An Approach for Classification of Integrated Circuits by a Knowledge Conserving Library Concept

D. Wagenblaßt, W. Thronicke
Universität–GH Paderborn/ Cadlab — Physical System Design
33102 Paderborn – Germany

Abstract

A simulation based analysis of analog and mixed analog/digital circuits on printed circuit board level requires a lot of information about the involved components. Entering solely the data sheet information into a component library normally disregards structural knowledge which is well–known by the circuit designer. Therefore, the expert knowledge necessary for an optimum use of analysis tools is no more available. This paper presents an approach to overcome this problem. A concept to represent efficiently expert knowledge and classification information is introduced and a classification of circuits which conserves the structural information is presented.

1 Introduction

Physical effects like electromagnetic interference and EMC In addition to component parameters the effective analysis of complex analog and mixed analog/digital components and applications concerning physical effects like electromagnetic interference and EMC needs structural knowledge about the components. Therefore, a well structured component behaviour description is a prerequisite for software supported analysis. Typically, a component library contains only few information like device name and manufacturer, packaging, and some parameters given in a data sheet. To maintain EMC relevant component information, an extended library concept has to be set up [4]. On the one hand, as much data sheet information as possible has to be stored in the library and on the other hand, this information has to be structured in a suitable way.

The presented approach starts with the definition of a notation to represent classification knowledge in an efficient way. This can be done by the introduction of special tree structures. In the following a classification of components using this concept is described. From literature different classification methods are well–known [3]. As in general classifications cannot be right or wrong but useful, expedient, relevant, or not, the selection of a classification method depends on the objective. If the objects to be classified have more than one characteristic, the procedure may be complicated. Up to three numerical characteristics can expressively be presented in a graphic. In this case each object is represented by a dot. Classes are recognized by cluster areas. Although there are different well–known classification methods suitable for different fields of application, classification is always a heuristic task.

2 Feature Trees – Concept and Realization

The problem of finding a suitable classification and acquiring knowledge about a specific domain is closely related to the representation of this information in efficient data structures.

In the area of integrated circuits the major requirement for a knowledge representation is the efficiency and direct mapping of different classification schemes. In respecting these points tools like advisory systems and analysis environments benefit from simpler transformations into internal data structures and the applicability of specific algorithms on these structures.

The common form of classification knowledge is a tree structure that can naturally embody any hierarchical data.

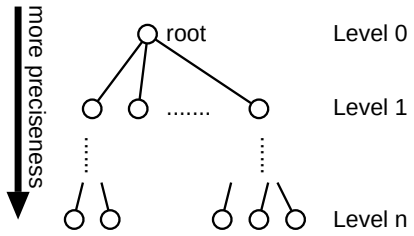

Fig. 1: Representation of knowledge in a feature tree

The central idea of a *feature tree* is to structure information according to their preciseness in the different levels of a tree (see fig. 1). The more precise the information associated to a tree node is, the greater its distance of it from the root will be. The knowledge available in a specific tree node is called a *statement*.

2.1 Definitions

A general treatment of terms for trees can be found in [2]. The following definitions focus on the relevant parts necessary for describing the properties of *feature trees*.

Definition: A *feature tree* FT is a tree where the following conditions are hold:

- Each tree node $x \in FT$ represents a statement S over one variable v.
- Let x be a tree node with a statement $S_x(V)$ and p the path from the root r to x, and

$$\forall y \in p : (\forall\ v \in domain(V) :$$

$$S_x(v)\ is\ true \Rightarrow S_y(v)\ is\ true)$$

Definition: An *feature association* A for a node n of a feature tree is the set of variables V_A where

$$\forall x \in V_A : S_n(x)\ is\ true.$$

This clearly defines how knowledge is to be represented in a feature tree. The tree itself defines a property to be represented.

2.2 Application

The rather straightforward definitions from section 2.1 can be applied to model classifications efficiently. Using multiple trees for different features the resulting wood of trees will characterize each data element associated to these trees.

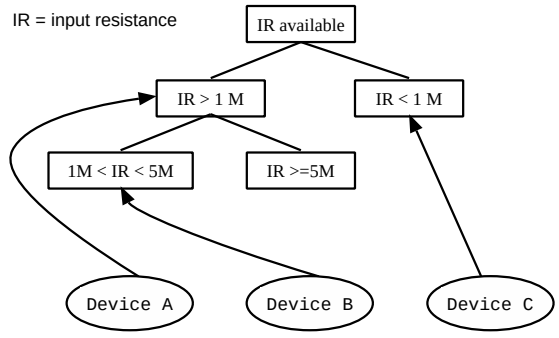

Fig. 2: Feature tree for input resistance

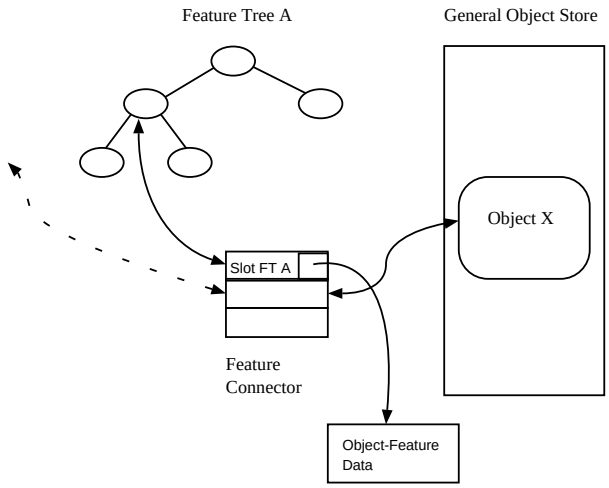

Fig. 3: Implemented data structure (feature tree and associated data)

For instance, the input resistance of a circuit can be described by an adequate tree (see fig. 2). This tree represents the classification for any data set to be stored according to the property *input resistance*. In this example the information about *Device A* is less precise than that of the other ones. The ability to represent this weak classification is very handy if some characteristics of the circuit are not available or if they are only approximations.

There are two methods to obtain information which is to be stored in the tree nodes. First, unsharp knowledge is coded in the nodes and is available with a link to the node. Second, data sheet information can be stored to reach a more precise knowledge. For instance, the range of an input resistance is given with the device technology. The exact value can be entered for more preciseness.

The classification by feature trees allows the use of efficient search algorithms ([1]). A library organized by feature trees offers efficient access to its data items. Because new properties can be added easily by adding a new feature tree, new acquired knowledge about an object can be added without disturbing the already collected informations. A feature tree itself can be expanded, if a classification can be refined, for instance due to new measurement techniques. If an object information can be refined according to this new tree nodes its association 'sinks' down to the respective node.

Feature trees also support the determination of 'compatible' objects. Consider a simulation task where models for electronic devices are required but not available for every element. The calculation of a 'close' match is sketched in the following procedure.

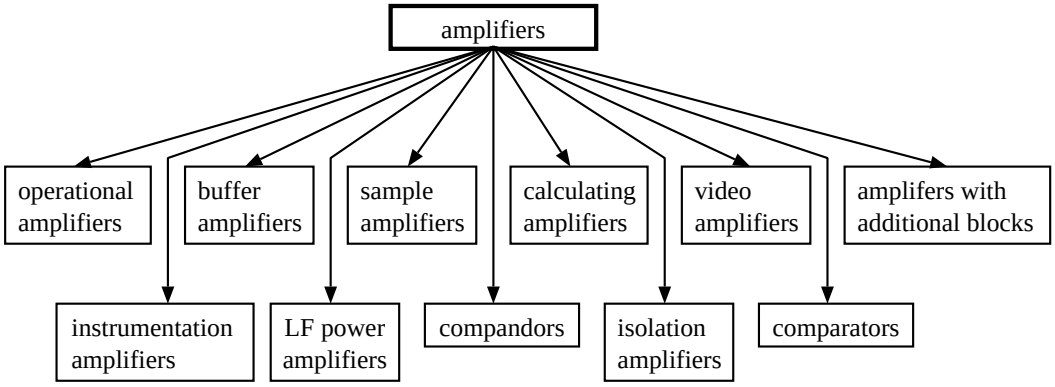

Fig. 4: Amplifier part of the class tree

Step 1: Get all nodes in the feature tree of the start-object.

Step 2: Find all associated objects of the nodes.

Step 3: If any of the objects satisfies the required property then *stop* and return the nodes.

Step 4: Select a node and remove it from the list of nodes. If the list is empty (no node can be selected) stop.

Step 5: Add the predecessor (level i-1) of the selected node to the list of nodes. If no predecessor exists goto Step 4:

Step 6: Recalculate all associated objects.

Step 7: Goto Step 3.

Different strategies are possible for step 4 which result in the set of the detected 'close' matches. A stack mechanism, for instance, will generalize one property first and then all the others, a queue will result in a balanced generalization over the set of properties. A deeper investigation will be subject to further research in the domain of analysis environments and goes beyond the scope of this paper.

2.3 Implementation

The ongoing implementation employs a slightly modified scheme than the one shown in figure 2 for better efficiency. The data objects are stored in a *General Object Store*. Each data object contains its private data. This includes invariant static information. For a certain device this can be technology, manufacturer, physical dimensions, the data from a data sheet, etc. The knowledge is represented by the links into the feature trees. Each link is bidirectional to allow unrestricted access. Every object is connected to a *feature connector* which serves as a multiplexer to all referenced feature trees. The data area for a certain feature tree link additionally offers the option to refer to an *Object-Feature-Data* component which stores informations only relevant to the combination of object and feature tree. This implementation schema is depicted in figure 3.

3 Concept of Circuit Classification

The classification of analog and mixed analog/digital integrated components is concerned with an unknown number of characteristics. Furthermore, the number of classes is unknown, too. To solve this multi–dimensional classification problem, circuit characteristics are divided into basic description elements:

- Overall functionality (*What is it to be used for?*),
- transfer function description (*How does it work?*), and
- pinout characteristics (*What is the interface behaviour?*).

In this way a set of three hierarchical structures has been defined:

- *class tree*,
- *transfer tree*, and
- *connection tree*.

A fourth structure, the *signal tree*, defines relations between two or more component pins. The interconnection of pins is done by physical nets. Each net can be described by signal characteristics.

The number of links to be established depends on the circuit class and the referenced tree nodes.

The tree definitions allow to code redundancy. The class tree, e.g. defines structural information of circuits. This knowledge defines in some cases whether references to nodes of the other trees are indispensable or necessary. The consistency of circuit description can be checked in this way. A second advantage of redundancy is the much more easier data access.

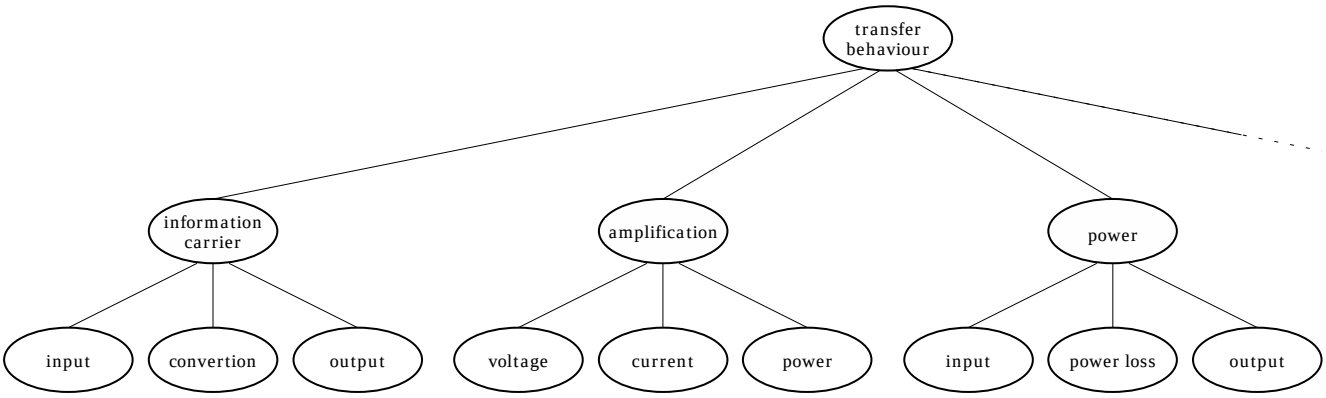

Fig. 5: Part of the transfer tree

3.1 Class Tree

The available functionality of analog and mixed analog/digital integrated circuits covers a wide range. The functional classification of these components is done with the class tree structure. The structure follows the view of application designers to circuits.

The main classes are *DC sources*, *AC sources*, *Switches and Multiplexers*, *high frequency circuits for radio, TV and wireless applications*, *control and regular circuits*, *amplifiers*, *converters*, *transducer*, and *application specific circuits*. A further division is given with forty sub–classes. The amplifier sub–classes are shown in figure 4.

Each class contains components with similar functionality. Therefore, the classes have typically equivalent circuits and herewith similar parasitic behaviour. The affiliation to a class defines a general EMC behaviour of a circuit. Most circuits belong to *one* class. But there are circuits which reference to *more than one* class, too. This is, for instance the analog–to–digital converter AD 678 which has an integrated sample&hold amplifier. In this case the circuit has references to the classes *analog–to–digital converter* and *sample&hold amplifier* to cover both characteristics.

3.2 Transfer Tree

The transfer behaviour of circuits has to be described by quite different characteristics. The validity of characteristics depends on the class affiliation. Main classes of the transfer tree are defined as *information carrier*, *transfer function*, *input/output coupling*, *amplification*, *power*, and *output signal shape*. Figure 5 illustrates a part of the tree structure.

The *information carrier* branch defines the description unit of the input and output signal. Further, the description of signal conversion is available, e.g. none, once, or multiple. If the information carrier of input and output signal are different, there must be at least one conversion.

The behaviour of some circuits can be described by a simple *transfer function*. This can be a linear characteristic of an operational amplifier as well as the sign control of a comparator. Mathematical functions can be defined to describe the transfer function of calculating amplifiers, too. In a common sense, the transfer function branch has to give a description like

$$\boldsymbol{\Phi}_o = f(\boldsymbol{\Phi}_i)$$

where $\boldsymbol{\Phi}_o$ and $\boldsymbol{\Phi}_i$ are the I/O signal vectors of a circuit. Circuits which do not have an easy description in this form have no reference to a transfer function.

The *input/output coupling* branch describes the coupling behaviour. Most circuits have a coupling from the input to the output and none in the opposite direction. Other circuits, e.g. analog switches, are bidirectional.

As a lot of analog circuits deal with *amplification*, the amplification description is very important. The branch is subdivided into voltage, current and power amplification.

The *power* branch covers three different aspects. The typical input power, the maximum/typical output power and the power loss.

Some analog or mixed analog/digital circuits have a well defined *output signal shape*. References to this branch are defined for functional generators and voltage regulators, e.g.

The more references of a circuit are given to the transfer tree, the more structural information are stored in the tree structure and the sharper is the circuit knowledge. Data sheet information are stored in assigned tree nodes. In this way, parameter values of a circuit are not given in a flat list. The structural information is conserved, too.

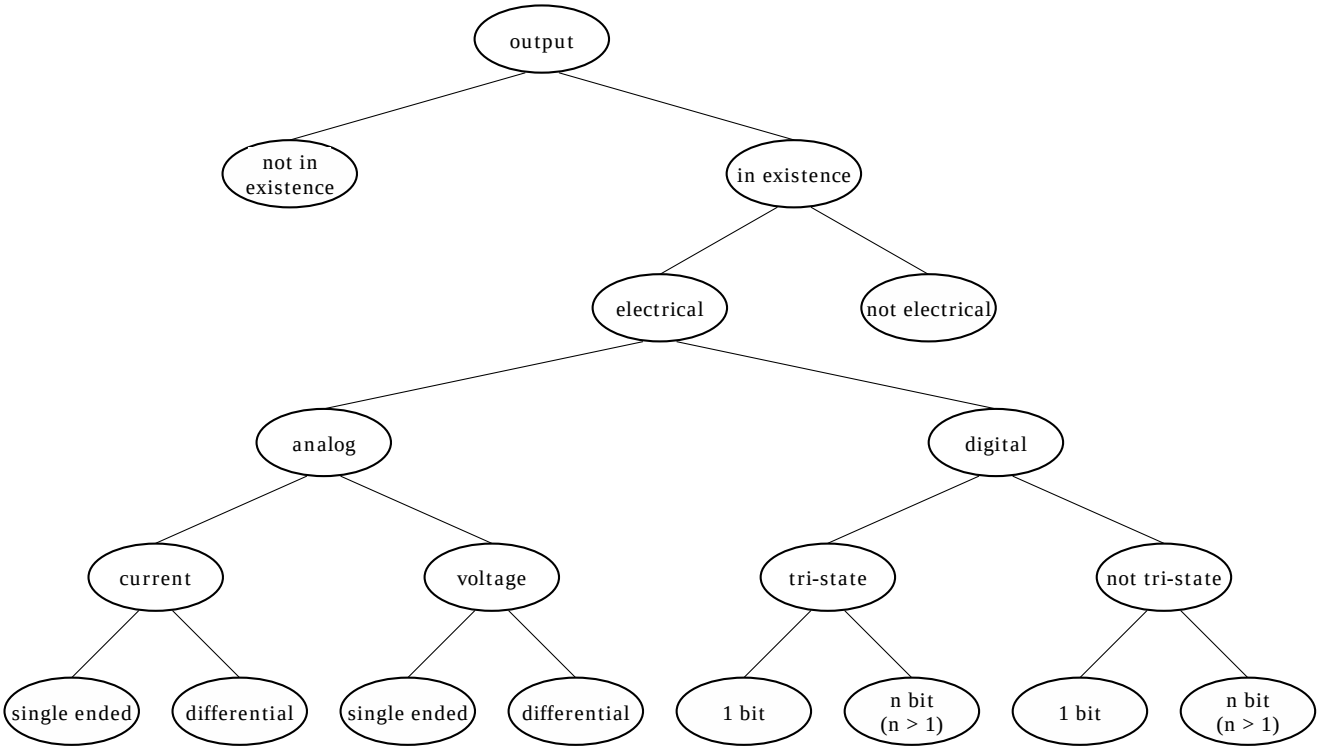

Fig. 6: Output part of the connection tree

3.3 Connection Tree

The circuits interface is given by the pinout behaviour. The characterization leads to the connection tree which consists of seven main classes and sixty–seven sub–classes. The main classes of device pins are defined as *input*, *output*, *reference*, *control*, *offset and compensation*, *power supply*, and *other*. Figure 6 shows the output part of the tree structure.

In the connection tree it is important to determine parameters which are necessary to describe the interconnection between pins. Data sheet information concerning pin parameters are stored in the tree nodes. Transmission line calculation needs pin parameter values as well as ideal circuit calculation.

Each pin of a circuit has exactly one reference to a connection tree node. If there are less references than pins, this is a lack of information. The data sheet values of the electrical pin characteristics are stored in data objects as shown in figure 3.

4 Application Example – Entering a New Component into the Library

The procedure to enter a new component into the library is a very complex task. The problem is that a lot of information is required which has to be stored context sensitive in the various tree nodes while keeping the dialogue with the user as simple as possible. The component entry is realized by four basic steps:

1. general dialogue,
2. pinout dialogue,
3. classification, and
4. transfer description.

The following example shows a simplified procedure. Especially the number of data sheet parameters is sharply reduced to keep the example brief.

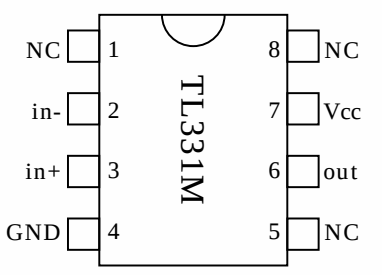

Fig. 7: Pinout of the comparator TL331M

With the general dialogue some basic inputs concerning device name, manufacturer, package, and management information like user name, date, and project are requested.

```
Device ID       : TL331M
Manufactorer    : Texas Instruments
Data Sheet      : Linear Circuits Data Book, Vol. 1, 1989,
                  pp. 4.81 - 4.82
Related Devices: TL331I, TL331C

User Name       : D. Wagenblasst
Date & Time     : 09-19-1995, 08:00
Project         : I/O Unit A18
Access Rights   : user
```

The pinout dialogue has to establish a link to a node of the connection tree for each pin. The connected tree node requires specific parameters which should be given by the data sheet. The following listing shows data object properties of the signal input pin no. 2:

```
Number      : 2
ID          : -I
Pair with   : 3
Model List : #1377
Model Index: 3
Input Offset Voltage: -/2/5 mV
Input Bias Current: /-25/-100 nA
Signal Unit: voltage
```

The classification assigns the component to one node of the class tree. The correct assignment requires some information about the overall functionality which is requested by a set of questions predefined by the corresponding tree structure. This means that the selection of a question depends on previously given answers. The structural pin information are used for the classification, too.

In this example only two questions have to be answered. The complete dialogue without available pin information would consist of ten questions. The classification dialogue for the comparator is shown below.

```
Question 1:
-----------
Are the information signal input(s) and
output(s) connected each other with a low
impedance less than 1000 Ohm?
 (y/n/h) > n

Question 2:
-----------
Is the information signal input part of
an oscillator, with an input signal
dependend frequency?
 (y/n/h) > n

Classification result:
----------------------
The component TL331M belongs to the
comparator class.
```

The last dialogue deals with the transfer description. The class tree defines typical transfer characteristics for each class. In this way, links can be set up to specific nodes of the transfer tree. The dialogue orders the data sheet values to store them in data objects which are connected to these nodes.

5 Conclusion

In this paper a concept for an overall characterization of analog and mixed analog/digital components is introduced. The basic description elements of a component, functionality, transfer characteristics, and pinout behaviour are considered by three hierarchical structures. With the help of these structures, a knowledge conserving component library can be built up. The field of application for this library is in the area of advisory systems and analysis environments focussing on EMC analysis on printed circuit board level. The conservation of structural data is an add–on to existing component libraries. The introduced concept guaranties the usability of this library for EMC analysis tools. This means, that new components can easily be added to the library while the process of insertion into the tree structures is kept transparent to the user. On the other hand the structural knowledge of the components is accessable at all times in a practical manner as well.

Acknowledgements

This research is part of the project JESSI AC 12 *Analog System Design Environment* and the national German project *Bibliothek für komplexe analoge Bauelemente*. Both projects are supported by the German government, Department of Research and Technology under grants 01 M 2885 D, 01 M 2885 D, and 13 MV 0032. The responsibility for this publication is held by the authors only.

References

[1] Knuth D. E., *The Art of Computer Programming Volume 3: Sorting and Searching*, Addison Wesley, New York, 1973.

[2] Horowitz E., Sahni. S., *Fundamentals of Data Structures*, 13th Printing, Computer Science Press, New York, 1985.

[3] Vogel F., *Probleme und Verfahren der numerischen Klassifikation*, 1st Printing, Vandenhoeck & Ruprecht, Göttingen, 1975.

[4] Wagenblaßt D., Rissiek W., *Layoutanalyse analoger Schaltungen basierend auf einem erweiterten Bibliothekskonzept*; Proceedings of 3. GME/ITG–Diskussionssitzung in Bremen, Entwicklung von Analogschaltungen mit CAE–Methoden, September 29/30, 1994, pp. 151 – 156.