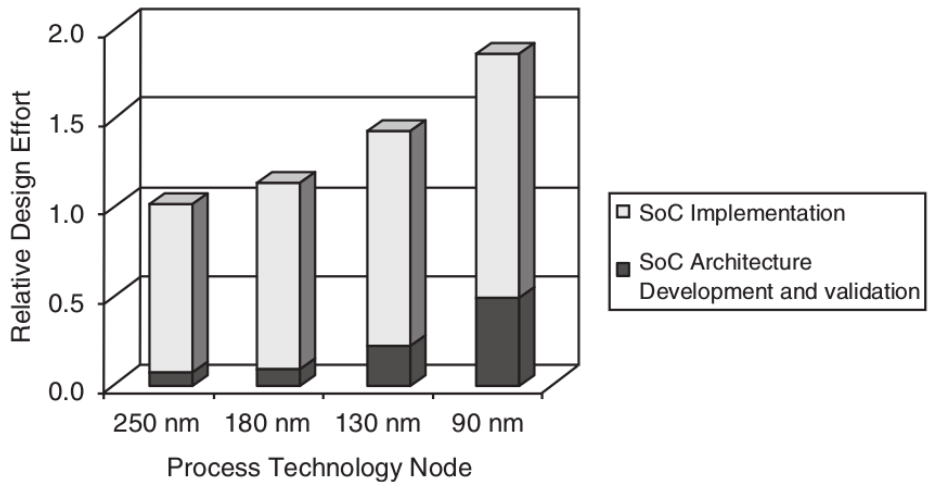
Motivation for ESL Design

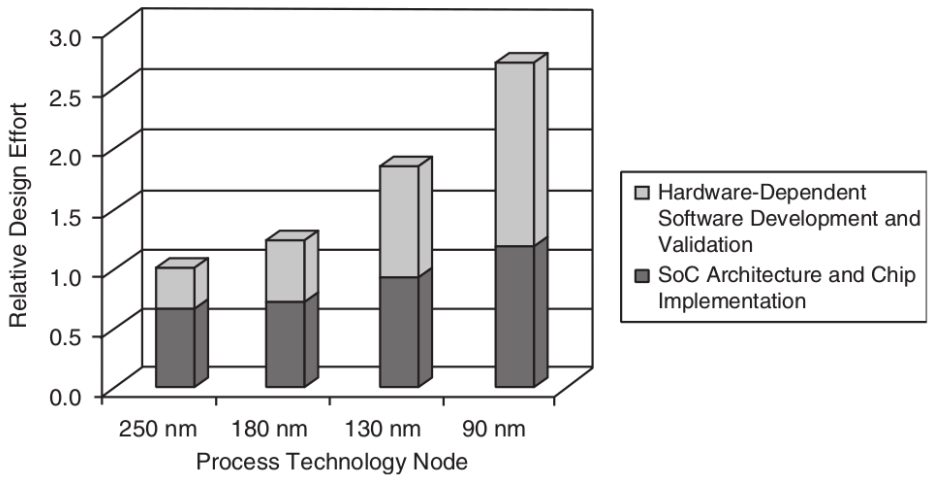
- The primary driver of ESL methodology adoption is the increasing failure of traditional methodologies to cope with the more complex system algorithm content necessitated by the integration of so much functionality.
- The implementation of algorithms to achieve the requisite QoS is subject to multiple constraints, among which are power consumption, time to market, and manufacturing cost.
- These constraints—especially in an SoC—often preclude the implementation of all algorithms purely in software.
- Simply deploying additional general-purpose processors or increasing the clock rate may well deliver the req- uisite software execution performance, but could violate the power consumption constraint.

Motivation for ESL Design

- Consequently, an embedded system—board or chip—generally deploys an optimized mix of both programmable and fixed-function hardware. A primary system design challenge is to devise such an optimized architecture.
- The optimization of a multiprocessor SoC implementation is even more complex. The software needs to split betwean more processors.
- Reports shows that the SoC architectural design effort has increased significantly.
- The trend toward multiprocessing has resulted in a significant increase in the effort expended on the development of hardware-dependent software such as Real-Time Operating System (RTOS) porting, firmware, and drivers.

Motivation for ESL Design





Traditional System Design Effectiveness

- Over 70% of designs missed pre-design performance expectations by at least 30%.
- Over 30% of designs missed pre-design functionality expectations by at least 50%.
- About 54% designs missed schedule, with an average delay of nearly 4 months.
- Of 45,000 design starts, nearly 6,000—about 13%—were canceled.

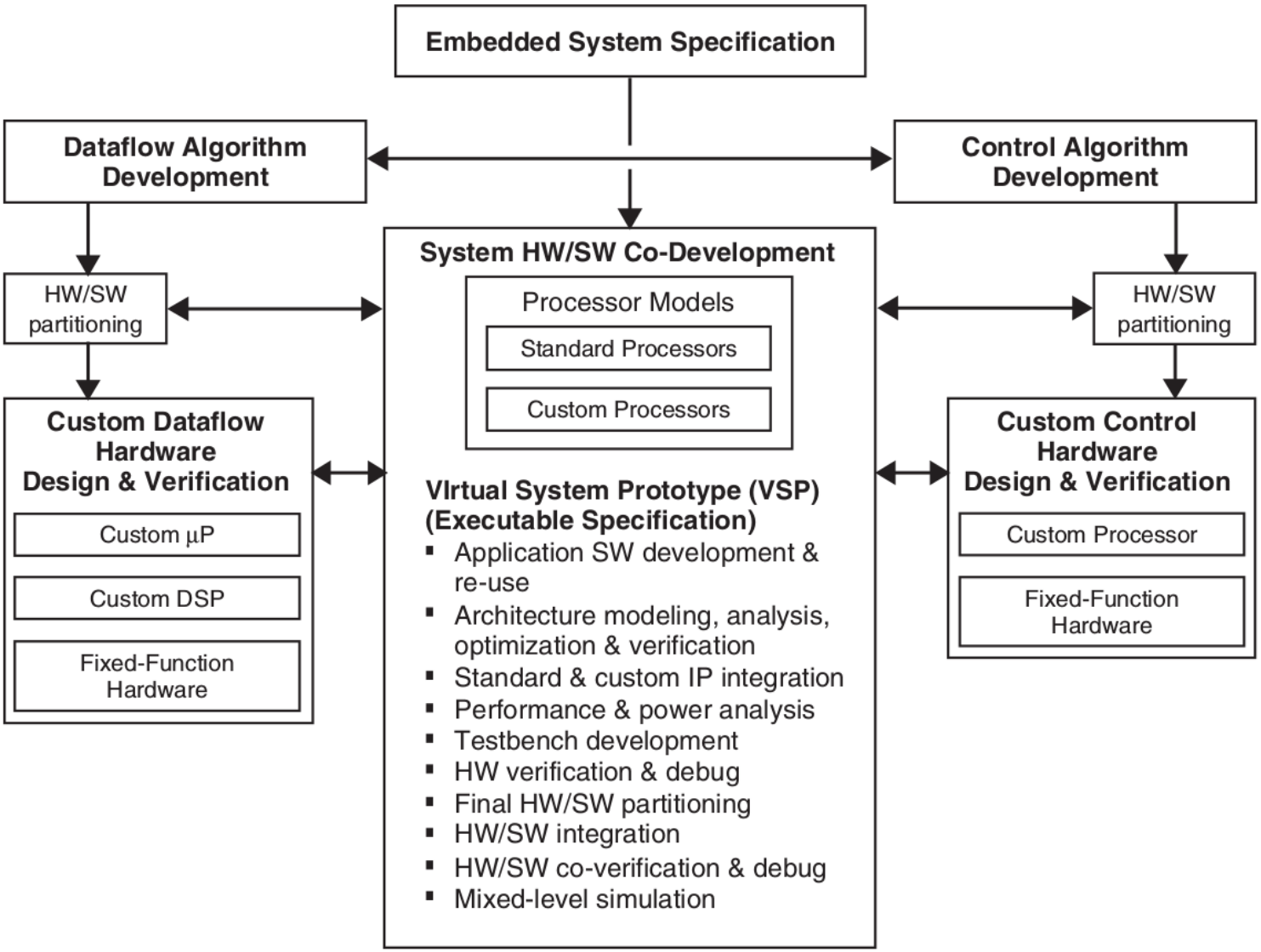
Traditional System Design Effectiveness

- The reasons given for these failures are:
  - Limited visibility into the complete system — 65%
  - Limited ability to trace — 54%
  - Limited ability to control execution — 42%
- Increasing the controllability and observability of board-level implementations using traditional methods clearly fails to produce the desired results. These approaches measure implementation-level characteristics such as individual signal levels and transitions. Similarly, in chip design, controllability and observability operate at the level of nanosecond-accurate logic transitions and bit-accurate data transfers. This level of abstraction obscures the system-level behavior with overwhelming implementation-level detail.

System Design with ESL Methodology

- System design proceeds at the abstraction level of processing, storage, and communications behavior. Design intent is realized in the form of processor instructions, function calls, memory accesses, and data packet transfers.
- Depending on the design task at hand, design timing accuracy varies from clock cycle-accurate through system event to no timing at all.
- these are the levels at which IP blocks are best evaluated, selected, and integrated. It is at these levels of abstraction that HW/SW partitioning and hardware architecture candidates are analyzed and optimized. These are the levels at which ESL design and verification is executed.

Methodology Map



Behavioral Modeling Methodology

- The centerpiece of an ESL design methodology is the system HW/SW codevelopment environment, the ultimate output of which is a system behavioral model, whichis called the Virtual System Prototype (VSP).
- Timed versions of the VSP are often described as "executable specifications." The timed VSP is a computational model that specifies system-level behavior, as independent of any particular implementation as possible, although any executable model contains implementation artifacts.

VSP: Potential Value

- Design and optimization of complex architecture requires the analysis of multiple candidate architectures and communication schemes, each with different performance, power, and cost tradeoffs.
- The proponents of VSP claim that it has the potential to fulfill this by enabling design teams to:
- 1. Complete complex system architectural design, analysis, optimization, and verification using an analytical approach that should eliminate the errors and omissions of hardware-based, hit-and-miss experimentation. Moreover, this design space exploration should consume considerably less time than that required by such experimentation.
- 2. Estimate system performance with reasonable accuracy before implementation decisions are made. For example, STMicroelectronics found that system-level estimates of transactions such as interrupt latencies and bus utilization were within 15% of RTL performance.

VSP: Potential Value

- 3. Perform analytical HW/SW partitioning, and optimum dynamic software partitioning, over multiple microprocessors, again with the aim of replacing hardware-based experimentation.
- 4. Commence application software development well in advance of hardware prototype availability, because a processing, storage, and communications model of the system is all that should be required. For example, STMicroelec- tronics began MPEG4 software development 6 months before the top-level netlist was available.
- 5. Achieve greater system determinism—that is, the ability to reproduce any given behavior in response to given system stimuli—by identifying undesirable nondeterminism in the specification.
- 6. Execute hardware verification several orders of magnitude faster than that at RTL. For example, STMicroelectronics used a cycle-accurate system level model of an MPEG4-SH device to encode and decode one Quarter Common Intermediate Format (QCIF) image in 2.5 seconds, compared with 3,600 seconds when executed with an RTL model—a factor of 1,440 times faster.

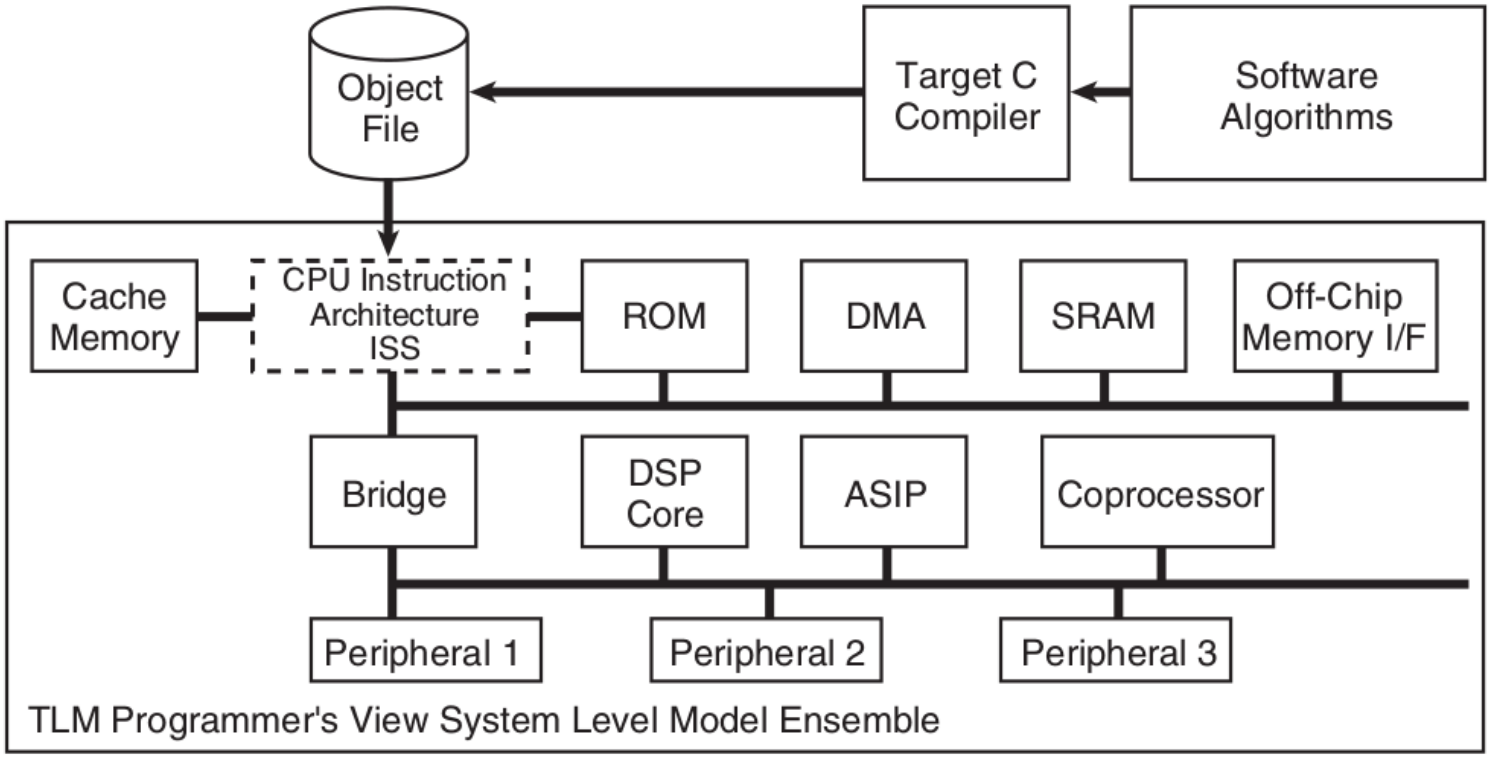
VPS: Potential Value

- 7. Execute HW/SW co-verification several orders of magnitude faster than that at RTL/C.
- 8. Deploy a system level testbed in which RTL implementations may be cosimulated with system models of other blocks to ensure correct "in-system" behavior.
- 9. Achieve greater performance improvements and power consumption savings than those achieved with traditional methodologies. This is because a full system-level design space exploration and analysis can reveal optimization alternatives that are not identifiable at RTL.
- 10. Undertake system model-to-RTL testbench development that ensures compliance of the RTL implementation with design intent.

VSP: Programmer's View

- The VSP is essentially a network model of the system's processing, storage, and communications resources. Indeed, at its uppermost levels of abstraction—the Programmers View (PV)—it leverages the Transaction-Level Modeling (TLM) and analysis techniques that have been used by network architects for decades.
- The PV model, as its name implies, is intended for use as a software development platform. It can be used in event-driven HW/SW co-simulation.

VSP: Programmers's View



- The PV model possesses the system visibility required for software development and debug, such as registers and interrupts, as well as links to the ISS and the rest of the software development and debug environment.

VSP: Programmer's View Plus Timing

- At an intermediate level of abstraction, the PV plus Timing (PV + T) is a timing-approximate version of the same behavioral model of the system.
- Its proponents claim that its ease of modification enables the rapid analysis of multiple "what-if " scenarios to develop a performance and power optimized architecture.
- The model enables the early validation of both application and hardware-dependent software, well in advance of RTL or hardware prototype availability.
- It also supports mixed-level simulation, enabling in-system verification of RTL blocks as they become available. It is thus both an executable specification and a methodology for reuse and derivative design.

VSP: Cycle-Accurate View

- At its lowest level of abstraction, the cycle-accurate view, the model is an executable specification, predictive of real chip timing.
- This model is used for final architectural verification, final HW/SW partitioning, HW/SW coverification, and system model-to-RTL testbench development.
- This model also supports mixed-level simulation, enabling in-system verification of RTL implementation blocks as they become available.
- A cycle-accurate system model can also be used for early partitioning and system verification, assuming that no faster model is available.

Behavioral Modeling Environments - Commercial Tools

- Today there are several tools available from Cadence, Synopsys and Mentor.
- The Trailblazer: Virtual Component Codesign (VCC)

Behavioral Modeling - Open-Source and Academic Technology

- The POLIS system is a HW/SW co-design environment developed at the Center for Electronic Systems Design of U.C. Berkeley.
- The Ptolemy project was commenced by U.C. Berkeley.
- The SpecC language and technology were developed by U.C. Irvine.
- Open SystemC Initiative (OSCI) SystemC Reference Simulator.

Historical Barriers to Adoption of Behavioral Modeling

- System behavioral modeling is over 20 old, and yet it is in only the last 6 or 7 years that this methodology has proliferated. The reasons lie in:
  - Demand side issues
  - The standards barrier
  - The lack of automated links to chip implementation

The Demand Side Issues

- On the demand side, the relative simplicity of the architecture and communication protocols of single-processor systems—and the majority of SoCs are still single-processor designs—did not require behavioral modeling.
- Such an SoC can be developed with an RTL integration platform methodology, whereby some degree of performance and power optimization is undertaken—within the architectural limitations of the platform—at the implementation level.
- The integration platform enables the identification of modest optimizations that more-or-less work, but the problem is that its architectural inflexibility generally hinders the significant optimizations that work best.
- In software-intensive designs with multiple interacting processors, non-simplistic cache memory architecture, and complex communication protocols, the integration platform methodology generally "breaks." At this point, behavioral modeling becomes essential.

Teh Standards Barrier

- However, even in those advanced designs that required behavioral modeling, there was another barrier: the perennial standards barrier.
- There was no standard system-level description language and no standard methodology for transaction level modeling—or anything else.
- It was impossible to construct a system-level IP model library that could be easily reused in any environment other than that in which it was originally constructed.
- There could be no means to develop the automatic system-level model generation tools that are essential to the reuse of legacy RTL IP. According to various RTL design engineers, manual system-level coding of an RTL functional block takes about 10% to 20% of the effort necessary for the original RTL design. Without automated generation of system-level models of legacy RTL IP, ESL design would never become a mainstream methodology.

Standards

- Open SystemC Initiative
- Open Core Protocol International Partnership
- SpecC Technology Open Consortium
- IP-XACT
- The System-Level Language War

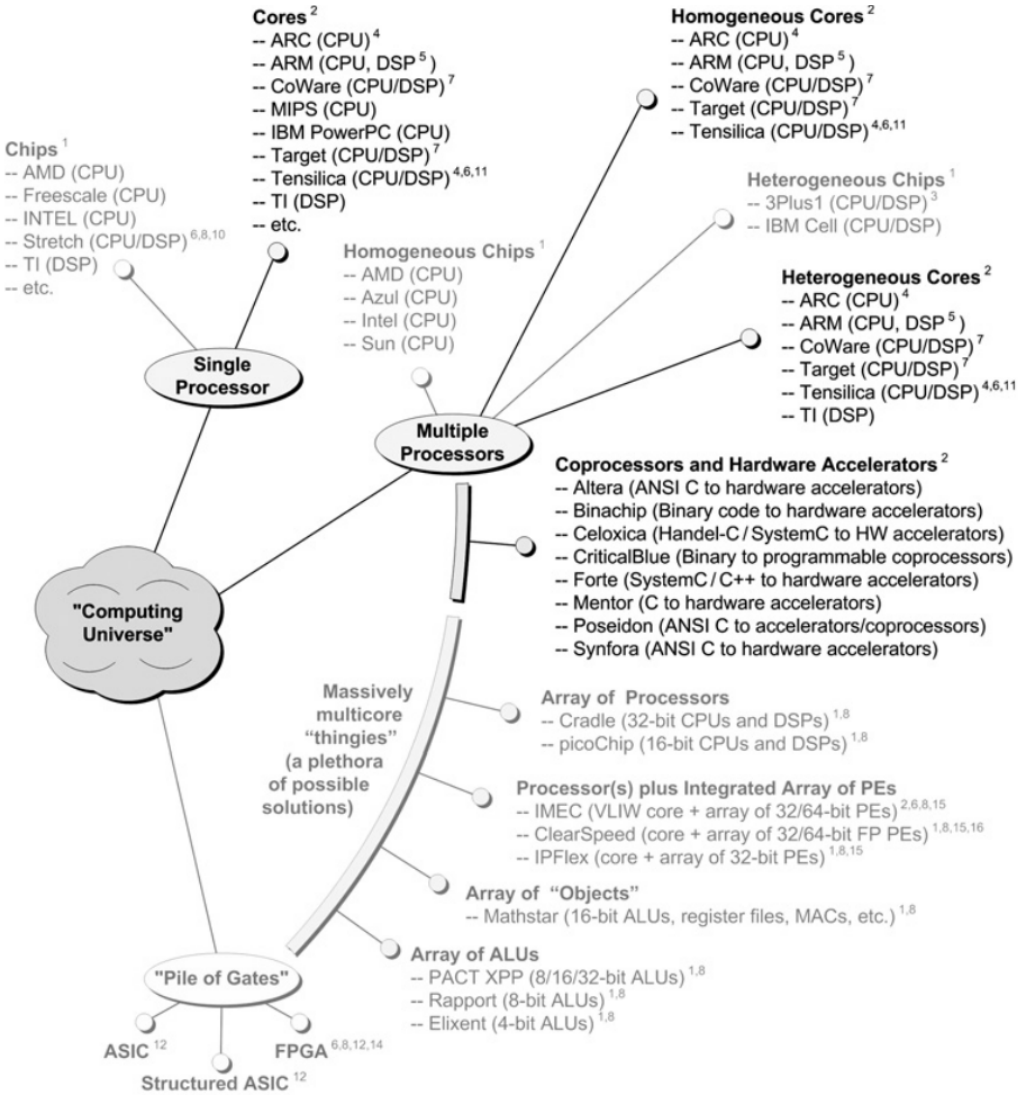
Automated Links to Chip Implementation

- The most serious genetic defect of behavioral modeling, one that has plagued its adoption for chip design since its very genesis, has been the "missing link" to hardware implementation.
- A great deal of hardware implementation is available in the form of pre-designed IP, such as microprocessor and DSP cores. Indeed, SoC development is as much an IP assembly task as it is a design task. However, there is often a need for custom design—either fixed-function hardware or custom programmable processors.
- Historically, this has been a manual RTL design process and, for more than a decade, ESL development signally failed to fulfill the promise of the higher implementation design productivity that should have resulted from working at a higher level of abstraction.
- With system level coding adding up to 20% to the effort of designing a functional block, but no automated means to design the RTL, ESL was seen by many RTL engineers as having a negative value.

Automated Links to Chip Implementation

- If ESL development is to enjoy widespread proliferation beyond the system architects and algorithm development engineers, RTL design from high-level models must be automated.
- The number of tools and methodologies are now available. Using ESL behavioral synthesis tools or configurable IP, the design team can create fixed-function hardware from a high-level algorithmic description.
- EDA tools and configurable IP can be used to develop Application-Specific Instruction set Processors (ASIPs) and coprocessors that accelerate software execution by deploying greater instruction-level parallelism and parallel processing capability than general-purpose processors can provide.

Computing Universe



Automated Implementation of Fixed-Function Hardware

- Commercial Tools
  - Mathematical Algorithm Development Tools
    - Mathworks HDL Coder
  - Graphical Algorithm Development Tools
    - Comdisco's Signal Processing Workbench (SPW)
    - Matlab HDL Coder
  - Latest Generation High-Level Synthesis Tools
    - Vivado HLS
    - Cadence Stratus
    - Mentors Catapult
    - Synopsys Synphony
- Open-Source and Academic Tools
  - SPARK Parallelizing High-Level Synthesis (PHLS)

Automated Implementation of Programmable Hardware

- An ESL behavioral modeling methodology can analytically determine an optimal system architecture, whereas automated links to fixed-function hardware implementation can increase design productivity.
- However, as previously noted, the motivation for adopting ESL design methodologies is the increase in system algorithm content, much of which is implemented in software.
- Software is easier to modify and reuse than hardware, and the post-implementation correction of errors is far easier also. Consequently, the system design team's increasingly natural predilection is to implement whatever it can in software, and whatever it must in fixed-function hardware—just as embedded system designers do.
- The consequent significant increase in software functionality challenges the ability of a single—often general-purpose—processor to execute it with the requisite performance.

Automated Implementation of Programmable Hardware

- This bottleneck is forcing the migration to multiprocessor systems, whose complex architecture requires the deployment of ESL design methodologies.
- A multiprocessor system consisting of general-purpose processors is then simply an array of bottlenecks operating in parallel.
- The general-purpose microprocessor is designed to execute both control and data processing functions, and is an increasingly inadequate compromise in the execution of compute-intensive software.
  - Inadequate instruction-level parallelism, which limits data throughput per cycle.
  - Inadequate parallel processing resources, which limits total throughput.
  - A general-purpose Instruction Set Architecture (ISA), which may be an inadequate match for the application tasks to be undertaken.

Automated Implementation of Programmable Hardware

- Multiprocessing seeks to achieve the requisite parallelism—but at a high cost in microprocessor components or IP, and a considerable effort in software development.
- Compute-intensive software developed for single-processor operation must be repartitioned over multiple processors, introducing synchronization problems that can significantly increase system latency.
- The complex communications between multiple processors and cache memories dramatically increases the functional verification cost and can increase system latency further, to an extent that is heavily dependent on the selected communications architecture.
- The introduction of the general-purpose DSP was a major first step in addressing the parallelism problem.

Automated Implementation of Programmable Hardware

- DSPs have been deployed extensively in mobile communications and multimedia equipment. However, as the performance requirements and power consumption constraints of mobile equipment increase, there will be an increasing requirement for microprocessors and DSPs optimized for performance and power consumption in specific applications.
- Using EDA processor design tools or IP-based approaches, the design team can create these application-specific or application-optimized components. Such components provide the instruction flexibility, instruction-level parallelism, and parallel processing resources that are necessary to execute compute-intensive application software.

Mainstreaming ESL Methodology

- SystemC models possess concurrency, bit-accuracy, and clocking attributes that C/C ++ models do not. Use of these attributes results in a system-level model that more closely resembles the chip architecture than do C/C ++ models—and is useful to the RTL designer.
- SystemC concurrency eliminates the necessity for the RTL designer to interpret the appropriate concurrency from a C model, an interpretation that may not be optimum or correct. SystemC thus captures the system architect's design intent in a manner that is unambiguous to the RTL designer.
- There is therefore a strong motivation for the system architect to adopt SystemC provided the RTL design team agrees.

Acceptance by RTL Teams

- Tools should provide "links to implementation" for programmable and fixed-function hardware. However, hardware engineers must also integrate the whole system-level block and communications ensemble at the RT level and verify it.
- Hence, ESL design methodology must support the "standard" RTL design and verification flow. ESL design acceptance by the RTL design team will be largely conditional upon the establishment of that support.
- Fixed-function hardware design has been—and continues to be—a significant bottleneck in chip design. Logic synthesis automates the implementation from RTL inputs, leaving the design of synthesizable RTL as a manual task.

Acceptance by RTL Teams

- To the extent that ESL behavioral synthesis and configurable IP approaches automate the generation of synthesizable RTL, it will assist ESL acceptance by RTL engineers.
- ESL behavioral synthesis may well evolve into a front-end for logic synthesis—as originally intended by the Behavioral Compiler visionaries—especially for those functions that require high levels of parallelism.
- Also, links to RTL verification are essential. The ESL verification environment is not an RTL verification environment. It is a self-checking, stimulus generation, and coverage measurement machine that verifies the overall behavior of the system model and its constituent models, such as bus functional models, bus monitors, memory models, and IP models.

Acceptance by RTL Teams

- This ESL verification environment provides a framework within which the RTL verification engineer may develop the RTL verification environment, complete with microlevel constraints, monitors, and assertions.
- Clearly, what is required is automatic generation of the system-to-RTL verification environment with self-directed stimulus generation and higher functional coverage than is currently achieved.

Conclusion

- The increasing implementation of algorithms—both control and dataflow—in software rather than hardware drives chip design towards a processor-centric methodology. Chip design is now system design. The ESL tools must support this HW/SW co-design approach.
- The evolution of ESL design demonstrates that technology does not drive adoption. ESL technologies will be adopted when designers have no choice, because of design complexity and the necessity for productivity gains.
- Standards for design input and model creation are essential for wider ESL adoption. Designers have been plagued by proprietary input and modeling languages. SystemC is a good start, but widespread ESL adoption needs further standards.

Conclusion

- Successful use of ESL by designers does not depend on tools from the largest EDA companies. The current ESL ecosystem is an ad hoc collection of various techniques and technologies from both academic research and many start-ups, as well as from the IP industry.
- Some areas of need, such as high-level synthesis tools and automated processor design solutions, will continue to require a high level of commercial investment. However, these may be most successfully delivered as part of the offerings of IP, semiconductor, and system platform providers, rather than by stand-alone ESL tools suppliers.