(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105051743 B
(45)授权公告日 2018.06.08

(21)申请号 201380062069.0

(22)申请日 2013.06.18

(65)同一申请的已公布的文献号
申请公布号 CN 105051743 A

(43)申请公布日 2015.11.11

(30)优先权数据
13/729,502 2012.12.28 US

(85)PCT国际申请进入国家阶段日
2015.05.28

(86)PCT国际申请的申请数据
PCT/US2013/046410 2013.06.18

(87)PCT国际申请的公布数据
WO2014/105135 EN 2014.07.03

(73)专利权人 英特尔公司
地址 美国加利福尼亚

(72)发明人 G·M·沃尔里克 K·S·叶
V·戈帕尔 J·D·吉尔福德

(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 邬少俊 王英

(51)Int.Cl.
G06F 21/72(2013.01)
G06F 21/64(2013.01)
G06F 21/60(2013.01)
G06F 9/30(2006.01)
H04L 9/06(2006.01)

(56)对比文件
CN 101350716 A,2009.01.21,全文.
CN 1684412 A,2005.10.19,全文.
US 2010250966 A1,2010.09.30,全文.
US 2012128149 A1,2012.05.24,全文.
US 2012257742 A1,2012.10.11,全文.

审查员 赵洋

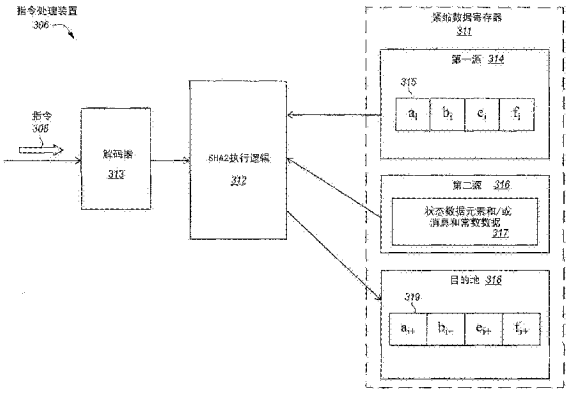
权利要求书3页 说明书28页 附图28页

(54)发明名称

用于处理安全哈希算法的指令处理器、方法、和系统

(57)摘要

一种方法的一方面包括接收指令。所述指令指示包括了针对安全哈希算法2(SHA2)哈希算法的当前轮(i)的状态数据元素a_i、b_i、e_i和f_i的第一紧缩数据的第一源。所述指令指示第二紧缩数据的第二源。所述第一紧缩数据具有的比特宽度小于SHA2哈希算法的八个状态数据元素a_i、b_i、c_i、d_i、e_i、f_i、g_i、h_i的组合的比特宽度。所述方法还包括响应于所述指令，在由所述指令指示的目的地中存储结果。所述结果包括已经通过至少一轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i、和f_i而更新的已更新的状态数据元素a_{i+}、b_{i+}、e_{i+}、和f_{i+}。



CN 105051743 B

1.一种处理器,包括:

多个紧缩数据寄存器;以及

执行单元,其与所述多个紧缩数据寄存器耦合,所述执行单元响应于指令而可操作,所述指令指示包括了针对安全哈希算法2SHA2哈希算法的当前轮i的状态数据元素a_i、b_i、e_i和f_i的第一紧缩数据的第一源,并且所述指令还指示第二紧缩数据的第二源,其中,所述第一紧缩数据具有的比特宽度小于所述SHA2哈希算法的八个状态数据元素a_i、b_i、c_i、d_i、e_i、f_i、g_i和h_i的组合的比特宽度,所述执行单元可操作以在由所述指令指示的目的地中存储结果,所述结果包括通过至少一轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+}、b_{i+}、e_{i+}和f_{i+}。

2.根据权利要求1所述的处理器,其中,所述执行单元可操作以响应于所述指令而将通过两轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}进行存储,并且其中,所述第一紧缩数据不包括状态数据元素c_i。

3.根据权利要求1所述的处理器,其中,所述执行单元可操作以响应于所述指令而将通过四轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}进行存储,并且其中,所述第一紧缩数据不包括状态数据元素c_i。

4.根据权利要求1所述的处理器,其中,所述指令仅将所述第一源和所述第二源指示为源,并且其中,所述第二紧缩数据包括:

状态数据元素c_i;

状态数据元素d_i;

第一和,其包括:状态元素h_i加上针对所述当前轮的消息输入W(i),加上针对所述当前轮的常数输入K(i),加上对针对所述当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算,加上对针对所述当前轮的状态元素e_i进行的Σ_1函数计算;以及

第二和,其包括:状态元素g_i加上针对在所述当前轮之后的一轮的消息输入W(i+1),加上针对在所述当前轮之后的一轮的常数输入K(i+1)。

5.根据权利要求1所述的处理器,其中,所述第二紧缩数据包括状态数据元素c_i、d_i、g_i和h_i。

6.根据权利要求5所述的处理器,其中,所述指令指示第三紧缩数据的第三源,其中,所述第三紧缩数据表示针对所述当前轮和针对在所述当前轮之后的一轮的消息输入和常数输入。

7.根据权利要求1所述的处理器,其中,存储所述结果包括:

将已经通过两轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}进行存储;并且将已经通过四轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}进行存储。

8.根据权利要求1所述的处理器,其中,所述第一紧缩数据仅包括所述八个状态数据元素中的四个。

9.根据权利要求1-8中的任意一项所述的处理器,其中,所述执行单元可操作以响应于

所述指令而执行操作，所述操作包括：

$(a_i$ ROTR 2) XOR (a_i ROTR 13) XOR (a_i ROTR 22)；以及

(a_i AND b_i) XOR (a_i AND c_i) XOR (a_i AND c_i)，

其中，ROTR表示向右旋转操作，AND表示逻辑与操作，且XOR表示逻辑异或操作。

10.一种处理器中的方法，包括：

接收指令，所述指令指示包括了针对安全哈希算法2SHA2哈希算法的当前轮i的状态数据元素a_i、b_i、e_i和f_i的第一紧缩数据的第一源，所述指令还指示第二紧缩数据的第二源，其中，所述第一紧缩数据具有的比特宽度小于所述SHA2哈希算法的八个状态数据元素a_i、b_i、c_i、d_i、e_i、f_i、g_i和h_i的组合的比特宽度；并且

响应于所述指令，在由所述指令指示的目的地中存储结果，所述结果包括已经通过至少一轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+}、b_{i+}、e_{i+}和f_{i+}。

11.根据权利要求10所述的方法，其中，存储包括将已经通过两轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}进行存储，并且其中，所述第一紧缩数据不包括状态数据元素c_i。

12.根据权利要求10所述的方法，其中，存储包括将已经通过四轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}进行存储，并且其中，所述第一紧缩数据不包括状态数据元素c_i。

13.根据权利要求10所述的方法，其中，所述指令仅将所述第一源和所述第二源指示为源，并且其中，所述第二紧缩数据包括：

状态数据元素c_i；

状态数据元素d_i；

第一和，其包括：状态元素h_i加上针对所述当前轮的消息输入W(i)，加上针对所述当前轮的常数输入K(i)，加上对针对所述当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算，加上对针对所述当前轮的状态元素e_i进行的Σ_1函数计算；以及

第二和，其包括：状态元素g_i加上针对在所述当前轮之后的一轮的消息输入W(i+1)，加上针对在所述当前轮之后的一轮的常数输入K(i+1)。

14.根据权利要求10所述的方法，其中，所述第二紧缩数据包括状态数据元素c_i、d_i、g_i和h_i，并且其中，所述指令指示第三紧缩数据的第三源，其中，所述第三紧缩数据表示针对所述当前轮和针对在所述当前轮之后的一轮的消息输入和常数输入。

15.根据权利要求10所述的方法，其中，存储所述结果包括：

将已经通过两轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}进行存储；并且将已经通过四轮的所述SHA2哈希算法从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}进行存储。

16.根据权利要求10所述的方法，其中，所述第一紧缩数据仅具有所述八个状态数据元素中的四个。

17.根据权利要求10所述的方法，还包括接收另一指令，所述另一指令用于分别使用状态数据元素a_i、b_i、e_i和f_i作为状态数据元素c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}，其中，状态数据元素c_{i+2}、

d_{i+2}、g_{i+2}和h_{i+2}表示通过两轮的所述SHA2哈希算法而更新的相对应的状态数据元素c_i、d_i、g_i和h_i。

18.一种处理器，包括：

多个紧缩数据寄存器；以及

执行单元，其与所述多个紧缩数据寄存器耦合，所述执行单元响应于指令而可操作，所述指令指示包括了针对安全哈希算法2SHA2哈希算法的当前轮i的状态数据元素g_i和h_i的第一紧缩数据的第一源，其中，所述第一紧缩数据具有的比特宽度小于所述SHA2哈希算法的八个状态数据元素a_i、b_i、c_i、d_i、e_i、f_i、g_i和h_i的组合的比特宽度，并且所述指令指示第二紧缩数据的第二源，所述第二紧缩数据表示针对所述当前轮和在所述当前轮之后的一轮的消息输入和常数输入，其中，所述第一源和所述第二源中的至少一个包括状态数据元素e_i和f_i，所述执行单元可操作以在由所述指令指示的目的地中存储结果，所述结果包括：

第一和，其包括：状态元素h_i加针对所述当前轮的消息输入W(i)，加上针对所述当前轮的常数输入K(i)，加上对针对所述当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算，加上对针对所述当前轮的状态元素e_i进行的Σ_1函数计算；以及

第二和，其包括：状态元素g_i加上针对在所述当前轮之后的一轮的消息输入W(i+1)，加上针对在所述当前轮之后的一轮的常数输入K(i+1)。

19.根据权利要求18所述的处理器，其中，所述第一紧缩数据包括状态数据元素c_i和状态数据元素d_i。

20.根据权利要求18所述的处理器，其中，所述第一紧缩数据包括状态数据元素e_i和状态数据元素f_i。

21.根据权利要求18所述的处理器，其中，所述执行单元可操作以将来自所述第一紧缩数据的两个额外的状态数据元素存储到所述结果。

22.根据权利要求18所述的处理器，其中，所述第一紧缩数据的比特宽度是所述八个状态数据元素的组合的比特宽度的一半，并且其中，所述指令仅将所述第一源和所述第二源指示为源。

23.一种处理器，包括用于执行根据权利要求10-17中的任意一项所述的方法的单元。

24.一种非瞬时性机器可读介质，其上存储有指令，所述指令在被机器执行时，使得所述机器执行根据权利要求10-17中的任意一项所述的方法。

25.一种电子设备，包括：互连单元；根据权利要求1-8中的任意一项所述的处理器，其与所述互连单元耦合；以及动态随机存取存储器，其与所述互连单元耦合。

用于处理安全哈希算法的指令处理器、方法、和系统

技术领域

[0001] 实施例涉及指令处理装置。具体地，实施例涉及用于处理安全哈希算法的指令处理装置和指令。

背景技术

[0002] 安全哈希标准(SHS)(FIPS PUB 180-3)，联邦信息处理标准公告，是由国家标准和技术研究所在2008年10月发布的。SHS标准指定了安全哈希算法SHA-224、SHA-256、SHA-384、以及SHA-512。在本申请中这四个算法还统称为SHA2哈希算法、SHA2算法、SHA2哈希等。

[0003] 这些SHA2哈希算法允许计算代表被称为消息的输入数据的浓缩表示的消息摘要。当长度小于2^64比特(对于SHA-224以及SHA-256)或小于2^128比特(对于SHA-384以及SHA-512)的消息被输入到哈希算法时，输出被称为消息摘要的结果。有时，消息摘要也被称为摘要或哈希。消息摘要对于SHA-224是224比特，对于SHA-256是256比特，对于SHA-384是384比特，或对于SHA-512是512比特。SHA-224和SHA-256是基于32比特字长的。SHA-384和SHA-512是基于64比特字长的。

[0004] 在该标准中指定的哈希算法被称为是安全的，这是因为对于给定的算法，认为1)找到与给定的消息摘要对应的消息，或2)找到产生相同的消息摘要的两个不同的消息在计算上是不可行的。这意味着对消息的任何改变将会(具有较高的可能性)导致不同的消息摘要。

[0005] SHA2算法在电子设备中被广泛使用以用于认证、验证、识别、完整性检查、或其它目的。它们可以用于各种不同的目的。SHA2算法的一种常见使用是验证消息的完整性和/或验证所检测到的对消息的改变。例如，可以针对消息生成初始的消息摘要，且然后，可以针对该消息重新生成另一消息摘要且假定消息自身没有被改变的情况下，该另一消息摘要应该与初始的消息摘要相同。安全哈希函数的其它示例包括但不限于生成数字签名、消息认证码、验证文件或消息的完整性、识别文件或数据、以及伪随机生成和密钥衍生。

[0006] 图1示出了单轮的SHA2算法100的细节。类似于所示的该轮的总共64轮可以用于计算最后的消息摘要。八个状态字a_i、b_i、c_i、d_i、e_i、f_i、g_i和h_i 101被输入到该轮。这八个状态字在标准中也被称为八个工作变量。对于SHA-224和SHA-256，这些状态字中的每一个都是32比特。对于SHA-384和SHA-512，这些状态字中的每一个都是64比特。此外，到该轮的输入为到当前轮的消息输入(即，W(i))102以及到当前轮的常数输入(例如，K(i))103。针对每一轮执行一组SHA2操作104。该组操作包括多个模加法(由内部具有加号的方框示出)，且对函数的计算被称为Ch、Σ_1、Maj、Σ_0。还存在对这些状态字的重映射。该轮的输出为八个已更新的状态字a_{i+1}、b_{i+1}、c_{i+1}、d_{i+1}、e_{i+1}、f_{i+1}、g_{i+1}和h_{i+1} 105。

[0007] 针对64轮中的每一个该组操作包括以下操作：

[0008] $\Sigma_0(a) = (a \text{ ROTR } 2) \text{ XOR } (a \text{ ROTR } 13) \text{ XOR } (a \text{ ROTR } 22)$

[0009] $\Sigma_1(e) = (e \text{ ROTR } 6) \text{ XOR } (e \text{ ROTR } 11) \text{ XOR } (e \text{ ROTR } 25)$

[0010] $Maj(a,b,c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$

[0011] $Ch(e,f,g) = (e \text{ AND } f) XOR ((NOTe) AND\ g)$

[0012] $T_1 = h + \sum_1(e) + Ch(e,f,g) + K_i + W_i$

[0013] $T_2 = \sum_0(a) + Maj(a,b,c)$

[0014] $h = g$

[0015] $g = f$

[0016] $f = e$

[0017] $e = d + T_1$

[0018] $d = c$

[0019] $c = b$

[0020] $b = a$

[0021] $a = T_1 + T_2$

[0022] 在上文中，“ROTR”指定了逐位右旋转操作其右侧的比特数目，“XOR”指定了逻辑异或操作，“AND”指定了逻辑与操作，“NOT”指定了逻辑非操作。旋转量是对于SHA-256特定的。其它的SHA2算法使用不同的移位和旋转量。

[0023] 如可以看到的，SHA2算法的每一轮涉及大量的操作。此外，生成消息摘要涉及六十四轮这样的轮。一个巨大的挑战在于传统地实现SHA2算法涉及在处理器上执行大量的指令。通常，SHA2算法中的每一轮可能采用从若干到许多个指令。例如，在一个可能的实现中，在一轮内，单独的指令可以用于执行旋转、逻辑AND、XOR、和NOT操作、加法等中的一个。这与存在64轮的现实相结合，可以趋向于使得实现SHA2算法是非常计算密集的并且花费大量的时间。

附图说明

[0024] 通过参考以下用于示出本发明实施例的描述和附图可以最佳地理解本发明。在附图中：

[0025] 图1示出了单轮安全哈希算法2(SHA2)哈希算法的细节。

[0026] 图2是具有包括对执行一个或多个SHA2算法有用的一个或多个指令的指令集的指令处理装置的框图。

[0027] 图3是具有SHA2执行逻辑的指令处理装置的实施例的框图，SHA2执行逻辑可操作以执行对处理SHA2安全哈希算法有用的指令的至少一个实施例。

[0028] 图4是处理对SHA2安全哈希算法有用的指令的方法的实施例的方框流程图。

[0029] 图5示出了一轮SHA2算法的一部分。

[0030] 图6是寄存器中的四个状态字或元素a、b、e和f的子集的实施例的框图。

[0031] 图7A是由SHA2输入指令的第一实施例执行的操作的框图。

[0032] 图7B是由SHA2两轮指令的实施例执行的操作的框图。

[0033] 图8A是由SHA2输入指令的第二实施例执行的操作的框图。

[0034] 图8B是由SHA2两轮低更新指令的实施例执行的操作的框图。

[0035] 图8C是由SHA2两轮高更新指令的实施例执行的操作的框图。

[0036] 图9是由SHA2 128比特的两轮指令的实施例执行的操作的框图。

[0037] 图10是由SHA2 256比特的两轮指令的实施例执行的操作的框图。

[0038] 图11是由SHA2 128比特的四轮指令的实施例执行的操作的框图。

[0039] 图12是由SHA2 512比特的四轮指令的实施例执行的操作的框图。

[0040] 图13A-C是合适的指令格式的框图。

[0041] 图14是适当组的紧缩数据寄存器的示例性实施例的框图。

[0042] 图15A示出了示例性的AVX指令格式，其包括VEX前缀、真实操作码字段、Mod R/M字节、SIB字节、位移字段、以及IMM8。

[0043] 图15B示出了来自图15A的哪些字段组成了完整的操作码字段以及基础操作字段。

[0044] 图15C示出了来自图15A的哪些字段组成了寄存器索引字段1544。

[0045] 图16是根据本发明一个实施例的寄存器架构的框图。

[0046] 图17A是根据本发明实施例的示出了示例性有序管线和示例性寄存器重命名、乱序发布/执行管线两者的框图。

[0047] 图17B示出了处理器内核，其包括耦合到执行引擎单元的前端单元，且这两者都耦合到存储器单元。

[0048] 图18A是根据本发明实施例的单个处理器内核，连同到其管芯上互连网络的连接以及其级别2(L2)的本地子集一起的框图。

[0049] 图18B是根据本发明实施例的在图18A中的处理器内核的一部分的扩展图。

[0050] 图19是根据本发明实施例的处理器的框图，该处理器可以具有超过一个的内核、可以具有集成存储器控制器、且可以具有集成显卡。

[0051] 图20示出了根据本发明一个实施例的系统的框图。

[0052] 图21根据本发明一个实施例示出了第一更具体的示例性系统的框图。

[0053] 图22根据本发明一个实施例示出了第二更具体的示例性系统的框图。

[0054] 图23根据本发明一个实施例示出了SoC的框图。

[0055] 图24是根据本发明实施例的对比了使用软件指令转换器来将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。

具体实施方式

[0056] 本申请中公开的是对执行SHA2哈希算法(例如，SHA-224、SHA-256、SHA-384、以及SHA-512)有用的指令、用于执行该指令的处理器、当处理或执行该指令时由该处理器执行的方法、以及并入一个或多个处理器以处理或执行该指令的系统。在下文的描述中，阐述了许多特定的细节(例如，特定的指令功能、数据格式、寄存器中的数据编排、指令格式、处理器配置、执行逻辑、微架构细节、操作序列等)。然而，要理解的是可以在没有这些具体细节的情况下实施本发明的实施例。在其它实例中，为了不模糊对本说明书的理解，尚未详细示出公知的电路、结构和技术。

[0057] 图2是处理器或其它指令处理装置206的示例性实施例的框图，该处理器或其它指令处理装置206具有指令集207，该指令集207包括对执行一个或多个SHA2算法有用的一个或多个指令208。处理器可以是各种复杂指令集计算(CISC)处理器、各种精简指令集计算(RISC)处理器、各种超长指令字(VLIW)处理器、和其中混合、或完全地其它类型的处理器中的任何一个。在一些实施例中，处理器可以是通用处理器(例如，在台式机、膝上型计算机、服务器等计算机上使用的类型)。或者，处理器可以是专用处理器。适合的专用处理器的示

例包括，但不限于，举几个例子来说，加密处理器、通信处理器、网络处理器、数字信号处理器(DSP)、加密协处理器、嵌入式处理器、图形处理器、以及控制器(例如，微控制器)。

[0058] 处理器和装置具有指令集架构(ISA)209。ISA表示与编程相关的处理器架构的一部分，且通常包括本地指令、架构寄存器(architectural register)、数据类型、寻址模式、存储器架构、中断和异常处理、以及处理器的外部输入和输出(I/O)。ISA与微架构不同，其通常表示选择用于实现ISA的特定处理器设计技术。

[0059] ISA包括指令集207。指令集的指令表示宏指令(例如，向处理器提供以执行的指令)，与微指令或微操作(例如，由解码宏指令的处理器的解码器所得到的那些)相对比。指令集包括一个或多个指令208，一个或多个指令208中的每一个对处理执行一个或多个SHA2算法是有用的。

[0060] ISA还包括在架构上可见的寄存器210。架构寄存器通常表示管芯上处理器存储器位置。架构寄存器还可以在本申请中被简单称为寄存器。短语架构寄存器、寄存器堆、以及寄存器在本申请中用于指代对软件和/或程序员可见(例如，软件可见)的寄存器和/或由宏指令指定以标识操作数的寄存器。这些寄存器与给定微架构中的非架构性或非架构可见的寄存器相对比(例如，由指令所使用的临时寄存器、重排序缓冲器、退出寄存器等)。所示出的寄存器包括紧缩(packed)数据寄存器211，紧缩数据寄存器211中的每一个可操作以存储紧缩的、矢量、或单指令多数据(SIMD)数据。用于处理SHA2算法208的指令可以指示紧缩数据寄存器中的源数据并且指示将在紧缩数据寄存器中存储的结果数据的目的地。

[0061] 处理器或装置还包括SHA2执行逻辑212。SHA2执行逻辑可以包括响应于指令的执行单元、功能单元、电路等。SHA2执行逻辑可操作以执行或处理指令208。

[0062] 图3是处理器或其它指令处理器装置306的示例性实施例的框图，上述处理器或其它指令处理器装置306具有执行逻辑312，该执行逻辑312可操作以执行对处理SHA2安全哈希算法有用的指令308的至少一个实施例。在一些实施例中，指令处理装置可以是处理器和/或可以被包含在处理器中。在一些实施例中，指令处理器装置可以被包含在图2的装置中，或指令处理装置306可以被包含在类似的或不同的装置中。

[0063] 装置306可以接收指令308。例如，指令可以从指令提取单元、指令队列、或存储器接收。指令可以代表机器指令、微指令、或由装置识别且控制装置执行特定操作的控制信号。上述指令可以显式指定(例如，通过比特或一个或多个字段)或另外指示(例如，隐式指示)第一源314。该指令还可以显式指定或另外指示第二源316。指令还可以显式指定或另外指示要存储指令结果的目的地318(例如，目的地存储位置)。在一些实施例中，第一源和第二源中的一个可以被重用为目的地(例如，指令的源/目的地字段可以指定被用作源和目的地的寄存器)。

[0064] 所示出的装置包括指令解码单元或解码器313。解码器可以接收和解码高级机器指令或宏指令并输出一个或多个低级微操作、微代码入口点、微指令、或反映原始高级指令和/或从原始高级指令得到的其它低级指令或控制信号。一个或多个低级指令或控制信号可以通过一个或多个低级(例如，电路级或硬件级)操作来实现高级指令的操作。可以使用各种不同的机制来实现解码器，各种不同的机制包括但不限于微代码只读存储器(ROM)、查询表、硬件实现、可编程逻辑阵列(PLA)、以及用于实现本领域已知的解码器的其它机制。

[0065] 在其它实施例中，可以使用指令仿真器、转换器、变形器、解释器、或其它指令转换

逻辑而不具有解码器313。各种不同类型的指令转换逻辑是本领域已知的，且其可以以软件、硬件、固件、或其组合来实现。指令转换逻辑可以接收指令，仿真、翻译、变型、解释、或另外将所接收的指令转换成一个或多个相对应的得到的指令或控制信号。在其它的实施例中，可以使用指令转换逻辑和解码器两者。

[0066] 装置还包括一组架构紧缩数据寄存器311。可以使用公知的技术在不同的微架构中以不同的方式来实现紧缩数据寄存器，且其不限于任何特定类型的电路。各种不同类型的寄存器是适合的。适合类型的寄存器示例包括，但不限于专用物理寄存器、使用寄存器重命名的动态地分配的物理寄存器、及其组合。如所示出的，在一些实施例中，第一源314、第二源316、以及目的地318各自可以是紧缩数据寄存器中的一个。或者，存储器位置及其它适合的存储位置可以用于这些中的一个或多个。

[0067] SHA2执行逻辑312与紧缩数据寄存器耦合311并且与解码器313耦合。SHA2执行逻辑可以从解码器接收一个或多个微操作、微代码入口点、微指令、其它指令、或反映指令308或从指令308得到的其它控制信号。响应于和/或作为指令308的结果，SHA2执行逻辑可以可操作以将紧缩数据结果319存储在由指令指定或另外指示的目的地318中。SHA2执行逻辑和/或装置可以包括可操作以执行和/或处理指令的具体或特定逻辑(例如，潜在地与固件和/或软件结合的电路或其它硬件)，并且响应于该指令(例如，响应于从该指令得到的一个或多个微指令或其它控制信号)而存储该结果。

[0068] 在一些实施例中，第一源314可以包括第一紧缩数据315，第一紧缩数据315包括针对SHA2哈希算法的当前轮(i)的状态数据元素a_i、b_i、e_i和f_i，且第二源316可以包括第二紧缩数据(例如，在各个实施例中状态数据元素和/或消息和常数数据)。在一些实施例中，结果319可以包括通过SHA2哈希算法的至少一轮而已经根据第一源314的相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+}、b_{i+}、e_{i+}和f_{i+}。例如，已更新的状态数据元素a_{i+}可以表示针对当前轮的、由SHA2哈希算法的一轮所更新的相对应的开始状态数据元素a_i，已更新的状态数据元素b_{i+}可以表示由SHA2哈希算法的一轮所更新的相对应的开始状态数据元素b_i，以此类推。

[0069] 在一些实施例中，第一紧缩数据315可以具有比SHA2哈希算法的八个数据元素(即，a_i、b_i、e_i和f_i以及其它的4个c_i、d_i、g_i和h_i)的组合比特宽度更小的比特宽度。在一些实施例中，第一紧缩数据的比特宽度可以为大约SHA2哈希算法的八个状态数据元素的组合比特宽度的一半。例如，在SHA-256的情况下，八状态数据元素中的每一个可以是32比特且八个状态数据元素的组合比特宽度可以是256比特，而第一紧缩数据可以仅具有128比特的宽度(例如，存储在128比特寄存器中)且可以仅保留八个32比特状态数据元素中的四个(例如，a_i、b_i、e_i和f_i)。作为另一示例，在SHA-512的情况下，八个状态数据元素中的每一个可以是64比特且八个状态数据元素的组合比特宽度可以是512比特，而第一紧缩数据可以仅具有256比特宽度(例如，存储在256比特寄存器中)并且能够仅保留八个64比特状态数据元素中的四个。

[0070] 在一些实施例中，该结果可以包括已经通过两轮的SHA2哈希算法而从第一源的相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}。在一些实施例中，该结果可以包括已经通过四轮的SHA2哈希算法而从第一源的相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}。下文将进一步

描述这些指令的特定示例。

[0071] 在一些实施例中，上述指令可以指定两个源或仅指定两个源(即，不具有第三源)。在其它实施例中，除了第一和第二源之外，该指令还可以指示第三源(例如，隐式指示或显式指定第三源)。下文将进一步描述这些指令的具体示例。

[0072] 如下文将进一步解释的，在一些实施例中，对SHA2轮的一些处理可以在执行指令的限制之外执行。例如，如下文将进一步解释的，在一些实施例中，X和Y的计算可以由另一指令执行。作为另一示例，在一些实施例中，可以在该轮之外执行消息数据的计算和/或消息和常数的加法。在一些实施例中，指令的执行可以包括Σ_0函数的操作(例如，针对SHA-256的$(a_i \text{ ROTR } 2) \text{XOR} (a_i \text{ ROTR } 13) \text{XOR} (a_i \text{ ROTR } 22)$)和/或Maj函数(例如，针对SHA-256的$(a_i \text{ AND } b_i) \text{XOR} (a_i \text{ AND } c_i) \text{XOR} (a_i \text{ ROTR } c_i)$)。

[0073] 优选地，在一些实施例中，单个指令可以用于通过至少一轮的SHA2算法来更新状态数据元素中的四个。这可以有助于显著提高实现SHA2算法的效率和/或速度。

[0074] 为了避免模糊该描述，已经示出并描述了相对简单的装置306。在其它实施例中，该装置可以可选地包括其它组件，诸如例如指令提取单元、指令调度单元、分支预测单元、指令和数据高速缓存、指令和数据翻译后备缓冲器、预提取缓冲器、微指令队列、微指令定序器、总线接口单元、第二或高级高速缓存、退出单元、寄存器重命名单元、处理器中包含的其它组件、以及它们的各种组合。实施例可以具有多个内核、逻辑处理器、或执行引擎。可操作以执行本申请中公开的至少一个指令的实施例的SHA2执行逻辑可以被包含在内核、逻辑处理器、或执行引擎中的至少一个中。照字面地，在处理器中存在这样的组件的许多种不同的组合/配置且本发明的范围不限于任何这样的组合/配置。

[0075] 图4是处理对SHA2安全哈希算法有用的指令的方法320的实施例的方框流程图。在各个实施例中，该方法可以由通用处理器、专用处理器(例如，加密的协处理器或内核)、或其它类型的指令处理器装置来执行。在一些实施例中，方法320可以由图2和/或图3中的装置或类似的装置来执行。或者，方法320可以由不同的装置来执行。本申请中描述的针对该装置的组件、特征、和特定的可选细节也可选地应用于方法320，在实施例中，方法320可以由装置和/或利用装置来执行。此外，图2和/或图3中的装置可以执行与图4中的操作和方法相同、类似、或不同的操作和方法。

[0076] 该方法包括在框321接收指令。在各个方面，可以在处理器、指令处理器装置、或其一部分(例如，解码器、指令转换器等)处从处理器外的源(例如，从主存储器、光盘、或总线或互连)、或从处理器上的源(例如，从指令高速缓存)接收指令。

[0077] 在一些实施例中，指令指示了针对SHA2哈希算法的当前轮(i)的包含状态数据元素a_i、b_i、e_i和f_i的第一紧缩数据的第一源。该指令还指示了第二紧缩数据的第二源。在一些实施例中，第一紧缩数据可以具有比SHA2哈希算法的八个状态数据单元的组合比特宽度更小的比特宽度。

[0078] 在框322，在目的地中存储结果，该目的地是由该指令响应于、作为其结果、和/或如由该指令所指定的而指定的或另外指示的。在一些实施例中，该结果可以包括已经通过至少一轮的SHA2哈希算法而从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的已更新的状态数据元素a_{i+}、b_{i+}、e_{i+}和f_{i+}。

[0079] 所示出的方法包括架构可见(例如，从软件的角度可见)的操作。在其它实施例中，

该方法可以可选地包括一个或多个微架构操作。例如，可以提取该指令并将其解码(或另外转换)成一个或多个指令或控制信号。可以访问和/或接收源操作数。可以启用执行单元或执行逻辑来执行由指令指定的操作，且执行单元或执行逻辑可以执行该操作(例如，可以执行微架构操作以实现指令的操作)。例如，如在本申请中的别处描述的，可以执行排他的OR操作、旋转操作、加法操作等。考虑了执行操作的不同的微架构方式。其它的方法实施例可以包括一个或多个这样的非架构性可见操作。

[0080] 图5示出了SHA2算法的轮534的一部分，其中更新了状态字或元素a-h。SHA2算法的性质是在每一轮内，仅状态字a和e的值是新的且不是预先确定的。基于SHA2算法的函数、消息输入和常数输入等在每一轮确定状态字a和e。在示出中，这些函数被并入到T_1和T_2中。所有其它的状态字都具有旧的或预先确定的值，其已经从其它状态元素中的一个被简单地移动。例如，在该轮之后，状态元素c_{i+1}的值等于当前轮的状态元素b_i的值等。因此，在两轮的SHA2算法之后，当前轮的状态元素a_i的值变成状态元素c_{i+2}的值，在两轮之后，b_i的值变成d_{i+2}的值，在两轮之后，e_i的值变成g_{i+2}的值，且在两轮之后，f_i的值变成h_{i+2}的值。

[0081] 图6是寄存器615中四个状态字或元素a_i、b_i、e_i和f_i的有用的子集的实施例的框图。可以以所示的顺序或替代地以各种不同的顺序存储这些状态字或元素。将这些四个状态字存储在寄存器中的有用后果和优点在于在两轮的SHA2算法之后，它们等于四个互补的状态元素c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}。在一些实施例中，寄存器615的元素a_i、b_i、e_i和f_i的值可以简单地移动、复制、或另外存储在寄存器619中，而无需计算c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}中的任何一个。

[0082] 图7A-B示出了可操作以生成已更新的状态数据元素c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}的互补的指令对的操作，这些已更新的状态数据元素已经通过两轮的SHA2哈希算法而从状态数据元素a_i、b_i、e_i和f_i进行了更新。在一些实施例中，指令仅使用两个源操作数(例如，对仅允许两个资源操作数被指定用于这些指令的ISA或微架构有用)。上述指令利用128比特紧缩数据和/或比特宽度是SHA2哈希算法的八个32比特状态数据元素的组合比特宽度(即，256比特)的一半的寄存器。尽管状态元素的完整宽度将适合两个这样的寄存器，但是SHA2算法的消息输入和常数输入也需要被引入。该对指令提供了一个指令(即，图7A中的指令)以引入消息输入和常数输入且提供了另一指令(即，图7B中的指令)以更新状态元素。在其它实施例中，类似的指令对可以用于使用256比特紧缩数据和/或寄存器的具有512比特的组合状态的SHA2算法。

[0083] 图7A是由SHA2输入指令(SHA2_in)的实施例执行的操作740的框图。上述指令指定或另外指示第一源714A、指定或另外指示第二源716A、并指定或另外指示目的地718A。在一些实施例中，第一源、第二源和目的地可以是128比特寄存器或其它存储器位置。第一源具有包括四个32比特的状态数据元素c_i、d_i、g_i和h_i的第一128比特紧缩数据。例如，在示出中，h_i存储在比特[31:0]中，g_i存储在比特[63:32]中，d_i存储在比特[95:64]中，且c_i存储在比特[127:96]中，但是并不要求该特定的顺序。

[0084] 第二源具有包括两个32比特状态数据元素e_i和f_i的第二128比特紧缩数据。在所示出的实施例中，e_i存储在比特[127:96]中，且f_i存储在比特[64:95]中，但是并不要求该特定的顺序。第二源也具有代表针对两轮的SHA2算法(即，当前轮和在当前轮之后的一轮)的消息输入和常数输入的两个32比特数据元素。在所示出的实施例中，在[31:0]中存储了第一数据元素，该第一数据元素表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K

(i)，且在[63:32]中存储了第二数据元素，该第二数据元素表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)。在另一实施例中，W(i)、W(i+1)、K(i)和K(i+1)中的每一个可以单独地存储在第二紧缩数据的四个数据元素中。

[0085] SHA2执行逻辑712A可操作以响应于指令而在目的地中存储128比特紧缩数据结果。在一些实施例中，该结果包括四个结果数据元素。在所示出的实施例中，第一结果数据元素(Y)存储在比特[31:0]中。第一结果数据元素(Y)表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)(即，W(i)+K(i))，加上针对当前轮的状态数据元素h_i，加上对针对当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算(即，Ch(e_i、f_i、g_i)，加上对针对当前轮的状态元素e_i进行的sigma 1函数计算(即，$\Sigma_1(e_i)$)的和。

[0086] 第二结果数据元素(X)存储在比特[63:32]中。第二结果数据元素(X)存储在比特[63:32]中。第二结果数据元素(X)表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)(即，W(i+1)加K(i+1))，加上针对当前轮的状态数据元素g_i的和。X和Y元素包括针对两轮的消息输入和常数输入且Y函数并入了对Ch和sigma 1函数的计算。X和Y元素在SHA2算法中未定义，相反X和Y元素是针对在本申请中公开的指令的这些参数的新组合，并且其被任意命名。

[0087] 该结果还包括存储在比特[95:64]中的针对当前轮的状态数据元素d_i以及存储在比特[127:96]的针对当前轮的状态数据元素c_i。不要求在目的地中的该特定的顺序。在一些实施例中，第一源被重用作为目的地且数据元素c_i、d_i、X、Y覆盖数据元素c_i、d_i、g_i和h_i。一旦已经确定X和Y元素，就不再需要状态元素g_i和h_i。在一些实施例中，这可以允许利用仅具有两个源操作数的单个指令来计算在两轮之后的状态元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}(例如，如在图7B中所示出的)。

[0088] 图7B是由SHA2两轮指令(SHA2_RNDS2)的实施例执行的操作730B的框图。该指令指定或另外指示第一源714B、指定或另外指示第二源716B、且指定或另外指示目的地718B。在一些实施例中，第一源、第二源、和目的地可以是128比特寄存器或其它存储位置。

[0089] 第一源具有包括四个32比特数据元素的第一128比特紧缩数据。在一些实施例中，第一128比特紧缩数据可以等于图7A中的SHA2输入指令的结果或与图7A中的SHA2输入指令的结果相同。在一些实施例中，SHA2输入指令中的目的地718A可以被指示为SHA2两轮指令的第一源714B。如在描述中所示出的，第一源包括在比特[31:0]中存储的第一数据元素(Y)。第一数据元素(Y)表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)(即，W(i)+K(i))，加上针对当前轮的状态数据元素h_i，加上对针对当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算(即，Ch(e_i、f_i、g_i)，加上对针对当前轮的状态元素e_i进行的sigma 1函数计算(即，$\Sigma_1(e_i)$))的和。第二数据元素(X)存储在比特[63:32]中。第二数据元素(X)表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对当前轮之后的一轮的常数输入K(i+1)(即，W(i+1)加K(i+1))，加上针对当前轮的状态数据元素g_i的和。紧缩数据还包括存储在比特[95:64]中的针对当前轮的状态数据元素d_i以及存储在比特[127:96]中的针对当前轮的状态数据元素c_i。不要求源中的该特定的顺序。

[0090] 第二源具有包括四个32比特状态数据元素a_i、b_i、e_i和f_i的第二128比特紧缩数据。在所示出的实施例中，a_i存储在比特[127:96]中，b_i存储在比特[95:64]中，e_i存储在比特

[63:32]中，且f_i存储在比特[31:0]中，尽管不要求该特定的顺序。

[0091] SHA2执行逻辑712B可操作以响应于该指令而在目的地718中存储128比特紧缩数据结果。在一些实施例中，该结果包括已经通过两轮的SHA2哈希算法分别从相对应的状态数据元素a_i、b_i、e_i和f_im更新的四个已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}或f_{i+2}。在所示出的实施例中，a_{i+2}存储在比特[127:96]中，b_{i+2}存储在比特[95:64]中，e_{i+2}存储在比特[63:32]中，且f_{i+2}存储在比特[31:0]中，尽管不要求该特定的顺序。其它实施例可以通过单轮而不是两轮或通过多于两轮来更新状态元素。

[0092] 如上文所讨论的，当前轮状态变量a_i、b_i、e_i和f_i分别等同于轮i+2状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}。当前轮状态变量a_i、b_i、e_i和f_i存储在第二源中。有利地，轮i+2状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}不需要单独计算。相反，当前轮状态变量a_i、b_i、e_i和f_i可以仅被重用作轮i+2状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}。例如，可以将第二源中的当前轮状态变量a_i、b_i、e_i和f_i与目的地中的结果结合，以提供已经通过两轮而更新的八个已更新的状态数据元素的完整组。

[0093] 有利地，这些指令允许通过两轮通过执行两个指令而更新SHA2算法的所有八个状态元素的值，除了用于生成消息输入和常数输入和将消息输入和常数输入相加(例如，W(i)+K(i))等的一些操作之外。在一些实施例中，两个指令都可以以大约每轮3个循环而执行，尽管本发明的范围不受此限制。状态元素a_i、b_i、e_i和f_i的新颖布置在该方面是有帮助的。此外，仅需要指定两个源。此外，寄存器可以是SHA2算法的状态元素组合的一半。与使用两倍的寄存器和执行宽度相对照，使用这样更小的寄存器，以及相关联的执行宽度通常有助于减小处理器或集成电路的成本和功耗。这可倾向于使得实现这些指令对低成本和/或移动或电池供电的电子设备有用。

[0094] 下文列出的是伪代码，在一些实施例中，这些伪代码可以使用SHA2-IN和SHA2_RNDS2指令，其中WKi＝W(i)+K(i)：

[0095] 给定:XMM0＝abef且XMM1＝cdgh

[0096] i轮：

```
MOV          XMM2, XMM0                 //XMM2=abef
BLEND        XMM0, mem(WK2,WK1)         //XMM0= WK2,WK1,e,f
```

[0097]

```
SHA2_IN      XMM1, XMM0                 //XMM1= cdX1Y1
SHA2_RNDS2   XMM1, XMM2                 // XMM1=a2,b2,e2,f2
             //XMM0=c2,d2,g2,h2
```

[0098] i+2轮：

```
MOV          XMM2, XMM1                 //XMM2=a2b2e2f2
BLEND        XMM1, mem(WK4,WK3)         //XMM0=WK4,WK3,e2, f2
```

[0099]

```
SHA2_IN      XMM0, XMM1                 //XMM1= c2d2X2Y2
SHA2_RNDS2   XMM0, XMM2                 // XMM1=a4,b4,e4,f4
                                        //XMM0=c4,d4,g4,h4
```

[0100] 图8A-C示出了可操作以生成所有的八个已更新的状态数据元素a_{i+2}、b_{i+2}、c_{i+2}、

d_{i+2}、e_{i+2}、f_{i+2}、g_{i+2}和h_{i+2}的三个指令的互补集的操作，上述已更新的状态数据元素已经通过两轮的SHA2哈希算法从状态数据元素a_i、b_i、c_i、d_i、e_i、f_i、g_i和h_i更新。三个指令中的每一个仅利用两个源操作数。上述指令利用128比特紧缩数据和/或具有比特宽度(即，128比特)是SHA2哈希算法的八个32比特状态数据元素的组合的比特宽度(即，256比特)的一半的寄存器。在其它实施例中，类似的一组三个指令可以用于使用256比特紧缩数据和/或寄存器、具有512比特的组合状态的SHA2算法。

[0101] 图8A是由SHA2输入指令(SHA2_in)的实施例执行的操作840的框图。该指令指定或另外指示第一源814A，指定或另外指示第二源816A，且指定或另外指示目的地818A。在一些实施例中，第一源、第二源和目的地可以是128比特寄存器或其它存储位置。第一源具有包括四个32比特状态数据元素e_i、f_i、g_i和h_i的第一128比特紧缩数据。例如，在示出中，h_i存储在比特[31:0]中，g_i存储在比特[63:32]中，f_i存储在比特[95:64]中，且e_i存储在比特[127:96]中，尽管不要求该特定的顺序。

[0102] 第二源具有第二紧缩数据，第二紧缩数据包括表示针对两轮的SHA2算法(即，当前轮和在当前轮之后的一轮)的消息输入和常数输入的两个32比特数据元素。如所示出的，在一些实施例中，第二源和/或第二紧缩数据可以是128比特，其中一半比特(例如，上半部)为不介意值(*)且比特的另一半保持两个32比特数据元素。或者，可以使用64比特源和/或紧缩数据。在所示出的实施例中，将表示针对当前轮的消息输入W(i)加上当前轮的常数输入K(i)的第一数据元素存储在[31:0]中，且将表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)的第二数据元素存储在[63:32]中。在另一实施例中，W(i)、W(i+1)、K(i)和K(i+1)中的每个可以单独存储在第二紧缩数据的四个数据元素中。

[0103] SHA2执行逻辑812A可操作以响应于该指令而在目的地中存储128比特紧缩数据结果。在一些实施例中，该结果包括四个结果数据元素。在所示出的实施例中，第一结果数据元素(Y)存储在比特[31:0]中。第一结果数据元素(Y)表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)(即，W(i)+K(i))，加上针对当前轮的状态数据元素h_i，加上对针对当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算(即，Ch(e_i、f_i、g_i))，加上对针对当前轮的状态元素e_i进行的sigma 1函数计算(即，$\Sigma_1(e_i)$)的和。

[0104] 第二结果数据元素(X)存储在比特[63:32]中。第二结果数据元素(X)表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)(即，W(i+1)加K(i+1))，加上针对当前轮的状态数据元素g_i的和。该结果还包括存储在比特[95:64]中的针对当前轮的状态数据元素f_i以及存储在比特[127:96]中的针对当前轮的状态数据元素e_i。不要求在目的地中的该特定的顺序。在一些实施例中，第一源被重用作目的地且该结果的数据元素e_i、f_i、X、Y覆盖第一源的数据元素e_i、f_i、g_i和h_i，尽管这是不要求的。

[0105] 图8B是由SHA2两轮低更新指令(SHA2_LO)的实施例执行的操作830B的框图。该指令指定或另外指示第一源814B，指定或另外指示第二源816B，以及指定或另外指示目的地818B。在一些实施例中，第一源、第二源、和目的地可以是128比特寄存器或其它存储位置。

[0106] 第一源具有包括四个32比特数据元素的第一128比特紧缩数据。在一些实施例中，第一128比特紧缩数据可以等于图8A中的SHA2输入指令的结果或与该结果相同。在一些实施例中，图8A中的SHA2输入指令的目的地818A可以被指示为图8B中的SHA2两轮低指令的第

一源814B。如在描述中所示出的，第一源包括存储在比特[31:0]中的第一数据元素(Y)。第一数据元素(Y)表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)(即，W(i)+K(i))，加上针对当前轮的状态数据元素h_i，加上对针对当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算(即，Ch(e_i、f_i、g_i)，加上对针对当前轮的状态元素e_i进行的sigma 1函数计算(即，$\Sigma_1(e_i)$)的和。第二数据元素(X)存储在比特[63:32]中。第二数据元素(X)表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)(即，W(i+1)加K(i+1))，加上针对当前轮的状态数据元素g_i的和。第一源还包括存储在比特[95:64]中的针对当前轮的状态数据元素f_i以及存储在比特[127:96]中的针对当前轮的状态数据元素e_i。不要求在第一源中的该特定的顺序。

[0107] 第二源具有包括四个32比特的状态数据元素a_i、b_i、c_i和d_i的第二128比特紧缩数据。在所示出的实施例中，a_i存储在比特[127:96]中，b_i存储在比特[95:64]中，c_i存储在比特[63:32]中，且d_i存储在比特[31:0]中，尽管不要求该特定的顺序。

[0108] SHA2执行逻辑812B可操作以响应于指令而在目的地818B中存储128比特紧缩数据结果。在一些实施例中，该结果包括已经通过两轮的SHA2哈希算法分别从相对应的状态数据元素e_i、f_i、g_i和h_i而更新的四个已更新的状态数据元素e_{i+2}、f_{i+2}、g_{i+2}或h_{i+2}。在所示出的实施例中，e_{i+2}存储在比特[127:96]中，g_{i+2}存储在比特[63:32]中，f_{i+2}存储在比特[95:64]中，且g_{i+2}存储在比特[63:32]中，且h_{i+2}存储在比特[31:0]中，尽管不要求该特定的顺序。其它实施例可以通过单轮而不是两轮或通过多于两轮来更新状态元素。

[0109] 图8C是由SHA2两轮高更新指令(SHA2-HI)的实施例执行的操作830C的框图。该指令指定或另外指示第一源814，指定或另外指示第二源816C，以及指定或另外指示目的地818C。在一些实施例中，第一源、第二源和目的地可以是128比特寄存器或其它存储位置。

[0110] 第一源具有包括四个32比特的状态数据元素a_i、b_i、c_i和d_i的第一128比特紧缩数据。在所示出的实施例中，a_i存储在比特[127:96]中，b_i存储在比特[95:64]中，c_i存储在比特[63:32]中，且d_i存储在比特[31:0]中，尽管不要求该特定的顺序。

[0111] 第二源具有包括四个32比特数据元素的第二128比特紧缩数据。在一些实施例中，第二128比特紧缩数据可以等于图8A中的SHA2输入指令的结果或与图8A中的SHA2输入指令的结果相同。在一些实施例中，图8A中的SHA2输入指令的目的地818A可以被指示为图8C中的SHA2两轮高指令的第二源816C。如在描述中所示出的，第二源包括存储在比特[31:0]中的第一数据元素(Y)。第一数据元素(Y)表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)(即，W(i)+K(i))，加上针对当前轮的状态数据元素h_i，加上对针对当前轮的状态元素e_i、f_i和g_i进行的Ch函数计算(即，Ch(e_i、f_i、g_i))，加上对针对当前轮的状态元素e_i进行的sigma 1函数计算(即，$\Sigma_1(e_i)$))的和。第二数据元素(X)存储在比特[63:32]中。第二数据元素(X)表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对当前轮之后的一轮的常数输入K(i+1)(即，W(i+1)加K(i+1))，加上针对当前轮的状态数据元素g_i的和。第二源还包括存储在比特[95:64]中的针对当前轮的状态数据元素f_i以及存储在比特[127:96]中的针对当前轮的状态数据元素e_i。不要求该第二源中的特定的顺序。

[0112] SHA2执行逻辑812C可操作以响应于指令而在目的地818C中存储128比特紧缩数据结果。在一些实施例中，该结果包括已经通过两轮的SHA2哈希算法分别从对相应的状态数据元素a_i、b_i、c_i和d_i而更新的四个已更新的状态数据元素a_{i+2}、b_{i+2}、c_{i+2}或d_{i+2}。在所示出的

实施例中，a_{i+2}存储在比特[127:96]中，b_{i+2}存储在比特[95:64]中，c_{i+2}存储在比特[63:32]中，且d_{i+2}存储在比特[31:0]中，尽管不要求该特定的顺序。其它实施例可以通过单轮而不是两轮或通过多于两轮来更新状态元素。

[0113] 在一些实施例中，SHA2输入指令、SHA2更新低指令、以及SHA2更新高指令的执行可以以总共大约6个循环或每轮3个循环来完成两轮的SHA2算法。在一方面，可以首先在大约3循环的管线中执行SHA2输入指令，然后是在随后的大约3个循环管线中的SHA2更新低指令和SHA2更新高指令。更新指令中的一个可以有效地隐藏在对其它指令的内行中(例如，可以跟随在其它指令一个循环后)。

[0114] 已经描述了仅指示两个源操作数的指令的实施例。其它实施例涉及指示三个源操作数的指令。在一些实施例中，这三个源操作数中的每一个已经具有最多为SHA2哈希算法中的八个状态元素的组合宽度的比特宽度的一半的紧缩数据。

[0115] 图9是由SHA2 128比特数据两轮指令(SHA256_2RND)的实施例执行的操作930的框图。该指令指定或另外指示第一源914、指定或另外指示第二源916、指定或另外指示第三源944、以及指定或另外指示目的地918。在一些实施例中，第一源、第二源、和第三源中的一个被重用作目的地。在一些实施例中，第一源、第二源、可能地第三源、以及目的地可以是128比特寄存器或其它存储位置。

[0116] 第一源具有包括四个32比特的状态数据元素c_i、d_i、g_i和h_i的第一128比特紧缩数据。例如，在示出中，h_i存储在比特[31:0]中，g_i存储在比特[63:32]中，d_i存储在比特[95:64]中，且c_i存储在比特[127:96]中，尽管并不要求该特定的顺序。

[0117] 第二源具有包括四个32比特的状态数据元素a_i、b_i、c_i和f_i的第二128比特紧缩数据。例如，在示出中，f_i存储在比特[31:0]中，e_i存储在比特[63:32]中，b_i存储在比特[95:64]中，且a_i存储在比特[127:96]中，尽管并不要求该特定的顺序。

[0118] 第三源具有包括两个32比特的数据元素的第三紧缩数据，这两个32比特的数据元素表示针对两轮的SHA2算法(即，当前轮和在当前轮之后的一轮)的消息输入和常数输入。如所示出的，在一些实施例中，第三源和/或第三紧缩数据可以是128比特，其中一半比特(例如，上半部)为不介意值(*)且比特的另一半保持两个32比特数据元素。或者，可以不同地布置数据。作为另一个选项，可以使用具有两个32比特数据元素的64比特源和/或紧缩数据。在所示出的实施例中，表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)的第一数据元素存储在[31:0]中，且表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)的第二数据元素存储在[63:32]中。在另一实施例中，W(i)、W(i+1)、K(i)和K(i+1)中的每一个可以存储在128比特紧缩数据的四个32比特数据元素中的不同的一个数据元素中。

[0119] SHA2执行逻辑912可操作以响应于指令而在目的地918中存储128比特紧缩数据结果。在一些实施例中，该结果包括已经通过两轮的SHA2哈希算法而分别从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的四个已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}或f_{i+2}。在所示出的实施例中，a_{i+2}存储在比特[127:96]中，b_{i+2}存储在比特[95:64]中，e_{i+2}存储在比特[63:32]中，且f_{i+2}存储在比特[31:0]中，尽管不要求该特定的顺序。其它实施例可以通过单轮而不是两轮或通过多于两轮来更新状态元素。

[0120] 图10是由SHA2 256比特数据两轮指令(SHA512RNDS2)的实施例执行的操作1030的

框图。指令/操作类似于图9中的那些指令/操作，除了针对具有两倍的状态(即，512比特的状态而不是256比特的状态)、对两倍大(即，64比特而不是32比特)的状态元素和消息输入和常数输入、并使用两倍大(即，256比特二不是128比特)的紧缩数据的SHA2算法之外。

[0121] 该指令指定或另外指示第一源1014、指定或另外指示第二源1016、指定或另外指示第三源1044、以及指定或另外指示目的地1018。在一些实施例中，第一源、第二源、和第三源中的一个被重用作目的地。在一些实施例中，第一源、第二源、可能地第三源、以及目的地可以是256比特寄存器或其它存储位置。

[0122] 第一源具有包括四个64比特的状态数据元素c_i、d_i、g_i和h_i的第一256比特紧缩数据。例如，在示出中，h_i存储在比特[63:0]中，g_i存储在比特[127:64]中，d_i存储在比特[191:128]中，且c_i存储在比特[255:192]中，尽管并不要求该特定的顺序。

[0123] 第二源具有包括四个64比特的状态数据元素a_i、b_i、c_i和f_i的第二256比特紧缩数据。例如，在示出中，f_i存储在比特[63:0]中，e_i存储在比特[127:64]中，b_i存储在比特[191:128]中，且a_i存储在比特[255:192]中，尽管并不要求该特定的顺序。

[0124] 第三源具有包括两个64比特的数据元素的第三紧缩数据，这两个64比特的数据元素表示针对两轮的SHA2算法(即，当前轮和在当前轮之后的一轮)的消息输入和常数输入。如所示出的，在一些实施例中，第三源和/或第三紧缩数据可以是256比特宽，其中一半比特(例如，上半部)为不介意值(*)且比特的另一半保持两个64比特数据元素。或者，可以不同地布置数据。作为另一个选项，可以使用具有两个64比特数据元素的128比特源和/或紧缩数据。在所示出的实施例中，表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)的第一数据元素存储在[63:0]中，且表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)第二数据元素存储在[127:64]中。在另一实施例中，W(i)、W(i+1)、K(i)和K(i+1)中的每一个可以存储在256比特紧缩数据的四个64比特数据元素中的不同的一个数据元素中。

[0125] SHA2执行逻辑1012可操作以响应于指令而在目的地1018中存储256比特紧缩数据结果。在一些实施例中，该结果包括已经通过两轮的SHA2哈希算法分别从相对应的状态数据元素a_i、b_i、e_i和f_i而更新的四个已更新的64比特状态数据元素a_{i+2}、b_{i+2}、e_{i+2}或f_{i+2}。在所示出的实施例中，a_{i+2}存储在比特[255:192]中，b_{i+2}存储在比特[191:128]中，e_{i+2}存储在比特[127:64]中，且f_{i+2}存储在比特[63:0]中，尽管不要求该特定的顺序。其它实施例可以通过单轮而不是两轮或通过多于两轮来更新状态元素。

[0126] 图9-10利用对一个源中的状态变量a_i、b_i、e_i和f_i的新颖布置。如上文所讨论的，当前轮状态变量a_i、b_i、e_i和f_i分别等图于i+2轮状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}。当前轮状态变量a_i、b_i、e_i和f_i存储在第二源中。有利地，i+2轮状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}不需要单独计算。相反，当前轮状态变量a_i、b_i、e_i和f_i可以仅被重用作i+2轮状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}。例如，可以将第二源中的当前轮状态变量a_i、b_i、e_i和f_i与目的地中的结果结合，以提供已经由两轮更新的八个已更新的状态数据元素的完整组。作为另一示例，随后的指令可以指示第二源中的当前轮的状态变量a_i、b_i、e_i和f_i，如同它们是i+2轮状态变量c_{i+2}、d_{i+2}、g_{i+2}和h_{i+2}，且它们可以被这样处理器并用于例如生成i+4轮等。

[0127] 有利地，这些指令允许通过两轮通过执行两个指令而更新SHA2算法的所有八个状态元素的值，除了用于生成消息输入和常数输入和将消息输入和常数输入相加(例如，W(i)

+K(i))等的一些操作之外。状态元素a_i、b_i、e_i和f_i的新颖布置在该方面有帮助。此外，寄存器可以是SHA2算法的状态元素组合宽度的一半。与使用两倍的寄存器和执行宽度相对照，使用这样更小的寄存器以及关联的执行宽度，通常有助于减少处理器或集成电路的成本和功耗。这可以倾向于使实现这些指令对低成本和/或移动或电池供电的电子设备有用。

[0128] 图11是由SHA2 128比特数据四轮指令(SHA256_4RND)的实施例执行的操作1130的框图。上述指令指定或另外指示第一源1114、指定或另外指示第二源1116、指定或另外指示第三源1144、指定或另外指示第一目的地1118A、以及指定或另外指示第二目的地1118B。在一些实施例中，这些源中的一个被重用作第一目的地且这些源中的另一个被重用作第二目的地。在一些实施例中，第一源、第二源、第三源和目的地可以是128比特寄存器或其它存储位置。

[0129] 第一源具有包括四个32比特的状态数据元素c_i、d_i、g_i和h_i的第一128比特紧缩数据。例如，在示出中，h_i存储在比特[31:0]中，g_i存储在比特[63:32]中，d_i存储在比特[95:64]中，且c_i存储在比特[127:96]中，尽管并不要求该特定的顺序。

[0130] 第二源具有包括四个32比特状态数据元素a_i、b_i、e_i和f_i的第二128比特紧缩数据。例如，在示出中，f_i存储在比特[31:0]中，e_i存储在比特[63:32]中，b_i存储在比特[95:64]中，且a_i存储在比特[127:96]中，尽管并不要求该特定的顺序。

[0131] 第三源具有第三128比特紧缩数据，第三128比特紧缩数据包括表示针对四轮的SHA2算法(即，当前轮(i)、在当前轮之后的一轮(i+1)、在当前轮之后的两轮(i+2)、以及在当前轮之后的三轮(i+3))的消息输入和常数输入的四个32比特数据元素。在所示出的实施例中，表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)的第一数据元素存储在[31:0]中，以及表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)的第二数据元素存储在[63:32]中。接着，表示针对在当前轮之后的两轮的消息输入W(i+2)加上针对在当前轮之后的两轮的常数输入K(i+2)的第三数据元素存储在[63:32]中，以及表示针对在当前轮之后的三轮的消息输入W(i+3)加上针对在当前轮之后的三轮的常数输入K(i+3)的第四数据元素存储在[63:32]中。在其它实施例中，可以不同地布置数据。

[0132] SHA2执行逻辑1112可操作以响应于指令而在由指令指示的第一目的地1118A中存储128比特紧缩数据结果，并且响应于指令而在由指令指示的第二目的地1118B中存储第二128比特紧缩数据。在一些实施例中，第一128比特紧缩数据结果包括四个已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}，状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}已经通过两轮的SHA2哈希算法而分别从相对应的状态数据元素a_i、b_i、e_i和f_i更新。在所示出的实施例中，a_{i+2}存储在比特[127:96]中，b_{i+2}存储在比特[95:64]中，e_{i+2}存储在比特[63:32]中，和f_{i+2}存储在比特[31:0]中，尽管不要求该特定的顺序。在一些实施例中，第二128比特紧缩数据结果包括四个已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}，状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}已经通过四轮的SHA2哈希算法而分别从相对应的状态数据元素a_i、b_i、e_i和f_i更新。在所示出的实施例中，a_{i+4}存储在比特[127:96]中，b_{i+4}存储在比特[95:64]中，e_{i+4}存储在比特[63:32]中，和f_{i+4}存储在比特[31:0]中，尽管不要求该特定的顺序。

[0133] 在一些实施例中，可以在已经完成两轮之后(例如，大约通过执行该指令的一半)在第一目的地中存储第一128比特紧缩数据结果，并且可以在已经完成四轮之后(例如，大

约完全执行该指令)在第二目的地中存储第二128比特紧缩数据结果。

[0134] 存储在第一目的地1118A中的两轮之后的状态变量a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}分别等同于四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}。有利地,无需单独计算四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}。相反,存储在第一目的地中的两轮之后的状态变量可以仅用作四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}。存储在第一目的地1118A中的两轮之后的状态变量a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}可以与存储在第二目的地1118B中的四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}组合,以获得的各自由四轮的SHA2算法更新的一组八个已更新的状态元素。在一些实施例中,第一源可以被重用作第一目的地且第二源可以被重用作第二目的地,尽管这不是要求的。

[0135] 图12是由SHA2 512比特数据四轮指令(SHA512RNDS4)的实施例执行的操作1230的框图。该指令/操作与图11中的指令/操作类似,除了针对具有两倍的状态(即,512比特的状态而不是256比特的状态)、对两倍大(即,64比特而不是32比特)的状态元素和消息输入和常数输入、并使用两倍大(即,256比特二不是128比特)的紧缩数据的SHA2算法之外。

[0136] 该指令指定或另外指示第一源1214,指定或另外指示第二源1216,指定或另外指示第三源1244、指定或另外指示第一目的地1218A,以及指定或另外指示第二目的地1218B。在一些实施例中,这些源中的一个被重用作第一目的地,且这些源中的另一个被重用作第二目的地。在一些实施例中,源和目的地可以是256比特寄存器或其它存储位置。

[0137] 第一源具有包括四个64比特状态数据元素c_i、d_i、g_i和h_i的第一256比特紧缩数据。例如,在示出中,h_i存储在比特[63:0]中,g_i存储在比特[127:64]中,d_i存储在比特[191:128]中,且c_i存储在比特[255:192]中,尽管不要求该特定的顺序。

[0138] 第二源具有包括四个64比特状态数据元素a_i、b_i、e_i和f_i的第二256比特紧缩数据。例如,在示出中,f_i存储在比特[63:0]中,e_i存储在比特[127:64]中,b_i存储在比特[191:128]中,且a_i存储在比特[255:192]中,尽管不要求该特定的顺序。

[0139] 第三源具有包括四个64比特的数据元素的第三256比特紧缩数据,这四个64比特的数据元素表示针对四轮的SHA2算法(即,当前轮(i)、在当前轮之后的一轮(i+1)、在当前轮之后的两轮(i+2)、以及在当前轮之后的三轮(i+3))的消息输入和常数输入。在所示出的实施例中,表示针对当前轮的消息输入W(i)加上针对当前轮的常数输入K(i)的第一数据元素存储在[63:0]中,以及表示针对在当前轮之后的一轮的消息输入W(i+1)加上针对在当前轮之后的一轮的常数输入K(i+1)的第二数据元素存储在[127:64]中。接着,表示针对在当前轮之后的两轮的消息输入W(i+2)加上针对在当前轮之后的两轮的常数输入K(i+2)的第三数据元素存储在[191:128]中,以及表示针对在当前轮之后的三轮的消息输入W(i+3)加上针对在当前轮之后的三轮的常数输入K(i+3)的第四数据元素存储在[255:192]中。在其它实施例中,可以不同地布置数据。

[0140] SHA2执行逻辑1212可操作以响应于指令而在由该指令指示的第一目的地1218A中存储第一256比特紧缩数据结果,并且响应于指令而在由该指令指示的第二目的地1218B中存储第二256比特紧缩数据。在一些实施例中,第一256比特紧缩数据结果包括四个已更新的状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2},状态数据元素a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}已经通过两轮的SHA2哈希算法而分别从相对应的状态数据元素a_i、b_i、e_i和f_i更新。在所示出的实施例中,a_{i+2}存储在比特[255:192]中,b_{i+2}存储在比特[191:128]中,e_{i+2}存储在比特[127:64]中,和f_{i+2}存储在比特[63:0]中,尽管不要求该特定的顺序。在一些实施例中,第二128比特紧缩数据结

果包括四个已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}，已更新的状态数据元素a_{i+4}、b_{i+4}、e_{i+4}和f_{i+4}已经通过四轮的SHA2哈希算法而分别从相对应的状态数据元素a_i、b_i、e_i和f_i更新。在所示出的实施例中，a_{i+4}存储在比特[255:192]中，b_{i+4}存储在比特[191:128]中，e_{i+4}存储在比特[127:64]中，和f_{i+4}存储在比特[63:0]中，尽管不要求该特定的顺序。

[0141] 在一些实施例中，可以在已经完成两轮之后(例如，大约通过执行该指令的一半)在第一目的地中存储第一128比特紧缩数据结果，并且可以在已经完成四轮之后(例如，大约通过完全执行该指令)在第二目的地中存储第二128比特紧缩数据结果。

[0142] 在第一目的地1218A中存储的两轮之后的状态变量a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}分别等同于四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}。有利地，无需单独计算四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}。相反，在第一目的地中存储的两轮之后的状态变量a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}可以仅用作在四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}。存储在第一目的地1218A中的两轮之后的状态变量a_{i+2}、b_{i+2}、e_{i+2}和f_{i+2}可以与存储在第二目的地1218B中的四轮之后的状态变量c_{i+4}、d_{i+4}、g_{i+4}和h_{i+4}组合，以获得的各自由四轮的SHA2算法更新的一组八个已更新的状态元素。在一些实施例中，第一源可以被重用作第一目的地且第二源可以被重用作第二目的地，尽管这不是要求的。

[0143] 图7至12示出了适合的指令/操作的若干具体实施例。然而，考虑了许多其它实施例，并且这些实施例对本领域的技术人员而言将是显而易见的，并且具有本公开的益处。例如，在这些实施例的每一个中，可以可选地重新布置源和目的地内的数据元素(例如，状态元素)的特定顺序。例如，状态元素可以以任何其它期望的顺序(诸如例如以f、e、b、a的顺序或以b、a、f、e的顺序或以任何其它期望的顺序)来存储，而不是以a、b、e、f的顺序来存储。作为另一示例，针对使用128比特寄存器的具有256比特的SHA2算法描述的实施例中的任何一个还可以用于使用256比特寄存器的具有512比特的SHA2算法。在这些实施例的每一个中，源中的一个或多个可以可选地是隐式的而不是显式的。在这些实施例的每一个中，源可以被用作源/目的地或替代地可以使用单独指定或指示的目的地。此外，虽然已经描述了128比特或256比特存储位置，但是如果期望存储128比特或256比特的紧缩数据，则可以使用更大的寄存器或其它存储位置。

[0144] 如上文所提及的，上述操作针对SHA-512与针对SHA-256是轻微不同的。尽管针对这些算法的操作在本领域是公知的，且完全在安全哈希标准(SHS)(FIPS PUB 180-3)中描述，但是针对SHA-512该组操作为如下所述：

[0145] $\sum_0^{512}(a) = (a \text{ ROTR } 28) \text{ XOR } (a \text{ ROTR } 34) \text{ XOR } (a \text{ ROTR } 39)$

[0146] $\sum_1^{512}(e) = (e \text{ ROTR } 14) \text{ XOR } (e \text{ ROTR } 18) \text{ XOR } (e \text{ ROTR } 41)$

[0147] $Maj(a,b,c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$

[0148] $Ch(e,f,g) = (e \text{ AND } f) \text{ XOR } ((\text{NOT } e) \text{ AND } g)$

[0149] $T_1 = h + \sum_1^{512}(e) + Ch(e,f,g) + K_i^{512} + W_i$

[0150] $T_2 = \sum_0^{512}(a) + Maj(a,b,c)$

[0151] $h = g$

[0152] $g = f$

[0153] $f = e$

[0154] $e = d + T_1$

[0155] $d=c$

[0156] $c=b$

[0157] $b=a$

[0158] $a=T_1+T_2$

[0159] 图13A是适合的指令格式的第一实施例的框图。指令格式包括操作代码或操作码1346A。操作码可以表示可操作以标识要执行的指令和/或操作的多个比特或一个或多个字段。该指令格式还包括第一源/目的地说明符1348A和第二源说明符1350A。通过示例的方式，这些说明符中的每一个可以包括用于指定寄存器的地址、存储器位置、或其它存储位置的比特或一个或多个字段。第一源/目的地说明符用于指定要具有第一源操作数的存储位置且相同的所指定的存储位置还被用作要在其中存储结果的目的地。或者，在另一实施例中，第一源/目的地和/或第二源中的一个或多个可以是对指令隐式的而不是明确指定的。该指令格式仅指定或另外指示两个源。上文针对图7-8示出和描述的指令对于这样的指令格式非常有用。

[0160] 图13B是适当的指令格式的第二实施例的框图。该指令格式包括操作代码或操作码1346B、第一源/目的地说明符1348B、以及第二源说明符1350B。这些中的每一个可以与第一指令格式中的那些类似或相同。该指令格式还包括可选的源说明符1352B，以指定在其中存储了第三源操作数的第三源存储位置。或者，第三源存储位置可以是对指令隐式的而不是明确指定的。该指令格式指定或另外指示三个源。上文针对图9-10示出和描述的指令对于这样的指令格式非常有用。

[0161] 图13C是适合的指令格式的第三实施例的框图。该指令格式包括操作代码或操作码1346C、第一源/目的地说明符1348C、以及第二源/目的地说明符1350C、以及可选的第三源说明符1352C。这些中的每一个可以与第二指令格式中的那些类似或相同，除了第二源说明符也用于目的地之外。该指令格式指定或另外指示三个源和两个目的地。上文针对图11-12示出和描述的指令对于这样的指令格式非常有用。如上文所描述的，在一些实施例中，可以在两轮之后大约通过执行指令的一半时对目的地中的一个进行写入，并且可以在所有四轮之后来对其它目的地进行写入。

[0162] 这些仅仅是几个示例性实施例。要理解的是，在其它实施例中显式说明符中的一个可以相反对指令是隐式的。替代实施例可以包括说明符的子集，可以添加额外的字段，可以重叠某些字段等。所示出的字段的顺序/布置不是必需的，而是可以重新布置这些字段。字段无需包括连续的比特序列而是可以由非连续或分开的比特组成。在一些实施例中，该指令格式可以遵守EVEX编码或指令格式(例如，在针对SHA-512算法的三个源字段和/或指令的情况下)，但是这不是必需的。

[0163] 图14是适当组的紧缩数据或矢量寄存器1454的特定的示例实施例的框图。紧缩数据寄存器包括三十二个标记为ZMM0至ZMM31的512比特紧缩数据寄存器。在所示出的实施例中，这些寄存器中的较低阶的十六个(即ZMM0-ZMM15)的较低阶256比特被化名为相应的256比特紧缩数据寄存器(标记为YMM0-YMM15)或被后者所叠加，尽管这不是必需的。类似地，YMM0-YMM15的较低阶128比特被化名为相应的128比特紧缩数据寄存器(标记为XMM0-XMM15)或被后者所叠加，尽管这也不是必需的。512比特寄存器ZMM0至ZMM31可操作以保留512比特紧缩数据、256比特紧缩数据、和/或128比特紧缩数据。256比特寄存器YMM0-YMM15

可操作以保留256比特紧缩数据和/或128比特紧缩数据。128比特寄存器XMM0-XMM1可操作以保留128比特紧缩数据。支持不同的数据元素大小，包括至少8比特字节数据、16比特字节数据、32比特双字或单精度浮点数据、以及64比特四字或双精度浮点数据。紧缩数据寄存器的替代实施例可以包括不同数目的寄存器、不同大小的寄存器、可以或可以不在较小的寄存器上化名为较大的寄存器，或另外与所示出的那些不同(例如，可以包括两组或更多不同组的寄存器)。

[0164] 此处的描述意在实现SHA2算法并获得与SHA2算法一致的哈希。在本申请中，将导致与由标准所描述的不同的哈希(例如，由于错别字或其它)的任何不一致是非故意的和错误的，且本领域的技术人员将意识到该标准是正确的并替换那些错别字。

[0165] 一些实施例与制品(例如，计算机程序产品)有关，包括存储在本申请中的别处描述的指令中的至少一个的机器可读存储介质。本申请中公开的指令中的任何一个是适合的且可以存储在介质上。

[0166] 在一些实施例中，机器可读存储介质可以是有形的和/或非瞬时性机器可读存储介质。在各个实施例中，机器可读存储介质可以包括软盘、光盘、CD-ROM、磁盘、磁光盘、只读存储器(ROM)、可编程ROM(PROM)、可擦除且可编程ROM(EPROM)、电可擦除且可编程ROM(EEPROM)、随机存取存储器(RAM)、静态RAM(SRAM)、动态RAM(DRAM)、闪速存储器、相变存储器、半导体存储器、或它们的组合。在一些实施例中，该介质可以包括一个或多个固态数据存储材料，诸如例如半导体数据存储材料、相变数据存储材料、磁数据存储材料、光固态数据存储材料等。

[0167] 适合的机器的示例包括但不限于处理器(例如，通用处理器和专用处理器)、指令处理装置、以及具有一个或多个处理器或指令处理器装置的电子设备。适合的电子设备的示例包括但不限于台式计算机、膝上型计算机、笔记本计算机、蜂窝电话、手持或移动计算机、服务器、网络元件、机顶盒、其它类型的计算机系统等。

[0168] 指令集包括一个或多个指令格式。给定的指令格式定义了各种字段(比特数目、比特位置)以指定，除了其它的之外，要执行的操作(操作码)以及关于其要执行操作的操作码。尽管定义了指令模板(或子格式)，但是一些指令格式进一步被打破。例如，可以将给定的指令格式的指令模板定义成具有指令格式字段的不同子集(所包含的字段通常按同样的顺序，但是至少一些具有不同的比特位置，这是因为包含了更少的字段)和/或定义成具有被不同解释的给定字段。从而，使用给定的指令格式来表示ISA的每一个指令(且，如果定义了，则以该指令格式的指令模板中的给定的一个)，并且ISA的每一个指令包括用于指定操作和操作数的字段。例如，示例性ADD指令具有特定的操作码和指令格式，该指令格式包括用于指定操作码的操作码字段和用于选择操作数的操作数字段(源1/目的地和源2)；并且该ADD指令在指令流中的出现将在选择特定操作数的操作数字段中具有特定的内容。被称为高级矢量扩展(AVX)(AVX1和AVX2)且使用矢量扩展(VEX)编码方案的一组SIMD扩展已经被发布和/或出版(例如，参见Intel®64和IA-32架构软件开发商手册，2011年10月；并参见Intel®64高级矢量扩展编程参考，2011年6月)。

[0169] 本申请中描述的指令的实施例可以以不同的格式实施。此外，下文详细描述了示例性系统、架构、和管线。可以在这些系统、架构和管线上执行指令的实施例，但是不限于那些详细描述的。

[0170] VEX指令格式

[0171] VEX编码允许指令具有超过两个操作数，且允许SIMD矢量寄存器长于158比特。使用VEX前缀提供了三个操作数(或更多)语法。例如，先前的两个操作数指令执行诸如A＝A+B的操作，该操作覆盖源操作数。使用VEX前缀使得操作数能够执行诸如A＝B+C的非破坏性操作。

[0172] 图15A示出了示例性AVX指令格式，包括VEX前缀1502、真实操作码(real opcode)字段1530、Mod R/M字节1540、SIB字节1550、位移字段1562、以及IMM8 1572。图15B示出了来自图15A的哪些字段组成了完整的操作码字段1574和基础操作字段1542。图15C示出了来自图15A的哪些字段组成了寄存器索引字段1544。

[0173] VEX前缀(字节0-2)1502以三字节形式编码。第一字节是格式字段1540(VEX字节0，比特[7:0])，其包含显式C4字节值(用于区分C4指令格式的特殊值)。第二个三字节(VEX字节1-2)包括提供特定能力的多个比特字段。具体地，REX字段1505(VEX字节1，比特[7-5])由VEX.R比特字段(VEX字节1，比特[7]-R)、VEX.X比特字段(VEX字节1，比特[6]-X)、以及VEX.B比特字段(VEX字节1，比特[5]-B)组成。指令的其它字段将寄存器索引的较低的三个比特编码为本领域已知的(rrr、xxx、和bbb)，使得Rrrr、Xxxx和Bbbb可以通过将VEX.R、VEX.X、以及VEX.B相加形成。操作码映射字段1518(VEX字节1，比特[4:0]-mmmmm)包括用于对隐含的前导操作码字节进行编码的内容。W字段1564(VEX字节2，比特[7]-W)——由符号VEX.W表示，并根据指令提供不同功能。VEX.vvvv1523的作用(VEX字节2，比特[6:3]-vvvv)可以包括：1)VEX.vvvv对第一源寄存器操作数编码、以反转的(1的补码)的形式来指定并且针对具有2个或更多个源操作数的指令是有效的；2)EVEX.vvvv对目的地寄存器操作数编码、针对某些矢量移位而以1的补码的形式来指定；或3)EVEX.vvvv不编码任何操作数，字段被保留并且应包含1111b。如果VEX.L 1568大小字段(VEX字节2，比特[2]-L)＝0，其指示158比特矢量；如果VEX.L＝1，则其指示256比特矢量。前缀编码字段1525(VEX字节2，比特[1:0]-pp)为基站字段提供了额外的比特。

[0174] 真实操作码字段1530(字节3)也称为操作码字节。在该字段中指定操作码的一部分。

[0175] MOD R/M字段1540(字节4)包括MOD字段1542(比特[7-6])、Reg字段1544(比特[5-3])、以及R/M字段1546(比特[2-0])。Reg字段1544的作用可以包括以下内容：对目的地寄存器操作数或源寄存器操作数进行编码(rrr或Rrrr)，或被当作操作码扩展且不用于对任何指令操作数进行编码。R/M字段1546的作用可以包括以下内容：对引用存储器地址的指令操作数进行编码，或对目的地寄存器操作数或源寄存器操作数进行编码。

[0176] 缩放、索引、基础(SIB)——缩放字段1550(字节5)的内容包括SS1552(比特[7-6])，其用于生成存储器地址。SIB.xxx 1554(比特[5-3])和SIB.bbb1556(比特[2-0])的内容先前已关于寄存器索引Xxxx和Bbbb而被引用。

[0177] 位移字段1562和立即数字段(IMM8)1572包含地址数据。

[0178] 示例性寄存器架构

[0179] 图16是根据本发明一实施例的寄存器架构1600的框图。在所示出的实施例中，存在32个515比特宽的矢量寄存器1610；这些寄存器被称为zmm0至zmm31。较低的19个zmm寄存器的较低阶256比特被寄存器ymm0-19叠加。较低的19个zmm寄存器的较低阶158比特(ymm寄

存器的较低阶158比特)被寄存器xmm0-18叠加。

[0180] 写掩码寄存器1618——在所示出的实施例中,存在8个写掩码寄存器(k0至k7),各自为64比特大小。在替代实施例中,写掩码寄存器1618是19比特大小。如先前所描述的,在本发明的一实施例中,矢量掩码寄存器k0不能被用作写掩码;当进行通常将会指示k0用于写掩码的编码时,其选择0xFFFF的硬接线掩码,针对该指令有效地禁用写掩码。

[0181] 通用寄存器1625——在所示出的实施例中,存在十六个64比特通用寄存器,其与现有的x86寻址模式一起使用以寻址存储器操作数。通过名称RAX、RBX、RCX、RDX、RBP、RSI、RDI、RSP、以及R8至R18来引用这些寄存器。

[0182] 标量浮点栈寄存器堆(x87栈)1645,在其上化名为MMX紧缩整型平直寄存器堆1650——在所示出的实施例中,x87栈是用于使用x87指令集扩展对32/64/80比特浮点数据执行标量浮点操作的八元素栈;虽然MMX寄存器用于对64比特紧缩整型数据执行操作,以及用于为在MMX与XMM寄存器之间执行的一些操作保留操作数。

[0183] 本发明的替代实施例可以使用更宽或更窄的寄存器。此外,本发明的替代实施例可以使用更多、更少、或不同的寄存器堆或寄存器。

[0184] 示例性内核架构、处理器和计算机架构

[0185] 为了不同的目的,且在不同的处理器中,可以以不同的方式来实现处理器内核。例如,这样的内核的实现可以包括:1)意在进行通用计算的通用有序内核;2)意在进行通用计算的高性能通用乱序内核;3)主要意在进行图形和/或科学(吞吐量)计算的专用内核。不同的处理器的实现可以包括:1)CPU,包含意在进行通用计算的一个或多个通用有序内核和/或意在进行通用计算的一个或多个通用乱序内核;以及2)协处理器,包含主要意在进行图形和/或科学(吞吐量)的一个或多个专用内核。这样的不同的处理器导致不同的计算机系统架构,其可以包括:1)在与CPU分离的芯片上的协处理器;2)在与CPU相同的封装中的分离的管芯上的协处理器;3)在与CPU相同的管芯上的协处理器(在该情况下,这样的协处理器有时被称为专用逻辑,例如集成图形和/或科学(吞吐量)逻辑,或被称为专用内核);以及4)片上系统,其可以包括所描述的在相同管芯上的CPU(有时被称为应用内核或应用处理器)、上文描述的协处理器、以及额外的功能。接下来描述示例性内核架构,然后是对示例性处理器和计算机架构的描述。示例性内核架构

[0186] 有序和乱序内核框图

[0187] 图17A是根据本发明实施例的示出了示例性有序管线和示例性寄存器重命名、乱序发布/执行管线的框图。图17B是根据本发明实施例的示出了要在处理器中包括的有序架构内核和示例性寄存器重命名、乱序发布/执行架构的示例性实施例。图17A-B中的实线框示出了有序管线和有序内核,而可选增加的虚线框示出了寄存器重命名、乱序发布/执行管线和内核。考虑到有序方面是乱序方面的子集,将描述乱序方面。

[0188] 在图17A中,处理器管线1700包括提取阶段1702、长度解码阶段1704、解码阶段1706、分配阶段1708、重命名阶段1710、调度(也被称为分发或发布)阶段1715、寄存器读取/存储器读取阶段1717、执行阶段1719、写回/存储器写阶段1721、异常处理阶段1722、以及提交阶段1724。

[0189] 图17B示出了处理器内核1790,其包括耦合到执行引擎单元1750的前端单元1730,且这两者都耦合到存储器单元1770。内核1790可以是精简指令集计算(RISC)内核、复杂指

令集计算(CISC)内核、超长指令字(VLIW)内核、或混合或替代内核类型。作为另一选项，内核1790可以是专用内核，诸如例如网络或通信内核、压缩引擎、协处理器内核、通用计算图形处理单元(GPGPU)内核、图形内核等。

[0190] 前端单元1730包括耦合到指令高速缓存单元1734的分支预测单元1732，指令高速缓存单元1734耦合到指令转换后备缓冲器(TLB)1736，指令转换后备缓冲器(TLB)1736耦合到指令提取单元1738、指令提取单元1738耦合到解码单元1740。解码单元1740(或解码器)可以解码指令，并生成从原始指令解码的、反映原始指令的、或从原始指令导出的一个或多个微操作、微代码入口点、微指令、其它指令、或其它控制信号而作为输出。可以使用各种不同的机制来实现解码单元1740。适合的机制的示例包括但不限于查询表、硬件实现、可编程逻辑阵列(PLA)、微代码只读存储器(ROM)等。在一个实施例中，内核1790包括微代码ROM或存储针对某些宏指令的微代码的其它介质(例如，在解码单元1740中或在前端单元1730内)。解码单元1740耦合到执行引擎单元1750中的重命名/分配器单元1752中。

[0191] 执行引擎单元1750包括耦合到退出单元1754和一组一个或多个调度器单元1756的重命名/分配器单元1752。调度器单元1756表示任何数目的不同的调度器，包括保留站、中央指令窗等。调度器单元1756耦合到物理寄存器堆单元1758。物理寄存器堆单元1758中的每一个表示一个或多个物理寄存器堆，其中不同的物理寄存器堆存储一个或多个不同的数据类型，例如标量整型、标量浮点型、紧缩整型、紧缩浮点型、矢量整型、矢量浮点型、状态(例如，指令指针，其是要被执行的下一指令的地址)等。在一实施例中，物理寄存器堆单元1758包括矢量寄存器堆、写掩码寄存器堆、以及标量寄存器单元。这些寄存器单元可以提供架构矢量寄存器、矢量掩码寄存器、以及通用寄存器。物理寄存器堆单元1758被退出单元1754重叠以示出在其中可以实现寄存器重命名和乱序执行的各种方式(例如，使用重排序缓冲器以及退出寄存器堆；使用未来文件(future file)、历史缓冲器、退出寄存器堆；使用寄存器映射和寄存器池；等)。退出单元1754和物理寄存器堆单元1758耦合到执行簇1760。执行簇1760包括一组一个或多个执行单元1762以及一组一个或多个存储器访问单元1764。执行单元1762可以执行各种操作(例如，移位、加法、减法、乘法)并对各种类型的数据(例如，标量浮点型、紧缩整型、紧缩浮点型、矢量整型、矢量浮点型)执行。虽然一些实施例可以包括专用于特定功能或功能组的多个执行单元，但是其它实施例可以仅包括执行所有功能的一个执行单元或全部多个执行单元。调度器单元1756、物理寄存器堆单元1758、以及执行簇1760被示出为是可能的复数，这是因为某些实施例针对某些类型的数据/操作(例如，各种具有其各自的调度单元、物理寄存器堆单元、和/或执行簇的标量整型管线、标量浮点/紧缩整型/紧缩浮点/矢量整型/矢量浮点管线、和/或存储器访问管线——且在分离的存储器访问管线的情况下，实现了某些实施例，其中仅该管线的执行簇具有存储器访问单元1764)创建分离的管线。还应该理解的是，在使用分离的管线的地方，这些管线中的一个或多个可以是乱序发布/执行且其余的是有序的。

[0192] 该组存储器访问单元1764耦合到存储器单元1770，存储器单元1770包括耦合到数据高速缓存单元1774的数据TLB单元1772，数据高速缓存单元1774耦合到级别2(L2)高速缓存单元1776。在一个示例性的实施例中，存储器访问单元1764可以包括加载单元、存储地址单元、以及存储数据单元，其中的每一个耦合到存储器单元1770中的数据TLB单元1772。指令高速缓存单元1734进一步耦合到存储器单元1770中的级别2(L2)高速缓存单元1776。L2

高速缓存单元1776耦合到一个或多个其它级别的高速缓存并最终耦合到主存储器。

[0193] 通过示例的方式，示例性的寄存器重命名、乱序发布/执行内核架构可以实现管线1700，如下所述：1)指令提取1738执行获取和长度解码阶段1702和1704；2)解码单元1740执行解码阶段1706；3)重命名/分配器单元1752执行分配阶段1708和重命名阶段1710；4)调度器单元1756执行调度阶段1715；5)物理寄存器堆单元1758和存储器单元1770执行寄存器读取/存储器读取阶段1717；执行簇1760进行执行阶段1719；6)存储器单元1770和物理寄存器堆单元1758执行写回/存储器写阶段1721；7)可以在异常处理阶段1722中涉及的各个单元；以及8)退出单元1754和物理寄存器堆单元1758执行提交阶段1724。

[0194] 内核1790可以支持一个或多个指令集(例如，x86指令集(具有已添加了较新的版本的一些扩展)；加州Sunnyvale的MIPS科技的MIPS指令集；加州Sunnyvale的ARM Holdings的ARM指令集(具有诸如NEON的可选的额外扩展)，包括本申请中描述的指令集。在一个实施例中，内核1790包括用于支持紧缩数据指令集扩展(例如，AVX1、AVX2)的逻辑，从而允许使用紧缩数据来执行由许多多媒体应用所使用的操作。

[0195] 应该理解的是，内核可以支持多线程(执行并行的两组或更多组操作或线程)，且可以以多种方式来这样做，包括分时多线程、同步多线程(其中，单个物理内核为物理内核是同步多线程的线程中的每一个提供了逻辑内核)，或其组合(例如，分时提取和解码以及其后的同步多线程，例如在Intel®超线程技术中的)。

[0196] 虽然在乱序执行的上下文中描述了寄存器重命名，但是应该理解的是，寄存器重命名可以在有序架构中使用。虽然所示出的处理器的实施例还包括分离的指令和数据高速缓存单元1734/1744和共享L2高速缓存单元1776，但是替代实施例可以具有针对指令和数据单个内部高速缓存，诸如例如级别1(L1)内部高速缓存、或多个级别的内部缓存。在一些实施例中，系统可以包括包括内部高速缓存和在内核和/或处理器之外的外部高速缓存的组合。或者，所有的高速缓存可以在内核和/或处理器之外。

[0197] 特定的示例性有序内核架构

[0198] 图18A-B示出了更具体的示例性有序内核架构的框图，其内核将是芯片中若干逻辑块中的一个(包括相同类型和/或不同类型的其它内核)。根据应用，上述逻辑块通过高带宽互连网络(例如，环形网)来与一些固定功能逻辑、存储器I/O接口、以及其它必要的I/O逻辑通信。

[0199] 图18A是根据本发明实施例的单个处理器内核，连同其到管芯上互连网络1802的连接以及其与级别2(L2)高速缓存1804的本地子集的连接的框图。在一个实施例中，指令解码器1800支持具有紧缩数据指令集扩展的x86指令集。L1高速缓存1806允许标量单元和矢量单元对高速缓冲存储器的低时延访问。虽然在一实施例中(为了简化设计)，标量单元1808和矢量单元1810使用分离的寄存器组(分别为标量寄存器1815和矢量寄存器1817)，且在它们之间传送的数据被写入存储器，并且接着从级别1(L1)高速缓存1806读回，但是本发明的替代实施例可以使用不同的方式(例如，使用单个寄存器组或包括允许在两个寄存器堆之间传送数据而不需要写或读回的的通信路径)。

[0200] L2高速缓存1804的本地子集是被划分成分离的本地子集的全局L2高速缓存的一部分，每个处理器内核一个。每一个处理器内核具有到L2高速缓存1804的其自身的本地子集的直接访问路径。由处理器内核读取的数据存储在其L2高速缓存子集1804中，且其可以

迅速地、与访问它们各自的本地L2高速缓存子集的其它处理器内核并行地被访问。由处理器内核写的数据被存储在其自身的L2高速缓存子集1804中，并且如果需要的话，从其它子集清除。环形网确保了所共享数据的一致性。环形网是双向的，以允许诸如处理器内核、L2高速缓存以及其它逻辑块指令的代理在芯片内彼此通信。每一环形数据路径在每方向上为1015比特宽。

[0201] 图18B是根据本发明实施例的在图18A中的处理器内核的部分的扩展图。图18B包括L1数据高速缓存1806A、L1高速缓存1804的部分、以及关于矢量单元1810和矢量寄存器1817的更多细节。具体地，矢量单元1810是19宽矢量处理器单元(VPU)(参见19宽ALU 1828)，其执行整型、单精度浮点型、以及双精度浮点型指令中的一个或多个。VPU支持利用搅拌单元1823搅拌寄存器输入，利用数字转换单元1822A-B进行的数字转换，以及利用复制单元1824对存储器输入进行的复制。写掩码寄存器1826允许预测所得到的矢量写入。

[0202] 具有集成存储器控制器和显卡的处理器

[0203] 图19是根据本发明实施例的处理器1900的框图，处理器1900可以具有超过一个内核，可以具有集成存储器控制器，且可以具有集成显卡。图19中的实线框示出了具有单个内核1902A、系统代理1910、一组一个或多个总线控制器单元1916的处理器1900，而可选添加的虚线框示出了具有多个内核1902A-N、在系统代理单元1910中的一组一个或多个集成存储器控制器单元1914、以及专用逻辑1908的处理器1900。

[0204] 因此，处理器1900的不同实现可以包括：1)CPU，其具有专用逻辑1908，该专用逻辑1908是集成图形和/或科学(吞吐量)逻辑(其可以包括一个或多个内核)，且上述内核1902A-N是一个或多个通用内核(例如，通用有序内核、通用乱序内核、这两者的组合)；2)协处理器，其具有内核1902A-N，内核1902A-N是主要用于图形和/或科学(吞吐量)的大量的专用内核；以及3)协处理器，其具有内核1902A-N，内核1902A-N是大量的通用有序内核。因此，处理器1900可以是通用处理器、协处理器或专用处理器，诸如例如网络或通信处理器、压缩引擎、图形处理器、GPGPU(通用图形处理器单元)、高吞吐量许多集成内核(many integrated core,MIC)协处理器(包括30或更多内核)、嵌入式处理器等。可以在一个或多个芯片上实现处理器。处理器1900可以是使用诸如例如BiCMOS、CMOS、或NMOS的多个处理技术中的任何一个的一个或多个基底的一部分和/或其可以在上述一个或多个基底上实现。

[0205] 存储器分级体系包括内核中的一个或多个级别的高速缓存、一组或一个或多个共享的高速缓存单元1906、以及耦合到该组集成存储器控制器单元1914的外部存储器(未示出)。该组共享的高速缓存单元1906可以包括一个或多个中级高速缓存，例如级别2(L2)、级别3(L3)、级别4(L4)、或其它级别高速缓存、最后一级高速缓存(LLC)、和/或其组合。虽然在一个实施例中，基于环形的互连单元1912将集成图形逻辑1908、该组共享的高速缓存单元1906、以及系统代理单元1910/集成存储器控制器单元1914互连，但是替代实施例可以使用任何数量的公知的技术来互连这些单元。在一个实施例中，可以维护一个或多个高速缓存单元1906与内核1902A-N之间的一致性。

[0206] 在一些实施例中，内核1902A-N中的一个或多个能够进行多线程处理。系统代理1910包括协调和操作内核1902A-N的那些组件。系统代理单元1910可以包括例如功率控制单元(PCU)和显示单元。PCU可以是或包括调节内核1902A-N和集成的图形逻辑1908的功率状态所需的逻辑和组件。显示单元用于驱动一个或多个外部连接的显示器。

[0207] 在架构指令集方面，内核1902A-N可以是同构或异构的；即，内核1902A-N中的两个或更多个可以能够执行相同的指令集，而其它的仅能够执行该指令集的子集或不同的指令集。

[0208] 示例性的计算机架构

[0209] 图20-23是示例性计算机架构的框图。本领域中已知的用于膝上型计算机、台式机、手持PC、个人数字助理、工程工作站、服务器、网络设备、网络集线器、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放机、手持设备、和各种其它电子设备的其它系统设计和配置也是适用的。在一般情况下，如本申请中所公开的，能够结合处理器和/或其它执行逻辑的各种各样的系统或电子设备一般是适用的。

[0210] 现在参考图20，所示的是根据本发明一个实施例的系统2000的框图。系统2000可以包括一个或多个处理器2010、2018，一个或多个处理器2010、2018耦合到控制器集线器2023。在一个实施例中，控制器集线器2023包括图形存储器控制器集线器(GMCH)2090和输入/输出集线器(IOH)2050(其可以在分离的芯片上)；GMCH 2090包括存储器和图形控制器，存储器1340和协处理器2045耦合到GMCH 2090；IOH 2050将输入/输出(I/O)设备2060耦合到GMCH 2090。或者，存储器和图形控制器中的一个或两个集成在处理器之内(如本申请中所描述的)，存储器2040和协处理器2045直接耦合到处理器2010，并且控制器集线器2023在单个芯片中与IOH 2050耦合。

[0211] 在图20中利用虚线表示了额外的处理器2018的可选性质。每个处理器2010、2018可以包括本申请中描述的处理内核中的一个或多个，且其可以是处理器1900的某个版本。

[0212] 例如，存储器2040可以是动态随机存取存储器(DRAM)、相变存储器(PCM)、或这两者的组合。对于至少一个实施例，控制器集线器2023经由诸如前端总线(FSB)的多点分支总线、诸如快速路径互连(QP)指令的点对点接口或类似的连接2095而与处理器2010、2018通信。

[0213] 在一个实施例中，协处理器2045是专用处理器，诸如例如高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。在一实施例中，控制器集线器2023可以包括集成图形加速器。

[0214] 在一系列的包括架构、微架构、热、功率消耗特性等的指标度量方面，物理资源2010、2018之间可能存在各种不同。

[0215] 在一实施例中，处理器2010执行控制通用类型的数据处理操作的指令。在指令中嵌入的可以是协处理器指令。处理器2010将这些协处理器指令识别为应该由附接的协处理器2045执行的类型。由此，处理器2010在协处理器总线或其它互连上向协处理器2045发布这些协处理器指令(或代表协处理器指令的控制信号)。协处理器2045接受并执行所接收的协处理器指令。

[0216] 现在参考图21，所示的是根据本发明一个实施例的第一更具体的示例性系统2100的框图。如在图21中所示出的，多处理器系统2100是点对点互连系统，并且包括经由点对点互连2150耦合的第一处理器2170和第二处理器2180。处理器2170和2180中的每一个可以是处理器1900的某个版本。在本发明的一个实施例中，处理器2170和2180分别是处理器2010和2018，而协处理器2138是协处理器2045。在另一实施例中，处理器2170和2180分别是处理

器2010、协处理器2045。

[0217] 处理器2170和2180被示出为分别包括集成存储器控制器(IMC)单元2172和2182。处理器2170也包含了作为其总线控制器单元一部分的点对点(P-P)接口2176和2178;类似地,第二处理器2180包括P-P接口2186和2188。处理器2170、2180可以使用P-P接口电路2178、2188经由点对点(P-P)接口2150交换信息。如在图21中所示出的,IMC 2172和2182将处理器耦合到相应的存储器,即存储器2132和存储器2134,这些相应的存储器可以是在本地附接到相应的处理器的主存储器的部分。

[0218] 处理器2170、2180各自可以使用点对点接口电路2176、2194、2186、2198经由个体P-P接口2152、2154与芯片组2190交换信息。芯片组2190可以可选地经由高性能接口2139与协处理器2138交换信息。在一实施例中,协处理器2138是专用处理器,诸如例如高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。

[0219] 共享的高速缓存(未示出)可以被包括在处理器中或在两个处理器之外,而仍经由P-P互连与处理器连接,使得如果处理器被置于低功率模式,则任何一个处理器或两个处理器的本地高速缓存信息可以被存储在共享的高速缓存中。

[0220] 芯片组2190可以经由接口2196耦合到第一总线2119。在一个实施例中,第一总线2119可以是外围组件互连(PCI)总线,或诸如PCI快速总线或另一第三代I/O互连总线的总线,结构本发明的范围不限于此。

[0221] 如在图21中所示出的,可以将各种I/O设备2117与总线桥2121一起耦合到第一总线2119,其中总线桥2121将第一总线2119耦合到第二总线2123。在一个实施例中,诸如协处理器、高吞吐量MIC处理器、GPGPU的、加速器(诸如例如图形加速器或数字信号处理器(DSP)单元)的一个或多个额外的处理器2118、现场可编程门阵列、或任何其它的处理器耦合到第一总线2119。在一个实施例中,第二总线2123可以是低引脚数(LPC)总线。在一个实施例中,各种设备可以耦合到第二总线2123,包括例如键盘和/或鼠标2122、通信设备2127和诸如磁盘驱动器或可以包含指令/代码和数据2130的其它大容量存储设备的存储单元2128。此外,音频I/O 2124可以耦合到第二总线2123。要注意的是,其它架构是可能的。例如,系统可以实现多点分支总线或其它这种架构,而不是图21中的点对点架构。

[0222] 现在参考图22,所示出的是根据本发明一个实施例的第二更具体的示例性系统2200的框图。图21和22中类似的元件具有类似的附图标记,且已经从图22中省略了图21中的某些方面,以避免模糊图22中的其它方面。

[0223] 图22示出了处理器2170、2180可以分别包括集成存储器和I/O控制逻辑("CL")2172和2182。因此,CL 2172、2182包括集成存储器控制器单元并包括I/O控制逻辑。图22示出了不仅存储器2132、2134耦合到CL2172、2182,而且I/O设备2217也耦合到控制逻辑2172、2182。遗留I/O设备2218耦合到芯片组2190。

[0224] 现在参照图23,所示的是根据本发明一个实施例的SoC 2300的框图。图19中的类似元件具有类似的附图标记。此外,虚线框是在更高级的SoC上的可选特征。在图23中,互连单元2302耦合到:应用处理器2310,其包括一组一个或多个内核232A-N以及共享的高速缓存单元1906;系统代理单元1910;总线控制器单元1919;集成存储器控制器单元1917;一组或一个或多个协处理器2323,其可以包括集成图形逻辑、图像处理器、音频处理器、以及视频处理器;静态随机存取存储器(SRAM)单元2330;直接存储器存取(DMA)单元2332;以及用

于耦合到一个或多个外部显示器的显示单元2340。在一个实施例中，协处理器2323包括专用处理器，诸如例如网络或通信处理器、压缩引擎、GPGPU、高吞吐量MIC处理器、嵌入式处理器等。

[0225] 本申请中公开的机制的实施例可以以硬件、软件、固件或者这样的实现方式的组合实现。本发明的实施例可以被实现为在包括至少一个处理器、存储系统(包括易失性和非易失性存储器和/或存储元件)、至少一个输入设备、以及至少一个输出设备的可编程系统上执行的计算机程序或程序代码。

[0226] 诸如图21中示出的代码2130的程序代码可以被应用于输入指令，以执行本申请中描述的功能并生成输出信息。输出信息可以以已知的方式而应用于一个或多个输出设备。为了本申请的目的，处理器系统包括具有诸如例如数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)、或微处理器的处理器的任何系统。

[0227] 可以以高级面向过程或面向对象编程语言来实现程序代码，以与处理系统通信。如果期望的话，则程序代码还可以以汇编或机器语言来实现。实际上，本申请中描述的机制并不受限于任何特定的编程语言的范围。在任何情况下，语言可以是编译或解释语言。

[0228] 至少一个实施例的一个或多个方面可以由在机器可读介质上存储的代表性指令来实现，该代表性指令代表了处理器内的各个逻辑，当由机器读取其时，使得该机器制造用于执行本申请中描述的技术的逻辑。这样的代表，被称为“IP内核”，其可以存储在有形的、机器可读介质上，并提供给各个用户或生产设施，以加载到实际制造逻辑或处理器的制造机器中。

[0229] 这样的机器可读存储介质可以包括但不限于由机器或设备制造或形成的非瞬时性、有形的制品布置，包括存储介质，例如硬盘、包括软盘、光盘(压缩盘只读存储器(CD-ROM)、压缩盘可重写(CD-RW))、以及磁光盘的其它类型的磁盘、半导体设备例如只读存储器(ROM)、诸如动态随机存取存储器(DRAM)、静态随机存取存储器(SRAM)的随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、闪速存储器、电可擦除可编程只读存储器(EEPROM)、磁或光盘、或适于存储电子指令的任何其它类型的媒体。

[0230] 由此，本发明的实施例还包括非瞬时性有形机器可读介质，其包含矢量友好指令格式的指令或包含设计数据，例如硬件描述语言(HDL)，其定义了本申请中描述的结构、电路、装置、处理器和/或系统特征。这样的实施例还可以被称为程序产品。

[0231] 仿真(包括二进制转换、代码融合等)

[0232] 在一些情况下，指令转换器可以用于将指令从源指令集转换到目标指令集。例如，指令转换器可以转化(例如，使用静态二进制转化、或包含动态编译的动态二进制转化)、变型、仿真、或另外将指令转换成要由内核处理的一个或多个其它指令。指令转换器可以以软件、硬件、固件或其组合实现。指令转换器可以在处理器上、在处理器外、或部分在处理器上且部分在处理器外。

[0233] 图24是根据本发明实施例的对比了使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。在所示出的实施例中，指令转换器是软件指令转换器，尽管可替代地，指令转换器可以以软件、固件、硬件、或其各种组合来实现。图24示出了高级语言2402的、可以使用x86编译器2404来编译以生成x86二进制编码2406的程序，x86二进制编码2406可以在本地由具有至少一个x86指令集内核2419的处理器来执行。

具有至少一个x86指令集内核2419的处理器代表可以通过兼容地执行或处理如下操作来执行与具有至少一个x86指令集内核的Intel处理器基本相同的功能：(1)Intel x86指令集内核的指令集的很大一部分或(2)目标在于在具有至少一个x86指令集内核的Intel处理器上运行的应用程序或其它软件的目标代码版本，以便实现与具有至少一个x86指令集内核的Intel处理器基本相同的结果。x86编译器2404表示可操作以生成x86二进制代码2406(例如，目标代码)的编译器，在有或没有额外的联动处理的情况下，x86二进制代码2406可以在具有至少一个x86指令集内核2419的处理器上被执行。类似地，图24示出了高级语言2402的程序可以使用替代的指令集编译器2408来编译，以生成替代的指令集二进制代码2410，替代的指令集二进制代码2410可以由不具有至少一个x86指令集内核2417的处理器(例如，具有执行加州Sunnyvale的MIPS科技的MIPS指令集和/或执行加州Sunnyvale的ARM Holdings的ARM指令集的内核的处理器)在本地执行。指令转换器2415用于将x86二进制码2406转换成可以由不具有x86、指令集内核2417的处理器在本地执行的代码。转换后的代码不太可能与替代的指令集二进制编码2410相同，这是因为难以制作能够这样做的指令转换器；然而，转换后的代码将完成通用操作并由来自替代指令集的指令组成。从而，指令转换器2415表示这样的软件、固件、硬件、或其组合：通过仿真、模拟或任何其它过程而允许不具有x86指令集处理器或内核的处理器或其它电子设备执行x86二进制代码2406。

[0234] 在说明书和权利要求书中，可能已经使用了术语“逻辑”。如在本申请中所使用的，术语逻辑可以包括但不限于硬件、固件、软件、或其组合。逻辑的示例包括集成电路、专用集成电路、模拟电路、数字典里、编程的逻辑设备、包括指令的存储器设备等。在一些实施例中，逻辑可以包括潜在与其它电路组件一起的晶体管和/或门。

[0235] 在说明书和权利要求书中，可能已经使用了术语“耦合”和“连接”，及其衍生词。应该理解的是，这些术语并不意在作为针对彼此的同义词。相反，在特定的实施例中，可以使用“连接”来指示彼此直接物理或电接触的两个或更多个元素。“耦合”可以指两个或更多元素直接物理或电接触。但是，“耦合”还可以指两个或更多个彼此不直接接触，但仍然合作或彼此交互的元件。

[0236] 可能已经使用了术语“和/或”。如在本申请中使用的，术语“和/或”指一个或另一个或两者(例如，A和/或B指A或B或A和B两者)。

[0237] 在上文的描述中，为了解释的目的，已经阐述了许多特定的细节以便提供对本发明实施例的透彻理解。然而，对于本领域的技术人员而言显而易见的是，可以在没有这些具体细节中的一些的情况下实践一个或多个其它实施例。所描述的特定实施例不提供用于限制本发明而是对其进行示出。本发明的范围不由上文提供的具体示例来确定而是仅由下文的权利要求书来确定。与那些在附图中示出的以及在说明书中描述的所有等同的关系都包含在本发明的实施例中。在其它实例中，公知的电路、结构、设备、和操作都已经以框图的形式示出或没有细节以避免模糊对本描述的理解。

[0238] 在认为合适的情况下，在附图中重复了附图标记或附图标记的端部以指示相应的或相似的元件，其可以可选地具有类似的或相同的特征，除非指定或非常明显的之外。在一些情况下，在已经示出或描述了多个组件的情况下，它们可以被并入到单个组件中。在其它情况下，当单个组件已经被示出或描述的情况下，可以将其分成两个或更多个组件。在附图中，箭头表示耦合而双向箭头表示双向耦合。

[0239] 已经描述了各种操作和方法。已经在流程图中以相对基本的形式描述了方法中的一些，但是操作可以可选地被增加到方法中和/或从方法移除。此外，虽然流程图根据示例性的实施例示出了操作的特定顺序，但是应该理解的是，特定的顺序是示例性的。替代实施例可以可选地以不同的顺序执行操作，组合某些操作，叠加某些操作等。可以对方法进行许多修改和调整且这些修改和调整是预期的。

[0240] 还应该理解的是，例如，贯穿本说明书提及的“一个实施例”、“一实施例”、或“一个或多个实施例”指的是特定特征可以被包含在对本发明的实践中。类似地，应该理解的是，在本说明书中，为了使本公开成流线型且有助于理解各个发明的方面的目的，有时在单个实施例、图或其描述中将各个特征组合在一起。然而，本公开的方法并不是要被解释为反映本发明需要比在各权利要求中明确记载的特征更多的特征的意图。相反，如以下权利要求书所反映的，发明的方面可以在于比所公开的单个实施例的所有特征更少的特征。因此，将具体实施例方式之后的权利要求书特此明确地并入到具体实施方式中，其中每个权利要求独立地作为本发明的单独的实施例。

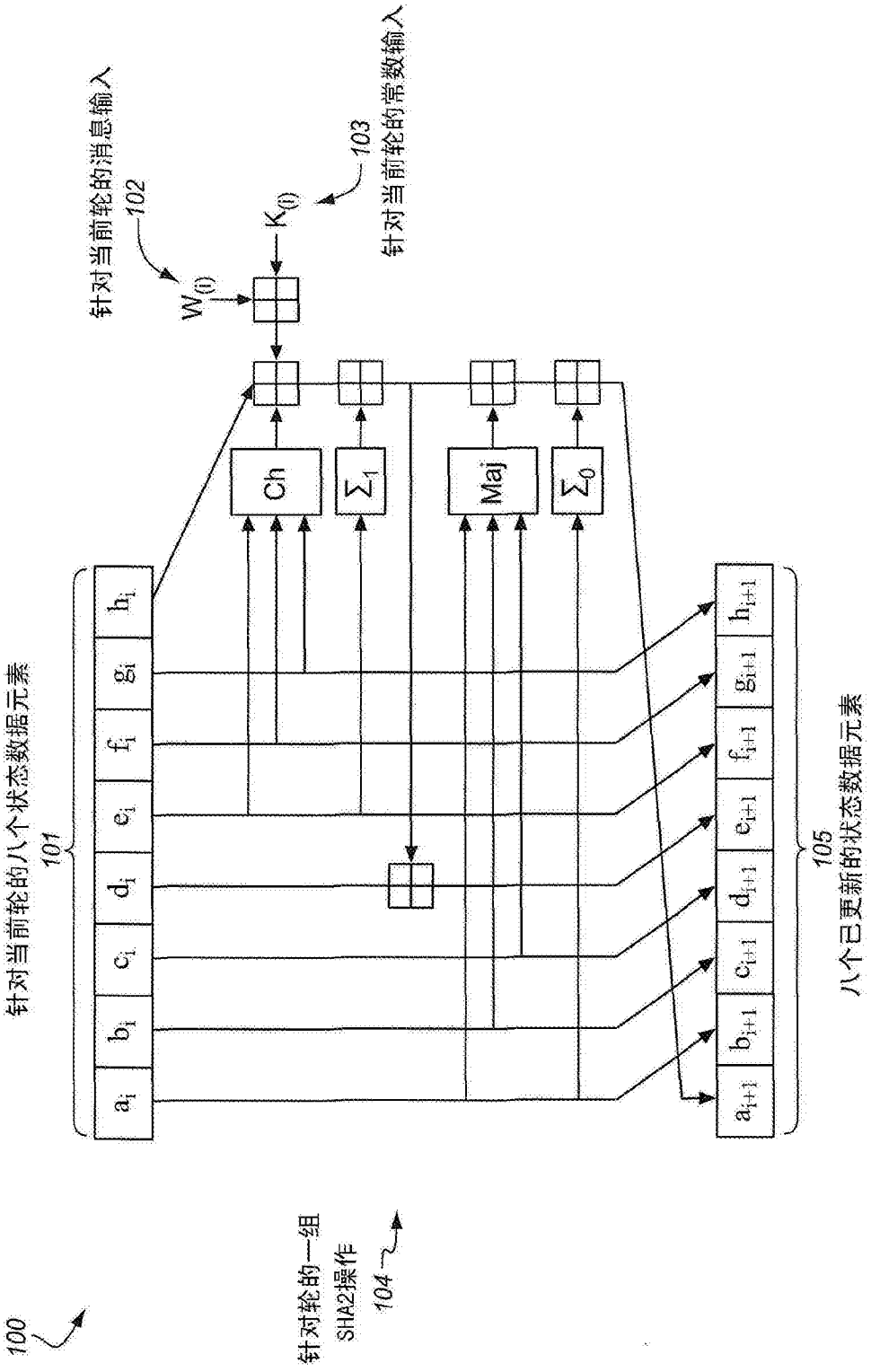

针对当前轮的消息输入
102
$W_{(i)}$
$K_{(i)}$
103
针对当前轮的常数输入
Ch
Σ_1
Maj
Σ_0
针对当前轮的八个状态数据元素
101
a_i b_i c_i d_i e_i f_i g_i h_i
a_{i+1} b_{i+1} c_{i+1} d_{i+1} e_{i+1} f_{i+1} g_{i+1} h_{i+1}
105
八个已更新的状态数据元素
针对轮的一组SHA2操作
104
100

图1现有技术

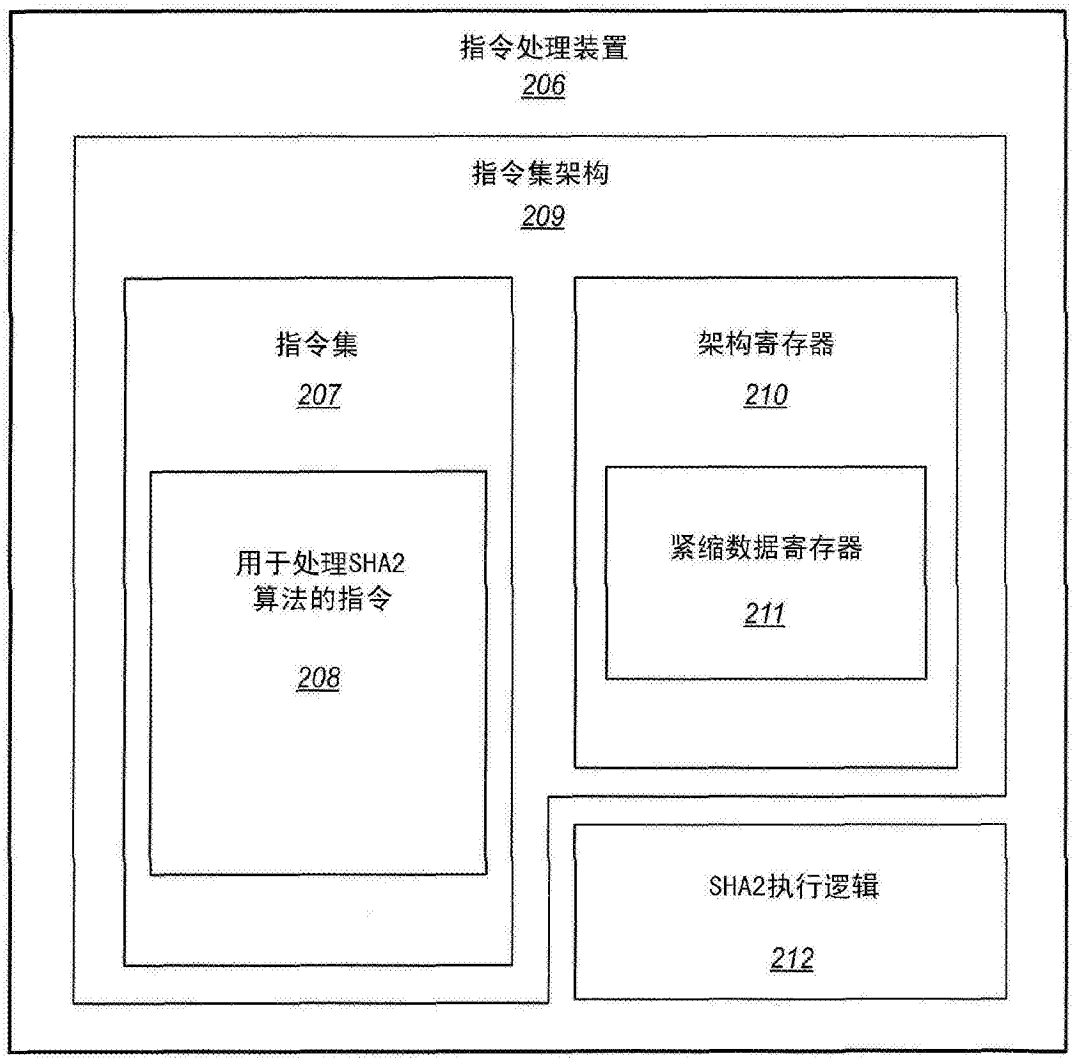

指令处理装置
206
指令集架构
209
指令集
207
用于处理SHA2
算法的指令
208
架构寄存器
210
紧缩数据寄存器
211
SHA2执行逻辑
212

图2

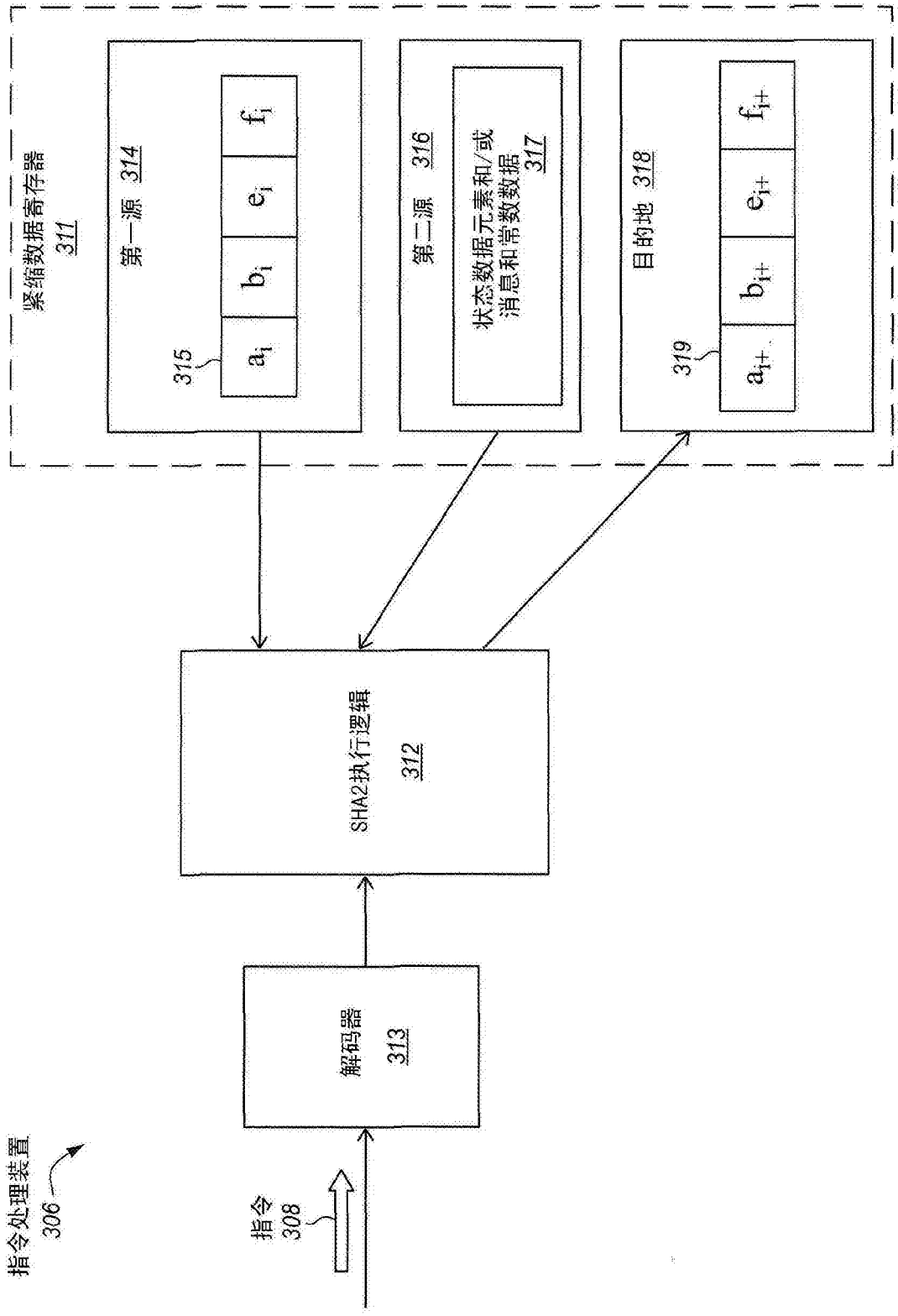

紧缩数据寄存器
311
第一源 314
315
a_i
b_i
e_i
f_i
第二源 316
状态数据元素和/或
消息和常数数据
317
目的地 318
319
a_{i+}
b_{i+}
e_{i+}
f_{i+}
SHA2执行逻辑
312
解码器
313
指令
308
指令处理装置
306

图3

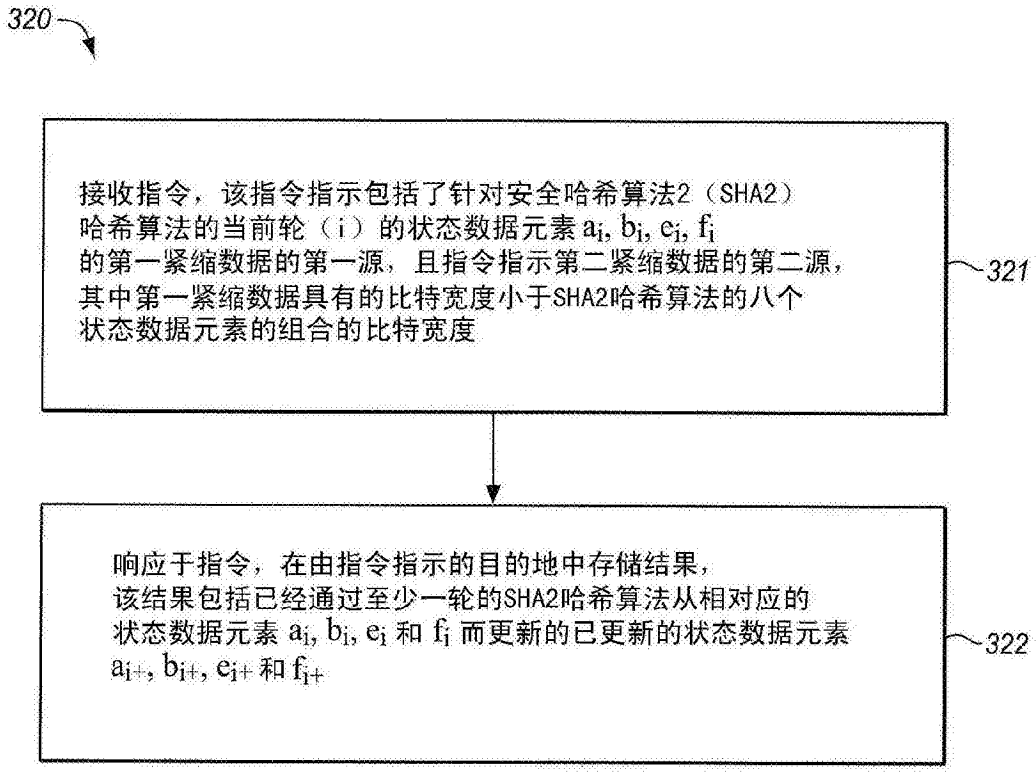

320
接收指令，该指令指示包括了针对安全哈希算法2（SHA2）哈希算法的当前轮（i）的状态数据元素 a_i, b_i, c_i, f_i 的第一紧缩数据的第一源，且指令指示第二紧缩数据的第二源，其中第一紧缩数据具有的比特宽度小于SHA2哈希算法的八个状态数据元素的组合的比特宽度
321
响应于指令，在由指令指示的目的地中存储结果，该结果包括已经通过至少一轮的SHA2哈希算法从相对应的状态数据元素 a_i, b_i, e_i 和 f_i 而更新的已更新的状态数据元素 a_{i+}, b_{i+}, e_{i+} 和 f_{i+}
322

图4

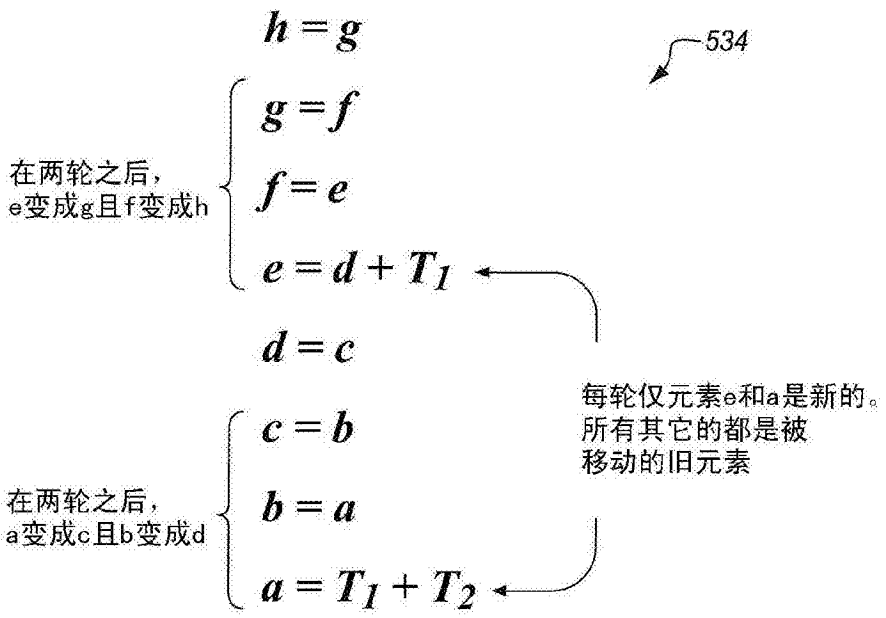

$h = g$
534
在两轮之后，e变成g且f变成h
$g = f$
$f = e$
$e = d + T_1$
$d = c$
每轮仅元素e和a是新的。所有其它的都是被移动的旧元素
在两轮之后，a变成c且b变成d
$c = b$
$b = a$
$a = T_1 + T_2$

图5

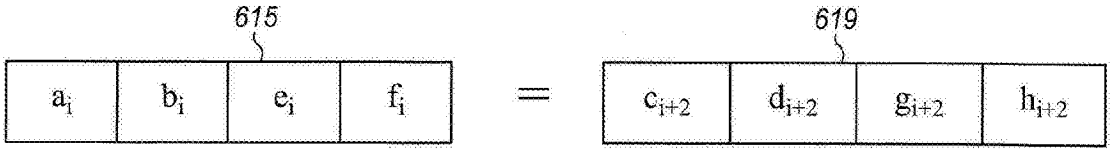

615
619
a_i
b_i
e_i
f_i
=
c_{i+2}
d_{i+2}
g_{i+2}
h_{i+2}

图6

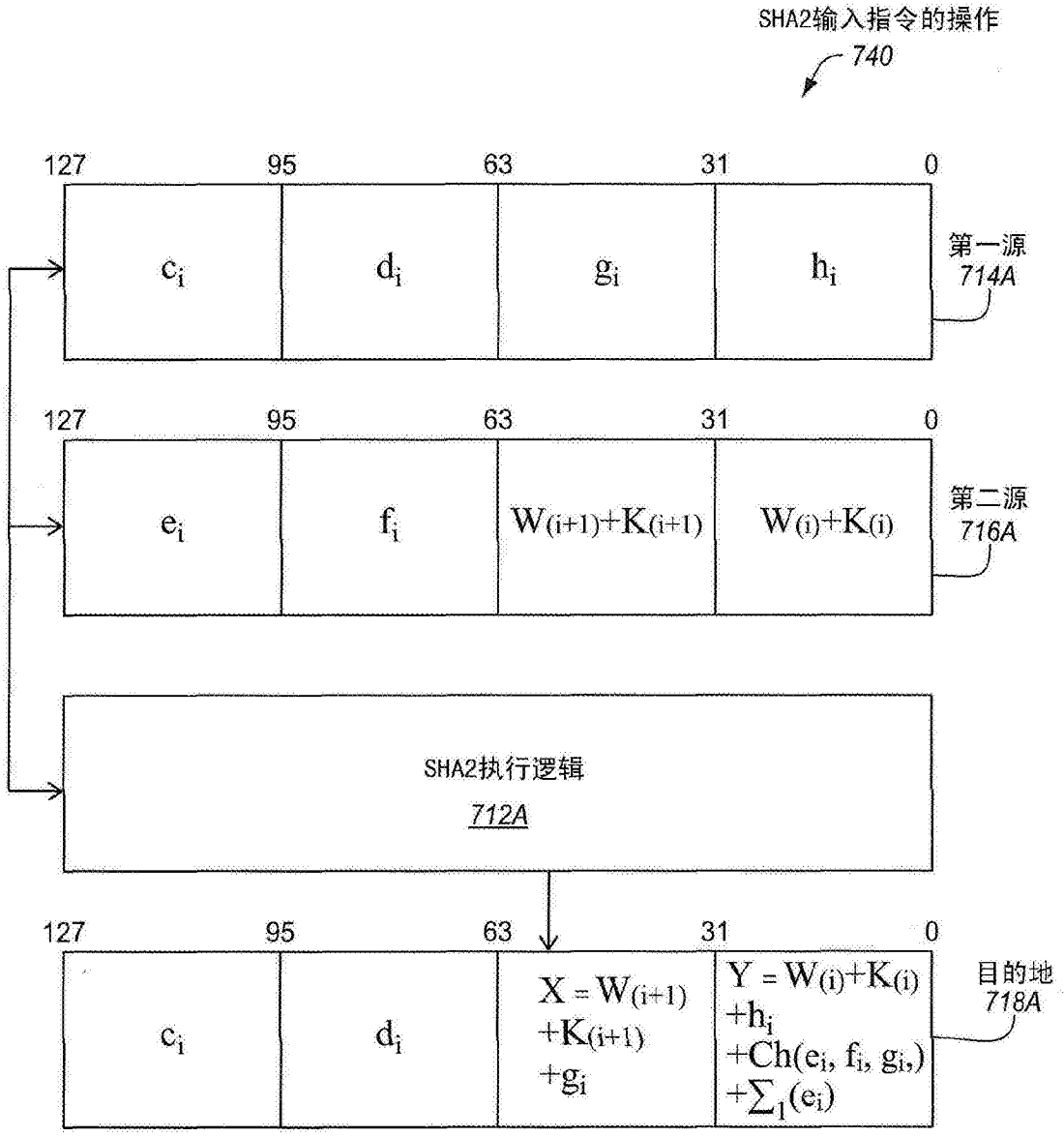

SHA2输入指令的操作
740
127
95
63
31
0
c_i
d_i
g_i
h_i
第一源
714A
e_i
f_i
$W_{(i+1)}+K_{(i+1)}$
$W_{(i)}+K_{(i)}$
第二源
716A
SHA2执行逻辑
712A
$X = W_{(i+1)} + K_{(i+1)} + g_i$
$Y = W_{(i)}+K_{(i)} + h_i + Ch(e_i, f_i, g_i,) + \sum_1(e_i)$
目的地
718A

图7A

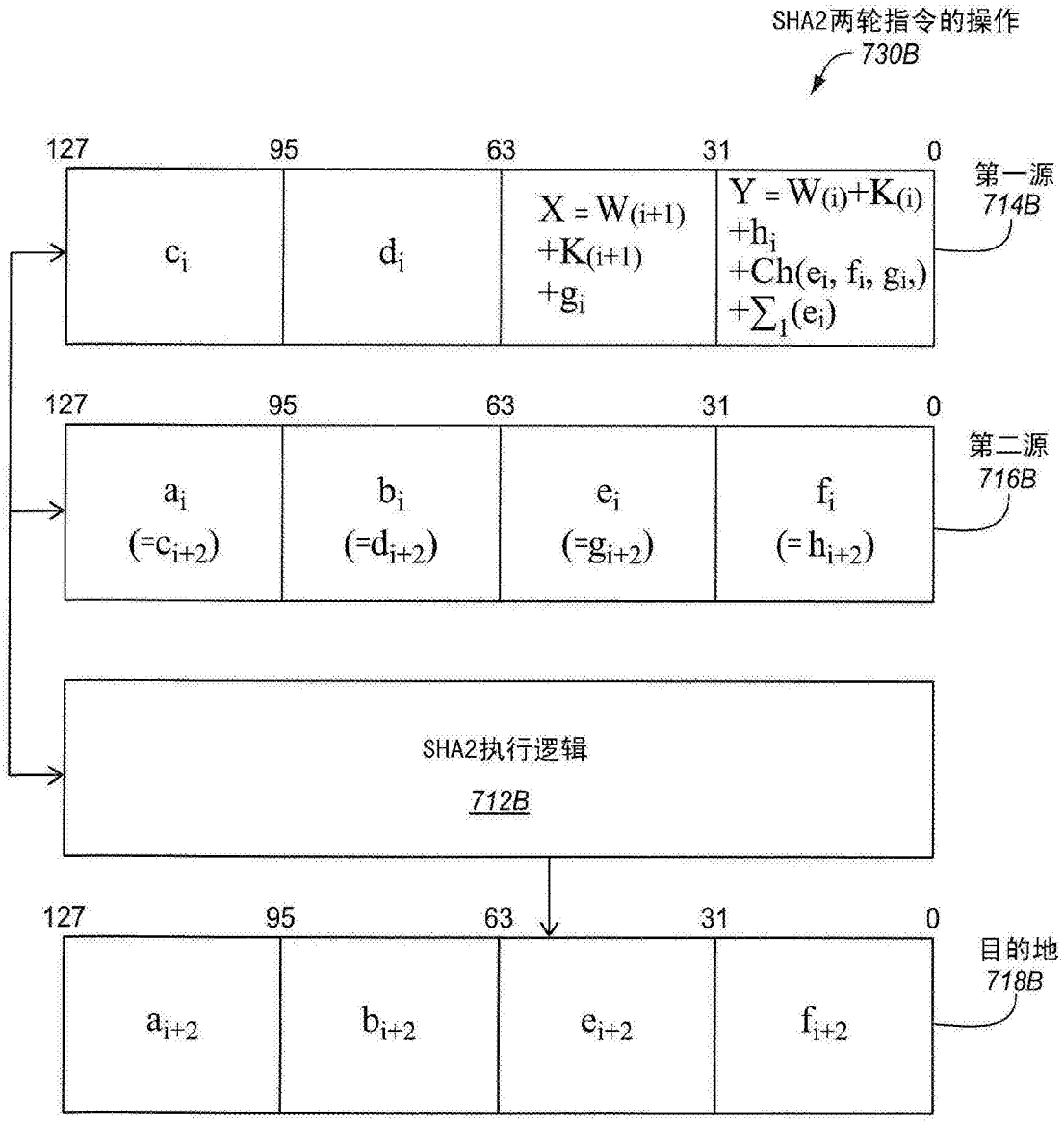

SHA2两轮指令的操作
730B
127
95
63
31
0
c_i
d_i
$X = W_{(i+1)} + K_{(i+1)} + g_i$
$Y = W_{(i)} + K_{(i)} + h_i + Ch(e_i, f_i, g_i,) + \sum_1(e_i)$
第一源
714B
a_i $(=c_{i+2})$
b_i $(=d_{i+2})$
e_i $(=g_{i+2})$
f_i $(=h_{i+2})$
第二源
716B
SHA2执行逻辑
712B
a_{i+2}
b_{i+2}
e_{i+2}
f_{i+2}
目的地
718B

图7B

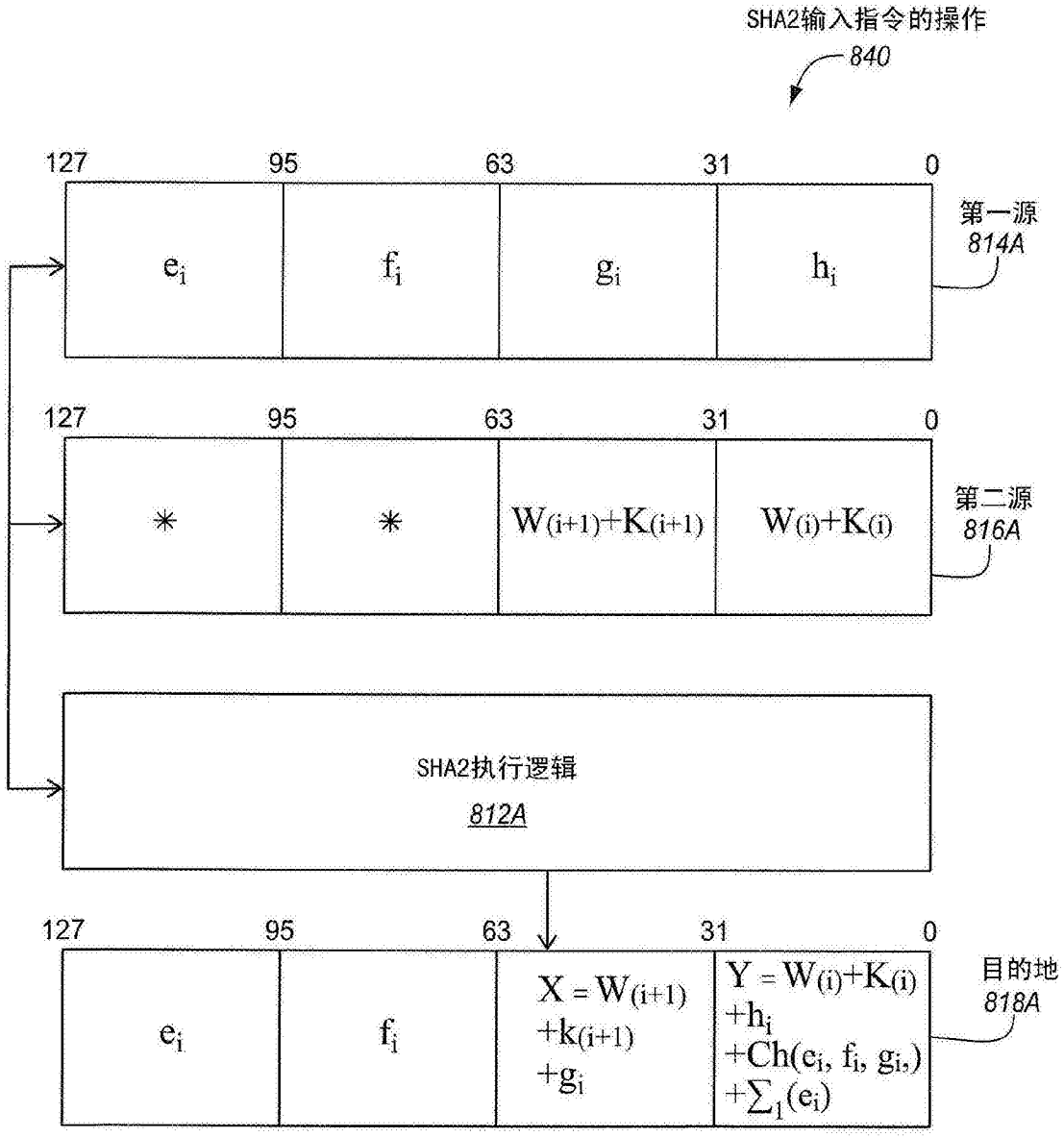

SHA2输入指令的操作
840
127
95
63
31
0
e_i
f_i
g_i
h_i
第一源
814A
*
*
$W_{(i+1)}+K_{(i+1)}$
$W_{(i)}+K_{(i)}$
第二源
816A
SHA2执行逻辑
812A
$X = W_{(i+1)} + k_{(i+1)} + g_i$
$Y = W_{(i)}+K_{(i)} + h_i + Ch(e_i, f_i, g_i,) + \sum_1(e_i)$
目的地
818A

图8A

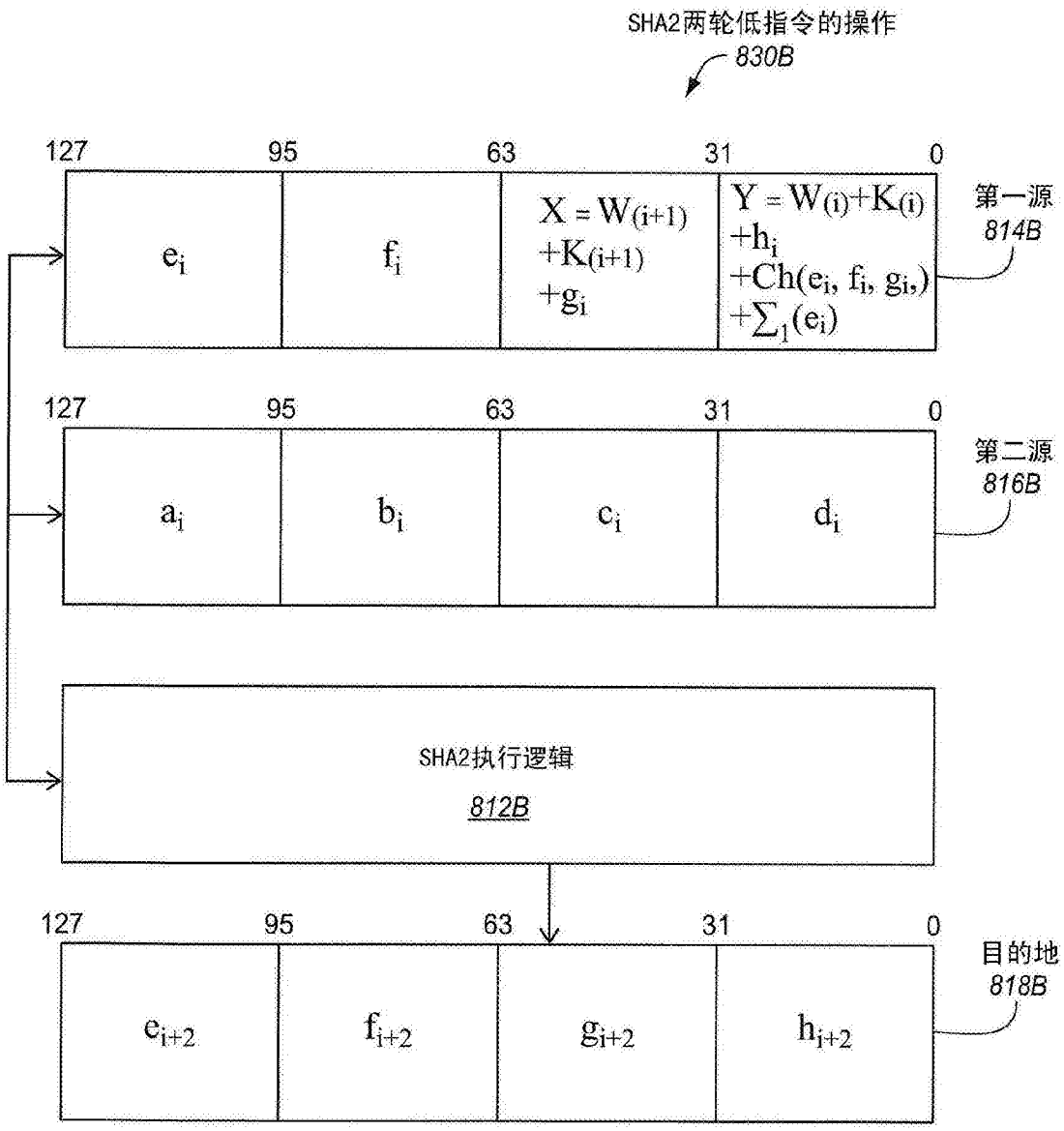

SHA2两轮低指令的操作
830B
127
95
63
31
0
e_i
f_i
$X = W_{(i+1)} + K_{(i+1)} + g_i$
$Y = W_{(i)} + K_{(i)} + h_i + Ch(e_i, f_i, g_i,) + \sum_1(e_i)$
第一源
814B
a_i
b_i
c_i
d_i
第二源
816B
SHA2执行逻辑
812B
e_{i+2}
f_{i+2}
g_{i+2}
h_{i+2}
目的地
818B

图8B

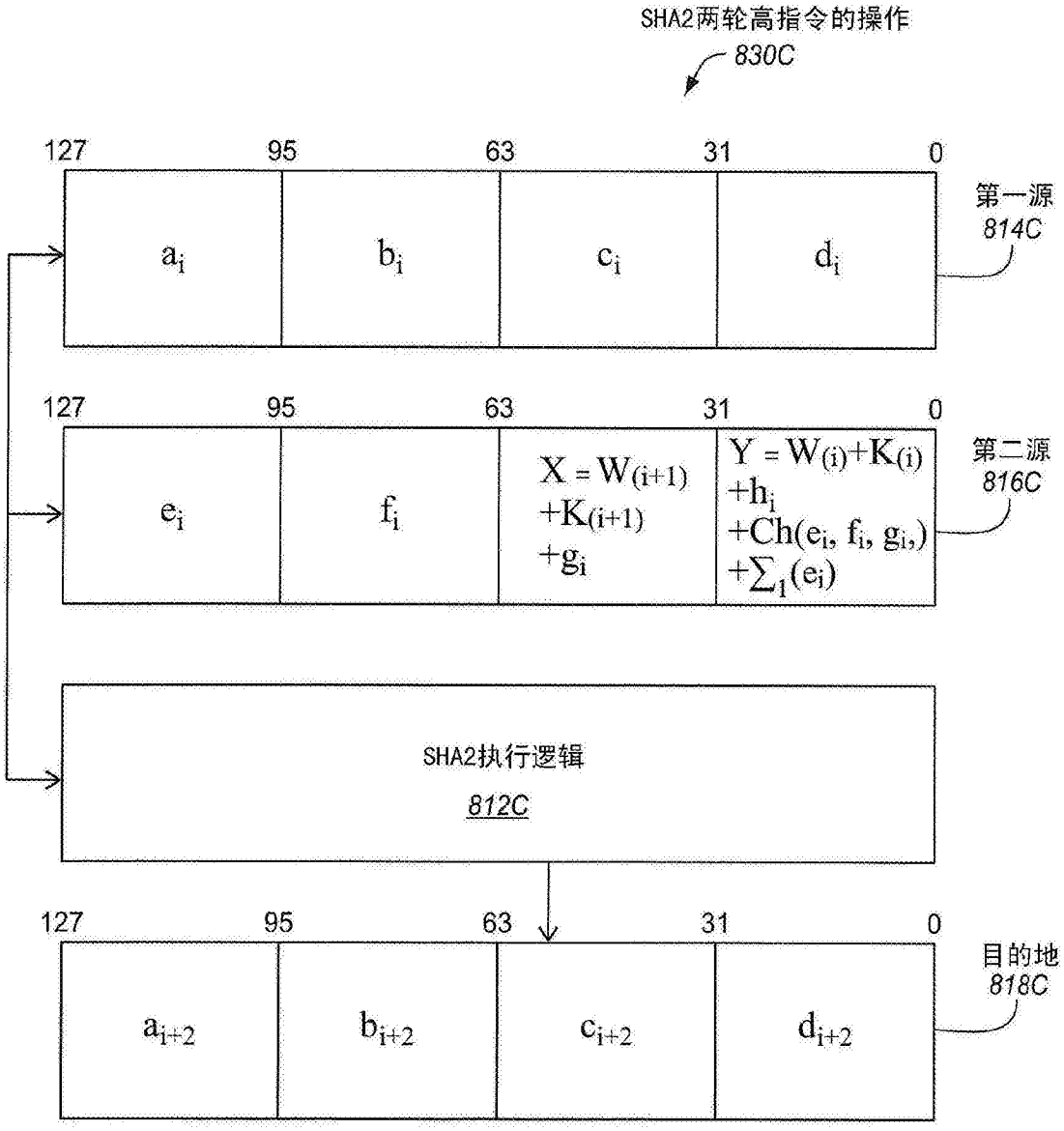

SHA2两轮高指令的操作
830C
127
95
63
31
0
a_i
b_i
c_i
d_i
第一源
814C
e_i
f_i
$X = W_{(i+1)} + K_{(i+1)} + g_i$
$Y = W_{(i)} + K_{(i)} + h_i + Ch(e_i, f_i, g_i,) + \sum_1(e_i)$
第二源
816C
SHA2执行逻辑
812C
a_{i+2}
b_{i+2}
c_{i+2}
d_{i+2}
目的地
818C

图8C

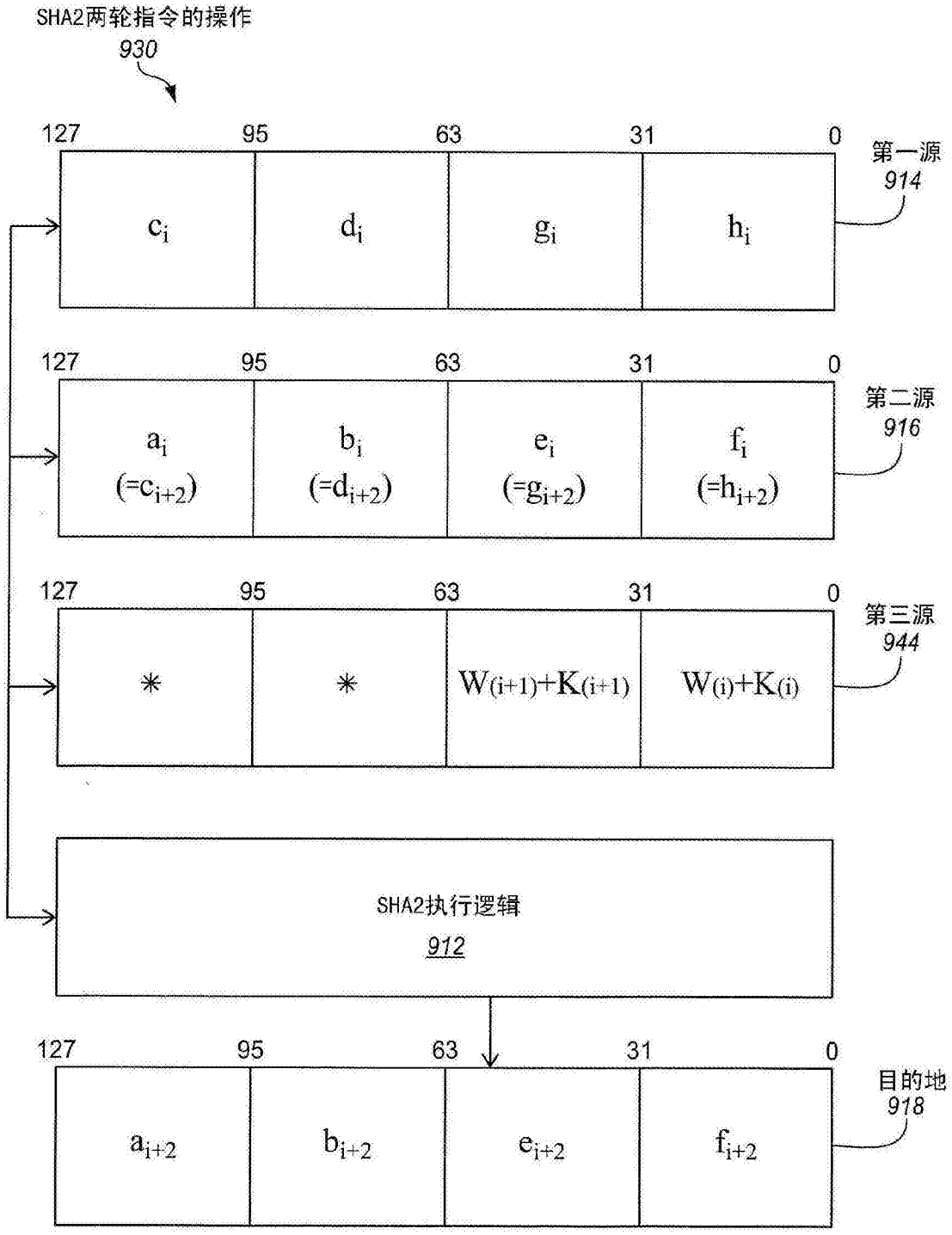

SHA2两轮指令的操作
930
127
95
63
31
0
第一源
914
c_i
d_i
g_i
h_i
第二源
916
a_i (=c_{i+2})
b_i (=d_{i+2})
e_i (=g_{i+2})
f_i (=h_{i+2})
第三源
944
*
*
W(i+1)+K(i+1)
W(i)+K(i)
SHA2执行逻辑
912
目的地
918
a_{i+2}
b_{i+2}
e_{i+2}
f_{i+2}

图9

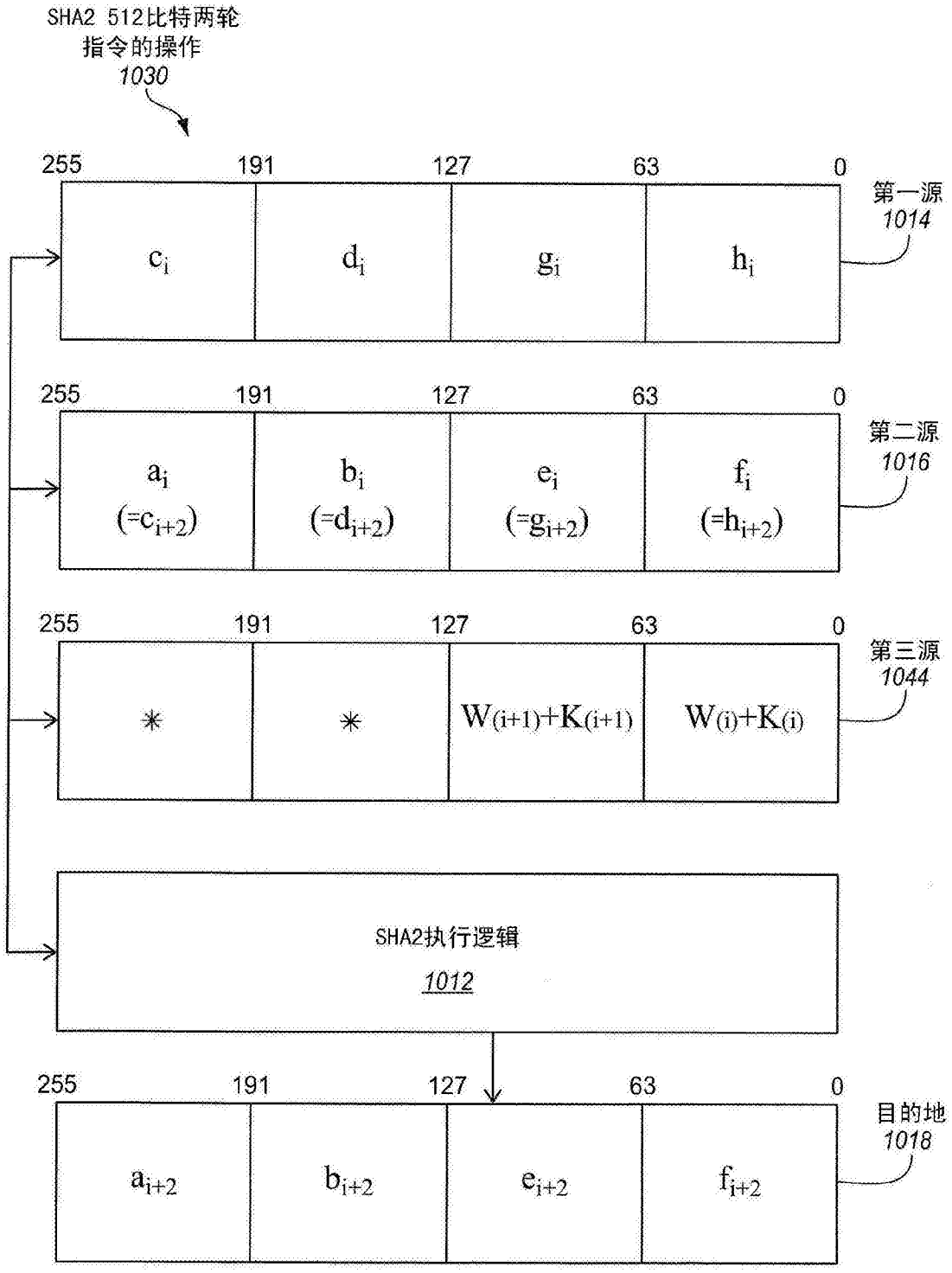

SHA2 512比特两轮
指令的操作
1030
255
191
127
63
0
第一源
1014
c_i
d_i
g_i
h_i
第二源
1016
a_i (=c_{i+2})
b_i (=d_{i+2})
e_i (=g_{i+2})
f_i (=h_{i+2})
第三源
1044
*
*
$W_{(i+1)}+K_{(i+1)}$
$W_{(i)}+K_{(i)}$
SHA2执行逻辑
1012
目的地
1018
a_{i+2}
b_{i+2}
e_{i+2}
f_{i+2}

图10

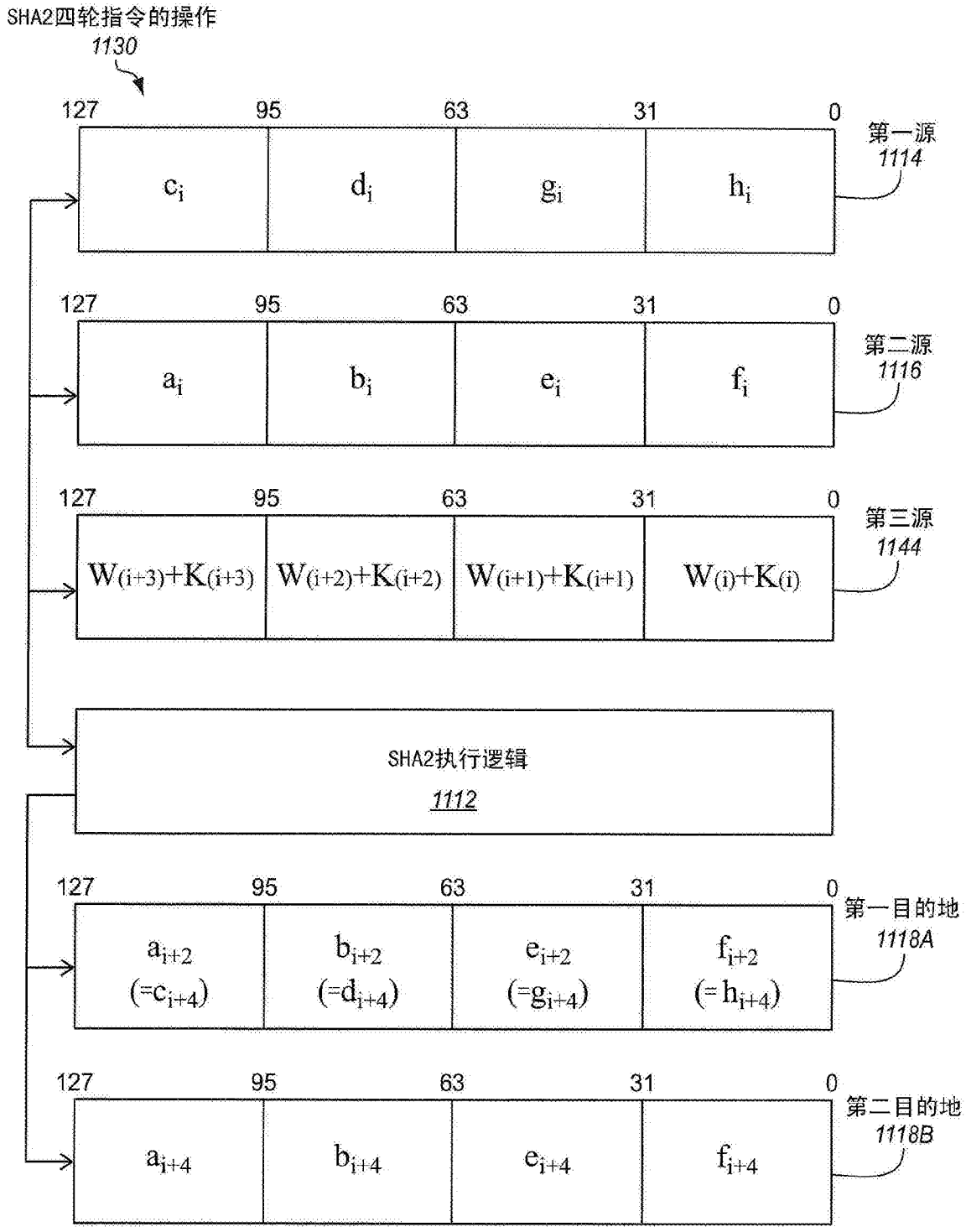

SHA2四轮指令的操作
1130
127
95
63
31
0
c_i
d_i
g_i
h_i
第一源
1114
a_i
b_i
e_i
f_i
第二源
1116
$W_{(i+3)}+K_{(i+3)}$
$W_{(i+2)}+K_{(i+2)}$
$W_{(i+1)}+K_{(i+1)}$
$W_{(i)}+K_{(i)}$
第三源
1144
SHA2执行逻辑
1112
a_{i+2} ($=c_{i+4}$)
b_{i+2} ($=d_{i+4}$)
e_{i+2} ($=g_{i+4}$)
f_{i+2} ($=h_{i+4}$)
第一目的地
1118A
a_{i+4}
b_{i+4}
e_{i+4}
f_{i+4}
第二目的地
1118B

图11

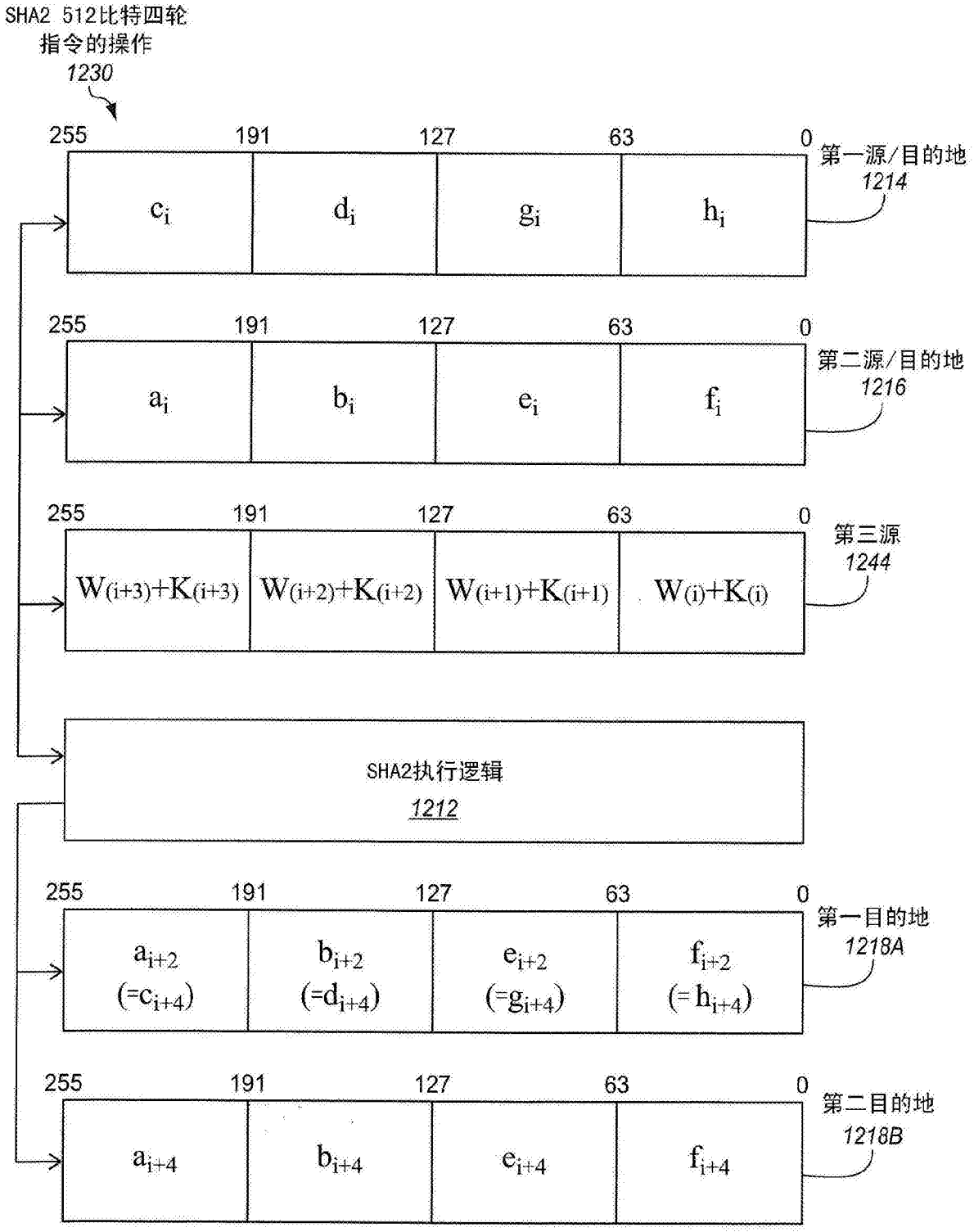

SHA2 512比特四轮
指令的操作
1230
255
191
127
63
0
c_i
d_i
g_i
h_i
第一源/目的地
1214
a_i
b_i
e_i
f_i
第二源/目的地
1216
$W_{(i+3)}+K_{(i+3)}$
$W_{(i+2)}+K_{(i+2)}$
$W_{(i+1)}+K_{(i+1)}$
$W_{(i)}+K_{(i)}$
第三源
1244
SHA2执行逻辑
1212
a_{i+2} ($=c_{i+4}$)
b_{i+2} ($=d_{i+4}$)
e_{i+2} ($=g_{i+4}$)
f_{i+2} ($=h_{i+4}$)
第一目的地
1218A
a_{i+4}
b_{i+4}
e_{i+4}
f_{i+4}
第二目的地
1218B

图12

图13A

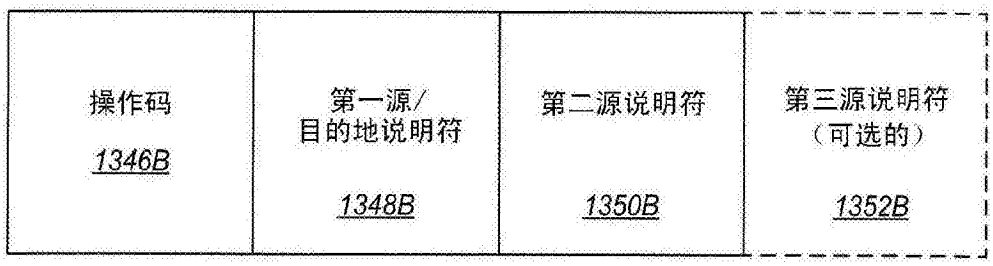


图13B

图13C

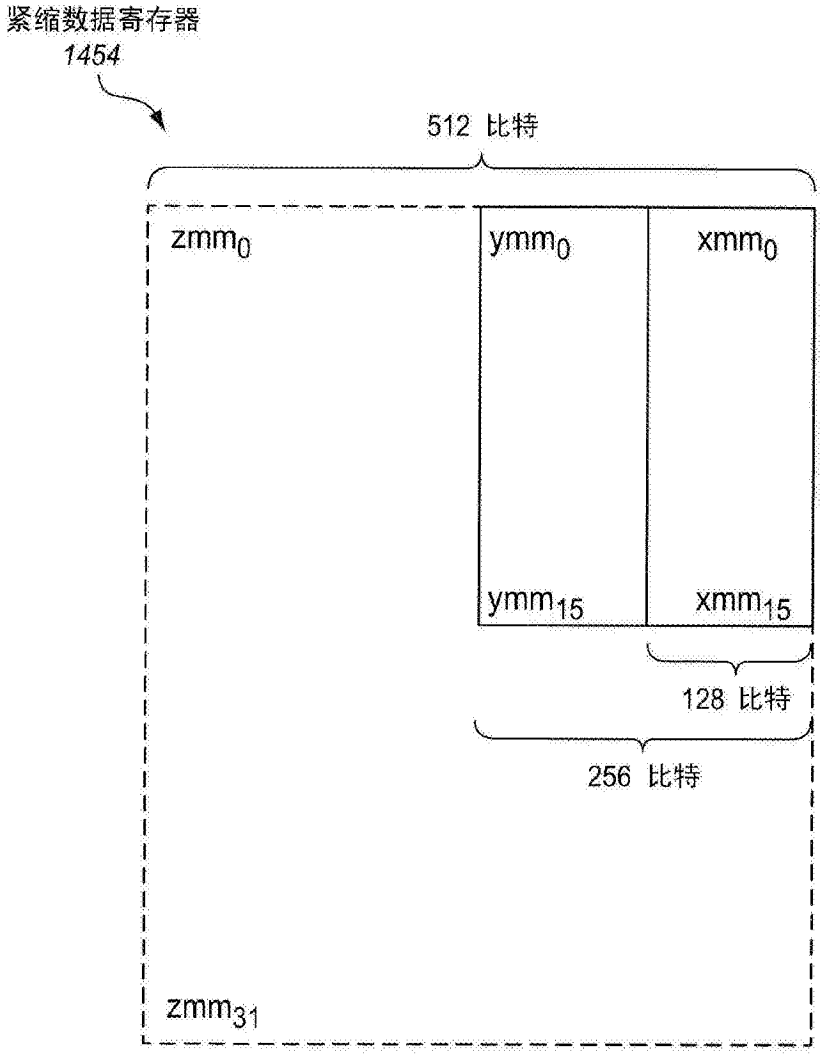

紧缩数据寄存器
1454
512 比特
zmm0
ymm0
xmm0
ymm15
xmm15
128 比特
256 比特
zmm31

图14

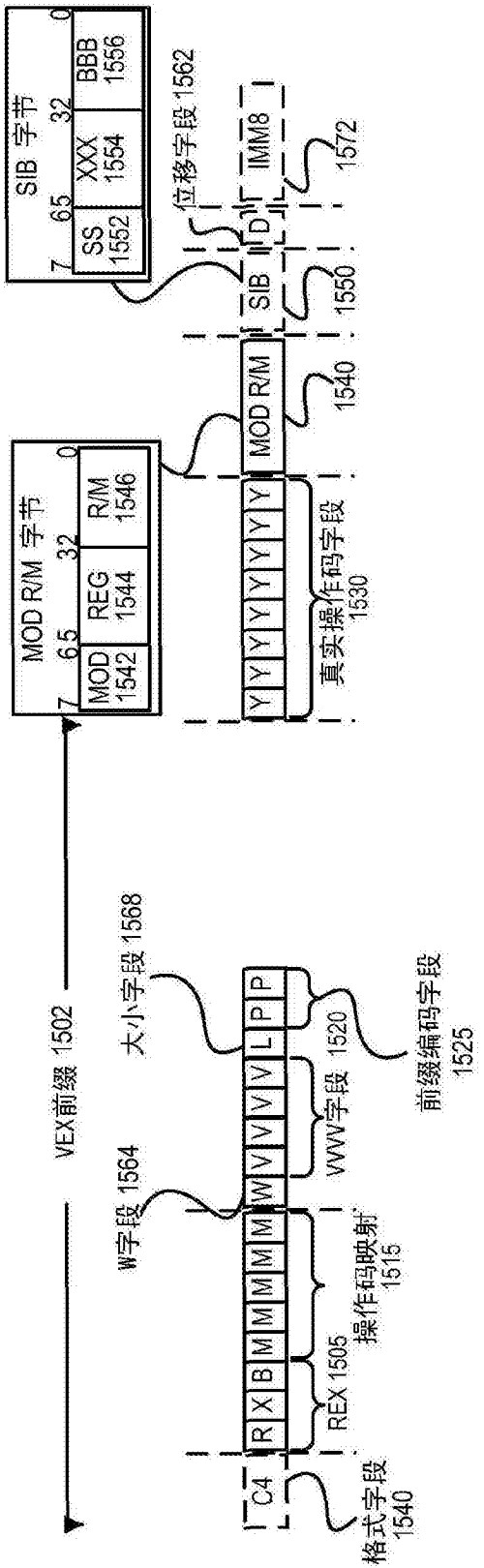

SIB 字节
7 6 5 3 2 0
SS 1552
XXX 1554
BBB 1556
位移字段 1562
MOD R/M 字节
MOD 1542
REG 1544
R/M 1546
SIB
D
IMM8
1572
1550
MOD R/M
1540
真实操作码字段 1530
Y Y Y Y Y Y Y Y
VEX前缀 1502
大小字段 1568
W字段 1564
VVVV字段 1520
前缀编码字段 1525
操作码映射 1515
REX 1505
C4
格式字段 1540
R X B M M M M M W V V V V L P P

图15A

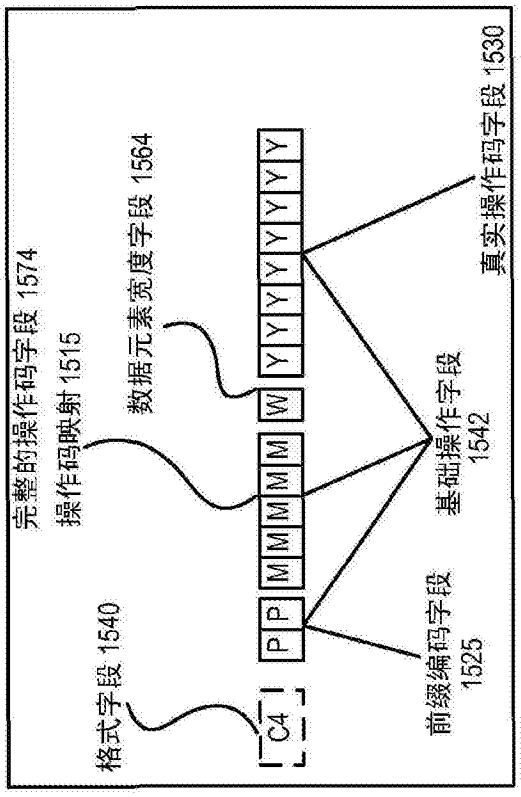

完整的操作码字段 1574
操作码映射 1515
格式字段 1540
C4
P P M M M M M W Y Y Y Y Y Y Y Y
数据元素宽度字段 1564
前缀编码字段 1525
基础操作字段 1542
真实操作码字段 1530

图15B

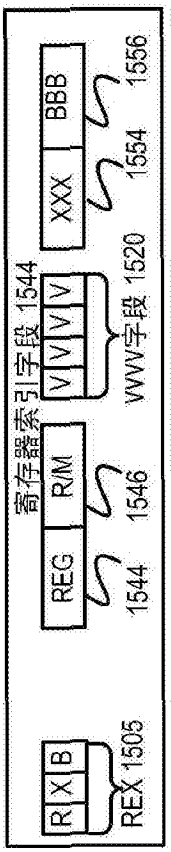

寄存器索引字段 1544
R X B
REG
R/M
V V V V
XXX
BBB
REX 1505
1544
1546
VVVV字段 1520
1554
1556

图15C

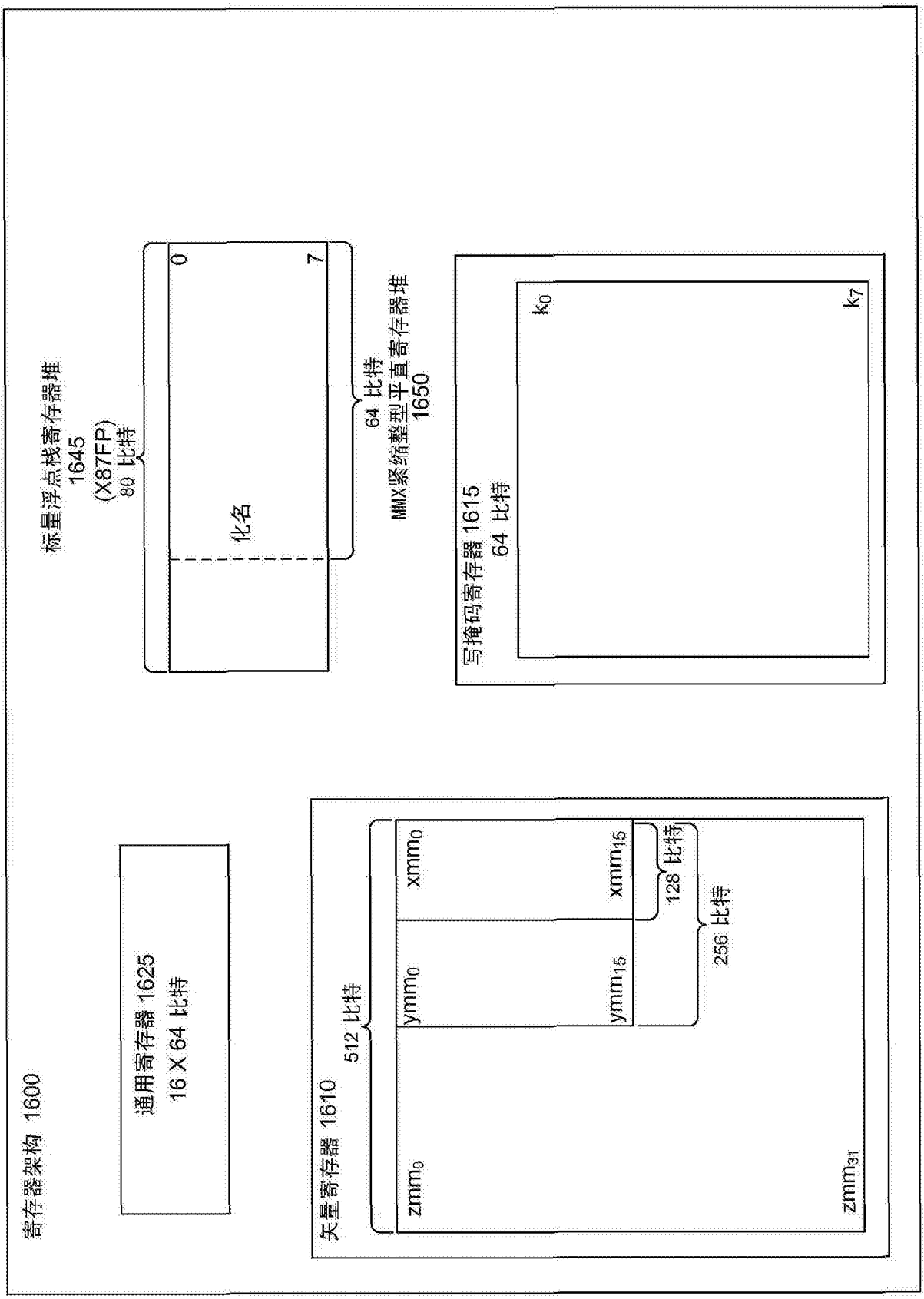

寄存器架构 1600
通用寄存器 1625
16 X 64 比特
标量浮点栈寄存器堆
1645
(X87FP)
80 比特
0
7
化名
64 比特
MMX紧缩整型平直寄存器堆
1650
写掩码寄存器 1615
64 比特
k0
k7
矢量寄存器 1610
512 比特
zmm0
ymm0
xmm0
ymm15
xmm15
128 比特
256 比特
zmm31

图16

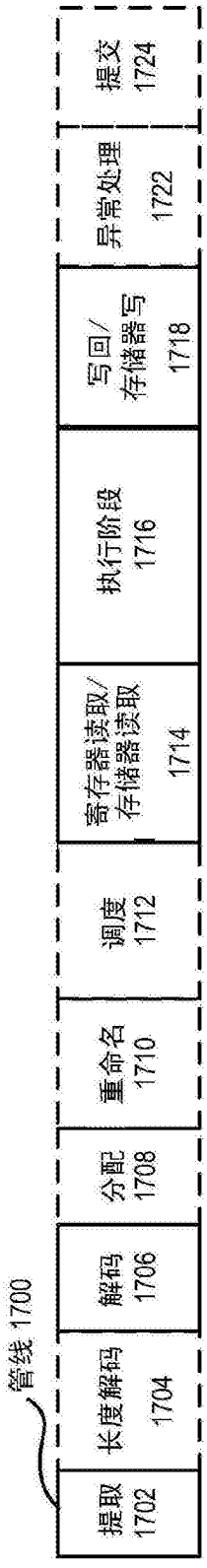

管线 1700
提取 1702
长度解码 1704
解码 1706
分配 1708
重命名 1710
调度 1712
寄存器读取/存储器读取 1714
执行阶段 1716
写回/存储器写 1718
异常处理 1722
提交 1724

图17A

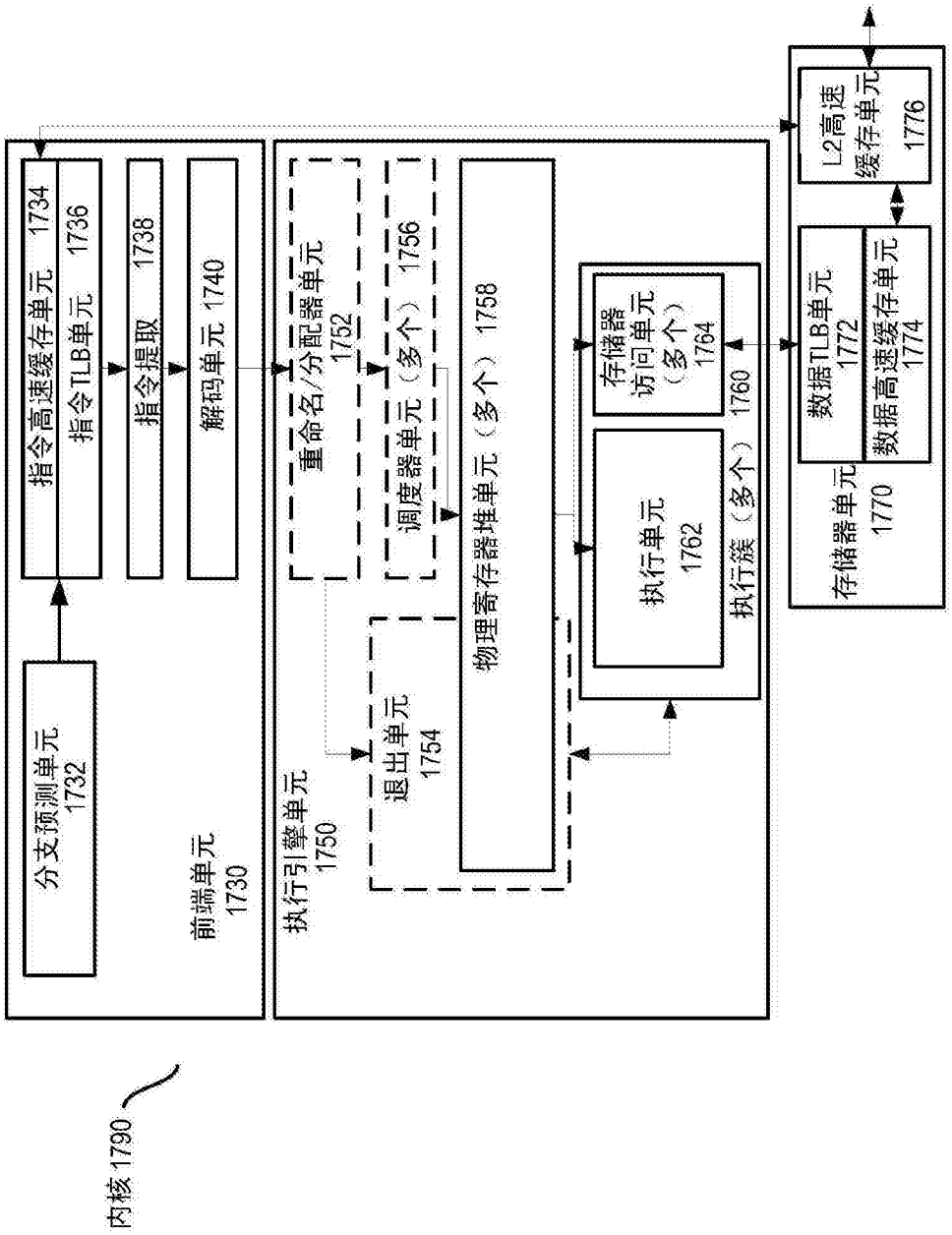

内核 1790
前端单元 1730
分支预测单元 1732
指令高速缓存单元 1734
指令TLB单元 1736
指令提取 1738
解码单元 1740
执行引擎单元 1750
重命名/分配器单元 1752
退出单元 1754
调度器单元（多个） 1756
物理寄存器堆单元（多个） 1758
执行簇（多个） 1760
执行单元 1762
存储器访问单元（多个） 1764
存储器单元 1770
数据TLB单元 1772
数据高速缓存单元 1774
L2高速缓存单元 1776

图17B

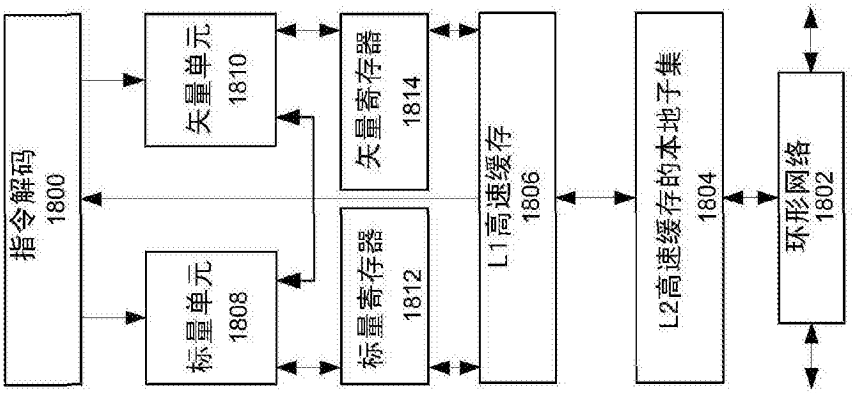

指令解码
1800
矢量单元
1810
标量单元
1808
矢量寄存器
1814
标量寄存器
1812
L1高速缓存
1806
L2高速缓存的本地子集
1804
环形网络
1802

图18A

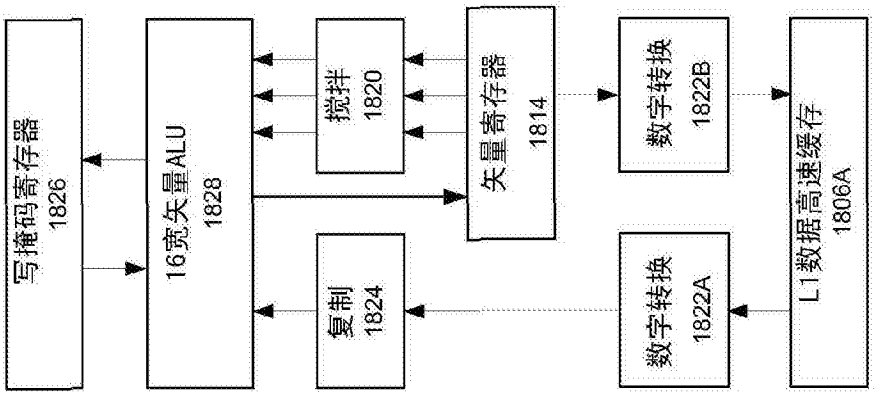

写掩码寄存器
1826
16宽矢量ALU
1828
搅拌
1820
矢量寄存器
1814
数字转换
1822B
复制
1824
数字转换
1822A
L1数据高速缓存
1806A

图18B

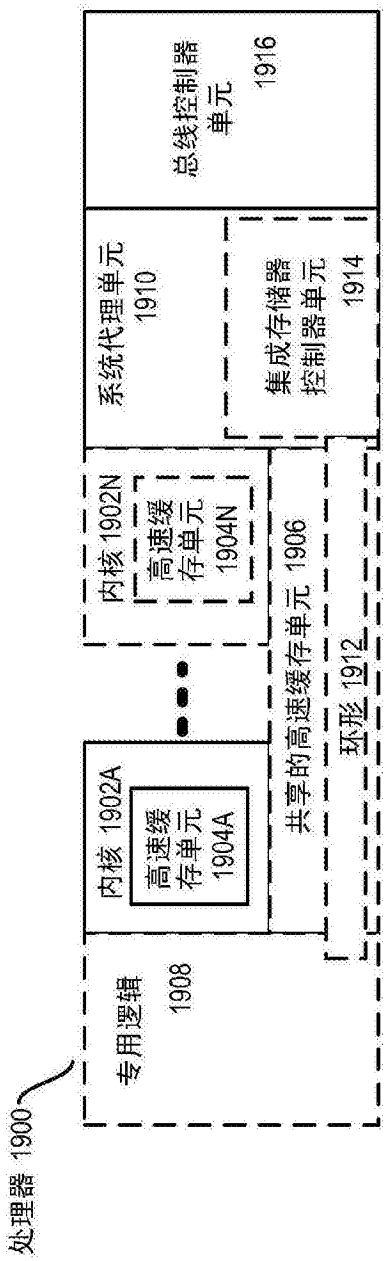

处理器 1900
专用逻辑 1908
内核 1902A
高速缓存单元 1904A
内核 1902N
高速缓存单元 1904N
共享的高速缓存单元 1906
环形 1912
系统代理单元 1910
集成存储器控制器单元 1914
总线控制器单元 1916

图19

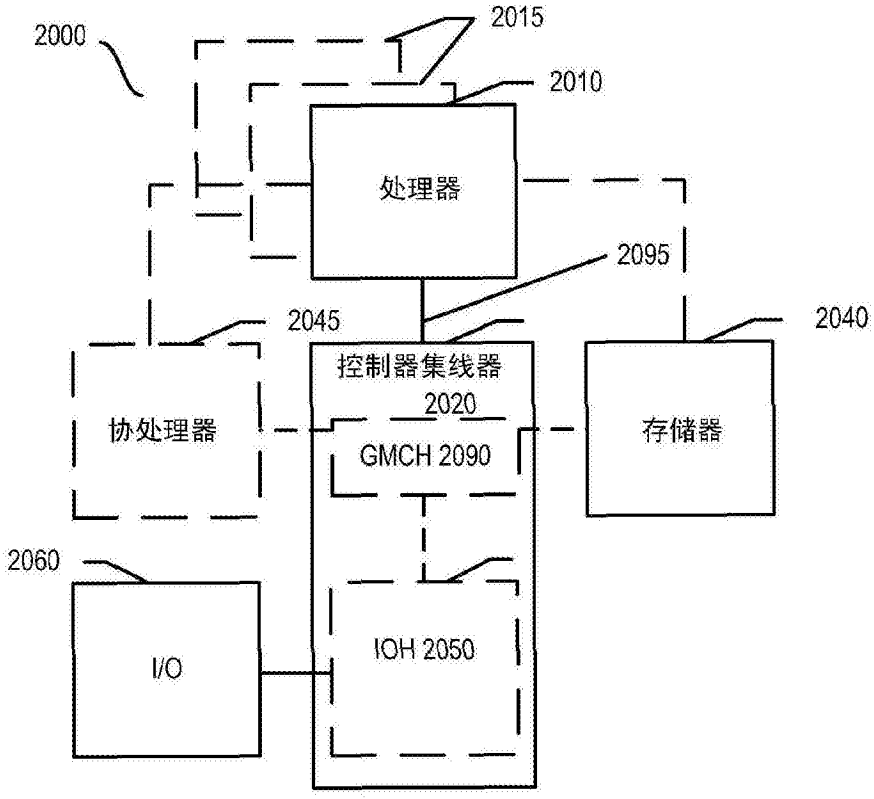

2000
2015
2010
处理器
2095
2045
2040
控制器集线器
2020
协处理器
GMCH 2090
存储器
2060
I/O
IOH 2050

图20

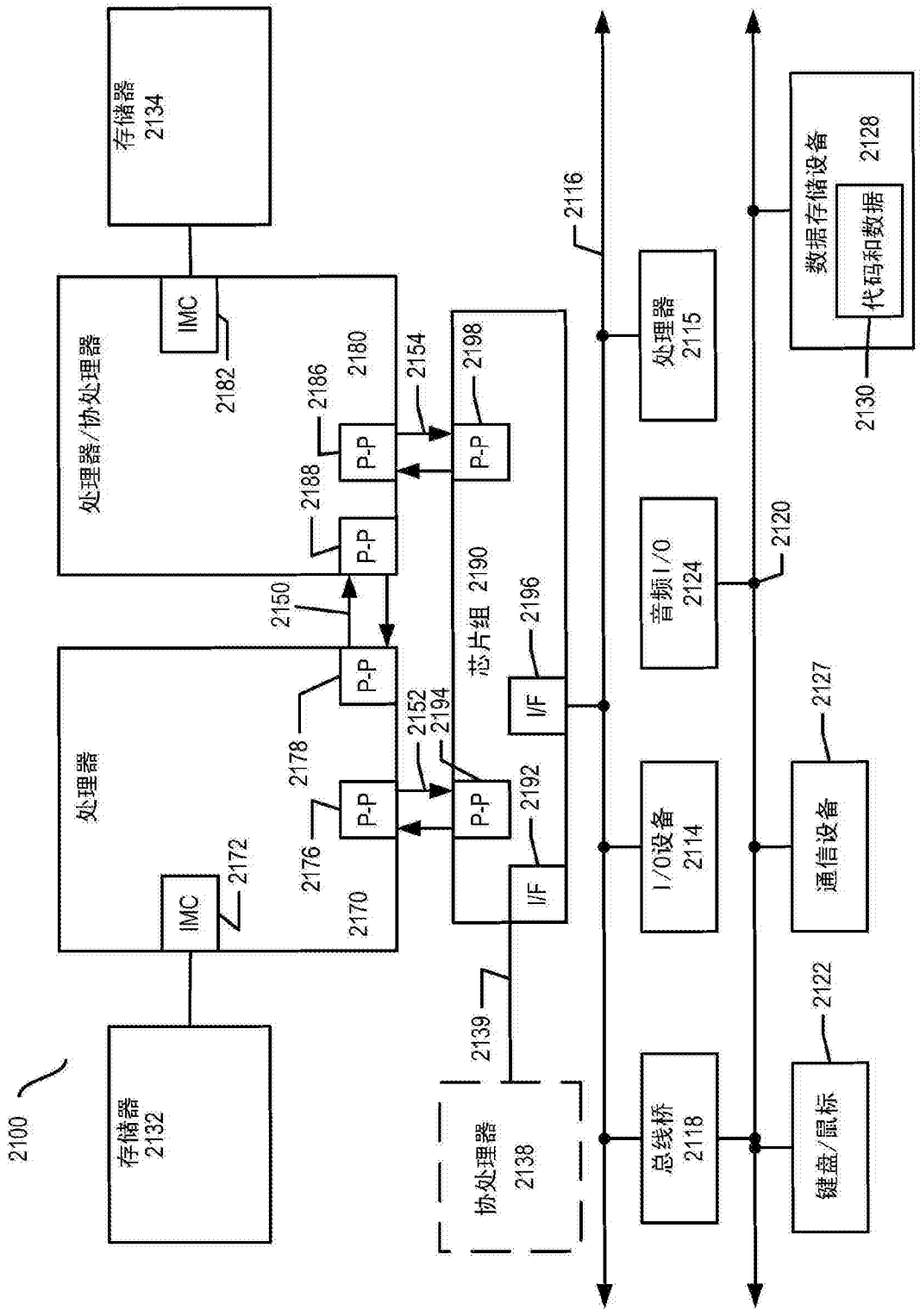

2100
存储器 2132
存储器 2134
处理器
处理器/协处理器
IMC
2172
2182
2170
2180
2178
2176
2186
2188
2150
2152
2154
2194
2198
2196
2192
芯片组 2190
P-P
I/F
协处理器 2138
2139
处理器 2115
2116
音频I/O 2124
I/O设备 2114
总线桥 2118
通信设备
2127
2120
数据存储设备 2128
代码和数据
2130
键盘/鼠标
2122

图21

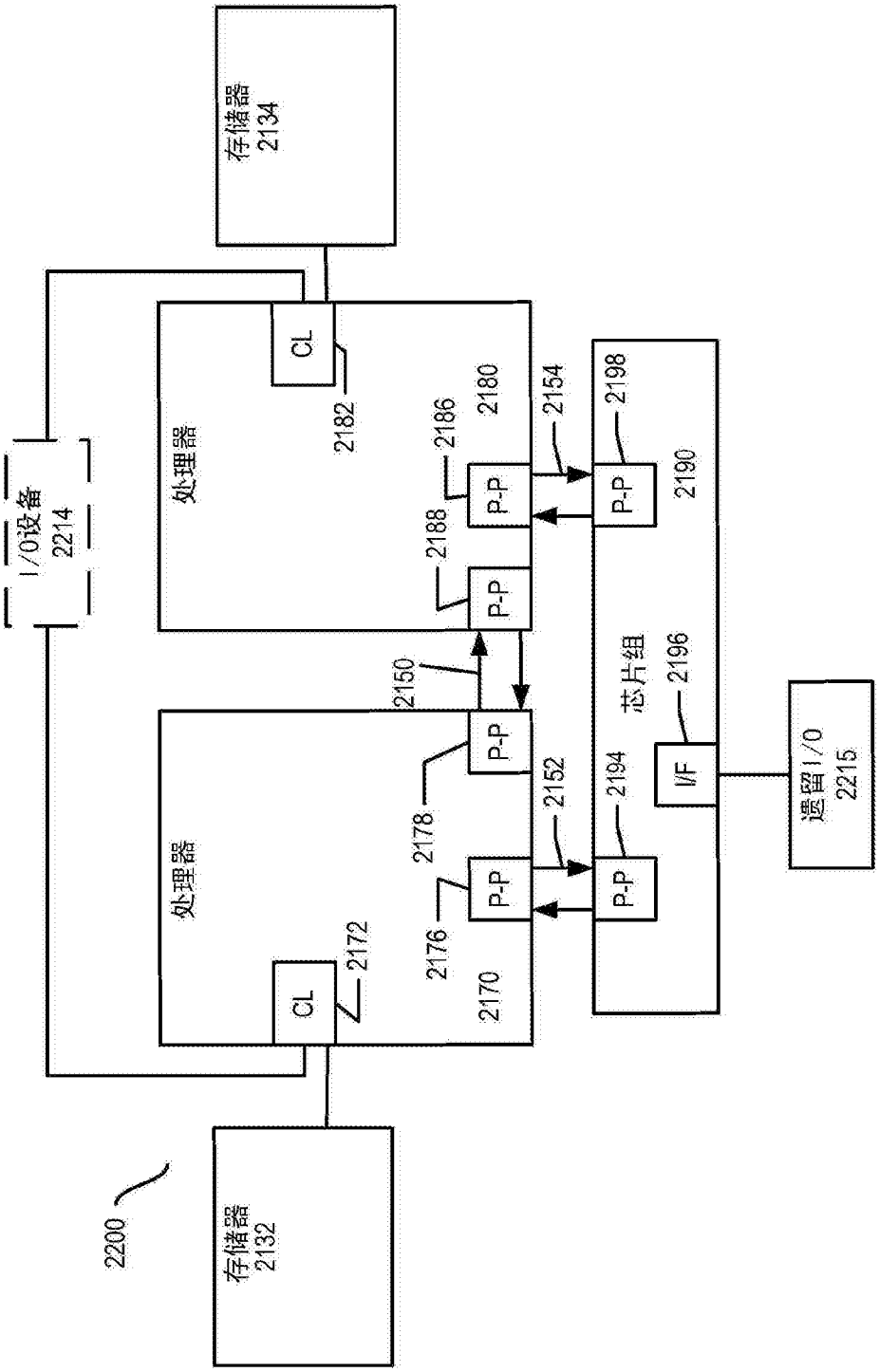

存储器
2134
CL
2182
处理器
2180
2186
P-P
2188
P-P
2154
2198
P-P
2190
芯片组
2196
I/F
遗留I/O
2215
2150
P-P
2178
P-P
2176
2152
2194
P-P
处理器
2170
CL
2172
I/O设备
2214
2200
存储器
2132

图22

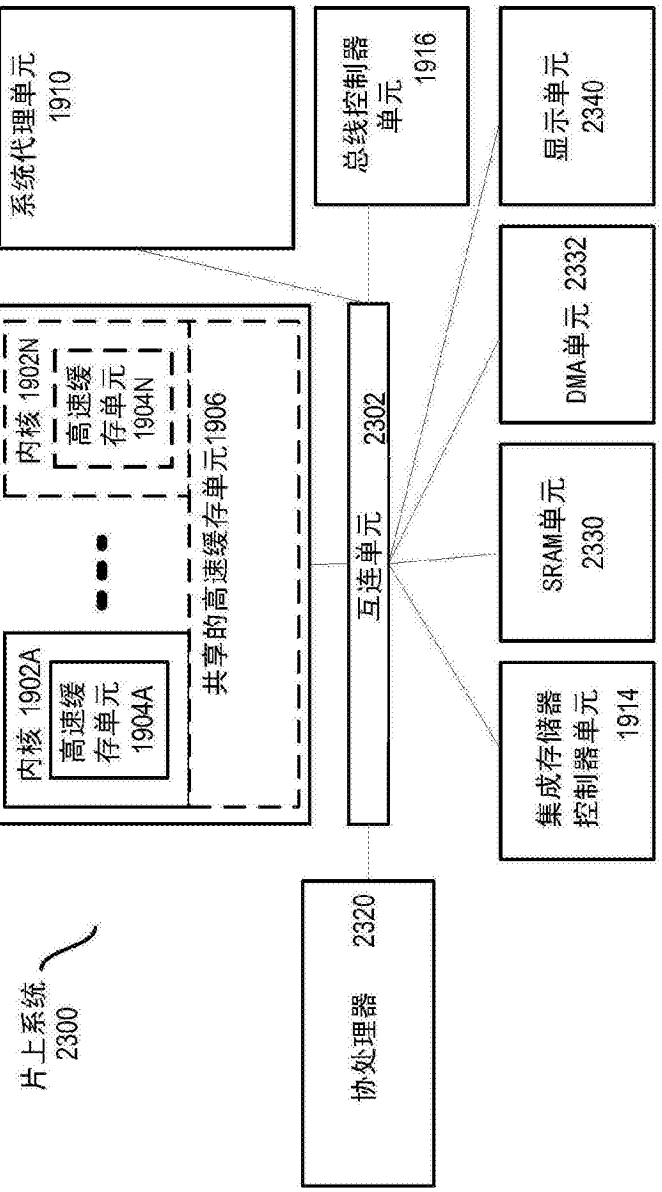

系统代理单元
1910
总线控制器
单元
1916
显示单元
2340
应用处理器
2310
内核 1902N
高速缓
存单元
1904N
内核 1902A
高速缓
存单元
1904A
共享的高速缓存单元1906
互连单元
2302
DMA单元
2332
SRAM单元
2330
集成存储器
控制器单元
1914
协处理器
2320
片上系统
2300

图23

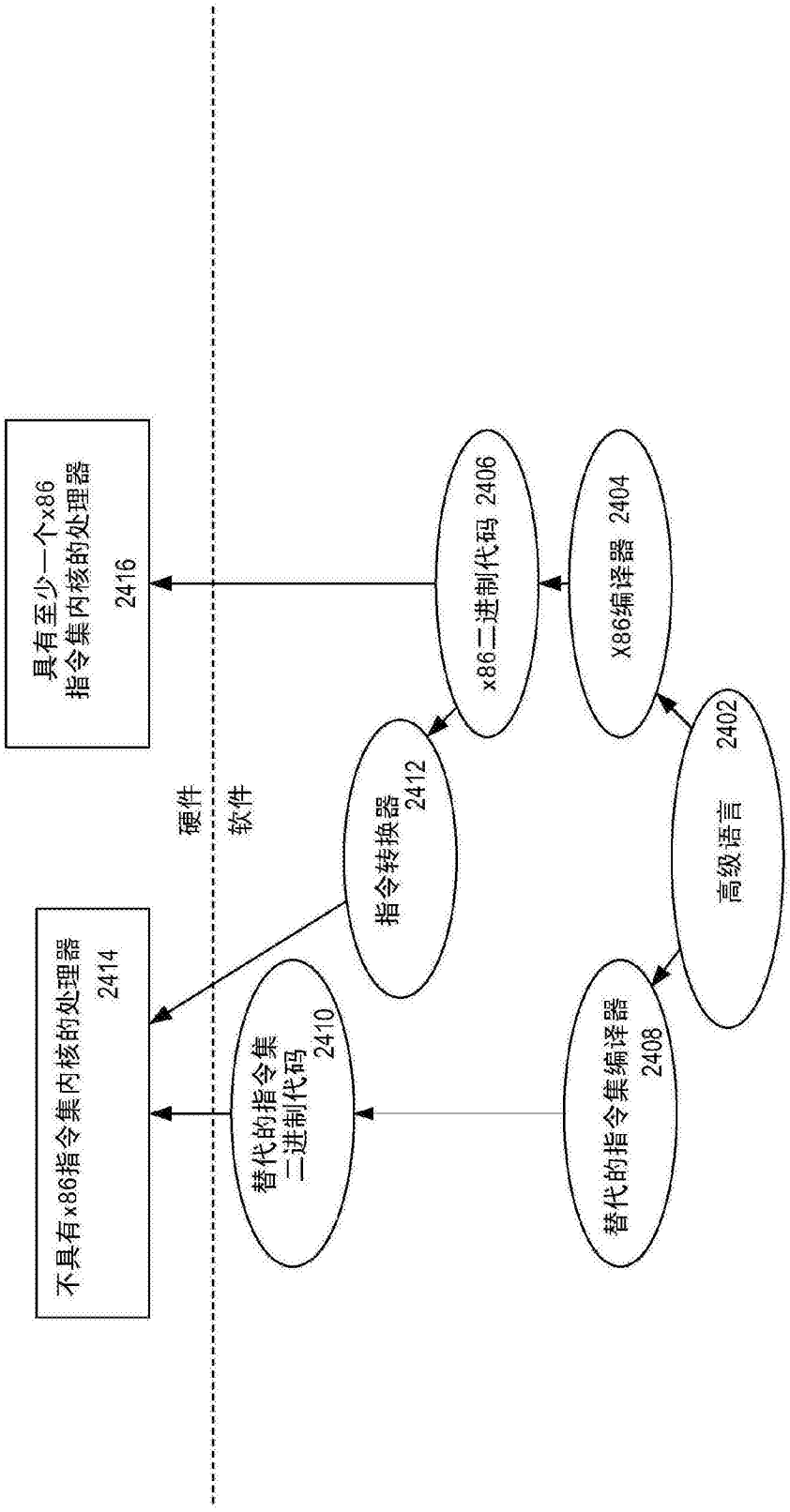

具有至少一个x86
指令集内核的处理器
2416
x86二进制代码 2406
X86编译器 2404
高级语言 2402
指令转换器
2412
硬件
软件
不具有x86指令集内核的处理器
2414
替代的指令集
二进制代码
2410
替代的指令集编译器
2408

图24