北京大学
PEKING UNIVERSITY

FPGA设计

授课教师：曹 健

课时安排

北京大学 PEKING UNIVERSITY

课次	日期	知 识 点	课 程 内 容
1	0221	Verilog语法	讲解Verilog语法，帮助学生掌握Verilog语法结构和八股使学生初步**掌握Verilog编程语言**。
2	0228	Genesys2使用（含测试）	带领学生通过FPGA自带的逻辑分析仪，寻找Verilog代码中的逻辑错误，使学生学会FPGA的使用和**测试方法**。
3	0307	Uart设计	带领学生设计一个Uart收发模块，实现Genesys2与PC端通过串口通信。
4	0314	CPU设计I	设计CPU完整代码，实现运算逻辑运算。
5	0321	CPU设计II	改进CPU代码，自动实现流水灯。 改进CPU代码，从7指令升级为RISCV64IM指令集架构。

课时安排

北京大学 PEKING UNIVERSITY

课次	日期	知 识 点	课 程 内 容
6	0328	CPU设计III	改进CPU代码，实现5级流水。
7	0404	CPU设计IV	改进CPU代码，实现乘除法模块设计。
8	0411	目标检测加速器I	整体架构设计,YOLOv3算法，软硬件划分，PS/PL通信。
9	0418	目标检测加速器II	卷积模块设计（PL），循环拆分，数据复用，硬件设计。
10	0425	目标检测加速器III	CORDIC后处理模块（PL），算法推导，硬件设计。
11	0509	目标检测加速器IV	小组汇报：加速器设计及改进 （抛砖引玉）

本节提纲

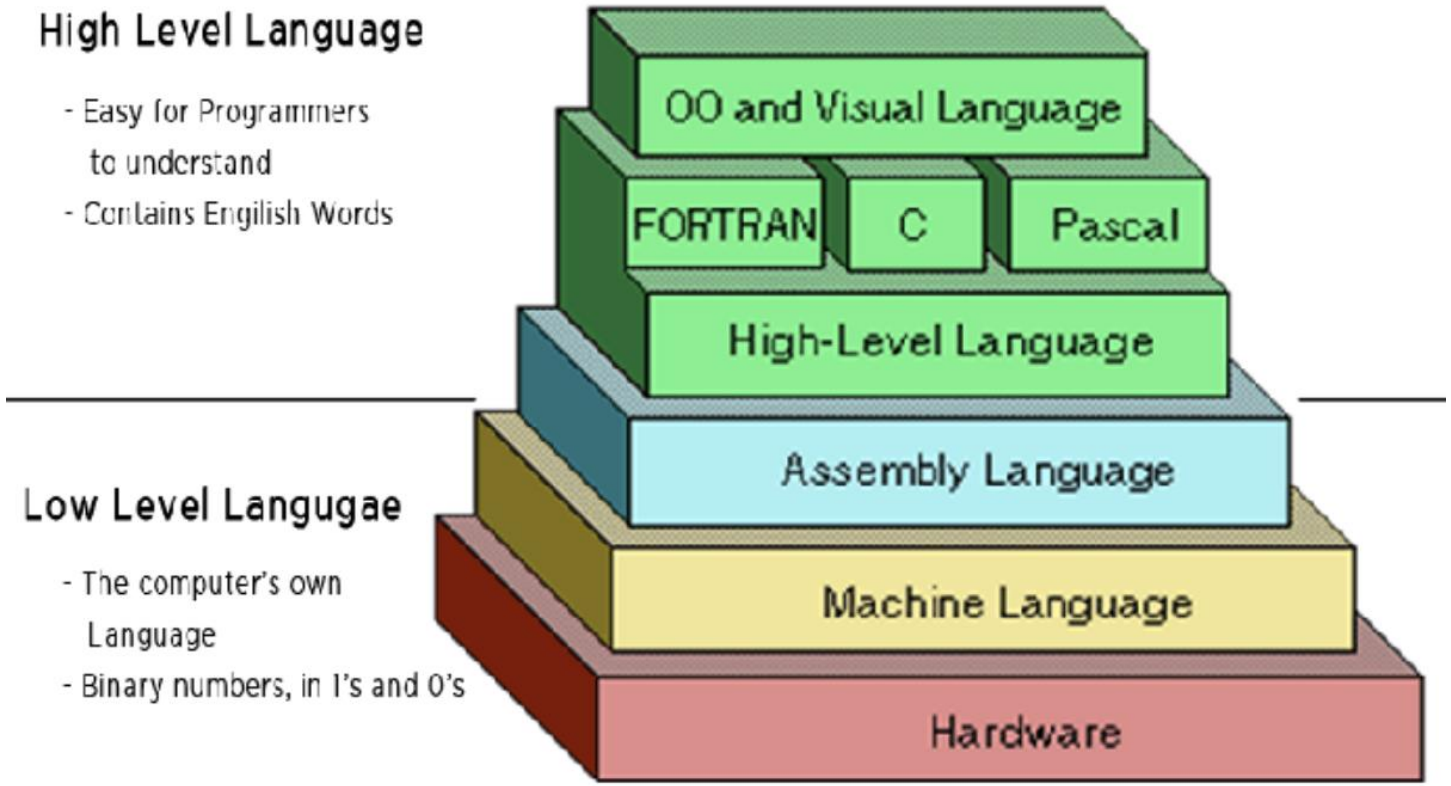
北京大学 PEKING UNIVERSITY

1.设计一个四指令简单CPU

2.使用ILA在线观察内部信号，调试程序

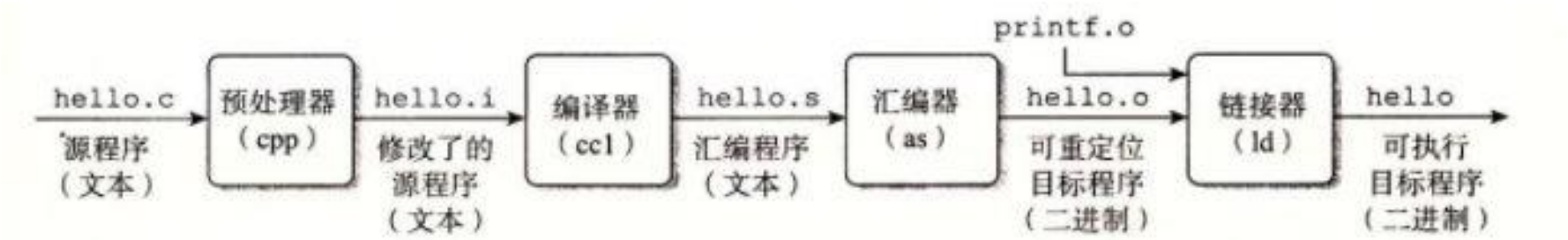
CPU的沟通语言：指令

什么是指令，如何理解指令?



- CPU工作方式：执行按照一定顺序排列的指令
- 计算机程序会被翻译成机器码，形成一组计算机能识别和执行的指令

如何获取指令?



将高层的C/C++语言编写的程序转换成为处理器能够执行的二进制代码的过程，一般包括四个步骤：

- 预处理（Preprocessing）
- 编译（Compilation）
- 汇编（Assembly）
- 链接（Linking）

获取指令的工具：GCC工具链

GCC：GUN Compiler Collection，GCC实质上不是一个单独的程序，而是多个程序的集合，因此通常称为GCC工具链。工具链软件包括GCC、C运行库、Binutils、GDB等。

- GCC
  GCC是编译工具，既支持本地编译，也支持交叉编译
- C运行库
  定义了C标准库函数原型
- Binutils
  一组用来开发和调试的二进制程序处理工具
- GDB
  GDB（GNU Project Debugger）是调试工具，可以用于对程序进行调试

指令的类型有哪些?

北京大学 PEKING UNIVERSITY

数据处理和内存操作指令

- 寄存器间数据搬运
- 寄存器与内存间的数据搬运
- 给寄存器赋常值

算术与逻辑运算指令

- 两个寄存器中数据的加减乘除运算
- 按位操作的与、或、异或等运算

分支指令

- 跳转至另一处继续执行

CPU的组成部分

北京大学 PEKING UNIVERSITY

取指/译码等 调度单元	乘法、除法、访存等 执行单元	寄存器、RAM等 存储单元
• 从内存中取出指令 • 识别指令类型、源和目的操作数 • 使能执行单元	• 执行乘法、除法、逻辑运算等 • 执行load/store等访存指令	

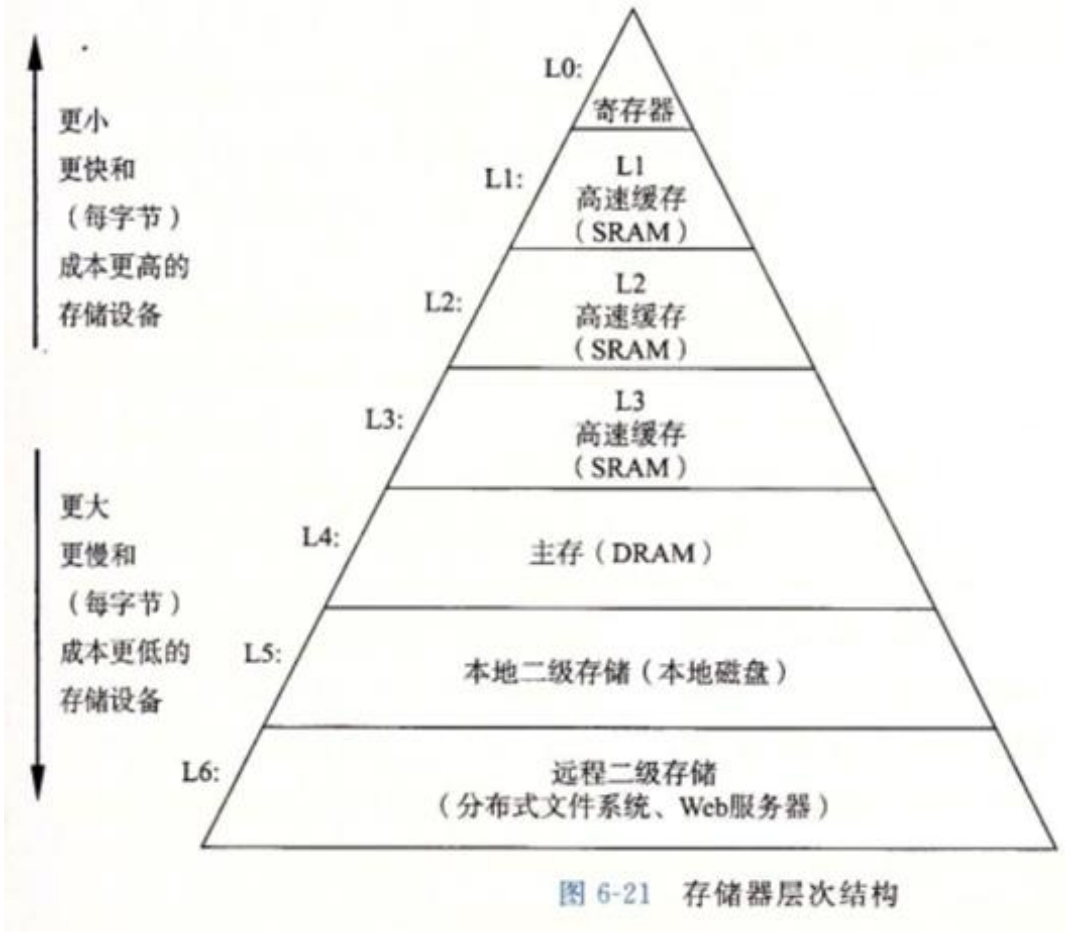


图 6-21 存储器层次结构

本次课程：实现一个仅包含寄存器的4指令简单CPU

不考虑RAM、不考虑load/store，后续课程会逐步增加。

设计简单CPU

北京大学 PEKING UNIVERSITY

- 本次课时目标：完成一个简单的四指令CPU

	mv	mvi	add	sub
编码	00	01	10	11
汇编指令	mv Rx, Ry	mvi Rx, #D	add Rx, Ry	sub Rx, Ry
功能	Rx <- [Ry]	Rx <- D	Rx <- [Rx] + [Ry]	Rx <- [Rx] – [Ry]

指令格式声明：

II XX YY， II 表示指令, XX 表示Rx寄存器, YY 表示Ry寄存器

mvi指令后跟随输入数据是立即数D

设计简单CPU

北京大学 PEKING UNIVERSITY

- 本次课时目标：完成一个简单的四指令CPU

基本要求：

如何实现？

- 开发板上开关状态模拟输入指令
- 实现mv mvi add sub四条指令功能
- 使用LED灯指示数据总线状态；ILA分析内部信号实现在线调试

设计简单CPU

模块化思维：

- 根据功能设计输入输出端口
- 布局规划内部模块

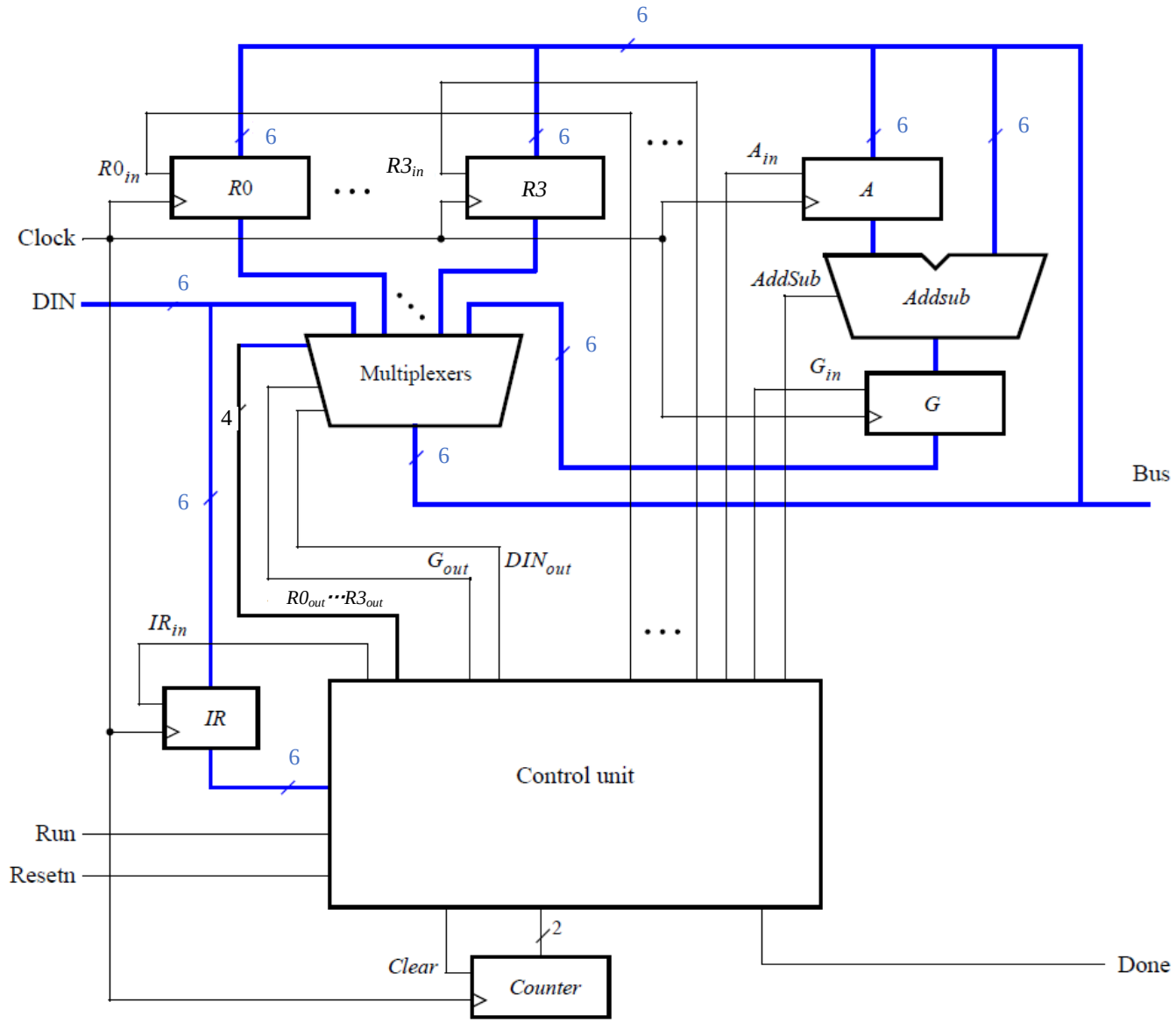
port：

DIN, Clock, Resetn, Run, Done, BusWires

block：

ALU(add/sub), controller(dec2to4, upcount), mux, regn

设计简单CPU



Mux控制信号名称	作用
Rout[3:0](独热码)	Mux选择R0~R3之一并送至Bus
Gout	Mux选择G并送至Bus
DINout	Mux选择DIN并送至Bus

IR控制信号名称	作用
IRin	Ins Reg的写入使能

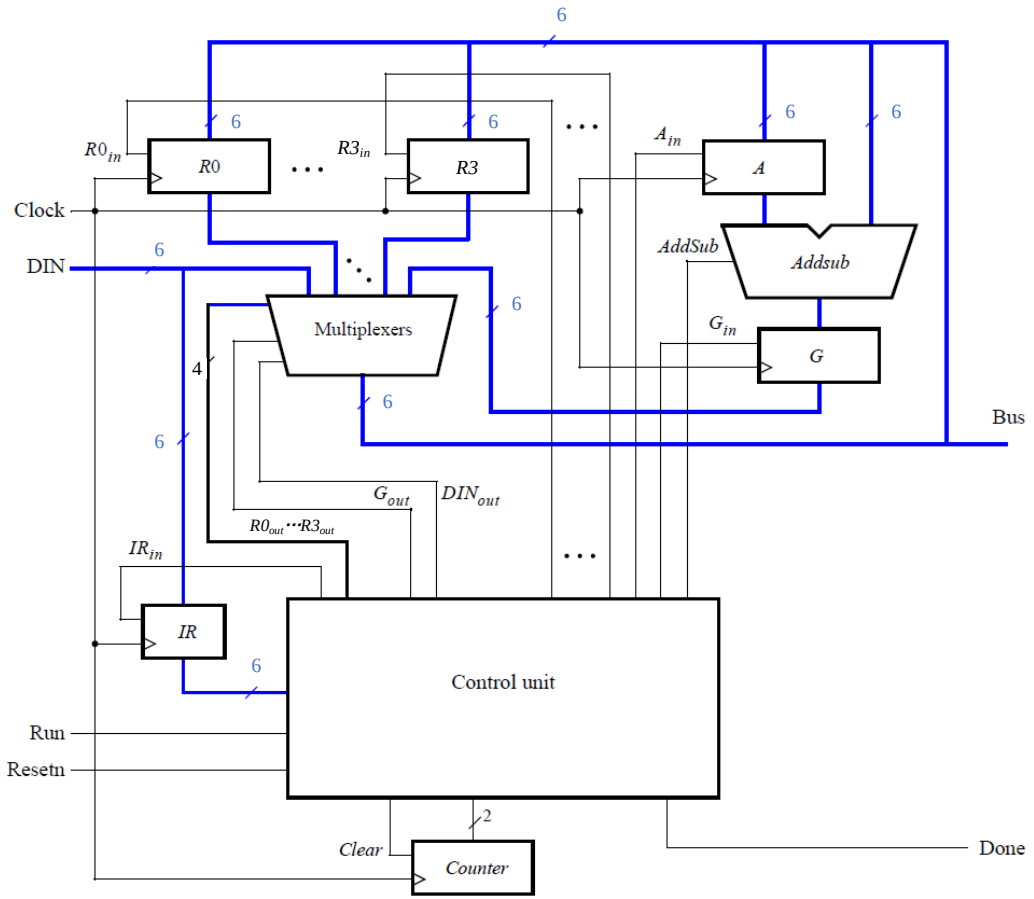
AddSub控制信号	作用
AddSub	1为加法，0为减法

寄存器控制信号	作用
Ain	寄存器A的写入使能
Gin	寄存器G的写入使能
Rin[3:0] (独热码)	寄存器R[_]的写入使能

控制时序

用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新

指令 \ 时间	T0	T1	T2	T3
(mv):I_0	IR_{in}	RY_{out}, RX_{in}, Done	—	—
(mvi):I_1	IR_{in}	DIN_{out}, RX_{in}, Done	—	—
(add):I_2	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}	G_{out}, RX_{in}, Done
(sub):I_3	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}, Addsub	G_{out}, RX_{in}, Done



Rout表示把某个寄存器中的数据放出到汇流台

Rin表示把汇流台中的数据放入指定的R寄存器

控制时序

用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新

时间 / 指令	T0	T1	T2	T3
(mv):I_0	IR_{in}	RY_{out}, RX_{in}, Done	—	—
(mvi):I_1	IR_{in}	DIN_{out}, RX_{in}, Done	—	—
(add):I_2	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}	G_{out}, RX_{in}, Done
(sub):I_3	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}, Addsub	G_{out}, RX_{in}, Done

mvi R3,3 01 11 ..... (SW5-SW0) clk1个T0

……11 clk1个T1

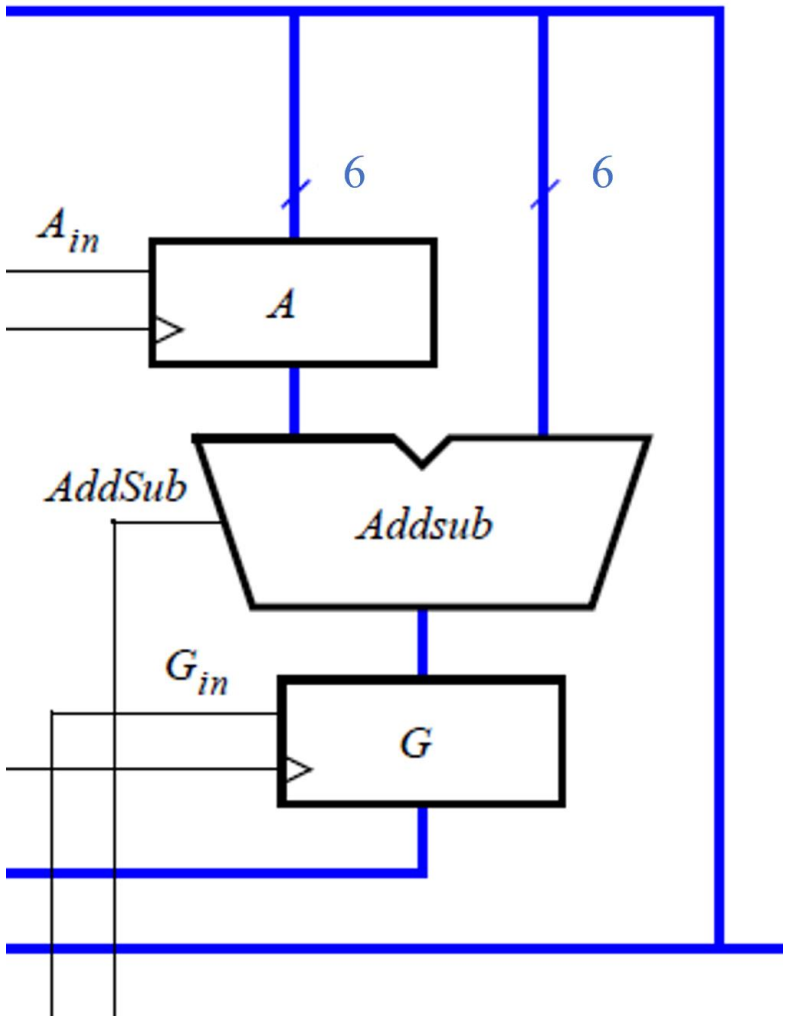
mvi R0,7 01 00 .... clk1个T0

……111 clk1个T1

Add R3,R0 10 11 00… clk3个T0 T1 T2 出

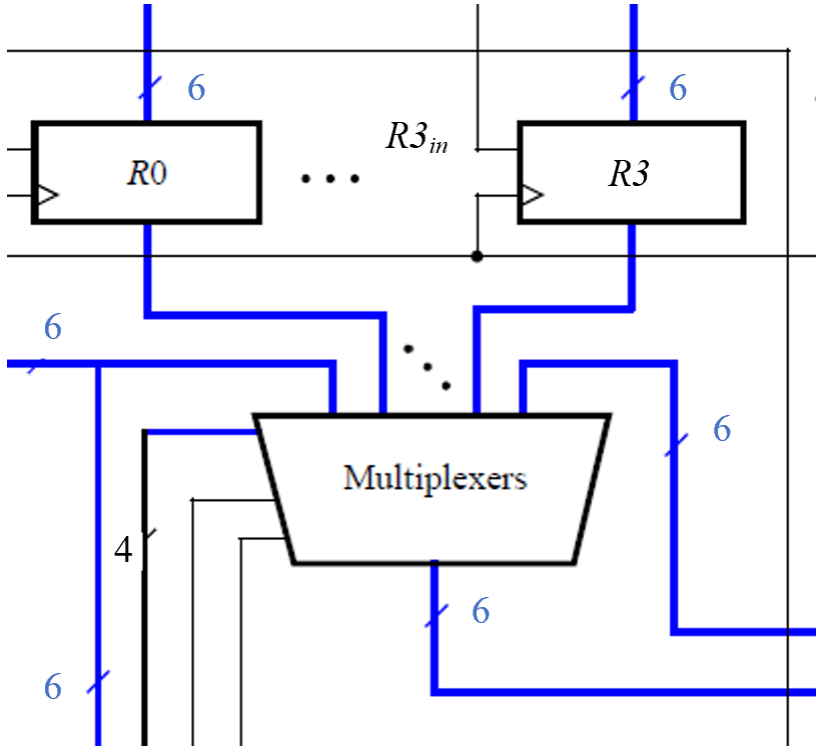
现到总线, 1个clkT3进入R3

AddSub




```
1   //Name: Add Sub
2   //Function:
3   //      AddSub = 0 Add
4   //      AddSub = 1 Sub
5   //Author: caojian
6
7   module addsub
8   #(parameter DATAWIDTH = 6)
9   (
10      input                       AddSub,
11      input      [DATAWIDTH-1:0] A,
12      input      [DATAWIDTH-1:0] BusWires,
13
14      output reg [DATAWIDTH-1:0] Sum
15  );
16
17      always@(*) begin
18          if(!AddSub)
19              Sum = A + BusWires;
20          else
21              Sum = A - BusWires;
22      end
23
24  endmodule //end of addsub
```

BusMux

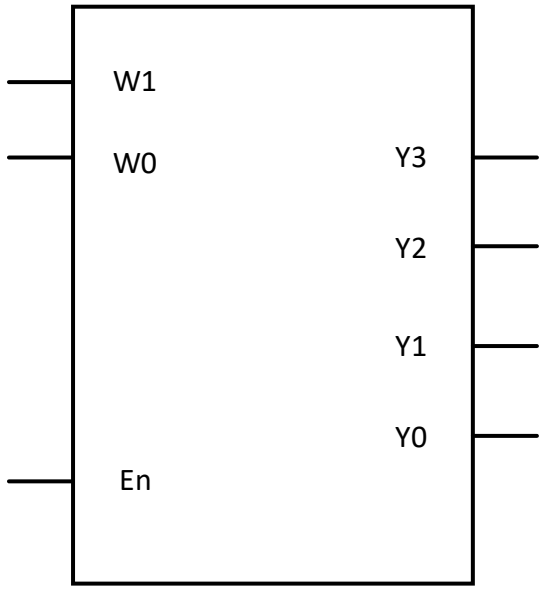



```
module busmux
#(parameter REG_NUM = 4, DATAWIDTH = 6)
(
    input       [REG_NUM-1:0]   Rout,
    input                       Gout, DINout,
    input       [DATAWIDTH-1:0] R0, R1, R2, R3, G, DIN,
    output reg  [DATAWIDTH-1:0] BusWires
);
    wire[REG_NUM+1:0] Sel;

    assign Sel = {Rout, Gout, DINout};
    always@(*)
    begin
        if(Sel == 'b100_000)
            BusWires = R0;
        else if(Sel == 'b010_000)
            BusWires = R1;
        else if(Sel == 'b001_000)
            BusWires = R2;
        else if(Sel == 'b000_100)
            BusWires = R3;
        else if(Sel == 'b000_010)
            BusWires = G;
        else BusWires = DIN;
    end

endmodule //end of busmux
```

Decoder

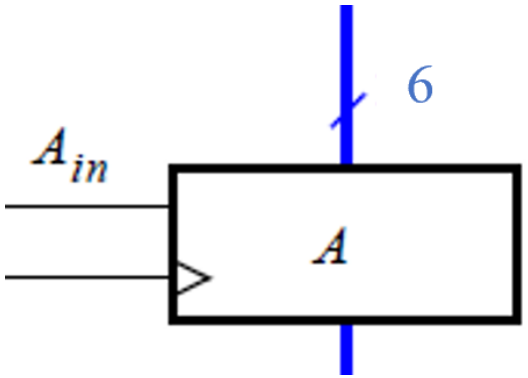



```
module decoder
#(parameter CODEWID = 2, OUTWID=4)
(
   input       [CODEWID-1:0] W,
   input                     En,
   output reg [OUTWID-1:0]  Y
);

always @ (*)
 begin
  if(En == 1)
     case(W)
       2'b00:   Y = 'b1000;
       2'b01:   Y = 'b0100;
       2'b10:   Y = 'b0010;
       2'b11:   Y = 'b0001;
     endcase
  else
     Y = 'b0000;
 end

endmodule
```

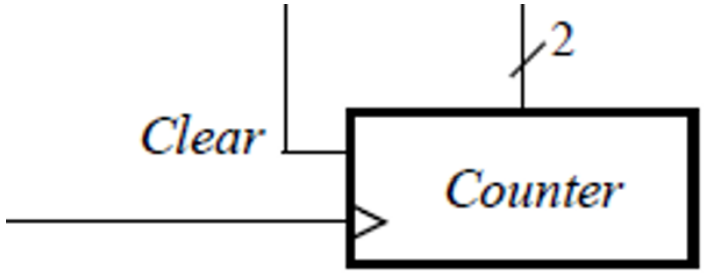
Regn




```
module regn(R,Rin,Clock,Q);
    parameter n = 6;
    input[n-1:0] R;
    input Rin,Clock;
    output[n-1:0] Q;
    reg [n-1:0] Q;

always @(posedge Clock)
    if(Rin)
  Q <= R;
endmodule
```

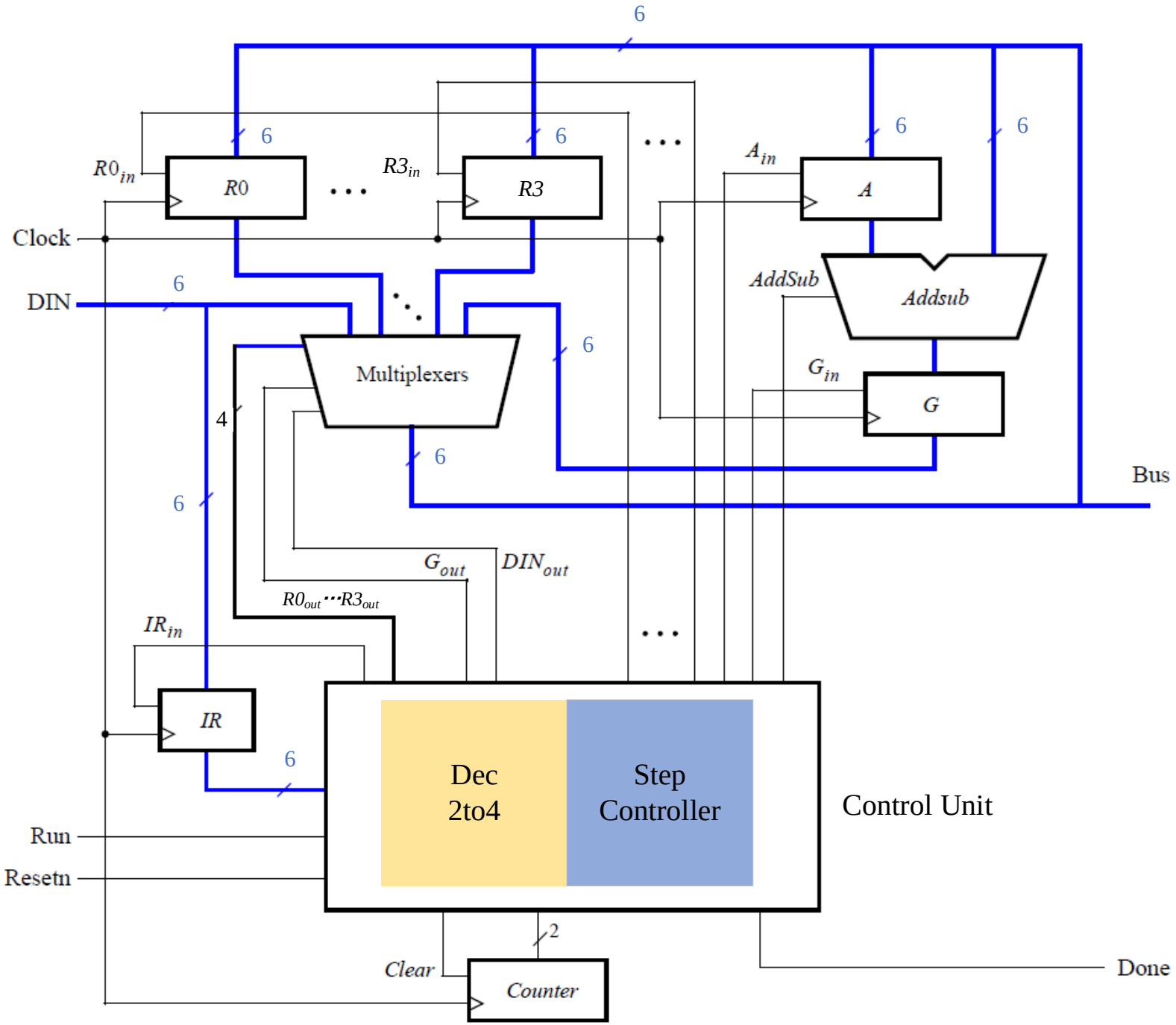
Upcount



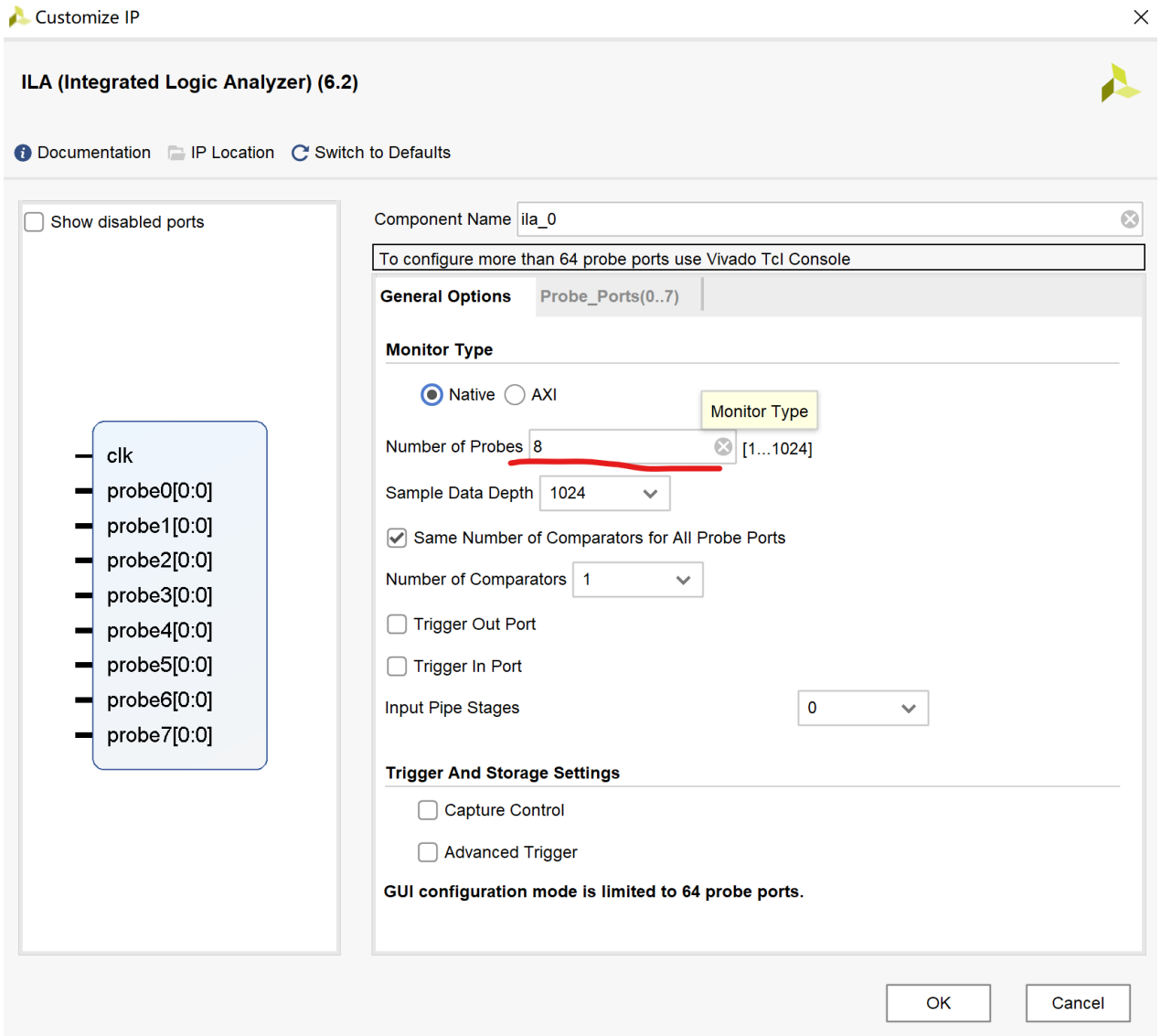

```
module upcount(Clear,Clock,Q);
    input Clear,Clock;
    output [1:0] Q;
    reg [1:0] Q;

    always @(posedge Clock)
        if (Clear)
            Q <= 2'b0;
        else
            Q <= Q+1'b1;
endmodule
```

系统框图


6
R0 in
R0
R3 in
R3
A in
A
Clock
DIN
AddSub
Addsub
Multiplexers
G in
G
4
Bus
G out
DIN out
R0 out ··· R3 out
IR in
IR
Dec
2to4
Step
Controller
Control Unit
Run
Resetn
2
Clear
Counter
Done

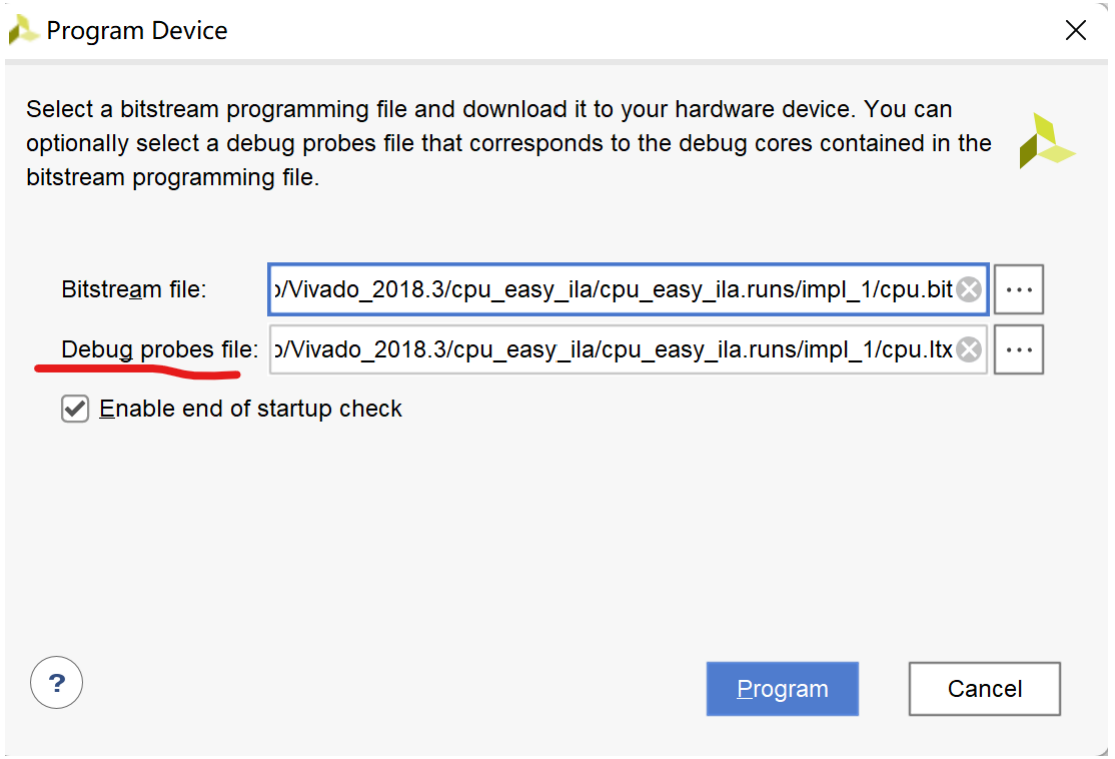
实验部分


Customize IP
ILA (Integrated Logic Analyzer) (6.2)
Documentation
IP Location
Switch to Defaults
Show disabled ports
clk
probe0[0:0]
probe1[0:0]
probe2[0:0]
probe3[0:0]
probe4[0:0]
probe5[0:0]
probe6[0:0]
probe7[0:0]
Component Name ila_0
To configure more than 64 probe ports use Vivado Tcl Console
General Options
Probe_Ports(0..7)
Monitor Type
Native
AXI
Monitor Type
Number of Probes 8 [1...1024]
Sample Data Depth 1024
Same Number of Comparators for All Probe Ports
Number of Comparators 1
Trigger Out Port
Trigger In Port
Input Pipe Stages 0
Trigger And Storage Settings
Capture Control
Advanced Trigger
GUI configuration mode is limited to 64 probe ports.
OK
Cancel

实验部分

北京大学 PEKING UNIVERSITY

```
ila_0 your_instance_name (
      .clk(CLOCK_50), // input wire clk
      .probe0(Clock), // input wire [0:0]  probe0
      .probe1(Resetn), // input wire [0:0]  probe1
      .probe2(Run), // input wire [0:0]  probe2
      .probe3(BusWires), // input wire [5:0]  probe3
      .probe4(R0), // input wire [5:0]  probe4
      .probe5(R3), // input wire [5:0]  probe5
      .probe6(Rin), // input wire [3:0]  probe6
      .probe7(Rout) // input wire [3:0]  probe7
 );
```



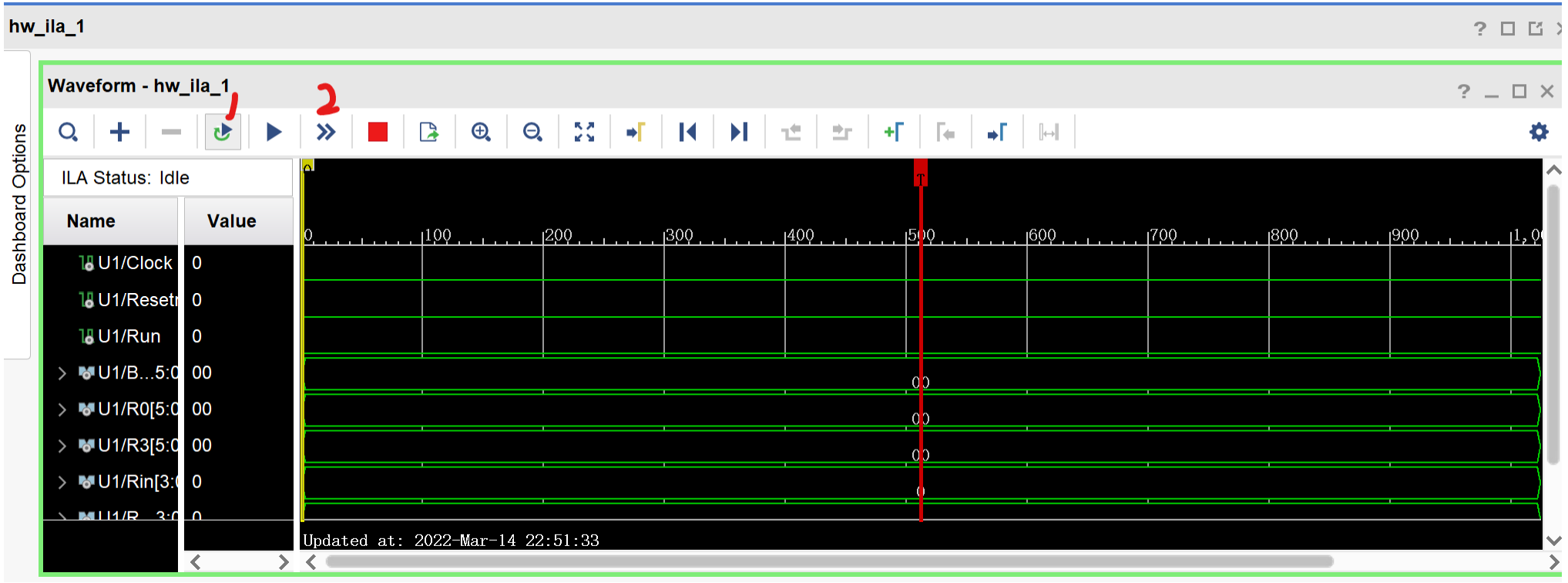
参考测试方法

用组合电路准备 clk触发前瞬间更新 → 用组合电路准备 clk触发前瞬间更新 → 用组合电路准备 clk触发前瞬间更新 → 用组合电路准备 clk触发前瞬间更新

时间 / 指令	T0	T1	T2	T3
(mv):I_0	IR_{in}	RY_{out}, RX_{in}, Done	—	—
(mvi):I_1	IR_{in}	DIN_{out}, RX_{in}, Done	—	—
(add):I_2	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}	G_{out}, RX_{in}, Done
(sub):I_3	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}, Addsub	G_{out}, RX_{in}, Done

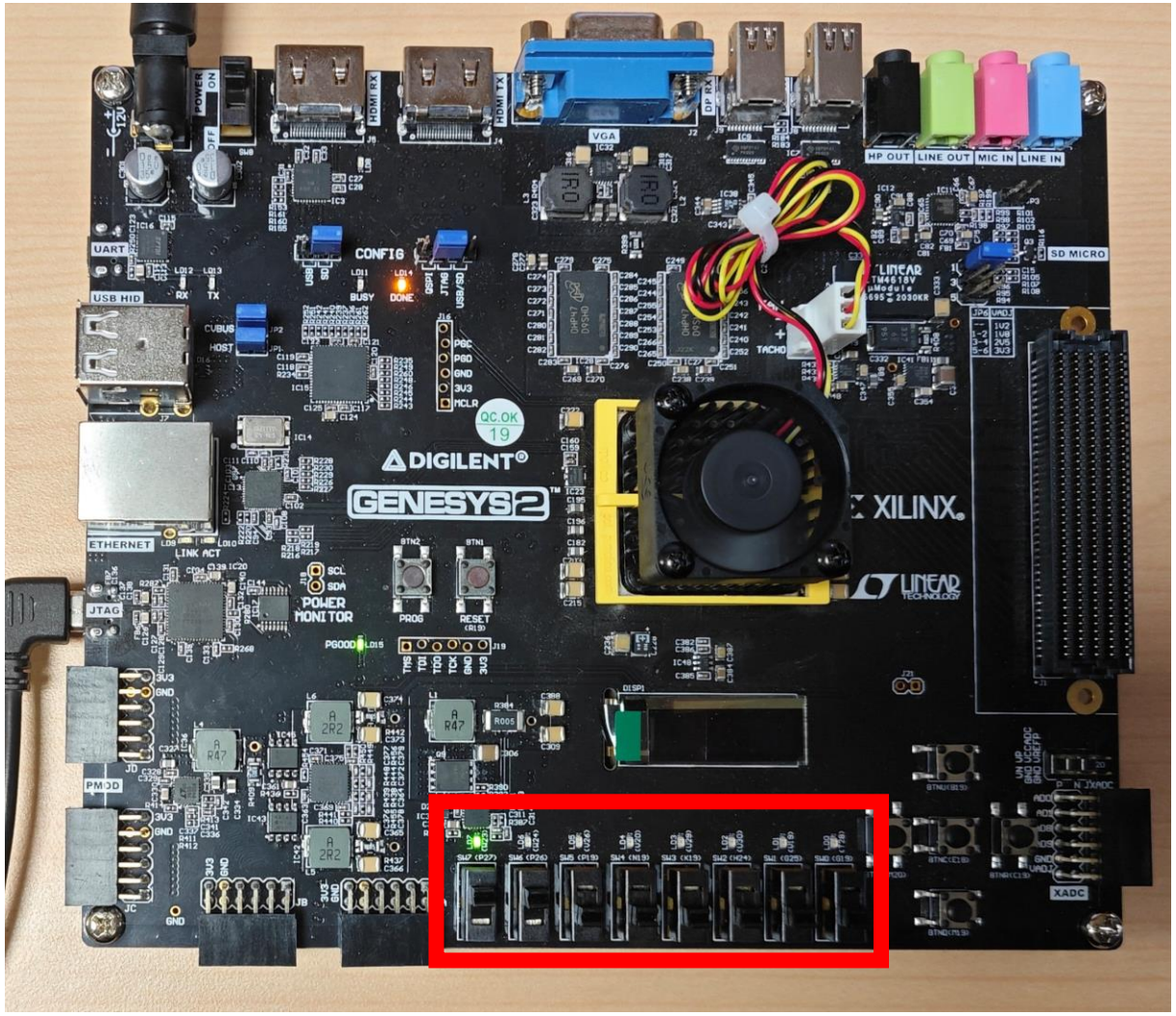
1. 装载立即数指令MVI测试： MVI R3, 0b110110 执行结果：R3 <- 0b110110。CPU执行过程：
   => 第1个cycle，SW输入：01_11_XX（Instruction=mvi, X_reg_ID=3, Y_reg_ID=无所谓）
   => 第2个cycle，SW输入：110110 （输入的6位立即数）
2. 移动寄存器指令MV测试：MV R2, R3 执行结果：R2 <- R3 。CPU执行过程：
   => 第一个cycle，SW输入：00_10_11（Instruction=mv, X_reg_ID=2, Y_reg_ID=3）
   => 第二个cylce，无需输入，因此SW无效。
3. 加法指令ADD测试：ADD R2, R3 执行结果：R2 <- (R2+R3) 。 CPU执行过程：
   => 第一个cycle，SW输入：10_10_11（Instruction=add, X_reg_ID=2, Y_reg_ID=3）
   => 第二个cycle，第三个cycle，第四个cycle，均无需输入，SW无效。
4. 减法指令SUB测试：SUB R2, R3 执行结果：R2 <- (R2-R3) 。 CPU执行过程：
   => 第一个cycle，SW输入：11_10_11（Instruction=sub, X_reg_ID=2, Y_reg_ID=3）
   => 第二个cycle，第三个cycle，第四个cycle，均无需输入，SW无效。

实验部分


hw_ila_1
Waveform - hw_ila_1
1
2
Dashboard Options
ILA Status: Idle
Name
Value
U1/Clock
U1/Reset
U1/Run
U1/B...5:0
U1/R0[5:0
U1/R3[5:0
U1/Rin[3:
0
00
100
200
300
400
500
600
700
800
900
Updated at: 2022-Mar-14 22:51:33

课堂实验

	7	6	5	4	3	2	1	0
LED	Done		BusWire[5:0]					
SW	Run	Rstn	Din[5:0]					



请每个组完成：

1. 梳理CPU原理，填充proc.c中的剩余代码；
2. 上板部署，并实现四种指令的演示；
3. 思考：done信号亮灯时间正确吗？