ASIP Designer: Design Tool for Application-Specific Instruction-Set Processors

Highlights

- Accelerates the design, programming and verification of ASIPs
- Based on a single description of the ASIP in nML
- Enables rapid architectural exploration, based on unique compiler-in-the-loop technology
- Patented automatic software development kit (SDK) creation including:
  - Retargetable C/C++ compiler
  - Retargetable instruction set simulator, both cycle- and instruction-accurate
  - Retargetable linker, assembler and disassembler
- Automatic register transfer-level (RTL) hardware generation
- Automatic generation of synthesis scripts supporting the Synopsys reference methodology

ASIP Designer is a tool suite for the design, verification and programming of application-specific instruction-set processors (ASIPs).

ASIPs form the basis of many modern multicore SoCs, which have to integrate dozens of complex system functions, each requiring its own optimal balance of performance, flexibility, energy consumption, communication, and design time. The traditional model of a general-purpose processor core with a number of fixed hardware accelerators is no longer sufficient to meet the demands of today's applications. ASIPs established themselves as a third implementation option for designs in which the power, performance and area efficiency of a standard processor IP is not sufficient, and fixed-function hardware accelerators are not flexible enough.

ASIP Designer supports all aspects of ASIP-based design, including architectural exploration and profiling, hardware generation, and verification. In addition, it generates a software development kit (SDK) featuring highly optimizing C and C++ compilation, instruction-set simulation, and debugging technology.

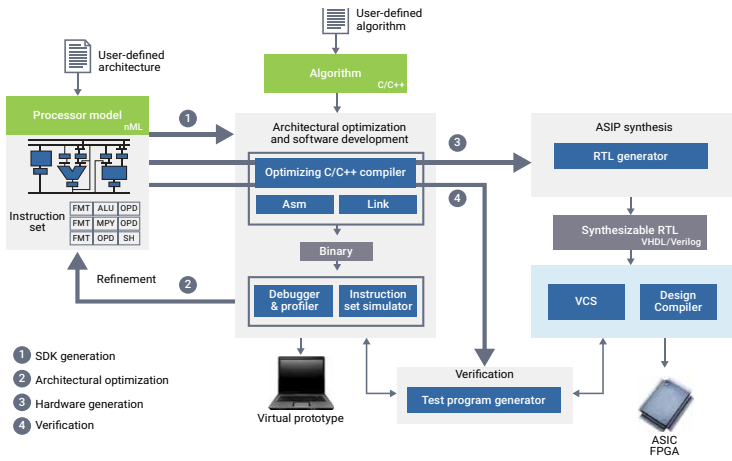

Figure 1: ASIP Designer Tool Flow

ASIP Designer supports a broad range of architectures, including small microprocessors, DSP-dominated cores, VLIW and vector processors, as well as programmable data-paths. The ASIP's architecture is modeled in nML, a high-level language at the abstraction level of a programmer's manual of a processor. The nML description captures processor resources, the instruction set, the instruction pipeline, and the bit-accurate behavior of the processor's primitive operations and I/O interfaces. All ASIP Designer tools use the same nML model, guaranteeing full consistency between the hardware implementation and the SDK, and between different simulation abstraction levels in the SDK.

Retargetable C/C++ Compiler

The compiler offers the following features:

- A unique approach to automatically adapt (retarget) the compiler to the processor architecture, based on the nML processor modeling language
- Architectural exploration using Compiler-In-The-Loop technology. Users can describe alternative processor architectures in nML and compare their performance by compiling benchmark C/C++ programs onto each architecture and evaluating the results
- Support for a wide range of processor architectures, from general-purpose processors to highly specialized ASIPs
- Support for the following programming languages:
  - C, optionally extended with user-defined data types and operators using C++ classes, member functions and function overloading
  - C++ (leveraging LLVM compiler front-end technology extended for user-defined data-types, native pointers, multiple address spaces, addressable unit sizes wider than byte)
  - OpenCL C (OpenCL kernel language)
- Efficient compiler optimizations, including:
  - High-level code optimizations, including alias analysis for effective software pipelining and exploitation of various addressing schemes
  - Code selection, exploiting the use of specialized instruction patterns (not restricted to tree patterns)
  - Register allocation, supporting distributed register files where instruction-level parallelism depends on the register choice. Separate register allocation and register assignment passes for effective interaction with scheduling
  - Efficient implementation of subroutines, including inter-procedural context-save optimization, multiple register sets for fast context switching, and reverse in-lining to reduce code size
  - Scheduling with software pipelining of loops, including exploitation of negative dependency lengths (aggressive scheduling) to deal with long latencies in deep pipelines
  - Support of advanced control-flow constructs in C programs for vector processors, using per-lane predication and vector predicate stacks
  - Whole-function vectorization and barrier synchronization of OpenCL C programs
  - Support of intrinsic function calls and of in-line assembly code
- Light-weight C/C++ library stack (libc++lite) tuned to embedded applications, offering maximum functionality while avoiding code bloat
- Generation of binary machine code in the Elf object-file format, including source-level debug information in Dwarf sections
- Integrated in ASIP Designer's graphical development environment (IDE), including an option for integration in Eclipse

Key Architectural Features Supported by the Retargetable Compiler	
Arithmetic	• General-purpose as well as application-specific arithmetic units
Data types	• General-purpose as well as application-specific data types (e.g. fractional, custom floating point, complex, and vector data-types)
Pipeline	• From shallow to deep instruction pipelining • Exposed or protected pipeline • Multi-cycle and multi-word instructions, delay slots • Resolution of pipeline hazards by the compiler
Instruction format	• From orthogonal to highly encoded instruction formats • Support of variable-length instructions and instruction compaction
Memory architecture	• Support of multiple memories and multiple memory ports • Large variety of addressing modes, including: indexed, direct and indirect addressing, with post-modification, circular buffering, etc. • Up to 64-bit address space support • Both little-endian and big-endian supported
Register architecture	• From general-purpose register-files to special-purpose registers • Support of coupled operand and/or result registers

Key Architectural Features Supported by the Retargetable Compiler	
Control flow	• Subroutine and interrupt support, with or without a software stack • Support of hardware loop instructions, residual control using mode registers, and predicated execution (including per-lane predication for vector processors)
Parallelism	• Support of multi-threaded processors • Instruction level parallelism (e.g. VLIW) and data-level parallelism (e.g. SIMD), including combinations of both

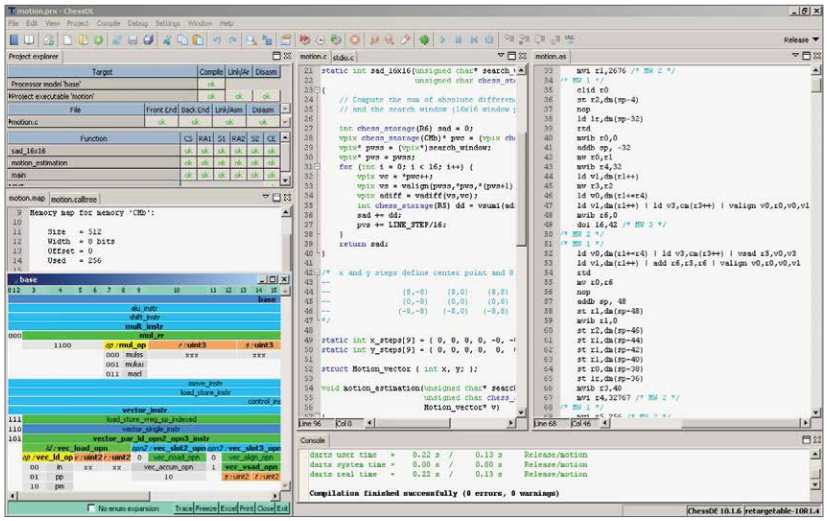

Figure 2: Development perspective in ASIP Designer's IDE, showing compilation of an MPEG4 motion estimation function on an ASIP

Retargetable Instruction-Set Simulator

- A unique approach to instruction-set simulator (ISS) retargetability, based on the nML processor modeling language
- Fast cycle-accurate simulation using just-in-time compilation techniques, monitoring the full instruction pipeline
- Fast instruction-accurate simulation using just-in-time compilation techniques
- Cycle- and instruction-accurate simulation models are both generated from the same nML description, eliminating the need to keep two models in sync
- Loading of Elf executable files, optionally containing source-level debug information in Dwarf format
- Integrated in ASIP Designer's graphical development environment (IDE), including an option for integration into Eclipse. This IDE can also connect to the processor hardware for on-chip debugging (e.g. via a JTAG port, with built-in support for a wide variety of 3rd-party debug cables)
- Source-level debugging, showing correspondence between executed instructions and source-code statements, and between register or memory locations and source-code variables
- Support of breakpoints on instructions and on source-code statements, and of watch points on register and memory locations
- Profiling of instructions, storages, functional units, pipeline hazards
- Application programming interface to 3rd-party simulators and integrated development environments, for co-simulation of the ASIP and its environment
- SystemC TLM2 interface generation allows for pre-silicon software development using virtual prototypes, such as those created with Synopsys Virtualizer
- Multi-ASIP simulation and on-chip debugging, supporting breakpoint export and synchronous stepping
- Native simulation: bit-accurate execution of C/C++ application programs written for the target ASIP architecture, applying the target architectures' data types and operators while executing on a 32-bit host workstation

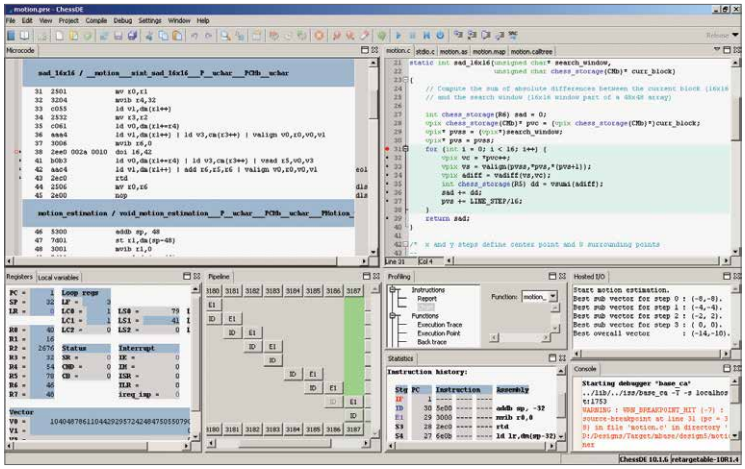

Figure 3: Debug perspective in ASIP Designer's IDE, showing instruction-set simulation of an MPEG4 motion estimation function on an ASIP

Retargetable Back-End Tools

- A retargetable linker to build executable files from separately compiled Elf/ Dwarf object files for different source files or functions
- A retargetable assembler and disassembler, to translate machine code from assembly into binary format and back. The assembly language syntax is user-definable and is specified as part of the processor's nML model

RTL Generator

ASIP Designer includes a retargetable RTL hardware generator. Once an ASIP has been optimized using the retargetable C/C++ compiler and instruction-set simulator, the RTL hardware generator provides a quick and efficient route to hardware for the new ASIP, including the following features:

- Automatic translation of the processor's nML description into synthesizable VHDL or Verilog code
- Supports a structural design style, using synchronous logic
- The generated RTL description can be synthesized efficiently with standard, commercially available ASIC or FPGA synthesis tools
- Automatically creates synthesis scripts for Design Compiler Graphical, including support for the physical guidance approach for IC Compiler that tightens the correlation of timing, area and power to enable significant reduction in routing congestion
- Generates files and scripts for ProtoCompiler, enabling a rapid path towards HAPS® FPGA-based prototyping systems, including on-chip software debugging support
- Existing hardware blocks can be integrated in the RTL design
- Automatic generation of on-chip debugging logic (e.g. using JTAG), interoperable with industry-standard third-party debugging solutions
- Debugging support for ASIPs integrated in an ARM® CoreSight™ system
- Many configuration parameters can be defined by the user, to influence the RTL style
- Supports low-power design optimizations, such as selective clock gating per register and operand isolation

ASIP Verification

ASIP Designer offers extensive support for verification of ASIP designs, including the following:

- Reduced verification effort resulting from the automatic consistency between generated RTL and ISS implementations (due to single-source ASIP description in nML)
- Automatic support of validation against known good targets (host workstation, standard compiler) provided by automatic

support of native execution (bit-accurate execution of C/C++ application programs written for the target ASIP architecture on a 32-bit host workstation)

- Automatic test creation for analyzing and diagnosing an ASIP's ability to support C/C++ compiler generation
- Formal analysis of processor properties such as resource conflicts, pipeline hazards and unique instruction encodings
- Regression suite for basic C/C++ code compliance testing, including methods to extend the test suite for ASIP-specific testcases
- Regression testing automation
- SystemVerilog class generation based on a processor's nML model, to be used in SystemVerilog programs to generate random instruction sequences, supporting the UVM verification methodology
- Constrained-random test program generation with ISS to RTL comparison and customizable verification coverage analysis

Example ASIP Models

Designers can choose from an extensive library of example ASIP models provided as nML source code. In combination with ASIP Designer, these models can be used as a starting point for architectural exploration, and customer-specific production designs.

Microntrollers	
Tmicro, Tnano	Compact 16-bit RISC microcontrollers
DLX (family)	Variants of Hennessy and Patterson's 32-bit RISC microcontroller with 5-stage protected pipeline—Additional family members implement hardware floating-point units, narrow SIMD andvarious forms of multi-threading
Tmcu	32-bit microcontroller with 16/32-bit variable-length instructions
Tzscale	32-bit microcontroller featuring the RISC-V instruction-set architecture
Trv (family)	Microcontrollers featuring the RISC-V instruction set architecture. Variants include versions with a 32-bit and 64-bit wide data path, with 3-stage and 5-stage pipeline version, and with custom extensions supporting zero-overhead hardware loops and Load/stores with post-modify address modes
Generic DSPs and Parallel Processors	
Tdsp	16/32-bit DSP with single multiply/accumulate unit, dual load/store units with indirect addressing and address post-modification, and 3-way instruction-level parallelism in 16/32-bit variablelength instructions
Tvec (family)	Variants of a wide SIMD processor, with per-lane predication controlled by either predicate registers or a predicate stack, and gather/scatter-based vector addressing—Additional family member supports compilation of OpenCL C kernels
Tvliw (family)	Variants of a 4-slot VLIW processor, with predication of VLIW slots and instruction compaction
Domain-specific Processors	
Tmotion	ASIP for acceleration of motion estimation kernels in video coding, using custom data-path elements, SIMD and instruction-level parallelism
Tcom8	IP for acceleration of communication kernels, with 8-lane SIMD and complex-number hardware
FFTcore	ASIP for scalar implementation of complex FFT
MXcore	ASIP for floating-point matrix processing in communication kernels
Primecore	ASIP for acceleration of FFT and DFT based on the prime-factor algorithm (e.g. in LTE modems), using custom data-path elements, SIMD and instruction-level parallelism
JEMA, JEMB	Dual ASIP for high-resolution JPEG encoding, respectively accelerating DCT and VLC
Tgauss	ASIP illustrating vectorization and memory management for image processing
MMSE	ASIP for acceleration of matrix operations as used in Minimum Mean Square Error Equalization
Tvox	ASIP for acceleration of voxel processing for SLAM (simultaneous localization and mapping)

For more information about Synopsys' ASIP design tools, visit: synopsys.com/asip