**CPU课程设计报告**

04017437 乔竞

# 一、课程目的

* 设计并仿真一个简易的微程序中心处理单元（Microgrammed Central Processing Unit, CPU），作为计算机系统的运算和控制核心，是信息处理、程序运行的最终执行单元；
* 进一步巩固计算机结构课程中理论部分的知识，加深对 CPU 的构成和原理的理解；
* 学习并掌握使用 XILINX 的 Vivado 软件进行 FPGA 的设计和仿真，提高分析和编程能力。

# 二、设计方案

## 2.1 指令集

此CPU设计中，使用的是单地址的指令格式。一条指令包含两个字段：操作码（operation code, opecode）和地址（address），前者定义了指令的操作和功能，后者则取决于指令的取址方式。若是直接取址，地址字段对应操作数在存储器中的地址；若是立即数取址方式，地址字段则直接说明了操作数的内容。这里规定除了JUMP指令使用立即数取址的方式外，其他指令均使用直接取址的方式。

存储器的大小定义为256×16，则指令字共16bit，其中操作码和地址各占8bit的内容，如图1所示。



图1 指令格式

涉及到的所有指令及其对应的操作码如表1所示。其中，[X]表示存储器中地址为X的内容，例如指令字00000011101110012 (03B916) 中，操作码字段为00000011即ADD操作，地址字段为10111001，表示CPU需要将存储器中地址为10111001的内容加到ACC中。又如指令字00000101000001112 (050716) 中，操作码字段为00000101即JMPGEZ操作，地址字段为00000111，表示此时若ACC中的数为正（即符号位为0），则令PC的值为00000111，否则（符号位为1）令PC加一。

表1 指令及其操作码

|  |  |  |
| --- | --- | --- |
| **Instruction** | **opcode** | **Comments** |
| STORE X | 00000001 | ACC→[X] |
| LOAD X | 00000010 | [X] →ACC |
| ADD X | 00000011 | ACC+[X] →ACC |
| SUB X | 00000100 | ACC-[X] →ACC |
| JMPGEZ X | 00000101 | If ACC≥0 then X→PC else PC+1→PC |
| JMP X | 00000110 | X→PC |
| HALT | 00000111 | Halt a program |
|  |  |  |
| MPY X | 00001000 | ACC×[X] →ACC,MR |
|  |  |  |
| AND X | 00001010 | ACC and [X] →ACC |
| OR X | 00001011 | ACC or [X] →ACC |
| NOT X | 00001100 | NOT [X] →ACC |
| SHIFTRL | 00001101 | Shift ACC to right 1bit, logic shift |
| SHIFTLL | 00001110 | Shift ACC to left 1bit, logic shift |
| SHIFTRA | 00001111 | Shift ACC to right 1bit, arithmetic shift |
| SHIFTLA | 00010000 | Shift ACC to right 1bit, arithmetic shift |

## 2.2 内部寄存器和存储器

CPU内部的各个寄存器和存储器都在设计中作为一个个单独的模块工作，最后再例化联系在一起，功能模块的结构如图2所示。



图2 内部结构图

**MAR (Memory Address Register)**

MAR用于存放从存储器中读取或写入存储器的地址，因此具有8bits以寻址Memory中共256个地址。输入输出端口如表2所示。

表2 MAR输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | addr |
| Rst |
| Control\_signal |
| Data\_from\_mbr |
| Data\_from\_pc |

**MBR (Memory Buffer Register)**

MBR用于存放从存储器中读取的内容或即将写入存储器的内容，具有16bits的长度。输入输出端口如表3所示。

表3 MBR输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Data\_to\_memory |
| Rst | Data\_to\_pc |
| Control\_signal | Data\_to\_mar |
| Data\_from\_memory | Data\_to\_ir |
| Data\_from\_pc | Data\_to\_br |
| Data\_from\_mr |
| Data\_from\_acc |

**PC (Program Counter)**

PC用于存放下一条指令的地址，长度为8bits。输入输出端口如表4所示。

表4 PC输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Data\_to\_mbr |
| Rst | Data\_to\_mar |
| Control\_signal |
| Data\_from\_mbr |

**IR (Instruction Register)**

IR存放一条指令的操作码，长度为8bits。输入输出端口如表5所示。

表5 IR输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Data\_to\_cu |
| Rst |
| Control\_signal |

**BR (Buffer Register)**

BR用于ALU的一个输入，存放给ALU的操作数，长度为16bits。输入输出端口如表6所示。

表6 BR输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | BRtoALU |
| Rst |
| Control\_signal |
| MBRtoBR |

**ACC (Accumulator)**

ACC用于ALU的另一个输入，且存放ALU的计算结果，长度为16bits。输入输出端口如表7所示。

表7 ACC输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Acc\_out |
| Rst |
| Control\_signal |
| Acc\_in |

**MR (Multiplier Register)**

MR存放乘法指令得到的结果的高16位，由于乘法操作得到的是32位的乘积，因此低16位存入ACC，高16位存入的MR。输入输出端口如表8所示。

表8 MR输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Mr\_out |
| Rst |
| Control\_signal |
| Mr\_in |

**LPM\_RAM\_DQ**

该RAM即作为存储器存放指令和操作数，大小为256×16，具有独立的输入和输出端口。输入输出端口如表9所示。

表9 RAM输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Data\_out |
| Rst |
| Data\_in |
| Addr |
| Control\_signal |

**ALU**

ALU是算术逻辑单元，它用于完成基本的算数和逻辑运算。输入输出端口如表10所示。

表10 ALU输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | ALUtoACC |
| Rst | ALUtoMR |
| Control\_signal | flag |
| BRtoALU |
| ACCtoALU |

ALU必须包含的操作如表11所示。

表11 ALU操作

|  |  |
| --- | --- |
| Operations | Explanations |
| ADD | ACC⇐ACC + BR |
| SUB | ACC⇐ACC - BR |
| AND | ACC⇐ACC and BR |
| OR | ACC⇐ACC or BR |
| NOT | ACC⇐Not ACC |
| SRL | ACC⇐Shift ACC to Right 1 bit Logic |
| SLL | ACC⇐Shift ACC to Left 1 bit Logic |
| SRA | ACC⇐Shift ACC to Right 1 bit Arithmetic |
| SLA | ACC⇐Shift ACC to Left 1 bit Arithmetic |

**Microprogrammed Control Unit**

控制单元是CPU中的核心部分，控制单元中的存储器存放了所需的所有微指令，它们可以产生控制信号以执行各种指令。控制单元的微结构如图3所示。我们在设计时没有完全按照图3的各个结构进行，而是作了简化，只保留CAR和Control Memory，其中Control Memory使用语言直接编写，而未用ROM完成。输入输出端口如表12所示。

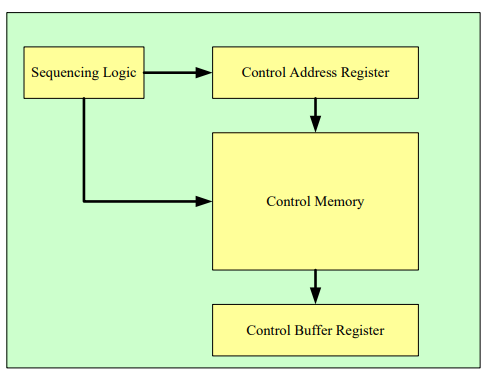


图3 控制单元微结构

表12 CU输入输出端口

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| Clk | Control\_signal |
| Rst |
| Data\_from\_ir |
| flags |

## 2.3 控制信号和微指令

此设计中用到的所有控制信号如表13所示，将它们组合成同一个32位的control\_signal信号来表征。

表13 控制信号表

|  |  |  |
| --- | --- | --- |
| Bit in Control Memory | Micro-operation | Comments |
| C0 | Memory<=MAR |  |
| C1 | MBR<=PC |  |
| C2 | MAR<=PC | Copy PC value to MAR for next address |
| C3 | PC<=MBR | MBR to PC |
| C4 | IR<=MBR | Copy MBR [15..8] to IR for OPCODE |
| C5 | MBR<=memory | Read from memory to MBR |
| C6 | BR<=MBR | Copy MBR data to BR for buffer to ALU |
| C7 | ALU<=ACC |  |
| C8 | MAR<=MBR | Copy MBR[7..0] to MAR for address |
| C9 |  |  |
| C10 | ACC<=MBR |  |
| C11 | MBR<=ACC | ACC to MBR |
| C12 | Memory<=MBR | Write MBR to memory |
| C13 | Control Unit<=IR |  |
| C14 | ALU<=BR |  |
| C15 | MBR<=MR |  |
| C16 |  |  |
| C17 | CAR<=CAR+1 | Increase CAR |
| C18 | CAR<=\*\*\* | Control Address Redirection, depends on the position of microinstruction |
| C19 | CAR<=0 | Reset Control Address to zero position |
| C20 | PC<=PC+1 | Increase PC |
| C21 | ACC<=0 | Reset ACC register to zero |
| C22 | ACC<=ACC+BR | Add BR to ACC |
| C23 | ACC<=ACC-BR | Sub BR to ACC |
| C24 | ACC<=ACC and BR | And |
| C25 | ACC<=ACC or BR | Or |
| C26 | ACC<=NOT ACC | Not |
| C27 | ACC<=SLL ACC 1 bit |  |
| C28 | ACC<=SRL ACC 1 bit |  |
| C29 | ACC<=ACC\*BR(L) MR<=ACC\*BR(H) | MPY and send results |
| C30 | ACC<=SLA ACC 1 bit |  |
| C31 | ACC<=SRA ACC 1 bit |  |

下面列举了所有出现的指令分解成微操作后的形式，每一条微操作都对应不同的控制信号，它们控制各个模块完成相应的任务。

**Auto fetch**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory,CAR<=CAR+1 | C5,C17 | 00020020 |
| IR<=MBR[15..8],CAR<=CAR+1 | C4,C17 | 00020010 |
| CU<= IR, CAR<=CAR+1 | Control Unit<=IR,C17 |  |
| CAR<=\*\*\* (\*\*\* is determined by OPCODE) | C18 | 00040000 |

**STORE**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR,PC<=PC+1, CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=ACC, CAR<=CAR+1 | C11,C17 | 00020800 |
| memory<=MBR, CAR<=CAR+1 | C12,C17 | 00021000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**LOAD**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, ACC<=0, CAR<=CAR+1 | C6,C21,C17 | 00220040 |
| ACC<=ACC+BR, CAR<=CAR+1 | C22,C17 | 00420000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**ADD**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=ACC+BR, CAR<=CAR+1 | C22,C17 | 00420000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**SUB**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=ACC-BR, CAR<=CAR+1 | C23,C17 | 00820000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**MPY**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=ACC\*BR(L), MR<=ACC\*BR(H), CAR<=CAR+1 | C29,C17 | 40020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**AND**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=ACC&&BR, CAR<=CAR+1 | C24,C17 | 01020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**OR**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=ACC||BR, CAR<=CAR+1 | C25,C17 | 02020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**NOT**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR, CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=!ACC, CAR<=CAR+1 | C26,C17 | 04020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**SLL**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR,CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=SLL(ACC),CAR<=CAR+1 | C27,C17 | 08020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**SRL**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR,CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=SRL(ACC),CAR<=CAR+1 | C28,C17 | 10020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**SLA**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR,CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=SLA(ACC),CAR<=CAR+1 | C30,C17 | 40020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**SRA**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| MAR<=MBR[7..0], PC<=PC+1,  CAR<=CAR+1 | C8,C20,C17 | 00120100 |
| Memory<=MAR, CAR<=CAR+1 | C0,C17 | 00020001 |
| MBR<=memory, CAR<=CAR+1 | C5,C17 | 00020020 |
| BR<=MBR,CAR<=CAR+1 | C6,C17 | 00020040 |
| ACC<=SRA(ACC),CAR<=CAR+1 | C31,C17 | 80020000 |
| MAR<=PC, CAR<=0 | C2,C19 | 00080004 |

**JMPGEZ**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| IF(FLAG=0)PC<=MBR,  CAR<=CAR+1 | C3,C17 | 00020008 |
| IF(FLAG=1)PC<=PC+1,  CAR<=CAR+1 | C20,C17 | 00120000 |
| MAR<=PC,CAR<=0 | C2,C19 | 00080004 |

**JMP**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| PC<=MBR[7..0],CAR<=CAR+1 | C3,C17 | 00020008 |
| MAR<=PC,CAR<=0 | C2,C19 | 00080004 |

**HALT**

|  |  |  |
| --- | --- | --- |
| Microprogram | Control signals | Microinstruction |
| CAR<=0 | C19 | 00080000 |

## 2.4 顶层结构

RTL原理图总体如图4(a)所示，为了更清晰地展示各个模块之间互相连接情况，如图4(b)(c)(d)所示。

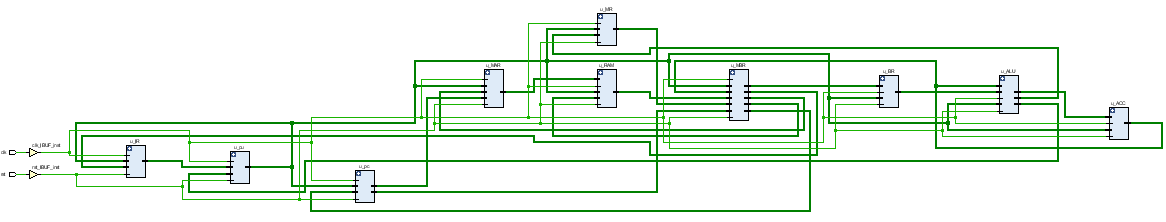


图4(a) RTL原理图总体

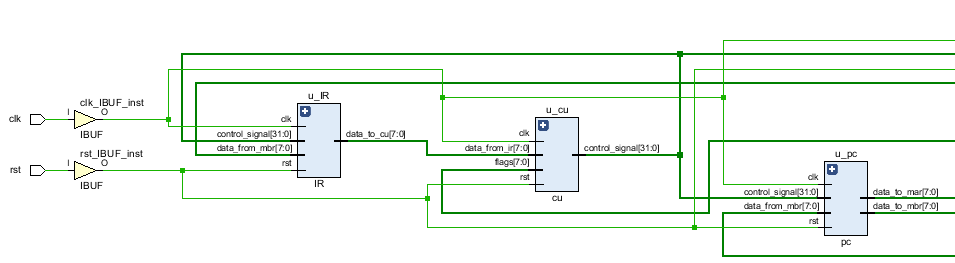


图4(b) RTL原理图——IR、CU、PC模块

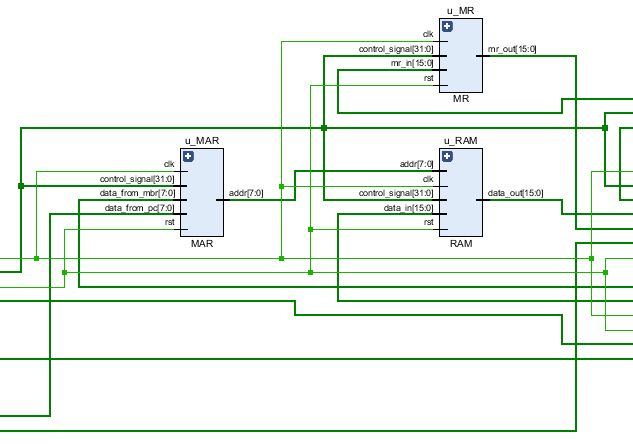


图4(c) RTL原理图——MAR、MR、RAM模块

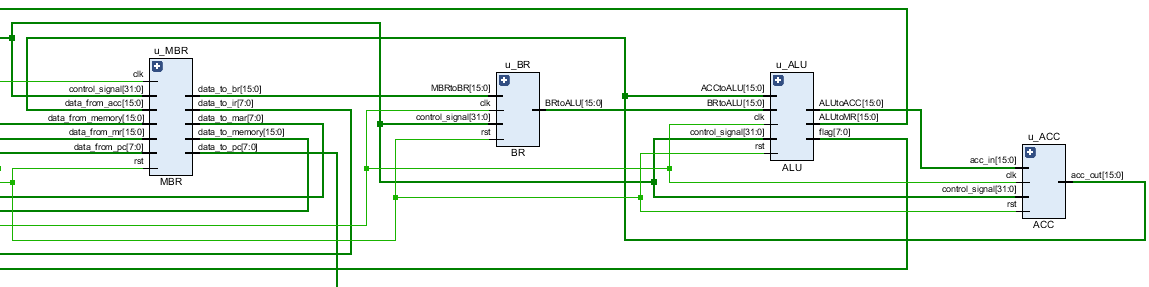


图4(d) RTL原理图——MBR、BR、ALU、ACC模块

# 三、结果仿真

# 四、总结和讨论