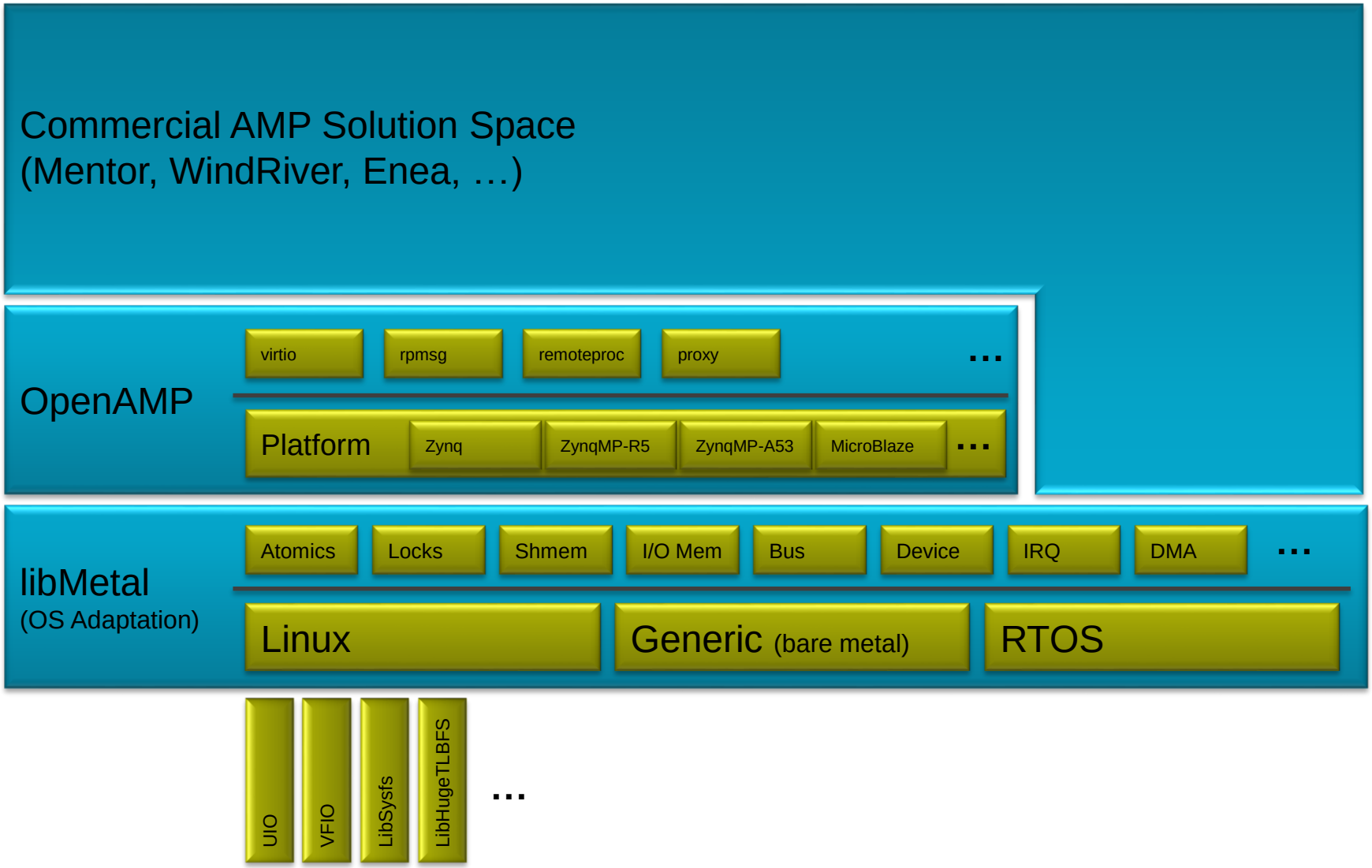
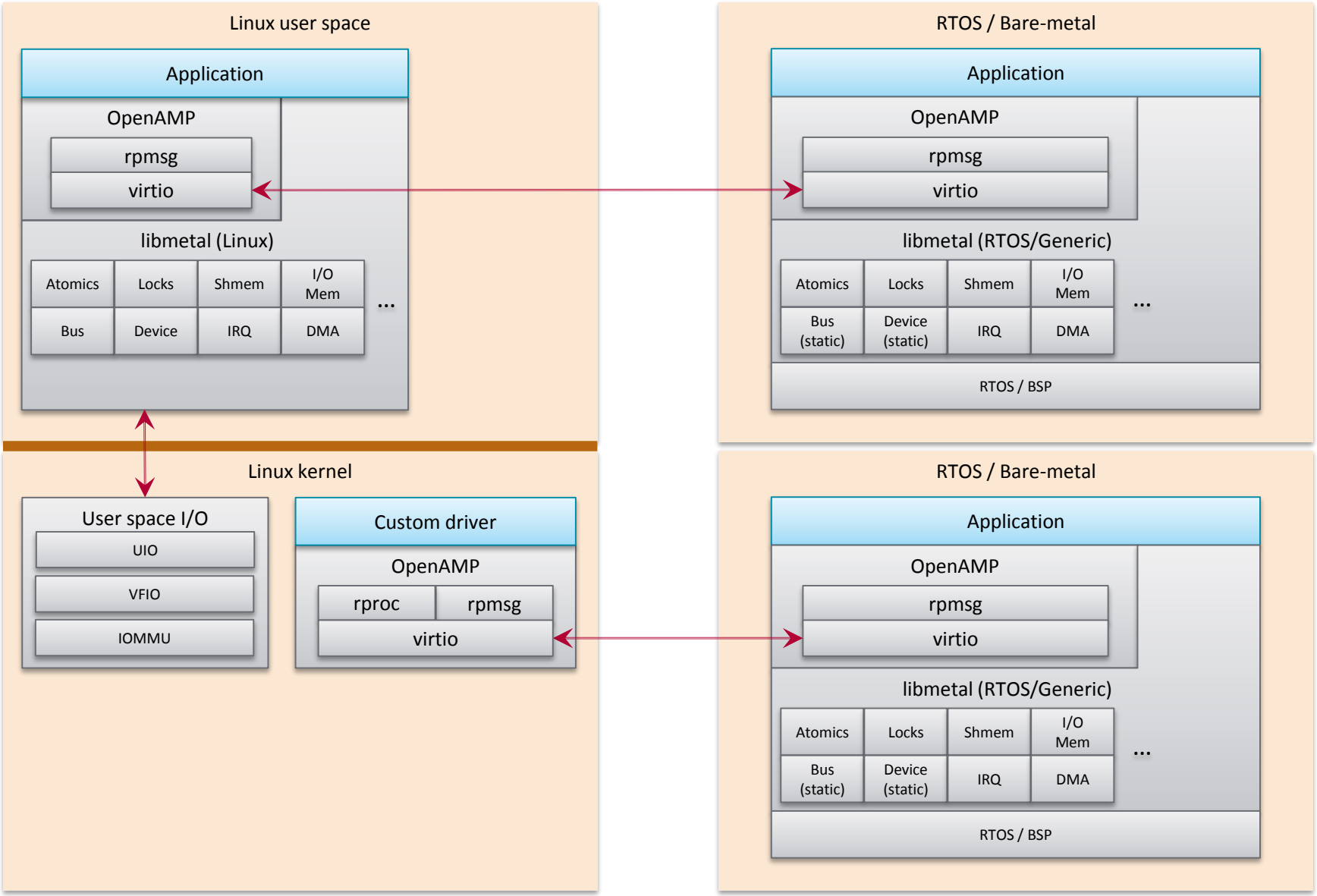
XILINX
ALL PROGRAMMABLE™

OpenAMP on top of Libmetal

OpenAMP on top of libmetal



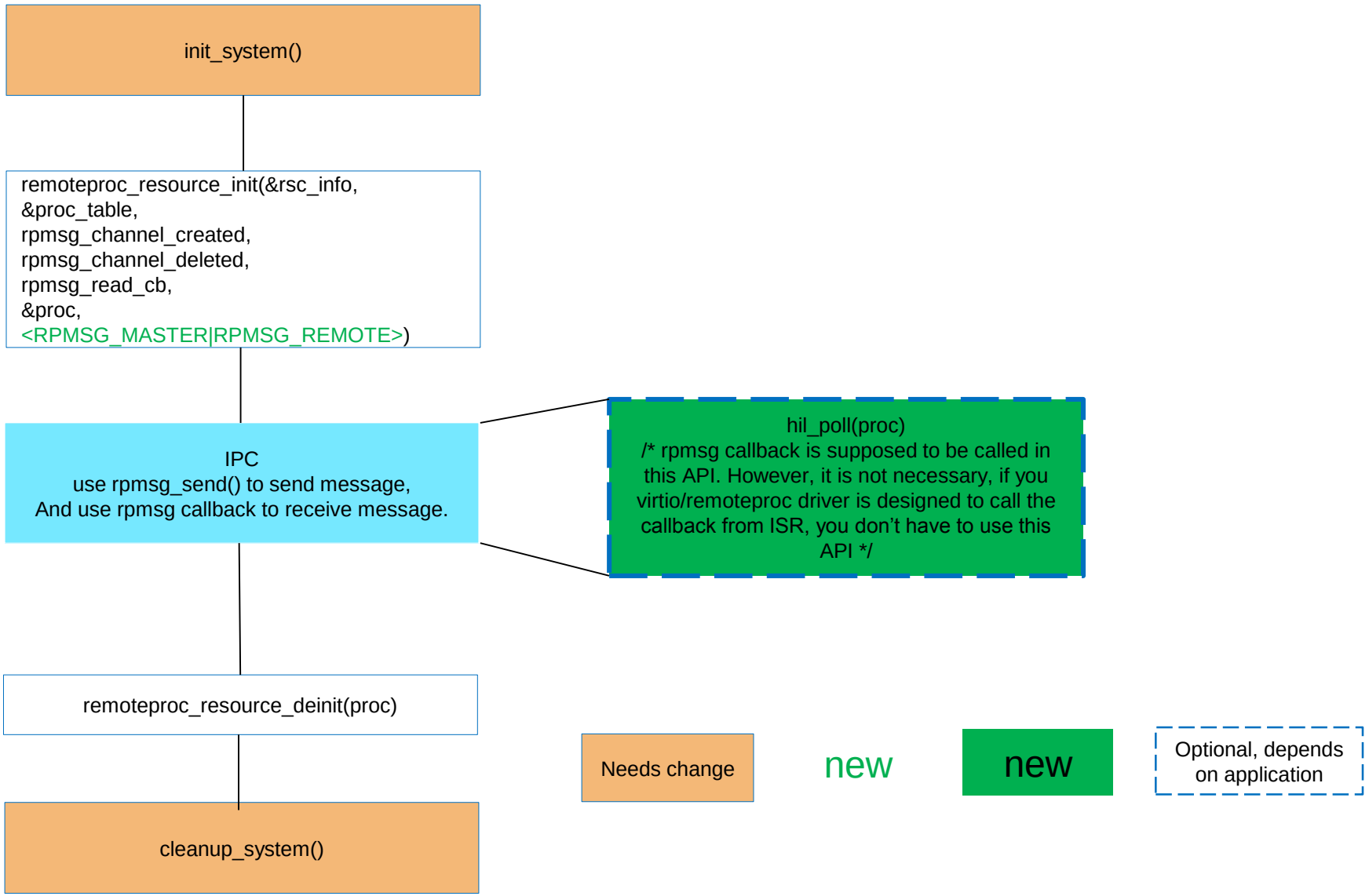
OpenAMP on Linux User Space



Source Code in Github

- libmetal: https://github.com/OpenAMP/libmetal/tree/master-rfc
- open-amp: https://github.com/OpenAMP/open-amp/tree/openamp-libmetal

OpenAMP Application Example Structure



OpenAMP Application Example Structure – system initialization

- Initialize application, system specifics
  - E.g. GIC
- Metal_init()
- Register IPI device and shared memory to libmetal
  - **This Step is for Baremetal/RTOS only, as they are specified in the device tree for Linux.**
  - IPI device, e.g. Xilinx Zynq Ultrascale+ MPSoC IPI
    - Base address, register range
  - Vring **device memory**
    - For RPMSG master for Baremetal/RTOS
      - Base address, size, set the I/O region memory flag as device memory
    - For RPMSG remote for Baremetal/RTOS
      - **Can use undefined address (0), and underdefined address range 0xffffffff, set the I/O region memory flag as device memory**
  - Shared memory
    - For RPMSG master for Baremetal/RTOS
      - Base address, size
    - For RPMSG master for baremetal
      - Can use undefined address (0), and underdefined address range 0xffffffff, set the I/O region memory flag as normal memory
  - E.g.: https://gitenterprise.xilinx.com/OpenAMP/open-amp/blob/xlnx-2016.3/apps/system/generic/machine/zynqmp_r5/sys_init.c

OpenAMP Application Example Structure – virtio/remoteproc driver platform data

- User will need to pass the IPI, vring and shared memory libmetal device and I/O region to the OpenAMP library
- hil_proc structure has been updated to include the libmetal device and I/O region information.

```
struct proc_shm {
        …
        /* sharmed memory I/O region */
        struct metal_io_region *io;
        …
};
```

```
struct proc_intr {
        …
        /* IPI metal device */
        struct metal_device *dev;
        /* IPI device I/O */
        struct metal_io_region *io;
        …
};
```

```
struct proc_vring {
        …
        /* Vring metal device */
        struct metal_device *dev;
        /* Vring I/O region */
        struct metal_io_region *io;
        …
};
```

- struct rproc_info_plat_local {} is introduced to provide another option for user to store the virtio/remoteproc driver platform data.

```
roc_data = {
        {
                PLAT_RSC_<TYPE>,
                data_field_N,
                …
        },
        …
};
```

OpenAMP Application Example Structure – cleanup_system

- metal_finish()
- Application, system specific cleanup

Changes to rpmsg

- Replace env mutex with libmetal mutex
- Replace llist with metal list (double linked list)
- Cache the cache if the shared memory is cacheable before it enqueues the buffer to virito.

Changes to virtio

- Memory barriers are replaced with atomic operations
- List of buffers are replaced with libmetal scatter list structure
- Conversion between virtual address and physical address are replaced with libmetal conversion between virtual address and physical address.

Changes to remoteproc

- Driver initialization and release
  - Introduce driver specific initialize() and release() API to allow each driver to have its own initialization and cleanup.
  - Introduce rproc_init_plat_data() to open the device and shared memory from libmetal
  - Introduce rproc_close_plat() to close device from libmetal
- poll()
  - Introduce an API for application to poll if it is kicked by the other end. And this poll() API is supposed to call the virtqueue_notification() which will then call the rpmsg callback. It is not mandatory to implement this API. You can still call the virqueue_notification (rpmsg callback) from ISR.
- Carve out memory and vring memory
  - Use libmetal memory map to enable access to these memory
- Interrupt handling
  - Register interrupt handler with libmetal metal_irq_register()

OpenAMP Env Layer Replacement with Libmetal

OpenAMP Env APIs	**Libmetal API**		
env_allocate_memory()	metal_allocate_memory()		
env_free_memory()	metal_free_memory()		
env_memset()	C memset() for normal memory, metal_memset_io() for device memory		
env_memcpy()	C memcpy() for normal memory, metal_memcpy_io() for device memory		
env_strlen()	C strlen()		
env_strcpy()	C strcpy()		
env_strncpy()	C strncpy()		
env_strcmp()	C strcmp()		
env_print()	removed, as not used		
env_assert()	removed, as not used		
env_map_vatopa()	metal_io_virt_to_phys(), it depends on libmetal I/O region		
env_map_patova()	metal_io_phys_to_virt(), it depends on libmetal I/O region		
env_mb(), env_rmb(), env_wmb()	C atomic, atomic_thread_fence()		
env mutex	libmetal mutex		
env_map_memory	metal_io_mem_map()		
env interrupt APIs	libmetal interrupt APIs		
env_get_timestamp()			
env_sleep_msec()			
Other helper functions			
llist	metal_list		

Future development

- Libmetal sleep and timestamp APIs
- VFIO support in libmetal
- Life cycle management
  - UIO/VFIO based implementation doesn't fit some use cases. E.g.:
    - Some systems want to use soft IRQs, there is no way to trigger soft IRQs from Linux userspace
    - Some systems have memory/registers protection.
  - Extend the kernel remoteproc to allow user space to have userspace rpmsg/virtio + kernel remoteproc for life cycle management