Rckemac in Detail

Part 2

Jae Min Choi

SNUCSE

emac_open

- After setup_emac
- Set network address
  - Get MAC address of core by calling *get_mac_address*
  - Consecutively save least 8 bits into *dev_addr*
- If interrupt is enabled
  - Call *emac_clear_interrupt*
    - Set APIC mask (*unset_lapic_mask*)
    - Set interrupt bit by reading and writing *priv->irq_address*
    - Reset by writing *priv->device* to *RA(IRQ_RESET, priv->pid * 2)*
  - Enable interrupt
    - Read and write *RA(IRQ_MASK, priv->pid * 2)*
    - Write *EMAC_IRQ_CONFIG* to *RA(IRQ_CONFIG, priv->pid)*
    - Call *request_irq* and check its return value
- Start network queue (netif_start_queue)
- If using polling, start RX schedule (netif_rx_schedule)

emac_rx

- Has 2 goto parts: *again* & *rxDone*
  - *again*: actual reception process
  - *rxDone*: update driver's RX read offset and call *again* if packets remain
- First check if write offset is greater than RX buffer max
- In *again*:
  - Increment *read_offset* and calculate the address (*addr*)
  - Get packet length from reading 2 bytes from *addr* using *U16* macro
  - Check for over/underflow (compare packet length with size of *iphdr* and 1536)
    - If *write_offset* > *priv->rx_buffer_max*, set *priv->shutdown* to 1
    - Goto *rxDone*
  - Allocate buffer (*skb*)
    - Call *dev_alloc_skb*, drop packet if low on memory
    - Call *skb_put*
    - Compare *read_offset* with *write_offset*
      - If former is smaller, *memcpy* packet data
      - Else copy the rest of the buffer and copy the remaining data
  - Set *skb* fields

emac_tx

- Check for over/underflow (no need to shutdown unlike *emac_rx*)
- Increment driver TX buffer write offset
- Read TX buffer read offset from GRB
- Calculate address of where the packet will be written
- Save frame length in the first 2 bytes
- Check if packet needs to be wrapped around
  - If not, just copy the packet data using *memcpy* and increment TX write offset
  - Else first copy to the end of buffer, and copy the rest starting in the front
- Update TX write offset and free skb

emac_change_mtu

- Check if *new_mtu* is smaller than *sizeof(struct iphdr)* or larger than *BUFFER_SIZE -1*
  - If so, return error
- Set *dev->mtu* to *new_mtu*

emac_timeout

- Call *netif_wake_queue*

emac_stop

- Set *priv->shutdown* to 1
- Call *free_irq* if using interrupt
- Call netif_stop_queue
- Disable TX/RX ports
  - Write to 0 corresponding GRB

emac_module_exit

- Remove & free network devices
  - unregister_netdev
  - free_netdev
- Unmap GRB and CRB from memory