Application of the Co-design Finite State Machines Model for the Design of a Multiprocessor System

M. Marzougui, I.E. Bennour & M. Abid
Electronic and Micro-Electronic Laboratory
Faculty of Sciences of Monastir
5000 Monastir. Tunisia.
e-mail: rached.tourki@fsm.rnu.tn

Abstract: - This paper presents the use of CFSMs, Co-design Finite State Machines, for the high level design of multiprocessor system. This technique is based on implementation-independant representation that can be generated from a variety of specification languages such as VHDL, for control-dominated oriented systems. This representation allows to preserve semantical correctness throughout the design process, since CFSMs assume unbounded, non-zero reaction delays, that correspond both to hardware and software behavior. Our technique illustrates the use of the CFSMs model to specify the multiprocessor target architecture. We focus on the specification of the control oriented part.

Key-Words: - Co-design, Co-verification, FSMS, CFSM.

1 Introduction

Due to the increasing complexity of electronic systems, and the decreasing cost of microprocessors, embedded systems become largely used architecture. Mixed hardware-software systems may contain components that proceed at different speeds. Synchronous hardware modules compute their next states and outputs at each clock cycle. Software procedures, on the other hand, run sequentially on a micro-controller, and the reaction to a given condition may take many hundreds of clock cycles to compute, and many hundreds of clock cycles to propagate to other system components that might waiting for it. Furthermore, this delay depends on a very complex interaction of factors (such as the activity of other procedures, interrupts and so on) and hence can be almost very difficult to model in determinist way. Hardware is always active and computes a function, while real-time control software is generally event-driven and computes a reaction.
These considerations make appear that classical Finite State Machines (FSMs) are not an efficient solution to describe the behavior of heterogeneous systems that proceed at different speeds. In fact, their utilization require many ameliorations to adapt it to these type of systems. While the CFSMs model allows the representation of asynchronous systems, it may be used as an efficient model to specify the hardware software applications.
In this paper we illustrate the usefulness of the CFSMs model in the case of an heterogenous multiprocessor system in order to verify its functionality correctness. This system is composed by an heterogeneous subsystems such as standard processors, shared memory, some ASICs and a communication controller. We present the CFSMs representations of the bus transactions and the communication controller traffics. We focus on the functionality correctness of the communication controller through a detailed example. The latter consists on a communication between a sub-system of the target architecture and a VME based medium. The paper is organized as follows: Section 2 briefly describes CFSMs and its theoritical foundation for hardware software co-design. Section 3 describes our target architecture. In section 4 we show how a network of CFSMs can be mapped into an equivalent network of FSMs to model the target architecture. Section 5 draws some conclusions and outlines opportunities for future work.

2 FSMs versus CFSMs

A standard FSM transforms a set of inputs to a set of outputs with only finite amount of internal states [6]. On the other hand, concurrent FSMs implie that all FSMs change state exactly at the same time [5]. In other word, these models of computation commonly share the synchrony hypothesis. The term "synchronous" has been used in the literature to mean at least three rather different concepts [5]:

1. *Clocked*, the synchronization of the system components is assured by a global signal, as opposed to asynchronous systems where synchronization is a local property.

2. *Zero reaction time*, describing a system where the components react instantaneously to an event.

3. *Without acknowledge*, describing a communication protocol where the sender does not wait for the receiver to ac-

knowledge the reception of the message.

Finite State Machine model is based on "zero reaction time" synchrony hypothesis. This is not always true for heterogeneous components proceeding at different speeds.
A Co-design Finite State Machine (CFSM), inspired from FSM, is a formel mechanism to describe the behavior of some control oriented systems. CFSM is like standard Finite State Machine, transforms a set of inputs into a set of outputs with only finite internal state. The difference between the two models is that the concurrent FSMs imply that all the FSMs change state exactly at the same time [5].
A CFSM $\mathcal{C} = (I, O, R, F)$ is basically composed of a set of input events I (each with its associated set of values), a set of output events O (each with an associated set of values and possibly with an initial value in R), and a transition relation F describing how the CFSM *reacts* to input events by causing output events to occur. Each transition is *triggered* by the input events with the appropriate values and *emits* the output events with the appropriate values. The *reaction time* is *unbounded non-zero*. The *state* of the CFSM is constituted by the set of those events that are at the same time *input* and *output* for it. The non-zero reaction time provides the "storage" capability that is required to implement the concept of state.
The entire system is described by a CFSMs network synchronized by a global signal and communicate through a very low-level primitives : events. Events are emitted by a CFSM and can be detected by one or more CFSMs. Each CFSM is not bound to product the computation results at the next clock edge but it takes a non-zero unbounded time to perform its task. The receiver waits for the sender to emit the event, but the sender can proceed after emitting the event without the need to wait. To satisfy these requirements, a queue FIFO is placed between the sender and each receiver and will save the event until it is detected (or overwritten).
At the design partitioning step, the designer can choose between different hardware and software implementations for each component of the system specification [3].
In the case of hardware synthesis, each transition function is implemented and optimized with a combinational circuit [6]. The circuit outputs are latched to ensure the non-zero reaction delay.
The CFSM sub-network mapped into a software structure is constituted by a number of procedures and a simple Real Time Operating System (RTOS). The RTOS is responsible to activate the corresponding procedure when an event occurs.

3 Modeling of the proposed architecture

The used architecture is a set of heterogeneous subsystems: standard processors such as 80486 Intel microprocessor and DSP processor TMS320C25, communication controller, one ASIC and one FPGA for real time signal processing or pattern recognition [7]. These sub-systems share the same memory via the same bus for data transferring (Figure 1). To allow maximum inter-communicating, a communication controller processor is used to resolve signal conflict and arbitrate transactions. The main role of the communication controller is switching buses to allow direct connection and to adapt signal between incompatible devices when data transfer occurs. The bus arbiter accepts requests for the bus on eight request lines. Each request line has a corresponding grant line . If the bus is idle when a request is received, the arbiter will immediately respond on the grant line corresponding to the level of request pending [8].

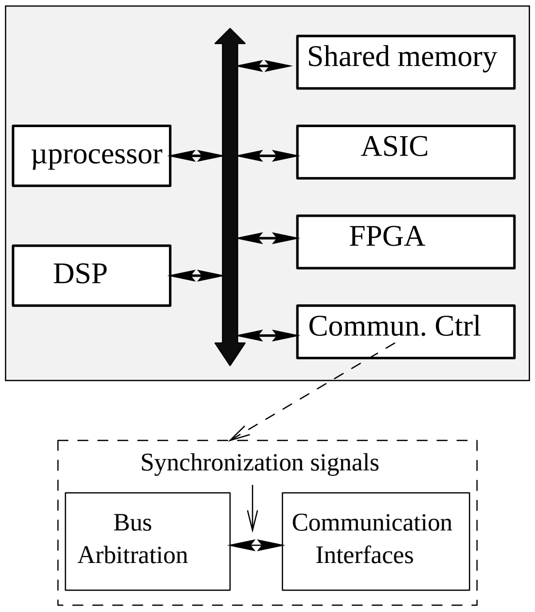

Figure 1: Plate-form architecture.

4 System modeling

The multiprocessors system is modeled with a communicated CFSMs network as shown in figure 2. Each CFSM reacts to input events and products outputs events. With this representation,the component ASIC can communicate with the FPGA, the microprocessor can dialog with the ASIC and the FPGA can use the shared memory. Any component requesting the bus access to communicate with another component should be authorized by the communication controller. The main objectives of this representation is to allow the verification of the functionality correctness of the communication controller. We focus in the ISA to VME communication function, while updating it for the co-simulation of many transactions between the proposed system and VME-bus based one. In the next subsections, we present the interactions between the microprocessor and the communication controller in order to read data from VME system.

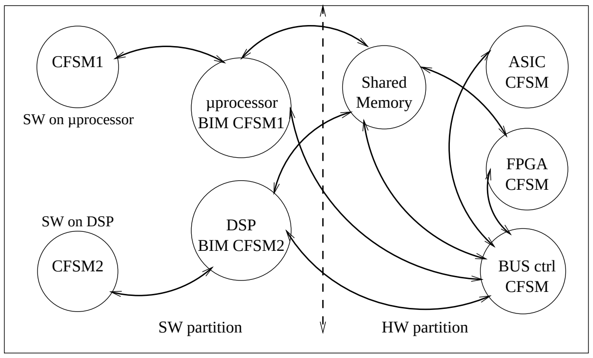

Figure 2: Modeling of the proposed architecture by the CFSMs model.

4.1 Bus transactions CFSMs based modeling

The interaction of the microprocessor with its environment may be performed by reading and writing data. In this case, there is a great deal of communication between the hardware and the software partitions of the system. This may be true, even if we abstract out all trivial communications such as instructions fetches and local data access. In other cases, it may be important to simulate all of the cycle level details of a set of transactions, but it is often true that we just need the data to be transferred and we aren't overly concerned with collecting the details of how it gets done each and every time. For this reason, the microprocessor and the DSP are abstracted to the *bus functional model.* So, the DSP and the microprocessor are abstracted to a test vector as shown in figure 3. The microprocessor sends its request to the communication controller and waits on the *Grant* signal to utilize the bus.

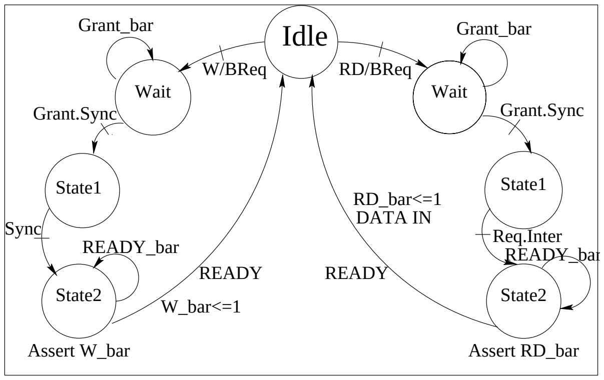

Figure 3: Bus transactions of the 80486 microprocessot CFSMs based modeling.

4.2 Communication controller CFSMs based modeling

The microprocessor react with its environment on read-write operations from the shared memory or from an external medium. This latter sends its request to the communication controller and waits until it receives the bus access acknowledge . At this moment, the Bus Control Unit (BCU) recognizes the communication nature based on the three least significant bits of the address bus. For the communication with an external resource, the BCU activates the corresponding interface to allow direct connection and to adapt incompatible signals.
Bus arbitration function allows to solve conflict problems by swiching buses between different requesters according to the priority option in which the communication controller is configured (fixed priority, round-robin priority, daisy-chain priority, etc). In many applications, the system may require to read data from external medium enclosing incompatible modules such as VME-bus, etc. Therefore, it is necessary that the communication controller allows the adaptation of some incompatible signals. For example, we focus on the case that the system requires to read data from VME-bus through VME-interface adapter. The interaction between the BCU and the ISA to VME adapter is shown in figure 4.

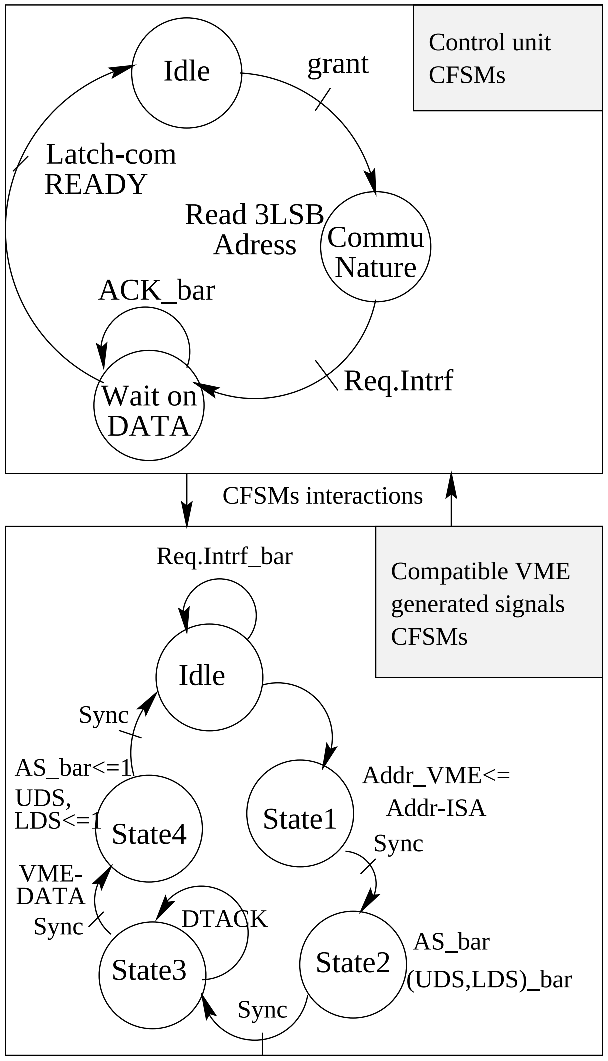

Figure 4: Control unit to interface adapter CFSMs interaction.

Initially, the interface will wait in the idle state until a VME bus access request (REQINTRF) is received. Thus, the VME interface will enter in the next state (State1). In this state, it connects system address to VME system address and activates the signal $R/\overline{W}$ to configurate the data bus transfer as read or write cycle. In state2, it generates the ($\overline{UDS}$, $\overline{LDS}$) signals to control the flow of data on the data bus, $\overline{AS}$ to validate address on the address bus and then enters in the state3. Thus, the VME-interface waits on the $\overline{DTACK}$ signal to steady that data transfer is completed. When $\overline{DTACK}$ is activated, the interface enters in the state4 and then latches data, negates $\overline{UDS}$,$\overline{LDS}$ and $\overline{AS}$ signals. When $\overline{DTACK}$ is negated, the interface returns to the idle state. The generated VME signals are shown in figure 5.

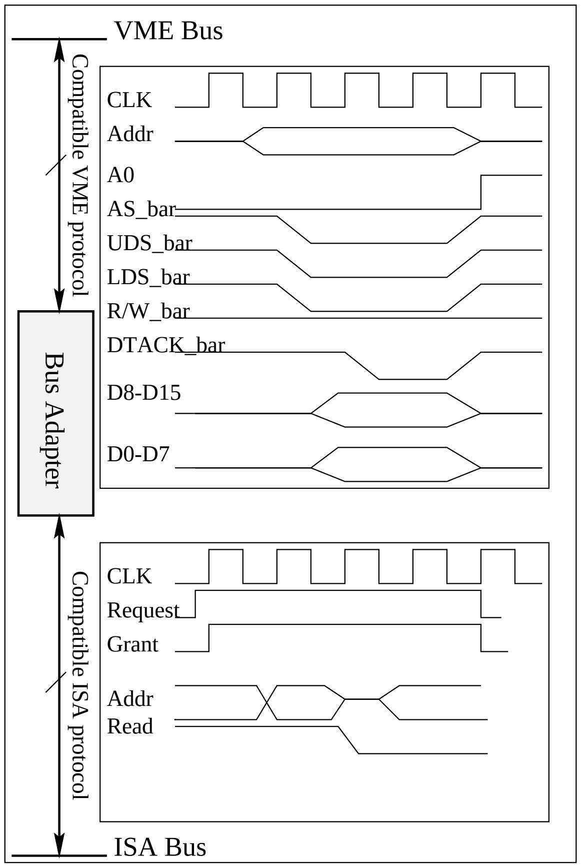

Figure 5: ISA to VME adapter.

5 Conclusion

This paper introduced Co-design finite State Machines model that is well suited to model control dominated real time applications.
The model is based on low level communication primitive called *events*. A key point of this approach is that the CFSMs specification is totally implementation independent (software or hardware), thus allowing designers to experiment with a number of implementation options. The possibility to applicate the model on a multiprocessor target architecture is presented.
Several complicated applications (control, signal processing for video compression, etc.) can be implemented in our target architecture. We are interested to implement a flow control and resource management algorithm, based on CFSMs, for ATM network. In this heterogeneous system, the flow control (Software) and the resource management subsystems are most conveniently modeled in much different ways, and yet there is a desire to study the interaction of these subsytems.
In the future, we are planning to explore the possibility to use Co-design finite State Machines model to co-simulate the functionnality correctness of the system at different levels of abstractions.

References

[1] Rowson, J. Hardware/Software co-simulation. In proceeding of *the design Automation Conference* (1994), pp. 439-440.

[2] Boriello. G, Chou. P and Ortega, *R. embedded system co-design - towards portability and rapid integration.* NATO, 1995.

[3] Gerard Berry and Laurent Causserat. Synchronous programming language and its mathematical semantics. In S.D brookes, A.W. Roscoe, and G. winskel, editors, *Seminar on concurrency, page 369-448.* Carnegie-Mellon University, 1984.

[4] D. Del Corso, H. Kirman, and J.D. Nicoud. *Microcomputer buses and links.* Academic press, London, 1986.

[5] M. Chiodo, P. Guisto, H. Hsieh, A. Jurescka, L. Lavagno and A. Sangiovanni-Vincentelli. *A formal Specification Model for Hardware/Software Co-design.* June 1, 1993.

[6] E.M. Sentovich, K.J. Singh, H. Savoj, R.K. Brayton and A.L. Sangiovanni-Vincentelli. Sequential Circuit Design Using Synthesis and Optimization. In *Proceeding of Internatioanal Conference On computer Design.* October 1992.

[7] C. Souani, M. Abid, A. Zitouni, M. Atri and R. Tourki. A heterogeneous Multiprocessor System with Dedicated Communication Controller. IN *Proceeding of the Fourth IEEE Conference On Electronics, Circuits and Systems ICECS'97.* December 1997.

[8] A. Zitouni, M. Abid, K. Torki, C. Souani, R. Tourki. *"Communication Synthesis Approach for Distributed Systems and its application During the Design of a Communication Controller."* In Proc Of the International Conference on MicroElectronics(ICM 98), pp.249 252, 1998.