Design Specifications for Hardware Assisted Rollback Computation

USC-88-014
Richard Thomson

Abstract

An overview of rollback computation is given, followed by a series of possible design strategies to implement rollback computations in hardware. Design issues and trade-offs are examined for each of the strategies as an iteration to the chosen strategy. The implementation of the chosen strategy is discussed along with an outline of remaining work.

1.0 Overview of the Rollback Mechanism for Time Warp Simulation

Rollback computations are a method of implementation for parallel simulations.[1] The main idea of the rollback process is to maintain the simulation state as a version controlled memory. Version controlled memory (**VCM**) is accessed by both a temporal and spatial index. A version controlled memory element is accessed by a tuple (**frame**, **line**, **byte**). **Frame** is a temporal index indicating which version of the specified **byte** within the **line** is desired.

Frames are accumulated by **marking** the current frame, which saves the simulation state at that point. As the simulation progresses, more frames become marked and frames are accumulated in the VCM. Since the simulation consists of parallel processes communicating with timestamped messages, it is conceivable that a process may receive a message that took place in its local past. That is, the receiving process may have advanced its local simulation clock past the timestamp in the message received. In this case, the process must **rollback** its computation in simulated time to restore the process state corresponding to the time of the message.

Given the large size of state information in many simulations, it is not feasible to implement VCM as a physical memory. The rollback hardware is used to implement the VCM as a large virtual memory. Paging of the virtual memory to secondary storage is not performed. The usage of VCM for most simulations is such that while the entire simulation state may be large, only small portions change between successive versions, or **frames**.

[1] "Design and Performance of Special Purpose Hardware for Time Warp", Richard M. Fujimoto, Jya-Jang Tsai and Ganesh Gopalakrishnan, Fifteenth Annual Symposium on Computer Architecture, June 1988

Several ancilliary data structures are maintained to improve the efficiency of VCM management. These are the **Rollback History (RBH)** stack, **Written Bit Memory (WBM)** and the **Archive Frame (AF)**.

The RBH stack maintains a list of the "deepest" rollback since the ith rollback. In addition to the RBH stack, there is a **current rollback index (CRBI)** corresponding to the current stack element in use. The CRBI value is used in conjunction with the WBM to manage the most recent version of a line. Each stack entry, RBH_i, contains the frame number of the ith rollback. This indicates that any frames beyond that value are invalid since the computation has been "rolled back" in time to a the indicated stack value.

The written bit memory is used to determine which lines in a frame have been written since the last mark operation. These bits, if set, indicate that the simulation state associated a portion of memory has been written in the current frame. Since there is a single written bit for every line in every frame, the written bit memory is paged to allow for a smaller physical memory to be used. In addition, the WBM is organized into groups of 16 bits, or **working areas**. This allows the MRV calculation to proceed through the mark frame stack one working area at a time. There is also a rollback history index associated with each working area. The value of the CRBI is written as the index when the written bits for this working area are changed. Using the rollback history information in conjunction with the written bits, the most recent version (**MRV**) of a memory line can be obtained.

Since there are a finite number of frames, data that is older than the oldest frame is stored in a special frame called the **archive frame**. If the most recent version of a line cannot be found within the VCM, then the data either resides within the archive frame or has never been written during the course of the simulation. The AF also facilitates garbage collection of 'fossil' frames. Fossil frames are frames for which the lower bound of the simulation clock has passed, i.e. there can be no rollbacks which would cause the fossil frames to be instantiated as the current version of the simulation state. The fossil frames cannot be discarded since they may still contain the most recent version of data that hasn't been written while the simulation has advanced sufficiently to fossilize the frames. Fossil frames are garbage collected and compiled into a single frame which is the archive frame. The frames

currently in use are between the **oldest mark frame**, or **OMF**, and the **current mark frame**, or **CMF**.

1.1 The Most Recent Version Calculation

The most critical operation in the rollback computations is the computation of the most recent version of a line of data. Calculation of the most recent version of a line of data involves access to the written bits, the rollback history stack and finally the frame stack in version controlled memory.

For a given line, VCM is searched from the current mark frame (CMF) to the oldest mark frame (OMF). The written bit corresponding to the line in the current search frame is evaluated in conjunction with its rollback history information. If there was a rollback deeper than this frame, then the written bit corresponding to this frame should be cleared since the frame was invalidated through a rollback operation. If the there was no rollback deeper than the current search frame when this written bit was written, then the bit is valid. If the written bit is set, the current search frame corresponds to the most recent version of the line. If all frames between the OMF and the CMF have been examined and no valid version of the line has been found, then the MRV of the line resides in the archive frame.

The calculation of the most recent version of a data line is the heart of the special hardware that facilitates the rollback computation. The management of version controlled memory is handled through the MRV calculation: a read from VCM must produce the MRV of the line in question, while a write to VCM must insure that the newly written version of the line becomes the MRV of the line for succeeding reads.

2.0 Possible Design Strategies

Several design strategies were explored for implementation of the rollback computation hardware. The initial design was to explore possibilities of a micro-sequencer based design. This involved the examination of micro-sequencers, data paths and microsequencing cycle times for such a system.

2.1 The initial design was conceptualized as a simple data path consisting of the individual data operators residing on a common bus with control lines to activate the current operator on the bus. This design was considered to be too slow because many of the operations to be performed in the data path were simple operations that should be grouped together and performed in a single clock cycle.

2.2 This bus-oriented design was refined and the focus was shifted to the implementation of the MRV calculation. Initial designs called for the rollback history stack, the written bit memory and its associated page table to reside in a single block of static RAM. While this design economized on board space it required a three cycle approach to the heart of the MRV algorithm.

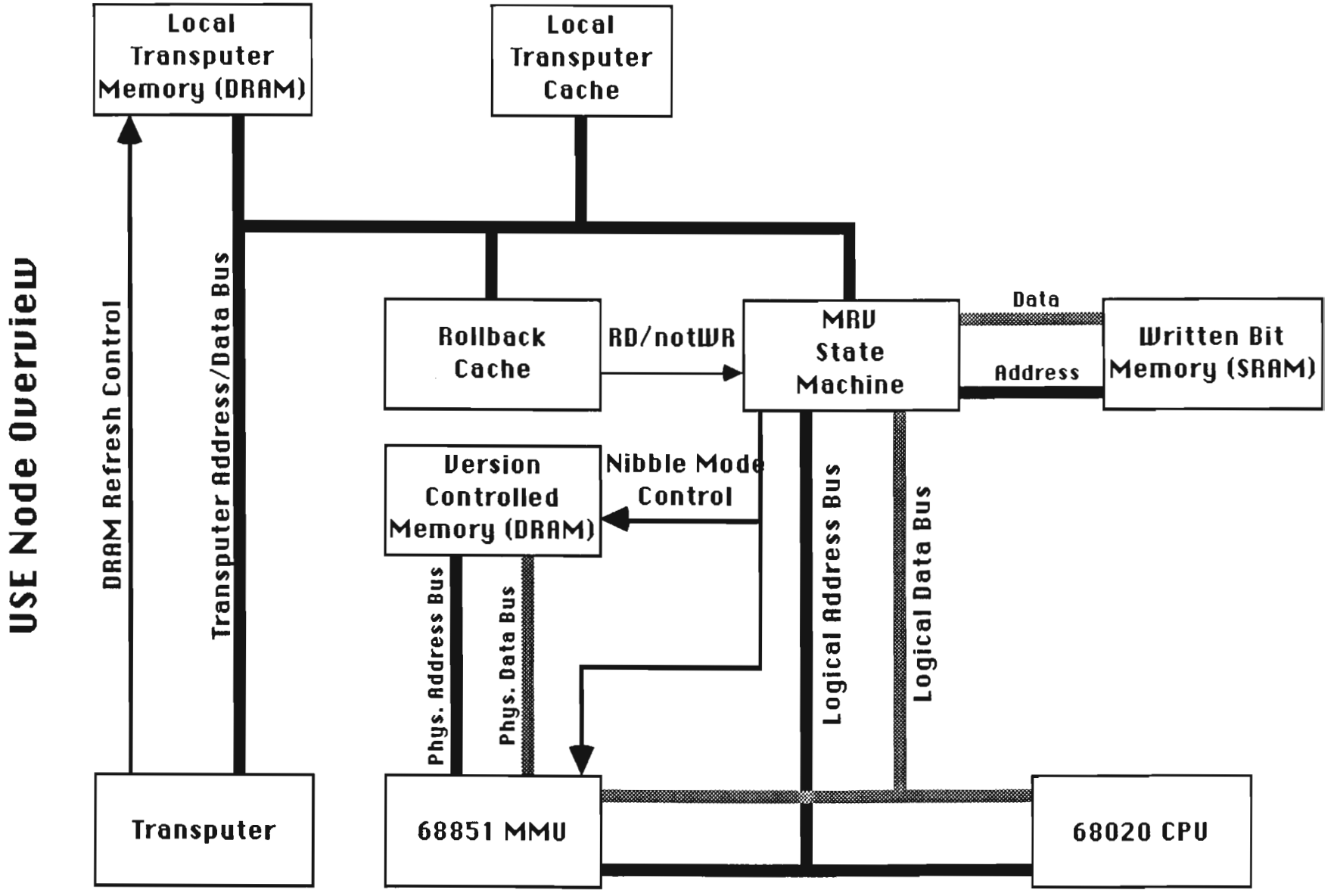
For each search frame between the CMF and the OMF, the first access would retrieve the paging information for the written bits. The second access would retrieve the written bits themselves along with a corresponding rollback history index containing the value of the CRBI at the time the bits were written. The third access would take the rollback history index and use it to retrieve the deepest rollback that has occurred since that index was written.

Once this "deepest rollback" value has been obtained, a comparison can be made to determine whether or not the written bits are valid. This is the point where written bits are cleared if necessary and written back out to the WBM.

If no rollback has occurred to invalidate these written bits, then the bits are examined to see if they contain a set bit. A set bit indicates that the frame corresponding to that bit contains the most recent version of the desired line. This frame number can then be used in an address translation to retrieve the MRV from physical memory.

2.3 An alternative to 2.2 was to implement the written bit page table, written bit memory and rollback history stack as three physically separate devices. This would allow for a full data-flow implementation of the core of the MRV calculation. While markedly simpler in terms of the implementation and design effort, it was too considered costly in terms of parts and board area.

2.4 Design 2.2 was further refined to include the Motorola 68851 Paged Memory Management Unit as the address translator for the version controlled memory. Due to the complexity of the device it was decided to add a 68020 to communicate directly with the MMU in order to initialize its address translation tables and to perform fossil collection in the background. This simplified the required capabilities of the MRV machine considerably since it need only present logical addresses to the 68851 in the form of 68020 style bus cycles to have them translated to physical addresses. This resulted in a much simpler interface to the MMU and fossil collector.


USE Node Overview
Local Transputer Memory (DRAM)
Local Transputer Cache
DRAM Refresh Control
Transputer Address/Data Bus
Rollback Cache
RD/notWR
MRU State Machine
Data
Address
Written Bit Memory (SRAM)
Version Controlled Memory (DRAM)
Nibble Mode Control
Phys. Address Bus
Phys. Data Bus
Logical Address Bus
Logical Data Bus
Transputer
68851 MMU
68020 CPU

3.0 Overview of USE Node

The USE node consists of the host processor, a T800 transputer, the custom circuitry used to implement the rollback functions and memory. The tranputer has its own local memory from which the simulation code is executed and local, non-version controlled variables are stored. To increase throughput, the transputer has its own local memory cache.

The version controlled memory is cached through the **Rollback Cache**, a direct addressed write-though cache that is maintained through the most recent version (**MRV**) state machine. When the rollback cache is read from or written to, the cache signals the MRV state machine to read or write, respectively, the most recent version of the line in question.

The MRV machine manages version controlled memory through Motorola 68020 bus cycles. The 68851 paged memory management unit is used to map the virtual address space of version controlled memory into the 4 megaword DRAM physical address space. The MRV machine also manages the written bit memory, a 64K word static RAM. In addition to bus cycles generated by the MRV machine, a 68020 is included as the fossil collection processor. The 68020 is also used to initialize the MMU with special coprocessor instructions specific to the 68000 coprocessor interface specification.

4.0 MRV State Machine Detail

The MRV state machine is the heart of the rollback hardware. It communicates through interfaces to the transputer bus, the rollback cache and the 68020 bus. It controls the management of written bit memory, the rollback history stack and the control registers containing the OMF, CMF, and CRBI.

4.1 Transputer Interface

Communication between the MRV machine and the transputer is accomplished through the use of a dual-port register file and an address latch. The register file contains the values of the OMF, CMF and CRBI registers as well as values used for written bit address translation and management of the rollback history stack. This register file is mapped into the address space of the transputer, allowing the transputer to change the value of the CMF, OMF, etc. This allows a mark operation, which simply increments the CMF, to take place on the transputer. In addition, a context switch means that all these values must be replaced for the new process. The register file approach allows the transputer to perform these operations in addition to its own process switching tasks. The address latch is used to obtain the **line** and **byte** fields from the address requested by the transputer when version controlled memory is accessed. These fields are only used when the request is not filled by the cache.

4.2 Rollback Cache Interface

The rollback cache signals to the transputer when a read or write of the most recent version of a line is necessary. In the case of a read this means that the MRV of the line is not in the cache and the MRV must be found in VCM and placed in the cache. In the case of a write the cache is indicating that a write operation just took place and the line in the cache must be written out to the CMF to maintain consistency.

The MRV machine interfaces to VCM using nibble mode DRAMs so that an entire line may be obtained with a single address assertion. A single address will be asserted on the 68020 bus by the MRV machine. This address will be translated by the 68851 to a physical address

asserted to VCM. Once this address translation has taken place and the data is placed on the bus by the DRAMs, the MRV machine can maintain control of the bus and retrieve an entire line of data from the DRAMs using the nibble mode control signals. During this time the MRV machine is the master of the 68020 bus.

4.3 68020 Bus Interface

The MRV machine communicates over the 68020 bus through an internal state machine that manages 68020 style bus cycles. This interface is used to request mastership of the bus in order to assert a VCM address to be translated. Once mastership of the bus has been granted, the internal state machine asserts the logical address onto the bus which will initiate an address translation by the MMU. If the address is translated successfully, a line of data consisting of four VCM memory words will be transferred through the MRV machine's internal data bus. The direction of data flow will be from VCM to the cache for the case of a read and from the cache to the VCM in the case of a write.

In addition to interfacing the MRV machine to the VCM through the 68020 bus, the internal state machine provides a method whereby the fossil collection processor (68020) can access values in the MRV machine's register file. These are implemented as 68020 coprocessor instructions.

4.4 Written Bit Memory Management

The written bit memory is managed by the MRV state machine itself. The written bit page table, the written bits and the rollback history stack are all contained in the same physical memory.

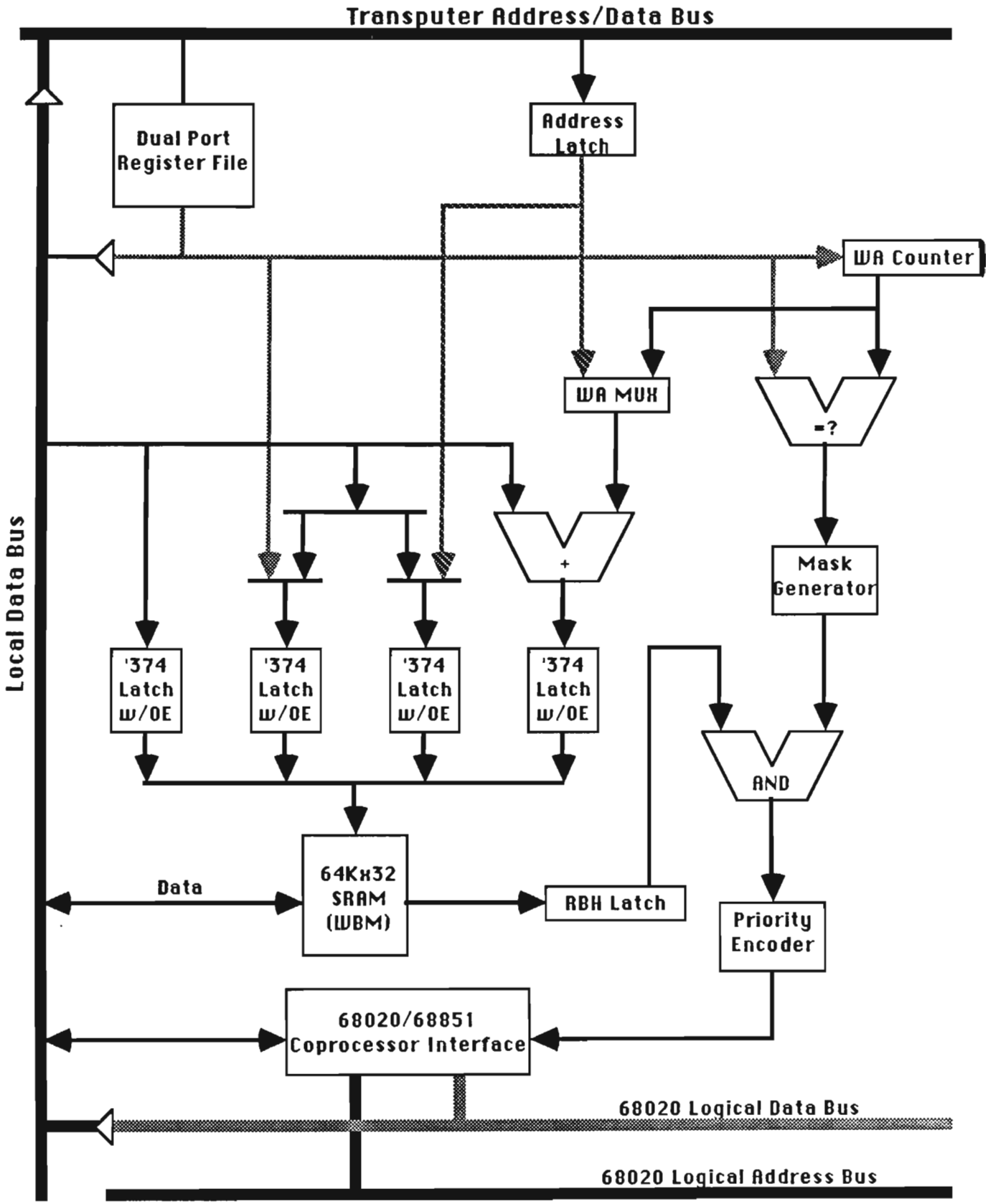
The written bits are translated through a single level of indirection. First the entry from the page table must be read by adding a portion of the requested address to the written bit page table pointer, contained in the register file. Once the page table entry has been read, the presence of a page is determined by examining the high order bit of the page table entry. If the bit is set, then the page has not been allocated. If the page has been allocated, the rest of the address is concatenated with the page table entry and used as an address to the WBM to retrieve the written bits and their associated rollback history index. The written bits are set when the MRV of a line is written to the CMF. The current value of CRBI is written as the rollback history index

when a word of written bits has been modified.

The written bit pages are allocated when a write MRV operation occurs and there is no page present for the current mark frame. A page is allocated and the written bits are written to the newly allocated page. During an MRV read if there is no page allocated for the written bits, then there are no written bits for that frame and the search can continue to an older frame whose written bits are present. If there are no written bit pages allocated for the rest of the search, then the MRV lies in the archive frame.

The rollback history stack is indexed through a rollback history index, the value of the CRBI when a group of 16 written bits are changed. This RBH stack index allows the validity of the written bits to be determined when they are read back from the WBM. This allows a rollback over many frames to occur without incurring the penalty of clearing all written bits that may have been set during the advancement of the frames that have been rolled back. The stack index is translated into a physical WBM address by concatenating the RBH index with the RBH base pointer. The RBH base pointer is contained in the register file discussed earlier.

MRU State Machine: Data Flow



5.0 MRV Data Flow Description

The entire data flow of the MRV machine is utilized in the MRV calculation. The organization of the written bit memory into words containing 16 written bits and an associated rollback history index allows the frames to be searched by working areas. A working area contains 16 frames. The MRV machine loops from the CMF to the OMF, working area by working area, until either the MRV of the line has been found or the OMF has been reached. If all frames between the CMF and the OMF have been searched and the MRV has not been found then the MRV of the line is in the archive frame. The archive frame resides in a fixed place in VCM, so its address is known immediately.

First, the MRV machine must initialize for the loop. This is done by loading the WA counter with the value of the current working area. The current working area is kept in a register in the register file. At this time, the coprocessor interface state machine may begin arbitrating for the 68020 bus, anticipating the address translation process.

A portion of the requested address and search working area is passed through the WA multiplexor to an input of the adder. The other input of the adder is received from the local data bus and contains the base pointer to the written bit page table. The result of the adder is latched onto the address lines of the WBM, while the page table entry on the data lines of the WBM are driving the bus for the next cycle.

The high order bit page table entry is examined to see if the page has been allocated. If no page has been allocated, the search continues onto the next working area. If the page has been allocated, the page table entry is concatenated with the remaining bits of the working area and address. This concatenated value is then asserted as the address to the WBM, while the written bits and rollback history index are driven onto the local data bus by the WBM data lines. The written bit latch is also enabled and the written bits are stored here for possible use in determining the frame number of the MRV.

The rollback history index is combined with the rollback base pointer, obtained from the register file, to produce the address corresponding to that entry in the rollback history stack. This address is presented to the WBM address lines and the resulting stack value is then routed to the working area comparator.

The comparison of the rollback stack value and the current search working area determine if the written bits need to be examined to determine if they may contain the MRV.

If the stack value is less than the search working area, this means that a rollback has occurred which invalidated these bits and the search may continue to the next working area.

If the working area values are equal, or the stack value is greater than the search working area, the written bits may contain the MRV. If the values are equal, the frame number of the rollback history value is used to prepare a mask so that the invalid written bits in the word may be cleared. The result of the comparison is fed to the mask generator which generates a mask that is anded with the written bits. This mask is all ones if the stack value is greater than the serach working area. The masked written bits are then priority encoded to determine the MRV of the line. If no bits are set in the masked result, then the search continues to the next working area.

If there is a set bit in the masked written bit word, then the priority encoder gives the frame within the working area containing the MRV of the line. This frame number, along with the current search frame is presented to the 68020 coprocessor interface so that it may request an address translation for the MRV of the line. When the data is received from the VCM, it is gated through the local data bus into the rollback cache. Four data transfers occur, moving an entire line from VCM into the rollback cache.

6.0 Further Work

Remaining work that needs to be completed is the full implementation of the 68020 coprocessor interface. This will consist of a bus arbitration state machine, coprocessor interface state machine, coprocessor registers and a simple interface to the MRV state machine.

The full control state of the MRV machine needs to be mapped out to handle the flow of control needed in the MRV read/write operations as well as handling management of the written bit pages. The suggested scheme for managing the pages is to maintain a list of free pages from which new pages will be allocated by removing the head from the list and writing the pages base pointer into the page table. Simple control sequences should also be provided to allow the transputer/68020 to access the internal data structures of the MRV machine to facilitate debugging.