现代计算机组成原理

潘 明　潘 松 编著

科学出版社

第 7 章

流水线结构RISC CPU设计

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

取指令 — 按照指令计数器的内容访问主存储器，取出一条指令送到指令寄存器。

指令分析 — 对指令操作码进行译码，按照给定的寻址方式，将地址字段中的内容形成操作数的地址，并用这个地址读取操作数。操作数可在主存中，也可在通用寄存器中。

指令执行 — 根据操作码的要求，完成指令规定的功能，即把运算结果写到通用寄存器或主存中。

1．非流水线结构数据通路

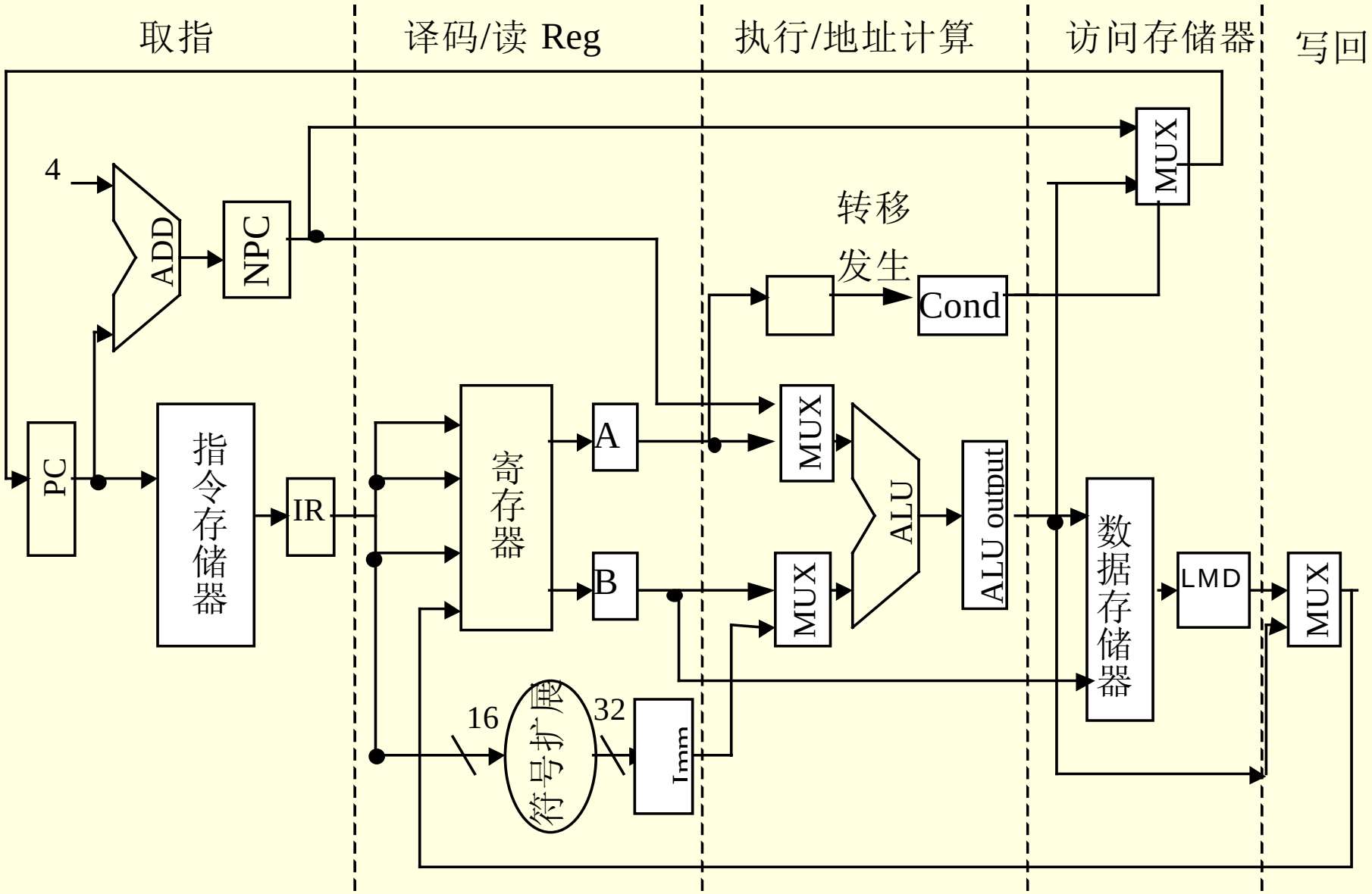


图7-1非流水线实现的指令解释数据通路

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

1．非流水线结构数据通路

（1）取指令周期（IF）：

IR ← Mem[PC]

NPC ← PC + 1

（2）译码/读寄存器周期（ID）

A ← Reg[IR 6..10]

B ← Reg[IR11..15]

Imm ← ((IR16)16## IR16..31)

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

1．非流水线结构数据通路

（3）执行/有效地址计算（ALU）

Load/Store： ALUoutput ←A+Imm

R-R ALU： ALUoutput ←A func B

R-I ALU： ALUoutput ←A op Imm

Branch： ALUoutput ←NPC + Imm;

Cond ←A op 0

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

1．非流水线结构数据通路

（4）存储器访问/转移完成（MEM）

Load/Store：　LMD ← Mem[ALUoutput]

Mem[ALUoutput] ← B

Branch：　　if　(Cond)　then　PC←ALUoutput

else　PC ← NPC

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

1．非流水线结构数据通路

（5）写回周期（**WB**）

R-R ALU：　　Regs[IR16..20] ← ALUoutput

R-I ALU：　　Regs[IR11..15] ← ALUoutput

Load：　　Regs[IR11..15] ← LMD

2．DLX基本指令流水线

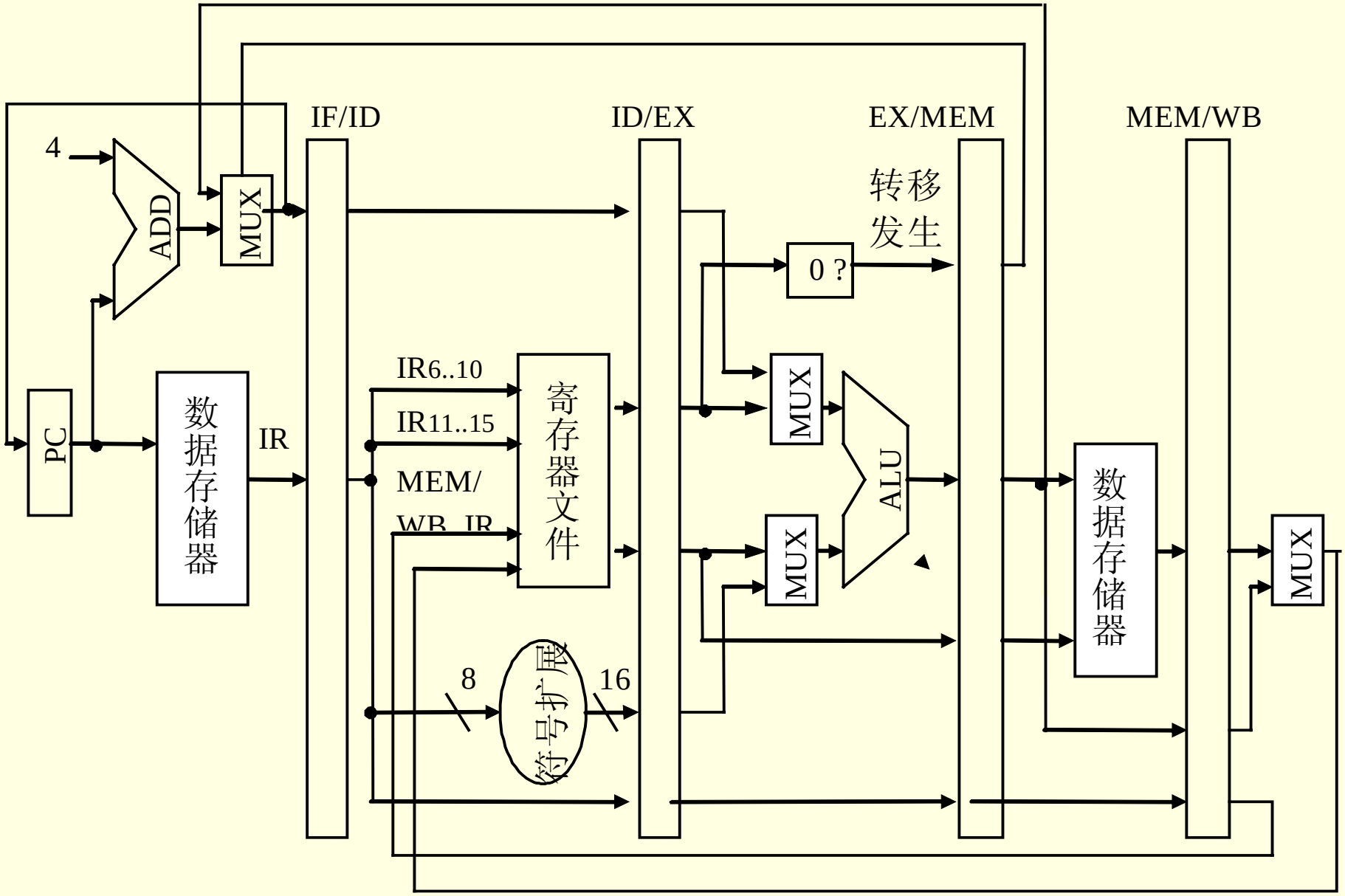


图7-2 DLX基本指令流水线

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

2. DLX基本指令流水线

表7-1　　五级流水线的每一级的具体操作

指令 / 流水段	ALU	LOAD/STORE	BRANCH
IF	取指令	取指令	取指令
ID	译码，读寄存器文件	译码，读寄存器文件	译码，读寄存器文件
EXE	执行	计算访存有效地址	计算转移目标地址，设置条件码
MEM	(空操作)	对存储器读或写操作	若条件成立，将转移地址送 PC
WB	结果写入寄存器文件	读出数据写入寄存器文件	(空操作)

7.1 流水线的一般概念

7.1.2流水线CPU的时空图

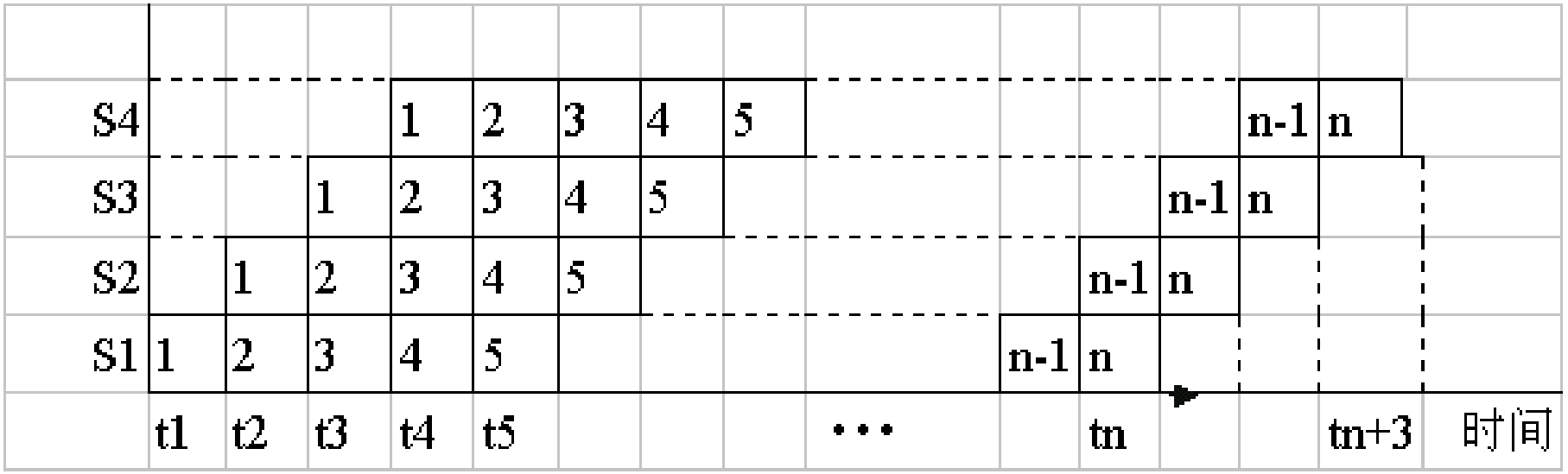


图7-3 流水线时空图

7.1 流水线的一般概念

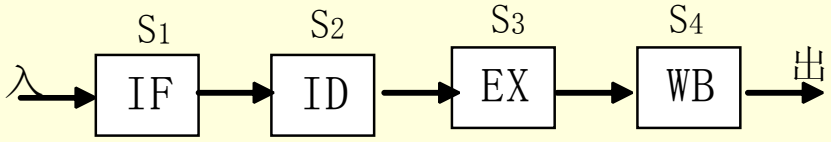
7.1.2流水线CPU的时空图

指令序列	流水时钟数								
	1	2	3	4	5	6	7	8	9
指令 i	IF	ID	EX	MEM	WB				
指令 i +1		IF	ID	EX	MEM	WB			
指令 i +2			IF	ID	EX	MEM	WB		
指令 i +3				IF	ID	EX	MEM	WB	
指令 i +4					IF	ID	EX	MEM	WB

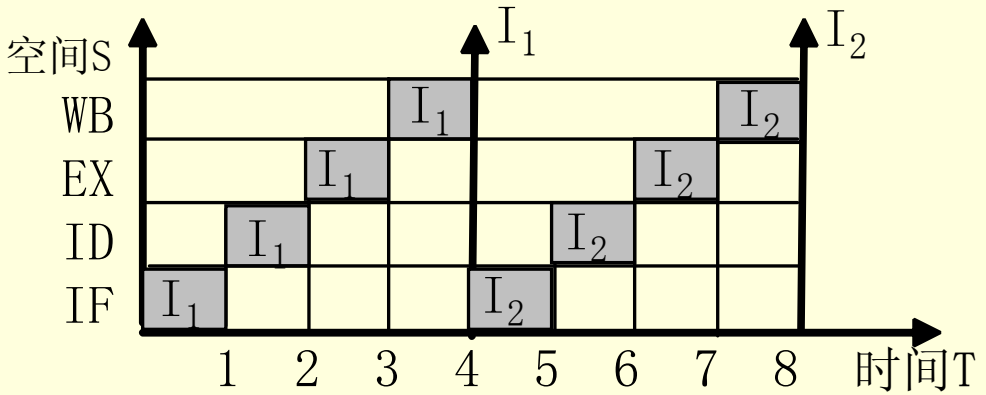
图7-4 指令流水线的时空图

7.1 流水线的一般概念

7.1.2流水线CPU的时空图



(a) 一个指令流水线过程段

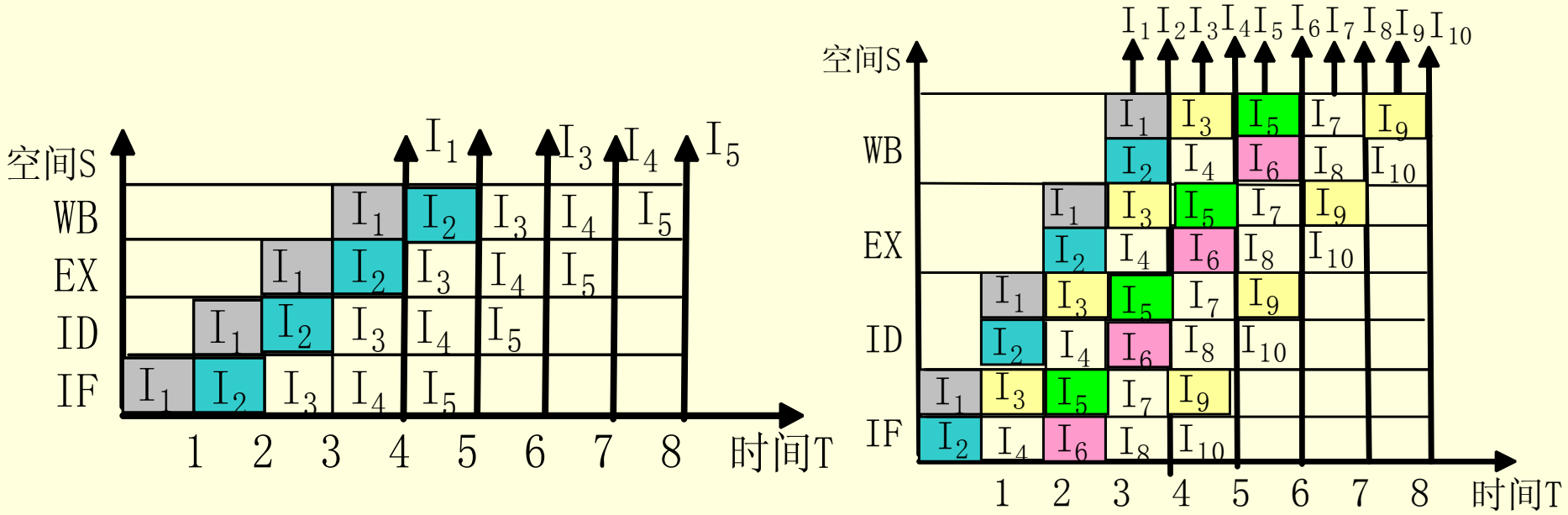


b）非流水线时空图

图7-5 流水线时空图

7.1 流水线的一般概念

7.1.2流水线CPU的时空图



(c) 标量流水时空图　　d）超标量流水线时空图

图7-5 流水线时空图

7.1 流水线的一般概念

7.1.3 流水线分类

指令流水线 ⇒ 指令步骤的并行

算术流水线 ⇒ 运算操作步骤的并行

处理机流水线 ⇒ 程序步骤的并行

7.2 流水线中的主要问题及处理

7.2.1 资源相关

解决主存资源冲突的方法

① 将后续的第(i+3)条指令推迟一拍进入流水线。

② 增设一个存储器。将指令和数据分别存放在两个存储器中。

③ 采用先行控制技术，或在处理器内部设置指令缓冲队列。

7.2 流水线中的主要问题及处理

7.2.2 数据相关及其分类

(1) 读后写(WAR)相关

```
MUL  R1，R2   ；(R1)×(R2)→R1，
  ADD  R3，R1   ；(R1)＋(R3)→R3
```

(2) 写后读(RAW)相关

```
MUL  R1，R 2   ；(R1)×(R2)→R1
MOV  R2，#00H  ；0→R2
```

(3) 写后写(WAW)相关

```
MUL  R1，R2   ；(R1)×(R2)→R1
MOV  R1，#00H  ；0→R1
```

7.2 流水线中的主要问题及处理

7.2.3 数据竞争的处理技术

【例7-1】数据相关实例

```
ADD   R1，R2，R3   ;    （R2）＋（R3）→R1
SUB   R4，R1，R5   ;    （R1）－（R5）→R4
AND   R6，R1，R7   ;    （R1）∧（R7）→R6
```

表7-2　　例7-1中出现的数据相关

	时钟 1	时钟 2	时钟 3	时钟 4	时钟 5	时钟 6	时钟 7	时钟 8
指 令 ADD	IF	ID	EX	MEM	WB			
指 令 SUB		IF	ID	EX	MEM	WB		
指 令 AND			IF	ID	EX	MEM	WB	

7.2 流水线中的主要问题及处理

7.2.3 数据竞争的处理技术

【例7-2】

（1）I1　ADD R1，R2，R3　　；（R2）＋（R3）→R1
　　I2　SUB R4，R1，R5　　；（R1）－（R5）→R4
（2）I3　STA M（x），R3　　；（R3）→M（x），M（x）是存储单元
　　I4　ADD R3，R4，R5　　；（R4）＋（R5）→R3
（3）I5　MUL R3，R1，R2　　；（R1）×（R2）→R3
　　I6　ADD R3，R4，R5　　；（R4）＋（R5）→R3

【例7-3】

```
ADD   R1，R2，R3
SUB   R4，R5，R1
AND   R6，R1，R7
OR    R8，R1，R9
XOR   R10，R1，R11
```

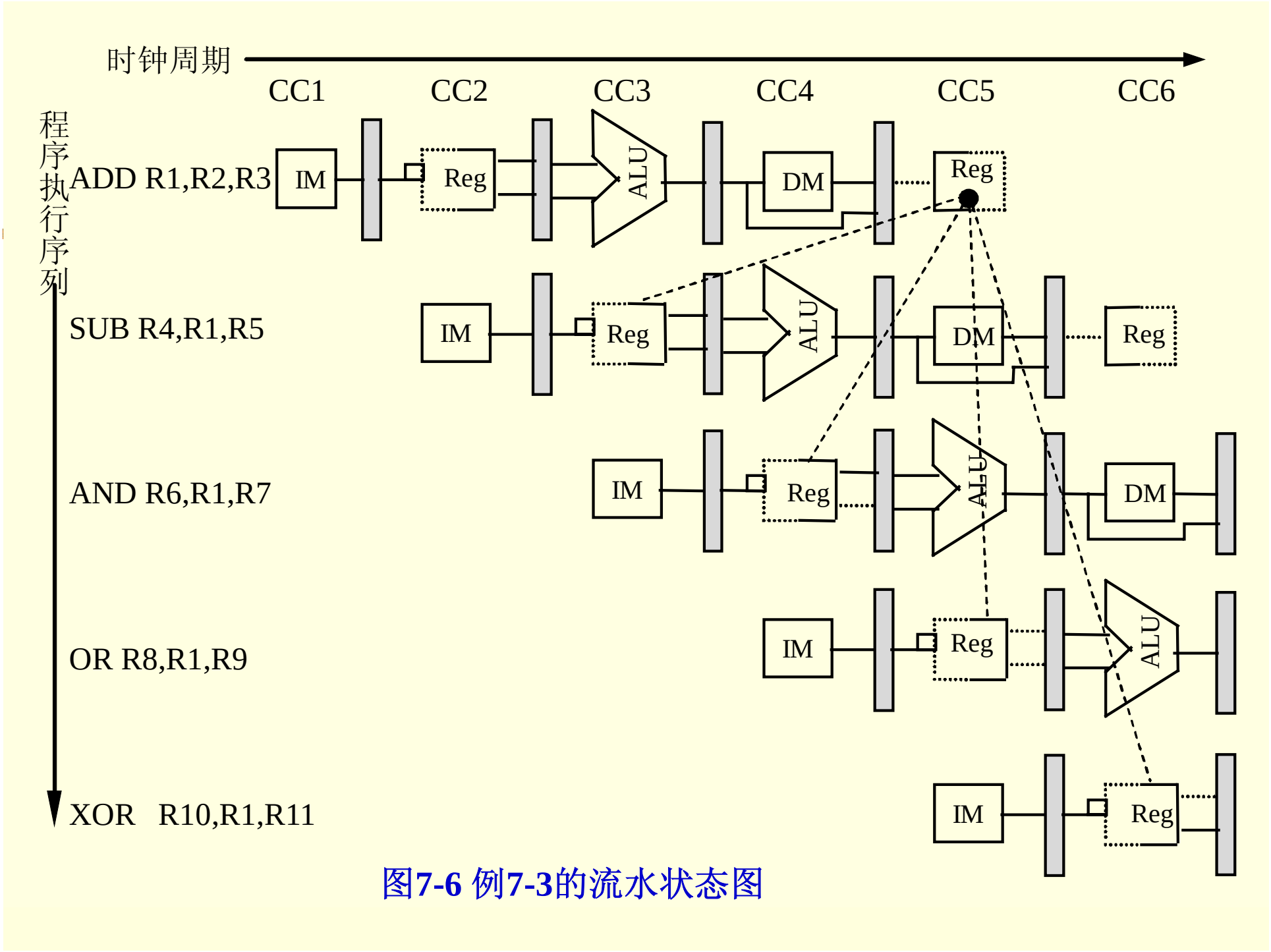


图7-6 例7-3的流水状态图

7.2 流水线中的主要问题及处理

7.2.3 数据竞争的处理技术

【例7-4】

```
LW   R1，0（R2）      ；R1最早要等到第四拍MEM结束，
                ；才能得到数据.(即才能从Data Memory读入，
                  存入暂存器)
SUB  R4，R1，R5     ；SUB所用的R1最迟在第三拍EX开始时要准备好，
                      与LW相差一拍。
AND  R6，R1，R7     ；同SUB，可用“提前”方法解决
OR   R8，R1，R9     ；同SUB，可用“提前”方法解决
```

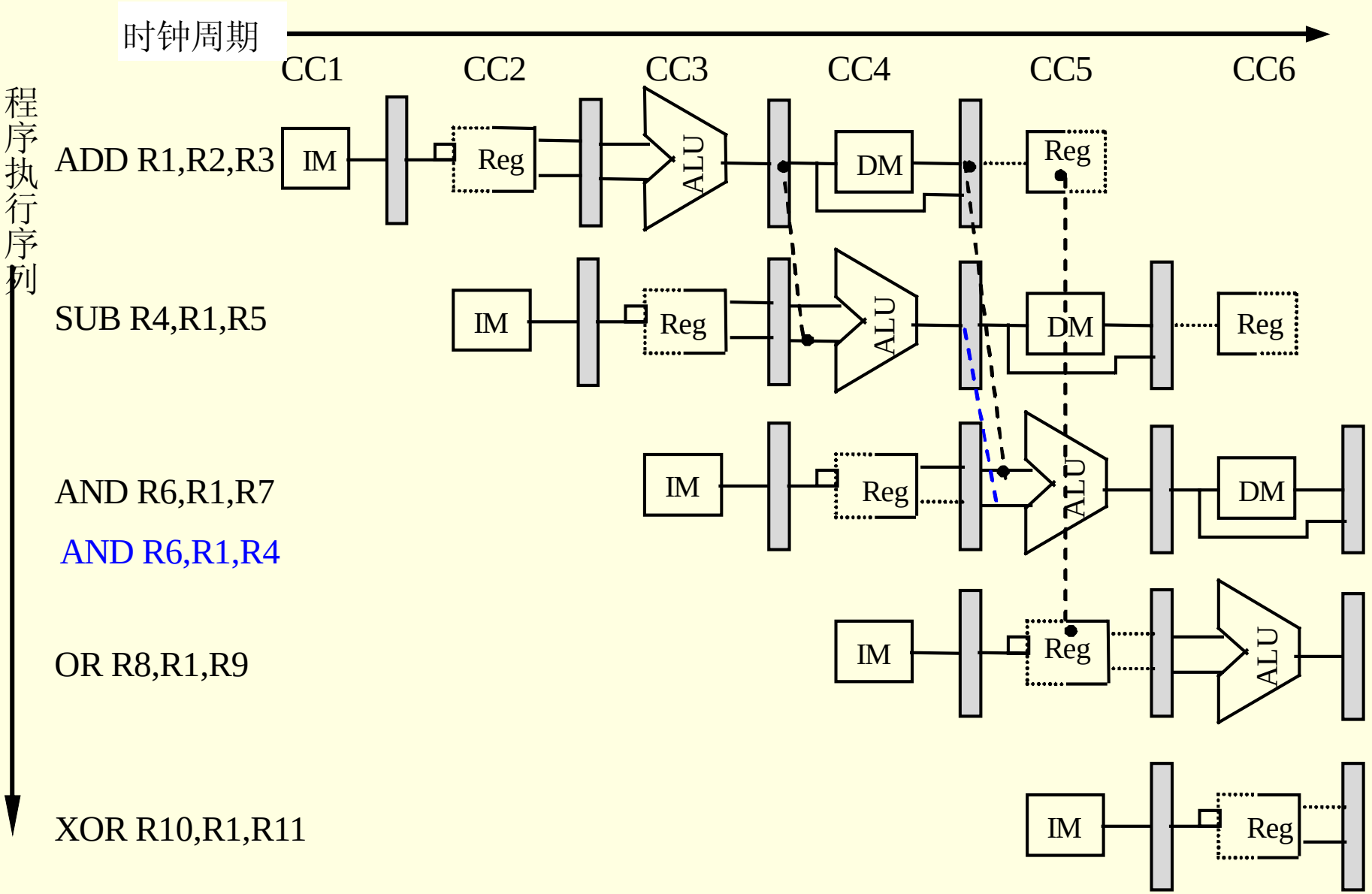


图7-7引入forwarding path后的状态图

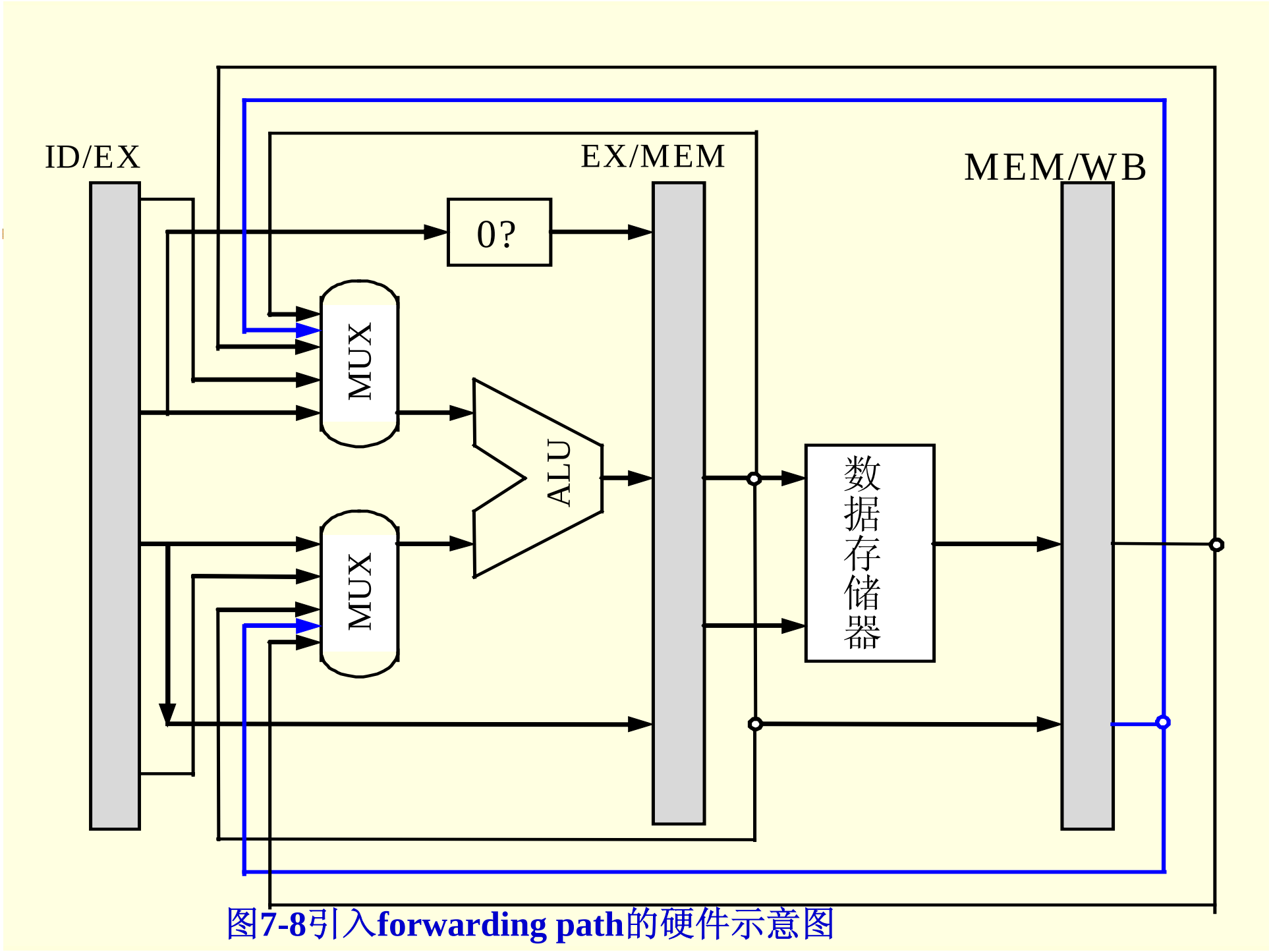


图7-8引入forwarding path的硬件示意图

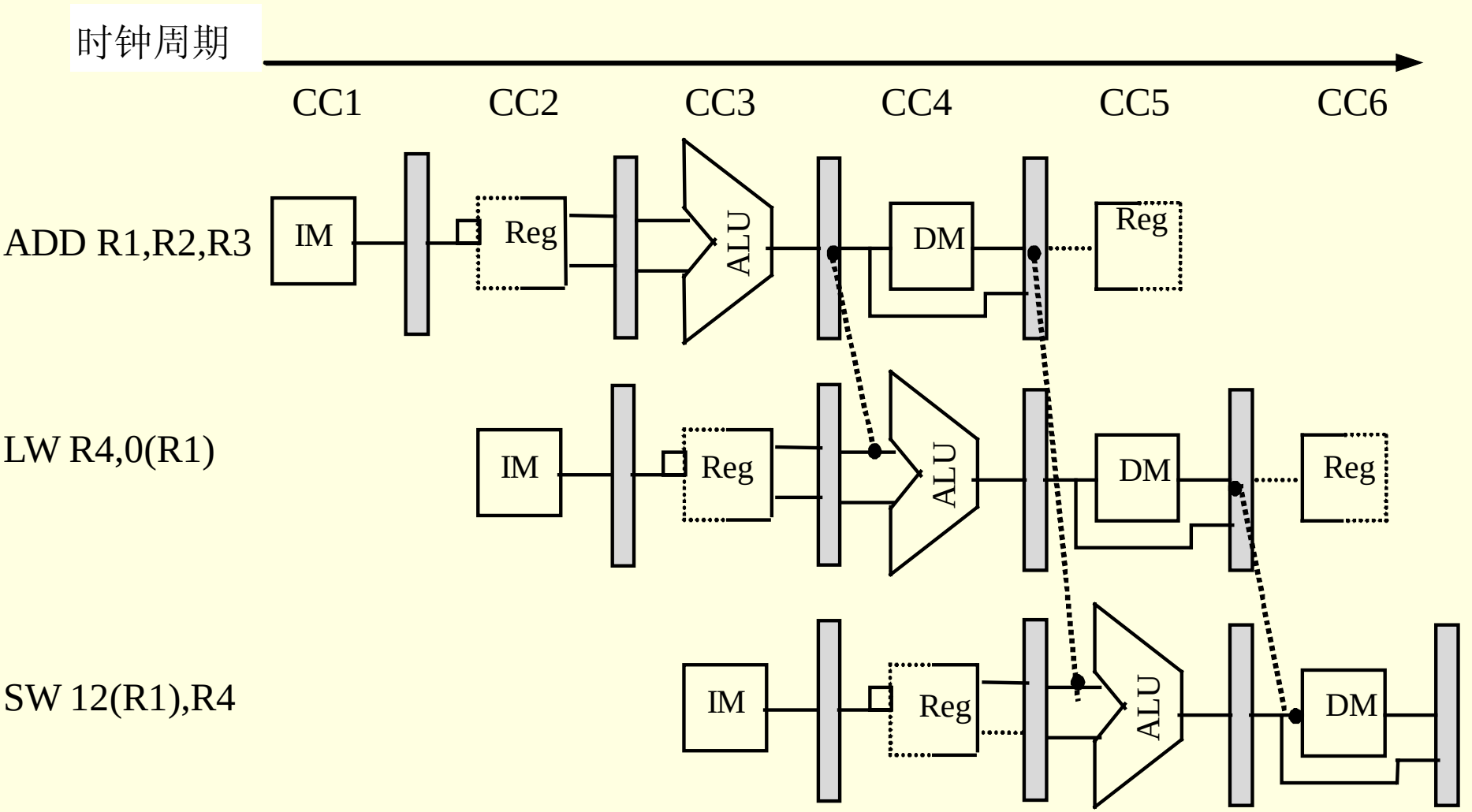


图7-9 引入forwarding后消除了Stall

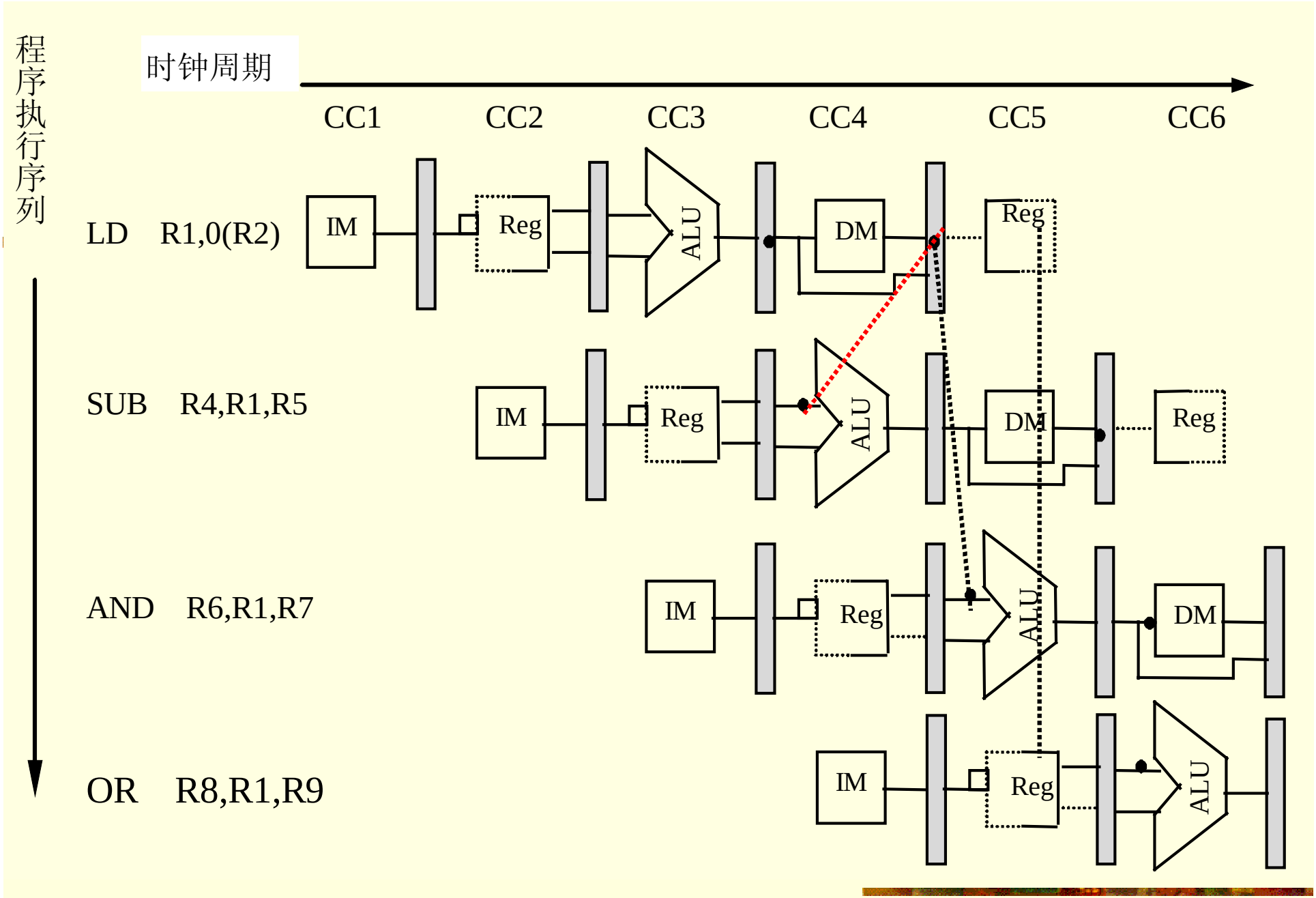


图7-10 例7-4的流水线状态图

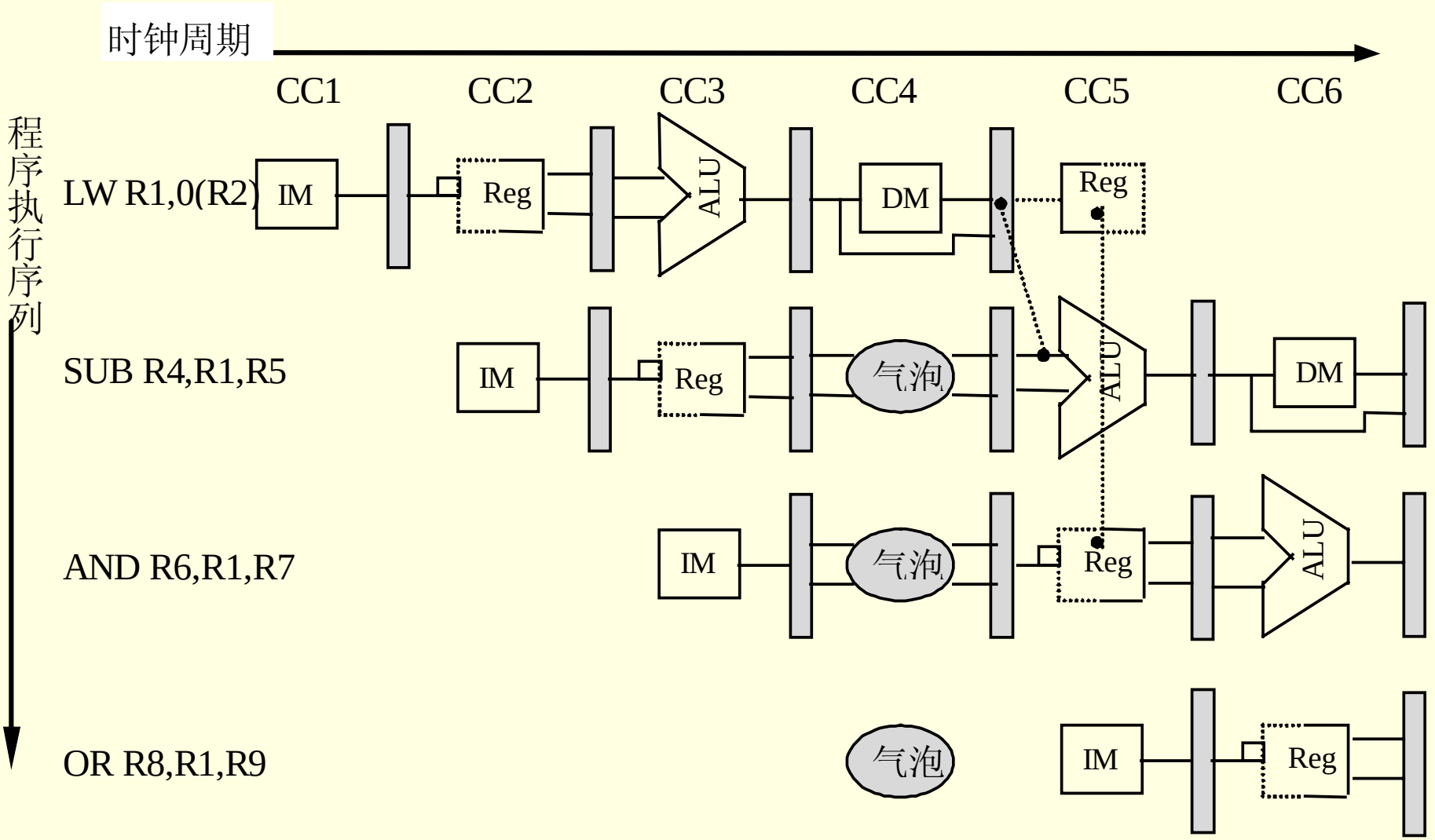


图7-11 插入Stall消除Load引起的竞争

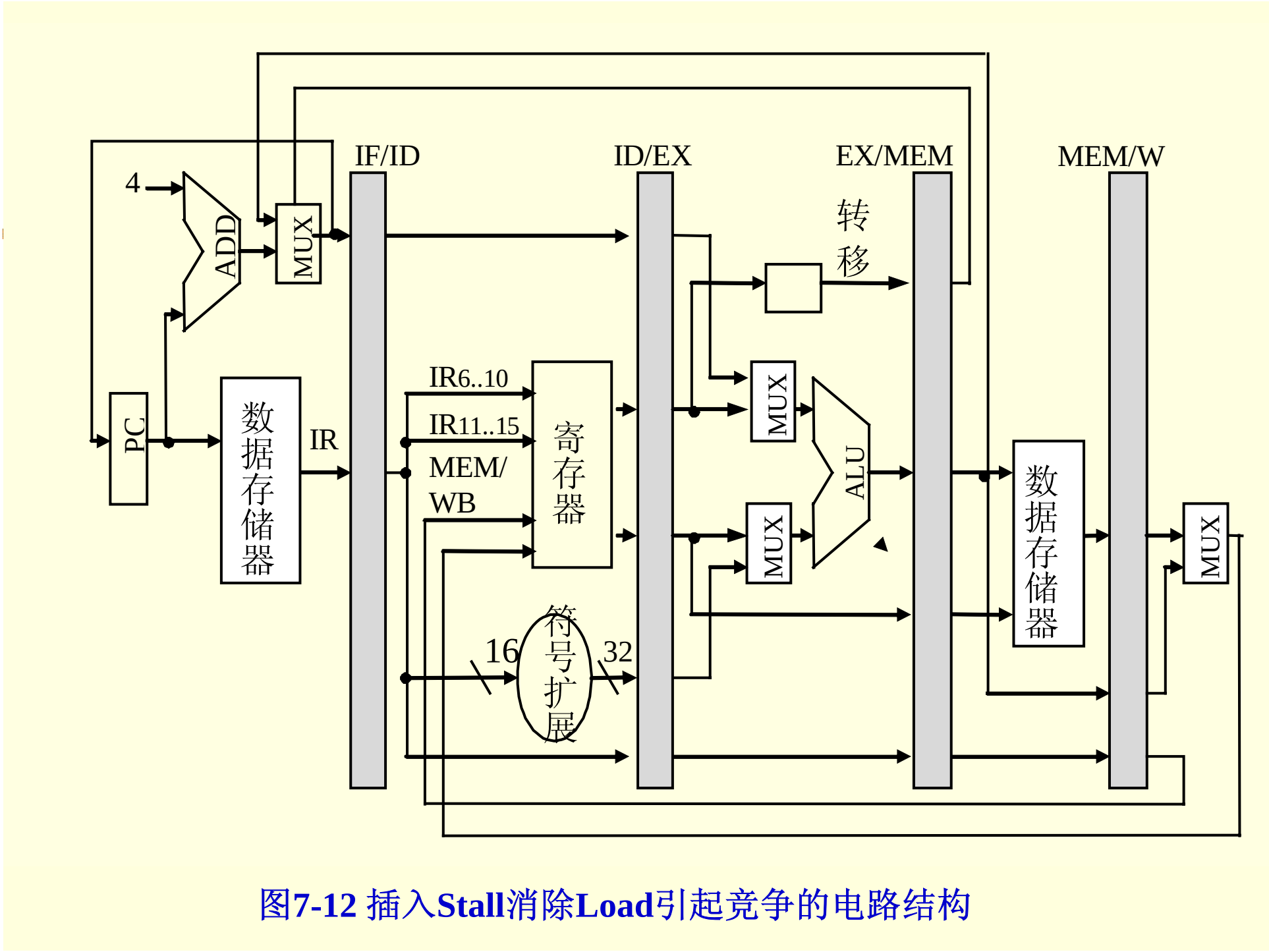


图7-12 插入Stall消除Load引起竞争的电路结构

7.2 流水线中的主要问题及处理

7.2.4 控制相关

在指令流水线中，为了减少因控制相关而引起的流水线性能下降，可采用如下方法：

（1）加快和提前形成条件码

（2）转移预测法

（3）优化延迟转移技术

7.2 流水线中的主要问题及处理

7.2.5 流水实现的关键技术

需要的解决关键技术有：

（1）首先必须保证在指令重叠时，不存在任何流水线资源冲突问题。

（2）解决ID段和WB段在使用寄存器文件时出现的数据相关问题。

（3）解决PC改写产生的控制竞争问题。

（4）使用流水线锁存器。

（5）配置不同用途的算术/逻辑运算部件。

（6）数据流向控制。

7.3流水线的性能评价

7.3.1 流水线的性能指标

1．流水线的主要性能指标

（1）吞吐率

$$\Delta t = \max\{t_1, \wedge t_i, \wedge t_m\} + t_j \qquad (7\text{-}1)$$

$$T_p = \frac{n}{m\Delta t + (n-1)\Delta t} = \frac{1}{\Delta t\left(1 + \frac{m-1}{n}\right)} = \frac{T_{p\max}}{1 + \frac{m-1}{n}} \qquad (7\text{-}2)$$

7.3流水线的性能评价

7.3.1 流水线的性能指标

1. 流水线的主要性能指标

（2）加速比

$$S_p = \frac{T_1}{T_k} = \frac{nm}{m+n-1} = \frac{m}{1+\frac{m-1}{n}} \qquad (7\text{-}3)$$

（3）使用效率

$$E = \frac{nm\Delta t}{m(m+n-1)\Delta t} = \frac{n}{m+n-1} = \frac{S_p}{m} = T_p\Delta t \qquad (7\text{-}4)$$

7.3流水线的性能评价

7.3.1 流水线的性能指标

2. CPU性能公式

$$CPU\text{时间} = \frac{\text{总时钟周期数}}{\text{时钟频率}} \qquad (7\text{-}5)$$

$$CPI = \frac{\text{总时钟周期数}}{IC} \qquad (7\text{-}6)$$

$$\text{总CPU时间} = \frac{\text{CPI} \times \text{IC}}{\text{时钟频率}} \qquad (7\text{-}7)$$

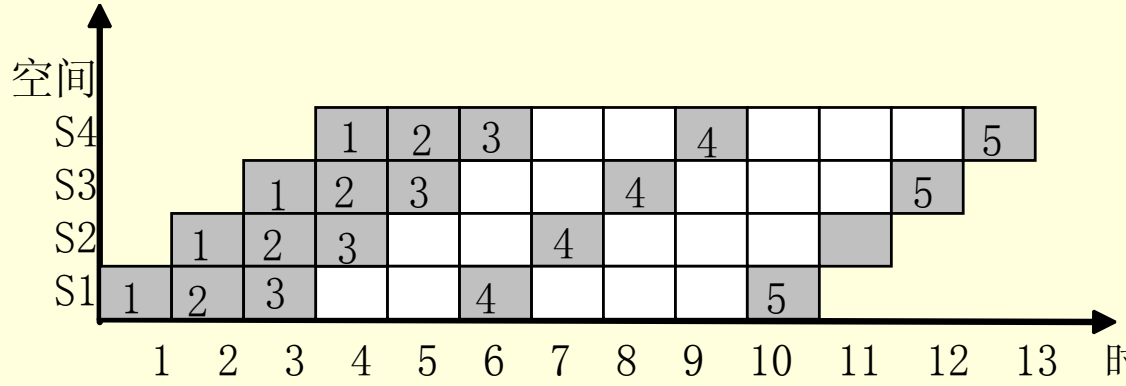
$$CPU\ \text{时间} = \frac{\sum_{i=1}^{n}(CPI_i \times IC_i)}{\text{时钟频率}} \qquad (7\text{-}8)$$

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times IC_i)}{IC} = \left(\sum_{i=1}^{n} CPI_i \times \frac{IC_i}{IC}\right) \qquad (7\text{-}9)$$

7.3流水线的性能评价

7.3.2 应用举例

1. 一般流水线的性能分析



7-13 用4段加法器求8个数和的流水线时空图

流水线的吞吐率***TP***为

$$TP = \frac{n}{T_k} = \frac{5}{13\Delta t}$$

流水线的加速比***S***为

$$S = \frac{T_0}{T_k} = \frac{4 \times 5 \times \Delta t}{13\Delta t} = \frac{20}{13} = 1.54$$

7.3流水线的性能评价

7.3.2 应用举例

2. 流水线延时与开销对流水线性能的影响

欲求若使用流水线，执行速度提高了几倍。计算方法是：

单条指令执行时间 = CC × 平均CPI = 10 ×（60% ×4 + 40% ×5）= 44ns

平均指令执行时间：CCpipeline = 11ns

于是得到：**Speedup = 44 /11 = 4**

欲求执行速度提高了几倍？计算方法如下：

平均指令执行时间 = 10+8+10+10+7 = 45ns，

而流水线时平均指令执行时间 = 11ns；

于是得到：Speedup = 45/11 = 4.1

7.3流水线的性能评价

7.3.2 应用举例

3．流水线障碍（流水线竞争）对流水线性能的影响

$$Speedup = \frac{\text{平均指令执行时间}_{unpipeline}}{\text{平均指令执行时间}_{pipeline}} = \frac{CPI_{unpipeline} \times CC_{unpipeline}}{CPI_{pipeline} \times CC_{pipeline}} \quad (7\text{-}10)$$

$$CPI\ pipeline = CPI\ ideal + \text{流水线}stall\text{周期}$$

$$= 1 + \text{流水线}stall\text{周期} \quad (7\text{-}11)$$

7.3流水线的性能评价

7.3.2 应用举例

3．流水线障碍（流水线竞争）对流水线性能的影响

$$Speedup = \frac{CPI.unpipeline}{CPI.pipeline} = \frac{CPI.unpipeline}{1 + 流水线stall周期} \qquad (7\text{-}12)$$

$$\frac{CCunpipeline}{CCpipiline} = 流水级$$

$$Speedup = \frac{流水级}{1 + 流水线stall周期} \qquad (7\text{-}13)$$

4. 结构竞争对流水线性能的影响

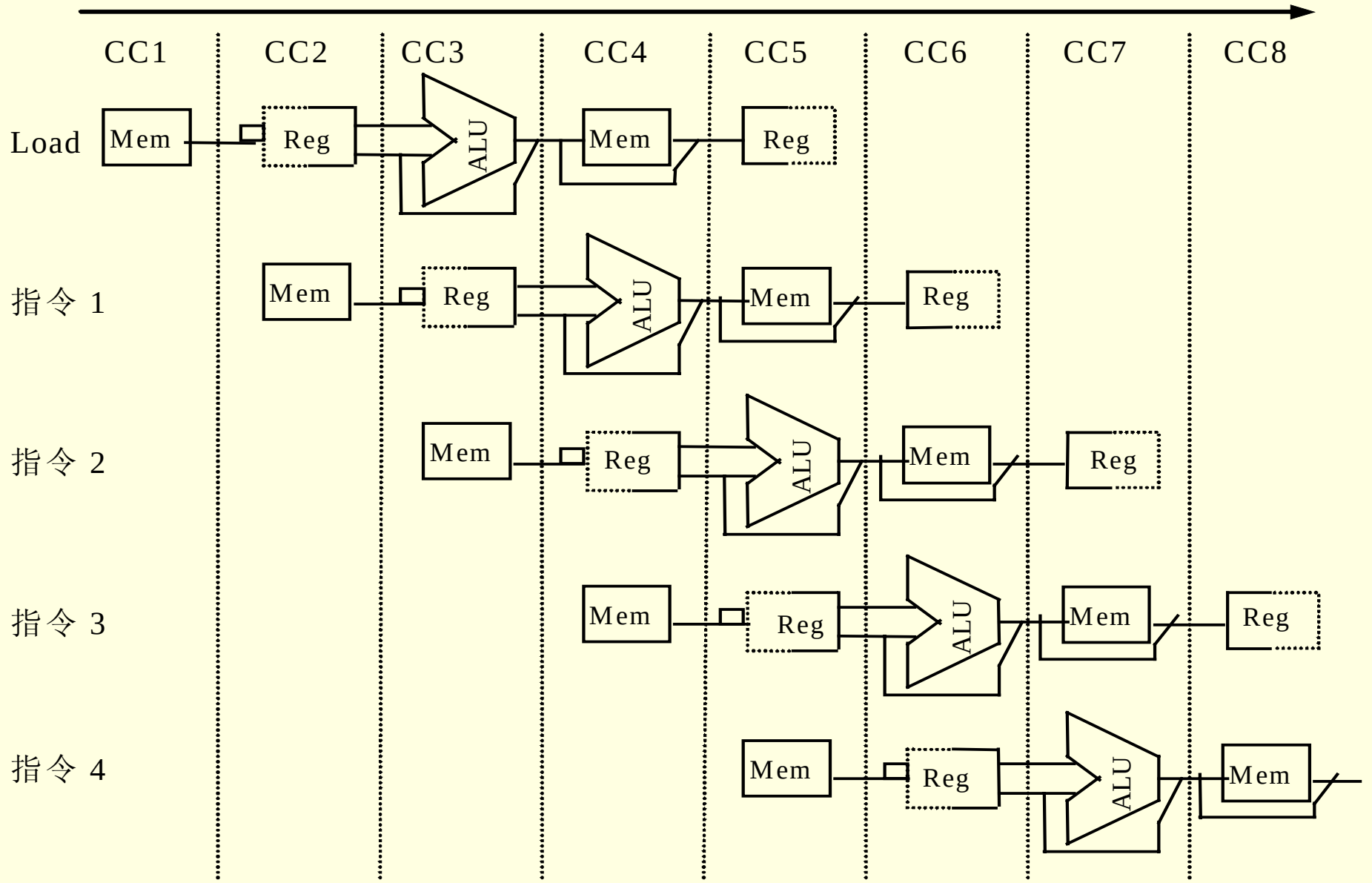


图7-14 7-13 结构竞争示意图

7.3流水线的性能评价

7.3.2 应用举例

4. 结构竞争对流水线性能的影响

指令序列	流水时钟数									
	1	2	3	4	5	6	7	8	9	10
Load 指令	IF	ID	EX	MEM	WB					
指令i+1		IF	ID	EX	MEM	WB				
指令i+2			IF	ID	EX	MEM	WB			
指令i+3				stall	IF	ID	EX	MEM	WB	
指令i+4						IF	ID	EX	MEM	WB
指令i+5							IF	ID	EX	MEM

图7-15 7-14 结构竞争流水线状态图

5．控制竞争对流水线性能的影响

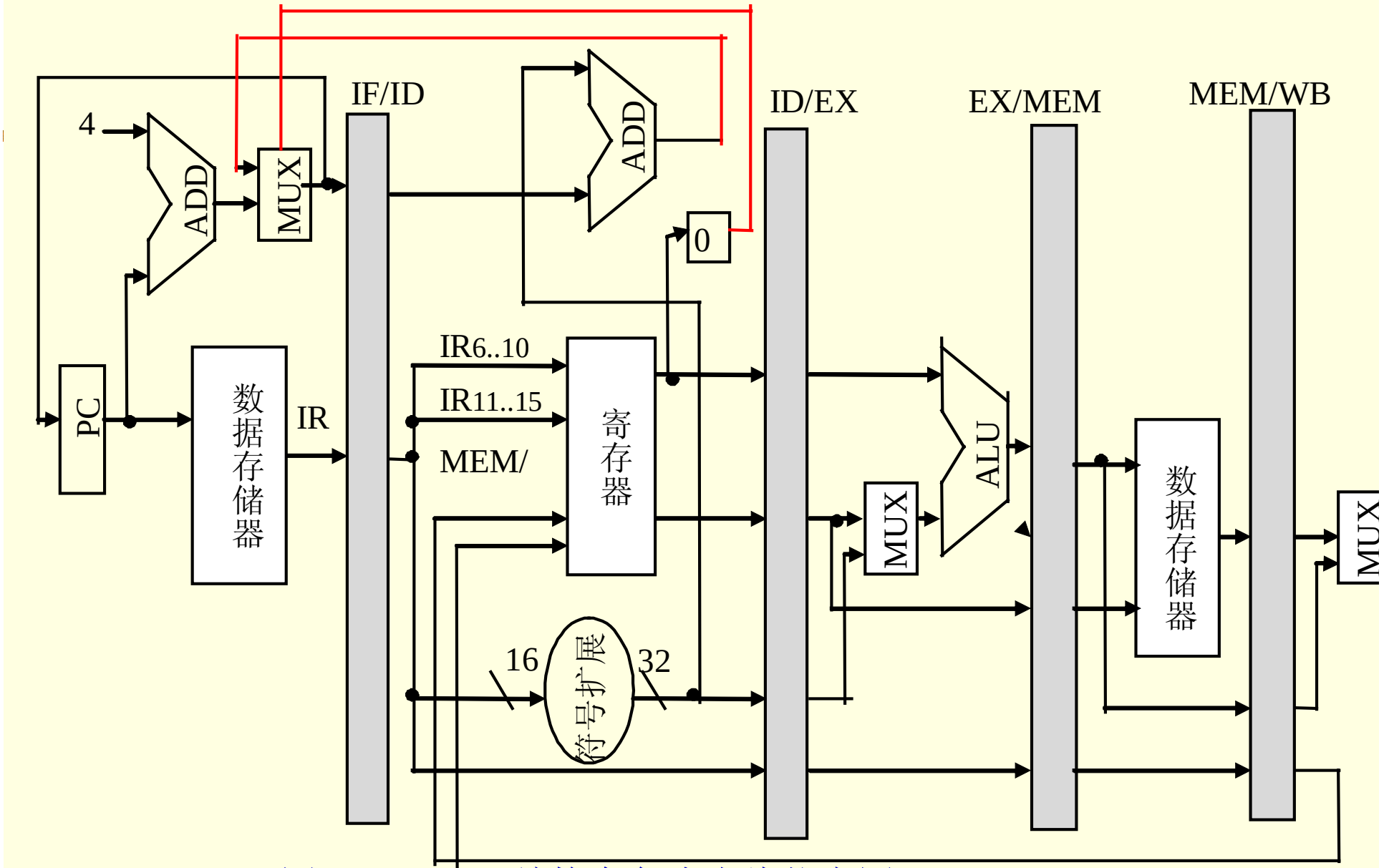


图7-15 7-14 结构竞争流水线状态图

7.3流水线的性能评价

7.3.2 应用举例

5. 控制竞争对流水线性能的影响

转移不成功指令	IF	ID	EX	MEM	WB				
指令i+1		IF	ID	EX	MEM	WB			
指令i+2			IF	ID	EX	MEM	WB		
指令i+3				IF	ID	EX	MEM	WB	
指令i+4					IF	ID	EX	MEM	WB

图7-17 预测成功无停顿

7.3流水线的性能评价

7.3.2 应用举例

5. 控制竞争对流水线性能的影响

转移成功指令	IF	ID	EX	MEM	WB				
指令i+1		IF	Idle	Idle	Idle	Idle			
目标指令			IF	ID	EX	MEM	WB		
目标指令+1				IF	ID	EX	MEM	WB	
目标指令+2					IF	ID	EX	MEM	WB

图7-18 预测失败停顿一个周期

7.3流水线的性能评价

7.3.3 Amdahl定律

$$Speedup = \frac{\text{使用增强部件后的性能}}{\text{未用增强部件时的整机性能}}$$

$$= \frac{\text{未用增强部件时执行一个任务的时间}}{\text{使用增强部件后执行该任务的时间}} = \frac{Told}{Tnew}$$

Amdahl定律还可以表为如下形式

$$Speedup = \frac{Told}{Tnew} = \frac{Told}{(1-F)Told + F/S} = \frac{1}{(1-F) + F/S}$$

习 题

7-1. 判断以下三组指令中各存在哪种类型的数据相关？

（1）I1　LAD　R1，A　；M（A）→ R1，M（A）是存储器单元

I2　ADD　R2，R1　；（R2）＋（R1）→ R2

（2）I3　ADD　R3，R4　；（R3）＋（R4）→ R3

I4　MUL　R4，R5　；（R4）＋（R5）→ R4

（3）I5　LAD　R6，B　；M（B）→ R6，M（B）是存储单元

I6　MUL　R6，R7　；（R6）×（R7）→ R6

7-2. 指令流水线有取指（IF）、译码（ID）、执行（EX）、访存（MEM）、写回寄存器堆（WB）五个过程段，现共有2条指令连续输入此流水线。画出流水处理的时空图，假设时钟周期为100ns。

习 题

7-3．假设有一个计算机系统分为四级，每一级指令都比它下面一级指令在功能上强M倍。即一条r+1级指令能够完成M条r指令的工作，且一条r+1指令需要N条r级指令解释。对于一段在第一级执行时间为K的程序，在第二、第三、第四级上的一段等效程序需要执行多少时间？

7-4．对于一台400MHz计算机执行标准测试程序，此程序中的指令类型，执行数量和平均时钟周期数如下表，求该计算机的有效CPI、MIPS和程序执行时间。

指令类型	指令执行数量	平均时钟周期数
整数	45 000	1
数据传送	75 000	2
浮点	8 000	4
分支	1 500	2

习 题

7-5．计算机系统中有三个部件可以改进，这三个部件的部件加速比如下：

部件加速比$_1$=30 ； 部件加速比$_2$=20 ；部件加速比$_3$=10

（1）如果部件1和部件2的可改进比例均为30%，那么当部件3的可改进比例为多少时，系统加速比才可以达到10%？

（2）如果三个部件的可改进比例分别为30%、30%和20%，三个部件同时改进，那么系统中不可加速部分的执行时间在总执行时间中占的比例是多少？

（3）如果相对某个测试程序，三个部件的可改进比例分别为20%、20%和70%．要达到最好改进效果，仅对一个部件改进时，要选择哪个部件？如果允许改进两个部件，又如何选择？

7-6．在流水线处理器中，可能有哪几种数据相关？这几种相关分别发生在什么情况下？解决操作数相关的方法有哪几种？

习 题

7-7. 假设在一台40MHz处理器上运行200 000条指令的目标代码，程序主要由四种指令组成。根据程序跟踪实验结果，已知指令混合比和每种指令所需的指令数如下：

指令类型	CPI	指令混合比
算术和逻辑	1	60%
高速缓存命中的加载/存储	2	18%
转移	4	12%
高速缓存缺失的存储器访问	8	10%

（1）计算在单处理机上用上述跟踪数据运行程序的平均CPI。

（2）根据（1）所得CPI，计算相应MIPS速率。

习 题

7-8. 对于一台40MHz计算机执行标准测试程序，程序中指令类型，执行数量和平均时钟周期数如下：

指令类型	指令执行数量	平均时钟周期数
整数	45 000	1
数据传送	75 000	2
浮点	8 000	4
分支	1 500	2

求该计算机的有效CPI、MIPS和程序执行时间。

7-9. 叙述Amdahl定律的主要内容。

实验与设计

实验7-1 乘法器实验

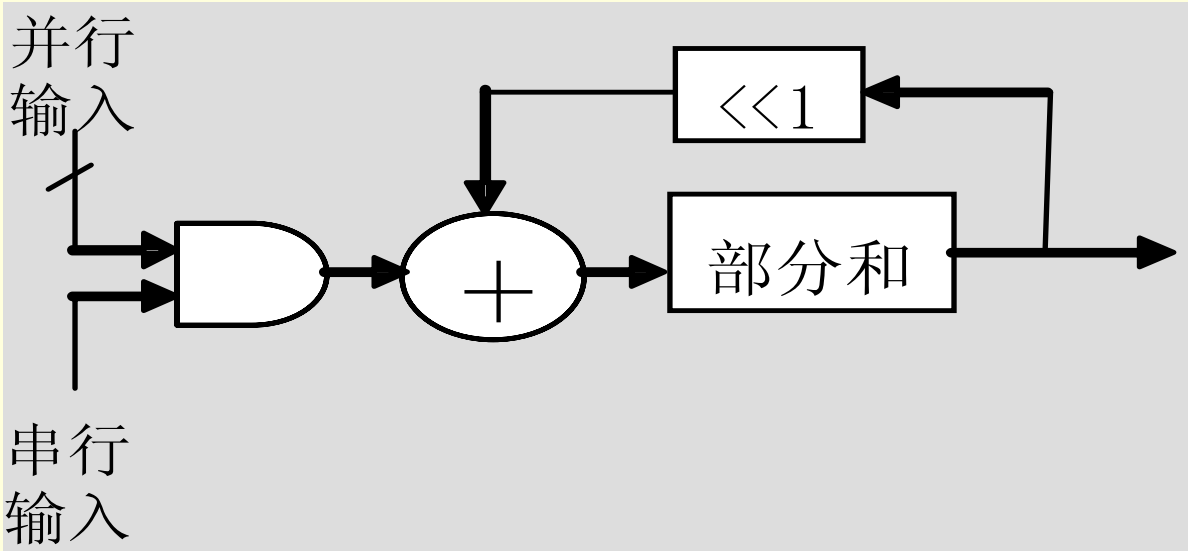


图7-19 最基本的硬件乘法器

实验与设计

实验7-1 乘法器实验

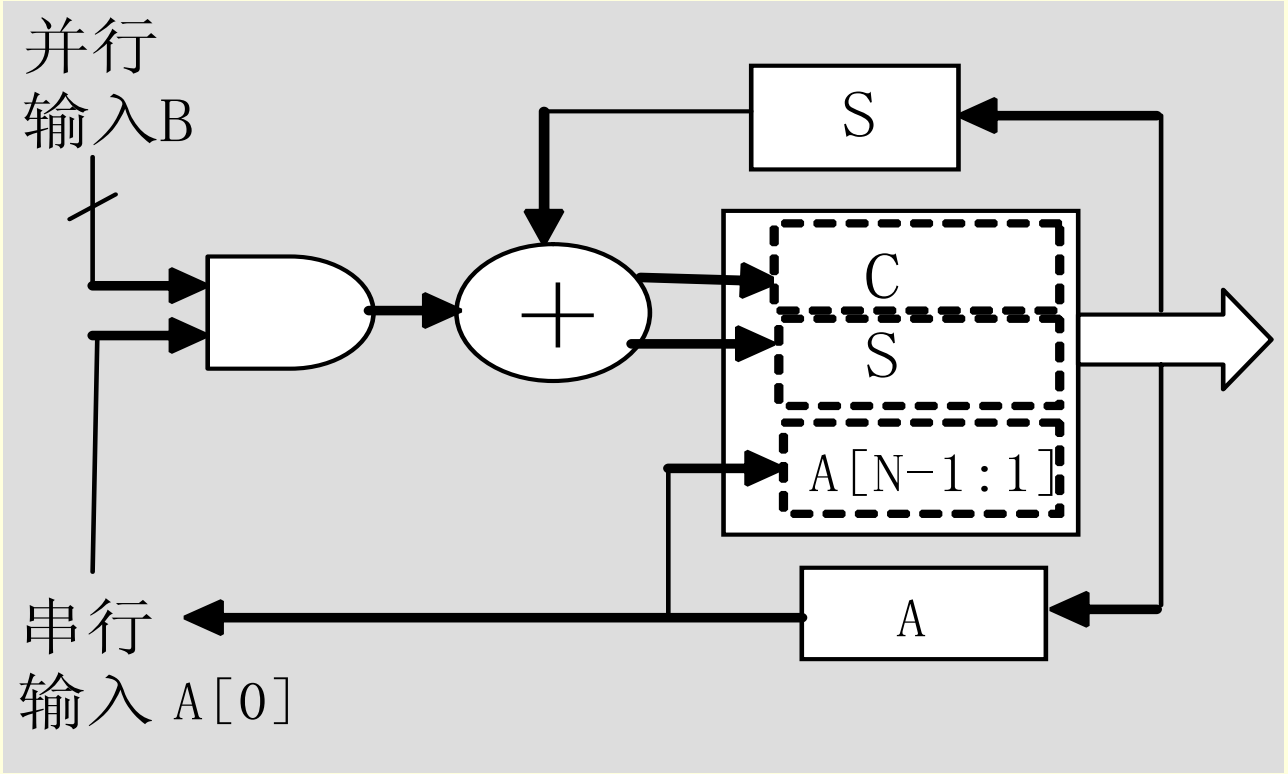


图7-20 改进后的硬件乘法器

【例7-5】

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity mult is
        port(mul_clk，ema，emb : in std_logic;
a，b : in std_logic_vector(15 downto 0);
                mul_a_out，mul_b_out : out std_logic_vector(15 downto 0));
end mult;
architecture rtl of mult is
        function and16_1(oper : in std_logic_vector(15 downto 0)；
                                                sel        :        in std_logic)
                        return std_logic_vector is
        begin
                if sel='1' then return oper;
                                                else return "0000000000000000"；
                end if;
        end function and16_1;
        signal   out0    :    std_logic_vector(31 downto 0);
        signal   a_temp0 :          std_logic_vector(30 downto 0);
        signal   a_temp1 :          std_logic_vector(29 downto 0);
        signal   a_temp2 :          std_logic_vector(28 downto 0);
        signal   a_temp3 :          std_logic_vector(27 downto 0);
        signal   a_temp4 :          std_logic_vector(26 downto 0);
```

接下页

```
signal a_temp5 :          std_logic_vector(25 downto 0);
        signal   a_temp6 :          std_logic_vector(24 downto 0);
        signal   a_temp7 :          std_logic_vector(23 downto 0);
        signal   a_temp8 :          std_logic_vector(22 downto 0);
        signal   a_temp9 :          std_logic_vector(21 downto 0);
        signal   a_temp10:          std_logic_vector(20 downto 0);
        signal   a_temp11:          std_logic_vector(19 downto 0);
        signal   a_temp12:          std_logic_vector(18 downto 0);
        signal   a_temp13:          std_logic_vector(17 downto 0);
        signal   a_temp14:          std_logic_vector(16 downto 0);
        signal   a_temp15:          std_logic_vector(15 downto 0);
begin
        process(mul_clk)
          begin
if rising_edge(mul_clk)   then
a_temp0(30 downto 15) <= and16_1(a，b(15))；a_temp0(14 downto 0) <= b"000000000000000" ；
a_temp1(29 downto 14) <= and16_1(a，b(14))；a_temp1(13 downto 0) <= b"00000000000000"  ；
a_temp2(28 downto 13) <= and16_1(a，b(13))；a_temp2(12 downto 0) <= b"0000000000000" ；
a_temp3(27 downto 12) <= and16_1(a，b(12))；a_temp3(11 downto 0) <= b"000000000000" ；
a_temp4(26 downto 11) <= and16_1(a，b(11))；a_temp4(10 downto 0) <= b"00000000000" ；
a_temp5(25 downto 10) <= and16_1(a，b(10))；a_temp5(9  downto 0) <= b"0000000000" ；
a_temp6(24 downto 9) <= and16_1(a，b(9)) ；a_temp6(8  downto 0) <= b"000000000" ；
a_temp7(23 downto 8) <= and16_1(a，b(8)) ；a_temp7(7  downto 0) <= b"00000000" ；
a_temp8(22 downto 7) <= and16_1(a，b(7)) ；a_temp8(6  downto 0) <= b"0000000" ；
a_temp9(21 downto 6) <= and16_1(a，b(6)) ；a_temp9(5  downto 0) <= b"000000" ；
```

接下页

```
a_temp10(20 downto 5) <= and16_1(a，b(5))；a_temp10(4 downto 0) <= b"00000"；
a_temp11(19 downto 4) <= and16_1(a，b(4))；a_temp11(3 downto 0) <= b"0000"；
a_temp12(18 downto 3) <= and16_1(a，b(3))；a_temp12(2 downto 0) <= b"000"；
a_temp13(17 downto 2) <= and16_1(a，b(2))；a_temp13(1 downto 0) <= b"00"；
a_temp14(16 downto 1) <= and16_1(a，b(1))；a_temp14(0)      <= '0'；
a_temp15          <= and16_1(a，b(0))；
end if;
end process;
process(a_temp0，a_temp1，a_temp2，a_temp3，a_temp4，a_temp5，a_temp6，a_temp7，
a_temp8，a_temp9，a_temp10，a_temp11，a_temp12，a_temp13，a_temp14，a_temp15)
variable        out1，c1 :          std_logic_vector(31 downto 0);
variable        out2       : std_logic_vector(29 downto 0);
variable        out3，c2 : std_logic_vector(27 downto 0);
variable        out4 : std_logic_vector(25 downto 0);
variable        out5，c3 : std_logic_vector(23 downto 0);
variable        out6 :std_logic_vector(21 downto 0);
variable        out7，c4 : std_logic_vector(19 downto 0);
variable        out8 : std_logic_vector(17 downto 0);
variable        c5 : std_logic_vector(31 downto 0);
variable        c6 : std_logic_vector(27 downto 0);
begin
out1 := ('0' & a_temp0) + a_temp1;
out2 := ('0' & a_temp2) + a_temp3；out3 := ('0' & a_temp4) + a_temp5；
out4 := ('0' & a_temp6) + a_temp7；out5 := ('0' & a_temp8) + a_temp9；
```

接下页

实验与设计

```
out6 := ('0' & a_temp10) + a_temp11 ;  out7 := ('0' & a_temp12) + a_temp13 ;
out8 := ('0' & a_temp14) + a_temp15 ;   c1 := out1 + out2 ;
c2   := out3 + out4 ;  c3   := out5 + out6 ;  c4   := out7 + out8 ;
c5   := c1 + c2 ;  c6 := ("0000" & c3)   + c4 ;  out0 <= c5 + c6;
end process;
process(ema，emb，out0)
begin
if ema='1'         then mul_a_out <= out0(15 downto 0) ;
else mul_a_out <= "ZZZZZZZZZZZZZZZZ" ;  end if ;
if emb='1'         then         mul_b_out <= out0(31 downto 16) ;
else mul_b_out <= "ZZZZZZZZZZZZZZZZ" ;  end if ;
          end process ;
end rtl ;
```

实验与设计

实验7-1 乘法器实验

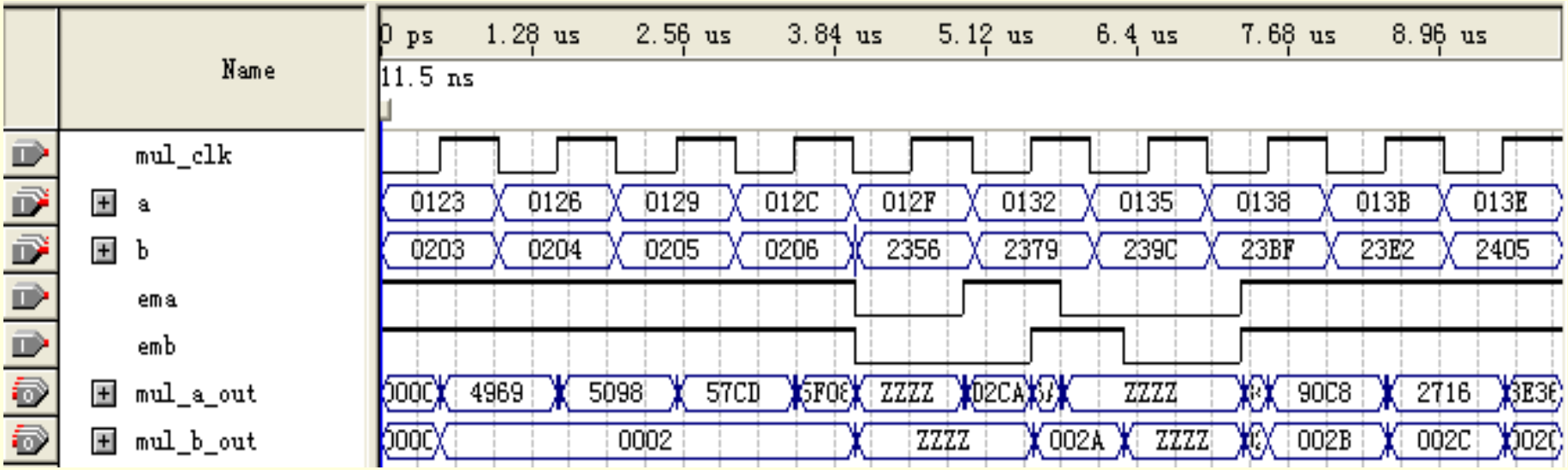


图7-21 乘法器仿真波形

实验与设计

实验7-1 乘法器实验

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	5.407 ns	a[10]	a_temp3[22]	--	mul_clk	0
2	Worst-case tco	N/A	None	16.665 ns	a_temp15[5]	mul_b_out[9]	mul_clk	--	0
3	Worst-case tpd	N/A	None	9.851 ns	ema	mul_a_out[9]	--	--	0
4	Worst-case th	N/A	None	0.037 ns	b[7]	a_temp8[22]	--	mul_clk	0
5	Total number of failed paths								0

图7-22 乘法器的时序分析

实验与设计

实验7-2 除法器实验

【例7-6】

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity div is
port(div_clk      :in std_logic;
ea，eb: in std_logic;
a，b              :in std_logic_vector(15 downto 0);
div_a_out，div_b_out    :          out std_logic_vector(15 downto 0));
end div;
architecture rtl of div is
signal  signa，signb      :          std_logic_vector(15 downto 0);
begin
process(div_clk)
```

接下页

```
variable a_temp，b_temp          : std_logic_vector(15 downto 0);
variable          p，q                          :          std_logic_vector(15 downto 0);
begin
if rising_edge(div_clk) then a_temp := a ; b_temp := b;
p := b"0000000000000000"；  q := b"0000000000000000";
for i in signa'range loop
p := p(14 downto 0) & a_temp(15);
a_temp := a_temp(14 downto 0) & '0'; p := p-b_temp;
if p(15) = '1' then q(i) :='0'；  p:= p+b_temp ；
else q(i):='1' ；   end if;
end loop;
signa <= q ；  signb <= p ；   end if ；
end process;
process(ea，eb，signa，signb)
begin
if ea='1' then div_a_out<=signa
else div_a_out<="ZZZZZZZZZZZZZZZZ"  end if ;
if eb='1' then div_b_out<=signb ;
else div_b_out<="ZZZZZZZZZZZZZZZZ" ；   end if ;
end process ;
end rtl ;
```

实验7-2 除法器实验

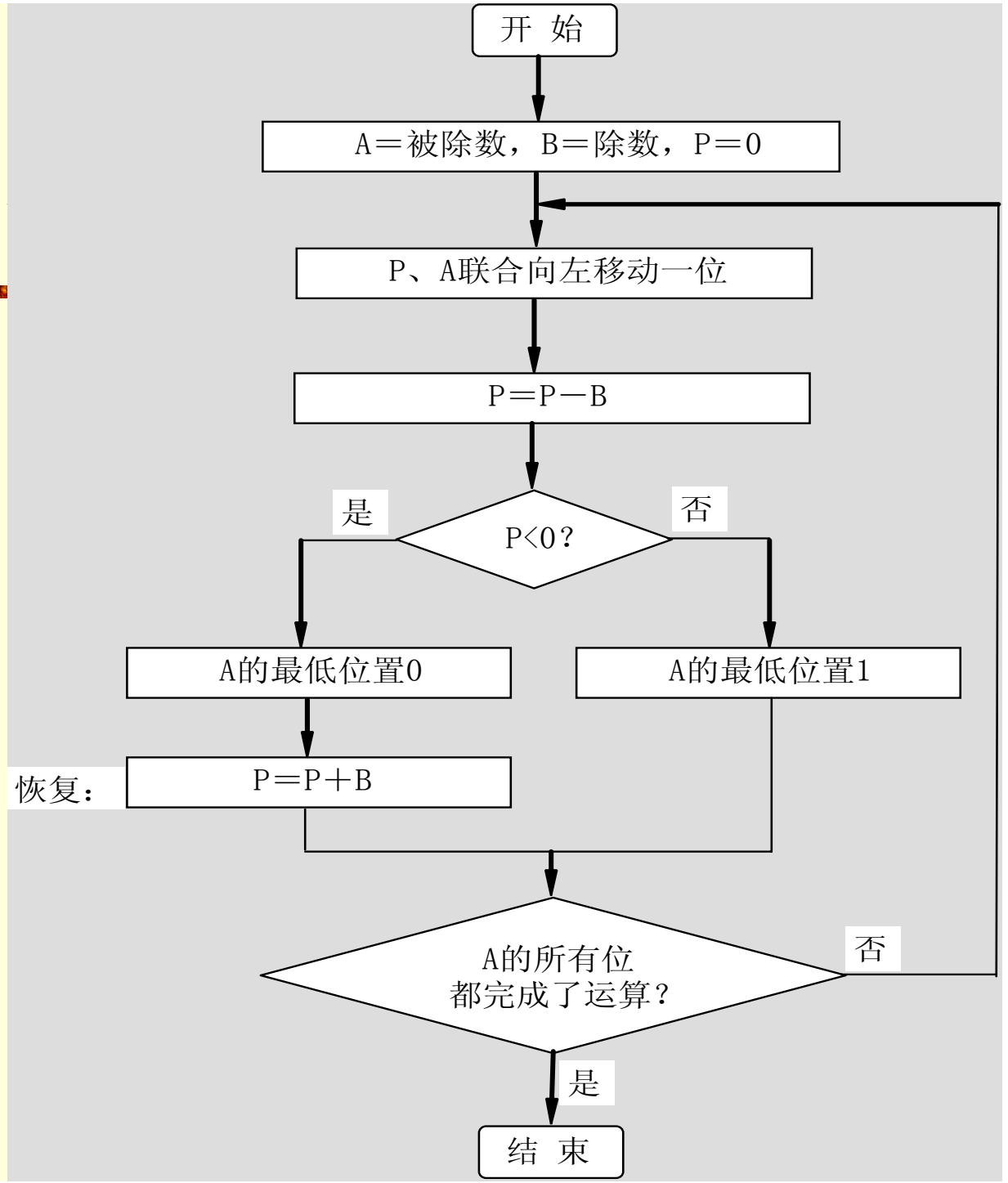


图7-23 除法运算流程图

实验与设计

实验7-2 除法器实验

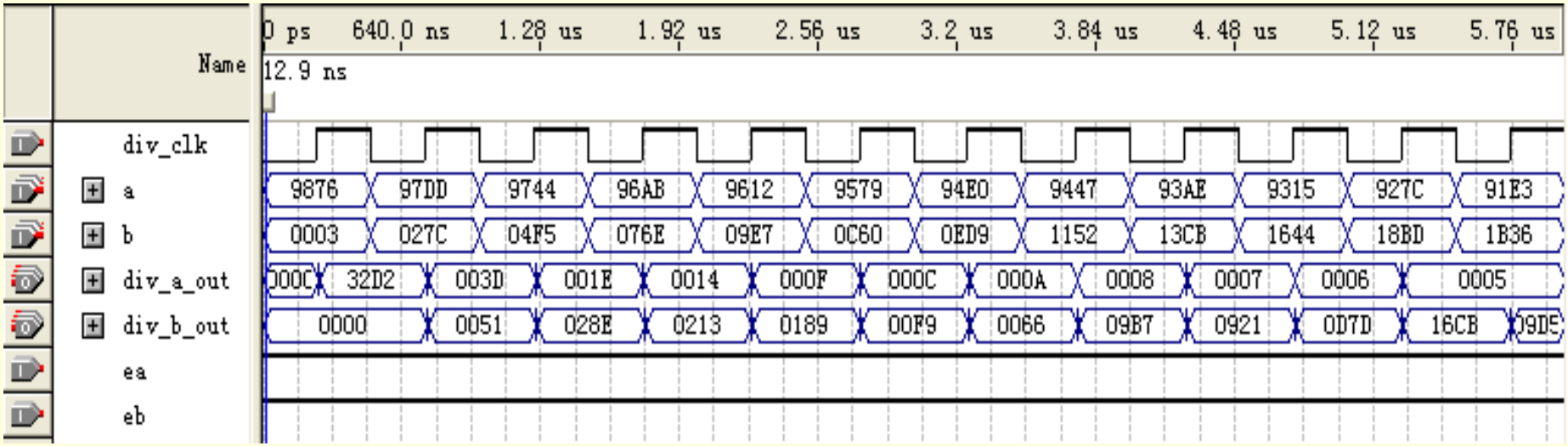


图7-24　16÷16除法器仿真运算结果