Active-HDL 15

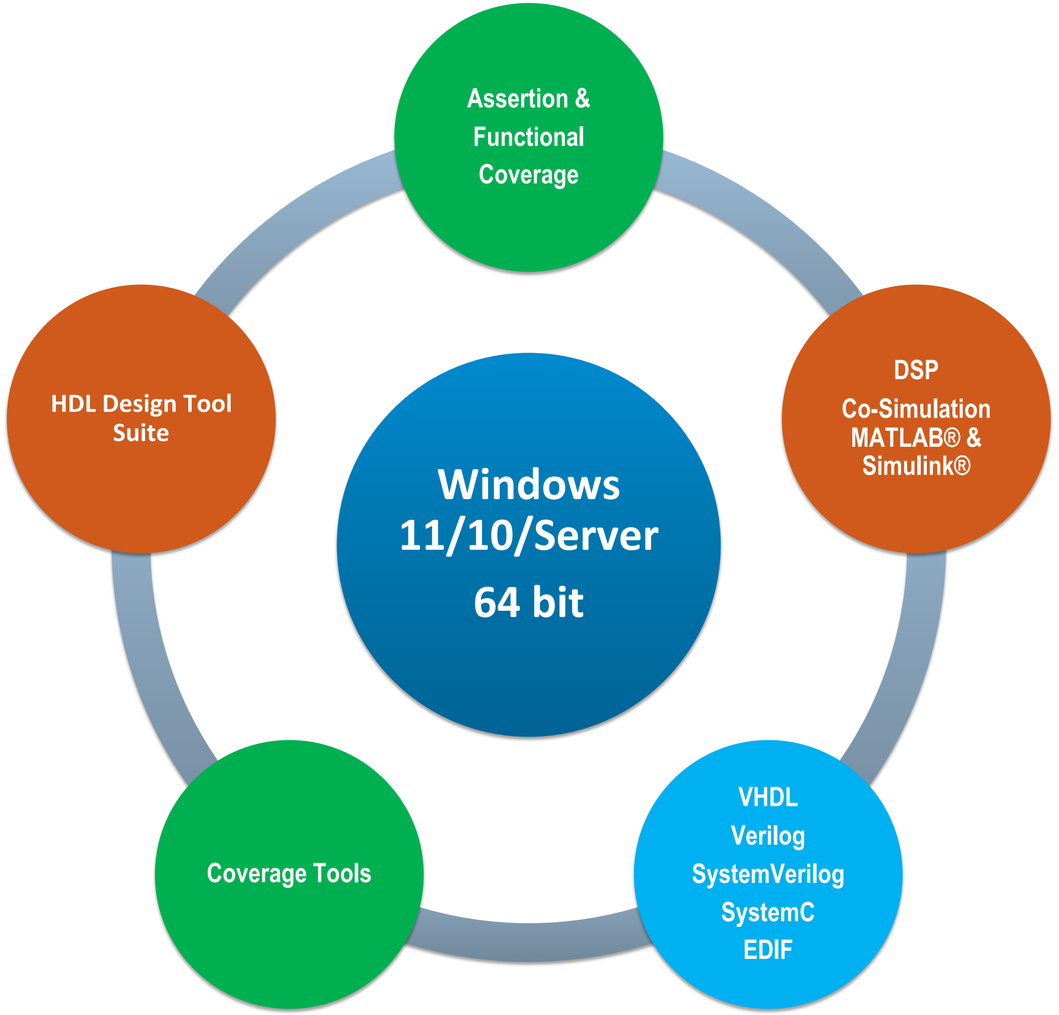
Overview Presentation

Integrated FPGA Design & Simulation

Active-HDL is both a ***HDL Design tool*** & a ***Mixed Language Simulator*** in one integrated Multi-FPGA (AMD®, Altera®, Lattice®, Microchip®) and design entry and simulation environment

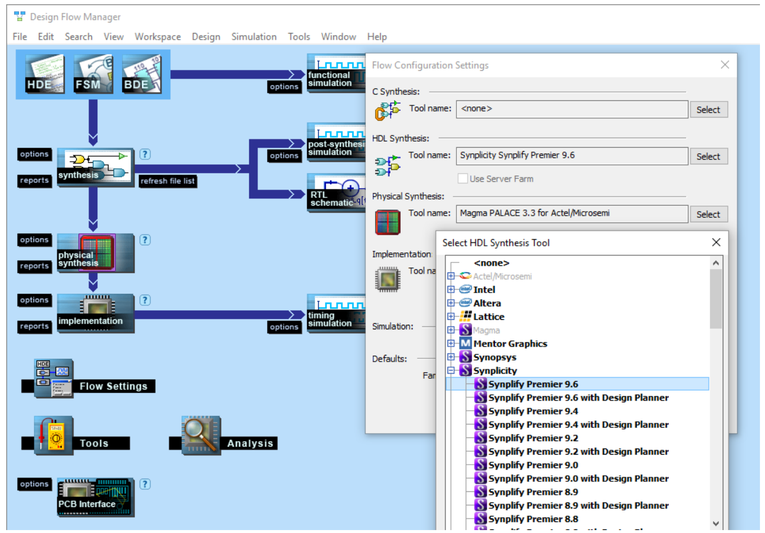
Included Technology

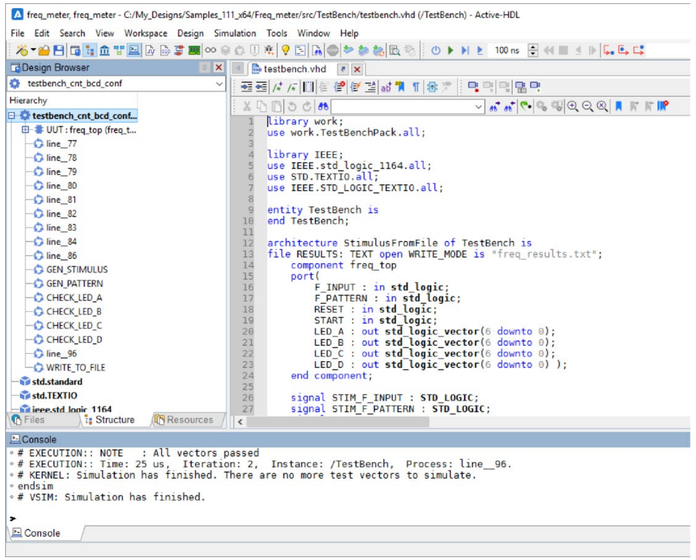
- Team-based Project Management
- HDL Design Tool Suite
- Mixed Language RTL Simulator
- VHDL, Verilog, SystemVerilog, SystemC
- Coverage Tools Bundle (Option)
- Assertions & Functional Coverage (Option)
- DSP Co-Simulation MATLAB & Simulink
- PCB Interface
- Documentation Generator



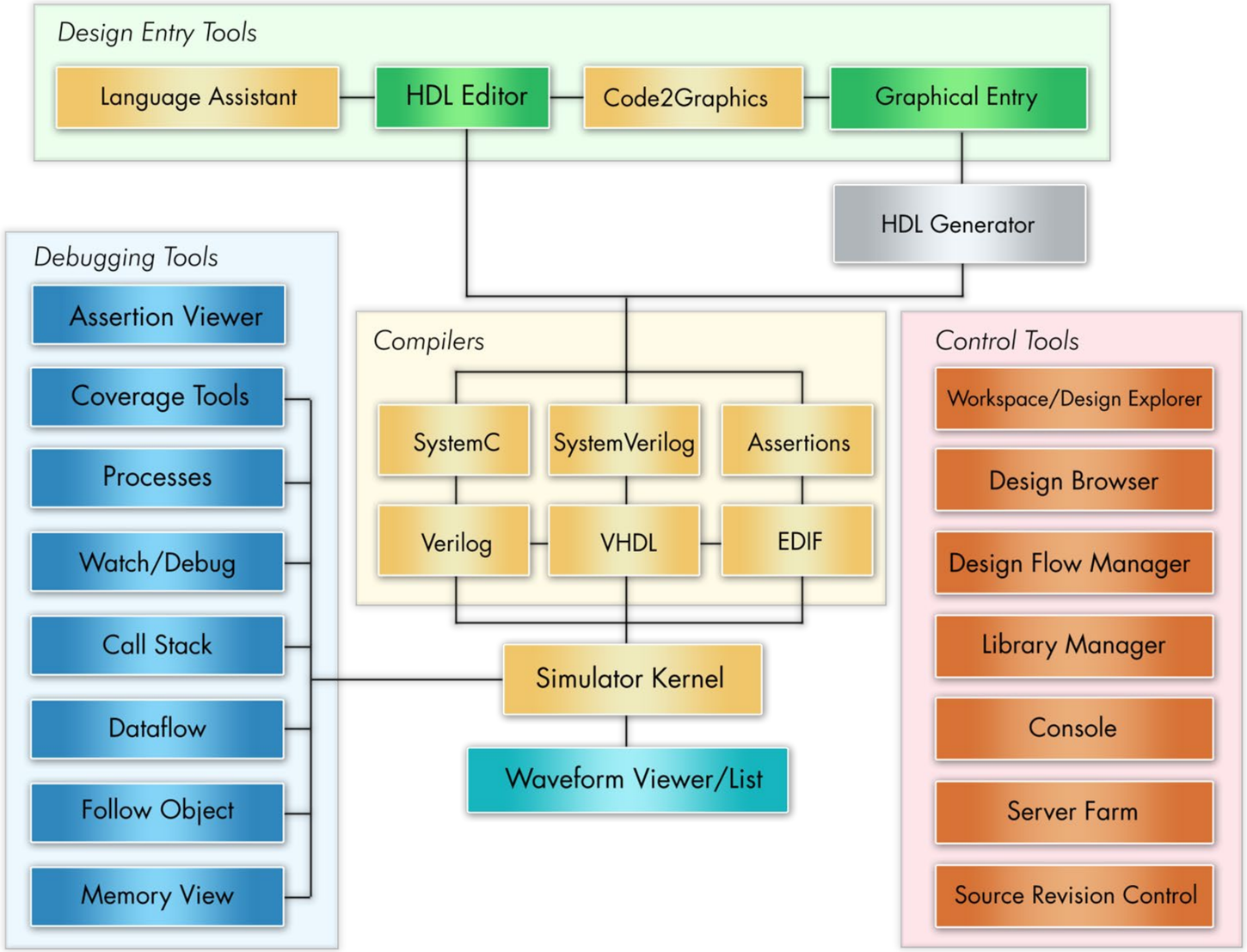
Aldec Active-HDL in a nutshell

- **Common-Kernel Mixed Language Simulator**
- **Languages: VHDL, Verilog, SystemVerilog, SystemC & EDIF**
- **HDL Design Tools:** Design entry, Design Creation, Code2Graphics™, Block and State Diagram, Waveform editor, stimulus generation, Language templates & auto-complete, scripting.
- **Design Flow Manager:** use popular third-party tools throughout the design flow within the same FPGA environment.
- **Debugging:** Code execution tracing, Waveform/Compare, Memory Viewer, Xtrace, Advanced Dataflow and Profiler.
- **Coverage:** Code Coverage, Toggle & Functional Coverage.
- **Additional Interfaces:** DSP/HDL algorithm MATLAB® and Simulink® Interfaces & Zuken CADSTAR PCB Design
- **Project Documentation** Entire workspace or single file export to PDF/HTML.
- **Assertion and Coverage** SystemVerilog and PSL support. Dedicated Assertion viewer, coverage, breakpoint editor.



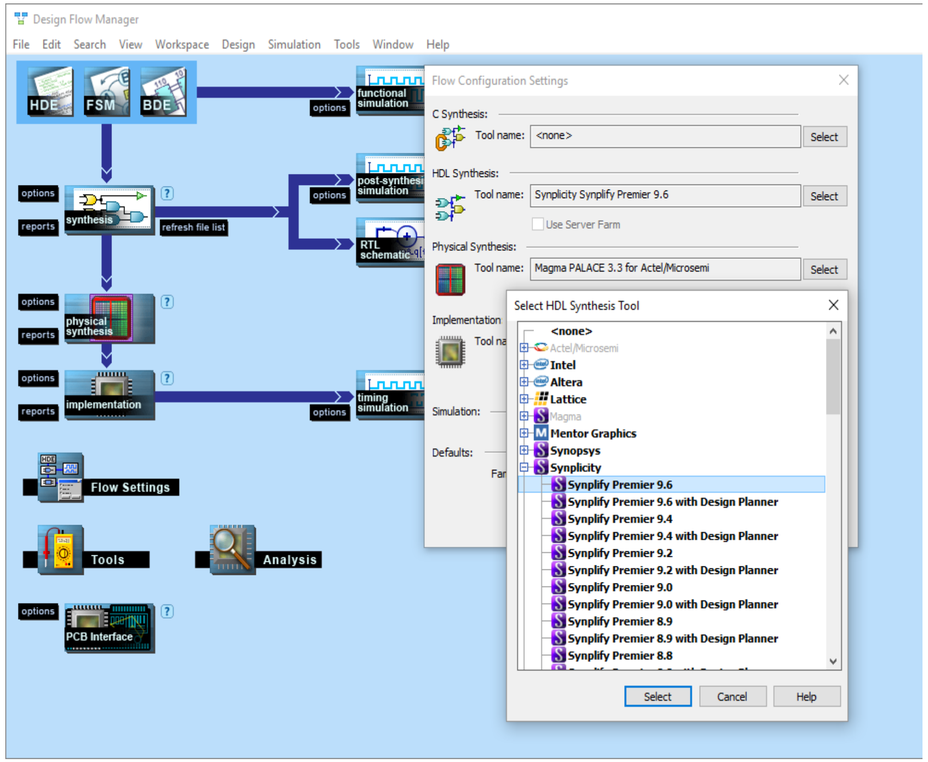


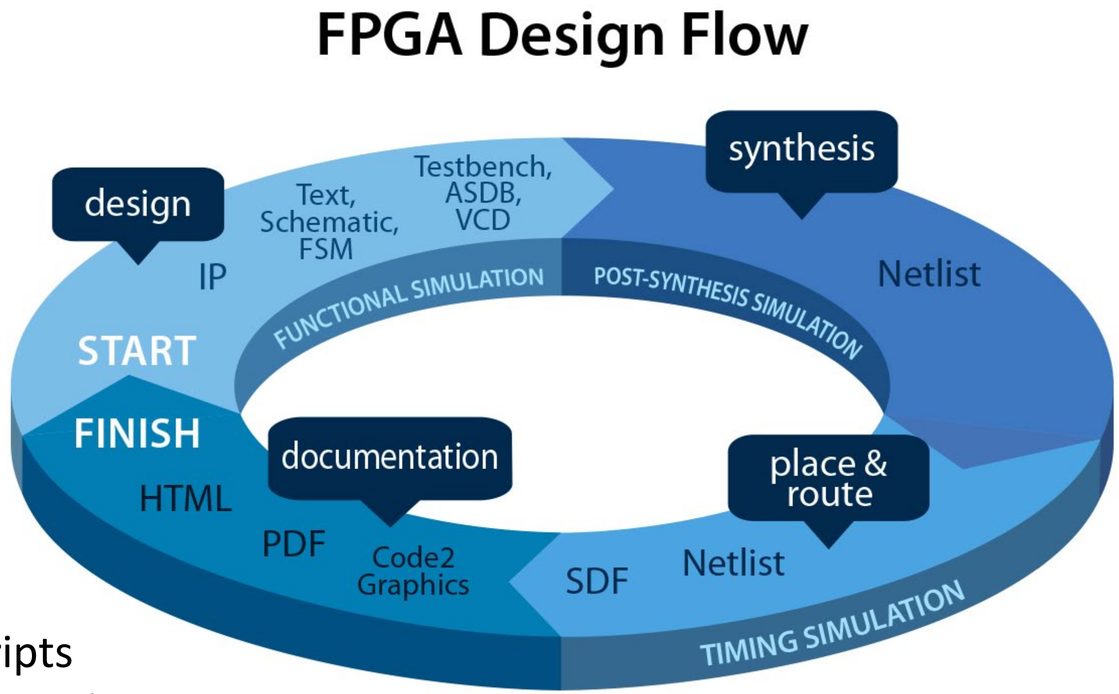
FPGA Vendor Independent IDE for RTL-User



Design Flow Manager

- Design Flow Manager interfaces to 200+ different 3rd party tools
- Manages the HDL, C and Physical Synthesis Tools
- Runs the implementation for any FPGA vendor

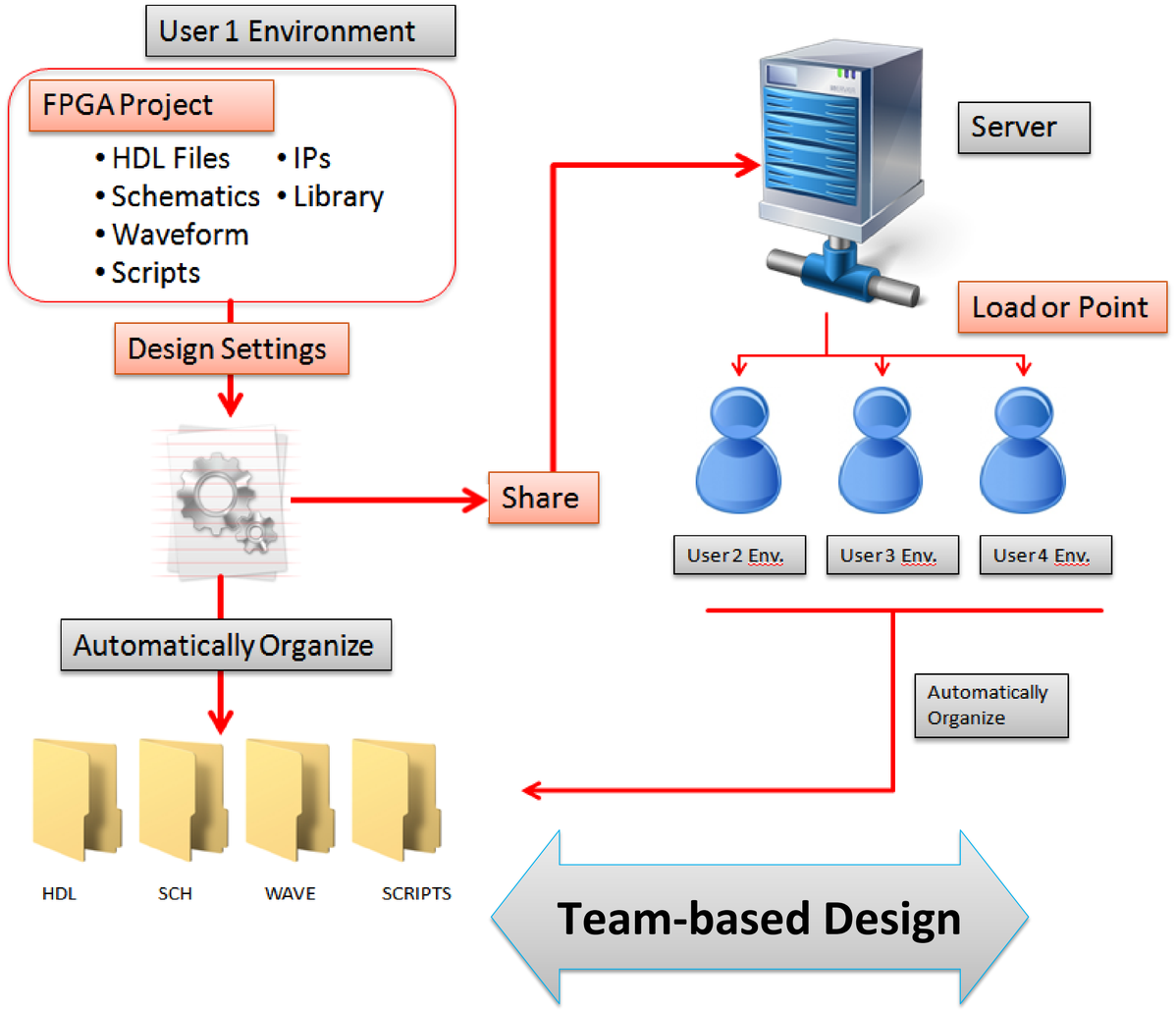




- Generates TCL scripts for advanced automation
- Runs the simulation at all stages of design
- Invokes external analysis tools provided by silicon vendors

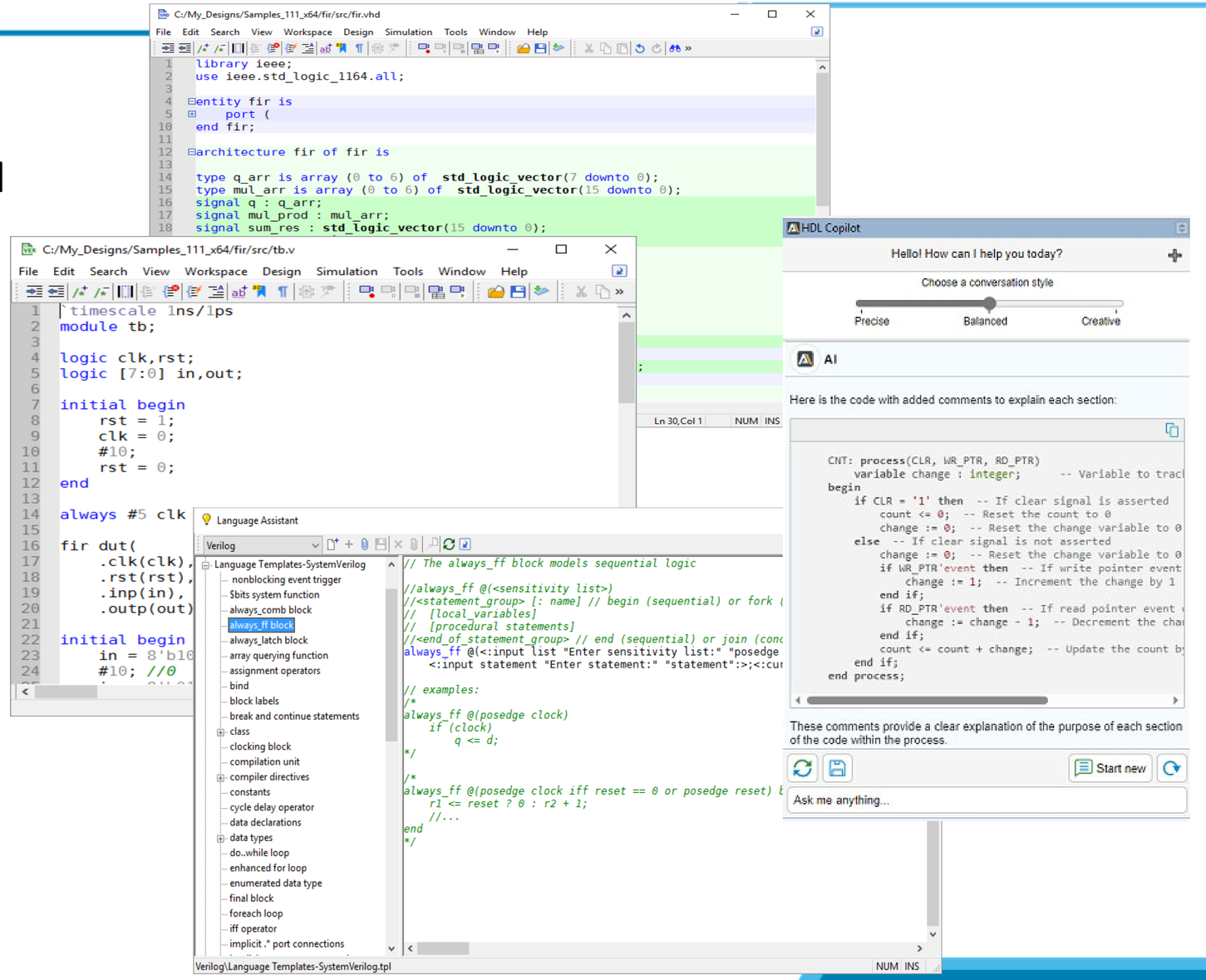
User-defined Design Management

- Separate HDL files from other types of files, e.g. schematics, text, waveform and scripts
- Easily maintain and adhere to design structure required by entire team or a company
- Create directory structure that is compatible to other synthesis and implementation tool



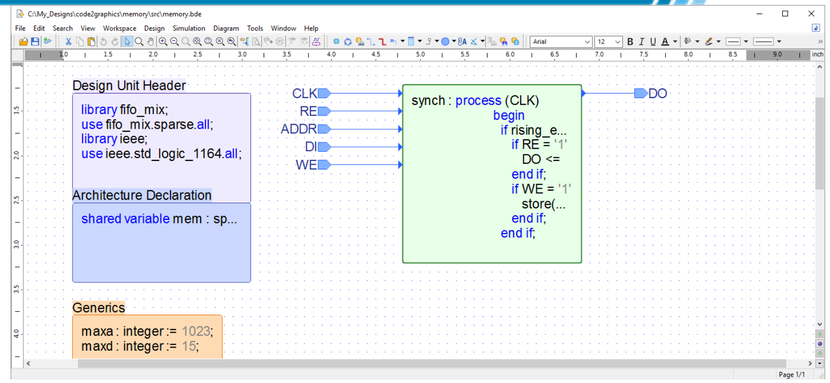
Advanced HDL Editor

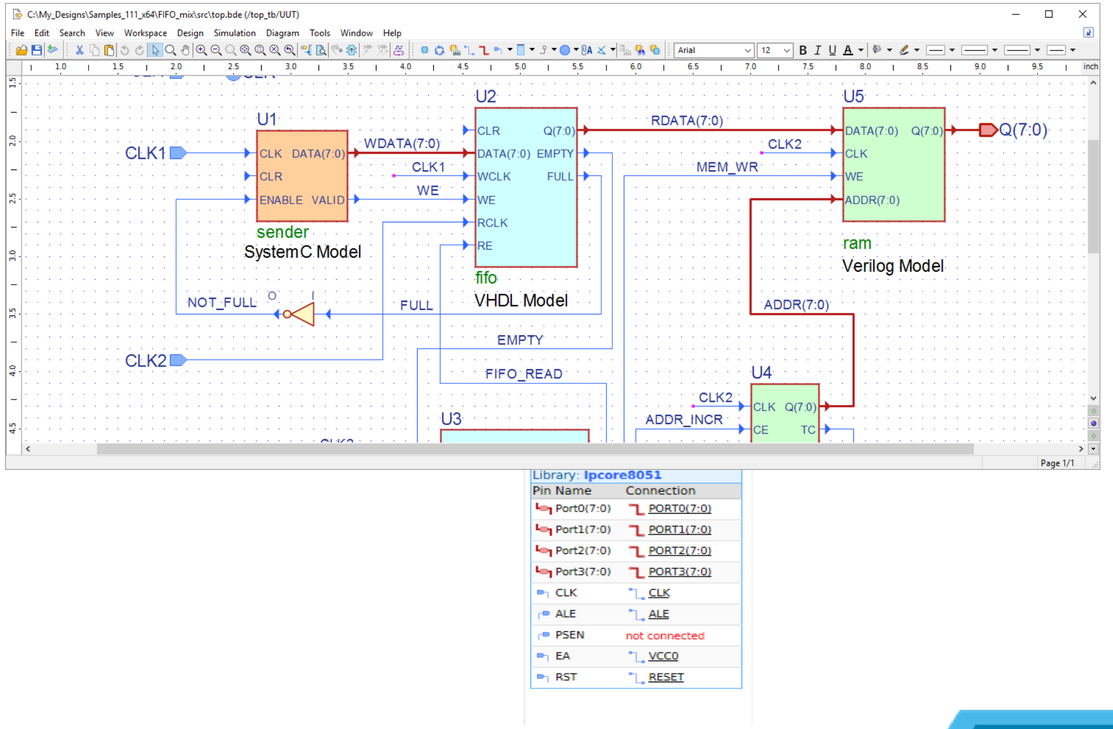
- Code analysis and navigation tools
- Action Recorder for often repeated operations
- Keyword and Signal name auto-completion
- Automatic structure generation for enhanced legibility
- Built-in customizable language assistant
- Source code auto-formatting
- Advanced Find, Find in Files and replace, Column Selection
- Navigation Bookmarks
- Ability to interface to third party text editors
- OpenAI helper included



Block Diagram Editor

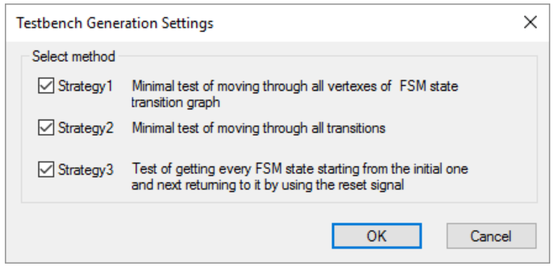
- Multi-page hierarchical block diagrams
- Multidimensional arrays and record signals supported
- Bottom-up and top-down design methodologies supported
- Allows mixed structural and behavioral elements
- Cross probing with generated code
- Handles mixed HDL designs
- Customizable design rules checking
- Imports schematics
- Customizable symbols

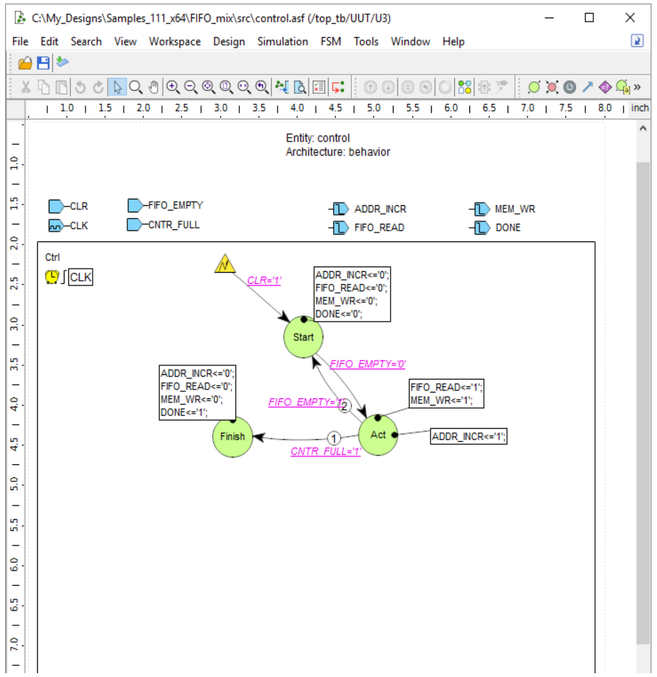


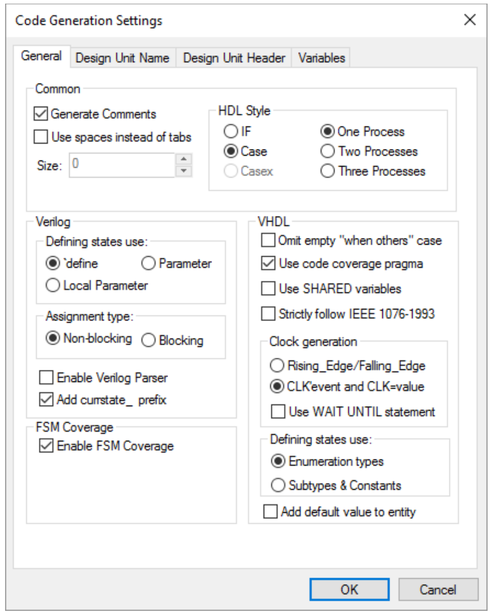


Finite State Machine Editor

- Multiple State Machines on a single diagram
- Full-Moore machines support
- Hierarchical states and junctions provided for legibility
- Delay states simplify control of machine timing
- Advanced code generation settings
- Automatic testbench generation
- Graphical FSM debug
- FSM Coverage included

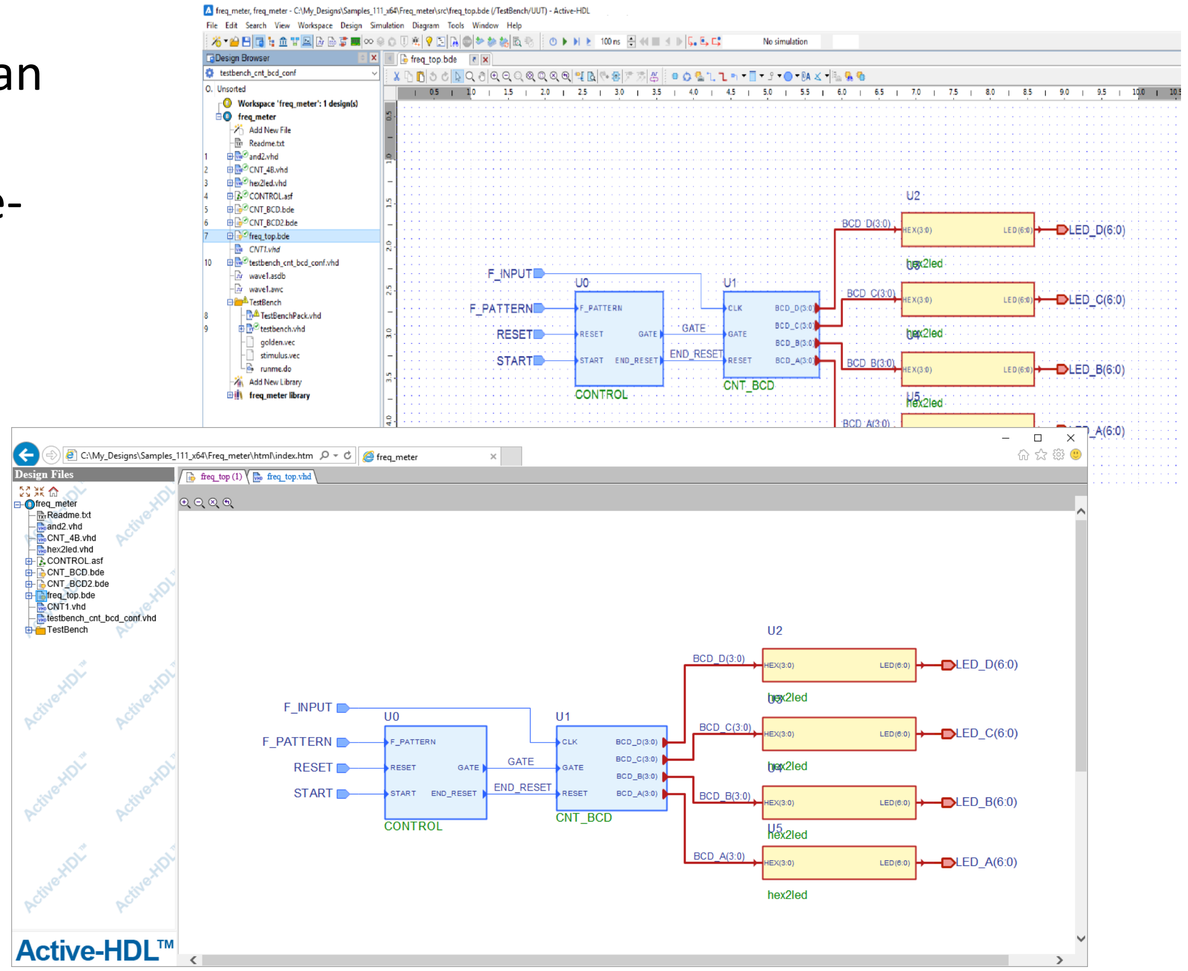






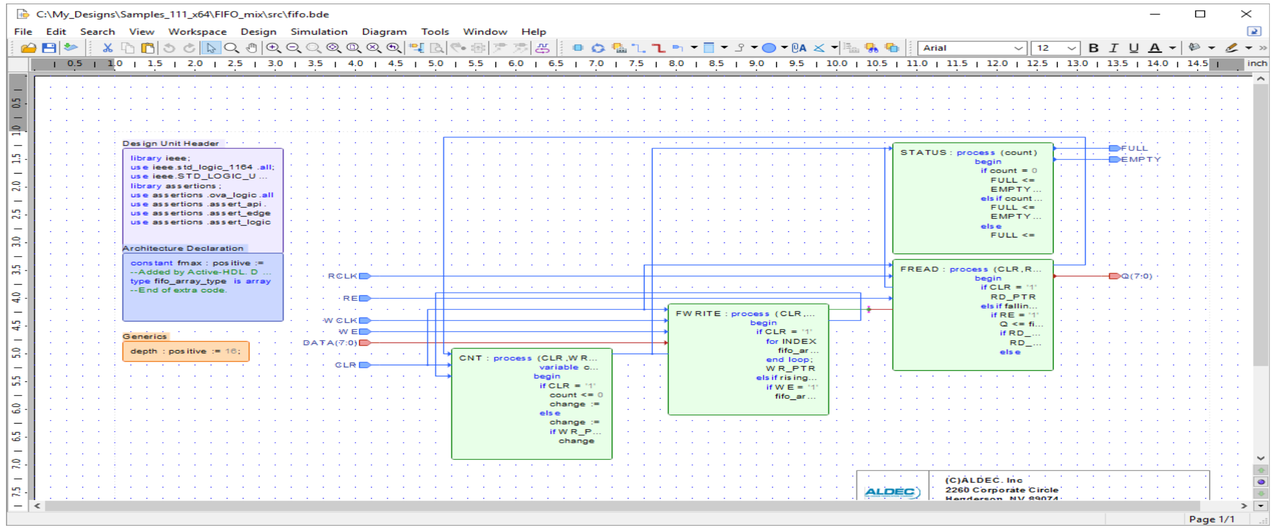
Project Documentation

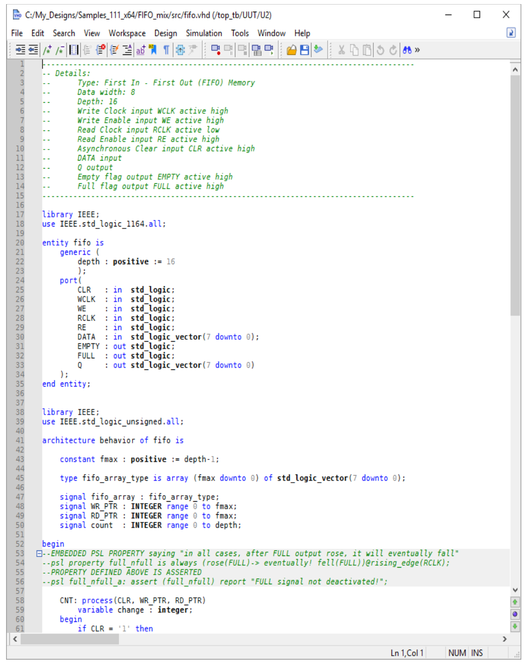
- Entire workspace, individual designs or files can be printed or exported to vector PDF
- The HTML document closely resembles Active-HDL environment layout
- Vector graphics and other advanced PDF options available
- Graphical Design Entry documents can be accompanied by generated code
- Ideal solution for design documentation



Code2Graphics™

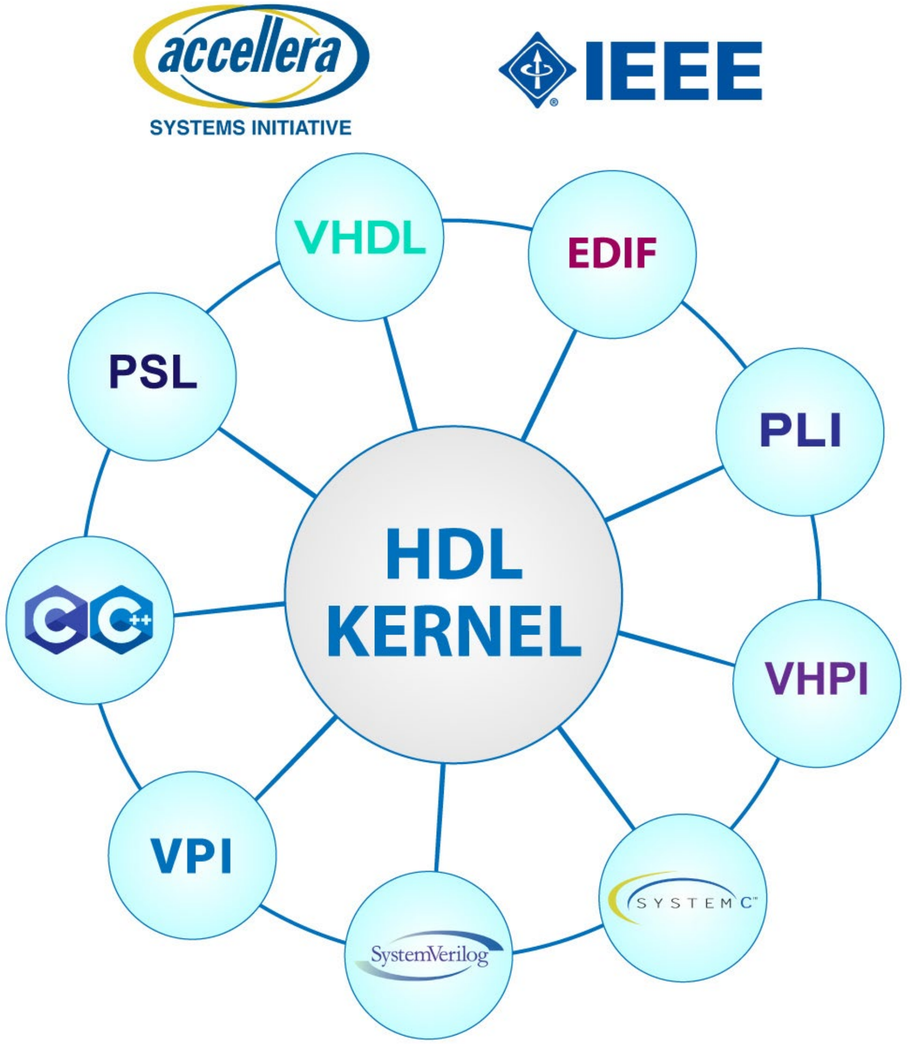
- Converts HDL code into graphical representation:
  - Block Diagrams
  - Finite Stae Machines
  - Editor files
- Creates perfect documentation for HDL designs
- Helps with debugging of large designs





Common Kernel Simulator

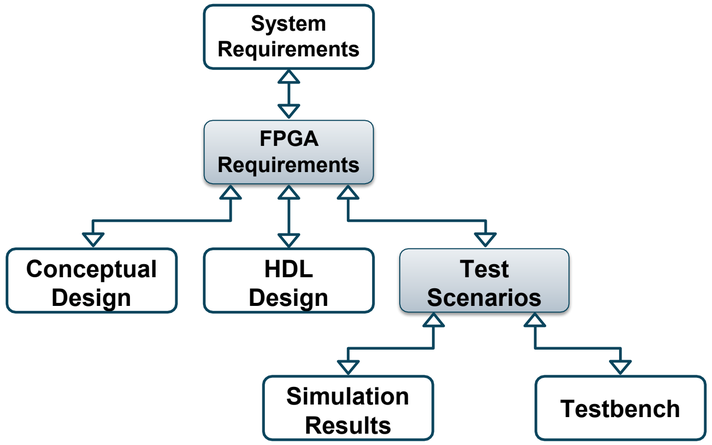
- **VHDL, Verilog, EDIF, SystemC** and **SystemVerilog**
- Ultra Fast Verilog Simulation
- Strict IEEE Standards Adherence:
  - **VHDL: IEEE 1076-1993 Standard, IEEE 1076™-2002 VHDL Standard, IEEE 1076™-2008 Standard and IEEE 1076™-2019 Standard.**
  - **Verilog: IEEE 1364-1995 Standard, IEEE 1364-2001 Standard, IEEE 1364-2005 Standard IEEE 1364-2005 Standard.**
  - **SystemVerilog: IEEE Std. 1800™-2012**
- **PSL** embedded in **VHDL**
- **VHPI** interfacing to **C/C++** code
- **Verilog PLI** (Programming Language Interface) and **VPI** (Verilog Procedural Interface) are supported as described in the IEEE 1364 Standard
- Performance Optimizations (Verilog and VHDL)
- Extensive debugging while strictly adhering to IEEE Standards.

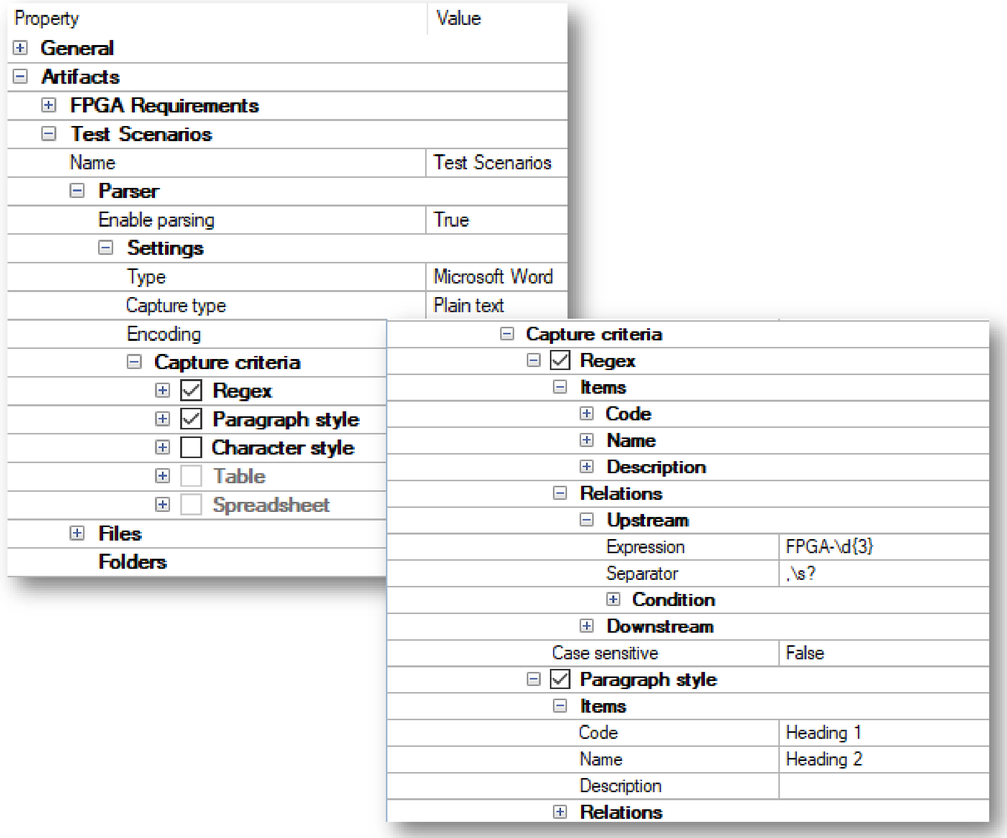


Spec-TRACER™

Traceability Capture and Reporting

- Document Types:
  - Microsoft® Word
  - Microsoft® Excel
  - Text/design files
- Capture methods:
  - Regular expression
  - Document style
  - Spreadsheet/table column
- Traceability Matrices Report
  - Upstream and Downstream coverage analysis
  - Traceability links consistency verification
  - Microsoft® Excel and CSV





FPGA Requirements			Test Scenarios		
Code	Name	File	Code	Name	File
FPGA-001	Name of fpga-001	fpga_requirements.txt	TST-001	Name of test1	test scenarios.txt
FPGA-002	Name of fpga-002	fpga_requirements.txt	TST-003	Name of test3	test scenarios.txt
FPGA-003	Name of fpga-003	fpga_requirements.txt			

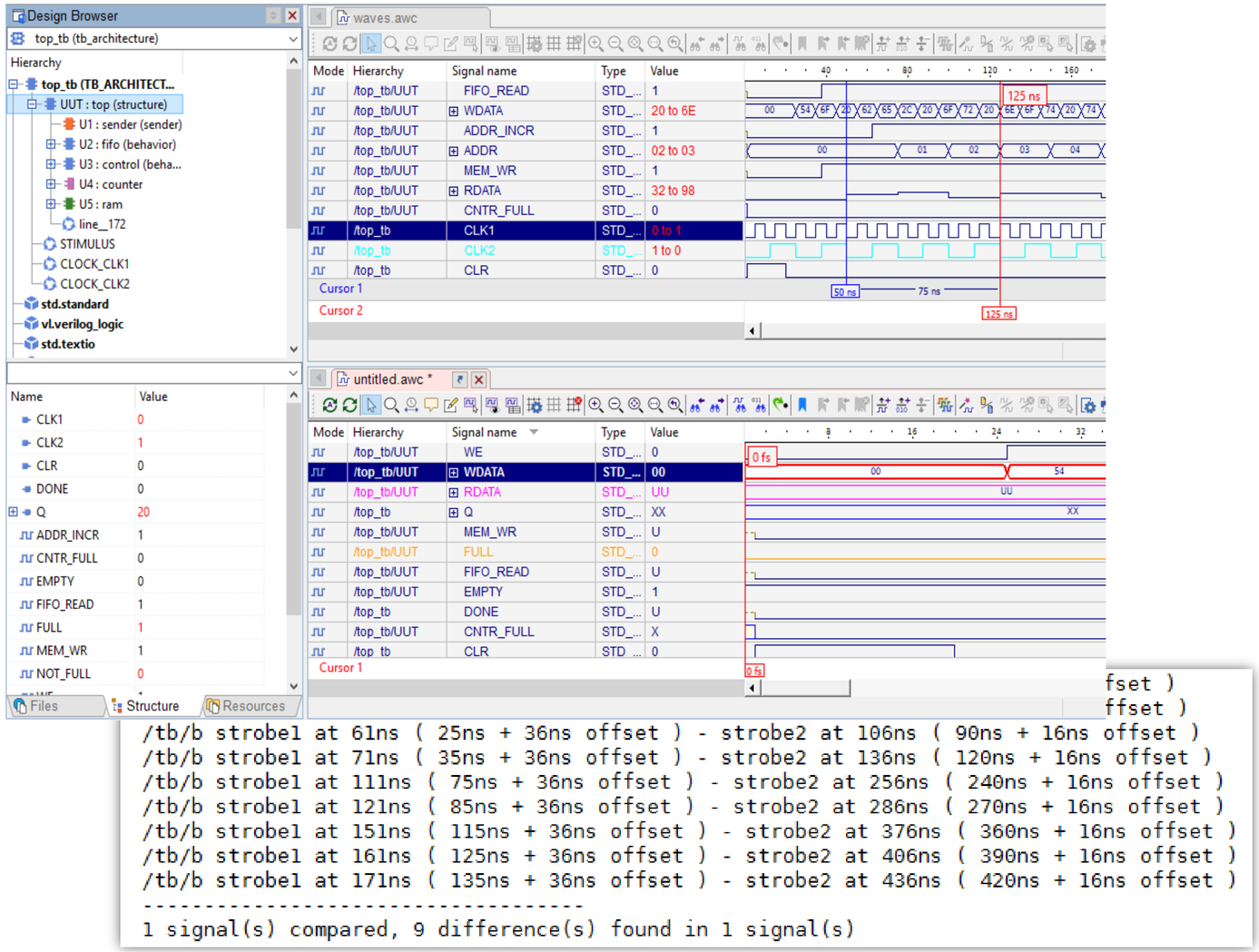
ALDEC® THE DESIGN VERIFICATION COMPANY

HDL Source Encryption / IP Protection

- Support for the latest industry standards:
  - **IEEE 1735** Version 1 Recommendations
  - **IEEE 1735-2014** Version 2 using delegated rights
  - Encryption methods defined in Verilog IEEE Std 1364-2005 and VHDL IEEE 1076-2008 Standards
  - ***Simulating encrypted files requires no action at Active-HDL user's side*** (given IP vendor encrypted the files for target simulation tool)
  - Decrypts the source on-the-fly, leaving no traces that could compromise the security of encryption
  - Supports DES, 3DES, AES and Blowfish encryption algorithms
  - Up to 256 bit encryption keys (AES)

Waveform Viewing and Editing

- Waveform Viewer for fast handling of large simulation databases
- Instant zooming and scrolling
- Multiple cursors and bookmarks
- Waveform comparison
- Sort signals and browse by differences in two waveform files.
- Export to VCD (Verilog and VHDL)
- Embedded List Window displays the object values in an ASCII format
- User-friendly Waveform Editor for manipulating signal data
- Hand-drawing stimulus for interactive simulation
- Preparation of data for automatic testbench generation
- Strobe Comparison command to compare signals probed by different strobe signals



Debug

Standard Debugging Tools:

- ✓Waveform Viewer
- ✓Interactive Code Execution Tracing
- ✓Breakpoint Management
- ✓Code Browser

Each FPGA vendor offers something but all different

Functional Simulation **Post-synth Simulation** **Timing Simulation**

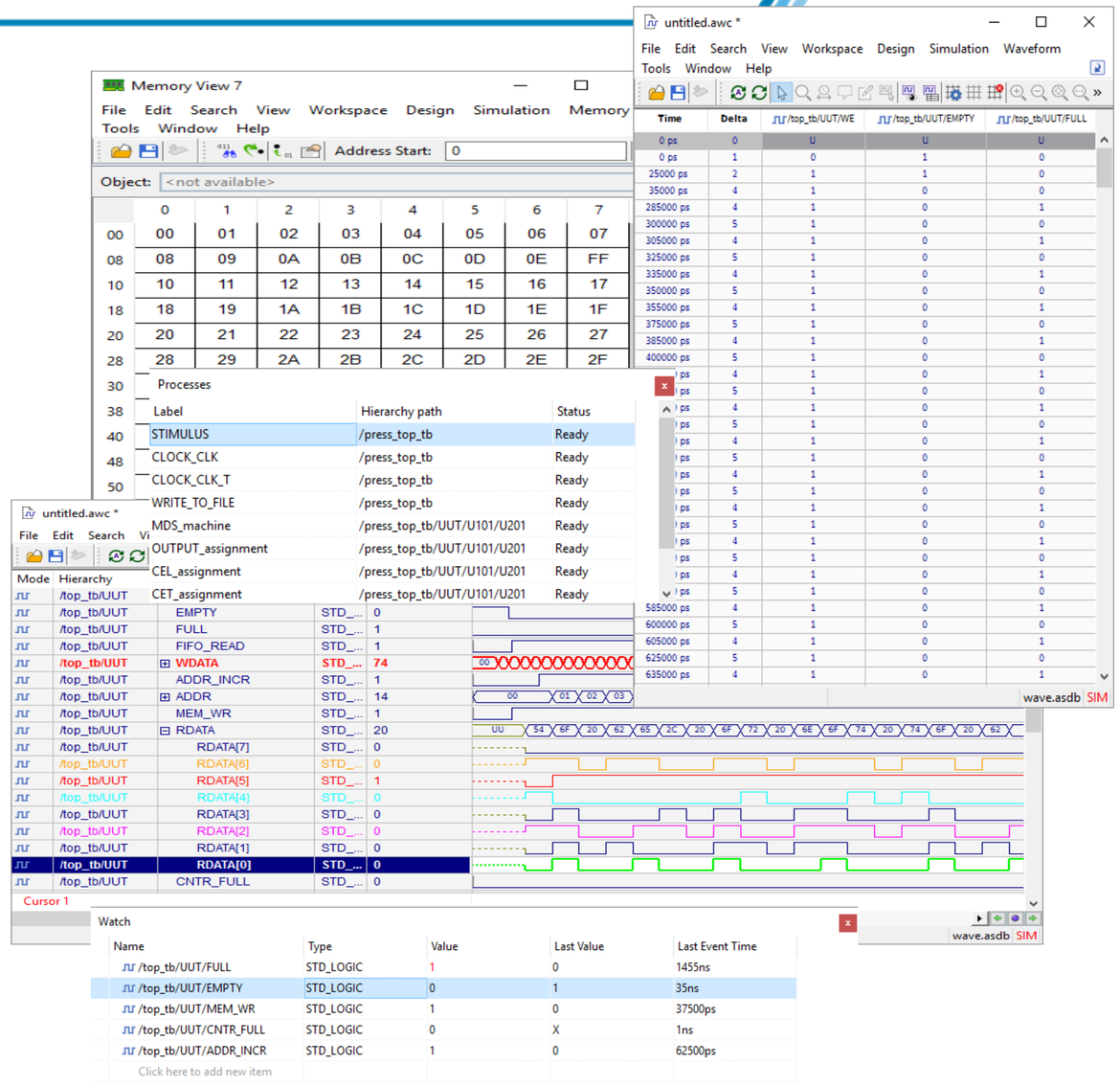
Advanced Debugging Tools:

- ✓XTrace
- ✓Advance Dataflow
- ✓Code Coverage
- ✓Static Rule Checking

Aldec provides all tools which are vendor independent and available as perpetual license

Debugging Tools

- View simulation results in
  - Waveform Viewer
  - List Viewer
  - Memory View
  - Watch
  - Advanced Dataflow
- Trace code execution with
  - Processes
  - Call Stack
- Graphical Debug
  - FSM
  - BDE

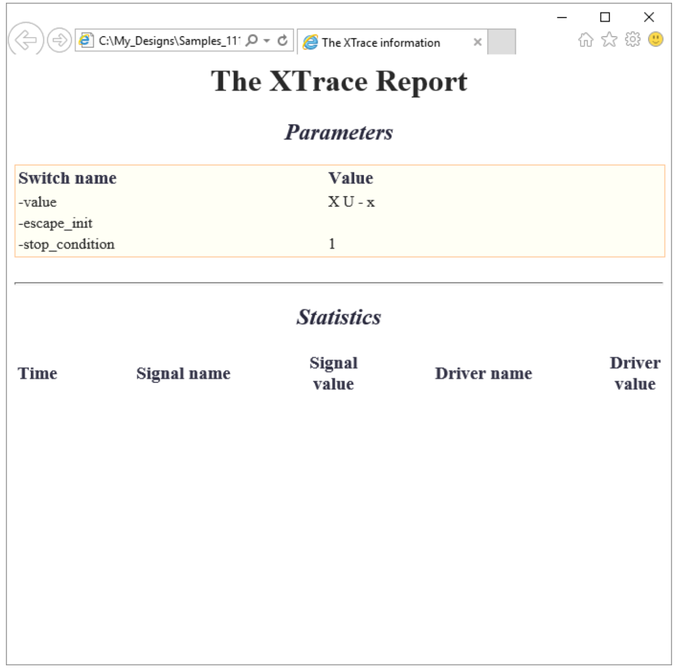


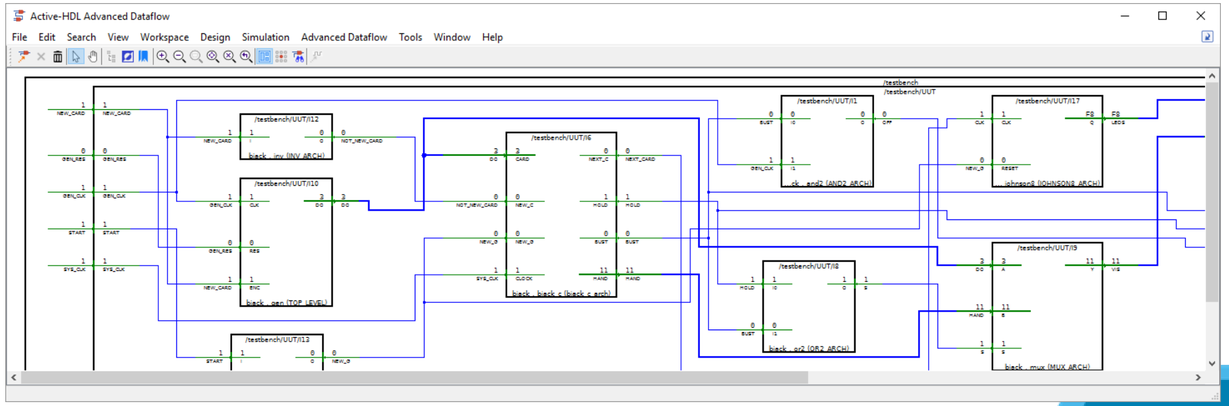
Stimulus

- **Graphical Stimulus Creation**
  Convenient GUI window (**Stimuli Window**) for creation of force stimulus for use in the waveform viewer.
- **Force command**
  The syntax of force command provides two kinds of stimuli based on file contents - custom and text.
  - When a text file stimulus is used, the force command reads the values of signals to be forced from a plain text file.
  - The custom stimulus can read test vectors from a number of file types:
    - Aldec Simulation Database (.ASDB)
    - Custom text stimulus (.TXT)
    - Value Change Dump (.VCD)
    - MATLAB Binary Data Files (.MAT)
- **Force Capabilities Inside HDL Code**
  The functionality of the force command is available not only in macros but also directly in VHDL or Verilog code (VHDL procedure force and Verilog task $force).

Advanced Dataflow and XTrace

- Allows exploration of design in graphical representation
- Shows the drivers-readers connection and simulation values
- The XWay feature finds the source of unknown value
- Gray mode provided for better legibility
- The XTrace feature allows detection of irregular values when they happen
- Combined with advanced dataflow creates powerful debugging tool





Coverage

- Code coverage checks how many times each statement is executed
- Branch/expression/condition coverage provide detailed statistics on execution of conditional statements and assignments
- Path Coverage collects information about the program execution and program sequences (program paths)
- Toggle coverage reports design activity in terms of changes in signal values
- FSM coverage allows to identify not visited states, transitions and sequences of states
- Functional coverage allows to collect coverage data from OSVVM code and SystemVerilog (Verification)

```
44                  end
45              endcase
46          endgenerate
47
48          base64_enc dut(.clk(clk),.reset(reset),.fd_in(fd_in),.fd_out(fd_out),.complete(dc
49
50          // clock generator
51          always
52  1           #25 clk = ~clk;
53
54          // main
55          initial begin
56  1           #100 reset = 1'b1; // reset done
57              fork
58  1               @(posedge done);
59  1               #100000;
60              join_any
61  1           if (done)
62  1               $info("Encryption done");
63              else
64  0               $warning("Encryption failed - timeout");
65  1           $fclose(fd_in);
66  1           $fclose(fd_out);
67  1           #10 $finish;
68          end
69      endmodule
```

CUMULATIVE SUMMARY

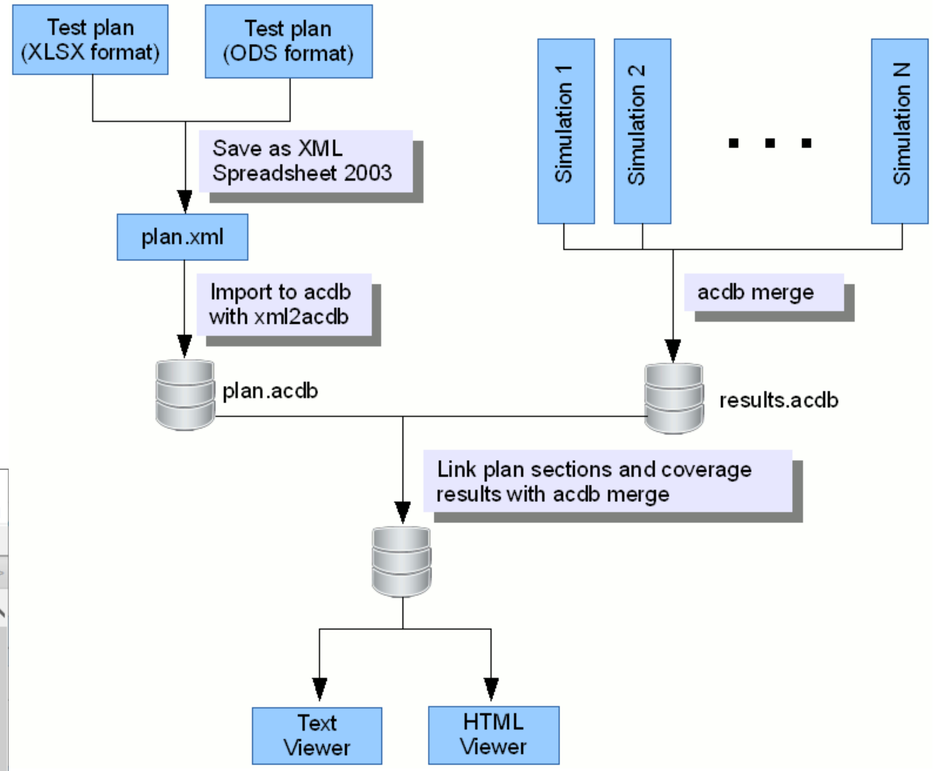
Coverage Type	Weight	Hits/Total
Statement Coverage	1	73.972%
Statements		108 / 146
Subprograms		2 / 4
Branch Coverage	1	70.247%
Branch paths		85 / 121
Branches		20 / 25
Covergroup Coverage	1	87.500%
Types		0 / 1
Expression Coverage	1	100.000%
Expression bins		2 / 2
Expressions		1 / 1
Condition Coverage	1	92.000%
Condition bins		23 / 25
Conditions		6 / 8
Toggle Coverage	1	100.000%
Toggle bins		26 / 26
Signal bits		13 / 13
Assertion Coverage	1	100.000%
Assertions		5 / 5
Covers		3 / 3

FSM	Hits/Total
▾ fsm_partsel	75.000%
▾ States	3/3
├ S1	6
├ S2	5
└ S3	5
▾ Transitions	3/6
├ S1->S2	5
├ S1->S3	0
├ S2->S1	0
├ S2->S3	5
├ S3->S1	5
└ S3->S2	0
▾ fsm_concat	66.666%
▾ States	4/4
├ S4	4
├ S5	4
├ S6	4
└ S7	4
▸ Transitions	4/12

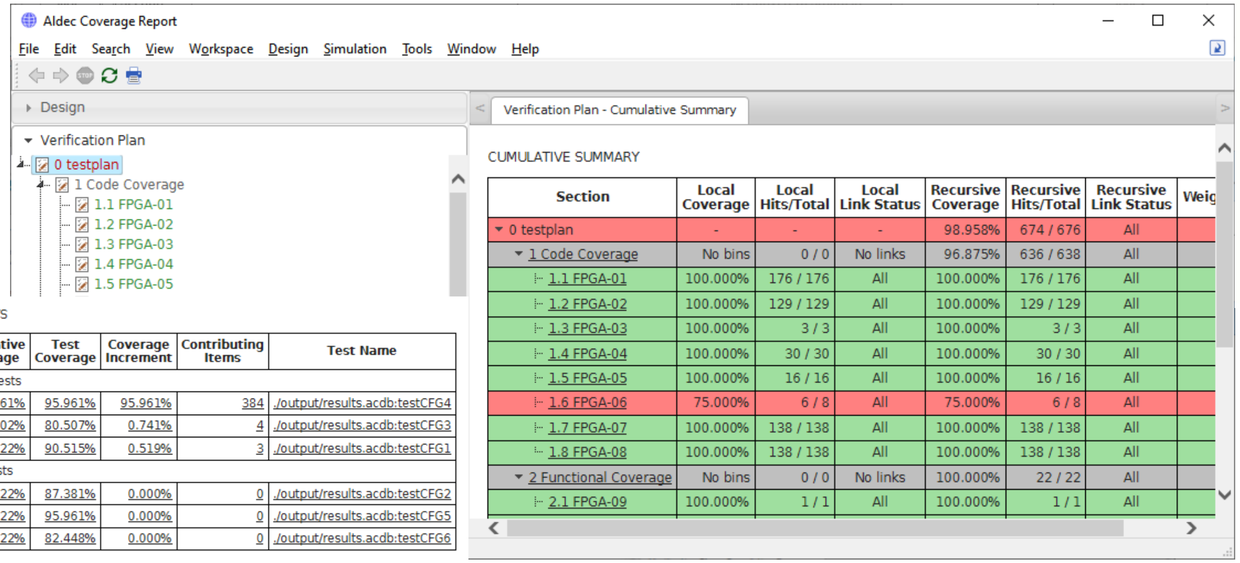
CUMULATIVE INSTANCE-BASED COVERAGE: 89.102%
COVERED INSTANCES: 1 / 4
FILES: 3

Verification Plan

- Enables **validating** the **verification process** with a **test plan**
- It allows users to link the **requirements** from **test plan** to **coverage results** to see if verification goals have been reached or not
- Allows user to **rank** tests to select minimal testsuit for **regression tests**



Verification flow with a user-defined test plan



Extensive list of Verification Libraries Support

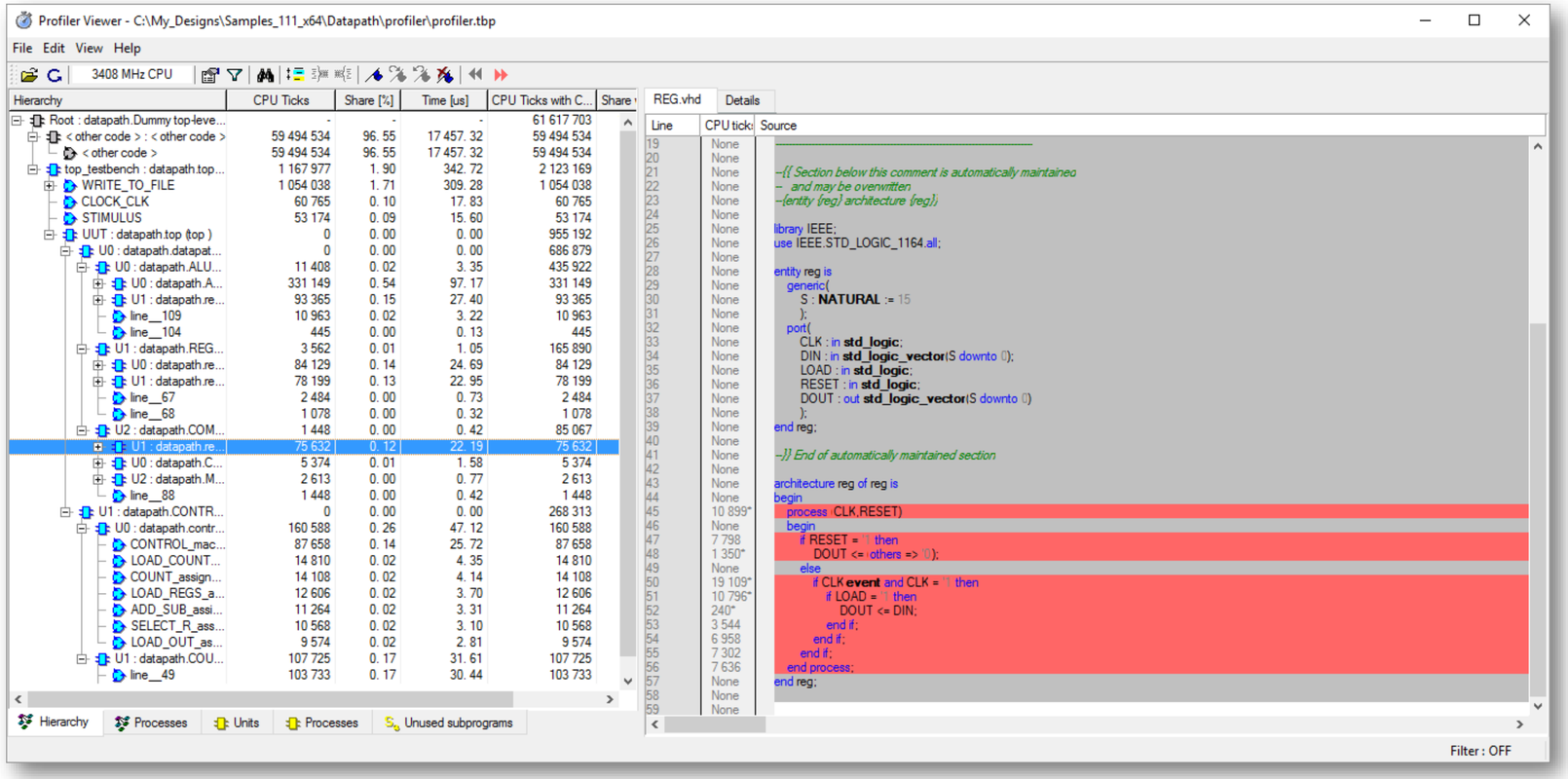
Active-HDL compiles and simulates HDL verification constructs. This provides the ability to simulate using advanced verification methods like functional coverage and constrained randomization.

Active-HDL supports followings verification libraries:

- UVM 1.1d, 1.2, 1800.2-2017, 1800.2-2020
- OSVVM
- UVVM
- VUnit

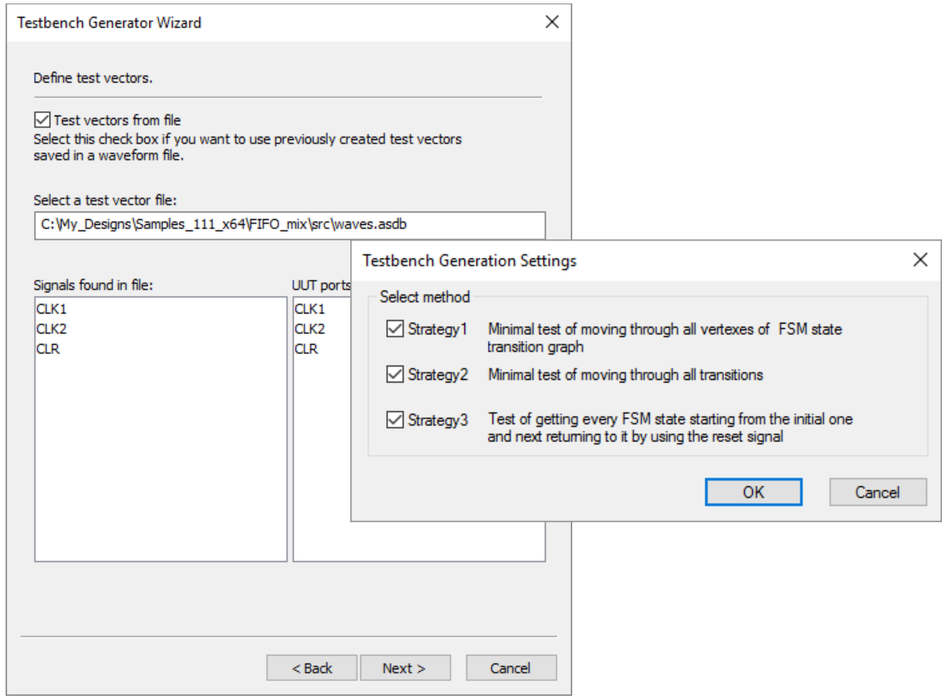
Profiler

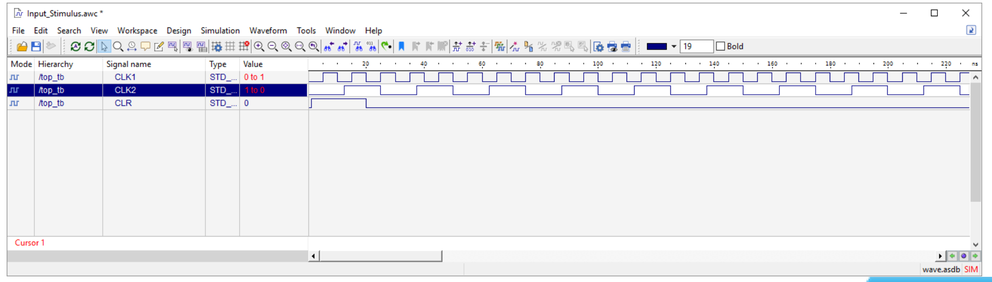
› Profiler measures information on statement execution time and helps to identify performance bottlenecks



Automatic Testbench Generation

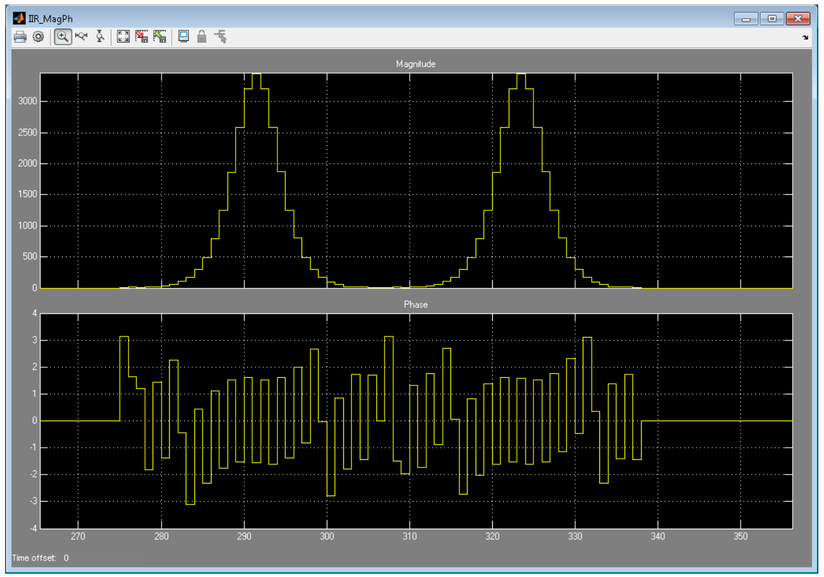
- Automatic generation of simulation testbench in different styles:
  - Generate testbench template for any design unit.
  - Generate complete testbench from test vector inputs such as waveform file.
  - Generate complete testbench that compares simulation result against golden test vectors.
- Automatic Testbench Generation for Finite State Machines (FSM) in three different styles

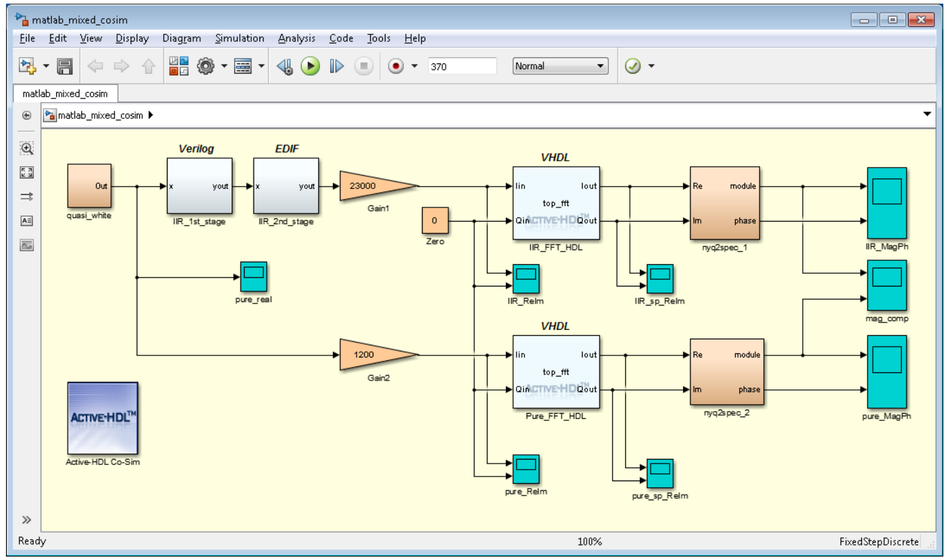




MATLAB and Simulink Co-Simulation

- Direct Co-Simulation Interface to:
  - MATLAB
  - Simulink
- Call MATLAB functions directly from HDL code or call Active-HDL from MATLAB
- High speed system-level testbench and co-verification environment

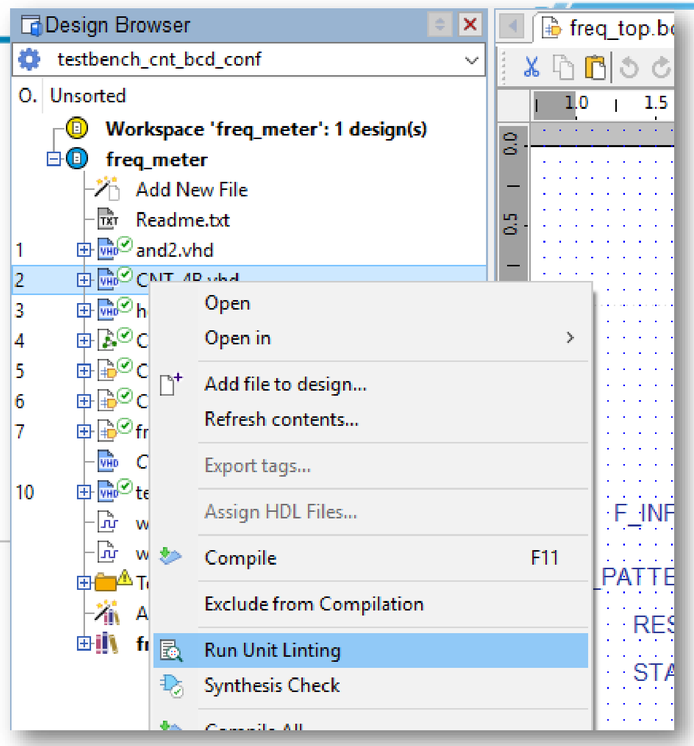




Simple and fast setup of all Simulink co-simulation parameters

Unit Linting

- Build-in interface to ALINT-PRO allows to lint selected units and present results within Active-HDL User Interface
- Pre-linting allows to get rid of most issues before full linting and synthesis process



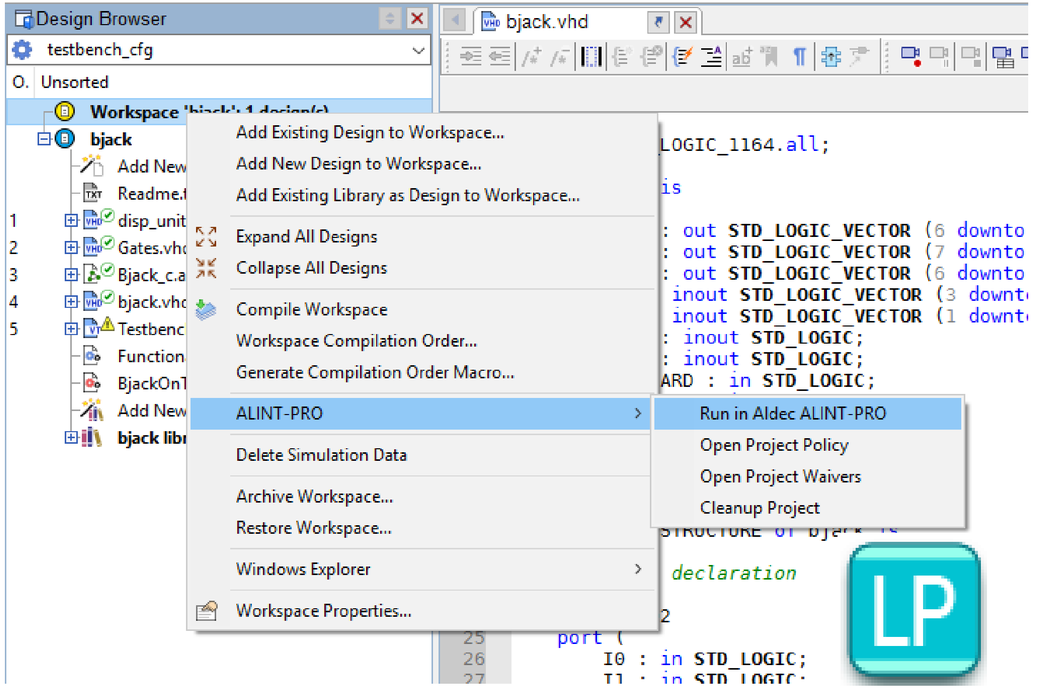

```
--VHDL model for Decoder of 5-bits Binary Numbers to Two BCD Digits
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity bin2bcd is
    port (
          BIN: in STD_LOGIC_VECTOR (4 downto 0);
          BCD_H: out STD_LOGIC_VECTOR (1 downto 0) ;
          BCD_L: out STD_LOGIC_VECTOR (3 downto 0));
end bin2bcd;

architecture BIN2BCD_ARCH of bin2bcd is

begin
    with BIN select
    BCD_L <=    "0001" when "00001" | "01011" | "10101" | "11111",
                "0010" when "00010" | "01100" | "10110",
                "0011" when "00011" | "01101" | "10111",
                "0100" when "00100" | "01110" | "11000",
                "0101" when "00101" | "01111" | "11001",
                "0110" when "00110" | "10000" | "11010",
```

Warning(s):
STARC_VHDL.1.1.6.1 Architecture "bin2bcd(BIN2BCD_ARCH)". The "BIN2BCD_ARCH" architecture name does not match required regular expression "RTL" that should be used for the RTL code description. Level: Recommendation 1.

```
Console
# ALINT-PRO: C:/My_Designs/Samples_111_x64/Bjack/src/disp_units.vhd : (76, 1): Indentation of "H_TMP" concurrent statement is 2 space character(s) less than it is required.
# ALINT-PRO: C:/My_Designs/Samples_111_x64/Bjack/src/disp_units.vhd : (81, 1): Indentation of "LED_L" concurrent statement is 2 space character(s) less than it is required.
# ALINT-PRO: C:/My_Designs/Samples_111_x64/Bjack/src/disp_units.vhd : (83, 1): Indentation of "LED_H" concurrent statement is 2 space character(s) less than it is required.
# ALINT-PRO: Warning: STARC_VHDL.1.1.6.1 C:/My_Designs/Samples_111_x64/Bjack/src/disp_units.vhd : (50, 1): Architecture "bcd2led(BCD2LED_ARCH)". The "BCD2LED_ARCH" architecture name
does not match required regular expression "RTL" that should be used for the RTL code description. Level: Recommendation 1.
# ALINT-PRO: Warning: RMM_VHDL.5.2.5.2 C:/My_Designs/Samples_111_x64/Bjack/src/disp_units.vhd : (53, 2): Architecture "bcd2led(BCD2LED_ARCH)". Comments are missed for the "L_TMP",
"H_TMP" signal declaration(s). Level: Recommendation 2.
# ALINT-PRO: Warning: STARC_VHDL.3.5.6.3_a C:/My_Designs/Samples_111_x64/Bjack/src/disp_units.vhd : (53, 2): Architecture "bcd2led(BCD2LED_ARCH)". No comments are provided for the
"L_TMP", "H_TMP" signal declaration(s). It is recommended to add comments in the same line with signal declaration in order to improve the code readability. Level: Recommendation 2.
# Linting finished successfully.
>
Console
```

ALINT-PRO Interface

- Detects wide variety of design problems at design entry
  - Be aware of ***hidden bugs*** (from synthesizability to CDC and DFT issues)
  - ***Prevent issues*** at subsequent stages of the flow (avoid costly re-spins)
- Get started quickly using predefined ***design checking policies***
- Pick a dedicated ***rule plug-in*** (STARC, DO-254, RMM, CDC and more)
- Available as an **option for DM and PE configurations**
- Included in **EE configuration**



Assertion Based Verification SVA and PSL

- **PSL and SystemVerilog**
  - Properties/assertions/covers embedded in HDL code or kept in separate files
  - Assertion and cover messages in simulator console
  - View assertions and covers in a dedicated **Assertion Viewer** (status and statistics for each assertion/cover)
  - Set **Assertion Breakpoints** and edit them in the new **Assertion** tab of **Breakpoint Editor**
  - **Assertions** and **Covers** tabs in **Code Coverage Viewer** : view statistics for the entire monitored design
  - Assertions displayed in **Structure** tab of the **Design Browser**
  - **HDL Editor and Language Assistant:** syntax highlighting, code auto-complete, and pre-defined language templates.

Conclusion

- Common Kernel Simulator supporting mixed VHDL, Verilog, EDIF, SystemC and SystemVerilog simulation
- Integrated Design Entry and Verification Environment
- Interfaces to FPGA and Logic Synthesis product from universal Design Flow manager – vendor independent
- Ability to invoke simulations in batch mode or GUI, compete user control
- Advanced Debugging and Analysis Tools
- Support for Encrypted IP delivery from FPGA Vendors and IP Providers
- Powerful interface to Matlab/Simulink for signal processing applications – co-simulation
- Assertions Support Available – SVA, PSL
- Advanced verification: UVM, OSVVM, UVVM, Cocotb.

Corp. Headquarters

2260 Corporate Circle , Suite 400
Henderson, NV 89074

Ph +1.702.990.4400
Fax +1.702.990.4414
sales@aldec.com

Training https://www.aldec.com/training

Support https://www.aldec.com/support

Blog https://www.aldec.com/company/blog

Solutions

Riviera-PRO™
Advanced Verification Platform

Active-HDL™
FPGA Design and Simulation

ALINT-PRO™
Design Rule Checking

HES-DVM™
HW/SW Validation Platform

TySOM™
High Performance Embedded Development

Spec-TRACER™
Requirements Lifecycle Management

DO-254/CTS™
FPGA Level In-Target Testing

HES™
SoC/ASIC Prototyping

RTAX/RTSX
Prototyping Microchip™ Rad-Tolerant Devices