西安电子科技大学
XIDIAN UNIVERSITY

# 基于 FPGA 的数字系统设计

## 实验报告

实验名称：第 6 次实验报告

任课教师：沈沛意老师

学号姓名：

提交日期：

# 一、 实验介绍

本实验将完成图 6.19 中算术逻辑单元 ALU 的 RTL 描述。

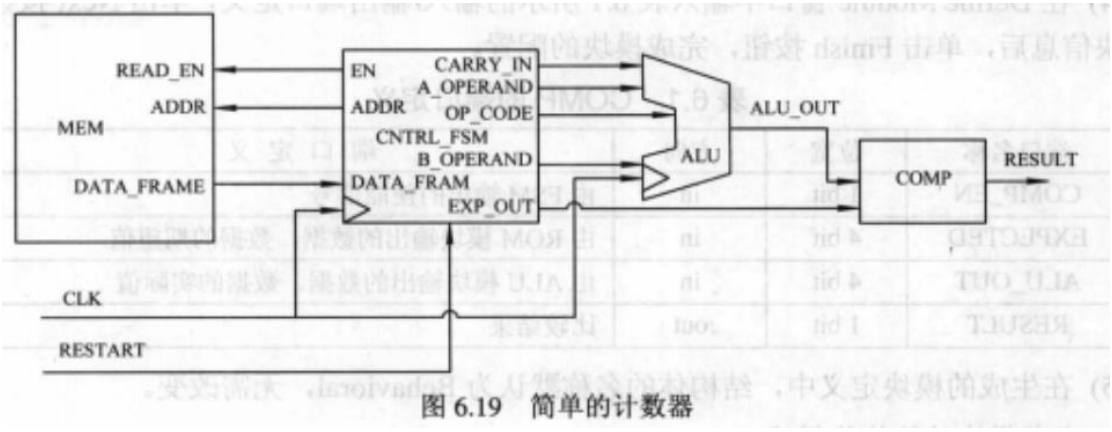



图 6.19 简单的计数器

# 二、 实验目标

学习使用 case 语句描述 ALU 的结构;

验证逻辑结构;

为计数器项目创建一个子模块;

为输出数据的比较做准备。

# 三、 实验过程与步骤

本实验包含三个主要的部分: 创建一个新的 ISE 工程; 创建 COMP 的行为描述代码; 使用 case 描述语句;估计输出的期望值以用于后面输出的比较。

## 1. 启动 ISE 创建一个新的工程 LAB6

过程不再赘述

## 2. 使用 case 描述语句

创建 ALU 模块的实体:

(1)选择菜单栏中的 Project→New Source。

(2)在 Select Source Type 窗口中,选择左侧的 VHDL Module,在右侧 File Name 栏中填入文件名 ALU。

(3)单击 Next 按钮,进入 Define Module 窗口。

(4)在 Define Module 窗口中输入表 6.2 所示的输入/输出端口定义,单击 Next 按钮,确认模块信息后,单击 Finish 按钮完成模块的配置

表 6.2 ALU 的端口定义

| 端口名称 | 位宽 | 方向 |
|---|---|---|
| A | 4 bit | in |
| B | 4 bit | in |
| C_IN | 1 bit | in |
| OP_CODE | 4 bit | in |
| CLK | 1 bit | in |
| EN | 1 bit | in |
| Y | 4 bit | out |

完成模块的结构体描述:

(1) 用 case 语句描述如表 6.3 所示的 ALU 的功能语句

表 6.3 ALU 的功能描述

| OP_CODE | | | | CARRY_IN C_IN | 操作 | 功能 |
|---|---|---|---|---|---|---|
| S3 | S2 | S1 | S0 | | | |
| 0 | 0 | 0 | 0 | 0 | Y<=A | 传输 A |
| 0 | 0 | 0 | 0 | 1 | Y<=A+1 | 递增 |
| 0 | 0 | 0 | 1 | 0 | Y<=A+B | 加和 |
| 0 | 0 | 0 | 1 | 1 | Y<=A+B+1 | 带进位的加和 |
| 0 | 0 | 1 | 0 | 0 | Y<=A+(not B) | 与补码相加 |
| 0 | 0 | 1 | 0 | 1 | Y<=A+(not B)+1 | 减 |
| 0 | 0 | 1 | 1 | 0 | Y<=A−1 | 递减 |
| 0 | 0 | 1 | 1 | 1 | Y<=A | 传输 A |
| 0 | 1 | 0 | 0 | 0 | Y<=A and B | 与 |
| 0 | 1 | 0 | 1 | 0 | Y<=A or B | 或 |
| 0 | 1 | 1 | 0 | 0 | Y<=A xor B | 异或 |
| 0 | 1 | 1 | 1 | 0 | Y<=not A | 反码 |
| 1 | 0 | 0 | 0 | 0 | Y<=0 | 传输 0 |

(2) 计算可能的输出结果。计算在给定的一些输入时,输出应该是什么。以下表格中的数据将被用于描述语句的功能验证。

| A_IN | B_IN | OP_CODE | C_IN | EXP_OUT |
|---|---|---|---|---|
| 0001 | 0001 | 0000 | 0 | 0001(A) |
| 0001 | 0001 | 0000 | 1 | 0010(A+1) |
| 0001 | 0001 | 0001 | 0 | 0010(A+B) |

| | | | | |
|---|---|---|---|---|
| 0001 | 0001 | 0001 | 1 | 0011(A+B+1) |
| 0010 | 0001 | 0010 | 0 | 0000 (A+ not B) |
| 0010 | 0001 | 0010 | 1 | 0001 （A-B） |
| 0010 | 0010 | 0011 | 0 | 0001(A-1) |
| 0010 | 0001 | 0011 | 1 | 0010(A) |
| 0011 | 0001 | 0100 | 0 | 0001(与) |
| 0011 | 0001 | 0101 | 0 | 0011（或） |
| 0101 | 0110 | 0110 | 0 | 0011（异或） |
| 0101 | 0110 | 0111 | 0 | 1010（反码） |
| 0101 | 0110 | 1000 | 0 | 0000（传输 0） |
| 0101 | 0110 | 0111 | 1 | X（不合法输出） |

语法检查:

在 Sources 窗口选中 ALU.VHD,在 Processes 窗口中展开 Synthesis,并双击 Check Syntax .

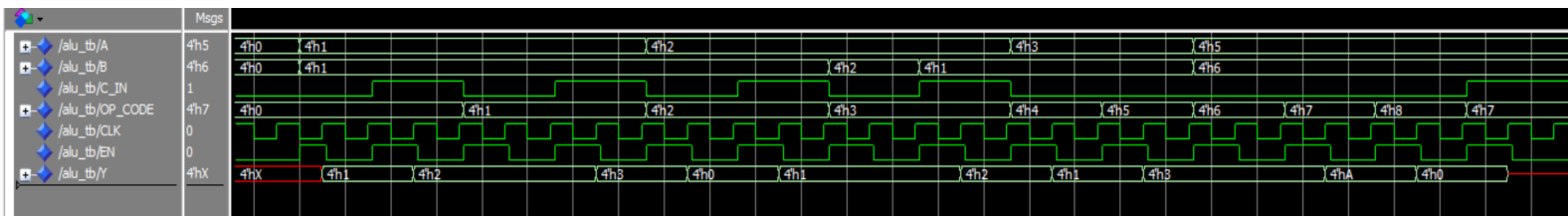
## 四、 实验总结

在本实验中,我们用 case 语句创建了计数器工程中的 ALU 模块,并为验证后续实验中的设计模块做好了准备。

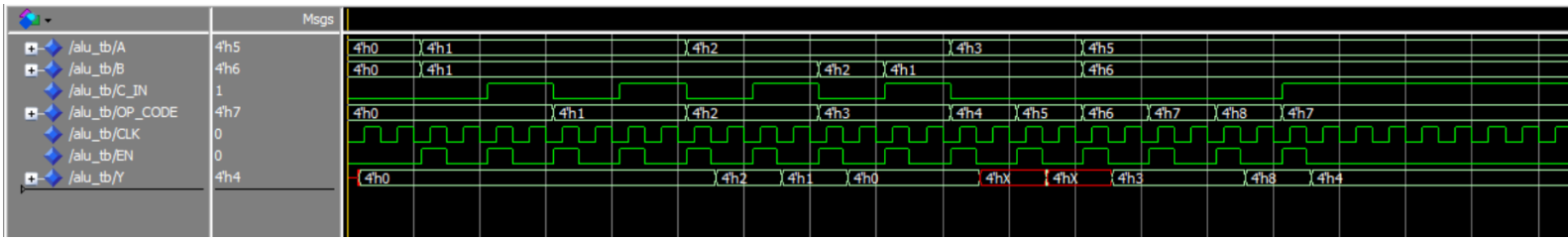
在后续的模块中,将从 MEM 模块中读出用于计算的数据和期望的输出结果,将用于计算的输入数据传入 ALU 中进行计算,计算的结果传输到 CMP 的输入接口。在 CMP 中,将 ALU 计算的结果和 MEM 模块中读出的期望结果进行比较,最后给出比较结果。

## 五、 实验结果及分析

前仿波形:



后仿波形：




## 六、 实验代码

ALU.vhd:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- 定义 ALU 实体
entity ALU is
    Port (
        A       : in  STD_LOGIC_VECTOR (3 downto 0); -- 输入 A，4 位
        B       : in  STD_LOGIC_VECTOR (3 downto 0); -- 输入 B，4 位
        C_IN    : in  STD_LOGIC;                     -- 进位输入
        OP_CODE: in  STD_LOGIC_VECTOR (3 downto 0); -- 操作码，4 位
        CLK     : in  STD_LOGIC;                     -- 时钟信号
        EN      : in  STD_LOGIC;                     -- 使能信号（高电平有效）
        Y       : out STD_LOGIC_VECTOR (3 downto 0)  -- 输出结果，4 位
    );
end ALU;

architecture RTL of ALU is
  -- 定义内部信号，将操作码和进位输入合并为 5 位信号
```

```
    -- 高 4 位为 OP_CODE，最低位为 C_IN，用于扩展操作选择
    signal OP_CODE_CI: STD_LOGIC_VECTOR(4 downto 0);
begin
    -- 合并操作码和进位输入，生成 5 位控制信号
    OP_CODE_CI <= OP_CODE & C_IN;

    -- 同步过程（仅在时钟上升沿触发）
    PROCESS(CLK)
    begin
        if rising_edge(CLK) then -- 检测时钟上升沿
            if EN = '1' then            -- 使能信号有效时执行操作
                case OP_CODE_CI is      -- 根据合并后的操作码选择操作
                    -- 以下为不同操作码对应的功能：
                    when "00000" => Y <= A;                  -- 直接输出 A（无操作）
                    when "00001" => Y <= A + 1;              -- A 自增 1（INC 操作）
                    when "00010" => Y <= A + B;              -- A 加 B（ADD）
                    when "00011" => Y <= A + B + 1;          -- A 加 B 加进位（ADD with Carry）
                    when "00100" => Y <= A + not B;          -- A 加 B 的反码（相当于 A - B - 1）
                    when "00101" => Y <= A + not B + 1; -- A 加 B 的反码加 1（相当于 A - B）
                    when "00110" => Y <= A - 1;              -- A 自减 1（DEC 操作）
                    when "00111" => Y <= A;                  -- 保留，与"00000"重复
                    when "01000" => Y <= A and B;            -- 按位与（AND）
```

```
            when "01010" => Y <= A or B;              -- 按位或（OR）
            when "01100" => Y <= A xor B;             -- 按位异或（XOR）
            when "01110" => Y <= not A;               -- A 取反（NOT 操作）
            when "10000" => Y <= (others=>'0');--  输出清零
            when others    => Y <= (others=>'X');--
        end case;
    end if;
  end if;
end process;
end architecture RTL;
```

# 七、 资源分析报告与最高工作频率分析

(一) 资源分析报告：

Device utilization summary:

| Device Utilization Summary | | | | [–] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of 4 input LUTs | 56 | 3,840 | 1% | |
| Number of occupied Slices | 28 | 1,920 | 1% | |
| Number of Slices containing only related logic | 28 | 28 | 100% | |
| Number of Slices containing unrelated logic | 0 | 28 | 0% | |
| Total Number of 4 input LUTs | 56 | 3,840 | 1% | |
| Number of bonded IOBs | 19 | 141 | 13% | |
| IOB Flip Flops | 4 | | | |
| Number of BUFGMUXs | 1 | 8 | 12% | |
| Average Fanout of Non-Clock Nets | 3.62 | | | |

关键指标解读

低 LUT 与 Slice 利用率

当前 ALU 设计仅占用 1%的 LUT 和 Slice 资源，表明其算术逻辑功能较为简单，未涉及复杂运算（如乘法、移位）或状态机控制。

高 IOB 利用率（13%）

I/O 端口占用 19 个，主要来自以下接口：

输入：A[3:0], B[3:0], OP_CODE[3:0], CLK, EN, C_IN（总计 11 输入）

输出：Y[3:0]（4 输出）

(二) 时序报告与相应最高工作频率计算:

Release 14.7 Trace (nt64)
Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.

D:\Xilinx\14.7\ISE_DS\ISE\bin\nt64\unwrapped\trce.exe -intstyle ise -v 3 -s 5
-n 3 -fastpaths -xml ALU.twx ALU.ncd -o ALU.twr ALU.pcf

Design file: ALU.ncd
Physical constraint file: ALU.pcf
Device,package,speed: xc3s200,pq208,-5 (PRODUCTION 1.39 2013-10-13)
Report level: verbose report

| Environment Variable | Effect |
| --- | --- |
| NONE | No environment variables were set |

INFO:Timing:2698 - No timing constraints found, doing default enumeration.
INFO:Timing:3412 - To improve timing, see the Timing Closure User Guide (UG612).
INFO:Timing:2752 - To get complete path coverage, use the unconstrained paths
option. All paths that are not constrained will be reported in the
unconstrained paths section(s) of the report.
INFO:Timing:3339 - The clock-to-out numbers in this timing report are based on
a 50 Ohm transmission line loading model. For the details of this model,
and for more information on accounting for different loading conditions,
please see the device datasheet.
INFO:Timing:3390 - This architecture does not support a default System Jitter
value, please add SYSTEM_JITTER constraint to the UCF to modify the Clock
Uncertainty calculation.
INFO:Timing:3389 - This architecture does not support 'Discrete Jitter' and
'Phase Error' calculations, these terms will be zero in the Clock
Uncertainty calculation. Please make appropriate modification to
SYSTEM_JITTER to account for the unsupported Discrete Jitter and Phase

Error.

Data Sheet report:

---

All values displayed in nanoseconds (ns)

Setup/Hold to clock CLK

| Source | Max Setup to clk (edge) | Max Hold to clk (edge) | Internal Clock(s) | Clock Phase |
|---|---|---|---|---|
| A<0> | 6.720(R) | -0.944(R) | CLK_BUFGP | 0.000 |
| A<1> | 6.169(R) | -0.982(R) | CLK_BUFGP | 0.000 |
| A<2> | 6.358(R) | -0.625(R) | CLK_BUFGP | 0.000 |
| A<3> | 4.561(R) | -0.678(R) | CLK_BUFGP | 0.000 |
| B<0> | 7.489(R) | -0.875(R) | CLK_BUFGP | 0.000 |
| B<1> | 6.957(R) | -1.062(R) | CLK_BUFGP | 0.000 |
| B<2> | 5.861(R) | -1.519(R) | CLK_BUFGP | 0.000 |
| B<3> | 5.762(R) | -1.957(R) | CLK_BUFGP | 0.000 |
| C_IN | 6.901(R) | -0.254(R) | CLK_BUFGP | 0.000 |
| EN | 1.013(R) | 0.748(R) | CLK_BUFGP | 0.000 |
| OP_CODE<0> | 5.631(R) | -1.358(R) | CLK_BUFGP | 0.000 |
| OP_CODE<1> | 5.855(R) | -0.688(R) | CLK_BUFGP | 0.000 |
| OP_CODE<2> | 4.025(R) | -0.399(R) | CLK_BUFGP | 0.000 |
| OP_CODE<3> | 3.395(R) | 0.090(R) | CLK_BUFGP | 0.000 |

Clock CLK to Pad

| Destination | clk (edge) to PAD | Internal Clock(s) | Clock Phase |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Y<0> | 6.404(R) | CLK_BUFGP | 0.000 |
| Y<1> | 6.404(R) | CLK_BUFGP | 0.000 |
| Y<2> | 6.404(R) | CLK_BUFGP | 0.000 |
| Y<3> | 6.404(R) | CLK_BUFGP | 0.000 |

Analysis completed Tue May 27 16:50:22 2025

---

Trace Settings:

---

Trace Settings

Peak Memory Usage: 4505 MB

根据时序报告中的关键路径数据：

最大时钟到输出延迟（Clock-to-Pad）： **6.404ns**（Y<0> 到 PAD 的上升沿延迟）

建立时间（Setup Time）： **6.720ns**（A<0> 到 CLK 的建立时间）

最高工作频率计算（理论极限频率）：

$$F_{\max} = \frac{1}{T_{\text{cycle}}}, \quad \text{其中 } T_{\text{cycle}} \geq T_{\text{co\_max}} + T_{\text{setup}}$$

代入数值

Tcycle=6.404ns+6.720ns=13.124ns

Fmax≈76.2MHz

该 ALU 的最高工作频率约为 **76.2** MHz