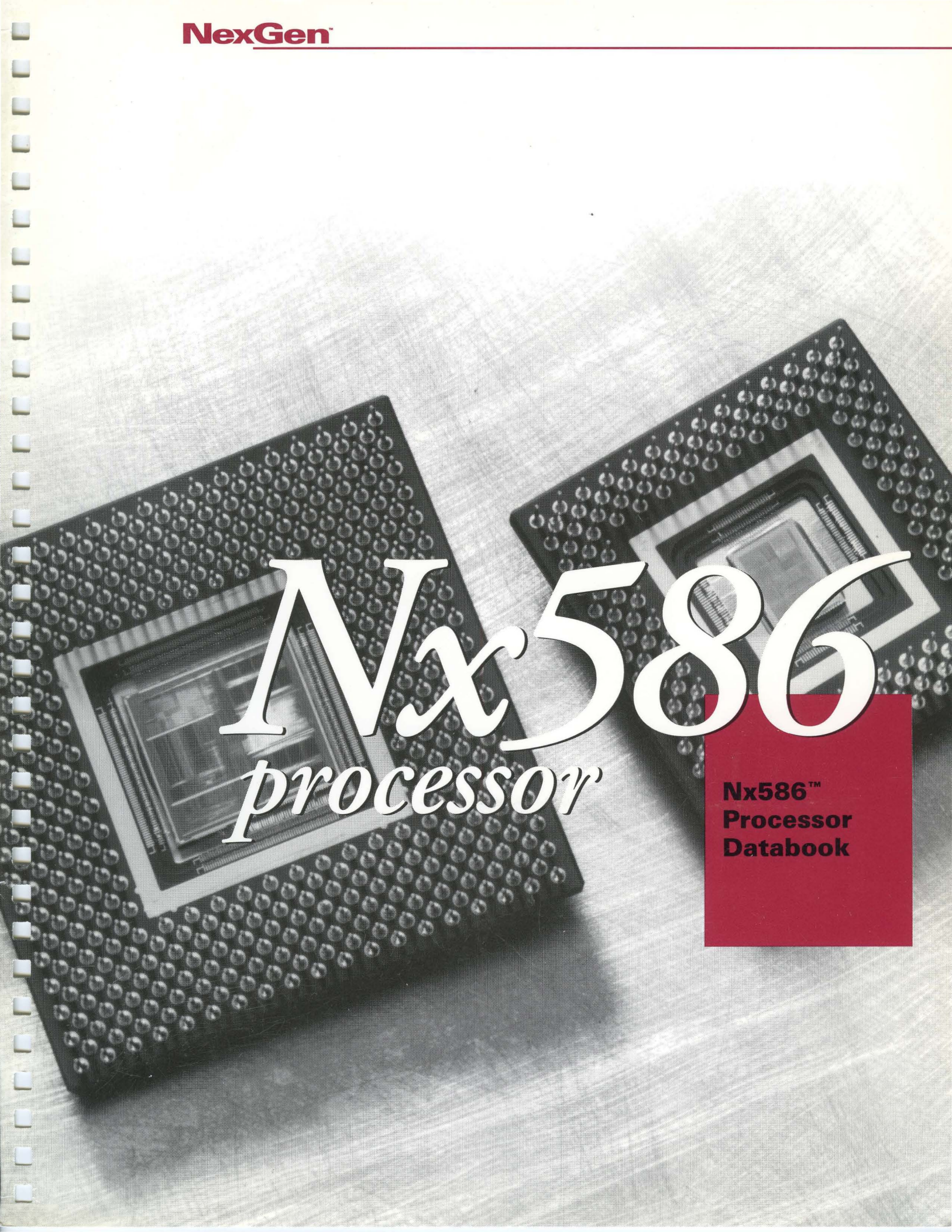
NexGen



Nx586 processor

Nx586™ Processor Databook

Nx586™ Processor Databook

PRELIMINARY
December 6, 1994

NexGen, Incorporated
1623 Buckeye Drive
Milpitas, CA 95035

Order # NxDOC-DB001-03-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, RISC86, NexBus, NxPCI, NxMC and NxVL are trademarks of NexGen, Inc.

IBM, PC AT, and PS/2 are registered trademarks of International Business Machines, Inc. Tri-state is a registered trademark of National Semiconductor Corporation. VESA and VL-Bus are 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 the Nx586™ processor (called *the processor).* 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.
- NxVL™ Systems Logic—The NxVL system controller described in the *NxVL System Controller Databook.*
- NxPCI™ Systems Logic—The NxPCI system controller described in the *NxPCI System Controller Databook.*
- NxMC™ Memory Logic—The NxMC memory controller described in the *NxMC Memory Controller Databook.*
- NexBus^5 System Bus—The NexGen system bus, including its multiplexed address/status and data bus (NxAD<63:0>) and related control signals.
- NexBus Processor Bus—The Nx586 processor bus, including its multiplexed address/status and data bus (AD<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.

- Active-High Signals—Signal names, such as GALE, that indicate active-high signals. They are said to be "asserted" or "active" in their high-voltage state and "negated" or "inactive" in their low-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.
- *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.
- *NxPCI System Controller Databook*, NexGen, Milpitas, CA, Tel: (408) 435-0202.
- *NxMC Memory Controller Databook*, NexGen, Milpitas, CA, Tel: (408) 435-0202.

Bus Standards

- VESA VL-Bus Version 2.0, Video Electronics Standards Association, San Jose CA 1993.
- PCI Local Bus Specification Revision 2.0, Peripheral Component Interconnect Special Interest Group, Hillsboro, Oregon, 1993.

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.

The Nx586 Processor Features

NexGen has independently developed a high performance x86 processor design utilizing state-of-the-art technologies. The Nx586 processor is the first implementation of NexGen's innovative and patented RISC86 microarchitecture and also includes the five key elements found in 5th generation processors: Superscalar execution, on-chip Harvard architecture L1 code and data caches, branch prediction, 64-bit wide buses, and advanced floating point capabilities. The NexGen Nx586 processor is an advanced 5th generation 32-bit Superscalar x86 compatible processor that provides market leading performance. The Nx586 represents the core building block 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 (Harvard Architecture) 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 internal Floating-Point Unit bus and is fully integrated into the processor microarchitecture.
- **NexBus and NexBus^5 Support**— The Nx586 supports both NexBus interface protocols.

Nx586 Processor with Floating-Point Execution Unit Features

The NexGen Nx586 Processor is available with an integrated floating-point execution unit. The floating-point execution unit is an expansion of the Nx586 superscalar pipelined microarchitecture. It adds specific x86 architecture floating point operations including arithmetic, exponential, logarithmic, and trigonometric functions. This execution unit is part of the RISC86 pipeline to ensure maximum floating-point calculation speed. This version of the Nx586 is plug compatible with the Nx586. The following are some of the key features:

- **Nx586 Feature Set**—Includes all the features of the Nx586.
- **MCM Technology**—The Processor and Floating-Point Unit are housed in a Multi-Chip-Module.
- **Fully Integrated Floating-Point unit into RISC86 Microarchitecture**—Operates in parallel with the Nx586 Address, and Integer Units. Increased performance due to Speculative floating-point requests.
- **Binary Compatible**—Runs all x86-architecture floating-point binary code.
- **Optional**— No hardware reconfiguration necessary if not present. Pin compatible with Nx586.
- **Dedicated Internal 64-Bit Processor Bus**—Fast, synchronous, non-multiplexed interface to Nx586 Processor Core.
- **High Bus Bandwidth**— Simple arbitration on the Floating-Point bus to maximize bandwidth. Arbitration and data transfers occur in parallel, one clock apart.

The Nx586 processor fully implements the industry standard x86 instruction set to be able to run the vast amount applications and operating systems available. This implementation is accomplished through the use of NexGen's patented RISC86 microarchitecture. The innovative RISC86 approach dynamically translates x86 instructions into RISC86 instructions. As shown in the figure below, the Nx586 takes advantage of RISC performance principles. Due to the RISC86 environment, each execution unit is more specialized, smaller and compact. The RISC86 microarchitecture contains many state-of-the-art computer science techniques to achieve very high performance, including Register Renaming, Data Forwarding, Speculative execution, and Out-of-Order execution.

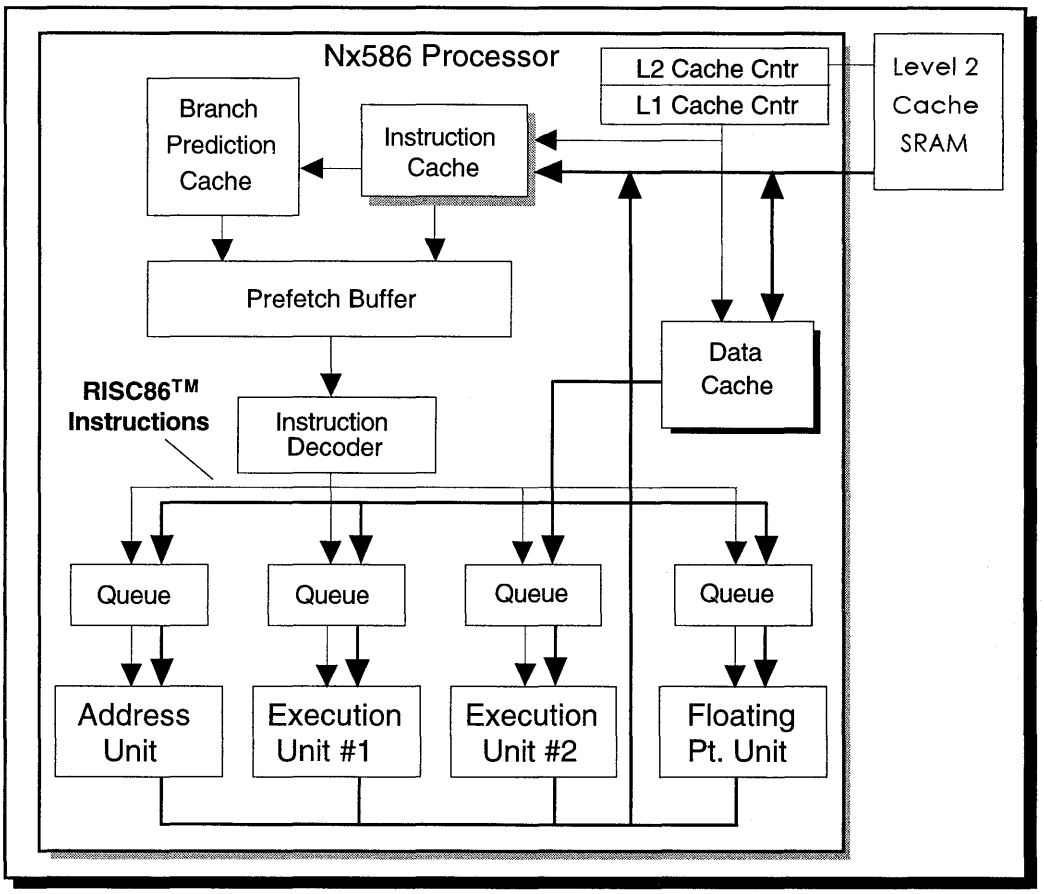


Figure 1 Nx586 Functional Block Diagram

The Level-2 cache controller is on chip for increased performance and reduced access overhead. The L2 cache controller does not have to arbitrate for the dedicated L2 Cache bus. L2 cache accesses can begin on any clock cycle. The greatest advantage comes when the CPU operating frequency is high. Accesses to the L2 cache remain at full speed and not at the slower system bus rate. Therefore, the Nx586 scales in performance linearly with respect to the operating frequency.

Nx586 Signals

Figure 2 shows the signal organization for the Nx586 processor. The processor core supports signals for NexBus (the processor bus) or NexBus^5 (the system bus), L2 cache, and the optional Floating-Point Unit. Many types of devices can be interfaced to the NexBus^5, including a backplane, multiple Nx586 processors, shared memory subsystems, high-speed I/O, and industry-standard buses. All signals are synchronous to the NexBus^5 clock (NxCLK) 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^5 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.

NexBus is the original processor protocol that defines how a localized processor, coprocessor and L2 cache are connected to the NexBus^5 system interface protocol in a multi-processor type of environment. Processors using the NexBus standard must provide bus transceivers to convert the NexBus interface to NexBus^5.

One type of NexBus signals deserve special mention:

- Buffered Address and Data Bus—Address, status and data phases are multiplexed on the AD<63:0> bus. This bus is interfaced to NexBus^5 through transceivers, for which control signals are provided by the processor.

Two types of NexBus^5 signals deserve special mention:

- *Group Signals*—There are several *group signals* on the NexBus^5, typically denoted by signal names beginning with the letter "G." Active-low signals such as ALE* are driven by each NexBus^5 device, and the NexBus^{5}arbiter derives an active-high group signal (such as GALE) and distributes it back to each device.
- *Central Bus Arbitration*—Access to the NexBus^5 is arbitrated by an external NexBus^5 Arbiter. NexBus^5 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. Typical systems logic will provide the combined functions of NexBus^5 Arbiter, and Alternate-Bus Interface (the system-logic interface to other system buses). The memory controller function may be included or designed as a separate device connected to NexBus^5.

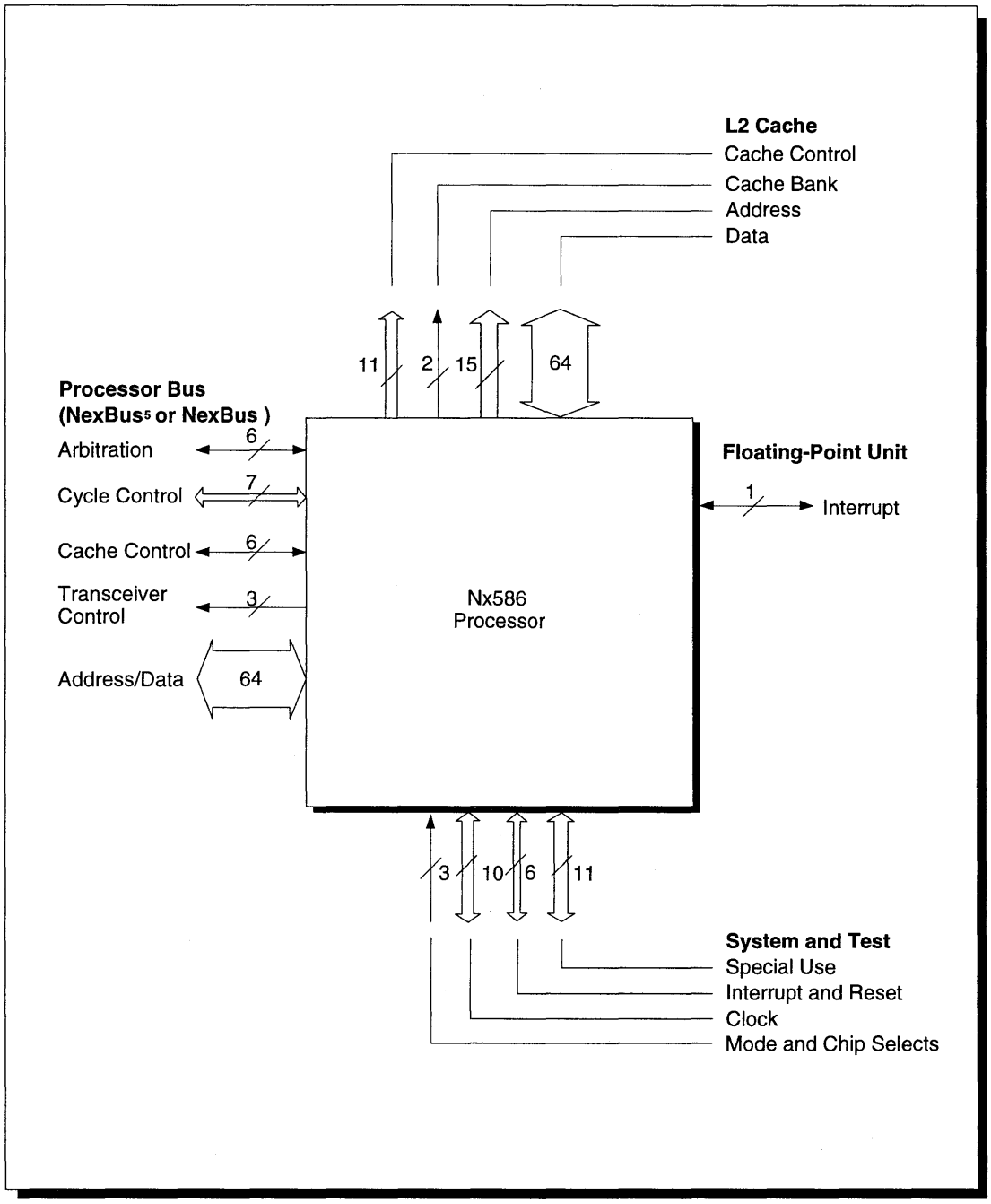


Figure 2 Nx586 Signal Organization

Nx586 Pinouts by Signal Names

PGA Pin#	JEDEC Pin#	Pin Type	Signal Name
449	J37	O	ALE*
18	AU1	I	ANALYZEIN
168	AL11	O	ANALYZEOUT
340	N31	O	AREQ*
34	AM2	O	CADDR<10>
90	AN5	O	CADDR<11>
107	AK6	O	CADDR<12>
88	AJ5	O	CADDR<13>
106	AH6	O	CADDR<14>
142	AF8	O	CADDR<15>
169	AN11	O	CADDR<16>
35	AP2	O	CADDR<17>
141	AD8	O	CADDR<3>
123	AE7	O	CADDR<4>
124	AF7	O	CADDR<5>
32	AH2	O	CADDR<6>
14	AJ1	O	CADDR<7>
33	AK2	O	CADDR<8>
15	AL1	O	CADDR<9>
89	AL5	O	CBANK<0>
16	AN1	O	CBANK<1>
100	T6	I/O	CDATA<0>
7	R1	I/O	CDATA<1>
27	V2	I/O	CDATA<10>
119	U7	I/O	CDATA<11>
118	R7	I/O	CDATA<12>
26	T2	I/O	CDATA<13>
82	U5	I/O	CDATA<14>
8	U1	I/O	CDATA<15>
11	AC1	I/O	CDATA<16>
103	AB6	I/O	CDATA<17>
29	K2	I/O	CDATA<18>
121	AA7	I/O	CDATA<19>
81	R5	I/O	CDATA<2>
139	Y8	I/O	CDATA<20>
28	Y2	I/O	CDATA<21>
102	Y6	I/O	CDATA<22>
10	AA1	I/O	CDATA<23>
13	AG1	I/O	CDATA<24>
105	AF6	I/O	CDATA<25>
31	AF2	I/O	CDATA<26>
86	AE5	I/O	CDATA<27>
122	AC7	I/O	CDATA<28>
85	AC5	I/O	CDATA<29>
136	P8	I/O	CDATA<3>
30	AD2	I/O	CDATA<30>
104	AD6	I/O	CDATA<31>
147	AT8	I/O	CDATA<32>
129	AU7	I/O	CDATA<33>
110	AT6	I/O	CDATA<34>
92	AU5	I/O	CDATA<35>
87	AG5	I/O	CDATA<36>
125	AJ7	I/O	CDATA<37>
176	AK12	I/O	CDATA<38>
184	AL13	I/O	CDATA<39>
24	M2	I/O	CDATA<4>
203	AU15	I/O	CDATA<40>
193	AM14	I/O	CDATA<41>
195	AT14	I/O	CDATA<42>
185	AN13	I/O	CDATA<43>
155	AU9	I/O	CDATA<44>
163	AT10	I/O	CDATA<45>
171	AU11	I/O	CDATA<46>
179	AT12	I/O	CDATA<47>
217	AN17	I/O	CDATA<48>
227	AT18	I/O	CDATA<49>
80	N5	I/O	CDATA<5>
225	AM18	I/O	CDATA<50>
224	AK18	I/O	CDATA<51>
201	AN15	I/O	CDATA<52>
211	AT16	I/O	CDATA<53>
209	AM16	I/O	CDATA<54>
219	AU17	I/O	CDATA<55>
240	AK20	I/O	CDATA<56>
251	AU21	I/O	CDATA<57>
249	AN21	I/O	CDATA<58>
248	AL21	I/O	CDATA<59>
6	N1	I/O	CDATA<6>
232	AL19	I/O	CDATA<60>
241	AM20	I/O	CDATA<61>
243	AT20	I/O	CDATA<62>
233	AN19	I/O	CDATA<63>
99	P6	I/O	CDATA<7>
9	W1	I/O	CDATA<8>
83	W5	I/O	CDATA<9>
361	V32	I	CKMODE
192	AK14	O	COEA*
138	V8	O	COEB*
117	N7	O	CWE<0>*
137	T8	O	CWE<1>*
120	W7	O	CWE<2>*
140	AB8	O	CWE<3>*
55	AU3	O	CWE<4>*
177	AM12	O	CWE<5>*
200	AL15	O	CWE<6>*
216	AL17	O	CWE<7>*
359	P32	O	DCL*
330	AK30	I	GALE
339	L31	I	GATEA20
378	R33	I	GBLKNBL
429	F36	I	GDCL
368	AM32	I	GNT*
113	E7	I	GREF
430	H36	I	GSHARE
322	P30	I	GTAL
349	AL31	I	GXACK
377	N33	I	GXHLD
36	AT2	I	HROM
375	J33	I	INTR*
323	T30	I	IREF
341	R31	O	LOCK*
1	C1	-	NC
2	E1	-	NC
3	G1	-	NC
4	J1	-	NC
5	L1	-	NC
12	AE1	-	NC
17	AR1	-	NC
19	B2	-	NC
20	D2	-	NC
21	F2	-	NC
22	H2	-	NC
23	K2	-	NC
25	P2	-	NC
37	A3	-	NC
56	B4	-	NC
74	A5	-	NC
77	G5	-	NC
78	J5	-	NC
79	L5	-	NC
84	AA5	-	NC
93	B6	-	NC
95	F6	-	NC
96	H6	-	NC
97	K6	-	NC
98	M6	-	NC
101	V6	-	NC
111	A7	-	NC
114	G7	-	NC
115	J7	-	NC
116	L7	-	NC
132	F8	-	NC
133	H8	-	NC
134	K8	-	NC
135	M8	-	NC
143	AH8	-	NC
148	A9	-	NC
151	G9	-	NC
156	B10	-	NC
158	F10	-	NC
159	H10	-	NC
164	A11	-	NC
166	E11	-	NC
167	G11	-	NC
172	B12	-	NC
174	F12	-	NC
175	H12	-	NC
180	A13	-	NC
182	E13	-	NC
183	G13	-	NC
187	AU13	-	NC
188	B14	-	NC
190	F14	-	NC
191	H14	-	NC
196	A15	-	NC
198	E15	-	NC
199	G15	-	NC
204	B16	-	NC
206	F16	-	NC
207	H16	-	NC
208	AK16	-	NC
212	A17	-	NC
215	G17	-	NC
220	B18	-	NC

Figure 3 Nx586 Pin List, By Signal Name

PGA Pin#	JEDEC Pin#	Pin Type	Signal Name
222	F18	-	NC
223	H18	-	NC
228	A19	-	NC
230	E19	-	NC
231	G19	-	NC
235	AU19	-	NC
236	B20	-	NC
238	F20	-	NC
239	H20	-	NC
244	A21	-	NC
246	E21	-	NC
247	G21	-	NC
252	B22	-	NC
254	F22	-	NC
255	H22	-	NC
256	AK22	-	NC
260	A23	-	NC
262	E23	-	NC
263	G23	-	NC
268	B24	-	NC
270	F24	-	NC
271	H24	-	NC
276	A25	-	NC
278	E25	-	NC
279	G25	-	NC
284	B26	-	NC
286	F26	-	NC
287	H26	-	NC
288	AK26	-	NC
292	A27	-	NC
294	E27	-	NC
295	G27	-	NC
300	B28	-	NC
302	F28	-	NC
303	H28	-	NC
308	A29	-	NC
310	E29	-	NC
311	G29	-	NC
316	B30	-	NC
318	F30	-	NC
334	A31	-	NC
336	E31	-	NC
337	G31	-	NC
338	J31	-	NC
353	B32	-	NC
356	H32	-	NC
371	A33	-	NC
373	E33	-	NC
380	W33	-	NC
390	B34	-	NC
408	A35	-	NC
427	B36	-	NC
428	D36	-	NC
445	A37	-	NC
436	Y36	I	NMI*
446	C37	O	NPIRQ*
321	M30	O	NREQ*
296	AL27	I/O	NxAD<0>
267	AU23	I/O	NxAD<1>
307	AT28	I/O	NxAD<2>
297	AN27	I/O	NxAD<3>
443	AP36	I/O	NxAD<4>
444	AT36	I/O	NxAD<5>
463	AU37	I/O	NxAD<6>
312	AL29	I/O	NxAD<7>
313	AN29	I/O	NxAD<8>
315	AU29	I/O	NxAD<9>
281	AN25	I/O	NxAD<10>
283	AU25	I/O	NxAD<11>
459	AJ37	I/O	NxAD<12>
460	AL37	I/O	NxAD<13>
441	AK36	I/O	NxAD<14>
348	AJ31	I/O	NxAD<15>
387	AN33	I/O	NxAD<16>
370	AT32	I/O	NxAD<17>
331	AM30	I/O	NxAD<18>
333	AT30	I/O	NxAD<19>
325	Y30	I/O	NxAD<20>
345	AC31	I/O	NxAD<21>
327	AD30	I/O	NxAD<22>
383	AE33	I/O	NxAD<23>
347	AG31	I/O	NxAD<24>
384	AG33	I/O	NxAD<25>
458	AG37	I/O	NxAD<26>
346	AE31	I/O	NxAD<27>
438	AD36	I/O	NxAD<28>
382	AC33	I/O	NxAD<29>
437	AB36	I/O	NxAD<30>
455	AA37	I/O	NxAD<31>
259	AT22	I/O	NxAD<32>
257	AM22	I/O	NxAD<33>
265	AN23	I/O	NxAD<34>
275	AT24	I/O	NxAD<35>
273	AM24	I/O	NxAD<36>
462	AR37	I/O	NxAD<37>
304	AK28	I/O	NxAD<38>
426	AU35	I/O	NxAD<39>
299	AU27	I/O	NxAD<40>
289	AM26	I/O	NxAD<41>
291	AT26	I/O	NxAD<42>
305	AM28	I/O	NxAD<43>
440	AH36	I/O	NxAD<44>
366	AH32	I/O	NxAD<45>
367	AK32	I/O	NxAD<46>
385	AJ33	I/O	NxAD<47>
407	AT34	I/O	NxAD<48>
389	AU33	I/O	NxAD<49>
350	AN31	I/O	NxAD<50>
352	AU31	I/O	NxAD<51>
343	W31	I/O	NxAD<52>
344	AA31	I/O	NxAD<53>
326	AB30	I/O	NxAD<54>
457	AE37	I/O	NxAD<55>
329	AH30	I/O	NxAD<56>
328	AF30	I/O	NxAD<57>
365	AF32	I/O	NxAD<58>
439	AF36	I/O	NxAD<59>
364	AD32	I/O	NxAD<60>
456	AC37	I/O	NxAD<61>
363	AB32	I/O	NxAD<62>
381	AA33	I/O	NxAD<63>
73	AT4	O	NxADINUSE
452	R37	I	NxCLK
447	E37	I	OWNABL
76	E5	I	P4REF
453	U37	I	PHE1
379	U33	I	PHE2
153	AN9	I	POPHOLD
272	AK24	I	PTEST
319	H30	I	PULLDOWN
355	F32	I	PULLDOWN
160	AK10	I	PULLHIGH
145	AM8	I	PULLHIGH
357	K32	I/O	PULLHIGH
376	L33	I/O	PULLHIGH
432	M36	I/O	PULLHIGH
433	P36	I/O	PULLHIGH
450	L37	I/O	PULLHIGH
451	N37	I/O	PULLHIGH
264	AL23	I/O	PULLLOW
214	E17	I	RESET*
362	Y32	I	RESETCPU*
150	E9	I/O	SCLKE
144	AK8	I	SERIALIN
280	AL25	O	SERIALOUT
448	G37	O	SHARE*
130	B8	I	SLOTID<0>
161	AM10	I	SLOTID<1>
152	AL9	I	SLOTID<2>
127	AN7	I	SLOTID<3>
431	K36	I/O	SRAMMODE
374	G33	I	TESTPWR*
108	AM6	I	TPH1
126	AL7	I	TPH2
57	E4	I	VCC4
58	F4	I	VCC4
59	H4	I	VCC4
60	K4	I	VCC4
61	M4	I	VCC4
62	P4	I	VCC4
63	T4	I	VCC4
64	V4	I	VCC4
65	Y4	I	VCC4
66	AB4	I	VCC4
67	AD4	I	VCC4
68	AF4	I	VCC4
69	AH4	I	VCC4
70	AK4	I	VCC4
71	AM4	I	VCC4
72	AP4	I	VCC4
94	D6	I	VCC4
109	AP6	I	VCC4
131	D8	I	VCC4
146	AP8	I	VCC4

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

PGA Pin#	JEDEC Pin#	Pin Type	Signal Name
157	D10	I	VCC4
162	AP10	I	VCC4
173	D12	I	VCC4
178	AP12	I	VCC4
189	D14	I	VCC4
194	AP14	I	VCC4
205	D16	I	VCC4
210	AP16	I	VCC4
221	D18	I	VCC4
226	AP18	I	VCC4
237	D20	I	VCC4
242	AP20	I	VCC4
253	D22	I	VCC4
258	AP22	I	VCC4
269	D24	I	VCC4
274	AP24	I	VCC4
285	D26	I	VCC4
290	AP26	I	VCC4
301	D28	I	VCC4
306	AP28	I	VCC4
317	D30	I	VCC4
332	AP30	I	VCC4
354	D32	I	VCC4
369	AP32	I	VCC4
391	D34	I	VCC4
392	F34	I	VCC4
393	H34	I	VCC4
394	K34	I	VCC4
395	M34	I	VCC4
396	P34	I	VCC4
397	T34	I	VCC4
398	V34	I	VCC4
399	Y34	I	VCC4
400	AB34	I	VCC4
401	AD34	I	VCC4
402	AF34	I	VCC4
403	AH34	I	VCC4
404	AK34	I	VCC4
405	AM34	I	VCC4
406	AP34	I	VCC4

PGA Pin#	JEDEC Pin#	Pin Type	Signal Name
324	V30	I	VDDA
38	C3	I	VSS
39	E3	I	VSS
40	G3	I	VSS
41	J3	I	VSS
42	L3	I	VSS
43	N3	I	VSS
44	R3	I	VSS
45	U3	I	VSS
46	W3	I	VSS
47	AA3	I	VSS
48	AC3	I	VSS
49	AE3	I	VSS
50	AG3	I	VSS
51	AJ3	I	VSS
52	AL3	I	VSS
53	AN3	I	VSS
54	AR3	I	VSS
75	C5	I	VSS
91	AR5	I	VSS
112	C7	I	VSS
128	AR7	I	VSS
149	C9	I	VSS
154	AR9	I	VSS
165	C11	I	VSS
170	AR11	I	VSS
181	C13	I	VSS
186	AR13	I	VSS
197	C15	I	VSS
202	AR15	I	VSS
213	C17	I	VSS
218	AR17	I	VSS
229	C19	I	VSS
234	AR19	I	VSS
245	C21	I	VSS
250	AR21	I	VSS
54	AR3	I	VSS
75	C5	I	VSS
91	AR5	I	VSS
112	C7	I	VSS

PGA Pin#	JEDEC Pin#	Pin Type	Signal Name
261	C23	I	VSS
266	AR23	I	VSS
277	C25	I	VSS
282	AR25	I	VSS
293	C27	I	VSS
298	AR27	I	VSS
309	C29	I	VSS
314	AR29	I	VSS
335	C31	I	VSS
351	AR31	I	VSS
372	C33	I	VSS
388	AR33	I	VSS
409	C35	I	VSS
410	E35	I	VSS
411	G35	I	VSS
412	J35	I	VSS
413	L35	I	VSS
414	N35	I	VSS
415	R35	I	VSS
416	U35	I	VSS
417	W35	I	VSS
418	AA35	I	VSS
419	AC35	I	VSS
420	AE35	I	VSS
421	AG35	I	VSS
422	AJ35	I	VSS
423	AL35	I	VSS
424	AN35	I	VSS
425	AR35	I	VSS
358	M32	O	XACK*
386	AL33	O	XBCKE*
461	AN37	O	XBOE*
320	K30	I/O	XCVERE*
454	W37	O	XHLD*
442	AM36	O	XNOE*
360	T32	O	XPH1
342	U31	O	XPH2
434	T36	O	XREF
435	V36	I	XSEL

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

Nx586 Pinouts by PGA Pin Numbers

PGA Pin	Pin Type	Signal Name
1	-	NC
2	-	NC
3	-	NC
4	-	NC
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	-	NC
20	-	NC
21	-	NC
22	-	NC
23	-	NC
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	-	NC
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	-	NC
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	-	NC
75	I	VSS
76	I	P4REF
77	-	NC
78	-	NC
79	-	NC
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	-	NC
94	I	VCC4
95	-	NC
96	-	NC
97	-	NC
98	-	NC
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
113	I	GREF
114	I	PULLHIGH
115	-	NC
116	-	NC
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	-	NC
133	I	PULLHIGH
134	-	NC
135	-	NC
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	-	NC
149	I	VSS
150	I/O	SCLKE
151	-	NC
152	I	SLOTID<2>
153	I	POPHOLD
154	I	VSS
155	I/O	CDATA<44>
156	-	NC
157	I	VCC4
158	-	NC
159	-	NC
160	I	PULLHIGH
161	I	SLOTID<1>
162	I	VCC4
163	I/O	CDATA<45>
164	-	NC
165	I	VSS
166	-	NC
167	-	NC
168	O	ANALYZEOUT
169	O	CADDR<16>
170	I	VSS
171	I/O	CDATA<46>
172	-	NC
173	I	VCC4
174	-	NC
175	-	NC
176	I/O	CDATA<38>
177	O	CWE<5>*
178	I	VCC4
179	I/O	CDATA<47>
180	-	NC
181	I	VSS
182	-	NC
183	-	NC
184	I/O	CDATA<39>
185	I/O	CDATA<43>
186	I	VSS
187	-	NC
188	-	NC
189	I	VCC4
190	-	NC
191	-	NC
192	O	COEA*
193	I/O	CDATA<41>
194	I	VCC4
195	I/O	CDATA<42>
196	-	NC
197	I	VSS
198	-	NC
199	-	NC
200	O	CWE<6>*
201	I/O	CDATA<52>
202	I	VSS
203	I/O	CDATA<40>
204	-	NC
205	I	VCC4
206	-	NC
207	-	NC
208	-	NC
209	I/O	CDATA<54>
210	I	VCC4
211	I/O	CDATA<53>
212	-	NC
213	I	VSS
214	I	RESET*
215	-	NC
216	O	CWE<7>*
217	I/O	CDATA<48>
218	I	VSS
219	I/O	CDATA<55>
220	-	NC
221	I	VCC4
222	-	NC
223	-	NC
224	I/O	CDATA<51>

Figure 4 Nx586 Pin List, By PGA Pin Number

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

PGA Pin	Pin Type	Signal Name
285	I	VCC4
286	-	NC
287	-	NC
288	-	NC
289	I/O	NxAD<41>
290	I	VCC4
291	I/O	NxAD<42>
292	-	NC
293	I	VSS
294	-	NC
295	-	NC
296	I/O	NxAD<0>
297	I/O	NxAD<3>
298	I	VSS
299	I/O	NxAD<40>
300	-	NC
301	I	VCC4
302	-	NC
303	-	NC
304	I/O	NxAD<38>
305	I/O	NxAD<43>
306	I	VCC4
307	I/O	NxAD<2>
308	-	NC
309	I	VSS
310	-	NC
311	-	NC
312	I/O	NxAD<7>
313	I/O	NxAD<8>
314	I	VSS
315	I/O	NxAD<9>
316	-	NC
317	I	VCC4
318	-	NC
319	I	PULLDOWN
320	I	XCVERE*
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	-	NC
335	I	VSS
336	-	NC
337	I	PULLHIGH
338	-	NC
339	I	GATEA20
340	O	AREQ*
341	O	LOCK*
342	O	XPH2
343	I/O	NxAD<52>
344	I/O	NxAD<53>

PGA Pin	Pin Type	Signal Name
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	-	NC
354	I	VCC4
355	I	PULLDOWN
356	-	NC
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	-	NC
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

PGA Pin	Pin Type	Signal Name
405	I	VCC4
406	I	VCC4
407	I/O	NxAD<48>
408	-	NC
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	--	NC
428	--	NC
429	I	GDCL
430	I	GSHARE
431	I/O	SRAMMODE
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	-	NC
446	O	NPIRQ*
447	I	OWNABL
448	O	SHARE*
449	O	ALE*
450	I/O	PULLHIGH
451	I/O	PULLHIGH
452	I	NxCLK
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 4 Nx586 Pin List, By PGA Pin Number (continued)

Nx586 Pinouts by JEDEC Pin Numbers

JEDEC Pin	Pin Type	Signal Name
A3	-	NC
A5	-	NC
A7	I/O	NPTAG<3>
A9	-	NC
A11	-	NC
A13	-	NC
A15	-	NC
A17	-	NC
A19	-	NC
A21	-	NC
A23	-	NC
A25	-	NC
A27	-	NC
A29	-	NC
A31	-	NC
A33	I/O	NPDATA<54>
A35	-	NC
A37	-	NC
B2	-	NC
B4	-	NC
B6	-	NC
B8	I	SLOTID<0>
B10	-	NC
B12	-	NC
B14	-	NC
B16	-	NC
B18	-	NC
B20	-	NC
B22	-	NC
B24	-	NC
B26	-	NC
B28	-	NC
B30	-	NC
B32	-	NC
B34	-	NC
B36	--	NC
C1	-	NC
C3	I	VSS
C5	I	VSS
C7	I	VSS
C9	I	VSS
C11	I	VSS
C13	I	VSS
C15	I	VSS
C17	I	VSS
C19	I	VSS
C21	I	VSS
C23	I	VSS
C25	I	VSS
C27	I	VSS
C29	I	VSS
C31	I	VSS
C33	I	VSS
C35	I	VSS
C37	O	NPIRQ*
D2	-	NC

JEDEC Pin	Pin Type	Signal Name
D4	I	VCC4
D6	I	VCC4
D8	I	VCC4
D10	I	VCC4
D12	I	VCC4
D14	I	VCC4
D16	I	VCC4
D18	I	VCC4
D20	I	VCC4
D22	I	VCC4
D24	I	VCC4
D26	I	VCC4
D28	I	VCC4
D30	I	VCC4
D32	I	VCC4
D34	I	VCC4
D36	--	NC
E1	-	NC
E3	I	VSS
E5	I	P4REF
E7	I	GREF
E9	I/O	SCLKE
E11	-	NC
E13	-	NC
E15	-	NC
E17	I	RESET*
E19	-	NC
E21	-	NC
E23	-	NC
E25	-	NC
E27	-	NC
E29	-	NC
E31	-	NC
E33	I/O	NPDATA<42>
E35	I	VSS
E37	I	OWNABL
F2	-	NC
F4	I	VCC4
F6	-	NC
F8	-	NC
F10	-	NC
F12	-	NC
F14	-	NC
F16	-	NC
F18	-	NC
F20	-	NC
F22	-	NC
F24	-	NC
F26	-	NC
F28	-	NC
F30	-	NC
F32	I	PULLDOWN
F34	I	VCC4
F36	I	GDCL
G1	-	NC
G3	I	VSS

JEDEC Pin	Pin Type	Signal Name
G5	-	NC
G7	I	PULLHIGH
G9	-	NC
G11	-	NC
G13	-	NC
G15	-	NC
G17	-	NC
G19	-	NC
G21	-	NC
G23	-	NC
G25	-	NC
G27	-	NC
G29	-	NC
G31	I	PULLHIGH
G33	I	TESTPWR*
G35	I	VSS
G37	O	SHARE*
H2	-	NC
H4	I	VCC4
H6	-	NC
H8	I	PULLHIGH
H10	-	NC
H12	-	NC
H14	-	NC
H16	-	NC
H18	-	NC
H20	-	NC
H22	-	NC
H24	-	NC
H26	-	NC
H28	-	NC
H30	I	PULLDOWN
H32	-	NC
H34	I	VCC4
H36	I	GSHARE
J1	-	NC
J3	I	VSS
J5	-	NC
J7	-	NC
J31	-	NC
J33	I	INTR*
J35	I	VSS
J37	O	ALE*
K2	-	NC
K4	I	VCC4
K6	-	NC
K8	-	NC
K30	I/O	XCVERE*
K32	I/O	PULLHIGH
K34	I	VCC4
K36	I/O	SRAMMODE
L1	I/O	NPTAG<0>
L3	I	VSS
L5	-	NC
L7	-	NC
L31	I	GATEA20

JEDEC Pin	Pin Type	Signal Name
L33	I/O	PULLHIGH
L35	I	VSS
L37	I/O	PULLHIGH
M2	I/O	CDATA<4>
M4	I	VCC4
M6	-	NC
M8	-	NC
M30	O	NREQ*
M32	O	XACK*
M34	I	VCC4
M36	I/O	PULLHIGH
N1	I/O	CDATA<6>
N3	I	VSS
N5	I/O	CDATA<5>
N7	O	CWE<0>*
N31	O	AREQ*
N33	I	GXHLD
N35	I	VSS
N37	I/O	PULLHIGH
P2	-	NC
P4	I	VCC4
P6	I/O	CDATA<7>
P8	I/O	CDATA<3>
P30	I	GTAL
P32	O	DCL*
P34	I	VCC4
P36	I/O	PULLHIGH
R1	I/O	CDATA<1>
R3	I	VSS
R5	I/O	CDATA<2>
R7	I/O	CDATA<12>
R31	O	LOCK*
R33	I	GBLKNBL
R35	I	VSS
R37	I	NxCLK
T2	I/O	CDATA<13>
T4	I	VCC4
T6	I/O	CDATA<0>
T8	O	CWE<1>*
T30	I	IREF
T32	O	XPH1
T34	I	VCC4
T36	O	XREF
U1	I/O	CDATA<15>
U3	I	VSS
U5	I/O	CDATA<14>
U7	I/O	CDATA<11>
U31	O	XPH2
U33	I	PHE2
U35	I	VSS
U37	I	PHE1
V2	I/O	CDATA<10>
V4	I	VCC4
V6	-	NC
V8	O	COEB*
V30	I	VDDA

Figure 5 Nx586 Pin List, By JEDEC Pin Number

JEDEC Pin	Pin Type	Signal Name
V32	I	CKMODE
V34	I	VCC4
V36	I	XSEL
W1	I/O	CDATA<8>
W3	I	VSS
W5	I/O	CDATA<9>
W7	O	CWE<2>*
W31	I/O	NxAD<52>
W33	-	NC
W35	I	VSS
W37	O	XHLD*
Y2	I/O	CDATA<21>
Y4	I	VCC4
Y6	I/O	CDATA<22>
Y8	I/O	CDATA<20>
Y30	I/O	NxAD<20>
Y32	I	RESETCPU*
Y34	I	VCC4
Y36	I	NMI*
AA1	I/O	CDATA<23>
AA3	I	VSS
AA5	-	NC
AA7	I/O	CDATA<19>
AA31	I/O	NxAD<53>
AA33	I/O	NxAD<63>
AA35	I	VSS
AA37	I/O	NxAD<31>
AB2	I/O	CDATA<18>
AB4	I	VCC4
AB6	I/O	CDATA<17>
AB8	O	CWE<3>*
AB30	I/O	NxAD<54>
AB32	I/O	NxAD<62>
AB34	I	VCC4
AB36	I/O	NxAD<30>
AC1	I/O	CDATA<16>
AC3	I	VSS
AC5	I/O	CDATA<29>
AC7	I/O	CDATA<28>
AC31	I/O	NxAD<21>
AC33	I/O	NxAD<29>
AC35	I	VSS
AC37	I/O	NxAD<61>
AD2	I/O	CDATA<30>
AD4	I	VCC4
AD6	I/O	CDATA<31>
AD8	O	CADDR<3>
AD30	I/O	NxAD<22>
AD32	I/O	NxAD<60>
AD34	I	VCC4
AD36	I/O	NxAD<28>
AE1	-	NC
AE3	I	VSS
AE5	I/O	CDATA<27>
AE7	O	CADDR<4>
AE31	I/O	NxAD<27>
AE33	I/O	NxAD<23>
AE35	I	VSS
AE37	I/O	NxAD<55>
AF2	I/O	CDATA<26>

JEDEC Pin	Pin Type	Signal Name
AF4	I	VCC4
AF6	I/O	CDATA<25>
AF8	O	CADDR<15>
AF30	I/O	NxAD<57>
AF32	I/O	NxAD<58>
AF34	I	VCC4
AF36	I/O	NxAD<59>
AG1	I/O	CDATA<24>
AG3	I	VSS
AG31	I/O	NxAD<24>
AG5	I/O	CDATA<36>
AG7	O	CADDR<5>
AG33	I/O	NxAD<25>
AG35	I	VSS
AG37	I/O	NxAD<26>
AH2	O	CADDR<6>
AH4	I	VCC4
AH6	O	CADDR<14>
AH8	-	NC
AH30	I/O	NxAD<56>
AH32	I/O	NxAD<45>
AH34	I	VCC4
AH36	I/O	NxAD<44>
AJ1	O	CADDR<7>
AJ3	I	VSS
AJ5	O	CADDR<13>
AJ7	I/O	CDATA<37>
AJ31	I/O	NxAD<15>
AJ33	I/O	NxAD<47>
AJ35	I	VSS
AJ37	I/O	NxAD<12>
AK2	O	CADDR<8>
AK4	I	VCC4
AK6	O	CADDR<12>
AK8	I	SERIALIN
AK10	I	PULLHIGH
AK12	I/O	CDATA<38>
AK14	O	COEA*
AK16	-	NC
AK18	I/O	CDATA<51>
AK20	I/O	CDATA<56>
AK22	-	NC
AK24	I	PTEST
AK26	-	NC
AK28	I/O	NxAD<38>
AK30	I	GALE
AK32	I/O	NxAD<46>
AK34	I	VCC4
AK36	I/O	NxAD<14>
AL1	O	CADDR<9>
AL3	I	VSS
AL5	O	CBANK<0>
AL7	I	TPH2
AL9	I	SLOTID<2>
AL11	O	ANALYZEOUT
AL13	I/O	CDATA<39>
AL15	O	CWE<6>*
AL17	O	CWE<7>*
AL19	I/O	CDATA<60>
AL21	I/O	CDATA<59>

JEDEC Pin	Pin Type	Signal Name
AL23	I/O	PULLLOW
AL25	O	SERIALOUT
AL27	I/O	NxAD<0>
AL29	I/O	NxAD<7>
AL31	I	GXACK
AL33	O	XBCKE*
AL35	I	VSS
AL37	I/O	NxAD<13>
AM2	O	CADDR<10>
AM4	I	VCC4
AM6	I	TPH1
AM8	I	PULLHIGH
AM10	I	SLOTID<1>
AM12	O	CWE<5>*
AM14	I/O	CDATA<41>
AM16	I/O	CDATA<54>
AM18	I/O	CDATA<50>
AM20	I/O	CDATA<61>
AM22	I/O	NxAD<33>
AM24	I/O	NxAD<36>
AM26	I/O	NxAD<41>
AM28	I/O	NxAD<43>
AM30	I/O	NxAD<18>
AM32	I	GNT*
AM34	I	VCC4
AM36	O	XNOE*
AN1	O	CBANK<1>
AN3	I	VSS
AN5	O	CADDR<11>
AN7	I	SLOTID<3>
AN9	I	POPHOLD
AN11	O	CADDR<16>
AN13	I/O	CDATA<43>
AN15	I/O	CDATA<52>
AN17	I/O	CDATA<48>
AN19	I/O	CDATA<63>
AN21	I/O	CDATA<58>
AN23	I/O	NxAD<34>
AN25	I/O	NxAD<10>
AN27	I/O	NxAD<3>
AN29	I/O	NxAD<8>
AN31	I/O	NxAD<50>
AN33	I/O	NxAD<16>
AN35	I	VSS
AN37	O	XBOE*
AP2	O	CADDR<17>
AP4	I	VCC4
AP6	I	VCC4
AP8	I	VCC4
AP10	I	VCC4
AP12	I	VCC4
AP14	I	VCC4
AP16	I	VCC4
AP18	I	VCC4
AP20	I	VCC4
AP22	I	VCC4
AP24	I	VCC4
AP26	I	VCC4
AP28	I	VCC4
AP30	I	VCC4

JEDEC Pin	Pin Type	Signal Name
AP32	I	VCC4
AP34	I	VCC4
AP36	I/O	NxAD<4>
AR1	-	NC
AR3	I	VSS
AR5	I	VSS
AR7	I	VSS
AR9	I	VSS
AR11	I	VSS
AR13	I	VSS
AR15	I	VSS
AR17	I	VSS
AR19	I	VSS
AR21	I	VSS
AR23	I	VSS
AR25	I	VSS
AR27	I	VSS
AR29	I	VSS
AR31	I	VSS
AR33	I	VSS
AR35	I	VSS
AR37	I/O	NxAD<37>
AT2	I	HROM
AT4	O	NxADINUSE
AT6	I/O	CDATA<34>
AT8	I/O	CDATA<32>
AT10	I/O	CDATA<45>
AT12	I/O	CDATA<47>
AT14	I/O	CDATA<42>
AT16	I/O	CDATA<53>
AT18	I/O	CDATA<49>
AT20	I/O	CDATA<62>
AT22	I/O	NxAD<32>
AT24	I/O	NxAD<35>
AT26	I/O	NxAD<42>
AT28	I/O	NxAD<2>
AT30	I/O	NxAD<19>
AT32	I/O	NxAD<17>
AT34	I/O	NxAD<48>
AT36	I/O	NxAD<5>
AU1	I	ANALYZEIN
AU3	O	CWE<4>*
AU5	I/O	CDATA<35>
AU7	I/O	CDATA<33>
AU9	I/O	CDATA<44>
AU11	I/O	CDATA<46>
AU13	-	NC
AU15	I/O	CDATA<40>
AU17	I/O	CDATA<55>
AU19	-	NC
AU21	I/O	CDATA<57>
AU23	I/O	NxAD<1>
AU25	I/O	NxAD<11>
AU27	I/O	NxAD<40>
AU29	I/O	NxAD<9>
AU31	I/O	NxAD<51>
AU33	I/O	NxAD<49>
AU35	I/O	NxAD<39>
AU37	I/O	NxAD<6>

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

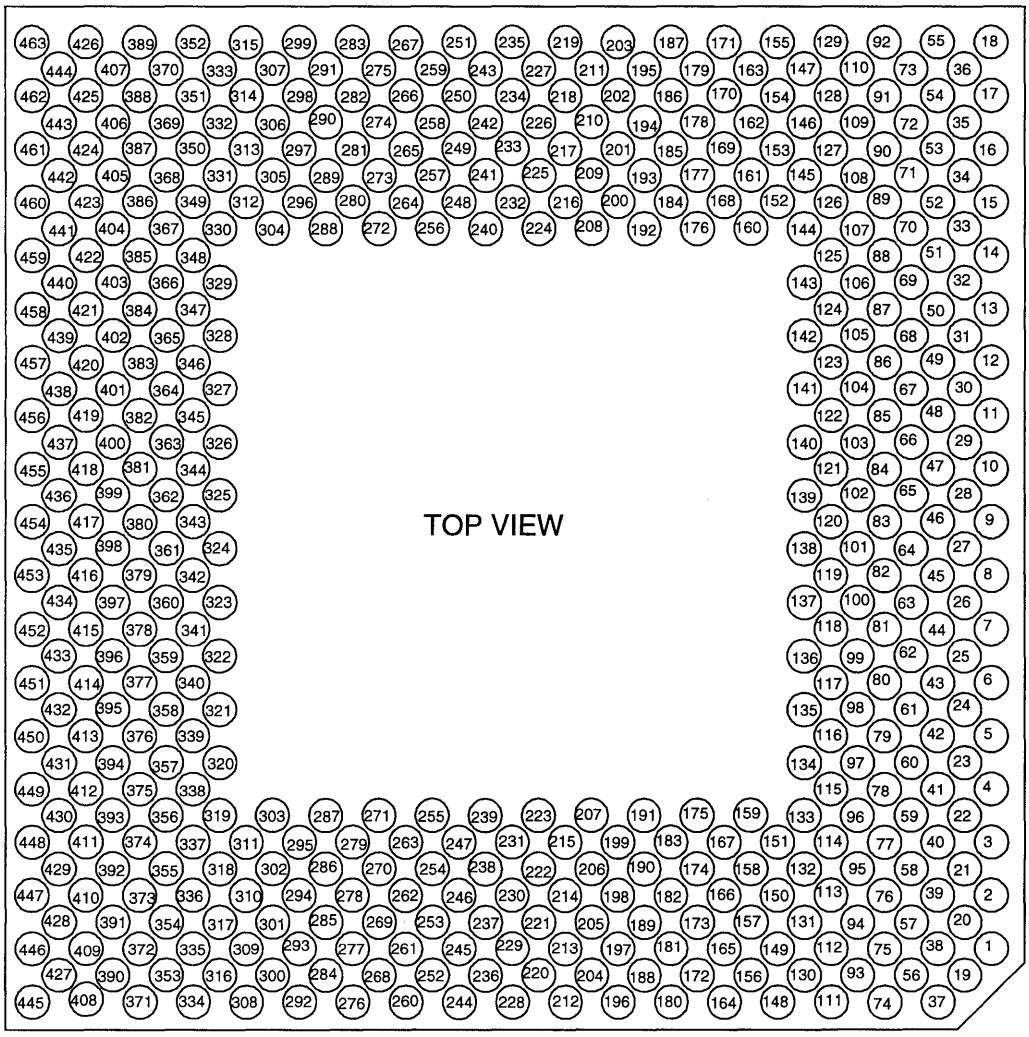


Figure 6 Nx586 PGA Pinout Diagram (Top View)

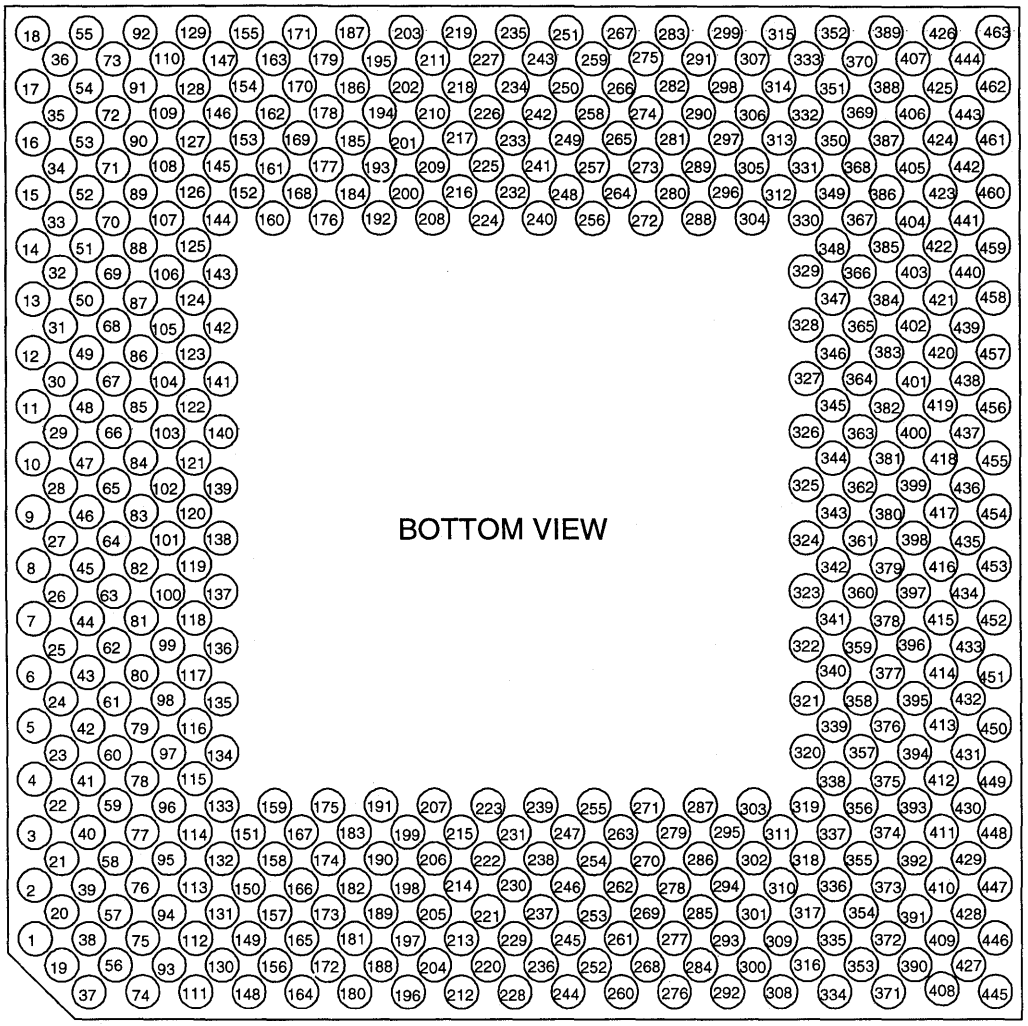


Figure 7 Nx586 PGA Pinout Diagram (Bottom View)

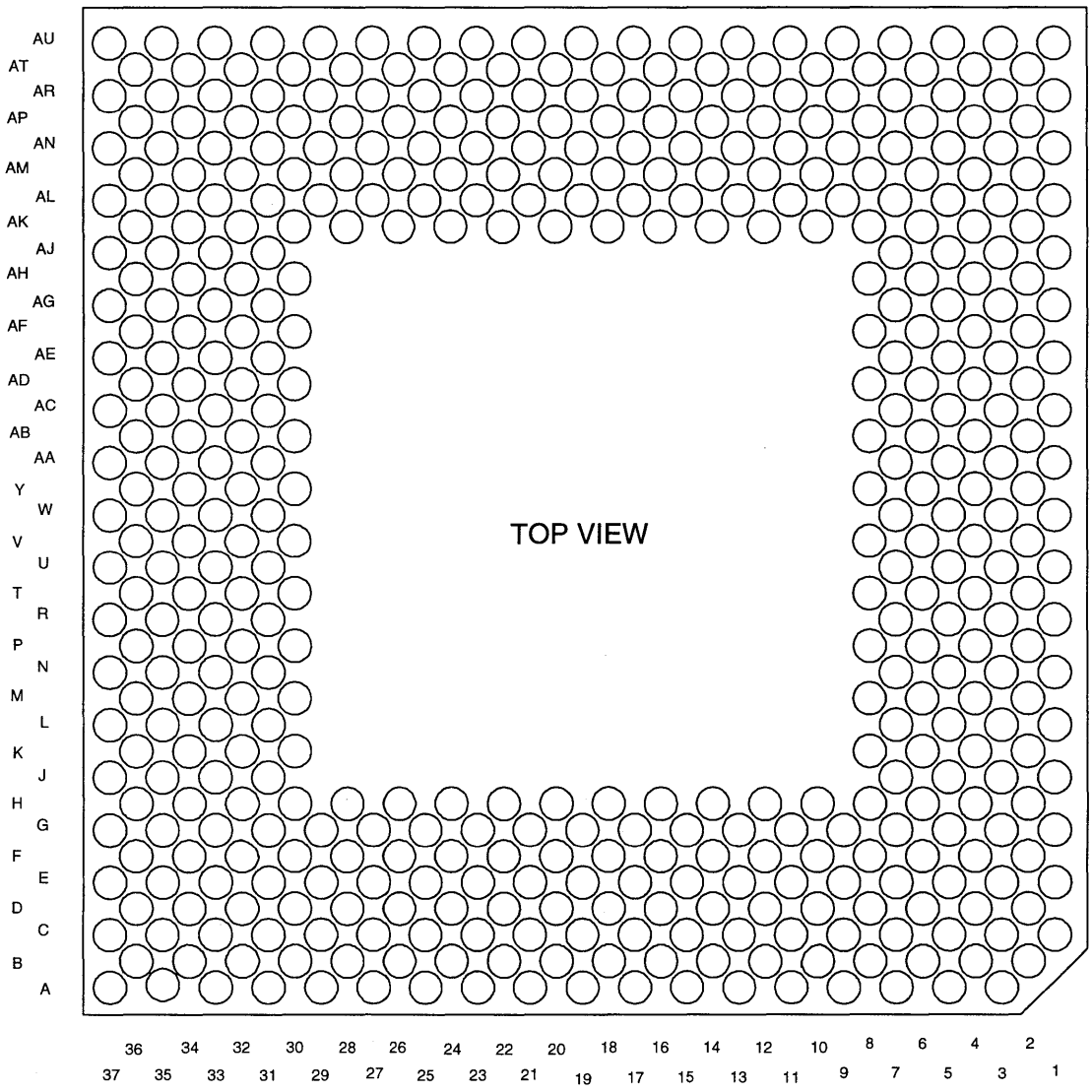


Figure 8 Nx586 JEDEC Pinout Diagram (Top View)

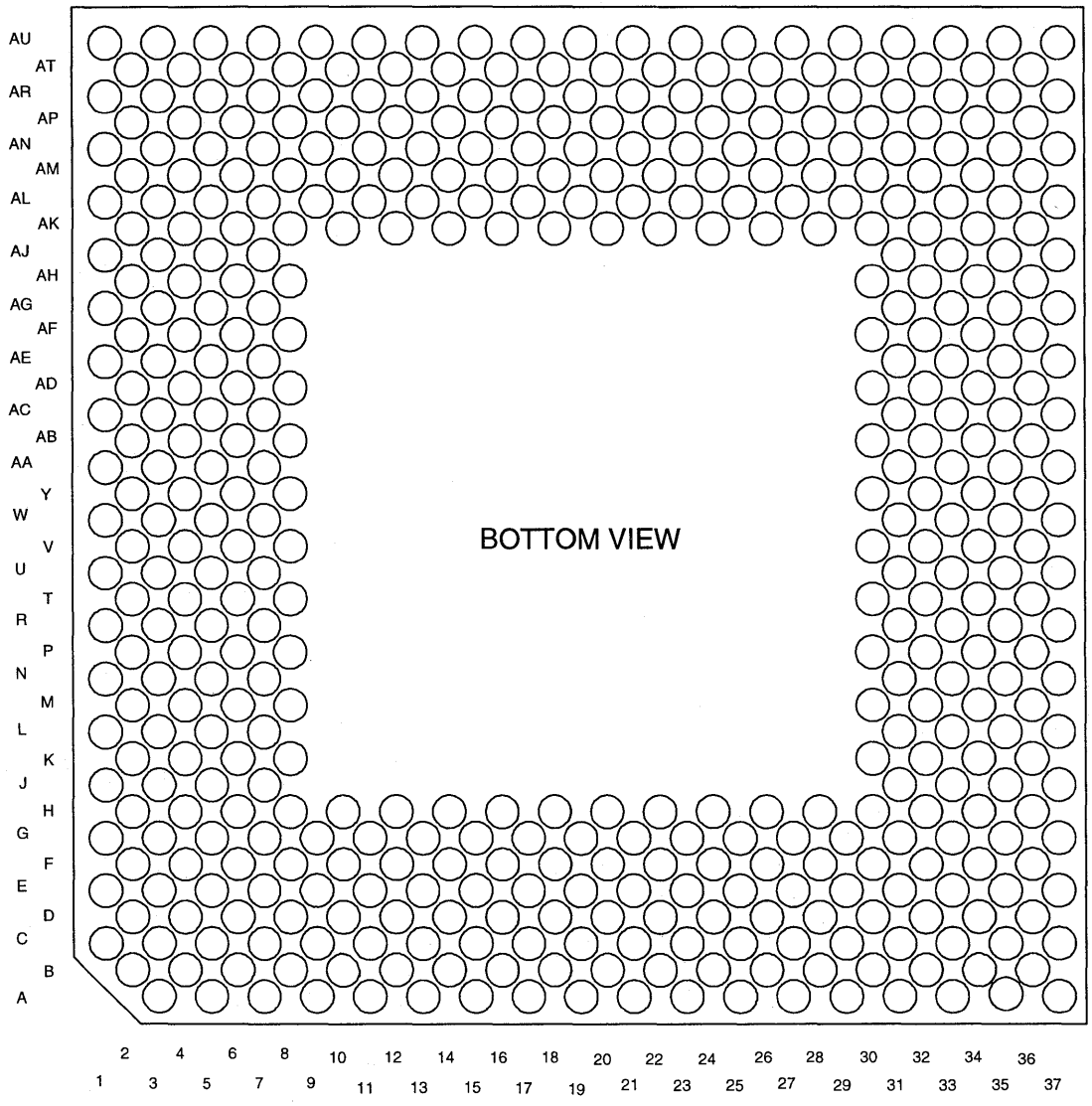


Figure 9 Nx586 JEDEC Pinout Diagram (Bottom View)

Nx586 NexBus/NexBus5 Signals

NexBus/NexBus5 Arbitration

NREQ*	O	**NexBus Request**—Asserted by the processor to the NexBus5 arbiter to secure control of the system bus. The Nx586 will drive NREQ* active on the rising edge of NxCLK. The bus is granted when the arbiter asserts GNT*. The "grant" becomes effective only when the Nx586 asserts ALE* or LOCK*. This signal remains active until one NxCLK 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. 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*. The processor at anytime may perform speculative cycles that prematurely terminate. This is done by asserting NREQ* and then subsequently removing NREQ* before GNT* is asserted.
AREQ*	O	**Alternate-Bus Request**—Asserted by the processor to the NexBus arbiter to secure control of the system bus and any other buses (called *alternate buses)* supported by the system. This signal remains active until GNT* is received from the NexBus5 Arbiter; unlike NREQ*, the processor does not make speculative requests with AREQ*. The arbiter does not issue GNT* until the other system buses are available. AREQ* is driven on the rising edge of NxCLK.
GNT*	I	**Grant NexBus**—Asserted by the NexBus5 arbiter to indicate that the processor has been granted control of the system bus. GNT* is asserted on the rising edge of NxCLK and is held active until a valid ALE*. GNT* can be active for a minimum of two NxCLKs if ALE* is driven immediately after GNT* is received.

LOCK*	O	**Bus Lock**—Asserted by the processor to the NexBus^5 arbiter when multiple bus operations should be performed sequentially and uninterruptedly. This signal is used by the NexBus^{5}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 may allow masters on system buses other than NexBus^5 (i.e., on an *alternate bus)* to intervene in a locked NexBus^5 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^5 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 systems logic that interfaces the NexBus^5 to other system buses (called the *alternate-bus interface).* This signal is asynchronous to the NexBus clock.

NexBus/NexBus^5 Cycle Control

ALE*	O	**Address Latch Enable**—Asserted by the processor to backplane logic or to the systems logic interface between the NexBus^5 and other system buses (called the *alternate-bus interface)* when the processor is driving valid addresses and status information on the NxAD<63:0> bus. ALE* is driven active on the rising edge of NxCLK after GNT* is received for one NxCLK. All ALE* signals are NANDed on the bus backplane or systems logic to generate GALE.
GALE	I	**Group Address Latch Enable**—Asserted by a backplane NAND of all ALE* signals, to indicate that the NexBus^5 address and status can be latched. GALE should be monitored by all devices on NexBus^5 to latch the address placed on the bus by the master.
GTAL	I	**Group Try Again Later**—Asserted by the systems logic interface between NexBus^5 and other system buses (called the *alternate-bus interface)* to indicate that the attempted bus-crossing operation cannot be completed, because the systems logic bus interface is busy or cannot access the other system buses. 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 systems logic bus interface asserting GTAL and without the processor asserting AREQ*, if the other system bus and its systems 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 systems logic interface is unavailable. The protocol prevents deadlocks and prevents the processor from staying on NexBus^5 until the other system bus becomes available. Unlike other group signals, which are the NAND 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^5 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 NexBus5 device is prepared to respond as a slave to the processor's current operation. The systems logic interface between the NexBus5 and other system buses (called the *alternate-bus interface)* monitors the XACK* responses from all adapters. In general, since the systems 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 systems logic interface between NexBus5 and other system buses (called the *alternate-bus interface)* in response to another NexBus5 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 active XHLD* indicates that the CPU is not ready to complete the transfer (This situtation may occur for speculative cycles). Slaves supply read data in the clock following the first clock during which GXACK is asserted and GXHLD (via XHLD* negated) is negated.
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^5 master, this signal is asserted by the processor to indicate that the location being accessed is contained in 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^5 and write back its modified data to main memory. While the data is being written to memory, the requesting master reads it off NexBus^5. The assertion of DCL* is the only way in which atomic 32-byte cache-block fills by another NexBus^5 master can be preempted by the processor for the purpose of writing back dirty data. During writes by another NexBus^5 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 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^5 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 Enable**—Asserted by a memory slave to enable block 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.

OWNABL	I	**Ownable**—Asserted by the systems 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 NexBus5 addresses are assumed to be cacheable in the *exclusive* state. The OWNABL signal is provided in case the systems logic needs to restrict caching to certain locations. In single-processor systems, the OWNABL signal is typically tied high for write-back configurations to allow caching in the *exclusive* state on all reads.
SHARE*	O	**Shared Data**—The purpose of SHARE* is to let NexBus5 caching devices (including caching devices on an alternate bus) indicate that the current read operation hit in a cache block that is present in another device's cache. It is asserted by the Nx586 during block reads by another NexBus5 master to indicate to the other master that its read hit is 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 NexBus5 device has declared via SHARE* that it has cached a copy. Instruction fetches are always *shared.*

NexBus Transceivers

XBCKE*	O	**NxAD Transceiver Bus Clock Enable**—Asserted by the processor to clock registered transceivers and latch addresses/status and data from the AD<63:0> bus for subsequent driving onto the NxAD<63:0> bus. There is no comparable clock-enable for the NexBus^5 side of these transceivers; they are always enabled on the NexBus^5 side. Note, NxCLK is normally connected to the clocking pin for the AD<63:0> registers and an inverted NxCLK is connected to the clocking pin for the NxAD<63:0> registers.
XBOE*	O	**Transceiver to AD Bus Output Enable**—Asserted by the processor to enable the registered transceivers and drive addresses and data onto the AD<63:0> bus from the NxAD<63:0> bus. Note, NxCLK is normally connected to the clocking pin for the AD<63:0> registers and an inverted NxCLK is connected to the clocking pin for the NxAD<63:0> registers.
XNOE*	O	**Transceiver to NxAD Bus Output Enable**—Asserted by the processor to enable registered transceivers and drive addresses and data onto the NxAD<63:0> bus from the AD<63:0> bus. Note, NxCLK is normally connected to the clocking pin for the AD<63:0> registers and an inverted NxCLK is connected to the clocking pin for the NxAD<63:0> registers.
XCVERE*	I	**NexBus^5 Transceiver Enable**—XCVERE* determines what type of bus is generated by the processor. When pulled high, the Nx586 will generate the NexBus processor bus which requires external transceivers to connect to the processor to the NexBus^5 system bus. If XCVERE* is tied low, the Nx586 generates NexBus^5 directly. This pin is sampled by the processor during reset active.

NexBus/NexBus5 Address and Data

NxAD<63:0> **AD<63:0>**	I/O	**NexBus or NexBus5 Address and Status, or Data**—This bus multiplexes address and status information during the "address and status phase" and with up to 64 bits of data during a subsequent "data phase". XCVERE* determines the local bus mode. The Nx586 generates NexBus (AD) for XCVERE* asserted and NexBus5 for XCVERE`* negated. The NexBus address and status is valid on the rising edge of XBCKE*. For either bus modes, the address and status is valid on NexBus5 when GALE is asserted. At that time, address NxAD<63:32> and status NxAD<31:0> is latched. 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.

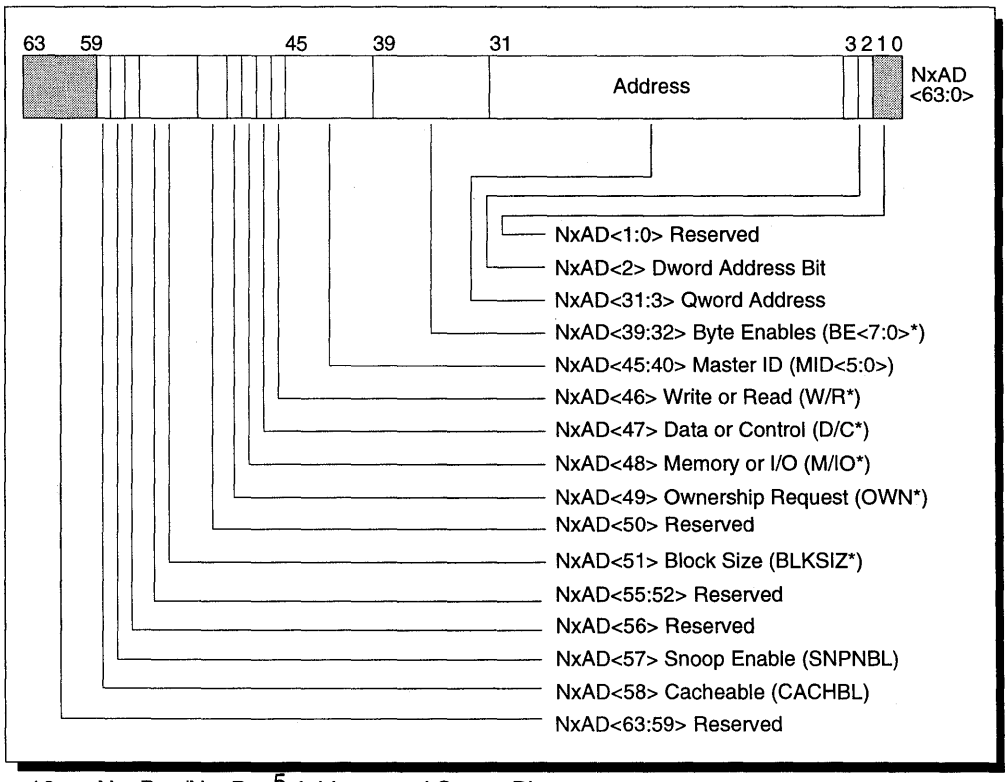


Figure 10 NexBus/NexBus5 Address and Status Phase

NxAD<1:0> **AD<1:0>** *address phase*	I/O	**Reserved**—These bits must be driven high by the bus master.
NxAD<2> **AD<2>** *address phase*	I/O	**ADDRESS<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> **AD<31:3>** *address phase*	I/O	**ADDRESS<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> **AD<39:2>** *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 11 and 12. 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 11 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 12 Byte-Enable Usage during Memory Transfers

NxAD<45:40> AD<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. MID 000000 is reserved for the systems logic.

NxAD<46> AD<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 13.
NxAD<47> AD<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 13.
NxAD<48> AD<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 13.

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 13 Bus-Cycle Types

NxAD<49> AD<49> *address phase*	I/O	**Ownership Request**—Asserted by a master when it intends to cache data in the *exclusive* state. This 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> AD<50> *address phase*	I/O	**Reserved**—This bit must be driven high.

NxAD<51> AD<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> AD<56:52> *address phase*	I/O	**Reserved**—These bits must be driven high.
NxAD<57> AD<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> AD<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 invalidation or write-backs.
NxAD<63:59> AD<63:59> *address phase*	I/O	**Reserved**—These bits must be driven high by the bus master.

Nx586 L2 Cache Signals

SRAMMODE	I	**L2 Cache SRAM mode Select**—Selects the use of either synchronous or asynchronous SRAM for the L2 cache memory. This pin is sampled during reset active. When SRAMMODE is left unconnected (floating), the internal pull down resistor configures the Nx586 for asynchronous SRAMs. If SRAMMODE is pulled high, the Nx586 is configured for synchronous SRAMs. In synchronous mode, the CKMODE pin generates the SRAM clocks and COEB* generates global SRAM write enables after RESET. Also, CKMODE can be disabled by driving SCLKE low or inactive. SRAMMODE contains an internal pull down resistor.
COEA*	O	**L2 Cache Output Enable A**—Enables reading from second-level cache SRAMs to drive the CDATA<63:0> bus. COEA* should be connected to a maximum of four devices.
COEB*/WE*	O	**L2 Cache Output Enable B**—Enables reading from second-level cache SRAMs to drive the CDATA<63:0> bus. COEB* should be connected to a maximum of four devices. COEB* has the identical function as COEA* when SRAMMODE is low. **Global Write Enable**—When SRAMMODE is pulled high, COEB* is reconfigured as a global write enable for synchronous SRAMs.
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 SRAMs.
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.

Nx586 System Signals

Nx586 Clocks

NxCLK	I	**NexBus Clock**—A TTL-level clock. All signals on NexBus/NexBus5 transition on the rising edge of NxCLK, except the asynchronous signals, INTR*, NMI*, GATEA20, and SLOTID<3:0>. If the Nx586 is configured for internal PLL mode, the processor's internal phase-locked loop (PLL) synchronizes the internal processor clocks at twice the frequency of NxCLK. For external PLL mode, NxCLK is used to generate the skew correcting reference clock for the external PLL circuitry. The skewed version of NxCLK is produced at XREF.
PHE1	I	**Clock Phase 1**—PHE1 is used as the processor clocking source when the Nx586 is configured for external PLL mode. The deskewed clock (normally twice the frequency of XREF) generated by the external PLL circuitry is connected to PHE1. For normal clocking operation, this signal should be pulled low.
PHE2	I	**Clock Phase 2**—PHE2 determines the relationship between the internal non-overlapped clocks. When pulled low, narrow non-overlapped clocks are generated. Wide non-overlapped clocks are produced for PHE2 pulled high. For normal clocking operation, this signal should be pulled low.
SCLKE	I	**Synchronous Clock Enable**—While in synchronous SRAM mode (see SRAMMODE), SCLKE is used to determine the output of CKMODE. If SCLKE is asserted, CKMODE generates a clock equal to the processor's internal frequency (twice NxCLK). While inactive, CKMODE is driven low. For normal clocking operation, this signal should be pulled low.
CKMODE	I	**Clock Mode**—For normal clocking operation, this signal should be pulled low. When SRAMMODE is pulled high, the Nx586 is configured for synchronous SRAMs. In synchronous SRAM mode, the CKMODE pin generates the SRAM clocks and COEB* generates global L2 SRAM write enables after RESET is inactive.

XSEL	I	**Clock Mode Select**—XSEL is used to select which PLL mode is utilized by the processor, internal or external. Internal PPL mode is selected when XSEL is tied low. For XSEL pulled high, the external PLL mode is selected. For normal clocking operation, this signal should be tied low.
XPH1	O	**Processor Clock Phase 1**—For normal clocking operation, this signal must be left unconnected.
XPH2	O	**Processor Clock Phase 2**—For normal clocking operation, this signal must be left unconnected.
IREF	I	**Clock Current Reference**—This signal must be pulled up to V_{DDA}. Refer to NexGen for the optimal value.
XREF	O	**Clock Output Reference**—For normal clocking operation, this signal must be terminated with a value that matches the characteristic impedance of the circuit board (PCB). A Thevenin type of termination is recommended. In external PLL mode, XREF is the skewed version of NxCLK and is normally connected to the input of the external PLL circuitry's phase comparator.
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

NPIRQ*	O	**Floating Point Unit Interrupt Request**—Asserted by the Floating-Point unit to the interrupt controller's IRQ13 that services floating-point errors in an PC-AT.
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 NxCLK.
NMI*	I	**Non-Maskable Interrupt**—Edge sensitive. Asserted by systems 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 NxCLK.
RESET*	I	**Global Reset (Power-Up Reset)**—Asserted by systems logic. The processor responds by resetting its internal state machines and loading default values into its registers and reading the hardware configuration pins (i.e. SRAMMODE, CKMODE, XSEL, etc.). At power-up it must remain asserted for a minimum of 1 millisecond after VCC and NxCLK have reached their proper AC and DC specifications.
RESETCPU*	I	**Reset CPU (Soft Reset)**—Asserted by the systems logic 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.
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 PC-AT 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 NxCLK.

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.
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 tri-states 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 individually pulled high to VCC4 for normal operation.
PULLLOW	I/O	**Reserved**—These signals must be individually 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 or L2 Synchronous Clock output
COEA*	O	L2 Cache Output Enable A
COEB*(WE*)	O	L2 Cache Output Enable B or Synchronous SRAM global write
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^5
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
NPIRQ*	O	*Reserved*
NREQ*	O	NexBus^5 Request
NxAD<63:0>	I/O	Bus Address/Status, or Bus Data
NxADINUSE	O	*Reserved*
NxCLK	I	NexBus^5 Clock

OWNABL	I	Ownable
P4REF	O	Power Reference
PHE1	I	Clock Phase 1
PHE2	I	Clock Phase 2
POPHOLD	I	*Reserved*
PTEST	I	*Reserved*
PULLHIGH	I/O	*Reserved*
PULLLOW	I	*Reserved*
RESET*	I	Global Reset (Power-Up Reset)
RESETCPU*	I	Reset CPU (Soft Reset)
SCLKE	I	Synchronous SRAM Clock Enable (CKMODE)
SERIALIN	O	Serial In
SERIALOUT	O	Serial Out
SHARE*	O	Shared Data
SLOTID<3:0>	I	NexBus Slot ID
SRAMMODE	I	L2 Cache SRAM Type Select
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
XCVERE*	I	Internal NexBus Transceiver Enable
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

Hardware Architecture

The Nx586 processor and the optional integrated floating-point execution unit 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. Both the Nx586 and Nx586 with the floating point have the identical system bus architecture. Therefore, the two devices are interchangeable within the processor socket.

Bus Structure

The Nx586 processor supports two external 64-bit buses: the processor bus, the L2 cache bus, and one internal 64-bit bus (for the integrated floating-point unit). All buses are synchronous to the NxCLK clock. The internal floating-point unit bus operates at the same speed as the processor core or twice the frequency of the local bus.

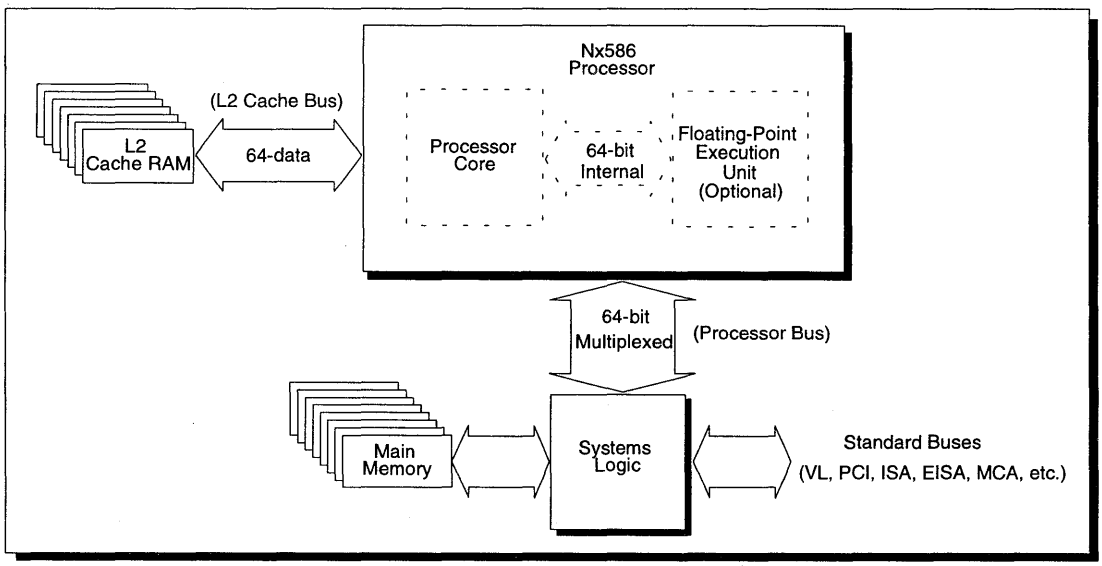


Figure 14 Nx586 Bus Structure Diagram

Processor Bus

The Nx586 supports two local bus interfaces, NexBus and NexBus^5. NexBus is considered a true CPU local bus. Where as, NexBus^5 is a NexGen proprietary system bus. During RESET* active, the XCVERE* pin is sampled for the local bus mode. XCVERE* determines what type of bus is generated by the processor. When pulled high, the Nx586 will generate the NexBus standard which requires external transceivers to connect the processor to the NexBus^5 system bus. Figure 15 is a system block diagram showing the Nx586 configured with a NexBus interface (XCVERE* = 1). The NexBus transceivers are high speed non-inverting registered transceivers controlled by signals provided by the Nx586.

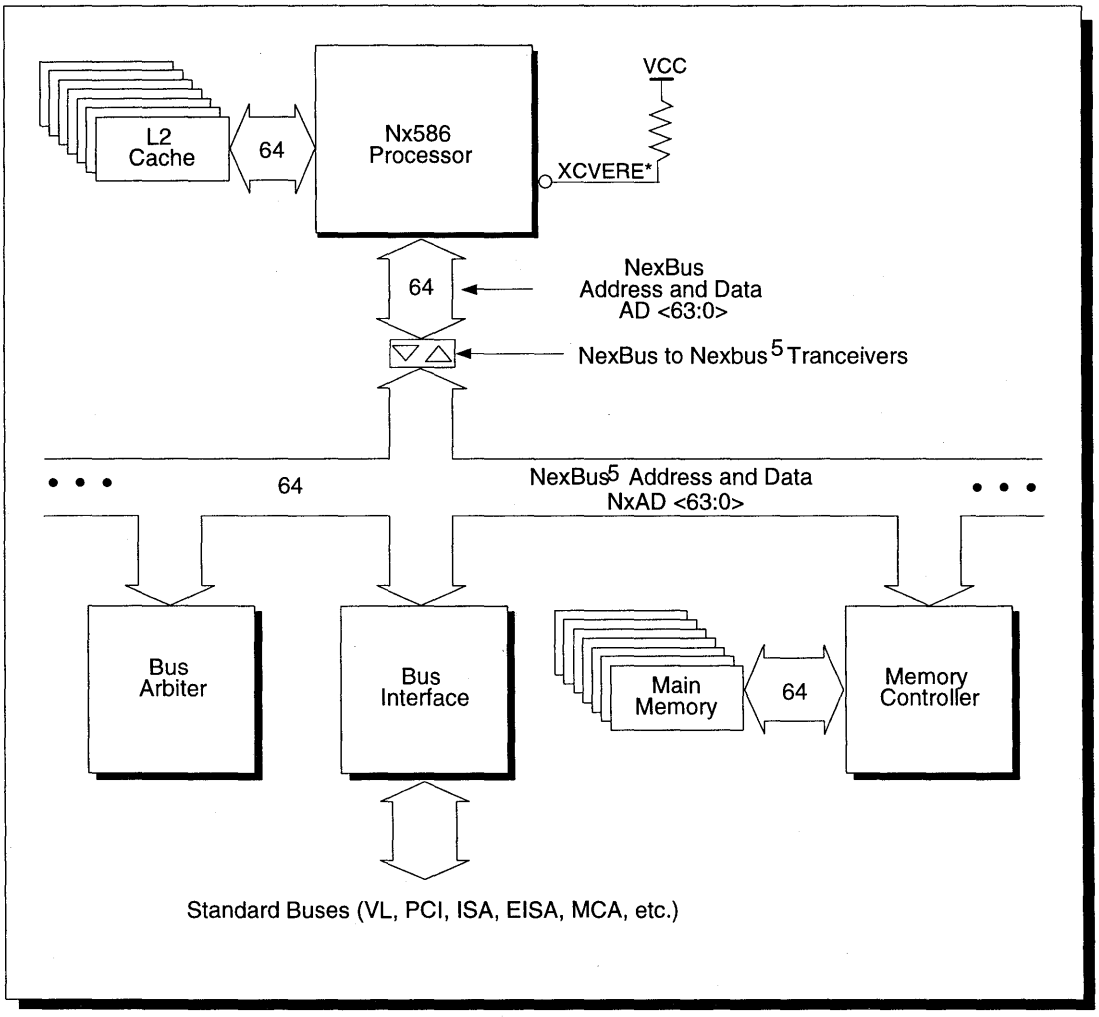


Figure 15 Nx586 Basic System Diagram

Another approach for processors configured with external NexBus transceivers is to include the transceiver within the systems logic. This however forces the systems logic to provide complete arbitration and signal routing to the processor via NexBus. As shown in figure 16, the example PC-AT compatible systems controller contains the system arbiter, the memory controller, the VL-Bus controller, and the ISA-Bus controller. The example systems logic is completely responsible for bus interconnections between NexBus, the memory bus, VL-Bus and ISA bus. The Integrated Peripheral Controller contains the interrupt controller, DMA controller, CMOS memory, timers and counters.

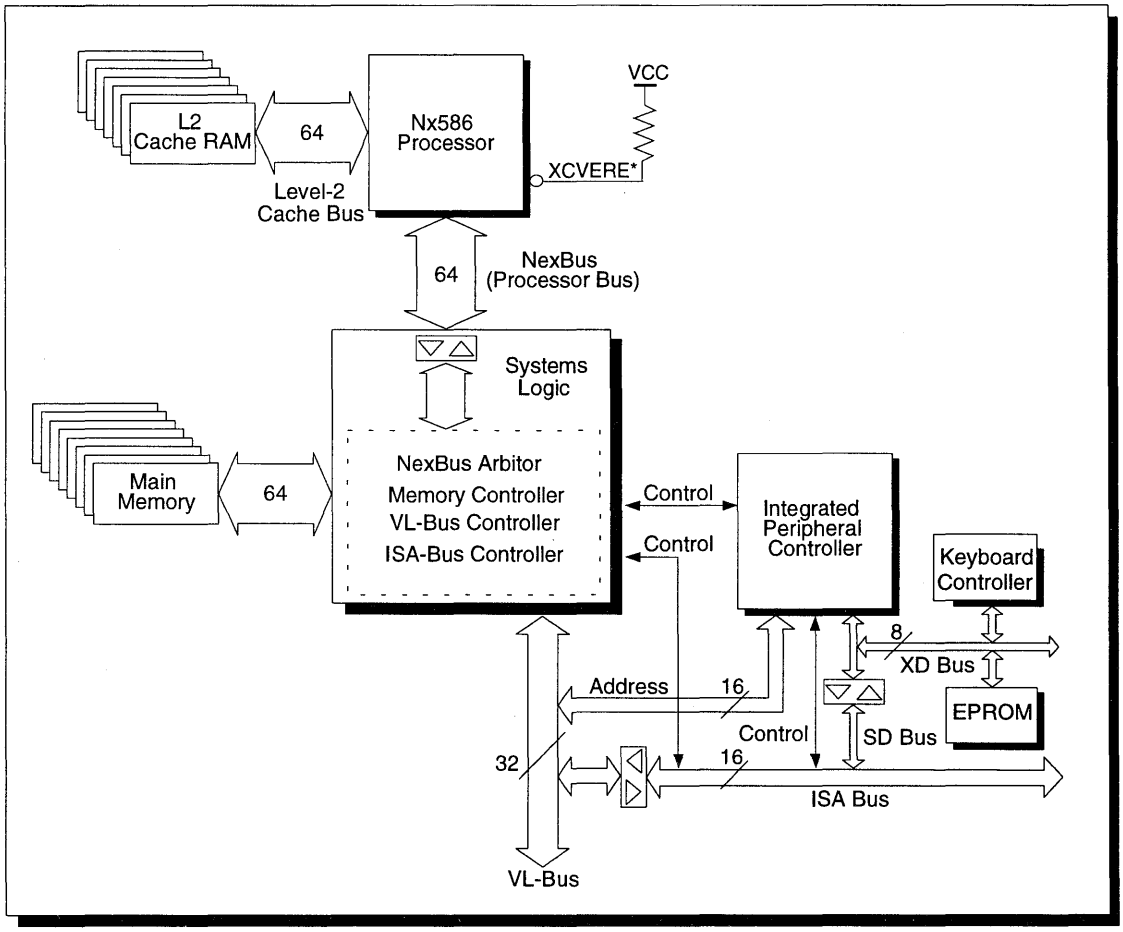


Figure 16 Nx586 System with Systems Logic containing NexBus Transceivers.

When XCVERE* is tied low, the Nx586 generates NexBus^5 directly. NexBus^5 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^5 to guarantee cache coherency.

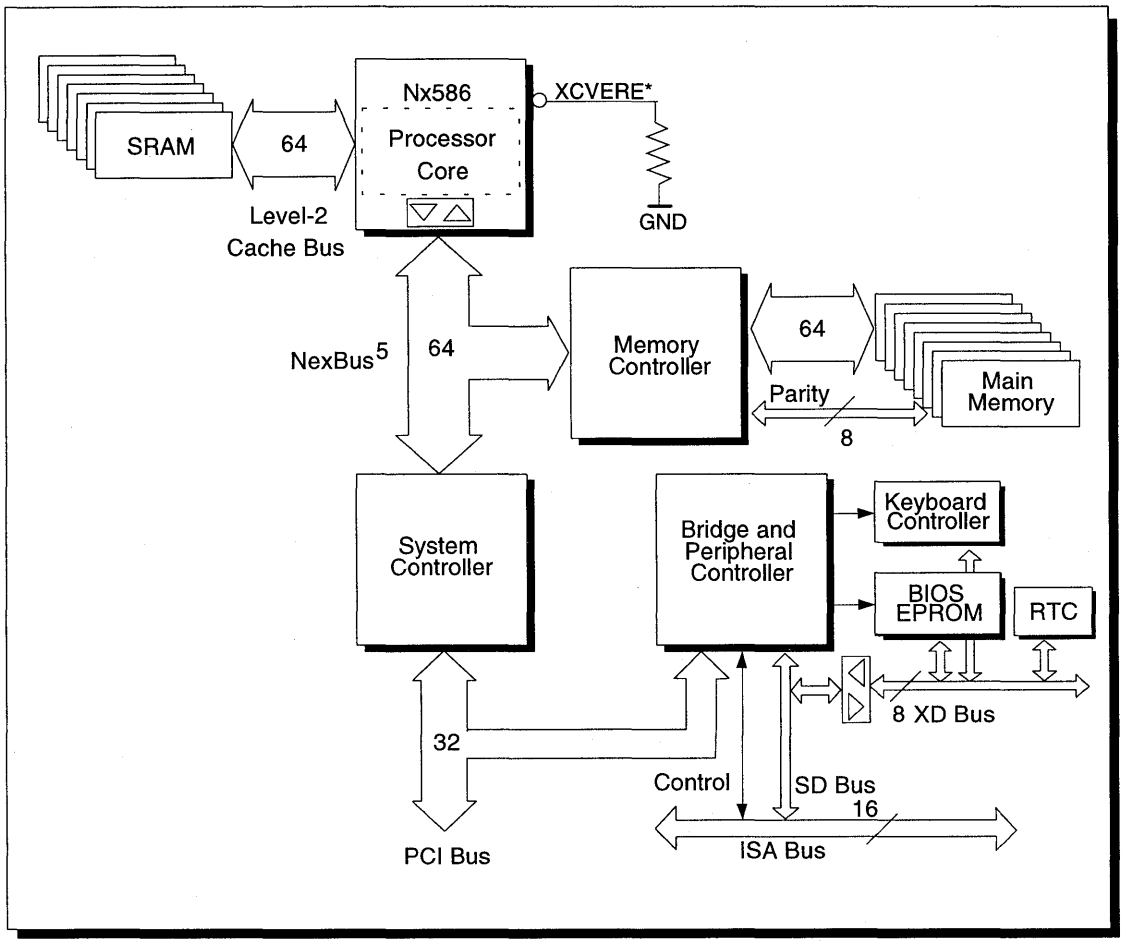


Figure 17 Example System with the Nx586 and NexBus^5.

The Nx586 based PCI system shown in figure 17 uses a chipset to connect the Nx586 via the NexBus^5 system bus. The example chipset is divided into two major components, the memory and systems controller. The system controller contains the NexBus^5 arbiter, and the PCI bus controller. Note, both the memory controller and the system controller are NexBus^5 devices and can respond directly to the processor. The ISA bus is generated by a PCI to ISA bridge chip.

L2 Cache Bus

The 64-bit L2 cache bus is dedicated to external 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 write-policy can be programmed for write-back or write-through. Optionally, at power-on reset the L2 cache controller can be programmed for synchronous or asynchronous SRAMs. Bus accesses for each mode are identical except for the existence of the SRAM clocking signal on CKMODE for the synchronous SRAMs. Note, the synchronous SRAMs operate at the same frequency as the processor not at half the frequency.

The processor manages cache-coherency for both L2 and L1 caches. The 64-bit L2 cache bus is fully isolated and decoupled from the processor local bus also known as NexBus. The L2 cache bus does not require any arbitration to gain control of the bus. In fact, the L2 cache controller can start a L2 cache cycle on any processor clock. In addition, speculative cycles are supported on the L2 cache bus. The processor can request data from the L2 cache controller and terminate the cycle at any time during the access. 32-bytes is the unit of transfer between the memory and the cache. There is no data bursting from the L2 cache memory. Since no arbitration is necessary, the L1 cache line fills are just back-to-back read cycles.

Internal 64-bit Execution Unit Bus

The Nx586 contains an internal 64-bit bus dedicated to the optional floating-point execution unit. Discrete 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 execution unit a tightly coupled component of the execution pipeline.

Both the processor core and the Floating-Point execution unit sometimes make speculative requests for the local bus (NexBus). For example, the processor requests the bus while it concurrently looks in its cache for the data to be transferred. The Floating-Point execution unit 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 18:

- *NexBus/NexBus5*—Operates at the frequency of the system clock (NxCLK).
- *Processor*—Operates at twice the frequency of the NxCLK. The Nx586 processor and the Floating Point Execution Unit both operate at the same frequency.
- *L1 (On-Chip) Cache*—Operates at twice the frequency of the processor clock.
- *L2 (Off-Chip) Cache*—Operates at the same frequency as the NxCLK. Transfers between L2-cache and the processor occur at the peak rate of one octet 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, most of the timing diagrams in the *Bus Operations* chapter are relative to the NxCLK clock, not the processor clock.

Figure 18 shows the relative clocking frequencies for a Nx586 processor. The NxCLK clock determines the systems overall operating speed. The NxCLK clock sets the NexBus/NexBus5 operating frequency. The processor's on-board PLL doubles the frequency of NxCLK making the Nx586 operate at twice the frequency of NexBus. The dual port nature of the L1 caches requires the L1 cache controllers to operate at double the frequency of the processor. The effective operating frequency of the L2-cache is half of the processor.

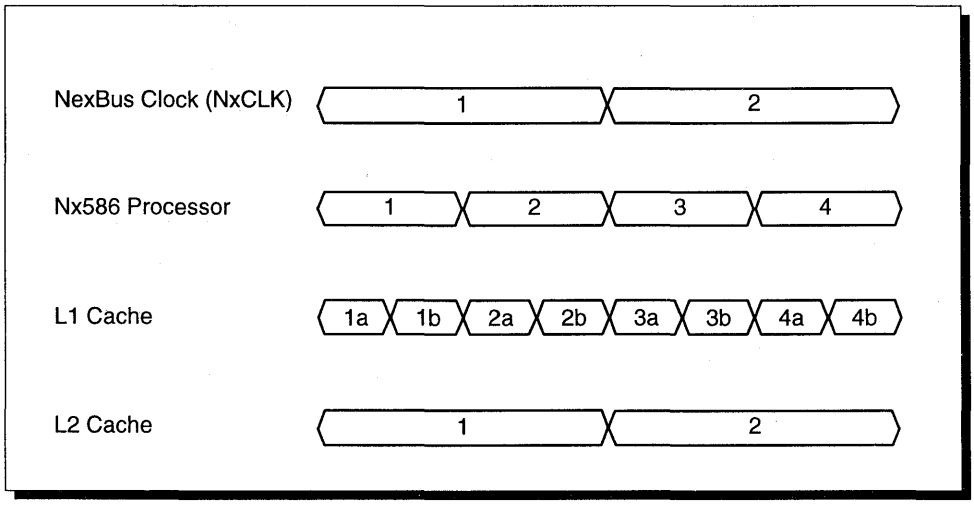


Figure 18 System Clocking Relationships

The processor uses an on-chip phase-locked loop and NxCLK to internally generates a two phase non-overlapping clock, shown in Figure 18 as the 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.

Internal Architecture

Figure 19 shows the relationship between functional units within the Nx586 processor. The main processing pipeline is distributed across five units:

- Decode Unit
- Address Unit
- Cache and Memory Unit
- 2 Integer Units
- Floating Point Execution Unit (optional)

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 (NexBus or NexBus^5) that is visible outside the processor.

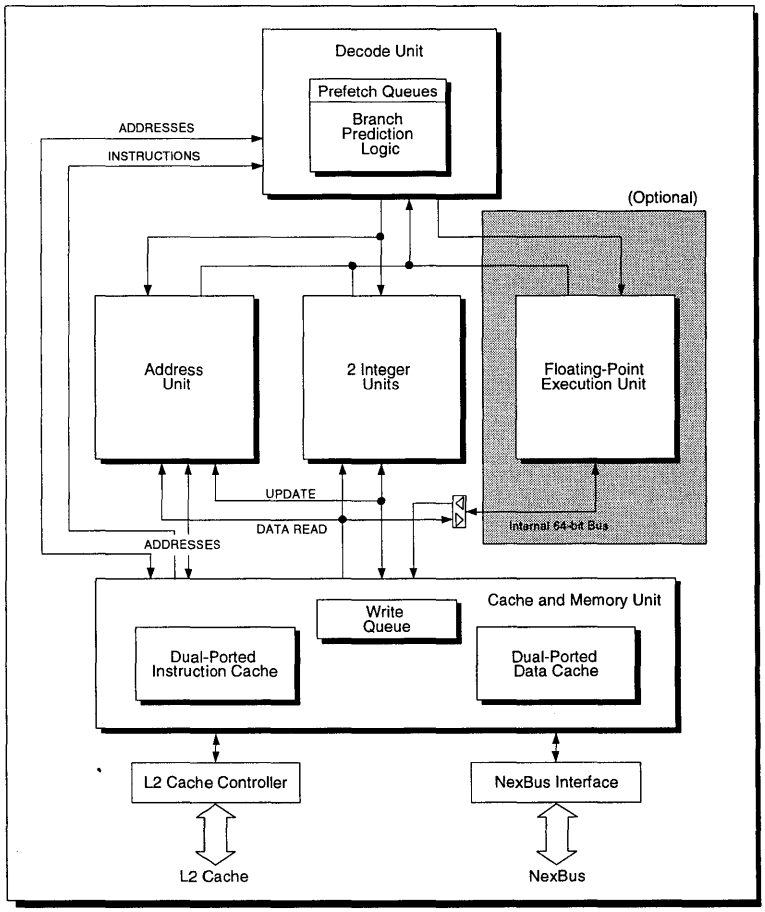


Figure 19 Nx586 Internal Architecture

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) branch prediction capability 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, a system using a memory controller chip that maintains a prefetch queue between the L2 cache and main memory can continuously pre-load 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 20 shows this hierarchy during a read cycle in a system. Figure 21 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^5 is multiplexed between address/status and data, the L2-cache data bus carries only data at 64 bits every NexBus^5 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.

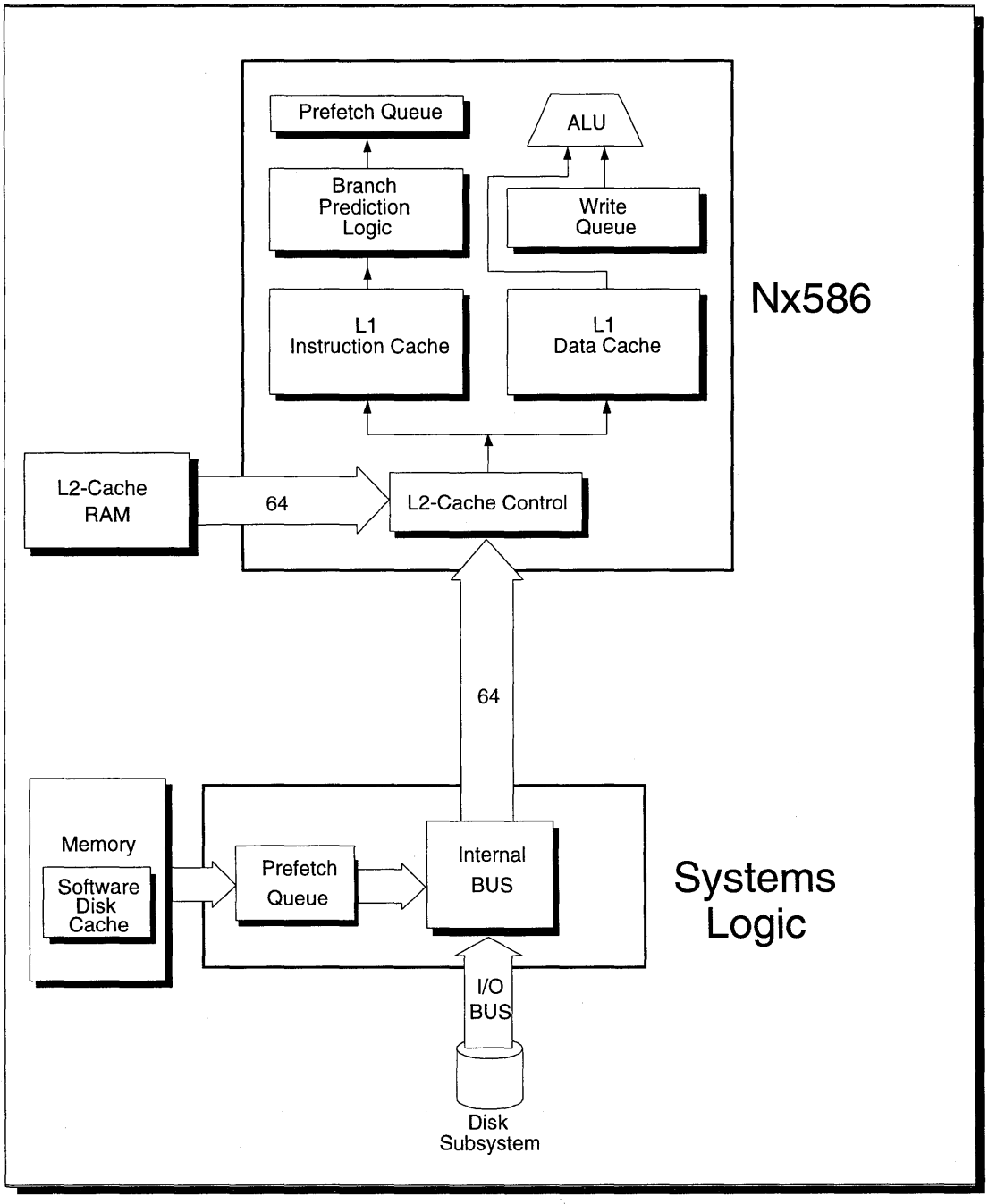


Figure 20 Storage Hierarchy (Reads)

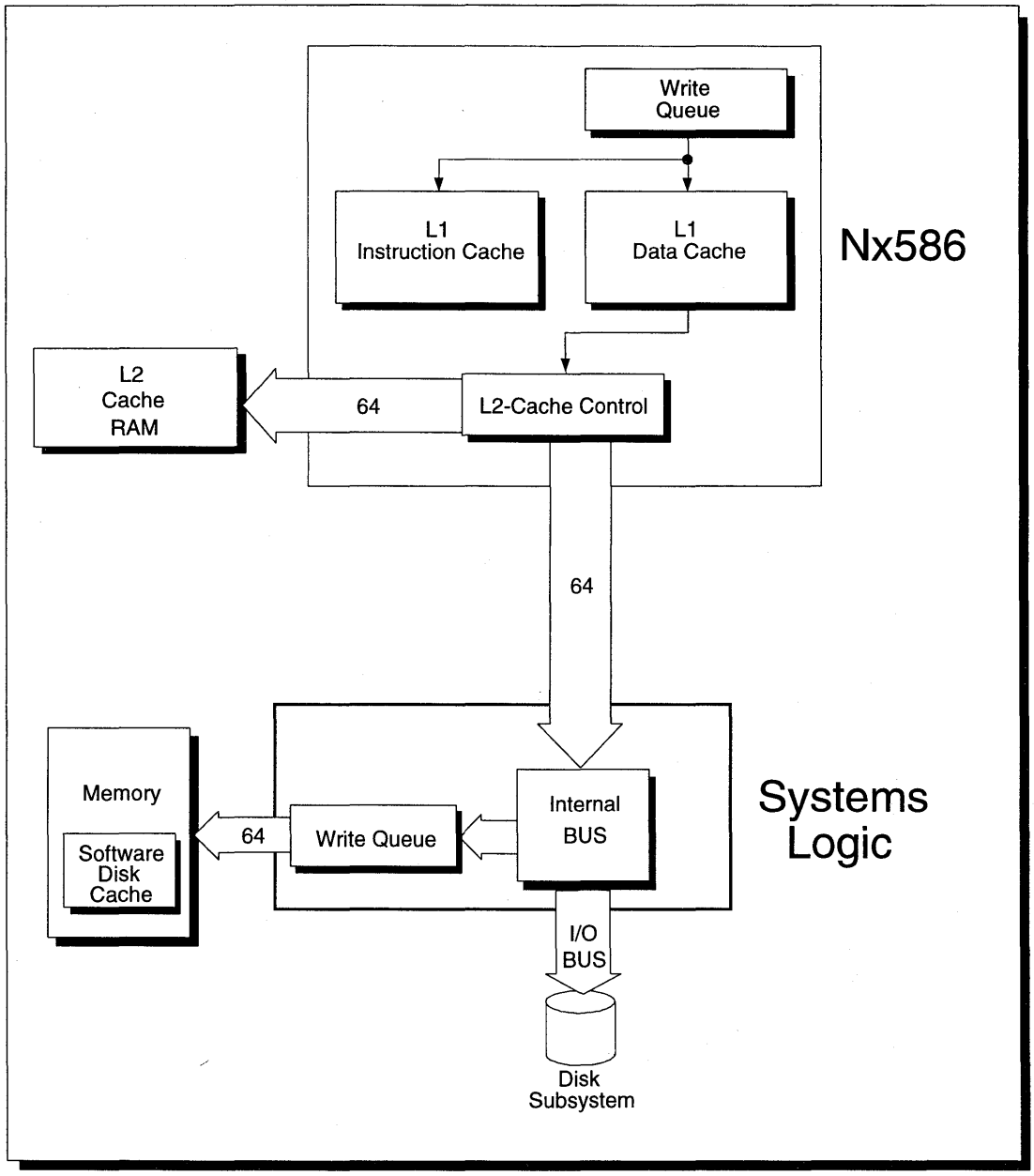


Figure 21 Storage Hierarchy (Writes)

Transaction Ordering

Interlocks enforce transaction ordering in a manner that optimizes read accesses. Interlocks are conditions where the execution of one function is deferred until the conflicting function has completed execution. With the exceptions detailed below, the *general rules* for transaction ordering are:

- *Memory Reads*—Memory reads (whether cache hits or reads on 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 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 can be built from an array of eight asynchronous (using 8-bit devices) or four synchronous (using 16/18-bit devices), or two synchronous (using 32/36-bit devices) 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	on-chip controller; 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 22 Cache Characteristics

If a write needs to go to NexBus for cache-coherency purposes, it does so before it goes to a cache. Whether the write is needed on 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[5] 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 go only to 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.

Level-2 Cache Power-on RESET Configurations

The Nx586 supports two types of Level-2 cache SRAMs, asynchronous and synchronous. Asynchronous SRAMs should be implemented for low speed cost effective system designs. On the other hand, synchronous SRAMs need to be used for systems operating the Nx586 at very high speeds (typically, 100MHz or higher). A group of pins are examined at power-on RESET to determine what mode the L2 cache controller is operating. The key pin is SRAMMODE. If SRAMMODE is pulled high, the on-chip L2 cache controller is configured for synchronous SRAMs. In synchronous SRAM mode, the CKMODE pin generates the SRAM clocks and COEB* generates global L2 SRAM write enables after RESET is inactive. While in synchronous SRAM mode, SCLKE is used to determine the output of CKMODE. If SCLKE is asserted, CKMODE generates a clock equal to the processor's internal frequency (double NxCLK). Due to loading and the loss of COEB* in synchronous mode, the type of synchronous SRAMs necessary are wide I/O or 32/36-bit. The basic access style for the synchronous SRAMs is "Flow Through". While SCLKE is inactive, CKMODE is driven low. When SRAMMODE is left unconnected (floating), the internal pull down resistor configures the Nx586 for asynchronous SRAMs. Figure 23 is a shows how to configure the Nx586 for the particular L2 Cache SRAM mode desired.

SRAM Mode Type	SRAMMODE	RESET*	SCLKE	CKMODE
—	0	1	X	PLL Mode Select
Asynchronous	0	0	X	floating
Synchronous	1	0	0	0 (low)
Synchronous	1	0	1	2x NxCLK

Figure 23 L2 Cache SRAM Interface Modes

Cache Coherency

The processor monitors (snoops) NexBus^5 operations by 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,* 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^5 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^5 cache-control bits.
- *Caches data* in a specific state by watching NexBus^5 cache-control input signals from system logic and the slave being accessed.
- *Snoops* NexBus^5 to detect other NexBus^5 transactions that hit in the processor's caches.
- *Intervenes* in the operations of other NexBus^5 devices 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 an 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 cache write.

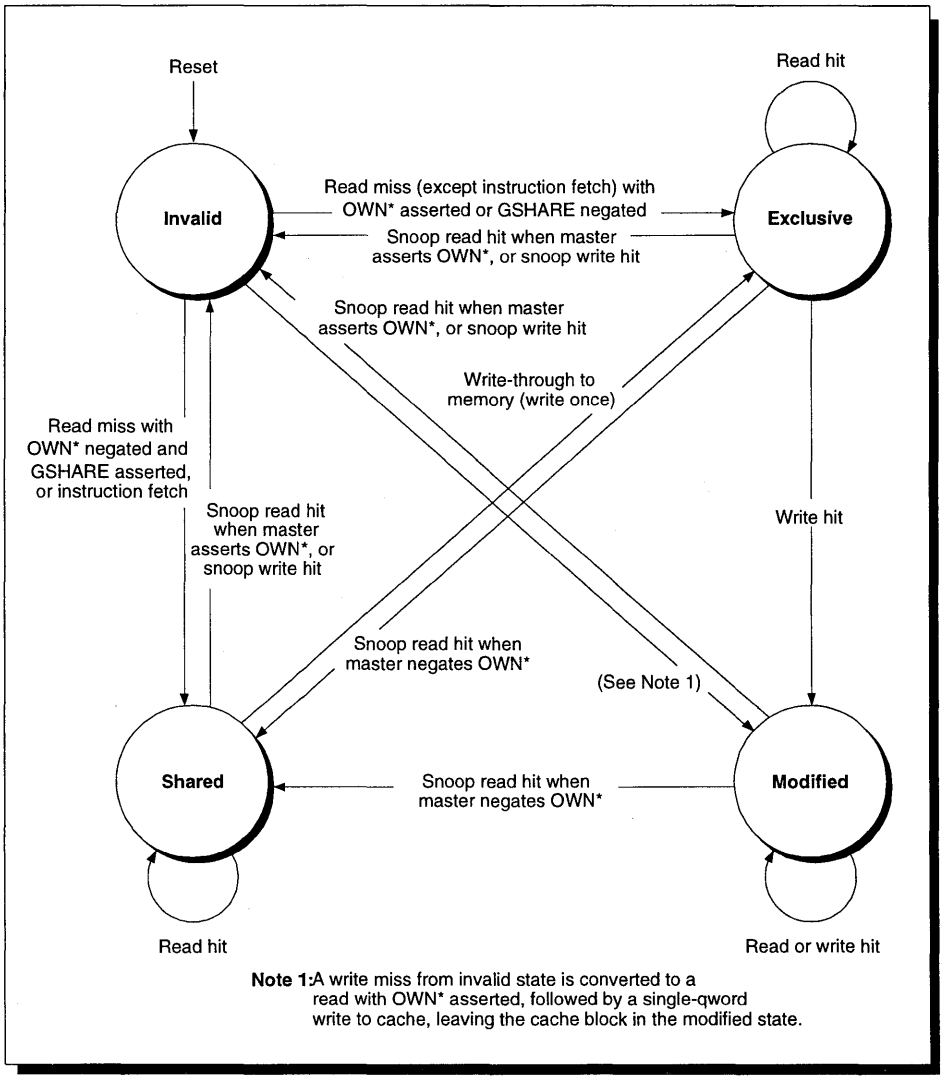


Figure 24 Basic Cache-State Transitions

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^5 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 logic that does not have an OWNABL signal, 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^5 master during block reads by another NexBus^5 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^5 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 NexBus5. The assertion of DCL* is the only way in which atomic 32-byte cache-block fills by another NexBus5 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 NexBus5 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* or invalidates the block during the operation. For external block writes (unlike the single-qword writes described above), the processor does not perform an intervenor operation with a 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 NxCLK 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 logic using the 82C206 integrated peripheral controller (IPC) or equivalent, use the IPC to handle interrupts. The systems logic typically 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 a INTA* output.

For Nx586s with the optional floating-point execution unit, the Nx586 generates unmasked floating point error interrupts on the NPIRQ* pin. The NPIRQ* function is included for PC-AT compatibility. This pin is typically inverted and then connected to IRQ13. Floating-point errors are cleared in the same manner as in a compatible PC-AT. However, the Nx586 detects and traps the I/O writes which normally clears the error and performs the clearing internally. Therefore, the supporting AT compatible chipset does not require a dedicated signal to the processor to clear floating-point errors.

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, NxCLK, PHE1, and PHE2. These signals determine one of four modes: Phase-Locked Loop (the normal operating mode), External Phase Inputs, Reserved, or External Processor Clock, as shown in Figure 27 and described in the sections below. Note, each clocking mode is determined at power-on RESET*. PHE2 determines the relationship between the internal non-overlapping clocks. When pulled low, narrow non-overlapped clocks are generated. Wide non-overlapped clocks are produced for PHE2 pulled high.

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

Figure 27 Clocking Modes

Mode #0: In the *phase-locked loop* mode, the internal clock phases are derived from the external NxCLK clock via a phase-locked loop (PLL). In all modes, the NxCLK 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.

Mode #1: In the *external processor clock* mode, the internal clock phases are derived from PHE1 input signal. The PHE1 input signal operates at twice the frequency of NxCLK. The falling edge of the internal phase2 will occur before the rising edge of XREF, which is a buffered NxCLK output, and can be observed on the XPH2 output. This mode allows bypassing the internal PLL for test purposes or to change the clock frequency, as when entering or leaving a low-power mode.

Mode #2: This is a reserved mode.

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 internally to drive the processor clock distribution system.

Bus Operations

This chapter covers NexBus processor cycles, NexBus^5 system bus cycles and cache-coherency operations. The processor bus cycles are conducted primarily on NexBus although their effects can also be seen on the L2 SRAM bus. The NxCLK clock, shown in the timing diagrams accompanying this text, runs at half the frequency of the processor's internal clock.

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

The notation regarding *Source* in the left-hand column of the timing diagrams shown in this section 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 NANDed in group-signal logic. In these cases, the signal source is shown with additional notations, where the additional notations indicate the device or logic that originally caused the change in the signal.

Level-2 Asynchronous SRAM Accesses

Figure 18 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.

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. An idle cycle is shown after the last read. The processor clock, which runs at twice the rate of the NexBus clock (NxCLK), 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 NxCLK. Two pins for COE* are shown, A and B. Both pins are identical in function and transition on the rising edge of PH1. The two pins are made available for loading considerations.

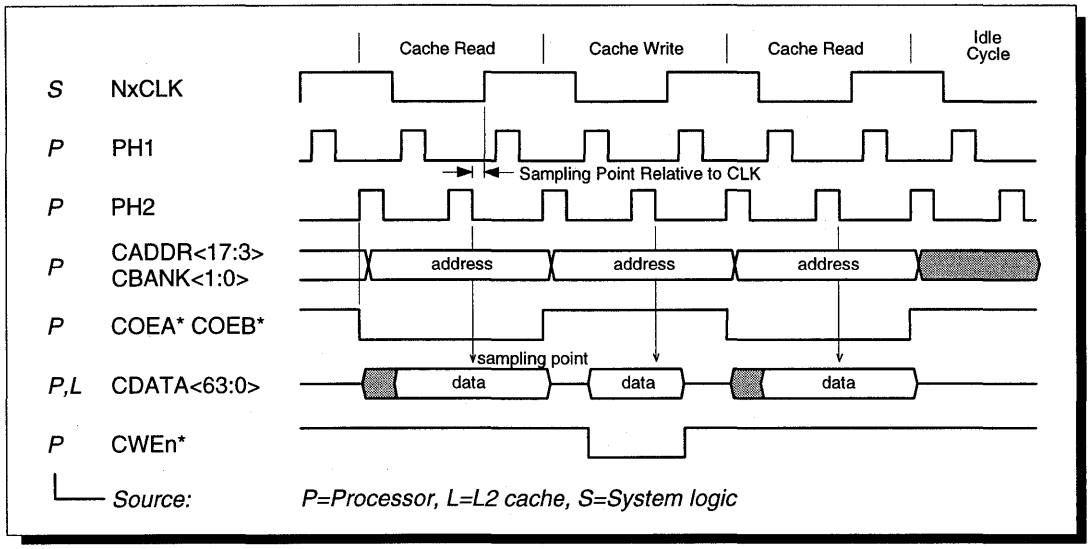


Figure 28 Level-2 Asynchronous Cache Read and Write

The L2 cache controller provides data to the processor in 3 CPU phases. In other words, the cache cycle time is 1.5 CPU clocks (one clock is equal to two phases). Data is provided to either the CPU core or the L1 cache in 1.5 CPU clocks. L2 cache address generation occurs before the cycle starts. A total of 7.5 clocks are required for a cache line fill, as shown is figure 29.

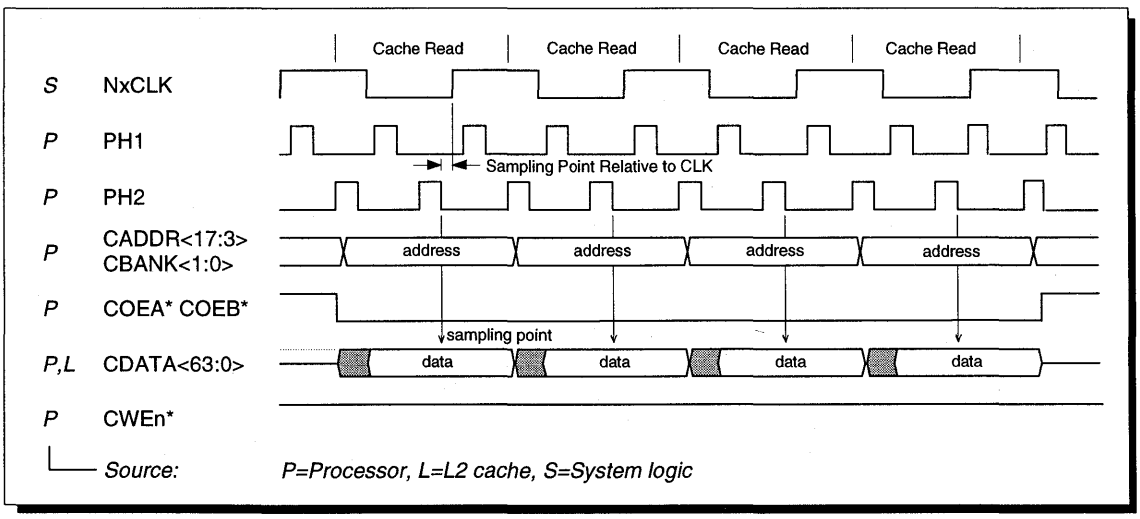


Figure 29 Level-2 to Level-1 Asynchronous Cache Line Fill.

Level-2 Synchronous SRAM Accesses

The type of SRAMs required for synchronous mode are "Synchronous Flow Through" with wide I/O (32 or 36 bits). A single clocking pin, CKMODE is used to initiate the read/write operations. At the rising edge of CKMODE, all addresses, write-enables, chip selects and data are registered within the SRAM. It is assumed that new signals can be applied to the SRAMs prior to data out valid. Read data is sampled on the next rising edge of CKMODE (approximately two PH2 clocks later). A dead cycle for bus turn around time is provided during read followed by write cycles (approximately one PH2 clock). Figure 30 shows the signal relationships for the synchronous SRAM mode.

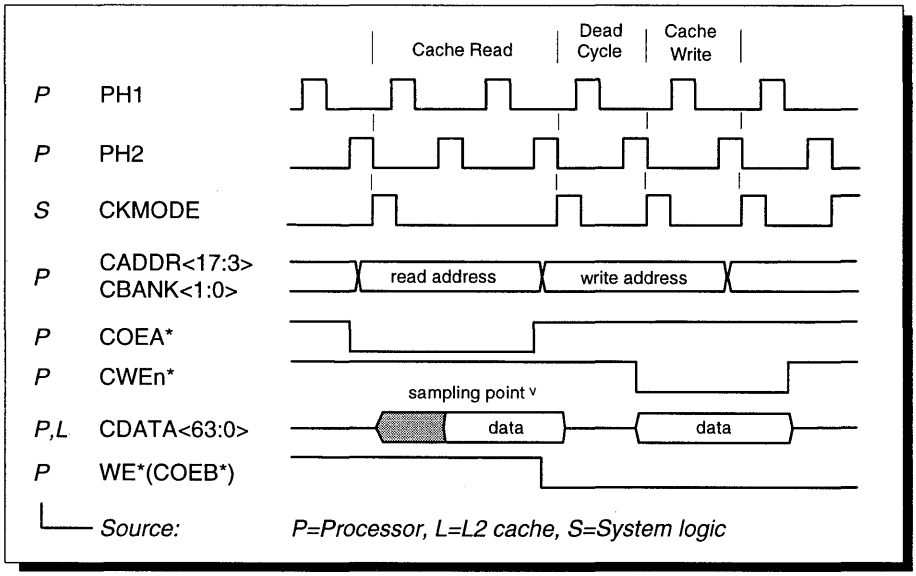


Figure 30 Level-2 Synchronous Cache Read and Write cycles

NexBus and NexBus5 Arbitration and Address Phase

Processor operations on NexBus/NexBus5 may or may not begin with arbitration for the bus. To obtain the bus, the processor asserts NREQ*, LOCK*, and/or AREQ* to the arbiter, which responds to the arbitration winner with GNT*. Automatic re-grant occurs when the 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 arbiter. In systems using the systems logic that interfaces directly to NexBus, NREQ* is typically 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 arbiter to determine the end of such a sequence. Cache-block fills are not locked; they are implicitly treated as atomic reads. 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 NexBus5 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 NexBus5.

NexBus Basic Operations

The Nx586 supports two local bus interfaces, NexBus and NexBus5. NexBus is considered a true CPU or processor local bus. Where as, NexBus5 is a NexGen proprietary system bus. During RESET* active, the XCVERE* pin is sampled for the local bus mode. XCVERE* determines what type of bus is generated by the processor. When pulled high, the Nx586 will generate the NexBus standard which requires external transceivers to connect the processor to the NexBus5 system bus. Figure 31 and 32 show the NexBus transceiver control signals for basic QWORD Read and Write operations. AD<63:0> is the multiplexed NexBus processor local bus while NxAD<63:0> is the multiplexed NexBus5 system local bus.

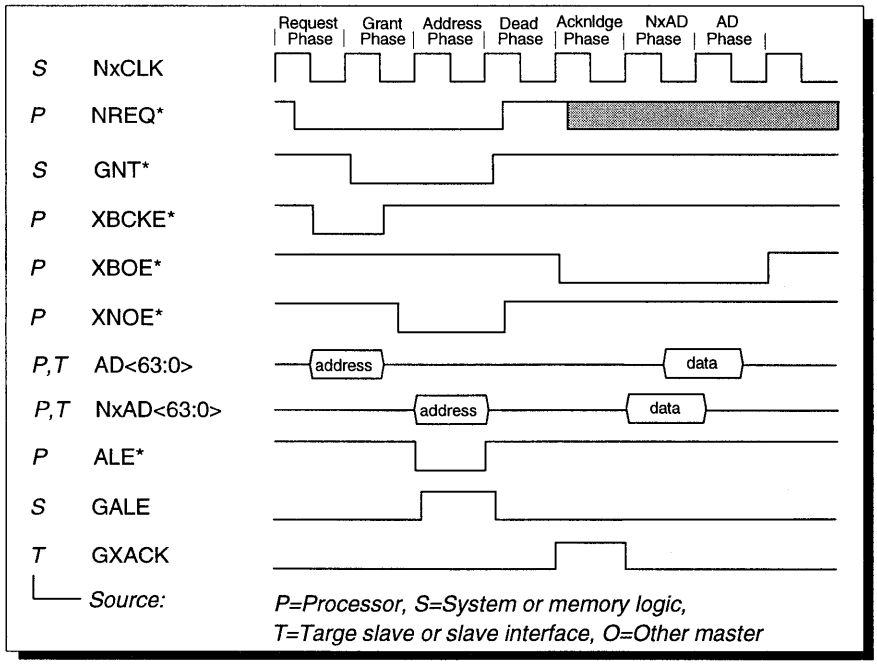


Figure 31 Fastest NexBus Single-Qword Read

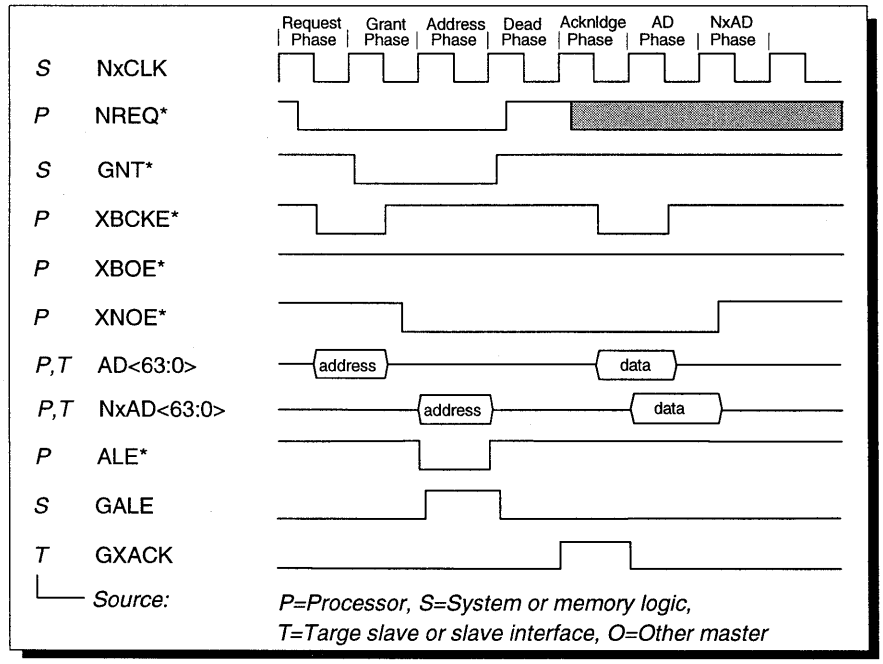


Figure 32 Fastest NexBus Single-Qword Write

NexBus5 Single-Qword Memory Operations

Figure 33 shows the fastest possible NexBus5 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 signals (such as GXACK) is shown with additional symbols indicating the device or logic that output the originally activating signal. For example, the source of the GXACK signal is shown as "S,P", 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 with additional symbols 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 systems logic interface between the NexBus5 and other system buses (called the *alternate-bus interface)* that the address must reside on the other system bus. In that case, the systems logic interface to that other bus assumes the role of slave and asserts GXACK.

Figure 33 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. In the example shown in Figure 34, the slave asserts GXACK at the latest allowable time, thereby effectively inserting one 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.

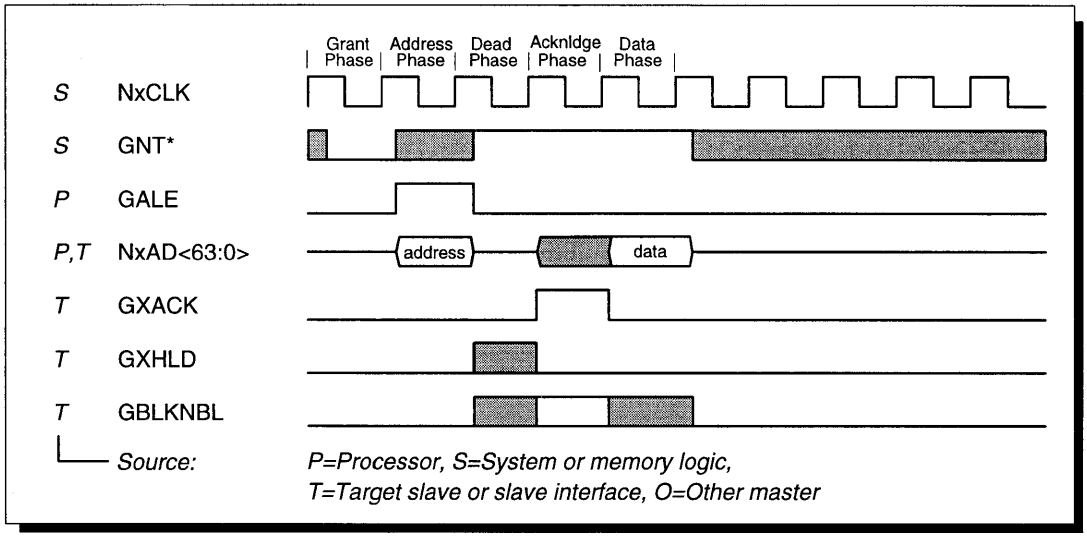


Figure 33 Fastest NexBus5 Single-Qword Read

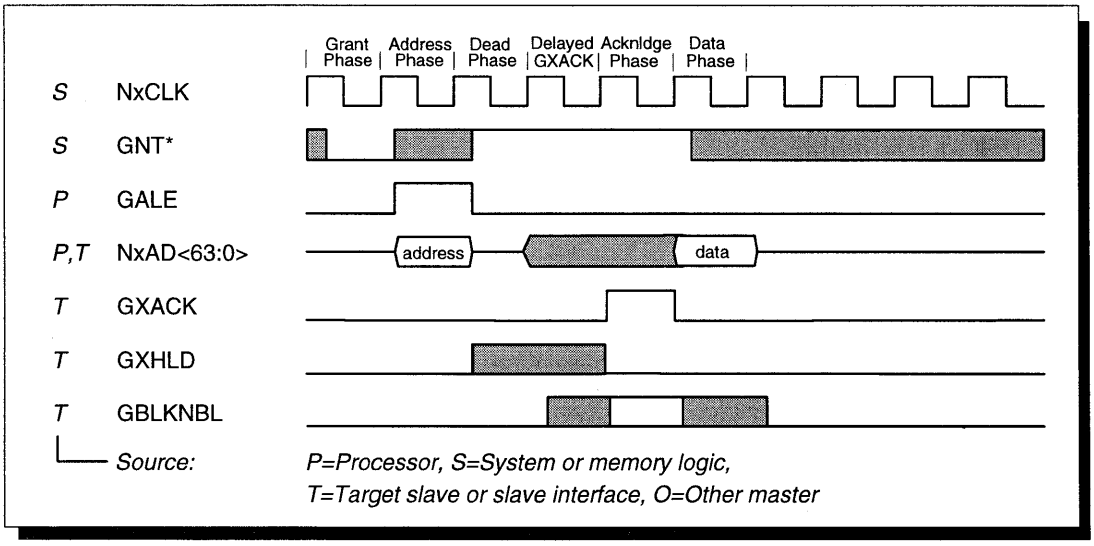


Figure 34 Fast NexBus5 Single-Qword Read with a delayed GXACK

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 35 and 36. 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 35, 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.

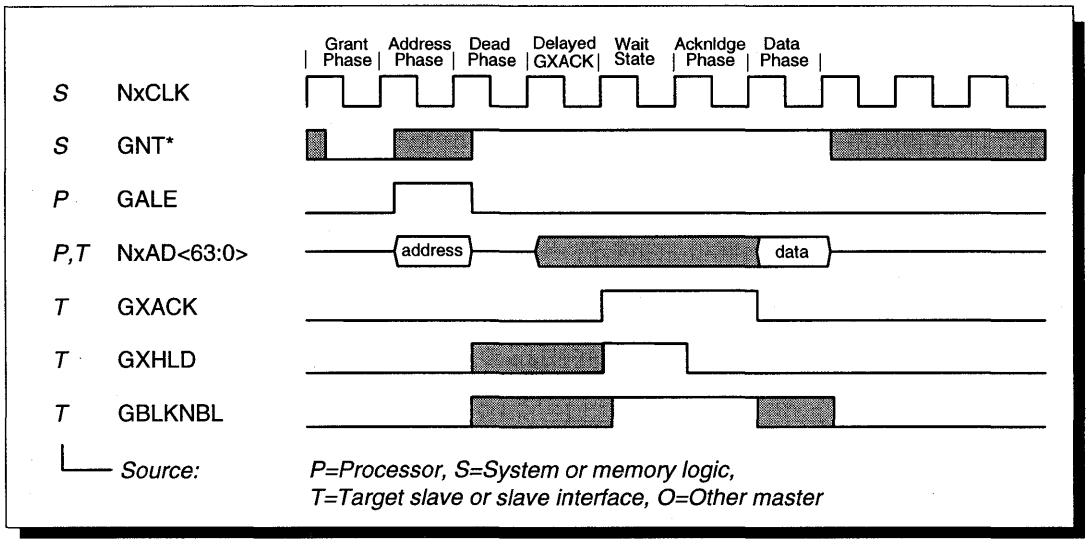


Figure 35 NexBus5 Single-Qword Read with Wait States using a delayed GXACK

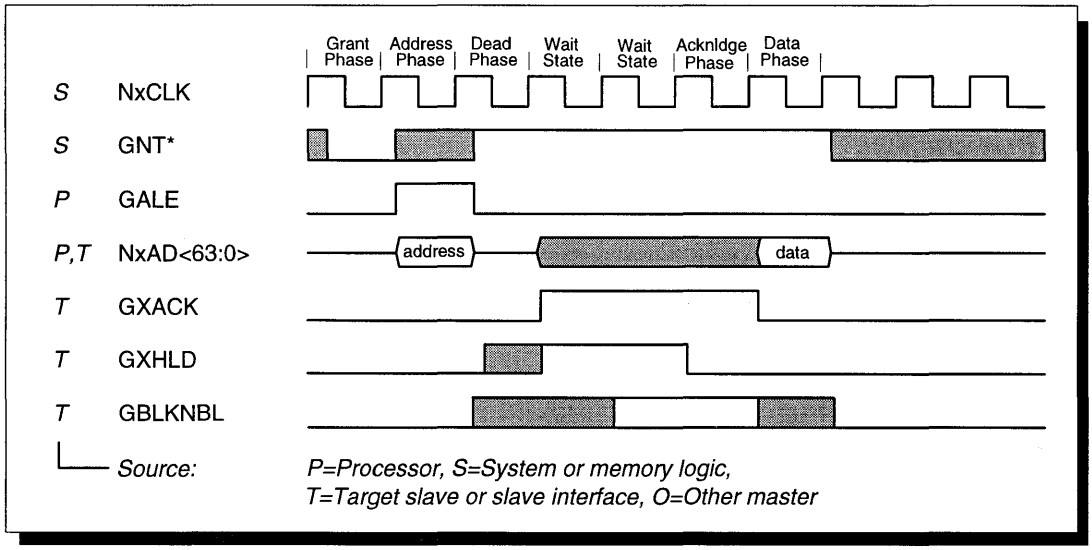


Figure 36 NexBus5 Single-Qword Read with Wait States using GXHLD only

A single-qword write operation is handled similarly. Figure 37 illustrates the fastest write operation possible. Figure 38 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.

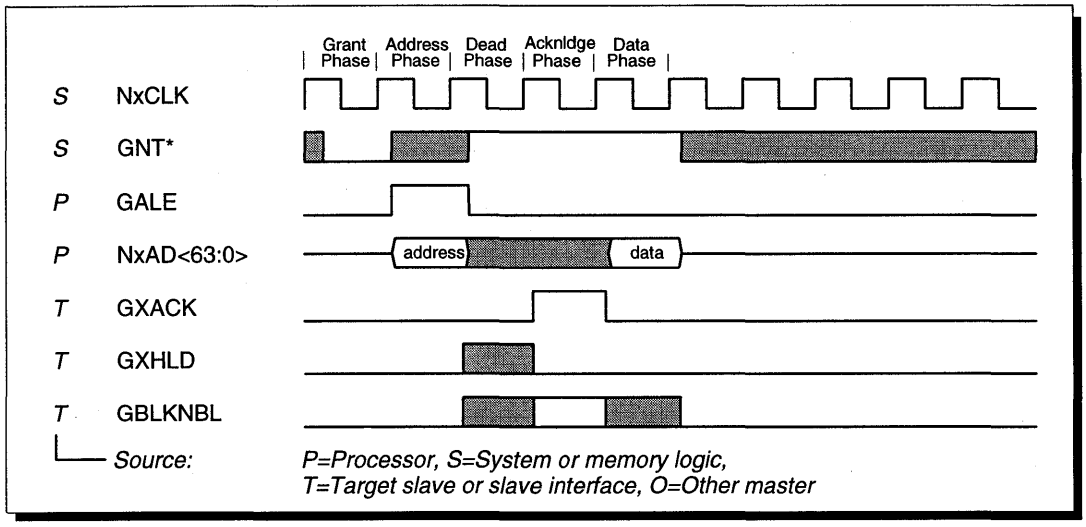


Figure 37 Fastest NexBus5 Single-Qword Write

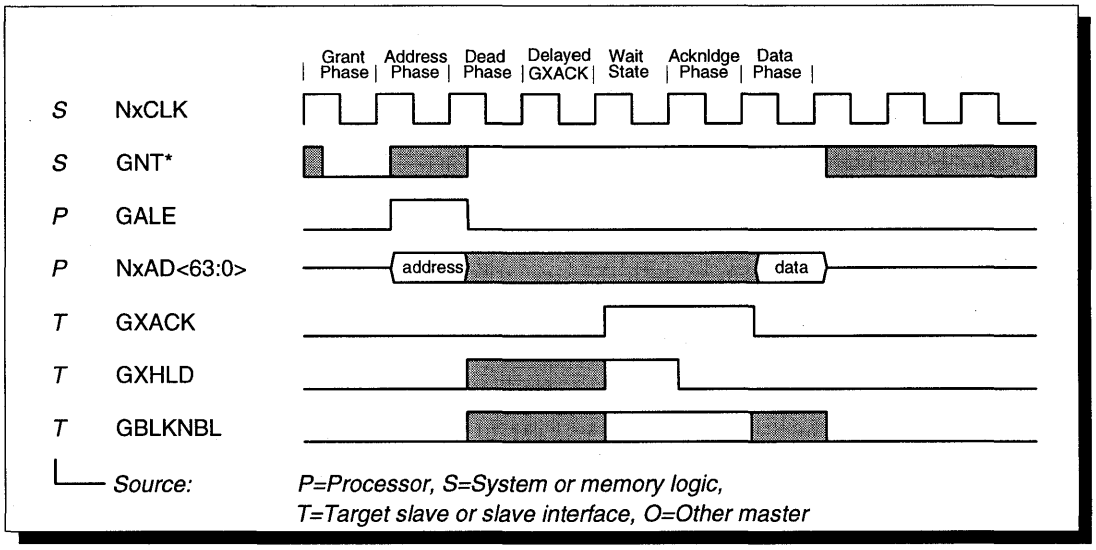


Figure 38 NexBus5 Single-Qword Write With Wait States

NexBus5 Cache Line Memory Operations

The processor performs cache line fill or block operations with memory at a much higher bandwidth than the single-qword operations described in the previous section. Block operations, 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 any order.

NexBus5 I/O Operations

I/O operations on the NexBus5 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.

NexBus5 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 39, consists of two back-to-back locked reads on NexBus5, 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.

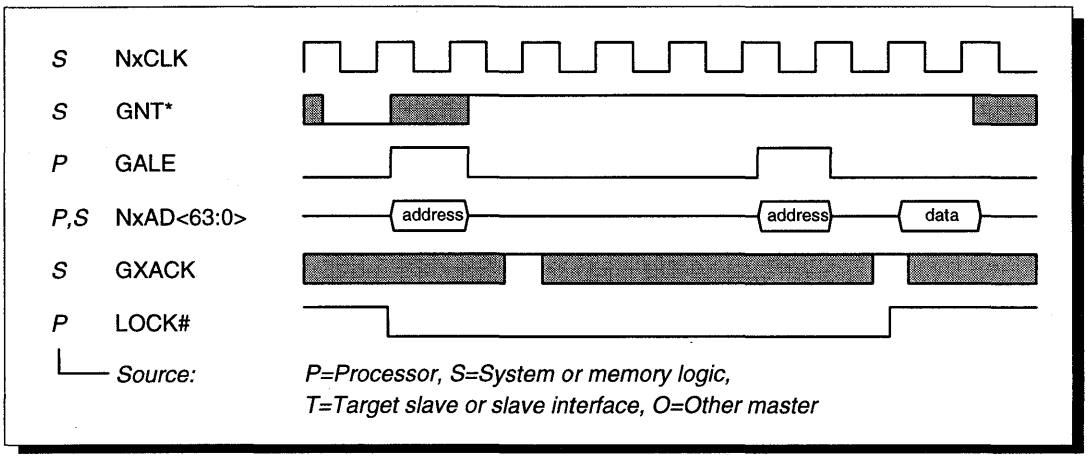


Figure 39 Interrupt Acknowledge Cycle

NexBus5 Halt and Shutdown Operations

Halt and shutdown operations are signaled on the NexBus5 by driving 001 on NxAD<48:46> during the address/status phase, as shown in Figure 40. 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	undefined	0
Shutdown	0	0	1	11111110	undefined	0

Figure 40 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 should 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 quadword wide 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^5 caching devices must snoop block reads. If any part of a block addressed by the processor's read operation resides in another NexBus^5 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^5 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^5 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^5, the systems logic interface between the NexBus^5 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.

NexBus5 Intervenor Operations

The examples given above assume that the addressed data does not reside in a modified cache block. When an operation by another NexBus5 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 NexBus5 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 NexBus5 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 NexBus5 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 NexBus5 sequence looks like a normal single-qword read from the memory followed by a block write by the intervenor. Figure 41 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 41, 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.

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.

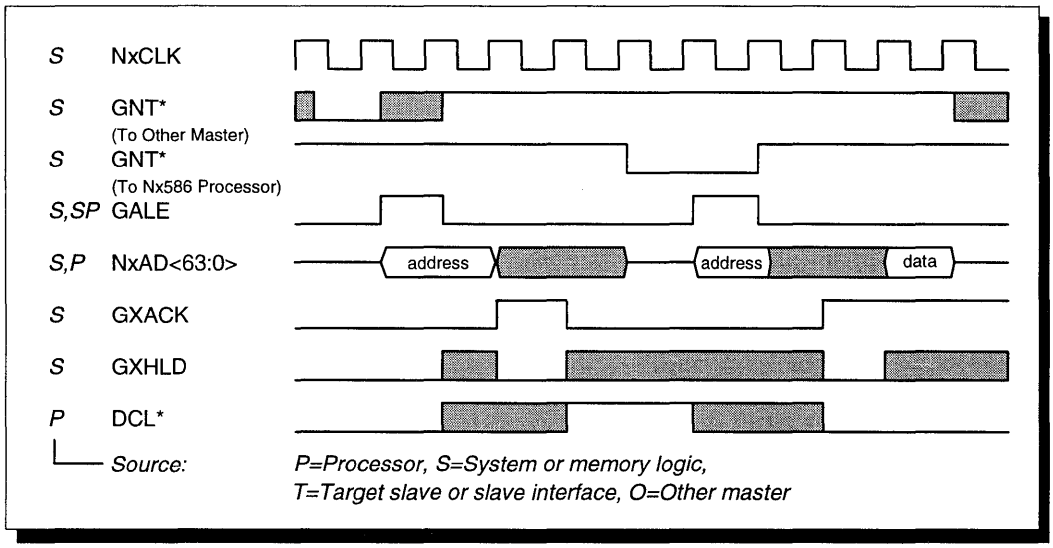


Figure 41 NexBus5 Single-Qword Read Hits Modified Cache Block

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

As described above for single-qword operations, a block read by another NexBus5 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 Electrical Specifications"
Order # NxDOC-ES001-01-W

Mechanical Data

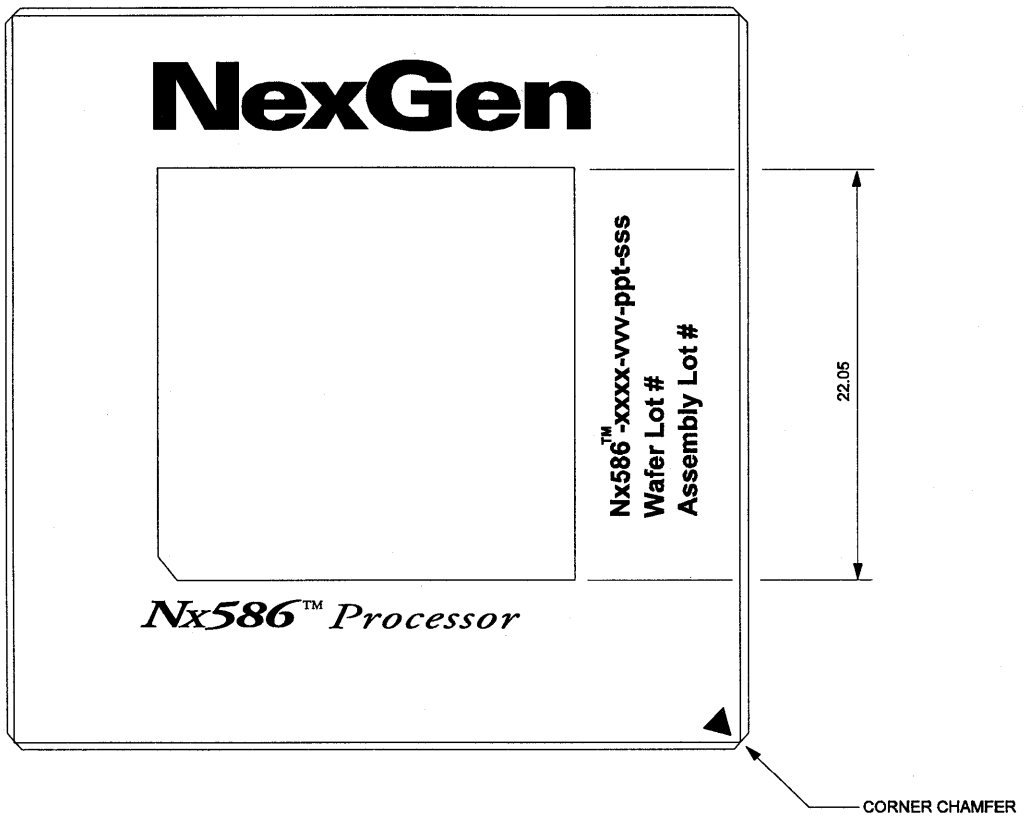


Figure 42 Nx586 Package Diagram (top)

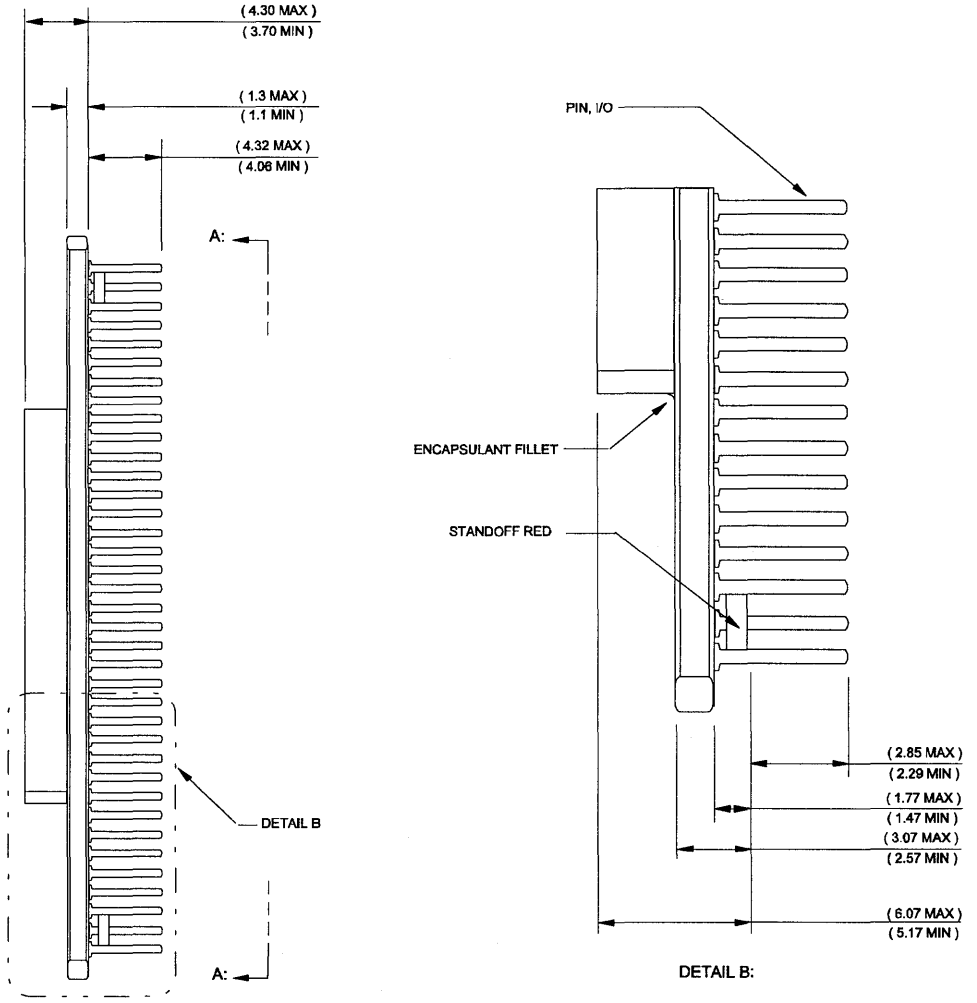


Figure 43 Nx586 Package Diagram (side)

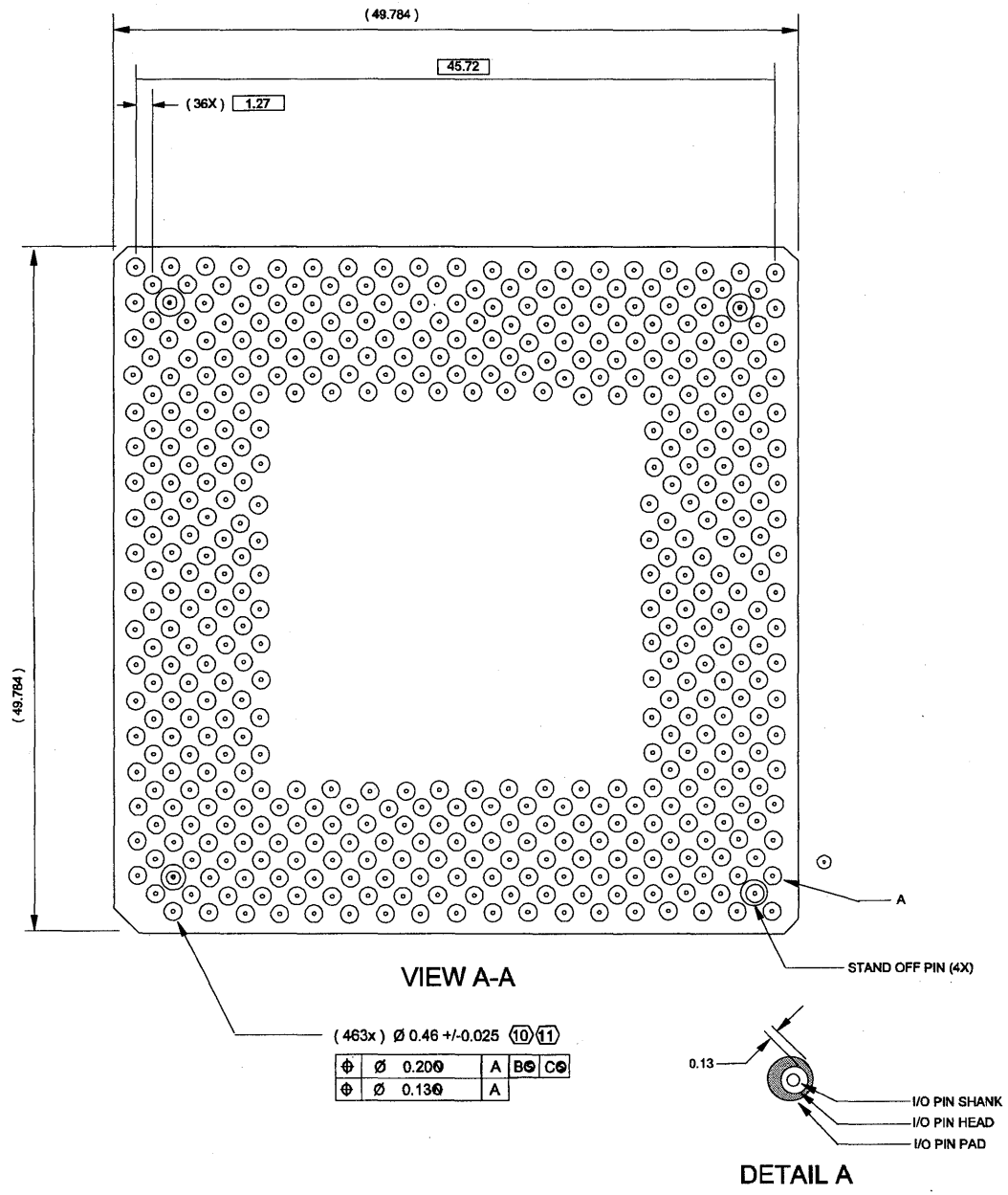


Figure 44 Nx586 Package Diagram (bottom)

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.

Asserted—For signals, "asserted" means driven to the state which asserts the description of the signal.

Active High—The signal or memory bit drive to its "asserted" state which is logically or physically high. For a memory bit, this would be a "1". For a signal, this would be near VCC voltage level.

Active Low—The signal or memory bit drive to its "asserted" state which is logically or physically low. For a memory bit, this would be a "0". For a signal, this would be near GND voltage level.

b—Bit.

B—Byte.

Bandwidth—The number of bits per second that can be processed by a memory, arithmetic unit, input/output processor, or communication system.

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

Branch Prediction—The use of history, statistical methods, or heuristic rules to predict the outcome of conditional branches.

Buffer—A fraction of real memory or a group of registers that serve as a buffer for data flowing to and from auxiliary memory.

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. The master to which bus ownership is granted controls only its own interface with the arbiter. The arbiter, 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 Coherence—The protocol among multiprocessors with private caches that assures that each variable in the shared memory space receives writes in a serial order, and no processor sees that sequence of values in any other order.

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.*

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.

Conditional Branch—A computer instruction that alters the sequence of execution if a condition is true, and otherwise falls through to the next instruction in sequence.

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 (NxCLK). 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 (NxCLK).

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

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).

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 Execution Unit—The Floating Point Execution Unit. The logic in the Floating Point Execution unit 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 a cache block to *invalid*.

L1 or Level-1—The level-1 or primary cache is located on the Nx586 processor chip.

L2 or Level 2—The level-2 or secondary cache is 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*.

Master—The Master is a device on the NexBus that initiates a transaction.

Memory—A RAM or ROM subsystem located on any bus, including the *main memory* most directly accessible to a processor. 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.*

Negated— For signals, "negated" means driven to the state which de-asserts the description of the signal. Or the opposite of "asserted".

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.

Nx586—The Nx586 processor (CPU).

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

Octet—A unit of data consisting of eight bytes and addressed on an eight-byte boundary.

Operation—See *bus 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.

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

PLL—Phase-locked loop.

POST—Power On Self Test. This procedure is performed when power is first applied to check the functionality of the system..

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. A eight-byte unit of data that is addressed on an eight-byte boundary.

Register Renaming—A technique used in processor design that assigns idle registers to serve in the place of program specified registers in order to avoid conflicts that could stall pipeline flow momentarily.

RISC—Reduced Instruction-Set Computer. A computer in which all instructions are simple instructions that take one cycle to execute, except possibly for delays introduced by conditional branches and cache misses.

Scalar Operation—Any operation performed on individual data.

Scalar Processor—A processor whose basic operations manipulate individual data elements rather than vectors or matrices.

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 system buses include the VL-Bus, PCI-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.

Superscalar—A computer architecture in which multiple scalar instructions are decode in each clock cycle sot that the instruction completed per cycle exceeds 1.0.

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

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

Write-Back Cache—A cache in which WRITEs to memory are stored in cache and written to memory only when a rewritten item is removed from cache.

Write-Through Cache—A cache in which WRITEs to memory are recorded concurrently both in cache and in main memory. The result is that the main memory slways contains valid 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