OpenAMP on top of Libmetal
OpenAMP on top of libmetal

Commercial AMP Solution Space
(Mentor, WindRiver, Enea, …)

OpenAMP
- virtio
- rpmsg
- remoteproc
- proxy
- …

Platform
- Zynq
- ZynqMP-R5
- ZynqMP-A53
- MicroBlaze
- …

libMetal
- Atomics
- Locks
- Shmem
- I/O Mem
- Bus
- Device
- IRQ
- DMA
- …

OS Adaptation
- Linux
- Generic (bare metal)
- RTOS
- …

UIO
- VFIO
- LibSys
- LibHuge TLBFS
- …

© Copyright 2016 Xilinx
OpenAMP on Linux User Space

Linux user space

- Application
- OpenAMP
  - rpmsg
  - virtio
- libmetal (Linux)
  - Atomics
  - Locks
  - Shmem
  - I/O Mem
  - Bus
  - Device
  - IRQ
  - DMA

RTOS / Bare-metal

- Application
- OpenAMP
  - rpmsg
  - virtio
- libmetal (RTOS/Generic)
  - Atomics
  - Locks
  - Shmem
  - I/O Mem
  - Bus (static)
  - Device (static)
  - IRQ
  - DMA

Linux kernel

- User space I/O
  - UIO
  - VFIO
  - IOMMU

Custom driver

- Application
- OpenAMP
  - rproc
  - rpmsg
  - virtio
- libmetal (Linux)
  - Atomics
  - Locks
  - Shmem
  - I/O Mem
  - Bus
  - Device
  - IRQ
  - DMA

RTOS / Bare-metal

- Application
- OpenAMP
  - rpmsg
  - virtio
- libmetal (RTOS/Generic)
  - Atomics
  - Locks
  - Shmem
  - I/O Mem
  - Bus (static)
  - Device (static)
  - IRQ
  - DMA
Source Code in Github

- **libmetal**: [https://github.com/OpenAMP/libmetal/tree/master-rfc](https://github.com/OpenAMP/libmetal/tree/master-rfc)
- **open-amp**: [https://github.com/OpenAMP/open-amp/tree/openamp-libmetal](https://github.com/OpenAMP/open-amp/tree/openamp-libmetal)
OpenAMP Application Example Structure

- `init_system()`
- `remoteproc_resource_init(&rsc_info, &proc_table, rpmsg_channel_created, rpmsg_channel_deleted, rpmsg_read_cb, &proc, <RPMSG_MASTER|RPMSG_REMOTE>)`
- `hil_poll(proc)` /* rpmsg callback is supposed to be called in this API. However, it is not necessary, if you virtio/remoteproc driver is designed to call the callback from ISR, you don’t have to use this API */
- `remoteproc_resource_deinit(proc)`
- `cleanup_system()`

**IPC**
- Use `rpmsg_send()` to send message,
- And use `rpmsg` callback to receive message.

- `needs change`
- `new`
- `new`

Optional, depends on application
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
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.

```c
struct proc_shm {
    ...
    /* shared memory I/O region */
    struct metal_io_region *io;
    ...
};
```

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

```c
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.

```c
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

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

© Copyright 2016 Xilinx
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