Top-Down Design Using Cycle Based Simulation: an MPEG A/V Decoder Example

Dale E. Hocevar, Ching-Yu Hung, Dan Pickens and Sundararajan Sriram
DSPS R&D Center, Texas Instruments, Inc.
P.O. Box 655474, MS 446
Dallas, TX 75265

Abstract

This paper presents a discussion of a top-down VLSI design approach which involves system level performance modeling, block level cycle based simulation, RTL/VHDL simulation and gate level emulation. An MPEG-2 Audio/Video decoder design example illustrates the use of this top-down approach. Most of the discussion concentrates on the concept of block level cycle based (BLCB) simulation. HW/SW co-design also played an important role in this work and our approach towards such co-design is discussed as well.

1. Introduction

In the past several years there has been a dramatic increase in the structural complexity of VLSI designs. Many of the modern CAD tools have advanced and can handle these large designs from a structural viewpoint. In addition, the behavioral complexity has dramatically increased as well and even though simulation capabilities have improved considerably, and new higher level simulation capabilities have become available, it is still difficult to deal with this high behavioral complexity. Structured top-down design offers a solution but has not been widely adopted. Partially this is due to the reluctance of design engineers to change their ways and to a lack of perceived benefit. In this paper we discuss an approach we developed for top-down design and which we applied in an actual design, a complex signal processing device.

One problem occurring in VLSI design stems from the architectural design being done without using any simulation model to verify the analysis and to address the high level communication, interface and synchronization issues. Another related problem occurs after this architectural stage when the engineers often proceed immediately to the sole task of designing their particular modules. These are later pieced together and made to obtain the correct system function with much pain and effort, a task that can take months. The reason for this difficulty is because many system level issues, control and synchronization in particular, and the top level interfaces have been left to be determined late in the design cycle when the simulation models are very detailed. At that point it can only be a slow and costly process to complete the design.

One solution to this problem is to apply a structured top-down design methodology, as mentioned above. We have devised such an approach for our particular situation and applied it to the design of a combined audio/video decoder for MPEG-2, [1]. Such an approach can greatly reduce the overall design time because much less time is required for the later debugging phases.

This approach consists of building a high level system model for designing the required system operation, synchronization and data flow between top level modules. Called the performance model or performance simulation, it also provides many performance measures for assisting during the architecture evaluation. Next, the concept of block level cycle based (BLCB) simulation was utilized taking the design another step lower in the process and through which considerably more design detail was added. The performance model was transformed into this BLCB model. This overall process entailed some novel approaches toward HW/SW co-design. The next phase in this process was to complete the synthesizable RTL descriptions for the modules and to verify the complete VHDL simulation model.

This paper discusses this overall approach and its application to the A/V MPEG decoder design, mainly concentrating on the BLCB model.

2. Decoder Architecture and Performance Model

The architecture for the audio/video decoder which utilized this design approach is shown in Figure 1. It con-

0-8186-8409-7/98 $10.00 © 1998 IEEE

sists of an MPEG-2, main level, main profile, video decoder, and associated audio decoder which is mostly self-contained. The video decoder uses an ARM7 processor for overall control and general processing. The bulk of the video data passes only through the video data-path starting with the Variable Length Decoder (VLD) and ending with the Video Display Unit and using the external SDRAM for temporary storage along the way.

In order to validate the architecture, develop the data flow synchronization, and resolve many system level issues, a performance simulation model was built. This model operated asynchronously at the level of block data transfers and did not actually pass or process video data. It did however actually parse the bitstream to allow for correct modeling of the data dependent operations such as the amount of VLD processing and the amount of motion compensation for each macroblock. This model provided extremely fast simulations, several frames a second, and allowed us to complete the system level design with the assurance that when the design was built it would operate correctly. Figure 2 shows the performance model for this decoder which was written in C++ using the simulation package CSIM from MCC [2]. All rounded corner boxes are "processes" operating concurrently in the simulation; note that those in the CPU represent software tasks. This model is discussed in more detail in [3].

3. Block Level Cycle Based Simulation

Cycle based simulation is an approach used to model synchronous hardware in a simplified manner in order to significantly increase the speed of simulation. The simulation computation occurs for all hardware pieces once for each clock edge; for our purposes just the rising edge. The computation involves determining the next values of the stored signals (registers and memories) from the present values of the stored signals and inputs, and then updating the storage elements. The hardware consists of logic portions and storage portions, and the logic portions must be evaluated as part of this computation on each cycle.

Block level cycle based (BLCB) simulation is essentially this same concept except that the design is presented as large modules which are not described as circuit logic, either structurally or behaviorally. Rather they are modeled at a higher behavioral level and are written in a high level procedural language such as C or C++. When such a block is executed at each clock, it utilizes its present interface signal values and returns its next clock signal values. The modeling internally can be accomplished in several ways as will be discussed later. It should be mentioned that though the blocks are modeled at a high level they are a functionally correct model of the actual hardware logic.

This block level design model then provides an exact description of the top level of signals; i.e., the signals interfacing between the blocks. This description is not just

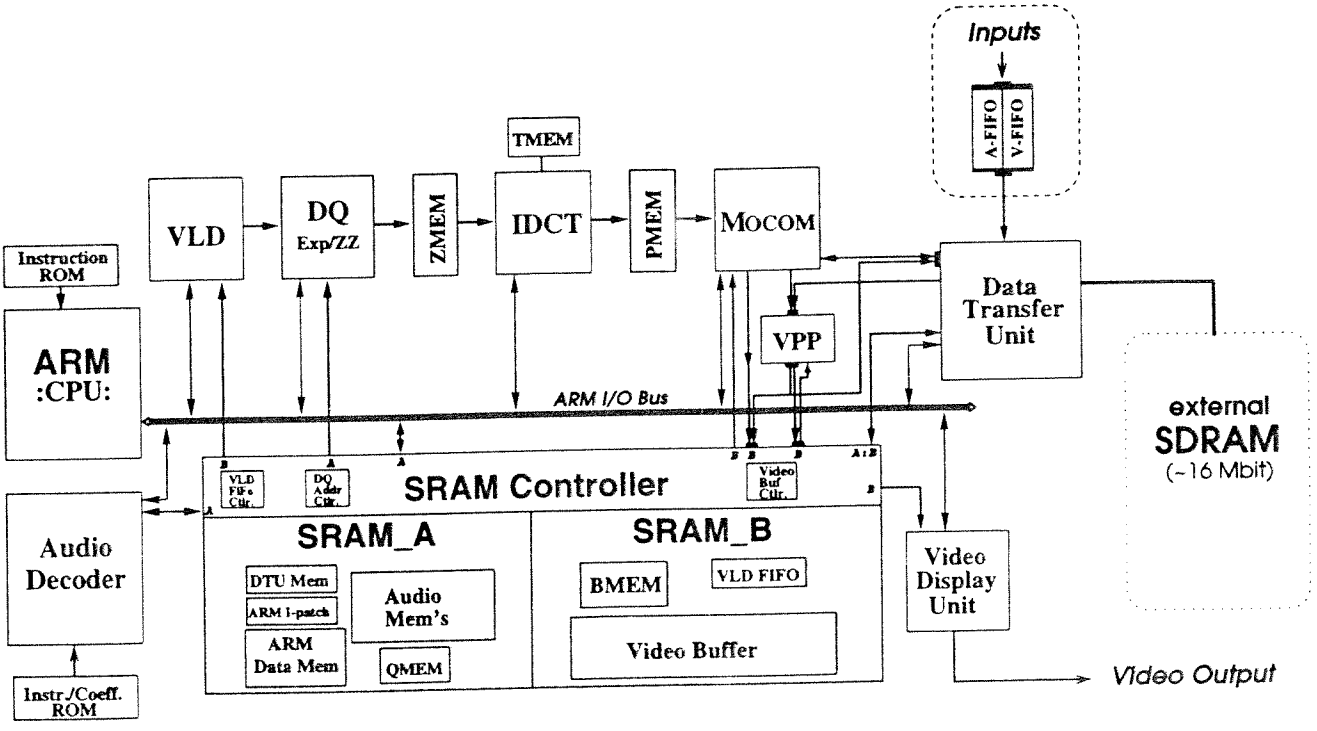

Figure 1. Hardware architecture for MPEG A/V decoder design.

structural, but is also behavioral in that for any set of device input signals, simulation of the model provides the clock by clock values for these top level signals.

The advantage of this block level cycle based simulation is that it provides a means for performing top-down design. One facet is that it allows the engineers to develop a very detailed simulation model, one which is actually a description or an encapsulation of the design down to this level. It also provides for very fast simulation which helps this model to be built more quickly. Once the model is constructed, it then drives the next stages of design in an effective manner.

3.1 Implementing the BLCB Model

Our initial version of the BLCB simulator model was based upon a simple transformation of the performance simulation model. This involved the introduction of a clock signal and the translation of all higher level, more abstract, CSIM synchronization mechanisms, such as events, mailboxes, event arrays, and facilities, into circuit signal mechanisms which were aligned with the clock.

Event mechanisms were handled as follows. Suppose we have event modA_snd, with some modules setting it as, modA_snd.set(), and some waiting on it to be set as, modA_snd.wait(). A new C++ variable (array of length two) would be created called modA_snd_EVT and the following changes would be made.

```
modA_snd.set()    <-->   modA_snd_EVT[NV] = 1;
```

and

```
modA_snd.wait()   <-->   while( !modA_snd_EVT[PV] )
                             clock.wait();
```

Here PV is the present value index of 0, and NV is the next value index of 1.

Event arrays were handled as an extension of the above method. Mailbox mechanisms were also handled in a similar fashion except that a count variable was maintained for the number of items sent but not yet received. This works only for mailbox constructs where the message content is nil.

This approach for transforming the performance model into the initial cycle based model works very well. We completed the decoder transformation in only one man-week. The biggest difficulty is in determining when to insert extra clock waits to allow the overall synchronization operations to work properly due to the change from asynchronous operation to synchronous.

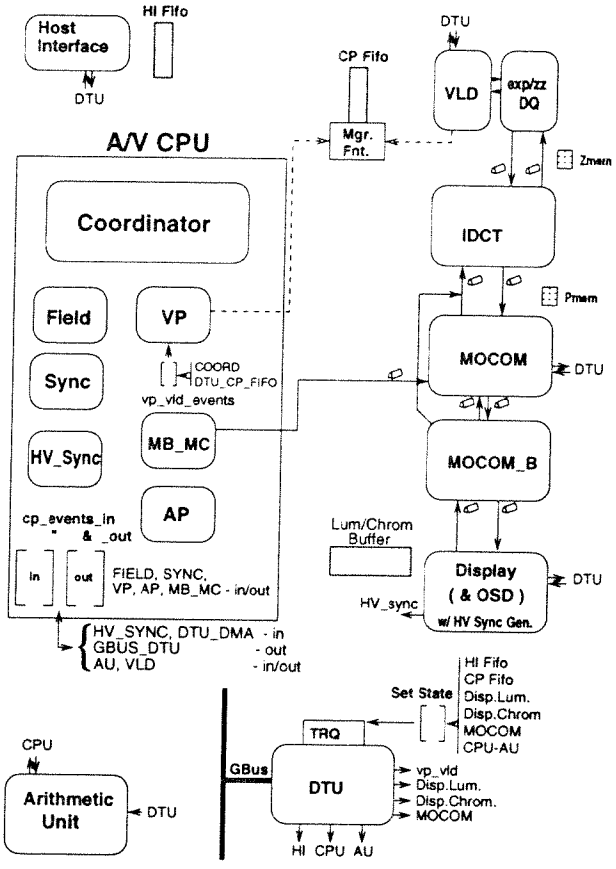

Figure 2. Schematic of performance simulation model for A/V decoder design.

3.2 Module Coding Approaches

As the design progresses, more detail is added to the modules, specifically for the signal interfaces. Hardware modules can be coded in a fairly free manner, similar to VHDL, but with the greater ease of C++. Since clock synchronization is obtained anywhere in the code by placing a clock.wait() function, the code can be structured with loops that execute on each clock and/or the clock.wait() functions can be distributed throughout the code section.

An extension of this which offers great flexibility is to use per cycle functions. In this case, the module takes the form of a function call which is invoked after each clock in an unending loop.

One example of this is for very large modules such as a CPU, here the per cycle function could invoke an intricate model of the CPU on a cycle basis. The CPU model itself could be implemented as a detailed event driven simulation model which simply simulates one clock cycle on each invocation by processing the necessary internal events in its

queue. The ARM CPU in our decoder was incorporated into our cycle based model in exactly this manner.

Another example concerns RTL modeling. Small to medium size modules can be modeled at the level of RTL through simple C coding methods. In the past (before VHDL or Verilog were commonplace), many designers developed the initial version of their designs with this type of modeling. The code is simply structured such that the top level function executes the RTL model for one clock cycle.

Such an RTL model for a block would contain much more design detail than the more abstract, behavioral approaches that could be used just to obtain correct signal interface and functional behaviors. One key feature that this approach provides is the ability to add more design detail to the overall design. More specifically, at later stages in the design process the RTL level can be designed, block by block, through replacing the behaviorally modeled blocks with their new RTL block descriptions.

3.3 Comparison to VHDL/Verilog

The approach we have taken towards block level simulation using C++ and CSIM could also have been implemented using VHDL or Verilog. If one is starting a new design this would be an option to consider. One advantage of VHDL and Verilog is that they directly contain the hardware signal mechanisms.

If one is building a dedicated design, with limited programmability, and the algorithms being implemented are of significant complexity, then our C/C++ approach using CSIM has some advantages. Especially if the algorithms need to be implemented first in a high level language in order to fully understand the design issues. The advantages are basically the ease of coding and the execution speed. HDLs are not very friendly for coding complex algorithms at the behavioral level. In addition, often an implementation of the algorithms may exist already in C/C++, in which case it allows for moving more quickly into such a model. Furthermore, if basic design correctness requires repeated simulations, each using millions of clock cycles, such as video decoders, then the speed advantage of the more streamlined CSIM approach is very advantageous.

4. BLCB Simulation and A/V Decoder Design

After the initial BLCB model was obtained from the performance simulation model, the next phase entailed designing the interface signals and incorporating the functional computation behaviors.

4.1 Design of Module Interface Signaling

The next step was to develop the signal interface mechanisms between modules and implement these into the simulation model. Existing signals from the performance model, which operated at an abstract, non-physical level, were gradually replaced with the newly designed, physical signals.

This design process progressed block by block through the design, all the while maintaining a correctly functioning model. The development started at the front of the video data path, the VLD, and progressed along the path to the Video Display Unit. For some modules not all the signals were implemented the first pass through, specifically, those dealing with the ARM bus and the centralized SRAMs were done later. This illustrates the flexibility of this approach.

The original performance model did not perform actual data computations on the bitstream data past the VLD. Thus, for all modules downstream, the algorithm computation was now also implemented into the modules. Of course, this was done behaviorally, not reflecting the RTL hardware to much degree.

The development process described above has some important advantages. It allows one to maintain a fully working model which decodes bitstreams through several picture frames (though perhaps without all the computation being carried out) and to step by step add more detail to that model. This is continued until eventually all the module interface signals are completed and full data computation is incorporated. The model can be tested with all types of bitstreams until correct functionality is achieved. This is much easier than later piecing together modules designed by different engineers in an HDL at the RTL or synthesizable level, and then working out the problems with slow and detailed simulations.

At this point in our process, the full ASIC design could be considered well over half way finished. The largest remaining work would be to complete the RTL and synthesis of the modules. But it should be remembered that during the architecture design and performance modeling phases that much of the internal module architectures, including rough RTL modeling, has already been devised. This is necessary to obtain reasonable size estimates and to determine feasibility of operation at the required speeds.

4.2 HW/SW Co-Design

Design of the video decoder also involved performing HW/SW co-design and software development for the ARM processor. The performance simulator modeled the CPU's

functions in an abstract manner as discussed in [3]. Though this modeling included parsing the bitstream, computing motion vectors, controlling the overall decoding process, etc., this was accomplished through modeling, not via software running on the ARM processor. Thus, for the BLCB model the ARM software needed to be developed.

Most of the code development was done using the C language; though, later some assembly language was used, in particular for the interrupt codes. TI's instruction level simulator and C source debugger for the ARM7 processor were utilized to develop this software.

The initial development on the ARM simulator was totally independent from the module development discussed above. The converted performance simulator's CPU model sufficed to drive the hardware development in the BLCB model to near completion. Also, the first phases of software development required only limited interaction with the non-ARM hardware, and this was easily modeled.

Later in the process, a cycle based, signal accurate, ARM simulator model (also available within TI but different from the instruction simulator) was merged into the BLCB decoder model. At the same time the abstract CPU modeling was removed. It was interfaced basically via the per cycle function method mentioned earlier. Though conceptually this is a simple step, in practice it requires some effort. To simplify the process this ARM model was first interfaced with a simple cycle based test bed, thus temporarily eliminating the complexity of the decoder model.

Once the ARM model was incorporated, its software could be run in conjunction with the actual hardware modules and further design continued. It only took a couple of weeks before entire picture frames could be correctly decoded with the full BLCB model.

5. Further Levels of Top-Down Design

The next stage of design involves the continued development and refinement of the software, and the design of synthesizable RTL for the hardware modules. At this point, for each module the designer has an exact executable behavioral description as well as the ability to capture all the interface signal values. This allows any particular module to be further designed by reworking the description down to an RTL mode in C++ and resimulated within the block level model. Or, the module RTL description could be worked out in VHDL/Verilog first, using the saved interface signals as test vectors. VHDL simulation of the entire design would not be necessary until the modules are completed and verified at the level of RTL, using this block level cycle based model. At this point, the design can be moved to Quickturn type emulation fairly quickly. The

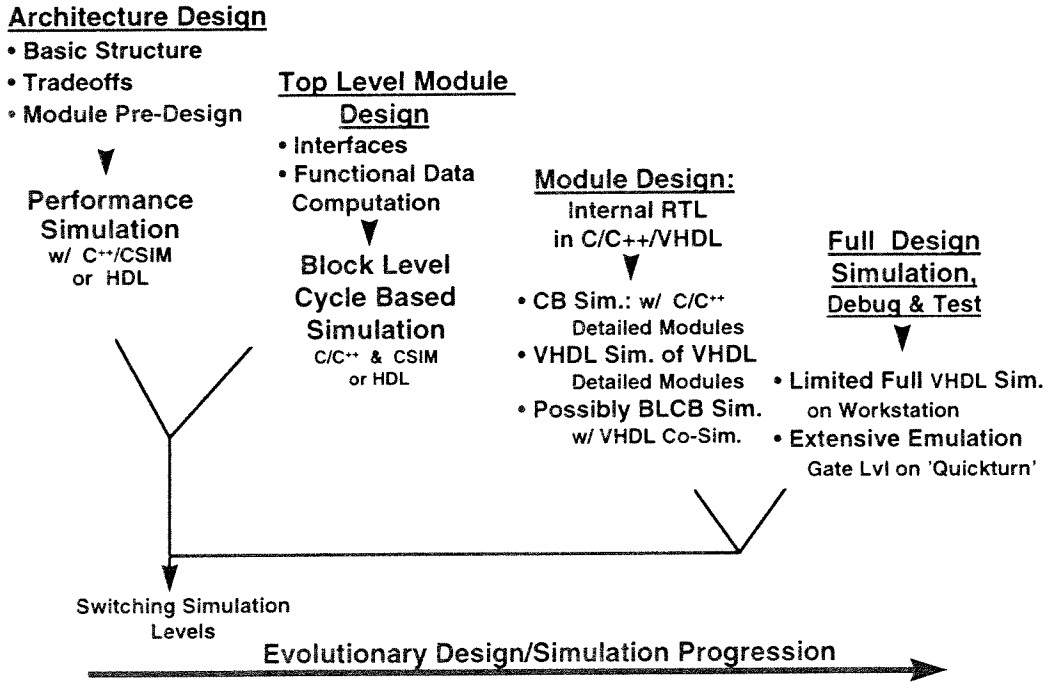

Figure 3. Top-down design levels and simulation models.

overall top-down design concept is shown in Figure 3.

This is opposed to the less structured approach of designing the modules separately then merging them together and performing full design simulation debug on the large detailed design. In that approach there are typically numerous issues involving signal interfaces and synchronization, etc., which were not thought out previously and now need to be worked out. This is a slow and painful procedure, in fact, this phase alone can take months to complete. The structured top-down approach described in this paper eliminates this problem.

6. Conclusions

This paper discussed an approach towards structured top-down design, specifically as demonstrated through the design of an MPEG-2 A/V decoder. Concentration was mainly on the concept and approach for building a block level cycle based simulation model.

The overall top-down approach begins with the construction of a performance simulation model [3], and provides a means for performing top-down design in a stepwise manner whereby over time more and more detail is added to the model. If desired, when the model is taken down to the RTL level, this can be done essentially within the same environment, before using an HDL.

There are many advantages to this approach. Specifically, the performance simulator allows for an early simulation model which encompasses the entire design behaviorally. This can be used to verify architectural and system level issues, and to develop the basic operational details such as control and module synchronization, etc. It can also be devised to provide whatever amount of functional data is desired at this level. The block level cycle based simulation model then provides a means for developing much of the module level detail, including the interface signals and mechanisms, as well as the internal functional operation. This code, together with the architecture design documents (which would have been produced), form a detailed specification for each module. Also, if there is an embedded processor, its application software would have been developed with the block level model so that full functionality could be achieved. Lastly, the overall block level model provides an exact model of the device which has the same functional and clock cycle behavior that the final ASIC will have. If desired, the module's detail can be developed down to the RTL level (in C) within the block level model. Alternatively, this can be done in VHDL using the block level model to provide any type of test vectors desired around the module's boundary, thus greatly easing the design time and effort.

References

[1] T. Sikora, "MPEG Digital Video-Coding Standards," *IEEE Signal Processing*, Vol. 14, No. 5, pp. 82-100, Sept. 1997.

[2] H. Schwetman, "Using CSIM to model complex systems," *Proc. of the 1988 Winter Simulation Conference*, pp. 246-253, 1988.

[3] Dale E. Hocevar, Sundararajan Sriram, and Ching-Yu Hung, "Performance Modeling for System Design: An MPEG A/V Decoder Example," *IEEE Int. Sym. Circuits & Sys.*, June 1998.