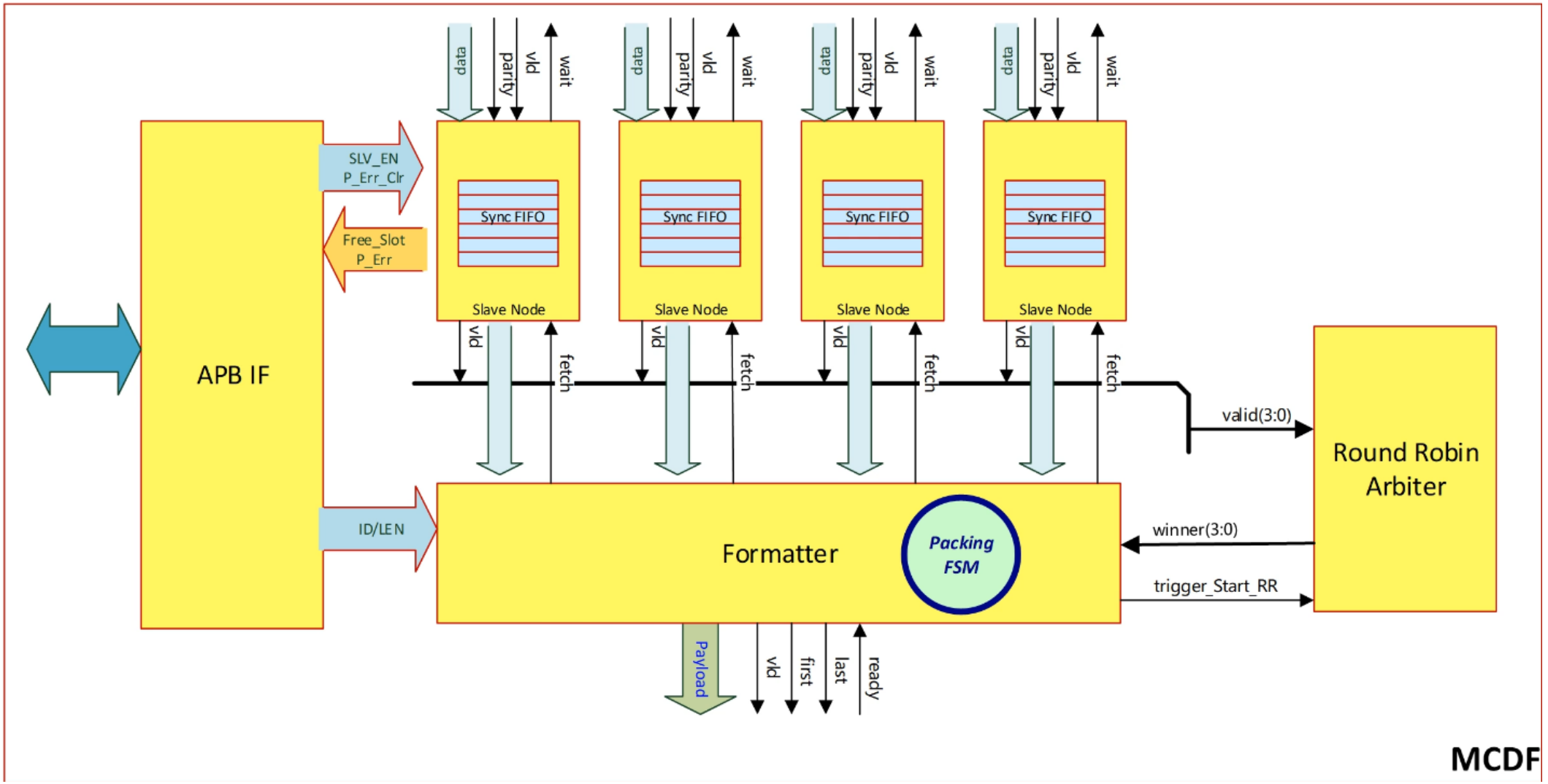
UVM项目实战部分6

路 桑

MCDF的设计更新


data
parity
vld
wait
Sync FIFO
Slave Node
SLV_EN
P_Err_Clr
Free_Slot
P_Err
APB IF
ID/LEN
fetch
valid(3:0)
Round Robin
Arbiter
winner(3:0)
trigger_Start_RR
Formatter
Packing
FSM
Payload
first
last
ready
MCDF

Agenda

1. Channel组件的更新
2. Formatter组件的更新
3. 环境复用的评估
4. 序列复用的评估

Channel组件的更新

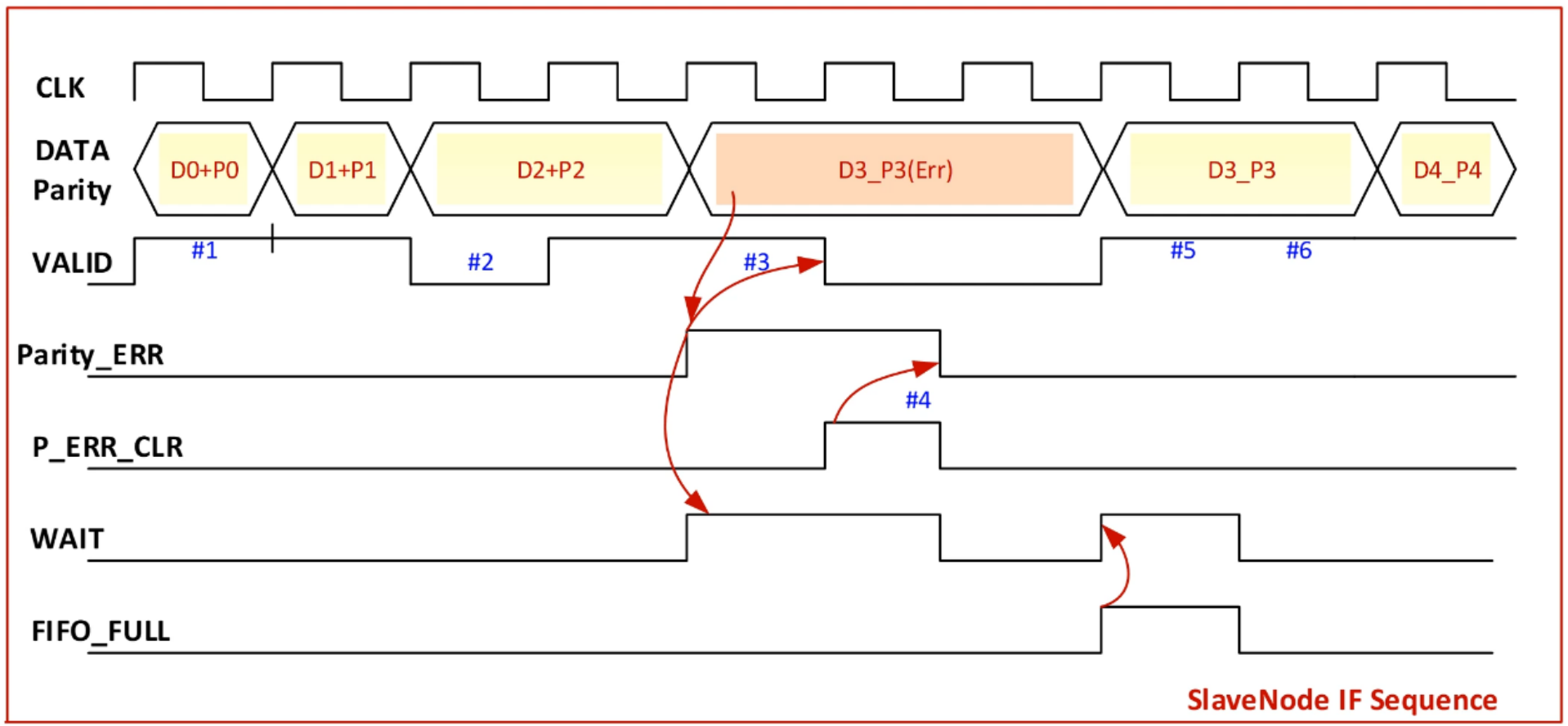
接口更新

Slave node 接口

- DATA(31:0)：通道数据输入。
- DATA_PARITY：Data数据的奇偶校验位。
- VALID：数据有效标志信号。
- WAIT：暂停接收
- PARITY_ERR：slave node 侧发现数据或数据校验位出错

Channel组件的更新

时序更新



Channel组件的更新

时序更新

Slave node接口时序

图例给出了6种传输情形：

#1：正常传输

#2：VALID=0,上行数据未准备好，等待

#3：数据校验错误，接收端置起WAIT将数据接收和输入挂起。

#4：错误标识位通过APB清除，数据准备重发。

#5：FIFO满，WAIT置起，数据发送被延期

#6：FIFO恢复，重发数据成功

Channel组件的更新

interface更新

- 内部信号更新，clocking块更新
- 顶层testbench中，interface的连接更新

Channel组件的更新
sequence item及sequence更新

- 成员变量更新。
- 随机约束更新。
- 在尽量保持原有sequence成员变量的情况下，完成sequence的更新以便在顶层virtual sequence中的复用。

Channel组件的更新
driver及monitor更新

- 更新driver的do_reset()。
- 更新driver的do_drive()。
- 更新driver的chnl_write()。
- 更新driver的chnl_idle()。
- 更新monitor的mon_trans()。

Channel组件的更新
driver及monitor更新

- 随堂练习：
  - 更新chnl_write()方法。

Agenda

1. Channel组件的更新
2. Formatter组件的更新
3. 环境复用的评估
4. 序列复用的评估

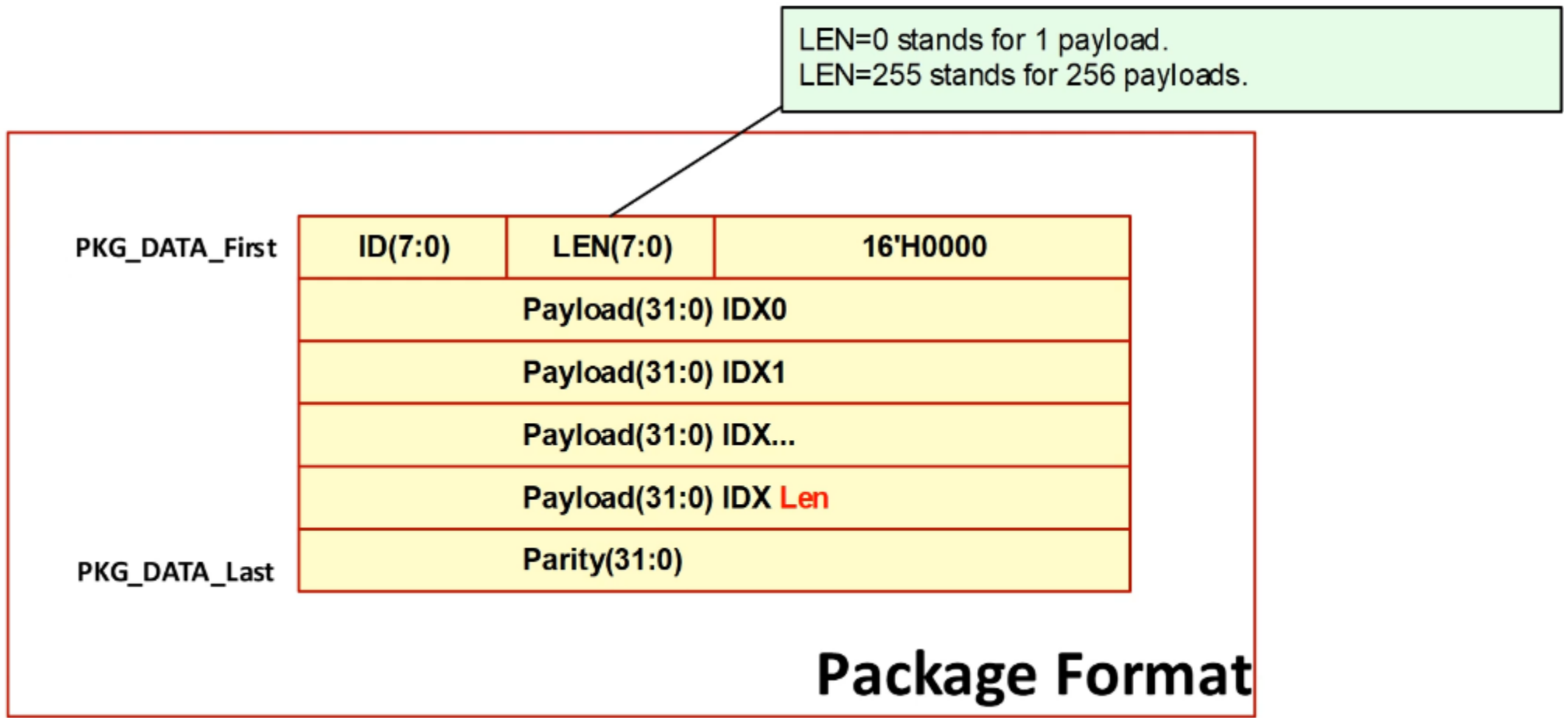
Formatter组件的更新
接口更新

Formatter 接口

- PKG_VLD：数据包有效标识
- PKG_FST：数据包首个数据标识符
- PKG_DATA(31:0)：数据输出端口
- PKG_LST：数据包末尾数据标识符
- REV_RDY: 接受侧准备就绪标识

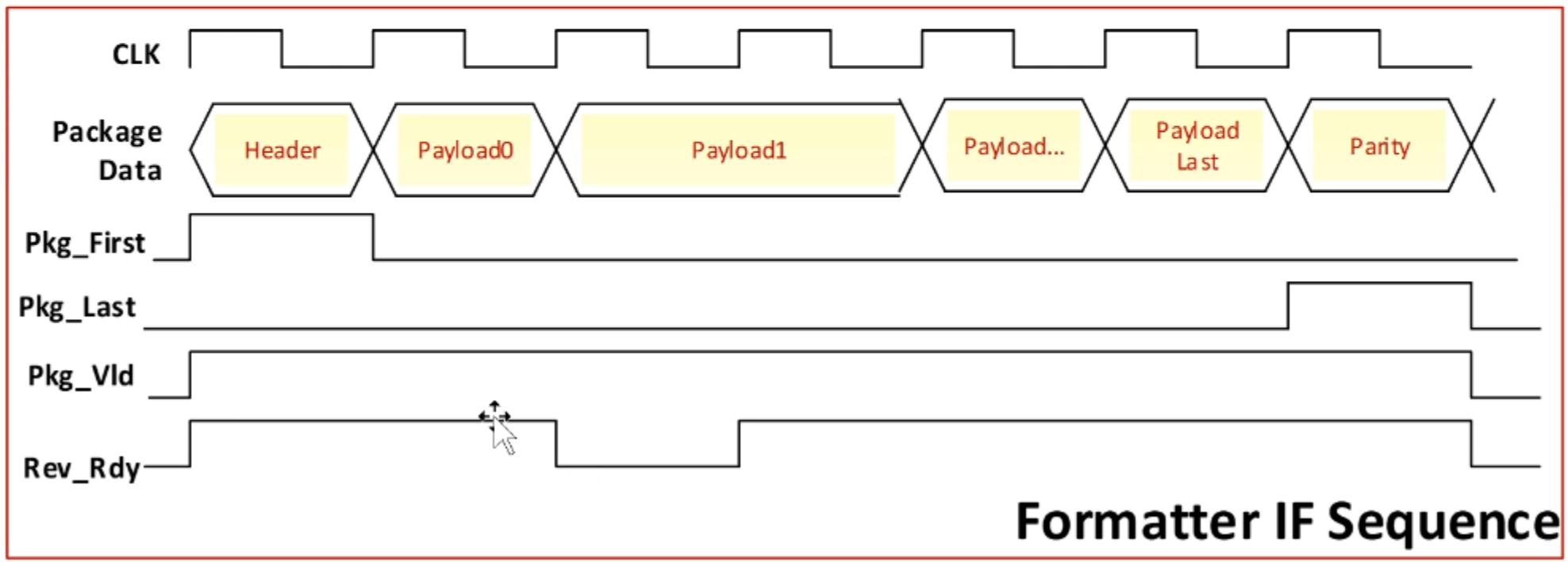
Formatter组件的更新

时序更新



Formatter组件的更新

时序更新



Formatter组件的更新

interface更新

- 内部信号更新，clocking块更新
- 顶层testbench中，interface的连接更新

Formatter组件的更新
sequence item及sequence更新

- 成员变量更新。
- 随机约束更新。
- 在尽量保持原有sequence成员变量的情况下，完成sequence的更新以便在顶层virtual sequence中的复用。

Agenda

1. Channel组件的更新
2. Formatter组件的更新
3. 环境复用的评估
4. 序列复用的评估

环境复用的评估

总体

- 顶层环境包括：
  - chnl_agent
  - reg_agent
  - fmt_agent
  - mcdf_checker
  - mcdf_coverage
  - mcdf_virtual_sequencer
  - mcdf_rgm
  - reg2mcdf_adapter
- 在考虑环境复用时，需要考虑这些因素：
  - 顶层环境中的组件是否需要更新?
  - 顶层环境中的连线是否需要更新?

环境复用的评估
组件更新 mcdf_checker

- 更新refmod的do_reset()。
- 更新refmod的get_field_value()。
- 更新refmod的do_reg_update()。
- 更新refmod的do_packet()。
- 更新checker的do_data_compare()。
- 更新checker的do_channel_disable_check()。
- 更新checker的do_arbiter_priority_check()。
- 更新checker的get_slave_id_with_prio()。

环境复用的评估
组件更新 mcdf_checker

- 更新refmod的do_reset()。
- 更新refmod的get_field_value()。
- 更新refmod的do_reg_update()。
- 更新refmod的do_packet()。
- 更新checker的do_data_compare()。
- 更新checker的do_channel_disable_check()。
- 更新checker的do_arbiter_priority_check()。
- 更新checker的get_slave_id_with_prio()。

环境复用的评估
其余组件更新

- 更新coverage的各个covergroup定义，这是由于设计的内部信号均发生改变,因此有必要针对已有的定义做信号更新，从而更新各个covergroup。
- mcdf_virtual_sequencer可以保持不变，因为它只是底层sequencer的路由器。
- mcdf_rgm在我们之前的寄存器模型自动化中已经练习了如何更新寄存器模型，以及如何使用定制的脚本来完成寄存器模型的更新。
- reg2mcdf_adapter需要修改reg2bus()以及bus2reg()两个方法，这是因为由于之前的reg_agent更换为了标准总线的apb_master_agent，因此需要将原有的bus transaction修改为标准的apb bus transaction。

环境复用的评估
顶层连接更新

- 顶层连线，即各个在mcdf_env中的组件之间的连接，它们是否需要更新取决于这些组件内部用于缓存的单元是否发生变化。
- 由于MCDF的结构未发生变化，因此之前所述的各个组件的更新仅限于内部功能的更新（参考设计的接口和功能更新），而MCDF的结构未发生变化，验证环境中例如缓存较多的部分包括refmod和checker均不需要更新缓存数量。
- 由以上条件可以看出，顶层的连接受益于MCDF的结构稳定，因此不需要做出更新。

环境复用的评估

- 随堂练习：
  - 更新refmod的do_packet()。
  - 更新checker的do_data_compare()。
  - 对照理解adapter的reg2bus()和bus2reg()的方法更新。

环境复用的评估

- 随堂练习：
  - 更新refmod的do_packet()。
  - 更新checker的do_data_compare()。
  - 对照理解adapter的reg2bus()和bus2reg()的方法更新。

Agenda

1. Channel组件的更新
2. Formatter组件的更
3. 环境复用的评估
4. 序列复用的评估

序列复用的评估

- 从环境复用的经验来看，应该尽可能：
  - 保证顶层环境的稳定
  - 尽量减少底层组件的更新
- 而在更新底层组件的时候，也需要考虑如何实施，从而可以保证顶层环境的稳定。
- 同样地，对于序列复用而言，也应该尽可能：
  - 保证底层序列的稳定
  - 尽量减少底层序列的更新
- 为了达成这一目的，我们也需要让底层序列的更新尽可能少地影响到顶层序列。

序列复用的评估
更新底层sequence

- 对于寄存器sequence，由于已经使用寄存器模型进行访问，因此不再需要修改底层的序列，只需要保证APB VIP提供必要的读写序列，完成adapter对应的方法即可。同时注意，由于寄存器模型发生了很大变化，这使得顶层序列针对寄存器模型的改变而需要更新。
- 对于channel sequence，更新对应的序列chnl_data_sequence。
- 对于formatter sequence，更新对应的序列fmt_config_sequence。

序列复用的评估

- 随堂练习：
  - 更新chnl_data_sequence。
  - 更新mcdf_data_consistence_basic_virtual_sequence::do_reg()。