4. Bus Arbiter and Mode Control
TABLE OF CONTENTS

4. Bus Arbiter and Mode Control

The external arbiter determines the use of the system address/data buses for the rigid disk controller or DMA controller after the IOP or PCE relinquishes the bus. A function called mode control is required to switch the bus between the IOP and the PCE 80186 during PC execution.

The two main tasks of the arbiter/mode control logic are:

1) to arbitrate Ethernet (ENET) and rigid disk controller (RDC) requests;

2) to allow either the IOP 80186 or PCE 80186 to use the A/address and A/data buses.

Figure 4.1 illustrates the data flow of the arbiter and mode control. The signals are described in section 4.1.

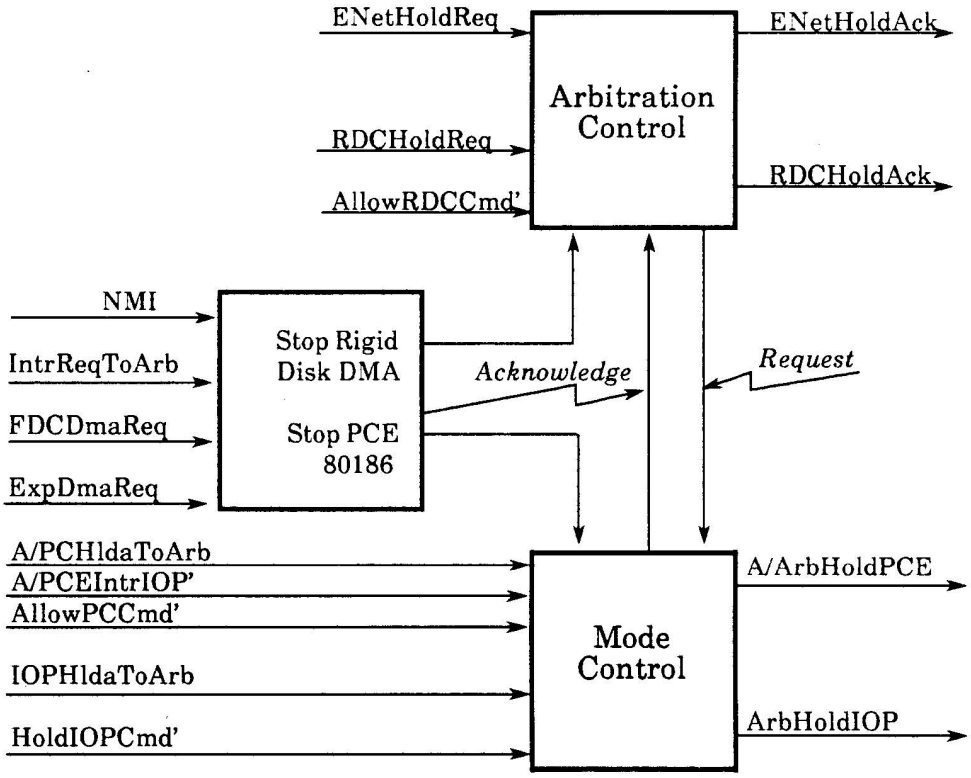

Figure 4.1. Arbitration and mode control block diagram

4.1 Hardware

The arbiter control logic consists of the following components.

- D type flip flops
- JK flip flops
- AND, OR, NOR, NAND, and inverter gates.

Table 4.1 lists the major control signals for the arbiter.

Table 4.1. Major Arbiter Logic Signals

Signal	I/O	Description
MasterRst'	Input	**Master reset.** Reset output of the 80186. This signal establishes a known state of logic.
IOPCLKK' IOPCLK'	Input	**Inverted IOP clock.** Inverted clock output of the 80186. This signal synchronizes the system functions.
IOPHldaToArb	Input	**IOP hold acknowledge to arbiter.** Signals the arbiter that the IOP 80186 is no longer using the system address and data buses.
IOPClk	Input	**IOP Clock.** Clock output of the 80186 synchronizes the system functions.
ENetHoldReq	Input	**Ethernet hold request.** Signal from the 82586 Ethernet controller indicating that it needs the system A/address and A/data buses.
RstENetCtlr'	Input	**Reset Ethernet controller.** IOP 80186 I/O control signal that resets the Ethernet controller and resets the Ethernet hold request latches to the arbiter.
IOPNMI	Input	**IOP non maskable interrupt.** This signal is connected to the 80186 NMI.
IntrReqToArb	Input	**Interrupt request to the arbiter.** This signal is controlled by the master 8259 PIC and signifies that a peripheral device requires the attention of the IOP 80186.
FDCDmaReq	Input	**Floppy disk controller/DMA request.** Signifies that the 8272 floppy disk controller requires DMA servicing from channel 0 of the integrated 80186 DMA controller.
ExpDmaReq	Input	**Expansion DMA request.** Signifies that the an expansion slot device requires DMA servicing from channel 1 of the integrated 80186 DMA Controller.
ADDr'/Data	Input	**Address/data.** Signal from the RDC DMA controller that identifies whether address or data is being placed on the system multiplexed A/D lines, as follows: ADDr'/Data 0 = Address ADDr'/Data 1 = Data
ResetDmaFifo'	Input	**Reset DMA and FIFO.** The IOP 80186 uses this signal to reset the rigid disk FIFO and the RDC DMA state machine.
RDCHoldReq	Input	**RDC hold request.** Signifies that the rigid disk DMA controller needs to use the system A/address and A/data buses.
A/PCEIntrIOP'	Input	**PCE interrupt to IOP.** Signifies that PC emulation requires IOP 80186 service.
A/PCEHldaToArb	Input	**PCE hold acknowledge to arbiter.** Signals the arbiter that the PCE 80186 is using the system A/address bus and A/data bus.

-more-

Table 4.1. Major Arbiter Logic Signals (continued)

Signal	I/O	Description
AllowRDCCmd'	Input	**Allow RDC command.** IOP executes this command to the arbiter by reading the I/O port address F4H, thus signaling the arbiter to begin processing the rigid disk DMA requests.
AllowPCCmd'	Input	**Allow PCE command.** IOP executes this command to the arbiter by reading I/O port address F8H, thus signaling the arbiter to allow the PCE 80186 access to the system A/address and A/data buses.
HoldIOPCmd'	Input	**Hold IOP command.** IOP executes this command to the arbiter by reading I/O port address F2H, thus signaling the arbiter to permanently hold the IOP 80186. This hold is maintained until a peripheral device requires the attention of the IOP 80186.
A/ArbHoldPCE	Output	**Arbiter hold PCE.** Control signal from the arbiter to the PCE 80186 that requests the processor to relinquish the system A/address bus and A/data bus.
RDCHoldAck	Output	**RDC hold acknowledge.** Control signal from the arbiter to the rigid disk DMA controller that the system A/address bus and A/data bus may be used.
ENetHoldAck	Output	**Ethernet hold acknowledge.** Control signal from the arbiter to the Ethernet controller that the system A/address bus and A/data bus may be used.
ArbHoldIOP	Output	**Arbiter hold acknowledge.** Control signal from the arbiter to the IOP 80186 that the system A/address bus and A/data bus may be used.

4.2 Theory of Operations

The arbiter theory of operations is described first by a series of tables listing the events that occur under specific circumstances. The tables, listed below, are followed by a series of flow diagrams, Figures 4.2 - 4.6, illustrating decisions after a service request.

The following tables list 19 scenarios for arbiter operations:

Table 4.2
- Ethernet requests service

Table 4.3
- RDC requests service (AllowRDCCmd' not executed)
- RDC requests service (AllowRDCCmd' executed)
- RDC requests service and interrupt occurs

Table 4.4
- RDC requests service then Ethernet requests service. No interrupt to IOP occurs.

Table 4.5
- RDC and Ethernet request service before an IOP acknowledge occurs

Table 4.6
- RDC and Ethernet request service at the same time
- RDC and Ethernet request service at the same time. An IOP interrupt occurs during Ethernet servicing.

Table 4.7
- RDC and Ethernet request service. An interrupt occurs during the RDC request.
- RDC and Ethernet request service. An interrupt is generated when the Ethernet is using the bus.

Table 4.8
- IOP executes an AllowPCCmd'

Table 4.9
- IOP requests service

Table 4.10
- IOP and PCE request service. PCE executes an I/O Rd' or Wr'

Table 4.11
- IOP, PCE, and RDC request service
- IOP, PCE, and RDC request service and an interrupt occurs

Table 4.12
- IOP, PCE, and ENet request service
- IOP, PCE, and Ethernet request service and an interrupt occurs

Table 4.13
- IOP, PCE, RDC, and Ethernet request service

Table 4.14
- IOP, PCE, and RDC request service. RDC requests service, then Ethernet requests service, then an interrupt occurs.

4.2.1 Ethernet and Rigid Disk Combinations

Arbitration of Ethernet and rigid disk requests for service are treated in this section. Arbitration among Ethernet, rigid disk, PCE and IOP requests are treated in section 4.2.2.

Table 4.2. Arbiter Flow Sequence:
Ethernet requests service

Signal	Sequence of events
	1) IOP 80186 is operating.
ENetHoldReq = 1	2) Ethernet controller sends a hold request to the arbiter.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	4) The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1	5) The arbiter sends a hold acknowledge to the Ethernet controller.
ENetHoldReq = 0	6) The Ethernet controller drops the hold request when it is finished transferring data.
ENetHoldReq = 0 ArbHoldIOP = 0	7) The arbiter drops the Ethernet hold acknowledge and drops the hold request to the IOP 80186.
	8) The IOP 80186 resumes operations.

Table 4.3. Arbiter Flow Sequence

RDC requests service

Allow RDCCmd' not executed	
Signal	**Sequence of events**
RDCHoldReq = 1	1) IOP 80186 is operating. 2) RDC/DMA controller sends a hold request to the arbiter. 3) Arbiter ignores the request because the IOP 80186 did not execute an AllowRDCCmd'.

Allow RDCCmd' executed	
Signal	**Sequence of events**
	1) IOP 80186 is operating.
AllowRDCCmd'	2) IOP 80186 executes an AllowRDCCmd' I/O instruction.
RDCHoldReq = 1	3) RDC/DMA controller sends a hold request to the arbiter.
ArbHoldIOP = 1	4) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	5) The IOP 80186 sends a hold acknowledge to the arbiter.
RDCHoldAck = 1	6) The arbiter sends a hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	7) The RDC/DMA controller drops the hold request when it is finished transferring data.
RDCHoldAck = 0 ArbHoldIOP = 0	8) The arbiter drops the hold acknowledge to the RDC/DMA controller and drops the hold request to the IOP 80186.
	9) The IOP 80186 resumes operations.

RDC requests service and an interrupt occurs	
Signal	**Sequence of events**
	1) IOP 80186 is operating.
RDCHoldReq = 1	2) RDC/DMA controller sends a hold request to the arbiter.
IntrReqToArb = 1	3) The arbiter fields a system interrupt.
ArbHoldIOP = 1	4) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	5) The IOP 80186 sends a hold acknowledge to the arbiter.
RDCHoldAck = 1	6) The arbiter sends a hold acknowledge to the RDC/DMA controller.
RDCHoldAck = 0	7) The RDC/DMA controller begins transferring data. During the current bus cycle, the arbiter drops the RDC/DMA controller hold acknowledge.
RDCHoldReq = 0	8) The RDC/DMA controller drops its hold request to the arbiter.
ArbHoldIOP = 0	9) The arbiter drops its hold request to the IOP 80186.
RDCHoldReq = 1	10) The RDC/DMA controller sends a hold request to the arbiter.
	11) The IOP 80186 resumes execution.
	12) The arbiter ignores the RDC/DMA controller hold request until the IOP 80186 executes an AllowRDCCmd' I/O instruction.

Table 4.4. Arbiter Flow Sequence

RDC requests service then Ethernet requests service
no interrupt to the IOP occurs

Signal	Sequence of events
	1) IOP 80186 is operating.
RDCHoldReq = 1	2) The RDC/DMA controller sends a hold request to the arbiter.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
IOPHoldAck = 1	4) The IOP 80186 sends a hold acknowledge to the arbiter.
RDCHoldAck = 1	5) The arbiter sends a hold acknowledge to the RDC/DMA controller.
	6) The RDC/DMA controller begins transferring data.
ENetHoldReq = 1	7) The Ethernet controller sends a hold request to the arbiter.
RDCHoldAck = 0	8) The arbiter drops the hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	9) The RDC/DMA controller drops the hold request to the arbiter after completing the current transfer cycle.
ENetHoldAck = 1	10) The arbiter issues a hold acknowledge to the Ethernet controller, which then begins to transfer data.
RDCHoldReq = 1	11) The RDC/DMA controller sends a hold request to the arbiter.
ENetHoldReq = 0	12) The Ethernet controller drops the hold request when finished transferring data.
ENetHoldAck = 0	13) The arbiter drops the Ethernet hold acknowledge.
RDCHoldAck = 1	14) The arbiter issues a hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	15) The RDC/DMA controller drops the hold request when finished transferring data.
RDCHoldAck = 0	16) The arbiter drops the hold acknowledge to the RDC/DMA controller.
ArbHoldIOP = 0	17) The arbiter drops the hold request to the IOP 80186.
	18) The IOP 80186 resumes operation.

Table 4.5. Arbiter Flow Sequence

RDC and Ethernet request service
before IOP Acknowledge occurs

Signal	Sequence of events
	1) IOP is operating.
RDCHoldReq = 1	2) The RDC/DMA controller sends a hold request to the arbiter.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
ENetHoldReq = 1	4) The Ethernet controller sends a hold request to the arbiter.
IOPHldaToArb = 1	5) The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1	6) The arbiter sends a hold acknowledge to the Ethernet controller (priority 1); the Ethernet controller transfers data.
ENetHoldReq = 0	7) When data transfer is finished, the Ethernet controller drops its hold request to the arbiter.
ENetHoldAck = 0 RDC HoldAck = 1	8) The arbiter drops the hold acknowledge to the Ethernet controller and sends a hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0 RDCHoldAck = 0	9) The RDC/DMA controller transfers data and then drops its hold request to the arbiter. The arbiter drops its hold acknowledge to the RDC/DMA controller.
ArbHoldIOP = 0	10) The arbiter drops its hold request to the IOP 80186.
	11) The IOP 80186 resumes operations.

Table 4.6. Arbiter Flow Sequence

RDC and Ethernet request service at the same time

Signal	Sequence of events
	1) IOP 80186 is operating.
RDCHoldReq = 1 ENetHoldReq = 1	2) The RDC/DMA controller and the Ethernet controller send a hold request to the arbiter.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	4) The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1	5) The arbiter sends a hold acknowledge to the Ethernet controller (priority 1); the Ethernet controller begins transferring data.
ENetHoldReq = 0	6) The Ethernet controller finishes transferring data and drops the hold request to the arbiter.
ENetHoldAck = 0	7) The arbiter drops the hold acknowledge to the Ethernet controller.
RDCHoldAck = 1	8) The arbiter sends the hold acknowledge to the RDC/DMA controller; the RDC/DMA controller begins transferring data.
RDCHoldReq = 0	9) The RDC/DMA controller finishes transferring data and drops the hold request to the arbiter.
RDCHoldAck = 0 ArbHoldIOP = 0	10) The arbiter drops the hold request to the RDC/DMA controller and then drops the hold request to the IOP 80186.
	11) The IOP 80186 resumes operations.

IOP interrupt occurs during Ethernet servicing

Signal	Sequence of events
	1) IOP is operating.
ENetHoldReq = 1 RDCHoldReq = 1	2) Ethernet and RDC/DMA controllers send a hold request to the arbiter.
ArbHoldIOP = 1 IOPHldaToArb = 1	3) The arbiter sends a hold request to the IOP 80186. The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1 IntrReqToArb = 1	4) The arbiter sends a hold acknowledge to the Ethernet controller (priority 1). The Ethernet controller transfers data. During this data transfer, an IOP interrupt is generated.
ENetHoldReq = 0 ENetHoldAck = 0	5) The Ethernet drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the Ethernet.
RDCHoldAck = 1 RDCHoldAck = 0	6) The arbiter sends a hold acknowledge to the RDC/DMA controller; the RDC/DMA controller begins data transfer. During the current bus cycle, the arbiter drops the hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	7) The RDC/DMA controller completes the current bus cycle, then drops the hold request to the arbiter.
ArbHoldIOP = 0	8) The arbiter drops the hold request to the IOP 80186.
RDCHoldReq = 1	9) The RDC/DMA controller sends a hold request to the arbiter.
	10) The IOP 80186 resumes operations.
	11) The arbiter ignores the RDC/DMA controller hold request until the IOP 80186 executes an AllowRDCCmd' I/O instruction.

Table 4.7. Arbiter Flow Sequence

Ethernet and RDC request service

Interrupt occurs during RDC request

Signal		Sequence of events
	1)	IOP is operating.
ENetHoldReq = 1	2)	The Ethernet controller sends a hold request to the arbiter.
ArbHoldIOP = 1 IOPHldaToArb = 1	3)	The arbiter sends a hold request to IOP 80186; the IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1 RDCHoldReq = 1	4)	The arbiter sends a hold acknowledge to the Ethernet controller. The Ethernet transfers data. During this data transfer, the RDC/DMA controller sends a hold request to the arbiter.
ENetHoldReq = 0 ENetHoldAck = 0 RDCHoldAck = 1	5)	When the Ethernet finishes transferring data, it drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the Ethernet controller and sends a hold acknowledge to the RDC/DMA controller.
IntrReqToArb = 1	6)	The RDC/DMA controller begins transferring data. During this data transfer, an IOP interrupt request is generated.
RDCHoldAck = 0	7)	The arbiter drops the RDC/DMA controller hold acknowledge.
RDCHoldReq = 0	8)	The RDC/DMA controller completes the current bus cycle then drops the hold request to the arbiter.
ArbHoldIOP = 0	9)	The arbiter drops the hold request to the IOP 80186.
RDCHoldReq = 1	10)	The RDC/DMA controller sends a hold request to the arbiter.
	11)	The IOP 80186 resumes operations.
	12)	The arbiter ignores the RDC/DMA controller hold request until the IOP 80186 executes an AllowRDCCmd' I/O instruction.

Interrupt is generated when the Ethernet is using the bus

Signal		Sequence of events
	1)	IOP 80186 is operating.
RDCHoldReq = 1	2)	The RDC/DMA controller sends a hold request to the arbiter.
ArbHoldIOP = 1	3)	The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	4)	The IOP 80186 sends a hold acknowledge to the arbiter.
RDCHoldAck = 1	5)	The arbiter sends a hold acknowledge to the RDC/DMA controller.
	6)	The RDC/DMA controller begins transferring data.
ENetHoldReq = 1	7)	The Ethernet controller sends a hold request to the arbiter.
RDCHoldAck = 0	8)	The arbiter drops the hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	9)	The RDC/DMA controller drops the hold request to the arbiter after the current bus cycle is complete.
ENetHoldAck = 1	10)	The arbiter issues a hold acknowledge to the Ethernet controller, which then begins to transfer data.
RDCHoldReq = 1	11)	The RDC/DMA controller sends a hold request to the arbiter.
ENetHoldReq = 0	12)	The Ethernet controller drops the hold request when finished transferring data.
IntrReqToArb = 1	13)	Before the Ethernet controller drops the hold request, the arbiter fields a system interrupt.
ENetHoldAck = 0 RDCHoldAck = 1	14)	The arbiter drops the hold acknowledge to the Ethernet controller and issues a hold acknowledge to the RDC/DMA controller.
RDCHoldAck = 0	15)	The RDC/DMA controller performs one bus cycle; during the RDC transfer the arbiter drops the hold acknowledge.
RDCHoldReq = 0	16)	At the end of a current bus cycle, the RDC/DMA controller drops the hold request to the arbiter.
ArbHoldIOP = 0	17)	The arbiter drops the hold request to the IOP 80186.
RDCHoldReq = 1	18)	The RDC/DMA controller sends a hold request to the arbiter.
	19)	The IOP resumes operation.
	20)	The arbiter ignores the RDC hold request until the IOP 80186 executes an AllowRDCCmd' I/O instruction.

4.2.2 IOP, PCE, Ethernet, and Rigid Disk Combinations

Arbitration of IOP and PCE requests for service, alone and in combination with each other, Ethernet, and RDC, are presented in this section.

Table 4.8. Arbiter Flow Sequence
IOP executes AllowPCCmd'

Signal	Sequence of events
	1) IOP 80186 is operating.
AllowPCCmd'	2) IOP executes an Allow PCCmd' I/O instruction.
ArbHoldIOP = 1 IOPHldAToArb = 1	3) The arbiter sends a hold request to the IOP 80186. The IOP sends a hold acknowledge to the arbiter.
ArbHoldPCE = 0 PCHldaToArb = 0	4) The arbiter drops the hold request to the PCE 80186. PCE 80186 drops the hold acknowledge to the arbiter.
	5) PCE 80186 executes instructions.
IntrReqToArb = 1 ArbHoldPCE = 1	6) Arbiter senses an IOP 80186 interrupt pending and sends an arbiter hold request to the PCE 80186.
PCHldaToArb = 1	7) The PCE 80186 sends a hold acknowledge to the arbiter.
ArbHoldIOP = 0	8) The arbiter drops the hold request to the IOP 80186.
	9) The IOP 80186 resumes operations.

Table 4.9. Arbiter Flow Sequence
IOP requests service

Signal	Sequence of events
	1) The IOP 80186 is operating.
HoldIOPCmd'	2) The IOP 80186 executes HoldIOPCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	4) The IOP 80186 sends a hold acknowledge to the arbiter.
IntrReqToArb = 1	5) The arbiter keeps the IOP in hold state until it senses an IOP interrupt pending.
ArbHoldIOP = 0 IOPHldaToArb = 0	6) The arbiter drops the hold request to the IOP 80186. The IOP 80186 drops the hold acknowledge to the arbiter and resumes operations.

Table 4.10. Arbiter Flow Sequence
IOP-PCE: (PCE executes I/O Rd' or Wr')

Signal	Sequence of events
	1) IOP 80186 is operating.
AllowPCCmd'	2) IOP 80186 executes an AllowPCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	4) The IOP 80186 sends a hold acknowledge to the arbiter.
ArbHoldPCE = 0	5) The arbiter drops the PCE 80186 hold request.
PCHldaToArb = 0	6) The PCE 80186 drops the hold acknowledge to the arbiter.
	7) The PCE 80186 begins executing instructions.
	8) The PCE 80186 executes I/O Rd' or Wr'.
A/PCEIntrIOP' = 0	9) The PCE 80186 generates an interrupt to the IOP 80186.
ArbHoldPCE = 1	10) The arbiter sends a hold request to PCE 80186.
A/PCHldaToArb = 1	11) The PCE 80186 sends a hold acknowledge to the arbiter.
ArbHoldIOP = 0	12) The arbiter drops the hold request to the IOP 80186.
IOPHldaToArb = 0	13) The IOP 80186 drops the hold acknowledge to the arbiter.
	14) IOP 80186 resumes operations.

Table 4.11. Arbiter Flow Sequence

IOP, PCE, and RDC request service

Signal	Sequence of events
	1) IOP 80186 is operating.
AllowPCCmd'	2) IOP 80186 executes an AllowPCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
RDCHoldReq = 1	4) The RDC/DMA controller sends a hold request to the arbiter.
IOPHldaToArb = 1 RDCHoldAck = 1	5) The IOP 80186 sends a hold acknowledge to the arbiter. The arbiter sends a hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0 RDCHoldACK = 0	6) The RDC transfers data. When it is finished, the RDC/DMA controller drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the RDC/DMA controller.
ArbHoldPCE = 0 PCHldaToArb = 0	7) The arbiter drops the hold request to the PCE 80186; the PCE 80186 drops the hold acknowledge to the arbiter.
	8) The PCE 80186 begins executing and will not stop until the arbiter senses an IOP 80186 interrupt pending or PCE 80186 executes I/O Rd' or Wr' instruction.

IOP, PCE, and RDC request service and an interrupt occurs

Signal	Sequence of events
	1) The IOP 80186 is operating.
AllowPCCmd'	2) The IOP 80186 executes an Allow PCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
RDCHoldReq = 1	4) The RDC/DMA controller sends a hold request to the arbiter.
IntrReqToArb = 1	5) The arbiter senses an IOP 80186 interrupt pending.
IOPHldaToArb = 1	6) The IOP 80186 sends a hold acknowledge to the arbiter.
RDCHoldAck = 1	7) The arbiter sends a hold acknowledge to the RDC/DMA controller.
	8) The RDC/DMA controller begins transferring data.
RDCHoldAck = 0	9) The arbiter drops the hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	10) The RDC/DMA controller completes the current bus cycle and drops the hold request to the arbiter.
ArbHoldPCE = 0	11) The arbiter drops the hold request to the PCE 80186.
RDCHoldReq = 1	12) The RDC/DMA controller sends a hold request to the arbiter. The arbiter ignores this request until the IOP 80186 executes an Allow RDCCmd' I/O instruction.
PCHldaToArb = 0	13) The PCE 80186 drops the hold acknowledge to the arbiter and the PCE 80186 begins executing.
ArbHoldPCE = 1	14) The arbiter sends a hold request to the PCE80186.
A/PCEHldaToArb = 1 ArbHoldIOP = 0 IOPHldaToArb = 0	15) The PCE 80186 sends a hold acknowledge to the arbiter; the arbiter drops the hold request to the IOP 80186. The IOP 80186 drops the hold acknowledge to the arbiter.
	16) The IOP 80186 resumes operations

Table 4.12. Arbiter Flow Sequence

IOP, PCE, and Ethernet request service

Signal	Sequence of events
	1) IOP is operating.
AllowPCCmd'	2) The IOP 80186 executes an AllowPCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
ENetHoldReq = 1	4) The Ethernet controller sends a hold request to the arbiter.
IOPHldaToArb = 1	5) The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1	6) The arbiter sends a hold acknowledge to the Ethernet controller.
ENetHoldReq = 0 ENetHoldAck = 0	7) The Ethernet controller transfers data. When it is finished, the Ethernet controller drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the Ethernet controller.
ArbHoldPCE = 0	8) The arbiter drops its hold request to the PCE 80186.
PCHldaToArb = 0	9) The PCE 80186 drops its hold acknowledge and begins executing and continues executing until the arbiter senses that the IOP 80186 has an interrupt pending or PCE executes I/O Rd' or Wr'.

IOP, PCE, and ENet request service and an interrupt occurs

Signal	Sequence of events
	1) IOP 80186 is operating.
AllowPCCmd'	2) IOP 80186 executes an AllowPCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
ENetHoldReq = 1	4) The Ethernet controller sends a hold request to the arbiter.
IntrReqToArb = 1	5) The arbiter fields a system interrupt.
IOPHldaToArb = 1	6) The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1 ENetHoldReq = 0 ENetHoldAck = 0	7) The arbiter sends a hold acknowledge to the Ethernet controller. The Ethernet transfers data and then drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the Ethernet controller.
ArbHoldPCE = 0 PCHldaToArb = 0	8) The arbiter drops the hold request to the PCE 80186. The PCE 80186 drops the hold acknowledge to the arbiter.
	9) PCE 80186 begins executing.
ArbHoldPCE = 1	10) The arbiter sends a hold request to the PCE 80186.
A/PCHldaArb = 1	11) The PCE 80186 sends a hold acknowledge to the arbiter.
ArbHoldIOP = 0 IOPHldaToArb = 0	12) The arbiter drops the hold request to the IOP 80186. The IOP 80186 drops the hold acknowledge to the arbiter.
	13) The IOP resumes operations.

Table 4.13. Arbiter Flow Sequence

IOP, PCE, RDC, and Ethernet request service

Signal	Sequence of events
	1) IOP is operating.
AllowPCCmd'	2) IOP 80186 executes an AllowPCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
RDCHoldReq = 1 ENetHoldReq = 1	4) The Ethernet and the RDC/DMA controllers send a hold request to the arbiter.
IOPHldaToArb = 1	5) The IOP 80186 sends a hold acknowledge to the arbiter.
ENetHoldAck = 1	6) The arbiter sends a hold acknowledge to the Ethernet controller.
ENetHoldReq = 0 ENetHoldAck = 0	7) The Ethernet controller transfers data and then drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the Ethernet controller.
RDCHoldAck = 1 RDCHoldReq = 0 RDCHoldAck = 0	8) The arbiter sends a hold acknowledge to the RDC/DMA controller. The RDC/DMA controller transfers data and then drops the hold request to the arbiter. The arbiter drops the hold acknowledge to the RDC/DMA controller.
ArbHoldPCE = 0 PCHldaToArb = 0	9) The arbiter drops the hold request to the PCE 80186. The PCE 80186 drops the hold acknowledge to the arbiter.
	10) The PCE 80186 begins executing and will continue executing until the arbiter senses that IOP 80186 has an interrupt pending or PCE executes I/O Rd' or Wr'.

Table 4.14. Arbiter Flow Sequence

IOP, PCE, RDC request service

RDC hold request then Ethernet hold request then interrupt.

Signal	Sequence of events
	1) The IOP 80186 is operating.
AllowPCCmd'	2) The IOP 80186 executes an AllowPCCmd' I/O instruction.
ArbHoldIOP = 1	3) The arbiter sends a hold request to the IOP 80186.
IOPHldaToArb = 1	4) The IOP 80186 sends a hold acknowledge to the arbiter.
ArbHoldPCE = 0	5) The arbiter drops the hold request to the PCE 80186.
PCHldaToArb = 0	6) The PCE 80186 drops the hold acknowledge to the arbiter.
	7) The PCE 80186 begins operating.
RDCHoldReq = 1	8) The RDC/DMA controller sends a hold request to the arbiter.
A/ArbHoldPCE = 1	9) The arbiter sends a hold request to the PCE 80186.
A/PCHldaToArb = 1	10) The PCE 80186 sends a hold acknowledge to the arbiter.
RDCHoldAck = 1	11) The arbiter sends a hold acknowledge to the RDC/DMA controller, which then begins transferring data.
ENetHoldReq = 1	12) The Ethernet controller sends a hold request to the arbiter.
RDCHoldAck = 0 RDCHoldReq = 0	13) The arbiter drops the hold acknowledge to the RDCDMA controller. The RDC/DMA controller completes the current bus cycle and drops the hold request to the arbiter.
ENetHoldAck = 1 ENetHoldReq = 0	14) The arbiter sends a hold acknowledge to the Ethernet controller, which transfers data and then drops the hold request to the arbiter.
IntrReqToArb = 1 ENetHoldAck = 0	15) The arbiter senses an IOP 80186 interrupt pending and drops the hold acknowledge to the Ethernet controller.
RDCHoldAck = 1	16) The arbiter sends a hold acknowledge to the RDC/DMA controller, which then resumes transferring.
RCHHoldAck = 0	17) The arbiter drops the hold acknowledge to the RDC/DMA controller.
RDCHoldReq = 0	18) The RDC/DMA controller completes the current bus cycle and drops the hold register to the arbiter.
ArbHoldPCE = 0 PCHldaToArb = 0	19) The arbiter drops the hold request to PCE 80186; the PCE 80186 drops the hold acknowledge to the arbiter.
	20) PCE 80186 resumes operations.
A/ArbHoldPCE = 1 A/PCEHldaArb = 1	21) The arbiter sends a hold request to the PCE 80186. PCE 80186 sends a hold acknowledge to the arbiter.
ArbHoldIOP = 0	22) The arbiter drops the hold request to the IOP 80186.
	23) The IOP 80186 resumes operations.

4.2.3 Arbiter Flow Diagrams

Figures 4.2 - 4.6 illustrate what happens when the arbiter receives one of the service requests described in the preceding sections.

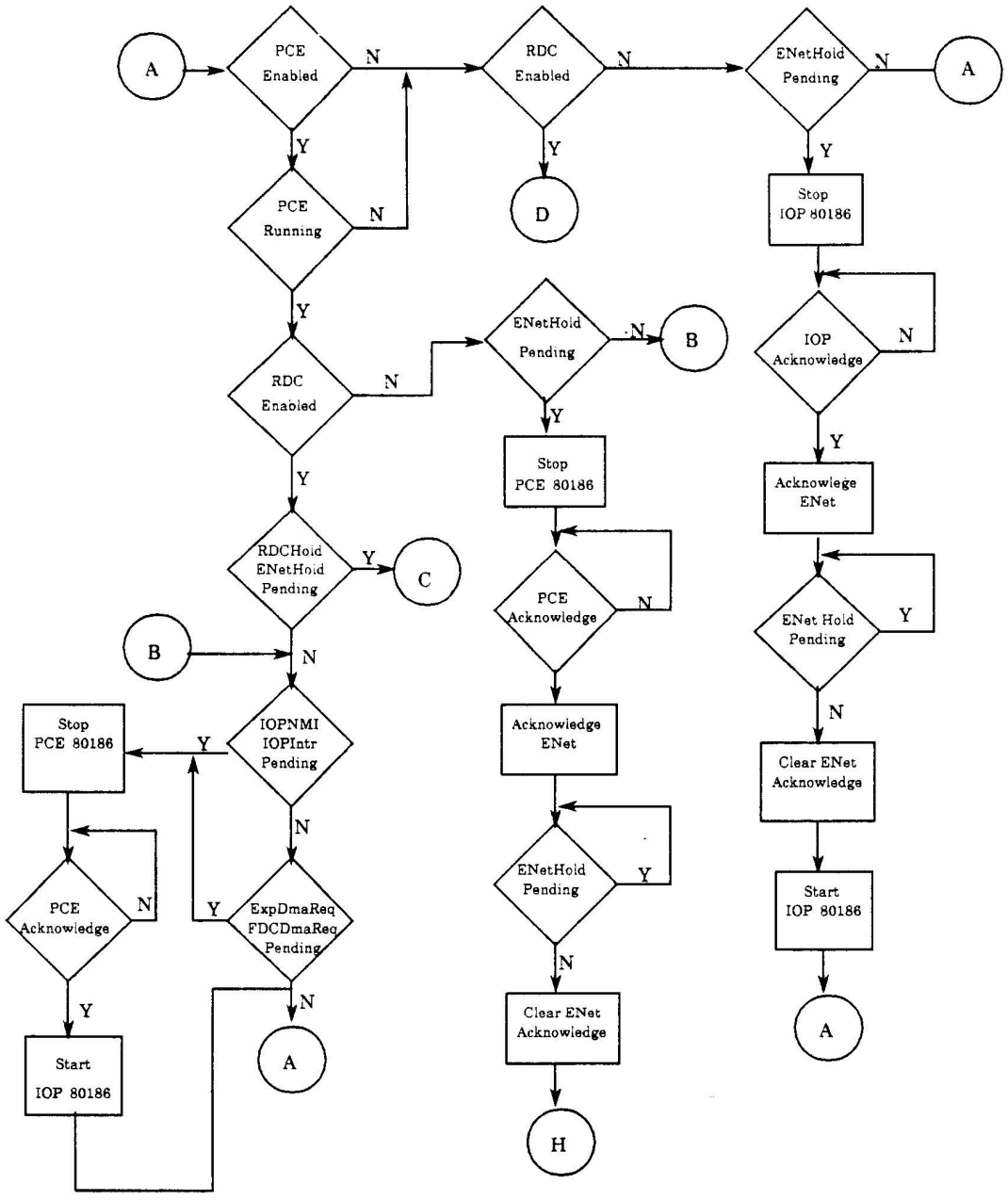

Figure 4.2. Arbiter flow diagram: service request

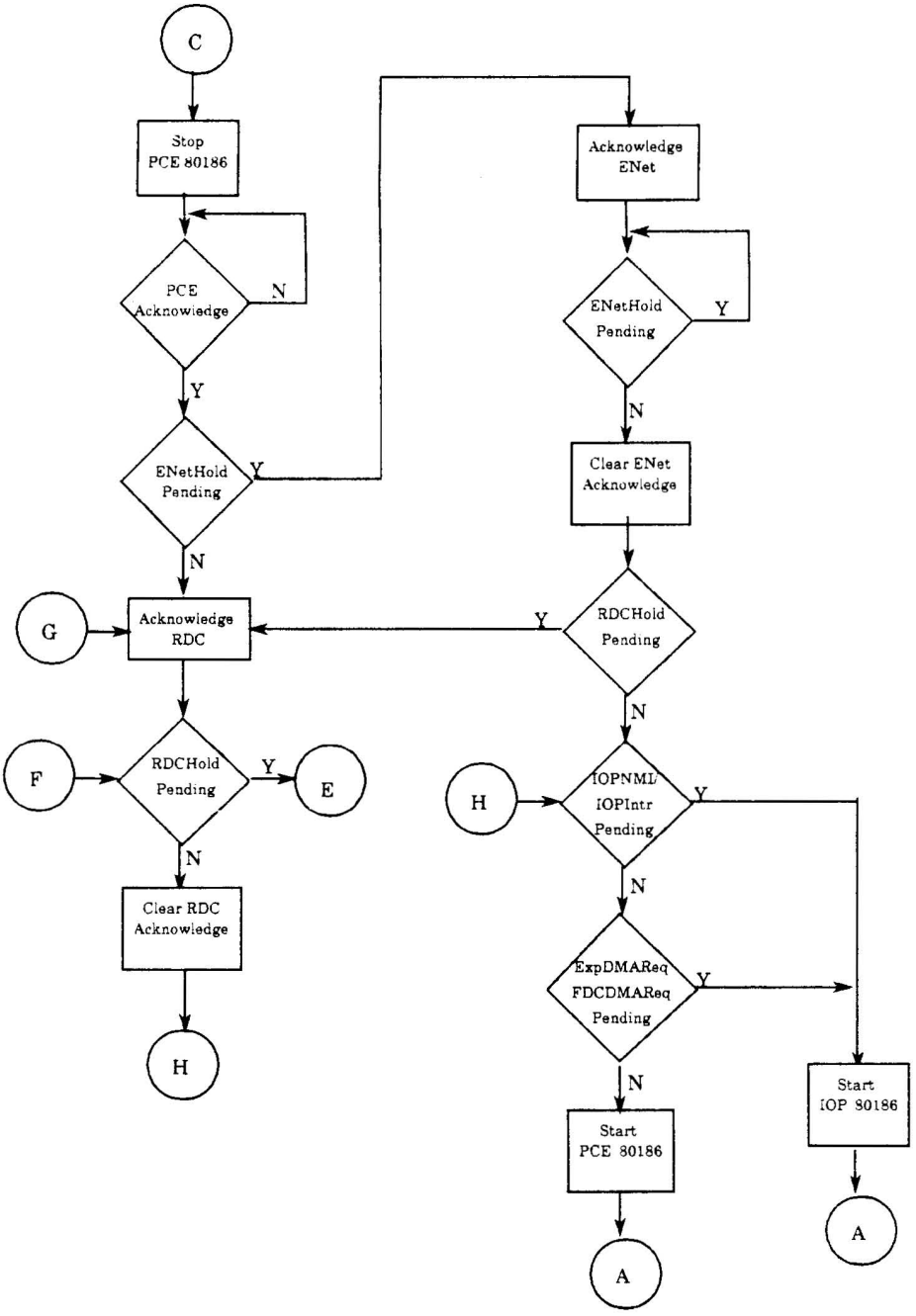

Figure 4.3. Arbiter flow diagram: service request (continued)

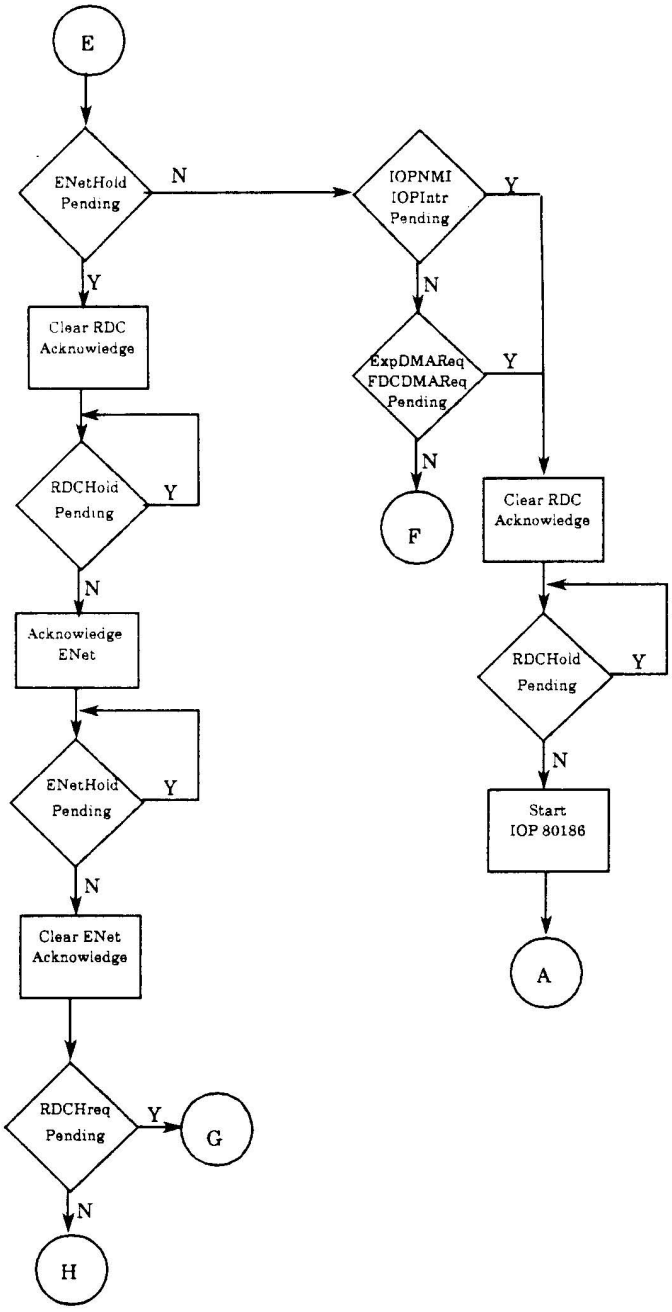

Figure 4.4. Arbiter flow diagram: service request (continued)

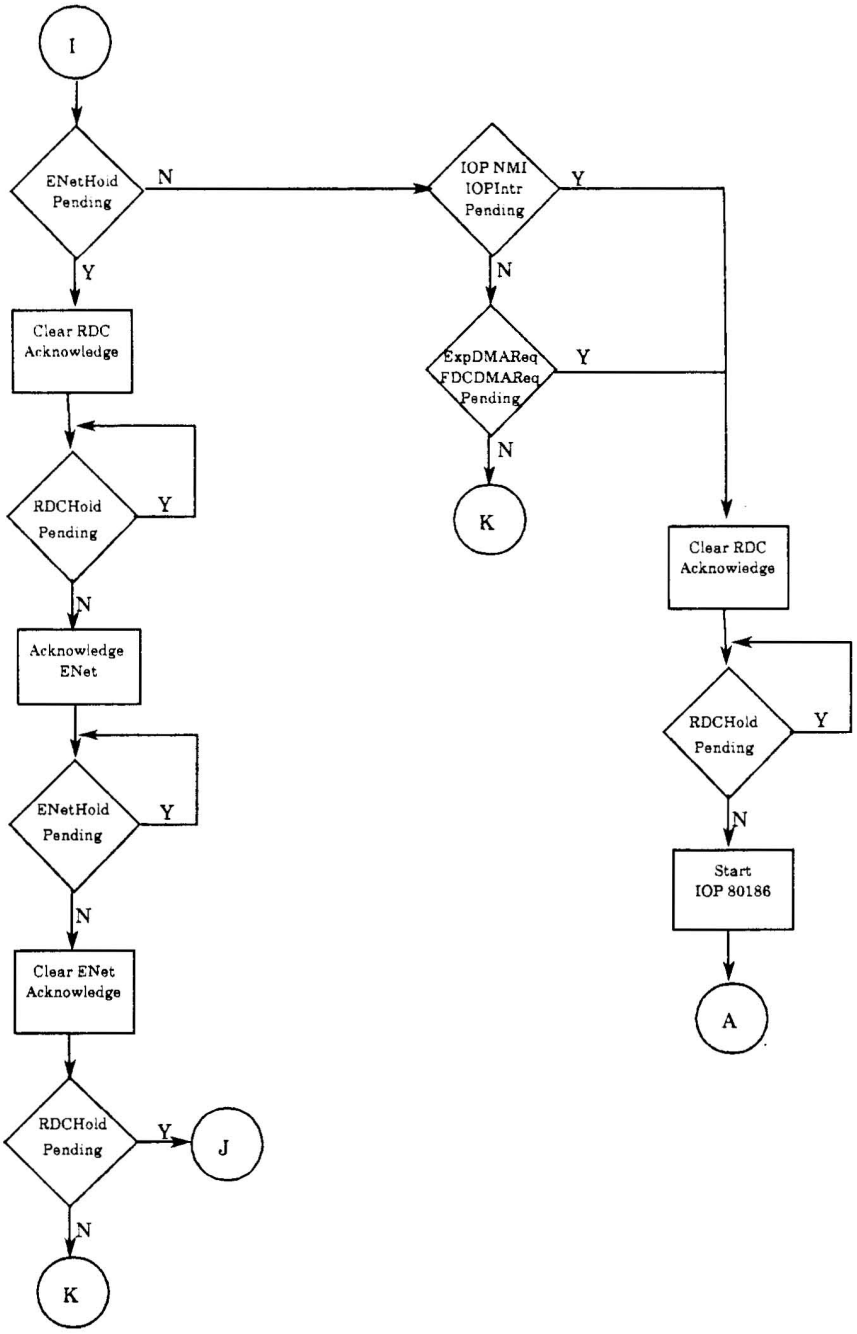

Figure 4.5. Arbiter flow diagram: service request (continued)

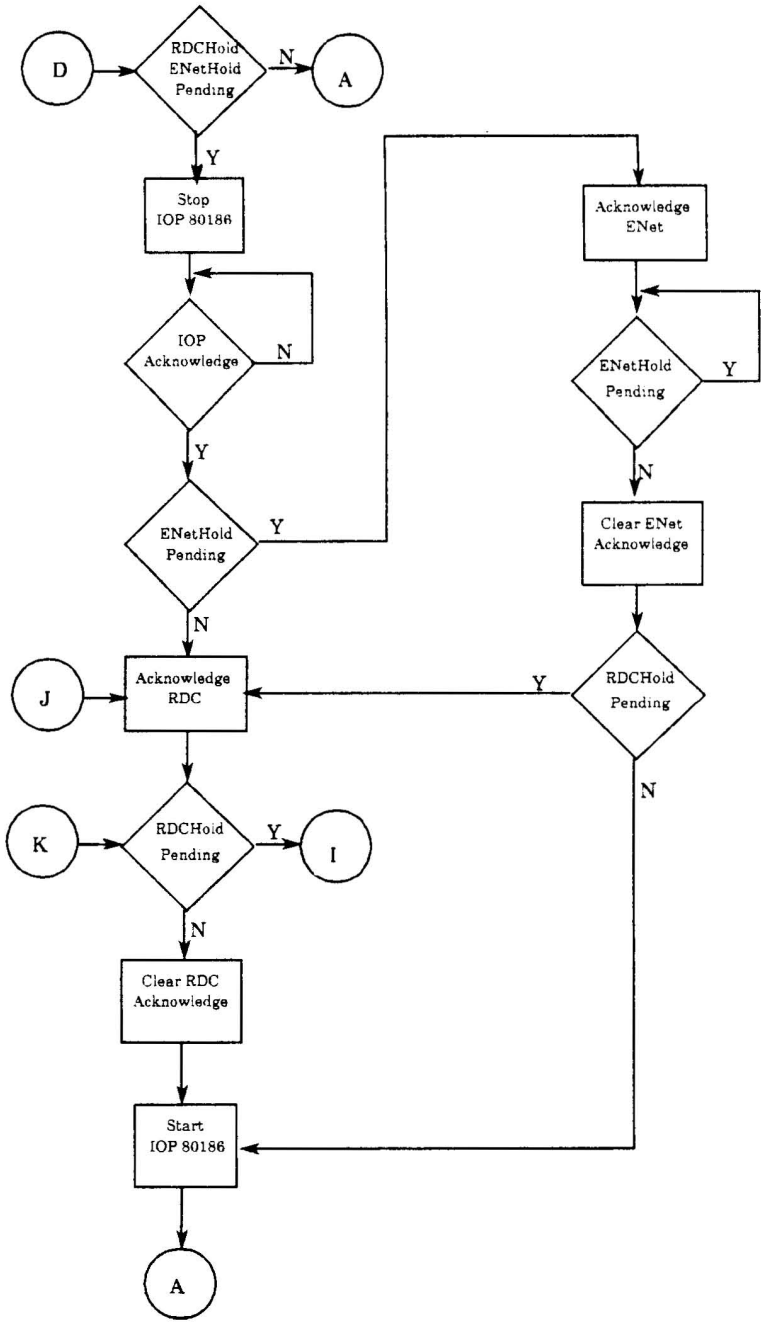

Figure 4.6. Arbiter flow diagram: service request (continued)