IBM eServer z900 system microcode verification by simulation: The virtual power-on process

by S. Koerner
M. Kuenzel
E. C. McCain

In the development of a large, complex computer system, the verification of its microcode by simulation can significantly decrease the time required for the integration, "bring up," and testing of the system. However, creating a process that integrates and aligns the smaller verification tasks to form a coordinated, seamless, and comprehensive system verification plan requires considerable effort. In this paper we present a brief summary of previous verification processes and describe a process, virtual power-on (VPO), which encompasses both hardware and software verification. We then compare the results achieved with that process with those achieved using previous processes. The VPO process was initially applied to the IBM eServer z900, resulting in a significant reduction in the time required for its development.

Introduction

System microcode is licensed software that has direct access to hardware facilities for control purposes. For decades, system microcode verification has been an appreciable challenge, and it continues to be so. The IBM eServer z900 design represented the most extensive change in server design since the Enterprise System/9000* (ES/9000*). Consequently, use of the "business as usual" approach would have required considerable "bring up" (or, simply, bring-up) time. That approach involves unit-testing the system microcode, then delivering it along with new hardware to bring up the system. System microcode is then executed on the hardware for the first time at "power-on" (the initial turning on of a large computer system). As a result, microcode bring-up and debugging severely hamper hardware verification.

Hence, the approach to system microcode verification required a radical change. The challenge was to verify the system microcode of a complex computer system by means of simulation so that the time and the amount of early hardware needed for the integration, bring-up, and testing of the system could be reduced, thus having a significant impact on development cost and time-to-market.

The VPO process made this possible. It evolved by re-engineering the system integration process to execute tasks that would normally be executed during system bring-up and executing them in a simulated environment. Code and hardware models had to be delivered early, and powerful verification engines had to be developed. These

0018-8646/02/$5.00 © 2002 IBM

verification engines enabled this method to execute many simulation cycles in a hardware and software co-verification environment, thus accelerating the problem-discovery rate.

To achieve virtual and actual power-on (respectively, the initial turning on of a large computer system in simulation or physically), the microcode required to control the system must be present and at least verified on a unit level. By generating a system "driver" (the term *driver* is used to describe a package of all of the run-control software needed to power on and initialize the system to be ready to run the user's control programs and applications) prior to power-on and using a full system model that is functionally equivalent to the real hardware, VPO can be initiated.

In this paper, we first present a brief summary of previous work on system microcode verification, and then describe the VPO process and its application to the development of the IBM eServer z900.

Evolution of system microcode verification

This section contains a summary of previous work on the use of simulation for system microcode verification, which, although brief, should indicate the essence of a comprehensive system microcode verification approach.

IBM ES/9000 Models 820 and 900

The simulation approach used to verify the microcode of the IBM ES/9000 Models 820 and 900 contained several components which can now be identified as VPO components. The components integrate a fully functional system model which contains the individual "pre-verified" unit models of all of the computer chips that make up the central electronic complex (CEC), or "system," with an external controller containing all of the unit-tested code necessary to power on and initialize the hardware for customer use. The verification engine used to drive this was a special-purpose parallel processor, designated as an engineering verification engine (EVE) [1]. The simulation approach used was to verify millions of lines of system control code, designated as processor controller Licensed Internal Code (PCLIC) by attaching an IBM 3092 processor controller element (PCE) to the EVE (containing the system hardware model) [2]. Attachment was achieved by means of a PCE-to-EVE adapter card. The PCLIC had to drive the system model and verify the resets (setting of initial states), recovery, manual operations, self-tests (hardware diagnostic), and finally power-on reset (POR). The latter is analogous to initial microcode load (IML), which executes in multiple steps to initialize the hardware and code used in the zSeries 900.

Although hardware and code problems were found, the approach used was the groundwork for using an initial program load (IPL), an architected "mini-boot" which allows an operating system load from external devices into memory—a key milestone in mainframe bring-up of an operating system such as System Assurance Kernel (SAK), an IBM internal-use-only tool for architecture verification. Thus, all of the initial bring-up operations could be modeled on the "test floor" during the powering on of the hardware.

Removing these errors ahead of time considerably reduced the time from power-on to SAK IPL. From a process point of view, this integration of code and hardware required tight system driver control. As a result of *all* of the simulation done for the ES/9000 Models 820 and 900, SAK was running 31 days after power-on. Thus, the above verification strategy could be regarded as an "ancestor" of the VPO process.

Strategic technology shift

In the early 1990s IBM decided to take a different direction in server technology. This caused a dramatic change in teams and personnel. The server development group became global and joined forces with the Boeblingen, Germany, and Endicott, New York, development groups to develop CMOS chip-based systems. This forced a rethinking of the verification strategies and processes—to an emphasis on establishing smaller subsystem verification environments to support system development.

IBM S/390 Enterprise Server Generations 3, 4, and 5*

System microcode simulation for the IBM S/390 Server Generations 3, 4, and 5 (or simply G3, G4, and G5) included some portions of the system initialization code that resided on the service element (SE, analogous to the PCE on ES/9000). The results of the G4 verifications were ten hardware problems and 35 code problems found in simulation. Power-on-to-SAK IPL required approximately 8 weeks on the G4 and 9½ weeks on the follow-on G5. The code needed to verify the initial microcode load (IML) components was isolated and could be extracted to run in the SE attached to the hardware system model located in the EVE 1.5 hardware accelerator [3].

However, because of the different focus of the code verification team and the hardware verification team, there was no comprehensive system-driver integration plan to line up with this verification schedule.

IBM S/390 Enterprise Server Generation 6

The system microcode simulation approach was essentially unchanged for the next generation of CMOS systems. However, there was a major change in tools. The SE attachment testing for IML component verification [3] was mainly executed with a ZFS (IBM internal software

simulator) environment running on VM attached via TCP/IP. In addition, a design verification system [4] attached to an SE via TCP/IP was being developed under a partnership with Quickturn.[1] The system, designated as the Concurrent Broadcast Array Logic Technology (CoBALT™)[2] system, was designed to provide many more simulation cycles to the IML verification effort. The intent of the CoBALT environment (system and associated software tools) was to take the SE-code driver as delivered for a real server system power-on and initiate an IML with the model of the server system running in the CoBALT system. In theory, every problem found and fixed means one less problem when the real hardware is powered on. The goal for the CoBALT environment during the G6 verification phase was to execute IML step 3, which included SIF (L2 serial interface), load of bootstrap code, and execution of bootstrap code before real power-on. Although we did not reach that goal, we found approximately 20 code problems, and the power-on-to-SAK IPL was two days. (The amount of change between the S/390 G5 and G6 was relatively smaller; that explains why bring-up was executed much more rapidly.)

On the basis of the experience gained in this effort, the decision was made to perfect this environment and develop a process to coordinate and bring together all of the development code and hardware teams to deliver *all* of the code needed to power on a system months prior to its real power-on date, and to execute a virtual bring-up: the VPO concept. We have learned from the experiences with previous microcode verification strategies and refined this "end-to-end" verification strategy.

A comparison of the time required to complete SAK IPL for the various systems described is shown in **Figure 1**. The figure demonstrates the positive impact of the new microcode verification concept by reducing system integration time despite an increase in system complexity.

We next describe what was done for the development of the verification process for the IBM eServer z900.

Microcode development process and verification

In applying the VPO approach to the IBM eServer z900, the objective was to achieve long-term improvement in the quality of the microcode delivered to system integration and bring-up, by detecting and facilitating the solution of the vast majority of code problems before the system is first switched on in simulation and emulation.

The integration phase of a new server system is strongly dependent on the quality of the system microcode. During the first major simulation phase for the z900 server before first power-on, more than 250 documented microcode problems and three hardware problems were identified in simulation and, even more significantly, solved. During the entire simulation phase, more than 500 problems were found and fixed. The following description covers the entire simulation process. All of the components are in regular use, and they all meet or exceed the expectations defined for them. Two other papers in this issue go into more detail on the components [4, 5].

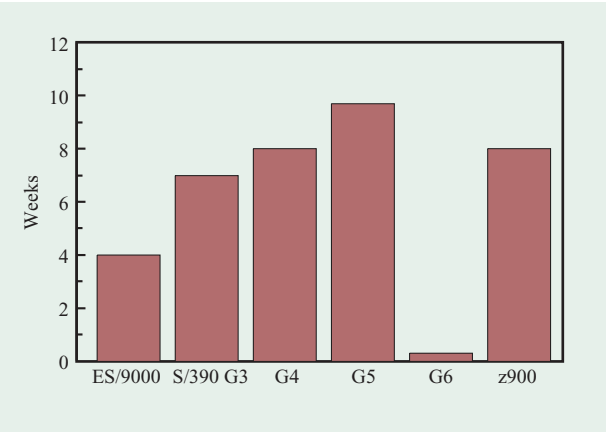

Figure 1

Actual time from power-on to SAK IPL.

Several constraints must be addressed with the introduction of this new simulation concept. It was not only necessary to develop simulation and emulation environments with improved coverage, speed, and networking between the components, but also to consider the impact on the entire microcode development process. Previously, the first microcode driver was available at power-on time, leaving little time for simulation activities. Hence, the focus was on debugging the real system. Putting more emphasis on simulation means an improvement of several months in the availability dates of specific microcode functionality. Beyond that, the simulation process must deal with the fact that design and coding are still ongoing, since not all microcode can be ready at the start of simulation. Good regression capability is therefore a requirement and an integral part of the simulation process. Thus, in order to be successful, a very detailed simulation and component delivery plan must be established and executed. Using VPO, the first microcode driver for the z900 server system was delivered five months prior to real power-on (see **Figure 2**).

Virtual power-on simulation process for the eServer z900

The system microcode simulation process for the z900 server was aimed at detecting microcode problems as early

[1] The Quickturn™ Corporation, a Cadence® Company, San Jose, CA 95134.
[2] Now a product of Quickturn.

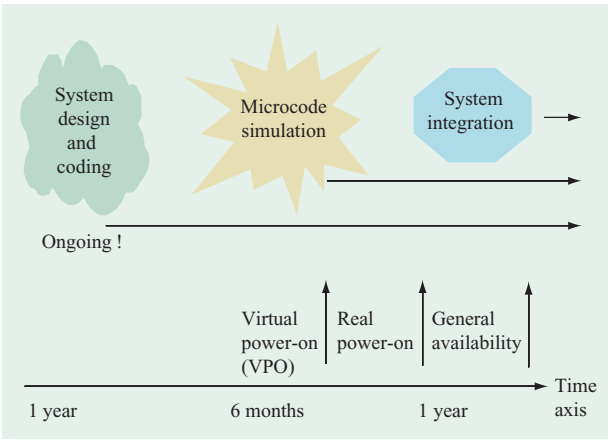

Figure 2

Microcode simulation in development process.

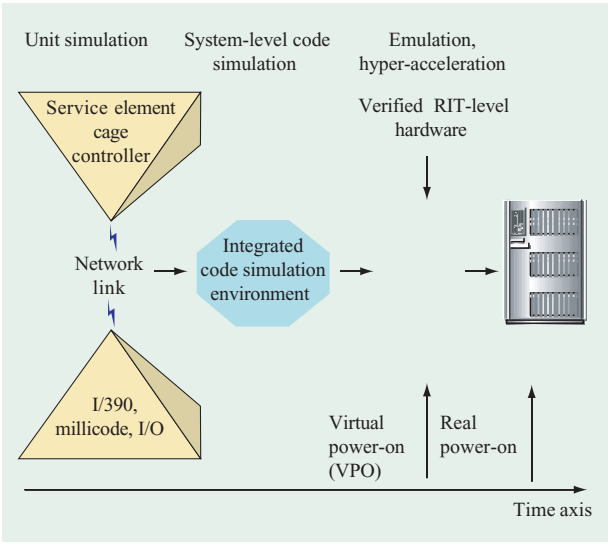

Figure 3

ESG microcode verification process (VPO concept).

as possible, starting with microcode components. In the first phase, the components were verified to the maximum extent possible in a standalone simulation. The components included the office mode of the service element code, the code emulator, and CECSIM [5]. After the first phase, the different unit test simulation environments were combined in order to detect interface problems.

The second phase of verification occurred exclusively in a code simulation environment without reference to any hardware model. Only that approach provided the necessary simulation speed and decoupling of code verification from the unstable and slow (even with emulation) hardware model in that early time frame [5].

In the third phase, the interaction of microcode and the processor hardware model was verified. The introduction of new hardware designs resulted in changes in the hardware access microcode layer, traditionally known to generate many errors. The third phase was carried out until the power-on date using the CoBALT system—with a successful first-time result, for a CMOS-based eServer system, of finding and removing a significant number of problems [4] (see **Figure 3**).

Targeted problem areas

On the basis of experience from past efforts, the following groups of microcode errors are targeted with the VPO process:

1. Software logic problems that can already be found in the unit-software simulation environments.
2. Interface problems among the different types of microcode.
3. Problems in getting the code to work together with the new hardware.

Microcode errors in categories 2 and 3 can be found only if the different simulation environments are linked together [5]. Those in category 3 can be found only with powerful emulation systems because of the model size and speed needed to drive this environment [4].

Although there was no overlap in the microcode errors found in the different simulation environments, it was obvious that some microcode components were missing, and therefore certain bugs were not revealed by the simulation process. Using the CoBALT system, more than sixty code and three hardware errors were detected; the errors were in the CEC initialization code (i.e., support element code and hardware reset files).

IML simulation using the CoBALT system for system initialization code verification ended at the point of initiating service word communication during bootstrap execution in IML Step 3. That step comprises six major functions, all of which were tested in VPO on the CoBALT system as well as subsystem simulation:

1. Run ABIST (array self-test) to initialize internal arrays.
2. Set all latches to their logic reset state.
3. Load bootstrap code.
4. Start clocks for the user chips.
5. Execute bootstrap code to finish hardware initialization in preparation for code execution.
6. Initialize service word communication.

Results

Overall, a total of 82 microcode bugs escaped until IPL completion. It was estimated that the entire simulation process saved more than 12 weeks in integration time. The microcode simulation coverage for the system was greater than 80% through IPL completion, and is expected to be greater than 95% for the next generation of zSeries servers.

Escape analysis

By analyzing the escapes, it was discovered that each simulation environment offered benefits. It was known that the emulation effort did not come as far as planned, impacting the distribution of the escapes and the length of the integration phase. It took 5½ weeks on the test floor to accomplish IML steps 2–5 and 3½ weeks to finish IML and then IPL the first operating system (SAK). The higher number of escapes at the beginning of IML testing was also induced by missing microcode in the simulation setup. Microcode that was verified successfully in the hyper-accelerated emulation environment (IML steps 2 and 3) executed very well after real power-on. The same applied for microcode that was verified in the CECSIM bringup vehicle [5] (see **Figure 4**).

Problem distribution

As expected, the problem distribution among the different system components showed that those that were totally new compared to those of previous systems tended to be more problematic, for example, as indicated in **Figure 5** for the component "Eng data/se," which contained new elements in the system control structure. Although new components tend to appear late in the development cycle, and only a few (if any) relevant simulations can be carried out before power-on, it is evident that they should receive closer attention in the future. Additionally, the simulation environments should be as close as possible to the real execution environment to minimize code changes for simulation-only purposes. Only the bare minimum of changes in time-out values and shortcuts because of missing interfaces should be allowed.

Network connection of the simulation environments

One key element of the simulation process was the ability to tie the different types of microcode together and connect them to the hardware and microcode simulation. Since the entire microcode of the z900 resides on the SE [6, 7], it plays a major role in the simulation strategy. It must connect to all simulators being used. The following describes how this was achieved.

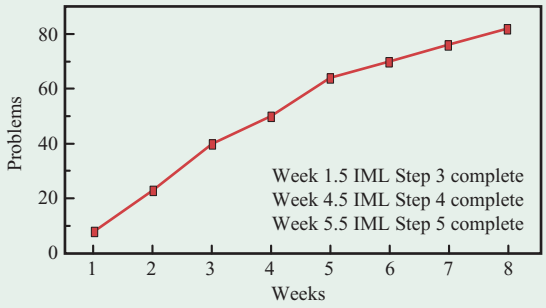

Figure 4

IML gates vs. time (SAK IPL complete after eight weeks).

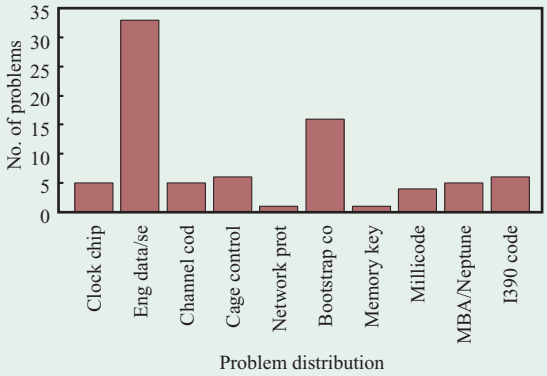

Figure 5

IML through SAK IPL: Number of problems found for various components.

Support element

The support element (SE) is the control and maintenance unit for eServer components. It is able to perform a wide variety of tasks; it controls the system configuration and checks the configuration for plausibility, initiates the loading of microcode into the PUs, and provides functionality to modify the hardware configuration or to replace defective parts while the system is operating. (For more detailed information on the SE, see [6, 7].)

Physically, the SE consists of an IBM ThinkPad* notebook PC that is mounted inside the frame of the host. It is connected to the host via one of the private Ethernet networks that exist within the cage controller structure [2], and it uses the cage controllers to obtain information on hardware, system status, and power status, and to

communicate with the microcode running on the zSeries PUs.

Since the scope of the cage controllers is limited to the cages in which they are physically housed, and the cage controllers do not have the ability to store persistent system information, the SE is the first element in the control hierarchy that has a complete system overview and can perform tasks requiring persistent configuration information.

SE standalone verification ("office mode")

The SE developers are faced with the situation that the basic functionality of the SE must be available prior to the existence of the hardware that the SE will have to control. Even after the first availability of prototype hardware and during the following bring-up and test phases, the amount of hardware available is far from sufficient to allow testing of all code changes on real hardware.

The standard solution for this problem is to create a version of the code that runs on a standalone personal computer. The functional code remains as unchanged as possible, and only the actual hardware access routines are replaced by "office" simulation access routines with artificial responses.

This method is implemented and widely used to test and debug SE applications. In fact, every code change is required to be tested in office mode prior to integration into the code libraries. Therefore, every developer must have access to an office-mode simulator to test and debug his or her code.

This way of testing is not very different from any standard test procedure that should be done in each software project; i.e., internal interfaces, internal functionality, appearance and consistency, usability, and other factors can all be evaluated.

Even after bring-up hardware is available, every code change has to be pretested in office mode to reduce the amount of test time required on the bring-up hardware. While this method provides sufficient test coverage for a wide range of SE functionality, it definitely has shortcomings in areas where the actual response from the hardware is not predictable via simple algorithms. In addition, the actual communication between the SE and the code running on the zSeries PUs cannot be tested.

SE-to-CECSIM connection

The code running on the zSeries PUs below the operating system level (millicode, microcode, i390 code) is tested on the CECSIM simulator [5]. In order to test the actual communication between this code and the SE code, these two components are connected via a TCP/IP connection. Since this setup is different from the structure that is present in a real system, this requires extensions and modifications on both sides. A TCP/IP stub is required for the CECSIM simulator; it sends and receives the data that is transferred on real systems via a state machine located in the CEC clock chip. On the SE side, the standalone simulation has to be altered to allow the communication stream to and from the PUs to be passed via the TCP/IP network, while all other communication requests have to be treated in the same way as in the standalone simulation.

The simulation environments are started separately, and the TCP/IP connection is established during the SE initialization phase (often referred to as "SE warmstart"). The actual communication, however, does not begin until a certain point in the initial microcode load (IML) sequence (also known as power-on reset) has been reached. At this point (on a real system), the SE issues a command to the hardware which starts the clocks and causes the previously loaded code to start the communication [8]. Since these kinds of hardware accesses are not possible in CECSIM mode, this command is modified to send a "shoulder tap" to the CECSIM, which synchronizes both simulation environments. In this setup, the shoulder tap synchronizes the two simulation environments with respect to the current flow of events. Both simulation environments from that point on continue with their respective simulations while the actual communication between the SE and the code running on the zSeries MCM is executed in the same way as it would be on a real system.

This method allows verification of the communication between the support element and the zSeries millicode and microcode, as well as the routing of commands to the zSeries operating system. In particular, the complete IML sequence can be verified in advance. Given that the amount of data that is exchanged during a single IML of a typical system is of the order of magnitude of several MB, and that multiple communication partners are involved on either side, the value of this type of simulation can hardly be overestimated, especially if the fact is considered that a successful IML is one of the first major milestones in real hardware bring-up once the first bring-up hardware is available. With the SE-to-CECSIM connection, often referred to as "CECSIM-BUV" [5], both the code running on the SE and the millicode and microcode running on the zSeries CPUs can be verified to work together prior to the availability of any bring-up hardware.

Since the CECSIM environment is based on a previous generation of IBM mainframe hardware, the execution time for the communication commands is at least of the same order of magnitude as on the actual hardware. However, there exist natural limitations to this sort of simulation, since the communication path is different. That is, the cage controller hardware and software are not part of the setup, and the connection is routed over the existing network hardware and may be affected by other

network traffic. Thus, some classes of problems, such as timing problems, are not detectable in the CECSIM-BUV environment. However, all problems that are related to the content of the communication are detectable in the CECSIM-BUV environment (see **Figure 6**).

Connecting the SE to the CoBALT system

One of the tasks of the SE is to access certain hardware registers of the mainframe directly by reading and writing so-called "scan rings" in certain chips on various components of the hardware. Therefore, the SE must have explicit knowledge about the hardware characteristics of a system [Engineering Data Interface (EDI), initial patterns (INPA), etc.] This knowledge is first required during the early phases of the initialization, but even during normal operation of a z900 there are many different kinds of hardware-related information that are accessible only via these hardware registers. Among the most critical and time-consuming tasks in the early bring-up phases has always been the verification of the layout and functionality of these registers in relation to the documented hardware design. With the availability of a hardware emulation as provided by the CoBALT system [4], there now exists the opportunity to create a model of the actual hardware from the VHDL design data prior to the availability of the first prototype hardware. By connecting this model with another modified version of the SE standalone simulator, these hardware access methods can be verified and tested in the time frame between the stabilization of the VHDL design and the availability of the first hardware. By connecting an SE to the CoBALT system, several areas can be verified prior to the availability of the first hardware: the actual accessibility of the hardware registers, the size of the scan rings as defined in the engineering data, the meaning and function of each latch, and the correct initialization pattern, to name only a few.

The connection to the CoBALT system is established in the same way as the connection to the CECSIM environment, during the "SE warmstart." Unlike that for CECSIM, however, the communication is started as soon as the SE attempts to access the hardware via scan rings. Since the speed of the emulator is necessarily several orders of magnitude slower than the speed of the actual hardware it is emulating, several aspects have to be considered when connecting an SE to the CoBALT system. First, all relevant explicit time-out values in the SE must be adapted or completely disabled. Next, the code must be checked for implicit timing dependencies, which must also be eliminated. Finally, the scope of the simulation must be adapted to focus first on the most relevant parts of the hardware access.

Nevertheless, with the SE-attached CoBALT system, we have been able to simulate not only the complete hardware initialization but also the hardware-related parts

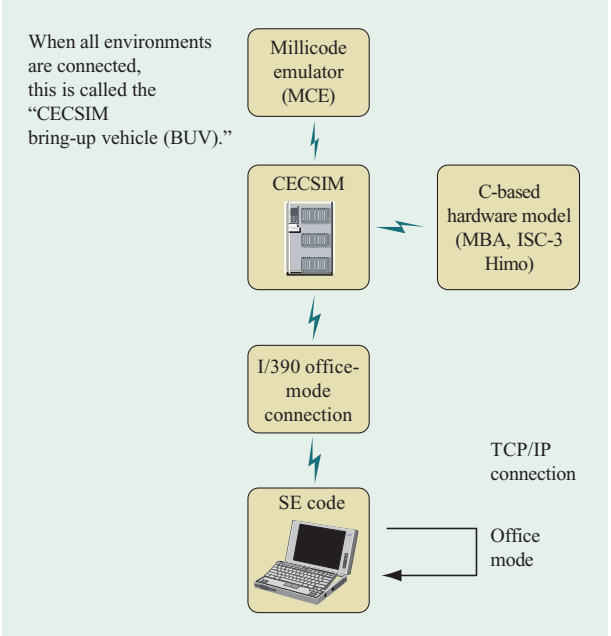

Figure 6

Code simulation environment.

of the IML up to the point where the first code is loaded into the PUs and the clocks are started. Since this kind of information is relevant even before any code can be loaded into the MCM, the scope of this environment is slightly different than at the CECSIM-BUV. In addition, since the clock chip is part of the CoBALT environment, it is also possible to actually load and exercise millicode, microcode, and i390 code in this environment and verify the actual communication path.

While the CECSIM-BUV focuses primarily on the simulation and verification of communication and code flow during the later phases of an IML, the SE-to-CoBALT connection focuses on hardware-status-related information.

In combination, these three simulation environments (SE standalone office-mode simulation, SE-to-CECSIM connection, and SE-to-CoBALT emulation connection) cover almost all aspects of hardware and microcode verification without the need for availability of actual hardware.

Setup

The setup of these simulation environments on the SE simulator is controlled by a set of run-time switches in the code and a parameter file that contains the IP addresses or domain names of the desired CECSIM or CoBALT counterparts, the IP port to use, the type of simulation, and other relevant information. The layer that directly

controls the actual PC hardware access is replaced by a special simulation version that checks the run-time switch settings, parses the parameter file, and sets up the connections to the simulation/emulation environments. Normally the CoBALT system or the CECSIM application is started first and waits at an early phase for the SE to be started. With the correct settings, the SE then establishes the connections and the simulation continues.

Concluding remarks

In this paper, we have summarized previous system microcode verification efforts—for the IBM Enterprise System/9000 Models 820 and 900, IBM S/390 Enterprise Server Generations 3, 4, 5, and 6—and current verification efforts on the IBM eServer z900. Efforts on the latter resulted in a significant (approximately three months) reduction in its development time. Furthermore, we have described the process changes and common network connections needed to support the appreciable advance in system microcode verification achieved for the latter system. Our objective was the removal of as many problems as possible to reduce development time. However, code and hardware problems were found during real system bring-up. An escape analysis was done, and it became evident that problems arose where simulation shortcuts were introduced to simplify the environment and expedite the testing.

In that regard, strategic enhancements must be made to the processes and simulation environments. Further efforts will be required on the integration of *all* microcode components into the virtual power-on process, with the aim of reducing and if possible eliminating all simulation shortcuts. Also, emulator capacity must grow at the same rate as system hardware design. Hence, it will be important to integrate the latest emulator technology into microcode simulation concurrently with ongoing verification efforts.

References

1. D. K. Beece, G. Deibert, G. Papp, and F. Villante, "The IBM Engineering Verification Engine," *Proceedings of the 25th Design Automation Conference*, IEEE, June 1988, pp. 214–218.
2. D. F. Ackerman, M. H. Decker, J. J. Gosselin, K. M. Lasko, M. P. Mullen, R. E. Rosa, E. V. Valera, and B. Wile, "Simulation of IBM Enterprise System/9000 Models 820 and 900," *IBM J. Res. & Dev.* **36,** No. 4, 751–763 (1992).
3. S. Koerner and S. M. Licker, "Run-Control and Service Element Code Simulation for the S/390 Microprocessor," *IBM J. Res. & Dev.* **41,** No. 4/5, 577–580 (1997).
4. J. Kayser, S. Koerner, and K.-D. Schubert, "Hyper-Acceleration and HW/SW Co-Verification as an Essential Part of IBM eServer z900 Verification," *IBM J. Res. & Dev.* **46,** No. 4/5, 597–605 (2002, this issue).
5. J. Von Buttlar, H. Böhm, R. Ernst, A. Horsch, A. Kohler, H. Schein, M. Stetter, and K. Theurich, "z/CECSIM: An Efficient and Comprehensive Microcode Simulator for the IBM eServer z900," *IBM J. Res. & Dev.* **46,** No. 4/5, 607–615 (2002, this issue).
6. B. D. Valentine, H. Weber, and J. D. Eggleston, "The Alternate Support Element, a High-Availability Service Console for the IBM eServer z900," *IBM J. Res. & Dev.* **46,** No. 4/5, 559–566 (2002, this issue).
7. A. Bieswanger, F. Hardt, A. Kreissig, H. Osterndorf, G. Stark, and H. Weber, "Hardware Configuration Framework for the IBM eServer z900," *IBM J. Res. & Dev.* **46,** No. 4/5, 537–550 (2002, this issue).
8. F. Baitinger, H. Elfering, G. Kreissig, D. Metz, J. Saalmueller, and F. Scholz, "System Control Structure of the IBM eServer z900," *IBM J. Res. & Dev.* **46,** No. 4/5, 523–535 (2002, this issue).

Received September 24, 2001; accepted for publication April 12, 2002

Stefan Koerner *IBM Server Group, IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen (koerners@de.ibm.com).* Mr. Koerner is a Senior Engineer in the IBM eServer z900 Hardware Development Group in the Boeblingen laboratories. He joined IBM in Boeblingen in 1981 after receiving an M.S. degree in electrical engineering from the Technical University of Furtwangen, and has held a number of positions in logic design, microcode development and hardware verification. He was the technical leader for the microcode verification and emulation of the IBM S/390 G7 system. Mr. Koerner holds three patents, is the author of 12 technical papers, and received an IBM Outstanding Innovation Award in 2001. He is currently the technical leader for microcode verification in the IBM Enterprise Systems Group.

Martin Kuenzel *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen (kuenzel@de.ibm.com).* Dr. Kuenzel joined the IBM Boeblingen laboratories in 1996 after completing his Ph.D. thesis in solid-state physics at the RWTH Aachen. Since then, he has worked in various areas in the zSeries Support Element Development Department of the IBM Server Group. He is currently the team leader of the Cage Communication Support Layer Development team. Dr. Kuenzel received an IBM Outstanding Technical Achievement Award for his efforts on Multiprise 3000 SE hardware access and an IBM Outstanding Technical Achievement Award for his efforts on the design and development of the IBM eServer z900.

Edward C. McCain *IBM Enterprise Server Group, 2455 South Road, Poughkeepsie, New York 12601 (mccain@us.ibm.com).* Mr. McCain is currently an Advisory Verification Engineer and Team Leader for the S/390 emulation program. He joined IBM in 1982 and has worked on engineering systems testing for the IBM 308X, 3090, ES/9000, and the S/390 G3, G4, G5, and G6. He has received a Leadership Award for his work on PR/SM & MPG, a Division Award for his work on the ES/9000, Excellence Awards for his work on S/390 Parallel Sysplex EDVT testing, S/390 G4 functional test leadership, S/390 G6 EST project leadership, and an IBM Outstanding Technical Achievement Award for his work on zSeries verification.