[19]中华人民共和国专利局

[51]Int.Cl[6]
G06F 9/00

[12] 发明专利申请公开说明书

[21] 申请号 97123134.6

[43]公开日 1998 年 6 月 24 日

[11] 公开号 CN 1185606A

[22]申请日 97.11.19

[30]优先权

[32]96.12.20[33]US[31]770349

[71]申请人 国际商业机器公司

地址 美国纽约

[72]发明人 罗伯特·迈克尔·丁克杰安

[74]专利代理机构 中国国际贸易促进委员会专利商标事务所

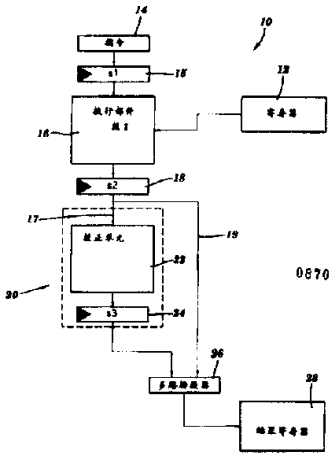
代理人 于 静

权利要求书 2 页 说明书 10 页 附图页数 8 页

[54]发明名称 面向算术/移位操作的单指令多数据校正电路

[57]摘要

公开一种处理算术/移位功能的指令的微处理器电路，这种算术/移位功能可以在操作的第一个时钟周期内执行一条指令的标准操作（例如，ALU 指令或移位指令），同时与算术/移位功能相关的校正电路用于把算术/移位功能提供的标准结果转换成执行一条 SIMD 指令所需要的结果形式。这种算术/移位功能是一条由算术逻辑单元（ALU）或是由一条移位指令所提供的指令。校正电路为逻辑指令不加改变地传送数据，但是根据 SIMD 指令提供条件代码。



权 利 要 求 书

1. 一个微处理器电路，包括:

a). 算术/移位功能，在操作的第一个时钟周期内执行指令的标准操作；及

b). 一个与所述的算术/移位功能相关的校正电路，用于把所述算术/移位功能所提供的标准结果转换成将要执行的单指令多数据指令所需要的结果。

2. 权利要求 1 的电路，其特征在于校正电路的转换在第二个时钟周期内执行。

3. 权利要求 2 的电路，其特征在于算术/移位功能可在第二个时钟周期内自由地执行第二条指令。

4. 权利要求 1 的电路，其特征在于算术/移位功能和校正电路能够在同一个时钟周期内各自执行一条指令。

5. 权利要求 1 的电路，其特征在于算术/移位功能是算术逻辑单元所提供的一条指令。

6. 权利要求 1 的电路，其特征在于校正电路为逻辑指令不改变地传送数据，但是根据 SIMD 指令提供条件代码。

7. 权利要求 5 的电路，其特征在于通过对基于有效位和子单元边界进位的标准数据进行操作，校正电路实现算术操作的校正。

8. 权利要求 5 的电路，其特征在于标准 ALU 包括加操作。

9. 权利要求 1 的电路，其特征在于算术/移位功能是一条移位指令。

10. 一个用于执行多媒体指令的微处理器包括:

a). 一个算术逻辑单元，在操作的第一时钟周期内执行指令的标准操作。

b). 一个与 ALU 相关的校正电路，用于把 ALU 提供的标准结果转换为单指令多数据所需要的结果。

11. 权利要求 1 的电路，其特征在于校正电路的转换在第二个时钟周期内执行。

12. 权利要求 2 的电路，其特征在于 ALU 在第二时钟周期内可自由地执行第二条指令。

13. 权利要求 1 的电路，其特征在于 ALU 和校正电路可以在同一时钟周期内各自执行一条指令。

14. 一个微处理器电路，包括:

a). 移位装置，在操作的第一个时钟周期内执行指令的标准操作;

b). 一个掩码生成电路，与移位装置电路并行工作;

c). 一个与所述的移位装置和所述的掩码生成电路相关的校正电路，用于把所述的移位装置提供的标准结果转换成执行 SIMD 移位指令所需要的结果。

15. 权利要求 13 中的电路，其特征在于在第二个时钟周期中实现转换。

16. 权利要求 14 中的电路，其特征在于由一个地址覆盖掩码电路执行转换。

17. 权利要求 14 中的电路，其特征在于移位装置在第二个时钟周期可自由地执行第二条指令。

18. 权利要求 13 中的电路，其特征在于移位装置和校正电路可以在同一时钟周期内各自完成一条指令。

19. 权利要求 13 中的电路，其特征在于移位装置执行移位操作。

说　　明　　书

面向算术/移位操作的单指令多数据校正电路

本发明涉及微处理器中电路的数据寄存器，尤其涉及用于修正算术/移位操作结果的单指令多数据(SIMD)校正电路。

在此之前，逻辑电路被用于数据处理中提高算术/移位操作的性能。随着对高速处理大量数据的这种需要的不断增长，提高算术/移位操作的效率变得非常重要。特别是多媒体技术遇到的困难之一，尤其涉及到图形，则必须要处理大量的数据。单指令多数据(SIMD)的一个特性就是每个SIMD指令对一个64位的操作数可以独立的对每8位，16位，32位或64位字段进行操作。

例如，SIMD的加法指令，能够在寄存器操作数的第一个，第二个，第三和第四个16位字段上进行加运算，好象SIMD加法是4个独立的16位加法指令。SIMD的移位指令，举例说明，能够在寄存器操作数的第一个，第二个，第三和第四个16位字段上进行移位操作，好象SIMD移位指令是4个独立的16位移位指令。另外，移位操作包括左移，逻辑右移，算术右移。

随着Intel MMX扩展的发布，SIMD获得了广泛地承认。MMX是SIMD体系结构。在X86体系结构上实现MMX扩展需要附加的执行部件以符合MMX格式。转换标准的执行部件使其能够执行标准的和SIMD的操作将会遇到前所未遇的困难。首先，附加的执行部件增加了关键线路例如进位传输线路的延迟，因为在SIMD中，两个SIMD的子操作数（16位或32位字段）之间的进位是必须被删除的。第二，附加的执行部件需要占用附加的硅片区域。第三，由于该执行部件是高度专用的电路，所以附加的执行部件需要增加开发的时间和开销。

本发明包括了一种校正电路，可以把标准逻辑的执行部件转变为可执行标准操作和SIMD操作。本发明具有如下的优点，在关键路径例如进位传输线路的延迟上缓解了难度；与增加附加的执行部件相比，本发明需要更少的硅空间；同时不需要进一步开发高度专用的执行部件，若执行部件是

高度专用的，因而对其进行修改将要耗费大量的劳动。本发明增强了在应用软件例如多媒体和信号处理中处理大量数据的能力。本发明排除了关键的线路，因为逻辑并不增加到关键路径中；因为只是重复使用相同的执行部件，所以节省了时间和硅区域；而且不需要重新构造具有附加逻辑的复杂部件。另一种在多媒体和信号处理中需要处理大量数据的操作是矩阵乘法。本发明可以应用于算术逻辑部件和移位逻辑中的算术/移位操作(例如，加，减，乘，除，移位)的标准逻辑。

本发明包括了一个带有执行部件的微处理器电路，用于执行算术/逻辑操作中的标准指令，以及一个与执行部件相关的校正电路，用于把执行部件提供的标准指令修正为正在执行的SIMD指令要求的结果。本发明提高了操作的效率，因为校正电路的修正可以在第二个时钟周期内执行，同时算术/移位操作又可以在第二个时钟周期内自由地执行第二条指令。算术/移位操作的结果可以通过执行算术逻辑部件或移位功能提供的指令而得到。校正电路对标准逻辑指令不加改变地传送数据，但是要根据SIMD指令提供条件代码。校正电路通过对标准数据进行基于有效位和子单元边界进位的处理修正算术操作。

在移位操作中，本发明包括一个移位装置，用于在操作的第一个时钟周期中执行指令的标准操作，一个掩码生成电路与移位装置电路并行执行，以及一个与移位装置和掩码生成电路相关的校正电路，用于把移位装置提供的标准结果修正为一个正在执行的SIMD移位操作所要求的结果。这种修正可以在第二时钟周期内通过地址覆盖掩码来实现，同时移位装置在第二时钟周期内可以自由地执行第二条指令。

本发明的示范性的最佳实施方式将在下文中结合附图加以说明，其中相同的符号代表相同的组成部分。

图.1 中所示的框图是一个可执行算术/移位操作的执行部件和根据本发明提出的第一种最佳实施方式的校正电路。

图.2 中所示的框图是一个可执行算术/移位操作的执行部件和根据本发明提出的第二种最佳实施方式的校正电路。

图.3 所示的是根据本发明的最佳实施方式，一个 SIMD 操作的高层视图。

图.4 所示的是根据本发明的一种最佳实施方式，公开校正单元的实施细节流程图。

图.5 所示的是根据本发明的一种最佳实施方式，移位功能的流程图。

图.6 所示的是图.5 所示的根据本发明的掩码生成装置的细节。

图.7 所示的是图.5 中所示的根据本发明的与/或掩码(AOM)细节。

图.8 所示的是根据本发明的第三种最佳实施方式的执行部件/校正单元和框图。

参考图 1，显示了本发明的微处理器电路 10 与一个校正电路 20。校正电路 20 把或者是 ALU(图 4)的标准操作或者是移位装置(图 5 - 7)的标准操作的结果转变为一条将要执行的 SIMD 指令所需要的结果。标准执行部件 16 从保留站(s1)15 接收一条将要执行的指令，而保留站用于保存该将要执行的指令 14。执行部件 16 提供了访问寄存器 12 的途径，并可以执行 ALU 操作或者移位操作。结束级(s2)18 中保存写回级中的指令 14 执行完毕时将要被写回的结果。

一个校正电路 20，如图所示包括用于 SIMD 应用的校正单元 22 和结束级（ S3 ） 24。在图 1 所示的第一种实施方式中，执行部件被描述为两级流水线。流水线的第一级 17 进入校正电路 20，流水线的第二级 19 绕过校正电路 20 而到达多路转换器 MUX 26 用于非 SIMD 操作。寄存器 28 中保留的二级流水线 17,19 的结果被送至寄存器 12。

图 2 描述了本发明的第二种实施方式的校正电路 30。除了去掉了 MUX 26 部分，这种实施方式与图 1 非常相似。允许在同一个时钟周期内象执行非 SIMD 操作一样执行 SIMD 或校正操作。 SIMD 校正指令通过流程线 29 传送，非 SIMD 指令则通过流程线 31 绕过校正单元 22。

然而在另外一种实现中，图 8 所示的是一种组合的执行部件/校正单元 100。通过将 ALU 第一条通道的或者移位算子级的结果 17， 101 反馈到 ALU 或者增加了适当的控制硬件的移位算子， ALU 或者执行部件/校正单元 100 的移位算子可以实现校正级。对于已经熟悉了本发明的熟练的技术人员，这些实现应当是很明显的，因而它包括在该公开所要求的权利范围之内。

使用标准 ALU 实现 SIMD

图 3 显示了一个 SIMD 操作 40 的高层视图。图中显示了一条指令 14。

一条简单SIMD指令44对寄存器12的分别为7,6,5,4,3,2,1,0的子集同时进行操作。如图所示，寄存器操作数(R2)44指向一个64位的寄存器12。操作数的指令OP具体指明了操作，例如，R1<-R1+R2加法指令。指令14将在从0到7的子集上独立的执行加操作。从0到7的每个子集可以是一个单独的字节，加指令可以使用64位寄存器操作数中8个字节的每一个即8个独立的加数进行加运算，子集可以是16位或32位。

参考图4，展示了图1和图2中校正电路20的具体内容，该图中使用了一个标准ALU。一个64位执行部件16有一个8字节的结果寄存器24。B7是高位字节，B(6),...B(0)是低位字节。每个字节都有三个附加的锁存C(n),CI(n),和ZB(n),C(n)是B(n)的最高位位的进位(本字节的进位)，CI(n)是向B(n)最高位的进位，ZB(n)指示B(n)的所有位都是0。

这些B(n)寄存器中的每一个都与一个标记为F(n)的强制置码框(foree box)相连。F(n)有两条输入控制线加上来自B(n)的数据。输入控制线是强制8(n)和0(n)。强制8(n)在该框的输出FB(n)强加16进制数‘80’，强制0(n)强加16进制数‘00’。如果两条控制线都关闭，输入总线就不加改变的传送到输出总线。FB(n)的输出字节传送到增量/减量框ID(n)。该框有两条控制输入IB(n)和DB(n)。IB(n)在输入总线上加1，DB(n)在输出总线上减1。两条控制线关闭时将输入总线不加改变地传送到输出总线。于是ID(n)有一个字节的输出总线和一条输出控制线CB(n)。该输出是进位/借位线而不是字节加1/减1线。即当DB(n)是有效的并且FB(n)=X’00’，或者IB(n)是有效的并且FB(n)=X’FF’，它应该是有效的。

下面将解释将要执行的操作的例子。本发明予期会包括各种可能性，而在当前的SIMD定义下组合所有的功能是不可能的。

校正级将被提供大量的指令参数。第一，字节(B)，字(W)，双字(DW)和四字(QW)数据尺寸(一个字是16位)将被提供。第二，提供加或者减指令。第三，指令可以是有符号和无符号格式。有符号数是标准的2的补码形式，而无符号数仅是正数。第四，指令可以在饱和或者非饱和状态下执行。非饱和是指如果结果超过了规定的大小，结果将发生绕回(wrap)。饱和是指如果结果超过了指定的大小，那么给定格式的最大或者最小的数将被插入。例如，一个无符号字节发生上溢出将会产生结果255或者16进制数

‘FF’，而下溢出(仅发生在减法运算中)将产生0或者16进制数‘00’。对于有符号字节，发生上溢出将会产生结果127或者16进制数‘7F’，下溢出(可能出现在加法或者减法中)将会产生结果－128或者16进制数‘80’。图1－4中定义的结构允许实现校正上述所有指令参数方案的组合。注意当强制置码框直接产生16进制数‘00’和‘80’时，可以通过在16进制数‘00’和‘80’上减1产生16进制常数‘FF’和‘7F’。

最简单的方案，无符号字节加法运算(非饱和)中，低位字节B(0)不需要校正。如果C(0)=1,下一个字节B(1)需要减1。于是DB(1) = C(0)。通常如果C(n-1) = 1或者DB(n) = C(n-1),B(n)应该减1。

在无符号字加法运算(非饱和)方案中，结果B(1)/B(0)不需要校正。如果C(1)=1，下一个字B(3)/B(2)需要减1。这意味着DB(2)=C(1)并且DB(3)=CB(2)。一般情况下，DB(n)=C(n-1)并且DB(n)=C(n-1)并且DB(n+1)=CB(n)，其中n=2，4，6。

无符号双字加法运算采用相似的模式，而无符号四字加法不需要校正。显然对于有符号加法(非饱和)以上模式是相同的。

在饱和状态下的无符号字节加法中只可能发生上溢出，因此仅需要强制设置16进制数‘FF’。低位字节不需要减1，但是如果C(0)=1应当强制设置16进制数‘FF’。这可以通过设置F0(0)=1和DB(0)=1来实现。于是F0(0)=C(0)以及DB(0)=C(0)。如果C(0)=1下一个字节B(1)需要减1，如果C(1)=1将被强制设置成16进制数‘FF’。于是F0(1)=C(1)并且DB(1)=c(0) OR C(1)。注意如果由于C(0)=1导致C(1)=1，强制设置成16进制数‘FF’是可以接受的，因为忽略C(0)的结果应当是16进制数‘FF’。所以通常F0(n)=C(n)并且DB(n)=C(n) OR C(n-1)。

至于饱和状态下的无符号字加法运算，低位字B(1)/B(0)不需要减1，但如果C(1)=1则两个字节需要被强制设置成16进制数‘FF’。所以DB(1) = DB(0)=F0(0)=F0(1)=C(1)。如果C(1)=1下一个字B(3)/B(2)需要减1，如果C(3)=1需要将其强制设置成16进制数‘FFFF’。对于B(2)，DB(2)=C(3) OR C(1)，并且F0(2)=C(3)。对于B(3),F0(3)=C(3)并且DB(3)=C(3) OR C(2)。通常F0(n+1)=F0(n)=C(n+1)，DB(n)=C(n+1) OR C(n-1),并且DB(n+1)=C(n+1) OR CB(n),其中n=2，4，6。

饱和状态下无符号双字/四字加法运算采用一种相似的模式。

在无符号加法中 C(n)代表了上溢出，其中不可能出现下溢出。在有符号加法中上溢出和下溢出都可能出现，所以比较复杂。让 OV(n)=C(n) NOT AND CI(n)代表上溢出， UV(n)=C(n) AND NOT CI(n)代表了下溢出，并且 V(n)=UV(n) OR V(n)代表某些上溢出/下溢出条件。

饱和状态下有符号字节加法要求强制设置 16 进制数 ‘ 7F ’ 或者 ‘ 80 ’ 。低位字节 B(0)不需要减 1 ， 但如果 OV(0)=1, 则 B (0) 需要设置成 16 进制数 ‘ 7F ’, 而如果 UV(0)=1, B(0)需要设置成 16 进制数 ‘ 80 ’, 所以如果 OV(1)=1 应当是 16 进制数 ‘ 7F ’ 。于是 F8(1)=V(1)并且 DB(1)=OV(1) OR C(0)。通常 F8(n)=V(n)并且 DB(n)=OV(n) OR C(n-1)。

饱和状态下有符号字加法增加了复杂性。当检测下溢出和上溢出必须分别把字设置成 16 进制数 ‘ 8000 ’ 和 ‘ 7FFF ’ 。于是低位字节必须被设置成 16 进制数 ‘ 00 ’ 和 ‘ FF ’ ， 而高位字节要设置成 16 进制数 ‘ 80 ’ 和 ‘ 7F ’ 。低位字 B(1)/B(0)不需要减 1 ， 但如果 OV(1)=1, 则 B(1)/B(0)要设置成 16 进制数 ‘ 7FFF ’ ， 并且如果 UV(1)=1 那么 B(1)/B(0)要设置成 16 进制数 ‘ 8000 ’ 。因此 F8(1)=F0(0)=V(1)， DB(0)=DB(1)=OV(1)。对于下一个字 B3/B2 ， 如果 C(1)=1 那么要减 1 ， 如果 OV(3)=1 要设置成 16 进制数 ‘ 7FFF ’ ， 如果 UV(3)=1 要设置成 16 进制数 ‘ 8000 ’ 。因此 F(3)=F0(2)=V(3)， DB(2)=OV(3) OR C(1)以及 DB(3)=OV(3) OR DB(2)。通常 F8(n-1)=F0(n)=V(n+1) ， DB(n)=OV(n-1) OR C(n) ， 以及 DB(n+1)=OV(n+1) OR DB(n)， 其中 n = 2 ， 4 ， 6 。

饱和模式下有符号双字/四字加法采用一种相似的处理模式。减法处理方案与加法相同，除了当进位是 1 时，校正应当减 1 ，进位是 0(借位是 1)时应当加 1 。下溢出和上溢出定义相同。

比较相等和比较大于需要取两个有符号的字节，字，双字或者四字的操作数进行比较。如果是真 (true) 结果域将全部设置成全 1 ，如果是假 (false)结果域将全部设置成全 0 。提供了 ZB(n)， C(n)和 CI(n)信号后，比较字节，字，双字和四字的相等或大于将变得很容易。已经定义的结构中允许设置所有的位为全 0 或全 1 。

参考图 4 ，进位逻辑可以按照本技术中已知的很多方式来实现。图中

采用了一种行波进位(ripple)方案，但是在本发明涉及的领域内可以采用(例如)进位预测和先行进位技术。

用标准移位装置实现SIMD移位指令

如图1所示，一个保留站(s1)15存放将被执行的指令，执行部件16包括了一个移位装置（图5），可访问寄存器12，当在回写级指令运行完毕后，结束级（S2）18保存将被写回的结果。图2所示的是本发明的当前实施方式，其中执行部件16是一个两级流水线并且非SIMD指令将绕过SIMD校正电路。本实施方式在前文中已经详细阐述。SIMD级取得移位装置操作的结果，并且校正子单元以适应SIMD操作。

在本发明的另外一种实施方式中MUX26被删除，允许SIMD指令象执行单时钟周期指令一样在同一个时钟周期内完成。

在本发明的另外一种实施方式中，通过把移位装置级的第一条通道的结果反馈到增加了控制硬件的移位装置，移位装置级可实现校正级20。当了解了本发明后对于那些熟练的技术人员所有这些实现将变得很明显因而属于本发明范围之内。

图3所示的是SIMD操作的概念，一条SIMD指令同时在寄存器操作数的子集上进行操作。寄存器(R2)44指向一个64位的操作数12。OP代码指出了具体的操作(例如R1<-R2移位指令)。SIMD可独立地在每个子集上进行移位操作。子集可以是一个字节，于是使用64位寄存器操作数的8个字节可以执行8个独立的移位操作，或者子集也可以是16位和32位。

图5所示的是本发明最佳实施方式的移位装置60的高层视图。移位计数(SCNT)62，操作数64和结果寄存器82是非SIMD指令的标准移位功能的几个组成部分。掩码生成装置76使用操作数64和移位计数62能使掩码寄存器与标准移位结果并行生成与-或掩码(AOM)84。掩码生成装置，MR，和AOM是图1和图2中所示的校正电路。AOM84的结果被锁存到最终结果(第二级)寄存器。

图6所示是掩码生成装置的详细内容。移位计数寄存器(SCNT)62连冈左移或右移指示(L/R)66共冈产生移位计数掩码(SCNTM)62。字节移位掩码(BSM)78为64位掩码中的每一个字节产生移位掩码。如果字节左移移位的数目不超过8，该掩码将是正确的。将数据与掩码进行“与”运算将会得

到左移的正确结果。表 2 描述了当移位计数等于或大于某一数字时所推导出的等式。例如，当 SCNT 大于或等于 8 时 SCT8 就是此等式。在表 3 中这些等式被用于定义在左移中强制设置 16 进制数 ‘00’ (F0Sn) 的功能，在右移中强制设置 16 进制数 ‘FF’ (F1Sn) 的功能。

至此，按照图 7 所定义的功能代码把 MR 中的掩码与结果寄存器中 64 位移位数据进行 AND/OR 操作，正确的结果将被产生。当数据要被保留左移产生的掩码有一些 1，而数据要被清零时，掩码有一些 0。在右移中 0 指出数据应当被保留，1 则指出了根据移位种类的不同(算术的或逻辑的) 0 或 1 应当被添入，以及高位位的处理。

图 4 所示的是将被送入 AOM 的功能字段的产生。参考图 7，AOM 接收 MR 和结果寄存器中的每个字节，以及执行表 5 中指出的基于 FN(X) 字段的功能。

利用具体的实施方式本发明被详细地阐述。它应当被理解，但是这里的实施方式仅仅是说明，而且本发明不必要仅限于此。在其思想和权利要求的领域内接着对本发现进行完善和丰富将是很明显的，而且那些熟练的技术人员将会赞同这一点。

表 1． 字节移位掩码（ BSM ）功能

移位计数器掩码	字节移位－掩码
2 1 0	7 6 5 4 3 2 1 0
0 0 0	1 1 1 1 1 1 1 1
0 0 1	1 1 1 1 1 1 1 0
0 1 0	1 1 1 1 1 1 0 0
1 0 0	1 1 1 1 0 0 0 0
1 0 1	1 1 1 0 0 0 0 0
1 1 0	1 1 0 0 0 0 0 0
1 1 1	1 0 0 0 0 0 0 0

表 2．移位计数等式

SCT8 = S3+S4+S5+S6
SCT16 = S4+S5+S6
SCT24 = S5+S6+(S3 A S4)
SCT32 = S5+S6
SCT40 = S6+S5 A (S3 + S4)
SCT48 = S6+ (S5 A S4)
SCT56 = S5 A S4 A S3 .
SCT64 = S6

表 3. 左移和右移

		FOS7	FOS6	FOS5	FOS4	FOS3	FOS2	FOS1	FOS0
	右	F1S7	F1S5	F1S5	F1S4	F1S3	F1S2	F1S1	F1S1
字节	左	0	0	0	0	0	0	0	0
	右	0	0	0	0	0	0	0	0
字	左	SCT16	SCT8	SCT16	SCT8	SCT16	SCT8	SCT16	SCT8
	右	SCT8	SCT16	SCT8	SCT16	SCT8	SCT16	SCT8	SCT16
双字	左	SCT32	SCT24	SCT16	SCT8	SCT32	SCT24	SCT16	SCT8
	右	SCT8	SCT16	SCT24	SCT32	SCT8	SCT16	SCT24	SCT32
四字	左	SCT64	SCT56	SCT48	SCT40	SCT32	SCT24	SCT16	SCT8
	右	SCT8	SCT16	SCT24	SCT32	SCT40	SCT48	SCT56	SCT64

表 5.ADM 功能

FN (1:0)	功能	表述
0 1	R AND MR	左移
1 0	R AND MR_{not}	右移 W/ 1
1 1	R OR MR	右移 W/ O

表 4 .FN（ X ）的生成

操作	FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0
SL (B,W,DW,QW)	01	01	01	01	01	01	01	01
SRL (B,W,DW,QW)	10	10	10	10	10	10	10	10
SRA(B)*	R63	R55	R47	R39	R31	R23	R15	R7
SRA(W)*	R63	R63	R47	R47	R31	R31	R15	R15
SRA(DW)*	R63	R63	R63	R63	R31	R31	R31	R31
SRA(QW)*	R63	R63	R63	R63	R63	R63	R63	R63

SL=（左移）
SRL=（逻辑右移）
SRA=（算术右移）
B- 字节
W=2 字节
DW=4 字节
QW=8 字节

* 算术右移（ SRA ）， FN 的两位代码是“ 1 ” R （ y ），其中 y 是一个 2 进制位的位量 （即， R15 是位 15 ）

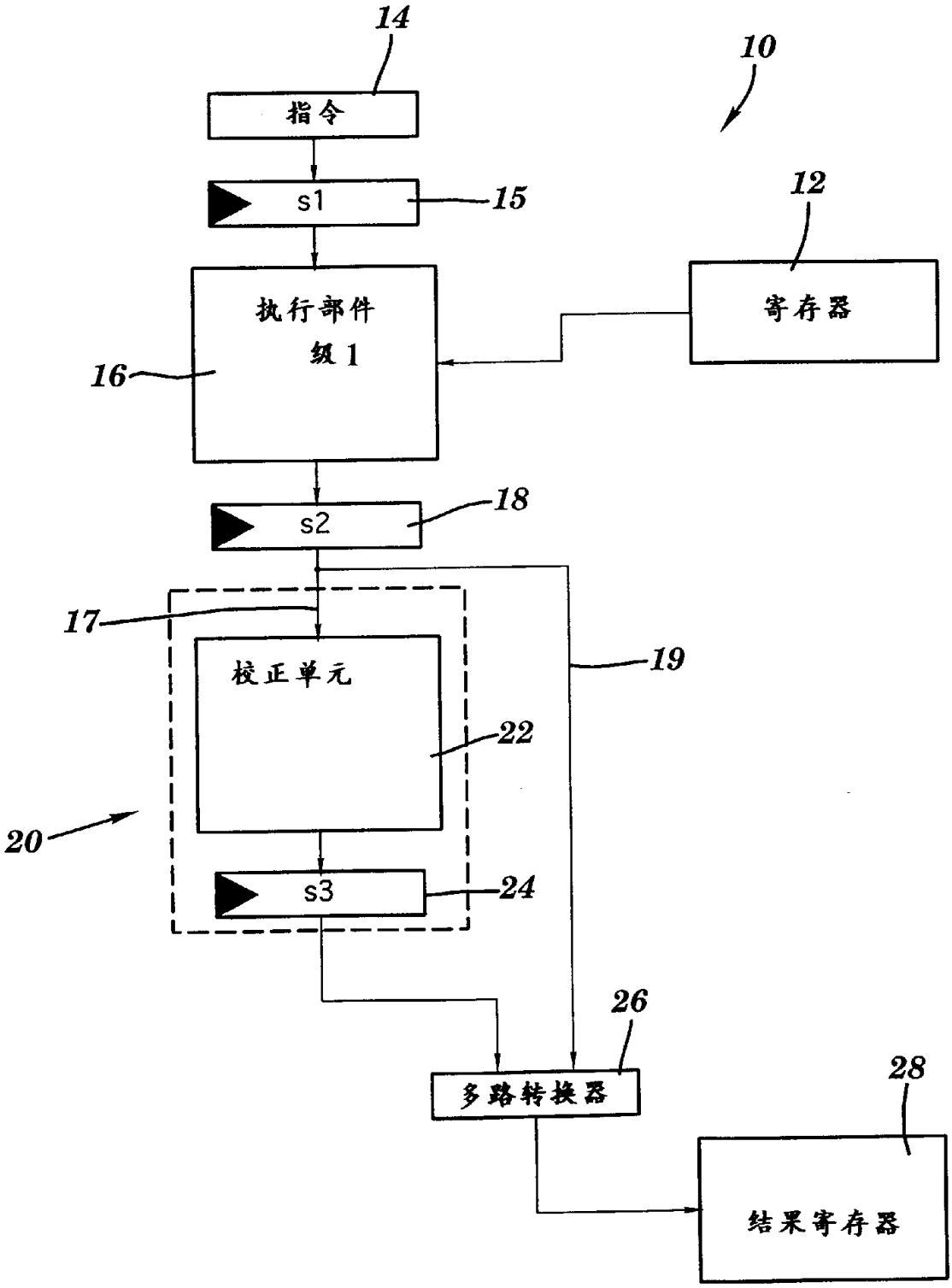

14
10
指令
s1
15
12
执行部件
级 1
寄存器
16
18
s2
17
19
校正单元
22
20
s3
24
26
28
多路转换器
结果寄存器

图 1

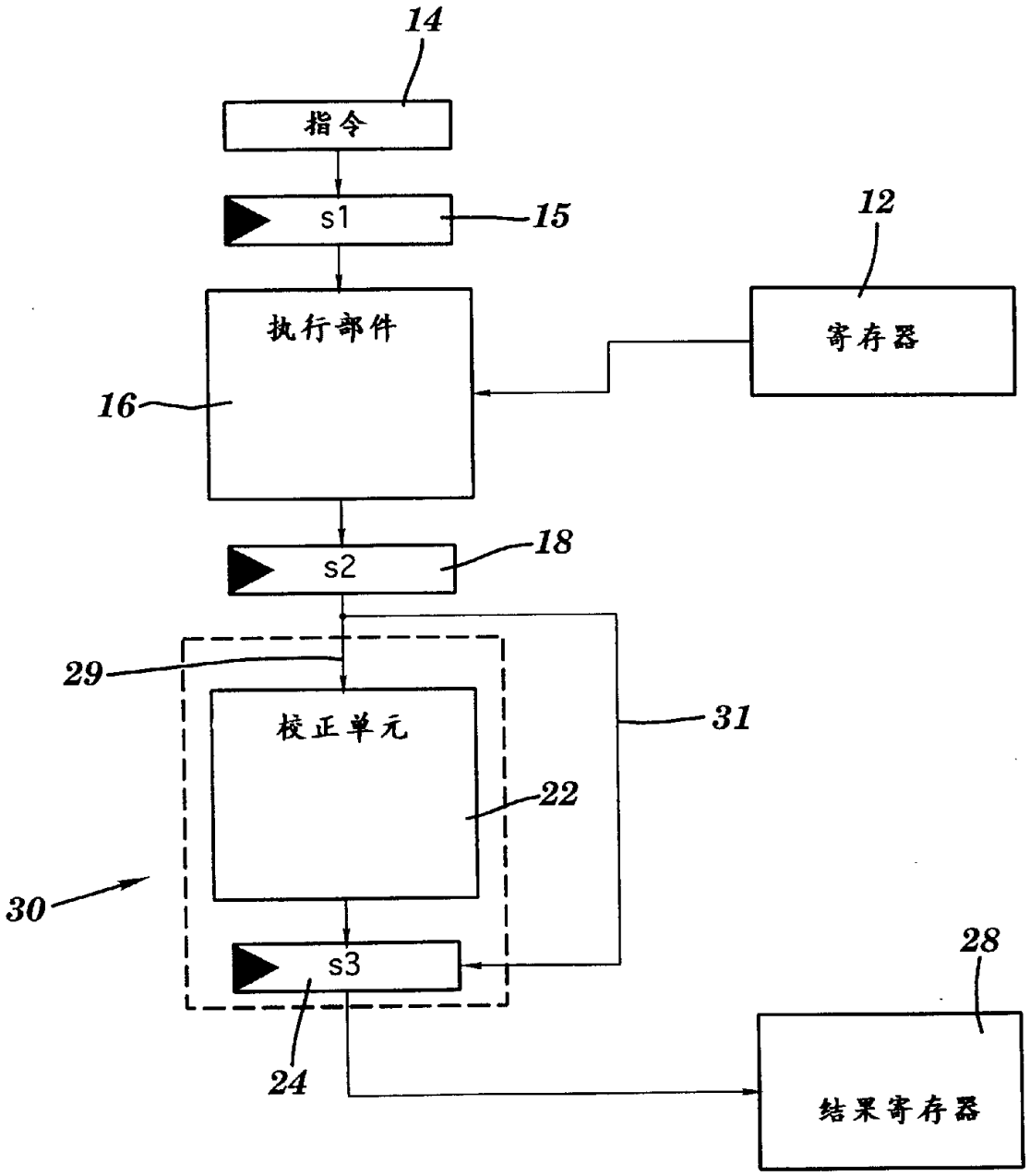


图 2

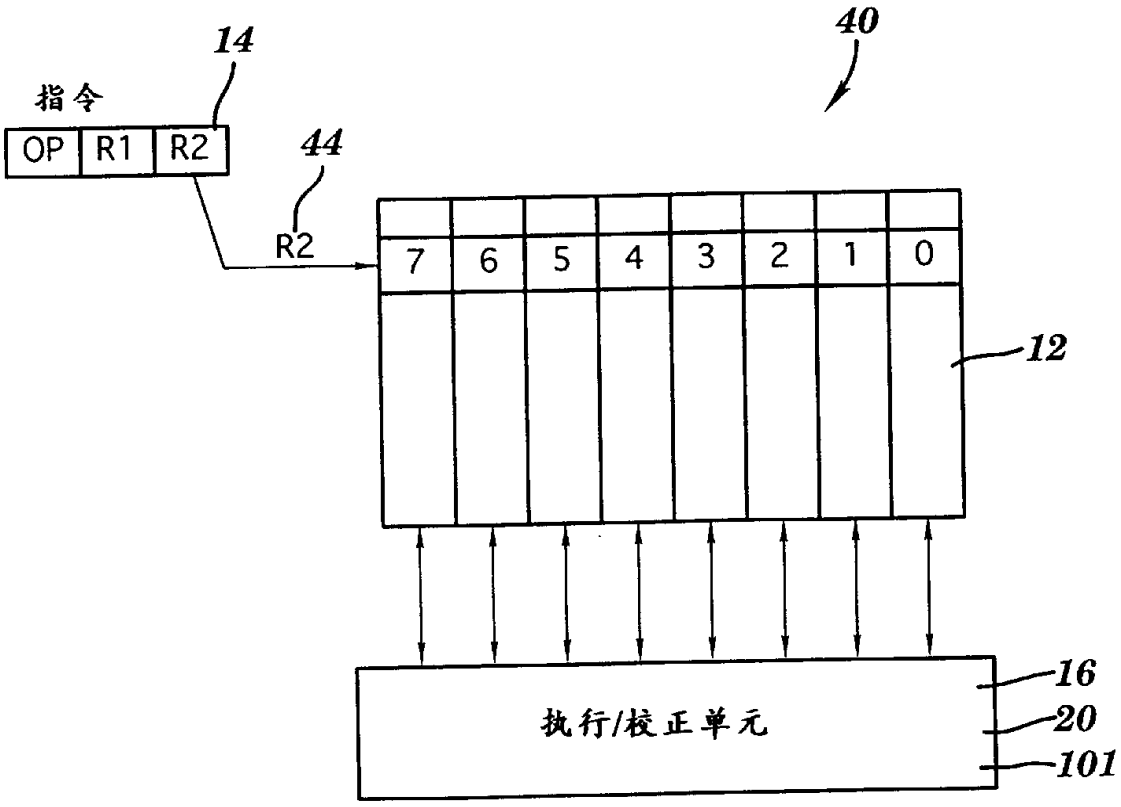


图 3

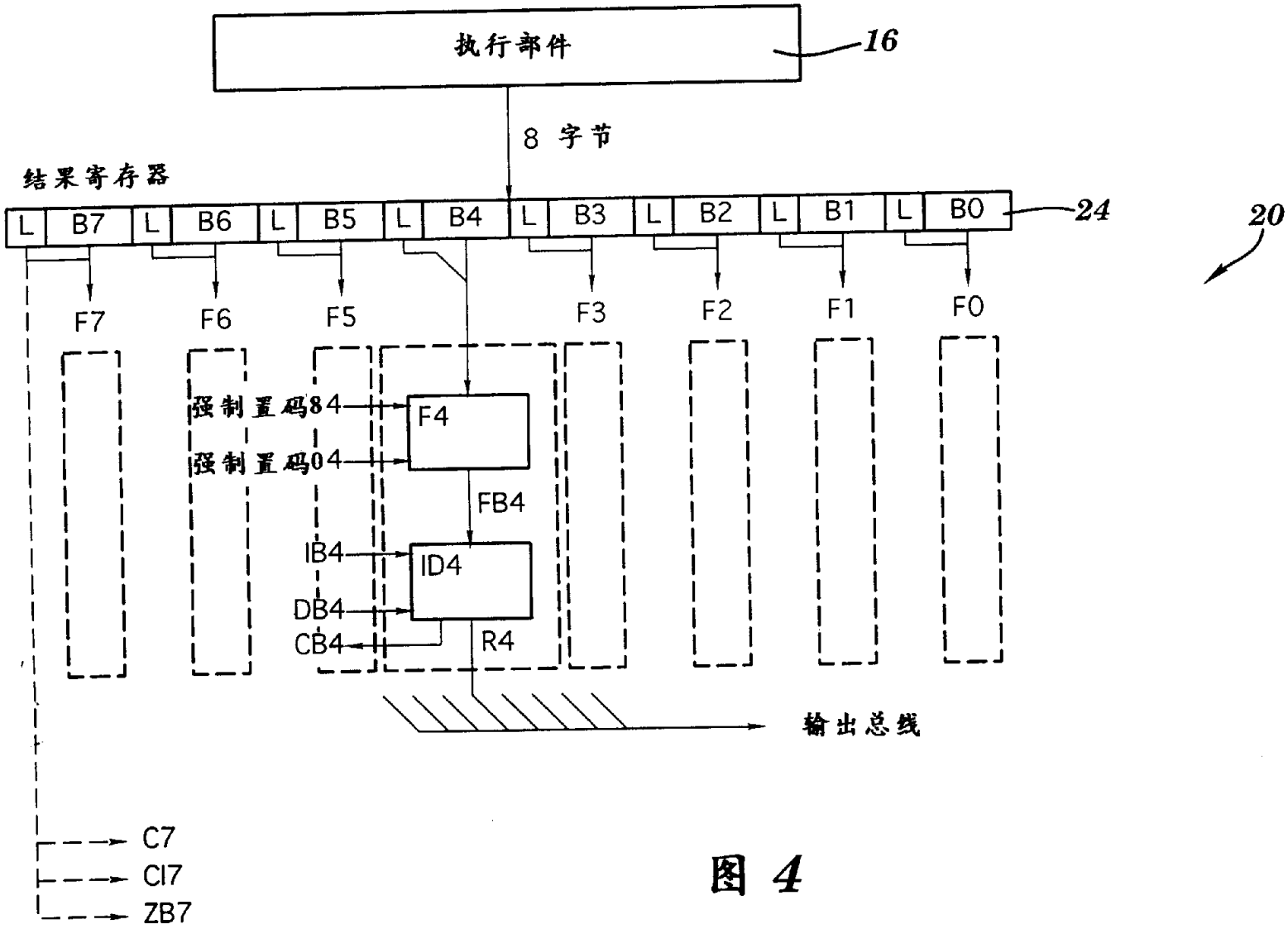

执行部件
16
8 字节
结果寄存器
L B7 L B6 L B5 L B4 L B3 L B2 L B1 L B0
24
20
F7
F6
F5
F3
F2
F1
F0
强制置码84
强制置码04
F4
FB4
IB4
ID4
DB4
CB4
R4
输出总线
C7
CI7
ZB7

图 4

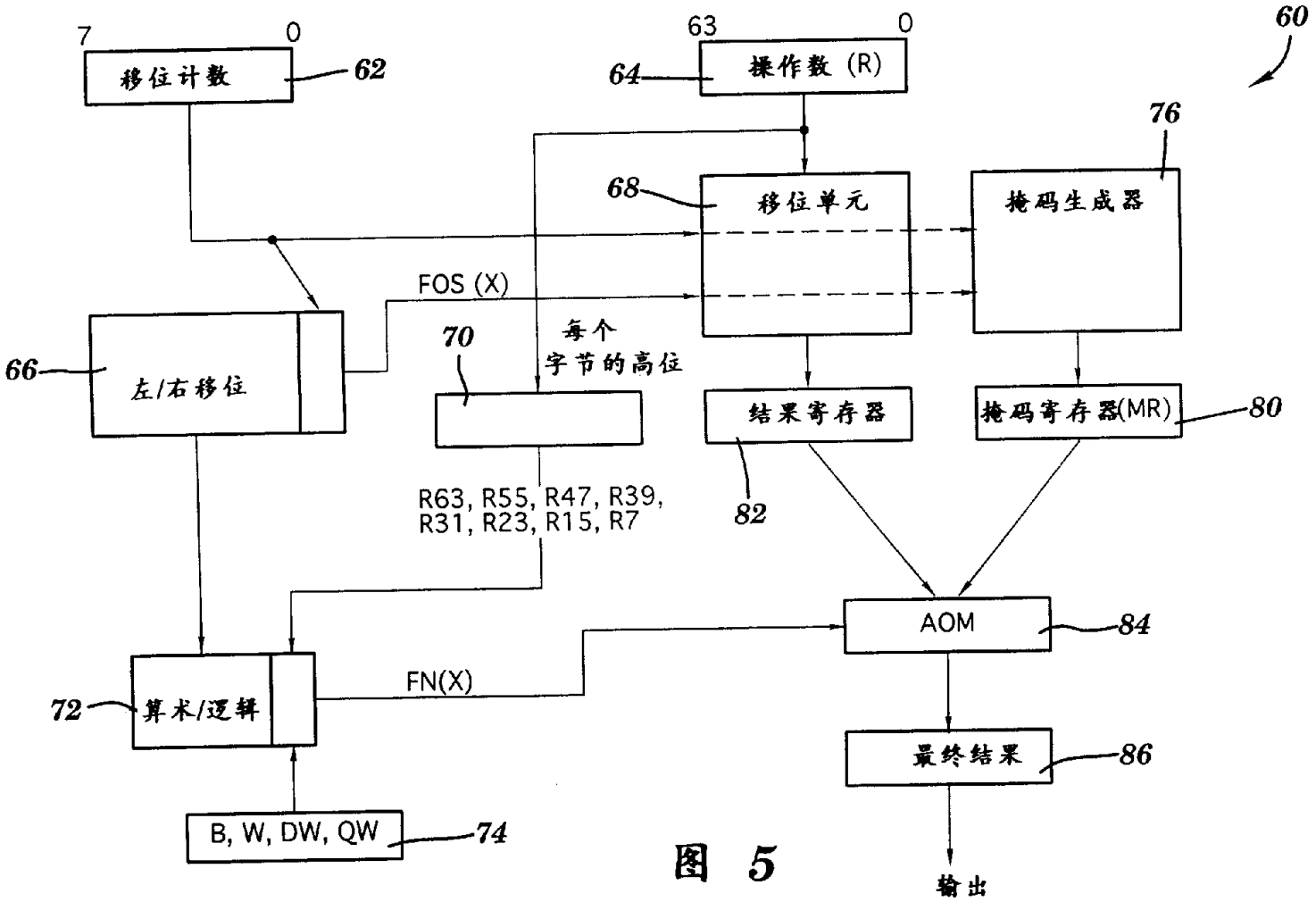
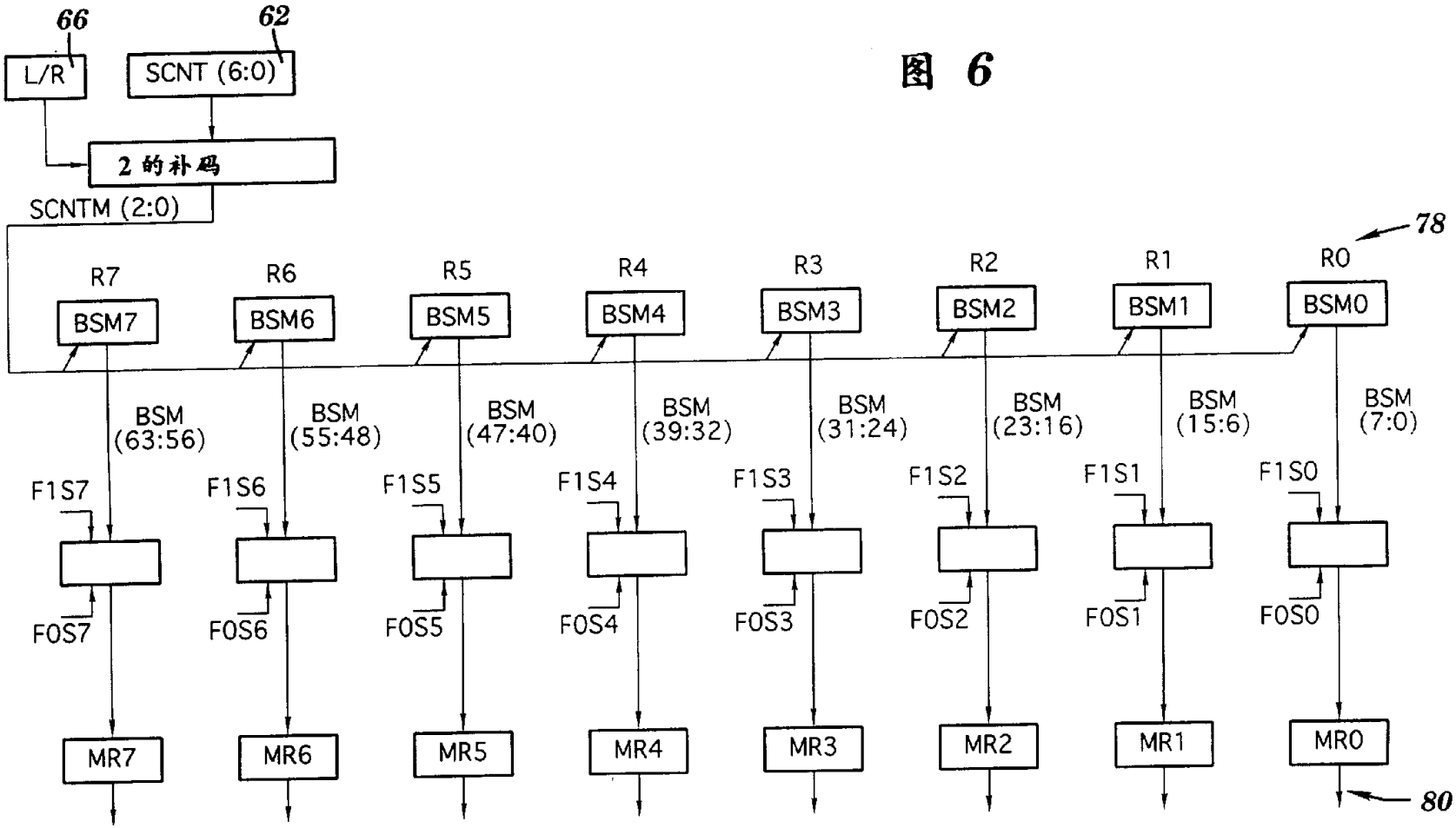

7
0
移位计数
62
63
0
64
操作数 (R)
60
76
68
移位单元
掩码生成器
FOS (X)
每个
字节的高位
70
66
左/右移位
结果寄存器
掩码寄存器(MR)
80
R63, R55, R47, R39,
R31, R23, R15, R7
82
AOM
84
FN(X)
72
算术/逻辑
最终结果
86
B, W, DW, QW
74
输出

图 5

图 6


66
62
L/R
SCNT (6:0)
2 的补码
SCNTM (2:0)
78
R7
R6
R5
R4
R3
R2
R1
R0
BSM7
BSM6
BSM5
BSM4
BSM3
BSM2
BSM1
BSM0
BSM (63:56)
BSM (55:48)
BSM (47:40)
BSM (39:32)
BSM (31:24)
BSM (23:16)
BSM (15:6)
BSM (7:0)
F1S7
F1S6
F1S5
F1S4
F1S3
F1S2
F1S1
F1S0
F0S7
F0S6
F0S5
F0S4
F0S3
F0S2
F0S1
F0S0
MR7
MR6
MR5
MR4
MR3
MR2
MR1
MR0
80

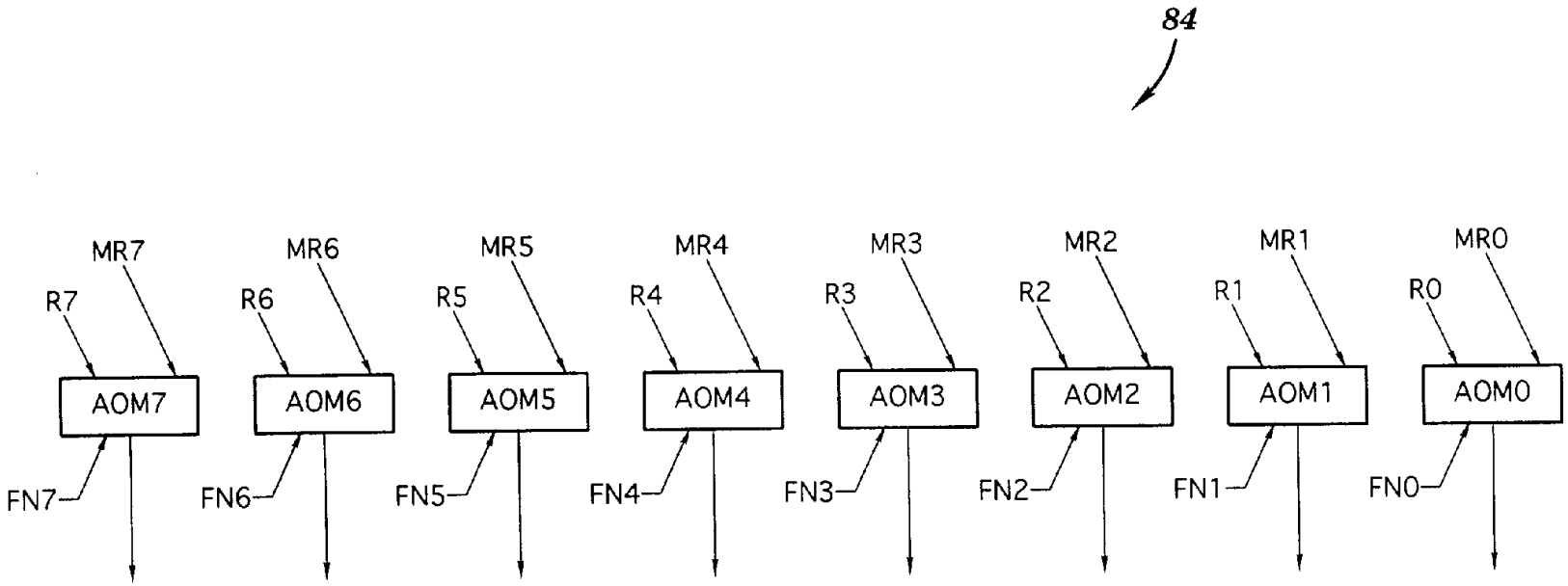

84
MR7
R7
AOM7
FN7
MR6
R6
AOM6
FN6
MR5
R5
AOM5
FN5
MR4
R4
AOM4
FN4
MR3
R3
AOM3
FN3
MR2
R2
AOM2
FN2
MR1
R1
AOM1
FN1
MR0
R0
AOM0
FN0

图 7

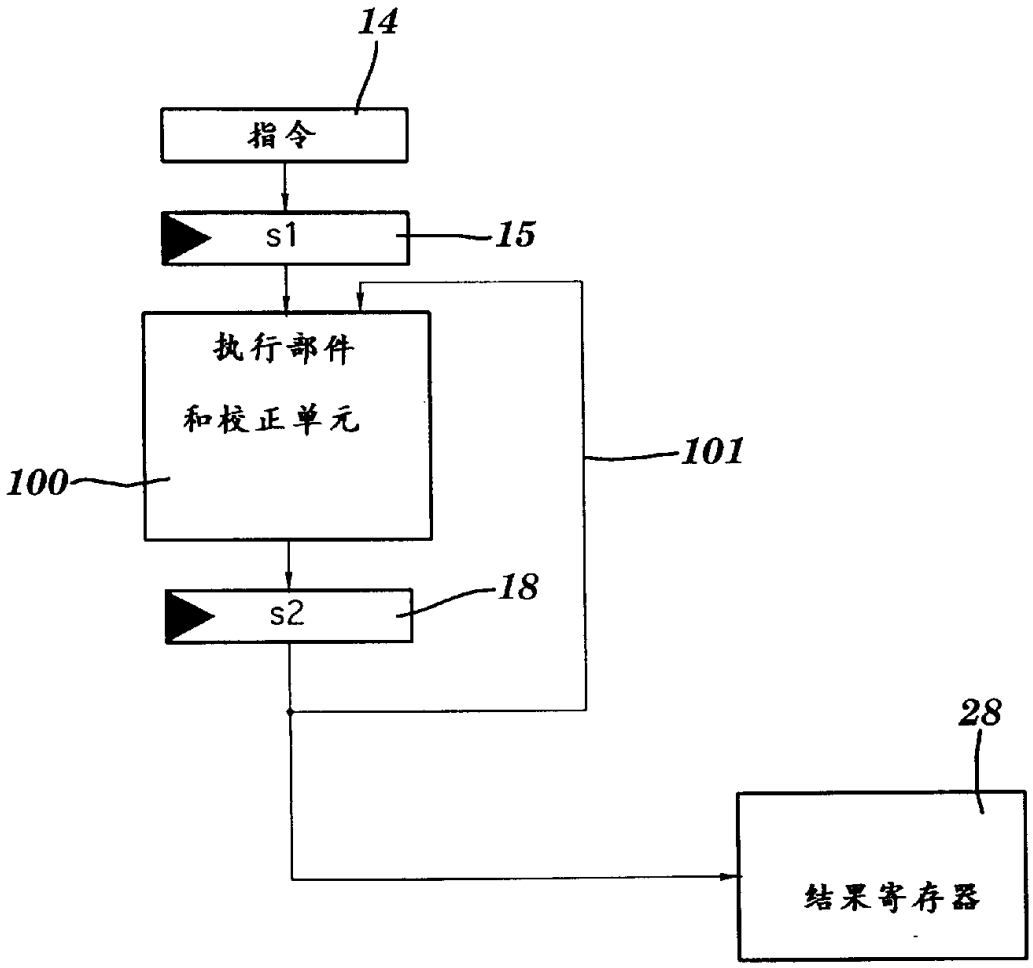


图 8