Proposal for change to reset notification / acknowledgment procedure in P1394.1 draft 0.02

1. Introduction

In the current P1394.1 draft (0.02), reset notification and acknowledgment includes the bus_id of the bus that was reset and a generation number. I propose removing these fields and simplifying the process.

2. Current status

Whenever a 1394 bus is reset, the phy_id part of node addresses can change. The current mechanism is designed to stop any packet that may be in transit across a net of bridged 1394 buses if the bus that the packet is destined for has been reset, as well as stopping additional erroneous packets from being transmitted. Bridges contain blocking mechanisms that are activated when resets are detected, and the individual parts of the mechanisms are deactivated when the nodes responsible for them acknowledge that they have been notified of the reset and have ensured that all packets in transit have been destroyed and all addresses used for new packets have been updated.

Currently, the reset notification includes the bus_id of the reset bus, so nodes receiving the notification can destroy only those packets destined for that bus and only addresses on that bus need to be redetermined. A generation number is also included, to help with handling multiple resets. If a node tries to disable a blocking mechanism with an incorrect bus_id or outdated generation number, then the mechanism will stay activated. Eventually the last notification will arrive at the node, and it will acknowledge the notification with the most current bus_id / generation number pair and the blocking will be disabled.

3. Problems

3.1 Multiple resets and/or missed notifications

If a lot of resets occur on multiple buses within a short time, there will be a flurry of reset notifications. It is possible that some of these will be missed by some nodes. If the last notification is missed, then the node will not succeed in unblocking itself because it will not have the correct bus_id / generation number pair to use in the reset acknowledge. This can be corrected by the node noticing that it cannot send packets to remote nodes (receives address errors), and reading the RESET_NOTIFICATION register of the bridge portal, and then writing a reset acknowledge with that value.

If any reset notification other than the last one in a flurry is missed, then the node will succeed in unblocking itself. If the node only re-establishes correct addresses for remote nodes on buses that it saw reset notifications for, then it will not know that some additional stored addresses are invalid. The blocking mechanism will no longer prevent the bad packets from flowing, so it is now possible for a mis-addressed packet to reach an incorrect destination node.

One possible solution is to require that nodes never miss a reset notification packet. This can be approximated by putting a large FIFO behind the RESET_NOTIFICATION register address, but even then the FIFO can overflow or packets can be lost for other reasons such as transmission errors. Another possible solution is to implement a bit in the node that would be set if a write to the RESET_NOTIFICATION address was lost. If the node sees this bit set, then it would re-establish all remote addresses. This takes care of the overflow case, but not the transmission error case.

3.2 Topology changes

This reset notification only handles resets of individual buses. If there is a change of topology in the net, this design does not have a means to notify the nodes that bus_ids may have changed.

3.3 Clearing the quarantine bit

It is not clear how the bridge manager knows when it is safe to turn off the quarantine bit after a bus is reset.

4. Proposed solutions

4.1 Simplified reset notification

My conclusion from this is that the only safe way to deal with this is for nodes to always re-establish all remote addresses after any reset notification, regardless of the bus_id involved. The node must first issue the reset acknowledgment, then re-establish the addresses. If more resets occur after the acknowledgment while the node is correcting the addresses, then another reset notification / reset acknowledgment cycle will occur.

If this is the correct solution, then it is no longer necessary to include the bus_id and generation number in the notification / acknowledgment, and the registers can be simplified to simple flags that are set or cleared by appropriate writes.

It is still necessary to treat the case of a lost reset notification due to transmission errors. Nodes will always need to deal with address error responses by assuming that a reset notification may have occurred. Retries may be attempted before doing a reset acknowledge, and the retries will continue to fail with address errors if a reset caused the problem. The node then does the reset acknowledge and re-establishes remote addresses. I do not see any problem (other than extra traffic) caused by an excess reset acknowledge / address re-establishment.

Since address errors need to be treated as possible remote reset indications, the explicit notification becomes optional. It is possible for a node to always rely on address errors and not bother with handling reset notification. We may want to consider making reset notification only go to portals and not to non-bridge nodes.

An intermediate method might be to broadcast the notification on each local bus. Broadcast does not guarantee delivery, but speeds up the process most of the time. Two arguments against this are:

1. Broadcasts in general are not considered "good citizens" on the bus, especially if they occur in flurries. Devices that don't care about remote buses and that have small FIFOs will prefer not to receive & then discard the packets.
2. The broadcast will usually result in many nodes trying to look up new remote addresses all at the same time. If each node is not aware of the reset until they send a request transaction to another bus (and see the address error response), then the attempts to look up new addresses will tend to be more spread out, avoiding flurries of activity.

4.2 Simplified reset acknowledge

I also suggest that the bits in the NODE_ENABLE register be turned back on after a reset by a lock compare/swap transaction rather than as a side effect of the RESET_ACKNOWLEDGE register. The RESET_ACKNOWLEDGE was important when the bus_id and generation number were used, but under this proposal it would have no real purpose.

4.3 Elimination of quarantine bit

I propose the it is possible to define certain behaviors of the flow of async packets through bridges such that the quarantine bit is no longer needed. This in turn eliminates the need for deciding when it is safe to turn the quarantine bit off.

This method does involve destroying more packets in the net than is strictly necessary when a reset occurs. This then causes more retransmissions and possibly more disruptions in async communications (note that this does not affect isochronous streams). The benefit of this method is that it is simpler, because it relies only on local actions of portals rather than "end to end" actions between portals and the bridge manager.

My proposal is that bridges do the following when a reset notification is received:

1. disable transmission of routed async packets from all portals (including aborting any packet currently being transmitted)
2. clear NODE_ENABLES
3. clear all routed async packets from the internal buffers
4. enter a state where received routable async packets are replied to with the usual ack_pending, but are discarded
   a) or maybe reply with address_error?
5. forward the reset notification to adjacent bridges
   a) directed transaction to portals on same buses as your own portals
   b) follow the tree structure to avoid loops
6. stay in the discarding state until at least two fair arbitration gaps have occurred without any routable async packets arriving
   a) think about this delay - is two gaps enough?
7. after the delay in step 6 has elapsed, return to normal routing
   a) note that the NODE_ENABLES register still requires individual nodes to enable themselves
   b) individual nodes can enable themselves any time, but if they try to do remote transactions to search for remote nodes too soon, the transactions will be discarded

I think that this should reliably flush all remote transaction packets out of the net, and then reopen routing in a way that is safe. Note that this method acts as a "wave front" proceeding outward from the bus that was reset.

5. Other notes

- The heartbeat process needs to be aware of resets, since the heartbeat is a routed async transaction.
- It may be useful to include something in the reset notification that indicates whether it is a "simple reset" (no change to net topology) or a "topology change reset". The portals that are attached to the reset bus (the portals that initiate the reset notification) could scan their local bus after the reset to see if any portals have been added or deleted over the reset. If no additions or deletions have happened, then it is a simple reset. Otherwise, a topology change is reported. The topology change notification could make all bridges hold in the "discarding" state, waiting for the bridge manager to reconfigure the net.
- Another possible enhancement could be implemented if the portals attached to a reset bus will scan the nodes on the bus after the reset. If the portals keep track of the EUI-64 values for every node on their local bus, then they can detect whether any phy_ids changed over the bus reset. If none changed, then no reset notification is needed.