App Note - VC5 MMU

NOTES ON V2 driver at the end of this document

The VC5 includes an MMU on 7271x0 / 7268x0 / 7260A0 and 7278A0. This should lead to improved performance in a CMA based system (such as Android), is immune to fragmentation and increases the memory available to Linux for applications. It also provides cross process memory protection and coherency required for upcoming standards such as Vulkan.

NEXUS_MEMC0_GRAPHICS_HEAP heaps on these platforms have been shrunk to provide only space for interoperability surfaces, such as output framebuffers. Currently they are 64MB, so should be suitable to run any of the BSEAV/lib/gpu/applications samples in a non MMU configuration.

Requirements

1) BOLT image v1.25 or later
   The MMU driver is based on the standard Linux DRM framework (https://en.wikipedia.org/wiki/Direct_Rendering_Manager) and requires a device tree to operate. **BOLT versions from v1.25 will auto configure.**

   Prior versions can be used with the following commands to add entries at BOLT> command prompt

   ```
   dt add node rdb gpu
   dt add prop rdb/gpu compatible s 'brcm,v3d-v3.3.0.0'
   ```

   Version number is important, so contact the 3d team for your specific platform.

2) Linux kernel 4.1-1.7 or later
   4.1-1.7 kernel shipped with CONFIG_DRM enabled as default.

   Prior versions can be used by rebuilding the kernel with the requisite option above. Our Linux kernel ships with .config copied to config, which various parts of the Nexus build rely on. Nexus also needs to be rebuilt against the modified version as the module will fail to install. In some versions of the kernel the CMA driver is initialized with all the memory allocated, it is therefore necessary to use ‘`cmatool resetall`’ at boot to free the CMA allocation. **However later 4.1 kernels (e.g. 4.1-1.12) and the 4.9 kernels do not require this step.**

3) Boot line
   The shared memory between CPU and GPU is determined via the boot parameters. Nexus would ordinarily suggest a config, such as 'vmalloc=374m

bmem=696m@1352m’. As an example we may wish to allocate a shared region of 512MB before the BMEM region. drm.debug is optional and is suggested the first time you run, giving additional verbosity on the kernel log.

```
boot stbbri-fs-1:/41-1.6/vmlinuz-initrd-arm 'vmalloc=374m
bmem=696m@1352m brcm_cma=512m@820m drm.debug=0x2'
```

Running

No additional changes are required to run your application. Some new console output is visible and additional debug tools are available.

Is the MMU being used?

When starting an application, you should see on the console.

```
DRM: Trying to open: /dev/dri/card0
*** 00:00:03.342 nexus_statistics_callback: TaskCallback callback 0x79b38:0xb5e99cf8 from
modules/hdmi_output/src/nexus_hdmi_output.c:457 202334 usec
*** 00:00:03.342 nexus_statistics_module: display[Default]
nexus/core/syncthunk/nexus_display_thunks.c:26 201727 usec
*** 00:00:03.342 nexus_statistics_api: NEXUS_Display_GetSettings[display:Default] 201816
usec
DRM: brcmv3d Version 1.1.0 20161207, Broadcom V3D GEM provider
DRM: No Nexus secure graphics heap available
```

If you observe the message - “Unable to open device - defaulting back to Nexus”, then the device is in legacy mode.

It should be working but is not

Below are some simple steps to triage a system if you believe the MMU is not being used.

1) Check that the kernel module is built and present in NEXUS_BIN_DIR
   For example obj.97271/nexus/bin/brcmv3d.ko
2) Check the module can be installed on the STB
   insmod brcmv3d.ko
3) Check the node is present on the STB
   Is /dev/dri/card0

If you get any error messages with any of these steps, report them to the 3d team.

Debug capabilities

The MMU driver exposes its config via the debug fs.

Checking what is currently using the driver.

```
# cat /sys/kernel/debug/dri/0/clients
             command   pid dev master a   uid      magic
                cube  1448   0   y    y     0          0
```

Total allocations made on a client

```
# cat /sys/kernel/debug/dri/0/<PID>-<MAGIC>/client
```

Here, a single instance of cube is running.

Checking allocations made by a client

```
# cat /sys/kernel/debug/dri/0/<PID>-<MAGIC>/objs
    handle       size HW Virtual    RO    WC   EXT                         DESC
         1   67174400 0x00010000     n     n     y         NEXUS Offscreen Heap
         2    1048576 0x04020000     n     n     n
         3      65536 0x04120000     y     n     n
         5      65536 0x04290000     n     n     n
         6      65536 0x042a0000     n     n     n
         7      65536 0x042b0000     y     n     n
         8      65536 0x042c0000     y     n     n
         9      65536 0x042d0000     y     n     n
        10      65536 0x042e0000     y     n     n
        11      65536 0x042f0000     y     n     n
        12    1441792 0x04300000     n     n     y
        13      65536 0x04460000     y     n     n
        14      65536 0x04470000     y     n     n
        15    1441792 0x04480000     n     n     y
        16      65536 0x045e0000     y     n     n
        17      65536 0x045f0000     y     n     n
```

The Nexus offscreen heap is always mapped into the address space so the device can write to the interop surfaces.

Checking current 2MB pages

The DRM driver uses 2MB blocks, and then breaks them internally to a smaller granularity.

```
# cat /sys/kernel/debug/dri/0/<PID>-<MAGIC>/cma
 CMA Block Physcal:         64K Page Allocation Mask
====================================================
0x000000002f000000: ****************************----
```

In this instance, cube is running using a single 2MB CMA block, with four 64kb pages remaining.

V2 MMU Driver

The MMU driver has been altered for URSR 18.2 to operate to two distinct modes. CMA mode is the legacy mode which is backward compatible with the previous driver. shm mode allocates pages directly from Linux, so CMA is not required. Prior to selecting a mode of operation please consider the following.

1) DTU.
   If you require memory to be claimed back from the video decoder, then CMA must be used. The video decoder returns pages to the CMA region, which can be subsequently allocated via the MMU driver.
2) Speed.
   Allocating Linux shm pages is slower because of the system bookeeping overhead.
3) Compartmentalization.
   Under CMA, graphics is explicitly constrained to the defined region. With shm Linux, pages are returned until an OOM condition. Without further system setup (below) any unconstrained application could cause system services to fail and/or system reset.

Providing brcm_cma on the kernel boot line provides V1 behavior otherwise V2 is selected.

CGROUPS

Linux kernel uses cgroups as an accounting mechanism for allocations. By default the shipped BRCM kernel has ‘Enable CONFIG_CGROUPS’ enabled but ‘Enable CONFIG_MEMCG’ is disabled. **To use this feature you must have CONFIG_MEMCG enabled.**

```
# mount -t tmpfs none /sys/fs/cgroup
# mkdir /sys/fs/cgroup/memory
# mount -t cgroup none /sys/fs/cgroup/memory -o memory
# echo 128M > /sys/fs/cgroup/memory/0/memory.limit_in_bytes
# echo $$ > /sys/fs/cgroup/memory/0/tasks
```

This constrains everything run under the current bash shell to 128M total space.

Note that shm is used for all pages, not just graphics, so this must encompass all memory requirements.