A Quick Solution for conventional H/W IP in the modern ARM based SoC

Renesas Electronics Corporation

Open Source Development Center
Automotive Information System Business Division

Yoshiyuki Ito

Rev. 1.01

Content

- Introduction
- Conventional H/W IP and Modern SoC
- Proposed method to upstream community: IOMMU
- Requirements to use conventional H/W IPs
- Simple and quick solution: static mapped IPMMU
- Pros & Cons
- Conclusion

Introduction

- Target:
  - To utilize over 32bit physical address space of ARM Cortex-A15
- Standard Solution:
  - Enable LPAE (Long Physical Address Extension) on the Linux Kernel
- Issue:
  - How to manage DMA from/to 32bit bus width H/W IP
- Method
  - Limiting and binding DMA area to 32 bit address space
  - Utilize MMU for H/W IP　(IPMMU facilities)

- Points of Concern
  - Must not conflict with upstream implantation
  - Consider about the Performance Stability

Conventional H/W IP and Modern SoC

- Modern ARM base SoC for Car Infotainment Systems
  - Uses : Cortex-A15 or any ARMv7a CPU Core

  → Has 40bit width address bus

- H/W IPs in the SoC
  - Peripherals:
    - Storage Controller, Data Connections like USB, etc.
  - For Multimedia:
    - Codecs, 3D Graphics, Camera Controller, etc.

  → Some of them are carried from Cortex-A9 generation

  → Has 32bit width address bus

Issue:

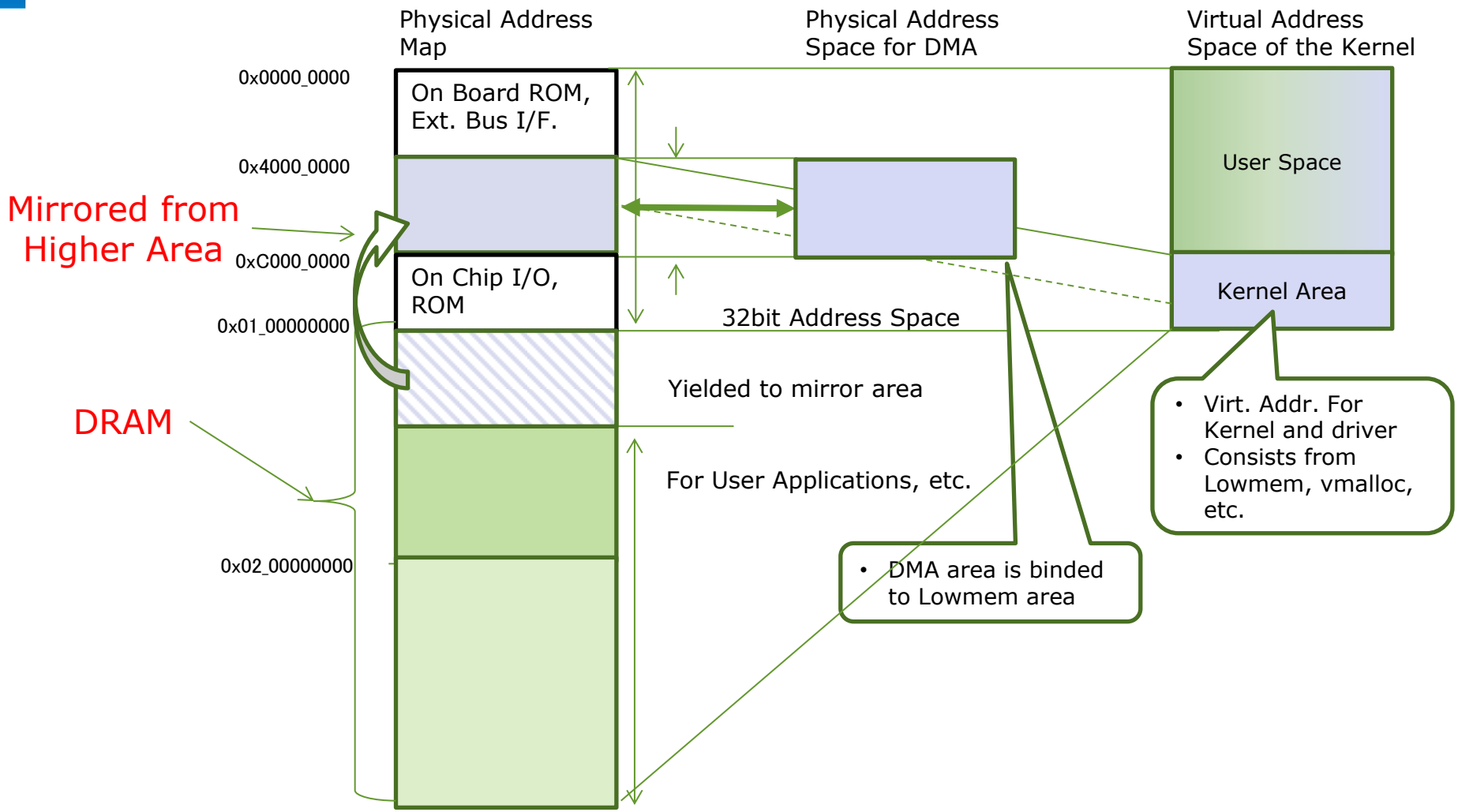
→ How to manage 32bit I/F IP on 40bit address space

Three Typical DMA model in the consideration

- Candidates to map 32bit address space to 40bit address space
  - Assumption:
    - Lower 32bit address space consists from
      - DRAM, Memory mapped registers, Boot ROMs, etc.

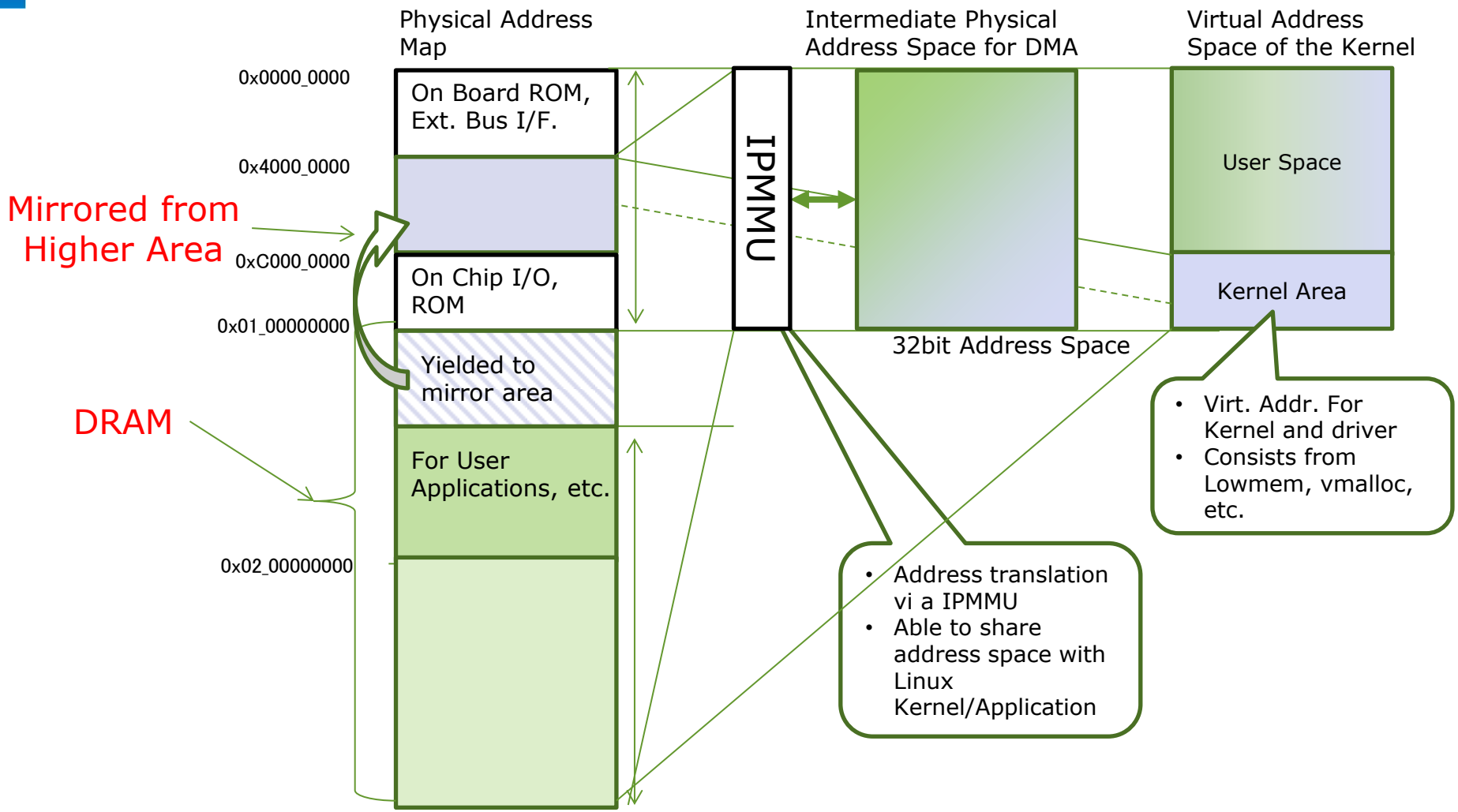
      → Some portion of DRAM is mapped into 32bit space
  1. Bind DMA space to lower 32 bit address space
  2. Static mapping to DRAM space (upto 4GByte) with Large Page Size PTE
  3. Dynamic mapping with ordinary PTE and IPMMU

1) Binding DMA area to lower 32 bit address space



Example: Reference Board Implementation of Renesas's R-Car H2

3) Dynamic mapping with ordinary PTE and IPMMU



Example: Reference Board Implementation of Renesas's R-Car H2

Proposed method to upstream community: IOMMU

- Laurent Pinchart of Renesas Linux kernel team proposing IOMMU to ARM SoC community

  Patch: "iommu/shmobile: Add iommu driver for Renesas IPMMU modules", etc.

  http://events.linuxfoundation.org/sites/events/files/slides/20140429-dma.pdf

  - IOMMU for ARM:
    - Delivered from PCI and IA32 environment to manage MMU of IPs or off-chip devices
    - Dynamic page manipulation for each 4KByte to H/W IPs

→ General and Scalable Solutions for DMA with H/W IP

Requirements to use conventional H/W IPs

It's a best solution "IOMMU" but..

- Overspec:
  - Dynamic Page Mappings is not so needed on current multimedia
    - Contiguous memory are used in Multimedia
    - Memory allocation is done at driver layer
      - User space buffer is not strongly needed

- Overhead:
  - TLB Walk at unexpected timing
    - Many TLB entries consumed: Linear memory walk, Write once, Many Source
  - Page Table Walks required in Cache Manipulation
    - H/W IP handlings uses DMA: Cache Flash/Invalidation needed

Requirements to use conventional H/W Ips (contd.)

- Tight performance request for current SoC
  - Example:

    Full HD Video Stream:

    Not so huge on single video transaction, but

    1080x1920 x 60fps x yuv422 ≅ 0.5GByte/Sec

    X Video Output, Decoder (in, out), Image Composition

    X to HDMI, to WiFi, to USB, ….

    Many transactions must handled without "under run"

→ Consideration needed from performance point of view

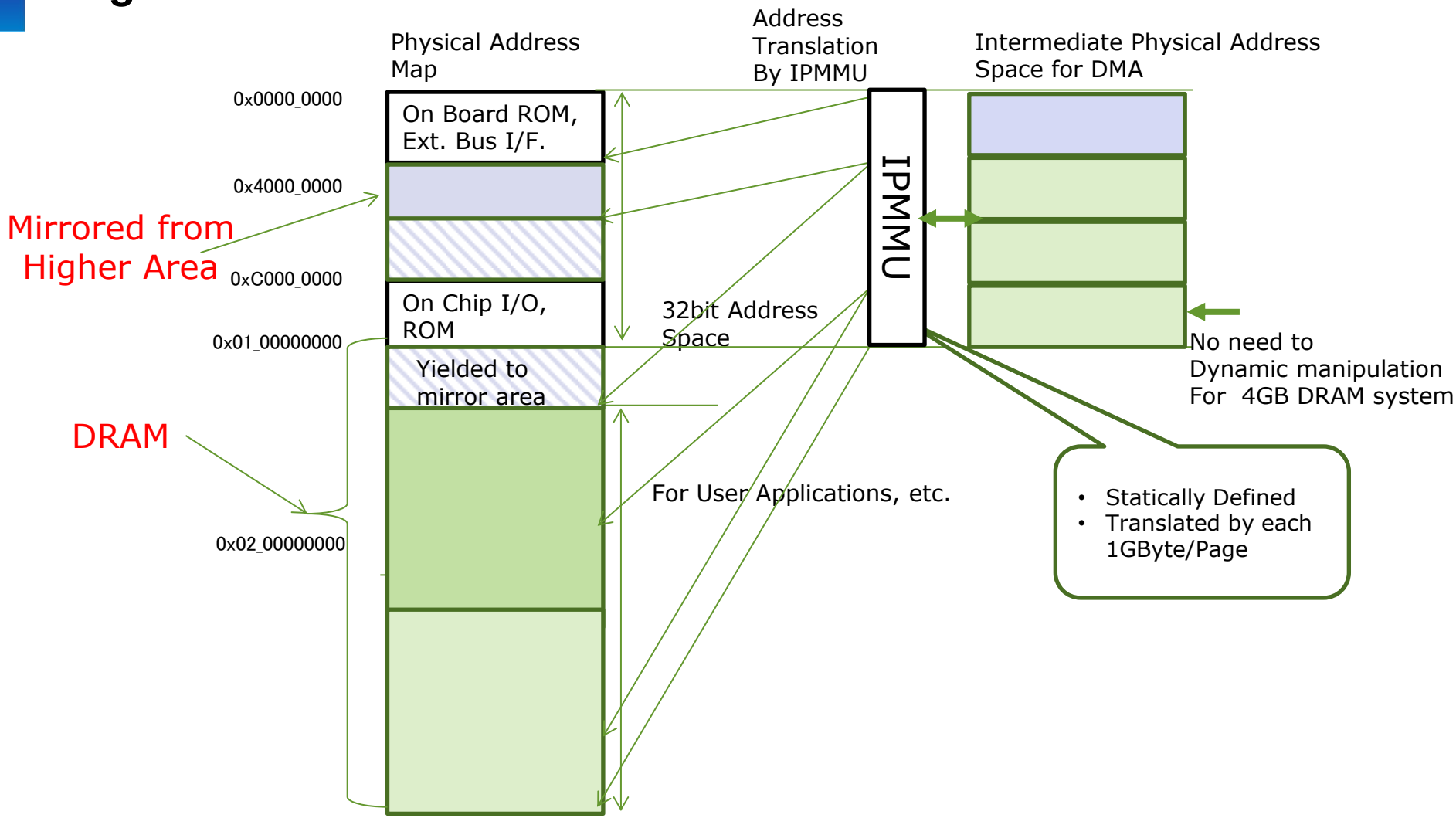
Simple and quick solution: static mapped IPMMU

- To avoid performance instability: Static Map used in IPMMU
- To Simplify:
  - Targeted to..
    - Contiguous memory area
      - Allocated inside driver layer
      - Statically mapped in driver initialization
    - Up to 4GByte DRAM address space
      - Assuming condition of current DDR specs. on the CIS
      - It regard as future issue that partial dynamic manipulation for “over 4GB DRAM”

→ Quick Solution:

- Using..
  - Large page size:
    - 1GByte for each page
  - Intra drive implementation:
    - To avoid conflict with dynamic IOMMU

2) Static mapping to DRAM space (up to 4GByte) with Large Page Size PTE



Example: Reference Board Implementation of Renesas's R-Car H2

To avoid conflict with IOMMU

- Need to coexist with IOMMU and static mapped IPMMU

→ So it made under strategies as:

1. Scope of functions defined inside each drivers
   - PTE definition, IPMMU init., Address Translation
2. Divide H/W resources for IOMMU and IPMMU
   - Able to assign IPMMU H/W for each, in R-Car Series
3. General Memory Allocation method must be use (CMA e.g.)
   - To share memory area between dynamic/static map
4. Must use Physical Address Pointer to inter-driver communication
   - Phys. Addr. is common address space both dynamic/static mapping
   - Simple address translation functions needed

→ Able to quick implementation even if satisfy above condition

Implementation

- IPMMU manipulation
  - Only 200 lines for each IPMMU — On Each Drivers
    - Initialization of PTE for IPMMU
    - IPMMU enable and disable
    - Address translate function
  - Made a IPMMU management driver to share an IPMMU from several drivers — On Each Drivers
  - Modified DMA setting method in each drivers — On Each Drivers
    - Using Intermediate Physical Address
  - Memory reservation method implemented in board dependent kernel code — Upstream Impl.
    - To setup memory pool before execute DMA from each driver
    - Contiguous Memory Allocation is used

Pros & Cons

- Pros:
  - Simple:
    - Only 4 pages for 4GB DRAM
      →Able to use:
      Static Mapping
      Address Translate w/o PTE walk (bi-direction)
    - Stable and high performance
      →Easy to verify, Able to aim performance high edge
- Cons:
  - Scalability:
    - Over 4GB DRAM system
      →Must prepare to combine dynamic manipulation
  - Not able to DMA with Non-Contiguous pages
    - User Address Space, etc.
      →Must coexist with dynamic IOMMU

Conclusion

- Renesas is keeping propose general/scalable IOMMU solution to ARM based SoCs
- Also developing simple, stable and high performance solution with static mapped IPMMU
  - To specific purpose: Multimedia, 3D graphics, Image conposition etc.
  - Of cause it’s implemented as to be able to coexist with IOMMU

Renesas Electronics Corporation