AGNISYS®

SYSTEM DEVELOPMENT WITH CERTAINTY

THE MOST COMPREHENSIVE & CUSTOMIZABLE SOLUTION FOR AUTOMATED SOC ASSEMBLY & HARDWARE-SOFTWARE INTERFACE (HSI) DESIGN

What's Inside

- Design entry using an integrated development system
- Import SystemRDL, PSS, IP-XACT, YAML, RALF, CSV, JSON, and custom XML
- Parameterize and generate RTL Code and UVM register models
- Generate C/C++ header files, C-tests, API and classes
- Generate SoC datasheet documentation as HTML, PDF, Markdown, and Word documents
- Generate a complete UVM test environment with sequences
- Specify algorithms and generate portable sequence code for verification, firmware and post-silicon validation
- Assemble the entire SoC
- Get the best support in the EDA industry
- Industry standards-compliant and certified by TÜV SÜD for ISO26262

What's the root cause of functional flaws?

- The functional specification remains the number one root cause of IP/SoC functional flaws. It's also the main source of all design and verification activities. Any errors in the specification often lead to higher costs for fixing them and can even be the difference between success and failure for a given project.
- The Hardware/Software Interface (HSI) data is a major component of the specification because it controls the configuration of peripherals and their communication with the software.

The HSI

- Includes the definition of addressable registers, interrupts, sequences, ports, and APIs
- Is often manually captured and maintained in Word™ or Excel™ , and used in conjunction with various files such as IDSNG or SystemRDL
- Is used by multiple isolated teams including system architects, designers, verification engineers, firmware developers, lab testers, and software developers
- Registers need to be designed in SystemVerilog or VHDL, and the corresponding UVM models need to be created for verification
- C/C++ APIs are needed for the development of the firmware
- The documentation is central to the correct and timely development of the IP/SoC

Specification to code generation

- *How much time do you spend creating register design files, and verification environments?*
- *Did you know that you can save precious time in your development schedule and bridge the gap between specification, design, verification, and validation?*

How much time does your team spend in connecting 3rd party or home grown IPs? How many design flaws are introduced when hierarchy restructuring is done manually? How much time do you spend creating register design files, and verification environments?

Did you know that you can save precious time in your development schedule and bridge the gap between specification, design, verification, and validation?

With the amount of data and the various SoC teams who create and rely on the HSI, it can easily get chaotic without a process in place – this is the primary reason why the specification is the main culprit of functional flaws.

Industry's most comprehensive and powerful HSI design and verification solution

Established in 2007, Agnisys® has been exclusively focused on solving the challenges associated with HSI. Our intuitive register and sequence editors are easy to use, and the powerful code generators are customizable for generating signoff quality code in different formats for various IP/SoC teams. You can specify a single test sequence and our portable sequence generator generates sequences in various formats. We generate a complete UVM test environment that provides 100% register functional coverage without manually writing any UVM code.

We generate C tests as well as a hybrid C-UVM environment that helps validate your design in a system context. All the above and much more can be accomplished using the IDesignSpec Suite that includes:

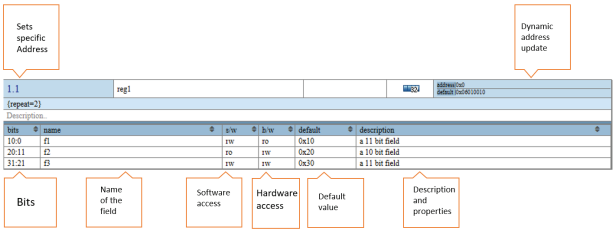
- IDesignSpec™ GDI
- IDS-Verify™
- IDS-Integrate™
- IDS-Batch™ CLI
- IDS-Validate™
- IDS-IPGen™

IDesignSpec™ GDI : Graphical Design Interface and Interactive Generator

Originally developed in 2007, our Editors are equipped with user-friendly register templates to assist you during register specification. Simple and complex registers can be created hierarchically such that large SoC designs are divided into manageable sub-blocks that are represented symbolically, designed, and connected together. This methodology enables you to work on different parts of the design in parallel with a large team.

Using IDesignSpec™ GDI graphical interface and interactive generator, the IDS-NG™ editor or an add-in for Word, Excel or OpenOffice allow you to:

- Specify the registers including register field names, widths, description, and access types, and various properties
- Import and integrate IP-XACT, SystemRDL, YAML, or JSON
- Define RTL Properties for clocking, reset, special registers, counter, interrupt, multiple domains, and memory mapping technology
- Parameterize the auto-generated code using static values and expression with common operators
- Check for 1000s of consistency errors

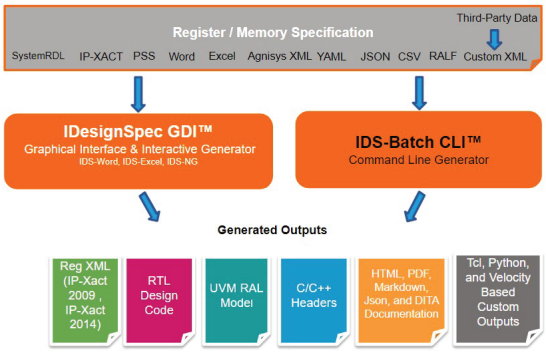


IDesignSpec™ GDI

With our extensive support for parameterization, you can specify various parameters and use them as macros in the specification. The value of the parameter can be static or based on expressions.

The specification linter helps you create a correct-by-construction specification by checking the following data:

- Address calculation and back-annotation on the specification
- Inserts Register Map – Table of Contents (address, default values)
- Overlaps (bit, register, reggroup, and block)
- Incomplete data or incorrect data
- Duplicate or illegal names
- Invalid access or incompatible access



IDesignSpec™ GDI data flow diagram

Benefits of IDesignSpec™ GDI

- Includes a platform-independent editor that enables you to capture the specification of your IP/SoC
- Word/Excel-like Reg field view to create specification on any platform
- Enterprise team can collaborate and create IP/SoC with versioning and configuration using Git integration feature
- A single UI for capturing all information related to IP/SoC
- Capture sequences and checker view for registers

Code Generation

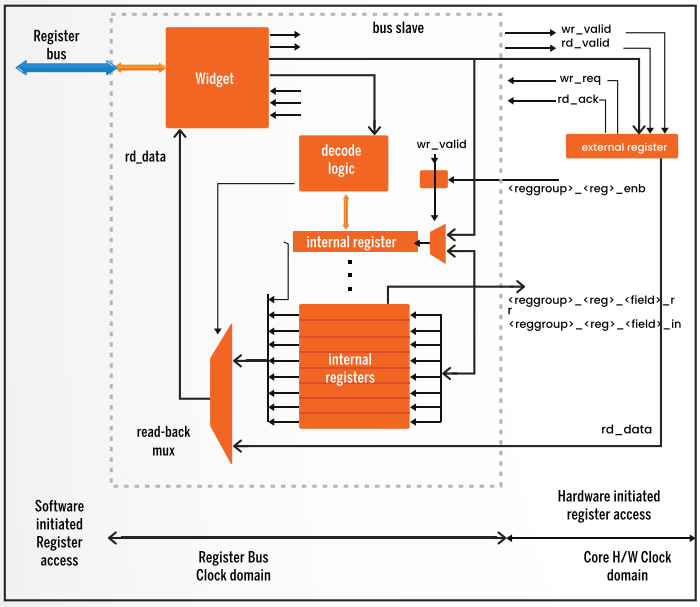
Based on the golden specification, various SoC teams can use the high-performance code generators via the GUI through IDesignSpec ™ GDI or the command line using IDesignSpec™ CLI.

IDesignSpec™ CLI: Command Line Generator

IDesignSpec™ CLI is a command line tool for transforming register specifications. It accepts various input formats and can generate a variety of outputs. IDesignSpec™ CLI supports more than 400 special register types, including indirect, indexed, read-only/write-only, alias, lock, shadow, FIFO, buffer, interrupt, counter, paged, virtual, external, read/write pairs, and combinations of these types. This tool is typically used by design, verification, firmware, and documentation groups for managing register map and memory map information.

RTL Code Generation

The generated VHDL, Verilog, SystemVerilog, or SystemC for the registers is human-readable with easy-to-follow comments. The RTL also includes a bus slave and decode logic specific to the bus protocol **(APB, Avalon, Wishbone, AHB, AHB-Lite, AXI4, AXI5, AXI4-Lite, Tilelink, I2C, SPI, or Custom)**, ensuring instant connection of the application logic to the register bus.



RTL generated by IDesignSpec™ GDI

A comprehensive list of RTL Properties can be defined hierarchically in the golden specification and reflected in the generated RTL code. This leads to a more customizable RTL to meet your design requirements. For example, you can easily specify `clock_edge=posedge`, `reset_type=async`, or `reset_level=high`, and the generated RTL code reflects them.

The RTL Properties can be used to specify special registers such as Shadow, Interrupt, Counter, Alias, Indirect, Lock, FIFO Trigger Buffer, Wide, Multi-Dimensional, RO-WO Pair, Virtual, TMR, and Paged registers.

The generated RTL code supports the following:

- External Registers - a register or reggroup implemented by the user outside the generated RTL
- Pipeline Stages - in order to meet special timing requirements
- Special Control Signals - examples include Write Pulse, Write One Pulse, Write Zero Pulse, Read Pulse, and Clearing field on remote signals
- Low Power Output - eliminates assigning the same value at every clock edge or eliminating write operations altogether

Functional Safety and Other Advanced Features

When we generate RTL registers, we build a system with automatic protection against failures requiring additional features such as:

- Parity: Adds a parity bit to a sequence of binary valued registers in the digital logic so that any corruption of the design by a single bit would lead to an inversion of polarity and would be detected.
- CRC: Calculated on a piece of the time of transmission (write transaction) and checked at the receiver time (read transaction).
- SECDED: Single Error Correction, Double Error Detection, which generates parity bits for some data bits, transmits the parity bits along with the data, and checks at the receiver's end by generating parity bits for the data using the same method, and then XORing with the transmitted parity to check if an error occurred or not.

Additional advanced features are required for complete SoC development:

- CDC: Various IP blocks within an SoC are often required to work in different clock domains in order to satisfy the power constraints. Various techniques such as mux synchronizer, flop synchronizer, and handshake synchronizer are used to avoid metastability as signals cross from one clock domain to another.
- MBD: You can specify multiple bus domains in which the block or chip resides. Two or more domains can be described; this architecture is widely used for improving performance
- Clock Gating: This is a method to turn off the power when it is not needed. It is used in SoC design today as an effective technique to save power.

C Header Files and API Generation

Other project-critical files can be generated, including SystemRDL, CMSIS-SVD, IP-XACT, and XML as well as the C/C++ header files and API needed by the software team.

The Document Generator can output file formats such as HTML, Markdown, DITA, .doc, .PDF, .xls, IP-XACT, or SystemRDL, which are customizable.

Output Customization

You can customize the outputs using our Velocity Template based on a very simple architecture with two components: a data model and a template. The data model is basically the input which describes the design specification, and the template is a plain text file that describes the static output and dynamic elements from the data model. This enables you to generate specialized code and documents required by your customers.

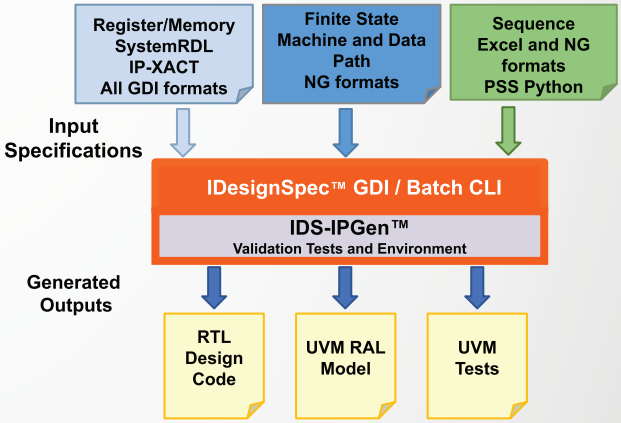
In addition, Tcl and Python APIs are provided to create custom outputs. You can use these to create workflows based on your specific needs

IDS-IPGen™: Configurable Standard and Custom IP

A collection of standard IP generators is provided to accelerate your development. IPs such as GPIO, PWM, PIC, Timer, SPI, I2C, I2S, AES, UART, and DMA are currently included in the library. So why waste time on creating them manually when you can generate them?

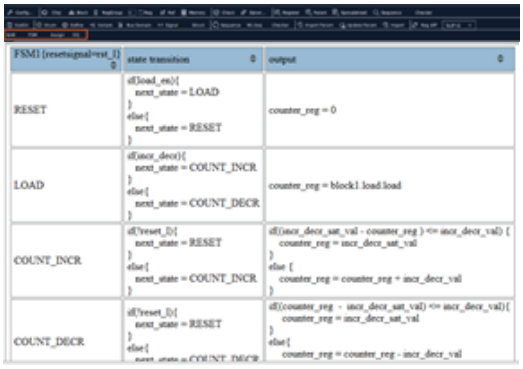
- All IPs are totally configurable and customizable (e.g., adding your own registers to IP register space).
- Customization includes: Adding fields to existing registers, adding additional registers, and even having dependencies on IP events and adding arbitrary logic to the IP.
- Unencrypted code: All the generated files are available as plain text for easy debugging and use by downstream tools.
- IPs come with standard programming sequences so the firmware/-software team can program them.

Once the register specification is captured, designers work on creating a synthesizable application logic layer for the intended functionality using these addressable hardware registers. Creating this application logic layer often consists of using various design constructs. Automation techniques save time and eliminate the manual work of creating application logic with the help of IDS-IPGen™.



IDS-IPGen™ Data Flow

- IDS-IPGen offers the creation of a synthesizable RTL application logic layer, a UVM prediction model, and tests by using a few predefined templates
- The following outputs can be generated by IDS-IPGen:
  - RTL design code
  - UVM model
  - Automated tests
  - Deadlock Detection
- These templates capture all the necessary information required to build up an application logic, including:
  - Finite state machines (FSMs)
  - Custom flip flops
  - Continuous assignments



UVM Model Generation

The generated UVM register model is also human-readable, with easy-to-follow comments, and includes hierarchy, register arrays, memories and coverage, constraints model, and hdl_path. Special registers specified in the spec using properties are also implemented so as to reflect their behavior accurately.

By defining a UVM property called “coverage” you can control the coverage for any particular register, reggroup, memory, or block. This property is hierarchical so if applied at the chip level then it automatically sets the coverage for the rest of the elements.

You can add the hdl_path using a property named “hdl_path” with a value set to the hierarchical path. hdl_path is a mechanism by which each individual element in a UVM model is connected to the RTL model of the element.

Beyond the UVM Register Abstraction Layer (RAL) model, the IDesignSpec™ Suite can also generate the register tests, environment, and formal assertions using IDS-Verify™.

IDS-Verify™: Verification Tests and Environment

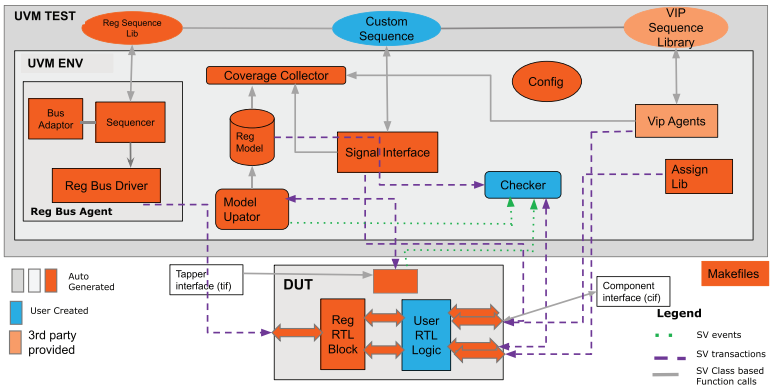
Our solution automates the tedious effort of creating a UVM based verification environment that provides ~100% functional coverage for registers. You can obtain faster coverage closure with the use of constrained-random stimulus generation, and auto-generation of coverage groups and illegal bins for different register behavioral scenarios.

Based on the golden specification, you can generate the following for verification:

- Complete and fully connected UVM test environment including components, hdl_paths, covergroups, constraints, and illegal bins.
- Sequences for positive and negative register functionality.
- A “Makefile” to run the simulations and collect results from the simulation database, appropriate for the simulator being used (Xcelium, Questa or Riviera-PRO).

The following tests are automatically generated:

- UVM standard sequences
- Sequences for all register/field access
- Sequences for special registers such as Lock, Shadow, Virtual, Counter, Interrupt, Alias, etc.



Generated UVM Testbench

Automatic Register Verification flow chart and generated environment

Based on the golden specification, you can also generate the following code for formal verification:

- SystemVerilog properties and assertions to check the register access policies and compliance to bus protocols
- Top-level file to bind your design RTL as well as third-party design IP with the assertions.

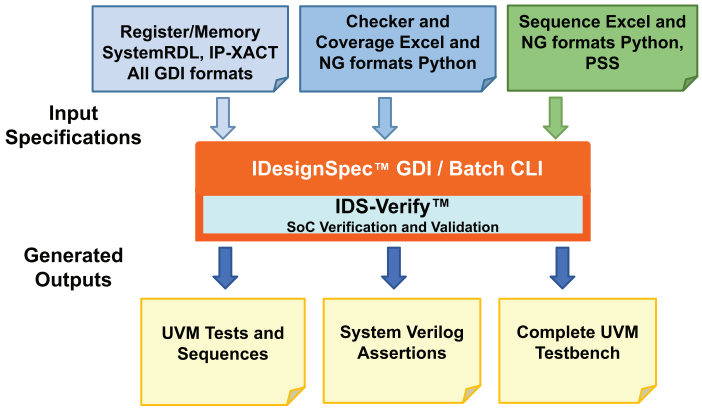
Assertions generated include:

- Concurrent assertions to check the HSI
- Assertions for special registers such as Lock, Shadow, Virtual, Alias, Interrupts, RW Pair, etc.
- Assertions to check connections at the SoC level
- Assertions generated in RTL for SystemVerilog constraints

With IDS-Verify™, we can also specify custom register verification test sequences at a high level and generate the UVM tests and environment. This saves time for IP verification. IDS-Verify™ also includes a sophisticated syntax and semantics checker for the sequence descriptions to catch common user errors.

Benefits of IDS-Verify™

- Complete UVM infrastructure (regmodel, agents, coverage collector, and functionality based checkers)
- Support for multiple buses
- Makefile for Xcelium, Questa, and Riviera-PRO simulators
- Run standard UVM tests
- Auto mirroring of registers
- Generate test sequences for registers both simple and complex
- Automatic generation of SystemVerilog properties and assertions to check the register access policies and compliance to bus protocols



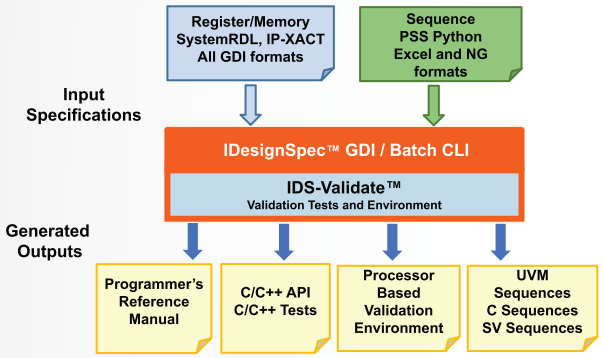
IDS-Verify™ Data Flow

Going from SystemVerilog based verification to C based validation tests is done using IDS-Validate™.

IDS-Validate™: Validation Tests and Environment

- C-tests and API are automatically generated from the register specification based on different bus access types and register properties.

- Automated tests for special registers, special register accesses, Interrupts, volatile registers, etc. with 100% out-of the box functional coverage.
- Custom sequence API can be created and hooked into the environment from a single source.
- On-chip debugging through openOCD that accesses the SweRV JTAG interface virtually.
- System emulation environment can be created around the SweRV core for dynamic applications to be built on the Zephyr OS to test the IPs.
- Centralize creation of custom portable sequences from a single specification and generate various output formats for multiple SoC teams:
  - UVM, System Verilog, C, TCL, CSV or MATLAB
  - HTML
  - Platform
  - Programmer's Reference Manual



IDS-Validate™ Data Flow

Portable Sequences

Just as you can automate the registers, you can also automate the algorithms used to program these registers. When there are thousands of registers, there are millions of ways to program them and one wrong step could cause system failure. These register programming sequences are used by multiple teams involved in verification and post-silicon validation stages. But most teams lack a unified flow for creating sequences.

Our solution unifies the creation of portable sequences from a golden specification. You can capture the sequences in Python,spreadsheet or PSS(Portable Stimulus Standard) format and generate multiple output formats for a variety of domains:

- UVM sequences for verification
- SystemVerilog sequences for validation
- C code for firmware and device driver development
- Specialized formats for automatic test equipment (ATE)
- Hooks to the latest Portable Stimulus Standard (PSS)
- Documentation outputs such as HTML and graph

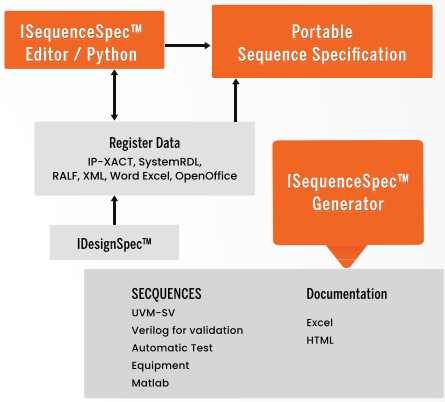
APC(AGNISYS PSS COMPILER/EDITOR)

Integration of a dedicated PSS (Portable Stimulus and Test Language) Editor. This new addition enables you to work with PSS files, create and edit portable stimulus models and tests with ease and ensures a seamless experience for engineers and testers working with this industry-standard language.

The sequence constructs include loops, if-else, wait, and switch statements to change the interfaces, specify encoding formats, deal with time-unit differences, use macros, specify variants, and use return statements to return user errors from sequences. The constructs support constrained variables for randomized sequences, and handling of indirect and interrupt registers.

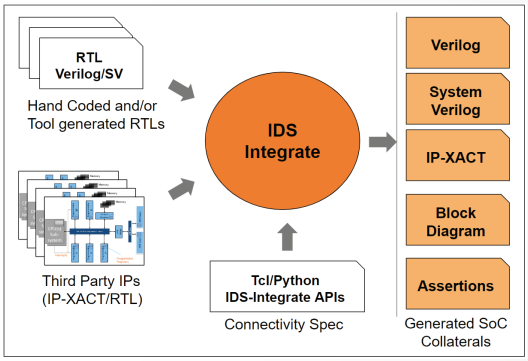
The sequences support the following:

- Read/Write on repeated components, hardware interface, signals, and external memories
- Read_check command to detect mismatch in write/read
- Sanity checking for sequences, e.g., wrong register used in sequence step, value written to read-only register, etc.
- Structures in arguments:- refers to the datatype of the argument
- Configurable data types for variables/arguments, concatenation of variables, and arrays
- Optimized firmware sequences by Read-Modify-Write



IDS-Integrate™: Smart SoC Assembly

Manually hooking up hundreds or thousands of blocks into a top-level SoC design is a tedious and error-prone process. Agnisys® has a solution to apply specification automation to SoC-level assembly and interconnection.

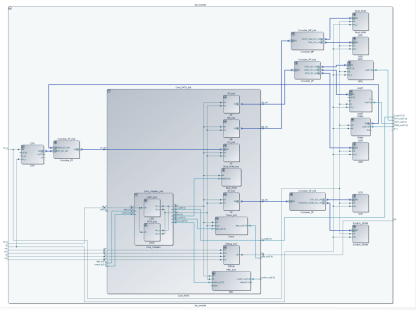


Supported Input Formats

- IP-XACT (2009, 2014, 2022)
- Verilog (1995, 2001)
- System Verilog

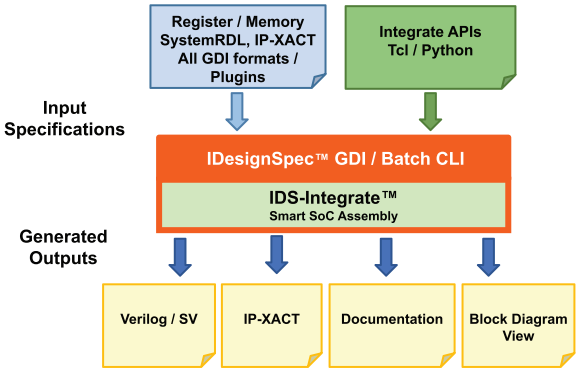
Generated Outputs

- RTL (Verilog, System Verilog)
- Schematic Diagrams
- IP-XACT (2009, 2014, 2022)
- C/C++ Header (requires IDS)
- SystemVerilog Assertions



- A flexible and customizable environment for design assembly
- Helps create, package, integrate, and reuse IPs and SoC/FPGA
- Supports the latest IP-XACT standard and popular vendor extensions in addition to RTL (Verilog/SystemVerilog) and other IDesignSpec™ Suite supported formats
- Smart assembly, as it generates components such as aggregator, bridges, and muxes, as needed
- Flexibility to abstract ports to efficiently capture connections
- Ability to easily specify tie-offs, intentional opens, and other special cases, where needed
- Ability to easily reconfigure an IP as needed or update the current version with a later version
- Built in design rule checks to validate design connectivity before generating RTL
- Generating SystemVerilog assertions for connectivity checks using formal tools

- Providing an intuitive graphical view of blocks and their connectivity with simple navigation through the chip hierarchy
- TGI support
- Support for automatic generation of multi-initiator and multi-target crossbar with arbiter as needed
- Glue logic support for various logic gates
- Schematic editor for interactive GUI editing of designs
- Support for passthrough connections
- Ability to specify and generate selected logic using `define-`ifdef`ifn-def pairs
- IP Packaging: Generates IP-XACT output for top design along with components, bus definitions, abstraction definitions, filesets, etc.
- Generates catalog files for the top design listing all associated collat-
- erals centrally in the IP-XACT catalog in version 2014 onwards



IDS-Integrate™ Data Flow

Specialized Solution Packages

IDesignSpec™ GDI for FPGA

IDesignSpec™ GDI includes support for your FPGA designers. IDesignSpec ™ GDI can read the specifications for pre-defined IP blocks provided by FPGA vendors for integration into larger designs. IDesignSpec™ GDI generates UVM models, C/C++ headers, and documentation for these IP blocks automatically. IDesignSpec™ GDI also generates target scripts for use in the FPGA vendor implementation tools. Agnisys® directly partners with both Xilinx and Intel to support your team.

IDS Tool Qualification Kit for Functional Safety (ISO 26262)

The IDesignSpec™ Suite of software products and development flow have been certified by the internationally recognized testing organization TÜV SÜD as achieving the stringent tool qualification criteria defined in the ISO 26262 functional safety standard for road vehicles. The certificate also covers the IEC 61508 industrial functional safety standard. The process of certification by TÜV SÜD included an audit of the Agnisys ® safety management, tool development, and supporting processes.

ABOUT AGNISYS®

Agnisys®, Inc. is the leading supplier of Electronic Design Automation (EDA) software for solving complex design and verification problems for system development. Its products provide a common specification-driven development flow to describe registers and sequences for system-on-chip (SoC), Field Programmable Gate Array (FPGA), and Intellectual Property (IP) enabling faster design, verification, firmware, and validation. Based on patented technologies and intuitive user interfaces, its products increase productivity and efficiency while eliminating system design and verification errors. Founded in 2007, Agnisys® is based in Boston, Massachusetts with R&D centers in the United States and India.

US CORPORATE OFFICES

East Coast
75 Arlington St. Suite 500
Boston, MA – 02116

+1 (855) VERIFYY [+1 (855) 837-4399]
+1 (866) 927-3653 (fax)