本设计频率至少可达 102.74MHz（周期 10ns，余量 0.266ns）。

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.266 ns	Worst Hold Slack (WHS):	0.094 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	7082	Total Number of Endpoints:	7082	Total Number of Endpoints:	3714

All user specified timing constraints are met.

支持 30 余条算术逻辑、访存与分支跳转指令。采用经典的五级流水，可很方便地添加级数和指令逻辑。实现的指令包括：
空指令：nop (0x00000000,即 sll $0,$0,0)
存储访问指令：lw, sw, lui
算术指令：add, addu, sub, subu, addi, addiu
逻辑指令：and, or, xor, nor, andi, sll, srl, sra, slt, sltu, sltiu
分支指令（beq、bne、blez、bgtz、bltz)和 跳转指令(j、jal、jr(前一条指令不可数据冒险，否则在汇编加 nop)、jalr)；

本设计采用完全的 forwarding 电路解决数据关联问题。对于 Load-use 类竞争采取阻塞一个周期+Forwarding 的方法解决。
对于分支指令在 EX 阶段判断，在分支发生时刻取消 ID 和 IF 阶段的两条指令。经测试比在 ID 阶段判断的频率更高。
对于 J 类指令在 ID 阶段判断，并取消 IF 阶段指令。
通过在寄存器堆内部加入数据旁路的硬件方式实现寄存器实现先写后读功能，实现了寄存器堆的优化，比旁路转发的方式频率更快。
通过在汇编层面加 nop 解决 jr 的数据冒险问题。

本设计实现了延迟槽设计，即跳转和分支指令后一条指令（延迟槽）的执行与跳转和分支是否发生无关。
本设计实现了串口功能，使用两块 RAM 作为指令内存和数据内存，初始值通过串口传输进内存。支持规定外设地址的读写。

本设计在汇编指令层面也进行了优化，特别是串口发送和接受的汇编指令，如对关键函数（即循环部分）进行的优化和减少指令数，以及尽量将 If 语句转化为 case 语句。
本设计因为频率超过了系统时钟的 100MHz，所以无需设计分频器，也能更好地优化频率。