Converting ABEL Design Files to CUPL

This application note is intended to assist users in converting designs written in ABEL-HDL language to CUPL. It also includes an example in ABEL and equivalent representation in CUPL. Atmel® no longer offers ABEL compilers. Instead users are encouraged to convert their designs to CUPL and use Atmel Design software tools such as Atmel-WinCUPL™ or ProChip Designer™.

Background for ABEL and CUPL

ABEL-HDL and CUPL-HDL are behavioral design languages used to describe logic circuits at a high level. ABEL evolved over the eighties and early nineties as a language that was written to take advantage of the architectural features of an EPLD. As late as 1995, Atmel continued to offer ABEL V5.1 (DOS-based program). This required a Dongle (Key from Data I/O™ Corp. WA) to be plugged into the parallel port of a PC. Subsequently, Atmel offered an EDA package called Atmel-Synario™ that included a windows version of the ABEL compiler until the year 2000. Atmel-Synario V4.11 was an OEM version specific for Atmel EPLDs and ABEL 6.5 (windows version) was the last version of ABEL-HDL offered by Atmel as part of this package. Subsequently, Data I/O spun off Synario as an EDA company and a little later Synario's assets became a part of MINC Inc., another EDA Company. MINC then re-sold specific tools from the Synario package to Xilinx®, Inc.

Logical Devices, Inc. developed CUPL and the structure of the language has not changed much for the last two decades. Atmel offered a DOS version of CUPL until the late nineties. The most recent DOS version of Atmel-WinCUPL shipped by Atmel was Rev 4.8. Subsequently a windows version of CUPL (Rev 5.x) was offered and called Atmel-WinCUPL.

ABEL/CUPL Design File Conversion

Application Note

Rev. 3303A–PLD–08/02

Compiler and Tool Options

The CUPL compiler is available in Atmel-WinCUPL Version 5.2.16 software as well as part of Atmel-ProChip Designer that uses a Third Party tool from Altium™ called Design Explorer™ 99SE.

For simple designs, users are encouraged to use Atmel-WinCUPL which is a free tool and available for download from Atmel's website.

The ABEL compiler (Rev. 6.5) used to compile the ABEL example was part of Atmel-Synario Version 4.11 software, which is no longer offered.

Process of Conversion of an ABEL Example File to CUPL

The conversion is presented in the form of a Table (Table 1) and shows comparative implementation in ABEL and CUPL. Users can first go through this example and then refer to “Overview of Syntax Differences between ABEL and CUPL” on page 7 for Syntax details. Simulation files are not required for every design unless users specifically want to functionally generate a set of Test Vectors that can be applied on a Third Party Programmer hardware. The process of writing Test vector files is listed in the section titled “Converting an ABEL Simulation Input File to CUPL” on page 5.

Please note that in ABEL, it is possible to include Test vectors as part of the main source file. The ABEL compiler will then extract the test vectors (.TMV) for simulation purposes and to append the test vectors to the Jedec file.

Description of Example

The following example in Table 1 shows how to implement a 4-bit loadable counter that can count up (from 0-15 in decimal mode) as well as count down (15-0). In this example, the counter resets to zero if rst is one. If ld is set, then the output (q3..q0) will be set to the input (d3..d0). If cnten is set, then the counter is enabled and will count up/down depending on the state of u_d (control pin). If cnten is not set the output will be held to the last count.

Table 1. 4-bit Loadable Counter Implementation

ABEL Source File (.ABL)	CUPL Source File (.pld)
Module unicnt Interface (d3..d0, clk, rst, cnten, ld, u_d -> q3..q0); Title '4 bit counter with load, reset, count up, count down'; //Constants X, C, Z = .X., .C., .Z. ;	Name counter; PartNo 00 ; Date 6/30/02; Revision 01; Designer Engineer; Company XYZ; Assembly None; Location San Jose; Device virtual; /*See Table 6 for description of each field in the header section*/

//Inputs
d3..d0 pin;
clk pin;
rst pin;
cnten pin;
ld pin;
u_d pin;
	/* Input */
	pin = [d3..0];
	pin = clk;
	pin = rst;
	pin = cnten;
	pin = ld;
	pin = u_d;

//Output
q3..q0 pin is_type 'reg';
//Counter output, user can choose reg_d to select //D type registers or reg_t for T-type Registers.
	/* Output */
	pin = [q3..0];

//Sets
data = [d3..d0]; //Data Set
count = [q3..q0]; //Counter Set
// Forming group of signals into a vector
MODE = [cnten,ld,u_d];
	/* Data Set */
	field data = [d3..0];
	field count = [q3..0]; /* field is a way to group a set of signals */
	/* Forming a group of signals into a vector */
	field MODE = [cnten,ld,u_d];

// Selecting different modes base on vector values
// possible values are 0, 1, or don't cares
LOAD = (MODE == [X, 1, X]);
HOLD = (MODE == [0, 0, X]);
UP = (MODE == [1, 0, 1]);
DOWN = (MODE == [1, 0, 0]);
	/* Selecting different modes based on vector values possible values are 0, 1, or don't cares */
	load = MODE:'b'X1X;
	hold = MODE:'b'00X;
	up = MODE:'b'101;
	down = MODE:'b'100;

Table 1. 4-bit Loadable Counter Implementation (Continued)

ABEL Source File (.ABL)	CUPL Source File (.pld)

Equations
when LOAD then count := data;
// Abel does things sequentially
else when UP then count := count + 1;
// Count up logic
else when DOWN then count := count - 1;
// Count down logic
else when HOLD then count := count;
// Hold otherwise
count.clk = clk; // Assign Flip-Flop clock pin
count.ar = rst; // Assign asynchronous reset for Flip-Flop
END unicnt //This specifies the end of the //equations section of the module
	/* The following will create a Moore FSM where the output will be a function of the state */
	SequenceD count { /* Explicitly choose D-FF, */
	$REPEAT i = [0..15] /* This macro will expand from 0 to 15 */
	Present 'h'{i} /* similar to case statement for each state */
	If !load & up Next 'h'{(i+1)%16}; /* Logic for count up */
	If !load & down Next 'h'{((i-1)+16)%16}; /* Logic for count down */
	If !load & hold Next 'h'{i}; /* Logic for hold */
	$REPEND}
	APPEND count.d = load & data; /* This is when we want to load */
	count.AR = rst; /* Asynchronous reset */
	count.ck = clk;

ABEL Test Vectors	**CUPL Test Vectors**
test_vectors ([clk, rst, cnten, ld, u_d, data] -> count) [.c., 1, 0, 0, 0, 0] -> 0; [.c., 0, 0, 1, 0, 8] ->8 ; [.c., 0, 1, 0, 1, 8] -> 9; [.c., 0, 1, 0, 1, 8] -> 10; [.c., 0, 1, 0, 1, 8] -> 11; [.c., 0, 1, 0, 1, 8] -> 12; [.c., 0, 0, 1, 0, 15] -> 15; [.c., 0, 1, 0, 0, 15] -> 14; [.c., 0, 1, 0, 0, 15] -> 13; [.c., 0, 1, 0, 0, 15] -> 12; [.c., 1, 0, 0, 0, 15] -> 0; // The abel test vectors can be included in the source code file(.abl). See “Converting an ABEL Simulation Input File to CUPL” on page 5 for further information.	CUPL test vectors cannot be part of the Source file. A separate (.si) file must be created as described on page 5. See “Converting an ABEL Simulation Input File to CUPL” on page 5 for further information.

Converting an ABEL Simulation Input File to CUPL

This section describes the features of ABEL and CUPL Simulation input files and the process of converting an ABEL Simulation input file to a CUPL file. Test vectors must be created for the simulator to function and they specify the expected functional operation of a PLD by defining the outputs as a function of the inputs. Test vectors are also used to do functional testing of a device once it has been programmed, to see if the device functions as expected.

CUPL

There are two tools within Atmel-WinCUPL that can be used to simulate the test vectors.

- WinSim® is a windows-based graphical tool used for creating and editing simulator (.si) input files and for displaying the results of the simulation in the form of a waveform. The CUPL simulator requires that a CUPL source file be successfully compiled prior to running simulation. The CUPL compiler generates an intermediate file (with extension .ABS) that is used by the simulator to run functional simulation.
- CSIM is a device-specific simulator and VSIM is a virtual simulator (virtual device) that is text-based and inherently a DOS process. A test specification source file (filename.si) is the input to CSIM/VSIM. The ATF15xx family of Atmel devices only runs VSIM.

The source file may be created using a standard text editor in non-document mode. The source specification file contains three major parts: header information and title block, ORDER statement and a VECTORS statement.

A .si file must have the same header information as .pld (source) to ensure that the proper files, including current revision level, are being compared against each other. Therefore, first copy .pld to .si and then use a text editor to delete everything in .si, except the header and title block.

ABEL

There are two ways to specify test vectors. The most common method is to place test vectors in the ABEL source file. If the user decides to use this method, the Project Navigator (Atmel-Synario) will detect the presence of test vectors in the source file and create a “dummy” test vector file. This file indicates to the system that the actual test vectors are in the ABEL source file.

The other way to specify test vectors is to create a “real” test vector file by selecting the "New" menu item in the Source menu and then choosing test vectors. Note that test vector files have the .ABV file extension and must have the same name as the top-level module. The user must use the Module and End statements exactly as he does when creating an ABEL source file.

Table 2 shows comparative implementation of describing test vectors for ABEL simulation (.ABV) and CUPL simulation (.SI) for the 4-bit counter.

Table 2. Test Vector Description

Counter.abv	Counter.si

Module unicnt
"Constants
X, C, Z = .X., .C., .Z.;
//**Inputs**
d3..d0 pin;
clk pin;
rst pin;
cnten pin;
ld pin;
u_d pin;
//**Output**
q3..q0 pin istype 'reg';
//Counter output,
//Sets
data = [d3..d0]; //Data Set
count = [q3..q0]; //Counter Set
test_vectors
([clk, rst, cnten, ld, u_d, data] -> count)
[.c., 1, 0, 0, 0, 0] -> 0;
[.c., 0, 0, 1, 0, 8] -> 8;
[.c., 0, 1, 0, 1, 8] -> 9;
[.c., 0, 1, 0, 1, 8] -> 10;
[.c., 0, 1, 0, 1, 8] -> 11;
[.c., 0, 1, 0, 1, 8] -> 12;
[.c., 0, 0, 1, 0, 15] -> 15;
[.c., 0, 1, 0, 0, 15] -> 14;
[.c., 0, 1, 0, 0, 15] -> 13;
[.c., 0, 1, 0, 0, 15] -> 12;
[.c., 1, 0, 0, 0, 15] -> 0;
End
	Name counter;
	PartNo 00 ;
	Date 6/30/02 ;
	Revision 01 ;
	Designer Engineer ;
	Company XYZ ;
	Assembly None ;
	Location San Jose;
	Device virtual ;
	ORDER: clk, rst, cnten, ld, u_d, d3, d2, d1, d0, q3, q2, q1, q0;
	VECTORS:
	c 1000 0000 LLLL
	c 0010 1000 HLLL
	c 0101 1000 HLLH
	c 0101 1000 HLHL
	c 0101 1000 HLHH
	c 0101 1000 HHLL
	c 0010 1111 HHHH
	c 0100 1111 HHHL
	c 0100 1111 HHLH
	c 0100 1111 HHLL
	c 1000 1111 LLLL

Overview of Syntax Differences between ABEL and CUPL

The following section includes various tables that show the syntax differences between the two languages pertaining to extensions, operators and keywords.

Reserved Identifiers (Keywords)

Table 3. Syntax Differences

ABEL Keyword	CUPL Keyword
ASYNCH_RESET	None
CASE	IF (in a CONDITION statement)
DECLARATIONS	None
DEVICE	PARTNO
ELSE	ELSE
END	}
ENDCASE	}
ENDWITH	None
EQUATIONS	None
EXTERNAL	None
FLAG (OBSELETE)	None
FUNCTIONAL_BLOCK	None
FUSES	FUSES
GOTO	PRESENT, NEXT
IF	IF (In a CONDITION statement)
IN	None
INTERFACE	None
ISTYPE	**Note 1**
LIBRARY	None
MACRO	FUNCTION
MODULE	None
NODE	NODE/PINNODE
OPTIONS	None
PIN	PIN
PROPERTY	PROPERTY **(Note 2)**
STATE	PRESENT and $DEFINE
STATE_DIAGRAM	SEQUENCE
STATE_REGISTER	No equivalent but can be achieved with FIELD
SYNC_RESET	None
TEST_VECTORS	Generated .SI file
THEN	NEXT

Table 3. Syntax Differences (Continued)

ABEL Keyword	CUPL Keyword
TITLE	NAME
TRACE	None
TRUTH_TABLE	TABLE
WHEN	No equivalent but can be replaced by CONDITION {}
WITH	None

Notes: 1. Instead of using “ISTYPE” in ABEL, one can use a suitable extension in CUPL. Extensions such as .D (specify input to a D-type flip flop) can be used with any pin name. The compiler will determine whether it is valid. The ISTYPE statement defines attributes (characteristics) of signals (pins and nodes) in ABEL. Please refer to Table 4 for further details on ATTRIBUTES. The user should use signal attributes to remove ambiguities in architecture-independent designs. Even when a device has been specified, using attributes ensures that the design operates consistently if the device is changed later.

2. Property statements are used specifically for the ATF1500A and the ATF15xx family of devices to describe specific feature of the device that can be used by the Device Fitter to generate the appropriate FITTER and Jedec files.
   For Example:
   Atmel-ABEL defines such as: ATMEL property 'DEDICATED_INPUT ON';
   Atmel-CUPL defines such as: Property ATMEL {DEDICATED_INPUT ON};

Table 4. Attributes Table

Signal Attributes	Description
'buffer'	No Inverter in Target Device
'collapse'	Collapse (remove) this signal
'com'	Combinational output
'dc'	Unspecified logic is don't care
'invert'	Inverter in Target Device
'keep'	Do not collapse this signal from equations
'neg'	Unspecified logic is 1
'pos'	Unspecified logic is 0.
'retain'	Do not minimize this output. Preserve redundant product terms
'reg'	Clocked Memory Element
'reg_d'	D Flip-flop Clocked Memory Element
'reg_g'	D Flip-flop Gated Clocked Memory Element
'reg_jk'	JK Flip-flop Clocked Memory Element
'reg_sr'	SR Flip-flop Clocked Memory Element
'reg_t'	T Flip-flop Clocked Memory Element
'xor'	XOR Gate in Target Device

Comments in ABEL and CUPL

ABEL: Comments begin with a double quotation mark (") or double forward slash (//).

CUPL: Comments begin with /* and end with */.

Number Representation in Different Bases

Table 5. Number Representation in Different Bases

Base Name	Base	Symbol	
		ABEL	CUPL
Binary	2	^b	'b'
Octal	8	^o	'o'
Decimal	10	^d	'd'
Hexadecimal	16	^h	'h'

Header Information Keywords

Table 6. Header Information Keywords

ABEL	CUPL	Description
Module	Name	Just a filename.
Title	None	Used to give a title or description for the module. (Optional)
None	Partno	The part number for the particular PLD design.
None	Revision	Begin with 01 when first creating a file and increment each time a file is altered.
None	Date	Change to the current date each time a source file is altered.
None	Designer	Specify the designer's name.
None	Company	Specify the company's name.
None	Assembly	Give the assembly name or number of the PC board.
None	Location	The abbreviation LOC can be used.
None	Device	Used to set the default device type for the compilation.

Logical Operator

Table 7. Logical Operator

ABEL	CUPL	Description
!	!	NOT (ones complement)
&	&	AND
#	#	OR
$	$	XOR (exclusive OR)
! $	! $	XNOR (exclusive NOR)

Arithmetic Operators

CUPL arithmetic operators can only be used inside $REPEAT and $MACRO blocks.

Table 8. Arithmetic Operators

ABEL	CUPL	Description
-	-	Subtraction
+	+	Addition
*	*	Multiplication
/	/	Division
%	%	Modulus
<<	None	Shift left by bits
>>	None	Shift right by bits

Relational Operators

Table 9. Relational Operators

ABEL	CUPL	Description
==	None	Equal
! =	None	Not equal
<	None	Less than
<=	None	Less than or equal
>	None	Greater than
>=	None	Greater than or equal

Assignment Operators

Table 10. Assignment Operators

ABEL	CUPL	Set	Description
=	=	ON(1)	Combinational or detailed assignment
: =	=	ON(1)	Implied registered assignment
? =	None	DC(X)	Combinational or detailed assignment
?:=	None	DC(X)	Implied registered assignment
?:=	None	DC(X)	Implied registered assignment

Operator Priority

Table 11. Operator Priority

ABEL	CUPL	Priority	Description
-	None	1	Negate
!	!	1	NOT
&	&	2	AND
<<	None	2	Shift left
>>	None	2	Shift right
*	*	2	Multiply
/	/	2	Unsigned division
%	%	2	Modulus
+	+	3	Add
-	-	3	Subtract
#	#	3	OR
$	$	3/4	XOR: exclusive OR
!$	None	3	XNOR: exclusive NOR
==	None	4	Equal
!=	None	4	Not equal
<	None	4	Less than
<=	None	4	Less than or equal
>	None	4	Greater than
>=	None	4	Greater than or equal

Dot Extension

The dot extensions valid for pins or specific signals in CUPL as well as ABEL are listed in Table 12.

Table 12. Dot Extension

ABEL	CUPL	Description
.ACLR	None	Asynchronous clear
.ASET	None	Asynchronous set
.CLK	.CK	Clock input to an edge-triggered flip-flop
.CLR	None	Synchronous clear
.COM	None	Combinational feedback normalized to the pin value
.OE	.OE	Output enable
.PIN	None	Pin feedback
.SET	None	Synchronous set
.AP	.AP	Asynchronous preset
.AR	.AR	Asynchronous reset

Table 12. Dot Extension (Continued)

ABEL	CUPL	Description
.CE	.CE	Clock-enable input to a gated-clock flip-flop
.D	.D	Data input to a D-type flip-flop
.J	.J	J input to a JK-type flop-flop
.K	.K	K input to a JK-type flip-flop
.LD	None	Register load input
.LE	None	Latch-enable input to a latch
.LH	.LE	Latch-enable (high) to a latch
.PR	.PR	Register preset
.Q	None	Register feedback
.R	.R	R input to an SR-type flip-flop
.RE	None	Register reset
.S	.S	S input to an SR-type flip-flop
.SP	.SP	Synchronous register preset
.SR	.SR	Synchronous register reset
.T	.T	T input to a T-type (toggle) flip-flop
Note 1	.CKMUX	Clock multiplexer selection
.FB **(Note 2)**	.DFB	D registered feedback path selection
.Q **(Note 2)**	.DQ	Q output of D-type flip-flop
None	.INT	Internal feedback path for registered macro cell
None	.IO	Pin feedback path selection
None	.IOCK	Clock for pin feedback register
None	.IOD	Pin feedback path through D register
None	.IOL	Pin feedback path through latch
None	.L	D input of transparent latch
None	.LEMUX	Latch enable multiplexer selection
None	.LFB	Latched feedback path selection
None	.LQ	Q output of transparent input latch
None	.OEMUX	Tri-state multiplexer selection
None	.TFB	T registered feedback path selection

Notes: 1. The .CKMUX dot extension used in CUPL is specific to the Atmel ATV750B and ATF750C devices. The .CKMUX extension is used to connect the clock input of a register to the Synchronous clock pin. This is needed because some devices have a multiplexer for connecting the clock to one set of pins.

2. .DFB and .DQ on CUPL are only used for D-type flip-flop. However, .FB and .Q in ABEL can be used for any type of flip-flops such as D, T, JK, SR flip-flops.

Extensions Applicable for Atmel EPLD Devices

Table 13 lists specific extensions valid for Atmel EPLD devices.

Table 13. Valid Atmel EPLD Device Extensions

Atmel PLDs	Valid Extensions
ATF16V8B/BQ/BQL	OE, D
ATF16V8C/CZ	
ATF20V8B/BQ/BQL	
ATF20V8C/CQ/CQZ	
ATF22V10C/CQ/CQZ	OE, D, AR, SP
ATF22LV10C/CZ/CQZ	
ATV750/L	D, AR, CK, OE, SP, DFB, IO
ATV750B/BL	D, T, AR, CK, CKMUZ, OE, SP, DFB, IO
ATF750C/CL/LVC/LVCL	
ATF1500A/AL/ABV	D, AR, CK, CE, OE, AP, IO, T, L, LE
ATV2500B/BL/BQ/BQL	D, T, AR, CK, OE, SP, IO, CE
ATF2500C/CQ/CQL	
ATF1502AS/ASL/ASV/ASVL	D, T, S, R, OE, OEMUX, CK, CKMUX, AR, DQ, LQ, IO
ATF1504AS/ASL/ASV/ASVL	
ATF1508AS/ASL/ASV/ASVL	

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

3303A–PLD–08/02 xM