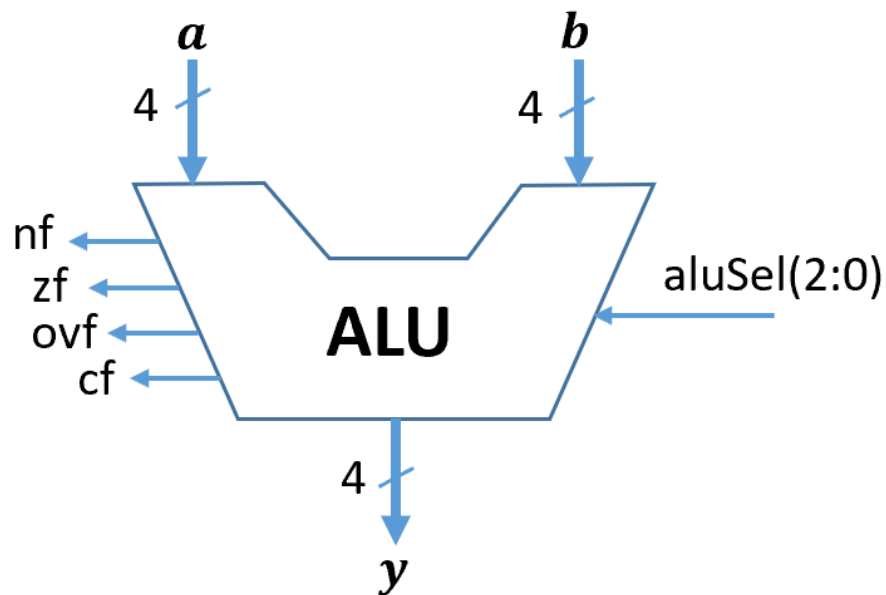


实验3：算术逻辑单元



实验2：七段数码管 (分层设计)

Sources



Design Sources (1)

▼ x7seg_Top (x7seg_Top.sv) (1)

▼ X7 : x7seg (x7seg.sv) (1)

● H7 : Hex7Seg (Hex7Seg.sv)

```
1 module Hex7Seg(input logic [3:0] digit,
2               output logic [6:0] a2g );
3
4 // a2g format {a, b, c, d, e, f, g}
5 always_comb
6     case (digit)
7         'h0: a2g = 7'b0000001;
8         'h1: a2g = 7'b1001111;
9         'h2: a2g = 7'b0010010;
10        'h3: a2g = 7'b0000110;
11        'h4: a2g = 7'b1001100;
12        'h5: a2g = 7'b0100100;
13        'h6: a2g = 7'b0100000;
14        'h7: a2g = 7'b0001111;
15        'h8: a2g = 7'b0000000;
16        'h9: a2g = 7'b0000100;
17        default: a2g = 7'b0000001; //0
18    endcase
19 endmodule
```

```
1 module x7seg( input logic [31:0] data,
2               input logic      clk,
3               input logic      clr,
4               output logic [6:0] a2g,
5               output logic [7:0] an, //数码管使能
6               output logic      dp ); //小数点
7
8     logic [2:0] s; //选择哪个数码管
9     logic [3:0] digit;
10    logic [19:0] clkdiv;
11
12    //实例化 7段数码管
13    Hex7Seg H7(.digit(digit), .a2g(a2g));
14
15    assign dp = 1; // DP off
16    assign s = clkdiv[19:17]; // 190Hz
17
18    //4个数码管 4选1 (MUX44)
19    always_comb
20        case(s)
21            0: digit = data[3:0]; //SW[3:0]
22            1: digit = data[7:4]; //SW[7:4]
```

.....

```
1 module x7seg_Top(
2     input logic      CLK100MHZ,
3     input logic [15:0] SW,
4     output logic [6:0] A2G,
5     output logic [7:0] AN,
6     output logic      DP );
7
8     x7seg X7(.data({16'h1234, SW}),
9             .clk (CLK100MHZ),
10            .clr (1'b0),
11            .a2g (A2G),
12            .an  (AN),
13            .dp  (DP) );
14 endmodule
```

模块实例化

模块名称 实例名(父模块与子模块之间端口信号关联方式)



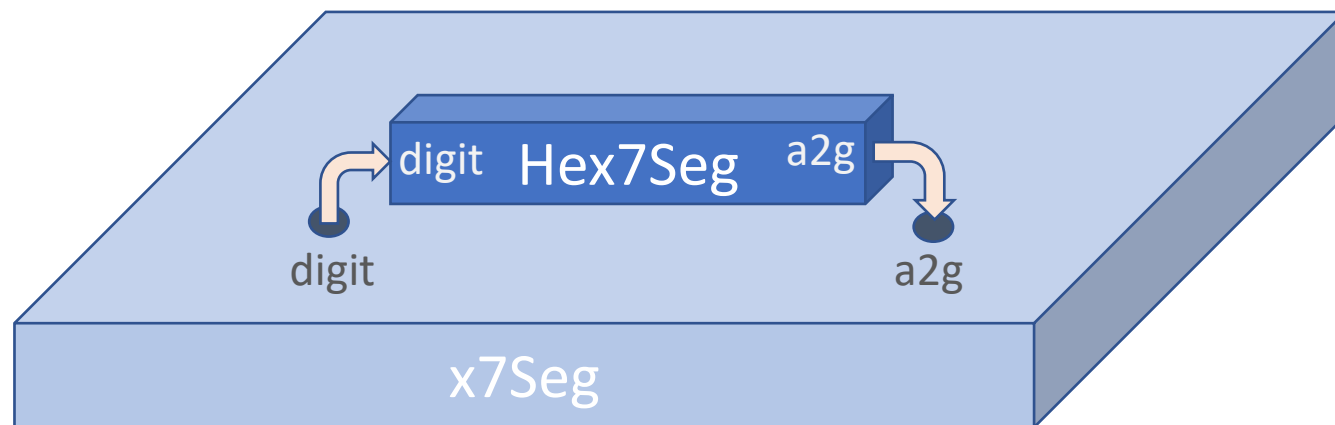
```
1 module Hex7Seg(input logic [3:0] digit,  
2               output logic [6:0] a2g );
```

位置关联法

//实例化 7段数码管

```
Hex7Seg H7(.digit(digit), .a2g(a2g))
```

x7Seg



模块实例化

模块名称 实例名(父模块与子模块之间端口信号关联方式)



```
1 module x7seg( input logic [15:0] data,  
2               input logic      clk,  
3               input logic      clr,  
4               output logic [6:0] a2g,  
5               output logic [3:0] an,  
6               output logic      dp );
```

```
1 module Hex7Seg(input logic [3:0] digit,  
2               output logic [6:0] a2g );
```

位置关联法

//实例化 7段数码管

Hex7Seg H7(digit, a2g);

x7Seg

```
1 module x7seg_Top(  
2     input logic      CLK100MHZ,  
3     input logic [0:0] SW,  
4     output logic [6:0] A2G,  
5     output logic [3:0] AN,  
6     output logic      DP );  
  
7  
8     logic [15:0] x;  
9     assign x = 'h1234; //test value  
10  
11     x7seg X7(. data(x),  
12              . clk (CLK100MHZ),  
13              . clr (SW[0]),  
14              . a2g (A2G),  
15              . an  (AN),  
16              . dp  (DP) );  
17 endmodule
```

名称关联法

形式名称

实际名称

实例化 注意事项

- 端口较少时用**位置关联法**，**名称关联法**可不考虑端口排列次序。
- **不能混合使用**位置关联法、名称关联法。
- 允许某些端口**不连接**(空白)。综合时，输入端口置高阻态，输出端口没使用。
- 模块只能以实例化的方式嵌套在其它模块内，不能在always内嵌套。
- 实例化的模块可以是：设计文件(Verilog, VHDL...)、元件库元件、IP核。

always 语句

```
1 module GoSoccer2(  
2     input logic SW0, SW15,  
3     output logic LED0, LED8, LED15 );  
4     // 无条件赋值  
5     assign LED0 = SW0;  
6     assign LED15 = SW15;  
7     assign LED8 = SW0 & SW15;  
8 endmodule
```

↑ 并行

串行 $\begin{cases} B = A \\ C = B + 1 \end{cases}$ B 将被优化没有了

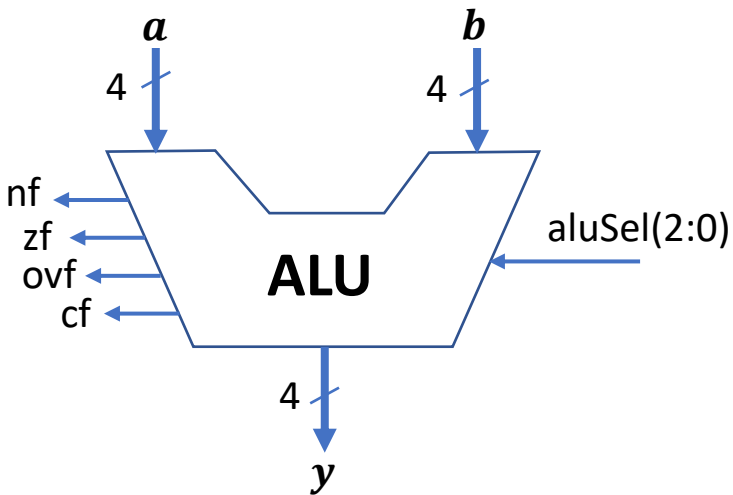
并行 $\begin{cases} B \leq A \\ C \leq B + 1 \end{cases}$

```
1 module GoSoccer_always(  
2     input logic SW0, SW15,  
3     output logic LED0, LED8, LED15 );  
4  
5     // always @(*)  
6     // always @(SW0, SW15)  
7     always_comb  
8     begin //有无条件赋值  
9         LED0 = SW0;  
10        LED15 = SW15;  
11        LED8 = SW0 & SW15;  
12    end  
13 endmodule
```

↓ 串行

ALU: 算术逻辑单元

将多种算术和逻辑运算组合到一个单元内。



4个输出标志信号

信号	含义
nf	负数标志, 当 $y(3)=1$ 时 $nf=1$
zf	零标志, 当 $y=0000$ 时, $zf=1$
ovf	溢出标志
cf	进位/借位标志

ALU中的运算和逻辑操作

aluSel(2:0)	函数	输出
000	传递 a	a
001	加法	$a + b$
010	减法1	$a - b$
011	减法2	$b - a$
100	取反	Not a
101	与	$a \text{ AND } b$
110	或	$a \text{ OR } b$
111	异或	$a \text{ XOR } b$

▼ SIMULATION

Run Simulation

➤ RTL ANALYSIS

➤ SYNTHESIS

➤ IMPLEMENTATION

Run Behavioral Simulation

Run Post-Synthesis Functional Simulation

Run Post-Synthesis Timing Simulation

Run Post-Implementation Functional Simulation

Run Post-Implementation Timing Simulation

4位ALU

```

1 module ALU4(
2     input logic [2:0] alusel, // input wire [2:0] alusel,
3     input logic [3:0] a,      // input wire [3:0] a,
4     input logic [3:0] b,      // input wire [3:0] b,
5     output logic nf,          // output reg nf, 负数标志
6     output logic zf,          // output reg zf, 零标志
7     output logic cf,          // output reg cf, 进位标志
8     output logic ovf,         // output reg ovf, 溢出标志
9     output logic [3:0] y );   // output reg [3:0] y );

```

```

10
11 logic [4:0] temp;           // reg [4:0] temp;

```

```

12
13 always_comb

```

```

14 begin

```

```

15     cf = 0;

```

```

16     ovf = 0;

```

```

17     temp = 5'b00000;

```

```

18     case(alusel)

```

```

19         3'b000: y = a; // 传递a

```

```

20         3'b001:      // a + b

```

```

21         begin

```

```

22             temp = {1'b0, a} + {1'b0, b};

```

```

23             y = temp[3:0];

```

```

24             cf = temp[4];

```

```

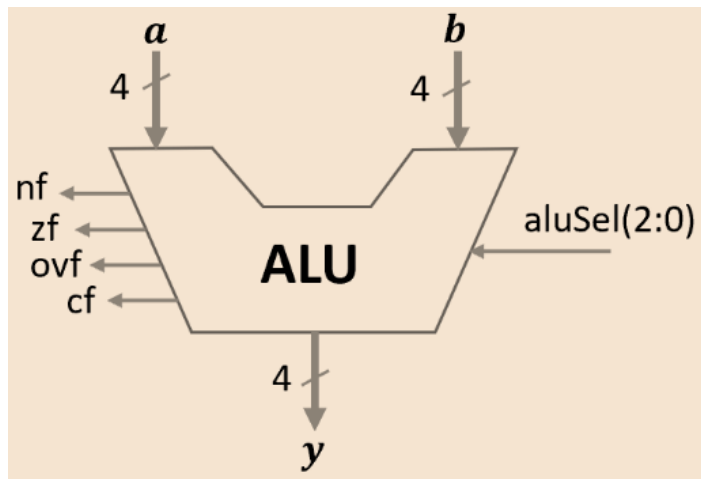
25             ovf = y[3] ^ a[3] ^ b[3] ^ cf;

```

```

26         end

```



```

27         3'b010:      // a - b

```

```

28         begin

```

```

29             temp = {1'b0, a} - {1'b0, b};

```

```

30             y = temp[3:0];

```

```

31             cf = temp[4];

```

```

32             ovf = y[3] ^ a[3] ^ b[3] ^ cf;

```

```

33         end

```

```

34         3'b011:      // b - a

```

```

35         begin

```

```

36             temp = {1'b0, b} - {1'b0, a};

```

```

37             y = temp[3:0];

```

```

38             cf = temp[4];

```

```

39             ovf = y[3] ^ a[3] ^ b[3] ^ cf;

```

```

40         end

```

```

41         3'b100: y = ~a; // NOT a

```

```

42         3'b101: y = a & b; // a AND b

```

```

43         3'b110: y = a | b; // a OR b

```

```

44         3'b111: y = a ^ b; // a XOR b

```

```

45         default: y = a;

```

```

46     endcase

```

```

47     nf = y[3];

```

```

48     if(y==4'b0000) zf = 1;

```

```

49     else          zf = 0;

```

```

50     end

```

```

51 endmodule

```


仿真

```
1  `timescale 1ns / 1ps
2  module ALU4_Sim( );
3      logic [2:0] aluSel;
4      logic [3:0] a, b, y;
5      logic nf, zf, cf, ovf;
6
7      ALU4 A4(aluSel, a, b, y, nf, zf, cf, ovf);
8
9      initial
10     begin
11         #0 aluSel = 3'b001;    // a + b
12         #0 a = 4'b0001; b = 4'b0010;    //1+2=3
13         #10 a = 4'b0001; b = 4'b0111;    //1+7=8 溢出
14
15         #10 aluSel = 3'b010;    // a - b
16         #0 a = 4'b0011; b = 4'b0001;    //3-1=2
17         #10 a = 4'b0001; b = 4'b0011;    //1-3=-2
18         #10 a = 4'b1111; b = 4'b0001;    //-1-1=-2
19         #10 a = 4'b0000; b = 4'b1000;    //0-(-8)=8 溢出
20     end
21 endmodule
```

The screenshot displays the ALU4_Sim_behav.wcfg simulation window. The top toolbar includes search, save, zoom, and playback controls. Below the toolbar is a table with the following data:

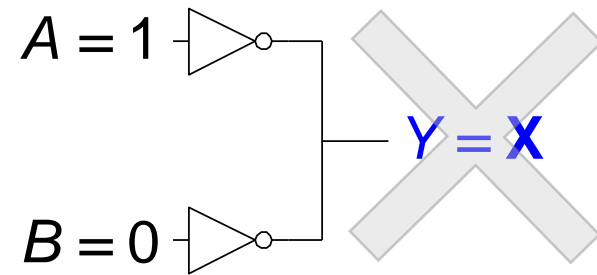
Name	Value
aluSel[2:0]	001
a[3:0]	000
b[3:0]	011
y[3:0]	100
nf	1
zf	0
cf	0
ovf	1

The waveform area shows a time axis from 0.000 ns to 60.000 ns. A context menu is open over the waveform, listing actions such as Cut, Copy, Paste, Delete, Find..., and Waveform Style. The 'Radix' menu is also open, showing options like Default, Binary (selected), Hexadecimal, Octal, ASCII, Unsigned Decimal, Signed Decimal, Signed Magnitude, Real, and Real Settings... The 'Sources' panel on the right lists design sources (ALU4), constraints, simulation sources (sim_1), and utility sources. The 'ALU4_Sim (ALU4_Sim.sv) (1)' source is highlighted.

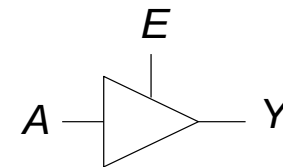
非法值 X、浮空值 Z



- **X**: 同时被0、1驱动
 - 可能值: 0、1、或中间任意值
 - 危害: 可能因**竞争**使得电路发热并损坏
 - 结论: **必须避免**!
 - 注意: 在真值表中X表示无关项, 在电路中表示非法值



- **Z**: 高阻态。既没有被1驱动, 也没有被0驱动
 - 可能值: 0、1、或中间任意值
 - 原因: 忘记将电压连接到输入端。
也可能是电路的需要, 如三态缓冲器。

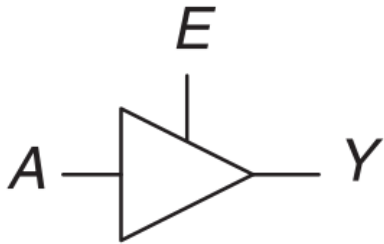


E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

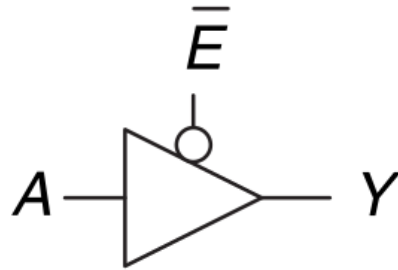


三态门缓冲器 Tristate Buffer

使能端

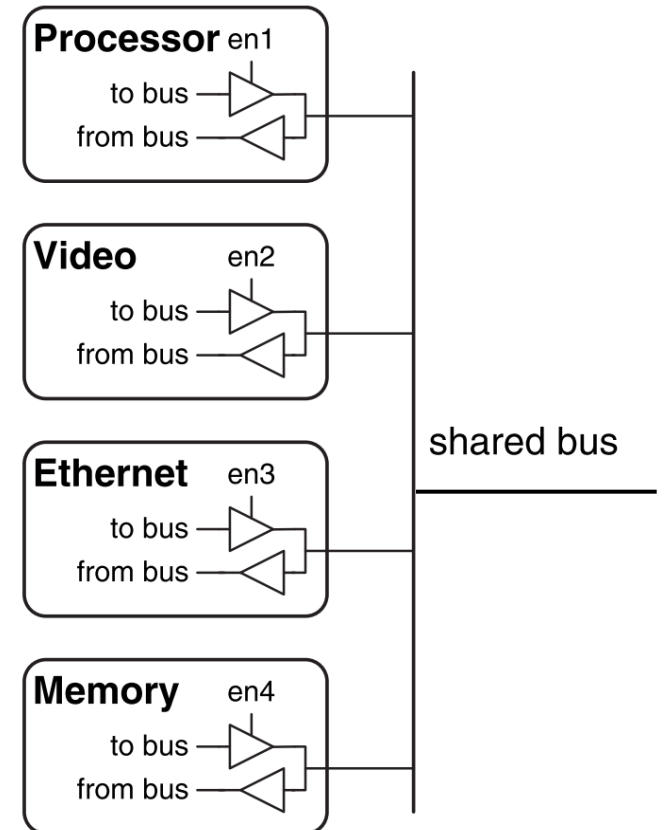


E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1



\bar{E}	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

常用于多个信号共享一条总线



- 输出有：逻辑1、逻辑0、高阻抗
- 当E有效(如，逻辑1)时，输出取决于输入
- 当E无效(如，逻辑0)时，输出高阻抗，即属于与后面连接的电路断开

数值的表示

number	stored value	comment
5'b11010	11010	
5'b11_010	11010	_ ignored
5'o32	11010	
5'h1a	11010	
5'd26	11010	
5'b0	00000	0 extended
5'b1	00001	0 extended
5'bz	zzzzz	z extended
5'bx	xxxxx	x extended
5'bx01	xxx01	x extended
-5'b00001	11111	2's complement of 00001
'b11010	0000000000000000000000000000000011010	extended to 32 bits
'hee	0000000000000000000000000000000011101110	extended to 32 bits
1	0000000000000000000000000000000000000001	extended to 32 bits
-1	11	extended to 32 bits

常用运算符

operation	symbol	description
Arithmetic	+	addition
	-	subtraction
	*	multiplication
	/	division
	%	modulus
	**	exponentiation
Shift	>>	logical right shift
	<<	logical left shift
	>>>	arithmetic right shift
	<<<	arithmetic left shift
Relational	>	greater than
	<	less than
	>=	greater than or equal to
	<=	less than or equal to
Reduction	&	reduction and
		reduction or
	^	reduction xor

operation	symbol	description
Equality	==	equality
	!=	inequality
	===	case equality
	!==	case inequality
Logical	!	logical negation
	&&	logical and
		logical or
Bitwise	~	bitwise negation
	&	bitwise and
		bitwise or
	^	bitwise xor
Concatenation	{ }	concatenation
	{ { } }	replication
Conditional	? :	conditional

bitwise and Logical operation examples

a b		$a \& b$	$a b$	$a \&\& b$	$a b$
0	1	0	1	0 (false)	1 (true)
000	000	000	000	0 (false)	0 (false)
000	001	000	001	0 (false)	1 (true)
011	001	001	011	1 (true)	1 (true)