



```
#include <stdio.h>
#include <systemc.h>

static sc_uint<8> shift[3];
static sc_uint<8> coeffs[3] = { 15,10,5 };
static sc_uint<8> sum_of_coeffs;
static bool    initialized = 0;
static sc_uint<8> values[16] =
{129,244,13,16,19,41,29,171,129,129,229,41,19,16,17,4 };

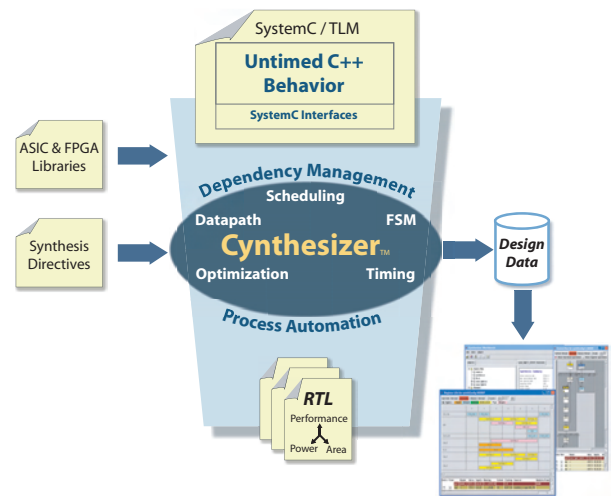
sc_uint<8> filter_func( sc_uint<8> input_data )
{
    if ( !initialized ) {
        shift[2] = shift[1] = shift[0] = input_data;
        sum_of_coeffs = coeffs[0] + coeffs[1] + coeffs[2];
        initialized = 1;
    } else {
        shift[2] = shift[1];
        shift[1] = shift[0];
        shift[0] = input_data;
    }
    return ((shift[0]*coeffs[0]) + (shift[1]*coeffs[1]) +
    (shift[2]*coeffs[2])) / sum_of_coeffs;
}
```

Cynthesizer reduces design effort for production ASIC and SoC designs by automating the generation of high-quality RTL code from high-level design descriptions. Its practical approach to behavioral synthesis enables a systematic methodology that accelerates design creation, verification, and closure for silicon-proven results. Many of the world's largest systems and semiconductor companies are using Cynthesizer for production designs.

Most productive path to silicon

Traditional RTL design methods achieve approximately 200K verified gates per designer per year for new blocks. Cynthesizer users report results as high as 2.0 million gates/designer/year. These productivity gains are even more pronounced for derivative designs where features are added, algorithms are changed, or performance targets are modified. Cynthesizer enables behavioral IP reuse, which dramatically reduces risk in your design schedules by completing tasks in hours or days that used to take weeks or months.

These dramatic productivity gains can be achieved without compromising on performance, silicon area, or power. Cynthesizer users have demonstrated the ability to rapidly produce high-quality RTL for high-volume consumer products over more than five years. They consistently achieve results equal to or better than hand-written RTL in silicon area and power consumption.

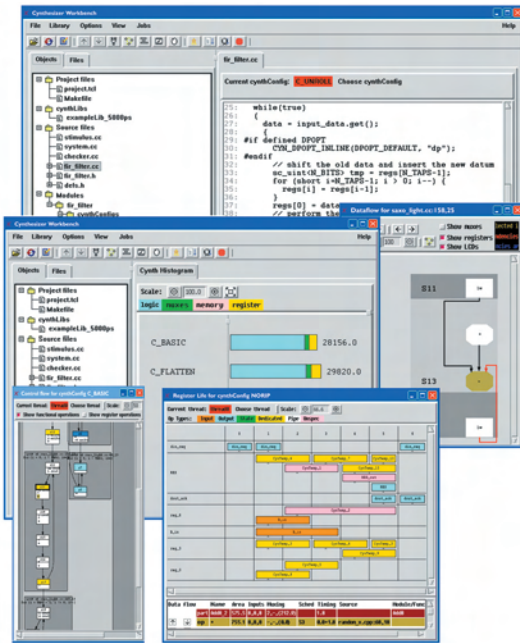


Key Benefits and Features

- **Most productive path to silicon**
 - Production-quality RTL
 - 2.0M gates/designer/year
- **Increased return on engineering investment**
 - Easy retargeting
 - Effective IP reuse
 - High-level abstraction
- **Standards-based SystemC synthesis**
 - Untimed C++ behavior
 - Structural hierarchy
 - Multiple concurrent processes
- **Maximum interface flexibility**
 - Built-in interface classes
 - Custom interfaces written in SystemC
- **Rapid design exploration**
 - Find an optimal architecture
 - Change performance without rewrite
- **Most complete design flow**
 - Integrates with existing RTL flow
 - Automates execution of all tools
 - Allows use of any process tech library
- **Eliminate backend timing problems**
 - Exhaustively analyzes all timing paths
 - Systematically produces RTL that meets timing

Increased return on your engineering investment

Semiconductor companies spend millions of dollars on design engineering and verification to produce production-worthy design source code. Unfortunately, with the traditional RTL design flow, the low-level cycle-by-cycle, explicit FSM nature of this source code means that you often have to create brand new RTL to make a conceptually small modification to the design. Using Cynthesizer with high-level SystemC, your verified source code can be reused without modification for widely different process technologies and clock speeds. Modifications to the algorithm, architecture, or interface can be made incrementally at the high level, while they often require a complete rewrite of RTL. Cynthesizer will significantly reduce your overall design effort and maximize your ROI.



Standards-based SystemC synthesis

Cynthesizer supports industry standard IEEE-1666 SystemC semantics. This allows designers to use familiar hierarchical decomposition techniques to manage design complexity. By supporting multiple threads, it allows design and verification of multiple algorithms operating concurrently, providing a level of accuracy not possible with other ESL languages.

Because SystemC is a C++ library that adds hardware constructs, C or C++ algorithm code can be put into the context of a hardware module with hardware interfaces without any translation.

Rapid design exploration

When designing in RTL, the structure of your code is highly variable depending on design attributes such as throughput, latency, silicon area, timing and power. By automatically creating the detailed RTL structure, Cynthesizer allows you to quickly and easily trade off these critical design attributes using a straightforward progressive methodology. With Cynthesizer's design exploration capabilities, you'll get the right design in less time.

Maximum interface flexibility

Cynthesizer provides the maximum interface flexibility of any high-level synthesis product. It includes automated generators for many interfaces so you can quickly configure and deploy interfaces for the most common interface problems. In addition, Cynthesizer allows you to specify any custom interface that you may require by adding it to your design as synthesizable SystemC code. All of these interfaces can be simulated at the PIN-level in a system context or using the same testbench that will verify the resulting RTL. Cynthesizer includes built-in support for a wide variety of interfaces:

- **Cynw_p2p interface classes**
Supports streaming point-to-point connections
- **Memory generator**
Flexible specification of embedded memories
- **Buffer interface**
Supports single- and double-buffered memory interfaces
- **Circular buffer interface**
Provides tight coordination between processes sharing a memory
- **Line buffer interface for 2-D imaging applications**
- **Trigger-done interface**
Synchronization of master-slave processes

All of Cynthesizer's interface options provide automatic synchronization to avoid access conflicts so you can build reliable systems without debugging communication problems.

Eliminate backend timing problems

Cynthesizer ensures easy timing closure for the generated RTL by exhaustively analyzing each path and scheduling operations so they easily fit in the given clock period. You begin the high-level synthesis process by specifying a target clock period and a technology library (foundry .lib file) for a specific process technology. Cynthesizer uses its patented datapath optimization technology to precisely determine the propagation

delay of each needed operation (such as an adder, multiplier, or multiplexor). Directives are available to allow user control of how aggressively Cynthesizer packs these operations into the available clock period.

Design with Cynthesizer and save months of effort by eliminating backend timing closure problems.

Most complete design flow

Cynthesizer comes with a fully integrated automation system called the Behavioral Design Workbench (BDW). BDW automates the time-consuming task of writing makefiles and tool integration scripts. It provides dependency management and process automation to implement a complete high-level design and verification flow. BDW automates the execution of:

- **C++ compilation and linking**
- **Wrapper generation for co-simulation with HDL simulators**
- **Logic synthesis**
- **Formal equivalence checking between RTL and gates**
- **Power analysis**
- **RTL analysis and debug**

FORTE DESIGN SYSTEMS

Corporate Worldwide Headquarters:

Forte Design Systems
100 Century Center Court, Ste 100
San Jose, California 95112 USA
T: 408.487.9340 / 1.800.585.4120
F: 408.487.9344
E: sales@ForteDS.com
www.ForteDS.com

International Offices:

Japan
Forte Design Systems KK
Kaede Dai2 Building 3F
2-5-10, Shin Yokohama, Kouhoku-ku
Yokohama-shi, Kanagawa-ken, 222-00333
T: 045-478-2268
E: japan@ForteDS.com

Europe

Forte European Headquarters
WTC Business Center
5 Place Robert Schuman
38000 Grenoble, France
T: +33 468 940 808
E: europe@ForteDS.com

Copyright 2008 Forte Design Systems. All rights reserved. Rev 05/08.
Cynthesizer is a trademark of Forte Design Systems.
All other company and product names are trademarks or registered trademarks of their respective companies.