TP-LINK®

Hawkeye End驱动研究与分析

SMB交换 邱俊源

目的

- 熟悉**End**驱动开发流程
- 分析**Hawkeye Bsp**中**End**驱动的层次结构，为后续**Broadcom**平台机型的**End**驱动开发做准备

大纲

Hawkeye END层次结构

MUX接口层

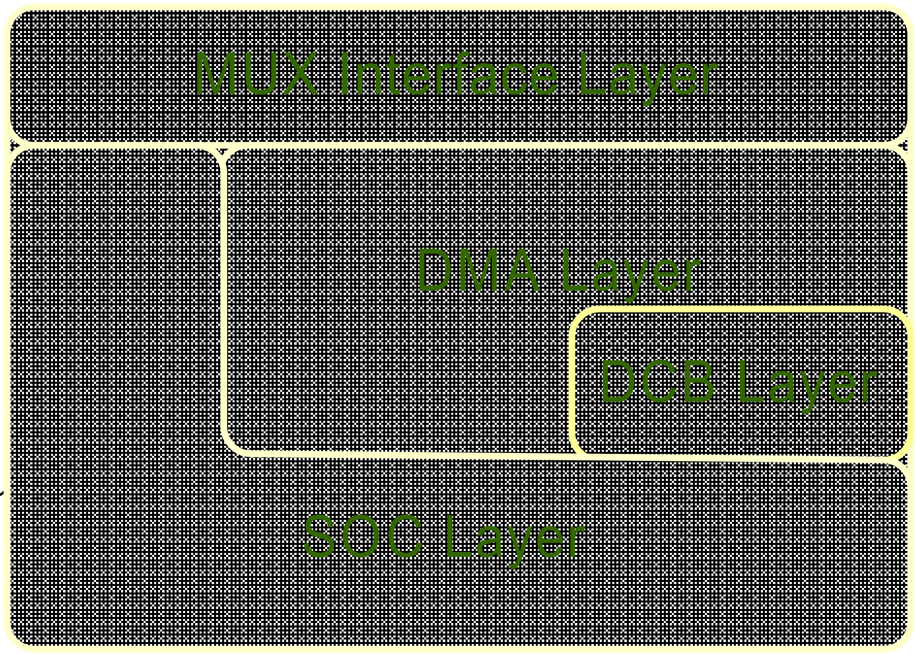
硬件实现层

END初始化流程

数据包收发流程

Hawkeye End层次结构

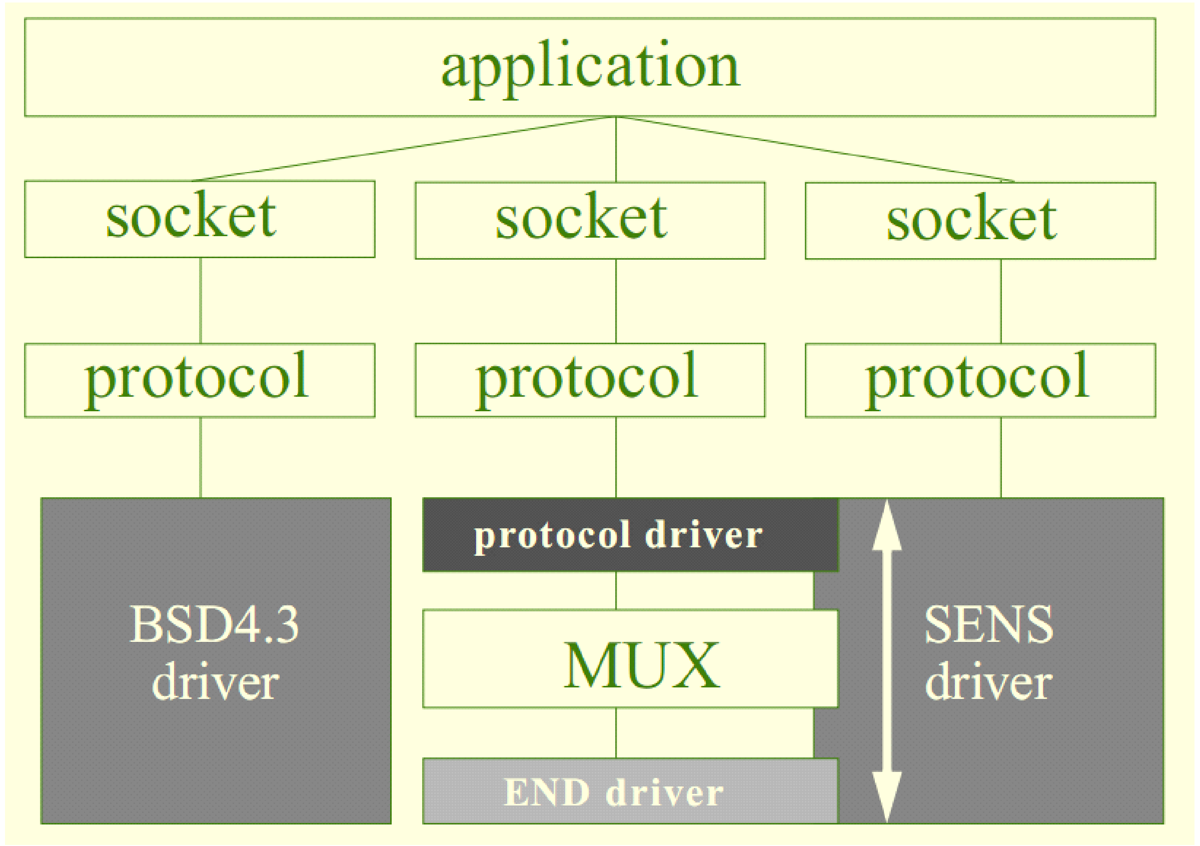
- **END**分层
  - **MUX**接口层
  - 硬件实现层
    - **DMA**处理层
    - **DCB**操作层
    - **SOC**操作层（初始化 **PCI**、**REG**、**MEM** **MII r/w**）



Hawkeye End层次结构

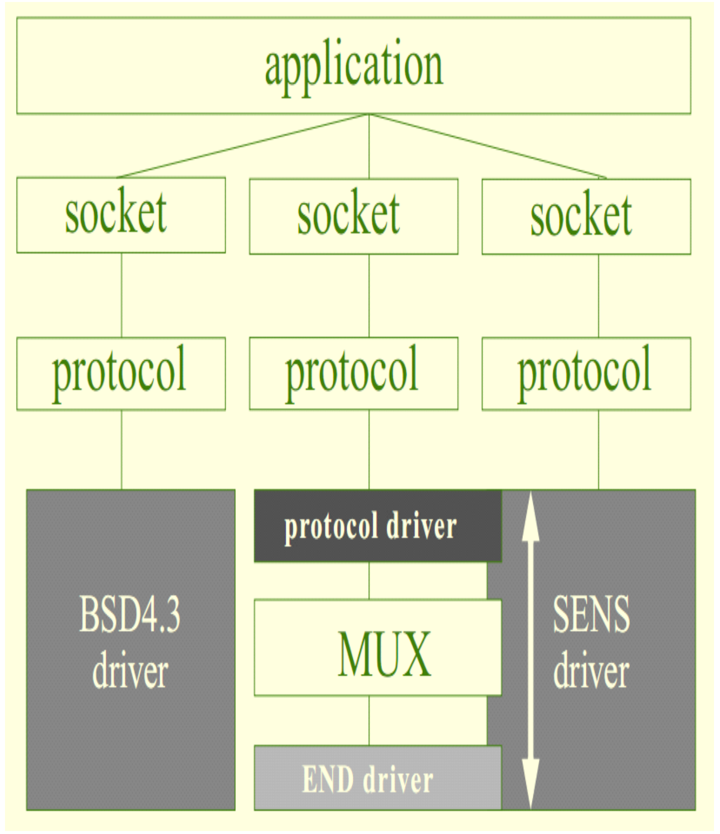
END基础

- **Enhanced Network Drivers**



END基础

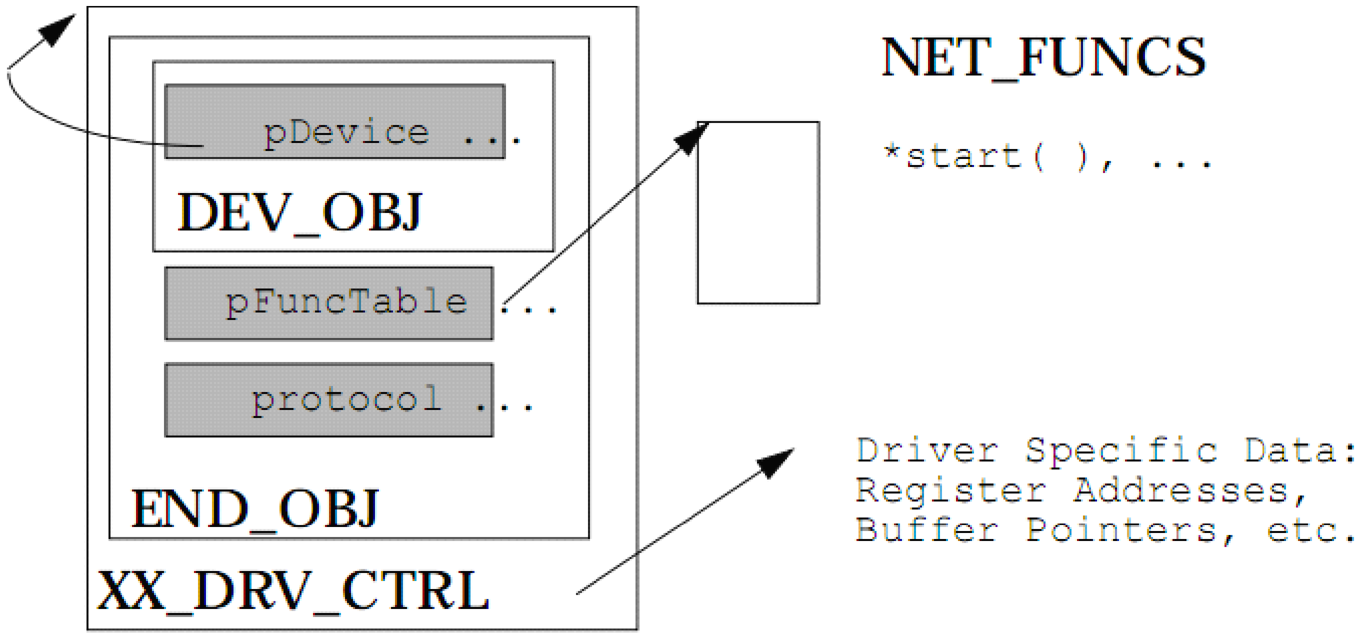
- **Scalable Enhanced Network Stack**
- **Multiplex Layer**
- **END**
  - **Status and Control Registers**
  - **Interrupts Events**
  - **DMA Hardware**



MUX接口层_数据结构

```
typedef struct end_object
{
        NODE                node;
        DEV_OBJ             devObject;          /* driver object */
        STATUS              (*receiveRtn) (void*, M_BLK_ID);
        BOOL                (*outputFilter) (void*, long,M_BLK_ID, M_BLK_ID, void*);
        BOOL                attached;           /* attachment flag. */
        SEM_ID              txSem;              /* transmit semaphore. */
        long                flags;              /* various flags. */
        NET_FUNCS *         pFuncTable;         /* END functionstable */
        M2_INTERFACETBL     mib2Tbl;            /* counters */
        LIST                multiList;          /* address list head*/
        int                 nMulti;             /* list count */
        LIST                protocols;          /* NET_PROTOCOL list. */
        BOOL                snarfProto;         /* snarf flag */
        NET_POOL_ID         pNetPool;           /* memory cookie */
} END_OBJ;
```

MUX接口层_数据结构


pDevice ...
DEV_OBJ
pFuncTable ...
protocol ...
END_OBJ
XX_DRV_CTRL
NET_FUNCS
*start( ), ...
Driver Specific Data:
Register Addresses,
Buffer Pointers, etc.

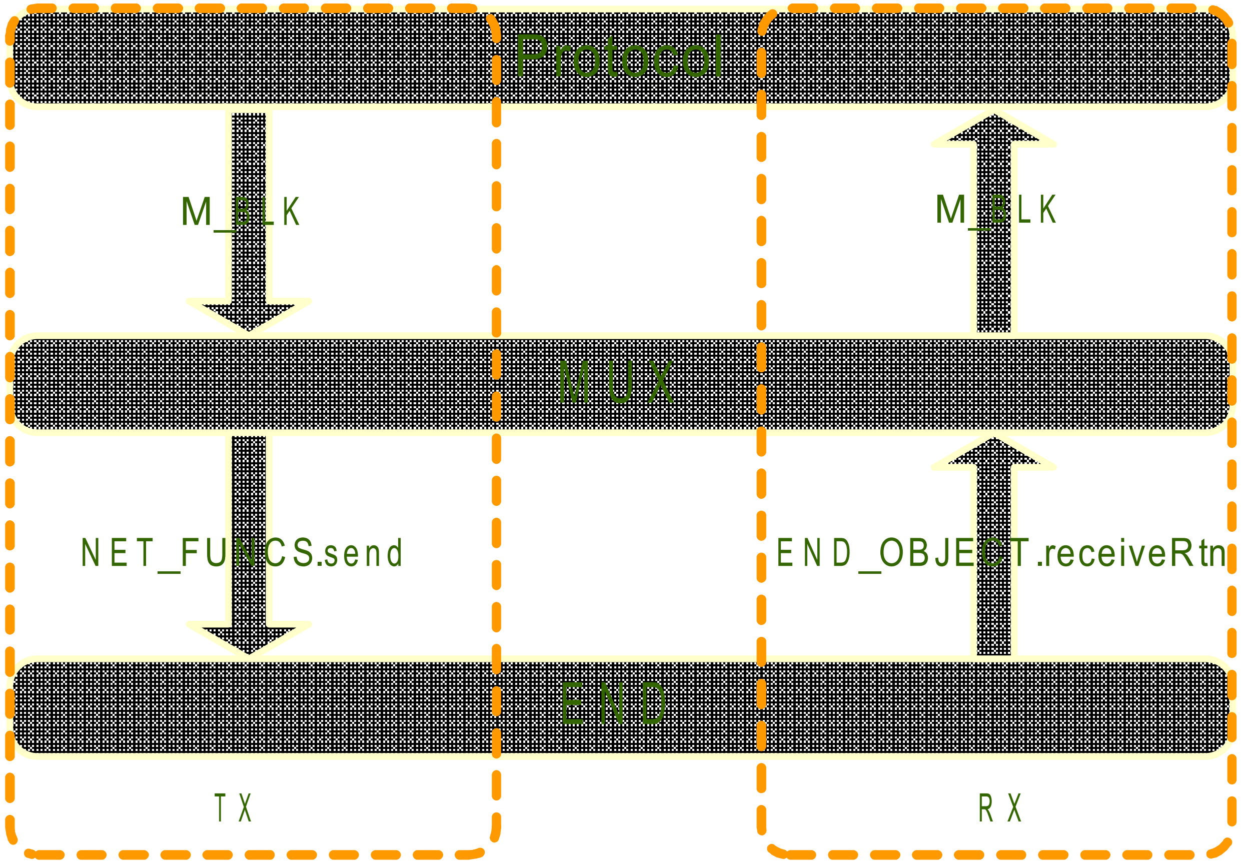
MUX接口层_数据结构

```
typedef struct net_funcs
{
        STATUS    (*start) (void*);
        STATUS    (*stop) (void*);
        STATUS    (*unload) (void*);
        int       (*ioctl) (void*, int, caddr_t);
        STATUS    (*send) (void*, M_BLK_ID);
        STATUS    (*mCastAddrAdd) (void*, char*);
        STATUS    (*mCastAddrDel) (void*, char*);
        STATUS    (*mCastAddrGet) (void*, MULTI_TABLE*);
        STATUS    (*pollSend) (void*, M_BLK_ID);
        STATUS    (*pollRcv) (void*, M_BLK_ID;
        M_BLK_ID (*formAddress) (M_BLK_ID, M_BLK_ID,M_BLK_ID);
        STATUS    (*packetDataGet)(M_BLK_ID,LL_HDR_INFO *);
        STATUS    (*addrGet) (M_BLK_ID, M_BLK_ID, M_BLK_ID,M_BLK_ID,
M_BLK_ID);
} NET_FUNCS;
```

MUX接口层_ END驱动加载

- **Driver's load routine**
  - **Creates END_OBJ with NET_FUNCS**
  - **Init netpool mib etc.**
  - **Registered in endDevTbl of configNet.h**
- **MUX Initialization**
  - **muxDevLoad()**
  - **muxDevStart()**
  - **muxBind() [ipAttach]**

MUX接口层_PKT TX/RX



Hawkeye End层次结构

硬件实现层

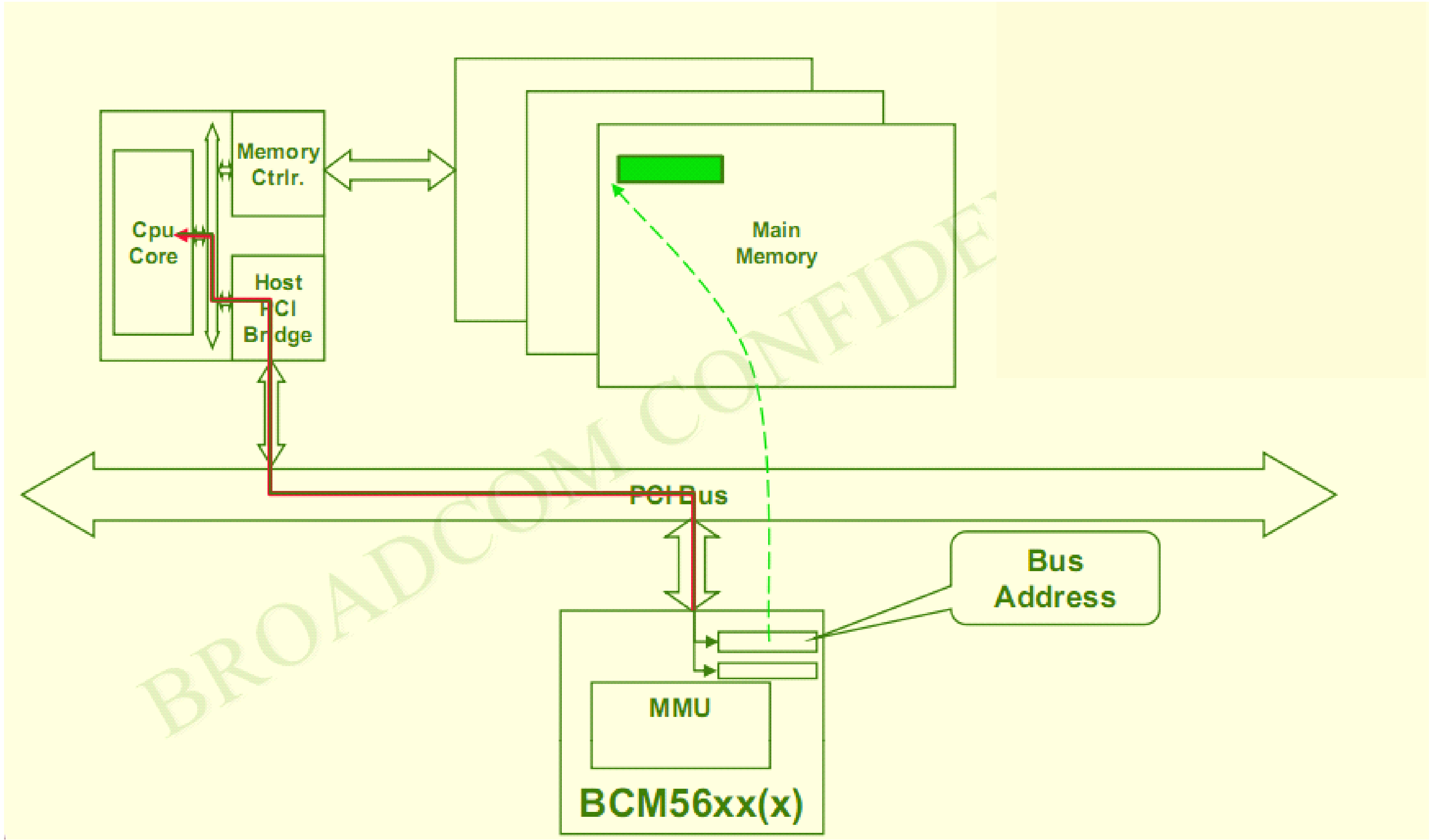
硬件实现层_基本原理

- **Packet DMA**
  - **CPU <-> SWITCH**
  - **CPU <-> MAC CORE <-> SWITCH**

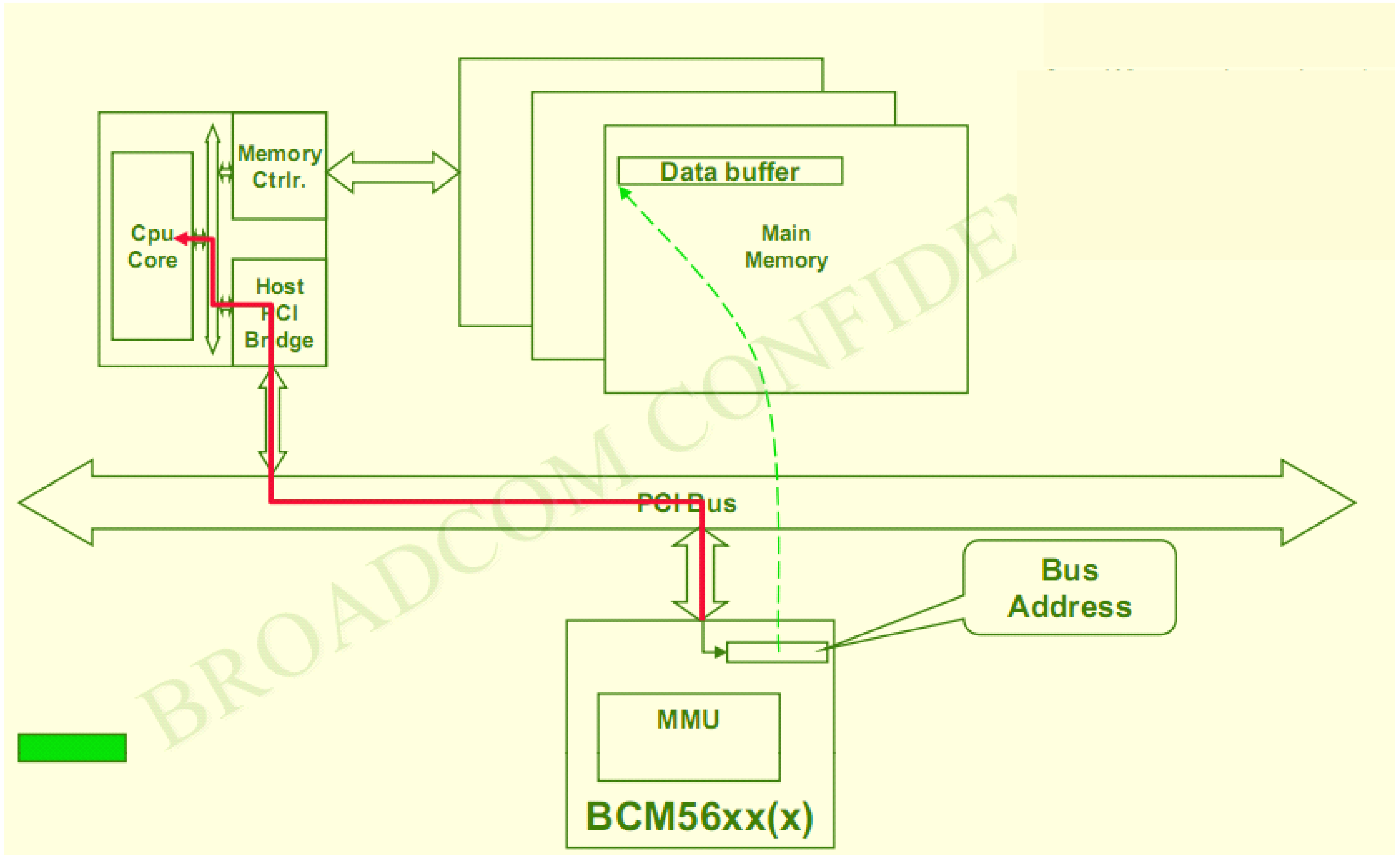
硬件实现层_基本原理

- CMIC
  - Cpu Management Interface Controller Block
- DCB
  - DMA control block (DMA Descriptor)
  - The DCB contains all of the information required for a packet data transfer.

硬件实现层_DMA TX

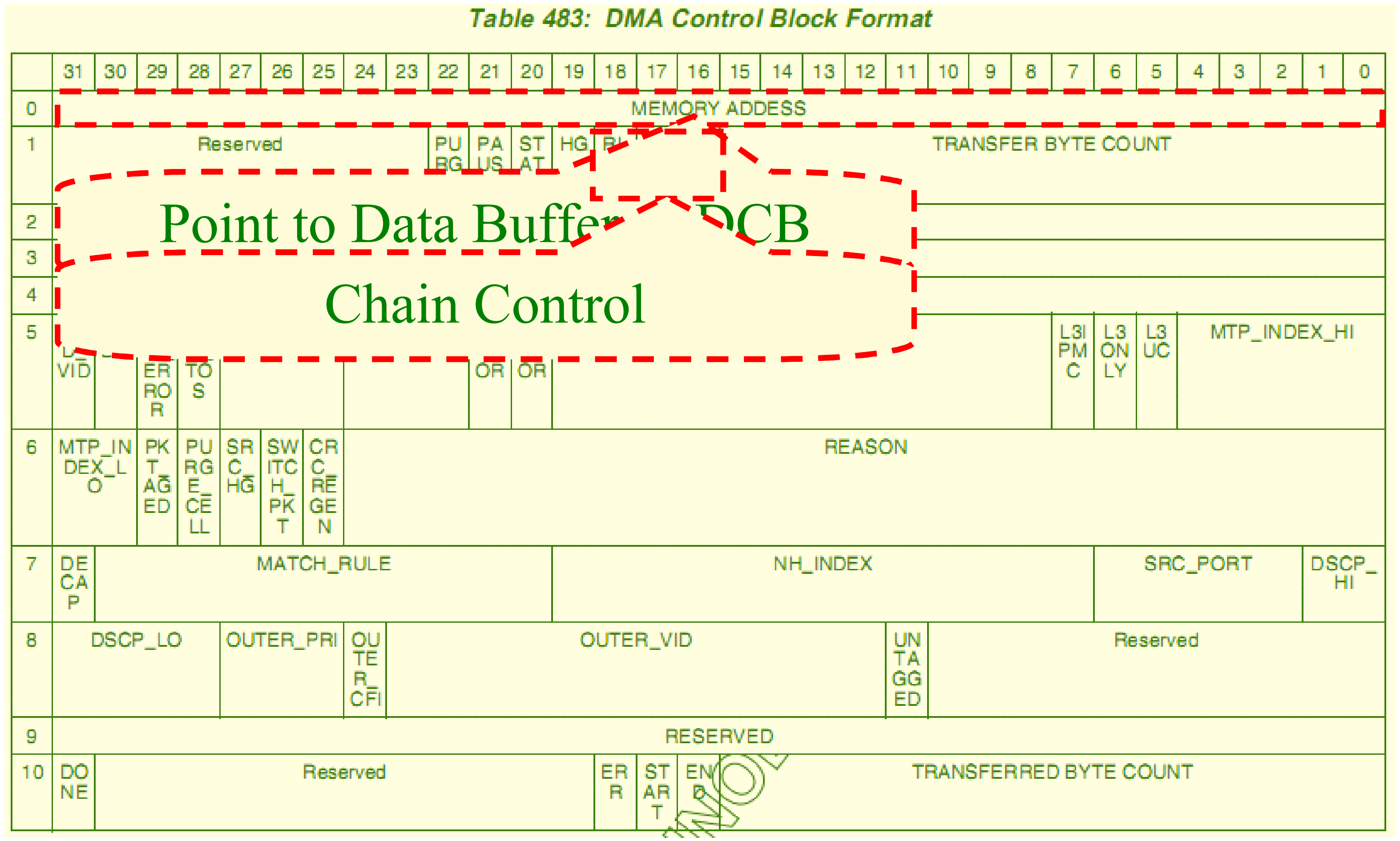

Memory Ctrlr.
Cpu Core
Host PCI Bridge
Main Memory
PCI Bus
Bus Address
MMU
BCM56xx(x)

硬件实现层_DMA RX


Memory Ctrlr.
Cpu Core
Host PCI Bridge
Data buffer
Main Memory
PCI Bus
Bus Address
MMU
BCM56xx(x)

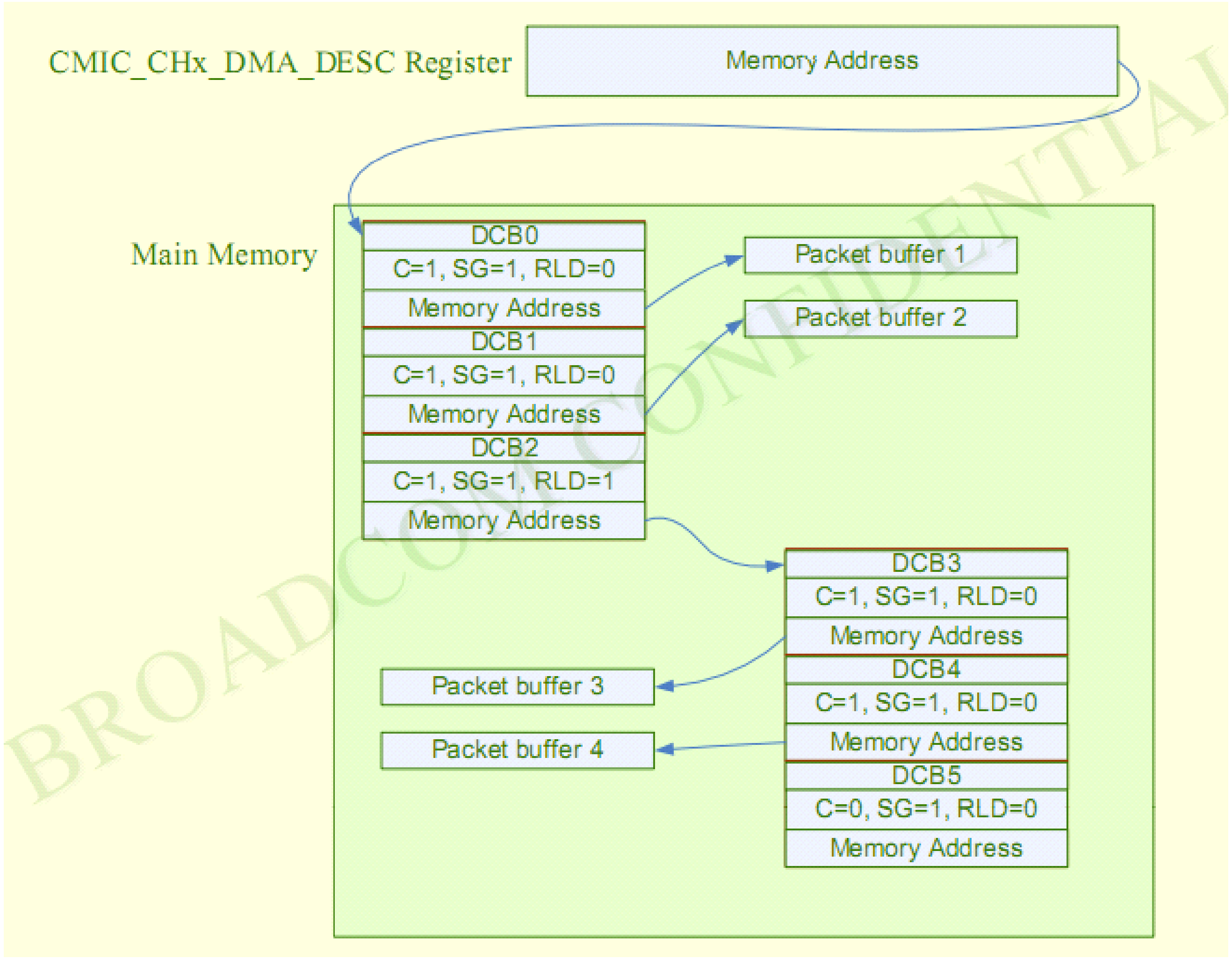
硬件实现层_DCB Format

Table 483: DMA Control Block Format

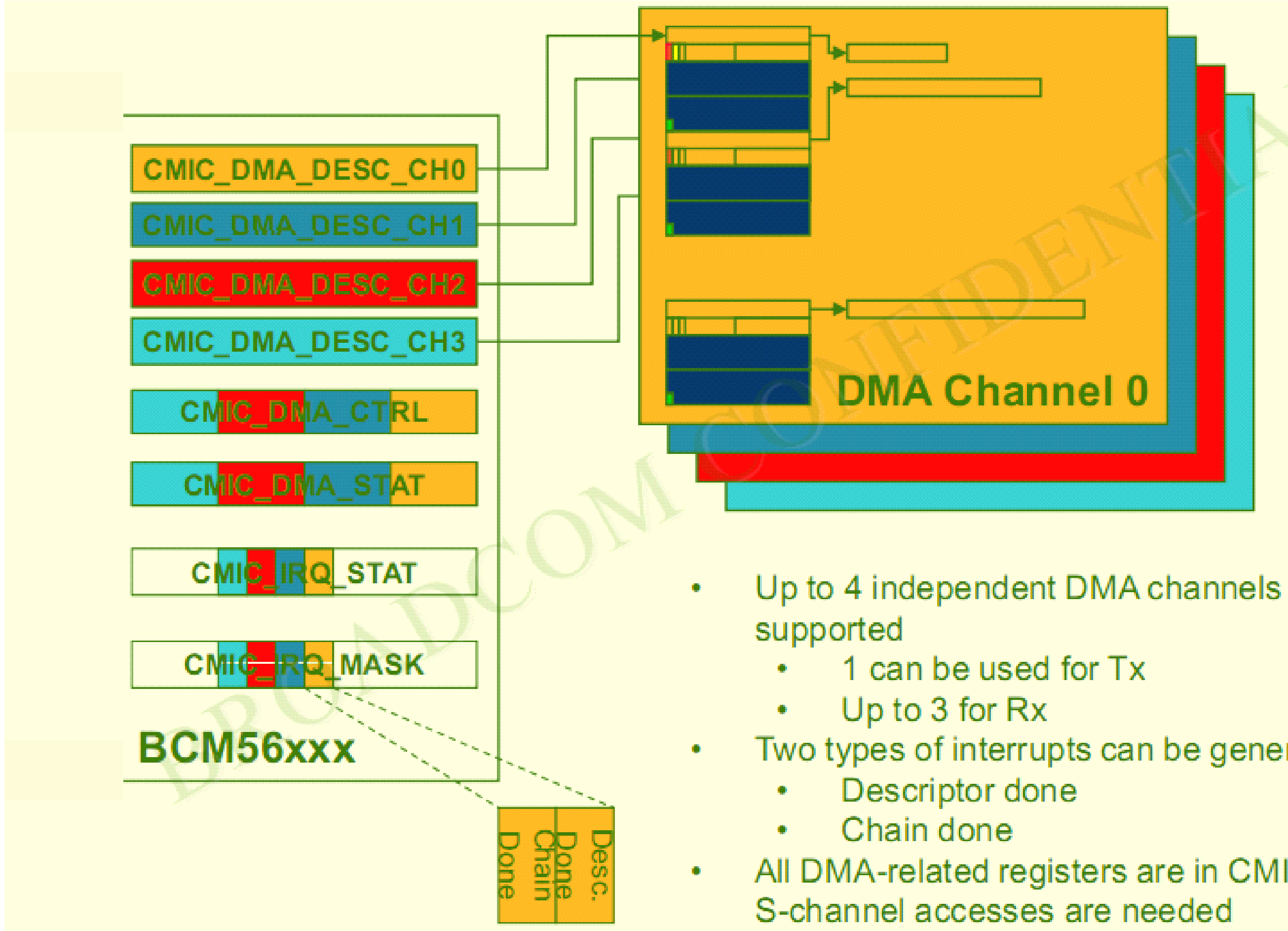

Point to Data Buffer
DCB
Chain Control

Word	Fields (bit 31 → bit 0)
0	MEMORY ADDESS
1	Reserved; PURG; PAUS; STAT; HG; RL; TRANSFER BYTE COUNT
2	
3	
4	
5	VID; ERROR; TOS; OR; OR; L3IPMC; L3ONLY; L3UC; MTP_INDEX_HI
6	MTP_INDEX_LO; PKT_AGED; PURGE_CELL; SRC_HG; SWITCH_PKT; CRC_REGEN; REASON
7	DECAP; MATCH_RULE; NH_INDEX; SRC_PORT; DSCP_HI
8	DSCP_LO; OUTER_PRI; OUTER_CFI; OUTER_VID; UNTAGGED; Reserved
9	RESERVED
10	DONE; Reserved; ERR; START; END; TRANSFERRED BYTE COUNT

硬件实现层_DCB链


CMIC_CHx_DMA_DESC Register
Memory Address
Main Memory
DCB0
C=1, SG=1, RLD=0
Memory Address
DCB1
C=1, SG=1, RLD=0
Memory Address
DCB2
C=1, SG=1, RLD=1
Memory Address
Packet buffer 1
Packet buffer 2
DCB3
C=1, SG=1, RLD=0
Memory Address
DCB4
C=1, SG=1, RLD=0
Memory Address
DCB5
C=0, SG=1, RLD=0
Memory Address
Packet buffer 3
Packet buffer 4

硬件实现层_DMA Channel



- Up to 4 independent DMA channels are supported
  - 1 can be used for Tx
  - Up to 3 for Rx
- Two types of interrupts can be generated:
  - Descriptor done
  - Chain done
- All DMA-related registers are in CMIC. No S-channel accesses are needed

DMA处理层

- 与**MUX**接口层的数据包交互
- 管理**DMA Channel** 及相应的**DCB**链
- 数据包的实际发送及接收处理

DMA处理层_主要数据结构

- soc_dma_manager_t
- dv_t
- rx_ctl_t/rx_chan_ctl_t
- bcm_pkt_t
- dcb_op_t

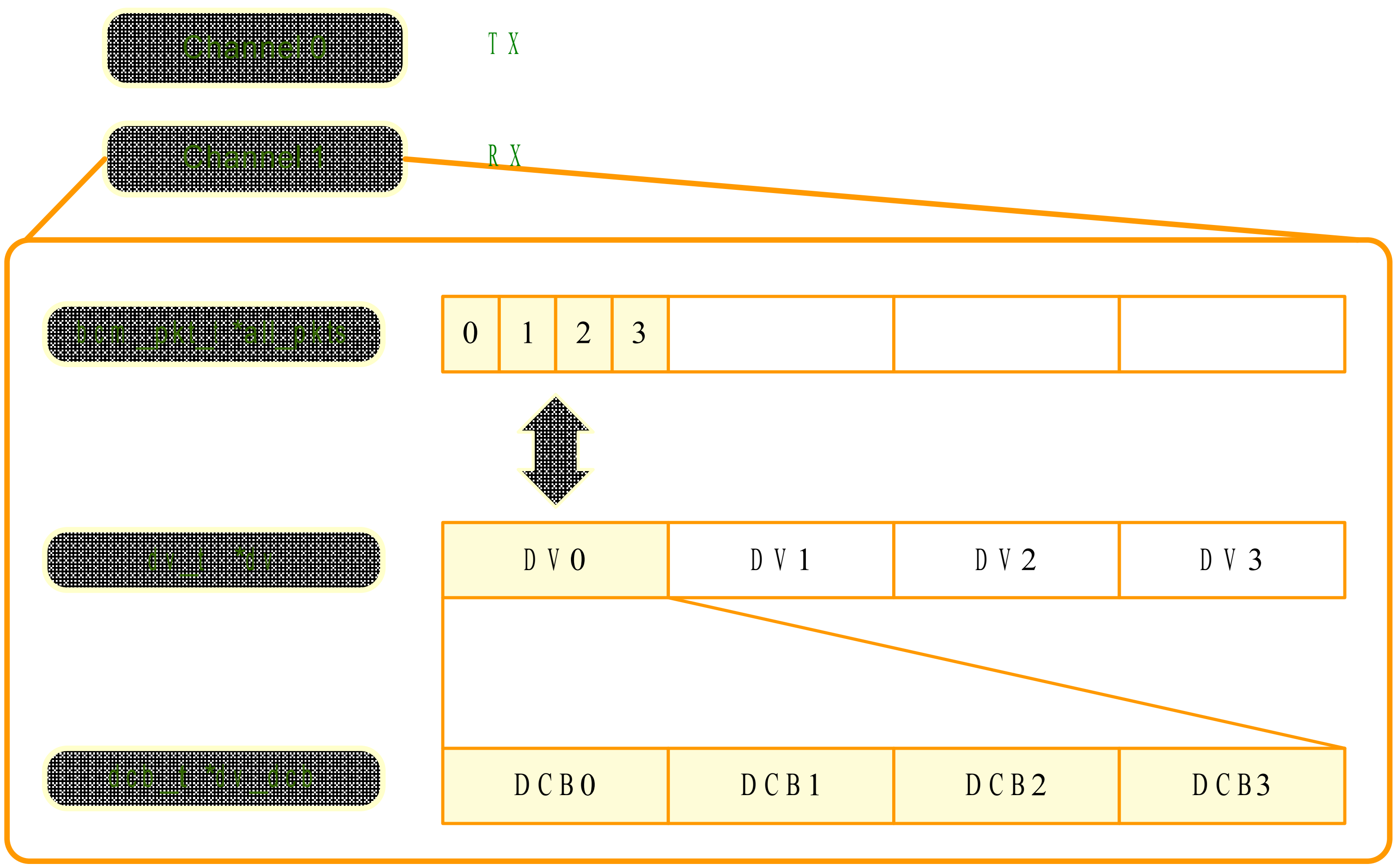
DMA处理层_主要数据结构

```
typedef struct _soc_dma_manager
{
    uint32      soc_dma_lock;                   /* Lock for updating DMA operations etc. */
    sdc_t       soc_channels[N_DMA_CHAN];
    int         soc_max_channels;               /* maximum channel count */
    sdc_t      *soc_dma_default_rx;             /* Default RX channel */
    sdc_t      *soc_dma_default_tx;             /* Default TX channel */
    int         dma_droptx;                     /* Any channels in drop tx mode */
    dv_t       *soc_dv_free;                    /* Available DVs */
    int         soc_dv_free_cnt;                /* # on free list */
    int         soc_dv_cnt;                     /* # allowed on free list */
    int         soc_dv_size;                    /* Number DCBs in free list entries */
    uint32     *tx_purge_pkt;                   /* DMA able buffer for TX Purge */
    soc_stat_t  stat;
    dcb_op_t    dcb_op;                         /* DCB operations */

    sal_mutex_t schanMutex;                     /* S-Channel mutual exclusion */
    sal_mutex_t miimMutex;
} soc_dma_manager_t;
```

DMA处理层_主要数据结构

```
typedef struct dv_s {
    struct dv_s      *dv_next,             /* Queue pointers if required */
                     *dv_chain;            /* Pointer to next DV in chain */
    int              dv_unit;              /* Unit dv is allocated on */
    uint32           dv_magic;             /* Used to indicate valid */
    dvt_t            dv_op;                /* Operation to be performed */
    dma_chan_t       dv_channel;           /* Channel queued on */
    int              dv_flags;             /* Flags for operation */

    int16      dv_cnt;                     /* # descriptors allocated */
    int16      dv_vcnt;                    /* # descriptors valid */
    int16      dv_dcnt;                    /* # descriptors done */
    void       (*dv_done_chain)(int u, struct dv_s *dv_chain);
    void       (*dv_done_desc)(int u, struct dv_s *dv, dcb_t *dcb);
    void       (*dv_done_packet)(int u, struct dv_s *dv, dcb_t *dcb);
    …
    uint8      dv_dmabuf[SOC_DV_PKTS_MAX][24];
    dcb_t       *dv_dcb;
} dv_t;
```

DMA处理层_数据结构关系

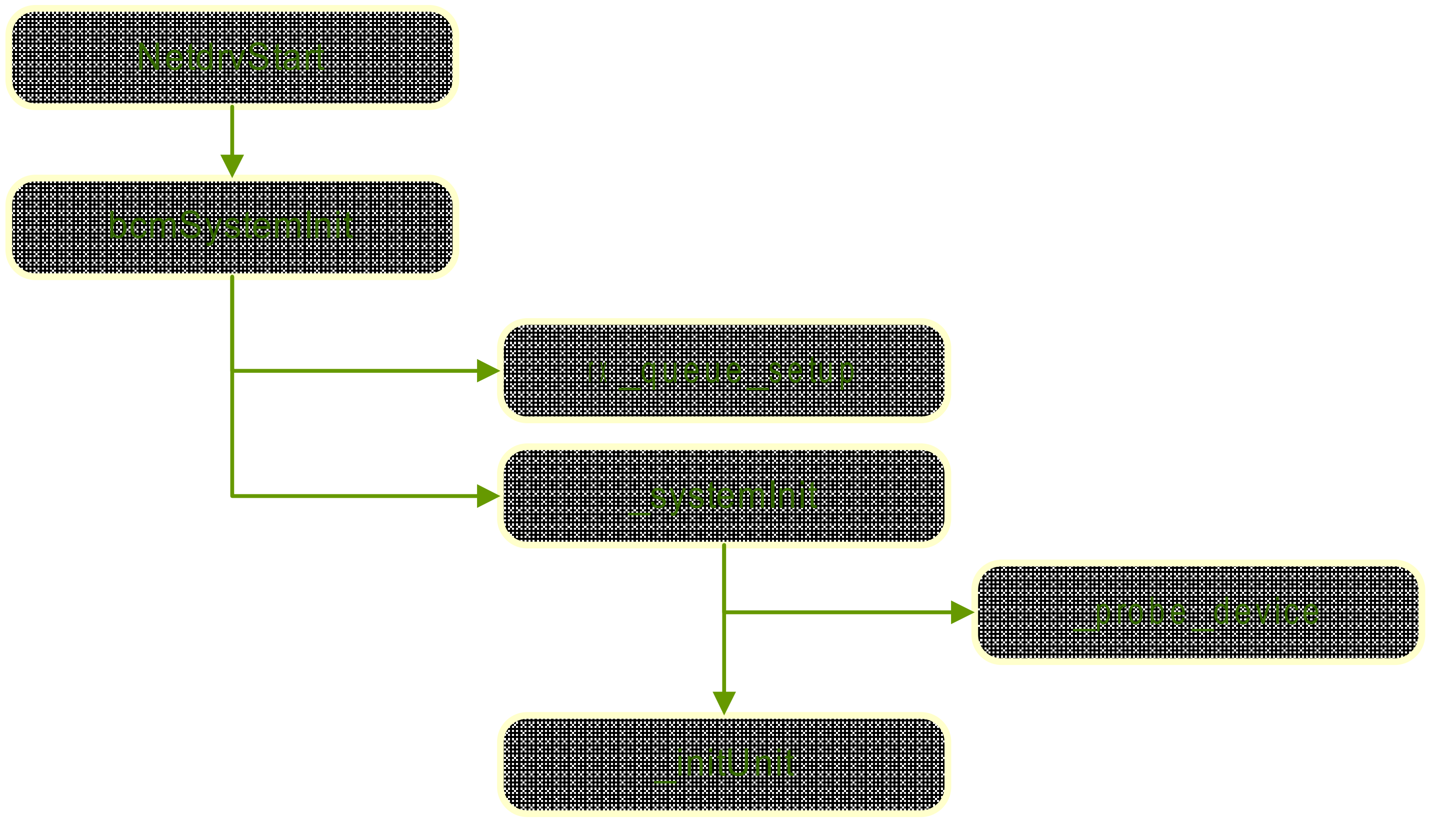


END 初始化流程

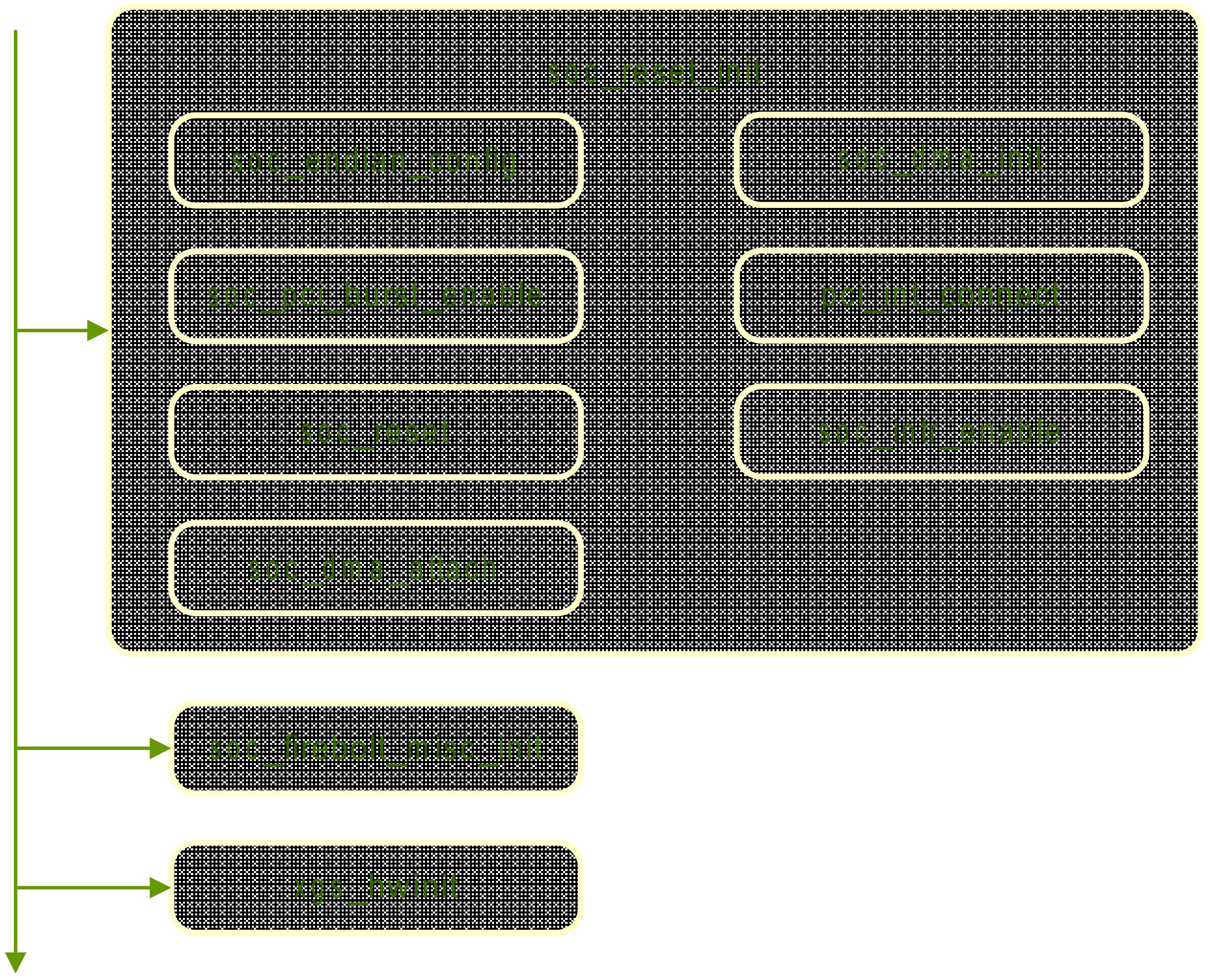
END初始化

- **SOC**初始化
- **DMA**相关寄存器初始化
- 中断使能与绑定
- **DMA**处理层初始化

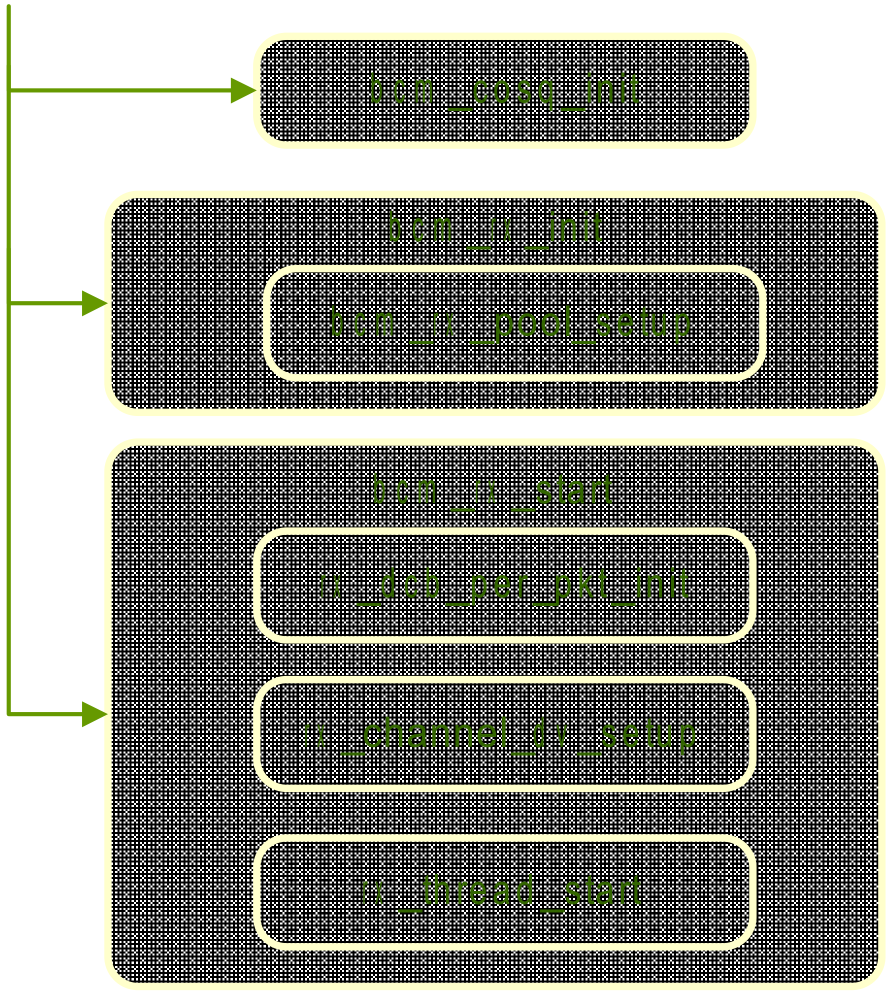
END初始化


NetdrvStart
bcmSystemInit
rx_queue_setup
_systemInit
_probe_device
_initUnit

END初始化_initUnit

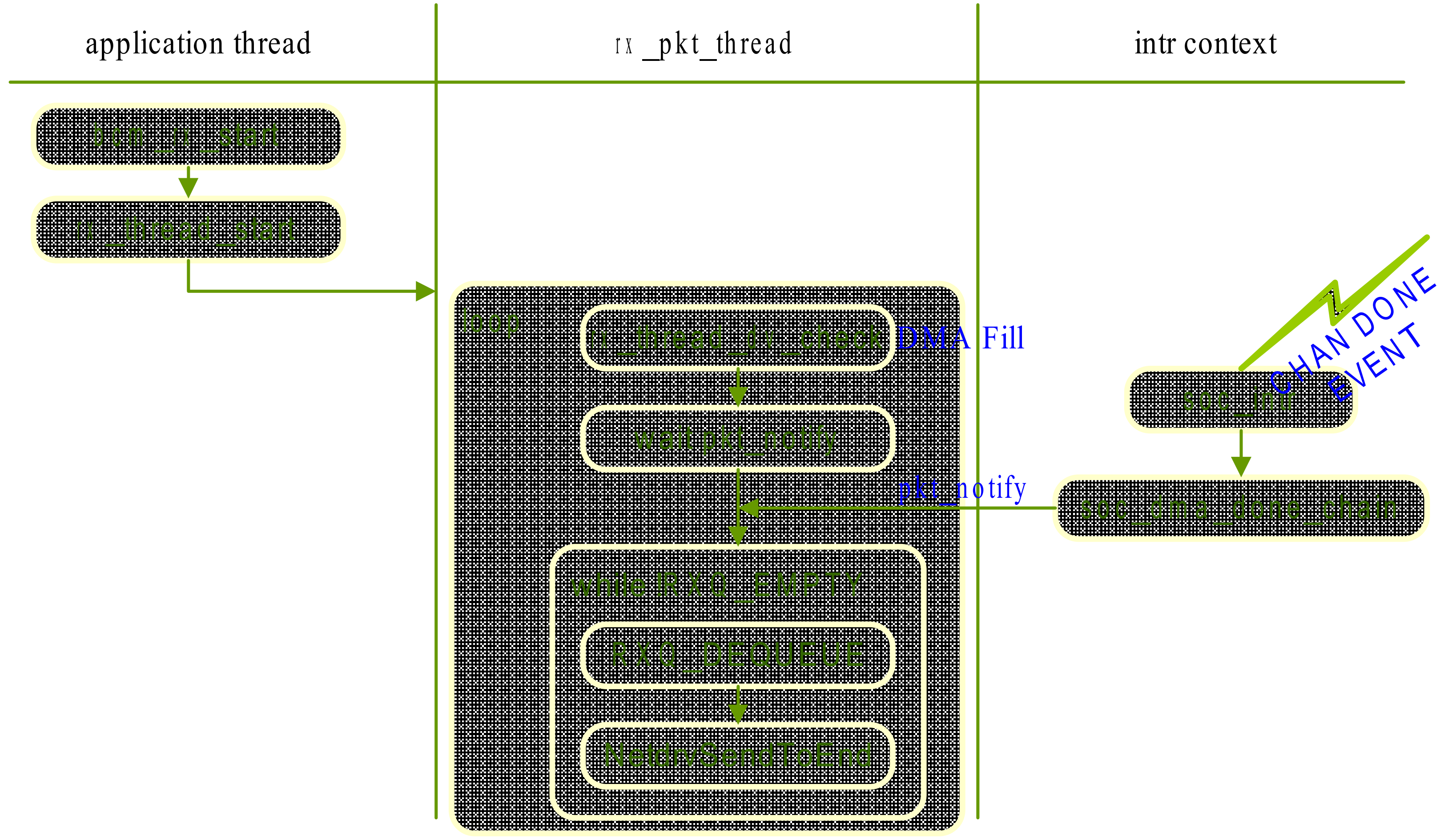


END初始化_initUnit

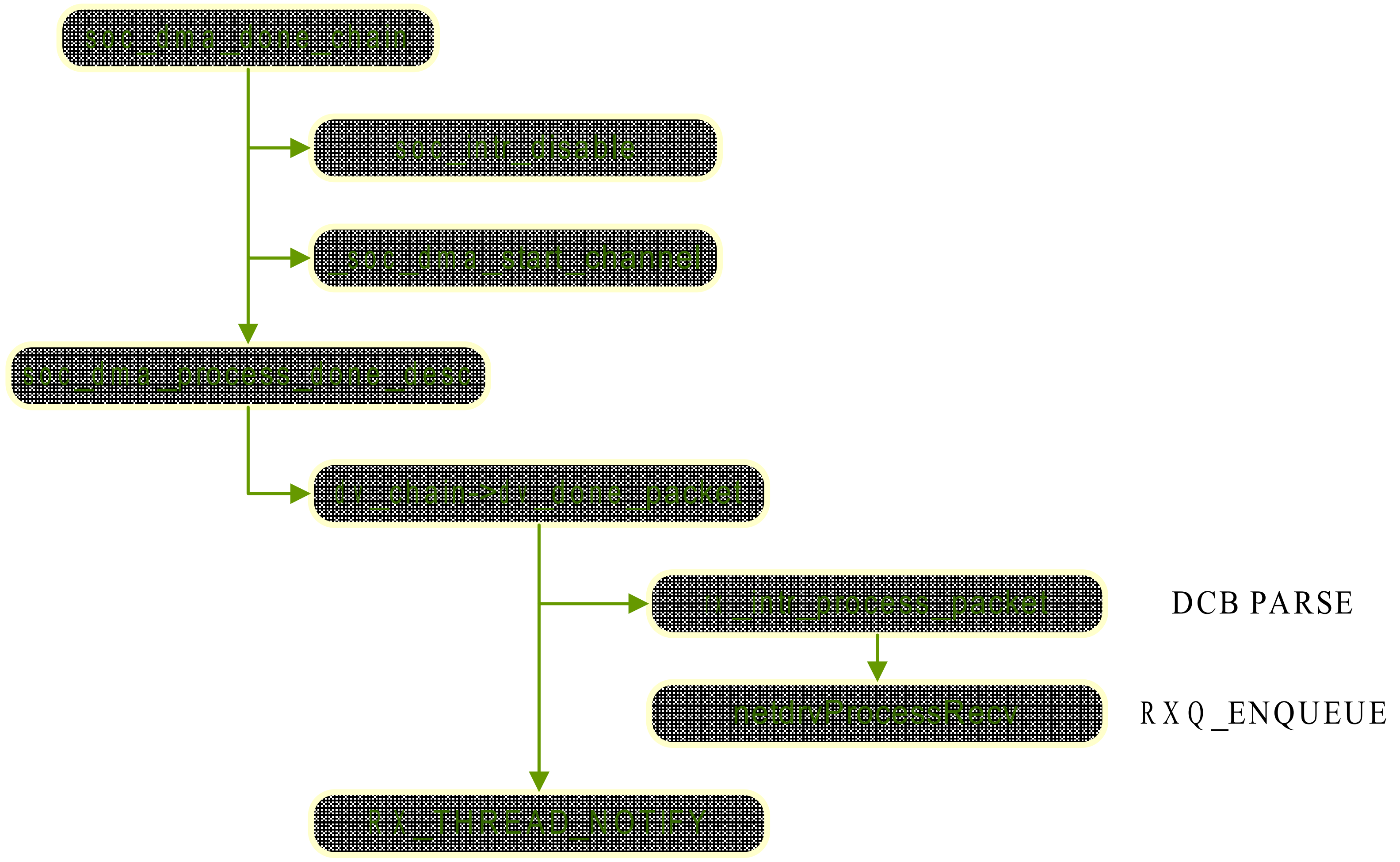


数据包收发流程

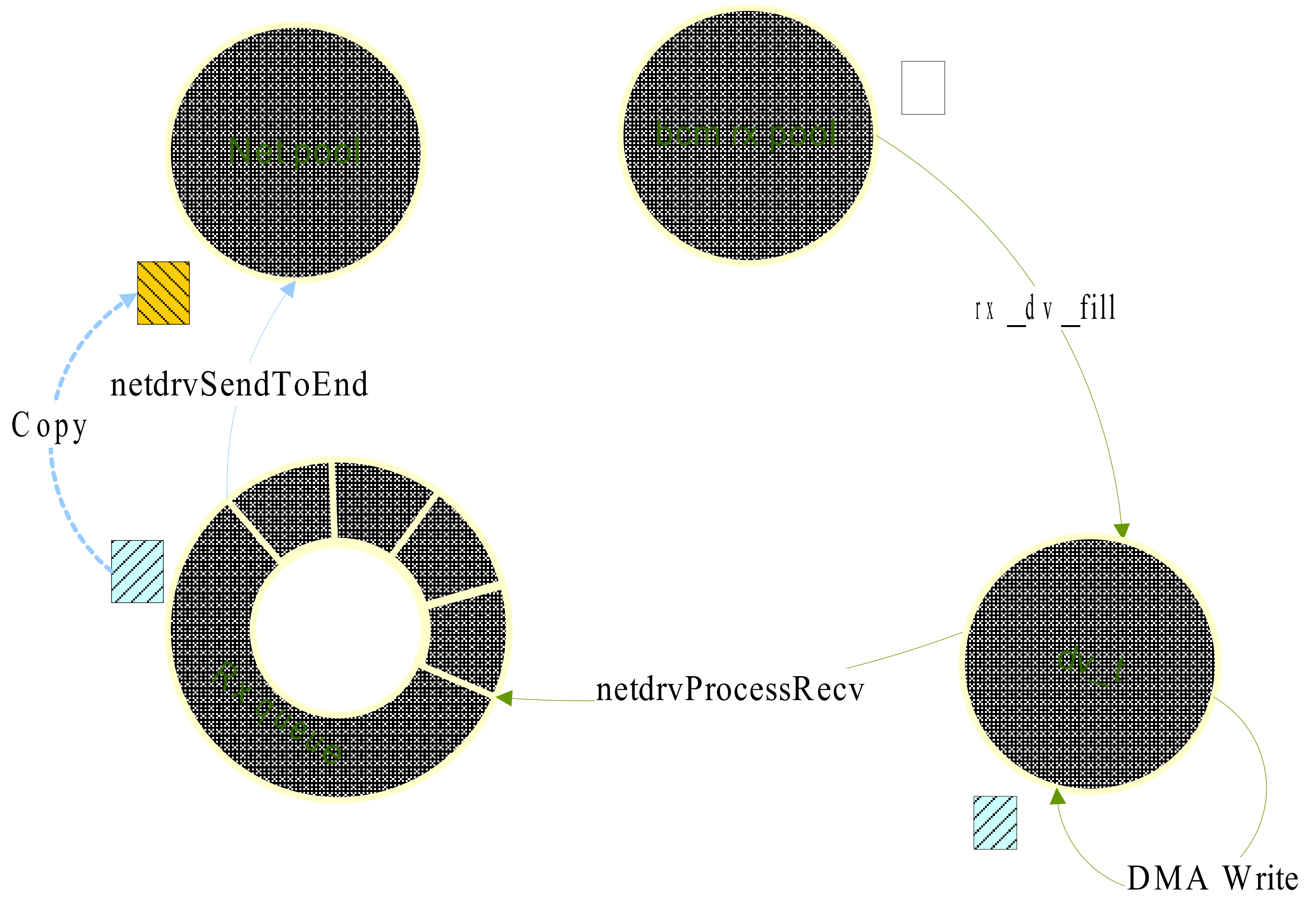
数据包接收流程


application thread
rx _pkt_thread
intr context
bcm_rx_start
rx_thread_start
loop
rx_thread_dv_check
DMA Fill
wait pkt_notify
pkt_notify
while !RXQ_EMPTY
RXQ_DEQUEUE
NetdrvSendToEnd
CHAN DONE EVENT
soc_intr
soc_dma_done_chain

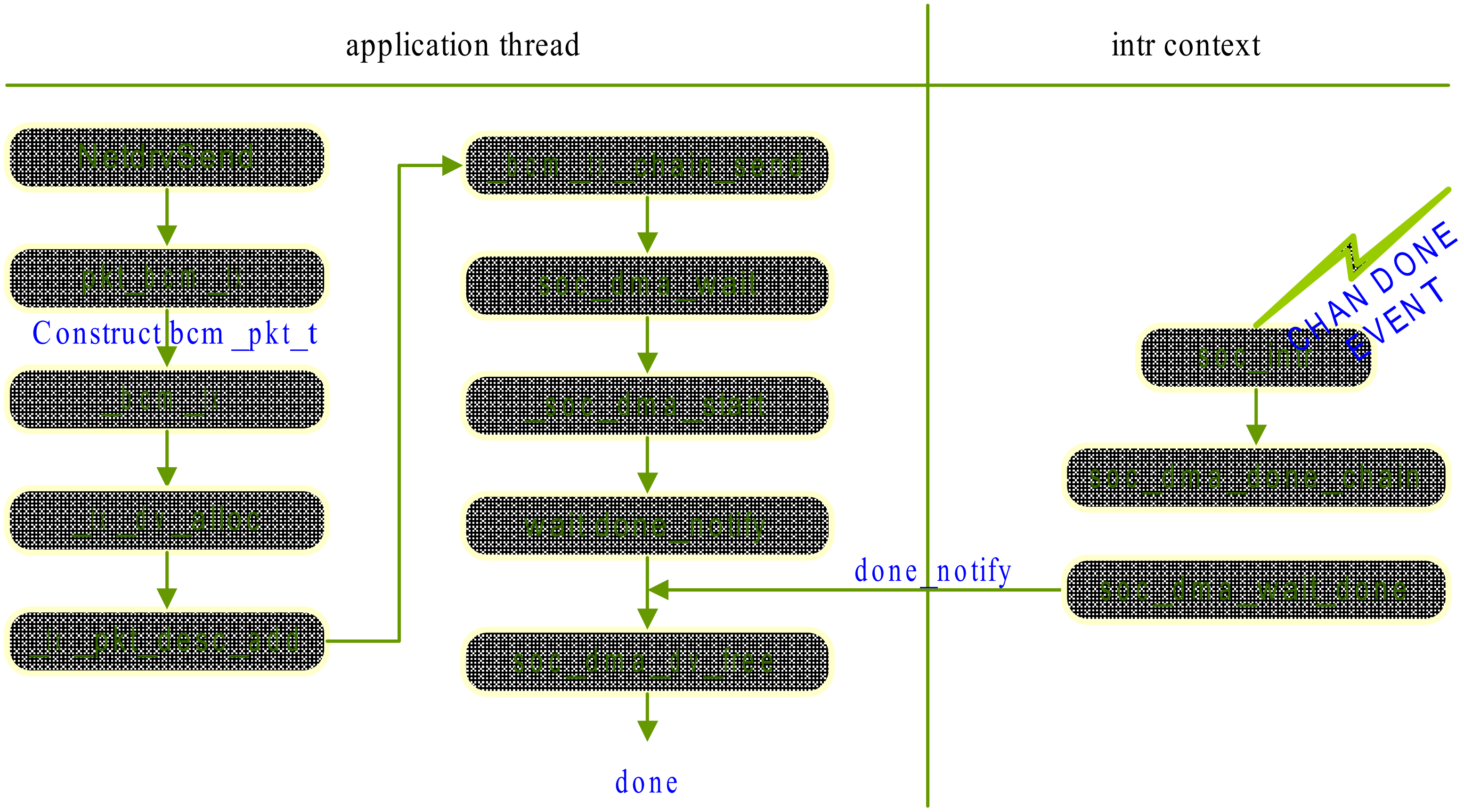
数据包接收流程


soc_dma_done_chain
soc_intr_disable
_soc_dma_start_channel
soc_dma_process_done_desc
dv_chain->dv_done_packet
rx_intr_process_packet
DCB PARSE
netdrvProcessRecv
RXQ_ENQUEUE
RX_THREAD_NOTIFY

数据包接收流程


Net pool
bcm rx pool
rx_dv_fill
netdrvSendToEnd
Copy
rx queue
netdrvProcessRecv
dv_t
DMA Write

数据包发送流程


application thread
intr context
NetdrvSend
pkt_bcm_lx
Construct bcm _pkt_t
_bcm_lx
_lx_dv_alloc
_lx_pkt_desc_add
_bcm_lx_chain_send
soc_dma_wait
_soc_dma_start
wait done_notify
soc_dma_dv_free
done
done_notify
CHAN DONE EVENT
soc_intr
soc_dma_done_chain
soc_dma_wait_done

END性能优化

- Hardware Layer
- Software Layer
  - Pooled Memory alloc & free
  - Avoid Memory Copy
  - Keep Busy
  - Others

Hawkeye END缺陷？

- 不必要的**DCB**解析操作
- **Synchronized Tx**

END移植

- **SOC**操作函数实现
- 初始化流程
- 确定**DCB**格式