Application Report
SPRABK8—December 2011

PCIe Use Cases for KeyStone Devices

High-Performance and Multicore Processors

Abstract

This document gives examples of PCIe usage in KeyStone devices including address translation, multi-device connection, and programming examples. It also contains detailed descriptions of PCIe features that supplement the information in the PCIe user's guide. See the KeyStone Architecture PCI Express User's Guide (SPRUGS6) for information about the PCIe registers and functions.

Contents

1 PCIe Features in KeyStone Devices 2
1.1 Dual Operation Mode 2
1.2 Link Rate and Lane Numbers 2
1.3 Outbound and Inbound Payload Size 2
2 PCIe Throughput in KeyStone Devices 3
2.1 Transaction Layer Packet (TLP) Configuration 3
2.2 Data Link Layer Packet (DLLP) Configuration 3
2.3 EDMA Considerations 3
2.4 Measured Throughput 4
3 PCIe Address Translation 5
3.1 PCIe Outbound Address Translation 6
3.1.1 PCIe Outbound Address Translation Examples 8
3.2 PCIe Inbound Address Translation 10
3.2.1 PCIe Base Address Register (BAR) 10
3.2.2 PCIe BAR Mask Register 11
3.2.3 PCIe Inbound Address Translation Examples 12
4 PCIe Multiple Devices Setup Example 14
5 PCIe Programming Example 15
6 Revision History 16

List of Tables

Table 1 PCIe Read Throughput Performance 4
Table 2 PCIe Write Throughput Performance 4
Table 3 BAR Register Field Descriptions 11

List of Figures

Figure 1 PCIe Transaction Layer Packet 3
Figure 2 PCIe Address Translation Modules 5
Figure 3 PCIe Outbound Address Translation 7
Figure 4 PCIe Inbound Address Translation 10
Figure 5 BAR Register Fields 11
Figure 6 PCIe Multiple Devices Setup Example 14

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this document.

1 PCIe Features in KeyStone Devices

The following sections provide supplementary information on important PCIe features that will help users achieve optimal results with their KeyStone device implementation.

1.1 Dual Operation Mode

The PCIe module in KeyStone devices supports both Root Complex (RC) and End Point (EP) operation modes. In EP mode, the PCIe module also supports both legacy EP mode and native PCIe EP mode. All three mode selections can be chosen from the bootstrap pins PCIESSMODE[1:0] at powerup (00->EP, 01->Legacy EP, 10->RC). The software can overwrite the setting by changing the PCIESSMODE bits in the DEVSTAT register (see the device-specific data manual). Only one mode (RC or EP) can be selected before the link training; this means that switching to another mode requires a reset of the PCIe module.

Note—Legacy EP supports legacy-style interrupt generation (INTx) using message requests but must also support MSI generation using memory write transactions. However, legacy devices are not required to support 64-bit memory addressing capability.

1.2 Link Rate and Lane Numbers

The KeyStone PCIe module supports both Gen1 (2.5 Gbps) and Gen2 (5.0 Gbps) link rates.

The PCIe module has a single port (link) with x2 lanes. This enables use of one lane (Lane 0) or two lanes (both Lane 0 and Lane 1, which doubles the throughput). When used together, both lanes must be configured with the same link data rate (2.5 Gbps or 5.0 Gbps).

1.3 Outbound and Inbound Payload Size

The KeyStone PCIe module supports an outbound payload size of 128 bytes and an inbound payload size of 256 bytes.

Outbound transfer means the local device initiates the transactions to write to or read from the external device. The CPU or the device-level EDMA is used for outbound data transfer. The PCIe module does not have built-in EDMA.

Inbound transfer means the external device initiates the transactions to write to or read from the local device. The PCIe module has a master port to transfer the data to or from the device memory; no CPU or EDMA is needed for inbound transfer in the local device.

Larger payload size means less packet overhead and more throughput. We can take advantage of the 256-byte payload size of inbound transfer as long as the external device also has 256 bytes (or more) of outbound payload size. For example, in the communication between two KeyStone devices via PCIe link, we cannot achieve a throughput payload size larger than 128 bytes because the maximum outbound payload size is only 128 bytes.

2 PCIe Throughput in KeyStone Devices

The PCI Express IP does not limit the throughput, but we must follow the PCIe protocol. The key protocol features are as follows:

- 8b/10b encoding is at the physical layer. This takes away 20 percent of the raw bandwidth.
- Acknowledge and flow control update at the data link layer using the data link layer packets (DLLPs).
- Packet overhead at the transaction layer. The payload size in each packet has a significant effect on throughput. Larger payload size reduces the overhead in the transaction layer packet (TLP) and increases the effective throughput.

Figure 1 shows an example of TLP in PCIe. All the parts of the packet other than **Data Payload** are overhead introduced by the PCIe protocol.

Figure 1 PCIe Transaction Layer Packet

2.1 Transaction Layer Packet (TLP) Configuration

The TLP header is 12 bytes for the 32-bit addressing mode and 16 bytes for the 64-bit addressing mode. ECRC is optional and can be disabled by programming the ECRC_GEN_EN field in PCIE_ACCR register.

Therefore, using 32-bit addressing with ECRC disabled reduces the overhead introduced by the transaction layer.

2.2 Data Link Layer Packet (DLLP) Configuration

DLLPs are used for link management functions including TLP acknowledgement associated with the ACK/NAK protocol, power management, and exchange flow control information. The ACK/NAK protocol and flow control improve reliability at the expense of throughput due to added overhead.

If not required by the application, DLLP flow control packets can be disabled by programming the FC_DISABLE field in the LANE_SKEW register. If DLLP acknowledge packets are not required by the application, transmission of these packets can be disabled by programming the ACK_DISABLE field in the LANE_SKEW register. If acknowledge packets are required by the application, the frequency of the acknowledge packets can be configured to the rate required by the application by programming the ACK_FREQ register.

2.3 EDMA Considerations

As mentioned above, the maximum payload size of KeyStone PCIe module is 128 bytes for outbound transfer and 256 bytes for inbound transfer.

If using EDMA for the PCIe outbound transfer, the data payload in the TLP is equal to the data burst size (DBS) of the EDMA transfer controller (TC) if the DBS is less than or equal to the maximum PCIe payload size.

In KeyStone devices, not all EDMA TCs use the same data burst size and it can be 64 bytes or 128 bytes (see the device-specific data manual for the information about EDMA configuration). For example, an outbound transfer of 1KB of data via PCIe using an EDMA TC with a 128-byte DBS generates 1KB/128B=8 packets; using the EDMA TC with a 64-byte DBS generates 1KB/64B=16 packets, which introduces more overhead.

Therefore, better performance is achieved for outbound transfer using the EDMA transfer controller with a larger data burst size.

2.4 Measured Throughput

We measure the PCIe throughput with two KeyStone devices set up—one C6678 as RC and another C6678 as EP—for two lanes, Gen2 mode. The transfer is uses 32-bit addressing with ECRC disabled. DLLP configuration is by default (with acknowledge and flow control enabled). RC initializes the read and write transactions using EDMA (both 64-byte and 128-byte DBS has been tested). The throughput has been tested across multiple memory endpoints: L2, MSMC SRAM, and DDR. The data buffer size ranges from 2KB to 48KB.

The PCIe read and write measured results are shown in Table 1 and Table 2 as follows:

Table 1 PCIe Read Throughput Performance

Data Burst Size	Throughput (Gbps)	Throughput (MBps)
128 bytes	6.45	806.25
64 bytes	5.51	688.75
End of Table 1-1		

Table 2 PCIe Write Throughput Performance

Data Burst Size	Throughput (Gbps)	Throughput (MBps)
128 bytes	5.91	738.75
64 bytes	5.55	693.75
End of Table 1-2		

The throughput results are very similar for transactions between different memory endpoints. So only the throughput from Device1 DDR3 memory to Device2 L2 memory has been shown in the tables above as an example.

The read performance is slightly better than write performance in the 128-byte EDMA DBS scenario while the read and write performances are similar in 64-byte DBS case.

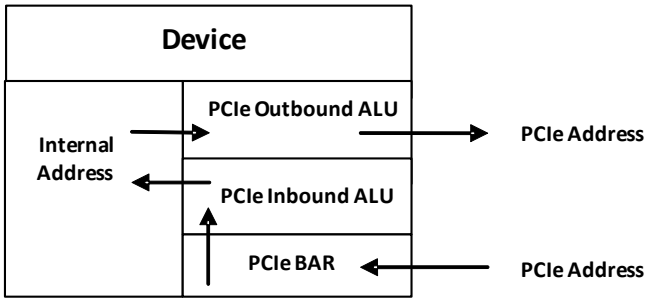
3 PCIe Address Translation

PCIe devices need to use PCIe addresses to send/receive packets over a PCIe link. The Address Translation Unit (ATU) within the PCIe module translates the device internal address into a PCIe address and vice versa. The PCIe address could be 32-bit or 64-bit (legacy EP may not support 64-bit addresses).

For the outbound transaction, the outbound ATU translates the device internal address into a PCIe address. The data with a PCIe address is transferred over the PCIe link to the other device.

For the inbound transaction, the Base Address Register (BAR) in the PCIe module accepts certain PCIe addresses and rejects the others. The data with an accepted PCIe address goes through the inbound ATU and is transferred to the device internal memory after address translation.

Figure 2 PCIe Address Translation Modules



Both of the Outbound and Inbound translation modules can be enabled or disabled.

For Outbound transfer, we can use EDMA to move the device memory data to the PCIe data space (0x6000_0000~0x6FFF_FFFF in KeyStone devices) so that it can be sent to the PCIe link.

If the Outbound translation is disabled (CMD_STATUS[OB_XLT_EN] = 0), the PCIe address would be the PCIe data space address, which is being used as the destination address in EDMA transfer. For example, if we transfer L2 memory at 0x1084_4800 to PCIe data space 0x6000_1234, the PCIe address will be 0x6000_1234 (without being translated since Outbound address translation is disabled).

If Outbound translation is enabled (CMD_STATUS[OB_XLT_EN] = 1), the outbound PCIe address (0x6000_0000~0x6FFF_FFFF) can be modified to a new address based on the Outbound translation rules.

The Inbound translation module can also be disabled (CMD_STATUS[IB_XLT_EN] = 0). If the incoming address is already mapped to device internal memory, there is no need to modify it.

However, this is probably not the case. So the Inbound translation module should be enabled (CMD_STATUS[IB_XLT_EN] = 1) to redirect the incoming PCIe address to some new address that is mapped to internal memory regions.

3.1 PCIe Outbound Address Translation

The following set of registers are used for outbound address translation if outbound translation is enabled:

- OB_SIZE: identify the size of 32 equally-sized translation regions to be 1 MB/2 MB/4 MB/8 MB
- OB_OFFSET_INDEXn: represent bits[31:20] of the 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_OFFSETn_HI: represent bits[63:32] of the PCIe address for 64-bit addressing; must be zero for 32-bit addressing

Note—OB_OFFSET is the PCIe address, which replaces the PCIe data space address (0x6000_0000~0x6FFF_FFFF) in the KeyStone device.

There are 32 translation regions, which need five bits (32=2^5) to determine the mapping of each region.

- If OB_SIZE = 1 MB: 0x10_0000 = 0001 0000_0000 0000 0000 0000b
  Bits[24:20] of internal address are used for region identification
  Bits[31:20] of OB_OFFSET_INDEXn are used for translation
  In KeyStone devices, the region allocation is as follows:
  - 0x6000_0000~0x600F_FFFF is for Region 0 (bits[24:20]=0)
    0x6010_0000~0x601F_FFFF is for Region 1 (bits[24:20]=1)
    .....
    0x61F0_0000~0x61FF_FFFF is for Region 31 (bits[24:20]=0x1F=31)
    0x6200_0000~0x620F_FFFF is for Region 0 (bits[24:20]=0)
    .....
    0x6FF0_0000~0x6FFF_FFFF is for Region 31 (bits[24:20]=0x1F=31)
- If OB_SIZE = 2 MB: 0x20_0000 (0010 0000_0000 0000 0000 0000)
  Bits[25:21] of internal address are used for region identification
  Bits[31:21] of OB_OFFSET_INDEXn are used for translation
  In KeyStone devices, the region allocation is as follows:
  - 0x6000_0000~0x601F_FFFF is for Region 0 (bits[25:21]=0)
    0x6020_0000~0x603F_FFFF is for Region 1 (bits[25:21]=1)
    .....
    0x63E0_0000~0x63FF_FFFF is for Region 31 (bits[25:21]=0x1F=31)
    0x6400_0000~0x641F_FFFF is for Region 0 (bits[25:21]=0)
    .....
    0x6FE0_0000~0x6FFF_FFFF is for Region 31 (bits[25:21]=0x1F=31)
- If OB_SIZE = 4 MB: 0x40_0000 (0100 0000_0000 0000 0000 0000)
  Bits[26:22] of internal address are used for region identification
  Bits[31:22] of OB_OFFSET_INDEXn are used for translation
  In KeyStone devices, the region allocation is as follows:
  - 0x6000_0000~0x603F_FFFF is for Region 0 (bits[26:22]=0)

0x6040_0000~0x607F_FFFF is for Region 1 (bits[26:22]=1)

.....

0x67C0_0000~0x67FF_FFFF is for Region 31 (bits[26:22]=0x1F=31)

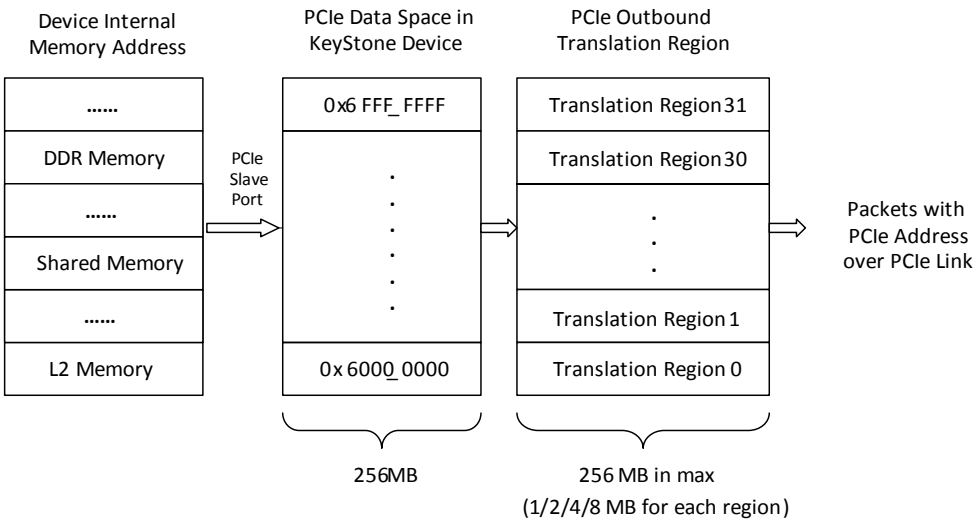
0x6800_0000~0x683F_FFFF is for Region 0 (bits[26:22]=0)

.....

0x6FC0_0000~0x6FFF_FFFF is for Region 31 (bits[26:22]=0x1F=31)

- If OB_SIZE = 8 MB: 0x80_0000 (1000 0000_0000 0000 0000 0000)
  Bits[27:23] of internal address are used for region identification
  Bits[31:23] of OB_OFFSET_INDEXn are used for translation
  In KeyStone devices, the region allocation is as follows:
  – 0x6000_0000~0x607F_FFFF is for Region 0 (bits[27:23]=0)
    0x6080_0000~0x60FF_FFFF is for Region 1 (bits[27:23]=1)
    .....
    0x6F00_0000~0x6F7F_FFFF is for Region 30 (bits[27:23]=0x1E=30)
    0x6F80_0000~0x6FFF_FFFF is for Region 31 (bits[27:23]=0x1F=31)

Figure 3 PCIe Outbound Address Translation



3.1.1 PCIe Outbound Address Translation Examples

The following examples demonstrate how to translate a KeyStone device memory address to a PCIe address for outbound transfer.

Example 1 Outbound Write, 32KB Source Buffer in L2, 32-bit Addressing

Assume the L2 source buffer is located at: 0x1084_4800~0x1084_C7FF (32KB)

CMD_STATUS[OB_XLT_EN] = 1 (outbound address translation enabled)
Assume OB_SIZE=1 MB (bits [24:20] decide which translation region being used)

We can use EDMA to do the transfer. Choose any PCIe data space address as destination, but pay attention to the address boundary alignment:
dataSize = 32KB;
srcAddr = 0x1084_4800;
dstAddr = 0x6001_5678;

Bits [24:20] of 0x6001_5678 = 00000b = 0, so Translation Region 0 is used for translation.
OB_OFFSET_INDEX0 = 0x9000_0001 (bit [0]=1 enables this region)
(0x9000_0000 is the PCIe address, which can be chosen randomly)
OB_OFFSET0_HI = 0x0 (upper 32 bits are zero for 32-bit addressing)

Then the translated address = bits[31:20] of 0x9000_0000 + bits[19:0] of 0x6001_5678 = 0x9001_5678
So the internal L2 addresses 0x1084_4800~0x1084_C7FF are translated into 0x9001_5678~0x9001_D677 (32KB) as the PCIe address, which could be accepted by the other PCIe device over the link whose BAR window covers this range 0x9001_5678~0x9001_D677.

End of Example 1

Example 2 Outbound Read, 12 MB Destination Buffer in DDR, 64-bit Addressing

Assume the DDR destination buffer is located at: 0x8912_5678~0x89D2_5677 (12 MB)

CMD_STATUS[OB_XLT_EN] = 1 (outbound address translation enabled)
Assume OB_SIZE=8 MB (bits [27:23] decide which translation region being used)

Because buffer size (12 MB) is larger than OB_SIZE (8 MB), we must choose two contiguous translation regions (2*8 MB=16 MB>12 MB) for the transfer. But there is no limit for which two regions are chosen.

For example, if we choose Region 18 and Region 19 for the translation.
Bits [27:23] = 18 = 10010b --> 0x6900_0000;
Bits [27:23] = 19 = 10011b --> 0x6980_0000;

OB_OFFSET_INDEX18 = 0x7000_0001 (bit [0]=1 enables this region)
(0x7000_0000 is the PCIe address, which can be chosen randomly)
OB_OFFSET18_HI = 0x1234_5678 (64-bit addressing)

OB_OFFSET_INDEX19 = 0x7080_0001
(make Region 19 offset consistent as Region 18, with 8 MB spacing)
OB_OFFSET19_HI = 0x1234_5678

We can use EDMA to do the transfer.
srcAddr = 0x6900_0000;
dstAddr = 0x8912_5678;
dataSize = 12 MB;

The translated address = bits[63:23] of 0x12345678_70000000 + bits[22:0] of 0x6900_0000 = 0x12345678_70000000

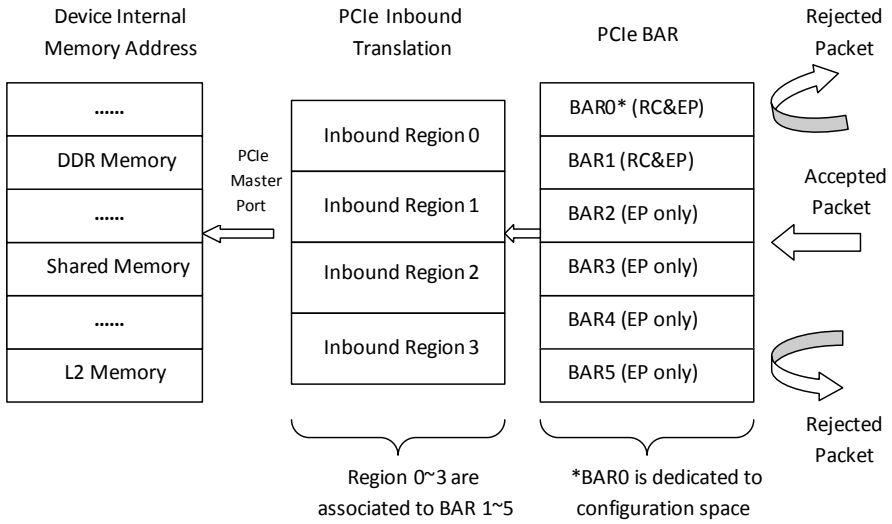
Then the data packets with PCIe address 0x12345678_70000000~0x12345678_70BFFFFF (12 MB) will be transferred to the internal DDR location as 0x8912_5678~0x89D2_5677. The other PCIe device over the link whose BAR window (64-bit) including this PCIe address range will be the target to be read from.

End of Example 2

3.2 PCIe Inbound Address Translation

Both BARs (Base Address Registers) and inbound region registers must be programmed correctly for the PCIe inbound address translation.

Figure 4 PCIe Inbound Address Translation



3.2.1 PCIe Base Address Register (BAR)

There are two BARs (BAR0~1) in RC mode and six BARs (BAR0~5) in EP mode.

In EP mode, BAR0 is dedicated to a configuration space (application registers) of 32-bit addressing. BAR0 and BAR1 are dedicated to a configuration space of 64-bit addressing.

In EP mode, if the PCIe address is in the range configured in the BAR, the EP accepts the packet and passes it to the internal side. If the address is outside the BAR, the packet is rejected.

In RC mode, there are another three sets of registers to define the range of rejection.

- Memory Space (MEM_BASE, MEM_LIMIT)
- Prefetchable Memory Space (PREFETCH_MEM,PREFETCH_BASE,PREFETCH_LIMIT)
- IO Space (IO_BASE, IO_LIMIT)

BASE defines the start address of the range; LIMIT defines the end address of the range. The range defined by BASE/LIMIT should cover all the regions used by EPs connected to the RC. Any packets within the range defined by those registers are misrouted and will be rejected by RC. Only packets outside the range will be accepted by RC.

Figure 5 BAR Register Fields

31	4	3	2	1	0
BASE_ADDR		PREFETCH	TYPE		MEM_SPACE
R/W-0		R-0	R-0		R-0

Legend: R = Read only; W = Write only; R/W = Read/Write; -*n* = value after reset

Table 3 BAR Register Field Descriptions

Bit	Field	Description
31-4	BASE_ADDR	Base Address. Some bits may be masked (not writeable) by the BAR Mask register.
3	PREFETCH	0 = Non-prefetchable means a read will change the contents, so the original contents will not be available again later. 1 = Prefetchable means reading a location does not affect the contents, so it can be prefetched, discarded, and read again later with no adverse effects.
2-1	TYPE	0 = 32-bit BAR (BAR0~BAR5 can be used as six separate 32-bit BARs) 2 = 64-bit BAR (BAR0 and 1, BAR2 and 3, BAR4 and 5 are used as three 64-bit BARs)
0	MEM_SPACE	0 = Memory BAR 1 = I/0 BAR, used for I/O transaction which is not required by the PCIe device; used only for PCI back-compatibility.
End of Table 3		

3.2.1.1 PCIe BAR0 in KeyStone Devices

BAR0 cannot be remapped to any other location than to application registers (starting from 0x2180_0000 in KeyStone device). It allows the RC device to control EP in the absence of dedicated software running on EP.

For example, if the RC outbound region 0 translation address is set as 0x7000_0000, EP BAR0 is also 0x7000_0000. RC writes 0x1 to 0x6000_0300 (translated to 0x7000_0300), then the IB_BAR0 in EP device will be 0x1 because 0x7000_0000 matches the starting address of the EP application registers and 0x300 is the offset of the IB_BAR0 register.

There is another shortcut for RC to set up EP configuration registers if RC is directly connected to a single EP between KeyStone devices. RC writes to its own local MMR PCIe register base address + 0x2000 (0x2180_2000 in KeyStone device) is equal to writing to the configuration register (starting from 0x2180_1000 in KeyStone device) of the remote EP.

3.2.2 PCIe BAR Mask Register

BAR Mask Registers are overlaid on the BAR registers (sharing the same address offset of BAR). BAR Mask registers are accessible to configure BARs only when operating as EP and only when DBI_CS2 (bit 5 in CMD_STATUS register) is enabled.

BAR Mask registers are writeable but NOT readable. The return of read of memory window shows only the BAR registers values, not the values of BAR Mask registers.

BAR Mask register can set the Base Address bits [31:1] (bit[0] will enable/disable the corresponding BAR).

One way to verify if the BAR Mask registers have been set correctly is to write the pattern values into BAR registers and see if the bits have been masked correctly.

For example, enable DBI_CS2, write BAR0 MASK register = 0x0000_3FFF, disable DBI_CS2, write BAR0 = 0xFFFF_FFF0; BAR0 should be read as 0xFFFF_C000 because the lower bits of BAR0 have been masked.

In another example, BAR1 MASK register = 0x0007_FFFF, write BAR1 = 0xFFFF_FFF0; BAR1 should be read as 0xFFF8_0000.

See the KeyStone PCIe user's guide for BAR/BAR Mask registers setup examples.

3.2.3 PCIe Inbound Address Translation Examples

The following examples demonstrate how to translate a KeyStone device memory address to a PCIe address for inbound transfer.

Example 3 Inbound Write, 32KB Destination Buffer in L2, 32-bit Addressing

CMD_STATUS[IB_XLT_EN] = 1 (inbound address translation enabled)

The target packets with PCIe address 0x9001_5678~0x9001_D677 (32KB) (Based on the previous outbound address translation in Example 1 on page 8).

BAR1 = 0x9000_0000
(bits [3:0]=0000b, means 32-bit, non-prefetchable, memory BAR).
BAR1 Mask register is set as 0x00FF_FFFF, so BAR1 window size is 16 MB.
BAR1 accepts this inbound write request since the window covers the PCIe address of the target packets.

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

Then the starting device memory address
= PCIe address - (IB_START0_HI:IB_START0_LO) + IB_OFFSET
= (0x9001_5678) - (0x9000_0000) + (0x1086_0000) = 0x1087_5678

So the data packets with PCIe address 0x9001_5678~0x9001_D677 are transferred to the destination buffer as 0x1087_5678~0x1087_D677 by the PCIe master port (no EDMA or CPU is involved in local device for the inbound transfer).

End of Example 3

Example 4 Inbound Read, 12 MB Source Buffer in DDR, 64-bit Addressing

CMD_STATUS[IB_XLT_EN] = 1 (inbound address translation enabled)

The target packets with PCIe address 0x12345678_70000000~0x12345678_70BFFFFF (12 MB) (Based on the previous outbound address translation in Example 2 on page 9).

BAR2 = 0x7000_000C (bits [3:0]=1100b, means 64-bit, prefetchable, memory BAR).
BAR3 = 0x1234_5678 (BAR3 Mask register is 0x0, non-masked)
BAR2 Mask register is set as 0x0FFF_FFFF, so BAR2&3 window size is 256 MB.
BAR2&3 accept this inbound read request because the window covers the PCIe address of the target packets.

IB_BAR1 = 2 (BAR2&3 are selected for IB Region 1 for 64-bit addressing)
IB_START1_HI = 0x1234_5678 (same as upper 32 bits of BAR2&3)
IB_START1_LO= 0x7000_0000
IB_OFFSET1 = 0x8700_0000

Then the starting internal device address
= PCIe address - (IB_START0_HI:IB_START0_LO) + IB_OFFSET
= (0x12345678_70000000) - (0x12345678_70000000) + (0x8700_0000)
= 0x8700_0000

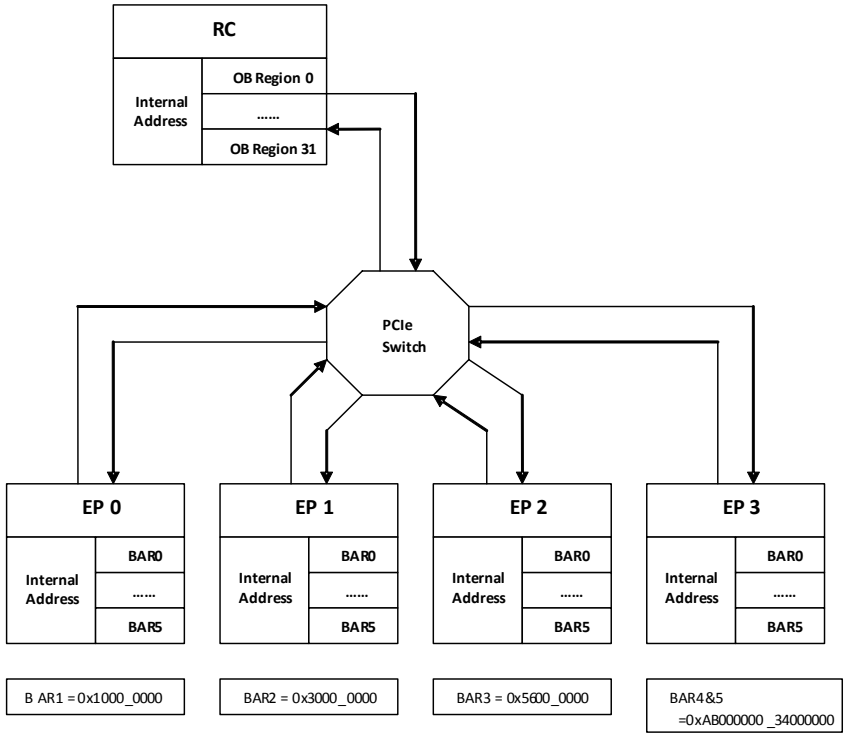
So the data located in the buffer as 0x8700_0000~0x87BF_FFFF are fetched by the PCIe master port with PCIe address 0x12345678_70000000~0x12345678_70BFFFFF (no EDMA or CPU is involved in local device for the inbound transfer).

End of Example 4

4 PCIe Multiple Devices Setup Example

The PCIe switch can be used to connect multiple PCIe devices together as one system. One PCIe device can be set as RC to configure the other EPs. For example, the BARs of EPs are configured with the values shown in Figure 6.

Figure 6 PCIe Multiple Devices Setup Example



For example, the outbound regions of the RC are programmed as follows:

- OB Region 0 translates to PCIe address starting from 0x1000_0000, which is targeted to EP0 BAR1
- OB Region 1&2 translate to PCIe address starting from 0x3000_0000, which are targeted to EP1 BAR2
- OB Region 3 translates to PCIe address starting from 0x5600_0000, which is targeted to EP2 BAR3
- OB Region 4~11 translate to PCIe address starting from 0xAB000000_34000000, which are targeted to EP3 BAR4&5.

Then we can map the any memory buffers in the RC device to any of the regions for PCIe transaction.

The RC rejection regions should also be programmed to cover all the EP BARs ranges. In this example, we program the following:

- MEM_BASE = 0x1000_0000
- MEM_LIMIT = 0x7FFF_FFFF
- PREFTECH_MEM_BASE = 0x2000_0000
- PREFETCH_MEM_LIMIT = 0x9FFF_FFFF
- PREFETCH_BASE = 0xA000_0000 (upper 32-bit in 64-bit addressing)
- PREFETCH_LIMIT = 0xAFFF_FFFF (upper 32-bit in 64-bit addressing)

5 PCIe Programming Example

This section includes the PCIe programming example for both RC mode and EP mode. The two examples can be used on two devices to complete the write transaction from RC to EP.

Example 5 Programming Example for RC Mode

```
/* Initialization sequence is as follows */
/* Suppose PCIe module is disabled (PCIESSEN=0) */
/* Enable power/clock domain of PCIe module */
enable_module (pciex_pdctl, pciex_mdctl); //please refer to PSC user's guide
/* Set PCIe operation mode as RC */
/* First unlock boot configuration by setup KICK0&1 if locked before */
DEVSTAT[PCIESSMODE] = 0x10;
/* Program and enable SerDes PLL based on reference clock */
PCIE_SERDES_CFGPLL = 0x1C9; //reference clock assumed as 100MHz
/* Wait for PLL lockup */
While (PCIE_SERDES_STS[LOCK]  !=1);
/* Disable link training */
CMD_STATUS[LTSSM_EN] = 0;
/* Setup configuration registers */
switch (lane_num) {
    Case 1: //single lane
      PL_LINK_CTRL[LINK_MODE] = 0x1; //enable x1 lane
    Case 2: //two lanes
      PL_LINK_CTRL[LINK_MODE] = 0x3; //enable x2 lanes
}
switch (data_rate) {
    case (Gen1):
      PL_GEN2[DIR_SPD] = 0x0; //stay in Gen1 after linkup
    case (Gen2):
      PL_GEN2[DIR_SPD] = 0x1; //change to Gen2 after linkup
}
CMD_STATUS[DBI_CS2] = 1; //enable DBI_CS2 to unlock writing to BAR mask registers
BAR0 = 0x0FFFFFFF; //BAR0 Mask register sets BAR0 window size=256MB
BAR1 = 0x0FFFFFFF; //BAR1 Mask register sets BAR1 window size=256MB
CMD_STATUS[DBI_CS2] = 0; //disable DBI_CS2 to lock writing to BAR mask registers
STATUS_COMMAND | = 0x146; //enable memory access; mastership of bus enabled
                          //detect poisoned TLP; enable error message generation
DEV_STAT_CTRL | = 0xF; //enable UR, fatal, non-fatal and correctable error
PCIE_ACCR | = 0x1E0; //enable ECRC check and generation
/* Setup Outbound registers */
OB_SIZE = 1MB;
OB_OFFSET_INDEX0 = 0x70000001;  //bit[0]=1 means the region is enabled
OB_OFFSET_INDEX0_HI = 0x0;
CMD_STATUS[OB_XLT_EN] = 1; //enable outbound address translation
/* Enable link training */
CMD_STATUS[LTSSM_EN] = 0x1;
/* Wait for link up */
while (DEBUG0[4:0] != 0x11)//wait for L0 state (link is ready)

/* Configure EP registers */
/* EP needs to set BAR Mask registers by itself to ensure BAR window is large enough*/
PCIE_REMOTE_CFG_BAR1 = 0x70000000; //EP->BAR covers RC->OB_OFFSET

/* Define payload size and buffer size */
pcie_max_payload = 128; //assume pcie maximum payload size is 128 bytes
buff_size= 2048; // assume the transfer buffer size 2048 bytes
/* Define PCIe data starting address for EDMA transfer */
/* PCIe data starting address defined in device-specific data manual */
pcieAddr = 0x60000000; // PCIe data space in KeyStone device

/* Setup EDMA module and enable the DMA region */
/* Setup EDMA PARAM */
/* The OPT register contains following bit fields:
* ITCCHEN = 0x0; TCCHEN = 0x0; ITCINTEN = 0x0; TCINTEN = 0x1;
* TCC = 0x0; TCCMODE = 0x0; FWID = 0x0; STATIC = 0x0; SYNCDIM = 0x1;
* DAM = 0x0; SAM = 0x0 */
paramSetup_pcie.option = CSL_EDMA3_OPT_MAKE(0,0,0,1,0,0,0,0,1,0,0);
paramSetup_pcie.aCntbCnt =
CSL_EDMA3_CNT_MAKE(pcie_max_payload,buff_size/pcie_max_payload);
paramSetup_pcie.srcDstBidx =
CSL_EDMA3_BIDX_MAKE(pcie_max_payload,pcie_max_payload);
paramSetup_pcie.srcDstCidx = CSL_EDMA3_CIDX_MAKE(0,0);
paramSetup_pcie.cCnt = 1;
paramSetup_pcie.linkBcntrld =
CSL_EDMA3_LINKBCNTRLD_MAKE(CSL_EDMA3_LINK_NULL,0);
paramSetup_pcie.srcAddr = (Uint32)srcAddr;
```

```
paramSetup_pcie.dstAddr = (Uint32)pcieAddr;
/* Setup EDMA Channel */
/* Enable the EDMA Channel */
/* Wait for a transmit completion */
```

End of Example 5

Example 6 Programming Example for EP Mode

```
/* Suppose PCIe module is disabled (PCIESSEN=0) */
/* Enable power/clock domain of PCIe module*/
 enable_module (pciex_pdctl, pciex_mdctl); //refer to PSC user's guide
/* Set PCIe operation mode as EP */
/* First unlock boot configuration by setup KICK0&1 if locked before */
DEVSTAT[PCIESSMODE] = 0x0;
/* Program and enable SerDes PLL based on reference clock */
PCIE_SERDES_CFGPLL = 0x1C9; //reference clock assumed as 100MHz
/* Wait for PLL lockup */
While (PCIE_SERDES_STS[LOCK]  !=1);

/* Disable link training */
CMD_STATUS[LTSSM_EN] = 0;
/* Setup EP configuration registers */
switch (lane_num) {
  <same as RC configuration>
}
switch (data_rate) {
  <same as RC configuration>
}
CMD_STATUS[DBI_CS2] = 1; //enable DBI_CS2 to unlock writing to BAR mask registers
BAR0 = 0x000FFFFF; //BAR0 Mask register sets BAR0 window size=1MB
BAR1 = 0x0FFFFFFF; //BAR1 Mask register sets BAR1 window size=256MB
BAR2 = 0x007FFFFF; //BAR2 Mask register sets BAR2 window size=8MB
BAR3 = 0x03FFFFFF; //BAR3 Mask register sets BAR3 window size=64MB
BAR4 = 0x0FFFFFFF; //BAR4 Mask register sets BAR4 window size=256MB
BAR5 = 0x0; //BAR5 is not masked, may be used as upper 32bits of BAR4
            //(64-bit addressing)
CMD_STATUS[DBI_CS2] = 0; //disable DBI_CS2 to lock writing to BAR mask registers
STATUS_COMMAND | = 0x146; //enable memory access; mastership of bus enabled
                          //detect poisoned TLP; enable error message generation
DEV_STAT_CTRL | = 0xF; //enable UR, fatal, non-fatal and correctable error
PCIE_ACCR | = 0x1E0; //enable ECRC check and generation
/* Setup Inbound registers */
IB_BAR0 = 1; //BAR1 is selected for inbound region 0
IB_START0_LO = 0x70000000;
IB_START0_HI = 0x0;
IB_OFFSET0 = 0x82000000;
CMD_STATUS[IB_XLT_EN] = 1 //enable Inbound address translation

/* May setup BARn values by EP itself or wait for RC to configure BARn after linkup */
BAR1 = 0x70000000;

/* Enable link training */
CMD_STATUS[LTSSM_EN] = 0x1;
/* Wait for link up */
while (DEBUG0[4:0] != 0x11)//wait for L0 state (link is ready)
/* Ready for PCIe transactions */
```

End of Example 6

6 Revision History

Revision	Date	Description of Changes
SPRABK8	December 2011	Initial Release

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2011, Texas Instruments Incorporated