The QorIQ DPAA Ethernet Driver

Authors: - Madalin Bucur <madalin.bucur@nxp.com> - Camelia Groza <camelia.groza@nxp.com>

DPAA Ethernet Overview

DPAA stands for Data Path Acceleration Architecture and it is a set of networking acceleration IPs that are available on several generations of SoCs, both on PowerPC and ARM64.

The Freescale DPAA architecture consists of a series of hardware blocks that support Ethernet connectivity. The Ethernet driver depends upon the following drivers in the Linux kernel:

- Peripheral Access Memory Unit (PAMU) (* needed only for PPC platforms)
  drivers/iommu/fsl_*
- Frame Manager (FMan)
  drivers/net/ethernet/freescale/fman
- Queue Manager (QMan), Buffer Manager (BMan)
  drivers/soc/fsl/qbman

A simplified view of the dpaa_eth interfaces mapped to FMan MACs:

```
dpaa_eth       /eth0\     ...       /ethN\
driver        |      |             |      |
-------------   ----   -----------   ----   -------------
     -Ports  / Tx  Rx \    ...    / Tx  Rx \
FMan        |          |         |          |
     -MACs  |   MAC0   |         |   MACN   |
           /   dtsec0   \  ...  /   dtsecN   \ (or tgec)
          /              \     /              \(or memac)
---------  --------------  ---  --------------  ---------
    FMan, FMan Port, FMan SP, FMan MURAM drivers
----------------------------------------------------------
    FMan HW blocks: MURAM, MACs, Ports, SP
----------------------------------------------------------
```

The dpaa_eth relation to the QMan, BMan and FMan:

```
                 ________________________________
dpaa_eth   /            eth0                \
driver    /                                   \
---------   -^-   -^-   -^-   ---    ---------
QMan driver / \   / \   / \  \   /   | BMan    |
           |Rx | |Rx | |Tx | |Tx |   | driver  |
---------  |Dfl| |Err| |Cnf| |FQs|   |         |
QMan HW    |FQ | |FQ | |FQs| |   |   |         |
           /   \ /   \ /   \  \ /    |         |
---------   ---   ---   ---   -v-    ---------
          |        FMan QMI          |         |
          | FMan HW       FMan BMI   | BMan HW |
            ----------------------    --------
```

where the acronyms used above (and in the code) are:

DPAA	Data Path Acceleration Architecture
FMan	DPAA Frame Manager
QMan	DPAA Queue Manager
BMan	DPAA Buffers Manager
QMI	QMan interface in FMan
BMI	BMan interface in FMan
FMan SP	FMan Storage Profiles
MURAM	Multi-user RAM in FMan
FQ	QMan Frame Queue
Rx Dfl FQ	default reception FQ
Rx Err FQ	Rx error frames FQ
Tx Cnf FQ	Tx confirmation FQs
Tx FQs	transmission frame queues
dtsec	datapath three speed Ethernet controller (10/100/1000 Mbps)
tgec	ten gigabit Ethernet controller (10 Gbps)
memac	multirate Ethernet MAC (10/100/1000/10000)

DPAA Ethernet Supported SoCs

The DPAA drivers enable the Ethernet controllers present on the following SoCs:

PPC - P1023 - P2041 - P3041 - P4080 - P5020 - P5040 - T1023 - T1024 - T1040 - T1042 - T2080 - T4240 - B4860

ARM - LS1043A - LS1046A

Configuring DPAA Ethernet in your kernel

To enable the DPAA Ethernet driver, the following Kconfig options are required:

```
# common for arch/arm64 and arch/powerpc platforms
CONFIG_FSL_DPAA=y
CONFIG_FSL_FMAN=y
CONFIG_FSL_DPAA_ETH=y
CONFIG_FSL_XGMAC_MDIO=y

# for arch/powerpc only
CONFIG_FSL_PAMU=y

# common options needed for the PHYs used on the RDBs
CONFIG_VITESSE_PHY=y
CONFIG_REALTEK_PHY=y
CONFIG_AQUANTIA_PHY=y
```

DPAA Ethernet Frame Processing

On Rx, buffers for the incoming frames are retrieved from the buffers found in the dedicated interface buffer pool. The driver initializes and seeds these with one page buffers.

On Tx, all transmitted frames are returned to the driver through Tx confirmation frame queues. The driver is then responsible for freeing the buffers. In order to do this properly, a backpointer is added to the buffer before transmission that points to the skb. When the buffer returns to the driver on a confirmation FQ, the skb can be correctly consumed.

DPAA Ethernet Features

Currently the DPAA Ethernet driver enables the basic features required for a Linux Ethernet driver. The support for advanced features will be added gradually.

The driver has Rx and Tx checksum offloading for UDP and TCP. Currently the Rx checksum offload feature is enabled by default and cannot be controlled through ethtool. Also, rx-flow-hash and rx-hashing was added. The addition of RSS provides a big performance boost for the forwarding scenarios, allowing different traffic flows received by one interface to be processed by different CPUs in parallel.

The driver has support for multiple prioritized Tx traffic classes. Priorities range from 0 (lowest) to 3 (highest). These are mapped to HW workqueues with strict priority levels. Each traffic class contains NR_CPU TX queues. By default, only one traffic class is enabled and the lowest priority Tx queues are used. Higher priority traffic classes can be enabled with the mqprio qdisc. For example, all four traffic classes are enabled on an interface with the following command. Furthermore, skb priority levels are mapped to traffic classes as follows:

- priorities 0 to 3 - traffic class 0 (low priority)
- priorities 4 to 7 - traffic class 1 (medium-low priority)
- priorities 8 to 11 - traffic class 2 (medium-high priority)
- priorities 12 to 15 - traffic class 3 (high priority)

```
tc qdisc add dev <int> root handle 1: \
       mqprio num_tc 4 map 0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3 hw 1
```

DPAA IRQ Affinity and Receive Side Scaling

Traffic coming on the DPAA Rx queues or on the DPAA Tx confirmation queues is seen by the CPU as ingress traffic on a certain portal. The DPAA QMan portal interrupts are affined each to a certain CPU. The same portal interrupt services all the QMan portal consumers.

By default the DPAA Ethernet driver enables RSS, making use of the DPAA FMan Parser and Keygen blocks to distribute traffic on 128 hardware frame queues using a hash on IP v4/v6 source and destination and L4 source and destination ports, in present in the received frame. When RSS is disabled, all traffic received by a certain interface is received on the default Rx frame queue. The default DPAA Rx frame queues are configured to put the received traffic into a pool channel that allows any available CPU portal to dequeue the ingress traffic. The default frame queues have the HOLDACTIVE option set, ensuring that traffic bursts from a certain queue are serviced by the same CPU. This ensures a very low rate of frame reordering. A drawback of this is that only one CPU at a time can service the traffic received by a certain interface when RSS is not enabled.

To implement RSS, the DPAA Ethernet driver allocates an extra set of 128 Rx frame queues that are configured to dedicated channels, in a round-robin manner. The mapping of the frame queues to CPUs is now hardcoded, there is no indirection table to move traffic for a certain FQ (hash result) to another CPU. The ingress traffic arriving on one of these frame queues will arrive at the same portal and will always be processed by the same CPU. This ensures intra-flow order preservation and workload distribution for multiple traffic flows.

RSS can be turned off for a certain interface using ethtool, i.e.:

```
# ethtool -N fm1-mac9 rx-flow-hash tcp4 ""
```

To turn it back on, one needs to set rx-flow-hash for tcp4/6 or udp4/6:

```
# ethtool -N fm1-mac9 rx-flow-hash udp4 sfdn
```

There is no independent control for individual protocols, any command run for one of tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6 is going to control the rx-flow-hashing for all protocols on that interface.

Besides using the FMan Keygen computed hash for spreading traffic on the 128 Rx FQs, the DPAA Ethernet driver also sets the skb hash value when the NETIF_F_RXHASH feature is on (active by default). This can be turned on or off through ethtool, i.e.:

```
# ethtool -K fm1-mac9 rx-hashing off
# ethtool -k fm1-mac9 | grep hash
receive-hashing: off
# ethtool -K fm1-mac9 rx-hashing on
Actual changes:
receive-hashing: on
# ethtool -k fm1-mac9 | grep hash
receive-hashing: on
```

Please note that Rx hashing depends upon the rx-flow-hashing being on for that interface - turning off rx-flow-hashing will also disable the rx-hashing (without ethtool reporting it as off as that depends on the NETIF_F_RXHASH feature flag).

Debugging

The following statistics are exported for each interface through ethtool:

- interrupt count per CPU
- Rx packets count per CPU
- Tx packets count per CPU
- Tx confirmed packets count per CPU
- Tx S/G frames count per CPU
- Tx error count per CPU
- Rx error count per CPU
- Rx error count per type
- congestion related statistics:
    - congestion status
    - time spent in congestion
    - number of time the device entered congestion
    - dropped packets count per cause

The driver also exports the following information in sysfs:

- the FQ IDs for each FQ type /sys/devices/platform/soc/<addr>.fman/<addr>.ethernet/dpaa-ethernet.<id>/net/fm<nr>-mac<nr>/fqids
- the ID of the buffer pool in use /sys/devices/platform/soc/<addr>.fman/<addr>.ethernet/dpaa-ethernet.<id>/net/fm<nr>-mac<nr>/bpids