Warp2® Verilog Development System for CPLDs

Features

- **Verilog (IEEE 1364) high-level language compiler with the following features:**
  - **— Facilitates device independent design**
  - **— 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 dsign**
- ***Warp2*®provides synthesis of IEEE Standard 1364 Verilog including:**
  - **— Reduction and conditional operators**
  - **— Blocking and non-blocking procedural assignments**
  - **— While loop**
  - **— Integers**
- **Several design entry methods support high and low-level design descriptions including:**
  - **— Behavioral Verilog (IF…ELSE; CASE…)**
  - **— Boolean**
  - **— Structural Verilog (RTL)**
  - **— Aldec Active-HDL™ FSM graphical Finite State Machine editor (PC only)**
  - **— Designs can include multiple Verilog entry methods in a single design**
- **State-of-the-art optimizations and reduction algorithms including:**
  - **— Automatic selection of optimal flip-flop type (D type/T type)**
  - **— Automatic pin assignment**
- **UltraGen™ Synthesis and Fitting Technology with the following features:**
  - **— Infers "modules" like adders, comparators, etc., from behavioral descriptions**
  - **— Replaces operator internally with an architecture specific circuit based on the target device**
  - **— User selectable speed and/or area optimization on a block-by-block basis**
- **Supports all Cypress Programmable Logic Devices including:**
  - **— Ultra37000™ CPLDs (now with FBGA support)**
  - **— FLASH370i™ CPLDs**
  - **— MAX340™ CPLDs**
  - **— Industry standard PLDs (16V8, 20V8, 22V10)**
- **Verilog and VHDL timing model output for use with third-party simulators**
- **Timing simulation provided with Active-HDL™ Sim Release 3.3 from Aldec (PC only) including:**
  - **— Graphical waveform simulator**
  - **— Entry and modification of on-screen waveforms**
  - **— Ability to probe internal nodes**
  - **— Display of inputs, outputs, and high impedance (Z) signals**
  - **— Automatic clock and pulse creation**
  - **— Support for buses**
- **Year 2000 Compliant**
- **PC support (Windows 95™ , Windows 98™ and Windows NT™ 4.0)**
- **Workstation support for Sun Solaris™ and HP-UX™**
- **On-line documentation and help**

Functional Description

Warp2 is a state-of-the-art HDL compiler for designing with Cypress's Complex Programmable Logic Devices (CPLDs). *Warp2* utilizes a subset of IEEE 1364 Verilog as its Hardware Description Language (HDL) for design entry. Another design entry method supported by *Warp*2 is through Aldec's Active-HDL™FSM graphical Finite State Machine Editor (PC only). *Warp2* accepts Verilog, synthesizes and optimizes the entered design, and outputs a JEDEC map for the desired PLD or CPLD (see *Figure 1*). For simulation, *Warp2* provides a timing simulator (PC only), as well as VHDL and Verilog models for use with third-party simulators.

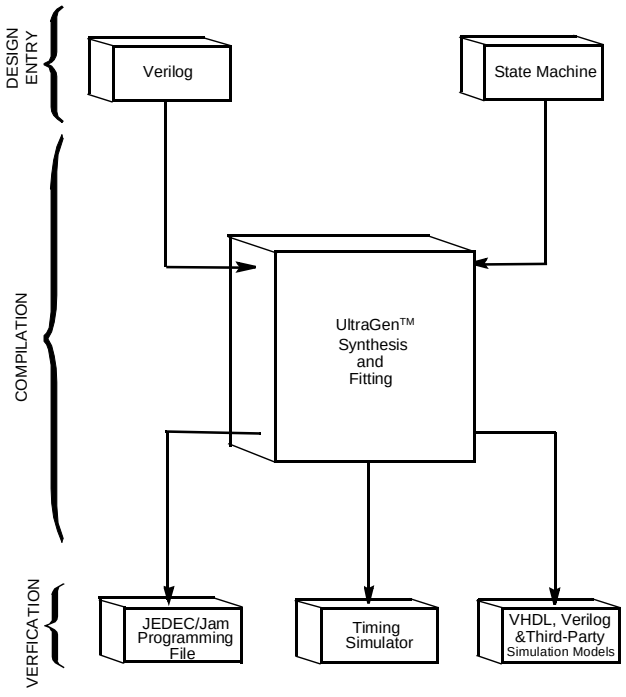

Figure 1. *Warp2* Verilog Design Flow

Verilog Compiler

Verilog is a powerful, non-proprietary language that is a standard for behavioral design entry and simulation, and is supported by major vendors of EDA tools. Verilog 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 can be done very early in the design to verify correct functionality, which significantly speeds the design process.

The *Warp2* Verilog syntax also 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 RTL (Register Transfer Language) descriptions. *Warp2* 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 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 Verilog 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 *Warp2* for Cypress CPLDs and convert to high volume gate arrays using the same Verilog behavioral description with industry-standard synthesis tools.

The Verilog language 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 also provides control over the timing of events or processes.

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 a hardware system. In addition, models created in Verilog can readily be transported to other EDA Environments. *Warp2* supports IEEE 1364 Verilog including loops, reduction and conditional operators.

Designing with *Warp2*

Warp2 features a newly-designed user interface to facilitate the design entry process. A sample screen shot of the *Warp2* editor is shown in *Figure 2*.

Design Entry

Warp2 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)

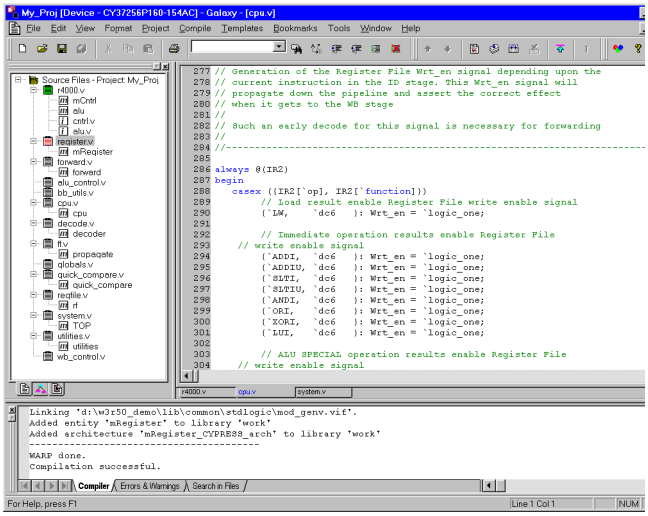

Figure 2. *Warp2* Graphic User Interface

The part of a *Warp2* 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 *Warp2*. 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)
       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 *Warp2* Verilog users is Boolean equations. *Figure 3* displays a schematic of a simple one-bit half adder. The following code describes how this one-bit half adder can be implemented in *Warp2* with Boolean equations:

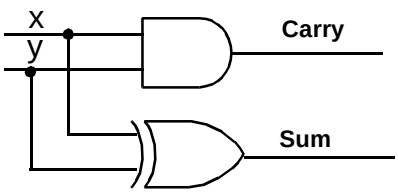

Figure 3. One-Bit Half Adder

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

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

Structural Verilog (RTL)

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 (also called RTL), the designer simply lists the components that make up the design and specifies how the components are wired together. *Figure 4* displays the schematic of a simple 3-bit shift register and the following code shows how this design can be described in *Warp2* 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;
```

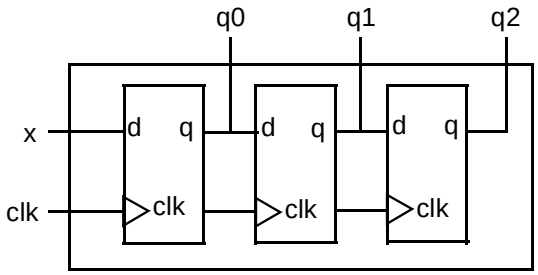

Figure 4. Three-Bit Shift Register Circuit Design

All of the design-entry methods described can be mixed as desired. Verilog has the ability to combine both high- and low-level entry methods in a single file. The flexibility and power of Verilog allows users of *Warp2* to describe designs using whatever method is appropriate for their particular design.

Finite State Machine Editor (PC only)

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.

Compilation

Once the Verilog description of the design is complete, it is compiled using *Warp2*. 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 Verilog description is synthesized to a logical representation of the design. *Warp2* synthesis is unique in that the input language (Verilog) supports device-independent design descriptions. Competing programmable logic compilers require very specific and device-dependent information in the design description

Warp2 synthesis is based on UltraGen technology. This technology allows *Warp2* to infer "modules" like adders, multipliers, comparators, etc., from behavioral descriptions. *Warp2* then replaces that operator internally with an architecture specific circuit based on the target device. This circuit or "module" is also pre-optimized for either area or speed, and *Warp2* uses the appropriate implementation based on user directives.

The second step of compilation is an interactive process of optimizing the design and fitting the logic into the targeted device. Logical optimization in *Warp2* is accomplished using Espresso algorithms. The optimized design is automatically fed to the *Warp2* 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, *Warp2* creates a JEDEC file for the specified PLD or CPLD.

To facilitate design conversion, a JEDEC translator is provided to convert FLASH370i JEDEC files to JEDEC files that target Ultra37000 devices.

Automatic Error Tracking

Of course, the compilation process may not always go as planned. Verilog syntax errors should be identified and corrected in the pre-synthesis functional simulation stage. During the compilation phase, *Warp2* will detect errors that occur in the fitting process. *Warp2* 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, *Warp2* 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.

Simulation

Warp2 includes a post-synthesis timing simulator called Active-HDL Sim, available for PCs only. Active-HDL Sim features a graphical waveform simulator that can be used to simulate PLD/CPLD designs generated in *Warp2*. The simulator provides timing simulation for PLDs/CPLDs and features interactive waveform editing and viewing. The simulator also provides the ability to probe internal nodes and automatically generate clocks and pulses.

Warp2 will also output standard Verilog and VHDL timing models on all operating systems supported. These models can be used with many third-party simulators to perform functional and timing verifications of the synthesized design.

Programming

The result of *Warp2* compilation is a JEDEC file that implements the input design in the targeted device. Using this file, Cypress devices can be programmed on any qualified third-party programmer.

Cypress's Ultra37000 and FLASH370i In-System Reprogrammable™ (ISR™) devices can also be programmed on board with an ISR programmer. The JEDEC files are used to program FLASH370i CPLDs or are converted to Jam files by Cypress's ISR software for use with Ultra37000 CPLDs. Once in Jam format, Ultra37000 CPLDs may be programmed using the Jam player with Cypress's ISR software and 37000 UltraISR cable. For more information on Cypress's ISR software, refer to the data sheet for either the Ultra37000 ISR Programming Kit (CY3700i) or the FLASH370i ISR Programming Kit (CY3600i).

System Requirements

For PCs

- IBM PC or equivalent (Pentium® class recommended)
- 32 Mbytes of RAM (64 Mbytes recommended)
- 110 Mbytes Disk Space
- CD-ROM drive
- Windows (including Japanese) 95, Windows 98, or Windows NT 4.0

For Sun Workstations

- SPARC™ CPU
- Solaris™ 2.6
- 32 Mbytes of RAM (64 Mbytes recommended)
- CD-ROM drive

For HP 9000™ Workstations

- HP-UX™ 10.20
- 32 Mbytes of RAM (64 Mbytes recommended)
- CD-ROM drive

Product Ordering Information

Product	Code Description
CY3110R52	*Warp2* Verilog development system for PCs
CY3115R52	*Warp2* Verilog development system for Unix Workstations
CY3110JR52	*Warp2* Verilog Japanese development system for PCs

***Warp2* Verilog includes:**

- CD-ROM with *Warp2*, Aldec Active-HDL Sim & FSM, and on-line documentation (Getting Started Manual, User's Guide, HDL Reference Manual)
- Registration Card
- Release Notes

- Document #: 38–00734–C