ICT NCIC 系统互连组

控制平面（CEU）说明文档

v1.0-b

Revision History

Date	Author	Comment	Version
20/06/2020	康宁	1. 完成 CEU 接口数据流信息的说明 2. 完成了对 AXI 总线连接及模块相应存储内容的设计 3. 完成了 IB HCA 相关信息存储的设计	v0.1-b
10/07/2020	康宁	1. 描述了 CEU 状态机的状态转移过程	v0.5-a
22/07/2020	康宁	1. 修改了 CEU 状态机部分内容	v0.6-a
14/09/2020	康宁	1. 修改 CEU 接口，添加 head 信号，修改数据包（头）格式 2. 添加 CEU 模块结构图，修改 CEU 状态机 3. 取消 AXI 接口，原来与 AXI 接口相关命令直接在内部解析执行	v1.0-a

1 控制平面（CEU）总体说明

控制平面是软件对 IB HCA 的控制操作的通路，包含软件驱动（内核态和部分用户态驱动）、软硬件接口、IBA 管理架构（暂未实现）、CEU 以及通过 CEU 配置的相关存储（位于 CEU 内部，用于查询设备属性，配置端口信息等）。其中，软硬件接口详细内容请参见软硬件接口文档 v1.2-a，IBA 管理架构目前暂不涉及，本文主要对 CEU 以及通过 CEU 配置的 HCA 内相关寄存器的基本结构及连接关系进行说明。

2 CEU 模块及 CEU 相关存储

CEU 是 IB HCA 控制路径上的核心组件，其负责响应软件下发的所有命令。这些命令包括：初始化命令，配置信息查询及写入命令，队列（EQ，CQ，QP）创建及修改、查询命令等。命令详细内容见软硬件接口文档 v1.2-a。

2.1 CEU 模块接口

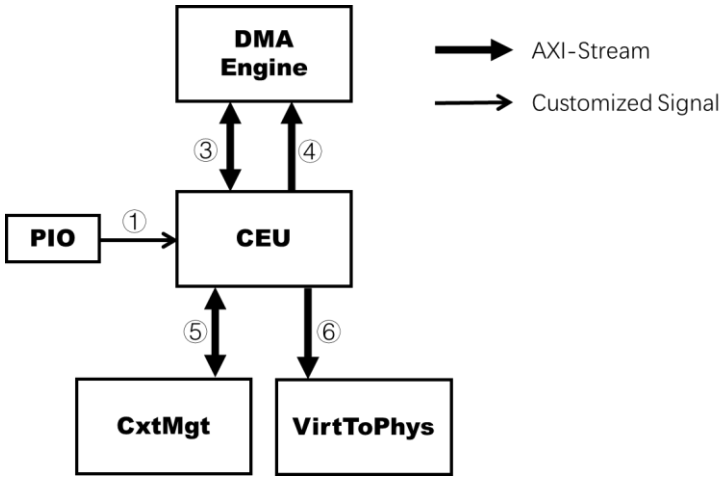

图 2-1 CEU 与其他模块连接关系

CEU 与其他模块连接关系如图 2-1 所示。图中，CEU 模块是子系统的核心模块，它用来解析、执行 HCR 命令。

①用于连接 PIO 模块，访问 HCR 寄存器中的内容，从而执行命令。该总线使用自定义的数据格式，其信号格式如表 2-1 所示。

表 2-1 CEU 与 MemAccCtl 的接口及含义

信号	in/out	位宽	描述
in_param	in	64	输入参数或系统用于存放输入参数的地址
in_modifier	in	32	in_param 的修饰符
out_dma_addr	in	64	系统用于存放输出参数的地址
out_param	out	64	输出参数（暂不实现）

token	in	32	命令的令牌符
status	out	8	命令执行的状态报告（暂不实现）
go	in	1	1'b1 代表新命令到达
clear	out	1	1'b1 代表命令执行完成
event	in	1	1'b1 代表命令执行完成要向 EQ 报告（暂不实现）
op_modifier	in	8	操作码修饰符
op	in	12	命令操作码

③是读取 input mailbox 的接口，该接口由两个流式接口组成，一个用于向 DMA Engine 发送读请求（CEU 为 Master），另一个用于接收 DMA Engine 读到的数据（CEU 为 Slave）。这两个流式接口都分别仅有一种数据流类型，其中，读请求接口数据流及读返回接口数据流要求的信息内容如表 2-2 所示。其中，Requester 包含的数据包头如表 2-3 所示，Responder 仅包含响应数据，没有数据包头。

表 2-2 input mailbox 两个 AXIS 接口所含信号内容

流式接口	信号	位宽(bit)/方向	信号说明
Inbox Read Requester	dma_rd_req_valid	1 / out	dma 读请求有效位
	dma_rd_req_last	1 / out	dma 读请求最后一个 cycle
	dma_rd_req_head	128 / out	dma 读请求头，仅在第一个传输周期有效，格式见表 2-3
	dma_rd_req_data	256 / out	dma 读请求数据，恒为 0
	dma_rd_req_ready	1 / in	dma 读请求接收方准备就绪
Inbox Read Responder	dma_rd_rsp_valid	1 / in	dma 读响应有效位
	dma_rd_rsp_last	1 / in	dma 读响应最后一个 cycle
	dma_rd_rsp_head	128 / in	dma 读响应头，仅在第一个传输周期有效，格式见表 2-3
	dma_rd_rsp_data	256 / in	dma 读响应数据
	dma_rd_rsp_ready	1 / out	dma 读响应接收方准备就绪

表 2-3 input mailbox Read Request 流式接口的数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch																																
08h	inbox_addr[63:32]																															
04h	inbox_addr[31:0]																															
00h																	inbox_len															

④是写 output mailbox 的接口，该接口由一个流式接口组成，用于向 DMA Engine 发送写请求（CEU 为 Master）。该接口仅有一种类型的数据流，即写 output mailbox，该接口中数

据流要求的信息内容如表 2-4 所示。

表 2-4 output mailbox 写请求流式接口所需信息内容

流式接口	信号	位宽(bit)/方向	信号说明
Outbox Write Requester	dma_wr_req_valid	1 / out	dma 写请求有效位
	dma_wr_req_last	1 / out	dma 写请求最后一个 cycle
	dma_wr_req_head	128 / out	dma 写请求头，仅在第一个传输周期有效，格式见表 2-5
	dma_wr_req_data	256 / out	dma 写请求写向 outbox 的数据
	dma_wr_req_ready	1 / in	dma 写请求接收方准备就绪

表 2-5 output mailbox Write Request 流式接口的数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch																																
08h	outbox_addr[63:32]																															
04h	outbox_addr[31:0]																															
00h																	outbox_len															

⑤是读写上下文的接口，该接口由两个流式接口组成，一个用于向 CxtMgt 模块发送请求（读请求或写请求），另一个用于 CxtMgt 响应 CEU 的请求（读响应）。其中，请求接口一共有 12 种类型的数据流，分别是：1). 请求读取一个 QP 上下文条目（CMD_QUERY_QP，见表 2-9）；2). 修改 QP 上下文的一个条目（CMD_MODIFY_QP，见表 2-10）；3). 写 CQ 上下文的一个条目（CMD_SW2HW_CQ，见表 2-11）；4). 修改 CQ 上下文的一个条目（CMD_RESIZE_CQ，见表 2-12）；5). 无效 CQ 上下文的一个条目（CMD_HW2SW_CQ，见表 2-13）；6). 写 EQ 上下文的一个条目（CMD_SW2HW_EQ，见表 2-14）；7). 使能或失能 EQ 上下文一个条目的相关功能（CMD_MAP_EQ，见表 2-15）；8). 无效 EQ 上下文的一个条目（CMD_HW2SW_EQ，见表 2-16）；9). 将 QP，CQ，EQ 上下文资源的 ICM 虚拟起始地址及大小写入 CxtMgt（CMD_INIT_HCA，见表 2-17）；10). 将 QP，CQ，EQ 上下文资源的 ICM 虚拟起始地址及大小无效（CMD_CLOSE_HCA，见表 2-18）；11). 将 QP，CQ，EQ 上下文资源的 ICM 虚拟地址对应的物理地址写入 CxtMgt（CMD_MAP_ICM，见表 2-19）；12). 将 QP，CQ，EQ 上下文资源的 ICM 虚拟地址对应的物理地址无效（CMD_UNMAP_ICM，见表 2-20）。而响应流式接口只有一种数据流类型：1). 返回要读取的一个 QP 上下文条目（CMD_QUERY_QP，见表 2-9）。该接口中请求流式接口和响应流式接口要求的信息内容如表 2-6 所示。其中，数据流 DATA 信号的详细字段信息将在 CxtMgt 文档中进行说明，而 HEAD 信号的详细字段信息见表 2-8～表 2-20。

表 2-6 读写上下文的两个流式接口的信号

流式接口	信号	位宽(bit)/方向	信号说明
读写上下文请求接口	cm_req_valid	1 / out	Context Management（读写）请求有效位
	cm_req_last	1 / out	Context Management（读写）请求最后一个 cycle
	cm_req_head	128 / out	Context Management（读写）请求头，仅在第一个传输周期有效，通用格式内容信息见表 2-7
	cm_req_data	256 / out	Context Management（读写）请求写向 CM 的数据
	cm_req_ready	1 / in	Context Management（读写）请求接收方准备就绪
读写上下文响应接口	cm_rsp_valid	1 / in	Context Management（读写）响应有效位
	cm_rsp_last	1 / in	Context Management（读写）响应最后一个 cycle
	cm_rsp_head	128 / in	Context Management（读写）响应头，仅在第一个传输周期有效，通用格式内容信息见表 2-7
	cm_rsp_data	256 / in	Context Management（读写）响应写向 CEU 的数据
	cm_rsp_ready	1 / out	Context Management（读写）响应接收方准备就绪

表 2-7 读写上下文的两个流式接口 HEAD 信号的通用格式内容

AXIS 接口	所需信息	位宽	信息说明
读写上下文请求接口	req_cxt_type	4 bit	请求类型： RD_QP_CXT WR_QP_CXT WR_CQ_CXT WR_EQ_CXT WR_ICMMAP_CXT MAP_ICM_CXT
	req_cxt_opcode	4bit	该请求类型要实现的具体操作，详细内容参见表 2-8
	req_cxt_addr	32 bit	请求要访问的上下文资源的条目 index（QP 号，CQ 号，EQ 号，ICM 虚拟地址） WR_ICMMAP_CXT 不需要该字段

读写上下文响应接口	rsp_cxt_type	4bit	请求类型： RD_QP_CXT
	rsp_cxt_opcode	4bit	该请求类型要实现的具体操作，详细内容参见表 2-8
	rsp_cxt_addr	32bit	请求要访问的上下文资源的条目 index（QP 号）

表 2-8 读写上下文请求接口不同上下文类型对应的操作码

上下文请求类型	req_cxt_opcode	说明
RD_QP_CXT	RD_QP_ALL	CEU 使用，读取一个上下文条目
	RD_QP_SQ	RDMA 引擎使用，读取 SQ 头指针
	RD_QP_RQ	RDMA 引擎使用，读取 RQ 头指针
WR_QP_CXT	WR_QP_ALL	CEU 使用，写一个 QP 上下文条目
RD_CQ_CXT		RDMA 引擎使用
WR_CQ_CXT	WR_CQ_ALL	CEU 使用，写一个 CQ 上下文条目
	WR_CQ_MODIFY	CEU 使用，修改 CQ 上下文条目
	WR_CQ_INVALID	CEU 使用，将一个 CQ 上下文条目无效
RD_EQ_CXT		RDMA 引擎使用
WR_EQ_CXT	WR_EQ_ALL	CEU 使用，写一个 EQ 上下文条目
	WR_EQ_FUNC	CEU 使用，使能或失能 EQ 条目的事件功能
	WR_EQ_INVALID	CEU 使用，将一个 EQ 上下文条目无效
WR_ICMMAP_CXT	WR_ICMMAP_EN	CEU 使用，将上下文资源表写入 CxtMgt
	WR_ICMMAP_DIS	CEU 使用，无效写入的上下文资源表
MAP_ICM_CXT	MAP_ICM_EN	CEU 使用，将上下文资源映射的物理地址写入 CxtMgt
	MAP_ICM_DIS	CEU 使用，无效写入的上下文资源表

表 2-9 CMD_QUERY_QP 请求及响应数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	QP_num[31:0]																															
04h																																
00h																																

表 2-10 CMD_MODIFY_QP 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	QP_num[31:0]																															
04h																																

00h	

表 2-11 CMD_SW2HW_CQ 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	CQ_num[31:0]																															
04h																																
00h																																

表 2-12 CMD_RESIZE_CQ 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	CQ_num[31:0]																															
04h																																
00h																																

表 2-13 CMD_HW2SW_CQ 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	CQ_num[31:0]																															
04h																																
00h																																

表 2-14 CMD_SW2HW_EQ 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	EQ_num[31:0]																															
04h																																
00h																																

表 2-15 CMD_MAP_EQ 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	EQ_num[31:0]																															
04h	event_mask[63:32]																															
00h	event_mask[31:0]																															

表 2-16 CMD_HW2SW_EQ 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	QP_num[31:0]																															

04h	
00h	

表 2-17(a) CMD_INIT_HCA 请求数据包头格式(CxtMgt & V2P)

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h																																
04h																																
00h																																

表 2-17(b) CMD_INIT_HCA 请求数据包格式(CxtMgt)

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1Ch																																
18h																																
14h	qpc_base[63:32]																															
10h	qpc_base[31:8]																								log_num_qps							
0Ch	cqc_base[63:32]																															
08h	cqc_base[31:8]																								log_num_cqs							
04h	eqc_base[63:32]																															
00h	eqc_base[31:8]																								log_num_eqs							

表 2-18 CMD_CLOSE_HCA 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h																																
04h																																
00h																																

表 2-19 CMD_MAP_ICM 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	chunk_num[31:0]																															
04h																																
00h																																

表 2-20 CMD_UNMAP_ICM 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	page_cnt[31:0]																															
04h	virt[63:32]																															
00h	virt[31:0]																															

⑥是写 TPT 的 AXIS 接口，该接口由一个 AXIS 接口组成，一个用于向对接模块内的 Req Channel 发送写请求（CEU 为 Master）。写请求的类型包括：1). 向 MPT 表中写入一个表项（CMD_SW2HW_MPT，见表 2-23）；2). MPT 表中的一个表项无效（CMD_HW2SW_MPT，见表 2-24）； 3). 向 MTT 表写入一系列物理地址表项（CMD_WRITE_MTT，见表 2-25）；4). 将 MPT，MTT 的 ICM 虚拟起始地址及大小写入 VirtToPhys（CMD_INIT_HCA，见表 2-26）；5). 将 MPT，MTT 的 ICM 虚拟起始地址及大小无效（CMD_CLOSE_HCA，见表 2-18）；6). 将 MPT，MTT 的 ICM 虚拟地址对应的物理地址写入 VirtToPhys（CMD_MAP_ICM，见表 2-19）；12). 将 MPT，MTT 的 ICM 虚拟地址对应的物理地址无效（CMD_UNMAP_ICM，见表 2-20）。该接口中数据流要求的信息如表 2-21 所示。其中，数据流详细字段信息将在 VirtToPhys 文档中进行说明。

表 2-21(a) 写 TPT 的流式接口中的信号内容

流式接口	信号	位宽(bit)/方向	信号说明
写 TPT 请求接口	v2p_req_valid	1 / out	Virtual to Physical 写请求有效位
	v2p_req_last	1 / out	Virtual to Physical 写请求最后一个 cycle
	v2p_req_head	128 / out	Virtual to Physical 写请求头，仅在第一个传输周期有效，通用格式内容信息见表 2-21(b)
	v2p_req_data	256 / out	Virtual to Physical 写请求写向 V2P 的数据
	v2p_req_ready	1 / in	Virtual to Physical 写请求接收方准备就绪

表 2-21(b) 写 TPT 的流式接口中 HEAD 信号的通用格式内容

AXIS 接口	所需信息	位宽	信息说明
TPT 写请求	req_tpt_type	4 bit	请求类型： WR_MPT_TPT WR_MTT_TPT WR_ICMMAP_TPT MAP_ICM_TPT
	req_tpt_opcode	4 bit	该请求类型具体实现的操作，详细内容参见表 2-22
	req_tpt_addr	10 bit	请求要访问的上下文资源的条目 index（MTT index，MPT index，ICM 虚拟地址） WR_ICMMAP_TPT 不需要

表 2-22 TPT 写请求接口不同资源类型对应的操作码

上下文请求类型	req_cxt_opcode	说明
WR_MPT_TPT	WR_MPT_WRITE	CEU 使用，向 MPT 表中写入一个表项
	WR_MPT_INVALID	CEU 使用，MPT 表中的一个表项无效

WR_MTT_TPT	WR_MTT_WRITE	CEU 使用，向 MTT 表中写入一系列表项
	WR_MTT_INVALID	CEU 使用，将 MTT 表中一系列表项无效
WR_ICMMAP_TPT	WR_ICMMAP_EN	CEU 使用，将 TPT 资源表写入 VirtToPhys
	WR_ICMMAP_DIS	CEU 使用，无效写入的 TPT 资源表
MAP_ICM_TPT	MAP_ICM_EN	CEU 使用，将 TPT 资源映射的物理地址写入 VirtToPhys
	MAP_ICM_DIS	CEU 使用，无效写入的 TPT 资源表

表 2-23 CMD_SW2HW_MPT 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	mpt_index[31:0]																															
04h																																
00h																																

表 2-24 CMD_HW2SW_MPT 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	mpt_index[31:0]																															
04h																																
00h																																

表 2-25 CMD_WRITE_MTT 请求数据包头格式

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h	mtt_num[31:0]																															
04h	mtt_start_index[63:32]																															
00h	mtt_start_index[31:0]																															

表 2-26(a) CMD_INIT_HCA 请求数据包 HEAD 格式（Virt2Phys）

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0Ch	type				opcode																											
08h																																
04h																																
00h																																

表 2-26(b) CMD_INIT_HCA 请求数据包格式(V2P)

offset	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1Ch																																
18h																																

14h		
10h		
0Ch	mpt_base[63:32]	
08h	mpt_base[31:8]	log_mpt_sz
04h	mtt_base[63:32]	
00h	mtt_base[31:0]	

2.2 CEU 模块逻辑说明

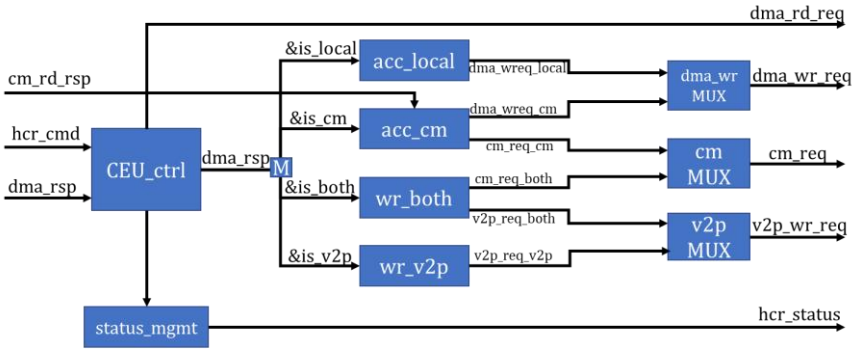

图 2-2 CEU 模块的整体结构图

CEU 模块的整体结构如图 2-2 所示。CEU 模块主要包含 5 个子模块：CEU_ctrl，acc_local，acc_cm，wr_both，wr_v2p。剩下的 status_mgmt 模块用于 hcr 寄存器的输出管理；三个 MUX 模块为子模块信号的数据选择器，选择输出到模块外部。下面对 5 个主要的子模块进行详细说明。

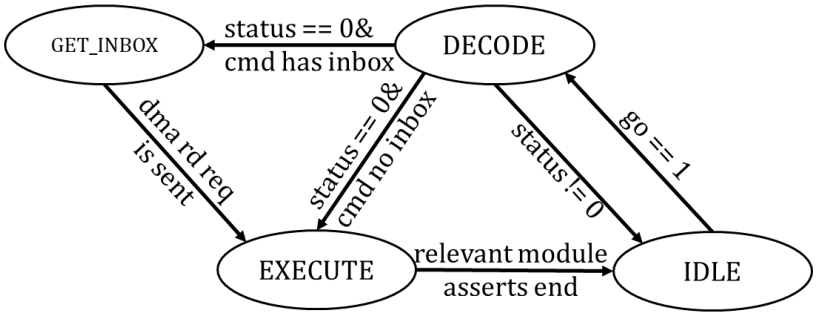

图 2-3 CEU_ctrl 状态转移图

CEU_ctrl：用于控制整个 CEU 状态机的执行流程。CEU_ctrl 模块的状态转移如图 2-3 所示。该状态机共包含 4 个状态，其中，IDLE 用于等待 HCR 命令的到来；DECODE 状态用于对到来的命令进行解码，同时对命令的正确性进行验证；GET_INBOX 状态用于对需要 INBOX 的命令发送 INBOX 读请求操作；EXECUTE 状态用于开始具体执行命令操作。

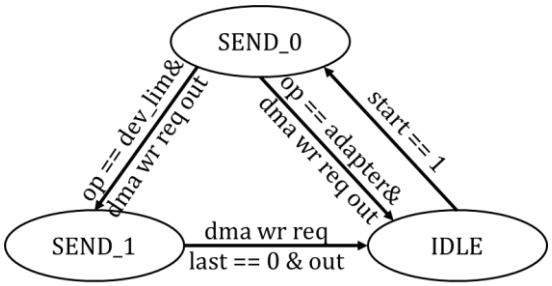

图 2-4 acc_local 状态转移图

acc_local：用于执行仅需要访问 CEU 本地存储的命令，目前，这些命令包括：不含 inbox，含 outbox：CMD_QUERY_DEV_LIM，CMD_QUERY_ADAPTER。以上两个命令读取 CEU 本地存储，并将读取的内容写到了 outbox 中。acc_local 状态转移图如图 2-4 所示。

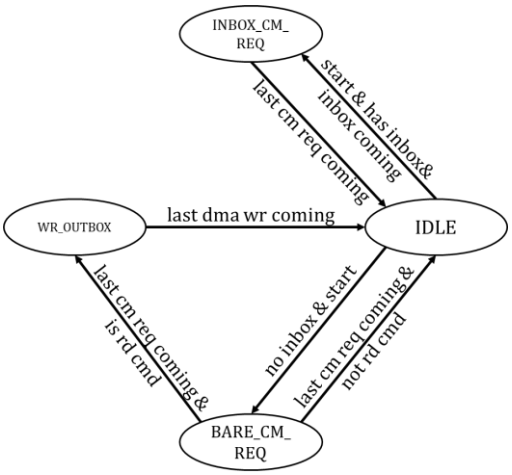

图 2-5 acc_cm 状态转移图

acc_cm：用于传输仅会访问 Context Management 模块的命令,这些命令包括：含 inbox，不含 outbox：CMD_MAP_ICM，CMD_SW2HW_CQ，CMD_RESIZE_CQ，CMD_SW2HW_EQ；不含 inbox，不含 outbox：CMD_UNMAP_ICM，CMD_MAP_EQ，CMD_HW2SW_CQ，CMD_HW2SW_EQ；不含 inbox，含 outbox：CMD_QUERY_QP；含或不含 inbox，不含 outbox：IS_MODIFY_QP。acc_cm 的状态转移图如图 2-5 所示。

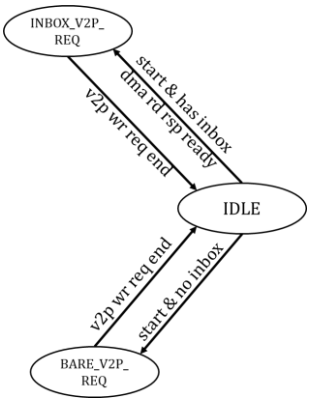

图 2-6 wr_v2p 模块状态转移图

wr_v2p：用于传输访问 Virtual to Physical 模块的命令，解析的命令包括：含 inbox，不含

outbox: CMD_MAP_ICM，CMD_SW2HW_MPT，CMD_WRITE_MTT；不含 inbox，不含 outbox: CMD_UNMAP_ICM，CMD_HW2SW_MPT。wr_v2p 的状态转移图如图 2-6 所示。

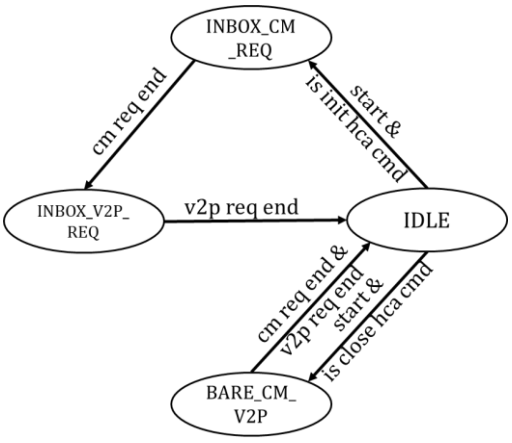

图 2-7 wr_both 状态转移图

wr_both：用于传输同时会访问 Context Management 模块和 Virtual to Physical 模块的命令，这些命令包括：含 inbox，不含 outbox: CMD_INIT_HCA；不含 inbox，不含 outbox: CMD_CLOSE_HCA。wr_both 的状态转移图如图 2-7 所示。