A SNOOPING CACHE COHERENCE PROTOCOL FOR A RING CONNECTED MULTIPROCESSOR

Luiz A. Barroso and Michel Dubois

Technical Report 91-03

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089
(213)740-4475
barroso@priam.usc.edu; dubois@priam.usc.edu

February 11, 1991

Abstract

The Express Ring is an architecture project under development at the University of Southern California. Its main goal is to demonstrate that a pipelined unidirectional ring with very fast point-to-point interconnections can be at least ten times faster than a shared bus, using the same technology, and may be the topology of choice for future shared-memory multiprocessors. In this paper we present a snooping cache coherence protocol for the Express Ring architecture, that shows how consistency of shared memory accesses can be efficiently maintained in a ring-connected multiprocessor. We analyze the proposed protocol and compare it to other more usual alternatives to point-to-point connected machines, such as the SCI cache coherence protocol and directory based protocols. The snooping protocol is not only the least expensive but it also outperforms the other alternatives by showing shorter latencies and generating less traffic on the average cases.

1. Introduction

It is a well known fact that bus-based architectures, which are the most popular topologies in current commercial systems, have serious electrical problems that have kept them from reaching higher bandwidths, thus seriously limiting their scalability [Benn90][DelC86]. Moreover, processors clocked at 40 MHz are already available on the market, and we believe that even a small number of these could generate enough traffic to saturate current interboard buses, in a caching shared-memory multiprocessor. On the other hand, point-to-point unidirectional interconnections can be made much faster than bus interconnections since they do not require arbitration cycles, and lack most of the electrical problems with shared buses, namely signal propagation delays and signal reflection/ attenuation. Current IEEE standard proposals are based on 500 Mbps, 16 bits wide unidirectional point-to-point interconnects [SCI90]. One of the goals of the *Express Ring* project at USC is to build a cache-coherent shared-memory multiprocessor with the simplest type of point-to-point interconnection network: a unidirectional ring. We believe that such a ring can be made at least ten times faster than state-of-the-art buses, therefore showing much better scalability.

The use of local caches in a shared-memory multiprocessor greatly enhances overall memory access performance, since a large fraction of processor references is likely to be present in the caches, and is resolved at processor speed. Private caches also affect performance by decreasing the traffic in the interconnect, thus avoiding conflicts and queueing delays. However, in order to enforce a consistent view of multiple cached copies of shared writable data, one requires a *cache-coherence protocol*. In this report we develop a cache-coherence protoco for a pipelined unidirectional ring-connected multiprocessor. We believe that this protocol is a simple and quite efficient solution for the coherency problem in a ring architecture, and we compare it with other protocol options, such as directory based [Sten90][Chai90] schemes and the SCI standard cache protocol [Jame90].

Snooping protocols [Good83][Katz85] are very popular in bus-based systems, mainly because of their simplicity and low cost of implementation. Unfortunately this class of protocols rely heavily on the broadcasting of information, and is inadequate for most point-to-point interconnected systems, where broadcasting is generally very expensive. Several cache-coherence protocols for point-to-point interconnected machines have recently been proposed in the literature [Sten90], most of them making use of directories to keep track of the copies for every memory block. These schemes are relatively complex to implement. Because of the natural broadcasting structure of the Express Ring, we believe that an adapted version of a snooping cache coherence protocol may perform as well as any directory based mechanism, with the advantages of being much simpler to implement and scale to larger configurations.

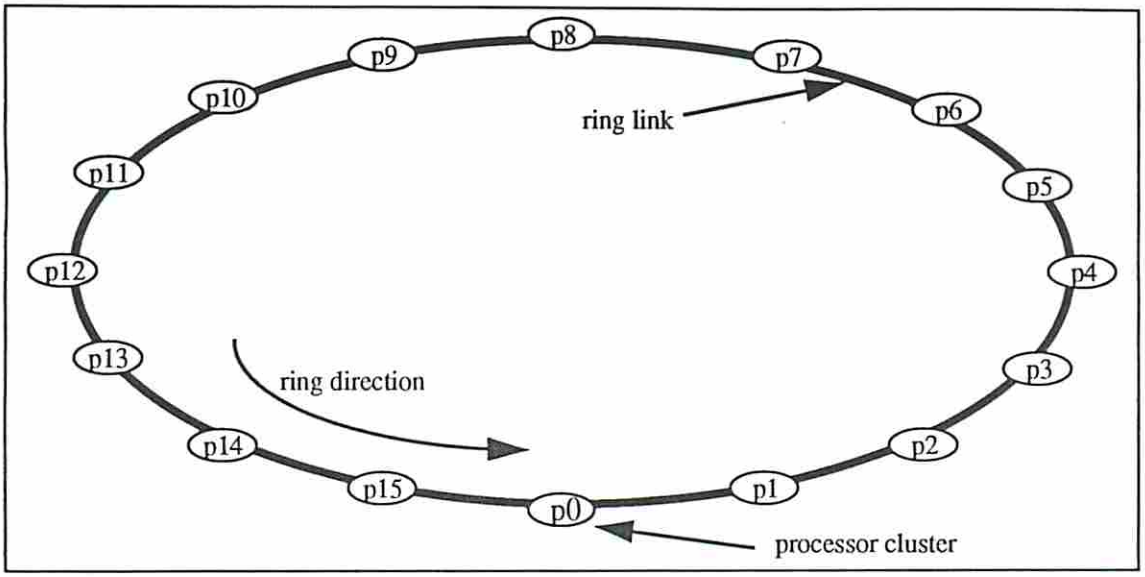
In the next Section we briefly describe the Express Ring architecture, its pipeline features and memory organization. In Section 3 we present a cache coherence protocol for this architecture, including some implementation issues. A detailed description of the ring interface behavior is shown in the Appendix. The performance characteristics of the proposed protocol are presented in Section 4, in which we also compare it with other protocol alternatives in the context of the Express Ring architecture. Final remarks are drawn in Section 5.

2. The Express Ring Project

The Express Ring project at USC intends to demonstrate that a high-speed pipelined ring is not only feasible but may be superior to a shared bus as the interconnection structure for shared-memory multiprocessors. An important feature of the Express Ring Architecture is that, as far as the programmer is concerned, the system behavior is identical to a bus-based architecture, since all ring transactions experience the same latency. This characteristic, together with the use of relatively large private caches, makes the Express Ring suitable for general purpose computing.

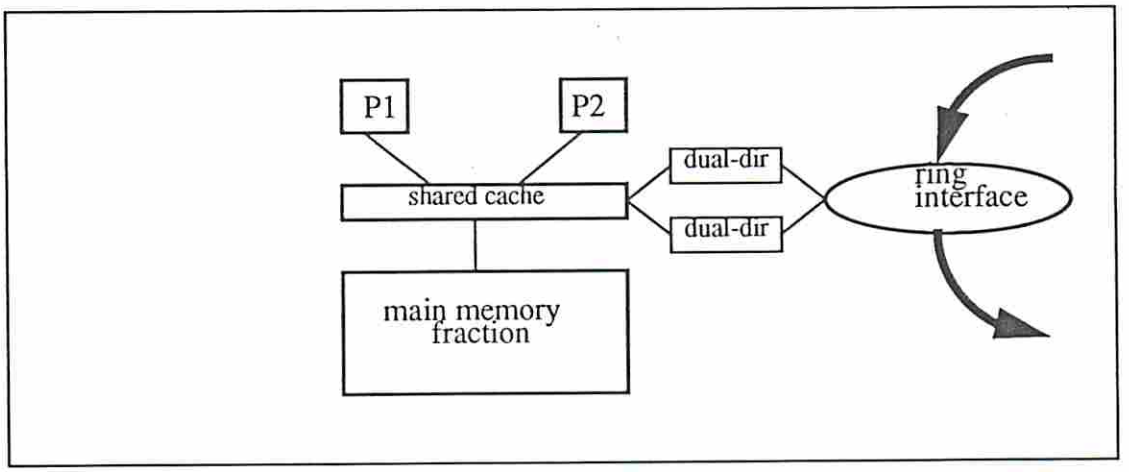
The Express Ring architecture, which is schematically shown in Figure 1, consists of a set of high-performance processor clusters interconnected by a pipelined ring structure. Each processor cluster, as depicted in Figure 2, consists of 2 processors connected to a shared dual-port cache, a fraction of the system's physical memory space (including main memory and swapping disk space) and the ring interface. Transfers on the ring are synchronized

by the same clock, but they are asynchronous to the processors operation.



The Express Ring Topology
Figure 1.

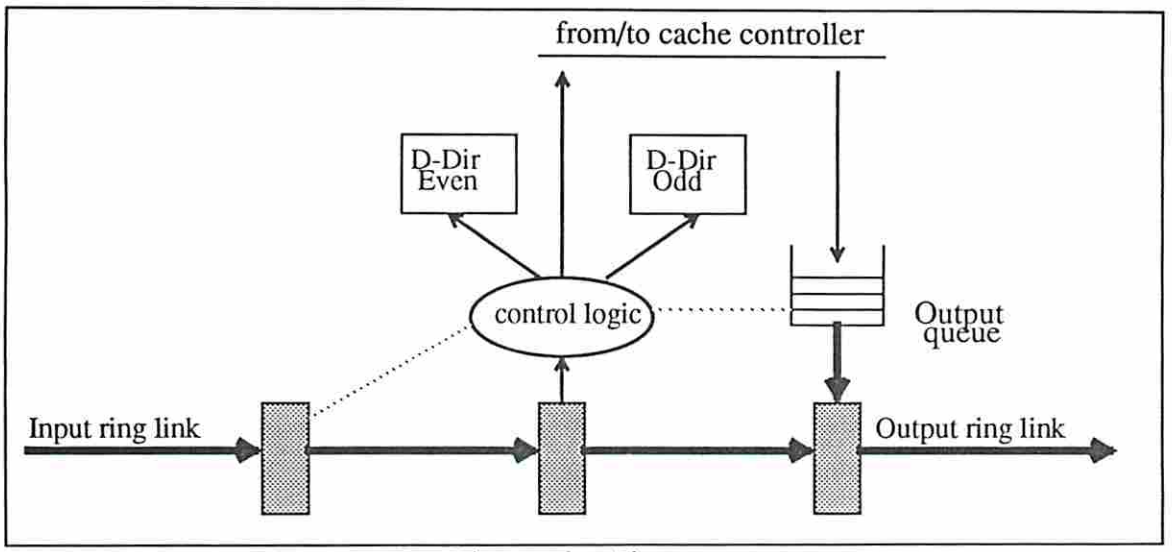
Using a ring topology to interconnect processing elements is not a new idea. Extensive research have been done in the design and analysis of ring-type interconnections for local area networks (LANs), such as the token ring [Farb72] and the slotted ring [Pier72] networks. Ring interconnections have also being studied in the context of more tightly-coupled systems, as in the CDC CYBERPLUS system [Ferr87] and the MIT Concert Multiprocessor [Hals86]. However, we are not aware of the use of a pipelined ring network as the interconnection of a shared-memory cache coherent multiprocessor system. The main benefit of pipelining the ring is that several messages can be transmitted at the same time, which increases the communication bandwidth.



Processor Cluster diagram
Figure 2.

The Express Ring interconnection network can be viewed as a circular pipeline, with 3*P stages, where P is the number of processor clusters (or nodes) in the system, i.e., each processor cluster accounts for 3 pipeline stages, as shown in Figure 3. At each ring clock cycle, 64-bit wide packet slots are transferred from a stage of the pipeline to the next one. Messages are inserted in the ring by filing packet slots with useful information, and removed by marking a packet slot as empty. There are 3*P packet slots circulating in the ring. The pipeline clock can be extremely fast, since no centralized arbitration is required for the packet slots: a processor cluster may use the first

empty slot that passes through it to send a message. Such a scheme may lead to starvation of a cluster, specially when the ring traffic is relatively heavy, and the utilization of the slots approaches 100%. A simple and effective mechanism to prevent starvation, without requiring any central arbitration or reservation of slots, is discussed in Section 3.



Ring Interface Diagram
Figure 3.

Routing in this architecture is very straightforward. An incoming message is either consumed by the cluster or forwarded to the next cluster in the ring order. The ring interface has a very fast logic that scans the first bits of each message and determines whether it has to be forwarded or not. A block diagram of the ring interface is shown in Figure 3. It is important to note that the speed of the forwarding logic is critical in the Express Ring design, and it is likely to become the bottleneck on the pipeline clock frequency.

There is a single physical address space shared by all processors in the system. The memory is distributed among the clusters in the ring and the location of a memory block is determined by the higher address bits. The processor cluster to which a block is mapped is called the block's *home cluster*. Swapping space is provided by disks attached to some of the clusters. A single bit, called *dirty* bit, is associated with each cache block frame in the main memory. The dirty bit will be used to indicate if the current main memory copy of a block is updated or not, and it is critical to the performance of the proposed coherence protocol.

There is a dual-directory to serve coherence requests coming from the ring, that contains a copy of the local cache directory (tags + state information), so that snooping activity does not interfere with normal processor accesses to the cache. The dual directory is 2-way (even/odd) interleaved to be able to keep up with the rate at which coherence messages arrive from the ring. Caches may contain blocks that map to the local cluster's physical space or to any other cluster's memory. Both cache misses and invalidations may generate messages to the ring in order to satisfy local accesses and maintain global system coherence.

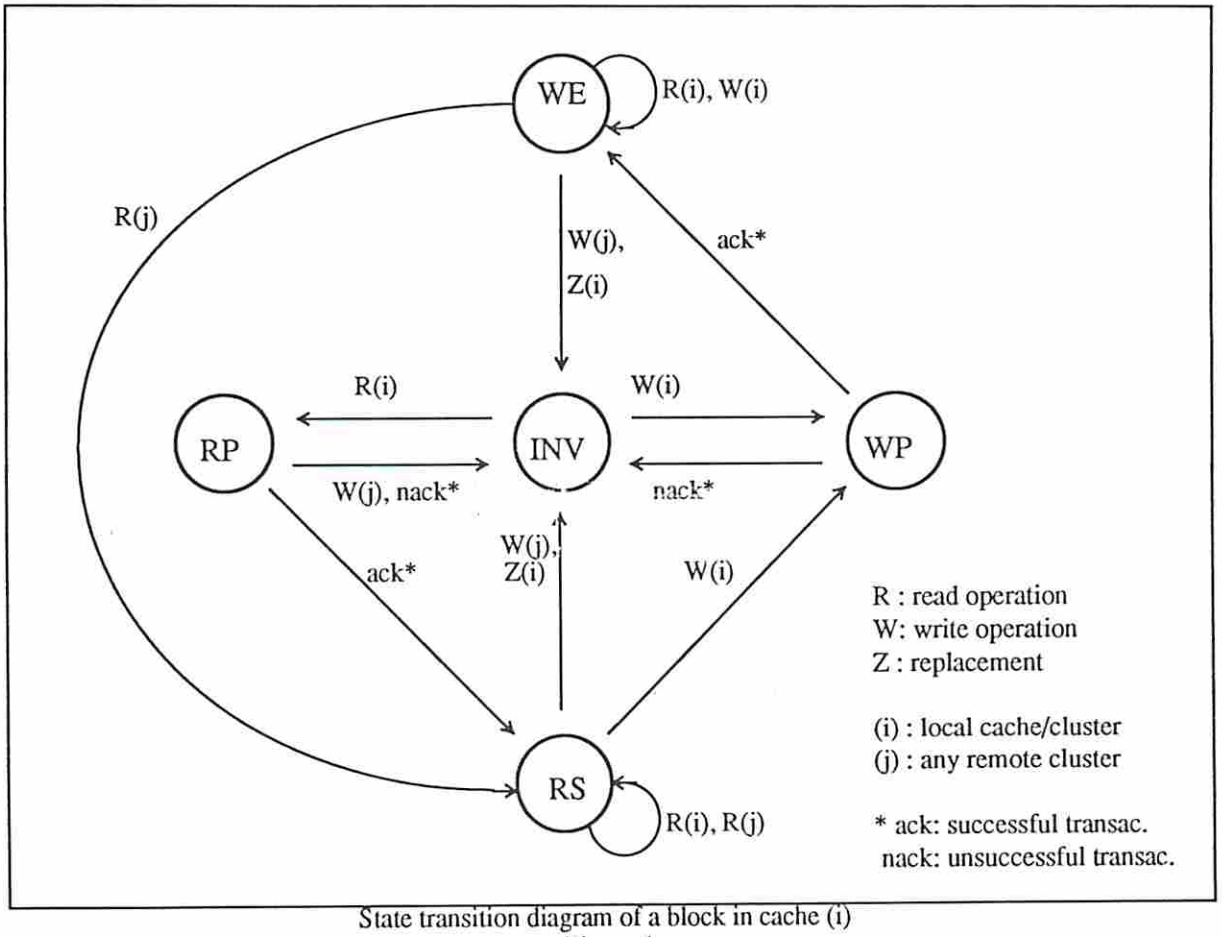
In order to demonstrate the feasibility of the Express Ring architecture we have to show that cache-coherence can be efficiently implemented in a pipelined unidirectional ring interconnection network.

3. The Express Ring Cache Coherence Protocol

3.1 Description

In this Section we present a snooping cache coherence protocol for the Express Ring architecture. We propose a write-back/write-invalidate protocol with five cache block states and two memory block states. Three

of the cache block states, *Invalid* (INV), *Read-Shared* (RS) and *Write-Exclusive* (WE), are taken from the basic write-back/write-invalidate snoopy protocol. A cache block in INV state is assumed to be not present in the cache. A cache block in RS state can be accessed for reads but not for writes; several RS cached copies of a block may exist at the same time. A cache block in WE state can be accessed for both reads and writes, and the protocol ensures that when a block is in WE no other cache has a valid (WE or RS) copy of it. Transitions among these three states are clearly atomic in a bus-based system, since at most one coherence transaction can be performed on the bus at a time, being acknowledged (or aborted) in the same bus operation. However, the situation in the Express Ring is not so simple because multiple messages may be traversing the pipelined ring at the same time and a cache block in transition may see multiple coherence messages before completing its transition. In our protocol, such non-atomic transitions are represented by two extra cache states: *Read-Pending* (RP) and *Write-Pending* (WP)[1]. A cache block in RP (WP) state is in transition to a RS (WE) state but has not yet committed, i.e. has not yet received the corresponding acknowledgment. An acknowledgment for a miss is a message that provides the missing block, and the acknowledgment for an invalidation is a pair of signals that indicate that all system caches successfully invalidated their copies of a block. A cache block in RP (WP) state may also abort its transition to RS (WE) state if some kind of negative acknowledgment is received. We discuss the acknowledgment mechanisms in detail later in this section.The diagram in Figure 4 shows the possible states of a block in a given cache (i). W(i) and R(i) denote respectively write and read operations generated at the local cluster (i), and (j) denotes any cluster other than (i).



State transition diagram of a block in cache (i)
Figure 4.

The two memory states indicate if the current main memory copy of a block is valid or not, and they are implemented by the dirty bit described in the previous section. The memory copy of a block is not valid when some system cache has a copy of that block in WE state. We note that when a block's memory state is valid, there may or may not be cached copies of it. The two memory states are named *modified/unmodified*. The five cache block states

1. WP and RP states are important since we want to allow more than one request to be pending at a given time.

may be encoded in three bits: a *valid* bit, a *pending*[1] bit and a *write* bit. The specific codes for the cache states are shown in Table 1. Note that there are two codes for the WP state, since it may be reached from either INV or RS states. When it is reached from the RS state, the cached copy is assumed to be valid, but when it is reached from the INV state, there is no valid copy in the cache.

codes			state
V	P	W	
0	0	x	INV
0	1	0	RP
0	1	1	WP (from INV)
1	1	1	WP (from RS)
1	0	0	RS
1	0	1	WE

V=1 (valid); V=0 (invalid)
P =1 (pending); P=0 (not pending)
W=1 (read/write); W=0 (read only)

Encoding of cache states
Table 1.

Coherence actions are procedures that have to be enforced by the coherence protocol in order to preserve an overall consistent view of the shared memory. Coherence actions are triggered by one of the following events: read miss, write miss, invalidation and replacement. Below we explain the protocol behavior in further detail by describing the state transitions and data movement involved in every coherence action.

Read Miss: Occurs when a processor attempts to read from a block that is not present in its cluster's cache (i.e. a block in INV cache state). A read miss immediately changes the cache state of the missing block from INV to RP. The requesting cluster's cache state changes to RS when the missing block is received. If the memory block state is *unmodified* the home cluster will provide the block to the requesting cluster, and the memory block state remains *unmodified* after the transition is completed. Otherwise a dirty cluster (a cluster with the WE copy) will send the valid copy to the requesting cluster and to the home cluster, so that the memory copy will also be updated. The cache state in the dirty cluster changes to RS and the memory state changes to *unmodified*.

Write Miss: Occurs when a processor attempts to write on a block that is not present in its cluster's cache (i.e., a block in INV cache state). The local cache state of the missing block goes immediately to WP, and changes to WE when a valid copy of the block is received. In a write miss, not only a copy of the block has to be fetched but also all other existing copies have to be invalidated. The block is provided by the home cluster if the memory state is *unmodified* or by the dirty cluster, if the memory state is *modified*. The final cache state in all clusters other than the requesting one is INV, and the final memory state is *modified*.

Invalidation: Generated when a processor attempts to write on a block that is present in its cluster's cache but in RS state. An invalidation action will ensure that the cluster has a unique copy of the block before it is allowed to write on it. The requesting cluster cache state goes immediately to WP, and changes to WE when a positive invalidation acknowledgment is received. One should note that invalidations only happen when the memory state is *unmodified*. The final cache state in all clusters other than the requesting one is INV, and the memory state changes to *modified*.

Replacement: Generated when a cache needs the block frame that is presently occupied by a block in WE state (i.e., a

1. A pending bit associated with each block frame is clearly not necessary. We use it here to simplify the description of the protocol.

block that has been altered). The block has to be *flushed* to the home node, so that the updates are not lost. The requesting cache state goes immediately to INV, and the memory state changes to *unmodified*. Blocks in any other cache state (WP, RP, RS and INV) do not have to be written back since they have not been altered.

In some situations, a cache block in RP or WP state may not reach its respective final state because of conflicts. For instance, if at a given moment more than one cache is in WP state, only one of them will be allowed to change to WE and all others will become INV. Another conflicting situation happens when there is at least one cache in RP state and at least one other cache in WP state. Depending on specific timings, the caches in RP state may or may not be allowed to change to RS state. Such conflicting situations are a result of the lack of centralized control in the Express Ring architecture, and they cannot occur in systems with centralized arbitration like bus-based multiprocessors. However those conflicts are resolved with very little performance impact in the Express Ring performance. Moreover, they tend to be rare since the average number of invalidated caches per write operation has been observed to be very low for a representative number of parallel algorithms [Agar88].

3.2 Implementation

We now discuss some implementation issues of our protocol, also detailing its behavior in conflicting situations. The protocol is implemented as a set of *coherence messages* exchanged through the pipelined ring in order to implement the coherence actions described earlier. A miss in a cache block that is mapped to the local cluster's physical space (i.e., the requesting cluster and the home cluster are the same) is called a local miss. A local read miss does not generate a coherence message to the ring if the block's memory state is *unmodified*. In all other situations misses, invalidations and replacements cause coherence messages to be sent to the interconnection structure. A summary of the protocol coherence messages is shown in Table 2.

Event	Command	Response
read miss	Read-Block	Send-Block Send-Block-Update
write miss	Read-Exclusive	Send-Block
invalidation	Invalidate	...
replacement	Send-Block	...

Protocol Commands/Responses

Table 2.

Coherence Messages

For the description that follows we should recall the definitions of *home* cluster, *dirty* cluster and *requesting* cluster. The home cluster of a block is the cluster that contains the physical memory partition to which the block address maps. The dirty cluster of a block is the cluster that has the WE copy of that block (note that many times there will not be any dirty cluster for a given block). The requesting cluster is the cluster that initiates a coherence action by sending one of the commands listed in Table 2.

Responses to a *Read-Block* message (generated when a read miss occurs) vary depending on the current memory state of the block. If the block's memory state is *unmodified*, the main memory contains a valid copy and provides it to the requesting cluster by replying with a *Send-Block* message. However if the block's memory state is *modified*, it has to be provided by the cluster which cache contains the WE copy, i.e. the dirty cluster. In this situation, the dirty cluster replies with a *Send-Block-Update* message that contains the requested block. The requesting cluster forwards a *Send-Block* message to the block's home node when it receives the *Send-Block-Update* message, so

that the main memory can be updated. A *Read-Exclusive* message is generated when a write miss occurs, and it requests an exclusive copy of the block. Either the dirty cluster (if the block is *modified*) or the home cluster (if the block is *unmodified*) replies to a *Read-Exclusive* message by providing a valid copy of the block. In this case, the reply is always a *Send-Block* message, even if the block's memory state is *modified*. In other words, we do not update the main memory copy of a block when a WE copy simply migrates from one cluster to another. A Read-Exclusive message will also invalidate every other copy of that block in the ring, since the required copy will become *dirty* as soon as it arrives at the requesting cluster. An Invalidate message is generated when a cluster attempts to write to a block in RS state in its local cache. In this case there is no need to fetch a copy of the block, but it is important to invalidate all other cached copies of it.

Message Formats

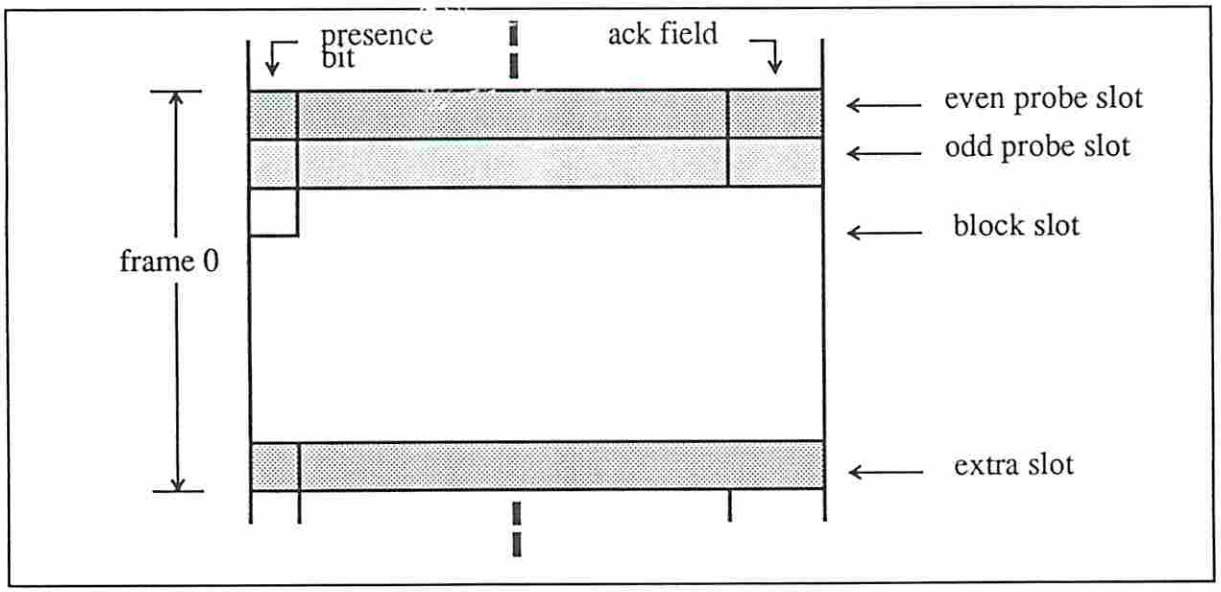
There are two message formats that will traverse the ring. *Read-Block*, *Read-Exclusive*, and *Invalidate* are very short messages, named *probe* messages, with the following format:

| <message type> | <requester address> | <block address> | <ack> | = 44 bits
| 4 | 6 | 32 | 2 |

Send-Block and *Send-Block-Update* messages (called *block* messages) are larger since they carry a cache block. Here we consider a cache block size to be eight 32-bit words (256 bits). The format of the block messages is:

| <message type> | <requester address> | <block address> | <block> | = 298 bits
| 4 | 6 | 32 | 256 |

Probe messages fit in a single 64-bit packet, and occupy only one pipeline stage (a *probe slot*) in the Express Ring, while block messages will occupy 5 consecutive pipeline stages (a *block slot*). Probe slots and block slots are arranged in *frames*, that circulate through the pipelined ring. Frames are composed of one probe slot for blocks with even addresses (even probe slot), one probe slot for blocks with odd addresses (odd probe slot), one block message slot (that can be even or odd), and an extra slot that is used for intercluster interrupt signals. The frame format is shown in Figure 5.



Frame organization
Figure 5.

We say that a given slot is empty if no valid information is currently being transmitted through it. A

cluster wanting to transmit a probe has to wait until it detects an empty probe slot of the same parity. A block message may be transmitted using any empty block slot. There are three basic motivations to arrange probe and block slots in frames:

(1) Enforce a minimum interarrival time for probes with same parity, giving enough time for the interleaved dual-directory to respond. Using frames, the minimum interarrival time for probes with same parity will be eight ring cycles, which gives 40 nsec. (ring clock freq. of 200 MHz) for the dual-directory to respond.

(2) Eliminate the problem of having to deal with two different size messages on the ring, making arbitration simpler. If a pipeline slot could be used for probes and block messages a cluster waiting to transmit a block message wouldn't be able to determine if there are enough consecutive empty slots to transmit its message when it sees an empty slot. Transmitting block messages in non-consecutive pipeline slots would make the ring interface much more complex, and that would certainly slow down the ring clock.

(3) Provide an efficient acknowledgment mechanism for probes. As we can see in Figure 5, there is an acknowledgment field in every probe slot. Acknowledgments for an even probe in frame i will be put in the <ack> field of the even probe of the subsequent frame (frame i+1). The reason why acknowledgments for a probe are not "piggybacked" in the same probe is to allow the ring interface logic enough time to respond.

A probe message traverses the whole ring exactly once, being consumed by the cluster that issued it. This is necessary because all the cluster's directories have to be consulted in a miss/invalidation, following the snooping scheme. However, a block message does not circulate through the whole ring, but only through the path between its source and destination. Assuming balanced access patterns, a block message traverses one half of the ring on the average, and this is the reason why we have chosen to put two probe slots and one block slot per frame. The sender of a probe and the destination of a block message are responsible for removing them from the ring. A message is removed by marking its slot as empty.

When a probe message passes through a cluster it generates a cache directory lookup relative to its nature (read miss, write miss or invalidation) but does not wait for the result of the lookup, being forwarded to the next cluster without delay. Any acknowledgment for a probe uses the <ack> field of the next probe of same parity. This mechanism was included to resolve conflicting situations that may arise when probes for the same block are issued by distinct clusters in a very short time interval. It also allows a cluster to reject a probe when it is busy and cannot snoop.

Conflict Resolution

Conflicting situations occur when a cache receives a probe for a block that is currently in a pending state (RP or WP). The protocol has to be able to resolve those situations not only keeping the global state of the block consistent, but also avoiding deadlocks, livelocks and starvation of a cluster. To be able to do that, we included the <ack> field, consisting of two bits in the respective probe slot of the next frame. Those bits are defined as:

a-bit : acknowledgment bit. Set by the home cluster or the dirty cluster of the block.
ns-bit: no-snoop bit. Set by a cluster that was not able to consult its directory for a probe.

Both the a-bit and the ns-bit are reset by the requesting cluster. The a-bit is set by the cluster that has the valid copy of a block being requested by a *Read-Block* probe (which can be either the home cluster or the dirty cluster) signaling that the request was received and that the requested block will be provided. When the requesting cluster sees it coming back, it removes it from the ring and waits for the acknowledgment that comes in the next frame. If the a-bit is set it waits for the requested block to arrive (still in RP state), otherwise it re-issues the probe in the next available slot. A situation where no cluster acknowledges a *Read-Block* probe may happen when the valid block is currently being sent from the home cluster to the dirty cluster or back. In this case, during a short time interval, no cluster actually has a valid copy, and consequently no one sets the probe's a-bit. Another situation is when the cluster with the valid copy is busy and cannot snoop the probe. When this occurs, the ns-bit of the respective <ack> field will be set. In both cases, it is likely that the re-issued probe will be successful. However, even when the *Read-Block* probe

has been acknowledged, the transition to RS can still be aborted. When a *Read-Exclusive* or an *Invalidate* probe for a given block passes through a cluster that has that block in RP state, it immediately invalidates that block entry. If later the requested block arrives at the requesting cluster as a *Send-Block* message, it is simply discarded and its block slot is marked as empty. If the requested block arrives as a *Send-Block-Update* message it is ignored by the local cache but it is forwarded to the home cluster by changing it into a *Send-Block* message to the home cluster. It is important to note that if the *Read-Block* probe had been acknowledged the requested block would arrive some time later, and if the probe had to be re-issued, the first copy of the block to arrive would be discarded.

Another hazardous situation occurs when multiple *Invalidate* or *Read-Exclusive* probes are issued concurrently. In this case, the cluster with the valid copy (either the home or the dirty cluster) serves as the *arbitrator cluster*. The probe that reaches the arbitrator cluster first will have its a-bit set. The cluster that receives the probe with the a-bit set "wins" the arbitration and is allowed to write on the block (i.e., change from WP to WE state). The remaining clusters receive the a-bit reset and they have to re-issue the request. Note that a cluster that had sent an *Invalidate* probe has to retry with a *Read-Exclusive* probe, since it no longer can assume that its previous RS copy of the block is valid. A cluster that has sent an *Invalidate* probe ignores other probes to that same block until it receives the respective <ack> field, since until then it does not know who is the winner of the arbitration. A cluster that has send a *Read-Exclusive* probe and has won the arbitration still has to wait until it receives a valid copy of the block. Only then the cluster's cache state changes to WE.

The ns-bit is set by a cluster that has its directory/cache busy, and cannot snoop the probe request. This ns-bit is not important when the probe is a *Read-Block* message, since the only cluster it needs to snoop is either the home cluster or the dirty cluster, and both of them already respond through the a-bit. However, if a cluster that sent an *Invalidate* or *Read-Exclusive* probe receives both the a-bit and the ns-bit set in the <ack> field, although it won the arbitration, its probe may have not invalidated a RS copy of that block, and another *Invalidate* probe has to be issued and received with the ns-bit reset. Only then the requesting cluster is allowed to write. Note that only the ns-bit has to be checked when the second *Invalidate* probe returns. While the cluster that won the arbitration is trying to make sure it had invalidated all other copies of that block, i.e., it is waiting to receive a probe acknowledgment with the ns-bit reset, no conflicts can occur since no probe to that block receive a positive acknowledgment (i.e. the a-bit set) until that cluster's copy actually changes to WE state.

While we don't think that fairness will be a serious problem in the Express Ring interconnection, it is possible that a given cluster has to wait an arbitrary long period of time until it finds an empty slot to send a message. If that turns out to be a problem for an specific application the following policy can increase fairness in slot allocation. We simply enforce that every time a cluster removes a message from the ring it does not use that slot but passes it to the next cluster, even if it has messages to transmit. This mechanism works like a token passing scheme, where the token is actually the empty slot. The upper bound on the time to acquire an empty slot of any kind will be a small multiple of the total ring traversal time, since there are multiple frames in the pipelined ring (for the Express Ring prototype there are 6 frames in the ring).

A detailed description of the functional behavior of the ring interfaces is presented in the Appendix I.

4. Protocol Evaluation

In this section we calculate the expected latencies to satisfy coherence messages in the Express Ring cache coherence protocol, as well as in a directory based protocol and in the SCI cache coherence protocol. We compare the three protocols in the context of the Express Ring architecture, i.e., a unidirectional pipelined ring interconnect. Our analysis will not consider contention for resources and the subsequent queueing delays, thus it can be seen as an upper bound approximation for the coherence transaction times. We also don't consider the case when a local read miss occurs in an unmodified block, since no coherence message is generated by any of the protocols. Brief

descriptions of centralized directory and the SCI protocols are given below.

4.1 Centralized Directory Protocols

In this class of protocols every coherence message is directed to the block's home cluster, which keeps track of the state and location of all cached copies of the block. We assume a write-invalidate directory protocol, since we want to compare it the proposed protocol which is write-invalidate. In directory protocols, the home cluster allows sharing of a block for read-only copies but enforces exclusive access for writable copies. The home cluster satisfies all misses if there is no writable (dirty) copy of the block, forwarding the request to the dirty cluster otherwise. When receiving an invalidation, the home cluster sends invalidation messages selectively to all the clusters sharing a particular block, and replies to the requesting cluster once all copies have been invalidated. The main difference between directory and snooping schemes is that directory-based protocols do not rely in broadcasting of coherence information.

4.2 SCI Cache Coherence Protocol

The SCI standard which is being defined by IEEE proposes a type of distributed directory protocol in which a linked list of clusters is maintained for each cached block, called the sharing list.The list is implemented by forward and backward pointers in each cache block entry. Coherence requests are sent to the home cluster which satisfies them if the block is currently not cached. If the block is being cached, the home cluster forwards the request to the cluster in the head of the list for that block, which will take the appropriate coherence actions and respond to the requester. In case of a read miss, the missing cluster is inserted at the head of the list either by the home cluster or the previous head. In case of a write miss or an invalidation, all clusters are sequentially removed from the sharing list until the writing cluster is the only one left.

4.3 Latency Analysis

The architectural parameters of the Express Ring that are used in our analysis are as follows.

Cache block size: 8 x 32-bit words = 32 bytes
Ring links: 64-bit parallel paths
Number of pipeline stages per cluster: 3
Number of pipeline stages occupied by a probe message: 1
Number of pipeline stages occupied by a block message: 5

We also define the following variables with their default values between parenthesis.

P : number of clusters in the system (16)
f : pipeline clock frequency (200 MHz)
tc : pipeline cycle time (5 nsec. = 1/f)
td : time to consult a cache directory (25 nsec.)
tm : time to fetch a whole block from memory or a cache (150 nsec.)
h : position of the home cluster in relation to the requesting cluster (1 to P-1)
d : position of the dirty cluster in relation to the requesting cluster (1 to P-1)

In analyzing the latencies we differentiate miss latencies from invalidation latencies. The latency to satisfy write misses, which can be seen as both a miss and an invalidation, can be approximated by whatever action takes longer, e.g., fetch the missing block or invalidate current cached copies. We will further assume that directory

and SCI protocols make use of message formats similar to our probes and block messages.

Miss Latency

In our snooping protocol, only one ring traversal is required to satisfy a miss. A probe is inserted in the ring by the requesting (missing) cluster, which takes 1 pipeline cycle. The probe reaches the valid cluster (either the home or the dirty cluster) after *3*K* cycles, where *K* is either *h* or *d*. The valid cluster takes time *td* to determine whether it has the valid copy, time *tm* to fetch the block, and 5 cycles to insert it in the ring. The missing block arrives at the requesting cluster *3*(P-K)* cycles later. Note that even though all caches in the system are snooped, the time to snoop at each cluster (*td*) does not add up to the miss latency because a probe message does not wait for the result of the snoop, but is immediately forwarded to the next cluster. In other words, the snooping time overlaps with the probe round-trip time. Thus, the total miss latency is

$$M_s = td + tm + (3*P + 6)*tc$$

It is interesting to note that the M_s is independent of the position of the valid cluster, which makes the pipelined ring a uniform access interconnection under this protocol, similar to a shared bus. This feature is not present in the other two protocols.

In a directory scheme [Leno90] the miss latency depends on the global state of the missing block. If the block is not dirty, the latency is equal to M_s, since in this case the home cluster provides a valid copy of the block. However, if the block is dirty, the position of the dirty cluster in relation to the home cluster significantly affects the miss latency. If the dirty cluster is in the path from the home cluster to the requesting cluster, the miss will be satisfied in one ring traversal (but with two directory lookups), otherwise it will take two complete ring traversals for the missing block to reach the requesting cluster. The miss latency when the block is dirty is given by the following expression:

$$Md_d = \begin{cases} 2*td + tm + (3*P + 7)*tc & (\text{if } d >= h) \\ 2*td + tm + (6*P + 7)*tc & (\text{if } d < h) \end{cases}$$

A reasonable assumption is to consider the probabilities for both cases identical, which will be the case if remote shared memory accesses are balanced among all clusters. In this case, the miss latency when the block is dirty is

$$Md_d = 2*td + tm + (4.5*P + 7)*tc$$

The overall expected miss latency depends on the probability that a given block is dirty when it is accessed, say p_d. We can then write the expected miss latency for the centralized directory protocol as being

$$M_d = (1-p_d)*M_s + p_d*Md_d$$

It is clear that M_d is always greater than M_s.

In the SCI protocol, the home cluster satisfies a miss only if the block is not currently being cached, i.e., the sharing list is empty. In this case the miss latency is M_s. If the block is being cached, the head of the list is responsible for providing a copy of the block to the missing cluster, and then the latency of satisfying the miss depends on the relative position of the head. The expression for the latency in this case is similar to the one for M_d, but the weighting probability is now p_c, the probability that a block is cached when it is accessed. The latency of misses in the SCI

protocol is then given by

$$M_{sci} = (1-p_c)*M_s + p_c*Md_d$$

Note that p_c is always greater than p_d since the event subspace to which p_d maps is a subset of the event subspace to which p_c maps. Consequently, M_{sci} is greater than M_d.

Invalidation Latency

Invalidation messages in the proposed snooping protocol always require only one ring traversal unless a given cluster rejects the invalidation probe (by setting the ns-bit). We will not consider this possibility since such rejections would also compromise the performance of the other protocols, and it would greatly complicate this analysis. The latency of invalidations in the snooping protocol is

$$I_s = (3*P+1)*tc$$

In the directory protocol, invalidations require more than one ring traversal if the clusters to be invalidated are not in the path from the home node to the requesting node, but no more than two complete ring traversals are required in any situation. Since the expected latency in this case depends on parameters that are harder to estimate, we list the best and worst case latencies below:

$$I_d(min) = td + (3*P+2)*tc; \qquad I_d(max) = 2*td + (6*P+3)*tc$$

Invalidations in the SCI protocol may require *0* to *P-1* ring traversals. If the cluster that generates the invalidation is the only one in the sharing list, no message is generated and the write operation takes place with no further delay. However, if there are *N* clusters in the sharing list, the invalidation may involve *N-1* ring traversals if the order in which the clusters appear in the sharing list is exactly inverted in relation to their order in the ring direction. The expressions for the best and worst case of invalidation latencies in the SCI protocol are given by

$$I_{sci}(min) = td; \qquad I_{sci}(max) = td + (P-1)*(3*P + td + 1)$$

4.4 Discussion

By examining the latency expressions derived above we verify that the proposed cache coherence protocol outperforms the centralized directory protocol in terms of latency of coherence accesses. The only situation in which the SCI protocol outperforms the snooping protocol is when a cluster attempts to write to a RS block and it has the only cached copy of that block. In this case the snooping protocol sends an invalidation because the cluster does not know that it has the only cached copy and that there is no copy to invalidate. In the SCI protocol, the cluster sees it as the only one in the linked list, and allows the block to be modified without delay. However, in all remaining situations, the SCI protocol exhibits much longer latencies than the snooping protocol, which we believe would lead to worse average performance figures. It is also quite straightforward to extend our snooping protocol in order to avoid invalidation messages when a cluster has a private RS copy. This can be accomplished by including a *Read-Exclusive* cache state, as it has been done in the Illinois protocol [Papa84]. This extension also requires the inclusion of another memory state to indicate if the block is being cached by some cluster or not, but its cost/complexity ratio is likely to be favorable. We chose to omit the *Read-Exclusive* state from the previous snooping protocol description for the sake of simplicity, but its inclusion is a minor modification in the overall protocol structure.

The proposed snooping protocol is by far the least complex and requires minimum storage of state information both in the memory and in the cache directories, making it also the least expensive of the three alternatives. In the proposed protocol the latency of a coherence transaction can involve more than one ring traversal only when a

probe has to be reissued, which is expected to happen rather rarely.

The rate in which the dual-directories are consulted (i.e., load on the dual-directories) will be much higher in our protocol than in the other alternatives, since it is a snooping protocol, and every single cache in the ring will be consulted for every miss or invalidation in the system. In spite of that, the bandwidth consumed by misses and invalidations in our protocol will not be greater than in directory-based or SCI protocols, due to the fact that the Express Ring is a natural broadcast environment. By arranging probes into frames and interleaving the dual-directory we can keep the probe rejection rate very low, and the increased directory load will not substantially affect the system performance.

To give an estimate of the processing power of the Express Ring with our coherence protocol lets consider a configuration with 16 clusters and a 200 MHz pipeline clock rate. The pipelined ring will have 48 stages which accommodates 6 frames. Note that, on the average, a miss occupies a probe slot for a full ring traversal and a block slot for half a ring traversal. Based on that, we can say that two misses can be satisfied as a frame makes a full ring traversal, and the expression for the maximum system miss rate that can be supported by the Express Ring underthese assumptions is

$$\text{Maximum Miss Rate} = 6/(48*tc) = 25 \text{ M misses/sec}$$

For cache hit rates of 98%, and an average memory access per instruction rate of 1.2, such an Express Ring is able to support a peak performance of

$$25x10^6/(1.2*0.02) = 1.04 \text{ GIPS}$$

5. Final Remarks

The objective of this study was to demonstrate that cache coherence can be efficiently maintained in a pipelined ring architecture, by means of a relatively simple cache coherence protocol which is based on the classic bus snoopy protocol. We believe that the Express Ring protocol demonstrates that point, and is an attractive alternative to maintain coherence in a shared-memory ring-connected multiprocessor. The proposed protocol not only outperforms directory-based protocols and the SCI protocol, but is also less complex and requires less storage than both alternatives, in an architecture such as the Express Ring.

Another important characteristic of our protocol is that, regardless of the relative positions of clusters in the ring, all ring transactions have the same latency, making the pipelined ring an uniform access media. As a result of that, at the programming level the Express Ring behaves as a very fast bus-connected multiprocessor, which makes it appropriate to general purpose computing.

The main limitation of the Express Ring architecture is that latency of accesses increases linearly with the number of clusters in the ring. Because of that, configurations with hundreds of clusters would experience very large delays to satisfy coherence requests. To be able to deal with that, we plan to experiment with latency tolerant techniques such as lockup-free caches [Dubo90a] and delayed consistency [Dubo90b]. Such techniques allow computation to be overlapped with communication, thus reducing the penalty for ring transactions. We are also interested in investigating how our protocol can be extended to allow configurations that include multiple Express Rings connected together in various manners. The Aquarius Multi-Multi protocol [Carl90] may be an interesting scheme to adopt.

6. Acknowledgments

The authors wish to thank Dr. Kai Hwang for his support and Dr. Alvin Despain for valuable discussions about implementation issues. Mr. Pong Fong also helped us with some insight in the behavior of the SCI

protocol.

7. References

[Agar88] Agarwal A. et al., "An Evaluation of Directory Schemes for Cache Coherence", *Proc. of the 15th International Symp. on Computer Architecture*, June 1988, pp. 280-289.

[Benn90] Bennett, J. et al., " Distributed Shared Memory Based on Type-Specific Memory Coherence", *Proc. of the 2nd ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, March 1990, pp. 168-176.

[Carl90] Carlton, M. and Despain, A., "Multiple-Bus Shared Memory System", *IEEE Computer*, Vol. 23, No. 6, June 1990, pp. 80-83.

[Chai90] Chaiken, D., et al.. "Directory-Based Cache Coherence in Large-Scale Multiprocessors", *IEEE Computer*, Vol. 23, No. 6, June 1990, pp. 49-59.

[DelC86] Del Corso, D., Kirrman, M., and Nicoud, J., "Microcomputer Buses and Links", Academic Press, 1986.

[Dubo90a] Dubois, M. and Scheurich, C., "Lockup-Free Caches in High-Performance Multiprocessors", to appear in *The Journal of Parallel and Distributed Computing*, 1991.

[Dubo90b] Dubois, M., "Delayed Consistency", *USC Technical Report*, CENG 90-21, July 1990.

[Ferr87] Ferrante, M., "CYBERPLUS and MAP V Interprocessor Communications for Parallel and Array Processor Systems", *Multiprocessors and*
Array Processors, W. J. Karplus editor, The Society for Computer Simulations, 1987, pp. 45-54.

[Good83] Goodman, J. R., "Using Cache Memory to Reduce Processor/Memory Traffic", Proc. 10th Int. Symp. on Computer Architecture, June 1983, pp. 124-131.

[Hals86] Halstead Jr., R., et al., "Concert: Design of a Multiprocessor Development System", *Proc. of the 13th International Symp. on Computer Architecture*, June 1986, pp. 40-48.

[Farb72] Farber, D. and Larson, K., "The System Architecture of the Distributed Computer System - the Communication System", *Symp. on Computer Networks*, Polytechnic Institute of Brooklyn, April 1972.

[Jame90] James, D., "SCI (Scalable Coherent Interface) Cache Coherence", *Cache and Interconnect Architectures In Multiprocessors*, M. Dubois and S. Thakkar editors, Kluwer Academic Publishers, Massachusetts, 1990, pp. 189-208.

[Katz85] Katz, R. et al., "Implementing a Cache Consistency Protocol", *Proc. of the 12th International Symposium on Computer Architecture*, June 1985, pp. 276-283.

[Leno90] Lenoski et al., "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor", *Proc. of the 17th International Symposium on Computer Architecture*, June 1990, pp. 148-160.

[Papa84] Papamarcos, M., and Patel, J., "A Low Overhead Coherence Solution for Multiprocessors with Private Cache Memories", *Proc. of the 11th International Symp. on Computer Architecture*, New York, 1986, pp. 414-423.

[Pier72] Pierce, J., "How Far Can Data Loops Go?", *IEEE Trans. on Communications*, Vol COM-20, June 1972, pp.

527-530.

[SCI90] "SCI (Scalable Coherent Interface): An Overview", IEEE P1596: Part I, doc171-i, Draft 0.59, February 1990.

[Sten90] Stenstrom, P., "A Survey of Cache Coherence Schemes for Multiprocessors", *IEEE Computer*, Vol. 23, No. 6, June 1990, pp. 12-25.

Appendix I - Ring Interface Functional Description

We now specify the actions of a ring interface when it sees each of the protocol messages. A cluster that has a block in INV state (and is not the home node) neither affects nor is affected by a coherence message to a that block, and so we exclude it from the discussion below. Note that requesting, home and dirty clusters are related to a specific memory block.

Read-Block message

Requesting cluster: removes the message from the ring marking its slot as empty and waits for the <ack> field in the next frame. If the a-bit is set it waits until a *Send-Block* or *Send-Update* message with that block arrives. Otherwise it issues another *Read-Block* probe.

Home cluster: if the memory state is *unmodified*, it sets the a-bit in the corresponding <ack> field and issues a *Send-Block* message addressed to the requesting cluster. Otherwise it ignores the probe.

Dirty cluster[1]: it immediately changes its state to RS and issues a *Send-Block-Update* message addressed to the requesting cluster.

Cluster with RS, RP or WP entry: does nothing.

Read-Exclusive message

Requesting cluster: removes the message from the ring marking its slot as empty and waits for the <ack> field in the next frame. If the a-bit is set it waits until a *Send-Block* or *Send-Update* message with that block arrives. Otherwise it issues another *Read-Exclusive* probe. If the ns-bit is also set it issues an *Invalidate* message for that same block. The a-bit corresponding to the new *Invalidate* message is set by the requesting cluster, since it already has the exclusive copy.

Home cluster: if the memory state is *unmodified*, it sets the a-bit in the corresponding probe's <ack> field, changes the memory state to *modified*, and issues a *Send-Block* message addressed to the requesting cluster. Otherwise it ignores the probe.

Dirty cluster: it immediately changes its state to INV, sets the corresponding a-bit, and issues a *Send-Block* message addressed to the requesting node.

Cluster with RS or RP entry: it immediately changes its state to INV. If the state was RP, another *Read-Block* probe has to be issued.

Cluster with WP entry: ignores the probe.

Invalidate message

Requesting Cluster: removes the message from the ring marking its slot as empty and waits for the <ack> field in the next frame. If the a-bit is set and the ns-bit is reset it changes its state to WE. If the a-bit and the ns-bit are set it sends another *Invalidate* probe but with the corresponding a-bit already set, since it knows it won the arbitration. If the a-bit is reset it assumes it lost the arbitration and it issues a *Read-Exclusive* probe.

Home cluster: if the memory state is *unmodified*, it sets the a-bit in the corresponding probe's <ack> field, and changes the memory state to *modified*. Otherwise it ignores the probe.

Dirty cluster: does not exist.

Cluster with RS or RP entry: it immediately changes its state to INV. If the state was RP, another *Read-Block* probe has to be issued.

Cluster with WP entry: ignores the probe.

Send-Block message

Requesting cluster: if the block was in RP state, the block is loaded in the cache and the state changes

1. a cluster with a WE copy of the block

to RS unless an invalidation for that block has been received. In this case the block is discarded. If the block was in WP state it is loaded in the cache but is only allowed to change its state to WE if all caches have been successfully invalidated. In all cases the message is removed from the ring.

Home node: if the message is actually addressed to the home node that means it is a write-back. The message is removed from the ring, the main memory copy is updated and the memory block state changes to *unmodified.*

All other clusters: ignore the block message.

Send-Block-Update message

Requesting cluster: in this case the block is known to be in RP state, and it will be loaded in the cache if no invalidation for that block has been received. However, in all cases the message is not removed from the ring, but changed to a *Send-Block* message directed to the home node.

All other clusters: ignore the block message.

MAXIMAL DIAGNOSIS FOR WIRING NETWORKS

BY

Jung-Cheun Lien and Melvin A. Breuer

CEng Technical Report 91-02

February 7, 1991

Electrical Engineering Systems

University of Southern California

Los Angeles, CA. 90089-0781

Maximal Diagnosis for Wiring Networks*

Jung-Cheun Lien and Melvin A. Breuer
Department of EE-Systems
University of Southern California
Los Angeles, CA 90089-0781

Keywords: Interconnect Test, Boundary Scan, Board Test, Diagnosis.

Abstract

Previous work on the diagnosis of faults in a wiring network has been based on the assumption that both open and short faults do not exist on the same net. When this assumption is released, these results fail to identify all diagnosable faults. The non-diagnosability of these faults, including shorts between nets, represent a serious deficiency. In this report we analyze and explain the causes for these deficiencies. In addition, we provide new theoretical results and test algorithms, and show how these deficiencies can be overcome. A test set, which is generated based on these results, can identify all diagnosable faults in a wiring network with arbitrary open and short faults. Two adaptive diagnosis algorithms are also presented which can reduce the number of test vectors while retaining the same level of diagnostic resolution.

1 Introduction

Detecting and locating faults in wiring networks on a printed circuit board has drawn much attention since the emerging of the boundary scan architecture [1]. In this architecture each primary input/output pin of a chip is associated with a boundary scan (B-S) cell. Each chip has a boundary scan register consisting of all the B-S cells. During the test mode a scan chain is formed by cascading boundary scan registers of several chips. Through this chain a test controller can access the I/O pins of every chip. Thus a virtual bed-of-nails capability is achieved. With this capability the wiring nets can be isolated from the chips and tested without the need to physically probe the board. In this way it is possible to test the new generation of boards which allow for limited probing due to the use of surface mounted devices and tape automated bonding technology. Note that if non-boundary scan devices are used, physical probes may still be required.

Many papers have dealt with the problem of finding test sets for detecting and locating faults in a wiring network [2, 3, 4, 5, 6, 7, 8, 9, 10]. Usually opens are modeled as stuck-at faults and are diagnosed separately from the shorts. This is based on the assumption that a net cannot be associated with both an open and short faults. Very comprehensive results are presented by Jarwala and Yau [7], where a framework for the detection and diagnosis of wiring networks is discussed. In particular, the *diagonally independent* property is identified. It is shown that a test set with this property is sufficient for the diagnosis of all shorts in one-step, where the results are analyzed after all test vectors have been applied.

However, as shown in this report, when the assumption concerning open and short faults is released, a test set having the diagonally independent property is insufficient for achieving complete diagnosis. In fact, there are certain faults that cannot be diagnosed without repair. Some of these non-diagnosable faults are listed in Section 2.3. Furthermore, it is shown that none of the previous results can identify all diagnosable faults. The causes for this deficiency are identified and characterized in Lemmas 1 and 2. A diagnostic level DR5, which refers to the case where all diagnosable faults can be identified, has been formulated. A term called maximal diagnosis is defined using two conditions. We show that these two conditions are both necessary and sufficient for achieving the diagnostic level DR5.

Both one-step and two-step diagnosis are ad-

*This work was supported by the Defense Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under Contract No. JFBI90092. The views and conclusions considered in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U. S. Government.

dressed in this report. In the former case, responses are analyzed only after all test vectors have been applied. In the latter case, responses are analyzed after a fixed part of the test vectors have been applied. Based on this analysis, additional test vectors are then generated and applied. Final analysis is then carried out.

For one-step diagnosis, a property called *set-cover independent* is identified. Based on this property a fundamental theorem on diagnosing wiring faults is presented. The theorem gives both necessary and sufficient conditions for identifying all diagnosable faults. A universal test set is then presented. This test set can achieve maximal diagnosis for an arbitrary network without assuming a specific fault model.

For the two-step diagnosis, two adaptive diagnosis algorithms are presented. Compared with one-step diagnosis, these algorithms can reduce the number of test vectors while retaining the same level of diagnostic resolution by using a two-step scheme. In the first step, a detection sequence is applied and the responses are evaluated. Based on the initial results, a second sequence is applied to achieve the required diagnosis. The test vector size is reduced since some information about the network has been employed in the generation of the second sequence.

This report is organized as follows. In Section 2 the preliminaries for the diagnosis of wiring networks are introduced. Addressed are the fault model, notation and definitions, non-diagnosable faults, diagnostic resolution, some previous results and the deficiencies of these results. In Section 3 theorems dealing with diagnosizing wiring faults are presented, followed by a universal test set. In section 4 two adaptive algorithms are presented. Conclusions are presented in Section 5.

2 Preliminaries

A wiring network consists of many nets. A net contains one or more drivers and one or more receivers. The logic value of a net can be controlled via one of its drivers and observed by all of its receivers. For a multi-driver net, only one driver can be enabled at a time. The others must be disabled. In addition, while testing a wiring network, only the drivers and receivers of nets are accessible. A fault-free net can transfer the logic value from an enabled driver to its receivers correctly. A receiver of a fault-free net can only receive from its associated drivers. The objective of diagnosis of a wiring network is to find a set of test vectors which can be applied to identify as many faults in the network as possible. No structural information about the network is assumed in this work.

2.1 Fault Model

Two types of physical faults, namely open and short, are assumed. More than one physical break are possible in an opened net. Ignoring fan-out nodes and shorts, multiple opens are modeled as a single open along a wire segment. Also more than one physical bridge are possible between two shorted nets. Multiple shorts are modeled as a single short between two wire segments.

If two or more nets are shorted, the resulting behavior can be modeled either (1) a wired-OR, (2) a wired-AND, or (3) a strong-driver, where one driver dominates the resulting behavior. In all cases, all nets involved in a short will have the same resulting logic value. The wired-OR fault model is assumed in this discussion unless mentioned otherwise.

A net shorted to a power line VCC (GND) will exhibit a stuck-at 1 (stuck-at 0) behavior. If a net contains an open, the logic value interpreted by all floating receivers of the opened net will be the same, which could be either a soft stuck-at 1 or a soft stuck-at 0. A logic value of a net shorted to a wire segment at a soft logic value cannot be forced to this soft value. The soft stuck-at 1 model is assumed for a floating net unless mentioned otherwise. In Figure 1, the logic value of the point A is soft stuck-at 1.

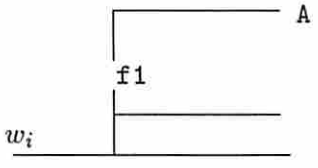


Figure 1: A soft stuck-at 1 case.

Both opens and shorts can occur on the same net. If a net contains both opens and shorts, the logic value received by the receivers is determined by the combined effect of both opens and shorts.

A short to an open net is illustrated in Figure 2, where A takes the logic value of B.

2.2 Notation and definitions

The notation and definitions used in this report follow the conventions established in [7]. For convenience, some of this information is repeated here.

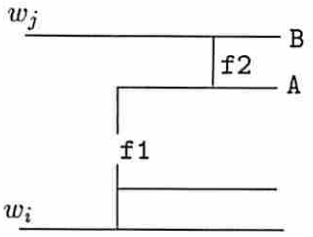


Figure 2: A short in an opened net.

- Parallel Test Vector (PTV): the vector applied to all nets of a wiring network in parallel.
- Sequential Test Vector (STV): the vector applied to a net, over a period of time, by a sequence of PTVs.
- Test Set (or test sequence) S: the collection of all STVs. Each column of S is a PTV and each row of S is a STV.
- Sequential Response Vector (SRV): the response of a net to a STV.
- Syndrome: the SRV of a faulty net.
- Aliasing syndromes: the resulting syndrome of a set of faulty nets is the same as a correct SRV of a net not in the set.
- Confounding syndromes: the identical syndromes that results from multiple independent faults.

The following definitions are also used.

- OR-Cover: A vector V_i *OR-covers* another vector V_j if for every bit position in V_j that is 1, the corresponding bit in V_i is also 1. For example, STV_i=(1101) OR-covers STV_j=(0101), or STV_j is OR-covered by STV_i. The OR-cover is used in the wired-OR fault model. In a similar fashion one can define an AND-cover for the wired-AND fault model. In this report, the term OR-cover is abbreviated as *cover*.
- Independent set: A test set S is an *independent set* if no STV_i can cover another $STV_j, j \neq i$.
- Set-cover: Let V_J be the result of wire-ORing a set of vectors $V_{j1}, \ldots, V_{jk}$. A vector V_i *set-covers* the vectors $V_{j1}, \ldots, V_{jk}$ if V_i covers V_J.
- Set-covering syndrome: A *set-covering syndrome* is a syndrome that results from a set of shorted nets (W') that either covers a SRV or is covered by a SRV of some net w_i not in W'.
- Set-cover independent: Let S=(STV_1, ..., STV_n) be a test set for a set of nets $W = (w_1, \ldots, w_n)$. S is *set-cover independent* if for $i = 1, \ldots, n$, STV_i is not covered by or covers the union (for wired-OR, intersection for wired-AND) of any subset of vectors in $S - STV_i$. In the other words, for every SRV_i in S no set-covering syndrome can exist.

2.3 Non-diagnosable Faults

A fault f is said to be *non-diagnosable* if there does not exist a test set S and an algorithm A such that by applying S to the network and processing the responses using algorithm A, f can be identified. Note that all single faults are diagnosable. Based on the described model, there are faults that are non-diagnosable. Some of these non-diagnosable faults are listed below.

- In a set of nets that are shorted with each other, there are some opens that are non-diagnosable. For example, in Figure 3 the open fault f1 on net w_2 is non-diagnosable. Since w_1 and w_2 are electrically common, it is impossible to find a test to detect the open.

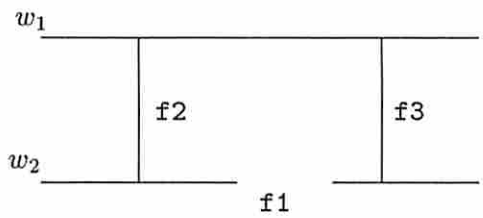


Figure 3: An open that is non-diagnosable.

- The short between a set of opened nets is non-diagnosable. For example, in Figure 4, it is impossible to identify the short f1 between w_1 and w_2 since no receivers are connected to the shorted wires.

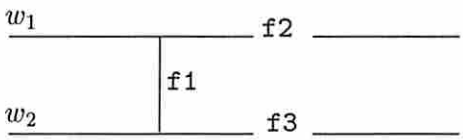


Figure 4: A short that is non-diagnosable.

- There exists three possible reasons for a faulty net that has all-1 responses to a test (1) the net is shorted with a VCC power line, (2) the net is

opened and floating (soft stuck-at 1 model), and (3) the net is shorted with other nets such that the combined result is an all-1 vector (wired-OR model). The third case can be distinguished from the others by applying an all-0 PTV. The first two cases cannot be distinguished.

2.4 Diagnostic Resolution

Various levels of diagnostic resolution are possible in testing a wiring network. Listed below are six such levels. They are listed in ascending order of their diagnostic resolution, i.e., DR1 has the lowest diagnostic resolution, and DR6 has the highest diagnostic resolution.

DR1: Determine whether the entire network is fault-free.

DR2: Identify all faulty nets.

DR3: For each and every net, determine whether it is fault-free without knowing the response of the other nets.

DR4: Identify all faulty nets. In addition, for nets without shorts, identify the existence of nets having opens. For a faulty net without open faults, identify all nets that are shorted to it.

DR5: Identify all faulty nets. In addition, identify all faults that are diagnosable.

DR6: Identify all faulty nets. In addition, identify all the opens and shorts in the network.

In DR1, one is only interested in determining the health status of the entire network. No further diagnostic information is provided. In DR2, all faulty nets are identified. No information about what type of faults associated with each net is provided. In DR3, all faulty nets are identified. In determining the health status of net, only its response is required. No information about the response of other nets are needed. This scheme is most suitable for a built-in self-test type of design. In DR4, all faulty nets are identified. The faults associated with each nets can be identified if they belong to those cases described above. In DR5, all faulty nets are identified. More faults can be identified than in the case of DR4. In DR6, all faulty nets are identified. In addition, all the faults, including opens and shorts, are identified.

For the purpose of repairing a wiring network, it is desirable that as many faults as possible be identified. Due to the fact that some faults cannot not be identified without first repairing other faults, DR6 cannot be reached. An example would be a net with multiple opens. In this work, we focus on achieving DR5.

2.5 Previous Results

Previous results focus on diagnostic resolution ranging from DR1 to DR4. Some typical results for the testing of a network consisting of 4 nets are listed below. These results are based on the assumption that both opens and shorts cannot exist on the same net.

Counting Sequences: Kautz[2]

$$S = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

This test set consists of a simple counting sequence, where all-0 and all-1 STVs are not used. This test can achieve the DR1 diagnostic levels. The size of the test set is $\lceil log(n+2) \rceil$, where n is the number of nets.

Complementary Counting Sequence: Wagner[4]

$$S = \left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

This test set consists of a counting sequence and its complement. All-0 and all-1 STVs can be used. The size of the test set is $2\lceil logn \rceil$. This test set can achieve diagnostic level DR3. By eliminating the aliasing syndromes, the *self-diagnosis* property is achieved, which allows the determination of the health status of a net by examining only its SRV.

Maximal Independent Set: Cheng et al. [9]

$$S = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Constant weight codes, which means that every STV has the same number of 1s, is a class of independent test sets. These test sets can achieve diagnostic level DR3. The size of the test set is minimal for self-diagnosis when the number of 1s in a SRV is half of the number of PTVs, which is referred to as a maximal independent set [9].

Diagonally Independent Sequence: Jarwala and Yau[7]

$$S = \begin{bmatrix} x & x & x & 1 \\ x & x & 1 & 0 \\ x & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The x represents either a 0 or a 1. This test set can achieve the diagnostic level DR4. Both aliasing and confounding syndromes can be eliminated. All pairs of nets that are shorted are identified.

2.6 Deficiencies in Previous Results

Recall that these results are based on the assumption that both opens and shorts cannot exist on the same net. However, when this assumption is released, there exists certain types of opens and shorts that cannot be identified. For example, the diagonally independent sequence

$$S = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

cannot identify the short fault f1 in Figure 5 nor the open fault f2 in Figure 6.

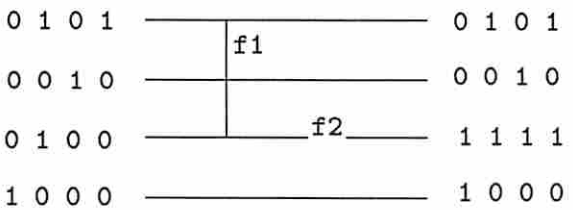


Figure 5: A short that cannot be identified by a diagonally independent sequence.

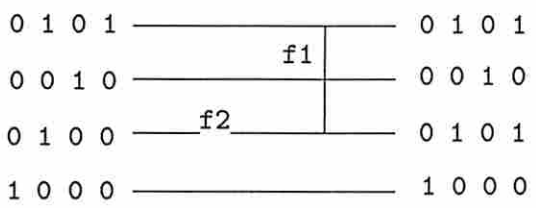


Figure 6: An open that cannot be identified by a diagonally independent sequence.

The following lemmas summarize these cases which cannot be completely handled by previous results.

Lemma 1 *A test set S cannot identify the short between two nets w_i and w_j if (a) there is an open which is closer to the receiver of net w_i than the short, and (b) STV_i is covered by STV_j.*

Proof:
SRV_i is the all-1 vector since no logic value is transferred to the receiver. $SRV_j = STV_j$ since STV_j covers STV_i. Therefore, it is impossible to know whether there is a short between w_i and w_j. □

Figure 5 is an example of Lemma 1.

Lemma 2 *A test set S cannot identify the open in a net w_i if there exists another net w_j such that (a) there is a short between w_i and w_j which is closer to the receiver of w_i than the open, and (b) STV_j covers STV_i.*

Proof:
Since $SRV_j = SRV_i = STV_j$, one cannot be certain whether there is a "contribution" from STV_i to SRV_i. Therefore, the open cannot be identified. □

Figure 6 is an example of Lemma 2.

Both Lemma 1 and 2 can be generalized to multiple nets. For example, in Figure 7 the short fault f1 cannot be identified by a maximal independent set. This is because STV_3 is set-covered by $SRV_1 = SRV_2 = 1110$. Therefore it is impossible to determine whether the short f1 exists or not.

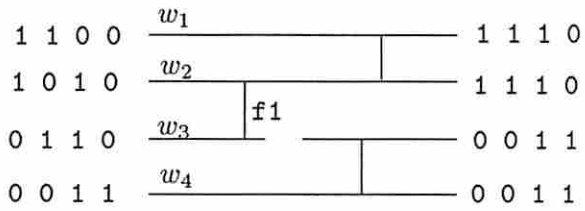


Figure 7: A short that cannot be identified by an independent test set.

In summary, none of the previous results can identify all faults described above, these includes the diagonally independent sequence [7], the maximal independent set[9], and the complementary counting sequence[4].

The existence of non-identified shorts and opens in a network represents a major deficiency in diagnosis. We next describe a test set that can diagnose these shorts and opens.

3 One-Step Diagnosis

Due to the existence of non-diagnosable faults in a wiring network, it is impossible to identify all faults without repair or access to points of the nets other than the drivers and receivers. The term **maximal diagnosis** is defined as follows.

Let $W = (w_1, w_2, \ldots, w_n)$ be a set of nets to be tested, D_i be the set of drivers of net w_i, and R_i be the set of receivers of net w_i. A test set S can achieve *maximal diagnosis* for W when the following two conditions are verified.

- Condition C1: For $i = 1, \ldots, n$, by analyzing the responses obtained from the application of S, the existence of the connection (D_i, R_i) can be determined.

  The connection (D_i, R_i) exists if for all k, each driver $d_{ik} \in D_i$ can *transfer* its logic value to all receivers in R_i correctly. A driver in D_i is said to *transfer* its logic value to a receiver R_i if SRV_i covers STV_i. Note that only one driver can be enabled for a given net at one time. In other words, if the connection (D_i, R_i) exists, then the application of STV_i to the enabled driver of w_i will make true one of the following: (1) $SRV_i = STV_i$, or (2) SRV_i covers STV_i.

- Condition C2: For all $i, j, i \neq j$, by analyzing the responses obtained from the application of S, the existence of the connection (D_i, R_j) can be determined.

  The connection (D_i, R_j) does not exist if for all k, each driver $d_{ik} \in D_i$ does not transfer its logic value to any receivers in R_j. In other words if the connection (D_i, R_j) does not exist, the application of STV_i to the enabled driver of w_i will make true one of the following: (1) $SRV_j \neq STV_i$, or (2) SRV_j does not cover STV_i.

Theorem 1 *Diagnostic level DR5 can be achieved by a test set S iff S can achieve maximal diagnosis.*

Proof:
(a) Sufficiency:
The test set S can verify both Condition C1 and C2 since S can achieve maximal diagnosis for a network W. By definition DR5 is achieved if all diagnosable faults in W can be identified. There are two type of faults in W, namely opens and shorts. We discuss them separately in the following.

(1) opens: An open on a net w_i can be diagnosed if the connection (D_i, R_i) does not exist. Since S can achieve maximal diagnosis, this connection is determined in Condition C1. Thus S can identify all diagnosable opens in W.

(2) shorts: A short between two nets w_i and w_j can be diagnosed if one of the following is true: (a) there exists a driver D_k such that both connections (D_k, R_i) and (D_k, R_j) exist; (b) there exists a receiver R_k such that both connections (D_i, R_k) and (D_j, R_k) exist. The existence of these connections can be determined by S using Condition C2. Thus S can identify all diagnosable shorts in W.

From (1) and (2), all diagnosable faults can be identified. Therefore the diagnostic level DR5 can be achieved. Thus we prove the sufficiency aspect of maximal diagnosis.

(b) Necessity:
Assume that S cannot achieve maximal diagnosis. By definition, there exist at least one connection the existence of which cannot be determined. Two cases are possible: (1) if the connection is of the form (D_i, R_i) then there can be an open fault on net w_i that cannot be identified; (2) if the connection is of the form $(D_i, R_j), i \neq j$, then there can be a short between w_i and w_j that cannot be identified. In both cases, the faults can be identified by a walking ones sequence. Thus, by definition, these faults are diagnosable. Therefore the diagnostic level DR5 cannot be achieved. Thus the maximal diagnosis property is necessary. □

In this report the generation of test sets that can achieve maximal diagnosis are discussed. The generated test sets thus can achieved diagnostic level DR5 in which all diagnosable faults are identified. This is the best diagnostics one can achieve without accessing points on the net other than the drivers and receivers.

Lemma 3 *For the wired-OR (wired-AND) model, any test set S is set-cover independent iff S has the walking ones (zeros) sequence as its subsequence.*

Proof:
(a) Sufficiency:
This is obvious since the walking ones sequence is set-cover independent.

(b) Necessity:
Suppose that the test set S is set-cover independent. For a given STV_i, there must exist a PTV such that its ith bit is 1 and all other bits are 0. This is true for all $STV_i, i = 1, \ldots, n$. By arranging S properly

(by swapping rows and columns), a walking ones sequence can be constructed. Therefore, S contains a subsequence which is a walking ones sequence. □

We next present a theorem from which a test set can be derived for achieving maximal diagnosis.

Theorem 2 *A test set S can achieve maximal diagnosis for a network W in one-step iff S is set-cover independent.*

Proof:
(a) Sufficiency:
Suppose that S is set-cover independent. No set-covering syndrome can exist. For each and every net w_i Condition C1 can be verified by checking if SRV_i covers STV_i. If this is not true then there is at least an open fault between the driver and the receivers of the net w_i. Since no set-covering syndrome can exist, no drivers of other nets can cover the STV_i.

Condition C2 can be verified as follows. If $STVi$ cannot cover SRV_i, then for every bit in SRV_i that is not covered by the STV_i, there is a short between the receiver of w_i and a driver w_j whose STV_j covers that bit.

Since both Conditions C1 and C2 can be verified, maximal diagnosis is achieved and the sufficiency aspects of the theorem have been demonstrated.

(b) Necessity:
Suppose that S is not set-cover independent. There exists at least one STV_i that is covered by another STV_j. When the following two faults occur the existence of the connection (D_i, R_i) cannot be determined: (1) a short between w_i and w_j, and (2) an open on w_i that is closer to the driver than the short. This means that Condition C1 cannot be satisfied. Thus, by definition, maximal diagnosis is not achieved. □

From Lemma 3 and Theorem 2, we can conclude that any test set that can achieve maximal diagnosis must have a walking ones (zeros) sequence as its subsequence. From Theorem 2 and 1 we can conclude that a set-cover independent test set can achieve the diagnostic level DR5.

Example 1: The short that cannot be identified by a diagonally independent sequence in Figure 5 can be identified by a set-cover independent sequence (see Figure 8).

Universal Test Set:
Assuming the wired-OR model and that a floating net is modeled as a soft stuck-at 1, a test set that can achieve maximal diagnosis for a network consisting of 3 nets can be constructed as follows.

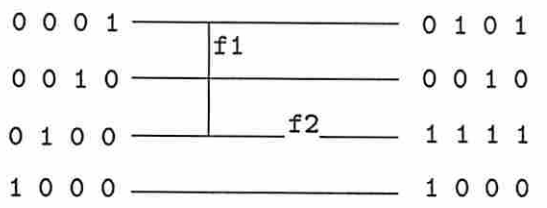


Figure 8: Achieving maximal diagnosis using a set-cover independent sequence.

$$\left[\begin{array}{c|ccc} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

The all-0 PTV is used to distinguish between the cases (1) all nets are shorted together (all 1s for SRVs) and (2) all nets are opened.

Similarly, if the wired-AND model is used and an open and floating net is modeled as a soft stuck-at 0, a test set for maximal diagnosis can be constructed by a walking zeros sequence followed by an all-1 PTV.

In summary, a universal test set for maximal diagnosis, without making any assumption on the nature of the faults, is as follows.

$$S_{universal} = \left[\begin{array}{c|ccc|c|ccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}\right]$$

4 Two-Step Diagnosis

Two-step diagnosis refers to the fact that diagnosis is done by applying two test sequences. The results of the first test sequence is used in generating the second test sequence. This type of diagnosis is also known as adaptive diagnosis.

Two adaptive algorithms that can achieve maximal diagnosis with a reduced number of PTVs are presented next. The test sets for both algorithms do not have the set-cover independent property since certain information about the network is employed in generating the second test sequence.

4.1 Adaptive Algorithm A1

1. Apply a maximal independent set (S_M). Collect and analyze the responses. Stop if no faults are detected.

2. Partition the nets into two groups. The partitioning is done as follows. For a net $w_i, i = 1, ..., n$, if (a) $STV_i = SRV_i$ and (b) SRV_i is unique, include w_i into group 0, else group 1.

3. Apply a walking ones sequence S_F to all nets in group 1, and all-0 vectors to all nets in group 0.

The objective of the first sequence is to achieve the self-diagnosis property by eliminating aliasing syndromes. The maximal independent set is the minimal size test set that can achieve this objective [9]. The number of PTVs required by the first sequence is p, where p is the smallest integer satisfying $C^p_{\lfloor p/2 \rfloor} \geq n$, and $C^p_{\lfloor p/2 \rfloor}$ represents all possible combinations of choosing $\lfloor p/2 \rfloor$ items out of p items. The total number of PTVs required by this algorithm is $p + F$, where F is the number of nets in group 1.

We next show by an example that algorithm A1 can achieve maximal diagnosis. Let W be a network of 4 nets, where w_1 and w_2 are two nets in group 0 and w_3, w_4 are in group 1. If both Conditions C1 and C2 can be verified then maximal diagnosis is achieved.

Let S_{A1} be the test generated by Algorithm A1,

$$S_{A1} = (S_F, S_M) = \left[\begin{array}{cc|cccc} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

Table 1 shows when a connection between a driver and a receiver is checked. Note that the existence of all connections must be able to be determined in order to achieve maximal diagnosis. Checking the existence of connections of the form (D_i, R_i) is necessary to verify Condition C1; checking the connections $(D_i, R_j), i \neq j$, is necessary to verify Condition C2.

Seq.	Connections checked
S_M	(D_1, R_1), (D_1, R_2), (D_1, R_3), (D_1, R_4) (D_2, R_1), (D_2, R_2), (D_2, R_3), (D_2, R_4)
S_F	(D_3, R_1), (D_3, R_2), (D_3, R_3), (D_3, R_4) (D_4, R_1), (D_4, R_2), (D_4, R_3), (D_4, R_4)

Table 1: Checking connections using Algorithm A1.

The connection (D_1, R_1) does exist since (a) in the sequence (S_M), for every i it is impossible to find a subset from $S - STV_i$, whose wired-OR result equals STV_i, (b) $STV_1 = SRV_1$ and (c) SRV_1 is unique. Similarly, the connection (D_2, R_2) exists. The connections (D_1, R_2) and (D_2, R_1) do not exits since the existence of any one of them will make $SRV_1 = SRV_2$, which contradicts the fact that $SRV_1 \neq SRV_2$.

The existence of the connection (D_1, R_3) is also verified by (S_M) for the follow reason. If this connection exists, SRV_1 must equal SRV_3 since there is no open on w_1. However SRV_1 is unique, so we know that this connection doesn't exist. In a similar manner the other connections are checked by (S_M).

The connections (D_3, R_i) and $(D_4, R_i), i = 1, .., 4$ are verified by the sequence S_F. Maximal diagnosis can be achieved in this example since the existence of all connections can be determined. The efficiency of this algorithm depends on the value of F. In the case when $F \approx n$, the number of PTVs is close to that of a walking ones sequence.

4.2 Adaptive Algorithm A2

1. Apply a maximal independent set (S_M). Collect and analyze the responses. Stop if no faults are detected.

2. Partition nets into group 0 and 1 (as Algorithm A1).

3. Partition nets in group 1 such that all nets with the same SRV are in the same group. Number these new groups from 1 to G. Let K be the cardinality of the largest group.

4. Apply a walking ones sequence S_G to all groups in parallel except to group 0 for which the all-0 vectors are applied.

5. Apply another walking ones sequence S_K across groups. That is, all nets in the same group are modeled as a single net. Again the all-0 vectors are applied to all nets in group 0. The number of PTVs is G.

The total number of PTVs for Algorithm A2 is $p + G + K$. In general this algorithm requires fewer PTVs than Algorithm A1. This is because in Algorithm A1 the F nets in group 1 is now partitioned into G groups in Algorithm A2. It is obvious that $F \geq G + K$.

We next show that Algorithm A2 can achieve maximal diagnosis by checking Conditions C1 and C2.

Let W be a network with seven nets. After the application of S_M, three groups are formed. Let w_1,

w_2 be in group 0, w_3, w_4 be in group 1, and w_5, w_6, w_7 be in group 2. In this example $n = 7$, $G = 2$ and $K = 3$. According to Algorithm A2, the total test set S_{A2} consists of a maximal independent set of 5 PTVs (S_M), followed by 3 walking ones PTVs (S_G), which again are followed by another 2 walking ones PTVs (S_K).

$$S_{A2} = \left[\begin{array}{cc|ccc|ccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

Seq.	Connections checked
S_M	$(D_1, R_1), (D_1, R_2), (D_1, R_3), (D_1, R_4), (D_1, R_5), (D_1, R_6), (D_1, R_7), (D_2, R_1), (D_2, R_2), (D_2, R_3), (D_2, R_4), (D_2, R_5), (D_2, R_6), (D_2, R_7)$.
S_G	$(D_3 \mid D_5, R_1), (D_3 \mid D_5, R_2), (D_3 \mid D_5, R_3), (D_3 \mid D_5, R_4), (D_3 \mid D_5, R_5), (D_3 \mid D_5, R_6), (D_3 \mid D_5, R_7), (D_4 \mid D_6, R_1), (D_4 \mid D_6, R_2), (D_4 \mid D_6, R_3), (D_4 \mid D_6, R_4), (D_4 \mid D_6, R_5), (D_4 \mid D_6, R_6), (D_4 \mid D_6, R_7), (D_7, R_1), (D_7, R_2), (D_7, R_3), (D_7, R_4), (D_7, R_5), (D_7, R_6), (D_7, R_7)$.
S_K	$(D_3, R_1), (D_3, R_2), (D_3, R_3), (D_3, R_4), (D_3, R_5), (D_3, R_6), (D_3, R_7), (D_5, R_1), (D_5, R_2), (D_5, R_3), (D_5, R_4), (D_5, R_5), (D_5, R_6), (D_5, R_7) (D_4, R_1), (D_4, R_2), (D_4, R_3), (D_4, R_4), (D_4, R_5), (D_4, R_6), (D_4, R_7), (D_6, R_1), (D_6, R_2), (D_6, R_3), (D_6, R_4), (D_6, R_5), (D_6, R_6), (D_6, R_7)$.

Table 2: Checking connections using Algorithm A2.

Table 2 shows how Conditions C1 and C2 can be verified. From the discussion in Algorithm A1, it is clear that for $i = 1, \ldots, 7$ connections (D_1, R_i) and (D_2, R_i) are checked after the application of S_M.

The existence of connections $(D_7, R_i), i = 1, \ldots, 7$, are checked by S_G since it contains a PTV which applies 1 to w_7 and 0 to all other nets. The existence of connections (D_i, R_j), (D_k, R_j) or both is denoted by $(D_i | D_k, R_j)$. After the application of S_G, the existence of both $(D_3 | D_5, R_3)$ and $(D_3 | D_5, R_5)$ are checked. Suppose that one of them does not exist, then S_K can easily distinguish among the three possible connections. Suppose that both $(D_3 | D_5, R_3)$ and $(D_3 | D_5, R_5)$ exist, then one of the following 9 cases is true:

1. (D_3, R_3) and (D_3, R_5) exist;
2. (D_3, R_3) and (D_5, R_5) exits;
3. (D_3, R_3), (D_3, R_5) and (D_5, R_5) exist;
4. (D_5, R_3) and (D_3, R_5) exist;
5. (D_5, R_3) and (D_5, R_5) exits;
6. (D_5, R_3), (D_3, R_5) and (D_5, R_5) exist;
7. (D_3, R_3), (D_5, R_3), and (D_3, R_5) exist;
8. (D_3, R_3), (D_5, R_3), and (D_5, R_5) exits;
9. (D_3, R_3), (D_5, R_3), (D_3, R_5) and (D_5, R_5) exist.

The connections (D_3, R_3) and (D_3, R_5) cannot both exist for the following reason. Suppose that they both exist. Then one can conclude that $SRV_3 = SRV_5$, which contradicts the fact that w_3 and w_5 are in different groups. Similarly, the connections (D_5, R_3) and (D_5, R_5) cannot both exist. Therefore, the above 9 cases are reduced to two cases, namely cases 2 and 4. The sequence S_K can distinguish between these two cases.

Similarly, the existence of both $(D_4 | D_6, R_4)$ and $(D_4 | D_6, R_6)$ are checked by S_G. The sequence S_K then can distinguish the two cases, namely (1) the existence of connection (D_4, R_4) and (D_6, R_6), or (2) the existence of connections (D_4, R_6) and (D_6, R_4).

All other connections are similarly checked. Since both Conditions C1 and C2 can be verified, maximal diagnosis is achieved.

4.3 Deficiencies in Previous Adaptive Algorithms

Method 3: [9]
This algorithm first applies a counting sequence, based on the initial results, a second sequence is applied. The STV of the second sequence represents the number of 0s in the corresponding STVs of the first sequence. The purpose of the second sequence is to make sure that the overall test set S is independent. This algorithm can achieve self-diagnosis. No confounding syndromes can be identified. Also, the fault f1 in Figure 7 cannot be detected by this algorithm.

W-Test Algorithm: [3]
This algorithm is similar to Algorithm A1 except that the first sequence is a counting sequence S_C, i.e., in the W-Test algorithm the test set consists of only S_C and S_F. We next show an example in which the W-Test Algorithm cannot achieve maximal diagnosis.

Part of a wiring network is shown in Figure 9,

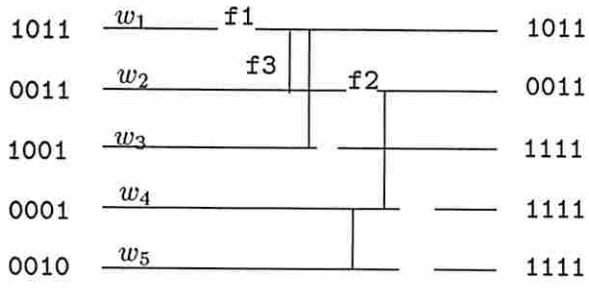


Figure 9: Deficiency in the W-Test Algorithm.

where the counting sequence S_C have been applied. For $i = 1, 2$ $STV_i = SRV_i$ and SRV_i is unique, thus both w_1 and w_2 will be put into group 0. This means that the opens f1, f2 and the short f3 will not be identified since during the application of a walking ones sequence all nets in group 0 will be kept at 0.

Therefore, to avoid putting a net into group 0 by mistake, it is necessary to apply an independent set which can achieve the *self-diagnosis* property. Both Algorithm A1 and A2 will not put w_1 and w_2 into group 0. Thus the faults associated with them can be identified by either S_F or (S_G, S_K).

C-Test Algorithm: [7]
The C-Test Algorithm first applies a counting sequence, then based on the analysis of the syndromes, one or more PTVs are applied.

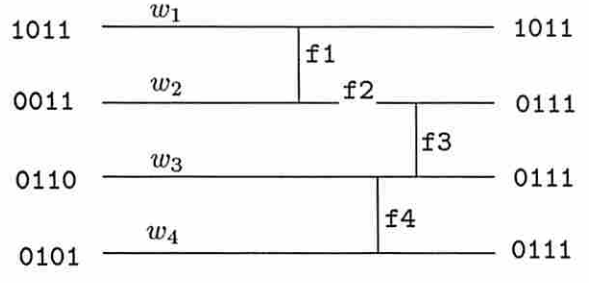


Figure 10: Deficiency in the C-Test Algorithm.

Part of a wiring network is shown in Figure 10, where a counting sequence has been applied. In the C-Test Algorithm both short faults f3 and f4 can be identified immediately. However, since no aliasing or confounding syndromes are related to $SRV_1 = 1011$, the fault f1 and f2 cannot be identified.

Using the same example, the diagnosis sequence in both Algorithm A1 and A2 will apply a walking ones sequence to w_2, w_3, w_4 since they are in the same group. Thus all faults in the network can be identified.

5 Conclusions

The results presented in this report outperform all previous results in that all diagnosable faults can be identified. We have shown that there exists diagnosable faults in a wiring network that cannot be identified by any of the previous results, which includes the complementary counting sequence [4], the independent set [9], the diagonally independent sequence [7], the W-Test Algorithm [3], the C-Test Algorithm [7] and the Method 3 in [9]. The faults that lead to the deficiencies for previous results are summarized and explained in Lemma 1 and 2.

Various levels of diagnostic resolution are defined. DR5 refers to the diagnostic level where all diagnosable faults are identified. We have also defined maximal diagnosis in terms of Conditions C1 and C2, which, as shown in Theorem 1, proved to be both necessary and sufficient for achieving diagnostic level DR5.

A property called set-cover independent is also presented. A test sequence that is set-cover independent must have a walking ones sequence as its subsequence. We have shown in Theorem 2 that a set-cover independent set is both necessary and sufficient for achieving maximal diagnosis. Based on this theorem a universal test set is presented. This test set can achieve maximal diagnosis for a wiring network without a specific fault model.

Two adaptive algorithms that can achieve maximal diagnosis have been presented. They can reduce the size of the test set by employing two-step diagnosis. Both algorithms first apply a maximal independent set to eliminate the aliasing syndromes. The responses are then analyzed and based on the initial results, the second part of the test set is generated. Without the information from the first part, it is impossible to reduce the size of the test set. Algorithm A2 requires less PTVs than Algorithm A1. However, we do not know whether Algorithm A2 generates the shortest possible test set.

References

[1] IEEE Standard 1149.1-1990, "IEEE Standard Test Access Port and Boundary Scan Architecture," *IEEE Standards Board, 345 East 47th Street, New York, NY 10017,* May 1989.

[2] W.H. Kautz, "Testing for Faults in Wiring Networks", *IEEE Trans. on Computers*, Vol. C-23, No. 4, pp. 358-363, April 1974.

[3] P. Goel and M.T. McMahon, "Electronic Chip-In-Place Test", *Proc. Int'l Test Conf.*, pp. 83-90, 1982.

[4] P.T. Wagner, "Interconnection Testing with Boundary Scan", *Proc. Int'l Test Conf.*, pp. 52-57, 1987.

[5] A. Hassan, J. Rajski, and V.K. Agarwal, "Testing and Diagnosis of Interconnects using Boundary Scan Architecture", *Proc. Int'l Test Conf.*, pp. 126-137, 1988.

[6] A. Hassan, V.K. Agarwal, J. Rajski and B.N. Dostie, "Testing of Glue Logic Interconnects Using Boundary Scan Architecture", *Proc. Int'l Test Conf.*, pp. 700-711, 1989.

[7] N. Jarwala and C.W. Yau, "A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects", *Proc. Int'l Test Conf.*, pp. 63-70, 1989.

[8] C.W. Yau and N. Jarwala, "A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects", *Proc. Int'l Test Conf.*, pp. 71-77, 1989.

[9] W.T. Cheng, J.L. Lewandowski and W. Wu, "Diagnosis for Wiring Interconnects", *Proc. Int'l Test Conf.*, pp. 565-571, 1990.

[10] G.D. Robinson and J.G. Deshayes, "Interconnect Testing of Boards with Partial Boundary Scan", *Proc. Int'l Test Conf.*, pp. 572-581, 1990.

MAXIMAL DIAGNOSIS FOR WIRING NETWORKS

Jung-Cheun Lien and Melvin A. Breuer

CEng Technical Report 91-02

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA. 90089-0781
(213)740-4469

February 7, 1991