哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2025 春季

课程名称：计算机组成原理（实验）

实验名称：AXI4 总线接口设计

实验性质：设计型

实验学时：4 地点：T2506

学生班级：7 班

学生学号：2023311709

学生姓名：宁中昊

作业成绩：

实验与创新实践教育中心制

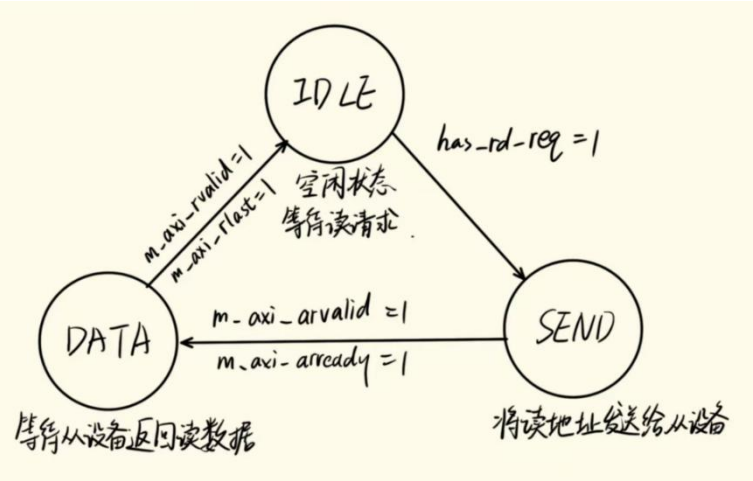
2025 年 4 月

1、设计与实现

要求：结合关键代码详细描述 axi_master 的设计实现，也可结合状态图、流程图、电路框图或时序图等工具辅助描述。

实现了 axi_master 的读地址（AR）和读数据（R）两个通道，写通道在模板中已经提供。

状态转移逻辑与状态图：



在 IDLE 状态中，优先处理 DCache 的读请求，将读地址赋值给 m_axi_raddr，使用 is_dc_req 辅助判断是 ICache 还是 DCache 发起的读请求。

在 SEND 状态中，如果读地址有效且从设备已经准备好，即可将 m_axi_arvalid 信号拉低，进入 DATA 状态。

```
/* 给AR通道的输出信号赋值 */
always @(posedge aclk or posedge areset) begin
    if(areset) begin
        m_axi_araddr  <= 32'h0;
        m_axi_arlen   <=  8'd7; // 连续传输 8 个数据包
        m_axi_arsize  <=  3'd2; // 数据包大小为 4 字节
        m_axi_arburst <=  2'd1; // INCR 模式
        m_axi_arvalid <=  1'b0;
        is_dc_req     <=  1'b0;
    end else begin
        case(cur_state)
        IDLE: begin
            if(has_dc_rd_req) begin
                m_axi_arvalid <= 1'b1;
                m_axi_araddr  <= dc_cpu_raddr;
                is_dc_req     <= 1'b1;
            end else if(has_ic_rd_req) begin
                m_axi_arvalid <= 1'b1;
                m_axi_araddr  <= ic_cpu_raddr;
                is_dc_req     <= 1'b0;
            end else begin
                m_axi_araddr  <= 0;
            end
        end
        SEND: begin
            m_axi_arvalid <= (m_axi_arvalid & m_axi_arready) ? 1'b0 : 1'b1;
        end
        DATA: begin
        end
        default: begin
            m_axi_arvalid <= 1'b0;
            m_axi_araddr  <=    0;
        end
        endcase
    end
end
```

读数据通道准备：

在 SEND 状态中，判断是 ICache 还是 DCache 发起的读请求，相应地拉低 ic_dev_rrdy 或 dc_dev_rrdy 信号，为 DATA 状态读数据做准备，直到 DATA 状态结束再拉高 ic_dev_rrdy 与 dc_dev_rrdy 信号。

```
/* 1: 给ic_dev_rrdy、dc_dev_rrdy信号赋值（接收到读请求后ready置0，读请求处理完成后置1）*/
always @(posedge aclk or posedge areset) begin
    if(areset) begin
        dc_dev_rrdy <= 1'b1;
        ic_dev_rrdy <= 1'b1;
    end else begin
        case(cur_state)
        IDLE: begin
            dc_dev_rrdy <= 1'b1;
            ic_dev_rrdy <= 1'b1;
        end
        SEND: begin
            if(is_dc_req) begin
                dc_dev_rrdy <= 1'b0;
            end else begin
                ic_dev_rrdy <= 1'b0;
            end
        end
        DATA : begin
            if(m_axi_rvalid & m_axi_rlast) begin
                dc_dev_rrdy <= 1'b1;
                ic_dev_rrdy <= 1'b1;
            end
        end
        default: begin
            ic_dev_rrdy <= 1'b1;
            dc_dev_rrdy <= 1'b1;
        end
        endcase
    end
end
```

在接收到最后一个数据包后，根据是 ICache 还是 DCache 发起的读请求，相应地拉高 ic_dev_rvalid 或 dc_dev_rvalid 信号，表示接收到的信号有效，同时将数据赋值给对应的 ic_dev_rdata 或 dc_dev_rdata 信号。

```
/* 2: 给ic_dev_rvalid、dc_dev_rvalid信号赋值（返回给Cache的数据块就绪后置1）*/
always @(posedge aclk or posedge areset) begin
    if(areset) begin
        dc_dev_rvalid <= 1'b0;
        ic_dev_rvalid <= 1'b0;
    end else begin
        if(m_axi_rvalid & m_axi_rlast) begin
            dc_dev_rvalid <=  is_dc_req;
            ic_dev_rvalid <= ~is_dc_req;
        end else begin
            dc_dev_rvalid <= 1'b0;
            ic_dev_rvalid <= 1'b0;
        end
    end
end
```

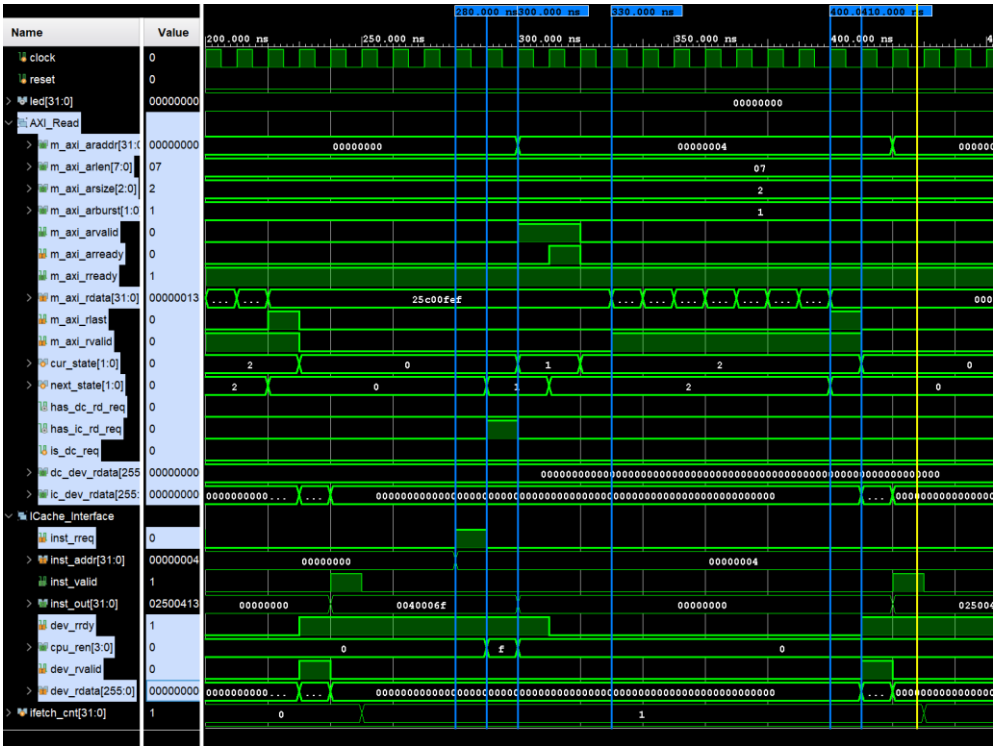
```
/* 3: 从总线上接收读数据，生成DCache重填所需的数据块dc_dev_rdata及ICache重填所需的数据块ic_dev_rdata */
always @(posedge aclk or areset) begin
    if(areset) begin
        rdata_temp <= 0;
    end else begin
        if(m_axi_rvalid) begin
            rdata_temp <= {m_axi_rdata, rdata_temp[255:32]};
        end
    end
end

always @(posedge aclk or posedge areset) begin
    if(areset) begin
        rdata_temp <= 0;
    end else begin
        if(m_axi_rvalid & m_axi_rlast) begin
            if(is_dc_req) begin
                dc_dev_rrdy <= 1'b1;
            end else begin
                ic_dev_rrdy <= 1'b1;
            end
        end else begin
            dc_dev_rdata <= 0;
            ic_dev_rdata <= 0;
        end
    end
end

always @(posedge aclk or posedge areset) begin
    if(areset) begin
        dc_dev_rdata <= 0;
        ic_dev_rdata <= 0;
    end else begin
        if(m_axi_rvalid & m_axi_rlast) begin
            if(is_dc_req) begin
                dc_dev_rdata <= {m_axi_rdata, rdata_temp[255:32]};
            end else begin
                ic_dev_rdata <= {m_axi_rdata, rdata_temp[255:32]};
            end
        end
    end
end
```

2、调试报告

要求：仿真截图要求包含 axi_master 处理 ICache 和 DCache 读请求的两种情况，每种情况列举 1 个测试用例进行详细分析。仿真分析的**思路可参考**指导书实验步骤的仿真波形分析示例，但需**把 axi_master 内部的关键信号加入波形并进行相应分析**。



【280ns】CPU 发出取指请求，故 inst_rreq 信号为高电平。

【290ns】Cache 模块发出读访存请求，故 cpu_ren 信号为 4'hF，访存地址为 32'h0。

【300ns】axi_master 总线模块通过 AR 通道向下游的从设备发出读地址请求，arvalid 信号有效表示当前读地址请求有效，arlen 信号为 8'h7 表示发起长度为 8 个数据包的猝发传输，arsize 信号为 3'h2 表示数据包大小为 32 位，arburst 信号为 2'h1 表示地址递增模式。在下一个时钟周期，aready 信号有效，表示读请求已被从设备接收。

【330ns】rvalid 信号有效，表示从设备开始返回数据给 axi_master 模块，往后每个时钟返回一个数据。

【400ns】rlast 信号有效，axi_master 接收最后一个数据。

【410ns】axi_master 拉高 dev_rvalid 信号，同时把接收完整的 256 位的数据块返回给 Cache 模块。

3、实验总结

要求：总结实验过程中遇到的有价值的问题及解决方法、收获等。此外，对本课程的实验提出合理的意见和建议（不限于实验 4）。

在实验四中，一开始认为实现逻辑较明晰，因此没有使用状态机，而是直接使用 if-else 条件控制语句，导致需要不断重复实现条件判断状态转移，导致代码冗余、实现逻辑反而不清晰。因此使用状态机对原本的实现逻辑进行了重构。

在有涉及到状态转移的逻辑时就应直接使用状态机与三段式以简化实现逻辑，这样可以使得条件判断在三段式的第二段状态转移逻辑中提前实现，而在第三段中可以更加专心于输出逻辑的实现，避免思路混乱。

另外关于课程建议，希望实验安排可以放在更早开始。实际上实验中所需的理论内容并不多要求也不高，相对于理论的复习更可以作为理论课的预习，这样可以与理论课将要结课时大二学生众多的课程论文与截止时限错开，大家也可以分出更多的时间和精力在有趣的实验上而非先应付过检查再说。