Hardware Memory Allocator

1

Generated by Doxygen 1.8.3.1

Fri Aug 2 2013 13:55:21

Contents

Chapter 1

Hardware Memory Allocator

HMA는 Hardware(Hyunwoo?) Memory Allocator의 약자로 gfx, jpeg decoder, video decoder, display engine 등 hardware element가 사용하는 메모리를 관리하는 기능을 갖는 모듈이다. 이 메모리 할당자는 CMA(Contiguous Memory Allocator)의 초기 버전을 많이 참고하였다.

GP4에서 gfx, jpeg decoder 등 각각의 hardware element는 고정된 위치의 메모리를 갖고 있는데, jpeg 그림을 decoding하거나 decoding된 그림을 display engine으로 출력할 때 그 상황에 맞는 크기의 하드웨어 메모리가 필요하며, 필요한 메모리 갯수 또한 상황에 따라 달라진다.

예를들어 jpeg decoding의 경우 decoding된 그림을 저장하는 메모리를 항상 고정된 위치의 메모리 주소로 사용하고 있는데, 이는 web browser, MJPEG video play 등 다중 프로세서에서 동시에 jpeg decoder로 접근할 때 decoded memory의 일관성이 깨지는 문제가 생길 수 있다. 또한, 현재 gfx에 구현된 graphic surface용 hardware memory allocator의 경우 alloc 후 free를 하지 않고 프로세서를 끝내면 메모리 누수가 생기는 문제를 갖고 있어 이에 대한 보안도 필요하다.

이러한 문제를 해결하기 위해 hardware memory를 관리하는 간단한 모듈을 만들어 사용하고자 한다. 항상 동작하지 않는 hardware에 고정으로 할당된 memory 또한 동적 할당을 할 수 있을 것으로 기대하며, 이런 경우 현재 부족한 메모리 자원을 효과적으로 사용할 수 있을 것으로 기대한다.

1.1 구성 및 구현 내용

1.1.1 memory pool

hardware module은 특정 영역의 메모리만을 접근할 수 있는 제약을 갖고 있기도 하다. 예를 들어 display engine의 경우 첫번째 hardware memory bank(?)를 주로 접근할 수 있다. 또한, memory allocator에서 사용할 수 있는 memory 공간이 연속적이지 않고 몇군데로 나누어져 있을 수 있다. 이러한 경우를 고려해 메모리 영역의 특성과 사용가능한 메모리 공간을 적당한 구역으로 나누어 memory pool로 구분하였다. memory pool은 커다란 memory 덩어리로써, memory allocator가 메모리를 할당할 수 있는 영역이 된다. 즉, 메모리를 할당하고자 하는 수요자는 자기가 메모리를 할당하고자 하는 memory pool을 미리 알고 있어야 한다.

memory pool은 한개만 있을 수도 있지만, 두개 이상이 존재할 수도 있다. 당연히 memory pool이 없다면 memory allocator가 동작할 수 없으며, memory allocator는 memory pool에서 적당히 빈 공간을 찾아 메모리를 할당해 준다.

1.1.2 memory allocator

memory allocator는 주어진 memory pool에서 요청된 크기의 메모리를 요청된 address align에 맞도록 메모리를 할당한다. 보통 하드웨어 메모리의 경우 빈번히 할당/해제가 되지 않고, 모듈의 시작지점에서 필요한 memory를 할당하고 모듈의 동작이 끝나면 할당된 memory를 해제하므로 그렇게 고성능의 allocation algorithm이 필요하지 않다. 또한, 하드웨어에 따라 요구하는 address align이 다르므로 allocation algorithm을 최적화 시키기가 힘들다. 이러한 이유 때문에, HMA에서 구현된 memory allocaton의 동작은 최대한 간단하게 했으며, 수행 성능은 특별히 고려하지 않았다.

memory allocator는 chunk 를 최소 단위로 해 linked list로 관리한다. chunk 는 메모리의 주소, 크기, 현재 할당되어 사용중인지, 할당한 프로세스 이름은 무엇인지 등의 정보를 갓고 있다. chunk 의 linked list는 pool 에 속해 있으며, pool 은 memory pool의 이름, 시작주소, 크기, chunk 를 접근할 때 사용하는 spin lock, chunk 의 root pointer 등을 갓고 있다.

1.1.3 /proc interface

/proc 는 kernel 에서 제공하는 파일 시스템으로, 커널 내부의 드라이버나 각종 정보를 사용자에게 전달해 주는 파일 시스템이다. HMA의 memory allocator는 내부 상태, 메모리의 사용 정보, 할당 정보, 등의 모든 정보를 /proc/hma 디렉토리 아래에 있는 파일들로 사용자에게 전달한다. 또한 할당된 메모리 내용을 dump 할수 있도록 하는 기능또한 지원한다.

memory allocator에서 사용하는 각각의 pool 은 /proc/hma 디렉토리 아래에서 하나의 디렉토리로 나타나며, 디렉토리 이름은 그 pool의 이름이다. 그 디렉토리 안의 status 파일의 내용은 그 pool 에서 할당된 메모리 상태를 나타낸다.

enable_dump 모듈 파라미터가 0이 아닌 값을 가질 때에는 hma_alloc() 함수 등에서 할당하는 메모리에 대해 /proc/hma/pool_name/ 디렉토리 아래에 메모리 덤프 용도로 할당 된 시작 주소에 해당하는 파일 이름을 생성한다. 이 파일을 읽음으로써 할당된 memory chunk 의 내용을 읽을 수 있으며, dd, cat 프로그램을 이용해 해당 메모리 내용을 dump 할수 있다.

1.1.4 user level interface

memory allocator는 hma_alloc_user(), hma_alloc(), hma_free() 등 메모리를 할당하고 반납하는 두개의 함수를 하드웨어 메모리가 필요한 드라이버에게 제공한다. 하지만, 이는 커널 내부 함수로써, 드라이버 등 커널 레벨 함수이기 때문에, 사용자 어플리케이션에서 바로 사용할 수 없다. 사용자 어플리케이션에서도 이러한 하드웨어 메모리를 사용할 일이 있는데, 이때 hma_alloc(), hma_free() 함수를 제공하기 위해 character device driver를 만들었다.

여기서 제공하는 character device driver는 하나의 file descriptor에서 하나의 chunk 를 할당할 수 있다. 사용자 어플리케이션에서 여러개의 메모리 할당을 원한다면, 각각의 메모리 할당을 할때마다 여기서 제공하는 character device driver를 open 해야 할 것이다.

사용자 어플리케이션에서 하나의 메모리를 할당하기 위해서는 먼저 device driver를 open 한 다음 ioctl을 통해 사용하고자 하는 pool 이름과 할당하는 메모리 크기, 메모리 시작주소의 alignment 값을 hma_ioctl_alloc 구조체를 통해 전달하면 된다. 사용한 메모리를 반납하기 위해 device driver를 close 하면 된다.

1.1.5 user level library

user level interface 의 경우 device driver를 통해 open, ioctl, close system call 함수를 호출해야 하므로 사용하기가 번거롭다. 사용자 라이브러리에서 이를 사용하기 쉽게 하기 위해 user level interface 를 간단히 감싸는 라이브러리를 제공한다.

사용자 어플리케이션에서 메모리를 할당하기 위해서는 libhma_alloc() 함수를 이용해 메모리를 할당한다. 할당한 메모리를 반납하기 위해서는 libhma_free() 함수를 이용한다.

만약 사용자 어플리케이션을 끝내기 전에 할당한 메모리를 반납하는 경우라면, libhma_free 함수는 호출할 필요가 없다. 할당한 메모리를 그대로 둔 채 사용자 프로그램을 끝내면(exit() 함수를 호출 하거나 main()함수를 끝내버리면) 리눅스 커널에서 현재 open 된 파일을 자동으로 close 해주기 때문에 user level interface 드라이버에서 자동으로 hma_free() 함수를 호출 해 줄 것이다.

1.2 참고 링크

1. kernel debugging using proc "sequence" files
2. contiguous memory allocator

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

Module Documentation

4.1 HMA

Functions

- unsigned long hma_alloc_user (const char *poolname, int size, int align, const char *user)
  메모리 할당을 한다.
- unsigned long hma_alloc (const char *poolname, int size, int align)
  메모리 할당을 한다.
- void hma_free (const char *poolname, unsigned long addr)
  사용 후 메모리를 반환한다.
- int hma_pool_register (char *name, unsigned long start, int size)
  memory pool을 등록한다.
- void hma_pool_unregister (char *name)
  memory pool을 더이상 사용하지 않게 등록을 취소한다.

4.1.1 Detailed Description

hardware memory를 관리하는 핵심 모듈이다. hma_pool_register(), hma_pool_unregister() 등의 함수로 memory pool을 관리하며, hma_alloc_user(), hma_alloc(), hma_free() 등의 함수로 특정 메모리 pool 안에서 원하는 크기의 메메모리를 할당하고 반납한다.

memory pool은 사용 가능한 빈 공간의 메모리 덩이를 말하며, HMA 모듈이 동작하기 위해서는 하나 이상의 memory pool이 주어져야 한다. 각각의 memory pool은 null byte를 포함해 HMA_POOL_LEN 크기의 이름을 가지며, 이 이름을 이용해 hma_alloc_user(), hma_alloc(), hma_free() 에서 memory pool을 지정할 수 있다.

4.1.2 Function Documentation

4.1.2.1 unsigned long hma_alloc (const char * *poolname*, int *size*, int *align*)

메모리 할당을 한다.

poolname에 해당하는 memory pool 이름에서 *size* 크기의 메모리를 할당한다. 메모리의 시작주소는 *align* 에 align 시킨다.

multithread에 안전하다.

Parameters

poolname	memory pool name
size	memory size
align	memory align. ex. 0x1000 for 12bit aligned.

4.1.2.2 unsigned long hma_alloc_user (const char * *poolname*, int *size*, int *align*, const char * *user*)

메모리 할당을 한다.

poolname 에 해당하는 memory pool 이름에서 *size* 크기의 메모리를 할당한다. 메모리의 시작주소는 *align* 에 align 시킨다. hma_alloc() 함수와 같은 기능을 하나 메모리 사용자 이름을 *user* 를 통해 정해줄 수 있다.

사용자 이름으로 정하는 *user* 는 디버깅 용도로 사용하게 된다. 사용자 이름을 특별히 정하지 않으려면 hma_alloc() 함수를 사용할 수 있다.

multithread에 안전하다.

Parameters

poolname	memory pool name
size	memory size
align	memory align. 0x1000 for 12bit aligned.

4.1.2.3 void hma_free (const char * *poolname*, unsigned long *addr*)

사용 후 메모리를 반환한다.

hma_alloc() 혹은 hma_alloc_user() 함수에 의해 할당받은 메모리를 반환한다. 반환 하고자 하는 메모리의 memory pool name을 모르고 있다면 *poolname* 을 NULL로 할 수 있다.

multithread에 안전하다.

Parameters

poolname	memory pool name
addr	memory start address

4.1.2.4 int hma_pool_register (char * *name*, unsigned long *start*, int *size*)

memory pool을 등록한다.

memory pool을 등록한다. memory pool은 여러개가 등록될 수 있으며, hardware memory를 사용하고자 하는 곳에서 원하는 memory pool에서 hma_alloc() 함수를 이용해 memory를 할당한다.

보통 플랫폼 초기화 과정이나 driver module 초기화 과정(insmod)에서 사용하며, multithread에 대해 안전하지 않다.

등록된 memory pool은 hma_pool_unregister() 함수를 이용해 등록을 취소한다.

Parameters

name	memory pool name
start	start address of the memory pool
size	size of the memory pool

4.1.2.5 void hma_pool_unregister (char * *name*)

memory pool을 더이상 사용하지 않게 등록을 취소한다.

hma_pool_register() 함수를 이용해 등록한 memory pool을 더이상 사용하지 않도록 한다. 현재 해당 memory pool에서 hma_alloc() 함수를 이용해 할당 받은 메모리에 대해서는 더이상 메모리 관리의 책임을 지지 않는다.

보통 kernel driver module을 kernel에서 내릴 때(rmmod) 호출되며, multithread에 대해 보호를 하지 않는다.

Parameters

name	memory pool name

4.2 HMA user level interface driver

사용자 어플리케이션에서 HMA 메모리를 할당하고 반납하도록 도와주는 드라이버이다.

4.3 HMA user level library

Functions

- unsigned long libhma_alloc (const char *name, int size, int align, void **vaddr)

  HMA 메모리를 할당한다.(noncached memory)
- unsigned long libhma_alloc_cached (const char *name, int size, int align, void **vaddr)

  HMA 메모리를 할당한다.(ncached memory)
- int libhma_cache_ctl (unsigned long paddr, void *vaddr, size_t size, enum libhma_cache_control ctl)

  hma에서 할당받은 메모리를 cached로 사용하려면 cache를 수동으로 control하는 것이 필요하며 이를 제공함. paddr : libhma_alloc_cached 으로 메모리를 할당 받았을 때 return 된 값. 해당 메모리의 키값으로 사용됨. vaddr : 할당 받은 메모리의 가상주소, 컨트롤 하려는 시작 주소값. size : 컨트롤 하려는 크기 ctl : cache invalidate(HMA_CACHE_INV), cache clean(HMA_CACHE_CLEAN)
- void libhma_free (unsigned long paddr)

  HMA 메모리를 반납한다.

4.3.1 Detailed Description

사용자 어플리케이션에서 HMA 메모리를 할당하고 해지하는 함수를 제공해준다. HMA는 연속적인 하드웨어 메모리를 관리하는 모듈로 사용자 어플리케이션에서는 HMA user level library 모듈을 통해 HMA user level interface driver 을 이용해 메모리를 할당한다.

메모리를 할당할 때에는 libhma_alloc() 함수를 사용하면 되며, libhma_free() 함수를 이용해 메모리를 해지하면 된다.

4.3.2 Function Documentation

4.3.2.1 unsigned long libhma_alloc (const char * *name*, int *size*, int *align*, void ** *vaddr*)

HMA 메모리를 할당한다.(noncached memory)

aa

4.3.2.2 void libhma_free (unsigned long *paddr*)

HMA 메모리를 반납한다.

free hma memory

Chapter 5

Data Structure Documentation

5.1 hma_drv_mem_desc Struct Reference

Data Fields

- unsigned long **paddr**
- void * **vaddr**
- int **size**
- struct vm_area_struct * **vma**
- char **poolname** [HMA_POOL_LEN]
- int **cached**

The documentation for this struct was generated from the following file:

- hma_drv_ext.h

5.2 hma_ioctl_alloc Struct Reference

```
#include <hma_drv.h>
```

Data Fields

- char pool [HMA_POOL_LEN]
- int size
- int align
- unsigned long paddr

5.2.1 Detailed Description

HMA_IOCTL_ALLOC ioctl system call 에 사용되는 구조체이다. 할당하고자 하는 메모리 pool의 이름, 메모리 크기, 메모리 시작주소의 alignment를 나타낸다.

5.2.2 Field Documentation

5.2.2.1 int hma_ioctl_alloc::align

할당 받을 메모리 시작주소의 aligment

5.2.2.2 unsigned long hma_ioctl_alloc::paddr

할당 받은 메모리의 시작주소, 물리 주소

5.2.2.3 char hma_ioctl_alloc::pool[HMA_POOL_LEN]

메모리를 할당받을 때 사용할 memory pool 이름

5.2.2.4 int hma_ioctl_alloc::size

할당 받을 메모리 크기

The documentation for this struct was generated from the following file:

- hma_drv.h

5.3 hma_ioctl_cache_ctl Struct Reference

`#include <hma_drv.h>`

Data Fields

- unsigned long **paddr**
- void * **vaddr**
- size_t **size**
- int **operation**

5.3.1 Detailed Description

HMA_IOCTL_CACHE_CTL ioctl system call 에 사용되는 구조체이다. L1, L2 cache를 invalidate, clean 하기 위해서 필요한, 유저영역 가상주소, 물리 주소, 크기를 나타낸다 L1은 가상 주소를 사용하며, L2는 물리 주소를 사용한다. cache 의 용어가 혼용되기 때문에 hma에서 제공하는 cache operation을 정확히 설명한다. FIXME CACHE_INV(cache invalidate) : cache line의 속성이 valid, dirty인 것을 unvaild로 변경 CACHE_CLEAN(cache clean) : cache line의 속성이 dirty인 것을 main memory에 write하고 unvaild로 변경 CACHE_FLUSH(cache flush) : x

The documentation for this struct was generated from the following file:

- hma_drv.h

Index