Computer Science, Vol. 736)*, pages 209–229. Springer-Verlag, 1993.

[5] L. Benders and M. Stevens. Petri Net Modelling of Task Level Behavioral VHDL for VLSI. In *Second European Conference on VHDL Methods*, pages 180–183, Stockholm, Sept. 1991.

[6] M. Brielmann and B. Kleinjohann. A formal model for coupling computer based systems and physical systems. In *European Design Automation Conference*, pages 158–163, 1993.

[7] C. Grimm, P. Oehler, and K. Waldschmidt. Eine Zwischendarstellung zum Entwurf hybrider Systeme durch Kopplung verschiedener Konstruktionswerkzeuge (in German). In *7. E.I.S.-Workshop* [2].

[8] G. G. Jong. Data flow graphs: system specification with the most unrestricted semantics. In *The European Design Automation Conference*, pages 401–405, Amsterdam, The Netherlands, Feb. 1991.

[9] T. Leyendecker, P. Oehler, and K. Waldschmidt. *Spezifikation hybrider Systeme (in German).* Internal Report 11/95, FB Informatik, J. W. Goethe-University, 1995.

[10] P. Oehler, C. Grimm, and K. Waldschmidt. *Begriffshierarchien zur wissensbasierten Konstruktion von Filtern, mathematischen Operatoren und Wandlern (in German).* In *7. E.I.S.-Workshop* [2].

[11] P. Oehler, C. Grimm, and K. Waldschmidt. Kandis - a tool for construction of mixed analog/digital systems. In *EuroDAC*, Brighton, UK, Sept. 1995.

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

[13] U. Steinhausen and et al. System-Synthesis Using Hardware/Software Codesign. *Int. Workshop on Hardware-Software Co-Design, Cambridge, MA, Oct. 7-8*, 1993.