Proceedings of the 2002 IEEE
International Symposium on Intelligent Control
Vancouver, Canada • October 27-30, 2002

Co-design of Embedded Controllers for Power Electronics and Electric Systems

Slim Ben Saoud
L.E.C.A.P.-E.P.T./ I.N.S.A.T.
B.P. 676, 1080 Tunis Cedex, TUNISIA
slimbensaoud@fulbrightweb.org

Daniel D. Gajski and Andreas Gerstlauer
Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA

Abstract– **Today, control algorithms are being more and more sophisticated due to the customer and governments demands. Then, their real-time implementation becomes a difficult task and needs more and more specific hardware systems with dedicated processors and usually systems-on-chip (SOCs).**

With the ever-increasing complexity and time-to-market pressures in the design of these specific control systems, a well-defined design methodology is more than even necessary.

In this paper we present a seamless approach for the design of control systems for power electronics and electric drives. We discuss the case of a DC system Control and describe in details different stages undergone. Generalization to others systems can be done easily using the same steps and transformations.

Index terms– **Co-design, Embedded Systems, Control, Electric process.**

I. INTRODUCTION

Today, variable speed motor control systems have a wide range of applications from industrial robotics to domestic washing machines, each with a specific set of requirements. Therefore, Motor control is being a vast market (estimated to be $5 billion annually for motors and motor controllers [1]) and the motor control industry is being a strong aggressive sector. Each industry to remain competitive has to answer the customer and governments demands for lower cost, greater reliability, environmental concerns regarding power consumption, emitted radiation and requirements for greater accuracy. This is achievable only by the use of sophisticated control systems [2,3,4].

The shortest time-to-market is a pressing requirement, consequently development time of new algorithms and new control device and debugging them must be minimized. This requirement can be satisfied only by using a well-defined System-level design methodology and by reducing the migration time between the algorithm development language and the hardware specification language.

The goal of this work is to introduce a new seamless approach for the development of complex control systems. This approach will be discussed using an application of the DC motor control. A generalization of this study to any other control system can be done easily using the same steps discussed in the following sections.

The control device will be described in four models, which represent four different levels of abstraction in our design approach [5,6]. All these models are executable and validated by simulation.

The rest of the paper is organized as follows: We first begin with a brief presentation of the used approach. Then we describe an executable specification model of the control system and we discuss the refinement of this model into architecture model, which accurately reflects the system architecture. Based on the retained architecture model, communication protocols between the system components are defined and communication model is developed.

II. DESIGN APPROACH

Managing the complexity at higher levels of abstraction is not possible without having a very well-defined system-level design flow. Therefore, in this project we propose a new seamless approach, which is a set of models and transformations on the models (Figure 1). The models written in programming language (SpecC language) are executables descriptions of the same system at different levels of abstraction in the design process. The transformations are a series of well-defined steps through which the initial specification is gradually mapped onto a detailed implementation description ready for manufacturing.

This new approach is based on 4 well-defined models, namely a specification model, an architecture model, a communication model, and finally, an implementation model. After each design step, the design model is statically analyzed to estimate certain quality metrics such as performance, cost, and power consumption. Analysis and estimation results are reported to the user and back-annotated into the model for simulation and further synthesis.

0-7803-7620-X/02/$17.00 © 2002 IEEE

In this paper we focus on the synthesis flow which contains the steps of specification, architecture exploration and communication synthesis. Implementation can then be done easily using standard tools.

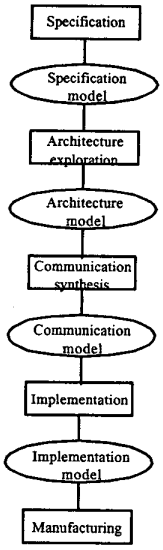

Figure 1: Design Approach

III. SPECIFICATION

The system design process starts with the specification model written by the user to specify the desired system functionality. This model is a purely functional, abstract model that is free of any implementation details.

Figure 2 shows the specification model of the DC control system in SpecC language. The used control algorithm (*CTL_Alg*) is composed of two control loops: an outer motion loop (*M_Alg*) and an inner current loop (*C_Alg*). Each of them is specified in a separate sub-behavior and associated to a clock-behavior that generated the synchronization event to activate the corresponding control loop at the predefined periodic step.

The I/O modules necessary for the control device functioning are specified in two behaviors: the *PWM* behavior represents the PWM[1] module functioning while the *ACQ* behavior represents the information acquisition modules.

The *PWM* behavior generates two complementary signals C_0 and $\bar{C}$ with the same frequency as the current control module clock and according to the pulse width value α (for C_0) obtained by the current control behavior.

The current acquisition behavior (Acq_i) captures the current value (N_{im} obtained from the used ADC[2] component) and computes its average value over the current control period (i_m). While the speed acquisition behavior (Acq_Ω) computes the speed value (Ω_m) from the two signals S0 and S1 generated by the optical incremental encoder (sensor used on the process under control).

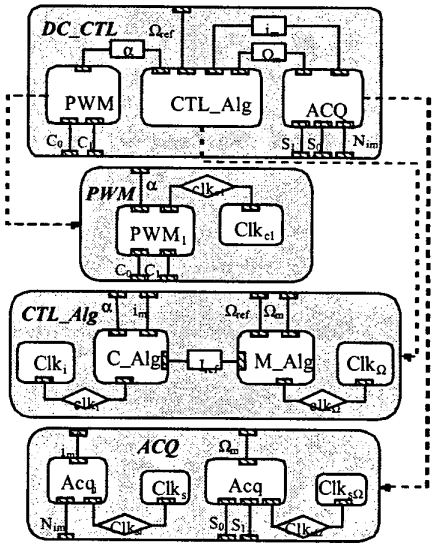

Figure 2: Specification model of the control device

As shown on figure 2, the SpecC specification describes the control device functionality in a clear and precise manner.

IV. ARCHITECTURE EXPLORATION

Architecture exploration is the first part of the system synthesis process that develops system architecture from the specification model. The purpose of architecture exploration is to map the computational parts of the specification onto the components of system architecture. The steps involved in this process are allocation, partitioning and scheduling. Through this process, the specification model is gradually refined into the architecture model.

A. Allocation

The first task of the architectural exploration process is the allocation of a system target architecture consisting of a set of components and their connectivity. Allocation selects the number and types of processing elements (PEs), memories and busses in the architecture, and it defines the way PEs and memories are connected over the system busses.

[1] Pulse Width Modulation
[2] Analog to Digital Converter

Components and protocols are taken out of a library and can range from full-custom designs to fixed IPs[3].
After an architecture has been allocated, the first step in implementing the specification on the given architecture is to map the specification model behaviors onto the architecture's processing elements.
For the control device application, usually the I/O modules are done by hardware modules (ADC, Timers, ...) while the control algorithm is implemented in a standard processor. The retained model is composed of a processor core (DSP56600 core) running control algorithm and a hardware component (ASIC) for the I/O functions (Figure3).

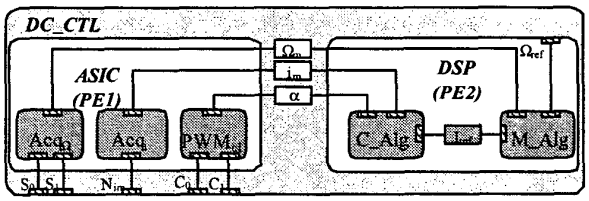

Figure 3: Architecture models after behavior partitioning

Formerly local variables used for communication between behaviors mapped to different components now become global, system-level variables (α, i_m, Ω_m).

B. Variable Partitioning

After behavior partitioning, communication between behaviors mapped to different PEs is performed via global, shared variables. Global variables have to be assigned to local memory in the PEs or to a dedicated shared memory component. In the refined model after variable partitioning, global variables are replaced with abstract channels and code is inserted into the behaviors to communicate variable values over those channels.

In our application, we use local copies of these variables in each PEs (Figure 4). Updated data values are communicated between ASIC and DSP through 3 abstract channels (C_α, C_{im} and $C_{\Omega m}$).

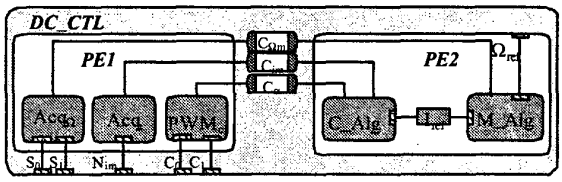

Figure 4: Architecture model after variable partitioning

[3] Intellectual Property

C. Scheduling

Scheduling determines the execution order of behaviors that execute on inherently sequential PEs. Scheduling may be done statically or dynamically [7].

Figure 5 shows the scheduling of the parallel control algorithm running on the DSP core. Due to the dynamic timing relation between motion loop and current loop tasks, a dynamic scheduling scheme is implemented. The motion control represents the main program, which executes in periodic manner (T_s=20ms). Whenever a new current period arrives (T_c=284µs), the main task is interrupted in order to execute the current control.

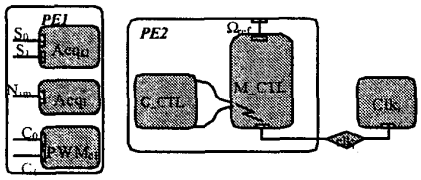

Figure 5: Architecture model after scheduling

According to this scheduled model and in order to simplify synchronization for communication, all exchanges are done at the beginning of each current control loop which means at each period T_c (Figure 6). The Ω_m value will be then a local variable of the DSP as well as I_{ref}.

Exchanges synchronization can be done by an external clock (Figure 5) or by an event generated by the ASIC and precisely by the PWM module since it will integrate a temporization function at the period of T_c.

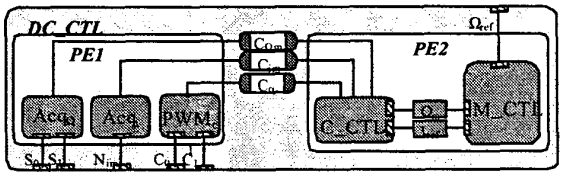

Figure 6: Modification of variable partitioning

D. Channel Partitioning

Channel partitioning is the process of mapping and grouping the abstract, global communication channels between components onto the busses of the target architecture. In the refined model, additional top-level channels are used to represent system busses. Then channel partitioning is reflected by hierarchically grouping and encapsulating the abstract, global channels under the top-level bus channels (Figure 7).

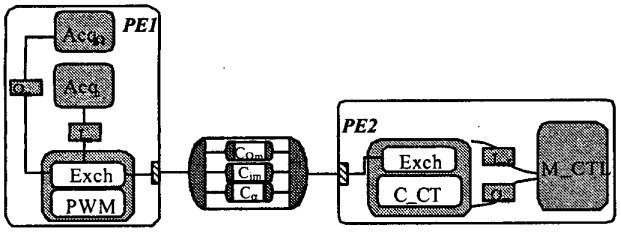

Figure 7: Architecture model after channel partitioning

V. COMMUNICATION SYNTHESIS

The purpose of communication synthesis is to refine the abstract communication in the architecture model into an actual implementation over the wires of the system busses. This requires insertion of communication protocols for the busses, synthesis of protocol transducers to translate between incompatible protocols, and inlining of protocols into hardware and software.

A. Protocol Insertion

During the protocol insertion, a description of the protocol is taken out of the protocol library in the form of a protocol channel and inserted into the corresponding virtual system bus channel (Figure 8).

The abstract communication primitives provided of the bus channel are rewritten into an implementation using the primitives provided by the protocol layer. The outer application layer of the bus channel implements the required semantics over the actual bus protocol. This includes tasks like synchronization, arbitration, bus addressing, data slicing, and so on.

All the abstract bus channels in the model are replaced with their equivalent hierarchical combinations of protocol and application layers that implements the abstract communication of each bus over the actual protocol for that bus.

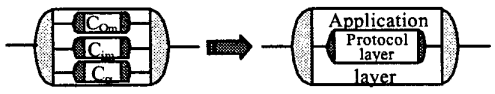

Figure 8: Protocol insertion principle

In this example, after protocol insertion, the processor is the central component and the master of the system bus. The software on the processor initiates all data transfers on the processor bus from and to the hardware component. However, these exchanges are initiated either by an external clock or by the hardware component that send an event (IT) at each T_c period to the processor by triggering its interrupt in order to execute the exchanges process (Figure 9).

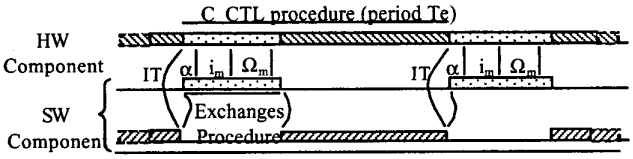

Figure 9: HW/SW Synchronization diagrams

The protocol channel in the system bus and the wrapped processor model describe and implement the DSP56600 bus protocol according to its timing diagram [8], shown in figure 10.

B. Protocol Inlining

Protocol inlining is the process of inlining the channel functionality into the connected components and exposing the actual wires of the busses. The communication code is moved into the components where it is implemented in software or hardware. On the hardware side, FSMDs that implement the communication and bus protocol functionality are synthesized. On the software side, bus drivers and interrupt handlers that perform the communication using the processor's I/O instructions are generated or customized.

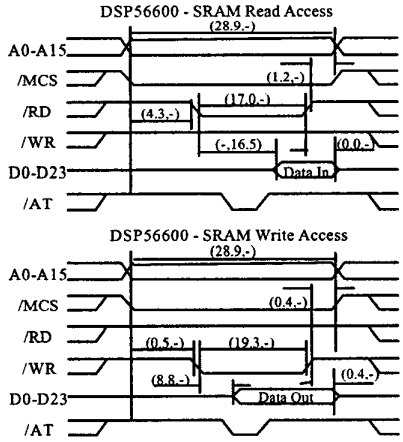

Figure 10: Protocols of the DSP56600 external bus

The communication model obtained after protocol inlining is shown in Figure 11. For the ASIC, communication primitives are inlined into the exchanges sub-behavior. Therefore, exchanges SFSMD model is created and inserted into the ASIC SFSMD model [Figure 12].

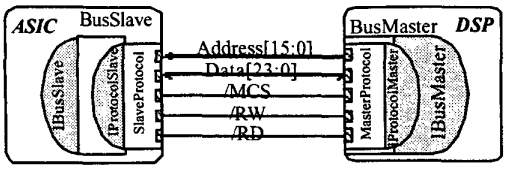

Figure 11: Communication model after protocol inlining

The exchanges hardware module synchronizes with the DSP by raising the processor's interrupt line IRQC in its first state S1 until a transfer with the address of the custom hardware is recognized. Then the WR control signal is sampled until a falling edge has been detected that signals the beginning of a bus write cycle. Communication continues at the same manner for two read cycles.

The obtained communication model is validated and is ready for use directly to generate the implementation model. The leaf behaviors of the design model will be fed into different tools in order to obtain their implementation [9].

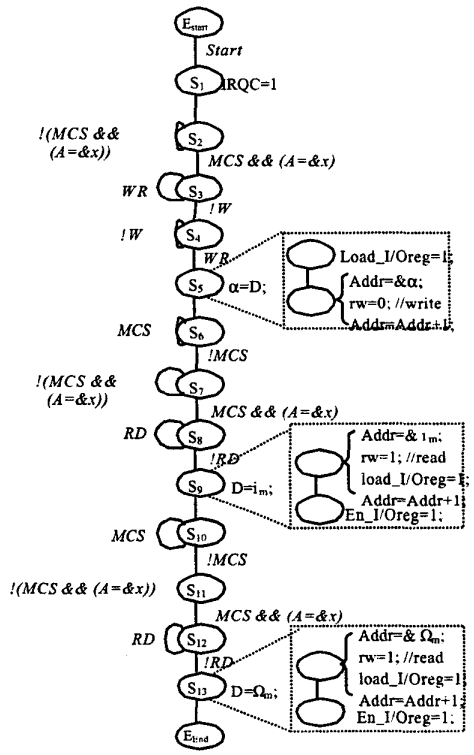

Figure 12: HW Communication SFSMDs

VI. CONCLUSIONS

In this paper we introduce a new seamless approach for the design of control systems of Power Electronics and Electric drives processes. We presented the study of a DC motor drive, which can be easily generalized to any other process control.

We have shown the various steps that gradually refines the initial specification down to an actual communication model ready for implementation and manufacturing.

The well-defined nature of the presented approach models and transformations helps focusing design efforts on central issues, provides the basis for design automation tools, and enables application of formal methods.

The use of the same language for the specification and for the design process reduces significantly the time-to-market by minimizing largely communication among designers and customers.

REFERENCES

[1] Analog Devices, Products and Datasheets, Whitepapers, "ASSPs for Motion Control Applications Use Embedded Digital Signal Processing Technology", 2001

[2] D. Krakauer, "Single chip DSP Motor Control Systems Catching on in Home Appliances", Appliance magazine, October 2000

[3] C. Cecati, "Microprocessors for Power Electronics and Electrical Drives Applications", IES Newsletter, vol. 46, no. 3, September 1999

[4] A. Murray, P. Kettle, "Towards a single chip DSP based motor control solution", Proceedings PCIM - Intelligent Motion, May 1996, Nurnberg, Germany, pp. 315-326

[5] D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, S. Zhao, "SpecC: Specification Language and Methodology", Kluwer Academic Publishers, 2000

[6] A. Gerstlauer, R. Dömer, Junyu Peng, D. Gajski, "System Design: A Practical Guide with SpecC", Kluwer Academic Publishers, 2001

[7] A. Gerstlauer, S. Zhao, D. Gajski, A. Horak, "Design of a GSM Vocoder using SpecC Methodology", University of California, Irvine, Technical Report ICS-TR-99-11, February 1999

[8] Motorola, Inc., Semiconductor Products Sector, DSP Division, DSP 56600 16-bit Digital Signal Processor Family Manual, DSP56600FM/AD, 1996

[9] R. Dömer, D. Gajski, J. Zhu, "Specification and Design of Embedded Systems", *it+ti magazine*, Oldenbourg Verlag, Munich, Germany, No. 3, June 1998.