Vol.11 No.6 J. of Comput. Sci. & Technol. November 1996

The RTL Binding and Mapping Approach of VHDL High-Level Synthesis System HLS/BIT

Yan Zongfu (颜宗福) and Liu Mingye (刘明业)

Department of Computer Science & Engineering
Beijing Institute of Technology, Beijing 100081

Received September 11, 1995; revised January 18, 1996.

Abstract

This paper describes a VHDL high-level synthesis system HLS/BIT with emphasis on its register-transfer level (RTL) binding and technology mapping subsystem. In more detail, the component instantiation mechanism and the knowledge-driven approach to RTL technology mapping are also presented.

Keywords: High-level synthesis, RTL synthesis, technology mapping.

1 Introduction

Though the function and the design complexity of electronic systems have been increasing exponentially, the design cycle of a system is being shortened sharply under the competition pressure. The demand on the development tools of electronic design automation (EDA) is becoming more and more urgent. This advances the development of EDA.

Since the traditional logic-level design method cannot adapt to such intense competition now, the high-level design method has been introduced. The high-level design method allows the electronic design to be accomplished at the highest abstract level with the aid of efficient physical means and can give users higher efficiency. By this method, designers can work on behavioral or structural level instead of gate or transistor level, can describe the system function at behavioral level that is independent of specific technology, and can rapidly complete highly efficient design by synthesizing the system specification to actual circuit that satisfies constraints given by users. Now, the new generation EDA tools whose kernel is high-level synthesis system emerge as the times require. The new generation EDA tools adopt the structural design method and the unified standard hardware description language, such as VHDL. They can accomplish automatic high-level synthesis and use the mature artificial intelligence techniques.

The VHDL high-level synthesis system HLS/BIT, which is developed by Research Center of ASIC, Beijing Institute of Technology, is a prototype system of the new generation EDA tool. This paper will introduce the system HLS/BIT with the emphasis on RTL binding and technology mapping. The rest of this paper is organized as follows. Section 2 provides an overview of the system HLS/BIT. The intelligent binding component library (IBCL) used for RTL mapping and the component instantiation mechanism are explained in Section 3. Section 4 describes the knowledge-driven RTL technology mapping method. Finally, the status and some future directions are presented in Section 5.

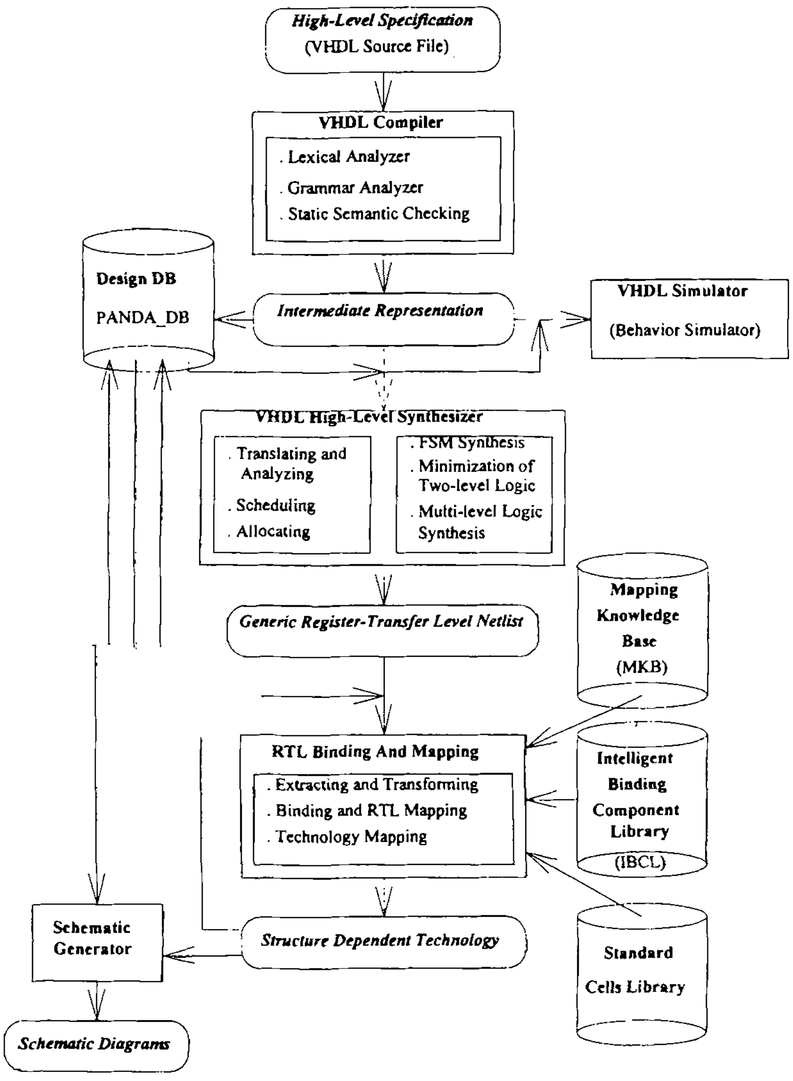
High-Level Specification
(VHDL Source File)
VHDL Compiler
. Lexical Analyzer
. Grammar Analyzer
. Static Semantic Checking
Design DB
PANDA_DB
Intermediate Representation
VHDL Simulator
(Behavior Simulator)
VHDL High-Level Synthesizer
. Translating and Analyzing
. Scheduling
. Allocating
. FSM Synthesis
. Minimization of Two-level Logic
. Multi-level Logic Synthesis
Generic Register-Transfer Level Netlist
Mapping Knowledge Base (MKB)
RTL Binding And Mapping
. Extracting and Transforming
. Binding and RTL Mapping
. Technology Mapping
Intelligent Binding Component Library (IBCL)
Schematic Generator
Structure Dependent Technology
Standard Cells Library
Schematic Diagrams

Fig.1. System architecture of HLS/BIT.

2 An Overview of VHDL High-Level Synthesis System HLS/BIT

Fig.1 shows the overall framework of the VHDL high-level synthesis system HLS/BIT that is mainly composed of five parts as follows.

(1) VHDL compiler

The VHDL compiler accepts high-level (algorithm level) design descriptions in VHDL. After lexical processing, grammatical analyzing and static semantic checking, the VHDL compiler produces the intermediate representation that is in BIF (Behavior Interface Format) form and is stored in the design database PANDA_DB.

(2) VHDL simulator

The VHDL simulator adds testing vectors to the information extracted from BIF in PANDA_DB and carries out simulation to verify the source behavioral design descriptions. Of course, it can be used to simulate the functions of any VHDL description.

(3) VHDL high-level synthesizer

The VHDL high-level synthesizer is composed of two parts: (a) data flow synthesis subsystem, and (b) control flow synthesis subsystem.

First, the synthesizer translates the intermediate representation into CDFG (Control/Data Flow Graph), and analyzes the data flow and control flow. On the basis of this, the data flow synthesis subtasks such as operator scheduling and component allocating progressively transform the abstract behavioral design specification to a netlist of generic register-transfer level (RTL) components. Simultaneously, a state sequencing table is created in terms of control states and state transitions.

Second, the control flow synthesis subsystem transfers the above state sequencing table into control logic through the FSM (Finite State Machine) synthesis, the minimization of two-level logic, and the multi-level logic synthesis.

(4) Binding and technology mapping subsystem

This subsystem accomplishes the RTL synthesis that changes the structure of generic RTL components to technology dependent structure and is divided into three processing phases:

(a) Extracting and transforming the generic RTL structure

First, the result of high-level synthesis, namely a netlist of generic RTL components, is extracted from the design database PANDA_DB.

Second, the above result is translated into a directed super graph of components.

(b) Binding and RTL technology-independent mapping

The tasks of this phase are to select components from IBCL (Intelligent Binding Component Library), to instantiate them using the constraints, and to bring out the technology-independent optimized structure.

(c) RTL technology mapping

The transformation from technology-independent structure to technology-specific one is finished in this phase using the knowledge-driven mapping method.

(5) Schematic generator

The schematic generator automatically generates schematic diagrams with the netlist information extracted from the results of synthesis.

When the VHDL high-level synthesis system HLS/BIT is designed and implemented, several different techniques and strategies have been accepted according to the features of problems in different processing stages. Most problems are definitive to be solved with the algorithm based method. But the RTL binding and technology mapping deal with many undefined problems and the knowledge-based method is suited to solve this kind of problem. Therefore, we introduce the following strategy: combining the algorithm-based method with knowledge-based method.

3 Intelligent Binding Component Library and Component Instantiation

Unlike basic logic components, register-transfer components might have many options. The result of the high level synthesis is an abstract generic RTL structure and needs to be translated into a concrete one using knowledge through RTL synthesis. This kind of translation must choose a component library as the objective. So we present the concept of intelligent binding component library (IBCL). IBCL provides some parameters related to functionality, electronic and geometrical properties for each component, including area, delay, number of bits, I/O ports and so on. IBCL is composed of two parts: (1) component library, and (2) some support facilities.

3.1 Component Library

Component library is the storage medium. There are four component types in component library: functional, storage, interconnection and miscellaneous. Component library describes each component with two aspects: design information stored in database format (a view of the design database PANDA_DB) and function information stored in UNIX file.

The design database of the system HLS/BIT is the database system PANDA_DB. Since the schema of PANDA_DB is a hierarchical structure with three storage levels: View-Element-Field, the description in database format for each component is constructed as follows:

```
View W_component
{
   version: VERSION_OBJECT; -version information
   property: W_PROPERTY; -overall information
   port: W_PORT; -I/O ports information
   function: W_FUNCTION; -function information
   parameter: W_PARAMETER; -parameter description
}
W_PROPERTY
{
   name: LONGSTRING; -component name
   type: SHORTSTRING; -component type
```

```
    portNum: INT; –number of ports
    funcNum: INT; –number of functions
    paraNum: INT; –number of parameters
    area: FLOAT; –area
    delay: FLOAT; –delay
    fileName: LONGSTRING; - name of the module file
}
W_PORT
{
    id: INT; –identifier
    name: LONGSTRING; –name
    inout: INOUT; –I/O type
    sigType: SIGTYPE; –signal type
    bits: INT; –number of bits
}
W_FUNCTION
{
    id: INT; –identifier
    name: LONGSTRING; –name
    condition: LONGSTRING; –condition
}
W_PARAMETER
{
    id: INT; –identifier
    name: LONGSTRING; –name
    desc: LONGSTRING; –specification
}
```

The description in the file format for each component is a UNIX text file of VHDL called module file.

Component library is a parameterizable library organized as a hierarchy of type attributes (the specification of abstract function), generator class (a family of similar components and instances) and components or instances (components instantiated).

3.2 Support Facilities

IBCL provides some facilities to manage, maintain and operate the component library, including:

(1) Knowledge acquisition mechanism

Users can implement the component library through the mechanism. The knowledge acquired includes two types: (a) component information, and (b) component instantiation rules.

(2) Component library manage and maintain mechanism

The mechanism is used to manage the component library and to assure its correctness, validity and integrity. The mechanism includes the following functions: (a) to operate the component library, for example add, modify, delete, retrieve or query certain component, (b) to check the validity and correctness of the component, and (c) to create and maintain the indexes.

(3) Intelligent instantiation mechanism

According to the information of the generic component and the constraints given by users, the mechanism implements the component instantiation process with parameters related to functionality, structural and operational properties and brings out an intermediate structure for technology mapping.

4 Knowledge-Driven Method for Technology Mapping

4.1 Mapping Strategies

The aim of technology mapping is to obtain a technology-specific structure. The kernel of technology mapping in high-level synthesis is mapping onto a technology cell library. The implementation ideas of technology mapping in HLS/BIT are as follows.

(1) Combining formalization algorithm with knowledge-based method

The knowledge is used to control the process of technology mapping and an efficient formalization algorithm is used to accomplish the concrete operations, such as substitution and modification.

(2) Embedding optimization in the process of mapping

Commonly, optimization is based on the factors such as area, load, delay and so on, and is finished by modifying the circuit and substituting some components, so it can be embedded in the process of mapping.

The way to do this is not translating the network of components for mapping into the knowledge format but rather controlling the process of mapping with the knowledge. The mapping operation is finished using formalization algorithms.

4.2 Mapping Knowledge

Mapping knowledge is acquired through interviews with expert designers. This knowledge can be classified into the following categories:

(1) Matching method selection

The matching method selection rules are used for selecting the formalization matching method.

(2) Constraint check

The constraint check rules are used for detecting and eliminating violations of design constraints such as area, delay and fanout and deciding how to meet them.

(3) Optimization

The optimization rules are used for removing redundant components or for replacing a group component with a single component.

(4) Function expansion

The function expansion rules are the knowledge about how to organize component using basic parts in order to implement its function and properties.

(5) Other rules

These rules are used to insert I/O and clock buffers and to connect unused pins of component to dummy cells which represent connections to the ground or pull-up circuits.

4.3 Implementation

Fig.2 gives the processing flow of technology mapping. Note that the knowledge is used to control the process of mapping.

```
Add initial facts to base and call initialization algorithm,
    then produce some new facts;
Add new facts to base
While ( NOT mapping finished)
    {
    Reason and start one formalization algorithm to operate using above
        facts, then produce other new facts;
    Add these new facts to base
    }
Produce ending facts and call the algorithm for output
```

Fig.2. The process of technology mapping.

In detail, the mapping is done in five stages:

(1) Add initial facts to base and call initialization algorithm to preprocess the components and the library cells, including setting the state of components and sorting the library cells.

(2) Select matching algorithm for mapping the component to library cells. The optimization rules might be used to call algorithms for removing the component or replacing the component group.

(3) Use the constraint check rules to call corresponding formalization algorithm to improve mapping results.

(4) If design constraints are met, the function expansion rules are used to call the algorithm for expanding the macro component.

(5) Output the results, including some files and the netlist.

4.4 Comparison

Compared with single-formalization method, it is clear that the above new mapping method has some strong points:

(1) The mapping method makes it easy to add new design knowledge to adapt the system to different technologies.

(2) For algorithm-based method, the mapping is accomplished by restricting the components to a limited set of alternatives. In this way the constraints and the specialized knowledge about design trade-offs are hard-wired into the mapping process. So the flexibility of the system is poor. The knowledge-driven mapping method is more flexible and is inexpensive to upgrade.

(3) Using the knowledge-driven mapping method, the ripe experience of designers can be acquired as rules which will expand or update the knowledge base.

5 Status and Future Direction

We are implementing the knowledge-based approaches and strategies used in RTL binding and technology mapping of VHDL high-level synthesis system HLS/BIT in wider areas. For future research, the size of parametrization of IBCL, the acquisition of design knowledge and the management of the knowledge base will be investigated.

References

[1] Liu Mingye , Guo Shuming. Intelligent problems of design automation system. *Journal of Computer Aided Design & Graphics,* 1989, 1(2): 66-72.

[2] Chen Gwodong, Gajski D D. An intelligent component database for behavioral synthesis. In *Proc. of 27th DAC,* 1990, pp.150-155.

[3] Nikil D Dutt, James R Kipps. Bridging high-level synthesis to RTL technology libraries. In *Proc. of 28th DAC,* 1991, pp.526-529.

[4] Berkelaar M R C M, Jess J A G. Technology mapping for standard-cell generators. In *ICCAD,* Nov., 1988 pp.470-473.

Yan Zongfu received his B.S. degree in computer software from Xi'an Jiaotong University in 1989, and his M.S. degree in computer engineering from People's University of China in 1992. He is now a Ph.D candidate in computer engineering at the Beijing Institute of Technology. His current research interests include EDA, knowledge-based system and database system.

Liu Mingye is currently a Professor of computer engineering at Beijing Institute of Technology. His research interests are intelligent CAD, ASIC design and VLSI design automation.