[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 15/82 (2006.01)
G06F 9/44 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200480011371.4

[43] 公开日 2006 年 5 月 31 日

[11] 公开号 CN 1781092A

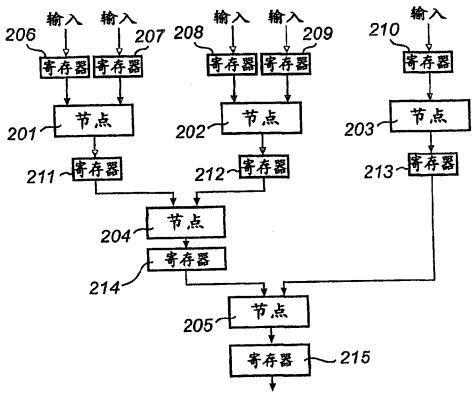
[22] 申请日 2004.3.17
[21] 申请号 200480011371.4
[30] 优先权
[32] 2003.3.17 [33] SE [31] 0300742-4
[86] 国际申请 PCT/SE2004/000394 2004.3.17
[87] 国际公布 WO2004/084086 英 2004.9.30
[85] 进入国家阶段日期 2005.10.27
[71] 申请人 米特里昂尼克斯股份公司
地址 瑞典隆德
[72] 发明人 史蒂芬·莫尔 庞图斯·伯格
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李德山
权利要求书 9 页 说明书 19 页 附图 6 页

[54] 发明名称

数据流机

[57] 摘要

公开了用于自高级源代码规格产生数字逻辑描述的方法。至少部分源代码规格被编译成包括具有至少一个输入或一个输出的功能节点和指示功能节点之间的互连的连接的多有向图表示。针对该图的每个功能节点和功能节点之间的每个连接定义硬件单元。最终，定义该图的每个功能节点的触发规则。



1.一种由图示产生用于实现数字逻辑电路的数字控制参数的方法，所述图示包括具有至少一个输入或至少一个输出的功能节点，和指示功能节点之间的互连的连接，其特征在于

针对该图的每个功能节点定义指定硬件单元的数字控制参数，

针对功能节点之间的每个连接定义指定硬件单元的数字控制参数，和

针对该图的每个功能节点定义指定硬件单元中的触发规则的数字控制参数。

2.如权利要求1所述的方法，其特征在于该图示是有向图。

3.如权利要求1或2所述的方法，其特征在于由高级源代码规格产生该图示。

4.如权利要求1-3所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定可以独立地并行访问的片上存储器单元的数字控制参数。

5.如权利要求1-4所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定数字寄存器的数字控制参数。

6.如权利要求1-4所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定至少一个触发器的数字控制参数。

7.如权利要求1-4所述的方法，其特征在于在于针对功能节点之间的至少一个连接定义指定至少一个锁存器的数字控制参数。

8.如前面任何权利要求所述的方法，其特征在于针对每个功能节点定义指定组合逻辑的数字控制参数。

9.如权利要求1-7所述的方法，其特征在于针对至少一个功能节点定义指定至少一个状态机的数字控制参数。

10.一种由图示产生用于实现数字逻辑电路的数字控制参数的设备，所述图示包括具有至少一个输入或至少一个输出的功能节点，和指示功能节点之间的互连的连接，其特征在于该设备适于

针对该图的每个功能节点定义指定硬件单元的数字控制参数，

针对功能节点之间的每个连接定义指定硬件单元的数字控制参数，和

针对该图的每个功能节点定义指定硬件单元中的触发规则的数字控制参数。

11.如权利要求10所述的设备，其特征在于该图示是有向图。

12.如权利要求10或 11所述的设备，其特征在于由高级源代码规格产生该图示。

13.如权利要求10-12所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定可以独立地并行访问的片上存储器单元的数字控制参数。

14.如权利要求10-13所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定数字寄存器的数字控制参数。

15.如权利要求10-13所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定至少一个触发器的数字控制参数。

16.如权利要求10-13所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定至少一个锁存器的数字控制参数。

17.如权利要求10-16所述的设备，其特征在于该设备适于针对每个功能节点定义指定组合逻辑的数字控制参数。

18.如权利要求10-16所述的设备，其特征在于该设备适于针对每个功能节点定义指定至少一个状态机的数字控制参数。

19.一种数据流机，其特征在于

执行数据变换的第一组硬件单元，

互连第一组硬件单元的第二组硬件单元，和

针对第一组硬件单元的每个硬件单元建立至少一个触发规则的电子电路。

20.如权利要求19所述的数据流机，其特征在于该图示是有向图。

21.如权利要求19或 20所述的数据流机，其特征在于第二组硬件单元的至少一个单元具有可以独立地并行访问的片上存储器单元的形式。

22.如权利要求19-21所述的数据流机，其特征在于第二组硬件单元的至少一个单元具有寄存器的形式。

23.如权利要求19-21所述的数据流机，其特征在于第二组硬件单元的至少一个单元具有触发器的形式。

24.如权利要求19-21所述的数据流机，其特征在于第二组硬件单元的至少一个单元具有锁存器的形式。

25.如权利要求19-24所述的数据流机，其特征在于第一组硬件单元具有组合逻辑的形式。

26.如权利要求19-24所述的数据流机，其特征在于第一组硬件单元中的至少一个单元具有至少一个状态机的形式。

27.如权利要求19-26所述的数据流机，其特征在于通过 ASIC，FPGA， CPLD或其它 PLD来实现数据流机。

28.一种直接可加载到具有数字计算机能力的电子设备的存储器中的计算机程序产品，包括当所述产品被所述电子设备运行时用于执行权利要求 1 - 9中的任何权利要求的步骤的软件代码部分。

29.如权利要求 28所述的计算机程序产品，其体现在计算机可读介质上。

30.一种由图示产生用于实现数字逻辑电路的数字控制参数的方法，所述图示包括功能节点和指示功能节点之间的互连的连接，所述功能节点包括至少一个输入或至少一个输出，其特征在于

针对至少第一和第二功能节点定义指定合并硬件单元的数字控制参数，所述数字控制参数指定合并硬件单元以执行第一和第二功能节点的功能，

针对由第一和第二功能节点的合并产生的硬件单元定义指定触发规则的数字控制参数。

31.如权利要求30所述的方法，其中所产生的数字控制参数指定第

一功能节点的至少一个输出和第二功能节点的至少一个输入之间的直接连接。

32.如权利要求 30或 31所述的方法，其特征在于针对合并硬件单元定义指定不同于第一和第二功能节点的触发规则的触发规则的数字控制参数。

33.一种用于由图示产生用于实现数字逻辑电路的数字控制参数的设备，所述图示包括功能节点和指示功能节点之间的互连的连接，所述功能节点包括至少一个输入或至少一个输出，其特征在于该设备适于：

针对至少第一和第二功能节点定义指定合并硬件单元的数字控制参数，所述数字控制参数指定合并硬件单元以执行第一和第二功能节点的功能，

针对由第一和第二功能节点的合并产生的硬件单元定义指定触发规则的数字控制参数。

34.如权利要求33所述的设备，其中该设备适于定义指定第一功能节点的至少一个输出和第二功能节点的至少一个输入之间的直接连接的数字控制参数。

35.如权利要求 33或 34所述的设备，其特征在于该设备适于针对合并硬件单元定义指定不同于第一和第二功能节点的触发规则的触发规则的数字控制参数。

36.一种直接可加载到具有数字计算机能力的电子设备的存储器中的计算机程序产品，包括当所述产品被所述电子设备运行时用于执行权利要求 30 - 32中的任何权利要求的步骤的软件代码部分。

37.如权利要求36所述的计算机程序产品，其体现在计算机可读介质上。

38.一种允许启动数据流机中的第一和第二互连硬件单元的方法，所述方法包括：

为第一硬件单元提供第一数字数据单元，其中第一硬件单元中第一数字数据单元的提供允许启动第一硬件单元，

从第一硬件单元传送第一数字数据单元到第二硬件单元，其中第二硬件单元中第一数字数据单元的接收允许启动第二硬件单元，并且第一数字数据单元从第一硬件单元的交出禁止启动第一硬件单元。

39.如权利要求38所述的方法，其中在第一数字数据单元的交出之后可以为第一硬件单元提供第二数字数据单元。

40.如权利要求38或 39所述的方法，其中在第一硬件单元中产生数字数据单元。

41.如权利要求38或 39所述的方法，其中数字数据单元在分立的硬件单元中产生并且被传送到第一硬件单元。

42.如权利要求38-41中任何一个所述的方法，其中数字数据单元被从第二硬件单元交出并且返回给第一硬件单元。

43. 一种包括第一和第二互连硬件单元的数据流机，其特征在于为第一硬件单元提供第一数字数据单元，并且第一硬件单元适于在所述第一数字数据单元存在于第一硬件单元中时被允许启动，并且第一硬件单元适于从第一硬件单元传送第一数字数据单元到第二硬件单元，其中第二硬件单元适于作为至少第一数字数据单元的接收的结果而被允许启动，并且第一硬件单元适于作为数字数据单元的交出的结果而被禁止启动。

44.如权利要求43所述的数据流机，其中第一硬件单元适于在向第二硬件单元传送第一数字数据单元之后被提供第二数字数据单元。

45.如权利要求43或 44所述的数据流机，其中第一硬件单元适于产生数字数据单元。

46.如权利要求43或 44所述的数据流机，其中第一硬件单元适于从分立硬件单元接收数字数据单元。

47.如权利要求43-46所述的数据流机，其特征在于通过 ASIC，FPGA， CPLD或其它 PLD来实现数据流机。

48. 一种保证数据流机中的数据完整性的方法，其中连接到至少第一和第二硬件单元的至少一个停止线被如此安排，使得在数据流机中提供数据路径，当停止信号在该停止线上活跃时，该数据路径被提

供用于暂停数据流，该数据流在处理周期期间在数据路径中从第一硬件单元前进到第二硬件单元，其特征在于

在第一片上存储器单元的第一输入处从第二硬件单元接收停止信号，

在第二片上存储器单元的第一输入处从第一硬件单元接收数据，

在第一和第二片上存储器单元中将接收数据和接收的停止信号缓冲至少一个处理周期，

在第一片上存储器单元的第一输出处提供缓冲的停止信号到第一硬件单元，并且

在第二片上存储器单元的第一输出处提供缓冲的数据到第二硬件单元。

49. 如权利要求**48**所述的方法，其中至少一个片上存储器单元是寄存器。

50. 一种保证数据流机中的数据完整性的设备，其中连接到至少第一和第二硬件单元的至少一个停止线被如此安排，使得在数据流机中提供数据路径，当停止信号在该停止线上活跃时，该数据路径被提供用于暂停数据流，该数据流在处理周期期间在数据路径中从第一硬件单元前进到第二硬件单元，其特征在于该设备适于

在第一片上存储器单元的第一输入处从第二硬件单元接收停止信号，

在第二片上存储器单元的第一输入处从第一硬件单元接收数据，

在第一和第二片上存储器单元中将接收数据和接收的停止信号缓冲至少一个处理周期，

在第一片上存储器单元的第一输出处提供缓冲的停止信号到第一硬件单元，并且

在第二片上存储器单元的第一输出处提供缓冲的数据到第二硬件单元。

51. 如权利要求**48**所述的方法，其中至少一个片上存储器单元是寄存器。

52.一种由图示产生用于实现数字逻辑电路的数字控制参数的方法，所述图示包括功能节点和指示功能节点之间的互连的连接，所述功能节点具有至少一个输入或至少一个输出，其特征在于

针对功能节点或功能节点之间的连接定义指定至少第一组硬件单元的数字控制参数，和

定义指定至少一个重定序硬件单元的数字控制参数，该至少一个重定序硬件单元对从至少一个第一组硬件单元发出的数据单元进行排序，使得数据单元按照与它们进入第一组硬件单元的顺序相同的顺序从第一组硬件单元发出。

53.如权利要求52所述的方法，其特征在于该图示是有向图。

54.如权利要求52或53所述的方法，其特征在于由高级源代码规格产生该图示。

55.如权利要求52-54中任何一个所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定可以独立地并行访问的片上存储器单元的数字控制参数。

56.如权利要求52-55中任何一个所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定数字寄存器的数字控制参数。

57.如权利要求52-55中任何一个所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定至少一个触发器的数字控制参数。

58.如权利要求52-55中任何一个所述的方法，其特征在于针对功能节点之间的至少一个连接定义指定至少一个锁存器的数字控制参数。

59.一种由图示产生用于实现数字逻辑电路的数字控制参数的方法，所述图示包括功能节点和指示功能节点之间的互连的连接，所述功能节点具有至少一个输入或至少一个输出，其特征在于该设备适于

针对功能节点或功能节点之间的连接定义指定至少第一组硬件单元的数字控制参数，和

定义指定至少一个重定序硬件单元的数字控制参数，该至少一个

重定序硬件单元对从至少一个第一组硬件单元发出的数据单元进行排序，使得数据单元按照与它们进入第一组硬件单元的顺序相同的顺序从第一组硬件单元发出。

60.如权利要求59所述的设备，其特征在于该图示是有向图。

61.如权利要求59或60所述的设备，其特征在于由高级源代码规格产生该图示。

62.如权利要求59-61中任何一个所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定可以独立地并行访问的片上存储器单元的数字控制参数。

63.如权利要求59-62所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定数字寄存器的数字控制参数。

64.如权利要求59-62所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定至少一个触发器的数字控制参数。

65.如权利要求59-62所述的设备，其特征在于该设备适于针对功能节点之间的至少一个连接定义指定至少一个锁存器的数字控制参数。

66.一种数据流机，其特征在于

执行数据变换的第一组硬件单元，和

至少一个重定序硬件单元，该至少一个重定序硬件单元对从至少一个第一组硬件单元发出的数据单元进行排序，使得数据单元按照与它们进入第一组硬件单元的顺序相同的顺序从第一组硬件单元发出。

67.如权利要求66所述的数据流机，其特征在于该图示是有向图。

68.如权利要求66或67所述的数据流机，其特征在于第二和第三组硬件单元的至少一个单元具有可以独立地并行访问的片上存储器单元的形式。

69.如权利要求66-68所述的数据流机，其特征在于第二和第三组硬件单元的至少一个单元具有寄存器的形式。

70.如权利要求66-68所述的数据流机，其特征在于第二和第三组

硬件单元的至少一个单元具有触发器的形式。

71.如权利要求66-68所述的数据流机，其特征在于第二和第三组硬件单元的至少一个单元具有锁存器的形式。

72.如权利要求66-71所述的数据流机，其特征在于通过 ASIC，FPGA， CPLD或其它 PLD来实现数据流机。

73.一种直接可加载到具有数字计算机能力的电子设备的存储器中的计算机程序产品，包括当所述产品被所述电子设备运行时用于执行权利要求 52 - 58中的任何权利要求的步骤的软件代码部分。

74.如权利要求73所述的计算机程序产品，其体现在计算机可读介质上。

75.如权利要求30到32中任何一个所述的方法，其中由高级源代码规格产生该图示。

76.如权利要求30到35中任何一个所述的设备，其中由高级源代码规格产生该图示。

77. 如权利要求 43-46 所述的数据流机，其中数字数据单元被从第二硬件单元交出并且返回给第一硬件单元。

数据流机

技术领域

本发明一般涉及数据处理方法和设备，尤其涉及用于在数字硬件中通过使用精细粒度并发和大流水线深度的数据流机执行高速数据处理的方法和设备。

背景技术

近年来，已经使用许多不同的易用编程语言解决方案来进行硬件描述，以提供数字电路设计的快速和容易的方式。当对数据流机进行编程时，可以使用不同于硬件描述语言的语言。原则上，用于在数据流机上执行特定任务的算法描述只须包括描述本身，而要直接在集成电路中执行的算法描述必须包括该算法在硬件中的特定实现的许多细节。例如，硬件描述必须包含有关寄存器的布局的信息，以便规定最优时钟频率，使用哪些乘法器等等。

许多年来，数据流机被认为是并行计算的良好模型，因此已经进行许多尝试来设计高效的数据流机。出于各种原因，较早的数据流机设计尝试与其它可用并行计算技术相比在计算性能方面产生不良结果。

当翻译程序源代码时，当前可用的多数编译器使用数据流分析和数据流描述 (被称为数据流图或 DFG)以便优化所编译程序的性能。对算法执行的数据流分析产生数据流图。数据流图说明算法内存在的数据依赖。更具体地，数据流图通常包括指示算法对所处理的数据执行的特定操作的节点，和指示图中节点之间的互连的弧。因此，数据流图是特定算法的抽象描述，并且被用于分析算法。另一方面，数据流机是可以基于数据流图实际执行该算法的计算机器。

与例如冯诺依曼体系结构 (个人计算机中的普通处理器是冯诺依曼体系结构的例子)的控制流设备相比较，数据流机以根本不同的方

式操作。在数据流机中，程序是数据流图，而不是要由处理器执行的操作系列。数据被组织成称为标记(tokens)、位于数据流图的弧上的包。标记能够包含将由通过弧形连接的节点操作的任何数据结构，类似于位，浮点数，数组等等。根据数据流机的类型，每个弧形可以至多拥有单个标记 (静态数据流机)，固定数量的标记 (同步数据流机)，或不定数量的标记 (动态数据流机)。

数据流机中的节点等待标记出现在足够数量的输入弧上，使得可以执行其操作，因此它们消费那些标记并且在其输出弧上产生新标记。例如：执行 2个标记的加法的节点将进行等待，直到标记已经出现在其两个输入上，消费这 2个标记并且接着产生作为其输出弧形上的新标记的结果 (在这种情况下为输入标记的数据的和)。

不是象在 CPU中进行的那样根据条件分支选择不同操作以对数据进行操作，数据流机根据条件分支将数据引导到不同节点。于是，数据流机具有可以有选择地在特定输出上产生标记的节点 (称作开关节点)以及可以有选择地消费特定输入上的标记的节点 (称作合并节点)。公共数据流操作节点的另一个例子是有选择地从数据流中清除标记的门节点。可以存在许多其它数据流操作节点。

图中的每个节点可以潜在地独立于该图中的所有其它节点地执行其操作。一旦一个节点在其相关输入弧上具有数据，并且存在空间以在其相关输出弧上产生结果，则该节点可以执行其操作 (称为触发(firing))。无论其它节点是否能够触发，节点均会触发。 于是，不存在例如控制流设备中的执行节点操作的特定顺序；数据流图中的操作执行顺序是无关的。执行顺序可以例如是可以触发的所有节点的同时执行。

如上所述，根据其设计，数据流机通常被分成 3个不同类别：静态数据流机，动态数据流机，和同步数据流机。

在静态数据流机中，相应数据流图中的每个弧在每个时刻可以只拥有单个标记。

在动态数据流机中，每个弧可以拥有等待接收节点准备好接受它

们的不定数量的标记。这允许构造当设计数据流机时递归深度未知的递归过程。这种过程可以反置递归中处理的数据。当在完成递归之后执行计算时，这会导致标记的错误匹配。

通过在协议中增加指示每个标记的序号的标记(marker)，可以处理上述情况。连续监视递归内的标记的序号，并且当一个标记退出递归时，它不被允许继续前进，只要它不能匹配递归外的标记。

在递归不是尾递归的情况下，在每次递归调用时，必须以和在使用普通 (冯诺依曼)处理器执行递归时将上下文存储在堆栈中相同的方式将上下文存储在缓冲区中。最终，动态数据流机可以并行执行数据相关递归。

在没有能力使标记在接收节点准备其自身的同时在弧上等待的情况下，同步数据流机能够进行操作。取而代之，事先计算每个节点的标记的产生和消费之间的关系。通过这个信息，能够针对可以同时位于弧上的标记的数量确定如何放置节点以及向弧分配大小。于是能够保证每个节点产生与后续节点消费的标记一样多的标记。于是可以这样设计系统，使得每个节点始终可以产生数据，因为后续节点将总是消费数据。缺点是该构造中可能不存在例如数据相关递归的不确定延迟。

数据流机多数通常通过传统 CPU中运行的计算机程序来付诸实践。通常使用计算机集群，或某种印制电路板上的 CPU阵列。使用数据流机的主要目的曾经是利用其并行性来构造试验性超级计算机。已经进行许多尝试来直接用硬件构造数据流机。这已经通过用专用集成电路 (ASIC)创建多个处理器来实现。与在电路板上使用处理器相比，这个解决方案的主要优点是相同 ASIC上的处理器之间的更高通信速率。到目前为止，在使用数据流机进行计算方面没有任何尝试取得商业成功。

现场可编程门阵列 (FPGA)和其它可编程逻辑器件 (PLD)也可以被用于硬件构造。FPGA是可在运行时刻重新配置的硅芯片。它们基于小型随机访问存储器，通常为静态随机存取存储器 (SRAM)的阵

列。每个 SRAM拥有用于布尔函数的查找表，于是允许 FPGA执行任何逻辑操作。FPGA也类似地拥有允许信号从 SRAM行进到 SRAM的可配置路由资源。

通过向 SRAM分配硅芯片的逻辑操作并且配置路由资源，可以实现任何足够小型以便安装在 FPGA表面上的硬件构造。与 ASIC相比， FPGA能够在相同量的硅表面上实现更少的逻辑操作。FPGA的优点是能够简单地通过向 SRAM查找表输入新值并且改变路由来将其转换为任何其他硬件构造。FPGA能够被视为能够接受任何硬件构造的空硅表面，而且能够以非常短的时间 (小于 100毫秒)转变为任何其他硬件构造。

其它公共 PLD可以被熔断连接，于是被永久配置。熔断连接 PLD优于 ASIC的主要优点是容易构造。为制造 ASIC，需要非常昂贵和复杂的工艺。相反，能够通过简单工具在几分钟内构造 PLD。存在若干用于 PLD的进化技术，其可以克服熔断连接 PLD和 FPGA的某些缺点。

一般地，为了对 FPGA进行编程，必须使用由 FPGA的厂商提供的放置和接通工具(place - and - route tools)。放置和接通软件通常从合成软件接受连线表或从硬件描述语言 (HDL)接受其直接合成的源代码。放置和接通软件于是在描述文件中输出用于在编程单元中对FPGA进行编程的数字控制参数。类似技术被用于其它 PLD。

当设计集成电路时，通常将电路设计为状态机，因为它们提供简化硬件的构造的框架。当实现其中数据将以取决于先前计算的各种模式流过逻辑操作的复杂数据流时，状态机尤其有用。

状态机也允许重用硬件单元，于是优化电路的物理尺寸。这允许以更低的成本制造集成电路。

前面使用专用硬件的数据流机的构造基于彼此连接状态机或专用 CPU (作为状态机的特例)。这些通常与专用路由逻辑，在某些情况下为专用存储器连接。更具体地，在数据流机的较早设计中，状态机被用于仿真数据流机的行为。此外，较早的数据流机通常也具有动态

数据流机的形式，所以通常使用专用标记匹配和重定序部件。

US 5,021,947公开了如上所述，通过在三维结构中安排多达 512 个处理单元 (PE)来仿真数据流机的多处理系统。每个 PE用其自身的用于程序和数据存储的本地存储器构成一个完整的 VLSI实现的计算机。以包含要处理的数据以及标识目的 PE的地址和标识 PE内的操作装置的地址的数据分组的形式在不同 PE之间传送数据。此外，互连 PE的通信网络被设计成可靠的，其中自动对错乱的消息进行重试，具有分布式总线仲裁，可选路径分组路由等等。 另外，计算机的模块化性质允许增加更多处理单元以满足各种吞吐率和可靠性需求。

因此，根据 US 5,021,947的仿真数据流机的结构非常复杂，并且没有完全利用数据流图中已经存在的数据流结构。并且，机器中来回传送的分组的监视意味着增加更多额外的逻辑电路。

Verdoscia等人的文献" Conditional and Iterative Structures Using a Homogeneous Data Flow Graph Model "也公开了包括获得同构数据流的一组处理器的数据流机。在公开于相同作者的文献" Actor Hardware Design for Static Dataflow Model "中的称作" Alfa "的设备中实现了数据流机。然而这些文献公开的机器没有针对较早建立的数据流图的结构进行优化，即在建立数据流图之后执行许多步骤，这些步骤使得机器适于使用计算机形式的硬件单元来实现。因此，这些文献公开的机器利于通过一组相同硬件单元 (计算机)的同质数据流，但是没有公开如何以在计算效率方面最优的方式通过硬件实现数据流图。

通过建立具有大量数据流机形式的处理器的超级计算机，希望实现高度的并行性。已经进行尝试，其中处理器由许多 CPU或许多 ASIC构成，均包括许多状态机。由于较早数据流机的设计已经包含在 ASIC中使用状态机 (通常具有处理器的形式)，在类似于 FPGA的可编程逻辑设备中实现数据流机的最直接方法也将是使用状态机。所有先前已知的数据流机的一般特性是所建立的数据流图的节点不对应于最终硬件实现中的特定硬件单元 (公知为功能单元， FU)。相反，在

特定时刻正好可用的硬件单元被用于执行由数据流图中受影响的节点指定的计算。如果数据流图中的特定节点要被多于一次地执行，则每当执行该节点时可以使用不同功能单元。

此外，前面的数据流机已经全部通过使用状态机或处理器执行数据流机的功能来实现。每个状态机能够执行数据流图中的任何节点的功能。这需要允许每个节点在任何功能单元中执行。由于每个状态机能够执行任何节点的功能，与当前执行节点无关的任何其他节点所需的硬件将会休眠。应当注意，状态机 (有时具有用于标记操作的支持硬件)是数据流机其自身的实现。情况并不是这样，即数据流机由某些其它装置实现，并且正好将状态机包含在其功能节点中。

当前使用的多数编程语言是所谓的命令语言，例如 Java，Fortran 和 Basic 语言。在不放松并行性的情况下，这些语言几乎不可能或至少非常难以改写为数据流。

相反，使用功能语言而不是命令语言可简化数据流机的设计。功能语言的特征在于它们表现出称作引用透明的特性。这意味着只有即时分量表达式的含义或值在确定较大复合表达式的含义时有意义。由于当且仅当它们具有相同含义时才是相等的，因此引用透明意味着在较大表达式的上下文中能够互换相等子表达式以提供相等结果。

如果操作的执行具有除了提供输出数据之外的影响 (例如在操作执行期间显示器上的读数)，它不是引用透明的，因为执行操作的结果与不执行操作的结果不同。针对以引用透明语言编写的程序的全部通信被称作副作用(side-effect) (例如存储器访问，读出，等等)。

WO 0159593公开了将算法的高级软件描述编译成数字硬件实现。通过使用编译工具来解释编程语言的语义，该编译工具分析软件描述以产生控制和数据流图。于是，这个图是用于优化，变换和注释的中间格式。所产生的图接着被翻译成硬件实现的寄存器传送层次或连线表层次描述。分立的控制路径被用于确定流图中的节点应何时向相邻节点传送数据。通过分割控制路径和数据路径可以实现并行处理。通过使用控制路径，可以实现"波前(wavefront)处理"，这意味着通过

实际硬件实现的数据流作为由控制路径控制的波前。

控制路径的使用意味着只有该硬件的部分可以在执行数据处理的同时使用。其余电路等待第一波前通过流图，使得控制路径可以启动新的波前。

US 6145073公开了预先设计和验证的数据驱动的硬件核心，其被装配为在单芯片上产生大型系统。利用 1位就绪信号和 1位请求信号通过专用连接在核心之间同步传送标记。就绪请求信号握手对于标记传送是必要和足够的。并且，每个连接的核心必须至少具有有限状态机的复杂度。最终，没有一般触发机制的构思，所以没有数据流的条件重定向能够执行。于是，没有数据流机能够用这个系统来建立。更确切地，用于核心之间数据交换的协议聚焦于保持核心内的流水线的完整。

Mihai Budiu 的文献" Application - Specific Hardware : Computing Without CPUs "公开了组合可重新配置硬件和编译器技术以产生专用硬件的通用计算体系结构。每个静态程序指令由专用硬件实现来表示。程序被分解成称作分相抽象机 (SAM)的较小片段，分相抽象机在硬件中被合成为状态机，并且通过互连网络来组合。在程序的执行期间， SAM能够处于以下3个状态之一：不活动，活动或被动。在不同 SAM之间传递标记，标记允许 SAM开始执行。这意味着每次只有少数 SAM可以执行实际数据处理，其余 SAM等待标记允许执行。通过这个过程实现低功耗，但是同时减少了计算容量。

发明内容

本发明寻求提供一种改进数据处理系统的性能的方法。

更具体地，本发明的一个目的是通过用硬件实现数据流机来增加计算能力，其中获得巨大的并行性。

本发明的另一个目的是改进可以同时使用可用硬件资源，即较大部分可用逻辑电路 (门，开关等等)的利用率。

这个目的已经通过由高级源代码规格产生数字逻辑描述的方法实现，其中至少部分源代码规格被编译成多有向图表示(multiple

directed graph representation)，该多有向图表示包括具有至少一个输入或一个输出的功能节点，和指示功能节点之间的互连的连接。此外，针对该图的每个功能节点定义硬件单元，其中硬件单元代表由功能节点定义的功能。针对功能节点之间的每个连接定义附加硬件单元，其中附加硬件单元代表从第一功能节点到第二功能节点的数据传送。最终，定义该图的每个功能节点的触发规则，其中所述触发规则定义功能节点在其输出处提供数据和消费其输入处的数据的条件。

并且，本发明的一个目的是克服对计算效率的限制，该限制由于使用允许不同功能单元之间的数据流的专用控制路径而出现在现有技术数据流机中。另外，与现有技术解决方案相比，作为数据流机中的高效数据存储的结果，根据本发明的硬件实现允许提高计算能力，而无需与外部存储器进行大量通信。

因此，本发明以高效方式通过硬件实现了由数据流图描述的功能，而无需专用互连的 CPU或高级数据交换协议。本发明充分利用数据流机和 RTL (寄存器传送层次)逻辑之间语义的相似性，其中使用组合逻辑而不是 CPU，并且使用硬件寄存器而不是 RAM (随机访问存储器)，底板或以太网。

本发明的另一个目的是允许由高级编程语言描述设计硅硬件。高级编程语言是侧重于其自身的算法的描述，而不是特定类型硬件的算法的实现的编程语言。通过高级编程语言和由通过该语言编写的程序自动设计集成电路描述的能力，可以使用软件工程技术设计集成电路。对于能够低成本或无成本地用许多不同硬件设计重新配置的 FPGA和其它重新配置 PLD，这尤其有利。

除由许多不同的容易产生的硬件设计带来的益处之外，FPGA和其它PLD能够因通过高级语言进行的自动硬件描述设计而具有效率方面的益处。如果系统能够挖掘出大量并行性，则能够向尽可能大的PLD部分填充有意义操作，从而提供非常高的性能。这与通常侧重于产生尽可能小的设计的传统硬件设计形成对比。

通过阅读以下对优选实施例的详细公开可以更加清楚地理解本

发明的其它目的，特征和优点。

附图说明

下面参照附图描述本发明的优选实施例，其中：

图1a是说明本来已知的第一数据流图的示意图，

图1b是说明本来已知的第二数据流图的示意图，

图2说明了本发明的第一实施例。

图3图解了本发明的第二实施例，其中不同数据路径的长度得到均衡。

图4a是根据本发明第三实施例的节点的详细示意图。

图4b说明了用于建立触发规则的逻辑电路的例子。

图4c相应说明了数据流机中节点之间的寄存器中使用的逻辑电路的例子。

图5a图解了本发明的第三实施例，其中不同数据路径的长度通过节点合并得到均衡。

图5b是有关图5a中的 2个节点的合并的更加详细的图解。

图6说明了根据本发明的停止截断器的实施例。

具体实施方式

一般地，通过数据流分析将源码程序变换成数据流图。执行数据流分析的简单方法是：在程序的所有输出处开始。找到每个输出的即时源。如果它是操作，则用节点替换该操作，并且用弧将它连接到输出。如果源是变量，则用弧替换该变量并且将其连接到输出。针对缺少完全指定的输入的所有弧和节点进行重复。

图1a说明了本来已知的数据流图。出于清楚的原因，在本文中，术语节点被用来指示数据流图中的功能节点。该图中示出 3个处理层次：顶端节点 101,102， 103在其输入处从一或多个源接收输入数据，这些数据将在其流过该图时得到处理。顶端节点执行的实际数学，逻辑或过程性功能是特定于每个实现的，因为它取决于数据流图所源于的源代码。例如，第一节点 101可以对来自 2个输入的数据执行简单加法，第二节点 102可以从第二输入处接收的数据中减去第一输入处

接收的数据，第三节点 103可以例如用其输入处接收的 2个数据执行定点乘法。每个节点的输入数量，每个节点中执行的实际处理等等对于不同实现当然可以是不同的，并且不限于上述例子。例如，节点可以执行更加复杂的计算，甚至访问外部存储器，这将在下面描述。

数据从第一节点层次流向第二节点层次，其中在这种情况下来自节点 101和 102的数据被从节点 101和 102的输出传送到节点 104的输入。 根据上述讨论，节点 104根据其输入处接收的信息执行特定任务。

在第二层次的处理之后，数据被从节点 104的输出传送到节点 105的第一输入，该节点位于第三层次。通过图1可以发现，在节点 105的第二输入处接收来自层次 1的节点 103的输出的数据。节点 103和 105之间没有存在第二层次节点的事实意味着当在节点 105的第一输入节点处可得到数据之前将在节点 105的第二输入处可得到来自节点 103的数据 (假定每个节点处的组合延迟相等)。为了高效地处理这种情况，将为每个节点提供触发规则，触发规则定义了使节点在其输出处提供数据的条件。

更具体地，触发规则是控制数据流图中的数据流的机制。通过使用触发规则，在根据节点的功能变换数据的同时，数据被从输入传送到节点的输出。仅当在节点的输入处真实存在可用数据时，才可以进行来自该输入的数据的消费。相应地，如果不存在来自前一计算的阻塞路径的数据 (即后续节点一定消费了前面的数据项)，则只可以在输出处产生数据。然而在某些情况下，可以在输出处产生数据，即使旧数据阻塞路径；输出处的旧数据接着将被新数据替换。

一般触发规则的规格通常包括:

1)针对节点的每个输入、为了使该节点消费输入数据的条件，

2)针对节点的每个输出、为了使该节点在输出处产生数据的条件，和

3)执行节点的功能的条件。

条件通常取决于输入数据的值，输入或输出处有效数据的存在，

应用于输入的功能的结果或功能的状态，但是原则上可以取决于系统可用的任何数据。

通过针对系统的节点 101 - 105建立一般触发规则，能够控制各种程序而无需专用控制路径。然而，通过触发规则，针对某些特殊情况能够实现控制流。另一个特例是没有触发规则的系统，其中所有节点 101 - 105仅当在节点 101 - 105的所有输入处可得到数据时进行操作。

通过合并节点能够提供触发规则的功能的特定例子。通过这种节点，能够控制数据流，而无需控制流。合并节点通常具有 2个数据输入，其中将选择来自其中一个数据输入的数据。此外，它也具有用于选择从其获取数据的数据输入的控制输入。最终，它具有一个数据输出，在该数据输出处传送所选的输入数据值。

例如，假定节点具有 2个输入 T和 F。 在输入 C上接收控制节点的条件，在输出 R上提供结果。 下面的触发规则将在节点的输出处产生数据，即使只存在在一个输入处可得到的数据。在这种情况下，例如如果 C = 1，则没有数据需要存在于输入 F处。 即，消费节点的输入处的数据的条件是:

(C=1 AND T=x) OR (C=0 AND F=x)

其中 x表示存在有效值。

此外，提供节点的输出处的数据的条件是:

(C=1 AND T=x) OR (C=0 AND F=x)

并且该节点的功能是:

R = IF (C==1) T ELSE F

另一种用于控制数据流的节点是开关。开关节点通常具有 2个输出 T和 F，一个数据输入 D和一个控制输入 C。 当在数据输入和控制输入处可得到数据时，节点将在其输出之一处提供数据。消费来自输入的数据的条件是:

C=x AND D=x

并且在输出处提供数据的条件是:

T: C=1 AND D=x

F: C=0 AND D=x

并且节点的功能是:

T = IF(C==1) D

F = IF(C==0) D

图 1b说明了用于控制数据流机中的数据流的合并和开关节点的使用。在这种情况下，数据流机根据以下函数计算s的值:

$$s = \sum_{i=1}^{n} f(x_i)$$

遵循上述推论，能够针对所有类型的可能节点建立触发规则，例如: 真门(True - gates)(一个数据输入 D，一个控制输入 C，一个输出 R和函数 R = IF (C == 1) D); 不确定优先合并 (2个数据输入 D1和 D2 ，一个输出 R和函数R = IF (D1) D1 ELSE IF (D2) D2); 加法 (2个数据输入D1和D2，一个输出R和函数R = D1 + D2); 复制(一个数据输入D，一个控制输入C，一个输出R和函数R = D); 和布尔流 (Boolstream) (没有输入，一个输出R和函数:

R = IF (state==n) 设置 state=0, 返回 1

ELSE 递增state, 返回 0

然而，独立于节点的功能地，在处理其输入处的数据之后，节点105在其输出处提供数据处理的最终值。在这种情况下， 5个输入处的数据在单个输出处产生数据。

当严密检查数据流机的语义时，能够观察到，那些语义非常类似于数字电路操作的方式，尤其是在寄存器传送层次 (RTL)上。在数据流机中，数据位于弧上，并且通过对数据执行某种操作的功能节点被从一个弧传递到另一个弧。在数字电路中，数据驻留在寄存器中，并且通过对数据执行某种功能的组合逻辑在寄存器之间传递。由于这种紧密相似性存在于数据流机的语义和数字电路的操作之间，因此能够直接在数字电路中实现数据流机。这意味着数据通过数据流机的传播能够在数字电路中实现，而无需类似于状态机的模拟设备执行数据流机的操作。取而代之，通过用组合逻辑替换节点并且用可以独立地并

行访问的寄存器或其它快速存储器元件替换弧，能够直接实现数据流机。

其优点主要是执行速度。这种实现会能够利用比通过处理器或其它状态机的实现水平更高的并行性。它对流水线更加容易，并且并行性的水平能够具有更加精细的粒度。应当注意，避免使用状态机实现数据流机本身仍然允许数据流机的节点包含状态机。

本发明一个可选描述将是，特殊寄存器节点被插入在数据流图的功能节点之间，并且边(edge)被实现成简单的连线。为了清楚，我们以节点作为组合逻辑并且边作为寄存器，而不是使用功能节点，寄存器节点和边的方式描述本发明。

图2说明了本发明的第一简单实施例。更具体地，该图图解了图1的数据流图的硬件实现。 图1的功能节点 101-105已经被节点 201-205替代，节点 201-205执行图1的数据流图中定义的数学或逻辑函数。 这种函数能够通过组合逻辑执行，但也能够例如通过状态机或某种流水线设备来执行。

在图2中，例如寄存器 206-215或触发器的连线和快速并行数据存储硬件已经替代图1的不同节点之间的连接。 在节点 201-205的输出处提供的数据被存储在寄存器 206-215中，以便即时或后续传送到另一个节点 201-205。从图中可以看出，寄存器213使得能够存储来自节点203的输出值，同时在节点204中处理来自节点201和202的数据。如果在不同节点 201-205之间不存在可用的寄存器 206-215，由由于相同路径中前面节点的不同组合延迟，某些节点的输入处的数据可能不稳定 (改变值)。

例如，假定已经在节点 201-203的输入处提供 (通过寄存器 206-210)第一组数据。在节点中的处理之后，将在节点 201-203的输出处可得到数据。节点201和202接着将提供数据到节点204，同时节点203将提供数据到节点 205。由于节点205也从节点204接收数据，必须在数据被传送到节点205之前在节点204中处理该数据。如果在数据传播通过节点204之前在节点 201-203的输入处提供新数据，则节点203的

输出也许已改变。因此，节点205的输入处的数据将不再正确，即与节点205提供的数据相比，节点204提供的数据来自于较早的时刻。

上述推论是简化的推论。实际上，需要高级时钟模式，通信协议，附加节点/寄存器或附加逻辑电路以保证提供给不同节点的数据是正确的。图3示出了对问题的最直接的解决方案，其中附加节点316及其相关寄存器317已经插入在数据路径中。节点316执行 NOP (无操作)，因此不改变其输入处提供的数据。通过插入节点316，在图的每个数据路径中获得相同长度。效果是 203和205之间的弧现在可以拥有 2个单元。

图4a图解了另一个解决方案，其中为每个节点 401提供附加信号线，用于在每个时刻提供正确数据。第一附加线传递"有效(valid)"信号402，其指示前面节点在其输出处具有稳定数据。类似地，当节点401的输出处的数据稳定时，节点401向数据路径中的后续节点提供"有效"信号403。通过这个过程，每个节点能够确定其输入处的数据的状态。

此外，第二附加线传递"停止(stall)"信号404，"停止"信号404向前一节点指示当前节点 401没有准备在其输入处接收任何附加数据。类似地，节点401也从数据路径中的后续节点接收"停止"线405。通过使用停止线，能够临时停止特定路径中的数据流。如果在某些时刻的节点执行具有不确定延迟的费时数据处理，例如循环或存储器访问，那么这是重要的。停止信号的使用仅仅是本发明的一个实施例。根据选择的协议，能够使用若干其它信号。例子包含"数据消费"，"就绪接收"，"确认"或"不确认"信号，和基于脉冲或转变的信号，而不是高或低信号。其它信号模式也是可以的。"有效"信号的使用使得能够表示弧上数据的存在或不存在。于是，不仅能够构造同步数据流机，而且能够构造静态和动态数据流机。"有效"信号不必被实现为专用信号线，它能够以若干其它方式实现，类似于选择特殊数据值来表示"空(null)"值。至于停止信号，存在许多其它可能的信号模式。为了清楚，本文的其余部分将只引用停止和有效信号。将本发明的功能扩展到其它信号模式是简单的。

由于特定停止信号的存在，能够实现更高效率。停止信号使得节点能够知道即使该时刻下面的弧是完整的，它会能够在下一时钟周期接受输出标记。在没有停止信号的情况下，节点必须进行等待，直到在其能够触发之前下面的弧上没有有效数据。这意味着在至少每个其它周期弧将为空，从而失去效率。

图4b图解了用于针对节点401产生有效 402,403和停止 404，405信号的逻辑电路的例子。这个图中示出的电路被使用在当所有输入上可得到数据时触发的节点中。通常，触发规则可能更加复杂，并且必须根据单个节点401的功能来建立。

图4c相应说明了数据流机中节点之间的寄存器406中使用的逻辑电路的例子。这个电路保证如果目的节点尚未准备接受数据，则寄存器将保留其数据，并且将此情况通知源节点。如果寄存器为空，或如果目的节点将要接受寄存器的当前内容，它也将接受新数据。在这个图中，为了清楚的原因只图解一个数据输入 407和一个数据输出 408。然而，需要强调的是输入和输出的实际数量取决于系统的总线宽度(即标记有多少位宽)。

在复杂数据流机的情况下，停止线与信号传播速度相比可能变得非常长。这个可以导致停止信号不到达路径中需要停止的每个节点，导致数据丢失 (即还没有处理的数据被新数据改写)。

有 2个解决此问题的常见方法。非常谨慎地均衡停止信号传播路径以保证它及时到达所有目标寄存器。可选地，在可停止的块之后放置 FIFO缓冲区，从而完全避免在该块内使用停止信号。当流水线数据从流水线流出时，FIFO被用来收集流水线数据。对于较大的流水线块，前一解决方案非常难以实现并且耗时。后者则需要较大的缓冲区以便能够保存该块内可能存在的整个数据集合。

应对这种有限信号传播速度的更好方式是利用称作"停止截断器(stall cutter)"的如图6图解的特征。 停止截断器基本上是从后续节点接收停止线并且将其延迟一个周期的寄存器。这在该点切断了停止信号的组合长度。

当停止截断器接收有效停止信号时，它在一个处理周期期间缓冲来自前一节点的数据，并且同时将停止信号延迟相同的量。通过延迟停止信号并缓冲输入数据，保证即使使用非常长的停止线也没有数据被丢失。停止截断器能够大大简化数据循环的实现，尤其是流水线数据循环。在这种情况下，用于控制数据流的协议的许多变化将需要停止信号采取与通过循环的数据相同，通常相反的路径。这将产生停止信号的组合循环。通过在循环内放置停止截断器，能够避免这种组合循环，从而允许使用否则将难以或不可能实现的许多协议。

最终，从数据流机中数据传播的角度观察，停止截断器是透明的。这意味着能够根据需要以自动方式增加停止截断器。

图5a图解了本发明的另一个实施例，其中图中数据路径已通过节点合并得到均衡。对于使用全局时钟信号的设计，通过最慢的处理单元确定最高可能时钟频率。这意味着有能力在高频处进行操作的每个处理单元被约束为在由最慢单元设置的频率上进行操作。因此，期望获得具有相等或几乎相等大小的处理单元，因为没有单元将使其它单元减速。即使对于没有全局时钟信号的设计，分支计算中的 2个数据路径具有相等长度，即每个数据路径中存在的节点数量相同也是有利的。通过保证数据路径具有相等长度， 2个分支中的计算将以相同速度执行。

如图5a所示，图3的 2个节点 304和 305已经合并成一个节点 504。如上所述，可以进行此处理，以便均衡不同数据路径的长度，或优化设计的总体处理速度。

通过消除某些节点之间的寄存器来执行节点合并，其中随着合并节点变得更大，节点数量将减少。通过系统地合并所选节点，节点的组合深度变得基本相等，并且不同节点之间的处理速度得到均衡。

当合并节点时，其各自的功能也被合并。这通过不经任何中间寄存器地连接不同逻辑单元来进行。随着节点被合并，必须确定新触发规则以便使节点当需要时在其输出处提供数据。

更具体地，如图5b中所示，当合并 2个节点 507， 508时，产生

具有与初始节点具有的输入和输出弧的相同数量减去连接被组合的 2个结点507， 508的弧的新节点 509。如上所述，对于类似于相加，相乘等等的基本功能节点，触发规则将在所有输入上存在数据并且所有输出自由接收数据时触发 (下面称作 nm-触发规则的触发规则)。合并2个这样的节点 507， 508将产生具有 3个输入和一个输出的新节点509。来自相加的 2个输入，来自相乘的 2个输入，在 2个节点之间的连接中使用的一个输入提供合并节点的 3个输入。来自相加的一个输出，来自相乘的一个输出和用来连接 2个节点的一个输出提供了来自合并节点的单个输出。合并节点的触发规则将需要所有其 3个输入处的数据触发。事实上，具有 nm -触发规则的节点的任何合并本身将具有 nm -触发规则，尽管输入和输出的数量将已经改变。通过根据先前连接它们的弧直接将来自第一组合块的输出连接到其它组合块的输入，合并初始 2个节点 507,508的功能。先前表示节点之间的弧的寄存器被消除。于是，结果是更大的组合块。

因此，对于始终需要其输入处的数据并且始终在其输出处提供数据的节点，例如执行算术功能的节点，合并节点的触发规则将与初始节点相同。

如上所述，功能编程语言的使用对于在数据流机中实现巨大并行性是必要的。根据本发明，通过标记处理副作用的问题。通过使用称作实例标记(instance token)的特殊标记，能够控制对副作用的可能访问的数量，以及这些访问发生的顺序。

除了普通数据输入之外，希望使用副作用的每个节点必须具有用于与所涉及的副作用相关的实例标记的专用数据输入。除了用于实例标记的数据输入之外，它也必须具有用于实例标记的输出。用于实例标记的数据路径充当数据流机中的其它数据路径，即节点在其能够执行其特定操作之前必须具有所有相关输入上的数据。

需要访问副作用的节点的触发规则使得其必须在其实例标记输入 (即实例标记本身)上具有数据。当完成对副作用的访问时，节点释放其输出处的实例标记。这个输出接着可以被连接到需要访问相同副

作用的后续节点的实例标记输入。通过这个过程，在需要访问特定副作用的所有节点之间建立实例标记路径。实例标记路径决定节点获得对副作用的访问的顺序。

因此，对于特定副作用 (例如存储器或指示器)，存在一或多个沿着其实例标记路径移动的实例标记。由于该链中的所有节点需要在其输入上具有数据以便获得对副作用的访问，能够通过限制实例标记数据路径上的数据单元数量 (即限制实例标记的数量)来约束对副作用的同时访问的数量。如果在一个特定时刻只有一个实例标记被允许存在于实例标记路径上，则保证从不从两个或更多节点同时访问副作用。此外，通过实例标记路径明确确定访问副作用的顺序。如果在某些环境期间使不止一个节点访问副作用是安全的，则能够同时在路径中引入不止一个实例标记。分割实例标记路径也可以是安全的，从而将实例标记复制到分割的两个路径。

例如，当访问作为副作用的存储器时，如果两个路径只包含从存储器的读取，则分割实例标记路径将通常是安全的。在这种情况下，对存储器的同时访问将由存储器控制器任意仲裁，但是由于读取的执行顺序彼此不影响，这是安全的。相反，如果 2个路径包含写入，则实际执行 2个写入的顺序将是必要的，因为这将决定存储器最终保存什么值。在这种情况下，实例标记路径不能被安全地分割。

在实例标记路径的单线程上将若干实例标记放置在彼此之后将通常表示通过不同"代(generation)"的流水线计算对存储器的访问。如果例如已知由于它们不访问存储器的相同部分，因而2个代是无关的，则将多个实例标记插入在彼此之后将是安全的。

也可以在彼此之后放置对若干不同副作用 (例如存储器或其它输入或输出单元)的访问。这将具有明确确定对路径上每个实例标记的每个副作用的访问的顺序的效果。例如，从输入单元的读取能够被放置在对实例标记路径上输出单元的写入之前，如果若干实例标记同时存在于路径上，则读和写的总体顺序可能是未确定的，但是对于路径上的每个单个实例标记，在副作用之间将存在清楚的定序。

最终，当设计数字电路时，能够混合不同类型的数据流机。例如，可以使静态数据流机中具有取决于数据的数量的迭代的循环作为动态数据流机的部分。这样可以并行执行迭代，而在静态数据流机中是不能如此的。在没有动态数据流机的完全标记匹配系统的情况下，这样的静态数据流机的本地动态部分能够进行操作。相反，只需要保证标记按照它们进入动态部分的相同顺序来退出动态部分。因为机器的其余部分是静态的并且不对标记重定序，这将使得标记匹配。

在通过用序号标记(tagging)进入递归的每个标记(token)并且使用缓冲区收集不按顺序(out of order)完成递归的标记来完成递归之后，能够以正确顺序重新排列标记。更具体地，在递归步骤之后安排缓冲区。如果标记不按顺序退出递归，其将被放置在缓冲区中，直到具有更低序号的所有标记退出递归。因此，缓冲区的尺寸决定有多少标记可以不按顺序地退出递归，同时保证在递归完成之后标记能够被正确地排列。在某些情况下，能够知道退出递归的标记的顺序是无关的，例如如果要执行退出递归的标记的值的简单求和。在这样情况下，可以省略用序号标记数据标记和缓冲区。

除数据相关循环之外，本地标记匹配和重定序模式的使用也可以被用于节点或子图的其它类型的重定序。

已经参照优选实施例描述了本发明。然而如所附权利要求书所定义的，除在这里公开的实施例之外的其它实施例也在本发明的范围内。

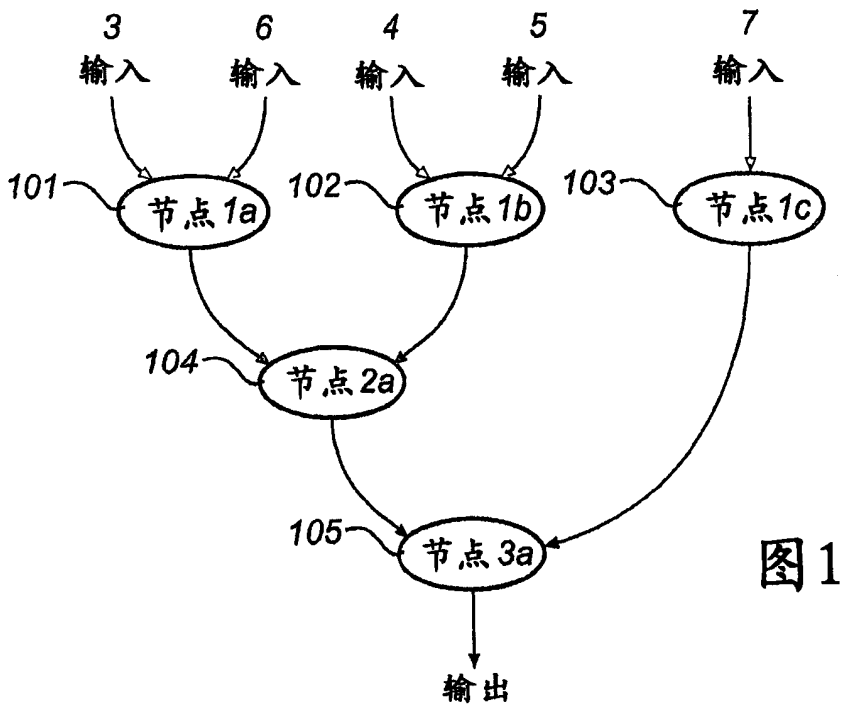

3
输入
6
输入
4
输入
5
输入
7
输入
101
节点1a
102
节点1b
103
节点1c
104
节点2a
105
节点3a
输出

图1a

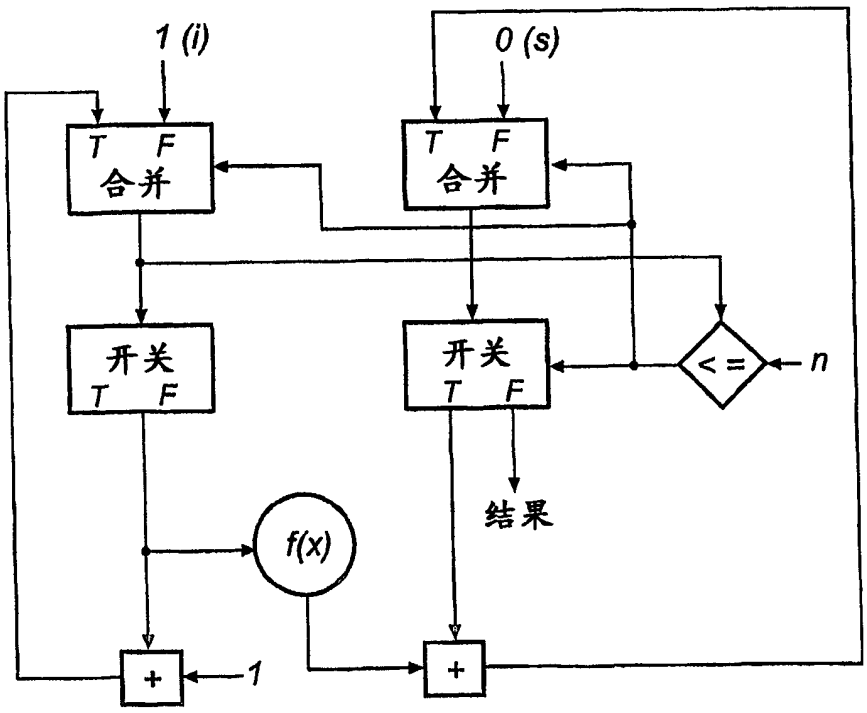

1 (i)
0 (s)
T F
合并
T F
合并
开关
T F
开关
T F
<=
n
结果
f(x)
+
1
+

图1b

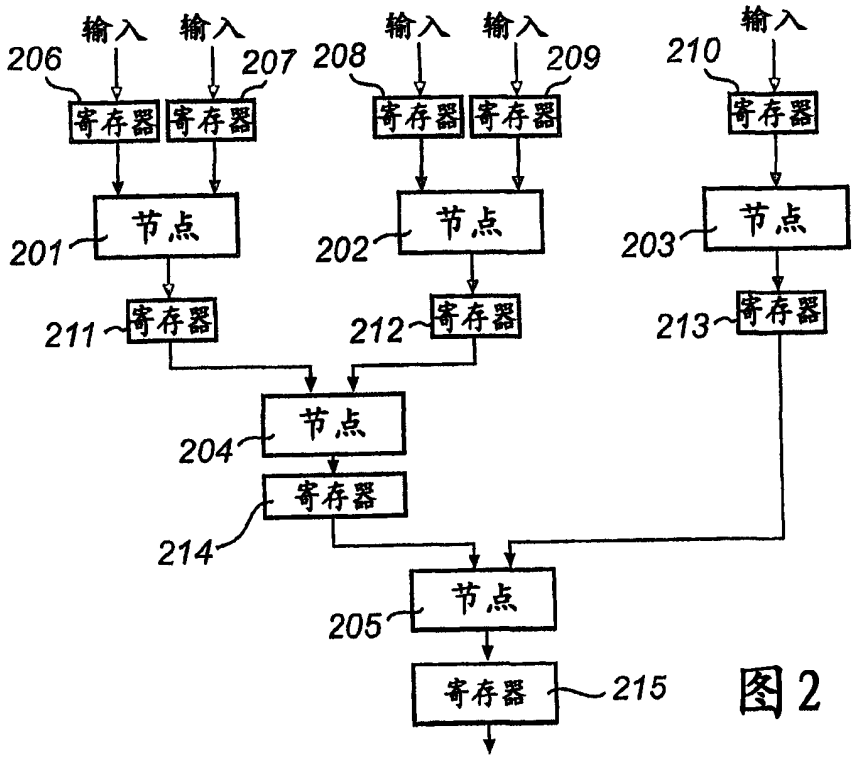

输入
输入
输入
输入
输入
206
207
208
209
210
寄存器
寄存器
寄存器
寄存器
寄存器
节点
节点
节点
201
202
203
211
212
213
寄存器
寄存器
寄存器
节点
204
寄存器
214
节点
205
寄存器
215

图 2

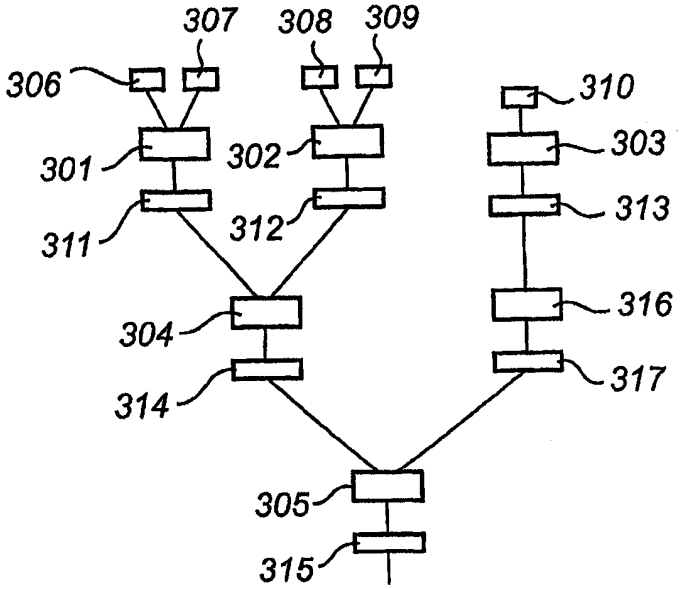

307
308
309
306
310
301
302
303
311
312
313
304
316
314
317
305
315

图 3

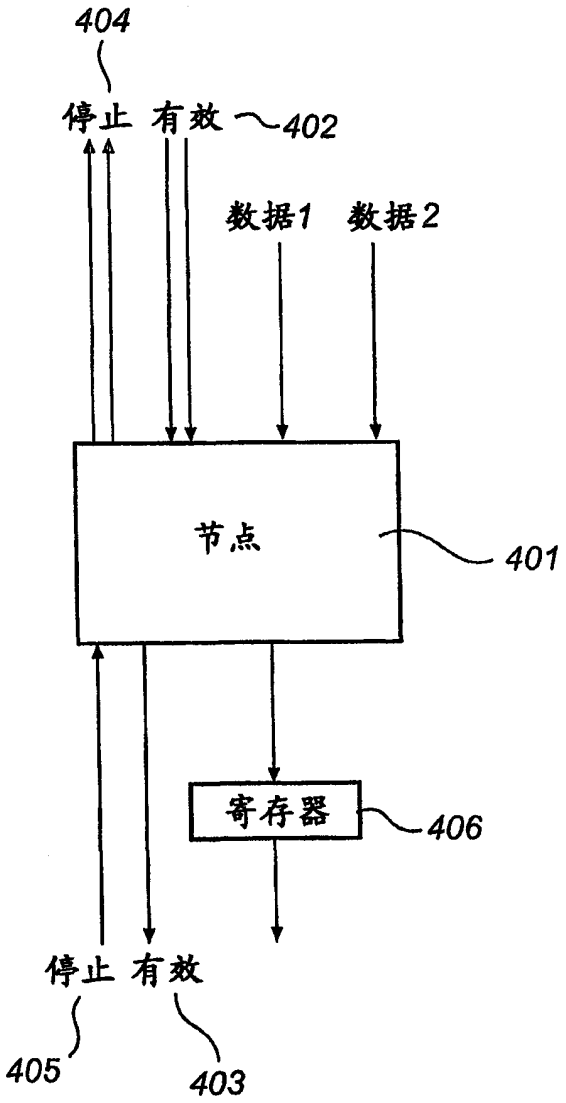


图4a

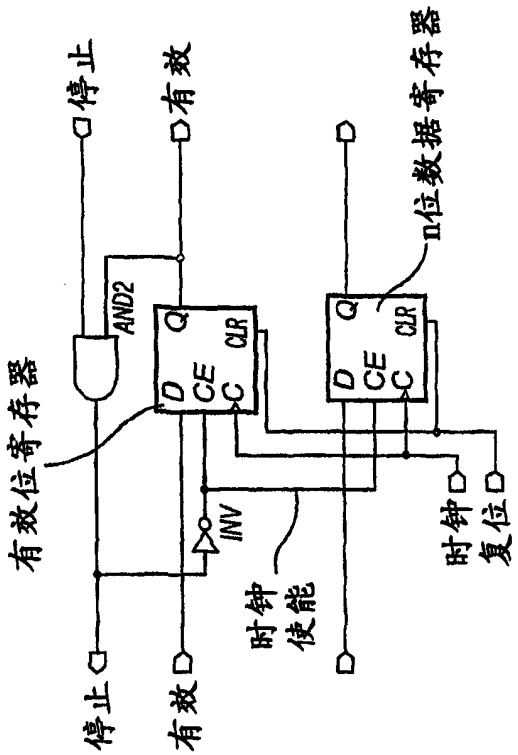

停止
有效
AND2
Q
D
CE
C
CLR
有效位寄存器
INV
时钟
使能
n位数据寄存器
时钟
复位
停止
有效

图4c

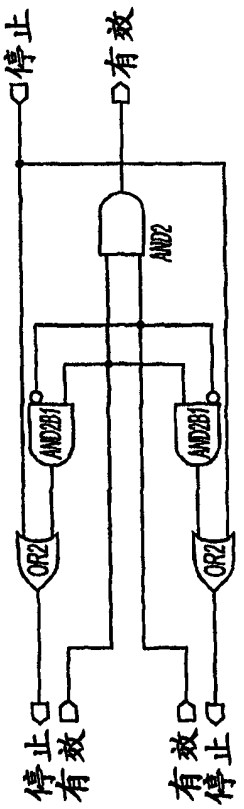

停止
有效
AND2
AND2B1
OR2
停止
有效
有效
停止

图4b

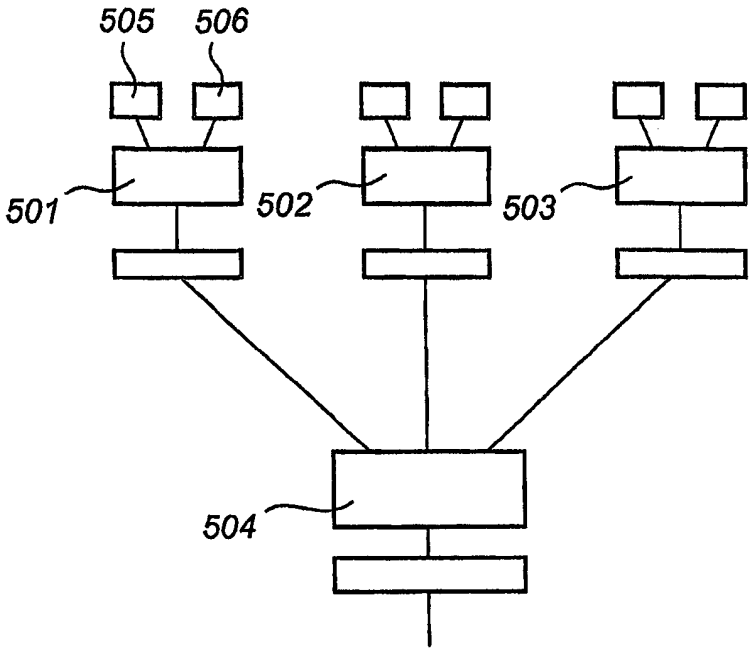


图5a

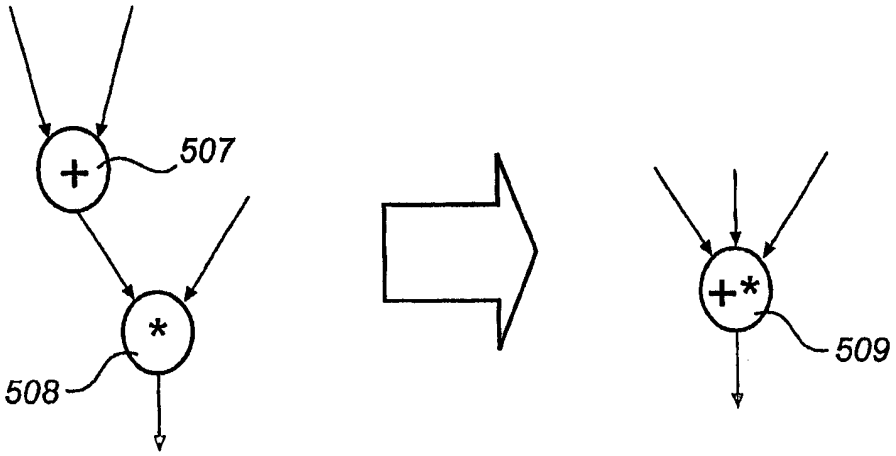


图5b

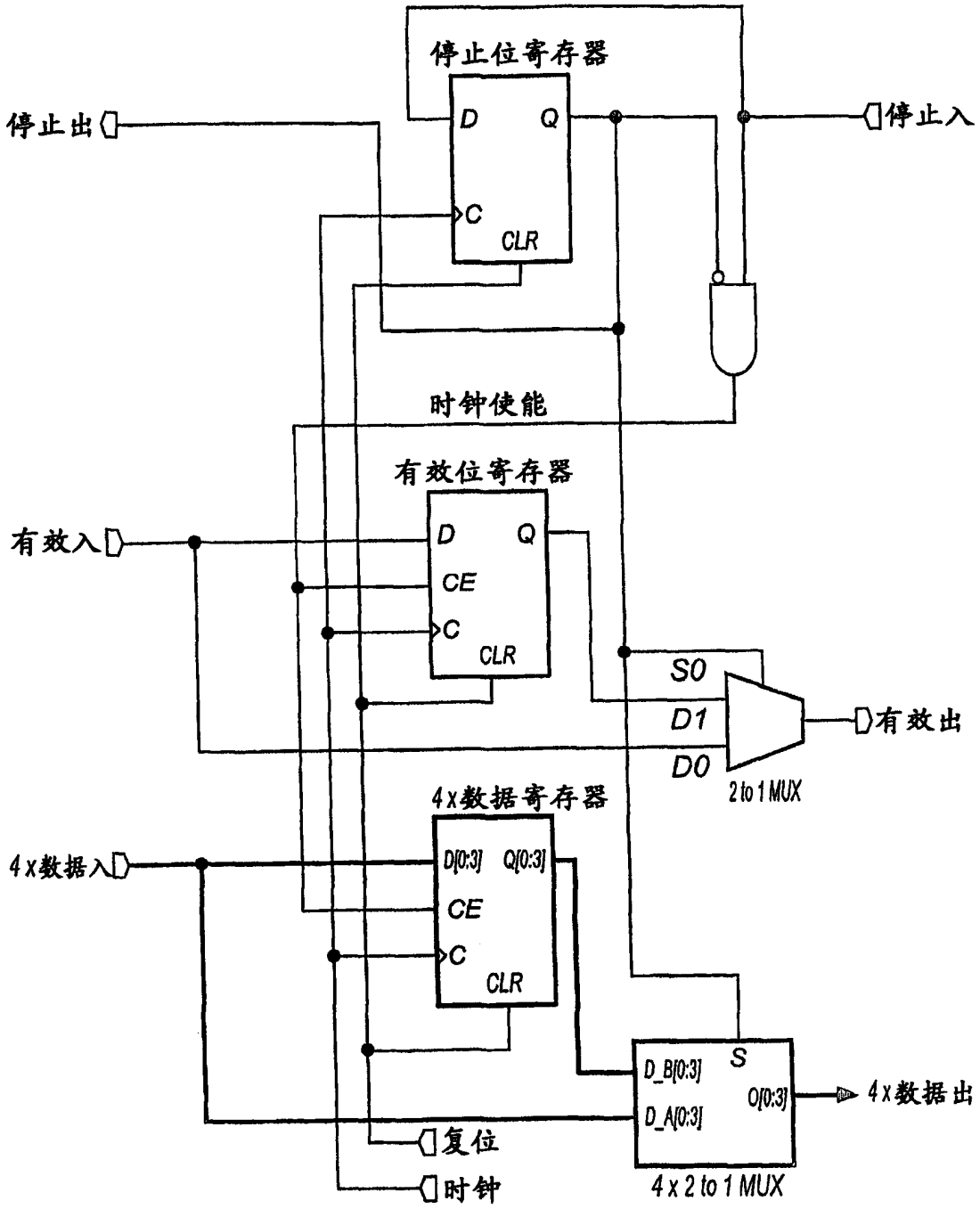

停止位寄存器
停止出
D
Q
C
CLR
停止入
时钟使能
有效位寄存器
有效入
D
Q
CE
C
CLR
S0
D1
D0
有效出
2 to 1 MUX
4 x数据寄存器
4 x数据入
D[0:3]
Q[0:3]
CE
C
CLR
D_B[0:3]
D_A[0:3]
S
O[0:3]
4 x数据出
复位
时钟
4 x 2 to 1 MUX

图6