An Approach for Interface Generation in the PISH Co-design System

Cristiano C. Araújo and Edna Barros
Departamento de Informática, UFPE
Caixa Postal 7851 – Cidade Universitária
50732-970 Recife – PE Brazil
cca2, ensb@di.ufpe.br

Abstract

This paper describes a methodology for interface generation in the PISH co-design system, which allows the hardware/software co-design of concurrent processes described in occam. The hardware/software partitioning generates a description including processes to be implemented in software, processes to run in hardware and processes for communication purposes. This new description is generated by applying algebraic transformation rules according the results of a cost analysis based on clustering techniques. This strategy preserves the semantics of the initial description. All of these are generated from the initial description in a constructive and correct way.

1. Introduction

With the growing complexity of the digital systems and the need for reducing the time to market, techniques for supporting hardware /soft-ware co-design have been developed in order to permit the joint specification, design and synthesis of mixed hardware/software systems [5][12]. Such systems consists of common-off-the-shelf (COTS) and ASIC components and have a variety of implementation technologies and interfaces, and a wide range of real-time data rates. The need for early prototypes to validate the specification and to provide the customer with feedback during the design process is another key factor motivating hardware/software co-design.

Some tools and methodologies supporting hardware/software co-design have been published in the last years [5] [6] [7][8][9][10] [12]. In most of them, however, once the initial description was partitioned, the interface between the hardware and the software components is synthesized by hand or in a semi-automated way.

This work takes into account the PISH co-design system, which allows the partitioning of occam descriptions by considering hardware/ software trade-off but also distinct hardware implementations [11]. Additionally, the correctness of the partitioning process can be assured through the use of formal verification techniques, in a constructive way [12][13] and a virtual prototype can be obtained in an early phase of the design process. The partitioning output is a set of communicating processes, some of them to be implemented in hardware, others in software and others for communication purposes. The next step is the generation of a real prototype, a very time consuming and error prone activity, and in the PISH co-design system it has been done by hand.

The complexity of the interface generation depends on the flexibility of the target architecture. The most systems with automatic partitioning taken into account a pre-defined target architecture, which makes the interface generation easier. But also in this case, automatic interface generation is not easy due to the semantic gap between the descriptions of the virtual and the real prototypes. Due to this fact, techniques for automatic interface generation is a feature of a small number of co-design systems [2][3][16].

An important support for interface generation is the communication generation during the partitioning process. When the communication among modules is made explicit and assured to be correct, the mapping of the virtual prototype into the real one can be done in a most natural manner.

The main goal of this work is the design of techniques for interface generation in the PISH co-design system. Particularly, this work focuses techniques for generating the interface between hardware and software. This paper is organized as follows: the next section gives an overview of the PISH co-design system including interface generation. A more detailed description of the proposed approach is given in section 3. Section 4 illustrates an example. Some conclusions are presented in section 5.

2. Interface Generation in the PISH Co-design System

The approach for automatic interface generation is being developed in the context of the PISH co-design

system. This system uses occam as specification mechanism [14] as depicted in Figure 1.

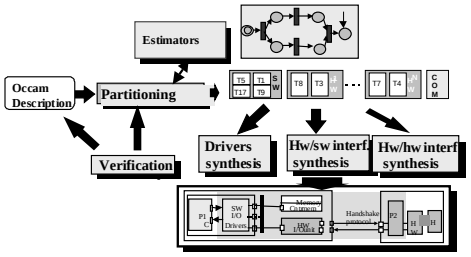

Figure 1- The PISH Design Flow

The main reason to use occam is that, being based on CSP [15] occam has a simple and a elegant semantics, given in terms of algebraic laws, which allows the partitioning be performed by applying a series of algebraic transformations into the initial occam description in order to preserve the semantics. The set of transformation rules is applied according to the results of a cost analysis based on clustering techniques [11]. The output of the partitioning is a set of concurrent processes, which communicates through processes generated only for this purpose. This feature is a important support for the interface generation, since the communication among processes has been made explicit and is correct. The interface generation depends on the target architecture taken into account and the most co-design systems considers a very simple architecture composed of one software component. In order to have a pre-defined protocol some systems considers the hardware running as a co-processor, i.e. hardware and software do not execute concurrently [1][10].

In this work, software and hardware can run concurrently and for that device drivers must be generated at the software side, as well as specific hardware to make transparent for the hardware side which processor is being used. The interface between hardware modules must also be synthesized.

3. A Methodology for Interface Generation

The proposed methodology for interface generation takes into account a pre-defined target architecture composed of one software component, but with distributed control flow. The Figure 2 illustrates the proposed approach.

The interface between hardware modules and the processor is implemented by the communication unit. This unit makes transparent for the hardware side which processor is being used. The processor is viewed by the hardware processes as another hardware process. For its implementation a parameterized VHDL description is generated, which can be synthesized.

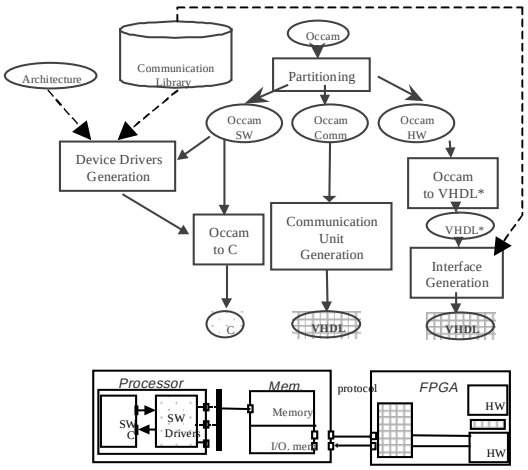

Figure 2- The interface generation approach

The interface between hardware components is generated by the Interface Generation module. First, the occam description of the hardware processes is translated into a VHDL* description, which includes send and receive functions for synchronous communication. After that, the Interface Generation module generates standard VHDL code, by using communication circuits, which implement send and receive operations. A detailed description can be found in [4].

3.1. Hardware/Software Interface Synthesis

First the occam description of the software processes is translated into C, where the communication statements are implemented by the high-level functions “send()” and “receive()”. These high-level functions depend only on the type of the data being transmitted or received by the software process, they call low-level functions as many times as it is necessary to transfer to/from the hardware processes.

The low-level functions perform the i/o operations between the communication unit and the processor. The set of high and low-level functions is referred in this work as device drivers and are architecture dependent. The software processes can run in a variety of processors each one with different speeds, instruction sets and I/O ports. Even with the same processor one can have several system architectures that use the same processor in different ways, like using a serial port of the processor instead of parallel one. Due to the diversity of architectures and processors, it is difficult to define a standard interface between the processor and the hardware components. In our approach the communication unit can be generated automatically and depends on the used processor. The current version considers the 8051 microcontroller family.

3.1.1. Device Driver Generation

As it can be seen in Figure 3, the device driver generator uses a library containing information about the target architecture such as its type, its resources etc. Additionally, for each device, the library must include information such as port description, access routines that encapsulates timing diagrams, and the number of transmitted bits on each low level i/o operation. By using this information, the device drivers generator can generate low level routines for "input()" and "output()" for each architecture taken into account. The "input()" and "output()" functions can transfer only the basic number of bits to/from the communication unit, and are called in the high-level functions as many times as it is necessary to complete the hole transfer. Figure 3 shows the algorithm for the high-level "send()" function.

```
void _sendChannelIO00 (_DataTypeChannel,dataType) {
unsigned char size;
   _DataTransferredIO* d_out;
   size = sizeof(_DataTypeChannel0);
   d_out = (_DataTransferred*)  dataType;
   _sendChannelIO0 (channelIO,  size, d_out);
   }
```

Figure 3 - High level send function.

3.1.2. Communication Unit Generation

Figure 4 shows the communication unit as an interface between the processor and the hardware components.

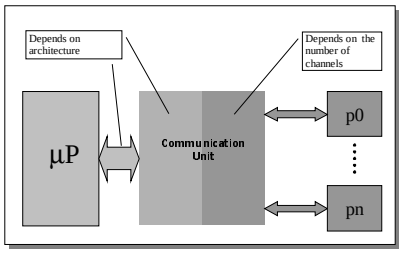

Figure 4 - Model of Communication Unit

The number and type of ports of this unit depends on the used processor and also on the number of communication channels at the hardware side. If there is communication actions occurring in parallel that uses the same channel, the communication unit must include an arbitration circuit.

4. Example

In this section the interface generation of a simple example is presented , which includes three processes running in parallel, is presented. Figure 5 (a) shows the occam description of such processes. First process p0 tests the variable "A", when this variable changes its value the process sends it through the channel c1 to process p1. This process does some computation and sends the new value to the process p2 through the channel c2. Process p2 also does some computation and sends the new value back to process p0 by using the channel c3. The processes p0, p1 and p1 uses distinct variables since in occam there is not the concept of global variable. Process p0 is implemented in software, whereas processes p1 and p2 will be implemented in hardware. Figure 5 (b) shows the C representation of process p0, The part (c) of the same figure shows the VHDL* description of process p1 including "send" and "receive" statements and in Figure 5(d) we have the additional states for implementing "receive" operation in a standard VHDL description.

5. Conclusions

In this paper we have presented an approach for interface generation in the PISH co-design system, which partitions concurrent processes in occam to be implemented in software and in hardware. The communication among such processes is generated in such a way, that the semantics of the initial description is preserved. After that, the virtual prototyping of the partitioned system is mapped into a real one, with the software processes running into some microcontroller and the hardware processes implemented as specific hardware (either as ASIC or FPGA´s). Beside hardware and software synthesis, this mapping includes the synthesis of the interface between processor and hardware, as well as interface between processes in hardware. In this work we have presented an approach for interface generation between hardware e software.

The proposed approach is based on a library of communication units including software units as well as hardware units. The software communication units or device drivers are functions for implementing the message passing operations *send* and *receive*. We have two kinds of such functions: high-level functions, which are independent of the target architecture, and low level functions, which implement communication for some architecture taken into account. A library for the 8051 microcontroller family has been implemented. The library of hardware components includes parameterized circuits for implementing synchronous communication, sender and receiver.

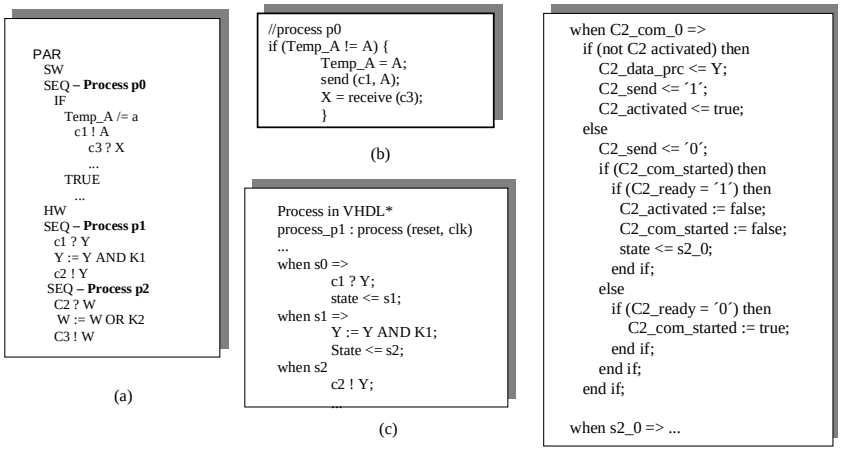

Figure 5 - (a) Occam description (b) Process p0 in C (c) Process P1 in VHDL* (d) states for send operation

References

[1] Knudsen, P.V.; and Madsen, J., *Communication Estimation for Hardware/Software Codesign, Proceedings of the International Workshop in Hardware/Sotware Co-Design - CODES 1998.*

[2] B. Lin, S. Vercauteren, H. De Man, *Embedded Architecture Co-Synthesis and System Integration*, International Workshop on Hardware/Software Codesign, March 1996.

[3] B. Lin, S. Vercauteren, H. De Man, ‘‘ Embedded Architecture Co-Synthesis and System Integration”, International Workshop on Hardware/Software Codesign, March 1996.

[4] C. Araújo and E. Barros , *Automatic Interface Generation among VHDL Processes in Hardware/Software Co-Design*, FDL´99, August 1999.

[5] D. Gajski and F. Vahid, *Specification and Design of Embedded Hardware-Software Systems*–IEEE Design and Test of Computers, pp.53-67, Spring 1995

[6] T. BenIsmail, M. Abid, K. O´Brien and A. Jerraya,,*An Approach for Hardware/Software Codesign*, Proceedings of the RSP 94, França, 1994

[7] A. Kalavade , E. Lee, *A Hardware-Software Codesign Methodology for DSP Applications* – IEEE Design and Test of Computers, pp. 16-28, September 1993

[8] D. E. Thomas, J. K. Adams, H. Schmit, *A Model and Methodology for Hardware/Software Codesign*– IEEE Design and Test of Computers, pp. 6-15, September 1993

[9] R.K. Gupta. , C.N. Coelho , G. De Micheli, *Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components*– Proceedings of the 29th DAC, 1992

[10] R. Ernst , J. Henkel, T. Benner, *Hardware-Software Co-Synthesis for Microcontrollers*– IEEE Design and Test of Computers, pp. 64-75, December 1993

[11] E.Barros and W. Rosenstiel *A Clustering Approach to Support Hardware/Software Partitioning*". *In:* K. Buchenrieder, and J. Rozenblit (eds.), *Computer Aided Software/Hardware Engineering.* Chapter 11- IEEE Press, 1994.

[12] E. Barros and A. Sampaio,. *Towards Probably Correct Hardware/ Software Partitioning Using Occam*. In Proceedings of the Third International Workshop on Hardware/Software Codesign, (1994) 210-217, IEEE Press.

[13] L. Silva, A. Sampaio and E. Barros, *A Normal Form Reduction Strategy for Hardware/Software Partitioning. In the Proceedings of the Conference Formal Methods Europe'97*

[14] D. Pountain and D. May, *A Tutorial Introduction to OCCAM Programming.* Inmos BSP Professional Books, (1987).

[15] C. A. R. Hoare, *Communicating Sequential Processes* Prentice-Hall, 1985

[16] P. Chou, R.B. Ortega and G. Borriello, *The Chinook Hardware/Software Co-synthesis System*. Proceedings of the 8th International Symposium on System Synthesis. 1995.