XC4000 XCELL SPECIAL EDITION

A NEWSLETTER FOR XILINX XC4000 USERS

Special Edition April 1991

Development software for the XC4000 devices, Xilinx's third generation of Field Programmable Gate Arrays, has been available since late November 1990. Right after this first release, work started on the next release, which is planned for July 1991.

This special edition of the XCELL newsletter is intended to help the active XC4000 designer understand and overcome some of the limitations of the first release and its documentation.

Xilinx is committed to the continuous improvement and enhancement of the new XC4000 family. If you have any technical questions, call us at (800) 255-7778 or (408) 879-5199.

Randy Saldinger, editor

Table of Contents

XC4000 Family Status

First samples of the XC4005 are being evaluated by Xilinx Engineering. No errors have been found to date. Functional evaluation will be finished and speed characterization will begin this month (April). This will result in more accurate data sheet parameters and complete speed files. Limited samples will be available in April, with additional samples in June and July.

The May/June update of the XC4000 data sheet will add more package outlines and pin descriptions. The final XC4000 data sheet with complete timing specifications for the –10, –7 and –5 speed grades, more configuration details, and a more complete description of the JTAG option will be available in October or November.

The XACT 4000 Development System has been available on the PC since late November 1990. Versions for the Apollo and Sun workstations began shipping in late March 1991.

The next release, planned for July, will include several additions and enhancements. The algorithms used by the PPR utility are being improved. The most visible result will be significantly reduced routing delays and, subsequently, increased device performance. The software improvements will, of course, include fixes for the bugs identified in the initial release.

The bitstream generator is being verified using the first XC4005 samples, and will be included in the July release. (Customers receiving samples before July will receive a beta-site version.) The new download/readback cable, which connects to the serial port of a PC or workstation, will be shipped in July, along with device files for the XC4003 and XC4006.

Availability of XC4000 Devices

	Package	Samples	Production
XC4005 (14x14 CLBs, up to 112 I/O)	PG156 PC84 PQ160	2Q91 3Q91 4Q91	3Q91 4Q91 1Q92
XC4003 (10x10 CLBs, up to 80 I/O	PC84	3Q91	4Q91
XC4010 (20x20 CLBs, up to 160 I/O)	PG191	4Q91	1Q92
XC4008 (18x18 CLBs, up to 144 I/O)	PG191	4Q91	1Q92

Partition, Place & Route

The PPR program takes an XC4000 XNF file, partitions the design into LCA resources, and places and routes the design on the device. The role of PPR in the XC4000 design flow is similar to that of the XNFMAP, MAP2LCA and APR programs in the XC2000/XC3000 flow. This article explains how the PPR program works.

As PPR implements an XC4000 design, it reports its progress by displaying status messages such as "Checking Netlist" or "Running Maze Router." These messages correspond to different phases of the implementation process. Within each phase, one or more sub-programs may be called. PPR refers to these sub-programs as "sub-tools." For each phase of the process, the table on the opposite page defines the sub-tools used by PPR.

The various sub-tools used by PPR are normally transparent to the user. However, if PPR issues an error or warning message, the name of the sub-tool is used to identify the problem. For example, if an I/O pad carries a name which is reserved as an IOB block name (such as B12), PPR reports

```
Warning [PIC2LCA:PKG_PIN_NAME]
The block name could not be used
because it is a reserved name
```

The code enclosed in square brackets is similar to an error number, but identifies which sub-tool generated the message.

Because PPR's messages are not always self-explanatory, Applications has compiled a list of possible causes for each error, which begins on page 4. This list will be expanded. For updated information on PPR messages, contact Applications.

As mentioned above, PPR issues general status messages as it processes a design. These messages are also echoed to the PPR.LOG file in the working directory. PPR can be instructed to display more detailed information, or to write more detail into PPR.LOG. Increasing the level of detail can help in tracking down the source of a problem.

The file called XACTINIT.DAT in the \XACT\DATA directory defines four variables which control the level of detail:

```
/*/plog_file_level      = normal
/*/plog_stdout_level    = normal
/*/*/plog_file_level    = warning
/*/*/plog_stdout_level  = warning
```

The "plog_stdout" variables control the detail output to the screen, and the "plog_file" variables control the detail output to the PPR.LOG file. The "/*/*/" variables control a lower level of detail than the "/*/" variables. To change these variables, simply replace the "normal" or "warning" values with the desired level. Setting all of these variables to "verbose" usually gives as much information as you might want.

If you have any problems with the PPR program and none of these techniques helps to resolve them, please contact Xilinx Applications. We may not have an immediate answer, but we will do our best to find a solution.

The first release of the PPR program has some weaknesses, and we are trying to fix these problems as we find them. But we cannot fix problems we don't know about. If you find a problem with PPR or any Xilinx software, please let us know. Your feedback is always appreciated.

PPR Sub-Tools

PHASE	SUB-TOOL	FUNCTION
CONVERSION	xnf2mxn	*Converts the top-level XNF file and any unflattened lower-level XNF files to PPR's internal netlist format.*
	mxnflat	*Flattens any lower-level XNF files into the top-level netlist using FILE= parameters.*
PREPARATION	mxnprep	*Checks netlist for design errors and performs logic reduction.*
PARTITION	fgmake	*Creates function generators from all combinatorial logic.*
	ingroup	*Groups related items (such as a pad and a buffer, or a flip-flop and a function generator) to create "cells."*
	mxn2pic	*Creates netlist of cells. If no part type was specified, also determines an optimal XC4000 part to use.*
	cstmerge	*Merges schematic constraints from XNF file with constraints from CST file. (Constraints in CST file take precedence.)*
	mincut	*Performs preliminary MINCUT placement on cells.*
	blkmake	*Creates actual LCA blocks, such as CLBs and IOBs.*
IMPROVEMENT	gfdrf	*Improves upon the preliminary block placement. Number of iterations may be altered by IMPROVECOUNT= parameter.*
ROUTING	mxmazer	*Routes design using "maze router."*
LCA	pic2lca	*Converts internal netlist into standard LCA file format.*
REPORT	pprsum	*Generates a report (RPT) file for the resulting LCA.*

PPR Error Codes

Xilinx Applications has compiled the following glossary of PPR error messages. For each PPR error or warning message listed below, one or more possible causes are given. The messages are listed according to the error code generated by PPR, which appears in square brackets before the actual error message text. See the article "Partition, Place & Route" for more information.

`Abnormal program termination: Memory protection fault`

- Check for the condition where a single gate has one pin tied to V_{CC} or ground and another pin left sourceless.
- Check for an illegal I/O configuration, such using both an IBUF and an INREG for the same PAD (which is illegal because INREG requires both IOB input pins).
- Verify that no more than 4 global primary buffers (BUFGP) are used.
- Check for more than one source to an I/O PAD.
- Check that the QEMM-386 memory manager is installed.
- There may be a conflict between the Phar Lap memory extender and another device driver on the system. Remove unrelated statements from the CONFIG.SYS file and re-run PPR.
- If none of the above problems are found, contact Xilinx Applications. Please have information about your system configuration available.

`BLKMAKE:CONSTRAIN_INVALID`

- Try removing constraints on TBUFs or associated logic. These constraints may cause PPR to align TBUFs improperly, which prevents the output signal from being routed.

`BLKMAKE:INST_NO_FIT`

- Try removing constraints from the design. Some combinations of constraints may fool PPR into thinking that the design does not fit into the selected part.

`BLKMAKE:INST_NO_LOC`

- Verify that no more than 4 global secondary buffers (BUFGS) are used.

`BLKMAKE:NO_RESOURCE`

- Try removing constraints from the design. Some combinations of constraints may fool PPR into thinking that the design does not fit into the selected part.

`BLKMAKE:TBUF_FAIL_FIRSTTRY`

- This message indicates a problem in the partitioning phase that will likely be solved later in the process. If a real problem exists, a different message will be issued. Thus this message can be ignored.

`BLKMAKE:TBUF_LINE_UP_FAILn`

- Try removing constraints on TBUFs or associated logic. These constraints may cause PPR to align TBUFs improperly, which prevents the output signal from being routed.

`CSTMERGE:CST_NO_WHERE_PAD`

- Try removing constraints on unbonded IOBs. If these constraints are removed, this warning should not be issued.

CSTMERGE:ILLEGAL_PIN_LOCATION

- If the warning specifies the location as UXX, locate any PADU symbols in the design and remove the "Uxx" label from the symbol. The FutureNet interface does not know to remove this label and places it in the XNF file.
- Check for constraints on unbonded IOBs. If these constraints are removed, this warning should not be issued.

CSTMERGE:UNKNOWN_CONSTRAINT_CODE

- If a CST file is being used, check if there are "place instance" and "flag net" constraints which refer to the same instance name. Although one refers to a net and the other to a symbol, this situation causes PPR to issue this warning. If the constraints are specified on the schematic this does not occur.

FGMAKE:NET_NOSOURCE

- Check for the condition where a V_{CC} or ground signal is connected to the input of an INV or BUF symbol. Connecting the power signal directly to a boolean gate or flip-flop pin should eliminate this error.

FGMAKE:RAMROM_NO_INIT_VALUE

- If issued for a RAM, this message should be ignored. See "Using RAM and ROM" for more information on this topic.

GFDRF:NOBEL

- Check for a multiple-location constraint (such as LOC=TL;LOC=TR) on a global buffer (BUFGP or BUFGS). Removing the constraint should eliminate the error.
- Check for X (explicit) attributes in the design. Although nets tagged with an X flag will not be partitioned into a function generator, they will not always be external to a CLB. An X flag may sometimes cause this error message to appear.

MXMAZER:E_MISALIGNED_TBUFS

- Try removing constraints on TBUFs or associated logic. These constraints may cause PPR to align TBUFs improperly, which prevents the output signal from being routed.

MXMAZER:E_RESERV_CONFLICT

- Check for an illegal I/O configuration, such using both an IBUF and an INREG for the same PAD (which is illegal because INREG requires both IOB input pins).

MXMAZER:E_SOURCELESS_NET

- Check for the condition where a V_{CC} or ground signal is connected to the input of an INV or BUF symbol. Connecting the power signal directly to a boolean gate or flip-flop pin should eliminate this error.

MXMAZER:PATHOLOGICAL_NET

- Verify that no more than four decode outputs are indicated for a group of edge decoder inputs. PPR does not yet support the half-length splitters on the edge decode lines.
- If the design is targeted for an XC4008 and uses the boundary scan, a problem in the XC4008 data files causes this error. This will be corrected in the next release of PPR.
- Check for an illegal I/O configuration, such using both an IBUF and an INREG for the same PAD (which is illegal because INREG requires both IOB input pins).
- Two hard macros may be placed such that they share a single routing resource. Rerunning PPR to achieve a new placement should alleviate this problem. The next release of PPR will keep such hard macros apart.

MXMAZER:SIGNAL_COLLISION

- Verify that no more than four decode outputs are indicated for a group of edge decoder inputs. PPR does not yet support the half-

MXN:FILE_SYNTAX_ERROR

- Check for a large number of USER records in the XNF file. Some XNF interfaces may generate many of these records. Since they are not used or preserved by PPR, they should be removed from the XNF file to avoid this problem.

PIC:MWRITER_OBJECT_USED

- Verify that no more than four decode outputs are indicated for a group of edge decoder inputs. PPR does not yet support the half-length splitters on the edge decode lines.

XNF2MXN:BAD_FILE_OPEN

- If the specified XNF file matches the name of a primitive symbol — such XOR.XNF or RAM.XNF — check for a symbol of that type with unconnected input pins.
- If the specified XNF file is a lower-level module referenced by a FILE= attribute, check that the indicated file is in the current directory.

XNF2MXN:BAD_PARAM

- If a PAD symbol and its corresponding buffer are on different levels of the hierarchy and a FILE= parameter is used, this error may be issued. The quickest workaround is to remove the FILE= parameter and allow the schematic-to-XNF translator to do the flattening, since FILE= has no meaning to PPR.
- Check for a W (weight net) attribute without a value. A W flag must carry a numerical value to define the net weight.

XNF2MXN:BAD_PIN_NAME_MATCH

- Check for more than one source to an I/O PAD.

XNF2MXN:NO_EXT_PAD

- If a PAD symbol and its corresponding buffer are on different levels of the hierarchy and a FILE= parameter is used, this error may be issued. The quickest workaround is to remove the FILE= parameter and allow the schematic-to-XNF translator to do the flattening, since FILE= has no meaning to PPR.

LCA2XNF Calls PPR Output "Unrouted"

For timing simulation, the LCA2XNF program is first used to back-annotate delay information from the routed LCA file. However, if used on an LCA file created by the PPR program, LCA2XNF reports the following warning:

```
Warning 23: Design is unrouted.
Unit delay model will be generated.
```

Although the LCA file generated by PPR **is** routed, the file does not contain any delay information. Thus, LCA2XNF assumes the design is unrouted and creates a unit-delay model.

There are two methods for writing delay information into the LCA file. The easiest is to run the MakeBits program with the –w option, which causes MakeBits to re-write the LCA file with net delays. This is the method used by the XMAKE program.

The other method of writing delays into the LCA file is to load the design in the XACT Design Editor (XDE) and save it again. XDE calculates the delay on each net when the design is loaded, and always saves this information in the LCA file.

Why QEMM?

The XACT 4000 Development System for the PC includes the QEMM-386 memory manager (Quarterdeck Office Systems). Many users have called Xilinx Applications to ask why QEMM is necessary. This article explains why the program is included with every XACT 4000 package.

The XACT software requires more memory than the 640 KB accessible through MS-DOS. Xilinx avoids this limitation by using a "DOS extender" to run programs in extended memory. A DOS extender allows programs to run in the "protected mode" of the '286 and later processors. In '286 protected mode, up to 16 MB of memory can be accessed; in '386 and '486 protected mode, the limit is 4 gigabytes.

Previous releases of the XACT software have used the DOS/16M extender (Rational Systems) to access extended memory. With the release of the XACT 4000, several of the larger programs have been switched to the 386 | DOS-Extender (Phar Lap Software) to take advantage of the more powerful '386 and '486 processors. The result is a significant speed improvement over the '286-mode programs.

The XACT Design Manager (XDM) and the XMAKE program still use the '286-based DOS/16M extender. DOS/16M allows these programs to call the Phar Lap-based programs, such as PPR or the XACT Design Editor. If XDM and XMAKE used the Phar Lap extender, they could not call these '386-based programs. Without this ability, automatic design translation with XMAKE would be impossible.

Having both the DOS/16M and Phar Lap extenders running simultaneously does generate one problem. Neither extender is capable of controlling the memory usage of the other, which may lead to memory conflicts. **To avoid potential conflicts, the QEMM-386 memory manager is used.**

QEMM-386 supports a specification called the "virtual control program interface" (VCPI). If QEMM is installed, both the DOS/16M and Phar Lap extenders will access extended memory through the VCPI interface. This allows QEMM to control all memory allocation and prevents conflicts between DOS/16M and Phar Lap.

QEMM-386 also uses the extended memory on a '386 or '486 machine to emulate expanded (EMS) memory. QEMM converts all extended memory into expanded memory, allowing programs which use expanded memory to run in extended. For this reason, memory statistics generated by QEMM-386 or by Quarterdeck's Manifest program will show that zero extended memory is available. However, since QEMM supports the VCPI interface, this converted memory is still available to DOS/16M and Phar Lap.

QEMM-386 does allow extended memory to be reserved, using the EXTMEM parameter on the QEMM command line. **However, any such reserved memory is unavailable to QEMM, and therefore to DOS/16M and Phar Lap.** This may cause the XACT software to run out of memory, even if there is sufficient extended memory installed. If in doubt, use QEMM with no optional parameters (this is the default).

For future releases of the XACT 4000 software, we are examining alternatives to the DOS/16M and Phar Lap memory extenders which will make QEMM-386 unnecessary.

Hard Macros and Carry Logic

The XC4000 architecture provides fast carry logic in each CLB, with dedicated connections to propagate carry signals between adjacent CLBs. This allows for faster and more efficient implementations of adders, subtractors, counters and the like. Many of the hard macros provided with the XACT 4000 development system use the carry logic to provide high performance. This article describes how the carry logic works, and explains potential problems in designs which use the hard macros with carry logic.

The carry logic in an XC4000 CLB has approximately 40 possible configurations. Each configuration performs a unique carry-out function, which may depend on the function generator inputs and/or the carry-in to the CLB.

If a CLB uses the carry logic, the desired configuration is identified by a mnemonic assigned with the ConfigCarry command in the XACT Design Editor (XDE). Any hard macro which uses carry logic will generate the appropriate ConfigCarry statements in the LCA file. Examples of carry logic mnemonics are ADD-FG-CI (add in the F and G function generators with carry-in from the CIN pin) and INC-G-F1 (increment in the G function generator with carry-in from the F1 pin). Additional tags determine the flow of carry signals between adjacent CLBs.

LCA2XNF Errors

To simulate designs with carry logic, the LCA2XNF program uses the ConfigCarry mnemonic to create a model for the carry logic in each CLB. The resulting XNF file contains gates to model the dedicated carry logic.

Two carry logic modes are not handled properly by LCA2XNF Versions 4.00 and 4.01:

- An error in the ADDSUB-FG-CI model causes the carry-out from the first half of the CLB to be ignored. This results in functional errors when the design is simulated. The only hard macros which use this carry logic mode are the adder/subtractors (ADSU8H, ADSU16H, ADSU24H, and ADSU32H). Expect to see simulation errors on these hard macros, and on manually-entered carry logic designs which use the ADDSUB-FG-CI configuration.

- An error in the ADDSUB-G-F1 model causes LCA2XNF to abort with Error 138:

```
Pin CIN not found for carry mode
```

This occurs because LCA2XNF expects to find a signal on the CIN pin, but none is required in this configuration. The only hard macros which use this carry logic mode are the 24-bit and 32-bit adder/subtractors (ADSU24H and ADSU32H). Both of these hard macros have other problems as well (described below).

Both of these modeling errors will be fixed in the next release of the LCA2XNF program. Contact Xilinx Applications for the latest version of LCA2XNF.

Hard Macro Errors

Aside from the modeling errors explained above, some of the hard macros contain design errors which cause LCA2XNF to abort. The problem macros are the 24-bit and 32-bit adder/subtractors (ADSU24H and ADSU32H), the 32-bit magnitude comparator (COMPM32H), and the 24-bit and 32-bit counters (CUP24H and CUP32H). Minor design errors in these hard

macros cause LCA2XNF to abort with one or more of the following Error 138 messages:

```
Pin G4 not found for carry mode
Pin G1 not found for carry mode
Pin F4 not found for carry mode
```

Corrected versions of these five hard macros are available from Applications or the Xilinx Technical Bulletin Board. After installing the new hard macro files, run the design through PPR again to incorporate the corrections.

DRC Errors on Hard Macros

Aside from the five macros mentioned above, all hard macros in the first release of the XACT 4000 are correct. However, the DRC in XDE may report warnings or errors on some hard macros. These errors occur because the DRC does not understand every legal carry logic configuration.

The following DRC errors may occur on hard macros:

```
FATAL 301: input unused in CLB
  but 1 input pip is on
Warning 305: function has inputs
  but no outputs
FATAL 319: carry function must be
  configured with ...
Warning 605: pin unused in CLB
```

These errors should be ignored. A corrected DRC will be included with the next release of XDE.

Timing Simulation with Carry Logic

The XC4000 carry logic allows for quick propagation of carry signals through a chain of CLBs. Only the first and last CLBs in the chain incur delays greater than T_{ILO}: specifically T_{CARRY} for the first CLB (8 ns for the -7 speed grade) and T_{SUM} for the last CLB (9 ns for -7). Each CLB in between adds a delay of T_{BYP} to the carry signal (2 ns for -7). Longer carry chains are more efficient, since each new CLB adds only 2 ns.

In simulation, however, carry logic signals appear to be much slower. This is due to errors in the delay models generated by LCA2XNF Versions 4.00 and 4.01. For the middle CLBs, the delays used by LCA2XNF are about 7 times too large. There is no problem with the carry logic or the hard macro itself. The simulation model is at fault.

The LCA2XNF program is being fixed to model these carry logic delays correctly. Contact Applications to obtain the latest version of LCA2XNF.

Workarounds for Hard Macro Problems

- **Functional errors in simulation of adder/subtractor hard macros** → contact Applications for a corrected version of LCA2XNF (fixes modeling errors).
- **Error 138 in LCA2XNF for ADSU24H, ADSU32H, COMPM32H, CUP24H, or CUP32H hard macros** → use corrected version of hard macros and contact Applications for the latest version of LCA2XNF (fixes modeling errors).
- **Simulation shows large delays through carry logic** → contact Applications for a corrected version of LCA2XNF (fixes carry logic delay model).
- **XDE DRC warnings or errors on hard macros** → ignore messages.

Using RAM and ROM

Using the MEMGEN memory compiler, read/write or read-only memories can be easily implemented in an XC4000 device. A single XC4000 CLB can function as either a 32x1 or a 16x2 memory. For a RAM or ROM which requires more than one CLB, the MEMGEN program automatically generates the necessary control logic.

The first release of the XACT 4000 software contains a couple of bugs in the handling of RAM and ROM. One of these affects the control logic in any MEMGEN-produced memory. The other impacts the simulation of a ROM with P/C-Silos.

PPR and MEMGEN Control Logic

A memory created by MEMGEN is linked to a schematic symbol by a "FILE=" parameter. This parameter specifies the name of the XNF file produced by MEMGEN. When the PPR program finds the FILE= parameter in the input design file, it searches for and reads the XNF file for the MEMGEN module. Unfortunately, when PPR Version 1.00 reads in this lower-level XNF file, it loses the inversion tag on the bubbled inputs of Boolean gates. Thus any control logic generated by MEMGEN will no longer be correct.

Xilinx Applications has written a program to replace inverted inputs with discrete inverters, which prevents PPR from losing the inversion. The XNF2PPR program is available from the Xilinx Technical Bulletin Board or from Applications.

Initial Values on ROM

The contents of an XC4000 ROM are contained in the configuration bitstream. These contents are defined by placing an INIT value on a 16x1 or 32x1 ROM primitive. For larger memories, MEMGEN automatically splits the ROM data into the necessary 16x1 or 32x1 segments. (Note that an initial value on a RAM cannot be guaranteed. See the box below for details.)

The new P/C-Silos interface does not handle initial values on a ROM correctly. XNF2SILO Version 4.00 corrupts initial values on both 32x1 and 16x1 ROMs, which are used to construct all larger memories.

An initial value on a 32x1 ROM is represented by an 8-digit hexadecimal number. The least significant bit of this number is written into location 31 of the ROM, the next significant into location 30, and so on to location 0. However, XNF2SILO defines the ROM contents in the opposite direction: the least significant bit is written into location 0, the next significant into location 1 and so on. Thus a ROM will read out backwards in simulation. This bug is fixed in Version 4.02 of XNF2SILO, available from the Xilinx Bulletin Board or from Applications.

Initial Values Not Supported on RAM

The documentation for the XACT 4000 software states that an initial value may be assigned to the RAM in an XC4000 CLB. In fact, it is **not** possible to define the contents of the RAM at power-up.

Although the RAM contents can be defined in the configuration bitstream, the Write Enable circuitry is combinatorial, and is potentially alive but not properly defined during configuration. Thus, conflicting data may be written into the RAM. The initial contents of a RAM can, therefore, not be guaranteed, but the contents of a ROM can be.

As with all other SRAMs, data must be explicitly written into the RAM in an XC4000 CLB before anything meaningful can be read.

4005PC84 Package Files

The first release of the XACT 4000 software includes support for the XC4005 in an 84-pin PLCC package. The 4005PC84 package file shipped with the PC-based development system contains a few errors. The bonding of I/O pads to package pins is represented incorrectly for some IOBs.

The errors in the 4005PC84 package file appear as inconsistencies between the pinout described in the XC4000 data sheet and that shown by PPR or the XACT Design Editor (XDE). For example, a V_{CC} pin appears as a user I/O in XDE.

To identify the erroneous package file, examine the first line of the 4005PC84.PKG file found in the \XACT directory. The first line of the bad package file is as follows:

```
package 4005pc84 1 2 1 0
```

With this version of the package file installed, **an LCA file generated for a 4005PC84 design will be incorrect and will not produce a valid bitstream.**

The corrected package file is available from Applications or the Xilinx Technical Bulletin Board. The first line of this corrected package file reads:

```
package 4005pc84 1 4 1 0
```

Any designs processed with the incorrect package file must be completely re-processed from an XNF file once the new package file is installed. Re-run the PPR program to generate a correct LCA file.

Unconnected Hard Macro Pins

The XACT 4000 hard macros are designed at the CLB level and offer guaranteed performance. The tradeoff in using hard macros instead of gate-level "soft" macros is a minor loss of flexibility. Soft macro inputs or outputs can be left unconnected, and the unneeded logic will be removed by PPR. Hard macros do not provide that luxury.

If a hard macro input pin is left unconnected, the unused signal will be tied off appropriately. For example, an unused clock-enable will be tied to V_{CC}. However, no logic reduction is performed on the unused input. Any logic made unnecessary by an unused input is simply disabled, not removed.

If a hard macro output pin is left unconnected, no logic is removed. Although the output signal does not appear as an unrouted net in the LCA file, the source logic is still present and DRC errors may be generated. There is no workaround for unconnected hard macro outputs except to manually remove the unused logic in the XACT Design Editor.

Future releases of PPR will provide better optimization of unused hard macro inputs and outputs. In the meantime, using hard macros may mean sacrificing some flexibility.

PPR Creates Underscore Net Names

An XC4000 design implemented by the PPR program may contain some nets for which an underscore character and an integer have been appended to the original user name. PPR splits a signal into multiple segments for one of two reasons, both described below.

- The XC4000 CLB provides several "feed-through" paths which can be used to buffer high-fanout signals. A net attached to any of the C inputs (C1 to C4) may be fed directly to the XQ or YQ output. Doing so reduces capacitive loading on high-fanout signals. When PPR buffers a net by passing it through a CLB, two nets are created from the original signal.

- The XC4000 IOB provides two input pins, I1 and I2. These are located on opposite sides of the IOB, which allows both a direct input and registered input to be accessed internally. Both pins are also used if an input signal sources both the edge decoders and internal logic. In this case, PPR must split the net into two segments, since two source pins are used.

Whenever PPR splits a net, two nets are created from the original signal. PPR names these segments by appending "_1" and "_2" to the original net name. This identifies these nets as segments of the original signal.

This renaming of nets may cause some confusion when simulating an XC4000 design. If a net is flagged as unknown by the simulator but is listed in the PPR report file, the net may have been split. Append a "_1" to the original name to probe the first segment of the split net.

Split nets can also be examined in the XACT Design Editor, by using the QueryNet command with the wildcard character. For example, the following command would find all segments of the original net named ADD:

```
QueryNet ADD_*
```

We are examining alternative methods of handling these split nets for future releases of PPR.

Simulating with P/C-Silos

A new interface for the P/C-Silos simulator was released along with the XACT 4000 software. This new interface, which includes XNF2SILO Version 4.00, adds support for XC4000 designs.

Unfortunately the new Silos interface contains a number of bugs. Most of the problems in the XNF2SILO translator will cause the Silos program itself to report errors or warnings. We are planning another update to the DS22 package (P/C-Silos and the XNF2SILO program). Until then, the best way to avoid the pitfalls in XNF2SILO Version 4.00 is to obtain Version 4.02 through Xilinx Applications or the Xilinx Technical Bulletin Board.

XNF2SILO Version 4.02 incorporates some modeling changes made after the 4.00 version. These changes require that LCA2XNF Version 4.01 or later be used in conjunction with XNF2SILO. If LCA2XNF Version 4.00 is used with XNF2SILO Version 4.02, timing errors may occur. The newer version of LCA2XNF is also available through Applications or the Xilinx Bulletin Board.

XC4000 Data Sheet Corrected and Clarified

Since the publication of the preliminary data sheet for the XC4000 devices, a few errors have been found. Some of the more significant corrections are noted below.

- Page 18: The multiplexers on the far right side of Figure 18 — all controlled by the boundary scan EXTEST signal — are upside-down for each IOB three-state and input signal. The "1" and "0" labels on the multiplexers driving the T and I pins should be reversed. Obviously, if EXTEST is Low, the boundary scan logic becomes transparent.

- Page 23: The schematic for Synchronous Peripheral mode in Figure 22 shows the CCLK output going to subsequent slave devices. Actually, CCLK is an input to the lead LCA, and any slave LCAs must be clocked by the same CCLK which controls the lead device.

- Page 26: The chart at the top of Figure 25 shows 1111 as the postamble code. The correct postamble for XC4000 devices is 0111.

- Page 27: The flow chart in Figure 26 shows an approximate time of 150 ns to clear each frame of configuration memory. The correct clear time is ~750 ns per frame.

- Page 34: The pin description for CCLK says that it is an input for Slave mode configuration. CCLK is also an input for Synchronous Peripheral mode.

- Page 34: The pin descriptions state that the mode pins (M0, M1 and M2) and the boundary scan pins (TDO, TDI, TCK and TMS) may be configured as user-programmable inputs and/or outputs. However, the development software does not assign I/Os to these pins unless explicitly specified. Special pads are provided for this purpose.

- Page 35: The pin description for the read and write strobe inputs is somewhat vague on the interpretation of $\overline{WS}$ and $\overline{RS}$ being active simultaneously. If both $\overline{WS}$ and $\overline{RS}$ are Low, the cycle is ignored.

- Page 35: D0–D7 are used in Peripheral and Master Parallel modes, not in Slave or Master Serial modes as indicated. After configuration, these are user-programmable I/O pins.

- Page 39: In the DC characteristics table, Note 1 applies to the Quiescent LCA supply current (I_{CCO}).

- Page 40: The specification for "T_{CH} / T_{CL}" is somewhat confusing. Both the minimum clock High time (T_{CH}) and minimum clock Low time (T_{CL}) are specified. The minimum for each is 5.5 ns for –10 and 4 ns for –7.

- Page 41: The timing for a 16x1 RAM is identical to the timing shown for a 16x2 RAM.

- Page 41: The delay specifications for the CLB fast carry logic — T_{CARRY}, T_{SUM} and T_{BYP} — are shown in the minimum column. These are actually maximum values.

- Page 47: The maximum frequency and minimum High and Low times for CCLK are incorrect. The maximum CCLK frequency (F_{CC}) should be 8 MHz, and the corresponding High and Low times, T_{CCH}=T_{CCL}=60 ns.

- Page 49: In the lower diagram, V_{CC} is shown on pin B9. As the upper diagram shows, V_{CC} is actually on pin B8.

- Page 49: The footnote reads "unlabeled pin = unrestricted I/O pin." However, there are 13 unconnected package pins which are not labeled either. These are A4, A12, D1, D2, D16, E15, M1, M2, M15, N16, R5, R12 and T12.

Output Levels

XC2000 and XC3000 devices have a complementary output structure, an n-channel output transistor pulling Low, a p-channel output transistor pulling High. This results in rail-to-rail switching. In the absence of any dc load, the active output levels are exactly 0V or V_{CC}.

This scheme offers well-defined output levels, but suffers from two disadvantages:

- When input thresholds are set for TTL compatibility (1.4 V), the output switching delays are asymmetrical. The High-going transition is recognized as High after only 24% of the total rise time from ground to V_{CC}, the Low-going transition must move 76% of the way from V_{CC} to ground, before it is recognized as Low.

This lack of symmetry is partially compensated by the difference in output impedance (60 Ω pulling High, 25 Ω pulling Low), but it still leaves a significant difference between the delays for the two edges.

The large output swing also generates excessive noise and crosstalk and causes current spikes in the ground inductance, creating ground-bounce voltages that might cause errors.

- The second disadvantage of the complementary output structure is more subtle: The junctions forming the p-channel pull-up transistor act as a parasitic diode that prevents the I/O pad from ever going more than ~1 V positive with respect to the V_{CC} pad.

In normal operation, this is irrelevant, but there are cases where this diode causes harm: Assume a bus-oriented system where at least one of the bus lines is being driven High to somewhere between 3.5 and 5 V. Assume a device with a complementary output connected to this bus line, but not being powered up. The High level on the bus line will forward-bias the clamping diode and will try to pull the V_{CC} node of the powered-down device High. If this V_{CC} node is lightly loaded, the bus may be successful in pulling the inactive V_{CC} to 2.5 V or higher, and the rest of the system may be operating properly. If the V_{CC} node is heavily loaded or decoupled, then the bus will be clamped below its normal High level, and the system will not work properly. This is called "bus hogging." It is an inherent characteristic of complementary outputs.

For the XC4000 family, Xilinx chose an output structure with two n-channel devices, one pulling Low, the other pulling High. This structure overcomes the two problems mentioned above.

The output High level is one threshold below V_{CC} (similar to the popular TTL "totem pole" structure, where the High level is one or two diode drops below V_{CC}). The High-to-Low switching speed is improved because the output High level is lower, and the two active levels are almost symmetrical around the 1.4 V input threshold.

The n-channel output pull-up transistor does not create a clamping diode to V_{CC}; inputs can, therefore, be pulled higher than V_{CC}. A separate protection device clamps at ~7 V, protecting the input gate oxide against destructive breakdown from electro-static discharge.

Other manufacturers of high-performance CMOS devices are incorporating similar non-complementary output structures. The only disadvantage is that such outputs cannot drive devices with CMOS input thresholds. A pull-up resistor would remedy this, but only for slow interfaces.

Since XC4000 outputs cannot drive HC-type inputs, use HCT devices instead. Because the XC4000 outputs are not compatible with a 2.5 V input threshold, the CMOS input threshold option was eliminated from the XC4000 devices. It was considered meaningless or confusing to offer a device that cannot drive its own inputs.

Xilinx Programmable Gate Array Training Courses

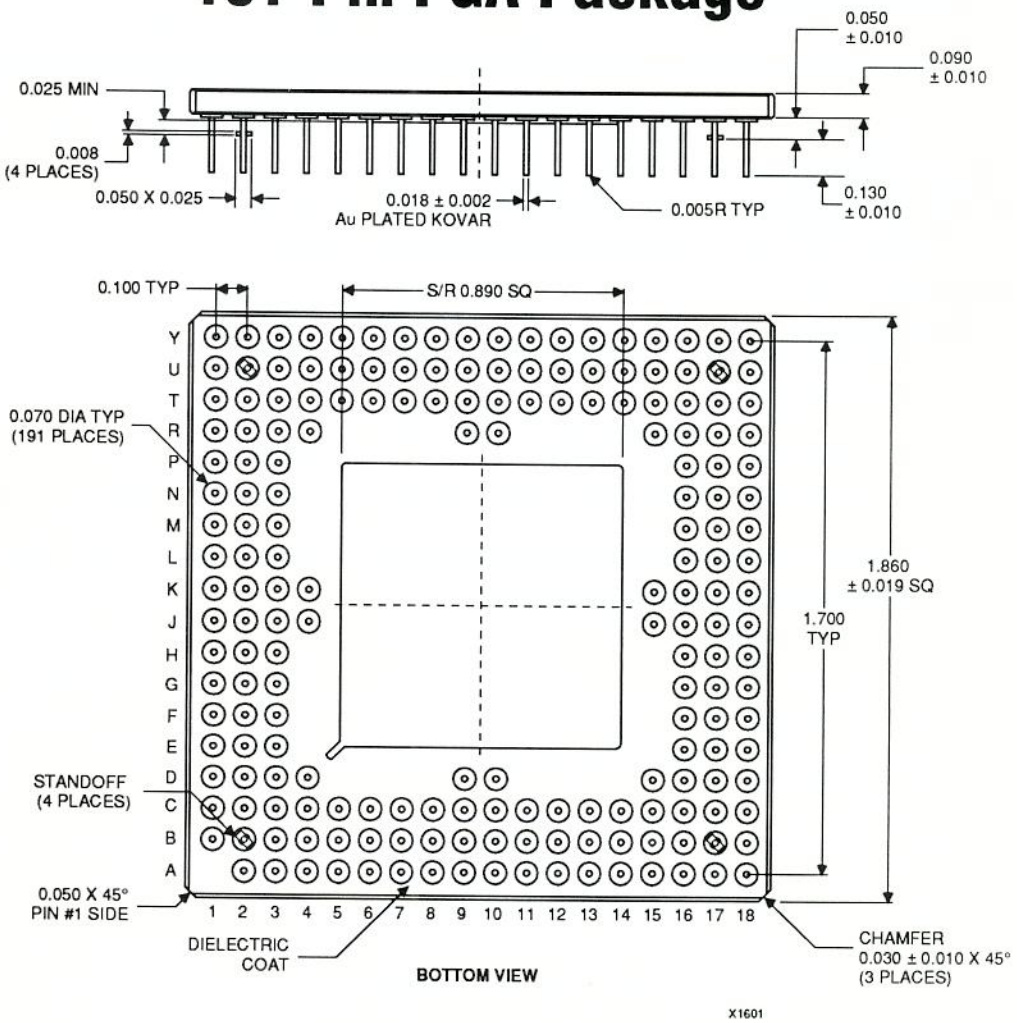
Two training courses are currently being offered, an XC3000 family course and an XC4000 family course. A combined course will be offered starting later this quarter and will provide a quick way to get up-to-speed on all device families. All courses provide hands-on experience as well as an in-depth look at the appropriate architecture and design implementation options.

Each course is available every month in the factory. *We can also bring the course to your facility.* For more information, please contact Annie Smith, Xilinx Training Coordinator, at (408) 879-5090.

The chart below describes the tuition and recommended prerequisites for the various classes.

Days	Device Family	Prerequisite	Tuition
4	XC3000 and XC4000	None	$1000
4	XC3000	None	$1000
2	XC4000	Familiarity with XC3000	$750

191-Pin PGA Package



New Structure for XACT Documentation

With the release of the XACT 4000 Development System, Xilinx has introduced a new organization for the user documentation. Three new binders are included with the XC4000 implementation package: the *XACT 4000 Design Implementation User Guide*, the *XACT 4000 Design Implementation Reference Guide*, and the *XACT Applications* binder. The user guide describes the basics of implementing an XC4000 design, including the MEMGEN memory generator and the PPR program. The reference guide explains the function of specific programs in the XC4000 design flow, and explains the options and parameters used by each program. The *Applications* binder is currently empty but is intended for future applications notes and tutorials.

The schematic editor and simulator documentation has also been updated. As support for the XC4000 family is added to each schematic interface, two new binders will be included with the interface package: the *XACT Design Interface User Guide* and the *XACT Design Interface Macro Libraries*. The user guide explains schematic entry and translation for the appropriate schematic editor. This user guide also provides a home for the simulator documentation, which is shipped with each simulation interface package. The *Macro Libraries* binder describes the primitives and macros contained in the Xilinx libraries.

With future releases of the development software, the remaining documentation will be updated for this new binder set, and a comprehensive index will be created for each binder. We hope that these improvements will provide faster answers to your questions and make your LCA design easier.

XILINX

2100 Logic Drive
San Jose, CA 95124-3450

U.S. POSTAGE
PAID
FIRST CLASS
PERMIT NO. 2196
SAN JOSE, CA