# 实验 1　熟悉 Vivado 环境

Chen-Yuanmeng *

2024/10/17

## 1　实验目的

1. 熟悉 Vivado 设计流程
2. 掌握利用 Vivado 创建设计的方法（以实现 4 位加法器为例）
3. 熟练 Verilog 语法
4. 掌握编写 Testbench 的方法，以及行为仿真方法

## 2　实验环境

- Microsoft Windows 10.0.19043.928
- Vivado v2017.4 (64-bit)
- 玉泉路一机房

## 3　原理说明

### 3.1　4 位加法器

直接使用计算功能进行计算即可, 代码如下:

```
assign {c_out, out} = add_0 + add_1 + c_in;
```

### 3.2　全加器 (门电路版本)

根据全加器的定义, 知道 $S = A \oplus B \oplus C_i$, $C_o = AB + AC_i + BC_i$. 据此写出 Verilog 代码即可:

```
assign s = (a ^ b) ^ ci;
assign co = (a & b) | (a & ci) | (b & ci);
```

### 3.3　3–8 译码器 (高电平有效)

在控制序列的三位分别为 `3'b100` 的时候, 将输入的 3 位二进制数转化为相应的一个电平信号输出, 可以用 `case` 语句. 控制语句可用 `if` 判断. 具体如下:

```
always @(*) begin
    case(a)
        3'b000: out = 8'b1;
```

*Email: chenyumeng23@mails.ucas.ac.cn

```
        3'b001: out = 8'b10;
        3'b010: out = 8'b100;
        3'b011: out = 8'b1000;
        3'b100: out = 8'b10000;
        3'b101: out = 8'b100000;
        3'b110: out = 8'b1000000;
        3'b111: out = 8'b10000000;
    endcase

    if (cs != 3'b110) begin
        out= 8'b0;
    end
end
```

# 4 接口定义

## 4.1 4 位加法器

```
input [3:0] add_0,   \\ 加数 1
input [3:0] add_1,   \\ 加数 2
input c_in,          \\ 进位输入
output [3:0] out,    \\ 输出
output c_out         \\ 进位输出
```

## 4.2 全加器 (门电路版本)

```
input a,  \\ 加数 1
input b,  \\ 加数 2
input ci, \\ 进位输入
output s, \\ 本位输出
output co \\ 进位输出
```

## 4.3 3–8 译码器 (高电平有效)

```
input [2:0] a,   \\ 输入的 3 位二进制数码
input [2:0] cs,  \\ 控制序列 (当且仅当为 110 时, 有效输出)
output [7:0] y   \\ 输出的 8 位电平
```

# 5 调试过程及结果

点击左侧 “SIMULATION” 下的 “Run Simulation”, 以默认设置进行模拟调试. 调试结果分别如图所示:

## 5.1 4 位加法器

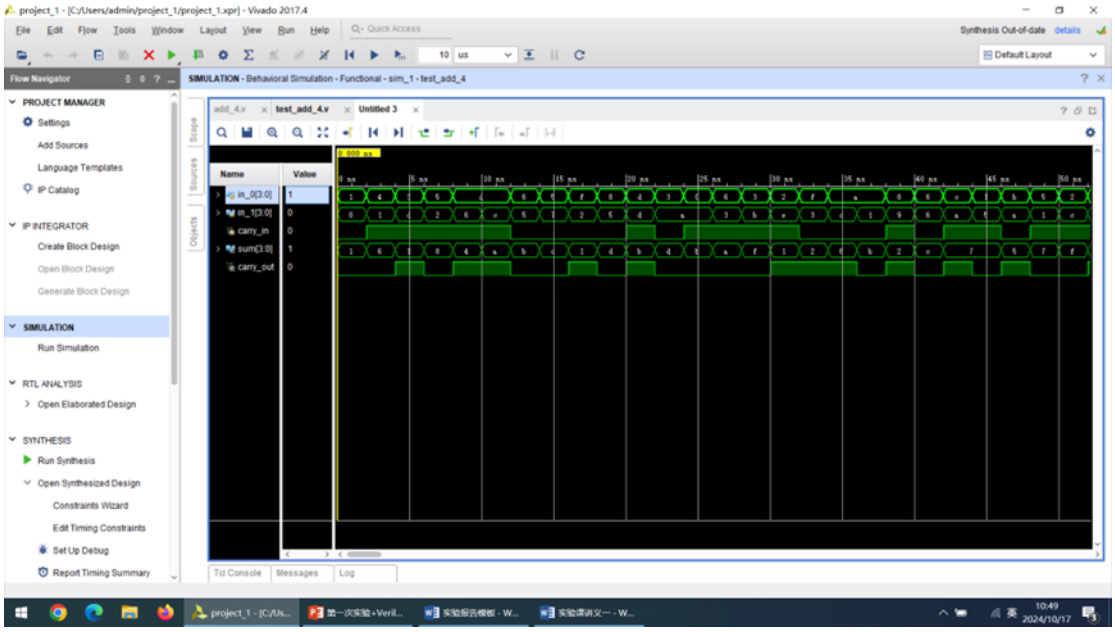

图 1: 4 位加法器 Simulation 结果

## 5.2 全加器 (门电路版本)

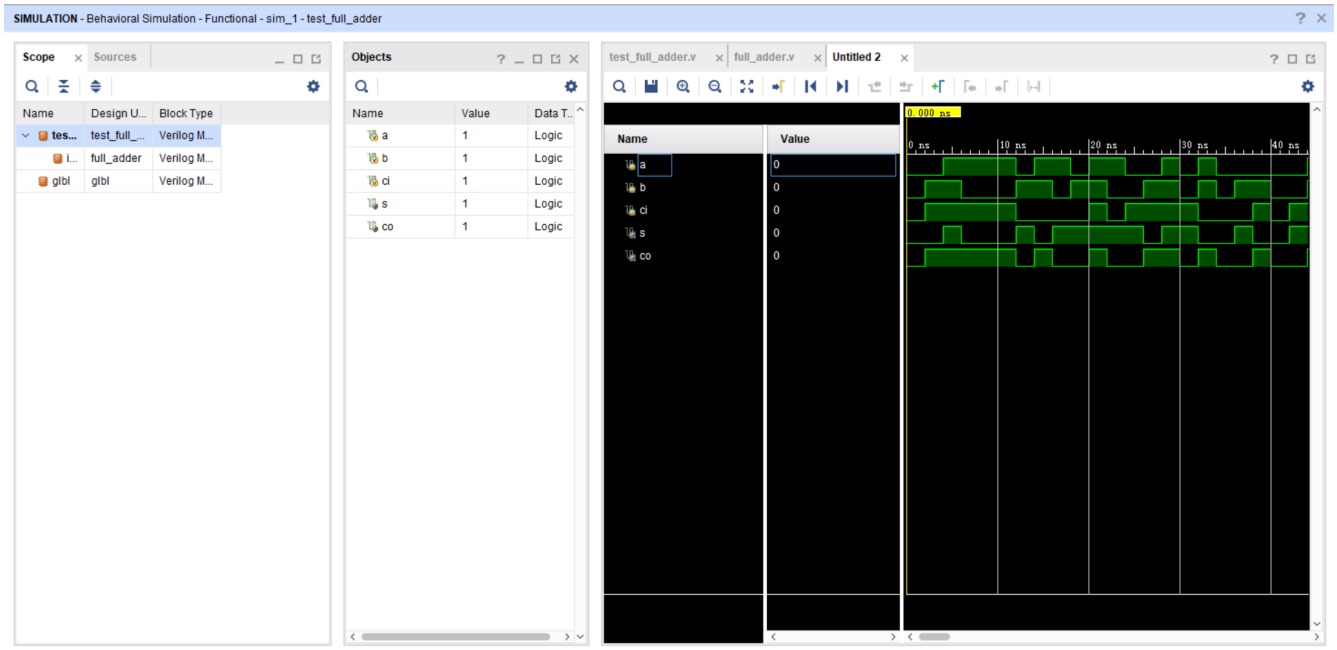


图 2: 全加器 Simulation 结果

### 5.3 3–8 译码器 (高电平有效)

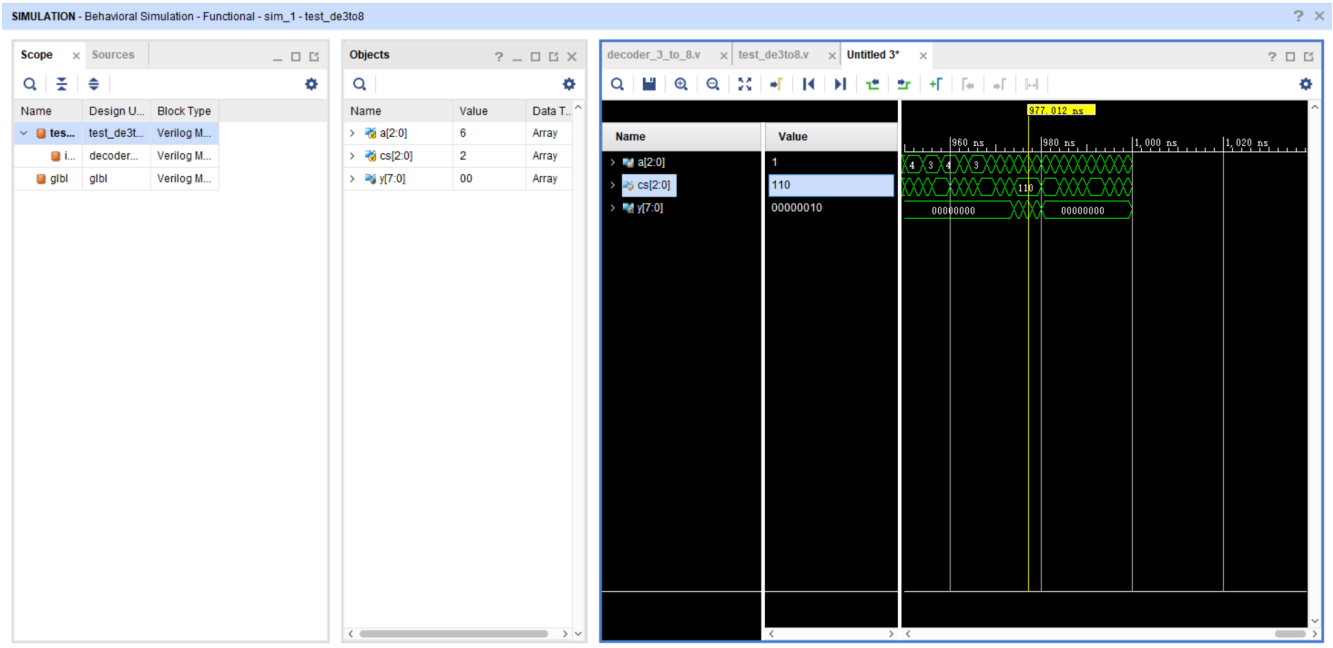


图 3: 3–8 译码器 Simulation 结果

## 6 实验总结

通过本次实验，我熟悉了 Vivado 软件的操作过程和操作逻辑，进行了四位加法器的设计及其他电路的设计，验证了其功能。我得以深入理解组合逻辑电路的设计机制以及 Verilog 语言的使用，为日后更深入学习数字电路课程打好了基础。

## 7 源代码

### 7.1 4 位加法器

File: add_4.v

```
`timescale 1ns / 1ps

module add_4(
    input [3:0] add_0,
    input [3:0] add_1,
    input c_in,
    output [3:0] out,
    output c_out
    );
    assign {c_out, out} = add_0 + add_1 + c_in;
endmodule
```

File: test_add_4.v

```
`timescale 1ns / 1ps

module test_add_4();
```

```
    reg [3:0] in_0;
    reg [3:0] in_1;
    reg carry_in;
    wire [3:0] sum;
    wire carry_out;

    add_4 instance_add_4 (
        .add_0(in_0),
        .add_1(in_1),
        .c_in(carry_in),
        .out(sum),
        .c_out(carry_out)
    );

    initial begin
        in_0 = 4'b0001;
        in_1 = 4'b0000;
        carry_in = 1'b0;
    end

    always begin
        #2;
        in_0 = $random() % 16;
        in_1 = $random() % 16;
        carry_in = $random() % 2;
    end
endmodule
```

## 7.2 全加器 (门电路版本)

File: full_adder.v

```
`timescale 1ns / 1ps

module full_adder(
    input a,
    input b,
    input ci,
    output s,
    output co
    );
    assign s = (a ^ b) ^ ci;
    assign co = (a & b) | (a & ci) | (b & ci);
endmodule
```

File: test_full_adder.v

```
`timescale 1ns / 1ps

module test_full_adder();
    reg a, b, ci;
    wire s, co;
```

```
    full_adder inst_fa(
        .a(a),
        .b(b),
        .ci(ci),
        .s(s),
        .co(co)
    );

    initial begin
        a = 1'b0;
        b = 1'b0;
        ci = 1'b0;
    end

    always begin
        #2;
        a = $random() % 2;
        b = $random() % 2;
        ci = $random() % 2;
    end

endmodule
```

## 7.3 3–8 译码器 (高电平有效)

File: decoder_3to8.v

```
`timescale 1ns / 1ps

module decoder_3_to_8(
    input [2:0] a,
    input [2:0] cs,
    output [7:0] y
    );

    reg [7:0] out;

    always @(*) begin
        case(a)
            3'b000: out = 8'b1;
            3'b001: out = 8'b10;
            3'b010: out = 8'b100;
            3'b011: out = 8'b1000;
            3'b100: out = 8'b10000;
            3'b101: out = 8'b100000;
            3'b110: out = 8'b1000000;
            3'b111: out = 8'b10000000;
        endcase

        if (cs != 3'b110) begin
            out= 8'b0;
        end
```

```
    end

    assign y = out;
endmodule
```

File: test_de3to8.v

```
`timescale 1ns / 1ps

module test_de3to8();
    reg [2:0] a, cs;
    wire [7:0] y;

    decoder_3_to_8 inst_de3t8(
        .a(a),
        .cs(cs),
        .y(y)
    );

    initial begin
        a = 3'b0;
        cs = 3'b0;
    end

    always begin
        #2;
        a = $random() % 8;
        cs = $random() % 8;
    end
endmodule
```