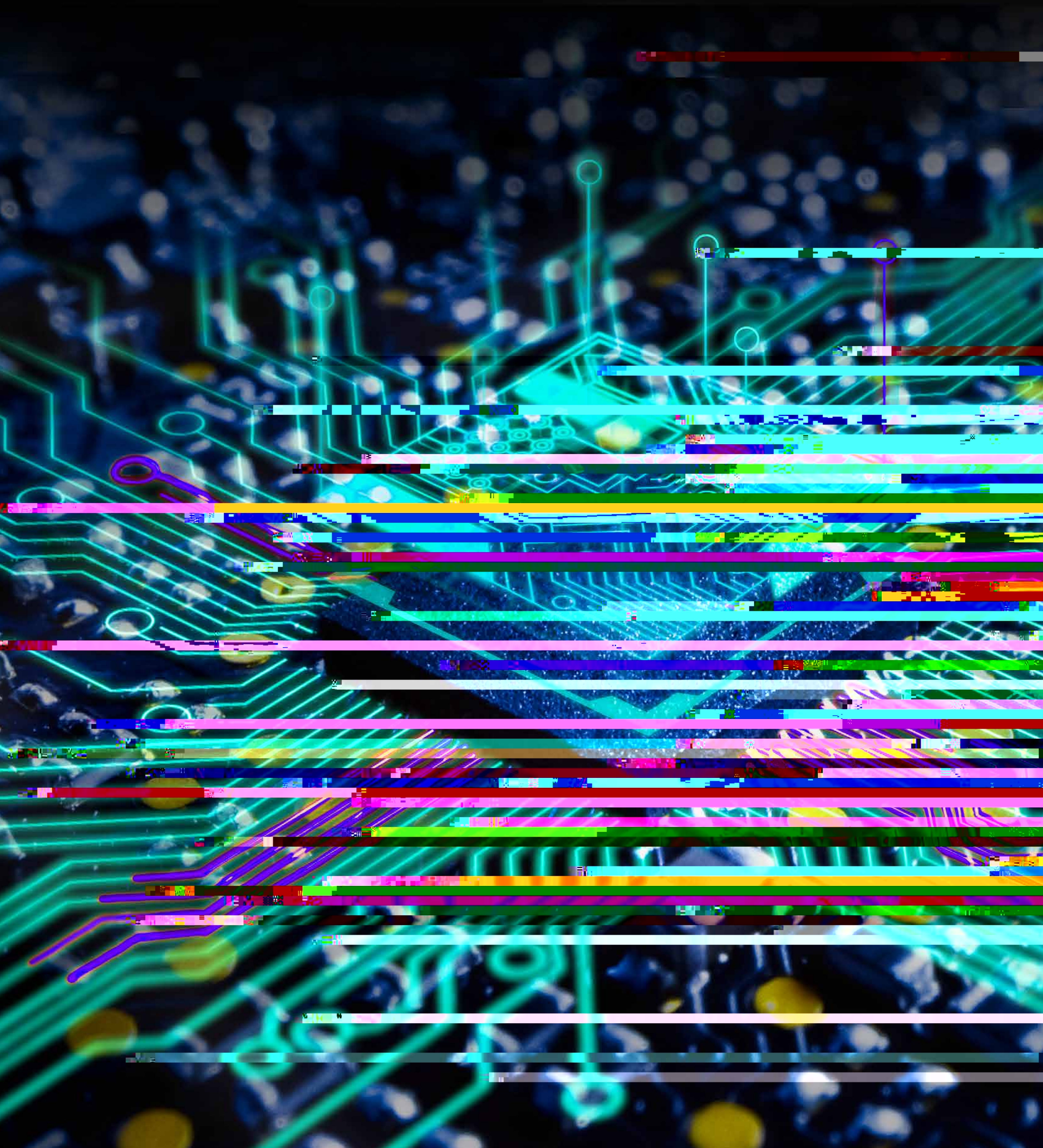
S OPS



Enabling the Design of Multicore SoCs with Application-Specific Instruction Set Processors

Offloading performance or power-critical functions from a merchant processor into specialized accelerators is commonplace in today's SoC designs. Such accelerators are specialized to the application in order to deliver the needed performance in the lowest power envelope, and they can be activated only when needed. These accelerators are implemented as application-specific instruction-set processors (ASIPs) or as fixed-function hardware.

The design of ASIPs comes with the need to define the best suited processor architecture, and to develop both the hardware implementation and the associated software development kit (SDK). While the value of an ASIP is well-understood, SoC design teams have often gone back to standard processors because they were unable to complete an ASIP design on time and within budget.

Moving functions into fixed-function hardware accelerators comes with a heavy cost, as well: loss of programmability, and therefore loss of flexibility after manufacture. This is intolerable with advanced process technology nodes, where high mask costs necessitate the reuse of silicon in multiple products and product generations, to expand the revenue lifetime of an SoC.

Fortunately, today's SoC designers can rely on tools such as Synopsys' ASIP Designer to build multicore SoCs with ASIPs. Such designs can be specialized to the application in order to meet the performance and power requirements, but still retain software programmability.

Automating Application-Specific Instruction-Set Processor Design—ASIP Designer

ASIPs rely on similar techniques as used in the design of hardware accelerators to reach high performance and low power: heavy use of parallelism and specialized datapath elements. Yet ASIPs

- ASIP Designer comes with a large set of example models, written in nML. These include controllers, SIMD and VLIW architectures, as well as application-specific accelerators. All models are provided in source code and serve both as a reference, as well as a starting point for your own ASIP.
- Unique compiler-in-the-loop technology, enabled by the automatic generation of a comprehensive software development kit (SDK) for each ASIP modeled in nML, containing the following components:
  - An optimizing compiler, recognized for its efficient code generation and quick and automatic retargetability to new ASIP architectures. The compiler supports C, optionally extended with user-defined data types and operators, C++, and OpenCL C (OpenCL kernel language).
  - A fast instruction-set simulator, offering both cycle-accurate and instruction-accurate abstraction levels and easy integration into signal-level and transaction-level (TLM2) virtual prototypes
  - A flexible (multicore) debugger that can be used in connection to both instruction-set simulators and on-chip debug hardware (via JTAG)
  - Multi-faceted profiling capabilities to analyze the instruction-set architecture for hot-spots and to drive the architectural optimization process
- Automatic generation of a power- and area-efficient hardware implementation of each ASIP, in synthesizable Verilog or VHDL. This allows to analyze implementation-related metrics such as power consumption, area efficiency, and timing closure at any stage of the design process, enabling a synthesis-in-the-loop approach.
- Multi-faceted verification capabilities, including a wide range of formal checks, the automatic generelated

Silicon to S