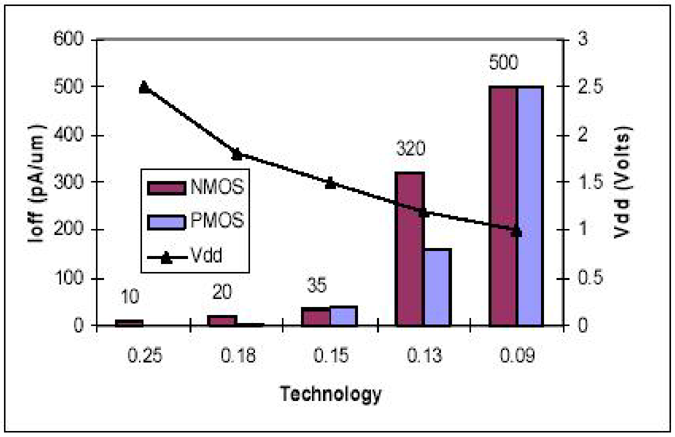
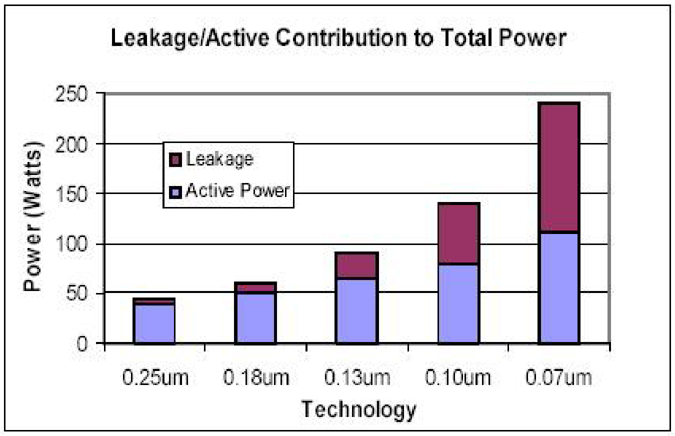
High Speed Design Techniques in SoC – From Architecture to layout

Dr. Danny Rittman, April 2004

Abstract

System-on-a-chip (SoC) ASIC technology is one of the most effective ways to produce high-speed, low-power products. Various techniques can be used to reduce the power consumption and increase the speed of SoC designs. The continuing advances in process technology give us the ability, in principle, to design ever-more-complex systems-on-chip at higher speeds. Hence those complexities, combined with the more complex device and interconnect models that these processes require, create a design crisis in which designers spend more and more time on iterating through cycles of synthesis, place and route, physical design and verification. System-on-chip solutions typically include high-speed, high-bandwidth mixed-signal interfaces; large, complex digital blocks that implement multilayer protocols; and significant amounts of on-chip memory. Designers of those devices push the limits of our EDA tools.





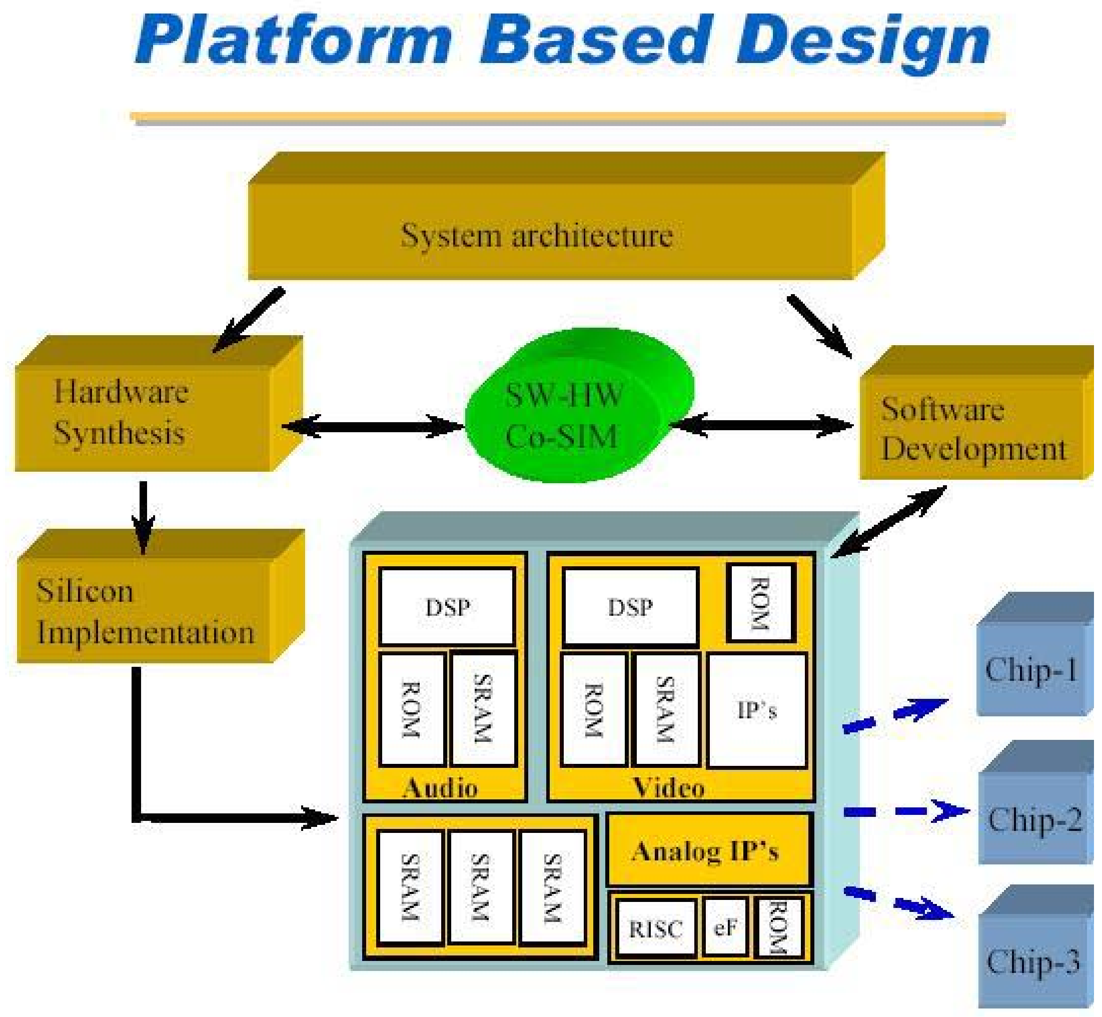
Power considerations in SoC's - Source: EETimes June 2002

Generating a design that is truly correct by construction, we need a design flow that can predict physical-layout parameters based on accurate models early in the design process. If that proves intractable, we may need to develop a radically new approach in which we give up some measure of density and performance in order to achieve reasonable design times, just as we did in the transition from full-custom to standard cells. Next-generation communication applications present some of the more challenging issues in IC design due to its high speed nature. The constant demand for higher speeds in SoC creates an entire world of challenges for designers.

The Evolution of platform-based SoC designs

Initially, ASICs were used to replace glue logic. These were assembled manually as a schematic at the transistor or gate level with almost no reuse of previously used logic or functions. With the rise of RTL languages such as Verilog and VHDL, EDA tools emerged to simulate and synthesize logic from an RTL description to a gate-level netlist. Reusability emerged in cell-based libraries and portions of reusable HDL code. The ability to reuse HDL functional code from one design to the next led to the beginnings of a block-based design methodology. Blocks could be described in RTL, synthesized into gates and laid out in a physical implementation as virtual components (VC) referred to as soft, firm or hard cores.

The advancement of process technology approaching 60 nm from 0.13 micron only a few years ago has opened up a significant number of new applications that can be integrated onto a single chip. Complexities of few millions gates are now moving to ten's million-plus gates with hundreds million gates in sight. It would be a challenge to simply maintain the design cycles of 8 to 16 months of a few years ago with this increased complexity. However, demand in consumer and communications products for new features and capabilities is driving market windows down; the upshot is that those 8- to 16-month design cycles are now approaching even shorter time with derivative products requiring shorter introduction times. Consumers are demanding more functionality in smaller packages at a lower price, which is yielding to the requirement for full systems to be integrated onto a single chip, known as system-on-chip, or SoC.



Source: National Semiconductors 2003

with off-chip interfaces to memory and real world communications framed with an SoC integration architecture or busing scheme. SoC designs include both hardware and software components that together implement the desired system functionality. Examples of SoC applications include cellular phones, PDAs, set-top boxes, portable consumer and Internet appliances, automotive engine controllers and network switches.

To meet the demands of SoC, reusability must encompass greater amounts of IP. Block-based reuse has yielded to subsystem reuse, and platform-based reuse is coming on. Platform-based design offers high productivity through extensive intentional reuse of known verified VCs that have undergone integration as a base SoC integration platform. Using this platform, or application-specific SoC integration platform, follow-on derivative products are created by adding or replacing the blocks that implement the derivative feature sets.

SoC Architecture

Any processor-driven SoC product requires a number of architectural functions. These include timers, DMA engines, interrupt controllers and memory controllers. In many cost-sensitive applications, a shared memory structure can be utilized to reduce memory component costs. Architecture is needed that addresses the memory needs of all devices without severely degrading the performance of any single device and yet offer flexibility to address a variety of architectures to support a wide range of applications. Proposed integration architecture should display the following attributes:

- Foundry, processor and technology independence
- Centered around shared memory
- Flexible to address a variety of SoC architectures
- Modular for a plug-and-play modification environment
- Easily synthesizable and works with standard design tools

Platform-based SoC design should not offer a burden when directed to different foundries and fabrication process rules. If the product has to be recoded to support another library, one of the major benefits of platform-based design is lost: time-to-market. Processor independence allows derivative applications to embed a processor that best fits those applications requirements. A processor-centric architecture makes this difficult; a memory-centric architecture reduces the problem of embedding a new processor typically to that of replacing the processor local bus bridge, usually only a matter of a few hundred gates.

The flexibility of the architecture allows derivative platform designs to change the number and type of peripheral blocks as well as the type of processor supported, for example a Von Neumann vs. a Harvard-type processor. Modularity is a key to making derivative changes efficiently and should provide a plug-and-play development environment so that derivative platforms are capable of being spun off relatively quickly. Obviously, if the architectural components are not able to work efficiently with today’s design tools and environment, efficient derivative designs will not be possible. This means that common bus attributes such as tri-stating, dual-edge clocking of signals, bus keepers and complex signal protocols make efficient use of design tools difficult.

One SoC architecture that has been offered to meet these criteria Palmchip's CoreFrame SoC Integration Architecture. It was designed with a blank sheet specifically to optimize it to SoC development and performance, rather than migrating a motherboard and bus model. As such, concerns such as routing and addressing that are important in motherboard design become irrelevant, while on-chip ones such as simplified design and interfacing can be optimized. The architecture does not use the traditional bidirectional bus concepts, which eliminates the need for tri-state bidirectional bus drivers. This enhances performance and simplifies on-chip design and verification using standard ASIC design tools.

Communication takes place through "channels" rather than on generic buses. The channel hardware transparently handles address and speed differences among various IP modules, allowing virtually any core to be used by simply providing a channel interface socket, which handles protocol, clock domain, bursting and bandwidth matters. Cores plug in to sockets in the CoreFrame architectural model. The socket channel model is set up to keep to a basic ASIC development flow and tools, which simplifies connecting IP modules into the architecture. DMA communications, CPU instruction and data fetching take place on separate channels, allowing independent high-speed data movement without tying up the CPU bus. Each peripheral appears to software as a FIFO, a relatively simple interfacing standard that facilitates quick and easy construction of the system. The channel-based approach can accommodate multiple clock domains through synchronization FIFOs to allow speed matching without loss of throughput.

High Speed SoC Designs

When talking about high performance, high speed design we have to keep in mind the basic key rule; High Performance = Low Power! One efficient solution is a system-on-chip (SoC) application specific integrated circuit (ASIC). The SoC ASIC provides the optimal mix between hardware and software, allowing functional components to be partitioned to provide the best mix of speed and power enhancements.

In particular, components that can gain from the benefits of hardware implementation will be implemented in hardware accelerators and discrete logic. Software is written to provide the necessary hardware initialization and configuration, but many time-extensive, number crunching operations (such as power-hungry) are provided by the hardware. The tradeoff is that programmability will be limited to the flexibility of the hardware accelerators. Considering lower power consumption, resulting higher speeds may be an acceptable compromise for many power conscious applications. Additional hardware interfaces, as well as software functionality, will help offset any programmability concerns.

SoC Speed Enhancement (Power reduction) techniques

Various techniques can be used to increase SoC speeds and to reduce the power consumption of SoC ASIC designs, including dynamic frequency control, dynamic power management and the ability to idle embedded processors. These techniques were developed during the past decade with the rapid evolution of SoC.

Dynamic frequency control

A SoC ASIC external reference clock and internal clock generator can be used to provide dynamic frequency control. The reference clock frequency is proportionally related to the SoC ASIC's power consumption (e.g. lower reference clock frequency results in lower power consumption).

The reference clock is provided by the system (host) and can be scaled (externally) based on the intended mode of operation. An internal clock generator can also be used to scale system clock frequencies (and power consumption) dependent on the desired mode of operation. This internal clock generator will contain a Phased Lock Loop circuitry (PLL) used for setting the internal clock rate.

The PLL logic contains three programmable dividers designated as reference, feedback, and output. The maximum and minimum values of the reference clock frequency input and the VCO output affect the phase jitter, which affects the ASIC's performance. Figure 1 shows a sample PLL-based variable clock-generator circuit.

Disabling, or *turning-off*, the internal clock to unused or idled functional SoC ASIC sub-blocks will decrease the amount of power consumed. For example, every piece of logic hardware (or gate) that is clocked will consumes some amount of power. By applying the appropriate amount of dynamic clock control or power management, the amount of power consumption can be reduced significantly for a specific mode of operation.

Dynamic power management

Dynamic power management requires some degree of up-front planning and organization. The SoC ASIC needs to be divided into the appropriate functional blocks to ensure that the maximum benefit can be achieved by disabling a specific piece of the hardware design.

The SoC ASIC will need to contain the logic necessary to control power up, power down, and reset of individual function blocks. This may include a clock tree register that enables or disables the clock to a specific functional block.

Each functional block can be powered down by setting the appropriate power down bit in this register that disables the clock to that block. Each functional block can also be initialized to a known state by setting the reset bit. Dynamic power management is an internal SoC ASIC function controlled by external software.

Idling embedded processors

Some SoC ASIC designs contain an embedded processor. Software is written for this processor to perform the necessary configuration and control operations. Most modern-day embedded processors contain an instruction that will place the processor into an idle, or *sleep*, state. Once the processor enters this state, only an external stimulus (such as defined interrupt) can wake-up the processor.

The processor will consume a very minimal amount of power while in the idle state. This low power consumption is a benefit for SoC ASIC designs assuming that no, or limited, software intervention is required for a particular function.

Once the SoC ASIC has been configured, the processor can idle itself and only be utilized during specific times (such as initialization or mode change). A complete up-front system design and hardware/software partitioning is required to reap the maximum benefits of processor idling.

SoC Challenges

Fig. 1 below represents a generic SOC block diagram that shows many of the challenges of SOC design and the limitations of conventional approaches to implementing SOCs. The device illustrated contains:

- One or more high-speed input datastreams (for example, network traffic for a router chip, video input from a sensor chip for a Camcorder, or cell data packets from a radio IC in a wireless device)

- A series of high-performance, algorithm-intensive compute engines that perform the heavy lifting of the computing load inside the chip. For the three examples listed above, the required processing would be packet processing and classification for the networking case, video scaling and compression in the camcorder, and filtering and channel coding in the wireless device. These elements are implemented in hardwired logic using traditional RTL design methodologies.

- A conventional rigid-ISA embedded CPU core that runs an operating system, performs general housekeeping tasks, and executes some of the lower-complexity algorithms. This CPU requires a closely-coupled hardware accelerator, which was also designed using RTL methodology, to accelerate the key inner loops of the algorithms that run on the CPU because the rigid-ISA limits the CPU’s performance to a level well below what’s required.

- A programmable co-processor, which is typically a conventional 16-bit DSP for signal processing functions. While perhaps not applicable in the networking example, the DSP might perform audio encoding in the camcorder or voice coding in the wireless device.

- A series of peripheral device controllers connected to the main CPU bus via a bus bridge.

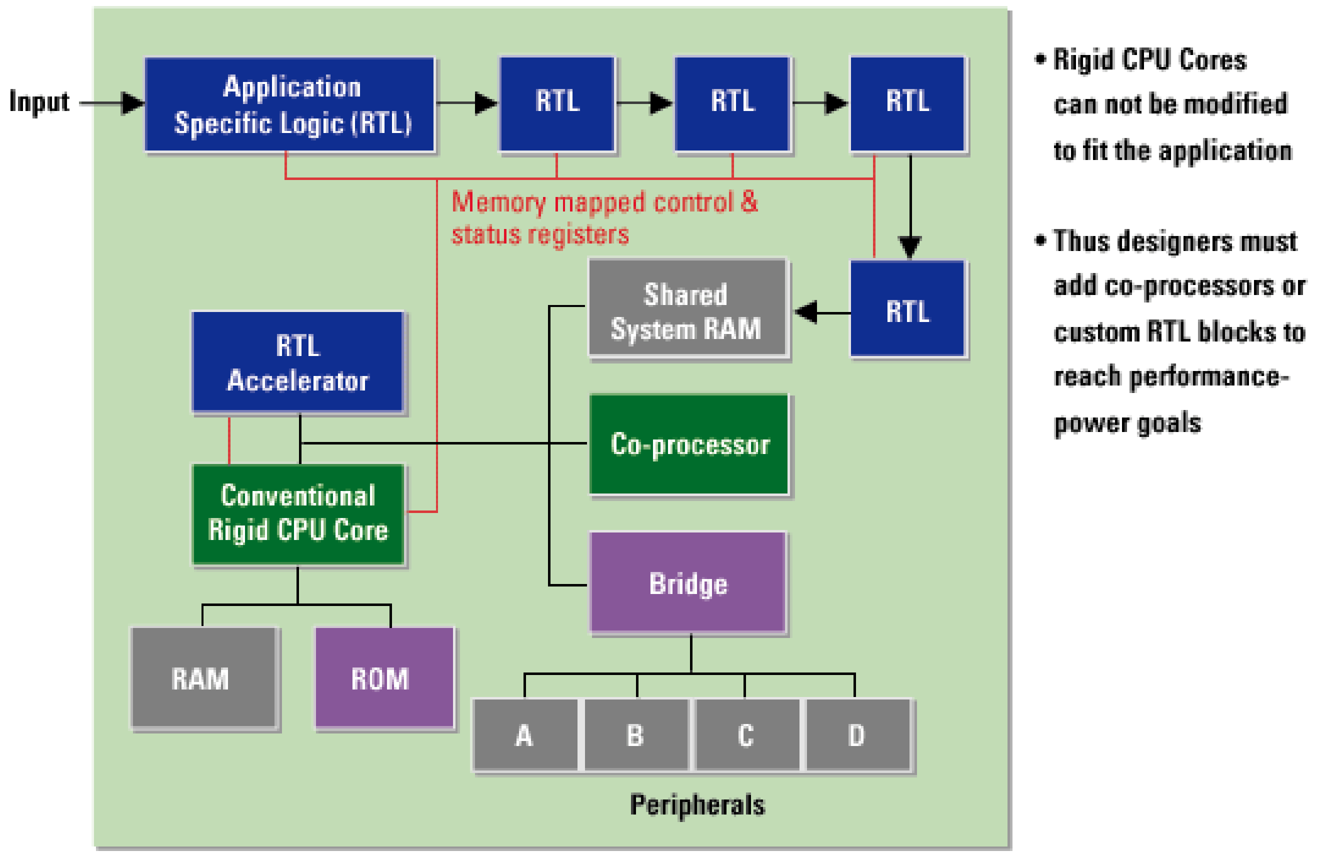

Input
Application Specific Logic (RTL)
RTL
RTL
RTL
Memory mapped control & status registers
RTL
Shared System RAM
RTL Accelerator
Co-processor
Conventional Rigid CPU Core
Bridge
RAM
ROM
A
B
C
D
Peripherals
• Rigid CPU Cores can not be modified to fit the application
• Thus designers must add co-processors or custom RTL blocks to reach performance-power goals

Fig. 1

Source: Tensillica

Hi Speed SoC Physical Design Challenges

When describing the complexity of VDSM systems on chip (SoCs), most engineers tend to refer to a combination of gate count, amount of embedded memory, and frequency of operation. If one's task is to assess the complexity of the physical design effort for a given SoC, then there are numerous additional factors that can create challenges far more significant than the sheer size or frequency of the design. Especially with high speed SoC ASIC designs!

The SoC physical design challenges are a direct result of the following design features:

• The increasing complexity of single-chip systems means that a design must be hierarchically partitioned into modules of a size that can be effectively managed by a designer and efficiently processed by the tools.

• New fine-line-width CMOS technologies can no longer be characterized by the simple physical models that were used in previous generations. Complex second-order effects (resistance, inductance, crosstalk, leakage, electromigration and the like) are not easily modeled above the physical layer.

• Higher-level tools run efficiently on small blocks but must use closely approximate models of any parameter affected by the final physical layout.

• Problems that are not identified until chip-level physical layout is complete lead to long verify-modify-redesign-reassemble-re-verify loops that consume enormous amounts of design time.

• Floorplanning is a crucial factor. The capability to efficiently provide the optimal floorplanning is essential to the SoC yield and performance. Early physical planning of big SoC designs is a pre-requisite. An early floorplan showing location of the high speed I/O, block and memory location quickly gives an idea of the feasibility of the physical design and goes one stage further than the RTL Analysis tools.

Reliability Issues

Many of the reliability issues are already addressed via tool automation and methodology changes. These include:

1. Metal antennae effects - where an electron charge can build up on long nets during manufacturing and blows up the transistor connected to it. Avoided by inserting diodes or adding metal jogs to the routing to force a layer change. The latter can cause many extra vias in the layout which has its own reliability issues if not carefully controlled.

2. Metal Slotting effects – this is where wide wires cause "metal dishing" effects due to processing limitations. Avoided by splitting wide wires.

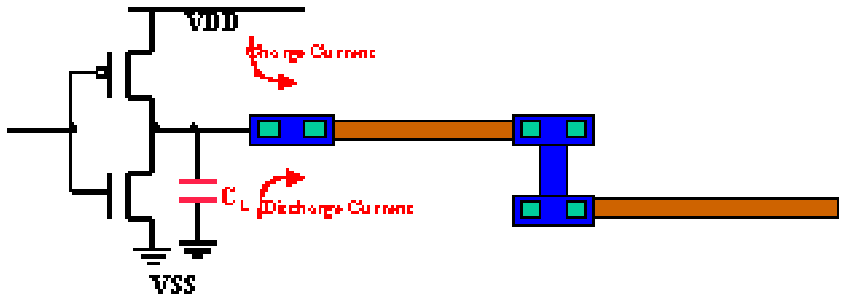
3. Simultaneously Switching Outputs (SSO) – where noise is injected into the power rails from many output changes at the same time and causes false signal values. Avoided by adding power/ground pads and by I/O isolation.

4. Soft Errors – Alpha particles, both naturally occurring and from lead in packaging, can cause state inversion of a flip-flop or memory element. With shrinking technology the charge induced becomes more significant. Avoided by hardened flip-flops, error correction built into the memories and by fault tolerant system architectures.

5. Memory yield – With memory taking an ever-larger proportion of the die, roughly 60% in the example above, overall good die per wafer will be lower than with pure logic. Avoided by adding redundant rows/columns and using Built-In Self Repair (BISR) with the larger embedded memories.

6. Electromigration (EM) is a key reliability effect that will worsen in 90nm. EM is caused by decreasing metal widths and increasing current density. When overstressed metal ions tend to migrate over time eventually causing the connection

to break. LSI Logic runs "lsisignalem" after placement to set routing rules to ensure that metal and via structures are robust enough to avoid the EM issues that can occur on signal nets. Post route checking is also performed to ensure that the avoidance was successful.



- charge and discharge current at the output of drivers might exceed the electromigration AC limit of a single via
- Some cells, depending on the load and the transition time, will need multiple vias at their output and on the driven net

Additional known issues that are successfully covered by recent EDA tools technology are time propagation delay, crosstalk, clocks skew, and voltage drop and database size. When looking at volume production requirements the need for lowest cost, smallest die, lowest power and fastest speed will always push SoC design teams to the leading edge of technology. Foundries are already running early 90nm silicon at an R&D level and early SPICE rules are already available.

Conclusion

Using advanced techniques and through the proper partitioning and classification of hardware and software requirements, optimal High Speed SoC ASICs can be designed and developed. Dynamic frequency and clock control, processor idling and functional grouping are common techniques to provide low power consumption for High Speed SoC ASIC designs. The physical design stage of SoC has become more complex due to the VDSM phenomenon, yet the new generation of EDA tools are providing a successful solution. While some may believe the industry is at its lowest stage for years there is already a wide variety of VDSM Hi Speed SoC infrastructures being put in place that will yield leading edge products within the next decade.

References

M.R. Stan, Wayne P. Burleson. Bus-Invert Coding for Low Power I/O. IEEE Transactions on Very Large Scale Integration Systems, 1995.

A. Malik, B. Moyer, D. Cermak. A Programmable Unified Cache Architecture for Embedded Applications. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, 2000.

D.H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. MICRO, 1999.

S.M. Kang. Accurate Simulation of Power Dissipation in VLSI Circuits. IEEE Journal of Solid-State Circuits, vol. CS21, no. 5, 1986.

G.Y Yacoub, W.H. Ku. An Accurate Simulation Technique for Short- Circuit Power Dissipation Based on Current Component Isolation. International Symposium on Circuits and Systems, 1989.

R. Tjarnstorm. Power Dissipation Estimate by Switch Level Simulation. International Symposium on Circuits and Systems, 1989.

T.H. Krodel. PowerPlay - Fast Dynamic Power Evaluation Based on Logic Simulation. International Conference on Computer Aided Design, 1991.

E. Macii, M. Pedram. High-Level Power Modeling, Evaluation, and Optimization. IEEE Transactions on Computer Aided Design, vol. 17, no. 11, 1998.

D. Marculescu, R. Marculescu, M. Pedram. Information Theoretic Measures for Power Analysis. IEEE Transactions on Computer Aided Design, vol. 15, no. 6, 1996.

M. Nemani, F. Najm. Toward a High Level Power Evaluation Capability. IEEE Transactions on Computer Aided Design, vol. 15, no. 6, 1996.

V. Tiwari, S. Malik, A. Wolfe. Power Analysis of Embedded Software: A First Step Toward Sofware Power Minimization. IEEE Transactions on Very Large Scale Integration Systems, vol. 2, no. 4, 1994.

C.T. Hsieh, M. Pedram, H. Mehta, F. Rastgar. Profile Driven Program Synthesis for Evaluation of System Power Dissipation. Design Automation Conference, 1997.

C. Barndolese, W. Fornaciari, F. Salice, D. Sciuto. Energy Evaluation for 32-bit Microprocessor. International Workshop on Hardware/Software Co-Design, 2000.

R.J. Evans, P.D. Franzon. Energy Consumption Modeling and Optimization for SRAMs, IEEE Journal of Solid-State Circuits, vol. 30, no. 5, 1995.

T. Givargis and F. Vahid. Interface Exploration for Reduced Power in Core-Based Systems, International Symposium on System Synthesis, 1998.

E. Boemo, “Some Notes on Power Management on FPGA-based Systems,” Lecture Notes in Computer Science, No. 975, pp. 149-157 (Berlin: Springer-Verlag 1995).

N. Sklavos, “Low-Power Implementation of an Encryption/Decryption System with Asynchronous Techniques,” VLSI Design, vol. 15, Issue 1, 2000, pp. 455-468.

R. Schmalbach, “Specifications for Sierra II ASIC,” 12016-0317, rev 5.1, February 2003.

The RTL Virtual Prototype, Gary Smith, Principal Analyst, Dataquest. April 22, 1996

Reuse Methodology Manual for SoC Designs, Second Edition, Michael Keating Synopsys, Inc - Pierre Bricaud, Mentor Graphics Corp, Kluwer Academic Publishers

IP RTL versus FPGA Optimized Netlist Functional Equivalence, Alexa Vignollet and Pierre Bricaud, Mentor Graphics Corp, IP Based Design 2000 Proceedings

Nuts and Bolts of Core and SoC Verification, Ken Albin, Motorola, Inc, Austin, TX

B. Ackland et al., “A Single Chip, 1.6-Billion, 16-b MAC/s Multiprocessor DSP,” *IEEE J. Solid-State Circuits*, Mar. 2000, pp. 412-424.
A. Agrawal, “Raw Computation,” *Scientific Am.*, Aug. 1999, pp. 60-63. January 2002

L. Benini and G. De Micheli, “System-Level Power Optimization: Techniques and Tools,” *ACM Trans. Design Automation of Electronic Systems*, Apr. 2000, pp. 115-192.

R. Hegde and N. Shanbhag, “Toward Achieving Energy Efficiency in Presence of Deep Submicron Noise,” *IEEE Trans. VLSI Systems*, Aug. 2000, pp. 379-391.

W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge Univ. Press, New York, 1998.

J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach,* IEEE CS Press, Los Alamitos, Calif., 1997.

P. Guerrier and A. Grenier, “A Generic Architecture for On-Chip Packet-Switched Interconnections,” *Proc. IEEE Design Automation and Test in Europe* (DATE 2000), IEEE Press, Piscataway, N.J., 2000, pp. 250-256.

R. Ho, K. Mai, and M. Horowitz, “The Future of Wires,” *Proc. IEEE*, Apr. 2001, pp. 490-504.

D. Sylvester and K. Keutzer, “A Global Wiring Paradigm for Deep Submicron Design,” *IEEE Trans. CAD/ICAS*, Feb. 2000, pp. 242-252.

Todd Moore and Rick Schmalbach Sep 1, 2003 - RFDesign

T. Theis, “The Future of Interconnection Technology,” *IBM J. Research and Development*, May 2000, pp. 379-390.

J. Walrand and P. Varaiya, *High-Performance Communication Networks*, Morgan Kaufmann, San Francisco, 2000.

H. Zhang et al., "A 1-V Heterogeneous Reconfigurable DSP IC for Wireless Baseband Digital Signal Processing," *IEEE J. Solid-State Circuits*, Nov. 2000, pp. 1697-1704.

Down to the Wire, Lavi Lev et al, Cadence
http://www.cadence.com/feature/pdf/4064_NanometerWP_fnlv2.pdf

Failures plague 130-nanometer IC processes, Ron Wilson, EETimes
http://www.eetimes.com/story/OEG20020826S0022