module axi\_stream\_insert\_header (

parameter DATA\_WD = 32,

parameter DATA\_BYTE\_WD = DATA\_WD / 8,

parameter BYTE\_CNT\_WD = $clog2(DATA\_BYTE\_WD)

）（

input wire clk,

input wire rst,

input wire [DATA\_WD-1:0] data\_in,

input wire valid\_in,

input wire ready\_in,

input wire [DATA\_BYTE\_WD -1:0] keep\_in,

input wire last\_in,

input wire [DATA\_WD-1:0] data\_insert,

input wire valid\_insert,

input wire ready\_insert,

input wire [DATA\_BYTE\_WD-1:0] keep\_insert,

input wire [BYTE\_CNT\_WD:0] byte\_insert\_cnt,

output wire [DATA\_WD-1:0] data\_out,

output wire valid\_out,

output wire ready\_out,

output wire [DATA\_BYTE\_WD-1:0] keep\_out,

output wire last\_out

);

// 定义状态机的状态

localparam IDLE = 2'b00;

localparam HEADER = 2'b01;

localparam DATA = 2'b10;

// 定义状态机的变量

reg [1:0] state;

reg [DATA\_WD-1:0] data\_reg;

reg [DATA\_BYTE\_WD-1:0] keep\_reg;

reg last\_reg;

reg [BYTE\_CNT\_WD:0] byte\_cnt;

reg [DATA\_WD-1:0] header\_reg;

reg [DATA\_BYTE\_WD-1:0] header\_keep;

reg [BYTE\_CNT\_WD:0] header\_byte\_cnt;

reg [BYTE\_CNT\_WD:0] header\_skip\_cnt;

// 初始化状态机

always @(posedge clk or negedge rst) begin

if (!rst) begin

state <= IDLE;

data\_reg <= 0;

keep\_reg <= 0;

last\_reg <= 0;

byte\_cnt <= 0;

header\_reg <= 0;

header\_keep <= 0;

header\_byte\_cnt <= 0;

header\_skip\_cnt <= 0;

end else begin

case (state)

IDLE: begin

if (valid\_in && ready\_insert) begin

state <= HEADER;

header\_reg <= data\_insert;

header\_keep <= keep\_insert;

header\_byte\_cnt <= byte\_insert\_cnt;

header\_skip\_cnt <= 0;

end else if (valid\_in && ready\_in) begin

state <= DATA;

data\_reg <= data\_in;

keep\_reg <= keep\_in;

last\_reg <= last\_in;

byte\_cnt <= 0;

end else begin

state <= IDLE;

end

end

HEADER: begin

if (header\_skip\_cnt < $countones(header\_keep)) begin

header\_skip\_cnt <= header\_skip\_cnt + 1;

end else if (valid\_in && ready\_in) begin

state <= DATA;

data\_reg <= data\_in;

keep\_reg <= keep\_in;

last\_reg <= last\_in;

byte\_cnt <= 0;

end else begin

state <= HEADER;

end

end

DATA: begin

if (byte\_cnt < $countones(keep\_reg)) begin

data\_out <= data\_reg;

keep\_out <= keep\_reg;

valid\_out <= 1;

byte\_cnt <= byte\_cnt + 1;

if (byte\_cnt == $countones(keep\_reg)-1 && last\_reg) begin

last\_out <= 1;

end else begin

last\_out <= 0;

end

if (valid\_in && ready\_in) begin

data\_reg <= data\_in;

keep\_reg <= keep\_in;

last\_reg <= last\_in;

byte\_cnt <= 0;

end else begin

data\_reg <= {data\_reg[DATA\_WD-1], data\_reg[DATA\_WD-1:8]};

keep\_reg <= {keep\_reg[DATA\_BYTE\_WD-1], keep\_reg[DATA\_BYTE\_WD-1:1]};

last\_reg <= 0;

end

end else if (header\_byte\_cnt > 0) begin

data\_out <= header\_reg;

keep\_out <= header\_keep;

valid\_out <= 1;

byte\_cnt <= byte\_cnt + 1;

header\_byte\_cnt <= header\_byte\_cnt - 1;

if (header\_byte\_cnt == 1) begin

last\_out <= 1;

end else begin

last\_out <= 0;

end

header\_reg <= {header\_reg[DATA\_WD-1], header\_reg[DATA\_WD-1:8]};

header\_keep <= {header\_keep[DATA\_BYTE\_WD-1], header\_keep[DATA\_BYTE\_WD-1:1]};

end else if (valid\_in && ready\_insert) begin

state <= HEADER;

header\_reg <= data\_insert;

header\_keep <= keep\_insert;

header\_byte\_cnt <= byte\_insert\_cnt;

header\_skip\_cnt <= 0;

end else begin

state <= DATA;

end

end

endcase

end

end

// 定义握手信号

assign ready\_out = state == DATA ? ready\_in : 1;

assign valid\_out = state == DATA ? valid\_in : 0;

endmodule