ASIP Programmer: Software Development Kit for Proprietary Processors

Highlights

- Full-featured SDK: C/C++ compiler with linker and assembler, graphical debugger including profiling support, cycle- and instruction-accurate instruction set simulator, on-chip debugger and sophisticated IDE
- C++ language support including the Standard C++ Library
- Enables software developers to easily take advantage of the unique and specialized architectural features available with the proprietary processor architecture
- (Optional) Multicore debugging support
- Supports on-chip debugging vendors/probes from Arm, Digilent, FTDI, Lauterbach, Macraigor, Segger

Overview

With ASIP Programmer, MIPS offers professional software development kits (SDKs) for in-house designed processors, also referred to as application-specific instruction set processors (ASIPs). Such an SDK consists of:

- An optimizing C/C++ compiler with linker and assembler
- An Instruction Set Simulator (ISS), with support for both cycle-accurate as well as instruction-accurate simulations
- A powerful debugging solution that works both on top of the ISS as well as with on-chip debugging solutions
- All accessible from a graphical integrated development environment (IDE)

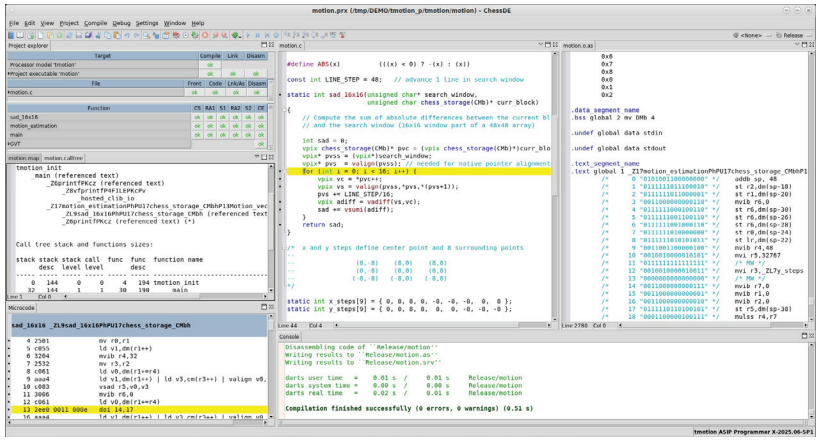

Figure 1: Development perspective in ASIP Programmer's IDE, showing compilation of an MPEG4 motion estimation function

ASIP Programmer provides SDKs for ASIPs such as those that have been described in nML, ASIP Designer's processor modeling language. Proprietary processors developed using ASIP Designer can be programmed with ASIP Programmer by leveraging the processor model that has already been developed. For other proprietary processors MIPS can develop a processor model as a service, which can then be used with ASIP Programmer.

ASIP Programmer builds upon a more than 20-year legacy of industryleading compiler, simulator, and debugger technology. It delivers the fastest performance and smallest code size, along with the best hardware and software visibility for software developers targeting code for application-optimized processor architectures.

ASIP Programmer comes in two flavors.

- **ASIP Programmer Flex** can be used to develop, debug, and analyze code for any number of processors, as long as a processor package is available (which is normally exported from ASIP Designer)
- **ASIP Programmer <myASIP>** can be used to develop, debug, and analyze code for the one processor for which it was created (i.e. the processor named <myASIP>). It is the solution of choice for deploying an SDK to a broad audience of software developers

Highly Optimized and Robust C/C++ Compiler

- Supported programming languages:
  - C, optionally extended with processor-specific data types and operators
  - C++ (based on the LLVM compiler front-end and extended for processor-specific data types)
- 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
  - Support of intrinsic function calls and of in-line assembly code
  - Lightweight 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

Instruction-Set Simulator

- 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, with automatic abstraction from the cycle to the instruction level
- Fast bit-accurate simulation of the application code on the simulation host processor (“native simulation”)
- Loading of Elf executable files, optionally containing source-level debug information in Dwarf format
- Application programming interface to 3rd-party simulators and integrated development environments, for co-simulation of the ASIP and its environment
- (Optional) Multi-ASIP simulation and on-chip debugging, supporting breakpoint export and synchronous stepping, requires a MultiCore IDE license

Back-End Tools

- Linker to build executable files from separately compiled Elf/ Dwarf object files for different source files or functions. It applies symbol-based linking (for minimal code size), reverse inlining and common tail elimination, selection of shorter call instructions
- Assembler and disassembler, to translate machine code from assembly into binary format and back

Powerful Graphical User Interface

- All tools are integrated in ASIP Programmer's graphical development environment, including options for integration with Eclipse or with Lauterbach's TRACE32® debugger
- The integrated development environment 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, including those of Arm, Digilent, FTDI, Lauterbach, Macraigor, Segger)

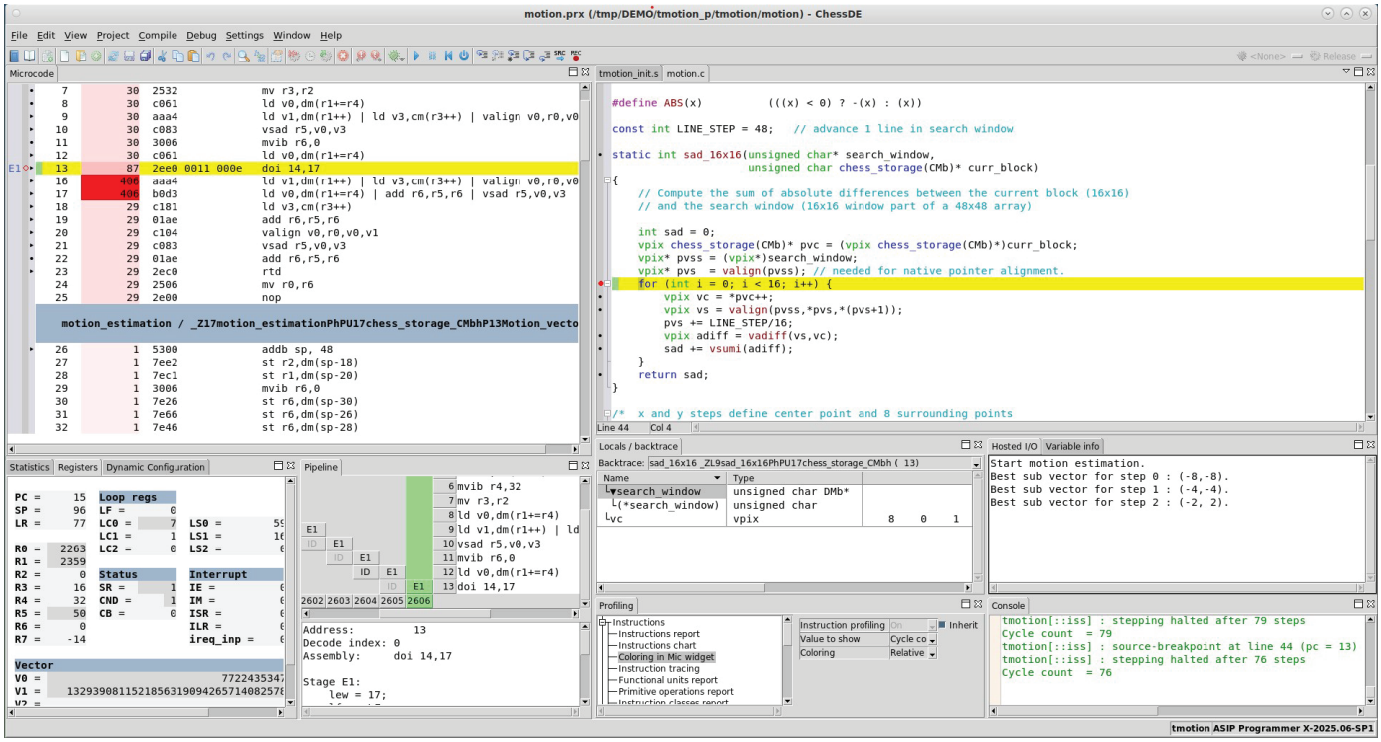

Figure 2: Debug perspective in ASIP Programmer's IDE, showing instruction-set simulation of an MPEG4 motion estimation function

Debugging and Profiling

- Source-level debugging, showing
  - Correspondence between executed instructions and source-code statements
  - Correspondence between register or memory locations and source-code variables, including a "de-pipelined" view on variable updates for a better debug experience in deeply pipelined processors
- Support for
  - Breakpoints on instructions and on source-code statements
  - Watchpoints on register and memory locations
  - Mix of hardware and software breakpoints in case of on-chip debugging
- Profiling
  - Profiling of instructions, storages, functional units, pipeline hazards
  - Profile reports at instruction and function level, visualized in the Microcode view (Figure 3)
  - Source code coverage reports, visualized in the source code windows

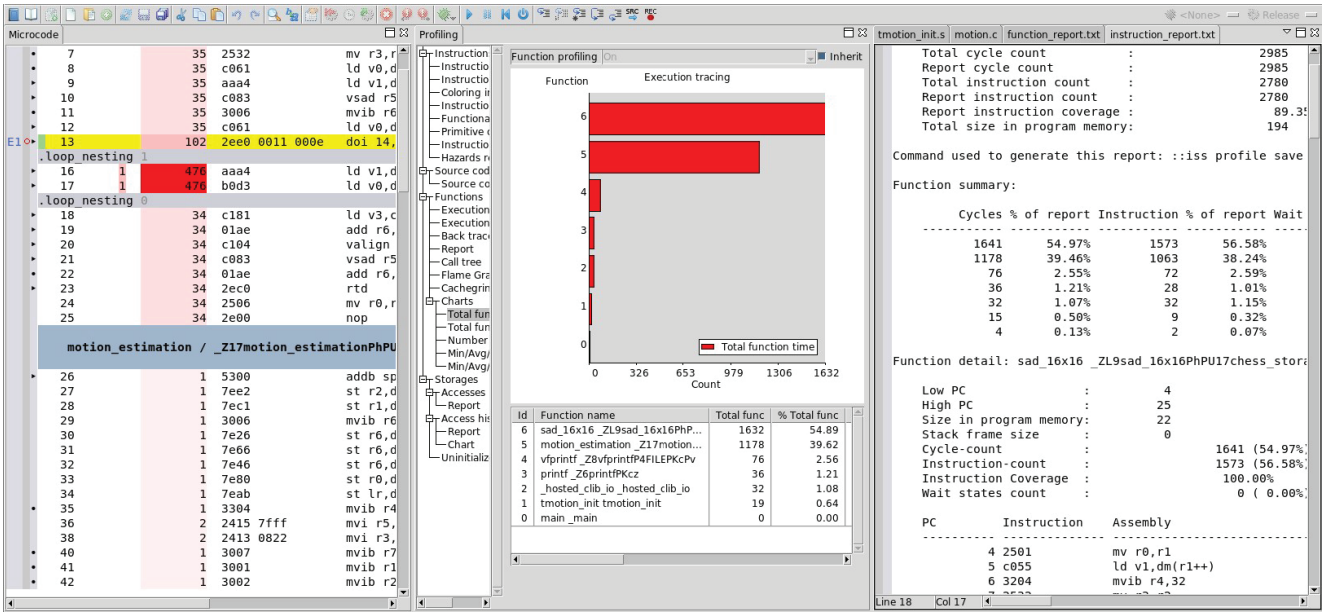

Figure 3: Profiling perspective in ASIP Programmer's IDE

ASIP Programmer <myASIP>

The ASIP Programmer <myASIP> product offering addresses customers with a need to provide an SDK for one specific processor (with the processor named <myASIP>) to a larger number of software developers.

For any such <myASIP> processor, MIPS will create an independently supported product. This processor-specific SDK, named ASIP Programmer <myASIP>, will get its own unique MIPS product code, and can be ordered as a standalone product. MIPS will serve as the distribution and support partner to make the SDK available to the market. Sublicensing is possible, i.e. the customer might order the specific product from MIPS, and makes it available to the end user. Using the ASIP Programmer <myASIP> offering eliminates the need for the owner of the <myASIP> processor to deal with the complexity of licensing, protecting, packaging, and distributing the SDK. Here is a summary of benefits:

- SDK specific to the <myASIP> processor becomes an official MIPS product, with a unique product code
- Fully packaged SDK, with installer and integrated license management
- Fully documented
- Guaranteed product availability
- Maintenance: When MIPS makes improvements to the underlying software, the SDK can be seamlessly upgraded to take advantage of the improvements
- Distribution control: Open source distribution rights and export compliance checks performed by MIPS
- Access control: Given <myASIP> is a unique architecture, the processor owner might want to restrict access to the SDK. So ASIP Programmer <myASIP> can only be ordered by the processor owner, or a 3rd party authorized by the owner
- Sublicensing: As an option, the processor owner can order the product from MIPS, and make it available to 3rd parties. Under the sublicensing option, MIPS is not involved in the business arrangement between the processor owner and the 3rd party
- Optionally, the SDK can be added to the MIPS Evaluation Portal. Through this portal, evaluation requests for the SDK can be filed and processed by both the processor owner and 3rd parties in an efficient, streamlined, and secure way.

Key Architectural Features Supported by ASIP Programmer	
Arithmetic	• General-purpose as well as application-specific arithmetic units
Data Types	• General-purpose as well as application-specific data types (e.g. fractional, 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.
Register Architecture	• From general-purpose register-files to special-purpose registers • Support of coupled operand and/or result registers
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	• Instruction level parallelism (e.g. VLIW) and data-level parallelism (e.g. SIMD), including combinations of both • Support of multi-threaded processors

Table 1: Key architectural features supported by ASIP Programmer

ASIP Programmer Add-On Products

MultiCore IDE

The Multicore IDE Add-On allows multi-ASIP simulation and on-chip debugging, supporting breakpoint export and synchronous stepping, both in batch mode and in a graphical IDE.

About MIPS:

MIPS by GlobalFoundries delivers software to silicon with RISC-V for building physical AI platforms. MIPS delivers software-hardware co-design, optimized AI, and custom ASSP design and manufacturing. Together with ARC, MIPS delivers the open, standards-based processor IP portfolio for embedded applications. Physical AI is built on MIPS.

For more information, visit **www.mips.com/arc.**