STMicroelectronics 10/100/1000 Synopsys Ethernet driver

Author: Giuseppe Cavallaro <peppe.cavallaro@st.com>

This is the driver for the MAC 10/100/1000 on-chip Ethernet controllers
(Synopsys IP blocks); it has been fully tested on STLinux platforms.

Currently this network device driver is for all STM embedded MAC/GMAC
(7xxx SoCs).

DWC Ether MAC 10/100/1000 Universal version 3.41a and DWC Ether MAC 10/100
Universal version 4.0 have been used for developing the first code
implementation.

Please, for more information also visit: www.stlinux.com

1) Kernel Configuration
The kernel configuration option is STMMAC_ETH:
 Device Drivers ---> Network device support ---> Ethernet (1000 Mbit) --->
 STMicroelectronics 10/100/1000 Ethernet driver (STMMAC_ETH)

2) Driver parameters list:
        debug: message level (0: no output, 16: all);
        phyaddr: to manually provide the physical address to the PHY device;
        dma_rxsize: DMA rx ring size;
        dma_txsize: DMA tx ring size;
        buf_sz: DMA buffer size;
        tc: control the HW FIFO threshold;
        tx_coe: Enable/Disable Tx Checksum Offload engine;
        watchdog: transmit timeout (in milliseconds);
        flow_ctrl: Flow control ability [on/off];
        pause: Flow Control Pause Time;
        tmrate: timer period (only if timer optimisation is configured).

3) Command line options
Driver parameters can be also passed in command line by using:
        stmmaceth=dma_rxsize:128,dma_txsize:512

4) Driver information and notes

4.1) Transmit process
The xmit method is invoked when the kernel needs to transmit a packet; it sets
the descriptors in the ring and informs the DMA engine that there is a packet
ready to be transmitted.
Once the controller has finished transmitting the packet, an interrupt is
triggered; So the driver will be able to release the socket buffers.
By default, the driver sets the NETIF_F_SG bit in the features field of the
net_device structure enabling the scatter/gather feature.

4.2) Receive process
When one or more packets are received, an interrupt happens. The interrupts
are not queued so the driver has to scan all the descriptors in the ring during
the receive process.
This is based on NAPI so the interrupt handler signals only if there is work to
be

done, and it exits.
Then the poll method will be scheduled at some future point.
The incoming packets are stored, by the DMA, in a list of pre-allocated socket buffers in order to avoid the memcpy (Zero-copy).

4.3) Timer-Driver Interrupt
Instead of having the device that asynchronously notifies the frame receptions, the
driver configures a timer to generate an interrupt at regular intervals.
Based on the granularity of the timer, the frames that are received by the device
will experience different levels of latency. Some NICs have dedicated timer
device to perform this task. STMMAC can use either the RTC device or the TMU
channel 2 on STLinux platforms.
The timers frequency can be passed to the driver as parameter; when change it,
take care of both hardware capability and network stability/performance impact.
Several performance tests on STM platforms showed this optimisation allows to spare
the CPU while having the maximum throughput.

4.4) WOL
Wake up on Lan feature through Magic Frame is only supported for the GMAC core.

4.5) DMA descriptors
Driver handles both normal and enhanced descriptors. The latter has been only
tested on DWC Ether MAC 10/100/1000 Universal version 3.41a.

4.6) Ethtool support
Ethtool is supported. Driver statistics and internal errors can be taken using:
ethtool -S ethX command. It is possible to dump registers etc.

4.7) Jumbo and Segmentation Offloading
Jumbo frames are supported and tested for the GMAC.
The GSO has been also added but it's performed in software.
LRO is not supported.

4.8) Physical
The driver is compatible with PAL to work with PHY and GPHY devices.

4.9) Platform information
Several information came from the platform; please refer to the
driver's Header file in include/linux directory.

```
struct plat_stmmacenet_data {
        int bus_id;
        int pbl;
        int has_gmac;
        void (*fix_mac_speed)(void *priv, unsigned int speed);
        void (*bus_setup)(unsigned long ioaddr);
#ifdef CONFIG_STM_DRIVERS
        struct stm_pad_config *pad_config;
#endif
        void *bsp_priv;
};
```

Where:
- pbl (Programmable Burst Length) is maximum number of
  beats to be transferred in one DMA transaction.
  GMAC also enables the 4xPBL by default.
- fix_mac_speed and bus_setup are used to configure internal target
  registers (on STM platforms);
- has_gmac: GMAC core is on board (get it at run-time in the next step);
- bus_id: bus identifier.

```
struct plat_stmmacphy_data {
        int bus_id;
        int phy_addr;
        unsigned int phy_mask;
        int interface;
        int (*phy_reset)(void *priv);
        void *priv;
};
```

Where:
- bus_id: bus identifier;
- phy_addr: physical address used for the attached phy device;
            set it to -1 to get it at run-time;
- interface: physical MII interface mode;
- phy_reset: hook to reset HW function.

TODO:
- Continue to make the driver more generic and suitable for other Synopsys
  Ethernet controllers used on other architectures (i.e. ARM).
- 10G controllers are not supported.
- MAC uses Normal descriptors and GMAC uses enhanced ones.
  This is a limit that should be reviewed. MAC could want to
  use the enhanced structure.
- Checksumming: Rx/Tx csum is done in HW in case of GMAC only.
- Review the timer optimisation code to use an embedded device that seems to be
  available in new chip generations.