A **Virtual Platform** allows the development of virtual machine guests that could be moved among different hypervisor systems and/or HW platforms *without further modification* through establishing an industry standard / de-facto standard.

**VISION:** Run Guests without modifications

**Hypervisor A**

**SoC A**

**Hypervisor B**

**SoC B**
Legend for VIRTIO terms

**Device** refers to the implementation of the virtual/para-virtual device, also known as Backend or Server

**Driver** refers to the guest driver, also known as Frontend or Client

**Device Host** is the guest that provides the Device to other guests

**Device Guest** is the consumer of a Device

**Guest** is a partition or virtual machine
Virtualized device Architecture with VIRTIO

Bulk data transport via DMA-like memory model
- Buffer allocations handled by „Driver“ part (client)
- Direct R/W access to allocated buffers in the „Device“ part (server)

Metadata transport via virt-queues (ring buffers, asynchronous pipeline)
VIRTIO-GPU

Virtio-GPU 2D
- Virtual Display, much like a VGA port
- Accepted for VIRTIO 1.1
- Framebuffers object are allocated in the driver
- Four Basic Commands
  - Get screen information
  - Attach framebuffer via sg_list
  - Set scan-out by reference
  - Flush scan-out

Virtio-GPU 3D
- Forward sanitized OpenGL Commands
- Support for OpenGLES 3.1, OpenGL 4.1, Vulkan support in development
- Based on open-source libvirgilrenderer on the host (BSD licensed)
- Based on Mesa driver for guest
- Shaders are transferred in intermediate format (TGSI)
Virtio GPU Architecture (QEMU)

Host Kernel
- DRAM/KMS
- GPU Driver

Guest Kernel
- Virtio-GPU driver

Guest
- MESA
  - Virgl Gallium

QEMU-KVM
- Virtio-GPU Device
  - libvirgilrenderer
  - LibGL
# Virtio GPU Overview

<table>
<thead>
<tr>
<th>Virgl Gallium</th>
<th>Virtio-GPU driver</th>
<th>libvirgilrenderer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implements Gallium driver for mesa</td>
<td>Screen metadata</td>
<td>Gallium to GL interface conversion</td>
</tr>
<tr>
<td>Generate virtio-GPU Command stream</td>
<td>Modesetting</td>
<td>TGSI-&gt;GLSL</td>
</tr>
<tr>
<td>Converts GLSL into TGSI</td>
<td>Context Management</td>
<td>Host side context lifecycle management</td>
</tr>
<tr>
<td></td>
<td>3D resource management</td>
<td>OpenGL version arbitration</td>
</tr>
<tr>
<td></td>
<td>DMA transfer initiation</td>
<td>Issues OpenGL commands to native LibGL</td>
</tr>
<tr>
<td></td>
<td>Command stream submission</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fencing</td>
<td></td>
</tr>
</tbody>
</table>