

## HDL FRONT END

### Customer Overview

A large semiconductor & system design company in Japan. The customer has an in-house CAD group involved in developing tools for internal requirements. This HDL Front End was developed as part of a long term engagement with this group. This tool can be used as a key building block for any HDL based product development.

### Objectives

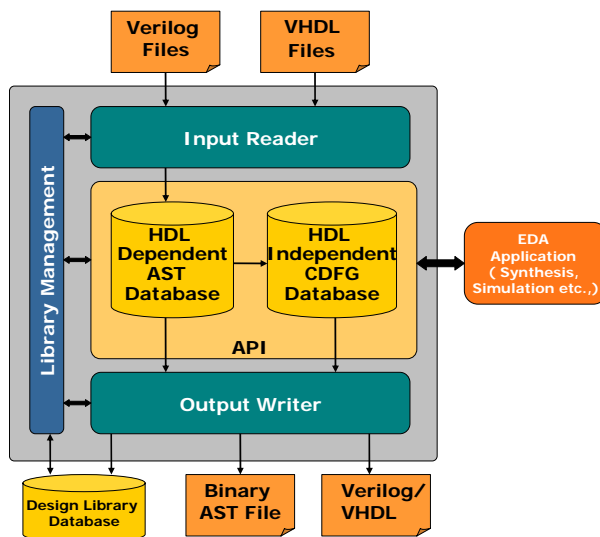
The tool was to be used as Front End for tools such as Logic Synthesis, RTL Inferencing, RTL Code Style Checking and HDL based formal verification. Hence, it had to be easy to integrate with any HDL based tool.

The tool was required to be able to accept VHDL or Verilog as input and was to be fine tuned to handle multi-million gate designs. Multiple Platforms like Sun Solaris, Linux, Windows NT were to be supported.

### Solution

The HDL Front End has the following components:

#### HDL Front End Architecture



**Verilog and VHDL Compilers** read the input Verilog or VHDL description respectively, and populate an Abstract Syntax Tree [AST] database. The compilers also report semantic /syntax errors

in the HDL code. In addition, the compiler,

- Stores the design as a library
- Translates AST database into CDFG database
- Performs optimizations on the CDFG database
- Stores AST and CDFG database in a binary file

**AST Database:** These are the key features of this AST database of the HDL Front End tool:

- In-memory representation of input
- Source code association through file name, line number and column number information
- User-specified comments in comments-database

**CDFG Database:** The Verilog / VHDL Compiler translates the AST database into a Control and Data Flow Graph [CDFG] representation:

- Represents control flow and data flow
- Independent of the HDL
- Preserves the semantics of the input language

**API's for database access:** The API's enable other applications to interface easily with the HDL Front End and also allow the designer to further elaborate and optimize the design. Some of the high level

Optimization APIs	Post Processing APIs (for VHDL)
<ul style="list-style-type: none"> <li>■ Static expression folding</li> <li>■ Constant propagation within/across design hierarchy</li> <li>■ Loop Unrolling</li> <li>■ Inline expansion of function/procedure calls</li> <li>■ Design hierarchy flattening</li> </ul>	<ul style="list-style-type: none"> <li>■ Expansion of generate statement</li> <li>■ Expansion of aggregate construct</li> <li>■ Resolution of unconstrained ports/parameters</li> </ul>

optimization & post-processing API's include

**Library Management:** The Verilog/VHDL compiler also stores the HDL description into a library, whose features include:

- Separate/Incremental Compilation of the Design into user-specified design library. Components present in one library can be instantiated in another
- Dynamic linking of design description from any level of the design hierarchy

**Database Utilities:** These are provided to enable reading from and writing into the binary AST and CDFG databases:

- Database Reader loads the binary AST and CDFG database file and performs dynamic linking
- Database Writer stores the AST and CDFG database in binary file and generates Verilog/VHDL back.

**RTL Inferencing:** In a subsequent engagement, we have enhanced the tool with an RTL Inferencing component:

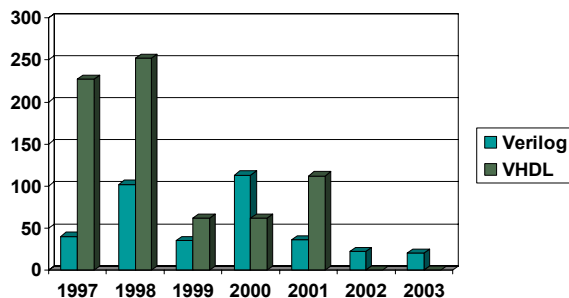
- For synthesizable RTL subset, performs special checks for conformance to industry standard subset and issues appropriate error and warning messages
- Performs RTL Inference of logic gates, sequential elements, state machines, tri-state buffers from 'Z' assignment and functional modules from operators
- After inference of all RTL components, a Verilog netlist in terms of a generic library is generated.

## Results

Currently, this tool is being used by over 2000 designers in the customer organization as part of various HDL based tools.

HDL Front End has been tested extensively on million gate designs for performance. A feature-wise test bench was used for completeness in testing

### Bug Statistics



The test results are an indication of the quality and stability of the software.

The tests proved the speed, memory efficiency and scalability of HDL Front End for large designs. As shown in the diagram below, the speed and memory requirements are linear with respect to size of input.

### Speed & Memory Characteristics

