Enhancing SBP-2 Performance

Eric Anderson
FireWire Software
Apple Computer, Inc.

Problem

- SBP-2 works well with large transfers and large task sets
- But small tasks, executed serially, suffer from high start-up latency

- Focus of this proposal:

  Disk drives and other fast storage devices

SBP-2 Review

- Initiator - for example, a computer
- Target - for example, a disk drive
- 1394 allows the target to directly access memory in the Initiator without interrupts or software overhead

Operation Request Block

- Initiator assembles ORB(s) in system memory, each with:
  - A command (approximately 12 bytes)
  - A data buffer for I/O (usually)
  - A next_ORB pointer (may be null)
  - Various transfer parameters (direction, length, etc.)
- A typical ORB is 32 bytes (may vary)
- ORB Pointer is 64-bit 1394 address of ORB structure

SBP-2 Operation
(first time)

- Initiator writes Target's ORB Pointer register
- Target sees write (1394 8-byte block write) and:
  - Reads ORB from Initiator memory
  - Executes ORB
  - Transfers data to or from buffer in 1394 memory
  - May send status to Initiator upon ORB completion
  - Executes additional ORBs linked to first ORB (if any)
  - Enters Suspended (idle) state when final ORB complete

SBP-2 Operation

(after first time - normal)

- Initiator writes new ORB Pointer into final ORB's next_ORB field (in Initiator memory)
  - Target may or may not have reached final ORB
- Initiator writes Target's Doorbell register
  - If Target is Suspended (idle), Target will re-examine next_ORB field in final ORB
  - If Target is not Suspended, Doorbell has no effect
- Either way, the new ORB will be executed as soon as possible

SBP-2 Operation

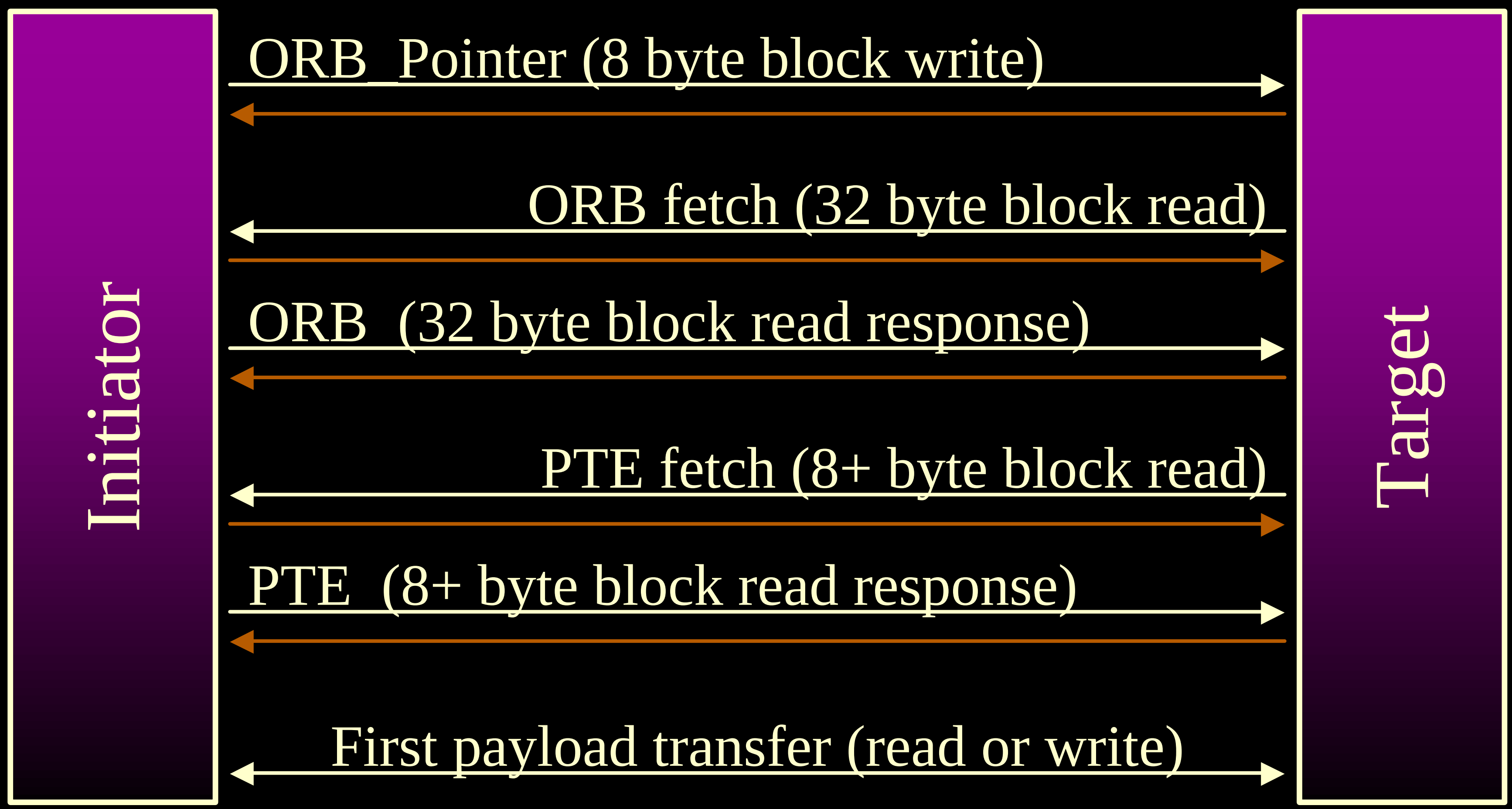
(after first time - special)

- Initiator knows Target is in Suspended (idle) state
  - Possible if all previous ORBs have completed
- Initiator may write a new ORB Pointer directly to the Target's ORB Pointer register
- Target will fetch the new ORB immediately
- Indirection of re-fetching the previous ORB is avoided
- This optimization is allowed by SBP-2 today

SBP-2 Start-up Latency

- To execute an ORB, the Target must know
  - The ORB Pointer
  - The ORB contents (command, buffer, etc.)
  - Page Tables (for some transfers)
- Target needs three transactions on 1394 to learn each of these three items
  - Each is a small packet on 1394, so arbitration (5-6 times) and overhead (headers and CRC) dominate
  - Between each packet, other nodes may use the bus

SBP-2 Start-up Latency



Does Latency Matter?

- If Target device is kept busy with multiple requests, Target can fetch ORBs in advance (in parallel with ORB execution)
- If transfers are large, payload I/O dominates bus usage, and 1394 efficiency is high
- But this usage is not typical on most computers

When Latency Matters

- Small transfers, such as VM paging
  - Just 4K per I/O; only on demand; random addresses
- Serial transfers
  - Each I/O must complete before next can be requested
  - Example: Directory scan (transfers are small too)

Proposal: Fast-start Packet

- Initiator sends ORB pointer, ORB contents, and page table together in one 1394 packet
- "Fast-start" packet is larger, so more efficient
- Several 1394 arbitration cycles are avoided
- Only used when Initiator knows Target is Suspended (idle)
  - Otherwise, use traditional ORB append (Doorbell)

Fast-start Packet Format

ORB_Pointer (8 bytes)
ORB (32 bytes is typical)
PTE(s) (8 bytes or more)

Implementing Fast-start

- Few new demands on Target
  - Target already stores ORB Pointer, ORB contents, and page table entries (at least one)
  - Target must indicate fast-start capability
  - Target must recognize fast-start packet
- Few new demands on Initiator
  - Initiator must assemble Fast-start packet
  - Initiator must identify fast-start packet as such
  - Initiator "good behavior" reduces Target complexity

Indicating Fast-start Capability

- New Key in Target's 1394 Config ROM
  - Root, Unit, or LUN directory
- Or, indicate in SBP-2 Login response
- Proposal: New key in Unit or LUN directory
  - Not all LUNs (or Units) created equal
  - New key ignored by legacy software
- Cost: 4 bytes in Config ROM

Identifying a Fast-start Packet

- ORB_Pointer register is 8 bytes
  - Too small to absorb Fast-Start packet
- Proposal: Define a new register address based on Fetch Agent address
  - SBP-2 already allows vendor / protocol-specific registers in this area
  - So, allow the Target to indicate an offset in the new key

Fetch Agent Registers

Agent_State
Agent_Reset
ORB_Pointer
Doorbell
Unsolicited_Status_Enable
reserved
reserved

SBP-2 Features Key

- 4-byte key in 1394 Config ROM indicates:
  - Capability for Fast-start
  - Maximum PTE count in Fast-start packet
  - Address of Fast-start register
- Key has room for additional SBP-2 features

SBP-2 Features Key

$3E_{16}$	reserved	fs	offset
8	13	3	8

- $3E_{16}$: proposed Key_Type / Key_Value pair
- fs: fast_start_feature
  - Zero if not supported
  - Encodes max PTE count in Fast_start packet (2^{n-1})
- offset: location of Fast_start register
  - Relative to Fetch agent base location (in quadlets)

Summary of Fast-start Advantages

- Reduced latency for small and / or serialized I/O
- More efficient use of 1394 bus
- No performance loss in any scenario
- Full backwards compatibility
- Completely optional
- Low cost in ROM and Target firmware

Contact Information

Questions, suggestions, and discussion are welcome:

Eric Anderson
ewa@apple.com
+1-408-974-1394