xTractor: An Academic High-Level Synthesis Tool for Control and Memory Intensive Applications

Peeter Ellervee

Department of Computer Engineering, Tallinn University of Technology,
Raja 15, 12618 Tallinn, Estonia
phone: +372 620 2258, fax: +372 620 2246, e-mail: lrv@cc.ttu.ee

1. Motivation

The introduction of ***high-level synthesis*** (HLS), a.k.a ***behavioral synthesis***, promised to automate the transformation of a design from system/behavioral level to register-transfer (RT) level as efficiently as the introduction of logic synthesis automated transformation from logic to physical level. Although HLS has been successfully used in many cases and there exist rather many HLS tools, it is still not as indispensable today as layout or logic synthesis. Experiments with designs that are dominated not by the data flow but ***control flow and data transfers*** have pointed out that the traditional data flow oriented synthesis strategy does not work well. A different approach is needed which would take into account the main characteristics of ***control and memory intensive systems*** (CMIST).

The newest developments both in design automation and design for testability areas are pointing out the need for tools that are ***modifiable*** to add new sub-tasks and even to integrate tasks that traditionally have been solved separately. Also, the existing commercial tools are not always good for academic activities because it is very hard, if not impossible, to get into details of the synthesis process:

- there must exist a way to show to the students how a HLS tool works step-by-step; and
- researchers need rather often a possibility to modify an algorithm in the synthesis process, or even to modify the execution order of steps.

xTractor, has been developed to test HLS methodology of CMIST applications. The nature of CMIS applications defined the overall synthesis flow and transformations needed to convert a behavioral description of a design into RTL description. [1][2][3]

2. Synthesis Flow

IRSYD Generator - compiles a subset of VHDL or C into CDFG. Only bit-vector like data types are allowed.

Data-Path Transformations - only the most obvious ones have been implemented - *constant and variable propagation*, and *simplification of operations with constants*.

Memory Extractor lists arrays and/or maps them onto memories, and adds memory access operations.

State Marker generates states while traversing the control flow of the IRSYD. It uses segment-based scheduling.

Allocator/Binder allocates and binds operations and variables into functional units and registers.

RTL HDL Generator - generates RT level VHDL or Verilog code for Logic Level synthesis tools.

There exist four pre-defined synthesis flow styles with different degrees of designer activity.

3. Target Architecture

Modern logic synthesis tools can handle rather complex descriptions but for fast and efficient synthesis the different styles should be segregated. CMIST applications have very few large operations worth of reusing. The rest of the design consists of operations that can be very efficiently optimized by logic optimization techniques. Keeping arithmetic operations free of implementation details allows a better exploitation of the back-end (logic synthesis) tools.

Four different architectures, selected by the designer, can be generated:

- merged FSM and DP - data-path is a part of the FSM;
- separate FSM and DP - the traditional architecture;
- separate FSM and DP, large functional units extracted - allows to optimize arithmetic units separately; and
- separate FSM and sliced DP, large functional units extracted - the slicing allows to divide large DP into smaller parts to simplify back-end logic optimization.

4. Structure and Component Tools

xTractor consists of an interactive shell, written in Tcl/Tk, and component tools, written in C/C++, can be ported to several platforms (e.g., Linux, Solaris). The shell organizes the overall synthesis flow.

The eight component tools execute the steps of the synthesis flow and/or analyze CDFG (estimation, integrity checking, etc.). Additional tools can be added by modifying setup file(s) of the shell.

5. References

[1] P. Ellervee, “High-Level Synthesis of Control and Memory Intensive Applications.” Ph.D. Thesis ISRN KTH/ESD/AVH--2000/1--SE, Stockholm, 2000.

[2] P. Ellervee, “xTractor: An Academic High-Level Synthesis Tool for Control and Memory Intensive Applications.” The 20th NORCHIP Conference, Copenhagen, Denmark, pp.,253-258, Nov. 2002.

[3] “xTractor,” URL: “http://mini.li.ttu.ee/~lrv/xtractor/”