Hw/Sw Codesign of an ATM Network Interface card starting from a System Level Specification

Nacer-Eddine Zergainoh, G.F. Marchioro, A.A. Jerraya

TIMA Laboratory – SLS Group – 46 Avenue Felix Viallets - F-38031 Grenoble - Cedex France,

Ph.: (+33) 4 76 57 48 09, Fax: (+33) 4 76 47 38 14, e-mail: zergaino@imag.fr

ABSTRACT

This paper discusses the uses of SDL for the co-design of an ATM Network Interface Card (NIC). In this study, the initial specification is given in SDL. The architecture generation is made using Cosmos, a co-design tool for multiprocessor architecture. Several architectures are produced starting from the same initial SDL specification. The performance evaluation of these solutions was made using hardware/software co-simulation. This paper describes the experiment and the lessons learned about the capabilities and the restrictions of SDL and Cosmos for hardware/software co-design of distributed systems. The use of SDL allows for drastic reduction of the model size when compared to hardware/software model given in C/VHDL. SDL simulation may be 30 times faster than C/VHDL simulation.

1. INTRODUCTION

Advances in modern CAD tools and design methodologies are enabling the design of complex electronic systems under short time-to-market constraints. In this paper the word system refers to a multiprocessor distributed real time system composed of programmable processors executing software and dedicated hardware processors communicating through a complex network. Such a system may be implemented as a single chip, a board or a geographically distributed system. In a traditional design methodology, designers make the hardware/software partitioning at an early stage during the development cycle. The different parts of the system are designed by different groups. The integration of these different parts leads generally to a late detection of errors meaning higher cost and longer delay needed for the integration step. Besides, this early partitioning restraints the ability to investigate a better trade-off. The different parts of the system are generally oversized in order to reduce last-minute risks. A new generation of methods and tools for system design is emerging; they are able to handle the design of mixed hardware/software systems starting from system-level specification. These are called co-design or embedded system design tools; they provide a drastic increase in the productivity [2, 5, 7, 8, 9, 12, 17, 19, 21, 22]. This gain in productivity may be used to explore several architectural solutions to improve the quality and to reduce the cost of the final design.

This paper discusses the co-design of an ATM network interface card (NIC) using a co-design tool called Cosmos. This experiment allowed design exploration through the generation of several hardware/software partitioning solutions starting from the same initial specification given in SDL. The next section introduces the ATM NIC system. Section 3 deals with the co-design process of the NIC. Section 4 highlights the results obtained and section 5 presents the lessons learned.

2. THE ATM NETWORK INTERFACE CARD

The NIC system is aimed to link applications to the physical layer connected to the network. The NIC is composed of a stack made of four protocol layers: TCP, IP, AAL and ATM. Figure 1 shows an example of two applications connected to the network through the NICs.

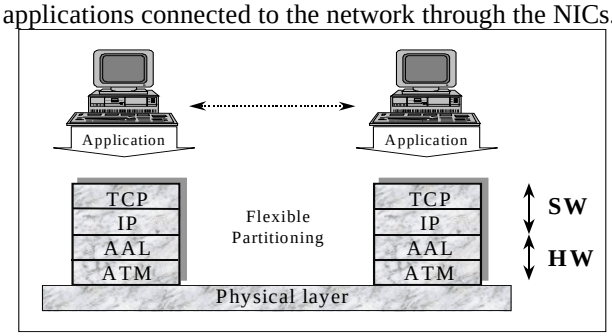

Figure 1. Applications connected via NICs.

2.1 System Requirements

Several simplifications have been made in order to allow the completion of the experimentation in reasonable time. These are:

– The NIC model takes into account only point to point communication. All the algorithms related to routing, traffic congestion and resources management have not been implemented;

– Error management is not implemented. For instance, the Internet Control Message Protocol (ICMP) will not be present;

– The reordering of frames will not be considered. The sliding window algorithm is not implemented.

Despite these simplifications, the remaining part of the NIC constitutes a quite complex system. In fact, the four layers act on data blocks of different sizes and performs different encapsulation of data. The rest of this section outlines the capabilities of the NIC. Because of lack of space only the TCP layer will be detailed.

The Transport Control Protocol layer (TCP) [3] provides a reliable communication between systems processing at different speed. This layer is in charge of establishing connections. The data transmitted is organized into frames. A frame is made of a header part and a data part. The size of the data part is variable.

Figure 2 shows the state diagram of the TCP layer. This model takes into account the restrictions listed above. "Closed" is the start state. Transitions are labeled by the enabling signals (Italic) and the produced signals. Some of the states of this diagram are hierarchical. For instance the state "Established" hides another state diagram that performs the data exchange whenever a connection is established.

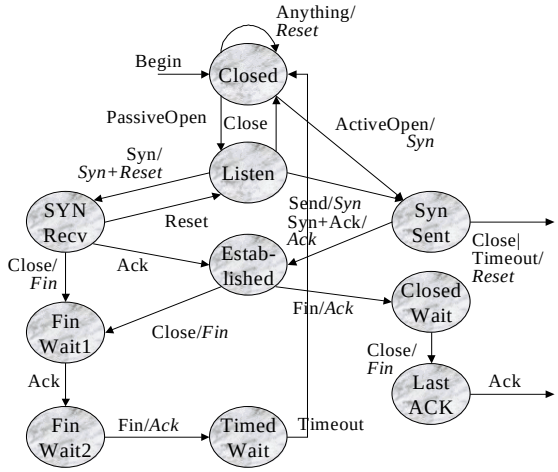

Figure 2. State diagram of the TCP layer.

The IP layer links TCP to the AAL/ATM layers. It adds specific headers to the segments sent by TCP. This layer acts on specific messages structures called datagrams. IP exchanges messages with both TCP and AAL layer. The AAL (ATM Adaptation Layer) [16] is in charge of the decomposition (resp. recomposition) of datagrams into (resp. starting from) ATM cells (53 bytes). The segmentation of the IP datagrams into ATM cells is made into several steps. First the datagrams are decomposed into packets made of 1 to 65535 bytes of data. Secondly, the packets are decomposed into cells made of 48 bytes; these are used for communication with the ATM layer. The reassembling uses to reverse scheme. The ATM layer provide links to the physical layer. It receives cells made of 48 bytes from the AAL side and produces ATM cells made of 53 bytes by adding a header.

2.2 System-level Specification

We used SDL (Specification and Description Language) for the specification of the NIC. SDL is intended for the modeling and simulation of communicating systems. It is standardized by the ITU [10]. A system described in SDL is regarded as a set of concurrent processes that communicate with each others using two concepts: signal routes and channels. The block concept is used to the model hierarchy. A block may be composed of a set of other blocks or a set of processes.

Figure 3 shows the top hierarchy of an SDL description of the NIC system. This model is made of four blocks communicating through channels. Each block in this model correspond to a layer of the NIC system. The lines correspond to channels. Each channel is defined by its name and a set of messages (signals) carried by the channel. For instance TCP_layer and IP_layer communicate through two channels TCPIP_Cntrl and TCPIP_Packets. The first carries control messages and the second data messages. The channel TCPIP_Cntrl carries 5 kind of messages (IPTCP_len, TCPIP_Daddr, TCPIP_len, TCPIP_SaddrS, TCPIP_SaddrD). Each of these blocks may be refined into a set of other blocks or processes. The leaf units are called processes. In SDL, a process is described as finite state machine that communicates asynchronously with other processes. Each process has an input queue where signals are buffered on arrival. Signals are buffered and consumed in the order in which they arrive (FIFO queues). Each process is composed of a set of states and transitions. The arrival of an expected signal in the input queue activate a transition and the process can then execute a set of actions such as manipulating variables, procedures call and emission of signals. The received signals determine the transition to be executed. When a signal has initiated a transition it is removed from the input queue. In SDL, a variables are owned by a specific process and cannot be modified by others processes. The synchronization between processes is achieved mainly using the exchange of messages (called signals in SDL).

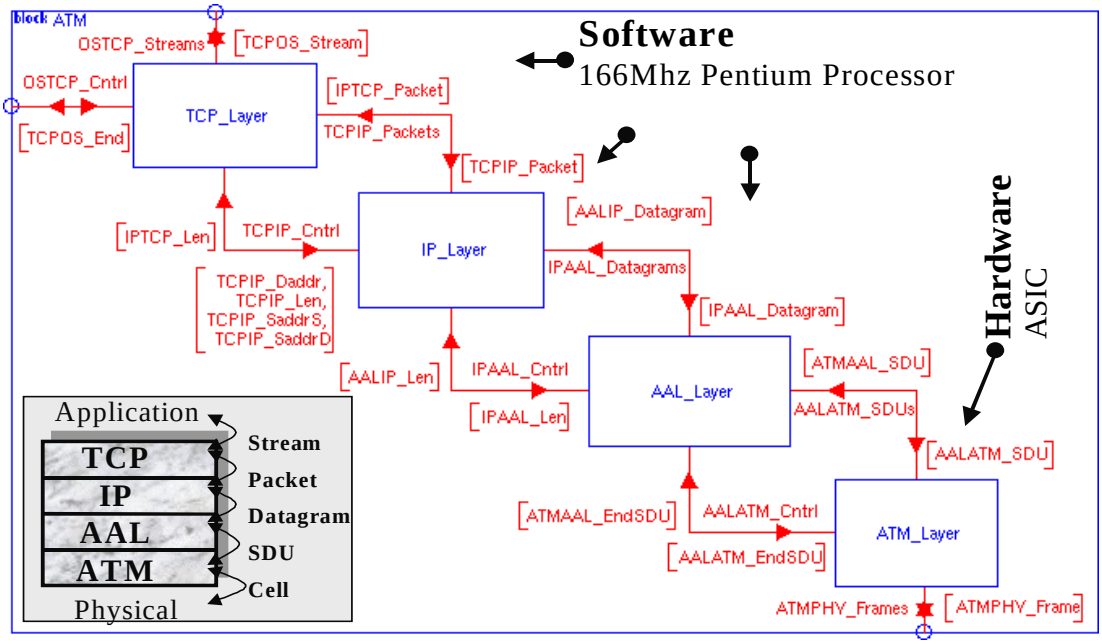

Figure 3. Structure of the NIC system described in SDL.

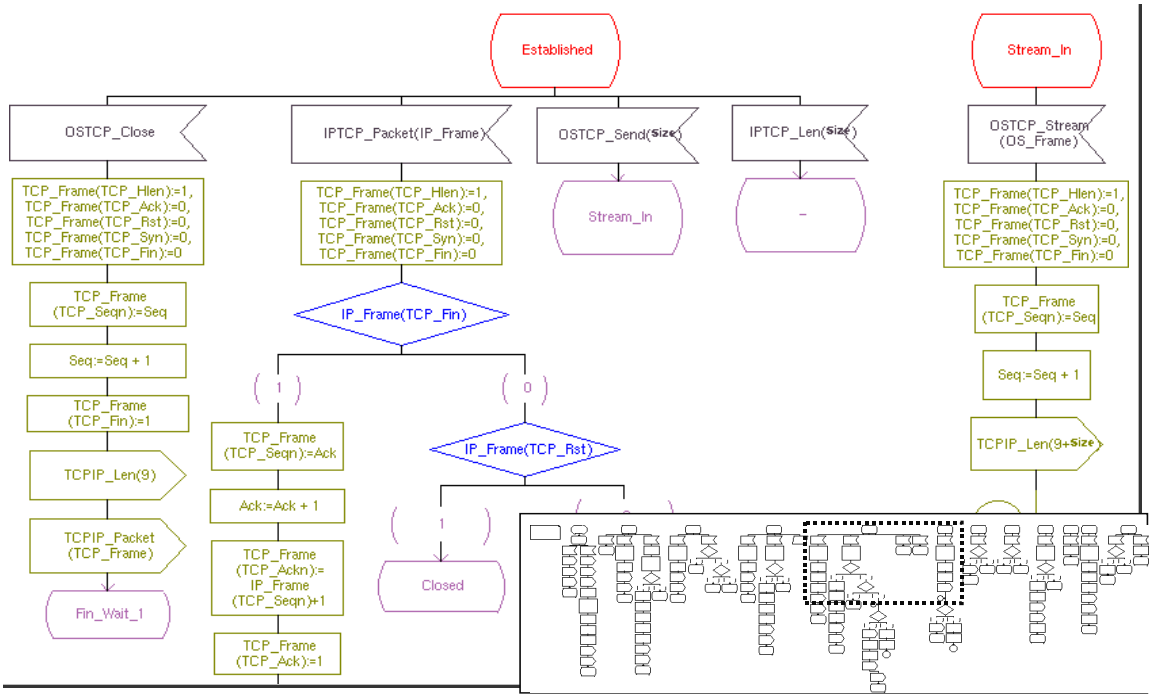

Figure 4: Extract of the behavioral description of the TCP layer.

Figure 4 represents one state extracted from the SDL description of the process corresponding to the TCP layer. Four transitions may be enabled starting from this state. The full TCP layer is made of 13 states and 32 transitions. Of course, these are system-level states. Each transition may hide complex computations, including loops and procedure calls, that may hide internal states. For instance, the transitions triggered by the signal IPTCP_Packet include several conditional statements and a loop. The overall specification is made of 9 processes.

3. THE CO-DESIGN APPROACH

The design flow adopted in this experimentation combines three tools (figure 5): Object Geode [13] for SDL specification and simulation, Cosmos for hardware/software co-design and VCI [20] for C/VHDL cosimulation. The design flow is an iterative scheme including three main steps.

3.1 System-level specification and validation:

This step includes the specification of the system in SDL and the functional validation [1]. Object Geode [13], a commercial framework, is used during this step. The validation includes simulation and verification. This step ensures the functional validation of the system. One can note that the validation tools are more powerful than what the classical CAD tools may provide. These act as model checking tools allowing to prove some property of the specification. For instance you may check that a given signal is never lost due to asynchronous communication. This verification is based on exhaustive simulation that should be handled carefully in order to avoid state explosion problems.

3.2 Hardware software co-design:

This step makes use of Cosmos (see figure 6), a hardware/software co-design tool for multiprocessor architecture [18]. Cosmos starts from an SDL model, it performs hardware/software partitioning and produces a distributed model made of hardware processors described in VHDL and software processors described as C-programs (see figure 7). Each processor may execute one or several processes of the initial specification. The SDL communication is refined into communication controllers and interconnections through simple wires.

3.3 Architecture co-simulation and calibration:

This step includes the validation of the produced C/VHDL model and the measurement of the performances of the architecture in terms of number of clock cycles. For co-simulation we use a tool called VCI [20]. The performances estimation is mostly manual in the present version, it combines the results of simulation with manual techniques in order to estimate the speed of the solution. The design flow of figure 5 is iterative and allows for architecture exploration. When a solution is obtained and its performances estimated, the design process may proceed to implementation or loop in order to produce a new solution through another partitioning step or through modifications of the initial model. The next sections report on several architectural solutions obtained using this model.

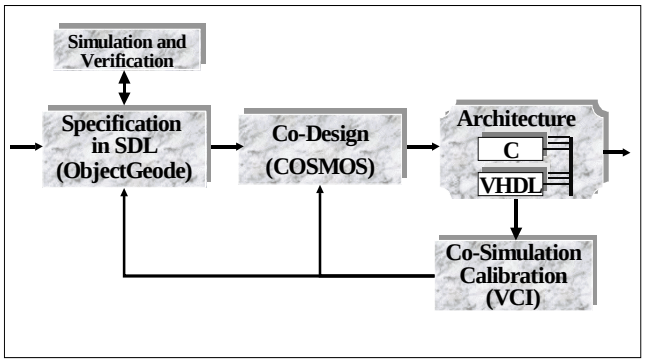

Figure 5. Design flow

4. ARCHITECTURE EXPLORATION

Five solutions have been produced starting from the initial specification in SDL. Each solution correspond to a partitioning solution produced by Cosmos under the control of the designers. Each solution generation takes about 30 minutes on a SPARC station.

The architecture exploration process acts on coarse gain level. Each protocol layer is considered as an indivisible task that should be allocated to one processor. The performances of the five solutions are represented in figure 8. The left part of figure shows the number and types of processors that compose the architecture. Each square labeled HW (resp. SW) corresponds to a hardware (resp. software) processor. This part of the figure shows also the allocation of the different task to processors. The right part of figure 8 shows the performances of the solutions. The speed of each solution is expressed in terms of throughputs (Megabits/second) and in terms of number of cycles needed to process one ATM cell. 25 Mb/s throughput corresponds to 5000 cycles/cell. We assume that the software is executed on Pentium processors. For example, the first solution is made of two processors. The software processor executes the three top layers of the protocol (TCP, IP, AAL) and the hardware processor is dedicated to the ATM layer.

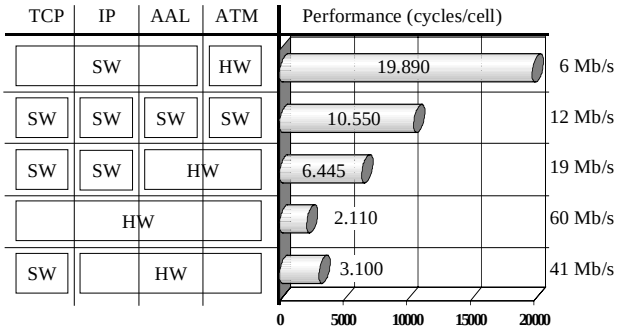

Figure 8. Performance of different implementations

The performances are computed based on simulation of the C/VHDL model produced by Cosmos. The cycle count for hardware execution is given by simulation and corresponds to the exact performances of the final solution. The cycle count for software execution is based on approximation.

Table 1 summarizes the size of the C/VHDL code produced and the simulation and co-simulation time. This table shows clearly the benefit of using system-level specification. The initial SDL specification is 10 times smaller than the produced C/VHDL model. The difference is mainly due to the communication refinement [4, 14]. The simulation time of the SDL model is 15 times faster than the VHDL model produced for solution 5 and 120 times faster than the co-simulation of the C/VHDL model produced for the first solution. The difference is also related to the communication. In the SDL model processes may exchange large data structures through message passing by using implicit queues. In the C/VHDL model these message passing are implemented using specific protocols where the queues are explicit and connected through physical buses. In the case of mixed C/VHDL models the simulation is even slower. In this case, we use a C/VHDL co-simulation based on UNIX/IPC [20].

Model	Lines		Simulation
	Behavior	ommunication	
SDL	794	103	1 min
VHDL	7.210	5.382	15 min [3]
C/VHDL	[1]	[1]	30 min [2]

Table 1. SDL vs. C/VHDL co-design and simulation.
[1] same order of magnitude than VHDL model;
[2] for solution 1 in figure 6; [3] for all hardware solution

5. EVALUATION AND LESSONS LEARNED

There are mainly 2 lessons learned from this application. These are related to the capabilities and limitations of SDL as specification model and of Cosmos as a co-design environment.

From the SDL point of view this experimentation shows clearly that SDL is very suited for the specification of protocols at the system-level. The main strength of SDL is the use of a powerful communication model based on asynchronous message passing. Additionally, the availability of powerful CASE environment [13] makes the use of SDL very practical and convenient.

However, this experiment also pointed out some restrictions of SDL. These are mainly:

– SDL lacks some arithmetic operation (such as modulo). The set of predefined arithmetic operation is quite restricted in SDL. SDL supports Abstract Data Types (ADT [1]), which allows the user to define new operators in C. However these are quite difficult to use and are not currently supported by Cosmos;

– SDL includes no explicit loop statement. Loops may be expressed using decision, join and label concepts. However this makes the use of loops quite difficult;

– Several hardware oriented aspects such as bit manipulation are not supported. This makes difficult the specification of functions such as CRC computation.

The above restrictions make SDL not very suitable for some application such as DSP where behavioral specification generally requires the use of complex arithmetic expressions and nested loops. One can note that the single communication model may induce some inefficiency in the model when asynchronous communication is not needed.

From the Cosmos point of view this experiment shows clearly the capabilities of the Cosmos approach. The main strengths of Cosmos: It supports a large subset of SDL, and it allows a quite fast and easy design space exploration. However this experiment pointed out several weakness in the approach:

– The non-availability of a standard library of communication unit to implement SDL queues implied lots of extra work. Several communication units where described in order to make the full path possible. This problem should be solved in the next version where new communication synthesis will allow accessing standard C & VHDL component.

– The maximal performances obtained by this automatic co-design is an order of magnitude less than what a designer may produce. In fact the fastest solution we obtained has a 60 Mb/s throughput. The ATM design may require faster implementation. The main restriction of Cosmos is the non-optimization of memory management. The use of transformation similar to those provided by Automium [15] should induce a drastic increase in performances.

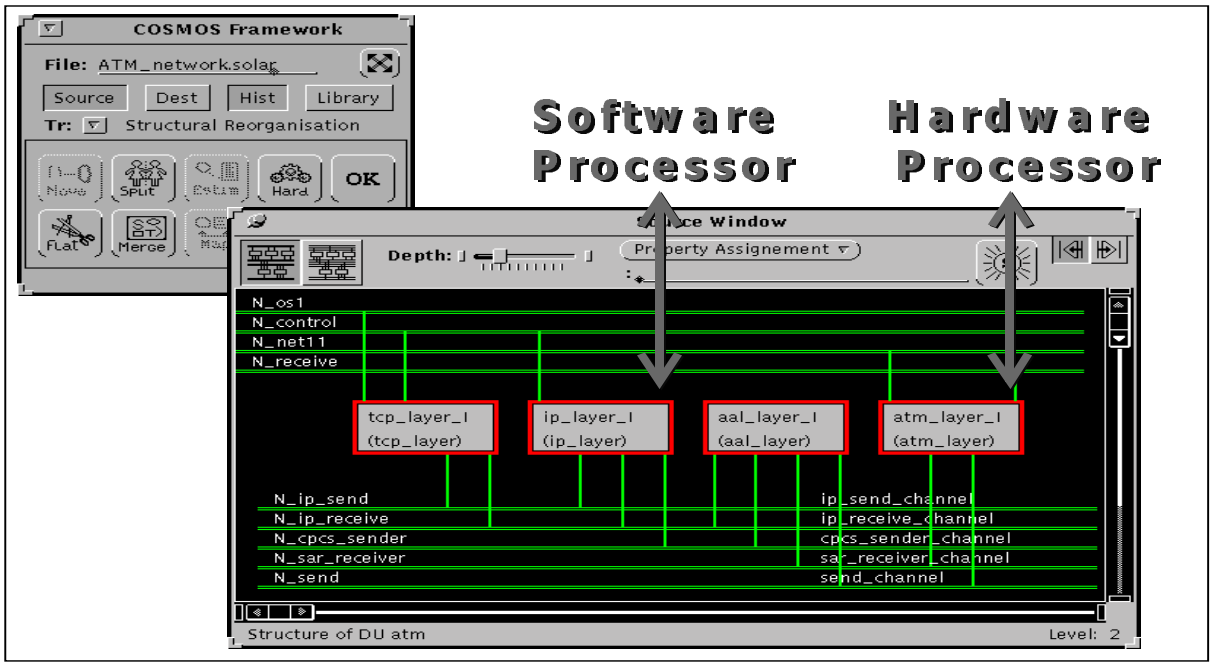

Figure 6. Refinement Steps in Cosmos Codesign Environment

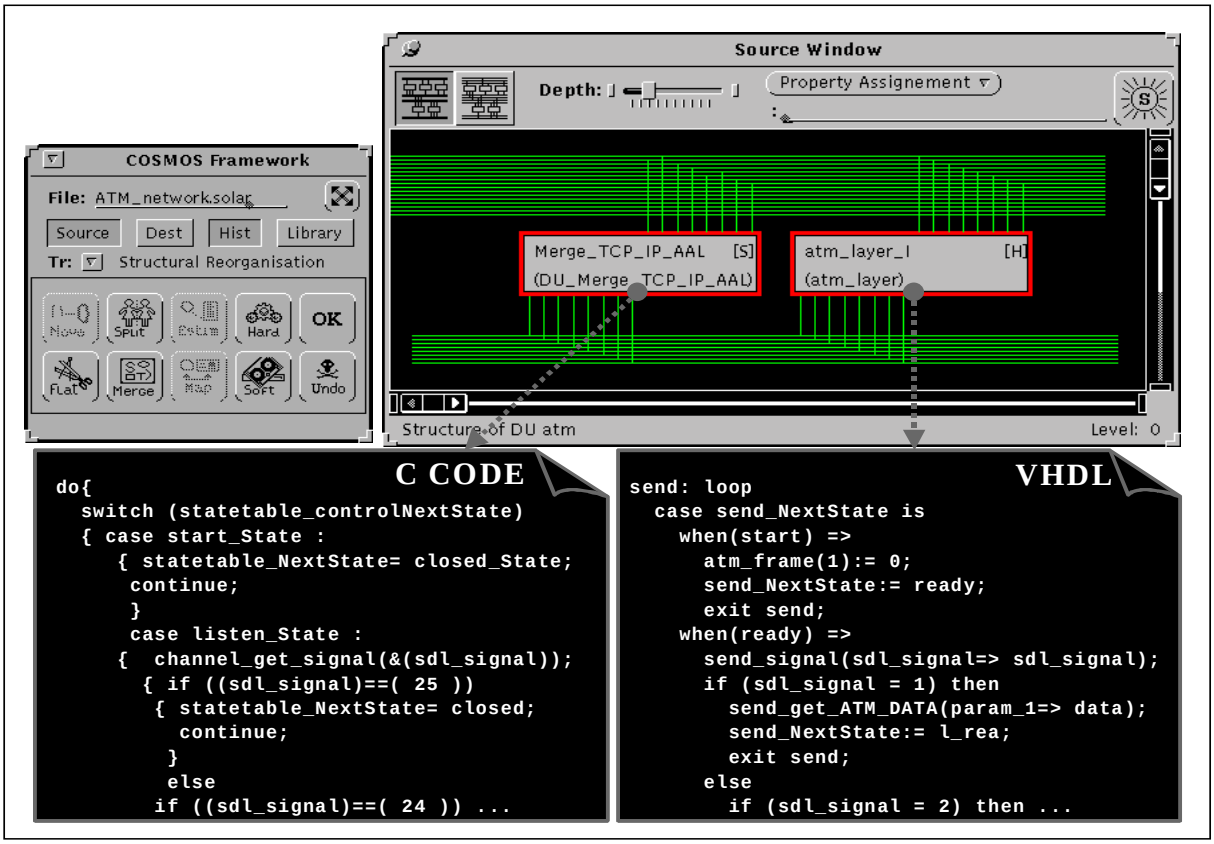

Figure 7. The produced C/VHDL model in Cosmos

6. CONCLUSION

This paper presented the results of the co-design of an ATM Network Interface Card. The initial description of the system was given in SDL. The co-design process was performed using Cosmos, an SDL based co-design environment. Several hardware/software architectures were produced starting from the same initial SDL model. The main lessons learned from this study are:

– SDL is clearly very suited for the specification of control and protocol systems. However it lacks some facilities for detailed hardware modeling and DSP like computation minimal capitals, usually only for the first letter of headings, captions, chapter title names, proper nouns and acronyms and abbreviations.

– The use of a system-level model may induce a drastic reduction in the description size and the simulation time when compared to lower level models such as C and VHDL.

– The use of co-design tools such as Cosmos allows for fast design space exploration starting from high-level description.

ACKNOWLEDGEMENTS

This work was supported by: France Telecom/CNET; SGS-Thomson; Esprit program under project COMITY and project CODAC; MEDEA program under project SMT; Aerospatiale and Verilog.

REFERENCES

[1] F. Belina, et al., SDL with Applications from Protocol Specification, Prentice Hall International, 1991.

[2] M. Chiodo, et al., A Case Study in Computer Aided Codesign of Embedded Controllers, Design Automation for Embedded Systems, Vol. 1, No. 1-2, pp. 51-67, January 1996.

[3] D. Comer, TCP/IP, Architecture, Protocole, Application, Collection iaa, InterEditions, ISBN , 1996.

[4] J.M. Daveau, et al., VHDL generation from SDL specifications, Proceedings of the IFIP Conference on Hardware Description Languages and their Application, pp. 182-201, April 1997.

[5] B. Felice, Hardware-Software Co-Design of Embedded Systems – The Polis Approach, Kluwer Academic Publishers, 1997.

[6] D. Kloos, et al., From Lotos to VHDL, Current Issue in Electronic Modelling, Vol. 3, pp. 111-140, September 1995.

[7] D. Gajski, et al., Specification and Design of Embedded Hardware/Software Systems, IEEE Design & Test of Computers, pp. 53-67, Spring 1995.

[8] R.K. Gupta, et al., Program Implementation Schemes for Hardware Software Systems, IEEE Design & Test of Computers, Vol. 27, No. 1, pp. 48-55, January 1994.

[9] J. Henkel, et al., COSYMA : A Software Oriented Approach to Hardware/Software Codesign, The Journal of Computer and Software Engineering, Vol 2, No 3, pp. 293-314, 1994.

[10] ITU-T Z.100 Functionnal Specification and Description Language, Recommandation Z.100 - Z.104, March 1993.

[11] J. Madsen, B. Hald, An Approach to Interface Synthesis, Proceedings of the 8th International Symposium on System Synthesis, pp. 16-21, September 1995.

[12] G.De Micheli, M Sami, Hardware/Sofware Co-Design, Kluwer Academic Publishers, 1995.

[13] ObjectGeode, www.verilogusa.com /og/og.html.

[14] R.B. Ortega, G. Borriello, Communication Synthesis for Embedded Systems with Global Considerations, Proceedings of the European Design Automation Conference with Euro-VHDL, pp. 69-73, September 1997.

[15] P. Slock, et al., Fast and Extensive System-level Memory Exploration for ATM Applications. 10th International Symposium on System Synthesis (Isss97), Belgium, September 17-19, 1997.

[16] M. De Pryker, ATM Mode de Transfert Asynchrone, Masson/Prentice Hall, ISBN 2-225-848-X, 1995.

[17] K.V. Rompaey, et al., CoWare - A Design Environnement for Heterogeneous Hardware/Software Systems, Proceedings of the European Design Automation Conference with Euro-VHDL, pp. 252-257, September 1996.

[18] J. Staunstrup, W. Wolf, Hardware/Software Co-Design: Principles and Practice, Kluwer Academic Publishers, 1997.

[19] D.E. Thomas, et al., A Model and Methodology for Hardware/Software Codesign, IEEE Design & Test of Computers, Vol. 10 No. 4, pp. 6-15, December 1993.

[20] C. Valderrama, et al., A Unified Model for Cosimulation and Cosynthesis of Mixed Hardware/Software Systems, Proceedings of the European Design and Test Conference, pp. 180-184, March 1995.

[21] J. Wilberg, et al., Design Flow for Hw/Sw Co-Synthesis of a Video Compression System, Proceedings of the Third International Workshop on Hardware/Software Codesign, pp. 73-80, September 1994.

[22] W. Wolf, Hw/Sw Codesign of Embedded Systems, Proceedings of the IEEE, Vol 82, No 7, pp. 967-989, 1994.