第7章

流水线CPU原理

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

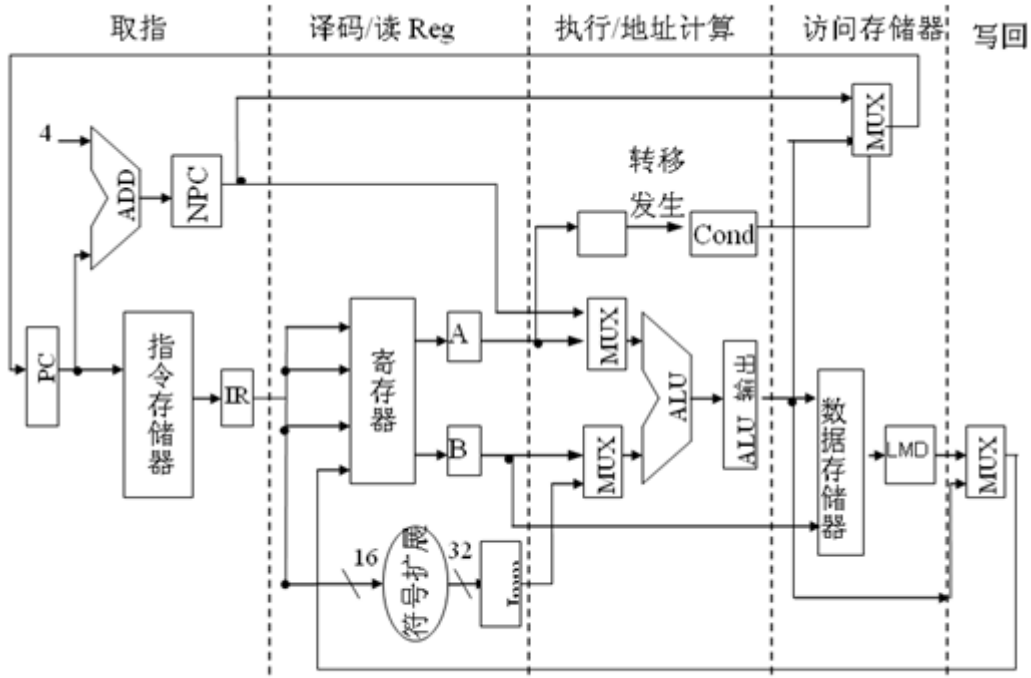


图 7-1 非流水线实现的指令解释数据通路

7.1 流水线的一般概念

7.1.1 DLX指令流水线结构

2．DLX基本指令流水线

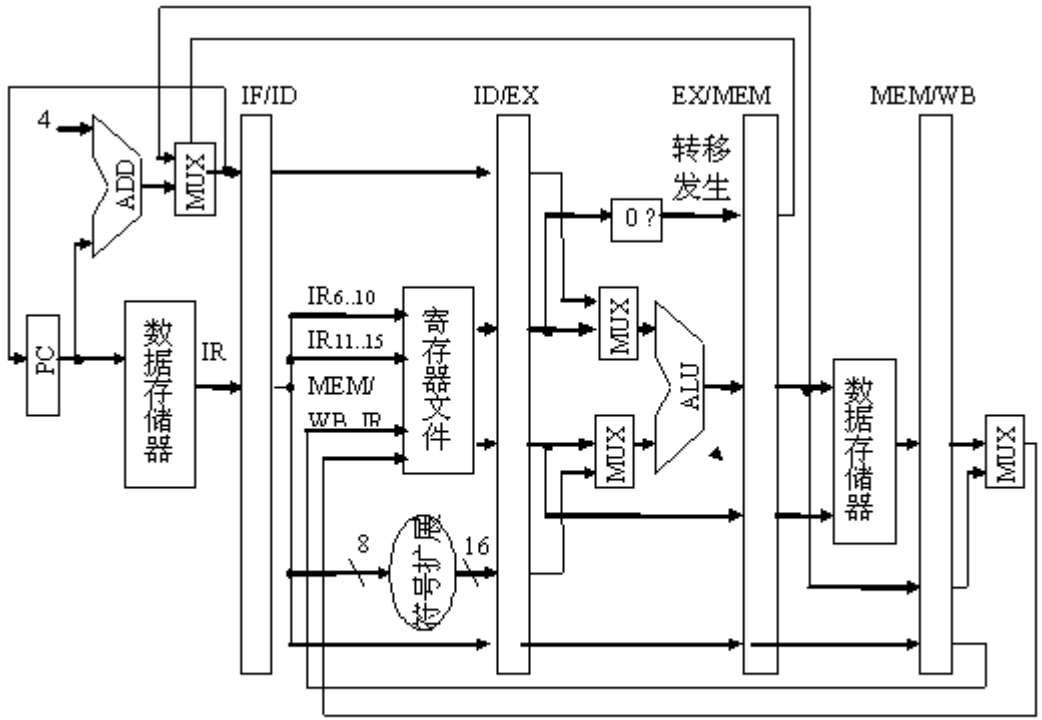


图 7-2 DLX 基本指令流水线

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

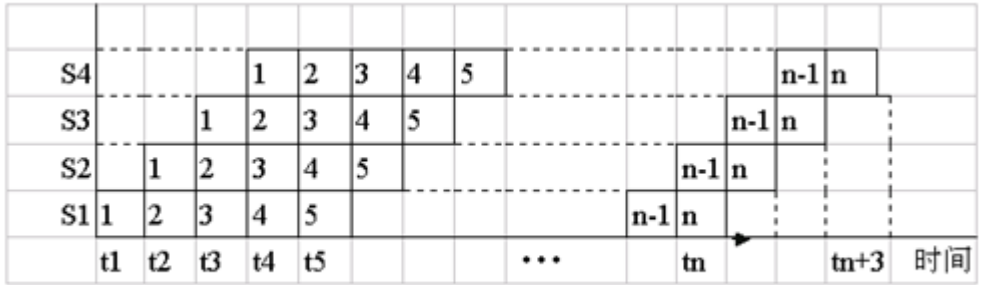
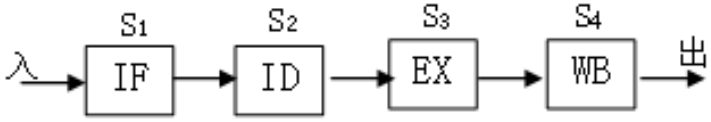


图 7-3 流水线时空图

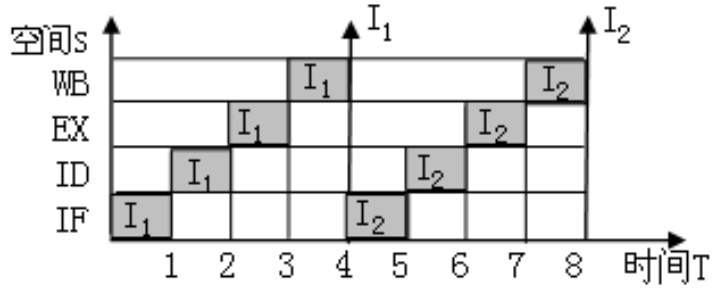
指令序列	流水时钟数								
	1	2	3	4	5	6	7	8	9
指令 i	IF	ID	EX	MEM	WB				
指令 i +1		IF	ID	EX	MEM	WB			
指令 i +2			IF	ID	EX	MEM	WB		
指令 i +3				IF	ID	EX	MEM	WB	
指令 i +4					IF	ID	EX	MEM	WB

图 7-4 指令流水线的时空图

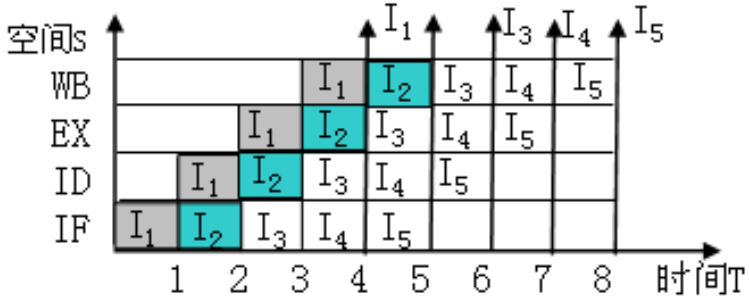
7.1 流水线的一般概念



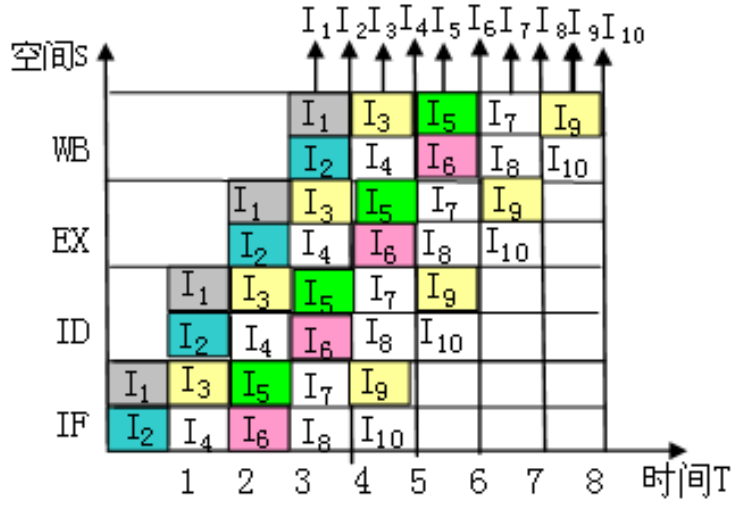
(a) 一个指令流水线过程段



b）非流水线时空图



(c) 流水线(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
- 算术流水线
- 处理机流水线

7.2 与流水线技术相关的问题及处理方法

7.2.1 资源相关及其冲突

7.2.2 数据相关及其分类

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

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

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

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

【例 7-1】

```
ADD     R1，R2，R3     ;     功能：  （R2）＋（R3）→R1
SUB     R4，R1，R5     ;     功能：  （R1）－（R5）→R4
AND     R6，R1，R7     ;     功能：  （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 与流水线技术相关的问题及处理方法

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
```

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

【例 7-3】

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

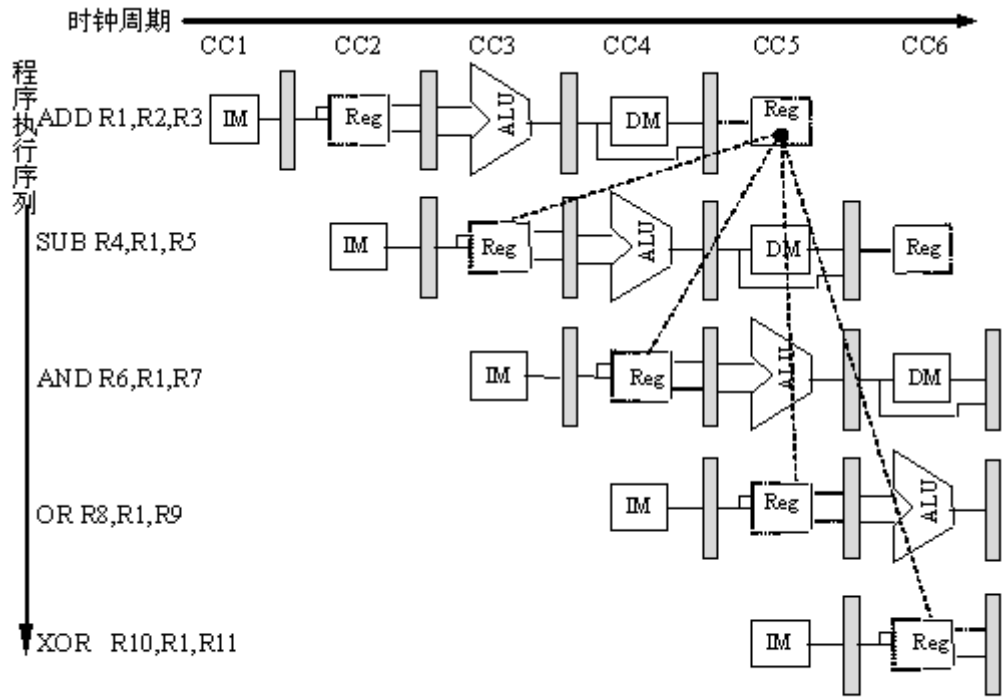


图 7-6 例 7-3 的流水状态图

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

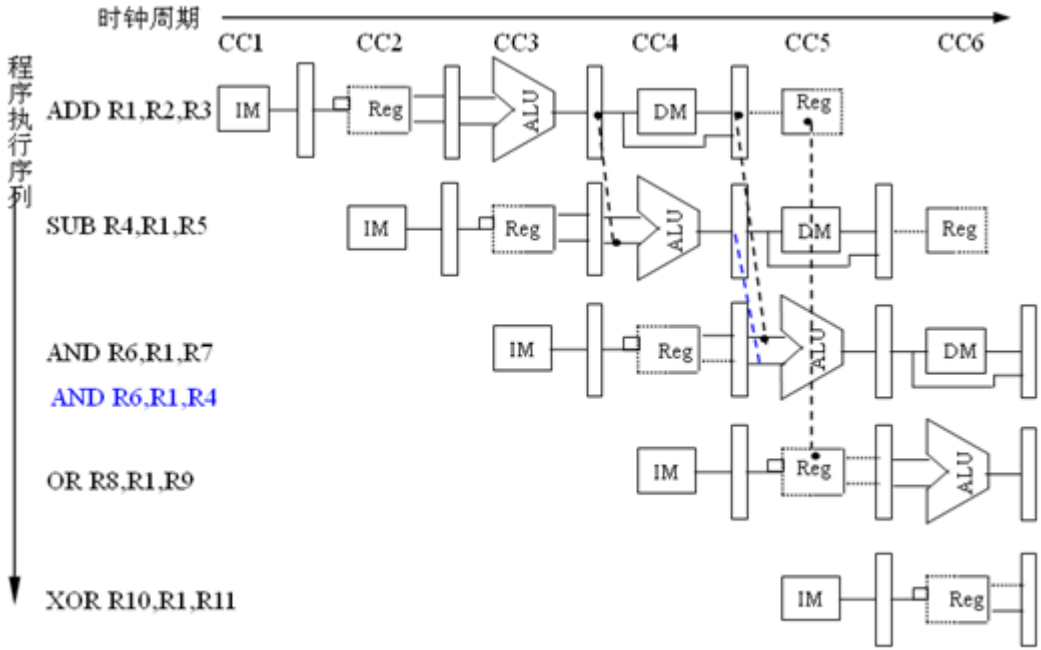


图 7-7 引入 forwarding path 后的状态图

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

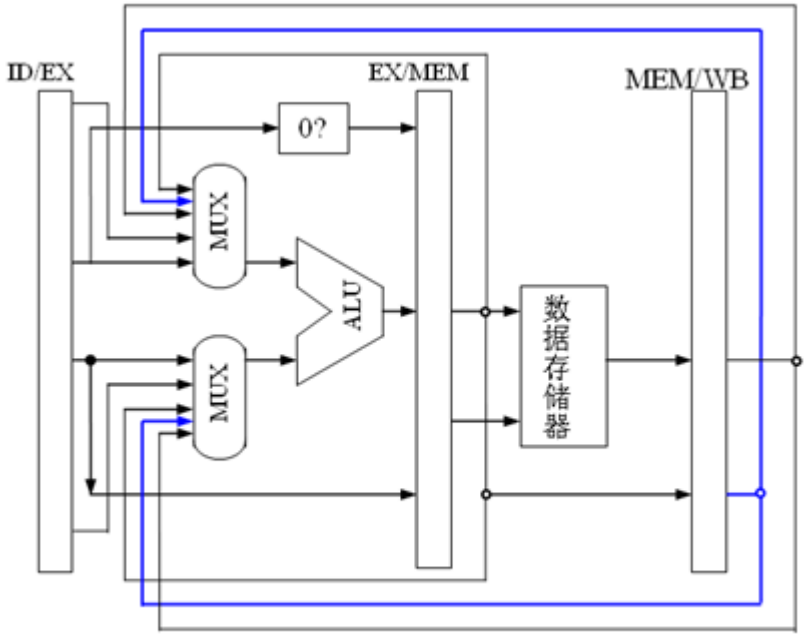


图 7-8 引入 forwarding path 的硬件示意图

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

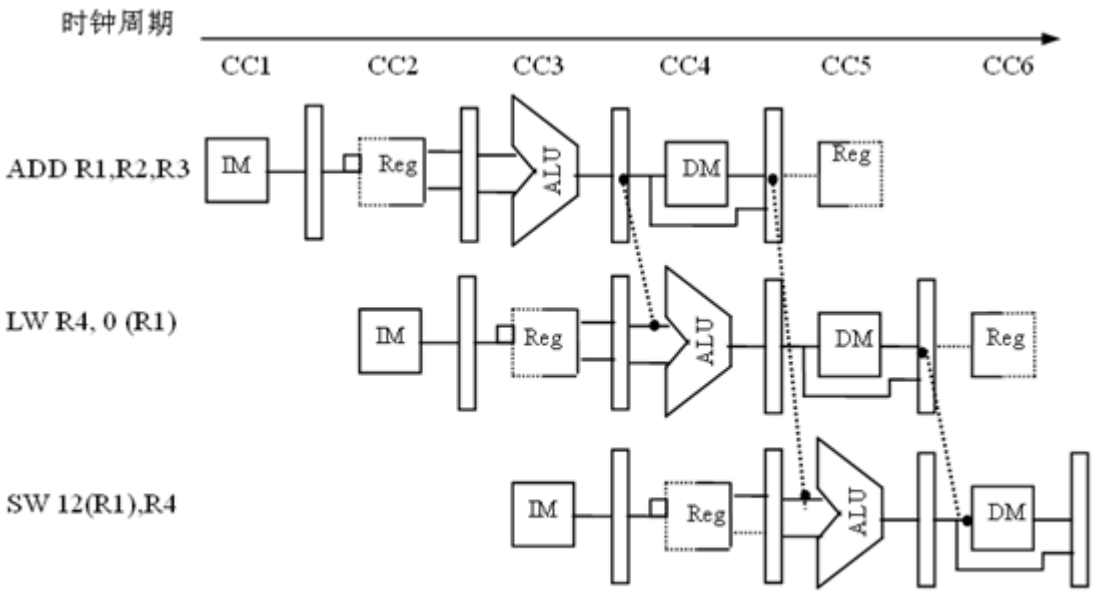


图 7-9 引入 forwarding 后消除了 Stall

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

【例 7-4】

LW　　R1，0（R2）

SUB　R4，R1，R5

AND　R6，R1，R7

OR　　R8，R1，R9

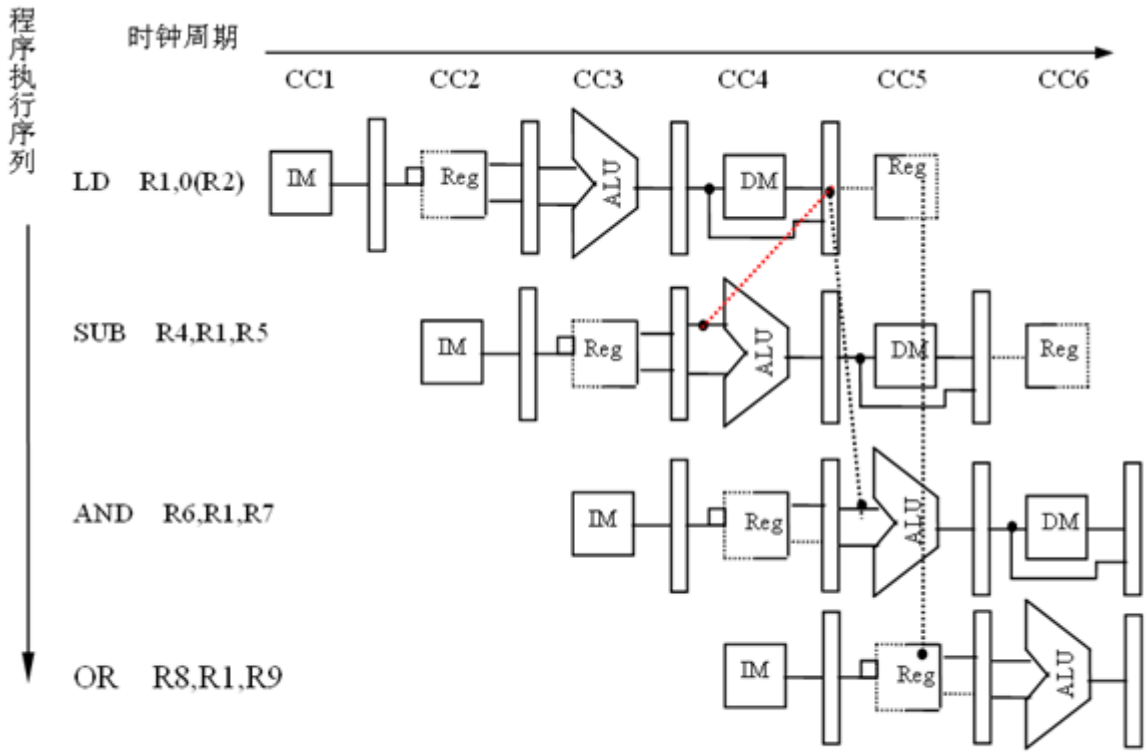


图 7-10　例 7-4 的流水线状态图

7.2 与流水线技术相关的问题及处理方法

7.2.3 数据竞争的处理技术

LD	IF	ID	EX	MEM	WB		ID/EX		
ADD		IF	ID	Stall	EX		MEM	WB	IF/ID
SUB			IF	Stall	ID		EX	MEM	WB

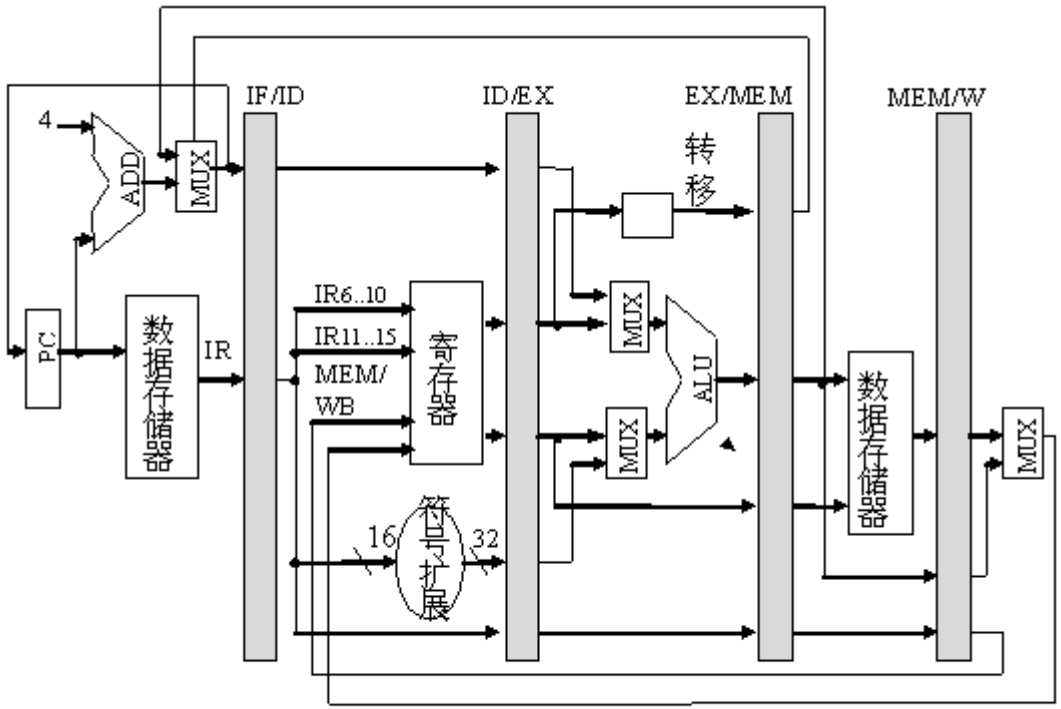


图 7-11 插入 Stall 消除 Load 引起竞争的电路结构

7.2 与流水线技术相关的问题及处理方法

7.2.4 控制相关

（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 与流水线技术相关的问题及处理方法

7.2.5 流水实现的关键技术

（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。

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

$$T = m\Delta t + (n-1)\Delta t$$

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

当$n >> m$时，有$T_p \approx T_{p\max}$

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。

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

当$n >> m$时，有$S_p \approx m$

（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

$$CPU\text{时间} = \text{总时钟周期数} / f_{CLOCK} \quad (7\text{-}5)$$

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

$$\text{总}CPU\text{时间} = \frac{CPI \times IC}{f_{CLOCK}} \quad (7\text{-}7)$$

$$CPU\text{时间} = \frac{\sum_{i=1}^{n}(CPI_i \times IC_i)}{f_{CLOCK}} \quad (7\text{-}8)$$

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

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

1. 一般流水线的性能分析

$$sum = [(A_1 + A_2) + (A_3 + A_4)] + (A_5 + A_6)$$

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

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

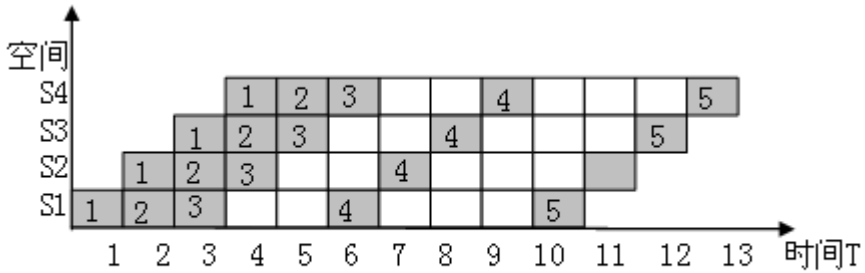


图 7-12 用 4 段加法器求 8 个数和的流水线时空图

$$\text{CPU 时间}_B = IC_B \times CPI_B \times \text{时钟周期}_B = 0.8 \times IC_A \times 1.25 \times (1.25 \times \text{时钟周期}_A)$$
$$= 1.25 \times IC_A \times \text{时钟周期}_A$$

$$\text{CPU 时间}_B = IC_B \times CPI_B \times \text{时钟周期}_B = 0.8 \times IC_A \times 1.25 \times (1.10 \times \text{时钟周期}_A)$$
$$= 1.10 \times IC_A \times \text{时钟周期}_A$$

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

2．流水线延时对其性能的影响

设在 4 段流水线浮点加法器中，若每段所需的时间为：T_1＝60ns，T_2＝50ns，T_3＝90ns，T_4＝80ns。求流水线加法器的加速比，以及若每段的时间都是 75ns（包括缓冲时间），求加速比。解法如下：

(1) 加法器的流水线时钟周期至少为 T＝90ns。若采用同样的逻辑电路(不使用流水线)，则浮点加法所需的时间是：

单条指令执行时间=T_1+ T_2+ T_3+ T_4＝60+50+90+80＝280ns

因此，加速比为 S=280/ 90≈3.1

(2) 若每个过程段的时间都是 75ns，则： 加速比 S＝（75×4）/ 75＝4

流水线性能计算的示例如下：

已知：unpipeline: CC=10ns；则 ALU(40%)、Branch(20%) 共 4CC；

Load/Store (40%) ，则有 5CC； pipeline 有： CC＝(10+1)＝11ns

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

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

3．流水线竞争对流水线性能的影响

（1）结构竞争。
（2）数据竞争。
（3）控制竞争。

$$Speedup = \frac{\text{平均指令执行时间}unpipeline}{\text{平均指令执行时间}pipeline} = \frac{CPIunpipeline \times CCunpipeline}{CPIpipeline \times CCpipeline} \quad (7\text{-}10)$$

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

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

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

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

4. 结构竞争对流水线性能的影响

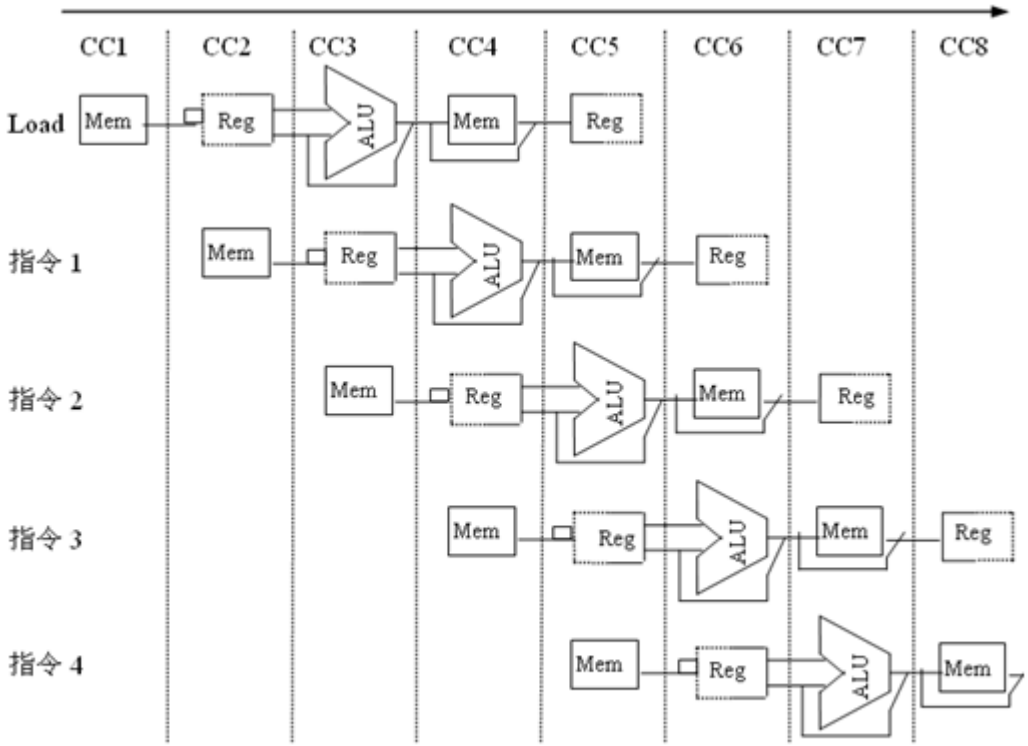


图 7-13 结构竞争示意图

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

4. 结构竞争对流水线性能的影响

平均指令执行时间 $=IC\times CPIhazard\times CChazard=IC\times(1+0.4\times1)\times CCideal/1.05$
$=1.3\times IC\times CCideal$

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 7-14 结构竞争流水线状态图

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

5. 控制竞争对流水线性能的影响

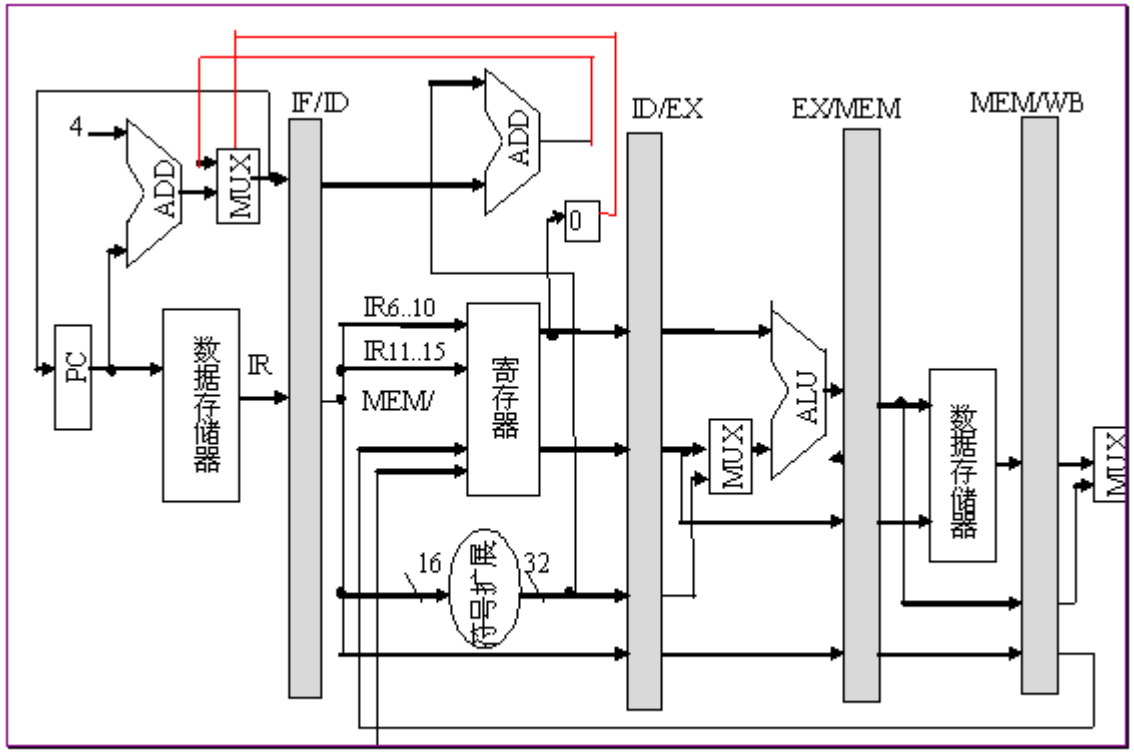


图 7-15 目标地址计算和条件判断提前后的流水线

7.3 流水线的性能评价

7.3.2 流水线性能评估举例

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 7-16 预测成功无停顿

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 7-17 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

$$Speedup = \frac{\text{使用增强部件后的性能}}{\text{未用增强部件时的整机性能}} = \frac{\text{未用增强部件时执行一个任务的时间}}{\text{使用增强部件后执行该任务的时间}} = \frac{Told}{Tnew}$$

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

实验与设计

7-1 高速硬件乘法器设计实验

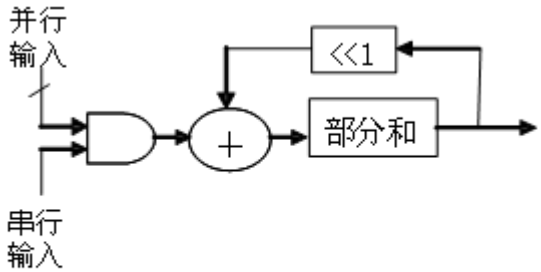


图7-18 最基本的硬件乘法器

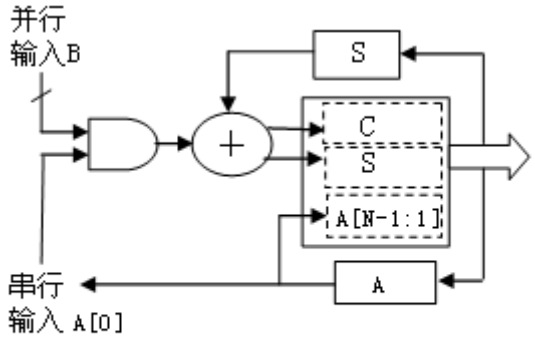


图7-19 改进后的硬件乘法器

图 7-20 乘法器仿真波形

【例 7-5】

```
module MULT16  (input [15:0] A,B,input CLK, output reg [31:0] DATOUT);
   reg [30:0] at0 ;   reg [29:0]  at1 ;   reg [28:0]  at2 ;
   reg [27:0] at3 ;   reg [26:0]  at4 ;   reg [25:0]  at5 ;
   reg [24:0] at6 ;   reg [23:0]  at7 ;   reg [22:0]  at8 ;
   reg [21:0] at9 ;   reg [20:0]  at10;   reg [19:0]  at11;
   reg [18:0] at12;   reg [17:0]  at13;   reg [16:0]  at14;
   reg [15:0] at15;   reg [31:0] out1,c1; reg [29:0] out2;
   reg [27:0] out3,c2;  reg [25:0] out4;    reg [23:0] out5,c3;
   reg [21:0] out6;     reg [19:0] out7,c4; reg [17:0] out8;
   reg [31:0] c5;   reg [27:0] c6;
  always @(posedge CLK)    begin
  at0 <={(B[15]? A:0),15'H0000}; at1 <={(B[14]? A:0),14'H0000};
  at2 <={(B[13]? A:0),13'H0000}; at3 <={(B[12]? A:0),12'H000};
  at4 <={(B[11]? A:0),11'H000};  at5 <={(B[10]? A:0),10'H000};
  at6 <={(B[9] ? A:0), 9'H000};  at7 <={(B[8] ? A:0), 8'H00};
  at8 <={(B[7] ? A:0), 7'H00};   at9 <={(B[6] ? A:0), 6'H00};
  at10<={(B[5] ? A:0), 5'H00};   at11<={(B[4] ? A:0), 4'H0};
  at12<={(B[3] ? A:0), 3'b000};  at13<={(B[2] ? A:0), 2'b0};
  at14<={(B[1] ? A:0), 1'b0};    at15<={(B[0] ? A:0)};
  end
 always @(*)    begin
  out1 = {1'b0,at0} + at1; out2={1'b0,at2}+at3;
  out3 = {1'b0,at4} + at5; out4={1'b0,at6} + at7;
  out5 = {1'b0,at8} + at9; out6 = {1'b0,at10} + at11;
  out7 = {1'b0,at12} + at13; out8 = {1'b0,at14} + at15;
  c1 = out1 + out2 ; c2 = out3 + out4 ;
  c3 = out5 + out6 ; c4 = out7 + out8 ;
  c5 = c1 + c2 ; c6 = {4'b0000,c3}+ c4 ;
  DATOUT <= c5 + c6;
 end
endmodule
```

实验与设计

7-2 高速硬件除法器设计实验

【例7-6】

```
module DIV16 (input CLK,
input [15:0] A,B,
output reg [15:0] QU,RE);
reg [15:0] AT,BT,P,Q;  integer i;
always @(posedge CLK)  begin
  AT = A ;  BT = B;
  P = 16'H0000; Q = 16'H0000 ;
 for (i=15; i>=0; i=i-1)  begin
  P = {P[14:0], AT[15]};
  AT = {AT[14:0],1'B0} ; P=P-BT;
  if (P[15]==1)  begin  Q[i]=0;
   P = P+BT ; end
  else Q[i]=1 ; end
end
always @( * )  begin
QU = Q; RE = P ; end
endmodule
```

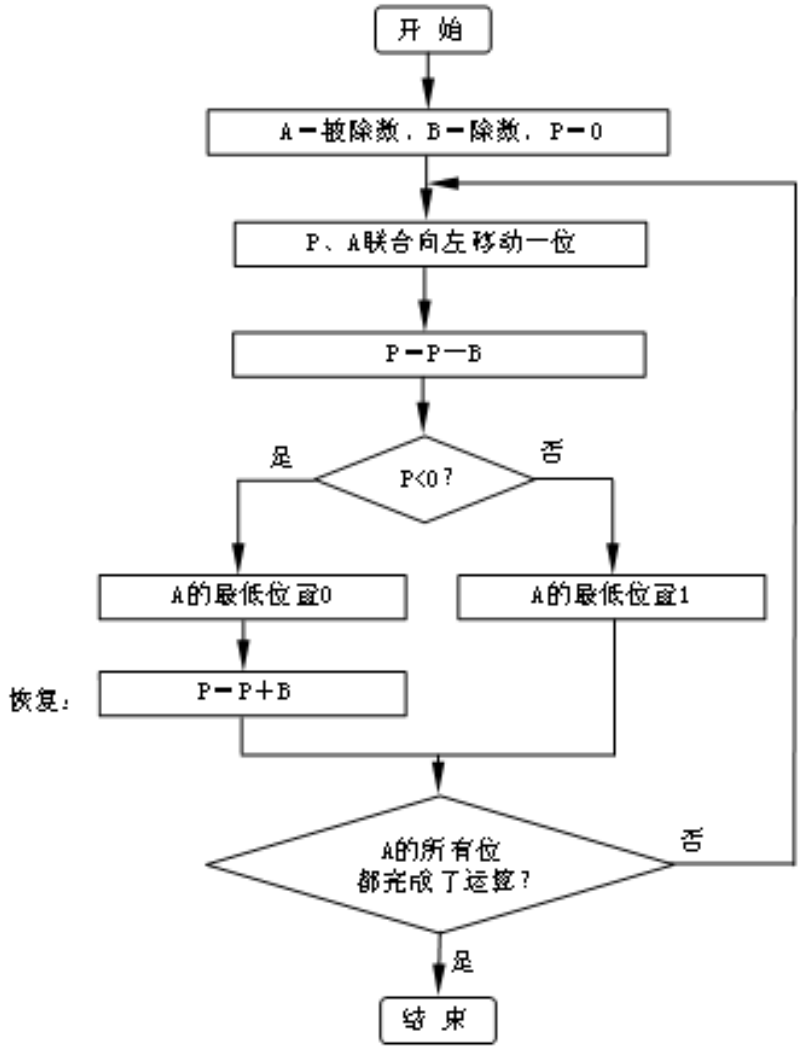


图 7-21 除法运算流程图

实验与设计

7-2 高速硬件除法器设计实验

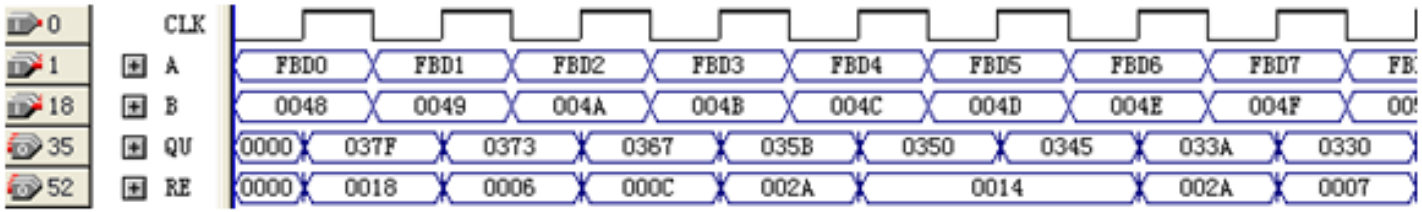


图 7-22　**16÷16** 除法器仿真运算结果

实验与设计

7-3 CACHE实验

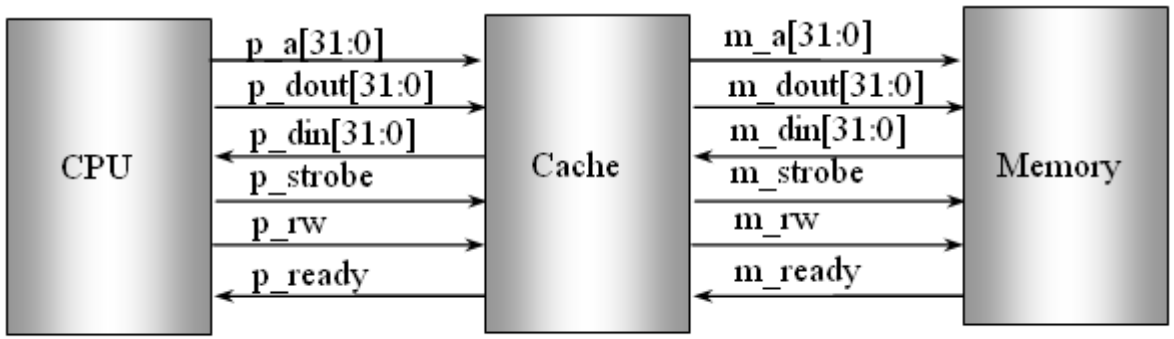


图 7-23　Cache 与 CPU 和存储器之间的接口信号

实验与设计

7-3 CACHE实验

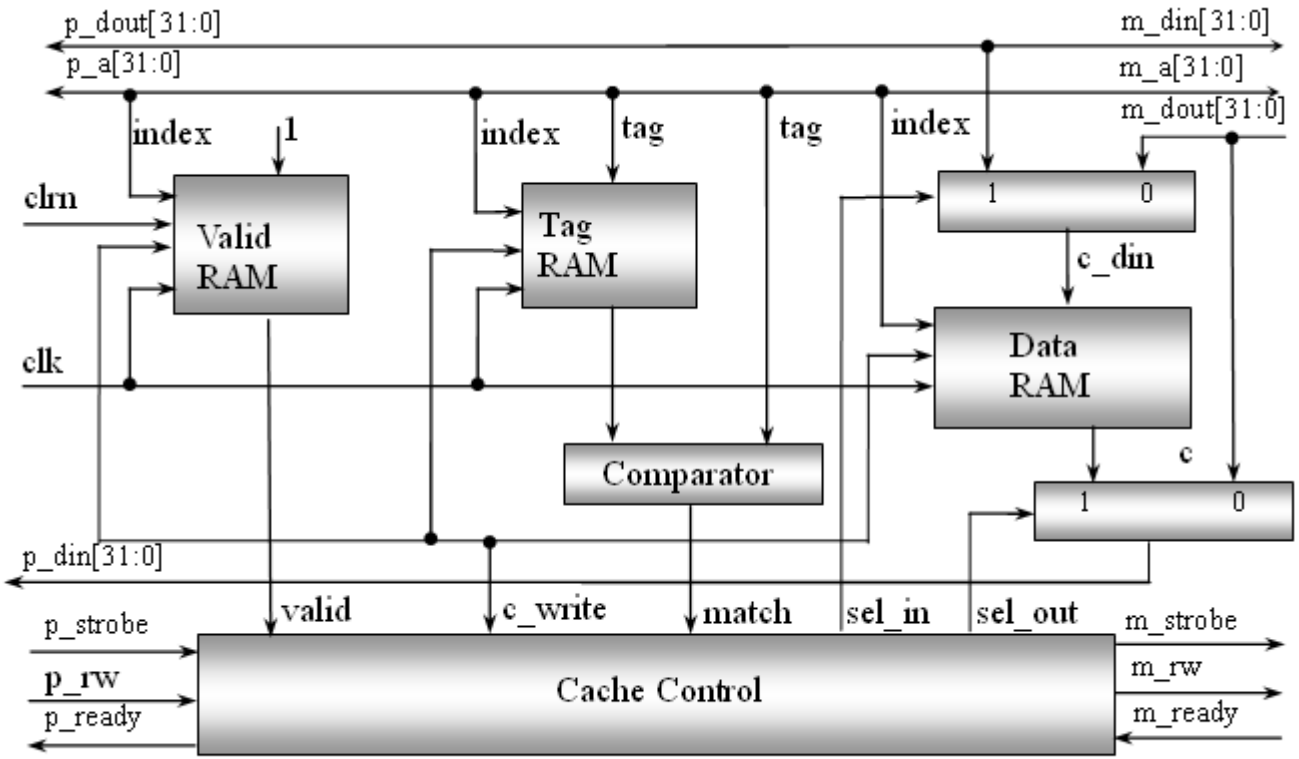


图 7-24　直接映像 Cache 内部结构图

实验与设计

【例7-7】

```
module cache  (p_a, p_dout, p_din, p_strobe, p_rw, p_ready, clk,
       clrn,  m_a, m_dout, m_din, m_strobe, m_rw, m_ready,cache_hit);
  parameter A_WIDTH = 32, C_INDEX = 6;
  input [A_WIDTH-1:0] p_a; output [A_WIDTH-1:0]m_a;
  input  [31:0] p_dout, m_dout; output [31:0] p_din, m_din;
  input  p_strobe, p_rw,  clk,clrn, m_ready;
  output p_ready, m_strobe, m_rw, cache_hit;
  localparam T_WIDTH =A_WIDTH - C_INDEX -2;   //  1 block = 1 word
  reg d_valid [0:(1<<C_INDEX)-1];
  reg [T_WIDTH-1:0] d_tags [0:(1<<C_INDEX)-1];
  reg [31:0]  d_data [0:(1<<C_INDEX)-1];
  wire [C_INDEX-1:0]  index = p_a[C_INDEX+1 :2];
  wire [T_WIDTH-1:0]  tag = p_a[A_WIDTH-1:C_INDEX+2];
always @(posedge clk or negedge clrn ) //write to cache
      if (clrn == 0)  begin   integer i;
        for  (i =0; i<(1<<C_INDEX); i=i+1)    d_valid[i]<=1'b0;
      end else if (c_write) d_valid[index] <= 1'b1;
always @(posedge clk)
   if (c_write) begin   d_tags[index]<=tag;  d_data[index]<=c_din; end
```

接下页

实验与设计

7-3 CACHE实验

```
//read from cache
wire valid=d_valid[index]; wire [31:0] c_dout=d_data[index];
wire [T_WIDTH-1:0]  tagout=d_tags[index];
//cache control
assign  cache_hit = valid & (tagout == tag);    //hit
wire    cache_miss  = ~cache_hit;
assign  m_din = p_dout;   assignm_a  = p_a;
assign  m_rw     = p_strobe & p_rw ;    //write through
assign  m_strobe = p_strobe & (p_rw | cache_miss);
assign  p_ready  =~p_rw & cache_hit |(cache_miss |p_rw) & m_ready;
wire    c_write  = p_rw | (cache_miss  & m_ready );
wire sel_in = p_rw;  wire sel_out   = cache_hit;
wire    [31:0]c_din = sel_in  ? p_dout : m_dout;
assign      p_din   = sel_out ? c_dout : m_dout;  endmodule
```

实验与设计

7-3 CACHE实验

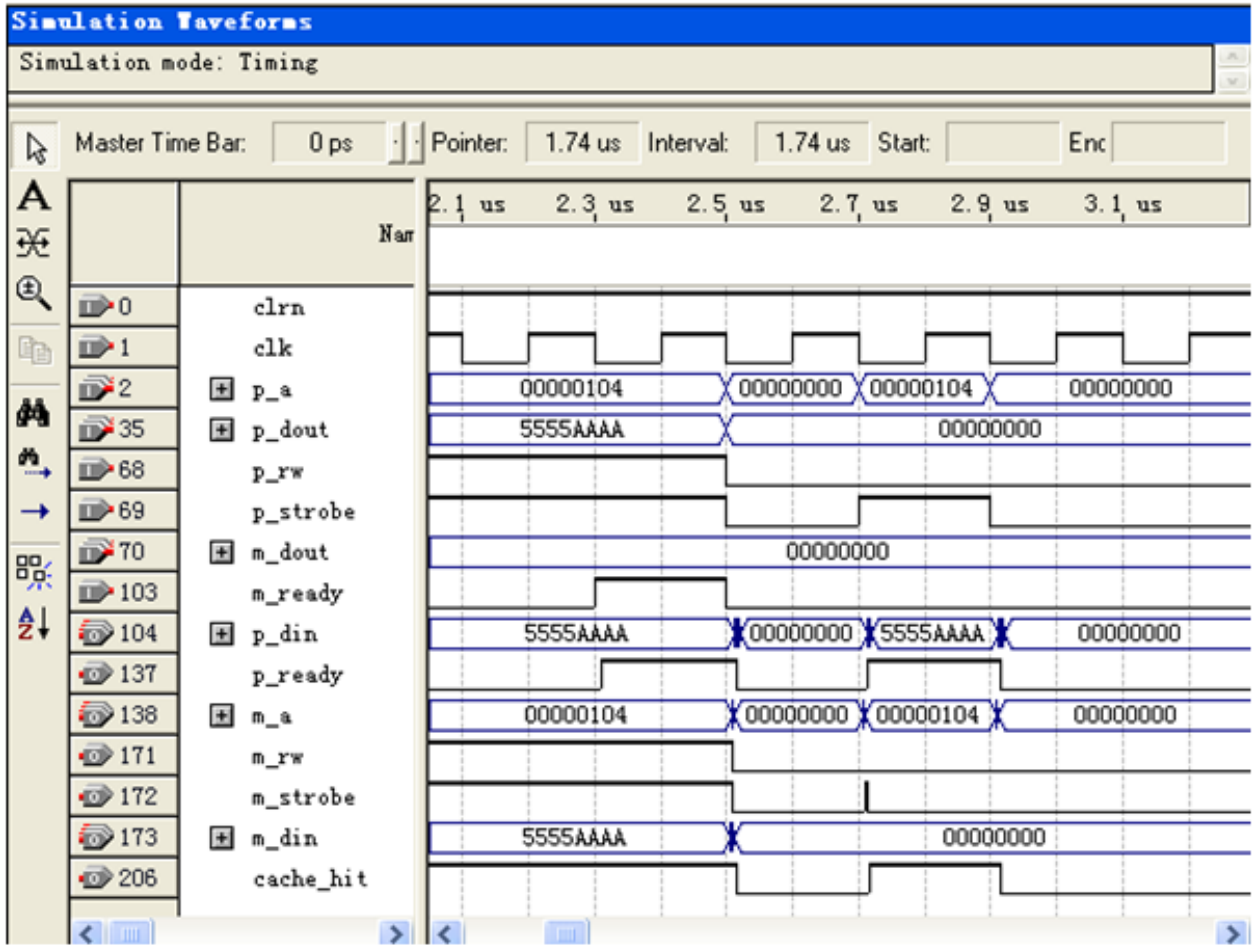


图 7-25 数据 Cache 仿真波形（写操作）

实验与设计

7-3 CACHE实验