NOM™

The Language Independent Netlist Object Model

Highlights

- Preserves design hierarchy
- Readers and writers for Verilog, VHDL, and EDIF 200
- Intuitive API for language independent front-end development
- Language-independent expression support
- API functions to access, modify, and delete nodes on the netlist
- Support for RTL and behavioral constructs
- User controlled, fully instantiated tree available
- Binary dump and restore capabilities
- Customizable error handling
- Optimized for memory usage and runtime performance
- Support for preserving or flattening design hierarchy in a netlist
- Efficient and memory optimal data structures to represent bus based designs
- High run time performance while working with large vector based designs

Interra's Netlist Object Model (NOM) provides EDA tool developers with a language independent front-end for netlist-based applications. Supporting Verilog, VHDL, and EDIF designs, NOM represents connectivity information of a design at structural level. NOM's Application Program Interface (API) enables EDA applications to traverse, access, and modify connectivity information.

NOM stores the connectivity information in a hierarchical, language-independent object model. NOM's interface hides language-specific details from an application program and provides a unified view of any netlist description through a common API. The API provides intuitive functions to access and modify the in-memory object model. NOM can be easily extended, allowing users to build on the existing capabilities to meet application-specific requirements from a netlist representation.

Robust and field-tested, NOM is widely used in a wide range of applications, including language translators, schematic generators, FPGA partitioning, and front-end to auto place and route tools.

NOM is available on Solaris, HP-Unix, Linux, and Windows platforms.

Key Advantages

- Language independent object model
- Support for Verilog, VHDL, and EDIF 200
- Backed by Interra's field-proven expertise in developing VHDL and Verilog front-ends
- Comprehensive coverage of language constructs
- Comprehensive validation of syntax and semantics

The NOM Features

Readers and Writers for Verilog, VHDL, and EDIF

Readers and writers for the standard netlist formats - Verilog, VHDL, and EDIF 200, are available as off-the shelf objects. The readers incorporate the widely accepted HDL language analyzers from Interra. The writers are customizable to application-specific requirements. With readily available front-ends, you can concentrate on your core competencies, without having to bother with language interfaces and compliance.

Dynamic Object Model

NOM provides a complete set of access functions for retrieval of the netlist data from the in-memory object model. The API supports a rich complement of functions for search and traversal of the netlist hierarchy. In addition, NOM provides a complete set of APIs for creation, modification, and deletion of netlist components. The API functions enable you to dynamically change the netlist during application runtime. You can even use API functions to populate NOM for proprietary netlist and formats.

Support for Expressions

NOM supports expression in a language-independent manner. Various expression classes - unary, binary, range selects, concatenations, conditional operators, and constants provide you with sufficient functionality to write real-world application programs using NOM.

Binary Dump and Restore Mechanism

NOM provides real time dump and restore mechanism. You use NOM API to dump the object model to a binary file or restore a binary dump to the memory. You can either dump a part of the object model or the complete object model. The dump and restore mechanism is useful for fast loading of previously read netlists, for quick netlist interchange between application programs, or to reduce the peak memory consumption. The dump and restore mechanism uses compression methods to generate small dump files.

Fully Instantiated Object Model

NOM API enables you to control generation of either a hierarchical or a fully instantiated object model. If you choose to generate the fully instantiated tree view, you can traverse fully instantiated networks using API functions.

RTL and Behavioral Constructs

NOM API functions enable you to anticipate RTL and behavioral representation in your applications. Although a netlist is a structural-level representation of a design, NOM supports RTL and behavioral constructs in a limited way. NOM retains the expressions and models in a language independent manner. In addition, NOM also stores continuous assignments of Verilog, and concurrent signal assignments of VHDL.

Constructs that are not supported by NOM are converted into black boxes and incorporated in the netlist, such as always blocks of Verilog and process blocks of VHDL.

Customization of Errors

You can customize information, warning, and error messages to the needs of your application. You can enable or disable messages, change message formats, modify message severity, and even add new messages.

Optimized for Performance

NOM is designed to optimize memory usage, ensure low memory footprint, and result in high performance of frequently used functions.

Many of today's advanced designs mostly deal with vectors like buses. There is a lesser need to access the individual bits. For such designs, when not required, NOM is capable of preserving the vector property without creating individual bit components. This ensures substantial gain in terms of high performance and low memory footprint.

Object Oriented Design

NOM is designed using object oriented modeling techniques. The object oriented design makes the data structures easy to learn and intuitive to use.

Interra Systems, Inc.
1601 S. De Anza Boulevard, Suite 212, Cupertino, CA 95014
Phone: +1 408 579 2000, Fax: +1 408 579 2050
Email: info@interradesign.com Web: www.interradesign.com