MIPS 多周期处理器实验报告

林宝诚 2019080030 无 98

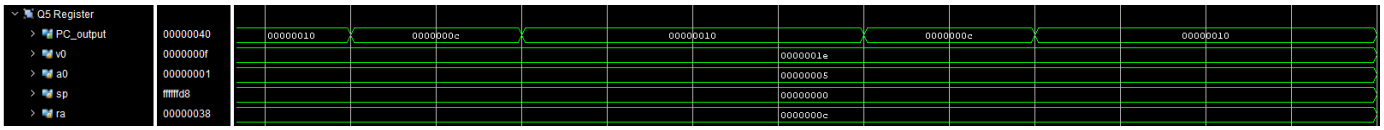
首先，理论课多周期 CPU 大作业 3.5 节中的汇编程序代码已提前翻译至机器码，并写入了指令储存器中。

```
RAM_data[8'd0] <= {6'h08, 5'd0 , 5'd4 , 16'h0005};    //addi
RAM_data[8'd1] <= {6'h00, 5'd0 , 5'd0 , 5'd2 , 5'd0, 6'h26};   //xor
RAM_data[8'd2] <= {6'h03, 26'd4};    //jal
RAM_data[8'd3] <= {6'h04, 5'd0 , 5'd0 , 16'hffff};    //beq
RAM_data[8'd4] <= {6'h08, 5'd29, 5'd29 , 16'hfff8}; //addi
RAM_data[8'd5] <= {6'h2b, 5'd29, 5'd31 , 16'h0004}; //sw
RAM_data[8'd6] <= {6'h2b, 5'd29, 5'd4 , 16'h0000}; //sw

RAM_data[8'd7] <= {6'h0a, 5'd4, 5'd8 , 16'h0001}; //slti
RAM_data[8'd8] <= {6'h04, 5'd8, 5'd0 , 16'h0002}; //beq
RAM_data[8'd9] <= {6'h08, 5'd29, 5'd29 , 16'h0008}; //addi
RAM_data[8'd10] <= {6'h00, 5'd31, 15'd0 , 6'h08}; //jr
RAM_data[8'd11] <= {6'h00, 5'd2, 5'd4, 5'd2, 5'd0, 6'h20}; //add
RAM_data[8'd12] <= {6'h08, 5'd4, 5'd4 , 16'hffff}; //addi
RAM_data[8'd13] <= {6'h03, 26'd4}; //jal
RAM_data[8'd14] <= {6'h23, 5'd29, 5'd4 , 16'h0000}; //lw
RAM_data[8'd15] <= {6'h23, 5'd29, 5'd31 , 16'h0004}; //lw
RAM_data[8'd16] <= {6'h08, 5'd29, 5'd29 , 16'h0008}; //addi
RAM_data[8'd17] <= {6'h00, 5'd4, 5'd2 , 5'd2, 5'd0, 6'h20}; //add
RAM_data[8'd18] <= {6'h00, 5'd31, 15'd0 , 6'h08}; //jr
```

图【1】汇编程序代码对应的机器码

将指令运行一段时间后，仿真结果如下：



图【2】指令运行仿真结果

仿真结果与设想中的程序结果一致，即个变量的最终值如下：

$v0：0x0000_001e=30

$a0：0x0000_0005=5

PC 输出：在 0x0000_0010 与 0x0000_000c 轮流交替（陷入了指令 3：beq 的无限循环，代表程序运行结束）

$sp：栈指针恢复为 0

$ra：返回地址为 0x0000_000c（即指令 3 的 beq，代表递归完成，程序运行结束）

完成仿真后，我将代码综合实现，并将代码录入 FPGA 板进行验证。代码在 FPGA 上的表现与仿真结果一样。如下为输入输出设置和其他实验设置：

外部输入 1：系统时钟 sysclk（100MHz）

外部输入 2：复位信号 BTNU

处理器时钟输入：10Hz 时钟（利用系统时钟分频）

七段数码管时钟输入：1kHz 时钟（利用系统时钟分频）

**注：处理器的时钟输入也能以用户输入 BTND 来实现，只是要进行按键消抖（debounce）

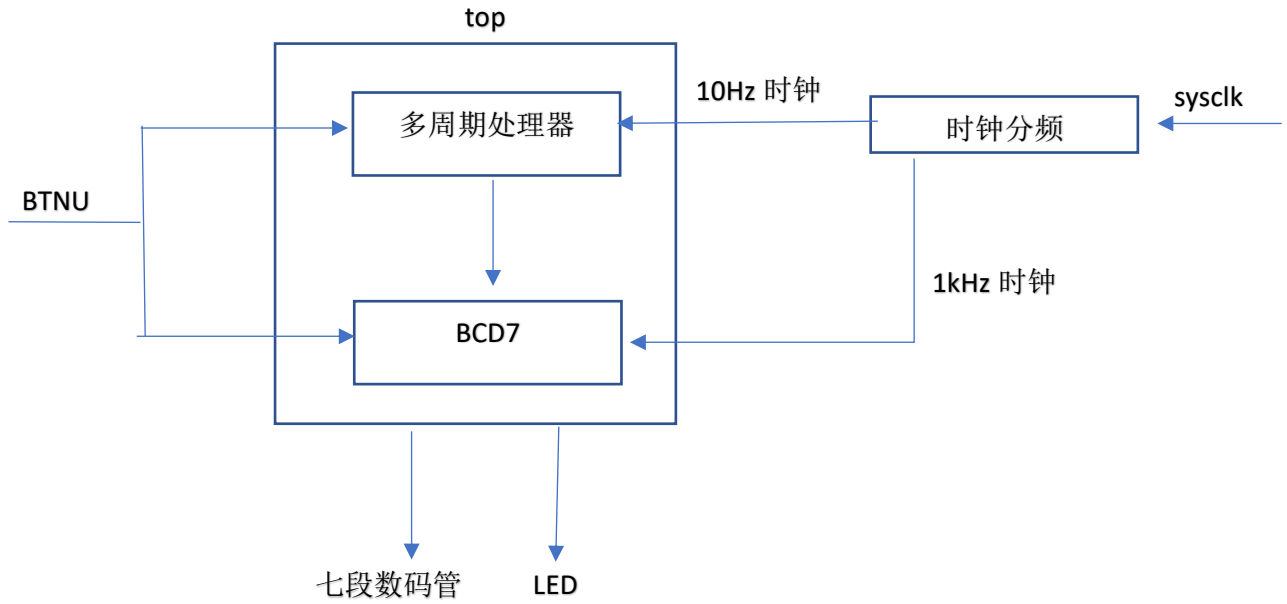
外部输出 1：8 颗 LED 指示灯（用以显示 PC 值的最低 8bits）

外部输出 2：七段数码管（用以显示$a0，$v0，$sp，$ra 的最低 16bits）

**由于七段数码管没办法同时输出所有 4 个变量，因此使用 2 个开关（SW0，SW1）控制当前的输出变量，逻辑如下：

SW0 \ SW1	0	1
0	$v0	$a0
1	$sp	$ra

总体设计图



FPGA 资源消耗情况

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)
top	4929	9440	1336	320
BCD7 (BCD7)	11	2	0	0
clk_gen (clk_gen)	22	18	0	0
clk_gen_10Hz (clk_ge...	28	33	0	0
MultiCycleCPU_1 (Mult...	4852	9387	1336	320

时序分析

假设输入（复位信号 reset）和输出（七段数码管和 LED）都没有延迟，MIPS 处理器时钟周期为 20ns

结果：

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	4.677 ns	Worst Hold Slack (WHS):	0.064 ns	Worst Pulse Width Slack (WPWS):	9.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	18763	Total Number of Endpoints:	18763	Total Number of Endpoints:	9505

All user specified timing constraints are met.

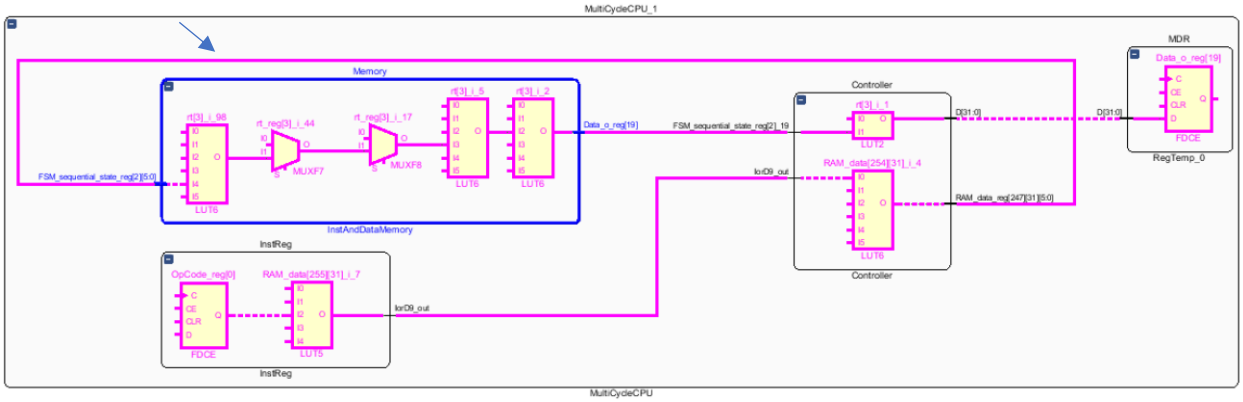
因此，最短时钟周期为 $20ns - 4.677ns = 15.323ns$，对应最高工作频率$\frac{1}{15.323ns} = 65.26MHz$。

Intra-Clock Paths - sysclk - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source
Path 1	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 2	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 3	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 4	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 5	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 6	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 7	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 8	4.677	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.194	1.933	13.261	20.000	sysclk
Path 9	4.684	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.187	1.928	13.259	20.000	sysclk
Path 10	4.684	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/MDR/Data_o_reg[19]/D	15.187	1.928	13.259	20.000	sysclk
Path 11	4.739	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/InstReg/rt_reg[3]/D	15.132	1.933	13.199	20.000	sysclk
Path 12	4.739	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/InstReg/rt_reg[3]/D	15.132	1.933	13.199	20.000	sysclk
Path 13	4.739	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/InstReg/rt_reg[3]/D	15.132	1.933	13.199	20.000	sysclk
Path 14	4.739	8	2206	MultiCycleCPU_1/InstReg/OpCode_reg[0]/C	MultiCycleCPU_1/InstReg/rt_reg[3]/D	15.132	1.933	13.199	20.000	sysclk

通过查询不同路径的 slack 情况，可以发现到限制工作频率的路径（slack 最低的路径）为从 InstReg（指令寄存器单元）到 MDR（储存数据寄存器单元）。

关键路径



因此，关键路径含有的单元为 InstReg（指令寄存器），Controller（控制器），Memory（指令与数据储存器），MDR（储存数据寄存器）。

详细数据通路如下：

Data Path

Delay Type	Incr (ns)	Path (...	Location	Netlist Resource(s)
FDCE (Prop_fdce_C_Q)	(r) 0.456	5.528	Site: SLICE_X40Y61	MultiCycleCPU_1/InstReg/OpCode_reg[0]/Q
net (fo=21, routed)	1.073	6.601		MultiCycleCPU_1/InstReg/Q[0]
			Site: SLICE_X40Y61	MultiCycleCPU_1/InstReg/RAM_data[255][31]_i_7/I2
LUT5 (Prop_lut5_I2_O)	(r) 0.152	6.753	Site: SLICE_X40Y61	MultiCycleCPU_1/InstReg/RAM_data[255][31]_i_7/O
net (fo=18, routed)	1.154	7.907		MultiCycleCPU_1/Controller/IorD9_out
			Site: SLICE_X46Y48	MultiCycleCPU_1/Controller/RAM_data[254][31]_i_4/I0
LUT6 (Prop_lut6_I0_O)	(r) 0.332	8.239	Site: SLICE_X46Y48	MultiCycleCPU_1/Controller/RAM_data[254][31]_i_4/O
net (fo=2206, routed)	5.287	13.526		MultiCycleCPU_1/Memory/FSM_sequential_state_reg[2][0]
			Site: SLICE_X6Y132	MultiCycleCPU_1/Memory/rt[3]_i_98/I4
LUT6 (Prop_lut6_I4_O)	(r) 0.124	13.650	Site: SLICE_X6Y132	MultiCycleCPU_1/Memory/rt[3]_i_98/O
net (fo=1, routed)	0.000	13.650		MultiCycleCPU_1/Memory/rt[3]_i_98_n_0
			Site: SLICE_X6Y132	MultiCycleCPU_1/Memory/rt_reg[3]_i_44/I1
MUXF7 (Prop_uxf7_I1_O)	(r) 0.214	13.864	Site: SLICE_X6Y132	MultiCycleCPU_1/Memory/rt_reg[3]_i_44/O
net (fo=1, routed)	0.000	13.864		MultiCycleCPU_1/Memory/rt_reg[3]_i_17/I1
			Site: SLICE_X6Y132	MultiCycleCPU_1/Memory/rt_reg[3]_i_17/O
MUXF8 (Prop_uxf8_I1_O)	(r) 0.088	13.952	Site: SLICE_X6Y132	MultiCycleCPU_1/Memory/rt_reg[3]_i_17_n_0
net (fo=1, routed)	1.204	15.155		MultiCycleCPU_1/Memory/rt[3]_i_5/I3
			Site: SLICE_X17Y128	MultiCycleCPU_1/Memory/rt[3]_i_5/O
LUT6 (Prop_lut6_I3_O)	(r) 0.319	15.474	Site: SLICE_X17Y128	MultiCycleCPU_1/Memory/rt[3]_i_5_n_0
net (fo=1, routed)	2.416	17.890		MultiCycleCPU_1/Memory/rt[3]_i_2/I2
			Site: SLICE_X37Y92	MultiCycleCPU_1/Memory/rt[3]_i_2/O
LUT6 (Prop_lut6_I2_O)	(r) 0.124	18.014	Site: SLICE_X37Y92	MultiCycleCPU_1/Controller/FSM_sequential_state_reg[2]_19
net (fo=1, routed)	1.215	19.229		MultiCycleCPU_1/Controller/rt[3]_i_1/I1
			Site: SLICE_X37Y72	MultiCycleCPU_1/Controller/rt[3]_i_1/O
LUT2 (Prop_lut2_I1_O)	(r) 0.124	19.353	Site: SLICE_X37Y72	MultiCycleCPU_1/MDR/D[19]
net (fo=2, routed)	0.913	20.266		MultiCycleCPU_1/MDR/Data_o_reg[19]/D
FDCE			Site: SLICE_X41Y63	
Arrival Time		20.266		

MultiCycleCPU_1/Memory/rt_reg[3]_i_44/O

*注：下面单元耗时的计算仅考虑逻辑耗时和单元内部的路径耗时，不考虑从一个单元到另一个单元的路径延时

InstReg 单元耗时：$0.456 + 1.073 + 0.152 = 1.681ns$

Controller 单元耗时：$0.332 + 0.124 = 0.456ns$

Memory 单元耗时：$0.124 + 0.214 + 0.088 + 1.204 + 0.319 + 2.416 + 0.124 = 4.489ns$

因此，**Memory 单元**是最耗时的。

另外，单元间耗时最长的路径为从 controller 到 Memory（见图中箭头），延时为 5.287ns。

遇到的问题与解决方案

问题 1：使用 FPGA 板的代码执行结果有错误，与仿真结果不同。

解决方案：由于仿真结果正确且综合实现都能没有出现运行错误的问题，我一开始认为使用 FPGA 板进行验证会很顺利，没想到，将代码录入板子后，程序没有正常运行，输出也不是我所预料的。当时，我还没整理控制器的代码（代码很乱，可能会存在一些问题），因此我决定先将控制器的代码整理好，再进行调试已解决仿真和上板验证结果不一致的问题。果不其然，我在整理代码中发现，我将一些非阻塞赋值符号“<=”错写成了“=”，后者为阻塞赋值符号，使得我有一部分程序出现异常。完成控制器代码的整理后，板子上的程序运行结果就与我设想中的一致了。

问题 2：代码中出现锁存器（Latch）（在综合的 Warning 中有提及）。

我有了解到在进行电路设计中应该尽可能避免使用锁存器，因为锁存器属于异步器件，稍有不慎可能会产生无法预料的错误。根据提示，出现锁存器的部分是在决定控制信号的组合逻辑中，因为对于每个不同情况（状态和指令类型），我并未完整地将所有信号的值列出，仅列出了关键信号。为了避免锁存器的出现，我只能将不同情况中的控制信号写全（即使控制信号与上一个状态是一样的）。我个人认为以后也要养成将所有可能性写完整的习惯，因为这不仅可以避免难以使用的锁存器出现，也能使代码可容易看懂。

感想：

经过了一学期的数字逻辑与处理器实验，我觉得这门课不仅训练了我的动手能力，也加深了我对理论知识的掌握。每一次自主动手完成实验后（尤其是较耗时的多周期处理器），我都能够感受到满满的成就感，之前调试代码时的烦躁顿时一扫而空。另外，值得一提的是，这次多周期处理器的设计帮助我复习了许多期末考试的内容，减轻了期末复习的压力。

针对课程内容，在这里我想表达对老师和助教的感谢，因为每一次的实验都挺有趣的（虽然一开始真的觉得很难，可是克服后感觉自己又成长了）。另外，由于我这学期无法返校，只能通过线上进行上课和验收，因此我也想感谢老师和助教对于线上学生的体谅，让我有办法在线上完成验收，顺利跟着大队完成这学期的实验课。当然，这门课还没结束（还剩下一个实验需要在暑假完成），希望到时能够准时完成实验，顺利为这门实验课划下句点吧！