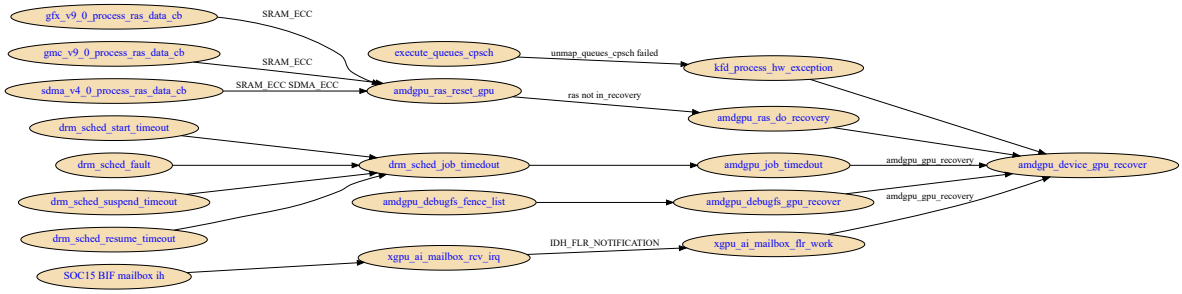
GPU 故障修复流程分析

1、何时触发故障修复?

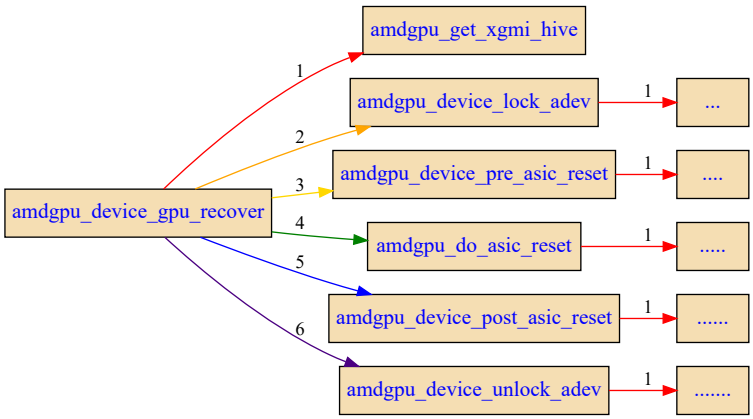


模块参数amdgpu_gpu_recovery开关可以控制部分场景是否做修复。

以下场景触发修复：

1）执行cp队列调度，unmap_queues_cpsch失败的时候；
2）gfx/gmc/sdma收到ecc相关的ras错误信息；
3）drm_sched_xxx相关调度超时、fault、挂起超时的时候；
4）通过debugfs注入（cat /sys/kernel/debug/dri/x/amdgpu_gpu_recover），主动进入修复流程；
5）mailbox中断处理，如果收到BIF mailbox ih中断事件为IDH_FLR_NOTIFICATION复位通知，则启动修复；

2、gpu recovery的主要步骤



1）获取hive, 如果xgmi.num_physical_nodes>1，则trylock hive->reset_lock (num_physical_nodes从mmMC_VM_XGMI_LFB_CNTL读取出来的，因为full_reset会重置hive中的所有节点，获取hive->reset_lock锁避免多个node间的并发)；

2）首先lock当前adev->lock_reset，执行kfd_pre_reset；执行pre asic reset；

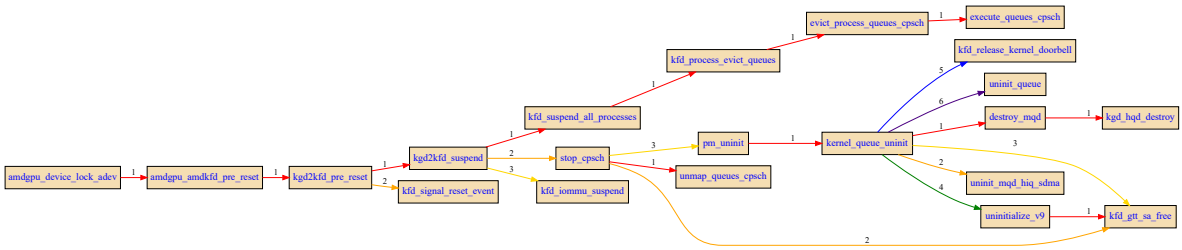
3）如果是full reset（当GMC，SMC，ACP，PSP中任何一个hang住的时候必定会full reset），将该hive下的所有adapter都执行一遍amdgpu_device_lock_adev，amdgpu_device_pre_asic_reset；

4）amdgpu_do_asic_reset；full reset会对hive下的所有adev执行一次复位；

5）amdgpu_device_post_asic_reset（full reset下遍历hive下所有的adev），遍历adev下所有队列，对队列调度器的ring_mirror_list上挂起的jobs执行resubmit，如果不支持Display Core Driver（支持），则强制还原CRTC到旧的mode setting configuration；

6) 遍历hive下所有的adev完成kfd_post_reset，kfd_resume；unlock adev->lock_reset ，最后释放hive->reset_lock；

3、kfd pre reset



lock住adev->lock_reset后主要执行kfd_pre_reset操作：

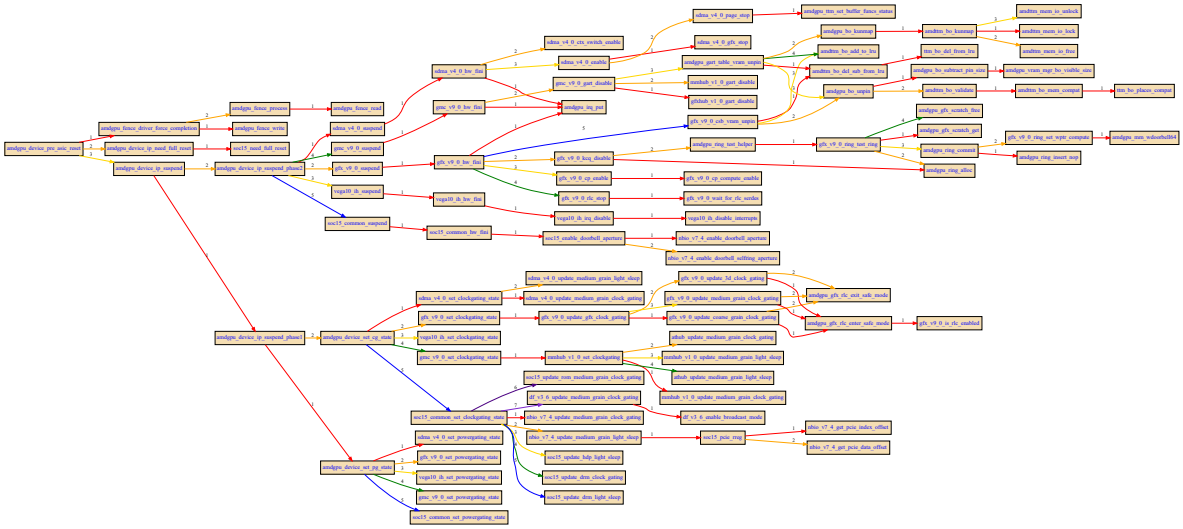
1）在kfd pre reset首先挂起所有的process，将process的queue从cp调度器中淘汰出去；通过execute_queues_cpsch UNMAP ALL QUEUES（sdma queue，computer queue）；

2）stop cp schedule

3）suspend iommu

4）signal reset event

4、pre asic reset



asic pre reset将设备各个ip挂起，主要分2步骤完成（amdgpu_device_ip_suspend_phase1、amdgpu_device_ip_suspend_phase2）

4.1 phase1

将soc common (DF/NBIO/HDP/DRM/ROM)、gmc、ih、gfx、sdma 各IP的power gating、clock gating迁移到UN_GATE状态（disable），由于vega20 不具备power gate特性，power gate状态迁移都未做处理；各IPclock gating状态迁移如下：

1）SOC：DF、NBIO、HDP、DRM、ROM关闭clock gating，NBIO及DRM还会关闭light sleep；

2）GMC：关闭**mmhub** medium grain **clock gating**；关闭**athub** medium grain **clock gating**；关闭**mmhub** medium grain mem **light sleep**；关闭athub medium grain mem **light sleep**；

3）IH：无

4）GFX：关闭粗粒度clock gating（CGCG + CGLS）；3d clock gating（CGCG /CGLS for GFX 3D）；中等粒度clock gating（MGCG + MGLS）；

5）SDMA：关闭中等粒度clock gating（mmSDMA0_CLK_CTRL、mmSDMA1_CLK_CTRL）；disable sdma0 & sdma1 mem light sleep（mmSDMA0_POWER_CNTL，mmSDMA1_POWER_CNTL）

IP		ACTION	REG
COMMON	DF	disable DF medium grain clock gating Exit broadcast mode	mmDF_PIE_AON0_DfGlobalClkGater mmFabricConfigAccessControl
	NBIO	disable NBIO medium grain clock gating disable NBIO medium grain light_sleep	None（vega20 NBIO无clock gating特性） smnPCIE_CNTL2
	HDP	disable HDP light sleep	mmHDP_MEM_POWER_CTRL
	DRM	disable DRM clock gating disable DRM light_sleep	mmMP0_MISC_CGTT_CTRL0 mmMP0_MISC_LIGHT_SLEEP_CTRL
	ROM	disable ROM medium grain clock gating	mmCGTT_ROM_CLK_CTRL0 of SMUIO
GMC		disable **mmhub** medium grain clock gating	mmATC_L2_MISC_CG、 mmDAGB0_CNTL_MISC2、 mmDAGB1_CNTL_MISC2
		disable **athub** medium grain clock gating	mmATHUB_MISC_CNTL
		disable **mmhub** medium grain lock gating	mmATC_L2_MISC_CG
		disable **athub** medium grain mem light sleep	mmATHUB_MISC_CNTL
GFX		disable GFX coarse grain clock gating	mmRLC_CGCG_CGLS_CTRL
		disable GFX 3D clock gating	mmRLC_CGCG_CGLS_CTRL_3D
		disable GFX medium grain clock gating	mmRLC_CGTT_MGCG_OVERRIDE、 mmRLC_MEM_SLP_CNTL、 mmCP_MEM_SLP_CNTL
SDMA		disable SDMA medium grain clock gating	mmSDMA0_CLK_CTRL、 mmSDMA1_CLK_CTRL
		disable SDMA medium grain mem light sleep	mmSDMA0_POWER_CNTL、 mmSDMA1_POWER_CNTL

4.2 phase2

依次执行sdma、gfx、ih、gmc，soc各个的hw_fini挂起各个IP（次序和phase1相反）：

1）sdma： sdma_v4_0_hw_fini, 释放sdma ecc_irq INSTANCE0 & INSTANCE1的引用计数；停止DMA引擎上下文切换（mmSDMA0_CNTL / mmSDMA1_CNTL）；关闭SDMA（mmSDMA0_F32_CNTL / mmSDMA1_F32_CNTL）；

2）gfx：gfx_v9_0_hw_fini，释放cp_ecc_error_irq、priv_reg_irq、priv_inst_irq中断的引用计数；

disable KCQ 避免CPC访问memory（封装num_compute_rings个PACKET3_UNMAP_QUEUES PM4包，通过kiq队列发送到CPC）；关闭MEC（mmCP_MEC_CNTL，MEC1/MEC2）；关闭RLC（RLC_CNTL）；

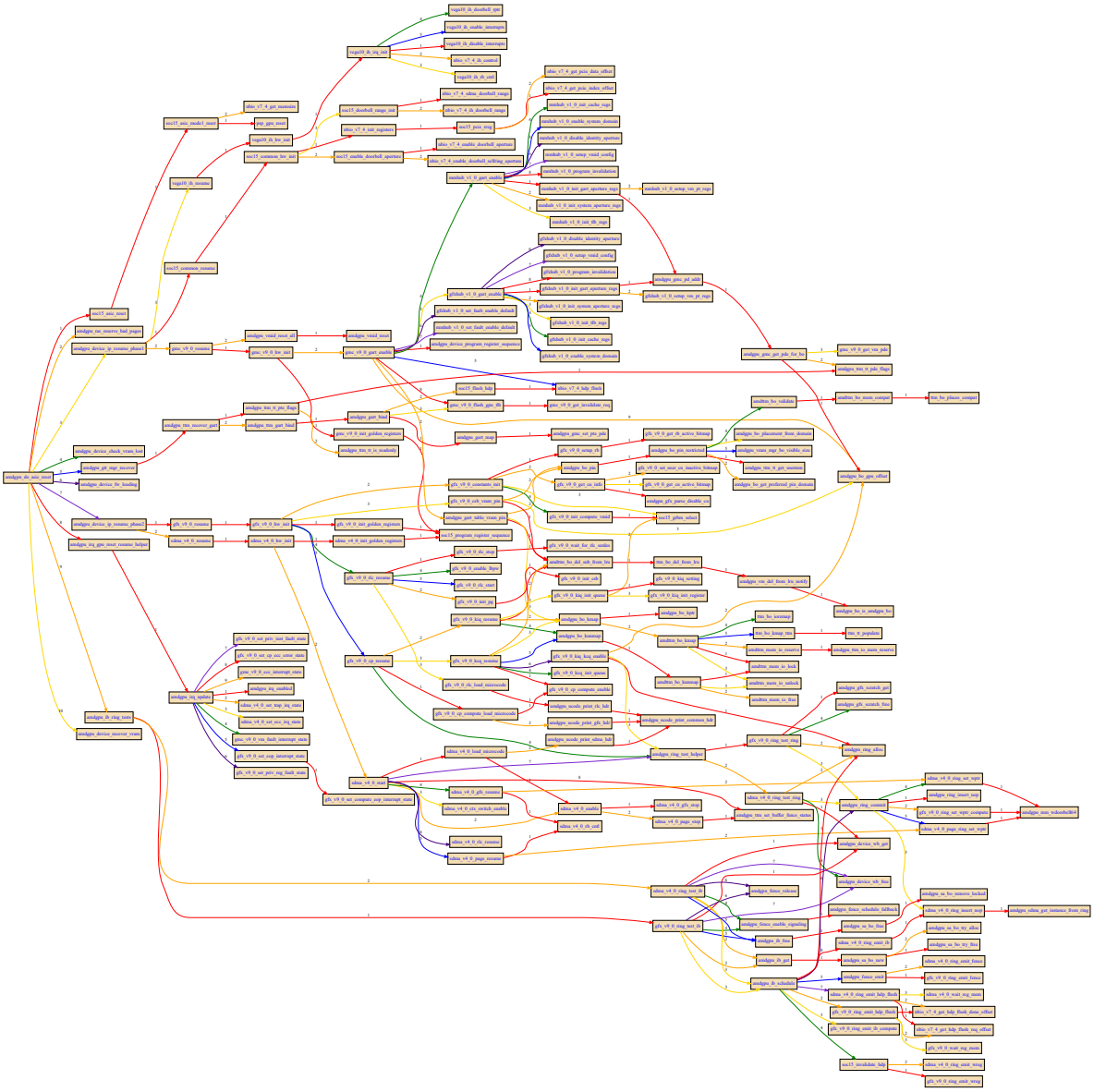
3）ih：vega10_ih_hw_fini，关闭中断disable ring buffer（mmIH_RB_CNTL，mmIH_RB_RPTR，mmIH_RB_WPTR，mmIH_RB_CNTL_RING1，mmIH_RB_RPTR_RING1，mmIH_RB_WPTR_RING1；mmIH_RB_CNTL_RING2，mmIH_RB_RPTR_RING2，mmIH_RB_WPTR_RING2）；

4）gmc： gmc_v9_0_hw_fini，释放gmc.ecc_irq、gmc.vm_fault中断引用计数; disable gfxhub gart table（mmVM_CONTEXT0_CNTL...mmVM_CONTEXT15_CNTL, mmMC_VM_MX_L1_TLB_CNTL, mmVM_L2_CNTL3）; disable mmhub gart table (mmVM_CONTEXT0_CNTL...mmVM_CONTEXT15_CNTL, mmMC_VM_MX_L1_TLB_CNTL, mmVM_L2_CNTL, mmVM_L2_CNTL3)

5）soc： soc15_common_hw_fini, disable the doorbell aperture，写nbio的RCC_DOORBELL_APER_EN关闭doorbell aperture，写nbio的mmDOORBELL_SELFRING_GPA_APER_CNTL关闭doorbell_selfring_aperture；

IP	ACTION	REG/PACKET
SDMA	disable sdma ctx switch	mmSDMA0_CNTL mmSDMA1_CNTL
	disable sdma	mmSDMA0_F32_CNTL mmSDMA1_F32_CNTL
GFX	disable kcq	PACKET3_UNMAP_QUEUES
	disable mec	mmCP_MEC_CNTL
	stop rlc	RLC_CNTL
IH	disable the interrupt ring buffer	mmIH_RB_CNTL mmIH_RB_RPTR mmIH_RB_WPTR mmIH_RB_CNTL_RING1 mmIH_RB_RPTR_RING1 mmIH_RB_WPTR_RING1 mmIH_RB_CNTL_RING2 mmIH_RB_RPTR_RING2 mmIH_RB_WPTR_RING2
GMC	disable gfxhub gart table	mmVM_CONTEXT0_CNTL… mmVM_CONTEXT15_CNTL mmMC_VM_MX_L1_TLB_CNTL mmVM_L2_CNTL3
	disable mmhub gart table	mmVM_CONTEXT0_CNTL … mmVM_CONTEXT15_CNTL mmMC_VM_MX_L1_TLB_CNTL mmVM_L2_CNTL mmVM_L2_CNTL3
SOC	disable nbio doorbell aperture	RCC_DOORBELL_APER_EN
	disable nbio doorbell selfring aperture	mmDOORBELL_SELFRING_GPA_APER_CNTL

5、do asic reset



1）do_asic_reset遍历hive下所有的adev，并发执行 asic reset，发送mode1 reset命令到psp（寄存器mmMP0_SMN_C2PMSG_64，命令GFX_CTRL_CMD_ID_MODE1_RST）执行asic reset；reset后读取mmRCC_CONFIG_MEMSIZE（如果非0xffffffff）判断复位完成；

2）do_asic_reset遍历hive下所有的adev，通过device ip resume操作执行各个IP的**hw_init**，**device ip_resume_phase1** 初始化SOC、IH、GMC； **device ip_resume_phase2**初始化GFX、SDMA；

3）**device ip_resume_phase1**执行完成后，检查check vram lost（读取vram gart table 0位置，对比保存在adev中的64字节 vram magic）；恢复gtt管理器中的mm_node，将gtt page rebind到gart表（更新页表）；在fw加载方式为psp方式的情况下loading fw（需要PSP支持）；执行**device ip_resume_phase2**，hw init GFX、SDMA；重填vram magic（如果check vram lost）；更新xgmi拓扑信息；

4）执行asci 复位完成后，amdgpu_irq_gpu_reset_resume_helper遍历所有IP的中断源，更新中断状态到ENABLE状态**开启中断**，分别开启gmc ecc interrupt，sdma trap interrupt，sdma ecc interrupt、gmc vm fault interrupt、gfx eop interrupt、gfx priv reg fault interrupt、gfx priv inst fault、gfx cp ecc error interrupt；

5）恢复完成后，通过amdgpu_ib_ring_tests在所有的ring（gfx和smda ）上提交IB报文测试，以验证恢复是否成功；

6）最后一步amdgpu_device_recover_vram，通过SDMA copy，将adev->shadow_list上位于gtt域的bo还原到VRAM中；

5.1 phase1

1）SOC：soc15_common_hw_init，初始化nbio寄存器，配置nbio寄存器使能 doorbell aperture和doorbell selfring aperture；初始化SDMA和IH ring的doorbell范围；

2）IH：vega10_ih_hw_init，关闭ih 中断；设置nbio的中断控制寄存器（mmINTERRUPT_CNTL2，mmINTERRUPT_CNTL）；配置ih ring buffer（ring1，ring2），使能ih ring doorbell；开启ih 中断；

3）GMC：gmc_v9_0_hw_init，首先配置一系列的golden寄存器（mmhub，athub，hdp）；然后gmc_v9_0_gart_enable使能gart功能：

3.1）将gart page table pin在vram中；

3.2）gfxhub_v1_0_gart_enable：

3.2.1）初始化gart aperture寄存器，配置gart page table地址到gc的mmVM_CONTEXT0_PAGE_TABLE_BASE_ADDR，配置gart地址空间的范围到mmVM_CONTEXT0_PAGE_TABLE_START、mmVM_CONTEXT0_PAGE_TABLE_END寄存器；

3.2.2）初始化system aperture寄存器，设置AGP地址空间、SYSTEM APERTURE地址空间；

3.2.3）初始化TLB相关寄存器；

3.2.4）初始化cache相关寄存器（主要是L2 cache）；

3.2.5）enable system domain、disable identity aperture、setup vmid config；清零TLB缓存；

3.3）mmhub_v1_0_gart_enable：处理过程同gfxhub gart 使能，同名寄存器在gfxhub和mmhub各一份；

3.4）INVALIDATE HDP CACHE，flush hdp；配置gfxhub & mmhub的 GART / VM 的地址访问fault处理；

flush tlb；

5.2 phase2

1）GFX：gfx_v9_0_hw_init

1.1) gfx_v9_0_init_golden_registers: 配置一系列golden寄存器（gc，gc_vega20，gc_common）；

1.2) gfx_v9_0_constants_init:

1.2.1） 获取gfx active_rbs，获取cu info；

1.2.2）初始化vmid（0~7）寄存器，选择ME0，PIPE0，QUEUE0的VMID相关寄存器，设置内存对齐模式、关闭 RETRY（SH_MEM_CONFIG）；当vmid == 0时，mmSH_MEM_BASES配置为0；当vmid为1~7时，配置gmc.private_aperture_start和gmc.shared_aperture_start到mmSH_MEM_BASES寄存器；

1.2.3）gfx_v9_0_init_compute_vmid，初始化计算vmid（8~15），选择ME0，PIPE0，QUEUE0的VMID相关寄存器，配置地址模式（64bit）、对齐方式到mmSH_MEM_CONFIG寄存器，设置内存基地址0x60006000到mmSH_MEM_BASES；

1.2.4）选择为广播模式，将sc_prim_fifo_size_frontend、sc_prim_fifo_size_backend、
sc_hiz_tile_fifo_size、sc_earlyz_tile_fifo_size配置到所有shader的mmPA_SC_FIFO_SIZE寄存器；

1.3）gfx_v9_0_rlc_resume，rlc初始化：将rlc 的clear state bo memory从vram中pin住；首先stop rlc，关闭
CG（mmRLC_CGCG_CGLS_CTRL）；init_csb将clear state memory配置到rlc；初始化rlc save restore
list；初始化gfx power gating；从新load rlc microcode；关闭电源load blance（可由amdgpu_lbpw参
数控制）；开启rlc；

1.4）gfx_v9_0_cp_resume，cp初始化：首先加载mec microcode；配置kiq（kernel interface queue）；
配置并使能kcq（kernel computer queue，通过kiq下发SET_RESOURCES，循环下发MAP_QUEUES报
文）；最后对配置起来的compute ring(计算队列)，循环运行ring tests判断是否成功初始化;

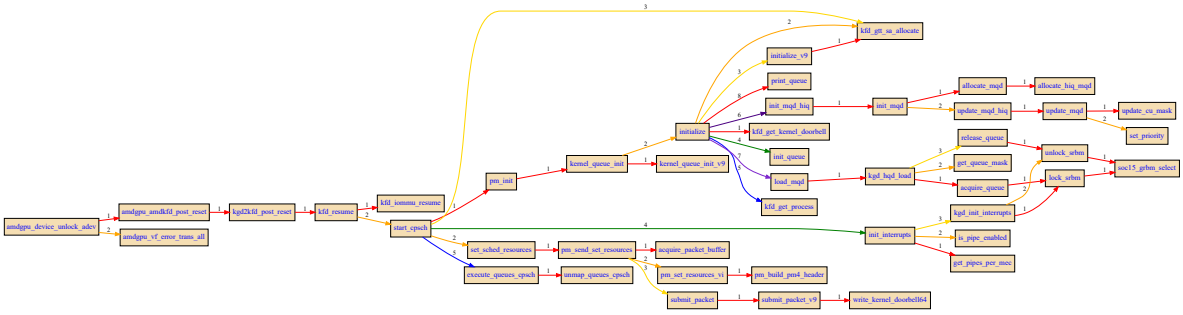
2）SDMA：sdma_v4_0_hw_init

2.1）初始化sdma golden寄存器（sdma0_init、smda0、sdma1）；

2.2）加载sdma micro code；unhat SDMA f32（sdma0，sdma1）；开启sdma ring preemption；配置
sdma0、sdma1的sdma ring和page ring；

2.3）测试sdma0，sdma1的sdma ring和page ring；

6、kfd post reset



1) kfd_iommu_resume，初始化iommu device，设置最大passid到iommu device；向iommu驱动设shutdown
回调函数（for process shutdown），和invalid_ppr_cb回调函数（called by IOMMU driver on PPR
failure）；遍历所有的进程列表，将process的pasid绑定到iommu device（amd_iommu_bind_pasid）

2）start_cpsch：

2.1) pm_init包管理器初始化，安装包管理器操作集kfd_v9_pm_funcs；创建并初始化HIQ；

2.1.1）HIQ初始化：

2.2）设置调度器资源；

2.3）init_interrupts：配置mec1的4个pipe的ring0，开启TIME_STAMP和OPCODE_ERROR中断；

2.4）execute_queues_cpsch：unmap掉DYNAMIC_QUEUES，重新map所有的队列；