Texas Instruments CPSW switchdev based ethernet driver

Version: 2.0

Port renaming

On older udev versions renaming of ethX to swXpY will not be automatically supported

In order to rename via udev:

```
ip -d link show dev sw0p1 | grep switchid

SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}==<switchid>, \
        ATTR{phys_port_name}!="", NAME="sw0$attr{phys_port_name}"
```

Dual mac mode

- The new (cpsw_new.c) driver is operating in dual-emac mode by default, thus working as 2 individual network interfaces. Main differences from legacy CPSW driver are:
  - optimized promiscuous mode: The P0_UNI_FLOOD (both ports) is enabled in addition to ALLMULTI (current port) instead of ALE_BYPASS. So, Ports in promiscuous mode will keep possibility of mcast and vlan filtering, which is provides significant benefits when ports are joined to the same bridge, but without enabling "switch" mode, or to different bridges.
  - learning disabled on ports as it make not too much sense for segregated ports - no forwarding in HW.
  - enabled basic support for devlink.

```
devlink dev show
        platform/48484000.switch

devlink dev param show
platform/48484000.switch:
name switch_mode type driver-specific
values:
        cmode runtime value false
name ale_bypass type driver-specific
values:
        cmode runtime value false
```

Devlink configuration parameters

See Documentation/networking/devlink/ti-cpsw-switch.rst

Bridging in dual mac mode

The dual_mac mode requires two vids to be reserved for internal purposes, which, by default, equal CPSW Port numbers. As result, bridge has to be configured in vlan unaware mode or default_pvid has to be adjusted:

```
ip link add name br0 type bridge
ip link set dev br0 type bridge vlan_filtering 0
echo 0 > /sys/class/net/br0/bridge/default_pvid
ip link set dev sw0p1 master br0
ip link set dev sw0p2 master br0
```

or:

```
ip link add name br0 type bridge
ip link set dev br0 type bridge vlan_filtering 0
echo 100 > /sys/class/net/br0/bridge/default_pvid
ip link set dev br0 type bridge vlan_filtering 1
ip link set dev sw0p1 master br0
ip link set dev sw0p2 master br0
```

Enabling "switch"

The Switch mode can be enabled by configuring devlink driver parameter "switch_mode" to 1/true:

```
devlink dev param set platform/48484000.switch \
name switch_mode value 1 cmode runtime
```

This can be done regardless of the state of Port's netdev devices - UP/DOWN, but Port's netdev devices have to be in UP before

joining to the bridge to avoid overwriting of bridge configuration as CPSW switch driver copletly reloads its configuration when first Port changes its state to UP.

When the both interfaces joined the bridge - CPSW switch driver will enable marking packets with offload_fwd_mark flag unless "ale_bypass=0"

All configuration is implemented via switchdev API.

Bridge setup

```
devlink dev param set platform/48484000.switch \
name switch_mode value 1 cmode runtime

ip link add name br0 type bridge
ip link set dev br0 type bridge ageing_time 1000
ip link set dev sw0p1 up
ip link set dev sw0p2 up
ip link set dev sw0p1 master br0
ip link set dev sw0p2 master br0

[*] bridge vlan add dev br0 vid 1 pvid untagged self

[*] if vlan_filtering=1. where default_pvid=1

Note. Steps [*] are mandatory.
```

On/off STP

```
ip link set dev BRDEV type bridge stp_state 1/0
```

VLAN configuration

```
bridge vlan add dev br0 vid 1 pvid untagged self <---- add cpu port to VLAN 1
```

Note. This step is mandatory for bridge/default_pvid.

Add extra VLANs

1. untagged:

   ```
   bridge vlan add dev sw0p1 vid 100 pvid untagged master
   bridge vlan add dev sw0p2 vid 100 pvid untagged master
   bridge vlan add dev br0 vid 100 pvid untagged self <---- Add cpu port to VLAN100
   ```

2. tagged:

   ```
   bridge vlan add dev sw0p1 vid 100 master
   bridge vlan add dev sw0p2 vid 100 master
   bridge vlan add dev br0 vid 100 pvid tagged self <---- Add cpu port to VLAN100
   ```

FDBs

FDBs are automatically added on the appropriate switch port upon detection

Manually adding FDBs:

```
bridge fdb add aa:bb:cc:dd:ee:ff dev sw0p1 master vlan 100
bridge fdb add aa:bb:cc:dd:ee:fe dev sw0p2 master <---- Add on all VLANs
```

MDBs

MDBs are automatically added on the appropriate switch port upon detection

Manually adding MDBs:

```
bridge mdb add dev br0 port sw0p1 grp 239.1.1.1 permanent vid 100
bridge mdb add dev br0 port sw0p1 grp 239.1.1.1 permanent <---- Add on all VLANs
```

Multicast flooding

CPU port mcast_flooding is always on

Turning flooding on/off on swithch ports: bridge link set dev sw0p1 mcast_flood on/off

Access and Trunk port

```
bridge vlan add dev sw0p1 vid 100 pvid untagged master
bridge vlan add dev sw0p2 vid 100 master


bridge vlan add dev br0 vid 100 self
ip link add link br0 name br0.100 type vlan id 100
```

Note. Setting PVID on Bridge device itself working only for default VLAN (default_pvid).

NFS

The only way for NFS to work is by chrooting to a minimal environment when switch configuration that will affect connectivity is needed. Assuming you are booting NFS with eth1 interface(the script is hacky and it's just there to prove NFS is doable).

setup.sh:

```
#!/bin/sh
mkdir proc
mount -t proc none /proc
ifconfig br0  > /dev/null
if [ $? -ne 0 ]; then
        echo "Setting up bridge"
        ip link add name br0 type bridge
        ip link set dev br0 type bridge ageing_time 1000
        ip link set dev br0 type bridge vlan_filtering 1

        ip link set eth1 down
        ip link set eth1 name sw0p1
        ip link set dev sw0p1 up
        ip link set dev sw0p2 up
        ip link set dev sw0p2 master br0
        ip link set dev sw0p1 master br0
        bridge vlan add dev br0 vid 1 pvid untagged self
        ifconfig sw0p1 0.0.0.0
        udhchc -i br0
fi
umount /proc
```

run_nfs.sh::

```
#!/bin/sh
mkdir /tmp/root/bin -p
mkdir /tmp/root/lib -p

cp -r /lib/ /tmp/root/
cp -r /bin/ /tmp/root/
cp /sbin/ip /tmp/root/bin
cp /sbin/bridge /tmp/root/bin
cp /sbin/ifconfig /tmp/root/bin
cp /sbin/udhcpc /tmp/root/bin
cp /path/to/setup.sh /tmp/root/bin
chroot /tmp/root/ busybox sh /bin/setup.sh

run ./run_nfs.sh
```