前言

基于Xilinx Zynq SoC / MPSoC的系统的常见要求之一是为特殊用途预留内存。预留的内存区域需要从linux内核的使用区域中分离出来，仅给特点的驱动程序使用。

reserved-memory 架构包含了预留内存的功能。预留内存的功能又与内核中的DMA-API和CMA框架密切相关。

本文旨在展示和解释一些可用的用例，并且已经使用Petalinux构建工具进行了测试。由于本文中的修改仅涉及DTS文件定制和设备驱动程序中分配内存位置的改动，还是可以将其导出到Yocto或OSL工作流程中的。

预留内存给设备驱动

为了从系统地址空间预留内存，设备树须配置预留内存的节点。

每个节点定义一个特定的内存空间，并且可以根据内核文档中关于可用于预留内存节点的说明配置不同的参数。

然后就可以通过memory-region参数将预留的内存空间分配给特定的设备驱动程序使用。

用于64位Cortex-A53 MPSoC的system-top.dts文件中的设备树节点：

```
rved-memory {
ess-cells = <2>;
-cells = <2>;
s;

ved: buffer@0 {
p;
 <0x0 0x70000000 0x0 0x10000000>;
 };


ved-driver@0 {
tible = "xlnx,reserved-memory";
y-region = <&reserved>;

```

或32位Cortex-A9 Zynq上，最新的基于Yocto的Petalinux的自定义的类似的设备树节点：

```
ude/ "system-conf.dtsi"

ved-memory {
ess-cells = <1>;
-cells = <1>;

```

```
s;

ved: buffer@0x38000000 {
p;
 <0x38000000 0x08000000>;
 };


ved-driver@0 {
tible = "xlnx,reserved-memory";
y-region = <&reserved>;
```

在设备驱动程序中，可以通过解析设备树节点来处理内存区域的属性，并且一旦知道了物理地址和大小，就可以使用memremap / ioremap调用来映射内存区域。

下面的代码使用了预留的内存分配：

```
t reserved memory region from Device-tree */
of_parse_phandle(dev->of_node, "memory-region", 0);
np) {
rr(dev, "No %s specified\n", "memory-region");
error1;


of_address_to_resource(np, 0, &r);
c) {
rr(dev, "No memory address assigned to the region\n");
error1;


addr = r.start;
addr = memremap(r.start, resource_size(&r), MEMREMAP_WB);
nfo(dev, "Allocated reserved memory, vaddr: 0x%0llX, paddr: 0x%0llX\n", (u64)lp->vaddr, lp->
```

由于保留的内存区域已被内核排除，并标记为no-map，因此iomem信息（/ proc / iomem）显示系统RAM小于主板中的内存量。

```
plnx_aarch64:~# cat /proc/iomem
000-6fffffff : System RAM
```

```
000-00b37fff : Kernel code
000-012b8fff : Kernel data
```

加载设备后，可以确认内存分配：

```
6.191774] reserved-memory reserved-driver@0: Device Tree Probing
6.198595] reserved-memory reserved-driver@0: Allocated reserved memory, vaddr: 0xFFFFFF802000
```

通过DMA API预留内存

有的时候设备驱动程序需要采用DMA的方式使用预留的内存，对于这种场景，可以dts中的节点属性设置为shared-dma-pool，从而生成为特定设备驱动程序预留的DMA内存池。

```
ved-memory {
ess-cells = <2>;
-cells = <2>;
s;

ved: buffer@0 {
tible = "shared-dma-pool";
p;
 <0x0 0x70000000 0x0 0x10000000>;
 };


ved-driver@0 {
tible = "xlnx,reserved-memory";
y-region = <&reserved>;

```

这样，设备驱动程序仅需要以常规方式使用DMA API，但无需使用默认的CMA内存池，它将使用该特定设备的预留内存区域。

（独占这一块内存，仍然使用DMA的接口）

```
itialize reserved memory resources */
of_reserved_mem_device_init(dev);
```

```
) {
rr(dev, "Could not get reserved memory\n");
error1;


locate memory */
et_coherent_mask(dev, 0xFFFFFFFF);
addr = dma_alloc_coherent(dev, ALLOC_SIZE, &lp->paddr, GFP_KERNEL);
nfo(dev, "Allocated coherent memory, vaddr: 0x%0llX, paddr: 0x%0llX\n", (u64)lp->vaddr, lp->
```

内核启动的log:

```
0.000000] Reserved memory: created DMA memory pool at 0x0000000070000000, size 256 MiB
0.000000] Reserved memory: initialized node buffer@0, compatible id shared-dma-pool
0.000000] cma: Reserved 128 MiB at 0x0000000068000000

```

驱动加载的log:

```
plnx_aarch64:~# insmod /lib/modules/4.6.0-xilinx/extra/reserved-memory.ko
0.745166] reserved-memory reserved-driver@0: Device Tree Probing
0.750183] reserved-memory reserved-driver@0: assigned reserved memory node buffer@0
1.220878] reserved-memory reserved-driver@0: Allocated coherent memory, vaddr: 0xFFFFFF802000
```

-
- 1
- 2
- 3
- 4
- 5
-

给CMA预留内存

有时，不需要将预留的内存分配给特定的设备驱动程序，而只打算给CMA内存池分配一块固定的内存区域。 对于这种场景，可以使用一个额外的属性来指向内核，以将预留的内存区域用作默认的CMA内存池。

属性:

reusable;

linux,cma-default

```
ved-memory {
ess-cells = <2>;
-cells = <2>;
s;

ved: buffer@0 {
tible = "shared-dma-pool";
ble;
 <0x0 0x70000000 0x0 0x10000000>;
,cma-default;
 };

```

内核启动关于CMA分配的log：

```
[    0.000000] Reserved memory: created CMA memory pool at 0x0000000070000000, size 256 MiB
[    0.000000] Reserved memory: initialized node buffer@0, compatible id shared-dma-pool
```