修订一：P61 程序 3.38

```
initial  clock=0;
always  #5 clock=~clock;          //生成时钟信号
initial
begin
  d=1;
  reset=1;
  #12 reset=0;
  #11 reset=1;                    //仿真信号产生
  #17 $stop;                      //仿真控制
end
```

修订二：P90

（6）Implementation（实现）

1）Implementation Settings（实现设置）。

2）Run Implementation（运行实现）。

3）Open Implementation Design（打开实现后的设计）。

修订三：P101 程序 4.2

```
set_property IOSTANDARD LVCMOS18 [get_ports led]
set_property IOSTANDARD LVCMOS18 [get_ports {sw[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {sw[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {sw[0]}]
```

修订四：P104 程序 4.3

```
module led_tb();
reg [2:0]sw;

wire led;

led uut(sw,led);
```

修订五：P149 程序 4.8

```
set_property IOSTANDARD LVCMOS18 [get_ports {led[15]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[14]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[13]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[12]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[11]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[10]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[9]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[8]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports clk]
```

修订六：P171

1．在 Objects 窗口中查看

如图 5.28 所示，Objects 窗口中显示了信号变量的位数及数值。如果默认未显示 Objects 窗口的话，可以在 View 菜单栏中把它调出来。

修订七：P192

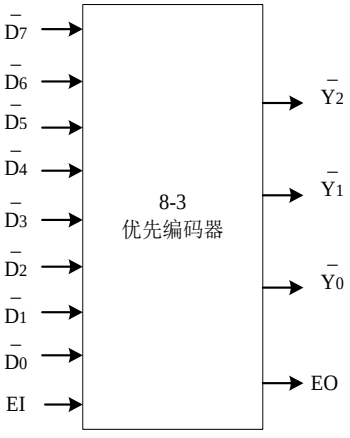
（2）Verilog 代码描述：

```
module extend #(parameter WIDTH = 16)(
    input [WIDTH-1:0] a,
    input sext, //sext 有效是高电平为符号扩展，否则为 0 扩展
    output [31:0] b
);
assign b=sext? {{(32-WIDTH){a[WIDTH-1]}},a} : {(32-WIDTH)'b0,a};
endmodule
```

（3）TestBench 代码描述：

```
`timescale 1ns/1ns
 module extend_tb;
    reg [15:0] a,sext;
    wire [31:0] b;
// Instantiate the Unit Under Test (UUT)
    extend uut (.a(a),.sext(sext),.b(b));
    initial
```

修订八：P199 图 6.14



$\overline{D}_0$	$\overline{D}_1$	$\overline{D}_2$	$\overline{D}_3$	$\overline{D}_4$	$\overline{D}_5$	$\overline{D}_6$	$\overline{D}_7$	$\overline{Y}_2\overline{Y}_1\overline{Y}_0$			EI	EO
×	×	×	×	×	×	×	×	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	0	0
×	×	×	×	×	×	×	0	0	0	0	0	1
×	×	×	×	×	×	0	1	0	0	1	0	1
×	×	×	×	×	0	1	1	0	1	0	0	1
×	×	×	×	0	1	1	1	0	1	1	0	1
×	×	×	0	1	1	1	1	1	0	0	0	1
×	×	0	1	1	1	1	1	1	0	1	0	1
×	0	1	1	1	1	1	1	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	0	1

修订九：P206 表 6.16

表 6.16　8 位加法器 XDC 文件配置

变量	iData_a [0]~[7]	iData_b [0]~[7]	iC	oData [0]~[7]	oData_C
N4 板上的管脚	SW0~7 (J15、L16、M13、R15、R17、T18、U18、R13)	SW8~15 (T8、U8、R16、T13、H6、U12、U11、V10)	BNTR (M17)	LD0~7 (H17、K15、J13、N14、R18、V17、U17、U16)	LD8 (V16)

修订十：P212

4. 实验步骤

(1) 用 logicsimLogisim 画出同步模 8 计数器电路原理图，验证逻辑。

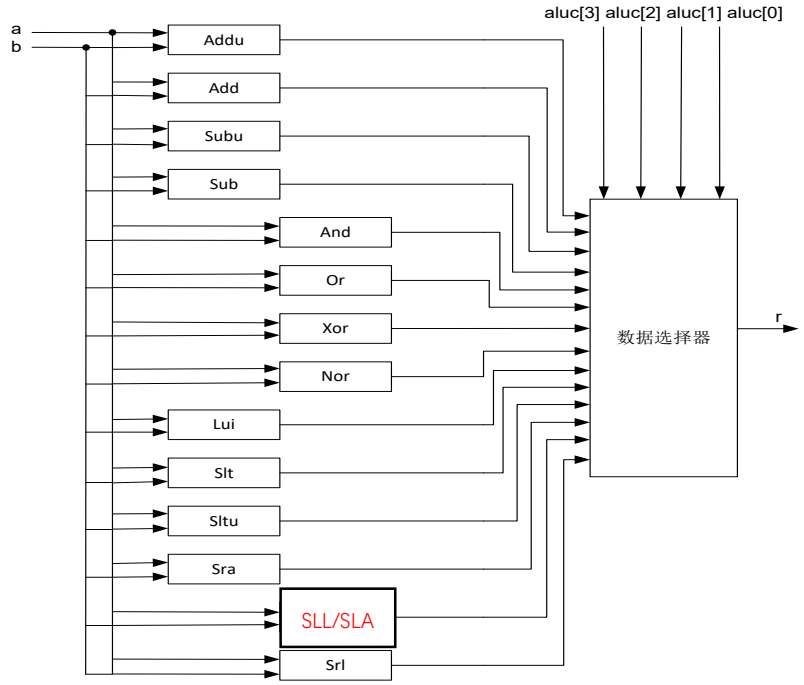
(2) 新建 Vivado 工程，编写各个模块。

(3) 用 ModelSim 仿真测试各模块。

(4) 配置 XDC 文件，综合下板，并观察实验现象。

(5) 按照要求书写实验报告。

修订十一：P216 图 6.30



修订十二：P217 表 6.22

SLT	r=(a<b)?1:0 有符号	1	0	1	1
SLTU	r=(a<b)?1:0 无符号	1	0	1	0
SRA	r=b>>>a	1	1	0	0
SLL/SLA	r=b<<a	1	1	1	X
SRL	r=b>>a	1	1	0	1

修订十三：P232 表 7.4

srlv	000000	rs	rt	rd	00000	000110	srlv $1,$2,$3	$1=$2>>$3	rd <- rt >> rs ；(logical)其中 rs＝$3，rt=$2, rd=$1
srav	000000	rs	rt	rd	00000	000111	srav $1,$2,$3	$1=$2>>$3	rd <- rt >> rs ；(arithmetic) 注意符号位保留 其中 rs＝$3，rt=$2, rd=$1
jr	000000	rs	00000	00000	00000	001000	jr $31	goto $31	PC <- rs
I-type	op	rs	rt	immediate					
addi	001000	rs	rt	immediate			addi $1,$2,100	$1=$2+100	rt <- rs + (sign-extend)immediate ； 其中 rt=$1,rs=$2
addiu	001001	rs	rt	immediate			addiu $1,$2,100	$1=$2+100	rt <- rs + (sign-extend)immediate ； 其中 rt=$1,rs=$2

修订十四：P234

ADDIU:

格式: ADDIU rt, rs, immediate

目的: 使 32 位数据与一个立即数相加

描述: rt ← rs + immediate

一个 16 位无符号的立即数与通用寄存器 rs 中的 32 位数相加产生一个 32 位的数存入目标寄存器 rt。

在任何情况下都不会有溢出的异常。

修订十五：P256

Memory)、寄存器堆（Regfile)、ALU、带符号扩展模块 Ext18、加法器 ADD，完成转移地址的计算。如表 7.18 所示。

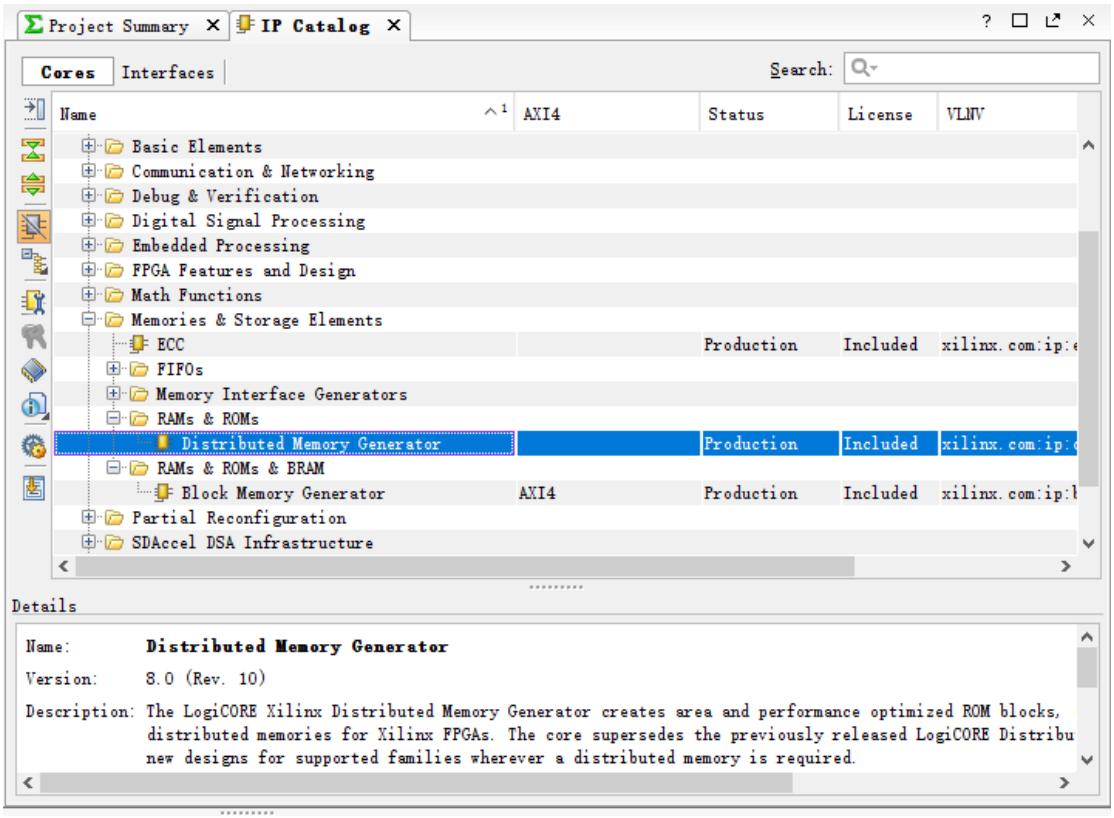
修订十六：P261 表 7.22

表 7.22 ALU 功能控制

	ALUC2	ALUC1	ALUC0	备注
Add	0	0	0	ALU 完成“加”
Sub	0	0	1	ALU 完成“减”
Or	0	1	0	ALU 完成“或”
Sll	0	1	1	ALU 完成“左移”

修订十七：P299 图 7.53

原图替换为：



修订十八：P306 接口定义

```
module MULTU(
    input clk,          //乘法器时钟信号
    input reset,        //复位信号，高电平有效
    input [31:0] a,     //输入数 a(被乘数)
    input [31:0] b,     //输入数 b（乘数）
    output [63:0] z     //乘积输出 z
);
```

修订十九：P307 接口定义

```
module MULT(
    input clk,          //乘法器时钟信号
    input reset,        //复位信号，高电平有效
    input [31:0] a,     //输入数 a(被乘数)
    input [31:0] b,     //输入数 b（乘数）
    output [63:0] z     //乘积输出 z
);
```

修订二十：P308

```
reg [15:0] add4_5;
reg [15:0] add6_7;
reg [15:0] add0t1_2t3;
reg [15:0] add4t5_6t7;
reg [15:0] add0t3_4t7;

 always @(posedge clk or posedge reset)
 begin
     // reset 置零
   if(reset) begin
       temp <= 0;
       stored0 <= 0;
       stored1 <= 0;
       stored2 <= 0;
       stored3 <= 0;
```

修订二十一：P310 接口定义

```
module DIVU(
    input [31:0]dividend,        //被除数
    input [31:0]divisor,         //除数
    input start,                 //启动除法运算
    input clock,
    input reset,                 //高电平有效
    output [31:0]q,              //商
    output [31:0]r,              //余数
    output busy                  //除法器忙标志位
    );
```

```
module DIV(
    input [31:0]dividend,        //被除数
    input [31:0]divisor,         //除数
    input start,                 //启动除法运算
    input clock,
    input reset,                 //高电平有效
    output [31:0]q,              //商
    output [31:0]r,              //余数
    output busy                  //除法器忙标志位
    );
```