[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/455 (2006.01)
G06F 15/00 (2006.01)
G06F 15/76 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 02813381.1

[43] 公开日 2006 年 11 月 8 日

[11] 公开号 CN 1860441A

[22] 申请日 2002.5.2 [21] 申请号 02813381.1

[30] 优先权

[32] 2001. 5. 2 [33] US [31] 60/288,298

[86] 国际申请 PCT/US2002/011870 2002.5.2

[87] 国际公布 WO2002/103518 英 2002.12.27

[85] 进入国家阶段日期 2003.12.31

[71] 申请人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 J·林纳 G·赖 P·林
M·E·罗林斯 V·丁克维奇
C·B·格林伯格
C·E·菲利普斯 H·王
B·L·泰勒

[74] 专利代理机构 中国专利代理(香港)有限公司

代理人 邹光新 王 勇

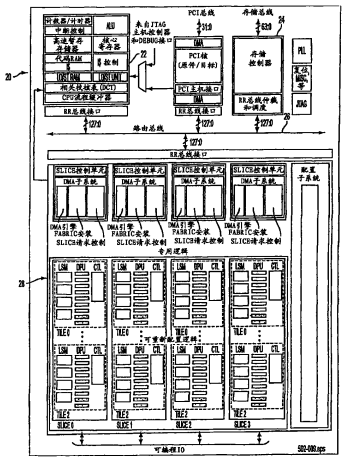
权利要求书 4 页 说明书 37 页 附图 25 页

[54] 发明名称

用于可重新配置环境中的高效高性能数据操作元件

[57] 摘要

一个可重新配置的芯片(20)，具有多个可重新配置的功能单元，这些单元包括一个移位单元、一个算术逻辑单元，及多路复用器。 数据通路与其它数据通路互相连接。 该互连线路传送长度为一个字的数据。 移位器使得能够对长度为一个字的数据进行调整，以便将其用于算术逻辑单元。 可重新配置功能单元是通过可重新配置功能单元指令控制的。可重新配置功能单元指令被存储在一个可重新配置功能单元指令存储器中，该存储器由芯片上的一个状态机寻址。



1. 一个可重新配置的芯片，包括：

多个可重新配置的功能单元，用于实现不同的功能，该可重新配置的功能单元包括多路复用器，至少一个移位单元和至少一个算术逻辑单元，该可重新配置的功能单元是通过一个可重新配置功能单元指令配置的，该指令控制多路复用器，移位单元和算术逻辑单元的配置；

互连单元，用于将一些可重新配置功能单元有选择地互相连接起来。

2. 权利要求 1 中所描述的可重新配置芯片，其中的可重新配置功能单元指令被划分为一些域，包括一个多路复用器域，一个移位单元域和一个算术逻辑单元域。

3. 权利要求 1 中所描述的可重新配置芯片，其中的可重新配置功能单元包含数据通路单元。

4. 权利要求 1 中所描述的可重新配置芯片，其中的互连单元用于传送字长数据。

5. 权利要求 4 中所描述的可重新配置芯片，其中的字长数据是 32 位或更长。

6. 权利要求 1 中所描述的可重新配置芯片，还包括一个指令存储器，存储可重新配置功能单元的多个指令。

7. 权利要求 1 中所描述的可重新配置芯片，其中的移位单元可以配置为一些不同的模式。

8. 权利要求 7 中所描述的可重新配置芯片，其中的可重新配置功能单元指令包含一个移位器单元域，控制移位器单元的模式。

9. 权利要求 1 中所描述的可重新配置芯片，其中至少一个多路复用器与一个延时单元输入和一个旁路该延时单元的输入相联系，以实现可变延时系统。

10. 权利要求 1 中所描述的可重新配置芯片，其中的可重新配置功能单元包含有寄存器，用于暂时存储该可重新配置功能单元中的值。

11. 一个可重新配置芯片，包括：

多个可重新配置功能单元，该可重新配置功能单元包括多路复用器，至少一个移位单元和至少一个算术逻辑单元，移位器单元的作用

是，使算法逻辑单元能够对可重新配置功能单元的字长输入数据中的不同的位进行操作；和

互连单元，用于将一些可重新配置功能单元有选择地互相连接起来，该互连单元用于传送字长数据。

12. 权利要求 11 中所描述的可重新配置芯片，其中的字长数据是 32 位或更长。

13. 权利要求 12 中所描述的可重新配置芯片，其中的字长数据是 32 位字长。

14. 权利要求 11 中所描述的可重新配置芯片，其中可重新配置的功能单元是通过可重新配置功能单元指令配置的。该指令控制多路复用器，移位单元和算术逻辑单元的配置。

15. 权利要求 11 中所描述的可重新配置芯片，其中还包括一个指令存储器，存储可重新配置功能单元的多个指令。

16. 权利要求 11 中所描述的可重新配置芯片，其中的移位单元可以配置为一些不同的模式。

17. 权利要求 11 中所描述的可重新配置芯片，其中一些多路复用器与一个延时单元输入和一个旁路该延时单元的输入相联系。

18. 一个可重新配置芯片，包括:

多个可重新配置功能单元，该可重新配置功能单元包括多路复用器，至少一个移位单元和至少一个算法逻辑单元，该可重新配置的功能单元是通过一个可重新配置功能单元指令配置的，该指令控制多路复用器，移位单元和算术逻辑单元的配置；和

一个指令存储器，存储可重新配置功能单元的多个指令。

19. 权利要求 18 中所描述的可重新配置芯片，其中一个指令存储器与每一个可重新配置功能单元相联系。

20. 权利要求 18 中所描述的可重新配置芯片，其中的指令存储器与一个状态机相联系，该状态机为指令存储器生成一个地址。

21. 权利要求 18 中所描述的可重新配置芯片，其中的可重新配置功能单元指令包含用于配置多路复用器的域，一个移位器单元控制域和一个算法逻辑单元控制域。

22. 权利要求 18 中所描述的可重新配置芯片，还包括一个互连单元，用于将一些可重新配置功能单元有选择地互相连接起来。

23. 权利要求 22 中所描述的可重新配置芯片，其中的互连单元用于传送字长数据。

24. 权利要求 18 中所描述的可重新配置芯片，其中的移位单元可以配置为一些模式。

25. 权利要求 24 中所描述的可重新配置芯片，其中的移位单元是由可重新配置单元指令的一个移位器单元域控制的。

26. 权利要求 18 中所描述的可重新配置芯片，其中至少一个多路复用器与一个延时单元输入和一个旁路该延时单元的输入相联系，以实现一个可变延时。

27. 一个可重新配置的芯片，包括:

多个可重新配置的功能单元，该可重新配置的功能单元包括多路复用器，至少一个移位单元和至少一个算法逻辑单元，移位器单元可被配置为一些模式;

互连单元，用于将可重新配置功能单元有选择地互相连接起来。

28. 权利要求 27 中所描述的可重新配置芯片，其中的移位器模式包含不同于逻辑和算术左移和右移的模式。

29. 权利要求 27 中所描述的可重新配置芯片，其中至少一种模式重新安排输入字的块。

30. 权利要求 27 中所描述的可重新配置芯片，其中的一种模式包含一个常量生成。

31. 权利要求 27 中所描述的可重新配置芯片，其中的一种模式包含将一组字节复制到另一组字节。

32. 权利要求 27 中所描述的可重新配置芯片，其中的一种模式包括将一些比特足于其他比特组相交换。

33. 权利要求 27 中所描述的可重新配置芯片，其中的可重新配置的功能单元是通过可重新配置功能单元指令配置的，可重新配置功能单元指令配置算法逻辑单元，移位单元和多路复用器。

34. 权利要求 33 中所描述的可重新配置芯片，其中的可重新配置功能单元指令包括一个控制移位单元的域，控制移位单元的模式。

35. 权利要求 27 中所描述的可重新配置芯片，其中的互连单元用于传送字长数据。

36. 权利要求 27 中所描述的可重新配置芯片，还包含指令存储

器，存储可重新配置功能单元的指令。

37. 权利要求 27 中所描述的可重新配置芯片，其中至少一个多路复用器与延时单元输入和一个旁路该延时单元的输入相联系，以实现可变延时系统。

38. 一个可重新配置芯片，包括:

多个可重新配置功能单元，该可重新配置功能单元包括多路复用器，至少一个移位单元和至少一个算法逻辑单元，其中至少一个多路复用器与延时单元输入和一个旁路该延时单元的输入相联系；和

互连单元，用于将可重新配置功能单元有选择地互相连接起来。

39. 权利要求 38 中所描述的可重新配置芯片，其中的可重新配置的功能单元是通过可重新配置功能单元指令配置的，该指令配置多路复用器，移位单元，算法逻辑单元。

40. 权利要求 39 中所描述的可重新配置芯片，其中的可重新配置功能单元指令包括一些不同的域，用于控制多路复用器，移位单元和算法逻辑单元的配置。

41. 权利要求 39 中所描述的可重新配置芯片，其中可重新配置功能单元的一条指令的一个域表明提炼模式。

42. 权利要求 38 中所描述的可重新配置芯片，其中的互连单元用于传送字长数据。

43. 权利要求 38 中所描述的可重新配置芯片，还包含指令存储器，存储可重新配置功能单元的多条指令。

44. 权利要求 38 中所描述的可重新配置芯片，其中的可重新配置功能单元包括一个移位单元，该单元可被配置为一些不同的模式。

用于可重新配置环境中的高效高性能数据操作元件

相关应用/优先级

本专利要求享受 2001 年 5 月 2 日申请的专利 No.60/288,298 的优先权。

技术领域

本发明有关可重新配置逻辑芯片，特别是有关用在可重新配置计算中的可重新配置逻辑芯片。

背景技术

域可编程门阵列（FPGA）是可编程芯片，可以实现不同的配置。一般，利用设计工具来生成一个设计，并为一个具体的设计来配置一个 FPGA。尽管可以更改设计，但一般来说，FPGA 使用的还是单一的配置，其原因是，改变一个配置所需要的时间远比芯片的工作时间长。

最近，设计出了一种可重新配置芯片，它可以将一个算法的某些部分快速切换到一个可重新配置的芯片上。设计出这些可配置芯片的目的是，利用该芯片的可重新配置元件，为一个算法中某些部分的实现提供资源。

我们希望在一个可重新配置芯片中使用一个数据操作元件或可重新配置功能单元，以便在一个可重新配置的芯片上实现一个更有效的算法。

发明内容

本发明有关一个可重新配置芯片，其中包含用于实现不同功能的多个可重新配置功能单元（例如一个数据通路单元）。可重新配置功能单元最好包含多路复用器，至少一个移位单元和至少一个算法逻辑单元（ALU）。可重新配置功能单元是通过可重新配置功能单元指令来配置的。这些指令控制多路复用器和移位单元及 ALU 的配置。可重新配置芯片还包括用于将各可重新配置功能单元连接在一起的互连结构。以这种方式，数据可以在各可重新配置功能单元之间流动。

可重新配置功能单元指令最好包含一些域，分别用于多路复用器、移位单元和 ALU。这些域以所要求的方式配置一个可重新配置功能单元中的相应单元。

在一个推荐实例中，每一个可重新配置功能单元都有一个相关的指令存储器。该指令存储器中为该可重新配置功能单元存储一组指令。在一个推荐实例中，状态机寻址该指令存储器，以便确定要被装入该可重新配置功能单元的下一条指令。在一个推荐实例中，可重新配置功能单元向状态机提供反馈，表明一项功能何时结束，下一项功能应在何时装入该可重新配置功能单元。

在一个实例中，采用互连单元来有选择地将一些可重新配置功能单元连接起来，以传送字长数据。所传输数据最好具有固定字长（32位或更多）。固定字长传输器能简化互连系统，其代价是损失了数据传输的灵活性。可重新配置功能单元中的移位单元使得算法逻辑单元能够对字长输入数据中不同的位进行操作，以补偿互连单元的固定结构所带来的不便。因此，若所需数据位于一个字的某一位置，移位器可以将该比特位置移动到能由算法逻辑单元操作的合适位置。

本发明的另一个实例包括使用一个多路复用器，该多路复用器具有一个延时单元输入和一个旁路该延时单元的输入。以这种方式，可重新配置功能单元可以实现一个可变的延时，以提高系统的灵活性。

附图说明

图 1 是本发明的一个实例中，一个可重新配置芯片的总图；

图 2 是本发明的一个实例中，一个可重新配置功能单元的简图；

图 3 是本发明的一个实例中，一个可重新配置功能单元的图形；

图 4 是可由本发明实例使用的一个乘法器单元的图形；

图 5 是图 1 所示可重新配置功能单元的一个 slice，例证了数据通路单元之间的互连；

图 6 例证了数据通路单元与水平和垂直总线之间的连接；

图 7 例证了一个 tile 中的数据通路单元与另一个 tile 中的数据通路单元之间的互连；

图 8 举例说明了本发明的一个实例中，数据通路单元与本地系统存储器之间的互连；

图 9 举例说明了一个状态机和功能块配置存储器，该功能块配置存储器为功能块数据单元生成配置信息指令；

图 10A 举例说明了本发明的一个状态机、配置状态存储器和数据通路单元的互相连接，显示了数据通路单元的指令和指令域；

图 10B 举例说明了一个数据通路单元，该单元为指令的至少一部分使用了译码器；

图 11 举例说明了作为本发明的一个实例，数据通路单元处的控制系统配置存储器；

图 12 举例说明了本发明的一个实例中使用的一个互连逻辑单元；

图 13A 和 13B 是图表，举例说明了 ALU 的部分指令；

图 14 举例说明了本发明一个实例中系统的标志；

图 15 举例说明了移位器的移位模式；

图 16 是一个移位器实例的说明；

图 17 举例说明了图 16 中移位器的操作；

图 18 中显示了本发明一个实例中的逻辑系统，该系统使用多个主锁存器；

图 19 举例说明了本发明一个实例的背景和前景平面锁存器；

图 20 是本发明的一个实例中，一个数据通路的可重新配置功能单元的实例；

图 21 是图 20 的系统中输入多路复用器的实例；

图 22 是本发明的一个实例中，移位器的移位模式图；

图 23 举例说明了本发明一个实例中，移位器的一些移位模式；

图 24 举例说明了本发明的一个实例中，一个 turbo 搜索表的实现。

具体实施方式

图 1 举例说明了一个可重新配置芯片 20。该可重新配置芯片 20 包括一个中央处理单元(CPU)22，推荐是一个缩减指令集(RISC)CPU。利用存储控制器 24 从外部存储器（未示出）传送数据。被称为路由总线的总线 26 用于从存储控制器向可重新配置 fabric28 传输数据。可重新配置 fabric(构件)28 被划分为一些 slice(单片)。每个 slice 又被划分为一些 tile(子片)。每个 tile 都包括一个数据通路单元(可重新配置功能单元)、控制单元和本地系统存储单元。本地系统存储单元按以下将要描述的方式与数据通路单元互相作用。在一个推荐实例中，每个 tile 还有一些多路复用器单元。

图 2 举例说明了本发明一个实例中的一个可重新配置功能单元。该可重新配置功能单元包括输入多路复用器 30 和 32。如以下将要介

绍的，输入多路复用器使得数据通路单元能够从不同的位置（包括邻近的数据通路单元以及数据总线）接收输入。输入多路复用器选定的输出被送给寄存器 36 和 38。另外，多路复用器 32 的输出送给移位单元 34。如以下将介绍的，移位单元 34 使得 ALU 40 能选择对不同的位进行操作。由于为简化互连系统，数据通路单元之间的互连使用的是固定字长连接，因此，在数据通路单元中使用移位单元，以便能够对包含在一个字中间的位进行访问。

如以下将介绍的，移位单元 34 最好具有一些模式，不仅仅是实现逻辑和算术左移和右移。这些不同的模式使得系统能够以一种更有效的方式进行操作。以下介绍的算法逻辑单元 40，最好使用数据通路单元指令的一个域来实现一个功能。ALU 40 的输出最好送给一个输出寄存器 42。该输出还可以被送给一个可选的比特移位器 44，以便生成一个经过移位的值。

在一个实例中，还使用了线路 46 上的一个旁路 ALU 反馈输出。这就使得在输出寄存器 42 控制从数据通路单元送出哪些输出的同时，数据通路单元的一些部分还能够工作。这在输出寄存器 42 用于寻址一个本地系统存储单元时很有用。

比特移位器 44 被用于实现线性反馈移位寄存器，见 Peter Lam，Attorney Docket 提出的专利应用 No.032001-060 “Modifications to Reconfigurable Functional Unit in a Reconfiguable Chip to Perform Linear Feedback Shift Register Function”。

应指出，多路复用器，移位单元 34 和 ALU 40 最好是由数据通路单元的一条指令控制的。该指令被划分为一些不同的域，包括用于多路复用器的多路复用器指令域，用于移位器 34 的移位器单元域和用于 ALU 40 的 ALU 指令域。在一个实例中，将一个译码器用于该指令的至少一部分。

图 3 详细描述了本发明的一个实例。输入多路复用器 50 和 52 从邻近的单元接收输入数据。在一个例子中，来自 16 个单元（包括数据通路单元和乘法器单元）的数据字被用作输入。该例中使用了全局垂直和水平互连。在一个实例中，所连接有：线性反馈移位寄存器的反馈信号，一个逻辑零常量输入和一个本地系统存储单元的输入。另一个输入是前一个数据通路单元的进位输入，该输入被直接提供给 ALU

54。多路复用器 50 与移位器 56 相连，移位器 56 包括一些不同的操作模式。移位器 56 又与另一个多路复用器 58 相连，因此，多路复用器 50 的输出既可以避开也可以使用移位器单元 56。移位器单元 56 还能以一些模式使用来自输入多路复用器 52 的 A 输入。多路复用器 58 和 52 的输出可分别送给寄存器 60 和 62。寄存器 60 和 62 还可从芯片外装入。逻辑 64 和 66 使该寄存器值能够作为系统的掩模寄存器。多路复用器 68 和 70 为 ALU 54 选择输入。ALU 的输出被送到一些不同的可能通路。应指出，多路复用器 72 送出的数据通路输出可以是来自输出寄存器 74 的值，也可以是来自多路复用器 76 的值（可以是 ALU 值或线路 78 上的本地系统存储器数据）。来自 ALU 的标志值被送给多路复用器 80 和 82，由多路复用器来选择所要求的标志值。该标志值被存储在寄存器 88 和 90 中，寄存器 88 和 90 的值被送给多路复用器 92 和 94，或者，使用多路复用器 80 和 82 所选定的值。CONF 值是指令中的一个域，表明选定的是哪一个标记。

在一个实例中，可通过图 18 中所显示的多个主辅锁存器来实现寄存器 60，62 和 74，以使得能够将背景配置数据装入寄存器中。在一个实例中，可通过可重新配置功能单元指令来控制这些寄存器的操作。

图 4 是一个乘法器单元的图形。乘法器单元与图 3 所示的可重新配置功能单元有些类似。不过，乘法器单元具有一个专用乘法器，而不是 ALU。

如图 5 所示，在一个实例中，对于一个 tile 中的每七个数据通路单元或可重新配置功能单元，都有两个乘法器单元。

图 6 举例说明了怎样将相邻数据通路单元和乘法器连接到数据通路单元的输入中。参看图 5，作为数据通路单元 100 的输入，数据通路单元 100 可以接收来自上面八个前面的数据通路单元（和乘法器）和下面七个后面的数据通路单元（和乘法器）的输出。数据通路单元 100 的输出还反馈给自己。利用系统的输入多路复用器，可以为 A 或 B 输入选择这些单元中任意单元的输出。

图 6 举例说明了怎样将一个 tile 的可重新配置功能单元（数据通路单元）连接到水平和垂直连接线上。借助于多路复用器，数据通路单元的输出和输入可被互连到垂直路由线路和水平路由线路上。

图 7 例证了怎样利用垂直互连线路将一个 tile 中的一个数据通路单元与另一个 tile 中的一个数据通路单元互连。应指出，在本发明系统中，推荐使用基于字的互连。在一个实例中，互连线路允许 32 位字宽数据的连接。一旦一个数据通路单元从互连系统接收到数据，其中的移位器单元就可以对该数据进行对齐处理。由于系统是以 32 位字发送数据的，因此就降低并简化了互连系统的复杂度，不过稍稍降低了互连的灵活性。

图 8 例证了数据通路单元和本地系统存储器之间连接。在一个推荐环境中，使用交替的数据通路单元来实现本地系统存储器的读和写。例如，数据通路单元 102 向本地系统存储器 104 提供读地址并从本地系统存储器 104 接收读出数据。数据通路单元 106 向本地系统存储器 104 提供写地址并写入数据。应指出，通过使用诸如通过门 106，108，110 和 102 之类的通过门，数据通路单元 102 和 106 可以连接到其它的本地系统存储器上，例如本地系统存储器 114，数据通路单元 116 和 118 可以连接到本地系统存储器 104 上。在另一个实例中，数据通路单元可以对一个本地系统存储器进行读写。数据通路单元的一个作用是向本地系统存储器提供一个地址，以便从本地系统存储器获得数据，数据通路单元可被放置在水平和垂直互连总线上。图 8 中所显示的连接是直接连接，直接从本地系统存储器读出并写入数据。在一个推荐实例中，本地系统存储器利用存储控制系统进行全局读写。这一通用存储控制系统用于配置系统并获得由数据通路单元操作的数据。应指出，如上面所描述的，在一个推荐实例中，数据通路单元包含有这样一些指令，这些指令允许数据通路单元在执行一些其它功能的同时，向本地系统存储器提供地址和数据。

图 9 描述了可重新配置功能单元 130 的一个控制 fabric 单元 132。在该实例中，控制 fabric 单元 132 为可重新配置功能单元 130 生成一个控制或指令行。在该实例中，推荐控制 fabric 单元 132 包含一个状态机单元 134 和一个功能块配置存储单元 136。状态机 134 向指令存储器 136 中生成地址。状态机 134 的一种实现方式是使用一个可重新配置的可编程积-和单元 136。

图 10A 举例说明了一个系统，该系统包括状态机配置单元 136，配置状态存储器 138'和数据通路单元 130'。应指出，来自配置状态存

储器 138'的配置可被认为是数据通路单元 130'的一个指令。该指令最好包括一些域，例如一个 ALU 配置域，移位寄存器配置域和一个多路复用器配置域。在一个实例中，来自数据通路单元 130'的一些标志被送给状态机 136'，以便在数据通路单元完成对一批数据的操作之后，切换数据通路单元的配置。配置状态存储器 138'也可以是从来自外部存储器或来自处理器的一个外部配置装定的。

图 10B 举例说明了一个数据通路单元，该单元使用一个译码器来对指令的至少一部分进行译码。

图 11 显示了一个控制系统，该系统包括不同配置状态存储器的状态机。如以上所述，数据通路单元标志被送给控制系统。

图 12 举例说明了一个算法逻辑单元。该单元包括一个算术单元 142，一个并行逻辑单元 140 和一个标志单元 144。图中还显示了一个进位选择单元 146。首先发送来自指令的 ALU 指令域，以选择 ALU 的操作。算术单元 142 使用一个进位输入。在一个推荐实例中，该进位值要么是前一个数据通路单元的进位，要么是控制信号，或一个作为指令一部分的进位值。

图 13A 和 13B 举例说明了一些 Opcode 的列表，这些代码用在本发明可重新配置功能单元的一个 ALU 的实例中。这些 Opcode 的详细介绍参见附录 I，在此引用以供参考。

图 14 是本发明中标志系统的图形。标志单元位于数据通路单元中，用于生成标志，该标志被送往控制单元及下一个数据通路单元。所用标志的选择是由可重新配置功能指令的一个域来控制的。以下给出了对一些标志的描述。

每个周期驱动一次 ROXR。它是由 conf==1 这个条件来选择的。

该操作是:

当 opcode[7]==0 时，　flag[1]==^(B[31:0])
　　flag[0]==^(B[15:0])

当 opcode[7]==1 时，　flag[1]==^(B[31:16])
　　flag[0]==^(B[15:0])

缩写:

CO: 加/减操作的进位输出

OV: 加/减操作溢出

EQ: 相等（A==B）

GT: 大于

LT: 小于

SN: 符号（结果的符号位）

前面的标志

Cin: 前一行的进位

Ctrl: 来自控制的进位

Max: 0x7fff[fff]（对于 16/32 位）

Min: 0x8000[0000] （对于 16/32 位）

图 15 举例说明了本发明一个实例中移位单元的移位模式和一些模式的操作。由于移位单元具有一些不同的模式，因此可以提高本发明系统的灵活性。

图 16 和 17 举例说明了一个使用多行多路复用器的移位单元的一种实现方式。附加逻辑也被用于生成一个专用的输出。图 17 举例说明了一些移位寄存器的操作。

数据通路单元中使用的这种移位器执行的不仅仅是左/右移位操作。该移位器包含一个多路复用器阵列，它们由多路复用器选择信号控制。在一个 4x6 路多路复用器阵列的移位器实例中，一个 32 位的操作数被划分为以 8 个信号为一组的四个组，它们被连接到第一行的四个多路复用器上。除了最后一行之外，前一行中多路复用器的输出被连接到下一行多路复用器的输入。阵列中的每一个多路复用器都是独立控制的。控制信号决定了信号怎样通过该阵列，并由此决定了对该操作数所执行的操作类型。在一个实例中，所做操作的例子有：32 位逻辑右/左移位，32 位算术右/左移位，低 16 位符号扩展到 32 位，常量生成，低 16 位复制到高 16 位，高 16 位复制到第 16 位，高 16 位和低 16 位交换，16 位算术右移，和字节交换。

[0062] 图 18 举例说明了本发明系统的一个实例中所用的一个多重主锁存器系统。在该例中，使用了两个主锁存器：一个用于系统的背景配置。另一个则从数据通路单元的流程或处理器中接收数据。锁存器 150 的输入是通过多路复用器 152 提供的。锁存器 154 与配置总线相连，用于从背景配置接收数据。多路复用器 156 可被用于为辅助锁存器 158 选择输入。在系统中使用一个背景配置存储器，可以加

快本发明系统的操作速度。

图 18 的存储单元具有多个主锁存器，这些主锁存器通过一个多路复用器共享一个单独的辅助锁存器，从而提供一个多功能存储介质。另外，通过共享一个辅助锁存器，可以明显地节省空间（大约 25%）。在使用大量存储单元的系统中，这种空间上的节省尤为明显。这种存储单元设计是基于这样一个事实，即，将配置位被装入存储单元这一事件很少发生。因此，按照本发明，不必使每一个主锁存器都有一个独立的与一个配置比特流信号相连接的辅助锁存器，而是使与配置比特流信号相连接的主锁存器与另一个主锁存器共享其辅助锁存器。因此，两个或多个主锁存器共享一个单独的辅助锁存器。在这些主锁存器与该单独的辅助锁存器之间连接一个多路复用器，用于选择将哪一个主锁存器与该辅助锁存器连接。

在一个实例中，一个主锁存器的输入与一个频繁请求存储单元功能的信号连接，另一个主锁存器的输入与一个不频繁请求存储单元功能的信号连接。第一个主锁存器与数据通路信号连接，第二个主锁存器与配置比特信号连接。在数据通路信号被送给辅助锁存器时，存储单元将数据通路流程划分为一些阶段。当配置比特流信号被送给辅助锁存器时，存储单元存储该配置比特。在另一个实例中，一个主锁存器与数据通路信号相连，多个主锁存器与一个配置比特信号相连，所有的主锁存器输出都与多路复用器相连，多路复用器用于选择并将来自主锁存器的信号之一送给共享的辅锁存器。

图 18 中：

. 一旦出现 'RESET'或 'INIT' 信号，则主锁存器复位

. 只有在出现 'RESET'信号时，辅助锁存器才复位

. 只要配置是启动的， A 路的多路复用器就选择配置通路（还受所选的具体 slice 限制）

. 在 arc 写入时， B 路的多路复用器选择 arc 总线（还受对相应 arc 地址(arc_address)译码结果的限制。请参阅地址变换的 ARC ext spec）。

. 只有在时钟为低时，主锁存器才是透明的

. 只有在时钟为高时，辅助锁存器才是透明的

. 在 latpipe 0 使能或出现了对该寄存器的 arc 写入时，主锁存

器 0 是透明的

. 在激活了配置装入并且相应的配置地址已被译码时，主锁存器 1 是透明的

. 在下列情况下，辅助锁存器是透明的:

1. 启动了对该 slice 的配置，或

2. 对该寄存器进行 arc 写入，或

3. 来自控制端的 latpipe 信号为高电平

. 该设置是基于这样一个假设条件，即: 配置和 arc 写入不是同时发生的。若它们同时发生，则配置的优先级更高。

本发明的另一个实例所关注的是本发明的可变延时单元。可变延时单元由一个多路复用器组成，它接收送给一个寄存器的第一单元和旁路该寄存器的第二输入。以这种方式，可以实现一个可变的延时。在图 3 的可重新配置功能单元中，连接到多路复用器 68 上的寄存器 60，连接到多路复用器 70 上的寄存器 62，连接到多路复用器 92 上的寄存器 88，连接到多路复用器 94 上的寄存器 90，和连接到多路复用器 72 上的寄存器 74，可以实现这样一个可变延时。一个多路复用器可以选择一个被延时的信号或一个旁路信号；该延时信号经过一个延时单元，就像一个触发器。

这种灵活的适应性延时单元包括一个存储设备（例如触发器，锁存器），该存储设备的输入端与一个输入信号相连，其输出端连接一个多路复用器的第一输入端。该多路复用器的其它输入端与输入信号相连。其结果是，多路复用器的第一输入端与输入信号相连，其第二输入端与经过存储设备被延时了的输入信号相连。选择信号用于对延时信号和非延时信号进行选择。

图 19 显示了一个背景前景平面配置的另一个实例。

本发明中引用了以前的专利应用，包括“A HIGH PERFORMANCE DATA PATH UNIT FOR BEHAVIORAL DATA TRANSMISSION AND RECEPTION”，发明者是 Hsinshih Wang，序列号为 09/307，072，于 1999 年 5 月 7 日提出申请（代理案号为 032001-014），“CONTROL FABRIC FOR ENABLING DATA PATH FLOW”，发明者是 Shaila Hanrahan 等，序列号为 09/401，194，于 1999 年 9 月 23 日提出申请（代理案号为 032001-016），以及“CONFIGURATION STATE MEMORY FOR FOUNCTIONAL BLOCKS ON A

RECONFIGURABLE CHIP”，发明者是 Shaila Hanrahan 和 Christopher E.Philips，序列号为 09/401，312，于 1999 年 9 月 23 日提出申请（代理案号为 032001-035）。

Vermont 实例。

图 20 举例说明了可重新配置功能单元或数据通路单元的一个最终实例。在该实例中，在 B 输入通路上，在移位器之前加入了一个寄存器和多路复用器。另外，对输入多路复用器稍稍进行了改动。在图 21 中显示了输入多路复用器。

图 22 举例说明了图 19 所显示的新实例中，移位器模式的列表。

图 23 举例说明了图 22 所显示的新模式的实现方式。

图 24 举例说明了本发明系统中使用的 turbo 查询表。在对以对数格式存储的数据进行加法运算时，该 turbo 查询表很有用。该表可用于很多通信系统中。在以前的实例中，为了对以对数格式存储的数据进行乘法运算，必须首先对数据进行指数扩展，将其转化为普通格式。随后，将经过指数扩展的数据相加，再将结果信息转换回对数格式。在该推荐实例中，该 turbo 查询表用于生成加入一个校正因子之后的估算值。该估算过程中，将 A 和 B 中的最大值作为 A 与 B 之和的第一个估算值。A 与 B 之差的绝对值作为查询表的输入，提供一个校正因子，加到最大值 A 或 B 中。通过向最大值 A 或 B 加入该校正因子，可以产生一个相对精确的估算值。应指出的事，查询表不必与 A 的输入比特数量相同。在一个推荐实例中，只需要几位的精确度。若 A 减 B 的幅值很大，则组合出来的值与 A 和 B 中的最大值并无明显差异。例如，1,000,000 与 0.1 相加，结果近似与 1,000,000。1,000,000 与 1,000,000 相加等于最大值的两倍。

附录 II 和 III 进一步例证了可重构配置单元的 Vermont 实例。

对于了解本项技术的人员来说，可以在不脱离本发明实质和特性的条件下，以其它具体的形式来实现本发明。此处所列出的实例只起举例说明的作用，并不限定本发明。本发明的范围是通过附加的权利要求规定的，并不是由前面的描述所定义的。凡是在权利要求的含义和范围之内的所有变动，都涵盖在本发明范围之内。

附录 I

1.9 Opcode 详述

名称	ADD
伪代码	result=A+B
描述	A 加 B
受影响标志	CO 该操作的进位 OV 若该操作溢出 EQ 若 a==b SN result[31]

名称	ADD16
伪代码	结果={(AH+BH),(AL+BL)}
描述	并行 A 加 B
受影响标志	CO,OV ,EQ,SN 类似于加法运算，只不过标记 flag[0]有效

名称	ADDC
伪代码	结果=A+B+Cin
描述	32 位操作的 A 加 B 加进位 Cin 是低 16 位的进位
受影响标志	CO,OV ,EQ,SN 类似于加法运算

名称	ADDCNT
伪代码	结果=A+B+Ctrl
描述	32 位操作的 A 加 B 加控制进位
受影响标志	CO,OV ,EQ,SN 类似于加法运算

名称	SUB
伪代码	结果=A-B
描述	A 减 B
受影响标志	CO: 操作（A+～B+1）的进位 OV: 若操作溢出 GT: 若 A>B

LT: 若 A<B
EQ: 若 A==B
SN: 结果的符号

名称	SUB16
伪代码	结果={(AH-BH),(AL-BL)}
描述	并行 A 减 B
受影响标志	CO,OV ,GT,LT,EQ,SN 类似于减法运算

名称	SUBC
伪代码	结果=A+~B+Cin
描述	带进位的 A 减 B，32 位操作
受影响标志	CO,OV ,GT,LT,EQ,SN 类似于减法运算

名称	SUBCNT
伪代码	结果=A+~B+Ctrl
描述	带控制进位的 A 减 B
受影响标志	CO,OV ,GT,LT,EQ,SN 类似于减法运算

名称	SADD
伪代码	if（溢出） 结果=最大值 else if（下溢） 结果=最小值 else 结果=A+B
描述	带饱和的 A+B
受影响标志	CO: A+B 的进位输出 OV: A+B 溢出 EQ: A==B SN: 结果的符号

名称 SADD16

伪代码 并行 16 位 SADD 运算

描述 带饱和的 A 加 B

受影响标志 CO, OV , EQ, SN 类似于 SADD, flag[0]有效

名称 SADDCNT

伪代码
```
if（溢出）
 结果=最大值
 else if（下溢）
  结果=最小值
else
  结果=A+B+Ctrl
```

描述 带控制进位及饱和的 A+B

受影响标志 CO, OV, EQ, SN 类似于 SADD

名称 SSUB

伪代码
```
if（溢出）
 结果=最大值
 else if（下溢）
  结果=最小值
else
  结果=A-B
```

描述 带饱和的 A-B

受影响标志
```
CO: A+～B+1 的进位
OV:  A-B 溢出
GT: A>B
LT: A<B
EQ:  A==B
SN:  结果的符号
```

名称 SSUB16

伪代码 并行 16 位 SSUB

描述 带饱和的 A-B
受影响标志 CO,OV,GT,LT,EQ,SN 类似于 SSUB，flag[0]有效

名称 SSUBCNT
伪代码 if（溢出）
结果=最大值
else if（下溢）
结果=最小值
else
结果=A+～B+Ctrl
描述 带控制进位和饱和的 A-B
受影响标志 CO,OV,GT,LT,EQ,SN 类似于 SSUB

名称 INC
伪代码 结果=B+1
描述 B 递增
受影响标志 CO: B+1 的进位
OV: B+1 的溢出位
SN: B+1 的符号

名称 DEC
伪代码 结果=B-1
描述 B 递减
受影响标志 CO: B+0xffffffff 的进位
OV: B-1 的溢出位
SN: B-1 的符号

名称 NEG
伪代码 结果=～B+1
描述 对 B 求反
受影响标志 SN: ～B+1 的符号

名称	ABS

伪代码	if(B 为负)
	结果=~B+1
	else
	结果=B

描述	B 的绝对值
受影响标志	

名称	ABS16
伪代码	并行 16 位 ABS
描述	32 位操作中，取 B 的绝对值
受影响标志	

名称	CSUB

伪代码	if(A-B>=0)
	结果=A-B
	else
	结果=A

描述	条件减
受影响标志	CO: A+~B+1 的进位 OV: A-B 溢出 GT: A>B LT: A<B EQ: A==B SN: 结果的符号

名称	AND
伪代码	结果=A&B
描述	逐位与
受影响标志	EQ: A==B SN: 结果的 bit[31]

名称	OR
伪代码	结果=A|B
描述	逐位或
受影响标志	EQ，SN 与 AND 操作相同

名称	NAND
伪代码	结果=~(A&B)
描述	逐位与再求反
受影响标志	EQ，SN 与 AND 操作相同

名称	NOR
伪代码	结果=~(A|B)
描述	逐位或再求反
受影响标志	EQ，SN 与 AND 操作相同

名称	XOR
伪代码	结果=A^B
描述	逐位异或
受影响标志	EQ，SN 与 AND 操作相同

名称	XNOR
伪代码	结果=~(A^B)
描述	逐位异或求反
受影响标志	EQ，SN 与 AND 操作相同

名称	PASSA
伪代码	结果=A
描述	传递 A
受影响标志	EQ，SN 与 AND 操作相同

名称	PASSB
伪代码	结果=B

描述 传递 B
受影响标志 EQ，SN 与 AND 操作相同

名称 NOTA
伪代码 结果=~A
描述 使 A 反转
受影响标志 EQ，SN 与 AND 操作相同

名称 NOTB
伪代码 结果=~B
描述 使 B 反转
受影响标志 EQ，SN 与 AND 操作相同

名称 MIN
伪代码
```
if(A<B)
   结果=A
 else
   结果=B
```
描述 返回 A 和 B 中的较小值
受影响标志
GT: A>B
LT: A<B
EQ: A==B
SN: 结果的 bit[31]
CO: A+~B+1 的进位
OV: A-B 溢出

名称 MIN16
伪代码 并行 16 位 MIN
描述 32 位操作，返回 A 和 B 中的较小值
受影响标志 与 MIN 操作相同，flag[0]也有效

名称 Max

伪代码
```
if(A>B)
  结果=A
else
  结果=B
```
描述 返回 A 和 B 中的较大值

受影响标志
GT: A>B
LT: A<B
EQ: A==B
SN: 结果的 bit[31]
CO: A+~B+1 的进位
OV: A-B 溢出

名称 MAX16

伪代码 并行 16 位 MAX

描述 32 位操作，返回 A 和 B 中的较大值

受影响标志 与 MIN 操作相同，flag[0]也有效

名称 PENC

伪代码
```
result=0
for( i=31; i>=0; i++)
{ if(B(i)==1)
  { result=i+1;
    break;
  }
}
```
描述 监测起始位 1

受影响标志 无

名称 MUXBBA

伪代码 result=in[A[4:0]]

描述 利用输入 A 的 4 个最低有效位乘 B 的 16 位输入

受影响标志 SN：结果的 bit[31]

名称 SHIFTBBA

伪代码

```
if(A[5])
    result=B<<A[4:0];
else
    result=B<<A[4:0];
```

描述 将 B 移动 A 位。16 位或 32 位操作是由配置存储器中的 con32 位决定的。在这两种情况下，移出的位都被传送给标志

受影响标志 结果的 bit[31]

1.10 DPU 接口信号（32 位 DPU）

信号名称	宽度	方向	描述
Misc 信号			
praddrmask	[9:6]	输入	LSM 地址的 OR 掩码
clk	[0]	输入	时钟
clken	[0]	输入	时钟使能
reset	[0]	输入	复位
load config[a/b/o]	[0]	输入	从配置总线装载 abo 寄存器
arc r w	[0]	输入	arc 读/写
arc bus	[31:0]	输入	来自 arc 的总线
arc_access_[a/b/o]	[0]	输入	arc 读/写 abo 寄存器
输入数据			
in0-in7	[31:0]	输入	从上面来的 8 路本地数据连接
in8-in14	[31:0]	输入	从下面来的 7 路本地数据连接
gnet0-gnet15	[31:0]	输入	全局数据连接
lsmval	[31:0]	输入	LSM 值
输入多路复用			
muxasel	[2:0]	输入	选择上面的 8 个输出（与 G0 和 G1

			多路复用器共享）
muxbsel	[2:0]	输入	选择上面的 8 个输出（与选择下面的 7 个输出和自回馈共享）
muxafgsel	[1:0]	输入	为 A 选择上面，下面或全局的数据通路
muxbfsel	[0]	输入	为 B 选择本地或全局数据通路
muxbcf	[0]	输入	为 B 选择移位或不移位
muxop[a/b]sel	[1:0]	输入	选择锁存、不锁存、与或非数据
配置总线&控制			
[a/b/o]总线	[31:0]	输入	配置总线
Muxlat[a/b/o]sel	[0]	输入	选择配置总线或 DPU 数据
Lat[a/b/o][m/s]en	[0]	输入	使能辅助锁存器和主锁存器（激活并对接）
ALU 信号			
Shiftamt	[4:0]	输入	位移数量
Shiftdir	[0]	输入	位移方向
Shiftmode	[2:0]	输入	移位器的移位模式
Ctrl	[0]	输入	加法器的控制进位
ConF	[2:0]	输入	标志控制
Opcode	[7:0]	输入	ALU 的 Opcode
cin	[0]	输入	上一行的进位输出
输出多路复用			
muxslulsmsel	[0]	输入	选择 ALU 输出或 LSM 值
muxoutsel	[0]	输入	选择锁存或不锁存
muxlsmsel	[0]	输入	选择向 lsm 发送锁存数据或未锁存数据
输出			
flag	[1:0]	输出	标志
out	[31:0]	输出	数据输出
aluout	[31:0]	输出	送给 gnet 以使结果值再流通
dpu lsm data	[31:0]	输出	送给 LSM 的数据

dpu lsm addr	[9:0]	输出	送给 LSM 的地址

1.11 DPU 功能——移位器/掩码之后

	16 位	32 位
算术	Add	Add
	Sub	Sub
	饱和 Add	饱和 Add
	饱和 Sub	饱和 Sub
	Inc	Inc
	Dec	Dec
	Neg	Neg
逻辑	AND	AND
	OR	OR
	XOR	XOR
	NAND	NAND
	NOR	NOR
	XNOR	XNOR
	NOT	NOT
	PASS (NOP)	PASS (NOP)
专用函数	ABS	ABS
	MIN	MIN
	MAX	MAX
	Rxor	Rxor
	N/A	DIV
	N/A	LFSR
	N/A	PENC
	N/A	MUXB_by_A(1)
	N/A	SHIFTB_by_A(2)

注意：可以为每个 slice 建立一个 896（28*32）位操作器

1：实现 N:1 多路复用器，其中 2<=N<=8

2：A[4:0]是移位的数量，A[5]是移位的方向

CS2212 ALU Opcode 补充

以下的 opcode 将被加入 CS2212:

ADD8

SUB8

ADDSUB16

SUBADD16

操作: ADD8: 8 位加法运算

Out[7:0]=A[7:0]+B[7:0]

Out[15:8]=A[15:8]+B[15:8]

Out[23:16]=A[23:16]+B[23:16]

Out[31:24]=A[31:24]+B[31:24]

Opcode:

?	?	?	?	?	?	?	?

比特粒度 8 位操作

受影响标志 无可用标志

操作: SUB8: 8 位减法运算

Out[7:0]=A[7:0]+~B[7:0]+1

Out[15:8]=A[15:8]+~B[15:8]+1

Out[23:16]=A[23:16]+~B[23:16]+1

Out[31:24]=A[31:24]+~B[31:24]+1

Opcode:

?	?	?	?	?	?	?	?

比特粒度 8 位操作

受影响标志 无可用标志

操作: ADDSUB16: 16 位加法和减法运算

Out[31:16]=A[31:16]+B[31:16]

Out[15:0]=A[15:0]+~B[15:0]+1

Opcode:

?	?	?	?	?	?	?	?

比特粒度 16 位操作

受影响标志 CO, OV, EQ, SN

操作: SUBADD16: 16 位加法和减法运算

Out[31:16]=A[31:16]+~B[31:16]+1

Out[15:0]=A[15:0]+B[15:0]

Opcode:

?	?	?	?	?	?	?	?

比特粒度 16 位操作

受影响标志 CO, OV, EQ, SN

CS2212 乘法器输出多路复用规范

为了使 A 或 B 操作数能被锁存在 O 寄存器中，在 CS2212 中，改变了乘法器输出的多路复用。这就有效地避开了乘法操作。不过，这要求在 MULT CSM 中的“muxmultlsmsel”域中加入一个比特。“muxmultlsmsel”将以下列方式选择送给 O 寄存器的输入:

Muxmultlsmsel[1:0]	O 寄存器或乘法器的输出结果
2'b00	乘法输出
2'b01	LSM 读出数据
2'b10	操作数 A
2'b11	操作数 B

这项功能使得用户能够在不使用乘法器的主功能时，将其用做一个可以动态配置的路由资源。

CS2212 流程寄存器规范

CS2212 具有一些寄存器，它们或者被用作掩模寄存器，或者被用作流程寄存器。为使用户能够在 A 和 B 操作数通路中使用流程寄存器，并使用掩模寄存器，CS2212 包含一些附加的寄存器。这些寄存器被加在 A 和 B 输入之后，被称为”apipe”和”bpipe”。这些寄存器可分别被”muxapipe”和”muxbpipe”信号旁路掉。参照 CS2212 DPU 方框图，

可以看到这些寄存器和多路复用器的布置。多路复用器是按以下方式选择的:

"muxapipe"	操作数 A	"muxbpipe"	操作数 B
0	旁路寄存器	0	旁路寄存器
1	流程寄存器	1	流程寄存器

CS2212 LSM 写数据多路复用规范

CS2212 中规定，写入 LSM 的可以是位移器输出，也可以是 ALU 输出。为实现这一附加功能，在 LSM 写数据通路上加入一个多路复用器。该多路复用器被称为"muxlsmwd"，并可按以下方式由"muxlsmwdsel"信号选择:

"muxlsmwd"	LSM 写数据
0	ALU 输出
1	移位器输出

参照 CS2212 方框图察看"muxlsmwd"的布置。

2.1 概述

fabric 是可以重新配置的，该过程由配置位控制。将配置位装入 fabric 的方法是，首先发布 arc 指令（通过装入存储器），随后配置控制器将配置位传送到该 fabric 的配置平面中。

下表提供了软件信息，这些信息有关每条配置信号对应于地址空间中的哪个地址。由于还没有确定配置的基地址，以下地址从 0 开始。

2.2 详述

. 头 16 位是嵌入地址

. 首位（bit[127]）是校验位。硬件检测每行（128 位）的偶校验。有关校验，请查阅 Dani 的 ARC 扩展协议。

. 以下的地址是相对于一些基地址的。

. 每行的 128 位中带有 112 位的配置数据

. 在装载配置期间，硬件在每个周期最多存储 112 位行。

. 对于某种配置来说，可以跳过该配置不需要的那些行。

. 现在，slice 地址将不被嵌入在当前的地址变换表中，并且可以暂时从当前的地址变换表中删除。用户可参照 ARC 扩展规范，察看怎样在一个操作中配置多个 slice。

2.3 地址变换

地址	数据	
	HL DPU/MULT 寄存器 32 31	0
0x0	保留	row0 寄存器 A
0x1		row0 寄存器 B
0x2		row0 寄存器 0
0x3		
0x4		row1 寄存器 A
0x5		row1 寄存器 B
0x6		row1 寄存器 0
0x7		
0x1c		row7 高 mult 寄存器 A
0x1D		row7 高 mult 寄存器 B
0X1E		row7 高 mult 寄存器 0
0X1F		
0X20		row7 低 mult 寄存器 A
0X21		row7 低 mult 寄存器 B
0X22		row7 低 mult 寄存器 0
0X23		保留
0X24-0X3F		
	111 84|83 CSM 56	55 28|27 0
0x40	row 0(第 0 行) DPU 配置 1	row 0 DPU 配置 0
0x41	row 0 DPU 配置 3	row 0 DPU 配置 2
0x42	row 0 DPU 配置 5	row 0 DPU 配置 4
0x43	row 0 DPU 配置 7	row 0 DPU 配置 6

0x44	row 1 DPU 配置 1		row 1 DPU 配置 0	
0x45	row 1 DPU 配置 3		row 1 DPU 配置 2	
0x46	row 1 DPU 配置 5		row 1 DPU 配置 4	
0x47	row 1 DPU 配置 7		row 1 DPU 配置 6	
.				
.				
0x5C	第 7 行 MULT 高位配置 3	row7 MULT 高位配置 2	row7 MULT 高位配置 1	row7 MULT 高位配置 0
0X5D	row7 MULT 高位配置 7	row7 MULT 高位配置 6	row7 MULT 高位配置 5	row7 MULT 高位配置 4
0X5E	row7 MULT 低位配置 3	row7 MULT 低位配置 2	row7 MULT 低位配置 1	row7 MULT 高位配置 0
0X5F	row7 MULT 低位配置 7	row7 MULT 低位配置 6	row7 MULT 低位配置 5	row7 MULT 低位配置 4
0X60-0X7F	保留			

DPU 配置细节	
2:0	Muxasel
5:3	muxbsel
7:6	muxbfsel
9:8	muxafgsel
14:10	Shiftamt
15	Shiftdir
23:16	Opcode
24	muxbcfsel
25	Latapipe
26	Latbpipe
27	Latopipe
29:28	muxopasel
31:30	muxopbsel
34:32	shiftmode

37:35	Conf
38	muxalulsmsel
39	muxoutsel
40	Ctrl
41	Lsm 读/写使能（交替地使能读和写，偶数行读，奇数行写）
55:42	保留
MULT 配置详述	
2:0	Muxasel
5:3	Muxbsel
7:6	Muxbfsel
9:8	Muxnfgsel
11:10	Muxbcfsel
12	Muxafghsel
13	Muxopasel
14	Muxopbsel
15	Muxmultlsmsel
16	Muxoutsel
17	Latapipe
18	Latbpipe
19	Latopipe
20	Lsm 读/写使能（类似于 editlepipe，为第 6 个 dpu 来的第 3 个 lsm 使用读使能，为高端乘法器使用写使能
27:21	保留
CONPROL 配置	
0x80	Tile A PLA 乘积项 0
0x81	Tile A PLA 乘积项 0
0x82	.
0x83	.
	.
0x9D	.
0x9e	Tile A PLA 乘积项 30

0x9f	Tile A PLA 乘积项 31
0xa0-0xbf	保留
0xc0	Tile A 基于 tile 的控制
0xc1	Tile A 基于行的控制 L0
0xc2	Tile A 基于行的控制 L1
0xc3	Tile A 基于行的控制 L2
0xc4	Tile A 基于行的控制 L3
0xc5-0xff	保留
PLA X 乘积项 Y 详述（其中，X 在(A,B,C,D)中，Y 在[0，15]中	
15:0	使比特位置 b 可用，即，PLA X 中的乘积项 Y 与～input[b]进行“与”操作
31:16	使比特位置 b 可用，即，PLA X 中的乘积项 Y 与 input[(b-16)]进行“与”操作
63:32	使比特位置 b 可用，即，PLA X 中的结果（b-32）与乘积项 Y 进行“或”操作
111:64	保留
基于 Tile 的控制详述	
1:0	SRBDMA: muxdmasetsel
33:2	GNET: hortnetsel: (net5, net4,..., net0)
45:34	GNET: vertnetsel: (net2, net1, net0)
53:46	LSMNET: 读地址(transen3 到 4, trien3, transen2 到 3, trien2, transen1 到 2, trien1, transen0 到 1, trien0)
61:54	LSMNET: 读数据(transen3 到 4, trien3, transen2 到 3, trien2, transen1 到 2, trien1, transen0 到 1, trien0)
69:62	LSMNET: 写地址(transen3 到 4, trien3, transen2 到 3, trien2, transen1 到 2, trien1, transen0 到 1, trien0)
77:70	LSMNET: 写数据(transen3 到 4, trien3, transen2 到 3, trien2, transen1 到 2, trien1, transen0 到 1, trien0)
基于行的控制详述	
Lx[1:0]	行(2x+1)-(2x): SRBR clken[0]x2
Lx[7:2]	行(2x+1)-(2x): SRBR csmaddrmask[2:0]x2
Lx[11:8]	行(2x+1)-(2x): SRBR muxinterruptsetsel[1:0]x2

Lx[13:12]	行(2x+1)-(2x): SRBR inten[0]x2
Lx[37:14]	行(2x+1)-(2x): PLA_in_sel[5:0]x4
Lx[39:38]	行(2x+1)-(2x): lsmregsel[0]x2
Lx[47:40]	行(2x+1)-(2x): HSTATE_sel[1:0]x4
Lx[51:48]	行(2x+1)-(2x): HSTATE_regsel[0]x4
Lx[55:52]	Lsm x: addrmatch[3:0]
Lx[59:56]	Lsm x: matchen[3:0]
Lx[61:60]	Lsm x: mode[1:0]
Lx[63:62]	Lsm x: wmode[1:0]
Lx[1:64]	行(2x+1)-(2x): SRB reginit[3:0]x2
0x100-0x1ff	Tile B配置
0x200-0x2ff	Tile C配置
基于slice的配置	
0x300[1:0]	完成多路复用选择
其它slice地址(在当前实现中未使用)	
0x400-0x7ff	Slice 1的配置
0x800-0xbf	Slice 2的配置
0xc00-0cfff	Slice 3的配置
基于fabric的配置	
0x380-0x383	Slice0 选择
0x780-0x783	Slice1 选择
0xb80-0xb83	Slice2 选择
0xf80-0xf83	Slice3 选择
基于fabric的配置详述	
L0 7:0	Tile A gnet三态使能1位x8
L0 23:8	Tile A hctl net三态使能1位x16
L1 7:0	Tile B gnet三态使能1位x8
L1 23:8	Tile B hctl net三态使能1位x16

L2 7:0	Tile C gnet 三态使能 1 位 x8
L2 23:8	Tile C hctl net 三态使能 1 位 x16
L3 1:0	PIO: 控制输入多路复用选择（输入到 fabric）2 位 x1
L3 5:2	PIO: 控制输出 mux one hot sel（fabric 的输出）1 位 x4
L3 7:6	PIO: 数据输入多路复用选择（输入 fabric）2 位 x1
L3 11:8	PIO: 数据输出 mux one hot sel（fabric 的输出）1 位 x4

CSMMULT	Muxasel	同 dpu
	Muxbsel	同 dpu
	Muxafgsel	同 dpu
	Muxbfsel	同 dpu
	Muxbcfsel	0: 低 16 位 1, 2: 高 16 位 3: 高 24 位（对于 24 位乘来说）
	Muxafghsel	0: 低 16 位 1: 高 16 位
	Latapipe	同 dpu
	Latbpipe	同 dpu
	Latopipe	同 dpu
	Muxopasel	0: 锁存的值 1: 未锁存的值
	Muxopbsel	
	Muxmultlsmsel	0: 乘法器输出 1: lsm 输出
	muxoutsel	0: 锁存的值 1: 未锁存的值
LSM	mode	0 字节 1 字 2 双字 3 双字
	wmode	
	wen	0: 写禁止 1: 写使能

	addrmatch	DPU 提供的 4 位地址与这些值相匹配，以便使能一个读/写操作
	matchen	单独地使能(1)/禁止(0)这 4 位匹配地址
	lsmregsel	来自 DPU（地址/数据）和 CSM（wen）的信号是有选择地可寄存的。0=寄存 1=不寄存
PLA	PLA_in_sel	请参阅下面的内容
GNET	hortnetsel	选择来自一个 tile 的 9 个数据通路输出（7 个 dpu, 2 个 mult），每个 tile 中有 6 个这种输出（有 6 个水平全局网和 3 个垂直全局网），水平全局网不仅驱动水平全局网，还驱动垂直全局网多路复用器的输入 0: dpu 0 1: dpu 1 2 : dpu 2 3: dpu 3 4: dpu 4 5: dpu 5 6: dpu 6 7: mult 7 其它: mult 8
	vertnetsel	从 6 个水平全局网或“本地”水平全局网 0, 1, 2 中选择（由于“本地”的三态可被禁止，因此这两个值可以不同）
	Hortnettrien	水平全局网的三态使能。每个全局网上有 4 个三态驱动器。三态使能是一个单 hot 值，由每个 slice 驱动一个比特。
单元名称	信号名称	注解
DPU/MULT	a/b/o 寄存器	DPU 和 MULT 中流程寄存器的值
CSMDPU	muxasel	与 muxF, muxG0 和 muxG1 交迭，选择: 值 muxA muxF muxG0 muxG1 0 前面 8 个 自身 gnet0 gnet8 1 前面 7 个 后面 1 个 gnet1 32'h0 2 前面 6 个 后面 2 个 gnet2 32'h0

		3 前面5个 后面3个 gnet3 32'h0 4 前面4个 后面4个 gnet4 LSM值 5 前面3个 后面5个 gnet5 ALU输出（寄存器0之前） 6 前面的2 后面6个 gnet6 {DPU输出[30:0],cin} 7 前面的1 后面7个 gnet7 32'h0
	Muxbsel	与上面相同
	muxafgsel	选择MuxA 选择MuxF 选择MuxG0 选择MuxG1
	Muxbfsel	同上
	muxbcfsel	0 选择MuxBF 1 选择移位器输出
	Latapipe	0: 寄存器A不锁存输入 1: 寄存器A锁存输入
	Latbpipe	0: 寄存器B不锁存输入 1: 寄存器B锁存输入
	latopipe	0: 寄存器O不锁存输入 1: 寄存器O锁存输入
	Muxopasel	00: 选择被锁存的数据 01: 选择未被锁存的数据 10: 选择经过“与”运算的数据 11: 选择经过“或”运算的数据
	muxopbsel	
	Shiftamt	提供给移位器的移位数量
	shiftdir	请参阅s:\doc\teams\dpu\dpu_specX.doc 1.12部分
	shiftmode	
	conf	标志选择 0 前面的标志 1 RXOR

		2 GT 3 LT 4 EQ 5 SN 6 CO 7 OV
	Ctrl	Addcnt, subcnt 等操作数的控制进位
	opcode	请参阅 s:/doc/teams/dpu/dpu_spec#.doc
	muxalulsmsel	0 选择 ALU 输出 1 选择 LSM 数据
	muxoutsel	0 选择被锁存的数据 1 选择未被锁存的数据
		4'bxxx1 或 4'b0000: slice 0 驱动三态总线 4'bxx10 : slice 1 驱动三态总线 4'bx100 : slice 2 驱动三态总线 4'b1000 : slice 3 驱动三态总线
LSMNET	三态使能	共有 4 个 lsmnet (本地系统存储器网络), 读/写, 地址/数据。由于在一个 tile 中有 16 个 lsm (本地系统存储器), 因此, 每一个都有 15 个三态使能 0: 禁止 1: 使能
	传输门使能	三态使能 0: 禁止 1: 使能
SRBDMA	muxdmasetsel	这是一个 DMA 请求落下之前的多路复用器控制信号。每个 tile 有一个该信号。它从每个 tile 的第 0 行取状态值。 0 第 0 行状态位[3] 1 第 0 行状态位[2] 2 第 0 行数据通路单元标志 dpuflag[1] 3 地
SRBR	每行有一个 SRBR。因此, 以下所有信号都是以行为基础的	

	Clken	这是一个以行为基础的时钟使能 0: 将 SRB[3]用作时钟使能 1: 总是对 SRB 计时
	Csmaddrmask	这是状态位驱动 CSM 之前的一个“与”掩码
	Muxinterruptsetsel	多路复用器选择中断寄存器之前的多路复用，该信号驱动触发器的“SET”输入
	Inten	中断使能 0: 禁止 1: 使能
	SRB reg_initval	状态位初始值
Hstate	hstate_sel	请参阅以下内容
	hstate_trien	类似于水平网三态使能，从每个 slics 的一个位中设置该信号的值，随后按以下方式区分优先级 4'bxxx1 或 4'b0000: slice 0 驱动三态总线 4'bxx10 : slice 1 驱动三态总线 4'bx100 : slice 2 驱动三态总线 4'b1000 : slice 3 驱动三态总线
	Hstate_regsel	在开始 pla 输入之前，有选择地寄存 hstate 位 0- 寄存 1- 不寄存
PIO	mux sel (输入到 fabric)	0: 选择存储单元 0 1: 选择存储单元 1 2: 选择存储单元 2 3: 选择存储单元 3
	mux sel(fabric 的输出)	这是一个 hot 编码，每一个比特都来自每个 slice 的 fabric 级配置的相应比特位置 4'bxxx1 或 4'b0000: 从 slice 0 输出 4'bxx10 : 从 slice 1 输出 4'bx100 : 从 slice 2 输出 4'b1000 : 从 slice 3 输出

. input_mux Slice 0 PLA0(每一位都具有单独的使能)

. PLAin[0]是与 DPU0 相关的多路复用器输入

. PLAin[1]是与 DPU0 相关的多路复用器输入

. STATE0[n]是 slice0 的主状态输出

. LSTATE0[n]是 slice0 的次级状态输出

. FLAG[n]是 slice0 的主标志输出

. LSTATE0[n]是 slice0 的下次级标志输出

. HBIT0[n]是水平状态总线

. HBIT0[n] 在输入多路复用器之前需要一个可选的寄存器

. IO[7:0]是来自与每个 slice 相关的管脚的 8 个 I/O 位

. bit 0 输入，DATA 是来自数据通路的 hortnet mux（水平网络多路复用）输出。每个 tile 有 8 个水平网络多路复用器。控制流程从每个 tile 选择低 16 位 hortnet mux7。应注意，三态门的输入是 mux7 的输出值。

. 去掉 tile D，因此，它所提供的数据（STATE*[31:24],FLAG[27:21]...）接地

PLA_in_sel 是 64 比 1 的多路复用选择。每个 PLA 都具有其中的 16 个选择（每一位都是单独控制的）

	bit 63:32	Bit32:4	bit3	bit2	bit1	bit0
PLAin0	STATE0[31:0]	FLAG0[27:0]	HBIT0[0]	LSTATE0[0]	IO[0]	DATA[0]
PLAin2	STATE0[31:0]	FLAG0[27:0]	HBIT0[2]	LSTATE0[1]	IO[1]	DATA[2]
PLAin4	STATE0[31:0]	FLAG0[27:0]	HBIT0[4]	LSTATE0[2]	IO[2]	DATA[4]
PLAin6	STATE0[31:0]	FLAG0[27:0]	HBIT0[6]	LSTATE0[3]	IO[3]	DATA[6]
PLAin8	STATE0[31:0]	FLAG0[27:0]	HBIT0[8]	LSTATE0[4]	IO[4]	DATA[8]
PLAin10	STATE0[31:0]	FLAG0[27:0]	HBIT0[10]	LSTATE0[5]	IO[5]	DATA[10]
PLAin12	STATE0[31:0]	FLAG0[27:0]	HBIT0[12]	LSTATE0[6]	IO[6]	DATA[12]
PLAin14	STATE0[31:0]	FLAG0[27:0]	HBIT0[14]	开始	IO[7]	DATA[14]
PLAin1	STATE0[31:0]	FLAG0[27:0]	HBIT0[1]	LFLAG0[0]	IRQ[0]	DATA[1]
PLAin3	STATE0[31:0]	FLAG0[27:0]	HBIT0[3]	LFLAG0[1]	IRQ[1]	DATA[3]
PLAin5	STATE0[31:0]	FLAG0[27:0]	HBIT0[5]	LFLAG0[2]	IRQ[2]	DATA[5]

PLAin7	STATE0[31:0]	FLAG0[27:0]	HBIT0[7]	LFLAG0[3]	IRQ[3]	DATA[7]
PLAin9	STATE0[31:0]	FLAG0[27:0]	HBIT0[9]	LFLAG0[4]	IRQ[4]	DATA[9]
PLAin11	STATE0[31:0]	FLAG0[27:0]	HBIT0[11]	LFLAG0[5]	IRQ[5]	DATA[11]
PLAin13	STATE0[31:0]	FLAG0[27:0]	HBIT0[13]	LFLAG0[6]	IRQ[6]	DATA[13]
PLAin15	STATE0[31:0]	FLAG0[27:0]	HBIT0[15]	DMA REQ	IRQ[7]	DATA[15]

. Horz_mux Slice0 PLA0(每位都具有单独的使能)

. HMUX[0]是16个水平状态行之一;

. 每个mux(多路复用)也都需要三态使能。

	Bit3	Bit2	Bit1	Bit0
HMUX[0]	STATE0[24]	STATE0[16]	STATE0[8]	STATE0[0]
HMUX[2]	STATE0[25]	STATE0[17]	STATE0[9]	STATE0[1]
HMUX[4]	STATE0[26]	STATE0[18]	STATE0[10]	STATE0[2]
HMUX[6]	STATE0[27]	STATE0[19]	STATE0[11]	STATE0[3]
HMUX[8]	STATE0[28]	STATE0[20]	STATE0[12]	STATE0[4]
HMUX[10]	STATE0[29]	STATE0[21]	STATE0[13]	STATE0[5]
HMUX[12]	STATE0[30]	STATE0[22]	STATE0[14]	STATE0[6]
HMUX[14]	STATE0[31]	STATE0[23]	STATE0[15]	STATE0[7]
HMUX[1]	STATE0[24]	STATE0[16]	STATE0[8]	STATE0[0]
HMUX[3]	STATE0[25]	STATE0[17]	STATE0[9]	STATE0[1]
HMUX[5]	STATE0[26]	STATE0[18]	STATE0[10]	STATE0[2]
HMUX[7]	STATE0[27]	STATE0[19]	STATE0[11]	STATE0[3]
HMUX[9]	STATE0[28]	STATE0[20]	STATE0[12]	STATE0[4]
HMUX[11]	STATE0[29]	STATE0[21]	STATE0[13]	STATE0[5]
HMUX[13]	STATE0[30]	STATE0[22]	STATE0[14]	STATE0[6]
HMUX[15]	STATE0[31]	STATE0[23]	STATE0[15]	STATE0[7]

5

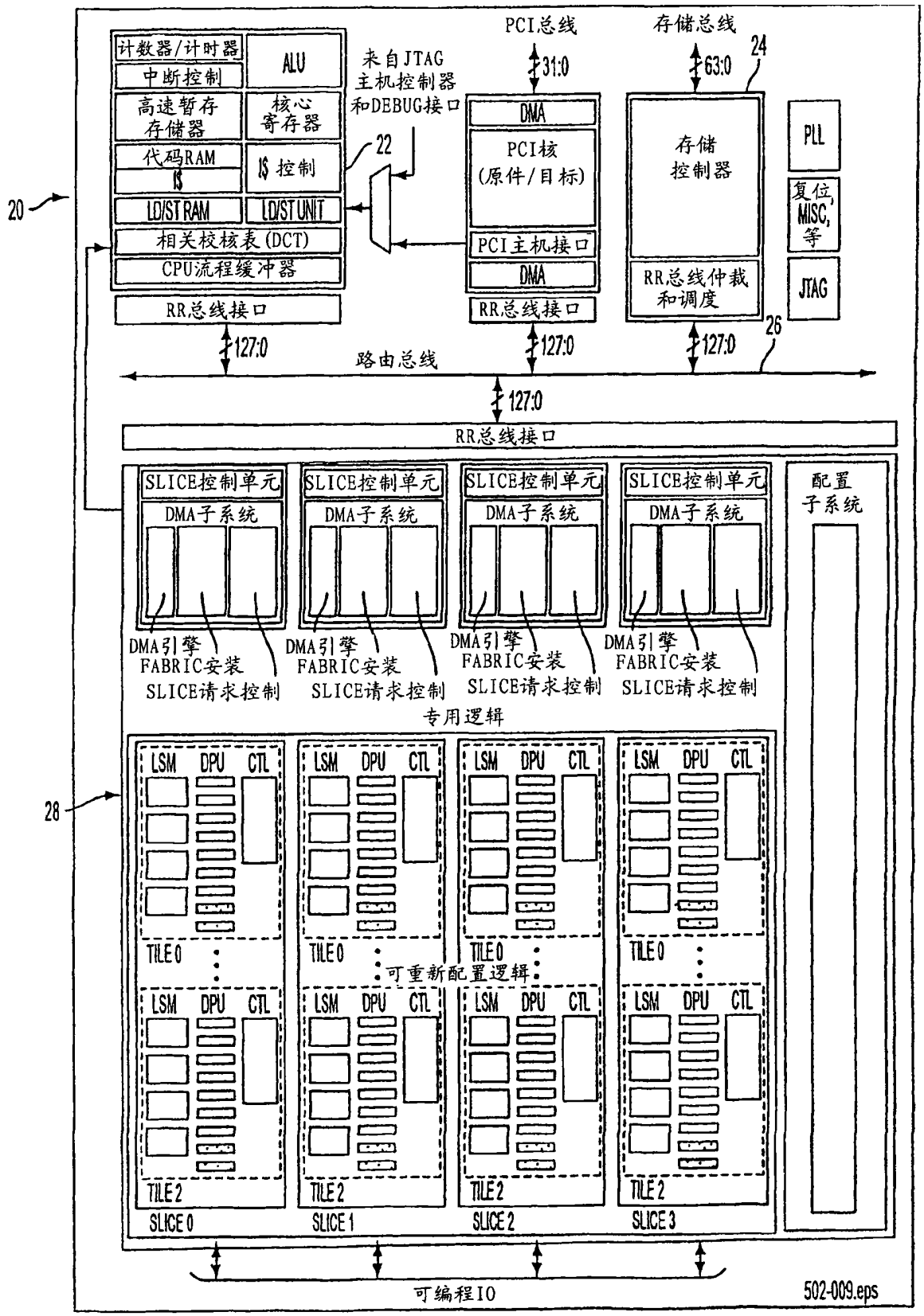

20
计数器/计时器
中断控制
ALU
高速暂存存储器
核心寄存器
代码RAM
IS
IS 控制
LD/ST RAM
LD/ST UNIT
相关校核表(DCT)
CPU流程缓冲器
RR总线接口
来自JTAG主机控制器和DEBUG接口
22
PCI总线
31:0
DMA
PCI核(原件/目标)
PCI主机接口
DMA
RR总线接口
存储总线
63:0
24
存储控制器
RR总线仲裁和调度
PLL
复位,MISC,等
JTAG
127:0
路由总线
26
RR总线接口
SLICE控制单元
DMA子系统
DMA引擎
FABRIC安装
SLICE请求控制
专用逻辑
配置子系统
28
LSM
DPU
CTL
TILE 0
TILE 2
可重新配置逻辑
SLICE 0
SLICE 1
SLICE 2
SLICE 3
可编程IO
502-009.eps

图 1

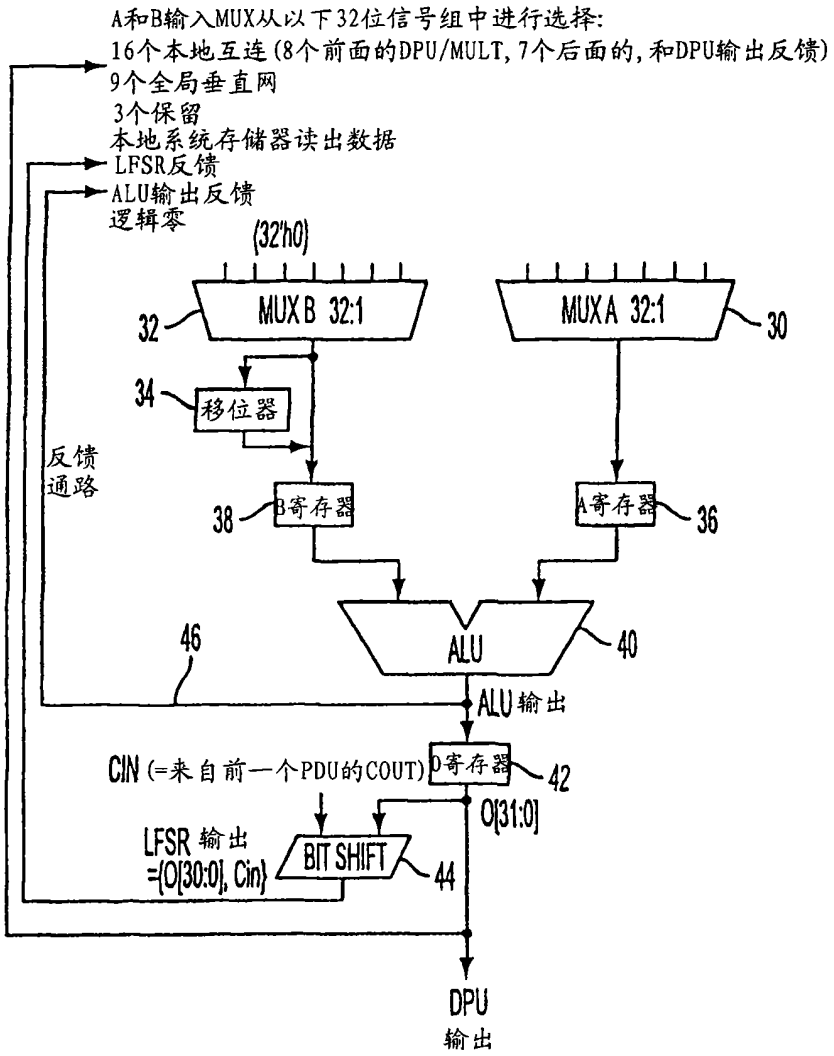

A和B输入MUX从以下32位信号组中进行选择:
16个本地互连(8个前面的DPU/MULT,7个后面的,和DPU输出反馈)
9个全局垂直网
3个保留
本地系统存储器读出数据
LFSR反馈
ALU输出反馈
逻辑零
(32'h0)
MUX B 32:1
32
MUX A 32:1
30
34
移位器
反馈
通路
38
B寄存器
A寄存器
36
ALU
40
46
ALU 输出
CIN (=来自前一个PDU的COUT)
D寄存器
42
O[31:0]
LFSR 输出
={O[30:0], Cin}
BIT SHIFT
44
DPU
输出

图 2

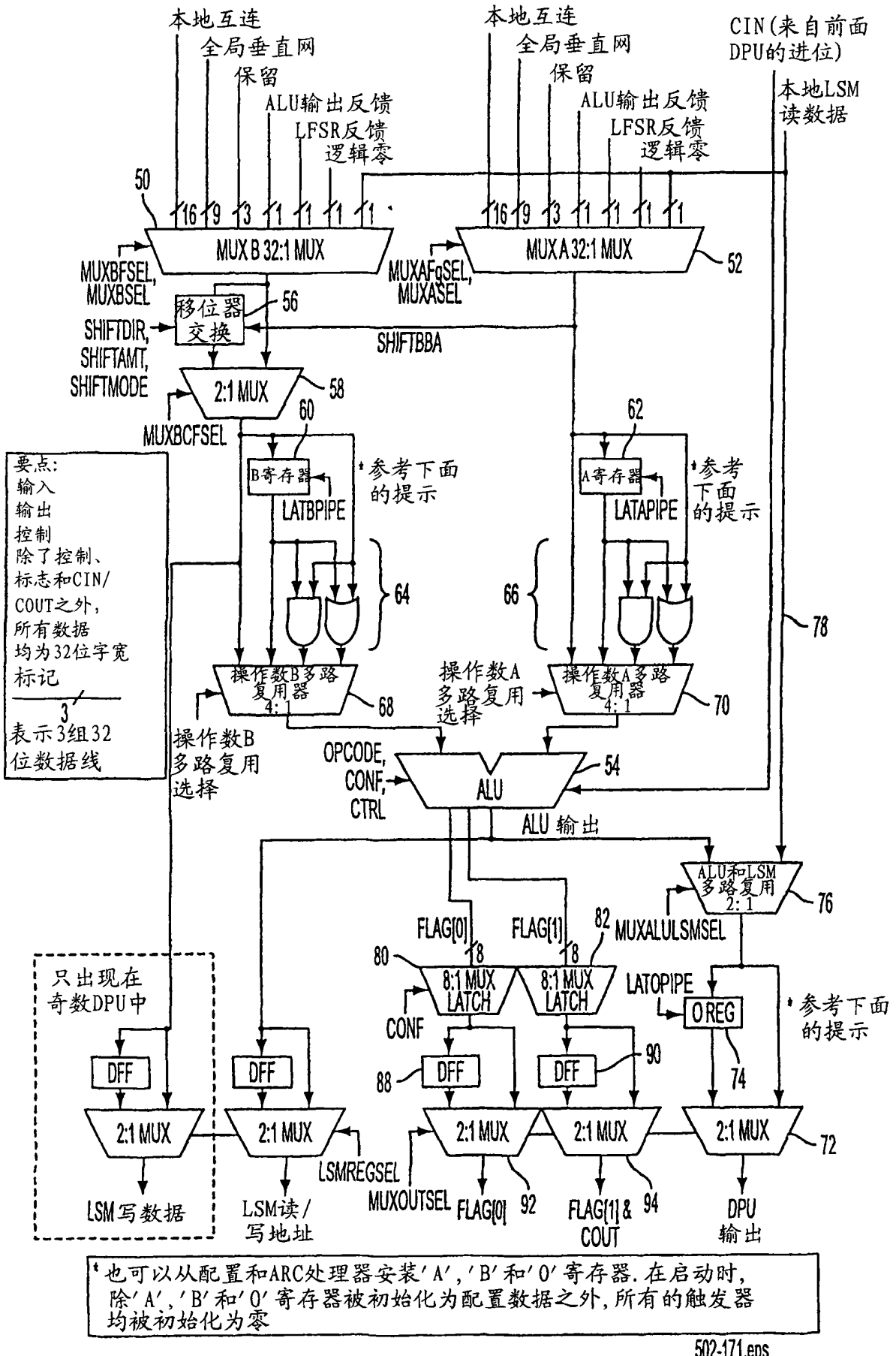

本地互连
全局垂直网
保留
ALU输出反馈
LFSR反馈
逻辑零
本地互连
全局垂直网
保留
ALU输出反馈
LFSR反馈
逻辑零
CIN(来自前面DPU的进位)
本地LSM读数据
50
16 9 3 1 1 1 1
16 9 3 1 1 1 1
MUX B 32:1 MUX
MUX A 32:1 MUX
52
MUXBFSEL, MUXBSEL
MUXAFgSEL, MUXASEL
移位器交换
56
SHIFTDIR, SHIFTAMT, SHIFTMODE
SHIFTBBA
2:1 MUX
58
MUXBCFSEL
60
62
B寄存器
A寄存器
LATBPIPE
LATAPIPE
*参考下面的提示
*参考下面的提示
要点:
输入
输出
控制
除了控制、标志和CIN/COUT之外,
所有数据
均为32位字宽
标记
3
表示3组32位数据线
64
66
78
操作数B多路复用器 4:1
操作数A多路复用器 4:1
68
70
操作数A多路复用选择
操作数B多路复用选择
OPCODE, CONF, CTRL
ALU
54
ALU 输出
ALU和LSM多路复用 2:1
76
FLAG[0]
FLAG[1]
82
MUXALULSMSEL
80
8:1 MUX LATCH
8:1 MUX LATCH
LATOPIPE
OREG
*参考下面的提示
只出现在奇数DPU中
CONF
DFF
DFF
DFF
DFF
88
90
74
2:1 MUX
2:1 MUX
2:1 MUX
2:1 MUX
2:1 MUX
72
LSMREGSEL
MUXOUTSEL
FLAG[0]
92
FLAG[1] & COUT
94
LSM写数据
LSM读/写地址
DPU输出
*也可以从配置和ARC处理器安装'A','B'和'0'寄存器.在启动时,除'A','B'和'0'寄存器被初始化为配置数据之外,所有的触发器均被初始化为零
502-171.eps

图 3

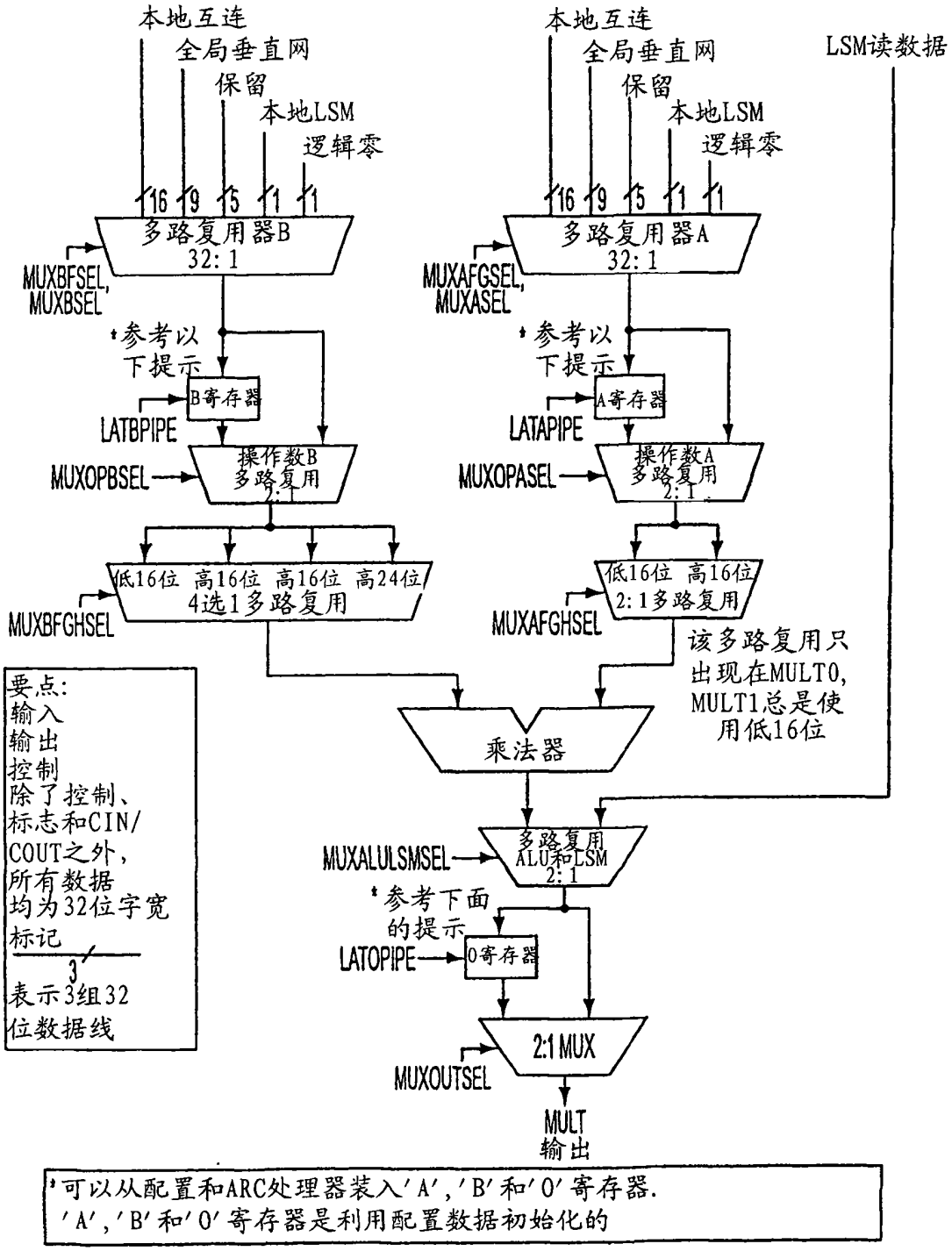

本地互连
全局垂直网
保留
本地LSM
逻辑零
16 9 5 1 1
多路复用器B
32:1
MUXBFSEL,
MUXBSEL
*参考以
下提示
B寄存器
LATBPIPE
操作数B
多路复用
2:1
MUXOPBSEL
低16位 高16位 高16位 高24位
4选1多路复用
MUXBFGHSEL
本地互连
全局垂直网
保留
本地LSM
逻辑零
LSM读数据
16 9 5 1 1
多路复用器A
32:1
MUXAFGSEL,
MUXASEL
*参考以
下提示
A寄存器
LATAPIPE
操作数A
多路复用
2:1
MUXOPASEL
低16位 高16位
2:1多路复用
MUXAFGHSEL
该多路复用只
出现在MULT0,
MULT1总是使
用低16位
乘法器
要点:
输入
输出
控制
除了控制、
标志和CIN/
COUT之外,
所有数据
均为32位字宽
标记
3
表示3组32
位数据线
多路复用
ALU和LSM
2:1
MUXALULSMSEL
*参考下面
的提示
0寄存器
LATOPIPE
2:1 MUX
MUXOUTSEL
MULT
输出
*可以从配置和ARC处理器装入'A','B'和'0'寄存器.
'A','B'和'0'寄存器是利用配置数据初始化的

图 4

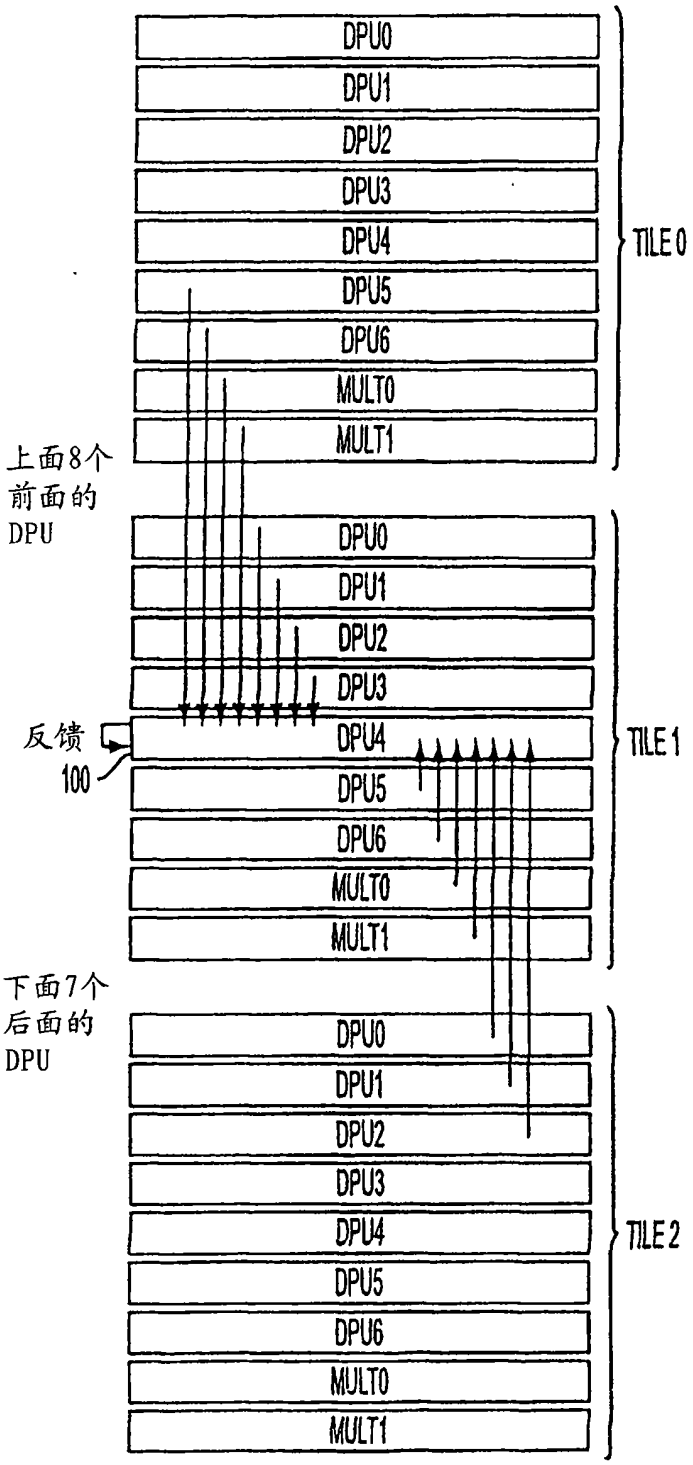

DPU0
DPU1
DPU2
DPU3
DPU4
DPU5
DPU6
MULT0
MULT1
TILE 0
上面8个
前面的
DPU
DPU0
DPU1
DPU2
DPU3
反馈
100
DPU4
DPU5
DPU6
MULT0
MULT1
TILE 1
下面7个
后面的
DPU
DPU0
DPU1
DPU2
DPU3
DPU4
DPU5
DPU6
MULT0
MULT1
TILE 2

图 5

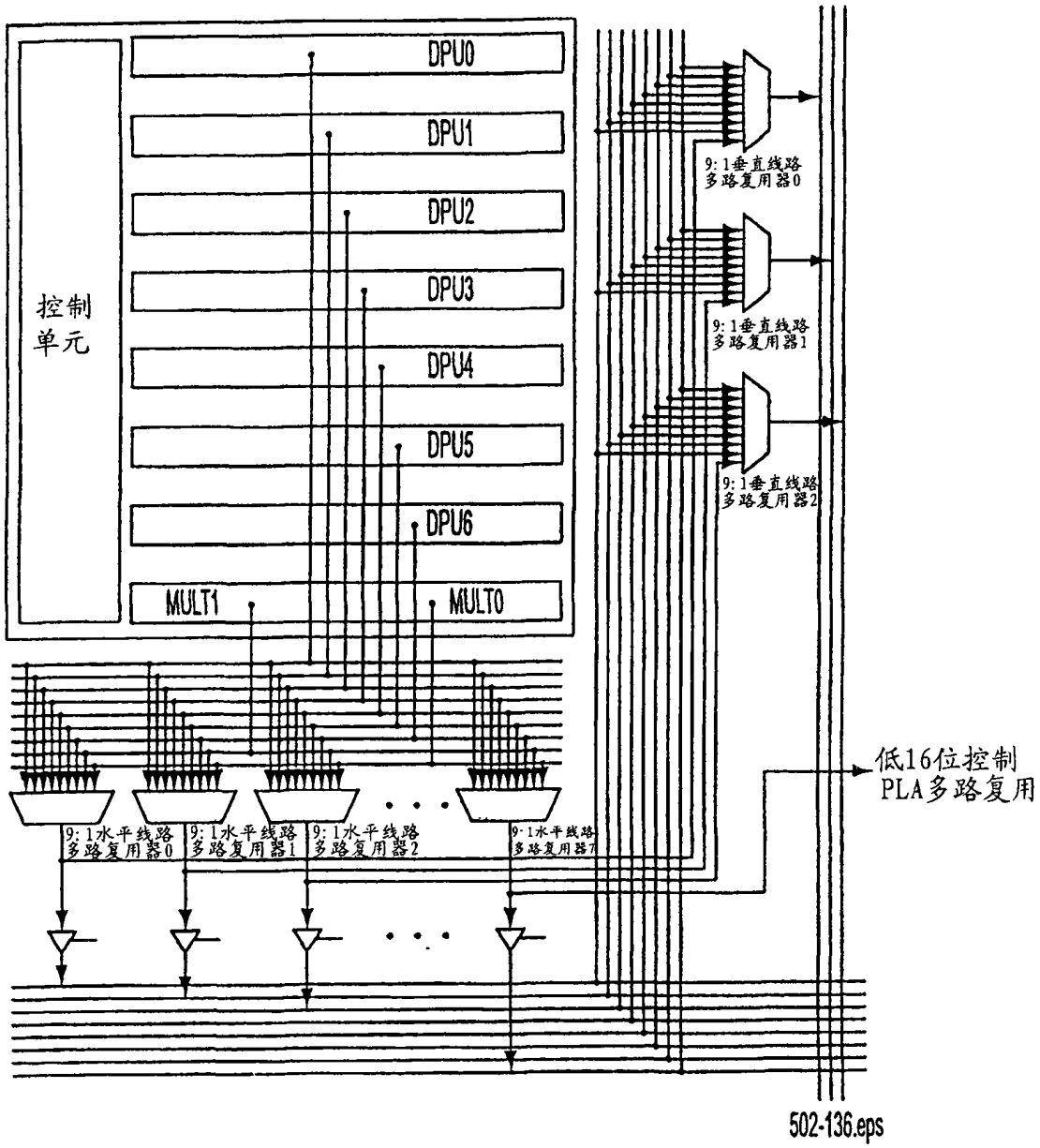

DPU0
DPU1
DPU2
DPU3
DPU4
DPU5
DPU6
MULT1
MULT0
控制
单元
9:1垂直线路
多路复用器0
9:1垂直线路
多路复用器1
9:1垂直线路
多路复用器2
9:1水平线路
多路复用器0
9:1水平线路
多路复用器1
9:1水平线路
多路复用器2
低16位控制
PLA多路复用
502-136.eps

图 6

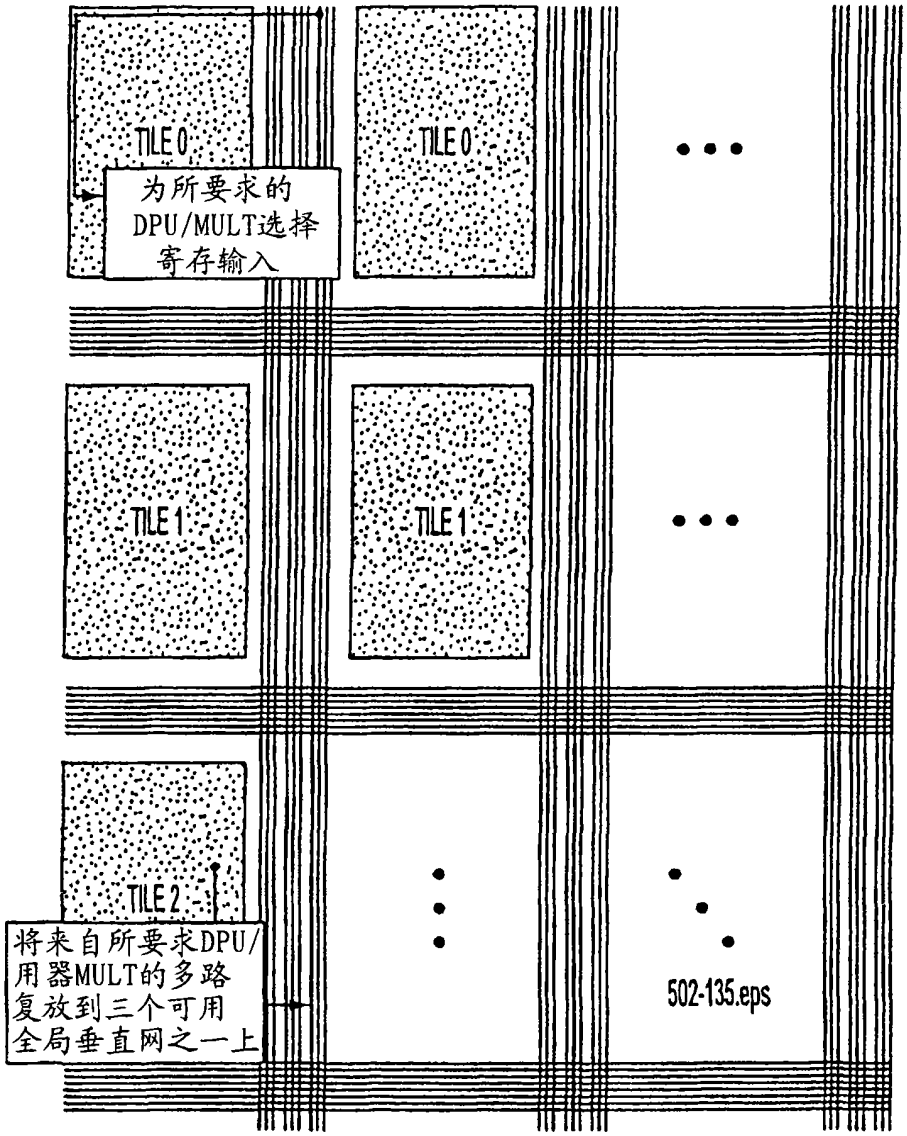

TILE 0
为所要求的
DPU/MULT选择
寄存输入
TILE 0
TILE 1
TILE 1
TILE 2
将来自所要求DPU/
用器MULT的多路
复放到三个可用
全局垂直网之一上
502-135.eps

图 7

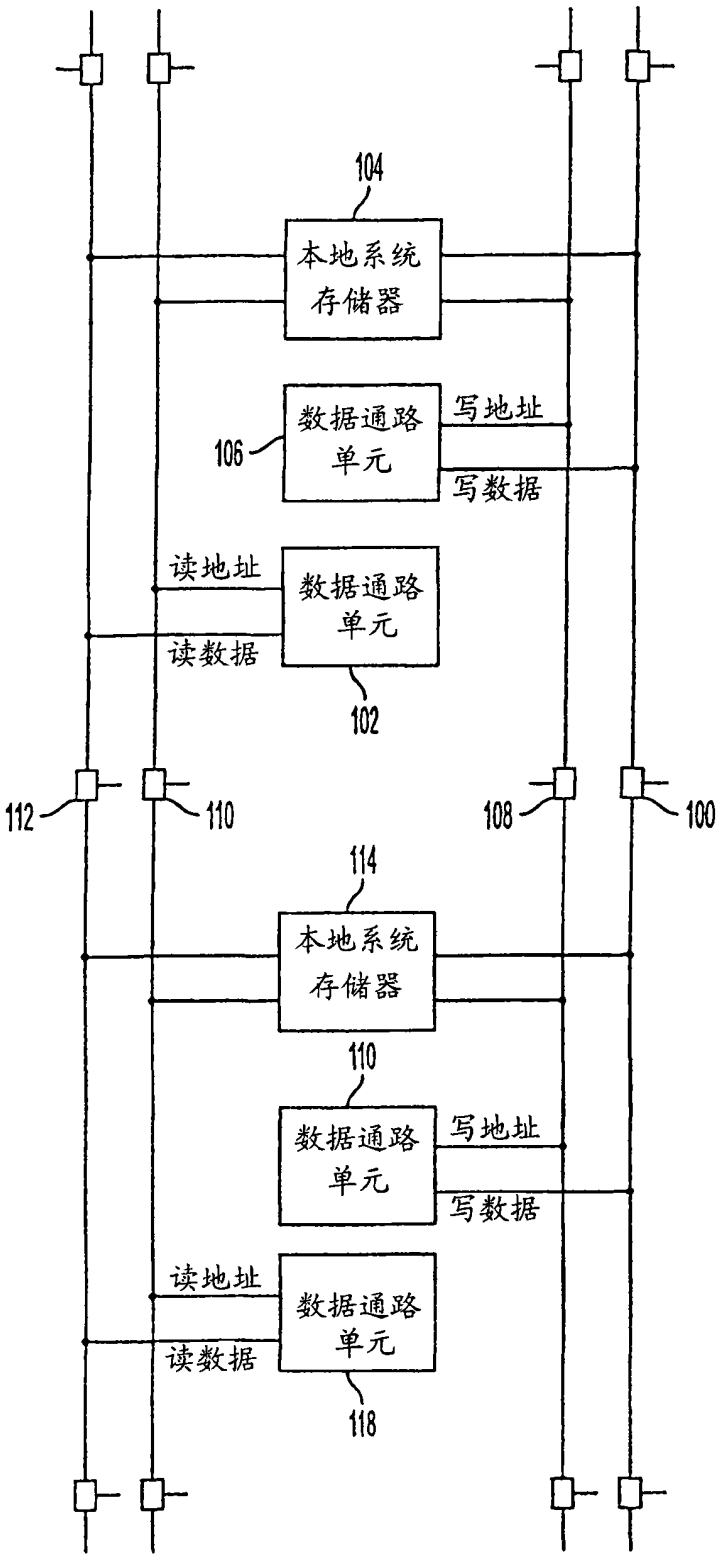

104
本地系统
存储器
数据通路
单元
106
写地址
写数据
读地址
数据通路
单元
读数据
102
112
110
108
100
114
本地系统
存储器
110
数据通路
单元
写地址
写数据
读地址
数据通路
单元
读数据
118

图 8

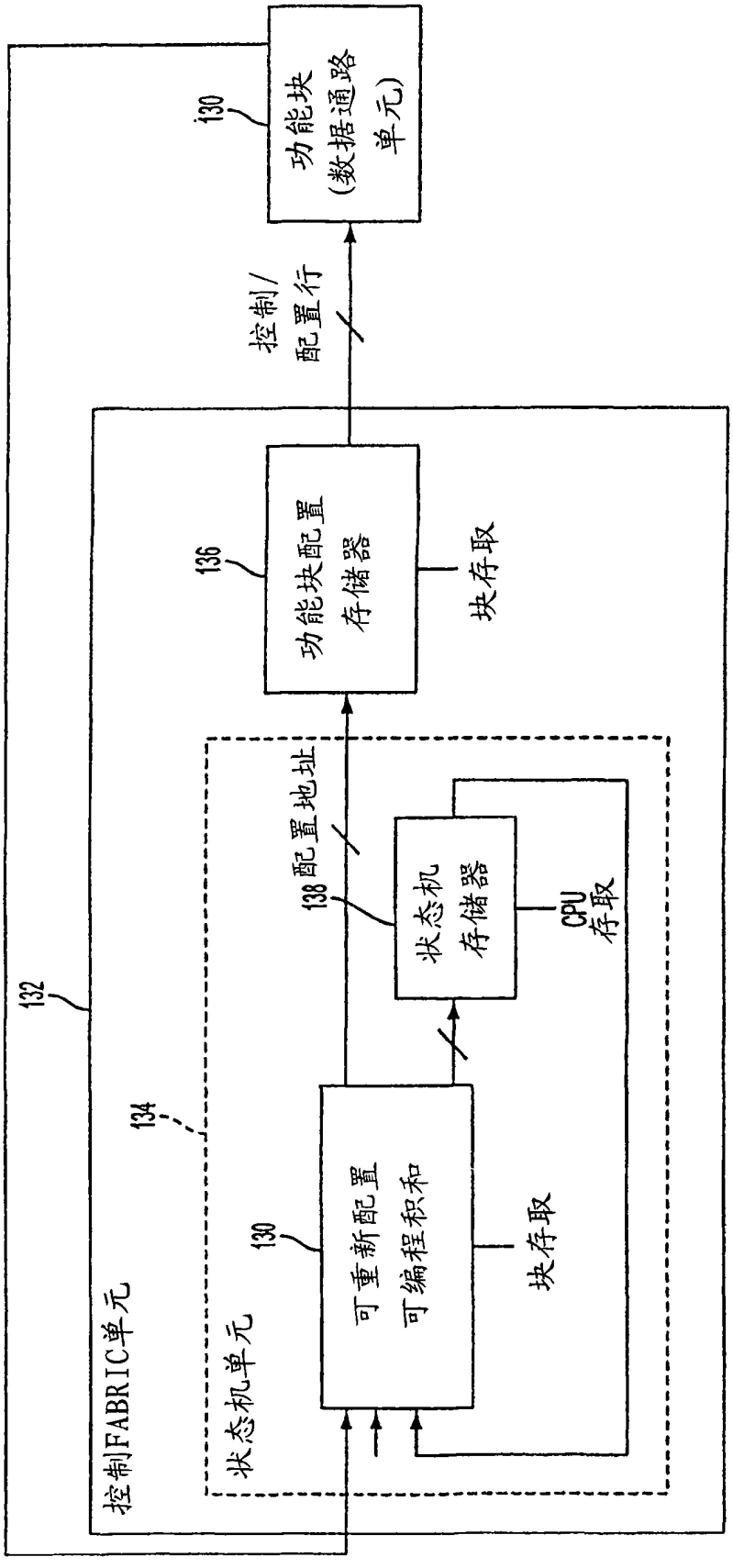

130
功能块
(数据通路
单元)
控制/
配置行
136
功能块配置
存储器
块存取
132
控制FABRIC单元
134
状态机单元
130
可重新配置
可编程积和
138配置地址
状态机
存储器
CPU
存取
块存取

图 9

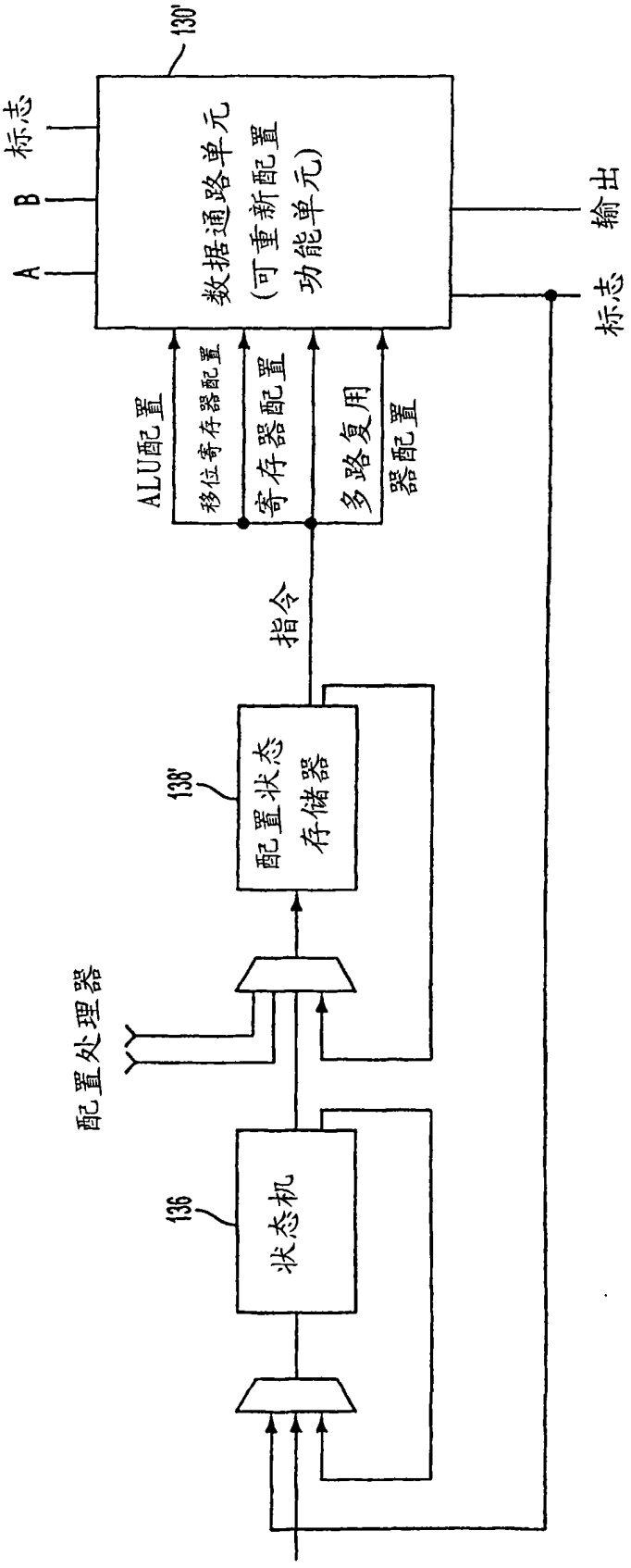

130'
A
B
标志
数据通路单元
(可重新配置
功能单元)
输出
标志
ALU配置
移位寄存器配置
寄存器配置
多路复用
器配置
指令
138'
配置状态
存储器
配置处理器
136
状态机

图 10A

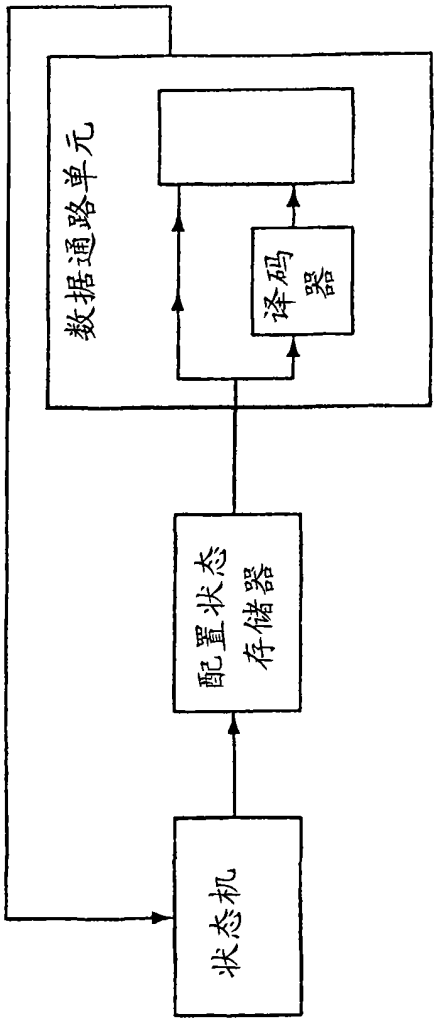


图 10B

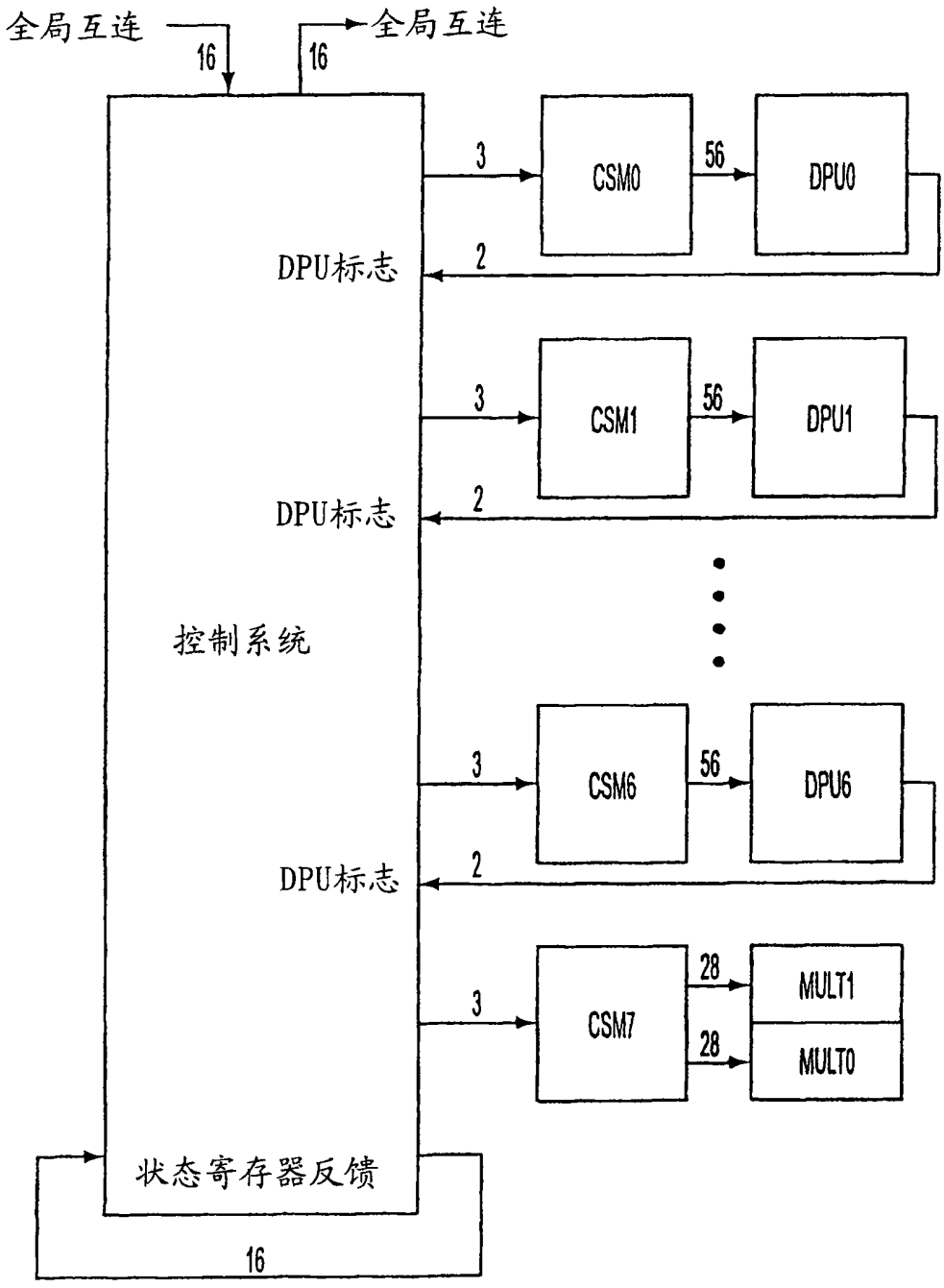

全局互连
16
全局互连
16
3
CSM0
56
DPU0
DPU标志
2
3
CSM1
56
DPU1
DPU标志
2
控制系统
3
CSM6
56
DPU6
DPU标志
2
3
CSM7
28
MULT1
28
MULT0
状态寄存器反馈
16

图 11

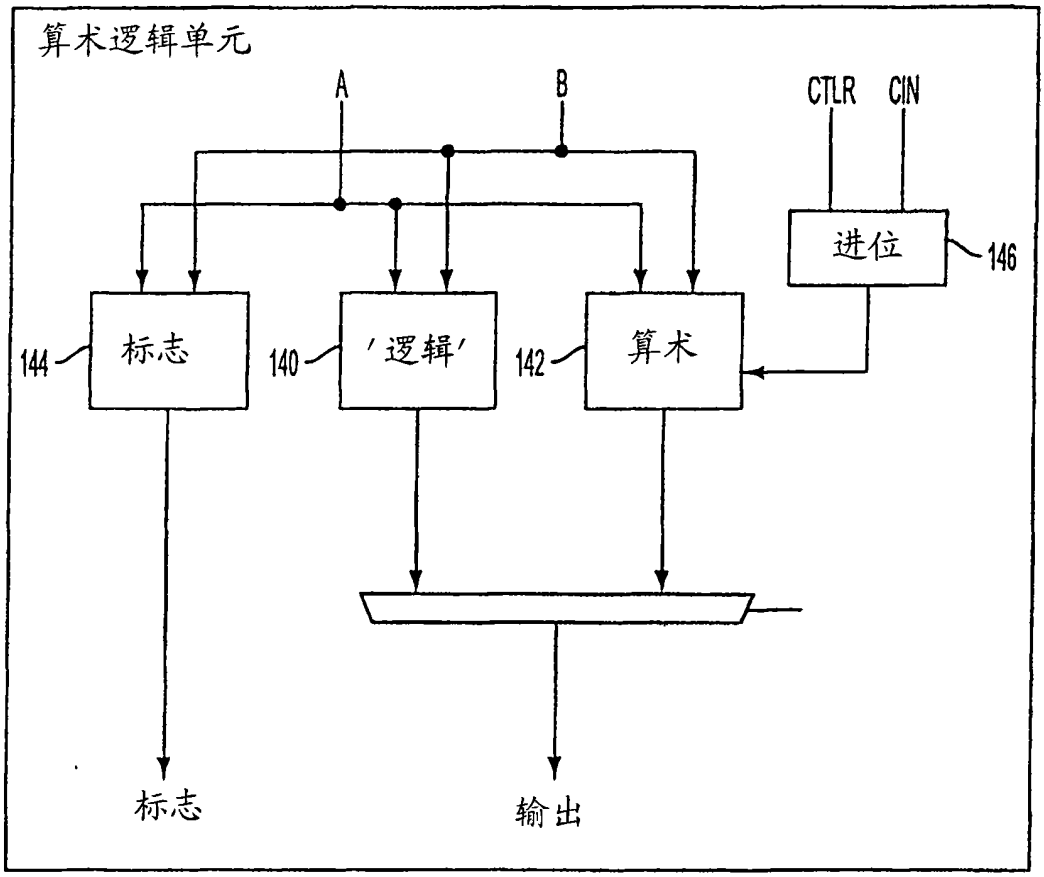


图 12

DPU的Opcode一览表		
OPCODE	编码(16进制)	描述
ADD	00	加
ADD16	80	两个并行16位加
ADDC	10	带进位的加
ADDCNT	18	带控制进位的加
SUB	09	减
SUB16	89	两个并行16位减
SUBC	11	带进位的减
SUBCNT	19	带控制进位的减
SADD	20	饱和加
SADD16	a0	两个并行16位饱和加
SADDCNT	38	带饱和的ADDCNT
SSUB	29	带饱和的减
SSUB16	a9	带饱和的16位减
SSUBCNT	39	带饱和的SUBCNT
INC	0c	B值递增
DEC	06	递减
NEG	0d	B值求反
ABS	2d	B的绝对值
ABS16	ad	两个并行16位求绝对值
CSUB	49	条件减
AND	60	逐位与
OR	62	逐位或
NAND	61	逐位NAND
NOR	63	逐位NOR
XOR	64	逐位XOR
XNOR	65	逐位XNOR
PASSA	66	通过A
PASSB	6e	通过B
NOTA	67	A反转
NOTB	6f	B反转
MIN	0b	求最小值
MIN16	8b	两个并行16位求最小值
MAX	2b	求最大值
MAX16	ab	两个并行16位求最大值
PENC	7f	优先编码
MUXBBA	ce	B路中选A路
SHIFTBBA	de	将B移A位

图 13A

快速译码的必要BIS								
OPCODE				Cmux1	Cmux0	Amux1	Amux0	Bmux
ADD	0	0	0	0	0	0	0	0
ADD16	1	0	0	0	0	0	0	0
ADDC	0	0	0	1	0	0	0	0
ADDCNT	0	0	0	1	1	0	0	0
SUB	0	0	0	0	1	0	0	1
SUB16	1	0	0	0	1	0	0	1
SUBC	0	0	0	1	0	0	0	1
SUBCNT	0	0	0	1	1	0	0	1
SADD	0	0	1	0	0	0	0	0
SADD16	1	0	1	0	0	0	0	0
SADDCNT	0	0	1	1	1	0	0	0
SSUB	0	0	1	0	1	0	0	1
SSUB16	1	0	1	0	1	0	0	1
SSUBCNT	0	0	1	1	1	0	0	1
INC	0	0	0	0	1	1	0	0
DEC	0	0	0	0	0	1	1	0
NEG	0	0	0	0	1	1	0	1
ABS	0	0	1	0	1	1	0	1
ABS16	1	0	1	0	1	1	0	1
CSUB	0	1	0	0	1	0	0	1
AND	0	1	1	0	0	0	0	0
OR	0	1	1	0	0	0	1	0
NAND	0	1	1	0	0	0	0	1
NOR	0	1	1	0	0	0	1	1
XOR	0	1	1	0	0	1	0	0
XNOR	0	1	1	0	0	1	0	1
PASSA	0	1	1	0	0	1	1	0
PASSB	0	1	1	0	1	1	1	0
NOTA	0	1	1	0	0	1	1	1
NOTB	0	1	1	0	1	1	1	1
MIN	0	0	0	0	1	0	1	1
MIN16	1	0	0	0	1	0	1	1
MAX	0	0	1	0	1	0	1	1
MAX16	1	0	1	0	1	0	1	1
PENC	0	1	1	1	1	1	1	1
MUXBBA	1	1	0	0	1	1	1	0
SHIFTBBA	1	1	0	1	1	1	1	0

bmux: 0: B 1: B
amux: 0: A 1: 20 3: 1
cmux: 0: 0 1: 1 2: cin 3: cnt

OR: 0: 正 1: 负
OR 0&1/2X3: 通过

图 13B

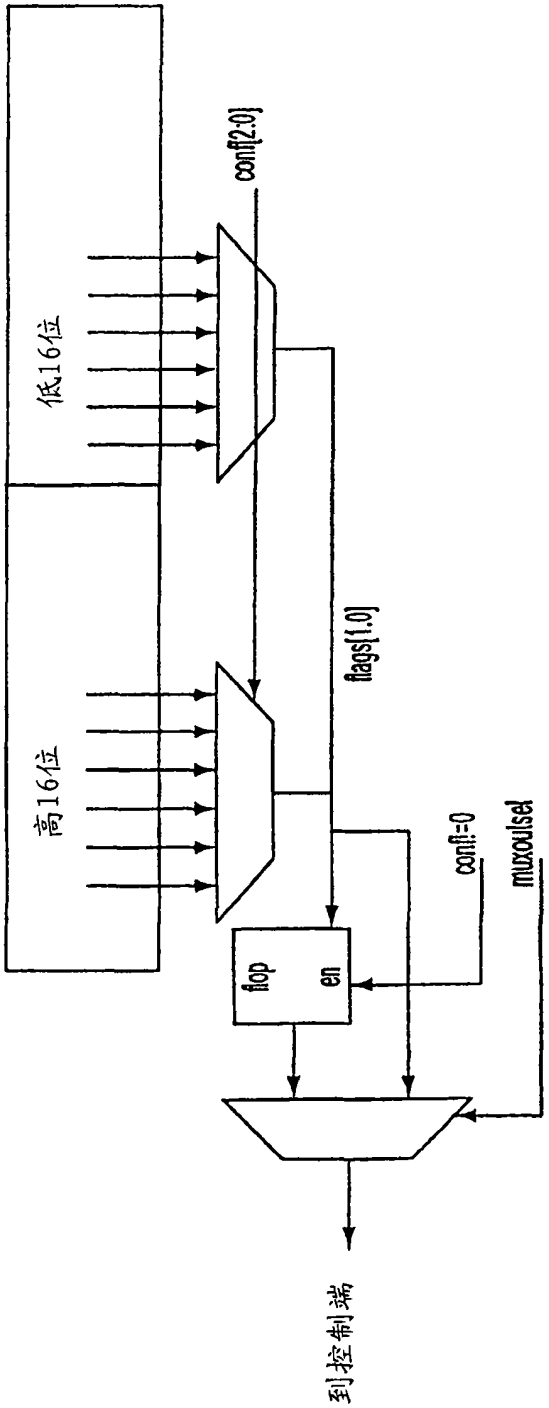


图 14

移位方向	移位模式	操作
0	000	32位逻辑右移
1	000	32位逻辑左移
0	001	32位算术右移
1	001	32位算术左移
X	010	(a) 低16位标志扩展到32位 (a)
X	011	(b) 从{shiftdir[0], shiftamt[4:0]} 生成常量
X	100	(c) 低16位复制到高16位
X	101	(d) 高16位复制到低16位
X	110	(e) 低16位与高16位互换
0	111	(f) 16位算术右移 (若shiftamt[4]为1, 则输出将是无法预料的)
1	111	(g) 字节互换 (32位比特从{b0, b1, b2, b3} 改变到{b3, b2, b1, b0}

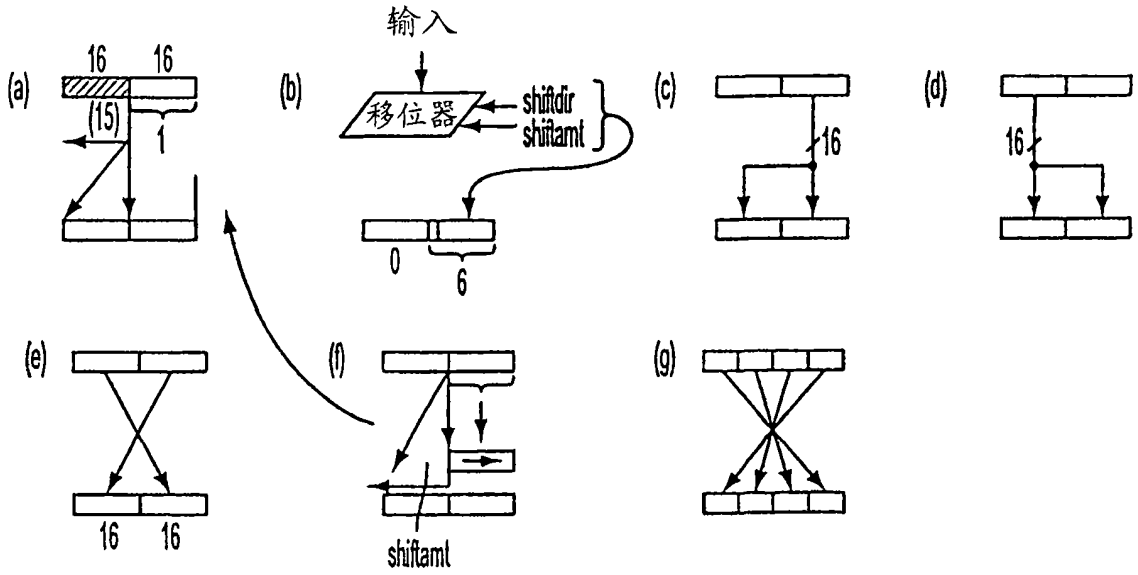


图 15

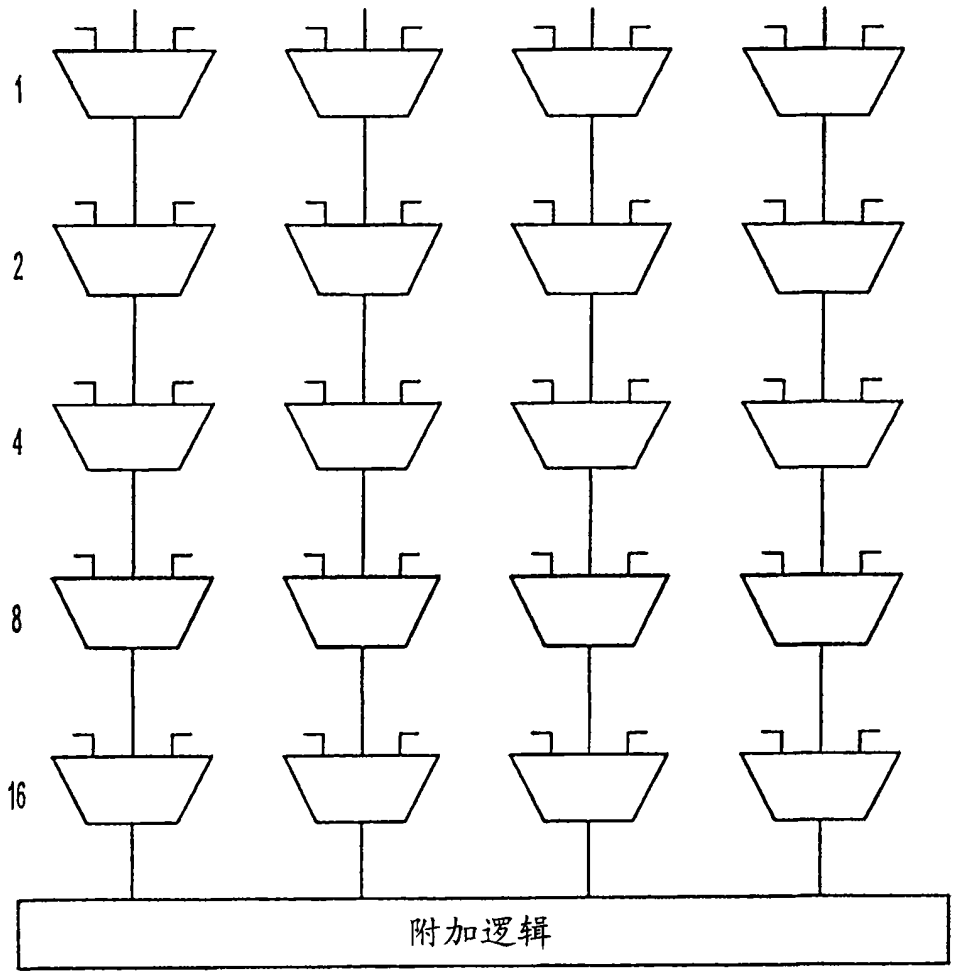


图 16

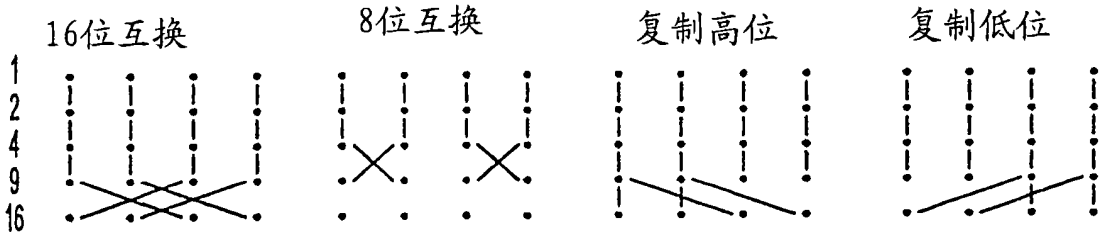


图 17

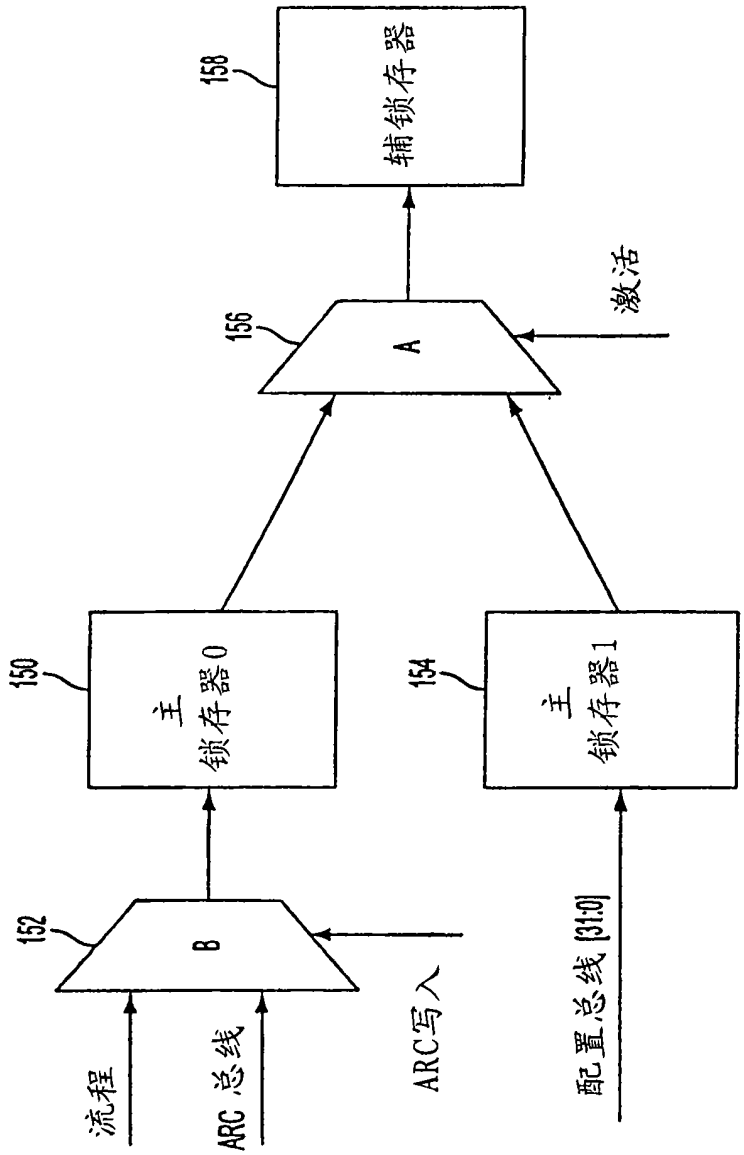

158
辅锁存器
156
A
激活
150
主
锁存器0
154
主
锁存器1
152
B
流程
ARC 总线
ARC写入
配置总线 [31:0]

图 18

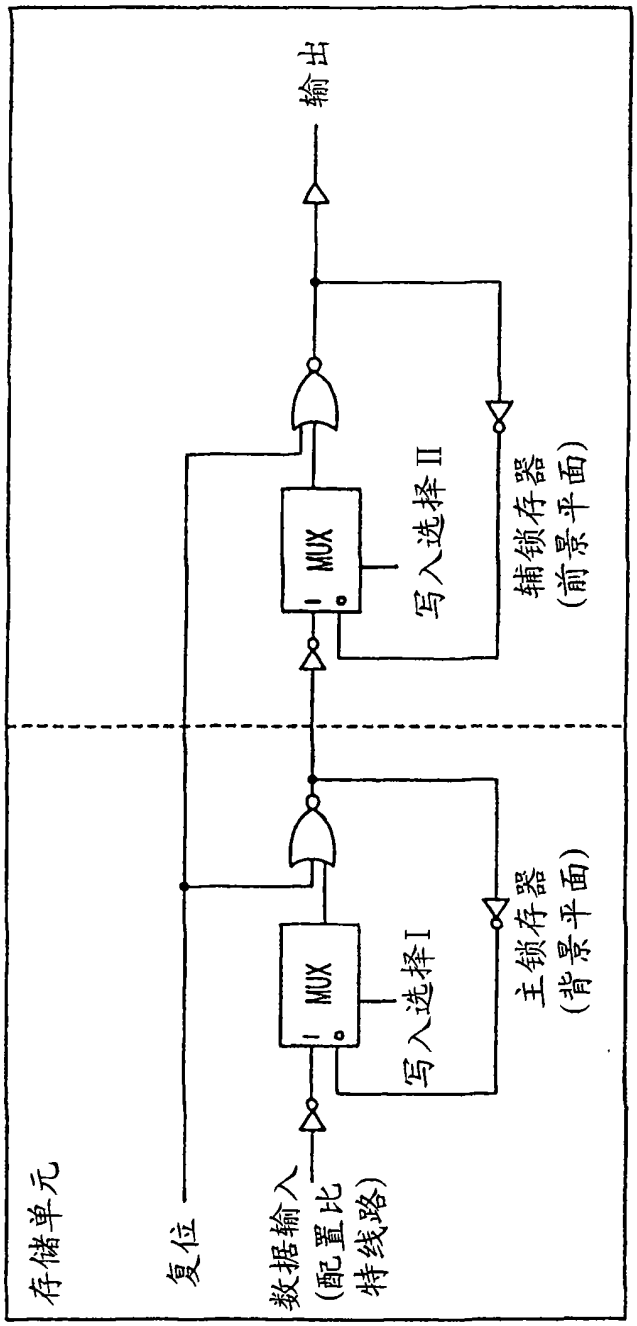

存储单元
复位
数据输入
(配置比特线路)
MUX
写入选择I
主锁存器
(背景平面)
MUX
写入选择II
辅锁存器
(前景平面)
输出

图 19

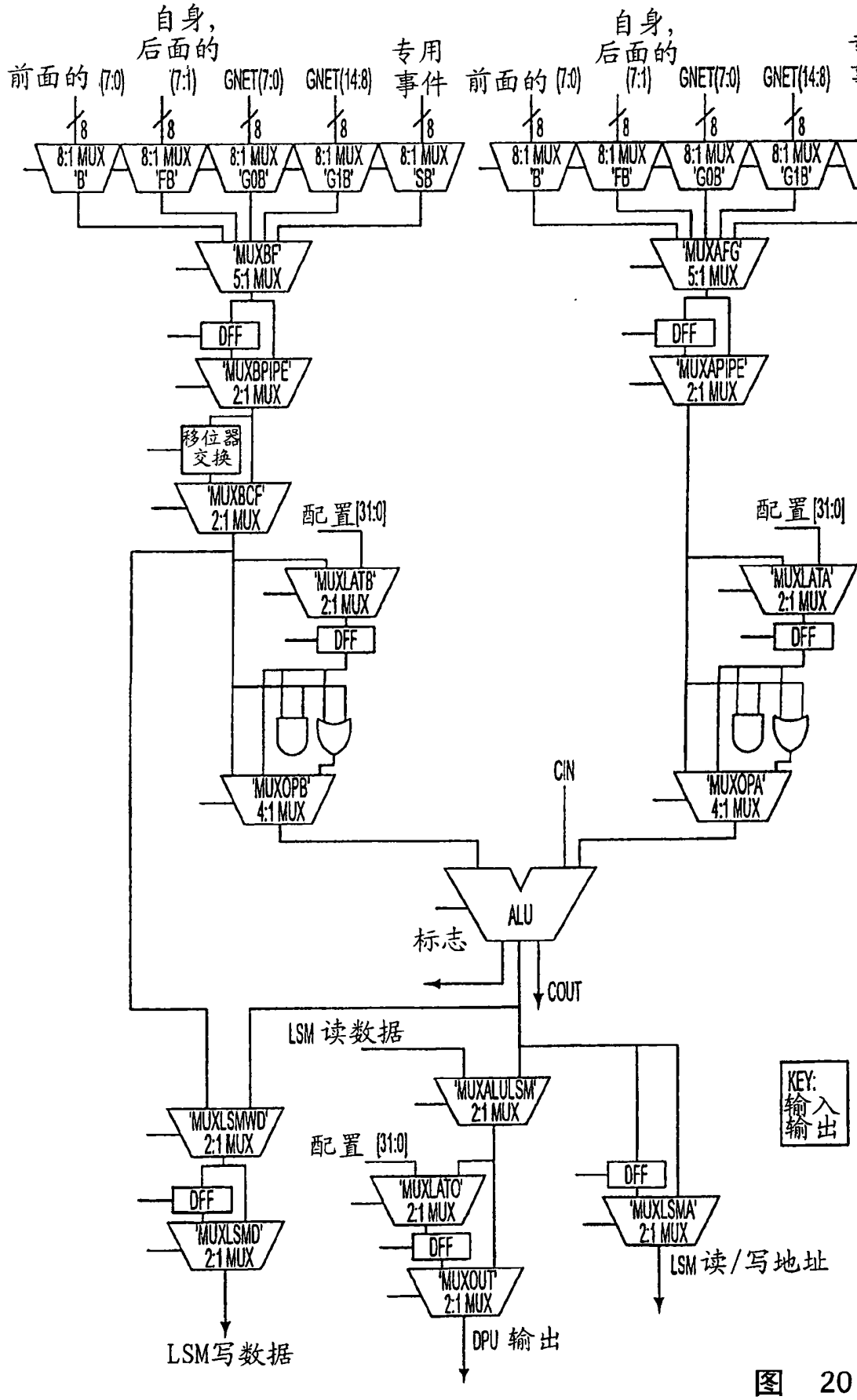

自身,
后面的
前面的 (7:0)
(7:1)
GNET(7:0)
GNET(14:8)
专用
事件
8
8:1 MUX 'B'
8:1 MUX 'FB'
8:1 MUX 'G0B'
8:1 MUX 'G1B'
8:1 MUX 'SB'
'MUXBF' 5:1 MUX
'MUXAFG' 5:1 MUX
DFF
'MUXBPIPE' 2:1 MUX
'MUXAPIPE' 2:1 MUX
移位器交换
'MUXBCF' 2:1 MUX
配置[31:0]
'MUXLATB' 2:1 MUX
'MUXLATA' 2:1 MUX
'MUXOPB' 4:1 MUX
'MUXOPA' 4:1 MUX
CIN
ALU
标志
COUT
LSM 读数据
'MUXALULSM' 2:1 MUX
'MUXLSMWD' 2:1 MUX
'MUXLATC' 2:1 MUX
'MUXOUT' 2:1 MUX
'MUXLSMD' 2:1 MUX
'MUXLSMA' 2:1 MUX
KEY:
输入
输出
LSM写数据
DPU 输出
LSM 读/写地址

图 20

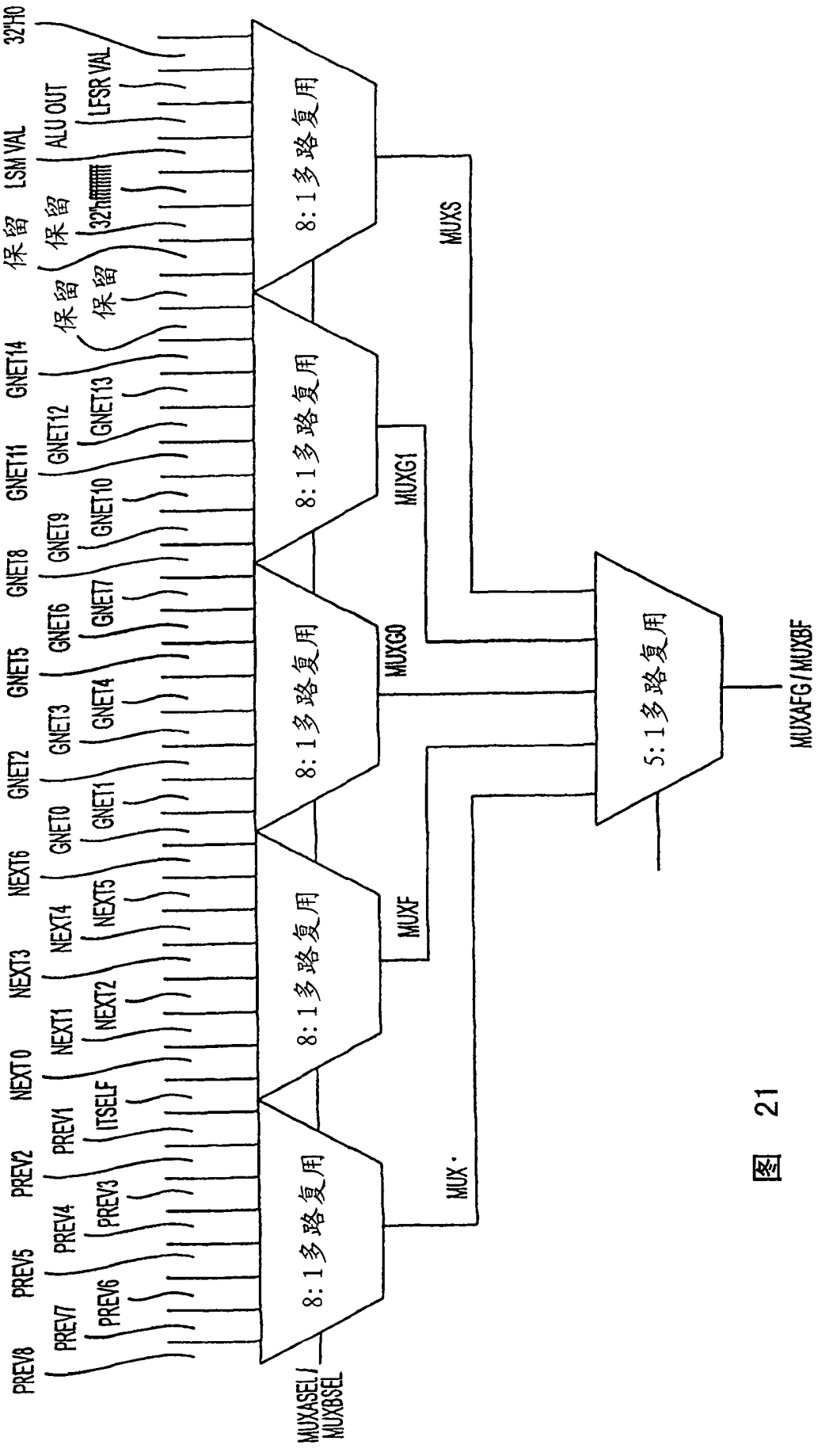

PREV8
PREV7
PREV6
PREV5
PREV4
PREV3
PREV2
PREV1
ITSELF
NEXT0
NEXT1
NEXT2
NEXT3
NEXT4
NEXT5
NEXT6
GNET0
GNET1
GNET2
GNET3
GNET4
GNET5
GNET6
GNET7
GNET8
GNET9
GNET10
GNET11
GNET12
GNET13
GNET14
保留
保留
保留
LSM VAL
32'h0
32'hffffffff
ALU OUT
LFSR VAL
MUXASEL/
MUXBSEL
8:1多路复用
8:1多路复用
8:1多路复用
8:1多路复用
5:1多路复用
MUX·
MUXF
MUXG0
MUXG1
MUXS
MUXAFG/MUXBF

图 21

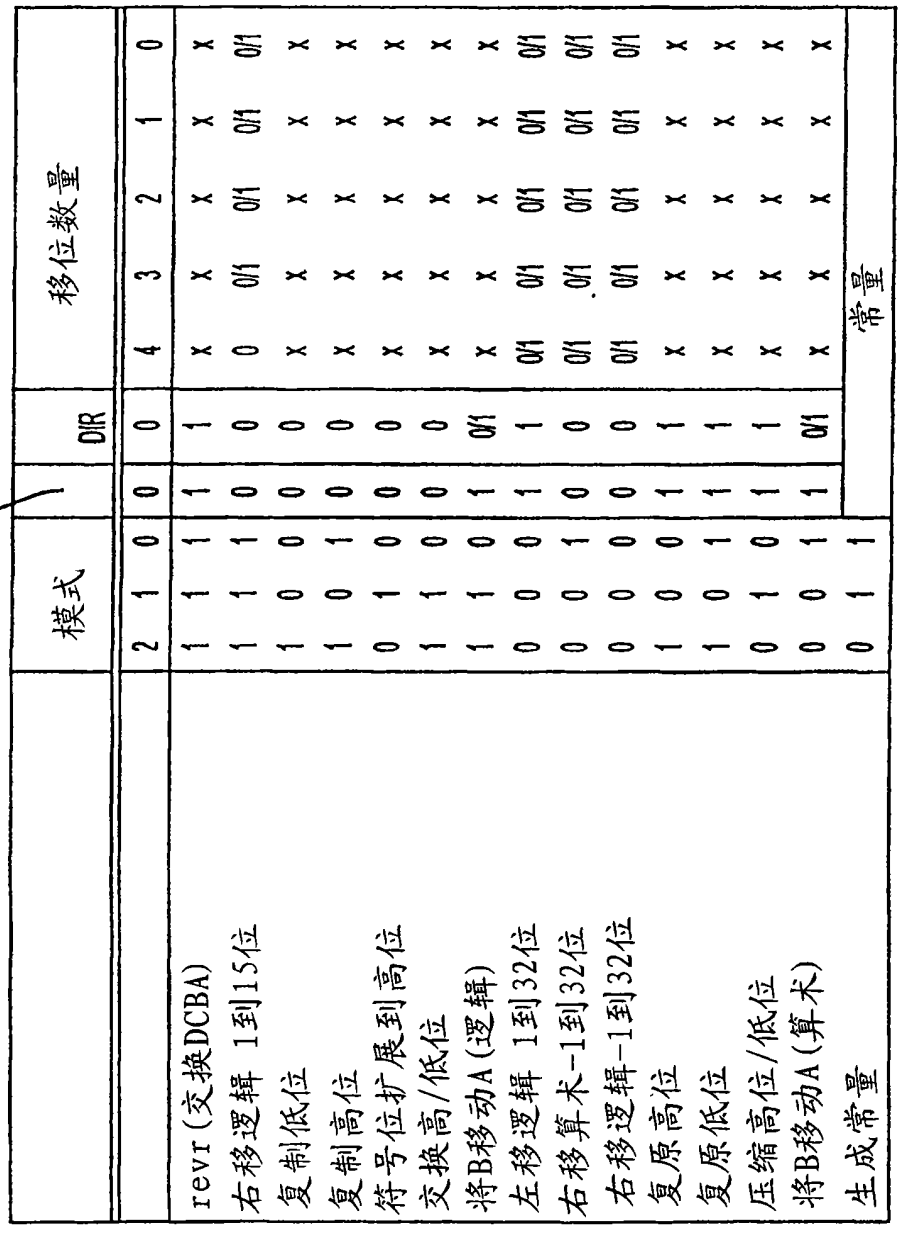
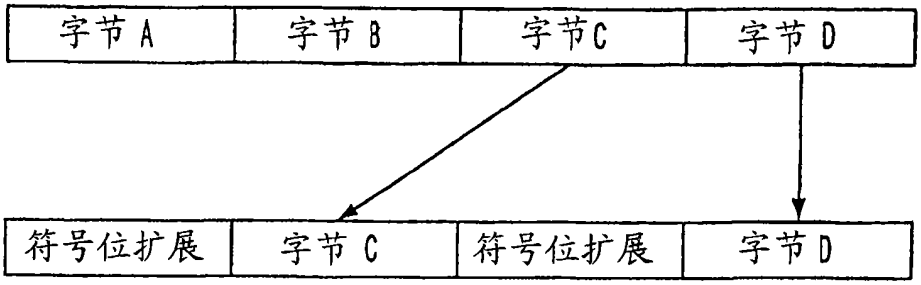
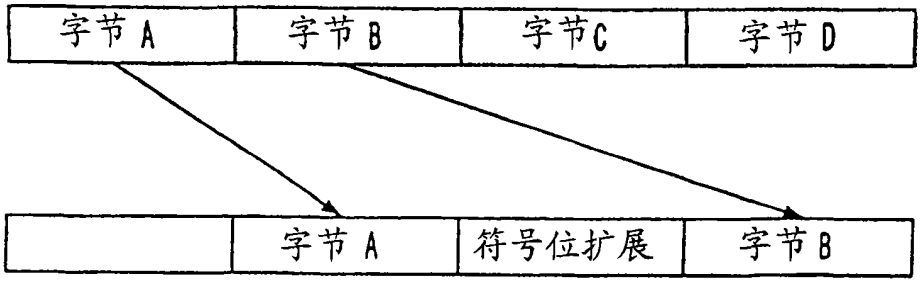

移位方向
模式
DIR
移位数量
2 1 0 0 4 3 2 1 0
revr(交换DCBA) 1 1 1 1 1 x x x x x
右移逻辑 1到15位 1 1 1 0 0 0 0/1 0/1 0/1 0/1
复制低位 1 0 0 0 0 x x x x x
复制高位 1 0 1 0 0 x x x x x
符号位扩展到高位 0 1 0 0 0 x x x x x
交换高/低位 1 1 0 0 0 x x x x x
将B移动A(逻辑) 1 1 0 1 0/1 x x x x x
左移逻辑 1到32位 0 0 0 1 1 0/1 0/1 0/1 0/1 0/1
右移算术-1到32位 0 0 1 0 0 0/1 0/1 0/1 0/1 0/1
右移逻辑-1到32位 0 0 0 0 0 0/1 0/1 0/1 0/1 0/1
复原高位 1 0 0 1 1 x x x x x
复原低位 1 0 1 1 1 x x x x x
压缩高位/低位 0 1 0 1 1 x x x x x
将B移动A(算术) 0 0 1 1 0/1 x x x x x
生成常量 0 1 1 常量

图 22

复原低位字



复原高位字



压缩高位低位字节

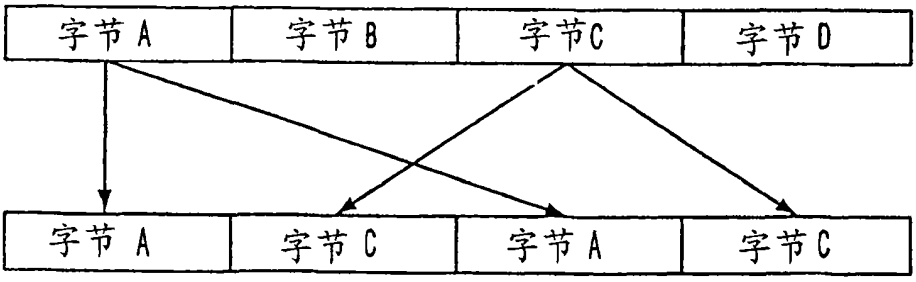


图 23

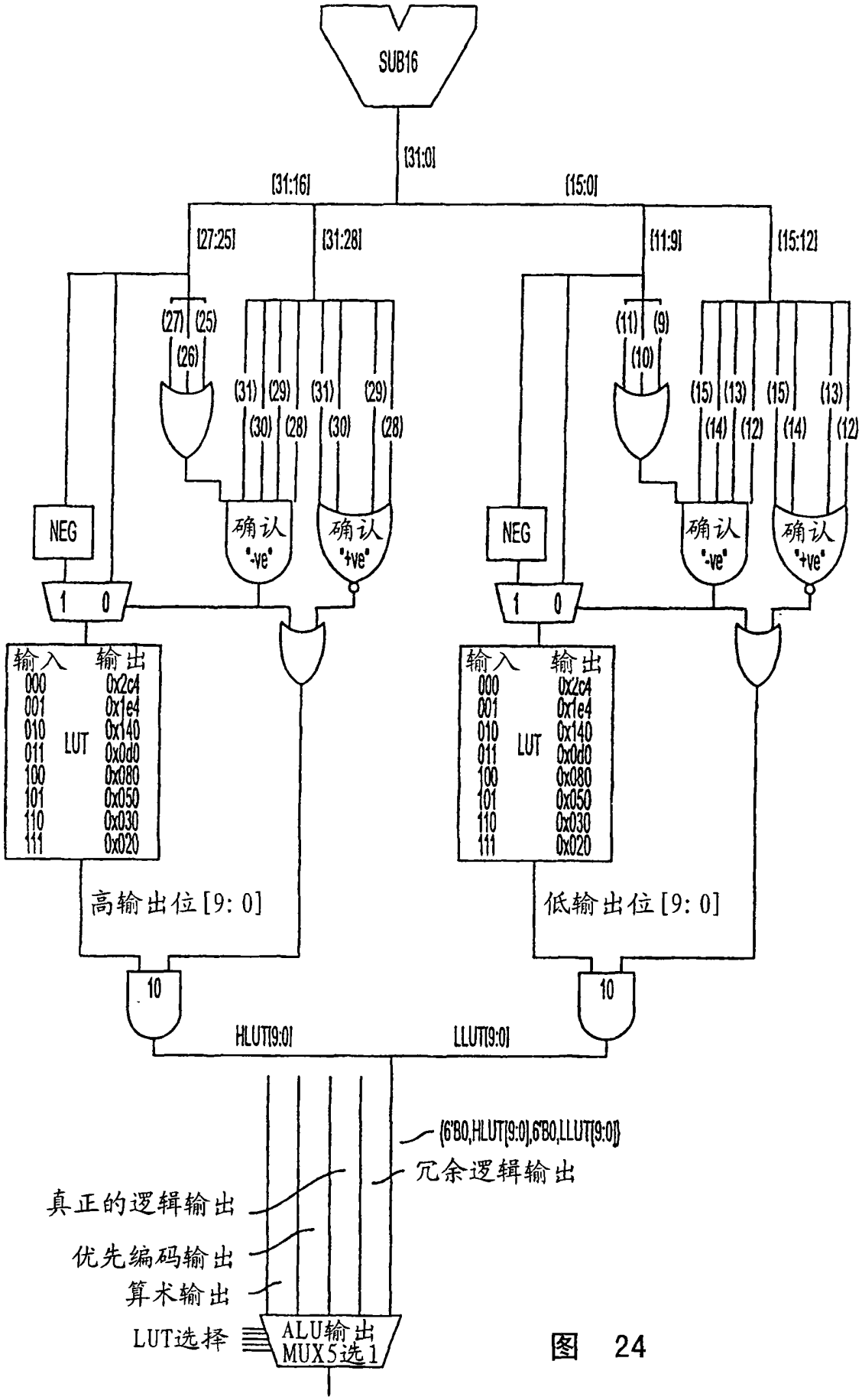

SUB16
[31:0]
[31:16]
[15:0]
[27:25]
[31:28]
[11:9]
[15:12]
(27)
(25)
(26)
(31)
(29)
(30)
(28)
(11)
(9)
(10)
(15)
(13)
(14)
(12)
NEG
确认 "-ve"
确认 "+ve"
1 0
输入 输出
000 0x2c4
001 0x1e4
010 0x140
011 0x0d0
100 0x080
101 0x050
110 0x030
111 0x020
LUT
高输出位[9:0]
低输出位[9:0]
10
HLUT[9:0]
LLUT[9:0]
{6'B0,HLUT[9:0],6'B0,LLUT[9:0]}
冗余逻辑输出
真正的逻辑输出
优先编码输出
算术输出
LUT选择
ALU输出 MUX5选1

图 24