# 基于 FPGA 的 RISC\_CPU 设计

# 1、 设计方案

RISC 指的是精简指令集计算机,可以将其分成 8 个基本部件来考虑: 时钟 发生器,指令寄存器,累加器,算术逻辑运算单元,数据控制器,状态控制器, 程序计数器,地址多路器。在本设计中,为了简化设计,降低设计的复杂度, 并没有采用多级流 水线形式来设计,而是采用了状态机的方法。取指,译码, 执行,访问,写回分别在几个周期内完成,等 完成该指令的操作后,在读取下

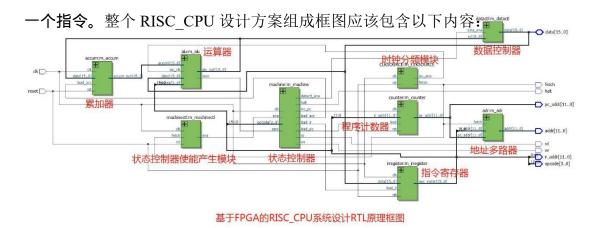


图 1 基于 FPGA 的 RISC CPU 架构组成原理图

从上述架构图可以看出,RICS\_CPU 结构比较复杂,但是它的基本部件并不复杂,因此整个方案设计可以从它的 8 个基本组成部分来考虑:

- (1) 时钟发生器
- (2) 指令寄存器
- (3) 累加器
- (4) 算术逻辑运算单元
- (5) 数据控制器
- (6) 状态控制器
- (7) 程序计数器

#### (8) 地址多路器

#### 1.1 时钟发生器

该模块的设计主要是利用外部时钟信号 clk 来产生一系列时钟信号,并且可以送到 CPU 的其它各个部件中。其中,fetch 是控制信号,clk 的 6 分频信号。 当 fetch 高电平时,使 clk 能触发 cpu 控制器开始执行一条指令;同时 fetch 信号还将控制地址多路器输出指令地址和数据地址。clk 信号还用作指令寄存器,累加器,状态控制器的时钟信号。

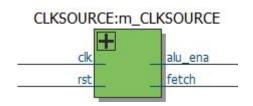


图 2 时钟发生器原理图

### 1.2 指令寄存器

指令寄存器的触发信号时 clk,在 clk 的正沿触发下,寄存器将数据总线送来的指令存入 16 位的寄存器中,但并不是每个 clk 的上升沿都寄存数据总线的数据,因为数据总线上有时传输指令,有时传输数据。什么时候寄存,什么时候不寄存由 CPU 状态控制器的 load\_ir 信号控制。load\_ir 信号通过 load\_ir 口输入到指令寄存器,复位后,指令寄存器被清为零。

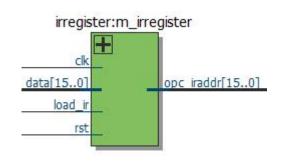


图 3 指令寄存器原理图

#### 1.3 累加器

累加器用于存放当前的结果,它也是双目运算中的一个数据来源。复位后, 累加器的值是零。当累加器通过 load\_acc 信号时,在 clk 时钟跳变沿时就受到来 自于数据总线的数据。

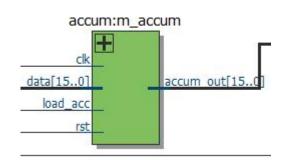


图 4 累加器原理图

### 1.4 算术运算器

算术逻辑运算单元,它根据输入的 16 种不同的操作码分别进行加减乘,与 或非等基本操作运算,利用这几种基本运算可以实现很多种其它运算以及逻辑判 断等操作。

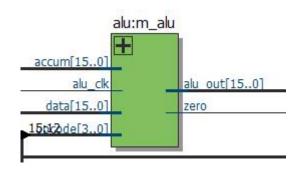


图 5 算术运算器原理图

#### 1.5 数据控制器

数据控制器,其作用是控制累加器的数据输出,由于数据总线是各种操作时 传送数据的公共通道,不同情况下传送不同的内容。有时要传输指令,有时要传 送 RAM 区或接口的数据。累加器的数据只有在需要往 RAM 区域或端口写时才 允许输出,否则应呈现高阻态,以允许其它部件使用数据总线。所以任何部件往 总线上输出数据时,都需要一控制信号。而此控制信号的启停则由 cpu 状态控制 器输出各信号控制决定。数据控制器何时输出累加器的数据则由状态控制器输出的控制信号 data ena 决定。

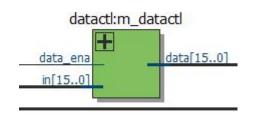


图 6 数据控制器原理图

### 1.6 地址多路器

地址多路器,它用于选择输出的地址是 PC 地址还是数据/端口地址。每个指令周期的前 3 个时钟周期用于从 ROM 中读取指令,输出的应是 PC 地址,后 3 个时钟周期用于 RAM 或端口的读写,该地址有指令给出。地址的选择输出信号由时钟信号的 6 分频 fetch 提供。

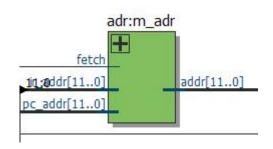


图 7 地址多路器原理图

#### 1.7 程序计数器

它用于提供指令地址,以便读取指令。指令按地址顺序存放在存储器中。有两种途径可形成指令地址;其一是顺序执行的情况,其二是遇到要改变顺序执行程序的情况,例如执行 JMP 指令后,需要形成新的地址。

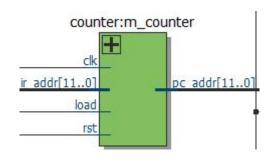


图 8 程序计数器原理图

## 1.8 状态控制器

状态机控制器接收复位信号 reset, 当 reset 有效时,通过信号 ena 使其为零,输入到状态机中,以停止状态机的工作。状态机是 cpu 的控制核心,用于产生一系列的控制信号,启动或停止某些部件。cpu 何时进行指令来读写 I/O 端口及 RAM 区等操作,都是由状态机来控制的。状态机的当前状态,由变量 state 记录, state 的值就是当前这个指令周期中已过的时钟数。

指令周期是有6个时钟周期组成,每个时钟周期都要完成固定的操作,即

- (1) 第 0 个时钟, cpu 状态控制器的输出 rd, data\_ctl 和 load\_ir 为高电平,inc\_pc 从 0 变为 1 故 pc 加 1, ROM 送来的指令代码寄存在指令寄存器中。
- (2) 第1个时钟空操作
- (3) 第2个时钟。若操作符为HLT,则输出信号HLT为高。如果操作符不为HLT,除了PC增1外,其他各控制线输出为零。
- (4)第3个时钟,若操作符为AND,ADD,XOR,LDA,NOT,MUL,SUB,OR,RL,RR,POP,PUSH,读取相应地址的数据;若为STO,输出累加器数据。
- (5) 第 4 个时钟, 若操作符为 AND, ADD 等算术运算, 算术运算器就进行相应的运算, 若操作符为 LDA, 就把数据通过算术运算符送给累加器, 若为 SKZ, 先判断累加器的值是否为 0, 如果为 0, PC 加 1, 否则保持原值; 若为 JMP, 锁存目标地址, 若为 STO, 将数据写入地址处。
- (6) 第5个时钟空操作

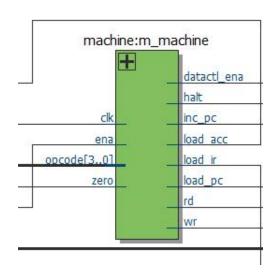


图 9 状态控制器原理图

# 2、 仿真结果

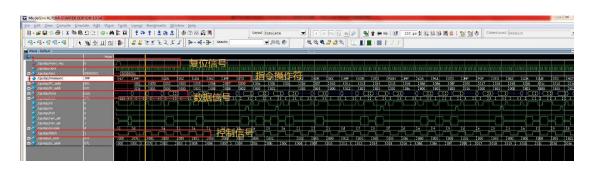


图 10 RISC\_CPU 顶层系统级仿真及部分信号说明

	17aga																				
/cputop/reset_req	0	5	_		高电	评复	1V														
/cputop/test	00000001	=		(0000000				_			_=	_=		==							
/cputop/mnemoric	JMP		HLT						IMP												LDA
/cputop/PC_addr	001								(001						07:				_		100
/putop/TR_addr	07c		_						07c						)(00)						180
/cputop/data	2222							(60/6						(d001						(58)0	
/cputop/addr	000		(000			CDH		5 / HOL	翻复	TAGET	拉结	拟置	1000	07c		001			(000	(001	
/cputop/rd	0					CFU	K A LI In c	3) I J E	百日リタ	עו או	四旧	HP I	730								
/cputop/vir	0																				
/cputop/halt	0				-					-											
/cputop/ram_sel	0							_													
/cputop/rom_sel	1																				
/cputop/opcode	d		0		-				14												à
/cputop/fetch			000			_			107:												900
/cputop/ir_addr	076	FREE		_	-	_	-		10/6	-					001		=				900
/cputop/pc_addr	001	(000			_			1001	_	_		).07c		070				001		0002	

图 11 RISC CPU 复位及内部寄存器初始值仿真

仿真结果说明: RISC\_CPU 的复位和启动操作是通过 reset 引脚的信号触发执行。当 reset 信号一进入高电平,RISC\_CPU 就会结束先行操作,并且只要 reset 停留在高电平状态,cpu 就维持在复位状态。在复位状态,cpu 各个内部寄存器都被设有初值,全部为零。数据总线为高阻态,地址总线为 000H,所有控制信号均为无效状态,reset 回到低电平后,接着到来的第一个 fetch 上升沿奖启动

RISC\_CPU 开始工作,从 ROM 的 000 处开始读取指令并执行相应操作,波形见图 11。

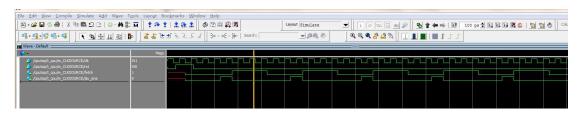


图 12 RISC CPU 时钟分频模块仿真-6 分频

	Mag	1									-												
/ /cputop/reset_req								-															
/cputop/dodx			$\neg$								$\neg$									$\neg$			
	00000001	00000103																					
/cputop/mnerionic		JMP						LDA						\$42						LDA			
/cputop/FC_addr	002	001	[070					(002						- 0						(005			
/cputop/IR_eddr /coutop/deta	800	07c	1001					X800						0	が 合い	数据				X501			
	a900	Ø001					4800	=		(0000		(6300			刊川口	1女人]/广			3801		-	0009	
/cputop/eddr /cputop/id	001	107c		),001		[000	X001	_	1820			(001	1002		(100			1003	X004		)821		
/aputopjid /aputopjid		_												-	CDIII	/-b- F			- > L				
(cputop/halt	l'														CPU.	写信号		以半维	ヨ父▽				
/coutop/ram_sel															1	7111			1000				
[cputop[rom_sel																					=		
/cputop/spcode		d						8						1									
/cputop/fetch				$\neg$																			
/cputop/r_addr	800		011					800						010						801			
/cpx/top/pc_nddr	002	(074			1001		(002						1003				004		(005				

图 13 CPU 总线写操作—RISC\_CPU 写入数据时序仿真

	Msg:																	
/ /cputop/reset_req / /cputop/clock //cputop/test //cputop/mnemonic	0 1 00000001 LDA	нт	(0000000				JMP											LDA
- /cputop/PC_addr - /cputop/IR_addr - /cputop/data - /cputop/addr - /cputop/rd	002 800 2227 800 0	(000			Χfff	(807		<sup>®</sup> 数据	送到	数据	总线	(d001 (07c	),07 ),00	) 1 ) (001		X000	(a800 )001	
<ul> <li>/cputop/wr</li> <li>/cputop/hait</li> <li>/cputop/ram_sel</li> <li>/cputop/rom_sel</li> </ul>	0 0 1 0	<u> </u>			<u>_</u> [_			CPU	J读信	号高	高有效	ζ						
3-4 /cputop/opcode 4 /cputop/fetch 3-4 /cputop/ir_addr	800	000					107c						001					a 800
3-4 /cputop/pc_addr	002	 000				1001			总约	輸出	<b>BPC</b> t	也让			(001		(002	

图 14 RISC\_CPU 读取数据时序仿真

仿真结果说明:每个指令周期的前 0~2 个时钟周期用于读指令;第 2.5 个周期处,存储器或端口地址就输入到地址总线上;第 3~5 个时钟周期,读信号 rd 有效,数据送到数据总线上,以便累加器锁存,或参与算术,逻辑运算。



图 15 RISC CPU 指令寄存器模块仿真



图 16 RISC\_CPU 运算器模块仿真