

OrCAD Support for Atmel PLDs

- Atmel Device Support for OrCAD PLD 386+
- ATV750/ATV750B Device Family
- ATV2500/ATV2500B Device Family
- For ATF1500 support please contact Atmel PLD applications
- For 16V8, 20V8, and 22V10, use the standard OrCAD PLD library

OrCAD's Programmable Logic Design Tools 386+ is used to create, compile, test, and fit logic for programmable devices. This application note describes how to use OrCAD PLD 386+ to create an object file to program the Atmel PLD and CPLD devices.

Overview

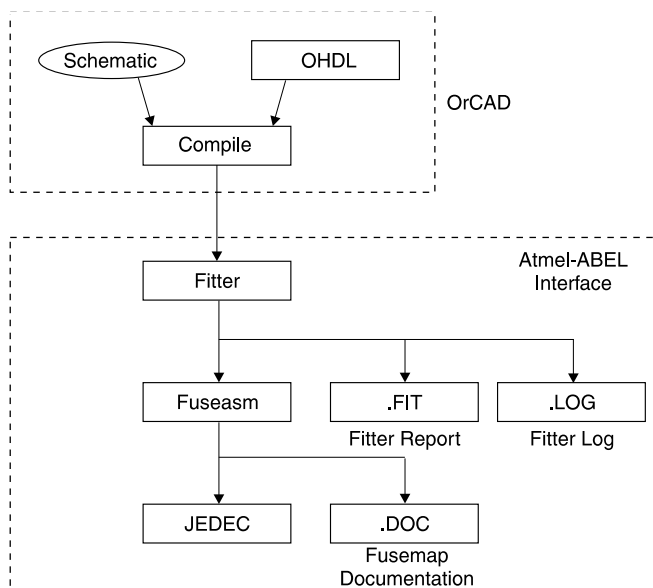
OrCAD PLD 386+ can be used in conjunction with Atmel-ABEL Fitter software (available from Atmel) to create JEDEC files. Figure 1 illustrates the programs that are involved.

PLD 386+ with Atmel ABEL Software

The OrCAD Hardware Description Language (OHDL) allows you to express your design using indexed boolean equations, numerical maps, state machines, etc. When used in conjunction with SDT 386+, PLD 386+ also allows you to design with schematics or a mix to schematics and OHDL. Fit Logic with the Atmel-ABEL Device Fitter translates your OHDL design into an Open-ABEL (.tt2) logic interchange file. The JEDEC writer, FUSEASM, creates a programming file.

(continued)

Figure 1. Atmel/OrCAD Interface



0718A

PLD 386+ with Atmel ABEL Software (Continued)

Related documents

Atmel SmartPart ATMEL 750/2500 Device Fitter
 Intelligent Device Fitting Software User Manual
 DATA I/O Programmable Logic Design Tools Language
 Guide
 Programmable Logic Design Tools 386+ User's Guide
 ATF1500 CPLD Device Fitter Manual

Setting up Atmel-ABEL

This section describes the Atmel-ABEL software system requirements that should be satisfied before running the Fit Logic processor. These notes reflect the installation and setup information described in Atmel's *ATMEL-ABEL Design Software, Installation Guide* document.

Note: To generate a JEDEC file for an Atmel PLD family device with PLD 386+ requires a complete installation of the Atmel-ABEL Device Fitter.

1. The following environment variables must be set. The following lines should be added to the AUTOEXEC.BAT: SET ABEL4DEV=[path]\lib4\
 [SET ABEL5DEV=[path]\lib5]
 where [path] is the subdirectory preceding the Atmel-ABEL environment.
2. The Atmel software node must be added to the PATH of the AUTOEXEC.BAT. An example path is shown below:
 PATH C:\DOS;C:\ORCADEXE;C:\ATMEL;
3. The CONFIG.SYS should contain the following parameters:
 files=20
 buffers=20
 shell=c:\command.com/p /e: 102
4. Attach the DATA I/O software security key to the parallel port.

Note: The Atmel-ABEL installation software automatically implements the modifications to your AUTOEXEC.BAT and CONFIG.SYS files.

Figure 2. Implicit and Explicit Signal Assignment Example

```

Explicit assignment
|1:  SIGNALA,
|2:  SIGNALB,
:
|55: SIGNALn1,

Implicit assignment
|pins: (SIGNALA, SIGNALB, ... SIGNALn), |External pin signals
|nodes: (SIGNALA1, SIGNALB1,... SIGNALn1) |Internal node signals
  
```

How to target an Atmel Device

To generate the proper logic interchange file, identify the device architecture in your OHDL source file. If you are using the source composition technique the sample source lines below demonstrate how to target any device of the Atmel Device Family:

ATV750 (DIP)
 ATV750C (PLCC)
 ATV750B (DIP)
 ATV750BC (PLCC)
 ATV2500 (DIP)
 ATV2500C (PLCC)
 ATV2500B (DIP)
 ATV2500BC (PLCC)

If you are using the schematic capture technique, use the a **PLACE Text** command in **Edit Schematic Logic**, then use one of the type keywords below:

Type: "ATV750"
 Type: "ATV750C"
 Type: "ATV750B"
 Type: "ATV750BC"
 Type: "ATV2500"
 Type: "ATV2500C"
 Type: "ATV2500B"
 Type: "ATV2500BC"

How to assign signals

You can assign signals explicitly in the *design.pld* OHDL source file or allow them to float. If a signal floats, the Atmel-ABEL Device Fitter will assign the signal to a specific **pins:** or **nodes:** To identify floating signals to the logic compiler, use the **pins:** and **nodes:** keywords. The source lines in Figure 2 demonstrate how to assign signals to pins and nodes explicitly or allow them to float. If you are using the schematic capture technique, use the **Get? PINPAD** or **NODEPAD** command in **Edit Schematic Logic**.

5-Bit Counter Example Design

This design example illustrates implementing a 5-bit counter with a state machine into the ATV750 (PLCC) device.

Change to the TUTOR design

Use the Design Management Tools to change to the TUTOR design. Display the Programmable Logic tools screen.

Adjust the Tool Set Configuration

1. Set Compile Source, and Configure PLD Tools. The Configure PLD screen appears.
2. Pan down to the Processing Options. Select Create error message file with line and column numbers.
3. Pan back up to the top, select OK to save the change.

Capture the design

1. The source OHDL for the counter is shown in the Figure 3 *5COUNT.PLD OHDL Sample*.

Use the **Edit File** editor to key in the source file. Select **Edit File** and Execute. Type in 5COUNT.PLD as the File to Edit. Select OK. Type in the source file. Save the file. Exit the text editor.

Note: to save time, you only have to key in source lines that begin with a vertical bar |p, any other line is regarded as a comment by the logic compiler.

Compile the Design

1. Select Compile Source, Local Configuration, and Configure LOGIC. The Configure Compile Source screen displays.
2. In the File area, select 5COUNT.PLD from the list box. The Source entry box displays the name of the selected file.
3. Select OK to save the configuration.
4. Select **Compile Source** and Execute. As it processes, a view window appears near the bottom of the screen.

If any errors or warnings are encountered by the compiler, the design environment will allow you to View Source, View Output, or (ignore the messages) OK. If you View Source, your text editor will appear with the 5COUNT.PLD file. If you are using the Stony Brook M2EDIT Text Editor, the function key will display any syntax errors that may have been generated by the source file.

Edit and recompile until the logic compiler returns to the Programmable Logic Tools screen without errors or warnings. The logic interchange file, 5COUNT.PLA, and a report about the design compilation, 5COUNT.LST, can be viewed by the **Edit File** editor.

Test the Logic

1. Select **Test Logic**, **Logic Configuration**, and **Configure VECTORS**. The **Configure Test Logic** screen displays.
2. In the File area, select 5COUNT.PLD from the list box. The Source entry box displays the name of the selected file.
3. Select **Initialize internal registers to Low**.
4. Select **OK** to save the configuration.
5. Select Test Logic and Execute. As it processes the design environment displays in "full-screen" mode the activity of the logic tester.
6. The Programmable Logic Tools screen reappears when **Test Logic** is complete. The 5COUNT.LOG test log can be viewed by the **Edit File** editor.

Fit the Logic

1. Select **Fit Logic**, Local Configuration, and Configure FITLOGIC. The Configure Fit Logic screen displays. In the File area, select 5COUNT.PLA from the list box. The Source entry box displays the name of the selected file. The information message **Target Device: ATV750c (PLCC)** appears.
2. Select OK to save the configuration.
3. Select **Fit Logic** and Execute. As it processes the design environment displays in "full-screen" mode the activity of the Atmel-ABEL software.
4. The Programmable Logic Tools screen reappears when **Fit Logic** is complete. The Open-ABEL logic interchange 5COUNT.TT3, and reports about the fitter process (5COUNT.TT3 and 5COUNT.RPT) can be viewed by the **Edit File** editor.
5. Assemble the fuse map. Run from the DOS command line:

```
fuseasm 5count-0 5count.jed
```

To append test vectors to the JEDEC file, rerun **Test Logic**.

Expressing the logic as a schematic

A similar process is used to capture the design in schematics. Use the **Edit Schematic Logic** editor to draft the logic and **Convert Schematic Logic to Source** to generate an OHDL source file.

Figure 3. 5COUNT.PLD OHDI. Sample

```

-----
FILE:          5COUNT.PLD
               This test case was translated from the ABEL
               design file COUNT5.ABL
INCLUDE FILES:  None
-----
|ATV750C
|16: Q2                |Output signals
|17: Q0,
|Pins: (Q1,
|   CLOCK                |Input signals
|   CLEAR,
|   OUTEN)
|
|   Title:          "count to five counter"
-----
|   Conditioning:  OUTEN ?? Dff(Q[2-0],CLOCK)
|   Procedure:    Q[2-0] {
|   States:      S0 = 0,
|               S1 = 1,
|               S2 = 2,
|               S3 = 3,
|               S4 = 4
|
|   S0. |CLEAR?      -> S1                |If |CLEAR then S1 else S0
|       -> S0
|   S1. |CLEAR?      -> S2                |If |CLEAR then S2 else S0
|       -> S0
|   S2. |CLEAR?      -> S3                |If |CLEAR then S3 else S0
|       -> S0
|   S3. |CLEAR?      -> S4                |If |CLEAR then S4 else S0
|       -> S0
|   S4. -> S0
|   S5. -> S0                |Decode illegal states
|   S6. -> S0
|   S7. -> S0 }
-----
|Vectors: {
|  d (CLOCK)c, CLEAR, OUTEN, "-->", Q[2-0]
|  s OUTEN
|  s CLEAR
|  t CLOCK
|  c CLEAR
|  t CLOCK=10(0,1) }

```