
PCI
SIG®

Conversion of PCI-X® ASIC to PCI Express® 1.1

Grant Wheeler/Ryan Haraden

Engineering and Technology Services

IBM

Agenda

- Why convert PCI-X design to PCI Express?
- How to convert PCI-X design to PCI Express
- Real ASIC example
- Gotchas
- Summary

Why convert PCI-X design to PCI Express?

- Market forces drive development
  - ✓ Support a new set of emerging IO adapters
  - ✓ Quick response to changes
  - ✓ Schedules are more aggressive
- Market forces drive us to a lower cost solution
  - ✓ More performance/function with less cost
  - ✓ Reuse of existing infrastructure allows meeting budgetary goals

Why convert PCI-X design to PCI Express?

- Send and receive data simultaneously
- Fewer pins
  - ✓ 4x requires only 16 data pins
- Efficient data transfer
  - ✓ A credit based protocol alleviates waiting for bus arbitration
  - ✓ Full duplex data transfer eliminates bus idle time in turning around the bus
- Easily scalable
  - ✓ If you require more bandwidth, add more lanes.

Why convert PCI-X design to PCI Express?

- Hot Plug implementation simplified
  - ✓ Standardized set of registers
    - Much of the complexity is pushed to software
    - Allows standard drivers to be used
  - ✓ Provides signal de-bouncing on input pins
  - ✓ Provides event notification and command completion status through interrupt mechanism
    - Simple interrupt implementation

Agenda

- Why convert PCI-X design to PCI Express?
- How to convert PCI-X design to PCI Express
- Real ASIC example
- Gotchas
- Summary

How to convert PCI-X design to PCI Express

- How can we shorten product development time?
  - ✓ Reuse the same code and software interfaces
  - ✓ Don't reinvent, instead use standard IP from vendors/foundry
- How can we reduce product cost?
  - ✓ Shorten product development time
  - ✓ Simplify product integration

How to convert PCI-X design to PCI Express

- Hardware similarities with PCI-X
  - ✓ Ordering rules
  - ✓ Overall transaction types
    - Memory reads/writes, IO reads/writes, config reads/writes
  - ✓ Buffering
  - ✓ Software configuration
    - Same base register set as previous generations
  - ✓ System configuration/architecture

How to convert PCI-X design to PCI Express

- PCI Express is software compatible with previous PCI versions
- PCI interfaces are typically well defined. This allows designers to pick up/mix and match cores
  - ✓ Reduces development time and risk
  - ✓ Allows future migration
- So replace the PCI-X core with a PCI Express core

Agenda

- Why convert PCI-X design to PCI Express?
- How to convert PCI-X design to PCI Express
- Real ASIC example
- Gotchas
- Summary

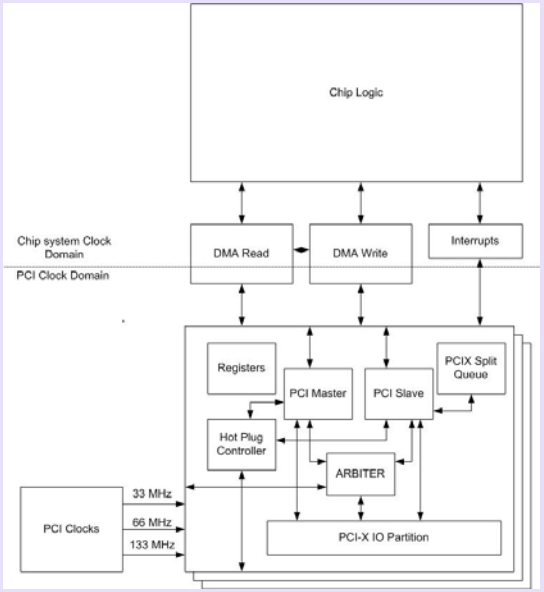
Real ASIC example

- Problem: How to enable PCI Express on the shortest possible schedule
  - ✓ Time to market was critical when developing this solution
    - Achieved ASIC tape-out 5 months after funding

- Solution: Convert a PCI-X ASIC to PCI Express
  - ✓ Technology unchanged (130 nm)
  - ✓ External interface doesn't change – logic reuse
  - ✓ PCI Express cores integrated into the ASIC with minimum logic changes
  - ✓ Software designed to drive PCI-X ASIC will continue to work with minimum firmware changes for new PCI Express functionality.

Real ASIC example

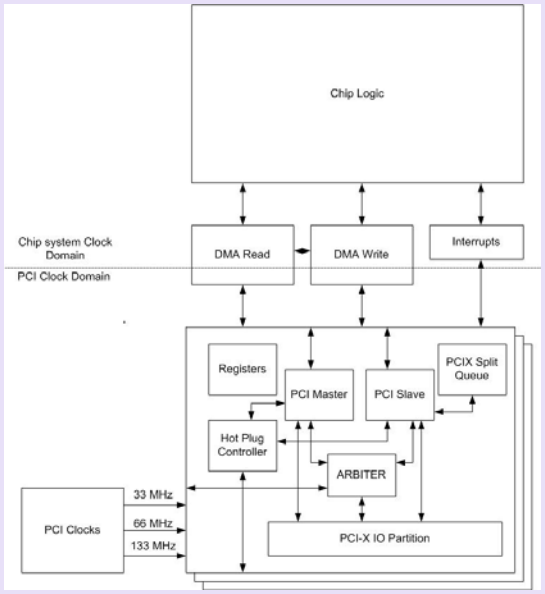
PCI-X ASIC



- DMA Read and Write buffers serve as asynchronous interface between external interface and PCI-X cores
- PCI logic is clocked at the desired bus frequency

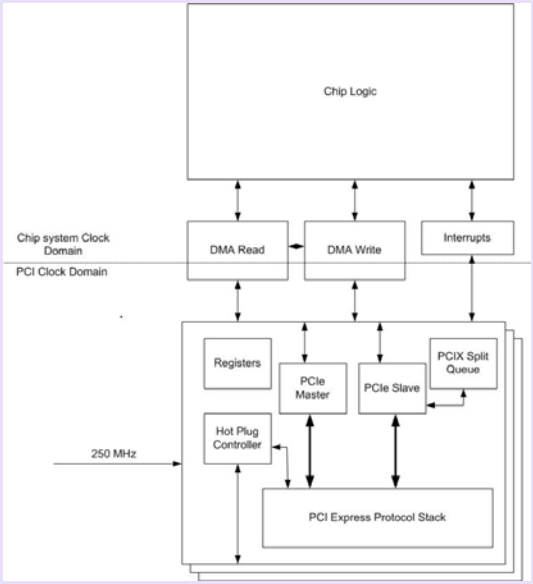
Real ASIC example

PCI-X ASIC



- DMA Read and Write buffers serve as asynchronous interface between external interface and PCI-X cores
- PCI logic is clocked at the desired bus frequency

PCI Express ASIC



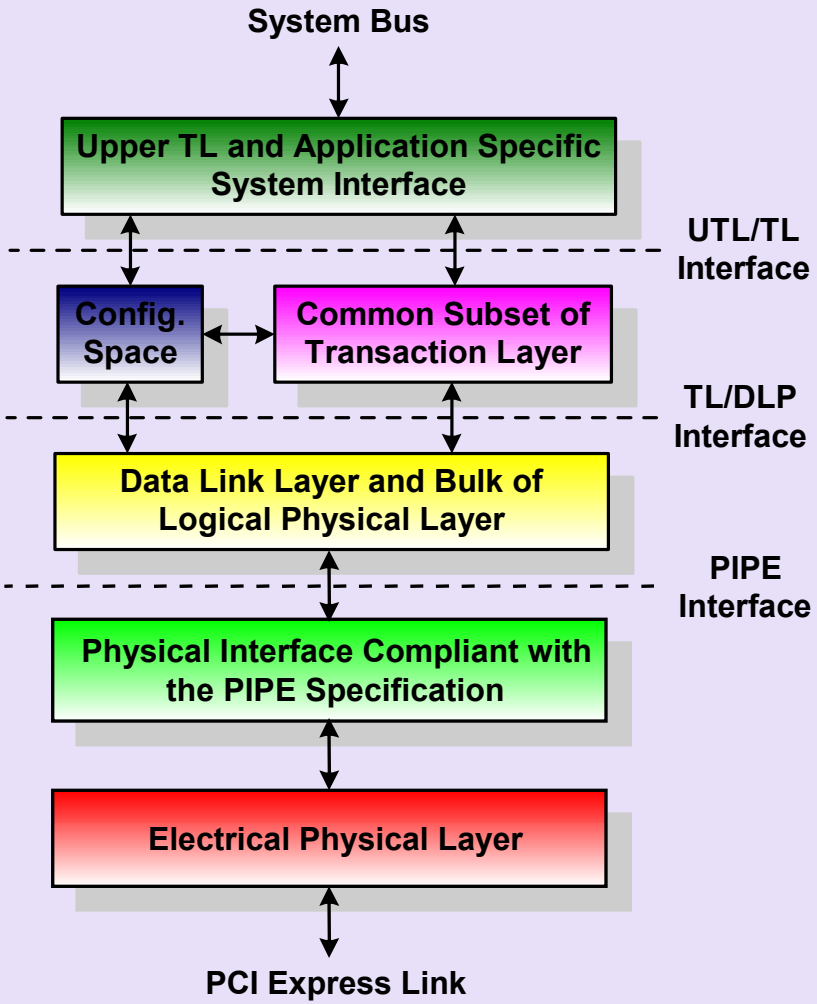
- DMA Read and Write buffers remain the asynchronous interface with rest of the ASIC
- PCI Express logic clocked at 250 MHz
- Changes in PCI master and slave logic
- Hot plug controller logic simplified

Real ASIC example

- Fundamental differences
  - Simultaneous transmit and receive
    - This will drive logic changes
  - Packet based serial protocol vs. request/grant
    - May drive logic changes
  - Single endpoint vs. a multi-drop bus
  - Credit based
    - Buffering decisions must be made
  - No more sideband signals (SERR , INT's)
    - The same functionality is provided through the link/core
  - PERR# (ECC for PCI-X mode 2) vs. end-to-end CRC checking across the link
    - Changes to error handling

Real ASIC example

- General Use PCI Express Cores
  - ✓ Technology Independent and Dependent Parts of the core
  - ✓ Which layer to interface to?
  - ✓ What to watch out for when picking your cores
    - Chip Size impacts
    - Buffering Requirements
    - Future Plans



Agenda

- Why convert PCI-X design to PCI Express?
- How to convert PCI-X design to PCI Express
- Real ASIC example
- Gotchas
- Summary

Gotchas

- Advanced PCI Express features
  - ✓ Multiple traffic classes & virtual channels
    - Bandwidth is divided into n channels
    - Guarantees bandwidth to a particular traffic class
    - Virtual channels have no ordering requirements between themselves and credits are handled independently per channel.
  - Requires independent buffering per channel
  - Logic duplication
  - New Software Support

Gotchas

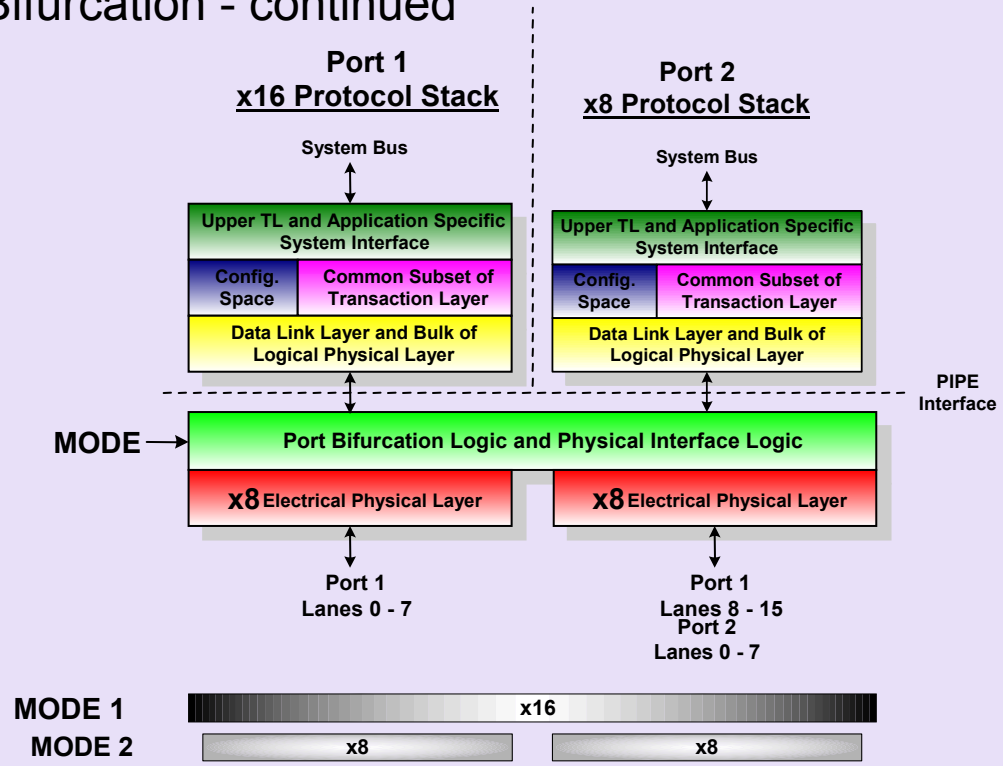
- Advanced PCI Express features
  - ✓ Virtual channel 0 is enough
    - For our application, high-performance end points have their own dedicated links and only low performance devices are being gathered under a switch network.
    - So when converting PCI-X designs to PCI Express take the simple route and have just virtual channel 0.
      - Saves buffering
      - Simplifies the design
      - Little if any functional loss for today's market
    - Future use
      - Delay use of multiple virtual channels until PCIe™ 2.0 applications.

Gotchas

- Advanced PCI Express Features
  - ✓ Bifurcation
    - Enables reuse of SerDes lanes for more than one link
      - A 16x link can become two 8x links
    - Implies the use of multiple protocol stacks
      - Good trade-off since protocol logic area < (SerDes + PHY) area
    - How do we convert this structure?

Gotchas

- Advanced PCI Express Features
  - ✓ Bifurcation - continued



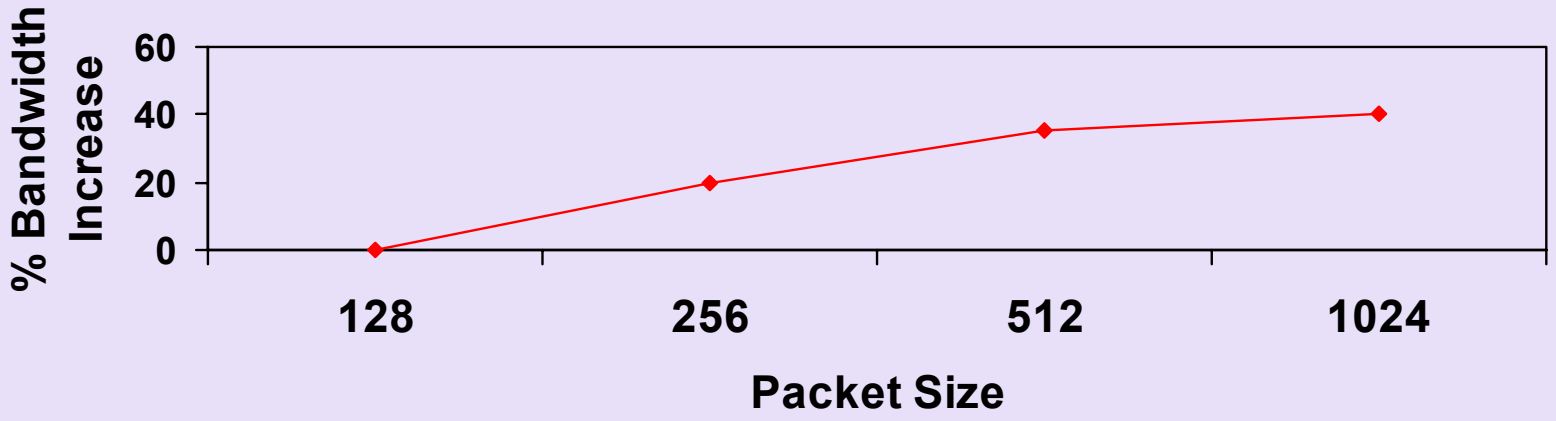
  - Simply consider each port an independent PCI-X bus in your old design
  - Port 2 is then disabled when the link is functioning at 16x

Gotchas

- Buffering
  - ✓ We added a SECOND set of buffering within the PCI Express core logic block.
  - ✓ This buffering is used for credit advertisement
    - Header Credits, Data Credits etc .
    - So you must ensure that you have enough buffering here to keep your express links full but this can be kept to a minimum as we have all the ‘OLD’ buffering above us.
    - Assume these buffers will be emptied very quickly and efficiently

Gotchas

- Buffering
  - ✓ Buffering also affects our maximum packet size
    - This could be your single greatest performance bottleneck
    - There are two trade off's : Bandwidth and Latency
      - Bandwidth indicates you want a large packet size
      - Latency says you want a small packet size
    - This argument is similar to what transaction size you should use in PCI-X
    - % Bandwidth increase is independent of link width



Gotchas

- Buffering
  - ✓ Buffering also affects our maximum packet size – cont.
    - What are the buffering differences going to these different packet sizes?
      - Doubling? No typically less
      - With Smaller packets you need more of them in flight
    - Our analysis indicated a 512 byte payload size was the sweet spot between bandwidth, latency, and buffering
      - Less than 512 bytes low link utilization and poor performance
      - Greater than 512 bytes if you have the chip area

Gotchas

- Master or Slave can now be sending and receiving at the same time (previously you could only be doing one or the other)
- Master and Slave arbitration for common data buffers now needed

Gotchas

- New register integration
  - ✓ Add in new register set while keeping it software compatible
    - New transaction layer registers
    - Expanded PCI configuration space for PCI Express
  - ✓ Overlapping Registers
    - Old PCI Configuration space

Gotchas

- New register integration
  - ✓ We chose to use the new core register partition instead of adding the registers to our existing design.
    - Removed our old Configuration space registers and used the core version instead.
    - Gets us the error handling changes
    - Makes our design more flexible for future modification.
    - We still have two registers partitions as the old one is needed to maintain backwards compatibility for other chip configuration registers

Gotchas

- How big was that SerDes/PHY?
  - ✓ Saving IO's doesn't mean saving ASIC die size unless you are IO bound
  - ✓ Area Sensitive?
    - – Reduce Link width? Not necessarily….
    - – 8x link core is ~38% larger than a 4x link core
      - Same buffering (4% larger without SerDes)
    - – 16x link core is ~71% larger than an 8x link core
      - Increased buffering (35% larger without SerDes)
- Some old logic needs to time @ 250 MHz
  - ✓ Critical paths are generally redesigned
  - ✓ May be an issue depending on locations of asynchronous interfaces.
  - ✓ Must ensure asynchronous interfaces are properly designed.

Gotchas

- Performance Considerations
  - ✓ Bandwidth
    - Where are the bottlenecks?
    - Were there any bandwidth limitations on the initial chip?
    - Sharing send and receive resources may limit bandwidth
  - ✓ External interface
    - What is the maximum bandwidth?
    - What are the latency requirements?
    - This may drive selection of PCI Express lane width

Gotchas

- Performance Considerations - continued
  - ✓ What about using a bridge to convert to PCI-X to PCIe instead?
    - Considered as a possible solution
  - ✓ Chose to convert existing ASIC instead
    - One chip has better performance than two
    - Is a cheaper system solution
    - Minimum design investment
    - Less overall system costs

Gotchas

- Ordering rules
  - ✓ Backward compatible with PCI-X
  - ✓ Re-examine the Y/N answers in the table if possible
  - ✓ REMOVE blocking on non-posted stores
    - Will reduce overall system performance if blocking is enabled.
    - PCI-X it was not typical to split the configuration write request (but was allowed)
    - Express this always occurs
    - Thus if you have the blocking rule you can slow down the response resulting in a longer stall to the processor than what is required.

Future road map

- PCIe 2.0
  - ✓ We are well positioned to take advantage of this when the cores become available.
  - ✓ We highly recommend a 16 byte data interface at @250 MHz to allow for future growth.
  - ✓ Our PCIe 1.x core solution has a 16 byte interface to the cores which allow data transfers up to 4 GB/s .
  - ✓ This will support up to an 8x PCIe 2.0 link and a 16X PCIe 1.x link without any changes besides core updates.

Agenda

- Why convert PCI-X design to PCI Express?
- How to convert PCI-X design to PCI Express
- Real ASIC example
- Gotchas
- Summary

Summary

- Able to achieve very aggressive schedule
  - ✓ 5 months from project funding to tape-out
    - High level of design reuse
      - Able to create ASIC out of previous partitions – building block approach
      - Started physical design early in the project
      - Testcase reuse
      - Simulation reuse
      - Large portion of board design reuse

Summary

- We met our goals
- Well positioned for future enhancements

- It was a rollercoaster ride, but we had fun doing it

Good Luck on your efforts!

Questions?

Thank you for attending the
PCI-SIG Developers Conference 2005.

For more information please go to
www.pcisig.com

Conversion of PCI-X 2.0 ASIC to PCI Express 1.1

Grant Wheeler/Ryan Haraden

Engineering and Technology Services

IBM


PCI
SIG®