FPGA Based Prototyping for Verification and Evaluation in Hardware-Software Cosynthesis

Th.Benner, R.Ernst, I.Könenkamp, U.Holtmann,
P.Schüler, H.-C.Schaub and N.Serafimov

Technische Universität Braunschweig
Institut fuer Datenverarbeitungsanlagen
D–38106 Braunschweig, Germany

Abstract. COSYMA is a HW/SW-cosynthesis system for small embedded controllers. The final simulation of the COSYMA output leads to impractical computation time. Therefore, we decided to employ a hardware prototyping system. The HW/SW prototyping system consists of a SPARC processor, an FPGA-based coprocessor with HW/SW debugging features realized with a high speed microcontroller.

1 Introduction

COSYMA (CoSYnthesis of embedded Micro Architectures) is one of the first systems for hardware-software cosynthesis [1]. It is targeted to the design of small embedded controllers. Given an input description in a superset of C, C^x, consisting of one or more tasks with time constraints, and given a fixed core processor, COSYMA tries to map as much of the system as possible to software. When the time constraints cannot be met, it automatically partitions parts of the system description to an application specific coprocessor such that the timing constraints are met with minimum hardware overhead. The partitioning process regards and minimizes communication overhead. Significant speedups have been observed for real examples with more than 1000 lines of C code and a 33MHz SPARC as core processor. Figure 1 outlines the design flow. Partitioning is done on the basic block level using simulated annealing based on speedup, communication time and hardware overhead estimation. The software part is enhanced by a communication protocol, translated to C and compiled to object code. The hardware part is generated by high-level synthesis. The approach is flexible enough to permit the use of different high-level synthesis systems; currently these are OLYMPUS [2] from Stanford taking a HardwareC description as input and our own synthesis system BSS [3] taking a CDFG as input. A run time analysis is able to accurately estimate the execution time of the resulting hardware-software system on a clock cycle basis [4] and thus show if the timing constraints are met. Currently, we are working on an approach to adapt the estimation in hardware-software partitioning to the actual results. Once the timing constraints are met, the high-level synthesis system generates a hierarchical netlist on the logic level consisting of the controller and the data path in SLIF format (Stanford Intermediate Logic Format). In order to verify this netlist, we

currently use HW/SW cosimulation. It turned out that logic level simulation of the coprocessor and the communication hardware is only possible for small examples in acceptable computation time, even though the processor is simulated at the register transfer level. So, we decided that for hardware-software co-verification a hardware prototyping system would be very helpful.

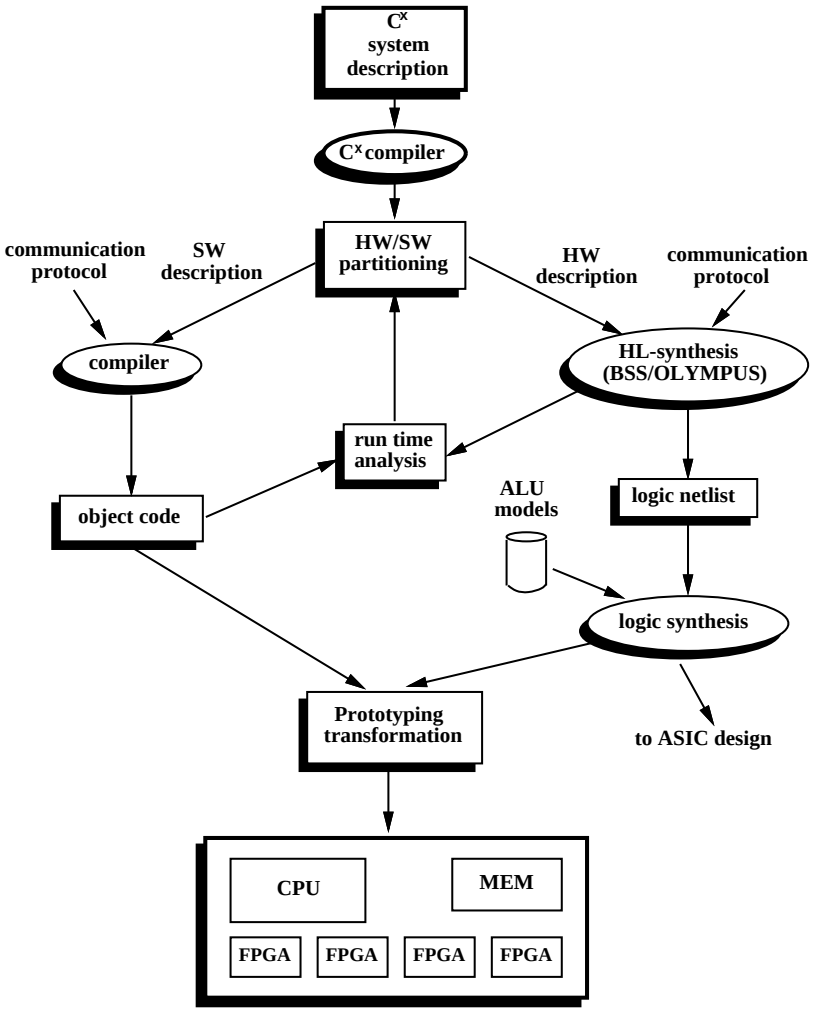

Fig. 1. The COSYMA experimental system

2 Prototyping System Requirements

Because the generated hardware-software system uses memory coupling, it seemed reasonable not to use a very expensive commercial prototyping system, which would have to be extended by a SPARC core, anyway, but to develop a specialized system. Because the coprocessor typically is rather small, we concluded that it should be possible to design the hardware-software prototyping system to run close to real time (33MHz SPARC clock) and so even use it for system evaluation in the real environment. Furthermore, symbolic debugging features with software breakpoints were considered essential for COSYMA verification. A major requirement was that the whole design flow from the high-level synthesis output should work without manual interaction.

3 The Architecture of the Prototyping System

The prototyping system consists of three different boards. Figure 2 gives an overview of the system. The LSI SPARC processor ([5]) and the coprocessor board are memory coupled by RAM A and RAM B of the coprocessor board. The arbiter separates the processor bus from the system bus, which enables the processor to fetch an instruction from its instruction RAM, while the coprocessor is fetching one or two 32 bit words in the same clock cycle.

The FPGA board contains four XILINX XC4010 ([6]). We chose the FPGA family, because of its sizes. For multiplications, we added an AMD 29C323.

For the purpose of time measurement, debugging and the connection to the host computer, a Motorola MC 68332 ([7]) is used.

3.1 The FPGA Board

The major function of the prototyping system is the emulation of an application specific coprocessor. A typical design generated by COSYMA could include up to five ALUs and 20 registers. This exceeds the capacity of one single FPGA. So, the design was partitioned into clusters. With up to 5 arbitrarily connected 32 bit ALUs and 10 more more registers, we would quickly run out of pins when allocating complete ALUs and registers to FPGAs. Instead we chose a bit slice architecture (fig. 3). The FPGA board consists of four XILINX XC4010, each handling an eight bit slice of all ALUs and registers. So, every FPGA holds up to five 8-Bit-ALU slices and an application dependent number of 8 bit register slices.

In order to handle the large number of control signals, a copy of the whole controller is included in each slice. Therefore it is ensured that all important control signals are generated on each slice. The carry signal of the ALUs is handled by a carry look ahead logic on one of the slices (no. 3). Additional flags for comparison decrease the controller overhead.

On every slice there are two RAM ports. Each of the ports is able to read or write 8 bits of the 32 bit word. Therefore, the control signals are generated on

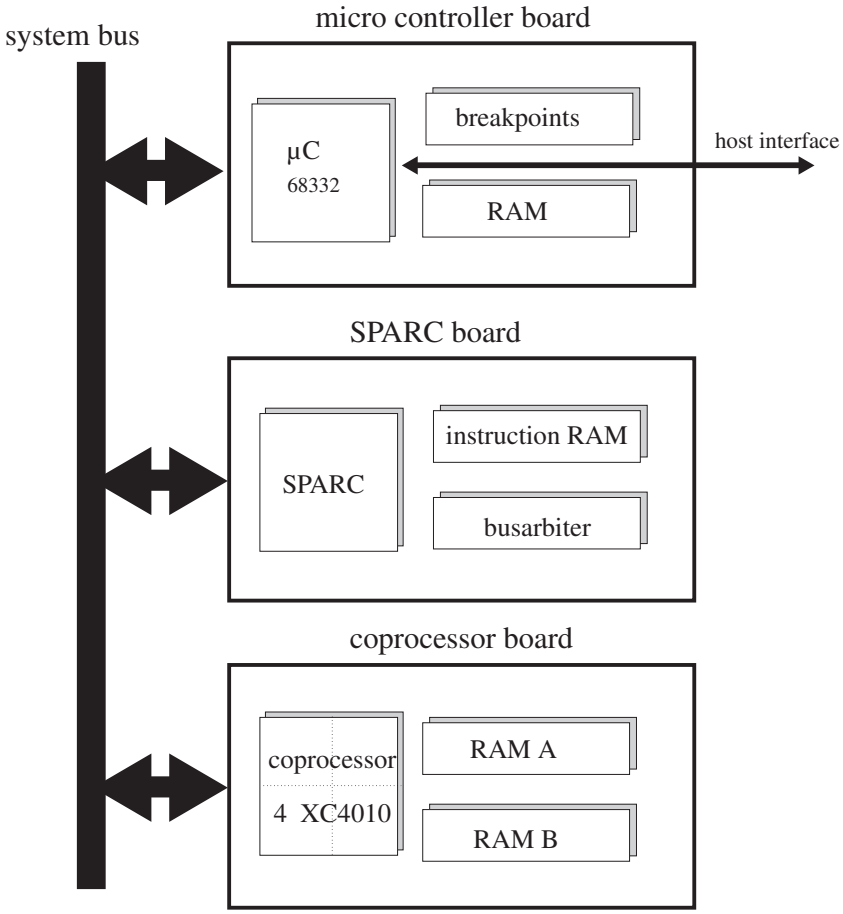

Fig. 2. The architecture of the prototyping system

one of the slices (no. 2). The memory consists of two banks, whose address spaces can be mapped in two different ways. The address space of port B can be mapped following port A. Another possibility is to address port A and port B alternately in steps of four bytes for interleaving. The address modes are configured by the micro controller.

3.2 Communication Between the SPARC and the FPGA Boards

Currently, processor and coprocessor communicate through shared memory with mutual exclusive access. In order to increase the parallelism in the future, every slice uses two separate memory ports. The coprocessor is able to fetch two 32

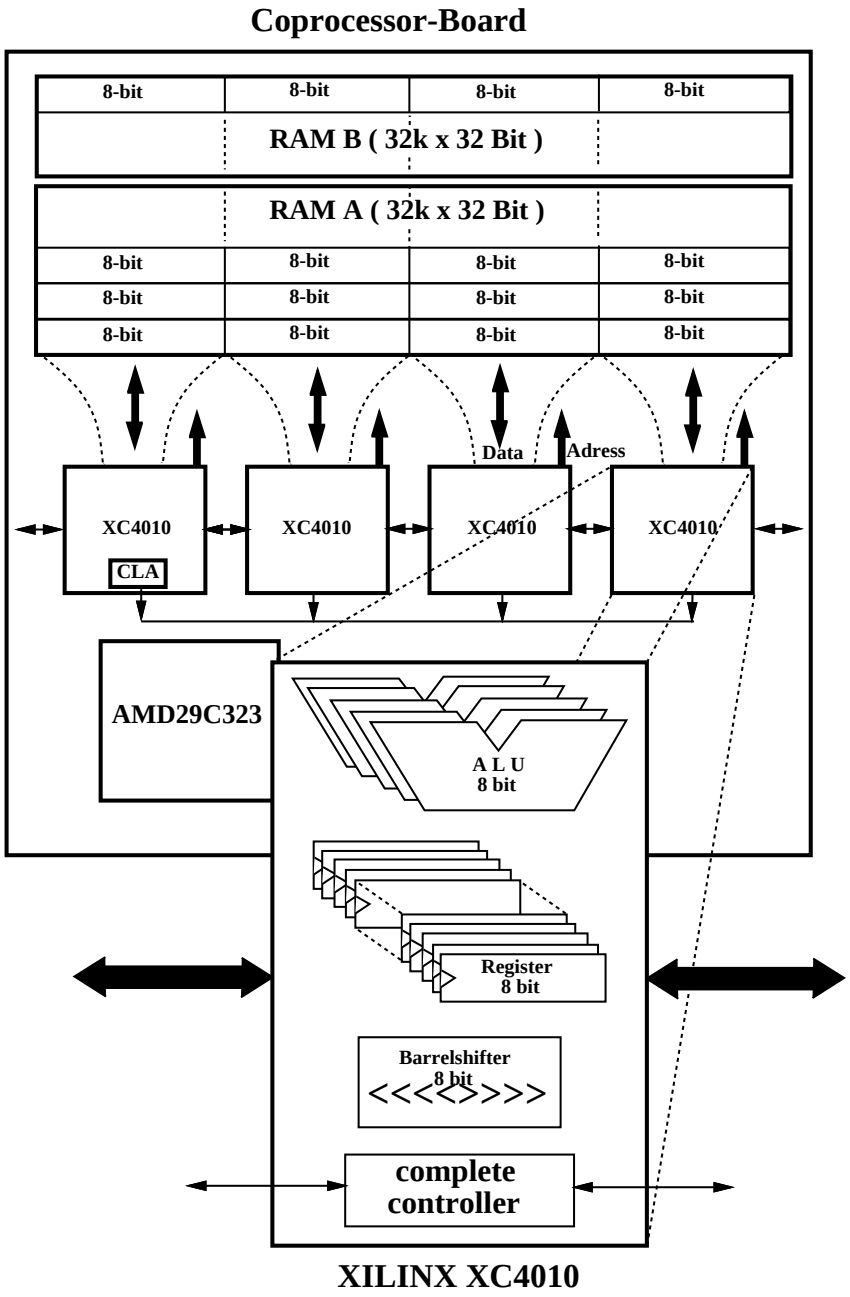

Fig. 3. Architecture of the FPGA board

bit values concurrently. The two memory banks are mapped into the SPARC address space.

Up to now, the SPARC receives a HOLD signal when the coprocessor is active. We are going to increase the performance of the design by allowing parallel execution in the SPARC and the coprocessor. Therefore, a separate instruction memory is inserted, allowing the SPARC to fetch an instruction during a data fetch of the coprocessor.

3.3 Time Measurement and Debugging

As mentioned above, a microcontroller is used for time measurement and debugging. One interrupt channel of the MC68332 is triggered by a tag RAM, which contains the addresses of up to eight breakpoints. Whenever one of eight maskable tag breakpoint appears on the address bus, the microcontroller receives an interrupt while processor and coprocessor are stopped in the same cycle. Then, user is able to read the memory map.

The hardware breakpoint are also used for the time measurement. One of the MC68332 timers is triggered by the tag RAM. Each can be used either for time measurement or for debugging. The state of the system bus during a breakpoint event is stored in a register which can be read by the micro controller.

4 Design Flow

The result of the partitioning process in COSYMA consists of software for a processor core, and a coprocessor. For the software parts, COSYMA generates ANSI-C code which is then compiled.

As an output of high-level synthesis ([2],[3]), the coprocessor is described as a logic level netlist in SLIF format. This netlist is a technology independent description which has to be partitioned into the four bit-slices (fig. 4). Each slice uses a local copy of the hierarchical netlist description where all bits belonging to different slices are removed. This is currently done on text level by a script written in "perl". While this works well for the data path, special care has to be taken not to remove signals entering or leaving the controller or crossing bit-slices.

The description of modules like ALUs, RAMs, ... is extended by those signals necessary for the interconnection of the bit-slices. This happens already before partitioning. Because of a predefined set of interconnections, only a fixed set of signals (5) from data path is available in the controller.

After a translation to Verilog, logic optimization is done by the Synopsys system on each of the FPGA-netlists separately. At this point VHDL models of the ALUs and the barrel shifter are included. The synthesized slices are mapped to the FPGAs. The output of the Xilinx tools is a bitstream, which is then downloaded to the FPGAs controlled by the MC68332.

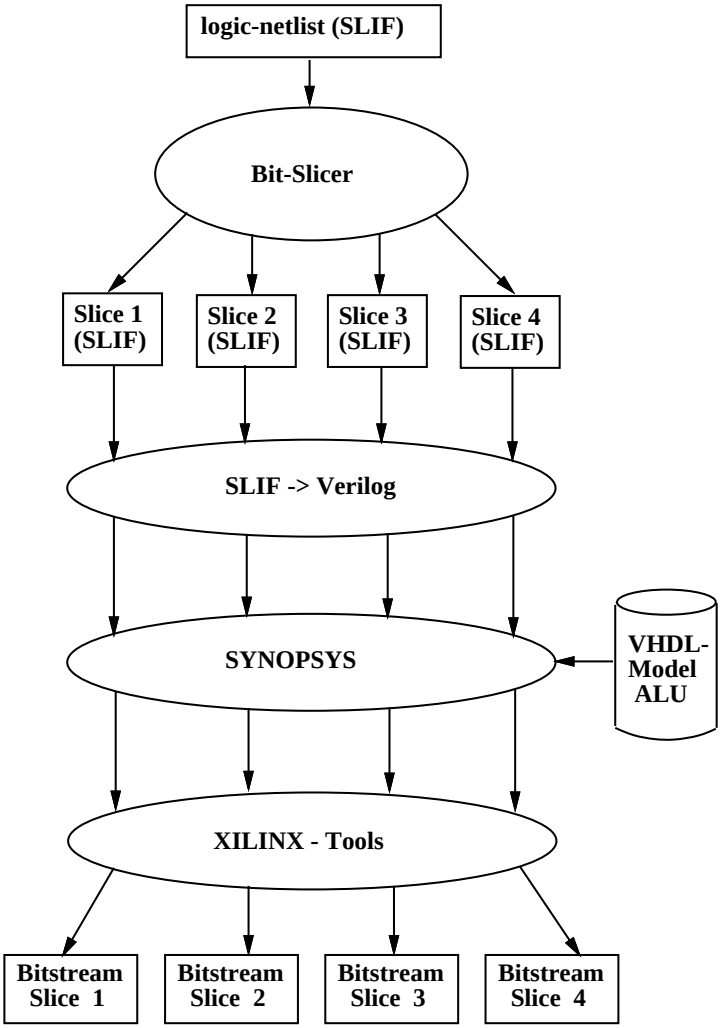

Fig. 4. The design flow

5 Future Directions

Now, the system is completed as wire-wrap prototype, which is clocked at 10 MHz. We are going to build a 30 MHz version using multilayer PCBs. One problem of the system is the capacity of the XILINX 4010. Using four XC 4025 instead of XC 4010, an internal multiplier could be realized on the chips and a large no. of registers.

We are going to extend COSYMA to small heterogeneous multiprocessor systems. As a first extension, the Motorola DSP 96002 will be added to the prototyping system. The DSP-Board will be prototyped using Aptix interconnection matrixes.

6 Acknowledgement

J. Herrmann and P. Meier were responsible for the mechanical construction of the prototyping system and H.-O. Leilich and F. Rabe gave very helpful hints when we debugged the hardware.

References

[1] R. Ernst, J. Henkel and Th. Benner, Hardware/Software Co-Synthesis for Microcontrollers, IEEE Design & Test of Computers, pp. 64–75, Dec. 1993.
[2] G. De Micheli et al., The Olympus Synthesis System, IEEE Design & Test of Computers, pp. 37–53, Oct. 1990.
[3] U. Holtmann, Hierarchical Behavioural Representation in the Braunschweig Synthesis System BSS, IFIP Workshop on Application of Synthesis and Simulation, Lenggries, 25-28.8.1993.
[4] W. Ye, R. Ernst, Th. Benner, J. Henkel, Fast Timing Analysis for Hardware-Software Co-Synthesis, Proc. of ICCD 1993, Cambridge, pp. 452–457 , 1993.
[5] LSI Logic Corporation, L64831 - SPARC Integrated IU/FPU Technical Manual, Milpitas, Calif. 1992.
[6] Xilinx, Inc., The Programmable Logic Data Book, San Jose, Calif., 1993.
[7] Motorola Inc., MC68332 User's Manual, Phoenix, Arizona, 1990.

This article was processed using the LaTeX macro package with LLNCS style