Conversion

The Basic Process

At its most simple level, the process of going from an FPGA or PLD design into a lower cost alternative device can be broken down into three steps (Figure 1). The first step is to convert the netlist from the FPGA or PLD form into the new technology. Step two is to take some action to verify that this conversion was done correctly, and to ensure ultimate testability of the final device. Step three is to actually manufacture the new device. Two of these steps are fairly similar regardless of vendor, and one varies significantly. The netlist conversion task typically is a fairly automated procedure, usually consisting of some kind of logic synthesis program to convert the netlist. The back end, building the parts is also similar, consisting of making a mask, metalizing wafers, and assembling parts. The verification step, however, tends to differ.

Design Verification— Alternative Approaches

There are three approaches to the design verification task that are commonly in use. The most typical is the traditional ASIC design flow. Using this method, the customer is responsible for generating vectors, running simulations, and signing-off these simulations. This may involve expending significant effort on tasks with which the engineer may not be familiar. It also means that the customer is responsible for ensuring the correctness of the converted design. The customer assumes any risk in the event of failures at the system level.

A second approach simplifies the task for both the designer as well as the silicon vendor: this is to assume that the conversion was done correctly, and just build the parts with no prior verification. Production testability is accomplished by adding internal scan paths. This approach simplifies the conversion process, but it significantly increases the risk that the parts will not work correctly in the system the first time, as the logic synthesis programs tend to introduce errors in some 25% to 35% of the cases. While the vendor may commit to fix any errors, having to re-spin a design will impact the production schedule significantly.

MHS Verify-Before-Silicon

The third approach is MHS' Verify-Before-Silicon technique. In this approach, MHS undertakes the task of verifying that the conversion has been done correctly. This is done by comparing a simulation of the ULC conversion to the actual FPGA or PLD prior to making a mask and building the ULC, and making sure that they match.

There are two options for vector generation for design verification on a ULC design: 1) the customer can supply the vectors, or, (2) vectors may be generated by MHS. If MHS generates the vectors there may be a substantial impact on conversion. MHS will primarily use Automatic Test Vector Generation software or ATVG. ATVG can successfully cover a substantial portion of many circuits but not in all cases. Design features such as gated clocks, complex state machines, and encryption logic can cause blockage for the ATVG. When this occurs, MHS may: 1) accept the level of coverage and add internal scan paths for acceptable production testing; 2) assign an engineer to analyze the circuit to generate vectors manually; or 3) not accept the circuit for conversion without customer-provided vectors or modifications to improve testability.

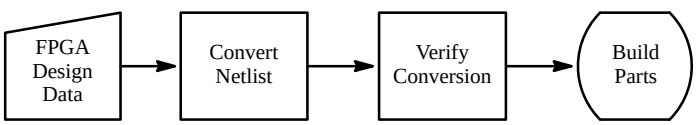

Figure 1. Simplified Conversion Process

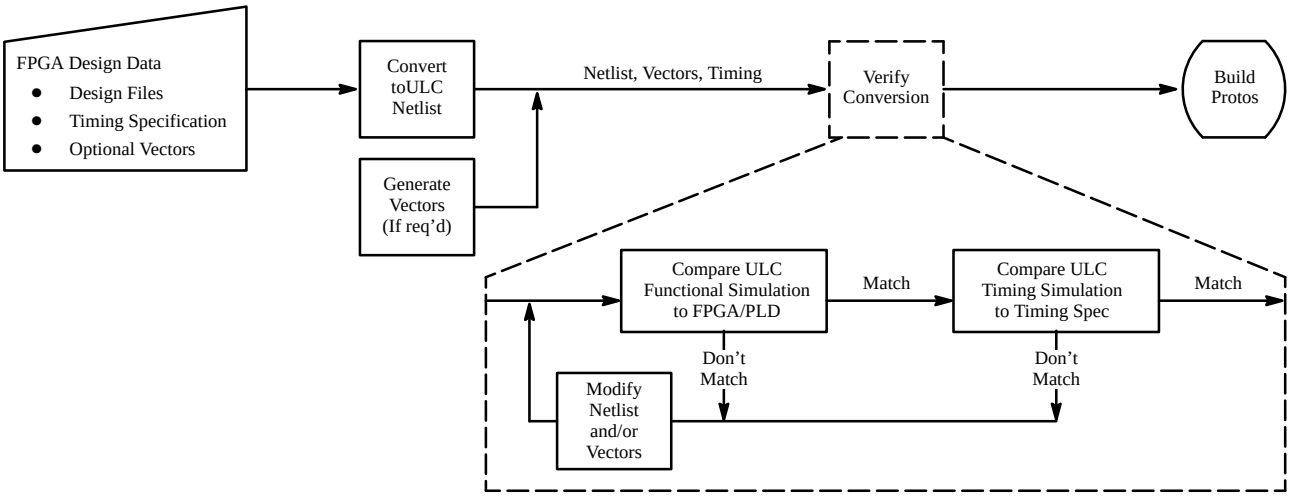

Figure 2. MHS Verify-Before-Silicon Process

Design Considerations

When converting from an FPGA or PLD to a ULC, there are a number of issues of which the designer must be aware. Many of these relate to differences in the performance of the technologies. Some of these have potential impact on the design, others merely affect the documentation which must be provided.

Timing

When converting an FPGA and also older (slow) PLD and CPLD devices, the ULC generally will be faster than the original device. Design shortcomings, which were masked by the lower performance of the programmable device, may become problems in the converted ULC. MHS' Verify-Before-Silicon methodology will discover some of these, but not necessarily all of them. The designer is wise to avoid these situations wherever possible, since such performance-related problems can potentially be harmful even if no conversion is ever made. The majority of devices on the market are not specified with minimum timing specs. In the future, any given speed FPGA or PLD is likely to run faster because of continued improvements in the technologies.

For the timing specification of a PLD or CPLD, the pin-to-pin timing specification of the original device is sometimes an adequate timing specification for the ULC. For an FPGA, this is seldom the case, as the pin-to-pin timing is much more dependent on the placement and routing of the design. Thus, to the extent that the design involves critical timing requirements in this type of design, it is more important that a timing specification be provided by the circuit designer.

Programmed Delays

It is possible to program delays into an FPGA or PLD by specific routing or stacking of logic. While it is possible to achieve a working circuit with this technique, there is no guarantee that it will continue working over time as noted above. When converting to a ULC, it is extremely unlikely that the circuit will function as expected. The best solution is to redesign the circuit prior to submission. If this is not practical, then it may be possible to achieve the desired delay within the ULC. This can only be achieved, however, if the required timing is documented along with the ULC design checklist.

Deglitching Circuits

If any deglitching circuits have been included, it is important to document them and what was their objective. It is completely reasonable to have the ULC duplicate the operation of the FPGA or PLD in these circuits, if properly documented. Without documentation two dangers exists: 1) the logic synthesis process may reduce or eliminate the logic, and, 2) even if the logic is matched the ULC will not have the same logic or routing delays, and so will not function as expected. However, if the objective is documented, any such problems can be corrected.

Design Considerations (Cont'd)

Gated Clocks

Clock signals which pass through logic which toggles them on or off are known as gated clocks. This type of circuit is may generate glitches on the clock line which could cause unexpected operation. Small glitches on a clock line that are too short to have any effect in the FPGA, may cause toggling in the ULC. While this type of problem will typically be discovered in verification, fixing it may delay the conversion. Gated clocks also substantially reduce the effectiveness of ATVG. If it is not possible to avoid gated clocks, they should be documented, and so designed that the inputs are stable and avoid the possibility of a glitch.

External Timing

This is the most critical performance-related issue when converting to a ULC. Changes in device timing can potentially cause violations of setup or hold times to other devices. They can also lead to race conditions at the system level. While such issues internal to the device will generally be discovered and corrected by MHS' Verify-Before-Silicon methodology, there is no way for MHS to know about them at the system level. Therefore, it is very important that any critical timing requirements that the ULC must meet be specified by the designer.

Redundant or Fault-Tolerant Circuits

If any redundant or fault-tolerant circuits have been included, it is important to document them. It is very likely that the logic synthesis process will recognize and remove them unless optimization is disabled around them.

Combinational Latches

If any latches have been built using equations in PLD architecture devices, it is important to document them. Such latches are typically built because the designer needed more latches than were available in the original PLD, but did not want to use a larger part with more resources. These will be manually replaced during the conversion with a normal latch which has more robust operation and greater repeatability for testing.

Internal Tri-State Busses

Internal tri-state busses should be designed such that they are not allowed to float, i.e., they should have pull-ups or pull-downs. If they have not been designed this way, please document them and indicate whether there are any circumstances under which they would be floating in actual operation. MHS will add pull-ups (or pull-downs if preferred) to the ULC.

Power-On Reset

It is generally good practice to avoid relying on a device's internal power-on reset (POR). Typically there will be alternative approaches to reset, such as a system reset on the board. If POR is required for correct operation of the ULC in the board, please indicate this on the design documentation supplied with the checklist.

Internal Scan Path Testing

To improve the test coverage of circuits with low testability through functional vectors, MHS will frequently add internal scan paths. With this technique, normal flip-flops are replaced with special scan flip-flops which have a transparent scan test mode that allows data to be loaded directly into these (typically inaccessible) nodes. A "scan chain" is built incorporating the scan path flip-flops and latches. This then permits data to be serially loaded and read from them in the scan test mode. A compact production test program can thus be created to test a very high percentage of the chip area. This program can also be written without substantial knowledge of the chip's function, making the development much faster. The scan chain is generally accessed though unused pins, hidden dual-mode pins, or through the JTAG boundary-scan Test Access Port. (See Boundary-Scan Support.)

Boundary Scan Support

The Joint Test Action Group (JTAG) has defined a standard protocol, IEEE STD. 1149.1-1990 (IEEE Standard Test Access Port and Boundary-Scan Architecture), for integrated circuits. This makes it possible to test the interconnection on a circuit board as well as of the device itself. Many of the newer FPGA and CPLD devices now include support for this protocol, which is commonly referred to as JTAG Boundary-Scan Testing (BST) or just JTAG. JTAG support can be included in a ULC conversion, but it must be specifically requested. In addition, while the interface and architecture for JTAG are well defined, there are options which can be taken during the conversion of a device which affect compatibility and cost.

The JTAG standard defines a test access port (TAP) that consists of five pins, one of which is optional. The five pins are test clock (TCK), test mode select (TMS), test data in (TDI), test data out (TDO), and the optional test reset (TRS). Commands and data are serially loaded into and read from these pins to control operation and testing. Commands are in the form of an opcode, generally two or three-bits which allows either four or eight instructions. Three instructions are mandatory: 1) EXTEST, drives external I/O through the boundary scan registers, 2) BYPASS, shifts internal data out to adjacent devices, bypassing a device's boundary scan registers, and, 3) SAMPLE/PRELOAD, loads data into or out of the registers. The actual BST registers are a (set of) register(s) at each I/O pin for capturing and loading data, as well as enable and direction control. The actual numbers of registers varies from FPGA to FPGA and often by pin type. For a more complete description of the JTAG implementation of a specific FPGA or CPLD, please read the appropriate device data sheet from the original supplier.

There are three options for the level of JTAG support in a ULC conversion of a device which supports JTAG: no support, which is the default; MHS standard JTAG; and full emulation of the original JTAG.

Full OEM JTAG Support

In converting to a ULC, it is possible to maximize test-program compatibility by fully implementing the scan register arrangement of the original device. In this approach, all of the registers are implemented in each chain just as in the FPGA or CPLD. This allows a test program to be used for either the ULC or the programmable device. However, only the three mandatory instructions would be supported. The disadvantage of this approach is that it uses significant silicon resources which frequently perform no function which can increase the cost of the ULC.

Standard ULC JTAG Support

In an FPGA or CPLD, most of the pins are I/Os which require multiple registers to support BST. Once the device is configured, however, many of these pins become unidirectional, which would only require a single register. Other pins are not even used in the actual design, or may not even exist in the selected package, yet would require all of the registers for full support. It is generally more cost-effective to reduce the scan-chain to only what is needed in the specific design. MHS has defined the following standard for this.

With MHS JTAG only the necessary signal registers are included in the scan chain:

- for inputs only one bit (in)
- for outputs only one bit (out)
- for 3-state outputs two bits (out and enable)
- for bi-directionals three bits (in, out and enable)

In addition, there is only one enable bit in the scan chain for an entire bus (several bi-directionals and 3-state outputs all using the same enable signal).

The TAP specific pins (TCK, TMS, TDI and TDO) are not included in the JTAG chain, whereas they may be in the FPGA. The clock signals are buffered, taking into account the value of the JTAG register for these pins. In the FPGA, the global clock pins may be buffered from the pad not taking this value into account.

The instruction register can have either two bits (three or four instructions) or three bits (five or more instructions). The three mandatory instructions, EXTEST, SAMPLE/PRELOAD, and BYPASS are supported. No support can be provided for any instructions which are specific to the FPGA or CPLD architecture such as Readback or Configure. The internal scan chain may be designed for access using one of the user instructions (INTEST).

An Identification register can be provided which identifies the part and the supplier (MHS).

Actel FPGA Conversion

FGPA Description

Actel devices come in four families for which ULC conversions are supported: ACT1 (A1010, A1020), ACT 2 (A1225, A1240 and A1280), ACT3 (A1415-A14100), and the 1200XL family. These devices have an FPGA architecture with a channeled gate-array-like structure for routing. They use an Antifuse technology for programmable interconnect. The three families differ in size, routability, and performance, but they can be treated similarly from the standpoint of conversion to a ULC. Actel devices are not reprogrammable.

The basic element in an Actel FPGA is called a Logic Module. ACT 1 logic modules are an 8-input, one- output circuit made of three 2–1 multiplexers and one 2-input OR gate. ACT 2 and ACT 3 devices add inputs to the module, and separate modules into two parts: a simpler C-Module, which can only implement combinational logic and the S-Module, which can implement sequential logic as well. ACT 1 and ACT 2 I/O pins may be inputs, outputs, tri-state outputs or bi-directional I/Os. ACT 3 devices add registers to the I/O cells along with additional clock lines.

Input Files

Input files required are the .ADL file and the .PIN file produced by the Actel design system.

Dedicated Pins

Two pins, MODE and V_{PP} are dedicated for probing and programming, and not supported by the ULC.

Issues for Conversion

Resetability and timing are the two main issues the FPGA designer needs to consider in preparing an Actel design for conversion to ULC.

Resetability

Unlike the majority of other FPGA and PLD devices, the Actel architecture does not include a built-in master reset for registers and latches. This is no doubt due to the fact that the Actel Logic Module is a more generic structure than the basic elements of many other architectures, and does not include specific registers. However, this presents a special problem for the ULC conversion. The Verify-Before-Silicon methodology requires placing the actual programmed Actel FGPA on a tester and testing it. This action requires that the FPGA design have some method for resetting any registers, flip-flops and latches. An Actel design which does not meet this requirement cannot be converted to a ULC.

To avoid this problem, all sequential elements of the circuit can be designed explicitly as elements with reset and have their reset lines tied to an I/O which becomes a user's master reset. This technique may have the undesirable effect of using more Actel resources than are readily available, however. Alternatively, sequential elements can be designed so that they are all loadable (or resetable) with an *explicit* input sequence (clock for an unknown number of clocks until a desired state is achieved is *not* acceptable). However, it may not be practical to reach all sequential elements with this technique. A combination of these approaches may also be possible.

Timing

The FPGA architecture of Actel devices means that the input to output timing cannot be easily determined primarily from the data sheet. Rather, it is largely a function of the layout and routing of the specific design. (The exception to this is the clock-to-output timing of the registered I/Os in an ACT 3 device.) It is therefore important for the FPGA designer to understand the system's timing requirements and to document any critical timing requirements in the ULC design input package. This should be system-level timing on a pin-to-pin basis rather than the internal timing of the Actel. It could include propagation delays for combinatorial paths, clock to output delays for sequential paths, as well as set-up and hold times for input and output pins.

It is important to be aware that asynchronous path timing in the ULC device is likely to be significantly faster than the timing of the FPGA. This could potentially lead to race conditions or set-up and hold- time violations at the system level. These would not be discovered by the ULC design verification process. Additional delays can frequently be added to the ULC to meet system level minimum timing constraints, but only where the requirement is documented by the FPGA designer. The FPGA designer typically does not need to be concerned about timing violations internal to the ULC, as these will be discovered by MHS during the conversion.

Altera EPLD Conversion

Altera offers both Complex PLD (CPLD) architecture devices and FPGA architecture devices for which ULC conversions are supported. Altera FPGA conversion is discussed in the next section.

EPLD Description

Altera EPLDs (*Erasable* PLD) come in three families: the "Classic" EP series, the "MAX" EPM5000 series, and the "MAX" EPM7000 series. These families implement logic using a common PLD sum-of-product structure. The basic element of the EPLDs is a logic block called a macrocell. Each macrocell contain a programmable AND array called "product terms," followed by an OR gate to complete the sum, an XOR gate for product term inversion, and a programmable register.

In the Classic family, product terms come directly from input pins or feedback from other macrocells (in devices with more than one macrocell). In the MAX families, macrocells are grouped together in sets of four (MAX 5000) or eight (MAX7000) to form Logic Array Blocks or LABs. The MAX7000 architecture adds a product term select matrix to the macrocell, allowing additional flexibility in the use of the product terms. Inputs to the LABs come from the PIA or from other shared product terms within the LAB. The PIA is a programmable matrix which routes signals from the input and I/O pins, as well as the feedback terms, from the macrocells to the LABs. Classic and MAX5000 devices are programmed via on-board EPROM, and windowed versions can be re-programmed. MAX7000 devices are programmed via on-board EEPROM, and are reprogrammable.

Input Files

The .EDO EDIT output from the MAX Plus compiler is required. All other design files, including schematics, simulation, and equation files are requested.

Dedicated Pins

There are no dedicated programming pins on the Altera EPLD devices. Global clock and Output Enable pins are treated as normal inputs for the purpose of conversion.

Issues for Conversion

There primary issue for consideration in conversion of the EPLD devices is timing.

Timing

The PLD-based architecture of the Altera EPLD devices makes the considerations for timing of the converted device much simpler than in FPGA devices. The data sheet parameters for timing are a reasonable guideline for the maximum timing parameters. However, minimums are generally not given, and a faster part will still meet the specification. This is of special concern when converting from the older, slower technologies, where the application may be predicated on a specific delay. In such cases, it is recommended that the system be tested before attempting conversion with a faster programmable device to ensure compatibility. If specific minimum delays are required, they should be noted on a separate, customer supplied specification.

At the opposite end of the spectrum in the EPLDs, the newer technologies, such as the MAX7000 series, offer versions with very fast global clock to output timing. If the application is using a faster version of one of these devices for T_{PD}, but does not require the global T_{CO} specification, it should be noted. This will impact the required ULC technology, and in some cases affect the feasibility of conversion.

Altera FPGA Conversion

FGPA Description

The “FLEX” EPF8000 series has a segmented FPGA architecture with a channeled gate-array-like structure for routing. Segments are called Logic Array Blocks (LABs) which have global interconnect lines running in both the rows and columns between them. Interconnect programming is provided by a large matrix of switches which are configured by the contents of an onboard static RAM. The SRAM is volatile, which means that a FLEX8000 series FPGA must be re-configured each time following power-up. A FLEX8000 series FPGA may be re-configured in-system at any time. But if the re-configurability is utilized, then the design typically cannot be converted into a ULC.

The basic element in a FLEX8000 FPGA is called a Logic Element or LE. Each LE contains a four-input combinatorial function block or Look Up Table (LUT), an AND/OR cascade block, a register, and one output. The LUT can be configured to any combination of the four inputs as well as several special arithmetic and counter modes with three inputs plus carry-in. There are also preset, clear, and clock inputs, and cascade and carry outputs. LEs are grouped into sets of eight to form the LABs. The LABs also contain local interconnect and control signals. Configuration of the LEs and LABs is also performed by the contents of the onboard static RAM which is loaded at startup.

The FLEX8000 series also supports a JTAG boundary-scan testing mode as a selectable option.

Input Files

The .EDO EDIT output from the MAX Plus compiler is required. All other design files, including schematics, simulation, and equation files are requested.

Dedicated Pins

There are seven dedicated pins on a FLEX 8000 FPGA, plus a number of Dual-Purpose pins which have specific functions during configuration, then become I/O pins for user mode. The ULC will support *none* of these pins unless specifically requested. Special clock pins are treated as normal inputs for ULC conversion. Table 1 summarizes the dedicated and special purpose pins of the FLEX 8000 FPGA.

Table 2: FLEX8000 Dedicated and Dual-Purpose Pins

Pin[a]	I/O	Description
Dedicated Pins (only supported if specifically requested)		
nSP, MSEL1, MSEL0	I	Configuration scheme selection pins; ignored.
nSTATUS	O	Open-drain output; high always, no internal pull-up.
nCONFIG	I	Low resets device.
CONF_DONE	I	Input with pull-up. Must be held low for access to dual-purpose configuration pins.
DCLK	I	Programming clock input. Not supported as output.
Dual-Purpose Pins (only supported if boundary-scan requested)		
TDI, TCK, TMS	I	Boundary-scan pins; Test Data In, Test Clock, Test Mode Select.
TDO	O	Test Data Out for boundary-scan.
Dual-Purpose Pins (only supported if specifically requested) These pins require CONF_DONE to be held low to operate in configuration mode.		
RDYnBUSY	O	High when CONF_DONE is low.
HDC	O	High when CONF_DONE is low.
LDC	O	Low when CONF_DONE is low.
nRE, nWS, CS, nCS	I	Low on nCS and nRS and high on nWS and CS puts RDYnBUSY on DATA7.
DATA7 to DATA0	I(/0)	Internally pulled-up whenever CONF_DONE is low. DATA7 follows RDYnBUSY output as defined above.

Note
a. Pins not listed are not supported in configuration mode.

Altera FPGA Conversion (Cont'd)

Issues for Conversion

For the FLEX8000 FPGA timing, programming emulation and JTAG support must be considered.

Timing

In the segmented FPGA architecture of the FLEX8000 series, Altera claims that the timing characteristics offer greater predictability than other FPGA architectures. While this may be true, the very flexibility of the FPGA architecture means that the timing will not be deterministic. That is to say, the input to output timing cannot be easily determined primarily from the data sheet; it is rather, largely a function of the layout and routing of the specific design. (The exception to this is of course the clock to output timing of the registered I/Os.) It is therefore important for the FPGA designer to understand the system timing requirements and to document any critical timing requirements in the ULC design input package. This should be system-level timing on a pin-to-pin basis rather than the internal timing of the FLEX device. It could include propagation delays for combinatorial paths, clock-to-output delays for sequential paths, as well as set-up and hold times for input and output pins.

It is important to be aware that asynchronous path timing in the ULC device is likely to be quite different than the timing of the FPGA. This could potentially lead to race conditions or set-up and hold time violations at the system level, which would not be discovered by the ULC design verification process. Additional delays can frequently be added to the ULC to meet system level minimum timing constraints; however, this can only be accomplished where the requirement is documented by the FPGA designer. The FPGA designer typically does not need to be concerned about timing violations internal to the ULC as these will be discovered by MHS during the conversion.

Programming Emulation

The are six different modes for programming the FLEX8000 FPGAs, plus corresponding modes for each to program multiple FLEX8000 devices together. Two of the modes are serial, and four are parallel. In three of the modes, the FPGA controls the programming (Active); the other three are controlled by an external source (Passive). The method used by the FPGA designer must be stated in the ULC design input package, along with the required handshaking for correct system operation, (if different from the ULC default). Configuration options selected with the Device Configuration Option Bits must be explicitly requested in the documentation supplied in the ULC design checklist.

The most common mode is programming a single FPGA with a stand-alone serial PROM (Active-serial). In this mode, the FPGA automatically starts clocking and reading from the PROM at power-up or upon release of nCONFIG. In this mode, the converted ULC will operate in the default mode, with CONF_DONE being released right away if required. CONF_DONE does not respond to nCONFIG and is an output only unless specifically requested. The configuration PROM is superfluous to the ULC and may be deleted from the board. Active-Parallel modes, where the FPGA reads its data in parallel from a standard PROM operate similarly.

In the passive modes, the FPGA acts as a peripheral to a microprocessor or the system bus. Some of the user I/Os on the device come up at power-on as inputs on the system bus. In this mode, the system feeds data into the FPGA in parallel or serially until the FPGA receives all its data, at which time it is configured and responds with CONF_DONE, then shifts into normal operation. The ULC will not completely emulate these modes, and it will not support them in its default mode. If the system firmware does not recognize that the ULC starts up programmed, it either must be modified to do so, or the board must be modified to hold CONF_DONE low until all data has been sent, to keep the ULC from entering user mode. RDYnBUSY will be continuously high, and CONF_DONE will be immediately released, unless some other level of support is agreed upon.

Other modes involve connecting more than one FPGA together in parallel or in a parallel bit-slice configuration on a bus for programming. With some limitations, ULCs can support these modes also. If all of the FPGAs in the chain are being replaced, then the ULC in the first position will provide the appropriate emulation as defined above, and the others will operate in the default mode. If not all of the devices in the chain are to be replaced with ULCs, then the configuration data needs to be modified. Multi-device passive parallel modes are subject to the same considerations as single device parallel modes (see Table 2).

Altera FPGA Conversion (Cont'd)

JTAG Support

JTAG support can be provided at three levels: none, partial, and full, as described in the previous section. Keeping in mind that there are potential cost implications of this decision, the appropriate level must be selected.

Table 3: ULC Support of FLEX8000 Configuration Modes

Mode		Support[a]	Comments
Active Serial	AS	Yes	Default Mode. DCLK not generated. PROM(s) may be deleted.
Passive Serial	PS	Yes	Default Mode.
Active Parallel Up	APU	Yes	Default Mode. DCLK and RDCLK not generated. ADD17 to ADD0 and DATA7 to DATA0 immediately in user I/O mode. PROM(s) may be deleted.
Active Parallel Down	APD	Yes	
Passive Parallel Asynchronous	PPA	Yes[b]	Supported in default mode if firmware recognises CONF_DONE without trying to load data. Otherwise, if firmware must load data, CONF_DONE must be held low externally while configuration data is being loaded. RDYnBUSY high when CONF_DONE low.
Passive Parallel Synchronous	PPS	Yes[b]	
Multi-Device Sequential Active Serial	MD-SAS	Limited	If all devices are ULCs, supported in default mode. Otherwise, first device in series cannot be ULC, and configuration data must be modified to eliminate data for ULCs.
Multi-Device Active Serial Bit-Slice	MD-ASB	Limited	
Multi-Device Passive Serial Bit-Slice	MD-PSB	Yes	Default Mode.
Multi-Device Passive Parallel Synchronous	MD-PPS	Yes[b]	If devices are ULCs, supported in default mode if firmware recognizes CONF_DONE without trying to load data. If firmware must load data, CONF_DONE must be held low externally while configuration data is being loaded. RDYnBUSY high when CONF_DONE low. In mixed ULC/FLEX configuration, configuration data must be modified to eliminate data for ULCs.
Multi-Device Passive Parallel Asynchronous	MD-PPA	Yes[b]	
Multi-Device Active Parallel Hybrid	MD-PAH	Limited	If all devices are ULCs, supported in default mode. Otherwise, first device in series cannot be ULC, and configuration data must be modified to eliminate data for ULCs.

Note
a. Support other than default must be specifically requested.
b. Support subject to potential system modification requirements as noted.

AMD/MACH EPLD Conversion

PLD Description

The AMD MACH families are classic examples of the Complex PLD architecture. The simplest description of these devices is several large PAL blocks on a single chip with a switching matrix for interconnect. These PAL blocks implement logic using a sum or products technique. This is formed by a programmable AND array to create the custom product terms driving a fixed OR array to complete the sum. The inputs to the AND array come only from the switching matrix, which is fed by input and I/O pins as well as feedback from macrocells. In the MACH devices the OR array is actually programmably allocable through the Logic Allocator block to the various output (and buried) macrocells, thus giving added flexibility. The macrocells may be programmed to be combinatorial or one of several types of registers or latches. Inputs to the macrocells may also come directly from an input or I/O pin. Outputs may go to a pin (output macrocell only) and/or feedback into the switching matrix. Programming is accomplished by EEPROM, thus making MACH devices reprogrammable.

Input Files

Required input files are the PALASM source file (.PDS) and the JEDEC programming file (.JED).

AMD/MACH EPLD Conversion (Cont'd)

Dedicated Pins

There are generally no pins dedicated for programming on this type of device. There are dedicated clock lines, the number of which depends on the device.

Issues for Conversion

Conversion of MACH devices is generally very straightforward if the correct file types are available.

Timing

The PLD architecture allows that all signals will have the same delay from input to output, unless through a feedback path, which is easily determined. Thus the device data sheet is a valid pin to pin timing specification for the converted part. MACH devices offer versions with very fast clock to output timing. If the application is using a faster version of one of these devices for T_{PD}, but does not require the T_{CO} specification, please so indicate, as this will impACT the required ULC technology, and in some cases affect the feasibility of conversion.

Miscellaneous

MACH devices types support the ability to pre-load registers from the outputs. ULCs will not support this feature. This feature is typically only used during design and debug in any event.

AT&T FPGA Conversion

FGPA Description

AT&T devices come in two families for which ULC conversions are supported: the ATT3000 series and the ORCA family. ATT3000 series devices are second-sources for the Xilinx XC3000 series; please refer to the Xilinx FPGA section for information about these devices.

The ORCA series has an FPGA architecture with a channeled gate-array-like structure for routing. The ORCA 2C series segments the routing by splitting the device into quadrants with additional routing channels provided for interquad routing. Further, in the largest of the 2C devices, quadrants are broken into subquads, with additional routing resources provided between them. Programmable interconnect is provided by a large matrix of switches which are configured by the contents of an onboard static RAM. The SRAM is volatile, which means that an ORCA FPGA must be re-configured each time following power-up. An ORCA FPGA may be re-configured in-system at any time. But if the re-configurability is utilized, then the design typically cannot be converted into a ULC.

The basic element in an ORCA FPGA is called the Programmable Logic Cell or PLC. Each PLC contains a combinatorial function block or Look Up Table (LUT), four registers/latches, and internal interconnect and routing muxes. There are 14 data inputs, one for carry and four for control. There are five outputs and one carry output. The LUT has six inputs and can be subdivided into two blocks with five inputs or four blocks with four inputs. The LUT can also be used in a ripple mode or as two blocks of memory 16 bits x 2 bits. Configuration of the PLC is performed by the contents of the onboard static RAM which is loaded at start-up.

The ORCA series also supports a JTAG boundary-scan mode as a selectable option.

Input Files

The EDIF netlist output from Viewlogic and the .BIT file output from the NeoCAD FPGA Foundry for ORCA.

Dedicated and Special Purpose Pins

There are seven dedicated pins on an ORCA FPGA plus a number of special-purpose pins which have specific functions during configuration, then become User I/O pins for normal operation. The ULC will support *none* of these pins unless specifically requested. Special clock pins are treated as normal inputs for ULC conversion. Table 4 summarizes the dedicated and special purpose pins of the ORCA FPGAs

AT&T FPGA Conversion (Cont'd)

Table 4: ORCA Dedicated and Special Purpose Pins

Pin[a]	I/O	Description
Dedicated Pins (only supported if specifically requested)		
RESET	I	Resets device when low.
CCLK	I	Programming clock input. Not supported as output.
DONE	I	Input with pull-up. Must be held low for access to special purpose configuration pins.
PRGM	I	Resets device when low.
RD CFGN	I	Tri-states all I/O. Only implemented if boundary-scan support requested. Readback function not supported.
RD_DATA/TDO	O	Test Data Out for boundary-scan (if requested). Readback function not supported.
Special Purpose Pins (only supported if boundary-scan requested)		
TDI, TCK, TMS	I	Boundary-scan pins Test Data In, Test Clock, Test Mode Select. Internal pull-up enabled during configuration.
Special Purpose Pins (only supported if specifically requested) These pins require DONE to be held low to operate in configuration mode.		
RDY/BUSY	O	High when DONE is low
DIN	I	For slave serial mode, data on DIN appears on DOUT after T_{PD}.
DOUT	O	For slave serial mode, data on DIN appears on DOUT after T_{PD}. DOUT not supported for other modes.
M0,M1,M2	I	Configuration mode pins. Internal pull-up ignored.
INIT	O	Open-drain output with pull-up. High when DONE is low.
HDC	O	High when DONE is low.
LDC	O	Low when DONE is low.
CS0, CS1, RD, WR	I	In peripheral modes, low on CSO andRD and high on WR and CS1 puts RDY/BUSY on D7.
D(7:0)	I(/O)	Internally pull-up when DONE is low. D7 follows RDY/BUSY as defined above.

Note
a. Pins not listed are not supported in configuration mode.

Issues for Conversion

Timing, programming emulation, and RAM are the three main issues the FPGA designer needs to consider in preparing an ORCA FPGA design for conversion to ULC.

Timing

The FPGA architecture of the ORCA devices means that the basic timing for a design is not deterministic. That is to say, the input to output timing cannot be easily determined primarily from the data sheet; rather, it is largely a function of the layout and routing of the specific design. It is therefore important for the FPGA designer to understand his system timing requirements and to document any critical timing requirements in the ULC design input package. This should be system-level timing on a pin-to-pin basis rather than the internal timing of the ORCA device. It could include propagation delays for combinatorial paths, clock-to-output delays for sequential paths, as well as set-up and hold times for input and output pins.

Support other than default must be specifically requested.

Note: Support is subject to potential system modification requirements as noted.

It is important to be aware that asynchronous path timing in the ULC device is likely to be somewhat to significantly faster than the timing of the FPGA. This could potentially lead to race conditions or set-up and hold time violations at the system level, which would not be discovered by the ULC design verification process. Additional delays can frequently be added to the ULC to meet system level minimum timing constraints; however, this can only be accomplished where the requirement is documented by the FPGA designer. The FPGA designer typically does not need to be concerned about timing violations internal to the ULC as these will be discovered by MHS during the conversion.

AT&T FPGA Conversion (Cont'd)

Programming Emulation

The are a number of different modes for programming the ORCA FPGAs. The method used by the FPGA designer must be stated in the ULC design input package, along with the required handshaking for correct system operation, if different from the ULC default. It is also possible to read-back the loaded contents of the FPGA in some modes; however, the ULC will not support this feature, as there is no loaded program to read back.

The most common modes involve programming a single FPGA with a stand-alone serial or parallel PROM. In this modes, the FPGA automatically starts clocking and reading from the PROM at power-up or upon release of RESET. For these modes, the converted ULC will operate in the default mode, with DONE being available right away, if required. DONE does not respond to RESET unless specifically requested. The configuration PROM is superfluous to the ULC and may be deleted from the board.

Two other modes have the FPGA acting as a peripheral to a microprocessor or the system bus. Some of the user I/Os on the device come up at power-on as inputs on the system bus. The system feeds data in parallel into the FPGA until it releases Program, at which time the FPGA is configured and responds with DONE, then shifts into normal operation. For emulation of this modes, the ULC will respond appropriately, if DONE is held low, and the programming data will actually be ignored. Otherwise DONE, if required, will be true immediately and the device will be in normal mode.

Other modes involve connecting more than one ORCA FPGA together in a daisy-chain for programming. The first device in the chain is the master, and operates in one of the previously described modes, then shifts data out to the remaining devices in the chain, which are slaves. With some limitations, ULCs can support these modes also. If all of the FPGAs in the chain are being replaced, then the ULC in the master position will provide the appropriate emulation as defined above, and the slaves will operate in the default mode. If not all of the devices in the chain are to be replaced with ULCs, the master position cannot be a ULC. Any ULCs in the chain will pass configuration data directly on to the subsequent device after tPD, however the ULC will not recognize or keep its own configuration data. Therefore if a ULC is to be followed by an FPGA in a daisy chain, the configuration data will need to be altered. (See Table 5.)

RAM

ULCs do support ORCA LUT RAM blocks. But as large amounts of RAM can significantly affect the size of the ULC die required, it is important to indicate that RAM is being used, and to provide the total number of bits required.

JTAG Support

JTAG support can be provided at three levels: none, partial, and full as described in the previous section. Keeping in mind that there are potential cost implications of this decision, the appropriate level must be selected.

Table 5: ULC Support of ORCA Configuration Modes

Mode	Support[a]	Comments
Master Serial	Yes	Default Mode. CCLK not generated. PROM(s) may be deleted.
Slave Serial	Yes	Data on DIN passed directly to DOUT after T_{PD}. Device does not absorb its own bits.
Master Parallel Up	Yes	Default Mode. CCLK not generated. A(17:0) and D(7:0) immediately at non-config I/O modes. PROM(s) may be deleted.
Master Parallel Down	Yes	
Asynchronous Peripheral	Yes[b]	Supported in default mode if firmware recognizes DONE without trying to load data. Otherwise, if firmware must load data, DONE must be held low externally while configuration data is being loaded. RDY/$\overline{\text{BUSY}}$ is high continuously when DONE is low. CCLK not generated and DOUT not supported for daisy-chain.
Synchronous Peripheral	Yes[b]	
Slave Parallel	Yes[b]	
RAM_W (Scan)	TBD	
Daisy Chained	Limited	If all devices are ULCs, supported in default mode. Otherwise, master cannot be ULC, and configuration data must be modified to eliminate data for ULCs.

Note
a. Support other than default must be specifically requested.
b. Support subject to potential system modification requirements as noted.

Lattice pLSI EPLD Conversion

PLD Description

The Lattice PLSI families are examples of the Complex PLD architecture. There are three families, pLSI1000, pLSI2000, and pLSI3000 with slight architectural differences: ispLSI devices have the same architecture, but are programmable in-system. The simplest description of these devices is many small PLD blocks called Generic Logic Blocks (GLB) on a single chip with a switching matrix called the Global Routing Pool (GRP) for interconnect. Inputs to the GRP come from the I/O pins and from GLB feedback.

The GLBs implement logic using a sum or products technique which is a programmable AND array to create the custom product terms driving Product Term Sharing Array (Programmable OR array) to complete the sum, which in turn feeds a group of four configurable registers. The registers may be programmed to be one of several types of registers or latches and may be bypassed for a straight combinatorial path. The GLB's inputs to the AND array come from the GRP plus two dedicated inputs which are shared with other GLBs within a megablock.

A megablock in the pLSI1000 series consists of the 8 GLBs, 16 I/O cells, and a switching matrix called the Output Routing Pool (ORP) by which the GLB outputs are feed to the I/Os. In the pLSI2000 series, each megablock has two ORPs and 32 I/O cells.

The pLSI3000 series groups two GLBs together with a common AND array, into a "Twin-GLB." Twin GLBs have no dedicated inputs. The pLSI3000 megablock groups four Twin-GLBs together with one ORP and 16 I/O cells. pLSI3000 also has a different global clocking arrangement and support for JTAG Boundary-scan.

Programming is accomplished by EEPROM, thus making these devices reprogrammable.

Input Files

Required input files are a Verilog netlist output from the Lattice design system, or if the design was originally created using Viewlogic rather than Lattice, a pre-layout EDIF netlist from Viewlogic is acceptable.

Dedicated Pins

There is one dedicated pin for ispLSI programming, and several others which have dual functionality. None of these programming functions are supported by the ULC; only the normal mode functions of the dual function pins are supported. In the 3000 series, these dual-purpose pins perform the boundary-scan functions in normal mode; these would be supported if boundary-scan support is requested in the ULC. There are dedicated clock lines, the number of which depends on the device.

Issues for Conversion

Conversion of pLSI is relatively straightforward, provided the correct input file types are available.

Timing

The modified PLD architecture allows that all signals will have the approximately same delay from input to output, unless through a feedback path, which is easily determined. The only variability is that the delay is effected by GLB loading. But this effect is fairly small, unless there are large differences in loading between product terms. Thus the device data sheet is a valid pin-to-pin timing specification for the converted part.

JTAG Support

For the pLSI3000 series, JTAG support can be provided at three levels: none, partial, and full as described in the previous section. Keeping in mind that there are potential cost implications of this decision, the appropriate level must be selected.

PALs and GALs Conversion

PLD Description

PALs and GALs are the most basic form of the programmable logic device or PLD. These devices typically implement logic using a sum or products technique. This will usually mean a programmable AND array to create the custom product terms driving a fix OR array to complete the sum. The OR array is optionally followed by a series of (programmable) registers, which may have feedback into the array, and which drive (typically) tri-state outputs or I/Os. Programming is accomplished by fuses, EPROM or EEPROM.

Input Files

Required input files are: PALASM source file (.PDS) or *OPEN* ABLE source file (.ABL) plus the JEDEC programming file (.JED).

Dedicated Pins

There are generally no pins dedicated for programming on this type of device.

Issues for Conversion

Conversion of PALs and GALs is generally very straightforward.

Timing

The PLD architecture allows that all signals will have the same delay from input to output, unless through a feedback path, which is easily determined. Thus the device data sheet is a valid pin-to-pin timing specification for the converted part. However, if converting from an older, slower device, it is very important to test the application with a device which will more closely match the ULC speed. This will help determine whether the faster device will not cause system problems.

Miscellaneous

Many PAL and GAL device types support the ability to pre-load registers from the outputs. ULCs will not support this feature. This feature is typically used only during design and debug.

Xilinx FPGA Conversion

FGPA Description

Xilinx devices come in three main families for which ULC conversions are supported: XC2000, XC3000, and XC4000. Other, derivative families, such as the XC3000A and XC3100 need not be given separate treatment from the standpoint of conversions. These devices have an FPGA architecture with a channeled gate-array-like structure for routing. Programmable interconnect is provided by a large matrix of switches which are configured by the contents of an onboard static RAM. The SRAM is volatile, which means that a Xilinx FPGA must be reconfigured each time following power-up. A Xilinx FPGA may be reconfigured in-system at any time, however, if the reconfigurability is utilized, then the design typically cannot be converted into a ULC. The three families offer differences in size, routability, and performance for the FPGA designs; however, they can be treated similarly from the standpoint of conversion to a ULC. The XC4000 series offers the additional capability of providing user accessible RAM blocks, which can also be readily incorporated in the ULC conversion.

The basic element in a Xilinx FPGA is called a Configuration Logic Block or CLB. The CLBs vary in complexity between the families. Each CLB contains a combinatorial function block, two flip-flops, and multiplexers to direct signals within the CLB and to its two to four outputs. The combinatorial section can be configured to any combination of the four or five inputs, including being separated into two distinct blocks of three or four inputs. Additional inputs are provided for control which in some cases may be used as additional function inputs. Configuration of the CLB is performed by the contents of the onboard static RAM which is loaded at startup.

Xilinx XC4000 series devices support JTAG boundary-scan testing as a built-in option.

Xilinx FPGA Conversion (Cont'd)

Input Files

Input files required are the (post-layout) .LCA file, the .MCS (or .BIT) file and a simulatable .XNF file. Designs with hard macros or user-configured CLBs must be flattened. This can typically be accomplished by running the programs XNF2LCA then LCA2XNF from the Xilinx development system. The header of the .XNF file will tell which program generated it.

Dedicated and Special Purpose Pins

There are seven dedicated pins on a Xilinx FPGA plus a number of Special-Purpose pins which have specific functions during configuration, then become User I/O pins for normal operation. The ULC will support *none* of these pins unless specifically requested. Special clock pins are treated as normal inputs for ULC conversion. Table 6 summarizes the dedicated and special purpose pins of the Xilinx FPGAs.

Issues for Conversion

Timing, programming emulation, RAM, and JTAG support are the four main issues the FPGA designer needs to consider in preparing a Xilinx FPGA design for conversion to ULC.

Table 6: Xilinx Dedicated and Special Purpose Pins

Pin[a]	I/O	Description
Dedicated Pins (only supported if specifically requested)		
CCLK	I	Programming clock input. Not supported as output.
DONE (XC4000) D/$\overline{P}$ (XC2/3000)	I	Input with pull-up. Must be held low for access to special purpose configuration pins.
$\overline{RESET}$ (XC2/3000)	I	Resets device when low.
$\overline{PROGRAM}$ (XC4000)	I	Resets device when low.
M0/RTRIG	I	Configuration pin, Readback Trigger. Ignored, Readback not supported. (XC2/3000)
M1/$\overline{RDATA}$	I	Configuration pin, Readback Data. Ignored, Readback not supported.
$\overline{PWRDWN}$	I	Not Supported
User I/O Pins That Can Have Special-Functions (only supported if boundary-scan requested) [XC4000]		
TDI, TCK, TMS	I	Boundary-scan pins; Test Data In, Test Clock, Test Mode Select. Internal pull-up enabled during configuration.
TDO	O	Test Data Out for boundary-scan.
User I/O Pins That Can Have Special-Functions (only supported if specifically requested) These pins require DONE to be held low to operate in configuration mode.		
RDY/$\overline{BUSY}$	O	High when DONE is low
DIN	I	For slave serial mode, data on DIN appears on DOUT after T_{PD}.
DOUT	O	For slave serial mode, data on DIN appears on DOUT after T_{PD}. DOUT not supported for other modes.
M0,M1	I	Configuration mode pins. Ignored. (XC4000)
M2	I	Configuration mode pin. Ignored.
$\overline{INIT}$	O	Open-drain output with pull-up. High when DONE is low.
HDC	O	High when DONE is low.
$\overline{LDC}$	O	Low when DONE is low.
$\overline{CS0}$, CS1, $\overline{RD}$, $\overline{WR}$	I	In peripheral modes, low on $\overline{CSO}$ and $\overline{RD}$ and high on $\overline{WR}$ and CS1 puts RDY/$\overline{BUSY}$ on D7.
D(7:0)	I(/O)	Internally pull-up when DONE is low. D7 follows RDY/$\overline{BUSY}$ as defined above.

Note
a. Pins not listed are not supported in configuration mode.

Xilinx FPGA Conversion (Cont'd)

Timing

The FPGA architecture of Xilinx devices means that the input to output timing cannot be easily determined primarily from the data sheet. Rather, it is largely a function of the layout and routing of the specific design. (The exception to this is of course the clock to output timing of the registered I/Os.) It is therefore important for the FPGA designer to understand the system timing requirements and to document any critical timing requirements in the ULC design input package. This should be system-level timing on a pin-to-pin basis rather than the internal timing of the Xilinx device. It could include propagation delays for combinatorial paths, clock to output delays for sequential paths, as well as set-up and hold times for input and output pins.

It is important to be aware that asynchronous path timing in the ULC device is likely to be somewhat to significantly faster than the timing of the FPGA. This could potentially lead to race conditions or set-up and hold time violations at the system level, which would not be discovered by the ULC design verification process. Additional delays can frequently be added to the ULC to meet system level minimum timing constraints; however, this can only be accomplished where the requirement is documented by the FPGA designer. The FPGA designer typically does not need to be concerned about timing violations internal to the ULC as these will be discovered by MHS during the conversion.

Programming Emulation

Support other than default must be specifically requested.

Note: Support is subject to potential system modification requirements as noted.

There are a number of different modes for programming the Xilinx FPGAs. The method used by the FPGA designer must be stated in the ULC design input package, along with the required handshaking for correct system operation, if different from the ULC default. It is also possible to read-back the loaded contents of the FPGA in some modes; however, the ULC will not support this feature, as there is no loaded program to read back.

The most common modes involve programming a single FPGA with a stand-alone serial or parallel PROM . In this modes, the FPGA automatically starts clocking and reading from the PROM at power-up or upon release of RESET. For these modes, the converted ULC will operate in the default mode, with DONE being available right away, if required. DONE does not respond to RESET unless specifically requested. The configuration PROM is superfluous to the ULC and may be deleted from the board.

Table 7: ULC Support of Xilinx Configuration Modes

Mode	Support[a]	Comments
Master Serial	Yes	Default Mode. CCLK not generated. PROM(s) may be deleted.
Slave Serial	Yes	Data on DIN passed directly to DOUT after T_{PD}. Device does not absorb its own bits.
Master Parallel Up/Down (XC4000)	Yes	Default Mode. DCLK and RDCLK not generated. ADD17 to ADD0 and DATA7 to DATA0 in user mode at start. PROM(s) may be deleted. CCLK not generated and DOUT not supported for daisy-chain.
Peripheral (XC2/3000) Asynchronous Peripheral (XC4000)	Yes[b]	Supported in default mode if firmware recognizes DONE without trying to load data. Otherwise, if firmware must load data, DONE must be held low externally while configuration data is being loaded. RDY/$\overline{\text{BUSY}}$ is high continuously when DONE is low. CCLK not generated and DOUT not supported for daisy-chain.
Synchronous Peripheral (XC4000) Only	Yes[b]	
Daisy-Chained	Limited	If all devices are ULCs, supported in default mode. Otherwise, master cannot be ULC, and configuration data must be modified to eliminate data for ULCs.

Note
a. Support other than default must be specifically requested.
b. Support subject to potential system modification requirements as noted.

Xilinx FPGA Conversion (Cont'd)

Two other modes have the FPGA acting as a peripheral to a microprocessor or the system bus. In these some of the user I/Os on the device come up at power-on as inputs on the system bus. The system then feeds data in parallel into the FPGA until it releases Program, at which time the FPGA is configured and responds with DONE, then shifts into normal operation. For emulation of this modes, the ULC will respond appropriately, if DONE is held low, and the programming data will actually be ignored. Otherwise DONE, if required, will be true immediately and the device will be in normal mode.

Other modes involve connecting more than one Xilinx FPGA together in a daisy-chain for programming. The first device in the chain is the master, and operates in one of the previously described modes, then shifts data out to the remaining devices in the chain, which are slaves. With some limitations, ULCs can support these modes also. If all of the FPGAs in the chain are being replaced, then the ULC in the master position will provide the appropriate emulation as defined above, and the slaves will operate in the default mode. If not all of the devices in the chain are to be replaced with ULCs, the master position cannot be a ULC. Any ULCs in the chain will pass configuration data directly on to the subsequent device after T_{PD}. But the ULC will not recognize or keep its own configuration data. Therefore, if a ULC is to be followed by an FPGA in a daisy chain, the configuration data will need to be altered. Table 7 summaries these modes.

RAM

ULCs do support XC4000 RAM blocks. As large blocks of RAM can significantly affect the size of the ULC die required, it is important to indicate that RAM is being used, and the total number of bits required. All RAM blocks are broken down by the Xilinx software into smaller blocks of 16X1, 16X2 or 32X1, so the higher-level configuration of RAM blocks is not significant for the ULC.

JTAG Support

JTAG support can be provided at three levels: none, partial, and full as described in the previous section. Keeping in mind that there are potential cost implications of this decision, the appropriate level must be selected.

Conversion Information for Other Families

MHS is continuously evaluating new architectures and families for ULC conversions. Please check with your TEMIC representative for the current status of other families.

Test Vectors

Introduction

The traditional ASIC development process requires the generation of several different classes of "test vectors" which exercise a design. Some of these have applicability in FPGA or PLD designs, and some do not. A general definition of vectors is a set of input stimuli and (optionally) the corresponding output results for a given device. *Customer provided vectors of any kind are optional for most ULC designs.*

Timing simulation vectors are often used by designers to verify the performance of sections of a design (critical paths). Functional simulation vectors are used to exercise the functionality of a circuit, without regard to timing performs to verify that the logic is designed correctly and does the desired function. Both of these vector classes would be used in the simulation of a design, and either could have applicability to FPGA or PLD design, if the designer is performing a simulation.

More commonly discussed are **production test vectors** which are used in the development of a production test program. These generally have very strict rules relating to tester operation. They also typically pose requirements relating to the "fault coverage" of the design. Fault coverage is a measure of the likelihoof that flaws in the final production part will be detected by the test program. It also often involves issues with little relation to the priorities of the designer. This class of vectors would seldom have any applicability to FPGA or PLD design, as it relates more specifically to the silicon. MHS has automatic procedures for generating production test vectors and programs.

The conversion of a PLD or FPGA design into a ULC requires a functional vector set which ideally exercises all of the functional blocks of the circuit as they are used by the application, to verify that the conversion was done correctly. MHS has automatic test vector generation (ATVG) tools to assist in the development of these vectors. In some cases, however, manual intervention is required to achieve acceptable coverage of the design. This can substantially increase the time required for conversion.

As previously noted, customer provided vectors are optional for most ULC designs. However, there are certain situations where a customer may wish to provide vectors. Most notably, customer provided functional vectors can improve the design lead time significantly in many cases. They also will reduce the required minimum order size. This is especially true for larger FPGAs and CPLDs, where 50% to 75% of the design time is spent developing and debugging test vectors. Additionally, if the customer wishes to review any critical timing prior to release of the design (which is not required), then the provision of timing simulation vectors for the desired critical path(s) are required. The ATVG generated vectors do not isolate the required path(s) sufficiently for review by the customer.

If a complete functional vector set is not available, but partial simulations have been run, these may still be useful, and they could result in a reduction of the actual conversion time. Of particular value would be a vector set or sequence for initializing the circuit. Equally valuable vector sets or sequences for any parts of the circuit which require a specific sequence of events or combination of inputs to occur for operation. These might include a state machine, encoders or decoders, or an encryption circuit. This information can be sent as a vector set in one of the forms defined below, or as a sequence in the form of a truth table, written description or waveform. All inputs should be included even if they are "X".

Generating Vectors

If vectors are to be submitted with a ULC design, there are several rules which should be followed in their design. These issues are independent of the format in which the vectors are to be provided, which is described in the section “Vector Formats.”

1. The inputs should be periodic, not based on asynchronous system events. 100 ns is the preferred period, although other periods are acceptable. A 50-MHz clock, for example, might require a 20-ns period or shorter if both edges are used.

2. Bidirectional I/Os should be stable for at least one full period.

3. Clocks and input data should not be toggled in the same period.

4. One or two files are preferred, rather than a collection of smaller files. A file containing vectors for inputs, I/Os and I/O control pins only is required; one with all pins is optional.

5. All pins should be specified in binary radix, with one column per pin. Hex busses are not allowed.

6. The vector set should toggle all input pins and cause all outputs to toggle. This includes all I/Os being put in output mode and toggling.

7. The vector set should provide at least 85% fault coverage.

8. The number of vectors is limited to 15,000. If the number of vectors required to achieve sufficient functional coverage is greater that 15,000, please discuss this issue in advance as there may be cost implications.

9. An information file (text) should be provided with the vector files (or included in them). The following information is desired:

   a. which simulator was used to generate the vectors;

   b. which I/O control pins apply to which I/Os and logic;

   c. definition of states used, ie., X = don’t care, U= pull-up (resistive high);

   d. format of the header (if it is not obvious).

Vector Formats

There are a significant number of different simulators and vector formats in use today for the design of PLDs and FPGAs. MHS is supporting two common formats which are easily generated by most of the generic and proprietary EDA tools in general use: JEDEC and a basic truth table format. Specific instructions for Viewlogic users are also included here.

The **JEDEC** format (Figure 1) can be easily generated for the small PALs, Altera EP classic, MACH, and Lattice pLSI devices with the design software available for each. In this format, the vectors are included in the JEDEC file used to program your part and follow the fusemap. The truth table format is supported by most of the simulators used in the design of FPGAs.

Vector Formats (Cont'd)

Figure 1. JEDEC Format Example

```
Created by arctojed VER.      V1.013/20/92
Date                  11/19/1993
Time                  10:4:14
Design Name           P06
.np1 file used created  11/19/1993 10:31
Clock pins(s)         NONE
*
V0001 000000000N0H000XHLLN*
V0002 000000000N0H000XHLLN*
V0003 000000000N0L001XHLLN*
V0004 000000000N0H000XHLLN*
V0005 000000001N0H110HLLLN*
                      .   .   .   .
                      .   .   .   .
V0019 100000000N0H010HHLLN*
V0020 000000000N0H010LHLLN*
V0021 100000000N0H010LHLLN*
V0022 100000000N0H110LHLLN*
C0000*
0000
```

The **truth table** format (Figure 2) should have all the inputs listed one per column and should have an even time stamp, that is, it should be created with the simulator output not in a print-on-change mode. While a vector set with both inputs and outputs is acceptable, a table with inputs only is preferred. Bidirectional pins and their control signals should also be included in the table.

Figure 2. Truth Table Example

```
PPPPPPPPPPPPPPPPC
IIIIIIIIIIIIIIIIN
NNNNNNNNNNNNNNNNT
12345678911111111R
12345678L TIME
    0 1101101101011112Z1
 2000 1101101101011112Z1
 4000 0101101101010111 10
 6000 0010011101011102Z1
 8000 1100111101101002Z1
      .   .   .   .
      .   .   .   .
26000 0100100000010101 00
28000 1000011000110001 00
30000 1101110010111002Z1
Symbols
1 : Forcing High
0 : Forcing Low
Z : Tristate (floating)
* : Indeterminate (X)
L : Resistive Low (pull–down)
H : Resistive High (pull–up)

Bidirectional Pins
Pin17, Pin18 – Enable = CNTR
```

Vector Formats (Cont'd)

In order to insure that vectors generated with **Viewlogic** are in fact correct (ie., generated with the appropriate libraries and other settings), **.CMD** files are not accepted as an input format for ULC vectors. You must generate a truth table format output file from your Viewlogic **.CMD** file. This is done by making the following changes (Figure 3) to your **.CMD** file, then re-running Viewsim which will create an acceptable input file for ULC vectors. Basically you will create a variable (**all_pr**) that contains all your pins and use that variable to create a generic truth table output file.

Figure 3. Viewlogic CMD File

```
|VIEWLOGIC stimuli file for MHS ULC
|
vector all_st +
|List all your inputs—you probably have this already
|
|
vector all_pr +
|List all your PINS both input and output, this is the |variable that will be
used to create the output file.
|The radix should be binary, no HEX busses please.
|
watch +
|This is a copy of the "all_pr" variable above
|
break all_pr ? do (print > filename.ext)
|This line sends the printout to a file "filename.ext"
|(you choose the name), this is the file that will be
|converted and used.
|
|Your vectors follow:
wfm all_st @0     = ZZZZZZZZZZZ
wfm all_st @0     = 11101110000
wfm all_st @4000  = 11111111000
wfm all_st @8000  = 01111111010
wfm all_st @12000 = 11111111000
.
.
.
wfm all_st @8204000 = 10010110100
wfm all_st @8208000 = 00010110110
wfm all_st @8212000 = 10010110100
|
run
|
```

ULC™ Conversion

Design Requirements

A ULC Design Checklist must be submitted with each design package (each code). A copy of the checklist is in Section 7. This section defines these requirements in greater detail.

1. Technical Contact

Frequent communication is not generally necessary for a ULC conversion. However, in the event that a question should arise regarding the design, please include the name and phone number of an appropriate contact.

2. Technical Input Data

Block diagram, schematic, description and pin-out

Since MHS is performing the verification of a ULC conversion, it is generally necessary to develop some understanding of what is happening inside. The documentation requested here is the minimum necessary to meet this requirement:

a) Supply a block diagram that identifies the major circuit blocks in the design and shows the flow of the signals from inputs to outputs. A full schematic is desirable if available, and may take the place of a block diagram.

b) Include a pin-out description. A schematic representation of the device with all pins labeled with signal names as produced by many design packages is helpful. At a minimum, all pins should be defined: input, output, I/O, V_{DD}, GROUND, clock, etc. Any special requirements on any pins should also be identified here, such as pull-ups, pull-downs, CMOS vs. TTL levels, special drive, etc.

c) Provide a functional description of the circuit and how it performs. Identify and give a short description of the function of all major blocks and identify any special features such as a counter that resets itself after *n* counts. This description can typically be handled in a half page for moderate-sized designs.

Miscellaneous

Include any other information that would be useful in converting your design. Specify problem areas encountered during the original FPGA or PLD design. Frequently, this information is useful for the conversion and facilitates MHS' commitment to giving you the best possible product.

3. Part Masters

Part Masters are needed for MHS' "verify-before-silicon" process. For non-volatile technologies, two masters are required. For volatile technologies, a single blank is necessary. Please double-check that the masters match the provided files (both source and programming). Mismatched parts and files are a frequent source of delays in the conversion process.

4. Marking Instructions

ULCs are marked with the ULC part number, MHS custom program number, and have a line available for your custom program number. The desired part number must be provided. Alternatively, a detailed marking specification may be submitted if, for example, you require your company's logo. Non-standard markings may impact the cost.

5. Purchase Order

A purchase order with deliveries contingent on prototype acceptance, which meets the minimums specified in our quotation, is required to begin the conversion.

6. Design Files

The essential design files are listed in Table 1. These may be transmitted on a DOS floppy, or electronically via Internet or BBS.

7. Initialization Information

Initialization/reset sequence

The initialization/reset sequence should be the first part of your functional input vector set if you are providing vectors, and is the input pattern required to bring the entire design to a known state. This includes resetting or loading all memory devices or registers/latches and all counters, resetting all combinational feedback loops, and initializing all state machines. If you are not providing vectors for your design, this description should be *very* detailed, identifying the states and sequences on *all* pins even if they are "don't cares." The most effective format to provide this is a truth table format (see preceding section, Test Vectors, for example).

Resetability

The Actel architecture does not have a built-in master reset or power-on reset function, which is provided for all other PLD and FPGA architectures. It is therefore necessary to explicitly design-in reset capability into an Actel design, either with a specific reset pin or a specific set of input vectors which will perform the required initialization as described above. Any Actel design without this capability cannot be converted. Even with other FPGA and PLD types, dependence on the built-in power-on reset for proper operation of your circuit is not recommended, this function is typically unreliable.

Power-up programming protocol

The power-up programming protocol applies only to SRAM based FPGAs such as Xilinx LCAs and the Altera 8000 series. These are volatile devices and must be programmed on power-up each time. Identify the power-up programming scheme used, and whether the system is monitoring any of the programming pins. By default, these pins are not supported. In most cases emulation can be provided on a case-by-case basis.

8. Clocking Scheme Information

Identify all clock pins and the overall clocking scheme used in the design. This includes internally derived clocks and an explanation of how the clocks interact with each other (multiple clock sequence). Clocks and clock pins should be identified both in the description and on the block diagram. Latch enable description should also be included.

9. Critical Timing Information

The system timing requirements should be identified along with any special requirements for pin-to-pin propagation, set-up, or hold times. For PAL architecture devices, the timing is adequately described by the PLD device data sheet, but for FPGAs timing is more a function of the design and specific layout than of the data sheet parameters. Any special problems encountered with the speed or routing of the PLD or FPGA should be noted.

10. Input and Output Levels

Some FPGAs allow use of CMOS levels on inputs or outputs. Select the appropriate threshold levels for inputs. Some FPGAs, such as Xilinx and Actel, define V_{OH} = 3.84 V minimum at rated I_{OH}. Though not specified, these devices typically have outputs which will swing "rail-to-rail." The default specification for ULC outputs is a standard TTL compatible specification: V_{OH} = 2.4 V minimum at rated I_{OH}, and the standard outputs on smaller ULCs will *not* swing "rail-to-rail." These ULC drivers will not have any trouble driving TTL compatible inputs but may cause problems driving any linear circuitry such as resistor networks, caps, LEDs, and integrators. If you *require* V_{OH} = 3.84 V minimum and/or "rail-to-rail" output swing, please indicate; this will preclude the use of the UD technology and may require additional quoting.

11. Functional Input Patterns

Functional Input Patterns are optional for most ULC conversions. If they are not provided, however, there will be a significant impact on the conversion time, as they will have to be developed by MHS. See previous section (Test Vectors) for more information about vector formats. An adequate vector set must have at least 85% fault coverage, and all signal pins must toggle.

12. Any Requirements Different from FPGA

The ULC design methodology can frequently be used to develop parts which are not drop-in replacements for an FPGA or PLD. These might include special packaging requirements, an output drive, or faster timing. While this type of requirement should be well understood and quoted specifically prior to preparing a design checklist package, please include a detailed specification for such requirements, if any.

13. Environmental

Check required temperature range and any special processing requirements.

14. JTAG Support

MHS can support device series that employ support the JTAG standard, such as Xilinx XC4000, Altera Flex 8000, and the Lattice pLSI3000. Please indicate whether you require this support.

Table 1: Design Files by Family

PAL, GAL and FPLD Files

- Source File (.PDS, .ABL)
- JED fuse File

ACTEL FPGA Files

- .ADL Files
- .PIN File

ALTERA "Classic" Design Software Files

- .ADF File
- .LEF File
- .JED Fuse File
- .SCH Schematic File

ALTERA MAX Plus 2 Files

- All Design Files: .GDF, .{PF, .TDF, .FIT, .RPT
- .EDO EDIF Netlist (if using MAX2 software)
- .TBL Sim File Using Binary Radix with All Pins (or test patterns in truth table format)

AT&T ORCA FPGA Files

- EDIF Netlist File
- .BIT Programming File

LATTICE pLSI EPLD Files

- Verilog Netlist File
- JED Fuse File

XILINX FPGA Files

- .XNF Netlist File (XNF created by LCA2SNF)
- .LCA Post Layout File
- .BIT Downloadable File
- .MCS Downloadable File (created from .BIT)

Conversion to Standard Cell

The ULC to Standard Cell program is designed to provide an easy migration path with cost reduction for customers who potentially have very large volumes. The basic program involves first doing a normal ULC conversion, and shipping ULCs for the early production ramp. As the production volume moves into high gear, a second conversion is started, moving from the ULC into a second-stage cost reduction with a standard cell implementation, and then doing a smooth transition into the new device.

The decision to make this second transition involves a somewhat more detailed analysis than what is required initial ULC conversion. Unlike the initial ULC, there are significant Non-Recurring-Engineering charges (NREs) associated with the standard cell, as well as substantial volume commitments. There is also additional engineer effort and a longer conversion and ramp-up time.

The amount of additional savings will vary significantly from application to application. A large, heavily utilized, core-limited FPGA, with large RAM blocks but few pins, may benefit substantially by a shrunk core. But a lightly used FPGA, using all of the available pins, will only benefit if the pad-pitch can be reduced in the standard cell, or if better use of the entire periphery can be made. MHS can give a detailed estimate after the ULC conversion is complete; prior to that, only a very rough estimate can be made.

Due to the NREs and significant implementation time, the volume at which conversion to standard cell begins to make sense typically occurs if the ULC volume would fall between $250,000 to $500,000 minimum per year with a minimum run of at least one year. Thus the unit volume is heavily dependent on the cost of the ULC. A $15.00 device may only require a few tens of thousands of units to justify this second conversion, while a $2.00 device would require hundreds of thousands.

While MHS does all the conversion work, the ULC to Standard Cell conversion also requires greater customer involvement than the normal ULC conversion. A more detailed specification for the device is required, as well as a customer sign-off of the post-layout simulation. Due to this requirement, it is generally advisable that the customer provide the original vectors for simulation of the ULC, as machine generated vectors are very difficult to interpret. In addition, while having a working ULC based on the same technology minimizes the risk of something not working due to major changes in timing for example, it does not eliminate this risk. MHS only provides a normal ASIC commitment on the standard cell: that it will meet the provided specification and the approved simulation.

A standard cell device is an all mask-level custom device. To facilitate production start-up, an entire lot of wafers is run with the prototypes. This means that production volumes of can be phased-in and shifted over with MHS coordinating between the two much more rapidly than the normal lead-time of the standard cell would suggest. At later stages, the ULC can still be used to fill in unexpected peaks in production that don’t have sufficient lead-time to be filled with the standard cell.