XCELL

THE NEWSLETTER FOR XILINX PROGRAMMABLE GATE ARRAY USERS

Issue 4 Fourth Quarter 1989

LCAs Offer Flexibility, Modern Technology, Low Power, and High Performance

Flexibility

Xilinx Logic Cell Arrays provide not only the advantages of customized VLSI, but their re-programmability can add important flexibility in manufacturing

The user can change the hardware configuration inside the LCA without affecting the printed circuit board. Hardware has become as flexible as software.
Here are some examples:

Hardware modifications, design fixes, or design improvements can be loaded into the system even after it is installed in the field.

The manufacturer can produce one common hardware design in high volume, then customize it to specific market requirements, e.g. for export to different countries, or customize it to different levels of sophistication, by adding or deleting features right on the shipping dock or even on the end-user's premises.

This combines the economy of high volume production with the advantages of product diversification and fast response to customer demands and market shifts.

Modern Technology

Programmable Gate Arrays compete successfully against established low-cost microprocessor peripheral circuits, which have not kept up with IC technology advances. Major microprocessor manufacturers still sell antiquated n-channel peripheral devices dating back to the seventies, but designers today demand the low power consumption and higher speed offered by 1989 CMOS technology.

Counters are a good example: Dedicated microprocessor peripheral counters can barely reach a clock speed of 10 MHz, but Xilinx LCAs can count synchronously at 40 MHz, asynchronously up to 100 MHz, and can be combined with inexpensive ECL two-modulus prescalers to resolve up to 1000 MHz. Standard ICs cannot come close to this performance.

Low Power

High power consumption is another drawback of these older n-channel devices. Today's board densities tolerate less heat per device, and more equipment is being battery-powered. This makes CMOS the technology of choice for almost all modern designs.

While the manufacturers of established peripheral ICs are reluctant to redesign their 10-year old circuits ("Why go through the trouble and expense of creating a new part, only to compete with yourself?"), Xilinx offers a modern, customizable and therefore perhaps simplified solution using 1990 low-power, fast CMOS technology. Xilinx Programmable Gate Arrays have a static power consumption of only a few milliwatts and can be powered down to the microwatt level.

High Performance

Xilinx LCA performance might beat even the fastest gate arrays, obviously not through raw circuit speed, but because the designer using LCAs has more freedom to experiment, to learn from the experiment and improve the design. (Software designers do this all the time.) In a conventional gate array project, such experimentation would be reckless and would jeopardize any budget in time and money. In an LCA it is feasible, fast, free, and fun. PA

Table of Contents

XILINX

Component Availability

(December 1989)

		44 PIN	48 PIN		68 PIN		84 PIN		100 PIN		132 PIN		164 PIN	175 PIN	
		PLASTIC PLCC	PLASTIC DIP	CERAMIC DIP	PLASTIC PLCC	CERAMIC PGA	PLASTIC PLCC	CERAMIC PGA	PLASTIC PQFP	CERAMIC CQFP	PLASTIC PGA	CERAMIC PGA	CERAMIC CQFP	PLASTIC PGA	CERAMIC PGA
		-PC44	-PD48	-CD48	-PC68	-PG68	-PC84	-PG84	-PQ100	-CQ100	-PP132	-PG132	-CQ164	-PP175	-PG175
XC2064	-50		C	I	C I	C I M									
	-70				C I	C I									
	-100				C *	C *									
XC2018	-33							M B							
	-50				C I		C I	C I M B							
	-70				C I		C I	C I							
	-100				C *		C *	C *							
XC3020	-50				C I		C I	C I M B	C I*	C I M B					
	-70				C I		C I	C I	C I*	C I					
	-100				C		C	C *	C *	C *					
XC3030	-50	C I			C I		C I	C I M	C I*						
	-70	C I			C I		C I	C I	C I*						
	-100	C *			C *		C *	C *	C *						
XC3042	-50						C I	C I M	C I*	C I M B*	C I*	C I M B			
	-70						C I	C I	C I*	C I*	C I*	C I			
	-100						C *	C *	C *	C *	C *	C *			
XC3064	-50										C I*	C I M			
	-70										C I*	C I			
	-100										C *	C *			
XC3090	-50												C I M	C I	C I M B
	-70												C I	C I	C I
	-100												C *	C	C

* In Development

Current Software List

The following is a list of the current software revision levels for Xilinx's development system products. This is a listing of the diskettes currently being shipped with each of the development system products, as of December 1, 1989.

DS112 ENHANCED SERIAL CONFIGURATION PROM PROGRAMMER
XPP program ver. 3.00

DS21 XACT Graphical Editor ver. 2.30
XACT ver. 2.12
DOS 16/M Loader ver. 2.49
Xilinx Design Manager ver. 1.0
XC3030/XC3042 die files
XC3020-PC84 package file
XC3042-PG132 package file
XC3064 Update
CQ164, CQ/PQ100 package files
Speeds file update: 5/19/89

DS22 P-SILOS (16K) ver. 2.20
P-Silos ver, 3C. 8–16K
xnf2silo ver. 2.12

DS221 P-SILOS (5K) ver. 2.20 (fomerly DS122)
P-Silos ver. 3C.8–5K
xnf2silo ver. 2.12

DS23 ADI ver. 2.21

ADI ver. 2.21
Speeds file update: 5/19/89

DS28 XACTOR ver. 2.10
XACTOR 2.10

DS31 FUTURENET DASH INTERF. ver. 2.31
FutureNet interface and library, PIN2XNF ver. 2.31

DS311 FUTURENET TTL LIBRARY (formerly DS 40)
FutureNet TTL Library

DS32 SCHEMA II+ INTERFACE ver. 2.30c

DS33 DAISY INTERFACE (DNIX) ver. 3.1

DS34 MENTOR INTERFACE ver. 2.20
Mentor IDEA interface&library ver. 2.20

DS35 OrCAD/SDT INTERFACE ver. 1.0
OrCAD/SDT interface and library ver. 1.0

DS 51 SCHEMA II+, ADI, AND XACT ver. 2.30
Schema II+ ver. 2.21
Xilinx/Schema interface ver. 2.30c
DS23 ADI ver. 2.21
DS21 XACT ver. 2.30
Speeds file update: 5/19/89

DS54 DASH-LCA and ADI ver. 2.30
DASH-LCA ver. 4.10d
PIN2XNF ver. 2.31
DS23 ADI ver. 2.21
DASHEVAL
Speeds file update: 5/19/89

DS53 DASH-LCA, ADI, +XACT ver. 2.30
DS54 ver. 2.30
DS21 XACT ver. 2.30

DS501XACT Dev. System ver. 2.31
DS21 ver 2.30
DS23 ver 2.21

DS501-AP1 XACT Dev. System ver. 2.21 on Apollo

ADI, MakeBits, MakeProm ver. 2.21
DS21 ver 2.30 (on PC)

DS501-SN1 XACT Dev. System ver. 2.21 on SUN3

ADI, MakeBits, MakeProm ver. 2.21
DS21 ver 2.30 (on PC)

XACT Update Comes In March

Xilinx will soon release XACT 3.0, an update to the XACT Design Editor that includes a wider range of color options, a more user-friendly interface, and additional features to provide more capability with less effort on the part of the user.

The basic four-color display has been expanded. Now you can use any of the supplied 4, 8, or 16-color palettes, or choose from the 64 available colors to make your own custom palette. Whereas older versions only highlighted in red, XACT 3.0 lets you choose the highlight color. It's easier to distinguish between two highlighted nets since you can give them different colors. And you can also assign permanent net colors. You can choose a default color other than yellow, or better yet, assign different colors to different sets of signals. This makes it much simpler to trace a routing path while editing.

In addition to new features, this release contains improvements to old commands. The XACT autorouter has been redesigned to give simpler, straighter routing paths. And the TIE option is smarter and avoids any potential stack-overflow problems.

Since designs are becoming larger, and more people are using XACT to optimize performance, many new commands have been added to make the editing process simpler and faster.

Below are some of the new commands in XACT 3.0:

ALIGNPIN - Moves a net to a specific pin. This command is a fast, effective way to line up pins in order to take advantage of longlines.

APRCST - Sets constraint file values. This is easier and faster than using a text editor. Also, with the correct SHOW options set, the screen can display locked and prohibited blocks, to show the effects of the constraints as they are added to the file.

BLKCOLORS - Assigns colors to all items in the block editor display.

COLORBLK - Assigns a chosen color to a block. This command is useful for making registers and counters stand out from the rest of the logic.

COLORNET - Assigns a permanent color to a net. This command makes nets easier to find and trace. Critical nets can be colored to show their importance. Nets that are currently being routed can be colored to make them stand out from the rest of the interconnect.

HILIGHT - Temporarily colors routed interconnect and pin stubs. Hilight has been changed to allow you to select the color that each net should be drawn in.

HILIGHTWIDE - Changes the width of the highlight lines. Any net highlighted with this option will be drawn with "fat" lines. This is a very effective way to make hilighted nets stand out on printouts and on monochrome screens.

PIECOLORS - Assigns colors to all items in the Editor screen.

ROUTEPOINT - Routes a net along a general path specified by the user. Rather than letting AUTOROUTE choose the entire route, or having to specify every pip with EDITNET, you choose the general path that the route should follow, and XACT makes all of the connections. This combines the tight control of EDITNET with the speed of AUTOROUTE.

SAVEBLK - Saves a block in a temporary storage space. This command is useful when moving blocks around a full chip.

SHOW LOCKED - Draws all locked blocks with very wide outlines.

SHOW PROHIBITED - Draws a large X through all prohibited blocks.

SHOW RATSNEST - Draws a line from the most recently programmed pip to all destinations of this net.

SHOWBLKCONN - Draws lines (in the World View) from all pins on a block to all other pins they connect to. This gives a good idea of how blocks should be placed relative to each other.

SHOWNETCONN - Draws lines (in the World View) from a net's source to all of its loads. This gives a good idea of where blocks should be placed to minimize net length.

SNAP - Places the cursor at the nearest pin or pip. You no longer have to line the cursor up exactly on a point. Put the cursor near the point and it will lock onto it automatically.

SPLITNET - Divides a net into two distinct nets, a quick way to split high fan-out nets. This reduces delay times and increases routability.

BON

Just Say NO to Asynchronous Design

Synchronous designs are safer than asynchronous designs, more predictable, easier to simulate and to debug. Asynchronous design methods may ruin your project, your career and your health, but some designers still insist on creating that seemingly simple, fast little asynchronous circuit.

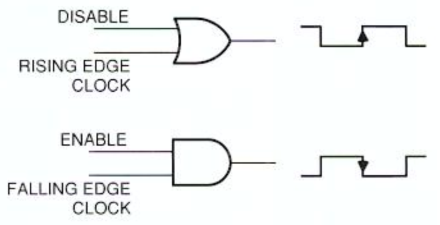
Twenty years ago, TTL-MSI circuits made synchronous design attractive and affordable; fifteen years ago, synchronous microprocessors took over many hardware designs; more recently, synchronous State Machines have become very popular, but some designers still feel the itch to play asynchronous tricks.

The recent popularity of ASICs has created a new flurry of asynchronous designs in a specially treacherous environment: Gate Arrays and Programmable Gate Arrays are being customized at the gate level, and may tempt the designer to develop bad asynchronous habits, especially dangerous since it is very difficult to inspect internal nodes, and impossible to calm them down with capacitive loads, the BandAid of simpler technologies.

Here is a short description of the ugly pitfalls in asynchronous design, documented for the benefit of the inexperienced designer. Veterans are familiar with the problems and may even know their way around them to design safe asynchronous circuits.

Clock Gating

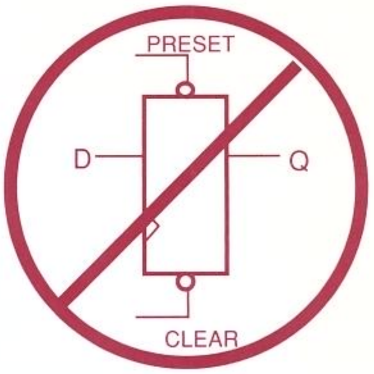
Gating a clock signal with an **asynchronous** enable or multiplex signal is an invitation to disaster. It will occasionally create clock pulses of marginal width, and will sometimes move the clock edge. A **synchronous** signal can be used to gate the clock reliably, as shown below, but this still introduces an additional clock delay, which can cause hold time problems.



Reliable *Synchronous* Clock Gating

Ripple Counters

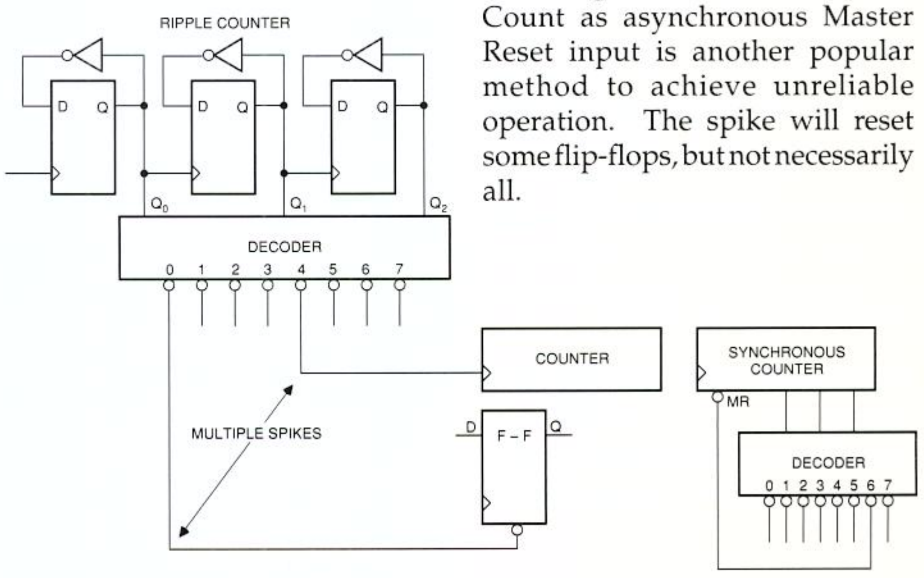
Using the output of one flip-flop to clock its neighbor can generate a binary counter of arbitrary length. The problem occurs when the counter increments from $2^n - 1$ to 2^n. It takes n delays from the incoming clock to the resulting change in bit n. In a 16-bit counter, this delay will be longer than 100 ns. At a 10 MHz clock rate, certain codes will never exist, the LSB will have changed before the MSB reached its new value. Decoding such a counter will produce dangerous decoding spikes. Note that these spikes are independent of the incoming clock rate. Designers of slow systems are actually most vulnerable to this problem, since they are less sensitive to delicate timing issues.



Decoders Driving Clocks and Reset Inputs

Indiscriminate use of decoder outputs to clock flip-flops or set/reset them asynchronously is one way to cause unpredictable and unreliable operation. The decoded outputs from synchronous counters are even more devious. While the decoding spikes from ripple counters are fairly wide and somewhat predictable, decoding spikes from synchronous counters are entirely the result of small but unpredictable differences in routing and decoding delays.

Using the decoded Terminal Count as asynchronous Master Reset input is another popular method to achieve unreliable operation. The spike will reset some flip-flops, but not necessarily all.



Unreliable Use of Decoders

Synchronizing One Input in Several Flip-Flops

A single asynchronous input should be synchronized in only one flip-flop. There will be an occasional extra metastable delay as described in the Xilinx 1989 Databook, pages 6-20,21. This extra delay is acceptable in all but the very fastest systems. Synchronizing one input in more than one flip-flop is another matter. The set-up times and input routing delays of the various flip-flops will inevitably differ by one or several nanoseconds. Any input change occuring during this time difference will be clocked differently into the individual flip-flops, and the error will last for a full clock period. Synchronize any input with only one flip-flop!

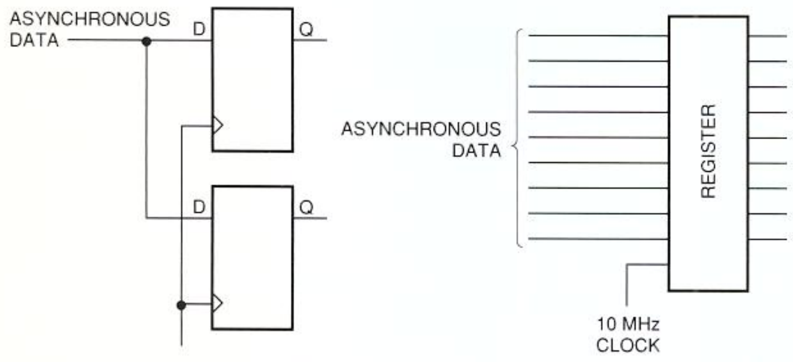
Synchronizing Multiple Inputs in One Register

Synchronizing an asynchronous parallel data word can lead to wrong results when the asynchronous inputs change during the register set-up time. For the duration of one clock period the register might then contain any imaginable mixture of old and new bit values. There is no simple solution, the most popular is to pipeline the result and compare the previous and present values. Any difference declares the data invalid. This operation is sometimes performed in software.

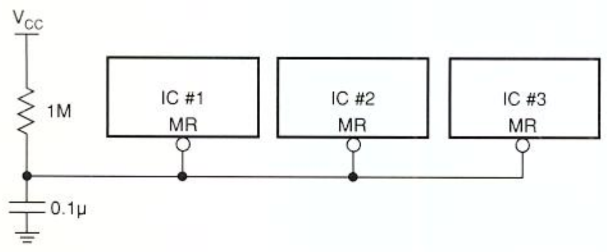
Asynchronous Reset of Multiple Circuits

A simple RC combination, perhaps augmented by a diode, is a popular power-on reset circuit. When it is used to drive several ICs in parallel, the system must accept wide variations in the reset duration.Differences in input threshold voltage will cause some circuits to start operating while others are still being held reset. If that is unacceptable, the RC combination must drive only one IC which, in turn, controls the reset operation of all others.

PA



Dangerous Methods of Synchronizing Asychronous Inputs



Asynchronous Reset of Multiple Circuits

Dot Your T's!

Schematic capture packages have an obsession about details. Some of them insist that a connecting dot be put on every T-joint, even on a connection to a bus. So, even if you think that it's redundant or ugly, put in the dots. It might avoid strange problems later on. One day in the future, we'll have true Artificial Intelligence, and computers will become our servants, not our masters. Until then, dot yourT's!

PA

Edit LCAs in the World View

Any command that XACT can execute in the full-size screen can also be performed in the World View. This feature is little known, since most users only show the World View while panning with the cursor.

The alternative is to always show the World View, but it takes up a large portion of the screen, and more often than not, just gets in the way while you're editing. The best solution would be to have the World View appear on command. This is very easy to do. With SHOW NOWORLD set, make the World View appear by pressing a mouse button and moving the cursor. Now stop moving the mouse, press the space bar, and release the mouse button. The World View will stay on the screen. You can now move or swap blocks across the LCA without having to pan around with the mouse trying to find a suitable location.

This feature will be very useful in XACT 3.0, which has new commands that display routing information in the World View.

BON

Alternate Routing Capability In ADI 2.21 Gives Shorter Propagation Delays In Small LCAs

The recent Xilinx DS23 development system update v2.21 contains an improved user-directed partitioner and two different routing programs. The released documentation and default program flow describe one router which automatically routes more nets in large arrays (3042, 3064, & 3090). Since routing is very design-dependent, it is difficult to achieve optimum results with only one router.

Thus an additional, undocumented router has been included in the software package. This router may result in better propagation delays while still offering nearly 100% autorouting for smaller arrays (2064, 2018, 3020). This router might also provide better propagation delays on larger arrays; it may, however, require interactive XACT graphical routing edits to complete the design.

This alternative router can be invoked if the default router (APR 2.24) achieves unsatisfactory results. CPU run time is short, since the placement is unaffected. The alternatively-routed design is stored under a different name, so the user can run report analysis to compare the propagation delays and select the best result.

PC SYSTEMS:

AROUTE is an exec file that initiates the alternative route program FROUTE 2.23f. The user can initiate the alternative routing sequence by typing

AROUTE name.lca newname.lca

Another routed lca file will be created with the user defined *newname* identification. The following files are created:

newname.lca	*alternatively routed, previously placed lca file*
newname.rpt	*net routing order & alternative routing propagation delays*
newname.out	*set of messages sent to user's screen during operation*

Several temporary files created in this process should be deleted:

xtempx.lca
xtempx.rpt
xtempx.out

PC Notes:

AROUTE may fail with the error message:

```
Error in 'NProgram' statement at line 318 in
file 'name'
```

The NProgram statements produced by APR2.24 may be incompatible with the alternative router APR v2.23f. These statements must be removed from the input LCA file before AROUTE will run successfully. (Removing these statements deletes the net routing and may require copying the file to preserve the original routing.)

Input command & output screen display for *successful* run of AROUTE.

```
    AROUTE filename.lca  newname.lca
AROUTE — Version 1.00
Copyright (C) 1989 Xilinx, Inc.  All rights reserved.
    Input Design File:   filename.lca
    SCP Constraints File:         (none)
    User Constraints File:        filename.cst
    Temporary Design File:        xtempx.lca
    Temporary Report File:        xtempx.rpt
    Temporary Message File:       xtempx.out
    Output Design File: newname.lca
    Report File: newname.rpt
    Message File:        newname.out
Routing input design...
Generating report file...
Done.
```

Input command & output screen display for *unsuccessful* run of AROUTE.

```
    AROUTE wrongname.lca  newname.lca
AROUTE — Version 1.00
Copyright (C) 1989 Xilinx, Inc.  All rights reserved.
    Input Design File:    wrongname.lca
    SCP Constraints File:         (none)
    User Constraints File:        (none)
    Temporary Design File:        xtempx.lca
    Temporary Report File:        xtempx.rpt
    Temporary Message File:       xtempx.out
    Output Design File: newname.lca
    Report File: newname.rpt
    Message File:         newname.out
Routing input design...
AROUTE: Error(s) detected.  See file 'aroute.out' and/or file 'xtempx.out'.
AROUTE: Execution terminated.
```

File aroute.out is empty and file xtempx.out contains the following:

```
AUTOMATIC PLACE AND ROUTE PROGRAM
Version 2.23f
Copyright (C) 1986,1987,1988 by Xilinx, Inc. All Rights Reserved.
Error: Can't open input design file 'wrongname.lca'.
FROUTE: Execution terminated.
```

Input command & output screen display for *interrupted* run of AROUTE.

```
    AROUTE filename.lca  newname.lca
AROUTE — Version 1.00
Copyright (C) 1989 Xilinx, Inc.  All rights reserved.
    Input Design File:    filename.lca
    SCP Constraints File:         (none)
    User Constraints File:        (none)
    Temporary Design File:        xtempx.lca
    Temporary Report File:        xtempx.rpt
    Temporary Message File:       xtempx.out
    Output Design File: newname.lca
    Report File: newname.rpt
    Message File:         newname.out
Routing input design...
User interrupts with CTRL-BREAK
AROUTE: Error(s) detected.  See file 'aroute.out' and/or file 'xtempx.out'.
AROUTE: Execution terminated.
```

File 'newname.out' is empty and file 'xtempx.out' contains the following:

```
AUTOMATIC PLACE AND ROUTE PROGRAM
Version 2.23f
Copyright (C) 1986,1987,1988 by Xilinx, Inc. All Rights Reserved.
    Input Design File:    filename.lca
    Schematic File:       (none)
    Constraints File:     filename.cst
    Options:     -l -w -r31861
    Output Design File: xtempx.lca
    Report File: xtempx.rpt
    Message File:         xtempx.out
Reading design...
  User interrupted execution with Ctrl-Break
FROUTE: Execution terminated.
```

APOLLO & SUN WORKSTATIONS:

The alternative router can be initiated after the APRv2.24 default placement and routing. Use the FROUTE command, followed by re-executing.APRv2.24 -JLP to prepare a report of the calculated propagation delays. (The AROUTE exec, which creates *xtempx* temporary files and renames the newly routed design and report files, is not available on the workstation version of ADI2.21). Alternative routing can be created with the following sequence of commands:

FROUTE -L name.lca newname.lca

creates:

newname.lca *contains new placement and routing*
newname.rpt *net routing order*
newname.out *set of messages sent to user's screen during operation*

APR -JLP newname.lca fname.lca

creates:

fname.lca = newname.lca *contains new place & routing*
fname.rpt *net routing order & alternate block/net propagation delays*
fname.out *set of messages sent to user's screen during operation*

FROUTE may fail with the error message:

```
Error in 'NProgram'
statement at line 318 in
file 'name'
```

The NProgram statements produced by APR2.24 may be incompatible with the alternative router. The problem NProgram statements must be removed from the input LCA file before FROUTE will run successfully. (Removing these statements deletes the net routing and may require copying the file to preserve the original routing).

DS

Helpful Hints for DASH-LCA Users

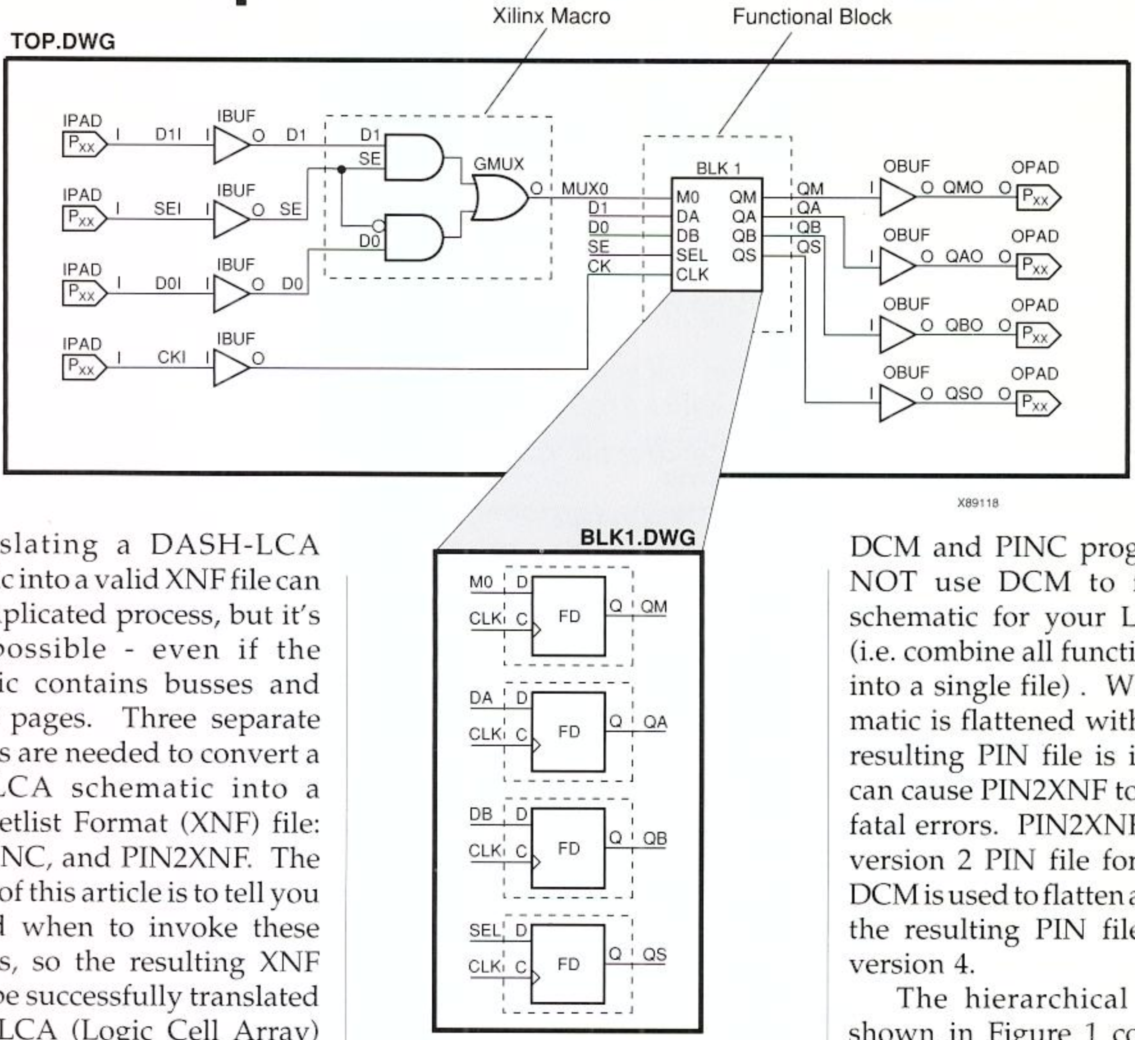


Figure 1

Translating a DASH-LCA schematic into a valid XNF file can be a complicated process, but it's not impossible - even if the schematic contains busses and multiple pages. Three separate programs are needed to convert a DASH-LCA schematic into a Xilinx Netlist Format (XNF) file: DCM, PINC, and PIN2XNF. The purpose of this article is to tell you how and when to invoke these programs, so the resulting XNF file can be successfully translated into an LCA (Logic Cell Array) file.

Don't be overwhelmed with all the steps in the DASH-LCA design translation process. The Xilinx Design Manager contains an automatic design translation program called XMAKE which invokes all the programs (including DCM, PINC, and PIN2XNF) needed to convert your design into an LCA file. All you have to do is push a button and watch XMAKE work!

For more information about DATA I/O's DCM and PINC programs, refer to Volume I of the FutureNet Schematic Designer Manual. Read the "FutureNet Interface" chapter in Volume II of the XACT Manuals, to learn more about PIN2XNF.

PROCESSING A DESIGN

A DASH-LCA schematic must be processed with DATA I/O's DCM and PINC programs before it can be converted to a Xilinx Netlist Format (XNF) file. The DCM pre-processor establishes connection data for a schematic drawing and outputs a binary DCM file. PINC, the DASH pin list generator, reads a DCM file and creates an ASCII pin list file (PIN file). Each PIN file contains a listing of the pins on all of the symbols in the schematic and the names of the signals connected to these pins.

You must create DCM and PIN files for each user-created drawing in your schematic by invoking the DCM and PINC programs. DO NOT use DCM to flatten the schematic for your LCA design (i.e. combine all functional blocks into a single file) . When a schematic is flattened with DCM, the resulting PIN file is invalid and can cause PIN2XNF to abort with fatal errors. PIN2XNF requires a version 2 PIN file format; when DCM is used to flatten a schematic, the resulting PIN file format is version 4.

The hierarchical schematic shown in Figure 1 contains one top level drawing (TOP.DWG), one functional block (BLK1), and one XILINX macro (GMUX). The drawing file for BLK1 is appropriately called BLK1.DWG, and is also shown in Figure 1. The following steps are required to create an XNF file for TOP:

1. Create a DCM file for the top level schematic:
   DCM TOP
2. Create a PIN file for TOP.DCM:
   PINC TOP
3. Create a DCM file for the functional block drawing:
   DCM BLK1
4. Create a PIN file for BLK1.DCM:
   PINC BLK1
5. Create an XNF file for the TOP design:
   PIN2XNF TOP

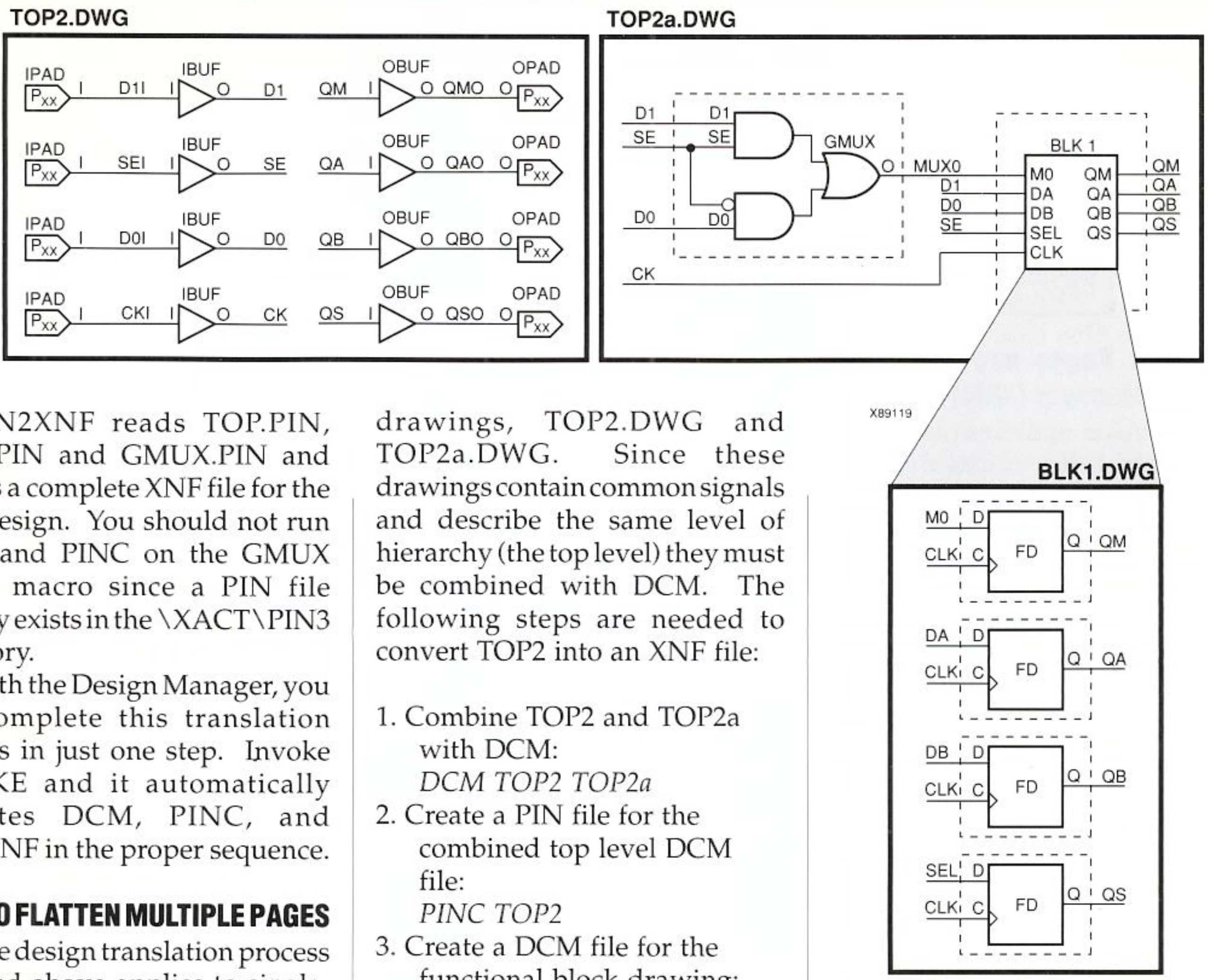


Figure 2

PIN2XNF reads TOP.PIN, BLK1.PIN and GMUX.PIN and creates a complete XNF file for the TOP design. You should not run DCM and PINC on the GMUX Xilinx macro since a PIN file already exists in the \XACT\PIN3 directory.

With the Design Manager, you can complete this translation process in just one step. Invoke XMAKE and it automatically executes DCM, PINC, and PIN2XNF in the proper sequence.

HOW TO FLATTEN MULTIPLE PAGES

The design translation process outlined above applies to single-page hierarchical schematics, i.e. schematics in which each level of hierarchy is described by a single drawing. The TOP design, for example, consists of a single-sheet top level drawing (TOP.DWG), and a functional block which is also described by a single-sheet drawing (BLK1.DWG). When a schematic contains multiple pages at the same hierarchical level, however, this translation process is modified. You MUST use DCM to combine multiple-page drawings into a single DCM file. Once these drawings are combined, the design translation processes for single-page and multiple-page drawings are identical.

Multiple Pages at Top Level

The TOP2 schematic shown in Figure 2 has two top level drawings, TOP2.DWG and TOP2a.DWG. Since these drawings contain common signals and describe the same level of hierarchy (the top level) they must be combined with DCM. The following steps are needed to convert TOP2 into an XNF file:

1. Combine TOP2 and TOP2a with DCM:
   DCM TOP2 TOP2a
2. Create a PIN file for the combined top level DCM file:
   PINC TOP2
3. Create a DCM file for the functional block drawing:
   DCM BLK1
4. Create a PIN file for BLK1.DCM:
   PINC BLK1
5. Create an XNF file for the TOP2 design:
   PIN2XNF TOP2

Multiple Pages for a Functional Block

When a functional block is described by multiple drawings, these drawings must be combined with DCM, and the name of this combined file should appear in the functional block symbol with an attribute 8 (file attribute). The functional block MUST contain only one file name (attribute 8).

The functional block in TOP3 (Figure 3), for example, is described by two drawings, BLK1 and BLK1a. These drawings are combined with DCM and the resulting file name (BLK1) appears inside the functional block on TOP3a.DWG. The design translation process for TOP3 is completed with the following commands:

1. Combine TOP3 and TOP3a with DCM:
   DCM TOP3 TOP3a
2. Create a PIN file for the combined top level DCM file:
   PINC TOP3
3. Combine BLK1 and BLK1a with DCM:
   DCM BLK1 BLK1a
4. Create a PIN file for the combined DCM file:
   PINC BLK1
5. Create an XNF file for the TOP3 design:
   PIN2XNF TOP3

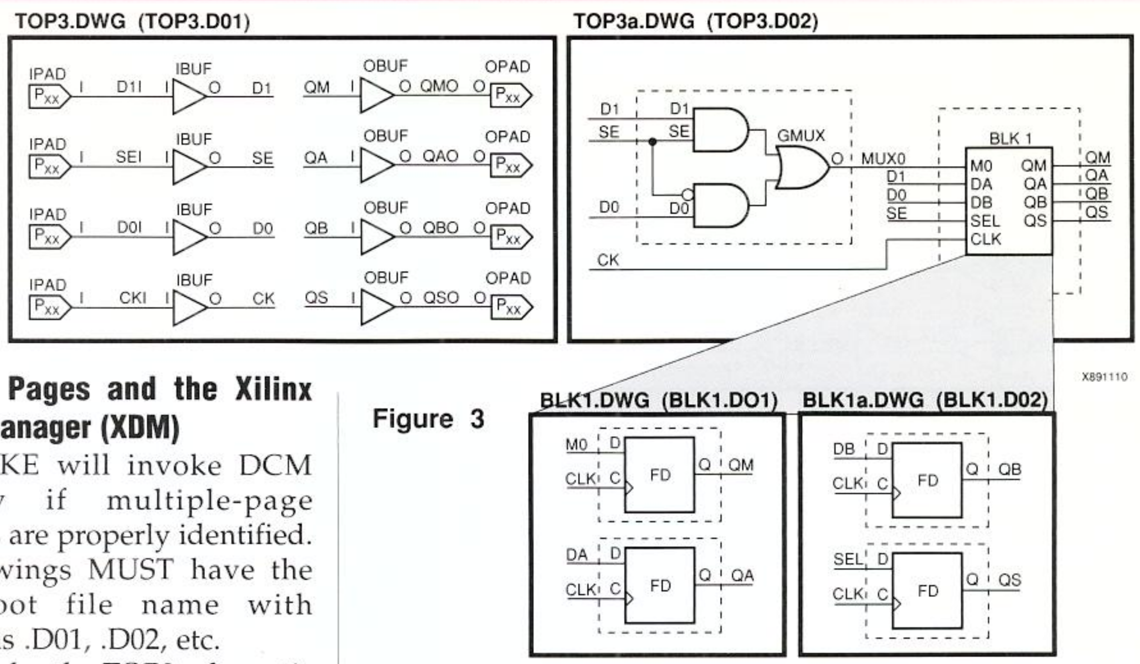


Figure 3

Multiple Pages and the Xilinx Design Manager (XDM)

XMAKE will invoke DCM correctly if multiple-page drawings are properly identified. The drawings MUST have the same root file name with extensions .D01, .D02, etc.

Consider the TOP3 schematic shown in Figure 3. Before the design is translated, rename TOP3.DWG to TOP3.D01 and TOP3a.DWG to TOP3.D02. Also rename BLK1.DWG to BLK1.D01 and BLK1a.DWG to BLK1.D02. XMAKE will recognize that TOP3 and BLK1 contain multiple pages and will invoke DCM with the following command lines:

DCM TOP3.D01 TOP3.D02
DCM BLK1.D01 BLK1.D02

The resulting DCM files will be called TOP3.DCM and BLK1.DCM; TOP3.DCM will contain logic from both TOP3 schematic pages, and BLK1.DCM will contain logic from both BLK1 schematic pages.

Attributes

Each text string on a DASH-LCA schematic has an associated attribute which provides information needed by post-processors. A signal name, for example, has attribute 5 (signal attribute), and a comment has attribute 0. Many attributes are available with DASH-LCA, but only a small subset can be used on an LCA schematic. Those attributes that are interpreted correctly by PIN2XNF are:

#	Definition
0	Comment
2	Location-assigns symbol name
3	Part-defines symbol type This attribute appears on Xilinx primitives and is used to define symbol types. **You must not create text with this attribute.**
5	Signal
8	File This attribute is assigned to the file name called out inside a functional block.
21	PINO - Input pin
22	PNBT - Bi-Directional pin
23	PINI - Output pin
80	PIN Numbers Defines IOB and CLB locations
81	LCA Part Type
82	XACT Config. Statements - used on IOB & CLB primitives
83	Special Options - "FILE=", BLKNM=, FAST I/O, DOUBLE pullups
84	CLBMAP

Use only these attributes for an LCA design; other attributes might not be interpreted correctly by PIN2XNF. Refer to the "FutureNet Interface" chapter in Volume II of the XACT manuals for more information.

Everything You Ever Wanted to Know About Using Busses in DASH-LCA

In DASH-LCA, busses are drawn with "type 2" (/2) lines, and bus names have attribute 5. Individual bus signal names also have attribute 5, and can be connected directly to the bus; no connection dots are needed.

When a bus connects directly to a functional block, a special bus pin is needed. This pin is drawn with the ".=" command in DASH-LCA and must be assigned a pin name, just like any other functional block pin. The pin can have any name (it does not have to match the name of the attached bus) and should have attribute 23 (PINI), 21 (PINO), or 22 (PNBT). An example schematic which contains a bus connected to a functional block is shown in Figure 4.

The functional block in Fig. 4 represents the logic in a drawing called BLK.DWG. Since the functional block contains a bus pin called MYBUS, the BLK schematic must contain a bus called MYBUS. **The bus signals which make up MYBUS must have the same names as the signals which make up TOPBUS on the top level schematic.**

In the BLK schematic , MYBUS contains three signals, 0,1,and 2, which match the signal names on TOPBUS. When the TOP4 schematic is translated, MYBUS and TOPBUS are connected and the resulting bus is called TOPBUS.

When DCM is run on the TOP4 schematic, the following error message appears:

```
Warning Bus pin 'MYBUS'
terminates at symbol 11 in
file 'TOP4.DWG'
```

Ignore this message; PIN2XNF will properly translate the drawing.

New Bus Naming Convention

You must follow a new bus naming convention to insure that PIN2XNF v2.31 correctly processes bus signals which connect directly to functional blocks. A bus name (attribute 5) now MUST include the size of the bus and the individual bus signal names MUST be consecutive numbers. The bus in TOP4.DWG, for example, is named TOPBUS<0:2> and the signals are called 0, 1, and 2. The numbering sequence does not have to start with 0, but the smallest number must be listed first in the bus name. (The bus in TOP4.DWG could be named TOPBUS<1:3> with signals 1, 2, and 3; TOPBUS<3:1>, however, is invalid).

Limitation of PIN2XNF V. 2.21

If you are using PIN2XNF v.2.21 (or earlier), you cannot connect a bus directly to a functional block which contains a "FILE=" parameter.

PIN2XNF 2.31, which is currently shipping and is available on the bulletin board, corrects this problem.

Do Not Use PIN and POUT Macros

We recommend that you DO NOT use the PIN and POUT macros provided with the DASH-LCA library. When these macros are used, the signal between the pad and the buffer is not named. This signal should have a meaningful name, since it determines the name of the IOB in the LCA design file. You can name the net which connects the pad and buffer if you separately invoke the pad and buffer symbols.

CAL

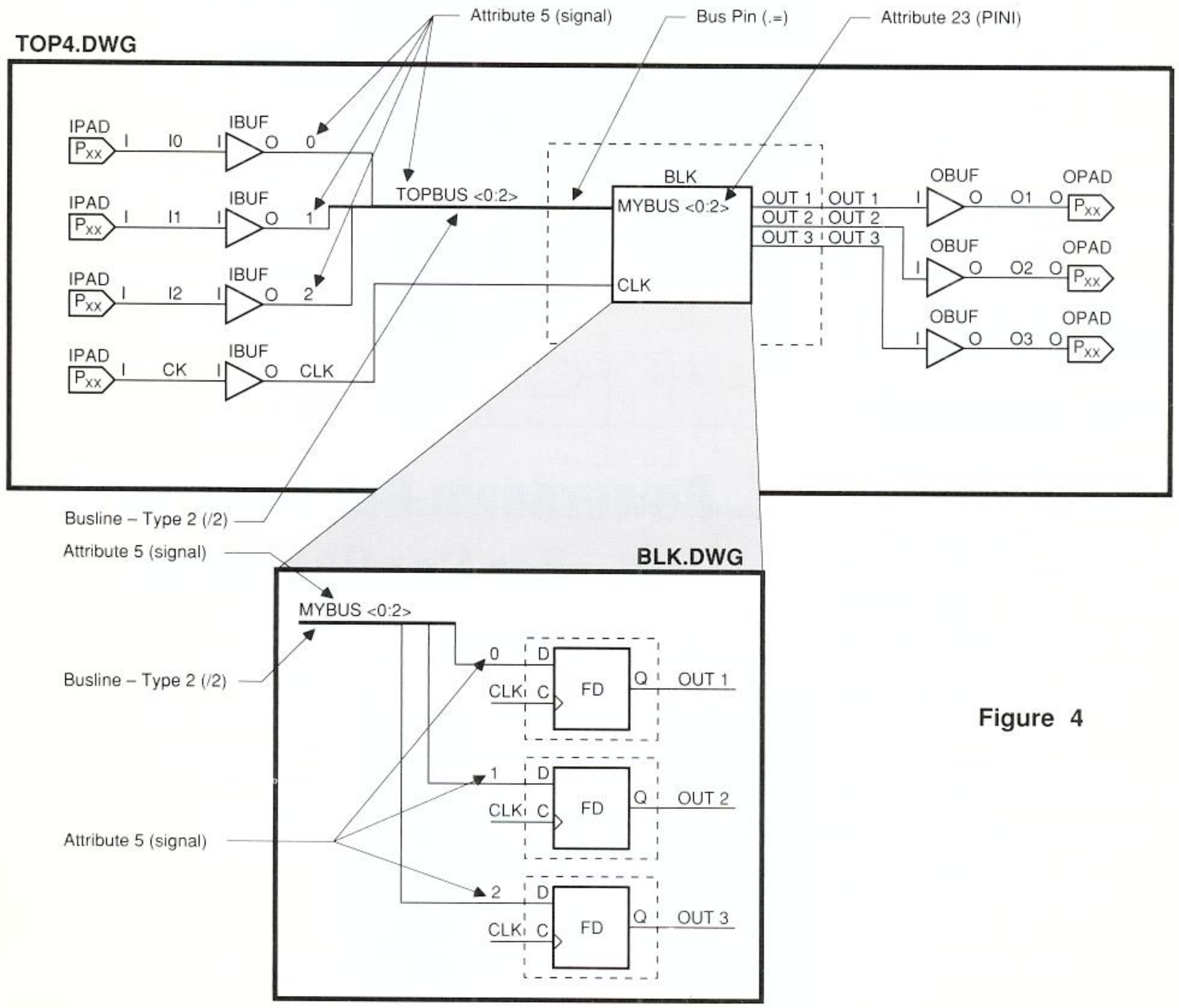


Figure 4

Design Security

Some Xilinx customers are concerned about design security. How can they prevent their designs from being copied or reverse-engineered?

There are two different methods of protection:

1. Copyrights

Copyrights provide legal protection of the configuration data stored in the serial or parallel EPROM or in a microprocessor's memory, just as they protect microcode and computer programs. The LCA bitstream can be protected by copyright laws that recently have been enforced more successfully than the intellectual property rights of circuit designs.

While it is easy, albeit illegal, to make an identical copy of the LCA configuration data, it is virtually impossible to use the bitstream in order to understand the design or make modifications to it. **Xilinx keeps the interpretation of the bitstream a closely guarded secret.**

Reverse-engineering an LCA would require an enormously tedious analysis of each individual configuration bit, which would still only generate an XACT view of the LCA, not a usable schematic.

The combination of copyright protection and the almost insurmountable difficulty of creating a non-identical design for the intended function provides good LCA design security.

The recent successes of small companies in reverse-engineering microprocessor support circuits show that a non-programmed device is actually more vulnerable than an LCA.

2.Battery-Back-Up

Some designs do not permanently store a source of configuration data. After the LCA was configured, the EPROM or other source was removed from the system, and configuration data is kept alive in the LCA through battery-back-up.

If the design does not contain the source of configuration data, then there is no conceivable way of copying this design. Opening up the package and probing thousands of latches in undocumented positions to read out their data without ever disturbing their content is impossible.

This mode of operation offers the **ultimate design security.** It is being used in applications where the design must be kept a secret, for example in cryptographic applications that might involve national security.

PA

Fixed Modulus Counters and Dividers

Many designs specify a loadable counter when the value to be loaded is actually a constant. Such a design can be greatly simplified. The most obvious and most general solution uses a variation of the "30-MHz Binary Counter with Synchronous Reset" from page 6-36 of the 1989 Xilinx Databook. The counter is set to any arbitrary value by changing the RESET to a PRESET on the appropriate bits. The maximum speed calculation has to take in consideration that the time between successive CEP signals may be shorter immediately after a preset of one or more of the least significant bits.

A faster solution uses a variation on the "40-MHz Presettable Counter" described on page 6-38 of the '89 Databook. Each di-bit uses only two CLBs, as opposed to three in the original version that is loadable with variable data.

PA

Powerdown Pin Must Be High For Configuration

A Low on the $\overline{\text{PWRDWN}}$ pin puts the LCA to sleep with a very low power consumption, typically less than one microwatt. The on-chip oscillator is stopped, and the low-Vcc detector is disabled.

During configuration time, $\overline{\text{PWRDWN}}$ must be High, since configuration uses the internal oscillator.

Whenever Vcc goes below 3V, $\overline{\text{PWRDWN}}$ must already be Low in order to prevent automatic re-configuration at low Vcc. For the same reason, Vcc must first be restored to 3V or more, before $\overline{\text{PWRDWN}}$ can be made High.

PA

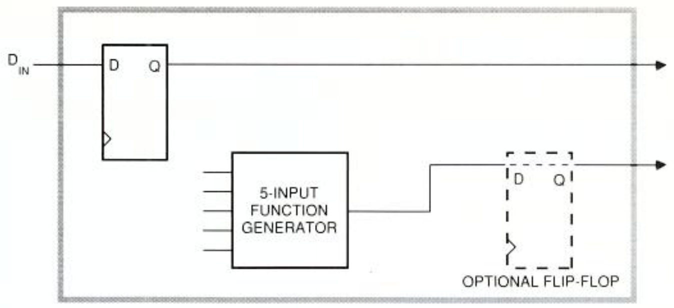
Avoid the Terrible DIN

DIN may seem to be a simple way to drive the D input of a CLB flip flop without going through a function generator. There are, however, several drawbacks. DIN cannot be swapped with any other pin, and DIN can only be driven from four different sources (pips). The user should restrict the use of DIN to those few situations where it really improves the design. There are only three cases, as shown in the figure at right..

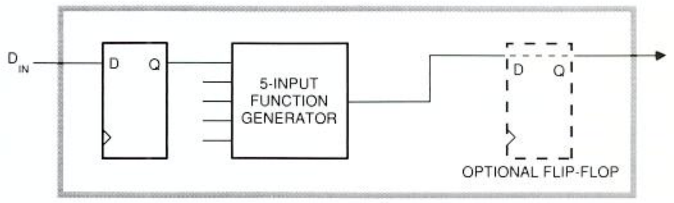
In all other cases, DIN should be substituted by a function generator input.

XNFMAP and XNF2LCA now have a -D option that never uses DIN.

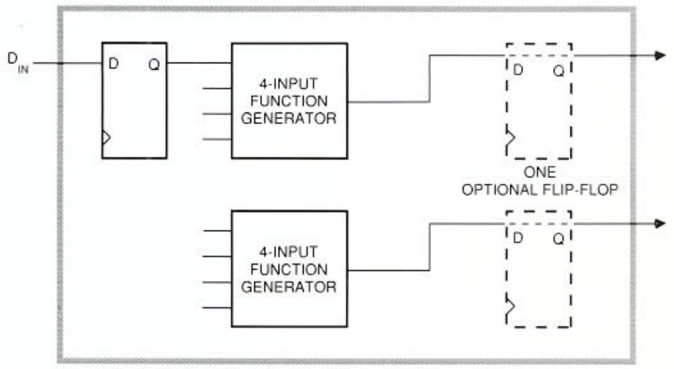
PA



1. A CLB Flip-Flop Independent of the 5-Input Function Generator in the Same CLB



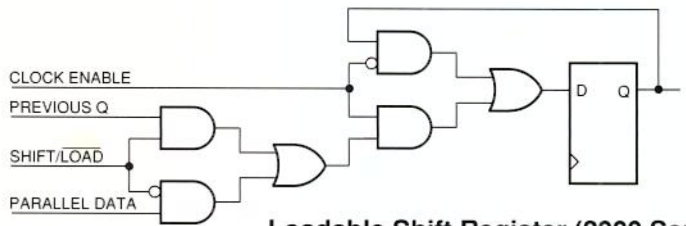
2. A CLB Flip-Flop Driving the 5-Input Function Generator in the Same CLB



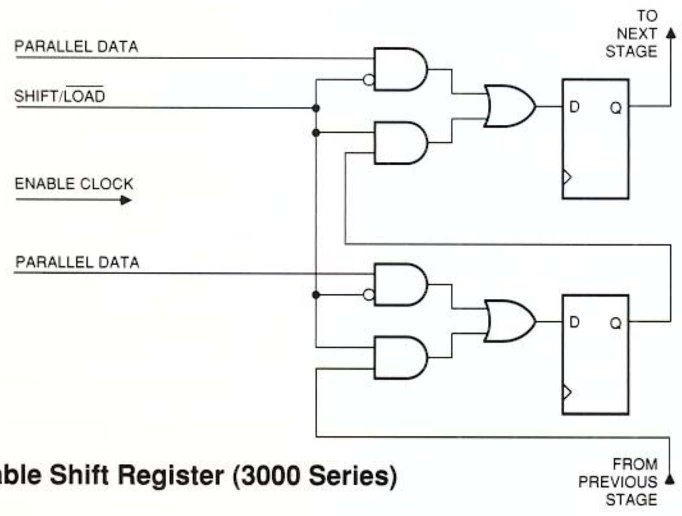
3. A CLB Flip-Flop Driving One 4-Input Function Generator in the Same CLB

X89113

Loadable Shift Register with Clock Enable



Loadable Shift Register (2000 Series)



Loadable Shift Register (3000 Series)

The 2000 Series CLB primitive shown at left is a building block for a shift register with synchronous load and clock enable, or for a bidirectional shift register with clock enable but without parallel load.

The 3000 Series CLB primitive shown at left is a 2-bit building block for a shift register with synchronous load and clock enable, or for a bidirectional shift register with clock enable but without parallel load.

PA

Magnitude Comparator: Small, Fast, Expandable

A magnitude comparator is more complex than an identity comparator, but simpler than an adder or subtractor. A magnitude comparator indicates not only when two operands are equal, but also which one is greater if they are unequal.

PA

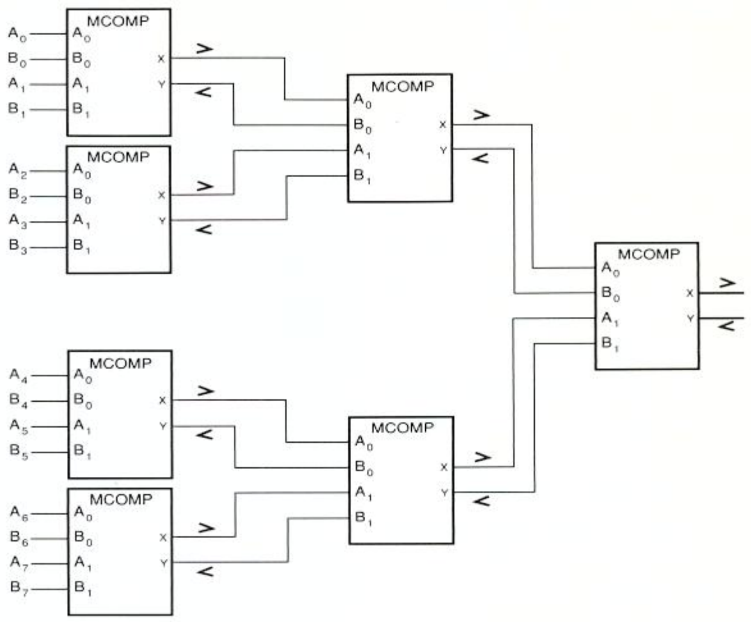
Truth Table

B1	A1	B0	A0	A>B	A<B
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	1	0	0

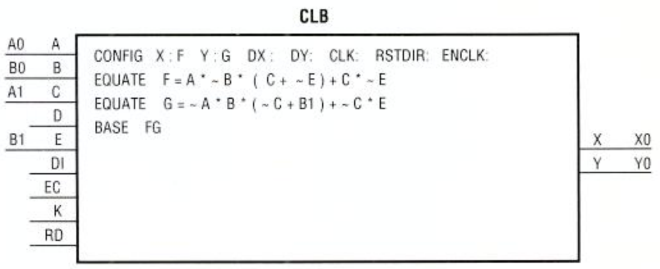
This truth table is represented by the following equations:

A>B:=A0*~B0*(A1+~B1)+A1*~B1

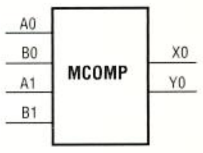
A<B:=~A0*B0*(~A1+B1)+~A1*B1



Magnitude Comparator Expands to Any Size



```
CONFIG  X:F  Y:G  DX:  DY:  CLK:  RSTDIR:  ENCLK:
EQUATE  F=A*~B*(C+~E)+C*~E
EQUATE  G=~A*B*(~C+B1)+~C*E
BASE  FG
```



XILINX Survived the Earthquake

As you undoubtably know, an earthquake measuring 7.1 on the Richter scale struck the San Francisco Bay Area on Tuesday, Oct. 17th shortly after 5 PM. Fortunately, no Xilinx employees were injured. Damage to Xilinx headquarters was very slight - just a few broken ceiling tiles. All our computers and test equipment functioned properly when power was restored, and we returned to "business as usual" on Thursday, less than 48 hours later.

Some of our employees suffered property losses and many had to cope with agonizing traffic delays, but we feel very fortunate that nobody in the "Xilinx family" was among the casualties.

We would like to thank all our friends and customers that called to express their concern.

BKF

1990 Training Course Schedule

The Xilinx Programmable Gate Array Training Course is a comprehensive class covering the Logic Cell Array component architecture and Xilinx development systems, with emphasis on the XC3000 family. This course is intended for design engineers using Xilinx LCAs in their applications who want to get "up to speed" as quickly as possible.

Courses run for a full four days, Monday through Thursday. A substantial amount of the class time is spent performing lab exercises on the Xilinx development system. The development systems are available to attendees on the Friday following the class for optional individual work and consultations with Xilinx application engineers.

The tuition fee is $850 per student. The courses are held at Xilinx headquarters in San Jose, CA. To enroll, call Cindi Potts, Training Administrator, at (408) 559-7778. Class size is limited, so early enrollment is recommended.

If a number of engineers at your company need training, courses can be arranged at your company's site. For more information about on-site courses, contact Brad Fawcett at (408) 879-5097.

The following courses are scheduled for the first half of 1990; additional courses will be added to this schedule if dictated by demand:

Jan. 22 - 26
Feb. 26 - Mar. 2
Apr. 2 - 6
Apr. 30 - May 4
June 4 - 8

BKF

Try Our Voicemail

If you want to leave a message for somebody at Xilinx in San Jose, and you know his or her 4-digit extention code (the trailing 4-digits of the direct dial number that starts with 408-879-....), then you can just dial **408-879-5191**, wait for the response "Express Messaging". Then you key in (Touch-tone only!) the 4-digit extention which always starts with a 5. **You must follow this with a trailing #.** This connects you to the employee's voice mailbox. You can leave a message, and a sign will start blinking on the employee's phone.

It's that easy!

PA

Training Course Outline

Day 1:
BASIC OF THE LCA ARCHITECTURE
- XC3000 Family Architecture
- CLB Structure
- IOB Structure
- Basics of Interconnect
- Dedicated Pins

DESIGN METHODOLOGY OVERVIEW
DESIGN ENTRY
- Schematic Capture
- PALASM to XNF Translator
- Logic Optimization

Day 2:
DESIGN IMPLEMENTATION
- Logic Partitioning
- Automatic Place and Route
- Configuration Bit Stream Generation
- Bit Stream Format
- PROM Formatter

LCA CONFIGURATION LOADING MODES
DESIGN IMPLEMENTATION ADVANCED TOPICS

Day 3:
DESIGN VERIFICATION
- Simulation
- XACT Download Cable
- XACTOR2 In-Circuit Debugger
- Readback

Day 4:
XACT DESIGN EDITOR
LCA ARCHITECTURE
- XC3000 Interconnect Details

SUMMARY
- Estimating Size and Performance
- Benefits of PGA Technology

Xilinx Mourns Passing Of Founder

When Ross Freeman, Vice President of Engineering and a co-founder of Xilinx, died on Oct. 22nd, we lost a leader and a friend.

Ross invented the first programmable gate array in 1984 and headed the design team that developed products based on that concept. The programmable gate array had been recognized for years as a major opportunity, and many designers at a number of companies had tried to find a solution. Where all had failed before, Ross succeeded by finding an unconventional solution: the use of static memory cells as the programming element. Three years after the introduction of Ross' brainchild, market research firm Dataquest Inc. agreed that this revolutionary new technology was the basis for a whole new logic market.

Of course, it took much more than technical talent to translate a product idea into a successful company. Ross had a powerful dream for a new way of designing digital logic, and his enthusiasm for realizing this dream was every bit as strong and as important as his technical skills. We remember how Ross would hold customers, investors, and potential employees spellbound as he described the potential of the Logic Cell Array.

More importantly, and unlike the stereotype of the successful technocrat, Ross always placed the welfare of his co-workers and employees over all other considerations. As noted in an article in the April 1988, issue of VLSI Systems Design, Ross' primary goal was to "keep Xilinx a nice place to work." The article goes on to describe Ross as "a team player, serious about hard work, company pride, and the close knot of employees he calls 'the tribe'... [Ross is] a man with broad ideals and high aspirations."

Prior to founding Xilinx in 1984, Ross was Director of Engineering for Zilog's Component Division. Prior to that, he was an IC design engineer at Teletype Corp., and had spent two years with the Peace Corps teaching mathematics in Ghana.

As a legacy, Ross leaves behind a thriving company and a strong design team that will continue the efforts that he pioneered.

XILINX

2100 Logic Drive
San Jose, CA 95124-3450

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