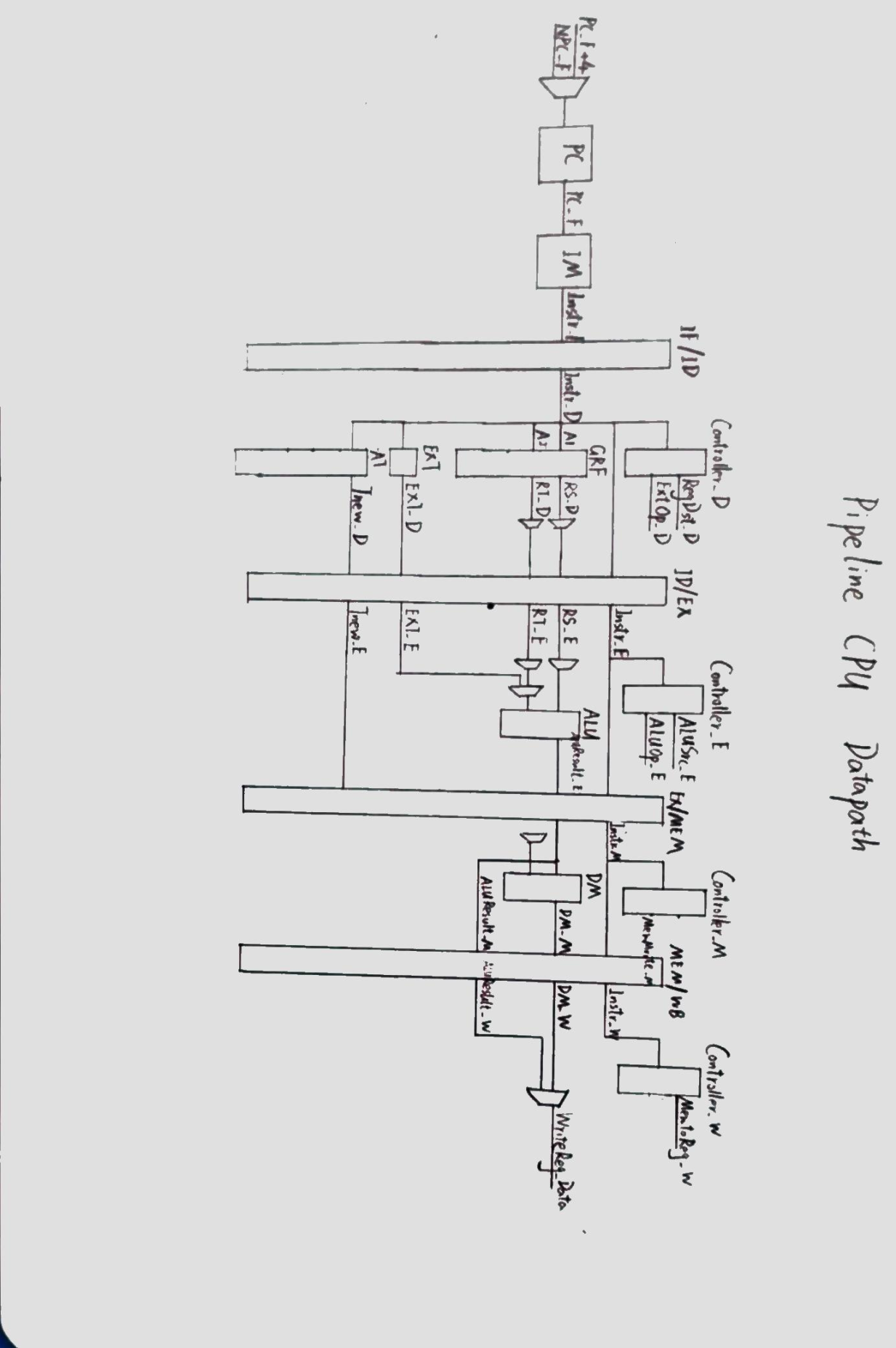
Verilog流水线CPU

1. 顶层设计



1. 模块设计
2. AT

表1 解码器接口定义表

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| AT.v | input [31:0] IR,  output Tuse\_RS0,  output Tuse\_RS1,  output Tuse\_RT0,  output Tuse\_RT1,  output Tuse\_RT2,  output reg [1:0] Tnew\_D,  output [4:0] A1,  output [4:0] A2 |

1. Forward

表2 转发信号控制接口定义表

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| AT.v | input RegWriteEn\_E,  input RegWriteEn\_M,  input RegWriteEn\_W,  input [4:0] A1\_D,  input [4:0] A2\_D,  input [4:0] A1\_E,  input [4:0] A2\_E,  input [4:0] A2\_M,  input [4:0] A3\_E,  input [4:0] A3\_M,  input [4:0] A3\_W,  input [1:0] Tnew\_E,  input [1:0] Tnew\_M,  input [1:0] Tnew\_W,  input jal,  output [2:0] ForwardRSD,  output [2:0] ForwardRTD,  output [2:0] ForwardRSE,  output [2:0] ForwardRTE,  output [1:0] ForwardRTM |

1. Stall

表3 暂停信号控制接口定义表

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| stall.v | input [31:0] IR\_D,  input Tuse\_RS0,  input Tuse\_RS1,  input Tuse\_RT0,  input Tuse\_RT1,  input Tuse\_RT2,  input [1:0] Tnew\_E,  input [1:0] Tnew\_M,  input [4:0] A3\_E,  input [4:0] A3\_M,  input RegWriteEn\_E,  input RegWriteEn\_M,  output UpdatePC,  output IF\_ID\_EN,  output ID\_EXE\_Clr |

Tuse为：这条指令位于D级的时候，再经过多少个时钟周期就必须要使用相应的数据。

Tnew为：位于某个流水级的某个指令，它经过多少个时钟周期可以算出结果并且存储到流水级寄存器里。

Tnew=0，说明结果已经算出，如果指令处于WB级，则可以通过寄存器的内部转发设计解决（关于内部转发请自行查阅资料思考或与同学讨论），不需要任何操作。如果指令不处于WB级，则可以通过转发结果来解决。

Tnew<=Tuse，说明需要的数据可以及时算出，可以通过转发结果来解决。（与1有何区别？）

Tnew>Tuse，说明需要的数据不能及时算出，必须暂停流水线解决。

1. 思考题

1.在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证覆盖了所有需要测试的情况；如果你是完全随机生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了特殊的策略，比如构造连续数据冒险序列，请你描述一下你使用的策略如何结合了随机性达到强测的效果。此思考题请同学们结合自己测试CPU使用的具体手段，按照自己的实际情况进行回答。

R型指令的冲突，I型指令的冲突，B型指令（beq）的冲突，load型指令（load）的冲突，save类型指令（sw）的冲突，jr指令的冲突。

1. 测试程序

汇编代码：

# test ori

ori $t0, 100

ori $s0, 100

ori $s1, 200

ori $t1, 0

ori $t2, 0

# test $0

ori $0, 0xffff

# test addu

addu $a0, $0, $t0

# test subu

subu $s1, $t0, $0

# test beq

loop\_beq\_1:

addu $t1, $t1, $t0

beq $t0, $s0, loop\_beq\_2

nop

ori $t3, 100

loop\_beq\_2:

addu $t2, $t2, $t0

addu $t0, $t0, 100

beq $t0, $s1, loop\_beq\_1

nop

ori $t4, 0xffff

ori $t5, 0xffc

# test sw

sw $t4, 0($0)

sw $t4, 4($0)

sw $t4, -4($t5)

sw $t4, 0($t5)

# test lw

lw $t6, 0($0)

lw $t7, 4($0)

lw $t8, -4($t5)

lw $t9, 0($t5)

# test lui

lui $s2, 0

lui $s3, 0xffff

lui $s4, 100

ori $s0, 100

ori $s1, 1

# test jal, jr, nop

loop\_jal\_1:

addu $s2, $s2, $s0

nop

jal loop\_jal\_2

nop

jal loop\_jal\_4

nop

loop\_jal\_2:

nop

addu $s2, $s2, $s0

nop

sw $ra, 0($s0)

jal loop\_jal\_3

nop

lw $ra, 0($s0)

jr $ra

nop

loop\_jal\_3:

addu $s5, $0, $s3

nop

jr $ra

nop

addu $s6, $s5, $0

end:

addu $s4, $s4, $s1

jr $ra

nop

loop\_jal\_4:

addu $s4, $s4, $s1

jal end

nop

addu $s5, $s5, 1

机器代码：

35080064

36100064

363100c8

35290000

354a0000

3400ffff

00082021

01008823

01284821

11100002

00000000

356b0064

01485021

3c010000

34210064

01014021

1111fff7

00000000

358cffff

35ad0ffc

ac0c0000

ac0c0004

adacfffc

adac0000

8c0e0000

8c0f0004

8db8fffc

8db90000

3c120000

3c13ffff

3c140064

36100064

36310001

02509021

00000000

0c000c27

00000000

0c000c38

00000000

00000000

02509021

00000000

ae1f0000

0c000c30

00000000

8e1f0000

03e00008

00000000

0013a821

00000000

03e00008

00000000

02a0b021

0291a021

03e00008

00000000

0291a021

0c000c35

00000000

3c010000

34210001

02a1a821