Nexus

DTU Usage

Broadcom
Corporate Headquarters: San Jose, CA

Web: www.broadcom.com

Revision History

Revision	Date	Change Description
0.2	10/1/15	First draft
1.0	5/19/17	First release

Table of Contents

Overview

The DRAM translation unit (DTU) allows software to construct large blocks of physically contiguous memory by remapping bus addresses to DRAM. The DTU is built into the memory controller (MEMC) of the 7271 B0, 7268 B0, 7278 A0 and subsequent chips. This document describes how to use the DTU with Broadcom Nexus and NxClient software.

Bus Addressing

The DTU creates a new type of physical addressing called “bus addressing” which allows discontiguous 2 MB super pages of DRAM to be remapped into contiguous addressing regions. Physical memory addresses above the DTU are considered “bus addresses” which are mapped to physical memory addresses below the DTU called “device addresses”.

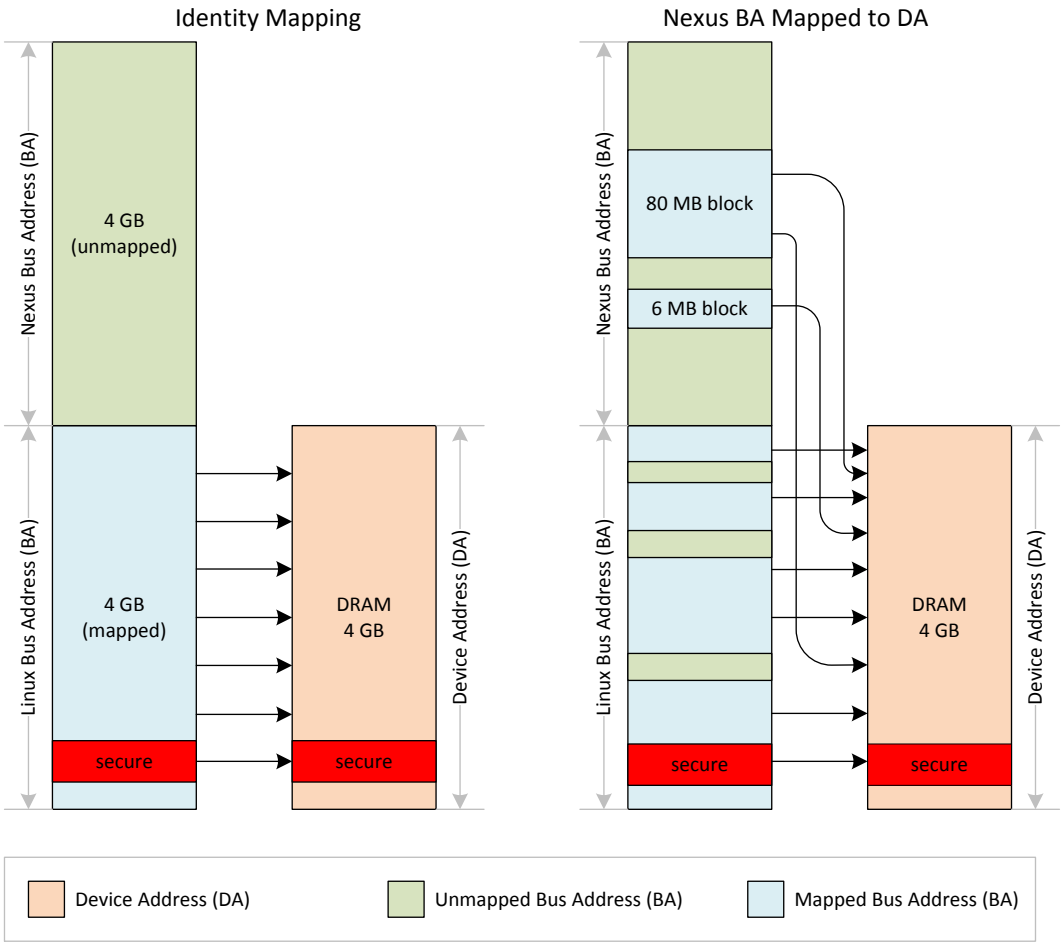
All hardware cores on the set-top pass through the DTU mapping and so they use bus addresses and can be remapped. No hardware core, including CPU or decoder or any other device other than the DTU itself, uses device addresses. When reading software documentation, all references to “physical addresses” should be understood as “bus addresses”. And, unless the DTU is specifically mentioned, it could be that the phrase “device address” may need to be understood as “bus address” when applied to a DTU system. The only device that actually works with “device addresses” is the DTU itself.

The following terms and abbreviations are used in this document:

Term (Abbreviation)	Meaning
Bus Address (BA)	Physical address used by all HW cores which will be translated by the DTU into DA
Bus Page (BP)	Index of the 2 MB super page in BA space
Device Address (DA)	DRAM address after DTU lookup. No HW block other than the DTU uses this.
Device Page (DP)	Index of the 2 MB super page in DRAM

The following diagram shows how DRAM is remapped with the DTU. When Linux boots, all DA is mapped to BA with the same mapping as if the DTU was not present. This is called an "identical mapping". It can also be referred to as "Linux Bus Addresses" because these are the only physical addresses that Linux will manage. After Nexus starts video decode, a set of discontiguous 2MB super pages have been mapped into contiguous BA space which is above the Linux Bus Address space. When the decode stops, the video memory is remapped to its original location and returned to Linux.

Mapping Bus Address (BA) to Device Address (DA)



The red "secure" regions show that some pages can be locked and never remapped by the DTU.

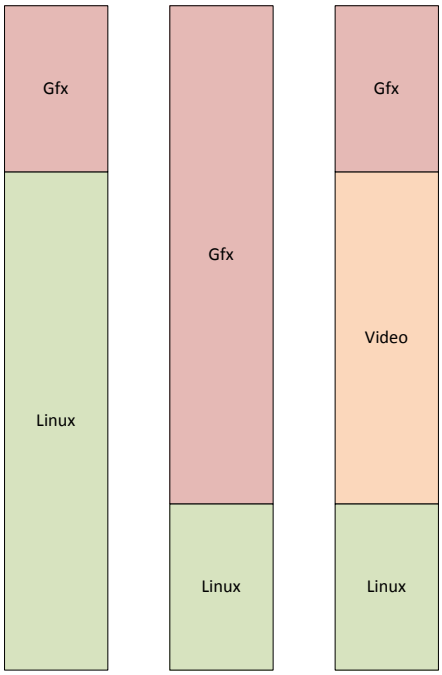
Each BP can either be unmapped or mapped to one DP. Each DP can either be unmapped or mapped to one BP. There are no one-to-many mappings for a BP or DP. In practice, all DP are always mapped, except momentarily during a remap operation. Also, BA can be viewed in two regions: the lower BA which is "identical mapped" to DA at boot time, and the higher BA which is used by Nexus to remap pages into contiguous addressing regions.

Use Cases

The DTU allows a Broadcom set-top box to toggle between use cases with dramatically different memory requirements. These use cases generally fall into three categories:

- Operating system (Linux) heavy – IP streaming, DLNA, web browsers, etc.
- Graphics heavy – 3D games, rich GUI's
- Video heavy – 10 bit 4K decode with simultaneous transcode or PIP

The three categories can be pictured as follows:



In typical systems, Linux will always use some minimum amount of DRAM, but may take significantly more, often bounded by the total memory available.

Similarly, graphics will always use some minimal amount because there is always some GUI, but may take significantly more, often bounded by the total memory available. Also, all systems with a DTU have a VC5 3D graphics core with an internal MMU. This is described in the following section, but it means that the DTU is not needed for memory sharing between Linux and 3D graphics.

Video memory is unlike Linux and graphics memory because it may take a minimum of zero (if no video is being decoded) and scales up to a maximum determined by the capabilities of the chip, not by the total amount of DRAM. Video memory requirements can be measured using Nexus Memconfig for existing systems and can be approximated for new systems (see

nexus/docs/Nexus_Memory.pdf). For example, the 97445 reference board has 3 GB of total memory and can use a maximum of around 512 MB of picture buffers in box mode 1. The 97439 has 2 GB of system memory and defaults to 300 MB of picture buffers. This video memory must be physically contiguous for each channel and on each memory controller. To guarantee the availability of contiguous video memory, systems without a DTU must pre-allocate video memory at system initialization, which means it cannot be shared with Linux or graphics.

The DTU allows this contiguous video memory to be constructed with 2MB pages allocated at runtime from Linux CMA (contiguous memory allocator). The user must declare the amount of CMA memory available to Nexus for the DTU using the Linux boot parameter "brcm_cma" documented below.

The DTU could also construct contiguous memory for other uses, like audio or transport, but those buffers are usually much smaller and don't impact the system as much. Nexus does not currently have any support for using the DTU for anything except video memory.

VC5 graphics memory

All systems that have a DTU also have a VC5 3D graphics core with an internal MMU. The MMU is able to construct contiguous memory for VC5 from memory allocated from Linux CMA. On these systems, Nexus will build a brcmv3d.ko driver which implements the Graphics Execution Manger (GEM) interface. The GEM driver will allocate CMA in 2 MB pages to minimize fragmentation when using in conjunction with the DTU for video memory.

The user must declare the amount of CMA memory available to the VC5 GEM driver using the Linux boot parameter "brcm_cma" documented below.

Linux brcm_cma Boot Parameter

At boot time, Linux owns all system memory and the DTU is configured with an "indentical map".

Broadcom's Linux distribution has a "bmem" boot parameter which carves out contiguous memory for Nexus. On systems without a DTU, all Nexus device memory must come from these bmem regions. On systems with a DTU, only non-video Nexus device memory must come from these bmem regions, like audio and transport, while video memory can come from Linux CMA (contiguous memory allocator).

The user must declare the amount of CMA memory available to Nexus for the DTU and for the VC5 GEM driver. When the CMA memory is not allocated by Nexus or the GEM driver, Linux will make use of it. Because of fragmentation and pinning, it's important that the total amount of CMA be a margin greater than the exact maximum amount of video and graphics memory needed. A safe margin would be 50% more. Lower margins could be used if confirmed with system test.

The CMA region must be placed with knowledge of the physical addressing of the chip and the requirements of Nexus bmem.

Given a single MEMC system with 2GB, we recommend that brcm_cma be placed at the top of the 2GB. For instance, if Nexus recommends bmem=724m@368m (which is a size of 724 MB starting at physical address 368 MB), and if you need 512MB of graphics and video memory, you would boot with:

```
bmem=724m@368m brcm_cma=512m@1536m
```

Nexus will call the Linux CMA driver to allocate 2 MB pages, then remap them using the DTU. Before returning CMA memory to Linux, each 2MB super page must be remapped to its original location. Linux performs no DTU operations and has no knowledge of DTU remapping.

Configuring Nexus

Nexus requires a small amount of init time configuration to use the DTU. All runtime CMA allocations and DTU remappings are handled internally after that.

After calling NEXUS_Platform_GetDefaultSettings, some or all picture buffer heaps should be set with the following:

- heap[].heapType should have the NEXUS_HEAP_TYPE_DTU bits set.
- heap[].offset should be set to unmapped BA space above the Linux Bus Address space. This requires the application to know and manage the unmapped BA space for the system.
- heap[].alignment should be set to 2MB (corresponding to the super page size)

Also, after calling NEXUS_GetDefaultMemoryConfigurationSettings, set videoDecoder[].dynamicPictureBuffers = true and stillDecoder[].dynamicPictureBuffers = true for each decoder using a DTU picture buffer heap. This tells Nexus to not allocate video decoder memory at init time. Video display memory is already allocated at runtime.

See nexus/examples/dvr/playback_dtu.c for an example of these settings.

After that, no other change is needed in your application. When you start decode, Nexus video_decoder and display will allocate memory from the picture buffer heap, which will be internally redirected to a series CMA allocations and a DTU remap request. When decode is stopped, the memory will be remapped to its original “identical mapping” and returned to Linux.

Configuring NxClient

To use the DTU, start nxserver with the "-dtu" command line parameter. NxClient will start Nexus with the configuration described above.

SVP

The DTU is designed to work with SVP (Secure Video Processor) systems. Nexus must work with SAGE to lock and secure DTU pages before any SVP decode is performed. The DTU also contains a scrubbing feature used by SAGE after an SVP decode.

Cost/benefit analysis

The DTU is not a silver bullet that solves all memory problems. Each customer must do a cost/benefit analysis to determine if the DTU should be enabled. If it is not needed, the DTU can be ignored or disabled.

If the DTU is used, Nexus will start with no video memory allocated. When starting decode, Nexus will try to allocate the required 2MB pages from Linux. If there are not enough, decode will fail. The way to make decode succeed again is to free up Linux memory from the rest of the system. This can only be done by a higher level software entity requesting applications to voluntarily give up memory or by forcibly killing off background or optional applications. Nexus and Linux cannot free up application memory by itself.

If the system cannot perform this aggressive memory management, it should continue to use the existing init-time allocation of video memory and not use the DTU.

Also, the DTU may have greater or lesser value relative to the required DRAM parts. If your system requires 1.6 GB, you will likely populate 2GB and the DTU may be of little help because of the extra memory available. But your system requires 1.2 GB or 2.2 GB, the DTU may allow you to successfully build a 1 GB or 2 GB system. The exact amount can only be determined by measuring the amount of memory needed in non-simultaneous use cases.

dtutool

A command line Linux tool to monitor the DTU is located at rockford/applications/dtu/dtutool.c. Run "dtutool -print" to see the current DTU mapping.

During development, Nexus is often restarted. Nexus may have crashed or the user may have forcefully terminated it. In kernel mode, Nexus will detect the termination and return all non-

secure pages to Linux. However, in user mode Nexus cannot detect this. Instead, run “dtutool -restore” to remap all DTU pages and return CMA to Linux. Then run Nexus again.