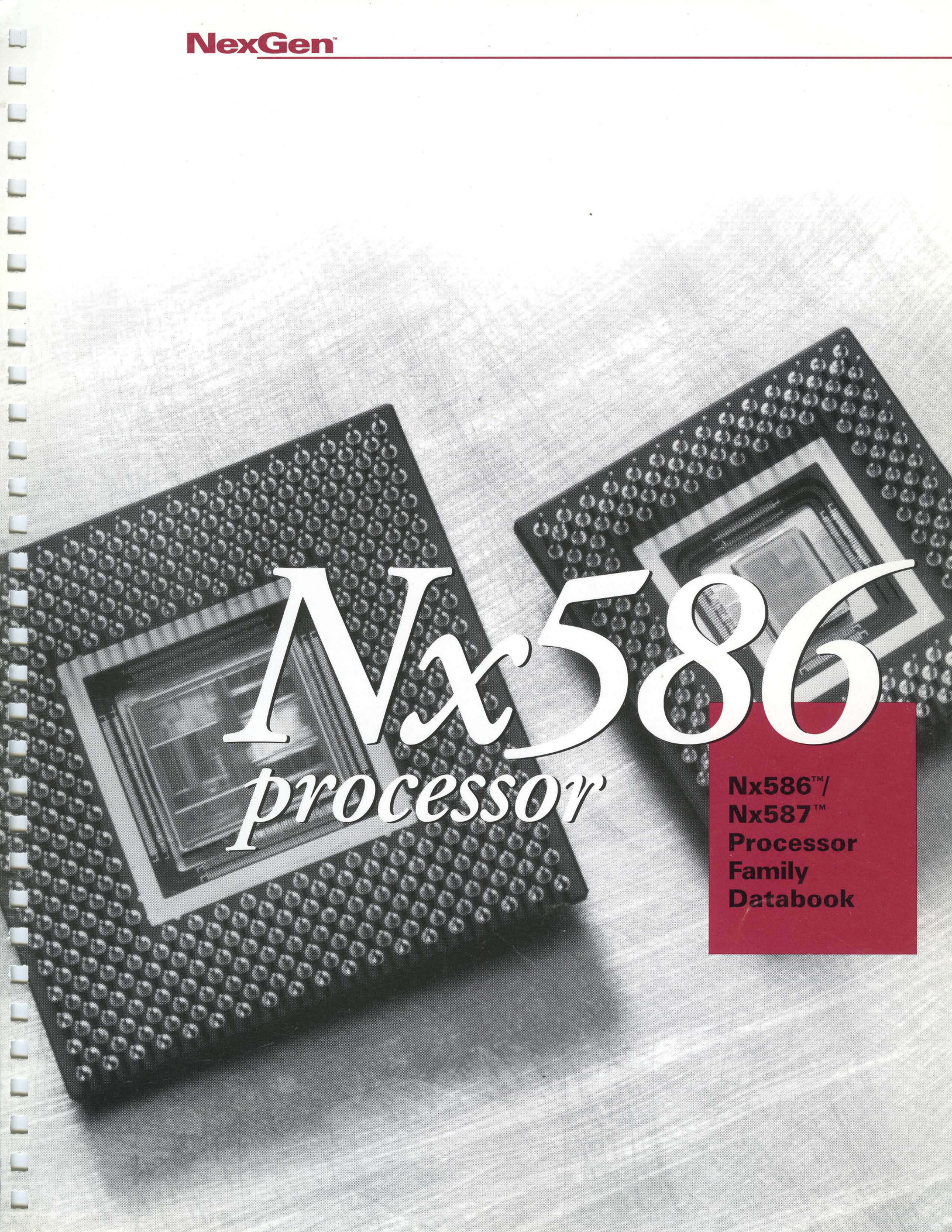
NexGen™



Nx586 processor

Nx586™/ Nx587™ Processor Family Databook

Nx586™ Processor and Nx587™ Floating Point Coprocessor Databook

PRELIMINARY
April 14, 1994

NexGen™ Microproducts, Inc.
1623 Buckeye Drive
Milpitas, CA 95035

Order # NxDOC-DB001-01-W

The goal of this databook is to enable our customers to make informed purchase decisions and to design systems around our described products. Every effort is made to provide accurate information in support of these goals. However, representations made by this databook are not intended to describe the internal logic and physical design. Wherever product internals are discussed, the information should be construed as conceptual in nature. No presumptions should be made about the internal design based on this document. Information about the internal design of NexGen products is provided via nondisclosure agreement ("NDA") on a need to know basis.

The material in this document is for information only and is subject to change without notice. NexGen reserves the right to make changes in the product specification and design without reservation and without notice to its users. THIS DOCUMENT DOES NOT CONSTITUTE A WARRANTY OF ANY KIND WITH RESPECT TO THE NEXGEN INC. PRODUCTS, AND NEXGEN INC. SHALL NOT BE LIABLE FOR ANY ERRORS THAT APPEAR IN THIS DOCUMENT.

All purchases of NexGen products shall be subject to NexGen's standard terms and conditions of sale. THE WARRANTIES AND REMEDIES EXPRESSLY SET FORTH IN SUCH TERMS AND CONDITIONS SHALL BE THE SOLE WARRANTIES AND THE BUYER'S SOLE AND EXCLUSIVE REMEDIES, AND NEXGEN INC. SPECIFICALLY DISCLAIMS ANY AND ALL OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING THE IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, AGAINST INFRINGEMENT AND OF MERCHANTABILITY. No person is authorized to make any other warranty or representation concerning the performance of the NexGen products. In particular, NexGen's products are not specifically designed, manufactured or intended for sale as components for the planning, design, construction, maintenance, operation or use of any nuclear facility or other ultra-hazardous activity, and neither NexGen nor its suppliers shall have any liability with respect to such use

Trademark Acknowledgments

NexGen, Nx586, Nx587, RISC86, NexBus, NxPCI, and NxVL are trademarks of NexGen Microproducts, Inc..

IBM, AT, and PS/2 are registered trademarks of International Business Machines, Inc. Intel is a registered trademark of Intel Corporation. i386, i387, i486 and Pentium are trademarks of Intel Corporation. Tri-state is a registered trademark of National Semiconductor Corporation. VL-Bus is a trademark of Video Electronics Standards Association.

Restricted Rights and Limitations

Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in technical Data and Computer Software clause at 252.277-7013.

Contents

Figures

Preface

This databook covers two products: the Nx586™ processor (called *the processor),* and the Nx587™ floating-point coprocessor. The databook is written for system designers considering the use of these devices in their designs. We assume an experienced audience, familiar not only with system design conventions but also with the x86 architecture. The *Glossary* at the end of the book defines NexGen's terminology, and the *Index* gives quick access to the subject matter.

NexGen's Applications Engineering Department welcomes your questions and will be glad to provide assistance. In particular, they can recommend system parts that have been tested and proven to work with NexGen™ products.

Notation

The following notation and conventions are used in this book:

Devices and Bus Names

- *Processor or CPU*—The Nx586 processor described in this book.
- *Floating Point Coprocessor*—The Nx587 floating-point coprocessor described in this book.
- NxVL™ Systems Logic—The NxVL system controller described in the *NxVL System Controller Databook.*
- NexBus™ System Bus—The Nx586 processor bus, including its multiplexed address/status and data bus (NxAD<63:0>) and related control signals.

Signals *and* Timing Diagrams

- Active-Low Signals—Signal names that are followed by an asterisk, such as ALE*, indicate active-low signals. They are said to be "asserted" or "active" in their low-voltage state and "negated" or "inactive" in their high-voltage state.

- Bus Signals—In signal names, the notation *<n:m>* represents bits *n* through *m* of a bus.
- Reserved Bits and Signals—Signals or bus bits marked "reserved" must be driven inactive or left unconnected, as indicated in the signal descriptions. These bits and signals are reserved by NexGen for future implementations. When software reads registers with reserved bits, the reserved bits must be masked. When software writes such registers, it must first read the register and change only the non-reserved bits before writing back to the register.
- Source—In timing diagrams, the left-hand column indicates the "Source" of each signal. This is the chip or logic that outputs the signal. When signals are driven by multiple sources, all sources are shown, in the order in which they drive the signal. In some cases, signals take on different names as outputs are logically ORed in group-signal logic. In these cases, the signal source is shown with a subscript, where the subscript indicates the device or logic that originally caused the change in the signal.
- *Tri-state*®—In timing diagrams, signal ranges that are high impedance are shown as a straight horizontal line half-way between the high and low level.
- *Invalid and Don't Care*—In timing diagrams, signal ranges that are invalid or don't care are filled with a screen pattern.

Data

- *Quantities*—A word is two bytes (16 bits), a dword or doubleword is four bytes (32 bits), and a qword or quadword is eight bytes (64 bits).
- Addressing—Memory is addressed as a series of bytes on eight-byte (64-bit) boundaries, in which each byte can be separately enabled.
- *Abbreviations*—The following notation is used for bits and bytes:

Bits	b	as in "64b/qword"
Bytes	B	as in "32B/block"
kilo	k	as in "4kB/page"
Mega	M	as in "1Mb/sec"
Giga	G	as in "4GB of memory space"

- *Little* Endian Convention—The byte with the address xx...xx00 is in the least-significant byte position (little end). In byte diagrams, bit positions are numbered from right to left: the little end is on the right and the big end is on the left. Data structure diagrams in memory show small addresses at the bottom and high addresses at the top. When data items are "aligned," bit notation on a 64-bit data bus maps directly to bit notation in 64-bit-wide memory. Because byte addresses increase from right to left, strings appear in reverse order when illustrated according to the little-endian convention.

- *Bit Ranges*—In a range of bits, the highest and lowest bit numbers are separated by a colon, as in <63:0>.
- *Bit Values*—Bits can either be *set* to 1 or *cleared* to 0.
- *Hexadecimal and Binary Numbers*—Unless the context makes interpretation clear, hexadecimal numbers are followed by an *h*, binary numbers are followed by a *b*, and decimal numbers are followed by a *d*.

Related Publications

The following books treat various aspects of computer architecture, hardware design, and programming that may be useful for your understanding of NexGen products:

NexGen Products

- *NxVL System Controller Databook*, NexGen, Milpitas, CA, Tel: (408) 435-0202.

x86 Architecture

- John Crawford and Patrick Gelsinger, *Programming the 80386*, Sybex, San Francisco, 1987.
- Rakesh Agarwal, *80x86 Architecture & Programming*, Volumes I and II, Prentice-Hall, Englewood Cliffs, NJ, 1991.

General References

- John L. Hennessy and David A. Patterson, *Computer Architecture*, Morgan Kaufmann Publishers, San Mateo, CA, 1990.

Nx586 Features and Signals

The NexGen Nx586 processor is an advanced 5th generation 32-bit Superscalar x86 compatible processor that provides market leading performance. The Nx586 along with the Nx587 floating-point coprocessor are the core building blocks of a new class of personal computers. The following are some of the key features of the Nx586 Processor:

- **Full x86 Binary Compatibility**—Supports 8, 16 and 32-bit data types and operates in real, virtual 8086 and protected modes.
- **Patented RISC86™ Superscalar Microarchitecture**—Multiple operations are executed simultaneously during each cycle.
- **Multi-Level Storage Hierarchy**—Branch prediction, readable write queue, on-chip L1 code and data caches and unified L2 cache.
- **Separate on-chip L1 Code and Data Caches**—supports on-chip 4-way, 16kByte Code and 16kByte Data caches using MESI Cache Consistency Protocol.
- **On-Chip L2 Cache Controller**— supporting 4-way, unified, MESI modified write-back cache coherency protocol on 256kB or 1MB of external cache using standard asynchronous SRAMs.
- **Patented Branch Prediction Logic**—Reduces both control dependencies and branch cycle counts.
- **Dual-Port Caches**—64-bit reads and writes are serviced in parallel in a single clock cycle.
- **Caches Decoupled From Processor Bus**—Both the L1 and L2 caches are accessed on separate dedicated buses.
- **Two-Phase, Non-Overlapped Clocking**—Integrated phase-locked loop bus-clock doubler. Processor operates at twice the system bus frequency.
- **Three 64-Bit Synchronous Buses**—NexBus (the processor bus), L2 SRAM bus, and Nx587 Floating-Point Coprocessor bus and is fully integrat3d into the processor microarchitecture.
- **Optional in Line Floating-Point Coprocessor**— Nx587 operates in parallel with the Nx586 pipeline.
- **Advanced State-of-the-Art Fabrication Process**—0.5 micron CMOS

Figure 1 shows the signal organization for the Nx586 processor. The processor supports signals for the NexBus (the processor bus), L2 cache, and the optional Nx587 Floating-Point Coprocessor. Many types of devices can be interfaced to the NexBus, including a backplane, multiple Nx586 processors, shared memory subsystems, high-speed I/O, and industry-standard buses. All signals are synchronous to the NexBus clock (CLK) and transition at the rising edge of the clock with the exception of four asynchronous signals: INTR*, NMI*, GATEA20, and SLOTID<3:0>. All bi-directional NexBus signals are floated unless they are needed during specific time periods, as specified in the *Bus Operation* chapter. The normal state for all reserved bits is high.

Two types of NexBus signals deserve special mention:

- *Group Signals*—There are several *group signals* on the NexBus, typically denoted by signal names beginning with the letter "G." Active-low signals such as ALE* are driven by each NexBus device, and the arbiter derives an active-high group signal (such as GALE) and distributes it back to each device. When the NxVL is used, these group signals are generated within the NxVL.
- *Central Bus Arbitration*—Access to the NexBus is arbitrated by an external NexBus Arbiter. NexBus masters request and are granted access by this Arbiter. For the Nx586 processor, central bus arbitration has the advantage of back-to-back processor access most of the time while supporting fast switching between masters. The NxVL provides the combined functions of NexBus Arbiter, Alternate-Bus Interface (the system-logic interface to other system buses), and memory controller. The NxVL gives the processor back-to-back use of the bus when no device on any other system bus needs access.

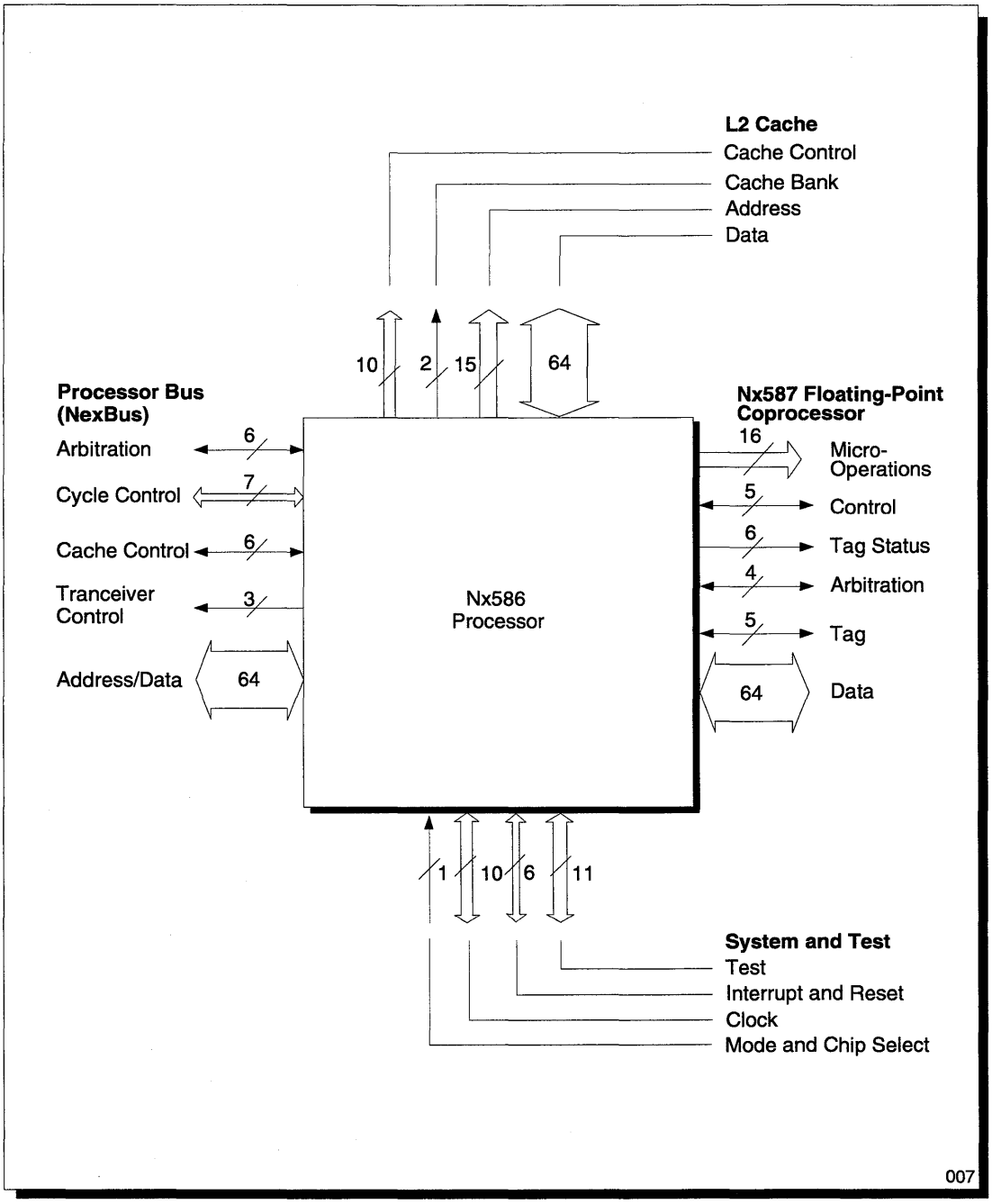


Figure 1 Nx586 Signal Organization

Nx586 Pinouts by Signal Names

Pin	Type	Signal
449	O	ALE*
18	I	ANALYZEIN
168	O	ANALYZEOUT
340	O	AREQ*
141	O	CADDR<3>
123	O	CADDR<4>
124	O	CADDR<5>
32	O	CADDR<6>
14	O	CADDR<7>
33	O	CADDR<8>
15	O	CADDR<9>
34	O	CADDR<10>
90	O	CADDR<11>
107	O	CADDR<12>
88	O	CADDR<13>
106	O	CADDR<14>
142	O	CADDR<15>
169	O	CADDR<16>
35	O	CADDR<17>
89	O	CBANK<0>
16	O	CBANK<1>
100	I/O	CDATA<0>
7	I/O	CDATA<1>
81	I/O	CDATA<2>
136	I/O	CDATA<3>
24	I/O	CDATA<4>
80	I/O	CDATA<5>
6	I/O	CDATA<6>
99	I/O	CDATA<7>
9	I/O	CDATA<8>
83	I/O	CDATA<9>
27	I/O	CDATA<10>
119	I/O	CDATA<11>
118	I/O	CDATA<12>
26	I/O	CDATA<13>
82	I/O	CDATA<14>
8	I/O	CDATA<15>
11	I/O	CDATA<16>
103	I/O	CDATA<17>
29	I/O	CDATA<18>
121	I/O	CDATA<19>
139	I/O	CDATA<20>
28	I/O	CDATA<21>
102	I/O	CDATA<22>
10	I/O	CDATA<23>
13	I/O	CDATA<24>
105	I/O	CDATA<25>
31	I/O	CDATA<26>
86	I/O	CDATA<27>
122	I/O	CDATA<28>
85	I/O	CDATA<29>
30	I/O	CDATA<30>
104	I/O	CDATA<31>
147	I/O	CDATA<32>
129	I/O	CDATA<33>
110	I/O	CDATA<34>
92	I/O	CDATA<35>
87	I/O	CDATA<36>
125	I/O	CDATA<37>
176	I/O	CDATA<38>
184	I/O	CDATA<39>
203	I/O	CDATA<40>
193	I/O	CDATA<41>
195	I/O	CDATA<42>
185	I/O	CDATA<43>
155	I/O	CDATA<44>
163	I/O	CDATA<45>
171	I/O	CDATA<46>
179	I/O	CDATA<47>
217	I/O	CDATA<48>
227	I/O	CDATA<49>
225	I/O	CDATA<50>
224	I/O	CDATA<51>
201	I/O	CDATA<52>
211	I/O	CDATA<53>
209	I/O	CDATA<54>
219	I/O	CDATA<55>
240	I/O	CDATA<56>
251	I/O	CDATA<57>
249	I/O	CDATA<58>
248	I/O	CDATA<59>
232	I/O	CDATA<60>
241	I/O	CDATA<61>
243	I/O	CDATA<62>
233	I/O	CDATA<63>
361	I	CKMODE
452	I	CLK
192	O	COEA*
138	O	COEB*
117	O	CWE<0>*
137	O	CWE<1>*
120	O	CWE<2>*
140	O	CWE<3>*
55	O	CWE<4>*
177	O	CWE<5>*
200	O	CWE<6>*
216	O	CWE<7>*
359	O	DCL*
330	I	GALE
339	I	GATEA20
378	I	GBLKNBL
429	I	GDCL
368	I	GNT*
113	I	GREF
430	I	GSHARE
322	I	GTAL
349	I	GXACK
377	I	GXHLD
36	I	HROM
375	I	INTR*
323	I	IREF
341	O	LOCK*
25	-	NC
101	-	NC
84	-	NC
12	-	NC
17	-	NC
187	-	NC
208	-	NC
235	-	NC
288	-	NC
256	-	NC
143	-	NC
380	-	NC
436	I	NMI*
244	I/O	NPDATA<0>
254	I/O	NPDATA<1>
292	I/O	NPDATA<2>
408	I/O	NPDATA<3>
336	I/O	NPDATA<4>
294	I/O	NPDATA<5>
286	I/O	NPDATA<6>
223	I/O	NPDATA<7>
206	I/O	NPDATA<8>
427	I/O	NPDATA<9>
255	I/O	NPDATA<10>
230	I/O	NPDATA<11>
236	I/O	NPDATA<12>
183	I/O	NPDATA<13>
212	I/O	NPDATA<14>
191	I/O	NPDATA<15>
390	I/O	NPDATA<16>
215	I/O	NPDATA<17>
199	I/O	NPDATA<18>
318	I/O	NPDATA<19>
262	I/O	NPDATA<20>
228	I/O	NPDATA<21>
295	I/O	NPDATA<22>
260	I/O	NPDATA<23>
445	I/O	NPDATA<24>
95	I/O	NPDATA<25>
428	I/O	NPDATA<26>
220	I/O	NPDATA<27>
303	I/O	NPDATA<28>
310	I/O	NPDATA<29>
268	I/O	NPDATA<30>
263	I/O	NPDATA<31>
356	I/O	NPDATA<32>
196	I/O	NPDATA<33>
302	I/O	NPDATA<34>
300	I/O	NPDATA<35>
287	I/O	NPDATA<36>
180	I/O	NPDATA<37>
207	I/O	NPDATA<38>
247	I/O	NPDATA<39>
198	I/O	NPDATA<40>
308	I/O	NPDATA<41>
373	I/O	NPDATA<42>
246	I/O	NPDATA<43>
278	I/O	NPDATA<44>
190	I/O	NPDATA<45>
338	I/O	NPDATA<46>
231	I/O	NPDATA<47>
276	I/O	NPDATA<48>
279	I/O	NPDATA<49>
238	I/O	NPDATA<50>
271	I/O	NPDATA<51>
270	I/O	NPDATA<52>
316	I/O	NPDATA<53>
371	I/O	NPDATA<54>
284	I/O	NPDATA<55>
188	I/O	NPDATA<56>
222	I/O	NPDATA<57>
311	I/O	NPDATA<58>
334	I/O	NPDATA<59>
239	I/O	NPDATA<60>
252	I/O	NPDATA<61>
204	I/O	NPDATA<62>
353	I/O	NPDATA<63>
446	O	NPIRQ*
337	I	NPNOERR
172	O	NPOUTFTYP<0>
98	O	NPOUTFTYP<1>
79	O	NPPOPBUS<0>
116	O	NPPOPBUS<1>
3	O	NPPOPBUS<2>
93	O	NPPOPBUS<3>
164	O	NPPOPBUS<4>
135	O	NPPOPBUS<5>
134	O	NPPOPBUS<6>
21	O	NPPOPBUS<7>
2	O	NPPOPBUS<8>
97	O	NPPOPBUS<9>
148	O	NPPOPBUS<10>
74	O	NPPOPBUS<11>
22	O	NPPOPBUS<12>
156	O	NPPOPBUS<13>
23	O	NPPOPBUS<14>
96	O	NPPOPBUS<15>
37	O	NPPOPTAG<0>
159	O	NPPOPTAG<1>
56	O	NPPOPTAG<2>
132	O	NPPOPTAG<3>
151	O	NPPOPTAG<4>
182	O	NPRREQ
174	O	NPRVAL
167	I	NPSPARE<0>
150	I	NPSPARE<1>
158	I	NPSPARE<2>
5	I/O	NPTAG<0>
77	I/O	NPTAG<1>
20	I/O	NPTAG<2>
111	I/O	NPTAG<3>
115	I/O	NPTAG<4>
19	O	NPTAGSTAT<0>
1	O	NPTAGSTAT<1>
175	O	NPTAGSTAT<2>
78	O	NPTAGSTAT<3>
4	O	NPTAGSTAT<4>
166	O	NPTAGSTAT<5>
133	I	NPTERM<0>
114	I	NPTERM<1>

Figure 2 Nx586 Pin List, By Signal Name

Pin	Type	Signal
355	I	NPWREQ
319	I	NPWVAL
321	O	NREQ*
296	I/O	NxAD<0>
267	I/O	NxAD<1>
307	I/O	NxAD<2>
297	I/O	NxAD<3>
443	I/O	NxAD<4>
444	I/O	NxAD<5>
463	I/O	NxAD<6>
312	I/O	NxAD<7>
313	I/O	NxAD<8>
315	I/O	NxAD<9>
281	I/O	NxAD<10>
283	I/O	NxAD<11>
459	I/O	NxAD<12>
460	I/O	NxAD<13>
441	I/O	NxAD<14>
348	I/O	NxAD<15>
387	I/O	NxAD<16>
370	I/O	NxAD<17>
331	I/O	NxAD<18>
333	I/O	NxAD<19>
325	I/O	NxAD<20>
345	I/O	NxAD<21>
327	I/O	NxAD<22>
383	I/O	NxAD<23>
347	I/O	NxAD<24>
384	I/O	NxAD<25>
458	I/O	NxAD<26>
346	I/O	NxAD<27>
438	I/O	NxAD<28>
382	I/O	NxAD<29>
437	I/O	NxAD<30>
455	I/O	NxAD<31>
259	I/O	NxAD<32>
257	I/O	NxAD<33>
265	I/O	NxAD<34>
275	I/O	NxAD<35>
273	I/O	NxAD<36>
462	I/O	NxAD<37>
304	I/O	NxAD<38>
426	I/O	NxAD<39>
299	I/O	NxAD<40>
289	I/O	NxAD<41>
291	I/O	NxAD<42>
305	I/O	NxAD<43>
440	I/O	NxAD<44>
366	I/O	NxAD<45>
367	I/O	NxAD<46>
385	I/O	NxAD<47>
407	I/O	NxAD<48>
389	I/O	NxAD<49>
350	I/O	NxAD<50>
352	I/O	NxAD<51>
343	I/O	NxAD<52>
344	I/O	NxAD<53>
326	I/O	NxAD<54>

Pin	Type	Signal
457	I/O	NxAD<55>
329	I/O	NxAD<56>
328	I/O	NxAD<57>
365	I/O	NxAD<58>
439	I/O	NxAD<59>
364	I/O	NxAD<60>
456	I/O	NxAD<61>
363	I/O	NxAD<62>
381	I/O	NxAD<63>
73	O	NxADINUSE
447	I	OWNABL
76	I	P4REF
453	I	PHE1
379	I	PHE2
153	I	POPHOLD
160	I	PULLHIGH
145	I	PULLHIGH
320	I/O	PULLHIGH
357	I/O	PULLHIGH
376	I/O	PULLHIGH
431	I/O	PULLHIGH
432	I/O	PULLHIGH
433	I/O	PULLHIGH
450	I/O	PULLHIGH
451	I/O	PULLHIGH
264	I/O	PULLLOW
272	I	PTEST
214	I	RESET*
362	I	RESETCPU*
144	I	SERIALIN
280	O	SERIALOUT
448	O	SHARE*
130	I	SLOTID<0>
161	I	SLOTID<1>
152	I	SLOTID<2>
127	I	SLOTID<3>
374	I	TESTPWR*
108	I	TPH1
126	I	TPH2
57	I	VCC4
58	I	VCC4
59	I	VCC4
60	I	VCC4
61	I	VCC4
62	I	VCC4
63	I	VCC4
64	I	VCC4
65	I	VCC4
66	I	VCC4
67	I	VCC4
68	I	VCC4
69	I	VCC4
70	I	VCC4
71	I	VCC4
72	I	VCC4
94	I	VCC4
109	I	VCC4
131	I	VCC4

Pin	Type	Signal
146	I	VCC4
157	I	VCC4
162	I	VCC4
173	I	VCC4
178	I	VCC4
189	I	VCC4
194	I	VCC4
205	I	VCC4
210	I	VCC4
221	I	VCC4
226	I	VCC4
237	I	VCC4
242	I	VCC4
253	I	VCC4
258	I	VCC4
269	I	VCC4
274	I	VCC4
285	I	VCC4
290	I	VCC4
301	I	VCC4
306	I	VCC4
317	I	VCC4
332	I	VCC4
354	I	VCC4
369	I	VCC4
391	I	VCC4
392	I	VCC4
393	I	VCC4
394	I	VCC4
395	I	VCC4
396	I	VCC4
397	I	VCC4
398	I	VCC4
399	I	VCC4
400	I	VCC4
401	I	VCC4
402	I	VCC4
403	I	VCC4
404	I	VCC4
405	I	VCC4
406	I	VCC4
324	I	VDDA
38	I	VSS
39	I	VSS
40	I	VSS
41	I	VSS
42	I	VSS
43	I	VSS
44	I	VSS
45	I	VSS
46	I	VSS
47	I	VSS
48	I	VSS
49	I	VSS
50	I	VSS
51	I	VSS
52	I	VSS
53	I	VSS

Pin	Type	Signal
54	I	VSS
75	I	VSS
91	I	VSS
112	I	VSS
128	I	VSS
149	I	VSS
154	I	VSS
165	I	VSS
170	I	VSS
181	I	VSS
186	I	VSS
197	I	VSS
202	I	VSS
213	I	VSS
218	I	VSS
229	I	VSS
234	I	VSS
245	I	VSS
250	I	VSS
261	I	VSS
266	I	VSS
277	I	VSS
282	I	VSS
293	I	VSS
298	I	VSS
309	I	VSS
314	I	VSS
335	I	VSS
351	I	VSS
372	I	VSS
388	I	VSS
409	I	VSS
410	I	VSS
411	I	VSS
412	I	VSS
413	I	VSS
414	I	VSS
415	I	VSS
416	I	VSS
417	I	VSS
418	I	VSS
419	I	VSS
420	I	VSS
421	I	VSS
422	I	VSS
423	I	VSS
424	I	VSS
425	I	VSS
358	O	XACK*
386	O	XBCKE*
461	O	XBOE*
454	O	XHLD*
442	O	XNOE*
360	O	XPH1
342	O	XPH2
434	O	XREF
435	I	XSEL

Figure 3 Nx586 Pin List, By Signal Name (continued)

Nx586 Pinouts by Pin Numbers

Pin	Type	Signal
1	O	NPTAGSTAT<1>
2	O	NPPOPBUS<8>
3	O	NPPOPBUS<2>
4	O	NPTAGSTAT<4>
5	I/O	NPTAG<0>
6	I/O	CDATA<6>
7	I/O	CDATA<1>
8	I/O	CDATA<15>
9	I/O	CDATA<8>
10	I/O	CDATA<23>
11	I/O	CDATA<16>
12	-	NC
13	I/O	CDATA<24>
14	O	CADDR<7>
15	O	CADDR<9>
16	O	CBANK<1>
17	-	NC
18	I	ANALYZEIN
19	O	NPTAGSTAT<0>
20	I/O	NPTAG<2>
21	O	NPPOPBUS<7>
22	O	NPPOPBUS<12>
23	O	NPPOPBUS<14>
24	I/O	CDATA<4>
25	-	NC
26	I/O	CDATA<13>
27	I/O	CDATA<10>
28	I/O	CDATA<21>
29	I/O	CDATA<18>
30	I/O	CDATA<30>
31	I/O	CDATA<26>
32	O	CADDR<6>
33	O	CADDR<8>
34	O	CADDR<10>
35	O	CADDR<17>
36	I	HROM
37	O	NPPOPTAG<0>
38	I	VSS
39	I	VSS
40	I	VSS
41	I	VSS
42	I	VSS
43	I	VSS
44	I	VSS
45	I	VSS
46	I	VSS
47	I	VSS
48	I	VSS
49	I	VSS
50	I	VSS
51	I	VSS
52	I	VSS
53	I	VSS
54	I	VSS
55	O	CWE<4>*
56	O	NPPOPTAG<2>

Pin	Type	Signal
57	I	VCC4
58	I	VCC4
59	I	VCC4
60	I	VCC4
61	I	VCC4
62	I	VCC4
63	I	VCC4
64	I	VCC4
65	I	VCC4
66	I	VCC4
67	I	VCC4
68	I	VCC4
69	I	VCC4
70	I	VCC4
71	I	VCC4
72	I	VCC4
73	O	NxADINUSE
74	O	NPPOPBUS<11>
75	I	VSS
76	I	P4REF
77	I/O	NPTAG<1>
78	O	NPTAGSTAT<3>
79	O	NPPOPBUS<0>
80	I/O	CDATA<5>
81	I/O	CDATA<2>
82	I/O	CDATA<14>
83	I/O	CDATA<9>
84	-	NC
85	I/O	CDATA<29>
86	I/O	CDATA<27>
87	I/O	CDATA<36>
88	O	CADDR<13>
89	O	CBANK<0>
90	O	CADDR<11>
91	I	VSS
92	I/O	CDATA<35>
93	O	NPPOPBUS<3>
94	I	VCC4
95	I/O	NPDATA<25>
96	O	NPPOPBUS<15>
97	O	NPPOPBUS<9>
98	O	NPOUTFTYP<1>
99	I/O	CDATA<7>
100	I/O	CDATA<0>
101	-	NC
102	I/O	CDATA<22>
103	I/O	CDATA<17>
104	I/O	CDATA<31>
105	I/O	CDATA<25>
106	O	CADDR<14>
107	O	CADDR<12>
108	I	TPH1
109	I	VCC4
110	I/O	CDATA<34>
111	I/O	NPTAG<3>
112	I	VSS

Pin	Type	Signal
113	I	GREF
114	I	NPTERM<1>
115	I/O	NPTAG<4>
116	O	NPPOPBUS<1>
117	O	CWE<0>*
118	I/O	CDATA<12>
119	I/O	CDATA<11>
120	O	CWE<2>*
121	I/O	CDATA<19>
122	I/O	CDATA<28>
123	O	CADDR<4>
124	O	CADDR<5>
125	I/O	CDATA<37>
126	I	TPH2
127	I	SLOTID<3>
128	I	VSS
129	I/O	CDATA<33>
130	I	SLOTID<0>
131	I	VCC4
132	O	NPPOPTAG<3>
133	I	NPTERM<0>
134	O	NPPOPBUS<6>
135	O	NPPOPBUS<5>
136	I/O	CDATA<3>
137	O	CWE<1>*
138	O	COEB*
139	I/O	CDATA<20>
140	O	CWE<3>*
141	O	CADDR<3>
142	O	CADDR<15>
143	-	NC
144	I	SERIALIN
145	I	PULLHIGH
146	I	VCC4
147	I/O	CDATA<32>
148	O	NPPOPBUS<10>
149	I	VSS
150	I	NPSPARE<1>
151	O	NPPOPTAG<4>
152	I	SLOTID<2>
153	I	POPHOLD
154	I	VSS
155	I/O	CDATA<44>
156	O	NPPOPBUS<13>
157	I	VCC4
158	I	NPSPARE<2>
159	O	NPPOPTAG<1>
160	I	PULLHIGH
161	I	SLOTID<1>
162	I	VCC4
163	I/O	CDATA<45>
164	O	NPPOPBUS<4>
165	I	VSS
166	O	NPTAGSTAT<5>
167	I	NPSPARE<0>
168	O	ANALYZEOUT

Pin	Type	Signal
169	O	CADDR<16>
170	I	VSS
171	I/O	CDATA<46>
172	O	NPOUTFTYP<0>
173	I	VCC4
174	O	NPRVAL
175	O	NPTAGSTAT<2>
176	I/O	CDATA<38>
177	O	CWE<5>*
178	I	VCC4
179	I/O	CDATA<47>
180	I/O	NPDATA<37>
181	I	VSS
182	O	NPRREQ
183	I/O	NPDATA<13>
184	I/O	CDATA<39>
185	I/O	CDATA<43>
186	I	VSS
187	-	NC
188	I/O	NPDATA<56>
189	I	VCC4
190	I/O	NPDATA<45>
191	I/O	NPDATA<15>
192	O	COEA*
193	I/O	CDATA<41>
194	I	VCC4
195	I/O	CDATA<42>
196	I/O	NPDATA<33>
197	I	VSS
198	I/O	NPDATA<40>
199	I/O	NPDATA<18>
200	O	CWE<6>*
201	I/O	CDATA<52>
202	I	VSS
203	I/O	CDATA<40>
204	I/O	NPDATA<62>
205	I	VCC4
206	I/O	NPDATA<8>
207	I/O	NPDATA<38>
208	-	NC
209	I/O	CDATA<54>
210	I	VCC4
211	I/O	CDATA<53>
212	I/O	NPDATA<14>
213	I	VSS
214	I	RESET*
215	I/O	NPDATA<17>
216	O	CWE<7>*
217	I/O	CDATA<48>
218	I	VSS
219	I/O	CDATA<55>
220	I/O	NPDATA<27>
221	I	VCC4
222	I/O	NPDATA<57>
223	I/O	NPDATA<7>
224	I/O	CDATA<51>

Figure 4 Nx586 Pin List, By Pin Number

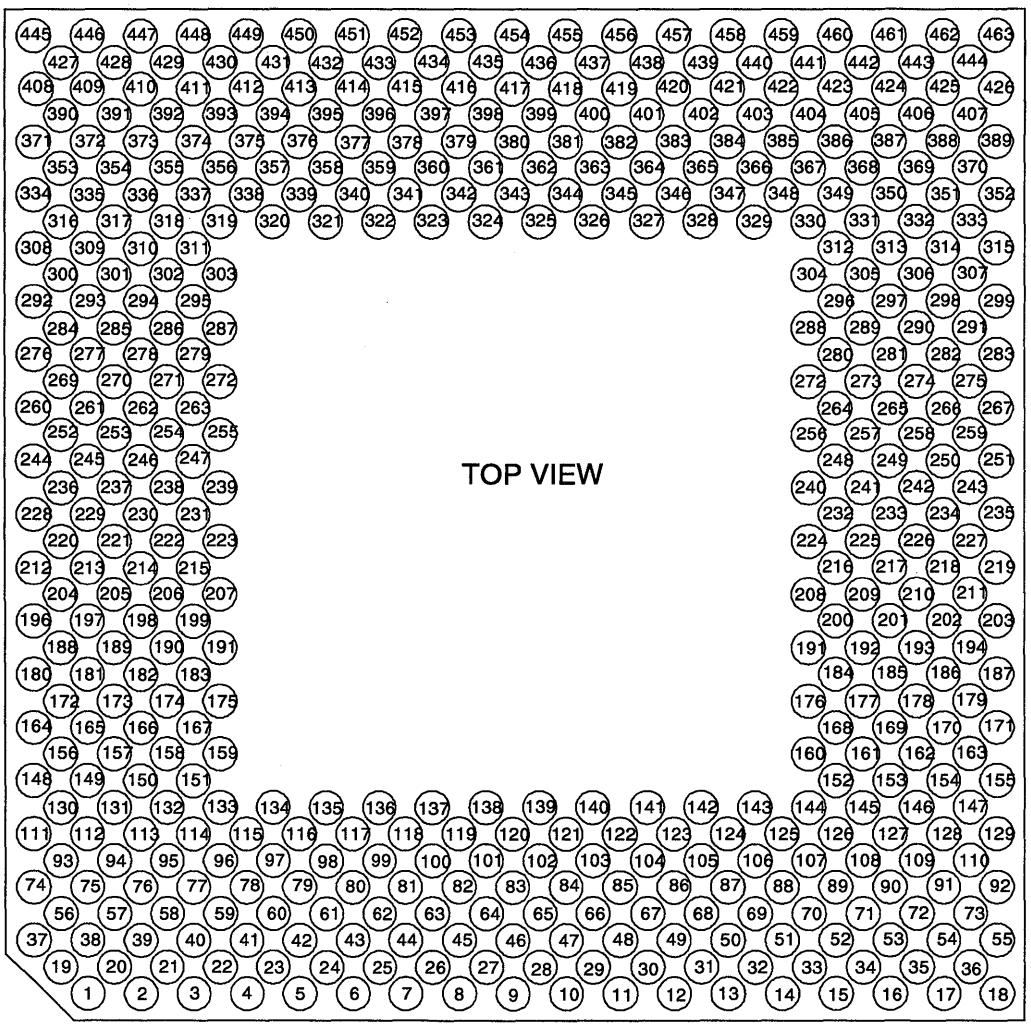
Pin	Type	Signal
225	I/O	CDATA<50>
226	I	VCC4
227	I/O	CDATA<49>
228	I/O	NPDATA<21>
229	I	VSS
230	I/O	NPDATA<11>
231	I/O	NPDATA<47>
232	I/O	CDATA<60>
233	I/O	CDATA<63>
234	I	VSS
235	-	NC
236	I/O	NPDATA<12>
237	I	VCC4
238	I/O	NPDATA<50>
239	I/O	NPDATA<60>
240	I/O	CDATA<56>
241	I/O	CDATA<61>
242	I	VCC4
243	I/O	CDATA<62>
244	I/O	NPDATA<0>
245	I	VSS
246	I/O	NPDATA<43>
247	I/O	NPDATA<39>
248	I/O	CDATA<59>
249	I/O	CDATA<58>
250	I	VSS
251	I/O	CDATA<57>
252	I/O	NPDATA<61>
253	I	VCC4
254	I/O	NPDATA<1>
255	I/O	NPDATA<10>
256	-	NC
257	I/O	NxAD<33>
258	I	VCC4
259	I/O	NxAD<32>
260	I/O	NPDATA<23>
261	I	VSS
262	I/O	NPDATA<20>
263	I/O	NPDATA<31>
264	I/O	PULLLOW
265	I/O	NxAD<34>
266	I	VSS
267	I/O	NxAD<1>
268	I/O	NPDATA<30>
269	I	VCC4
270	I/O	NPDATA<52>
271	I/O	NPDATA<51>
272	I	PTEST
273	I/O	NxAD<36>
274	I	VCC4
275	I/O	NxAD<35>
276	I/O	NPDATA<48>
277	I	VSS
278	I/O	NPDATA<44>
279	I/O	NPDATA<49>
280	O	SERIALOUT
281	I/O	NxAD<10>
282	I	VSS
283	I/O	NxAD<11>
284	I/O	NPDATA<55>

Pin	Type	Signal
285	I	VCC4
286	I/O	NPDATA<6>
287	I/O	NPDATA<36>
288	-	NC
289	I/O	NxAD<41>
290	I	VCC4
291	I/O	NxAD<42>
292	I/O	NPDATA<2>
293	I	VSS
294	I/O	NPDATA<5>
295	I/O	NPDATA<22>
296	I/O	NxAD<0>
297	I/O	NxAD<3>
298	I	VSS
299	I/O	NxAD<40>
300	I/O	NPDATA<35>
301	I	VCC4
302	I/O	NPDATA<34>
303	I/O	NPDATA<28>
304	I/O	NxAD<38>
305	I/O	NxAD<43>
306	I	VCC4
307	I/O	NxAD<2>
308	I/O	NPDATA<41>
309	I	VSS
310	I/O	NPDATA<29>
311	I/O	NPDATA<58>
312	I/O	NxAD<7>
313	I/O	NxAD<8>
314	I	VSS
315	I/O	NxAD<9>
316	I/O	NPDATA<53>
317	I	VCC4
318	I/O	NPDATA<19>
319	I	NPWVAL
320	I/O	PULLHIGH
321	O	NREQ*
322	I	GTAL
323	I	IREF
324	I	VDDA
325	I/O	NxAD<20>
326	I/O	NxAD<54>
327	I/O	NxAD<22>
328	I/O	NxAD<57>
329	I/O	NxAD<56>
330	I	GALE
331	I/O	NxAD<18>
332	I	VCC4
333	I/O	NxAD<19>
334	I/O	NPDATA<59>
335	I	VSS
336	I/O	NPDATA<4>
337	I	NPNOERR
338	I/O	NPDATA<46>
339	I	GATEA20
340	O	AREQ*
341	O	LOCK*
342	O	XPH2
343	I/O	NxAD<52>
344	I/O	NxAD<53>

Pin	Type	Signal
345	I/O	NxAD<21>
346	I/O	NxAD<27>
347	I/O	NxAD<24>
348	I/O	NxAD<15>
349	I	GXACK
350	I/O	NxAD<50>
351	I	VSS
352	I/O	NxAD<51>
353	I/O	NPDATA<63>
354	I	VCC4
355	I	NPWREQ
356	I/O	NPDATA<32>
357	I/O	PULLHIGH
358	O	XACK*
359	O	DCL*
360	O	XPH1
361	I	CKMODE
362	I	RESETCPU*
363	I/O	NxAD<62>
364	I/O	NxAD<60>
365	I/O	NxAD<58>
366	I/O	NxAD<45>
367	I/O	NxAD<46>
368	I	GNT*
369	I	VCC4
370	I/O	NxAD<17>
371	I/O	NPDATA<54>
372	I	VSS
373	I/O	NPDATA<42>
374	I	TESTPWR*
375	I	INTR*
376	I/O	PULLHIGH
377	I	GXHLD
378	I	GBLKNBL
379	I	PHE2
380	-	NC
381	I/O	NxAD<63>
382	I/O	NxAD<29>
383	I/O	NxAD<23>
384	I/O	NxAD<25>
385	I/O	NxAD<47>
386	O	XBCKE*
387	I/O	NxAD<16>
388	I	VSS
389	I/O	NxAD<49>
390	I/O	NPDATA<16>
391	I	VCC4
392	I	VCC4
393	I	VCC4
394	I	VCC4
395	I	VCC4
396	I	VCC4
397	I	VCC4
398	I	VCC4
399	I	VCC4
400	I	VCC4
401	I	VCC4
402	I	VCC4
403	I	VCC4
404	I	VCC4

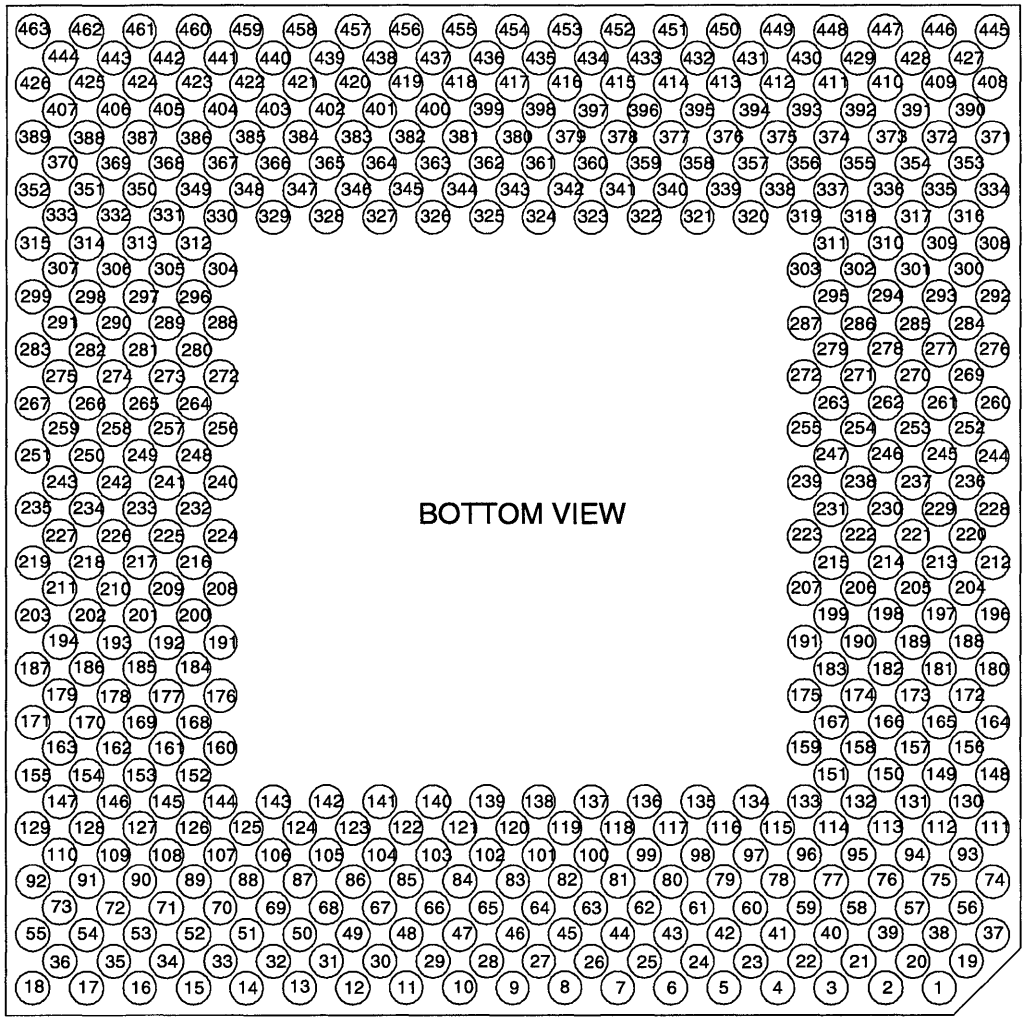
Pin	Type	Signal
405	I	VCC4
406	I	VCC4
407	I/O	NxAD<48>
408	I/O	NPDATA<3>
409	I	VSS
410	I	VSS
411	I	VSS
412	I	VSS
413	I	VSS
414	I	VSS
415	I	VSS
416	I	VSS
417	I	VSS
418	I	VSS
419	I	VSS
420	I	VSS
421	I	VSS
422	I	VSS
423	I	VSS
424	I	VSS
425	I	VSS
426	I/O	NxAD<39>
427	I/O	NPDATA<9>
428	I/O	NPDATA<26>
429	I	GDCL
430	I	GSHARE
431	I/O	PULLHIGH
432	I/O	PULLHIGH
433	I/O	PULLHIGH
434	O	XREF
435	I	XSEL
436	I	NMI*
437	I/O	NxAD<30>
438	I/O	NxAD<28>
439	I/O	NxAD<59>
440	I/O	NxAD<44>
441	I/O	NxAD<14>
442	O	XNOE*
443	I/O	NxAD<4>
444	I/O	NxAD<5>
445	I/O	NPDATA<24>
446	O	NPIRQ*
447	I	OWNABL
448	O	SHARE*
449	O	ALE*
450	I/O	PULLHIGH
451	I/O	PULLHIGH
452	I	CLK
453	I	PHE1
454	O	XHLD*
455	I/O	NxAD<31>
456	I/O	NxAD<61>
457	I/O	NxAD<55>
458	I/O	NxAD<26>
459	I/O	NxAD<12>
460	I/O	NxAD<13>
461	O	XBOE*
462	I/O	NxAD<37>
463	I/O	NxAD<6>

Figure 5 Nx586 Pin List, By Pin Number (continued)



068

Figure 6 Nx586 Pinout Diagram (Top View)



069

Figure 7 Nx586 Pinout Diagram (Bottom View)

Nx586 NexBus Signals

Note: The resistor value required for all signals to be pulled up or down should be in the range between 1kΩ and 5kΩ. The pull up resistor must be connected to the V_{CC} (4V) plane.

NexBus Arbitration

NREQ*	O	**NexBus Request**—Asserted by the processor to the NexBus Arbiter to secure control of the NexBus. This signal remains active until one CLK period after GALE* is received from the NexBus Arbiter. During speculative reads, the Nx586 may deactivate NREQ* before GNT* is received if the transfer is no longer needed. In systems using the NxVL as the NexBus Arbiter, NREQ* is treated the same as AREQ*; when the NexBus control is granted, control of all other buses is also granted at the same time. If the processor does not know which bus its intended resource is on, it asserts NREQ*. If a GTAL is subsequently returned, the processor assumes the resources are on another system bus and it retries the transfer by asserting AREQ*.
AREQ*	O	**Alternate-Bus Request**—Asserted by the processor to the NexBus Arbiter to secure control of the NexBus and any other buses (called *alternate buses)* supported by the system. This signal remains active until GNT* is received from the NexBus Arbiter; unlike NREQ*, the processor does not make speculative requests with AREQ*. The NexBus Arbiter does not issue GNT* until the other system buses are available. In systems using the NxVL as the NexBus Arbiter (shown in Figure 18), AREQ* and NREQ* have the same effect: either one causes the NxVL global bus arbiter to grant all buses to the winning requester at the end of the current bus cycle.
GNT*	I	**Grant NexBus**—Asserted by the NexBus Arbiter to indicate that the processor has been granted control of the NexBus.

LOCK*	O	**Bus Lock**—Asserted by the processor to the NexBus Arbiter when multiple bus operations should be performed sequentially and uninterruptedly. This signal is used by the NexBus Arbiter to determine the end of a bus sequence. Cache-block fills are not locked; they are implicitly treated as atomic reads. Some NexBus Arbiters (but not the NxVL) may allow masters on system buses other than NexBus (i.e., on an *alternate bus)* to intervene in a locked NexBus transaction. To avoid this, the processor must assert AREQ*. LOCK* is typically software configured to be asserted for read-modify-writes and explicitly locked instructions.
SLOTID<3:0>	I	**NexBus Slot ID**—These bits identify NexBus backplane slots. SLOTID 1111 (0Fh) is reserved for the system's primary processor. Normally, only the primary processor receives PC-compatible signals such as RESET*, RESETCPU*, INTR*, NMI*, and GATEA20, and this processor is responsible for initializing any secondary processors. SLOTID 0000 is reserved for the system logic that interfaces the NexBus to any other system buses (called the *alternate-bus interface).* The NxVL acts as an Alternate-Bus Interface. This signal is asynchronous to the NexBus clock.

NexBus Cycle Control

ALE*	O	**Address Latch Enable**—Asserted by the processor to backplane logic or to the system-logic interface between the NexBus and any other system buses (called the *alternate-bus interface)* when the processor is driving valid address and status information on the NxAD<63:0> bus.
GALE	I	**Group Address Latch Enable**—Asserted by a backplane NAND of all ALE* signals, to indicate that the NexBus address and status can be latched. Systems using the NxVL, GALE is generated by the NxVL.
GTAL	I	**Group Try Again Later**—Asserted by the system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* to indicate that the attempted bus-crossing operation cannot be completed, because the system-logic bus interface is busy or cannot access the other system bus. In response, the processor aborts its current operation and attempts to re-try it by asserting AREQ*, thereby assuring that the processor will not receive a GNT* until the desired system bus is available. A bus-crossing operation can happen without the system-logic bus interface asserting GTAL and without the processor asserting AREQ*, if the other system bus and its system-logic interface are both available when the processor asserts NREQ*. The GTAL and AREQ* protocol is only used when NREQ* is asserted while either the other system bus or its system-logic interface is unavailable. The protocol prevents deadlocks and prevents the processor from staying on the NexBus until the other system bus becomes available. Unlike other group signals, which are the logical OR of a set of active-low signals generated by each participating device in the group, GTAL does not have such a corresponding active-low signal.
XACK*	O	**Transfer Acknowledge**—This signal is driven active by the processor during a NexBus snoop cycle (Alternate Bus Master cycle), when the processor determines that it has data from the snooped address.

GXACK	I	**Group Transfer Acknowledge**—Asserted by a backplane NAND of all XACK* signals, to indicate that a NexBus device is prepared to respond as a slave to the processor's current operation. The system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* monitors the XACK* responses from all adapters. In systems using the NxVL as the Alternate-Bus Interface, when no XACK* response is forthcoming within three clocks, the NxVL asserts GXACK and initiates a *bus-crossing operation.* GXACK must be asserted for the transaction to continue. In general, since the system-logic interface to other system buses may take a variable number of cycles to respond to a GALE, the maximum time between assertion of GALE and the responding assertion of GXACK is not specified.
XHLD*	O	**Transfer Hold**—Asserted by the processor, as slave or master, to backplane logic or to the system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* in response to another NexBus master's request for data, when the processor is unable to respond on the next clock after GXACK. In case the processor is the master, an inactive XHLD* indicates that the CPU is not ready to complete the transfer.
GXHLD	I	**Group Transfer Hold**—Asserted by a backplane NAND of all XHLD* signals, to indicate that a slave cannot respond to the processor's request. GXHLD causes wait states to be inserted into the current operation. Both the master and the slave must monitor GXHLD to synchronize data transfers. During a bus-crossing read by the processor, the simultaneous assertion of GXACK and negation of GXHLD indicates that valid data is available on the bus. During a bus-crossing write, the same signal states indicate that data has been accepted by the slave.

NexBus Cache Control

DCL*	O	**Dirty Cache Line**—During reads by another NexBus master, this signal is asserted by the processor to indicate that the location being accessed is cached by the processor's L2 cache in a *modified* (dirty) state. The requesting master's cycle is then aborted so that the processor, as an intervenor, can preemptively gain control of the NexBus and write back its modified data to main memory. While the data is being written to memory, the requesting master reads it off the NexBus. The assertion of DCL* is the only way in which atomic 32-byte cache-block fills by another NexBus master can be preempted by the processor for the purpose of writing back dirty data. During writes by another NexBus master, this signal is likewise asserted by the processor to indicate that it has a *modified* copy of the data. But in this case, the initiating master is allowed to finish its write to memory. The NexBus Arbiter must then guarantee that the processor asserting DCL* gains access to the bus in the very next arbitration grant, so that the processor can write back all of its modified data *except* the bytes written by the initiating master. (In this case, the initiating master's data is more recent than the data cached by the processor asserting DCL*.)
GDCL	I	**Group Dirty Cache Line**—Asserted by a backplane NAND of all DCL* signals, to indicate that a NexBus device has, in its cache, a *modified* copy of the data being accessed. During reads, when the processor is the bus master, the processor aborts its cycle so that the other caching device can write back its data; the processor reads the data on the fly. During writes, when the processor is the bus master, the processor finishes its write before the device asserting DCL* writes back all bytes *other than* those written by the processor.
GBLKNBL	I	**Group Block (Burst) Enable**—Asserted by a memory slave to enable burst transfers, and to indicate that the addressed space may be cached. Paged devices (such as video adapters) and any other devices that cannot support burst transfers or whose data is non-cacheable should negate this signal. I.e. the NxVL system controller will deassert this signal on all alternate bus transfers.

OWNABL	I	**Ownable**—Asserted by the system logic during accesses by the processor to locations that may be cached in the *exclusive* state. Negated during accesses that may only be cached in the *shared* state, such as bus-crossing accesses to an address space that cannot support the MESI cache-coherency protocol. All NexBus addresses are assumed to be cacheable in the *exclusive* state. The OWNABL signal is provided in case the system logic needs to restrict caching to certain locations. In single-processor systems using the NxVL, that does not have an OWNABL signal and the processor's OWNABL input is typically tied high for write-back configurations to allow caching in the *exclusive* state on all reads.
SHARE*	O	**Shared Data**—Asserted by the processor during block reads by another NexBus master to indicate to the other master that its read hit in a block cached by the processor.
GSHARE	I	**Group Shared Data**—Asserted by a backplane NAND of all SHARE* signals, to indicate that the data being read must be cached in the *shared* state, if OWN* (NxAD<49>) is negated. However, if GSHARE and OWN* are both negated during the read, the data may be promoted to the *exclusive* state, since no other NexBus device has declared via SHARE* that it has cached a copy. Instruction fetches are always *shared.*

NexBus Transceivers

XBCKE*	O	**Transceiver NxAD-Bus Clock Enable**—Asserted by the processor to clock registered transceivers and latch addresses and data from the NxAD<63:0> bus for subsequent driving onto the AD<63:0> bus (see Figure 18). There is no comparable clock-enable for the NexBus side of these transceivers; they are always enabled on the NexBus side. In systems using the NxVL as the interface to other system buses, these NexBus transceivers are emulated within the NxVL, and this signal is tied to the same-named input on the NxVL.
XBOE*	O	**Transceiver-to-NxAD-Bus Output Enable**—Asserted by the processor to enable the registered transceivers and drive addresses and data onto the NxAD<63:0> bus from the AD<63:0> bus (see Figure 19). In systems using the NxVL as the interface to other system buses, these transceivers are emulated within the NxVL , and this signal is tied to the same-named input on the NxVL .
XNOE*	O	**Transceiver-to-NexBus Output Enable**—Asserted by the processor to enable registered transceivers and drive addresses and data onto the AD<63:0> bus from the NxAD<63:0> bus (see Figure 19). In systems using the NxVL as system controller, this signal is left unconnected.

NexBus Address and Data

NxAD<63:0>	I/O	**NexBus Address and Status, or Data**—This bus multiplexes address and status information during the "address and status phase" (see Figure 8) and with up to 64 bits of data during a subsequent "data phase". The address and status is valid when GALE is asserted. At that time, address NxAD<63:32> and status NxAD<31:0> is latched. The meanings of these fields are detailed immediately below. The data phase occurs on the cycle after GXACK is asserted and GXHLD is simultaneously negated. To avoid contention, the two phases are separated by a guaranteed dead cycle (a minimum of one clock) which occurs between the assertion of GALE and the assertion of GXACK.

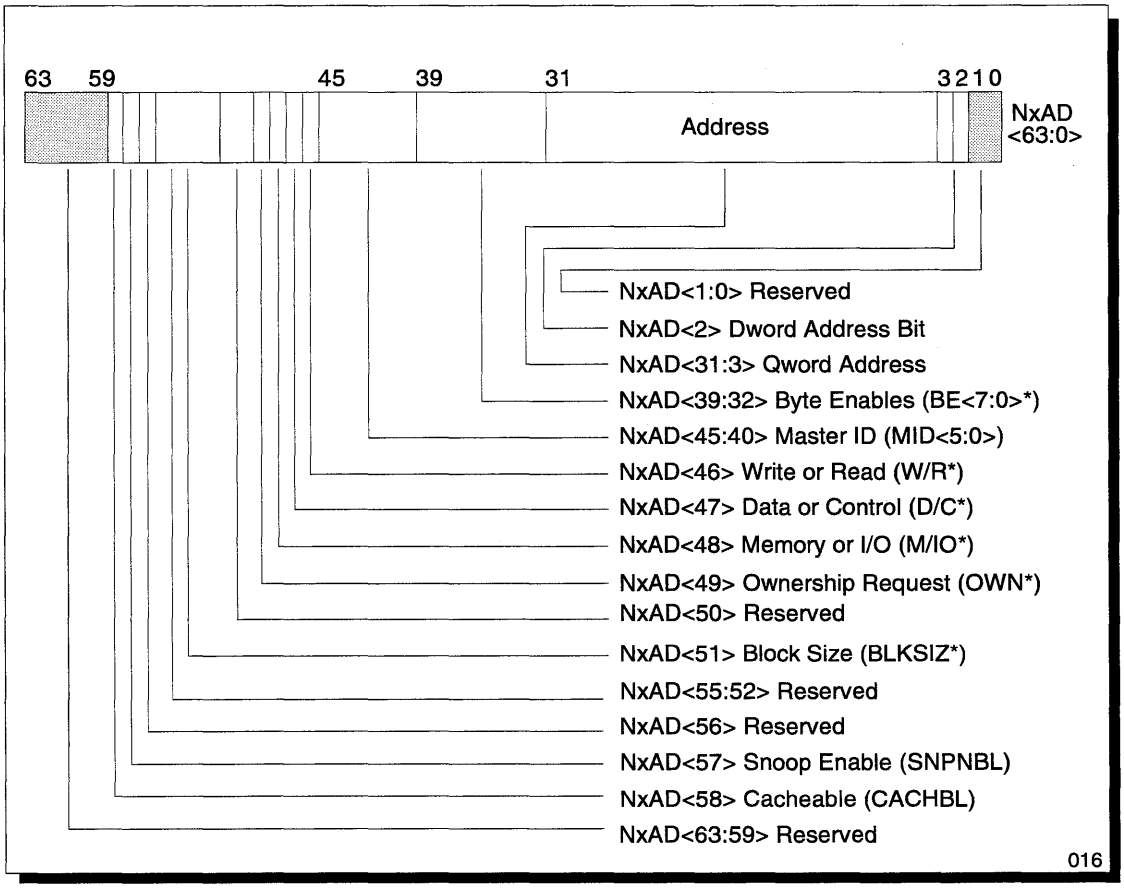


Figure 8 NexBus Address and Status Phase

NxAD<1:0> *address phase*	I/O	**Reserved**—These bits must be driven high by the bus master.
NxAD<2> *address phase*	I/O	**ADRS<2> (Dword Address)**—For I/O cycles, this bit selects between the four-byte doublewords (dwords) in an eight-byte quadword (qword). For memory cycles, the bit is driven but the information is not normally used.
NxAD<31:3> *address phase*	I/O	**ADRS<31:3> (Qword Address)**—For memory cycles, these bits address an eight-byte quadword *(qword)* within the 4GB memory address space. For I/O cycles, NxAD<15:3> specifies a qword within the 64kB I/O address space and NxAD<31:16> are driven low by the processor. In either case, the addressed data may be further restricted by the BE<7:0>* bits on NxAD<39:32>. Memory cycles (but not I/O cycles) may be expanded to additional consecutive qwords by the BLKSIZ<1:0>* bits on NxAD<51:50>.
NxAD<39:32> *address phase*	I/O	**BE<7:0>* (Byte Enables)**—Byte-enable bits for the data phase of the NxAD<63:0> bus. BE<0>* corresponds to the byte on NxAD<7:0>, and BE<7>* corresponds to the byte on NxAD<63:56>. The meaning of these bytes is shown in Figure 9 and 10. For I/O cycles, BE<3:0>* specify the bytes to be transferred on NxAD<31:0> and BE<7:4>* are driven high by the processor. For memory cycles, all eight bits are used to specify the bytes to be transferred on NxAD<63:0>.

Transfer Type	*Meaning of BE<7:0>**
I/O	BE<3:0>* specify the bytes to transfer on NxAD<31:0>. BE<7:4>* are driven high by the processor.

Figure 9 Byte-Enable Usage during I/O Transfers

Transfer Type		*Meaning of BE<7:0>**
Memory	Single Qword Read or Write	BE<7:0>* specify the bytes to transfer on NxAD<63:0>.
	Four-Qword Block Write	BE<7:0>* specify the bytes to transfer on NxAD<63:0> for first qword only. For all other qwords, BE<3:0>* are implicit zeros. and all bytes are transferred.
	Four-Qword Block Read (Cache-Block Fill)	BE<7:0>* specify the bytes that are to be fetched immediately.

Figure 10 Byte-Enable Usage during Memory Transfers

NxAD<45:40> *address phase*	I/O	**MID<5:0> (Master ID)**—These bits indicate to a slave, and to the system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* during bus-crossing cycles, the identity of the NexBus master that initiated the cycle. The most-significant four bits are the device's SLOTID<3:0> bits. The least-significant two bits are the device's DEVICE<1:0> bits. In systems using the NxVL as the interface to other system buses, MID 000000 is reserved for the NxVL .

NxAD<46> *address phase*	I/O	**W/R* (Write or Read*)**—This bit distinguishes between read and write operations on the NexBus. Bus cycle types are interpreted as shown in Figure 11.
NxAD<47> *address phase*	I/O	**D/C* (Data or Code*)**—This bit distinguishes between data and code operations on the NexBus. Bus cycle types are interpreted as shown in Figure 11.
NxAD<48> *address phase*	I/O	**M/IO* (Memory or I/O*)**—This bit distinguishes between memory and I/O operations on the NexBus. Bus cycle types are interpreted as shown in Figure 11.

NxAD<48> *M/IO**	*NxAD<47>* *D/C**	*NxAD<46>* *W/R**	*Type of Bus Cycle*
0	0	0	Interrupt Acknowledge
0	0	1	Halt or Shutdown
0	1	0	I/O Data Read
0	1	1	I/O Data Write
1	0	0	Memory Code Read
1	0	1	*(reserved)*
1	1	0	Memory Data Read
1	1	1	Memory Data Write

Figure 11 Bus-Cycle Types

NxAD<49> *address phase*	I/O	**Ownership Request**—Asserted by a master when it intends to cache data in the *exclusive* state. The bit is asserted for write-backs and reads from the stack. If such an operation hits in the cache of another master, that master writes its data back (if copy is modified) and changes the state of its copy to *invalid*. If OWN* is negated during a read or write, another master may not assume that the copy is in *shared* state when not asserting SHARE* signal.
NxAD<50> *address phase*	I/O	**Reserved**—These bit must be driven high.

NxAD<51> *address phase*	I/O	**BLKSIZ* (Block Size)**—For memory operations, this bit defines the number of transfers. It is low for four-qword transfers and high for single byte, word, dword or qword cycles. For I/O operations, this bit is also driven high by the processor. For single transfers and block (burst) writes, the bytes to be transferred in the first qword are specified by the byte-enable bits, BE<7:0>* on NxAD<39:32>. If the slave is incapable of transferring more than a single qword, it or the system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* may deny a request for subsequent qwords by negating the GXACK or GBLKNBL inputs to the processor after a single-qword transfer, or after returning all bytes specified by BE<7:0>* in the first qword.
NxAD<56:52> *address phase*	I/O	**Reserved**—These bits must be driven high.
NxAD<57> *address phase*	I/O	**SNPNBL (Snoop Enable)**—Asserted to indicate that the current operation affects memory that may be present in other caches. When this signal is negated, snooping devices need not look up the addressed data in their cache tags.
NxAD<58> *address phase*	I/O	**CACHBL (Cacheable)**—Asserted by the bus master to indicate that it may cache a copy of the addressed data. The master typically decides what it will cache, based on software-configured address ranges. This bit supports higher-performance designs by letting the NexBus interface know what the master intends to do with the data, thereby allowing other devices to sometimes prevent unnecessary invalidations or write-backs.
NxAD<63:59> *address phase*	I/O	**Reserved**—These bits must be driven high by the bus master.

Nx586 L2 Cache Signals

COEA* **COEB***	O	**L2 Cache Output Enable A,B**—Enables reading from second-level cache SRAMs to drive the CDATA<63:0> bus. Standard asynchronous static RAMs are used for this cache. Each signal should be connected to a maximum of four devices for a total of eight RAM devices. Both signals are driven simultaneously.
CWE<7:0>*	O	**L2 Cache Write Enable**—Enables writing to the second-level cache SRAMs. The CWE<0>* bit enables writing the byte on CDATA<7:0>. The CWE<7>* bit enables writing the byte on CDATA<63:56>.
CBANK<1:0>	O	**L2 Cache Bank**—Selects one of four banks (sets) in the four-way set associative second-level cache. Each bank is either 64kB or 256kB. These signals should be connected to the two least-significant address bits of the SRAM s.
CADDR<17:3>	O	**L2 Cache Address**—The address of an eight-byte quantity in the second-level cache bank selected by CBANK<1:0>. Bits 17:16 are not used for a 256kB L2 cache; they are only used for a 1MB cache.
CDATA<63:0>	I/O	**L2 Cache Data**—Carries either one to eight bytes of second-level cache data, or the tags and state bits for one to four second-level cache banks (sets). Transfers on this bus occur at the peak rate of eight bytes every two processor clocks, but the transfers can begin on any processor clock.

Floating Point-Coprocessor Bus Signals (on Nx586)

NPIRQ*	O	**Reserved**—This signal should be connected to the same-named signal on the Nx587 Floating Point Coprocessor. It is reserved for future use.
NPPOPBUS<15:0>	O	**Floating Point Coprocessor Micro-Operations Bus**—Driven by the Nx586 processor to the Nx587 Floating Point Coprocessor to provide a floating-point micro-operation at the peak rate of one per processor clock. The NPPOPBUS<15:0> bus carries both micro-operations and their associated tags, both of which are issued by the Nx586 processor's Decode Unit.
NPNOERR	I	**Floating Point Coprocessor No Error**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor for handshaking to implement the IBM-compatible mode of interrupt handling. This signal is enabled and disabled in software. The signal must be pulled up.
NPPOPTAG<4:0>	I/O	**Reserved**—These signals must be connected to the same-named signals on the Nx587 Floating Point Coprocessor, if the latter is used. Otherwise, the signals must be left unconnected.
NPOUTFTYP<1:0>	O	**Floating Point Coprocessor Output Type**—Asserted by the Nx586 processor to the Nx587 Floating Point Coprocessor for handshaking to implement the IBM-compatible mode of interrupt handling. These signals are enabled and disabled in software.
NPTERM<1:0>	I	**Floating Point Coprocessor Termination**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor to indicate completion of floating-point operations. This signal must be pulled up.
NPTAGSTAT<5:0>	O	**Floating Point Coprocessor Tag Status**—Driven by the Nx586 processor to the Nx587 Floating Point Coprocessor to synchronize the issuing, retiring, and aborting of instructions.
NPRVAL	O	**Floating Point Coprocessor Read Valid**—Asserted by the Nx586 processor to the Nx587 Floating Point Coprocessor in the clock following a successful request, to indicate that the data being transferred on the NPDATA<63:0> bus in the current clock is valid.

NPRREQ	O	**Floating Point Coprocessor Read Request**—Asserted by the Nx586 processor to the Nx587 Floating Point Coprocessor, to request use of the NPDATA<63:0> and NPTAG<4:0> buses to transfer data on the next clock. The NPRREQ signal has priority over the NPWREQ signal. When neither is requesting, the processor drives the bus. The processor sometimes makes speculative requests, such as when it concurrently does cache lookups for the data to be transferred. If the processor finds that it cannot use the bus after requesting it, it negates NPRVAL when the bus is granted, otherwise it asserts NPRVAL and transfers the data in the same clock.
NPWREQ	I	**Floating Point Coprocessor Write Request**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor, to request control of the NPDATA<63:0> and NPTAG<4:0> buses to transfer data on the next clock. The NPRREQ signal has priority over the NPWREQ signal. The signal must be pulled down. The Floating Point Coprocessor makes speculative requests concurrently with its first pass at formatting the output. If it discovers that more formatting is needed, it negates NPWVAL when the NPDATA<63:0> bus is granted, otherwise it asserts NPWVAL and transfers the data in the same clock.
NPWVAL	I	**Floating Point Coprocessor Write Valid**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor in the clock following a successful request, to indicate that the data being transferred on the NPDATA<63:0> bus in the current clock is valid. This signal must be pulled down.
NPTAG<4:0>	I/O	**Floating Point Coprocessor Tag Bus**—On each processor clock, this bus carries the five-bit micro-operation tag between the Nx586 processor and the Nx587 Floating Point Coprocessor. The tag identifies the instruction from which the micro-operation was decoded, and it corresponds to the data being transferred on the NPDATA<63:0> bus.

NPDATA<63:0>	I/O	**Floating Point Coprocessor Data**—On each processor clock, this bus carries up to 64 bits of read or write data between the Nx586 processor and the Nx587 Floating Point Coprocessor. The Nx586 processor uses it to provide read data to the Nx587 Floating Point Coprocessor, and the Nx587 Floating Point Coprocessor uses it to write results. The bi-directionality of the bus is implemented with arbitration among the NPRREQ and NPWREQ signals. Arbitration priority is given to the processor, hence reads prevail over writes. The winner gets the bus on the next clock. The arbitration and the bus transfer are pipelined one clock apart at the processor-clock frequency. Thus, in every clock, both a request and a transfer are made.

Nx586 System Signals

Nx586 Clock

CLK	I	**NexBus Clock**—A TTL-level clock with a duty cycle between 45% and 55%. All signals on the NexBus transition on the rising edge of CLK, except the asynchronous signals, INTR*, NMI*, GATEA20, and SLOTID<3:0>. The processor's internal phase-locked loop (PLL) synchronizes internal processor clocks at twice the frequency of CLK.
PHE1	I	**Clock Phase 1**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
PHE2	I	**Clock Phase 2**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
CKMODE	I	**Clock Mode**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
XSEL	I	**Clock Mode Select**—For normal clocking operation, this signal should be tied low. Refer to Figure 27.
XPH1	O	**Processor Clock Phase 1**—For normal clocking operation, this signal must be left unconnected. Refer to Figure 27.
XPH2	O	**Processor Clock Phase 2**—For normal clocking operation, this signal must be left unconnected. Refer to Figure 27.
IREF	I	**Clock Input Reference**—This signal must be pulled up to V_{DDA} with a 220kΩ resistor.
XREF	O	**Clock Output Reference**—For normal clocking operation, this signal must be left unconnected.
VDDA	I	**PLL Analog Power**—This input provides power for the on chip PLL circuitry and should be isolated from V_{CC} by a ferrite bead and decoupled with a 0.1 µF ceramic capacitor.

Nx586 Interrupts and Reset

INTR*	I	**Maskable Interrupt**—Level sensitive. This signal is asserted by an interrupt controller. The processor responds by stopping its current flow of instructions at the next instruction boundary, aborting earlier instructions that have been partially executed, and performing an interrupt acknowledge sequence, as described in the *Bus Operations* chapter. This signal is asynchronous to the processor and to the NexBus clock.
NMI*	I	**Non-Maskable Interrupt**—Edge sensitive. Asserted by system logic. The effect of this signal is similar to INTR*, except that NMI* cannot be masked by software, the interrupt acknowledge sequence is not performed, and the handler is always located by interrupt vector 2 in the interrupt descriptor table. This signal is asynchronous to the processor and to the NexBus clock.
RESET*	I	**Global Reset (Power-Up Reset)**—Asserted by system logic. The processor responds by resetting its internal state machines and loading default values into its registers. At power-up it must remain asserted for a minimum of several milliseconds to stabilize the phase-locked loop.
RESETCPU*	I	**Reset CPU (Soft Reset)**—Asserted by the system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* to reset the processor without changing the state of memory or the processor's caches. This signal is normally routed only to the primary processor in SLOTID 0Fh; on all other slots, this signal is normally tied high.
GATEA20	I	**Gate Address 20**—When asserted by the system controller or keyboard controller, the processor drives bit 20 of the physical address at its current value. When negated, address bit 20 is cleared to zero, causing the address to wrap around into a 20-bit address space. GATEA20 is asynchronous to the NexBus clock. This method replicates the 8086 processor's handling of address wraparound. All physical addresses are affected by the ANDing of GATEA20 with address bit 20, including cached addresses. This signal is asynchronous to the processor's internal clock and to the NexBus clock (CLK).

Nx586 Test and Reserved Signals

ANALYZEIN	I	**Reserved**—This signal must be pulled low for normal operation.
ANALYZEOUT	O	**Reserved**—This signal must be left unconnected for normal operation.
NC	-	**Reserved**—These signals must be left unconnected.
GREF	O	**Ground Reference**—This signal must be left unconnected for normal operation.
HROM	I	**Reserved**—This signal must be pulled low.
NPSPARE<2:0>	I	**Reserved**—These signals must be connected to the same-named signals on the Nx587 co-processor and pulled low.
P4REF	O	**Power Reference**—This signal must be left unconnected for normal operation.
POPHOLD	I	**Reserved**—This signal must be pulled low for normal operation.
PTEST	I	**Processor TEST**—This pin is to tri-state all outputs except for the following pins: XPH1, XPH2, and XREF. For normal operation, this input must be pulled low.
PULLHIGH	I/O	**Reserved**—These signals must be pulled high to VCC4 for normal operation.
PULLLOW	I/O	**Reserved**—These signals must be pulled low for normal operation.
SERIALIN	O	**Serial In**—The input of the scan-test chain. This signal must be left unconnected for normal operation.
SERIALOUT	O	**Serial Out**—The output of the scan-test chain. This signal must be left unconnected for normal operation.
TESTPWR*	I	**Test Power**—Powers-down CPU's static circuits during scan tests. This signal must be pulled high for normal operation.
TPH1	I	**Test Phase 1 Clock**—For scan test support. This signal must be pulled low for normal operation.
TPH2	I	**Test Phase 2 Clock**—For scan test support. This signal must be pulled low for normal operation.

Nx586 Alphabetical Signal Summary

ALE*	O	Address Latch Enable
ANALYZEIN	I	Analyze In
ANALYZEOUT	O	Analyze Out
AREQ*	O	Alternate-Bus Request
CADDR<17:3>	O	L2 Cache Address
CBANK<1:0>	O	L2 Cache Bank
CDATA<63:0>	I/O	L2 Cache Data
CKMODE	I	Clock Mode
CLK	I	NexBus Clock
COEA*	O	L2 Cache Output Enable A
COEB*	O	L2 Cache Output Enable B
CWE<7:0>*	O	L2 Cache Write Enable
DCL*	O	Dirty Cache Line
GALE	I	Group Address Latch Enable
GATEA20	I	Gate Address 20
GBLKNBL	I	Group Block (Burst) Enable
GDCL	I	Group Dirty Cache Line
GNT*	I	Grant NexBus
GREF	I	Ground Reference
GSHARE	I	Group Shared Data
GTAL	I	Group Try Again Later
GXACK	I	Group Transfer Acknowledge
GXHLD	I	Group Transfer Hold
HROM	I	*Reserved*
INTR*	I	Maskable Interrupt
IREF	I	Clock Input Reference
LOCK*	O	Bus Lock
NC	-	*Reserved*
NMI*	I	Non-Maskable Interrupt
NPDATA<63:0>	I/O	Floating Point Coprocessor Data
NPIRQ*	O	*Reserved*
NPNOERR	I	Floating Point Coprocessor No Error
NPOUTFTYP<1:0>	O	Floating Point Coprocessor Output Type

NPPOPBUS<15:0>	O	Floating Point Coprocessor Micro-Operations Bus
NPPOPTAG<4:0>	I/O	*Reserved*
NPRREQ	O	Floating Point Coprocessor Read Request
NPRVAL	O	Floating Point Coprocessor Read Valid
NPSPARE<2:0>	O	*Reserved*
NPTAG<4:0>	I/O	Floating Point Coprocessor Tag Bus
NPTAGSTAT<5:0>	O	Floating Point Coprocessor Tag Status
NPTERM<1:0>	I	Floating Point Coprocessor Termination
NPWREQ	I	Floating Point Coprocessor Write Request
NPWVAL	I	Floating Point Coprocessor Write Valid
NREQ*	O	NexBus Request
NxAD<63:0>	I/O	Bus Address/Status, or Bus Data
NxADINUSE	O	*Reserved*
OWNABL	I	Ownable
P4REF	O	Power Reference
PARERR*	O	*Reserved*
PHE1	I	Clock Phase 1
PHE2	I	Clock Phase 2
POPHOLD	I	Processor-Operation Hold
PTEST	I	*Reserved*
PULLHIGH	I/O	*Reserved*
PULLLOW	I	*Reserved*
RESET*	I	Global Reset (Power-Up Reset)
RESETCPU*	I	Reset CPU (Soft Reset)
SERIALIN	O	Serial In
SERIALOUT	O	Serial Out
SHARE*	O	Shared Data
SLOTID<3:0>	I	NexBus Slot ID
TESTPWR*	I	Test Power
TPH1	I	Test Phase 1 Clock
TPH2	I	Test Phase 2 Clock
VDDA	I	PLL Analog Power
XACK*	O	Transfer Acknowledge
XBCKE*	O	NexBus-Transceiver Clock Enable

XBOE*	O	NexBus-Transceiver Output Enable
XHLD*	O	Transfer Hold
XNOE*	O	NexBus-Transceiver Output Enable
XPH1	O	Processor Clock Phase 1
XPH2	O	Processor Clock Phase 2
XREF	O	Clock Output Reference
XSEL	I	Clock Mode Select

Nx587 Features and Signals

The NexGen Nx587 floating-point coprocessor is an expansion of the Nx586 superscalar pipelined microarchitecture. It adds specific x86 architecture floating point operations including arithmetic, exponential, logarithmic, and trigonometric functions. The Nx587 is tightly coupled to the Nx586 pipeline to ensure maximum floating-point calculation speed. When installed, the Nx587 resides on it own dedicated bus to obtain on-chip equivalent performance. The following are some of the key features:

- **Binary Compatible**—Runs all x86-architecture floating-point binary code.
- **Optional**—No hardware reconfiguration necessary if not present.
- **Dedicated 64-Bit Processor Bus**—Fast, synchronous, non-multiplexed interface to Nx586 processor.
- **High Bus Bandwidth**—Speculative requests and simple arbitration on the Nx586-Nx587 bus maximize bandwidth. Arbitration and data transfers occur in parallel, one clock apart.
- **Fully Integrated into Nx586 Pipeline**—Operates in parallel with the Nx586 Decode, Address, and Integer Units.
- **Advanced State-of-the-Art Fabrication Process**—0.5 micron CMOS.

Figure 12 shows the signal organization on the Nx587 Floating-Point Coprocessor. These include signals shared with the Nx586 processor, system signals (including an interrupt request signal, NPIRQ*, to an external interrupt controller), and test signals. The signals shared with the Nx586 processor operate at the processor-clock frequency and have the same functionality as those on the processor, but with reverse directionality. The normal state for all reserved bits is high.

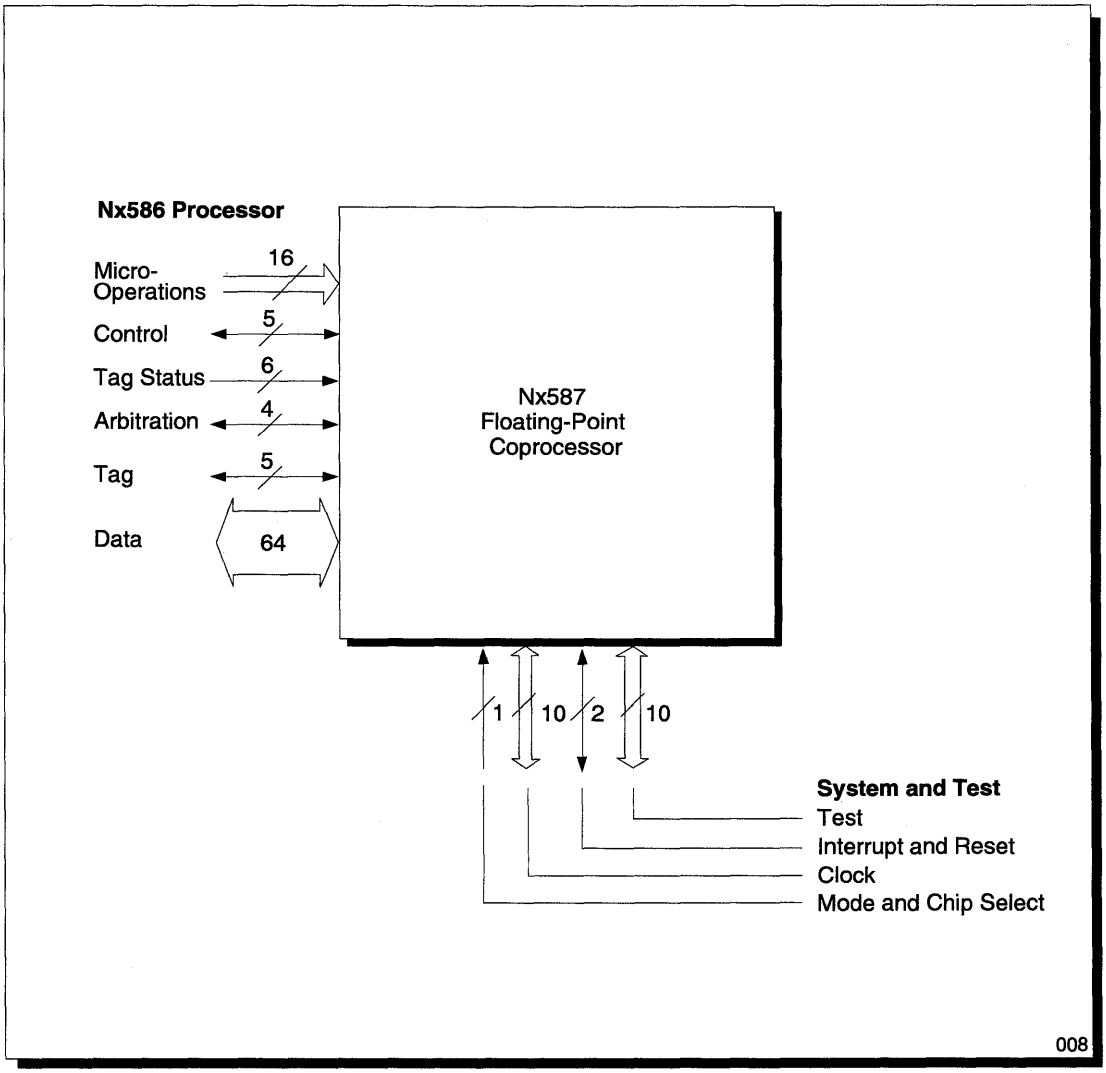


Figure 12 Nx587 Signal Organization

Nx587 Pinouts by Signal Names

Pin	Type	Signal
165	I	CKMODE
179	I	CLK
124	I/O	FPTEST
167	I	IREF
168	-	NC
110	-	NC
142	-	NC
129	-	NC
136	-	NC
140	-	NC
58	I/O	NPDATA<0>
57	I/O	NPDATA<1>
17	I/O	NPDATA<2>
11	I/O	NPDATA<3>
8	I/O	NPDATA<4>
4	I/O	NPDATA<5>
3	I/O	NPDATA<6>
76	I/O	NPDATA<7>
81	I/O	NPDATA<8>
25	I/O	NPDATA<9>
60	I/O	NPDATA<10>
69	I/O	NPDATA<11>
62	I/O	NPDATA<12>
91	I/O	NPDATA<13>
74	I/O	NPDATA<14>
92	I/O	NPDATA<15>
24	I/O	NPDATA<16>
75	I/O	NPDATA<17>
83	I/O	NPDATA<18>
7	I/O	NPDATA<19>
12	I/O	NPDATA<20>
66	I/O	NPDATA<21>
45	I/O	NPDATA<22>
41	I/O	NPDATA<23>
40	I/O	NPDATA<24>
121	I/O	NPDATA<25>
26	I/O	NPDATA<26>
70	I/O	NPDATA<27>
46	I/O	NPDATA<28>
6	I/O	NPDATA<29>
27	I/O	NPDATA<30>
53	I/O	NPDATA<31>
50	I/O	NPDATA<32>
82	I/O	NPDATA<33>
5	I/O	NPDATA<34>
18	I/O	NPDATA<35>

Pin	Type	Signal
44	I/O	NPDATA<36>
90	I/O	NPDATA<37>
84	I/O	NPDATA<38>
59	I/O	NPDATA<39>
85	I/O	NPDATA<40>
19	I/O	NPDATA<41>
10	I/O	NPDATA<42>
61	I/O	NPDATA<43>
1	I/O	NPDATA<44>
89	I/O	NPDATA<45>
51	I/O	NPDATA<46>
67	I/O	NPDATA<47>
43	I/O	NPDATA<48>
2	I/O	NPDATA<49>
65	I/O	NPDATA<50>
14	I/O	NPDATA<51>
15	I/O	NPDATA<52>
20	I/O	NPDATA<53>
23	I/O	NPDATA<54>
16	I/O	NPDATA<55>
86	I/O	NPDATA<56>
73	I/O	NPDATA<57>
47	I/O	NPDATA<58>
21	I/O	NPDATA<59>
68	I/O	NPDATA<60>
54	I/O	NPDATA<61>
78	I/O	NPDATA<62>
22	I/O	NPDATA<63>
93	I	NPRREQ
97	I	NPRVAL
9	O	NPWREQ
48	O	NPWVAL
13	O	NPIRQ*
49	O	NPNOERR
94	I	NPOUTFTYP<0>
176	I	NPOUTFTYP<1>
162	I	NPPOPBUS<0>
134	I	NPPOPBUS<1>
106	I	NPPOPBUS<10>
122	I	NPPOPBUS<11>
161	I	NPPOPBUS<12>
102	I	NPPOPBUS<13>
175	I	NPPOPBUS<14>
132	I	NPPOPBUS<15>
172	I	NPPOPBUS<2>
118	I	NPPOPBUS<3>

Pin	Type	Signal
98	I	NPPOPBUS<4>
163	I	NPPOPBUS<5>
173	I	NPPOPBUS<6>
159	I	NPPOPBUS<7>
171	I	NPPOPBUS<8>
133	I	NPPOPBUS<9>
135	I/O	NPTAG<0>
123	I/O	NPTAG<1>
158	I/O	NPTAG<2>
114	I/O	NPTAG<3>
143	I/O	NPTAG<4>
157	I	NPTAGSTAT<0>
170	I	NPTAGSTAT<1>
100	I	NPTAGSTAT<2>
160	I	NPTAGSTAT<3>
174	I	NPTAGSTAT<4>
101	I	NPTAGSTAT<5>
116	O	NPTERM<0>
115	O	NPTERM<1>
164	O	NPTERM<2>
117	O	NPTERM<3>
125	O	NPTERM<4>
177	O	NPTERM<5>
130	I	NPPOPTAG<0>
108	I	NPPOPTAG<1>
126	I	NPPOPTAG<2>
113	I	NPPOPTAG<3>
107	I	NPPOPTAG<4>
99	-	NPSPARE<0>
109	-	NPSPARE<1>
105	-	NPSPARE<2>
166	I	PHE1
138	I	PHE2
77	I	RESET*
183	I	SERIALIN
182	O	SERIALOUT
169	I	TPH1
156	I	TPH2
30	I	VCC4
32	I	VCC4
103	I	VCC4
104	I	VCC4
119	I	VCC4
120	I	VCC4
131	I	VCC4
141	I	VCC4

Pin	Type	Signal
146	I	VCC4
148	I	VCC4
151	I	VCC4
153	I	VCC4
35	I	VCC4
37	I	VCC4
42	I	VCC4
52	I	VCC4
63	I	VCC4
64	I	VCC4
79	I	VCC4
80	I	VCC4
137	I	VDDA
28	I	VSS
29	I	VSS
71	I	VSS
72	I	VSS
87	I	VSS
88	I	VSS
95	I	VSS
96	I	VSS
111	I	VSS
112	I	VSS
127	I	VSS
128	I	VSS
31	I	VSS
144	I	VSS
145	I	VSS
147	I	VSS
149	I	VSS
150	I	VSS
152	I	VSS
154	I	VSS
155	I	VSS
33	I	VSS
34	I	VSS
36	I	VSS
38	I	VSS
39	I	VSS
55	I	VSS
56	I	VSS
181	O	XPH1
180	O	XPH2
139	I	XREF
178	I	XSEL

Figure 13 Nx587 Pin List, By Signal Name

Nx587 Pinouts by Pin Numbers

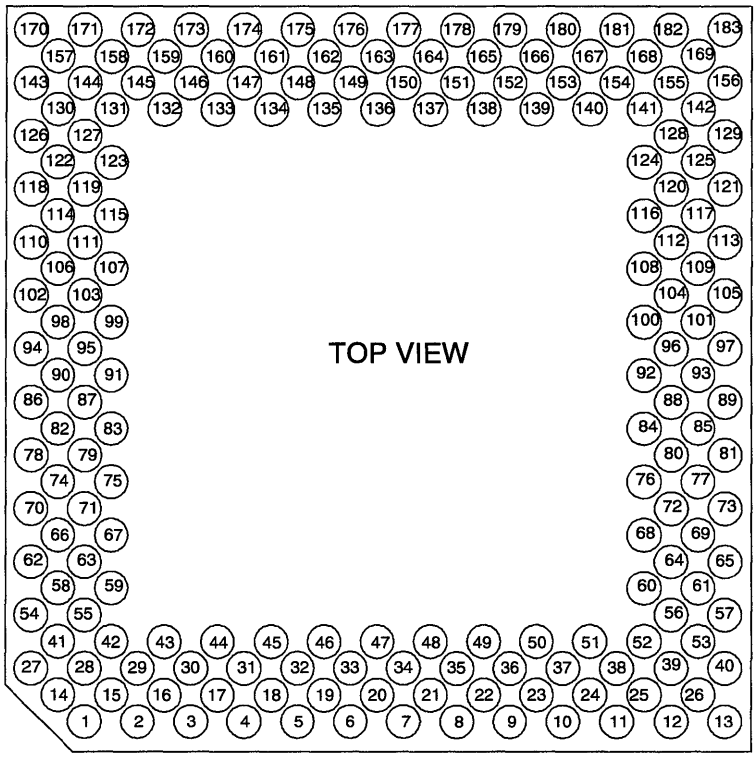
Pin	Type	Signal
1	I/O	NPDATA<44>
2	I/O	NPDATA<49>
3	I/O	NPDATA<6>
4	I/O	NPDATA<5>
5	I/O	NPDATA<34>
6	I/O	NPDATA<29>
7	I/O	NPDATA<19>
8	I/O	NPDATA<4>
9	O	NPWREQ
10	I/O	NPDATA<42>
11	I/O	NPDATA<3>
12	I/O	NPDATA<20>
13	O	NPIRQ*
14	I/O	NPDATA<51>
15	I/O	NPDATA<52>
16	I/O	NPDATA<55>
17	I/O	NPDATA<2>
18	I/O	NPDATA<35>
19	I/O	NPDATA<41>
20	I/O	NPDATA<53>
21	I/O	NPDATA<59>
22	I/O	NPDATA<63>
23	I/O	NPDATA<54>
24	I/O	NPDATA<16>
25	I/O	NPDATA<9>
26	I/O	NPDATA<26>
27	I/O	NPDATA<30>
28	I	VSS
29	I	VSS
30	I	VCC4
31	I	VSS
32	I	VCC4
33	I	VSS
34	I	VSS
35	I	VCC4
36	I	VSS
37	I	VCC4
38	I	VSS
39	I	VSS
40	I/O	NPDATA<24>
41	I/O	NPDATA<23>
42	I	VCC4
43	I/O	NPDATA<48>
44	I/O	NPDATA<36>
45	I/O	NPDATA<22>
46	I/O	NPDATA<28>

Pin	Type	Signal
47	I/O	NPDATA<58>
48	O	NPWVAL
49	O	NPNOERR
50	I/O	NPDATA<32>
51	I/O	NPDATA<46>
52	I	VCC4
53	I/O	NPDATA<31>
54	I/O	NPDATA<61>
55	I	VSS
56	I	VSS
57	I/O	NPDATA<1>
58	I/O	NPDATA<0>
59	I/O	NPDATA<39>
60	I/O	NPDATA<10>
61	I/O	NPDATA<43>
62	I/O	NPDATA<12>
63	I	VCC4
64	I	VCC4
65	I/O	NPDATA<50>
66	I/O	NPDATA<21>
67	I/O	NPDATA<47>
68	I/O	NPDATA<60>
69	I/O	NPDATA<11>
70	I/O	NPDATA<27>
71	I	VSS
72	I	VSS
73	I/O	NPDATA<57>
74	I/O	NPDATA<14>
75	I/O	NPDATA<17>
76	I/O	NPDATA<7>
77	I	RESET*
78	I/O	NPDATA<62>
79	I	VCC4
80	I	VCC4
81	I/O	NPDATA<8>
82	I/O	NPDATA<33>
83	I/O	NPDATA<18>
84	I/O	NPDATA<38>
85	I/O	NPDATA<40>
86	I/O	NPDATA<56>
87	I	VSS
88	I	VSS
89	I/O	NPDATA<45>
90	I/O	NPDATA<37>
91	I/O	NPDATA<13>
92	I/O	NPDATA<15>

Pin	Type	Signal
93	I	NPRREQ
94	I	NPOUTFTYP<0>
95	I	VSS
96	I	VSS
97	I	NPRVAL
98	I	NPPOPBUS<4>
99	I	NPSPARE<0>
100	I	NPTAGSTAT<2>
101	I	NPTAGSTAT<5>
102	I	NPPOPBUS<13>
103	I	VCC4
104	I	VCC4
105	I	NPSPARE<2>
106	I	NPPOPBUS<10>
107	I	NPPOPTAG<4>
108	I	NPPOPTAG<1>
109	I	NPSPARE<1>
110	I/O	NC
111	I	VSS
112	I	VSS
113	I	NPPOPTAG<3>
114	I/O	NPTAG<3>
115	O	NPTERM<1>
116	O	NPTERM<0>
117	O	NPTERM<3>
118	I	NPPOPBUS<3>
119	I	VCC4
120	I	VCC4
121	I/O	NPDATA<25>
122	I	NPPOPBUS<11>
123	I/O	NPTAG<1>
124	I/O	FPTEST
125	O	NPTERM<4>
126	I	NPPOPTAG<2>
127	I	VSS
128	I	VSS
129	I/O	NC
130	I	NPPOPTAG<0>
131	I	VCC4
132	I	NPPOPBUS<15>
133	I	NPPOPBUS<9>
134	I	NPPOPBUS<1>
135	I/O	NPTAG<0>
136	I/O	NC
137	I	VDDA
138	I	PHE2

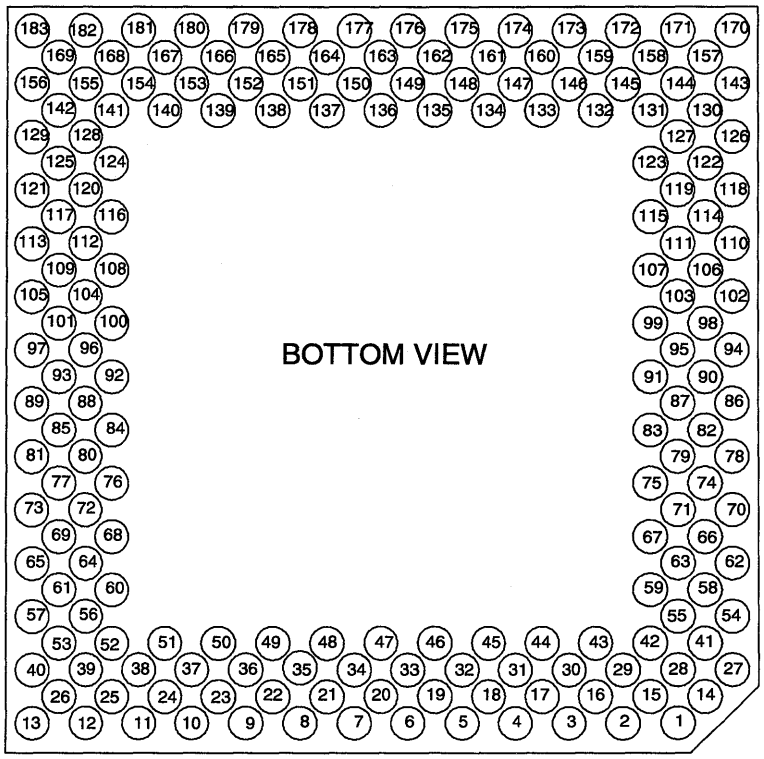
Pin	Type	Signal
139	I	XREF
140	I/O	NC
141	I	VCC4
142	I/O	NC
143	I/O	NPTAG<4>
144	I	VSS
145	I	VSS
146	I	VCC4
147	I	VSS
148	I	VCC4
149	I	VSS
150	I	VSS
151	I	VCC4
152	I	VSS
153	I	VCC4
154	I	VSS
155	I	VSS
156	I	TPH2
157	I	NPTAGSTAT<0>
158	I/O	NPTAG<2>
159	I	NPPOPBUS<7>
160	I	NPTAGSTAT<3>
161	I	NPPOPBUS<12>
162	I	NPPOPBUS<0>
163	I	NPPOPBUS<5>
164	O	NPTERM<2>
165	I	CKMODE
166	I	PHE1
167	I	IREF
168	I/O	NC
169	I	TPH1
170	I	NPTAGSTAT<1>
171	I	NPPOPBUS<8>
172	I	NPPOPBUS<2>
173	I	NPPOPBUS<6>
174	I	NPTAGSTAT<4>
175	I	NPPOPBUS<14>
176	I	NPOUTFTYP<1>
177	O	NPTERM<5>
178	I	XSEL
179	I	CLK
180	O	XPH2
181	O	XPH1
182	O	SERIALOUT
183	O	SERIALIN

Figure 14 Nx587 Pin List, By Pin Number



070

Figure 15 Nx587 Pinout Diagram (Top View)



071

Figure 16 Nx587 Pinout Diagram (Bottom View)

Floating Point Coprocessor Bus Signals (on Nx587)

NPPOPBUS<15:0>	I	**Floating Point Coprocessor Micro-Operations Bus**—Driven by the Nx586 processor to the Nx587 Floating Point Coprocessor to provide a floating-point micro-operation at the peak rate of one per processor clock. The NPPOPBUS<15:0> bus carries both micro-operations and their associated tags, both of which are issued by the Nx586 processor's Decode Unit.
NPNOERR	O	**Floating Point Coprocessor No Error**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor for handshaking to implement the IBM-compatible mode of interrupt handling. This signal is enabled and disabled in software.
NPOUTFTYP<1:0>	I	**Floating Point Coprocessor Output Type**—Asserted by the Nx586 processor to the Nx587 Floating Point Coprocessor for handshaking to implement the IBM-compatible mode of interrupt handling. These signals are enabled and disabled in software.
NPTERM<5:0>	O	**Floating Point Coprocessor Termination**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor to indicate completion of floating-point operations. Only bits 1:0 are connected to the Nx586 processor; the others must be left unconnected.
NPTAGSTAT<5:0>	I	**Floating Point Coprocessor Tag Status**—Driven by the Nx586 processor to the Nx587 Floating Point Coprocessor to synchronize the issuing, retiring, and aborting of instructions.
NPRREQ	I	**Floating Point Coprocessor Read Request**—Asserted by the Nx586 processor to the Nx587 Floating Point Coprocessor, to request use of the NPDATA<63:0> and NPTAG<4:0> buses to transfer data on the next clock. The NPRREQ signal has priority over the NPWREQ signal. When neither is requesting, the processor drives the bus. The processor sometimes makes speculative requests, such as when it concurrently does cache lookups for the data to be transferred. If the processor finds that it cannot use the bus after requesting it, it negates NPRVAL when the bus is granted, otherwise it asserts NPRVAL and transfers the data in the same clock.

NPRVAL	I	**Floating Point Coprocessor Read Valid**—Asserted by the Nx586 processor to the Nx587 Floating Point Coprocessor in the clock following a successful request, to indicate that the data being transferred on the NPDATA<63:0> bus in the current clock is valid.
NPWREQ	O	**Floating Point Coprocessor Write Request**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor, to request control of the NPDATA<63:0> and NPTAG<4:0> buses to transfer data on the next clock. The NPRREQ signal has priority over the NPWREQ signal. The Floating Point Coprocessor makes speculative requests concurrently with its first pass at formatting the output. If it discovers that more formatting is needed, it negates NPWVAL when the NPDATA<63:0> bus is granted, otherwise it asserts NPWVAL and transfers the data in the same clock.
NPWVAL	O	**Floating Point Coprocessor Write Valid**—Asserted by the Nx587 Floating Point Coprocessor to the Nx586 processor in the clock following a successful request, to indicate that the data being transferred on the NPDATA<63:0> bus in the current clock is valid.
NPTAG<4:0>	I/O	**Floating Point Coprocessor Tag Bus**—On each processor clock, this bus carries the five-bit micro-operation tag between the Nx586 processor and the Nx587 Floating Point Coprocessor. The tag identifies the instruction from which the micro-operation was decoded, and it corresponds to the data being transferred on the NPDATA<63:0> bus.
NPDATA<63:0>	I/O	**Floating Point Coprocessor Data**—On each processor clock, this bus carries up to 64 bits of read or write data between the Nx586 processor and the Nx587 Floating Point Coprocessor. The Nx586 processor uses it to provide read data to the Nx587 Floating Point Coprocessor, and the Nx587 Floating Point Coprocessor uses it to write results. The bus's bi-directionality is implemented with arbitration among the NPRREQ and NPWREQ signals. Arbitration priority is given to the processor, hence reads prevail over writes. The winner gets the bus on the next clock. The arbitration and the bus transfer are pipelined one clock apart at the processor-clock frequency. Thus, in every clock, both a request and a transfer are made.

Nx587 System Signals

Nx587 Clock

CLK	I	**NexBus Clock**—A TTL-level clock with a duty cycle between 45% and 55%. NexBus signals transition on the rising edge of CLK. The processor's internal phase-locked loop (PLL) synchronizes internal processor clocks at twice the frequency of CLK.
PHE1	I	**Clock Phase 1**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
PHE2	I	**Clock Phase 2**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
CKMODE	I	**Clock Mode**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
XSEL	I	**Clock Mode Select**—For normal clocking operation, this signal should be pulled low. Refer to Figure 27.
XPH1	O	**Processor Clock Phase 1**—For normal clocking operation, this signal must be left unconnected. Refer to Figure 27.
XPH2	O	**Processor Clock Phase 2**—For normal clocking operation, this signal must be left unconnected. Refer to Figure 27.
IREF	I	**Clock Input Reference**—This signal must be pulled up to V_{DDA} with a 220kΩ resistor.
XREF	O	**Clock Output Reference**—For normal clocking operation, this signal must be left unconnected.
VDDA	I	**PLL Analog Power**—This input provides power for the on chip PLL circuitry and should be isolated from V_{CC} by a ferrite bead and decoupled with a 0.1 µF ceramic capacitor.

Nx587 Interrupts and Reset

NPIRQ*	O	**Floating Point Coprocessor Interrupt Request**—Asserted by the Nx587 Floating Point Coprocessor to the interrupt controller that services the NexBus during floating-point exceptions. The same-named signal from the Nx586 must also be connected to this signal.
RESET*	I	**Global Reset (Power-Up Reset)**—Asserted by system logic. The processor responds by resetting its internal state machines and loading default values in its registers. At power-up it must remain asserted for a minimum of several milliseconds to stabilize the phase-locked loop. See the *Electrical Data* chapter.

Nx587 Test and Reserved Signals

NC	O	**Reserved**—For normal operation, these signals must be left unconnected.
FPTEST	I	**Floating Point TEST**—This pin is to tri-state all outputs except for the following pins: XPH1, XPH2, and XREF. For normal operation, this input must be pulled low.
NPPOPTAG<4:0>	I/O	**Reserved**—These signals must be connected to the same-named signals on the Nx586 processor.
NPSPARE<2:0>	I	**Reserved**—These signals must be connected to the same-named signals on the Nx586 processor and pulled low.
SERIALIN	I	**Serial In**—The input of the scan-test chain. This signal must be left unconnected for normal operation.
SERIALOUT	O	**Serial Out**—The output of the scan-test chain. This signal must be left unconnected for normal operation.
TPH1	I	**Test Phase 1 Clock**—Used for factory scan test support. This signal must be tied low for normal operation.
TPH2	I	**Test Phase 2 Clock**—Used for factory scan test support. This signal must be tied low for normal operation.

Nx587 Alphabetical Signal Summary

CKMODE	I	Clock Mode
CLK	I	NexBus Clock
FPTEST	I	Reserved
IREF	I	Clock Input Reference
NC	O	No Connect
NPDATA<63:0>	I/O	Floating Point Coprocessor Data
NPIRQ*	O	Floating Point Coprocessor Interrupt Request
NPNOERR	O	Floating Point Coprocessor No Error
NPOUTFTYP<1:0>	I	Floating Point Coprocessor Output Type
NPPOPBUS<15:0>	I	Floating Point Coprocessor Micro-Operations Bus
NPPOPTAG<4:0>	I/O	Reserved
NPRREQ	I	Floating Point Coprocessor Read Request
NPRVAL	I	Floating Point Coprocessor Read Valid
NPTAG<4:0>	I/O	Floating Point Coprocessor Tag Bus
NPTAGSTAT<5:0>	I	Floating Point Coprocessor Tag Status
NPTERM<5:0>	I	Floating Point Coprocessor Termination
NPWREQ	O	Floating Point Coprocessor Write Request
NPWVAL	O	Floating Point Coprocessor Write Valid
NPSPARE<2:0>	I	Reserved
PHE1	I	Clock Phase 1
PHE2	I	Clock Phase 2
RESET*	I	Global Reset (Power-Up Reset)
SERIALIN	O	Serial In
SERIALOUT	O	Serial Out
TPH1	I	Test Phase 1 Clock
TPH2	I	Test Phase 2 Clock
VDDA	I	PLL Analog Power
XPH1	O	Processor Clock Phase 1
XPH2	O	Processor Clock Phase 2
XREF	O	Clock Output Reference
XSEL	I	Clock Mode Select

Hardware Architecture

The Nx586 processor and Nx587 floating-point coprocessor are tightly coupled into a parallel architecture with a distributed pipeline, distributed control, and rich hierarchy of storage elements. While the features of the two devices are sometimes listed separately elsewhere in this book, they are treated as an integrated architecture in this chapter. The Nx587 Floating-Point Coprocessor is optional in a system, but if used, each Nx587 requires a companion Nx586 processor. Alternatively, the Nx586 processor can be used by itself, without the Floating-Point Coprocessor.

Bus Structure

The Nx586 processor supports three external 64-bit buses: the NexBus (the processor bus), the L2 cache SRAM bus, and the Floating-Point Coprocessor bus that is shared with the optional Nx587. All buses are synchronous to the NexBus clock, although the Floating-Point Coprocessor bus operates at twice the frequency of the other two buses.

NexBus

The NexBus is a 64-bit synchronous, multiplexed bus that supports all signals and bus protocols needed for cache-coherency. A modified write-once MESI protocol is used for cache coherency. The processor continually monitors the NexBus to guarantee cache coherency.

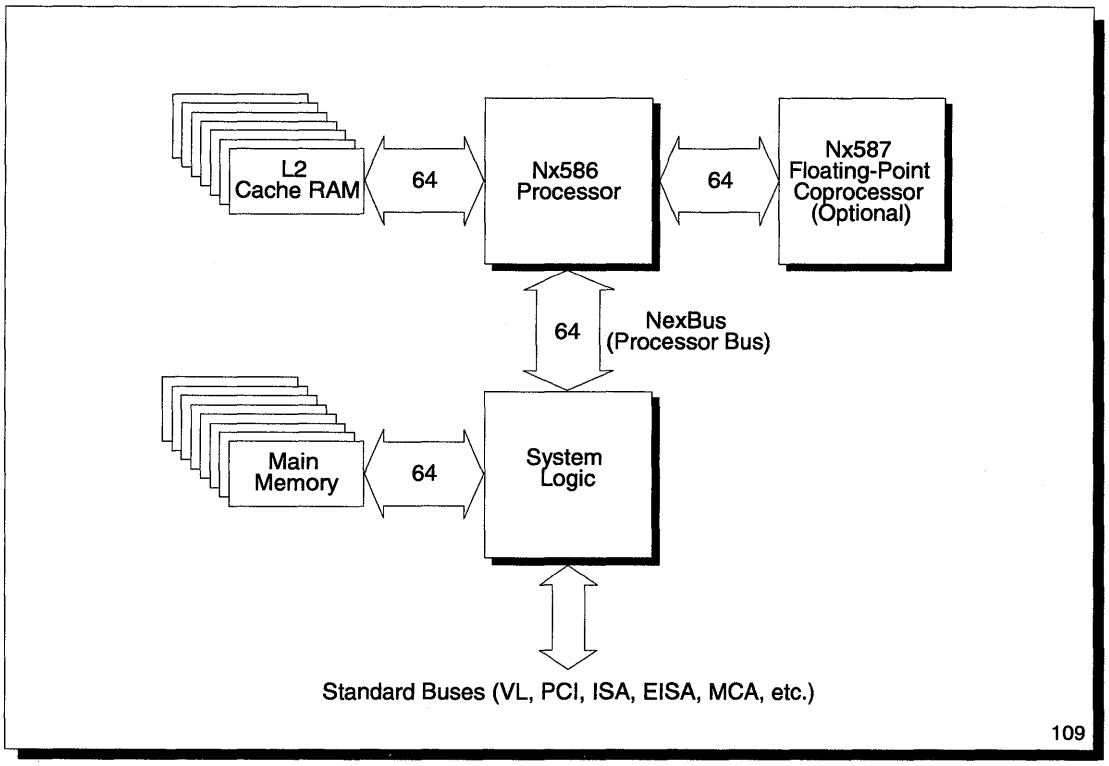


Figure 17 Nx586 based System Diagram

Figure 17 shows the general organization of a Nx586-based system. The systems logic on the NexBus includes the following functions:

- NexBus arbitration
- NexBus interface to standard buses (such as VL, PCI, ISA, EISA, MCA)
- NexBus interface to main memory and peripherals
- Main-memory control and arbitration
- Peripheral control
- System ROM

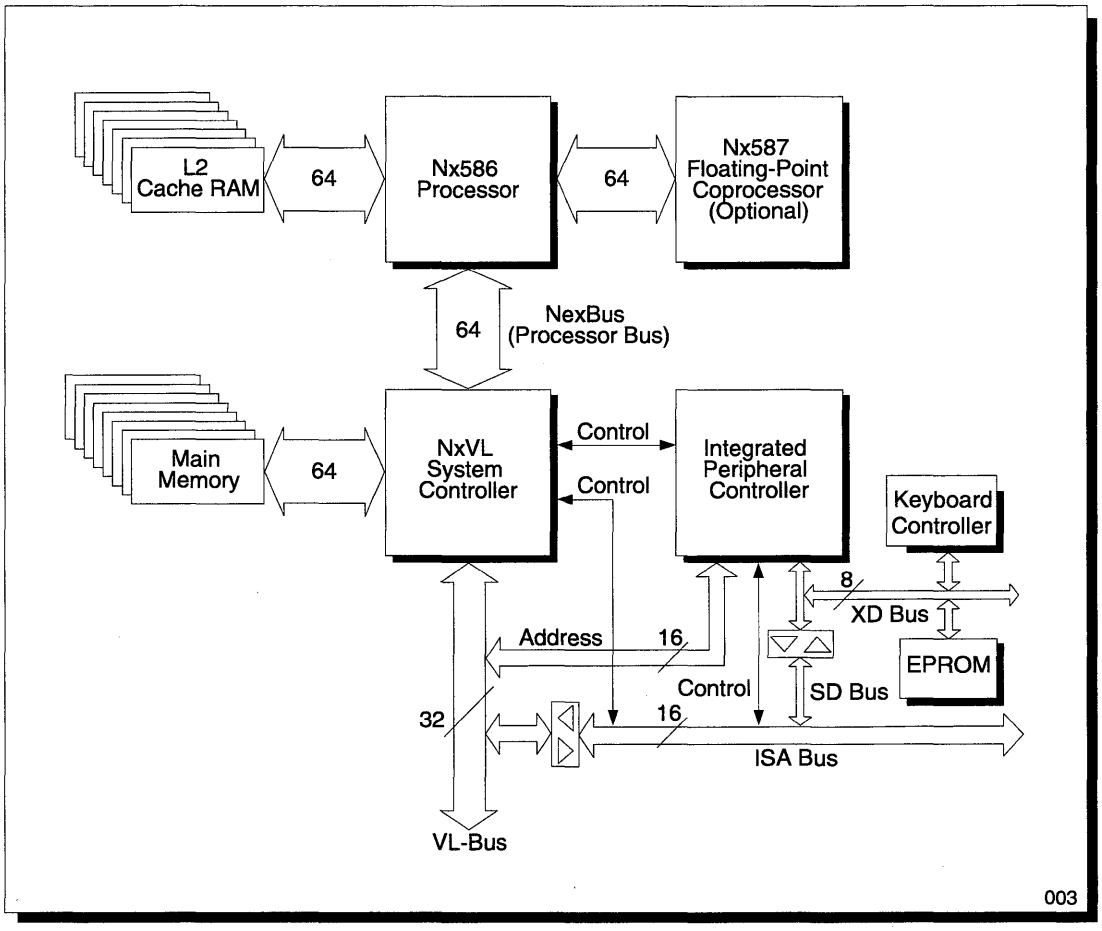


Figure 18 Nx586 based System using the NxVL Diagram

Figure 18 shows a specific implementation of a Nx586 system—one that uses the NxVL system controller.

L2 Cache Bus

The 64-bit L2 cache bus is dedicated to external asynchronous SRAM cache. The bus carries one to eight bytes of cache data, or the tags and state bits for one to four cache banks (sets). The L2 cache is a write-back cache. The processor manages cache-coherency for both L2 and L1 caches.

Floating-Point Coprocessor Bus

The 64-bit Floating-Point Coprocessor bus is dedicated to the optional Nx587 coprocessor. Two arbitration signals implement a simple protocol between the two devices. Arbitration priority is given to the processor, so reads prevail over writes. The winner gets the bus on the next clock. The arbitration and data transfers are pipelined one clock apart at the processor-clock frequency. Thus, in every processor clock, both a bus request and a data transfer can be performed, making the Floating-Point Coprocessor a tightly coupled component of the execution pipeline.

Both the processor and the Floating-Point Coprocessor sometimes make speculative requests for the bus. For example, the processor requests the bus while it concurrently looks in its cache for the data to be transferred. The Floating-Point Coprocessor makes speculative requests concurrently with its first pass at formatting the output, which may in fact need further formatting before transfer. If either device finds that it cannot use the bus after requesting it, it negates its request signal thereby allowing access to the bus by the other device.

Operating Frequencies

There are four operating frequencies associated with the processor, as shown in Figure 19:

- *NexBus*—Operates at the frequency of the system clock (CLK).
- *Processor*—Operates at twice the frequency of the NexBus clock. The Nx586 processor and Nx587 Floating Point Coprocessor both operate at the same frequency.
- *L1 (On-Chip) Cache*—Operates at twice the frequency of the processor clock (four times the frequency of the NexBus clock).
- *L2 (Off-Chip) Cache*—Operates at the same frequency as the NexBus clock. Transfers between L2-cache and the processor occur at the peak rate of one qword every two processor clocks, but the transfers (which can be back-to-back) can begin on any processor clock. Data is returned to the processor on the third clock phase after an access is started.

Unless otherwise specified in this book, a *clock cycle* means the Nx586 processor's clock cycle. However, the timing diagrams in the *Bus Operations* chapter are relative to the NexBus clock, not the processor clock.

Figure 19 shows the relative frequencies for a 66 MHz processor (actually 66.666...MHz). If the NexBus clock runs at 33 MHz (actually 33.333... MHz), the processor and Floating Point Coprocessor run at 66.666...MHz, the on-chip L1 caches run at 66.666... MHz, and the L2-cache bus runs at 33.333... MHz.

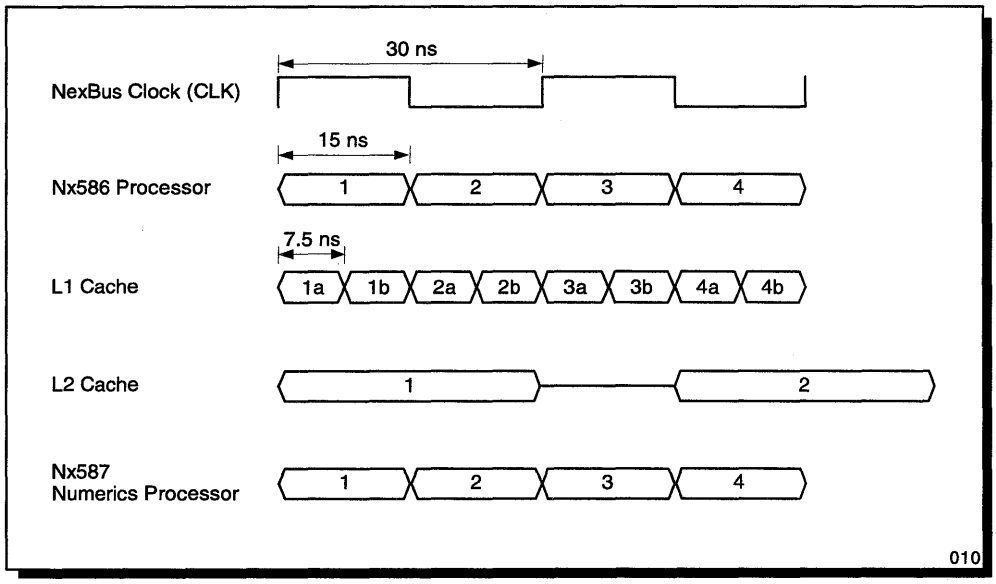


Figure 19 Operating Frequencies (66MHz Processor)

The processor uses an on-chip phase-locked loop and the NexBus clock to internally generate two non-overlapped phases of its own clock, shown in Figure 19 as the 7.5ns phases that drive the L1 cache. Most of the processor's pipeline stages operate on these phases. For example, a register-file access, an adder cycle, a lookup in the translation lookaside buffer (TLB), and an on-chip cache read or write all take a single phase of the processor clock.

The processor supports an average sustainable read and write bandwidth on NexBus of 152 MBytes per second for the 66MHz Nx586 processor, and a peak transfer rate of 267 MBytes per second for the 66MHz Nx586 processor. For additional information, consult the "Bus Operation" chapter.

With a special bus-clock reference scheme that does not use the on-chip phase-locked loop, the chips can operate at any clock frequency between zero and the specified maximum. There are no dynamic circuits that force a minimum frequency, so the chips can be brought to zero frequency without losing data.

Internal Architecture

Figure 20 shows the relationship between functional units in the Nx586 processor and the Nx587 Floating Point Coprocessor. The main processing pipeline is distributed across five units:

- Decode Unit
- Address Unit
- Cache and Memory Unit
- 2 Integer Units
- Floating Point Coprocessor (the optional Nx587)

All functional units work in parallel with a high degree of autonomy, concurrently processing different parts of several instructions. Only the Cache and Memory Unit has an interface that is visible outside the processor.

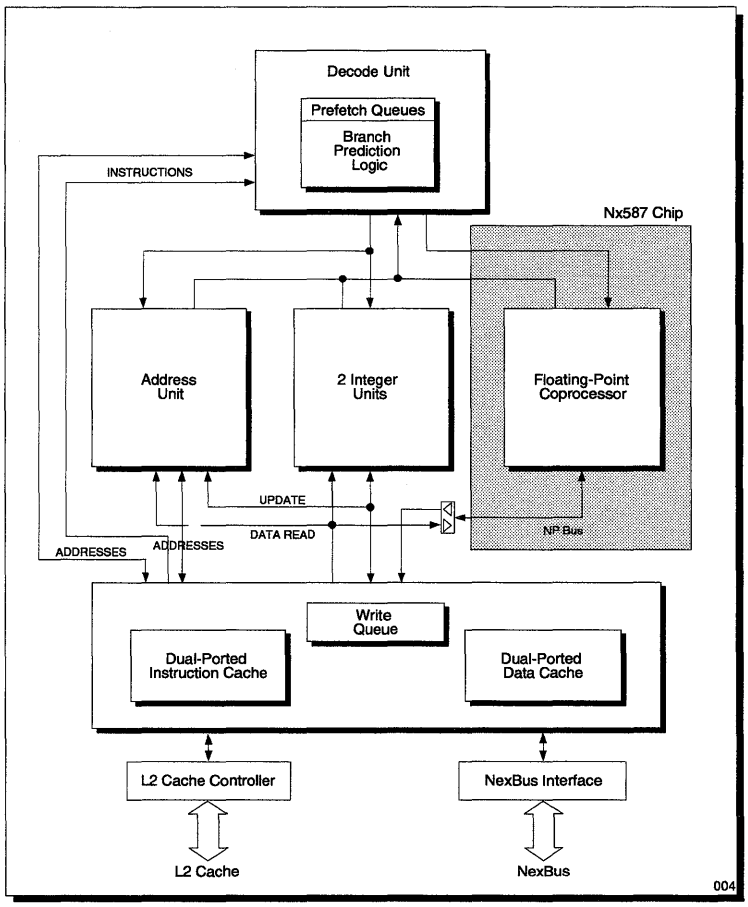


Figure 20 Nx586 Internal Architecture

Storage Hierarchy

The Nx586 architecture provides a rich hierarchy of storage mechanisms designed to maximize the speed at which functional units can access data with minimum bus traffic. Control for a modified write-once cache-coherency protocol (MESI) is built into this hierarchy.

In addition to the L1 and L2 caches, the processor also has three other storage structures that contribute to the speed of accessing information: (1) a prefetch queue in the Decode Unit, (2) a branch prediction in the Decode Unit, and (3) a write queue in the Cache and Memory Unit. The storage hierarchy can continue at the system level with other buffers and caches. For example, systems using the NxVL system controller chip, that chip maintains a prefetch

queue between the L2 cache and main memory that continuously pre-loads cache blocks in anticipation of the processor's next request for a cache fill. Bus masters on buses interfaced to the NexBus can also maintain caches, but those other masters must use write-through caches.

Figure 21 shows this hierarchy during a read cycle in systems supported by the NxVL. Figure 22 shows the analogous organization during a write cycle. All levels of cache and memory are interfaced through 64-bit buses. Physically, transfers between L2 cache and main memory go through the processor via NexBus, and transfers between L1 and L2 cache go through the processor via the dedicated L2-cache bus. While the NexBus is multiplexed between address/status and data, the L2-cache data bus carries only data at 64 bits every NexBus clock cycle. The disk subsystem and software disk cache are included in the figures for completeness of the hierarchy; the software disk cache is maintained in memory by some operating systems.

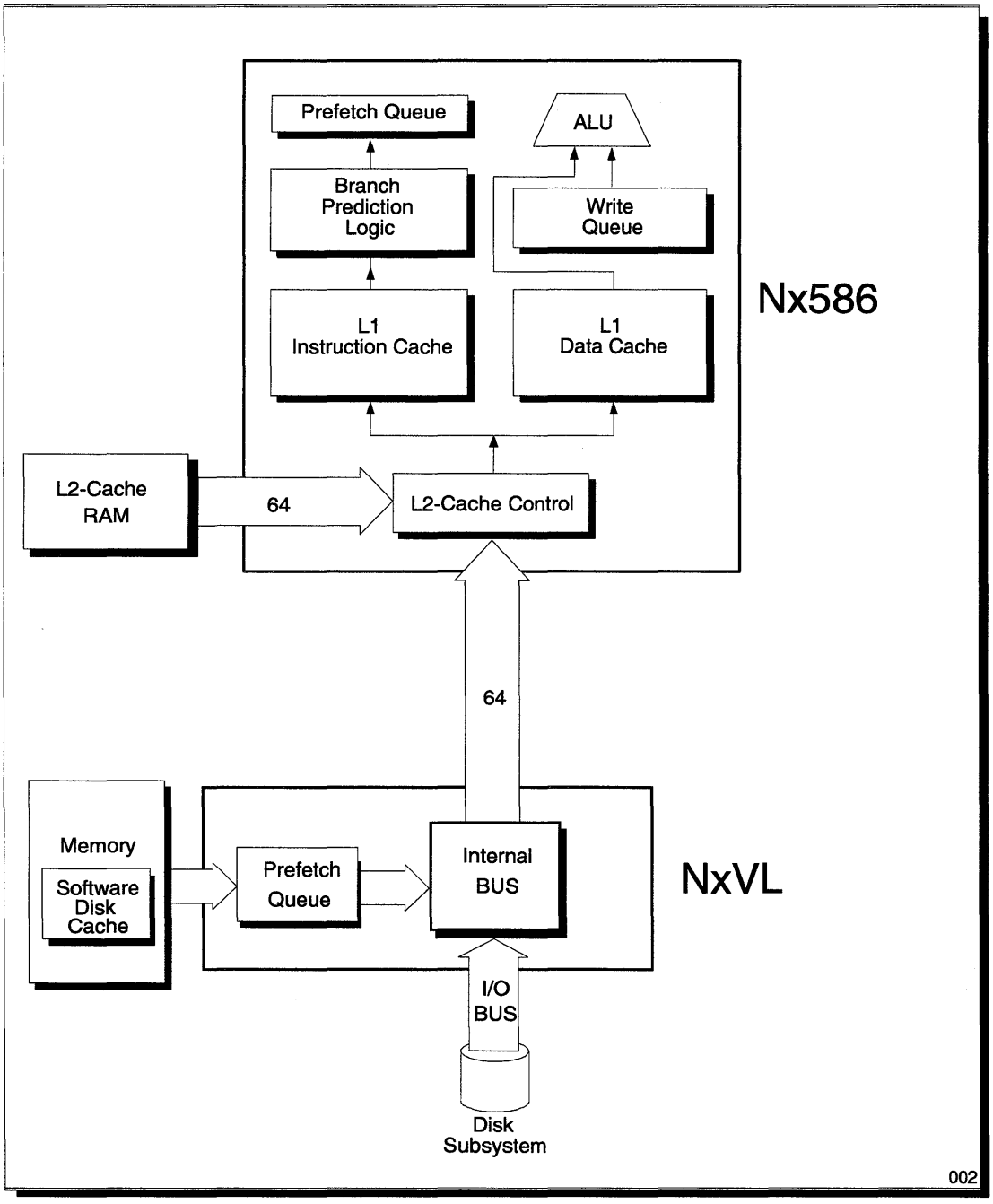


Figure 21 Storage Hierarchy (Reads)

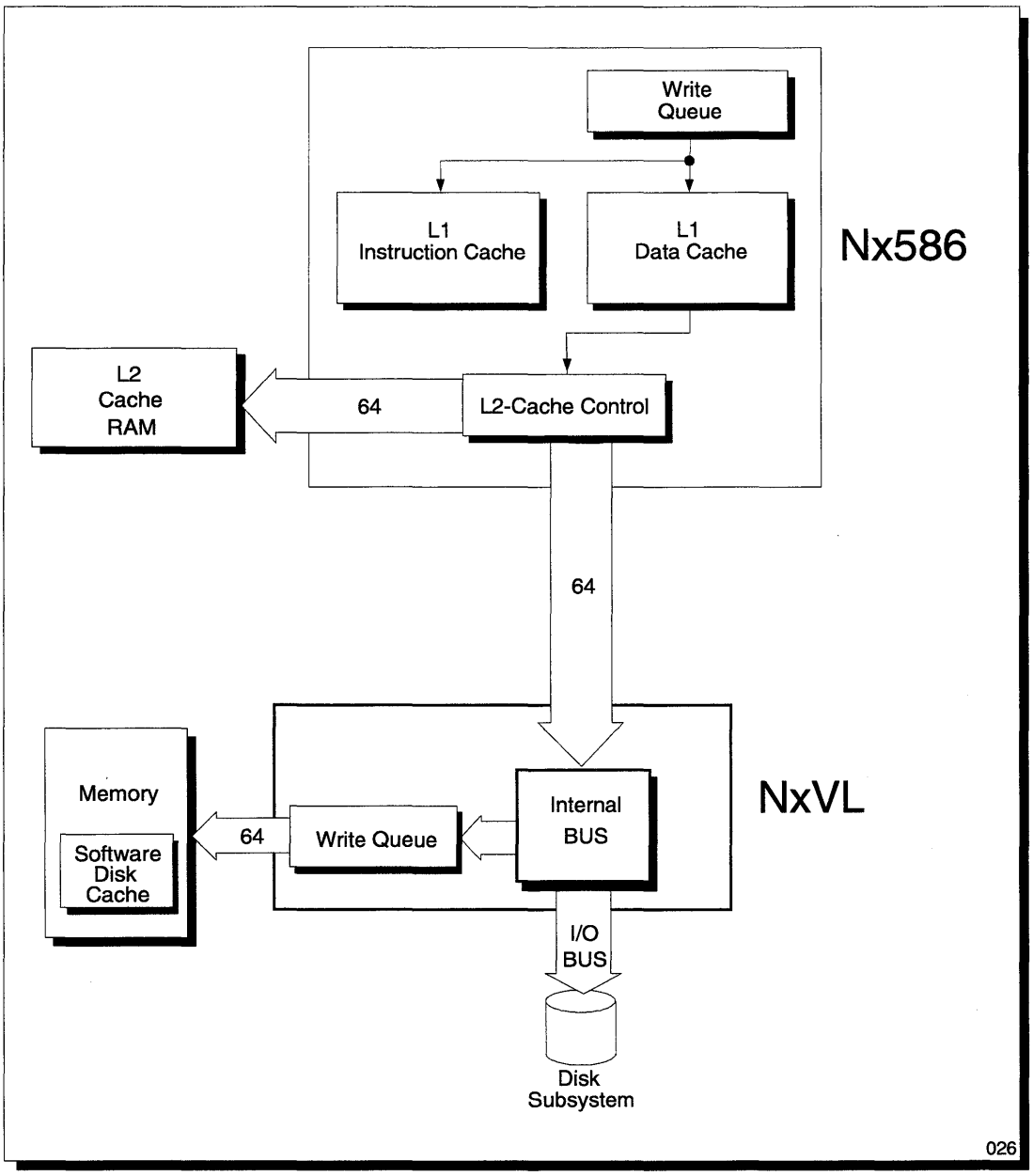


Figure 22 Storage Hierarchy (Writes)

Transaction Ordering

Interlocks enforce transaction ordering in a manner that optimizes read accesses. With the exceptions detailed below, the *general rules* for transaction ordering are:

- *Memory Reads*—Memory reads (whether cache hits or reads on the NexBus) are re-ordered ahead of writes, are performed out of order with respect to other reads, and are done speculatively. With respect to the most recent copy of data, the write queue takes priority over the cache. A hit in the write queue is serviced directly from that queue.
- *I/O and Memory-Mapped I/O Reads*—I/O reads are not done speculatively because they can have side effects in memory that may cause the I/O read to be done improperly. I/O reads have higher priority than memory reads, but all pending writes are completed first.
- *All Writes*—Writes are performed in order with respect to other writes, and they are never performed speculatively. Writes are always held in the write queue until the processor knows the outcome of all older instructions.
- Locked Cycles—Locked read-modify-writes are stalled until the write queue is emptied.
- *Cache-Hit Reads*—The processor holds reads that hit in the cache if any of the following conditions exist:
  - — The cache entry depends upon pending writes that have not yet received their data, are mapped as non-cacheable or are mapped as write-protected.
  - — The read is locked (hence, the rules below for Memory Reads on NexBus are followed).
- *Memory Reads on NexBus*—The processor holds memory reads on the NexBus (cache misses) if any of the following conditions exist:
  - — Reads are I/O or Memory-Mapped I/O.
  - — The write queue has pending writes to I/O or to memory that are mapped as non-cacheable I/O.
  - — The read is locked, and the write portion of a previous locked read-modify-write has not yet been performed.

Cache and Memory Subsystem

Characteristics

The cache and memory subsystem is a key element in the processor's performance. Each of the two on-chip L1 caches (instruction and data) are 16kB in size and dual-ported. The L2 cache is either 256kB or 1MB and single-ported. It is built from an array of eight asynchronous SRAMs. The L2 cache stores instructions and data in 32-byte cache blocks (lines), each of which has an associated tag and cache-coherency state. Separate external tag RAMs are not used. Instead, tag data is stored in a small part of the L2 cache. L2 is a random-access cache, with the L2 cache controller coupled very closely to the processor. Memory references of any kind can be interleaved without compromising performance. It responds to random accesses just as quickly as to block transfers. 32-bytes is the unit of transfer between memory and cache.

	L1 Cache		*L2 Cache*
Contents	Instructions (I Cache)	Data (D Cache)	Instructions and Data (Unified Cache)
Location	processor	processor	controller is on processor; SRAM accessed from 64-bit SRAM bus
Cache Size	16kB	16kB	256kB or 1MB
Ports	2	2	1
Clock Frequency, Relative to Processor Clock	2x	2x	0.5x

Figure 23 Cache Characteristics

If a write needs to go to the NexBus for cache-coherency purposes, it does so before it goes to a cache. Whether the write is needed on the NexBus depends on the caching state of the data: if the data is *shared* (as described later in the *Cache Coherency* section), all other NexBus caching devices need to know about the imminent write so that they can take appropriate action. The processor's caches can be configured so that specified locations in the memory space can be cacheable or non-cacheable and read/write or read only (write-protected).

The Cache and Memory Unit contains a write queue that stores partially and fully assembled writes. The queue serves several functions. First, it buffers writes that are waiting for bus access, and it reorders writes with respect to reads or other more important actions. Second, it assembles the pieces of a write as they become available. (Addresses and data arrive at the queue separately as they come out of the distributed pipelines of other functional units.) Third, the queue is used to back out of instructions when necessary. All writes remain in the queue until signaled by the Decode Unit that the instruction associated with the write is retired—*i.e.,* that there is no possibility of an instruction backout due to a branch not taken or to an exception or interrupt during execution.

Reads are looked up in the write queue simultaneously with the L1 cache lookup. A hit in the write queue is serviced directly from that queue, and write locations pending in the queue take priority over any L1-cache copy of the same location. Reads coming into the unit from NexBus are routed in a pipeline to the processor L2 cache and L1 caches. Reads coming in from the L2 cache are routed first to the processor, then to the L1 caches. Write-backs are going only to the NexBus. Pending writes in the queue go first to the L1 caches (both the instruction and data caches can be written), then to L2 if necessary, then to NexBus if necessary.

The dual ports on the L1 instruction and data caches protect the processor from stalls. In a single clock, the processor can read from port A on each cache while it reads or writes port B on each cache, such as for cache lookups, cache fills, and other cache housekeeping overhead. Both L1 caches may contain identical data, as when a 32-byte cache block contains both instructions and data and is loaded into both L1 caches in different cache-block reads.

Cache Coherency

The processor continually monitors (snoops) NexBus operations by other bus masters to guarantee coherency with data cached in the processor's L2 cache, L1 caches, and branch prediction logic. A type of write-invalidate cache-coherency protocol called modified write-once (MWO) or modified, exclusive, shared, or invalid (MESI) is used. In this protocol, each 32-byte block in the L2 cache is in one of four states:

- *Exclusive*—Data copied into a single bus-master's cache. The master then has the exclusive right (not yet exercised) to modify the cached data. Also called *owned clean* data.
- *Modified*—Data copied into a single bus-master's cache (originally in the exclusive or invalid state) but that has subsequently been written to. Also called *dirty, owned dirty,* or *stale* data.
- *Shared*—Data that may be copied into multiple bus-masters' caches and can therefore only be read, not written.
- *Invalid*—Cache locations in which the data is not correctly associated with the tag for that cache block. Also called *absent* or *not present* data.

The protocol allows any NexBus caching device to gain exclusive ownership of cache blocks, and to modify them, without writing the updated values back to main memory. It also allows caching devices to share read-only versions of data. To implement the protocol, the processor:

- *Requests data* in a specific state by asserting or negating NexBus cache-control bits and signals.
- *Caches data* in a specific state by watching NexBus cache-control input signals from system logic and the slave being accessed.
- *Snoops* the NexBus to detect operations by other masters that hit in the processor's caches.
- *Intervenes* in the operations of other NexBus masters to write back modified data to main memory if a hit occurs during a bus snoop.
- *Updates the state* of cached blocks if a hit occurs during a bus snoop.

The protocol name, *write-once*, reflects the processor's ability to obtain exclusive ownership of certain types of data by writing once to memory. If the processor caches data in the shared state and subsequently writes to that location, a write-through to memory occurs. During the write-through, all other caching devices with shared copies invalidate their copies (hence the name, write-invalidate). After the write, the processor owns the data in the exclusive state, since the processor has the only valid copy and it matches the copy in memory. Any additional writes are local—they change the state of the cached data to modified, although the changes are not written back to memory until a update or cache replacement snoop cycle by another bus master forces the

write-back. Write-once protocols maximize the processor's opportunities to cache data in the exclusive (owned) state even when the processor has not specifically requested exclusive use of data, thereby maximizing the number of transactions that can be performed from the cache.

There are also other means of obtaining ownership of data besides writing to memory, and write operations can be performed in a way that does not modify ownership. The protocol is compatible with caching devices that employ write-through caching policies, if the devices implement bus snooping and support cache-block invalidation. Caching devices that use a cache-block (line) size other than four-qwords must use a write-through policy.

State Transitions

Transitions among the four states are determined by prior states, the type of access, the state of cache-control signals and status bits, and the contents of configuration registers associated with the cache. Figure 24 shows only the basic state transitions for write-back addresses. Transitions occur when the processor reads or writes data (hits and misses), or when it encounters a snoop hit. No transitions are made for snoop misses. In the default processor configuration and depending on the cause of an operation, reads can be either for exclusive ownership or shared use, but *write misses are allocating* (fetch on write)—they initiate a read for exclusive ownership, followed by a write to the cache.

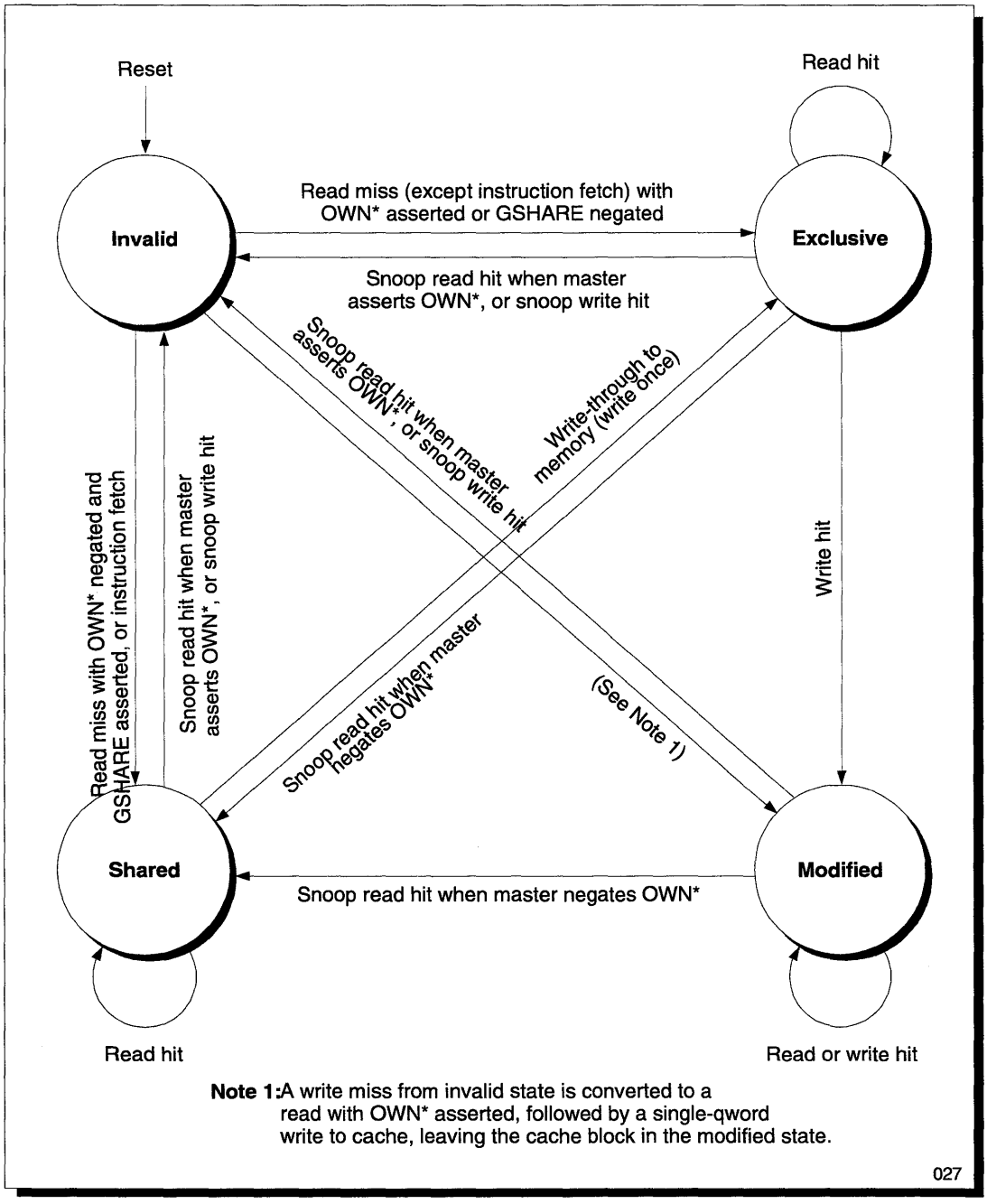


Figure 24 Basic Cache-State Transitions

Figure 25 describes the primary signals and status bits that affect the state transitions shown in Figure 24. The OWN* and SHARE* signals control many transitions. The assertion of OWN* implies that the data is both snoopable (SNPNBL) and cacheable (CACHBL). Figure 26 describes the signals and status bits that affect processor responses during bus snooping. The four sections following these tables describe the characteristics of the states in more detail.

OWN* **NxAD<49>** *address phase*	I/O	**Ownership Request**—Asserted by a master when it intends to cache data in the *exclusive* state. The bit is asserted for write-backs and reads from the stack. If such an operation hits in the cache of another master, that master writes its data back (if copy is modified) and changes the state of its copy to *invalid.* If OWN* is negated during a read or write, another master may not assume that the copy is in *shared* state when not asserting SHARE* signal.
OWNABL	I	**Ownable**—Asserted by the system logic during accesses by the processor to locations that may be cached in the *exclusive* state. Negated during accesses that may only be cached in the *shared* state, such as bus-crossing accesses to an address space that cannot support the MESI cache-coherency protocol. All NexBus addresses are assumed to be cacheable in the *exclusive* state. The OWNABL signal is provided in case system logic needs to restrict caching to certain locations. In systems using the NxVL, the NxVL does not have an OWNABL signal and the processor's OWNABL input is typically tied high for write-back configurations to allow caching in the *exclusive* state on all reads.
SHARE* **GSHARE**	O I	**Shared Data**—SHARE* is asserted by any NexBus master during block reads by another NexBus master to indicate to the other master that its read hit in a block cached by the asserting master, and that the data being read can only be cached in the *shared* state, if OWN* is negated. GSHARE is the backplane NAND of all SHARE* signals. If GSHARE and OWN* are both negated during the read, the data may be promoted to the *exclusive* state because no other NexBus device declared via SHARE* that it has cached a copy. Code fetches will stay in the *shared* state.

Figure 25 Cache State Controls

SNPNBL NxAD<57>	I/O	**Snoop Enable**—Asserted to indicate that the current operation affects memory that may be valid in other caches. When this signal is negated, snooping devices need not look up the addressed data in their cache tags. This signal is negated by the processor on write-backs.
DCL* **GDCL**	O I	**Dirty Cache Line**—Asserted during operations by another master to indicate that the processor has cached the location being accessed in a *modified* (dirty) state. During reads, the requesting master's cycle is aborted so that the processor, as an intervenor, can preemptively gain control of the NexBus and write back its modified data to main memory. While the data is being written to memory, the requesting master reads it off the NexBus. The assertion of DCL* is the only way in which atomic 32-byte cache-block fills by another NexBus master can be preempted by the processor for the purpose of writing back dirty data. During writes, the initiating master is allowed to finish its write. The NexBus Arbiter must then guarantee that the processor asserting DCL* gains access to the bus in the very next arbitration grant, so that the processor can write back all of its modified data *except* the bytes written by the initiating master. (In this case, the initiating master's data is more recent than the data cached by the processor asserting DCL*.)

Figure 26 Bus Snooping Controls

Invalid State

After reset, all cache locations are invalid. This state implies that the block being accessed is not correctly associated with its tag. Such an access produces a *cache miss*. A read-miss causes the processor to fetch the block from memory on the NexBus and place a copy in the cache. If OWN* is negated and GSHARE is asserted, the block changes state from invalid to shared, provided that the memory slave asserts the GBLKNBL signal when each qword is transferred. If the processor asserts OWN* when OWNABL is asserted, or if no other caching device shares the block (GSHARE negated), the processor may change the state of the block from invalid to exclusive. If GBLKNBL is negated, the data may be used by the processor but it will not be cached, and the cache block will remain invalid.

The processor will invalidate a block if another master performs any operation with OWN* asserted that addresses that block, and OWNABL and GXACK are simultaneously asserted. If the block's previous state was modified, the processor will also intervene in the other master's operation to write back the modified data.

Shared State

When the processor performs a read with OWN* negated and GSHARE asserted, and the read misses the cache, the block will be cached in the shared state. The shared state indicates that the cache block may be shared with other caching devices. A block in this state mirrors the contents of main memory. When the processor has cached data in the shared state, it snoops NexBus memory operations by other masters, ignoring only operations for which SNPNBL is negated. When the processor performs block reads that hit in a block shared with another master, that master asserts SHARE*.

When the processor performs a write with OWN* negated—or when it performs a write with OWN* asserted, OWNABL negated, and GXACK asserted—other masters may either invalidate their copy or update it and retain it in the shared state.

When the processor performs a write to a shared block, the processor (1) writes the data through to main memory while asserting OWN* so as to cause other caching masters to invalidate their copies, (2) updates its cache to reflect the write, and (3) if OWNABL and GXACK are both asserted during the write, the processor changes the state of the block to exclusive, otherwise the state remains shared.

If the processor performs a read or write in which OWN*, OWNABL, and GXACK are all asserted, other masters invalidate their copy of such blocks.

Exclusive State

When the processor performs a read with OWN* asserted or GSHARE negated, and the read misses the cache, the block will be cached in the exclusive (owned clean) state. In the exclusive state, as in the shared state, the contents of a cache block mirrors that of main memory. However, the processor is assured that it contains the only copy of the data in the system. Thus, any subsequent write can be performed directly to cache and need not be immediately written back to memory. The cache block so modified will then be in the modified state. Just as with shared cache blocks, the processor snoops NexBus memory operations when it has cached data in the exclusive state, except when SNPNBL is negated.

If another master asserts OWN* while hitting in an exclusive block in the processor, the processor invalidates its copy. A read by another master with

OWN* negated that hits in an exclusive block forces the processor to assert SHARE* and change the block to the shared state, if CACHBL is asserted. If a write by another master hits in an exclusive block, the processor invalidates the block. OWNABL has no effect on snooping the exclusive and modified states, since a cache block could not have been cached in these states if the block were not ownable.

Modified State

The modified (owned stale or dirty) state implies that a cache block previously fetched in the exclusive state has been subsequently written to and no longer matches main memory. As in the exclusive state, the processor is assured that no other master has cached a copy so the processor can perform writes to the cache without writing them to memory.

Reads and single-qword writes by other masters that address a modified block cause the processor to assert DCL* and perform an intervenor operation. The processor writes back its cached data to memory and the other master simultaneously reads it from the NexBus.

During external non-OWN* reads, the processor changes its copy of the block to the shared state. If an external non-OWN* single-qword write with CACHBL asserted hits in a modified block, the processor asserts DCL* and intervenes in the operation. The processor then either asserts SHARE* during the operation. During external block writes (unlike the single-qword writes described above) the processor does not perform an intervenor operation with write-back because the other master overwrites the entire cache block(s). If an external block write hits a modified processor block it invalidates the block.

Internal reads or writes do not change the state of a modified block. However, if another master attempts to write to a block that has been modified by the processor, the modified data (or portions thereof) is written back to memory. During the write-back, the processor negates SNPNBL to relieve other caching devices of the obligation to look the address up in their caches, since a modified block can never be in another cache.

Interrupts

The processor supports maskable interrupts on its INTR* input, non-maskable interrupts on its NMI* input, and software interrupts through the INT instruction. Hardware interrupts (INTR* and NMI*) are asynchronous to the NexBus clock. They are asserted by external interrupt control logic when that logic receives an interrupt request from an I/O device, system timer, or other source. When an active non-maskable interrupt request is sensed by the interrupt controller, the request is passed to the processor which then performs an interrupt acknowledge sequence, as defined in the *Bus Operations* chapter. Maskable interrupt requests must be asserted until cleared by the interrupt service routine.

Systems supported by the NxVL, a 82C206 peripheral controller handles interrupts. The NxVL generates the non-maskable interrupt (NMI*) input to the processor, and it passes along the processor's non-maskable interrupt acknowledge to the 82C206 via the NxVL's INTA* output. For a description of these interrupts, see the *NxVL System Controller Databook*.

Clock Generation

Five signals determine the manner in which the processor's internal clock phases (PH1 and PH2) are derived or provided. These signals include CKMODE, XSEL, CLK, PHE1, and PHE2. These signals determine one of foures: Phase-Locked Loop (the normal operating mode), External Phase Inputs, or External Processor Clock, as shown in Figure 27 and described in the sections below.

Mode Type	*Mode #*	*CKMODE*	*XSEL*	*PHE1*	*PHE2*
Phase-Locked Loop (normal operating mode)	0	0	0	0	0
External Processor Clock	1	0	1	Input at 2x the CLK frequency	1
Test Mode	2	1	0		
External Phase Inputs	3	1	1	Externally supplied at 2x the CLK frequency	Externally supplied at 2x the CLK frequency

Figure 27 Clocking Modes

Mode #0:

In the *phase-locked loop* mode, the internal clock phases are derived from the external NexBus clock (CLK) via a phase-locked loop (PLL). In all modes, the CLK input must be driven at one-half the processor's internal operating frequency so as to provide the bus-interface logic with a signal that defines the external clock cycle. For TTL compatibility, the rising edge of CLK is its significant edge. The Phase-Locked Loop mode is recommended for most system designs.

Mode #1:

In the *external processor clock* mode, the internal clock phases are derived from PHE1 input signal while PHE2 is pulled high. The PHE1 input signal operates at twice the frequency of CLK. The falling edge of the internal phase2 will occur before the rising edge of XREF, which is a buffered CLK output, and can be observed on the XPH2 output. This mode allows bypassing the PLL for test purposes or to change the clock frequency, as when entering or leaving a low-power mode.

Unlike the Phase-Locked Loop mode, the other two modes operate the internal phases at the externally supplied frequency that has to be twice the CLK frequency. In order to allow the External Phase Input modes to generate and control an external phase-locked loop, both internal clock phases are output via buffers on the XPH1 and XPH2 signals and an additional signal XREF is provided for CLK.

Mode #2:

In the Test mode, both phases are stopped in an off (low) state, which is necessary to employ scan logic.

Mode #3:

In the *external phase inputs* mode, the internal clock phases are controlled by the two external phase inputs, PHE1 and PHE2. These inputs are buffered to drive the internal clock distribution system.

Bus Operations

This chapter covers bus cycles and cache-coherency operations. The bus cycles are conducted primarily on NexBus although their effects can also be seen on the L2 SRAM bus. The NexBus clock, shown in the timing diagrams accompanying this text, runs at half the frequency of the processor's internal clock.

Operations between the processor and the L2-cache SRAM, as well as operations between the processor and the Nx587 Floating Point Coprocessor on the NP bus are not described here, since these operations are not intended for system logic interfacing. Instead, a typical design example is provided in the *Hardware Architecture* chapter in which the processor-to-SRAM and processor-to-587 connections are illustrated.

☞ In this chapter, the term "clock" refers to the *NexBus clock* not to the processor clock, as is meant elsewhere throughout this book.

Accesses on the Level-2 Cache Bus

Figure 19 in the *Nx586 Hardware Architecture* chapter compares the basic clock timing for the processor, its L1 caches, and the L2 cache. An L1 cache miss may cause an access to the L2 cache, which resides off-chip on a dedicated 64-bit bus. Figure 28 shows a read, write, and read to the L2 cache. Transfers can begin on any processor clock and occur at the peak rate of eight bytes every two processor clocks.

The notation regarding *Source* in the left-hand column of Figure 28 indicates the chip or logic that generates the signal. When signals are driven by multiple sources, all sources are shown, in the order in which they drive the signal. In some cases, signals take on different names as outputs are ORed in group-signal logic. In these cases, the signal source is shown with a subscript, where the subscript indicates the device or logic that originally caused the change in the signal.

In addition, Figure 28 shows a read followed by a write followed by a read cycle. Reads (or writes) can be back-to-back without dead cycles. A dead cycle is shown after the last read. The processor clock, which runs at twice the rate of the NexBus clock (CLK), is represented here by its two phases, PH1 and PH2. These phases are not visible at the pins except through the delayed outputs, XPH1 and XPH2. The data-sampling point is shown as the falling edge of PH2, which is relative to the rising edge of CLK. Two pins for COE* are shown, A and B. Both pins are indentical in function and transition on the rising edge of PH1. The two pins are made available for loading considerations

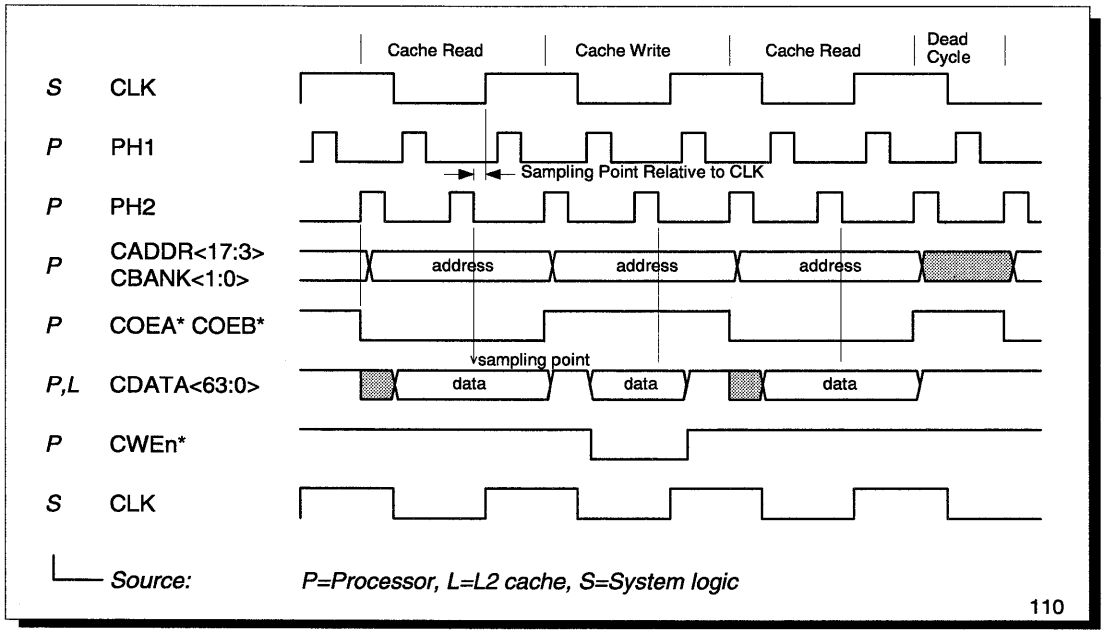


Figure 28 Level-2 Cache Read and Write

NexBus Arbitration and Address Phase

Processor operations on the NexBus may or may not begin with arbitration for the bus. To obtain the bus, the processor asserts NREQ*, LOCK*, and/or AREQ* to the NexBus Arbiter, which responds to the arbitration winner with GNT*. Automatic re-grant occurs when the NexBus Arbiter holds GNT* asserted at the time the processor samples it, in which case the processor need not assert NREQ*, LOCK*, or AREQ* and can immediately begin its operation.

NREQ*, when asserted, remains active until GNT* is received from the NexBus Arbiter. In systems using the NxVL as the NexBus Arbiter, NREQ* is treated the same as AREQ*; when NexBus control is granted, control of all other buses is also granted at the same time.

LOCK* is asserted during sequences in which multiple bus operations should be performed sequentially and uninterrupted. This signal is used by the NexBus Arbiter to determine the end of such a sequence. Cache-block fills are not locked; they are implicitly treated as atomic reads. A NexBus Arbiters may allow a master on another system bus to intervene in a locked NexBus transaction. To avoid this, the processor asserts AREQ*. LOCK* is typically software-configured to be asserted for read-modify-writes and explicitly locked instructions.

AREQ* is asserted to gain control of the NexBus or any other buses supported by the system. This signal always remains active until GNT* is received.

When GNT* is received, the processor places the address of a qword (for memory operations) on NxAD<31:3> or the address of a dword (for I/O operations) on NxAD<15:2>. It drives status bits on NxAD<63:32> and asserts its ALE* signal to assume bus mastership and to indicate that there is valid address on the bus. The processor asserts ALE* for only one bus clock. The slave uses the GALE signal generated by system logic to enable the latching of address and status from the NexBus.

Single-Qword Memory Operations

Figure 29 shows the fastest possible single-qword read. The notation regarding *Source* indicates the logic that originated the signal as an output. In this figure and others to follow, the source of group-ORed signals (such as GXACK) is shown subscript with a symbol indicating the device or logic that output the originally activating signal. For example, the source of the GXACK signal is shown as "Sp", which means that system logic (S) generated GXACK but that the processor (P) caused this by generating XACK*. In some timing diagrams later in this section, bus signals take on different names as outputs cross buses through transceivers or are ORed in group-signal logic; in these cases, the source of the signals is shown subscript with a symbol indicating the logic that originally output the activating signals.

The data phase of a fast single-qword read starts when the slave responds to the processor's request by asserting its XACK* signal. The processor samples the GXACK and GXHLD signals from system logic to determine when data is placed on the bus. The processor then samples the data at the end of the bus clock after GXACK is asserted and GXHLD is negated. The operation finishes with an idle phase of at least one clock.

This protocol guarantees the processor and other caching devices enough time to recognize a modified cache block and to assert GDCL in time to cancel a data transfer. A slave may not assert XACK* until the second clock following GALE. However, the slave must always assert XACK* during or before the third clock following GALE, since otherwise the absence of an active GXACK indicates to the system-logic interface between the NexBus and other system buses (called the *alternate-bus interface)* that the address must reside on the other system bus. In that case, the system-logic interface to that other bus assumes the role of slave and asserts GXACK.

Figure 29 shows when GBLKNBL may be asserted. If appropriate, the slave must assert GBLKNBL no later than it asserts XACK*, and it must keep GBLKNBL asserted until it negates XACK*. It must negate GBLKNBL at or before it stops placing data on the bus. Although not shown, OWNABL must also be valid (either asserted or negated) whenever GXACK is asserted.

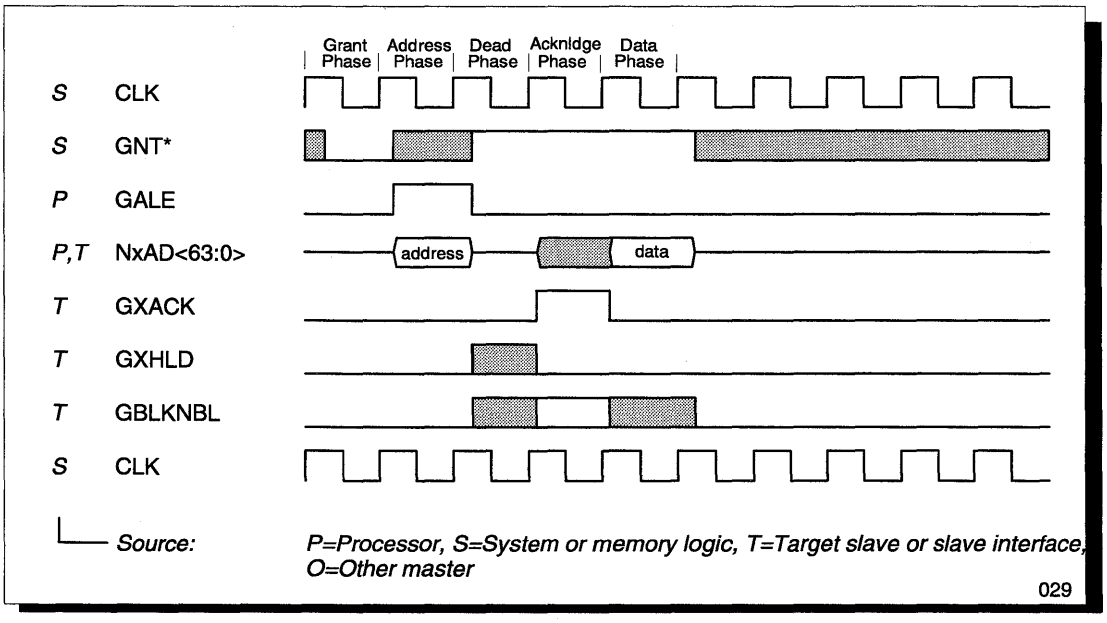


Figure 29 Fastest Single-Qword Read

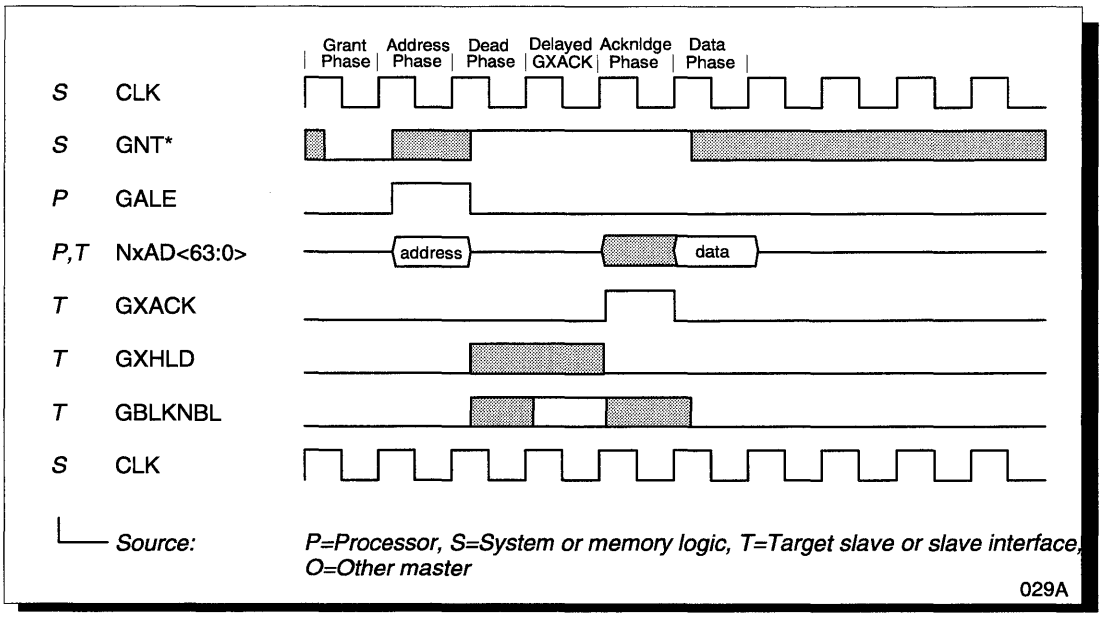


Figure 30 Fast Single-Qword Read with a delayed GXACK

If the slave is unable to supply data during the next clock after asserting XACK*, the slave must assert its XHLD* signal at the same time. Similarly, if the processor is not ready to accept data in the next clock it asserts its XHLD* signal. The slave supplies data in the clock following the first clock during which GXACK is asserted and GXHLD is negated. The processor strobes the data at the end of that clock. A single-qword read with wait states is shown in Figure 31 and 32. For such an operation, the slave must negate XACK* after a single clock during which GXACK is asserted and GXHLD is negated, and it must stop driving data onto the bus one clock thereafter. The processor does not assert XHLD* while GALE is asserted, nor may either party to the transaction assert XHLD* after the slave negates GXACK. In the example shown in Figure 31, the slave asserts GXACK at the latest allowable time, thereby inserting one wait state, and GXHLD is asserted for one clock to insert an additional wait state. The slave may or may not drive the NxAD<63:0> signals during the wait states. The processor will not drive them during the data phase of a read operation.

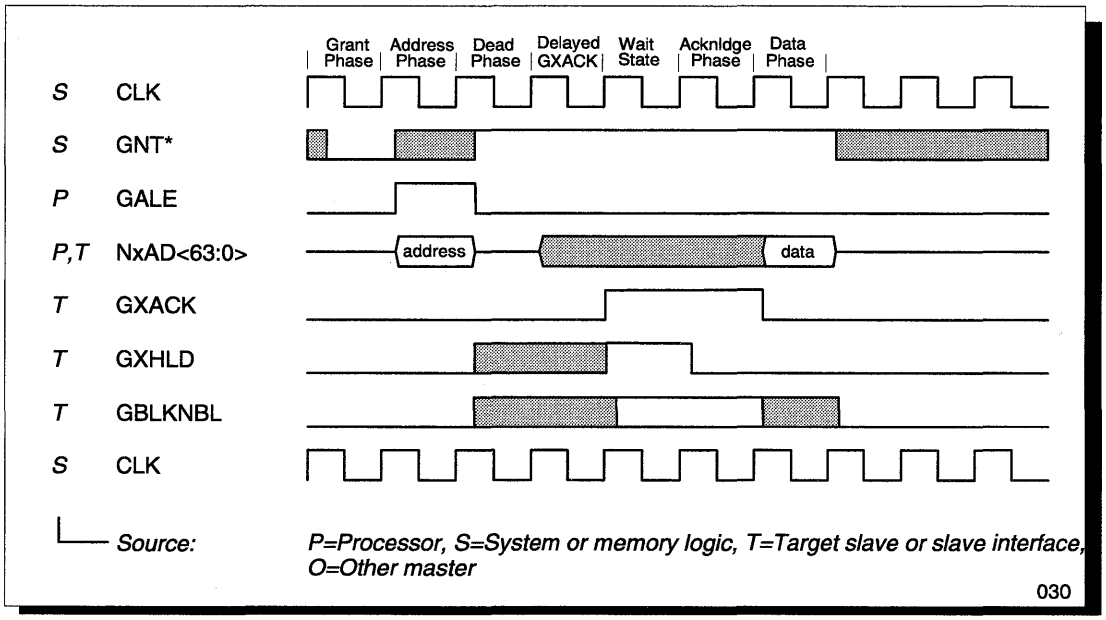


Figure 31 Single-Qword Read with Wait States using a delayed GXACK

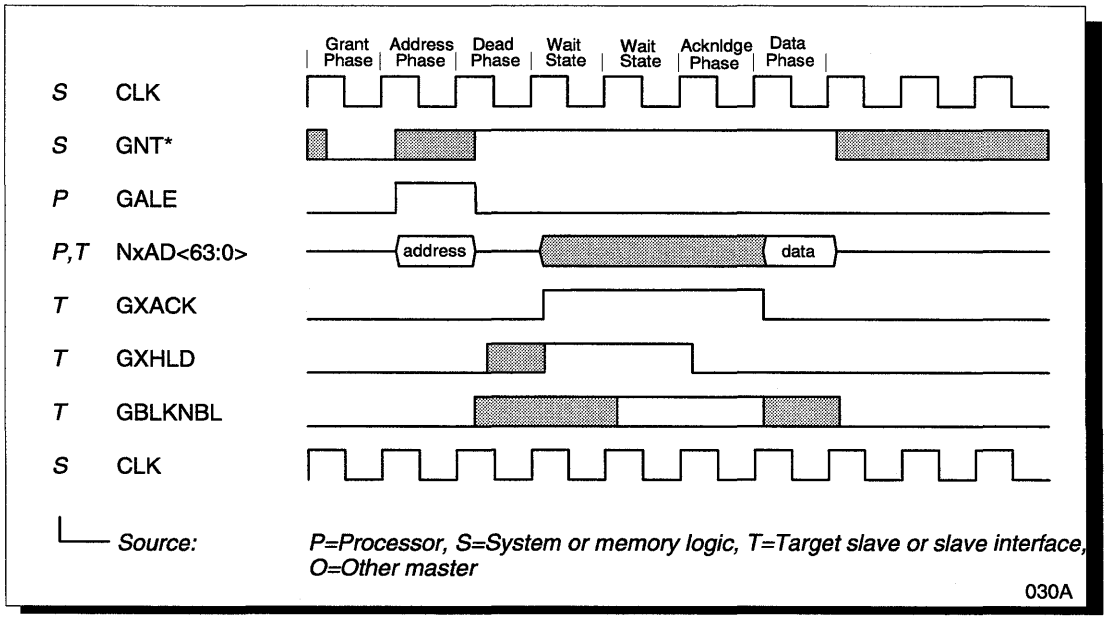


Figure 32 Single-Qword Read with Wait States using GXHLD only

A single-qword write operation is handled similarly. Figure 33 illustrates the fastest write operation possible. Figure 34 shows a single-qword write with wait states. After the bus is granted, the processor puts the address and status on the bus and asserts ALE*. As in the read operation, the slave must assert its XACK* signal during either the second or third clock following the assertion of GALE. If the slave is not ready to strobe the data at the end of the clock following the assertion of GXACK, it must assert its XHLD* signal. The processor places the data on the bus in the clock after the assertion of GXACK, which may be as soon as the third clock following the assertion of GALE. The slave samples GXHLD to determine when the data is valid. The processor will drive data as soon as it is able, and it continues to drive the data for one (and only one) clock after the simultaneous assertion of GXACK and negation of GXHLD. As in the read operation, the slave's XACK* is asserted until the clock following the trailing edge of GXHLD.

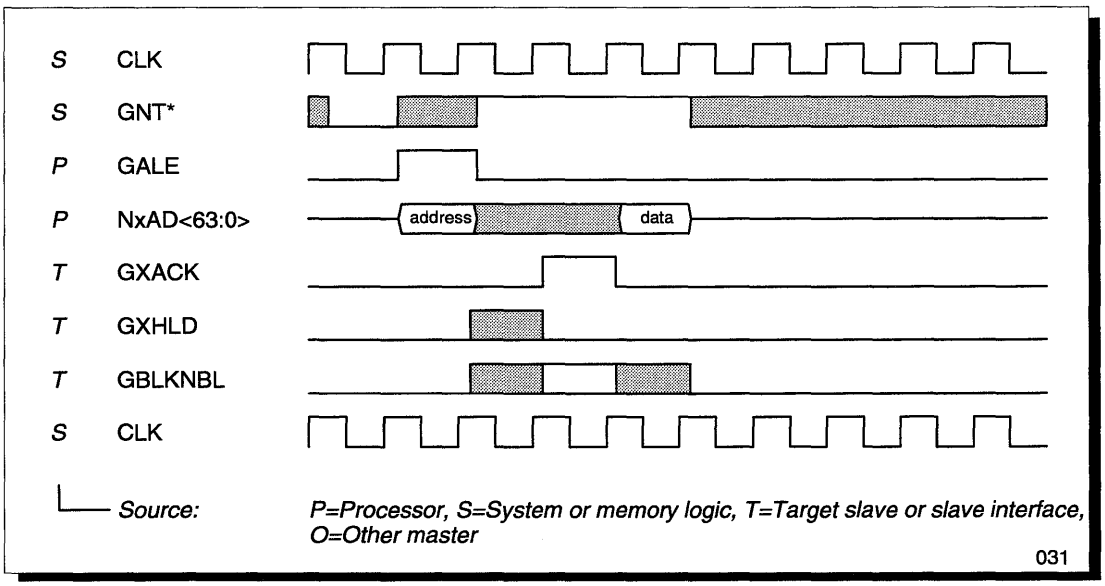


Figure 33 Fastest Single-Qword Write

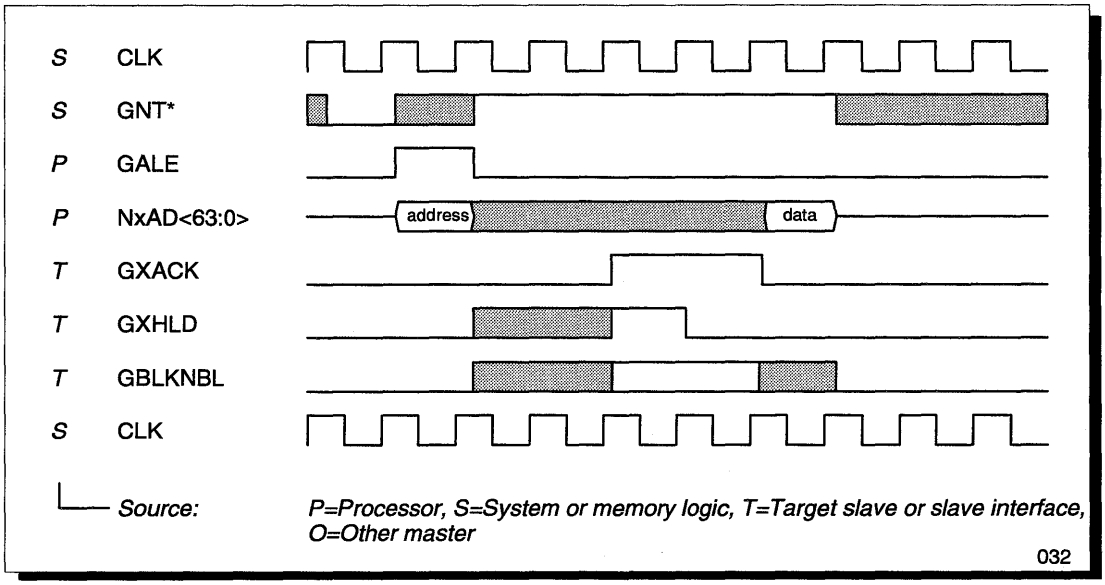


Figure 34 Single-Qword Write With Wait States

Cache Line Memory Operations

The processor performs cache line fill operations with memory at a much higher bandwidth than the single-qword operations described in the previous section. Bursts, both reads and writes, are done only in four-qword increments (32-bytes). All cache line reads are cache fills.

Cache line reads and writes are indicated by the assertion of BLKSIZ* during the address/status phase of the bus operations, as previously defined for single-qword operations.

A cache line operation consists of a single address phase followed by a multi-transfer data phase. The data transfer may begin with *any* qword in the block, as indicated by the address bits, but it then proceeds through additional qwords of the specified contiguous data in an order.

I/O Operations

I/O operations on the NexBus are performed exactly like single-qword reads and writes, with three exceptions. First, the I/O address space is limited to 64K bytes. Second, the 16-bit I/O address is broken into two fields: fourteen address bits and two byte-enable bits. I/O addresses do not use BE<7:2>* (which must be set to all 1's) but instead specify a quad address on NxAD<2>. Third, data is

always transferred on NxAD<15:0>, and NxAD<63:16> is undefined during the data transfer phase of an I/O operation.

I/O operations are indicated by driving 010 (data read) and 011 (data write) on NxAD<48:46> and all zeros on NxAD<31:16> when GALE is asserted. I/O space is always non-cacheable, so a slave should never assert GBLKNBL when responding to an I/O operation.

Interrupt-Acknowledge Sequence

When an interrupt request is sensed by external interrupt-control logic, the request is signaled to the processor by the control logic, the processor acknowledges the interrupt request (during which sequence the controller passes the interrupt vector), and the processor services the interrupt as specified by the vector. The hardware mechanism is described above in the *Hardware Architecture* chapter.

An interrupt-acknowledge sequence, shown in Figure 35, consists of two back-to-back locked reads on NexBus, where the operation type (NxAD<48:46>) is 000 and the byte enable bits BE<7:0>* = 11111110. The first (synchronizing) read is used latch the state of the interrupt controller. It is indicated by NxAD<2> = 1 (I/O-byte address 4). The second read is used to transfer the 8-bit interrupt vector on NxAD<7:0> to the processor, which uses it as an index to the interrupt service routine. This read is indicated by NxAD<2> = 0 (I/O-byte address 0). During these two reads only the least significant bit of the address field is driven to a valid state. The most significant bits are undefined. After the interrupt is serviced, the request is cleared and normal processing resumes.

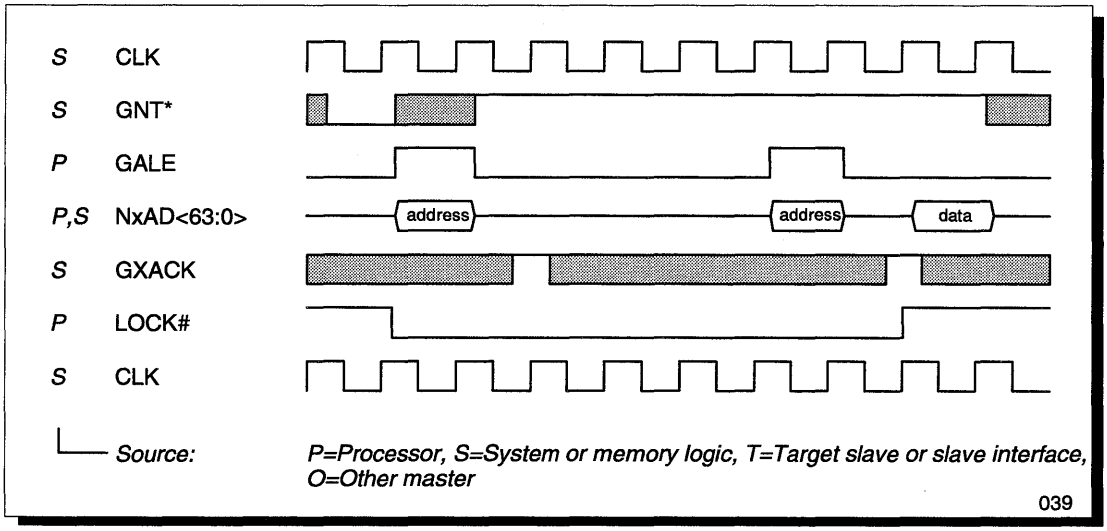


Figure 35 Interrupt-Acknowledge Cycle

Halt and Shutdown Operations

Halt and shutdown operations are signaled on the NexBus by driving 001 on NxAD<48:46> during the address/status phase, as shown in Figure 36. The halt and shutdown conditions are distinguished from one another by the address that is simultaneously signaled on the byte-enable bits, BE<7:0>* on NxAD<39:32>. The processor does not generate a data phase for these operations.

Type of Bus Cycle	*NxAD<48> M/IO**	*NxAD<47> D/C**	*NxAD<46> W/R**	*NxAD<39:32> BE<7:0>**	*NxAD<31:3>*	*NxAD<2>*
Halt	0	0	1	11111011	all zeros	0
Shutdown	0	0	1	11111110	all zeros	0

Figure 36 Halt and Shutdown Encoding

For the halt operation, the processor places an address of 2 on the bus, signified by BE<7:0>* bits (NxAD<39:32>) = 11111011. NxAD<2> = 0 and NxAD<31:3> are undefined. After this, the processor remains in the halted state until NMI*, RESETCPU*, or RESET* becomes active.

For the shutdown operation, the processor places an address of 0 on the bus, signified by BE<7:0>* bits (NxAD<39:32>) = 11111110. NxAD<2> = 0 and NxAD<31:3> are undefined. An external system controller such as the NxVL

will decode the shutdown cycle and assert RESETCPU*. After this, the processor performs a soft reset, RESETCPU*; that is, the processor is reset, but the memory contents, including modified cache blocks, are retained.

Because the Nx586 processor has a 64-bit data bus rather than a 32-bit data bus, eight total byte-enable bits (BE<7:0>*) are specified for double dword bus.

Obtaining Exclusive Use Of Cache Blocks

The processor can obtain ownership of a cache block either *preemptively* or *passively*. Preemptive ownership is gained by asserting OWN* during the address/status phase of a read or write operation. Whenever the processor needs to write a cache block that is either cached in the shared or invalid state, it performs a preemptive read-to-own operation by asserting OWN* during a single-qword write or four-qword block read.

Passive ownership is normally gained when the processor performs a block read, because other NexBus caching devices must snoop block reads. If any part of a block addressed by the processor's read operation resides in another NexBus device's cache, regardless of state, that device asserts SHARE* after the assertion of GALE but not later than the clock during which the first qword of the block is transferred. SHARE* remains asserted through the entire data transfer. If the processor sees GSHARE negated during a block read when it samples the first qword of the block, it knows that it has the only copy. It can therefore cache the block in the exclusive state rather than the shared state, if and only if OWNABL is asserted by system logic.

If another NexBus caching device is unable to meet this timing in the fastest possible case, it must assert XHLD* to delay the operation until it is able to perform the cache check. While it is possible to put a caching device on NexBus that is unable to check its cache and report SHARE* correctly, but instead always asserts SHARE*, this has a very negative effect on system efficiency. It is also possible to design a device that invalidates its cache block during any block read hit, in which case only the efficiency of that one device is impaired.

If the processor addresses a non-cacheable block on a system bus other than NexBus, the system-logic interface between the NexBus and the other system bus (called the *alternate-bus interface)* must indicate this by negating GBLKNBL, and it may not perform block reads or writes to such a block. If the block on the other bus is cacheable, it can only be cached in the shared state, since standard system buses (such as VL bus and ISA bus) do not support the MESI caching protocol, and it is not possible to cache their memory addresses in the exclusive state.

The OWNABL signal from system logic is used to indicate cacheability of locations on other system buses. Whenever OWNABL is negated during a bus operation, the processor will not cache the block in the exclusive state even if the processor asserted OWN*; instead, it may cache the block in the shared state if other conditions permit it.

GBLKNBL and GSHARE must be asserted by system logic at the same time that OWNABL is negated. The timing of these three signals is identical: they should be valid whenever GXACK is asserted. They may be (but need not be) asserted ahead of XACK*, and may (but, except for GSHARE, need not) be held one clock after the negation of XACK*. This timing differs from that of GSHARE, since when OWNABL is asserted GSHARE is not required to be valid until the clock following the negation of GXHLD—i.e., coincident with the data transfer.

Intervenor Operations

The examples given above assume that the addressed data does not reside in a modified cache block. When an operation by another NexBus master results in a cache hit to a modified block in the processor, the processor intervenes in the operation by asserting DCL*. The timing for DCL* is the same as that for SHARE*: the NexBus master samples GDCL on the same clock in which it samples NexBus data. An asserted GDCL indicates to the master that data cached by the processor is modified. To meet the fastest timing requirements, the processor asserts DCL* no later than the third clock following the assertion of GALE. If a MESI write-back caching device is unable to determine in a timely manner whether a transaction hits in its cache, it must assert XHLD* to delay the transfer.

If a block write operation by another master hits a modified cache block in the processor, the processor does not assert DCL*, since such a block write replaces all of a cache block. Instead, the processor invalidates the block.

An addressed slave that sees GDCL asserted during the first qword transfer of an operation must abort the operation by negating GXACK. It may then perform a block write-back starting with the first qword. Immediately after the operation is completed, as determined by the negation of GXACK, the NexBus Arbiter must grant the bus to the intervenor by asserting GNT*. The Arbiter must not grant the bus to any other requester, even if the previous master has asserted AREQ* and/or LOCK*, because DCL* has absolutely the highest priority. Upon seeing GNT* asserted, the intervenor (whether the processor or another master) immediately updates the memory by performing a block write, beginning at the qword address specified in the original operation. The intervenor negates DCL* before performing the first data transfer, but not before it asserts ALE*. During this memory update, the master must sample the data it requested (if the operation was a read) as it is sent to memory on

NexBus by the intervenor. If the master is not ready to sample the data, it can assert XHLD*, as can both the intervenor and the slave; all three parties to the operation examine GXHLD to synchronize the data transfer.

Modified Cache-Block Hit During Single-Qword Operations

During single-qword reads that hit in a modified cache block, the NexBus sequence looks like a normal single-qword read from the memory followed by a block write by the intervenor. Figure 37 illustrates the timing. The fastest time is shown for the operation, while both the fastest and slowest possible times are shown for the leading edge of GDCL. For a slow device intervening in a fast operation, GDCL is available to be sampled on the same clock as the first qword of data is available.

In Figure 37, two sources are shown for GALE and NxAD<63:0>, and one source (S_P) has a subscript. The source is the chip or logic that outputs the signal. The subscript for the source indicates the chip or logic that originally caused the change in the signal. In systems that use the NxVL for system and memory control, the source labeled "S" is the NxVL or other system logic.

During single-qword writes, the master with the modified cache block asserts DCL* to indicate that the single write will be followed by a block write. If the single write included only some of the bytes of the qword, the intervenor records this fact, and during the subsequent block write it outputs byte-enable bits indicating the other bytes of the qword. For example, if the byte-enable bits of the single write were 00000111, the intervenor outputs 11111000. In other words, the intervenor updates only those bytes that were not written by the master. Except for such intervening write-back operations, block writes must have all byte-enable bits asserted (00000000). During block write-backs, byte-enable bits apply only to the first qword, so all bytes of the final three qwords are written.

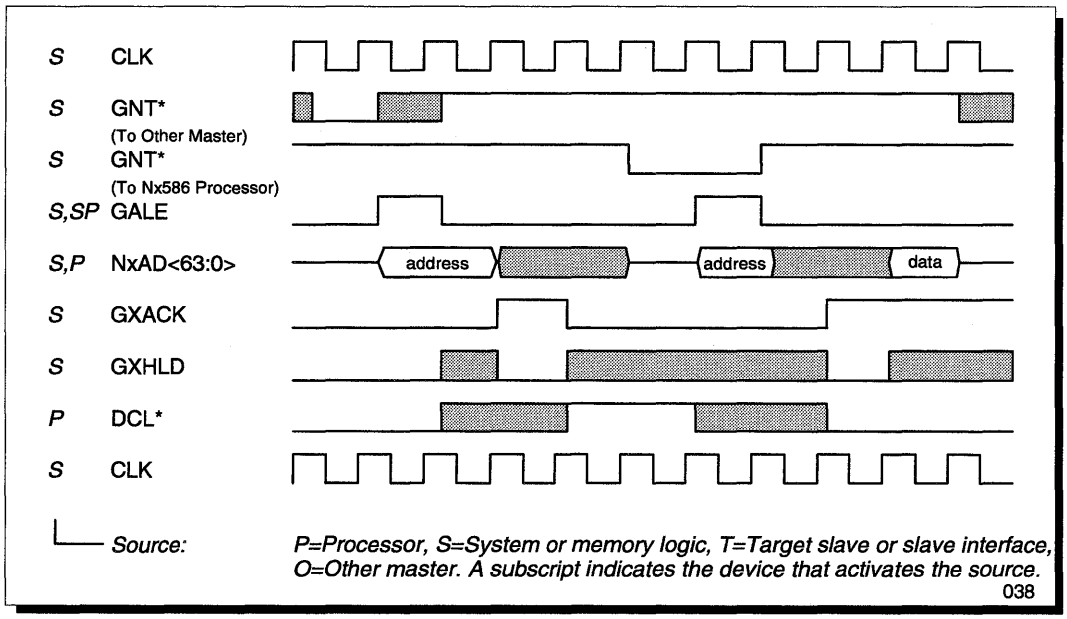


Figure 37 Single-Qword Read Hits Modified Cache Block

Modified Cache-Block Hit During Four-Qword (Block) Operations

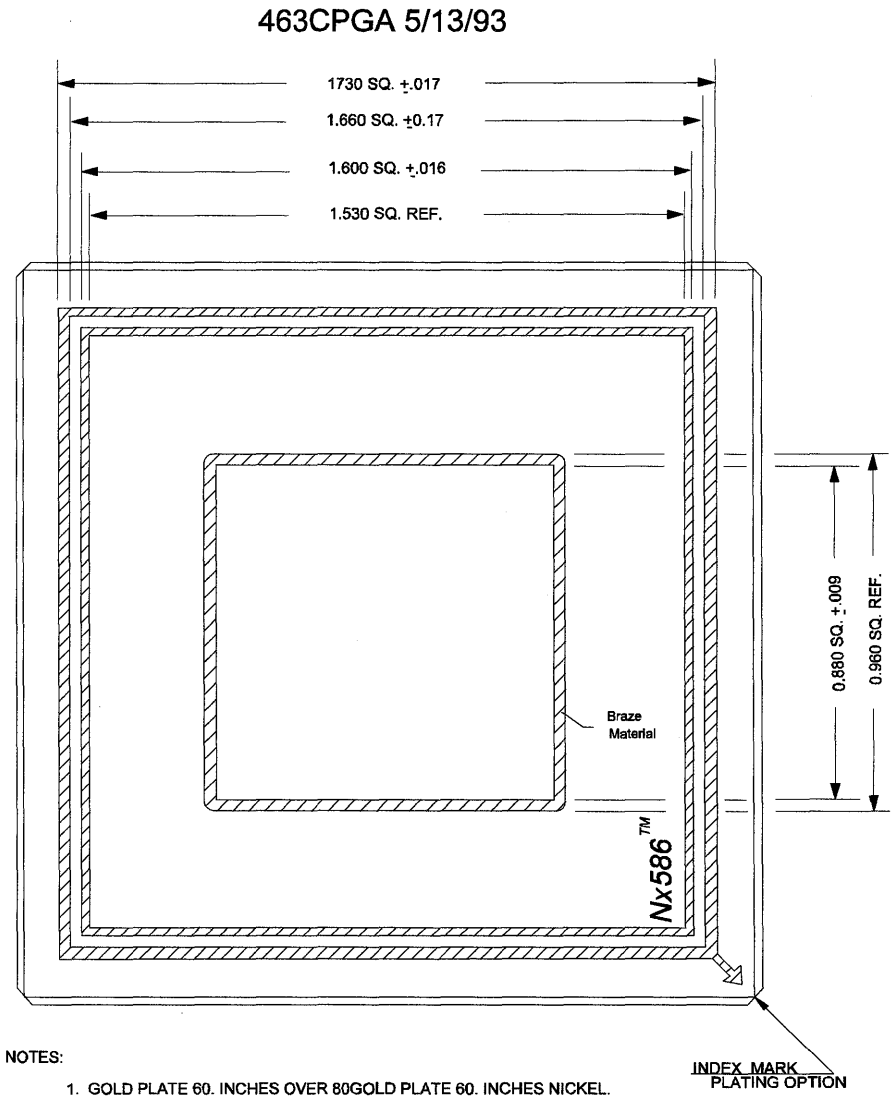
As described above for single-qword operations, a block read by another NexBus master may hit a modified cache block in the processor. When this happens, the processor responds exactly as for a single-qword operation: it asserts DCL*, waits for the assertion of GNT* following the negation of GXACK, and proceeds with a block write-back. It writes the entire four-qword block back to memory. The original bus master must sample the data in this second block operation while it is transferred to memory. The master may insert wait states by asserting XHLD*. Since the processor, as intervenor, begins its write-back with the address requested by the master, if the original block read is a four-qword operation, the master can intercept the data as it is transferred to memory and find it in the expected order.

Block writes can hit in a modified or exclusive cache block only if the operation was initiated by the DMA action of a disk controller, not by the processor. Since only complete block writes are permitted, no write-back is required and the processor invalidates its cache block.

Electrical Data

For Electrical Data See Document "Nx586/587 Electrical Specifications"
Order # NxDOC-ES001-01-W

Mechanical Data



NOTES:

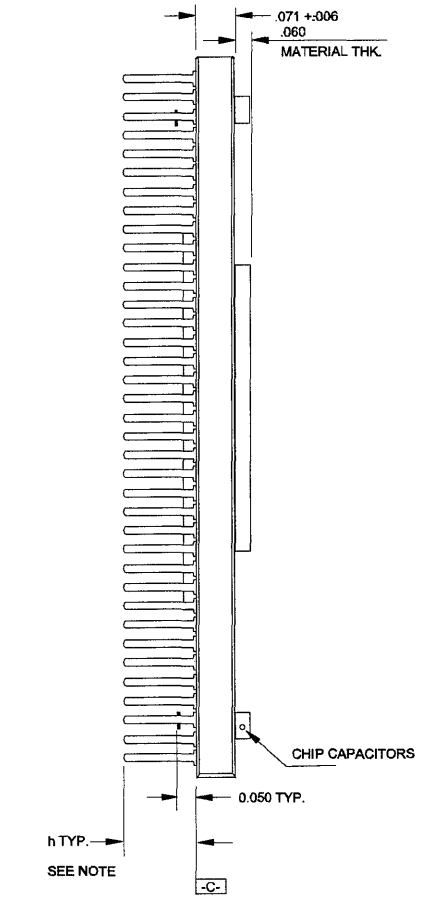
1. GOLD PLATE 60. INCHES OVER 80GOLD PLATE 60. INCHES NICKEL.
2. OPTION:

	h	
01	.156	PROTOTYPES
02	.155	PRODUCTION

3. LID AND HEATSINK ARE TIED TO GROUND
4. IF NOT INDICATED DIM TOLERANCE IS +/- 1%

Figure 38 Nx586 Package Diagram (top)

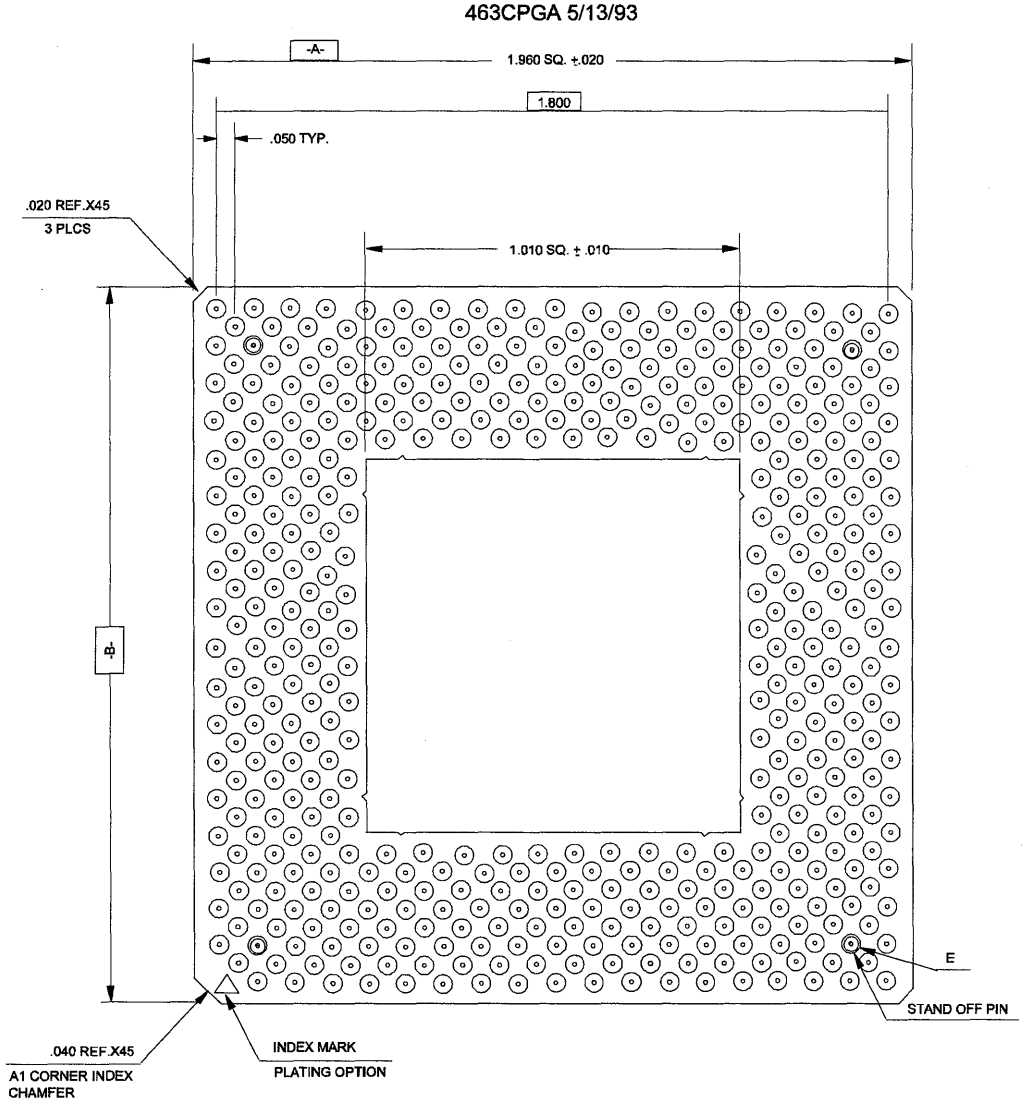
463CPGA 5/13/93



NOTES: OPTION:

	h	
01	.156	PROTOTYPES
02	.155	PRODUCTION

Figure 39 Nx586 Package Diagram (side)



NOTES:

1. GOLD PLATE 60. INCHES OVER 80GOLD PLATE 60. INCHES NICKEL.
2. OPTION:

	h	
01	.156	PROTOTYPES
02	.155	PRODUCTION

3. LID AND HEATSINK ARE TIED TO GROUND
4. IF NOT INDICATED DIM TOLERANCE IS +/- 1%

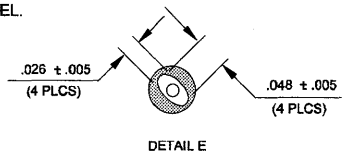


Figure 39 Nx586 Package Diagram (bottom)

183CPGA 5/13/93

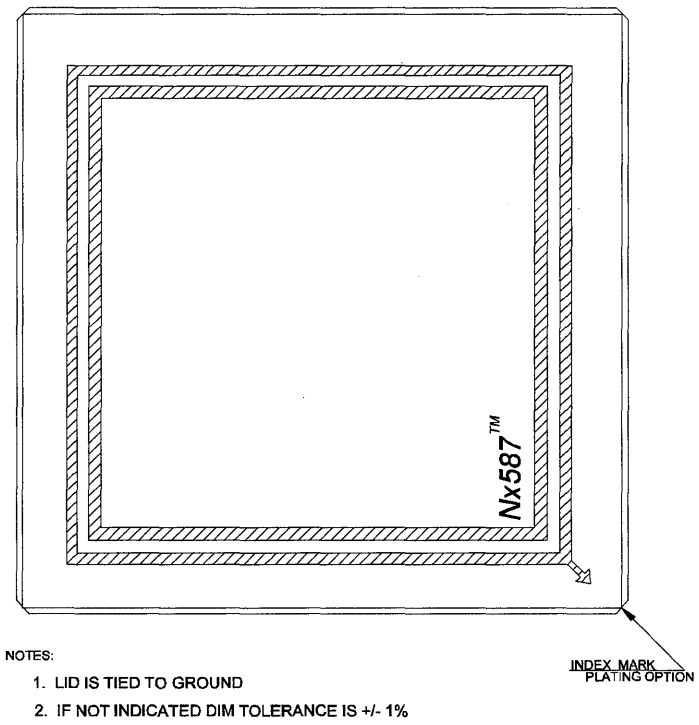


Figure 40 Nx587 Package Diagram (top)

183 CPGA 5/13/93

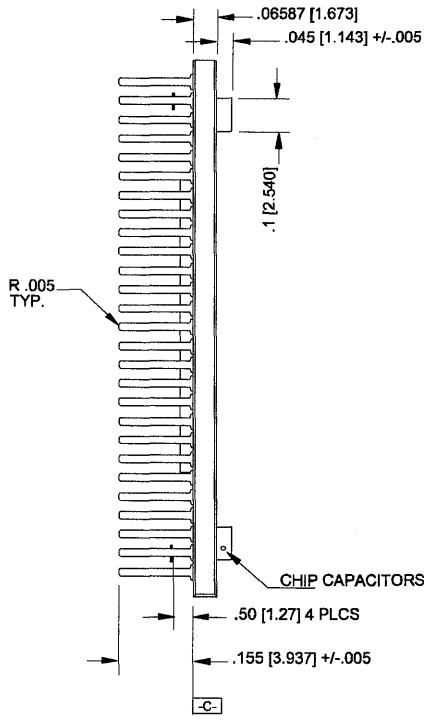


Figure 41 Nx587 Package Diagram (side)

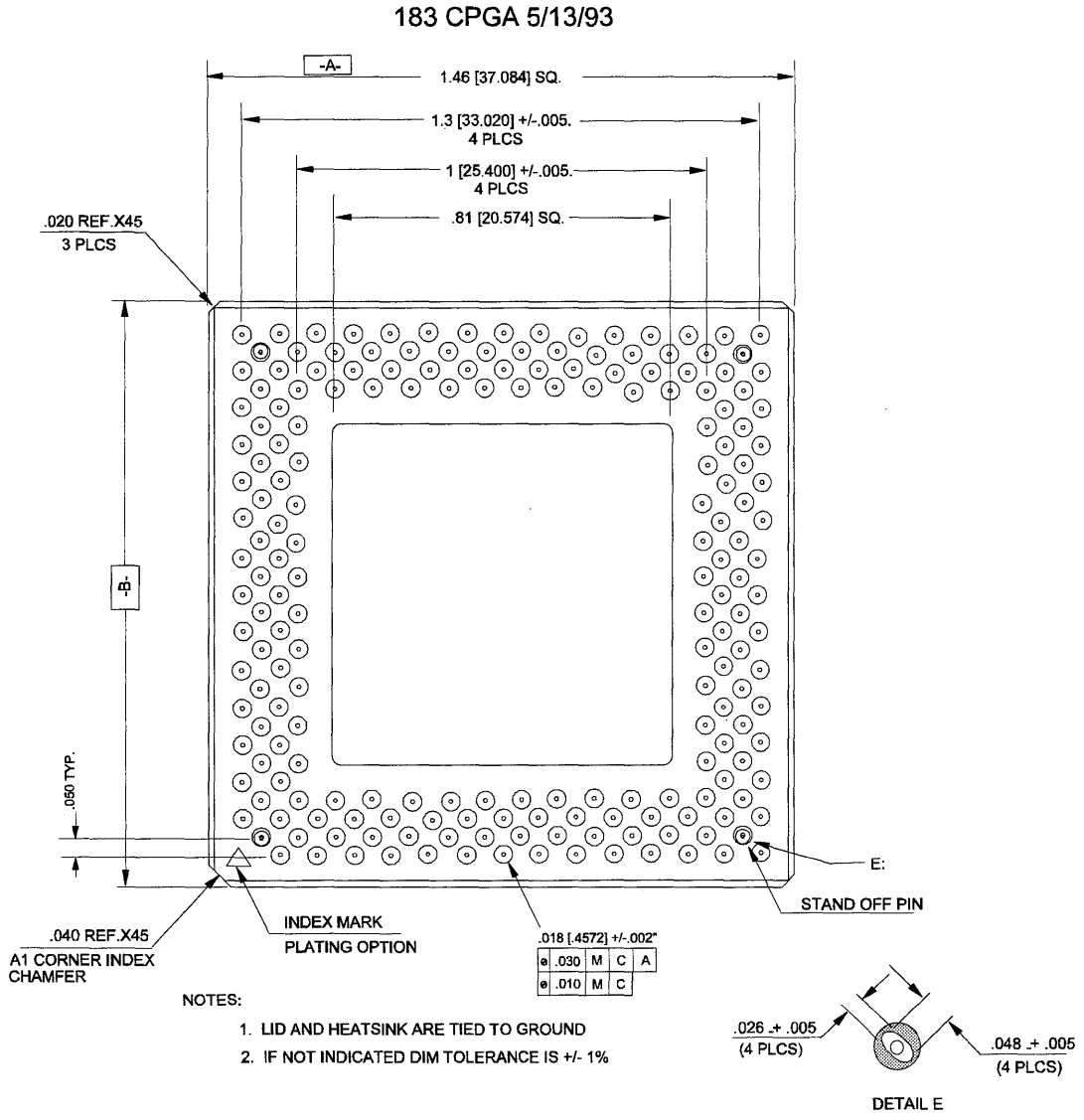


Figure 42 Nx587 Package Diagram (bottom)

Glossary

Access—A bus master is said to "have access to a bus" when it can initiate a *bus cycle* on that bus. Compare *bus ownership*.

Adapter—A central processor, memory subsystem, I/O device, or other device that is attached to a slot on the NexBus, VL-Bus, or ISA bus. Also called a *slot*.

Aligned—Data or instructions that have been rotated until the relevant bytes begin in the least-significant byte position.

Allocating Write—A read-to-own (read for exclusive ownership of cacheable data) followed by a write to the cache.

Arbiter—A resource-conflict resolver, such as the NexBus arbiter. The NxVL includes a NexBus arbiter.

b—Bit.

B—Byte.

Bank—In a cache, same as *set* and *way*. In main memory, a qword-wide group of addressable locations.

Bus Cycle—A complete transaction between a bus master and a slave. For the Nx586 processor, a bus cycle is typically composed of an address and status phase, a data phase, and any necessary idle phases. Also called a *bus operation*, or simply *operation*.

Bus Operation—Same as *bus cycle*.

Bus Ownership—A bus is said to be owned by a master when the master can initiate cycles on the bus. In systems supported by the NxVL, the NxVL arbitrates access to all buses. The master to which bus ownership is granted controls only its own interface with the NxVL. The NxVL, on behalf of that master, acts as a master on the other buses in the system. It does this so as to support the master in the event that a bus-crossing operation is requested. Compare *access*.

Bus Phase—Part of bus cycle that lasts one or more bus clocks. For example, it may be a transfer of address and status, a transfer of data, or idle clocks.

Bus Sequence—A sequence of bus cycles (or operations) that must occur sequentially due to their being explicitly locked by the continuous assertion of the master's AREQ* and/or LOCK* signals, or implicitly locked by the GDCL signal.

Cache Block—A 32-byte unit of data in a cache. The Nx586 processor's caches are organized around such blocks. Each cache block has an associated tag and MESI-protocol state. Cache blocks can be fetched atomically as a contiguous group of 32-bytes or in eight-byte subblock units. Compare *cache line*.

Cache-Block Tag—The high-order address bits of a cache block that identifies the area of memory from which it was copied. During a cache lookup, the high-order address bits of the processor's operand is compared with the tags of all blocks stored in the cache.

Cache Hit—An access to a cache block whose state is *modified, exclusive,* or *shared* (i.e., not *invalid).* Compare *cache miss.*

Cache Line—If a *cache block* can be fetched atomically (rather than in subblock units), the concepts of cache block and cache line are identical. However, in the Nx586 processor, cache blocks are often fetched in eight-byte subblock units, leaving only parts of the cache block valid. Compare *cache block.*

Cache Lookup—Comparison between a processor address and the cache tags and state bits in all four sets (ways) of a cache.

Cache Miss—An access to a cache block whose state is *invalid.* Compare *cache hit.*

Cache Subblock—An eight-byte (qword) sector of a 32-byte cache block, with state bits. Cache blocks can be fetched atomically (as a unit) or in eight-byte (qword) subblocks. See *cache block.* A cache subblock is sometimes called a *sector.*

Caching Master—A bus master that internally caches data originated elsewhere. The caching master must continually monitor the bus to guarantee cache coherency. Masters on buses other than the NexBus can maintain caches, but they must be write-through (not write-back) caches.

Clean—Same as *exclusive.*

Clock Cycle—Unless otherwise stated, this a *processor-clock cycle* rather than a bus-clock cycle. The Nx586 processor's clock runs at twice the frequency of the NexBus clock (CLK). The level-1 cache runs at the same frequency as the processor clock. The level-2 cache runs at the same frequency as the NexBus clock (CLK).

Clock Phase—One-half of a processor clock cycle.

Crossing Operation—Same as *bus-crossing operation.*

Cycle—See *bus cycle, clock cycle, bus phase,* and *clock phase.*

D Cache—The level-1 (L1) data cache.

Device—Same as *adapter.*

Dirty—Same as *modified.*

Dword—A doubleword. A four-byte (32-bit) unit of data that is addressed on an four-byte boundary. Also called a *dword* (doubleword). Same as *quad.*

Exclusive—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Exclusive* data is owned by a single caching device and is the only known-correct copy of data in the system. Also called *clean* data. When exclusive data is written over, it is called *modified* (or *dirty)* data.

Floating Point Coprocessor—The Nx587 Floating Point Coprocessor (NP) chip. The logic in the Floating Point Coprocessor is integrated into the parallel pipeline of the Nx586 processor.

Flush—(1) To write back a cache block to memory and invalidate the cache location, also called *write-back and invalidate*, or (2) to invalidate a storage location such as a register without writing the contents to any other location. This is an ambiguous term that is best not used.

Functional Unit—The Decode Unit, Address Unit, Integer Unit, Floating Point Coprocessor, or Cache and Memory Unit.

Group Signal—A NexBus control signal that represents the logical OR of several inputs. These signals typically have signal names that begin with the letter "G".

I Cache—The level-1 (L1) instruction cache.

Invalid—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Invalid* data is not correctly associated with the tag for its cache block.

Invalidate—To change the state of an cache block to *invalid.*

L1—The level-1 cache located on the Nx586 processor chip.

L2—The level-2 cache located in SRAM connected to the processor's SRAM bus and controlled by logic on the Nx586 processor.

Line—See *cache block.*

Main Memory—See *memory.*

Memory—A RAM or ROM subsystem located on any bus, including the *main memory* most directly accessible to a processor. In systems using the NxVL, main memory is the DRAM on the NxVL's memory bus. Also called *main memory.*

MESI—The cache-coherency protocol used in the Nx586 processor. In the protocol, cached blocks in the L2 write-back cache can have four states (modified, exclusive, shared, invalid), hence the acronym MESI. See *modified, exclusive, shared,* and *invalid.*

Modified Write-Once Protocol—The cache-coherency protocol used in the Nx586 processor. See *MESI.*

Modified—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Modified* data is *exclusive* data that has been written to after being read from lower-level memory, and is therefore the only valid copy of that data. Also called *dirty or stale.*

MWO—See *modified write-once protocol.*

NB—Same as *NexBus.*

NexBus—A 64-bit synchronous, multiplexed bus defined by NexGen.

No-Op—A single-qword operation with BE<7:0>* all negated. No-ops address no bytes and do nothing except consume processor cycles.

NP—Same as *Nx587* and *Floating Point Coprocessor.*

Nx586—The Nx586 processor (CPU).

Nx587—The Nx587 Floating Point Coprocessor (NP). See *Floating Point Coprocessor.*

NxVL—A NexBus system controller chip that supports a Nx586 processor or Nx586/587 pair, main memory, 82C206 peripheral controller, VL-Bus, and ISA bus.

Octet—Same as *qword.*

Operation—See *bus operation* and *micro-operation.*

Owned—A cache block whose state is *exclusive* (owned clean) or *modified* (owned dirty). See also *bus ownership.*

Ownership—See *bus ownership.*

Peripheral Controller—A chip that supports interrupts, DMA, timer/counters, and a real-time clock. The NxVL is designed to interface to an 82C206 peripheral controller.

Phase—See *bus phase* and *clock phase.*

PLL—Phase-locked loop.

Present—Same as *valid.*

Processor—Unless otherwise specified, refers to a Nx586 processor.

Processor Clock—The Nx586 processor clock. See *clock cycle.*

Qword—A quadword. An eight-byte unit of data that is addressed on an eight-byte boundary. Also called an *octet.*

Sector—Same as *cache subblock.*

Set—In a cache, one of the degrees of associativity. The group of cache blocks in such a set. Same as *bank* and *way.*

Shared—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Shared* data is valid data that can only be read, not written.

Snoop—To compare an address on a bus with a tag in a cache, so as to detect operations that are inconsistent with cache coherency.

Snoop Hit—A snoop in which the compared data is found to be in a *modified* state. Compare *snoop miss.*

Snoop Miss—A snoop in which the compared data is not found, or is found to be in a *shared* state. Compare *snoop hit.*

Source—In timing diagrams, the left-hand column of the diagram indicates the "source" of each signal. This is the chip that originated the signal as an output. When signals are driven by multiple sources, all sources are shown, in the order in which they drive the signal. The source of a signal that takes on a different name as it crosses buses through transceivers is shown as the transceivers overwhich the signals cross, subscripted with a symbol indicating the logic that originally output the signals. The source of group-ORed signals (such as GXACK) is likewise subscripted with a symbol indicating the logic that originally output the activating signal (such as XACK*).

Stale—Same as *modified.*

System Bus—A bus to which the NexBus interfaces. The NxVL supports two system buses, VL-Bus and ISA bus.

System Controller—The device or logic that provides NexBus arbitration and interfacing to main memory and any other buses in the system. The NxVL is a system controller.

T-Byte—An 80-bit floating-point number.

Word—An two-byte (16-bit) unit of data.

Index

NexGen™

1623 Buckeye Drive
Milpitas, CA 95035
Ph 1-800-8NEXGEN
Fax (408) 435-0262

18 bis rue du montceau
77133 Féricy (France)
Ph 33 (1) 64.23.68.65
Fax 33 (1) 64.23.61.91

Order # NxDOC-DB001-01-W