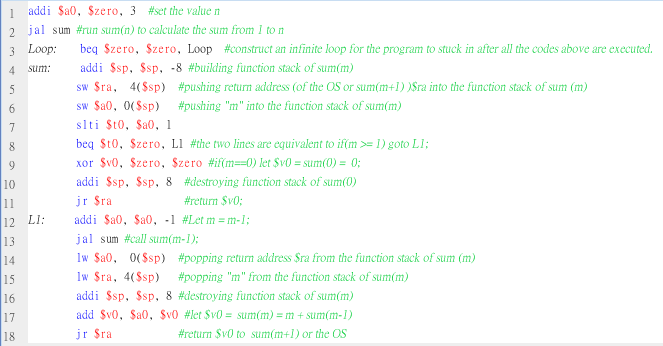
**数字逻辑与处理器基础实验报告**

学号：2017011090

班级：无78

姓名：游子權

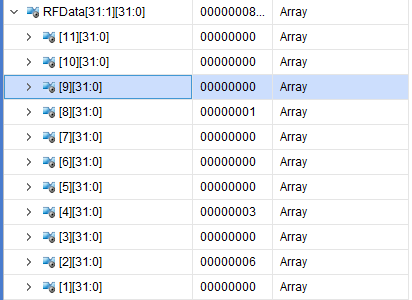
1. **实验名称：**流水线MIPS处理器
2. **实验目的：**于FPGA板上完成一基于MIPS指令集的五级流水线处理器，使其完成100个数据的排序功能。
3. **设计方案、原理说明与关键代码：**
   * 1. 设计原则
4. 分成五个阶段，分别为取指令（Instruction Fetch, IF）、解读指令（Instruction Decode, ID）、执行运算（Execution, EX）、存取资料记忆体（Memory Access, MEM）、写回寄存器堆（Write Back to register file, WB）。
5. 使用ROM作为Instruction Memory，采用于时钟上升沿读写的RAM作为Data Memory。采用于时钟上升沿写入的Register File。
6. 能采用转发（forwarding）时，采用完全的转发处理数据关联问题。
7. 不能转发时，采用尽量少的阻塞（stall）来解决竞争。
8. 分支（branch）、跳转（jump）类指令皆在ID阶段提前判断，在分支确认或跳转时取消IF阶段指令。
9. CPU设有七段数码管外设显示排序结果、设有定时器外设。
10. CPU可支援未知指令的异常与定时器中断。
    * 1. 系统框图
      2. 排序与中断汇编代码



**▲图 1：欲执行之程序及其注释**

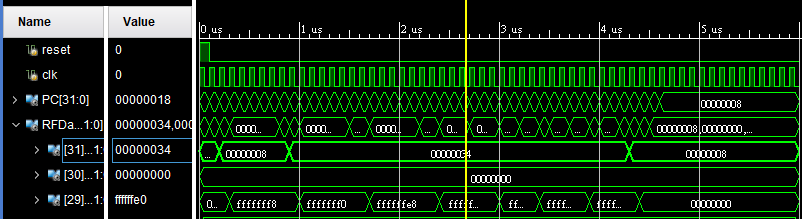
* 如果第一行的 3 是任意正整數n，這段程式可以輸出1+2+…+n的結果。
* 三段程式碼的功用分述如下：

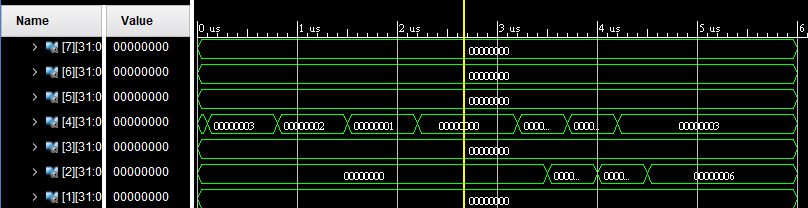
1. Loop（第3行）：構造無限迴圈，使得程序結束後進入此迴圈而使Program Counter不致越界。
2. sum（第4~11行）：構造sum(m)的function stack，並處理m==0時返回0之情形。
3. L1 （第12~18行）：是sum(m)中的分支，處理sum(m-1)的調用，並在sum(m-1)返回值後處理sum(m) = sum(m-1)+m 之情形。
4. **电路模拟情形：**
   1. 執行時間足夠長之後（如 5000ns 時），寄存器$a0（RFData[4]）的值即為第1行所給定的輸入n=3，$v0 （RFData[2]）的值即為1+2+…+n的結果，此例中為6。與預期結果相符。



**▲图 2：CPU模拟执行程序后各寄存器之终值**

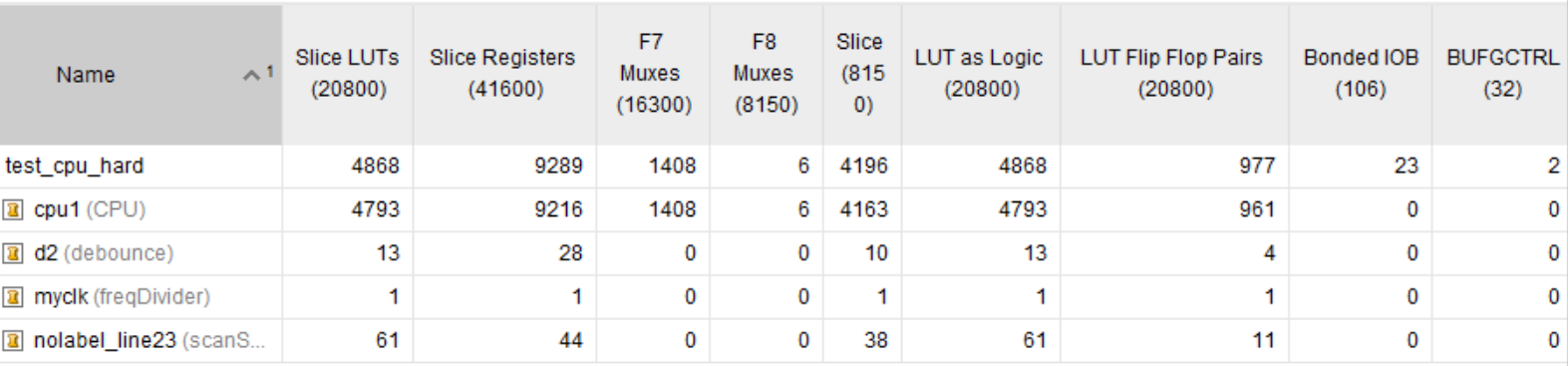
* 1. 觀察、描述並解釋 PC,$a0,$v0,$sp,$ra 如何變化。（波形檔案附於test\_cpu\_behav\_sum.wcfg中）
* PC是Program Counter，其永遠儲存目前執行的指令的地址。每一個時鐘上升沿變化，一般情形下，其值+4，遇jal, beq等跳轉指令時，依指令改寫PC至相應指令的地址。
* $a0傳入sum()函數的參數，在function stack的建構過程（每次jal前）中，其值自3遞減至0，在function stack解構（每次jr $ra前）過程中，其值自0遞增至3。
* $v0儲存函數返回值，每次函數返回（jr $ra）之前$v0將會更新。從模擬中可以看出，$v0按照0，1，1+2=3，1+2+3=6的規律變化。
* $sp是stack pointer。儲存stack頂端的地址。從模擬中可以看出，每一層函式呼叫，$sp遞減8（有新元素$a0和$ra入棧）。函數返回之前，$sp遞增8（$a0和$ra出棧）。
* $ra：在構建function stack過程中，主調函數必須保護現場，故使用jal儲存返回後的地址。在解構function stack過程中，會從stack讀出主調函數指定的返回地址至$ra並返回，故其在每次跳出時都會被重寫一次，惟本程式中jal儲存的位置僅有第4行或第13行，故$ra寄存器的值循0x00🡪0x08🡪0x34🡪0x08變化，最後停有0x08的無限迴圈中。

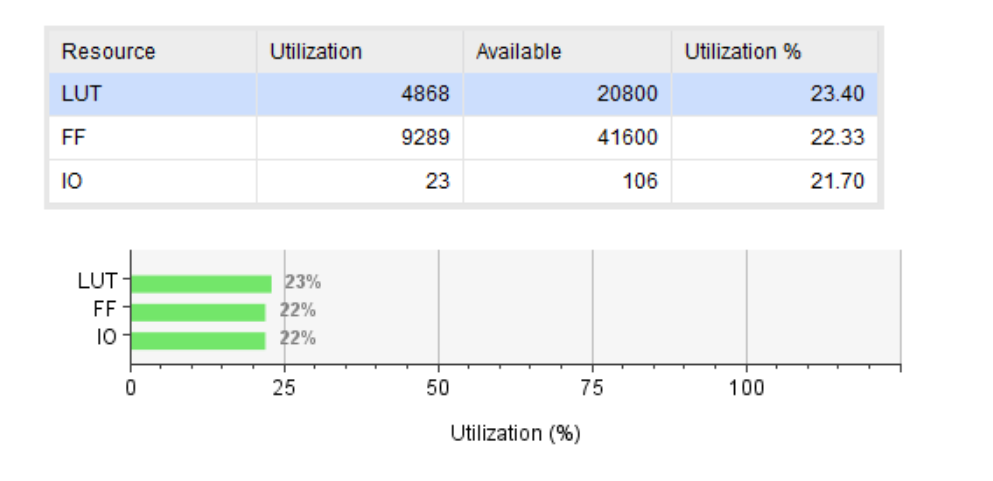




**▲图 3：CPU模拟执行程序后各寄存器之值之时序变化**

1. **性能评价：**
   * 1. 面积性能





**▲图4：面积资源使用情形**

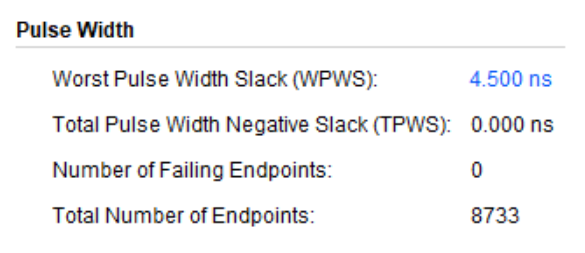
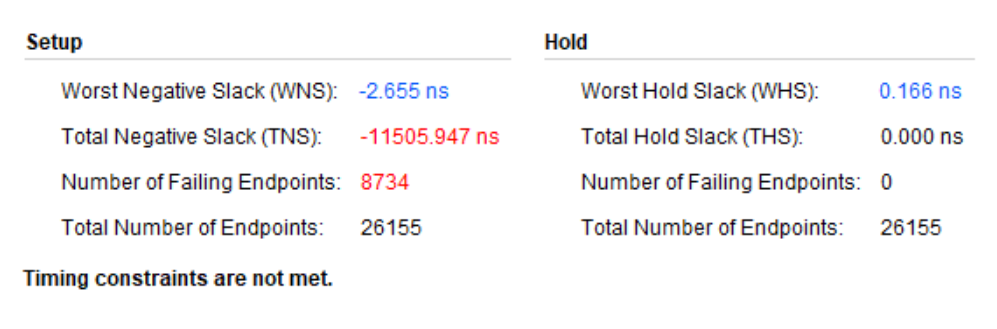
可见其用了4868个单位的查找表和9289个单位的触发器，占用了FPGA约1/5的资源。

* + 1. 时序性能：

原本的测试电路中因涉及寄存器值的扫描显示，而其中扫描显示又涉及分频，分频涉及计数器，计数器的全加器进位链无法有效优化，故成为关键路径。

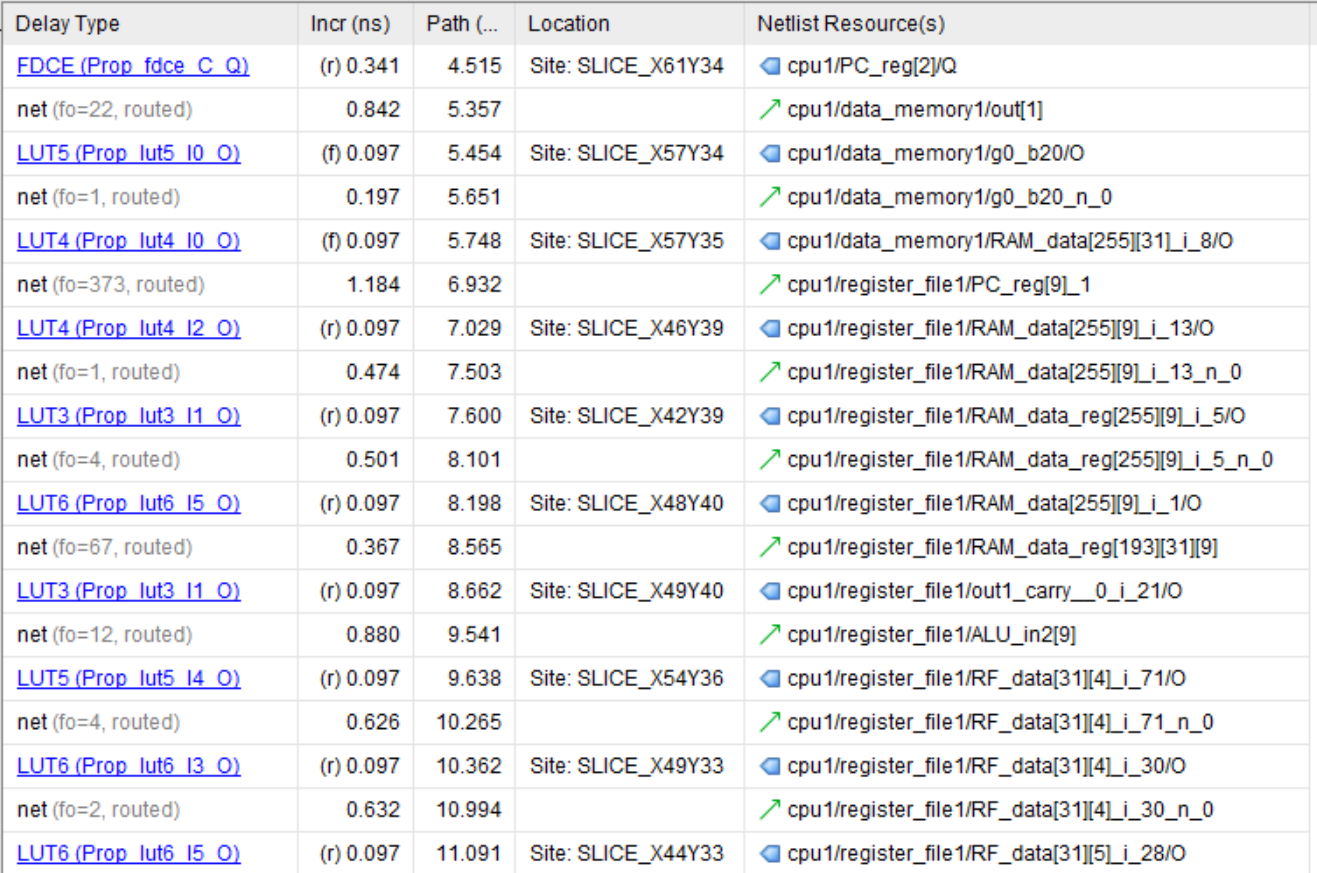
若我们将扫描显示的模块移除，在电路由100MHz时钟驱动的情形下（周期=10ns），其Setup时间的裕量为-2.655ns，故电路工作的最高主频为。

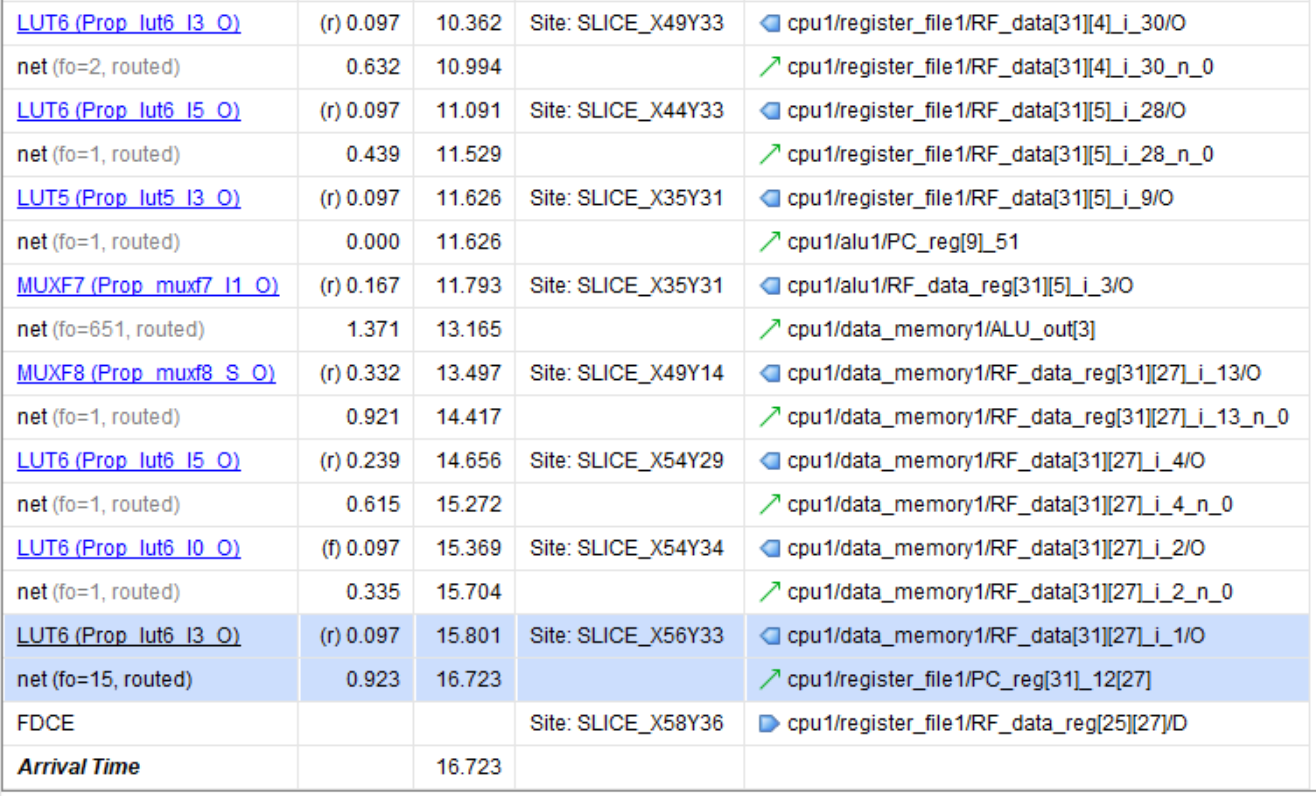
又我们是直接将100MHz时钟引入CPU模块，故CPU的最高主频。



**▲图5：时序性能情形**

其关键路径：





**▲图6：关键路径**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 单元名 | 类型 | 延时类型 | 延时值/ns | 累计延时/ns | 推测功能 |
| FDCE | D触发器 | Clock-to-Q | 1.183 | 5.357 | PC值更新 |
| LUT5 | 5输入查找表 | 组合逻辑延时 | 0.294 | 5.651 | Instruction Fetch |
| LUT4 | 4输入查找表 | 组合逻辑延时 | 1.281 | 6.932 |
| LUT4 | 4输入查找表 | 组合逻辑延时 | 0.571 | 7.503 | Instruction Decode：从Register取数 |
| LUT3 | 3输入查找表 | 组合逻辑延时 | 0.598 | 8.101 |
| LUT6 | 6输入查找表 | 组合逻辑延时 | 0.464 | 8.565 |
| LUT3 | 3输入查找表 | 组合逻辑延时 | 0.976 | 9.541 |
| LUT5 | 5输入查找表 | 组合逻辑延时 | 0.724 | 10.265 | Execution：运算出地址 |
| LUT6 | 6输入查找表 | 组合逻辑延时 | 0.729 | 10.994 |
| LUT6 | 6输入查找表 | 组合逻辑延时 | 0.535 | 11.529 |
| LUT5 | 5输入查找表 | 组合逻辑延时 | 0.097 | 11.626 |
| MUXF7 | 6输入多路选择器 | 组合逻辑延时 | 1.539 | 13.165 |
| MUXF8 | 7输入多路选择器 | 组合逻辑延时 | 1.252 | 14.417 | Memory Access：访问存储器 |
| LUT6 | 6输入查找表 | 组合逻辑延时 | 0.855 | 15.272 |
| LUT6 | 6输入查找表 | 组合逻辑延时 | 0.432 | 15.704 |
| LUT6 | 6输入查找表 | 组合逻辑延时 | 1.019 | 16.723 | Write Back：写回寄存器 |
| FDCE | D触发器 | Clock-to-Q |  |  |

**▲表 2：关键路径分析**

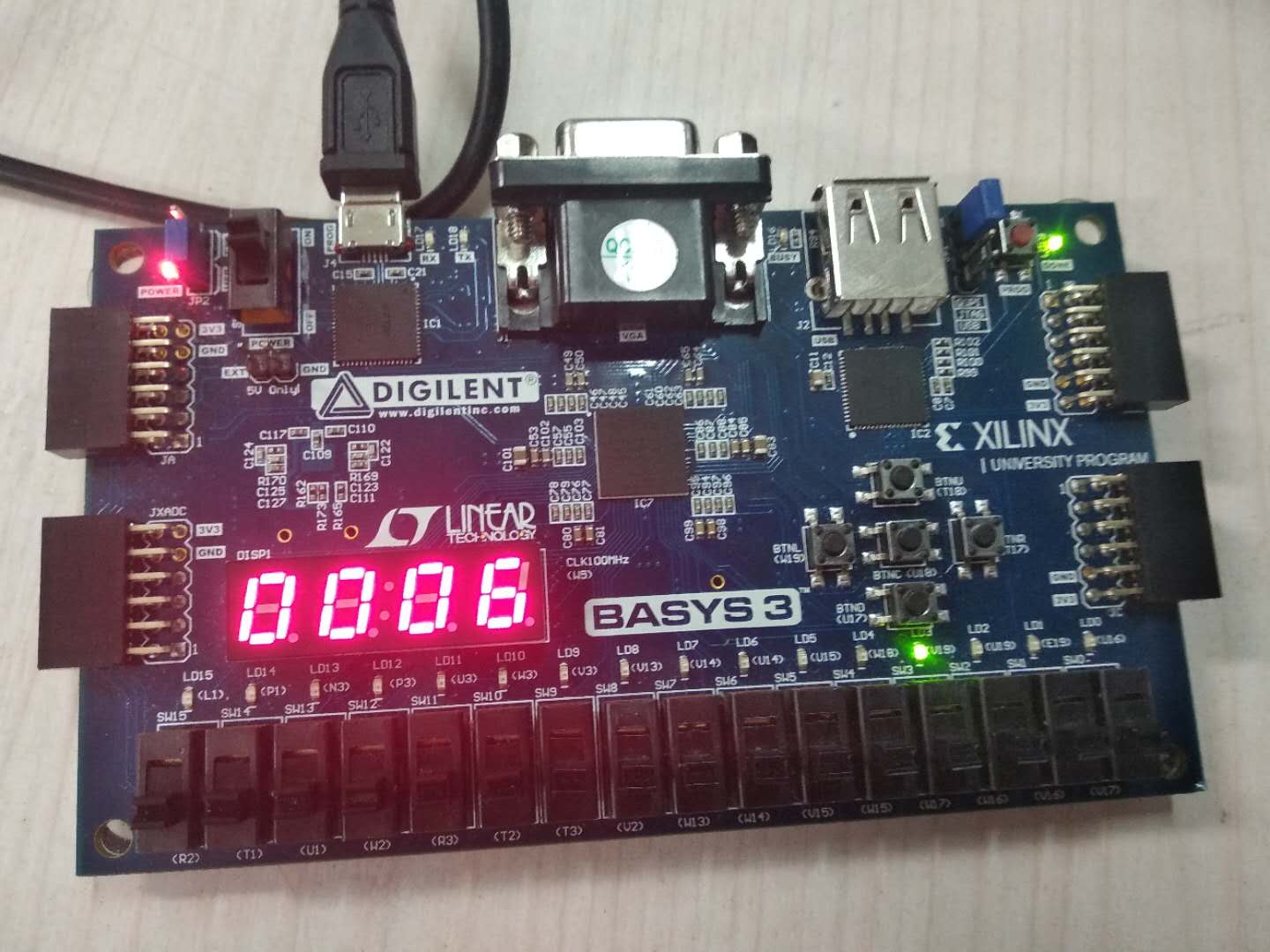
推测关键路径应为执行load word指令造成，此例中之路径应为自memory中载入返回地址并将其存入寄存器$ra中。

1. **硬件调试情形：**

本次硬件调试顺利成功，与预期结果相符。

* SW1和SW0用于选择在数码管上显示的寄存器序号：

|  |  |
| --- | --- |
| **SW[1:0]** | **寄存器** |
| 00 | $v0 |
| 01 | $a0 |
| 10 | $sp |
| 11 | $ra |



**▲图7：硬件调试情形**