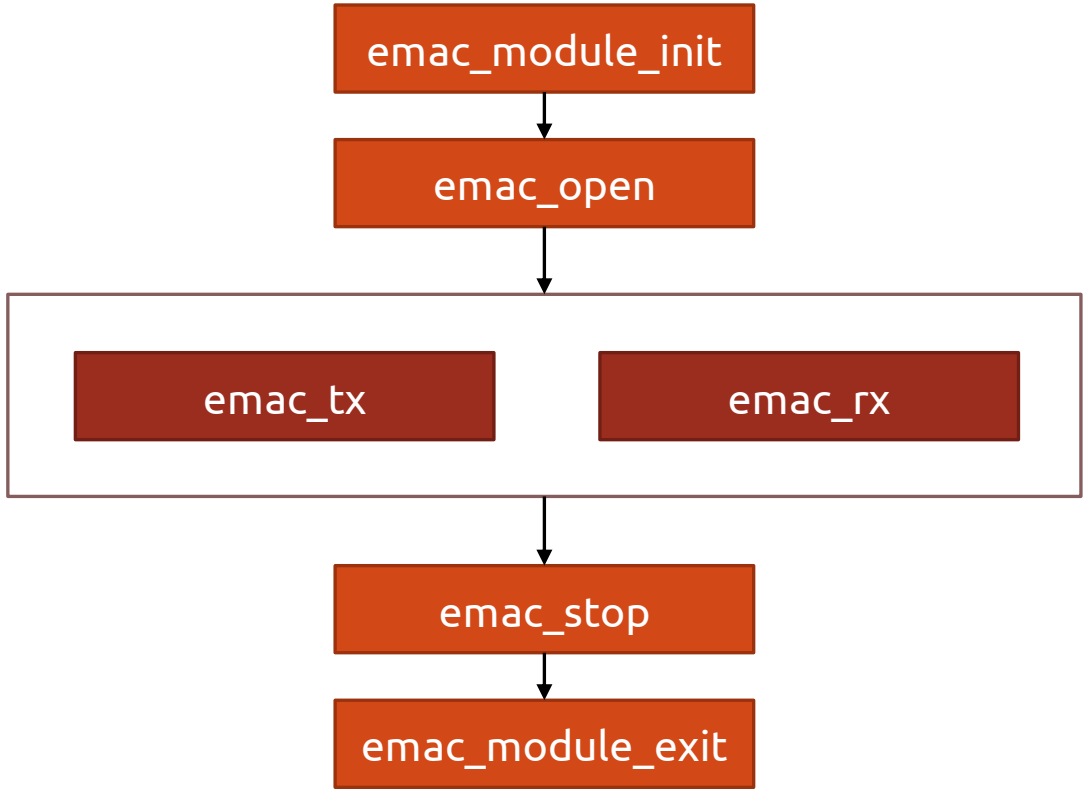
Rckemac in Detail

JAE MIN CHOI

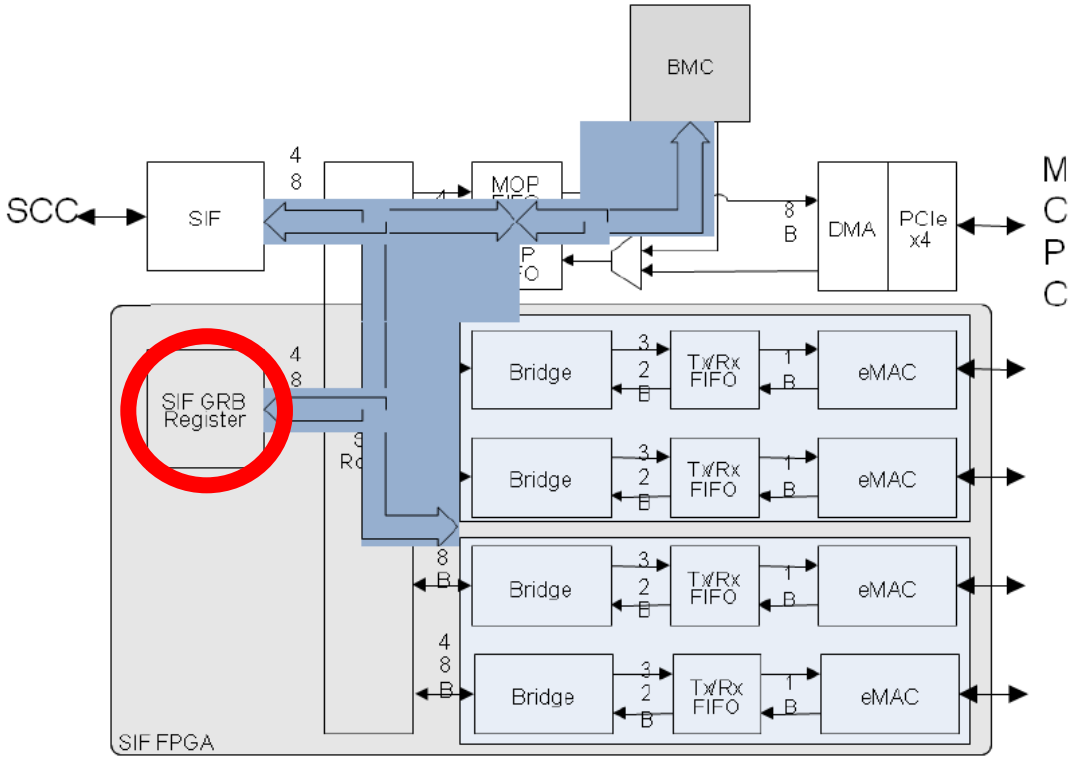
SNUCSE

Flow


emac_module_init
emac_open
emac_tx
emac_rx
emac_stop
emac_module_exit

Registers in Use

- **Global Register Bank (GRB)** : register located inside **FPGA**



Registers in Use

- **Configuration Register Bank (CRB)** : set of configuration registers for **SCC cores**
  - Control enabling local reset, core initialization & configuration, core interrupt handling, and L2 cache configuration
  - Each SCC core, L2 cache controller, and global clocking unit (GCU) has a dedicated config register
  - Writable by any core or the system interface unit
  - Byte-wise access not supported
  - Should perform read-modify-write on all 32 bits of a config register

Helper functions & macros

- *static int* ***emac_readl*** *(void *addr)*
  - Read long from eMAC at *addr*
  - Implemented with 2 *readl* calls
- *static void* ***emac_writel*** *(int value, void *addr)*
  - Write long *value* to eMAC at *addr*
  - Implemented with *writel*
- ***RA****(_x, _y)*
  - GRB address + (_*x*)+ (_*y* * 4)
- ***RA_CRB****(_x)*
  - CRB address + (_*x*)
- All accesses to the registers are done via these macros
- Both polling and interrupts can be implemented
  - Will suppose using interrupt

emac_module_init

1. Map GRB to 0xF9000000 with size 0x10000
2. Map CRB to 0xF8000000 with size 2 * (page size)
3. Read tile ID from (CRB + 0x0100) and determine core position
   - Bits 06:03 – x
   - Bits 10:07 – y
   - Bits 02:00 – z
4. Set core number
5. Get FPGA/sccKit port settings from (GRB + 0x822C)
6. Create eMAC interface (*emac0*, …)
   - Initilize Xilinx IP port (*init_xilinx_port*)
   - Allocate netdev structure (*alloc_netdev* and *emac_init*)
   - Allocate priv structure (*netdev_priv* and *memset*)

init_xilinx_port

- Called from *emac_module_init* to create an eMAC interface

1. Get transmitter & receiver address from GRB
2. Disable TX/RX flow control of eMAC (GRB + 0x32C0)
   - Set top 3 bits of flow control configuration to zero
3. Enable transmitter & set to full duplex mode (GRB + 0x3280)
   - Half duplex (Bit 26) = 0
   - Transmit enable (Bit 28) = 1
   - Reset (Bit 31) = 0
4. Enable receiver & set to full duplex mode (GRB + 0x3240)
   - Length/type error check disable (Bit 25) = 1
   - Half duplex (Bit 26) = 0
   - Receiver enable (Bit 28) = 1
   - Reset (Bit 31) = 0

init_xilinx_port

5. Set speed of eMAC to 1Gb/s (GRB + 0x3300)
   - Bit 31 = 1, Bit 30 = 0
6. Set to promiscuous mode (GRB + 0x3390)
   - Bit 31 = 1

- Be sure to conduct **sanity checks** after changing register values

emac_init

- Called by *alloc_netdev*
- Setup *net_device* structure
  - *ether_setup* to setup standard infos
  - Assign driver specific functions (open, stop, ioctl, stats, …)

```
void emac_init(struct net_device *dev) {
        /* set standard infos */
        ether_setup(dev);

        /* Network driver specific functions */
        dev->open = emac_open;
        dev->stop = emac_stop;
        dev->set_config = emac_set_config;
        dev->hard_start_xmit = emac_tx;
        dev->do_ioctl = emac_ioctl;
        dev->get_stats = emac_stats;
        dev->change_mtu = emac_change_mtu;
        dev->tx_timeout = emac_tx_timeout;

        dev->watchdog_timeo = 5;

        dev->poll = emac_poll;
        dev->weight = 64;

        dev->irq = EMAC_IRQ_NR;

        dev->hard_header_len = 2 + ETH_HLEN;
}
```

emac_open

- Called when eMAC interface is opened

1. Read tile ID and determine core position
2. Set interrupt request (IRQ) address
   - Core 0: CRB + 0x10
   - Core 1 : CRB + 0x18
3. Get *offset*, *subdest*, *route*, *mode*
   - First read value from (CRB + 0x800, Core 0) or (CRB + 0x1000, Core 1)
   - Calculate *offset*, etc. by shifting this value
4. Setup Ethernet port (*setup_emac*)
5. Set network address
   - First get MAC address (*get_mac_address*)
   - And iterate *for* loop to consecutively save lower 8 bits of address in *dev->dev_addr*
6. Enable interrupt
   - Clear interrupt (*emac_clear_interrupt*)
   - Read value from (GRB + 0xD200 + priv->pid * 2 * 4) and change it
   - Change value in (GRB + 0xD800+ priv->pid * 4)
   - Call *request_irq* and check its return value
7. Start network queue (*netif_start_queue*)

setup_emac

- Called from *emac_open*
- Reception configuration
  1. Set up ring buffer space
     - Set *priv->rx_buffer_max*
     - Allocate *priv->rx_buffer* using alloc_buffer
  2. Set start address of RX buffer (GRB + 0x9000)
  3. Read RX buffer write offset (GRB + 0x9200)
  4. Set RX buffer read offset to write offset (GRB + 0x9100)
  5. Set RX buffer size (GRB + 0x9300)
  6. Set RX buffer threshold (GRB + 0x9400)
  7. Set RX mode (GRB + 0x9500)
  8. Save MAC address (HI: GRB + 0x9600, LO: GRB + 0x9700)
  9. Activate network port (GRB + 0x9800)
- Transmission configuration
  1. Set up ring buffer space
  2. Set start address of TX buffer (GRB + 0x9900)
  3. Read TX buffer read offset (GRB + 0x9A00)
  4. Set TX buffer write offset to read offset (GRB + 0x9B00)
  5. Set TX buffer size (GRB + 0x9C00)
  6. Set TX mode (GRB + 0x9D00)
  7. Activate network port (GRB + 0x9E00)

TODO

- emac_rx
- emac_tx
- emac_stop
- emac_module_exit