数字逻辑串讲

信安171 刘朴淳

by Curled

链接:
https://pan.baidu.com/s/1SBxYH9L-yb0WwO4aVEuCvA 提取码: 2333

考点

名词解释
（判断）

verilog读程、填空、编程
（填空、大题）

有限状态机※
（大题）

名词解释

PLD实现方法中最常用的器件是_____和_____。

EDA、PLD、FPGA、CPLD、A

EDA：电子设计自动化

以计算机为工具，在EDA软件平台上，用硬件描述语言VerilogHDL完成设计文件，然后由计算机自动地完成逻辑编译、化简、分割、综合、优化、布局、布线和仿真，直至对于特定目标芯片的适配编译、逻辑映射和编程下载等工作。

PLD：可编程逻辑器件

FPGA(1K ~ 10M门，集成度大)

CPLD(500 ~ 50000 门，集成度小)

ASICS：专用集成电路
单客户

ASSP：专用应用标准产品
多客户

制造时已是硬连接

一句话说清楚它们之间的关系

利用EDA技术进行电子系统设计的最后目标是完成ASIC的设计和实现

FPGA和CPLD是实现这一途径的主流器件

下列 _____ 流程是正确的基于 EDA 软件的 FPGA / CPLD 设计流程。

A、原理图 /HDL 文本输入→适配→综合→功能仿真→编程下载→硬件测试

B、原理图 /HDL 文本输入→功能仿真→综合→适配→编程下载→硬件测试

C、原理图 /HDL 文本输入→功能仿真→综合→编程下载→适配→硬件测试

D、原理图 /HDL 文本输入→功能仿真→适配→编程下载→综合→硬件测试

HDL——硬件描述语言
Verilog HDL、VHDL

具有特殊结构能够对硬件逻辑电路的功能进行描述的一种高级编程语言

综合:从真值表产生逻辑电路的过程

面向RTL——目标：可综合

并行

IEEE 标准的硬件描述语言是 verilog HDL 和 VHDL 。

判断

1.用HDL描述数字系统时，最常用的描述层次是RTL级（）

2. 不管是RTL级还是行为级，都要求被描述的目标有可综合的限制。()

同步时序
异步时序

同步复位
异步复位

判断

同步时序逻辑和异步时序逻辑的区别是同步时序逻辑的所有触发器变化在同一时钟信号下发生（）

锁存器（Latch）和触发器的区别是，锁存器是时钟边沿触发的，触发器是电平触发的。（）

数字系统功能分类

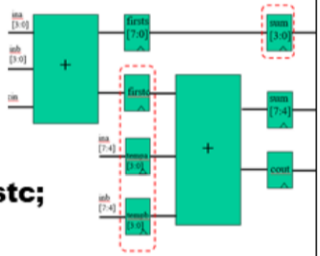
数据处理电路和控制电路

流水线

```
module adder_nonpipe(cout, sum,
ina, inb, cin, enable);

output cout;
output [7:0] sum;
input [7:0] ina, inb;
input cin, enable;
reg cout;
reg [7:0] sum;
reg [7:0] tempa, tempb;
reg tempc;

always @(posedge enable)
    begin
tempa = ina;
tempb = inb;
tempc = cin;
    end
always @(posedge enable)
    begin
{cout,sum} = tempa + tempb + tempc;
    end

endmodule
```

两级流水实现的8位加法器

```
module adder_pipe2(cout,sum,ina,inb,cin,clk);
input[7:0] ina,inb; input cin,clk; output reg[7:0] sum;
output reg cout; reg[3:0] tempa,tempb,firsts; reg firstc;

always @(posedge clk)
begin
   {firstc,firsts}=ina[3:0]+inb[3:0]+cin;
   tempa=ina[7:4];
   tempb=inb[7:4];
end

always @(posedge clk)
begin
   {cout,sum[7:4]}=tempa+tempb+firstc;
   sum[3:0]=firsts;
end
endmodule
```



将8位数每四位分2次相加，形成两级流水线运算过程。

数字系统设计描述

硬件描述语言

组合与时序

数字系统设计

状态机

Verilog读程、填空、编程

数值的表示方法

5’b00001 <位宽>’<进制><数字>

’b00001 ’<进制><数字>

1 <数字>

reg还是wire？

wire——一根物理连线

reg——一个存储单元

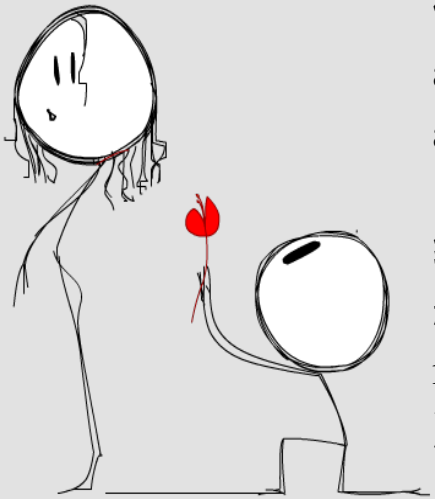
？

驱动

连续赋值：assign

过程赋值：always、initial

REG & WIRE



未赋值：reg为不定态X，wire为高阻态Z

仿真：wire对应于连续赋值，如assign；reg对应于过程赋值，如always，initial

综合：wire型的变量综合出来一般是一根导线；
reg变量在always块中有两种情况。

Golden Rules

必须用wire的情况

assign语句

元件例化时的输出

命名:必须以字母起头，
其中可以包括任何字母和数字以及下划线符 “_” 和美圆符号 $ 。

判断

在verilog标准里，9m0on是合法的标识符()。

数组还是Vector？

vector

```
reg [MSB:LSB] reg_name;
```

Array

```
reg reg_name [MSB:LSB];
```

判断

Verilog中，reg [7:0] a和reg b[7:0]表示的意义相同，都可以直接赋值a=0;b=0;()

阻塞还是非阻塞？

假设一个模块，有 2 个寄存器， b 和 c，初值都是 1，a初值为2。 a 为输入信号线。在某个时刻，因为某种原因，模块被触发执行。

```
always@(a ,b)
begin
   b = a;
   c = b;
end
```

```
always@(posedge clk)
begin
   b <= a;
   c <= b;
end
```

右式计算

左式更新

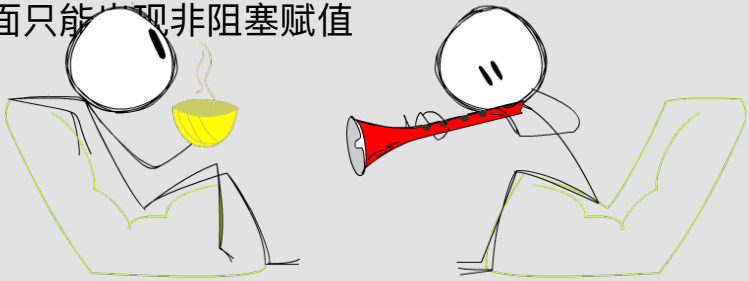
阻塞&非阻塞

Golden Rule

时序逻辑一定用非阻塞赋值“<=”,一旦看到敏感列表有 posedge 就用“<=”。

组合逻辑一定用“=”，一旦敏感列表没有 posedge 就用“=”，一旦看到 assign 就用”=”。

时序逻辑和组合逻辑分成不同的模块，即一个 always 模块里面只能出现非阻塞赋值“<=”或者“=”。



a=5d'5;b=3d'3;”那么{a,b}=_____;

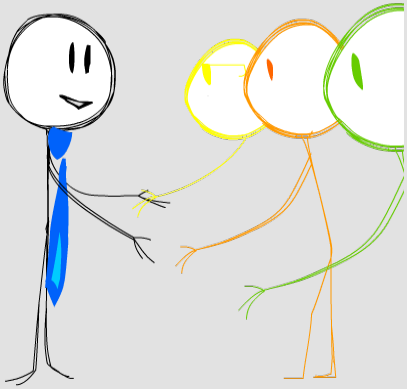
如果reg [5:0] c ; c = {a,b}。

时间尺度

`timescale time_unit / time_precision

time_unit

time_precision



```
`timescale 100ns / 10ns
module tb;
reg set;
parameter d1 = 20,d2 = 1.5,d3 = 1.54,d4 =
1.55;
initial begin
      #d1   set = 0;
  #d2   set = 1;
  #d3   set = 0;
      #d4   set = 1;
end
endmodule
```

判断

1.在verilog语句中，always和always@(*)含义完全相同。（）

2.在组合逻辑中，出现latch的原因是敏感信号不全，或者分支结构遍历不全（）

避免Latch

锁存器容易产生毛刺

锁存器在ASIC设计中应该说比ff要简单，但是在FPGA的资源中，大部分器件没有锁存器这个东西，所以需要用一个逻辑门和ff来组成锁存器，这样就浪费了资源。

锁存器的出现使得静态时序分析变得更加复杂

试设计一个 D锁存器，规定模块定义为 module D_latch(D,clk,Q), 其中 Q为锁存器输出， D为输入， clk为时钟信号输入。

试设计一个 D触发器，规定模块定义为 module D_ff(D,clk,Q), 其中 Q为锁存器输出， D为输入， clk为时钟信号输入。

试设计一个 3/8 译码器，规定模块定义为 module Decoder(Out,In,En), 其中 Out为译码器输出， In 为译码器输入， En 为译码使能输入。

假设Clock频率是102.4KHz,C9的频率是多少?

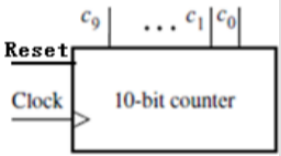
A 100Hz

B 1024Hz

C 102.4Hz

D 1000Hz

```
module clk_div_phase (
Clock, Reset, c0, c1, c2, c3, c4, c5, c6, c7, c8, c9)
  input Clock, Reset;
  output c0, c1, c2, c3, c4, c5, c6, c7, c8, c9 ;
  wire c0, c1, c2, c3, c4, c5, c6, c7, c8, c9 ;
  reg [9:0] cnt;

  always @ (posedge Clock)
    if (Reset)
       cnt<=10' b0000000000;
    else
       cnt <= cnt + 1;
  assign c0 = ~cnt [0];
  assign c1 = ~cnt [1];
  assign c2= ~cnt [2];
  assign c3 = ~cnt [3];
  assign c4 = ~cnt [4];
  assign c5= ~cnt [5];
  assign c6 = ~cnt [6];
  assign c7 = ~cnt [7];
  assign c8= ~cnt [8];
  assign c9 = ~cnt [9];
endmodule
```



时钟分频

N分频（N为偶数），计数器计数到N/2-1翻转。

N分频（N为奇数）？

偶数分频

实现一个四分频器，模块定义为module clk_2_diver(clkin,clkout,rst),

有限状态机
FSM

当前状态

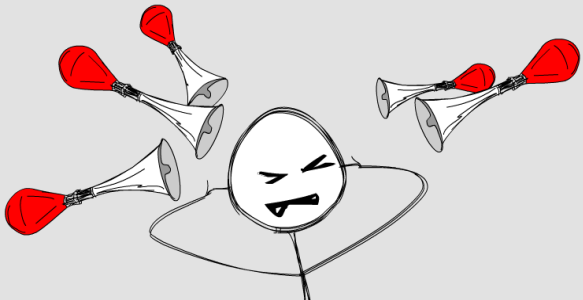
输出

次态

输入

Moore型

Mealy型



写一个状态机，验证一串二进制bit，包含偶数个 0 和奇数个 1。

判断一个 binary string 是否能被 3 整除。

旅鼠问题

旅鼠是一种头脑简单的动物，非常简单,这里将用一个有限状态机对它进行建模.

在旅鼠的二维世界中,旅鼠可以处于两种状态之一：靠左行走或靠右行走.如果碰到障碍物,它会改变方向.比如：如果一只旅鼠在左边撞了墙,它就会靠右走.如果它撞到右边的石头,它就会靠左走.如果它在两侧同时发生碰撞,它仍然会改变当前方向.

实现一个摩尔状态机来模拟旅鼠的行为：异步复位（高电平有效）到靠左走，左右是否发生碰撞为状态机的输入（bump_left,bump_right)，靠左走或靠右走为状态机的输出(walk_left,walk_right).

除了靠左和靠右走,如果地面消失了旅鼠会下落并大喊aaah!.

当地面消失时（ground=0）,旅鼠会下坠并说“aaah！”.当地面重新出现(ground=1)时,旅鼠将恢复为坠落前相同的方向行走.下落过程中撞击无效,与地面消失同一周期被撞击(但尚未坠落)、或落地同一周期被撞击,不影响行走方向.

具体时序见波形图.

建立一个模拟这种行为的有限状态机.

除了行走和下落之外,有时还可以告诉旅鼠做一些有用的事情,比如挖洞(dig=1时开始挖洞).如果一只旅鼠正在地面上行走(ground=1),它可以转换成挖掘状态直到到达端点(ground=0).在那一点上,由于没有地面,它会掉下来(aaah!),然后当它再次落地后将继续沿着它原来的方向行走.和坠落一样,在挖掘时被撞击没有效果,在坠落或没有地面时挖掘无效.

换言之,一个行尸走肉的旅鼠可能会跌落、dig或改变方向.如果满足这些条件中的一个以上,则跌落的优先级高于dig,dig的优先级高于切换方向.具体时序见波形图.

扩展之前的有限状态机来模拟这种行为.

虽然旅鼠可以走路，坠落和挖掘，但旅鼠并不是无懈可击的。如果一个旅鼠下降太久（大于等于20个时钟周期）然后击中地面，它会飞溅，并永远停止行走、下降或挖掘（即所有4个输出变为0）直到状态机被复位。（注意旅鼠只会在击中地面时飞溅，不会在半空中飞溅）

扩展前面的有限状态机以模拟此行为

假设有一车库电动门由电机驱动，上下移动，向上移动开门，向下关门，当门完全打开或者完全关闭时，电机自动停转。绘制一个电动门（ElectDoor）的状态机图。

交通灯

- 正常情况**主干道绿灯**，汽车通行，人行横道红灯，禁止通行；
- 有行人要过马路，进行人行请求，10秒后，**主干道绿灯**转**黄灯**；
- **主干道黄灯**亮3秒后转**红灯**，同时人行横道红灯转绿灯，行人通行；
- 行人通行30秒后，人行横道绿灯转黄灯；
- 人行横道黄灯亮3秒后转红灯，同时**主干道红灯**转**绿灯**，并在60秒内不再响应人行请求；
- 60秒后，行人可以继续进行人行请求过马路。

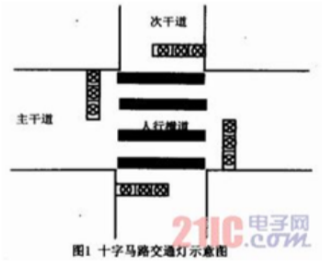


图1 十字马路交通灯示意图

谢谢