Simulation of Cypress CPLDs with Mentor's QuickSim II

Simulation of Cypress CPLDs and smaller programmable logic devices in the Mentor Graphics environment is possible without the need for purchasing third party simulation models. Designs ranging the entire density span of Cypress programmable logic devices can quickly be placed into a form that can be imported into the mentor QuickSim II environment. It will be assumed that the person attempting to perform this task has some familiarity with the Cypress *Warp*™ software and Mentor QuickSim II.

After a design has been successfully compiled in the *Warp* environment, four easy steps are needed to get the design in the final form that QuickSim II can understand. The first step is to create a Viewlogic VHDL simulation model from the *Warp* design environment. Please refer to your *Warp* documentation for detailed instructions on how to do so. The second step is to do some slight editing to the VHDL files associated with the part family chosen for the design and the VHDL file exported from *Warp*. Thirdly, a 'wrapper' file must be constructed around the output file to convert Viewlogic I/O to Quicksim I/O. Finally, all the files edited and produced above are placed onto a disk for transfer into the Mentor environment.

The Four Steps for Simulating Cypress PLDs & CPLDs in the QuickSim II Environment:

1. Export Viewlogic VHDL file from *Warp*.
2. Small editing to the VHDL file.
3. Create the wrapper file.
4. Transfer files to Mentor environment and compile.

Let's take a detailed look at the four above steps.

Step 1: Export Viewlogic VHDL File from *Warp*

The first step in the process is to generate a Viewlogic VHDL simulation file from the *Warp* design tool. Please refer to the *Warp* documentation for instructions on how to generate this file. Once the *Warp* design tool is run and your VHDL has been created, you will find it in the /vhd subdirectory of your current project. The filename will be the same as your source code top-level filename. Once you have located this file, you are ready for step 2.

Step 2: Small Editing to the VHDL File

The file that was just written out is in a format that Viewsim understands. To put the file in a format for QuickSim, first we modify the beginning of the file as shown in *Figure 1*. Notice that one line is commented out and two are added. The second line added will vary depending upon which part was chosen when the design was compiled. The last changes that need to be made in this file are to add the lines as shown in *Figure 2*. The lines shown in *Figure 2* will change depending on your target device.

The proper 'use work.c{devicename}p.all;' clause and 'FOR ALL:...' statement for each target device is listed in Appendix A.

Each of the files listed in the 'FOR ALL:...' statements (c37xclk.vhd, c37xinp.vhd, ..., and

```
--  CYPRESS NOVA XVL Structural Architecture
--  JED2VHD Reverse Assembler - Ver 0.09 Oct 26, 1993
--    Viewlogic HDL File: FORDT.vhd
--      Date: Tue Oct 18 22:15:45 1994
--  Disassembly from Jedec file for: c371
--  Device Ordercode is: CY7C371-143JC
-- library primitive;  **** Commented out this line ****
use work.pack1076.all; **** Added these two lines ****
use work.c37xp.all;    **** work.c37xp.all is used for any Flash370 device ***
```

Figure 1. First Modifications to Example File

```
ARCHITECTURE DSMB of design_FORDT is

-- stuff that needs to be added for MENTOR system 1076

FOR ALL: c37xclk use entity work.c37xclk(sim);-- These statements will
FOR ALL: c37xinp use entity work.c37xinp(sim);-- change with different target
FOR ALL: c37xm use entity work.c37xm(sim);    -- devices and/or families.
FOR ALL: c37xmux use entity work.c37xmux(sim);
FOR ALL: c37xoreg use entity work.c37xoreg(sim);
FOR ALL: c37xprod use entity work.c37xprod(sim);
--
```

Figure 2. Additions to the Architecture

c37xprod.vhd) also have small changes that must be made (see *Figure 3*). For ease of use, the Cypress BBS contains all of these files pre-modified and they can be downloaded at your convenience. The files are in a self-extracting archive file called: VHDL_SIM.EXE.

After completing all of these modifications, step 2 is complete.

```
------------------------------------------------------------------------
--    Entity / Architecture pairs
--    For c37xclk
------------------------------------------------------------------------
--
--  Copyright Cypress Semiconductor Corporation, 1994
--    as an unpublished work.
--
--      $Id: c37xclk.vhd,v 1.8 1994/09/22 20:08:23 hemmert Exp $
--
use work.pack1076.all;    -- This one line must be added to the top of
                          -- every device library (FOR ALL:...) file
```

Figure 3. Modification to FOR ALL:... Files, If Premodified Files Are Not Used

Step 3: Create the Wrapper File

Creating the wrapper file is accomplished by simply performing multiple cut-and-pastes and search-and-replaces. The wrapper is used to translate vlbits (Viewlogic bits) to qsim_states (Mentor simulation states). To do this, a pair of functions is used. One function translates from qsim_state to vlbit, and the other translates vlbit to qsim_state. The first step is to copy the **entity** from the *Warp*-produced VHDL file into the file that contains our two functions. Now with your text editor, search for **vlbit** and replace it with **qsim_state** *(Figure 4)*. This completes the entity of the wrapper.

```
--  CYPRESS NOVA XVL Structural Architecture
--  JED2VHD Reverse Assembler – Ver 0.09 Oct 26, 1993
--    Viewlogic HDL File: FORDT.vhd
--      Date: Tue Dec 27 16:23:47 1994
--  Disassembly from Jedec file for: c22v10
--  Device Ordercode is: PAL22V10C-10JC

use work.pack1076.all;  --  This line is part of the standard template.
use work.c22v10p.all;   --  For a Flash370 device, use work.c37xp.all;

LIBRARY mgc_portable;
USE mgc_portable.qsim_logic.all;

ENTITY FORDT IS
     PORT(
  clock      : in  qsim_state ;
  right      : in  qsim_state ;
  left       : in  qsim_state ;
  flash      : in  qsim_state ;
  brake      : in  qsim_state ;
  node6      : in qsim_state ;
  node7      : in qsim_state ;
  node8      : in qsim_state ;
  node9      : in qsim_state ;
  node10     : in qsim_state ;
  node11     : in qsim_state ;
  node12     : in qsim_state ;
  node13     : in qsim_state ;
  r_outer    : inout qsim_state ;
  r_inner    : inout qsim_state ;
  l_middle   : inout qsim_state ;
  vl1i139_H2 : inout qsim_state ;
  vl1i137_H2 : inout qsim_state ;
  vl1i136_H2 : inout qsim_state ;
  vl1i138_H2 : inout qsim_state ;
  l_inner    : inout qsim_state ;
  l_outer    : inout qsim_state ;
  r_middle   : inout qsim_state ;
  node24     : in qsim_state
     );
END FORDT;
```

Figure 4. The Wrapper Entity

The next step is to create the architecture of the wrapper. To start this step, first type in the function template that converts vlbits to/from qsim_states, as mentioned above (*Figure 5*).

Next, the design that we are wrapping around is called in as a component. The port mapping for the component is created by simply copying the original entity used above. This time, no search-and-replace is needed (*Figure 6*).

The final step in creating the wrapper is instantiating the design as a component and hooking up the I/O to the wrapper through the functions listed in *Figure 5*. For the port map, start by copying the entity from the top of the file once again. For all signals of type in, use the qsim_state2vlbit function on the right side of the port map. For all inout, vlbit2qsim_state is used on the left and qsim_state2vlbit is used once again on the right (*Figure 7*).

This ends the creation of the wrapper. The wrapper is shown in its entirety in Appendix B. The more I/O pins a device has, the larger the wrapper file will be. However, because the creation is simply several copy-and-pastes and search-and-replaces, the size of the design will not seriously increase the amount of time needed to put the wrapper together.

Step 4: Transfer Files to Mentor Environment and Compile

We are now ready to transfer the design into the Mentor environment. In addition to the VHDL file we modified (that was generated by *Warp*), copy the files listed in Appendix B to your transfer media (tape, floppy, punch cards?). Place these files in a directory in the Mentor environment. Compile the files in the following order:

```
ARCHITECTURE structural OF fordt_wrapper IS

 --  Mapping functions for viewlogic states to/from  qsim_state

function qsim_state2vlbit (i : in qsim_state) return vlbit is
   begin
      case i is
         when '0' =>      return '0';
         when '1' =>      return '1';
         when 'X' =>      return 'X';
         when 'Z' =>      return 'Z';
      end case;
   end;

-------------------------------------------------------------------
function vlbit2qsim_state (i : in vlbit) return qsim_state is
   begin
      case i is
         when '0' =>      return '0';
         when '1' =>      return '1';
         when 'X' =>      return 'X';
         when 'Z' =>      return 'Z';
      end case;
   end;
-------------------------------------------------------------------
```

Figure 5. Functions for vlbit to/from qsim_state

```
component design_FORDT
  PORT(
    clock      : in  vlbit ;
    right      : in  vlbit ;
    left       : in  vlbit ;
    flash      : in  vlbit ;
    brake      : in  vlbit ;
    node6      : in vlbit ;
    node7      : in vlbit ;
    node8      : in vlbit ;
    node9      : in vlbit ;
    node10     : in vlbit ;
    node11     : in vlbit ;
    node12     : in vlbit ;
    node13     : in vlbit ;
    r_outer    : inout vlbit ;
    r_inner    : inout vlbit ;
    l_middle   : inout vlbit ;
    vl1i139_H2 : inout vlbit ;
    vl1i137_H2 : inout vlbit ;
    vl1i136_H2 : inout vlbit ;
    vl1i138_H2 : inout vlbit ;
    l_inner    : inout vlbit ;
    l_outer    : inout vlbit ;
    r_middle   : inout vlbit ;
    node24     : in vlbit
  );
end component;

FOR ALL: design_fordt USE ENTITY work.design_fordt;
```

Figure 6. Calling in the Original Design as a Component

1. pack1076.vhd
2. c{devicename}p.vhd
   Example: c37xp.vhd or c22v10p.vhd
3. The rest of the device library files listed for your target device in Appendix B.
4. The wrapper file.

After successful compilation, the design is ready to be connected to a symbol for board and system-level simulation.

```
BEGIN
-------------------------
-- instantiate the design
-------------------------
u1: design_fordt
  port map
    (
      clock    => qsim_state2vlbit(clock),
      right    => qsim_state2vlbit(right),
      left     => qsim_state2vlbit(left),
      flash    => qsim_state2vlbit(flash),
      brake    => qsim_state2vlbit(brake),
      node6    => qsim_state2vlbit(node6),
      node7    => qsim_state2vlbit(node7),
      node8    => qsim_state2vlbit(node8),
      node9    => qsim_state2vlbit(node9),
      node10   => qsim_state2vlbit(node10),
      node11   => qsim_state2vlbit(node11),
      node12   => qsim_state2vlbit(node12),
      node13   => qsim_state2vlbit(node13),
      vlbit2qsim_state(r_outer)   => qsim_state2vlbit(r_outer),
      vlbit2qsim_state(r_inner)   => qsim_state2vlbit(r_inner),
      vlbit2qsim_state(l_middle)  => qsim_state2vlbit(l_middle),
      vlbit2qsim_state(l_middle)  => qsim_state2vlbit(l_middle),
      vlbit2qsim_state(vlli137_H2) => qsim_state2vlbit(vlli137_H2),
      vlbit2qsim_state(vlli136_H2) => qsim_state2vlbit(vlli136_H2),
      vlbit2qsim_state(vlli138_H2) => qsim_state2vlbit(vlli138_H2),
      vlbit2qsim_state(l_inner)   => qsim_state2vlbit(l_inner),
      vlbit2qsim_state(l_outer)   => qsim_state2vlbit(l_outer),
      vlbit2qsim_state(l_middle)  => qsim_state2vlbit(l_middle),
      node24   => qsim_state2vlbit(node24)
    );
end structural;
```

Figure 7. Instantiating and Mapping the Design

Appendix A. List of Files Needed for Mentor QuickSim II by Part Type

Part Type	Files Needed	Line Added Before the Entity	Lines Added in the Architecture
16L8	C16L8P.VHD PACK1076.VHD	use work.c16l8p.all; use work.pack1076.all;	
16R4	C16R4P.VHD PACK1076.VHD	use work.c16r4p.all; use work.pack1076.all;	
16R6	C16R6P.VHD PACK1076.VHD	use work.c16r6p.all; use work.pack1076.all;	
16R8	C16R8P.VHD PACK1076.VHD	use work.c16r8p.all; use work.pack1076.all;	
16V8	C16V8M.VHD C16V8P.VHD PACK1076.VHD	use work.c16v8p.all; use work.pack1076.all;	FOR ALL: c16v8m use entity work.c16v8m(sim);
20G10	C20G10CM.VHD C20G10CP.VHD C20G10M.VHD C20G10P.VHD PACK1076.VHD	use work.c20g10p.all; use work.pack1076.all;	FOR ALL: c20g10cm use entity work.c20g10cm(sim); FOR ALL: c20g10cp use entity work.c20g10cp(sim); FOR ALL: c20g10m use entity work.c20g10m(sim);
20RA10	C20RA10M.VHD C20RA10P.VHD PACK1076.VHD	use work.c20ra10p.all; use work.pack1076.all;	FOR ALL: c20ra10m use entity work.c20ra10m(sim);
22V10	C22V10M.VHD C22V10P.VHD PACK1076.VHD	use work.c22v10p.all; use work.pack1076.all;	FOR ALL: c22v10m use entity work.c22v10m(sim);
22VP10	C22VP10M.VHD C22VP10P.VHD PACK1076.VHD	use work.c22vp10p.all; use work.pack1076.all;	FOR ALL: c22vp10m use entity work.c22vp10m(sim);
7C331	C331CKMX.VHD C331M.VHD C331P.VHD PACK1076.VHD	use work.c331p.all; use work.pack1076.all;	FOR ALL: c331ckmx use entity work.c331ckmk(sim); FOR ALL: c331m use entity work.c331m(sim);
7C335	C335CKMX.VHD C335H.VHD C335IREG.VHD C335M.VHD C335P.VHD PACK1076.VHD	use work.c335p.all; use wor.pack1076.all;	FOR ALL: c335ckmx use entity work.c335ckmx(sim); FOR ALL: c335h use entity work.c335h(sim); FOR ALL: c335ireg use entity work.c335ireg(sim); FOR ALL: c335m use entity work.c335m(sim);

Appendix A. List of Files Needed for Mentor QuickSim II by Part Type (continued)

Part Type	Files Needed	Line Added Before the Entity	Lines Added in the Architecture
7C34X	C34XCKMX.VHD C34XEXIN.VHD C34XEXP.VHD C34XH.VHD C34XIN.VHD C34XM.VHD C34XPIA.VHD C34XP.VHD PACK1076.VHD	use work.c34xp.all; use work.pack1076.all;	FOR ALL: c34xckmx use entity work.c34xckmx(sim); FOR ALL: c34xexin use entity work.c34xexin(sim); FOR ALL: c34xexp use entity work.c34xexp(sim); FOR ALL: c34xh use entity work.c34xh(sim); FOR ALL: c34xin use entity work.c34xin(sim); FOR ALL: c34xm use entity work.c34xm(sim); FOR ALL: c34xpia use entity work.c34xpia(sim);
7C37X	C37XCLK.VHD C37XINP.VHD C37XM.VHD C37XMUX.VHD C37XOREG.VHD C37XPROD.VHD C37XP.VHD PACK1076.VHD	use work.c37xp.all; use work.pack1076.all;	FOR ALL: c37xclk use entity work.c37xclk(sim); FOR ALL: c37xinp use entity work.c37xinp(sim); FOR ALL: c37xm use entity work.c37xm(sim); FOR ALL: c37xmux use entity work.c37xmux(sim); FOR ALL: c37xoreg use entity work.c37xoreg(sim); FOR ALL: c37xprod use entity work.c37xprod(sim);

Appendix B. The Wrapper

```
--  CYPRESS NOVA XVL Structural Architecture
--  JED2VHD Reverse Assembler – Ver 0.09 Oct 26, 1993
--    Viewlogic HDL File: FORDT.vhd
--      Date: Tue Dec 27 16:23:47 1994
--  Disassembly from Jedec file for: c22v10
--  Device Ordercode is: PAL22V10C–10JC

use work.pack1076.all;  --  This line is part of the standard template.
use work.c22v10p.all;   --  For a 37x part, use work.c37xp.all;

LIBRARY mgc_portable;            -- These lines are added for Mentor's
USE mgc_portable.qsim_logic.all; -- System 1076 VHDL compiler

ENTITY FORDT IS
     PORT(
  clock      : in  qsim_state ;
  right      : in  qsim_state ;
  left       : in  qsim_state ;
  flash      : in  qsim_state ;
  brake      : in  qsim_state ;
  node6      : in qsim_state ; --Notice that unused pins are assigned a
  node7      : in qsim_state ; --node number equivalent to their pin number.
  node8      : in qsim_state ;
  node9      : in qsim_state ;
  node10     : in qsim_state ;
  node11     : in qsim_state ;
  node12     : in qsim_state ;
  node13     : in qsim_state ;
  r_outer    : inout qsim_state ;
  r_inner    : inout qsim_state ;
  l_middle   : inout qsim_state ;
  vl1i139_H2 : inout qsim_state ;
  vl1i137_H2 : inout qsim_state ;
  vl1i136_H2 : inout qsim_state ;
  vl1i138_H2 : inout qsim_state ;
  l_inner    : inout qsim_state ;
  l_outer    : inout qsim_state ;
  r_middle   : inout qsim_state ;
  node24   : in qsim_state
     );
END FORDT;

ARCHITECTURE structural OF fordt_wrapper IS

 --  Mapping functions for viewlogic states to/from  qsim_state
```

Appendix B. The Wrapper (continued)

```
function qsim_state2vlbit (i : in qsim_state) return vlbit is
   begin
      case i is
         when '0' =>      return '0';
         when '1' =>      return '1';
         when 'X' =>      return 'X';
         when 'Z' =>      return 'Z';
      end case;
   end;

-----------------------------------------------------------------------
function vlbit2qsim_state (i : in vlbit) return qsim_state is
   begin
      case i is
         when '0' =>      return '0';
         when '1' =>      return '1';
         when 'X' =>      return 'X';
         when 'Z' =>      return 'Z';
      end case;
   end;
-----------------------------------------------------------------------
component design_FORDT
     PORT(
  clock      : in  vlbit ;
  right      : in  vlbit ;
  left       : in  vlbit ;
  flash      : in  vlbit ;
  brake      : in  vlbit ;
  node6      : in vlbit ;
  node7      : in vlbit ;
  node8      : in vlbit ;
  node9      : in vlbit ;
  node10     : in vlbit ;
  node11     : in vlbit ;
  node12     : in vlbit ;
  node13     : in vlbit ;
  r_outer    : inout vlbit ;
  r_inner    : inout vlbit ;
  l_middle   : inout vlbit ;
  vl1i139_H2 : inout vlbit ;
  vl1i137_H2 : inout vlbit ;
  vl1i136_H2 : inout vlbit ;
  vl1i138_H2 : inout vlbit ;
  l_inner    : inout vlbit ;
  l_outer    : inout vlbit ;
  r_middle   : inout vlbit ;
  node24   : in vlbit
     );
end component;
```

Appendix B. The Wrapper (continued)

```
FOR ALL: design_fordt USE ENTITY work.design_fordt;
BEGIN

-------------------------
-- instantiate the design
-------------------------

u1: design_fordt
   port map
      (
  clock    => qsim_state2vlbit(clock),
  right    => qsim_state2vlbit(right),
  left     => qsim_state2vlbit(left),
  flash    => qsim_state2vlbit(flash),
  brake    => qsim_state2vlbit(brake),
  node6    => qsim_state2vlbit(node6),
  node7    => qsim_state2vlbit(node7),
  node8    => qsim_state2vlbit(node8),
  node9    => qsim_state2vlbit(node9),
  node10   => qsim_state2vlbit(node10),
  node11   => qsim_state2vlbit(node11),
  node12   => qsim_state2vlbit(node12),
  node13   => qsim_state2vlbit(node13),
  vlbit2qsim_state(r_outer) => qsim_state2vlbit(r_outer),
  vlbit2qsim_state(r_inner) => qsim_state2vlbit(r_inner),
  vlbit2qsim_state(l_middle) => qsim_state2vlbit(l_middle),
  vlbit2qsim_state(l_middle) => qsim_state2vlbit(l_middle),
  vlbit2qsim_state(vlli137_H2) => qsim_state2vlbit(vlli137_H2),
  vlbit2qsim_state(vlli136_H2) => qsim_state2vlbit(vlli136_H2),
  vlbit2qsim_state(vlli138_H2) => qsim_state2vlbit(vlli138_H2),
  l_inner  vlbit2qsim_state(l_inner) => qsim_state2vlbit(l_inner),
  vlbit2qsim_state(l_outer) => qsim_state2vlbit(l_outer),
  vlbit2qsim_state(l_middle) => qsim_state2vlbit(l_middle),
  node24 => qsim_state2vlbit(node24)
         );
end structural;
```

Warp is a trademark of Cypress Semiconductor Corporation.