



Toward a framework centered in simulation

M. Ridao, J.H. Doorn

*ISISTAN - Facultad de Ciencias Exactas - UNICEN, Comisión
de Investigaciones Científicas Provincia de Buenos Aires,
San Martín 57 - (7000) Tandil, Argentina*

Abstract

Frameworks centered in languages or centered in a given design methodology are very well known and used every day. However, when a simulation task is to be accomplished, few of them give the user a perfect fit to his/hers problem. Embedded microcontroller systems are characterized by a strong interaction with the environment in which they are going to be used. Due to this great interaction, the validation of the software and the hardware of microcontroller based applications has much to do with the 'environment'. In the present article, a prototype of a framework conceived with simulation as the main target is described.

1.0 Introduction

Unlike 'standard' application, the task of verification and validation of a microcontroller based application is rather complex. This is due to the testing phase which requires a totally different approach. If a portion of an standard system should be tested, a set of test data is supplied to it. In this case, the main task is to choose the appropriate set of data based in some test coverage criteria, such as instruction coverage, branch coverage, and others [1] [2]. The problem of how to supply such data to the system under testing is usually disregarded as a source of complexities, because it can be, in most cases, solved using very simple pieces of software. When testing microcontroller based applications, the selection of the test cases still remains, but the task to supply these cases usually is hard to accomplish in an automatic and controlled fashion without a painful user intervention. For these systems, a test case may become, for instance, the presence of an input in a given port, during at least some time,

74 Software for Electrical Engineering

but lasting less than a maximum period, while another input doesn't show up and the value converted in the A/D is greater than some figure. To supply this sort of data might be a problem by itself if the signals are coming from physical equipment. Then, two sources of difficulties are present.

On the other hand, embedded microcontroller system development is usually done with some degree of concurrence in the hardware and the software construction. Even in most cases, a first version of the hardware is ready before software development, at that time it is not totally tested because of the need of a fully operational software [3]. In addition to it, software testing would require a totally operational hardware. The overlapping of the hardware and software development takes place essentially during verification and validation phase, in which the errors are mainly located in the software, but sometimes in the hardware.

For the above reasons, the availability of testing aid tools is strongly needed. Several approaches have been used with different degrees of users satisfaction. It always seemed that simulations approach would be a promising one, since no actual equipments have to be handled, at least during the early stages of verification and validation process [4].

A tool to verify and validate microcontroller based applications, has been developed. It was conceived as the integrated simulation of the microcontroller and its environment. This framework is composed by a set of cooperative tools. Some of the components connected are: CAD's Post Processor, Macro Expansion, Compilers, Editor, Program Transformation, Man Machine Interface, help, etc. And, in the center of them, the Simulator, as the framework engine.

Each component in this framework has a specific role in the verification and validation task. By these means, it becomes possible to find faults in the hardware design or in the microcontroller software in the same debugging process. Besides, it can be accomplished without building the actual device, making the developer task easier [5].

To simulate the system behavior, it is necessary to describe all the relations between the microcontroller and its environment [6]. A close look to this environment shows that it can be seen as composed by two halves. First one is the nearby environment, which mainly comprises hardware components connected to the microcontroller, and the faraway one, constituted by the 'actual' environment of the application. Then, the events coming from those external worlds must be simulated [7]. The simulation of the external inputs has many features in common with stochastic simulation languages [8] [9].

2.0 Framework Description

This framework use the joining effort of a set of tools, allowing the user to simulate his application in a simple and efficient fashion, since each component has been conceived to simplify the developing task.

In order to describe this framework, individual items conforming the tool will be introduced. In the Figure 1, the interaction between the different components is showed.

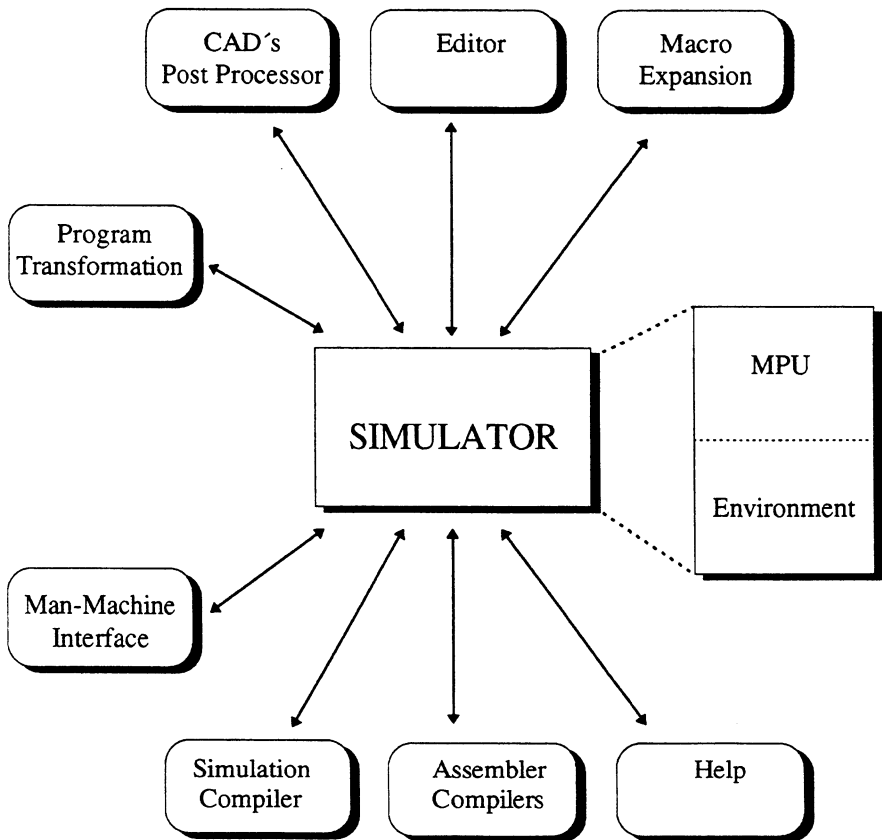


Figure 1: Framework Configuration

The information flows among all these components, each one makes some kind of processing on the data, with the simulation as a final target [10]. Following, a brief description is given for each component.

2.1 CAD's Post Processor

To be able to begin with the simulation process, a model must be created [11] [12]. The developer has to give the binary program that will run in the microcontroller and a full description of its environment. The last one can be obtained using three different approaches: using CAD output files, writing the

76 Software for Electrical Engineering

boolean equations that model the system, or invoking macroinstructions of the components used [13].

The first method consists in picking, if they are available, working files of any of the supported CAD's, and then generating, without any user intervention, macro invocations for each component involved [14]. This framework includes a macro instructions library, containing entries for most of the commercial component the designer may use.

The model can be also obtained writing the logical equations that link inputs and outputs of the components connected to the microcontroller. These equations have many points in common with those used in many Hardware Description Languages [15][16][17].

Since the developer knows component's functionality, he can use a macro invocation instead of writing the whole set of boolean equations for that component. So, he can write the invocations instead of generating them automatically. User written or automatically derived macroinstructions will produce logical equations by means of the use of a macro expansion process.

The three described procedures, are not mutually excluding. In other words, a given application can be partially described using CAD files generated macro instructions invocations, with macroinstructions invocations and boolean equations added by the user to complete the model. Also, sentences generating asynchronical events coming from the faraway environment, may be introduced.

2.2 Editor

A simple source program editor is included to write from scratch or to modify macro invocations or logical equations. This editor may be also used to input the source program for the assembler compiler which creates the binary program of the microcontroller.

2.3 Macro Expansion

The user written or the automatically derived macro instructions become inputs to a macro instructions preprocessor that expands them. As a result, a macro invocation is traduced to one or more boolean equations, and they will look like:

$$output = F(inputs) \quad (1)$$

These functions express what input combination produce high (1), low (0) or third state output. Also, for the output value 1, it can be 'regular' or 'Open collector'. In other words, for each output pin there will be at most three out of the following four equations:

$$\begin{aligned}output_0 &= g(inputs) \\output_1_N &= f_N(inputs) \\output_1_C &= f_C(inputs) \\output_T &= h(inputs)\end{aligned}\tag{2}$$

2.4 Program Transformation

Once the model was built, it appears as a set of equations that will become the source code for the simulation compiler. However, before compiling it, the equations will be searched in order to check if there are more than one equation with the same output. If they are found, a transformation of the original model at source code level is produced, to ensure that every variable is only one time on the left side of an equation [18]. In most cases, the existence of more than one equation with the same output is not due to user errors, but to the wiring of several outputs to the same point purposely by the designer. This connection is widely used and adds functionality to the application.

The search of connected output is easier than it may look like after a first sight, because the language that is going to run the model is non procedural, and for that reason, the order in which equations are written is meaningless.

2.5 Simulation Compiler

As usual, the simulation compiler performs a lexical analysis of the source code, and then a parsing phase, looking for syntax errors, making types verification, and generating run time code. Some of the equations will produce one or more events that make the simulation engine to run. These events are stored in the Future Events Table. An entry in this table tells when a signal is going to change.

2.6 Assembler Compilers

For every supported microcontroller there is an assembler compiler which creates binary code that is later “loaded” into the simulated PROM or EPROM of the processor. The outputs of these compilers may be sent directly to an EPROM programmer to test the software in the actual application or even to be used in a small scale production.

2.7 Man-Machine Interface

This framework allows the developer to validate simultaneously the microcontroller software and hardware [19].

The internal states and outputs presentation is made by means of metaphors, bringing the user the sensation of really being proving his system. The developer can see the system behavior in such a way that he feels like working with a virtual oscilloscope. Each channel may be assigned to any of the system inputs, outputs, and internal states.



78 Software for Electrical Engineering

Microcontroller registers, clocks, counters and other internal components may be displayed at any time in several formats by the microcontroller simulator.

User may stop the simulation at any wished time, at a given simulated time or when a condition is accomplished.

The simulation may proceed as fast as possible, slowed down by a given factor or executed step by step.

2.8 Simulator

The whole design of this framework has been done taking simulation as the goal. In other words, all other subsystems were conceived to provide the required services needed by the simulator.

A bug found during the simulation phase in the microcontroller software or in the hardware, can be traced back until the source code program or the original CAD data. This allows a high level maintenance of the design and the software.

Usual software debugging techniques such as break-points, can be freely combined with user creation of events at any simulated time, or any of the usual user inputs needed during simulation, to force the appropriated test of both sides of the device under development.

During the simulation, the earlier event is taken from the Table, and every routine containing the changing input is calculated. In this procedure new events are generated and included in the table. So, the simulation engine acts as a "task scheduler" among generated routines.

2.9 Simulation Manager

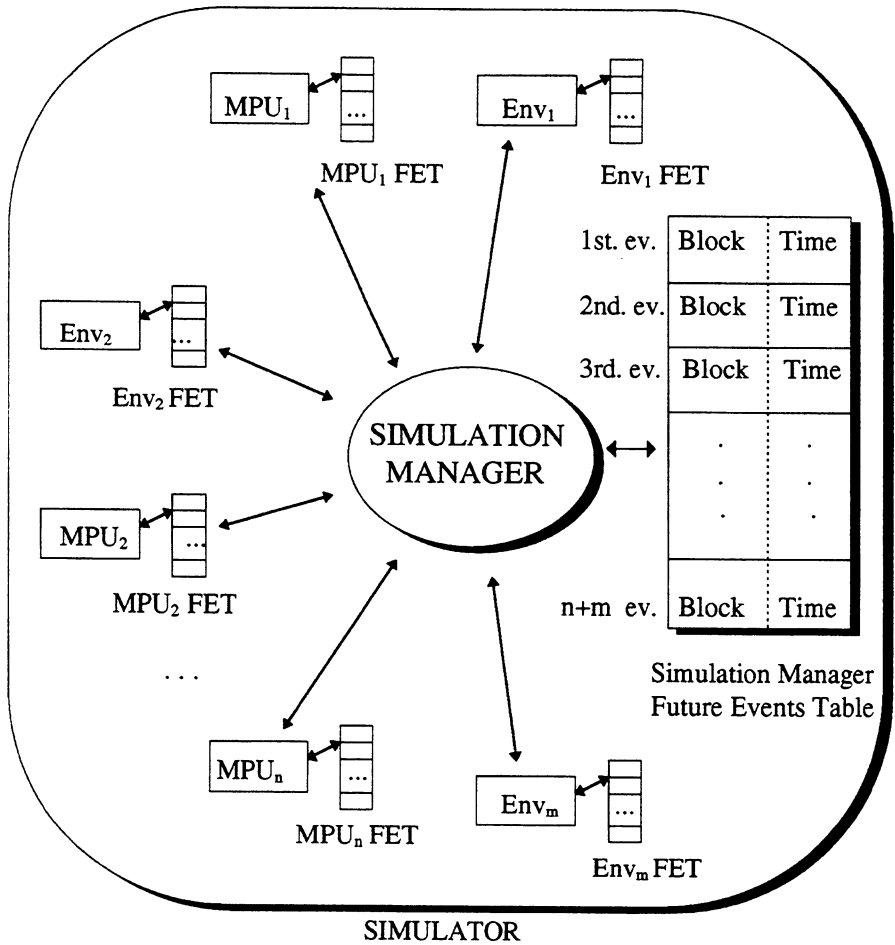
The complete simulation of the application is not restricted to a set of equations modeling the device and its inputs and the microcontroller itself. It may include more than one, even different, microprocessor and more than one set of equations describing different "boards".

Every set of equations has its own Future Events Table telling which signal is going to change, isolated from the other sets of equations. In the same way, every microcontroller has its own loaded program, its own registers, etc.

All the blocks made of microcontrollers or set of equations interchange information through their inputs and outputs. On the other hand, each block "exports" the top of its Future Events Table to the Simulation Manager Future Events Table, as it can be seen in Figure 2.

The Simulation Manager determines which is the first event that will take place and in which block. Control is transferred to the block having the earlier future event to simulate one step. The Manager and the affected blocks Future Events Tables, are recalculated and a new step of simulation is performed. The number of microcontroller blocks (n) and the environment blocks (m) may be different. Most likely m will be larger than n if there are present non

microcontroller based circuitry, or if the actual world surroundings (faraway environment) is described in a dedicated block.



MPU = Microcontroller Block

Env = Environment Block

FET = Future Events Table

Figure 2: Simulation Manager

3.0 Conclusions and Future Work

Because of the strong interaction between most microcontrollers and the environment it becomes necessary to model those interactions. A tool



representing the nearby and faraway environment constitutes a good solution to do this task.

This framework allows developer to validate the design's specifications without physical implementation. Then, the developer's task becomes easier. Furthermore, he can easily modify his design.

The process of automatic model transformation seems to be a valuable tool in hardware simulation.

In the near future, FORTH interpreters and compilers and C compilers are going to be included and all supported microcontrollers simulators are going to be joined in one generalized MPU simulator.

Bibliography

- [1] - D. Hamlet, "Theoretical Comparison of testing methods", in ACM SIGSOFT 3rd. Symp. Software Testing, Analysis, and Verification (de ACM Press), pp 28 to 37, *Proceedings of the ACM SIGSOFT 3rd. Symp. Software Testing, Analysis and Verification*, December 1989.
- [2] - P. G. Frankl, S. N. Weiss, "An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing", *IEEE Transactions on SOFTWARE ENGINEERING*, August 1993, Vol. 19, N.8, pp 774 to 787.
- [3] - M. Ridao, J. H. Doorn, "SAM: Un Lenguaje de Simulación de Hardware", in II Encuentro Chileno de Computación (ed Sociedad Chilena de Ciencias de la Computación), pp 169 to 176, *Actas del II Encuentro Chileno de Computación*, Concepción, Chile, 1994 .
- [4] - M. Ridao, J. H. Doorn, "V&V of microcontroller based systems", in Information Processing IP'94 (ed AMSE Press), pp 73 to 79, *International AMSE Conference 'Information Processing': Proceedings*, Vol. 2, New Orleans, USA, 1994.
- [5] - M. Ridao, J. H. Doorn, "Simulación de Hardware y de su interacción con el mundo real: Un enfoque tipo caja blanca", in SoftEng '91, Tandil, Argentina, 1991.
- [6] - R. Gabarrino, J. H. Doorn, M. Ridao, "Lenguaje de Simulación de Hardware", in II Encuentro Académico Tecnológico, Córdoba, Argentina, 1991.
- [7] - J.Reitman, "Computer simulation applications, discrete event simulation for synthesis and analysis of complex systems", ed R. E. Krieger Publishing Company, Chapter 1, pp 3 to 28, Chapter 2, pp 29 to 78, Chapter 5, pp 115 to 135, 1981.
- [8] - T. J. Schriber, "Simulating using G.P.S.S.", ed John Wiley & Sons, Chapter 2, *Basic GPSS Modeling Concepts*, pp 25 to 50, Chapter 3, *Sampling from Probability distributions in GPSS*, pp 143 to 164, 1974.



- [9] - S. Yakowitz, "Computational probability and simulation", ed Addison - Wesley Publishing Company, Chapter 2, pp 39 to 74, 1977.
- [10] -M. Ridao, J. H. Doorn, "Integración del diseño y la simulación de lógica digital", in BASE '92, Tandil, Argentina, 1992.
- [11] -M. A. Rodríguez, J. H. Doorn, "Modelización Automática de Sistemas Digitales", in II Encuentro Académico Tecnológico, Córdoba, Argentina, 1991.
- [12] -A. Law, D. Kelton, "Simulation modeling and analysis", ed Mc Graw - Hill, Chapter 3, pp 114 to 118, 1988.
- [13] -M. Ridao, J. H. Doorn, "Programación Transformacional en Ambientes de Simulación de Hardware" in III Encuentro Chileno de Computación (ed Leopoldo Bertossi, Pontificia Univ. Católica de Chile, Héctor Beck, Univ. de Tarapacá), pp 215 to 224, *Jornadas Chilenas de Computación, Actas del III Encuentro Chileno de Computación Arica, 30 de Octubre al 1 de Noviembre de 1995*, Arica, Chile, 1995.
- [14] -M. Ridao, J. H. Doorn, "Simulación Funcional de Lógica Digital", in Expociencia 25° Aniversario, Olavarría, Argentina, 1994.
- [15] -A. De Giusti, (ed EBAI Ediciones) "*Descripción y Validación de Hardware. Aplicaciones de tiempo real*", V EBAI, Nova Friburgo, Brasil, 1991.
- [16] -K. A. Kumar, B. Petrasko, "Designing a Custom DSP Circuit Using VHDL", *IEEE MICRO*, October 1990.
- [17] - J. M. Bergé, A.Fonkoua, S. Maginot, J. Rouillard, "*VHDL Designer's Reference*", Kluwer Academic Publishers, Dordrecht / Boston / London, 1992.
- [18] -M. A. Rodríguez, J. H. Doorn, "Consolidación de las Expansiones de Macros mediante Transformación de Programas" in SoftEng '91, Tandil, Argentina, 1991.
- [19] -F. Ingaglio, S.Dueñas, J. H. Doorn, "Ambiente de Desarrollo de Aplicaciones de Microcómputo" in XX JAIIO (ed Sociedad Argentina de Informática e Investigación Operativa), pp 1.2 to 1.16, *Anales de las 20° Jornadas Argentinas de Informática e Investigación Operativa*, Buenos Aires, Argentina, 1991.