Università degli Studi di Verona, Facoltà di Scienze MM. FF. NN.
Dipartimento di Informatica
EDALab: Embedded System Design Center

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

Verona, 02/03/2007

Automatic Synthesis from VHDL

Nicola Bombieri

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

1 Required Background

Students interested in learning automatic HDL synthesis are required to know the fundamentals of VHDL language. Moreover, it is advisable that students are familiar with concepts of combinational and sequential digital systems.

2 Goal

The goal of this lecture consists of analyzing the key concepts of digital circuit synthesis, that is, the translation from RT level towards gate-level. The main steps are described by using one of the more important commercial tools for the HDL synthesis.

Students will learn to:

- Analyze several ways to implement a digital circuit behavior at RTL;
- Synthesize RTL descriptions by setting the implementation details dependent on the architectural choices.

3 Introduction

After a system description has passed the verification phase by means of dynamic and/or static verification, it is ready to undergo the synthesis process. Figure 1 shows the whole design process and it underlines the exact point in which the synthesis task is set up.

The RTL device implementation is translated into a set of elementary blocks at gate-level. Since several architectures are available, the same behavioral description can be translated to different gate-level implementations. Software tools aid designers in this task, by offering the opportunity to automatically synthesize the RTL code with regard to the chosen target architecture.

Due to the difficulty in interpreting the full VHDL semantic, the VHDL syntax accepted by the synthesis tools is often reduced to a subset of synthesizable statements (i.e., FOR loop, wait condition, etc.). For this reason, one step of *behavioral synthesis* could be necessary before synthesize the circuit towards the gate level (see Figure 1).

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

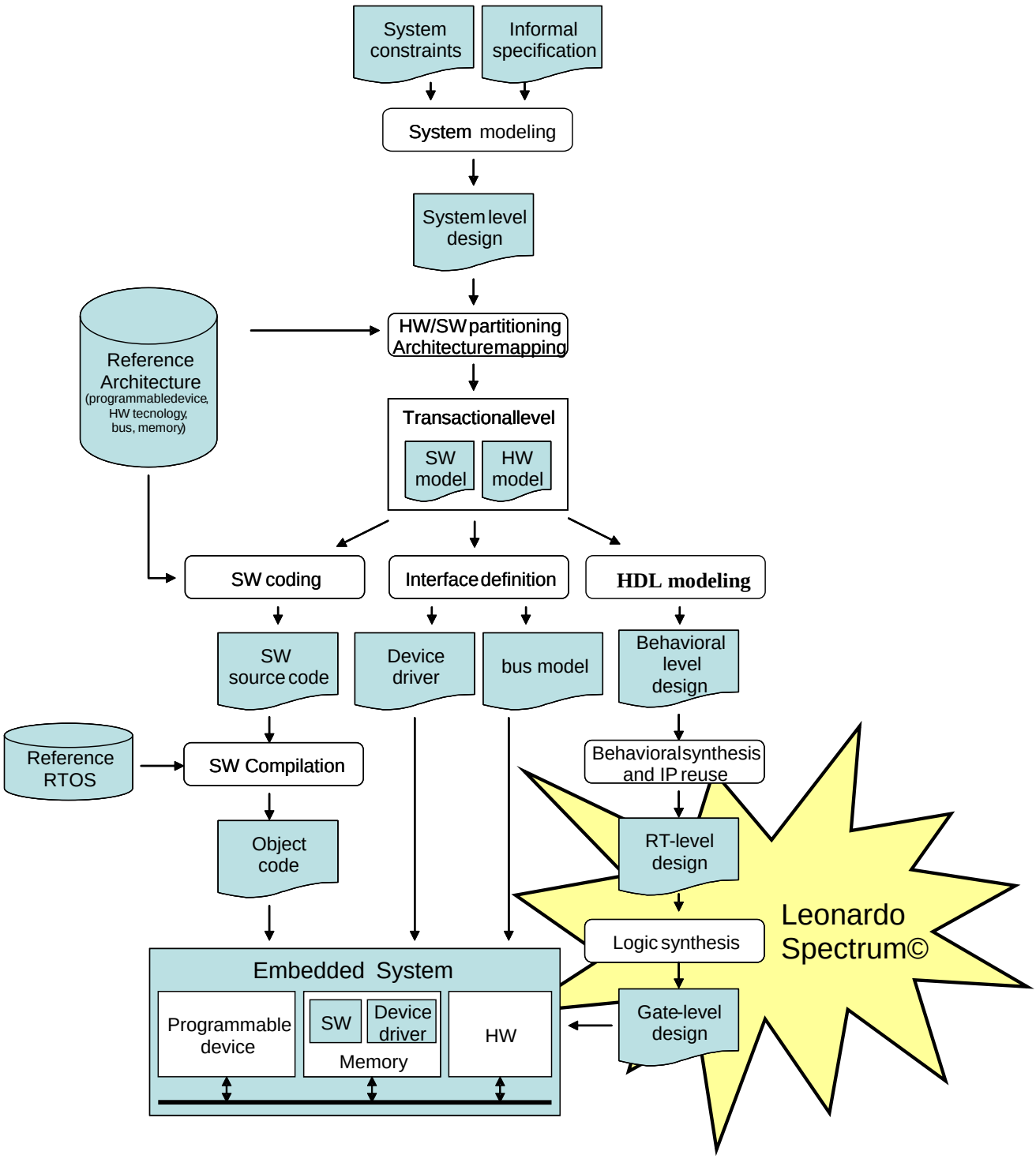

Figure 1: Embedded system design flow.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

4 LeonardoSpectrum

LeonardoSpectrum by Mentor Graphics is one of the more widespread tool for the automatic synthesis task. It drives designers through the architectural choices by means of a simple graphical interface.

In the following, a short guideline will be proposed for every step. The *adpcm* module of the V-CLIP system [1] will be used as case study.

4.1 Technology

Setting the advanced *FlowTabs* from the *Tools* menu, the designer starts the guided procedure of the synthesis.

Application Specific Integrated Circuit (ASIC) vs. Field Programmable Gate Array (FPGA) is the first choice, that is, the technology in which the device will be mapped. Deciding between ASICs and FPGAs requires designers to answer tough questions concerning costs, tool availability and effectiveness, as well as how best to present the information to management to guarantee support throughout the design process.

ASIC is generally adopted to implement a specific computational application and offers the best performances. On the other hand, its cost is extremely high.

FPGA is released as a standard product for general purposes. Its final implementation is defined by the user. The programmability feature is the main advantage w.r.t. ASIC. For example, in case when a bug is found during the verification phase, the block can be re-programmed and the bug fixed as a consequence.

The technology we are going to select for the adpcm module is an FPGA by Xilinx and the board feature are the following:

- Board: SPARTAN3;
- Device: 3S200ft256

Clicking on *Load Library*, the right window reports us if the library loading process is successfully completed.

4.2 Input

Selecting the *Working Directory*, the file (*adpcm_RTL.vhd*) is ready to be open. LeonardoSpectrum inputs the design when the designer clicks on the *Read* button. The read is accomplished in two phases:

1. *analyze*: the HDL is syntactically checked, dependencies are checked and generic parameters are resolved.
2. *elaborate:* during this phase LeonardoSpectrum synthesizes the HDL into an EDIF-like in-memory database. The design is composed of generic gates and

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

black-box operators. Later the black box operators will be replaced with efficient technology-specific operators from a vendor-supplied library.

4.3 Constraints

Designers specify the clock frequency, clock cycle, and global path constraints for the whole design. Hence, the smallest design for the give frequency is then created. All path between ports and registers are constrained to one clock period. Specific delays can be customized by specifying a maximum delay between each. The clock reference time is zero.

We set 100 MHz as clock frequency and then we apply the setting.

4.4 Optimize

There are two major optimization functions within LeonardoSpctrum. The optimize command performs global area optimization and may be run with or without timing constraint. Running without timing constraint produces the smallest area design. Global optimization is run on each module in the design hierarchy (if it exists) separately. Normally, LeonardoSpectrum has four groups of algorithms that can be run on each module.

The optimization level of effort can be set: the selected effort is traduced in time spent to optimize the synthesis with the selected constraint.

Students should try to synthesize the adpcm module, firstly imposing the maximum effort to optimized the area constraint and then to optimized the delay constraint. Students should compare the results to note the translation differences of the same starting block.

4.5 Report Timing and Examine Results

After the designer run timing optimization, he can generate a timing report. Unlike optimization algorithm, timing reports display timing paths through hierarchical boundaries.

If the design is still not meeting performance requirements, the designer can either loosen the timing constraints or run again timing optimization. If he is within 10% of his timing goals, he may want to run the design through the place/route tools to get the accurate post-place/route timing value. Leonardo calculates delays using a conservatives pre-layout estimate. If the designer is close to meet his timing constraint (according with Leonardo estimate), he may have actually met the timing constraint according to the accurate post-layout value produced by he place/route tool.

Specify a report file name (.txt) for both the area report and delay report.

4.6 Saving the design

After the design meets the area and timing constraints, the designer can write out the design as a netlist file. This netlist file can be read by a downstream tool (such as place/route tool) or read back into LeonardoSpectrum as a hierarchical block in a larger design.

Specify a filename, a format (VHDL) and click on *Write* to create the output result.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

5 References

[1] EDALab “V-CLIP: A Voice-Client over IP”, http://www.edalab.net.