PCI Express

From OSDev Wiki

The PCI Express bus is a backwards compatible, high performance, general purpose I/O interconnect bus, and was designed for a range of computing platforms. One of the key improvements of PCI Express, over the PCI Local Bus, is that it now uses a serial interface (compared to the parallel interface used by PCI). This improvement can be compared to the similiar serialization of the ATA interface.

This page is a work in progress and may thus be incomplete. Its content may be changed in the near future.

Contents

PCI Express Link

The PCI Express bus connects each device directly to the CPU and other system devices through a pair of high speed unidirectional differential links (transmit and recieve, respectively). These links operate at an effective rate of 2.5 GB/s and a single device may have multiple links. A single device may have x1, x2, x4, x8, x12, x16, or x32 links and can achieve a maximum bandwidth of 80 GB/s by utilizing x32 links.

Initialization

Extended Configuration Space

The PCI Express bus extends the Configuration Space from 256 bytes to 4096 bytes. This extended configuration space *cannot* be accessed using the legacy PCI method (through ports 0xCF8 and 0xCFC). Instead, an #Enhanced Configuration Mechanism is provided.

Changes from the PCI Configuration Space

Header Type	Register (Offset)	Bit Location	Difference
All Headers	Command Register (0x04)	3	Special Cycle Enable: Does not apply to PCIe. Hardwired to 0.
		4	Memory Write and Invalidate: Does not apply to PCIe. Hardwired to 0.
		5	VGA Palette Snoop: Does not apply to PCIe. Hardwired to 0.
		7	IDSEL Stepping/Wait Cycle Control: Does not apply to PCIe. Hardwired to 0.
		9	Fast Back-to-Back Transactions Enable: Does not apply to PCIe. Hardwired to 0.
	Status Register (0x06)	4	Capabilities List: All PCIe devices are required to implement the capability structure. Hardwired to 1.
		5	66 MHz Capable: Does not apply to PCIe. Hardwired to 0.
		6	Fast Back-to-Back Transactions Capable: Does not apply to PCIe. Hardwired to 0.
		10:9	DEVSEL Timing: Does not apply to PCIe. Hardwired to 0.
	Cache Line Size Register (0x0C)	All Bits	Implemented for legacy purposes only.
	Master Latency Timer Register (0x0D)	All Bits	Does not apply to PCIe. Hardwired to 0.
Type 0	Base Address Registers (0x10:0x24)	All Bits	PCIe Endpoint devices must set the BAR's prefetchable bit while the range does not contain memory with read side-effects or where the memory does not tolerate write merging. 64-Bit Addressing MUST be supported by non legacy Endpoint devices. The minimum memory address range requested by a BAR 128-bytes.
	Min_Gnt/Max_Lat Registers (0x3E:0x3F)	All Bits	Does not apply to PCIe. Hardwired to 0.
	Base Address Registers (0x10:0x24)	All Bits	PCIe Endpoint devices must set the BAR's prefetchable bit while the range does not contain memory with read side-effects or where the memory does not tolerate write merging. 64-Bit Addressing MUST be supported by non legacy Endpoint devices. The minimum memory address range requested by a BAR 128-bytes.
	Primary Bus Number (0x18)	All Bits	Implemented for legacy purposes only.

Type 1	Secondary Latency Timer (0x1B)	All Bits	Does not apply to PCIe. Hardwired to 0.
	Secondary Status Register (0x1E)	5	66 MHz Capable: Does not apply to PCIe. Hardwired to 0.
		7	Fast Back-to-Back Transactions Capable: Does not apply to PCIe. Hardwired to 0.
		10:9	DEVSEL Timing: Does not apply to PCIe. Hardwired to 0.
	Prefetchable Memory Base/Limit (0x24)	All Bits	Must indicate support for 64-bit addresses.
	Bridge Control Register (0x3E)	5	Master Abort Mode: Does not apply to PCIe. Hardwired to 0.
		7	Fast Back-to-Back Transactions Enable: Does not apply to PCIe. Hardwired to 0.
		8	Primary Discard Timer: Does not apply to PCIe. Hardwired to 0.
		9	Secondary Discard Timer: Does not apply to PCIe. Hardwired to 0.
		10	Discard Timer Status: Does not apply to PCIe. Hardwired to 0.
		11	Discard Timer SERR# Enable: Does not apply to PCIe. Hardwired to 0.

Enhanced Configuration Mechanism

The enhanced configuration mechanism makes use of a flat memory-mapped address space to access it's device configuration registers. Put simply, the memory address is used to determine the register accessed. There is an area for each host controller, so a system with 2 PCI express host controllers uses 2 memory mapped areas. On x86 and x64 platforms, the address of each memory area is determined by the ACPI 'MCFG' table. The format of this ACPI table is:

Offset	Length	Description
0	4	Table Signature ("MCFG")
4	4	Length of table (in bytes)
8	1	Revision (1)
9	1	Checksum (sum of all bytes in table & 0xFF = 0)
10	6	OEM ID (same meaning as other ACPI tables)
16	8	OEM table ID (manufacturer model ID)
24	4	OEM Revision (same meaning as other ACPI tables)
28	4	Creator ID (same meaning as other ACPI tables)
32	4	Creator Revision (same meaning as other ACPI tables)
36	8	Reserved
		Configuration space base address allocation structures. Each structure uses the

44	16 * n	following format:

Offset	Length	Description
0	8	Base address of enhanced configuration mechanism
8	2	PCI Segment Group Number
10	1	Start PCI bus number decoded by this host bridge
11	1	End PCI bus number decoded by this host bridge
12	4	Reserved

After determining the MMIO base address and the total number of busses in the address space, you can read from the extended configuration address space. To access a specific register, you must use the following formula: Address = MMIO_BASE + { bus number[27:20], device number[19:15], function number[14:12], extended register number[11:8], register number[7:2], offset [1:0] }.

```
readECS_BYTE:
    mov al, [esi]           ; Read uint8_t from MMIO Address [ESI] into AL
    ret                     ; Return to the calling code

readECS_WORD:
    mov ax, [esi]           ; Read uint16_t from MMIO Address [ESI] into A
    ret                     ; Return to the calling code

readECS_DWORD:
    mov eax, [esi]          ; Read uint32_t from MMIO Address [ESI] into E
    ret                     ; Return to the calling code

writeECS_BYTE:
    mov [esi], al           ; Write uint8_t from AL into MMIO Address [ESI
    ret                     ; Return to the calling code

writeECS_WORD:
    mov [esi], ax           ; Write uint16_t from AX into MMIO Address [ES
    ret                     ; Return to the calling code

writeECS_DWORD:
    mov [esi], eax          ; Write uint32_t from EAX into MMIO Address [E
    ret                     ; Return to the calling code
```

System Architecture

Transaction Layer

Data Link Layer

Physical Layer

Power Management

See Also

References

- PCI Express Base Specification, revision 1.1, PCI Special Interest Group, March 28, 2005

External Links

- http://lmgtfy.com/?q=pci+express+base+specification+3.0

Retrieved from "http://wiki.osdev.org/index.php?title=PCI_Express&oldid=17437"
Categories: In Progress | Buses

- This page was last modified on 1 January 2015, at 14:02.
- This page has been accessed 31,612 times.