面试项目题

请用Verilog实现文末定义的 `axi_stream_insert_header` 模块并仿真验证。

该模块的输入输出接口已给出，输入是两路AXI Stream信号，输出是一路AXI Stream信号。关于AXI Stream协议请自行查阅协议规范。

输入的两路AXI Stream信号，一路是data相关信号，一路是要添加的header相关信号。此处假设每个data_in可能有多拍，每个header_insert只有一拍。要求实现把header_insert添加到第一拍data_in之前，并且去掉header_insert开头的若干可能无效字节，然后把添加header后的data还是按照AXI Stream协议输出。

输入data相关信号除了用于握手的valid_in和ready_in之外：

- data_in是多位输入数据（位宽要求是2的整数幂次，如8、16、32、64等）；
- last_in用于标识是否为最后一拍输入数据；
- keep_in用于标识每一拍有多少字节有效，特别注意除了last_in为高的最后一拍输入数据，其他拍的所有字节都是有效的，只有last_in为高的最后一拍数据的结尾若干字节可能无效，如32位data信号其对应的最后一拍的keep_in信号可以取值四种情况 `4'b1111`，`4'b1110`，`4'b1100`，`4'b1000`。

输入header相关信号除了用于握手的valid_insert和ready_insert之外：

- header_insert是多位header数据，与data_in位宽相同；
- 没有last信号标识最后一个header，即每拍代表一个单独的header，或者可以理解为header的last信号一直为高；
- keep_insert用于标识header的有效字节，特别注意header开头若干字节可能无效，如32位data_insert信号其对应的keep_insert信号可以取值五种情况 `4'b1111`，`4'b0111`，`4'b0011`，`4'b0001`，`4'b0000`。

输出data相关信号除了用于握手的valid_out和ready_out之外：

- data_out是多位输出数据与data_in位宽相同；
- last_out用于标识是否为最后一拍数据；
- keep_out用于标识data_out的有效字节，特别注意除了last_out为高的最后一拍输出数据，其他拍的所有字节都是有效的，只有last_out为高的最后一拍数据的结尾若干字节可能无效，如32位data信号其对应的最后一拍的keep_in信号可以取值四种情况 `4'b1111`，`4'b1110`，`4'b1100`，`4'b1000`。

比如，AXI Stream输入五拍data，其data_in、last_in和keep_in分别如下：

- 五拍data_in分别是： `32'hABCD`， `32'hEF01`， `32'h2345`， `32'h6789`， `32'h0AXX` (X代表任意值)；
- 五拍last_in分别是： `1'b0`， `1'b0`， `1'b0`， `1'b0`， `1'b1` ；
- 五拍keep_in分别是： `4'b1111`， `4'b1111`， `4'b1111`， `4'b1111`， `4'b1100` ；

AXI Stream输入一拍header，其header_insert和keep_insert分别如下：

- 一拍header_insert： `32'hFEDC` ；
- 一拍keep_insert： `4'b0111` ；

则AXI Stream输出六拍data，其data_out、last_out和keep_out分别如下：

- 六拍data_in分别是： `32'hEDCA`， `32'hBCDE`， `32'hF012`， `32'h3456`， `32'h7890`， `32'hAXXX` ；
- 六拍last_in分别是： `1'b0`， `1'b0`， `1'b0`， `1'b0`， `1'b0`， `1'b1` ；
- 六拍keep_in分别是： `4'b1111`， `4'b1111`， `4'b1111`， `4'b1111`， `4'b1111`， `4'b1000` 。

```
module axi_stream_insert_header #(
    parameter DATA_WD = 32,
    parameter DATA_BYTE_WD = DATA_WD / 8
) (
    input                       clk,
    input                       rst_n,

    // AXI Stream input original data
    input                       valid_in,
    input   [DATA_WD-1 : 0]     data_in,
    input   [DATA_BYTE_WD-1 : 0] keep_in,
    input                       last_in,
    output                      ready_in,

    // AXI Stream output with header inserted
    output                      valid_out,
    output  [DATA_WD-1 : 0]     data_out,
    output  [DATA_BYTE_WD-1 : 0] keep_out,
    output                      last_out,
    input                       ready_out,

    // The header to be inserted to AXI Stream input
    input                       valid_insert,
    input   [DATA_WD-1 : 0]     header_insert,
    input   [DATA_BYTE_WD-1 : 0] keep_insert,
    output                      ready_insert
);
```

```
// Your code here

endmodule
```