实验 10-多周期处理器实验

一、 实验准备

前面我们已经完成的了单周期实验，搭建了完整的单周期处理器 datapath。这节课我们开始在单周期实验的基础上，将单周期处理器扩展为多周期处理器。在理论课上，我们讲解如何将单周期改造成多周期处理器，主要需要改动 2 处:(1) 在 datapath 中每个 stage（即取指、译码、执行、访存、回写 5 个阶段）中加入寄存器 （2）需要将 controller 由原来的组合逻辑变成状态机；通过前面的实验，单周期 CPU 的设计大家已经完成，CPU datapath 各个通路基本模块均已具备。本次实验实验可以依旧复用上节课的模块，完成本次实验。在进行本次实验前，你需要具备以下基础能力：

1.熟悉 Vivado 的仿真功能 (行为仿真)

2.熟悉课堂上讲的状态机的设计流程（用于实现控制器）

3.理解多周期处理器的处理流程，以及控制器的设计流程

二、 实验目的

1. 掌握多周期 CPU 数据通路及其设计方法；
2. 掌握状态机的设计方法并实现多周期 CPU 控制器；
3. 掌握多周期 CPU 的实现方法，代码实现方法；
4. 掌握测试多周期 CPU 的方法。

三、 实验设备

PC 机一台，Basys3 开发板，Xilinx Vivado 开发套件。

四、 实验任务

4.1 实验要求

先阅读实验原理部分，实现下列模块，并按照下列要求完成实验：

- Datapath，基于单周期处理器 CPU 的数据通路代码进行改造，在各个需要插入寄存器组 IR, MDR, A,B, ALUOut（系统框图**红色部分**），注意其中 IR 和 PC 寄存器都需要进行使能控制。用于控制在当前指令没有结束的时候，禁止下一条指令导入。
- 在多周期处理器里面，由于指令存储器 inst_mem(Single Port Rom)和数据存储器 data_mem(Single Port Ram)是在不同周期里面使用的，因此可以分时使用将两个存储器合并；同上次实验一样，使用 BlockMemory Generator IP 构造指令，注意考虑 PC 地址位数统一。 (参考前面的实验)
- 参照实验原理，将上述模块依指令执行顺序连接。
- 实验给出仿真程序，最终以仿真输出结果判断是否成功实现要求指令。仿真具体参照，参照实验 1 基础操作，以及《Basys3 入门指导手册》。

4.2 实验步骤

1. 根据控制器在每个周期的输出控制信号，根据状态机设计控制器；
2. 从上一次实验中，导入各个模块，并按照如图 1 进行改写 datapath；
3. 根据前面的存储器实验使用 Block Memory，生成统一的 memory;

4. 参考实验原理，连接各模块；
5. 导入顶层文件及仿真文件，运行仿真；

4.2 实验模块结构

依据单周期实验原来代码修改，修改 flopr.v 文件见附录；

五、 实验原理

多周期 CPU 指的是将整个 CPU 的执行过程分成几个阶段，每个阶段用一个时钟去完成，然后开始下一条指令的执行，而每种指令执行时所用的时钟数不尽相同，这就是所谓的多周期 CPU。CPU 在处理指令时，一般需要经过以下几个阶段：

(1) 取指令(IF)：根据程序计数器 pc 中的指令地址，从存储器中取出一条指令，同时，pc 根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入 pc，当然得到的“地址”需要做些变换才送入 pc。

(2) 指令译码(ID)：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的各种操作。

(3) 指令执行(EXE)：根据指令译码得到的操作控制信号，具体地执行指令动作，然后转移到结果写回状态。

(4) 存储器访问(MEM)：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得到数据地址单元中的数据。

(5) 结果写回(WB)：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。

实验中就按照这五个阶段进行设计，这样一条指令的执行最长需要五个(小)时钟周期才能完成，但具体情况怎样？要根据该条指令的情况而定，有些指令不需要五个时钟周期的，这就是多周期的 CPU。

5.1 实验总体框架及多周期 datapath 连线图

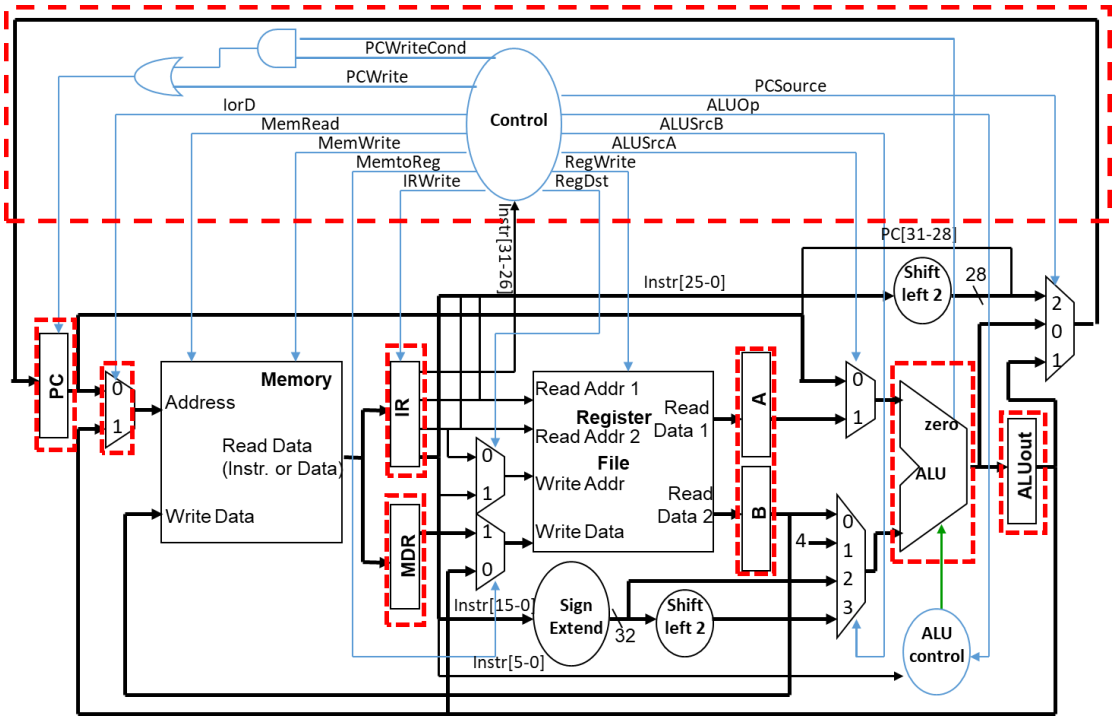


图 1：单周期 CPU 系统连线图

如图 1，完整的多周期 CPU 实现框架图。图中 datapath 大部分模块我们已经在单周期实验中完成了，我们只需要再此基础上进行相应的改写，并重新连接模块，既可以完成多周期 MIPS 数据通路的实现；

特别提示，图上增加 IR 指令寄存器，目的是使指令代码保持稳定，pc 写使能控制信号，是确保 pc 适时修改，原因都是和多周期工作的 CPU 有关。A、B、ALUout 四个寄存器不需要写使能信号，其作用是切分数据通路，将大组合逻辑切分为若干个小组合逻辑，大延迟变为多个分段小延迟。

思考：ALUout 是否可以和课堂 PPT 里面的 Target 寄存器合并？如果不可以，如何改动？

5.2 控制器设计

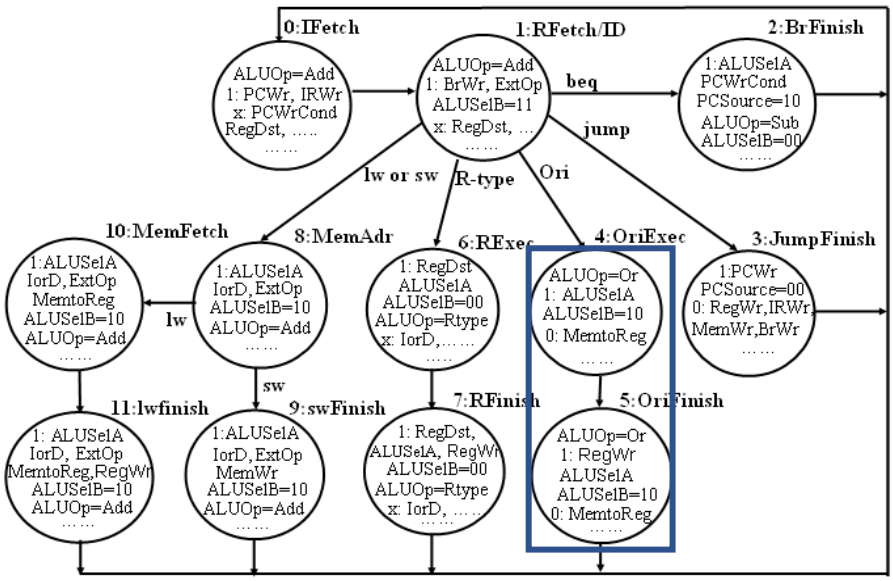
在前面我们讲过，多周期 CPU 是把 5 个不同的执行阶段放到 5 个单独的 clk 里面。这样带来的好处是，不同的指令有不同的周期数。同时，不同指令在不同的周期（stage）也有不同的输出控制信号。在不同指令不同周期，信号如何给，同学们可以参考我们上课的 PPT。下面我们首先给出控制信号的定义，然后给出控制器状态机表述，同时给出控制器在 IF 和 ID 阶段的控制信号（控制器状态机其他状态的状态机，需要同学们自己去查看 PPT）。

表 1 控制信号作用

控制信号名	描述
PCWrite	PCWrite =0，PC 不更改；PCWrite =1, PC 更改
ALUSrcA	ALUSrcA=0, ALU 的输入来自 PC; ALUSrcA=1, ALU 的输入来自寄存器堆；
ALUSrcB	ALUSrcB=00，ALU 的输入来自寄存器堆； ALUSrcB=01，ALU 的输入等于 4；用于做 PC+4; ALUSrcB=10，ALU 的输入来自与立即数； ALUSrcB=11，ALU 的输入来自与立即数迁移，主要用于 beq 指令；
MemtoReg	MemtoReg=0，来自 ALU 的输出写入 registerfile，例如 R-type 指令 MemtoReg=1，来自 Memory 出写入 registerfile，例如 lw 指令

RegWritem	Registerfile 写入控制信号，0 为读，1 为写入
RegDst	寄存器写入地址是 rd 还是 rt (和单周期类似) 注意：简单起见，我们未考虑 jal 指令
MemWrite	读写指令和存储存储器
IRWrite	IRWre =0；IR(指令寄存器)不更改；IR 寄存器写使能。向指令存储器发出读指令代码后，这个信号也接着发出，在时钟上升沿，IR 接收从指令存储器送来的指令代码。与每条指令都相关。
ExtOp	(zero-extend)immediate，相关指令：andi、xori、ori；(sign-extend)immediate，相关指令：addiu、slti、lw、sw、beq、bne、bltz；
PCSrc[1..0]	00：pc< - pc+4，相关指令：add、addiu、sub、and、andi、ori、xori、slt、slti、sll、sw、lw、beq(zero=0)、bne(zero=1)、bltz(sign=0)； 01：pc< - pc+4+(sign-extend)immediate ×4，相关指令：beq(zero=1)、bne(zero=0)、bltz(sign=1)； 10：pc< - {pc[31:28],addr[27:2],2'b00}，相关指令：j、jal；
ALUOp[2..0]	ALU 8 种运算功能选择(000-111)，见单周期实验
PCWriteCond	Beq 信号的 PC 更新旁路信号
IorD	选择选指令存储器还是数据存储器

不同的指令有不同的周期数，不同指令在不同的周期（stage）也有不同的输出控制信号。多周期的控制器需要依赖于状态机设计实现。根据理论课上的讲解内容，我们可以设计如下的状态机。



注意：蓝色部分代表的立即数的计算，不仅仅代表 Ori 指令。图中的圆圈代表不同指令在不同周期的控制信号输出的集合。

下面我们列举上面状态流程图的两个公共状态 IF 和 ID 的控制信号的输出，对应上图中状态 0 和 1 的圆圈里面的控制信号。其他状态的控制信号请大家在实验报告中补全(参照理论课上的 PPT 中内容)。

表 2 控制信号表

控制信号名	IFetch	ID	BrFinish	…	…	…

PCWrite	1	0				
ALUSrcA	0	0				
ALUSrcB	01	10				
MemtoReg	x	x				
RegWrite	0	0				
RegDst	x	x				
MemWrite	0	0				
IRWrite	1	0				
ExtOp	0	1				
PCSrc[1..0]	0	x				
ALUOp[2..0]	ADD	ADD				
PCWriteCond	x	0				
IorD	0	x				

六、 实验报告要求

1. 按照数据流过程，填充表 2 中其他状态的控制信号（总共 11 个状态），并核验其正确性；并按照给出的表格，设计多周期控制器的 controller。
2. 按照实验要求 4.1 搭建如图 1 所示实验系统。
3. 在指令存储器中存入下列程序，并通过仿真验证每一类指令的结果，附上仿真波形图，说明其正确性。并需要通过仿真结果，说明所设计的处理器是否能够跑完所有代码，在 end 处执行了正确的代码。

```
#          Assembly                Description                  Address   Machine
main:      addi $2, $0, 5          # initialize $2 = 5          0         20020005   20020005
           addi $3, $0, 12         # initialize $3 = 12         4         2003000c   2003000c
           addi $7, $3, -9         # initialize $7 = 3          8         2067fff7   2067fff7
           or   $4, $7, $2         # $4 <= 3 or 5 = 7           c         00e22025   00e22025
           and  $5, $3, $4         # $5 <= 12 and 7 = 4         10        00642824   00642824
           add  $5, $5, $4         # $5 = 4 + 7 = 11            14        00a42820   00a42820
           beq  $5, $7, end        # shouldn't be taken         18        10a7000a   10a7000a
           slt  $4, $3, $4         # $4 = 12 < 7 = 0            1c        0064202a   0064202a
           beq  $4, $0, around     # should be taken            20        10800001   10800001
           addi $5, $0, 0          # shouldn't happen           24        20050000   20050000
around:    slt  $4, $7, $2         # $4 = 3 < 5 = 1             28        00e2202a   00e2202a
           add  $7, $4, $5         # $7 = 1 + 11 = 12           2c        00853820   00853820
           sub  $7, $7, $2         # $7 = 12 - 5 = 7            30        00e23822   00e23822
           sw   $7, 68($3)         # [80] = 7                   34        ac670044   ac670044
           lw   $2, 80($0)         # $2 = [80] = 7              38        8c020050   8c020050
           j    end                # should be taken            3c        08000011   08000011
           addi $2, $0, 1          # shouldn't happen           40        20020001   20020001
end:       sw   $2, 84($0)         # write adr 84 = 7           44        ac020054   ac020054
```

3. 在实验报告中，详细阐述实验设计过程，代码模块结构，以及观察到实验结果以及验证原理（根据仿真波形说明你设计的处理器为什么是正确的，能运行的），并分析实验结果。

在本文档中，提供的相关内容对于设计可能不足或甚至有错误，希望同学们在设计过程中如发现有问题，请你们自行改正，进一步补充、完善。谢谢！

附录 1：状态机 verilog 模板，用于控制器实现

Step1: 申明变量和状态更新

Localparam 是定义的状态数目，

```
localparam IDLE=0, WAITFORB=1,
           DONE=2, ERROR=3;
reg [1:0] state,    // Current state
          nxtState; // Next state

always @(posedge clk) begin
  if (reset) begin
    state <= IDLE;   // Initial state
  end else begin
    state <= nxtState;
  end
end
```

Step2: 状态转移和输出

```
always @(*) begin
  nxtState = state;  // Default next state: stay where we are
  out = 0;           // Default output
  case (state)
    IDLE : begin
      if (B) nxtState = ERROR;
      else if (A) nxtState = WAITFORB;
    end
    WAITFORB : begin
      if (B) nxtState = DONE;
    end
    DONE : begin
      out = 1;
    end
    ERROR : begin
    end
  endcase
end
```

附录 2：带使能的触发器

```
module floprE #(parameter WIDTH = 8)(
    input wire clk,rst,
    input wire en,
    input wire[WIDTH-1:0] d,
    output reg[WIDTH-1:0] q
    );
    always @(negedge clk,posedge rst) begin
        if(rst) begin
            q <= 0;
        end else begin
            if(en)
                q <= d;
            else
                q <= q;
        end
    end
endmodule
```