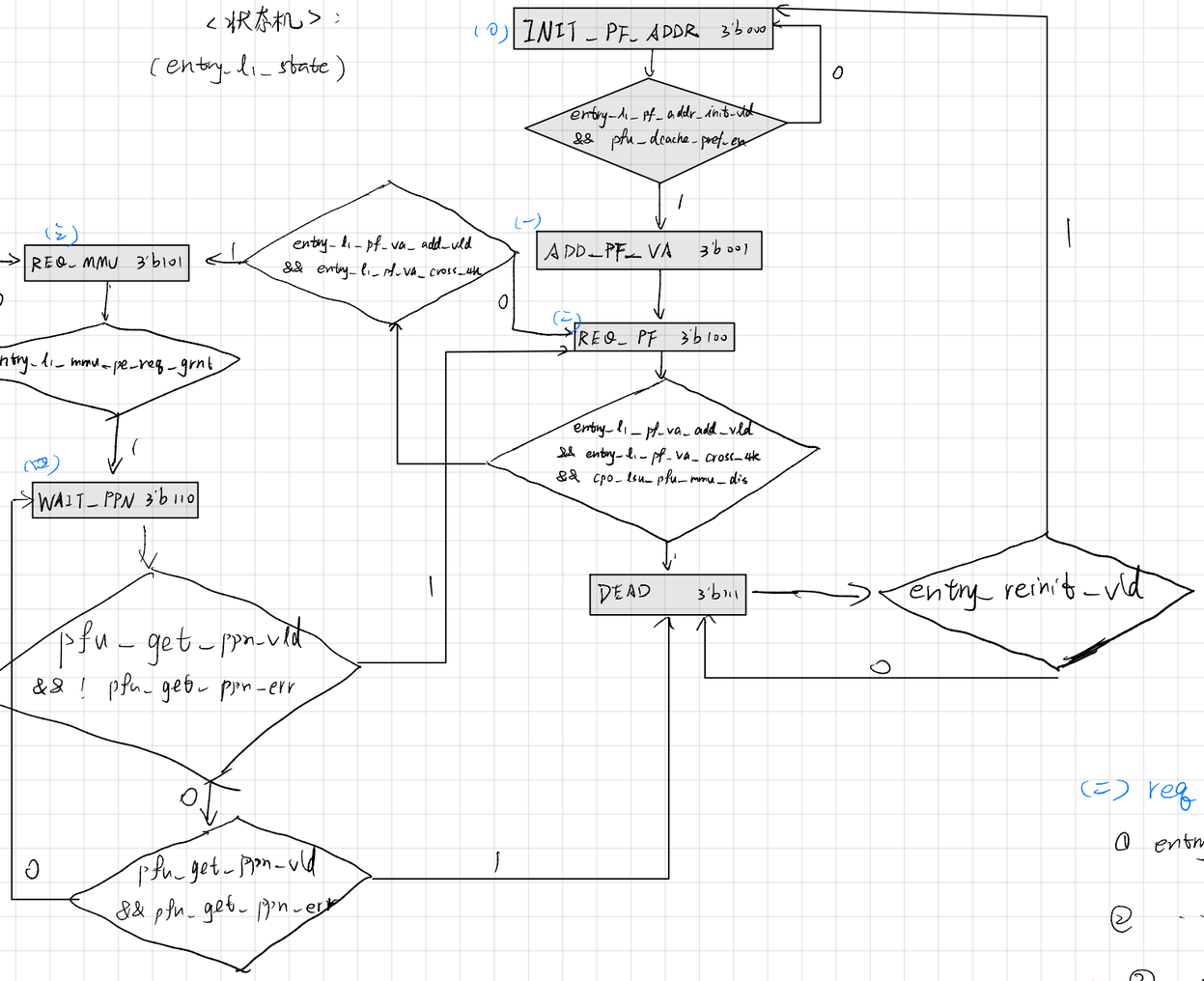
<状态机>：

(entry_l1_state)



State 逻辑：

(0) Init pf addr

entry_tsm_is_judge = 1 代表 初始预取的地址计算完成 (主要由它控制跳转)
(from tsm.v)

使能 pfu_dcache_pref_en = 1 时
(可能是外部)

满足跳转到 (一) 的条件

(一) add l1_pf_va (计算预取地址)

① req_PF 跳转条件之一，biu_grnt为1后分号

② 时钟门电路使能 (干什么用?) is_add_pf_va 为此状态 去l1的grnt顶

③ insts_new_va (from tsm) 我输入
(已赋给 entry_l1_pf_va)
加上步幅高位 → 第一次预取地址

④ 同 tsm中一样算 cross_4k,
作为 (二) 跳转到req的条件之一

后无条件转到 (二)

(二) req pf

① entry_l1_biu_pe_req (SRC那来的?)
↓给
② ---------- req_set → 给tsm 用以判断 biu请求
③ ---------- req_grnt biu 是否准许访问

④ entry_l1_pf_ppn 与 entry_l1_pf_va 连起来
cross_4k = 1
且 mmu_dis = 1 ⇒ 转 DEAD

mmu_dis = 0 ⇒ 转 (三) → 用 mmu 进行虚实页面翻译地址
翻译：从虚拟到物理

(三) mmu

① - ③ 同 (二)

④ 判断 (三) 是否→(四) mmu 即是否准许访问

(四) wait ppn 等待翻译

entry_l1_pf_ppn_up_vld (可能传到 entry 层)

```
//============================================================
//                  Some compare info
//============================================================
// &Force("output","entry_l1_pf_va_sub_inst_new_va"); @272
assign entry_l1_pf_va_sub_inst_new_va[`PA_WIDTH-1:0]  =
                entry_l1_pf_va[`PA_WIDTH-1:0]
                - entry_inst_new_va[`PA_WIDTH-1:0];//是一个组合逻辑，用于计算 L1 缓存中的虚拟地址与新指令的虚拟地址之间的差值。这个差值用于判断是否需要触发重新初始化请求

assign entry_l1sm_diff_sub_dist_strideh[`PA_WIDTH-1:0] =
                entry_l1_pf_va_sub_inst_new_va[`PA_WIDTH-1:0]
                - entry_l1_dist_strideh[`PA_WIDTH-1:0];//用于计算 L1 缓存中的虚拟地址与新指令的虚拟地址之间的差值与距离片上存储器的步长的差值。

assign entry_l1_pf_va_eq_inst_new_va = !(|entry_l1_pf_va_sub_inst_new_va[`PA_WIDTH-1:0]);//用于表示 L1 缓存的页框虚拟地址是否与新指令的虚拟地址相等。

assign entry_inst_new_va_surpass_l1_pf_va_set = (entry_stride_neg ^ entry_l1_pf_va_sub_inst_new_va[`PA_WIDTH-1])
                                              && !entry_l1_pf_va_eq_inst_new_va;//用于指示新指令的虚拟地址是否超过了 L1 缓存中的页框虚拟地址。



assign entry_in_l1_pf_region_set  = entry_stride_neg ^ entry_l1sm_diff_sub_dist_strideh[`PA_WIDTH-1];//用于指示新指令的虚拟地址是否位于 L1 缓存的页框区域内。
```