厦門大學
XIAMEN UNIVERSITY

电子系统设计 Part2

简单处理器设计

信息与通信工程系 陈凌宇

chenly@xmu.edu.cn

导言

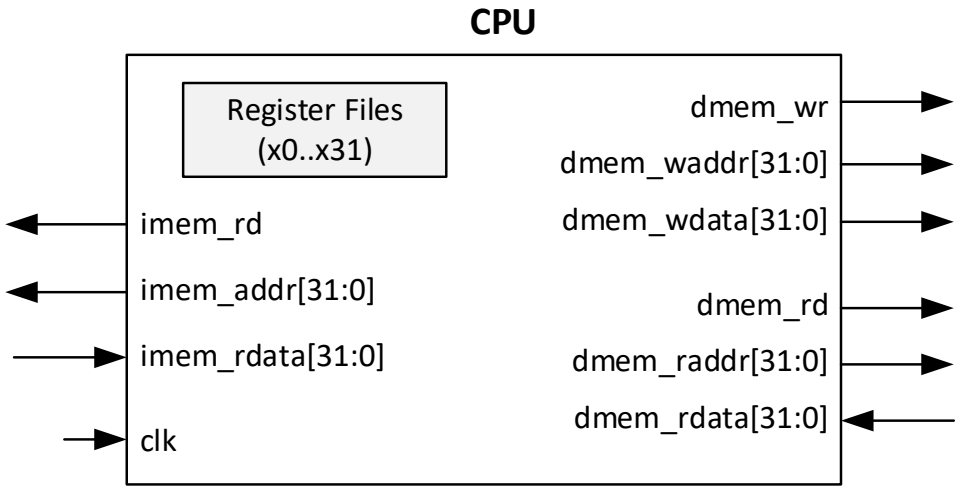
本部分设计了一个简单的CPU，通过编写机器代码，来实现简单的闪灯功能。在此例程的基础上，请同学们自主添加指令集，实现跑马灯。

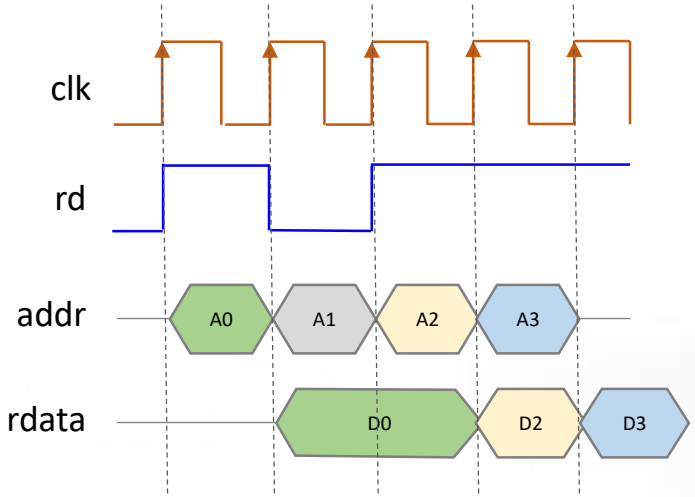
目录

- 处理器指令集
- 处理器体系设计与实现
- 软件编程
- 拓展实验

处理器指令集

- 一个简单的CPU
  - 内含有32个寄存器x0..x31，约定寄存器x0始终读出为0，写入x0没有实际效果；
  - 有1个指令读取端口，以及1个数据访问端口；





处理器指令集

- 支持指令集包括
  - 内存加载/存储指令：LW，SW；
  - 算数/逻辑运算：ADD, ADDI, SLT, LUI；
  - 跳转指令：BEQ, JAL；

处理器指令集

- LW: 字加载 (Load Word)， I-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					func3			rd					opcode						
imm[11:0]: -2048 ~ 2047												**rs1(0-31)**					**0**	**1**	**0**	**rd(0-31)**					**0**	**0**	**0**	**0**	**0**	**1**	**1**

从 [rs1] + sign-extend(imm)对应的地址中读取四个字节，写入[rd]寄存器

- SW: 字存储 (Store Word)，S-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:5]							rs2					rs1					func3			imm[4:0]					opcode						
imm[11:5]							**rs2(0-31)**					**rs1(0-31)**					**0**	**1**	**0**	**imm[4:0]**					**0**	**1**	**0**	**0**	**0**	**1**	**1**

将 [rs2] 存入 [rs1]+sign-extend(imm)对应的内存地址中

处理器指令集

- ADD: 寄存器加法 (Add)，R-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
func7							rs2					rs1					func3			rd					opcode						
0	**0**	**0**	**0**	**0**	**0**	**0**	**rs2(0-31)**					**rs1(0-31)**					**0**	**0**	**0**	**rd(0-31)**					**0**	**1**	**1**	**0**	**0**	**1**	**1**

把寄存器[rs2]、寄存器[rs1]的加和写入寄存器[rd]

- ADDI: 立即数加法(Add Immediate)，I-Type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					func3			rd					opcode						
imm[11:0]: -2048 ~ 2047												**rs1(0-31)**					**1**	**1**	**1**	**rd(0-31)**					**0**	**0**	**1**	**0**	**0**	**1**	**1**

把sign-extend(imm)加寄存器 [rs1]，结果写入寄存器[rd]

处理器指令集

- SLT: 小于则置位(Set if Less Than)，R-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
func7							rs2					rs1					func3			rd					opcode						
0	**0**	**0**	**0**	**0**	**0**	**0**	**rs2(0-31)**					**rs1(0-31)**					**0**	**1**	**0**	**rd(0-31)**					**0**	**1**	**1**	**0**	**0**	**1**	**1**

比较 [rs1]和 [rs2]中的数，如果 [rs1]更小，则[rd]写入 1，否则写入 0

- LUI: 高位立即数加载 (Load Upper Immediate)，U-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[31:12]																				rd					opcode						
imm[31:12]: 0~1048575																				**rd(0-31)**					**0**	**1**	**1**	**0**	**1**	**1**	**1**

使得[rd]的高20位与imm[31:12]一致，低位补0

处理器指令集

- BEQ: 相等时分支跳转 (Branch if Equal)，B-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					func3			imm[4:1|11]					opcode						
imm[12|10:5]							**rs2(0-31)**					**rs1(0-31)**					**0**	**0**	**0**	**imm[4:1|11]**					**1**	**1**	**0**	**0**	**0**	**1**	**1**

若寄存器 [rs1]和 [rs2]相等，则pc <= pc + sign-extend(imm)，默认imm[0]=0

- JAL: 跳转并链接 (Jump and Link)，J-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[20|10:1|11|19:12]																				rd					opcode						
imm[20|10:1|11|19:12]																				**rd(0-31)**					**1**	**1**	**0**	**1**	**1**	**1**	**1**

[rd] <= pc+4; pc += sign-extend(imm)， 默认imm[0]=0

处理器指令集约定

- 操作码OPCODE均位于指令的[6:0]；若有子操作码func3的，位于指令的[14:12]，子操作码func7位于[31:25];
- 若涉及到回写的目标寄存器，均位于指令[11:7]；
- 若涉及源寄存器的，源寄存器1均位于指令[19:15]；
- 若涉及2个源寄存器的，源寄存器2均位于指令[24:20]；
- 其余的为立即数imm;

目录

- 处理器指令集
- 处理器体系设计与实现
- 软件编程
- 拓展实验

处理器实现

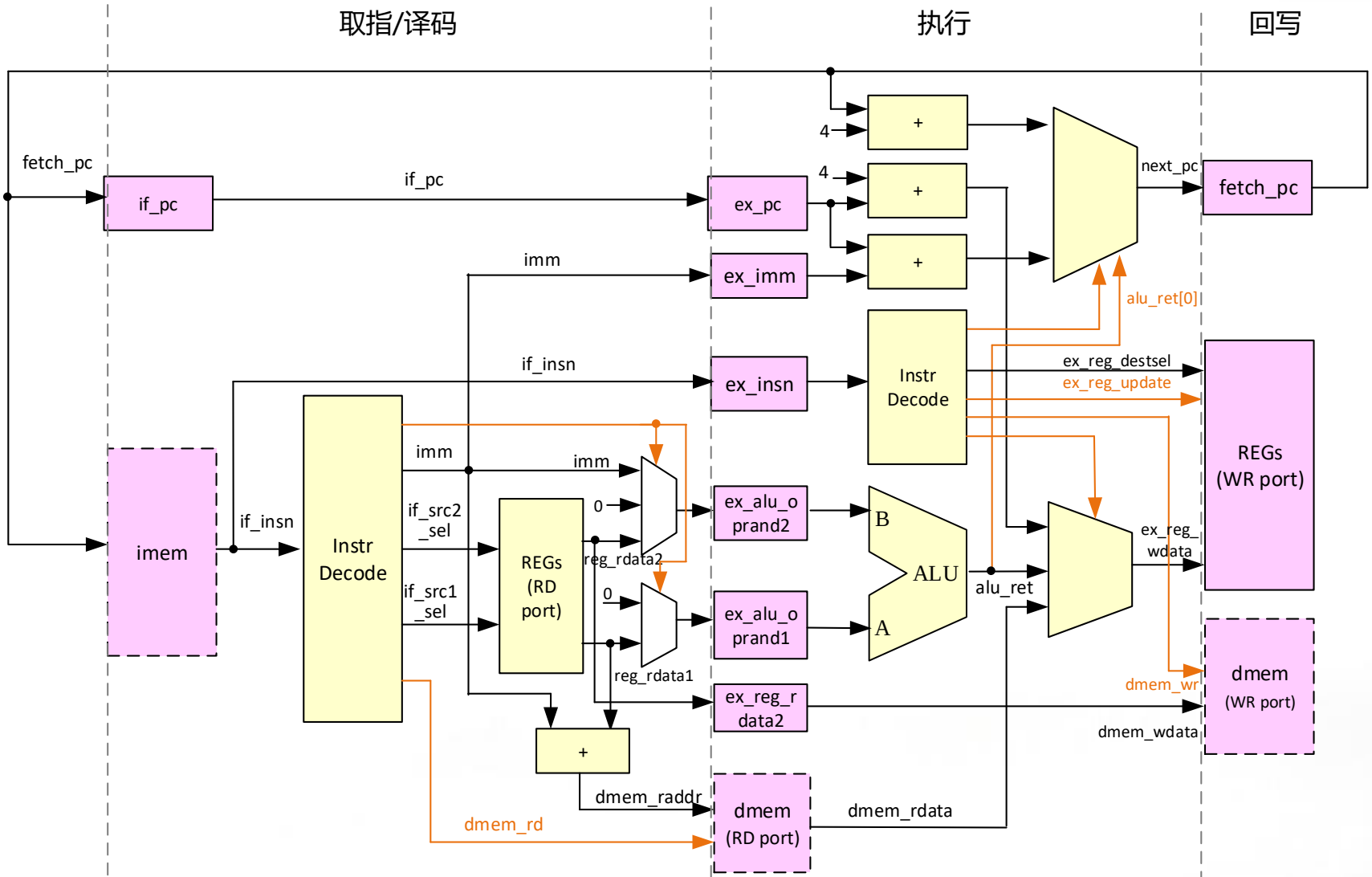
- 寄存器实现

```
reg  [31:0] regs     [31:1];

// Write Register file
integer       i;
always @(posedge clk or negedge rstn)  begin
   if (!rstn) begin
      for (i = 1; i < 32; i = i + 1)  regs[i] <= 32'h0;
   end else if (ex_reg_update) begin
      regs[ex_reg_destsel] <= ex_reg_wdata;
   end
end

// Read Register file
assign reg_rdata1[31: 0]   =  (if_src1_sel == 5'h0) ? 32'h0 : regs[if_src1_sel];
assign reg_rdata2[31: 0]   =  (if_src2_sel == 5'h0) ? 32'h0 : regs[if_src2_sel];
```

处理器实现


取指/译码
执行
回写
fetch_pc
if_pc
if_pc
ex_pc
4
+
4
+
+
next_pc
fetch_pc
imm
ex_imm
alu_ret[0]
if_insn
ex_insn
Instr
Decode
ex_reg_destsel
ex_reg_update
REGs
(WR port)
imem
if_insn
Instr
Decode
imm
imm
if_src2
_sel
if_src1
_sel
REGs
(RD
port)
0
reg_rdata2
0
reg_rdata1
ex_alu_o
prand2
ex_alu_o
prand1
B
ALU
A
alu_ret
ex_reg_
wdata
dmem
(WR port)
dmem_wr
ex_reg_r
data2
dmem_wdata
+
dmem_raddr
dmem_rd
dmem
(RD port)
dmem_rdata

目录

- 处理器指令集
- 处理器体系设计与实现
- 软件编程
- 拓展实验

流水线问题

- 数据冒险

```
addi x1, x0, 3
addi x2, x0, 4
add  x3, x1, x2
```

	T1	T2	T3
取值/译码	Instr 2	Instr 3 (x2旧)	
执行		Instr 2	Instr 3
回写			Instr 2(x2新)

第2条指令更新x2在回写阶段，
第3条指令读取x2在取值/译码阶段

- 控制冒险

```
beq  x1, x2, 50
addi x1, x0, 2
add  x4, x1, x4
```

	T1	T2	T3	T4	
取值/译码	Instr 1	Instr 2	Instr 3	Instr - NEW	Instr – NEW2
执行		Instr 1	Instr 2	Instr 3	Instr - NEW
回写			Instr 1(跳转)	Instr 2	Instr 3

在发生跳转时，后面2条语句会被执行

解决方案

- 对于数据冒险

```
addi x1, x0, 3
addi x2, x0, 4
addi x0, x0, x0
add  x3, x1, x2
```

- 对于控制冒险

```
beq  x1, x2, 50
addi x0, x0, x0
addi x0, x0, x0
addi x1, x0, 2
add  x4, x1, x4
```

- 总结
  - 对于数据冒险/内存读写，插入1条空指令；
  - 对于跳转语句，插入2条空指令；

例子

- 案例给出了软件编写的方法，通过以下网站，可以实现机器码的快速生成。

  https://luplab.gitlab.io/rvcodecjs/

目录

- 处理器指令集
- 处理器体系设计与实现
- 软件编程
- 拓展实验

拓展项目

- 在例程的基础上，自主添加指令集，在开发板上点亮8个LED，实现跑马灯。
  - 修改verilog代码，增加指令；
  - 修改软件代码，实现功能；
  - 可以利用vivado进行仿真，确认无误后下板测试。

Question ?