1394 HCI OPEN

Host Bus Error Handling/Recovery

David R. Wooten

Compaq Fellow

Compaq Computer Corporation

1394 HCI OPEN

Host Bus Errors

- Split transaction time-out
- ECC/parity error on data
- ECC/parity error on address
- Bad address (e.g., memory not present)
- Bad operation (e.g., write to read-only memory)
- Protection violation (e.g., access not allowed)
- etc.

1394 HCI OPEN

Open HCI Perspective of Host Bus Errors

- All host bus errors have a single classification -- access didn’t work
- Error handling is unit/mode dependent

1394 HCI OPEN

DMA Units

- Self ID Buffer
- Physical Read Request
- Asynchronous Request Transmit
- Asynchronous Response Transmit
- Isochronous Transmit
- Asynchronous Request Receive
- Asynchronous Response Receive
- Isochronous Receive
- Physical Write Request

1394 HCI OPEN

Self ID Buffer

1394 HCI OPEN

Self ID Buffer

- **Errors that are detected by HC**
  - **memory write error (can't write)**
  - **buffer overflow (tried to write too much)**
  - **FIFO overflow (can't write fast enough)**
- **In all cases:**
  - **set SelfIDCount.*selfIDerror***
  - **drop remaining self ID packets**

1394 HCI OPEN

Self ID Buffer (cont.)

- In the case of an error, IntEvent.*selfIDcomplete* is not set until end of self ID process
- It is not an error if self ID process starts and HCControl.*rcvSelfID* is not set

Physical Read Requests

1394 HCI OPEN

Physical Read Request

- Read from address in first 4GB from a node that is enabled in PhysDMAEnable and AsynchronousRequestFilter
- If memory read error at start of packet, before header sent, send response with resp_data_error.

1394 HCI OPEN

Physical Read Request (cont.)

- If memory error after packet starts
  - truncate packet
  - if requester responds with ack_data_error, then re-arbitrate and send response with response code of resp_data_error
  - if requester response with anything else, drop the response

1394 HCI OPEN

Physical Read Request (cont.)

- If FIFO underrun after packet starts
  - truncate packet
  - if requester responds with ack_data_error, then re-arbitrate and try to re-send
  - if requester responds with anything else, drop the response

1394 HCI OPEN

Physical Read Request (cont.)

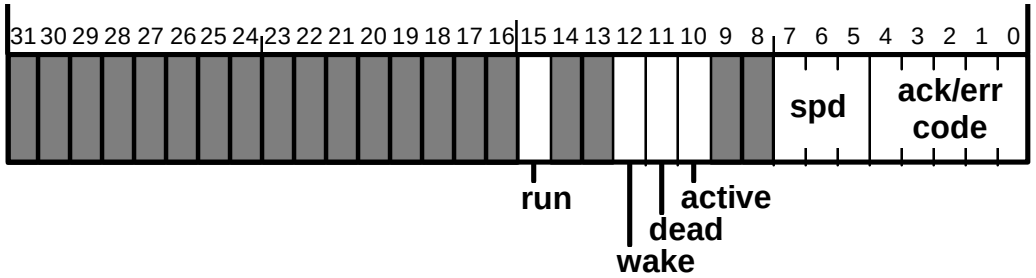
- On error, no interrupt event is generated and there is no logging of interrupt for statistics gathering.

1394 HCI OPEN

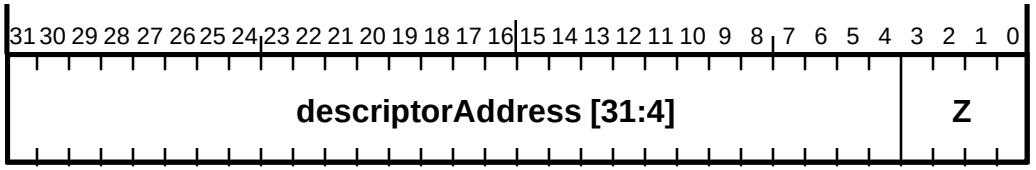
DMA Units with Standard Context

1394 HCI OPEN

Standard Context



ContextControl



CommandPtr

1394 HCI OPEN

Units Using Standard Context

- Asynchronous Request Transmit
- Asynchronous Response Transmit
- Isochronous Transmit
- Asynchronous Request Receive
- Asynchronous Response Receive
- Isochronous Receive

1394 HCI OPEN

‘Standard’ Failures

- **Descriptor Read**
  - **host bus error**
- **Data Read**
  - **host bus error**
  - **FIFO underrun**
  - **invalid tCode**

1394 HCI OPEN

‘Standard’ Failures (cont.)

- **Data Write**
  - **host bus error**
  - **FIFO overrun**
  - **buffer overrun**
- **Descriptor Status Update**
  - **host bus error**

Descriptor Read Error

1394 HCI OPEN

Descriptor Read Error

- **All standard contexts have same behavior**
  - **truncate packet transmit**
  - **set ContextControl.*dead***
  - **set IntEvent.*unrecoverableError***
  - **set ContextControl.*ack* to evt_descriptor_read**
  - **CommandPtr.*descriptorAddress* points to a descriptor within the failed descriptor block**

1394 HCI OPEN

Descriptor Read Error (cont.)

- Failure behavior is the same if descriptor block read in parts or as a whole
- In case of multiple errors, descriptor read error error takes precedence

1394 HCI OPEN

Descriptor Read Error (cont.)

- **Do not:**
  - **set the ‘normal’ interrupt for the context in IntEvent**
  - **update xferStatus, resCount or timeStamp**

Status Update Error

1394 HCI OPEN

Status Update Error

- Occurs when there is a host bus error in writing to the xferStatus and resCount/timeStamp in a descriptor
- Writing status quadlet to input buffer is <u>not</u> considered a status update error. It is a ‘data write’ for error purposes.

1394 HCI OPEN

Status Update Error (cont.)

- For all units with the ‘standard’ context the behavior is the same
  - set ContextControl.*dead*
  - retain ContextControl.*ack* and ContextControl.*spd* for completed packet
  - set IntEvent.*unrecoverableError*
  - Do <u>not</u> optionally set the context’s ‘normal’ IntEvent

Asynchronous Transmit Request and Response

1394 HCI OPEN

Data Read Errors

- Data read error can only occur on block write request, lock request, block read response or lock response
- Three types of errors
  - host bus read error
  - FIFO underrun
  - invalid tCode

1394 HCI OPEN

Host Bus Read Error

- Truncate packet and advance to last in descriptor block (OUTPUT_LAST*)
- Wait for response (unless broadcast)
- If response is ack_data_error then change ContextControl.*ack* to evt_data_read
  - else save response code as returned

1394 HCI OPEN

Host Bus Read Error (cont.)

- Conditionally update status in OUTPUT_LAST*
- Conditionally set context event bit in IntEvent

1394 HCI OPEN

FIFO Underflow

- **Behavior same as data read error except the replacement *ack/error code* is evt_underrun**

1394 HCI OPEN

tCode Error

- Means that tCode not appropriate for context (tCode ‘A’ in async context)
- tCode error also means that header length doesn’t match tCode
- Don’t put header into output FIFO -- don’t start packet
- Set ContextControl.*ack* to evt_tcode_err
- Conditionally save ContextControl in xferStatus of OUTPUT_LAST* and conditionally generate interrupt

1394 HCI OPEN

Isochronous Transmit

1394 HCI OPEN

Data Access Errors

- **Data read**
  - **host bus read error**
  - **FIFO underrun**
  - **invalid tCode**
- **Data write**
  - **Host bus write error during STORE_VALUE**

1394 HCI OPEN

Host Bus Read Error

- Truncate packet and advance to last in descriptor block (OUTPUT_LAST*)
- Set ContextControl.*ack* to evt_data_read
- Conditionally update status in descriptor containing OUTPUT_LAST and conditionally set IntEvent
- Continue

1394 HCI OPEN

FIFO Underflow

- **Behavior same as host bus read error except ContextControl.*ack* set to evt_underrun**

1394 HCI OPEN

tCode Error

- Means that tCode not appropriate for context (not iso tCode)
- tCode error also means that header length doesn't match tCode
- Don't put header into output FIFO -- don't start packet
- Set ContextControl.*ack* to evt_tcode_err
- Conditionally save ContextControl in xferStatus of OUTPUT_LAST* and conditionally generate interrupt

1394 HCI OPEN

Data Write

- Set ContextControl.ack to evt_data_write
- Skip to OUTPUT_LAST*
- Conditionally update xferStatus and set IntEvent

1394 HCI OPEN

A diversion...

Input FIFOs

1394 HCI OPEN

Input FIFOs

- Want to minimize number of input FIFOs
- Can combine things that never block into same FIFO
- Error strategy is to prevent any input FIFO from blocking so that they can all be combined
- Other strategies are possible - e.g., have multiple FIFOs.

1394 HCI OPEN

AR Response

- Prevention of buffer overrun of AR response context is responsibility of software
- If AR response buffer fills, response packets are dropped
  - leads to split transaction time-outs

1394 HCI OPEN

AR Request

- Dealing with buffer overruns of AR request context is responsibility of HC
- When overrun occurs
  - Packets that were given ack_pending by link are dropped
    - can't return resp_conflict_error because output queue would be blocking resource
  - Packets that were given ack_complete are reported to software or dropped

1394 HCI OPEN

Posted Write Requests

- If link sent ack_complete on write request before data is written to memory, packet is ‘posted’
- <u>May</u> drop, and not report, if write to offset above physical memory limit (48’h0000_FFFF_FFFF)
  - writes to CSR space are <u>never</u> posted

1394 HCI OPEN

Posted Write Requests (cont.)

- If posted to physical memory, local offset and source node ID reported to software through PostedWriteAddress registers
- Can't post more write requests than can be reported

1394 HCI OPEN

Buffer Space Tracking

- Can reduce number of request packets that are dropped by keeping track of amount of space in AR request buffer
  - do, pre-fetch of descriptors to determine remaining space (reqCount)
  - each reqCount value given to link to increase available space
  - as link accepts packets it reduces the available space
  - when link runs out of space, it sends ack_busy

1394 HCI OPEN

Dead Contexts

- **When context is *dead* or *run* not set link will 'reject' packets for the context -- <u>no ack sent!</u>**

and now back to...

Asynchronous Receive

1394 HCI OPEN

Asynchronous Receive

- Asynchronous receive only runs in buffer-fill mode.
- When HC can't write to buffer due to host bus error, buffer can no longer be 'parsed' by software.
- Therefore, unlike transmit, when data transfer error occurs (other than buffer full), HC must stop context

1394 HCI OPEN

Asynchronous Receive Request/Response

1394 HCI OPEN

Data Write Error

- Set ContextControl.*dead* (link 'rejects' packets for context)
- Set ContextControl.*ack* to evt_data_write
- Set IntEvent.*unrecoverableError*
- Drop packets that reach host side of input FIFO (only if shared FIFO)
  - if posted write, report failure to software
- CommandPtr points to descriptor where failure occurred

1394 HCI OPEN

FIFO Overflow

- Link side of FIFO truncates packet going into receive FIFO and sends ack_busy on 1394 bus
- System side of FIFO sees truncated packet and does not update status
- This makes it look like packet never happened (packet is ‘backed out’ of the buffer)

Isochronous Receive

1394 HCI OPEN

Buffer-fill Mode

- On FIFO/buffer overrun, back packet out of buffer
- On write error:
  - set *dead*
  - set *ack/error code* to evt_data_write
  - set *unrecoverableError* interrupt event
  - discard all packets for context

1394 HCI OPEN

Packet/buffer Mode

- **On data write error**
  - **set *ack/error code* to evt_data_write**
  - **conditionally update xferStatus in descriptor in which error occurred**
  - **continue as if packet was received correctly**

1394 HCI OPEN

?