初探Verilog

mofianger

20250410

Verilog vs FPGA

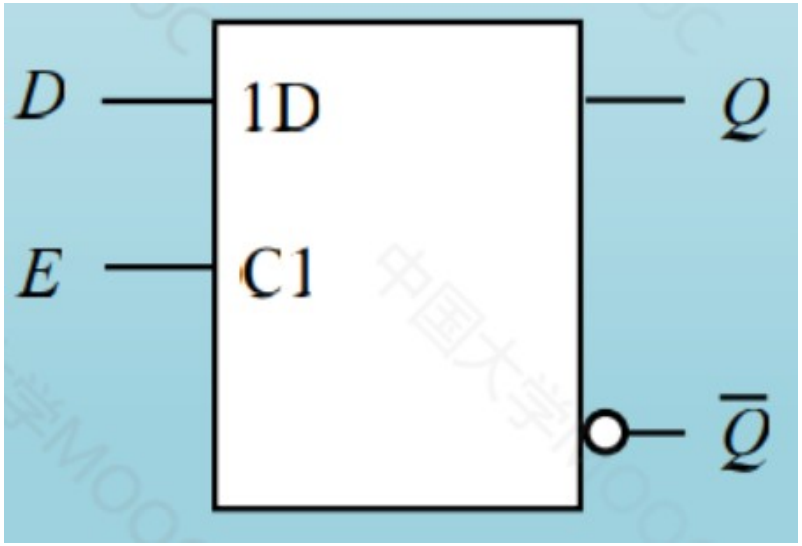
- Verilog 是硬件描述语言
- FPGA 是现场可编程门阵列
- C vs MCU

Verilog vs C

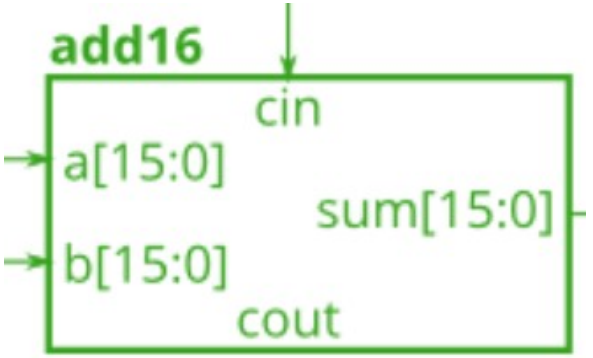
- Verilog 描述硬件，并行
- C 描述算法，串行
- 硬件的底层：无数个逻辑门

Verilog 设计思路

- 模块化
- 预期的功能，定义输入与输出
- 例子1：D锁存器
  - 输入 数据D，使能信号EN
  - 输出 Q
  - 功能，当EN有效：Q跟随D
  - 当EN失效：Q保持原来的值

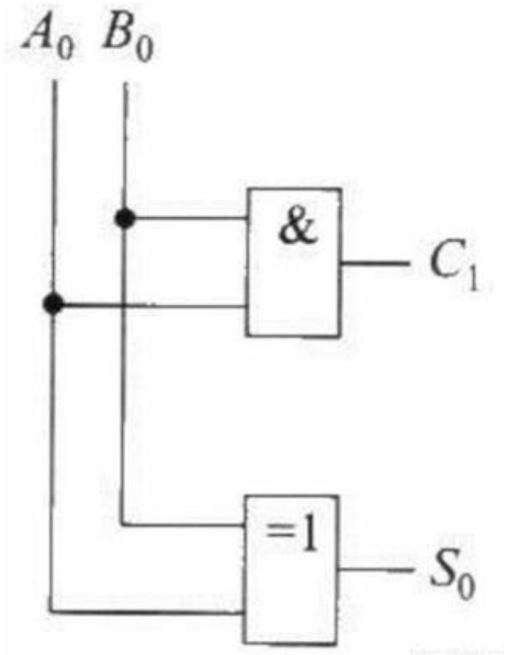


- 例子2：16位加法器
  - 输入 16位二进制数 A 和 B 表示待加数，1位二进制数 cin 表示低位进位
  - 输出 16位二进制数 S 表示求和，1位二进制数 cout 表示进位



组合逻辑与时序逻辑

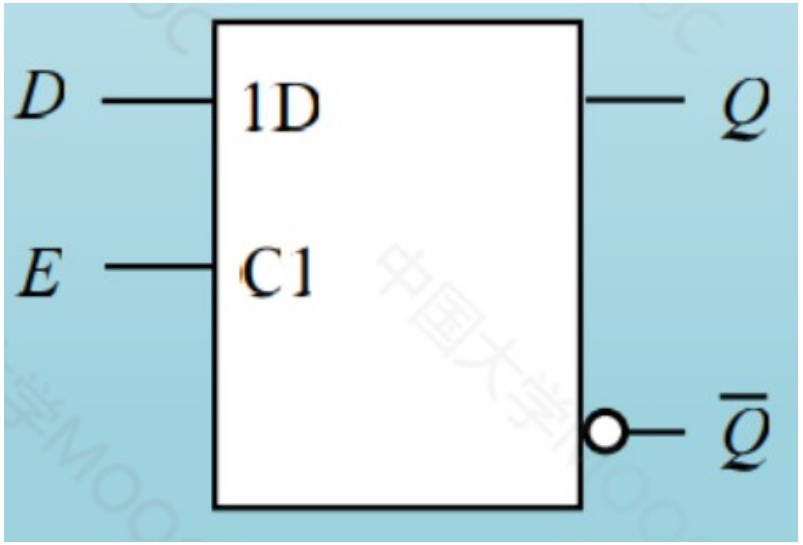
- 组合逻辑：StateA --> OutputA
- 时序逻辑：引入了记忆



A	B	C（进位）	S（和）
0	0	0	0
0	1	0	1
1	0	0	1
0	0	1	0

组合逻辑与时序逻辑

- 组合逻辑：StateA --> OutputA
- 时序逻辑：输出不仅与当前状态相关，还与过去状态相关 引入了**记忆**



D（数据）	E（使能）	Q（输出）
0	1	0
1	1	1
0	0	Don’t change
1	0	Don’t change

组合逻辑与时序逻辑

- 组合逻辑：StateA --> OutputA
- 时序逻辑：输出不仅与当前状态相关，还与过去状态相关 引入了**记忆**
- **学好数电很重要**

定义变量

- wire [15:0] A; //16位
- reg B; //1位
- wire C;
- reg [3:0] D;
- Wire：理解为连线，连接在一起
- Reg：寄存器，存储功能

简单的逻辑运算

- 与：&
- 或：|
- 非：~
- 异或：^

为变量赋值

- 为Wire型变量赋值
  - wire x;
  - assign x = a & b;
- 为Reg型变量赋值
  - 阻塞赋值
    - reg x;
    - always@(*) x = a & b;
  - 非阻塞赋值
    - reg x;
    - always@(*) x <= a & b;

定义与实例化模块

- module module_name(input a, input b, ……, output c, output d, ……);
  - 实现模块
- endmodule
- module_name instance1(.a(A), .b(B), ……);
  - .a(A) 将A信号连接到模块中的a
- 模块不嵌套定义，通过实例化，在模块中“使用”另一个模块

always块

- always@(*) begin
    - 一些操作
- end
- always@(posedge clk or negedge rst) begin
    - 一些操作
- end
- always中赋值的变量需要是reg类型

= 与 <= 辨析

- 一般原则：
- 组合逻辑 阻塞
- 时序逻辑 非阻塞
- 混合 非阻塞
- 不要同时在一个always块中使用阻塞和非阻塞赋值

开发流程

- 设计定义（要完成什么功能）
- 设计输入输出
- 分析和综合（写代码，软件综合）
- 功能仿真（前仿真，编写testbench，modelsim仿真）
- 布局布线
- 时序仿真（后仿真，关注延迟，更真实的表现）
- IO分配与配置文件生成
- 烧写，上板测试
- 在线调试

示例一 2-1mux 小灯

- 功能：实现一个2-1多路选择器，用于控制LED亮灭
- input:
  - 控制信号 sel
  - 输入信号 d0
  - 输入信号 d1
- output:
  - 控制小灯的信号led
- 逻辑：
  - 当sel = 0, led = d0;
  - 当sel = 1, led = d1;

示例二 全加器

- 功能：实现一位二进制数加法，a + b，考虑进位
- input:
  - 待加数 a
  - 待加数 b
- output:
  - 进位标志 c
  - 和 s
- 逻辑:
  - 算数逻辑