Synthesis of Communicating Controllers for Concurrent Hardware/Software Systems

R. Niemann
Dept. of Computer Science XII
University of Dortmund
Dortmund, Germany

P. Marwedel
Dept. of Computer Science XII
University of Dortmund
Dortmund, Germany

Abstract

Two main aspects in hardware/software co-design are hardware/software partitioning and co-synthesis. Most co-design approaches work only on one of these problems. In this paper, an approach coupling hardware/software partitioning and *co-synthesis will be presented, working fully-automatic. The techniques have been integrated in the co-design tool* COOL[1] *supporting the complete design flow from system specification to board-level implementation for multi-processor and multi-ASIC target architectures for data-flow dominated applications.*

1 Introduction

Most approaches in co-design work either on techniques for partitioning or co-synthesis using manual partitioning. Hardware/software partitioning has been explored intensively, using different algorithms to solve this complex optimization problem, e.g. dynamic programming, evolutionary strategies, mixed integer linear programming or several heuristic approaches. Techniques for interface and communication synthesis have been explored for example in CHINOOK [1] or COSMOS [2]. In both approaches, the systems are partitioned manually. This paper presents an approach for a coupled hardware/software partitioning and co-synthesis algorithm. It is based on the synthesis of a *system controller* (similar to [3]) steering all processors and ASICs according to the computed schedule. All techniques have been implemented in the co-design tool COOL.

2 The COOL framework

The codesign tool COOL is a framework for implementing hardware/software systems starting from a unified, implementation independent system level description. The way of specifying systems (based on a subset of `VHDL`) and the partitioning algorithms have been described in detail in [4]. Partitioning is either based on *mixed integer linear programming* (MILP), a combination of MILP and a heuristic, or on *genetic algorithms*. This paper describes the design steps in COOL after partitioning has finished.

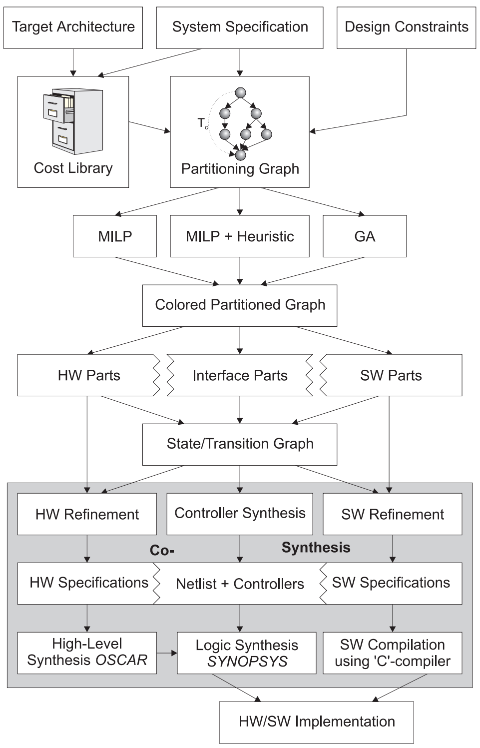

Figure 1: Design Flow in COOL

The result of the partitioning phase is (1) a coloured partitioning graph where each colour either represents a hardware or software resource and (2) a static schedule as depicted in figure 2. The goal of the co-synthesis approach in COOL is to generate synthesizable hardware specifications in `VHDL` and software specifications for compilation in `C`. In a first step, a *state/transition graph* (STG) is generated rep-

[1] COOL: COdesign toOL

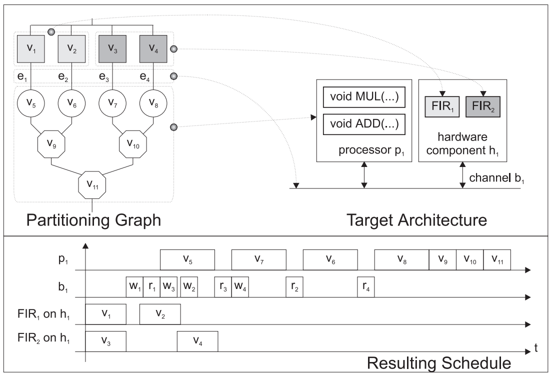

Figure 2: Partitioning Graph for a 4-band equalizer

resenting the fundamental data structure during co-synthesis. This graph is computed by adding a WAIT-(w), an EXECUTION-(x) and a DONE-state (d) for each node of the coloured partitioning graph to the STG. These states represent the execution order of a function steered by the system controller. In addition, RESET-states (r) are inserted for each hardware resource and processor and global system states (X, R, D) are added. Edges are added according to the computed schedule and the data dependencies. After the number of states of the STG has been minimized, memory cells are allocated (starting from a `base` address) for each edge representing a data transfer between different processing units. The result is depicted in figure 3.

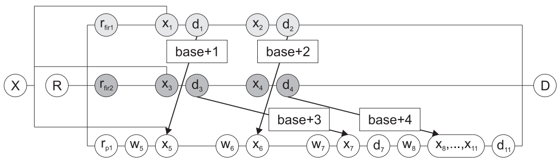

Figure 3: STG and Memory Allocation

Then, the original specifications of the hardware and software parts are refined using the information annotated at the STG. Communication mechanisms for *memory mapped I/O* and *direct communication* are inserted to replace the abstract communication channels. To implement a complete hardware/software system, additional parts are required: the *system controller*, steering the complete system according to the computed schedule, *data path controllers* to support hardware sharing, an *I/O controller* to communicate with the environment and *bus arbiters* to prevent conflicts. These additional pieces will be implemented in hardware, because hardware allows concurrent processes. COOL generates `VHDL`-specifications for all

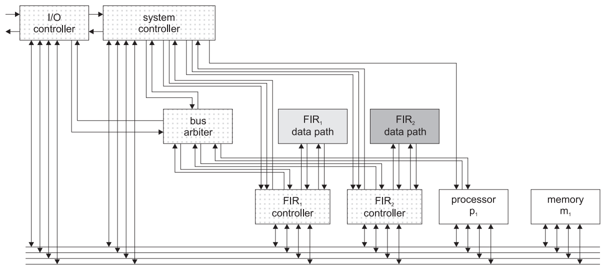

Figure 4: Generated Netlist

these additional pieces and a net-list wiring all them as depicted in figure 4. Then, these `VHDL`-specifications are synthesized by the high-level synthesis tool OSCAR and the commercial synthesis tool SYNOPSYS.

3 Results

COOL was used during a student project [5] dealing with the design of a *fuzzy controller*. This fuzzy system was specified with COOL (about 900 lines of code) resulting in a partitioning graph containing 31 nodes. This specification was implemented on a target architecture containing a Motorola DSP56001 placed on a plug-in card in a PC and two Xilinx FPGAs 4005 (with 196 CLBs each) on a board. In addition, a memory card with 64kB static RAM was build and all components were connected by a bus card. Different hardware/software partitions of the fuzzy controller were implemented and in all cases the time to execute the complete design flow from system specification to an implementation on the prototyping board took not more than about 60 minutes. The time-consuming factor was always the hardware synthesis which consumed more than 90% of the design time.

References

[1] P. Chou, R.B. Ortega and G. Borriello. *The Chinook Hardware/Software Co-Synthesis System*. ISSS, 1995.

[2] T.B. Ismail and A.A. Jerraya. *Synthesis Steps and Design Models for Codesign*. IEEE Computer, 1995.

[3] V. Mooney, T. Sakamoto and G. De Micheli. *Run-Time Scheduler Synthesis For Hardware-Software Systems and Application to Robot Control Design*. 5th Int. Workshop on HW/SW Codesign, 1997.

[4] R. Niemann, and P. Marwedel. *An Algorithm for Hardware/Software Partitioning using Mixed Integer Linear Programming*. Design Automation for Embedded Systems, pp. 165–193, 1997.

[5] S. Bashford, B. Landwehr, R. Niemann et al. *Endbericht der Projektgruppe 293*. Forschungsbericht, Universität Dortmund, 1997.