Key Features of the Design Methodology Enabling a Multi-Core SoC Implementation of a First-Generation CELL Processor

Dac Pham, Hans-Werner Anderson, Erwin Behnen, Mark Bolliger, Sanjay Gupta, Peter Hofstee, Paul Harvey, Charles Johns, Jim Kahle, Atsushi Kameyama[1], John Keaty, Bob Le, Sang Lee, Tuyen Nguyen, John Petrovick, Mydung Pham, Juergen Pille, Stephen Posluszny, Mack Riley, Joseph Verock, James Warnock, Steve Weitzel, Dieter Wendel

IBM Systems and Technology Group, Austin, TX
[1]Toshiba America Electronic Components, Austin, TX

Abstract-- **This paper reviews the design challenges that current and future processors must face, with stringent power limits and high frequency targets, and the design methods required to overcome the above challenges and address the continuing Giga-scale system integration trend. This paper then describes the details behind the design methodology that was used to successfully implement a first-generation CELL processor - a multi-core SoC. Key features of this methodology are broad optimization with fast rule-based analysis engines using macro-level abstraction for constraints propagation up/down the design hierarchy, coupled with accurate transistor level simulation for detailed analysis. The methodology fostered the modular design concept that is inherent to the CELL architecture, enabling a high frequency design by maximizing custom circuit content through re-use, and balanced power, frequency, and die size targets through global convergence capabilities. The design has roughly 241 million transistors implemented in 90nm SOI technology with 8 levels of copper interconnects and one local interconnect layer. The chip has been tested at various temperatures, voltages, and frequencies. Correct operation has been observed in the lab on first pass silicon at frequencies well over 4GHz.**

Index Terms—CELL Processor, multi-core, SOC, SOI, modularity, re-use, 64-bit Power Architecture, multi-threading, synergistic processor, flexible IO, Linux, multi-operating system, virtualization technology, real-time system, hardware content protection, correct-by-construction, thermal management, power management, clock distribution, high-performance latch, local clock buffer, design hierarchy, design environment, design dependency solution, linear sensor, digital thermal sensor.

I. Introduction

The architectural vision of "bringing supercomputer power to everyday life" is the driving force behind the CELL processor design, setting a new performance standard by exploiting parallelism while achieving high frequency [1]. CELL is designed for natural human interactions: photo realistic, predictable real time response, and virtualized resource for concurrent activities. CELL supports multiple operating systems including Linux, and is designed for flexibility with a wide variety of application domains. Other attributes include hardware content protection, and extensive single-precision floating-point capability. By extending the Power Architecture with Synergistic Processor Elements (SPE) having coherent DMA access to system storage and with multi-operating system resource management, CELL supports concurrent real time and conventional computing. With a dual-threaded Power Processor Element (PPE) and eight SPEs this implementation is capable of 10 simultaneous threads and over 128 outstanding memory requests.

The First-Generation CELL processor consists of the PPE and its L2 cache, eight SPEs [2] each with its own local memory (LS) [3], a high bandwidth internal Element Interconnect Bus (EIB) [4], two configurable non-coherent I/O interfaces, a Memory Interface Controller (MIC), and a Pervasive unit that supports extensive test, monitoring, and debug functions. The high level chip diagram is shown in figure 1 below.

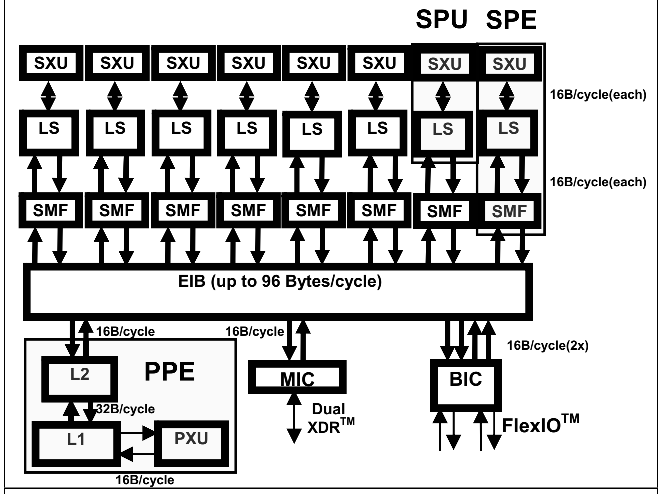

Fig. 1: Processor high level diagram

II. The Design Challenges for Giga-Scale Integration

II.1. Power & Frequency Walls

Over the last decade, technology scaling has resulted in leakage power increases of over 1000X (fig. 2). With gate dielectrics and other device features fast approaching fundamental limits, a continuation of historical trends would see passive power surpassing active power within the next few years. Furthermore, the technique of increasing frequency by deepening the pipeline has reached a point of diminishing performance returns if power is taken into consideration. In the face of this power/performance wall, increased design efficiency becomes essential. These factors drove the decision to support a wider processor issue width (e.g. multi-threading) and to increase the number of architected registers.

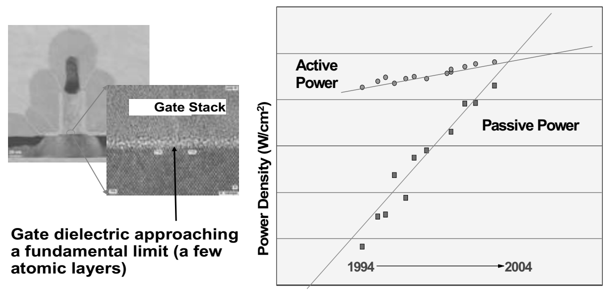

Fig. 2: Power Wall

II.2. System Trends and Giga- Scale Integration

Increased system integration is driving processors to take on many of the functions typically associated with the system: off load and acceleration, and integration of bridge chips as shown in figure 3.

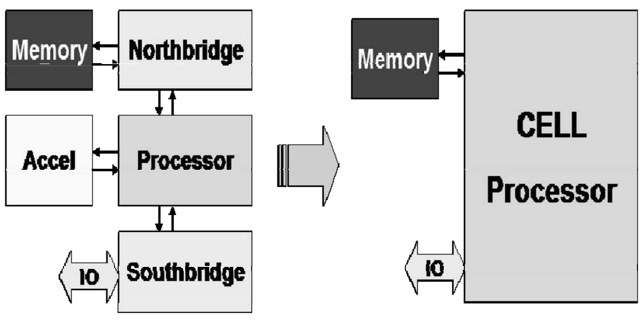

Fig. 3: System Trends

III. Design Implementation To Address Power And Frequency Walls

III.1. Components and Libraries Design

Given a short cycle time target, a significant amount of the chip power is consumed by latches, flip-flops, and other clocked elements. However, the delay overhead imposed by standard flip-flops is considerable. Therefore, a rich set of latches and flip-flops were developed to allow for both power and delay optimizations. The basic local clock splitter components are shown in figure 4. In addition to test controls, the base block accepts a local clock gating signal, with a small setup time relative to the falling global clock (cycle boundary). Input setup and hold times are specified against the falling clock edge, as a result of the built-in latching action of the base block. Local clocks, to drive typical master-slave flip-flops, are derived from the common output point of the base block.

For timing critical paths, a high-performance latch (HPL) [5, 6] was designed which combines a wide mux (up to 10-way), relying on a dynamic NOR gate, with a set-reset latch (fig. 5). The dynamic NOR starts evaluating with the launch clock, and the input data hold time is limited when all sel_b inputs are forced high after a fixed delay.

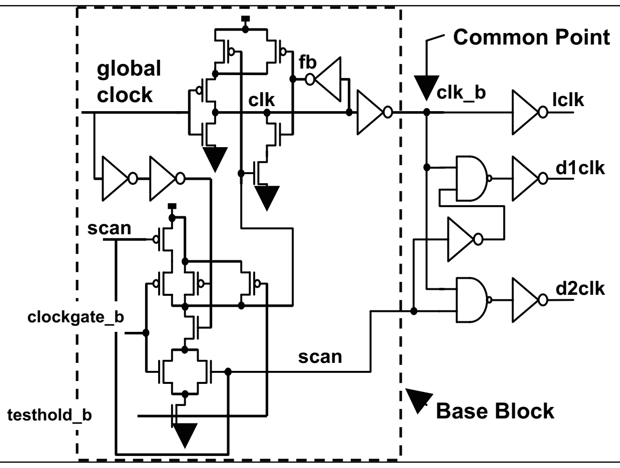

Fig. 4: Local Clock Generation

Dynamic circuits were used in several critical macros, in the arrays, and in PLAs. All dynamic macros were latch-bounded (macro-to-macro signals are static). Signals feeding dynamic logic were usually launched from the master portion of a flip-flop, and ANDed with the slave clock (lclk) to provide a signal which resets to 0 every cycle when lclk is low. Dynamic logic was always followed by a set-reset latch similar to that used for the HPL shown earlier.

In addition, various rules were adopted to ensure a "correct-by-construction" design methodology. All circuits used a common set of clocking components to ensure uniformity across the design, with no rotation of the components allowed. An extensive set of electrical and physical checks and audits were put in place. Finally, a customized series of yield-related checking rules was employed to ensure manufacturability of the chip.

The 90nm PD SOI technology offers three oxide thicknesses (thin oxide, thick oxide for high voltage device, and decoupling capacitor) and four different V_T settings for the thin oxide devices. Since power was such a critical design issue, static circuit implementations were favored for the majority of the design. A variety of static circuit families were used in full custom designs, with tuners and device width optimizers used for power-performance tuning. Higher threshold voltage devices were inserted wherever possible to cut down on leakage current (no low V_T devices were used), and the threshold voltage for the array devices was adjusted independently from that of the logic devices. Approximately 40% of the logic was implemented as synthesized random logic macros (RLMs), with the rest being full custom design.

The local clocking described in figure 4 has several important features. Overall clock latency and absolute clock uncertainty is minimized by this scheme since there are only three gate delays between the global clock input and the data launch clock (lclk). Also, the common point for both launch and capture clocks are at the output of the base block, minimizing the relative uncertainty between launch and capture clocks. When clocks are in the gated state, lclk is held inactive, and the capture clock is held high. The system state is therefore stored in the slave latch.

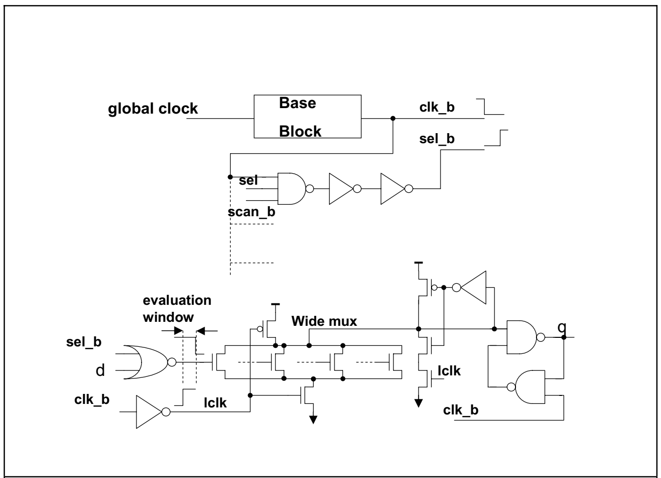

Fig. 5: High performance latch

For power reduction, the standard flip-flop can be run in pulsed-mode, with a clock configuration shown in figure 6. In this case the slave clock is pulsed in normal operation, and master clock is held high. There is also a "chicken switch" which allows running in normal master-slave clocked mode if race problems are seen in the hardware. A non-scannable pulsed latch was also supported, minimizing area, power, and latency in situations where a longer hold time could be tolerated.

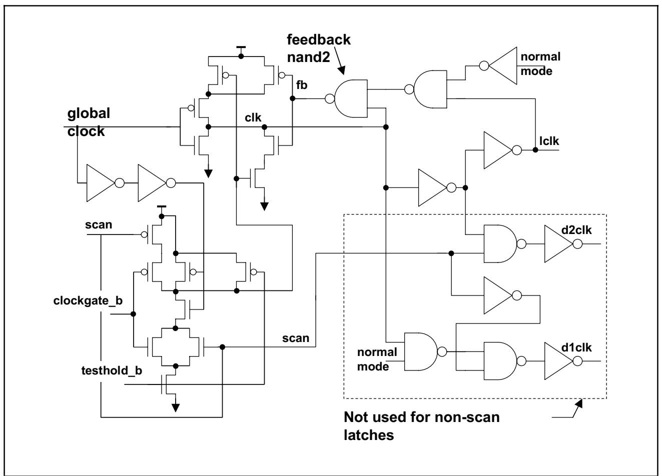

Fig. 6: Standard Flip-Flops

With the widespread use of pulsed latches, and the controlled use of clock delay elements, it was very important to have a robust methodology to check for race conditions. The timing methodology required a design margin to be applied which scaled with the total path delay of the racing paths (in addition to a certain fixed margin), as measured from the common point of divergence. This ensured that race conditions with larger uncertainties were designed with correspondingly larger margins.

III.2. Clock Distribution

The chip contains three distinct clock distribution systems, each sourced by an independent PLL, to support processor, bus interface, and memory interface requirements. A main high frequency clock grid covers over 85% of the chip, delivering the clock signal to the processors and miscellaneous circuits. Second and third clock grids, each operating at fractions of the main clock signal are interleaved with the main clock grid structure, creating multiple clock frequency islands within the chip. All clock grids were constructed on the lowest impedance final two layers of metal, and were supported by a matrix of over 850 individually tuned buffers. This enabled control of the clock arrival times and skews, especially on the main clock grid, which supports regions of widely varying clock load densities. As shown in figure 7, final worst-case clock skew across chip was less than 12ps. High frequency clock distribution optimization and verification needed models which included frequency sensitive inductance and resistance phenomena [7]. These models were built from data extracted from combined clock and chip power distributions, two dimensional cross sections, and capacitance models extracted from three dimensional sections. Reduced clock grid power dissipation was achieved through optimization of buffer drive strengths, grid wire periodicity, clock wire to return path spacing, and clock twig wire widths. Together, these techniques lowered clock distribution power dissipation by more than 20% compare to previous design [8].

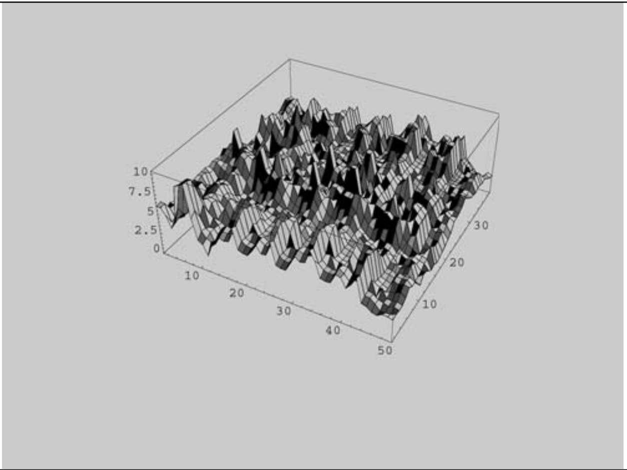

Fig. 7: Clock skew map

III.3. Thermal and Power management

This SOC presented new challenges in chip thermal design. The higher heat flux from smaller hot spots hindered spreading of the heat across the silicon substrate [9]. Extensive thermal analysis carried out early in the design cycle ensured that the maximum junction temperature, as well as the average temperature of the die, would end up within design specifications. Various workloads were simulated for each component and power maps were constructed. From these maps, a matrix of small power sources was created, for use with package and heat sink models. Thermal models were then created and used to simulate both steady state and transient thermal behavior. These data were analyzed to improve the design and floor plan of the chip, and also provided feedback for improved thermal sensor design (fig. 8).

Due to local heating caused by individual processing units, sophisticated local thermal sensing strategies and thermal control mechanisms were used to allow an aggressive low cost thermal design. The processor contains a linear sensor and 10 local digital thermal sensors. The linear sensor is essentially a diode connected to two external I/Os, used to measure the die's global temperature and to adjust the system cooling. The digital thermal sensors provide for early warning of any temperature increase and for thermal protection.

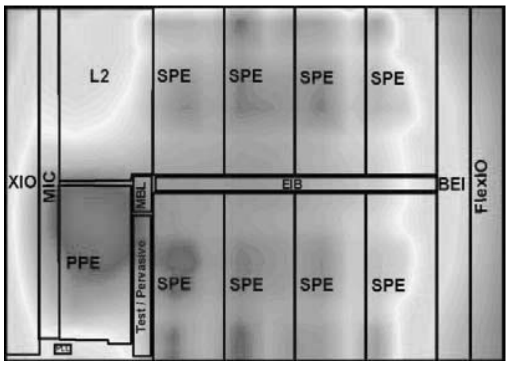

Fig. 8: Chip thermal map

III.4. Pervasive Design

The pervasive logic comprises all the function necessary for initialization, clock control, test, performance monitoring, and error checking and reporting. For a complex multi-core processor, the design of the pervasive logic is a key emphasis early in the design cycle. The pervasive function is implemented as a centralized controller and in distributed units across the chip. A Performance Monitor (PFM) is provided to assist with the debug and tuning of software applications, and an on board logic analyzer (LA) assists with hardware debug. Both the PFM and LA are capable of capturing information at speed from all units across the chip. The PFM and LA also provide the capability to view single or multiple units. Extensive debug and control capabilities are provided that can be accessed via an IEEE 1149.1 interface. For manufacturing support, the pervasive unit provides array and logic built in self test (BIST) engines. The ability to scan at speed is provided to assist with detection of AC related faults. Electronic fuses are used extensively for array repair and selected chip personalization.

III.5. Physical Design

Figure 9 shows the die photo with roughly 234M transistors from 17 physical entities, 580K repeaters and 1.4M nets implemented in 90nm SOI technology with 8 levels of copper interconnects and one local interconnect layer.

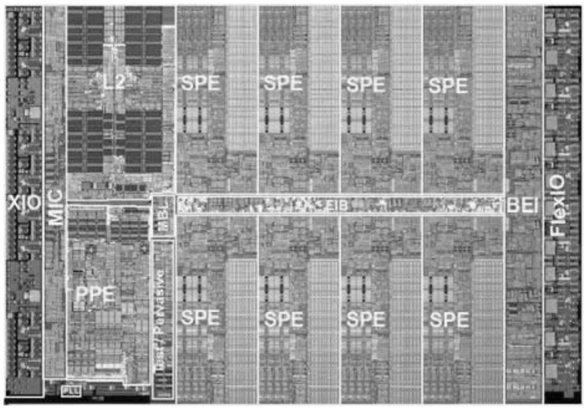

Fig. 9: Die Photo

At the center of the chip is the EIB, composed of four 128-bit data rings plus a 64-bit tag operated at half the processor clock rate. The wires were arranged in groups of four, interleaved with GND and VDD shields twisted at the center to reduce coupling noise on the two unshielded wires. To ensure signal integrity, over 50% of global nets were engineered with 32K repeaters. The SOC uses 2965 C4s with four regions of different row column pitches attached to a low cost organic package. This structure supports 15 separate power domains on the chip, many of which overlap physically on the die. The processor element design, power and clock grids, global routing, and chip assembly support a modular design in a building block like construction.

IV. KEY FEATURES OF DESIGN METHODOLOGY

IV.1. Hierarchical Design and Rule-Based Optimization Methodology

IV.1.1. Design Environment and Database Structure

There were many challenges in meeting the defined objectives for setting up the design environment and database system for the first-generation CELL processor project. First, the methodology had to support concurrent design execution of each *partition* (major core of the design such as PPE or SPE); meaning design work had to be done simultaneously and independently by different teams located in different geographical areas. The existing inherently hierarchical nature of the design was carefully considered when defining the physical partitions in order to minimize the impact of creating discrete physical partitions. Strict *naming convention* schemata were applied to the entire *design hierarchy* to facilitate parallelization and to prevent collisions.

Second, the *database structure* had to support both the hierarchical objective and also multiple design disciplines, namely logic design and verification, physical design and verification, integration, and timing, etc., to allow for efficient schedule interlocking. An AFS network file system was used to allow transparent access to design data by all team members across multiple geographical locations. Additionally, the database structure had to support common design libraries and many "shared" macros used in multiple units or partitions. Any dependency conflicts caused by usage of different levels of these libraries and macros across the design hierarchy were resolvable by *design dependency solution* algorithms, supported by the design environment.

Third, the *design environment* had to fully support the custom processor design methodology by providing tools, processes, and a workspace for every designer. The design environment fills the vital linkage between designers and the supporting database and must do so in a simple and effective way. Design environment initialization was simply done with a single command to set up all required tools and environment variables necessary for design work and the database interface.

IV.1.2. Front End Design and Verification Methodology

The front end logic design is captured in VHDL with all the verification done at the behavioral level. The chip verification uses Top down Specification / Bottom up Implementation strategy. For custom circuits, the schematic netlists and behavioral VHDL are verified for correctness with equivalency checking tools.

The design is divided into partitions, islands, units, and modules. All the verification environments and test coverage needed to create a high quality chip is planned during *High Level Design* phase. The verification process is hierarchical with all the environments and checkers created at lower level being used in the higher level environments. For performance and throughput purposes, there are options to turn off some checkers during run time. The test plan is based on the coverage plan to guarantee 100% coverage with written tests. The coverage is also hierarchical i.e. lower environments designate what portions of the lower level coverage needs to be hot at higher levels. Extra coverage and checkers are also added at higher levels for corner cases.

For a complex and large design such as CELL, a cycle based simulator is used for all the simulation. Both C++ based and Specman languages are support in the verification flow. Apart from the specialized test case generators used in processor core verification; Specman, C++ and Perl test cases are used for the rest of the design. Formal verification is also done at module level at various parts of the chip. Special tools were employed for Asynchronous clock boundary verification since the simulator used is cycle based.

In addition to functional verification, pervasive design is also verified at various levels. This includes Scan verification, POR verification, Test mode verification, RAS verification, Trace and Debug Bus verification, etc. Hardware based accelerators are also used for software workloads, Boot code, and OS boot verification.

The Grid computing usage for processor design is demonstrated in this project: over 1.5 trillion simulation cycles or about 2 million hours of simulation was completed over multiple Sim farms spanning throughout IBM US & Germany. This is one of the key attribute for over 98% of total logic bugs found, the processor core VHDL model booted Linux, and Chip Bring up exercisers ran in simulation prior to design tape out.

IV.1.3. Physical Synthesis

The increased volume of synthesized logic on the CELL processor requires maximizing the productivity of the random logic macro (RLM) designers. This is accomplished by accelerating timing closure and automating the build process.

The design of RLMs used physical synthesis to accelerate timing closure. Physical footprints were imported into the synthesis tool to allow accurate timing estimations during netlist creation and placement. The placed, optimized netlists were then fed into the physical build process. Early estimated abstracts allowed for synthesis and sizing before final contracts were available from the unit integrator.

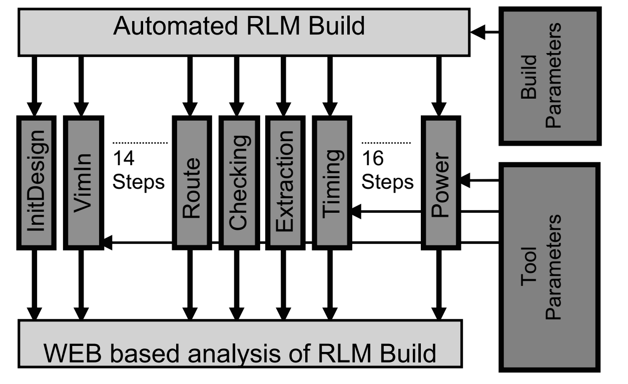

Fig. 10: Automated RLM Build Process

The RLM physical build process was streamlined and automated as shown in figure 10. This was accomplished by creating a supervisor program to "drive" over 30 individual design steps from netlist import through final checking. The supervisor script used customizable templates to control the individual tool interfaces, allowing designers an automated solution with the flexibility of a manual build flow. Job management was further improved by the *Report Generation's Tool* (XRG), which generated web based reports that allowed designers to easily identify failed job steps and quickly access log files.

To ensure high-availability of the tool set, automated daily regression tests were performed that exercised the build process and evaluated the results. This helped identify problems before they were encountered by designers.

Custom methodology checks were implemented to ensure that RLMs met design specifications before being delivered to unit integrators.

IV.1.4. Static Timing Methodology

To simplify timing closure and reduce runtime, all latches were modeled for the late mode timing run in the nominal process corner as non-transparent to remove timing loops. Custom designers; however, were still able to use cycle stealing techniques with an internally developed algorithm, which allowed the designer to specify the effective cycle boundary point within a given window of transparency. A timing adjust could be applied for all latches connected to a given LCB, allowing for an improved setup time, but delaying the launch of the data out of the L2 latch by a corresponding amount beyond that which the actual non-transparent latch modeling would require. To lower the cost for high-volume production, all Local Clock Buffers (LCB) and latches are designed to support at-speed scan to reduce manufacturing test time. For power management, each LCB included global and local clock gating signals. These signals have to work correctly on a cycle-by-cycle basis to allow switching from scan to functional mode in one clock cycle. At-speed scan operation allows us to time both functional and scan paths in a single timing run without the need to apply different phases to distinguish scan signals from regular ones.

For the early mode timing run in the fast corner, we wanted to ensure enough margins to cover a wide process range window needed for a high-volume product. To achieve this, we used the Linear Combination of Delays (LCD) feature of our Gate-level Static Timer. This feature allows combining different process corners [10]. Usually, the coefficients for the three corners, best, nominal, and worst, add up to be 1, e.g. 15% best, 70% nominal, and 15% worst case. We used a coefficient of 1.27 plus a fixed amount of offset for the worst case calculated timing delays. This allows an increased hold time margin by slowing down the clock propagation.

IV.1.5. Chip Integration & Physical Verification Methodology

The chip integration methodology was created to support parallel, concurrent design at high clock frequencies. Multiple levels of hierarchies were used to manage the design problem and enable concurrency. The high level design process consisted of top-down constraint setting which lead to the division of the design into functional islands and units. The constraints became a design budget for each floorplannable object. Those budgets dictated the size, aspect ratio, rectilinear outline, pin locations, and routing layers used for each object. The implementation process fulfilled the constraints passed down the hierarchy.

The integration methodology was tightly woven with timing throughout the design process. Very early in the Floorplanning process, timing shells represented each object in the hierarchy. These shells enable early timing feedback to drive partitioning, pipelining, and buffering decisions from the outset. As the data evolved through the design process, shell timing rules and Steiner estimates became schematic based timing rules with 2-D extracted parasitic and finally fully extracted timing rules and 3-D extracted routing parasitic. Buffering of signals is performed by an internally developed algorithm. Unit floorplans are filled with 4, 8, 16, or 32 bit buffer packs with all bits initially unused. A process based on Dijkstra's Algorithm finds the shortest path from source to sinks across available buffer packs. Routing was performed using a gridded router and 13 distinct non-default routing rules. Timing estimations that used particular non-default rules carried directly into the routing process, insuring that actual routes would mirror the estimation.

Later in the design cycle, each partition would analyze and correct coupled noise events predicted on closely routed nets. Noisy nets were fixed either through rerouting or by buffering. Electro-migration and missing via analysis on the power bus was also performed to insure that the power distribution met design requirements.

Physical verification of all Floorplan blocks consisted mainly of LVS, DRC, methodology checks, and formal Netlist verification. All physical verification is done with cover cells that represent fixed obstructions pushed down from the parent or the routing contract. Checking with these views insures that the object will not create a conflict when stitched into each level of hierarchy. Special methodology checks enforce specific design requirements beyond traditional design rules. This would include checks for pin accessibility, design shapes properly within the boundary, and power pins on proper pitch, among others. The formal verification process insured that the final, buffered Netlist was Boolean equivalent to the original vhdl description.

IV.2. Transistor Level Analysis

IV.2.1. Circuit & Array Methodology for an 11FO4 Design

For an 11FO4 design within an air-cooled power envelop, special emphasis was placed on power distribution, power consumption, clock distribution, signal distribution, variation due to hot spots, and inductance effects. Furthermore the chip team also had to plan for multiple clock domains, cross chip variations in delay, leakage, intra-chip interconnections, and array bit cell stability early in the design cycle. Strict design guidelines in layout and circuit topology were enforced to minimize design variations.

A major focus of the circuit methodology is on array design since memory arrays occupy an increasingly larger share of chip area and it is where more aggressive design techniques are used to ensure performance. There are three major challenges for array design at low voltage levels: stable cell operation (for functionality), leakage current reduction (for low power), and management of speed variations (for yield).

A critical part of the circuit/array methodology is a detailed statistical analysis of cell stability, leakage, and yield in the early design phase. This analysis will determine the optimal cell size for a given technology to achieve stability, power, and yield goals while reducing chip area. The analysis also helps guide the design team and the manufacturing team to decide on the device menu for the technology. A sample result of the statistical analysis is shown in Figure 11 below. This figure plots the failures of the cells at various voltage levels for the peripheral logic circuit (Vdd1) and the core cells (Vdd2). Note that as the design enters the sub-1V operating voltage range, it may be necessary to have a separate supply for the array core cells. This design decision will have significant impact across the whole methodology. Chip planning/integration, packaging, libraries, and tools will have to be adapted to support multiple supply domains.

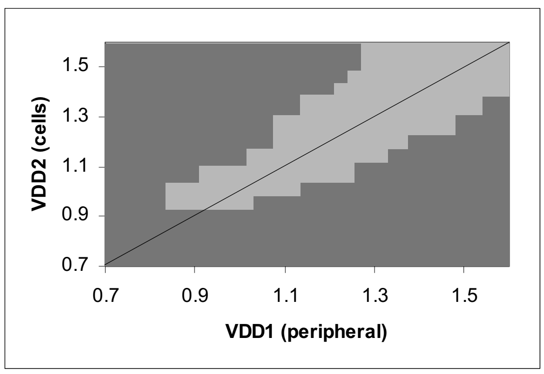

Fig. 11: Statistical analysis of SRAM cells – Vdd1 versus Vdd2 data

Transistor level analysis also plays an important role in the array verification methodology. For arrays, the high level design begins will the RTL and the implementation begins with the schematic design. There will be long lead time before layout is completed. So it is very important to have a methodology to provide accurate parasitic and interconnect models at the schematic level. The design methodology allows for a structured early floor-planning with accurate wire load models or Steiner based routing approximation to provide sufficient accuracy for schematic transistor level analysis. Logic extraction from array schematic is performed to build the test model, except for the array core which is synthesized from the high level RTL to reduce the model size. Then symbolic switch level simulation is run on the schematic and verified against the RTL as well as the test model. ATPG and marching patterns are also run on the array schematic using fast circuit simulator to verify the test patterns against the schematic.

IV.2.2. Transistor Level Timing

Static transistor level timing (TLT) was an integral part of the design methodology, with all custom macros, arrays, and even standard cell-based RLMs running through the tool, thereby providing comprehensive and consistent timing analysis and models. To meet aggressive frequency goals while satisfying area and power constraints, designers need to be able to quickly determine critical paths and delays in a circuit. Static timing at the transistor level using TLT helped achieve these goals [11]. The TLT team for the CELL project has improved the existing transistor-level timing methodology [12] in these four areas: improved timing margin calculation [13], local latch transparency modeling [14], pulse waveform timing in TLT templates [15], and improved method of timing model abstraction for simultaneously switching signals [16].

TLT is a transistor-level static timing tool that extends the capabilities of a Gate Level Static Timer to the transistor level. These extensions include a state analysis engine that is used to understand the timing behavior of groups of transistors and build timing models for them, and a fast circuit simulator [17] that is used for calculating propagation delays/waveforms through these transistor groups.

TLT uses piecewise-linear waveforms (rather than ramps) for timing accuracy and a modified version of AWE/RICE [18, 19] to propagate these waveforms through RC interconnect. TLT runs on flattened netlists from either schematics or extracted physical data. In addition to generating transistor level timing reports, it compiles a timing model or rule that is used for static timing at higher levels of the design hierarchy.

IV.2.3. Modularity and Integration of Black Box IP

The architectural modularity of the CELL processor also projects into the physical domain, where all 8 SPEs are instantiations of a single SPE design partition. To make this work correctly, the interaction between the local SPE layout and the global physical design structures had to be identical at all 8 locations where each SPE is instantiated. The C4 footprint, power busses, clock sector buffers, pervasive elements, and EIB components all had to be designed upfront to fit into this scheme in a modular way. The other extreme was taken with the integration of the high-speed IO and Memory interfaces on the left and right side of CELL [20]. These partitions were designed by a 3rd party vendor as “black box” IP, and used all layout resources from the silicon up to the C4 pins over their area. The only interaction with the core/chip happened at the boundary where predetermined power and signal pins were provided to cross the interface.

V. CONCLUSION

In conclusion, special circuit techniques, rules for modularity and reuse, customized clocking structures, and unique power and thermal management concepts were applied to optimize the design [21]. Correct operation has been observed in the lab on first pass silicon at frequencies well over 4GHz as shown in figure 12.

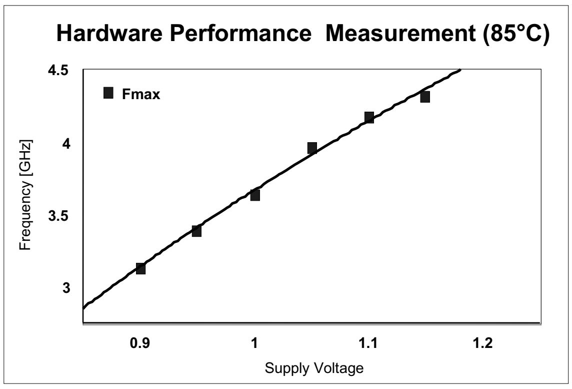

Fig. 12: First pass hardware in the Lab

VI. Acknowledgments

The authors gratefully acknowledge the many contributions from the entire Sony-Toshiba-IBM team who worked tirelessly side-by-side on the design of this processor.

VII. References

[1] D. Pham et al, "The Design and Implementation of a First-Generation CELL Processor", ISSCC 2005 Digest of Technical Papers, Feb. 2005, pp. 184-185.
[2] B. Flachs et al, "The Microarchitecture of the Streaming Processor for a CELL Processor", ISSCC 2005 Digest of Technical Papers, Feb. 2005, pp. 134-135.
[3] T. Asano et al, "A 4.8GHz Fully Pipelined Embedded SRAM in the Streaming Processor of a CELL Processor", ISSCC 2005 Digest of Technical Papers, Feb. 2005, pp. 486-487.
[4] S. Clark et al, "IBM CELL Interconnect Unit, Bus and Memory Controller", Hot Chip'05, Aug. 2005, Paper #1.2
[5] F. Klass, C. Amir, A. Das, K. Aingaran, C. Truong, R. Wang, A. Mehta, R. Heald, G. Yee, "A New Family of Semi-dynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors", IEEE J. Solid State Circuits, vol. 34, pp. 712-716 (1999).
[6] L. Sigal, J.D. Warnock, B.W. Curran, Y.H. Chan, P.J. Camporese, M.D. Mayo, W.V. Huott, D.R. Knebel, C.T. Chuang, J.P. Eckhardt, and P.T. Wu, "Circuit Design Techniques for the High-Performance CMOS IBM S/390 Parallel Enterprise Server G4 Microprocessor", IBM J. Res. & Dev. Vol 41 pp. 489-503 (1997).
[7] P. J. Restle, et al, "A Clock Distribution Method for Microprocessors", *IEEE J. Solid-State Circuits*, vol. 36, pp 792-799, May 2001
[8] P. J. Restle, et al, "The Clock Distribution of the Power4 Microprocessor", *IEEE International Solid-State Circuits Conference* 2002 Digest of Technical Papers, vol. 45, pp 144-145
[9] K. Yazawa and M. Ishizuka, "Thermal Modeling with Transfer Function for the Transient Chip-On-Substrate Problem", *Thermal Science and Engineering*, vol. 13, No. 1, Heat Transfer Society of Japan, 2005, pp. 37–40
[10] Posluszny, S. et al. "Timing Closure by Design," Proceedings for the 37th Conference on Design Automation, vol.37, pp.712-717, June 2000.
[11] Rao, V., J. Soreff, T. Brodnax, and R. Mains, "EinsTLT: Transistor Level Timing with EinsTimer," Proc. Of Int. Workshop on Timing Issues (TAU), 1999.
[12] Lee, Sang Y, J. Warnock, E. Behnen, J. Soreff, V. Rao, and S. Posluszny, "Improved Transistor-Level Timing Methodology for a CELL Microprocessor," ASPDAC 2006 *(submitted for publication)*
[13] Warnock, J.D., Erwin Behnen, Sang Y. Lee, and Jeffrey Soreff, "Improved Method for Timing Margin Calculation," *IBM Invention Publish*, Feb. 2004.
[14] Behnen, E., Jeffrey Soreff, James D. Warnock, and Dieter Wendel, "Method to Apply Latch Transparency Locally While Avoiding It Globally During Timing," *Filed with U.S. Patent Office*, May 2004.
[15] Soreff, J., Vasant Rao, James D. Warnock, Sang Y. Lee, and David Winston, "Pulse waveform timing in EinsTLT templates," *Filed with U.S. Patent Office*, May 2004.
[16] Warnock, J.D. and Jeffrey Soreff, "Improved Method of Timing Model Abstraction for Circuits Potentially Simultaneously Switching Internal Signals," *Filed with U.S. Patent Office*, May 2004.
[17] Devgan, A. and R.A.Rohrer, "Adaptively controlled explicit simulation," *IEEE Trans. Computer-Aided Design*, vol. 13, pp.746-762, June 1994.
[18] Pillage, L.T. and R.A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, vol. 9, No. 4, pp. 352-366, April 1990.
[19] Ratzlaff, C.L, N. Gopal, and L.T. Pillage, "RICE: Rapid interconnect circuit evaluator," *IEEE Trans. Computer-Aided Design*, vol. 13, No. 6, pp. 763-776, June 1994.
[20] K. Chang et al, "Clocking and Circuit Design for a Parallel I/O on a First-Generation CELL Processor", ISSCC'05 Paper #28.9
[21] Pham, D. et al. "Overview of the Architecture, Circuit Design, and Physical Implementation of a First-Generation CELL Processor," JSSCC, October. 2005 Special issue *(submitted for publication).*