CRC Proposal

Presentation to T10

May 4th, 1998

Bruce Leshay

Quantum, Corp

Why do we need CRC?

- As speed increases, margins decrease, ISI increases, parity becomes insufficient solution
- Parity also does not detect REQ/ACK mismatch errors
- Errors as likely, or more likely, to happen on data lines as on REQ/ACK
- Use CRC in double-edge implementations only

Which CRC?

- Use Fibre Channel 32-bit CRC algorithm
  - proven standard
  - well documented
  - existing implementations

Quantum
CAPACITY FOR THE EXTRAORDINARY™

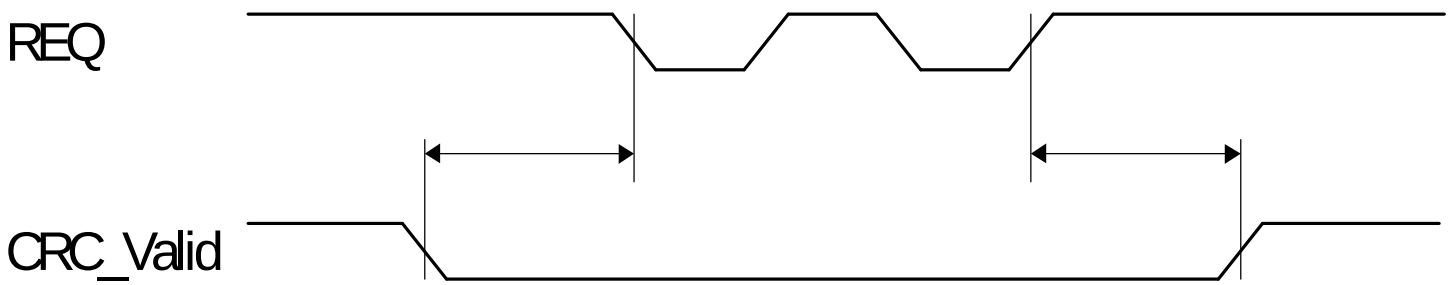
When does CRC happen?

- Allow target to choose when CRC is transmitted
  - Keeps interface target-driven
  - Avoids problems with weird block sizes (mode pages, etc) & variable block length devices (tapes)
- Parity[0] re-used as CRC Valid signal
  - During Data phase, parity signals will always be driven from target to initiator
  - parity
    - not needed if using CRC
    - nearly useless given error mechanism
    - currently routed similar to data lines
  - other potential signals
    - might use up available bus phases (MSG)
    - might cause unforeseen behavior by existing devices (SEL)
    - routing is very different from data

Example DataIn Phase

1. Target transfers data bytes
2. Target asserts CRC Valid
3. Target transfers CRC bytes
4. Target deasserts CRC Valid
5. Initiator checks CRC, reinitializes CRC checker
6. Target ends transfer or returns to step 1

CRC Timing



Example for wide transfers

Quantum
CAPACITY FOR THE EXTRAORDINARY™

DataOut Phase

1. Target asserts REQ's for data bytes
2. Target qualifies REQ's with CRC Valid
3. Initiator "marks" REQs which have CRC Valid asserted
4. Initiator calculates CRC as it transfers data
5. When initiator reaches ACK count corresponding to "CRC REQs", initiator transfers CRC and reinitializes CRC generator
6. Target counts ACKs, knows which bytes are CRC bytes, checks CRC and reinitializes checker

Odd Byte Cases

- When target transfers a multiple of four bytes and then asserts CRC valid, everything simple
- When target transfers something other than a four byte multiple, both initiator and target append fill character (0) up to 32 bit boundary in their internal CRC calculation
- Features:
  - No extra bandwidth for pad byte transfers
  - Devices don't need to "know" when to ignore pad bytes?

    (Device might allocate exactly enough room in a buffer, and have no place to put the pad)

Why no CRC Interval?

- Variable block size devices (tapes)
  - Could need a different CRC interval for each command
  - Tapes often disconnect at random points in the transfer
    - An interval would mean major restructuring
- If transfer is less than the interval:
  - Initiator is unaware the last four bytes are CRC until after phase change and status/message REQ received
  - CRC error reported during the Status/ Message Phase, for the previous Data phase

Double-Edged Clocking

- Double-Edged Clocking Problem
  - After odd number of transfers, REQ/ACK are left in asserted state
- Solution: Transfer extra "CRC symbol"
  - Target issues additional REQ edge after last CRC bytes, with CRC Valid asserted
  - Devices know CRC is four bytes, symbol easily ignored/discarded by hardware

Error Handling

- CRC error - treat same as parity errors
- Additional protocol errors
  - Termination of data phase without CRC
  - Too few/too many CRC bytes
  - Similar to illegal phase change
  - Initiator reports as Initiator Detected Errors