(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104158648 A
(43)申请公布日 2014.11.19

(21)申请号 201410199922.X

(22)申请日 2014.05.13

(30)优先权数据
102013208836.1 2013.05.14 DE

(71)申请人 罗伯特·博世有限公司
地址 德国斯图加特

(72)发明人 M. 刘易斯

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 臧永杰 胡莉莉

(51)Int.Cl.
H04L 9/00(2006.01)

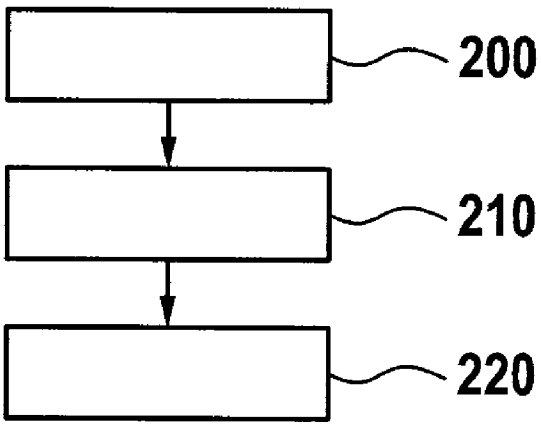
权利要求书4页 说明书18页 附图10页

(54)发明名称

用于生成哈希值的方法和装置

(57)摘要

本发明涉及生成哈希值的方法，包括：a. 将输入数据划分为十六个分别具有32*m位长度的输入数据块，m是大于等于1的整数，下标变量i=0,…,15表示第i个输入数据块M_i，b. 用可预先给定的值初始化八个工作数据块，八个工作数据块中的每个都具有32*m位的长度，下标变量k=0,…,7表示第k个工作数据块M_k，c. 按照如下规则改变输入数据块和工作数据块：对于i=1至15，将输入数据块$M_{i,n}$的内容指派给输入数据块$M_{i-1,n+1}$，n是大于等于零的整数并且代表处理周期；对于k=0，k=1，k=2以及对于k=4，k=5，k=6，将工作数据块$W_{k,n}$的内容指派给工作数据块$W_{k+1,n+1}$；将第一、第二、第三函数T、G、F的输出值分别指派给输入数据块$M_{15,n+1}$，工作数据块$W_{0,n+1}$，工作数据块$W_{4,n+1}$，其中改变的步骤c)被执行N次，N＞1。



CN 104158648 A

1. 一种用于根据数字输入数据(M)生成哈希值(HW)的方法,其中该方法包括下面的步骤:

a)将输入数据(M)划分(200)为分别具有 32*m 位长度的十六个输入数据块(M_0,M_1,…,M_{15}),其中 m 是大于等于 1 的整数,并且其中下标变量 i=0,…,15 表示第 i 个输入数据块 M_i,

b)用可预先给定的值初始化(210)八个工作数据块(W_0, W_1, …, W_7),其中八个工作数据块(W_0, W_1, …, W_7)中的每个都具有 32*m 位的长度,并且其中下标变量 k=0,…,7 表示第 k 个工作数据块 W_k,

c)按照如下规则改变(220)输入数据块(M_0,M_1, …,M_{15})和工作数据块(W_0,W_1, …,W_7):

c1)对于 i=1 至 15,将输入数据块 $M_{i,n}$ 的内容指派(222a)给输入数据块 $M_{i-1,n+1}$,其中 n 是大于等于零的整数并且代表处理周期,

c2)对于 k=0,k=1,k=2 以及对于 k=4,k=5,k=6,将工作数据块 $W_{k,n}$ 的内容指派(222b)给工作数据块 $W_{k+1,n+1}$,

c3)将第一函数 T 的输出值指派(224)给输入数据块 $M_{15,n+1}$,

c4)将第二函数 G 的输出值指派(226)给工作数据块 $W_{0,n+1}$,

c5)将第三函数 F 的输出值指派给(228)工作数据块 $W_{4,n+1}$,

其中改变(220)的步骤 c) 被执行 N 次,其中 N ＞ 1。

2. 根据权利要求 1 所述的方法,其中

A)在 m=1 的情况下

- 函数 T 被定义为 $T = M_{0,n} + M_{9,n} + (ROTR^{17}(M_{14,n})$ $XOR\ \ ROTR^{19}(M_{14,n})\ XOR\ SHR^{10}(M_{14,n})) + (ROTR^{7}(M_{1,n})$ $XOR\ ROTR^{18}(M_{1,n})\ XOR\ SHR^{3}(M_{1,n}))$,其中 $ROTR^{y}(x)$ 是操作数 x 向右逐位旋转 y 个许多位,其中 $SHR^{y}(x)$ 是操作数 x 向右逐位逻辑移位 y 个许多位,其中 XOR 是异或逻辑连接,

- 函数 G 被定义为 G=T0+T1,其中

$T0 = M_{0,n} +\ \ W_{7,n} + (ROTR^{6}(W_{4,n})\ XOR\ ROTR^{11}(W_{4,n})\ XOR\ \ \ ROTR^{25}(W_{4,n})$ $) + ((W_{4,n}\ AND\ W_{5,n})\ XOR\ (NOT(W_{4,n})\ AND\ W_{6,n})) +\ \ K_n$, 其中

$T1 = (ROTR^{2}(W_{0,n})\ XOR\ ROTR^{13}(W_{0,n})\ XOR$ $ROTR^{22}(W_{0,n})) + ((W_{0,n}\ AND\ W_{1,n})\ XOR\ (W_{0,n}\ AND\ W_{2,n})\ \ XOR\ (W_{1,n}\ AND\ W_{2,n}))$,

其中 AND 是与逻辑连接,其中 NOT 是逐位非,其中 $W_{k,n}$ 是处理周期 n 的第 k 个工作数据块,其中 k_n 是可预先给定的常数,其中

- 函数 F 被定义为 $F=W_{3,n}+T0$,

以及其中

B)在 m=2 的情况下

- 函数 T 被定义为 $T = M_{0,n} + M_{9,n} + (ROTR^{19}(M_{14,n})$ $XOR\ ROTR^{61}(M_{14,n})\ XOR\ \ SHR^{6}(M_{14,n})) + (ROTR^{1}(M_{1,n})\ XOR$ $ROTR^{8}(M_{1,n})\ XOR\ SHR^{7}(M_{1,n}))$,

- 函数 G 被定义为 G=T0+T1，其中 $T0=M_{0,n}+$

$$W_{7,n} + (ROTR^{14}(W_{4,n}) \text{ XOR } ROTR^{18}(W_{4,n}) \text{ XOR } ROTR^{41}(W_{4,n}))$$

$$+ ((W_{4,n} \text{ AND } W_{5,n}) \text{ XOR } (NOT(W_{4,n}) \text{ AND } W_{6,n})) + K_n,$$ 其中

$$T1 = (ROTR^{28}(W_{0,n}) \text{ XOR } ROTR^{34}(W_{0,n}) \text{ XOR } ROTR^{39}(W_{0,n})) +$$

$$((W_{0,n} \text{ AND } W_{1,n}) \text{ XOR } (W_{0,n} \text{ AND } W_{2,n}) \text{ XOR } (W_{1,n} \text{ AND } W_{2,n})),$$

其中

- 函数 F 被定义为 $F = W_{3,n} + T0$。

3. 根据权利要求 1 至 2 之一所述的方法，其中设置有 8 个哈希数据块（H_0，H_1，…，H_7），其中 8 个哈希数据块（H_0，H_1，…，H_7）中的每个都具有 32*m 位的长度，并且其中在（r*N）次执行步骤 c）之后将工作数据块（W_0，W_1，…，W_7）的内容优选逐块地加（229）到哈希数据块（H_0，H_1，…，H_7）的内容上，其中 r 是大于等于 1 的整数。

4. 根据权利要求 3 所述的方法，其中所述加（229）的步骤包括下面的步骤：

d1）将工作数据块 $W_{7,n}$ 和哈希数据块 $H_{7,n}$ 之和指派（229a）给哈希数据块 $H_{0,n+1}$，

d2）对于 I=1 至 7，将哈希数据块 $H_{I-1,n}$ 的值指派（229b）给哈希数据块 $H_{I,n+1}$。

5. 根据上述权利要求之一所述的方法，其中 m=1 和 / 或其中 N=64 和 / 或其中在八个工作数据块（W_0，W_1，…，W_7）的初始化（210）的步骤中进行下面的指派：

$$W_{0,0} = 0x6a09e667, W_{1,0} = 0xbb67ae85, W_{2,0} = 0x3c6ef372, W_{3,0} =$$

$$0xa54ff53a, W_{4,0} = 0x510e527f, W_{5,0} = 0x9b05688c, W_{6,0} = 0x1f83d9ab, W_{7,0}$$

$$= 0x5be0cd19,$$

和 / 或其中八个哈希数据块（H_0，H_1，…，H_7）通过下面的指派被初始化：

$$H_{0,0} = 0x6a09e667, H_{1,0} =$$

$$0xbb67ae85, H_{2,0} = 0x3c6ef372, H_{3,0} = 0xa54ff53a, H_{4,0} = 0x510e527f, H_{5,0} =$$

$$0x9b05688c, H_{6,0} = 0x1f83d9ab, H_{7,0} = 0x5be0cd19。$$

6. 根据权利要求 1 至 4 之一所述的方法，其中 m=2 和 / 或其中 N=80 和 / 或其中在八个工作数据块（W_0，W_1，…，W_7）的初始化（210）的步骤中进行下面的指派：

$$W_{0,0} = \begin{array}{l} 0x6a09e667f3bcc908, W_{1,0} = 0xbb67ae8584caa73b, W_{2,0} = \\ 0x3c6ef372fe94f82b, W_{3,0} = 0xa54ff53a5f1d36f1, W_{4,0} = \\ 0x510e527fade682d1, W_{5,0} = 0x9b05688c2b3e6c1f, W_{6,0} = \end{array}$$

$$0x1f83d9abfb41bd6b, W_{7,0} = 0x5be0cd19137e2179,$$

和 / 或其中八个哈希数据块（H_0，H_1，…，H_7）通过下面的指派被初始化：

$$H_{0,0} = 0x6a09e667f3bcc908, H_{1,0} = 0xbb67ae8584caa73b, H_{2,0} =$$

$0x3c6ef372fe94f82b$, $H_{3,0} = 0xa54ff53a5f1d36f1$, $H_{4,0} = 0x510e527fade682d1$, $H_{5,0} = 0x9b05688c2b3e6c1f$, $H_{6,0} = 0x1f83d9abfb41bd6b$, $H_{7,0} = 0x5be0cd19137e2179$。

7. 根据上述权利要求之一所述的方法，其中第一移位寄存器（SR_M）被用于至少暂时地存储输入数据块（M_0，M_1，…，M_{15}），和／或其中第二移位寄存器（SR_W）被用于至少暂时地存储工作数据块（W_0，W_1，…，W_7），和／或其中第三移位寄存器（SR_H）被用于至少暂时地存储哈希数据块（H_0，H_1，…，H_7）。

8. 根据权利要求 7 所述的方法，其中对于 i=1 至 15 将输入数据块 $M_{i,n}$ 的内容指派（222a）给输入数据块 $M_{i-1,n+1}$ 的步骤包括在第一移位寄存器（SR_M）中优选逐块地对输入数据块 $M_{i,n}$ 的内容进行移位，和／或其中对于 k=0，k=1，k=2 和对于 k=4，k=5，k=6 将工作数据块 $W_{k,n}$ 的内容指派（222b）给工作数据块 $W_{k+1,n+1}$ 的步骤包括在第二移位寄存器（SR_W）中优选逐块地对工作数据块 $W_{k,n}$ 的内容进行移位，和／或其中对于 I=1 至 7 将哈希数据块 $H_{I-1,n}$ 的值指派（229b）给哈希数据块 $H_{I,n+1}$ 的步骤包括在第三移位寄存器（SR_H）中优选逐块地对哈希数据块 $H_{I-1,n}$ 的内容进行移位。

9. 根据权利要求 7 至 8 之一所述的方法，其中在第一运行阶段（BP1）中第一移位寄存器（SR_M）和第二移位寄存器（SR_W）对于 N 个许多时钟周期共同地被时钟控制，以便对第一移位寄存器（SR_M）的内容的优选逐块的移位和第二移位寄存器（SR_W）的内容的优选逐块的移位进行控制，以及其中在接着第一运行阶段（BP1）的第二运行阶段（BP2）中，第二移位寄存器（SR_W）和第三移位寄存器（SR_H）对于 8 个时钟周期共同地被时钟控制，其中优选在第二运行阶段（BP2）期间不对第一移位寄存器（SR_M）进行时钟控制，和／或其中优选在第一运行阶段（BP1）期间不对第三移位寄存器（SR_H）进行时钟控制。

10. 根据权利要求 2 至 9 之一所述的方法，其中

i. 为了确定第一函数 T 的表达式 $ROTR^{17}(M_{14,n})$，$ROTR^{19}(M_{14,n})$，实施下面的步骤：

e1）确定表达式 $V1 = ROTR^{17}(M_{14,n})$，

e1）确定表达式 $V2 = ROTR^{2}(V1)$，以便得到 $ROTR^{19}(M_{14,n})$，

和／或其中

ii. 为了确定第一函数 T 的表达式 $ROTR^{17}(M_{1,n})$，$ROTR^{18}(M_{1,n})$ 实施下面的步骤：

f1）确定表达式 $V3 = ROTR^{7}(M_{1,n})$，

f2）确定表达式 $V4 = ROTR^{11}(V3)$，以便得到 $ROTR^{18}(M_{1,n})$，

和／或其中

iii. 为了确定第二函数 G 的表达式 $ROTR^{2}(W_{0,n})$，$ROTR^{13}(W_{0,n})$，$ROTR^{22}(W_{0,n})$ 实施下面的步骤：

g1）确定表达式 $V5 = ROTR^{2}(W_{0,n})$，

g2）确定表达式 $V6=ROTR^{11}(V5)$，以便得到 $ROTR^{13}(W_{0,n})$，

g3）确定表达式 $V7=ROTR^{9}(V6)$，以便得到 $ROTR^{22}(W_{0,n})$。

11. 一种用于根据数字输入数据（M）生成哈希值（HW）的装置（100；1100），其中该装置

(100)被构造用于实施下面的步骤：

a)将输入数据(M)划分(200)为分别具有 32*m 位长度的十六个输入数据块(M_0,M_1,…,M_{15}),其中 m 是大于等于 1 的整数,并且其中下标变量 i=0,…,15 表示第 i 个输入数据块 M_i,

b)用可预先给定的值初始化(210)八个工作数据块(W_0, W_1, …, W_7),其中八个工作数据块(W_0, W_1, …, W_7)中的每个都具有 32*m 位的长度,并且其中下标变量 k=0,…,7 表示第 k 个工作数据块 W_k,

c)按照如下规则改变(220)输入数据块(M_0,M_1, …,M_{15})和工作数据块(W_0,W_1, …,W_7):

c1)对于 i=1 至 15,将输入数据块 $M_{i,n}$ 的内容指派(222a)给输入数据块 $M_{i-1,n+1}$,其中 n 是大于等于零的整数并且代表处理周期,

c2)对于 k=0, k=1, k=2 以及对于 k=4, k=5, k=6,将工作数据块 $W_{k,n}$ 的内容指派(222b)给工作数据块 $W_{k+1,n+1}$,

c3)将第一函数 T 的输出值指派(224)给输入数据块 $M_{15,n+1}$,

c4)将第二函数 G 的输出值指派(226)给工作数据块 $W_{0,n+1}$,

c5)将第三函数 F 的输出值指派(228)给工作数据块 $W_{4,n+1}$,

其中该装置(100)被构造用于执行改变(220)的步骤 c)N 次,其中 N ＞ 1。

12. 根据权利要求 11 的装置(100),其中该装置(100)被构造用于实施根据权利要求 2 至 10 之一所述的方法。

13. 根据权利要求 11 至 12 之一所述的装置(100 ;1100),其中第一移位寄存器(SR_M)被设置用于至少暂时地存储输入数据块(M_0, M_1, …, M_{15}),和 / 或其中第二移位寄存器(SR_W)被设置用于至少暂时地存储工作数据块(W_0,W_1,…,W_7),和 / 或其中第三移位寄存器(SR_H)被设置用于至少暂时地存储哈希数据块(H_0, H_1, …, H_7)。

14. 根据权利要求 13 的装置(100),其中设置有第一函数块(1110)用于实施第一函数 T,和 / 或其中设置有第二函数块(1120)用于实施第二函数 G,和 / 或其中设置有第三函数块(1130)用于实施第三函数 G,其中优选第一函数块(1110)的输出端(1114)与第一移位寄存器(SR_M)的分配给输入数据块 M_{15} 的输入端(M_{15}E)连接,其中优选第二函数块(1120)的输出端(1124)与第二移位寄存器(SR_W)的分配给工作数据块 W_0 的输入端(W_0E)连接,其中优选第三函数块(1130)的输出端(1134)与第二移位寄存器(SR_W)的分配给工作数据块 W_4 的输入端(W_4E)连接。

15. 根据权利要求 13 至 14 之一所述的装置(100 ;1100),其中设置有加法器(1200),其被构造用于将工作数据块 W_7 的内容与哈希数据块 H_7 的内容相加,其中加法器(1200)的输出端(1204)优选与第三移位寄存器(SR_H)的分配给哈希数据块 H_0 的输入端(H_0E)连接。

16. 根据权利要求 13 至 15 之一的装置(100),其中该装置(100)被构造用于,在第一运行阶段(BP1)中对第一移位寄存器(SR_M)和第二移位寄存器(SR_W)对于 N 个许多时钟周期共同地进行时钟控制,以及在接着第一运行阶段(BP1)的第二运行阶段(BP2)中,对第二移位寄存器(SR_W)和第三移位寄存器(SR_H)对于 8 个时钟周期共同地进行时钟控制,其中优选在第二运行阶段(BP2)期间不对第一移位寄存器(SR_M)进行时钟控制,和 / 或其中优选在第一运行阶段(BP1)期间不对第三移位寄存器(SR_H)进行时钟控制。

17. 根据权利要求 13 至 16 之一所述的装置(100),其中该装置(100)优选完全地被构造为集成电路,尤其是在使用 CMOS 技术情况下来构造。

用于生成哈希值的方法和装置

技术领域

[0001] 本发明涉及用于根据数字输入数据生成哈希值的方法。本发明还涉及用于生成这种哈希值的装置。

背景技术

[0002] 作为输出值提供一个或多个哈希值的哈希函数尤其是被使用在密码学中，特别是针对安全相关的应用例如数字签名、密码存储以及文件的完整性检验等等。广泛流行的密码哈希函数组基于所谓的安全哈希算法版本2（SHA-2）标准，其尤其是在公布“Federal Information Processing Standards Publication，Secure Hash Standard，FIPS PUB 180-3，2008”中得以描述并且可以在因特网中在地址 http://csrc.insit.gov/publications/fips/180-3 下被调用。相应的专利出版物是US 6,829,355B2。

[0003] 一般，密码哈希函数接受任意长度的数字输入数据流并且由此生成所谓的哈希值、也即可预先给定的、尤其是固定的长度的数字输出数据。哈希值有时也被称为数字指纹。

[0004] 哈希值的特别重要的特性在于，哈希函数输入数据的轻微变化已经引起在由此计算的哈希值中的非常大的变化。

[0005] 此外，密码哈希算法可以具有三个特别的特性：

1. 所谓的“抗原像（Preimage Resistance）”，这意味着，必须证明：在有限的、实际可用的计算能力情况下，对于哈希算法的每个可设想的输出值不可能找到所属的输入数据值。

[0006] 2. 所谓的“抗第二原像（Second Preimage Resistance）”，这意味着，在知道由输入数据值和哈希函数的所属输出数据值（哈希值）组成的数据对的情况下实际上不可能找到第二输入数据值，该第二输入数据值通向同一输出数据值、也即哈希值。

[0007] 3. “抗碰撞（Collision Resistance）”，这意味着，实际上不可能找到两个输入数据值，它们通向同一哈希值。

发明内容

[0008] 本发明的任务在于，如下地改善开头所述类型的方法和装置：能够实现更简单并且高效的实施。

[0009] 该任务在开头所述类型的方法中通过如下方式被解决，即该方法具有如下步骤：

a）将输入数据划分为十六个分别具有32*m位长度的输入数据块，其中m是大于等于1的整数，并且其中下标变量i=0，…，15表示第i个输入数据块 M_i，

b）初始化带有可预先给定的值的八个工作数据块，其中这八个工作数据块中的每个都具有32*m位的长度，并且其中下标变量k=0，…，7表示第k个工作数据块 W_k，

c）按照如下规则改变输入数据块和工作数据块：

c1）对于 i=1 至 15，将输入数据块 $M_{i,n}$ 的内容指派给输入数据块 $M_{i-1,n+1}$，其中 n 是大于等于零的整数并且代表处理周期，

c2）对于 k=0，k=1，k=2 以及对于 k=4，k=5，k=6，将工作数据块 $W_{k,n}$ 的内容指派给工作数据块 $W_{k+1,n+1}$，

c3）将第一函数 T 的输出值指派给输入数据块 $M_{15,n+1}$，

c4）将第二函数 G 的输出值指派给工作数据块 $W_{0,n+1}$，

c5）将第三函数 F 的输出值指派给工作数据块 $W_{4,n+1}$，

其中改变的步骤 c）被执行 N 次，其中 $N > 1$。

[0010] 按照本发明认识到，前面定义的用于改变输入数据块和工作数据块的规则能够实现用于生成哈希值的方法的特别高效的技术实施。由此可以特别有利地实现如下实施：所述实施具有比例如基于 US 6,829,355 B2 的已知实施小得多的对等效门（GE，gate equivalents）的需求。

[0011] 借助于本发明的原理特别有利的是此外是如下事实：每个工作周期仅仅必须修改一个输入数据块，并且按照本发明所建议的函数 G、F 仅仅作用于两个工作数据块，也即 $W_{0,n+1}$，$W_{4,n+1}$。

[0012] 在一种优选的实施方式中，可以同时进行将输入数据划分成十六个输入数据块的步骤和初始化八个工作数据块的步骤。代替地，这些步骤也可以相继地或重叠地进行。

[0013] 在一种有利的实施方式中规定：

A）在 m=1 的情况下

- 函数 T 被定义为 $T = M_{0,n} + M_{9,n} + (ROTR^{17}(M_{14,n}) \text{ XOR } ROTR^{19}(M_{14,n}) \text{ XOR } SHR^{10}(M_{14,n})) + (ROTR^{7}(M_{1,n}) \text{ XOR } ROTR^{18}(M_{1,n}) \text{ XOR } SHR^{3}(M_{1,n}))$，其中 $ROTR^{y}(x)$ 是操作数 x 向右逐位旋转 y 个许多位（um y viele Bits），其中 $SHR^{y}(x)$ 是操作数 x 向右逐位逻辑移位 y 个许多位，其中 XOR 是异或逻辑连接，

- 函数 G 被定义为 G=T0+T1，其中

$$T0 = M_{0,n} + W_{7,n} + (ROTR^{6}(W_{4,n}) \text{ XOR } ROTR^{11}(W_{4,n}) \text{ XOR } ROTR^{25}(W_{4,n})) + ((W_{4,n} \text{ AND } W_{5,n}) \text{ XOR } (\text{NOT}(W_{4,n}) \text{ AND } W_{6,n})) + K_n,$$

其中 $T1 = (ROTR^{2}(W_{0,n}) \text{ XOR } ROTR^{13}(W_{0,n}) \text{ XOR } ROTR^{22}(W_{0,n})) + ((W_{0,n} \text{ AND } W_{1,n}) \text{ XOR } (W_{0,n} \text{ AND } W_{2,n}) \text{ XOR } (W_{1,n} \text{ AND } W_{2,n}))$，其中 AND 是与逻辑连接，其中 NOT 是逐位非（Negation），其中 $W_{k,n}$ 是处理周期 n 的第 k 个工作数据块，其中 k_n 是可预先给定的常数，其中

- 函数 F 被定义为 $F=W_{3,n}+T0$。

[0014] 特别有利的是，函数 $ROTR^{y}(x)$，$SHR^{y}(x)$ 以与在 "Federal Information Processing Standards Publication, Secure Hash Standard, FIPS PUB 180-3, 2008" 中一样的方式被定义。

[0015] 在另一有利的实施方式中规定，

B）在 m=2 的情况下

- 函数 T 被定义为 $T = M_{0,n} + M_{9,n} + (ROTR^{19}(M_{14,n})$ $XOR\ ROTR^{61}(M_{14,n})\ XOR\ \ SHR^{6}(M_{14,n})) + (ROTR^{1}(M_{1,n})\ XOR$ $ROTR^{8}(M_{1,n})\ XOR\ SHR^{7}(M_{1,n})),$

- 函数 G 被定义为 G=T0+T1，其中 T0=$M_{0,n}$+

$W_{7,n} + (ROTR^{14}(W_{4,n})\ XOR\ ROTR^{18}(W_{4,n})\ XOR\ ROTR^{41}(W_{4,n})\ )$

$+ ((W_{4,n}\ AND\ W_{5,n})\ XOR\ (NOT(W_{4,n})\ AND\ W_{6,n})) + K_{n}$，其中

$T1 = (ROTR^{28}(W_{0,n})\ XOR\ ROTR^{34}(W_{0,n})\ XOR\ ROTR^{39}(W_{0,n})) +$

$((W_{0,n}\ AND\ W_{1,n})\ XOR\ (W_{0,n}\ AND\ W_{2,n})\ XOR\ (W_{1,n}\ AND\ \ W_{2,n}))$，其中

- 函数 F 被定义为 $F = W_{3,n} + T0$。

[0016] 在前述实施方式的变型方案 A)中，因此将由其生产哈希值的输入数据划分成分别带有 32 位长度的 16 个输入数据块。当前实施方式的变型方案 A）代表用于说明用于生成哈希值的方法的起始点，该方法与 SHA256 类型的 SHA-2 标准兼容，如下面进一步示出的。

[0017] 前述实施方式的变型方案 B）是发明变型方案，它是用于按照 SHA-2 标准 SHA512 类型的哈希值形成的基础。

[0018] 关于对不同输入数据块和工作数据块的移位和旋转操作补充地参照开头所引用的 FIPS 标准。相应的函数在那里被详细地定义并且目前优选地以相同方式被使用。

[0019] 在另一优选的实施方式中设置有 8 个哈希数据块，其中 8 个哈希数据块中的每个均具有 32*m 位的长度，并且其中在（r*N）次执行按照权利要求 1 的步骤 c）之后将工作数据块的内容优选逐块地加到哈希数据块的内容上，其中 r 是大于等于 1 的整数。由此，在相应地执行了步骤 c)之后、也即按照本发明建议的规则改变输入数据块和工作数据块之后迭代地形成哈希值，其被存储在哈希数据块中。在优选的实施方式中，N=64 并且 m=1，使得哈希数据块具有分别为 32 位的长度。

[0020] 只要应当由其形成哈希值的输入数据的长度例如不超过 512 位，按照一种实施方式，将输入数据完全写到输入数据块中并且执行按照本发明的方法就足够了。在 N 次执行所述改变的步骤 c）之后，于是在工作数据块中已经存在可以被用作哈希值的数据。

[0021] 但是对于在使用本发明原理的情况下应当由输入数据（其长于 512 位）来形成哈希值的情况，可以在执行 N 次执行改变的步骤 c)之后首先如上面已经建议的那样将工作数据块的内容复制到哈希数据块中或者加到高哈希数据块上，并且接着可以进行改变的步骤 c）至少一个另外的 N 次实施，使得迭代地在哈希数据块中形成或累加哈希值，所述哈希值与全部的输入数据（大于 512 位）有关。

[0022] 根据另一有利的实施方式，将工作数据块的内容加到哈希数据块的内容上的步骤有利地包括下面的步骤：将工作数据块 $W_{7,n}$ 和哈希数据块 $H_{7,n}$ 之和指派给哈希数据块 $H_{0,n+1}$。

换句话说，当前时钟周期 n 的工作数据块 W_7 的内容和当前工作或时钟周期 n 的哈希数据块 H_7 的内容用作加法器的输入参量，并且其和对于随后的工作循环 n+1 而被指派给哈希数据块 H_0。此外，对于 I=1 至 7，将当前时钟周期 n 的哈希数据块 H_{I-1} 的相应值指派给接着的时钟周期 n+1 的哈希数据块 H_I。

[0023] 在另一有利的实施方式中规定，m=1 和 / 或其中 N=64 和 / 或其中在八个工作数据块的初始化的步骤中进行下面的指派：$W_{0,0} = 0x6a09e667, W_{1,0} = 0xbb67ae85, W_{2,0} = 0x3c6ef372, W_{3,0} = 0xa54ff53a, W_{4,0} = 0x510e527f, W_{5,0} = 0x9b05688c, W_{6,0} = 0x1f83d9ab, W_{7,0} = 0x5be0cd19$，和 / 或其中八个哈希数据块通过下面的指派被初始化：$H_{0,0} = 0x6a09e667$，$H_{0,1}$=0xbb67ae85，$H_{2,0}$ =0x3c6ef372，$H_{3,0}$=0xa54ff53a，$H_{4,0}$= $0x510e527f$，$H_{5,0} = 0x9b05688c$，$H_{6,0} = 0x1f83d9ab$，$H_{7,0} = 0x5be0cd19$。在该实施方式中，本发明方法在哈希值的结果方面相应于 SHA256 类型的 SHA-2 方法。因此（尽管基本上与现有技术不同的计算方法）仍然获得与在 SHA 256 情况下相同的哈希值。

[0024] 该发明变型方案因此特别有利地能够实现对标准化的 SHA 256 方法的完全兼容性，即使同时有利地比在已知装置情况下显著更高效的实施是可能的。在又一有利的实施方式中规定，m=2 和 / 或其中 N=80 和 / 或其中在八个工作数据块的初始化的步骤中进行下面的指派：$W_{0,0} = 0x6a09e667f3bcc908, W_{1,0} = 0xbb67ae8584caa73b, W_{2,0} = 0x3c6ef372fe94f82b, W_{3,0} = 0xa54ff53a5f1d36f1, W_{4,0} = 0x510e527fade682d1, W_{5,0} = 0x9b05688c2b3e6c1f, W_{6,0} = 0x1f83d9abfb41bd6b, W_{7,0} = 0x5be0cd19137e2179$，和 / 或八个哈希数据块通过下面的指派被初始化：

$H_{0,0}$= $0x6a09e667f3bcc908$, $H_{1,0}$ = $0xbb67ae8584caa73b$, $H_{2,0} = 0x3c6ef372fe94f82b$, $H_{3,0} = 0xa54ff53a5f1d36f1$, $H_{4,0} = 0x510e527fade682d1$, $H_{5,0} = 0x9b05688c2b3e6c1f$, $H_{6,0} = 0x1f83d9abfb41bd6b$, $H_{7,0} = 0x5be0cd19137e2179$。

[0025] 在该发明变型方案中有利地给出了与 SHA 512 标准的兼容性，其中又能够实现特别高效的实施，该实施要求比已知系统更少数量的等效门。

[0026] 在又一实施方式中也可以建立对现有标准 SHA224 和 SHA384 的兼容性。为此，代替 SHA256 或 SHA512 中的在前面提到的针对工作数据块和 / 或哈希数据块的初始化值可以使用来自如下文献的章节 5.3.2 (SHA224) 或 5.3.4 (SHA384) 的值：“FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS) CATEGORY:COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8900 March 2012”，其中对于 SHA224 可以选择参数 m=1，并且对于 SHA384 可以选择 m=2。

[0027] 此外，在一种针对 SHA224 兼容性的实施方式中可以规定：仅仅使用八个哈希数据块中的七个作为输出哈希值（7*32 位（m=1）得到 224 位）。

[0028] 此外，在一种针对 SHA384 兼容性的实施方式中可以规定：仅仅使用八个哈希数据块中的六个作为输出哈希值（6*64 位（m=2）得到 384 位）。

[0029] 在另一有利的实施方式中规定，第一移位寄存器被用于至少暂时地存储输入数据块。代替地或补充地，可以使用第二移位寄存器用于至少暂时地存储工作数据块。又代替地或补充地，可以有利地使用第三移位寄存器用于至少暂时地存储哈希数据块。

[0030] 将一个或多个移位寄存器用于存储相应的数据块是特别有利的，因为本发明方法利用函数 T，G 和 F 可以在使用移位寄存器的情况下非常高效地被实施。尤其是也可以由此放弃大量的如在传统实施中所需要的复用器或地址解码器等等，这又明显地降低了本发明的相应电路技术上的实施的复杂度和从而也明显地降低了成本。

[0031] 在另一有利的实施方式中规定，对于 i=1 至 15 将输入数据块 $M_{i,n}$ 的内容指派给输入数据块 $M_{i-1,n+1}$ 的步骤包括在第一移位寄存器中优选逐块地对输入数据块 $M_{i,n}$ 的内容进行移位，和 / 或其中对于 k=0，k=1，k=2 和对于 k=4，k=5，k=6 将工作数据块 $W_{k,n}$ 的内容指派给工作数据块 $W_{k+1,n+1}$ 的步骤包括在第二移位寄存器中优选逐块地对工作数据块 $W_{k,n}$ 的内容进行移位，和 / 或其中对于 I=1 至 7 将哈希数据块 $H_{I-1,n}$ 的值指派给哈希数据块 $H_{I,n+1}$ 的步骤包括在第三移位寄存器中优选逐块地对哈希数据块 $H_{I-1,n}$ 的内容进行移位。

[0032] 特别有利地在另一实施方式中规定，在第一运行阶段中第一移位寄存器和第二移位寄存器对于 N 个多个时钟周期共同地被时钟控制，以便对第一移位寄存器的内容的优选逐块的移位和第二移位寄存器的内容的优选逐块的移位进行控制。在该第一运行阶段期间，只要设置第三移位寄存器用于存储哈希数据块，该第三移位寄存器已经不必被时钟控制。在优选直接接着第一运行阶段的第二运行阶段中，规定：第二移位寄存器和第三移位寄存器对于 8 个时钟周期共同地被时钟控制（getaktet），其中优选在第二运行阶段期间不对第一移位寄存器进行时钟控制。

[0033] 通过这种方式能够实现按照本发明可使用的移位寄存器的特别高效和节省能量的运行。

[0034] 在另一有利的实施方式中规定：

i. 为了确定第一函数 T 的表达式 $ROTR^{17}(M_{14,n})$，$ROTR^{19}(M_{14,n})$，实施下面的步骤：

e1）确定表达式 $V1 = ROTR^{17}(M_{14,n})$，

e1）确定表达式 $V2 = ROTR^{2}(V1)$，以便得到 $ROTR^{19}(M_{14,n})$，

和 / 或其中

ii. 为了确定第一函数 T 的表达式 $ROTR^{7}(M_{1,n})$，$ROTR^{18}(M_{1,n})$ 实施下面的步骤：

f1）确定表达式 $V3 = ROTR^{7}(M_{1,n})$，

f2）确定表达式 $V4 = ROTR^{11}(V3)$，以便得到 $ROTR^{18}(M_{1,n})$，

和 / 或其中

iii. 为了确定第二函数 G 的表达式 $ROTR^{2}(W_{0,n})$，$ROTR^{13}(W_{0,n})$，$ROTR^{22}(W_{0,n})$ 实施下面的步骤：

g1）确定表达式 $V5 = ROTR^{2}(W_{0,n})$，

g2）确定表达式 $V6=ROTR^{11}(V5)$，以便得到 $ROTR^{13}(W_{0,n})$，

g3）确定表达式 V7=ROTR^{9}(V6)，以便得到 ROTR^{22}($W_{0,n}$)。

[0035] 前面的计算准则能够特别高效地确定用于分析按照本发明所建议的函数的相应项并且避免同样的表达式的不必要的多重计算。

[0036] 作为本发明任务的另外的解决方案说明了按照权利要求 11 的装置。其他的有利的扩展方案是从属权利要求的主题。

附图说明

[0037] 下面参照附图阐述本发明示例性实施方式。

[0038] 在附图中：

图 1 示意性示出了用于表明哈希值的应用的框图，

图 2a 示意性示出了一种实施方式的简化的流程图，

图 2b 示意性示出了按照另一实施方式的流程图，

图 2c 示意性示出了按照另一实施方式的流程图，

图 3 示意性示出了按照一种实施方式的装置的框图，

图 4 示意性示出了按照一种实施方式的输入数据块和工作数据块，

图 5a，b，c 分别示意性示出了在不同时钟周期或工作周期中按照一种实施方式的输入数据块和工作数据块，

图 6 示意性示出了本发明装置的另一实施方式的框图，以及

图 7 示意性示出了按照一种实施方式的用于表明不同运行阶段的时间图。

具体实施方式

[0039] 图 1 示意性示出了在其中根据第一消息 MSG1 在使用哈希算法情况下形成第一哈希值 HW1 的情景。第一消息 MSG1 可以是任意长度的数字数据，它们例如作为位序列（英文 bit string（位串））而存在。哈希值的形成在图 1 中象征性地通过箭头 A1 来表明。

[0040] 另外的、这次利用第二输入数据 MSG2 然而在使用同样的哈希算法的情况下的哈希值形成在图 1 中用箭头 A2 表示，所述第二输入数据与第一输入数据 MSG1 不同。结果得到第二哈希值 HW2。典型地，只要输入数据 MSG1、MSG2 彼此不同，哈希值 HW2 明显地与哈希值 HW1 不同，尤其是当输入数据 MSG1、MSG2 仅仅轻微地例如在位单元（Bitstelle）方面彼此不同时，哈希值 HW2 就已经明显地与哈希值 HW1 不同了。换句话说，通常通过哈希值形成 A1、A2 将在不同输入数据 MSG1、MSG2 之间的汉明距离转化成相应哈希值 HW1、HW2 的明显增大的汉明距离。

[0041] 图 2a 示出了用于生成哈希值的本发明方法的第一实施方式的简化的流程图。在第一步骤 200 中将应当由其形成哈希值的输入数据划分为 16 个分别带有 32*m 位长度的输入数据块，其中 m 是大于等于 1 的整数。在特别优选的实施方式中 m=1，使得 16 个输入数据块相应于 16 个 32 位数据字。换句话说，例如 m=2 等等同样是可以设想的。

[0042] 在接下来的步骤 210 中（参见图 2a），利用可预先给定的值来初始化 8 个工作数据块。类似于输入数据块，8 个工作数据块也分别具有 32*m 位的长度，也即目前分别为 32 位。

[0043] 在一个优选的实施方式中，也可以同时地进行划分 200 输入数据的步骤和初始化 210 八个工作数据块的步骤。代替地也可以相继地或者重叠地进行这些步骤。

[0044] 在接下来的步骤 220 中按照下面详细描述的规则来修改输入数据块或输入数据块至少之一和工作数据块或工作数据块至少之一，以便生成哈希值。

[0045] 图 3 为此示出了本发明装置 100 的一种实施方式的框图。

[0046] 装置 100 在输入侧获得消息 MSG 并且根据该消息 MSG 在使用本发明方法的情况下形成哈希值 HW，该装置在其输出端处输出该哈希值。该哈希值形成示例性地在处理单元 110 中进行，该处理单元被构造用于实施本发明方法。

[0047] 可选地，该装置 100 也可以拥有在图 3 中通过虚线矩形表示的数据划分单元 120，在将消息 MSG 以数字输入数据 M 的形式输送给装置 110 之前，所述数据划分单元调整（konditionieren）该消息 MSG。

[0048] 例如前面描述的、按照本发明实施方式的 16 个输入数据块可以总共接收 512 位数据，其中参数 m 可以被选择为 m=1。只要应当由其形成哈希值 HW 的消息 MSG 恰恰具有 512 位，消息 MSG 可以直接作为数字输入数据 M 被输送给装置 110 用于哈希值形成。

[0049] 只要消息 MSG 具有小于 512 位的长度，就例如可以设想，消息 MSG 的长度适配于 512 位的参照长度，其方式是以预先定义的方式填充位单元，尤其是借助填补（Padding）。所述填补例如可以涉及在消息 MSG 的开头处或在结尾处的可预先给定的位序列的增补（Anhängen）。在该情况下也即在使用填补的情况下从消息 MSG 中获得输入数据 M，其中填补例如可以通过单元 120 实施。

[0050] 只要消息 MSG 具有大于 512 位的长度，本发明方法同样是可应用的，其中在该情况下优选首先将消息 MSG 以块的方式分别分解成 512 位 + 必要时长度小于 512 位的剩余数据块，它们逐步地被输送给装置 110 用于哈希值形成。

[0051] 图 4 示例性示出了按照一种实施方式的十六个输入数据块 $M_0, M_1, \cdots, M_{15}$。在具有参数 m=1 的该发明变型方案中，每个输入数据块 $M_0, M_1, \cdots, M_{15}$ 具有 32 位的大小。所有输入数据块 $M_0, M_1, \cdots, M_{15}$ 因此一起得到 512 位，如前面已经描述的。与此对应地，在具有参数 m=2 的另一本发明实施方式中，每个输入数据块 $M_0, M_1, \cdots, M_{15}$ 具有 64 位的大小，其中所有输入数据块 $M_0, M_1, \cdots, M_{15}$ 因此一起得到 1024 位。图 4 同样示意性示出了八个工作数据块 W_0 至 W_7，如针对本发明方法的实施可使用的。只要输入数据块 $M_0, M_1, \cdots, M_{15}$ 具有 32 位的位宽，这同样优选适用于工作数据块 W_0 至 W_7。

[0052] 针对后面的描述认为的是，哈希值形成所基于的消息 MSG（图 3）具有恰恰 512 位的长度。在该情况下，消息 MSG 直接作为数字输入数据 M 被写到按照图 4 的输入数据块 M_0 至 M_{15} 中，由此这些输入数据块被初始化。

[0053] 工作数据块 W_0 至 W_7 同样可以在使用可预先给定的值的情况下被初始化。在另一实施方式中，这却不是需要的或者可以借助零值或随机值等等进行初始化。

[0054] 但是特别优选地，对于工作块 W_0 至 W_7 的初始化在另一实施方式中使用下面的值用于初始化：

$$W_{0,0} = 0x6a09e667,\ W_{1,0} = 0xbb67ae85,\ W_{2,0} = 0x3c6ef372,\ W_{3,0} = 0xa54ff53a,$$

$$W_{4,0} = 0x510e527f,\ W_{5,0} = 0x9b05688c,\ W_{6,0} = 0x1f83d9ab,\ W_{7,0} = 0x5be0cd19\ 。$$

[0055] 在前面的标记法中，前缀“0x”意味着，用于工作数据块的初始化值是十六进制数。第一下标说明，是这八个工作数据块中的哪个，并且第二下标说明用于实施哈希值形成的

工作周期。也即例如对于第 0 个工作周期(n=0)工作数据块 W_0 利用十六进制数 6a09E667 来初始化($W_{0,0}$=0x6a09e667),等等。

[0056] 在初始化之后,得到图 5a 中示意性描绘的状态,其相应于工作周期 n=0。输入数据块 M_0 至 M_{15} 具有其初始化值 $M_{0,0}$ 至 $M_{15,0}$,所述初始化值直接相应于 512 位(根据输入数据块被分组成各 32 位的 16 个块)。

[0057] 工作数据块 W_0 至 W_7 例如利用其初始化值 $M_{0,0}$ 至 $M_{7,0}$ 按照前面的实施来初始化。

[0058] 在相应于第零个工作周期或时钟周期、也即 n=0 (参见图 5a)的初始化之后,实施本发明方法。下面参照图 2b 来描述方法变型方案。在第一步骤 222a 中,对于 i=1 至 15 将输入数据块 $M_{i,n}$ 的内容指派给输入数据块 $M_{i-1,n+1}$。这意味着 :在接着的工作周期 n=1 中 15 个输入数据块 M_0 至 M_{14} 分别获得来自当前工作周期 n=0 的输入数据块 M_1 至 M_{15} 的内容作为值指派。该状态在图 5b 中示意性地被描绘。例如,输入数据块 M_0 现在、也即在周期 n=1 中包含相应于来自在前工作周期 n=0 的工作数据块 M_1 的内容的值 $M_{1,0}$ 作为内容,等等。

[0059] 指派的按照发明的步骤 222a (图 2b)因此相应于对来自第一周期 n=0 的 15 个输入数据块 $M_{1,0}$ 至 $M_{15,0}$ 的内容进行移位,这在本发明的技术实施时可以特别有利地在使用移位寄存器的情况下来实现。例如使用第一移位寄存器用于至少暂时地存储输入数据块 M_0 至 M_{15},其中该移位寄存器总共具有各 32 位的 16 个块。按照本发明的移位操作(该移位操作是图 2b 的步骤 222a 的主题)可以有利地例如通过对所涉及的输入数据块的逐块的移位来完成。

[0060] 在另一步骤 222b (图 2b)中,该步骤优选也可以与步骤 222a 同时被实施,对于 k=0,1,2 和对于 k=4,5,6,工作数据块 $W_{k,n}$ 的内容被指派给工作数据块 $W_{k+1,n+1}$。换句话说,也即工作数据块 W_0, W_1, W_2 的内容从在工作周期 n=0 时的初始化状态(图 5a)开始被指派给接着的工作周期 n=1 (图 5b)的工作数据块 W_1, W_2, W_3。也对于工作数据块 W_4, W_5, W_6 的内容得到类似的移位。只要使用第二移位寄存器用于至少暂时地存储工作数据块 W_k,可以优选与根据步骤 222a 的移位操作同时或同步地实施与本发明方法步骤 222b 相对应的移位操作,使得可以针对所涉及的移位寄存器使用相同的操控信号。

[0061] 如从在第一工作周期 n=0 (图 5a)时的输入数据块 M_0 至 M_{15} 和工作数据块 W_0 至 W_7 与按照接着的工作周期 n=1 (图 5b)的输入数据块 M_0 至 M_{15} 和工作数据块 W_0 至 W_7 的比较得出的,占优势数量的工作数据块或输入数据块或其内容(也即当前相应的 32 位值)在本发明步骤 222a、222b (图 2b)的范畴中仅仅在工作数据块 W 或输入数据块 M 内部移位。这可以借助移位寄存器特别高效地实施。

[0062] 仅仅输入数据块 M_{15} 以及工作数据块 W_0, W_4 不是通过移位操作而是通过分析按照本发明设置的函数 T、G、F 来获得其新的(在工作周期 n=1 时存在的)内容。

[0063] 对于工作周期 n=0,函数 T 的输出值因而被指派给按照图 5b 的输入数据块 M_{15},也即 $M_{15,1}=T_0$,并且函数 G,F 的相应输出值又分别以在第一工作周期 n=0 被分析的方式而被指派给工作数据块 W_0,W_4…,也即 $W_{0,1}=G_0$ 以及 $W_{4,1}=F_0$,以便对于工作周期 n=1 (图 5b)获得数据块 M_{15}, W_0, W_4 的相应的值。

[0064] 函数 T、G、F 的函数值至相应数据块的指派在根据图 2b 的流程图中在步骤 224(函数 T)、226 (函数 G)和 228 (函数 F)中进行。这些步骤中的两个或多个优选也可以并行地实施,由此相应地缩短了用于生成哈希值 HW (图 3)的总的处理时间。

[0065] 特别优选地，在一种实施方式中 N 次实施前面参照图 2b 和步骤 222a 至 228 所描述的方法流程，其中 N 大于 1，由此确保按照本发明所获得的哈希值 HW 满足对哈希值提出的密码安全要求。

[0066] 在另一有利的实施方式中，尤其是在 N 次实施按照图 2b 的步骤 222a 至 228 之后，于是存在于工作数据块 W_0 至 W_7 中的值 $W_{0,N-1}$ 至 $W_{7,N-1}$ 在必要时存在的哈希数据块 H_0 至 H_7（在 m=1 时八个哈希数据块 H_0 至 H_7 中的每个也相应地具有 32 位数据宽度）中可以被移位或者被加到包含于其中的值上，参见图 2b 的步骤 229。如果在按照图 2b 的步骤 222a 至 228 的第一 N 次实施中处理包含 512 位的第一块输入数据 M，以及如果在按照图 2b 的步骤 222a 至 228 的至少另一 N 次实施中处理包含 512 位的第二块输入数据 M，则这例如是合乎目的，如果哈希值形成基于的消息 MSG（图 3）具有 1024 位，则这例如是有意义的。通过这种方式，因此能够在哈希数据块 H_0 至 H_7 中迭代地获得哈希值，这些哈希值与输入数据 M 的多个块相关。只要哈希值形成基于的消息 MSG 具有 512 位或者更少的长度，则也可能的是，直接从工作数据块 W_0 至 W_7 中提取哈希值 HW。在该情况下，因此不需要哈希数据块。

[0067] 图 5c 示出了在工作周期 n=2 结束时输入数据块 M_0 至 M_{15} 和工作数据块 W_0 至 W_7 的内容。图 5b 和图 5c 的比较得出，输入数据块或工作数据块的内容的大部分相对于在前的工作周期 n=1（图 5b）又被移位了。例如来自图 5b 的输入数据块 M_{15} 的内容 T_0 被指派给来自图 5c 的输入数据块 M_{14}。类似的适用于按照图 5c 的工作数据块 W_5、W_1 的内容。

[0068] 通过重新分析本发明的函数 T（这次基于工作周期 n=1 的输入值）将相应的函数值 T_1 仅仅指派给了输入数据块 M_{15}，也即 $M_{15,2}=T_1$。同样的也适用于工作数据块 W_0，W_4，将新的函数值 G_1，F_1 指派给所述工作数据块，也即 $W_{0,2}=G_1$ 以及 $W_{4,2}=F_1$。

[0069] 在一种特别优选的实施方式中，步骤 222a 至 228 的本发明方法流程被重复 N=64 次，这在图 2b 中通过从步骤 228 至步骤 222a 的虚线箭头表示。在此情况下，得出特别好的哈希值，其在按照本发明所描述的方法流程的结束处包含在工作数据块 W_0 至 W_7 或相应的移位寄存器中。该哈希值可以直接被用作本发明装置 100（图 3）的输出值 HW。

[0070] 在一种特别优选的实施方式中，其中选择 m=1，对于本发明函数 T、G、F 的定义得到下面的：

$$T = M_{0,n} + M_{9,n} + (ROTR^{17}(M_{14,n}) \text{ XOR } ROTR^{19}(M_{14,n}) \text{ XOR } SHR^{10}(M_{14,n})) + (ROTR^{7}(M_{1,n}) \text{ XOR } ROTR^{18}(M_{1,n}) \text{ XOR } SHR^{3}(M_{1,n})), \quad [\text{等式 1}]$$

其中 $ROTR^{y}(x)$ 是操作数 x 向右逐位旋转 y 个许多位，其中 $SHR^{y}(x)$ 是操作数 x 向右逐位逻辑移位 y 个许多位，其中 XOR 是异或逻辑连接，

G=T0+T1，[等式 2]，

其中

$$T0 = M_{0,n} + W_{7,n} + (ROTR^{6}(W_{4,n}) \text{ XOR } ROTR^{11}(W_{4,n}) \text{ XOR } ROTR^{25}(W_{4,n})) + ((W_{4,n} \text{ AND } W_{5,n}) \text{ XOR } (\text{NOT}(W_{4,n}) \text{ AND } W_{6,n})) + K_n, \quad [\text{等式 3}]$$

其中

$$T1 = (ROTR^{2}(W_{0,n}) \text{ XOR } ROTR^{13}(W_{0,n}) \text{ XOR } ROTR^{22}(W_{0,n})) + ((W_{0,n} \text{ AND }$$

$W_{1,n}$) XOR ($W_{0,n}$ AND $W_{2,n}$) XOR ($W_{1,n}$ AND $W_{2,n}$)), [等式 4]

其中 AND 是逐位与逻辑连接，其中 NOT 是逐位非，其中 $W_{k,n}$ 是处理周期 n 的第 k 个工作数据块，其中 k_n 是可预先给定的常数(优选对于不同工作周期 n 分别说明用于常数 K_n 的另外的值)，其中函数 F 被定义为

$F=W_{3,n}+T0$，[等式 5]。

[0071] 应当注意：按照等式 3 和等式 4 的值 T0、T1 是用于计算函数 G、F 的辅助参量，并且尤其是按照等式 3 的辅助参量 T0 不应与参量 $T_{n=0}$ (也即简写为 T_0) 混淆，其中该参量 T_0 表示在工作周期 n=0 时函数 T (等式 1) 的输出值。

[0072] 因此为了例如对于工作周期 n=0 (初始化状态)确定函数 T 的输出值 T_0，输入数据块 $M_{0,0}$, $M_{9,0}$, $M_{14,0}$, $M_{1,0}$ 的内容作为输入参量被输送给函数 T 并且经历相应的移位或旋转操作。作为按照前述定义的各个表达式之和得出对于工作周期 n=0 函数 T 的输出值 T_0。该值按照本发明被指派给紧接着其的工作周期(这里：n=1，参见图 5b)的输入数据块 M_{15}，也即 $M_{15,1}=T_0$。类似地进行用于确定函数 F、G 的值。

[0073] 图 6 示意性示出了本发明装置 1100 的另一实施方式的框图，该装置例如作为集成电路能够实现特别高效的实施。所描述的实施方式能够实现快速并且同时能量高效的哈希值形成并且为了实现仅仅需要特别少数量的等效门(GE，等效门)。尽管如此仍然能够有利地实现对 SHA-2 标准例如 SHA 256 类型的完全兼容性。

[0074] 为了至少暂时存储 16 个输入数据块 M_0 至 M_{15}，在按照图 6 的装置 1100 情况下设置第一移位寄存器 SR_M，其相应地具有分别带有 32 位存储宽度(目前选择 m=1) 的 16 个块。该第一移位寄存器 SR_M 在利用图 6 中未描绘的控制信号相应操控的情况下有利地能够实现输入数据块 M_0 至 M_{15} 的内容在各 32 位的块中、也即相应输入数据块的单元中逐块移位。这意味着，在移位操作之后例如块 M_{15} 的内容移位到块 M_{14} 中。这在图 6 中通过在移位寄存器 SR_M 的下部区域中的未更详细标出的弧形箭头来表明，其从移位寄存器 SR_M 的确定的数据块指向移位寄存器 SR_M 的分别右边相邻的数据块。

[0075] 设置有第二移位寄存器 SR_W 用于至少暂时地存储工作数据块 W_0 至 W_7。第二移位寄存器 SR_W 与此相应地具有位宽分别为 32 位(目前选择 m=1)的总共 8 个数据块。在该第二移位寄存器 SR_W 情况下移位操作以与第一移位寄存器 SR_M 相应的方式、也即借助通过未在图 6 中描绘的操控信号的相应操控来进行。

[0076] 图 6 此外还示出第三移位寄存器 SR_H，其被设置用于至少暂时存储哈希数据块 H_0 至 H_7。第三移位寄存器 SR_H 与此相应地也具有位宽分别为 32 位的 8 个数据块(目前选择 m=1) 并且因此在八个各 32 位的数据块方面基本上(i.w.) 与第二移位寄存器 SR_W 结构相同。

[0077] 用于前面描述的移位寄存器的操控信号可以通过装置 1100 的控制单元(未示出)来产生，其例如以状态自动机的形式或也借助 ASIC 和 / 或 FPGA 等等来实现。

[0078] 装置 1100 还拥有第一函数块 1110，其被设置用于实施第一函数 T。为此，第一函数块 1110 拥有输入端 1112，通过其能将涉及的输入数据输送给第一函数块 1110。在选择参数 m=1 的情况下，这例如是输入数据块 M_0, M_1, M_9, M_{14} 的内容。在图 6 中通过指向输入端 1112 的箭头来象征性表示将相应输入数据输送给函数块 1110 的输入端 1112。在电路技术的实施中，相应输入数据至第一函数块 1110 的输送例如可以通过如下方式来实现，即第一

移位寄存器 SR_M 具有并行的输出端，这些输出端被分配给输入数据块 M_0，M_1，M_9，M_{14}，使得这些输入数据块的内容可以被输送给第一函数块 1110 的输入端 1112。为此优选考虑输入数据块 M_0，M_1，M_9，M_{14} 至组件 1112 的固定布线，其相对于如在已知的 SHA-2 实施中所需的带有复用器的寻址逻辑来说具有非常小的电路技术复杂度。

[0079] 第一函数块 1110 与此相应地分析第一函数 T 并且在其输出端 1114 处输出相应的函数值，该函数值被输送给移位寄存器 SR_M，确切地说在那里输送给相应于输入数据块 M_{15} 的那个数据块。由此实现前面描述的将第一函数 T 的输出值指派给输入数据块 M_{15} 的步骤 c3)。为此，第一函数单元 1110 的输出端 1114 优选直接与输入数据块 M_{15} 的优选并行的输入端 $M_{15}E$ 连接。在电路技术上，这例如可以通过第一函数单元 1110 的输出端 1114 至输入数据块 M_{15} 的输入端 $M_{15}E$ 的 32 位宽并行数据总线来实现。

[0080] 以类似方式也可以将输入数据(输入数据块 M_0，M_1，M_9，M_{14})经由并行的数据总线从第一移位寄存器 SR_M 引导给第一函数单元 1110 的输入端 1112。

[0081] 函数 T 或者函数块 1110 的在第 n 工作周期中基于输入数据 $M_{0,n}$，$M_{1,n}$，$M_{9,n}$，$M_{14,n}$ 构成的输出值用 T_n 表示，也参见图 5a 至 5c。第 n 工作周期的输出值 T_n 于是例如被指派给紧接着的工作周期 n+1 的数据块 $M_{15,n+1}$。

[0082] 有利地，根据图 6 的实施方式的组件 1114，$M_{15}E$ 的连接也能够实现固定布线，使得这里也不需要耗费的复用器等，由此用于装置 1100 的实施的门数量是很小的。

[0083] 同样，在图 6 中描绘出第二函数单元 1120，其被设置用于实施本发明函数 G。函数单元 1120 在其在图 6 中未更详细表示的输入端处接收为了分析函数 G 所需的输入数据、尤其是工作数据块 W_0，W_1，W_2，W_4，W_5，W_6，W_7 以及输入数据块 M_0 的内容和必要时与工作周期有关的常数 K_n，它们不是由移位寄存器 SR_W 或者 SR_M 提供，而是通过单独的数据源(未示出，例如 ROM，其他寄存器(RAM)等等)提供。

[0084] 根据这些输入数据，第二函数单元 1120 分析本发明的第二函数 G 并且在输出端 1124 处输出函数 G 的相应输出值(对于工作周期 n 的 G_n)。在特别优选的实施方式中，该输出值直接被输送给第二移位寄存器 SR_W 的与第一工作数据块 W_0 相对应的数据块。为此，优选在第二函数单元 1120 的输出端 1124 和第二移位寄存器 SR_W 的所涉及的数据块 W_0 的输入端 M_0E 之间设置有直接的数据连接，其例如可以以 32 位宽并行数据总线形式来构造。

[0085] 第三函数单元 1130 同样在图 6 中被描绘。第三函数单元 1130 用于根据输送给其的输入数据 W_3，T0 (参见前面描述的定义)分析本发明函数 F。在其输出端 1134 处，第三函数单元 1130 输出相应于函数 F 的输出值，所述输出值被指派给第二移位寄存器 SR_W 的工作数据块 W_4。为此，优选在第三函数单元 1130 的输出端 1134 和第二移位寄存器 SR_W 的所涉及的数据块 W_4 的输入端 M_4E 之间设置直接的数据连接，其例如可以以 32 位宽并行数据总线形式来构造。应当注意，为了分析本发明函数 G、F 对于每个工作周期 n 仅仅必须分析按照等式 3 的计算值 T0 一次。两个函数单元 1120、1130 (和 / 或与组件 1110)相互的函数整合在一种实施方式中与此相应地也是可能的。

[0086] 在图 6 中描绘的结构能够有利地实现前面参照图 2b 描述的方法的实施，其中按照本发明规定的指派操作有利地依靠移位寄存器 SR_M, SR_W 来实现。

[0087] 在特别优选的实施方式中，在第一步骤中将输入数据块 M_0 至 M_{15} (图 4)利用消息 MSG 或者数字输入数据 M 初始化，也参见图 2a 的步骤 200，其中这些数字输入数据至少构成

消息 MSG 的一部分(图 3)。工作数据块 W_0 至 W_7 根据一种实施方式优选在使用下面的指派的情况下被初始化：

$$W_{0,0} = 0x6a09e667, W_{1,0} = 0xbb67ae85, W_{2,0} = 0x3c6ef372, W_{3,0} = 0xa54ff53a,$$

$$W_{4,0} = 0x510e527f, W_{5,0} = 0x9b05688c, W_{6,0} = 0x1f83d9ab, W_{7,0} = 0x5be0cd19,$$

参见图 2a 的步骤 210。

[0088] 在一种特别优选的实施方式中，同样利用前面所述的值对哈希数据块 H_0 至 H_7（图 6）初始化：

$$H_{0,0} = 0x6a09e667, H_{1,0} = 0xbb67ae85, H_{2,0} = 0x3c6ef372, H_{3,0} = 0xa54ff53a, H_{4,0} = 0x510e527f, H_{5,0} = 0x9b05688c, H_{6,0} = 0x1f83d9ab, H_{7,0} = 0x5be0cd19$$

。

[0089] 在一种优选的实施方式中，装置 1100 除了前面参照图 6 所描述的组件之外还拥有加法器 1200，其作为输入参量获得工作数据块 W_7 的内容和哈希数据块 H_7 的内容。加法器 1200 与此相应地实施 32 位加法(在 m=1 时；对于 m=2 可以将加法器 1200 构造为 64 位加法器)并且在其输出端 1204 处输出相应的和值，其被指派给哈希数据块 H_0。这在电路技术上例如可以通过在输出端 1204 和哈希数据块 H_0 的输入端 H_0E 之间的直接数据连接来实现，例如以 32 位(m=1)的数据总线的形式来实现，从而这里也不需要用于地址或数据选择的耗费的复用器，如在已知的哈希装置时的情况那样。

[0090] 为了根据数字输入数据生成哈希值，其中所述数字输入数据在初始化过程的范畴中被写到输入数据块 M_0 至 M_{15} 中，按照一种特别优选的实施方式，实施下面描述的方法。

[0091] 从初始化状态出发(工作周期 n=0，工作数据块和哈希数据块如前述地利用当前作为十六进制数字说明的非极小值来初始化)来实现输入数据块 M_0 至 M_{15} 的改变，当前按照根据图 2b 的方法流程借助移位寄存器 SR_M 或者其控制或时钟控制(Taktung)来实施。

[0092] 与此类似地也按照图 2b 的方法流程来修改工作数据块 W_0 至 W_7。

[0093] 按照图 2b 的步骤 222a、222b、224、226、228 的流程在一种特别优选的实施方式中被重复 N=64 次。由此，定义了按照图 6 的装置 1100 的第一运行阶段 BP1。该第一运行阶段在图 7 的时间图中示意性示出。在第一运行阶段 BP1 期间因此对两个移位寄存器 SR_M，SR_W 如此进行时钟控制，使得它们从工作周期 n 到接着的工作周期 n+1，n=0，…，63 分别实现方法步骤 222a、222b (图 2b)的按照本发明的指派。同样，在每个工作周期中通过函数块 1110、1120、1130 (图 6)(根据图 2b 的步骤 224、226、228)分析函数 T、G、F，由此对于所涉及的第 n 个工作周期获得相应的函数值 T_n，G_n，F_n。

[0094] 在按照图 2b 的方法流程的第六十四次实施(相应于 n=63)之后，第二移位寄存器 SR_W 在其工作数据块 W_0 至 W_7 中包含数据 $W_{0,63}$，…，$W_{7,63}$，它们已经可以有利地被用作哈希值 HW (图 3)。只要消息 MSG 具有恰恰 512 位的长度，因此数据 M 相应于整个消息 MSG，本发明方法可以在该部位处被中断并且将第二移位寄存器 SR_W 的内容作为哈希值 HW 使用。

[0095] 但是只要哈希值形成基于的消息 MSG (图 3)具有大于 512 位的长度，就在 64 次实施按照图 2b 的方法流程之后首先开始第二运行阶段 BP2 (图 7)，其被用于将第二移位寄存器 SR_W (图 6)的当前内容加到第三移位寄存器 SR_H 的当前内容上。这可以理解为宽度分别为 256 位的两个数字值的加法(在 m=1 的情况下；对于 m=2，这是 512 位加法)。但是与真正的 256 位加法(或者对于 m=2 而言 512 位加法)不同，在一种实施方式中对于每个 32 位

块(m=1)或者对于每个64位块(m=2)优选逐块地进行加法,更确切地说优选在相邻块之间没有传输的情况下进行。就此而言存在与真正的256位宽数据字的加法的区别。因为在一种优选的实施方式中,在在前的64个工作周期(n=0,…,63)中第三移位寄存器SR_H已经不被时钟控制,因此在哈希数据块H_0至H_7中一如既往地存在初始化值。与此相应地,通过将第二移位寄存器SR_W的内容按照本发明加到第三移位寄存器SR_H的内容上得到如在第64个工作周期之后在第二移位寄存器SR_W中存在的"暂时哈希值"至第三移位寄存器SR_H的初始化值的加法。

[0096] 在一种特别优选的实施方式中,在使用加法器1200以及对移位寄存器SR_W,SR_H进行八次时钟控制的情况下进行第二移位寄存器SR_W的内容至第三移位寄存器SR_H的内容的加法,由此定义第二运行阶段BP2(图7)。

[0097] 在该第二运行阶段BP2期间不对第一移位寄存器SR_M进行时钟控制,由此减少电能量消耗。移位寄存器的时钟控制在一种实施方式中可以如此来进行,即在第一运行阶段中将第一移位使能信号(shift enable signal)SE1输送给移位寄存器SR_M,SR_W,该第一移位使能信号引起这些移位寄存器SR_M,SR_W的同步时钟,并且在第二运行阶段中将第二移位使能信号SE2输送给移位寄存器SR_W,SR_H,该第二移位使能信号引起这些移位寄存器SR_W,SR_H的同步时钟。

[0098] 接着,下面更详细地描述第二移位寄存器SR_W的内容至第三移位寄存器SR_H的内容的加法。

[0099] 在第二运行阶段BP2(图7)的开始,工作数据块W_7和哈希数据块H_7的内容被输送给加法器1200,所述加法器实施32位加法并且在其输出端1204处输出相应的和值。该和值对于总共八个被用于加法的工作周期中的下一个工作周期被指派给哈希数据块H_0。另外的哈希数据块H_1至H_7针对接着的工作周期通过哈希数据块H_0至H_6的移位得到其新的内容。这意味着,在第二运行阶段BP2的周期的第一次完成之后哈希数据块H_7具有来自在前周期的哈希数据块H_6的内容,并且哈希数据块H_6具有来自在前周期的哈希数据块的内容等等。结果,因此在寄存器SR_W,SR_H的加法过程的周期的第一次完成之后哈希数据块H_0具有由在前周期的数据块W_7,H_7的内容构成的和值,并且其他哈希数据块H_1至H_7包含来自在前周期的哈希数据块H_0至H_6的较早的值。

[0100] 例如也可以包括对移位寄存器SR_W,SR_H进行同步时钟控制的该过程总共重复八次,使得有效地进行了移位寄存器SR_W,SR_H的内容的"256位加法",其结果现在以哈希数据块H_0至H_7的形式存在于第三移位寄存器SR_H中。但是与真正的256位加法(或者对于m=2而言512位加法)不同,在一种实施方式中对于每个32位块(m=1)或者对于每个64位块(m=2)优选逐块地进行加法,更确切地说优选在相邻块之间没有传输(英文"carry(传送)")的情况下进行。就此而言存在与真正256位宽数据字的加法的区别。

[0101] 第二移位寄存器SR_W的内容在第二运行阶段BP2(图7)的结束处相对于第二运行阶段BP2的开始没有变化,因为第二移位寄存器SR_W具有八个32位宽数据块的总数量并且八次移位又引向第二移位寄存器SR_W在第二运行阶段BP2开始时的初始状态。

[0102] 只要对于第二运行阶段BP2引入另外的时标 v,该时标与第一运行阶段的时标或工作周期指标n不同,那么移位寄存器SR_W,SR_H的内容的前面描述的加法可以在第二运行阶段BP2中通过后面的准则来说明。

[0103] 在第二运行阶段 BP2 的开始时，其相应于第一运行阶段的结束，说明输入数据寄存器的工作周期的指标具有值 n=63。同时，用于第二运行阶段 BP2 的时标 v 被初始化：v =0。在工作数据块 W_0 至 W_7 中存在数据 $W_{0,n=63}, .., W_{7,n=63}$，它们下面也用 $W_{0,v=0}, .., W_{7,v=0}$ 来表示。同样，哈希数据块 H_0 至 H_7 下面也用 $H_{0,v}, .., H_{7,v}$ 来表示。

[0104] 在第二运行阶段 BP2 期间，时标 v 递增直至其最大值 v =7，由此定义了八个加法周期。在每个加法周期中进行下面的指派：

$$H_{0,v+1} = W_{7,v} + H_{7,v}$$

$$H_{k,v+1} = H_{k-1,v} \text{ 对于 } k=1, .., 7$$

图 2c 说明了前面描述的加法过程 229。在步骤 229a 中进行指派 $H_{0,v+1} = W_{7,v} + H_{7,v}$，并且在步骤 229b 中对于 k=1,...,7 进行指派 $H_{k,v+1} = H_{k-1,v}$。

[0105] 按照一种实施方式，因此首先在第一运行阶段 BP1 中对移位寄存器 SR_M, SR_W 进行时钟控制 N=64 次，以便在工作数据块 W 中形成至少暂时的哈希值，并且接着在第二运行阶段 BP2 中对移位寄存器 SR_W, SR_H 进行时钟控制八次，以便将来自这些工作数据块的至少暂时的哈希值加到哈希数据块的值上。因此迄今总共需要装置 1100 的 64+8=72 个工作周期或时钟周期。

[0106] 在一种优选的实施方式中，在前面描述的将第二移位寄存器 SR_W 的内容加到第三移位寄存器 SR_H 上之后，有利地进行输入数据块 M_0 至 M_{15} 的重新初始化，其中包括 512 位长度的消息 MSG 的下一数据块被写到第一移位数据块 SR_M 中。接着，这两个移位寄存器 SR_M, SR_W 又针对 64 个时钟（n=64 直至 n=127）以前面描述的方式（例如参见图 2b）被运行，其中函数 T, G, F 被分析，从而在 n=127 时又在第二移位寄存器 SR_W 中得到暂时的哈希值。接着，第二移位寄存器 SR_W 的内容可以重新地被加到第三移位寄存器 SR_H 的内容上（例如借助第二和第三移位寄存器按照 v =8 至 v =15 的时钟控制），这相应于对在 v =7 时在第三移位寄存器 SR_H 中存在的哈希值补充与第二组输入数据 M（例如消息 MSG 的位 512 至 1023，只要先前、也即在 n=0 至 n=63 时该消息 MSG 的位 0 至 511 被处理了）有关的哈希组成部分。该过程每次被重复，均直至哈希形成基于的消息 MSG（图 3）的所有位被处理了。只要消息 MSG 的总长度在具有 m=1 的实施方式情况下不是 512 位的整数倍，则消息 MSG 例如可以借助填补等等被带到 e*512 位（$e=1, 2, 3..$）的相应总长度上。只要消息 MSG 的总长度在具有 m=2 的实施方式情况下不是 1024 位的整数倍，则消息 MSG 例如可以借助填补等等被带到 e*1024 位（$e=1, 2, 3..$）的相应总长度上。

[0107] 关于包括（在 m=1 时）多于 512 位的消息 MSG（以及在 m=2 时多于 1024 位）的划分以及成 512 位块（1024 位块）的其分解或者关于填补的细节例如可以从标准文档 FIPS180-2 中提取，该标准文档 FIPS180-2 在开头已经被描述。

[0108] 只要为了初始化工作数据块 W 和哈希数据块 H 而使用根据一种实施方式建议的也是标准文档 FIPS180-2 基础的值，本发明方法在使用根据图 6 的实施以及值 m=1 的情况下提供与标准化的方法相同的输出值，因此完全对 SHA 256 标准兼容。

[0109] 前面所表示的初始化值例如也可以从如下文献中提取：

“ FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS) CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8900 March 2012”,更确切地说用于按照该文献的 5.3.3 章节对工作数据块和 / 或哈希数据块初始化,以及用于按照该文献的 4.2.2 章节对来自本申请的等式 3 的常数 Kn 初始化,也即对于 $K_0, \cdots, K_{63}$ 如下值 :

0x428a2f98, 0x71374491, 0xb5c0fbcf,

0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5, 0xd807aa98,

0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7,

0xc19bf174, 0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f,

0x4a7484aa, 0x5cb0a9dc, 0x76f988da, 0x983e5152, 0xa831c66d, 0xb00327c8,

0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967, 0x27b70a85,

0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e,

0x92722c85, 0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819,

0xd6990624, 0xf40e3585, 0x106aa070, 0x19a4c116, 0x1e376c08, 0x2748774c,

0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3, 0x748f82ee,

0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7,

0xc67178f2, 也即例如 K_0=0x428a2f98, K_7=0xab1c5ed5, K_8=0xd807aa98 。针于 Kn 的这些值也适用于 SHA224 ;然而为了初始化工作数据块和 / 或哈希数据块对于 SHA224 可以选择与对于 SHA256 不同的值。

[0110] 只要为了初始化工作数据块 W 和哈希数据块 H 使用根据另一实施方式建议的也是标准文档FIPS180-2基础的值,本发明方法在使用根据图6的实施以及值m=2的情况下提供与标准化方法相同的输出值,因此完全对 SHA 512 标准兼容。前面所表示的初始化值例如也可以从如下文献中提取 :“ FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS) CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8900 March 2012”,更确切地说用于按照该文献的5.3.5章节的对工作数据块和 / 或哈希数据块初始化,以及用于按照该文献的 4.2.3 章节对来自本申请的等式 3 的常数 Kn 初始化,也即对于 $K_0, \cdots, K_{63}$ 如下值 :0x428a2f98d728ae22, 0x7137449123ef65cd,

0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc, 0x3956c25bf348b538,
0x59f111f1b605d019, 0x923f82a4af194f9b, 0xab1c5ed5da6d8118,
0xd807aa98a3030242, 0x12835b0145706fbe, 0x243185be4ee4b28c,
0x550c7dc3d5ffb4e2, 0x72be5d74f27b896f, 0x80deb1fe3b1696b1,
0x9bdc06a725c71235, 0xc19bf174cf692694, 0xe49b69c19ef14ad2,
0xefbe4786384f25e3, 0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,
0x2de92c6f592b0275, 0x4a7484aa6ea6e483, 0x5cb0a9dcbd41fbd4,
0x76f988da831153b5, 0x983e5152ee66dfab, 0xa831c66d2db43210,
0xb00327c898fb213f, 0xbf597fc7beef0ee4, 0xc6e00bf33da88fc2,
0xd5a79147930aa725, 0x06ca6351e003826f, 0x142929670a0e6e70,
0x27b70a8546d22ffc, 0x2e1b21385c26c926, 0x4d2c6dfc5ac42aed,
0x53380d139d95b3df, 0x650a73548baf63de, 0x766a0abb3c77b2a8,
0x81c2c92e47edaee6, 0x92722c851482353b, 0xa2bfe8a14cf10364,
0xa81a664bbc423001, 0xc24b8b70d0f89791, 0xc76c51a30654be30,
0xd192e819d6ef5218, 0xd69906245565a910, 0xf40e35855771202a,
0x106aa07032bbd1b8, 0x19a4c116b8d2d0c8, 0x1e376c085141ab53,
0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8, 0x391c0cb3c5c95a63,
0x4ed8aa4ae3418acb, 0x5b9cca4f7763e373, 0x682e6ff3d6b2b8a3,
0x748f82ee5defb2fc, 0x78a5636f43172f60, 0x84c87814a1f0ab72,
0x8cc702081a6439ec, 0x90befffa23631e28, 0xa4506cebde82bde9,
0xbef9a3f7b2c67915, 0xc67178f2e372532b, 0xca273eceea26619c,
0xd186b8c721c0c207, 0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178,
0x06f067aa72176fba, 0x0a637dc5a2c898a6, 0x113f9804bef90dae,
0x1b710b35131c471b, 0x28db77f523047d84, 0x32caab7b40c72493,
0x3c9ebe0a15c9bebc, 0x431d67c49c100d4c, 0x4cc5d4becb3e42b6,
0x597f299cfc657e2a, 0x5fcb6fab3ad6faec, 0x6c44198c4a475817,

也即例如

K_0=0x428a2f98d728ae22, K_7=0xab1c5ed5da6d8118, K_8=0xd807aa98a3030242

。用于 Kn 的这些值也适用于 SHA384 ;然而为了初始化工作数据块和 / 或哈希数据块对于 SHA384 可以选择与对于 SHA512 不同的值。

[0111] 然而,与已知的 SHA-2 实施不同,本发明装置 1100 要求较为不复杂的结构,这尤其是在显著减少用于实施本发明装置 1100 所需的等效门的数量方面表现出。尤其是,按照图 6 的本发明实施要求小得多数量的复用器,这可以归因于函数 T、G、F 的按照本发明的结构和其至所描述的数据块 M、W 的“数据连接”。

[0112] 在另一优选的实施方式中,对于参数 m 选择值 2。因此在该情况下,输入数据块 M_0 至 M_{15}、工作数据块 W_0 至 W_7 以及必要时哈希数据块 H_0 至 H_7 分别具有 64 位的数据宽度。同

样的优选适用于必要时存在的在所涉及的数据块或包含其的寄存器之间的数据总线。数据块的数量本身不改变。就此而言，在图 6 中描绘的结构也可用于生成 SHA 512 类型的哈希值。

[0113] 与上述在下面描述的实施方式不同(该实施方式涉及具有 m=1 的 32 位实施)，对于 64 位实施(m=2)可以选择对于函数 T、G、F 的下面的定义：

$$T = M_{0,n} + M_{9,n} + (ROTR^{19}(M_{14,n}) \text{ XOR } ROTR^{61}(M_{14,n}) \text{ XOR } SHR^{6}(M_{14,n})) + (ROTR^{1}(M_{1,n}) \text{ XOR } ROTR^{8}(M_{1,n}) \text{ XOR } SHR^{7}(M_{1,n})),$$

G=T0+T1，其中

$$T0 = M_{0,n} + W_{7,n} + (ROTR^{14}(W_{4,n}) \text{ XOR } ROTR^{18}(W_{4,n}) \text{ XOR } ROTR^{41}(W_{4,n})) + ((W_{4,n} \text{ AND } W_{5,n}) \text{ XOR } (\text{NOT}(W_{4,n}) \text{ AND } W_{6,n})) + K_n,$$ 其中

$$T1 = (ROTR^{28}(W_{0,n}) \text{ XOR } ROTR^{34}(W_{0,n}) \text{ XOR } ROTR^{39}(W_{0,n})) + ((W_{0,n} \text{ AND } W_{1,n}) \text{ XOR } (W_{0,n} \text{ AND } W_{2,n}) \text{ XOR } (W_{1,n} \text{ AND } W_{2,n})),$$ 其中

$$F = W_{3,n} + T0$$。

[0114] 在选择参数 m=2，N=80 以及按照下面的等式工作数据块 W_0 至 W_7 和哈希数据块 H_0 至 H_7 的初始化值的情况下，有利地保证：本发明方法在所获得的哈希值方面完全对 SHA512 类型的 SHA-2 标准兼容。

$$W_{0,0} = 0x6a09e667f3bcc908, W_{1,0} = 0xbb67ae8584caa73b, W_{2,0} = 0x3c6ef372fe94f82b, W_{3,0} = 0xa54ff53a5f1d36f1, W_{4,0} = 0x510e527fade682d1, W_{5,0} = 0x9b05688c2b3e6c1f, W_{6,0} = 0x1f83d9abfb41bd6b, W_{7,0} = 0x5be0cd19137e2179,$$

$$H_{0,0} = 0x6a09e667f3bcc908, H_{1,0} = 0xbb67ae8584caa73b, H_{2,0} = 0x3c6ef372fe94f82b, H_{3,0} = 0xa54ff53a5f1d36f1, H_{4,0} = 0x510e527fade682d1, H_{5,0} = 0x9b05688c2b3e6c1f, H_{6,0} = 0x1f83d9abfb41bd6b, H_{7,0} = 0x5be0cd19137e2179$$。

[0115] 在另一有利的实施方案中，也可以使用与前面建议的初始化值不同的初始化值和 / 或选择常数 K_n 的其他值，其中于是没有给出对 SHA-2 方法的完全兼容性。尽管如此，这里也得到用于以小的硬件复杂度确定哈希值的有效率的实施。

[0116] 根据申请人的研究，在使用本发明原理的情况下得到在所需要的等效门数量方面大约 40% 的减少。此外，在对 SHA256 兼容的情况下仅仅需要 72 个工作周期并且在对 SHA512 兼容的情况下需要 88 个工作周期。

[0117] 如果代替借助移位寄存器 SR_W、SR_H 和 32 或 64 位宽的加法器 1200 (图 6)的八个时钟控制引起的“串行”加法(也参见图 2c)按照另一有利的实施方式设置 256 位加法器(在 m=1 时；对于 m=2 需要 512 位加法器)，则可以在一个唯一时钟中进行工作数据块至哈希数据块的加法，使得在 512 位长度的消息情况下对于 m=1 完整哈希值形成仅仅需要 65 个工作周期，因此可以节省 8 个时钟。然而与真正的 256 位加法(或者对于 m=2 而言 512 位加

法)不同,在一种实施方式中优选对于每个 32 位块(m=1)或对于每个 64 位块(m=2)逐块地进行加法,更确切地说优选在相邻块之间没有传输的情况下进行。就此而言存在与真正的 256 位宽的数据字的加法的区别。

[0118] 在另一有利的实施方式中(其中哈希值形成基于的消息 MSG 比 512 位长),第一移位寄存器 SR_M(对于周期 n=64 至 n=127)利用消息 MSG 的紧接着的块 M 的紧接着的初始化可以已经比针对前面描述的实施方式早 16 个工作周期地进行。这是可能的,因为按照本发明建议的函数 G、F 有利地仅仅需要输入数据块 M_0 的内容,然而不需要另外的输入数据块的内容。基于根据图 6 的实施方式的拓扑结构,已经提早 16 个工作周期就存在因此为重新分析函数 F、G 所需的、输入数据块 M_0 的值,也即在图 2b 中阐明的流程的第 48 工作周期(也即在 n=47 时)之后存在。在使用通过图 6 阐明的基于移位寄存器的架构的情况下,因此可以更早地开始利用哈希值形成基于的消息 MSG 的另外的部分来初始化输入数据块 M。在此情况下,可以预期另外 20% 的效率提高。

[0119] 通过图 6 所阐明的基于移位寄存器的架构的又一优点在于,可以高效地由外部单元(未示出,例如微控制器等等)将例如用于初始化的输入数据 M (图 3)引入到输入数据块中,例如通过串行或并行地将输入数据例如输送给输入数据块 M_{15}。一旦(当 m=1 时)例如将第一个 32 位输入数据传输到输入数据块 M_{15} 中,就可以对第一移位寄存器 SR_M 进行时钟控制一次,以便将输入数据块 M_{15} 的内容移位到输入数据块 M_{14} 中,等等。因此,提供输入数据 M 的外部单元仅仅需要至输入数据块 M_{15} 的数据连接(接口)。为此特别有利地也可以例如使用已经存在的输入端 $M_{15}E$,其在紧接着的哈希值形成时从第一函数块 1110 接受函数 T 的输出值 Tn。

[0120] 在另一有利的应用情况下,其中哈希值形成基于的消息 MSG 的长度恰好为 512 位,和形成(参见例如图 2b 的步骤 229)在移位寄存器 SR_W, SR_H 方面可以被减少几个工作周期,因为工作数据块 W_5, W_6, W 的内容仅仅是工作数据块 W_4 的移位复制。在此情况下,可能可以进一步节省四个工作周期,这导致大约 5% 的效率提高。在如已经描述的该消息长度的情况下,本身根本不需要和形成,相反可以直接使用工作数据块或第二移位寄存器 SR_W 的内容作为具有 512 位的消息 MSG、M 的哈希值 HW。只有当期望对 SHA 256 的兼容性时,才需要前面所述的和形成。

[0121] 本发明的原理可以以 ASIC 和 / 或 FPGA 和 / 或微控制器和 / 或 DSP (数字信号处理器)的形式来实现或者通过直接的电路技术实施来实现,其中得到小的复杂度或高效的哈希值形成的特别优点。根据申请人的研究,可以将本发明的原理例如在标准 CMOS 工艺中借助最大大约 12000 个门或等效门来实施。例如按照图 6 的装置 1100 可以在使用常用的半导体工艺(例如 CMOS 技术)的情况下被实施为集成电路。

[0122] 在另一特别优选的实施方式中,也可以将舍入常数(英文 :round constants)Kn 存储在 SRAM (静态随机存取存储器)中,由此进一步明显减少电路技术实施的复杂度。这尤其是在将本发明至少部分地以 FPGA 实施时提供。

[0123] 本发明的原理也可以例如以 VHDL 代码的形式来实现,其中规划的电子电路可以通过对相应的 VHDL 代码补充按照本发明的功能性来扩展。

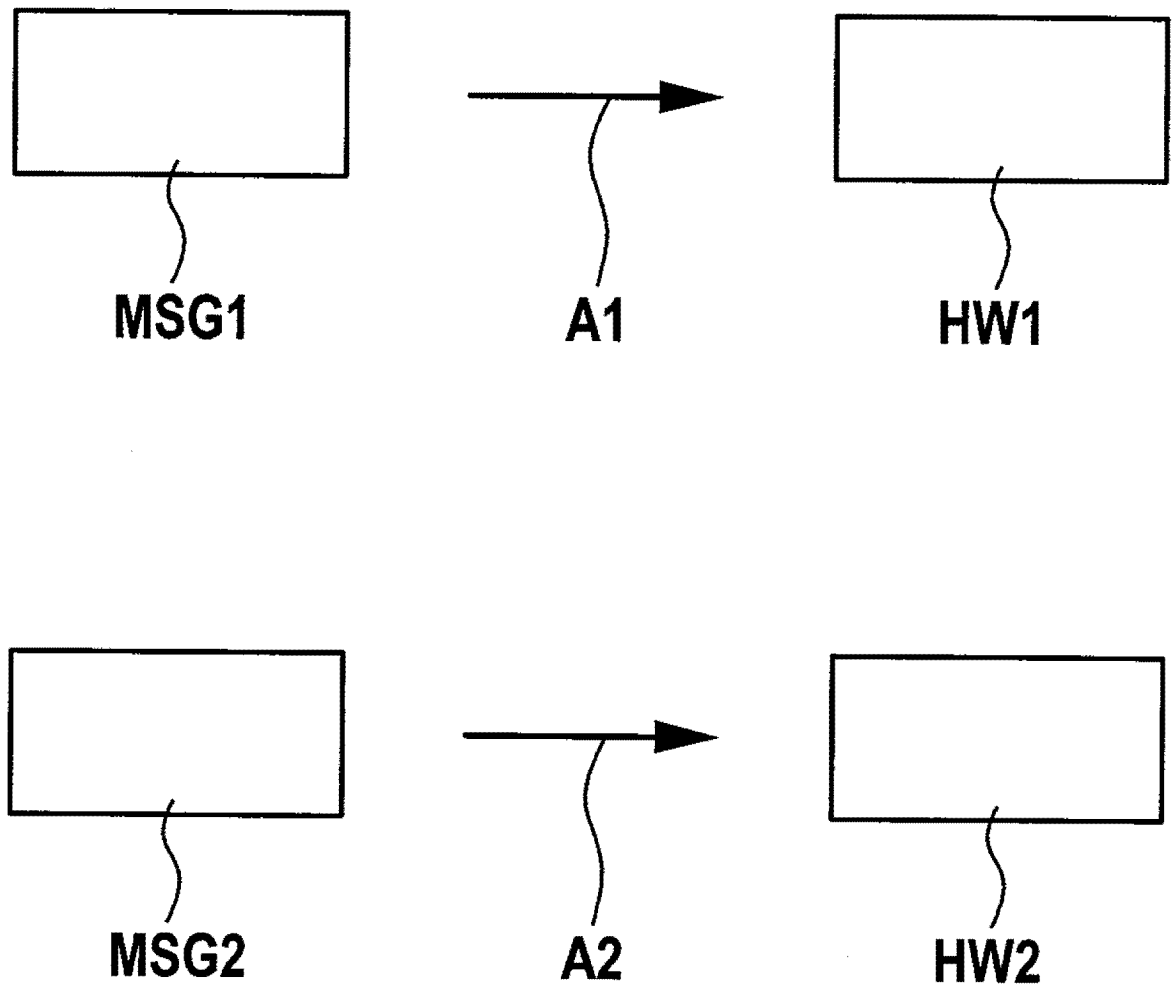


图 1

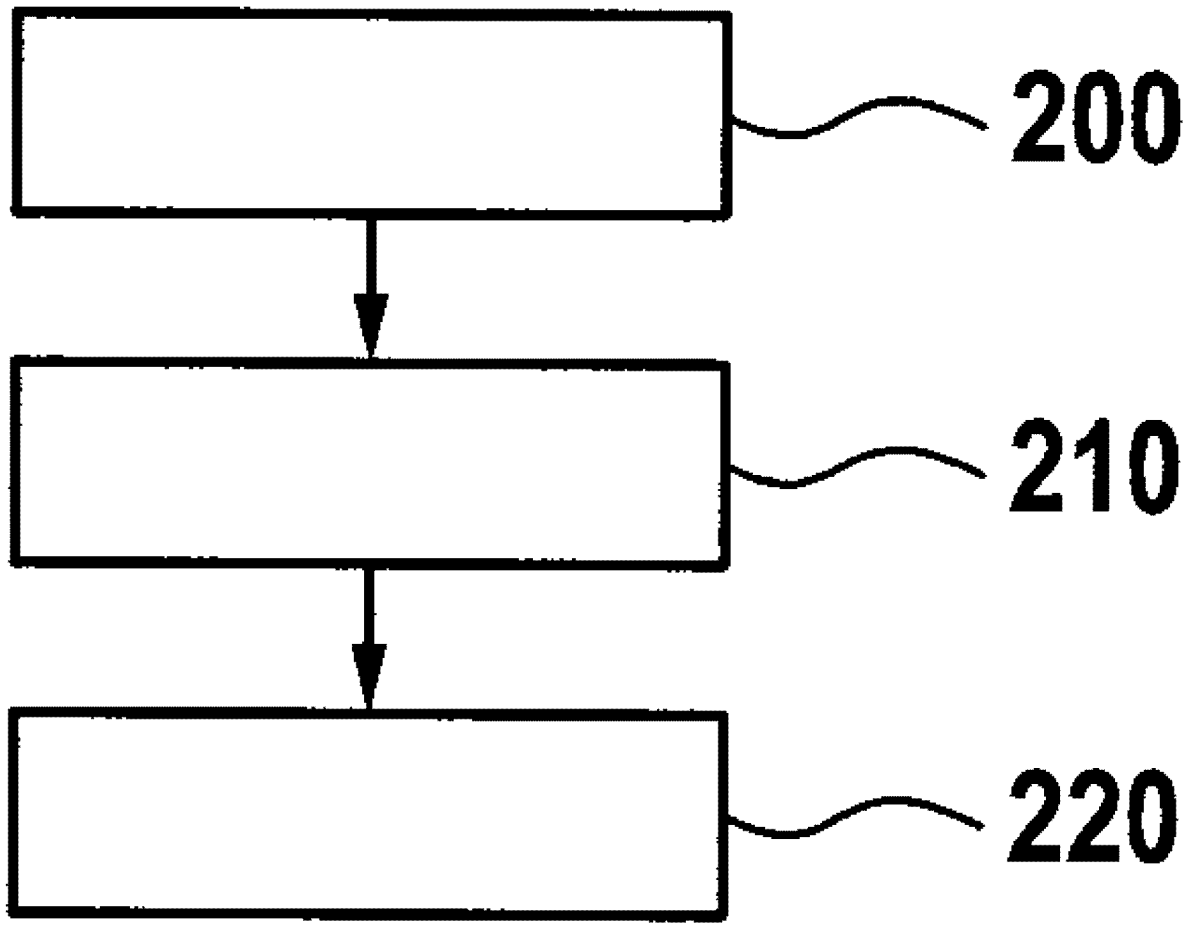


图 2a

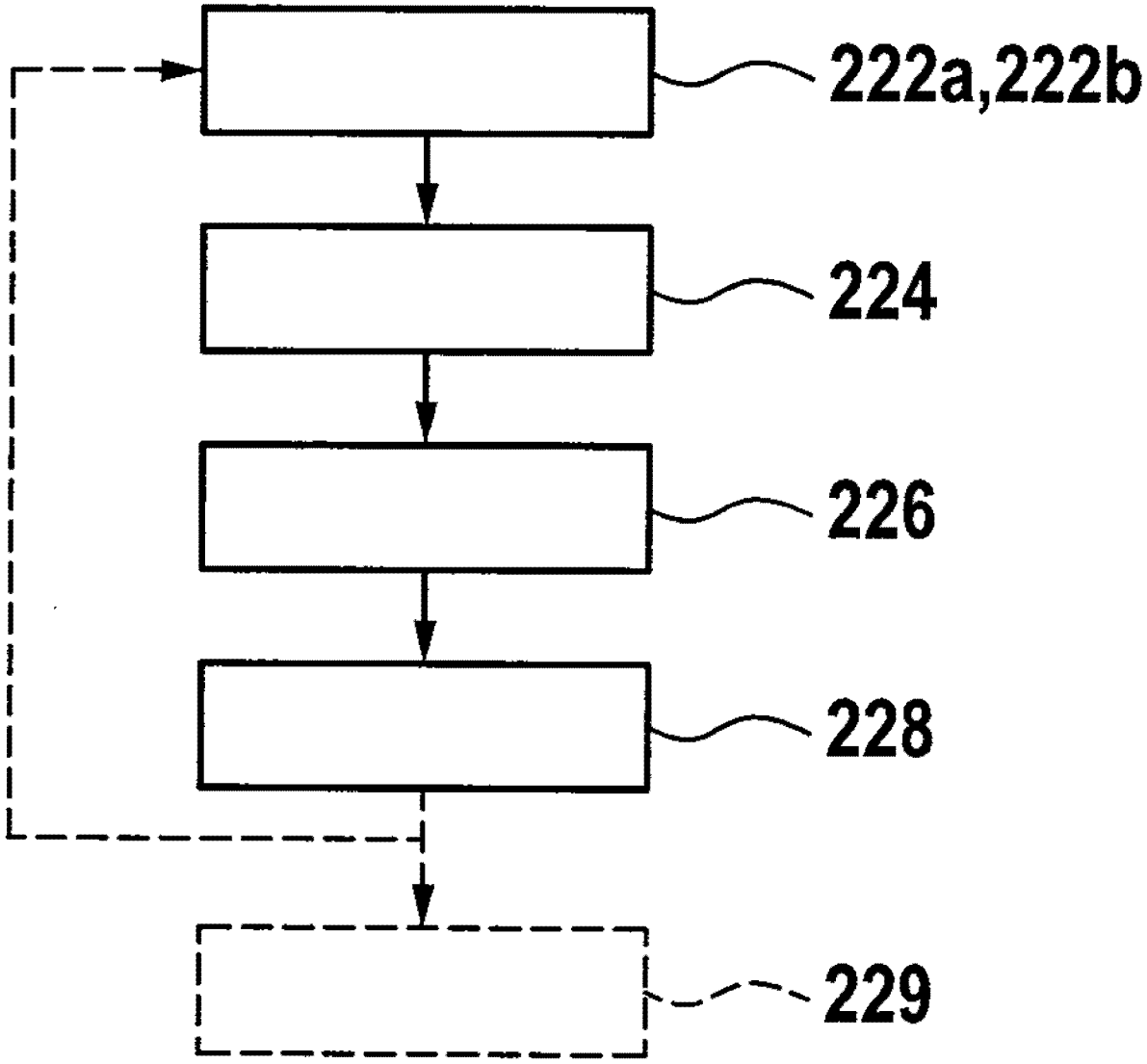


图 2b

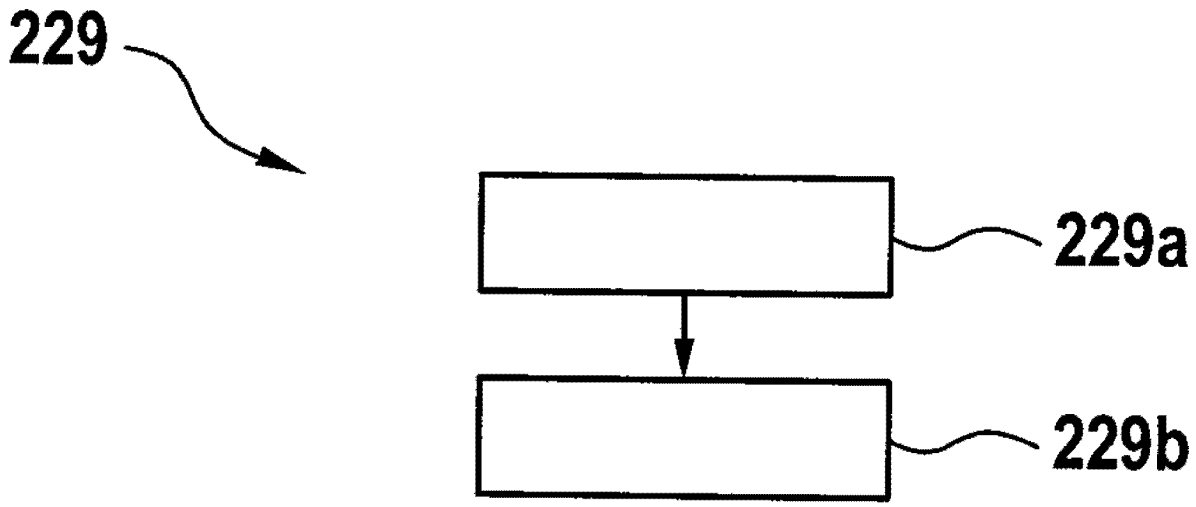


图 2c

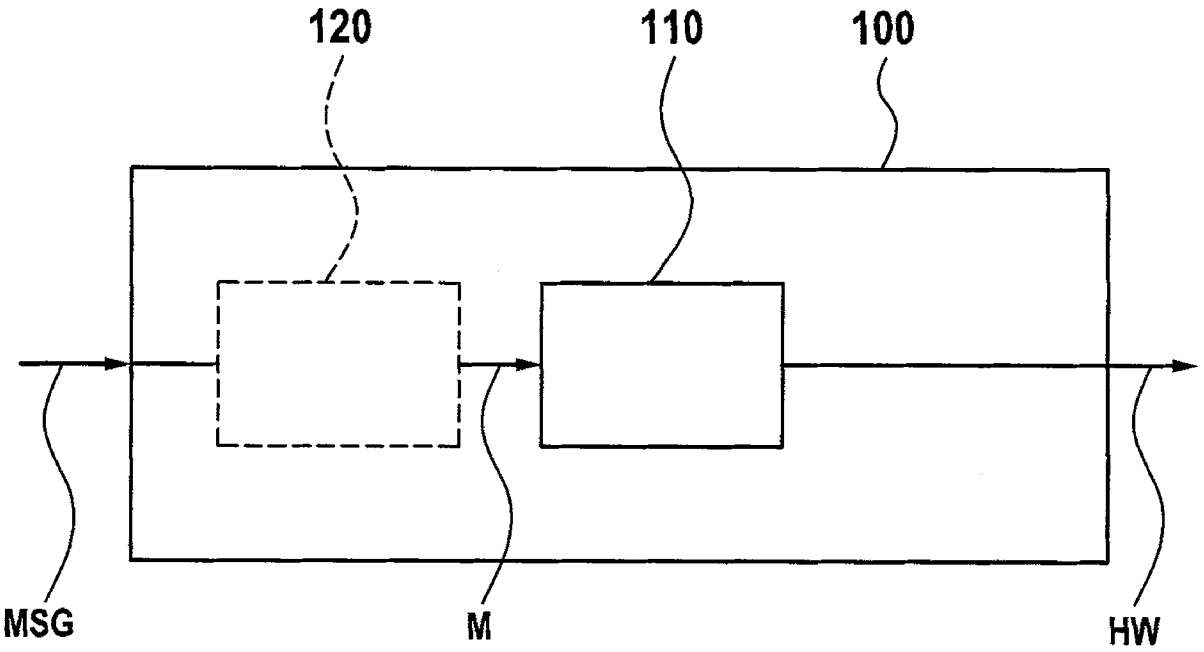


图 3

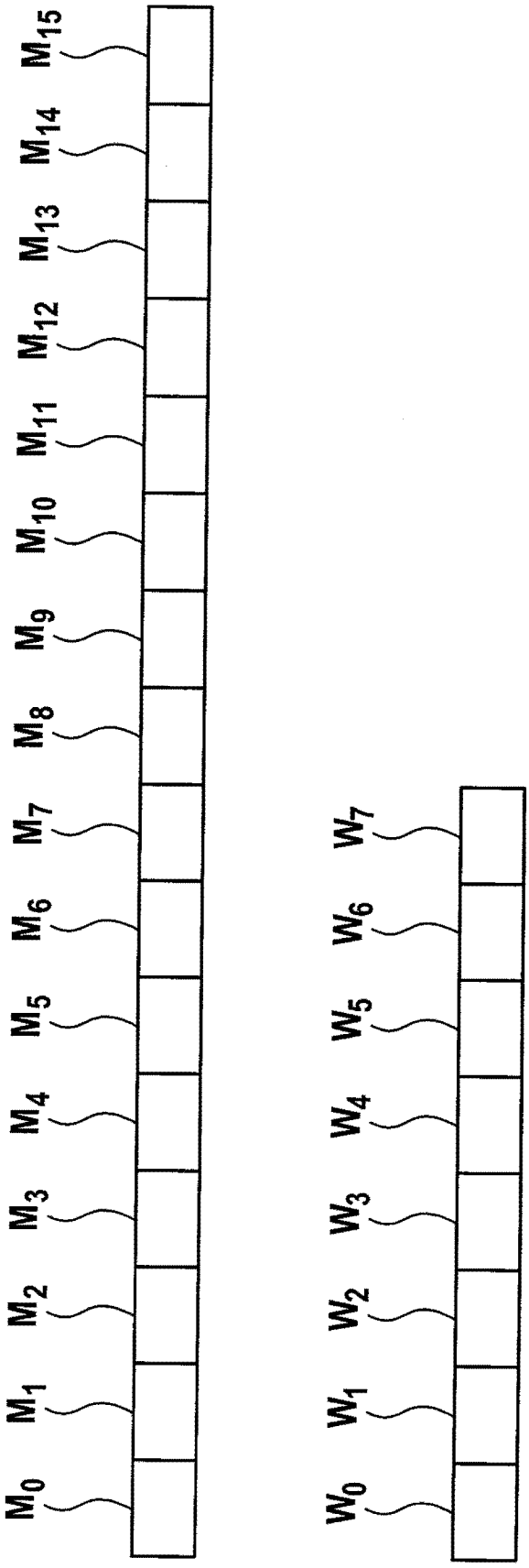

M15
M14
M13
M12
M11
M10
M9
M8
M7
M6
M5
M4
M3
M2
M1
M0
W7
W6
W5
W4
W3
W2
W1
W0

图 4

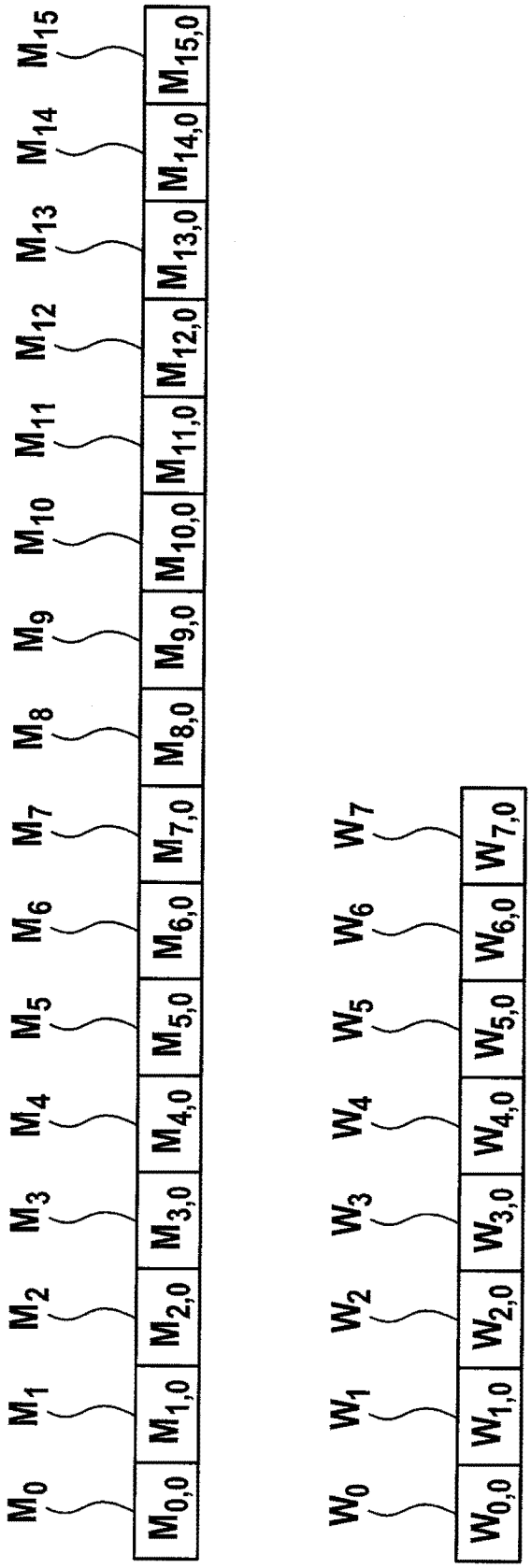

M0
M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M13
M14
M15
M0,0
M1,0
M2,0
M3,0
M4,0
M5,0
M6,0
M7,0
M8,0
M9,0
M10,0
M11,0
M12,0
M13,0
M14,0
M15,0
W0
W1
W2
W3
W4
W5
W6
W7
W0,0
W1,0
W2,0
W3,0
W4,0
W5,0
W6,0
W7,0

图 5a

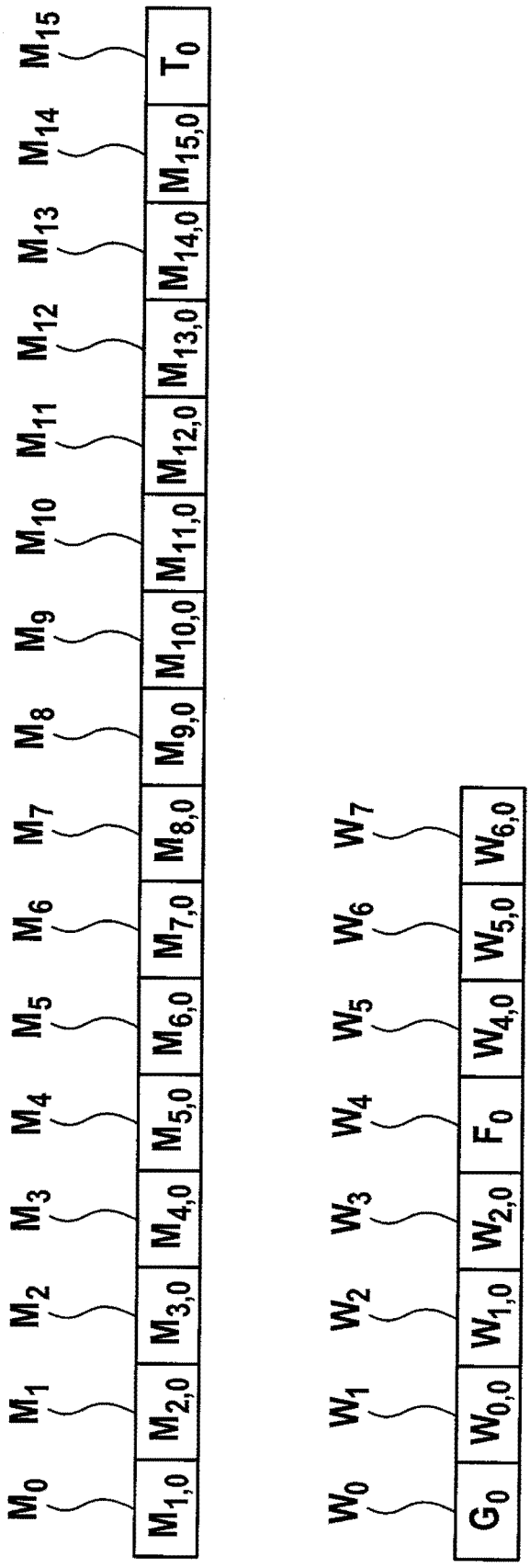

M0 M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14 M15
M1,0 M2,0 M3,0 M4,0 M5,0 M6,0 M7,0 M8,0 M9,0 M10,0 M11,0 M12,0 M13,0 M14,0 M15,0 T0
W0 W1 W2 W3 W4 W5 W6 W7
G0 W0,0 W1,0 W2,0 F0 W4,0 W5,0 W6,0

图 5b

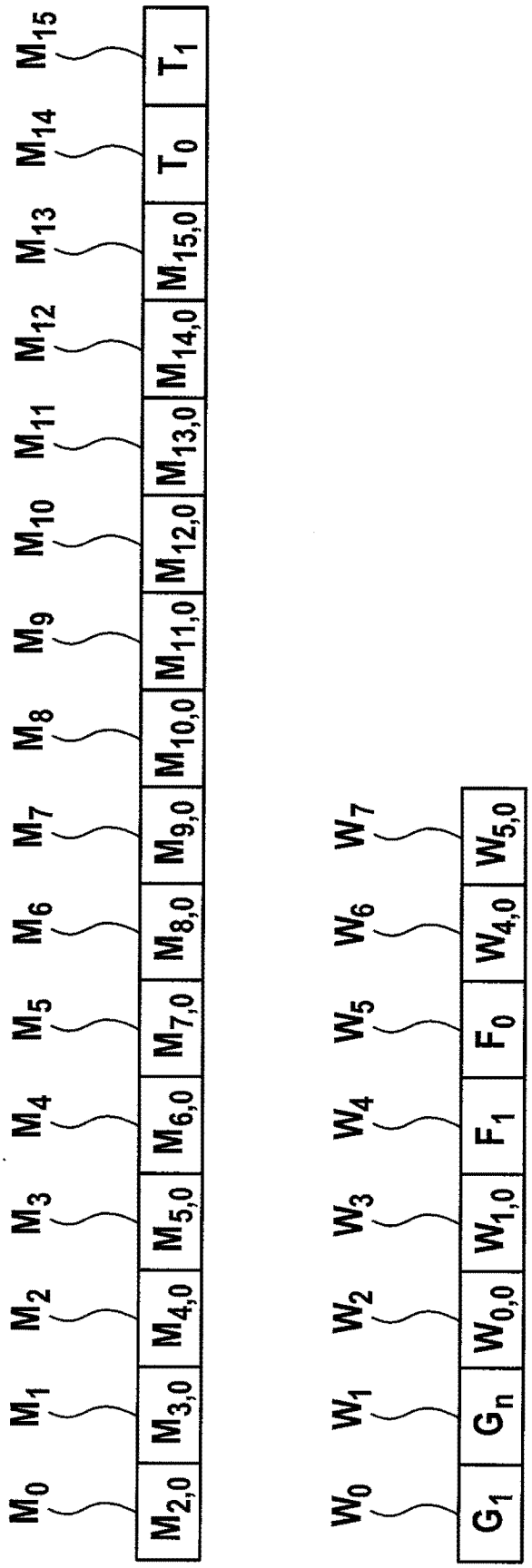

M0
M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M13
M14
M15
M2,0
M3,0
M4,0
M5,0
M6,0
M7,0
M8,0
M9,0
M10,0
M11,0
M12,0
M13,0
M14,0
M15,0
T1
T0
W0
W1
W2
W3
W4
W5
W6
W7
G1
Gn
W0,0
W1,0
F1
F0
W4,0
W5,0

图 5c

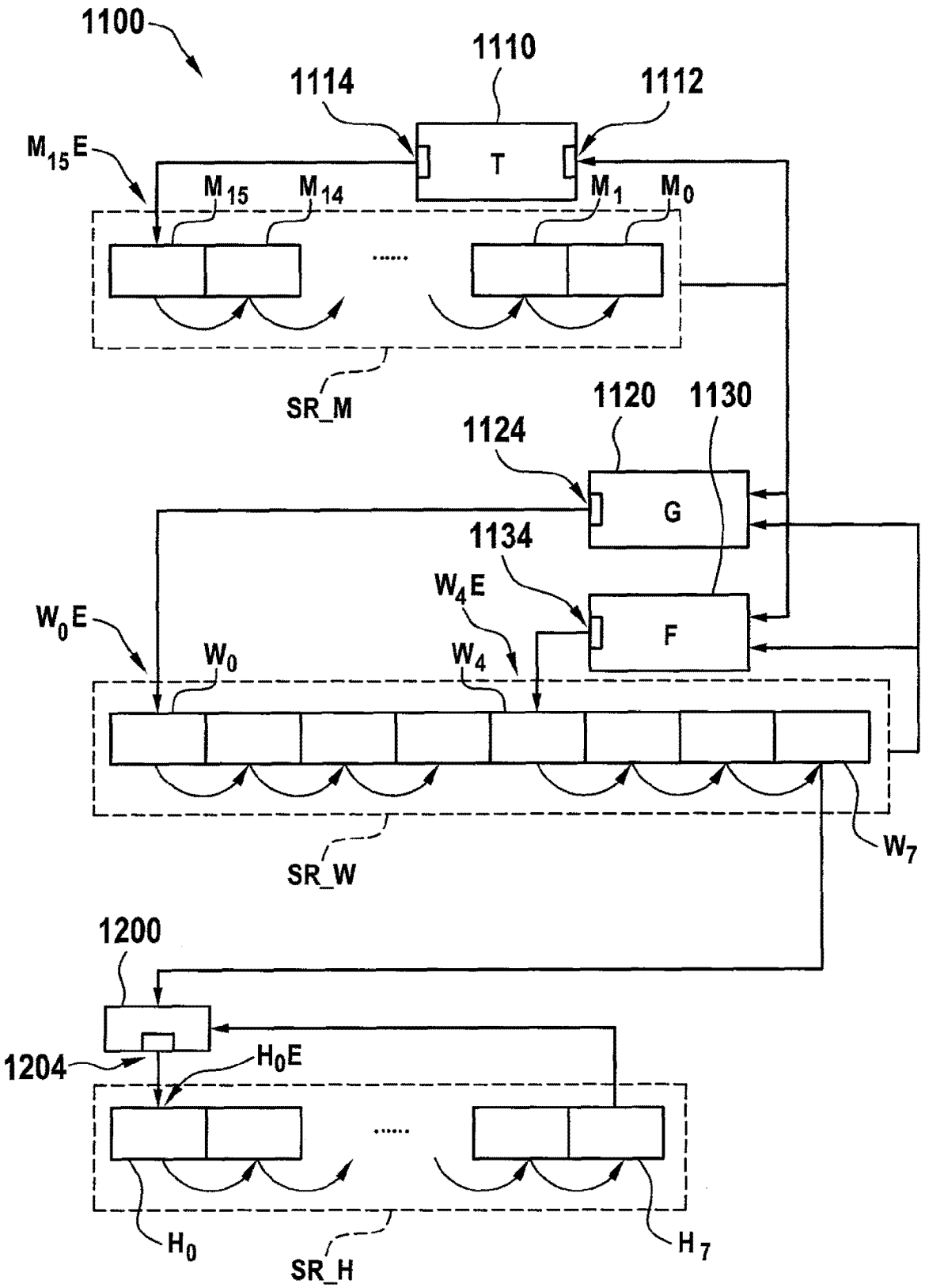

1100
1110
1114
1112
T
$M_{15}E$
M_{15}
M_{14}
M_1
M_0
SR_M
1120
1130
1124
G
1134
W_4E
F
W_0E
W_0
W_4
W_7
SR_W
1200
1204
H_0E
H_0
H_7
SR_H

图 6

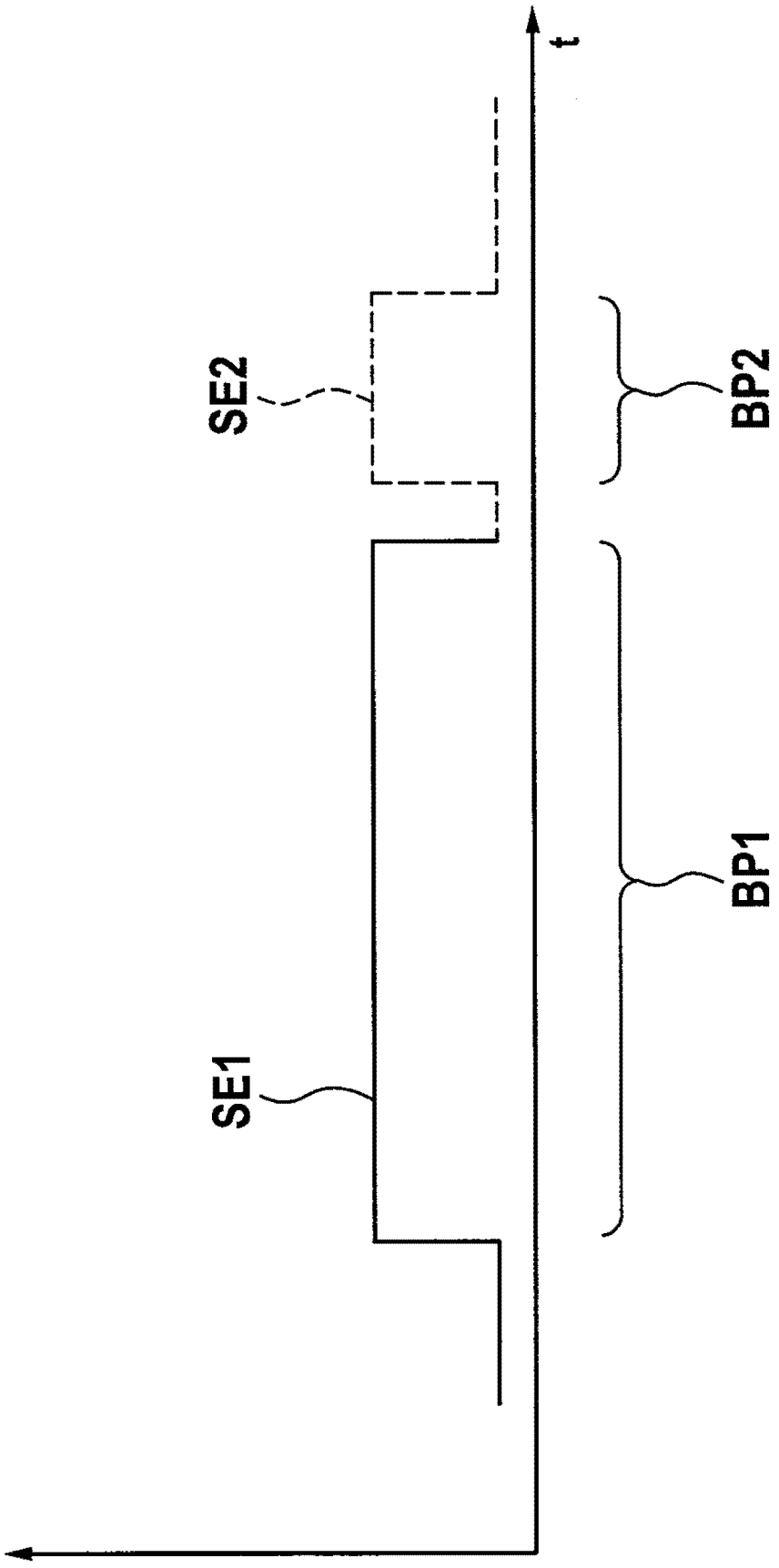


图 7