中国科学技术大学
University of Science and Technology of China

计算机组成原理

Lab1 运算器及其应用

计算机实验教学中心

2023-3-27

实验目标

- 掌握算术逻辑单元 (ALU) 的功能
- 掌握数据通路和有限状态机的设计方法
- 掌握组合电路和时序电路，以及参数化、结构化的Verilog描述方法

实验原理

1. Verilog语法复习

□ Verilog描述注意事项

- ✓ **推荐使用简单规范的描述方式**
- ✓ **组合电路**
  - 使用assign 或者 always @* 描述，“ = ” 赋值；
  - 必须配对使用if...else，case语句赋值完全，避免出现锁存器；
  - 避免出现反馈！例如，y = y + x
  - 无需复位，即组合函数的自变量中无复位信号
- ✓ **时序电路**
  - 使用always @(posedge clk, posedge rst]) 描述, “<=” 赋值
  - 边沿敏感变量表中避免出现除时钟和复位外的其他信号
  - 时钟信号避免出现在语句块内

实验原理

1. Verilog语法复习

□ 变量类型问题

- ✓ 使用always语句描述的变量，务必声明为reg类型（声明为reg类型的变量，综合后不一定生成寄存器）；
- ✓ 使用assign语句赋值的变量，应声明为wire类型；
- ✓ initial或always语句中，未定义直接赋值的变量默认为线网类型的标量（1位wire），向量变量(位宽大于 1)必须先定义后使用；
- ✓ 同一进程中尽量在一个if或case语句块中对一个变量赋值，否则后边的赋值会覆盖前面的赋值，可能导致逻辑上的问题（特例：组合逻辑描述时，为避免形成锁存器而在开始给变量赋初值）。

实验原理

1. Verilog语法复习

□ 变量类型问题

✓ **示例：变量类型**

```
wire [7:0]  a, b;
reg [7:0]  r1, r2, r3;

always @(*)        // (en, a, b)
   if (en)  r1 = a;
   else  r1= b;

always @(en, a)   // @(*)
   if (en)  r2 = a;

always @(posedge  clk)
   if (en)  r3 = a;
```

组合电路：
1. 敏感变量不要遗漏，建议用*
2. 所有条件分支均有赋值
3. 不能含有反馈，如r1=r1+1

锁存器：避免使用

寄存器：必有触发时钟

实验原理

1. Verilog语法复习

□ 多驱动与多重时钟问题

✓ **多驱动问题**

- 模块中所有的assign和always块都是并行执行的
- 不要在多个并行执行体中对同一变量赋值

✓ **多重时钟问题**

- 不能采用行为描述方式来综合实现多个时钟或多个边沿驱动的触发器

例如：

```
always @(posedge clka, posedge clkb)
always @(posedge clk, negedge clk)
```

实验原理

1. Verilog语法复习

□ 参数化模块：模块间传递参数

✓ **模块格式定义如下：**

```
module module_name
#(parameter_list 参数声明)
(端口声明) ;
 变量声明;
 逻辑功能描述;
 endmodule
```

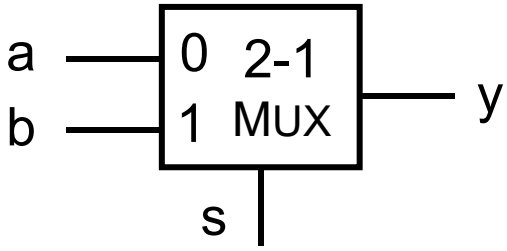
实验原理

1. Verilog语法复习

□ 参数化模块：模块间传递参数

✓ 示例1：MUX2

```
module mux2                    //模块名：mux2
   #(parameter MSB = 31,       //参数声明：数据最高有效位
   LSB = 0                     //数据最低有效位
   )
   (output [MSB : LSB] y,      //端口声明：输出数据
   input [MSB : LSB] a, b,     //两路输入数据
   input s                     //数据选择控制
   );
assign y = s? b : a;           //逻辑功能描述
endmodule
```



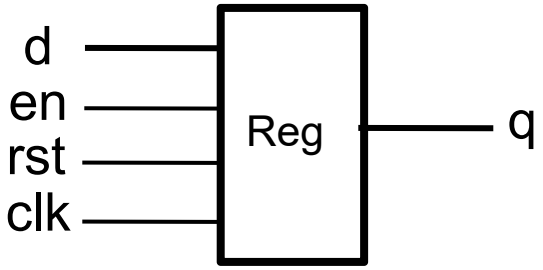
实验原理

1. Verilog语法复习

□ 参数化模块：模块间传递参数

✓ **示例2：寄存器**

```
module  register
   #(parameter WIDTH = 32, RST_VALUE = 0)
   (input  clk, rst, en,
   input  [WIDTH-1 : 0]  d,
   output reg  [WIDTH-1 : 0] q);
always @(posedge clk, posedge rst)
   if (rst) q <= RST_VALUE;
   else  if (en)
      q <= d;
endmodule
```



- d, q：输入、输出数据
- clk, rst, en：时钟、复位、使能

寄存器功能表

rst	clk	en	q	功能
1	x	x	0	复位
0	↑	1	d	置数
0	↑	0	q	保持

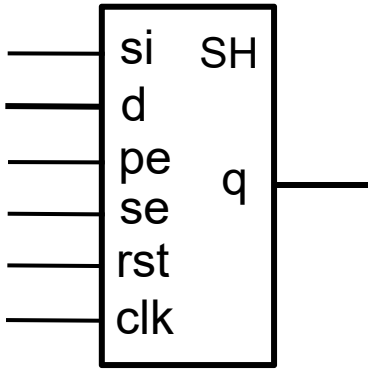
实验原理

1. Verilog语法复习

□ 参数化模块：模块间传递参数

✓ **示例3：移位寄存器**

```
module  shifter
   #(parameter N = 8, RST_VALUE = {N{1'b0}})
    (input  clk, rst, pe, se, si,
    input  [N-1: 0]  d,
    output reg  [N-1: 0] q);
 always @(posedge clk, posedge rst)
     if (rst) q <= RST_VALUE;
     else if (pe) q <= d;
     else if (se) q <= {si, q[N-1: 1]};
endmodule
```



移位寄存器功能表

rst	clk	pe	se	功能
1	x	x	x	复位
0	↑	1	x	置数
0	↑	0	1	右移
0	↑	0	0	保持

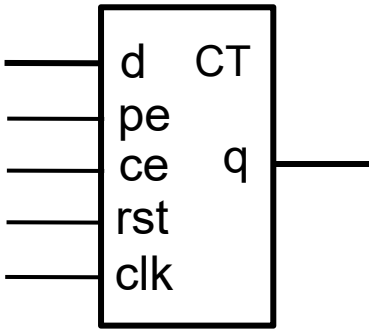
实验原理

1. Verilog语法复习

□ 参数化模块：模块间传递参数

✓ **示例4：计数器**

```
module  counter
    #(parameter N = 8, RST_VALUE = {N{1'b1}})
     (input  clk, rst, pe, ce,
     input  [N-1: 0]  d,
     output reg  [N-1: 0] q);
 always @(posedge clk, posedge rst)
     if (rst) q <= RST_VALUE;
     else if (pe) q <= d;
     else if (ce) q <= q-1;
endmodule
```



递减计数器功能表

rst	clk	pe	ce	功能
1	x	x	x	复位
0	↑	1	x	置数
0	↑	0	1	计数
0	↑	0	0	保持

实验原理

1. Verilog语法复习

□ 模块实例化

✓ **模块实例化语句格式：**

module_name #(parameter_map) instance_name (port_map);

✓ **端口映射方式：基于位置或者基于名字，不可混合使用**

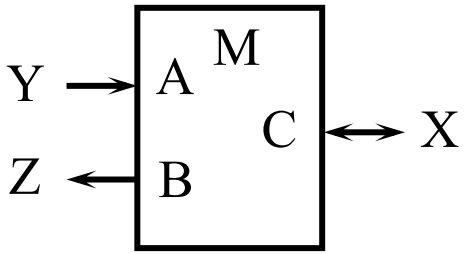
- 位置映射：按模块中端口定义的顺序传递

例如：模块定义为 moduel M(A, B, C);

M M1(Y, Z, X); //顺序很重要

- 名字映射：.PortName (value)

✓ **参数的映射方法类似**



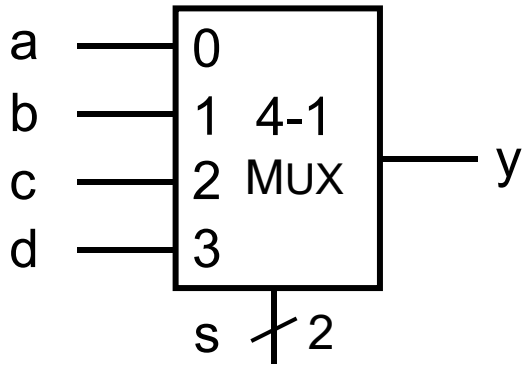
实验原理

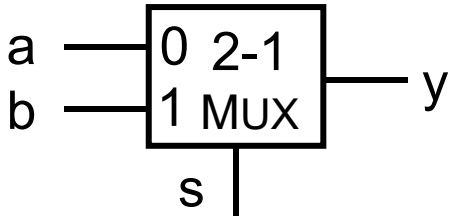
1. Verilog语法复习

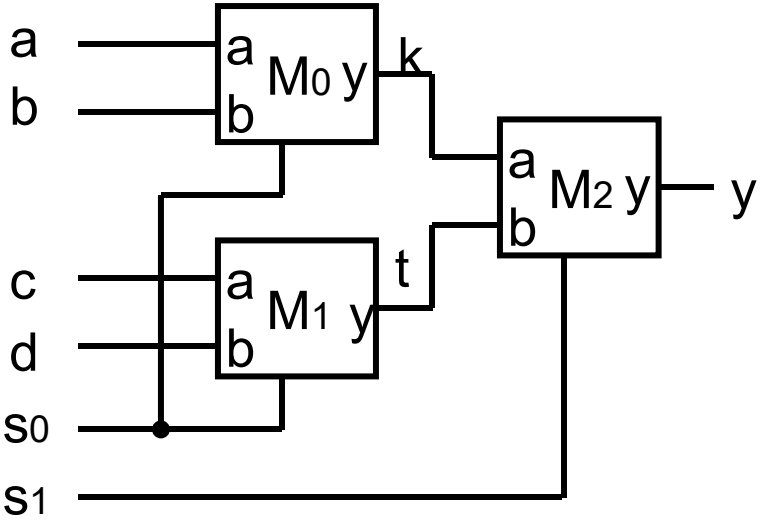
□ 模块实例化

✓ **示例：MUX4_8**

```
module mux4_8
  (output [7:0] y,
   input [7:0] a, b, c, d,
   input [1:0] s  );
wire [7:0] k, t;
//位置映射
mux2 #(7, 0) M0 (k, a, b, s[0]);
mux2 #(7) M1 (t, c, d, s[0]);
//名字映射
mux2 #(.MSB(7)) M2 (.s(s[1]), .a(k), .b(t), .y(y));
endmodule
```







实验原理

1. Verilog语法复习

□ 仿真文件编写

可参考数电实验手册《实验05_使用Vivado进行仿真》

例：时钟生成

```
reg clk;
// 时钟周期和个数
parameter CYCLE = 10, Number = 20;
//clk_gen method1:
initial  begin
clk = 0;
repeat  (2* Number)     //该repeat语句也可生成其他信号
 # CYCLE/2  clk = ~ clk;  end

//clk_gen method2:
initial  begin
 clk = 0;
 forever #CYCLE/2 clk = ~ clk; end
```

initial和#仅用于仿真，不会产生实际硬件电路

实验原理

2. FPGAOL实验平台使用

- 登录平台网站：fpgaol.ustc.edu.cn，使用统一身份认证登录，或者直接以游客身份登录

实验原理

2. FPGAOL实验平台使用

- 设备获取：点击“acquire”按钮获取一个FPGA结点
  - ✓ 成功后在下方link栏将显示链接，通过链接进入设备操作界面
  - ✓ 默认使用时长10分钟(自动释放结点，点击“release”按钮手动释放)

#	Device Type	Vacant/Total	Manual	xdc	Use
1	FPGAOL 1.0	80 / 81	FPGAOL v2.1 manual	fpgaol1.xdc	acquire release

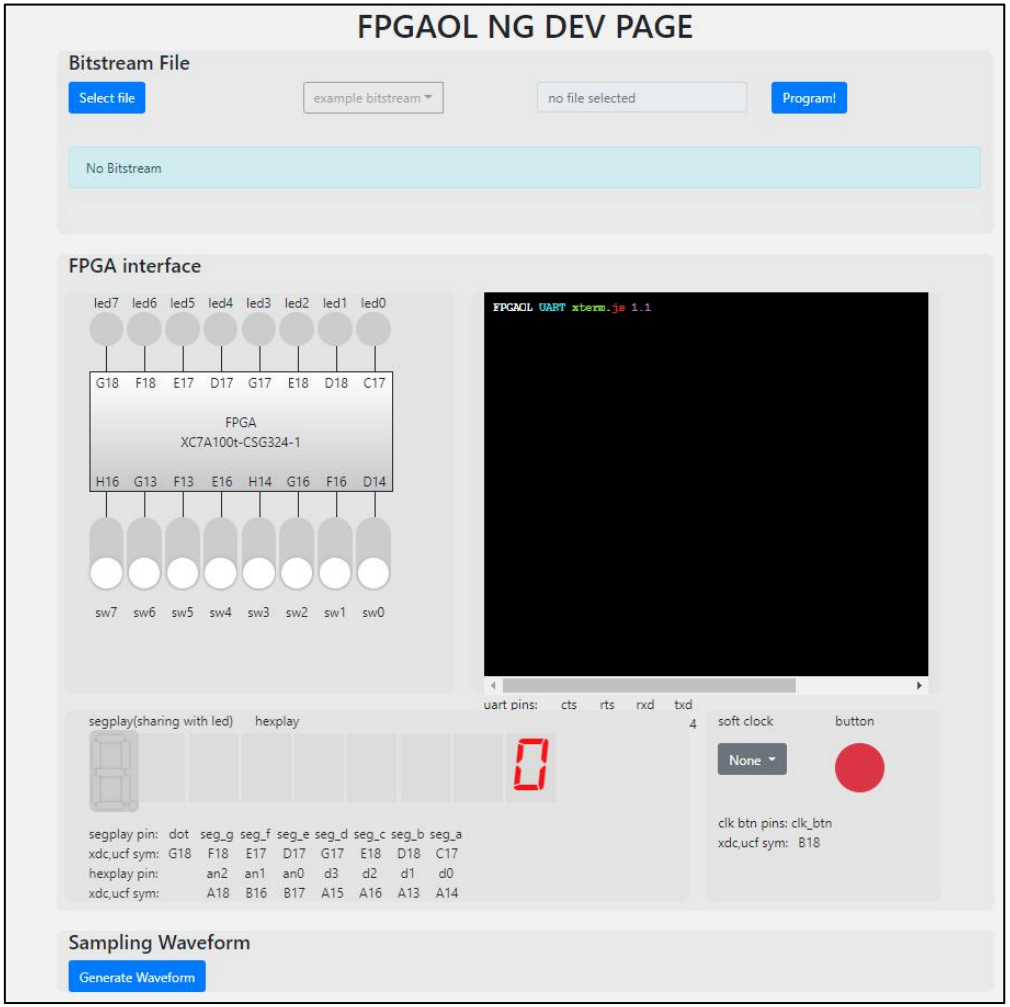
Device ID	204
Acquire Time	March 21, 2023, 3:48 p.m.
Expiration Time	March 21, 2023, 4:08 p.m.
Panel Link	http://202.38.79.134:12204/?token=HEZTAYTBHFQTCNZQME2GKOLFMI3DMNRUMIYGKNJTGJSWIY3FME4GIM3CMVRDCYZWP0614
XVC Server	202.38.79.134:-2

实验原理

2. FPGAOL实验平台使用

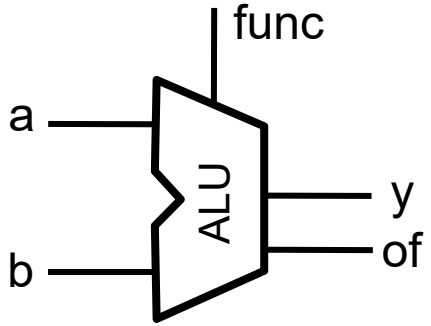
□ 烧写FPGA

- ✓ 点击“Select file”按钮
- ✓ 选择需要烧写的bit文件
- ✓ 点击“Program！”

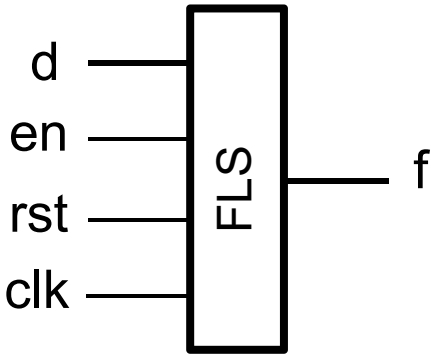


实验内容

1. 算术逻辑单元（ALU）及测试



2. ALU应用：计算斐波那契—卢卡斯数列（Fibonacci Lucas Series）

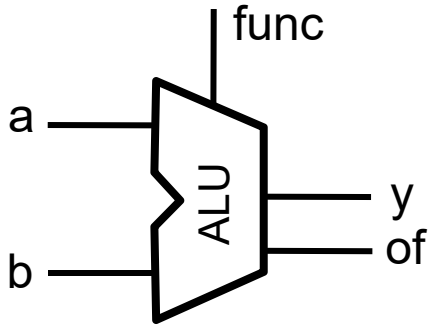


实验内容

1. 算术逻辑单元（ALU）及测试

◘ ALU端口定义及功能要求

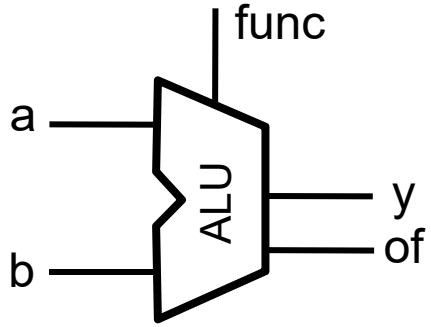
```
module alu #(parameter WIDTH = 6）     //数据宽度
(
   input  [WIDTH-1:0] a, b,        //两操作数（对于减运算，a是被减数）
   input  [3:0] func,              //操作功能（加、减、与、或、异或等）
   output  [WIDTH-1:0] y,          //运算结果（和、差 …）
   output  of                      //溢出标志of，加减法结果溢出时置1
 );
```



实验内容

1. 算术逻辑单元（ALU）及测试

□ ALU模块功能介绍



溢出判断只针对有符号数；
相等判断时，y输出为0或1（零扩展）；
比较判断时，y输出为0或1（零扩展）；
< 为带符号比较:
采用$signed(a) < $signed(b)
$<_u$ 为无符号比较:
采用 a<b
*** 表示根据运算结果设置.**

ALU模块功能表

func	y	of
0000	a + b	*
0001	a – b	*
0010	a == b	0
0011	$a <_u b$	0
0100	a < b	0
0101	a & b	0
0110	a | b	0
0111	a ^ b	0
1000	a >> b	0
1001	a << b	0
其他	0	0

绿色背景表项为必做内容，其他为选做内容

实验内容

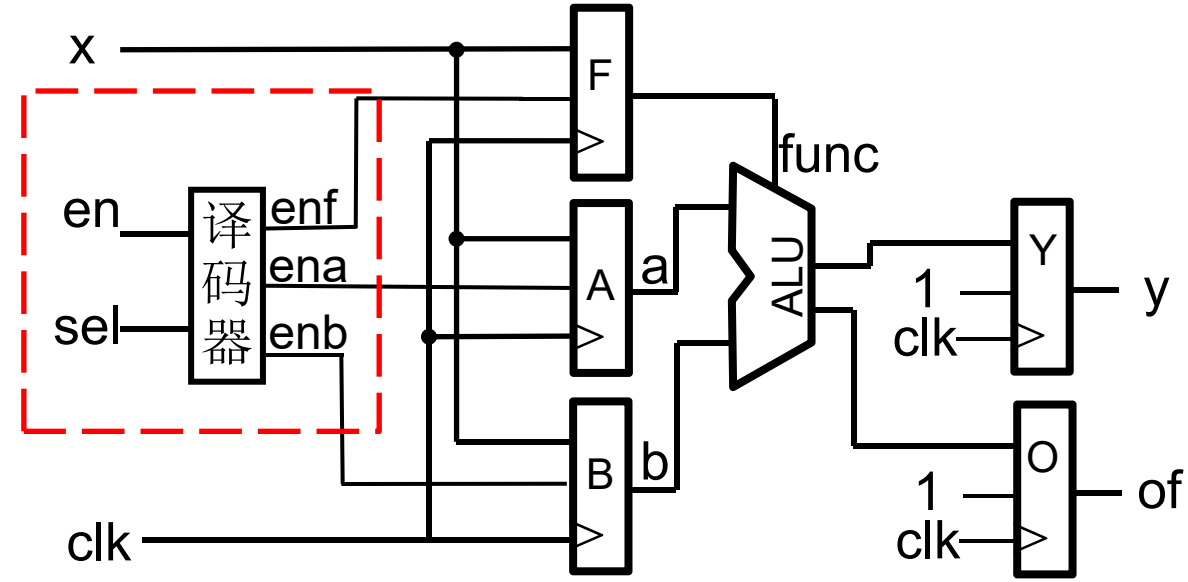
1. 算术逻辑单元（ALU）及测试

□ ALU测试电路设计

- ✓ ALU测试电路的数据通路如下图所示，该电路主要由寄存器，译码器，ALU单元组成。采用结构化描述方式设计ALU测试电路。
- ✓ 由于FPGAOL外设资源有限，因此端口需要分时复用：操作数a, b和功能f 复用开关输入x[5:0]。复用方法：通过sel和en ，译码生成寄存器使能信号ena，enb，enf（译码器真值表如下表所示），将开关输入x[5:0]分时存入寄存器 F(x[3:0])，A(x[5:0])，B(x[5:0])。

译码器真值表

en	sel	ena	enb	enf
1	00	1	0	0
1	01	0	1	0
1	10	0	0	1
1	11	0	0	0
0	xx	0	0	0



ALU测试电路的数据通路

实验内容

1. 算术逻辑单元（ALU）及测试

□ ALU测试电路端口定义及外设端口分配

```
module alu_test
(
    input clk,
    input  en,
    input  [1:0]sel,
    input  [5:0] x,
    output  [5:0] y,
    output  of
);
```

外设端口分配表

端口	外设
clk	100MHz
en	button
sel	sw[7:6]
x	sw[5:0]
y	led[5:0]
of	led[7]

实验内容

1. 算术逻辑单元（ALU）及测试

□ 查看工程电路原理图

✓ **查看工程电路原理图**

- RTL代码分析电路：Flow Navigator >>RTL Analysis >> Open Elaborated Design >> Schematic
- 综合后电路：Flow Navigator >> Synthesis >> Open Synthesized Design >> Schematic

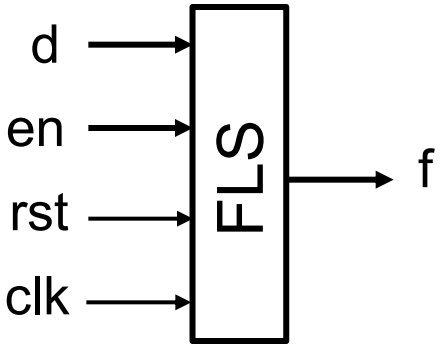
比较RTL代码分析电路原理图和综合后电路原理图电路异同

实验内容

2. ALU应用：计算斐波那契—卢卡斯数列（FLS）

□ FLS模块端口定义

```
module  fls
(
  input  clk, rst,        //时钟，复位（高电平有效）
  input en,               //输入输出使能，
                          按一次使能一个数据输入/输出
  input  [6:0]  d,        //输入数列初始项
  output [6:0]  f         //输出数列
);
```



□ FLS模块外设端口分配

端口	外设
clk	100MHz
rst	sw7
d	sw[6:0]
en	button
f	led[6:0]

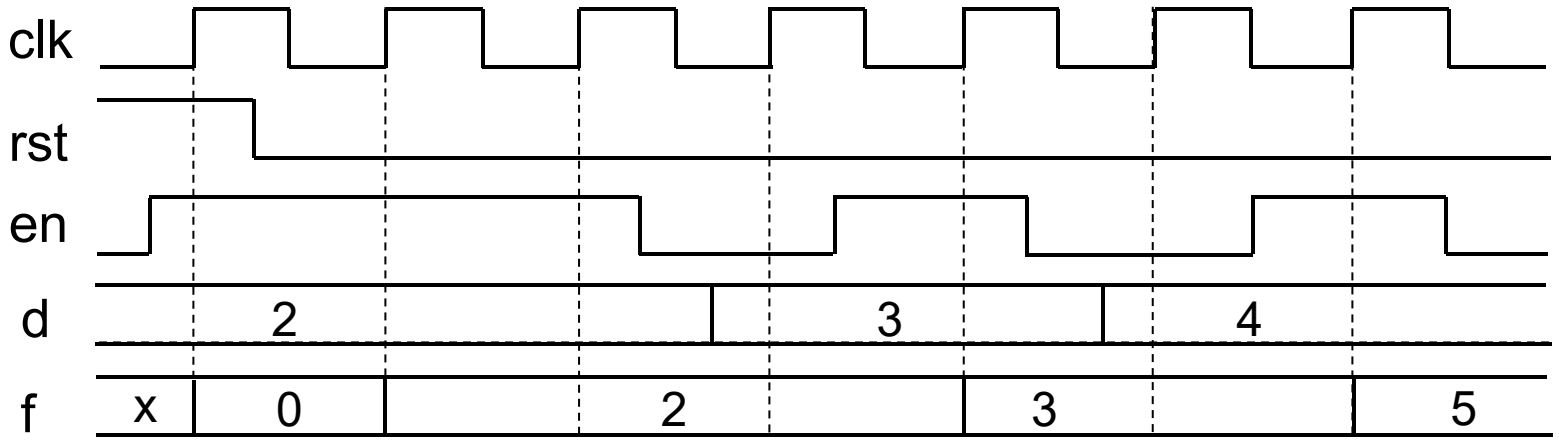
实验内容

2. ALU应用：计算斐波那契—卢卡斯数列（FLS）

□ FLS输入、输出逻辑

- ✓ 复位rst有效时（高电平），输出f = 7'd0;
- ✓ FLS数列的前两项从开关输入，即f0 = d0,f1 = d1;
- ✓ 正常工作时， fn = fn-2 + fn-1 (n > 1) ;
- ✓ 模块依次输出f = f0、f1、f2、f3 …fn…

□ FLS模块时序图



实验内容

2. ALU应用：计算斐波那契—卢卡斯数列（FLS）

□ FLS逻辑设计要求

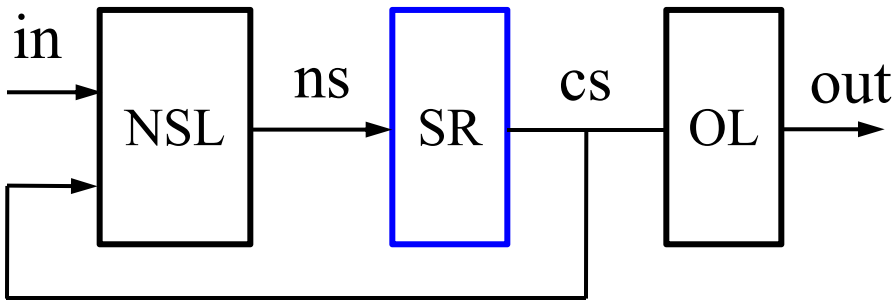
- ✓ 画出数据通路和有限状态机状态转换图
- ✓ 代码描述方式：结构化描述
- ✓ 控制器（数据通路中的控制信号）：采用Moore型有限状态机

□ FSM描述模式（采用三段式）

```
// 描述CS
always  @(posedge clk)
  if (rst)   cs <= S0;     //同步复位
  else   cs <= ns;

// 描述NS
always  @* begin
      ns = cs;                  //默认赋值
  case (cs)
        S0: begin
            ……
             end
  endcase
  end
```

```
//描述输出
assign out = (cs == S0) ? 1'b1:1'b0;
```



Moore型有限状态机

实验要求 [必做]

1. 算术逻辑单元（ALU）及测试

- 正确实现ALU单元必做部分的逻辑设计；
- 完成ALU测试电路的逻辑设计和功能仿真；
- 查看ALU测试电路的RTL分析和综合后电路图；
- 完成ALU测试电路下载测试。

2. ALU应用：计算斐波那契—卢卡斯数列（FLS）

- 完成FLS的逻辑设计：
  画出数据通路及状态机，代码采用结构化方式描述；
- 完成FLS功能仿真；
- 完成FLS下载测试。

实验要求 [选做]

1. 算术逻辑单元（ALU）及测试

- 完成ALU中的全部运算功能设计。

2. ALU应用：计算斐波那契—卢卡斯数列（FLS）

- 对FLS的数据通路进行修改，使得我们可以动态修改ALU的工作模式，而不是仅工作在加法模式。（相关外设使用规则自拟）

只有完成必做部分的全部内容后，选做部分的分数才会被计算

The End