19

A Compound Information Model for High-Level Synthesis

Peter Conradi
Center for Microelectronics
University of Kaiserslautern
Germany

Nikil Dutt
Information and Computer Science
University California at Irvine
USA

Abstract

High-level design and behavioral synthesis are the next natural steps in the evolution of CAD tools, dealing with the new generation of complex designs. Hence, a necessary future step in this progress of design automation is the standardization on more abstract levels of design. The basic problem is the lack of a good information model for high-level design. While the structure representation can be adapted from the CFI DR efforts, information models at the behavioral level are still not existent. This paper will outline the ingredients of a compact information model for high-level synthesis (HLS) from high-level languages such as VHDL.

Our contribution in this work is the presentation of an Information Model for HLS that captures the desired representations in a closed and minimized manner. The work is based on experiences with different HLS tools and algorithms at University California, Irvine.

1 Introduction

CAD research and development in the past two decades have resulted in fairly mature design tools and automation of several steps in the logic and layout design of electronic circuits. The CAD Framework Initiative (CFI, Inc.) has developed related framework and interoperability standards for these areas over the past 6 years. In January 1993, a first standard version for design representation was made available (DR 1.0). Within the CFI, different working groups and technical subcommittees are working on design representation issues for the logic and layout levels.

However, it is widely acknowledged that in order to cope with the explosion in design complexity resulting from advances in semiconductor manufacturing, the design process cannot be

F. J. Rammig et al. (eds.), *Electronic Design Automation Frameworks*

efficiently performed at the logic and layout levels.

Thus the interoperability demands for high-level synthesis have been growing over the last years, however without the necessary companion of suitable standardization work.

Since a number of years so called **Information Models (IM)** are used [1, 2] to explain relationships between different conceptional data entities in a formal and human readable manner. Thus standardization organizations like CFI, Inc. and EDIF agreed on the common IM language EXPRESS, developed by the STEP community.

Thus IM's for all EDA parts of design are essential for standardization and framework construction purposes. Using such an IM, procedural interfaces such as exchange formats or database schemes can be defined. In this paper we use the STEP-related format EXPRESS in its graphical form to present a common IM set for high level design and synthesis. The main problem in constructing and discussing an IM is to define a widely applied and efficient set of information used by the high-level design tasks.

2 Related work

Application topics for IM applications in CAD environments affect

- all tasks, which help the designer to **organize** his information space, version handling mechanisms, preparation for reuse, documentation etc.,
- all domains, where designers have to manipulate different **design descriptions** simultaneously (Gajski et al. [10], Rundensteiner [11], [12], Wu et al.[4]), and
- all design phases, where **different abstraction levels**, e.g., behavioral description, layout, technology information etc. are to be managed.

The user access in such a system could be done in a number of access modes that support specific needs on different views (black box information, floorplan information, technological conditions, RT-behavior, RT-structure, etc.).

Using IM techniques for design data abstraction has the following benefits: If the same IM is the base for concurrent software development, data adoption by each of the approaches will be easy. A database approach allows shared parallel development during component library design phase as well as shared usage of libraries. Object oriented treatment of component libraries and synthesis methods are state of the art. Links through different abstraction levels of a design enable reuse of design. The reason especially in high-level design is that use of components require knowledge about applicable operations. If the operations are not standard mathematical expressions, behavioral operation descriptions are necessary to meet the design goals.

Additionally, the technique of IM leads to more transparency of the design requirements in educational domains. The comparison and harmonization of information models of analog,

digital, high-level and system level designs could offer more general insights into design processes.

For many years various types of tool integration have been suggested in the literature. The main problem of tool integration is the **definition of a unified data format**, that must be variable in size and volume in order to contain the required information.

There has been some related work in defining unified representations for HLS. For example, in the design trajectory from HL-design to layout, some authors propose unified representations [3, 4, 8]. However, these efforts were tuned towards synthesis and design support; they did not address the issue of defining a complete, minimal, and realistic usable IM for HLS. On the other hand, some authors are working on deriving information models for standard HDLs that support design capture and simulation [3, 4], but they do not address the issues of IM for the complete design trajectory including tools, algorithms, datastructures and libraries.

3 Compound Information Modeling

High-level design typically starts with the description of the design intent in a behavioral fashion (using an HDL such as VHDL), along with design constraints (such as area and power consumption) and overall design goals (e.g., maximize speed of execution). The designer has to specify a target library of components to allocate specific components. The high-level design synthesis process refines this behavior into an implementation composed of a structural netlist (datapath) of the allocated library components, sequenced by a Finite State Machine (FSM). A feasible output of the high-level design process is a complete FSM sequencing a

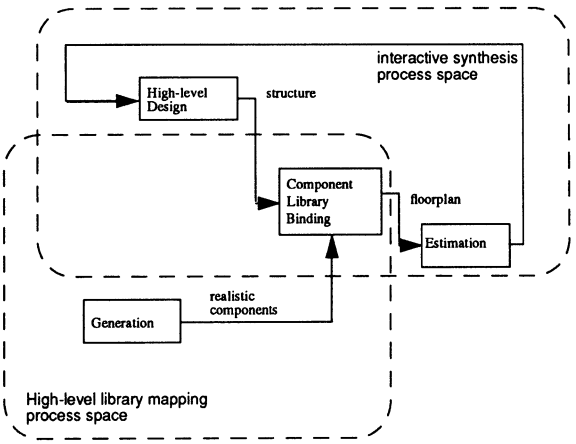

Figure 1:Typical incremental design process on system/high-level abstraction domains

datapath, that meets the design constraints, as well as the design goal. Thus, high level design is a repeated sequential process as shown in the SADT-diagram of Figure 1.

In this process the following representations have to be managed:

- Inputs:
  - Design behavior, expressed in a HDL (e.g., VHDL)
  - Design constraints
  - The goal functions for design
  - The resource or component library for refinement
- Internal Representation:
  - Control/data flow graph
  - Finite state machine or state table
  - Refined internal representation
- Outputs:
  - Structural netlist of library components (datapath)
  - Extended finite state machine (controller)
- Design Model: Finite State Machine with Datapath (FSMD)

In the complex multispace of behavior, structure and physical informations we are using a number of description methods. During this paper a **common view** of information will be constructed based on these diagrams.

We begin with the specification level, where no underlying realization model is existent. Specifying a high-level-design there are traditionally four kinds of different description methods. All these different representations could express different views of the same system object. While the **structure** net is a model of the units itself with their interconnection to other units, the **control flow** and **state transition diagram** show how the control flow is going between transitions; and the **data flow** diagram shows the activity of units and their flowing data.

High-level synthesis is based on specific design models, which are understood as the main frame for implementations. Following Gajski et al [9], the most suitable design model is the Finite State Machine with Datapath (FSMD).

In state-of-the-art high-level to layout synthesis, a set of all required design informations is to be stored for synthesis and reuse purposes. These different representations are linked after scheduling, binding, placement and routing tasks.

The basic design model used in HLS is the Finite-State-Machine with Datapath (FSMD) [9], where an FSM controls RT-level components in the datapath and sequences the design through various behavioral states. The FSMD model provides the link to the lower-level tasks of logic,

layout and physical design. Based on the FSMD model, we found that the following kinds of information are collected together in order to find this complete set for an IM:

- Finite State Machine
- Scheduling information
- Data Flow Modeling
- Binding Information
- Structure representation
- Placement and routing information
- Layout configuration

4 Minimal Information Modeling

There is a trade off between describing data in a minimal form and conserving the semantic of the necessary description. Of course, it may be possible to describe all kernel data by configurations of nodes and links like in [4], but such a description has limited application scope. In our approach, the goal is to extract an IM of given programs, algorithms, and examples, which has the following advantages:

- defining a simple control flow model, which can be used for structured and unstructured control (see [7] for a definition) of parallel or sequential control, with different kinds of extensions,
- combining data flow and structure into one bipartite net, using the minimal additional effort to model the needed features.

The specific partial models will be described in the next sections.

4.1 Basic FSM

A HL design is typically specified in an imperative HDL (such as VHDL), and is refined into the FSMD model, where the control is embedded in a symbolic FSM. Hence it is necessary to contain at least this information.

We can represent a general FSM controller using the Mealy-FSM model. Figure 2 shows the

information model:

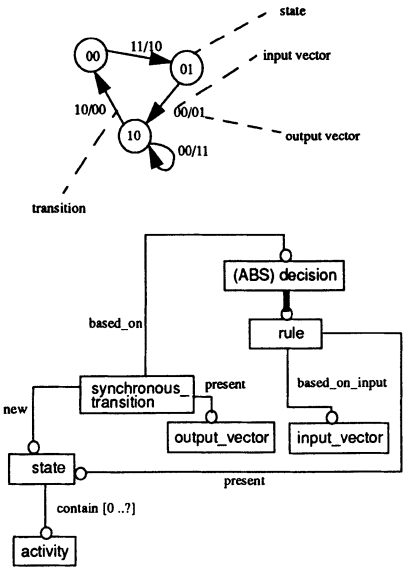

Figure 2: EXPRESS-G-diagram of states and transitions

The information model consists of the entities *state*, *transition* and *rule*. Each *transition* is connected with a distinct *rule*, e.g. realized as a EXPRESS-function or a string, in the last case interpreted by database method. The execution of the *rule* offers the next-state- *decision*, based on the contents of two linked entities: *input_vector* and *state*. The execution of the database method results in a fixed link to one new *state* relation of the entity *transition*. An additional database method would be needed to copy the new *state* attribute from the decision-activated *synchronous_transition* to the *rule* in order to allow navigation through the FSM states.

Additional to the usual definition we define one or more entities of type *activity* as the task, executed during one state. With this option we have the capability to implement state-containing control and data flow parts.

Here we consider only flatted FSMs. If needed, composed FSMs could be modeled in a hierarchical way using the synthesis model of the FSMD, where each datapath component could hierarchically represent a lower-level FSMD.

4.2 Unified control

The behavior of the design can be expressed in the form of flows of control, along with associated data transformations. In a general context, these control transitions can be synchronous (i.e., transition on a clock), or asynchronous (i.e., transition on an event). Furthermore, control flow itself can be serial (sequence), parallel (fork), or conditional (select). The atoms in the control views are *activities*, interpreted as data flow blocks like shown in the next section. Hence, for an *activity* that contains all inner-state actions, we have to define:

- synchronous or asynchronous control, and
- conditional, sequential or parallel tasks inside of a state

An activity is controlled by a so-called (asynchronous) *transition*, a meta-control unit, which can be interpreted as a superset of a selection (*select*), a sequence (*sequence*), or a parallel control flow (*fork*), including the individual related join construct, like shown in Figure 3.

The *transition* bases on a *decision*, which could be a *rule*, modeled by an *activity* on a more detailed level.

In this way the capability to model structured control-flow (CFG) and fork-join (FJG) constructs is given.

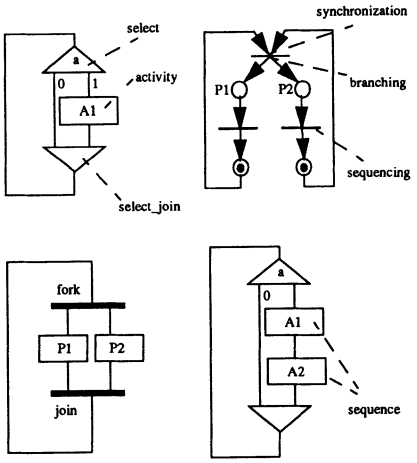

Figure 3: Different Control types

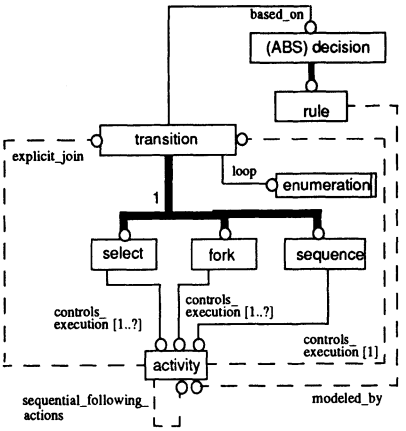

Figure 4: EXPRESS-G-diagram of unified control

Figure 2 and 4 have strong similarities. They can be combined quite easily by adding the *state*-construct to Figure 4. Also note that the *fork* construction in Figure 4 is equivalent to the Petri Net [13] form of control.

4.3 Structure, Data Flow

In a data flow view the same *activities* are containing scheduled Data Flow blocks. A Data Flow block contains operations such as READ and WRITE, algebraic operations and possibly decision nodes.

The complete Data Flow scenario could be understood as a interconnected network of activities by connecting affiliated write and read operations together.

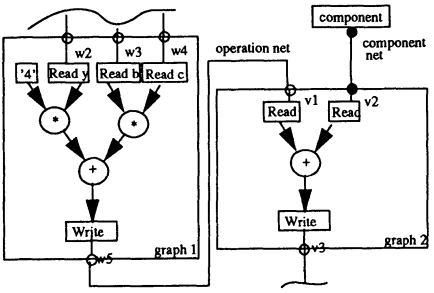

Figure 5: Dataflow graph and operation net

Exceptions are physical ports in Data Flow diagrams -- they are connected to real components; hence we have to combine the component-interconnections and operation-variables into one heterogeneous network.

For reasons of practical implementation, we propose to implement the Data Flow diagram in combination with the six-boxes connectivity model of the CFI, shown in Figure 6. In that way we understand operation nets as a partially coupled, but alternative representation of the design including the capability of defining relationships to component nets by bindings. Details of this model can be found in [6].

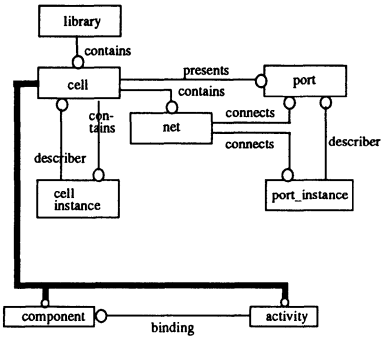

Figure 6: Extended CFI-DR model

4.4 Physical Layout

Floorplan and layout quantities will be described as physical location attributes, added to component and interconnection entities at detailed levels of granularity.

5 Summary

Our work is an initial effort towards the representation of a compound, integrated model of high-level design information. In particular, we presented a basic IM, consisting of

- integrated modeling of sequential, concurrent and conditional control flow
- combined data flow and structure modeling

This work was based on different experiments with design examples and synthesis tools, made at the University California, Irvine.

We believe this work complements the existing CFI work in the areas of DR, CIR, and VHDL IM. Future work will investigate applicability of this IM to a wider domain of HLS environments, as well as to its applicability in conjunction with existing IM efforts within a

broader design context.

Literature

[1] Lau, R.Y.W.: "Proposal for an Information Model for EDIF", Technical Report # UMCS-91-6-2, Dept. Computer Science, University of Manchester, U.K., 1991.

[2] Marshall, M.A.J., Kahn, H. : "A Structural Information Model of VHDL", in Mermet, J., (ed.), "VHDL for simulation, Synthesis and Formal Proofs of Hardware", Kluwer Academic Publisher, 1992.

[3] Knapp, D. W., Parker, A. C. : "A unified Representation for Design Information", Computer Hardware Description Languages and their Applications, IFIP, Elsevier, 1985.

[4] Wu, A.C.-H., Hadley, T.S., Gajski, D.D.: "An efficient Multi-view Design Model for Real Time Interactive Synthesis", Proc. ICCAD, 1992.

[5] CFI: "Design Representation, Electrical Connectivity Information Model and Programming Interface", CAD Framework Initiative Inc., Version 0.9.4-071092, 1992.

[6] Conradi. P.: "Information Analysis in High-level Synthesis", Technical Report # 94-39, Dept. Computer Science, University California, Irvine, 1994.

[7] Ang, R. P., Dutt, N.: "Equivalent Design Representations and Transformations for interactive scheduling", Proceedings of ICCAD, pp. 332 - 335, 1992.

[8] Chaiyakul, V., Gajski, D.D.: "Assignment Decision Diagram for High Level Synthesis", Technical Report # 92-103, Information and Computer Science, University California, Irvine, 1992.

[9] Gajski, D.D., Dutt N.D., Wu, A.C-H, and Lin, S.: "High-Level Synthesis: Introduction to Chip and System Design", Kluwer Academic Publishers, 1992.

[10] Gajski, D.D., Vahid, F., Narajan, S., Gong, J.: "Specification and Design of Embedded Systems", Kluwer, 1994.

[11] Rundensteiner, E., A., Gajski, D.D. :"A Design Representation model for High-Level-Synthesis", Technical Report, University California at Irvine, 1990.

[12] Rundensteiner, E.: "Object-Oriented Views: A novel approach for Tool Integration in Design Environments", Technical Report # 92-83, Information and Computer Science, University of California, Irvine, 1992.

[13] Peterson, J.: "Petri Net Theory and Modeling of Systems", Prentice Hall, 1981.