ACPI VIOT DRAFT V9

December 2020

The Virtual I/O Translation Table (VIOT) describes the topology of para-virtual I/O translation devices (currently virtio-iommu) and the endpoints they manage. We introduce a new ACPI table rather than modifying an existing specification, because each IOMMU vendor maintain their own ACPI table (IORT for the Arm SMMU, DMAR for Intel VT-d and IVRS for AMD IOMMU). A para-virtualized IOMMU, as a software component between hypervisor and virtual machine, is multi-platform and should be maintained conjointly.

1 The VIOT table

The table starts with a standard ACPI header:

Field	Length	Offset	Description
Signature	4	0	'VIOT'. Virtual I/O Translation Table.
Length	4	4	Length in bytes, of the entire VIOT.
Revision	1	8	0.
Checksum	1	9	The entire table must sum to zero.
OEMID	6	10	OEM ID.
OEM Table ID	8	16	For the VIOT, the table ID is the manufacture model ID.
OEM Revision	4	24	OEM revision of the VIOT for the supplied OEM Table ID.
Creator ID	4	28	The vendor ID of the utility that created the table.
Creator Revision	4	32	The revision of the utility that created the table.
Node count	2	36	Number of nodes in the table.
Node offset	2	38	Offset from the start of the table to the first node.
Reserved	8	40	0.

The rest of the table is a list of *Node count* nodes, each describing either endpoints or translation devices. The first node is located *Node offset* bytes from the beginning of the table. Each node has a *Length* field defining its length, and the following node is located *Length* bytes from the beginning of the current node. Nodes must be aligned on 8 bytes.

Each node identifies one or more devices using either their PCI Handle or their base MMIO (Memory-Mapped I/O) address. A PCI Handle is a PCI Segment number and a BDF (Bus-Device-Function) with the following layout:

Bits 15:8 Bus number.

Bits 7:3 Device number.

Bits 2:0 Function number.

This identifier corresponds to the one observed by the operating system when parsing the PCI configuration space for the first time after boot.

Endpoint nodes declare an *Output node* that corresponds to the offset from the beginning of the table to the node describing the next translation device that manages these endpoint. They also declare one or more endpoint IDs that system software uses to identify endpoints when programming the translation device.

1.1 virtio-iommu based on virtio-pci node

A virtio-iommu device based on the virtio-pci transport, identified by the BDF of the virtio device.

Field	Length	Offset	Description
Type	1	0	3 – virtio-pci IOMMU
Reserved	1	1	0.
Length	2	2	Length of the node in bytes.
PCI Segment	2	4	The PCI Segment number of the virtio-iommu programming interface as returned by _SEG in the namespace.
PCI BDF number	2	6	Identifier of the PCI device.
Reserved	8	8	0.

1.2 virtio-iommu based on virtio-mmio node

A virtio-iommu device based on the virtio-mmio transport, identified by the base address of the virtio device. Like other virtio-mmio devices, properties of the virtio-iommu are described with a LNRO0005 element in the ACPI namespace.

Field	Length	Offset	Description
Type	1	0	4 – virtio-mmio IOMMU
Reserved	1	1	0.
Length	2	2	Length of the node in bytes.
Reserved	4	4	0.
Base address	8	8	Base MMIO address of the device.

1.3 PCI range node

A range of PCI endpoints identified by their segment and BDF number.

Field	Length	Offset	Description
Type	1	0	1 – PCI range
Reserved	1	1	0.
Length	2	2	Length of the node in bytes.
Endpoint start	4	4	First endpoint ID.
PCI Segment start	2	8	First PCI Segment number in the range.
PCI Segment end	2	10	Last PCI Segment number in the range.
PCI BDF start	2	12	First Bus-Device-Function number in the range.
PCI BDF end	2	14	Last Bus-Device-Function number in the range.
Output node	2	16	Offset from the start of the table to the next translation element.
Reserved	6	18	0.

The node refers to a PCI endpoint if the endpoint's segment number is in the range [Segment_start, Segment_end] and its BDF number is in the range [BDF_start, BDF_end]. The

corresponding endpoint ID is obtained by combining segment and BDF:

$$\text{endpoint\_ID} = ((\text{segment} - \text{Segment\_start}) << 16) + \text{BDF} - \text{BDF\_start} + \text{Endpoint\_start}$$

1.4 Single MMIO endpoint node

A single endpoint identified by its base MMIO address.

Field	Length	Offset	Description
Type	1	0	2 – MMIO Endpoint
Reserved	1	1	0.
Length	2	2	Length of the node in bytes.
Endpoint	4	4	The endpoint ID.
Base address	8	8	Base MMIO address of the endpoint.
Output node	2	16	Offset from the start of the table to the next translation element.
Reserved	6	18	0.

2 References

ACPI IORT IO Remapping Table, DEN0049D, https://developer.arm.com/docs/den0049/latest

ACPI DMAR DMA Remapping Table: "Intel®Virtualization Technology for Directed I/O", http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/vt-directed-io-spec.pdf

ACPI IVRS I/O Virtualization Reporting Structure, http://support.amd.com/TechDocs/48882_IOMMU.pdf

VIRTIO-v1.1 Virtual I/O Device (VIRTIO) Version 1.1. Edited by Michael S. Tsirkin and Cornelia Huck. 11 April 2019. OASIS Committee Specification 01. https://docs.oasis-open.org/virtio/virtio/v1.1/cs01/virtio-v1.1-cs01.html. Latest version: https://docs.oasis-open.org/virtio/virtio/v1.1/virtio-v1.1.html.

Contact: Jean-Philippe Brucker <jean-philippe@linaro.org>