A Systematic Analysis of Reuse Strategies for Design of Electronic Circuits

Manfred Koegst¹, Peter Conradi², Dieter Garte¹, Michael Wahl²

¹ Fraunhofer-Institut für Integrierte Schaltungen, EAS Dresden, Zeunerstr. 38, D - 01069 Dresden, e-mail: koegst@eas.iis.fhg.de
² Universität-GH Siegen, Fachbereich 12, Fachgruppe Rechnerstrukturen, Hölderlinstr. 3, D - 57068 Siegen, e-mail: peter@rs.uni-siegen.de

Abstract
In this paper a number of reuse approaches for circuit design are analysed. Based on this analysis an algebraic core model for discussion of a general reuse strategy is proposed. Using this model, the aim is to classify different reuse approaches for circuit design, to compare the applied terms and definitions, and to formulate classes of typical reuse tasks. In a practical application with focus on retrieval and parameterisation techniques, this model is on the way to being applied to DSP design issues.

1 Introduction
In the past, synthesis of electronic circuits was thought of as building the formal specification from scratch. Reuse of electronic circuits means kinds of instantiation of existing electronic circuit concepts into new designs [7], [8]. Since this general idea is quite usual, many other authors have reported about these techniques in different CAD-domains. The daily work of electronic CAD engineers divides the design into two strictly separated regions: the newly created functionality and the reused components. A good example is given by the well known FSM/Datapath design paradigm, addressing two parts of design: the part control in combination with the part datapath, containing functionality, represented by reusable components. The process of control design requires selection of reusable components given, e.g., by VHDL descriptions [11], [13], [14]. This requires parameterisable behavioural specifications and usable design, verified in distinct parameter spaces. Examples for parameterisation are amount, bitlength, and kinds of operators and functions [2]. For reconstruction of synthesis, not only basic elements or complete synthesisable circuit specifications are necessary, but also archived design flows [15]. In typical cases they are parameterised; in some cases this is simplified by enumeration.

Due to the growing amount of reusable components, adequate formal methods for their description are necessary for two reasons: to work on suitably combined reuse-focused data management, retrieval, and parameterisation techniques. A first proposal for a suitable formal reuse approach was presented in [12]. Now the formal description is extended in this paper by a common view for retrieval and parameterisation in theory as well as experimental implementation.

The paper is structured as follows: First an overview of goals and problems for reuse of electronic circuits is presented. In sections 3 to 5 we describe our library organisation and the technique of design by reuse as well as design of reuse. Section 6 discusses the task of database management. We finish with specific implementation issues.

2 Reuse for circuit design
2.1 Reuse cycle for Case-Based Reasoning
Starting with learning and problem solving processes, some authors [1], [4] are formalising recognition and reuse as a paradigm of the so-called Case-Based Reasoning systems (CBR). Here, the strategy of reuse is presented as a cycle. It is mentioned that the process of learning and problem solving does not only contain reuse of learned patterns, but even the reuse of applied solution procedures. In Fig.1 we apply this cycle to one case of circuit design relying on a library of reusable components, called intellectual property (IP).

The reuse cycle contains four steps:
(a) retrieve (or identify) by selection of an IP for the specification under actual user-specified conditions [2],
(b) instantiate necessary subdesigns following a given specification, if an IP was selected; alternatively, definition of a design flow concerning the constraints,
(c) design under application of a fixed flow, verification and revision of design. The result is an optimised circuit,
(d) if the decision process leads to retaining the synthesised circuit in a library, its specification has to be generalised (for reuse-relevant parameters) and a design flow has to be generated.

In a project, founded by the Stiftung Innovation of Rhineland-Palatinate, Case-Based Reasoning aspects are under research, especially for the retrieval aspects of this cycle. This effort is resulting in a prototype system, named READEE (Reuse Assistant for Design in Electrical Engineering), discussed in more detail in section 7.

2.2 Classes of reuse tasks

From this cycle following problem classes can be derived:

• the standardisation and analysis of specifications (step a, d),
• the design of electronic circuits by usage of IPs (design by reuse, steps a, b, c),
• the design of IPs (design for reuse, step d),
• the development of parameterised procedures for synthesis (steps b, d),
• the development of parameterised environments for verification (steps c, d), and
• the database management for IP libraries (step d).

3 Libraries with reusable components

3.1 Definition of libraries

In generalised understanding of a standard library of devices (typical gates), the elements of an IP library have to be described by two components: first by a behaviour specification on an intermediate level (e.g. register transfer level) and second by a script (design flow) for the transformation of this specification to the destination level (e.g. gate level). In Jha and Dutt [10] this abstraction to a high-level library is classified as a key for reusing during circuit design.

Consequently, we describe a library $LIB$ of IPs by a library $LB$ of parameterised specifications and a library $LF$ of parameterised design flows.

Any element $B = (K_B, V_B, D_B)$ of library $LB = \{B\}$ of behavioural specifications consists of a parameterised specification $K_B$ of the behaviour to be synthesised, of the set $V_B$ of parameters of the specification and the set $D_B$ of allowed parameter values. Any value $d_B$ of $D_B$ defines a parameterless specification $K_B(d_B)$.

For two specifications $b$ and $b'$ we define the following relations:

- they are equal $(b = b')$ if the specifications are describing the same behaviour, and
- $b'$ is an extension of $b$ $(b \leq b')$, if the specification of $b'$ is containing the specification of $b$.

Any element $F = (K_F, V_F, D_F)$ of a library $LF = \{F\}$ of design flows consists of procedures $K_F$ for synthesis and environments for verification, a set $V_F$ of specification parameters and set $D_F$ of allowed parameter values.

Two design flows $f$ and $f'$ of $K_F$ are equal $(f = f')$ concerning a specification $b$, if the application of both design flows $f$ and $f'$ to $b$ leads to the same circuit $f(b) = f'(b)$. Any value $d_F$ out of $D_F$ is the base for a parameterless design flow $K_F(d_F)$.

On the base of two libraries $LB$ and $LF$ a library $LIB = \{I\}$ of IPs is defined. Any IP $I = (K_I, V_I, D_I)$ of $LIB$ consists of

- a pair $K_I = (K_B, K_F)$ of parameterised specifications for the behaviour $K_B$ and parameterised design flow $K_F$ for synthesis and verification,
- the set $V_I = V_B \cup V_F$ of parameters, and
- the set $D_I$ of allowed values of the parameters.

$D_I$ is a subset of all possible values for parameter set $V_I$, since not any combination of specifications of $LB$ and design flows of $LF$ will make sense. In the following we suppose $V_B \cap V_F = \emptyset$, then relation $D_I \subseteq D_B \times D_F$ holds. By every value $(d_B, d_F)$ of $D_I$ a parameterless IP $(b, f)$ with $b = K_B(d_B)$ and $f = K_F(d_F)$ is defined. If design flow $f$ is applied to specification $b$, the specification $b' = f(b)$ describes a synthesised circuit at destination level.

3.2 Relations and terms

For circuit design with reuse components every IP $I = ((K_B, K_F), V_I, D_I)$ is allocated to the class $S(I) = \{f(b) | \forall (d_B, d_F) \in D_I | b = K_B(d_B) \land f = K_F(d_F)\}$ of circuits synthesisable by IP $I$.
Thus for a pair \((I, I')\) of IPs two relations can be defined:

- \(I\) and \(I'\) are equal \((I = I')\), if \(S(I) = S(I')\)
- \(I'\) is an extension of \(I\) \((I \leq I')\), if \(S(I) \subseteq S(I')\), i.e., \(I\) can be substituted by \(I'\).

Furthermore, in [2] a similarity function is defined. E.g., the specification of a 32-bit adder and a 16-bit adder are very similar. Thus, it is much easier and faster to design a VHDL specification of a 32-bit adder adapting a code of a (likewise defined) 16-bit adder. By doing so, for the assignment of an IP to a given specification, a threshold must be defined to decide whether the calculated value for the similarity will be sufficiently high to map the specification to the IP.

The elements of the libraries LB, LF, and LIB consist of a triple \((K, V, D)\), where \(K\) is the kernel of the element. Set \(V\) of kernel parameters is noted as configuration, and \(D\) is a domain of tuples of valued parameters. A library element is named component. The pair \((V, D)\) of configuration and domain characterises the environment of the component.

By fixing the value \(d = (d_B, d_F)\) of \(D\) an IP \(I = ((K_B, K_F), V_I, D_I)\) is allocated to an instantiated IP \(I(d) = (b, f)\) where \(b = K_B(d_B, d_F)\) and \(f = K_F(d_B, d_F)\).

Moreover, we use the following definitions:

- An IP \(I = ((K_B, K_F), V_I, D_I)\) is configurable, if the set \(D_I\) contains more than one element, i.e., \(|D_I| > 1\).
- An IP \(I = ((K_B, V_I, D_I)\) is an instance, if the set \(D_I\) consists of exactly one element, i.e., \(|D_I| = 1\).
- If we treat the mapping process of a specification to an IP \(I = ((K_B, V_I, D_I)\), we understand the term instantiation as the task of selecting a distinct tuple \(d\) out of \(D_I\). The result is \(I(d) = ((K_B, V_I(d))\) or \(K_I(d)\). In the case of a generic VHDL-description of an IP, the environment \((V_I, d)\) is expressed by the keywords generic and port map. In this second form of description \(K_I(d)\) all parameters of the kernel are substituted by related values of the configuration.

### 4 Design by reuse

#### 4.1 Mapping at a reuse library

The design by reuse comprises the following four steps:

- Analysis of the specification with regard to correctness, ability of synthesis, and reusability.
- Decomposition of the specification to partial specifications \(P = (b_P, c_P)\), consisting of a behavioural specification \(b_P\) and relevant constraints \(c_P\).
- Design of partial specifications \(P\), i.e., allocation of IPs or complete new design, if an allocation is impossible.
- Composition of verified partial circuits (synthesised partial specifications) to a complete circuit and validation of the overall behaviour. The reduction of the verification effort is a major advantage of the reuse concept for electronic design [18].

For mapping to a reusable library three cases are to be distinguished (c.f. Fig. 2):

- selection and instantiation transform the specification into a partial circuit, which is itself an element of the library,
- selection and generating lead to an instantiated IP, consisting of specification and design flow,
- the partial specification could be an IP, thus no selection is necessary and after instantiation a circuit can be created immediately by applying the related design flow.

#### 4.2 Utilising Reuse Components

The fundamental task for design of a partial specification is the retrieval, i.e., the decision whether \(P\) can be mapped to an element of the IP-library (c.f. Fig. 3). If this is possible, an IP \(I = ((K_B, K_F), V_I, D_I)\) is chosen with a kernel \(K_I = (K_B, K_F)\), consisting of a parameterised behavioural specification \(K_B\) and a parameterised design flow \(K_F\). After that, a suitable instance \((d_B, d_F)\) of \(D_I\) is chosen under influence of constraint \(c_P\) having the property \(b_P \leq K_B(d_B)\). In case such a mapping is impossible, a design flow \(f_P\) should
be composed, dependent on the constraint \( c_P \).

The constraint \( c_P \) may be consist e.g. of a set of so-called Specification Mapping Control rules (SMC) [5]. The application of such rules performs plausibility checks and enables safe re-applicability of the multidimensional design space of the given circuit. The result is an instantiated IP \((b, f)\) with \( b = K_B(d_B) \) and \( f = K_F(d_F) \) or \( b = b_P \) and \( f = f_P \). Finally, the application of design flow \( f \) to the behavioural specification \( b \) leads to a circuit \( f(b) \) for the given partial specification \( P \) (Fig. 3).

The components of an IP \( I = ((K_B, K_F), V_B \cup V_F, D_I) \) based on the behavioural description \( B = (K_B, V_B, D_B) \) and a design flow \( F = (K_F, V_F, D_F) \) are as follows:
- \( K_B \) - VHDL behaviour description based on generics,
- \( V_B \) - set of generics used in the VHDL description,
- \( D_B \) - set of allowed values of the generics,
- \( K_F \) - design flow based on the Synopsys design compiler, possibly controlled by additional parameters,
- \( V_F \) - set of switches and parameters of the compiler,
- \( D_F \) - set of allowed values of those switches and parameters, and
- \( D_I \) - subset of allowed values of \( D_B \times D_F \).

5 Design of Components for Reuse

For the design of reusable components a specification \( b' \) (e.g. a synthesised circuit) is to be generalised to a class of behaviour specifications and design flows. The result should be an IP \( I = (K_I, V_I, D_I) \) with \( K_I = (K_B, K_F) \), \( V_I = V_B \cup V_F \) and \( D_I \subseteq D_B \times D_F \) which contains a tuple \((d_B, d_F)\) with \( b' = f(b) \) for \( b = K_B(d_B) \) and \( f = K_F(d_F) \).

The design of a reuse library component starts with a specification \( P = (b_P, c_P) \) containing a behaviour specification \( b_P \) and some constraints \( c_P \) as well as a fixed abstraction level (Fig. 4). For the generation of IPs in [9], [16] it is assumed that the behaviour is described in VHDL and the set of possible design procedures is given by the Synopsys design compiler [19].

6 Database management for reuse libraries

In the future, engineering databases will carry parameterised specifications, their documentation, and parameterised design flows. Database frontends will take over the tasks of IP-mapping.

In this respect, the following points are under development:
- data management of parameterised specifications on different levels of abstraction,
- standardisation and data compression for retrieval,
- suitable procedures for retrieval, and
- design or adaptation of hierarchical searching mechanisms.

These tasks can be solved in our work by combining the classical data management technique of academic and commercial CAD vendors [2], [6], [10] with CBR techniques [1], [3], [4].

Fig. 3: Reuse of components

Fig. 4: Design of components
7 Application

The current implementation of the above mentioned data model, knowledge base, and retrieval methods is performed by READEE, developed in an interdisciplinary project of the University of Kaiserslautern, Department of Computer Science, Centre for Learning Systems and Applications. READEE concentrates on two state-of-the-art problems:

- and innovative retrieval methods, and
- the parameterisable description of designs.

This program is designed as a generic database system with the capability to easily add categories for retrieval attributes. The process of retrieval and parameterisation is shown in Fig. 5.

![Fig. 5: Query and parameterisation technique](image)

The process description of Fig. 5 is based on SADT [17]. The query entry can be formulated in plain keywords. They could be derived by parsing specifications, written in a hardware description language. After retrieval of an IP, detailed specifications and/or constraints are added, based on a given circuit-specific data sheet. This questionnaire filling-in is controlled by a preliminary, mostly simple plausibility check, using interactive SMC rule reaction [5]. The reason for a technical two-step process is that the found circuit-specific data itself have to offer the detailed rules of specification matching this class of topologies. Our implementation strategy is based on a common visualisation of the two-stage retrieval and parametrisation process (c.f. gray box in Fig. 5).

Since the user entries for retrieval and parameterisation are of similar type, a method for their combination has the advantage of user access homogeneity. To implement this, both activities - database query and questionnaire entry - are integrated into one successively extended questionnaire. The application skill of the prototype READEE regards DSP applications.

References


