Practical introduction to PCI Express with FPGAs

Michal HUSEJKO, John EVANS
michal.husejko@cern.ch
IT-PES-ES
Agenda

• What is PCIe ?
  o System Level View
  o PCIe data transfer protocol
• PCIe system architecture
• PCIe with FPGAs
  o Hard IP with Altera/Xilinx FPGAs
  o Soft IP (PLDA)
  o External PCIe PHY (Gennum)
System Level View

- Interconnection
- Top-down tree hierarchy
- PCI/PCle configuration space
- Protocol
Interconnection

- Serial interconnection
- Dual uni-directional
- Lane, Link, Port
- Scalable
  - Gen1 2.5/ Gen2 5.0/ Gen3 8.0 GT/s
  - Number of lanes in FPGAs: x1, x2, x4, x8
- Gen1/2 8b10b
- Gen3 128b/130b

Image taken from “Introduction to PCI Express”
Tree hierarchy

- Top-down tree hierarchy with single host
- 3 types of devices: Root Complex, Endpoint, Switch
- Point-to-point connection between devices without sideband signalling
- 2 types of ports: downstream/upstream
- Configuration space

Image taken from “Introduction to PCI Express”
**PCIe Configuration space**

- Similar to PCI conf space – binary compatible for first 256 bytes
- Defines device(system) capabilities
- Clearly identifies device in the system
  - Device ID
  - Vendor ID
  - Function ID
  - All above
- and defines memory space allocated to device.
PCle transfer protocol

- Transaction categories
- Protocol
- Implementation of the protocol
Transaction categories

- Configuration – move downstream
- Memory – address based routing
- IO – address based routing
- Message – ID based routing
## Transaction Types

<table>
<thead>
<tr>
<th>Transaction Type</th>
<th>Non-Posted or Posted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Read</td>
<td>Non-Posted</td>
</tr>
<tr>
<td>Memory Write</td>
<td>Posted</td>
</tr>
<tr>
<td>Memory Read Lock</td>
<td>Non-Posted</td>
</tr>
<tr>
<td>IO Read</td>
<td>Non-Posted</td>
</tr>
<tr>
<td>IO Write</td>
<td>Non-Posted</td>
</tr>
<tr>
<td>Configuration Read (Type 0 and Type 1)</td>
<td>Non-Posted</td>
</tr>
<tr>
<td>Configuration Write (Type 0 and Type 1)</td>
<td>Non-Posted</td>
</tr>
<tr>
<td>Message</td>
<td>Posted</td>
</tr>
</tbody>
</table>

*Table taken from “PCI Express System Architecture”*
Non-posted read transactions

Legend:
MRd = Memory Read Request
IORd = IO Read Request
CfgRd0 = Type 0 Configuration Read Request
CfgRd1 = Type 1 Configuration Read Request
CplD = Completion with data for normal completion of MRd, IORd, CfgRd0, CfgRd1
Cpl = Completion without data for error completion of MRd, IORd, CfgRd0, CfgRd1

Image taken from “PCI Express System Architecture”
Non-Posted write transactions

Legend:
- IOWr = IO Write Request
- CfgWr0 = Type 0 Configuration Write Request
- CfgWr1 = Type 1 Configuration Write Request
- Cpl = Completion without data for normal or error completion of IOWr, CfgWr0, CfgWr1

Image taken from “PCI Express System Architecture”
Posted Memory Write transactions

Legend:
MWr = Memory Write Request. No completions for this transaction

Image taken from "PCI Express System Architecture"
Posted Message transactions

Legend:
Msg = Message Request without data
MsgD = Message Request with data

Image taken from “PCI Express System Architecture”
PCIe Device Layers

- 3 layer protocol
- Each layer split into TX and RX parts
- Ensures reliable data transmission between devices

Image taken from “PCI Express System Architecture”
Physical Layer

- Contains all the necessary digital and analog circuits

- Link initialization and training
  - Link width
  - Link data rate
  - Lane reversal
  - Polarity inversion
  - Bit lock per lane
  - Symbol lock per lane
  - Lane-to-lane deskew
Data Link layer

• Reliable transport of TLPs from one device to another across the link
• It’s done by using DLL packets:
  o TLP acknowledgement
  o Flow control
  o Power Management
Transaction layer

- It turns user application data or completion data into PCIe transaction – TLP
- Header + Payload + ECRC
- used in FPGAs IPs

*Image taken from “PCI Express System Architecture”*
Flow control

Transmitter

Buffer space available

TLP

VC Buffer

Receiver

Flow Control DLLP (FCx)
Flow control – posted transaction

ACK returned for good reception of Request or Completion
NAK returned for error reception of Request or Completion
Flow control – non-posted transaction

1a. Request → 2a. Request
4b. ACK → 3b. ACK

1b. ACK → 2b. ACK
4a. Completion → 3a. Completion

ACK returned for good reception of Request or Completion
NAK returned for error reception of Request or Completion
Building transaction

Information in core section of TLP comes from Software Layer / Device Core

Bit transmit direction

Start | Sequence Number | Header | Data | ECRC | LCRC | End

Created by Transaction Layer

Appended by Data Link Layer

Appended by PHY Layer

Image taken from “PCI Express System Architecture”
Information in core section of TLP is sent to Software Layer / Device Core.

1. Stripped by Transaction Layer
2. Stripped by Data Link Layer
3. Stripped by PHY Layer

Bit receive direction
Example
CPU MRd targeting an Endpoint

Requester:
- Step 1: Root Complex (requester) initiates Memory Read Request (MRd)
- Step 4: Root Complex receives CplID

Completer:
- Step 2: Endpoint (completer) receives MRd
- Step 3: Endpoint returns completion with data (CplID)

Image taken from “PCI Express System Architecture”
CPU MWr targeting Endpoint

Requester:
- Step 1: Root Complex (requester) initiates Memory Write Request (MWr)

Completer:
- Step 2: Endpoint (completer) receives MWr

Image taken from “PCI Express System Architecture”
Endpoint MRd targeting system memory

**Completer:**
- Step 2: Root Complex (completer) receives MRd
- Step 3: Root Complex returns completion with data (CplID)

**Requester:**
- Step 1: Endpoint (requester) initiates Memory Read Request (MRd)
- Step 4: Endpoint receives CplID

Image taken from “PCI Express System Architecture”
Packet constraints

- **Maximum Payload Size (MPS)**
  - default 128 Bytes
  - least denominator of all devices in the tree

- **Maximum Read Request Size (MRRS)**
  - Defined by RC

- **Maximum Payload/Read req. size 4 kB**
  - defined by spec
  - No 4kB boundary crossing allowed

- **Example:** Intel x58: MPS=256B, MRRS=512B
HEADER description

- Little endian
- 3DW or 4DW (Double Word – 4 bytes)
HEADER – base part

- Fmt – size of the header, is there payload?
- Length – in DW
- EP – Poisoned
- TC – Traffic class
- TD – TLP digest – ECRC field
- Attr – status (success, aborted)
HEADER Memory Request

- TAG - Number of outstanding request
- Requester ID
HEADER Completion

- TAG - Number of outstanding request
- Requester ID

Image taken from “Introduction to PCI Express”
PCIe System Architecture

• Switches
  o Extend interconnection possibilities
  o DMA
  o Performance improvement functions
  o Non Transparent Bridging

• Extending distance
  o Bus re-drivers
  o Copper and optical cables
PCIe switches

- Non Transparent Bridging (NTB)
- Virtual Partitioning
- Multicasting
- DMA
- Failover
NTB + Virtual Partitioning
Cabling

- Copper cables
- Optical cables
- Cable re-drivers (repeaters)

Image taken from www.ioxos.ch


www.idt.com
PCIe with FPGAs

• Technology overview:
  o Hard IP – Altera and Xilinx
  o Soft IP – PLDA
  o External PHY – Gennum PCIe to local bus bridge
• Vendor documents – app notes, ref designs, Linux/Win device drivers
• Simulation – Endpoint/Root port
Xilinx Hard IP solution

• User backend protocol same for all devices
  o Spartan – 6
  o Virtex – 5
  o Virtex – 6
  o Virtex – 7

• Xilinx Local Link (LL) Protocol and ARM AXI

• For new designs: use AXI

• Most of the Xilinx PCIe app notes uses LL
Xilinx Hard IP interface

- External world: gt, clk, rst – (example x1 needs 7 wires)
- CLK/RST/Monitoring
- TLP TX if
- TLP RX if
- CFG if
- MSG/INT if
PCIe LL protocol

- TLP packets are mapped on 32/64/128 bit TRN buses
Xilinx simulation
RP <-> EP

- Gen1, x8, Scrambling disabled in CORE Gen v 1.0
How to design with Xilinx PCIe Hard IP

- Application notes
- Reference designs
- CORE Gen Programmable IO (PIO) hardware/simulation examples
XAPP 1052

- Block DMA in Streaming mode
- No CplD transaction re-ordering
XAPP 1052

- GUI for Win (VisualBasic)
- GUI for Linux (Glade)
- Driver for Win/Linux
XAPP1052 – performance

- Intel Nehalem 5540 platform
- Fedora 14, 2.35. PAE kernel
- Gen1, x4, PCIe LeCroy analyser
- DMA config
  - Host configures (MWr) DMA engine – around 370 ns between 1DW writes
  - Host checks DMA status: MRd (1DW) to CplD (1DW) response time – around 40 ns
- DMA operation:
  - DMA MRd(1st) -> CplD response time around 2.76 µs
  - DMA MRd(8th) -> CplD response time around 3.82 µs
  - DMA MWr -> around 750-800 MB/s (Gen1,
XAPP 859

- Block DMA: Host <-> DDR2
- Jungo Win device driver
- C# GUI
Scanning PCIe Slots for Xilinx ML555 or ML505
Found Xilinx ML555 PCIe Board
Attemping to allocate 1MB host memory DMA buffer
1MB host memory DMA buffer successfully allocated
GUI is now initialized and ready
Xilinx V6 Connectivity Kit

- PCIe to XAUI
- PCIe to parallel loopback
- VirtualFIFO based on DDR3 (MIG, SODIMM)
- Northwest Logic User Backend IP – Packet (SG)
- DMA

v 1.0
Xilinx S6 Connectivity Kit

- PCIe to 1 Gb Eth
- PCIe to parallel loopback
- VirtualFIFO based on DDR3 (MIG, Component)
- Northwest Logic User Backend – Packet (SG) DMA
Altera Hard IP solution

• Target devices:
  o Cyclone IV GX
  o Arria I/II GX
  o Stratix II/IV GX

• Similar to Xilinx in terms of user interface – TLP over Avalon ST or User application with Avalon MM
  o ST – streaming mode, for high performance designs
  o MM – memory mapped, for SOPC builder, lower performance

• CvPCle – FPGA reconfiguration over PCIe
  o I/O and PCIe programmed faster than the rest of the core
Altera Megacore Reference Designs

- Endpoint Reference Design
  - PCIe High Performance Reference Design (AN456) – Chained DMA, uses internal RAM, binary win driver
  - PCIe to External Memory Reference Design (AN431) – Chained DMA, uses DDR2/DDR3, binary win driver
- Root Port Reference Design
- SOPC PIO

- Chained DMA documentation
  - also Linux device driver available
- BFM documentation
  - Extensive simulation with Bus Functional Models
SOPC Based Design

- SOPC Builder Based
- Gen 1, x4
- DMA
- Sim and HW
AN431 – PCIe to DDR3

Diagram showing the components and integration of PCIe to DDR3 systems.
PLDA PCIe IPs

• XpressLite
  o currently available at CERN
  o Soft IP, Gen1 Endpoint only, x1/x2/x4
  o Stratix GX, Stratix II GX, and Arria GX support
  o No S4GX, C4GX and A2GX Hard IP support

• EZDMA2 Altera/Xilinx
  o Support Hard IP inside Altera: Cyclone IV GX, Arria II GX, and Stratix IV GX
  o Hard IP inside Xilinx: Virtex-5/6, Spartan-6
  o Same user/DMA interface as XpressLite

• XpressRich – rich version
  o Are you rich?

• Northwest Logic?
PLDA XpressLite

- Stratix GX, Stratix II GX, and Arria GX support only
  - No S4GX, C4GX and A2GX Hard IP support

- Generated with JAVA GUI: Windows/Linux
- Synthesis: single VHDL/Verilog encrypted file
- ModelSim: pre-compiled lib (Win/Linux)
- Ncsim: protected lib (Linux)
- Testbench: RP emulation

- Device drivers, API, tools (C++ source available)
PLDA XpressLite

- Maximum 8 DMA channels with Scatter Gather
- Reference design:
  - PCIe Lite – Endpoint only
  - Single DMA engine – C2S(WR) + S2C(RD)
  - Single target module – accepts WR/RD into SRAM/registers
External PCIe chips - Gennum

- TLP interface with simple framing signalling
- FPGA serial programming
  - FPGA can be reprogrammed without affecting PCIe link
- GPIO interface/Interrupts
- IP (with DMA) provided for Altera and Xilinx
- Device drivers and Software DK provided
- Already used at CERN:
  - Open source IP for Xilinx device developed by CERN group
  - Wishbone
  - SG DMA
  - device driver
  - More info www.ohwr.org
Gennum PHY + Spartan6

- Open source IP, SG DMA, device driver
More information

• Books:
  o Introduction to PCI Express – CERN Library (hardcopy)
  o PCI Express standards – CERN Library – CDS.CERN.CH
  o PCI Express System Architecture – mindshare.com (ebook+ hardcopy)
eda.support@cern.ch

- PCIe demos available on request
- IDT PCIe Switch dev. kit. coming soon
- Evaluating EZDMA2 for Xilinx.
Extras
Host configures (MWr) DMA engine – around 370 ns between 1DW writes
XAPP 1052 DMA Config RD

- MRd (1DW) to CplD (1DW) – around 40 ns
MRd to System Memory

- Intel Nehalem 5540 platform
- MRd(1\textsuperscript{st}) -> CplD response time around 2.76 $\mu$s
- MRd(8\textsuperscript{th}) -> CplD response time around 3.82 $\mu$s
XAPP 859 – Write

Start Signal from Register File

Done Signal to Register File

~220 ns DDR2 Latency

Back-to-back 128-Byte Maximum Payload Size Transfers

Wait States
XAPP 859 – Read

- x8 512-Byte Read Request Packets on TX TRN Interface
- Done Signal to Register File Interface
- Start Signal from Register File
- Roundtrip System Read Latency ~840 ns
- x64 64-Byte Completer Packets on RX TRN Interface
Endpoint TB
Root Port TB

Testbench Top-Level

- Root Port DUT
  (variation_name_example_rp_pipen1b)
- PIPE Interconnection Module x8
  (altpcierd_pipe_phy)
- EP Model
  (altpcietb_bfm_ep_example_chaining_pipen1b)

- Root Port BFM
  (altpcietb_bfm_driver_rp)
AN456 – Chained DMA
Endianness

- 0x12345678
- Big-Endian stores the MSB at the lowest memory address. Little-Endian stores the LSB at the lowest memory address. The lowest memory address of multi-byte data is considered the starting address of the data. In Figure 1, the 32-bit hex value 0x12345678 is stored in memory as follows for each Endian-architecture. The lowest memory address is represented in the leftmost position, Byte 00.