List Manipulator Specification

Version 1-1

Cliff Young
Harvard University

(617) 496-4513
cyoung@das.harvard.edu
Pierce Hall 110C, 29 Oxford Street, Cambridge, MA 02138

This document specifies the Cell, Data/Management Active, and Credit Active List Manipulators on the Port Card. Parts of these designs may be reused in the Host Interface boards, but this is not currently part of the specification.

Because this document specifies three blocks, some sections will apply to only one or two of the blocks. Where not explicitly stated, observations apply to all three blocks.

1.0 Introduction

1.1 Purpose/Description

The list manipulators are hardware implementations of high-speed data structures needed by the Scheduler. For ease of description, it helps to define some terms. Relative to a given port card, an outbound VC is one whose data cells will be transmitted on the link managed by that port card. An inbound VC is one whose data cells will be received from that link.

For each outbound VC, the Cell List Manipulator (CLM) maintains a FIFO list of the addresses of cells in the core that have not yet been sent. This allows the Scheduler to specify that the next cell for a VC should be sent without needing to know the specific address of that cell. The CLM must monitor the W_IVC and W_CADR core buses to identify and remember relevant cells. It responds to IVCs dequeued from the DALM, retrieving the next CADR for that IVC and broadcasting the R_CADR on its bus during the slot for this port card.

The Data/Management Active List Manipulator (DALM) keeps FIFO lists of outbound VCs that are eligible to send cells out this port's output side. The DALM accepts

Enqueue requests from the VAR and answers Dequeue requests from the Scheduler. It must supply Status information to the Scheduler about which Active Lists are currently nonempty, and it must pass the Dequeued IVC to the CLM for cell address lookup and transmission on the R_IVC buses.

The Credit Active List Manipulator (CALM) manages FIFO lists of inbound VCs that need to send credit. It accepts Enqueue requests from the Credit Cell Generator (CCG). Like the DALM, the CALM answers Dequeue requests from the Scheduler. It also answers Dequeue requests from the CCG.

When the Scheduler schedules an empty cell or credit cell, or when the Scheduler underflows (i.e. it misses the slot where the core pipeline expects to receive an ALID to process), the DALM signals the CLM what has happened. The CLM then places an appropriate null cell address on the R_CADR bus.

Since outbound VC data and inbound VC credit compete for the output link, the Scheduler must choose between the two ALMs when scheduling cells for the outbound link.

1.2 Requirements

1.2.1 Shared Requirements

1. All List Manipulators must run at the core bus rate (this number is approximately 45MHz).
2. A cell cycle will consist of 32 core bus cycles.
3. All lists operate like FIFOs.
4. All List Manipulators will support at least 8K distinct IVC values.
5. The following 5 special IVC and CADR values will be defined and reserved, and will be used on the appropriate buses under the appropriate conditions: NULL, UNEQP, EMPTY, CREDIT, and NOSCHED. For the last three, special ALID values will also be defined and reserved.

   NULL_IVC and NULL_CADR appear on the W_IVC and W_CADR buses and indicate that the IPATH did not receive a cell that cycle.

   UNEQP_IVC and UNEQP_CADR indicate that a port card is not installed; this is probably the pullup/precharge state of the W_IVC, W_CADR, and R_CADR buses.

   EMPTY_IVC and EMPTY_CADR indicate that the scheduler explicitly told the DALM and CALM not to schedule a cell for this cycle. EMPTY_ALID is what the scheduler sends to the DALM to trigger this.

   CREDIT_IVC is sent from the DALM to the CLM when the scheduler tells the CALM to send a credit. The CLM then broadcasts CREDIT_CADR on the R_CADR bus.

   NOSCHED_IVC is sent from the DALM to the CLM when the scheduler misses its request slot to the DALM or CALM, causing a scheduling “underflow.” The CLM sends NOSCHED_CADR on the core buses.

6. All List Manipulators must allow the PC to perform one read or write per 32-clock cell cycle for maintenance access.
7. All List Manipulators must allow the PC to access a consistent memory state when in Initialization, Recovery, or Post-Mortem Mode. Note that there is no consistency requirement in Single-Step Mode because the port card pipeline continues operating at normal speed. Turning off inputs and outputs ensures no changes to port card pipeline data structures, so consistency between reads is supplied without changing the implementation of the List Manipulators.

1.2.2 CLM Requirements

1. CLM lists are identified by IVC. Their elements W_CADRs.
2. The CLM must support 64K CADRs.
3. The CLM must monitor the W_IVC and W_CADR buses. When it sees a W_IVC for which the Ownership[IVC] bit is set, it must enqueue the matching <W_IVC, W_CADR> pair.
4. The CLM must support 16 Enqueues per cell cycle (one for each possible input port that could send a cell to this output port).
5. The CLM must accept Dequeued R_IVCs from the DALM and perform a Dequeue on the R_IVC to find the correct R_CADR for that R_IVC. This means that the CLM must support one Dequeue per cell cycle.
6. The CLM must send R_CADR to the core buses in the appropriate slot for this port card.

1.2.3 DALM Requirements

1. DALM lists are identified by ALID. Their elements are IVCs.
2. The DALM must support a total of 8K distinct IVC values.
3. The DALM must support 4K ALIDs.
4. The DALM will accept 18 Enqueue requests per cycle from the VAR (16 input ports, one reactivation, one credit activation).
5. The DALM will support 1 Dequeue per cell cycle.
6. The DALM will send the Dequeued IVC to the CLM. If the Scheduler schedules an empty cell (by passing EMPTY_ALID to the DALM), a credit cell (by dequeueing from the CALM), or underflows (by failing to send an ALID to either the DALM or CALM), then the DALM will send a corresponding IVC (EMPTY_IVC, CREDIT_IVC, or NOSCHED_IVC) to the CLM.
7. The DALM will support Dequeues through a memory-mapped write interface. The DALM will include a single register that can hold an ALID until the correct schedule slot where the Dequeue is performed. The register is interlocked to block the Scheduler if the Scheduler attempts to overwrite a new ALID before the previous ALID has been serviced.
8. The DALM will support Status queries through a memory-mapped Read interface. ALIDs will be placed in groups of 32 based on ALID number. The status request will specify an active list GROUP_ID. The return value will be the bitmap for the 32 ALIDs in that group. For example, a Status request for address 2 would return a bitmap of status on ALIDs 64 through 95.

1.2.4 CALM Requirements

1. CALM lists are identified by ALID. Their elements are IVCs.
2. The CALM must support a total of 8K distinct IVC values.
3. The CALM must support 4K ALIDs.
4. The CALM will accept 16 Enqueue requests per cycle from the CCG (one for each possibly triggering output port).
5. The CALM will support 1 Dequeue per cell cycle.
6. The CALM will send the Dequeued IVC to the CCG.
7. The CALM will support Dequeues through a memory-mapped write interface. This interface is similar to the DALM dequeue interface.
8. The CALM will support Status queries from the Scheduler through a memory-mapped Read interface. This interface is similar to the DALM's Status interface.

1.3 Functions Supported

1.3.1 Enqueue (All)

Enqueue adds a specified element to a list. For the CLM, W_CADR is added to the list of the owning W_IVC. For the ALMs, the ENQ_IVC is added to the list of its ENQ_ALID. Note that "Smart" Enqueue is now handled by the DALM and CALM callers: it is up to the VAR and CCG, respectively, to ensure that they do not re-Enqueue an IVC that is already on some list. If they do so, the DALM and CALM will happily trash their data structures.

1.3.2 Dequeue (All)

Dequeue removes the first element from a list, passing the return value to the appropriate place. For the CLM, the first element of the DEQ_IVC is dequeued, yielding a R_CADR. This is sent to the R_CADR core bus at the appropriate time. For the DALM, the first element of the D_DEQ_ALID is dequeued and sent to the CLM. In the CALM, the first element of the C_DEQ_ALID is dequeued and sent to the CCG.

1.3.3 Status (DALM, CALM)

Status requests allow the Scheduler to tell which active lists are not empty. For each ALM, the active lists are bundled into groups of 32. The 32 members of a group differ only in the five LSBs of their ALIDs. To request status for a group, the AL_GROUP is generated by removing the five LSBs from a member's ALID. Status is returned in a 32-bit bitmap, where the ith bit is set if the ith group member is nonempty.

1.3.4 Mode Control (All)

If the INIT register is set to "OFF", all List Manipulators will float their buses and ignore the core. If the INIT register is switched to "ON", the List Manipulators will synchronize their operations to the fixed schedule as driven by this port card's frame pulse.

There may be just one control register for all of the CLM, VAR, DALM, CCG, and CALM, since these blocks are all synchronized to each other and to the core buses and frame pulse.

1.3.5 Auxiliary Access (All)

At least one cycle of the fixed 32-clock core cycle will allow the PC to perform a read or write to any of the memories in a List Manipulator block. Implementing this probably requires load and store buffers, with interlocks to prevent overwriting stores before they complete. We also want to be able to issue prefetch instructions (by storing to a different, parallel-mapped region of the PC's address space) to ensure that the read buffer can be read quickly later.

Unless the INIT register is "OFF," no consistency or atomicity is guaranteed *by the hardware* between sequential read and write operations. In Single-Step Mode, consistency is guaranteed by turning all inputs and outputs (which might change state) off. If the INIT register is "OFF", then the arbitration logic should give exclusive memory access to the PC.

1.4 Functional Block Diagram

Please see Shuo Huang's diagrams of these blocks.

1.5 Pseudocode

Please see Shuo Huang's pseudocode descriptions of these blocks.

1.6 Timing Diagrams

To avoid pipeline data hazards in the VAR block, the CLM, VAR, DALM, and CALM operate on a fixed 32-cycle schedule with specific times that each Enqueue and Dequeue operation takes place. Please see Dong Lin or Shuo Huang's descriptions of this fixed schedule.

1.7 VHDL

The VHDL specification is not required for this version of the specification.

2.0 Interfaces to other blocks

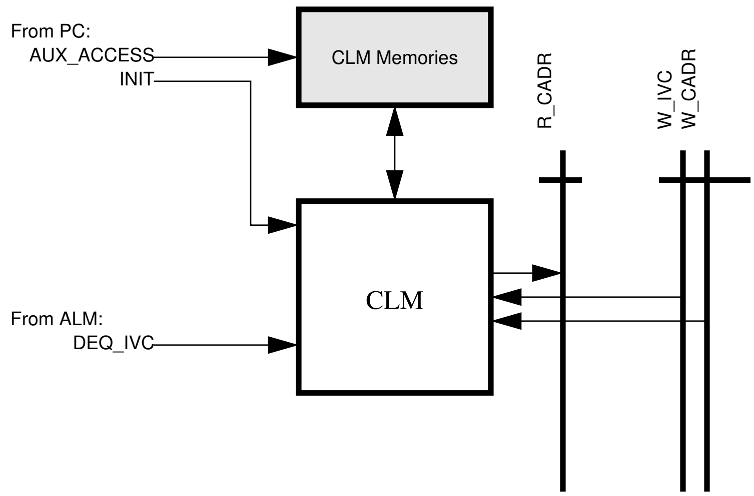
2.1 Purpose/Description

There are five major classes of interfaces to the List Manipulators:

1. Enqueue Interfaces.
2. Dequeue Interfaces.
3. Status Interfaces (DALM and CALM only).
4. INIT Register
5. Auxiliary Access

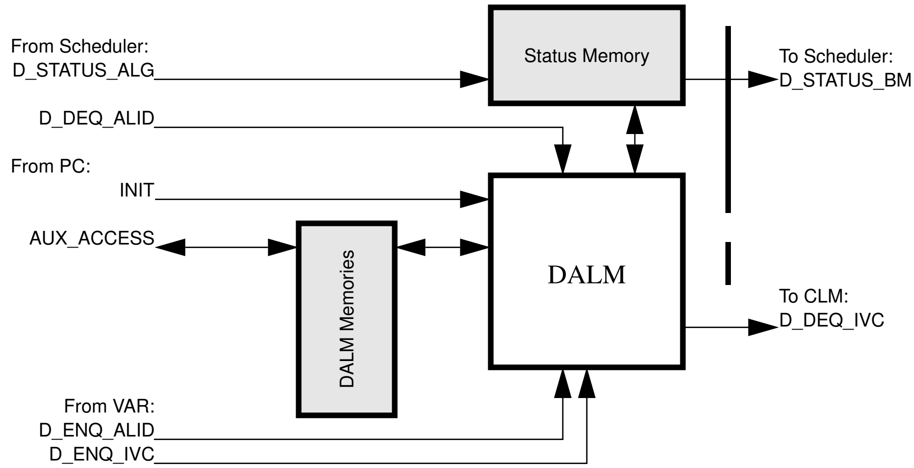
2.2 Signal definitions and names

2.2.1 CLM Signals



1. W_IVC[15:0]. This IVC number is written by the input port card when it writes an inbound cell to the core. It may have the special value EMPTY_IVC or UNEQP_IVC depending on the sending port card.
2. W_CADR[15:0]. This cell address is written by the Free List Manager (FLM) when it sees a valid W_IVC on the W_IVC bus. Because of the FLM processing delay, this may be one or more core bus cycles behind the W_IVC.
3. DEQ_IVC[15:0]. The ALM passes this IVC to the CLM to trigger Dequeues.
4. R_CADR[15:0] This is the CADR generated by dequeueing the DEQ_IVC. It is also written to the port card at the correct time slot for this port card. It may take on special values like EMPTY_CADR, CREDIT_CADR, or NOSCHED_CADR depending on the action taken by the Scheduler.
5. INIT. The PC asserts this line during initialization and when entering Post-Mortem mode. A fixed amount of time after this line is asserted, the CLM should have written consistent information to its memories and should have released control over those memories.
6. AUX_ACCESS. This is actually a bundle of signals that allow the PC to read and write CLM client memories. These signals include address strobe, chip select, r/w, address, and data. Some of these signals may be multiplexed in the final implementation.

2.2.2 DALM Signals



1. D_ENQ_ALID[15:0]. The VAR generated this signal; it specifies the target ALID.
2. D_ENQ_IVC[15:0]. The VAR generates this signal; it specifies the IVC to enqueue onto D_ENQ_ALID. This signal is NULL_IVC if the VAR does not want to perform an Enqueue on this cycle.
3. D_DEQ_ALID[15:0]: The Scheduler specifies this VC when it dequeues.
4. D_STATUS_ALG[11:0]. This specifies the active list group for which the Scheduler wants status information.It is connected to the lower-order bits of the Scheduler's Address pins. Note that these pins are just the Scheduler's lower-order address pins, so they are the same as the D_DEQ_ALID pins, with different address decoding of the higher-order address pins. .
5. D_DEQ_IVC[15:0]. The first IVC on the D_DEQ_ALID list. This is sent to the CLM.
6. D_STATUS_BM[31:0]. This is the return value from a Status query. It is connected to the 32 bits of the Scheduler's Data pins. The bits in the status query are always pessimistic, never optimistic. In other words, if a bit is set, the corresponding active list is definitely non-empty. If the Scheduler asks for Status while a Dequeue is pending, the Status request will block until the effect of the Dequeue has been propagated to the Status Memory.
7. INIT. The PC asserts this line during initialization and when entering Post-Mortem mode. A fixed amount of time after this line is asserted, the DALM should have written consistent information to its memories and should have released control over those memories.
8. AUX_ACCESS. Similar to CLM AUX_ACCESS, above.

2.2.3 CALM Signals

These are similar to the DALM signals, above.

2.3 Pin Definitions

See Shuo Huang's definitions.

2.4 Interface Timing Definitions and Timing Diagrams

See the PC design document for an explanation of the arbitrated AUX_ACCESS signals. For the remaining signals, see Shuo Huang's detailed designs.

3.0 Hardware Implementation

3.1 Detailed Block Diagram

See Shuo Huang.

3.2 Detailed Timing Diagrams

See Shuo Huang.

3.3 Internal Data Structures (ALM Example)

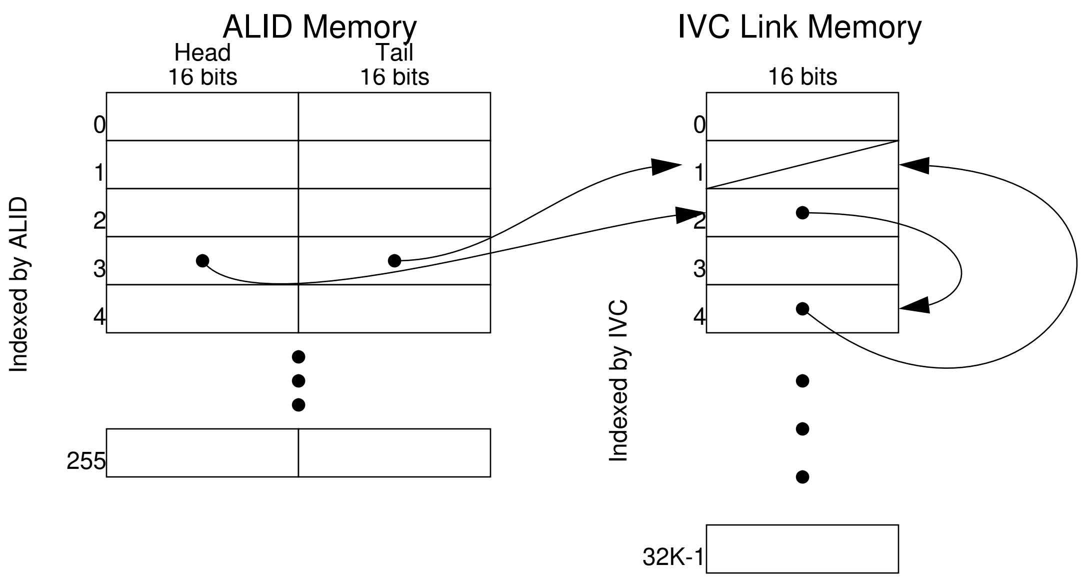
3.3.1 Head, Tail, and Link Memories

The following diagram illustrates how an active list is implemented using Head, Tail, and Link memories. Note that no overflow can occur.

Abstraction

ALID3: IVC2 → IVC4 → IVC1

Data Representation



3.3.2 Status Memory (DALM, CALM)

This dual-port memory goes through read/modify/write under normal Enqueue and Dequeue ALM operation. The read port is interlocked so that the Scheduler can request occasional status information. If a Dequeue is pending, the Scheduler will be blocked until the Status Memory is updated with the result of the Dequeue.

3.3.3 Ownership Map (CLM Only)

This memory occurs only in the CLM. It is indexed by IVC (8K tall) and is just one bit (for an 8Kx1 Single-Port SRAM). The CLM reads this memory when it sees a W_IVC on the W_IVC bus. If the bit is set, then the CLM will enqueue the <W_IVC, W_CADR> pair in its memory; otherwise it ignores the values on the bus.

3.3.4 DALM/CALM ALID Register (DALM, CALM)

This register holds the requested ALID from the Scheduler. It is interlocked, so that the Scheduler will block if it attempts to overwrite the previous ALID before it is used. It also knows when it holds valid information, so that Scheduler underflows can be detected.

3.3.5 DALM/CLM IVC Register (DALM, CLM)

This register decouples the DALM dequeue slot from the CLM dequeue slot.

3.4 Pipeline Description

3.4.1 Data Enqueue Pipeline Stages

Each List Manipulator has a three-stage enqueue operation. The first stage performs operand fetch. The second reads the appropriate memories; the third stage writes back new values to those memories. Collision detection and bypassing is implemented between the second and third stages.

Dequeues are not pipelined. In the CLM, Dequeues are interleaved (with proper bypassing) between Enqueues, which happen on alternate core clock cycles. In the DALM and CALM, the Dequeue operations occupy 3 of the 32 slots in the fixed schedule and are controlled by the central state machine.

3.5 Cost Estimates

See Shuo Huang.

4.0 Block Initialization

To initialize this block, the PC will clear all lists (set Head, Tail, and Link memories to NULL_IVC). At a signal from the MCP, the PC will change the INIT control from “OFF” to “ON”, starting and synchronizing the port card pipeline blocks to the core.

5.0 Block Diagnostics and Self Test

These will be specified by the hardware designer, Shuo Huang. Most likely, the only tests we will perform are power-on tests of the List Manipulator client memories (Head, Tail, Link, Status).

We might also specify some test transactions (via PC test signal generation) during switch initialization to ensure that the hardware blocks (and not just the memories and arbitrators) work.

Shuo Huang will design these diagnostics.

6.0 Instrumentation Hooks

The only hooks into the LM blocks are for auxiliary access. In Single-Step or Recovery Mode, this allows the PC to examine and modify the entire, consistent state of each LM. In Run Mode, the PC may take individual read snapshots of particular memories (for example to determine if this port card’s CLM thinks it owns a particular memory cell address).

7.0 Design Verification Plan

This will be created by Shuo Huang, who is designing the actual hardware.

8.0 Test Plan

Shuo Huang will supply this plan.

9.0 Delivery Schedule

Task	Status	Target Date
Block Diagram	Done	
Initialization Scheme	Done	
Diagnostics	In Progress	Ask Shuo
Instrumentation	Specified	
Design Verification Plan	Not Started	Ask Shuo
Test Plan	Not Started	Ask Shuo
VHDL Representation	In Progress	5/1/93
Test Vectors Generated	In Progress	5/1/93
Final Hardware Design	In Progress	Ask Shuo

10.0 Open Issues

Fixed Schedule: Dong or Shuo's proposal? This is only important for detailed design.