Keystone PCI Express Usage

Agenda

- PCIe Overview
- Keystone PCIe
- Address Translation
- Configuration
- MCSDK PCIe Demo

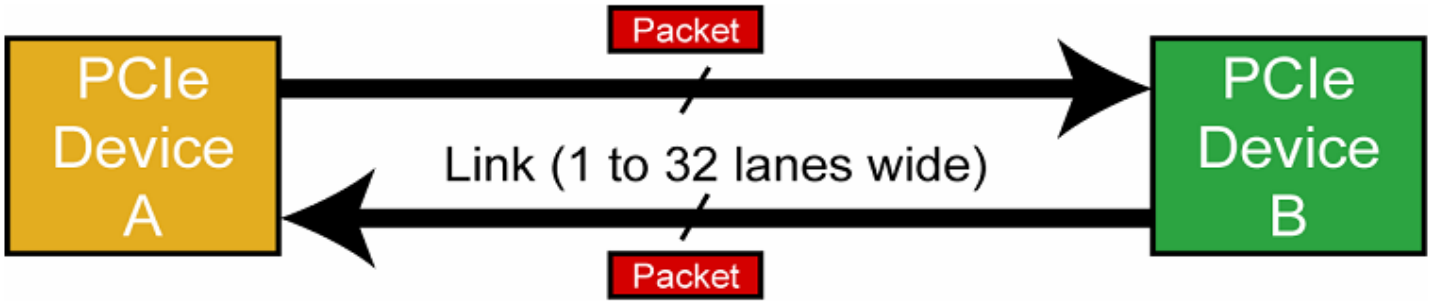
Agenda

- **PCIe Overview**
- Keystone PCIe
- Address Translation
- Configuration
- MCSDK PCIe Demo

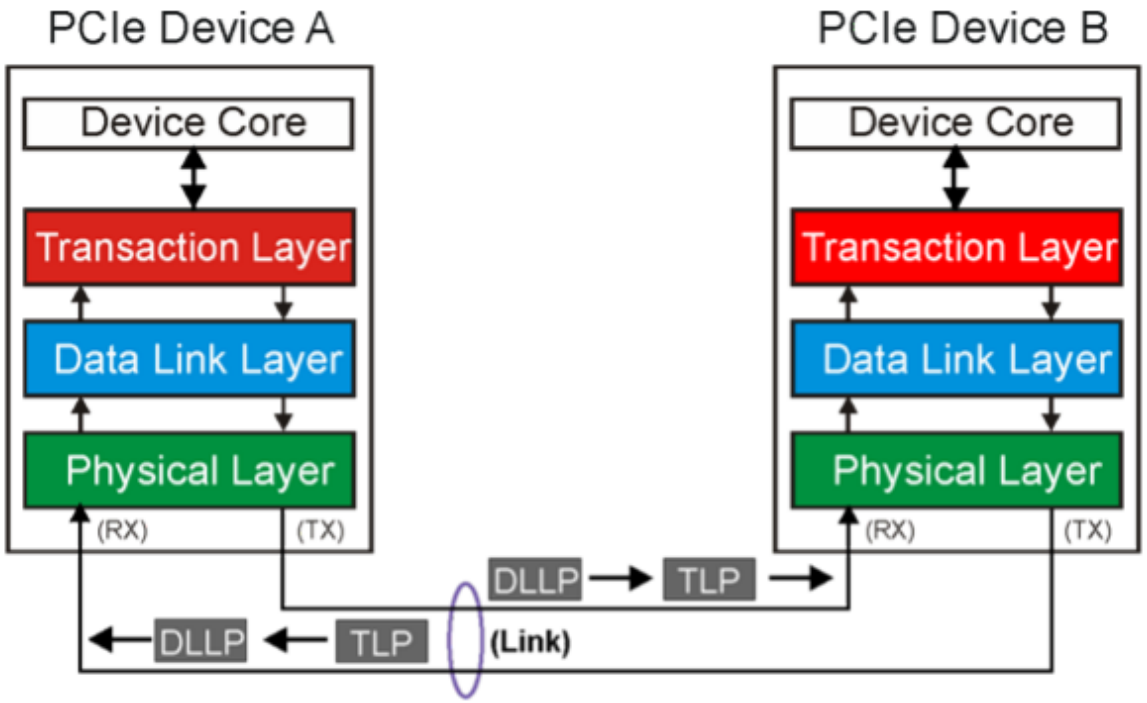
PCIe Overview: Introduction

Peripheral Component Interconnect (PCI) Express I/O incorporates:

- Serial point-to-point connection
- Multi-lane and scalable architecture: x1, x2, x4, x8, x12, x16, x32 lanes
- Symmetric design: Same number of lanes in each direction
- High-speed I/O with 2.5 Gbps / 5 Gbps in each direction simultaneously
- Packet-based transaction protocol:
  - Packet has unique identifier so response is directed to correct originator.
  - Packet format supports 32-bit or extended 64-bit memory addressing.
  - Packet lengths of up to 4 Kbytes are supported.
- Software backward compatible with PCI & PCI-X



PCIe Overview: Layered Protocol

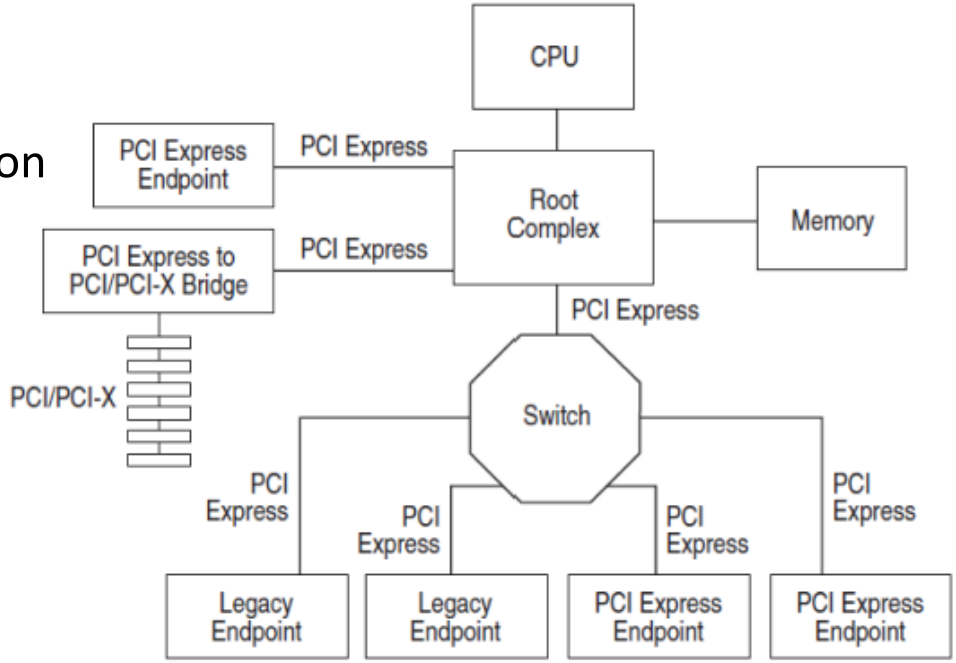


- **Transaction Layer Packet (TLP)**
  Memory R/W, configuration R/W, message request, IO read/write, etc.
- **Data Link Layer Packet (DLLP)**
  Ack/Nak, power management, flow control, etc…
- **Physical Layer**
  Link training (rate, width, reversal), scrambling, data distribution, etc.

PCIe Overview: Topology Example

Root Complex (RC)

- Connects CPU to PCIe topology
- May support one or more PCIe ports
- Each port connected to endpoint/switch
- Can generate PCIe transaction requests on behalf of CPU including: configuration, memory, I/O & message



Switch

- Provides fan-out and aggregation
- Connects multiple devices to each other

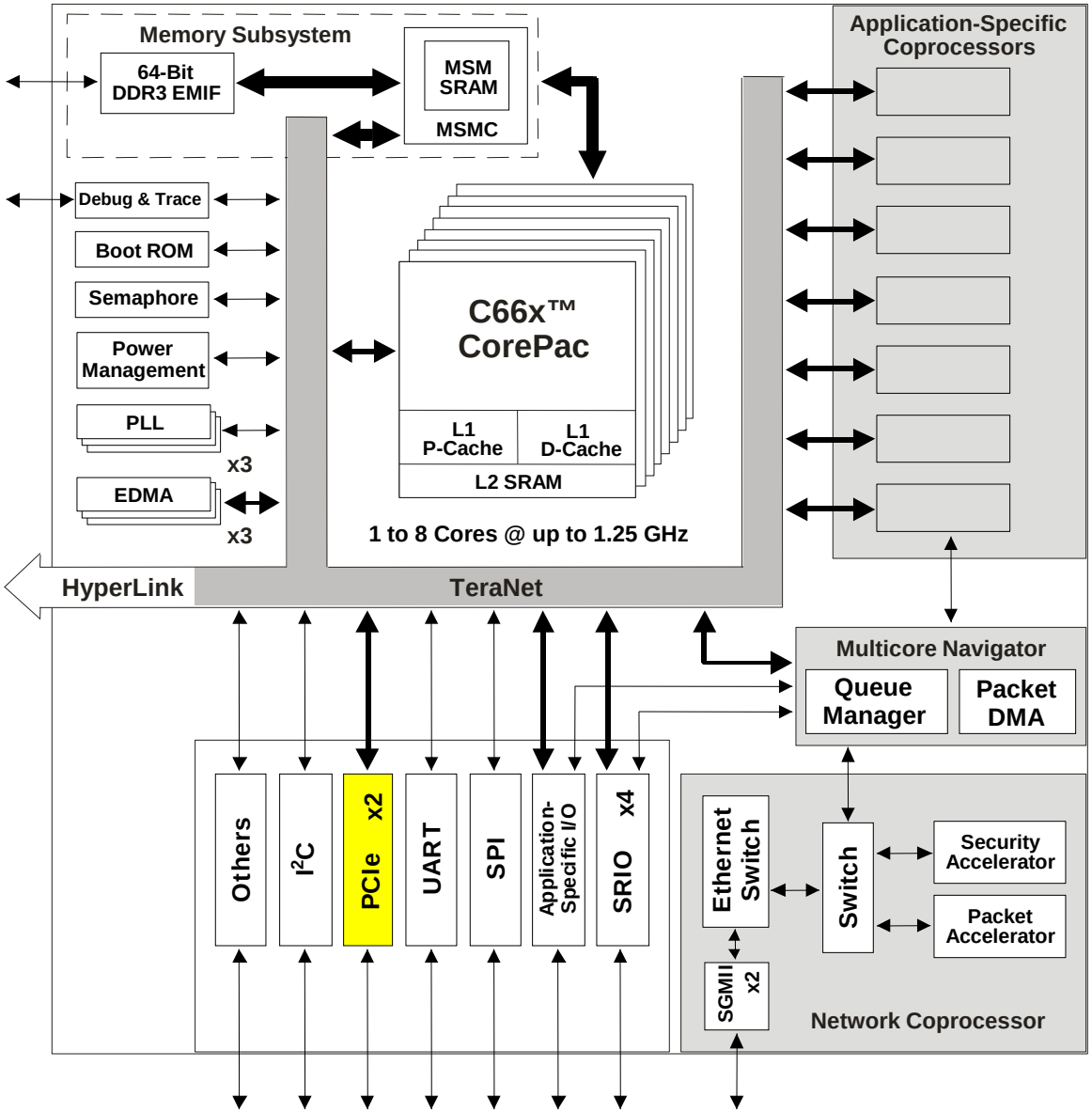
Endpoint (EP)

- Devices in a PCIe topology that are not RC and Switches
- Can act as requester or completer for PCIe transactions
- Two types of EPs exist:
  - Legacy EP: supports legacy-style and MSI interrupt generation
  - PCIe EP: supports MSI interrupt generation & 64-bit memory addressing

Agenda

- PCIe Overview
- **Keystone PCIe**
- Address Translation
- Configuration
- MCSDK PCIe Demo

Keystone PCIe



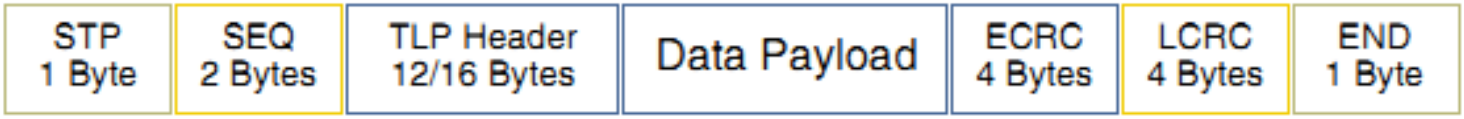
Keystone PCIe: Overview

- PCI-SIG: PCI Express Base Specification (Rev. 2.0)
- Root Complex (RC) and End Point (EP) operation modes
  - In EP mode, supports both legacy EP mode and native PCIe EP mode
  - Set from bootstrap pins PCIESSMODE[1:0] at power-up (00->EP, 01->Legacy EP, 10->RC)
  - Software can overwrite by changing PCIESSMODE bits in DEVSTAT reg.
- Gen1 (2.5 Gbps) and Gen2 (5.0 Gbps)
- Two lanes
  - Single Port: can connect to 1 PCIe device or 1 port of PCIe switch
  - Both lanes have to be configured to run at the same speed
- Outbound/Inbound max payload size of 128/256 bytes

KeyStone PCIe: Throughput

PCIe protocol overhead is as follows:

- Physical layer:
  - 20% reduction in bandwidth due to 8b/10b encoding
- Data link layer:
  - 1-5% reduction due to acknowledge and flow control updates
- Transaction layer:
  - Payload size influences throughput
  - Larger payload size reduces overhead
  - Typical packet overhead for TLP shown below



KeyStone PCIe: Throughput

	Payload Bytes	Header Bytes	ECRC Bytes	Link Bytes	Calculation	Percent Throughput
Worst Packet	16	16	4	8	16/44	36%
Typical Packet	128	16	0	8	128/152	84%
Typical Packet	256	16	0	8	256/280	91%
Typical Packet	512	16	0	8	512/536	96%
Best Packet	4096	12	0	8	4096/4116	99%

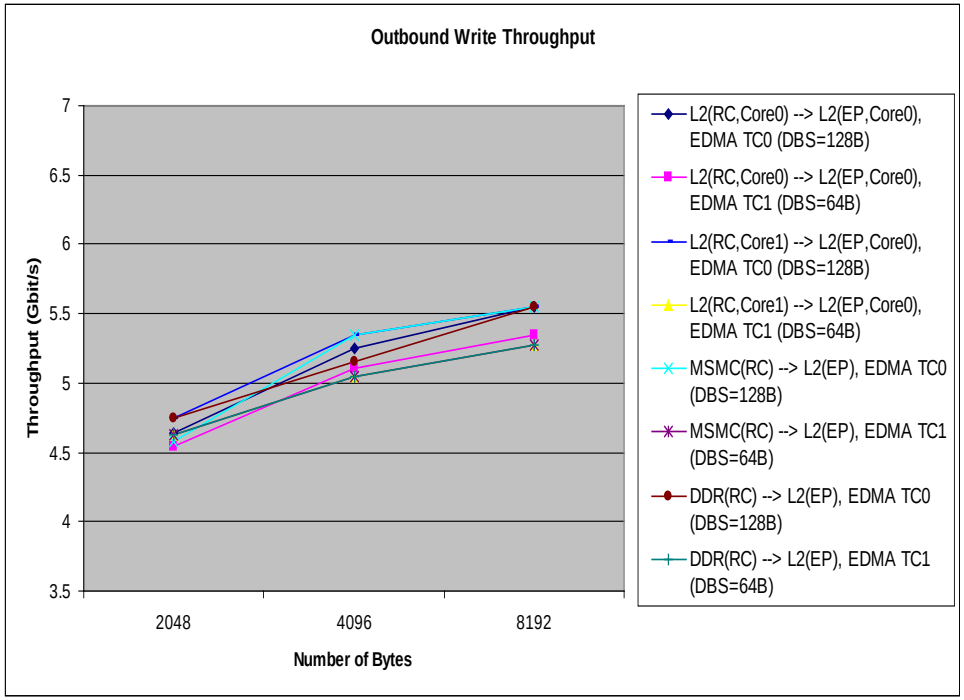
Relevant to Keystone:
128B payload, throughput is 84%

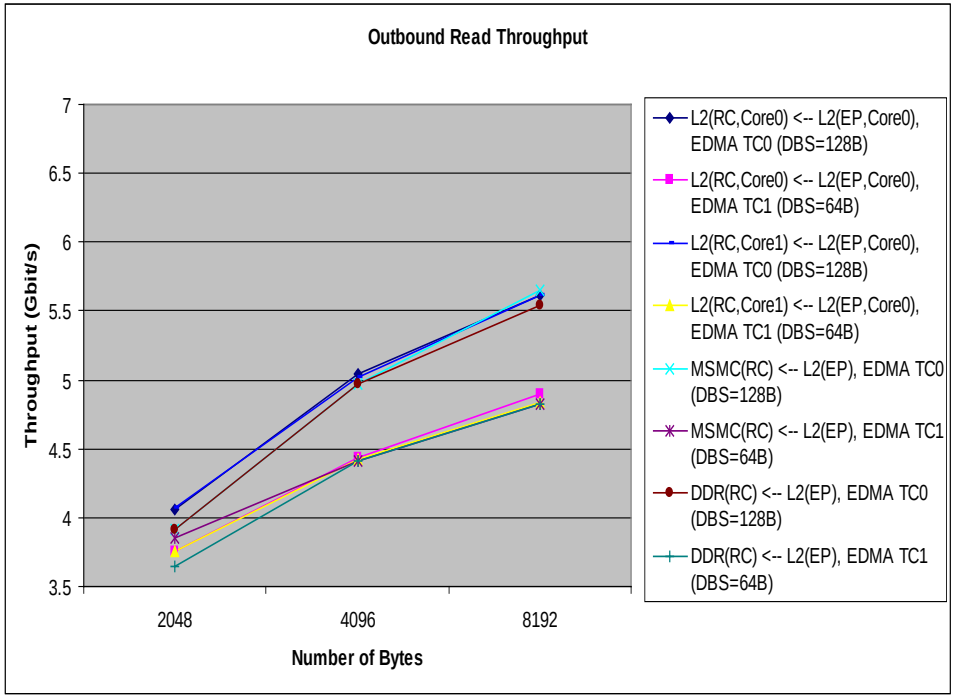
Max effective throughput: **840 MB/s**

Calculation for Keystone, PCIe Gen2,

5.0 Gbps × 2 Lane
= 10.0 Gbps × 8/10 (encoding)
= 8.0 Gbps × 1B/8bits
= 1000 MB/s × 84%
= 840 MB/s

KeyStone PCIe: Throughput





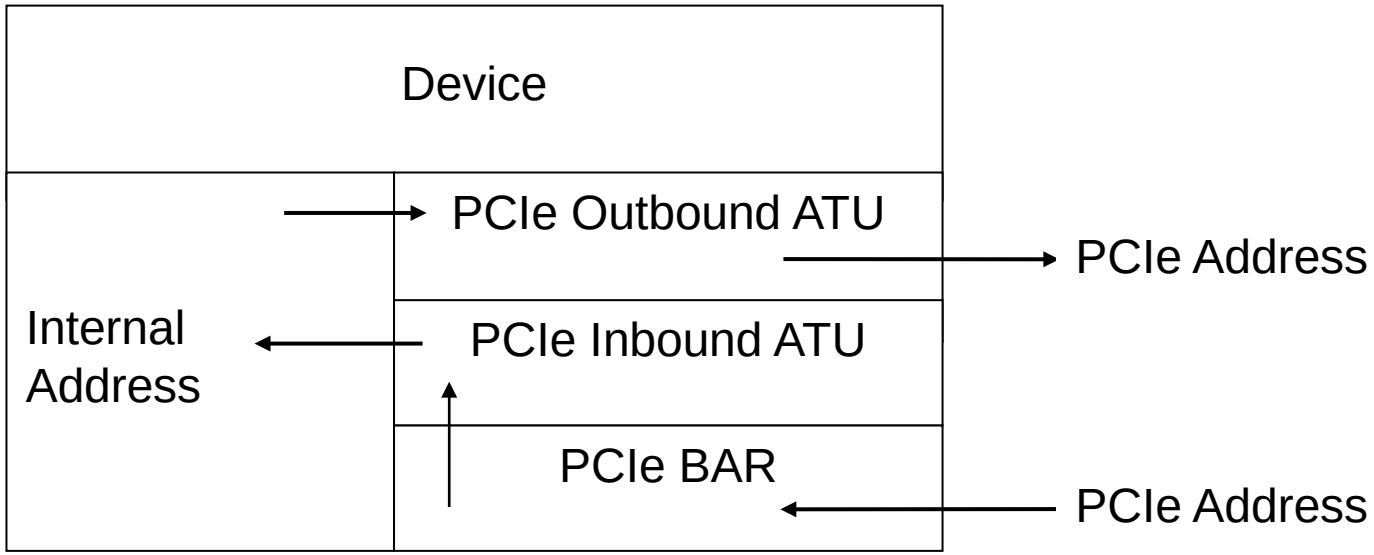
- Best to leverage EDMA for outbound transfer
- EDMA data burst size greater than PCIe payload size for optimal performance
- For 2 Keystone devices interconnected in RC → EP setup for outbound: Throughput = ~5.5Gbps = ~687.5MB/s for 2 Lanes (Gen2, 8KB buffer size)

Agenda

- PCIe Overview
- Keystone PCIe
- **Address Translation**
- Configuration
- MCSDK PCIe Demo

Address Translation

- PCIe device uses PCIe address to Tx/Rx packets over a PCIe link
- Address Translation Unit (ATU) within PCIe module
- ATU translates between device internal address and PCIe address
- Base Address Register (BAR) accepts/rejects based on incoming address



Address Translation: Outbound

- Local device initiates transactions to write to / read from external device
- PCIe module does not have built-in EDMA, PKTDMA, but CPU or device-level EDMA is used for outbound data transfer
- Outbound ATU will translate device internal address into PCIe address
- Data with PCIe address will be transferred over PCIe link to other device
- Outbound ATU can be enabled (1) or disabled (0) by CMD_STATUS[OB_XLT_EN] register field
- PCIe data space: 0x6000_0000 to 0x6FFF_FFFF (256MB range)
- Equally divided into 32 regions (2^5 = 5 bits required)

Address Translation: Outbound

Registers for outbound (OB)

- **OB_SIZE**:
  Identify size of 32 equally-sized translation regions as 1MB/2MB/4MB/8MB

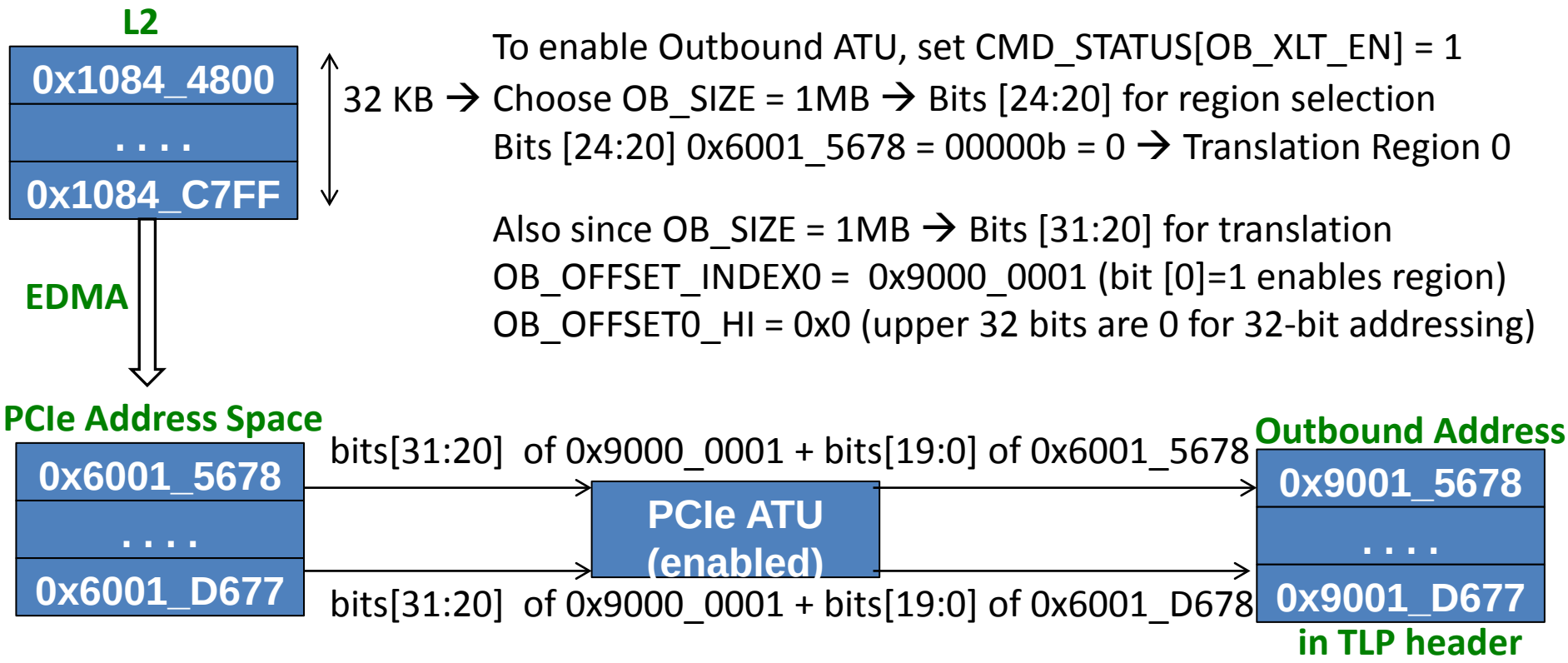
- **OB_OFFSET_INDEXn**
  - Represent bits[31:20] of PCIe address for 32-bit or 64-bit addressing
  - Not all bits will be used (depend on OB_SIZE)
  - Bit[0] enables the outbound region

OB_SIZE	OB_OFFSET_INDEXn	
	Region indexing	Translation
0 (1 MB)	[24:20]	[31:20]
1 (2 MB)	[25:21]	[31:21]
2 (4 MB)	[26:22]	[31:22]
3 (8 MB)	[27:23]	[31:23]

- **OB_OFFSETn_HI**: represent bits[63:32] of the PCIe address for 64-bit addressing; must be zero for 32-bit addressing

Address Translation: Outbound

EXAMPLE: Outbound write, source buffer 32KB in L2



Address Translation: Inbound

- External device initiates transactions to write to or read from local device
- PCIe module has a master port to transfer data to or from device memory
- No CPU or EDMA is needed for inbound transfer to the local device
- BAR in PCIe module accepts certain PCIe addresses and rejects others
- Data with accepted PCIe address → inbound ATU for translation → device internal memory
- Inbound ATU can be enabled (1) or disabled (0) by CMD_STATUS[IB_XLT_EN]

Address Translation: Inbound

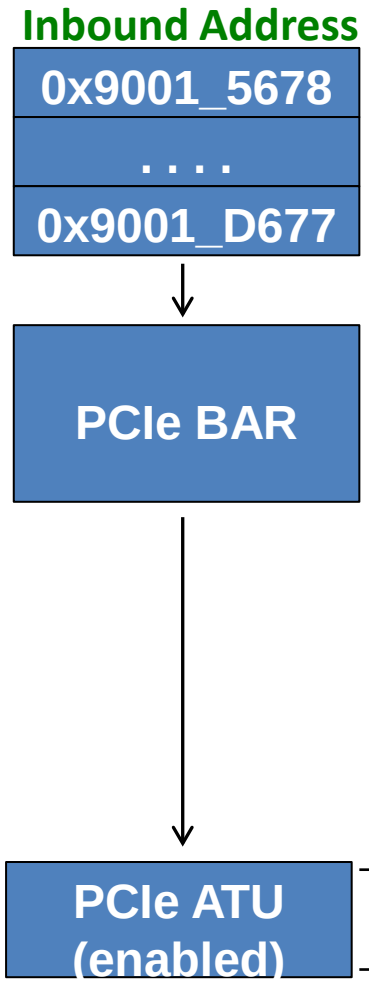
Registers for Inbound translation

- **BARn**:

  Two BARs (BAR0-1) in RC mode and six BARs (BAR0-5) in EP mode; overlay with BAR mask
- EP mode: If PCIe address is in the range configured in BAR, EP will accept packet and pass it to the internal side
- RC mode: If PCIe address is in range configured in BAR and outside range defined by three Base/Limit register sets, then that packet is accepted
- Four IB regions
  - **IB_BARn**: which BAR for inbound transaction
  - **IB_STARTn_LO**: the starting address bits [31:0] in PCIe address
  - **IB_STARTn_HI:** the starting address bits [63:32] in PCIe address
  - **IB_OFFSETn:** the internal bus address that will be the starting point of the mapped or translated PCIe address region
- BAR0 is dedicated to config. space starting at 0x21800000 (no remapping) This allows RC device to control EP in absence of dedicated software running on EP.

Address Translation: Inbound

EXAMPLE: Inbound write, destination buffer 32KB in L2



PCIe packet received with TLP header containing inbound addresses

To enable Inbound ATU, set CMD_STATUS[IB_XLT_EN] = 1

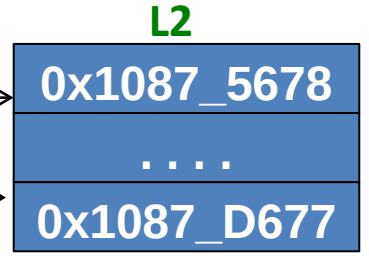
BAR1=0x9000_0000
BAR1 mask register= 0x00FF_FFFF → 16MB
Implies, BAR window is 0x9000_0000 to 0x90FF_FFFF
Therefore both 0x9001_5678 and 0x9001_D677 are accepted

IB_BAR0 = 1 → BAR1 is selected for IB Region 0
IB_START0_HI = 0x0 → upper 32 bits are zero for 32-bit addressing
IB_START0_LO= 0x9000_0000 ; IB_OFFSET0 = 0x1086_0000

Address translation logic is as follows:

0x9001_5678 – 0x9000_0000 + 0x1086_0000 = 0x1087_5678

0x9001_D677 – 0x9000_0000 + 0x1086_0000 = 0x1087_D677



Agenda

- PCIe Overview
- Keystone PCIe
- Address Translation
- **Configuration**
- MCSDK PCIe Demo

Configuration: PCIe LLD APIs

- **pcieSerdesCfg():** Configures PCIe SERDES PLL, transmitter and receiver
- **Pcie_setMode(PcieModeGbl):** Configures PCIe mode
- **pciePowerCfg():** Enables PCIe power & clock domain
- **pcieLtssmCtrl(handle, TRUE):** Enables PCIe link training
- **pcieWaitLinkUp(handle):** Wait for PCIe link up
- **Pcie_open(0, &handle):** Opening PCIe handle

Configuration: APIs for RC Mode

- **pcieCfgRC(handle):** Configures all required registers to operate in RC mode
- **Pcie_cfgBar(handle, &barCfg):** Configures required inbound BARs (Base Address Registers)
- **pcieIbTransCfg(handle, &ibCfg):** Configures inbound translation settings
- **pcieObTransCfg (handle, PCIE_OB_LO_ADDR_M, PCIE_OB_HI_ADDR_M, PCIE_OB_REGION_M):** Configures outbound translation settings

Configuration: APIs for EP Mode

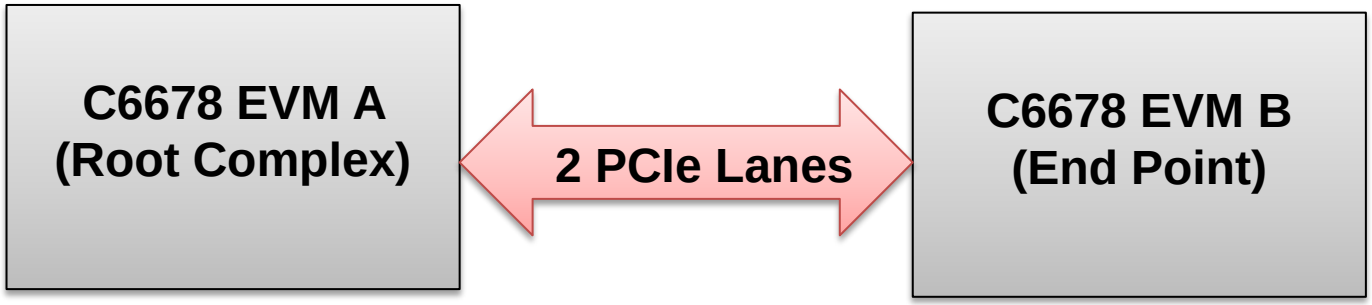
- **pcieCfgEP(handle)**
  Configures all required registers to operate in EP mode
- Following configuration steps can be initiated from EP device if software runs on it, or RC device through configuration read/writes if no software is running on the EP device
  - **pcie_cfgBar(handle, &barCfg)**
    Inbound BARs
  - **pcieIbTransCfg(handle, &ibCfg)**
    Inbound translation settings
  - **pcieObTransCfg (handle, PCIE_OB_LO_ADDR_M, PCIE_OB_HI_ADDR_M, PCIE_OB_REGION_M)**
    Outbound translation settings

Agenda

- PCIe Overview
- Keystone PCIe
- Address Translation
- Configuration
- **MCSDK PCIe Demo**

MCSDK PCIe Demo: Overview

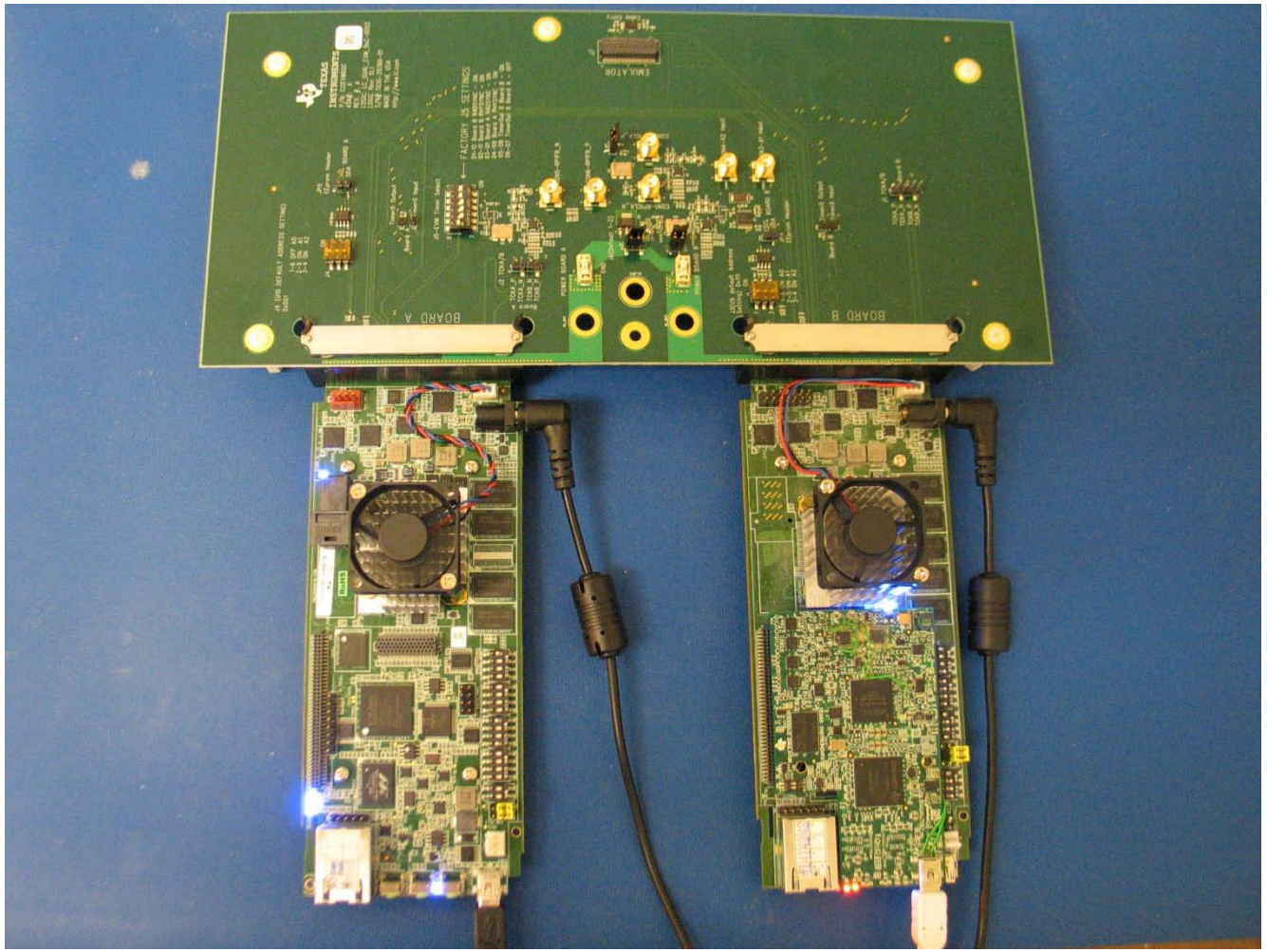
- PCIe demo in Multicore Software Development Kit
- Two C6678 EVMs connected over two PCIe lanes
- EVM A configured as Root-Complex (RC)
- EVM B configured as End-Point (EP)



- Location of demo in MCSDK:

  pdk_C6678_1_0_0_18\packages\ti\drv\exampleProjects\ PCIE_exampleProject

MCSDK PCIe Demo: Setup



MCSDK PCIe Demo: Flow

- Once PCIe link is established…
  - C6678 EVM A sends data to C6678 EVM B
  - C6678 EVM B waits to receive all the data
  - C6678 EVM B sends the data back to C6678 EVM A
  - C6678 EVM A waits to receive all the data
  - C6678 EVM A verifies if the received data matches the sent data and declares test pass or fail

- Value of PcieModeGbl (global variable) determines PCIe Mode (pcie_RC_MODE or pcie_EP_MODE)

For More Information

- For more information, refer to the PCI Express (PCIe) for KeyStone Devices User's Guide.
- For questions regarding topics covered in this training, visit the support forums at the TI E2E Community website.