intel®

APPLICATION NOTE

AP-344

October 1990

Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors

Some portions of this document were provided by Dr. Design of San Diego, CA

Order Number: 292076-001

Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors

CONTENTS

PAGE

1.0 INTRODUCTION 1-497
- 1.1 Scope 1-497
- 1.2 Fundamental Assumptions 1-497

2.0 GENERIC IMPLEMENTATION ISSUES 1-497
- 2.1 Block Definitions 1-497
- 2.2 Clocking 1-499
- 2.3 Reset Retiming 1-499
- 2.4 CA and PORT Generation 1-500
- 2.5 Arbitration 1-501
  - 2.5.1 Refresh Requests 1-501
  - 2.5.2 82596 Requests 1-501
  - 2.5.3 Arbiter Implementation 1-502
- 2.6 Signal Conversion 1-504
- 2.7 Wait State and Burst Generator .. 1-505
  - 2.7.1 General Information 1-505
  - 2.7.2 Single Cycle Bus Transfers 1-505
  - 2.7.3 Burst-Cycle and Multiple-Cycle Bus Transfers 1-505

3.0 MC68030/82596CA INTERFACE .. 1-506
- 3.1 Design Specifications 1-506
- 3.2 Clocking 1-506
- 3.3 Reset Retiming 1-506
- 3.4 CA and PORT Generator 1-506
- 3.5 Bus Arbiter 1-507
- 3.6 Memory Signal Conversion 1-507
- 3.7 Wait State and Burst Generator .. 1-508
  - 3.7.1 General Information 1-508
  - 3.7.2 Single Cycle Transfers 1-508
  - 3.7.3 Burst Cycle Bus Transfers .. 1-508

CONTENTS

Throughout this document, M68000 is used as a general reference to a family of microprocessors, which includes the MC68000, MC68020, MC68030. A reference to a particular member of the family will use the MC prefix followed by the specific number. 82596 is used as a general reference to a family of LAN coprocessors—the 82596CA, 82596DX, and 82596SX. A reference to a particular member of the family will use 82596 followed by the two letter suffix.

Table of Figures and Tables

CONTENTS

PAGE

FIGURES

TABLES

1.0 INTRODUCTION

1.1 Scope

The 82596 family of LAN coprocessors provide IEEE 802.3 MAC functions for use with 10BASE5 (Ethernet), 10BASE2 (Cheapernet), 10BASE-T (Twisted Pair Ethernet), 1BASE5 (StarLAN), and other CSMA/CD LANs with serial bit rates up to 20 Mb/s. The three members of the 82596 family differ only in the characteristics of their parallel interfaces; the FIFO and serial functions are identical. Table 1 shows the parallel bus differences.

This document describes the circuits required to interface the Intel 82596 family of LAN coprocessors with the M68000 family of microprocessors. First, general interface issues are identified and then three specific designs are provided—including the PLD equations, timing diagrams, and schematics.

- 82596CA and MC68030
- 82596DX and MC68020
- 82596SX and MC68000

1.2 Fundamental Assumptions

Each design is based on several fundamental assumptions about the memory subsystem. The circuits required to support these features are implemented in a few programmable components. If the 82596 is added to existing designs in which the required circuits are already implemented, these circuits do not have to be duplicated.

The following assumptions are made about the designs.

- The memory subsystem uses DRAM.
- Refresh request signals are asynchronous to the system clock.
- Interface logic is implemented in PLDs where possible.
- 82596 family type signals will be converted to M68000 family type signals.

2.0 GENERIC IMPLEMENTATION ISSUES

2.1 Block Definitions

Each design is broken into functional blocks. The generic block diagram is shown in Figure 1. The M68000 family and 82596 family are fixed from a functional standpoint. The designer has almost no flexibility in their connection or timing. For the purpose of these designs, the memory control block is also assumed to be fixed. It is set up only for M68000-type signals and timings. The blocks for which there is some design flexibility are grouped under the name control logic. These blocks include the following.

- **Clocking.** Provides the proper clock phases to the 82596 and M68000. It also provides the clock for the other blocks.
- **Reset Retiming.** Takes the active low $\overline{\text{RESET}}$ signal that goes to the M68000 and adjusts its timing and level to be compatible with the active high RESET for the 82596.
- **CA and PORT Generation.** Decodes the address lines and generates the Channel Attention (CA) and CPU Port ($\overline{\text{PORT}}$) signals to the 82596. The system designer selects the memory addresses to be decoded to activate these signals. The amount of decode logic will vary greatly depending on the system memory map.
- **Arbitration.** Determines which of the three master devices has control of the local bus: the M68000, the 82596, or the refresh controller. The refresh controller has the highest priority, followed by the 82596 and then the M68000. Additional master devices are supported through simple changes to the PLD equations in this block.
- **Memory Signal Conversion.** Takes the 82596 control signals, such as $\overline{\text{ADS}}$ and W/$\overline{\text{R}}$, and converts them to M68000-type control signals, such as $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$.
- **Wait State and Burst Generation.** Generates the $\overline{\text{RDY}}$ signal to the 82596 (and $\overline{\text{BRDY}}$ for the 82596CA). It also asserts the burst request ($\overline{\text{CBREQ}}$) to the memory controller.

Table 1. 82596-Family Parallel Bus Comparison

82596 Version	Address Bits	Data Bits	Parallel Clocking	Burst Access	Parity Pins	Maximum Frequency (MHz)
82596CA	32	32	x1	Yes	Yes	33
82596DX	32	32	x2	No	No	33
82596SX	24	16	x2	No	No	20

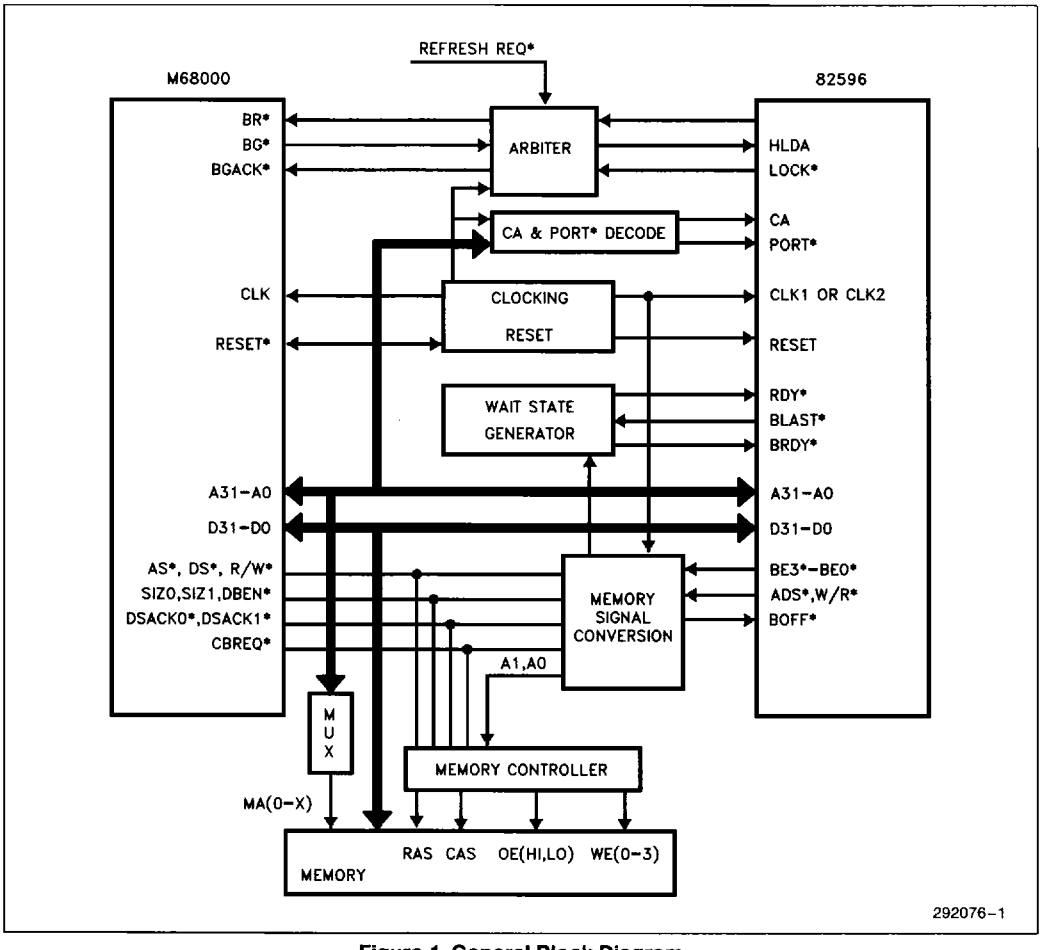


Figure 1. General Block Diagram

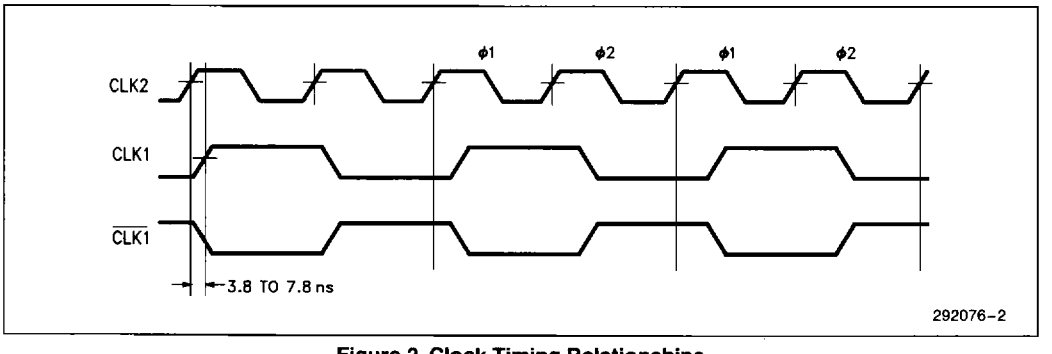


Figure 2. Clock Timing Relationships

2.2 Clocking

The 82596 family uses different types of clocking. The 82596DX and 82596SX use CLK2, which is twice as fast as the internal operating frequency. The two different phases of CLK2 for every CLK1 are defined as $\phi 1$ and $\phi 2$. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of $\phi 1$. The 82596CA uses CLK1, which is identical to the internal operating frequency. The 82596 clock timing relationships are shown in Figure 2.

In many cases the control logic is simplified by clocking it with CLK2, even if the CPU and 82596CA are clocked by CLK1 . In some other cases it is advantageous to invert the clocking signal to the 82596, which introduces a phase shift between the devices. The clocking specifications of the 82596 and M68000 are shown in Table 2.

All the designs use 74F74 flip-flops because of their high operating frequency, low propagation delay, and wide availability. If another type of flip-flop is used the timing analysis must be modified to reflect the different specifications.

2.3 Reset Retiming

Because the 82596DX and 82596SX use CLK2, the set-up and hold time specifications for the reset signal are very important. The deactivation of RESET is the only means by which the 82596DX and 82596SX determine which phases of CLK2 correspond to $\phi 1$ and $\phi 2$. Since many of the control signals are only valid at the beginning of certain phases, it is crucial that the arbitration, signal conversion, and wait state generation logic know the current phase of the clock. Failure to meet these specifications can cause improper memory accesses by the 82596. Figure 3 is the reset retiming block schematic.

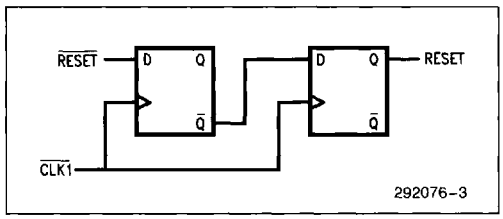


Figure 3. Reset Retiming Block

Table 2. Clocking Specifications

Component	Freq (MHz)	Clock	Max Rise	Max Fall	Min High	Min Low
			(nanoseconds)			
82596CA	25	CLK	3.0	3.0	14.0	14.0
82596CA	33	CLK	3.0	3.0	11.0	11.0
82596DX	25	CLK2	7.0	7.0	4.0	5.0
82596DX	33	CLK2	4.0	3.0	4.5	4.5
82596SX	16	CLK2	8.0	8.0	5.0	7.0
MC68030	25	CLK	4.0	4.0	19.0	19.0
MC68030	33	CLK	3.0	3.0	14.0	14.0
MC68020	25	CLK	4.0	4.0	19.0	19.0
MC68020	33	CLK	3.0	3.0	14.0	14.0
MC68000	16	CLK	5.0	5.0	27.0	27.0

The timings for the M68000 active low signal $\overline{\text{RESET}}$ are not compatible with the 82596 active high signal RESET. To prevent possible metastable conditions, the M68000 $\overline{\text{RESET}}$ passes through a two-stage synchronizer before going to the 82596. This will usually require two 74F74 flip-flops. Using two stages, rather than one, greatly reduces the probability of metastable conditions in the 82596. One of the stages is also used to invert the signal. Table 3 lists the relevant specifications for RESET and $\overline{\text{RESET}}$.

Table 3. Reset Specifications

Component	Freq (MHz)	Setup (ns)	Hold (ns)
82596CA	25	8.0	3.0
92596CA	33	10.0	3.0
82596DX	25	10.0	3.0
82596DX	33	8.0	3.0
82596SX	16	13.0	4.0
MC68030	25	ND	ND
MC68030	33	ND	ND
MC68020	25	ND	ND
MC68020	33	ND	ND
MC68000	16	ND	ND

ND = Not Defined by Motorola

2.4 CA and PORT Generation

The 82596 has two inputs that do not correspond to any signals generated by the CPU: Channel Attention (CA) and CPU Port ($\overline{\text{PORT}}$). Channel Attention is always monitored by the 82596, and the falling edge is internally latched. The 82596 responds to Channel Attention by reading the system control block command word, which is stored in memory. Several fields in this command word tell the 82596 what to do; e.g., acknowledge interrupts, change the state of the command unit, change the state of the receive unit, or load the bus throttle timers.

When the 82596 does not have the bus, it examines $\overline{\text{PORT}}$. If it is active the value on the data bus is stored in the 82596 in a special register. The value on the 4 least significant bits (D3–D0) indicates one of sixteen functions. Only four functions are defined (functions 4 through 15 are reserved and should not be used).

0 Do an internal reset (pins D31–D4 are ignored).

1 Do a self test (the results are placed at the location specified by pins D31–D4).

2 Execute a Dump command (the results are placed at the location specified by pins D31–D4).

3 Move the system configuration pointer to the location specified on pins D31–D4.

$\overline{\text{PORT}}$ has more stringent requirements for setup and hold times than CA does. $\overline{\text{PORT}}$ must meet specific setup and hold times with respect to the clock and must also be active for at least two consecutive clocks. CA is only required to be active for two clocks without meeting specific setup or hold times; however, CA only has to be active for one clock if the setup and hold times are met.

Because the M68000 family does not support a separate I/O address space, all I/O functions must be memory-mapped . The addresses for CA and $\overline{\text{PORT}}$ can be selected by the designer. In the design examples, Section 3 through 5, part of one PLD is used to generate both signals. The number of input pins on the PLD will determine the address range limitations. If the PLD has fewer input pins than the number of address lines to be decoded, one of the PLD inputs should be connected to the output of a secondary decoder. This secondary decoder must meet a worst-case propagation delay, which is listed in the comment fields of the PLD equations for each design.

Implementing CA and $\overline{\text{PORT}}$ will usually require four macro-cells, which is about one-half of a standard PLD. Two are used for the actual output signals and two are used as a state machine to control the timing of the output signals. Figure 4 is the CA and PORT generator block diagram and Figure 5 is the state transition diagram.

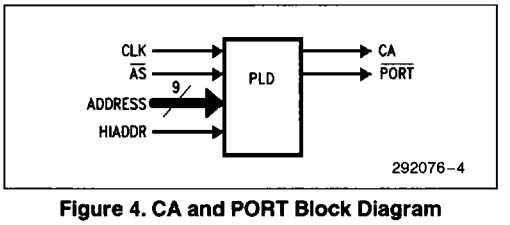


Figure 4. CA and PORT Block Diagram

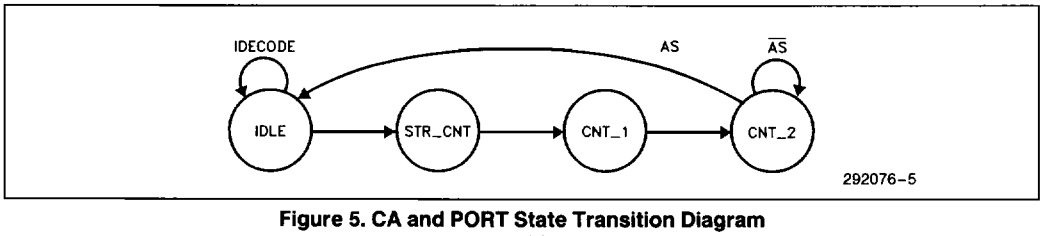


Figure 5. CA and PORT State Transition Diagram

2.5 Arbitration

All the design examples assume that the memory will use DRAMs. This means there will be at least three master devices attempting to gain access to memory: the M68000 CPU, the 82596 LAN coprocessor, and the refresh controller. The requests from the refresh controller must be given highest priority to avoid corrupting data in the DRAMs. The 82596 is given the second highest priority, so it is not forced to wait and eventually overrun or underrun. The M68000 has the lowest priority because of its internal Bus Request/Bus Grant mechanism. Because some of the M68000 family CPUs have an internal cache or instruction pipeline, they can fetch code or data internally while the 82596 or the refresh controller are using the local bus.

The M68000 family uses a three-signal arbitration scheme. A master device makes a bus request by asserting $\overline{BR}$ and waiting for the M68000 to assert $\overline{BG}$. The master device then drives $\overline{BGACK}$ while it is using the bus. When the master device no longer needs the bus it brings $\overline{BR}$ inactive, then the M68000 drives $\overline{BG}$ inactive. Finally the master device drives $\overline{BGACK}$ inactive.

2.5.1 REFRESH REQUESTS

Refresh requests are assumed to be asynchronous with the arbitration clock; therefore, the refresh signal must be synchronized—typically with a 74F74 flip-flop. At the completion of the refresh cycle the local bus will be released to the requestor having highest priority. The flip-flop is not needed if the refresh request timing is synchronous with the arbitration clock and meets the setup and hold times of the PLD.

If a refresh request arrives while the M68000 is the active bus master, the Bus Request signal ($\overline{BR}$) to the M68000 will be asserted. When the M68000 forces $\overline{BG}$ active the arbitration logic brings $\overline{BGACK}$ active and the refresh cycle begins. When the refresh has been completed $\overline{BR}$ goes inactive. If the 82596 is the active bus master when the refresh request arrives, the refresh cycle will not start until the 82596 has completed its transfers. $\overline{BR}$ to the M68000 will remain active until the refresh cycle has completed; $\overline{BR}$ will not deassert when the 82596 completes its transfers. If another 82596 request arrives during the refresh cycle, $\overline{BR}$ will remain active until both the refresh controller and the 82596 complete their transfers.

The designer is responsible for ensuring that enough refresh requests are made to avoid corrupting data in the DRAM. These designs assume that a refresh cycle signal goes into the memory controller and indicates that a refresh cycle is in progress. If a transparent technique is used for refreshing the DRAM, or if SRAM is used, then the arbiter can be greatly simplified.

There are several transparent DRAM refresh techniques. The most common method hides the refresh cycle as extra wait states in the normal CPU or 82596 accesses. This technique eliminates the arbitration overhead of the $\overline{BR}/\overline{BG}$ (HOLD/HLDA) protocol and simplifies the arbiter logic. The main disadvantage is that the wait state generator becomes more complex.

2.5.2 82596 REQUESTS

The 82596 acquires and holds the system bus via the HOLD/HLDA handshake. It requests the bus by activating HOLD. When the arbiter gives the local bus to the 82596 it asserts the HLDA signal, which is the inverted $\overline{LANCYC}$ signal from the arbiter. Overrun conditions can occur in some external devices if the 82596 holds the bus too long. The 82596's bus throttle timers can be used to regulate bus use; the timer can be activated two ways.

- **Externally.** A high state on the BREQ pin starts the timer.
- **Internally.** A high state on the HLDA pin starts the timer.

Instead of using bus arbitration schemes, the 82596CA can be forced off the bus by activating the backoff pin ($\overline{BOFF}$). This provides higher performance and faster refresh cycles. (The 82596DX and 82596SX do not have this backoff feature.)

Because the 82596 HOLD and HLDA signals are active high and the M68000 $\overline{BR}$ and $\overline{BG}$ are active low, the arbiter must invert the logic. In addition, the timings are not compatible. The arbitration signal timings are shown in Tables 4 and 5.

Table 4. Arbitration Signal Input Timings

Component	Frequency (MHz)	Signal	Output Valid Delay (ns)
82596CA	25	HOLD	3 to 22
82596CA	33	HOLD	3 to 19
82596DX	25	HOLD	4 to 22
82596DX	33	HOLD	3 to 19
82596SX	16	HOLD	4 to 32
MC68030	25	$\overline{BG}$	0 to 20
MC68030	33	$\overline{BG}$	0 to 20
MC68020	25	$\overline{BG}$	0 to 20
MC68020	33	$\overline{BG}$	0 to 20
MC68000	16	$\overline{BG}$	0 to 40

2.5.3 ARBITER IMPLEMENTATION

Local bus arbitration is mostly implemented in a synchronous PLD that uses the inverted CPU clock ($\overline{CLK1}$) as the arbitration clock. The arbiter has fixed priorities and responds to bus requests from the 82596 and the refresh controller by requesting the local bus from the M68000. The arbiter asserts the Bus Request ($\overline{BR}$) and Bus Grant Acknowledge ($\overline{BGACK}$) signals to the M68000, and enforces the bus arbitration protocol.

The arbiter does not immediately give the bus to the requestor. The arbiter is usually required to provide an adequate DRAM precharge time and will not release the bus until the precharge time has expired. The arbiter can be greatly simplified if other logic is used to control the precharge time. Figure 6 is the arbiter state transition diagram and the signal timings are shown in Figure 7.

Table 5. Arbitration Signal Output Timings

Component	Frequency (MHz)	Signal	Minimum Setup (ns)	Minimum Hold (ns)
82596CA	25	HLDA	10	3
82596CA	33	HLDA	8	3
82596DX	25	HLDA	10	3
82596DX	33	HLDA	8	3
82596SX	16	HLDA	11	8
MC68030	25	$\overline{BR}$	NA	NA
MC68030	25	$\overline{BGACK}$	NA	NA
MC68030	33	$\overline{BR}$	NA	NA
MC68030	33	$\overline{BGACK}$	NA	NA
MC68020	25	$\overline{BR}$	NA	NA
MC68020	25	$\overline{BGACK}$	NA	NA
MC68020	33	$\overline{BR}$	NA	NA
MC68020	33	$\overline{BGACK}$	NA	NA
MC68000	16	$\overline{BR}$	NA	NA
MC68000	16	$\overline{BGACK}$	NA	NA

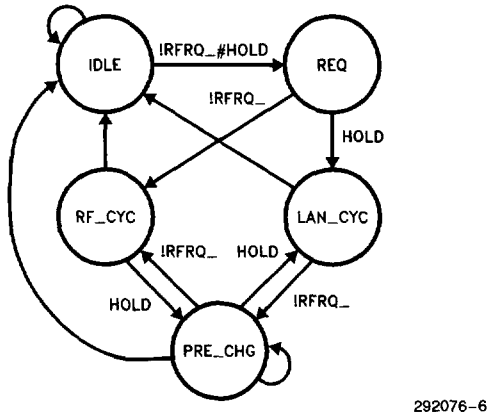


Figure 6. Arbiter State Transition Diagram

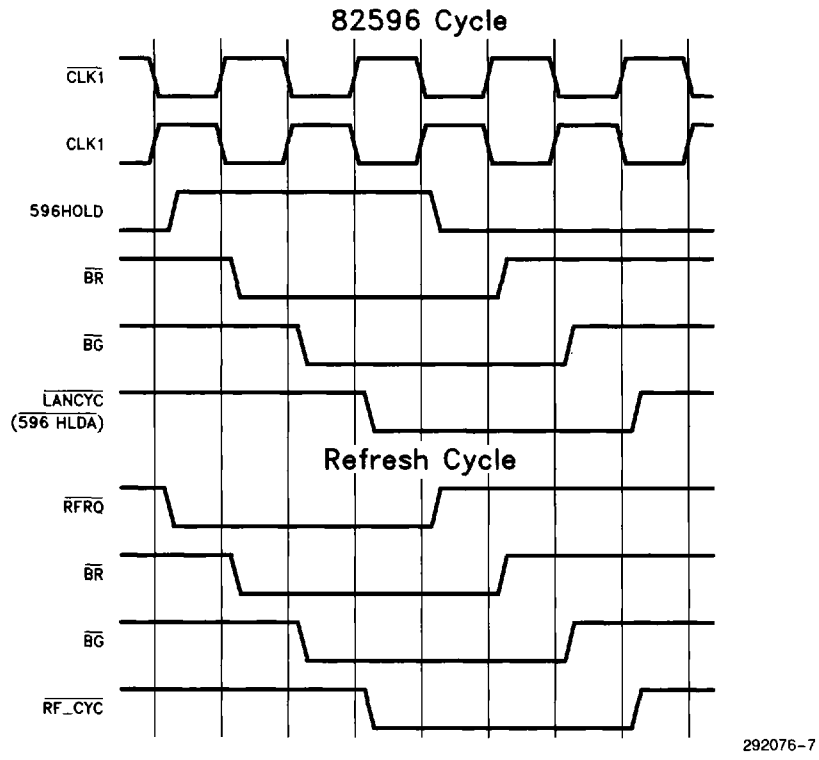


Figure 7. Arbiter Signal Timings

2.6 Signal Conversion

The memory signal conversion block modifies the 82596 bus signals to simulate M68000 signals. The new bus control signals are connected directly to the M68000's control signals and are tri-stated when the 82596 is not the bus master. This block will vary depending on which M68000 and 82596 combination is used. This block can be greatly simplified if the memory controller is capable of using both M68000 and 82596 signals and timings. The memory signal conversion block diagram is shown in Figure 8.

A single PLD generates the signals Address Strobe ($\overline{AS}$), Data Strobe ($\overline{DS}$), Read/Write (R/$\overline{W}$), and Data Bus Enable ($\overline{DBEN}$) from the 82596's signals $\overline{ADS}$ and W/$\overline{R}$. In 32-bit designs this PLD also generates SIZ0, SIZ1, A0, and A1 from the 82596's $\overline{BE0}$–$\overline{BE3}$. In 16-bit designs it generates $\overline{UDS}$ and $\overline{LDS}$ from the 82596SX's A1, $\overline{BHE}$, and $\overline{BLE}$ signals. The External Cycle Start ($\overline{ECS}$) and Operating Cycle Start ($\overline{OCS}$) signals are emulated with a tri-state buffer (e.g., a 74F244) enabled by $\overline{LANCYC}$. The input that corresponds to the $\overline{ECS}$ and $\overline{OCS}$ signals is $\overline{ADS}$ from the 82596. Figure 9 shows the different types of cycles for the M68000 and 82596.

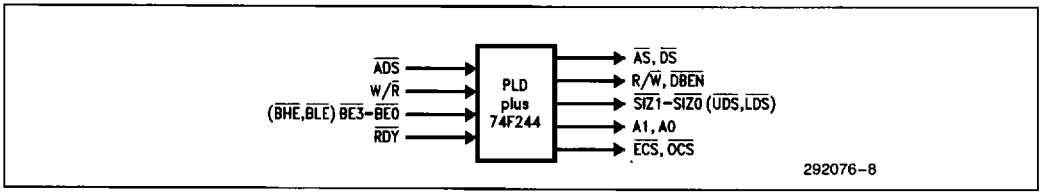


Figure 8. Memory Signal Conversion Block Diagram

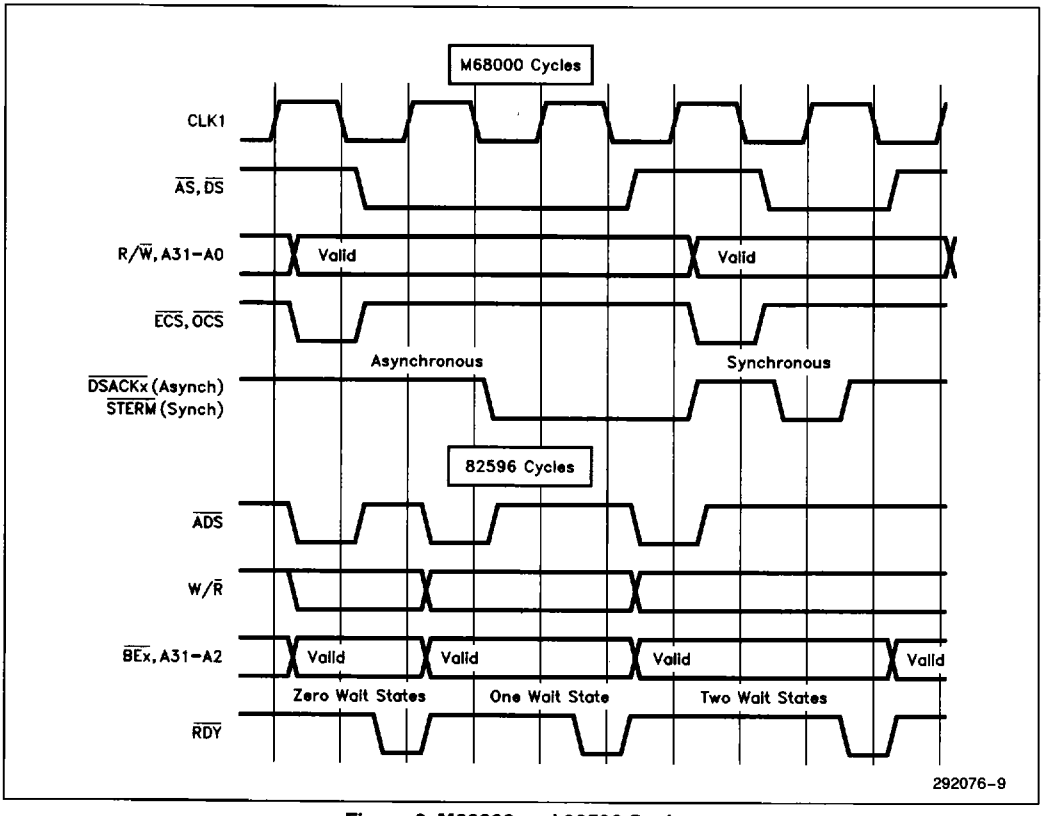


Figure 9. M68000 and 82596 Cycles

2.7 Wait State and Burst Generator

2.7.1 GENERAL INFORMATION

The 82596 and M68000 combinations can use three types of bus transfers (the 82596DX and 82596SX support only a basic single-cycle transfer). More than one single transfer can occur without interruption. Each transfer requires at least two clocks and begins with $\overline{ADS}$ going active during the first clock cycle and then $\overline{RDY}$ goes active in the last clock. Wait states are inserted by keeping $\overline{RDY}$ inactive. Figure 10 shows the wait state and burst generator block diagram.

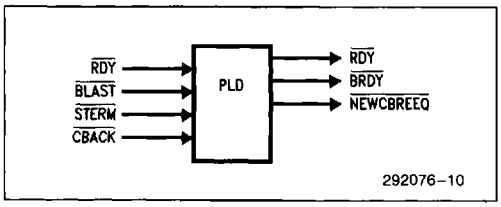


Figure 10. Wait-State and Burst Generator Block Diagram

The 82596CA supports all three types of transfers: single cycle, multiple cycle, and burst. The multiple cycle is simply several uninterrupted single-cycle transfers, with each bus cycle beginning with $\overline{ADS}$ going active during the first clock and then $\overline{RDY}$ going active in the last clock. Burst cycles can contain as many as four consecutive data transfers to four consecutive locations; however, $\overline{ADS}$ is only generated before the first data transfer. The maximum amount of data moved during a burst is 16 bytes (four 4-byte transfers). The wait state and burst generation block inserts the appropriate number of wait states during the bus cycle by driving $\overline{RDY}$ and $\overline{BRDY}$ active at the appropriate time.

The M68000 family supports a 3-clock asynchronous cycle. The MC68030 also supports a 2-clock synchronous cycle that is similar to the 82596CA burst access.

2.7.2 SINGLE-CYCLE BUS TRANSFERS

The 82596 begins a cycle after the rising edge of CLK2 (CLK for the 82596CA) by asserting $\overline{ADS}$ and driving W/$\overline{R}$ and the address lines (A31–A2) valid. The conversion PLD synchronizes $\overline{ADS}$ to the clock and generates an address strobe ($\overline{NEWAS}$). $\overline{NEWAS}$ is asserted during the same phase of the clock that a M68000 would assert $\overline{AS}$. $\overline{NEWAS}$ is also asserted during the second clock of the 82596 transfer.

The wait state generator delays the $\overline{RDY}$ signal to the 82596 from going active. This provides time to meet the data setup and hold specifications. The 82596 requires that data be set up a few nanoseconds before the rising edge of its clock. The M68000 also requires a setup time; however, it is usually only one nanosecond. The system designer will need to make provision for, at the least, the 82596 data setup time plus an additional 2 ns for clock skew. If this cannot be met another wait state will be needed for all 82596 memory read cycles. This can be provided by modifying the PLD equations to delay the assertion of $\overline{RDY}$ by one or more clocks at the end of a read cycle. The 82596CA asserts Burst Last ($\overline{BLAST}$) during the second clock of the first cycle, which indicates that the transfer is complete after a single cycle (the 82596DX and 82596SX do not use $\overline{BLAST}$).

2.7.3 BURST-CYCLE AND MULTIPLE-CYCLE BUS TRANSFERS

The 82596CA tries to burst cycles for any bus request that requires more than a single data cycle to consecutive addresses. The starting address must begin on an 8-byte boundary (xxxxxxx0h or xxxxxxx8h). The fastest burst cycle for this design assumes that 80 ns interleaved DRAMs are being used, which are fast enough to allow new data to be strobed into the 82596CA on each clock. The burst cycle requires 4 clocks for the first data strobe; however, subsequent data strobes are returned with each clock.

Burst cycles begin with the 82596CA driving a valid address and asserting $\overline{ADS}$ in the same manner as non-burst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding $\overline{BLAST}$ inactive during the second clock of the cycle. The ready logic then generates a Cache Burst Request ($\overline{NEWCBREQ}$) signal to the memory controller. If Cache Burst Acknowledge ($\overline{CBACK}$) is returned active it indicates that the memory can operate in burst mode. Then the ready logic waits for the Synchronous Termination ($\overline{STERM}$) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. When $\overline{STERM}$ is received the ready logic activates $\overline{BRDY}$ to the 82596CA, which indicates its willingness to allow a burst cycle. The 82596CA drives $\overline{BLAST}$ inactive for all but the last cycle in a burst. $\overline{BLAST}$ is driven active in the last cycle of the transfer to indicate that when either $\overline{BRDY}$ or $\overline{RDY}$ is next returned the transfer is complete. $\overline{RDY}$ is always returned in response to $\overline{BLAST}$ going active.

If the memory controller cannot perform a burst cycle CBACK will not go active and the ready logic will return RDY to the 82596CA, which indicates a non-burst multiple-cycle transfer will take place. Unlike the burst cycle, ADS will go active at the beginning of the second and all subsequent transfers in the multicycle transfer and RDY is used to end the cycle rather than BRDY.

The two data acknowledge signals for the MC68030 (DSACK0 and DSACK1) can be combined because the 82596CA only needs one RDY signal. Both DSACK signals connect to the inputs of an 74F08 AND gate.

3.0 MC68030/82596CA INTERFACE

3.1 Design Specifications

This interface example is based on the following assumptions.

- MC68030 CPU.
- 82596CA LAN coprocessor.
- 32-bit DRAM memory with burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 33 and 25 MHz operating frequencies.
- Interface logic implemented in PLDs where possible.
- 82596CA signals converted to MC68030 signal types.
- Refresh request signal asynchronous to 33 MHz clock.
- Burst accesses attempted whenever possible.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68030. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

3.2 Clocking

This design uses a clock operating at 66 MHz. It is divided by a 74F74 flip-flop to generate two 33 MHz clocks from the Q and $\overline{Q}$ outputs: CLK1 and $\overline{CLK1}$. The MC68030 uses CLK1, but the 82596CA and arbitration logic use $\overline{CLK1}$. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns.

3.3 Reset Retiming

The 82596CA reset retiming block is the same as that shown in Figure 2. The synchronizing flip-flops are clocked by $\overline{CLK1}$. There are two 82596CA specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 6 (all times are in nanoseconds).

3.4 CA and PORT Generator

The CA and $\overline{PORT}$ generation block is the same as that shown in Figure 4 and is based on a 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz). At either speed it is clocked by $\overline{CLK1}$. $\overline{AS}$, the address lines, and HIADDR are examined at the rising edge of $\overline{CLK1}$. The worst-case margin to this rising edge limits the maximum propagation delay of the secondary decoder. Each margin is calculated separately.

Because $\overline{AS}$ is generated later than the address, it is checked first. The setup time margin to the PLD's flip-flop is calculated as follows (all times are in nanoseconds).

$$\overline{CLK1}\text{ cycle} - \max \overline{AS}\text{ valid delay} - \max \overline{CLK1}\text{ to CLK1 skew} - \text{min PLD setup} \quad (3.1)$$

At 33 MHz = 30 − 15 − 2 − 10 = 3 ns (with 10 ns PLD)

At 25 MHz = 40 − 18 − 2 − 15 = 5 ns (with 15 ns PLD)

Table 6. 82596CA Worst-Case Reset Timing Margin

82596CA Frequency (MHz)	Clock-to-Output Delay Minimum	Clock-to-Output Delay Minimum	Minimum Setup	Minimum Hold	Margin Setup	Margin Hold
	(nanoseconds)					
33	3.8	7.8	8.0	3.0	14.2	0.8
25	3.8	7.8	10.0	3.0	22.2	0.8

The address has an additional margin because it is generated almost one-half clock earlier. This additional margin is calculated as follows.

$$\overline{CLK1}\text{ cycle} + \overline{CLK1}\text{ high time} - \text{max MC68030 address valid delay} - \text{max } \overline{CLK1}\text{ to CLK1 skew} - \text{min PLD setup} \quad (3.2)$$

At 33 MHz = 30 + 15 − 21 − 2 − 10 = 12 ns (with 10 ns PLD)

At 25 MHz = 40 + 20 − 25 − 2 − 15 = 18 ns (with 15 ns PLD)

Next, the worst-case setup and hold times to the 82596CA are calculated for CA and $\overline{PORT}$, which have identical timings. They go active based on the rising edge of $\overline{CLK1}$. The setup margins are calculated as follows.

$$\overline{CLK1}\text{ cycle} - \text{max PLD output valid delay} - \text{min 82596CA input setup} \quad (3.3)$$

At 33 MHz = 30 − 7 − 7 = 16 ns (with 10 ns PLD)

At 25 MHz = 40 − 12 − 7 = 21 ns (with 15 ns PLD)

The hold margins are calculated as follows.

$$\text{min PLD output valid delay} - \text{min 82596CA input hold} = 4 - 3 = 1\text{ ns (at 33 and 25 MHz)} \quad (3.4)$$

3.5 Bus Arbiter

The bus arbiter is implemented with a 20R8 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz). At either speed, it is clocked by $\overline{CLK1}$. The worst-case flip-flop setup margins to this rising edge is calculated as follows.

$$\overline{CLK1}\text{ cycle} - \text{max 82596CA HOLD output valid} - \text{min PLD input setup} \quad (3.5)$$

At 33 MHz = 30 − 19 − 10 = 1 ns (with 10 ns PLD)

At 25 MHz = 40 − 22 − 15 = 3 ns (with 15 ns PLD)

The signal $\overline{BR}$ does not need to meet any setup or hold times because it is internally synchronized by the MC68030. The PLD flip-flop setup time for $\overline{BG}$ is checked next. Because $\overline{BG}$ can go active 0 ns after the falling edge of CLK1, and because there can be up to 2 ns of skew between CLK1 and $\overline{CLK1}$, it is not completely safe to directly use $\overline{BG}$ in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, $\overline{BG}$ can go active about the same time as $\overline{CLK1}$ goes high. Because the PLD will not be clocked until the next rising edge of $\overline{CLK1}$, there will be at least one full clock cycle minus the PLD feedback delay for the output to reach a valid state.

The unused macro-cell in the 20R8 can be used to invert $\overline{LANCYC}$ to create HLDA to the 82596CA. The outputs of the arbiter, HLDA and $\overline{REFCYC}$, are internally synchronized at their destination, so no output timing analysis is required.

3.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) is used to convert the 82596CA-type control signals to MC68030-type control signals. When the 82596CA does not have the bus all the outputs go to a high-impedance state.

The signals SIZ1, SIZ0, A1, and A0 are generated by using simple combinatorial decodes of $\overline{BE3}$, $\overline{BE2}$, $\overline{BE1}$, and $\overline{BE0}$. The delay will be identical to the PLD propagation delay. The signals $\overline{NEWAS}$, $\overline{NEWDS}$, $\overline{NEWDBEN}$, and NEWR/$\overline{W}$ are generated by using the PLD's registered outputs, which are clocked by $\overline{CLK1}$. Their states are determined by the state of the 82596CA's signals $\overline{ADS}$, W/$\overline{R}$, and $\overline{RDY}$.

An 82596CA read or write cycle starts with $\overline{ADS}$ going active based on the rising edge of $\overline{CLK1}$. $\overline{NEWAS}$ and $\overline{NEWDBEN}$ will go active on the next rising edge of $\overline{CLK1}$. If the cycle is a read cycle $\overline{NEWDS}$ will also go active. If it is a write cycle $\overline{NEWDS}$ will go active one clock later. In general, the signals go inactive based on $\overline{RDY}$ going active. To meet the data hold times $\overline{NEWDBEN}$ stays active one extra clock during a write cycle. NEWR/$\overline{W}$ is simply the inverted and registered W/$\overline{R}$.

The timing of the PLD is checked next. The 82596CA control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

$$\overline{CLK1}\text{ cycle} - \text{max 82596CA output delay} - \text{PLD input setup} \quad (3.6)$$

At 33 MHz = 30 − 19 − 10 = 1 ns (with 10 ns PLD)

At 25 MHz = 40 − 22 − 15 = 3 ns (with 15 ns PLD)

NEWAS goes active based on the rising edge of CLK1, which is the same as the falling edge of CLK1. The PLD clock to output valid delay is 3 to 7 ns maximum at 33 MHz and 4 to 12 ns maximum at 25 MHz. The skew between the clocks will be −2 to +2 ns. This translates to a 1 to 5 ns delay at 33 MHz and 2 to 10 ns delay at 25 MHz, which is within the MC68030 specifications of 2 to 10 ns.

ECS and OCS are generated by taking ADS and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596CA has the bus ECS and OCS will go active about 4 to 8 ns after ADS goes active.

3.7 Wait State and Burst Generator

3.7.1 GENERAL INFORMATION

The 82596CA supports three types of bus transfers: single cycle, multiple cycle, and burst. Each bus cycle is at least two clocks long and begins with ADS going active during the first clock and RDY active in the last clock. A bus cycle contains one or more data transfers, each of which can be up to 32 bits. Burst cycles can contain as many as four data transfers, thus, the maximum amount of data moved during a burst is 16 bytes (4 transfers of 4 bytes each). The wait state and burst generation block inserts the proper number of wait states during the bus cycle. For this design it was assumed that the DRAM would allow for zero wait state accesses for the second through fourth data transfers during a burst cycle. If slower DRAMs are used, wait states will need to be inserted in the DSACK and RDY generation circuits.

3.7.2 SINGLE CYCLE TRANSFERS

The fastest single cycle transfer in this design requires three clocks for the 82596CA. The 82596CA initiates a cycle after the rising edge of CLK1 by asserting ADS and driving W/R and the address lines (A31–A2) valid. The conversion PLD synchronizes ADS and generates an address strobe (NEWAS). NEWAS is asserted during the same phase of the clock that a MC68030 would assert AS. NEWAS is also asserted during the second clock of the 82596CA transfer.

The wait state generator delays the RDY signal to the 82596CA. This provides enough time to meet the data setup and hold specifications. The 82596CA requires that data be valid at least 5 ns before the rising edge of its clock. The MC68030 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596CA has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596 memory read cycles. This can be done by modifying the PLD equations to delay the assertion of RDY by one or more clocks. The 82596CA asserts Burst Last (BLAST) during the second clock of the first cycle, which indicates that the transfer is complete after a single cycle.

3.7.3 BURST CYCLE BUS TRANSFERS

The 82596CA attempts burst cycles for any bus request that requires more than a single data cycle to consecutive addresses. The starting address must begin on an 8-byte boundary (xxxxxxx0h or xxxxxxx8h). The fastest burst cycle for this design assumes 80 ns interleaved DRAMs, which allow new data to be strobed into the 82596CA on each clock. The burst cycle requires four clocks for the first data strobe, but subsequent data strobes are returned with each clock.

Burst cycles begin with the 82596CA driving a valid address and asserting ADS in the same manner as nonburst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding BLAST inactive in the second clock of the cycle. The ready logic then generates a cache burst request (NEWCBREQ) signal to the memory controller. If the cache burst acknowledge signal (CBACK) is returned active it indicates that the memory can operate in burst mode. The ready logic then waits for the synchronous termination (STERM) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. The ready logic then activates BRDY to the 82596CA to indicate its willingness to permit a burst cycle. The 82596CA drives BLAST inactive for all but the last cycle in a burst. BLAST is driven active in the last cycle of the transfer to indicate that when RDY or BRDY is next returned the transfer is complete. RDY is always returned in response to BLAST going active.

If the memory controller cannot perform a burst cycle CBACK will not go active and the ready logic will return RDY to the 82596CA to indicate a nonburst multiple-cycle transfer will take place. This bus transfer is simply a sequence of two or more single cycle transfers. Unlike the burst cycles, ADS goes active during the first clock of the second through fourth data transfers. The timing margins for these cycles are identical to those for nonburst single cycle transfers.

Because the 82596CA requires only one RDY signal, the two data acknowledge signals for the MC68030 (DSACK0 and DSACK1) can be combined. Both DSACK signals connect to the inputs of an 74F08 AND gate.

4.0 MC68020/82596DX INTERFACE

4.1 Design Specifications

This interface example is based on the following assumptions.

- MC68020 CPU.
- 82596DX LAN coprocessor.
- 32-bit DRAM memory without burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 33 MHz operating frequency.
- Interface logic implemented in PLDs where possible.
- 82596DX signals converted to MC68020 signal types.
- Refresh request signal asynchronous to 33 MHz clock.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68020. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

4.2 Clocking

This design uses a clock operating at 66 MHz. The 66 MHz clock, CLK2, is directly by the 82596DX. It is divided by a 74F74 flip-flop to generate two 33 MHz clocks from the Q and $\overline{Q}$ outputs: CLK1 and $\overline{CLK1}$. The MC68020 uses $\overline{CLK1}$, but the arbitration logic uses CLK1. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of $\phi 1$.

4.3 Reset Retiming

The 82596DX reset retiming block is shown in Figure 11. The synchronizing flip-flops are clocked by CLK2. There are two 82596DX specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 7.

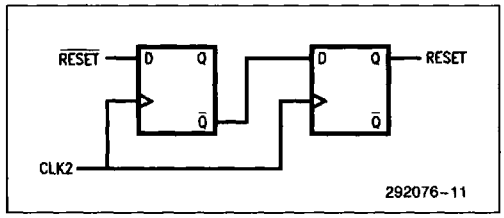


Figure 11. Reset Retiming Block

Table 7. 82596DX/SX Worst-Case Reset Timing Margin

82596 Frequency (MHz)	Clock-to-Output Delay		Minimum Setup	Minimum Hold	Margin	
	Minimum	Maximum			Setup	Hold
	(nanoseconds)					
33	3.8	7.8	8.0	3.0	14.2	0.8
25	3.8	7.8	10.0	3.0	22.2	0.8
16	3.8	7.8	13.0	4.0	45.2	−0.2

4.4 CA and PORT Generator

The CA and $\overline{PORT}$ generation block is the same as that shown in Figure 4 and is based on a 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) clocked by CLK1. $\overline{AS}$, the address lines, and HIADDR are examined at the rising edge of CLK1. The worst-case margin to this rising edge limits the maximum propagation delay of the secondary decoder. Each margin is calculated separately.

Because $\overline{AS}$ is generated later than the address, it is checked first. The setup time margin to the PLD's flip-flop is calculated as follows (all times are in nanoseconds).

$$\begin{aligned}\text{CLK1 cycle} &- \text{max MC68020 } \overline{AS} \text{ valid delay} \\ &- \text{max CLK1 to } \overline{CLK1} \text{ skew} \\ &- \text{min PLD setup}\end{aligned} \quad (4.1)$$

At 33 MHz = 30 − 15 − 2 − 10 = 3 ns
(with 10 ns PLD)

At 25 MHz = 40 − 18 − 2 − 15 = 5 ns
(with 15 ns PLD)

The address has an additional margin because it is generated almost one-half clock earlier. This additional margin is calculated as follows.

$$\begin{aligned}\text{CLK1 cycle} &+ \text{CLK1 low time} \\ &- \text{max MC68020 address valid delay} \\ &- \text{max } \overline{CLK1} \text{ to CLK1 skew} \\ &- \text{min PLD setup}\end{aligned} \quad (4.2)$$

At 33 MHz = 30 + 15 − 21 − 2 − 10 = 12 ns
(with 10 ns PLD)

At 25 MHz = 40 + 20 − 25 − 2 − 15 = 18 ns
(with 15 ns PLD)

Next, the worst-case setup and hold times to the 82596DX are calculated for CA and $\overline{PORT}$, which have identical timings. They go active based on the rising edge of CLK1. The setup margins are calculated as follows.

$$\begin{aligned}\text{CLK1 cycle} &- \text{max PLD output valid delay} \\ &- \text{min 82596DX input setup} \\ &- \text{max CLK2 to CLK1 clock skew}\end{aligned} \quad (4.3)$$

At 33 MHz = 30 − 7 − 7 − 7.8 = 8.2 ns
(with 10 ns PLD)

At 25 MHz = 40 − 12 − 7 − 7.8 = 13.2 ns
(with 15 ns PLD)

The hold margins are calculated as follows.

$$\begin{aligned}&\text{PLD output valid delay - min 82596DX input hold} \\ &+ \text{min CLK2 to CLK1 skew} \\ &= 4 - 3 + 3.8 = 4.8 \text{ ns (at 33 and 25 MHz)}\end{aligned} \quad (4.4)$$

4.5 Bus Arbiter

The bus arbiter is implemented with a 20R8 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) clocked by CLK1. The worst-case margins to this rising edge is calculated as follows.

$$\begin{aligned}\text{CLK1 cycle} &+ \text{min CLK2 to CLK1 skew} \\ &- \text{max 82596DX HOLD output valid} \\ &- \text{min PLD input setup}\end{aligned} \quad (4.5)$$

At 33 MHz = 30 + 3.8 − 19 − 10 = 4.8 ns
(with 10 ns PLD)

At 25 MHz = 40 + 3.8 − 22 − 15 = 7.8 ns
(with 15 ns PLD)

The signal $\overline{BR}$ does not need to meet any setup or hold times because it is internally synchronized by the MC68020. The PLD flip-flop setup time for $\overline{BG}$ is checked next. Because $\overline{BG}$ can go active 0 ns after the falling edge of $\overline{CLK1}$, and because there can be up to 2 ns of skew between CLK1 and $\overline{CLK1}$, it is not completely safe to directly use $\overline{BG}$ in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, $\overline{BG}$ can go active about the same time as CLK1 goes high. Because the arbiter's flip-flop will not be clocked until the next rising edge of CLK1, there will be a full clock cycle minus the PLD feedback delay for the output to reach a valid state.

The outputs of the arbiter, $\overline{LANCYC}$ and $\overline{REFCYC}$, are internally synchronized at their destination, so no output timing analysis is required. An external inverter is required for $\overline{LANCYC}$ to create HLDA to the 82596DX. If the PLD has an internal inverter then this will not be required.

4.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) is used to convert the 82596DX-type control signals to MC68020-type control signals. When the 82596DX does not have the bus all the outputs go to a high-impedance state.

The signals SIZ1, SIZ0, A1, and A0 are generated by using simple combinatorial decodes of $\overline{BE3}$, $\overline{BE2}$, $\overline{BE1}$, and $\overline{BE0}$. The delay will be identical to the PLD propagation delay. The signals $\overline{NEWAS}$, $\overline{NEWDS}$, $\overline{NEWDBEN}$, and NEWR/$\overline{W}$ are generated by using the PLD's registered outputs, which are clocked by CLK1. Their states are determined by the state of the 82596DX's signals $\overline{ADS}$, W/$\overline{R}$, and $\overline{RDY}$.

An 82596DX read or write cycle starts with $\overline{ADS}$ going active based on the rising edge of CLK1. $\overline{NEWAS}$ and $\overline{NEWDBEN}$ will go active on the next rising edge of CLK1. If the cycle is a read cycle $\overline{NEWDS}$ will also go active. If it is a write cycle $\overline{NEWDS}$ will go active one clock later. In general, the signals go inactive based on $\overline{RDY}$ going active. To meet the data hold times $\overline{NEWDBEN}$ stays active one extra clock during a write cycle. NEWR/$\overline{W}$ is simply the inverted and registered W/$\overline{R}$.

The timing of the PLD is checked next. The 82596DX control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

$$\text{CLK1 cycle} + \text{min CLK2 to CLK1 skew} - \text{max 82596DX output delay} - \text{PLD input setup} \quad (4.6)$$

$$\text{At 33 MHz} = 30 + 3.8 - 19 - 10 = 4.8 \text{ ns} \quad \text{(with 10 ns PLD)}$$

$$\text{At 25 MHz} = 40 + 3.8 - 22 - 15 = 7.8 \text{ ns} \quad \text{(with 15 ns PLD)}$$

$\overline{NEWAS}$ goes active based on the rising edge of CLK1, which is the same as the falling edge of $\overline{CLK1}$. The PLD clock to output valid delay is 2 to 7 ns maximum. The skew between the clocks will be −2 to +2 ns. This translates to a 0- to 5 ns delay, which is within the MC68020 specifications of 2 to 10 ns.

$\overline{ECS}$ and $\overline{OCS}$ are generated by taking $\overline{ADS}$ and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596DX has the bus $\overline{ECS}$ and $\overline{OCS}$ will go active about 8 ns after $\overline{ADS}$.

4.7 Wait State Generator

Each 82596DX bus cycle is at least two clocks long and begins with $\overline{ADS}$ going active during the first clock and $\overline{RDY}$ active in the last clock. The wait state block inserts the proper number of wait states during the bus cycle by delaying the $\overline{RDY}$ signal to the 82596DX. The fastest single transfer in this design requires three clocks for the 82596DX. This provides enough time to meet the data setup and hold specifications. The 82596DX requires that data be valid at least 5 ns before the rising edge of its clock. The MC68020 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596DX has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596DX memory read cycles. This can be done by modifying the PLD equations to delay the assertion of $\overline{RDY}$ by one or more clocks.

Because the 82596DX requires only one $\overline{RDY}$ signal, the two data acknowledge signals for the MC68020 ($\overline{DSACK0}$ and $\overline{DSACK1}$) can be combined. Both $\overline{DSACK}$ signals connect to the inputs of an 74F08 AND gate.

5.0 MC68000/82596SX INTERFACE

5.1 Design Specifications

This interface example is based on the following assumptions.

- MC68000 CPU.
- 82596SX LAN coprocessor.
- 16-bit DRAM memory without burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 16 MHz operating frequency.
- Interface logic implemented in PLDs where possible.
- 82596SX signals converted to MC68000 signal types.
- Refresh request signal asynchronous to 16 MHz clock.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68000. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

5.2 Clocking

This design uses a clock operating at 32 MHz. The 32 MHz clock, CLK2, is used directly by the 82596SX. It is divided by a 74F74 flip-flop to generate two 16 MHz clocks from the Q and $\overline{Q}$ outputs: CLK1 and $\overline{CLK1}$. The MC68000 uses $\overline{CLK1}$, but the arbitration logic uses CLK1. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of $\phi 1$.

5.3 Reset Retiming

The 82596SX reset retiming block is shown in Figure 11. The synchronizing flip-flops are clocked by CLK2. There are two 82596SX specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 7 (all times are in nanoseconds).

5.4 CA and PORT Generator

The CA and $\overline{PORT}$ generation block is the same as that shown in Figure 4 and is based on a 20R4-15 PLD clocked by CLK1. $\overline{AS}$, the address lines, HIADDR, $\overline{LDS}$, and CLK1 are decoded in a combinatorial macro-cell of the 20R4. The macro-cell output is sent to the input of one of the registered macro-cells, which is clocked at the rising edge of CLK1. Since propagation delay through the PLD is much less than the CLK1 cycle time, there will be a large margin on the flip-flop setup time.

Next, the worst-case setup and hold times to the 82596SX are calculated. The 82596SX timings are identical for both CA and $\overline{PORT}$. They go active based on the rising edge of CLK1. The setup margins are calculated as follows.

$$\begin{aligned} \text{CLK1 cycle} &- \text{max PLD output} \\ &- \text{min 82596SX input setup} \\ &- \text{max CLK2 to CLK1 skew} \\ &= 66 - 12 - 11 - 7.8 = 35.2 \text{ ns} \end{aligned} \quad (5.1)$$

Margins are calculated as follows.

$$\begin{aligned} &- \text{min PLD output valid delay} \\ &- \text{min 82596SX input hold} \\ &+ \text{max CLK2 to CLK1 skew} \\ &= 4 - 6 + 3.8 = 1.8 \text{ ns} \end{aligned} \quad (5.2)$$

5.5 Bus Arbiter

The bus arbiter is implemented with a 20R8-15 PLD clocked by CLK1. The worst-case margins to this rising edge is calculated as follows.

$$\begin{aligned} \text{CLK1 cycle} &- \text{max 82596SX output} \\ &- \text{min PLD input setup} \\ &+ \text{min CLK2 to CLK1 skew} \\ &= 66 - 32 - 15 + 3.8 = 22.8 \text{ ns} \end{aligned} \quad (5.3)$$

The signal $\overline{BR}$ does not need to meet any setup or hold times because it is internally synchronized by the MC68000. The PLD flip-flop setup time for $\overline{BG}$ is checked next. Because $\overline{BG}$ can go active 0 ns after the falling edge of $\overline{CLK1}$, and because there can be up to 2 ns of skew between CLK1 and $\overline{CLK1}$, it is not completely safe to directly use $\overline{BG}$ in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, $\overline{BG}$ can go active about the same time as CLK1 goes high. Because the arbiter's flip-flop will not be clocked until the next rising edge of CLK1, there will be almost 60 ns for the output to reach a valid state.

The outputs of the arbiter, $\overline{LANCYC}$ and $\overline{REFCYC}$, are internally synchronized at their destination, so no output timing analysis is required. An inverter is required for $\overline{LANCYC}$ to create HLDA to the 82596SX. If the PLD has an internal inverter then this will not be required. If not, one of the unused macrocells in the 20R8 can be used to perform the inversion.

5.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4-15 PLD is used to convert the 82596SX-type control signals to MC68000-type control signals. When the 82596SX does not have the bus all the outputs go to a high-impedance state.

The signals $\overline{UDS}$, $\overline{LDS}$, and A1 are generated by using simple combinatorial decodes of $\overline{BHE}$ and $\overline{BLE}$. The delay will be identical to the PLD propagation delay, which is 15 ns maximum. The signals $\overline{NEWAS}$, $\overline{NEWDS}$, $\overline{NEWDBEN}$, and NEWR/$\overline{W}$ are generated by using the PLD's registered outputs, which are clocked by CLK1. Their states are determined by the state of the 82596SX's signals $\overline{ADS}$, $\overline{RDY}$, and W/$\overline{R}$.

An 82596SX read or write cycle starts with $\overline{ADS}$ going active based on the rising edge of CLK2. $\overline{NEWAS}$ and $\overline{NEWDBEN}$ will go active on the next rising edge of

CLK1. If the cycle is a read cycle $\overline{NEWDS}$ will also go active. If it is a write cycle $\overline{NEWDS}$ will go active one clock later. In general, the signals go inactive based on $\overline{RDY}$ going active. To meet the data hold times $\overline{NEWDBEN}$ stays active one extra clock during a write cycle. NEWR/$\overline{W}$ is simply the inverted and registered W/$\overline{R}$.

The timing of the PLD is checked next. The 82596SX control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

$$\begin{aligned} &\text{CLK1 cycle} + \text{min CLK2 to CLK1 skew} \\ &\quad - \text{max 82596SX output delay} \\ &\quad - \text{PLD input setup} \\ &\quad = 66 + 3.8 - 36 - 15 = 18.8 \text{ ns} \end{aligned} \tag{5.4}$$

$\overline{NEWAS}$ goes active based on the falling edge of CLK1, which is the same as the rising edge of $\overline{CLK1}$. The PLD clock to output valid delay is 5 to 12 ns maximum. The skew between the clocks will be −2 to +2 ns. This translates to a 3 to 14 ns delay, which is within the MC68000 specifications of 3 to 40 ns.

$\overline{ECS}$ and $\overline{OCS}$ are generated by taking $\overline{ADS}$ and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596SX has the bus $\overline{ECS}$ and $\overline{OCS}$ will go active about 8 ns after $\overline{ADS}$.

5.7 Wait State Generator

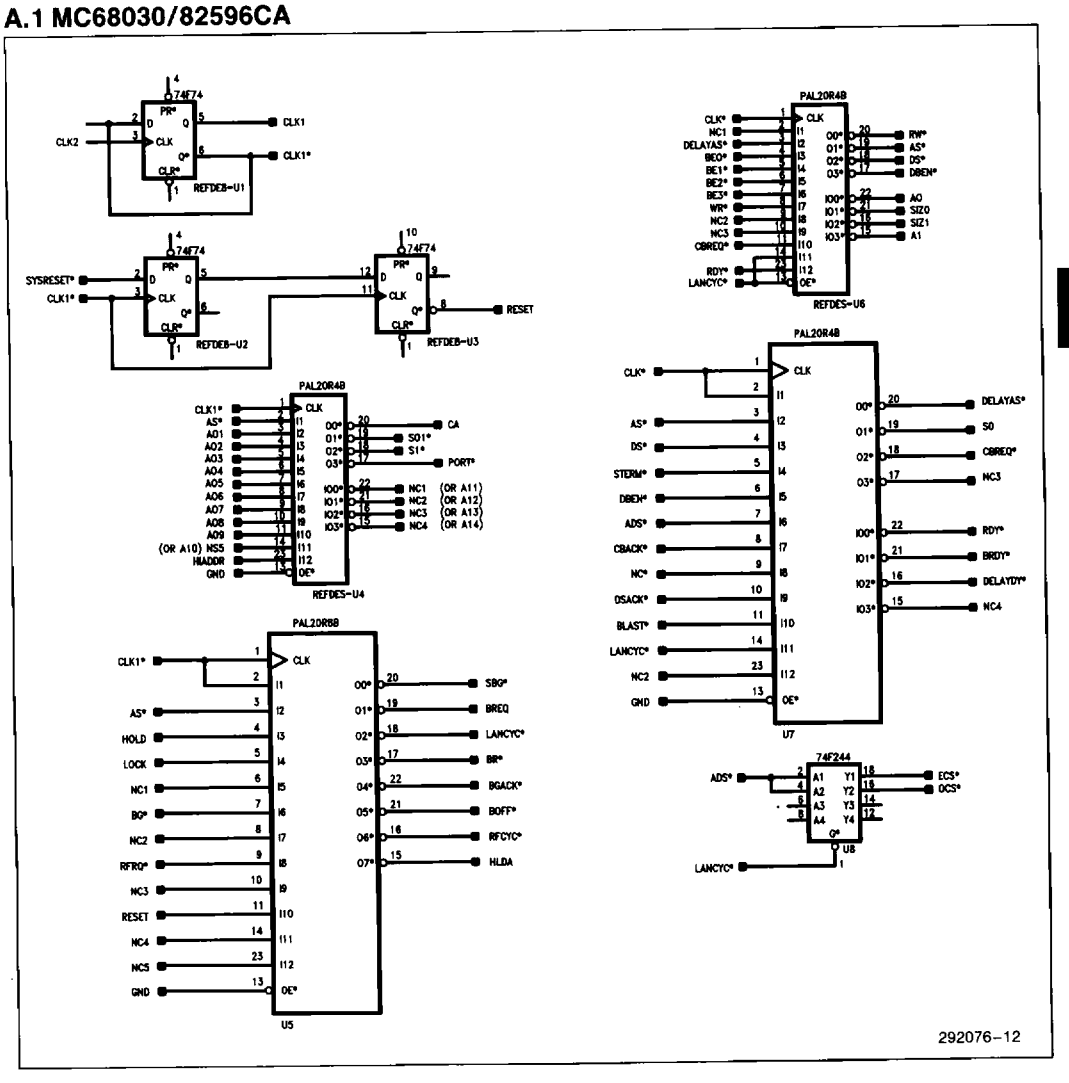
The 82596SX bus cycle is at least two clocks long and begins with $\overline{ADS}$ going active during the first clock and $\overline{RDY}$ active in the last clock. The wait state generation block inserts the proper number of wait states during the bus cycle. For this design it was assumed that the DRAM would allow for zero wait state accesses. If slower DRAMs are used, wait states will need to be inserted in the $\overline{DSACK}$ and $\overline{RDY}$ generation circuits.

The fastest single cycle transfer in this design requires three clocks for the 82596SX. The wait state generator delays the $\overline{RDY}$ signal to the 82596SX. This provides enough time to meet the data setup and hold specifications. The 82596SX requires that data be valid at least 5 ns before the rising edge of its clock. The MC68000 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596SX has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596 memory read cycles. This can be done by modifying the PLD equations to delay the assertion of $\overline{RDY}$ by one or more clocks.

APPENDIX A
SCHEMATICS

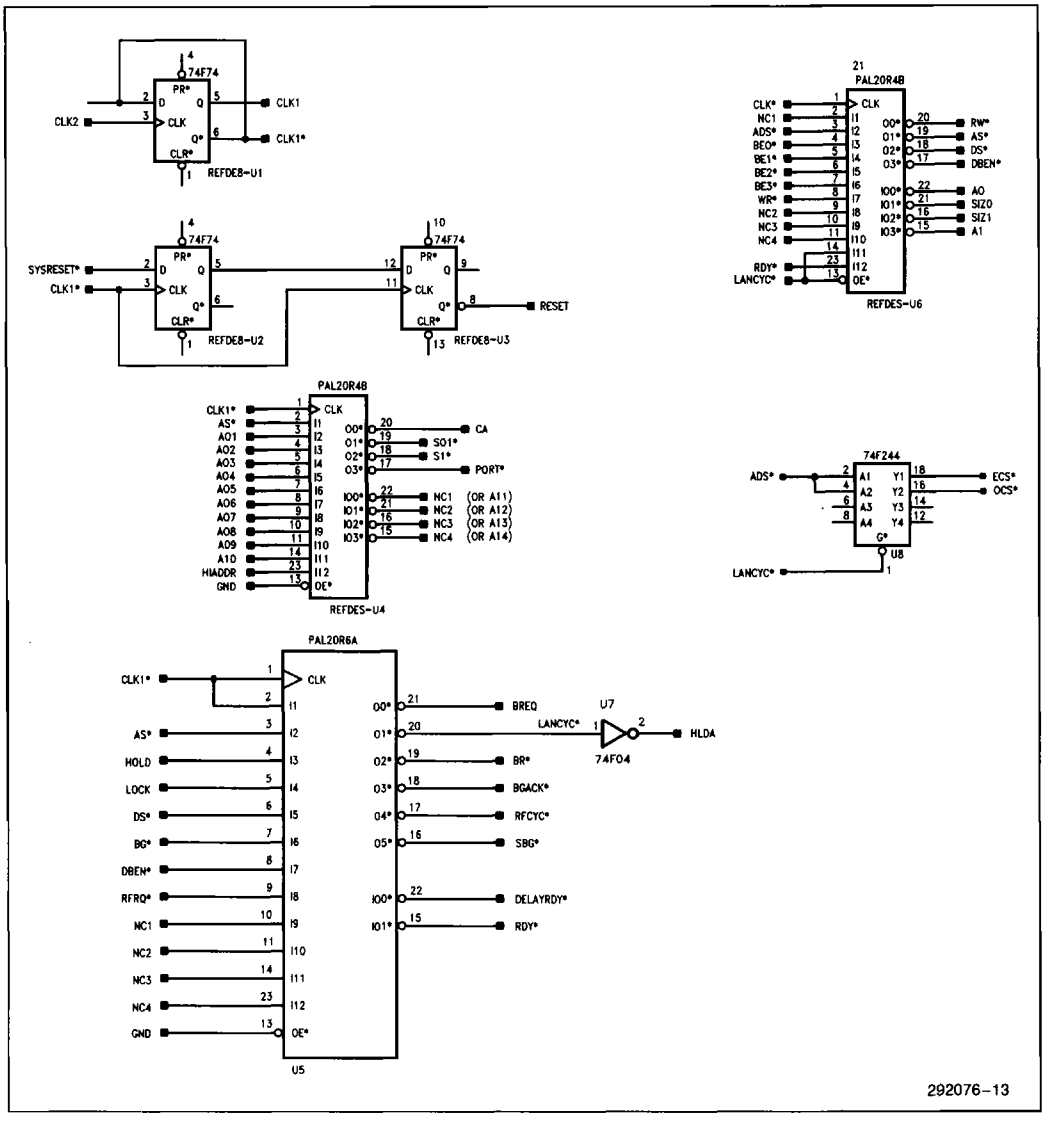
Each schematic includes only the logic needed to interface the 82596 to the M68000. The address and data bus connections between the two are not currently shown.

A.1 MC68030/82596CA

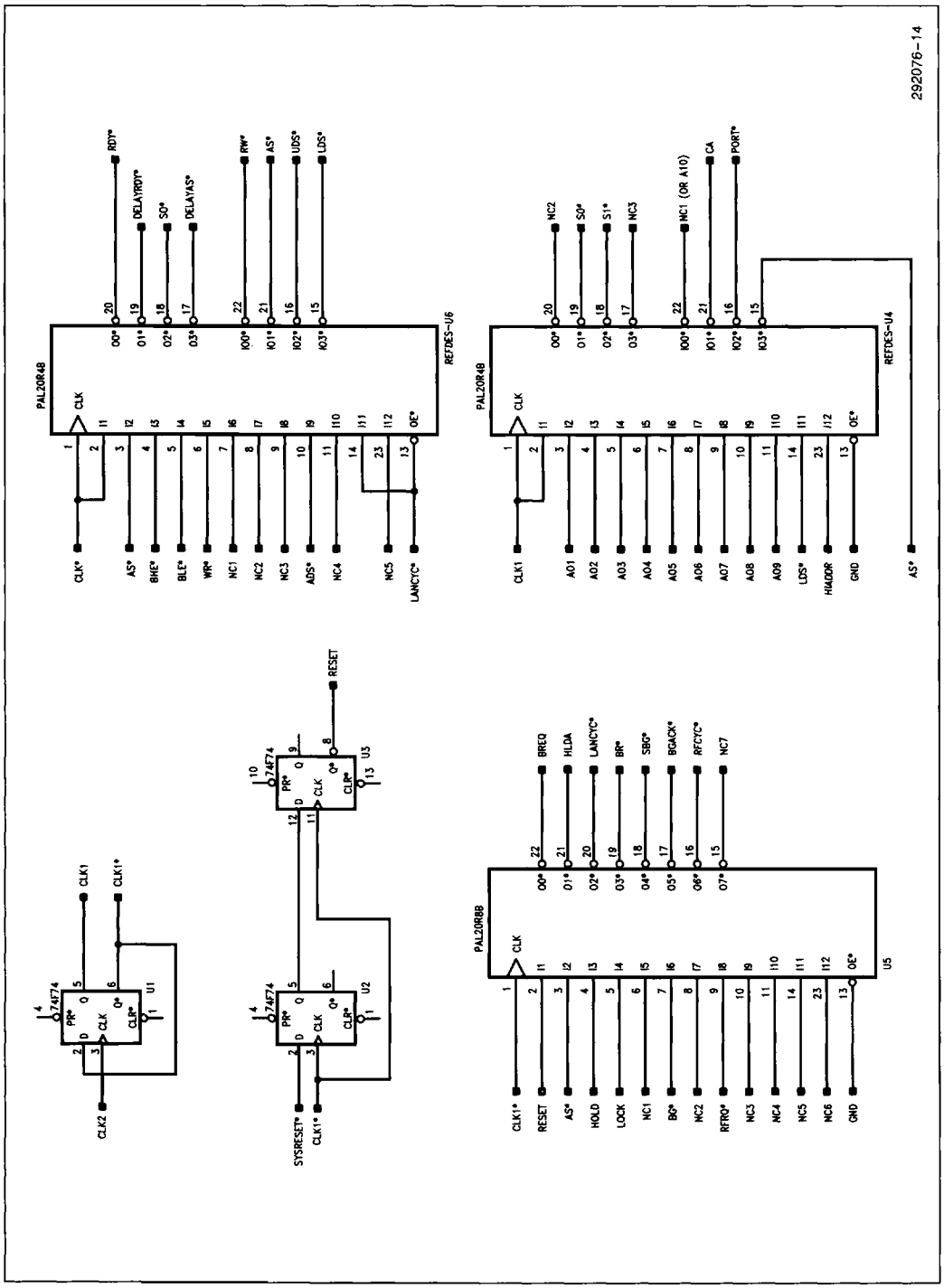


292076-12

A.2 MC68020/82596DX



A.3 MC68000/82596SX



APPENDIX B
PLD EQUATIONS

Several conventions are used in the PLD equations. These are designed to make the equations easier to understand. In general the equations are designed for PLD's with fixed macro-cells. If programmable macro-cells are available, then some reduction will be possible.

Pin signal name assignments are followed by a comment indicating the pin type:

- Input = I
- Combinatorial input/output = I/O
- Registered input/output = R, I/O

Names for active low signals are followed by an underscore. For example, AS__ is an active low signal and BREQ is an active high signal.

Logical operators are defined as:

- Logical AND = &
- Logical OR = #
- Logical NOT = !

Where possible, state and truth table formats are used.

General comments start with a " in the left-most column. Specific Comments appear on the same line as the individual equations or terms within the equation. If there is not room on the right side for the comment to fit on the same line, then it will be indented on the following line.

B.1 MC68030/82596CA

```
Module ARB  FLAG '-R3';
Title '82596CA Arbitration for Local Bus   Rev. B   02/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R8-10';

" ************************   Descrption   ************************
"
"  FOR:  82596CA / 68030 Interface
"
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus.  Refresh requests are
"  given highest priority, and the 82596 requests given second
"  highest priority.  The CPU normally controls the local bus
"  when no requests are pending.
"
"  Requestors are granted the bus by using the inverted CPU clock
"  (CLK1_).  Since CLK1_ is also used in the equations, it must
"  connected with a separate pin with a separate name (CCLK1_).
"
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to the inverted clock
"  (CLK1_).  If this set-up cannot be guaranteed, the request
"  must be synchronized through an external flip-flop, clocked
"  with CLK1_.
"
"  SBG_ is the synchronized 68030 Bus Grant (BG_) signal.

"  HLDA is the inverted LANCYC*.  Due to a lack of P-terms, HLDA
"  will be delayed by 1 clock.  If a PLD with inverter outputs is
"  available, then LANCYC* can be inverted and used directly as
"  HLDA.
"
"  The Bus Request signal, BREQ, and Backoff, BOFF, are used to
"  activiate the Bus Throttle Timers and backoff function.  The
"  equations are included but the outputs are always set
"  inactive.  It is left to the system designer to define input
"  conditions for this signal.
"
"  The two states LAN_OFF and LAN_RF can be used if the external
"  circuitry cannot guarantee that the 82596 will get off the bus
"  in time to do refresh cycles.  If these states are used, a
"  larger PLD will be needed to generate the BGACK to the 68030.
"
"  UNUSED INPUT  PINS                 :  5
"  UNUSED OUTPUT PINS (REGISTERED)    :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL) :  0
"
"*****************************************************************
```

292076-35

```
arb Device 'P20R8';

CLK1_      Pin   1; "I"          VCC        Pin 24;
CCLK1_     Pin   2; "I"          NC5        Pin 23; "I"
AS_        Pin   3; "I"          SBG_       Pin 22; "R, I/O"
HOLD       Pin   4; "I"          BREQ       Pin 21; "R, I/O"
LOCK       Pin   5; "I"          LANCYC_    Pin 20; "R, I/O"
NC1        Pin   6; "I"          BR_        Pin 19; "R, I/O"
BG_        Pin   7; "I"          BGACK_     Pin 18; "R, I/O"
NC2        Pin   8; "I"          BOFF_      Pin 17; "R, I/O"
RFRQ_      Pin   9; "I"          RFCYC_     Pin 16; "R, I/O"
NC3        Pin  10; "I"          HLDA       Pin 15; "R, I/O"
RESET      Pin  11; "I"          NC4        Pin 14; "I"
GND        Pin  12;              OE_        Pin 13; "I"

MODE    = [BR_, BGACK_, RFCYC_, LANCYC_, BOFF_];

IDLE    = [1,1,1,1,1];
REQ     = [0,1,1,1,1];    " Generic request to CPU for local bus.
RF_CYC  = [1,0,0,1,1];    " Refresh request has been granted.
LAN_CYC = [1,0,1,0,1];    " LAN request has been granted.
PRE_CHG = [1,0,1,1,1];    " Required for back-to-back cycles.

"  The following two lines are used only if the 82596 is required to
"  be kicked off the bus.  Most designs will not require these states.

" LAN_OFF = [1,0,1,0,0];    Refresh Request forces 82596 off bus.
" LAN_ON  = [1,0,0,1,0];    Refresh Request returns control to 82596


Equations

BREQ  := 0;               " Bus Throttle conditions will need to be
                          " defined by the system designer.

SBG_ = !BG_ & CCLK1_      " Set synchronized bus grant during high clock
                          "    to phase meet setup to ARB PLD.
     # !SBG_ & !BG_       " Hold until processor bus grant goes away.
     # !SBG_ & !CCLK1_    " Hold through low clock phase to met setup.

HLDA := !LANCYC_;         " Create HLDA from inverted LANCYC_.
```

292076-36

```
MODE  := RESET & IDLE;  " Initialize state machine to IDLE State on reset.


State_Diagram IN arb MODE

state IDLE      : IF (!RFRQ_ # HOLD) THEN REQ
                                     ELSE IDLE;

state REQ       : CASE (!RFRQ_ & !SBG_)                    :RF_CYC;
                       (HOLD & RFRQ_ & !SBG_)              :LAN_CYC;
                       (!((!RFRQ_ & !SBG_)
                         # (HOLD & RFRQ_ & SBG_)))         :REQ;
                  ENDCASE;

state RF_CYC    : CASE (RFRQ_ & !HOLD & SBG_)              :IDLE;
                       (RFRQ_ & HOLD)                      :PRE_CHG;
                       (!RFRQ_)                            :RF_CYC;
                  ENDCASE;

state LAN_CYC   : CASE (!HOLD &  RFRQ_ & !LOCK & SBG_)     :IDLE;
                       (!HOLD & !RFRQ_ & !LOCK)            :PRE_CHG;
"                      ( HOLD & !RFRQ_ & !LOCK)            :LAN_Off;
                       ( HOLD &  RFRQ_)                    :LAN_CYC;
                  ENDCASE;

state PRE_CHG   : CASE ( RFRQ_ & !HOLD & !LOCK & SBG_)     :IDLE;
                       (!RFRQ_ & !LOCK)                    :RF_CYC;
                       ( RFRQ_ &  HOLD)                    :LAN_CYC;
                       ( RFRQ_ & !HOLD & !LOCK & !SBG_)    :PRE_CHG;
                  ENDCASE;

"  state LAN_OFF :   IF (!HOLD) THEN LAN_RF
"                               ELSE LAN_OFF;

"  state LAN_RF  :   IF (RFRQ_) THEN LAN_CYC
"                               ELSE LAN_RF;

" ******************  Revision History  ************************
"
"  Rev. A  01/03/90 - KKP - First Version
"  Rev. B  02/20/90 - KKP - Put BG_ Synchroniziation into PLD.
"
" ***************************************************************

end ARB
```

292076-37

```
Module CAPORT  FLAG '-R3';
Title '82596CA Channel Attention and Port   Rev. A   01/03/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';

" ************************   Descrption   ************************
"
"  FOR:  82596CA / 68030 Interface
"
"  This PLD decodes the 68030 address lines and generates the
"  Channel Attention and PORT_ signals to the 82596.  The choice
"  of address is of address is left to the system designer.
"
"  Nine address decode lines are available.  They could be
"  connected to A31-A23.  NC1, NC2, NC3, and NC4 are
"  combinatorial outputs.  They can be used as extra address
"  inputs.  NC5 is a standard input that is also available as
"  an extra address term.  If even more decode lines are required,
"  then the HIADDR input is for the output of the external
"  decoder.  This decode must be done in less than 16 ns.
"
"  In the line ADDR = [A09,A08,....], the values for A09-A01
"  should be set high or low (inverted) for the desired range.
"  The decode values for CA_ACC and PORT_ACC (110 and 220) are
"  arbitrary and can be modified as needed.
"
"  S0_ and S1_ are state bits used for generating wait states for
"  PORT_ assertion.
"
"  UNUSED INPUT  PINS                  :  1
"  UNUSED OUTPUT PINS (REGISTERED)     :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL)  :  4
"
" ****************************************************************

caport Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
AS_       Pin   2; "I"          HIADDR    Pin 23; "I"
A01       Pin   3; "I"          NC1       Pin 22; "I/O"
A02       Pin   4; "I"          NC2       Pin 21; "I/O"
A03       Pin   5; "I"          CA        Pin 20; "R,I/O"
A04       Pin   6; "I"          S0_       Pin 19; "R,I/O"
A05       Pin   7; "I"          S1_       Pin 18; "R,I/O"
A06       Pin   8; "I"          PORT_     Pin 17; "R,I/O"
A07       Pin   9; "I"          NC3       Pin 16; "I/O"
A08       Pin  10; "I"          NC4       Pin 15; "I/O"
A09       Pin  11; "I"          NC5       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"
```

292076-38

```
"Declarations

   X,C    = .X.,.C.;

   ADDR   = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];   " User defined address.

   CA_ACC    MACRO  ( (ADDR == ^h110) & HIADDR & !AS_ );

   PORT_ACC  MACRO  ( (ADDR == ^h220) & HIADDR & !AS_ );

MODE      = [CA,PORT_,S0_,S1_];

IDLE      = [ 1,  1,  1 , 1 ];
PORT_SET  = [ 1,  0,  1 , 1 ];  " Set PORT_ to 82596.
PORT_HLD1 = [ 1,  0,  0 , 1 ];  " Hold for one clock state.
PORT_HLD2 = [ 1,  0,  0 , 0 ];  " Hold for a second clock state.
ACCESS_OFF= [ 1,  1,  0 , 0 ];  " Deassert PORT_ and CA.
CA_SET    = [ 0,  1,  1 , 1 ];  " Set CA to 82596.
CA_HLD1   = [ 0,  1,  0 , 1 ];  " Hold for one clock state.
CA_HLD2   = [ 0,  1,  0 , 0 ];  " Hold for a second clock state.


Equations

State_Diagram IN caport MODE

state IDLE       : CASE  (PORT_ACC)                :PORT_SET;
                         (CA_ACC)                  :CA_SET;
                         (!(PORT_ACC # CA_ACC))    :IDLE;
                   ENDCASE;

state PORT_SET   : GOTO PORT_HLD1;

state PORT_HLD1  : GOTO PORT_HLD2;

state PORT_HLD2  : GOTO ACCESS_OFF;

state ACCESS_OFF : IF AS_ THEN IDLE
                          ELSE ACCESS_OFF;

state CA_SET     : GOTO CA_HLD1;

state CA_HLD1    : GOTO CA_HLD2;

state CA_HLD2    : GOTO ACCESS_OFF;


" ***************  Revision History  ***************************
"
"  Rev. A    1/3/90 - KKP - First Version.
"
" ************************************************************

end CAPORT
```

292076-39

```
Module CNVRT  FLAG '-R3';
Title '82596CA Signal Conversion          Rev. A    01/03/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-10';

" ************************   Descrption   ************************
"
"  FOR:  82596CA / 68030 Interface
"
"  This PLD converts the 82596 signals to 68030 type signals.
"
"  DELAYAS_ is generated in the RDY PLD to delay AS_ until it is
"  known whether a multiple or burst transfer is to take place.
"
"  A PLD 20R4 was used in this example, requiring seperate
"  output enables (LANCYC_ and LANCYC2_, connected external to
"  the PLD) for the registered and latched outputs.
"
"  NEWRW_, NEWAS_, NEWDS_, and NEWDBEN_ are registered outputs.
"  NEWSIZ0, NEWSIZ1, NEWA0, and NEWA1 are combinatorial outputs.
"  All of these signals will be enabled when the 82596 has
"  control of the local bus, otherwise they will be tri-stated.
"
"  The combinatorial outputs are generated using a truth table.
"  For completeness, default settings are included for the
"  impossible BE# input combinations.
"
"  UNUSED INPUT  PINS                  :  3
"  UNUSED OUTPUT PINS (REGISTERED)     :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL)  :  0
"
" ***************************************************************

cnvrt Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
NC1       Pin   2; "I"          RDY_      Pin 23; "I"
DELAYAS_  Pin   3; "I"          NEWA0     Pin 22; "I/O"
BE0_      Pin   4; "I"          NEWSIZ0   Pin 21; "I/O"
BE1_      Pin   5; "I"          NEWRW_    Pin 20; "R,I/O"
BE2_      Pin   6; "I"          NEWAS_    Pin 19; "R,I/O"
BE3_      Pin   7; "I"          NEWDS_    Pin 18; "R,I/O"
WR_       Pin   8; "I"          NEWDBEN_  Pin 17; "R,I/O"
NC2       Pin   9; "I"          NEWSIZ1   Pin 16; "I/O"
NC3       Pin  10; "I"          NEWA1     Pin 15; "I/O"
NEWCBRQ_  Pin  11; "I"          LANCYC2_  Pin 14; "I"
GND       Pin  12;              LANCYC_   Pin 13; "I"
```

292076-40

```
Equations

ENABLE NEWAS_    = !LANCYC_;
ENABLE NEWDS_    = !LANCYC_;
ENABLE NEWDBEN_  = !LANCYC_;
ENABLE NEWRW_    = !LANCYC_;
ENABLE NEWSIZ0   = !LANCYC2_;
ENABLE NEWSIZ1   = !LANCYC2_;
ENABLE NEWA0     = !LANCYC2_;
ENABLE NEWA1     = !LANCYC2_;

!NEWAS_    :=

    !DELAYAS_                  " Start after BLAST_ valid.
  # !NEWAS_ & !NEWCBRQ_;       " Hold through multiple/burst transfer.

!NEWDS_    :=

    !WR_ & !DELAYAS_           " Start DS_ with AS_ during read cycle.
  #  WR_ & !NEWDBEN_ & RDY_    " Delay DS_ by 1 clock during a write cycle.
  # !NEWDS_ & !NEWCBRQ_;       " Hold until clock following RDY_ set.

!NEWDBEN_  :=

    !DELAYAS_                    " Enable data transceivers as soon as 82596
                                 "    begins its cycle.
  # !WR_ & !NEWDBEN_ & RDY_      " Hold as long as AS_ during read.
  #  WR_ & !NEWDBEN_ & !NEWAS_;  " Longer data hold during a write.

!NEWRW_    :=  WR_;              "invert WR to match processor

" The following truth table converts the byte enable signals from
" the 82596 into the 68030 SIZ signals and address lines A0 and
" A1.

Truth_Table

( [BE3_,BE2_,BE1_,BE0_,LANCYC2_] -> [NEWSIZ1,NEWSIZ0,NEWA1,NEWA0] )

  [ X , X , X , X ,   1    ] -> [   1   ,    1   ,  1  ,  1  ];
  [ 1 , 1 , 1 , 1 ,   0    ] -> [   1   ,    1   ,  1  ,  1  ];
  [ 1 , 1 , 1 , 0 ,   0    ] -> [   0   ,    1   ,  1  ,  1  ];
  [ 1 , 1 , 0 , 0 ,   0    ] -> [   1   ,    0   ,  1  ,  0  ];
  [ 1 , 0 , 0 , 0 ,   0    ] -> [   1   ,    1   ,  0  ,  1  ];
  [ 0 , 0 , 0 , 0 ,   0    ] -> [   0   ,    0   ,  0  ,  0  ];
  [ 1 , 1 , 0 , 1 ,   0    ] -> [   0   ,    1   ,  1  ,  0  ];
  [ 1 , 0 , 0 , 1 ,   0    ] -> [   1   ,    0   ,  0  ,  1  ];
  [ 0 , 0 , 0 , 1 ,   0    ] -> [   1   ,    1   ,  0  ,  0  ];
  [ 1 , 0 , 1 , 1 ,   0    ] -> [   0   ,    1   ,  0  ,  1  ];
  [ 0 , 0 , 1 , 1 ,   0    ] -> [   1   ,    0   ,  0  ,  0  ];
  [ 0 , 1 , 1 , 1 ,   0    ] -> [   0   ,    1   ,  0  ,  0  ];
```

292076-41

```
"  The following BE# input combinations are illegal and will
"  result in erroneous data transfers.

  [ X  , 0  , 1  , 0  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];
  [ 0  , 1  , 0  , X  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];
  [ 0  , 1  , 1  , 0  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];

" ****************  Revision History  ***************************
"
"  Rev. A    1/3/90 - KKP - First Version.
"
" ****************************************************************

end CNVRT
```

292076-42

```
Module RDY  FLAG '-R3';
Title '82596CA Ready and Burst Cycle Logic  Rev. A     01/03/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-10';

" ************************   Descrption   ************************
"
"  This PLD generates the RDY_ and BRDY_ signals to the 82596.
"  It also generates the Burst Request (CBREQ_) signal to the
"  memory controller.  It uses the 68030 signals Address Strobe
"  (AS_), Data Strobe (DS_), Data Bus Enable (DBEN_), Data
"  Acknowledge (DSACK_) and Synchronous Termination (STERM_).  It
"  also uses Cache Burst Acknowledge (CBACK_) from the memory
"  controller and Burst Last (BLAST_) from the 82596.  The
"  DELAYAS_ signal is used to delay the generation of AS_ to the
"  memory controller in order to determine whether a burst transfer
"  is about to take place.  Because CLK1_ is needed for both the
"  flip-flop registers and in the combinatorial equations, it
"  is connected to both pins 1 and 2.  Two separate names are
"  required in the equations (CLK1_ and CCLK1_).
"
"  The first three burst data transfers between the 82596 and the
"  memory will be acknowledged with the BRDY_ signal.  The last
"  (fourth) burst data transfer cycle will be acknowledged with a
"  RDY_.
"
"  This PLD must be 10 ns or faster to meet BRDY set-up to CLK1_
"  on 82596.
"
"  The output DELAYRDY_ is only used inside this PLD to generate
"  a delay for the RDY_ signal to the 82596.
"
"  UNUSED INPUT  PINS                  :  2
"  UNUSED OUTPUT PINS (REGISTERED)     :  1
"  UNUSED OUTPUT PINS (COMBINATORIAL)  :  1
"
" ***************************************************************

rdy Device 'P20R4';

CLK1_     Pin   1; "I"          VCC        Pin 24;
CCLK1_    Pin   2; "I"          NC2        Pin 23; "I"
AS_       Pin   3; "I"          RDY_       Pin 22; "I/O"
DS_       Pin   4; "I"          BRDY_      Pin 21; "I/O"
STERM_    Pin   5; "I"          DELAYAS_   Pin 20; "R,I/O"
DBEN_     Pin   6; "I"          S0         Pin 19; "R,I/O"
ADS_      Pin   7; "I"          NEWCBREQ_  Pin 18; "R,I/O"
CBACK_    Pin   8; "I"          NC3        Pin 17; "R,I/O"
NC1       Pin   9; "I"          DELAYRDY_  Pin 16; "I/O"
DSACK_    Pin  10; "I"          NC4        Pin 15; "I/O"
BLAST_    Pin  11; "I"          LANCYC_    Pin 14; "I"
GND       Pin  12;              OE_        Pin 13; "I"
```

292076-43

```
"Declarations

MODE      = [DELAYAS_,S0,NEWCBREQ_];

IDLE      = [1,1,1];
BLST_WT   = [0,1,1];  " Wait for BLAST to determine if burst data transfer.
NO_BURST  = [1,0,1];  " BLAST_ active so no burst transfer.
BRST_CYC  = [0,0,1];  " BLAST_ not active so multiple or burst transfer.
STERM_1   = [0,0,0];  " Wait for acknowledge.
STERM_2   = [0,1,0];  " Wait for acknowledge.
STERM_3   = [1,1,0];  " Wait for acknowledge.

Equations

!RDY_    = !BLAST_ & !DELAYRDY_ & CCLK1_ & !AS_ & CBACK_
             " Return RDY_ whenever BLAST_ asserted.
         # !STERM_ & NEWCBREQ_
             " Fourth burst transfer or synchronous transfer.
         # !RDY_ & !DBEN_;
             " Hold RDY_ until data requirement met.

!BRDY_   = !STERM_ & !NEWCBREQ_ & !CBACK_
             " Assert BRDY_ during burst cycles
         # !BRDY_ & !CCLK1_;
             " Hold so recognized on rising edge of CLK1_ to 82596.

!DELAYRDY_ =

     !DS_ & !LANCYC_ & !CCLK1_    " Delay RDY for data setup.
   # !DELAYRDY_ & !AS_;           " Hold until end of data cycle.


State_Diagram IN rdy MODE

state IDLE      : IF !ADS_ THEN BLST_WT
                           ELSE IDLE;

state BLST_WT   : IF !BLAST_ THEN NO_BURST
                             ELSE BRST_CYC;

state NO_BURST : GOTO IDLE;

state BRST_CYC : CASE (!BRDY_ & !CBACK_)              : STERM_1;
                      ((!BRDY_ & CBACK_) # !BLAST_)   : IDLE;
                      (BRDY_ & BLAST_)                : BRST_CYC;
                 ENDCASE;

state STERM_1   : CASE (!BRDY_ & !CBACK_)             : STERM_2;
                      ((!BRDY_ & CBACK_) # !BLAST_)   : IDLE;
                      (BRDY_ & BLAST_)                : STERM_1;
                 ENDCASE;
```

292076-44

```
state STERM_2  : CASE (!BRDY_ & !CBACK_)              : STERM_3;
                      ((!BRDY_ & CBACK_) # !BLAST_)   : IDLE;
                      (BRDY_ & BLAST_)                : STERM_2;
                 ENDCASE;

state STERM_3  : IF (!BRDY_ # !BLAST_) THEN IDLE
                                       ELSE STERM_3;

" ***************  Revision History  ***************************
"
"  Rev. A    1/20/90 - KKP - First Version.
"
" ********************************************************************

end RDY
```

292076-45

1

B.2 MC68020/82596DX

```
Module ARB  FLAG '-R3';
Title '82596DX Arbitration for Local Bus    Rev. A     01/12/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R6-10';

" ************************   Descrption   ************************
"
"  FOR:  82596DX / 68020 Interface
"
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus.  Refresh requests are
"  given highest priority, and the 82596DX requests given second
"  highest priority.  The CPU normally controls the local bus
"  when no requests are pending.  The RDY_ acknowledge signal to
"  the 82596DX is also generated in this PLD.  The signal
"  DELAYRDY_ is an embedded signal used only in this PLD to
"  generate RDY_.
"
"  Requestors are granted the bus by using the inverted CPU clock
"  (CLK1).  Because is required for both the registered and
"  combinatorial terms, CLK1 is connected to both the clock and a
"  combinatorial input.  The combinatorial term is called CCLK1.
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to CLK1.  If this
"  set-up cannot be guaranteed, the request must be synchronized
"  through an external flip-flop, clocked with CLK1.
"
"  The SBG_ signal is the synchronized 68020 Bus Grant (BG_)
"  signal.
"
"  Because the 82596 uses and active-high HOLD, LANCYC_ is
"  inverted with an external 74F04.
"
"  The equations and marco-cells are allocated for the Bus
"  Request (BREQ) signal, which is used to activiate the 82596DX
"  Bus Throttle Timers.  In these equations it is set inactive.
"  It is left to the system designer to define input conditions
"  for this signal.
"
"  UNUSED INPUT  PINS                    : 4
"  UNUSED OUTPUT PINS (COMBINATORIAL)    : 0
"  UNUSED OUTPUT PINS (REGISTERED)       : 0
"
" ***************************************************************
```

292076-46

```
arb Device 'P20R6';

CLK1      Pin   1; "I"          VCC       Pin 24;
CCLK1     Pin   2; "I"          NC3       Pin 23; "I"
AS_       Pin   3; "I"          DELAYRDY_ Pin 22; "I/O"
HOLD      Pin   4; "I"          BREQ      Pin 21; "R,I/O"
LOCK      Pin   5; "I"          LANCYC_   Pin 20; "R,I/O"
DS_       Pin   6; "I"          BR_       Pin 19; "R,I/O"
BG_       Pin   7; "I"          BGACK_    Pin 18; "R,I/O"
DBEN_     Pin   8; "I"          RFCYC_    Pin 17; "R,I/O"
RFRQ_     Pin   9; "I"          SBG_      Pin 16; "R,I/O"
NC1       Pin  10; "I"          RDY_      Pin 15; "I/O"
NC2       Pin  11; "I"          NC4       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"

MODE    = [BR_,BGACK_,RFCYC_,LANCYC_];

IDLE    = [1,1,1,1];
REQ     = [0,1,1,1];     " Generic request to CPU for local bus.
RF_CYC  = [1,0,0,1];     " Refresh request has been granted.
LAN_CYC = [1,0,1,0];     " LAN request has been granted.
PRE_CHG = [1,0,1,1];     " Required for back-to-back cycles.


Equations

BREQ  := 0;                 " Bus Throttle conditions will need
                            " to be defined by the system designer.

!SBG_  = !BG_ & CCLK1       " Set during high phase of ARB clock to
                            "    meet setup time into PLD.
       # !SBG_ & !BG_       " Hold with processor bus grant.
       # !SBG_ & !CCLK1;    " Hold through low phase of clock to meet setup time.

!DELAYRDY_ =

     !DS_ & !LANCYC_ & !CCLK1     " Delay RDY_ for data set-up.
   # !DELAYRDY_ & !AS_;           " Hold until end of data cycle.

!RDY_  =

     !DELAYRDY_ & CCLK1 & !AS_    " Return RDY_ after delay while
                                  "  data cycle still in progress.
   # !RDY_ & !DBEN_;              " Hold until end of data cycle.
```

292076-47

```
State_Diagram IN arb MODE

state IDLE      : IF (!RFRQ_ # HOLD) THEN REQ
                                     ELSE IDLE;

state REQ       : CASE (!RFRQ_ & !SBG_)                    :RF_CYC;
                       ( HOLD & RFRQ_)                     :LAN_CYC;
                       (!((!RFRQ_ & !SBG_)
                        # (HOLD & RFRQ_ & SBG_)))          :REQ;
                  ENDCASE;

state RF_CYC    : CASE (RFRQ_ & !HOLD)                     :IDLE;
                       (RFRQ_ & HOLD)                      :PRE_CHG;
                       (!RFRQ_)                            :RF_CYC;
                  ENDCASE;

state LAN_CYC   : CASE (!HOLD & RFRQ_ & !LOCK)              :IDLE;
                       (!HOLD & !RFRQ_ & !LOCK)            :PRE_CHG;
                       (HOLD)                              :LAN_CYC;
                  ENDCASE;

state PRE_CHG   : CASE (RFRQ_ & !HOLD & !LOCK)              :IDLE;
                       (RFRQ_ & HOLD)                      :LAN_CYC;
                       (!RFRQ_ & !LOCK)                    :RF_CYC;
                       (!SBG_ & RFRQ_ & !HOLD & !LOCK)     :PRE_CHG;
                  ENDCASE;

" ******************  Revision History  ************************
"
"  Rev. A  01/12/90 - KKP - First Version
"  Rev. B  02/20/90 - KKP - Moved BG_ synchronization into PLD.
"
" ***************************************************************

end ARB
```

292076-48

```
Module CAPORT  FLAG '-R3';
Title '82596DX Channel Attention and Port   Rev. A   01/12/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';

" ************************   Descrption   ************************
"
"  FOR:    82596DX / 68020 Interface
"
"  This PLD decodes the 68020 address lines and generates the
"  Channel Attention and PORT_ to the 82596DX.  The choice of
"  address is left to the system designer.
"
"  Nine address decode lines are available.  They could be
"  connected to A31-A23.  NC1, NC2, NC3, and NC4 are
"  combinatorial outputs.  They can be used as extra address
"  inputs.  NC5 is a standard input that is also available as
"  an extra address term.  If even more decode lines are required,
"  then the HIADDR input is for the output of the external
"  decoder.  This decode must be done in less than 22 ns.
"
"  In the line ADDR = [A09,A08,....], the values for A09-A01
"  should be set high or low (inverted) for the desired range.
"  The decode values for CA_ACC and PORT_ACC (110 and 220) are
"  arbitrary and can be modified as needed.
"
"  S0_ AND S1_ are state bits used for generating wait states for
"  PORT_ assertion.
"
"
"  UNUSED INPUT  PINS                 :  1
"  UNUSED OUTPUT PINS (REGISTERED)    :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL) :  4
""
" ********************************************************************

caport Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
AS_       Pin   2; "I"          HIADDR    Pin 23; "I"
A01       Pin   3; "I"          NC1       Pin 22; "I/O"
A02       Pin   4; "I"          NC2       Pin 21; "I/O"
A03       Pin   5; "I"          CA        Pin 20; "R,I/O"
A04       Pin   6; "I"          S0_       Pin 19; "R,I/O"
A05       Pin   7; "I"          S1_       Pin 18; "R,I/O"
A06       Pin   8; "I"          PORT_     Pin 17; "R,I/O"
A07       Pin   9; "I"          NC3       Pin 16; "I/O"
A08       Pin  10; "I"          NC4       Pin 15; "I/O"
A09       Pin  11; "I"          NC5       Pin 14; "I"
GND       Pin  12;              OE        Pin 13; "I"
```

292076-49

```
" Declarations

   X,C    = .X.,.C.;

   ADDR   = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];   " User defined address.

   CA_ACC    MACRO  { (ADDR == ^h110) & HIADDR & !AS_ };

   PORT_ACC  MACRO  { (ADDR == ^h220) & HIADDR & !AS_ };
```

292076-50

```
MODE       = [CA,PORT_,S0_,S1_];

IDLE       = [ 1,  1,  1 , 1 ];
PORT_SET   = [ 1,  0,  1 , 1 ];  " Set PORT_ to 82596DX.
PORT_HLD1  = [ 1,  0,  0 , 1 ];  " Hold for one clock state.
PORT_HLD2  = [ 1,  0,  0 , 0 ];  " Hold for a second clock state.
ACCESS_OFF = [ 1,  1,  0 , 0 ];  " Deassert PORT_ and CA.
CA_SET     = [ 0,  1,  1 , 1 ];  " Set CA to 82596DX.
CA_HLD1    = [ 0,  1,  0 , 1 ];  " Hold for one clock state.
CA_HLD2    = [ 0,  1,  0 , 0 ];  " Hold for a second clock state.


Equations

State_Diagram IN caport MODE

state IDLE       : CASE  (PORT_ACC)                :PORT_SET;
                         (CA_ACC)                  :CA_SET;
                         (!(PORT_ACC # CA_ACC))    :IDLE;
                   ENDCASE;

state PORT_SET   : GOTO PORT_HLD1;

state PORT_HLD1  : GOTO PORT_HLD2;

state PORT_HLD2  : GOTO ACCESS_OFF;

state ACCESS_OFF : IF AS_ THEN IDLE
                          ELSE ACCESS_OFF;

state CA_SET     : GOTO CA_HLD1;

state CA_HLD1    : GOTO CA_HLD2;

state CA_HLD2    : GOTO ACCESS_OFF;


" ****************  Revision History  ***************************
"
"  Rev. A    1/3/90 - KKP - First Version.
"
" ***************************************************************

end CAPORT
```

292076-51

```
Module CNVRT  FLAG '-R3';
Title '82596DX Signal Conversion          Rev. A     1/12/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-10';

" ************************   Descrption   ************************
"
"  FOR:   82596DX / 68020 Interface
"
"  This PLD converts the 82596DX signals to 68020 type signals.
"
"  These signals will be enabled when the 82596DX has control of
"  the local bus (LANCYC_ is low), otherwise they will be
"  tri-stated.
"
"  A PLD 20R4 was used in this example, requiring seperate
"  enables, LANCYC_ and LANCYC2_, which are the same signal
"  external to the PLD.  If the PLD does not require separate
"  output enables for registered and latched outputs then this is
"  not required.
"
"  NEWRW_, NEWAS_, NEWDS_, and NEWDBEN_ are registered outputs.
"  NEWSIZ0, NEWSIZ1, NEWA0, and NEWA1 are combinatorial outputs.
"
"  The combinatorial outputs are generated using a truth table.
"  For completeness, default settings are included for the
"  impossible BE# input combinations.
"
"  UNUSED INPUT  PINS                  : 4
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 0
"  UNUSED OUTPUT PINS (REGISTERED)     : 0
"
" ****************************************************************

cnvrt Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
NC1       Pin   2; "I"          RDY_      Pin 23; "I"
ADS_      Pin   3; "I"          NEWA0     Pin 22; "I/O"
BE0_      Pin   4; "I"          NEWSIZ0   Pin 21; "I/O"
BE1_      Pin   5; "I"          NEWRW_    Pin 20; "R,I/O"
BE2_      Pin   6; "I"          NEWAS_    Pin 19; "R,I/O"
BE3_      Pin   7; "I"          NEWDS_    Pin 18; "R,I/O"
WR_       Pin   8; "I"          NEWDBEN_  Pin 17; "R,I/O"
NC2       Pin   9; "I"          NEWSIZ1   Pin 16; "I/O"
NC3       Pin  10; "I"          NEWA1     Pin 15; "I/O"
NC4       Pin  11; "I"          LANCYC2_  Pin 14; "I"
GND       Pin  12;              LANCYC_   Pin 13; "I"


"Declarations

     X = .X.;
```

292076-52

```
Equations

ENABLE NEWAS_    = !LANCYC_;
ENABLE NEWDS_    = !LANCYC_;
ENABLE NEWDBEN_  = !LANCYC_;
ENABLE NEWRW_    = !LANCYC_;
ENABLE NEWSIZ0   = !LANCYC2_;
ENABLE NEWSIZ1   = !LANCYC2_;
ENABLE NEWA0     = !LANCYC2_;
ENABLE NEWA1     = !LANCYC2_;

!NEWAS_ :=

     !ADS_                  " Start AS_ during 68020 clock low cycle.
   # !NEWAS_ & RDY_;        " Hold until clock following RDY_ set.

!NEWDS_ :=

     !WR_ & !ADS_               " Start DS_ with AS_ during read.
   #  WR_ & !NEWDBEN_ & RDY_   " Delay DS_ by 1 clock during write.
   # !NEWDS_ & RDY_;           " Hold until clock following RDY_ set.

!NEWDBEN_ :=

     !ADS_                          " Enable data transceivers as soon
                                    " as 82596DX begins its cycle.
   # !WR_ & !NEWDBEN_ & RDY_        " Hold as long as AS_ during read.
   #  WR_ & !NEWDBEN_ & !NEWAS_;    " Longer data hold during a write.

!NEWRW_   :=  WR_;                  " Invert W/R_ to match processor.


" The following truth table converts the byte enable signals from
" the 82596DX into the 68020 SIZE signals and address lines A0
"  and A1.


Truth_Table

( [BE3_,BE2_,BE1_,BE0_,LANCYC2_] -> [NEWSIZ1,NEWSIZ0,NEWA1,NEWA0] )

  [ X , X , X , X ,  1  ] -> [  1  ,  1 , 1 , 1 ];
  [ 1 , 1 , 1 , 1 ,  0  ] -> [  1  ,  1 , 1 , 1 ];
  [ 1 , 1 , 1 , 0 ,  0  ] -> [  0  ,  1 , 1 , 1 ];
  [ 1 , 1 , 0 , 0 ,  0  ] -> [  1  ,  0 , 1 , 0 ];
  [ 1 , 0 , 0 , 0 ,  0  ] -> [  1  ,  1 , 0 , 1 ];
  [ 0 , 0 , 0 , 0 ,  0  ] -> [  0  ,  0 , 0 , 0 ];
  [ 1 , 1 , 0 , 1 ,  0  ] -> [  0  ,  1 , 1 , 0 ];
  [ 1 , 0 , 0 , 1 ,  0  ] -> [  1  ,  0 , 0 , 1 ];
  [ 0 , 0 , 0 , 1 ,  0  ] -> [  1  ,  1 , 0 , 0 ];
  [ 1 , 0 , 1 , 1 ,  0  ] -> [  0  ,  1 , 0 , 1 ];
  [ 0 , 0 , 1 , 1 ,  0  ] -> [  1  ,  0 , 0 , 0 ];
  [ 0 , 1 , 1 , 1 ,  0  ] -> [  0  ,  1 , 0 , 0 ];
```

292076-53

```
"  The following BE# input combinations are illegal and will
"  result in erroneous data transfers.

  [ X  , 0  , 1  , 0  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];
  [ 0  , 1  , 0  , X  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];
  [ 0  , 1  , 1  , 0  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];

" ***************** Revision History ****************************
"
"  Rev. A    01/12/90 - KKP - First Version.
"
" ***************************************************************

end CNVRT
```

292076-54

B.3 MC68000/82596SX

```
Module ARB  FLAG '-R3';
Title '82596SX Arbitration for Local Bus   Rev. A   01/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R8-15';

" ************************   Descrption   ************************
"
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus.  Refresh requests are
"  given highest priority, and the 82596SX requests given second
"  highest priority.  The CPU normally controls the local bus
"  when no requests are pending.
"
"  Requestors are granted the bus by using the 82596SX clock,
"  CLK1.
"
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to the clock (CLK1).
"  If this set-up cannot be guaranteed, the request must be
"  synchronized through an external flip-flop, clocked with CLK1.
"
"  The SBG_ signal is the synchronized 68000 Bus Grant (BG__)
"  signal.  It is be synchronized internally using a flip-flop
"  clocked with CLK1.  Because the 82596SX uses an active high
"  HLDA, LANCYC_ is inverted using one of the macro-cells.
"
"  The equations and macro-cells are allocated for the Bus
"  Request signal, which is used to activiate the Bus Throttle
"  Timers.  It is left to the system designer to define input
"  conditions for this signal.
"
"  UNUSED INPUT  PINS                  : 6
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 0
"  UNUSED OUTPUT PINS (REGISTERED)     : 1
"
" ********************************************************************

arb Device 'P20R8';

CLK1      Pin   1; "I"          VCC       Pin 24;
RESET     Pin   2; "I"          NC6       Pin 23; "I"
AS_       Pin   3; "I"          BREQ      Pin 22; "R,I/O"
HOLD      Pin   4; "I"          HLDA      Pin 21; "R,I/O"
LOCK      Pin   5; "I"          LANCYC_   Pin 20; "R,I/O"
NC1       Pin   6; "I"          BR_       Pin 19; "R,I/O"
BG_       Pin   7; "I"          SBG_      Pin 18; "R,I/O"
NC2       Pin   8; "I"          BGACK_    Pin 17; "R,I/O"
RFRQ_     Pin   9; "I"          RFCYC_    Pin 16; "R,I/O"
NC3       Pin  10; "I"          NC7       Pin 15; "R,I/O"
NC4       Pin  11; "I"          NC5       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"
```

292076-55

```
MODE    = [BR_,BGACK_,RFCYC_,LANCYC_];

IDLE    = [1,1,1,1];
REQ     = [0,1,1,1];     " Generic request to CPU for local bus.
RF_CYC  = [1,0,0,1];     " Refresh request has been granted.
LAN_CYC = [1,0,1,0];     " LAN request has been granted.
PRE_CHG = [1,0,1,1];     " Required for back-to-back cycles.


Equations

BREQ  := 0;              " Bus Throttle conditions will need to be
                         "  defined by the system designer.

HLDA  := !LANCYC_;       " 82596SX requires active-high HLDA.

SBG_  := BG_;            " Synchronized Bus Grant.

MODE  := RESET & IDLE;   " Initialize state machine to IDLE State on reset

State_Diagram IN arb MODE

state IDLE     : IF (!RFRQ_ # HOLD) THEN REQ
                                    ELSE IDLE;

state REQ      : CASE (!RFRQ_ & !SBG_)                  :RF_CYC;
                      (HOLD & RFRQ_)                    :LAN_CYC;
                      (!((!RFRQ_ & !SBG_)
                       # (HOLD & RFRQ_ & SBG_)))        :REQ;
                 ENDCASE;

state RF_CYC   : CASE (RFRQ_ & !HOLD)                   :IDLE;
                      (RFRQ_ & HOLD)                    :PRE_CHG;
                      (!RFRQ_)                          :RF_CYC;
                 ENDCASE;

state LAN_CYC  : CASE (!HOLD & RFRQ_ & !LOCK)           :IDLE;
                      (!HOLD & !RFRQ_ & !LOCK)          :PRE_CHG;
                      (HOLD)                            :LAN_CYC;
                 ENDCASE;

state PRE_CHG  : CASE (RFRQ_ & !HOLD & !LOCK)           :IDLE;
                      (RFRQ_ & HOLD)                    :LAN_CYC;
                      (!RFRQ_ & !LOCK)                  :RF_CYC;
                      (!SBG_ & RFRQ_ & !HOLD & !LOCK)   :PRE_CHG;
                 ENDCASE;

" ********************  Revision History  ************************
"
"  Rev. A  1/20/90 - KKP - First Version
"
" ****************************************************************

end ARB
```

292076-56

```
Module CAPORT  FLAG '-R3';
Title '82596SX Channel Attention and Port   Rev. A   1/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';

" ************************   Descrption   ************************
"
"  FOR:  82596SX / 68000 Interface
"
"  This PLD decodes the 68000 address lines and generates Channel
"  Attention and PORT_ to the 82596.  The choice of address is
"  left to the system designer.
"
"  Nine address decode lines are available.  They could be
"  connected to A23-A14.  NC1 is a combinatorial outputs and
". could be used as extra address input.  If even more decode
"  lines are required, then the HIADDR input is for the output
"  of the external decoder.  This decode must be done in less
"  than 60 ns.
"
"  In the line ADDR = [A09,A08,....], the values for A09-A01
"  should be set high or low (inverted) for the desired range.
"  The decode values for CA_ACC and PORT_ACC (110 and 220) are
"  arbitrary and can be modified as needed.
"
"  S0_ AND S1_ are state bits used for generating wait states for
"  PORT_ assertion.
"
"  UNUSED INPUT  PINS                  : 0
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 1
"  UNUSED OUTPUT PINS (REGISTERED)     : 2
"
"*****************************************************************************

caport Device 'P20R4';

CLK1     Pin   1; "I"          VCC      Pin 24;
CCLK1    Pin   2; "I"          HIADDR   Pin 23; "I"
A01      Pin   3; "I"          NC1      Pin 22; "I/O"
A02      Pin   4; "I"          CA       Pin 21; "I/O"
A03      Pin   5; "I"          NC2      Pin 20; "R,I/O"
A04      Pin   6; "I"          S0_      Pin 19; "R,I/O"
A05      Pin   7; "I"          S1_      Pin 18; "R,I/O"
A06      Pin   8; "I"          NC3      Pin 17; "R,I/O"
A07      Pin   9; "I"          PORT_    Pin 16; "I/O"
A08      Pin  10; "I"          AS_      Pin 15; "I/O"
A09      Pin  11; "I"          LDS_     Pin 14; "I"
GND      Pin  12;              OE_      Pin 13; "I"
```

292076-57

```
"Declarations

  X,C    = .X.,.C.;

  ADDR   = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];    " User defined address.

  CA_ACC    MACRO  ( (ADDR == ^h110) & HIADDR & !AS_ );

  PORT_ACC  MACRO  ( (ADDR == ^h220) & HIADDR & !AS_ );

MODE      = [S0_,S1_];

IDLE      = [ 1,  1 ];
STR_CNT   = [ 0 , 1 ];  " PORT_ or CA has been sent to 82596.
CNT_1     = [ 0,  0 ];  " Hold for one clock state.
CNT_2     = [ 1,  0 ];  " Hold for a second clock state.


Equations

!CA    =

    !LDS_ & !AS_ & CCLK1 & CA_ACC  " Start when data valid on bus.
  # !CA & !(S0_ & !S1_)            " Hold for at least 2 clocks.
  # !CA & CCLK1;                   " Guarantee CA hold time to 82596.

!PORT_ = !LDS_ & !AS_ & CCLK1 & PORT_ACC     " Start when data valid on bus.
       # !PORT_ & !(S0_ & !S1_)              " Hold for at least 2 clocks.
       # !PORT_ & CCLK1;                     " Guarantee PORT_ hold time to 82596.


State_Diagram IN caport MODE

state IDLE      : IF (!PORT_ # CA) THEN STR_CNT
                                   ELSE IDLE;

state STR_CNT   : GOTO CNT_1;

state CNT_1     : GOTO CNT_2;

state CNT_2     : IF AS_ THEN IDLE
                         ELSE CNT_2;


" ***************  Revision History  ***************************
"
"  Rev. A    1/20/90 - KKP - First Version.
"
" ************************************************************

end CAPORT
```

292076-58

```
Module RDY  FLAG '-R3';
Title '82596SX Ready and Signal Conversion  Rev. A   01/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';


" ************************   Descrption   ************************
"
"  This PLD generates the RDY_ signal to the 82596SX.  It also
"  converts the 82596SX signals BHE_, BLE_, ADS_ and WR_ to the
"  68000 equivalents, UDS, LDS, AS_, and RW_ and mimics their
"  timing to the memory controller.
"
"  The output DELAYRDY_ is only used inside this PLD to generate
"  a delay for the RDY_ signal to the 82596SX.
"
"  A 20R4 was used for this example requiring a separate input
"  for the combinatorial enable.
"
"  UNUSED INPUT  PINS                    : 5
"  UNUSED OUTPUT PINS (COMBINATORIAL)    : 0
"  UNUSED OUTPUT PINS (REGISTERED)       : 0
"
"*******************************************************************

rdy Device 'P20R4';

CLK1     Pin   1; "I"          VCC       Pin 24;
CCLK1    Pin   2; "I"          NC5       Pin 23; "I"
AS_      Pin   3; "I"          NEWRW_    Pin 22; "I/O"
BHE_     Pin   4; "I"          NEWAS_    Pin 21; "I/O"
BLE_     Pin   5; "I"          RDY_      Pin 20; "R,I/O"
WR_      Pin   6; "I"          DELAYRDY_ Pin 19; "R,I/O"
NC1      Pin   7; "I"          S0_       Pin 18; "R,I/O"
NC2      Pin   8; "I"          DELAYAS_  Pin 17; "R,I/O"
NC3      Pin   9; "I"          NEWUDS_   Pin 16; "I/O"
ADS_     Pin  10; "I"          NEWLDS_   Pin 15; "I/O"
NC4      Pin  11; "I"          LANCYC2_  Pin 14; "I"
GND      Pin  12;              LANCYC_   Pin 13; "I"

MODE     = [DELAYAS_,DELAYRDY_,RDY_,S0_];

IDLE     = [1,1,1,1];
STR_AS   = [0,1,1,1];  " Delay AS_ until clock phase of 68000 S2.
DLY_RDY  = [0,0,1,1];  " Delay RDY_ by 1 82596 clock state.
DLY_DS   = [0,0,1,0];  " Delay UDS_,LDS_ during write cycle.
STR_RDY  = [0,0,0,1];  " Initiate RDY_ for 68000 type 0 wait cycle.
```

292076-59

1

```
Equations

ENABLE NEWRW_  = !LANCYC2_;
ENABLE NEWUDS_ = !LANCYC2_;
ENABLE NEWLDS_ = !LANCYC2_;
ENABLE NEWAS_  = !LANCYC2_;

!NEWRW_     =  WR_;                    " Invert 82596SX signal
            # !NEWRW_ & !NEWAS_;       " Hold write until AS_ negated

!NEWUDS_    =

      !WR_ & !DELAYAS_ & !BHE_ & !CCLK1 & !AS_      " Start UDS_ with AS_ on read.
    #  WR_ & !S0_ & !BHE_ & !CCLK1 & !AS_           " Delay UDS on a write.
    # !NEWUDS_ & !WR_ & !DELAYAS_                   " Hold thru data cycle.
    # !NEWUDS_ & WR_ & RDY;

!NEWUDS_    =

      !WR_ & !DELAYAS_ & !BLE_ & !CCLK1 & !AS_      " Start LDS_ with AS_ on read.
    #  WR_ & !S0_ & !BLE_ & !CCLK1 & !AS_           " Delay LDS_ on a write.
    # !NEWUDS_ & !WR_ & !DELAYAS_                   " Hold thru data cycle.
    # !NEWUDS_ & WR_ & RDY;

!NEWAS_     = !DELAYAS_ & !CCLK1
            # !NEWAS_ & !WR_ & !DELAYAS_
            # !NEWAS_ & WR_ & RDY;


State_Diagram IN rdy MODE

state IDLE    : IF !ADS_ THEN STR_AS
                         ELSE IDLE;

state STR_AS  : IF !WR_  THEN DLY_RDY
                         ELSE DLY_DS;

state DLY_DS  : GOTO DLY_RDY;

state DLY_RDY : GOTO STR_RDY;

state STR_RDY : GOTO IDLE;

" ***************  Revision History  ***************************
"
"  Rev. A    1/20/90 - KKP - First Version.
" ************************************************************

end RDY
```

292076-60

APPENDIX C
TIMING DIAGRAMS

The following section includes the timing diagram for each specific design. A summary of the timing specifications is also included.

C.1 MC68030/82596CA

- Block Diagram
- MC68030 and 82596CA Clock Synchronization
- MC68030 and 82596CA CA and PORT Access
- MC68030 Local Arbitration (1 page)
- 82596CA Memory Access (2 pages)
- Timing Summary

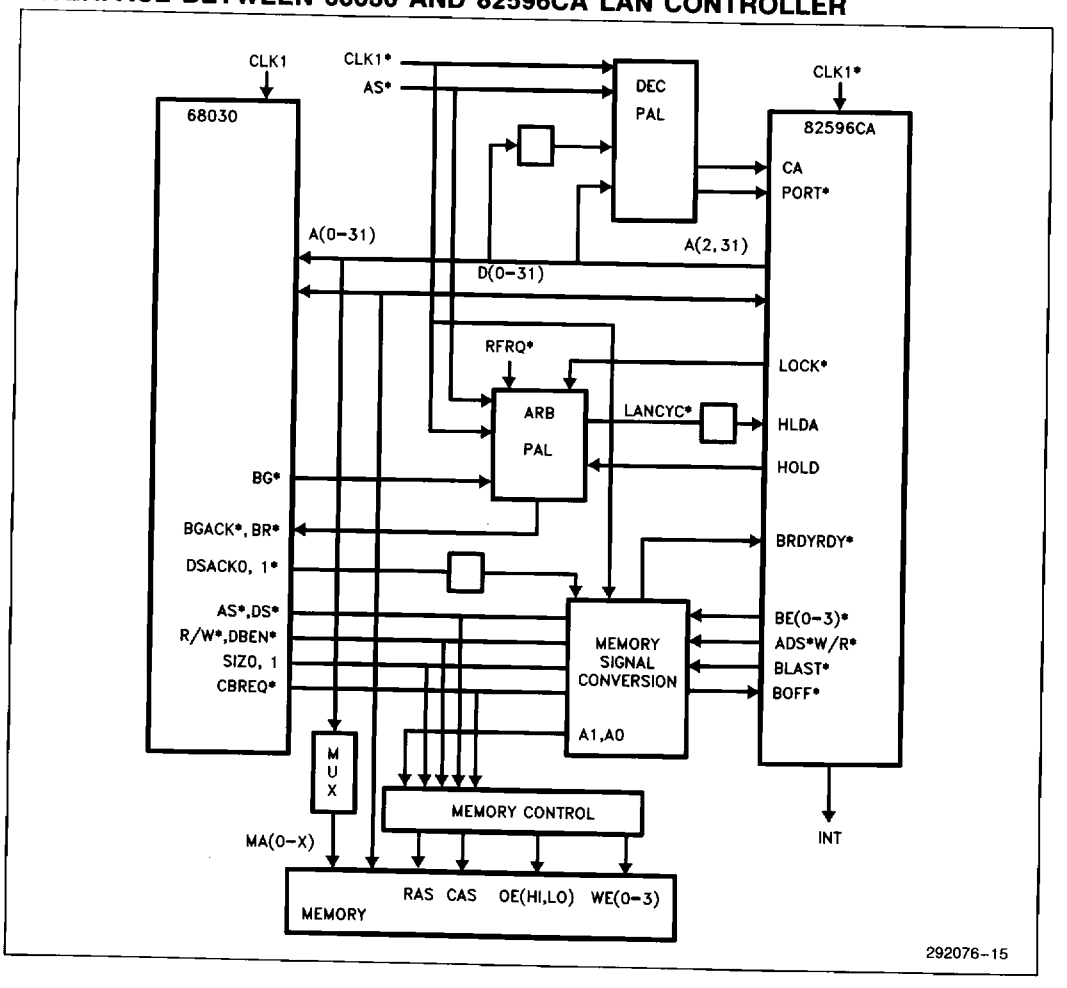
C.2 MC68020/82596DX

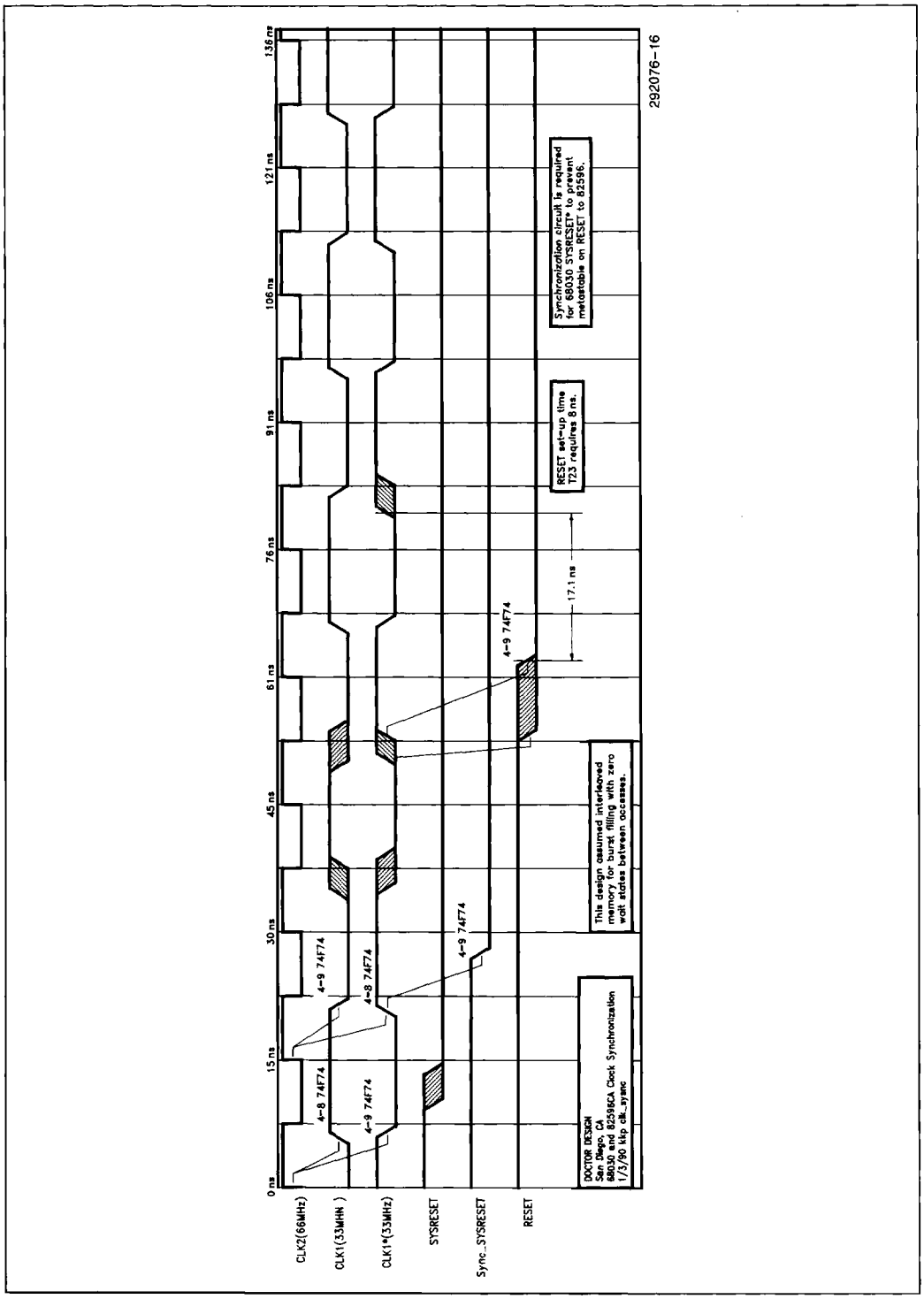
- Block Diagram
- MC68020 and 82596DX Clock Synchronization
- MC68020 and 82596DX CA and PORT Access
- MC68020 Local Arbitration (2 pages)
- 82596DX Memory Access
- Timing Summary

C.3 MC68000/82596SX

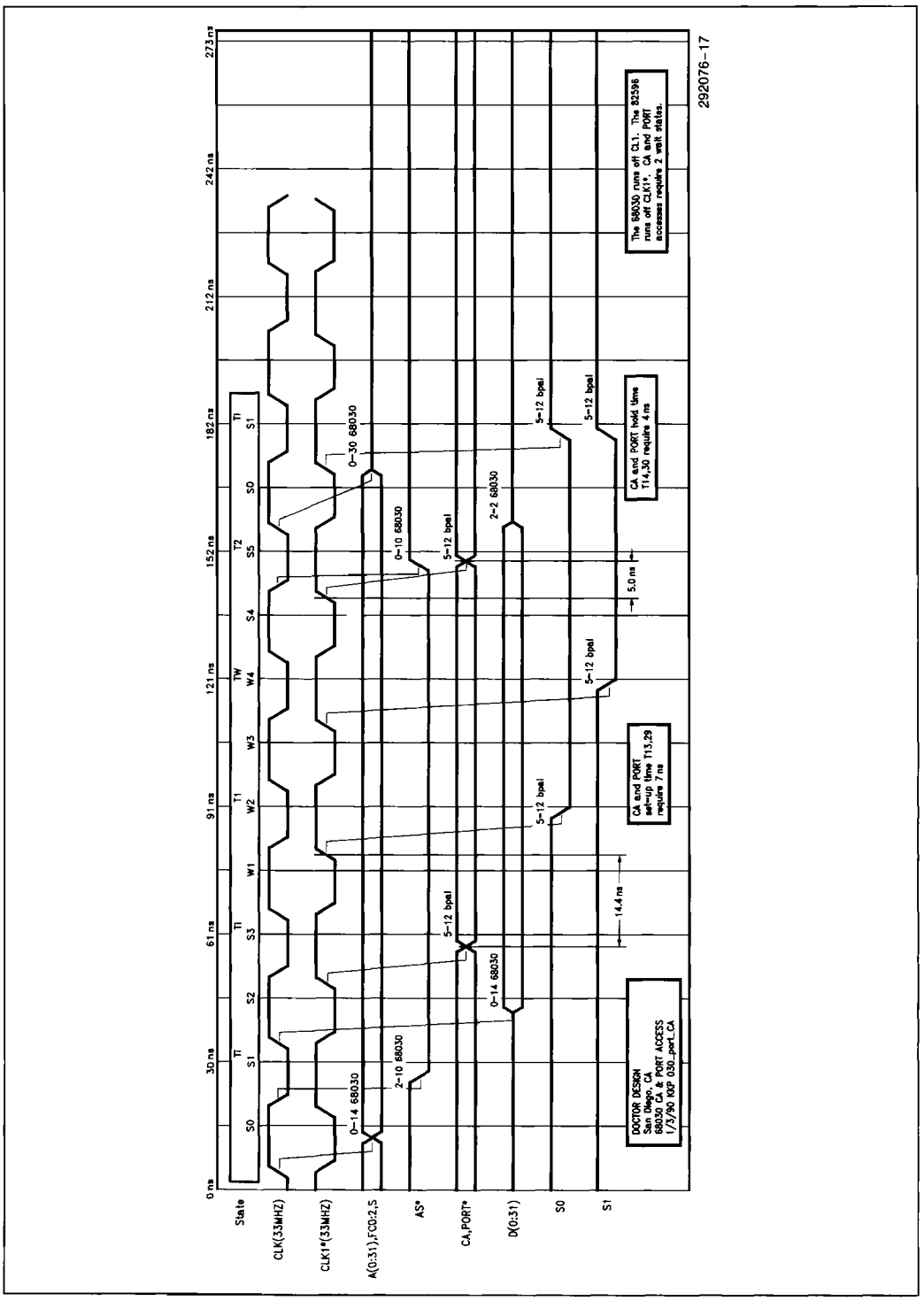
- Block Diagram
- MC68000 and 82596SX Clock Synchronization
- MC68000 and 82596SX CA and PORT Access
- MC68000 Local Arbitration (1 page)
- 82596SX Memory Access (2 pages)
- Timing Summary

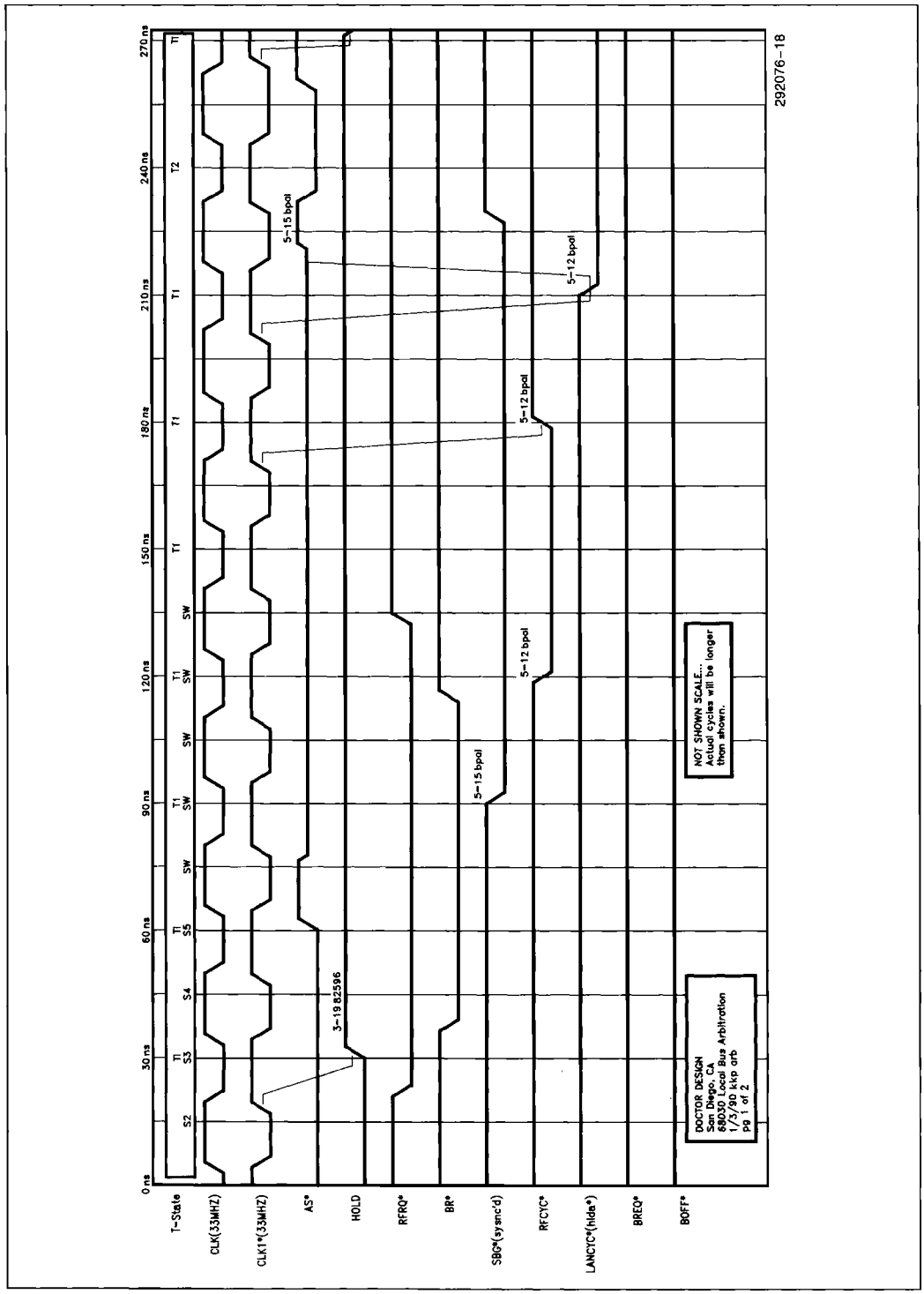
INTERFACE BETWEEN 68030 AND 82596CA LAN CONTROLLER



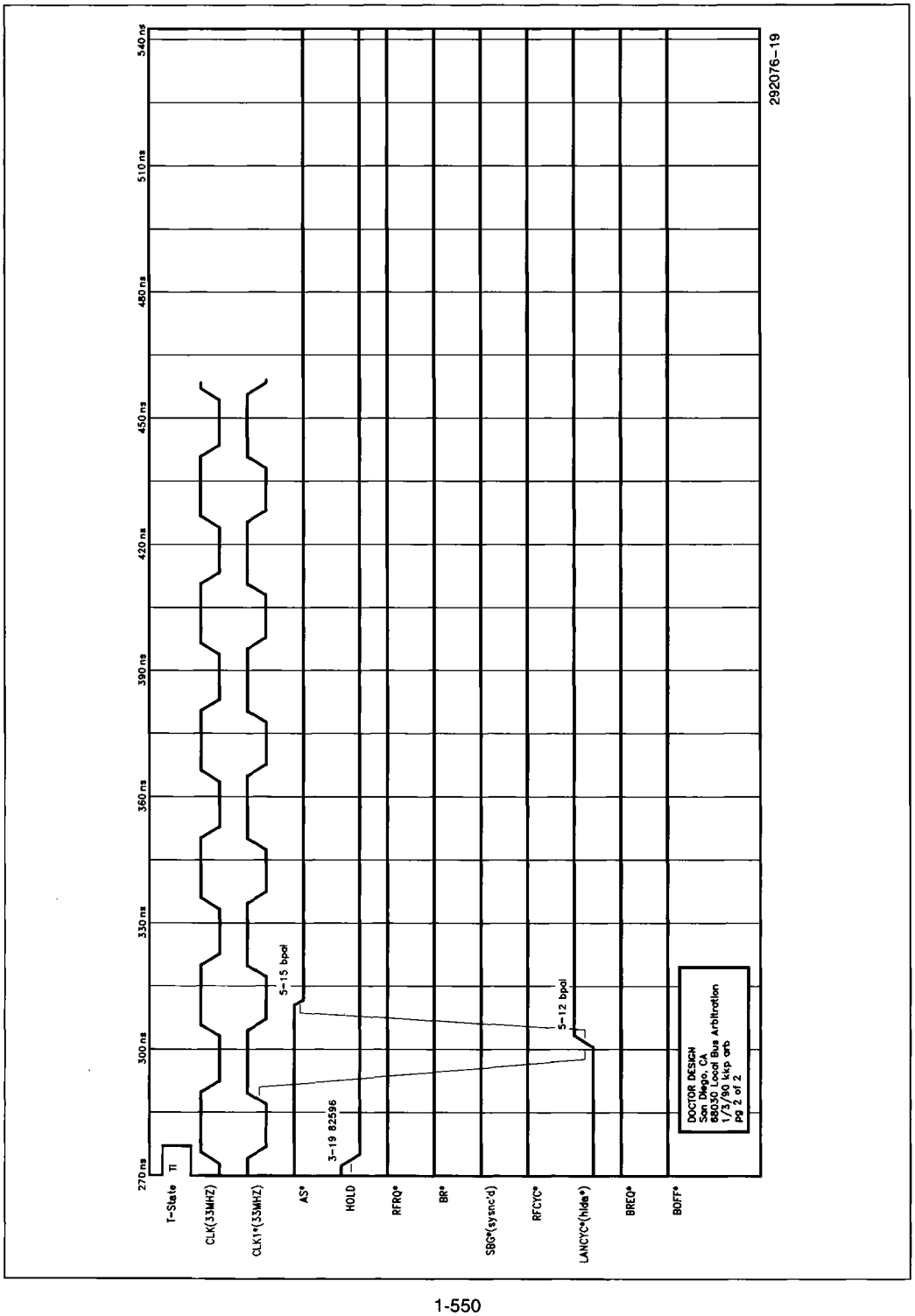

0 ns
15 ns
30 ns
45 ns
61 ns
76 ns
91 ns
106 ns
121 ns
136 ns
CLK2(66MHz)
CLK1(33MHN)
CLK1*(33MHz)
SYSRESET
Sync_SYSRESET
RESET
4-8 74F74
4-9 74F74
4-9 74F74
4-8 74F74
4-9 74F74
4-9 74F74
17.1 ns
RESET set-up time T23 requires 8 ns.
Synchronization circuit is required for 80030 SYSRESET* to prevent metastable on RESET to 82596.
This design assumed interleaved memory for burst filling with zero wait states between accesses.
DOCTOR DESIGN
San Diego, CA
80030 and 82596CA Clock Synchronization
1/3/90 kkp clk_sync
292076-16

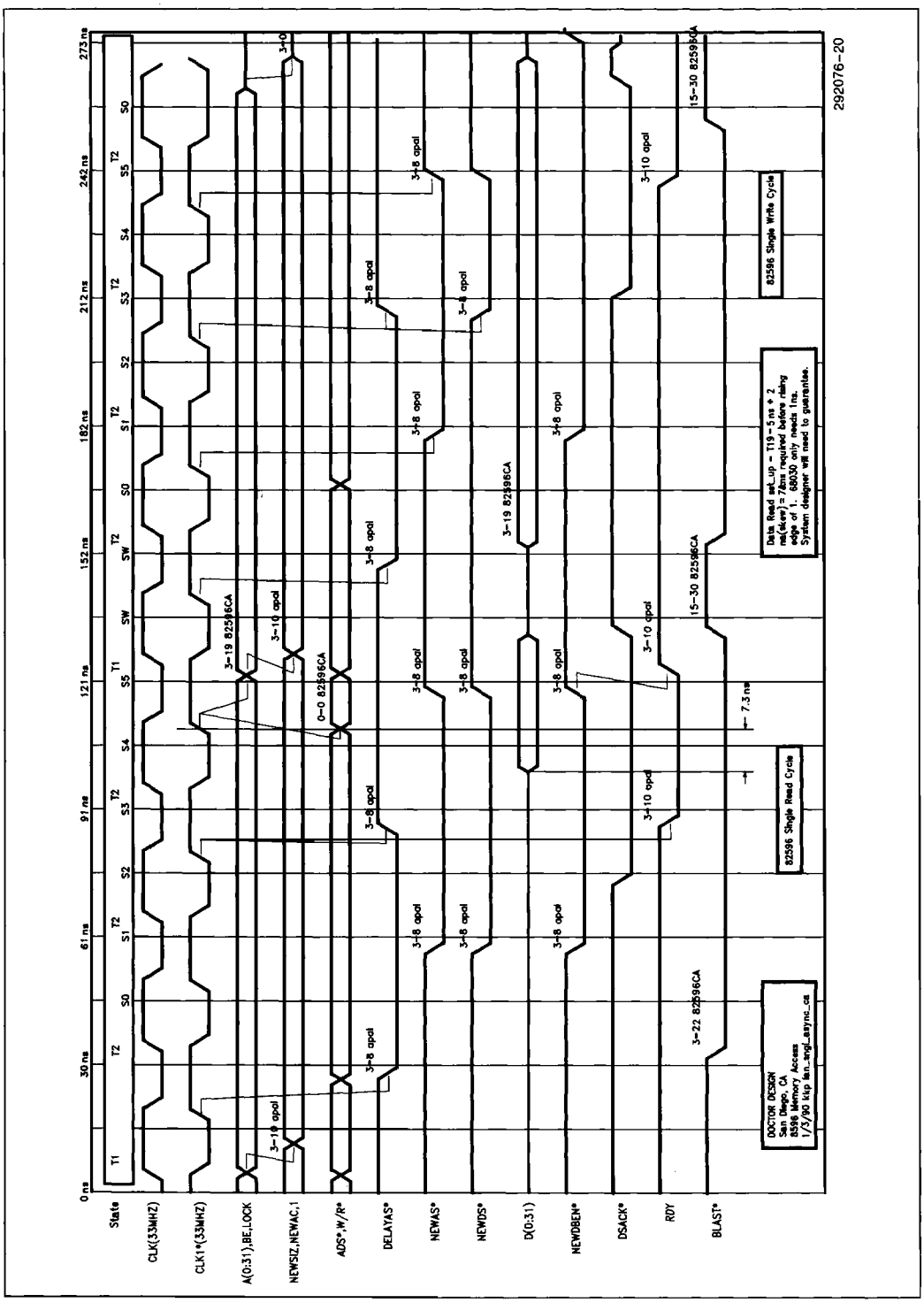
1

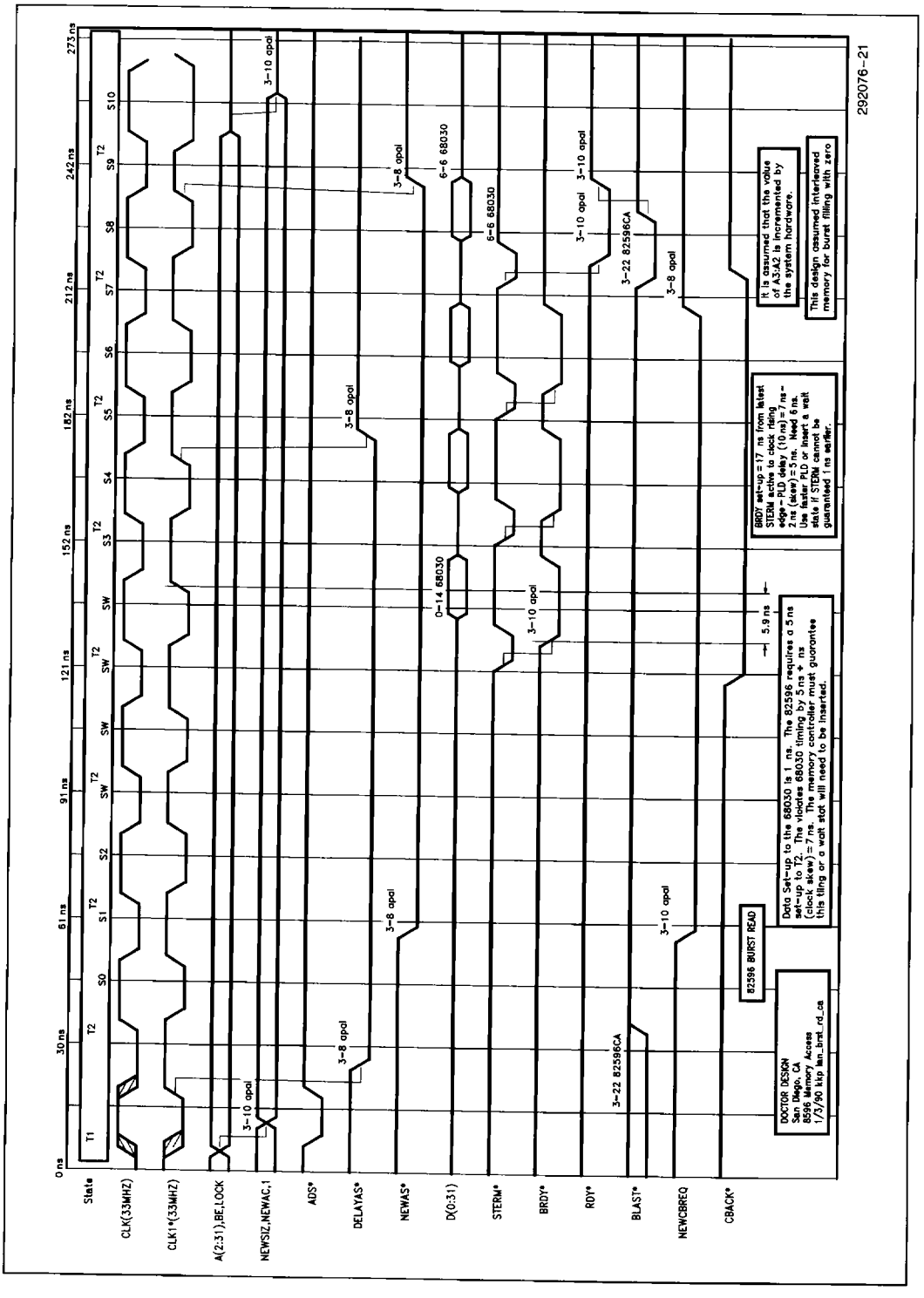

0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK(33MHZ)
CLK1/2(33MHZ)
A(0:31),FCO:2,S
AS*
CA,PORT*
D(0:31)
S0
S1
Ti
S0
S1
S2
S3
W1
W2
W3
W4
Tw
S4
S5
T2
S0
S1
Ti
0-14 68030
2-10 68030
5-12 bpal
0-14 68030
5-12 bpal
5-12 bpal
0-10 68030
5-12 bpal
2-2 68030
0-30 68030
5-12 bpal
5.0 ns
14.4 ns
DOCTOR DESIGN
San Diego, CA
68030 CA & PORT ACCESS
1/5/90 NXP 030_port_CA
CA and PORT
set-up time T13,29
require 7 ns
CA and PORT hold time
T14,30 require 4 ns
The 68030 runs off CLK1. The 82596
runs off CLK1/2. CA and PORT
accesses require 2 wait states.
292076-17

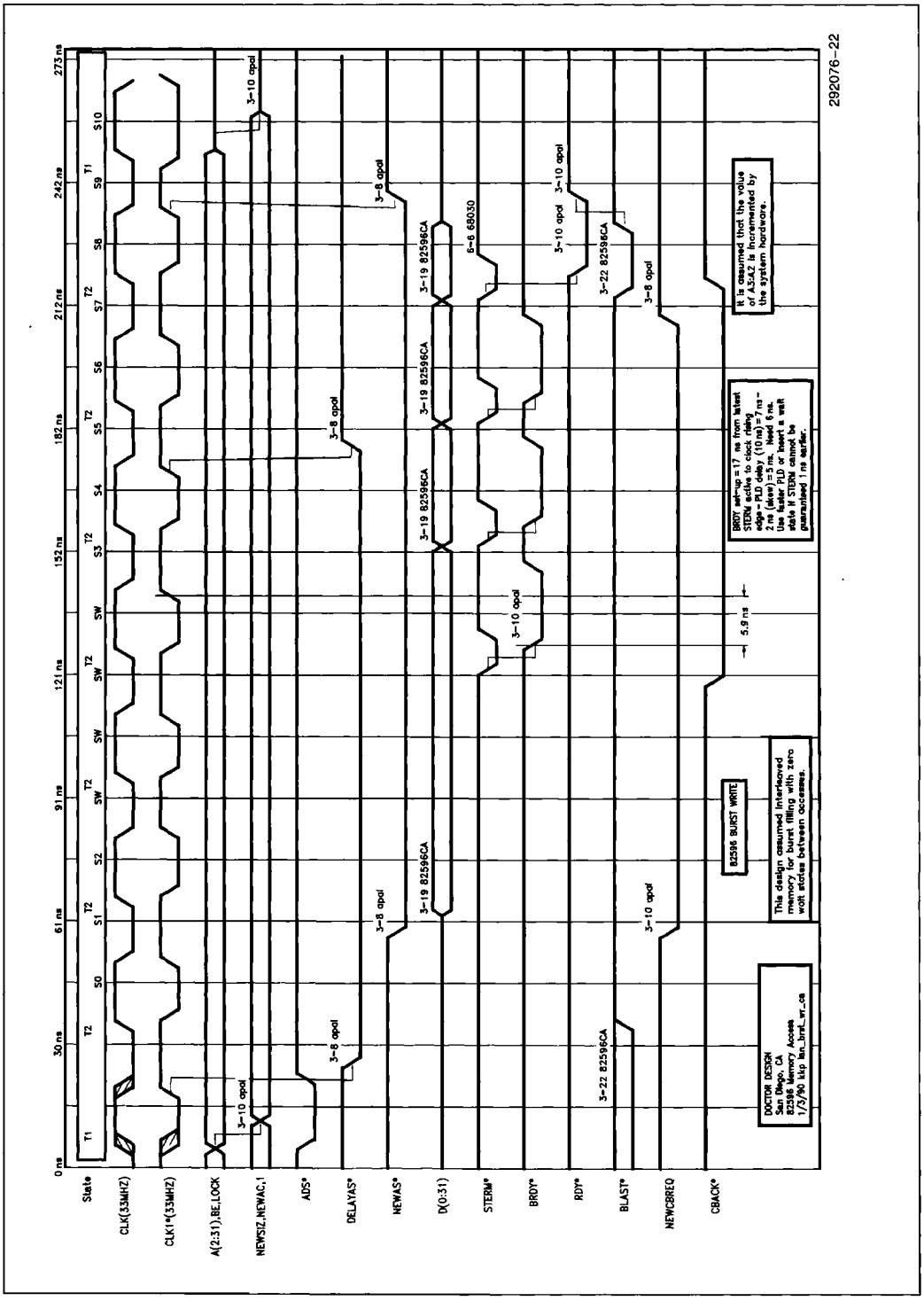

0 ns
30 ns
60 ns
90 ns
120 ns
150 ns
180 ns
210 ns
240 ns
270 ns
T-State
CLK(33MHZ)
CLK1*(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBG*(sync'd)
RFCYC*
LANCYC*(hlda*)
BREQ*
BOFF*
S2
S3
S4
S5
T1
SW
T2
5-15 bpal
5-12 bpal
3-19 82596
NOT SHOWN SCALE...
Actual cycles will be longer
than shown.
DOCTOR DESIGN
San Diego, Ca
68030 Local Bus Arbitration
1/5/90 lckp arb
pg 1 of 2
292076-18

1


270 ns
300 ns
330 ns
360 ns
390 ns
420 ns
450 ns
480 ns
510 ns
540 ns
T-State
T1
CLK(33MHZ)
CLK1*(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBRQ*(sysncd)
RFCYC*
LANCYC*(hlda*)
BREQ*
BOFF*
5-15 bpol
3-19 82596
5-12 bpol
DOCTOR DESIGN
Sam Ahgo, CA
80310 Local Bus Arbitration
1/3/90 lksp orb
Pg 2 of 2
292076-19


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK(33MHZ)
CLK1*(33MHZ)
A(0:31),BE,LOCK
NEWSIZ,NEWAC,1
ADS*,W/R*
DELAYAS*
NEWAS*
NEWDS*
D(0:31)
NEWDBEN*
DSACK*
RDY
BLAST*
3-10 opal
3-19 82596CA
0-0 82596CA
3-8 opal
3-22 82596CA
15-30 82596CA
7.3 ns
DOCTOR DESIGN
San Diego, CA
8596 Memory Access
1/3/90 klp lan_sngl_async_cs
82596 Single Read Cycle
82596 Single Write Cycle
Data Read set_up - T19 - 5 ns + 2 ns(skew) = 7ns required before rising edge of 1. 68030 only needs 1ns. System designer will need to guarantee.
292076-20


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK(33MHZ)
CLK1*(33MHZ)
A(2:31),BE,LOCK
NEWSIZ,NEWAC,1
ADS*
DELAYAS*
NEWAS*
D(0:31)
STERM*
BRDY*
RDY*
BLAST*
NEWCBREQ
CBACK*
T1
T2
S0
S1
S2
SW
S3
S4
S5
S6
S7
S8
S9
S10
3-10 apal
3-8 apal
0-14 68030
5-6 68030
6-6 68030
3-22 82596CA
5.9 ns
82596 BURST READ
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/3/90 ksp ilm_brst_rd_ca
Data Set-up to the 68030 is 1 ns. The 82596 requires a 5 ns set-up to T2. The violates 68030 timing by 5 ns + ns (clock skew) = 7 ns. The memory controller must guarantee this thing or a wait stat will need to be inserted.
BRDY set-up = 17 ns from latest STERM active to clock rising edge - PLD delay (10 ns) = 7 ns - 2 ns (skew) = 5 ns. Need 6 ns. Use faster PLD or insert a wait state if STERM cannot be guaranteed 1 ns earlier.
It is assumed that the value of A3:A2 is incremented by the system hardware.
This design assumed interleaved memory for burst filling with zero
292076-21


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK(33MHZ)
CLK1*(33MHZ)
A(2:31),BE_LOCK
NEWSIZ,NEWAC,1
ADS*
DELAYAS*
NEWAS*
D(0:31)
STERM*
BRDY*
RDY*
BLAST*
NEWCBREQ
CBACK*
T1 T2 S0 T2 S1 T2 S2 SW T2 SW T2 SW SW T2 S3 T2 S4 T2 S5 T2 S6 T2 S7 T2 S8 T2 S9 T1 S10
3-10 opal
3-8 opal
3-19 82596CA
6-6 68030
3-22 82596CA
5.9 ns
82596 BURST WRITE
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/3/90 kkp kn_brst_wr_cs
This design assumed interleaved memory for burst filling with zero wait states between accesses.
BRDY setup = 17 ns from latest STERM active or clock delay edge + PLD delay (10 ns) = 7 ns - 2 ns (skew) = 5 ns. Need 6 ns. Use faster PLD or insert a wait state if STERM cannot be guaranteed 1 ns earlier.
It is assumed that the value of A3:A2 is incremented by the system hardware.

292076-22

MC68030 AND 82596CA TIMING SUMMARY FOR 33 MHz

MC68030 PARAMETERS

6 82596 puts address out 1 clock phase before 68030 S0.

6A $\overline{ADS}$ used to generate $\overline{ECS}$ and $\overline{OCS}$ before $\overline{AS}$ asserted.

7 $\overline{LANCYC}$ off + buffer off = 15 + 10 = 25 ns.

9 Derived from PLD with clock to Q delay of 8 ns.

12 Derived from PLD with clock to Q delay of 8 ns.

12A Worst case could hold $\overline{ECS}$ and $\overline{OCS}$ as long as 20 ns (82596) + delay through buffer. Note to system designer.

13 Could be a violation for $\overline{AS}$, $\overline{DS}$ to address hold of 4 ns (82596) = 8 ns (PLD) − 2 ns (skew) = −6 ns. System designer must guarantee address hold.

14 30 ns + 30 ns − 3 ns (common path through PLD) = 57 ns.

15 30 ns − 2 ns (skew) = 28 ns.

16 Floated with $\overline{LANCYC}$ going high. Minimum 30 ns to next cycle.

17 R/W invalid 1 clock cycle after $\overline{AS}$/$\overline{DS}$ negated.

18 Set with addresses from 82596.

20 Set with addresses from 82596.

21 R/W set 1 clock cycle before $\overline{AS}$ = 30 ns.

22 Write cycle minimum setup to $\overline{DS}$ = 30 ns + 30 ns − 8 ns (R/W through PLD) + 5 ns ($\overline{DS}$ through common PLD) = 57 ns.

23 82596 provides required time.

25 Minimum time = 30 ns (clock) − 8 ns ($\overline{AS}$ through PLD) + 4 ns (82596) = 26 ns.

25A 30 ns − 2 ns (skew) = 28 ns.

26 30 ns + 30 ns − 19 ns (82596) + 3 ns (PLD) = 44 ns.

27 Memory controller must guarantee 1 ns.

28 N/A

29 30 ns + 4 ns (82596) − 8 ns ($\overline{AS}$ through PLD) = 26 ns.

31 N/A

32 Plenty of time

33 Latched in ARB PLD

34 Latched in ARB PLD

35 30 ns + 15 ns = 45 ns (1.5 clocks).

37 30 ns + 15 ns = 45 ns (1.5 clocks).

37A 30 ns in ARB PLD = 1 clock

40 Asserted with $\overline{AS}$, maximum of 8 ns into clock low cycle. This should meet requirements, system designer should verify.

41 Negated in PLD, maximum 8 ns.

42 Asserted in PLD, maximum 8 ns.

43 Negated in PLD, maximum 8 ns.

44 1 clock cycle = 30 ns.

45 Read = 60 ns − 2 ns (skew) = 58 ns. Write = 90 ns − 2 ns = 88 ns.

46 90 ns − 2 ns = 88 ns.

53 Data out from 82596 held valid for extra clock cycle to guarantee.

82596CA PARAMETERS

T13 30 ns − 12 ns (PLD) = 18 ns.

T14 5 ns through PLD for $\overline{PORT}$.

T17 30 ns − 10 ns (PLD) = 20 ns.

T18 3 ns ($\overline{DBEN}$ through PLD) + 3 ns (PLD) = 6 ns.

T19 May violate by 5 ns + 2 ns (skew) − 1 ns (memory controller) = 6 ns. System designer will need to guarantee extra 2 ns setup time.

T20 3 ns ($\overline{DS}$ from PLD) + delay through memory controller.

T21 30 ns − 8 ns = 22 ns.

T23 30 ns − 9 ns − 2 ns skew = 19 ns.

T24 4 ns minimum through flip-flop.

T26 3 CLK2 cycles.

T27 30 ns + 15 ns − 18 ns (68030) = 27 ns

T28 15 ns − 12 ns (PLD) + 2 ns (68030) − 2 ns (skew) = 3 ns

T29 30 ns − 12 ns = 18 ns.

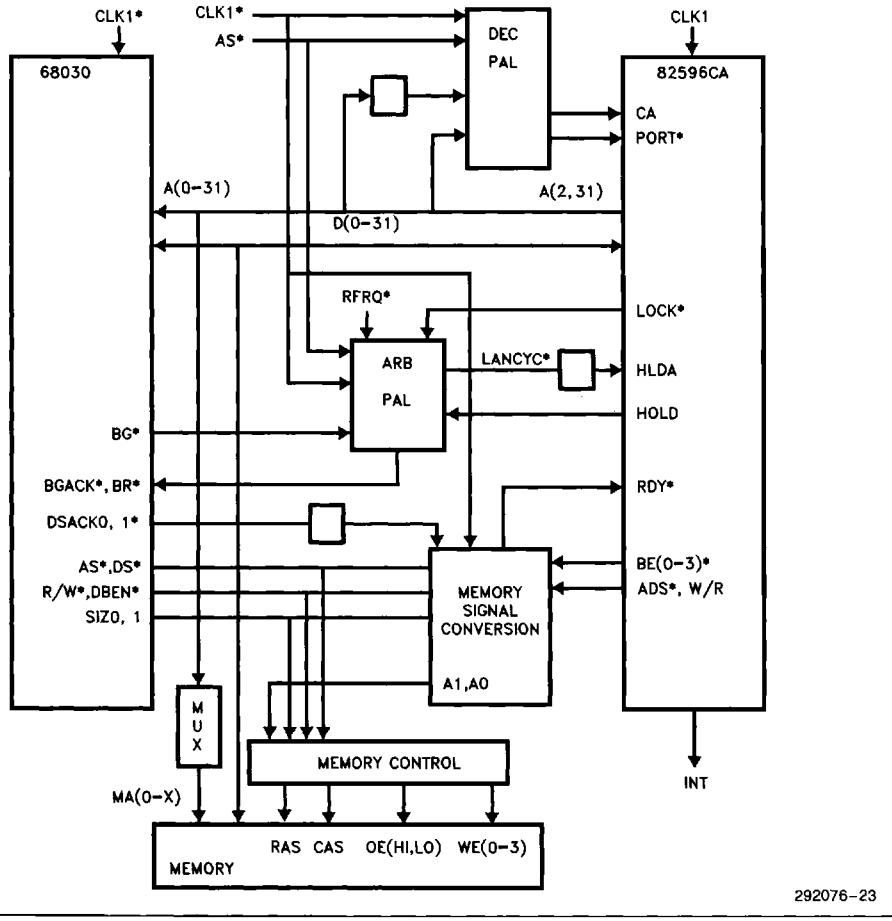
T30 Minimum 3 ns through 10, 12, or 15 ns PLD.

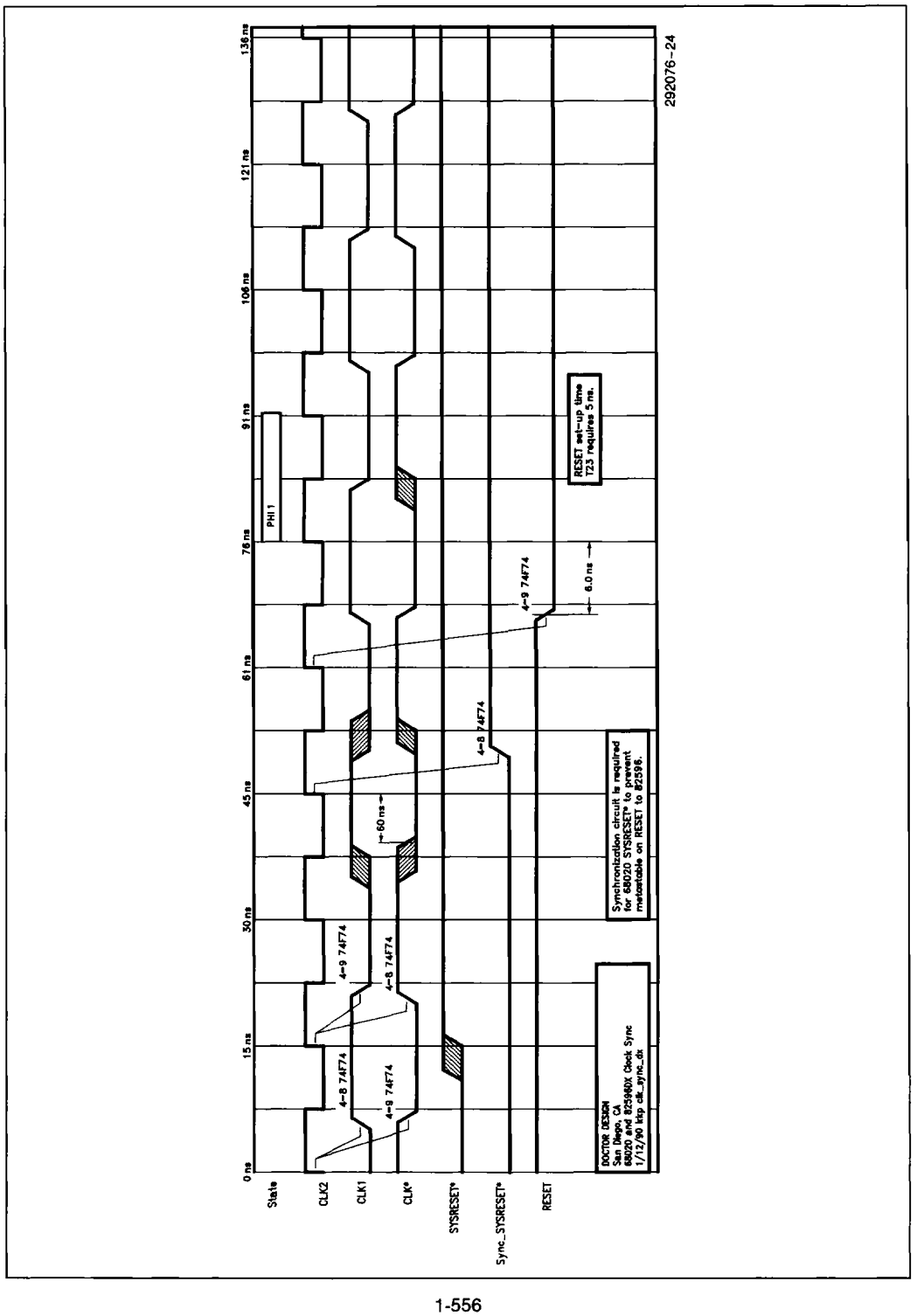
N/A = Not Applicable

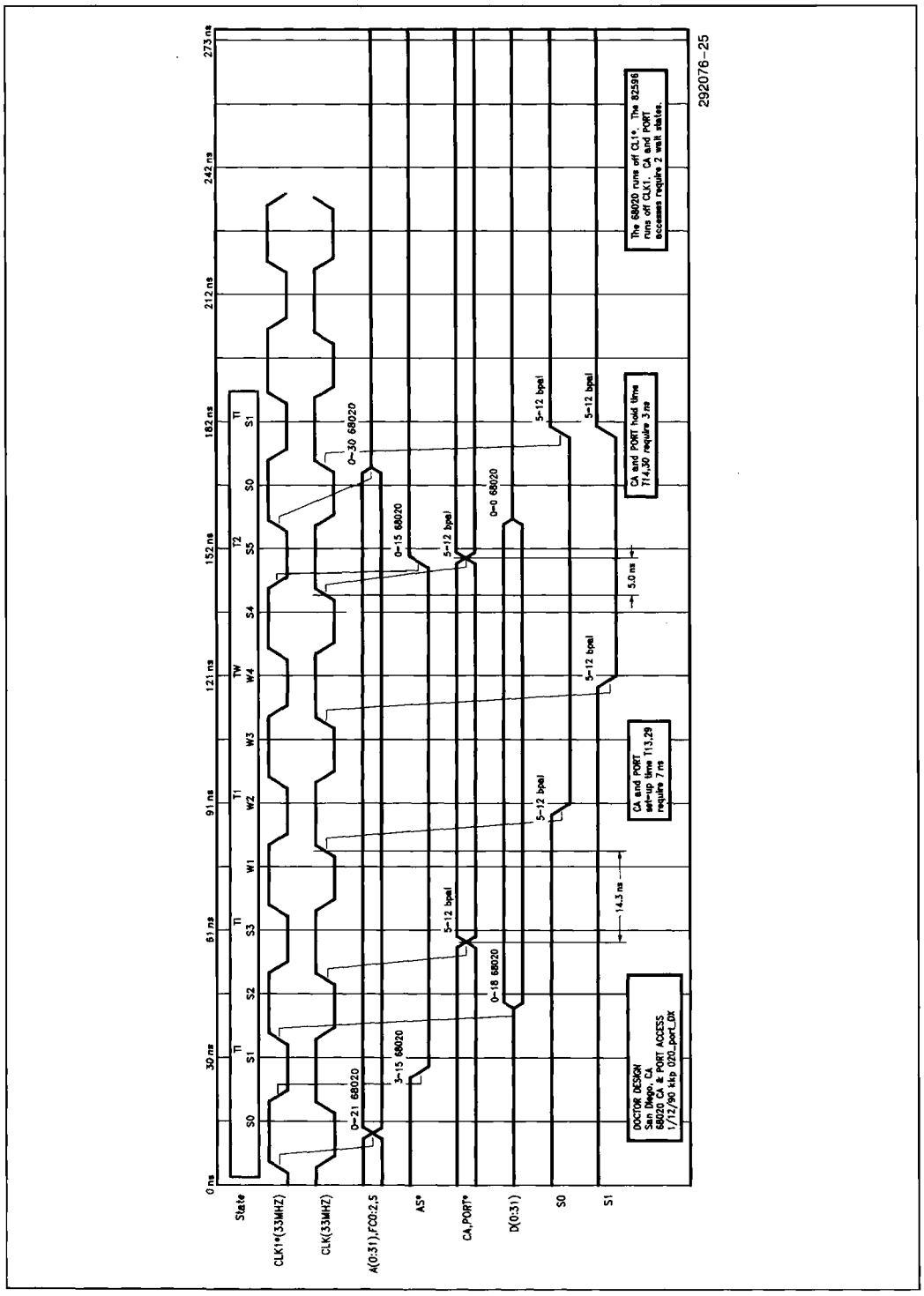
15 ns = ½ clock period

30 ns = 1 clock period

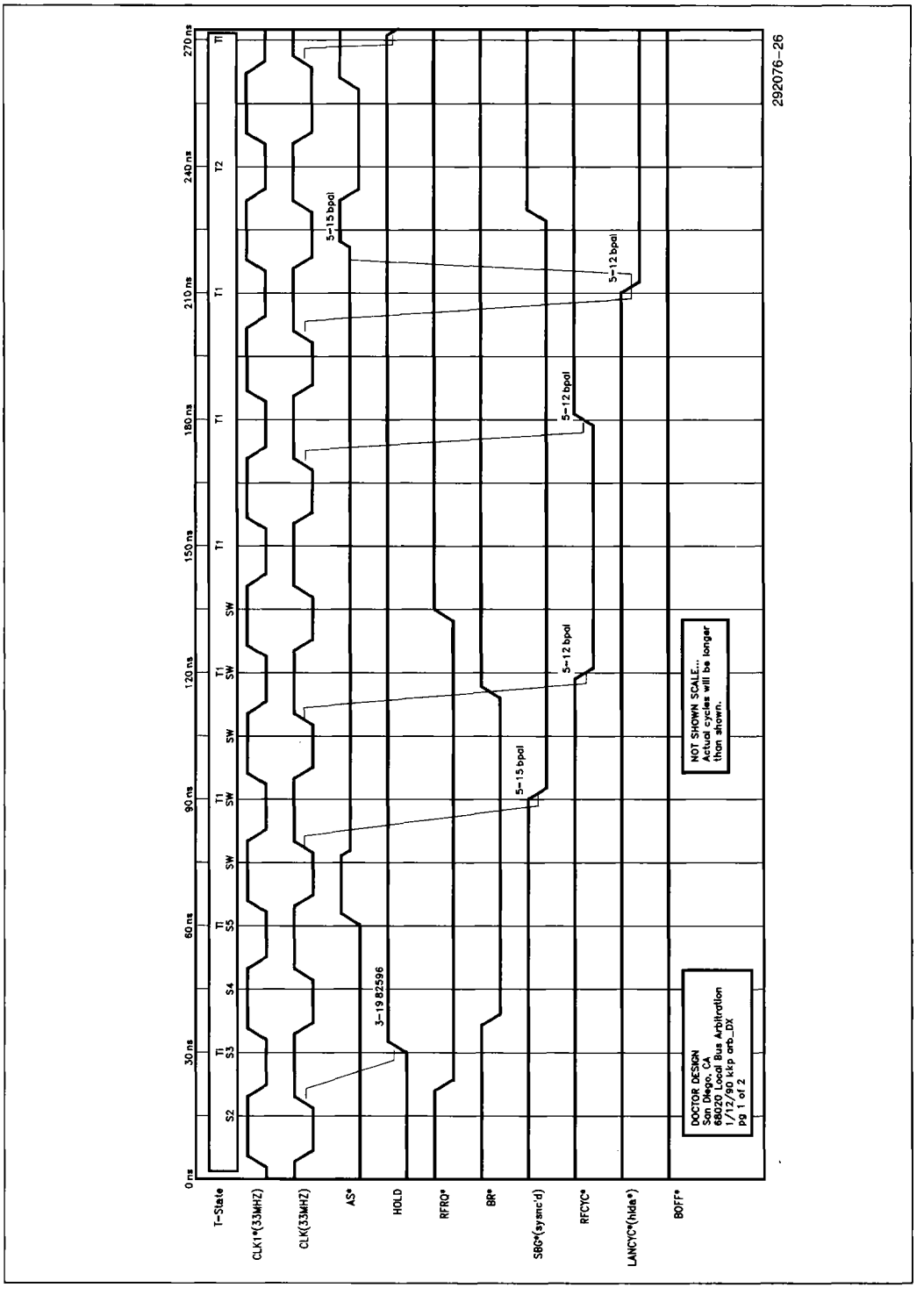
INTERFACE BETWEEN 68020 AND 82596DX LAN CONTROLLER

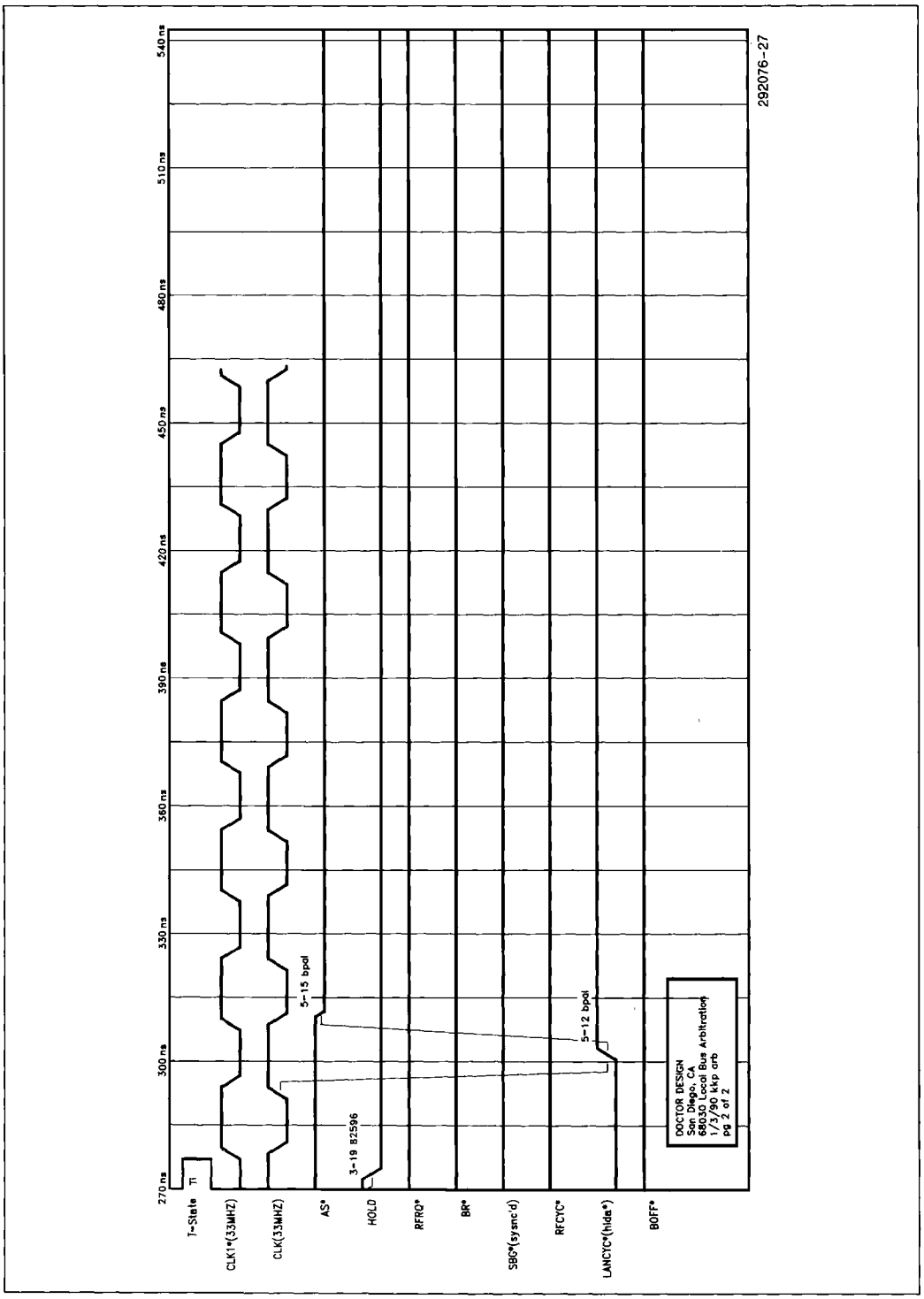


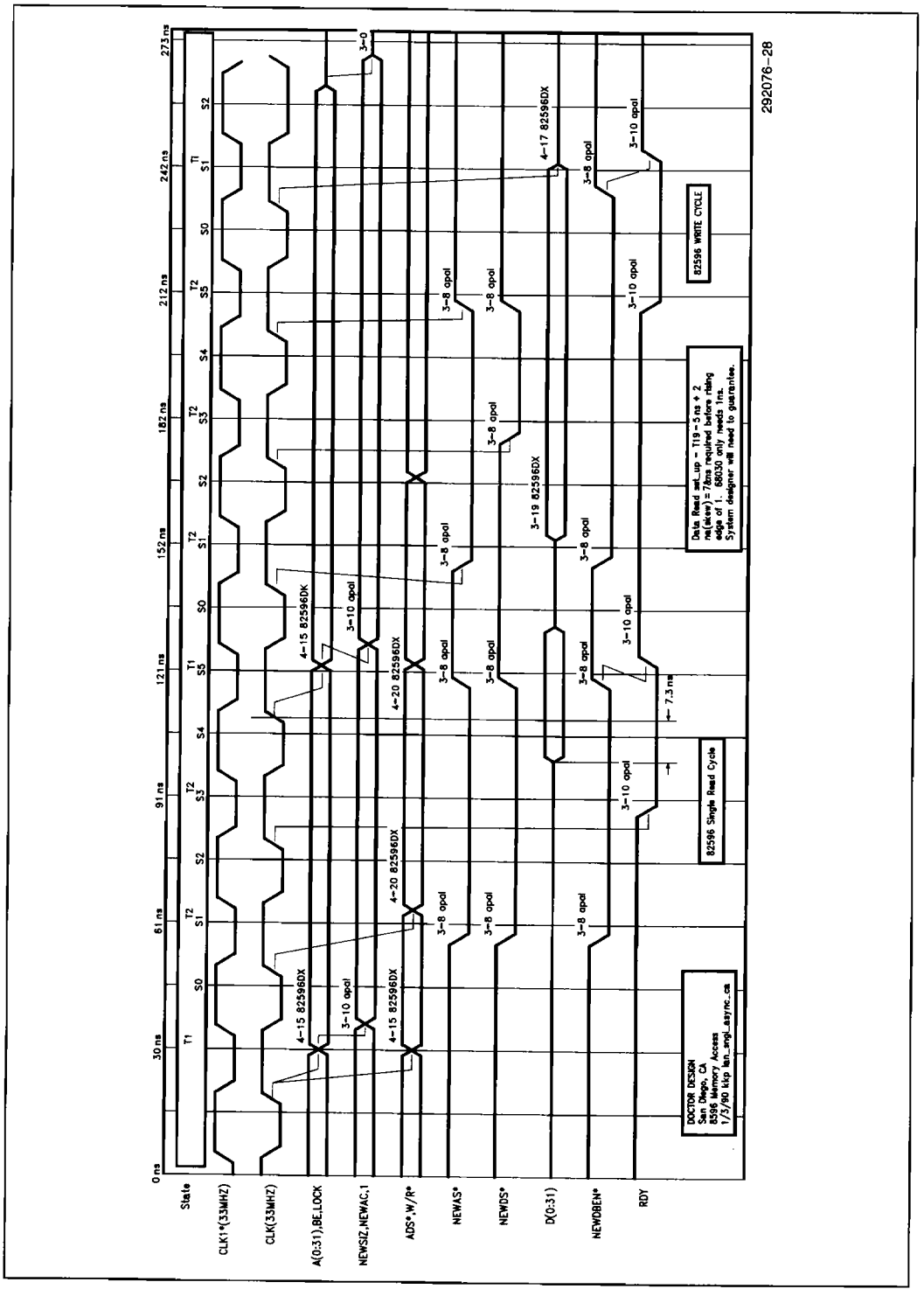

0 ns
15 ns
30 ns
45 ns
61 ns
76 ns
91 ns
106 ns
121 ns
136 ns
State
CLK2
CLK1
CLK*
SYSRESET*
Sync_SYSRESET*
RESET
PHI 1
4–8 74F74
4–9 74F74
4–8 74F74
4–9 74F74
4–8 74F74
4–9 74F74
60 ns
6.0 ns
RESET set-up time T23 requires 5 ns.
Synchronization circuit is required for 68020 SYSRESET* to prevent metastable on RESET to 82596.
DOCTOR DESIGN
San Diego, CA
68020 and 82596X Clock Sync
1/12/90 kkp clk_sync_dx
292076-24


292076-25
0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK1*(33MHZ)
CLK(33MHZ)
A(0:31),FC0:2,5
AS*
CA,PORT*
D(0:31)
S0
S1
S0 TI S1 S2 S3 TI W1 W2 TI W3 W4 TW S4 S5 T2 S0 TI S1
0=21 68020
3=15 68020
5=12 bpel
0=18 68020
0=30 68020
0=15 68020
0=0 68020
14.3 ns
5.0 ns
DOCTOR DESIGN
San Diego, CA
68020 CA & PORT ACCESS
1/12/90 ktb 020_port_DX
CA and PORT set-up time T13,29 require 7 ns
CA and PORT hold time T14,30 require 3 ns
The 68020 runs off CLK1*. The 82596 runs off CLK1. CA and PORT accesses require 2 wait states.

1


0 ns
30 ns
60 ns
90 ns
120 ns
150 ns
180 ns
210 ns
240 ns
270 ns
T-State
CLK1*(33MHZ)
CLK(33MHZ)
AS*
HOLD
RFRQ*
BR*
S80*(sync'd)
RFCYC*
LANCYC*(hlda*)
BOFF*
S2
S3
S4
S5
SW
T1
T2
5-15 bpol
5-12 bpol
3-19 82596
NOT SHOWN SCALE...
Actual cycles will be longer
than shown.
DOCTOR DESIGN
San Diego, CA
68020 Local Bus Arbitration
1/12/90 kkp arb_DX
pg 1 of 2
292076-26


270 ns
300 ns
330 ns
360 ns
390 ns
420 ns
450 ns
480 ns
510 ns
540 ns
T-State
CLK1*(33MHZ)
CLK(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBO*(sync'd)
RFCYC*
LANCYC*(hida*)
BOFF*
5-15 bpol
3-19 82596
5-12 bpol
DOCTOR DESIGN
San Diego, CA
68030 Local Bus Arbitration
1/3/90 kkp arb
pg 2 of 2
292076-27


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK1*(33MHZ)
CLK(33MHZ)
A(0:31),BELOCK
NEWSIZ,NEWAC,1
ADS*,W/R*
NEWAS*
NEWDS*
D(0:31)
NEWDBEN*
RDY
T1
T2
S0
S1
S2
S3
S4
S5
4-15 82596DX
4-15 82596DK
4-20 82596DX
3-19 82596DX
4-17 82596DX
3-10 opal
3-8 opal
3-0
7.3ns
82596 Single Read Cycle
82596 WRITE CYCLE
DOCTOR DESIGN
San Diego, CA
8596 Memory Access
1/3/90 kkp lan_sngl_asyn.cs
292076-28

MC68020 AND 82596DX TIMING SUMMARY FOR 33 MHz

MC68020 PARAMETERS

6 82596 puts address out 1 clock phase before 68020 S0.

6A $\overline{ADS}$ used to generate $\overline{ECS}$ and $\overline{OCS}$ before $\overline{AS}$ asserted.

7 $\overline{LANCYC}$ off + buffer off = 15 + 10 = 25 ns.

9 Derived from PLD with clock to Q delay of 8 ns.

12 Derived from PLD with clock to Q delay of 8 ns.

12A Worst case could hold $\overline{ECS}$ and $\overline{OCS}$ as long as 20 ns (82596) + delay through buffer. Note to system designer.

13 Could be a violation for $\overline{AS}$, $\overline{DS}$ to address hold of 4 ns (82596) = 8 ns (PLD) − 2 ns (skew) = −6 ns. System designer must guarantee address hold.

14 30 ns + 30 ns − 3 ns (common path through PLD) = 57 ns.

15 30 ns − 2 ns (skew) = 28 ns.

16 Floated with $\overline{LANCYC}$ going high. Minimum 30 ns to next cycle.

17 R/W invalid 1 clock cycle after $\overline{AS}$/$\overline{DS}$ negated.

18 Set with addresses from 82596.

20 Set with addresses from 82596.

21 Setting with $\overline{AS}$ could violate read cycle timing. System designer must guarantee that 5 ns setup is not required.

22 Write cycle minimum setup to $\overline{DS}$ = 30 ns − 8 ns (R/W through PLD) + 5 ns ($\overline{DS}$ through common PLD) = 27 ns. The system designer must verify that this meets memory controller timing.

23 82596 provides required time.

25 Minimum time = 30 ns (clock) − 8 ns ($\overline{AS}$ through PLD) + 4 ns (82596) = 26 ns.

25A 30 ns − 2 ns (skew) = 28 ns.

26 30 ns − 19 ns (82596) + 3 ns (PLD) = 14 ns.

27 Memory controller must guarantee 5 ns.

28 N/A

29 30 ns + 4 ns (82596) − 8 ns ($\overline{AS}$ through PLD) = 26 ns.

31 N/A

32 Plenty of time

33 Latched in ARB PLD

34 Latched in ARB PLD

35 30 ns + 15 ns = 45 ns (1.5 clocks).

37 30 ns + 15 ns = 45 ns (1.5 clocks).

37A 30 ns in ARB PLD = 1 clock

40 Asserted with $\overline{AS}$, maximum of 8 ns into clock low cycle. This should meet requirements, system designer should verify.

41 Negated in PLD, maximum 8 ns.

42 Asserted in PLD, maximum 8 ns.

43 Negated in PLD, maximum 8 ns.

44 Asserted with R/W in PLD. System designer will need to verify that 5 ns setup is not required.

45 Read = 60 ns − 2 ns (skew) = 58 ns. Write = 90 ns − 2 ns = 88 ns.

46 90 ns − 2 ns = 88 ns.

53 Data out from 82596 held valid for extra clock cycle to guarantee.

82596DX PARAMETERS

T13 30 ns − 12 ns (PLD) = 18 ns.

T14 5 ns through PLD for PORT.

T17 30 ns − 10 ns (PLD) = 20 ns.

T18 3 ns ($\overline{DBEN}$ through PLD) + 3 ns (PLD) = 6 ns.

T19 May violate by 5 ns + 2 ns (skew) − 5 ns (memory controller) = 2 ns. System designer will need to guarantee extra 2 ns setup time.

T20 3 ns ($\overline{DS}$ from PLD) + delay through memory controller.

T21 30 ns − 8 ns = 22 ns.

T22 3 ns (PLD) + external inverter.

T26 3 CLK2 cycles.

T27 30 ns + 15 ns − 18 ns (68020) = 27 ns

T29 30 ns − 12 ns = 18 ns.

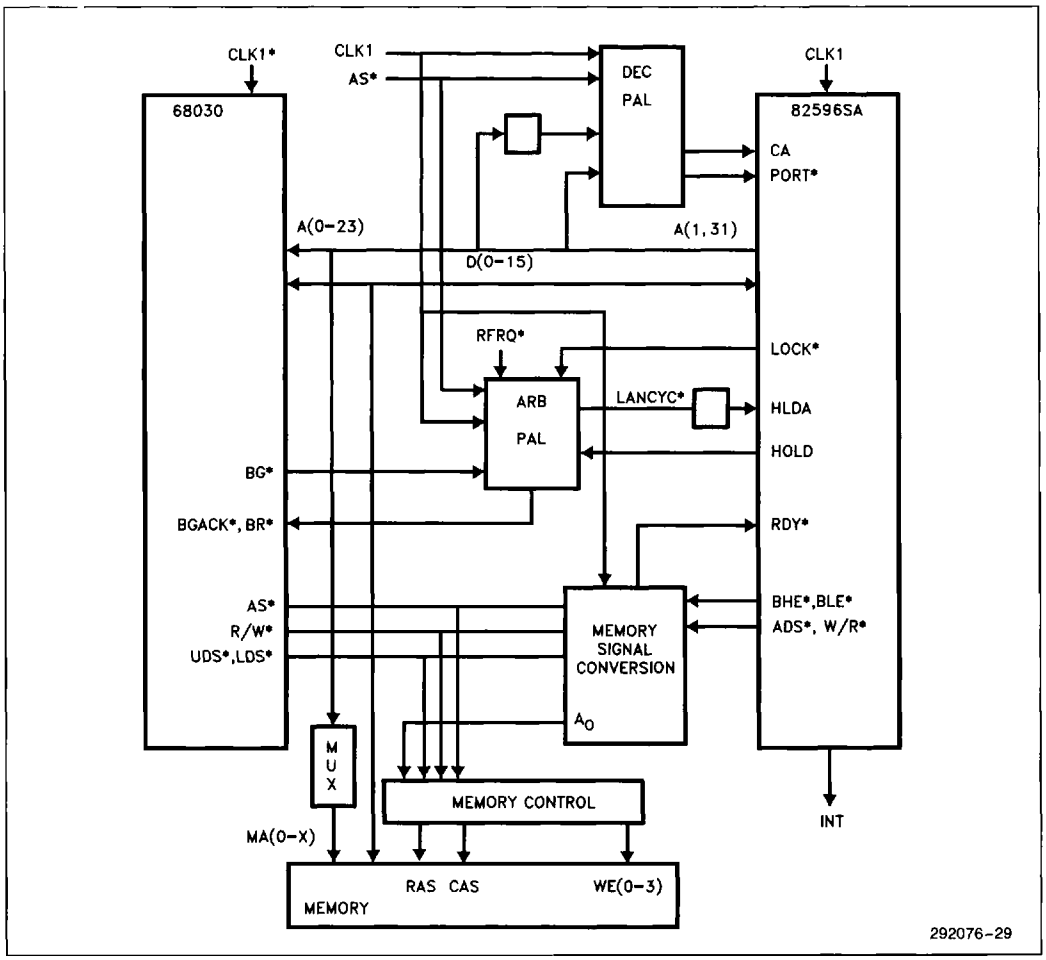
T30 Minimum 3 ns through 10, 12, or 15 ns PLD.

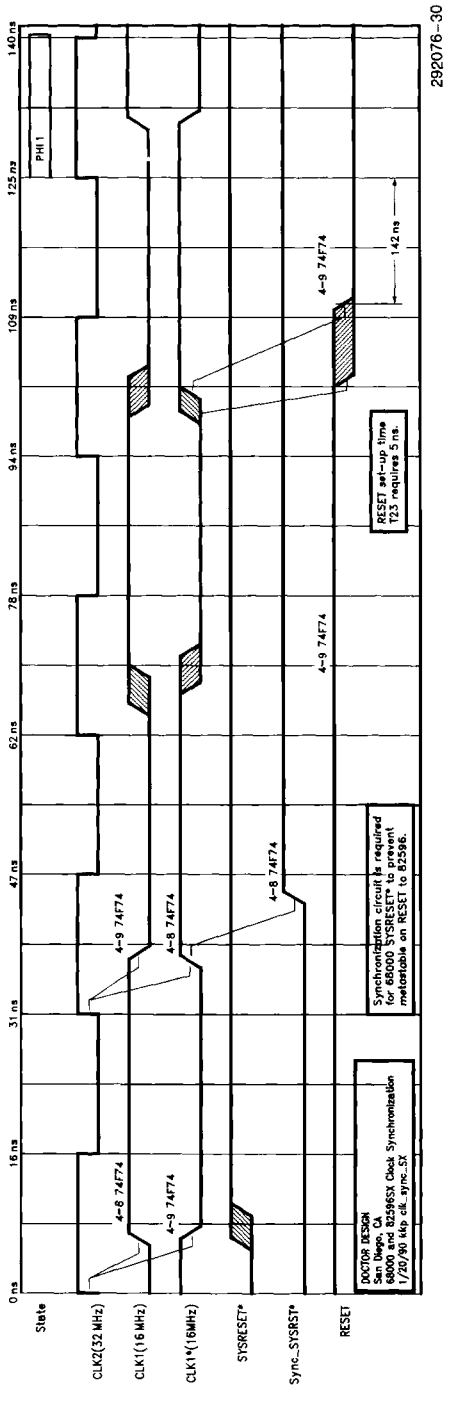
N/A = Not Applicable

15 ns = ½ clock period

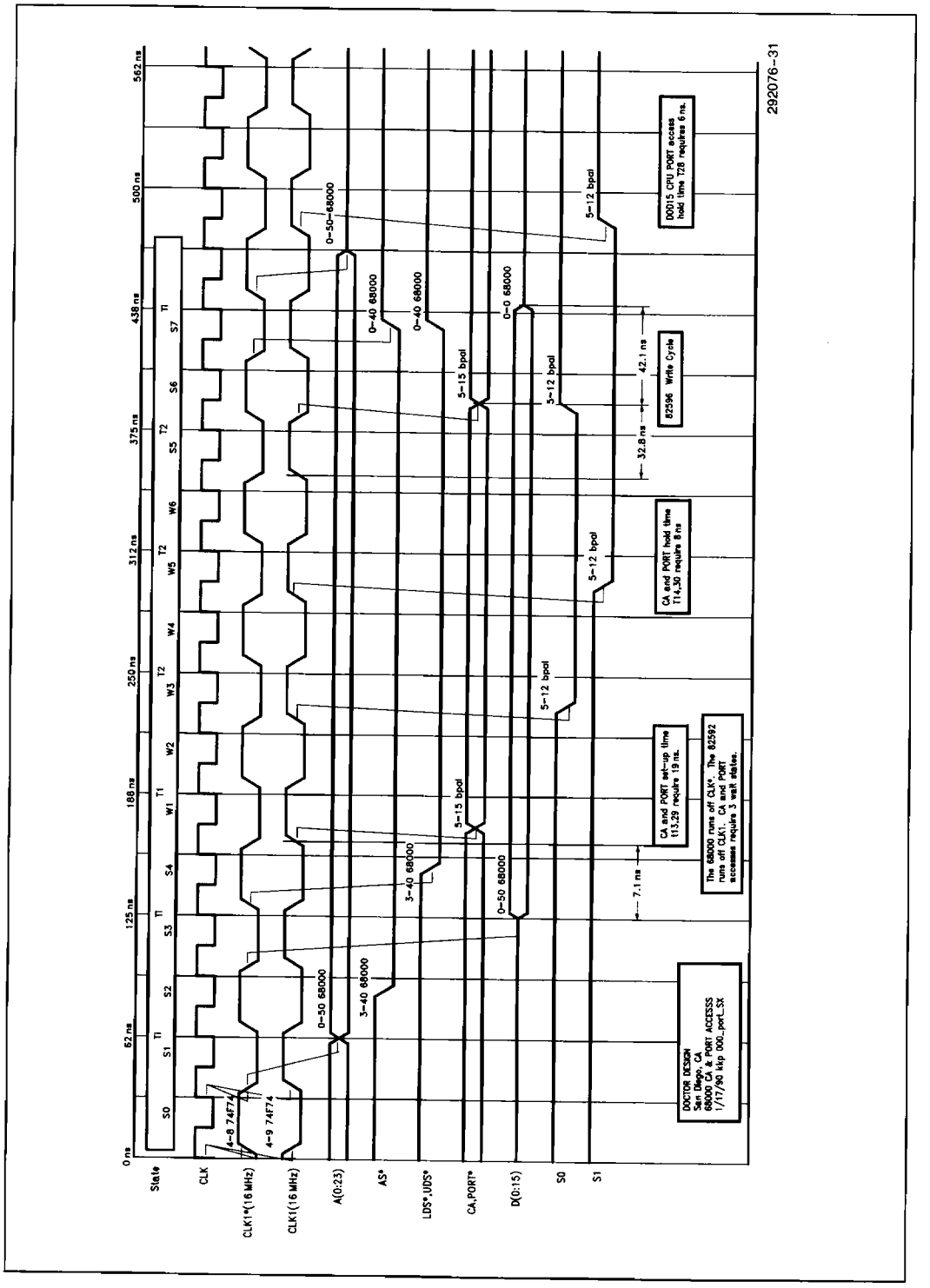
30 ns = 1 clock period

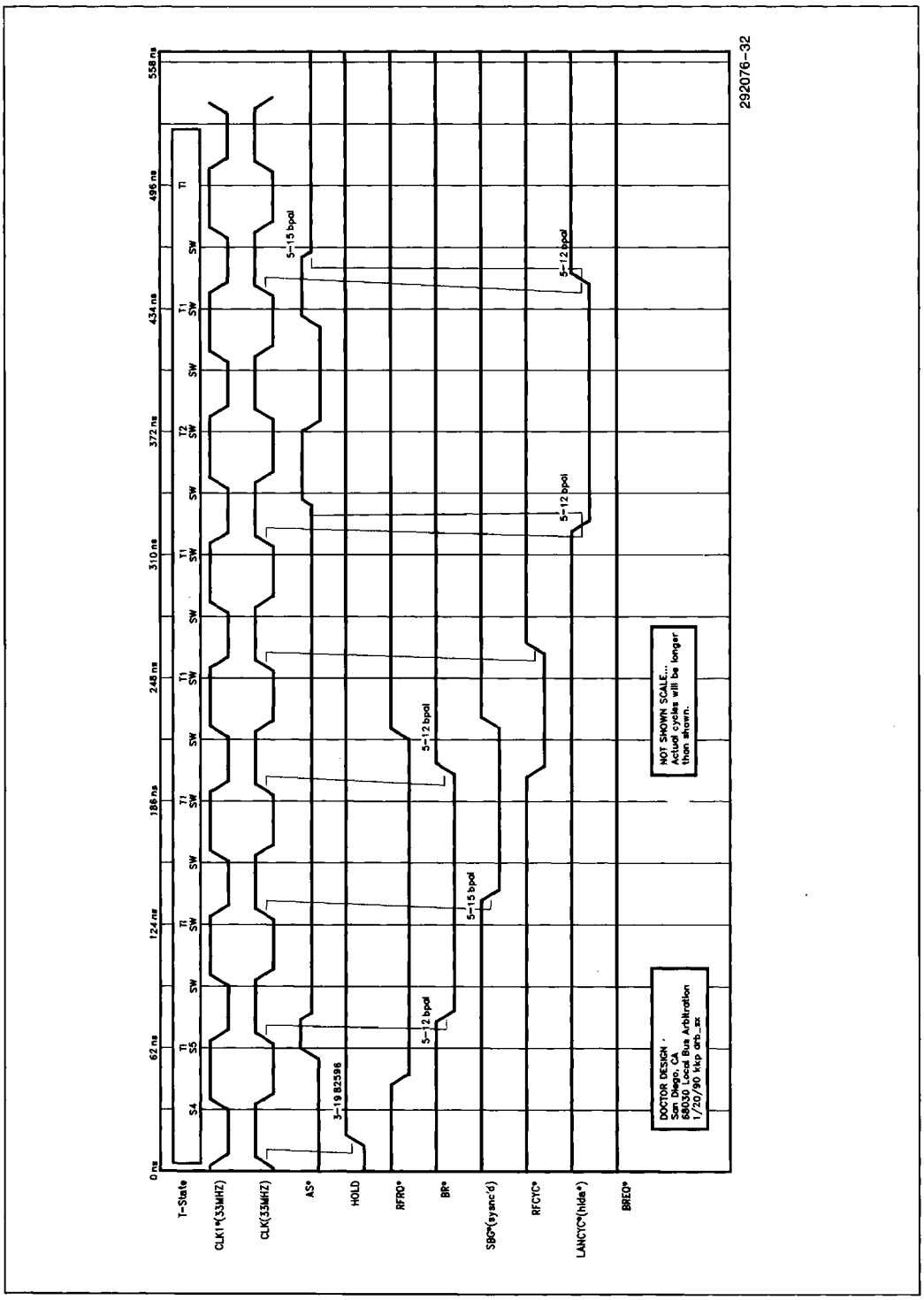
INTERFACE BETWEEN 68000 AND 82596SX LAN CONTROLLER



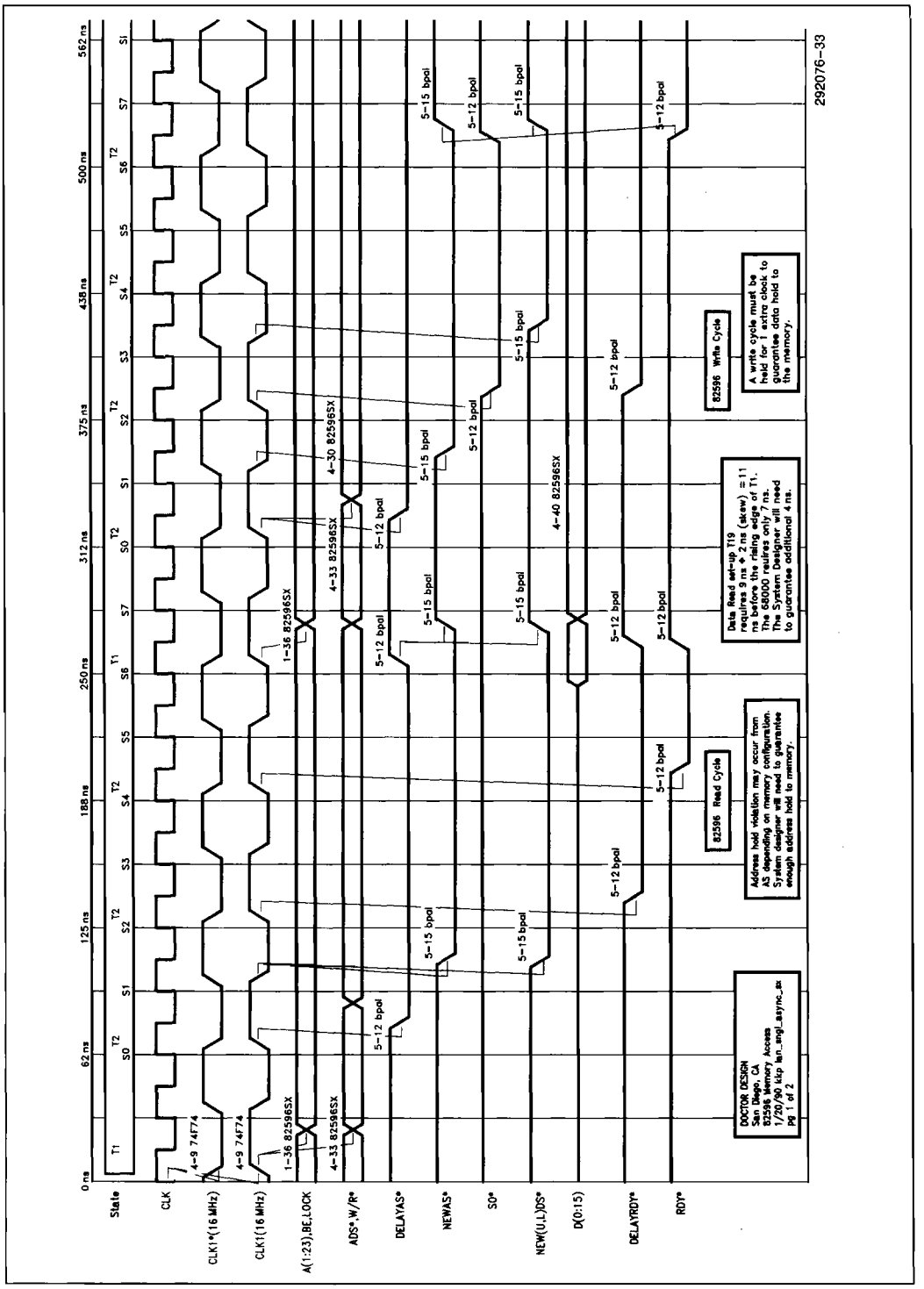

292076-30
0 ns
16 ns
31 ns
47 ns
62 ns
78 ns
94 ns
109 ns
125 ns
140 ns
PHI 1
142 ns
4-9 74F74
4-8 74F74
4-9 74F74
4-8 74F74
4-9 74F74
4-8 74F74
4-9 74F74
4-8 74F74
RESET set-up time T23 requires 5 ns.
Synchronization circuit is required for 68000 SYSRESET* to prevent metastable on RESET to 82596.
DOCTOR DESIGN
San Diego, CA
68000 and 82596SX Clock Synchronization
1/20/90 kkp clk_sync_SX
State
CLK2(32 MHz)
CLK1(16 MHz)
CLK1*(16MHz)
SYSRESET*
Sync_SYSRST*
RESET

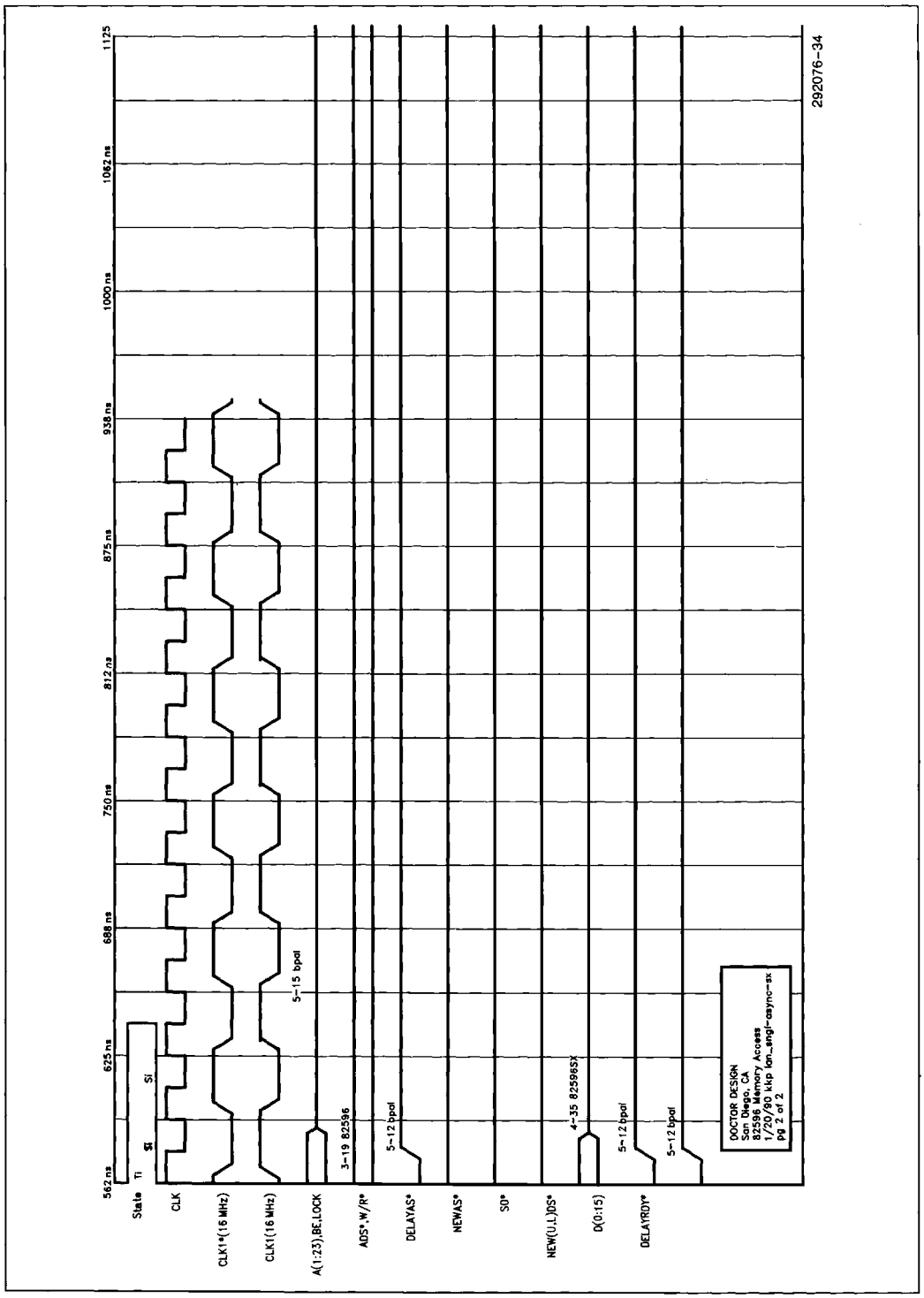
1


0 ns
62 ns
125 ns
188 ns
250 ns
312 ns
375 ns
438 ns
500 ns
562 ns
State
CLK
CLK1*(16 MHz)
CLK1(16 MHz)
A(0:23)
AS*
LDS*,UDS*
CA,PORT*
D(0:15)
S0
S1
S0 S1 S2 S3 S4 W1 W2 W3 W4 W5 W6 S5 S6 S7
T1 T2
4-8 74F74
4-9 74F74
0-50 68000
3-40 68000
5-15 bpal
5-12 bpal
0-40 68000
0-0 68000
0-50-68000
7.1 ns
32.8 ns
42.1 ns
CA and PORT set-up time 115,29 require 19 ns.
The 68000 runs off CLK*. The 82592 runs off CLK1. CA and PORT access require 3 wait states.
CA and PORT hold time 114,30 require 8 ns
82596 Write Cycle
DO015 CPU PORT access hold time T28 requires 6 ns.
DOCTOR DESIGN
Sam Hyde, CA
68000 CA & PORT ACCESS
1/17/90 hkp 000_port_SX
292076-31


292076-32
0 ns
62 ns
124 ns
186 ns
248 ns
310 ns
372 ns
434 ns
496 ns
558 ns
T-State
CLK1*(33MHZ)
CLK(33MHZ)
AS*
HOLD
RFRD*
BR*
SBCP*(sync'd)
RFCYC*
LANCYC*(hide*)
BREQ*
5-19 82596
5-12 bpol
5-15 bpol
DOCTOR DESIGN
San Diego, CA
82596 Local Bus Arbitration
1/20/90 kkp arb_tx
NOT SHOWN SCALE...
Actual cycles will be longer
than shown.

1


0 ns
62 ns
125 ns
188 ns
250 ns
312 ns
375 ns
438 ns
500 ns
562 ns
State
CLK
CLK1*(16 MHz)
CLK1(16 MHz)
A(1:23),BE,LOCK
ADS*,W/R*
DELAYAS*
NEWAS*
SO*
NEW(U,L)DS*
D(0:15)
DELAYRDY*
RDY*
4-9 74F74
1-36 82596SX
4-33 82596SX
4-30 82596SX
4-40 82596SX
5-15 bpal
5-12 bpal
82596 Read Cycle
82596 Write Cycle
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/20/90 kkp lan_angl_async_sx
pg 1 of 2
Address hold violation may occur from A3 depending on memory configuration. System designer will need to guarantee enough address hold to memory.
Data Read set-up T19 requires 9 ns + 2 ns (skew) = 11 ns before the rising edge of T1. The 68000 requires only 7 ns. The System Designer will need to guarantee additional 4 ns.
A write cycle must be held for 1 extra clock to guarantee data hold to the memory.
292076-33


562 ns
625 ns
688 ns
750 ns
812 ns
875 ns
938 ns
1000 ns
1062 ns
1125
State
T1
S1
S1
CLK
CLK1*(16 MHz)
CLK1(16 MHz)
A(1:23),BE,LOCK
5-15 bpal
3-19 82596
ADS*,W/R*
5-12 bpal
DELAYAS*
NEWAS*
S0*
NEW(U,L)DS*
4-35 825965X
D(0:15)
5-12 bpal
DELAYRDY*
5-12 bpal
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/20/90 kkp ion_engr-async-sx
pg 2 of 2
292076-34

1

MC68000 AND 82596SX TIMING SUMMARY FOR 33 MHz

MC68000 PARAMETERS

6 82596 puts address out 1 clock phase before 68000 S0.

6A FC valid when 82596 arbitrates for the bus, with $\overline{\text{LANCYC}}$ valid.

7 Address goes away with $\overline{\text{LANCYC}}$ going invalid.

8 Address goes away with $\overline{\text{AS}}$. System design must verify that this meets memory controller requirements.

9 Derived from 15 ns PLD.

11 62 ns − 36 ns (82596 address delay) + 31 ns + 5 ns ($\overline{\text{AS}}$ through PLD) = 62 ns.

11A FC valid when 82596 arbitrates for the bus, with $\overline{\text{LANCYC}}$ valid.

12 12 ($\overline{\text{DELAYAS}}$) + 15 ns ($\overline{\text{AS}}$) = 27 ns.

13 FC held until $\overline{\text{LANCYC}}$ goes invalid.

14 16 ns + 62 ns + 62 ns + 10 ns (2 PLDs) = 150 ns.

15 62 ns minimum.

16 Control bus held until $\overline{\text{LANCYC}}$ goes invalid.

17 $\overline{\text{AS}}$ negated to R/W negated is the minimum time through the RDY PLD. The system designer must verify that this meets memory controller requirements.

18 Maximum from clock high is 33 ns (82596) + 15 ns (PLD) + 2 ns (skew) − 31 ns (clock) = 19 ns.

20 Same as 18 above.

21 No timing relationship is given for the 82596 between address valid and W/R low. It is assumed that if address is delayed out of the 82596, W/R will be delayed by about the same amount. The W/R signal has an additional PLD delay for inversion.

22 Minimum time = 19 ns (18 above) − 31 ns (clock) + 62 ns (clock) = 50 ns.

23 82596 outputs data immediately on a write cycle.

25 $\overline{\text{AS}}$, $\overline{\text{DS}}$ negated 1 clock cycle before ending 82596 write to meet this parameter by 62 ns − 27 ns (2 PLD delays) − 2 ns (skew) = 33 ns.

26 82596 outputs data immediately on a write cycle.

27 Data setup to clock low for 68000 is 7 ns which could violate T19 below.

28 N/A

29 Memory controller guarantees 0 ns.

31 N/A

32 Transition time depends on flip-flop used for deriving RESET.

33 Setup to PLD.

34 Setup to PLD.

38 Synchronizing $\overline{\text{BG}}$ and generating $\overline{\text{LANCYC}}$ will be a minimum of 31 ns + 62 ns = 93 ns.

46 Minimum width low for 82596 cycle is read cycle − 4 clocks.

53 Minimum = 31 ns (clock) + 4 ns (82596) = 35 ns.

82596SX PARAMETERS

T13 62 ns − 15 ns (PLD) = 47 ns.

T14 31 ns + 5 (PLD) + 36 ns.

T17 62 ns − 12 ns (PLD) = 50 ns.

T18 5 ns (PLD).

T19 May violate by 9 ns (82596 setup) + 2 ns (skew) − 7 ns (memory controller) = 4 ns. System designer must verify that this meets requirements.

T20 5 ns ($\overline{\text{DELAYAS}}$ from PLD) + 5 ns ($\overline{\text{DS}}$ through PLD) + delay through memory controller.

T21 62 ns − 12 ns = 50 ns.

T23 31 ns − 8 ns (flip-flops) − 9 ns (FF) = 14 ns.

T24 4 ns (FF) − 4 ns (FF) = 8 ns

T26 N/A

T27 62 ns − 50 ns (68000) − 5 ns (PORT from PLD) = 69 ns

T28 62 ns − 5 ns (PORT through PLD) = 57 ns.

T29 62 ns − 15 ns = 47 ns.

T30 31 ns + 5 ns (PLD) = 36 ns.

N/A = Not Applicable

31 ns = ½ clock period

62 ns = 1 clock period

APPENDIX D
PARTS LISTS

Each parts list includes only those components that are part of the interface. The memory controller and memory components are not included.

D.1 MC68030/82596CA

Quantity	Generic Number	Description
1.5	74F74	Dual D Flip-Flop
3	20R4	24-pin PLD; 4 Registered Outputs
1	20R8	24-pin PLD; 8 Registered Outputs
0.5	74F244	Octal Tri-State Buffer

Each PLD must have no more than 10 ns propagation delay for 33 MHz design.
Each PLD must have no more than 15 ns propagation delay for 25 MHz design.

D.2 MC68020/82596DX

Quantity	Generic Number	Description
1.5	74F74	Dual D Flip-Flop
2	20R4	24-pin PLD; 4 Registered Outputs
1	20R6	24-pin PLD; 6 Registered Outputs
0.5	74F244	Octal Tri-State Buffer

Each PLD must have no more than 10 ns propagation delay for 33 MHz design.
Each PLD must have no more than 15 ns propagation delay for 25 MHz design.

D.3 MC68000/82596SX

Quantity	Generic Number	Description
1.5	74F74	Dual D Flip-Flop
2	20R4	24-pin PLD; 4 Registered Outputs
1	20R8	24-pin PLD; 8 Registered Outputs

Each PLD must have no more than 15 ns propagation delay for 16 MHz design.