3121: January 6, 1995
Revision: October 20, 1995

CYPRESS

PRELIMINARY

CY3121
CY3126

Warp2™ VHDL Compiler for PLDs and CPLDs

Features

- **VHDL (IEEE1076 and 1164) high-level language compiler**
  - **— VHDL facilitates device independent design**
  - **— VHDL designs are portable across multiple devices and/or CAE platforms**
  - **— VHDL facilitates the use of industry-standard simulation and synthesis tools for board and system-level design**
  - **— VHDL supports functions and libraries facilitating modular design methodology**
- ***Warp2*™ provides synthesis for a powerful subset of IEEE standard 1076 and 1164 VHDL including:**
  - **— enumerated types**
  - **— operator overloading**
  - **— for … generate statements**
  - **— integers**
- **State-of-the-art optimization and reduction algorithms**
  - **— Optimization for flip-flop type (D type/T type)**
  - **— Automatic selection of optimal flip-flop type (D type/T type)**
  - **— Automatic pin assignment**
  - **— Automatic state assignment (grey code, one-hot, binary)**
- **Several design entry methods support high and low-level design descriptions:**
  - **— Behavioral VHDL (IF…THEN…ELSE; CASE…)**
  - **— Boolean**
  - **— Structural VHDL (RTL)**
- **Designs can include multipleVHDL entry methods in a single design**
- **Supports all Cypress PLDs and CPLDs, including MAX340™ and FLASH370™**
- **Functional simulation provided with Cypress NOVA simulator:**
  - **— Graphical waveform simulator**
  - **— Entry and modification of on-screen waveforms**
  - **— Ability to probe internal nodes**
  - **— Display of input and output signals in different colors**
  - **— Automatic clock and pulse creation**
  - **— Waveform to JEDEC test vector conversion utility**
  - **— Support for buses**
- **PC, Sun, and HP platforms**
- **Windows™ 3.1 or above**
- **Motif® on Sun workstations**

Functional Description

Warp2 is a state-of-the-art VHDL compiler for designing with Cypress PLDs and CPLDs. *Warp2* is fully IEEE1076 and 1164 VHDL compliant. VHDL provides a number of significant benefits for the design engineer. *Warp2* accepts VHDL input, synthesizes and optimizes the entered design, and outputs a JEDEC map for the desired device (see *Figure 1*). For simulation, *Warp2* provides a graphical waveform simulator called NOVA.

VHDL Compiler

VHDL (Very High Speed Integrated Circuit Hardware Description Language) is a powerful, non-proprietary language that is a standard for behavioral design entry and simulation. It is mandated for use by the Department of Defense and is supported by every major vendor of CAE tools. VHDL allows designers to learn a single language that is useful for all facets of the design process.

VHDL 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.

Warp2's VHDL syntax also includes support for intermediate level entry modes such as state table 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, VHDL allows users 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 VHDL is an IEEE standard, multiple vendors offer tools for design entry, simulation at both high and low levels, and synthe-

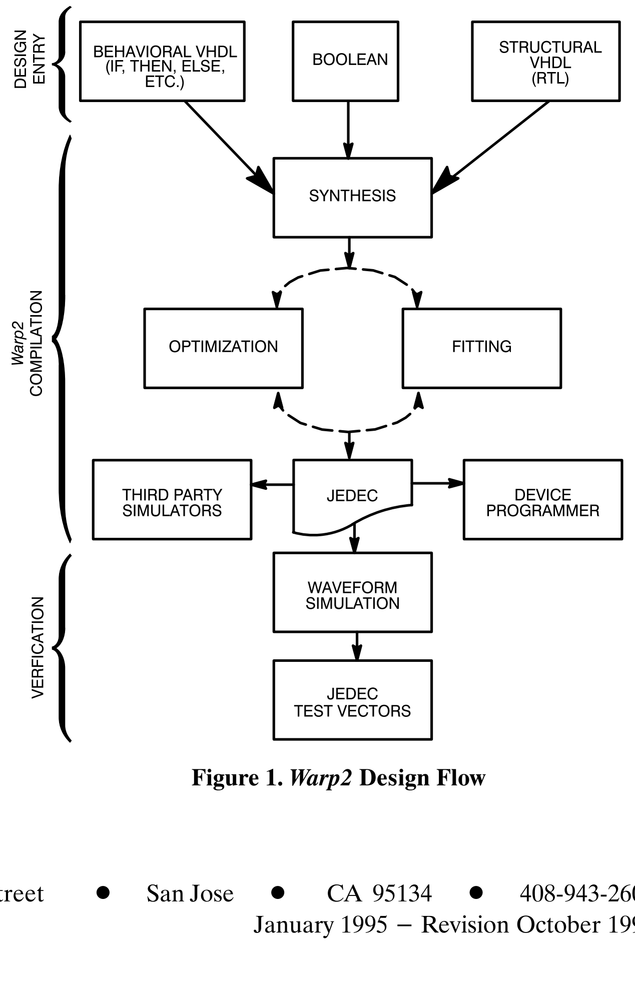

Figure 1. *Warp2* Design Flow

sis of designs to different silicon targets. The use of device independent behavioral design entry gives users the freedom to retarget designs to different devices. The wide availability of VHDL tools provides complete vendor independence as well. Designers can begin their project using *Warp2* for Cypress PLDs and convert to high volume gate arrays using the same VHDL behavioral description with industry-standard synthesis tools.

While design portability and device independence are significant benefits, VHDL has other advantages. The VHDL language allows users to define their own functions. User-defined functions allow users to extend the capabilities of the language and build reusable libraries of tested routines. As a result the user can produce complex designs faster than with ordinary "flat" languages. VHDL also provides control over the timing of events or processes. VHDL has constructs that identify processes as either sequential, concurrent, or a combination of both. This is essential when describing the interaction of complex state machines.

Cypress chose to offer tools that use the VHDL language because of the languages' universal acceptance, the ability to do both device and vendor independent design, simulation capabilities at both the chip and system level that improve design efficiency, the wide availability of industry-standard tools with VHDL support for both simulation and synthesis, and the inherent power of the language's syntax.

VHDL 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 of its depth and completeness, it is easier to describe a complex hardware system accurately in VHDL than in any other hardware description language. In addition, models created in VHDL can readily be transported to other CAE systems. *Warp2* supports a rich subset of VHDL including loops, for…generate statements, full hierarchical designs with packages, as well as synthesis for enumerated types and integers.

Designing with *Warp2*

Design Entry

Warp2 descriptions specify

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

The part of a *Warp2* description that specifies the behavior or structure of the design is called an entity/architecture pair. Entity/architecture pairs, as their name implies, are divided into two parts: an entity declaration, which declares the design's interface signals (i.e., tells the world what external signals the design has, and what their directions and types are), and a design architecture, which describes the design's behavior or structure.

Design Entity

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

Behavioral Description

The architecture portion of a design entity file specifies the function of the design. As shown in *Figure 1*, multiple design-entry methods are supported in *Warp2*. A behavioral description in VHDL often includes well known constructs such as If…Then…Else, and Case statements. Here is a code segment from a simple state machine design (soda vending machine) that uses behavioral VHDL to implement the design:

```
ENTITY drink IS
  PORT (nickel,dime,quarter,clock:in bit;
     returnDime,returnNickel,giveDrink:outbit);
END drink;


ARCHITECTURE fsm OF drink IS

TYPE drinkState IS (zero,five,ten,fifteen,
twenty,twentyfive,owedime);
SIGNAL drinkstatus:drinkState;

BEGIN

PROCESS BEGIN

  WAIT UNTIL clock = '1';

   giveDrink <= '0';
   returnDime <= '0';
   returnNickel <= '0';

   CASE drinkStatus IS

   WHEN zero =>
     IF (nickel = '1') THEN
       drinkStatus <= drinkStatus'SUCC
       (drinkStatus);
         -- goto Five
     ELSIF (dime = '1') THEN
       drinkStatus <= Ten;
     ELSIF (quarter = '1') THEN
       drinkStatus <= TwentyFive;
     ENDIF;
   WHEN Five =>
     IF (nickel = '1') THEN
       drinkStatus <= Ten;
     ELSIF (dime = '1') THEN
       drinkStatus <= Fifteen;
     ELSIF (quarter = '1') THEN
       giveDrink <= '1';
       drinkStatus <= drinkStatus'PRED
       (drinkStatus);
         -- goto Zero
     ENDIF;

   WHEN oweDime =>
     returnDime <= '1';
     drinkStatus <= zero;

   when others =>
   -- This ELSE makes sure that the state
   -- machine resets itself if
   -- it somehow gets into an undefined state.
     drinkStatus <= zero;
   END CASE;
   END PROCESS;

END FSM;
```

VHDL is a strongly typed language. It comes with several predefined operators, such as + and /= (add, not-equal-to). VHDL offers the capability of defining multiple meanings for operators (such as +), which results in simplification of the code written. For example, the following code segment shows that "count = count +1" can be written such that count is a bit vector, and 1 is an integer.

```
ENTITY sequence IS
  port (clk: in bit;
    s : inout bit);
end sequence;

ARCHITECTURE fsm OF sequence IS

SIGNAL count: INTEGER RANGE 0 TO 7;

BEGIN

PROCESS BEGIN

  WAIT UNTIL clk = '1';

    CASE count IS

    WHEN 0 | 1 | 2 | 3 =>
      s <= '1';
      count <= count + 1;
    WHEN 4 =>
      s <= '0';
      count <= count + 1;
    WHEN 5 =>
      s <= '1';
      count <= '0';
    WHEN others =>
      s <= '0';
      count <= '0';
    END CASE;

END PROCESS;

END FSM;
```

In this example, the + operator is overloaded to accept both integer and bit arguments. *Warp2* supports overloading of operators.

Functions

A major advantage of VHDL is the ability to implement functions. The support of functions allows designs to be reused by simply specifying a function and passing the appropriate parameters. *Warp2* features some built-in functions such as ttf (truth-table function). The ttf function is particularly useful for state machine or look-up table designs. The following code describes a seven-segment display decoder implemented with the ttf function:

```
ENTITY seg7 IS
  PORT(
    inputs: IN BIT_VECTOR (0 to 3)
    outputs: OUT BIT_VECTOR (0 to 6)
  );
END SEG7;

ARCHITECTURE mixed OF seg7 IS

CONSTANT truthTable:
  x01_table (0 to 11, 0 to 10) := (
-- input   &     output
-- ----------------------
  "0000"   &   "0111111",
  "0001"   &   "0000110",
  "0010"   &   "1011011",
  "0011"   &   "1001111",
  "0100"   &   "1100110",
  "0101"   &   "1101101",
  "0110"   &   "1111101",
  "0111"   &   "0000111",
  "1000"   &   "1111111",
  "1001"   &   "1101111",
  "101x"   &   "1111100", --creates E pattern
  "111x"   &   "1111100"
  );

BEGIN

    outputs <= ttf(truthTable,inputs);

END mixed;
```

Boolean Equations

A third design-entry method available to *Warp2* 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 *Warp2* with Boolean equations:

```
--entity declaration
ENTITY half_adder IS
  PORT (x, y : IN BIT;
    sum, carry : OUT BIT);
END half_adder;
--architecture body
ARCHITECTURE behave OF half_adder IS
BEGIN
  sum <= x XOR y;
  carry <= x AND y;
END behave;
```

Structural VHDL (RTL)

While all of the design methodologies described thus far are high-level entry methods, structural VHDL 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 3* 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 VHDL:

```
ENTITY shifter3 IS port (
    clk : IN BIT;
    x : IN BIT;
    q0 : OUT BIT;
    q1 : OUT BIT;
    q2 : OUT BIT);
  END shifter3;

ARCHITECTURE struct OF shifter3 IS
  SIGNAL q0_temp, q1_temp, q2_temp : BIT;
```

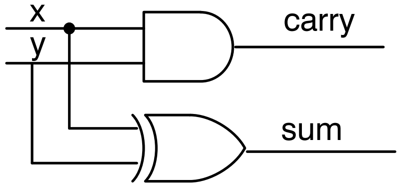

Figure 2. One-Bit Half Adder

```
BEGIN
  d1 : DFF PORT MAP(x,clk,q0_temp);
  d2 : DFF PORT MAP(q0_temp,clk,q1_temp);
  d3 : DFF PORT MAP(q1_temp,clk,q2_temp);
  q0 <= q0_temp;
  q1 <= q1_temp;
  q2 <= q2_temp;
END struct;
```

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

Compilation

Synthesis and optimization is a one button process. The first step is synthesizing the input VHDL into a logical representation of the design. *Warp2* synthesis is unique in that the input language (VHDL) supports device-independent design descriptions. Competing programmable logic compilers require very specific and device-dependent information in the design input file.

The second step of compilation is an iterative process of optimizing the design and fitting the logic into the targeted PLD. Logical optimization in *Warp2* is accomplished with Espresso algorithms. The optimized design is fed to the *Warp2* fitter, which applies the design to the specified target PLD. The *Warp2* fitter supports manual or automatic pin assignments as well as automatic selection of D or T flip-flops. After the optimization and fitting step is complete, *Warp2* automatically creates a JEDEC file for the specified PLD or CPLD.

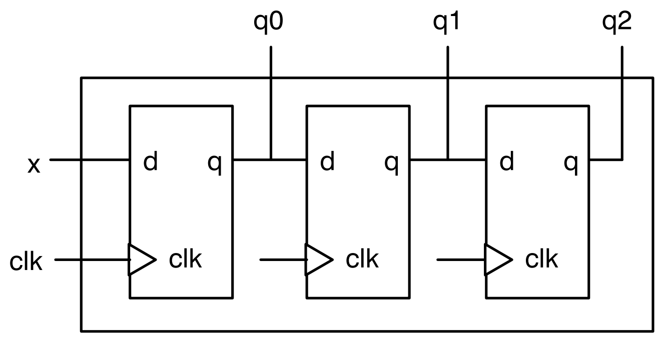

Figure 3. Three-Bit Shift Register Circuit Design

Simulation

Warp2 is delivered with Cypress's NOVA Simulator. NOVA features a graphical waveform simulator that can be used to simulate designs generated in *Warp2*. The NOVA simulator provides functional simulation and features interactive waveform editing and viewing. The simulator also provides the ability to probe internal nodes, automatically generate clocks and pulses, and to generate JEDEC test vectors from simulator waveforms.

Programming

The result of *Warp2* compilation is a JEDEC file that implements the input design in the targeted PLD. Using the JEDEC file, Cypress PLDs and CPLDs can be programmed on Cypress's Impulse3 programmer or on any qualified third-party programmer.

System Requirements

For PCs

IBM PC-AT or equivalent (486 or higher recommended)

PC-DOS version 3.3 or higher

16 Mbytes of RAM

35-Mbyte hard disk space

1.44-Mbyte floppy disk drive

Two- or three-button mouse

Windows Version 3.1

For Sun Workstations

SPARC CPU
Sun OS 4.1.1 or later
16 Mbytes of RAM
1.44-Mbyte 3½-inch disk drive

Ordering Information

CY3121 *Warp2* for Windows PLD Compiler includes:
- 3½-inch, 1.4-Mbyte floppy disks
- *Warp2* User's Guide
- *Warp2* Synthesis Reference
- Registration Card

CY3126 *Warp2* for Sun PLD Compiler includes:
- 3½-inch, 1.4-Mbyte floppy disks
- *Warp2* User's Guide
- *Warp2* Synthesis Reference
- Registration Card

Document #: 38−00433