XCELL

THE NEWSLETTER FOR XILINX PROGRAMMABLE GATE ARRAY USERS

Issue 5 First Quarter 1990

This issue is dominated by a center spread of detailed pin-outs for our PGA parts and for PLCC sockets. We hope that they will save you time and avoid errors when you lay out and debug your PC boards.

We heard about two different prototyping boards, and include a short description as well as the addresses of several socket vendors.

You will find the usual small design ideas and device explanations, but the most important part of this issue is the ***"Intelligent User's Guide to APR"*** *and the glimpse of the new release, APR 3.0. Over the past years we had taken some criticism about routing problems with big designs. It is, therefore, with relief, pride, and joy that we tell you about the new ADI 3.0. Everything you had wanted all the time: Smarter partitioning, placement, and routing, and significantly faster circuits. You will love it!*

Peter Alfke, Editor

Table of Contents

Faster Designs, Routed Automatically with ADI 3.0

Xilinx is about to introduce an ADI upgrade, version 3.0, to be shipped in April free of charge to all registered users who bought ADI during the prior year or have a current update contract. ADI 3.0 offers a significant improvement in two areas:

Routing of complex and dense designs is now more automatic, and circuit speed is substantially improved.

We know, for we have been using this new software in-house during the de-bugging, QA and early production phases. We have run 30 demanding designs, 20 of them complex 3090s, and we found the following improvements:

80% of these designs routed completely on the first attempt, compared to 10% with the present ADI 2.21. Average propagation delays, using the same constraint files, were 35% shorter, due to more intelligent partitioning, better placement and smarter routing. In some designs, the speed had more than doubled.

Over the years, we have learned that no single routing algorithm is best suited for all situations. For this reason, APR 3.0 will include three different router options, as well as an explanation of when to use which one.

There is also a new utility, called XNFDRC, which gives explanatory warnings about possible design flaws before the routing process begins. This will save time and aggravation.

Run-time per pass is essentially unchanged, the smarter algorithms are inherently faster, while the more exhaustive analysis adds to the execution time. But there is rarely any need for a second or third run, and no requirement for extensive "pip-poking".

We are enthusiastic about these new tools. If a design fits and can be routed, ADI 3.0 will route it for you. Even complex 3090 designs have been partitioned, placed, and routed in a day. And the performance of your design will be significantly enhanced.

PA

Component Availability (February 1990)

		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 February 1, 1989.

DS112 ENHANCED PROGRAMMER FOR SERIAL CONFIGURATION PROMS
XPP program ver. 3.00

DS21 XACT 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. 1.04 (ver. 3.1 on request)

DS34 MENTOR INTERFACE ver. 2.21
Mentor IDEA interface&library ver. 2.21

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

DS 51 SCHEMA II+, ADI, & XACT ver. 2.31
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 +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. 2.31
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)

The Intelligent User's Guide To APR

When running the Automatic Placement and Routing program, many users type:

```
apr input output
```

and let APR run blind. However, there are many things that can be done to increase performance, to make APR run faster and give better results.

Routability is influenced by part size, design size, speed requirements, and the type of function to be implemented. Usually, these factors can't be changed. But there are other actions you can take to make your design more routable:

Before Running APR

1) Don't lock IO blocks

It may be very tempting to lock the IO and immediately lay out your PC board. This might gain you some time, but if you lock the IOBs in bad locations and can't get the design routed or can't achieve proper performance, then your PC board will be worthless.

It is always best to let APR place the IO blocks.

2) Use GCLK and ACLK

If you have a clock signal, use one of the clock buffers. One long line in each column is reserved for GCLK, the global clock buffer. If you don't use it, you waste these long lines and you'll get inferior performance by having to route a clock signal through general interconnect. The alternate clock buffer ACLK does not have any interconnect specifically reserved for its use, but it does have long line connections to clock pins on all logic elements.

Use the GCLK buffer for your largest or highest frequency clock net.

Use the ACLK for the next most critical clock in the design.

3) Be aware of net fan-out

High fan-out nets are harder to route and have longer net delays. Nets with more than 16 loads (except clock or 3-state control signals) should be split up by using duplicate logic to create two nets for one signal.

4) Don't overuse constraints

Constraints play a role in both the placement and routing phases of APR. During the placement phase, the APR cost calculator multiplies a net's length by its weight to determine the cost. The net's weight statement also determines the routing order: A net with higher weight is always routed before any lower weight net.

Constraint flags and constraint files may cause APR to place blocks in non-optimal positions. Flagging large sets of nets, or assigning many high net weights may hurt more than help. APR will clump large groups of marginally related logic together, which will lead to poor routability and longer delay times.

Whenever possible, avoid using constraint flags to influence placement.

5) Never lock placement

Don't lock CLB primitives or CLBMAPs in place, and don't lock pins on CLBMAPS. In general, the more freedom APR has, the better the result.

6) Use Map-then-Merge

This ensures that unrelated logic will not be mapped together into the same CLB. This improves placement and decreases routing congestion.

Refer to the Application Note "Advanced Design Methodology"

7) TBUFs vs. Logic

Use a long line as wired-AND or multiplexer only if the number of inputs exceeds three. In XC3042 and larger the number should exceed five.

8) Avoid the DI pin

DI has fewer interconnection points (4) than the other CLB pins, and it cannot be swapped with any other pin on the CLB. It's better to lose a few nano-seconds going through a function generator and keep the flexibility that pin swapping gives you.

Use the -D option in XNFMAP to avoid DI .

9) Avoid the CE pin

Like DI, the CE pin has only four interconnection points, making it hard to route. Wherever possible use logic to perform the clock enable function.

continued on page 4

Running APR

As its name suggests, the automatic placement and routing program consists of two distinct phases: block **placement** and net **routing.**

The table at right shows the options that can be used to influence the way APR runs.

Hints for using command line options:

- You may end any phase of APR by hitting CTRL-BREAK on your keyboard. After at least one routing iteration has been completed, a CTRL-BREAK will exit the program and leave you with a routed .lca file, but with no .rpt file.

- If you have made changes to a design in XACT, or quit APR before a report file has been generated, you can make a report file by typing:

```
apr -jlp input output
```

- APR spends the majority of its time in the placement phase. However, once a suitable placement has been obtained, this phase can be skipped for all further APR runs. This is useful if a design does not get completely routed. You can make changes to a constraint file and run only the routing algorithm by typing:

```
  apr -l input output
```

- A quick way to unroute a design is to type:

```
 apr -jl input output.
```

APR Options

-a	Set the number of routing iterations. 13 is the largest meaningful number, but 5 is more practical. If it isn't routed after the fifth pass, it probably won't route
-b	Set the beginning temperature. It is best to let APR select the beginning temperature.
-c	Use a constraint file. Constraints can be used to assign locations for CLBs and IOBs, to weight nets, to lock blocks in their current position, or to prohibit the use of certain locations. Constraint files can have a negative impact on placement and routing, so they should be used with caution.
-e	Set the ending temperature. It is best to let APR select the ending temperature.
-g	Use a guide file. This is very useful for iterative design and map-then-merge techniques. A design can be generated and routed incrementally.
-j	Just place blocks. Stops before the routing phase.
-k	Explicitly set a percent change in temperature. 40% is useful for quickly placing a sparse design, where placement is not a determining factor in routability. 5% may achieve best placement, but uses far more time.
-l	Lock all blocks, just route, skip the placement phase.
-o	Redirect screen output to a file. Useful on workstations, don't use it on a PC.
-p	Keep existing routing. Maintains pre-routing done in XACT.
-q	Quench only. Skips the simulated annealing algorithm. This makes APR run faster, but may not achieve the best placement.
-r	Set the seed. (This will be called -s in the future ADI 3.0) The seed is a random number, no seed is better than any other. Use -r only in order to reproduce a specific run.

Routing Methodology

If you have a sparse design or a slow design, typing:

```
apr input output
```

should always give a routed design. The following section describes a design methodology that can be used for very full LCAs or for LCAs with tight timing requirements.

For the first APR run, you should use no .cst or .scp files, and you should use the default number of routing attempts. At this point we are less interested in getting a complete route, but we want to quickly get a good placement to be used for all future APR iterations.

After APR has finished, it will tell you how many pins are left unrouted. Check the report file and use XACT to see if your timing specifications were met.

- If you have less than five unrouted nets and the timing looks good, the quickest solution is to use XACT and manually route the pins. But if you'd rather not use XACT, go on to the next step.
- If you have less than ten nets unrouted, or if the net delays are too long, make a constraint file (.cst) and weight a few of the nets that are unrouted or have long delays. Re-run APR with the -l and -c options.
- Any time you have more than ten unrouted nets, poor placement could be the cause. You should go back and re-run APR a few times, looking for a better placement. If this doesn't help, then it's time to do some pre-routing in XACT and perhaps even some manual placement.

Print out a copy of the report file and use a marker to highlight the following items in the report file:

- All nets with a fan-out of more than eight loads.
- All nets with excessive delays
- All clock nets that are not on GCLK or ACLK.
- All CE pins that are used.
- All 3-state control pins (TBUF.XX.T).

These are the signals that cause the most routing problems.

In XACT, highlight one of the high fan-out nets that was marked on the report file. Without moving the blocks too far from their original position, try to align them in a column. Use the SWAPSIG command to put the net onto the same pin on all the blocks, so that a longline can be used.

Clock pins that are not on the GCLK or ACLK buffer should be pre-routed (routed before you run APR) since they are otherwise hard to reach.

CE pins and 3-state control pins contend for the same longline. If a column has TBUFs in it and the CLBs to the right use the CE pin, decide which one should get the longline, and pre-route it or move the CLBs.

Now run APR with the -p and the -l options. If you think you might have disturbed the placement, use only the -p option. This will take longer, but ultimately give a better placement. Constraint files can also be used to affect the routing order.

ADI 3.0 Preview

The new update, ADI 3.0, will simplify the routing process considerably:

The XNFDRC program gives warnings if a net has too many loads, if clock buffers are unused, or if there are too many or too few TBUFs on a long line.

The partitioner defaults to not using the DI pin. This saves you the trouble of specifying that command line option.

The new version of XNFMAP does a much better job of mapping logic optimally into CLBs. This reduces the need for map-then-merge, and it increases routability.

Each of the three new router options is significantly better than the current router.The default option will route most designs automatically.

In ADI 3.0, critical flags and longline flags have a greater effect on the design. This reduces the need for pre-routing.

All these factors combine to make the translation from schematics to working LCA a far more automated process.

And, most importantly:
Your routed design will have much shorter delays than you could achieve in the past.

BON

Programming a Set of 1736s

Multiple 1736s can be programmed with sequential data from a single file, using a programming module with a single socket. This operation is referred to as Serial Set Programming.

For UNISITE:

Select Xilinx 1736 using the **Select Device** screen.

Using the **Transfer Data** screen, download the desired file, which will contain more data than can fit in a single device.

In the **Program Device** screen, select the **all parameters** mode by pressing F4, if not in this mode already (this is a toggle function).

Set **Total Set Size** to the number of devices to be programmed with sequential data.

Set **Auto-increment** to **Y**.

Insert the first 1736 in the DIP socket on the left side module and hit Return.

When the first device has been programmed, **Next Device** will increment and the programmer will wait for the next device to be inserted. Insert it and hit Return. Continue this procedure until all devices in the set have been programmed.

For UNIPAK:

The UniPak allocates 8192 (2000HEX) bytes for each device, even though the 1736 holds only 4536 (11B8HEX) bytes.

To program two devices with one file, first download the whole file, then move the data starting in location 11B8 to location 2000; this can be done using Block Move. Insert the first device and program it starting at RAM location 0. When this is complete, insert the next device and program it starting at RAM location 2000.

This information was provided by Data I/O.

PDS2XNF Supports PALASM2 Equation Format Only

Although PALASM2 has both a Boolean Equation and a State Machine entry format, the PALASM2 to Xilinx Netlist Format (XNF) translator, PDS2XNF, only translates designs expressed in the **Boolean Equation format**. If you want to enter Xilinx designs in a **State Machine format**, you must use other programs, like ABEL or CUPL. These programs are available from Data I/O (800-247-5700) and Logical Devices (800-331-7766) respectively, not from Xilinx. ABEL version 3.11 or later and CUPL version 3.2 or later can output a PALASM2 format file which can be used as an input to the Xilinx PDS2XNF program. With these programs you can incorporate State Machines as well as Boolean Equations into schematics or convert PLD designs directly into LCAs.

TCW

Save $ 3800 on XACTOR

Lasting through June 90, Xilinx offers a special promotion:

A DS28 XACTOR controller for only $1650 instead of the regular price of $5450.

This means the whole XACTOR package, including DS26 pod and DS27 header goes for between $2500 and $3100, instead of the list price of $6300 to $6900.

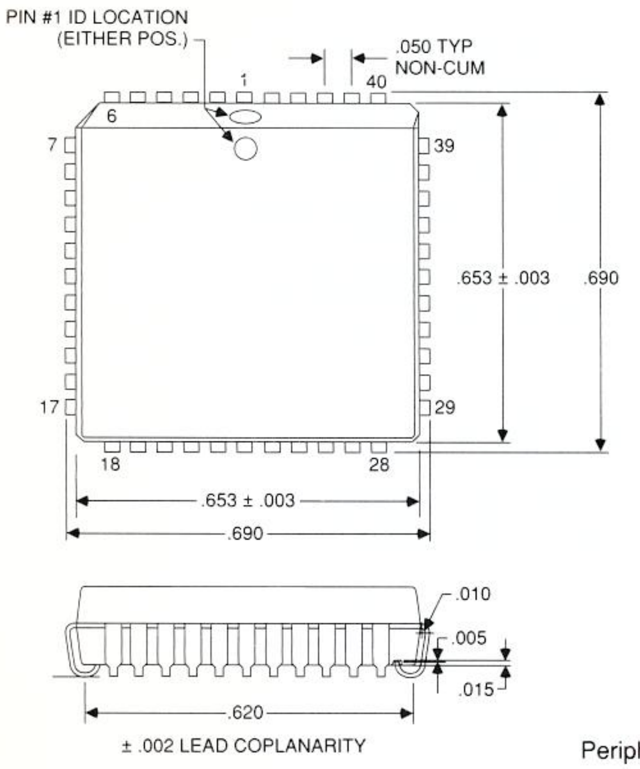
XACTOR in-circuit emulation offers:

- Real time in-circuit verification in your target system.
- Concurrent emulation of up to four LCA devices.
- Readback and display of internal flip-flop states.
- Isolation of control and I/O pins from the target system

For a description of XACTOR and its features and benefits, please read page 5-46 of our 1989 Data Book or call your Xilinx sales representative.

XC3030-PC44

Package Outline



Pin Assignment

1	**GND**	23	**GND**
2	I/O	24	I/O
3	I/O	25	I/O
4	I/O	26	XTL2(IN)-I/O
5	I/O	27	$\overline{\text{RESET}}$
6	I/O	28	DONE-$\overline{\text{PGM}}$
7	PWRDWN	29	I/O
8	TCLKIN-I/O	30	XTL1(OUT)-BCLK-I/O
9	I/O	31	I/O
10	I/O	32	I/O
11	I/O	33	I/O
12	**VCC**	34	**VCC**
13	I/O	35	I/O
14	I/O	36	I/O
15	I/O	37	I/O
16	M1-RDATA	38	DIN-I/O
17	M0-$\overline{\text{RTRIG}}$	39	DOUT-I/O
18	M2-I/O	40	CCLK
19	HDC-I/O	41	I/O
20	LDC-I/O	42	I/O
21	I/O	43	I/O
22	INIT	44	I/O

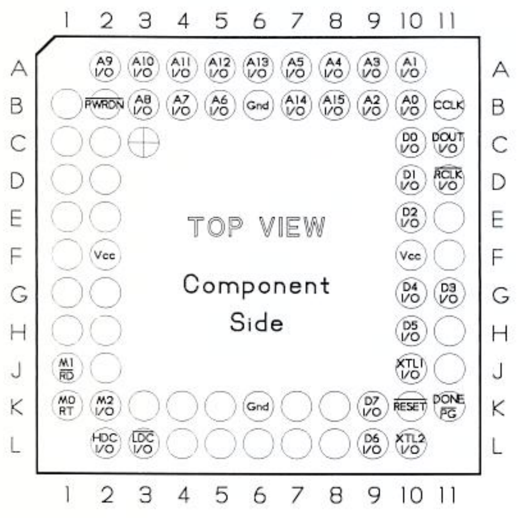
Peripheral mode and Master Parallel mode are not supported in the PC44 package

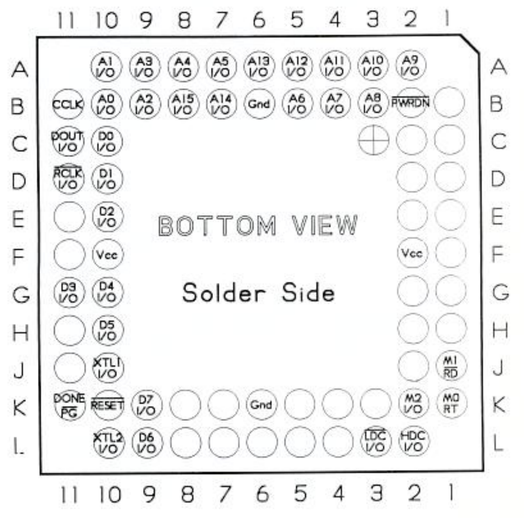
PG68 Pin-Outs

Top View

XC2064

Bottom View





⊕ = index pin which may or may not be electrically connected to pin C2

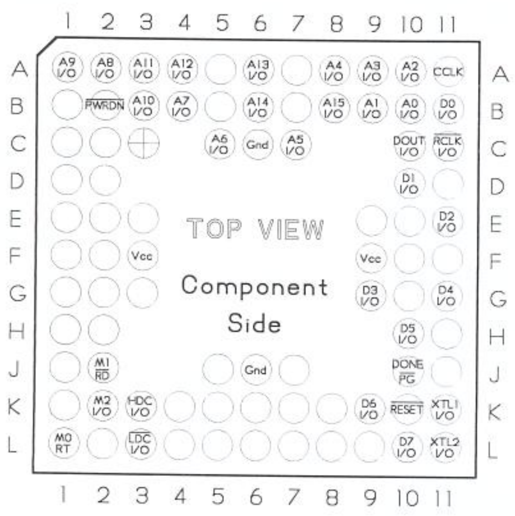
unlabeled pin = unrestricted I/O pin

⊕ = index pin which may or may not be electrically connected to pin C2

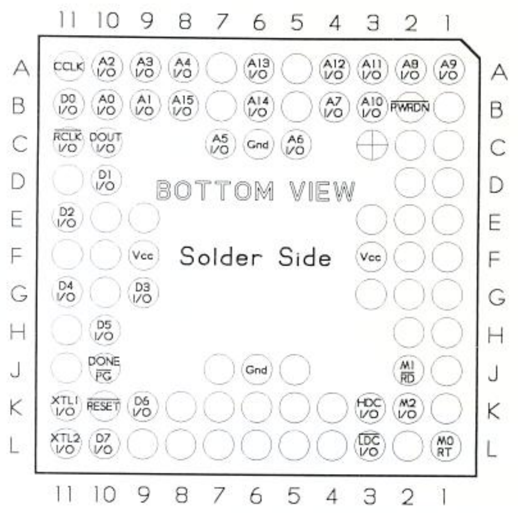
unlabeled pin = unrestricted I/O pin

PGA 84 Pin-Outs

Top View XC2018 Bottom View



TOP VIEW
Component Side
A: A9 I/O, A8 I/O, A11 I/O, A12 I/O, A13 I/O, A4 I/O, A3 I/O, A2 I/O, CCLK
B: PWRDN, A10 I/O, A7 I/O, A14 I/O, A15 I/O, A1 I/O, A0 I/O, D0 I/O
C: A6 I/O, Gnd, A5 I/O, DOUT I/O, RCLK I/O
D: D1 I/O
E: D2 I/O
F: Vcc, Vcc
G: D3 I/O, D4 I/O
H: D5 I/O
J: M1 RD, Gnd, DONE PG
K: M2 I/O, HDC I/O, D6 I/O, RESET, XTL1 I/O
L: M0 RT, LDC I/O, D7 I/O, XTL2 I/O



BOTTOM VIEW
Solder Side
A: CCLK, A2 I/O, A3 I/O, A4 I/O, A13 I/O, A12 I/O, A11 I/O, A8 I/O, A9 I/O
B: D0 I/O, A0 I/O, A1 I/O, A15 I/O, A14 I/O, A7 I/O, A10 I/O, PWRDN
C: RCLK I/O, DOUT I/O, A5 I/O, Gnd, A6 I/O
D: D1 I/O
E: D2 I/O
F: Vcc, Vcc
G: D4 I/O, D3 I/O
H: D5 I/O
J: DONE PG, Gnd, M1 RD
K: XTL1 I/O, RESET, D6 I/O, HDC I/O, M2 I/O
L: XTL2 I/O, D7 I/O, LDC I/O, M0 RT

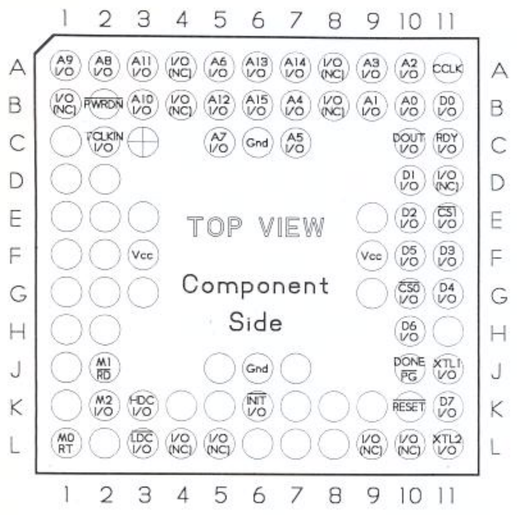
= index pin which may or may not be electrically connected to pin C2

unlabeled pin = unrestricted I/O pin

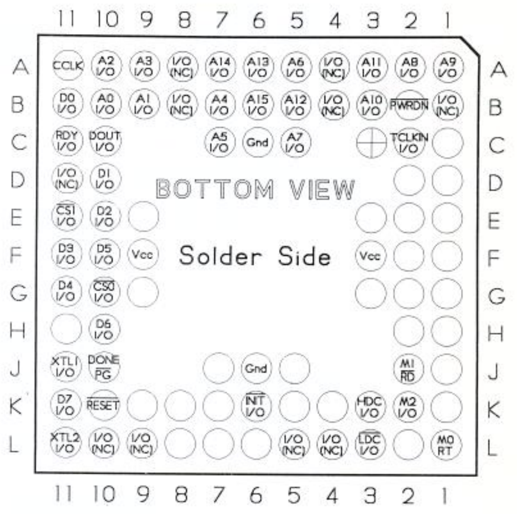
= index pin which may or may not be electrically connected to pin C2

unlabeled pin = unrestricted I/O pin

XC3042, 3030, 3020



TOP VIEW
Component Side
A: A9 I/O, A8 I/O, A11 I/O, I/O (NC), A6 I/O, A13 I/O, A14 I/O, I/O (NC), A3 I/O, A2 I/O, CCLK
B: I/O (NC), PWRDN, A10 I/O, I/O (NC), A12 I/O, A15 I/O, A4 I/O, I/O (NC), A1 I/O, A0 I/O, D0 I/O
C: TCLKIN I/O, A7 I/O, Gnd, A5 I/O, DOUT I/O, RDY I/O
D: D1 I/O, I/O (NC)
E: D2 I/O, CS1 I/O
F: Vcc, Vcc, D5 I/O, D3 I/O
G: CS0 I/O, D4 I/O
H: D6 I/O
J: M1 RD, Gnd, DONE PG, XTL1 I/O
K: M2 I/O, HDC I/O, INIT I/O, RESET, D7 I/O
L: M0 RT, LDC I/O, I/O (NC), I/O (NC), I/O (NC), I/O (NC), XTL2 I/O



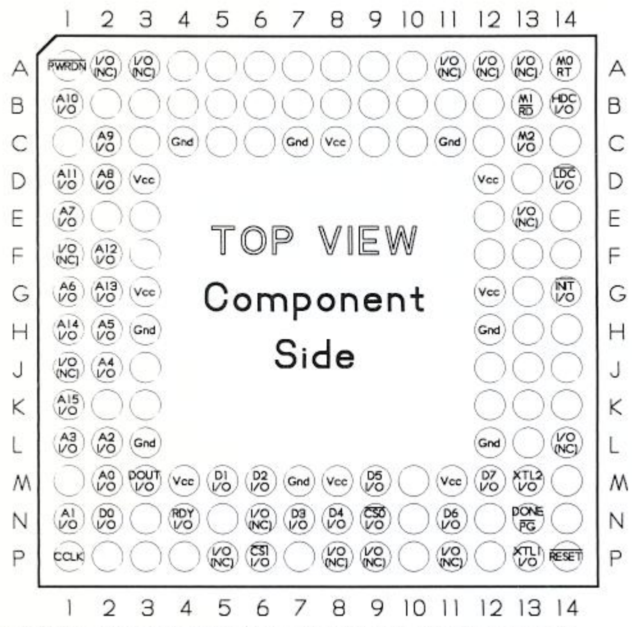
BOTTOM VIEW
Solder Side
A: CCLK, A2 I/O, A3 I/O, I/O (NC), A14 I/O, A13 I/O, A6 I/O, I/O (NC), A11 I/O, A8 I/O, A9 I/O
B: D0 I/O, A0 I/O, A1 I/O, I/O (NC), A4 I/O, A15 I/O, A12 I/O, I/O (NC), A10 I/O, PWRDN, I/O (NC)
C: RDY I/O, DOUT I/O, A5 I/O, Gnd, A7 I/O, TCLKIN I/O
D: I/O (NC), D1 I/O
E: CS1 I/O, D2 I/O
F: D3 I/O, D5 I/O, Vcc, Vcc
G: D4 I/O, CS0 I/O
H: D6 I/O
J: XTL1 I/O, DONE PG, Gnd, M1 RD
K: D7 I/O, RESET, INIT I/O, HDC I/O, M2 I/O
L: XTL2 I/O, I/O (NC), I/O (NC), I/O (NC), I/O (NC), LDC I/O, M0 RT

= index pin which may or may not be electrically connected to pin C2
(NC) = pin Not Connected for XC3020, unlabeled pin = unrestricted I/O pin

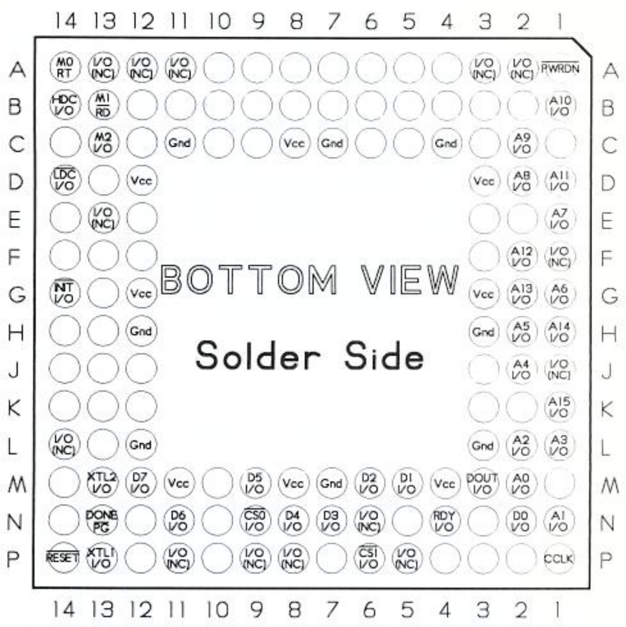
= index pin which may or may not be electrically connected to pin C2
(NC) = pin Not Connected for XC3020, unlabeled pin = unrestricted I/O pin

PGA 132 and PGA 175 Pin-Outs

Top View **XC3064, 3042, -PG, -PP** **Bottom View**

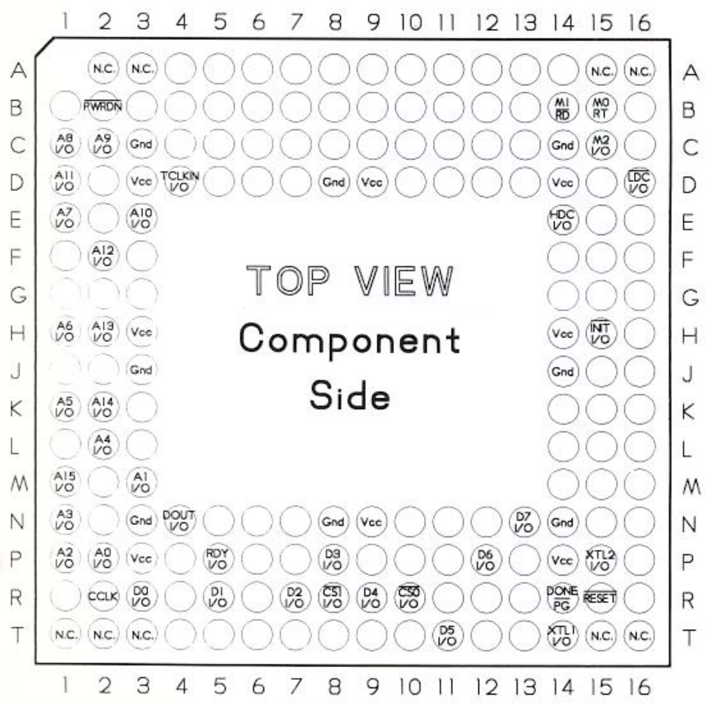


(NC) = pin Not Connected for XC3042, unlabeled pin = unrestricted I/O pin

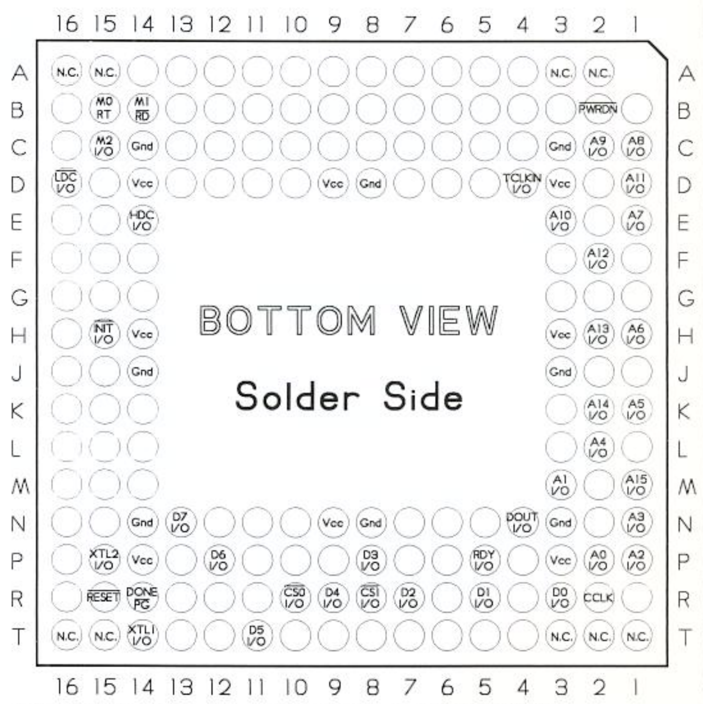


(NC) = pin Not Connected for XC3042, unlabeled pin = unrestricted I/O pin

XC3090-PG, -PP



N.C. = Not Connected unlabeled pin = unrestricted I/O pin

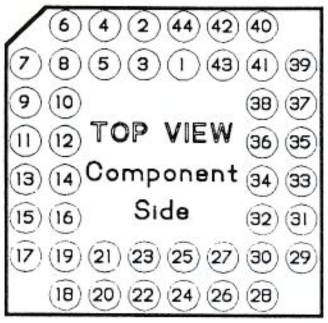


N.C. = Not Connected unlabeled pin = unrestricted I/O pin

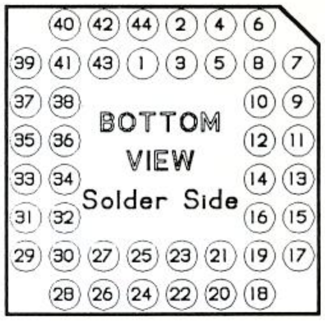
PLCC Through-Hole Socket Pin-Outs

44-Pin

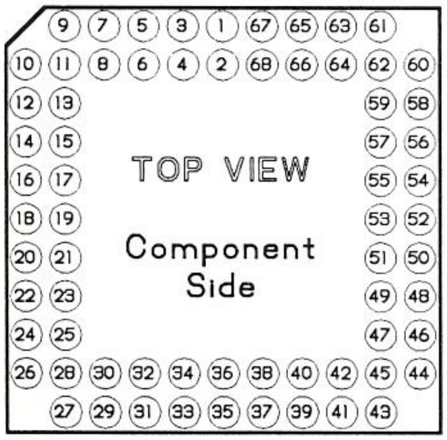
Top View

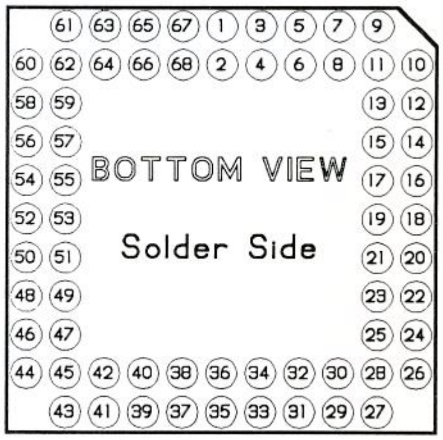


Bottom View

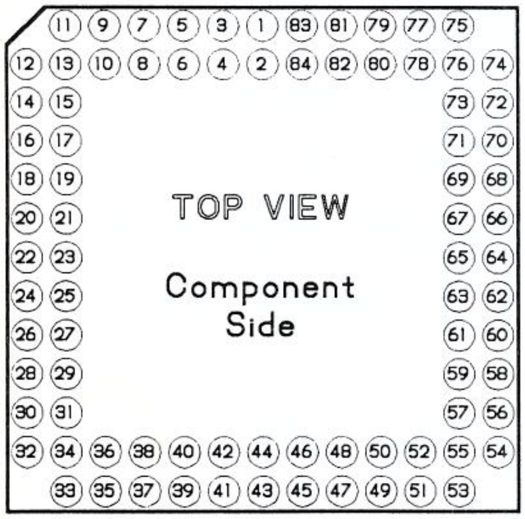


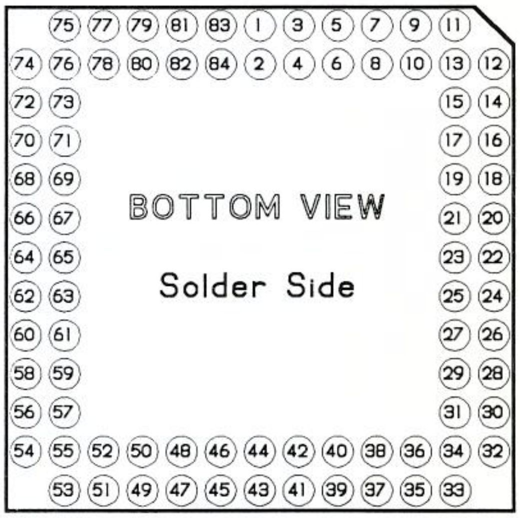
68-Pin





84-Pin

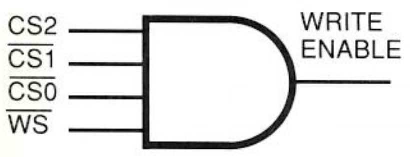




Chip Select and Write Strobe Timing in Peripheral Mode

There is really only one parameter, the write enable width. It starts when $\overline{CS0}$, $\overline{CS1}$ and $\overline{WS}$ are Low and CS2 is High. It ends when this AND condition is no longer met.

This is difficult to convey in a timing diagram, but obvious, once explained. It's the same as in any SRAM:



This Write Enable signal must have a minimum length or width. The Data Book, on page 2-47, specifies 500 ns, but the chip requires, worst case, less than 50 ns. This is a case of an unrealistically conservative timing specification. (Actual characterization data was below 20ns!)

We are still testing to the old 500 ns limit and cannot change the official spec until we change the test data, but we know that this parameter is similar to the data set-up time, which is specified and tested as 60 ns min.

PA

Max Ten 3090s per Daisy Chain

The infamous 64K byte segment size limitation of the '286 also applies to the Bitstream file. If you try to generate a configuration bitstream of more than 512K bits, the program will bomb ungracefully. The '386 behaves the same way, since we do not (yet) use it in its native mode.

The obvious alternative is to break the daisy chain into several shorter strings, create the bitstreams separately, and combine them in the host.

Note that only Peripheral and Master serial modes can handle configurations longer than 512K bits.

PA

16-Bit Adder Macros

Pages 6-28 and 6-30 of the 1989 Xilinx Data Book describe two 16-bit adder designs without giving detailed CLB maps, placement, and routing information. Both designs have now been implemented in an XC3020 and achieve respectable performance.

The Carry Lookahead adder runs at 13 MHz in a -100 part. The Conditional Sum adder runs at 30 MHz in a -100 part.

The XNF and LCA files are available from our bulletin board in the newly created MACRO section.

BON

Nanowatts, not Microwatts

LCA power consumption in the powerdown state has been somewhat of a mystery. The data book hints at nanowatts (page 2-27), but the published specifications on page 2-39 only guarantee milliwatts.

We tested a representative sample of parts and found the powerdown current at room temperature and 5V mostly below 50 nanoamps. This value is reduced in half at 2.5V, but doubles for every 10° C increase in temperature.

This is good news for battery-back-up. Even the tiniest lithium battery can power an LCA for years.

Why don't we update our guaranteed specification? One reason is the difficulty of measuring very small currents on a high-speed production tester. Another one is the potential yield loss when this parameter happens to be higher. No reason to scrap a part for a parameter that only a few users are interested in.

P.S. There was a small problem in 1988 when some 3000-family devices had a routing- and data-dependent powerdown current of about one milliamp. The problem was traced to a specific diode, and was fixed in early 1989. No part marked after April 1989 shows this larger current.

PA

Readback Clarified

The ability to read back configuration data, as well as data stored in flip-flops and latches, is crucial for the exhaustive device testing performed by Xilinx on every device before it leaves the factory.

Most of our customers have no need for this feature, but a few use Readback to verify that the configuration is still proper. This makes sense in applications that require uninterrupted operation, e.g. in telecom where the device may be configured once and then operate for months or years without ever being reconfigured.

To those few engineers who really need the readback feature we apologize for the user-unfriendly interface and the sometimes sketchy documentation.

Here are some important considerations:

Use Readback only when necessary. Less than 1% of all LCA applications use it.

Readback does not interfere with normal LCA operation, but the flip-flop data being read back will be almost impossible to interpret unless the LCA suspends its clocked operation during Readback.

Readback cannot be daisy-chained. Even when the devices were configured in a daisy-chain, they must be read back individually.

Readback data comes out inverted, a configuration 1 becomes a readback 0, and vice versa.

Readback data contains variable flip-flop or latch data in most of the locations that were left unused during configuration. If you want to compare readback against the configuration file, you must disregard (mask out) these locations as shown below.

Readback has no Preamble, and no second or third stop bit at the end of each frame.

The first frame starts with two dummy zeros instead of the single start bit (1) preceding every other frame. Remember, everything is inverted: Readback start bits are ones, stop bits are zeros.

Before the device is being configured, Readback must be enabled by the MAKEBITS menu:

0 means **never**,
1 means **once**, and
Cmd means **on command**.

Readback is initiated by a rising edge on M0. Rising edges on the CCLK input then clock out the Readback data, using the M1 pin as an output. The first rising edge of CCLK does nothing. The second and third rising edges clock out the two leading dummy zeros. The fourth and subsequent rising edges of CCLK clock out frame information, interspersed with a single 0 for stop at the end of each frame, followed by a single 1 for the start of the following frame. After the last frame stop bit has been clocked out, the M1 pin goes 3-state and further CCLK pulses are ignored.

Verifying Configuration Bitstream

In order to verify the integrity of the LCA configuration, you must compare the Readback bitstream against the configuration bit stream in all those positions not masked out by a 0 in the Mask bitstream.

Configuration bitstream and Mask bitstream have a common format, both are created from the MAKEBITS menu. Since the Readback bitstream format is different, as described above, you must adjust the formats before verification:

Either: Pad the Readback bitstream with preamble, two additional stop bits, and change the two dummy bits preceding the first frame to a normal start bit,

Or, better: Strip the Configuration and Mask bitstreams of the preamble, delete two of the three stop bits and create the two dummy bits at the beginning of the first frame. Always remember that Configuration and Readback have opposite polarity.

After the three bitstreams have been normalized you can perform the verification:

There is an error whenever the Readback bit equals the Configuration bit AND the Mask bit is a one .

PA

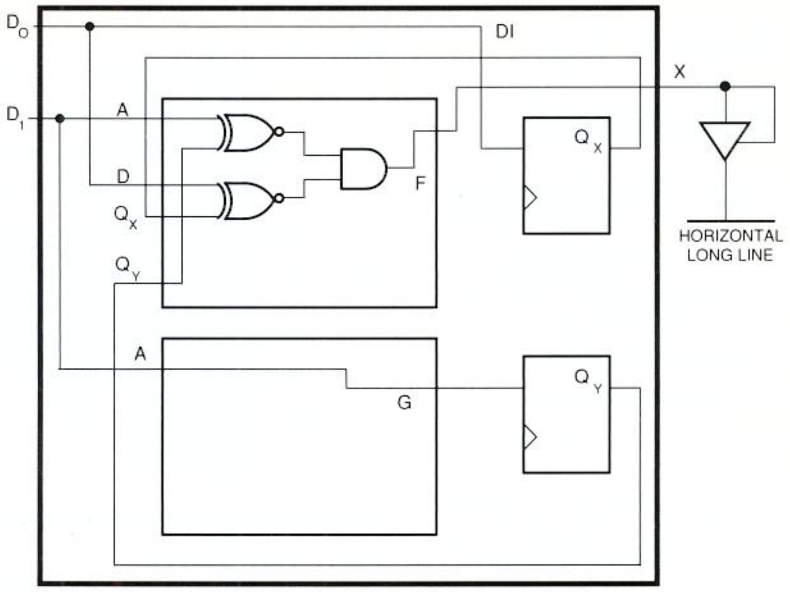
Comparing Data on a Bus

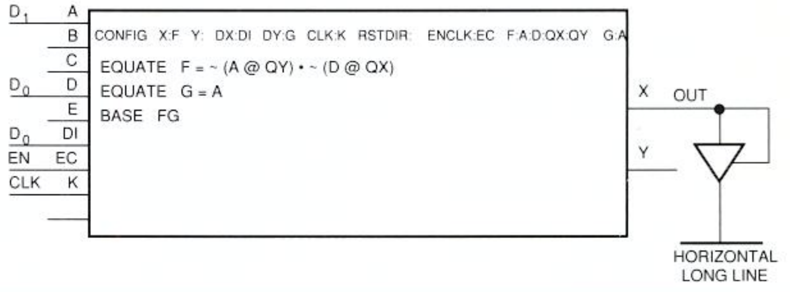
Some systems need to compare variable data on a bus against a value that had previously been loaded from the same bus. Such an identity comparator can store and compare **two data bits per CLB**, then using a Long Line to AND the result.

When Enable Clock is active, D0 (through .di) is clocked into Qx, while D1 (through .a) is clocked into Qy. D0 is also routed to the .d input.

The F function generator is brought out and drives the T input of a Long Line buffer. F is High when the two incoming bits match the registered bits.

PA





```
CONFIG  X:F  Y:  DX:DI  DY:G  CLK:K  RSTDIR:  ENCLK:EC  F:A:D:QX:QY  G:A
EQUATE  F = ~ (A @ QY) • ~ (D @ QX)
EQUATE  G = A
BASE  FG
```

Very Fast Accumulator with Pipelined Carry

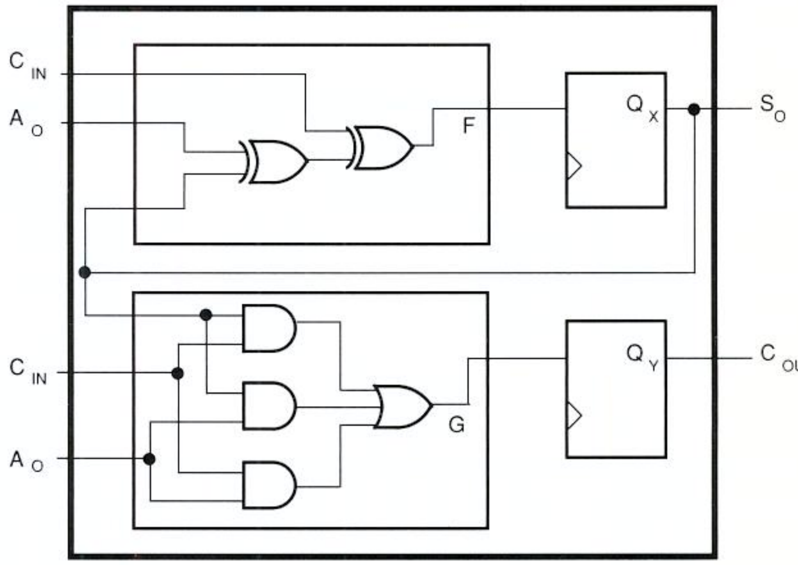
The XC3000 family can implement a very fast (>50MHz) accumulator with pipelined carry.

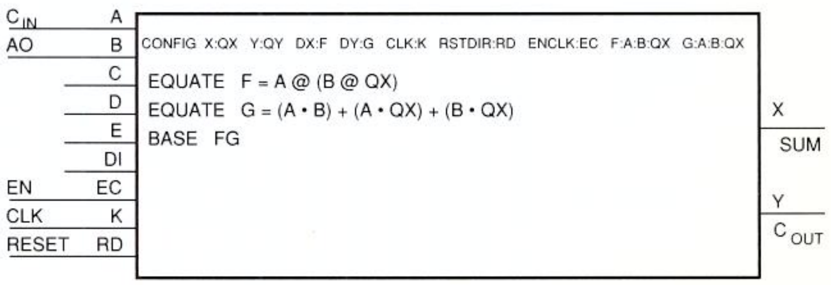
One CLB per bit stores the sum and the carry in its two flip-flops.

Each clock pulse updates the two flip-flops with the result of the addition of incoming operand plus stored sum.

There is, however, one drawback to this pipelined approach: An n-bit accumulator will need up to n-1 additional clock pulses after the last accumulation in order to flush out the carry flip-flops.

PA





```
CONFIG  X:QX  Y:QY  DX:F  DY:G  CLK:K  RSTDIR:RD  ENCLK:EC  F:A:B:QX  G:A:B:QX
EQUATE  F = A @ (B @ QX)
EQUATE  G = (A • B) + (A • QX) + (B • QX)
BASE  FG
```

Prototyping Board For Unformed CQ164 Flatpaks

A 6" by 8" board, designed by Mountain Engineering for their own prototyping work, allows placement of two unformed 164 pin CQFP packages with ample space for discrete DIP components and connectors.

This board has two milled cavities which accept the unformed CQFP164 packages and provide flat contact between the package pins and the solder pads. Power and ground are provided by the PC board, but all other interconnects must be wire-wrapped.

Boards are available immediately for $300 each, or $260 in multiples. For information, call C.J. Rigelsky, Mountain Engineering, at (303) 449-0270.

PA

SOCKET2U

Thru-Hole Mounting PLCCs

PLCCs can be soldered to the Advanced Murphy Circuits PLCC Adapter with 0.1" through-hole pin-to-pin spacing.

Advanced Interconnections
5 Energy Way
W. Warwick, RI 02893
(401) 823-5200

Low-Profile PLCC Socket

44, 68, and 84 pin surface-mount sockets with a total mated height of 0.2" are available from:

Methode Electronics, Inc.
1700 Hicks Road
Rolling Meadows, IL 60008
(312) 392-3500

ZIF Sockets For LCAs

Zero Insertion Force Sockets for PLCCs are available from:

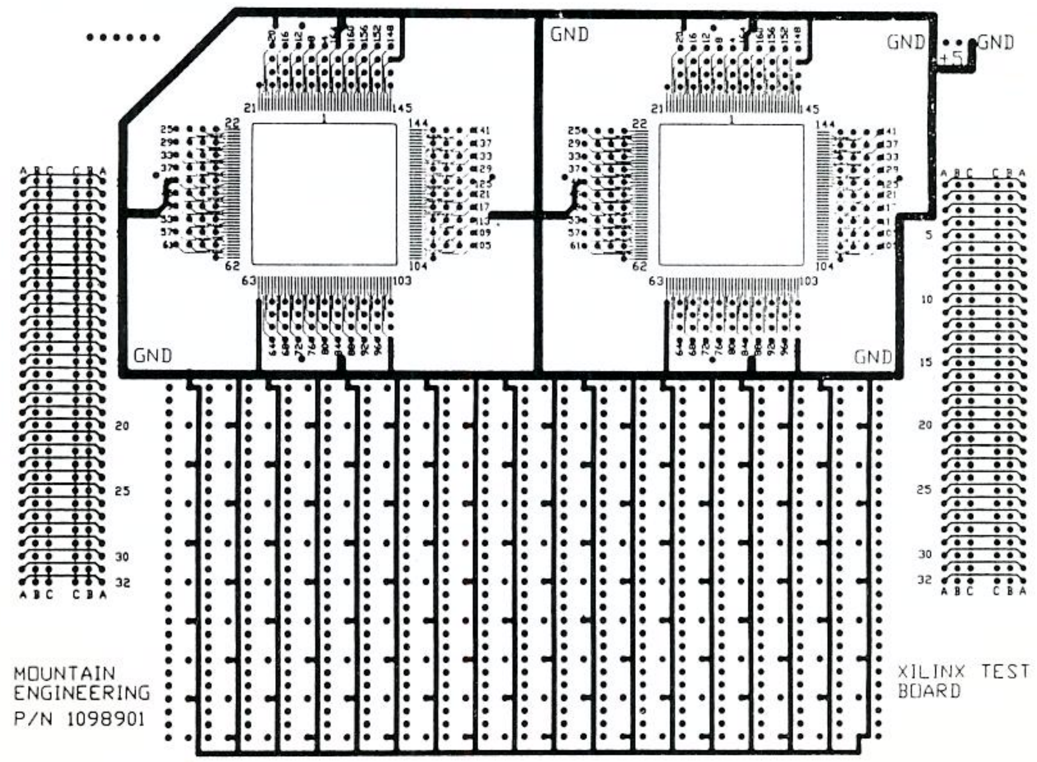
Nepenthe Co.
2471 East Bayshore Road
Palo Alto, CA 94303
(415) 856-9332
Part Numbers:
68 Pin IC 51-0684-390-1
84 Pin IC 51-0844-401-1

Wells Electronics, Inc.
1701 S. Main
South Bend, IN 46613
(219) 287-5941

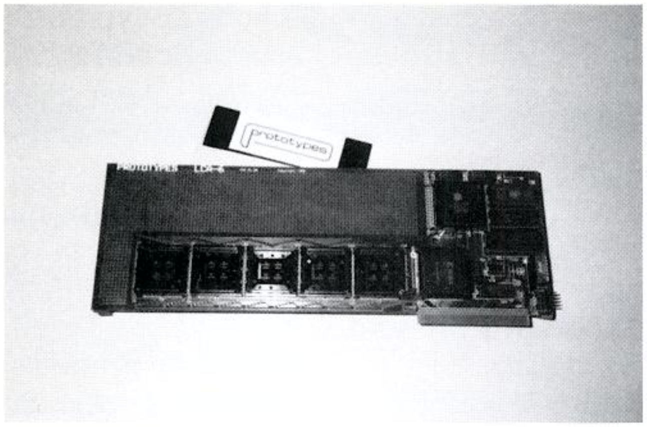
Stack LCAs For Prototyping

Two, three or four LCAs can be stacked in sockets dedicated to XILINX pin-outs of 68 and 84 pin PLCCs and PGAs :

EDI Corp. P.O. 366
Patterson, CA 95363
(209) 892-327



LCA-6 Prototyping Development System



- **Intelligent Prototyping**
- **Simulation / Debug**
- **Timing Diagram**
- **Up to 6 LCA Targets**
- **Prototyping Area**
- **Operating Environments:**
  - **PC bus linkage**
  - **Serial Linkage**
  - **Standalone**

The LCA-6 Prototyping Development System is a flexible hardware and software system that supports multiple aspects of the LCA design cycle:

- Functional verification
- Prototyping
- Design debugging
- Design documentation

The intelligent prototyping board supports up to six LCA targets and is supported by a dedicated DOS-compatible processor. The board can be used as an IBM/PC bus peripheral, as a serially linked peripheral, or as a standalone system.

Simulation testing is supported from the host system. Specific LCA modular functions can be designed and verified independently prior to final design debug. The LCA module to be tested is downloaded to the target LCA and driven by a vector script prepared by the user. Results are displayed in a timing diagram format. The simulator may also be used to drive a prototyped design to aid in debugging.

The LCA targets, prototyping area, and interconnect traces allow the designer to build entire systems on the card. The loading of the LCA targets can be handled by the on-board LCA image manager. During the active design phase, LCA images may be downloaded from the host using supplied software. The verified / debugged design may be stored in the board's EPROM or FPROM for automatic loading, or may be autoloaded from a disc file, as required.

Hardware

- Standard PC card form factor
- Single 5 volt power supply
- 6 LCA targets (PLCC 84) with bonding pads for all pin positions
- Operating environments:
  - Host bus peripheral
  - Serial linked peripheral
  - Standalone
- Intelligent Processor System
  - 8088 compatible, 16 MHz
  - 32K RAM
  - 64K EPROM
  - 128K Flash (Optional)

LCA Image Management

- Autoload from on-board memory
- DOS bus linkage
- Serial linkage
- Xilinx download cable

Simulation Testing

- Drive selected LCA design with user-prepared script
- Display sampled results in a timing diagram format

Intelligent Design Support (Optional)

- On-board monitor
- Microsoft 'C' compatible start-up module for PROM based designs
- Runtime library to support the processor and associated peripherals

PRICE: $995.00
(Standard Configuration)

Available from:

PROTOTYPES
304 Chestnut Hill Rd.
Wilton, CT 06897
(203) 834-1445

This information was supplied by the vendor.

Print FutureNet Drawings On Your LaserJet

If you have an H-P laser printer (or compatible), you now have some new printing options. A new software package called LASER CONTROL is available from Insight Development Corporation, Emeryville, CA, (415) 652-4115, and might be available at your local PC software store.

For about $90, LASER CONTROL lets you print FutureNet, Schema or OrCAD drawings on any H-P compatible laser printer. It runs approximately ten times faster than the DPLOT software. Also, since LASER CONTROL allows you to shrink an image to half or quarter size, you can print B-size and D-size drawings in an 8.5" x 11" format.

Unfortunately, quarter-size D-size drawings will still be printed in strips on two pages due to limits in the schematic capture program.

To use LASER CONTROL with FutureNet, you must set your printer options in FutureNet accordingly and print the file to disk:

```
PRINTOPT W
```

sets the output size to the widest possible;

```
PRINTOPT F
```

sets the output to the Epson FX format;

```
PRINTOPT FILE=<filename>
```

specifies output filename.

To begin printing the schematic, type `PRINT F`. Your schematic will then be printed to disk. After printing all of your schematics, exit from FutureNet and start the LASER CONTROL software. You may want to verify the correct size for the printout (e.g. full size, half size, or quarter size). The schematic printouts that you "printed" to disk can then be printed on paper by typing:

```
COPY /B <filename> LPT1:
```

LASER CONTROL has a variety of other uses as well. It works with most other PC software.

SKK

XILINX
2100 Logic Drive
San Jose, CA 95124-3450

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