SYNOPSYS®

DATASHEET

Synphony C Compiler

High-level synthesis from C/C++ to RTL

Overview

Synphony C Compiler reduces development time and cost by shifting the focus of hardware implementation from designing at the RTL level to designing at the algorithmic level. Designers describe hardware in algorithmic C/C++, using Synphony C Compiler to fine-tune the architecture for tradeoffs in performance, power, and area, and generate highly optimized RTL implementations for ASIC or FPGA. The tool infers the necessary interfaces and memory structures to implement an optimized architecture, eliminating the need to code explicit parallelism and control logic. It is capable of handling multi-million gate hierarchical designs with efficient hardware sharing across layers of hierarchy.

Key Benefits

- 5-10X productivity increase over manual RTL coding
- Superior quality-of-results (QoR) for area and timing using Design Compiler® for RTL synthesis and Synplify for FPGA synthesis
- Faster turnaround time implementing large, multi-million gate designs
- Seamless end-to-end verification from the original C/C++ model to RTL validation and verification environments

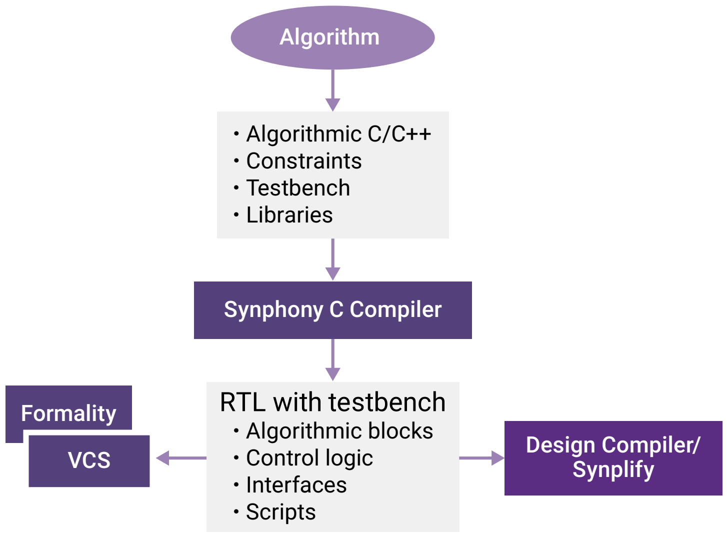

Figure 1: Synphony C Compiler enables high-level synthesis, simulation and debug from C/C++

Key Features

- High-level synthesis based on single-threaded untimed, sequential C/C++ model, constraints, testbench, and libraries
- Support of standard (AXI, AHB, OCP, etc.) and custom external interfaces to eliminate glue logic
- High-level synthesis optimizations for superior performance and area results
  - Automated single-to-multi-threaded transformations
  - Hierarchical block-level resource sharing
  - Automatic scheduling and pipelining
  - Timing optimizations for variable bounded loops
- RTL optimized for best QoR using Design Compiler
- Recursive hierarchical compilation enables an arbitrary number of hierarchy levels for optimization
- Architectural clock gating for fast implementation of complex, low power designs
- Automatic generation of testbench, design files, constraints, and scripts for logic synthesis, power optimization, and verification tools

Design at a Higher Abstraction Level

Capturing IP at an algorithmic, transaction level and performing high-level synthesis using Synphony C Compiler improves design productivity by 2X over manual coding of new RTL blocks (figure 1). Synphony C Compiler accepts a broad subset of C/C++. Accepted data types include native C, parameterized integer, and fixed-point. An algorithm can be described in terms of an untimed, sequential C/C++ model using a natural, sequential coding style that includes nested loops and multiple levels of C/C++ function hierarchy. In addition, Synphony C Compiler provides reusable fixed-point, image processing, and streaming libraries to accelerate the development of synthesizable C/C++ IP. Other high-level synthesis inputs include hardware constraints and a C/C++ testbench. If custom interfaces are desired, the user provides a C++ transaction-level model of the interface and constraints for signal naming and behavior.

Synphony C Compiler's high-level synthesis engine then applies single-to-multi-threaded transformations to create parallelism from sequential C-code, generating synthesizable RTL code with a simulation testbench, control logic interfaces, and scripts for synthesis, verification, and downstream tools.

To easily integrate the C/C++ IP into a system-on-chip, designers can use standard (AXI, AHB, OCP, etc.) or custom interfaces (figure 2). In the latter scenario, it is possible to specify the exact interface with signal names and cycle behavior, or a relative scheduling of operations. These interfaces may be encapsulated so that users can call them from a high-level API, preserving the abstraction benefits of high-level synthesis.

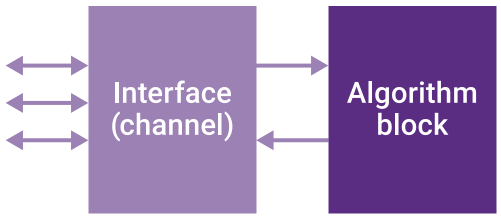

Figure 2: De-coupling the interface from the algorithm simplifies coding of the algorithm

Superior Quality-of-Results

Synphony C Compiler performs architectural optimizations such as resource sharing, automatic scheduling and pipelining, and timing optimizations for variable bounded loops to achieve designer-specified performance and area goals in the RTL implementation.

In addition, Synphony C Compiler performs specific optimizations that achieve superior QoR using Synopsys Design Compiler for RTL synthesis and Synopsys Synplify for FPGA synthesis. Synphony C Compiler's high-level synthesis engine makes decisions based on predicting area and timing results that Synopsys' logic synthesis solutions can achieve, and generates RTL that complies with the synthesis tools' coding style guidelines that produce the best QoR. For example, Synphony C Compiler generates RTL with hierarchy and syntax matching Synopsys Coding Guidelines for Datapath Synthesis to enable maximum datapath extraction, optimal timing, and optimal area in Design Compiler. The RTL structure also facilitates fast verification using Synopsys Formality® equivalence checking.

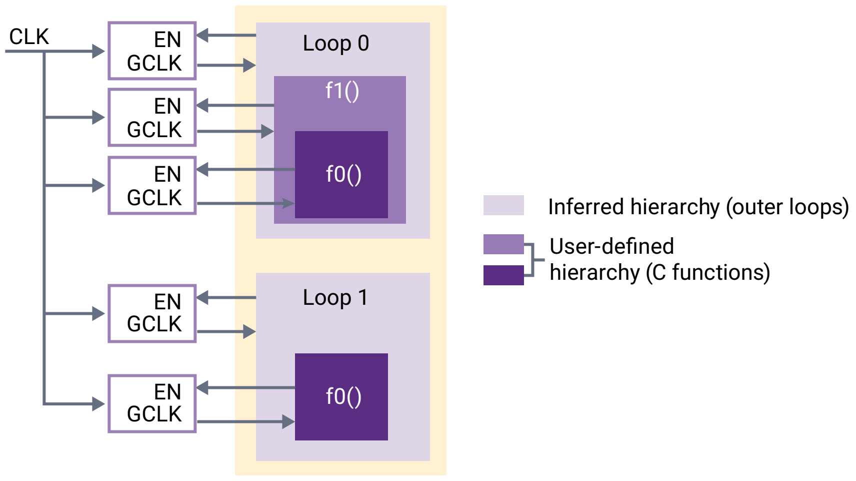

Figure 3: Architectural clock gating in Synphony C Compiler

Faster Turnaround Time for Large Designs

Synphony C Compiler enables efficient hardware sharing in large, complex systems designed from multiple levels of C/C++ function hierarchy. Recursive hierarchical compilation improves designer productivity by enabling an arbitrary number of hierarchy levels for optimization while providing users flexible control of constraints at each level.

Architectural clock gating enables fast implementation of complex, low power designs (figure 3). Designers can apply gating at the level of loops and/or C functions to shut-off entire processing blocks. Synphony C Compiler's block-level clock gating complements register-level clock gating performed by Power Compiler™ to achieve additional power savings.

Seamless End-to-End Verification

Synphony C Compiler provides seamless end-to-end verification from the original C/C++ model to RTL validation and verification environments. Dynamic linting of the C/C++ source early in the design cycle uncovers coding issues such as out-of-bound accesses, uninitialized variables, and other undefined behavior which could otherwise propagate to RTL. In addition, Synphony C Compiler generates a UVM-based SystemVerilog testbench. The testbench drives the Synopsys VCS® simulation with test vectors derived from the original C/C++ testbench, generating reports and graphical views that summarize the functional and performance results. A transaction-level view of the hardware operation is enabled in the Synopsys Discovery Visual Environment (DVE) graphical interface to VCS. The designer can easily simulate the system throughout the high-level synthesis flow and quickly hand off models, testbench, and test data to verification teams.

For more information about Synopsys products, support services or training, visit us on the web at: synopsys.com, contact your local sales representative or call 650.584.5000.

SYNOPSYS® Silicon to Software™