数字逻辑与处理器基础实验

流水线MIPS处理器设计实验报告

2019011008

无92 刘雪枫

**目录**

[一、实验名称与内容 1](#_Toc78695380)

[二、处理器设计 1](#_Toc78695381)

[（一）实现的指令集 1](#_Toc78695382)

[（二）数据通路 1](#_Toc78695383)

[（三）处理器模块 2](#_Toc78695384)

[（四）工作过程 3](#_Toc78695385)

[I. 工作阶段 3](#_Toc78695386)

[II. 分支与跳转 3](#_Toc78695387)

[III. 冒险 4](#_Toc78695388)

[三、综合与实现情况 6](#_Toc78695389)

[（一）时序性能 6](#_Toc78695390)

[（二）逻辑资源占用情况 7](#_Toc78695391)

[四、仿真验证 7](#_Toc78695392)

[五、硬件调试情况 8](#_Toc78695393)

[六、程序清单 9](#_Toc78695394)

[七、心得体会 10](#_Toc78695395)

[附录 11](#_Toc78695396)

[附录一 指令格式表 11](#_Toc78695397)

[附录二 指令说明表 12](#_Toc78695398)

# 一、实验名称与内容

实验名称：流水线MIPS处理器设计

实验内容：将春季学期实验四设计的多周期 MIPS 处理器改进为流水线结构，并利用此处理器完成背包算法。

# 二、处理器设计

## （一）实现的指令集

本次实验实现的指令集为：

1. R型算术指令：add、addu、sub、subu、and、or、xor、nor、slt、sltu、sll、、srl、sra；
2. I型算术指令：lui、addi、addiu、andi、sltiu；
3. 存取指令：lw、sw；
4. 分支指令：beq、bne、blez、bgtz、bltz；
5. 跳转指令：j、jal、jr、jalr。

上述指令的具体格式与各个指令的作用参见附录一与附录二。

## （二）数据通路

数据通路图如下：



## （三）处理器模块

根据上述指令集，我设计了如下模块：

1. PipelineCPU：总连线模块；
2. Controller：控制信号模块；
3. ALU：算术逻辑单元模块；
4. RegFile：寄存器堆模块；
5. DataMem：数据存储器模块；
6. InstMem：指令存储器模块；
7. PC：程序计数器模块；
8. ImmExtend：立即数扩展模块；
9. IF\_ID\_Reg、ID\_EX\_Reg、EX\_MEM\_Reg、MEM\_WB\_Reg：流水线寄存器模块；
10. DataHazard：数据冒险模块；
11. BranchAndJumpHazard：分支冒险模块；
12. TestBench：Test bench模块

## （四）工作过程

### I. 工作阶段

IF阶段负责程序计数器的更新以及取指令，以PC寄存器的输出为指令地址得到指令，并储存在IF/ID寄存器中。

ID阶段负责寄存器堆的读取和立即数扩展，并进行j、jr、jal、jalr跳转指令的判断。本阶段将寄存器堆中读取的寄存器数据以及立即数扩展结果存入到ID/EX寄存器中。

EX阶段进行ALU运算，以及将要写入的寄存器地址WriteAddr的选择。

MEM阶段进行数据存储器的读取或写入。

EX阶段选择写入寄存器堆的数据，并将数据写入寄存器堆。

### II. 分支与跳转

在ID阶段进行j、jr、jal、jalr跳转指令的跳转；在EX阶段进行beq、bne、blez、bgtz、bltz分支指令的跳转。由于分支指令的判断晚于跳转指令一个阶段，因此为了避免分支指令后面紧跟的跳转指令产生冒险，本处理器采取分支指令优先的方式，即在电路逻辑上先判断分支指令是否跳转，再判断跳转指令是否跳转。

对于分支指令，使用两个控制信号Branch和BranchClip进行控制。Branch控制信号用来控制本指令是否是分支指令。而分支指令是否跳转还取决于ALU运算的结果。由于分支指令是否跳转取决于操作数比较大小的结果，因此比较结果只可能是0或1，故根据ALU输出的zero信号进行判断。其中，beq和bne指令采用两个操作数之差是否为零进行判断，blez和bgtz采用大于运算进行比较，bltz采用小于运算进行比较。这样，5个分支指令有些指令是当zero为0时跳转，有些指令是当zero为1时跳转。因此引入BranchClip信号，代表是否对zero信号取反，从而统一为当信号为1时跳转。这样只需要BranchClip对zero做异或运算即可。这样，分支指令跳转的充要条件是：ID\_EX\_Reg.Branch && (ALU\_zero ^ ID\_EX\_Reg.BranchClip)。鉴于后续书写的简便考虑，代码中记录了不进行跳转的信号：no\_branch = (ID\_EX\_Reg.Branch == 0 || !(ALU\_zero ^ ID\_EX\_Reg.BranchClip))，即等价于ID\_EX\_Reg.Branch与(ALU\_zero ^ ID\_EX\_Reg.BranchClip)进行与非运算。

对于跳转指令，使用了Jump和JumpSrc两个控制信号。Jump信号用于表示当前指令是否是跳转指令，而JumpSrc信号则控制跳转的目标地址是来自于立即数（j、jal）还是寄存器（jr、jalr）。

根据之前所述，分支优先于跳转。因此，最终的逻辑为，当no\_branch为0时，即进行分支指令的跳转，否则当Jump为1时，进行跳转指令的跳转，否则不进行跳转。

关键代码为：

assign J\_out = Jump == 0 ? (PC\_o + 4) :

JumpSrc == 0 ? {PC\_Plus\_4[31:28], rs, rt, rd, Shamt, Funct, 2'b00

} : ReadData1Actual;

assign no\_branch = (ID\_EX\_Reg.Branch == 0 || !(ALU\_zero ^ ID\_EX\_Reg.BranchClip));

assign PC\_i = no\_branch ? J\_out : ID\_EX\_Reg.PC\_Plus\_4 + (ID\_EX\_Reg.imm\_ext << 2);

其中，PC\_i为PC模块的输入。

### III. 冒险

i) 数据冒险

关于寄存器值的冒险，注意到，由于最早用到寄存器值的阶段是jr指令，jr指令在ID阶段就要获取寄存器的值，因此对于算术指令，因此对于数据冒险，需要在ALU刚刚计算出结果时就要进行转发，并且需要转发到ID阶段，而不能在ALU计算结果写入EX/MEM寄存器之后将数据转发到ALU运算的输入，因为这会导致jr指令可能无法得到正确的值。因此将转发提前到ALU计算结果后并转发到ID阶段是必要的。因此转发的逻辑（关键的转发代码）为：

assign ForwardA =

                (ID\_EX\_RegWrite && ID\_EX\_WriteAddr != 0 && ID\_EX\_WriteAddr == rs) ? 2'b01 :

                (EX\_MEM\_RegWrite && EX\_MEM\_WriteAddr != 0 && EX\_MEM\_WriteAddr == rs) ? 2'b10 :

                2'b00;

assign ForwardB =

                (ID\_EX\_RegWrite && ID\_EX\_WriteAddr != 0 && ID\_EX\_WriteAddr == rt) ? 2'b01 :

                (EX\_MEM\_RegWrite && EX\_MEM\_WriteAddr != 0 && EX\_MEM\_WriteAddr == rt) ? 2'b10 :

                2'b00;

对于Load/Use冒险，采取阻塞一个周期的方式。使用LW\_Stall信号代表需要阻塞一个周期，关键代码为：

assign LW\_Stall =

                ID\_EX\_MemRead && (ID\_EX\_WriteAddr != 0) && (ID\_EX\_WriteAddr == rs || ID\_EX\_WriteAddr == rt);

ii) 控制冒险

当需要进行跳转时，会产生控制冒险，需要擦除掉已经执行的指令。由于跳转指令需要擦除一条指令，而分支指令需要擦除两条指令，因此关键代码为：

assign IF\_ID\_Flush = Jump || !no\_branch;

assign ID\_EX\_Flush = !no\_branch;

iii) 阻塞与擦除

综合上述冒险，当产生阻塞一个周期的请求时，需要擦除ID/EX阶段的指令，并让PC寄存器和IF/ID寄存器不更新。分别使用PC\_Hold和IF\_ID\_Hold信号来使其不更新。设控制冒险中要求ID/EX寄存器擦除的信号为Branch\_ID\_EX\_Flush，而最终的擦除信号为ID\_EX\_Flush，则最终的代码为：

assign PC\_Hold = LW\_Stall;

assign IF\_ID\_Hold = LW\_Stall;

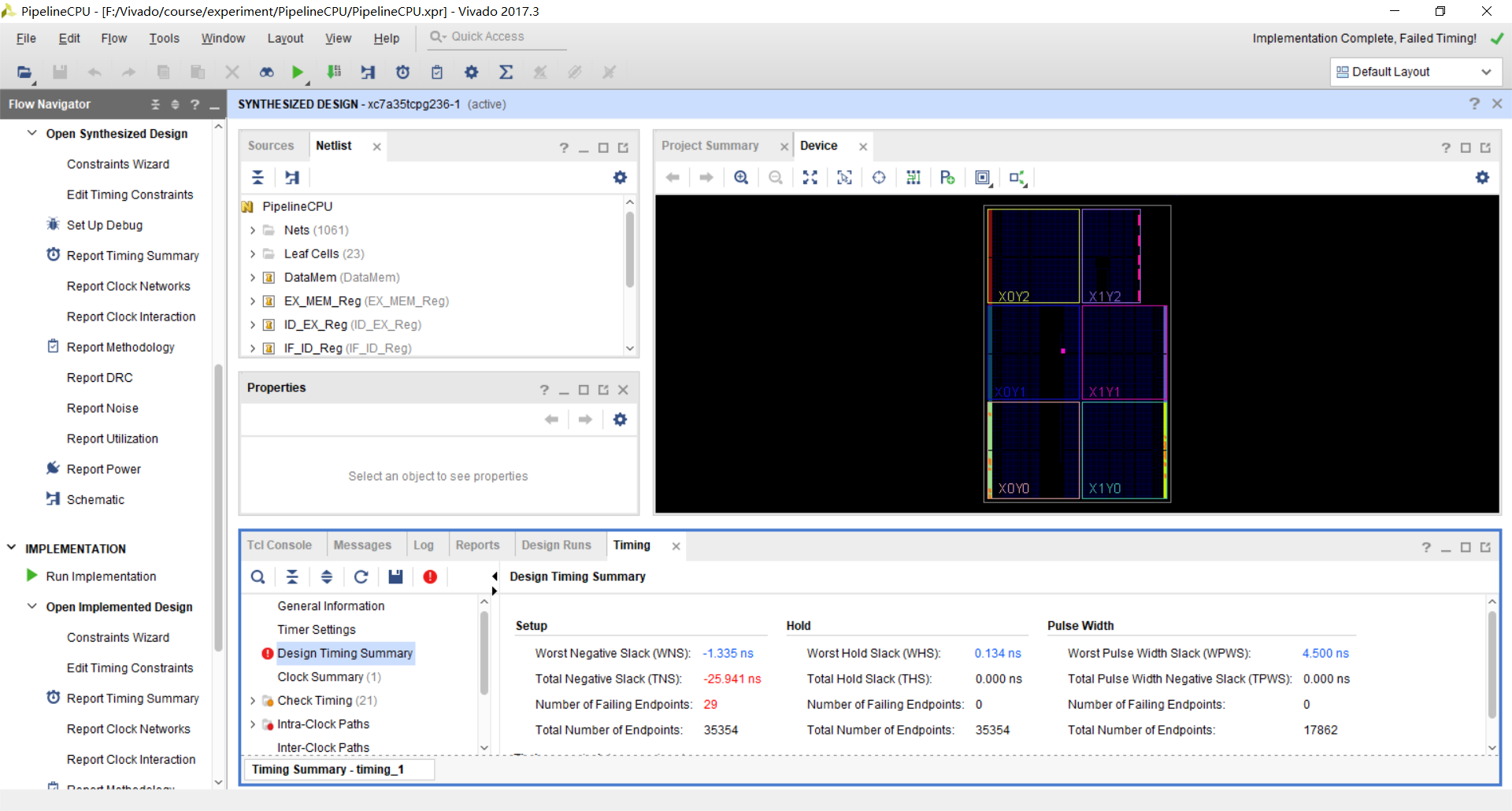
assign ID\_EX\_Flush = Branch\_ID\_EX\_Flush || LW\_Stall;

此外，IF\_ID\_Flush信号直接连接到控制冒险的IF\_ID\_Flush信号即可。

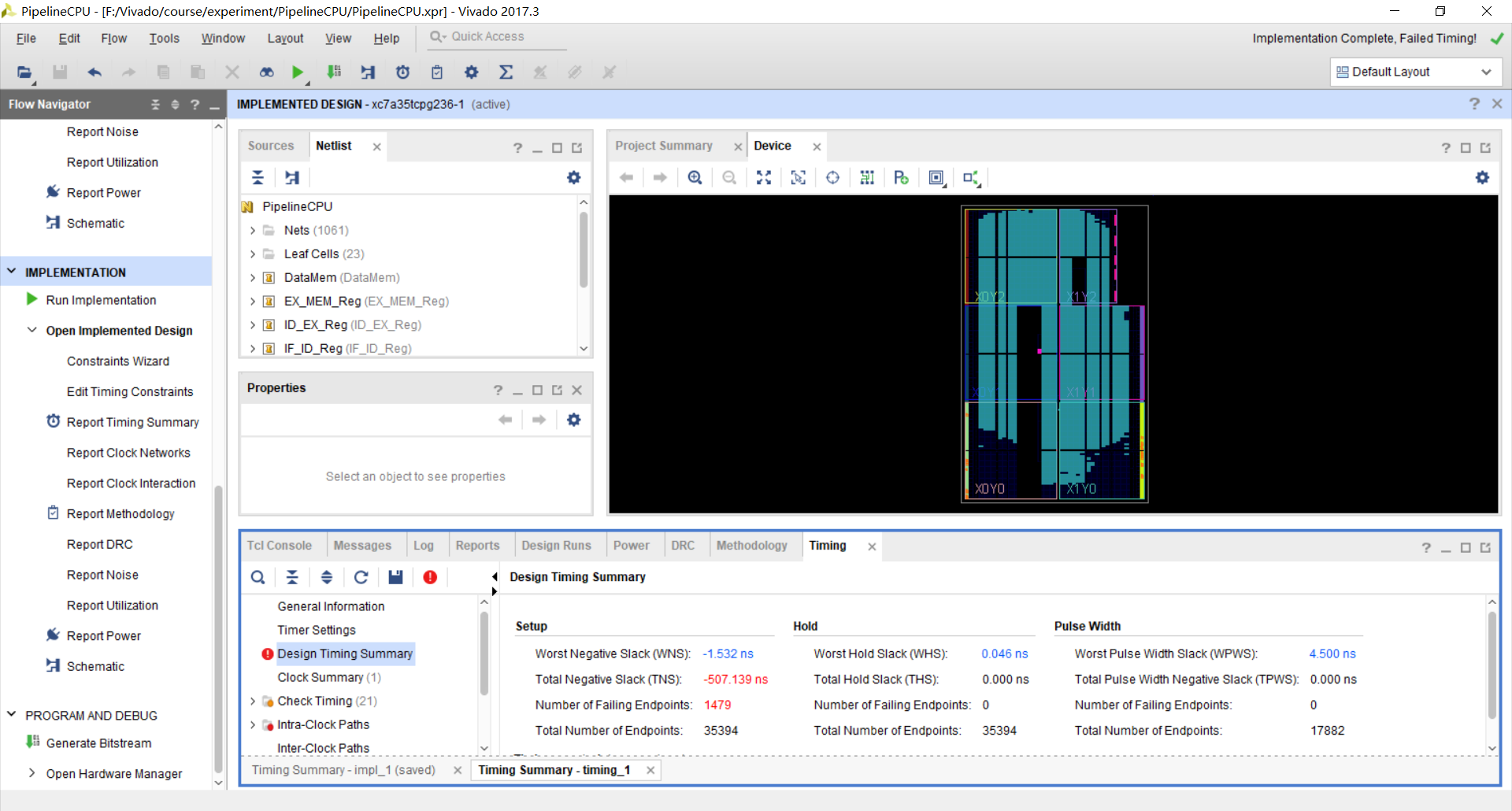
# 三、综合与实现情况

## （一）时序性能

本次的管脚约束中，将时钟周期设置为10.00ns，观察综合后的时序裕量为-1.335ns：

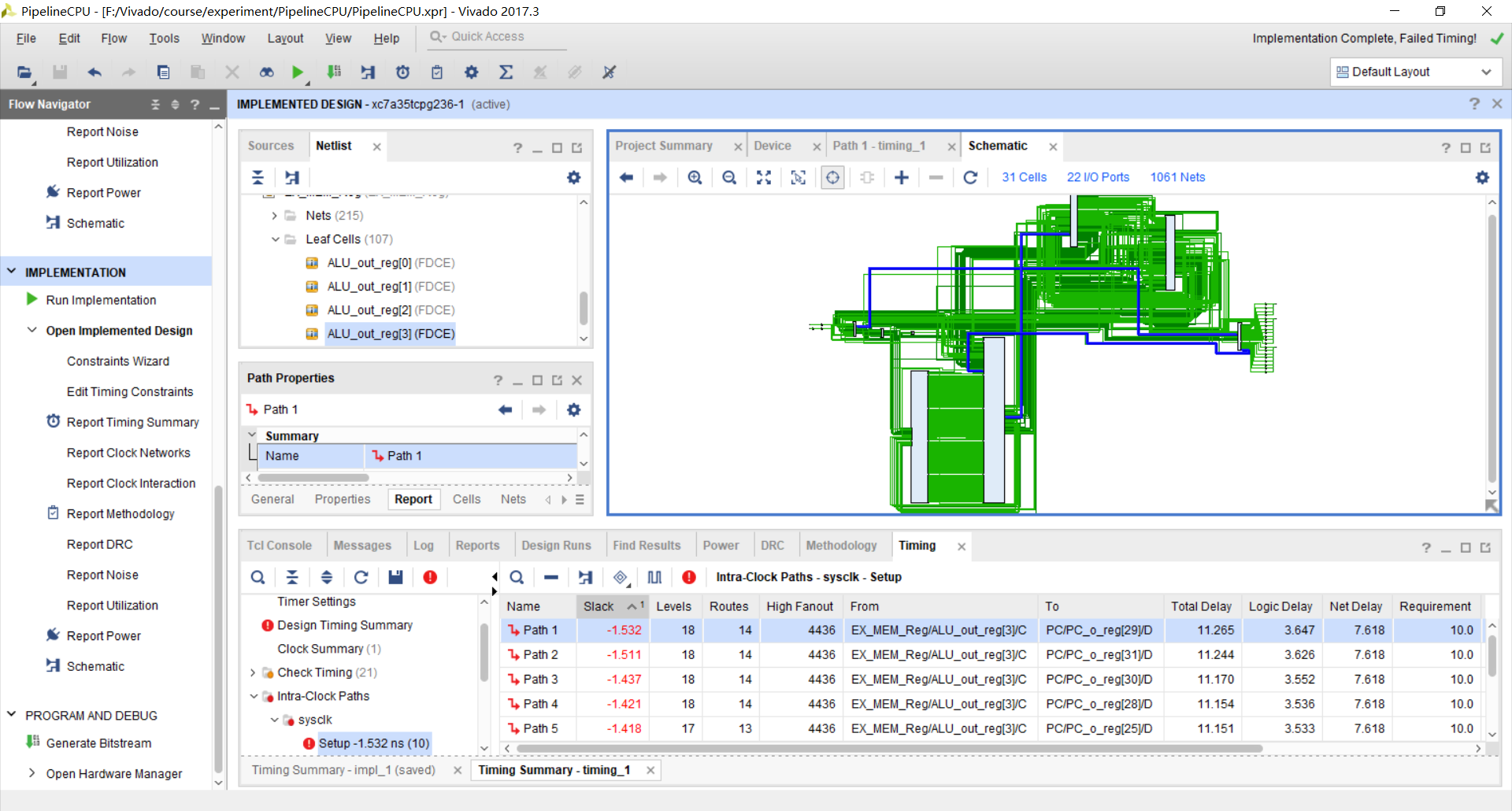


综合后的时序裕量为-1.532ns：



时序裕量为负值，说明时钟周期设置的偏短。预计时钟周期最短应为11.54ns，因此频率最高大致为86.7MHz。

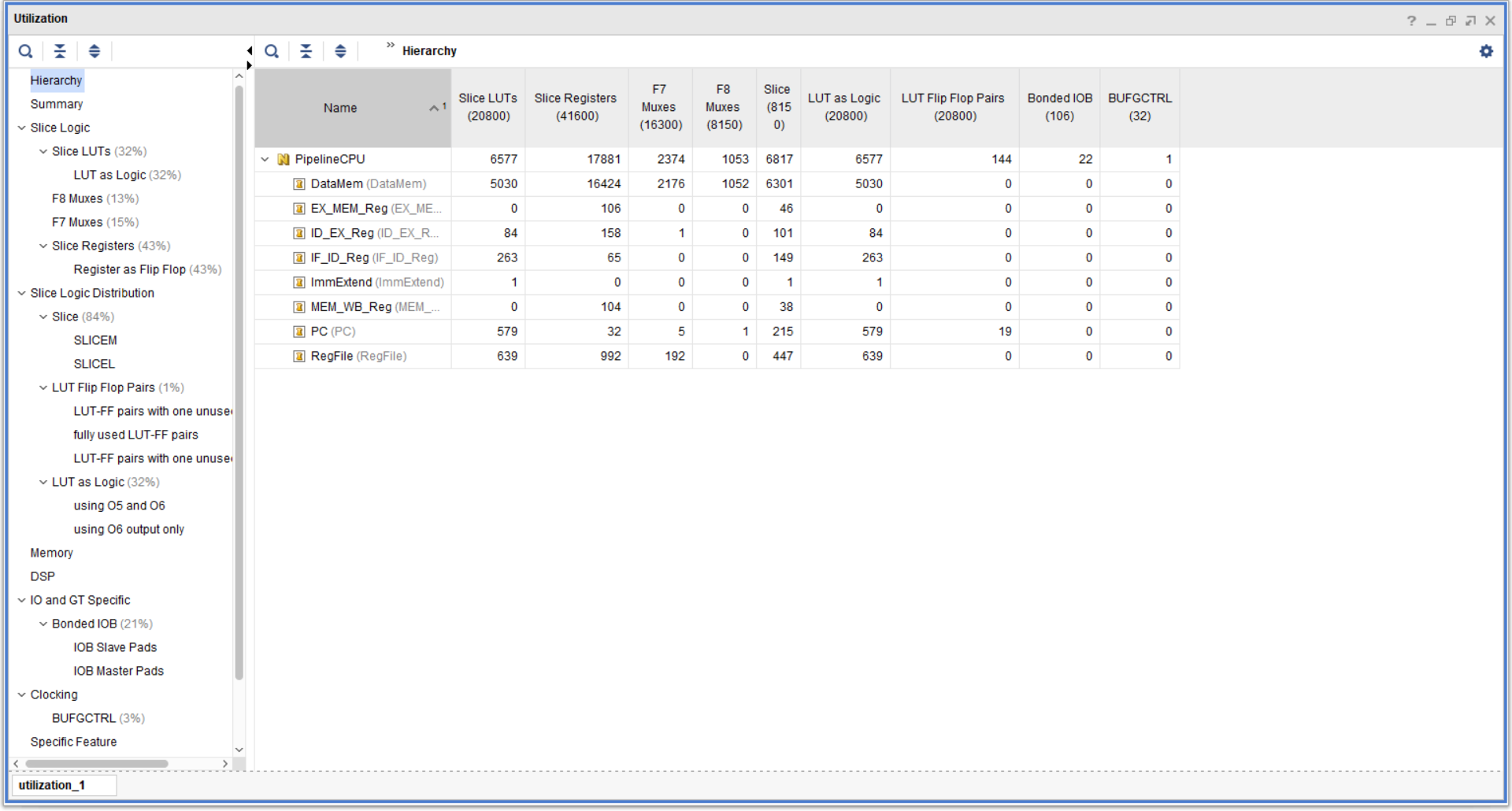
观察关键路径：

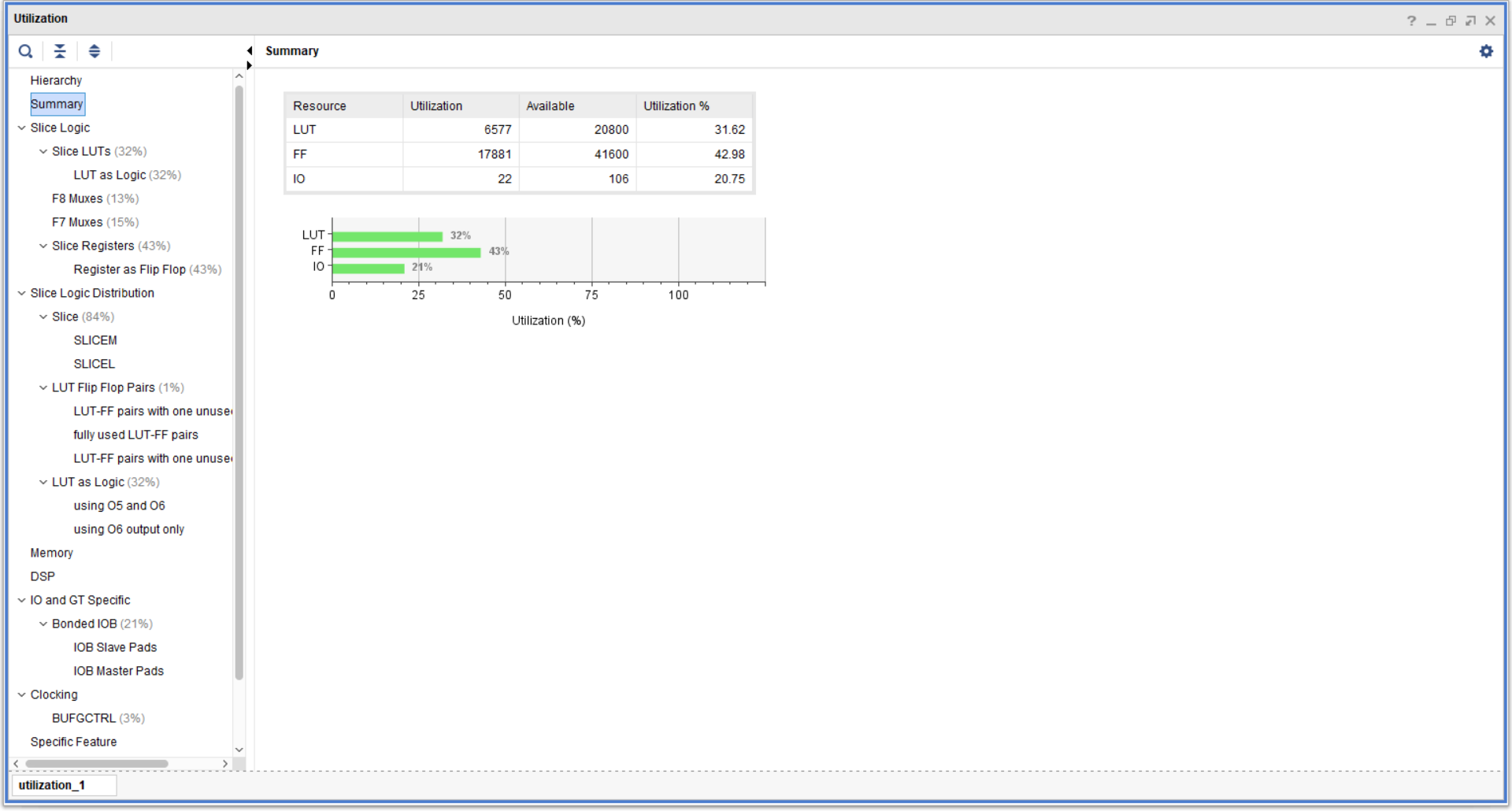


可以看到关键路径是包含数据寄存器DataMem和冒险检测单元的路径。因此处理器的频率主要被数据存储器和冒险检测单元所限制。

## （二）逻辑资源占用情况

处理器的逻辑资源占用情况如下：





# 四、仿真验证

本次实验经过了多次汇编程序对冒险进行测试，均通过测试。测试所用汇编程序参见asm\_program中的hazard\_test文件夹。

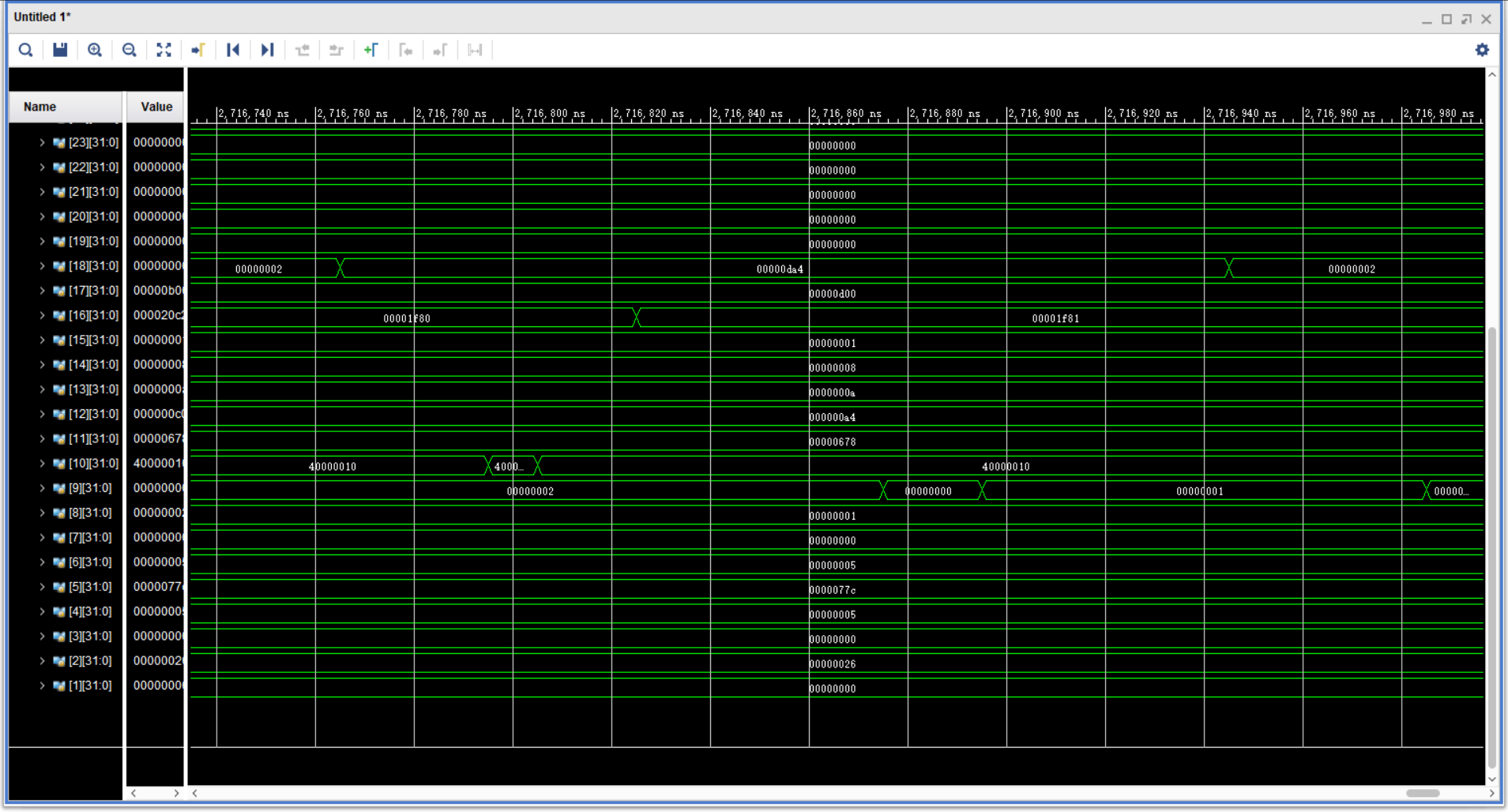
最终，用此CPU解决背包问题。背包问题的汇编程序参见asm\_program中的pack.asm。

本次背包的测试数据为：

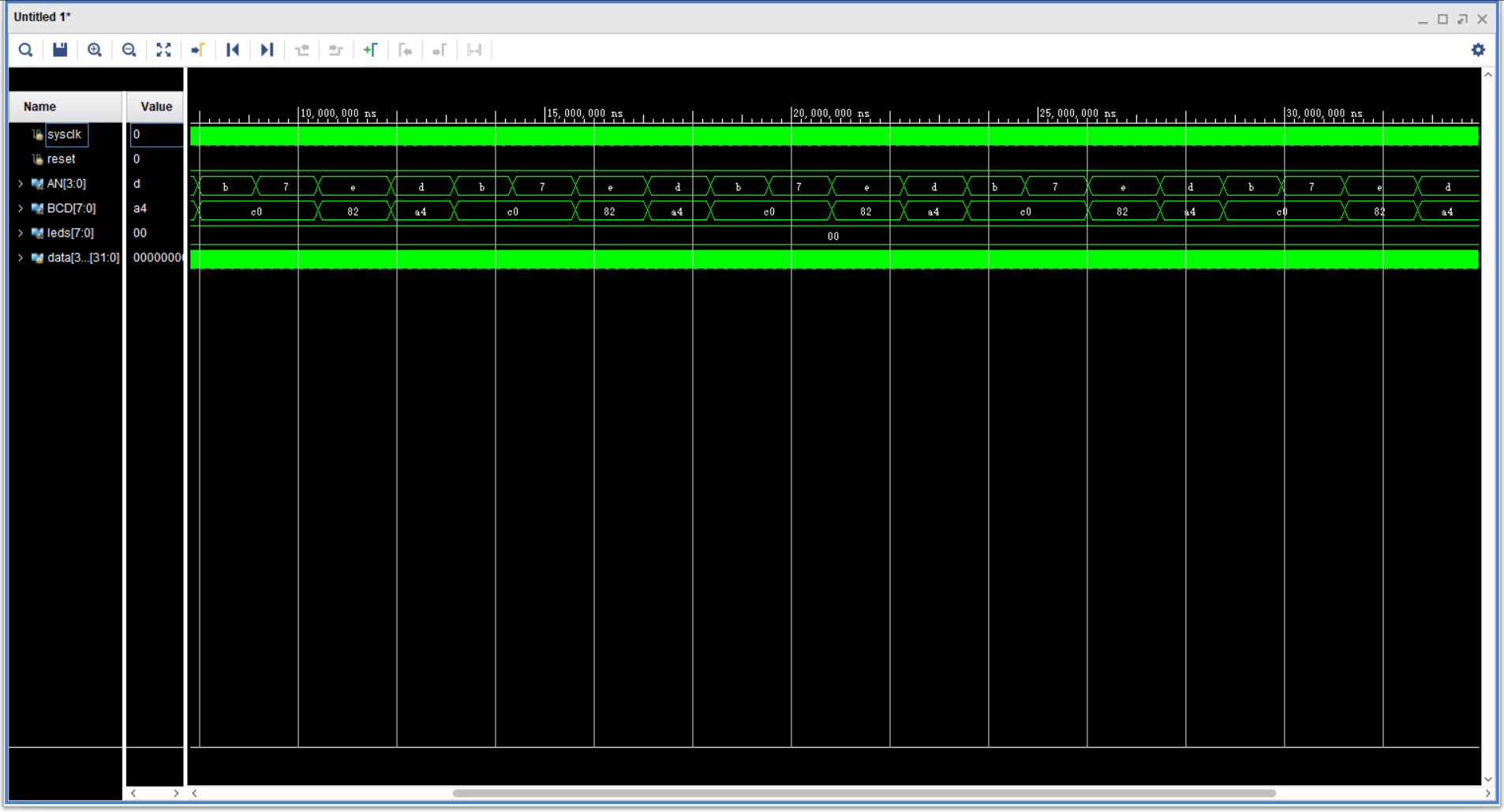
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 重量 | 2 | 1 | 3 | 2 | 1 |
| 价值 | 12 | 10 | 20 | 15 | 8 |

答案应为38，转为十六进制是26。

进行仿真结果如下：



可以看到，最终寄存器$v0的值确实是十六进制的26。继续仿真，观察七段数码管（即AN与BCD）的值：



可以看到，最终AN与BCD的值若显示在七段数码管上也确实是0026，因此仿真结果正确。

# 五、硬件调试情况

将设计代码使用Vivado 2020.2烧录到FPGA上之后运行，七段数码管上很快便出现结果“0026”，因此硬件调试成功。

# 六、程序清单

本次报告的目录结构如下：

│ 实验报告.pdf

│

├─asm\_program

│ │ pack.asm

│ │

│ └─hazard\_test

│ branch\_hazard.asm

│ data\_hazard.asm

│ jump\_hazard.asm

│ no\_hazard.asm

│ sp.asm

│

└─src

├─constrs

│ PipelineCPU.xdc

│

├─sim

│ TestBench.v

│

└─source

ALU.v

BranchAndJumpHazard.v

Controller.v

DataHazard.v

DataMem.v

EX\_MEM\_Reg.v

ID\_EX\_Reg.v

IF\_ID\_Reg.v

ImmExtend.v

InstMem.v

MEM\_WB\_Reg.v

PC.v

PipelineCPU.v

RegFile.v

其中，各个目录及文件的内容如下：

1. asm\_program：存放汇编程序；
   1. pack.asm：背包问题汇编语言程序
   2. 测试冒险的汇编语言程序
2. src：存放源代码
   1. constrs：存放管脚约束文件
   2. sim：存放test bench文件
   3. source：存放处理器源代码

上述文件均已经与本报告一同上交。

# 七、心得体会

本次实验总体上看还是比较顺利的。

在这次实验中，我基本上完全独立完成了本次实验。在编写代码的过程中，我也对课上所讲的流水线有了更深的理解。在编写代码的过程中，我也遇到了一些困难，例如jr指令对数据的要求时间比较早，要求提前进行转发，这个转发的细节耗费了我一些精力来思考；此外分支和跳转指令由于不在同一阶段进行判断，因此它们的优先级先后问题也是我需要特别思考的。此外，Load/Use冒险的阻塞在之前的理论课上没有特别思考过如何实现，本次实现阻塞时也耗费了一些时间来思考。

另外，这次试验也有一些不足之处。首先，处理器的频率较低，没有进行更深入的优化；第二，指令存储器没有使用IP Core，因此在切换汇编程序等操作上耗费了一些时间；第三，一些细节上也编写的稍显繁琐，没有进行优化，等等。

总而言之，本次实验还是让我有很多收获的。最后感谢老师在这一学期的倾情讲授，感谢助教的悉心指导，没有老师和助教的帮助，我也难以得到这么多的收获。

# 附录

## 附录一 指令格式表



## 附录二 指令说明表

