Experience in Designing a Large-scale Multiprocessor using Field-Programmable Devices and Advanced CAD Tools

S. Brown, N. Manjikian, Z. Vranesic, S. Caranci, A. Grbic, R. Grindley, M. Gusat, K. Loveless, Z. Zilic, and S. Srbljic
Dept. of Electrical and Computer Engineering, University of Toronto, Canada
Email: brown@eecg.toronto.edu

Abstract

This paper provides a case study that shows how a demanding application stresses the capabilities of today's CAD tools, especially in the integration of products from multiple vendors. We relate our experiences in the design of a large, high-speed multiprocessor computer, using state of the art CAD tools. All logic circuitry is targeted to field-programmable devices (FPDs). This choice amplifies the difficulties associated with achieving a high-speed design, and places extra requirements on the CAD tools. Two main CAD systems are discussed in the paper: Cadence Logic Workbench (LWB) is employed for board-level design, and Altera MAX+plusII is used for implementation of logic circuits in FPDs. Each of these products is of great value for our project, but the integration of the two is less than satisfactory. The paper describes a custom procedure that we developed for integrating sub-designs realized in FPDs (via MAX+plusII) into our board-level designs in LWB. We also discuss experiences with Logic Modelling Smart Models, for simulation of FPDs and other types of chips.

1 Introduction

This paper describes experiences designing a large-scale multiprocessor computer using state-of-the-art CAD for: (1) high-level definition of the computer architecture, (2) detailed design entry of prototype modules, (3) implementation of control circuitry in field-programmable devices, (4) chip-level and board-level simulation, and (5) printed circuit board layout. The focus of the paper is on problems encountered because of limitations of the CAD tools and difficulties of integrating CAD products from multiple vendors.

Our silicon strategy is based on two types of field-programmable devices (FPDs): field-programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs). These high-capacity user-programmable devices were selected for their practical advantages: low cost, instant manufacturing turnaround, and ease of design modification. However, FPDs complicate the design in two ways: (1) they are slower than custom-designed alternatives, requiring more careful design techniques to achieve the desired speed requirements, and (2) they have lower density than other technologies, and hence consume more board area. In addition, and particularly relevant for this paper, FPD-based design requires a separate set of CAD tools, and we encountered major problems with the integration of these tools with our main CAD system.

A major challenge of our multiprocessor project is achieving the goal of a 50 MHz clock rate for all system components, and 200 MHz for the processors. The difficulties of reaching such speeds is exacerbated by our decision to use FPGAs and CPLDs. The paper discusses these issues, such as the need for hand-tuning of designs to assist the CAD tools, and the difficulty of obtaining simulation models for the most recent, fastest FPDs.

The paper is organized as follows. Section 2 provides an overview of our CAD flow, and gives a general evaluation of the major products used. In Section 3 the multiprocessor being designed is briefly described. Section 4 focuses on design entry at both the board and chip (FPD) levels, and Section 5 discusses our simulation strategy. Section 6 is dedicated to PCB layout, and Section 7 concludes.

2 Overview of CAD Tools

An overview of our CAD flow is shown in Fig. 1. The starting point is a custom-built multiprocessor evaluation tool that we created to explore various architectural parameters of the machine. After selection of parameters for a prototype implementation, the machine was manually partitioned into multiple PCBs. As the figure shows, for each PCB there are two distinct types of components: (i) commodity chips that are simply entered into a board-level schematic, and (ii) FPDs to hold custom-designed logic circuitry. For (i) there is a direct path to Cadence Logic WorkBench (LWB) [1]. LWB is the main tool in the CAD flow, providing design entry, simulation, and layout of PCBs. Cadence Concept was used to create a schematic of each board, and the boards were simulated with the OpenSim backplane, which can link together different types of simulation models for various chips. Most chips were simulated using Verilog timing models, but for some devices Logic Modelling Corp.'s[1] Smart Models [2] library was employed. Final PCB layout was accomplished with Cadence Allegro.

For components designated as (ii) above, a separate CAD system was needed for logic design. The FPDs used include both SRAM-based FPGAs and EEPROM-based CPLDs from Altera Corp. [3]. Altera's MAX+plusII CAD system was used for design entry of all logic circuitry, simulation and debugging of each FPD, generation of Verilog for board-level simulation in LWB, and output of bit-patterns for programming the FPGAs and CPLDs. Although MAX+plusII is also capable of automatically partitioning logic into multiple devices, we chose to do this by hand to better control the assignment of board-level signals to individual chips. Integration of MAX+plusII with LWB proved to be difficult, as discussed in Section 4.

The following subsections provide general comments on our experiences with each of the CAD tools in Fig. 1.

2.1 Cadence Logic WorkBench

LWB is a state-of-the-art CAD system for physical implementation of digital systems. Based on our experience, a few preliminary comments on this product are given below:

- LWB is comprehensive, offering most of the required design tools. However, because LWB includes a large number of inter-related options that can subtly affect performance, the learning curve is steep and several months are needed to become productive with the entire package.

1. Logic Modelling Corp. merged with Synopsys in 1994.

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

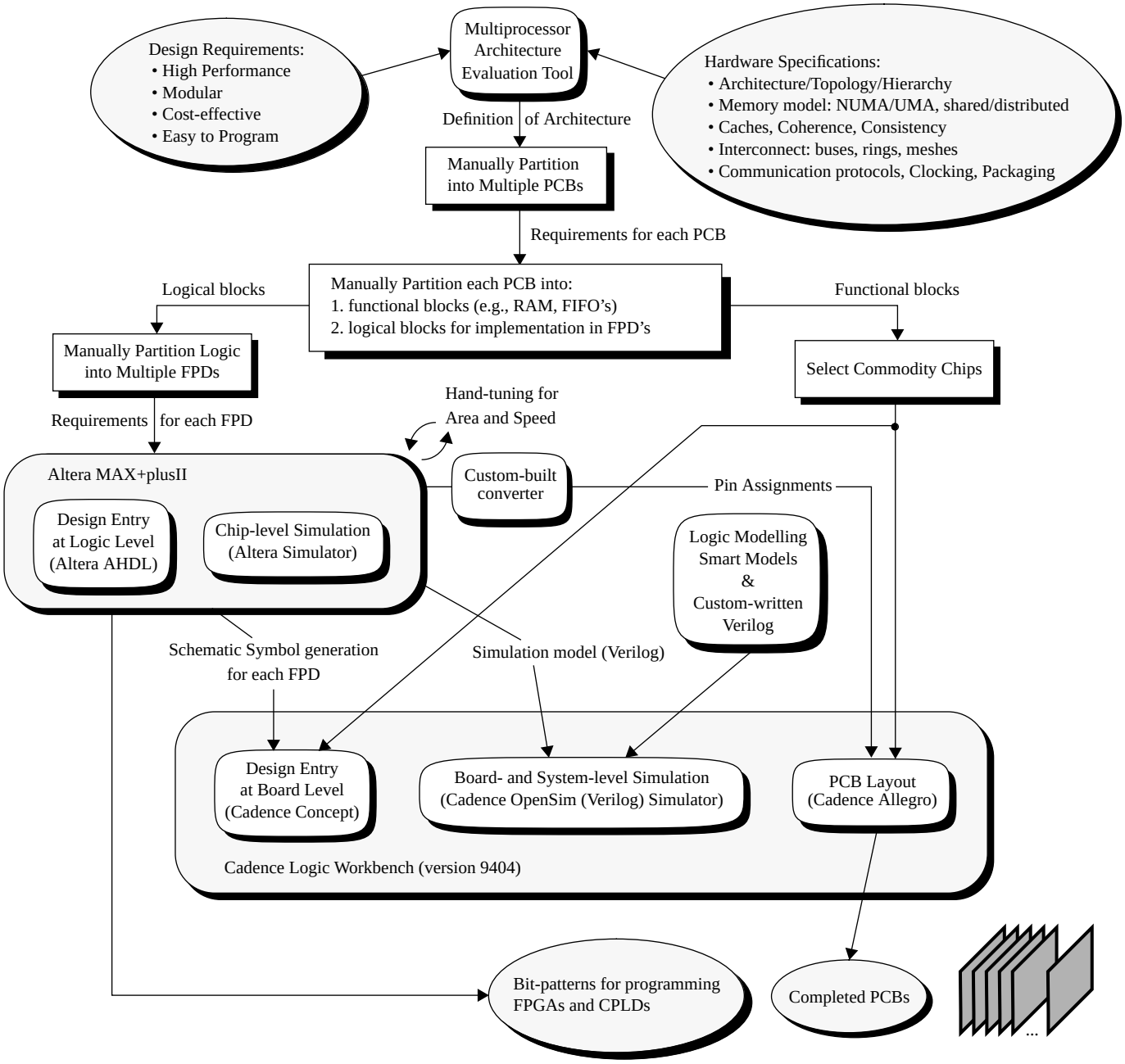

Figure 1 - *Overview of CAD System.*

- Technical support for LWB itself is superb. The easiest way to get help is to send Cadence electronic mail, which is usually answered quickly. However, we did not experience the same level of assistance for inquires related to interfacing LWB to *third-party* software.
- The quality of on-line documentation for LWB varies. Some packages were acquired from other organizations that were once separate entities, but are now part of Cadence, and some of the documentation has not been updated to match the LWB environment. Also, some documentation is duplicated (with differing content) and some documentation is present for non-existent products.

2.2 Altera MAX+plusII

MAX+plusII, as shown in Fig. 2, comprises all tools needed to implement circuits in Altera devices. Design entry may be performed using schematic capture, waveform entry, Altera HDL (AHDL), VHDL, or Verilog. Designs are automatically mapped into any Altera FPD and full timing simulation is available. Some general comments on this system are listed below:

- MAX+plusII is easy to learn, easy to use, and powerful. On-line help is excellent.
- Technical support from Altera is inconsistent; we were sometimes able to get quick answers to questions, but support is lacking with respect to the use of MAX+plusII with LWB.

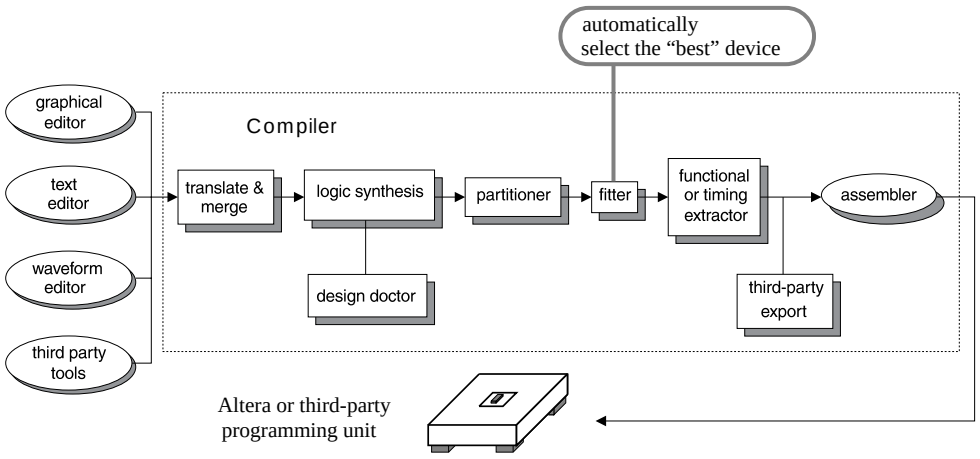

Figure 2 - *Altera MAX+plusII CAD System.*

- Both combinational and sequential logic optimization is provided, and is useful; however, the generated state machine assignments are usually not as efficient as manual ones.

2.3 Logic Modelling Smart Models Library

Smart Models is a library of simulation models for a wide assortment of commercially-available chips. These models may be incorporated into the LWB environment for simulation purposes. The following list summarizes the problems that we encountered:

- It was difficult to obtain answers to technical questions on this product (although there have been recent improvements, since data sheets are now available on the WWW).
- For chips that are included in the library, simulation with Smart Models is convenient. However, models are not available for recently-introduced FPDs. The latest FPD models in our (1994) Smart Models library are from 1991, whereas the parts we selected in our design were introduced in 1994-95.
- Furthermore, models for some commodity chips required in our design are also not available in the Smart Models library, as the models may lag a few years behind chip release.

3 The NUMAchine Multiprocessor

The system being designed is a large-scale shared-memory multiprocessor, called *NUMAchine*, comprising a number of processors connected to one another in a tightly-coupled fashion. The multiprocessor implements a NUMA (Non-Uniform Memory Access time) architecture in which the memory is physically distributed and the amount of time needed for a particular processor to access a specific memory module varies. The machine includes multiple levels of cache memory with hardware maintaining data coherence and consistency throughout the memory hierarchy. Designing a computer of this class is challenging, but such machines are of considerable commercial interest [4, 5].

The NUMAchine architecture consists of a hierarchical arrangement of *stations*, where each station comprises a number of processors (we chose the MIPS R4400), memory module(s), and I/O sub-system(s), connected via a bus. Multiple stations are interconnected through a hierarchy of bit-parallel slotted *rings*. Due to space constraints we will not provide further details on the NUMAchine architecture, but interested readers are encouraged to refer to our technical report [6].

This discussion will focus on the design of a single station, which comprises separate boards for each processor (including cache), memory module, I/O subsystem, and station-to-ring interface (with network-level cache). We selected the physical structure of the FutureBus+ standard for the station bus because of its high-speed transfer capability, but we devised our own bus control protocol. The physical appearance of a station is illustrated in Fig. 3.

Since there are many variable parameters in the system described above, we developed a custom multiprocessor evaluation tool, as mentioned in Section 2, to determine appropriate parameters for a prototype machine. This stage was the only one in which we did not use commercially available tools, because no product existed that suited our particular needs. As illustrated in Fig. 4, the inputs to the evaluation tool are of two types: (1) an R4400 parallel program (i.e., a real or synthetic benchmark), and (2) a file specifying speed, size and number of all system components: processors, caches, memory modules, buses, rings, etc.

The evaluation tool uses a separate public-domain instruction-level simulator called MINT [6] as a front-end to an architectural simulator developed in-house. MINT generates memory access and other traffic for the custom simulator, which then models the system at a cycle level and simulates the processors’ actions,

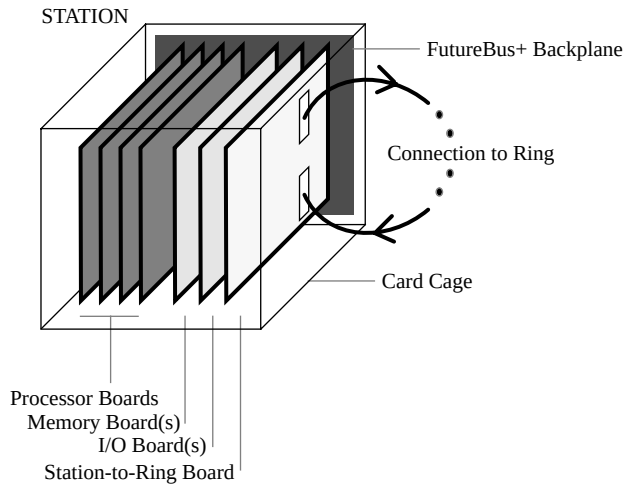

Figure 3 - *Partitioning of Prototype into Printed Circuit Boards.*

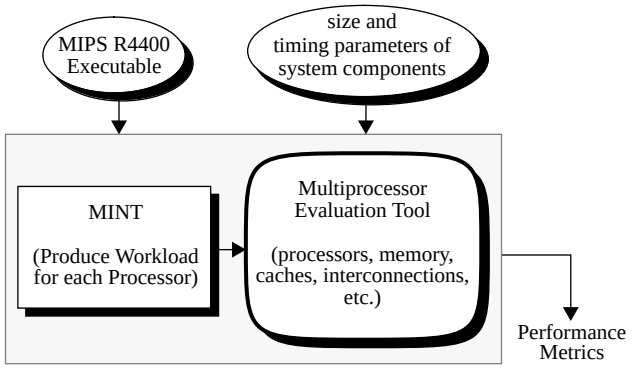

Figure 4 - *Custom-built Multiprocessor Evaluation Tool.*

latency to and from different parts of the memory hierarchy (memory, primary, secondary and network-level caches), arbitration for shared resources (such as buses), and synchronization operations. This allows measurement of both high-level parameters, such as bus and network utilization and contention, and lower-level details such as average and maximum queue depths for the various FIFOs used in the system. These performance metrics guided our choice of parameters for a prototype implementation.

4 Experiences in Logic Design

Our design strategy "on paper" for each PCB was done in a top-down fashion: each board was first partitioned into two categories of components: logical blocks for which we needed to design logic circuitry (FPDs), and functional blocks for which commodity chips could be used (e.g., RAM). The logical blocks were then further refined into one or more FPDs, and the functional blocks were partitioned into individual off-the-shelf chips. Our preference was to implement this top-down strategy with the CAD tools, using LWB to create a top-level schematic of each board, and then automatically interfacing to MAX+plusII for design of the logical blocks. Unfortunately, we were not able to use the tools in this manner because of problems with their integration, which is discussed in Section 4.1. Therefore, our actual CAD flow was "bottom-up," as depicted in Fig. 1. Since the process was essentially the same for all PCBs, our experiences will be related using the processor board as a representative example. The block diagram of a processor board appears in Fig. 5. It includes: (1) a MIPS R4400 processor with an external 1-MByte secondary cache, (2) a complex circuit known as the "external agent" for interfacing the processor to the rest of the system, (3) circuitry required to realize a "local bus" that connects simple devices like EPROMs and serial I/O ports to the processor, (4) special-purpose registers for interrupts, barriers, and performance monitoring, (5) FIFOs, bus control, and FutureBus+ standard-compliant buffers to interface the processor and the station bus, and (6) various connectors. In Fig. 5, the shaded blocks are especially interesting because they correspond to logic circuits that are implemented in FPDs. In particular, the external agent encompasses the most complex control functions which had to be implemented in FPDs for the processor board, as will be described shortly.

All of the components on the processor board were interconnected using Concept. Except for the sections to be realized in FPDs, it was straightforward to obtain schematic symbols for each chip, either by locating them in Cadence-supplied libraries, or by creating them with a convenient utility called *RapidPart*. Much difficulty was encountered, however, in integrating FPDs into the schematic, where an automated approach was needed due to the large number of I/O pins and variable pin-assignments. Recall from Fig. 5 that there are four main sections of the board that are targeted to FPDs; the process followed to integrate these sections into the Concept schematic will be illustrated by using the external agent as an example.

4.1 Implementation of the External Agent in FPDs

Our first choice for implementing sub-designs in FPDs was to remain within the LWB environment as much as possible. Cadence's recommended technique is to use a tool called *PIC Designer*, which is a framework within LWB that allows users to target (sub)-designs to FPDs. Users may perform design entry using schematic capture, Verilog, or VHDL, and PIC Designer either maps it directly into FPDs for small devices such as PALs, or exports more complex designs to a vendor-specific CAD system for FPGAs or CPLDs. Our experiences with PIC Designer were unsatisfactory. To implement designs in Altera FPD's, we needed to interface PIC Designer with MAX+plusII, but we were unable to do so. In this respect, we could not obtain adequate technical support from either Cadence or Altera.

Since PIC Designer was a failure for us, we attempted a more basic approach that would still allow us to remain within LWB. The idea was to use Concept for design entry, and then transfer designs from Concept to MAX+plusII via EDIF. We were not able to make this interface work either, because the vendor-supplied interfaces were several software releases behind (and incompatible with) the releases we were using. Our conclusion was to abandon the publicized "seamless integration" between the tools.

Following the difficulty in using LWB with MAX+plusII, we opted instead to perform design entry for the FPDs directly in MAX+plusII. We needed to develop a custom procedure for using MAX+plusII for FPD design, and then transferring the results back to LWB for board-level simulations and PCB layout.

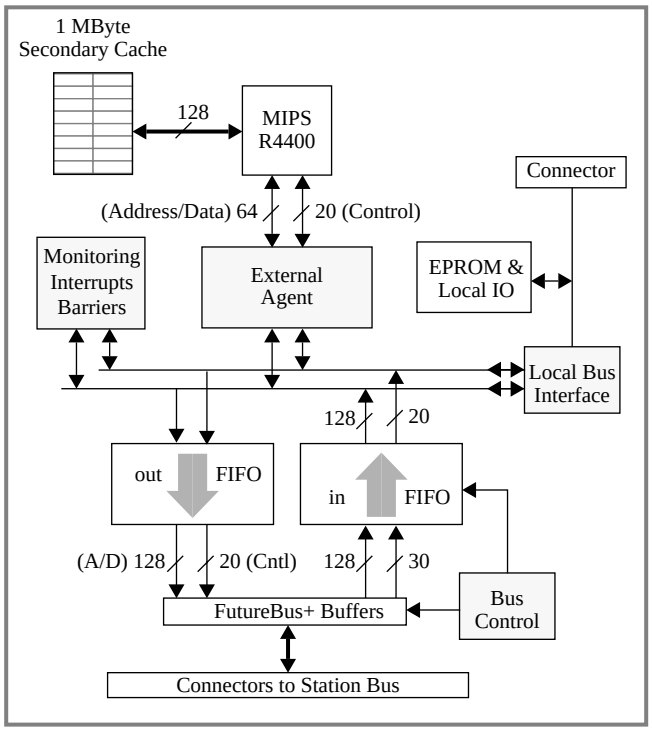

Figure 5 - *Block Diagram of a Processor Board.*

The FPD designs were done using AHDL, which proved to be the most efficient method for both small and large designs. We did not opt to use the other supported high-level languages, VHDL and Verilog, because the state-of-the-art in compiling designs written in such languages into FPDs yields results that are significantly less efficient (by 30%, as an estimate) than designs specified in a simpler language like AHDL, or created with a schematic.

Based on the number of I/Os connecting the external agent to other components (see Fig. 5), and on the speed requirements, the design was partitioned into multiple devices. Automatic partitioning in MAX+plusII was attempted, but this was abandoned because it was more appropriate to manually assign the board-level signals to individual FPDs. Since Altera offers many different FPDs, with differing speed-grades, package types, cost, etc., it was not obvious which ones should be used. To make the final selection, it was necessary to perform design entry and allow MAX+plusII to map the circuit into various chips, then assess the resulting speed-performance through simulation. We found that the entire circuit would fit into a single FLEX 8000A FPGA, but speed-performance was insufficient for our purposes (a 50 MHz clock rate is needed). On the other hand, using MAX 7000E CPLDs would achieve the desired speed, but would require multiple devices. We ultimately chose to implement the external agent in six CPLDs, a MAX 7256E and five MAX 7096E's, using the fastest speed-grade available for each chip. (The choice of whether FPGAs or CPLDs were more appropriate varied for the three FPD-targeted sections shown in Fig. 5.)

The process of obtaining the speeds that we required was not a simple matter of automatically mapping the design into the selected CPLDs. For many parts of the external agent, such as FSMs, we needed to carefully tune the design. The state assignments were obtained by studying the structure of the machines toward the goal of minimizing the amount of logic needed [8]. Also, extensive re-timing of sequential circuits was done. This is an example of how a demanding set of design requirements can tax the capabilities of today's CAD algorithms.

5 Experiences in Logic Simulation

In a large project such as NUMAchine, it is crucial to perform detailed simulations at both the chip-(FPD) level and board-level. In terms of complexity, ensuring that a single FPD will function correctly is a relatively minor task when compared to verification of a full board comprising numerous FPDs and other components such as buffers, FIFOs and memory. The key issue is that off-the-shelf components often have behavioral quirks that are not readily apparent from data sheets, and timing violations are extremely hard to detect and fix after hardware has been fabricated. To avoid long hardware debugging times, extensive effort was put into board-level simulation[1].

5.1 Simulation of Logic in Individual FPDs

We considered three possibilities for simulation and debugging of logic mapped by MAX+plusII into CPLDs and FPGAs: (1) using the simulation tools included in MAX+plusII, (2) using the stand-alone Verilog simulator provided by Cadence, and (3) using Smart Models.

1. As an example of simulation effectiveness, our fabricated memory modules (8-layer PCBs) operated correctly with only ONE jerry-rigged wire and a few iterations of re-programming of the logic in FPDs

Choice (3) was not successful for reasons discussed in Section 5.3, but both (1) and (2) proved to be good choices. Simulation with MAX+plusII is more convenient because it provides better access to all signals in a design, and its method of specifying test vectors is more efficient. In addition, state machine flip-flop values are displayed symbolically with meaningful names rather than as binary bit patterns. As a result, MAX+plusII was used heavily for simulation of both individual and multiple FPDs. Use of the Cadence stand-alone Verilog simulator is also straightforward because MAX+plusII can generate a Verilog file comprising a module that encapsulates all timing details of a design after it has been mapped into an FPD. It is then necessary to create a "testbed" file which specifies test vectors, and link this file with the Verilog output from MAX+plusII to perform the simulation. The advantage of this scheme is that it supports simulation of Altera FPDs with other types of chips. A drawback of the Cadence simulation environment is that it cannot display state machine signals symbolically, and it is more difficult to observe internal FPD signals.

A shortcoming regarding the simulation available with Altera's FPDs is that it is not possible to simulate the power-up loading of programming-bits into SRAM-based FPGAs. This caused considerable delays during our hardware debugging because we did not properly design the FPGA programming circuitry.

5.2 Board-level Simulation

LWB provides a powerful back-plane simulator, called OpenSim, that allows simulation of systems in which individual components can be represented in different ways. For example, it is possible to mix Verilog, VHDL, Smart Models, and other formats in a single simulation. The strategy that we ultimately used was based mostly on Verilog, but our first attempt was to rely heavily on Smart Models, hence our experiences with that tool will be described first.

5.3 Experience with Smart Models

Smart Models for FPDs can be configured for simulation by loading a mapped design (via a JEDEC file) into the model. Unfortunately, we found the library did not include models for the state-of-the-art FPDs we selected for our design. Furthermore, other commodity chips, such as the FIFOs and the Futurebus+ buffers in our design, were also not available in the Smart Models library. The library did contain models for similar chips, but none sufficiently close to the ones being used.

Besides simple buffers and gates, the only device for which we ultimately used the Smart Models library was the MIPS R4000 processor, and that was applied only in a limited capacity. The model was specified for 50 MHz, whereas our design required 200 MHz, but it proved useful for learning about details which were not clearly documented, such as exactly how the processor behaves when it is reset.

Since many of the chips used in our design were not available in the Smart Models library, we opted to write custom Verilog models to represent many of the chips on the board, including the MIPS R4400, FIFOs, and Futurebus+ buffers.

5.4 Experiences with Verilog for Simulation

Our installation of LWB includes two configurations of Cadence's Verilog compiler: a stand-alone version, and a version integrated into OpenSim. Use of the stand-alone version for simulation of individual FPDs was straightforward, as discussed in Section 5.1. However, to incorporate the Verilog code generated by MAX+plusII into the OpenSim board-level simulation, we had to develop a custom procedure.

5.4.1 Importing FPD's into LWB for Board-level Simulation

To complete the Concept schematic, we needed to generate schematic symbols for the sub-designs implemented in FPDs. The symbols must encapsulate the timing characteristics of the FPDs, and must also specify the physical properties of the chips, for PCB layout. The procedure that we developed for this purpose is illustrated in Fig. 6. After mapping of each sub-design the Verilog file generated by MAX+plusII was imported into Concept. This was done using a convenient command, called *GenView*, which can read a Verilog file and automatically create a schematic symbol. Using these symbols, a simulation model of the logic mapped into each FPD was connected to the other system components. The *automatic* generation of schematic symbols was crucial because each FPD involved more than a hundred IO pins, implying that any manual procedure would be prone to errors. To facilitate PCB layout, each symbol in Concept has an associated physical description that specifies package type, pin assignments, etc. To generate this information for each FPD, we created a custom-built tool for parsing MAX+plusII files, called *Assignment Control Files* (ACF)[1], and generating the equivalent information in Cadence format, which is called *chips_prt*. In Fig. 6, the graphics symbols in the Concept schematic are shown with two views: one is the logical symbol (Verilog) used for simulation, and the other, normally not visible in the schematic, is the physical symbol for specification of IC-package details. Our utility for exporting FPD pin assignments from MAX+plusII to LWB can be obtained at *ftp://www.eecg.toronto.edu/pub/software/acf2chip*. Note that no such utility is available from either Cadence or Altera.

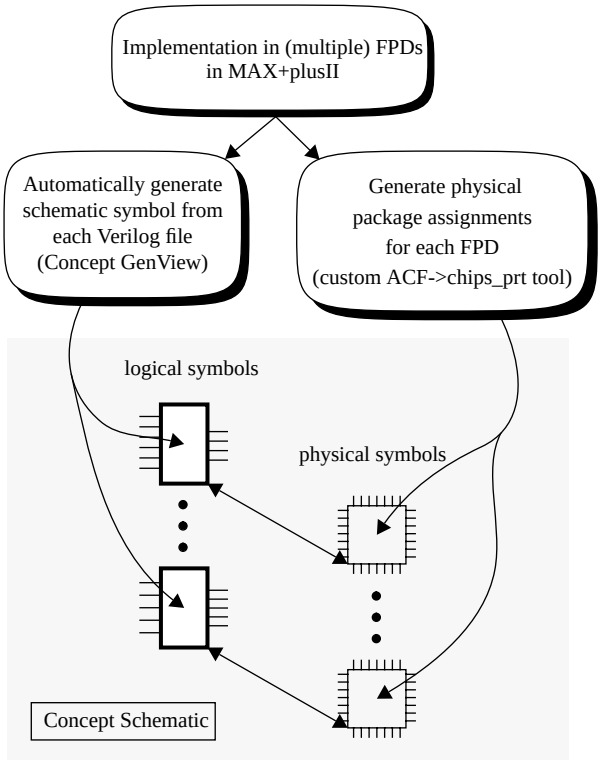

Figure 6 - *CAD Steps for Integrating MAX+plusII and LWB.*

1. The task of writing this converter was complicated by Altera changing the format of the ACF files in new releases of MAX+plusII. Also, when bugs were found in a new release it was difficult to retreat to the previous version because the new ACF format was not readable by the older release!

After connecting the FPD symbols to the rest of the board, OpenSim was used to provide test vectors and verify board-level operation. Although OpenSim is powerful and flexible, we found that the graphical interface provided for entry of test vectors is awkward. Fortunately, however, LWB on-line documentation also describes a powerful scripting language for specifying test vectors. We invested considerable time to learn this language, but consider this to be time well-spent.

6 Layout of the Printed Circuit Boards

PCB layout was accomplished with Cadence Allegro. The main difficulty here was in exporting pin assignments from MAX+plusII into LWB, and the solution to this problem was described in the previous section. It was easiest to hand-place all chips, and then use the automatic routing algorithms offered in Allegro; routing of the processor board took several days (on a SPARCstation 10), after which the approximately 5% of nets that could not be autorouted were completed by hand. The final board has 12 layers in total, alternating power/ground and signal layers, which greatly reduces noise and cross-talk in our 50 MHz system. A final comment on PCB creation is that we tried to perform noise analysis on the PCB using the tools included in LWB. Unfortunately, these tools require detailed I/O buffer characteristics, called *IBIS* models, for the pins on all chips, and after contacting the manufacturers we were unable to obtain such models for most of our devices.

7 Final Remarks

This paper has related our experiences with design of a high-performance computer system using advanced CAD tools, targeting all control circuitry to FPGAs and CPLDs. Comments have been provided on the usefulness of several CAD products for our purposes, including Cadence Logic Workbench, Altera MAX+plusII, and Smart Models. Besides the difficulty of achieving the required 50 MHz system clock rate in FPGAs and CPLDs, the main stumbling blocks were the lack of availability of simulation models for recently introduced chips, and the problems that we encountered with the integration of Cadence LWB and Altera MAX+plusII.

Our preference for the CAD flow was "top-down" for design of each PCB. However, our experience is that, although published marketing suggests otherwise, the tools are not sufficiently integrated to support this. We feel that CAD providers should focus on their strengths and encourage designers to use the properly supported flow: in our case, "bottom-up" design with MAX+plusII for FPD-targeted circuits, and LWB for board-level integration.

8 References

[1] Cadence Logic Workbench, release 9404, Cadence Design Systems Inc., 75 West Plumeria Drive, San Jose, CA 95134.

[2] Synopsys Smart Models, Synopsys, 700 East Middlefield Rd., Mountain View, CA 94043

[3] 1995 Data Book, Altera Corp, 2610 Orchard Parkway, San Jose, CA 95134

[4] Convex Exemplar Systems Overview, Convex Computer Corporation (Hewlett Packard), 3000 Waterview Parkway, Richardson Texas 75080.

[5] Kendall Square Research. KSR1 Technical Summary, 1992

[6] Z. Vranesic, et. al., "NUMAchine Technical Report," available on the WWW at *http://www.eecg.toronto.edu/~brown*

[7] J. E. Veenstra. Mint Tutorial and User Manual. Technical Report 452, Computer Science Department, University of Rochester, May 1993.

[8] Zeljko Zilic, Guy Lemieux, Kelvin Loveless, Stephen Brown, and Zvonko Vranesic, "Designing for High Speed-Performance in CPLDs and FPGAs," *3rd Canadian Workshop on Field-Programmable Devices (FPD'95)*, May 1995, pp. 108-113.