XEROX

XEROX
Private
Data

Dove IOP Review

Rigid Disk Subsystem

June 14, 1984

Xerox Corporation
Systems Development Department
3450 Hillview Avenue
Palo Alto, California 94304

Xerox Private Data

Contents: Dove IOP Rigid disk Subsystem

1 Overview

The rigid disk subsystem in the IOP provides support for rigid disk operation on the Dove workstation. The rigid disk is used on the workstation for the following functions:

Local permanent file storage

Virtual memory swapping for Pilot operating system

System booting

The subsystem supports labels and the various disk operations required by the Pilot operating system. This requires the use of a customized controller. The main components of the rigid disk subsystem are: the disk drive, the rigid disk controller (RDC), the DMA controller, and the rigid disk FIFO (see Figure 1).

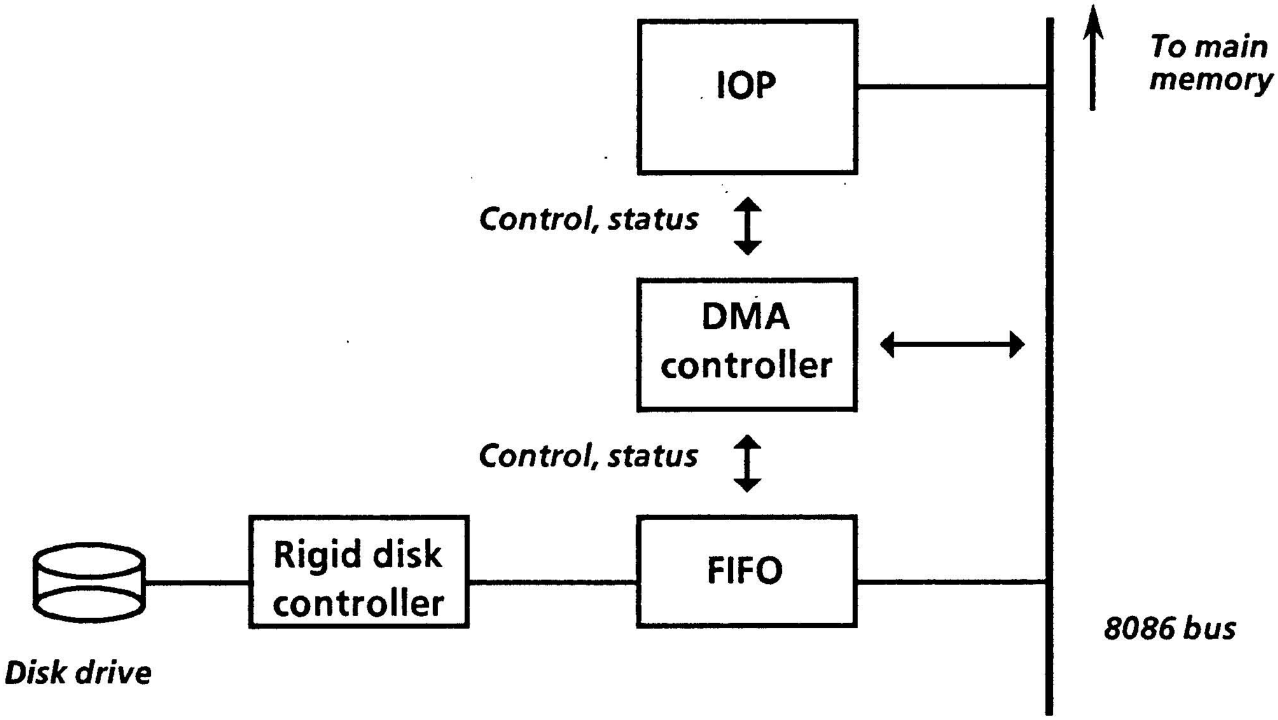


Figure 1. Rigid disk subsystem block diagram

1.1 Rigid disk drives

Only a single drive is supported, either half- or full height. The performance requirements of the drive should meet or exceed the Shugart SA1004 performance characteristics. The drive is a 5$\frac{1}{4}$" Winchester unit with the ST412/ST 506 interface. The data rate is 5 Mbps. The nominal unformatted capacities are 10 MBytes (baseline); 20, 40, and 80 MBytes (optional). Potential drives that can be used (together with their unformatted capacities) are:

Seagate ST 212 (12.8 MB), Quantum 520, 540 (21 MB, 42 MB),

Micropolis 1303 (35 MB), Atasi 3075 (75 MB), Maxstor XT-2085 (89 MB).

1.2 Rigid disk controller

The rigid disk controller provides direct control of the drive. It is implemented using an 8X305 microcontroller and a Western Digital rigid disk controller chip set. The 8X305 is a fast 8-bit microcontroller with a 256 byte scratchpad memory and a 1Kx24 prom control store. All RDC components are off-the-shelf components. The support of the non-industry-standard format and operations are implemented in the controller microcode. The controller will work with any 5¼" drive which supports the ST412/ ST506 interface. Control, data, and status blocks are transferred between main memory and the rigid disk controller using the DMA controller and FIFO. The controller can read, write, or verify any number of contiguous sectors; it will switch heads if necessary. The commands supported are: Format,. Verify, Write Data, Write Label, Read Data, Read Label, Check Sector, Read Next Sector.

1.3 DMA controller

The function of the DMA controller is to effect the direct data transfers between the rigid disk controller, FIFO. and main memory. The DMA controller is an 8086 bus master, and completes the data transfer in a single bus cycle. Other features are: it is programmable by the IOP (Starting Address, Word Count, and Transfer Direction), it can transfer up to 256 words (of 16 bits) at a time, it provides 24 bits of address to the main memory controller, it can transfer data at the full '186 bus rate, and it provides an end-of-transfer interrupt to the IOP.

1.4 Rigid disk FIFO

The function of the rigid disk FIFO is to buffer the data between the rigid disk controller and main memory. This is needed in order to isolate the processor-main memory subsystem from the inherent latency characteristics of the rigid disk. The FIFO is 512 words long (2 sectors), and supports bidirectional access (memory-to-disk and disk-to-memory). It is simultaneously and asynchronously accessible by the DMA and RDC.

2 Rigid disk drive requirements

2.1 Overview

This section describes a 5.25 inch Winchester drive with 3 bits of Head Select (8 heads max), four Drive Select lines (four drives max), 5 MBits/sec transfer rate, where the data encoding and decoding are done by the controller. These drives use dc spindle and head motors, so no drive problems with varying line voltages and frequencies occur-as long as the power supply can handle them.

One departure from the specifications is allowed, 4 bits of Head Select. This gives 16 heads maximum, as long as the drive does not require an external Reduce Write Current signal. The bit normally used for Reduce Write Current is reassigned to Head Select 3.

Spindle speed is required not to vary by more than 1% from nominal over the entire range of specified operating conditions.

Seek speed is important in this application. Our drives should seek at least as fast as a SA1004 on Dandelion.

These requirements are not difficult; most drives studied will pass. The allowance for 16 heads permits the possible use of 191 MByte (unformatted), 156 MByte (formatted) drives. Maxtor lists these drives in its current catalog.

Mini-Micro Systems magazine, April 19, 1984, lists 31 manufacturers of 124 models of 5.25" drives which might meet these specifications.

2.2 Rigid disk controllers

2.2.1 Why we do it the way we do

An obvious question is raised here. Why were Winchester controllers and controller chips with much higher levels of integration than this controller, and which are available from several companies not used?

The answer is simple. Labels. No highly integrated controller could be found to handle labels in the required way. Furthermore, a system interface which works much better in Dove than the commercially available controllers and controller chips was developed.

2.2.2 Format

The format of a disk is the pattern written on the disk which defines the sector numbers, in our case labels the sector, and provides a defined space for the data.

The disks of a drive are divided into Cylinders. A cylinder is the surface of the disks swept by all the heads as the disk revolves, without moving the heads. A cylinder is further divided into Tracks. A track is the surface of a disk swept by one head as the disk revolves, without moving the heads. A Sector is a subdivision of a track. In a soft-sectored drive, as is used in this design, each sector begins with a Header Block. The header marks the beginning of a sector and uniquely numbers the sector. In our case, each sector has a unique combination of cylinder number, track number within the cylinder (head number), and sector number within that track.

Each sector is divided into three Blocks; a Header Block, a Label Block, and a Data Block. Each block has the same parts: a Synch field, an Address Mark byte, a Block ID byte (the Address Mark Byte and the Block ID byte together are frequently called the Address Mark), a Data field, a CRC or ECC field, and a Guard field. Each block may be written at a different time; however, once a block is written, the remaining blocks in that sector must be written. The header block can only be written when all headers of a track are written. In this controller there is no "Write Header" command. Headers may only be written using the "Format" command, and the minimum length for a format command is one track.

The following remarks apply to Dove and to those Dandelions with 8" rigid disk drives. Other systems may be organized differently.

A track extends from the leading edge of the Index Pulse to the next leading edge of an Index Pulse. Following the Index pulse is a gap of 50 bytes, the Index Gap. This gap is filled with zeros, as are all

Dove Rigid Disk Layout

Robert Swenson June 6, 1984

Index — 16 sectors per track — Index

Sector 1 — Sector 2

Gap 1	Header Block 1	Gap 2	Label Block 1	Gap 3	Data Block 1	Gap 4	Header Block 2	Gap 2	Label Block 2	Gap 3	Data Block 2	Gap 4	Header Block 3	Gap 2	Data Block n	Gap 5
50	~~24~~	6	40	6	534	12	~~24~~	6	40	6	534	12	~~24~~	6	534	

Bytes

HEADER BLOCK:

Synch	AM	Data	CRC	Guard	
14	2	4	2	2	= 24 bytes

Bytes

LABEL BLOCK:

Synch	AM	Data	CRC	Guard	
14	2	20	2	2	= 40 bytes

Bytes

DATA BLOCK:

Synch	AM	Data	ECC	Guard	
14	2	512	4	2	= 534 bytes

Bytes

NOTES:

Nominal Track length = 10416 bytes
1 minimum track = 10311 bytes (nominal - 1%)

1 track = Gap 1 + [Header + Gap2 + Label + Gap3 + DataField] * 16 + Gap4 * 15 + Gap5
= 50 + [610] * 16 + 12 * 15 + Gap5
= 50 + 9760 + 180 + Gap5 = 9990 + Gap5

Since a nominal revolution = 10416 bytes, Gap5 (Nominal) = 426 bytes
Since a minimum revolution = 10311 bytes. Gap5 (Minimum) = 321 bytes
(Gap5 must be at least Gap4 + 1% of the track length excluding Gap5 = 111 bytes)

Figure 1

Filed:[Iris]<Swenson>Format>DvRDLayout.sil gaps. This gap allows for the format operation to overrun the Index pulse by a few bytes. It also provides room for the read amplifiers to recover after switching heads.

The first sector follows the Index gap. The composition of this sector has been described. A few notes: The sync field gives room for the Phase Locked Loop to lock. The Guard field prevents problems caused by ending write current too close to the CRC field. The gap between blocks provides time to set up the next operation and allows for variations in disc speed. The gap between one data block and the next header is long enough to allow write current to be turned off without bothering the next header. This takes in to account all possible variations in disk speed. The gap at the end of the track provides enough space so that a track formatted at the slowest allowable disk speed will fit completely between index pulses.

2.2.3 Figures

Note: for all drawings in this section see attached Sil drawings.

2.2.3.1 Rigid Disk Controller Block Diagram (See Figure RD-1.)

This diagram illustrates the rigid disk controller. In the center is the 8x305 MicroController, which controls the Rigid Disk Controller. To the left, the communication with the IOP is shown, and to the right, the disc control hardware.

Command & Status Registers

The Command and Status registers are each 8 bits long. The IOP loads the Command register, and the 305 reads it. The 305 loads the Status register and the IOP reads it. The 305 can also interrupt the IOP. There is no interrupt for the 305. These two registers are only used to permit the IOP and the 305 to keep together using a protocol. The IOP gives the 305 commands such as "Get the Command Block now in the FIFO"; "Execute the Command Block you now have"; "Load the Ending Status Block into the FIFO"; "Reset the Controller". The Command Blocks and the Ending Blocks, as well as data, are passed to and from the Controller via the FIFO and the DMA.

FIFO and DMA

The FIFO and the DMA will be discussed in detail under that heading. The Controller is 8 bits wide while the FIFO is 16 bits wide. Therefore, the controller uses two 8-bit registers to read the FIFO and another two 8-bit registers to load it. One of the I/O commands for each set of registers just reads/writes its register, while the other, in addition to reading/writing, causes the FIFO to present the next 16 bit word to the output registers or to load the input registers into the FIFO.

Scratch Pad

The Scratch Pad local memory is shown in the upper right part of Figure RD 1. The Scratch Pad stores the image of the header and label and other information needed by the controller. It is discussed in more depth under Figure RD 2.

Disk Write and Read Logic

On Figure RD 1 the write and read Logic is shown on the right, below the Scratch Pad local memory. The write and read logic is covered more fully under Figure RD 3. At this time, notice the Controller

RIGID DISK CONTROLLER
BLOCK DIAGRAM

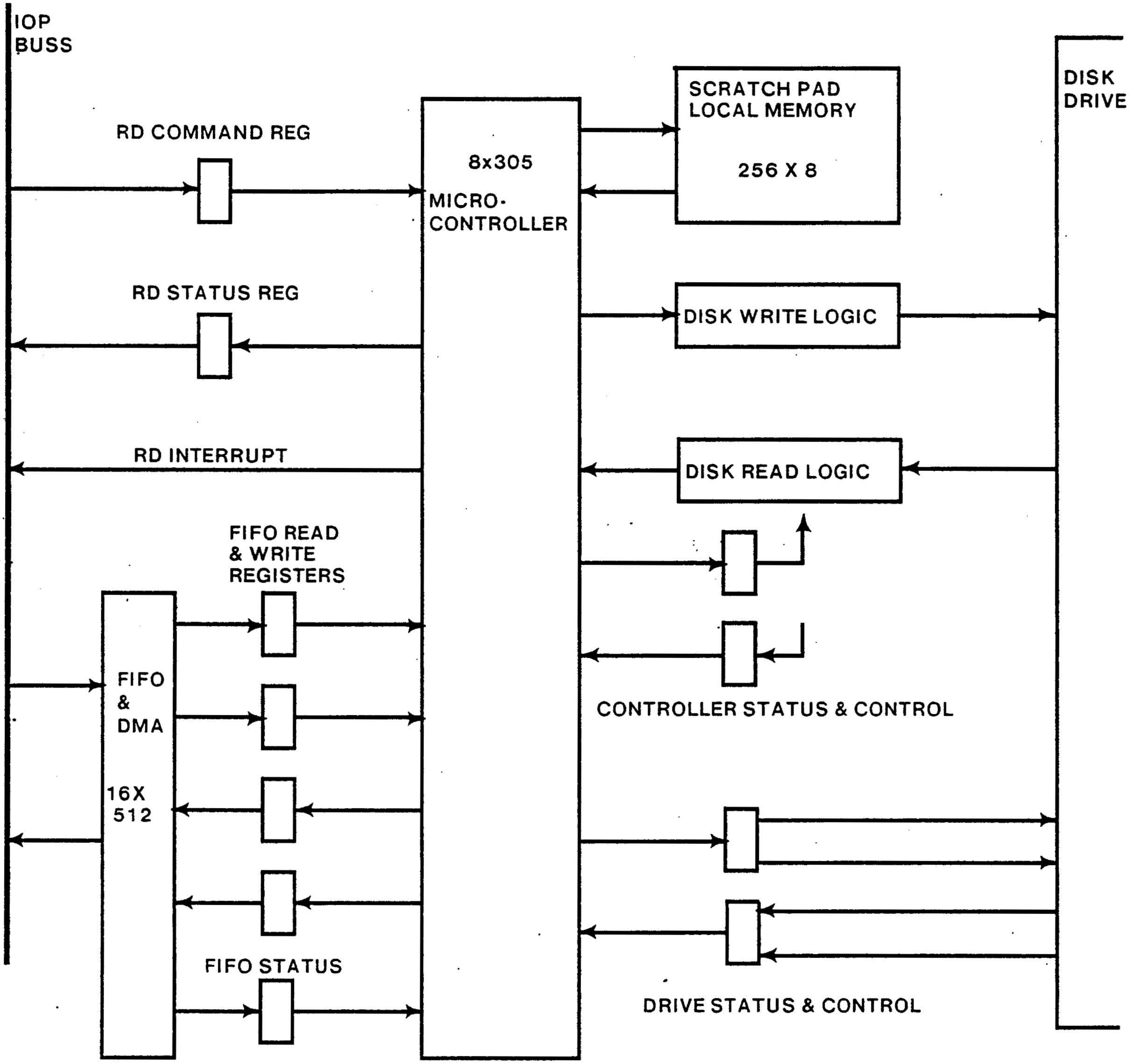


FIGURE RD 1

RDILL1.sil

R Swenson 6/4/84

Status and Control registers. As their names imply, these registers monitor the status of and control the operation of the read and write logic.

Drive Status and Control Registers

At the right bottom of Figure RD 1, the Drive Status and Drive Control registers are shown. With these registers, the 8x305 controlls (with signals Head Select, Drive Select, StepDirection In, Step, WriteEnable, etc) and monitors (with signals Track000, Ready, SeekComplete, WriteFault, Index) the current state of the drive.

2.3.3.2 **Microcontroller Detail** (See Figure RD-2)

This drawing shows some of the detail of the 8x305 MicroController. The 305 itself is in the center of the drawing. The code for the 305 is in the 2 PROMs to the left. Note that the 305 sends address signals to the PROMs which in turn sends instructions back to the 305. The Input/Output portions of the commands are stored in the PROM at the center left of the page. The I/O portion of each instruction is sent to I/O control at the same time the 305 portion is sent to it.

The 8x305 communicates with the outside world via an 8-bit, bi-directional I/O Bus. All the connections to the 305 shown on Figure RD 1 are actually connections to this I/O Bus. The 305 sends out timing signals to synchronize the outside world with its own internal timing. The clock for the 305 is not shown. The clock is the 10MHz 2x WriteClock used by the disk write circuits. The 305 executes each instruction in 200nsec.; each bit to or from the disk also takes 200nsec. The 305 can be reset by the IOP.

Scratch Pad Local Memory

The Scratch Pad memory is one high speed 256 x 8 RAM. The Scratch Pad Address Register is two 4 bit parallel loadable counters. The WriteMA instruction parallel loads this address counter, and subsequent reads and writes to and from the scratch pad will use this address. Any read (into 305) command with a certain bit set will increment the address, after the data has been read by the 305. This increment MA feature can only be used with a read into 305 command. The timing is not right for a write out of 305 command.

A hardware note: The address counter looks like it is connected to decrement the address instead of incrementing it. The counters are defined for positive-true logic while the 305 I/O Bus is negative-true. With negative-true logic, the positive-true decrement becomes a negative-true increment.

2.3.3.3 **Read and Write Logic** (See Figure RD-3)

Write Logic

The Write Logic data path is shown across the top of Figure RD 3. Timing, control, and status are not shown. The 8-bit parallel data from the I/O Bus is loaded into the parallel to serial converter. The serial output from this chip, in NRZ form, is sent to the CRC/ECC chip, where the CRC or ECC is accumulated. From the CRC/ECC chip the serial data goes to the MFM Generator, where the data is converted to MFM format. The MFM Generator also provides the Early, OnTime, and Late signals necessary for precompensation. If precomp is not enabled, only OnTime is used. The MFM signal goes through the delay line which produces three outputs, each 12 nsec apart. These Early, OnTime or Late MFM signals from the MFM Generator are gated with the Early, OnTime, or Late signals

RIGID DISK CONTROLLER
MICROCONTROLLER DETAIL

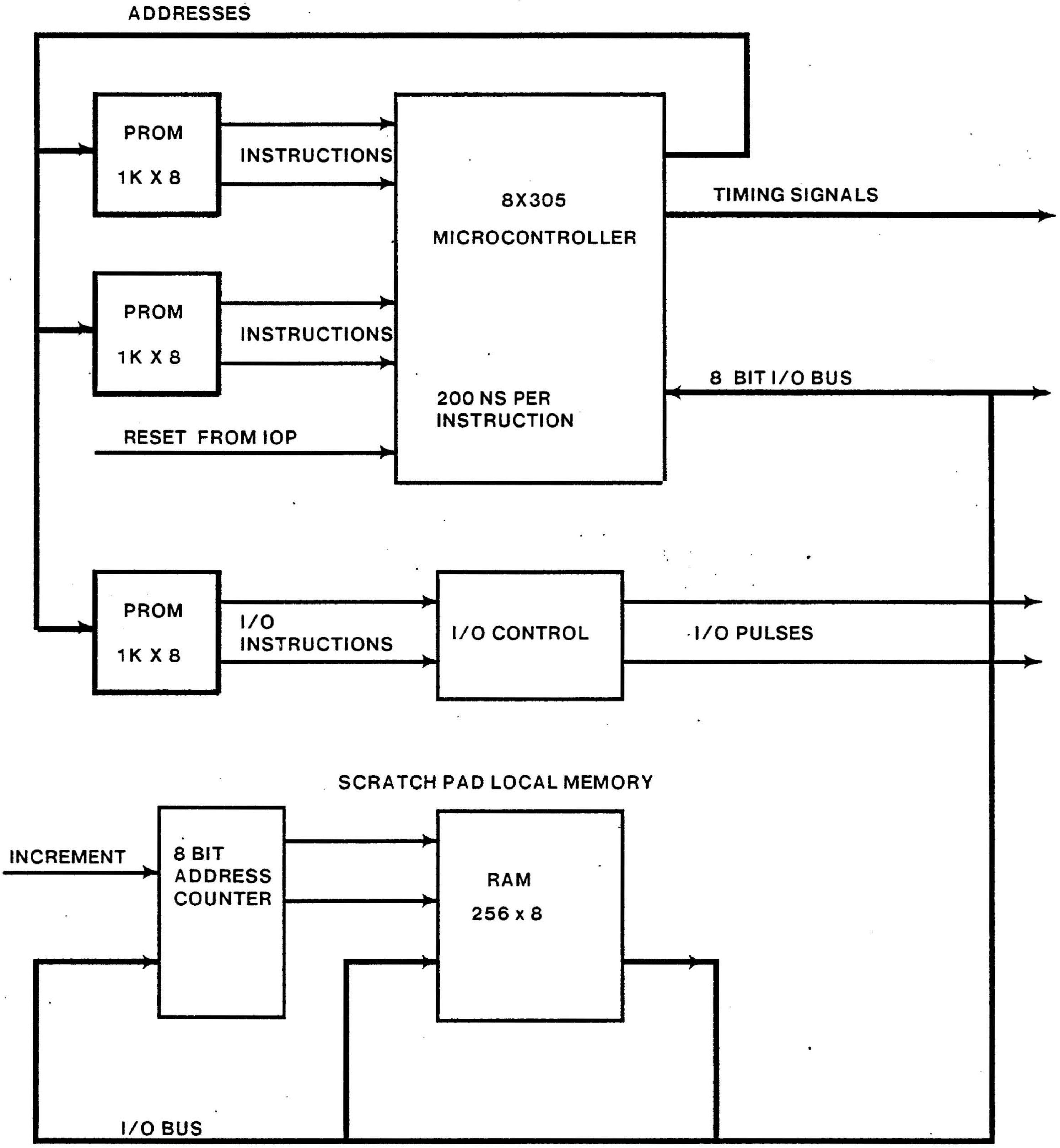
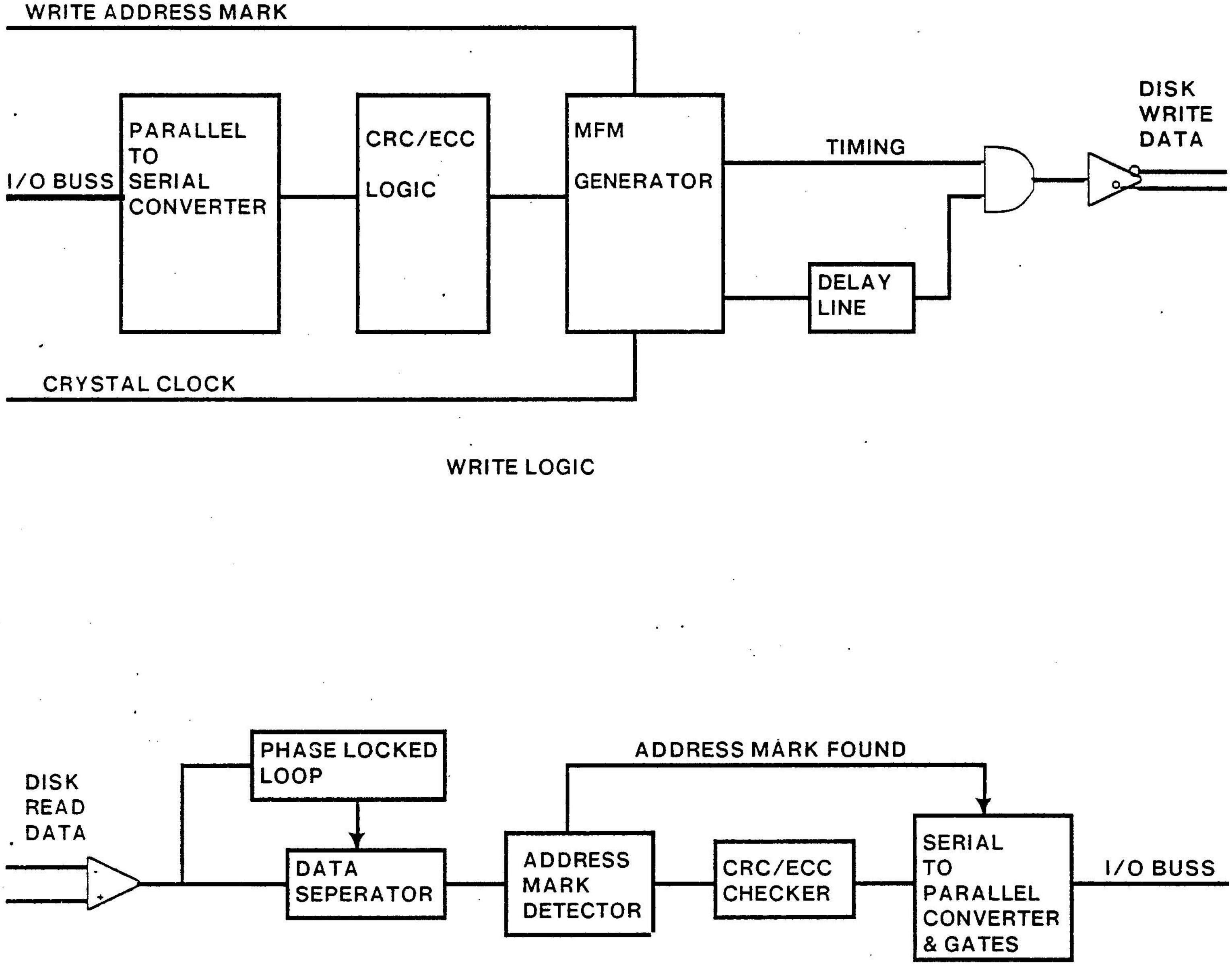


FIGURE RD 2

RDILL2.SIL

R Swenson 6/6/84

RIGID DISK CONTROLLER
READ & WRITE LOGIC



WRITE LOGIC

READ LOGIC

FIGURE RD 3

RDILL3.sil

R Swenson 6/6/84

from the delay line to produce the write output. The write data signal is converted to a differential signal for noise immunity, and sent to the drive.

When the data has completely passed through the CRC/ECC Generator, a control signal from the 305 changes the internal configuration of the CRC/ECC chip so that instead of accumulating the check bytes, the check bytes are sent out following the data. The check bytes are written just like any other bytes.

The MFM Generator also has a provision to delete a clock, producing the missing clock needed to generate the Address Mark.

Control logic note:

> There are two control signals with names that include write: Write and WriteGate. Write sets chips which have both a read and a write function to the write function. WriteGate actually enables write current in the drive head, and is turned on when everything is ready to actually write.

Read Logic

The read logic is shown across the bottom of Figure RD 3. Control and status signals and paths are not shown.

The disk is normally in a read condition with ReadGate off. That is, where a chip has both read and write uses, that chip is normally in read mode and switches to write mode when Write is turned on. Read is enabled with ReadGate.

Differential data from the disk enters at the left of the drawing, is converted to single-ended data, and goes to both the Phase Locked Loop and to the Data Seperator. When WriteGate is off the PLL looks at 2x write clock, and thus is near the frequency required to read data. When WriteGate is turned on by the 305, the PLL switches to Disk Read Data. When the PLL locks, it generates a signal LockDet. The DataSeparator separates the data pulses from the clock pulses and sends the separated pulses to the Address Mark Generator. The Data Separator produces a signal RdDataFnd when the first "one" bit is detected.

The Address Mark Detector looks for a certain pattern in the data and clock output of the Data Seperator. This unique pattern is the AddressMark. The AddressMarkFound signal enables the Serial to Parallel Converter.

Data passes to the CRC/ECC checker. This is the same chip used to generate the CRC/ECC check bytes when writing. The data passes through this checker. When it is finished, including the check bytes, the internal shift register should contain all zeros. Anything else is an error. When the entire data stream has passed through this checker, the 305 sends a signal and the internal shift register is shifted out, following the data, to the Serial to Parallel Converter and then to the 305.

The Serial to Parallel Converter takes the data, including the status of the CRC/ECC register, accumulates 8-bit words, and makes these words available to the 305. This converter has an internal counter which controls the transfer of data between the internal shift register and the internal buffer register, and which also signals the 305 that another byte is available. The 305 now has seven instruction times to take the data before the next byte is loaded on top of it.

Control Sequence

The sequence the 305 uses to control the read operation is:

1. Set CRC/ECC checker to proper position-CRC for header and label, ECC for data.

2. Read the Serial to Parallel Converter, clearing it.

3. Pulse ECCIZ on, then off. This signal, ECC InitaliZe, sets the CRC/ECC to start a new read field.

4. ReadGate on.

5. Wait for PLL Lock

6. Turn Search on. This enables the Address Mark Detector to look for an AddressMark byte.

7. Wait for RdDataFnd. A "one" bit has been found.

8. Wait exactly 10 bit times (10 305 instructions) and check if AMOut is on. If it is, a block is beginning. If AMOut is not on, this is not the beginning of a block, turn ReadGate off, and go back to step 2 above.

9. Wait for BDONE. A byte is ready to be read from the Serial to Parallel Converter.

10. Read byte into 305 and do the proper thing-verify the proper block ID byte, verify against header or label image, keep it to build up a word and then send to the FIFO, etc. Reading the data byte resets BDone

11. Repeat 9 and 10 as many times as required.

12. After data itself has been read, Wait for BDONE.

13. Read data and ignore. This allows one byte of CRC or ECC to pass through the checker. The read data command resets BDONE to keep in step with the data. When doing 2d to last read, turn IBLA on while keeping Read on. This sets up to read the error syndrome. After the last read, turn READ off with IBLA still on.

14. Repeat 12 and 13 once for CRC, three times for ECC.

15. Wait for BDONE

16. Read byte in, check that it is zero. This is the first byte of the error check syndrome. If not zero, an error has occured

17. Repeat 15 and 16 once more for CRC, three times for ECC.

18. Turn all read controls off. The read operation is over. If an error has occured and ECC is being used, the 4 byte syndrome will be needed by the Mesa world to correct the read error.ures

3 DMA/FIFO

In Dove system a Rigid Disk is present for non-volatile, long term storage. The Disk is controlled by a controller (RDC). In addition, a facility exists between the Main Memory and the Rigid Disk (and its

Controller) which provides a higher speed data tansport between the two storage media. The transport is in the form of Direct Memory Access (DMA). This facility has two elements: DMA Controller and FIFO. Since they are dedicated to the Rigid Disk access, they are grouped along with Rigid Disk device and its Controller under Rigid Disk Subsystem.

The controller for DMA function has a custom implementation and provides the controlling task as programmed. It interfaces the Main Memory through 80186 bus and is one of the '186 Bus Masters.

3.1 Why a Custom DMA

There are several requirements that led to choosing a custom implementation of the DMA Controller. Some of these were byproducts of peculiar architecture of the workstation system. For example, the Rigid Disk Controller had to be on the 186 bus and therefore in order to meet performance criteria, required a sizable FIFO which then necessitated a more specialized interface to the FIFO. Other reasons were due to specific performance limitations which were present in commercial DMA's. For example, the DMA controller inside the 186 chip could not be used for this purpose simply because DMA should be able to deliver address and keep track of the number of words transfered without requiring IOP's direct attention. Furthermore, it takes two memory cycles (about 1 microsecond without Tw states) for the 186 DMA to transfer one word while a normal rate of one memory cycle per word is required. The custom DMA further provided freedom in implementation. This enabled us to utilize a more suitable priority scheme for managing the '186 Bus traffic. As an example, the current implementation allows us to temporarily suspend an ongoing DMA transfer in order to service Ethernet Controller with an impact of very close to the real-time service of Ethernet packets (in the order of 4 T cycles for the arbitration over the real time consumed by Ethernet or other bus user).

3.2 Some Definitions

Memory Access One Word READ or WRITE from or to the Main Memory

IOP I/O Processor for Dove. It refers to the 80186 chip which functions as IO Processor

RDC Rigid Disk Controller. Also refered to as HDC or CTL in this text, the schematics and/or on the support materials

DMA unless otherwise noted, it refers to the Custom RDC DMA Controller for Dove (is the same for both Daisy and Daybreak)

Memory unless otherwise noted, same as Main Memory

DMA Operation
(or DMA Transfer) The sequence of events within DMA Controller to transfer data between Main Memory and the FIFO. It begins with a "Start DMA" request by the IOP and ends when such a transfer is completed.

Starting Address A 24 bit address provided by or through the IOP186, to be used as the REAL address of the first location in the Main Memory for DMA operation. (Note: no Virtual to real conversion is done within DMA Controller) The 24 bits is given by two IO accesses (see Programing Notes for more details)

Word Count The number of Words to be transferred in one DMA operation

3.3 Some Important Signals

StartDMA' or **StartDMAStrobe'**

Is a negative true pulse generated when IOP issues a StartDMA command (see Programing Notes)

DMAActive'

A negative true signal which indicates whether or not the DMA is active (a DMA Operation is in progress). This signal is removed (DMA not Active) when the DMA operation is temporarily suspended by outside factors (such as Ethernet, a qualified interrupt to IOP or the IOP itself)

RunSM' (Run State Machine)

Goes low in response to IOP's "StartDMA" request and allows the DMA to run. It remains active (low) until the completion of DMA Operation. During the time that DMA operation is temporarily suspended by outside factors, this signal will still maintain its active (low) level indicating that DMA transfer is not finished yet. This signal is available as one bit of the Status Register (see Programing Notes).

EndOfXfer'

A negative true signal that flags when the total number of words transferred (between Main Memory and the FIFO) equals to the number of words originally requested for DMA transfer.

ADDr'/Data

Is Low only during the T1 States of bus cycle while DMA operation is in progress. It identifies the type of information on the bus during DMA transfer.

For FIFO

FIFODIR

Programmable by IOP, indicates the direction of DMA transfer (1 = Memory to Disk, 0 = Disk to Memory)

FIFOOutOfBound

Depending on DMA Transfer direction (FIFODIR), this signal flags if the FIFO is not capable of servicing the DMA operation (i.e. when FIFO is full and direction is from Memory to Disk or when FIFO is empty and direction is from Disk to Memory

186 Bus Interface

Bus Control

RDCHoldReq

(i.e. RDC's DMA Hold Request. equivalent to HLD in 186)

RDCHoldAck

(i.e. RDC's DMA Hold Acknowledge. equivalent to HLDA of 186)**S.2'-S.0'** and **186BHE'**

Data Bus

AD.15-AD.00

AA.23-AA.16

IOPARDY (ARDY generated by the "A" chip in Daisy and through combinational logic in Daybreak)

ALE (not used directly. IOP circuit provides latched address as needed.)

DMA as a Peripheral for IOP 186

Rd'

WrL'

186DEn'

IOPPCS.4x' Peripheral Chip Select dedicated tothe RDC/FIFO/DMA (generated by IOP)

RDiskDmaIntr'

- Operates only on word quantities (single bytes are not supported)
- Programmable by IOP (Starting Address, Word COunt and transfer Direction)
- Can transfer up to 256 words (of 16 bits) in one DMA operation
- Provides 24 bits of address to Main Memory
- Data transfer at '186 bus rate.
- One word transfer / memory cycle: significantly higher bandwith for DMA with respect to the 186's internal DMA
- Semi-intelligent :Provides graceful recovery from, and minimizes its own impact on the 186 Bus traffic
- End-of-transfer always interrupts IOP

3.4 Functional Description

DMA Controller consists of two major sections: Control Section and Data Path section. Control Section consists of a Control register, a Status register, a State machine (which generates appropriate control signals based on its existing state and inputs from other parts of the DMA circuitry) and other control logic required to make the system fully functional. Data Path section, on the other hand, primarily includes the buffers, registers and latches needed to direct the data back and forth between the FIFO and 186 bus.

Upon programming by the IOP, DMA Controller in response to a StartDMA (again from IOP) will initiate a DMA transfer in the direction already specified (programmed) beginning with the word at the location given by the Starting Address. It will transfer the words with the consequtive addresses (incrementing the Address) up to the total number in the Word Count . This transfer will occur at the rate of one word per memory cycle at the memory speed (i.e. 4 T states plus Tw's as needed by the memory). The DMA transfer may be interrupted according to the priority scheme to which the Bus Arbiter is programmed. The transfer will resume from the same point where interrupted and will proceed to the end of DMA (or until the next interruption as the case might be).

Such a transfer is subject to availability of storage space in the FIFO in the case of Memory-to-Disk direction (or availability of data in the FIFO in the case of Disk-to-memory). If such availability does not exist any time during the transfer, DMA Controller will automatically relinquish the 186 bus (making it available to other resources) and wait until FIFO is available once again. It will then request the control of the bus and once the control is granted, it will continue the transfer.

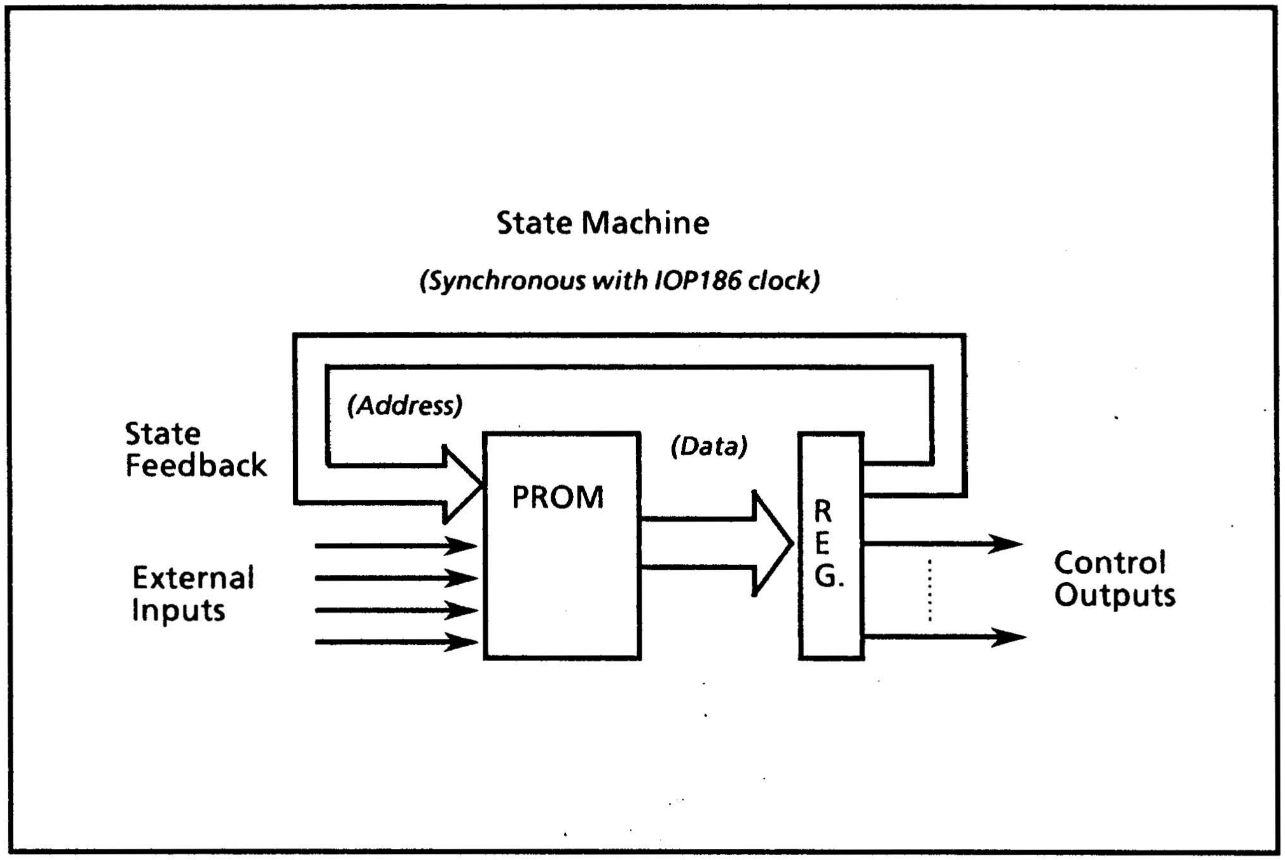


Figure 3.1. DMA Block Diagram - I
State Machine

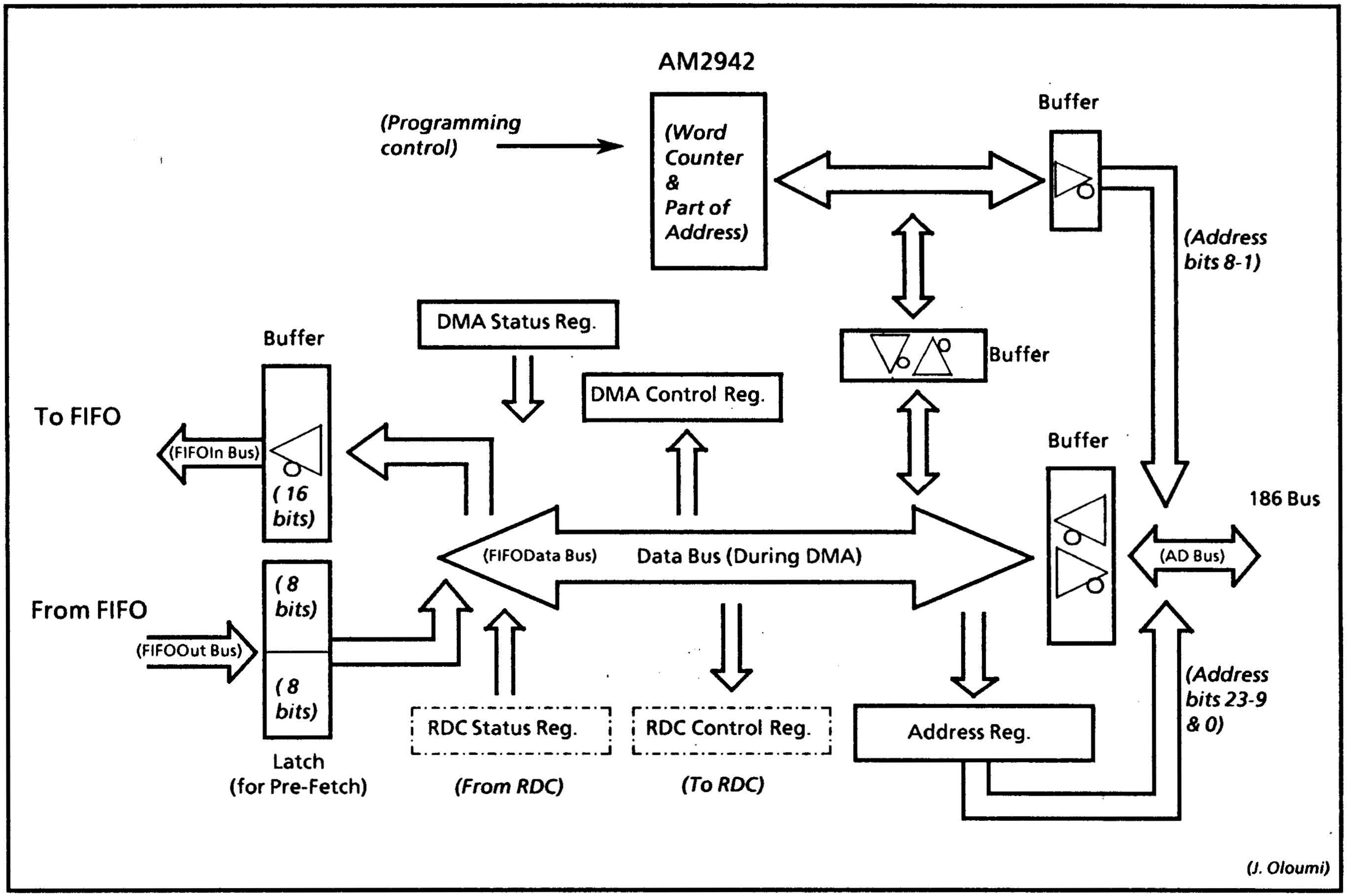


Figure 3.2. DMA Block Diagram - II
Address/Data Paths

3.4.1 Impact of ARDY (Daisy vs Daybreak)

DMA is capable of transfering data (1 Mem Cycle / Word) without any Wait states (Tw). However, if ARDY signal is pulled low by memory, DMA Controller will introduce additional Wait States as long as ARDY is low. This permits access to slower memories who comply with 186 bus protocol.

Note: DMA Controller will not give-up the control of the bus until the current memory access is completed. Bus arbiteration will only occur between the memory accesses. This will gurantee the data integrity during DMA without retry penalties .

3.4.2 States of the State Machine

The following is a walk-through of the states for the State Machine. Diagrams of these states and their inter-relationship are also included for reference.

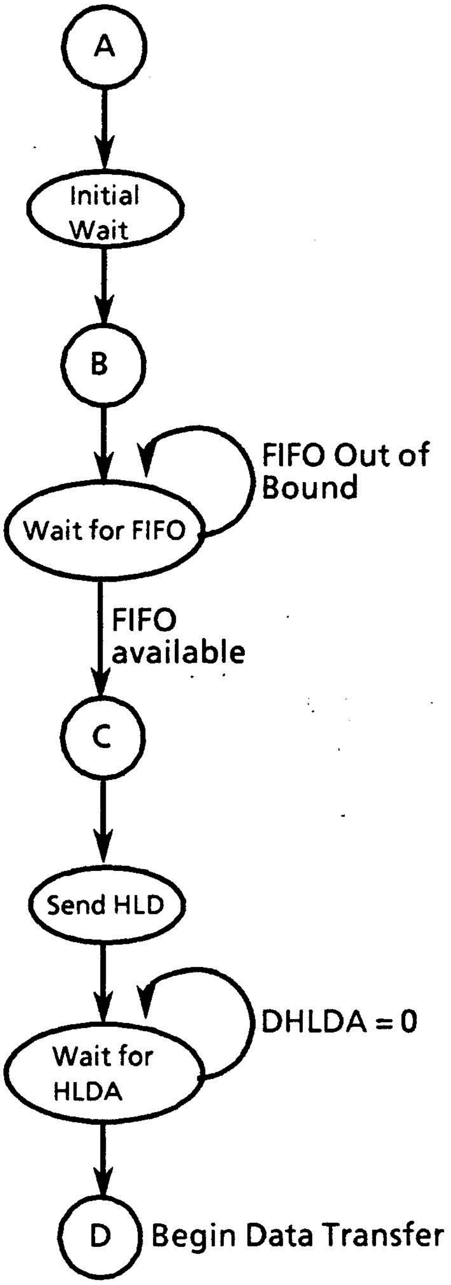
The state machine normally is in the "Initial Wait" mode. This occurs at the power-up time by resetting the DMA Controller or at other times once an existing DMA transfer is successfully completed. The StartDMA command from IOP will cause the state machine to go to the next state where it verifies if the FIFO is available for DMA operation. If so, it will go to the next state. Otherwise it will wait in the same state (without requesting the 186 bus of course) until FIFO becomes available. It will then send a Hold Request (HLD) to the Bus Arbiter and wait for Hold Acknowledge. Once the Acknowledge is received (HLDA) which means the 186 bus is clear and available for DMA transfer, it will then begin the bus cycles for the transfer. First, it will introduce an intial T state (Ts, equivalent to the last T state of an ongoing memory access) and then go to T1, T2 and T3 states consequtively while providing signals according to the 186 bus protocol for memory access and appropriate control signals to the FIFO and Address/Word Count logic. Then it will go to T4 state if Main Memory can complete that memory access in four T cycles.

If the Main Memory needs more time to complete the access, it will inform DMA Controller by pulling ARDY signal low. In response, the State Machine after T3 will go to a Tw state and continue introducing Tw cycles until the Main Memory announces completion of access by pulling the ADRY signal back High. DMA Controller will then move from Tw state to T4 state and complete one word transfer. At T4 state, the State Machine will branch into three possible states:

1. T1, if there are more data to be transfered and the FIFO is available and no other Bus master is requesting the 186 bus.

2. To another state if total number of words requested for DMA transfer has been reached to send Interrupt request to IOP (and dropping the HLD line at the same time). This occurs when End-of-Transfer condition is reached regardless of other factors like FIFO unavailable or another Bus Master requesting the bus.

3. If the FIFO is not available for the next word to be transfered or another 186 Bus Master is requesting the bus, it will go to a third state where it will drop its Hold Request line (HLD) (thus relinquishing the 186 bus) . In this case, after dropping the HLD line depending on the cause, it will either go to a state waiting for FIFO to become available, or will introduce one more T state (Delay) before it goes back to "SendHLD" state to send the Hold Request again.



FIFO Out of Bound (for DMA):
FIFO Full & Mem-To-Disk
or
FIFO Empty & Disk-To-Mem.

(J.Oloumi)

Figure 3.1. DMA States - 1

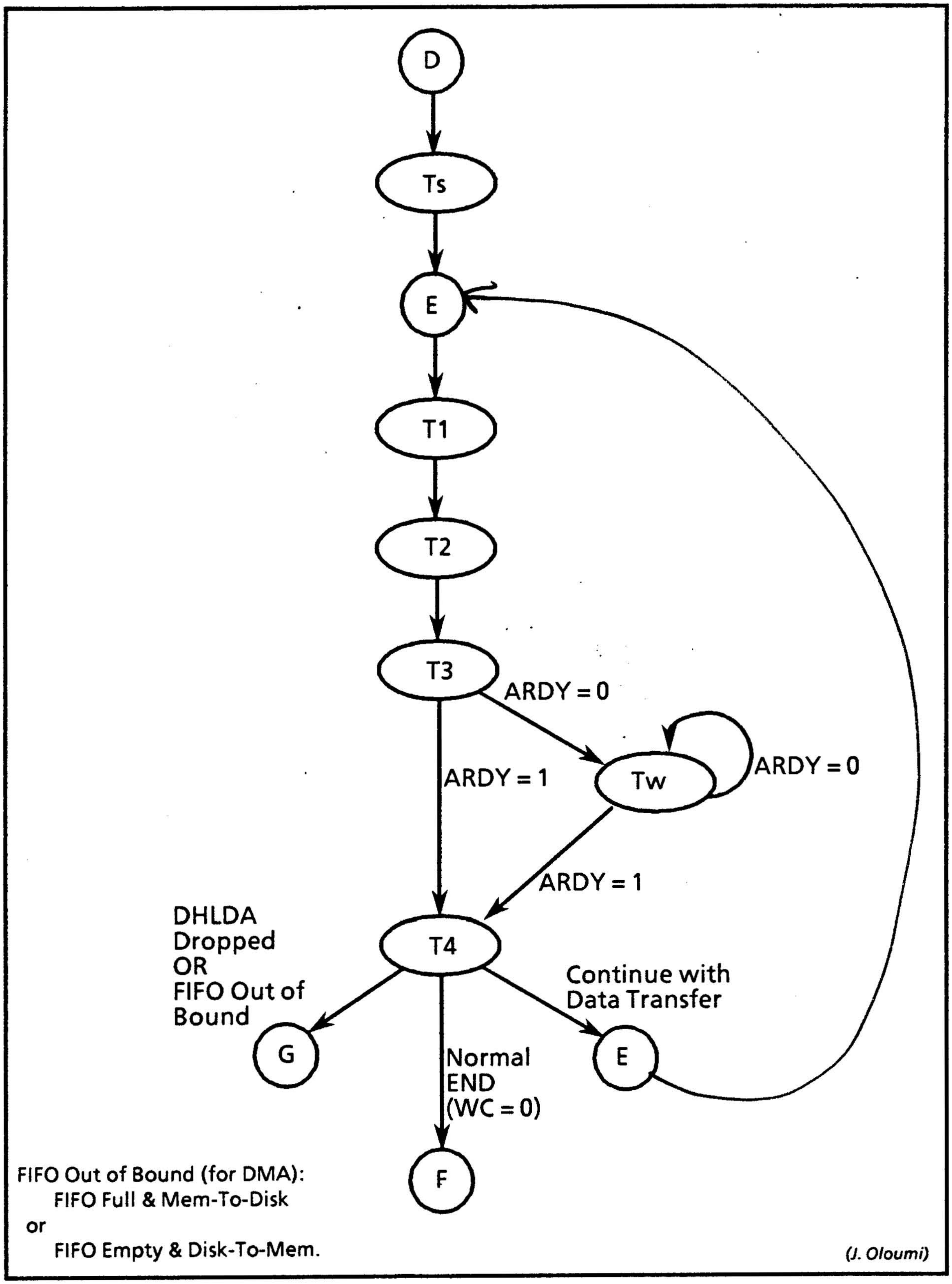
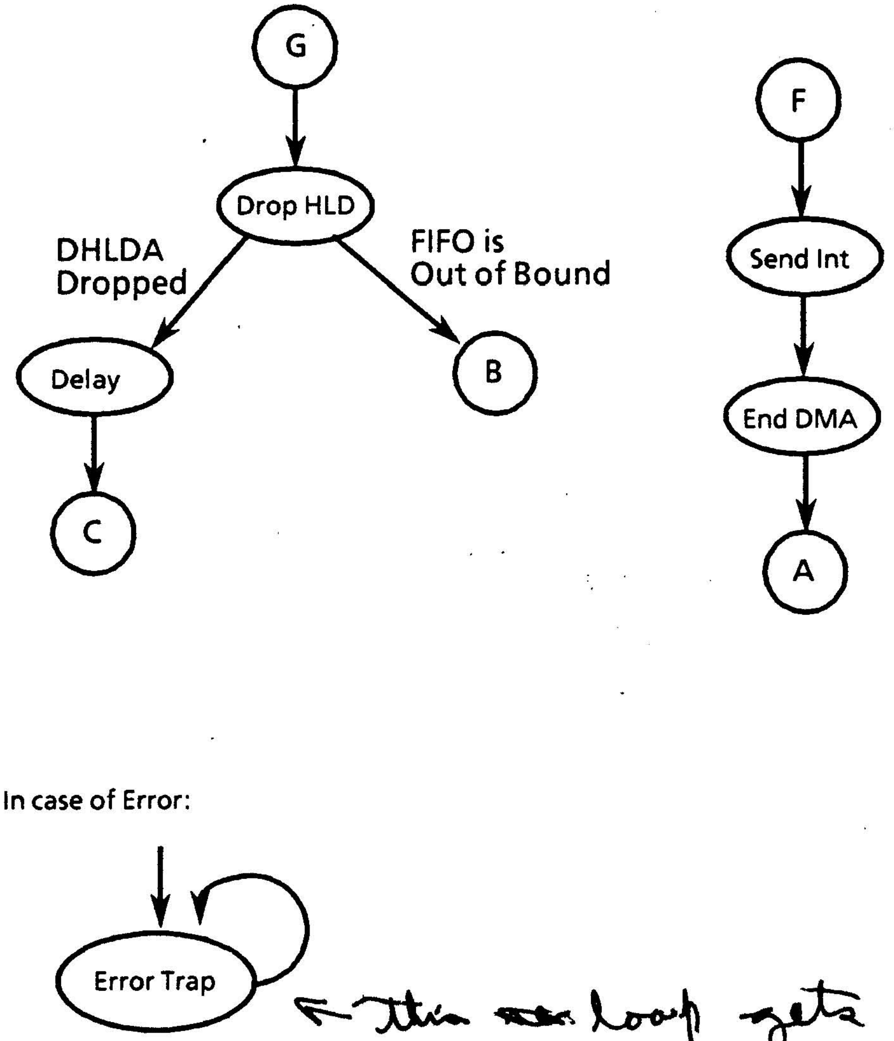

D
Ts
E
T1
T2
T3
ARDY = 0
Tw
ARDY = 0
ARDY = 1
ARDY = 1
DHLDA Dropped OR FIFO Out of Bound
T4
Continue with Data Transfer
G
Normal END (WC = 0)
E
F
FIFO Out of Bound (for DMA):
FIFO Full & Mem-To-Disk
or
FIFO Empty & Disk-To-Mem.
(J. Oloumi)

Figure 3.2. DMA States - 2



← this loop gets broken by outside (drop hold) interrupt

FIFO Out of Bound (for DMA):
FIFO Full & Mem-To-Disk
or
FIFO Empty & Disk-To-Mem.

Figure 3.3. DMA States - 3

(J.Oloumi)

4 First-In-First-Out (FIFO) Buffers

In order to further isolate the Processor/Main Memory subsystem from the inherent latency characteristics of the Rigid Disk, a temporary storage is provided in the form of First-In First-Out between Main Memory and the Rigid Disk.

4.1 Characteristics of the FIFO

- Sizable Data buffering capability (512 words long = 2 Mesa pages = 2 sectors): minimizes the impact of Disk latencies on DMA operation
- Bidirectional access (Memory-to-Disk & Disk-to-Memory): Programable
- Simultaneously and asynchronously accessible by DMA and RDC
- Full, Half-full and Empty indications
- Resetabe at the power-up time and under program control

4.2 Functional Description

The FIFO is made of two 512X9 components that are used to arrange a 512X16 structure. Some logic to provides the Half-full capability and appropriate buffering at both ends such that data received or delivered to RDC in Byte quantities and to DMA (and 186 bus) in word quantities. Based on the value of the direction bit in Control Register, FIFO structure is either in Memory-to-Disk direction or Disk-to-Memory. And will only change when programmed otherwise (by the IOP), This temporary storage is used to transfer Data or IOCBs between Rigid DIsk/RDC in one end and Main Memory in the other.

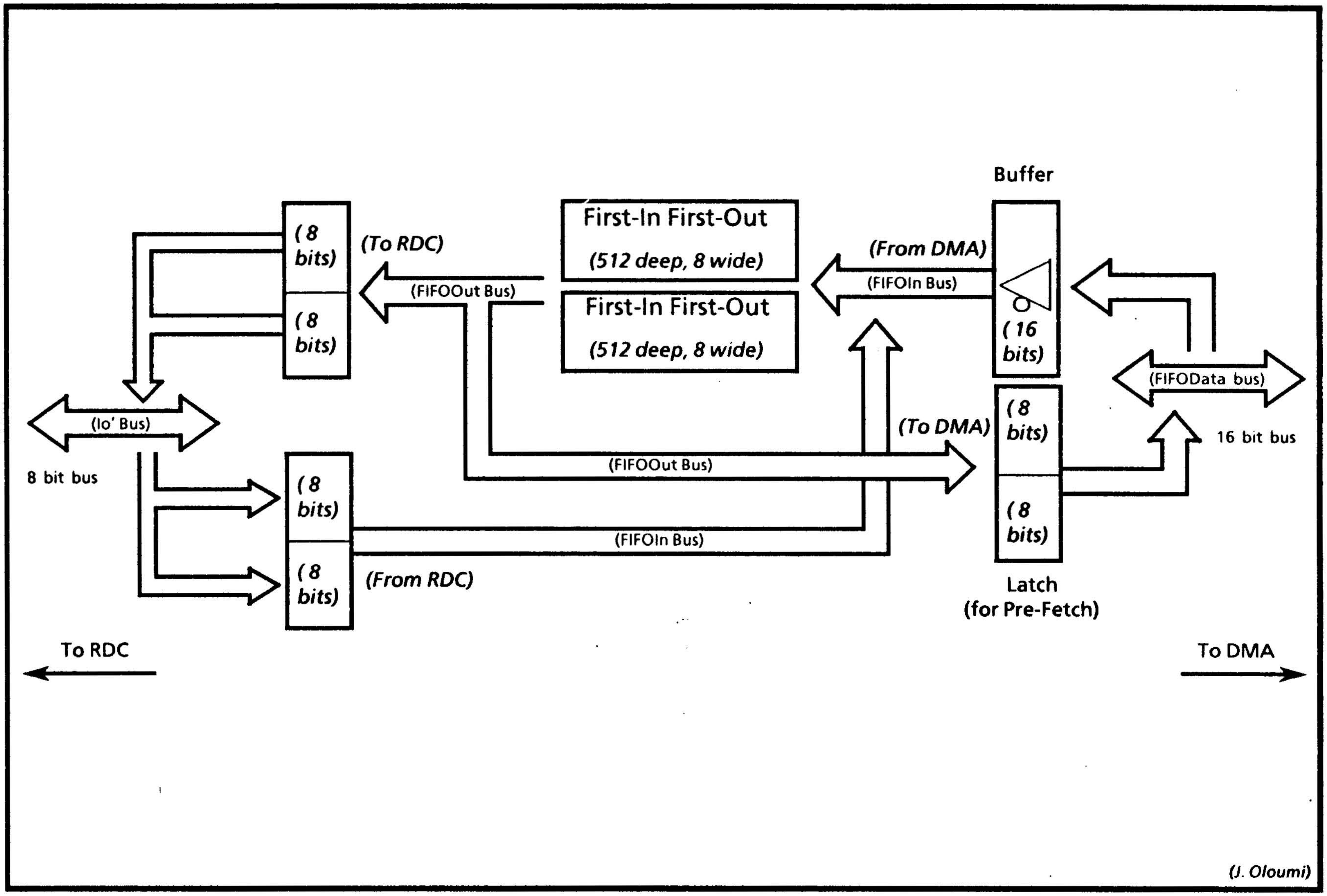


Figure 4.1. FIFO Block Diagram

5 Additional Considerations

5.1 DMA State Machine

If an illegal condition is presented to the inputs of the State Machine, it will create an Error condition, set the Error bit (in DMA Status Register), freeze the clock for the Status Register (and therby preserving the latest status just before the error condition) and send an Interrupt to IOP. It is reccommended that IOP in response to each DMA interrupt, verify if that is due to an error condition or really indicates successful completion of DMA transfer.

5.2 The FIFO

If at any time FIFO is not available for DMA transfer (i.e. is FULL when DMA logic transfers data from Main Memory to FIFO or is EMPTY in the reverse direction) then DMA logic will relinquish the '186 bus and will wait until FIFO is again available to resume the DMA transfer.

There is no explicit or implicit indication as to the contents of FIFO at any given time. The surrounding software will ensure such integrity using the facilities available at both ends of the FIFO. Those include the Starting Address and the Word Count at DMA side and 8X305 facilities (i.e. Control and Status registers) at Disk Controller side. In addition no direct communication facilities are implemented between the Disk Controller and DMA circuitries. They operate totally asynchronous with respect to each other. Therefore no knowledge about the contents of the FIFO is exchnaged between the two circuits. Data inside the FIFO whether it be part of an IOCB, or part of a DATA block, are treated in a First-Come First-Serve fashion. Direction of transfer is of essence in this implementation. IOP software has to ensure that the Direction bit is correct and will NOT change until a data transport is successfully completed from the origin to destination.

6 Appendix

6.1 Disk DMA Programming Notes

6.2 Introduction

This document describes some of the considerations that are necessary in writing control modules, programs, maintenance routines etc. for the Daisy Rigid Disk DMA/FIFO interface.

6.3 Architecture

There are four registers within the RDC/FIFO/DMA area which can be used by the IOP to communicate with this subsystem. They are DMA Command Register, DMA Status Register, RDC Control Register and RDC Status Register. Bit description for these registers is as follows:

1. DMA Command register (a Write-Only register) (IO address "210H")Currently it only contains one bit which is the "FIFO Direction". The bit is the least significant bit of the register (Data bit 00). Value = 1 is the direction from Main Memory to Disk. Value = 0 is from Disk to Main Memory.

2 DMA Status Register (a Read-Only register, and has Reset-on-Read characteristic) (IO address "210H")Currently the lower 7 bits are used. They are:

Bit 00 = Error.
Normally is 0. It becomes 1 when DMA State Machine is led to and incorrect state. In such a case, all other bits of this register (currently bits 6-1) will maintain the latest information just prior to occurance of that error. This bit remains 1 until a "Reset" occurs. Note that a read from Status Register will NOT clear this particular bit.

Bit 01 = RunSM'
Is 0 when DMA has not completed a DMA transfer yet. i.e. a DMA operation is in progress.

Bit 02 = EndOfXfer'
Is 0 when a DMA transfer is completed and DMA logic has no other request pending

Bit 03 = FIFOOutOfBound'
Is 0 if FIFO is found Empty during a read atempt by DMA or is found full during a Write attempt to the FIFO by DMA.

Bit 04 = FIFOEmpty'
Is 0 if FIFO is Empty.

Bit 05 = FIFOFull'
Is 0 if FIFO is Full.

Bit 06 = B1

Output of the FIFO for a Status bit carried by Data through the FIFO. It has meaning only when compared with A1, the input to FIFO for the same bit (For programming A1, see below under "IO addresses for RDC/FIFO/DMA"). In addition to the internal use of A1 and B1 by the Disk Controller during DMA transfer, these two bits can be used to monitor transport of data words through the FIFO.

Bit 07 = FIFODIR (FIFO Direction) Loop back of the direction bit from DMA Command Register.

3. RDC Control Register (Write-Only register) (IO address "214H")
This register can be loaded by performing an IO Write to the address A04-01=1010 (while PCS4 is selected). The contents of bits 7-0 of Data Bus (Lower significant byte) will be stored in that register (for bit description of the register please refer to Disk Controller documents).

4. RDC Status Register (Read-Only register) (IO address "214H")
An IO Read from the same address (A04-01=1010) while PCS4 is selected will put the contents of Disk Status Register on bits 7-0 of Data Bus (Lower significant byte). For bit description of this register please refer to Disk Controller documents.

6.4 11-10 Addresses for RDCIFOIDMA

IOP accesses the RDC/FIFO/DMA by selecting Peripheral Chip Select #4 and reading or writing to the following IO addresses (A summary of the commands along with full IO address to be used for each case is given at the end of this section):

Upper bits of address from A15 through A05 are ignored (except as it defines PCS4).

A04-A01 = 0XXX All are reserved for programming AM2942
A04-A01 = 1000 DMA Command/Status (Write = Command register, Read = Status register)
A04-A01 = 1001 Pre-Set "A1". (A1 is the Status bit input to the FIFO)
A04-A01 = 1010 Rigid Disk Controller Control/Status (Write = Control register, Read = Status register)
A04-A01 = 1011 Start DMA transfer (write-Only, contents of data bus ignored at that time

6.5 Programming the DMA

There are 6 registers relating to DMA function. They are:

- Starting Address Register bits 8-1
- Starting Address Register bits 23-9 (and bit 0)
- Current Address Register bits 8-1
  (changes during DMA transfer to provide a current address)
- Word Count Register
  (keeps count of number of words left to be transfered)
- Word Count Shadow Register
  (maintains the original value given as word count, i.e. number of words to be transfered in one DMA operation)
- Control Register within AM2942 (CR2-CR0)
  (used to program that component)

The following is how to access them:

6.5.1 Writing the Starting address

The 24-bit address is given by two IO write commands (i.e. "OUT"s)

A- Bits 08-01 = > Do a write to address "20AH" while data is given on bits 08-01 of the Data Bus (i.e. AD bus). Note that bit 00 of Data bus is ignored (and can be 0). This means that a Word Address can be given without any need for shift.

B- Bits 23-09 = > Do a write to address "208H" while data is given on bits 14-00 of the Data Bus (i.e. AD bus). In this case, bit 15 of the data is forwarded as bit 00 of the 24-bit Starting Address Register. Note that the DMA address bit 00 for all DMA transfers must be 0 (all of the Rigid Disk & DMA related address/data/control/status are at word boundaries).

Important Note: writing bits 23-09 of Starting Address Register will re-initialize both the Word Counter and the Starting Address Register bits 8-1. That means their current value (if different from the original) will be changed and become equal to what was last given per paragraphs 1-A above and 3 below.

6.5.2 Reading the Current Address

Whenever DMA is not bus master, IOP can read the Current Word Count and part of the Current Address.

In this implementation, only bits 8-1 of the latest address is accessible by the IOP. For this reason, reading the address register may not be of significant help. The reading however, is done in a similar manner as a write. That is by an IO Read ("IN") function from "206H" bits 08-01 of the Current Address will be delivered on bits 08-01 of AD bus as data. Other bits of Address Register are NOT accessible.

6.5.3 Writing the Word Count

The word count will be 8 bits wide (maximum of 256 words can be transfered in one DMA operation). The count should be given over the AD bus bits 08-01 (i.e. shifted left by one) in the form of two's complement of the actual number of words to be transfered (that is, to transfer 256 words, write "00"HEX; for 1 word, "FF"HEX and for 2 words, "FE"HEX etc. to bits 8-1 of AD bus). An IO write command ("OUT") to address "20CH" with the word count on the data bus as specified above will implement this function.

6.5.4 Reading the Word Count

Is done in th same order as Write but using address "204H" instead and performing an IO read ("IN"). Remeber that the contents are in two's complement form and are shifted left by one bit.

6.5.5 Writing the Control Register within AM2942 (CR2-CR0)

An IO write to address "200H" will load bits 3-1 of the Data bus into bits 2-0 of CR register. Since AM2942 is intended to be used in mode 3 (refer to the chip specs sheet for more details) the contents of CR register must be CR2-0 = 011 (this means write 0006H to that IO port).

6.5.6 Reading the CR2-CR0

If a read from CR register is desired, it may be done by an IO read from address "202H". Contents of CR register will be provided on bits 3-1 of data bus. Other bits should be ignored.

Note: for more details on Programming the Starting addres and Word count, please refer to the AMD2942 Specifications. This implementation intends to use that component in Mode 3.

6.6 Summary of IO Addresses for RDCIFIFOIDMA

Note: 2942 chip within DMA, is used in mode 3.
IO Address Description

200H	Write CR2-CR0 (inside 2942)
202H	Read CR2-CR0 (inside 2942)
204H	Read Word Count (in 2's Complement form) (inside 2942)
206H	Read Address Counter bits 8-1 (inside 2942)
208H	Load Starting Address bits 23-9 (& 0) and Re-Initialize Counters (inside 2942)
20AH	Load Starting Address bits 8-1 (inside 2942)
20CH	Load Word Count (2's Complement) (inside 2942)

20EH	Enable Counters (inside 2942) -- Not used
210H	DMA Command/Status reg's (Write = Command Reg, Read = Status Reg)
212H	Preset "A1'" (Write) (for Diagnostics)
214H	Controller's Cont/Status reg's (Write = Control Reg, Read = Status Reg)
216H	Start DMA (Write)
218H	Not Used
21AH	Not Used
21CH	Not Used
21EH	Not Used