

Constructing High Level Macrocell Models Using the Shlaer-Mellor Method

David Whipp

**GEC Plessey Semiconductors
Cheney Manor, Swindon, UK**

Shlaer-Mellor is a software engineering method for constructing models of systems. This paper describes how this method was used to construct a high level model of a PCMCIA macrocell. This model is then translated into a C++ implementation using automatic code generation. The paper contrasts the high level model with the hardware implementation of the cell.

Introduction

PCMCIA is an interface standard [1] for connecting peripheral cards to a host computer. The PCMCIA macrofunction is a reusable component, developed for the GPS SystemBuilder® library, that complies to this standard. The term “macrofunction” describes a block that has both hardware and software facets, which is used in conjunction with a programmable core such as an ARM processor. The PCMCIA macrofunction provides a bridge between the on-chip bus [2,7] and the PCMCIA bus standard.

When the macrofunction development was started, the complexity of the macrocell (the hardware component of the macrofunction) was estimated at 20,000 gates. Whilst considerably less than a typical processor core, this number of gates is significantly more than simpler cells such as serial port controllers. It seems reasonable to suppose that future developments will require even more complex macrocells [6].

Systems design using complex components requires models that abstract away the block’s design. Such models should abstract implementation details so that future redesigns of the block, or its design-level interface will not require the model to be changed. A “behavioural” model in an HDL, such as VHDL or Verilog, is inadequate because such models fail to fully abstract interfaces — they may abstract signal values, but not their meaning.

Constructing A Model

Shlaer-Mellor [3,4] is a software engineering method for modelling systems. Its distinguishing features from other software techniques are its emphasis on translation from a neutral model and explicit separation into domains of conceptually different subject matter. Although conceived as a software engineering method, its emphasis on translation from a neutral model, rather than focusing

on a specific implementation technology such as object-oriented design, makes it better suited to modelling hardware than those software methods that focus on a specific implementation technology such as object oriented design.

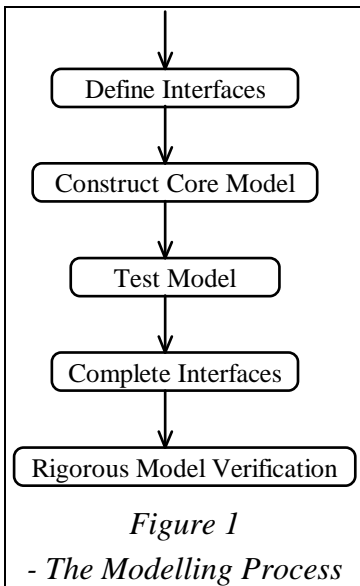


Figure 1 shows the steps used in the construction of a model. The initial step is to define the interfaces to the macrofunction. At this stage we are interested in the semantic atoms of communication, not the physical signals. The actual model will be built in terms of these atoms, not physical signals.

Although the method does not rely on object-oriented design, it does use object oriented ideas. The next step is to construct an “Object Oriented Analysis” of the macrocell. This is an executable model that communicates using the previously defined interfaces. This model can be tested; test cases can be defined and reviewers can carry out “walk-through” exercises to check the behaviour.

To complete the model, its semantic interface can be mapped to the design-level interfaces of the macrocell. In their simplest form, these signals may be characterised by their values and transitions; and the time at which the signal is sampled or the transition occurs.

Once the model has been connected to its physical interface, it can be connected to the same testbench that is used to verify the macrocell’s design. This is a rigorous check that the model and the design both exhibit the same behaviour.

The Interface Model

The purpose of the PCMCIA macrofunction is to connect the on-chip bus (B μ ILD) to the PCMCIA bus. This implies that the block must interface to each of these buses. To be more precise, the block acts as a PCMCIA slave, receiving access requests and satisfying these by acting as a B μ ILD bus master. The block must also act as a B μ ILD slave to allow other B μ ILD masters to access its control registers.

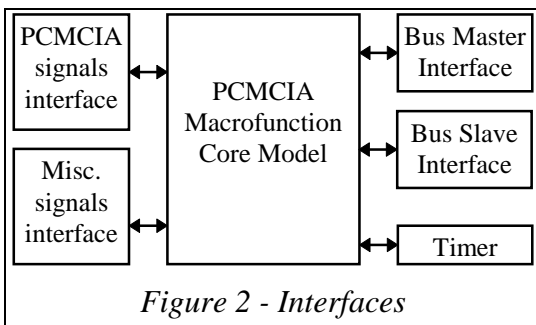


Figure 2 shows these interface blocks. The use of distinct interface blocks provides groups that may be encapsulated for a system-level netlist of a system that includes the macrofunction. For example, a system level description, e.g. [5], only needs to know what the on-chip bus is to enable automatic connection of all the bus signals, and possibly the bus-slave itself (I have written a script that uses a set of register definitions to expand an RTL VHDL B μ ILD-slave template).

A Modelling Example

In a Shlaer-Mellor model, an “Object Information Model” (OIM) describes the information that is needed to describe the macrofunction. This is not a model of the information stored by the macrofunction. An example from a B μ ILD-slave information model is given in Figure 3. Note that this should not be part of the core model, because it presumes a design based on registers.

This example shows a purely static model. Shlaer-Mellor allows each object to be associated with a state model that describes the dynamic behaviour

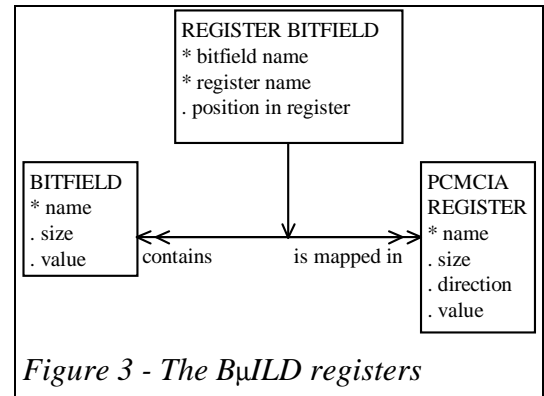


Figure 3 - The B μ ILD registers

Using the Model

A Shlaer-Mellor model is not intended to be used directly. It must first be translated into a programming language that can be compiled and then run. Simulation of the model is possible using CASE tools — these allow model verification via the use of highly instrumented code. The first stage of verification of the model was to use this simulator to run some simple test cases. When run within the simulator, the core interfaces can be used directly to communicate with the model.

To complete the verification, the model must be run with an independently produced testbench. This is the same testbench that is used to test the macrocell design - thus it uses the reified interfaces that have been mapped to physical signals.

The problem of the interface mapping is largely solved by existing hardware components and by components under development. For example, the hardware design of the macrocell produced a block which converts the asynchronous PCMCIA bus signals to a more friendly interface for use within the cell. This same block can be reused as one of the interface blocks to the model.

We therefore have a basic co-simulation capability. Higher simulation speeds can be obtained by discarding the simulator replacing it with c++ code that is automatically generated from the model. Yet higher speeds of simulation can be obtained by discarding VHDL entirely and running system models in a software environment. The speedup is obtained because two blocks are then able to communicate without the bottleneck of the physical interface signals.

Macrocell Design Structure

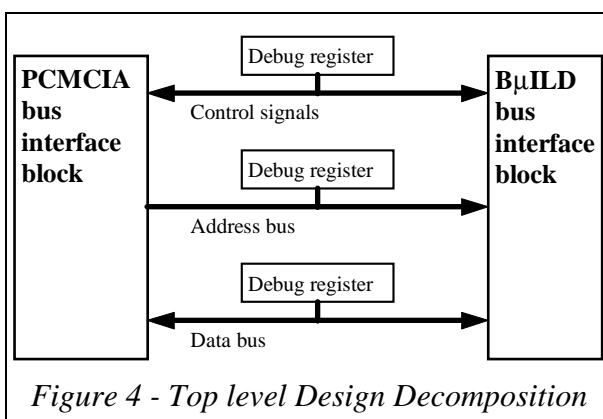


Figure 4 - Top level Design Decomposition

The purpose of the model is to provide a simple statement of what the macrocell does. The purpose of a design is to describe how the behaviour can be implemented in a given technology. The two descriptions were developed independently from a detailed implementation specification.

The differences between the model and design start from the initial decomposition. The initial structuring of the model was to identify all the interfaces. The design splits these out over several levels of hierarchy. Figure 4 shows the top level

decomposition of the macrocell design. The basis for this structure is the implementation of the blocks. The PCMCIA interface must consist of asynchronous logic whilst the remainder of the block is synchronised with the system clock.

It is common (and correct) for a design to focus on implementation issues. Unfortunately, a correct structure for implementation is not always the best structure for understandability. The hardware design has a deep hierarchical structure whilst the software model is much flatter.

Although organised differently, hierarchical blocks within the hardware design can usually be found to correspond to features of the model. The different organisational principles are probably best illustrated by the timer interface. The high level model eliminates time from the core behaviour so a time-out requires the explicit use of the timer interface. In the model, all signal synchronisation occurs within the interface blocks, not the core.

In the design, the situation is different. Hardware design is explicitly synchronous so a clock signal is distributed throughout the design. As a result there is no explicit timer interface — a counter can be included at the lowest levels in the design that is synchronised to the clock.

Although the clock is available throughout the block, a separation between behaviour and synchronisation can still be found. In VHDL it is common practice to describe state machines as two processes: one to define the combinatorial logic and the other to provide the next-state assignment. Thus the distinction between core behaviour and interface synchronisation is distributed throughout the design instead of being isolated in a few well-defined blocks.

Conclusions

The Shlaer-Mellor method provides a means of modelling macrocells that abstracts the interface away from design-level signals. This provides a greater separation between model and implementation than can be achieved through the use of type-systems, such as those used by behavioural HDLs. These models provide 4 key advantages over hierarchical design blocks:

- increased understanding of the operation of the component,
- isolation of the component from implementation technology, including that of the interfaces,
- faster simulation through reduction of detail,
- protection of Intellectual Property — there is no need to distribute designs, only models.

References

- [1] *The Personal Computer Memory Card International Association, "The PCMCIA Standard", February 1995.*
- [2] *Tim Frost, "BμILD-2: Robust Microcontroller Architecture", 1996. ref. RDS/ARM/156.06.2.*
- [3] *Sally Shlaer and Steve Mellor, "Object Lifecycles: Modelling the World in states", Prentice Hall, 1992.*
- [4] *www.projtech.com - Up to date information about the Shlaer-Mellor method.*
- [5] *Derrick Morris et al, "Model Based Object Oriented Systems Engineering", Springer-Verlag, 1996.*
- [6] *www.cfi.org/roadmap/roadmapHomePage.html, "The EDA Industry Standards Roadmap".*
- [7] *"AMBA: Advanced Microcontroller Bus Architecture Specification", 1995. Ref. ARM Ltd, ARM IHI 0001C*