实验五流水线 CPU 设计

Momo Tori

May 8, 2022

目录

流水线 cpu 设计

数据通路

数据寄存器的通路如下

```
//data path of pipeline
always @(posedge clk or negedge rstn) begin
  if(~rstn)
  begin
  PCD_r<=0;
  IR_r<=32'h00000013;//NOP
  PCE_r<=0;
```

```
A_r<=0;
B_r<=0;
Imm_r<=0;
Rd_r<=0;
Y_r<=0;
MDW_r<=0;
RdM_r<=0;
MDR_r<=0;
YW_r<=0;
RdW_r<=0;

IR_EX_r<=32'h00000013;
IR_MEM_r<=32'h00000013;
IR_WB_r<=32'h00000013;
end
else begin

if(B_Hazard)//branch hazard, IR_EX 的 Instruction 清为 NOP
    IR_r<=32'h00000013;
else if(LD_R_Hazard)//Load hazard, stall 一个周期
    IR_r<=IR_r;
else
    IR_r<=Ins;

if(LD_R_Hazard)begin//Load hazard, IR_EX 的 PC stall 一个周期
  PCD_r<=PCD_r;
end
else begin
  PCD_r<=pc;
end

//ID_EX 段寄存器
PCE_r<=PCD_r;
A_r<=Reg1Data;
```

```
  B_r<=Reg2Data;
  Imm_r<=Imm;
  Rd_r<=IR_r[11:7];

  //EX_Mem 段寄存器
  Y_r<=ALUResult;
  MDW_r<=B_r_fixed;//B_r_fixed 为 forwarding 之后的 B_r
  RdM_r<=Rd_r;

  //Mem_Wb 段寄存器
  MDR_r<=ReadData;
  YW_r<=Y_r;
  RdW_r<=RdM_r;

  //特殊地，指令寄存器
  IR_EX_r<=IR_r;
  IR_MEM_r<=IR_EX_r;
  IR_WB_r<=IR_MEM_r;
  end
end
```

流水线的控制信号传递

```
//control sign of pipeline

//EX 段信号
always @(posedge clk or negedge rstn) begin
  if(~rstn)
  begin
  MemtoReg_r_EX<=0;
  MemWrite_r_EX<=0;
  ALUSrc_r_EX<=0;
  RegWrite_r_EX<=0;
  MemRead_r_EX<=0;
  PCChange_r_EX<=0;
  AUIPC_r_EX<=0;
  end
  else begin
  if(LD_R_Hazard|B_Hazard)//发生 Hazard, 原 ID 周期的写入全部丢弃
  begin
    MemtoReg_r_EX<=0;
    MemWrite_r_EX<=0;
    RegWrite_r_EX<=0;
    MemRead_r_EX<=0;
    PCChange_r_EX<=0;
  end
  else
  begin
    MemtoReg_r_EX<=MemtoReg;
    MemWrite_r_EX<=MemWrite;
    RegWrite_r_EX<=RegWrite;
    MemRead_r_EX<=MemRead;
    PCChange_r_EX<=PCChange;
  end
  ALUOp_r<=ALUOp;

  ALUSrc_r_EX<=ALUSrc;
  AUIPC_r_EX<=AUIPC;
  end
end
```

```
//Mem 段信号
always @(posedge clk or negedge rstn) begin
  if(~rstn)
  begin
  MemtoReg_r_MEM<=0;
  MemWrite_r_MEM<=0;
  RegWrite_r_MEM<=0;
  MemRead_r_MEM<=0;
  end
  else begin
  MemtoReg_r_MEM<=MemtoReg_r_EX;
  MemWrite_r_MEM<=MemWrite_r_EX;
  RegWrite_r_MEM<=RegWrite_r_EX;
  MemRead_r_MEM<=MemRead_r_EX;
  end
end
```

```
//Wb 段信号
always @(posedge clk or negedge rstn) begin
  if(~rstn)
  begin
  MemtoReg_r_WB<=0;
  RegWrite_r_WB<=0;
  end
  else begin
  MemtoReg_r_WB<=MemtoReg_r_MEM;
  RegWrite_r_WB<=RegWrite_r_MEM;
  end
end
```

Forwarding 单元

对写入的数据提前返回需要的 EX 段

```
//forwarding unit
//EX 进行的指令是否有 SR1 或 SR2 源寄存器
wire SR1,SR2;
assign SR1=~((IR_EX_r[6:2]==5'b11011)|AUIPC_r_EX);
assign SR2=(IR_EX_r[3:2]==2'b00&IR_EX_r[5]);
//是否需要 Wb to Ex 或 Mem to Ex
wire Wb2Ex_sr1,Wb2Ex_sr2,Mem2Ex_sr1,Mem2Ex_sr2;
assign Wb2Ex_sr1=(RdW_r==IR_EX_r[19:15]) & SR1 & RegWrite_r_WB;
assign Wb2Ex_sr2=(RdW_r==IR_EX_r[24:20]) & SR2 & RegWrite_r_WB;
assign Mem2Ex_sr1=(RdM_r==IR_EX_r[19:15]) & SR1 & RegWrite_r_MEM;
assign Mem2Ex_sr2=(RdM_r==IR_EX_r[24:20]) & SR2 & RegWrite_r_MEM;
```

A 和 B 根据前面的信号，有

```
reg [31:0] A_r_fixed;//修正后的 A
reg [31:0] B_r_fixed;//修正后的 B

always @(*) begin
  case({Mem2Ex_sr1,Wb2Ex_sr1})
  2'b00:A_r_fixed=A_r;
  2'b01:A_r_fixed=WriteData;
  2'b10:A_r_fixed=Y_r;
  2'b11:A_r_fixed=Y_r;
  default:A_r_fixed=32'hxxxxxxxx;
  endcase

  case({Mem2Ex_sr2,Wb2Ex_sr2})
  2'b00:B_r_fixed=B_r;
  2'b01:B_r_fixed=WriteData;
  2'b10:B_r_fixed=Y_r;
  2'b11:B_r_fixed=Y_r;
  default:B_r_fixed=32'hxxxxxxxx;
  endcase
end
```

Hazard 单元

有 Load-Use Hazard 和 Branch Hazard，分别如下

```
//Load-Use Hazard 部分
//对于 LD-R 类型编排的指令必须有 Load-Use Hazard
//若 LD_R_Hazard 为真则阻塞 EX 段的输入，反而输入 EX 为一个 NOP

//判断是否有数据冲突
wire SR1_ID,SR2_ID,isLWHazard;
```

```
assign SR1_ID=~((IR_r[6:2]==5'b11011)|AUIPC);
assign SR2_ID=(IR_r[3:2]==2'b00&IR_r[5]);
assign isLWHazard=( (Rd_r==IR_r[19:15]) & SR1_ID & (| IR_r[19:15]) )
                  | ( Rd_r==IR_r[24:20] & SR2_ID &(|IR_r[24:20]) );

//LD_R_Hazard=EX 为 LW 且 ID 需要 Mem 读出的数据
wire LD_R_Hazard;
//其中 MemtoReg_r_EX=LW_EX 为 EX 段是否是 LW 命令
assign LD_R_Hazard=MemtoReg_r_EX&isLWHazard;
```

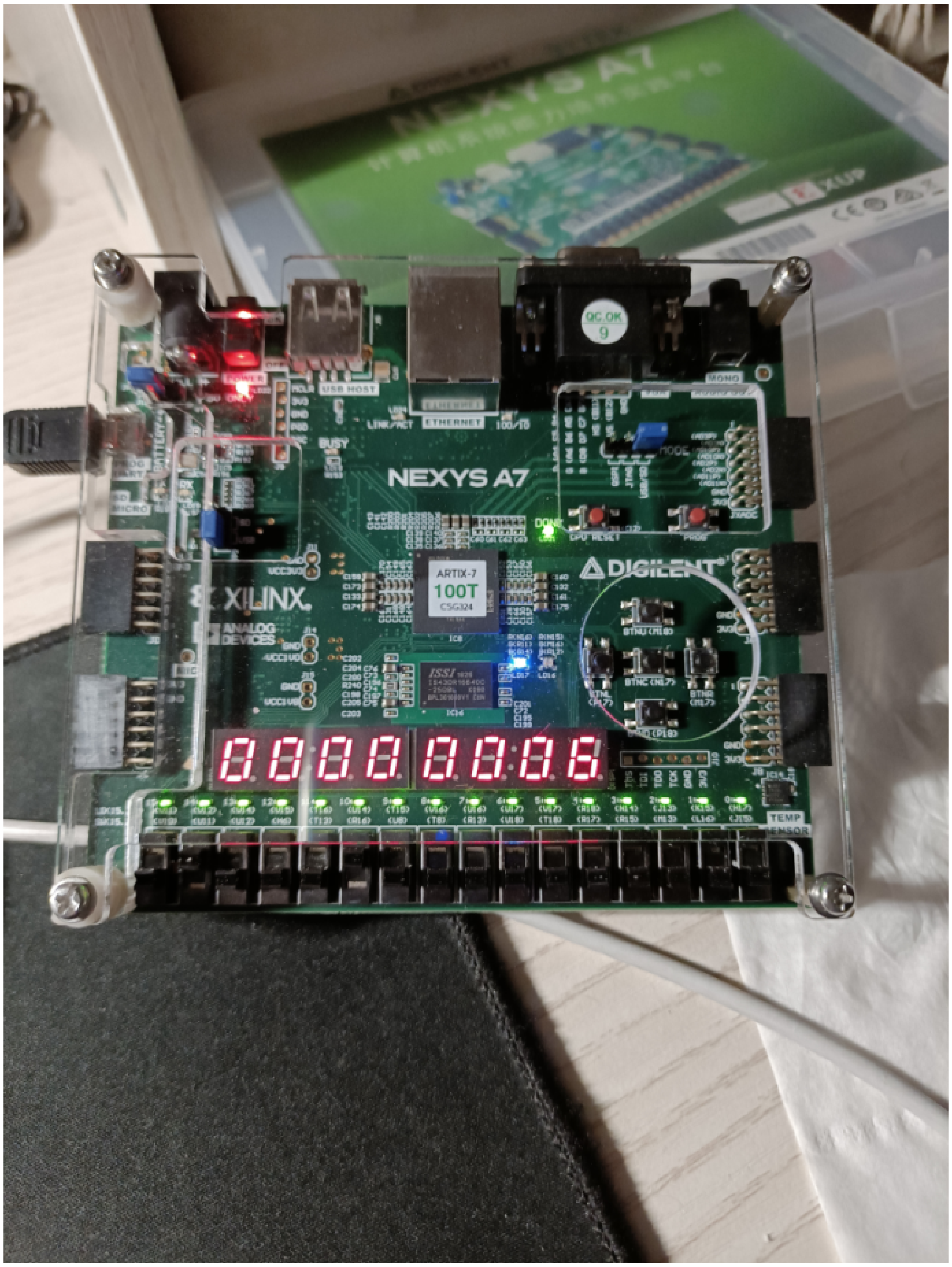
```
//Branch Hazard 部分
//跳转成功时将已经进入流水线的两个指令清除为 NOP
wire B_Hazard;
assign B_Hazard=PCChange_r_EX&(((IR_EX_r[3:2]==2'b00)&zero)|IR_EX_r[2]);
```

流水线 CPU 测试下载

通过测试文件测试均正常运行

流水线 CPU 数组排序下载测试

测试图如下，可以正常输出通过 switch 输入的数组


NEXYS A7
ARTIX-7
100T
CSG324
XILINX
DIGILENT
00000006

实验总结

通过本实验了解并制作了流水线 CPU，学习了从改变方式到发现因其产生的矛盾再到解决矛盾的思路