CY3138

Warp Enterprise™ Verilog CPLD Software

Features

- **Verilog (IEEE 1364) high-level language compilers with the following features:**
  - **Designs are portable across multiple devices and/or EDA environments**
  - **Facilitates the use of industry-standard simulation and synthesis tools for board- and system-level design**
  - **Support for functions and libraries facilitating modular design methodology**
  - **Support for reduction and conditional operators, blocking and non-blocking procedural assignments, while loops and integers**
- **Several design entry methods support high-level and low-level design descriptions:**
  - **Graphical HDL Block Diagram editor with a library of blocks and a text-to-block conversion utility from Aldec**
  - **Aldec Active-HDL™ FSM graphical Finite State Machine editor**
  - **Behavioral Verilog (IF...THEN...ELSE; CASE...)**
  - **Boolean**
  - **Structural Verilog**
  - **Designs can include multiple entry methods (but only one HDL) in a single design.**
- **Language Assistant library of Verilog templates**
- **Flow Manager Interface to keep track of complex projects**
- **UltraGen™ Synthesis and Fitting Technology:**
  - **Infers "modules" such as adders, comparators, etc., from behavioral descriptions and replaces them with circuits pre-optimized for the target device.**
  - **User-selectable speed and/or area optimization on a block-by-block basis**
  - **Perfectly integrated synthesis and fitting**
  - **Automatic selection of optimal flip-flop type (D type/T type)**
  - **Automatic pin assignment**
- **Support for all Cypress Programmable Logic Devices**
  - **PSI™ (Programmable Serial Interface)**
  - **Delta39K™ CPLDs**
  - **Quantum38K™ CPLDs**
  - **Ultra37000™ CPLDs**
  - **FLASH370i™ CPLDs**
  - **MAX340™ CPLDs**
  - **Industry standard PLDs (16V8, 20V8, 22V10)**
- **VHDL or Verilog timing model output for use with third-party simulators**
- **Active-HDL™ Sim Release 4.1 timing simulation from Aldec**
  - **Graphical waveform simulator**
  - **Graphical entry and modification of all waveforms**
  - **Ability to compare waveforms and highlight differences before and after a design change**
  - **Ability to probe internal nodes**
  - **Display of inputs, outputs, and high impedance (Z) signals in different colors**
  - **Automatic clock and pulse creation**
  - **Support for buses**
  - **Unlimited simulation time**
- **Architecture Explorer™ analysis tool and Dynamic Timing Simulator for PSI, Delta39K and Quantum38K devices:**
  - **Graphical representation of exactly how your design will be implemented on your specific target device**
  - **Zoom from the device level down to the macrocell level**
  - **Determine the timing for any path and view that path on a graphical representation of the chip**
- **Static Timing Report for all devices**
- **Source-Level Behavioral Simulation and Debugger from Aldec**
- **Testbench Generation**
- **UltraISR Programming Cable**
- **Delta39K\Ultra37000 prototype board with a CY37256V 160-pin TQFP device and a CY39100V 208-pin device**
- **On-line documentation and help**

Functional Description

Warp Enterprise™ is an integration of the *Warp* Enterprise CPLD Development package with additional sophisticated EDA software features from Aldec. In addition to accepting IEEE 1364 Verilog text and graphical finite state machines for design entry, *Warp* Enterprise Verilog provides a graphical HDL block diagram editor with a library of graphical HDL blocks pre-optimized for Cypress devices. Plus, it provides a utility to convert HDL text into graphical HDL blocks. *Warp* Enterprise synthesizes and optimizes the entered design, and outputs a JEDEC or Intel hex file for the desired PLD or CPLD (see *Figure 1*). For simulation, *Warp* Enterprise provides a timing simulator, a source-level behavioral simulator, as well as VHDL and Verilog timing models for use with third party simulators. *Warp* Enterprise also provides the designer with important productivity tools such as a testbench generation wizard and the Architecture Explorer graphical analysis tool.

Verilog Compiler

Verilog is a powerful, industry-standard language for behavioral design entry and simulation, and is supported by all major vendors of EDA tools. It allows designers to learn a single language that is useful for all facets of the design process.

Verilog offers designers the ability to describe designs at many different levels. At the highest level, designs can be entered as a description of their behavior. This behavioral description is not tied to any specific target device. As a result, simulation

Downloaded from Elcodis.com electronic components distributor

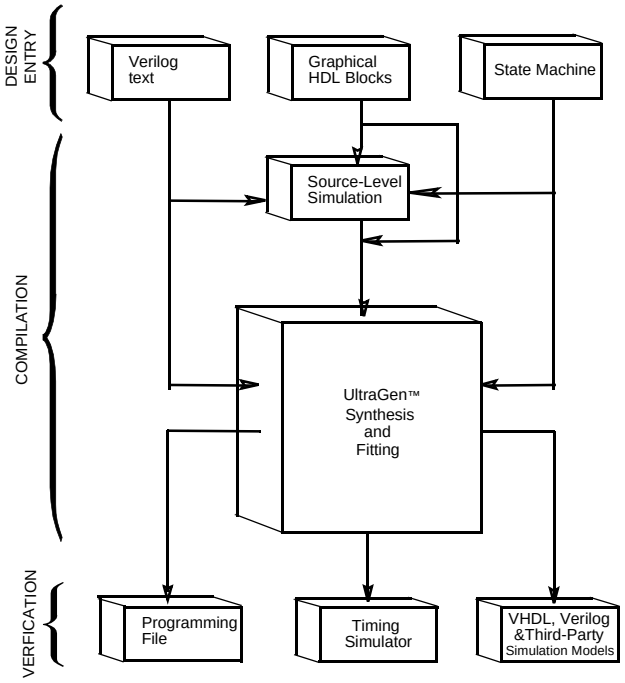

Figure 1. *Warp*® Design Flow

can be done very early in the design to verify correct functionality, which significantly speeds the design process.

The *Warp*® syntax for Verilog includes support for intermediate level entry modes such as state tables and Boolean entry. At the lowest level, designs can be described using gate-level descriptions. *Warp* Enterprise gives the designer the flexibility to intermix all of these entry modes.

In addition, Verilog allows you to design hierarchically, building up entities in terms of other entities. This feature allows you to work either "top-down" (designing the highest levels of the system and its interfaces first, then progressing to greater and greater detail) or "bottom-up" (designing elementary building blocks of the system, then combining these to build larger and larger parts) with equal ease.

Because this languages is an IEEE standard, multiple vendors offer tools for design entry and simulation at both high and low levels and synthesis of designs to different silicon targets. The use of device-independent behavioral design entry gives users the freedom to easily migrate to high volume technologies. The wide availability of Verilog tools provides complete vendor independence as well. Designers can begin their project using *Warp* Enterprise for Cypress CPLDs and convert to high volume ASICs using the same Verilog behavioral description with industry-standard synthesis tools.

The Verilog language also allows users to define their own functions. User-defined functions allow users to extend the capabilities of the language and build reusable files of tested routines. Verilog provides control over the timing of events or processes. It has constructs that identify processes as either sequential, concurrent, or a combination of both. This feature is essential when describing the interaction of complex state machines.

Verilog is a rich programming language. Its flexibility reflects the nature of modern digital systems and allows designers to create accurate models of digital designs. Because it is not a verbose language it is easy to learn and compile. In addition, models created in Verilog can readily be transported to other EDA Environments. *Warp* Enterprise Verilog supports IEEE 1364 Verilog including loops, reduction and conditional operators.

A Verilog Design Example

Design Entry

Warp Enterprise descriptions specify:

- The behavior or structure of a design, and
- the mapping of signals in a design to the pins of a PLD/CPLD (optional)

The part of a *Warp* Enterprise description that specifies the behavior or structure of the design is called a module. The module declares the design's interface signals (i.e., defines what external signals the design has, and what their directions and types are).

The module portion of a design file is a declaration of what a design presents to the outside world (the interface). For each external signal, the module specifies a signal name, a direction and a data type. In addition, the module declaration specifies a name by which the entity can be referenced in other modules. This section shows code segments from four sample design files. The top portion of each example features the module declaration.

Behavioral Description

The module portion of a design file specifies the function of the design. As shown in *Figure 1*, multiple design-entry methods are supported in *Warp* Enterprise. A behavioral description in Verilog often includes well known constructs such as If…Else, and Case statements. Here is a code segment from a simple state machine design (soda vending machine) that uses behavioral Verilog to implement the design:

```
MODULE drink (nickel, dime, quarter, clock,
            returnDime, returnNickel,
            giveDrink);

  INPUT nickel, dime, quarter, clock;
  OUTPUT returnDime,returnNickel,giveDrink;
  REG returnDime, returnNickel, giveDrink;

  PARAMETER zero = 0, five = 1, ten = 2,
  fifteen = 3, twenty = 4, twentyfive = 5
  owedime = 6;

REG[1:0] drinkStatus;

ALWAYS@ (POSEDGE clock)
BEGIN

  giveDrink = 0;
  returnDime = 0;
  returnNickel = 0;

  CASE(drinkStatus)

    zero: BEGIN
       IF (nickel)
```


Downloaded from Elcodis.com electronic components distributor

```
      drinkStatus = five;
    ELSE IF (dime)
      drinkStatus = ten;
    ELSE IF (quarter)
      drinkStatus = twentyfive;
  END

  five: BEGIN
    IF (nickel)
      drinkStatus = ten;
    ELSE IF (dime)
      drinkStatus = fifteen;
    ELSE IF (quarter)
      BEGIN
        drinkStatus = zero;
        giveDrink = 1;
      END

  END

  // Several states are omitted in this
  // example. The omitted states are ten
  // fifteen, twenty, and twentyfive.

  owedime: BEGIN
    returnDime = 1;
    drinkStatus = zero;
  END

  default: BEGIN
  // This makes sure that the state
  // machine resets itself if
  // it somehow gets into an undefined state.
    drinkStatus = zero;
  END

 ENDCASE
 END
 ENDMODULE
```

Verilog is not a strongly typed language. The simplicity and readability of the following code is increased by use of the CASEX. The CASEX command accepts “Don’t Cares” and chooses the branch depending on the value of the expression.

```
MODULE sequence (clk, s);
  INPUT clk;
  INOUT s;
  WIRE s;
  REG temp;
  REG[3:0] count;

ALWAYS@(POSEDGE clk)
  CASEX(count)
4’b00XX: BEGIN
             temp=1;
             count=count+1;
             end
4’b01XX: BEGIN
             temp=0;
             count=count+1;
             end
4’b100X: BEGIN
             temp=1;
             count=count+1;
             end
default: BEGIN
             temp=0;
             count=0;
             end

ENDCASE
ASSIGN s=temp;
ENDMODULE
```

Boolean Equations

A second design-entry method available to *Warp* Enterprise users is Boolean equations. *Figure 2* displays a schematic of a simple one-bit half adder. The following code describes how this one-bit half adder can be implemented in *Warp* Enterprise with Boolean equations:

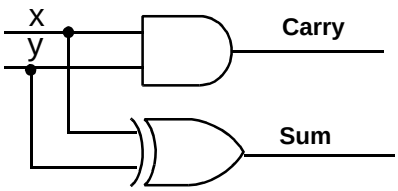

Figure 2. One-Bit Half Adder

```
MODULE half_adder(x, y, sum, carry);
  INPUT x, y;
  OUTPUT sum, carry;

ASSIGN sum = x^y;
ASSIGN carry = x&y;
ENDMODULE
```

Structural Verilog

While all of the design methodologies described thus far are high-level entry methods, structural Verilog provides a method for designing at a very low level. In structural descriptions, the designer simply lists the components that make up the design and specifies how the components are wired together.
Figure 3 displays the schematic of a simple 3-bit shift register and the following code shows how this design can be described in *Warp* Enterprise using structural Verilog.

```
MODULE shifter3 (clk, x, q0, q1, q2);
  INPUT clk, x;
  OUTPUT q0, q1, q2;
  WIRE q0, q1, q2;
  REG q0_temp, q1_temp, q2_temp;

    DFF d1(x,clk,q0_temp);
    DFF d2(q0_temp,clk,q1_temp);
    DFF d3(q1_temp,clk,q2_temp);
    ASSIGN q0 = q0_temp;
    ASSIGN q1 = q1_temp;
    ASSIGN q2 = q2_temp;

ENDMODULE;
```


Downloaded from Elcodis.com electronic components distributor

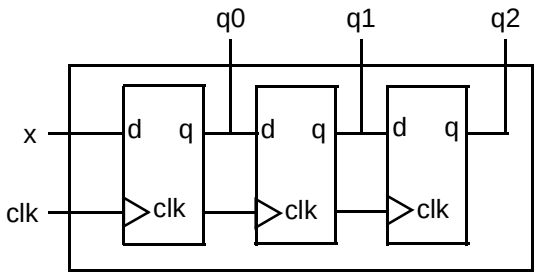

Figure 3. Three-Bit Shift Register Circuit Design

All of the design-entry methods described can be mixed as desired so long as only one HDL is used. Verilog have the ability to combine both high- and low-level entry methods in a single file. The flexibility and power of Verilog allow users of *Warp* Enterprise to describe designs using whatever method is appropriate for their particular design.

Finite State Machine Editor

Aldec's Active-HDL™ FSM finite state machine editor, allows graphic design entry through the use of graphical state diagrams. A design may be represented graphically using state diagrams and data flow logic. This tool will automatically generate the HDL code of the design.

HDL Block Diagram Editor

The HDL block diagram editor lets you represent portions of your code with graphical symbols. This representation allows you to view the high-level structure of your complex designs and lets you copy and paste entire modules of your design within or between designs. The editor comes with a library of HDL blocks optimized for Cypress devices. *Warp* Enterprise comes with utility that converts HDL text into these blocks.

Language Assistant

The language assistant is a library of language templates that you can browse and automatically insert into your HDL text. They provide syntax and structure and give examples to aid users who are new using a particular HDL.

Flow Manager

The flow manager is a special interface that helps you keep track of your complex projects. It arranges the tools as part of the logical flow the designer takes through a project and remembers what steps have been completed on which designs.

Source-Level Simulation

Warp Enterprise's source-level behavioral simulator helps you catch problems with your code early in the design process by letting you simulate a design before synthesis. The tool lets you graphically watch inputs and outputs, gives you timing information and allows you to step through your code line by line.

Compilation

Once the Verilog description of the design is complete, it is compiled using *Warp* Enterprise. Although implementation is with a single command, compilation is actually a multistep process as shown in *Figure 1*. The first part of the compilation process is the same for all devices. The input description is synthesized to a logical representation of the design. *Warp* synthesis is unique in that the input languages support device-independent design descriptions. Competing programmable logic compilers require very specific and device-dependent information in the design description.

Warp synthesis is based on UltraGen technology. This technology allows *Warp* Enterprise to infer adders, subtractors, multipliers, comparators, counters and shifters from the behavioral descriptions. *Warp* Enterprise then replaces these operators internally with an architecture-specific circuit. This circuit or "module" is also pre-optimized for either area or speed. *Warp* Enterprise uses the appropriate implementation based on user directives.

The second step of compilation is an iterative process of optimizing the design and fitting the logic into the targeted device. Logical optimization in *Warp* Enterprise is accomplished using Espresso algorithms. The optimized design is automatically fed to the *Warp* Enterprise fitter for targeting a PLD or CPLD. This fitter supports the automatic or manual placement of pin assignments as well as automatic selection of D or T flip-flops. After optimization and fitting, *Warp* Enterprise creates a JEDEC or Intel hex file for the specified PLD or CPLD.

Automatic Error Tracking

Warp Enterprise features automatic error location that allows problems to be diagnosed and corrected in seconds. Errors from compilation are displayed immediately in a window. If the user highlights a particular error, *Warp* Enterprise will automatically open the source code file and highlight the offending line in the entered design. If the device fitting process includes errors, a window will again describe them. A detailed report file is generated indicating the resources required to fit the input design and any problems that occurred in the process.

Timing Simulation

The Aldec Active-HDL Sim post-fitting timing simulator provides timing simulation for PLDs/CPLDs and features interactive waveform viewing as well as graphical creation and editing of all waveforms. The simulator also provides the ability to probe internal nodes, and automatically generate clocks and pulses. The version in *Warp* Enterprise has the ability to compare waveforms and highlight differences before and after a design change. In *Warp* Enterprise there is no maximum simulation time. To use the timing simulator in *Warp* Enterprise Verilog you must use a Verilog netlist.

Warp Enterprise Verilog can also output standard VHDL or Verilog timing models that all third-party simulators can use to perform functional and timing verifications of a synthesized design.

Architecture Explorer

The Architecture Explorer graphically displays how the design will be implemented on the chip. It provides a view of the entire device to show what memory elements and logic clusters have been used for what part of the design. This gives the designer an idea of what resources are free. The Architecture Explorer allows you to zoom in multiple times. At maximum zoom it displays the logic gate implementation in each macrocell. The Architecture Explorer is available for PSI, Delta39K, and Quantum38K devices.

Downloaded from Elcodis.com electronic components distributor

Timing Analyzer

The Timing Analyzer gives the time across any path as well as the breakdown of what steps are causing the timing delays. This tool does not simply display the general specification for the target device but a worst-case simulation of the actual path being taken through the device. When you highlight a path on the timing analyzer, the source and destination of that path are displayed on the Architecture Explorer. The timing analyzer graphical interface is also available for PSI, Delta39K, and Quantum38K devices. However, for other devices the same information is available in a report file.

Programming

Cypress's FLASH370i, Ultra37000, Quantum38K and Delta39K In-System Reprogrammable™ (ISR™) devices can be programmed on board with an ISR programmer. For PSI, Delta39K and Quantum38K devices *Warp* Enterprise produces an Intel hex file. The ISR programmer converts this file into STAPL and programs the device. For Ultra37000 and FLASH370i devices, *Warp* Enterprise produces a JEDEC file. For Ultra37000, the ISR programmer converts this file into JAM/STAPL and programs the device. For FLASH370i, the JEDEC file is used directly to program the device.

Warp Enterprise comes with a UltraISR Programming Cable and a Delta39K\Ultra37000 prototype board with a CY37256V 160-pin TQFP device and a CY39100V 208-pin TQFP device*.

The JEDEC and Intel hex files produced by *Warp* Enterprise can also be used with any qualified third party programmer to program Cypress CPLDs.

For more information on Cypress's ISR software see the ISR Programming Kit (CY3900i) data sheet.

Warp® Software System Requirements

- IBM PC or equivalent (Pentium® class recommended)
- 32 Mb of RAM (64 Mb recommended)
- 110 Mb Disk Space
- CD-ROM drive
- Windows 98, or Windows NT 4.0
- *Warp* Enterprise for Verilog hardware key

ISR Software PC System Requirements

- IBM PC or compatible running Windows 98, Windows 98 Second Edition, Windows ME, Windows NT 4.0 Service Pack 5 or later, or Windows 2000 Service Pack 1 or later
- One free parallel port
- Minimum of 32 MB of RAM
- Approximately 30 MB free hard disk space

Product Ordering Information

Product Code	Description
CY3138R62	*Warp* Enterprise Verilog CPLD software for PCs

Warp Enterprise includes:

- Cypress Lab CD-ROM with *Warp* Enterprise, ISR software, on-line documentation (Getting Started Manual, User's Guide, HDL Reference Manual, Databook) and other Cypress software
- UltraISR Programming Cable
- Delta39K\Ultra37000 prototype board with a CY37256V 160-pin TQFP device and a CY39100V 208-pin TQFP device[1]
- Registration Card
- *Warp* Enterprise for Verilog hardware key

Warp Professional, *Warp* Enterprise, UltraGen, Ultra37000, Quantum38K, Delta39K, PSI, MAX340, ISR, In-System Reprogrammable, and FLASH370i are trademarks of Cypress Semiconductor Corporation.
Warp is a registered trademark of Cypress Semiconductor Corporation.
Pentium is a registered trademark of Intel Corporation.
Windows 98, Windows 2000 and Windows NT are trademarks of Microsoft Corporation.
Solaris is a trademark of Sun Microsystems Corporation.

Note:

1. Cypress reserves the right to substitute prototype boards based on product availability.

© Cypress Semiconductor Corporation, 2002. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress Semiconductor product. Nor does it convey or imply any license under patent or other rights. Cypress Semiconductor does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress Semiconductor products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress Semiconductor against all charges.

Downloaded from Elcodis.com electronic components distributor

Document Title: CY3138 *Warp* Enterprise™ Verilog CPLD Software Document Number: 38-03045				
REV.	**ECN NO.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	109902	09/22/01	SZV	Change from Spec number: 38-01032 to 38-03045
*A	111241	01/21/02	CNH	Update product code, remove references to Windows95

Downloaded from Elcodis.com electronic components distributor