PCI Express

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

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.

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.

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.

However, the legacy configuration space for PCIe devices can still be accessed using the latter.

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.
		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.
	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.
Type 1	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.
	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.

Extended PCI Bus Numbering

Older variations of PCI (e.g. "PCI Conventional") were limited to a maximum of 256 PCI bus segments. PCI Express extends this by introducing "PCI Segment Groups", where a system could (in theory) have up to 65536 PCI Segment Groups with 256 PCI bus segments per group, thereby allowing a single computer to have up to a maximum of 2^{24} (16777216) PCI bus segments.

PCI Segment Groups are numbered, and in most systems there is only one PCI Segment Group (PCI Segment Group number 0). Note that legacy PCI configuration space access mechanism #1 (which still exists for backward compatibility) has no "PCI Segment Group" field and therefore can only be used to access the PCI configuration space for PCI Segment Group number 0. The #Enhanced Configuration Mechanism must be used to access the PCI configuration space for any devices in other PCI Segment Groups.

Enhanced Configuration Mechanism

The enhanced configuration mechanism makes use of memory mapped address space range/s to access PCI configuration space. Put simply, the memory address determines the segment group, bus, device, function and register being accessed. 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:

<table>
<tr><th>Offset</th><th>Length</th><th>Description</th></tr>
<tr><td>0</td><td>4</td><td>Table Signature ("MCFG")</td></tr>
<tr><td>4</td><td>4</td><td>Length of table (in bytes)</td></tr>
<tr><td>8</td><td>1</td><td>Revision (1)</td></tr>
<tr><td>9</td><td>1</td><td>Checksum (sum of all bytes in table & 0xFF = 0)</td></tr>
<tr><td>10</td><td>6</td><td>OEM ID (same meaning as other ACPI tables)</td></tr>
<tr><td>16</td><td>8</td><td>OEM table ID (manufacturer model ID)</td></tr>
<tr><td>24</td><td>4</td><td>OEM Revision (same meaning as other ACPI tables)</td></tr>
<tr><td>28</td><td>4</td><td>Creator ID (same meaning as other ACPI tables)</td></tr>
<tr><td>32</td><td>4</td><td>Creator Revision (same meaning as other ACPI tables)</td></tr>
<tr><td>36</td><td>8</td><td>Reserved</td></tr>
<tr><td rowspan="6">44 + (16 * n)</td><td rowspan="6">16</td><td colspan="3">Configuration space base address allocation structures. Each structure uses the following format:</td></tr>
<tr><td>Offset</td><td>Length</td><td>Description</td></tr>
<tr><td>0</td><td>8</td><td>Base address of enhanced configuration mechanism</td></tr>
<tr><td>8</td><td>2</td><td>PCI Segment Group Number</td></tr>
<tr><td>10</td><td>1</td><td>Start PCI bus number decoded by this host bridge</td></tr>
<tr><td>11</td><td>1</td><td>End PCI bus number decoded by this host bridge</td></tr>
<tr><td>12</td><td>4</td><td>Reserved</td></tr>
</table>

For non-x86 systems the method varies, but usually systems provide themselves with a Devicetree which is parsed at runtime.

To access a specific register within a device's PCI configuration space, you have to use the device's PCI Segment Group and bus to determine which memory mapped PCI configuration space area to use, and obtain the starting physical address and starting bus number for that memory mapped area. Once you have the correct starting physical address and starting bus number for that memory mapped area you would use the following formula to determine where the (4096-byte) area for a function's PCI configuration space is: `Physical_Address = MMIO_Starting_Physical_Address + ((Bus) << 20 | Device << 15 | Function << 12)`. Here, the `MMIO_Starting_Physical_Address` is the base address of the ECAM region from the MCFG table, and is always relative to bus number zero (ie. the configuration space for all bus 0 functions resides starting at the address in the table itself). You must start enumerating at the start bus number decoded by the host bridge.

Note that it may be a good idea to determine "physical address for this function's PCI configuration space" as part of PCI enumeration and store this physical address alongside any other information you're using to manage PCI devices and drivers (e.g. in your "device manager's" hierarchical tree of device info).

Also note that for absolute maximums (with 65536 PCI segment groups and 256 bus segments per segment group), the amount of physical address space consumed by memory mapped PCI configuration space ranges would be (up to) 16 TiB (or 2^{44} bytes); and ACPI's "MCFG" table may (in theory) be slightly larger than 256 MiB (a 16-byte entry for each individual PCI bus within each PCI segment group plus the 36-byte table header).

More information

Accessing a specific device within the space can be as follows: `(((bus * 256) + (slot * 8) + func) * 4096) + offs` (each device descriptor is 4096 bytes or 4K long), therefore the former can be interpreted as an array of the type: `((pcie_ecam[bus][slot][func] * 4096) + offs)`.

Finding devices can be done the same as PCI, with the difference that the kernel accesses a memory region rather than using CPU I/O.

See Also

References

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