Hardware Emulation for Functional Verification of K5

Gopi Ganapathy, Ram Narayan, Glenn Jorden, Denzil Fernandez
Advanced Micro Devices
5900 E. Ben White Blvd.
Mail Stop 615
Austin, TX 78741

Ming Wang, Jim Nishimura
Quickturn Design Systems
440 Clyde Avenue
Mountain View, CA 94043-2232

The K5™ microprocessor is a 4 Million transistor, superscalar, X86 microprocessor. The K5™ microprocessor is an AMD original design, verifying compatibility with the existing X86 architecture and software is crucial to its success in the market place. The X86 architecture has been constantly evolving over several years without any published specification. The primary mechanism for functional design verification of an X86 processor is simulation. The ability to execute a good sample set of the X86 software base on a model of the processor architecture before tapeout, is key to achieving very high confidence first silicon. The Quickturn Hardware Emulation system allows us to map a model of the design onto hardware resources and execute it at high speeds. In this paper we present the emulation methodology that was jointly developed for K5 and applied successfully to meet our functional verification goals.

I. INTRODUCTION

The fastest hardware accelerators commercially available today, can simulate a full gate level model of the K5™ microprocessor in the low 100Hz range. At this simulation speed, booting an operating system like Microsoft® Windows®, and then starting up an application like Word or Excel is practically impossible. [1] and [2] have described the benefits and the value of a hardware emulation-based verification methodology.

Hardware Emulators, map a gate level model of the design onto Field Programmable Gate Arrays (FPGA) on the emulation system. This model can be executed in the high 100KHz range, giving us three to six orders of magnitude improvement in model execution performance. This speed combined with the very high instructions execution rate per clock cycle (IPC) of the four issue, superscalar K5™ core, allows us to interface the emulator directly onto real hardware like a simple PC and execute X86 applications in minutes, instead of days and months on a simulator.

II. EMULATION HARDWARE

The K5™ emulation system used two generations of emulation hardware from Quickturn Design Systems. The initial system was the MARS™ IP2500 (Fig. 1). This system was comprised of five IP500 chassis interconnected together with approximately 400 cables. Each IP500 chassis has 5 logic boards (LBM) with a rated capacity of 100K gates each, and a total system capacity of upto 2.5 million gates. However, the actual gate densities vary depending on the topology and regularity of the logic structures being mapped onto a board. LBM-LBM interconnections are all unidirectional, three unidirectional pins are required to model every bi-directional bit. A Configurable Memory Module (CMM) is used for modeling array structures. The target system is a generic the X86 PC system running in the MHz range with a custom interface board to buffer transactions to the slower emulator side (Fig. 2).

SATURN™, the current generation emulation system from Quickturn, utilizes higher capacity FPGAs and better interconnect technology. The entire K5 design now in a single Saturn™ chassis with 11 logic boards.

III. DESIGN PARTITIONING

The K5™ microprocessor has a four issue superscalar processor core, with two ALUs, one Branch Unit, one Load Store Unit, a Floating Point Adder, Floating Point Multiplier and Floating Point Shifter. There are 18 shared, 42 bit bi-directional buses that connect between all of the above functional units. All these buses feed into the multi-port register file and reorder buffer. Additionally, there are wide buses from the Instruction and Data Caches feeding the instruction decode block and the load store sections. Partitioning the K5 into 25 LBMs in such a way as not to overflow LBM and FPGA pin limits, proved to be a major challenge.

Special modeling and identification of all tristate signals is also necessary. The multiport register-file and reorder buffer had to be modeled along write port boundaries with duplication of storage bits. Tristateable wired-OR structures had to be converted to n-input static OR gates with tristate buffers on the OR gate output to reduce the number of signals traversing LBMs. The initial bringup time of the K5 hardware emulation system was dominated by this partitioning and mapping iterations to meet emulator constraints. The SATURN™ emulation system with its higher logic and pin capacity reduced the partitioning problem to a large extent.

33rd Design Automation Conference®

DAC 96 - 06/96 Las Vegas, NV, USA
©1996 ACM 0-89791-779-0/96/0006..$3.50

IV. DESIGN MAPPING

Both generations of Quickturn Emulation systems use the Xilinx™ FPGA as basic building blocks for mapping logic. The entire microprocessor has to be mapped into a set of basic primitives like flip-flops, muxes and four input combinational functions that are implemented in a Xilinx FPGA. For example, there is no direct support for the thousands of complex, switch level circuits implemented in a full custom processor like the K5™ microprocessor. So a gate level emulation model that is functionally equivalent to the real transistor level implementation of the design has to be generated.

Quickturn software uses memory compilers that map memory arrays to FPGAs directly. This requires that internal memory arrays be represented as black boxes with read and write enables, address and data buses. Simplified versions of the IOPADS, Clock Generators, Bus drivers and receivers are built at a gate level for emulation. All delay dependent logic, like self timed structures, are removed from the emulation model [3] and replaced with functional equivalents. External timing signals are used.

Maintaining consistency between this additional emulation view and the "real" representation of the design was expensive and required dedicated emulation model build resources. The initial K5™ emulation model was verified by simulating the K5™ test base on a hardware accelerator. Gate-Switch equivalence checkers were later used to maintain this model.

Thousands of FPGAs are involved in mapping a K5™ class design on the emulator. The partitioning and placement of a cone of combinational logic, onto FPGAs that are physically far apart, and routing signals between these FPGAs using long cables makes it necessary to handle very large delays and signal skews on the emulation system.

K5 is a latch based design using 2 Phase, non-overlapping clocks. There is a significant emphasis during the development phase to make the timing analysis of the K5 emulation model as simple as possible. A complete list of all gated clock locations is provided in advance to make sure that that setup violations on the conditional signal to a gated clock can be handled. By varying the non-overlap between the phases and the duty cycle of the phases, hold time and critical path issues were identified and resolved.

Mapping the emulation view of K5 onto the Quickturn hardware involves four major steps. The first step is Model Build, where an emulation view of the design is generation. The second step is Partitioning, where the design is partitioned into LBMs in such a way as to meet the logic and pin constraints of each LBM. This is followed by Compilation, where the logic partition assigned to each LBM is flattened, checked for connectivity and timing consistency and further partitioned into smaller logic clusters, each of which can fit into a single Xilinx FPGA. This involves steps like placement and routing within and between FPGAs. The final step is Linking, where all of the LBMs are linked and the final interconnections are completed. At this point the design is ready to be downloaded onto the emulator and tested.

The process of starting from a netlist and generating the emulator program files is very compute and disk intensive. The large number of LBMs and the thousands of FPGAs involved allows us to parallalize this task. We used several workstations to compile LBMs in parallel. The FPGA place and route step used hundreds of networked workstations to generate the required program files for the FPGAs in the emulation system..

V. EMULATOR OPERATION MODES

The K5™ simulation environment models a simple X86 PC system with external memory(EXTMEM), memory controller(EXTCTL), and event generators and monitors that mimic the bus master and other devices on a PC. Test suites are developed in X86 assembly language and assembled, linked and loaded in EXTMEM. EXTMEM is mapped onto the CMM, and can be read and written without disturbing the rest of the design. This mode is referred to as the *Simulation* (SIM) mode.

The SIM mode of operation on the emulation hardware allows us to use the emulator as a very fast simulator. An external monitor board is used to drive the clocks, reset the system and display the pass or fail status of a test on a custom built monitor board. This mode is very useful during the initial validation phase to verify the functionality of the emulation model through full regression of the K5™ test suite. The DebugWare, a combination tester and logic analyzer, that is part of the emulation system is used to probe key signals and "readback" most internal nets for debugging failures. The readback operation utilizes a scan architecture [4] to retrieve the internal state of the desired net at specified times.

In the *In-Circuit Emulation* (ICE) mode the pins of the K5™ microprocessor are brought out to the X86 PC target system. This is the in-circuit mode used to execute key X86 applications. An in-circuit buffering mechanism (Fig. 3.) is implemented to facilitate switching between ICE and SIM modes without disturbing the design loaded onto the emulator. This reduces the need to maintain 2 different emulation views of the design, one for each mode of operation.

Incremental changes to the design can be made at differently levels. For small changes localized to a single LBM, the partitioned database for that specific LBM is carefully edited, and the affected FPGAs are re partitioned and compiled. This is accomplished in a few minutes. For more complex changes affecting multiple LBMs, the LBM database needs to be edited and re-partitioned. This level of incremental changes takes from a few hours to a day, depending on the number of LBMs affected.

VI. K5 HARDWARE EMULATION CYCLE

During the course of the K5 project the hardware emulation has been called upon to perform a

variety of tasks. The K5 hardware emulation project cycle could be broken down to key phases:

Methodology Phase: This phase involves the plan, design and initial experiments that results in the hardware emulation methodology. Compilation and partitioning of the entire K5 design was achieved by working with smaller blocks, and getting a better estimate for the optimal partition of the design. The close interaction of the Quickturn R&D team and AMD engineers was critical to the success of this phase. Once the first K5 database was successfully compiled, the methodology and the design is initially verified with very small test cases, running for a few hundred clock cycles. A number of hardware and software issues were resolved before these small test cases were successfully executed [5].

Validation Phase: The goal of this phase is to verify the K5™ processor in emulation with the validation suites. This phase is a confidence builder in the emulation methodology and system. Experiments are performed to arrive at the optimal parameters for operating the emulator. The hardware is operated in primarily the SIM mode. In parallel, the emulation interface board and target system is manufactured and tested at the emulation frequencies.

Systems Verification Phase: This is the most critical phase of the project cycle where we execute X86 applications. Initially simple diagnostic tests that verify the hardware interface are run to test the ICE mode operation. These diagnostics are run from the ROM on the motherboard. The next step is to run billions of cycles of real X86 application software to verify key compatibility and functionality issues. This level of verification and testing can be achieved before first silicon solely due to the execution speed of hardware emulation. In this phase the hardware emulator is the primary failure detection mechanism, ably supported by extensive random test generation in simulation.

Silicon Test Support Phase: In this phase the primary function of the hardware emulator is to reproduce and isolate the failures found using silicon validation boards. The emulator is more of a failure isolation and fix verification platform. Incremental changes made in the design are verified on emulation before committing these fixes to the design database.

Systems Regression Phase: The primary task of hardware emulation in this phase is systems regression for verifying every revision of the design prior to tapeout. The regression/debug process starts with executing a number of 16 bit and 32 bit X86 applications. A harsh environment is created with other bus masters and event generators plugged into the target system for stressing the design.

VII. DISCUSSION

The design and development costs of successful emulation are significant. Large emulators are expensive, and thus limited resources. There is a high startup cost in full custom designs, due to the modeling and partitioning constraints enforced. A dedicated team of engineers is often necessary to effectively compile and debug a design on the emulator. When compared to simulators, the time-to-debug, measured as the time it takes to be productively debugging on the emulator from an initial netlist is measured in hours, instead of minutes. Debug productivity is also affected due to the cost of adding probes to observe additional signals and iterating in this process to isolate failures.

However, compared to debugging silicon the emulator provides a friendlier and more productive environment for isolating failures and verifying bug fixes. It is possible to try out different options for tuning architectural performance or bug fixes before deciding on an optimal fix. The emulation compilation time is insignificant compared to silicon fabrication and system development time is very attractive.

Speed and the ability to directly interface with real hardware are the two major differentiating factors between emulators and traditional simulators. The very high emulation speeds give us the ability to debug X86 applications on a model of the design well before first silicon with a simulator like debugging environment. Successfully executing key operating systems and applications before tapeout is a huge confidence builder and positions the design for silicon validation on hundreds of PC systems as soon as first silicon is available. This gives us the ability to quickly go after those bugs that are difficult to find. If we relied solely on directed tests and random test generators, it is possible to miss that one bug which prevents us from running key applications on first silicon, thus delaying silicon validation and hence final production.

The K5™ emulation system was used to debug and verify several key 16 bit and 32 bit X86 applications. Several bugs were identified and fixed in this process. Emulation was instrumental in achieving very functional first silicon on the K5™ microprocessor.

VIII. CONCLUSIONS

Hardware emulation of a design the size and complexity of K5™ microprocessor has proved challenging. A large number of operating systems including Windows® 3.1, Windows® NT, Windows® 95, OS/2®, Linux® and their application have been demonstrated on the emulator. Despite a rigorous directed test generation flow and an extensive random test generator flow a few key bugs have been found and fixed using the emulator. The K5™ hardware emulation system has been extremely valuable in getting the first versions of K5™ microprocessor into production.

In the future with higher emulation speeds, we expect the emulator to be a very useful tool for tuning architectural performance against a wide range of applications and benchmarks.

IX. ACKNOWLEDGMENTS

The success of the K5 hardware emulation is attributed to the contributions of Elango Rajagopal, Tom Maciukenas, Rama Gopal, David Sone, Ravi Subramanian, Murali Chinnakonda, Jong-Won Yuk and the authors. The authors acknowledge the support

extended by the Quickturn R&D and Expert Users groups.

REFERENCES

[1] James Gateley et. al., "Ultra Sparc-I Emulation." 32nd Design Automation Conference Proceedings, 1995.

[2] Jainendra Kumar et. al., "Emulation Verification of the Motorola 68060". Proceedings of the 1995 International Conference on Computer Design, 1995.

[3] Gopi Ganapathy, "Hardware emulation methodology on Krypton." *AMD Internal Report, 1993.*

[4] E. B. Eichelberger & T. W. Williams, "A Logic Design Structure for LSI Testability", Journal of Design Automation and Fault-Tolerant Computing., Vol. 2, pp. 165-178, May 1978.

[5] Ram Narayan, "Performance monitoring and verification methodology in RTL simulation and hardware emulation", *AMD Internal Report, September 19,1995.*

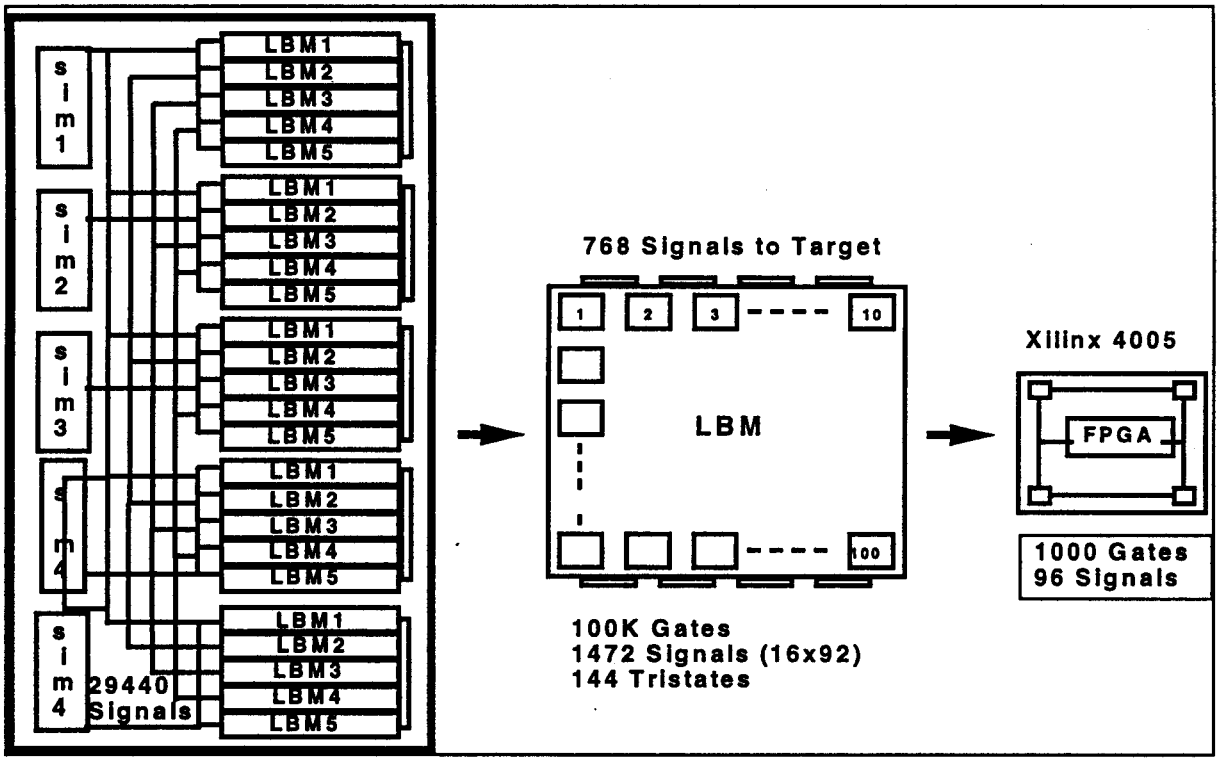

Fig. 1. MARSIII Emulator Building Blocks

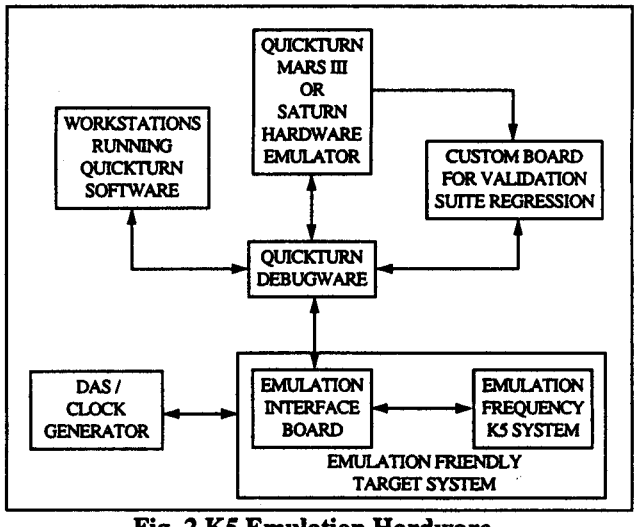

Fig. 2 K5 Emulation Hardware

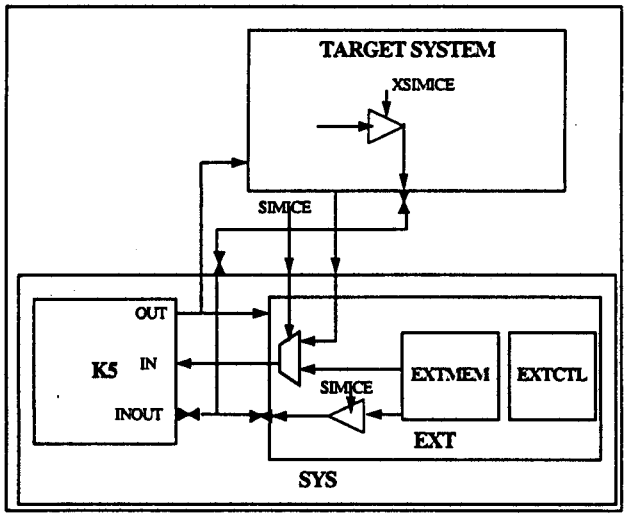

Fig. 3. Buffering Scheme for In-Circuit Emulation