

[19] 中华人民共和国国家知识产权局



[12] 发明专利申请公开说明书

[21] 申请号 00819216.2

[51] Int. Cl⁷

G06F 12/00

G06F 12/06 G06F 12/10

G06F 12/12 G06F 9/00

G06F 9/46

[43] 公开日 2003 年 8 月 20 日

[11] 公开号 CN 1437728A

[22] 申请日 2000.12.26 [21] 申请号 00819216.2

[30] 优先权

[32] 1999.12.27 [33] US [31] 09/472,930

[86] 国际申请 PCT/US00/35209 2000.12.26

[87] 国际公布 WO01/48610 英 2001.7.5

[85] 进入国家阶段日期 2002.8.27

[71] 申请人 格里高里·V·朱德诺夫斯基

地址 美国纽约州

共同申请人 戴维 V·朱德诺夫斯基

[72] 发明人 格里高里·V·朱德诺夫斯基

戴维 V·朱德诺夫斯基

[74] 专利代理机构 中国专利代理(香港)有限公司

代理人 吴立明 王 勇

权利要求书 6 页 说明书 25 页 附图 35 页

[54] 发明名称 多组、容错、高性能存储器寻址系统及其方法

[57] 摘要

本发明提供一种多组存储器寻址系统以及通常提供间隔为 1 的访问方式下无组冲突并在其他所关心的访问方式下很少发生组冲突的方法。在一个实施例中,所提供的存储器件具有多个包括多个可寻址存储器单元的存储器组。每个存储器单元具有逻辑地址和对应的物理地址,物理地址包括组号和存储器组(2)中的局部地址。存储器件含有包括地址翻译单元(1)的地址系统,对于每个逻辑地址得到对应的物理地址。

I S S N 1 0 0 8 - 4 2 7 4

1. 一种存储器件，具有多个，N 个，存储器组，每个存储器组包括多个寻址存储器单元，每个存储器单元具有逻辑地址和对应的物理地址，物理地址包括存储器组号和局部地址；存储器件含有地址翻译单元，对于每个逻辑地址得到对应的物理地址，其中对于逻辑地址的至少一个明确的访问序列，在对应的物理地址的序列中得到的物理地址具有形成的重复方式不小于 N+1 的周期，并在对应的物理地址的序列中平均重复组号不在大约 N 个地址之内的存储器组号。

2. 根据权利要求 1 的存储器件，其中在至少一个明确的逻辑地址的访问序列中的每个逻辑地址在序列中与另一个地址间隔固定的值。

3. 根据权利要求 1 的存储器件，其中至少一个明确的逻辑地址的访问序列是在一维或多维表中的线性访问序列。

4. 根据权利要求 1 的存储器件，其中对于多个明确的逻辑地址的访问序列，所得到的对应于物理地址的序列中的物理地址具有形成的重复方式不小于 N+1 的周期并在相应的物理地址的序列中平均重复组号不在大约 N 地址之内的存储器组号。

5. 根据权利要求 4 的存储器件，其中在多个明确的逻辑地址的访问序列中的每个逻辑地址在序列中与另一个地址间隔固定的值。

6. 根据权利要求 4 的存储器件，其中多个明确的逻辑地址的访问序列是在一维或多维表中的线性访问序列。

7. 根据权利要求 1 的存储器件，其中对于每一个逻辑地址 a，相应的组号 B 和相应的局部地址 la 在模乘法结果 $\Lambda \cdot a'$ 中的位的范围得到，其中 a' 含有 a 的位的范围， Λ 代表不等于 1 的整数乘数。

8. 根据权利要求 7 的存储器件，其中 Λ 由下面确定：

(a) 将 Λ 设为值的范围；

(b) 对于每个值计算与在至少一个明确的访问方式上的组号的均匀分布的偏差；并且

(c) 从产生最小偏差的值中选择 Λ 。

9. 根据权利要求 7 的存储器件，其中每组包含子组数量 S，并且对于每个逻辑地址 a，对应的组号 B 进一步由 S 获得。

10. 根据权利要求 9 的存储器件，除了 S 个存储器子组，还包括

至少一个缺陷存储器子组。

11. 根据权利要求 9 的存储器件, 还包括:

- (a) 存储值 Λ 的 Λ 存储器;
- (b) 存储值 N 的 N 存储器;
- 5 (c) 存储值 S 的 S 存储器; 以及
- (d) 将组号映射到无缺陷组号的组表;

其中如果在一个或多个组中检测到错误, Λ 存储器、 N 存储器、 S 存储器和组表是可修改的。

12. 根据权利要求 9 的存储器件, 还包括存储 S 的值的 S 存储器。

10 13. 根据权利要求 7 的存储器件, 还包括存储 Λ 的值的 Λ 存储器。

14. 根据权利要求 7 的存储器件, 还包括存储 N 的值的 N 存储器。

15. 根据权利要求 7 的存储器件, 除了 N 个存储器组, 还包括至少一个缺陷存储器组。

15 16. 根据权利要求 7 的存储器件, 还包括将组号映射到无缺陷组号的组表。

17. 根据权利要求 1 的存储器件, 其中对于每一个逻辑地址 a , 相应的组号 B 和相应的局部地址 la 由号 a' 的查表结果中的位的范围得到, 其中 a' 含有 a 的位的范围。

20 18. 根据权利要求 1 的存储器件, 其中对于每一个逻辑地址 a , 相应的组号 B 和相应的局部地址 la 由 a' 的一个系数在有限或无限域上的多项式乘法的结果中的位的范围得到, 其中 a' 含有 a 的位的范围。

25 19. 根据权利要求 1 的存储器件, 其中对于每一个逻辑地址 a , 相应的组号 B 和相应的局部地址 la 由具有输入 a' 的组合电路的输出中的位的范围得到, 其中 a' 含有 a 的位的范围。

20. 一种具有多个, N 个, 存储器组的存储器件, 每个存储器组包括多个寻址存储器单元, 每个存储器单元具有逻辑地址 a 和对应的物理地址, 物理地址由存储器组号 B 和局部地址 la 组成, 存储器件含有地址翻译单元, 对于每个逻辑地址得到对应的物理地址, 其中

30 (a) 对于逻辑地址的至少一个明确的访问序列, 在对应的物理地址的序列中得到的物理地址具有平均重复组号不在对应的物理地址的序列中的大约 N 个地址之内的存储器组号, 并且

(b) N 的值可以重新设置为在所述存储器件上小于可用组的最大数量的任何数量的组。

21. 根据权利要求 20 的存储器件, 其中在至少一个明确的逻辑地址的访问序列中的每个逻辑地址在序列中与另一个地址间隔固定的值。

22. 根据权利要求 20 的存储器件, 其中至少一个明确的逻辑地址的访问序列是在一维或多维表中的线性访问序列。

23. 根据权利要求 20 的存储器件, 其中对于多个明确的逻辑地址的访问序列, 所得到的对应于物理地址的序列中的物理地址具有形成的重复方式不小于 $N+1$ 的周期并在对应的物理地址的序列中平均重复组号不在大约 N 地址之内的存储器组号。

24. 根据权利要求 23 的存储器件, 其中在多个明确的逻辑地址的访问序列中的每个逻辑地址在序列中与另一个地址间隔固定的值。

25. 根据权利要求 23 的存储器件, 其中多个明确的逻辑地址的访问序列是在一维或多维表中的线性访问序列。

26. 根据权利要求 20 的存储器件, 其中:

- (a) 存储器件包含一个或多个缺陷存储器组;
- (b) 对于这种有缺陷的存储器组的数量, 值 N 小于在所述存储器件上的可用存储器组的最大数量; 并且
- (c) 重新配置所述存储器件, 使 N 个存储器组的工作不受影响。

27. 根据权利要求 20 的存储器件, 其中:

- (a) 存储器件包含一个或多个保留的存储器组;
- (b) 对于这种保留的存储器组的数量, 值 N 小于在所述存储器件上的可用存储器组的最大数量; 并且
- (c) 重新配置所述存储器件, 使 N 个存储器组的工作不受影响。

28. 根据权利要求 20 的存储器件, 其中:

- (a) 存储器件包含一个或多个被忽略的存储器组;
- (b) 对于这种被忽略的存储器组的数量, 值 N 小于在所述存储器件上的可用存储器组的最大数量; 并且
- (c) 重新配置所述存储器件, 使 N 个存储器组的工作不受影响。

29. 根据权利要求 20 的存储器件, 还包括将组号映射到无缺陷组号的组表。

30. 根据权利要求 20 的存储器件，还包括存储 N 的值的 N 存储器。

31. 一种系统，包括：

多个， N 个，存储器组，每个存储器组包括多个寻址存储器单元，
5 每个存储器单元具有逻辑地址 a 和对应的物理地址，物理地址包括存储器组号 B 和局部地址 la ；

多个， M 个，处理单元，与所述存储器单元通信

地址翻译单元，对于每个逻辑地址得到对应的物理地址，

其中，对于逻辑地址的至少一个明确的访问序列，在对应的物理
10 地址的序列中得到的物理地址具有平均重复组号不在对应的物理地址的序列中的大约 N 个地址之内的存储器组号，

并且其中系统进一步的特征为下面中的一个，从下面组成的组中选取：

(i) N 的值可以重新设置为在所述系统上小于这种可用单元的最大
15 数量的存储器组的任何数量，并且

(ii) M 的值可以重新设置为在所述系统上小于这种可用单元的最大数量的处理单元的任何数量。

32. 根据权利要求 31 的系统，其中对于逻辑地址的多个明确的访问序列，在对应的物理地址的序列中得到的物理地址具有平均重复组
20 号不在对应的物理地址的任何序列中的大约 N 个地址之内的存储器组号。

33. 一种在具有多个， N 个，包括多个寻址存储器单元的存储器组的存储器件，每个存储器单元具有逻辑地址 a 和对应的物理地址，物理地址包括存储器组号 B 和局部地址 la ，一种方法，其中相应的组
25 号 B 和相应的局部地址 la 在模乘法结果 $\Lambda \cdot a'$ 中的位的范围得到，其中 a' 含有 a 的位的范围， Λ 代表整数乘数，其中 (a) 对于逻辑地址的至少一个明确的访问序列，在对应的物理地址的序列中得到的物理地址具有平均重复组号不在对应的物理地址的序列中的大约 N 个地址之内的存储器组号，以及 (b) N 的值可以重新设置为在所述存储
30 器件上小于可用组的最大数量的组的任何数量。

34. 根据权利要求 33 的方法，其中在给定的逻辑地址的访问序列中的每个逻辑地址与序列中的另一个地址间隔固定的值。

35. 根据权利要求 33 的方法, 其中每个明确的逻辑地址的访问序列是在一维或多维表中的线性访问序列。

36. 根据权利要求 33 的方法, 还包括存储将组号映射到无缺陷组号的组表的步骤。

5 37. 根据权利要求 33 的方法, 还包括在 N 存储器中储存 N 的值的步骤。

38. 根据权利要求 33 的方法, 其中每组包含子组数量 S, 并且对于每个逻辑地址 a, 对应的组号 B 进一步由 S 获得。

10 39. 根据权利要求 38 的方法, 还包括在 S 存储器中储存 S 的值的步骤。

40. 根据权利要求 38 的方法, 还包括步骤:

- (a) 在 Λ 存储器中存储 Λ 的值;
- (b) 在 N 存储器中存储 N 的值;
- (c) 在 S 存储器中存储 S 的值;
- 15 (d) 存储将组号映射到无缺陷组号的组表; 以及
- (e) 如果在一个或多个组中检测到错误, 修改 Λ 存储器、N 存储器、S 存储器和组表中的一个或多个。

41. 根据权利要求 33 的方法, 还包括在 Λ 存储器中存储 Λ 的值的步骤。

20 42. 一种存储器件, 包括:

多个, N 个, 存储器组, 每个存储器组由多个寻址存储器单元组成, 每个存储器单元具有逻辑地址和对应的物理地址, 物理地址包括存储器组号和局部地址; 以及

25 一种地址翻译单元, 对于每个逻辑地址得到对应的物理地址, 其中, 对于逻辑地址的至少一个明确的访问序列, 在对应的物理地址的序列中得到的物理地址具有形成的重复方式不小于 $N+1$ 的周期的存储器组号, 并其中在对应的物理地址的序列中组冲突的数量平均少于访问的 $1/e$ 。

43. 一种存储器件, 包括:

30 多个, N 个, 存储器组, 每个存储器组包括多个寻址存储器单元, 每个存储器单元具有逻辑地址和对应的物理地址, 物理地址包括存储器组号和局部地址;

一个地址翻译单元，对于每个逻辑地址得到对应的物理地址，其中，对于逻辑地址的至少一个明确的访问序列，在对应的物理地址的序列中得到的物理地址具有形成的重复方式不小于 $N+1$ 的周期的存储器组号，并在对应的物理地址的序列中平均重复组号不在大约 N 个地址之内的存储器组号；

其中 N 等于 2^b ，并且

逻辑地址 a 包含 2^K 地址空间，

并且其中对于每一个逻辑地址 a ，加扰地址 A 包含 $\Lambda a \bmod 2^K$ ，

相应的组号 B 由 A 的第一选择位得到，并且

相应的局部地址 la 由 A 的第二选择位得到。

44. 一种存储器件，包括：

多个， N 个，存储器组，每个存储器组由多个寻址存储器单元组成，每个存储器单元具有逻辑地址和对应的物理地址，物理地址包括存储器组号和局部地址；

一个地址翻译单元，对于每个逻辑地址得到对应的物理地址，其中，对于逻辑地址的至少一个明确的访问序列，在对应的物理地址的序列中得到的物理地址具有形成的重复方式不小于 $N+1$ 的周期的存储器组号，并在对应的物理地址的序列中平均重复组号不在大约 N 个地址之内的存储器组号；

其中 N ：

(a) 对于每一个逻辑地址 a

(i) 加扰地址 A 包含 $\Lambda a'$ ，其中 Λ 是不为 1 的奇整数乘数并且 a' 含有 a 的选定位，以及

(ii) 相应的组号 B 由 A 的第一选择位得到，并且

(iii) 相应的局部地址 la 由 A 的第二选择位得到。

45. 一种存储器件，包括：

多个， N 个，存储器组，每个存储器组由多个寻址存储器单元组成，每个存储器单元具有逻辑地址和对应的物理地址，物理地址包括存储器组号和局部地址；以及

一个地址翻译单元，对于每个逻辑地址得到对应的物理地址，其中，每个逻辑地址及其对应的物理地址通过准晶体映射相联系。

多组、容错、高性能存储器寻址系统及其方法

发明领域

5 本发明通常涉及存储器寻址系统及其方法，并且具体的涉及提供高性能访问具有任意组数的多组存储器的存储器寻址系统和方法。

发明背景

对于实现最高性能的常规存储器设计，存储器空间由独立元件组成，通常称为组，其数量通常是2的幂。存储器空间在组中交叉存取，
10 即连续的地址通常映射到不同的组。在这种情况下，组数可以通过简单地看地址的最低位A来确定：即，组=A模 2^b ，其中 2^b 为组数N。该方法已被用于采用多达512组存储器的高性能系统中。增加存储器组的数量通常增加了存储器的处理量并由此增加了存储器系统到处理单元的带宽。该处理量通常是计算机运算的最弱点。

15 该存储器的已知的问题表现在当以偶数或能被2的更高次幂整除的间隔访问数组或其它数据结构时性能的降低。例如，在16组的系统中，以16的间隔访问将导致最差的性能，因为只有16组中的一组被访问。在许多实际的应用中，阵列访问具有能被2的高次幂整除的间隔。例如，在 $2^m \times 2^m$ 尺寸的矩阵中，由于 $m \geq b$ ，列访问只能提供峰值性能的 $1/N$ ，因为整个列属于同一个存储区组。类似的
20 性能降低发生在其他类型的明确图形的访问中（即，明确限定的访问顺序，通常称为有规律的访问顺序，例如，上述矩阵中的对角线访问）。

该存储器的另一个问题表现为缺乏容错性，特别是在单个芯片中具有多个存储器组的存储器器件中。在这种单芯片器件中，单个坏组
25 （即，具有至少一个不能使用的存储器单元的组）通常导致或者整个存储器件报废或者所用的组数减少为2的更低一次幂。该问题特别明显，因为新存储器部件的产量非常低，所以，制造这种器件的设备的部分生产能力被用于制造卖不出去的产品。在具有嵌入存储器单元的
30 新一代多处理器芯片中，该问题进一步恶化。例如，这种芯片可能包含 2^b 个微处理器和 2^b 个存储单元（例如，每个单元具有1到8Mbit的DRAM），理论上，相互通信需要全 $2^b \times 2^b$ 的纵横交换网络。在这

种芯片中的存储器可以作为 $2^b \cdot 2^m$ 的存储单元的浮动地址空间的共享存储器方式来处理, 其中 2^m 为每个独立存储单元的尺寸。嵌入存储器芯片比普通存储器单元复杂得多; 因此, 这种芯片报废或降级的成本相对比普通存储器单元这样做的成本高。

5 尝试解决这些问题并未取得完全的成功。

例如, RAMBUS 和其它类似的技术尝试对不分组存储器通过更快的存储器操作或简化交叉存取的多组存储器来缓解处理器-存储器之间的瓶颈。但是, 主要在仅用于连续存储器请求上取得了进步。此外, 随着处理单元速度的显著增加, 瓶颈依然存在。

10 特别针对组冲突问题的另一个技术在 P. P. Budnick and D. J. Kuck, "The organization and use of parallel memories", IEEE Trans. Computers, 20, pp. 1566-1569 (1971)中进行了说明。Budnick 等人建议采用 p 个存储器组来实现存储器, 其中 p 为素数。在这种情况下, 对于线性阵列的组冲突仅发生在间隔能被 p 整除的情况下。可以证明,
15 在这种情况下组冲突可能比实际的少。但是, 译码逻辑显著地增加了一——特别是需要除数为 p 的全整数除法电路。对于请求的地址 A , $A \bmod p$ 的余数给定了地址的组, 同时 A/p 的商给定了组内的地址。早期的 BSP (巴勃斯科学处理机) 具有这种类型的存储器系统, 其中 $p=17$ 。除了译码逻辑的增加, 这种解决方法是不适当的, 因为组数限定为素数对于例如布局、布线和接口造成了太多的限制, 非素数组数,
20 特别是 2 的幂是最佳选择。

另一个组冲突问题的补救方法是采用伪随机数发生器产生逻辑地址 A 和对应组之间的映射。例如, 在 R. Raghavan, J. P. Hayes, "On randomly interleaved memories", Proceedings of Supercomputing, pp.
25 49-58, 1990 中说明了一个这种系统。伪随机发生器为输入值的有序序列产生输出值的随机序列, 但对于给定的输入值将一直产生相同的输出值。这种技术的一个问题是对间隔为 1 的访问将产生组冲突。间隔为 1 的访问在大多数计算机应用中是最普通的访问模式 (例如, 当读取指令流时发生), 因此, 对于这种访问在存储器性能的任何降
30 低都是不能接受的。通常的问题是伪随机或真随机映射产生的组冲突即使对于相当大的 N 平均不少于访问的 $1/e\%$ (即, $36.78\ldots\%$) (其中 e 为自然对数的底)。这导致相当大的峰值性能降低。此外, 某种已知

的伪随机数发生器可能在地址空间和所有组之间产生不一致的映射（即，某些组可能比其它组具有更多的地址映射），从而增加组冲突并降低性能。

因此，当简单地址翻译方案（标准交叉存取方案或来自 Budnick-Kuck 翻译的各种方案）产生用于固定间隔方式的序列（并由此在许多间隔遇到可重复的组冲突）的组数的简单周期序列时，一般的地址不规则映射产生用于任意具体访问序列的组数的随机序列。如上所述，属于相应的物理地址的这些组数的序列具有统计上的显著的组冲突（在 N 地址的序列内），并且，充分随机化后，没有小于加扰的地址空间的尺寸的周期。

因此，通常提供用于间隔为 1 的访问方式的无冲突访问的低复杂性的容错加扰技术、其它特别重要的明确访问方式，例如偶间隔方式、2 的幂间隔方式或对角线和其它感兴趣的访问方式是非常需要的。

发明概要

因此，本发明的目的是提供一种多组存储器寻址系统以及通常提供间隔为 1 的访问方式下无组冲突并在其他所关心的访问方式下很少发生组冲突的方法。在一个实施例中，所提供的存储器件具有 N 个包括多个可寻址存储器单元的存储器组。每个存储器单元具有逻辑地址和对应的物理地址，物理地址包括组号和存储器组中的局部地址。存储器件含有包括地址翻译单元的地址映射系统，对于每个逻辑地址得到对应的物理地址。在优选实施例中，地址翻译单元工作，从而，对于逻辑地址的至少一个明确的访问序列（例如，序列中的每一个逻辑地址与序列中的其它地址相隔间隔值），在对应的物理地址的序列中得到的物理地址具有形成的重复方式不小于 $N+1$ 的周期并在对应的物理地址的序列中平均重复组号不在大约 N 地址之内的存储器组号（或者甚至周期小于地址空间的尺寸）。

由地址翻译单元进行的映射在这里称作“有限准晶体映射”。该术语是由根据本发明的优选实施例的翻译单元对于大多数间隔产生几乎是周期性的组访问方式（即，准晶体形）得出的；例如，所选择的组通常间隔固定的值，但偶尔也会间隔不同的值。为了说明的目的，一个给定间隔的 16 组存储器系统的准晶体映射的例子为 0、2、4、6、8、10、13、15、1、3、5、7、9、12、14...，其中组编号为 0 到 15。

在该例子中，序列中的组编号通常间隔为 2，但偶尔间隔为其它值（例如 3，从 10 到 13 和从 9 到 12）。对于特别明确的访问方式的优选准晶体映射为每个存储器组的访问次数大致相等。在优选实施例中，差异（在这里该术语的意思为组访问与均匀分布的偏差）为最小。这里
5 每组的偏差仅仅为 $O(1)$ (order 1)。

在一个实施例中，准晶体映射通过采用下面形式的模变换加扰地址 a 来实现：

$$a-A=\Lambda a \bmod 2^K$$

其中 A 为对应于 a 的加扰后的地址， 2^K 为地址空间（其中 K 依赖于
10 存储器制造工艺，并且在下面的例子中对于字对准存储器通常为大约 21）， Λ 为奇数常数。在该例子中的组号由加扰地址 A 的高位得到。

为了在该方案中得到有限准晶体映射，选择 Λ 以便使在 2^K 的地址空间上的重要的明确访问方式（例如变化的固定间隔或在二维或多维
15 表中访问的线性序列，包括对角线访问方式）中发生的组号的均匀分布的偏差最小。

采用各种技术使 Λ 的适当范围变窄。例如，最小化与组号均匀分布的偏差类似于最小化与 $\{n\cdot\theta\}$ 的小数部分的一致分配偏差的问题。因此，类似于二次方程的无理解 (quadratic irrationality) 的
20 乘数 Λ 给出了更好的均匀分布特性。（例如，参考 H. Behnke, Zür Theorie der Diophantischen Approximationen, I, Abh. Math Sem. Hamburg 3 (1924), pp. 261-318)。受到黄金分割 $\tau=(\sqrt{5}-1)/2$ （大约为 0.6180）的启发，一种方法是对于 $M\leq K$ 将 Λ 设为接近于 $\tau\cdot 2^M$ 的整数。这不是优选实施例，并且遇到性能下降的问题。下面说明更好的
25 的实施例。

此外，通过用于有理数的公式 $\Lambda/2^K$ 的连续小数展开式算法的优化使 Λ 的潜在的合适范围变窄。例如，参考 Rockett 和 Szűsz 的 Continued Fractions, World Scientifica Publishing Co. Pte. Ltd. (1994)。优化算法设法找到潜在的适当的整数乘数 Λ ，从而两个条件
30 同时满足：(a) 对于 $M\leq K$ ，在 $\Lambda/2^M$ 的连续小数展开式中的初项 a_i 都很小（例如，1 或 2）；并且 (b) 在满足条件 (a) 的乘数中 Λ 的二进制（或 Booth 编码二进制）展开式中的非零位的位数最少。该

非线性优化提供用于加扰和实现加扰器的最小化电路所需的最佳乘数 Λ 。 Λ 的最终选择仅仅根据用于地址空间上的各种明确的访问方式的组访问的均匀分布偏差的最小化。通过在整个地址空间上对各种间隔或其它明确的访问方式的组访问的彻底模拟来计算偏差。通过对 Λ 的所有可能值的偏差的彻底计算来选择合适的 Λ_S （即，奇数，并在范围 $1 \leq \Lambda \leq 2^K$ 内）。

本领域的技术人员应当理解：各种组合电路、查表法或者甚至模拟方法而不是模乘法可以与实现低差异映射相同的效果构成有限准晶体映射。

10 如果所有的组是无缺陷的（ $N=2^b$ ），组和逻辑地址可由如下的加扰地址 A 得到： A 的最高 b 位为组号，而 A 的其它位为组中的局部地址。

但是，如果一个或多个组有缺陷，则地址空间收缩到 $N \cdot 2^m$ 个存储单元，其中 $N < 2^b$ 。在这种情况下，需要将有效范围为 $0 \dots N \cdot 2^m - 1$ 的逻辑地址 a 翻译为在范围 $0 \dots N - 1$ 中的唯一组号 u ，和在范围 $0 \dots 2^m - 1$ 的局部地址 la 。实现翻译的硬件逻辑的复杂性是至关重要的。

在具有在同一个芯片中多处理器与多个存储单元通信的嵌入逻辑的多组存储器部分中特别重要。在芯片上具有嵌入存储器的多处理器芯片的常规结构由 $N=2^b$ 个微处理器和每个尺寸为 2^m （例如，每个为 1 到 8Mbit 的 DRAM）的 $N=2^b$ 个存储器组（单元）组成，通过交换网络相互通信。该交换网络为全 $2^b \times 2^b$ 的纵横交换网络。如上所述，在该芯片中的全部存储器可以作为 $N \cdot 2^m$ 的存储单元的平坦地址空间的共享存储器方式来处理。因为在部件中的所有多处理器都需要这些翻译单元，地址翻译逻辑的硬件实现的简单性是至关重要的。作为实际的例子，在这里和后面，我们考虑 $b=6$ 的情况，64 个存储器组与 64 个微处理器，每个存储器组包含 2^{13} 个高速缓存线（每个高速缓存线有多达 32 个字节）。在这种情况下，地址空间在“无缺陷”的情况下的可寻址单元（所说的高速缓存线）为 2^{19} 。由于相对较大的芯片面积，缺陷是常见的，并且好的处理器的数量 N 可能低于 32 或者更低。这些部件只能由存储器翻译单元来补救。对于高带宽容错大存储器模块和特别是具有嵌入存储器作为多组模块的大的片上系统产品，具有上述额外加扰特性的动态地址重映射单

元的结构是至关重要的应用。

本发明针对存储器翻译（重映射）问题提供了几种低成本解决方法，并采用加扰技术实现更好的固定间隔访问（以及其它明确的访问方式）。这些解决方法基于普通的有限准晶体映射方法来实现高
5 性能。在优选实施例中的解决方法采用模乘法（具有额外的低偏差特性）。

可能的实现方法之一是将地址空间细分为 N 组并同时进行加扰的新颖方法。例如，对于 2^K 的地址空间（如前所述）和 N 个存储器单元（组），其中 N 为任意数，首先进行加扰映射：

$$10 \quad a - A = \Lambda a \bmod 2^K$$

然后确定单元数 $u = A \cdot N / 2^K$ ，具有局部地址 $la = A - u \cdot 2^K / N$ 的存储单元的位置。这里 N 为短常数， $2^K / N$ 为（更长的）常数。除了标准的加扰，该方法只要求 2 乘以短数（6 位）和加法/减法。可以将各种模乘法（加扰和翻译）合并到一个模块中，以加速整个过程，从而
15 优选在一个周期时间 T （ ~ 2.5 ）ns 内完成。

在上述的例子中，每单元的局部子组的数量为 1（ $S=1$ ）。这使下面说明的普通方法的最简单的情况。对于可变组数 N 和子组数 S 操作的方案，明显优于上述例子，并且由于它们具有最小的复杂性和高性能，我们推荐他们。这种由加扰地址 A 获得组号 u 和局部地
20 址 la 的低复杂性的技术在下面的本发明的实施例的详细说明中提供。

附图简要说明

图 1 是本发明的存储器寻址系统的一个实施例的框图。

图 2 是本发明的一个实施例的存储器部分的框图。

25 图 3 是本发明的一个实施例的翻译单元的框图。

图 4 示出了 S-Box 工作的流程图。

图 5 示出了 M-Box 工作的流程图。

图 6 示出了 N-Box 工作的流程图。

图 7 示出了连接到单存储器阵列的本发明的实施例。

30 图 8 示出了连接到可切换总线结构中的单存储器阵列的本发明的实施例。

图 9 示出了多处理器和存储器系统。

图 10 示出了在多处理器和存储器系统中连接到局部存储器单元的本发明的实施例。

图 11 示出了本发明的另一个实施例的流程图。

图 12-37 为本发明的实施例的硬件实现的层次设计图。

5

附图中采用的公共原理图模块

如在图 12-37 中所述, 下面列出在分层设计原理图中采用的公共功能模块。他们为本领域的技术人员所熟知的逻辑模块, 并由一个或多个数据确定其参数。对于每一个模块类型, 由图 12-37 中的全部原理图说明一个例子。

10

模块名称和功能说明

1. Bits_N_K_L 输入 N 位总线 A[N-1:0], 输出子总线 A[L:K]。
2. AddN 在 A[N-1:0] 和 B[N-1:0] 上的 N 位加法器。
3. CSA 进位存储 (全) 加法器。
4. CSA-N 具有 X[N-1:0]、Y[N-1:0]、Z[N-1:0] 输入, S[N-1:0]、C[N-1:0] 输出的 CSA 的 N 个长阵列。
5. DECNE (其中 N 为任意数)
具有使能 E 的 $\lceil \log_2 N \rceil$ 到 N 的译码器。
6. GNDxN 返回 N 个地 (0); $Q[N-1:0] = 0$ 。
7. INS_N_M_L 接收 A[N-1:0] (N 个输入), 并在其下面填充 L 个地, 在其上面插入适合 M 个输出的 A[J:0] 位, 并且如果 $N+L < M$, 在结果 Q[M-1:0] 的上面增加地 (0)。
8. N_K_M 具有 M 位输出 Q[M-1:0] 的硬接线的十进制常数 K。
9. Pad_N_M 在 A[N-1:0] 下面加入 M-N 个地 (0) 以得到 M 位输出。
10. PDMuxN_M 预译码多路器。输入 N 个独立选择信号 S[N-1:0] 和矢量总线 A[0][M-1:0], ..., A[N-1][M-1:0], 得到输出 Q[M-1:0]。
11. Up_N_M 接收 A[N-1:0] 并在其低位加入 M 个 0 (左移 M 位)。结果为 N+M 长。
12. TrAddN 带进位输入 CI 和进位输出 CO 的 N 位加法器
13. SubN N 长度的减单元
14. CPM_N (截断乘积多路器 (chopped product mux))
以译码的形式从 0、B[N-1:0]、左移仅左移 N-1 位的 B[N-

1:0]和 B[N-1:0]的反中以该顺序选择。选择线为 A[2:0]。

15. ProdMux_N B[N-1:0]和{0, 1, 2, 3 = -1}的简化有符号的乘积。在最高位(第 N 位)上加 VCC (=1)。

16. Recode 将 0、1、2、3、进位输入(CI)以基数 4 重新编码(recoding)为 0、1、2、-1、进位输出(CO)。

17. Recode_{6,8} 采用重新编码模块连续重新编码 6 或 8 位。

详细说明

I. 2 的幂组数的实施例

如果组数 N 为 2 的幂(即, $N = 2^b$), 根据本发明的实施例的地址翻译单元执行如下步骤:

(1) 设置 $A = \Lambda a \bmod 2^K$, 其中 A 为对应于逻辑地址 a 的加扰后的地址, 2^K 为地址空间(其中 K 对于字对准存储器通常为大约 21), Λ 为奇数值常数; 并且

(2) 将组号设置在 A 的高 b 位并将逻辑地址设置在 A 的剩余位。

选择 Λ 从而最小化与在 2^K 的地址空间上发生在所关心的明确的访问方式(例如, 各种固定间隔或对角线访问方式)中的组号的均匀分布的偏差。对 Λ 的选择依赖于用户的要求, 并且本发明不限于任何特定的标准。适当的 Λ 可以通过例如直接计算机模拟将 Λ 的所有可能的值(即, $1 \leq \Lambda \leq 2^K$)的偏差的彻底计算来选择, 选择 Λ 以最小化与用于所关心的明确的一维和多维存储器访问方式的特定类型的组访问的均匀分布的偏差。此外, 基于 Λ 在数论上的性质的分析技术可用于预选择满足最小化与均匀分布的偏差的所需性质的 Λ 的类型; 从而减少彻底模拟所需要的时间。

模块 2^K 的选择是最佳的, 因为 $\Lambda a \bmod 2^K$ 的乘法可以用相对简单的硬件来实现, 就像在本领域的技术人员理解的那样。该方案在其实实现中只要求几个门延时, 并由此在存储器访问上至多增加一个额外的管线级。模乘法 mod, 比方说 2^{16} , 只占 16 位定点乘法器的芯片面积的一半(即, 只需要一半的门)。它也足够快, 因为乘数 Λ 预先是已知的, 并可以是布斯译码以减少在乘法树(Wallace)中的级数。通过在硬件中固定 Λ 的值可以进一步减小电路的复杂性。

在下面介绍在相当典型的多组存储器片上子系统的情况下准晶体地址映射方案的特殊例子。在该特殊例子中，片上有 16 个存储器组，并且访问流由在每个存储器组的输入端的 FIFO 和在输出端的 FIFO 缓冲。缓冲确保了存储器访问的正确顺序。组冲突的定义只是基于每个存储器组的周期时间。在该例子中，代表当前先进技术的系统周期时间为 T_{ns} ($T \sim 2.5$)，并且每个存储器组具有 $10T_{ns}$ 的周期时间（或者在下一代技术中甚至为 $8T_{ns}$ ）。因此，访问的非冲突方式是不很严格的条件。在该例中，组冲突只发生在如果在 10 个连续的存储器请求之内访问同一个组时。相反地，如果在 10 个连续的存储器请求之内没有发生组冲突，则访问在没有额外等待时间的情况下进行并且处在系统的峰值带宽。虽然随后的讨论假设该存储器结构是恰当的，但是本发明并不限于该结构，并且本领域的技术人员应当理解具体的制造商可能具有不同的结构或不同的定时特性。

在该特殊例子中，在存储器中采用逻辑地址到物理地址的伪随机或随机映射将产生存储器冲突并将显著降低系统的峰值性能。这里所有的明确的逻辑地址访问方式，包括最重要的间隔为 1 的访问方式，将变为具有不可避免的组冲突的物理地址的随机访问。例如，在存储器组的输入和输出上具有深度为 2 的 FIFO，处理量仅为 66%。增加 FIFO 的深度到 6 使处理量增加到 96%，但要花费超过三倍的平均等待时间。

例如，在对于尺寸达到不是 13 的倍数的 $O(2^M)$ 的所有间隔的存储器访问中，可选择 Λ 以产生非冲突的组访问，并由此最小化等待时间。数字 13 是具有该特性的最大（并且因此是最佳的）数字，但是可以采用其它数字，例如 11 或其它更小的素数。例如，对于 $M=14$ ，加扰乘数 $\Lambda = 3781$ 提供了 100% 的存储器利用率，并且对于直到不能被 13 整除的 350 的所有间隔等待时间最小。

该特定例子对于大多数间隔和任意数量的存储器组保持了非冲突组访问（包括，但不限定于 2 的幂的组数），同时提供随机化的其它访问。该加扰器对于固定的乘数 Λ 具有最小的复杂度（其尺寸仅为输入/输出的线性关系），这对于该寻址方案的实际实现很重要，因为它减少了电路中门的数量。此外，在该加扰方案中对于固定间

隔阵列的组访问方式类似于有限准晶体 tiling。该存储器翻译单元的例子特征在于在这里公开的系统的优选实施例中的地址加扰方案。

5 在 Λ 的选择中可采用类似的优化方法以最小化与对于间隔大于 1 的逻辑地址的其它序列和其它明确的存储器访问的一维和多维方式的组访问的均匀分布的偏差。在其它的这种例子中，可选择 Λ 从而对于间隔为 1（连续）的阵列提供无冲突访问，对于所有更大的不能被 89 整除的间隔（直到 $O(2^M)$ ）提供 100% 的带宽，但是对于小于 89 的某些间隔组访问具有比最小等待时间大的等待时间。

10 在该存储器组结构的特殊例子中，本发明的总的性能，如在模拟中所测定的，几乎是单处理器的存储器处理量的有效带宽的两倍。

II. 任意组数的实施例

但是，如果一个或多个组有缺陷，则地址空间收缩到 $N \cdot 2^m$ 个存储单元，其中 $N < 2^b$ 。在这种情况下，必须将具有有效范围 $0 \dots N \cdot 2^{m-1}$ 的逻辑地址 a 翻译为在 $0 \dots N-1$ 范围内的唯一组号 u ，并且局部地址 la 在 $0 \dots 2^m-1$ 的范围内。进行翻译的硬件逻辑的复杂性对于应用（特别是当这种地址翻译单元与多处理器系统中的各处理单元相连时）是至关重要的。此外，存储器组的尺寸是可变的。这通常通过进一步将 N 个组的每一个细分为 S 个子组来实现（由此 N 和 S 都可变）。设计存储器地址翻译（和加扰）单元使得当存储器组数 N 和子组数 S 被设置（动态地）为小于或等于在存储器系统中的可能的最大值时，该存储器地址翻译（和加扰）单元能够工作。

25 下面提供用于在这种一般情况下由加扰的地址 A 得到的组数 u 和局部地址 la 的低复杂性技术的特殊例子。

A. 定义/基本假设

在这里所用到的术语“逻辑地址”是指器件向外输出到存储器系统的用于识别存储器中的一个对象的地址。术语“物理地址”是指物理存储器单元并包含组号和组内的局部地址。在整个地址空间上逻辑地址和物理地址的映射是一一对应的。

该实施例结合 N 组存储器进行说明，其中 $1 \leq N \leq 64$ ，在每组

中包含 $S \cdot 2^{10}$ 个字, 其中 $1 \leq S \leq 8$ 。“字”或“高速缓存线”是可寻址的存储器单元, 并且可以是任意长度的。在典型的存储器系统中, 每个可寻址的存储器单元包含预定的字节数, 例如 32 字节。在该例子中, 地址空间为 $N \cdot S \cdot 2^{10}$, 最大为 2^{19} 有效字节。因此, 所有的有效存储器单元可由 19 位逻辑地址 A 表示, 范围为 $0 \leq A \leq N \cdot S \cdot 2^{10}$ 。每个逻辑地址对应于物理存储器单元, 其中物理存储器单元由组号和组内的局部地址确定; 即 $A \rightarrow (\text{组}, \text{局部})$, 其中 $0 \leq \text{组} \leq N$, $0 \leq \text{局部} \leq S \cdot 2^{10}$ 。本发明并不局限于具有上述结构的存储器。例如, 本发明可以应用于具有多于或少于 64 组或者每组多于 $8 \cdot 2^{10}$ 字或少于 $1 \cdot 2^{10}$ 字的存储器。

B. 存储器寻址系统说明

在地址翻译和加扰单元中, 根据优选实施例, 组号 Bank ($= u$) 和局部物理地址 Local ($= la$) 都由逻辑地址 A 的位的几个模块上的模乘法确定, 然后选择相应结果的位范围。如上所述, 该原理在 $N = 2^b$ 的情况下非常容易实施, 并且实施起来对于任意的组号 N 和子组号 S 变化较小。这里对于存储器翻译单元采用模乘法作为构造有限准晶体映射的方法 (多个可能的方法之一, 但在本实施例中是最佳的)。

图 1 是根据本发明的用于任意组数的存储器寻址系统的框图。在该实施例中, 翻译单元 1 接收 19 位逻辑地址 A、组号 N、子组号 S 和乘数 Λ , 并将逻辑地址 A 翻译为 6 位组号和 13 位局部地址, 然后输出。然后组号被用于在存储器部分 2 中寻址将组号依次映射为无缺陷组的物理组号的无缺陷组表 6。物理组号和局部地址用来寻址附属的存储器件。无缺陷组表 6 优选可写存储器 (例如 RAM)。或者, 如果只设置一次, 例如在附属存储器件初始测试之后, 可以是只读存储器 (ROM)。在一个实施例中, 无缺陷组表 6 为 64×6 位的表, 其中行号对应于逻辑组号, 并且表中每一行的内容提供了相应的无缺陷组的物理组号。如果少于 64 组, 表中的行并不是都被使用。存储器表 6 显示为 64×6 位的存储器是因为在该例中的有效组最大为 64。当然, 如果存储器系统具有更多的组, 则需要更大的存储器, 而如果存储器系统具有更少的组, 则只需要较小的存储器。

N、S 和 Λ 的值分别储存在存储器部分 2 的寄存器 3、4 和 5 中。
或者，它们的值可以储存在只读存储器或硬连线中。

5 优选的，配置寄存器 3、4 和 5 以及无缺陷组表 6 从而他们能够采用例如图 2 所示的扫描路径加载来更新（注意，在不同的图中相同的编号代表相同的元件）。采用该技术，N、S 和 Λ 的值进入寄存器 3，进入寄存器 3 的每一位使得寄存器 3、4 和 5 的内容右移，同时寄存器 3 的最后位移入寄存器 4，寄存器 4 的最后位移入寄存器 5，直到所有三个寄存器填入所需要的值。在无缺陷组表 6 中的值采用同样的方法设置。

10 图 3 是翻译单元 1 的框图。S-Box 12 在其 B 输入端接收 A 的高 9 位（即， $A[18:10]$ ，在下面表示为 A_{top} ）并在其 S 输入端接收 S。它产生 A_{top}/S 的商 Q 和余数 R，在本实施例中，其中 S 的值在 1 到 8 之间；即， $A_{top} = Q \cdot S + R$ 。Q 和 R 可以采用如下方法更有效的得到： $Q = (L(S) \cdot A_{top} + C(S))[高位:低位]$ ，其中 L(S) 和 C(S) 是只依赖于 S 的常数，并且[高位:低位]定义了位的范围。例如，L(S) 和 C(S)
15 可存储在寄存器或在硬件中实现。R 为 $A_{top} - Q \cdot S$ 模 8；即， $A_{top} - Q \cdot S$ 的最低三位。对于 L(S) 和 C(S) 的值， $1 \leq S \leq 8$ ，对于位范围[15:10]如下：

20 Case S = 1 $\rightarrow$ L = 1024, C = 0
 Case S = 2 $\rightarrow$ L = 512, C = 0
 Case S = 3 $\rightarrow$ L = 342, C = 0
 Case S = 4 $\rightarrow$ L = 256, C = 0
 Case S = 5 $\rightarrow$ L = 205, C = 0
25 Case S = 6 $\rightarrow$ L = 171, C = 0
 Case S = 7 $\rightarrow$ L = 146, C = 128
 Case S = 8 $\rightarrow$ L = 128, C = 0

此外，例如 Q 可以如下确定：

30 Case S = 1 $\rightarrow$ Q = $(A_{top} \cdot 128)[12:7]$ 或 Q = $A_{top}[5:0]$
 Case S = 2 $\rightarrow$ Q = $(A_{top} \cdot 64)[12:7]$ 或 Q = $A_{top}[6:1]$

Case $S = 3 \rightarrow Q = (A_{\text{top}} \cdot 171)[14:9]$

Case $S = 4 \rightarrow Q = (A_{\text{top}} \cdot 32)[12:7]$ 或 $Q = A_{\text{top}}[7:2]$

Case $S = 7 \rightarrow Q = (A_{\text{top}} \cdot 73+64)[14:9]$

5 图 4 示出了用于确定 Q 和 R 的上述过程。在步骤 20 中, 输入 A_{top} 和 S 的值并设置 B 等于 A_{top} 。在步骤 21、22 和 23 中, $L(S)$ 、C 和范围[高位:低位]的值分别根据 S 的值确定。在步骤 24 中, 商 Q 设置为 $(B \cdot L(S) + C)[\text{高位:低位}]$ 。在步骤 25 中, R 设置为 $(B - S \cdot Q)$ 模 8。在步骤 26 中, Q 和 R 为输出。

10 再参考图 3, M-Box 14(a)在其 B 输入端接收 A 的低 10 位 (即, $A[9:0]$, 在下面表示为 A_{bot}), 并在其 A 输入端输入乘数 Λ , (b)计算 $D = (A_{\text{bot}} \cdot \Lambda)$ 模 2^{10} , 并且(c)在其 Q 输出端输出结果 D。该过程在图 5 中的步骤 31、32 和 33 中分别示出。

例如, 通过彻底计算重新确定适当的 Λ ; 即, 通过在翻译单元
15 中采用这里所述的各种值并确定对于 N 和 S 的特定值产生最佳组访问方式的 Λ 值。如上所述, 当两个逻辑地址被映射到同一个组号时发生冲突。这种冲突每 $N+1$ 次访问至少发生一次。此外, 最佳组访问方式对于所关心的明确的访问方式使组冲突最小化, 例如, 固定间隔方式和线性两维和多维访问方式 (包括在矩阵中的对角线访问方式)。在优选实施例中, 间隔 1 冲突和对于所关心的其它明确访问方式的冲突 (包括固定间隔方式) 平均发生的次数不超过每 N 次访问。
20 对于各种间隔, 产生最少组冲突的 ΛS 为最佳。根据该说明书的原理构成的地址翻译单元包括采用模乘法或下面所述的其它选择中的一个的地址翻译单元, 避免例如在“发明背景”中所述的伪随机映射的缺点。
25

对于 N 的各种值, 优选的 ΛS 对于低范围间隔并且特别是间隔 1 的优化在下面提供; 对于所有的 S 在其允许的范围 $1 \leq S \leq 8$ 内采用相同的值:

30 $N = 33, \Lambda = 33, 83, 99, 165, 231, 247, 249, 353, 415, 459;$
 $N = 34, \Lambda = 25, 33, 83, 99, 165, 231, 247, 249, 353, 415,$
459;

$N = 35, \Lambda = 25, 33, 75, 83, 99, 119, 165, 231, 247, 249, 353, 415, 459;$

$N = 36, \Lambda = 25, 33, 75, 83, 99, 119, 165, 231, 247, 249, 381, 415;$

5 $N = 37, \Lambda = 25, 33, 75, 83, 99, 119, 165, 231, 249, 381, 415;$

$N = 38, \Lambda = 25, 75, 83, 119, 165, 231, 249, 381, 415;$

$N = 39, \Lambda = 25, 75, 83, 119, 231, 249, 305, 381, 415;$

$N = 40, \Lambda = 25, 75, 83, 119, 249, 305, 381;$

10 $N = 41, \Lambda = 25, 75, 83, 119, 209, 249, 305, 381;$

$N = 42, \Lambda = 25, 75, 83, 119, 209, 249, 305, 381;$

$N = 43, \Lambda = 25, 75, 83, 87, 119, 209, 249, 305, 381, 425;$

$N = 44, \Lambda = 25, 87, 119, 209, 249, 305, 381, 425;$

$N = 45, \Lambda = 25, 87, 119, 209, 305, 381, 425, 483;$

15 $N = 46, \Lambda = 25, 87, 209, 305, 381, 425, 483;$

$N = 47, \Lambda = 25, 87, 209, 305, 381, 425, 483;$

$N = 48, \Lambda = 25, 87, 209, 305, 381, 425, 483;$

$N = 49, \Lambda = 109, 209, 305, 381, 425, 483;$

$N = 50, \Lambda = 109, 209, 305, 381, 425, 483;$

20 $N = 51, \Lambda = 109, 209, 243, 305, 425, 483;$

$N = 52, \Lambda = 109, 209, 243, 305, 425, 483;$

$N = 53, \Lambda = 109, 209, 235, 243, 305, 397, 483;$

$N = 54, \Lambda = 109, 209, 235, 243, 305, 397, 483;$

$N = 55, \Lambda = 109, 209, 235, 243, 319, 397, 483;$

25 $N = 56, \Lambda = 209, 235, 243, 319, 397, 483;$

$N = 57, \Lambda = 107, 235, 243, 319, 483;$

$N = 58, \Lambda = 107, 235, 243, 319, 483;$

$N = 59, \Lambda = 107, 235, 243, 319, 483;$

$N = 60, \Lambda = 107, 235, 243, 319, 483;$

30 $N = 61, \Lambda = 107, 235, 243, 319, 483;$

$N = 62, \Lambda = 107, 235, 243, 319;$

$N = 63, \Lambda = 107, 235, 243, 375;$

$N = 64, \Lambda = 107, 235, 243, 375;$

重新参考图 3, N-Box 16 在其 Q 输入端接收 S-Box 12 的 Q 输出;
在其 D 输入端接收 M-Box 14 的输出, 并且组号 N 在其 N 输入端。N-
5 Box 16 计算并输出组号和局部地址的低 10 位 LA[9:0], 如图 6 所述。
由 N-Box 16 输出的局部地址的低 10 位与由 S-Box 的 R 输出端输出的
高 3 位 LA[12:10]组合形成整个局部地址。

由 N-Box 16 执行的过程在图 6 中示出。在步骤 41 中, Q、D 和
N 输入。步骤 42 将 X 设置为 $D \cdot N + Q$ 。步骤 43 将逻辑组号设置为
10 X 的位[15:10]; 即, Logical Bank = X[15:10]。步骤 44 将局部地址的
位[9:0]设置为 X[9:0]; 即, Local Address[9:0] = X[9:0]。步骤 45 输
出逻辑组号和局部地址的低位。

然后逻辑组号发送并作为无缺陷组表 6 的地址, 如上面结合图 1
的说明。

15 下面提供的是 S-Box 12、M-Box 14 和 N-Box 16 对于 $N = 64, S =$
 $8, \Lambda = 235 (= 2^8 - 2^4 - 2^2 - 2^0)$ 和逻辑地址 $A = 0x3DF29$ (或二进制
 0111101111100101001) 的工作的例子:

	十进制	十六进制	二进制
A	253737	3DF29	0111101111100101001
S-Box			
A_{top}	247	0F7	011110111
S	8	8	1000
L(S)	128	80	10000000
$(A_{top} \cdot L(S))$	31616	7B80	0111101110000000
$Q = (A_{top} \cdot L(S))[15:10]$	30	1E	011110
$R = (A_{top} - S \cdot Q) \bmod 2^3$	7	7	111
M-Box			
A_{bot}	809	329	1100101001
Λ	235	EB	11101011
$(A_{bot} \cdot \Lambda)$	190115	2E6A3	00101110011010100011
$D = (A_{bot} \cdot \Lambda) \bmod 2^{10}$	675	2A3	1010100011
N-Box			
N	64	40	01000000
$X = D \cdot N + Q$	43230	A8DE	1010100011011110
$X[15:10]$	42	2A	101010
$X[9:0]$	222	DE	0011011110
Output			
Local[12:10] = R	7	7	111
Local[9:0] = X[9:0]	222	DE	0011011110
Local	7390	1CDE	1110011011110
Bank	42	2A	101010

- 所说明的该实施例具有 2^K 地址空间，其中 $K = 19$ 。不需要任何进一步的改进，通过例如采用地址的 19 位子集（例如低 19 位）所得到的组号即可在任意 $K > 19$ 的地址空间上实施。在这种情况下，局部地址可通过例如将由 19 位子集所获得的局部地址添加到 K 位地址的未使用位来获得。

此外,在该实施例中所说明的技术可以容易地适应下面值的范围的任何一个: K 大于或小于 19; N 大于或小于 64; S 大于或小于 8。在这种变型中,参数 Λ 的选择根据上述的准晶体映射的原理作出。建立在适当改进的 S-、M-和 N-Box 上的正确选择 Λ 的地址翻译单元的性能随着 K 的增加 (对于 $K > 19$) 而提高。

C. 硬件实现

对于图 1-6 中示出的实施例的硬件实现,在图 12-37 中示出的层次设计图中进行了说明。在这些原理图中的低级模块的实现代表说明性的目的,并且在产品中以库和技术特定方式来实现。本领域的技术人员应理解特定实现的范围并能选择适当的库和特定工艺。例如,在较新的技术中采用缓冲器来减小写长度,而在较早的技术中优选更小门延时的更长的线。

图 12 示出了 S-Box 12 的硬件实现。Sub3 101 接收 4 四位常数 $S[3:0]$ 和来自硬连线的常数 114 (或者,从寄存器) 的值 1,并由 S 中减去 1。这将包含在范围 1 到 8 中的 S 转换为包含在范围 0 到 7 中,作为索引。结果发送到选择的相应的 $L(S)$ 的多路器 $MUX8 \times 8$ 104 的 S 输入端和确定 C 的相应值和范围[高位:低位]的译码器 DEC8 110。

多路器 $MUX8 \times 8$ 104 根据输入 S 的值选择并输出八个输入值 $A-H$ 中的一个。这些输入值 (对应于图 4 中的 $L(S)$) 来自常数排 102。如图所示,常数排 102 在位置 0 到 7 中分别包含如下的硬编码常数: 128、64、171、32、205、171、146 和 128。此外,在排 102 中的值可存储在寄存器中。

译码器 DEC8 110 也接收 Box Sub3 101 的输出,并将其输出 $Q0-Q7$ 中的一个根据接收的值置为高 (例如,如果接收的值为 0,则 $Q0$ 置为高;如果接收的值为 1,则 $Q1$ 置为高;等)。

SBox_Mult 105 计算 $B \cdot L(S) + C$, 其中 C 对于 $S = 7$ 等于 128, 否则为 0, 并在其 Q 输出端输出结果。它在其 A 输入端接收来自多路器 104 的 $L(S)$, 并在其 B 输入端接收记作 $B[8:0]$ 的逻辑地址 A 的高 9 位。在该实施例中, C 只在 $S = 7$ 时非零,在此情况下译码器 DEC8 110 的 $Q6$ 输出端输出 1 并送到 SBox_Mult 105 的 ADD128 输入端。

元件 106、107 和 108, PDMux3_6 109 和 OR 门 111 和 112 选择

SBox_Mult 105 的输出的位范围 (即, [高位:低位])。范围取决于译码器 DEC8 110 的输出。元件 106 引导这里为 SBox_Mult 105 的输出 Q 的位 Q[12:7]到 PDMux3_6 109 的输入端 A; 元件 107 引导位 Q[14:9]到 PDMux3_6 109 的输入端 B; 元件 108 引导位 Q[15:10]到 PDMux3_6 109 的输入端 C。PDMux3_6 109 为具有只有一个为逻辑 1 的三个独立选择线 S[2:0]的预译码多路器, 选择并输出三个对应的输入 A、B 和 C 中的一个。在该实施例中, 如果 S 等于 1、2 或 4 (即, 由译码器 DEC8 110 到 OR 门 111 的输出 Q0、Q1 或 Q3 为逻辑 1) 则选择输入端 A (范围[12:7]); 如果 S 等于 3 (即, 译码器 DEC8 110 的输出 Q2 为逻辑 1) 则选择输入端 B (范围[14:9]); 如果 S 等于 5、6、7 或 8 (即, 由译码器 DEC8 110 到 OR 门 112 的输出 Q4、Q5、Q6 或 Q7 为逻辑 1) 则选择输入端 C (范围[15:10])。多路器 PDMux3_6 109 的输出对应于图 1 所示的 S-Box 2 的 Q 输出。

SBox_BMSQ 113 计算 $(B - S \cdot Q) \bmod 2^3$; 具体地, 它在其 B[2:0]、S[2:0]和 Q[2:0]输入端上分别接收 B[2:0]、S[2:0]和多路器 109 的输出的低三位, 并在其 R 输出端上输出结果。每个输入仅为 3 位, 因为计算只决定结果的低三位 (即, 模 2^3)。

在图 14 中示出了 SBox_Mult 105 的实现。Recode8 250 重新编码在 8 位输入端 A 的位, 以利于高效率的乘法。在图 15 中示出了具有四个 Recode 300 模块的 Recode 250 的实现。在图 16 中示出了 Recode 300 模块的实现。ProdMux_9 251、252、253 和 254 根据其 A 输入端是否为 0、1、2 或 3 分别计算其 9 位 B 输入端与 0、1、2 或 -1 的简单有符号乘积, 并产生 11 位输出 Q, 其中 Q[10]只在输入端 A 等于 3 时置 1, 而 Q[11]一直置为 1。ProdMux_9 可以以与图 17 中所示的 ProdMux_10 相同的方式实现。元件 256、257、258、259 和 260 进行总线交换。元件 261 和 262 为通用形式 Pad_N_M 的电路; 每个在其长度为 N 的输入端加入 (Pad) M-N 个地 (0), 以产生长度为 M 的输出。在图 18 中示出了 Pad_N_M 的例子 Pad_6_10; 它在其 6 位输入中加入 4 个 0 以产生 10 位输出。在图 18 中的元件 Gndx4 340 为返回 N 个地 (0) 的通用形式 GndxN 的元件; 在 Gndx4 340 的情况下返回 4 个地。Up_14_2 263 为通用形式 Up_N_M 的电路; 它接收 N 位输入并在其低位加入 M 个 0, 产生长度为 N+M 的结果。在图 19 中示

出了 Up_N_M 的例子 Up_8_2。元件 264、265、269、271 和 272 为通用形式 Ins_N_M_L 的电路；每个接收长度为 N 的输入，在其低位增加 L 个地，在其高位插入适合 M 个输出的位数，并且，如果 $N + L < M$ ，在结果的高位增加地（0）。在图 20 中示出了 Ins_N_M_L 的例子
 5 Ins_14_20_5。CSA_16 266、270 和 273 以及 CSA_14 268 为通用形式 CSA_N 的电路；每个为 N 位长的进位存储（全）加法器 CSA 的阵列。在图 21 中示出了 CSA_N 的例子 CSA_10，并且在图 22 中示出了构成 CSA 电路的实现。如图 22 所示，CSA 分别计算其三个输入 X、Y 和 Z 的和以及进位 S 和 C。如果 $(X \oplus Y) \oplus Z$ 为 1，XOR 135 设置和
 10 S 为 1，其它情况为 0（即，如果 X、Y 和 Z 中的一个为 1 或它们都为 1，则 S 置 1）。如果 X、Y 和 Z 中的至少两个为 1，则 AND 门 136 和 OR 门 139 设置进位 C 为 1，否则设为 0。最后，在图 14 中的 Add16 274 为 N 位加法器的通用形式 AddN 的电路。在图 23 中示出了 AddN 电路的一个例子 Add16；它由四个作为 TrAddN 电路的例子的 TrAdd4
 15 290 电路组成。TrAddN 电路为具有进位输入（CI）和进位输出（CO）的 N 位加法器。在图 24 中示出了 TrAddN 电路的一个例子 TrAdd4 290。

在图 13 中示出了 Sbox_BMSQ 113 的实现。AND 门 200、201 和 202、Up_2_1 203 和 Up_1_2 204、CSA_3 205、Ins_3_3_1 206 和 Add3
 20 207 计算由 Add3 207 输出的 $S \cdot Q$ 。Sub3 208 接收 Add3 207 的输出并将其从 B 中减去（其中 B 是 A_{top} 的低三位）。在上面说明了在 Up_2_1 203 和 Up_1_2 204、CSA_3 205、Ins_3_3_1 206 和 Add3 207 中的电路。Sub3 208 是减去 N 位输入产生 N 位输出的通用形式 SubN 的电路。在图 25 中示出了 SubN 电路的一个例子 Sub3 208。

25 图 25 示出了 3 位输入 A 减去 3 位输入 B 并输出 3 位结果的电路。NOT 门 130 输出 B 的反码，记为 Y。输入 A 记为 X。CSA 131、132 和 133 为进位储存（全）加法器，每个在其 S 输出端输出结果，并在其 C 输出端输出进位。CSA 131 计算结果的低位 Q[0]，CSA 132 计算结果的中间位 Q[1]，而 CSA 133 计算结果的高位 Q[2]。具体地，CSA
 30 131 将 X[0]、Y[0] 和硬连线 1（即，VCC）相加并在其 S 输出端输出结果的第一位 Q[0]，在其 C 输出端输出进位值。CSA 132 接收并将 X[1]、Y[1] 和 CSA 131 的 C 输出相加并在其 S 输出端输出结果的第

二位 $Q[1]$ ，在其 C 输出端输出进位值。CSA 133 接收并将 $X[2]$ 、 $Y[2]$ 和 CSA 133 的 C 输出相加并在其 S 输出端输出结果的第三位 $Q[2]$ 。

在图 26-28 中示出了图 12 中的 MUX8x8 104 的实现。如图 26 所示，多路器 104 为 8 个 MUX8 150 单元的硬件阵列，一个单元对应于 8 位输入值 A-H 的每一位。在图 27 中示出了 MUX8 150 单元并包含 7 个 MUX 单元 170-176。每个 MUX 单元根据其 S 输入选择并输出其输入端 A 或 B 中的一个。MUX 单元 170-173 根据 S 的低位（即， $S[0]$ ）选择输出；MUX 单元 174-175 根据 $S[1]$ 选择输出；MUX 单元 176 根据 $S[2]$ 选择输出。在图 28 中示出了 MUX 单元的实现，并且其工作原理对于本领域的技术人员是容易理解的。

在图 12 中的常数排 102 和 N_1_3 114 为通用形式 N_K_M 的硬连线常数，其中 K 为常数的值，M 为输出的位数。在图 29 中示出了 N_K_M 的例子 N_171_8 。它在 8 位总线上输出值 171（二进制 10101011）。

在图 12 中的元件 106、107 和 108 为通用形式 $Bit_N_K_L$ 的子总线结点；结点接收 N 位总线 $A[N-1:0]$ ，并输出子总线 $A[L:K]$ 。在图 30 中示出了 $Bit_N_K_L$ 的例子 $Bit_16_9_14$ 。它输出 16 位输入总线的 9-14 位。

图 12 中的 PDMux3_6 109 为通用形式 $PDMuxN_M$ 的预编码多路器；这种多路器接收 N 个独立的选择线并输出宽度为 M 的 N 个矢量总线中的一个。在图 31 中示出了 $PDMuxN_M$ 的例子 $PDMux3_10$ 。它具有三个选择线 S 选择三个 10 位宽的输入 A、B、C 中的一个的选择线 S 并输出选中的输入。

在图 36 中示出了译码器 DEC8 110 的实现。DEC8 110 根据其 3 位输入端 A 接收的值将其输出 $Q0-Q7$ 中的一个置为高。它包括通用形式的电路的例子的 DEC2E 281 和 DEC4E 283 和 285；每一个接收 $\log_2 N$ 宽的输入，并且如果其使能输入 E 也高，则设置其 N 个输出中的一个为高。在图 37 中示出了 DECNE 的例子 DEC2E。

图 32 示出了 M-Box 14 的硬件实现。它在其 A 输入端接收 8 位值 A，并在其 B 输入端上接收地址 A 的低 10 位（即， $A[9:0]$ 或 A_{bot} ），计算 $A_{bot} \cdot A \bmod 2^{10}$ ，并输出 10 位结果。上面已讨论了图 32 的所有元件，除了 CPM_10 301、CPM_8 302、CPM_6 303 和 CPM_4 304，他们中的每一个都是 CPM_N 形式的 chopped product 多路器。CPM_N

多路器接收 N 位输入 $B[N-1:0]$ ，并根据输入 $A[2:0]$ 的选择线哪一个为逻辑 1 输出 $B[N-1:0]$ （即， $1 \cdot B$ ）、 $B[N-1:0]$ 左移一位（即， $2 \cdot B$ ）或 $B[N-1:0]$ 的反码（即， $-1 \cdot B$ ）。在图 33 中示出了 CPM_N 的例子 CPM_10301。在该例子中，输入为 10 位宽。

5 图 34 示出了 N-Box 16 的硬件实现。它在其 D 输入端上接收 M-Box 14 的输出、在其 Q 输入端上接收 S-Box 12 的 Q 输出并在其 N 输入端上接收组号 N 。D、Q 和 N 输入分别送到 Nbox_Mult 350 的 B、C 和 A 输入端，计算 $D \cdot N + Q$ ，并输出 16 位的结果。当没有缺陷组时，对于 $N = 64$ 的情况也有特殊的规定。

10 在图 35 中示出了 Nbox_Mult 350 的硬件实现。图 35 的所有元件已在上面说明过了。

优选的硬件实现具有单周期操作，通常用于常规系统。在系统中，存储器子系统通过例如当服务上一个请求时，准备新的未完成的请求可容许等待状态，“S-，M-，N-Box”的流水线操作是有利的——他能够
15 能够在只增加几个用于保存中间结果的短寄存器的成本的条件下显著缩短循环时间。另一个可能性是在上述 S-，M-，N-Box 中的所有短乘法采用异步实现。对于异步访问存储器阵列，这种方法去掉了建立/保持常数，并为大多数数据方式提供了最快的时间。根据异步乘法器阵列的实现，这种方法导致数据相关定时。

20 D. 备选实施例

根据模乘法作为产生准晶体映射的方法，对于本系统的地址翻译的各种实现有广泛的选择。它们在实现的复杂性以及对于不同的组数和不同的存储器访问的明确方式上变化。在这些优选的实施例中，对于任意的组数存在最小复杂度的硬件实现。

25 对于由加扰地址 A （其中 $A = \Lambda a \bmod 2^K$ ）得到的组号 u 和局部地址 la ，一个低复杂度技术是当每组的子组数 S 为 1（ $S=1$ ）时，设 $u = (A \cdot N) / 2^K$ 和 $la = A - u \cdot 2^K / N$ ，其中 N 为组数。这里 N 为短常数，对于固定数 N ， $2^K / N$ 为（更长的）常数。因此，除了标准加扰以外，该方法只要求两个短数（例如，6 位）乘法和加法/减法。此外，各种
30 模乘法（加扰和翻译）可以合并到一个模块中，以加速整个过程，结果在过程中能在时间 T_{ns} 内完成。

图 11 示出了根据本发明的翻译单元的备选实施例。这里，如上

所述, 组数 N 不超过 64。如在上一个实施例中, 翻译单元接收地址 A 并产生相应的组号和局部地址。在步骤 51 中, A_Top 设为 $A[18:13]$, A_Mid 设为 $A[12:7]$, A_Bot 设为 $A[6:0]$ 。在步骤 52 中, A_S 设为 $A_Top \cdot 2^7 + A_Bot$ 。步骤 51 和 52 从 A 中选 12 位用于确定组号。也可以用在
5 在这些步骤中指定的位以外的其它位。在步骤 53 中, $Middle_bits$ 设为 $(A_S \cdot \Lambda)[18:13]$ 。在步骤 54 中, $Middle_bits$ 通过无缺陷组表或类似的翻译机制映射到组号。最后, 在步骤 55 中, 局部地址 A_Local 设为 $A_Mid \cdot 2^7 + A_S[E:S]$, 其中 $[E:S]$ 对于奇数 N 和 $N = 64$ 为 $[6:0]$; 对于偶数 N 但 $N \neq 64$ 为 $[N2 + 6:N2]$, 其中 $N = N_Odd * 2^{N2}$ (即, N_Odd
10 为 N 的奇数系数并且 $N2$ 为 2 的最大幂次除以 N)。

根据本发明的另一个实施例, 采用模乘法以产生准晶体映射, 对于各种明确的存储器访问方式用于最小化组冲突。在又一个这种实施例中, A 首先与 19 位常数 LL 相乘, 看乘积 $A \cdot LL$ 的位 $[36:18]$ 。在这种实施例的不同例子中, 采用如下的模乘法 (变换) 方法:

$$15 \quad A \rightarrow A \cdot 4161 + 127$$

乘积位 $[17:14]$ 和 $[19:18]$ 提供有效的组号, 同时乘积的更高位提供局部地址。

上面所示的实施例立即采用更多的门, 并且可能比图 1-4 中所示的实施例慢。在某些情况下, (例如, $N = 61$) 它们对于固定间隔访问
20 较好, 而对于其它 (例如, $N = 63$) 它们要差一些。

本领域的技术人员应当理解, 各种组合电路, 包括加法器 (例如, 在有限或无限域上多项式乘法的实现等)、查表和模乘法的变形, 或者甚至是模拟方案而不是模乘法可用来构成用在地址翻译单元中的与
25 在多组存储器的明确访问方式中减少存储器组冲突具有相同效果的有限准晶体映射 (包括, 但不限于在多维表中的固定间隔方式和/或线性访问方式)。

E. 容错工作

上述的重映射和加扰系统允许整个存储器系统容错工作, 以及片上多处理器和多组存储器的容错工作。

30 对于任何容错工作的先决条件是确定个存储器组或模块失效的存储器测试和/或控制这些组或模块 (单元) 的处理器单元。该测试可在存储器和/或处理器使用中的任何阶段进行——在元件制造后的初始测

试，或在寿命期和元件工作期间的任何时间。该测试可在外部（例如，通过外部存储器总线进行存储器访问的方法）或内部采用现有技术各种形式，例如串行或并行扫描路径、BIST（内建自测试）或产生测试图形用于全面测试的专用的片上电路。

- 5 一旦进行测试并发现缺陷元件——存储器模块、组或处理单元，必须存储缺陷信息，从而能够被重映射电路有效地利用。有很多普通的方法存储该信息。首先，在制造期间进行测试，缺陷信息可以在芯片中以硬连线的形式存储——但是，通常不能进行进一步的修改。其次，在测试之后，缺陷信息可以写入或下载到芯片上的 RAM 区。这
10 要求分别存储坏元件号。这种存储可在外部的 PROM 或其它机械可读形式（条形码、磁条码、系统存储等）进行。第三，在芯片上的专用非易失性区域可用于这种存储和后来的访问或进一步测试后的重新写入。第四，在许多情况下，测试和重新写入可完全由软件进行，例如，起动过程后软件测试存储器并将坏元件的列表下载到芯片上的
15 RAM（或寄存器）区中。在上面结合图 1 和 2 说明的无缺陷组表 6 特别适合于第三和第四种技术。本领域的技术人员应当理解，用于存储缺陷信息的其它技术也可以采用，并且本发明并不局限于上述特定技术。

- 关于缺陷元件的信息（例如，存储器组、单元或处理单元）可采用
20 多种形式。例如，如果预计只有几个缺陷元件，可存储缺陷元件号，用重映射确保这些元件号不被使用。在缺陷元件的数量很多的情况下，可采用简单的 RAM 存储好（无缺陷）元件的列表。作为这种实现的典型情况，考虑 64 个存储器组（单元）的系统，其中任何数量的组可以标记为有缺陷的，并且只有 N 个是好的（无缺陷的）。在图
25 1 中的无缺陷组表 6 是这种 RAM 的一个例子。如上所述，它含有存储每个有效的 6 位组号 Bank ($1 \leq \text{Bank} \leq N$) 的 64x6 RAM（或 ROM）阵列，Bank 是在芯片上的 N 个无缺陷组中的一个的实际号。

- 此外，缺陷或无缺陷组的列表可以采用各种用于 RAM 阵列的压缩技术进行压缩，从而显著地减少存储缺陷信息所需要的位数。例如，
30 位数可由 64x6 位（即，如果几乎所有的组都有缺陷，不压缩所需的最大位数）减少到最多 $W \times 6$ 位，其中 W 为最小的缺陷或无缺陷组的数量。虽然这种压缩产生额外的解压缩电路的成本（大约为 $O(W)$

个门), 以及翻译组号的额外的时间延迟。只有当在芯片上的小 RAM (ROM) 模块不能被有效的利用时, 并且在用存储阵列而不是寄存器来实现的情况下, 它可能是有用的。

其它参数的值可能也需要调整, 以提供容错工作, 例如 N 和 S 的值——即, 存储器组的数量和子组 (单元)、常数 L 和乘数 Λ 或 LL。N、S、L 和/或 Λ 的值可以与无缺陷单元的列表一起存储或下载。它们应当保存在有缓冲输出信号的快速寄存器中, 或直接硬连线的 (如果在制造期间进行了测试和更改)。可以下载或用于 L、 Λ 或 LL 的固定值的常数 L、 Λ 或 LL 的值可以是简单的硬连线。如果相当数量的模块失效 (大于 50%), 硬连线 L、 Λ 或 LL 会降低性能, 但也能显著地减少在图 3 中的 S-Box 12 和 M-Box 14 中的门的数量。

存储有缺陷或无缺陷元件的列表的位置依赖于放置在芯片上的存储器系统的类型。在如图 7 所示的表示具有用于所有存储器元件的单个存储器总线的单个存储器模块的芯片中, 或如图 8 所示的具有可切换总线的单个存储器模块中, 一个位置用于存储列表 (无缺陷组表和地址翻译参数值), 一个位置用于重映射和加扰电路, 即, 存储器区 2 和翻译单元 1。(在图 7 中, 无缺陷组标记为 60, 有缺陷组标记为 61。)

但是, 在片上系统中, 如图 9 和 10 所示, 可能多个处理器访问多个存储器单元和组, 有缺陷或无缺陷单元的列表以分布方式与各处理单元存储在一起 (或其簇)。此外, 重映射和加扰电路与各处理单元 (或它们的簇) 放置在一起。这显著增加了实现重映射和加扰电路所需的最少的门, 并使图 1-6 的实现为最佳。图 9 示出了多处理器和由单元 70 组成的存储器系统。图 10 示出了各单元 70 中与存储器相关的部分。翻译单元 81 和存储区 82 控制对局部存储单元 (子组) B1-B8 83 的访问。开关 80 为局部和全局地址以及控制位的通道, 并为存储器数据信息进出各单元 (70) 往来于整个片上系统的通信开关提供通路,

提出重映射电路 (或其软件实现) 的目的是允许有许多存储器和处理元件的大系统的容错工作, 其能容忍许多失效的存储器或处理元件而不降低系统性能。为以降低的是适当减小的可用存储容量 (或处理器性能)。地址翻译和加扰单元确保与可用 (无缺陷的) 存储器系统相同质量的存储器访问。

此外，如果在芯片上要求具有固定数量的存储器组和/或处理单元，提出的容错解决方案是在芯片上增加一定数量的额外（称作空闲或保留）存储器组和/或处理单元。这种空闲组或单元的数量由成品率和工艺因素决定并且可以变化。此外，也可以配置系统使某些存储器组5 组优于其它原因而被忽略。在这里公开的系统允许这种可变性。具有空闲组或单元的备选方法对用户有效地隐藏了在芯片上不可避免地存在的缺陷和失效元件。此外，在不允许出故障的关键性任务（mission-critical）的应用中，比如说3个处理器/存储器单元执行相同的任务，通过比较，一个可以对芯片的内部工作进行连续的检验。10 这允许对各元件进行动态运转检查，允许他们离线运转而不对芯片的工作造成任何中断。与空闲的概念结合，这能保持芯片对于在其寿命期和工作期间的故障能够不降低性能的工作。

在上述实施例上进行许多变形对于本领域的技术人员是可能的和显然的，并且本发明并不限于这些实施例。例如，在上述实施例中示15 出的电路对于特定的硬件可以很容易地改变和优化，并且可以包含更多或更少的电路和元件。而且，除了以片上电路和分立元件组合的形式硬件实现以外，本方面可以将虚拟地址访问重映射到物理地址或重新组织的访问重映射到程序可用的可变存储器阵列，从而用软件实现。此外，虽然本发明的某些实施例结合19位（字或高速缓存线）20 地址空间和特定最大数量的组和子组进行说明，但是本发明可以很容易地适应更大或更小的地址空间和更大或更小的可变数量的组和子组和处理器单元。

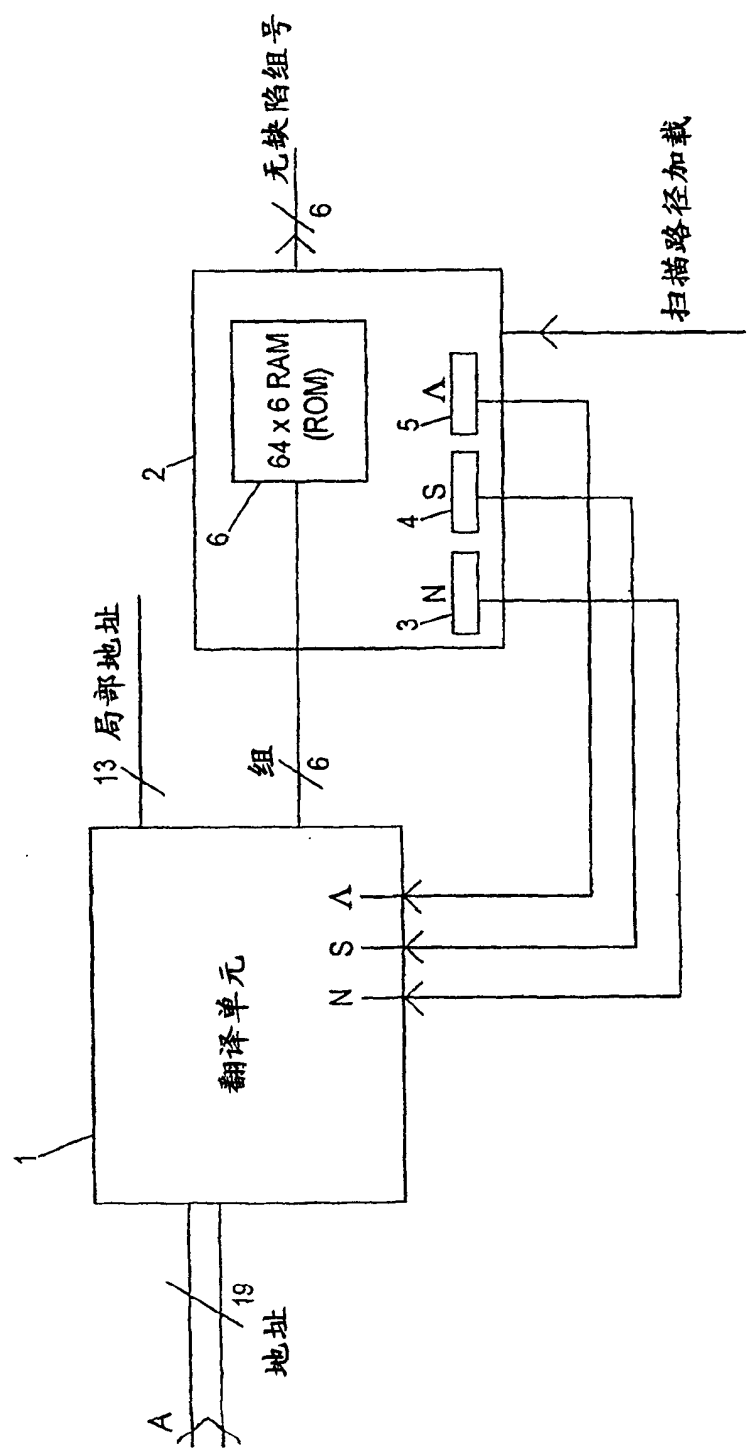


图 1

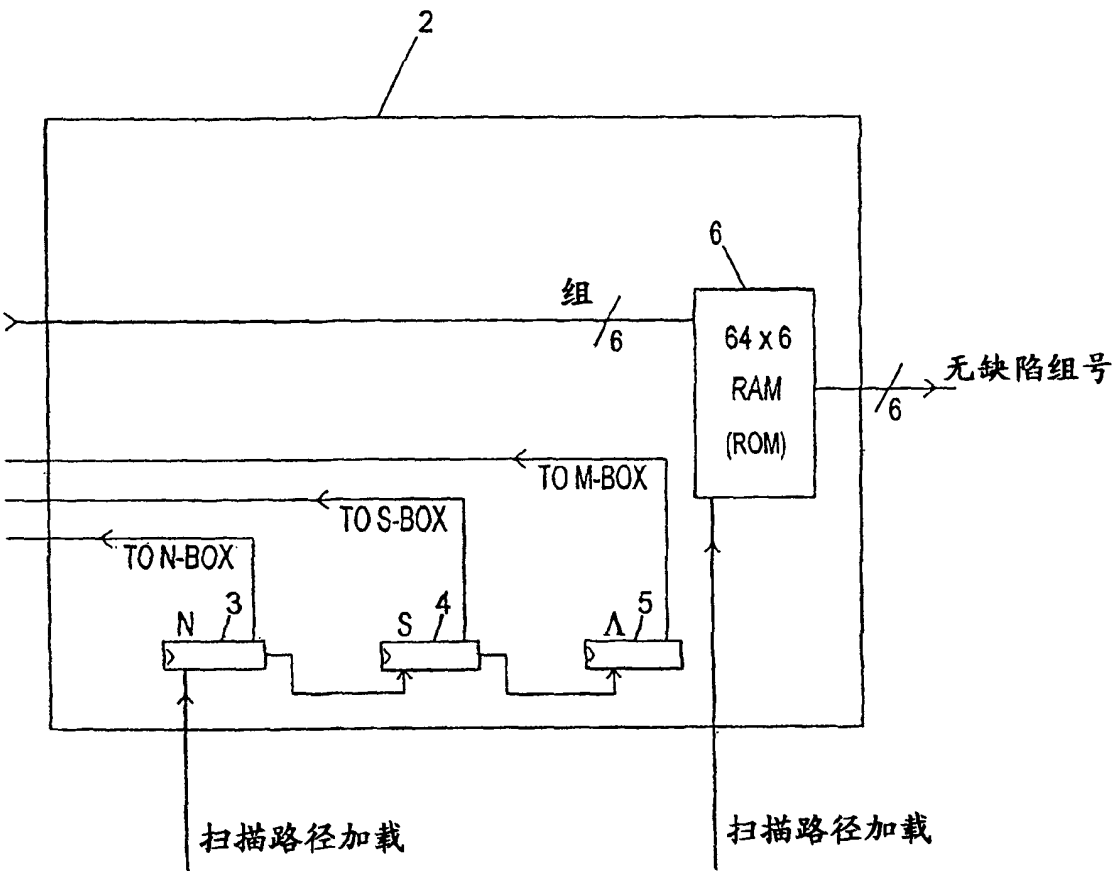


图 2

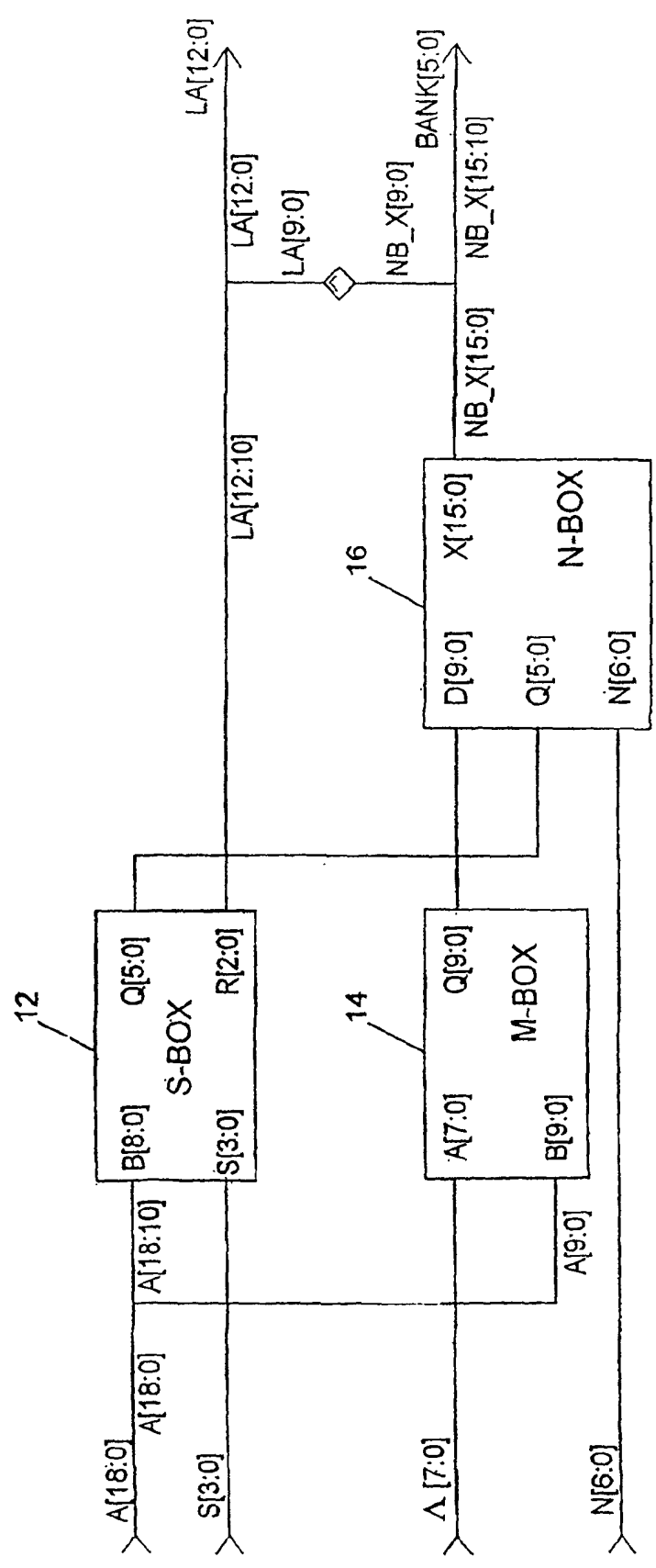


图 3

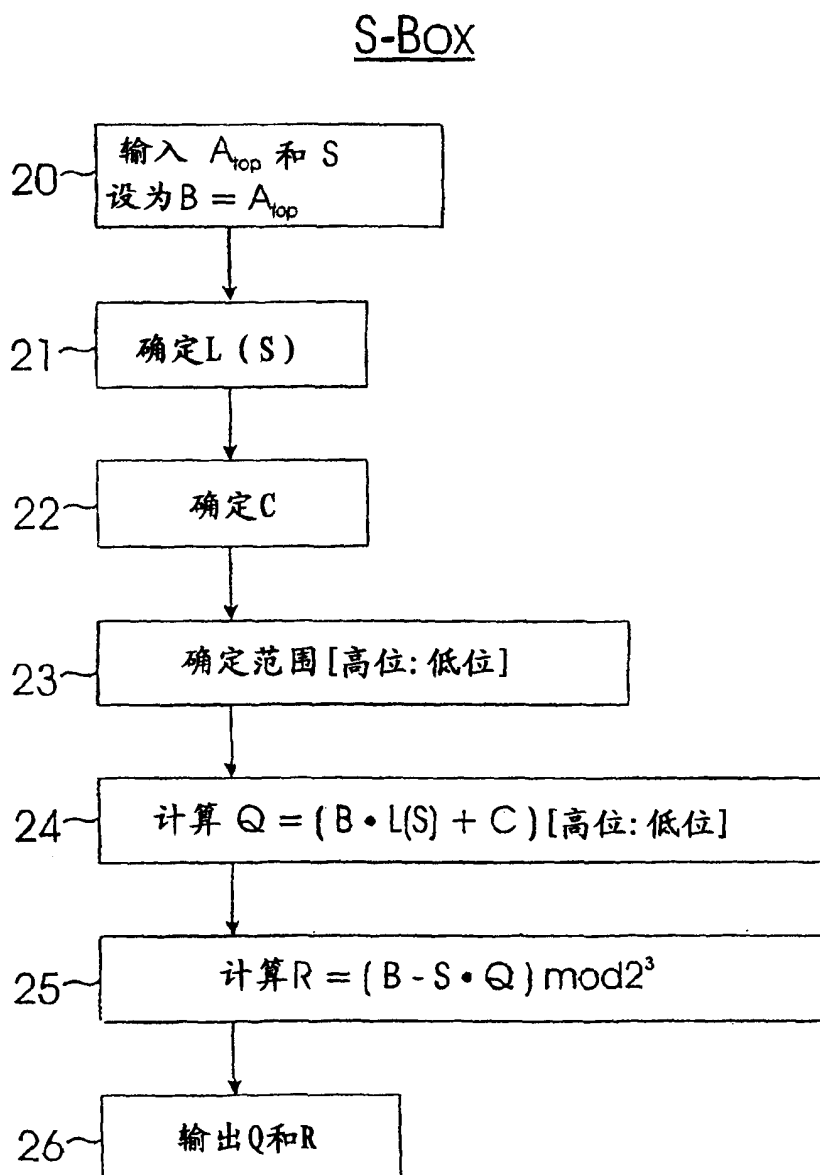


图 4

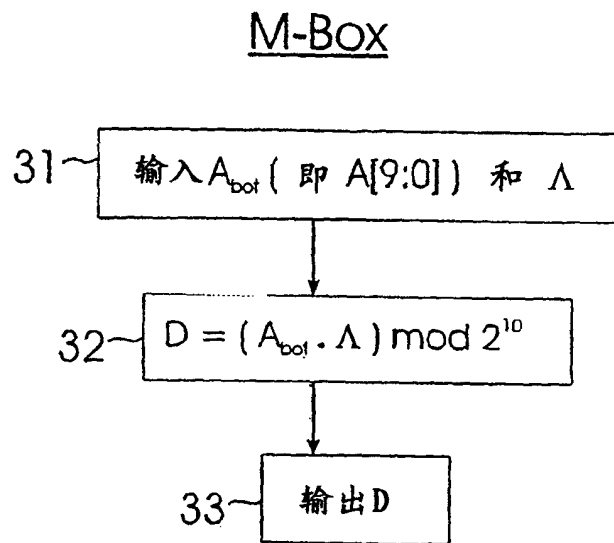


图 5

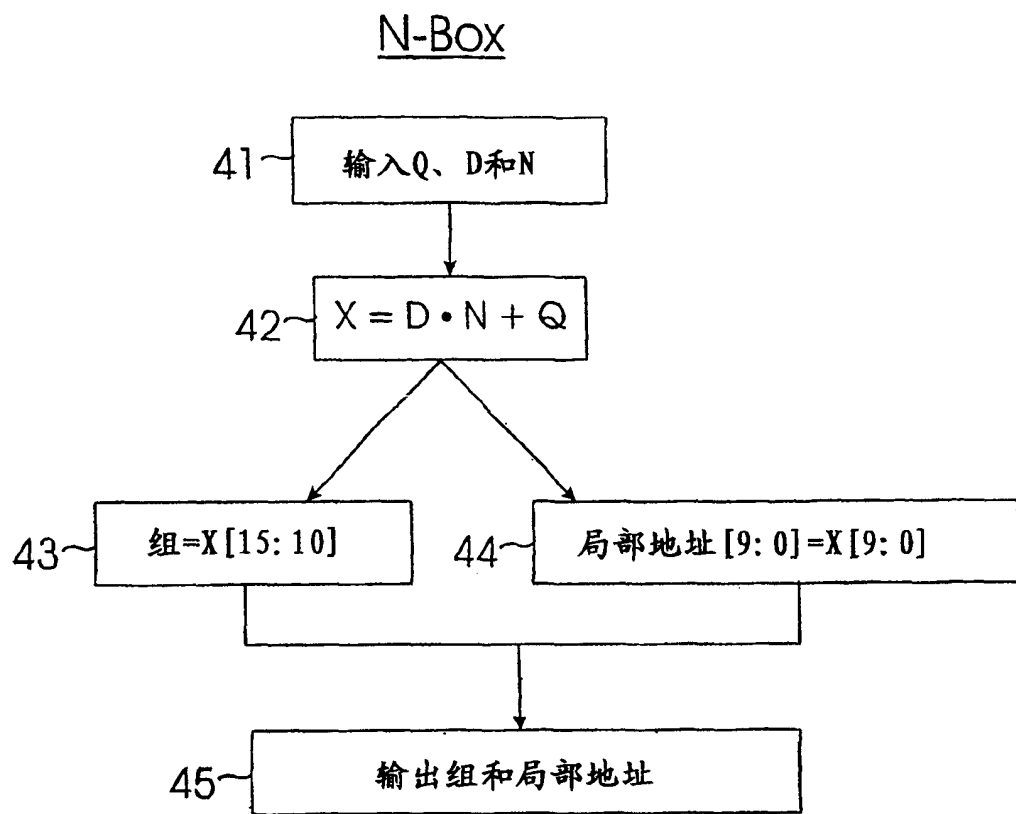


图 6

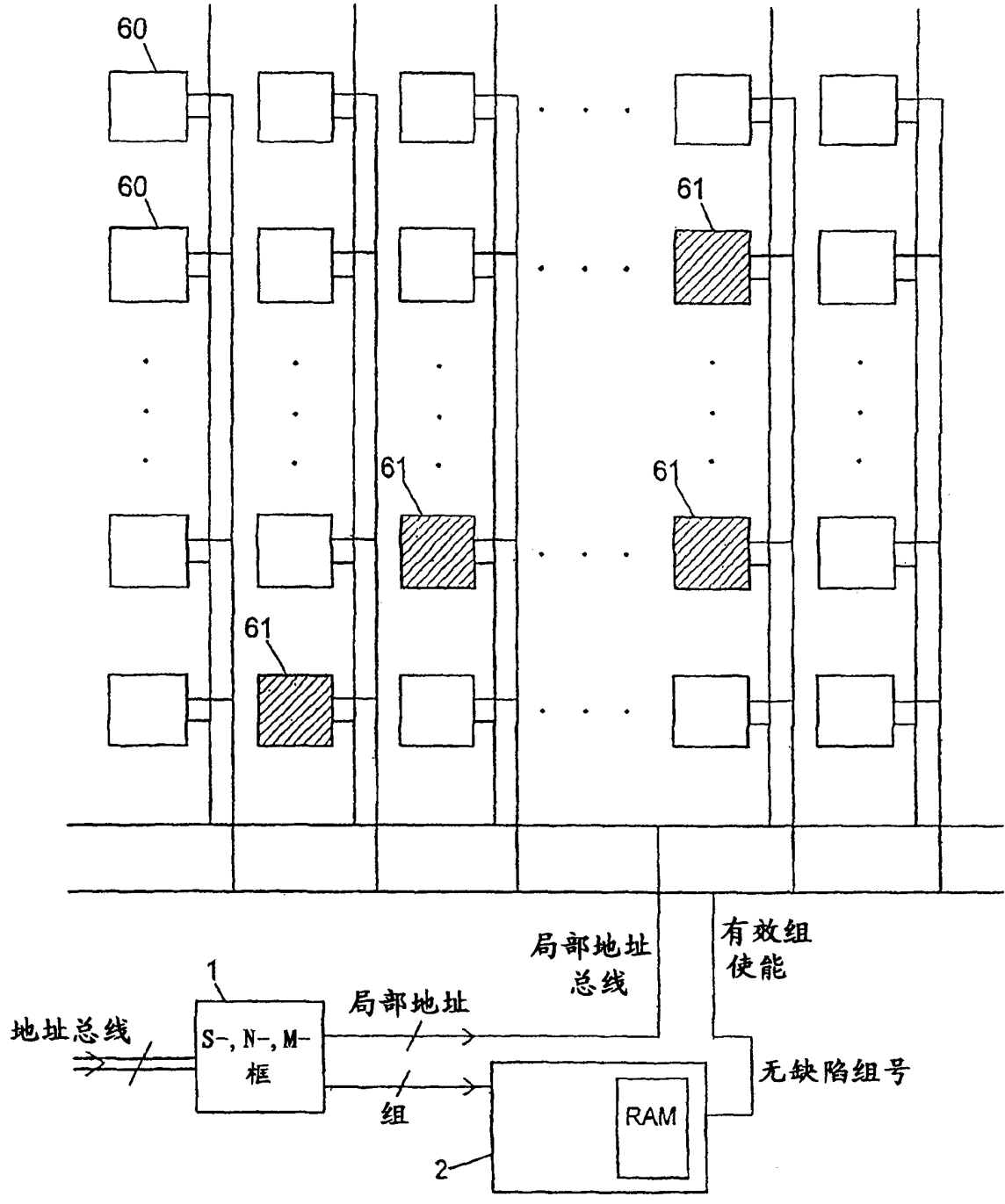


图 7

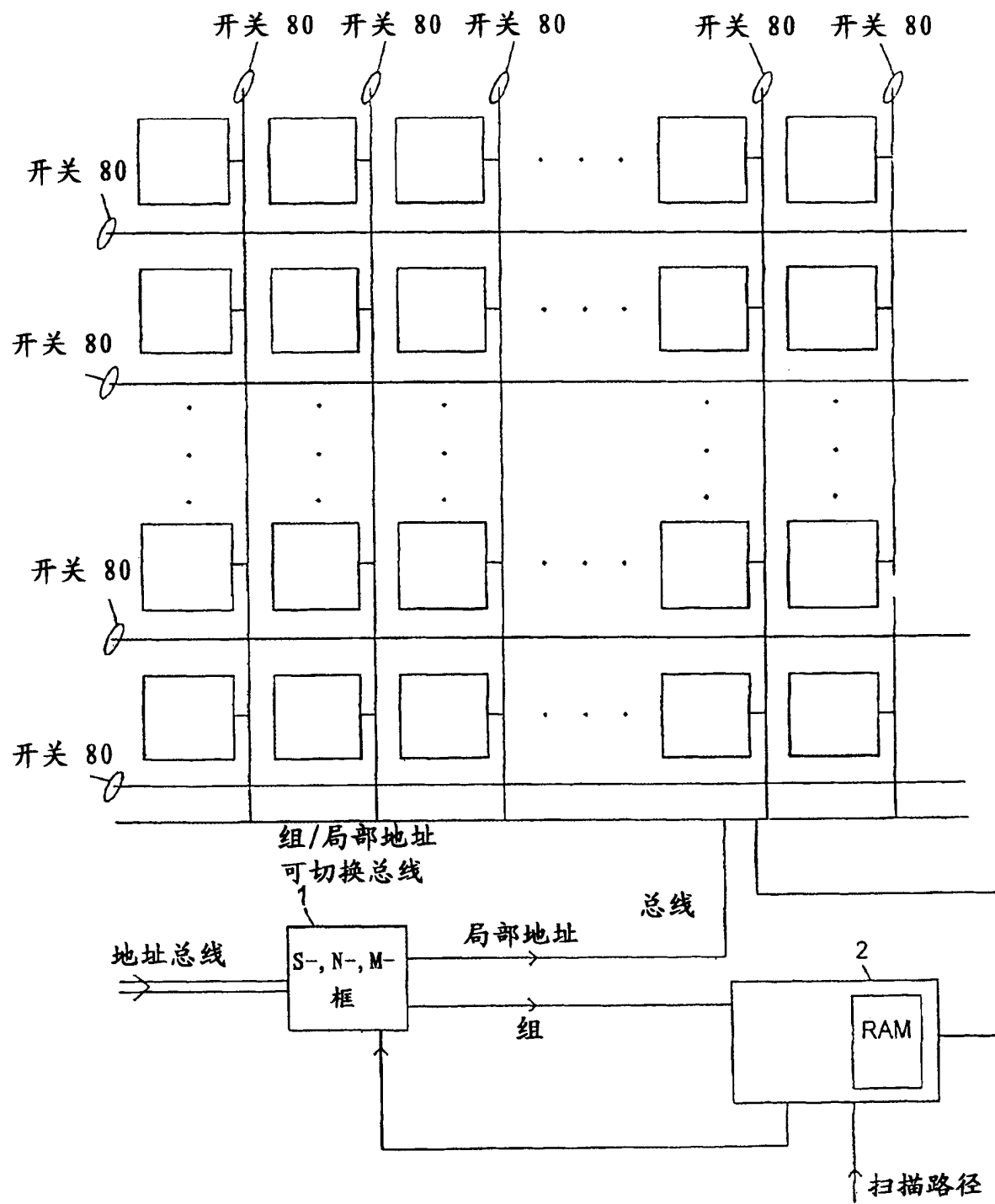


图 8

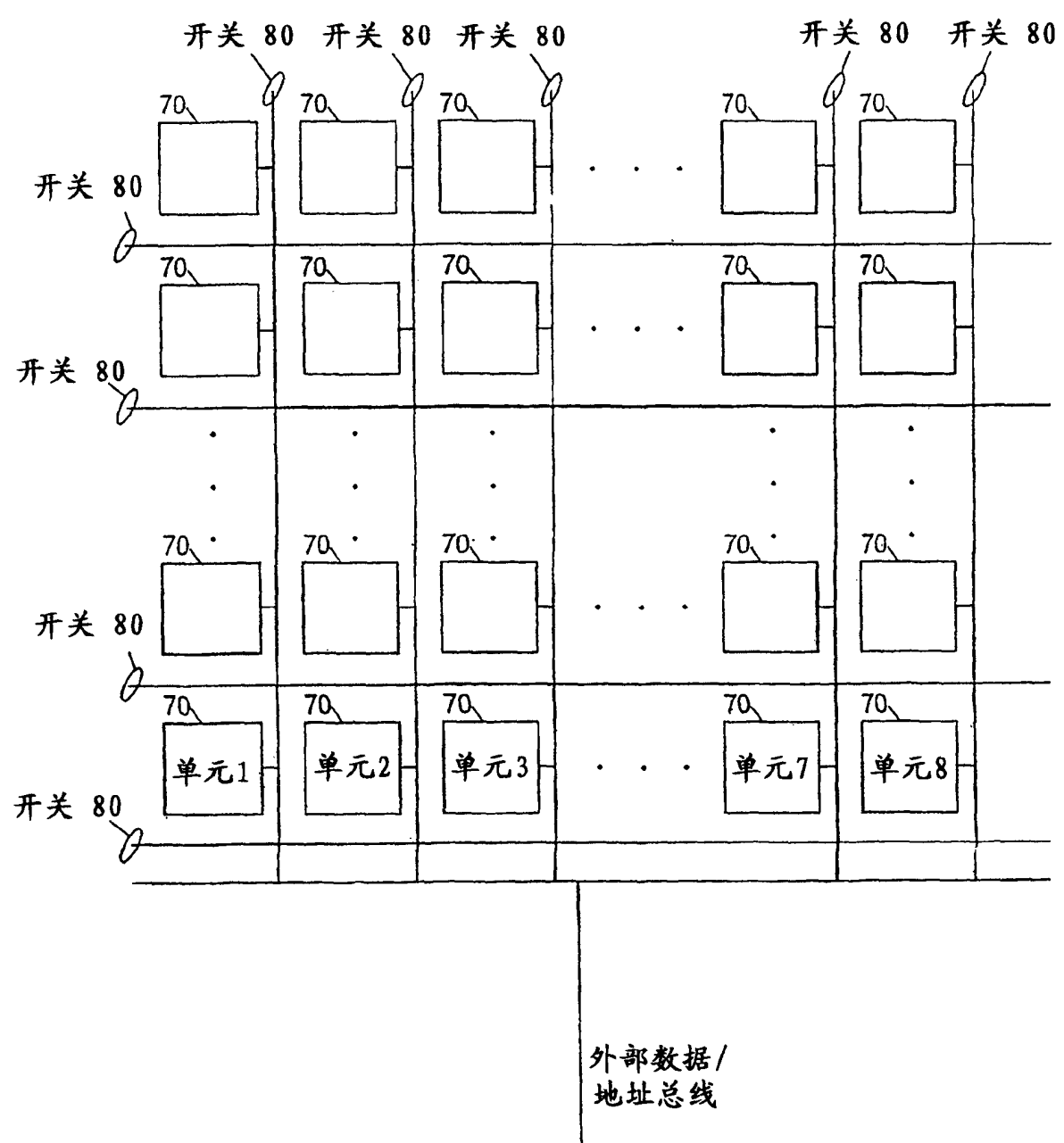


图 9

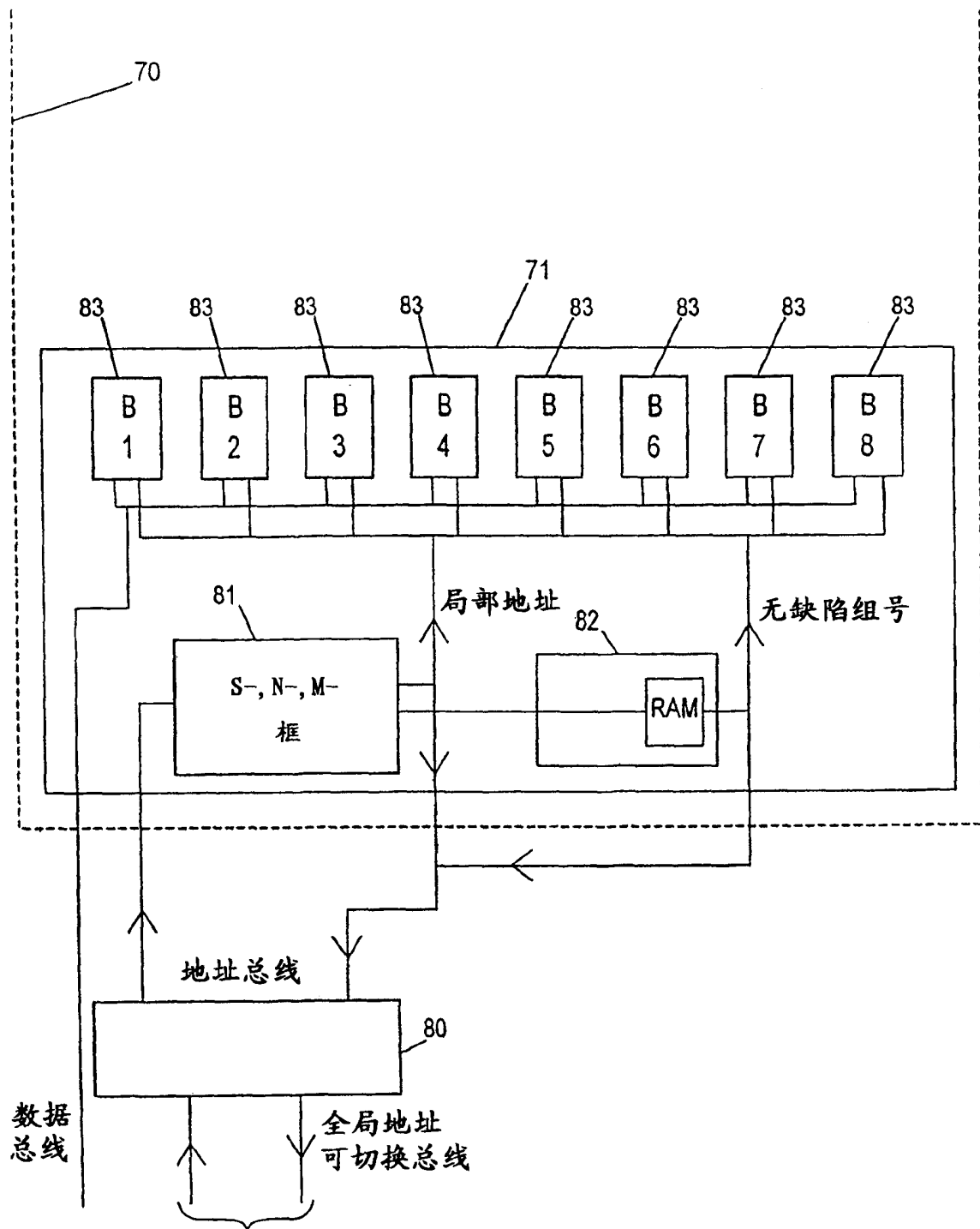


图 10

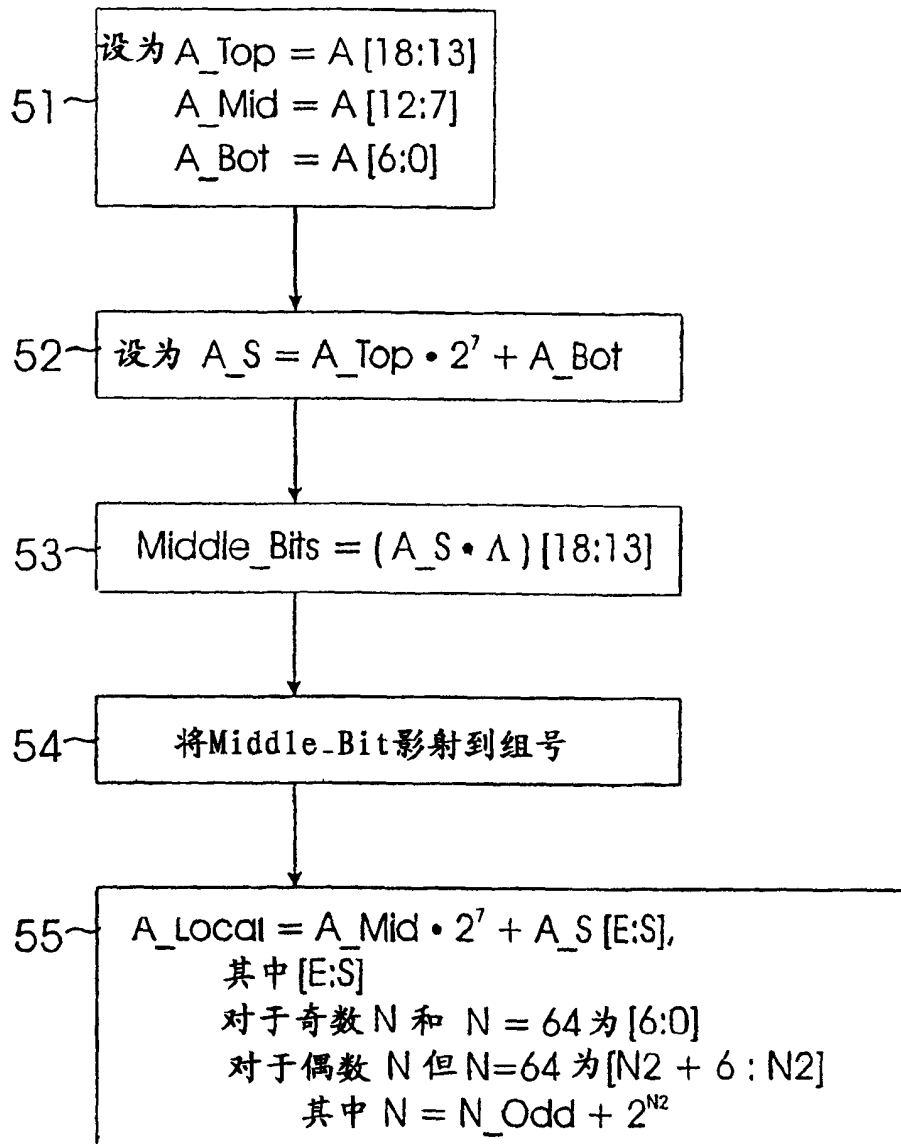


图 11

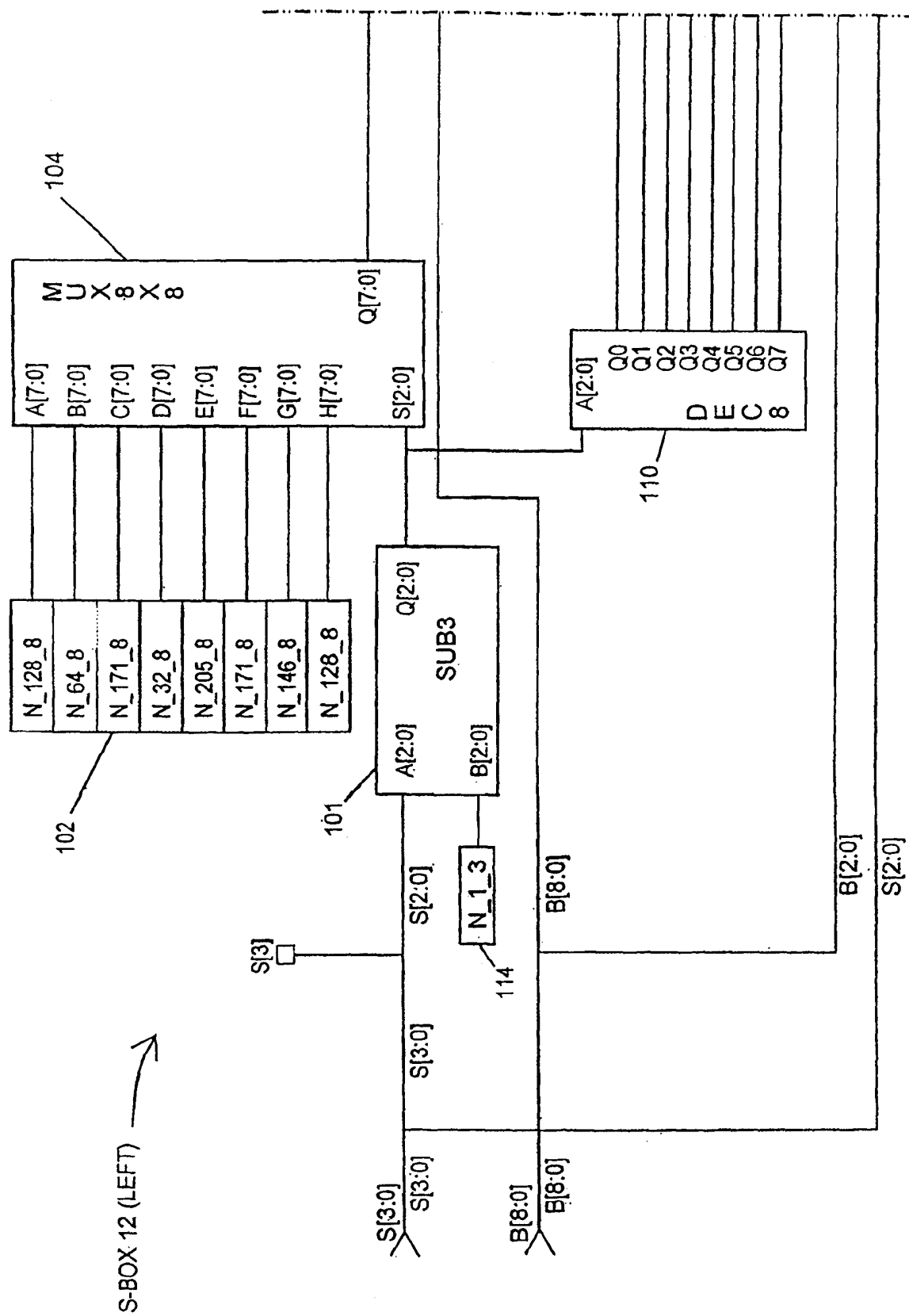


图 12A

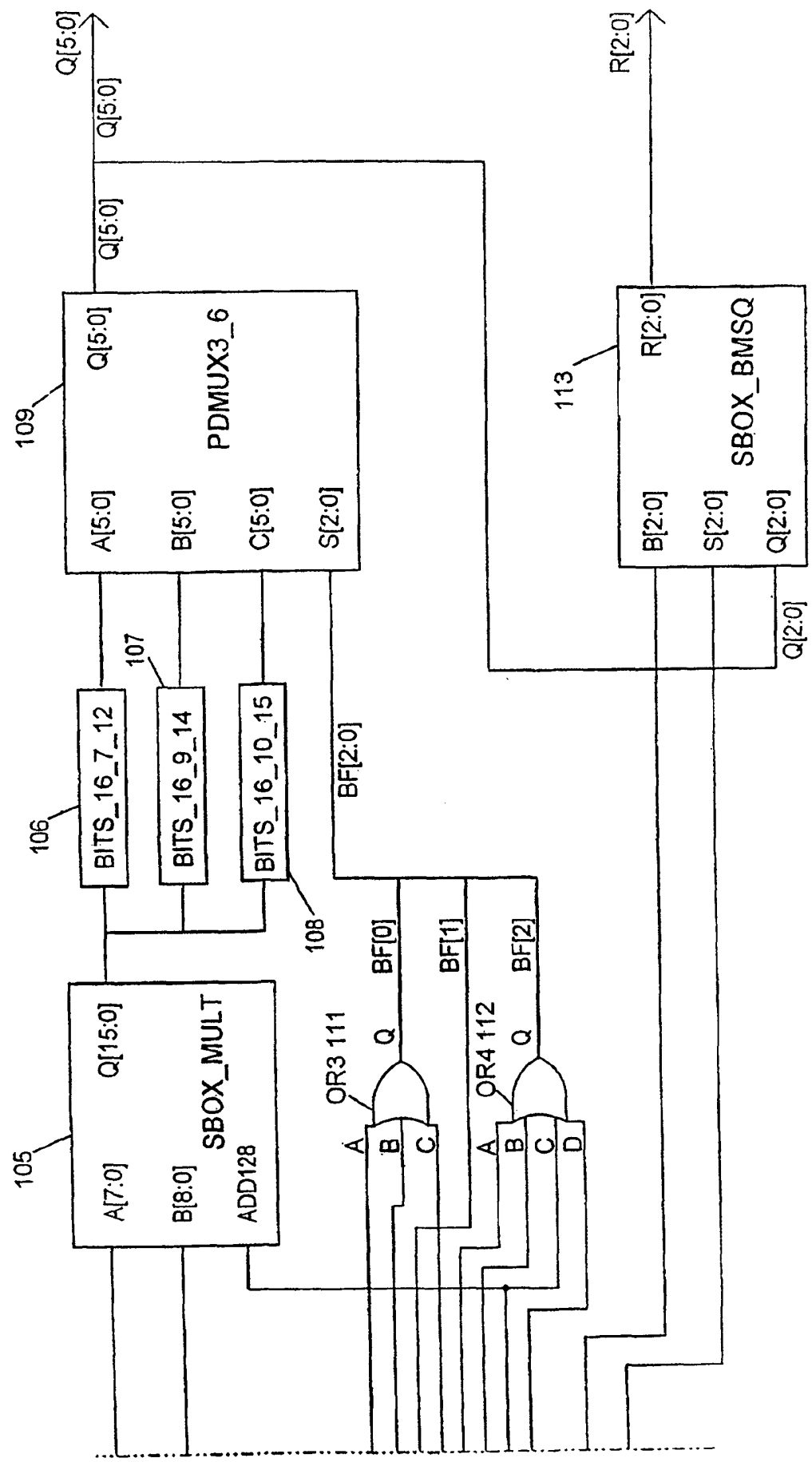
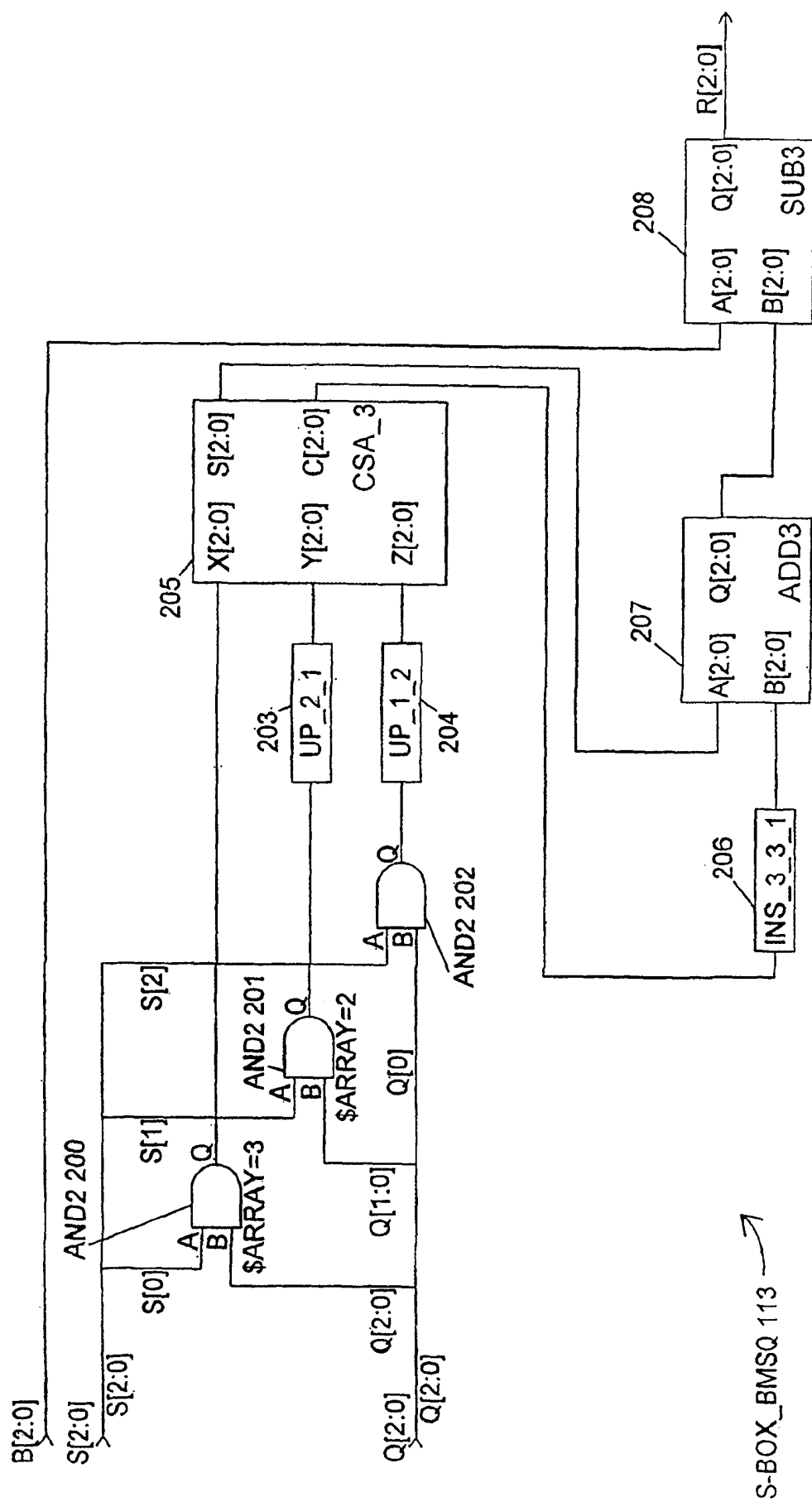


图 12B



13

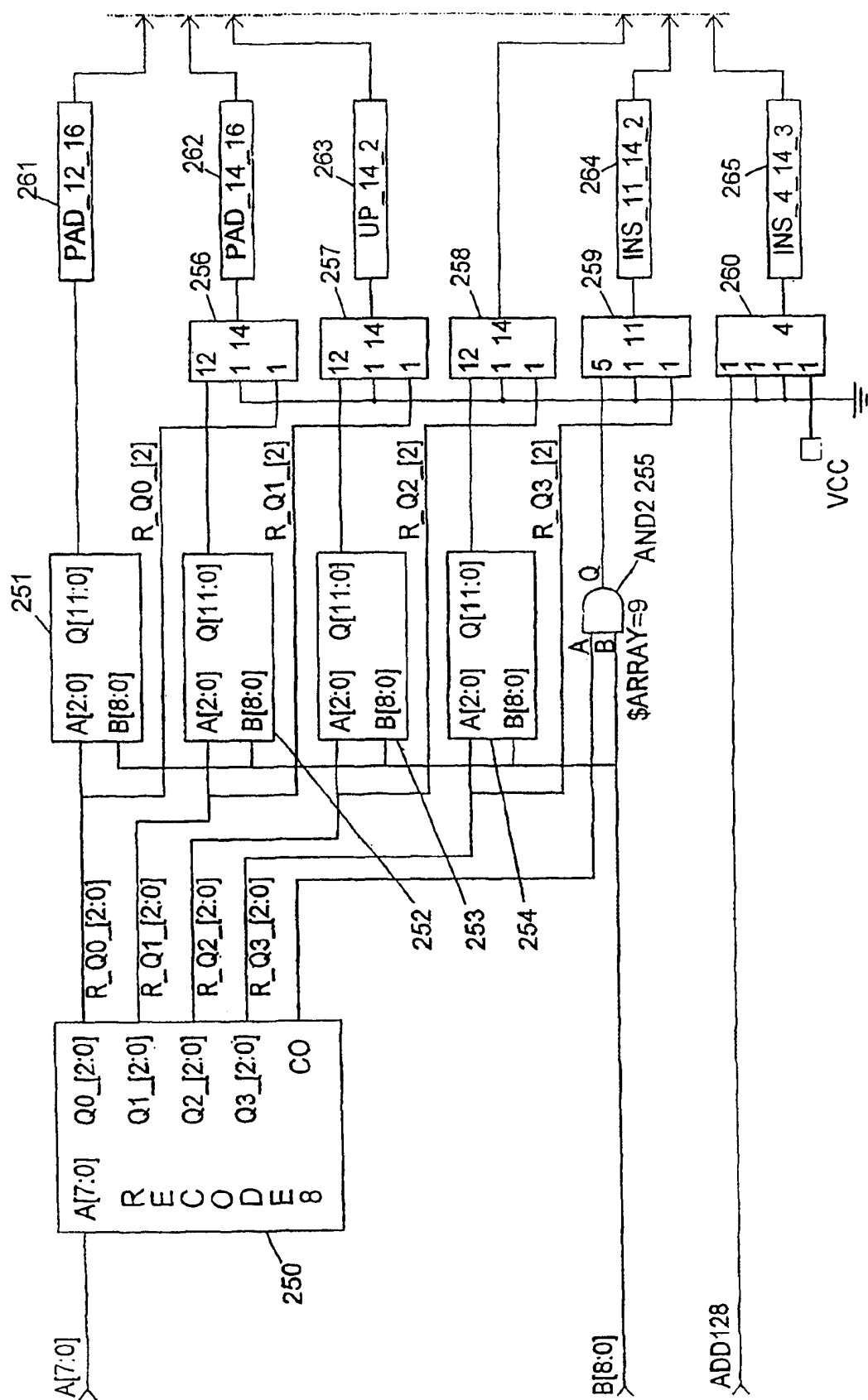


图 14A

SBOX_MULT 105 (左) →

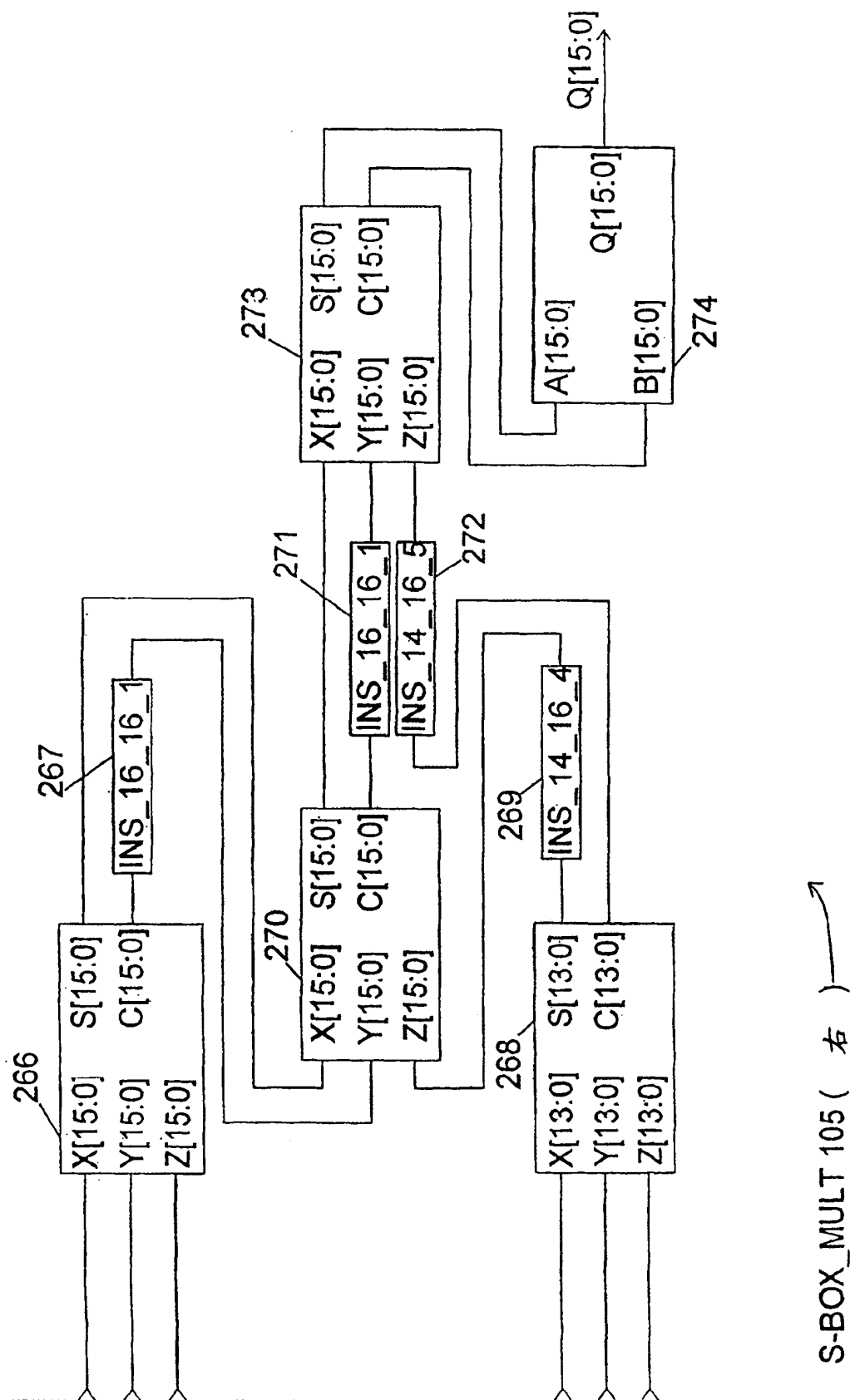


图 14B

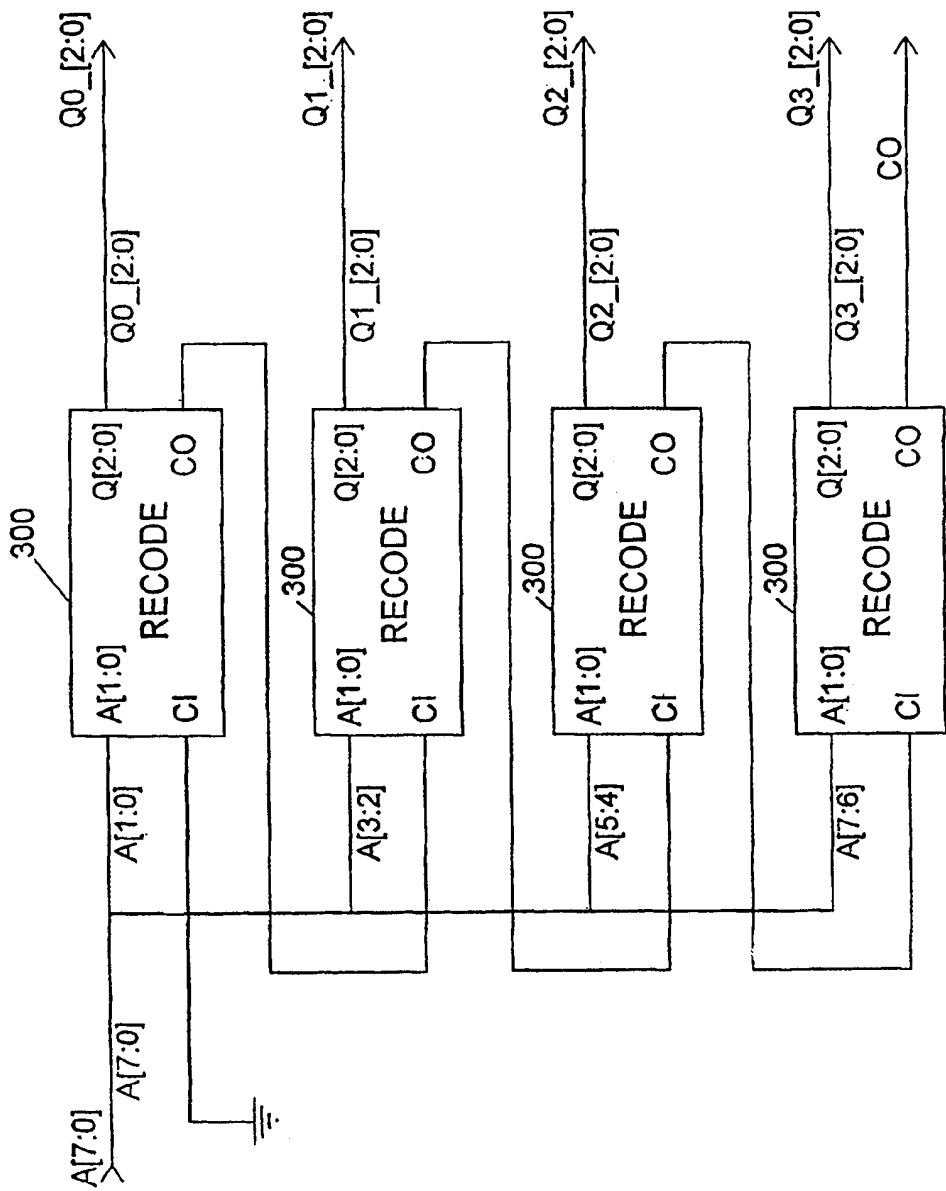


图 15

RECODE8 250 →

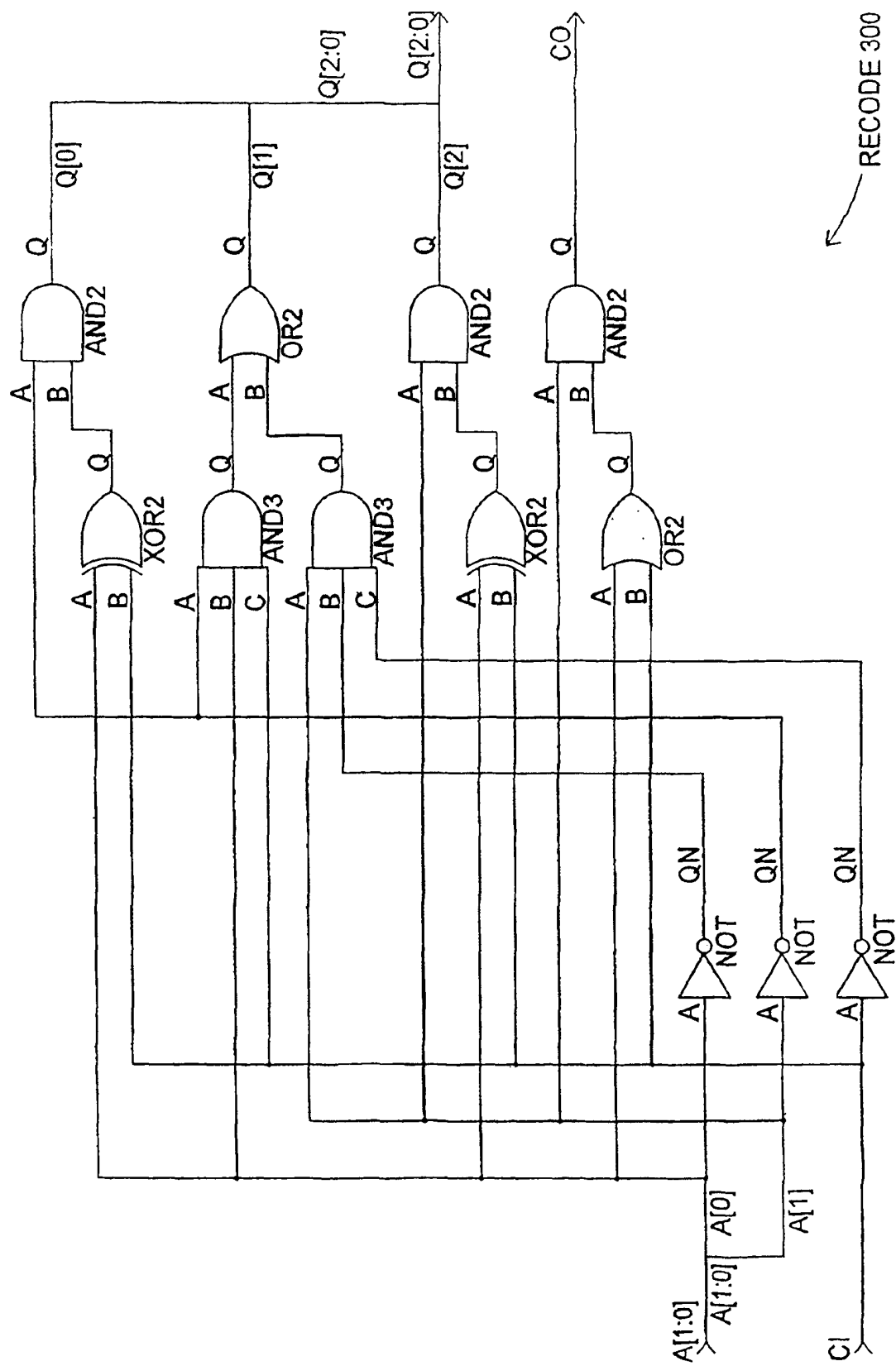
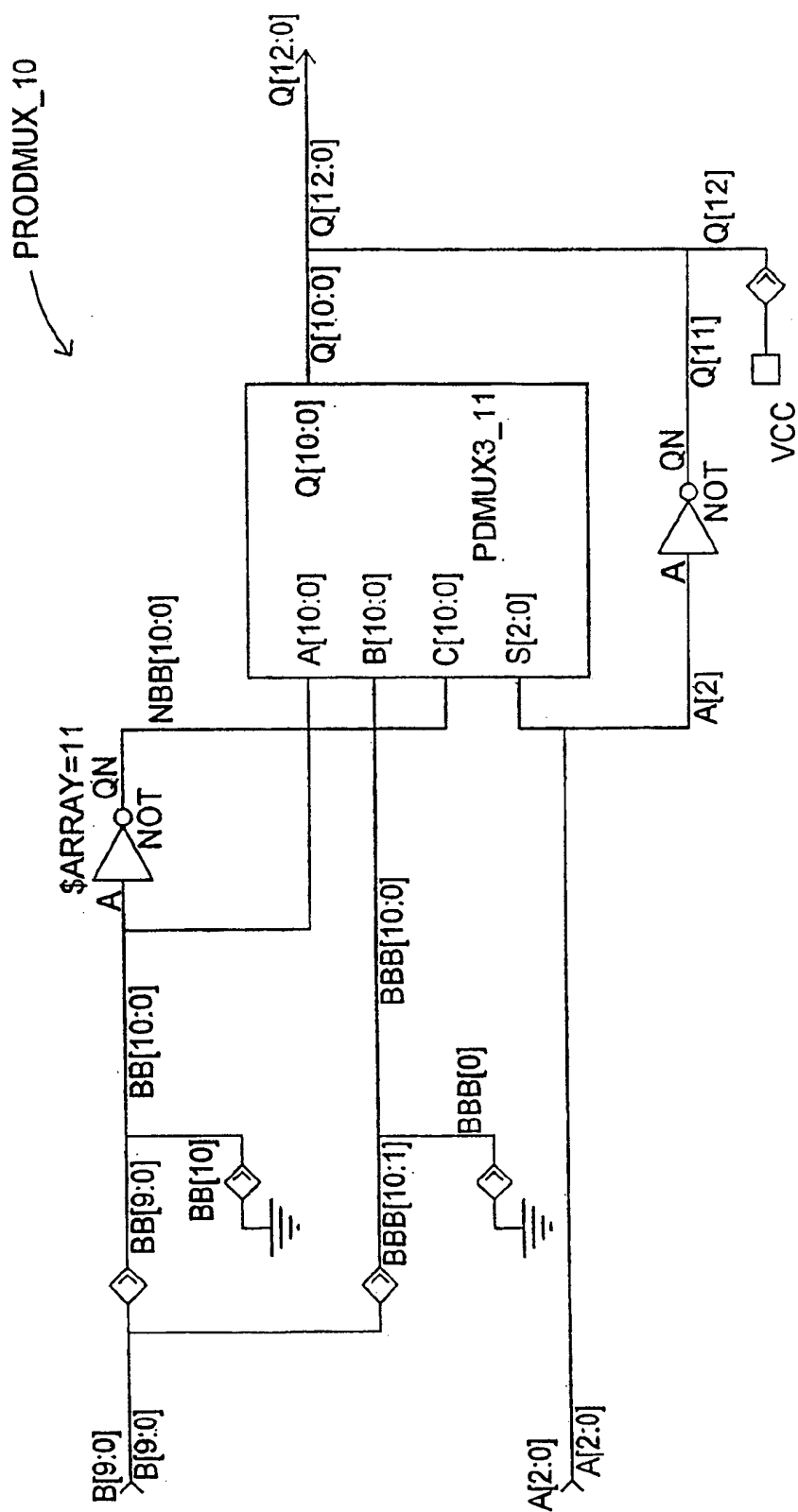


图 16



17
四



图 18

图 19

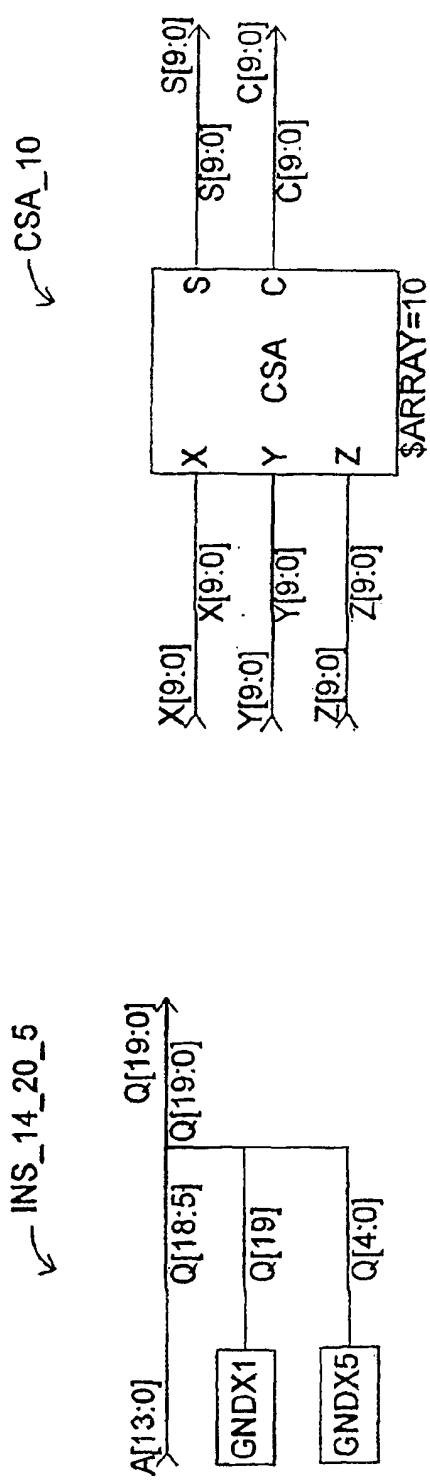


图 20

图 21

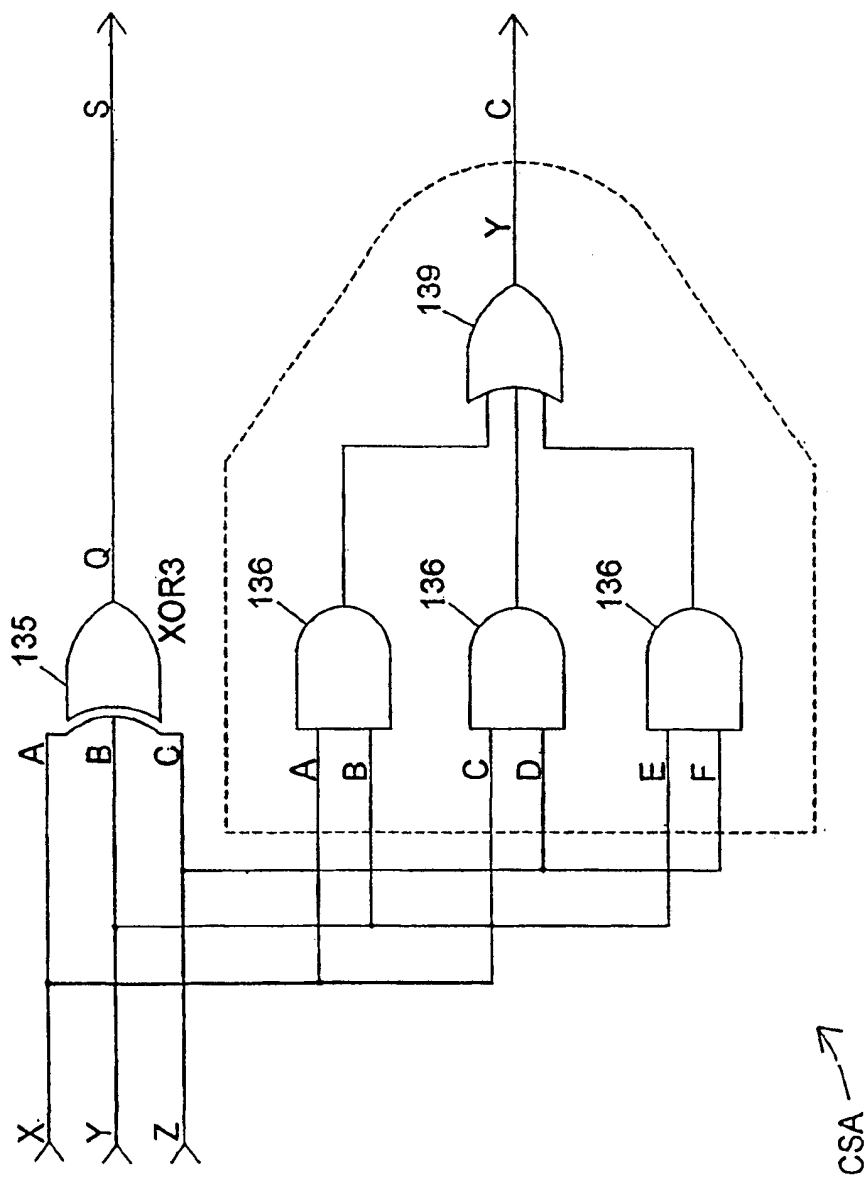


图 22

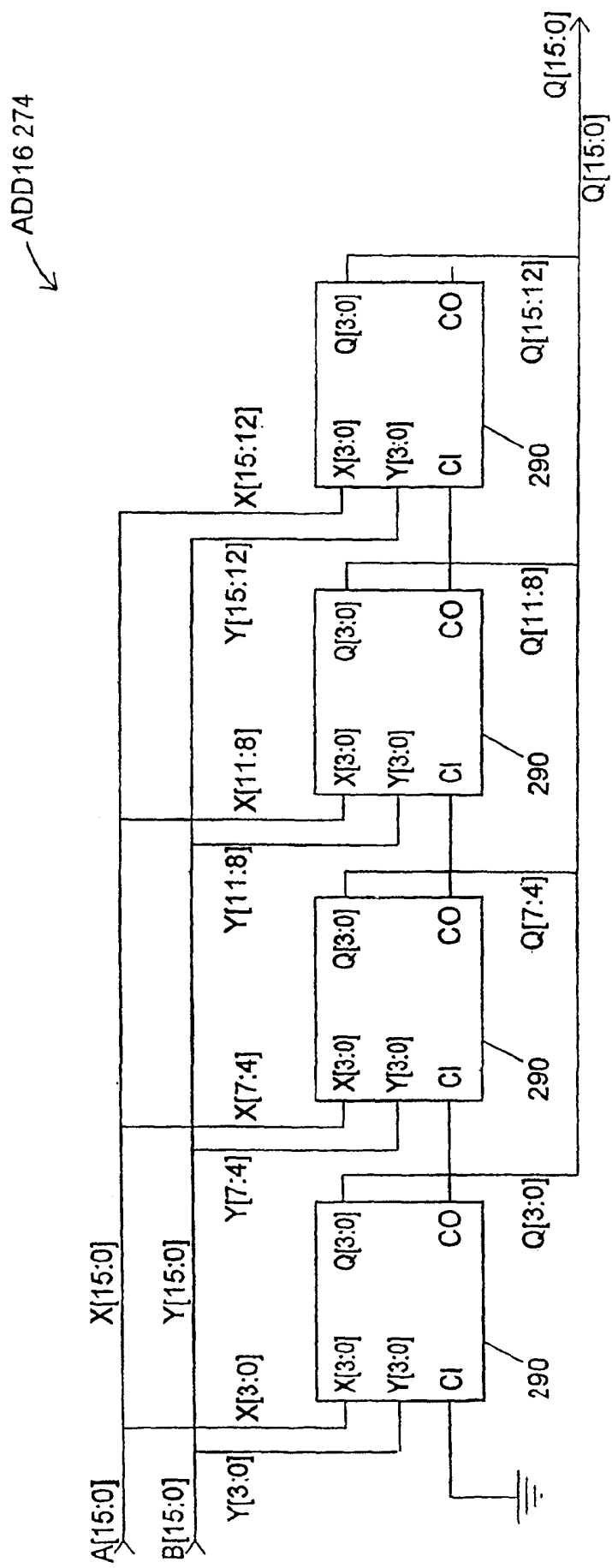


图 23

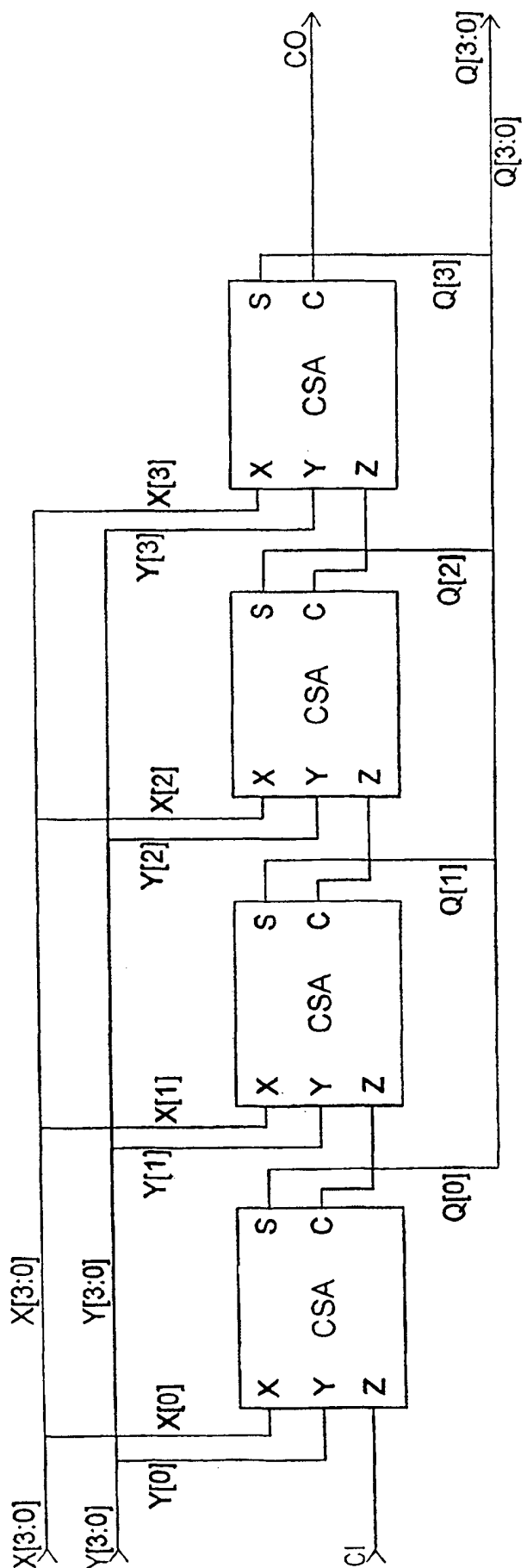
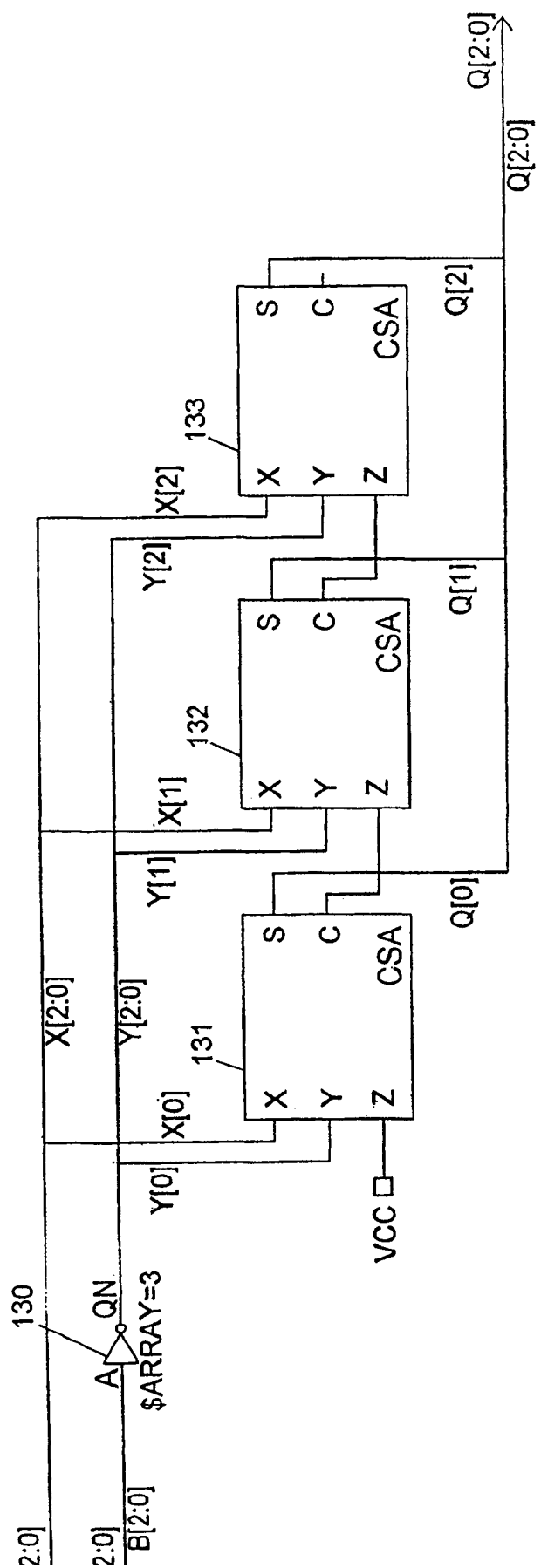


图 24

TRADD4 290→



SUB3 208 →

图 25

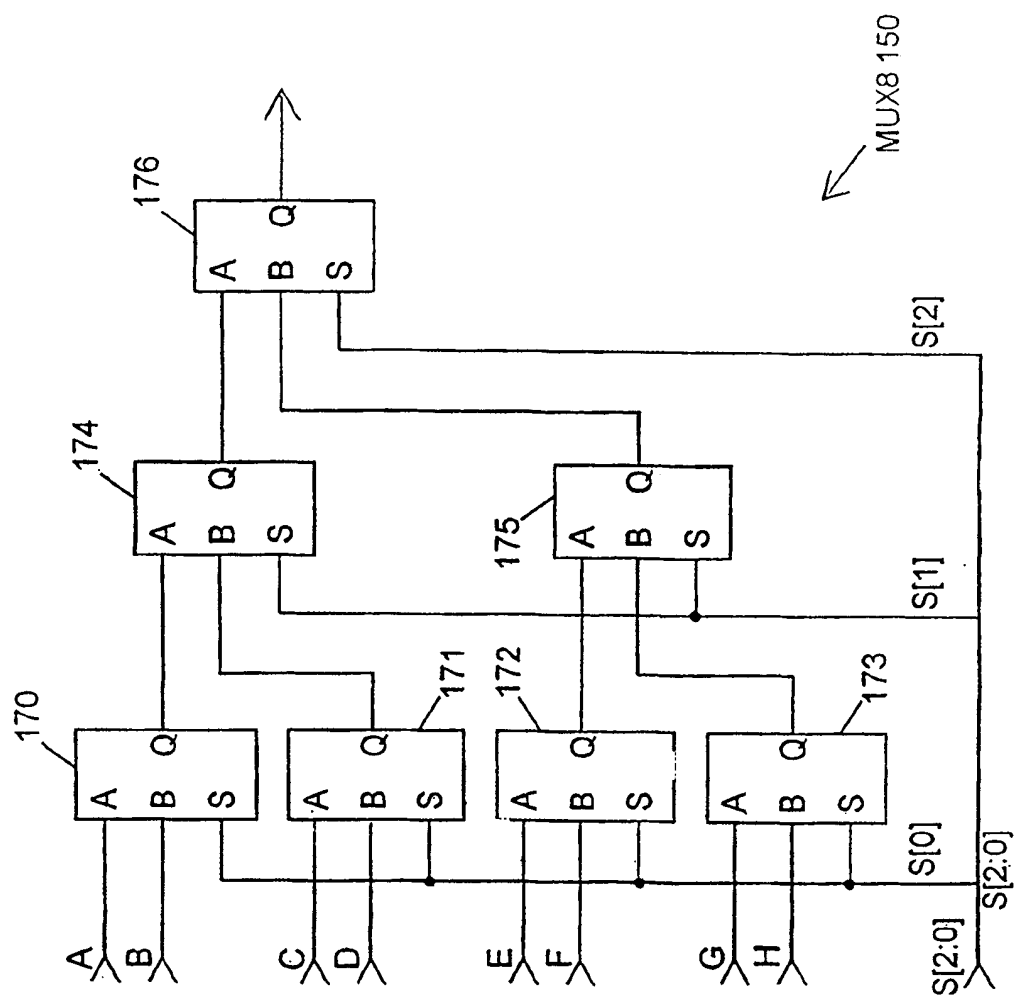


图 26

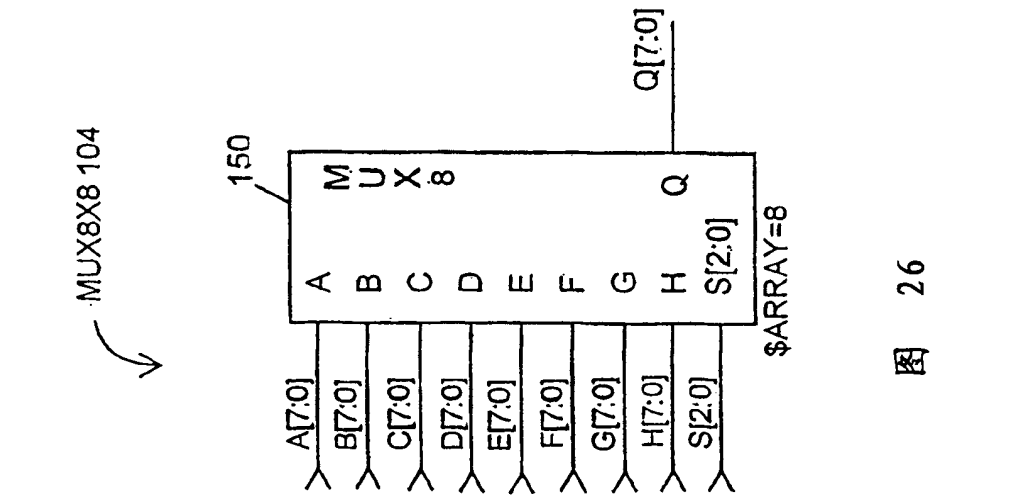


图 27

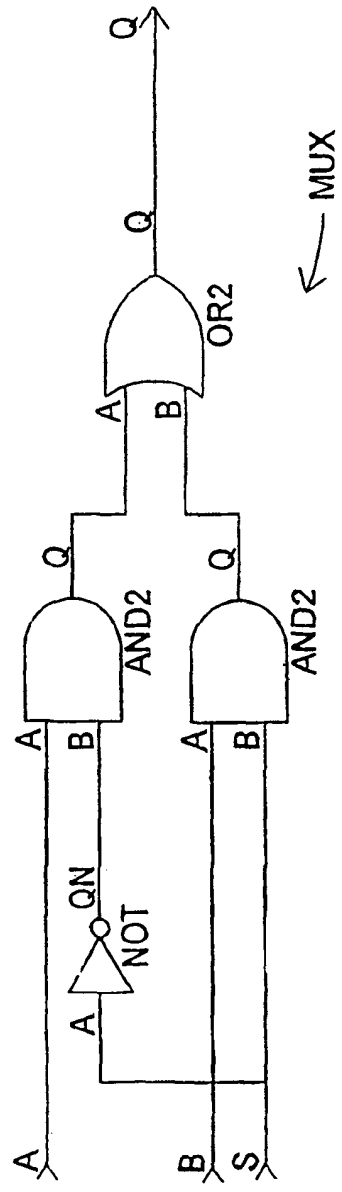


图 28

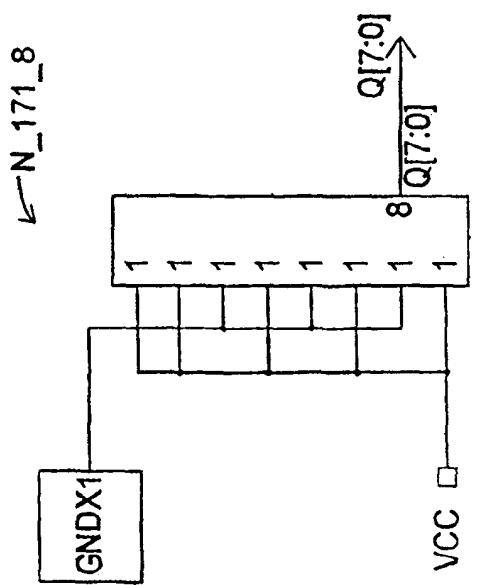


图 29

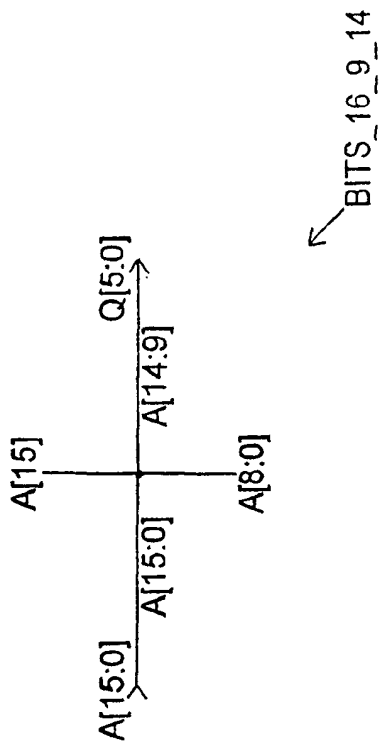


图 30

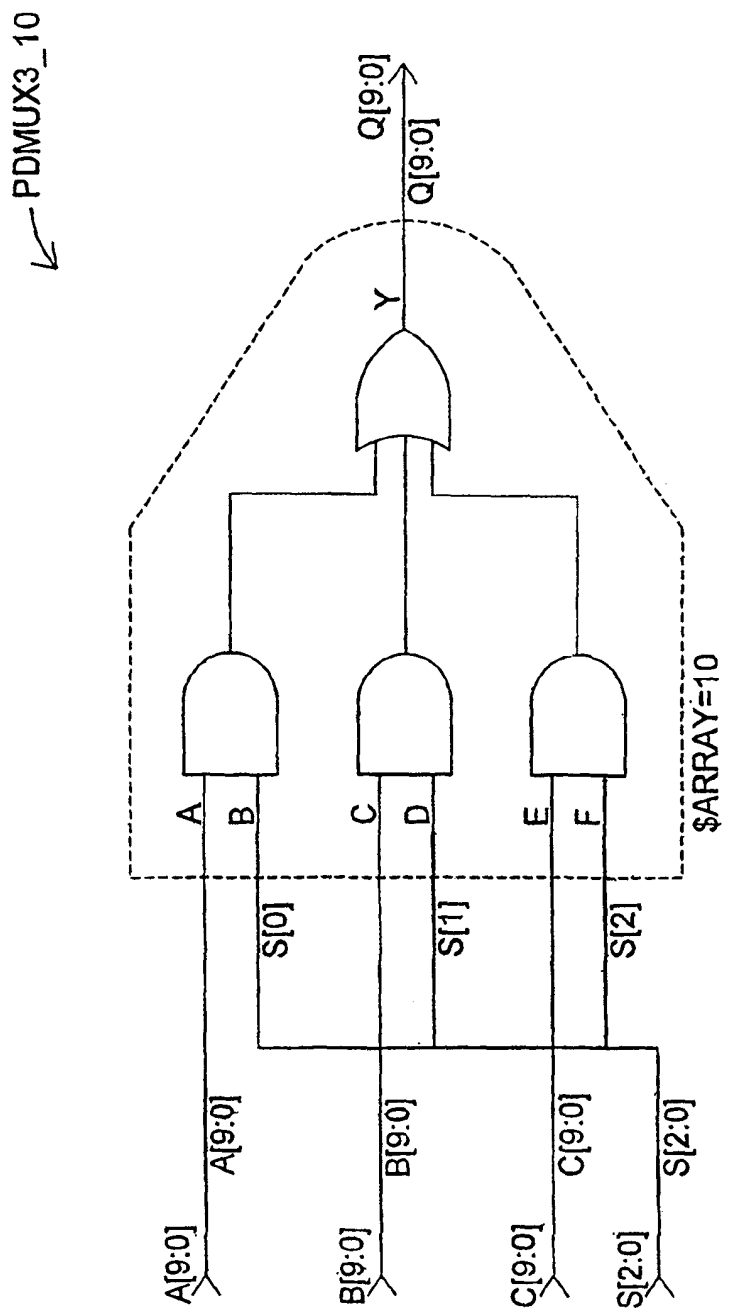


图 31

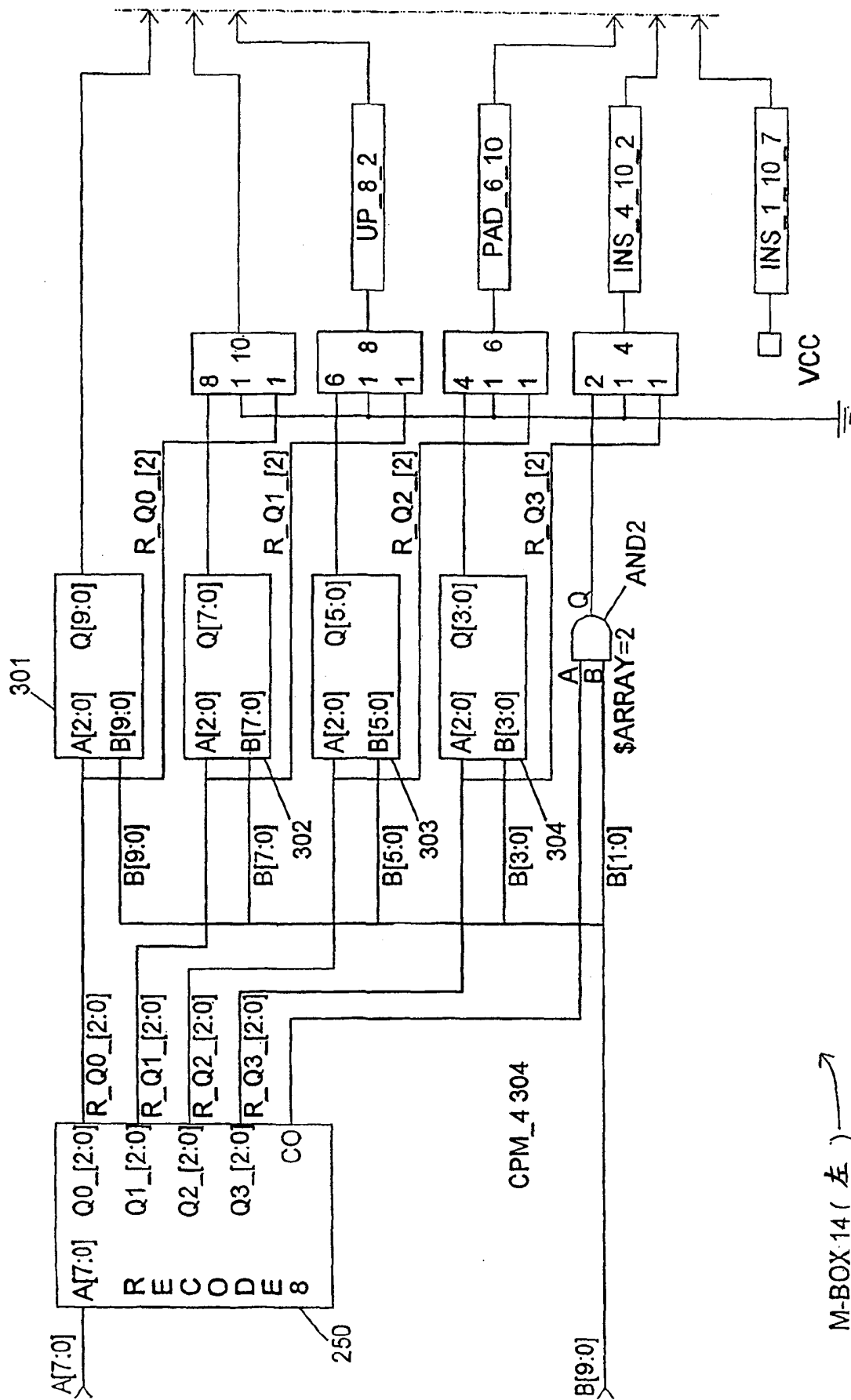


图 32A

M-BOX 14 (左)

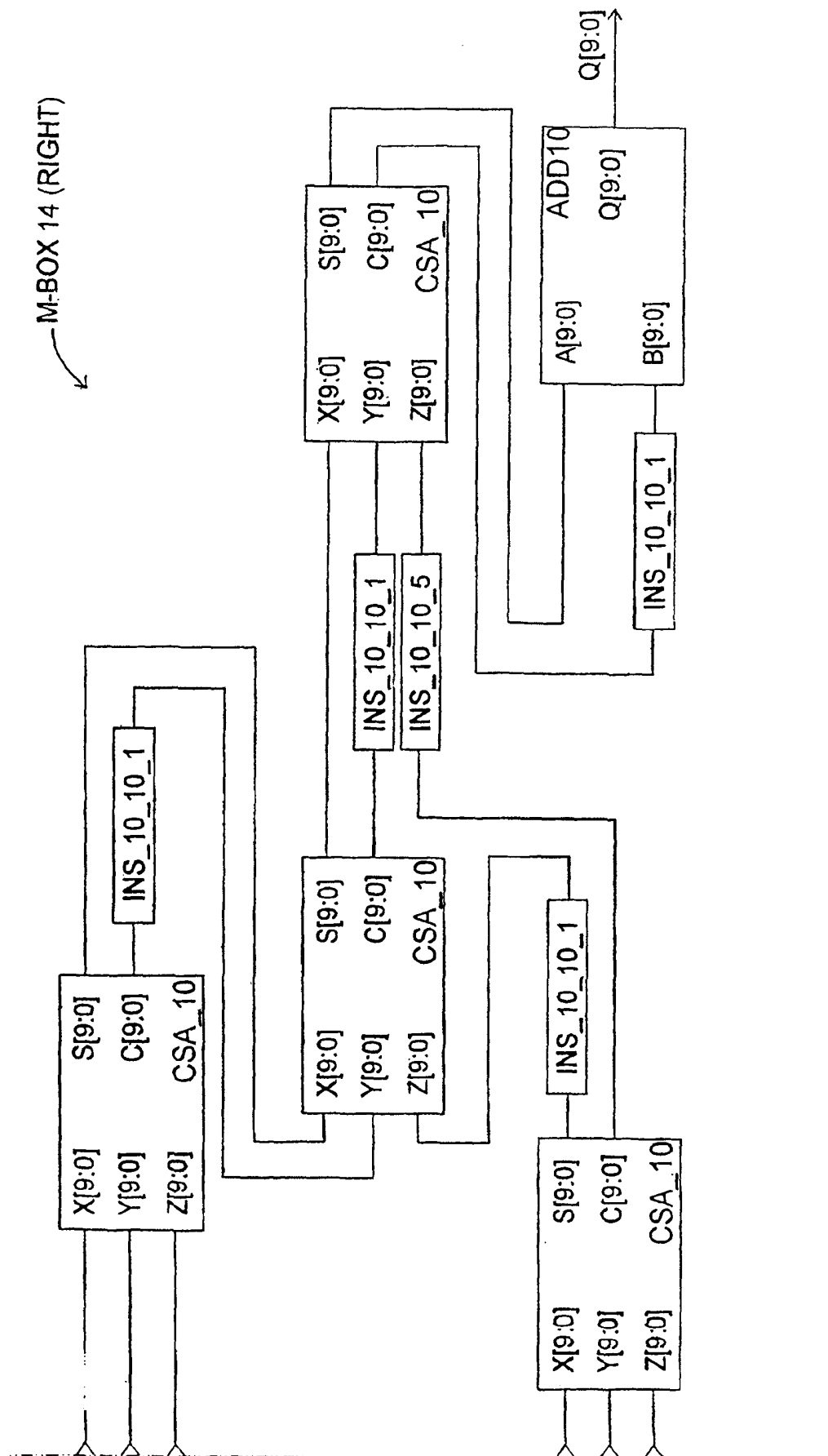


图 32B

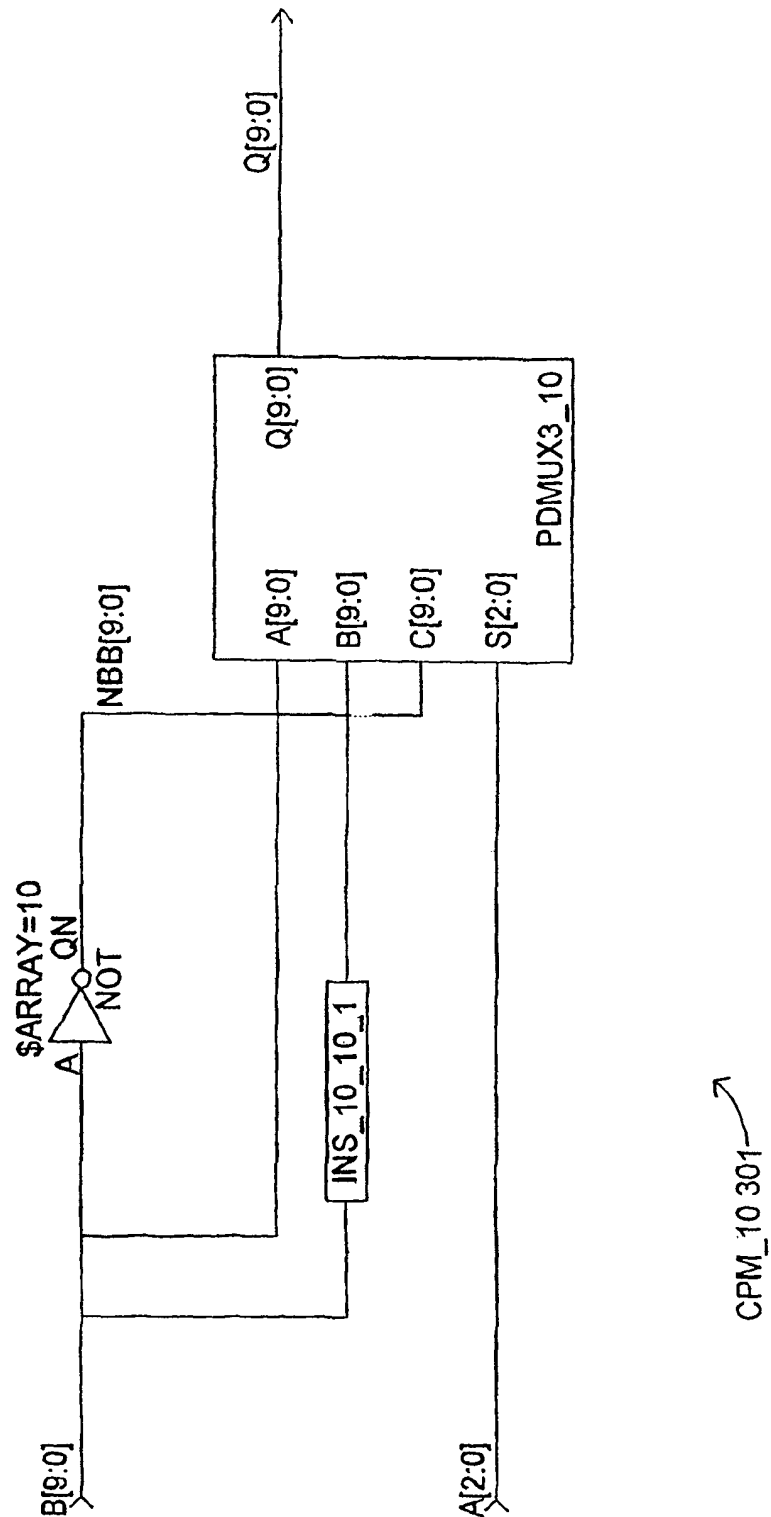


图 33

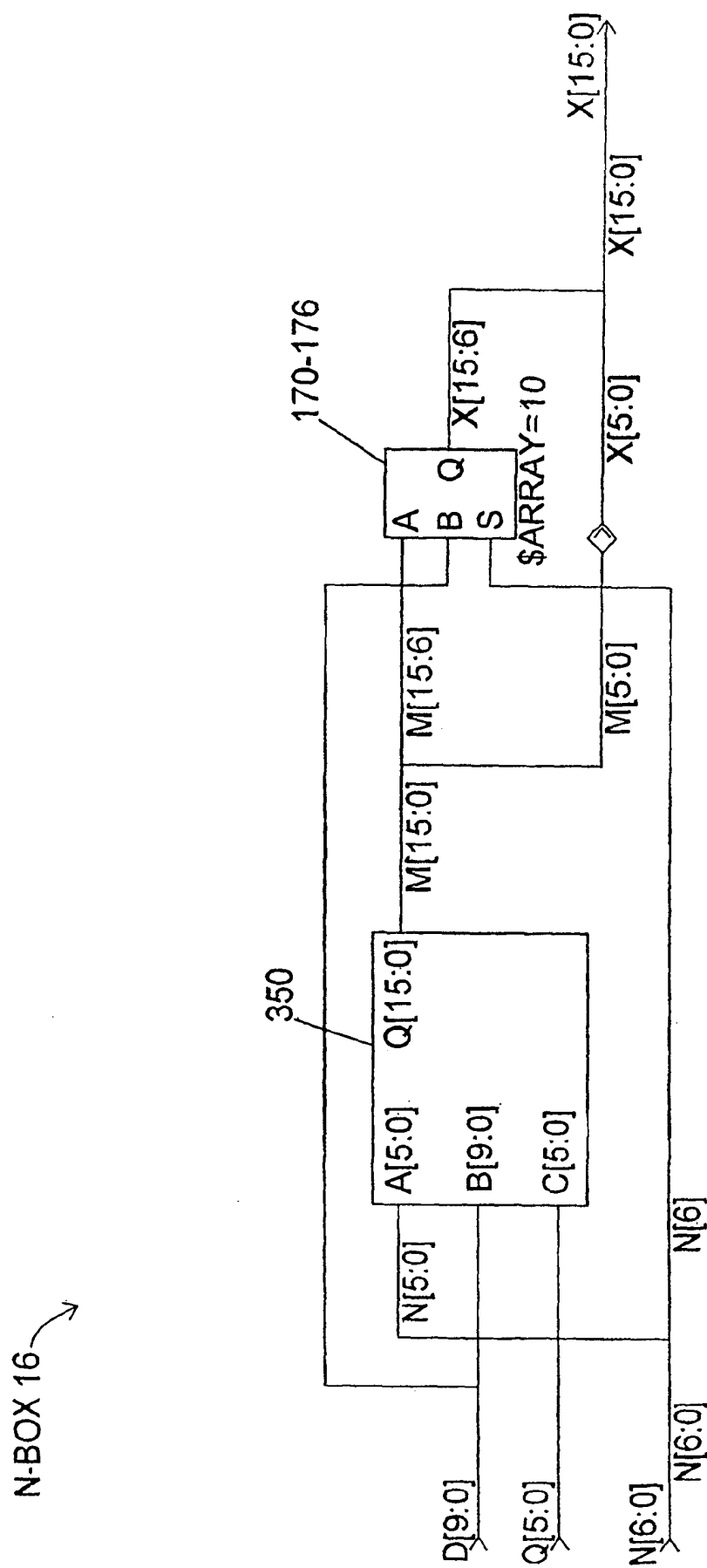
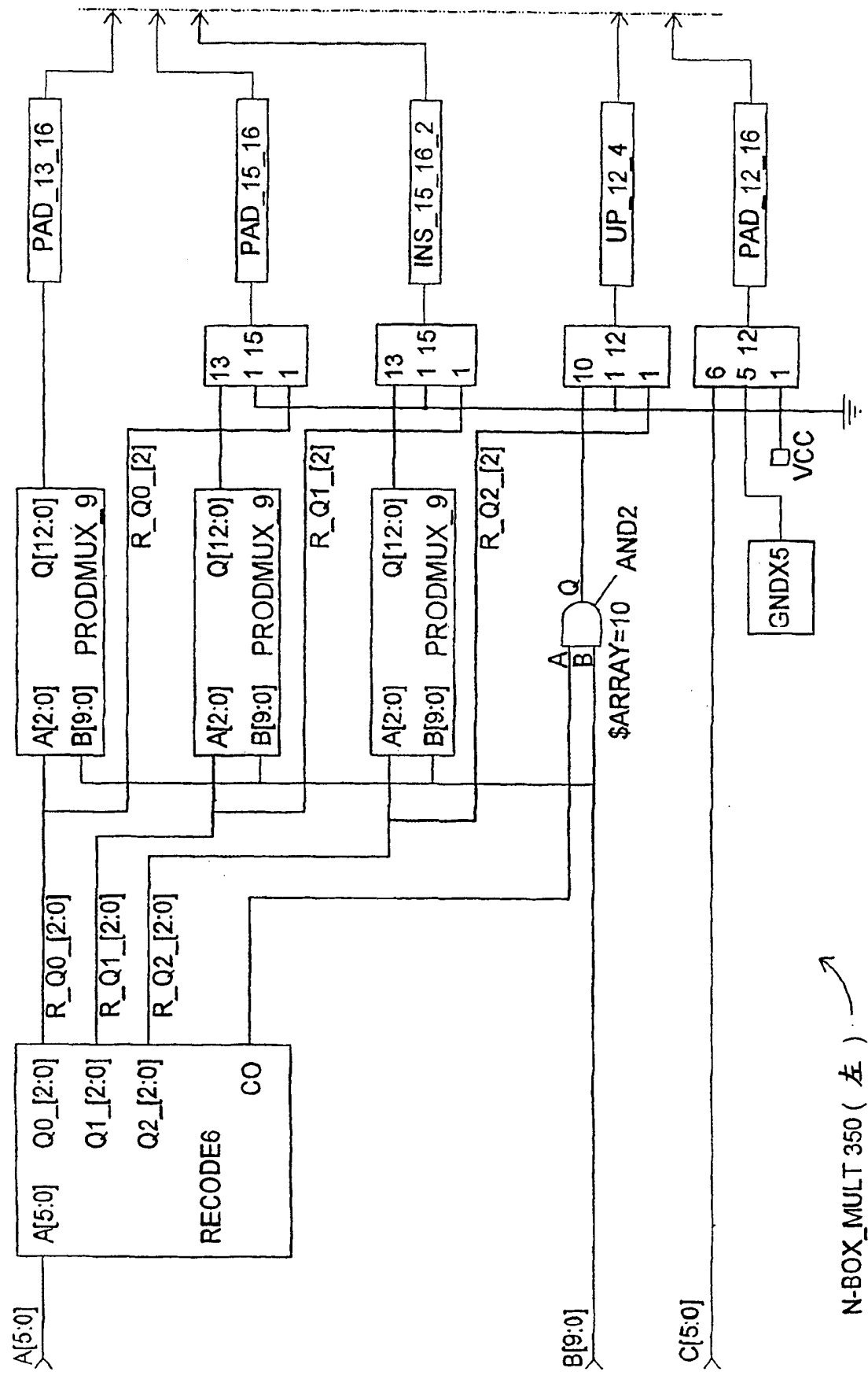


图 34



N-BOX_MULT 350 (左)

图 35A

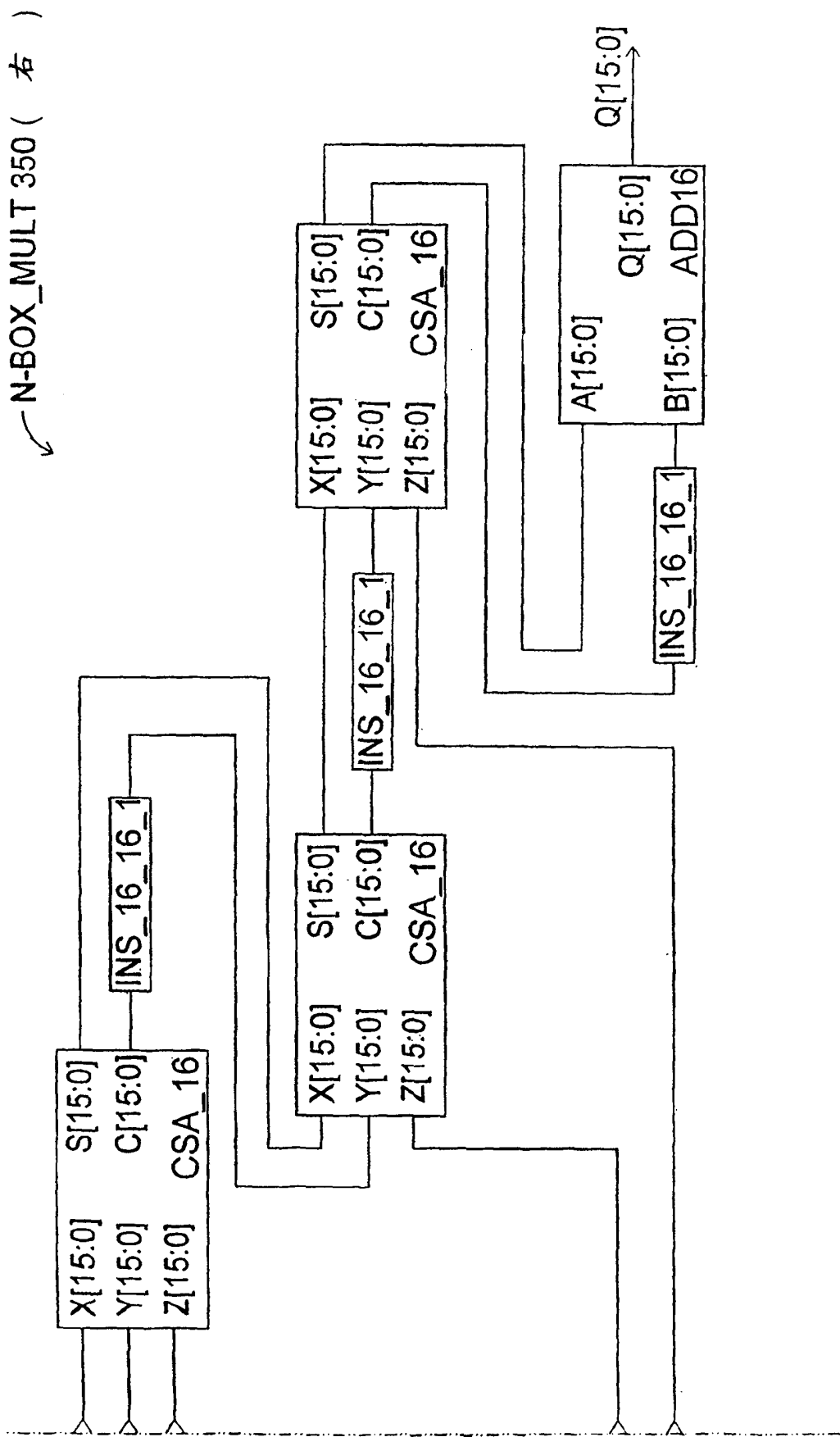


图 35B

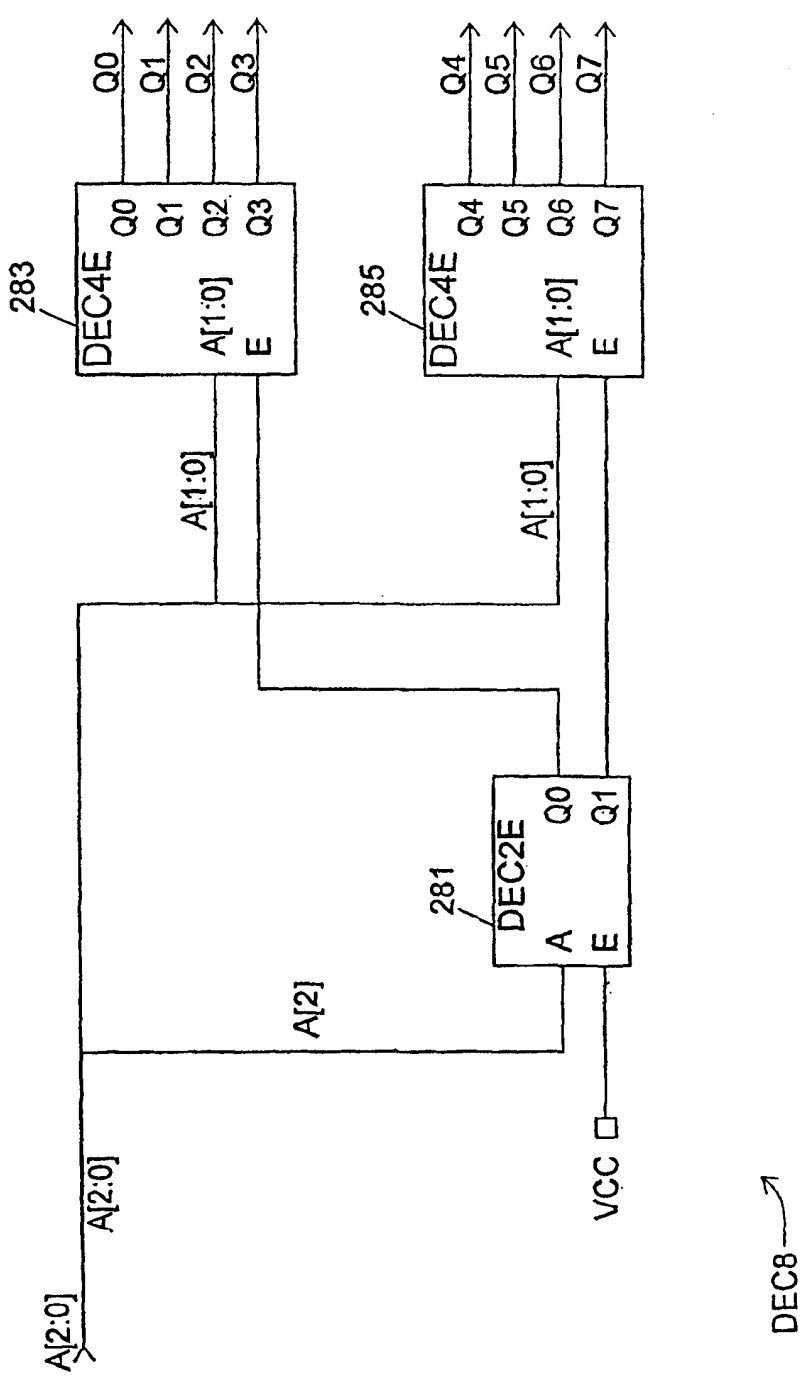


图 36

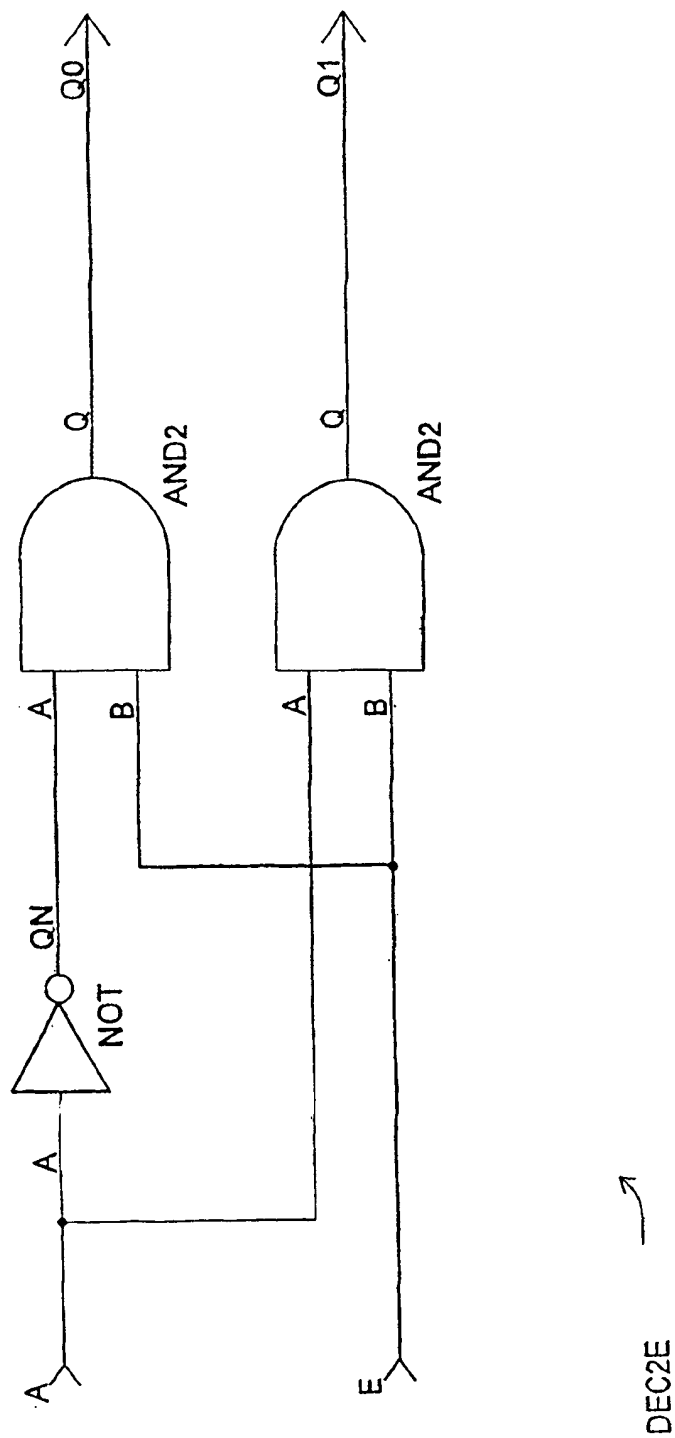


图 37