Rockchip Developer Guide Linux GMAC DPDK

文件标识：RK-KF-YF-487

发布版本：V1.1.0

日期：2024-07-22

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文提供 Rockchip 平台 GMAC 接口的以太网 DPDK 介绍使用文档。

产品版本

芯片名称	内核版本
RK3568、RK3588、RK3576	4.19+

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	吴达超	2023-03-06	初始版本
V1.1.0	吴达超	2024-07-22	添加常见问题

目录

Rockchip Developer Guide Linux GMAC DPDK
- 代码编译
  - 内核
  - DPDK 编译
- 运行 DPDK 程序
  - 挂载巨页
  - 加载 KO
  - 设置 performance 模式
  - 运行 testpmd
  - 运行 l2fwd
  - 运行 l3fwd
- Pktgen
  - 下载 pktgen-dpdk 源码
  - DPDK 编译
  - Pktgen 编译
  - 运行 Pktgen 程序
  - 常见性能问题：
    - 硬件功能正确
    - 设置 performance 模式
    - 长时间打流会丢包
    - 物理内存超4G空间
    - 编译选项

代码编译

内核

- 首先使能 DTS 中 UIO 节点，以 RK3568-evb1 参考：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
index 0cb57e9d8529..c7729258e51d 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
@@ -262,6 +262,14 @@
        status = "okay";
 };
```

```
+&gmac_uio0 {
+       status = "okay";
+};
+
+&gmac_uio1 {
+       status = "okay";
+};
+
 /*
  * power-supply should switche to vcc3v3_lcd1_n
  * when mipi panel is connected to dsi1.
```

- 编译 KO

```
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 rockchip_linux_defconfig
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 menuconfig, 配置 CONFIG_UIO=m,
CONFIG_STMMAC_UIO=m, CONFIG_HUGETLBFS=y
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 rrk3568-evb1-ddr4-v10-
linux.img -j8
烧写 boot.img
adb push drivers/uio/uio.ko, adb push
drivers/net/ethernet/stmicro/stmmac/stmmac_uio.ko
```

DPDK 编译

DPDK 测试使用的开发板跑的系统是 Debian 11，DPDK 版本 21.11，编译参考 http://doc.dpdk.org/guides-21.11/linux_gsg/cross_build_dpdk_for_arm64.html

- PC 交叉编译

```
wget https://developer.arm.com/-/media/Files/downloads/gnu-a/9.2-
2019.12/binrel/gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz
tar -xvf gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz
export PATH=$PATH:<cross_install_dir>/gcc-arm-9.2-2019.12-x86_64-aarch64-none-
linux-gnu/bin
同时修改 config/arm/arm64_armv8_linux_gcc

apt install build-essential
apt install pkg-config-aarch64-linux-gnu
apt-get install python3 python3-pip
apt install meson
apt install ninja-build

meson aarch64-build-gcc --cross-file config/arm/arm64_armv8_linux_gcc
meson --reconfigure aarch64-build-gcc --cross-file
config/arm/arm64_armv8_linux_gcc -Dplatform=arm64 -Dexamples=l2fwd,l3fwd
ninja -C aarch64-build-gcc
```

- 开发板编译 DPDK

```
apt-get install python3 python3-pip
apt install meson
apt install ninja-build
apt-get install libdpkg-perl
apt-get install build-essential
apt-get install python3-pyelftools

meson -Ddisable_drivers='common/cnxk' build
cd build
ninja
ninja install
```

运行 DPDK 程序

挂载巨页

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

加载 KO

```
insmod uio.ko
insmod stmmac_uio.ko
```

默认开机起来后的网络走的还是原生内核网络，这里我们通过 insmod stmmac_uio.ko 和 rmmod 该 ko 来做到 DPDK 与内核原生网络的切换，即卸载 KO 后，会回到原来的内核网络状态。

- 进入 DPDK 网络控制：

```
root@linaro-alip:/home/linaro#insmod stmmac_uio.ko
[   41.232761] Generic PHY stmmac-1:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-1:00, irq=POLL)
[   41.236447] dwmac4: Master AXI performs any burst length
[   41.236497] rk_gmac-dwmac fe010000.ethernet eth0: No Safety Features support
found
[   41.236886] rockchip_eth_uio_drv fe010000.uio: Registered uio_eth0 uio
devices, 3 register maps attached
[   41.241453] Generic PHY stmmac-0:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[   41.257084] dwmac4: Master AXI performs any burst length
[   41.257138] rk_gmac-dwmac fe2a0000.ethernet eth1: No Safety Features support
found
[   41.257523] rockchip_eth_uio_drv fe2a0000.uio: Registered uio_eth1 uio
devices, 3 register maps attached
```

- 返回内核网络控制：

```
root@linaro-alip:/home/linaro# rmmod stmmac_uio
[ 2200.304636] Generic PHY stmmac-0:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[ 2200.318163] dwmac4: Master AXI performs any burst length
[ 2200.318205] rk_gmac-dwmac fe2a0000.ethernet eth1: No Safety Features support
found
[ 2200.318227] rk_gmac-dwmac fe2a0000.ethernet eth1: IEEE 1588-2008 Advanced
Timestamp supported
```

```
[ 2200.318443] rk_gmac-dwmac fe2a0000.ethernet eth1: registered PTP clock
[ 2200.319414] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[ 2200.322971] Generic PHY stmmac-1:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-1:00, irq=POLL)
[ 2200.324883] dwmac4: Master AXI performs any burst length
[ 2200.324921] rk_gmac-dwmac fe010000.ethernet eth0: No Safety Features support
found
[ 2200.324945] rk_gmac-dwmac fe010000.ethernet eth0: IEEE 1588-2008 Advanced
Timestamp supported
[ 2200.325468] rk_gmac-dwmac fe010000.ethernet eth0: registered PTP clock
[ 2200.325814] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 2205.385406] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

设置 performance 模式

```
echo performance | tee $(find /sys/ -name *governor) /dev/null || true
```

运行 testpmd

- 转发模式测试命令
  - --vdev=net_stmmac0 --vdev=net_stmmac1 表示指定的虚拟设备，目前是名字是固定的
  - --main-lcore=0 表示 0 核用作管理，2 和 3 核用于转发
  - --iova-mode=pa 因为设备不支持 iommu，故 iova-mode 规定为 pa 模式
  - -- 用于分隔 eal 参数与 testpmd 的参数
  - -i 表示进入 dpdk-testpmd 命令交互模式

```
root@linaro-alip:/home/linaro# ./dpdk-testpmd  --iova-mode=pa --
vdev=net_stmmac0 --vdev=net_stmmac1 -l 0,2,3 --main-lcore=0 -- -i
EAL: Detected CPU lcores: 4
EAL: Detected NUMA nodes: 1
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
TELEMETRY: No legacy callbacks, legacy socket not created
Interactive-mode selected
Warning: NUMA should be configured manually by using --port-numa-config and --
ring-numa-config parameters along with --numa.
testpmd: create a new mbuf pool <mb_pool_0>: n=163456, size=2176, socket=0
testpmd: preferred mempool ops selected: ring_mp_mc
Configuring Port 0 (socket 0)
stmmac_net: stmmac_eth_link_update()Port (0) link is Up
Port 0: BA:A0:3F:FD:B2:8C
Configuring Port 1 (socket 0)
stmmac_net: stmmac_eth_link_update()Port (1) link is Up
Port 1: B6:A0:3F:FD:B2:8C
Checking link statuses...
stmmac_net: stmmac_eth_link_update()Port (0) link is Up
stmmac_net: stmmac_eth_link_update()Port (1) link is Up
Done
```

- 开启 testpmd 64 字节转发模式测试

```
testpmd> start
```

```
io packet forwarding - ports=2 - cores=1 - streams=2 - NUMA support enabled, MP
allocation mode: native
Logical Core 2 (socket 0) forwards packets on 2 streams:
  RX P=0/Q=0 (socket 0) -> TX P=1/Q=0 (socket 0) peer=02:00:00:00:00:01
  RX P=1/Q=0 (socket 0) -> TX P=0/Q=0 (socket 0) peer=02:00:00:00:00:00

  io packet forwarding packets/burst=32
  nb forwarding cores=1 - nb forwarding ports=2
  port 0: RX queue number: 1 Tx queue number: 1
    Rx offloads=0x0 Tx offloads=0x0
    RX queue: 0
      RX desc=0 - RX free threshold=0
      RX threshold registers: pthresh=0 hthresh=0  wthresh=0
      RX Offloads=0x0
    TX queue: 0
      TX desc=0 - TX free threshold=0
      TX threshold registers: pthresh=0 hthresh=0  wthresh=0
      TX offloads=0x0 - TX RS bit threshold=0
  port 1: RX queue number: 1 Tx queue number: 1
    Rx offloads=0x0 Tx offloads=0x0
    RX queue: 0
      RX desc=0 - RX free threshold=0
      RX threshold registers: pthresh=0 hthresh=0  wthresh=0
      RX Offloads=0x0
    TX queue: 0
      TX desc=0 - TX free threshold=0
      TX threshold registers: pthresh=0 hthresh=0  wthresh=0
      TX offloads=0x0 - TX RS bit threshold=0
```

- 查看转发数据:

```
 testpmd> show port stats all
  ######################## NIC statistics for port 0  ########################
  RX-packets: 86276096   RX-missed: 0          RX-bytes:  5176569365
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 0          TX-errors: 0          TX-bytes:  0

  Throughput (since last show)
  Rx-pps:       1125857          Rx-bps:    576438784
  Tx-pps:             0          Tx-bps:            0
  ############################################################################

  ######################## NIC statistics for port 1  ########################
  RX-packets: 0          RX-missed: 0          RX-bytes:  0
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 46621099   TX-errors: 0          TX-bytes:  2797265940

  Throughput (since last show)
  Rx-pps:             0          Rx-bps:            0
  Tx-pps:       1124867          Tx-bps:    575931904
  ############################################################################
```

- 多核设置
  比如双向转发，需要用到2个核(2和3)，一个核一个方向:

```
set corelist 2,3
```

- 设置多 port 转发:
  比如 prot0 和 port2 转发，port1 和 port3 转发

```
set portlist 0,2,1,3
```

- 其它常用设置:
  - --nb-cores 表示指定 dpdk-testpmd 用作转发工作的核的数量
  - --rxd 接收队列描述符
  - --txd 发送队列描述符
  - --rxq 表示指定 dpdk-testpmd 接收队列数，RK3568 队列数为1
  - --txq 表示指定 dpdk-testpmd 发送队列数，RK3568 队列数为1

运行 l2fwd

l2fwd 默认至少要有两个核才能测试转发

```
./dpdk-l2fwd -l 0,2,3  --main-lcore=0 --iova-mode=pa --vdev=net_stmmac0 --
vdev=net_stmmac1 -- -q 1 -p 0x3
```

l2fwd 运行的串口信息每隔10s钟会刷新一次，考虑可能会导致丢包，建议将串口信息重定向到文件。

运行 l3fwd

```
./dpdk-l3fwd -l 3 -n 1 --iova-mode=pa --vdev=net_stmmac0 --vdev=net_stmmac1 --
-p 0x3 -P --config="(0,0,3),(1,0,3)" --parse-ptype
```

- -p PortMask 参数指定使用的网口掩码
- -P 参数表示将所有网口设置为混杂模式，以便收到所有数据包
- --config (port,queue,lcore)，[(port,queue,lcore)] 参数用以配置网口、队列、核之间的对应关系，例如，--config (0,0,3) 表示网口 0 的队列 0 由核 3 进行处理

值得注意的是，上述输出中打印了 l3fwd 的默认路由规则，即

```
LPM: Adding route 198.18.0.0 / 24 (0) [net_stmmac0]
LPM: Adding route 198.18.1.0 / 24 (1) [net_stmmac1]
LPM: Adding route 2001:200:: / 64 (0) [net_stmmac0]
LPM: Adding route 2001:200:0:1:: / 64 (1) [net_stmmac1]
```

也就是说，目的 IP 为 198.18.0.0/24 段的数据包将会通过网口 0 进行转发，目的 IP 为 198.18.1.0 / 24 段的数据包将会通过网口 1 进行转发。上述默认路由规则是在源码中配置的，所以在l3fwd测试的时候，需要设置好测试数据的目标ip和源ip。

Pktgen

在板子上跑Pktgen ，是基于 dpdk，所以需要先编译和安装好 DPDK，编译参考1.2章节的开发板编译 DPDK。

下载 pktgen-dpdk 源码

```
git clone http://dpdk.org/git/apps/pktgen-dpdk
apt-get install libpcap-dev
apt-get install libnuma-dev
apt install meson
apt install ninja-build pkg-config
```

DPDK 编译

```
cd build
ninja
ninja install
ldconfig
export PKG_CONFIG_PATH=/usr/local/lib/aarch64-linux-gnu/pkgconfig/
```

Pktgen 编译

```
cd pktgen-dpdk
meson build
cd build
ninja
```

运行 Pktgen 程序

```
./build/app/pktgen --iova-mode=pa --vdev=net_stmmac0 -l 6,7 --proc-type auto --
log-level debug -- -P -m 7.0
```

其中，EAL options 参数部分可以参看 DPDK EAL parameters，最重要的一个参数就是 -l 参数，用它来指定使用的核列表，比如：-l 1,2 或者 -l 1-2，表示使用核 1 和核 2。

值得注意的是，pktgen 至少要指定两个核，因为 pktgen 需要一个核与用户进行交互，比如响应测试过程中用户的输入。
pktgen 自有参数部分最重要的是 -m 参数，用它来指定网口与核之间的对应关系，比如：
-m 2.0：表示让核 2 来处理网口 0。值得注意的是，若要指定多个对应关系（使用多个网卡和多个核），则需多次使用 -m 参数。
如果要收包，最好也指定一下 -P 参数，表示让所有网口进入混杂模式，以便接收到所有数据包。

设置数据包格式并开启 Pktgen：

```
set 0 size 64
set 0 src ip 198.18.0.100/24
set 0 dst ip 198.18.1.101
set 0 dst mac ba:a0:3f:fd:b2:8c
start 0
```

常见性能问题：

硬件功能正确

首先要先确保硬件功能，信号这些是正确的，不能出现硬件上的丢包，出现丢包一样会影响性能，这里可以简单地先通过内核的以太网网卡进行 iperf 测试（达到930Mbits/s），不需要在dpdk上测试。常见的硬件问题：

- tx/rx delayline 设置不正确，如何设置 delayline，参考文档《Rockchip_Developer_Guide_Linux_GMAC_RGMII_Delayline_CN.pdf》
- io_domain 设置不正确，有些芯片有 io_domain 配置项，比如RK3568；这里一定要注意，如果设置不对，iperf 测试也可能是正常的，但潜在风险是会损坏芯片 IO；如何设置，参考文档《Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf》

设置 performance 模式

```
echo performance | tee $(find /sys/ -name *governor) /dev/null || true
```

长时间打流会丢包

- 设置核隔离
  在 cmdline 加入隔离的核，比如隔离核2和3，加入 isolcpus=2,3

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
index c7e309645099..cf88ad29dcf6 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
@@ -13,7 +13,7 @@ aliases {
        };

        chosen: chosen {
-               bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 rw rootwait";
+               bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 isolcpus=2,3 rw rootwait";
        };

        fiq-debugger {
```

是否修改成功通过 “cat /sys/devices/system/cpu/isolated” 来确认。

- 设置 no full_hz

确保一下编译选项打开，编译内核

```
+CONFIG_NO_HZ_FULL=y
```

在 cmdline 加入对应隔离核，比如核2 和 3，nohz_full=2,3 ；如果是RK3588，使用大核作为隔离核：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
index c7e309645099..cf88ad29dcf6 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
@@ -13,7 +13,7 @@ aliases {
        };

        chosen: chosen {
-               bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 rw rootwait";
+               bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 isolcpus=2,3 nohz_full=2,3 rw rootwait";
        };

        fiq-debugger {
```

是否修改成功通过 “cat /proc/cmdline” 来确认。

物理内存超4G空间

因为 RK3568 和 RK3588 的 GMAC 只能支持4G以下的物理内存空间，需要对 dpdk的 memory 使用做限制，否则有可能出现异常

uboot 的 log 先确认是否超 4G(0x00000000 - 0xffffffff)
例如下面这个log显示，大概有256M在4G以上：

```
Adding bank: 0x00200000 - 0x08400000 (size: 0x08200000)
Adding bank: 0x09400000 - 0xf0000000 (size: 0xe6c00000)
Adding bank: 0x1f0000000 - 0x200000000 (size: 0x10000000)
```

可以简单的修改 uboot 的代码限制在 4G 以下，测试是否会影响了性能：

```
diff --git a/arch/arm/mach-rockchip/param.c b/arch/arm/mach-rockchip/param.c
index 21a45705f0..b9895bae2a 100644
--- a/arch/arm/mach-rockchip/param.c
+++ b/arch/arm/mach-rockchip/param.c
@@ -287,10 +287,12 @@ struct memblock *param_parse_ddr_mem(int *out_count)
            return 0;
        }

+       count = 1;
        for (i = 0, n = 0; i < count; i++, n++) {
            base = t->u.ddr_mem.bank[i];
            size = t->u.ddr_mem.bank[i + count];

+           size = SZ_4GB;
            /* 0~4GB */
            if (base < SZ_4GB) {
                mem[n].base = base;
```

但是这个修改会导致系统内存变小，正式补丁：

- 内核是4.19，打上补丁：Linux4.19_under_4G.tar.gz
- 内核是5.10及以上版本，打上补丁：Linux5.10-mm-hugetlb-limit-hugetlb-page-under-4G.patch

打上补丁后，需要修改对应dts的 cmdline 参数，指定 hugapage 参数，比如下面修改是在0x40000000位置预留了 256M 给 dpdk 的 hugepage 使用（大小以及起始位置可以根据实际情况自行修改，但得是4G以下且是连续的）：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
index c7e309645099..3ed48d5c5ac8 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
@@ -13,7 +13,7 @@
        };

        chosen: chosen {
-               bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 rw rootwait";
+               bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 isolcpus=2,3 nohz_full=2,3 default_hugepagesz=32M
hugetlb_cma=256M@0x40000000 rw rootwait";
        };

        fiq-debugger {
```

是否修改成功通过 “cat /proc/cmdline” 来确认。
因为已经指定了32M的hugepage，启动dpdk之前的挂载巨页就要使用32M的hugepage：

```
echo 8 > /sys/kernel/mm/hugepages/hugepages-32768kB/nr_hugepages  //分配8个32页
```

编译选项

DPDK 编译选项如果带上 -Dbuildtype=debug，会影响性能，需要删除。

另外如果编译到 cnxk 报错，可以加上编译选项: -Ddisable_drivers='common/cnxk'。

重新配置编译选项需要带上 --reconfigure。