

[12]发明专利申请公开说明书

[21]申请号 98804866.3

[43]公开日 2000年9月13日

[11]公开号 CN 1266512A

[22]申请日 1998.4.30 [21]申请号 98804866.3

[30]优先权

[32]1997.5.8 [33]US [31]60/045,951

[32]1997.11.6 [33]US [31]08/965,540

[86]国际申请 PCT/US98/08719 1998.4.30

[87]国际公布 WO98/50852 英 1998.11.12

[85]进入国家阶段日期 1999.11.8

[71]申请人 艾瑞迪公司

地址 美国加利福尼亚州

[72]发明人 T·C·波夫 J·S·米那米

R·柯亚玛

[74]专利代理机构 上海华东专利事务所

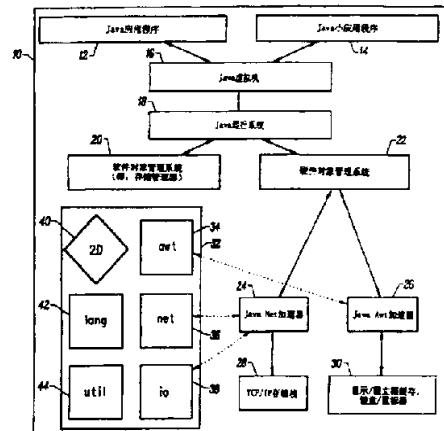
代理人 肖剑南

权利要求书 6 页 说明书 38 页 附图页数 20 页

[54]发明名称 适用于面向对象编程语言的硬件加速器

[57]摘要

本发明公开了一种用于加速面向对象编程语言的方法和装置,它是在硬件门级上提供的。在一个附从于Java语言的实施例中,一种Java应用程序结构已在硬件中实现。在本发明的最佳实施例中支持Java. AWT, Java. NET, 和 Java. I O 的应用程序结构。应用程序结构分类被存储在共享存储器库中。由小应用程序执行的支持应用程序结构分类的实体和方法被卸载到硬件对象管理系统。一个软件占位程序提供作为介于硬件对象管理系统和中央处理器之间的接口程序。通过修改或替换软件占位程序能够支持另外的应用程序结构。硬件对象管理系统请求是由一个应用程序特别结构硬件加速器来执行的。需要时可从共享存储器中检索应用程序结构分类,而执行指令则被存储在共享存储器中以便由中央处理器进行存取。当硬件对象管理系统的硬件加速器执行程序请求时,中央处理器可连续处理不配套的应用程序结构指令。



权利要求书

1. 一种用于加速运行面向对象编程语言处理器的装置，其特征在于，它包含：

 用于完成至少一种所述面向对象编程语言的应用程序结构的硬件加速器；
以及

 用于把所述硬件加速器连接到所述处理器的软件占位程序。

2. 按照权利要求 1 所述的装置，其特征在于：所述面向对象编程语言是 Java。

3. 按照权利要求 1 所述的装置，其特征在于，还包含：用于管理所述应用程序结构的实体和分配给所述实体的数值状态的硬件对象管理系统。

4. 按照权利要求 1 所述的装置，其特征在于，所述硬件加速器包含：

 用于接收和顺序地存储每一待决执行指令的所述指令的输入/输出请求队列；

 用于处理所述指令的任务处理器；以及

 用于追踪引用计数数到一实体并用于去除未在使用实体的现用对象目录。

5. 按照权利要求 1 所述的装置，其特征在于：所述硬件加速器实现 Java.AWT 应用程序结构。

6. 按照权利要求 5 所述的装置，其特征在于，还包含：

 一窗口/观察系统；以及

 一映射驱动程序。

7. 按照权利要求 6 所述的装置，其特征在于，所述窗口/观察系统包含：

用于建立图文框和组元并用于把数据传送到所述映射驱动程序的一通用

图形控制器；

用于管理所述图文框的窗口管理器；

用于在所述图文框之内管理包容文件分层的布局管理器；以及

用于在所述包容文件之内管理组元分层的组元管理器。

8.按照权利要求 1 所述的装置，其特征在于：所述硬件加速器实现 Java.NET 和 Java.IO 应用程序结构。

9.按照权利要求 8 所述的装置，其特征在于，还包含：

一窗口/观察系统；以及

一连接驱动程序。

10.按照权利要求 9 所述的装置，其特征在于，所述窗口/观察系统包含：
对所述硬件对象管理系统敏感的网络控制器，所述网络控制器实现 Java.NET 结构等价的微代码，其中所述网络控制器通过由所述连接驱动程序支持的协议作为抽象层运行；

用于执行 DNS 查找并用于把报告结果传送到上述网络控制器的网络查询机制；

用于管理由应用程序使用的网络界面的网络界面接口管理器；以及

用于通过由网络协议支持的链路往返空间输入和输出数据的数据流管理器。

11.按照权利要求 1 所述的装置，其特征在于：所述硬件加速器构成作为应用程序专用集成电路的一部份。

12.一种用于加速运行 Java 编程语言的处理器的装置，其特征在于，它

包含:

用于实现至少一种 Java 应用程序结构的硬件加速器;

用于管理应用程序结构的实体和分配给所述实体的数值状态的硬件对象管理系统，所述硬件对象管理系统包含用于接收和顺序地存储每一待决执行指令的所述指令的输入/输出请求队列，用于处理所述指令的任务处理器，以及用于追踪引用计数数到一实体并用于去除未在使用实体的现用对象目录；以及

用于把所述硬件对象管理系统连接到所述处理器的软件占位程序。

13.按照权利要求 12 所述的装置，其特征在于：所述硬件加速器实现 Java.AWT 应用程序结构。

14.按照权利要求 13 所述的装置，其特征在于，还包含：

一窗口/观察系统；以及

一映射驱动程序。

15.按照权利要求 14 所述的装置，其特征在于，所述窗口/观察系统包含：

用于建立图文框和组元并用于把数据传送到所述映射驱动程序的通用图形控制器；

用于管理所述图文框的窗口管理器；

用于在所述图文框之内管理包容文件分层的布局管理器；以及

用于在所述包容文件之内管理组元分层的组元管理器。

16.按照权利要求 12 所述的装置，其特征在于：所述硬件加速器实现 Java.NET 和 Java.IO 应用程序结构。

17.按照权利要求 16 所述的装置，其特征在于，还包含：

一窗口/观察系统；以及

一连接驱动程序。

18.按照权利要求 17 所述的装置，其特征在于，所述窗口/观察系统包

含：

对所述硬件对象管理系统敏感的网络控制器，所述网络控制器实现 Java.NET 结构等价的微代码，其中所述网络控制器通过由所述连接驱动程序支持的协议作为抽象层运行；

用于执行 DNS 查找并用于把报告结果传送到上述网络控制器的网络查询机制；

用于管理由应用程序使用的网络界面的网络界面接口管理器；以及

用于通过由所述协议支持的链路往返空间输入和输出数据的数据流管理器。

19.按照权利要求 12 所述的装置，其特征在于：所述硬件加速器构成作为应用程序专用集成电路的一部份。

20.一种用于加速运行面向对象编程语言处理器的方法，其特征在于，它包含如下步骤：

用硬件加速器完成至少一种所述面向对象应用程序结构的应用程序结构；以及

用软件占位程序把所述硬件加速器连接到所述处理器。

21.按照权利要求 20 所述的方法，其特征在于：所述面向对象编程语言是 Java。

22.按照权利要求 20 所述的方法，其特征在于，还包含：用硬件对象管理系统管理所述应用程序结构的实体和分配给所述实体的数值状态的步骤。

23.按照权利要求 20 所述的方法，其特征在于，所述实现至少一个应用程序结构的步骤包含如下步骤：

接收和顺序地把所述指令存储在每一待决执行指令的输入/输出请求队列中；

在任务处理器中处理所述指令；以及

借助现用对象目录追踪引用计数数到一实体并去除未在使用的实体。

24.按照权利要求 20 所述的方法，其特征在于：所述硬件加速器实现 Java.AWT 应用程序结构。

25.按照权利要求 24 所述的方法，其特征在于，还包含如下步骤：

用窗口/观察系统建立并显示窗口和视图；以及

用映射驱动程序在显示屏上绘制所述窗口和视图。

26.按照权利要求 25 所述的方法，其特征在于，还包含如下步骤：

提供通用图形控制器用于建立图文框和组元并用于把数据传送到所述映射驱动程序；

提供窗口管理器用于管理所述图文框；

提供布局管理器用于在所述图文框之内管理包容文件分层；以及

提供组元管理器用于在所述包容文件之内管理组元分层。

27.按照权利要求 20 所述的方法，其特征在于：所述硬件加速器实现 Java.NET 和 Java.IO 应用程序结构。

28.按照权利要求 27 所述的方法，其特征在于，还包含如下步骤：

用窗口/观察系统建立并显示窗口和视图；以及
应用连接驱动程序操纵网络连接。

- 29.按照权利要求 28 所述的方法，其特征在于，还包含如下步骤：
提供对所述硬件对象管理系统敏感的网络控制器，所述网络控制器实现
Java.NET 结构等价的微代码，其中所述网络控制器通过由所述连接驱动程
序支持的协议作为抽象层运行；
提供网络查询机制用于执行 DNS 查找并用于把报告结果传送到所述网
络控制器；
用网络界面接口管理器管理由所述应用程序使用的网络界面；以及
用数据流管理器通过由所述协议支持的链路往返空间输入和输出数据。

30.按照权利要求 20 所述的方法，其特征在于：所述硬件加速器的构成
为应用程序专用集成电路的一部份。

31.按照权利要求 22 所述的方法，其特征在于：所述硬件对象管理系统
提供对象 IDs 返回到调用系统，以及其中所述对象 ID 可能是可变位长
度。

32.按照权利要求 22 所述的方法，其特征在于，所述硬件对象管理系统
可接受来自调用系统的对象 IDs 以便识别，所述对象识别各对象，所述方法
还包含如下步骤：

- 把基对象 ID 存储在基引用寄存器中并使所述基对象 ID 与空闲对象定
位单元相关联；以及
把所有另外的对象 IDs 引用到所述基引用寄存器使所述对象 ID 与一
对象定位单元相关联。

说 明 书

适用于面向对象编程语言的硬件加速器

发明的背景

技术领域

本发明涉及面向对象编程语言。更确切地说，本发明涉及一种适用于面向对象编程语言的硬件加速器。

现有技术的描述

面向对象编程 (OOP) 是基于 "对象" 概念的一种编程语言和技术的分类名称。一个对象是在程序中有特定作用的独立组元。OOP 语言包括 C++，以及由美国 Sun 微系统公司(Sun Microsystems,Inc.) 开发的 Java 语言。仅仅为了讨论的目的，本文中述及的 OOP 语言指的是 Java。

OOP 定义对象的分类。"类"是一个对象或一组共享公共结构和特征的对象集合的原型。对象也称为 "实体"，是类的特定表示法。类的每一实体根据它们的特定属性加以区分。这些属性由 "实体变量" 予以定义。在类中定义这些实体变量的类型和名称。然而，实体变量的数值却是在对象中设定和变换的。

类的特征决定该类的实体如何操作。类的特征是由与所述类有关的，被称为 "方法" 的一组例行程序确定的。在对象上的操作是应用方法来完成的。方法对一特定类的所有实体是共用的。

类是分层次排列的。在层次中每一类可以有一个类置于该层之上称之为 "超类"，或有一个类置于其下被称为 "子类"。子类从它们的超类中 "继承" 属性和特征。这样，一个子类就不必对在超类中定义的特征重新定义，但

却能够从超类中接受方法和变量。

在 OOP 中，过程调用以报文传送术语来描述。报文指定方法并可任选地包括其他自变量。当将报文发送到对象时，对象的类寻求其中指定的方法，从而确定如何在给定对象上完成所请求的操作。若该方法未被定义用于上述对象的类，则搜索该对象的超类。该过程要贯穿所述类的整个层次，一直持续到不是找到上述方法定义就是再没有更高层的超类可找为止。

调用过程自变量的排列由一调用约定加以确定。该调用约定确定特定的指令，按这些指令，将自变量推向存储栈或进入寄存器，同样它还担负消除上述自变量。

藉助调用约定或接口程序，应用访问操作系统和其他服务被称为应用程序编程接口(API)。在 Java 中，核心 Java API 定义最小的功能集，考虑到 Java 的顺应性，它必须受到计算机系统基础技术的支持。

Java 支持称之为应用程序的独立程序和支持称之为小应用程序的从属程序。Java 小应用程序是通过电子网络传输的程序，这些程序由接收装置上的应用程序执行。上述电子网络的实例包括因特网和局部区域网络系统，在此处称之为内联网。接收装置包括计算机，个人数字助理，以及能连因特网的电话。

此处为了讨论起见，假设一个 Java 小应用程序将通过因特网传送到接收的计算机供显示在万维网 (Web) 网页中用。一个 Java 应用程序是可独立应用的程序，并不需要借助一个应用程序来执行，例如，在接收的计算机上设置一个 Web 浏览器。

Java 是独立的计算机平台语言，能够靠任何 Java 应允的计算机系统运

行。Java 开发环境包括一 Java 编译程序和一 Java 解释程序。该 Java 编译程序从一 Java 程序中生成字节码。Java 字节码是机器指令而不是特别的计算机平台。

将一个特别的计算机平台字节码翻译器用于执行 Java 程序。该字节码翻译器被称为 Java 虚拟机。适用于小应用程序，把上述字节码翻译器装入一启动 Java 的 Web 浏览器内部。Java 虚拟机和它的支持代码统称为 Java 运行系统。

Java 虚拟机包括字节码指令集，一组寄存器，一用以存储方法的区域，一存储栈和一无用单元收集堆。Java 虚拟机寄存器暂时保存表示机器状态的数据。上述寄存器对机器运行起反应，在执行每个字节码以后进行更新数据。上述方法区域存储 Java 字节码，它们执行 Java 系统中的大部分方法。

存储堆栈用于为字节码和方法组供应参数，并收回它们的结果。栈帧包含用于方法调用的局部变量，其执行环境，以及其操作数栈。

所述堆是存储单元，从存储单元中重新产生的实体予以保留。实体会自动地在 Java 中被“无用存储单元收集”。无用单元收集程序是一种程序设计子例程，将它设计成能追踪每个被建立的实体，并在对上述实体的最终访问已消失后，释放存储一个实体的存储器。

在 Java 中，把在不同程序中打算反复重新使用的 对象/分类 作为一个“分类库”加以存储。一组分类体现用以解决许多相关问题的抽象设计，通常称为可重用基本设计结构。存储在上述 Java 分类库中的核心应用可重用基本设计结构包括 Java.Lang, Java.Util, Java.Applet, Java.IO, Java.NET, 以及 Java.awt。

Java.Lang 包括把数据类型和系统权能添加到语言本身的分类，而 Java.Util 包含实用程序分类和简单类集分类。Java.Applet 包含用以执行 Java 小应用程序的分类。

Java.IO 包含从数据流中写入和读出的输入和输出分类，例如，标准的输入和输出。Java.IO 还包括适用于处理文件的输入和输出分类。

Java.NET 包含联网支持分类。该分类包括按照比如标准万维网协议，例如 TCP/IP 和 IP，用以连接和检索文件的那些分类，以及用以建立网络界面接口比如那些用在 UNIX 的应用程序的那些分类。

Java 抽象分屏成套软件工具 (Java.awt) 提供用以建立基于图形用户接口 (GUI) 小应用程序和应用程序的分类和特征。例如上述 Java.awt 能够用在屏幕绘图，建立窗口、按钮、选项栏、滑动块区和其他用户界面元素。Java.awt 还能用于处理用户输入，例如鼠标器单击和击键。

面向对象软件可重用基本设计结构一般是使用计算机的随机存储器 (RAM)，只读存储器 (ROM) 或虚拟存储器 (VM) 来执行的。例如在计算机上启动 Java 应用程序或小应用程序时，首先执行的一项操作是建立 GUI。

虽然，因为基于软件的工具正在应用，是能够要求计算机操作系统处理能力的有效百分率来实际生成并显示所述 GUI 的。但为此当 GUI 元素正呈现在显示设备上的时候，由上述操作系统处理的其他指令就会明显地放慢下来。另外，在显示正进行更新的同时，倘若发生鼠标或键盘事件，Java 运行可能进入一种死锁状态，在更新事件发生的同时，系统也正尝试处理待办的鼠标或键盘事件。最后运行系统不能在合理的时间范围内抓起并处理所有的事件。直到最终用户程序出现中止工作，明显地拒绝进一步的命令和输入。

Java 是一个多处理单元/多线程处理系统。因而，它支持许多程序，以及同时执行许多各自在其自己地址空间内的处理过程。线程是执行这些处理过程中之一的代码序列。因此，Java 编程环境和运行系统两者都有一个多个线程体系结构。

在一个 Java 的多线程程序中，运行 Java 程序的一个单线程可能会被阻塞，直到例如一个窗口被拉下。此外，任何依赖于该单线程的其他线程也可能被阻塞。举例来说，通过网络传输信息，阻塞能够约束或延迟。阻塞就这样被定义为，一个 Java 线程在重新开始执行之前等待任务完成的能力。

往往希望是作为嵌入式环境的一部分来运行 Java 程序，例如，在诸如能连因特网的电话机那样的设备上运行 Java 应用程序和小应用程序。然而，需要 Java 代码和一快速字节码处理器来使上述设备与 Java 相容。可是这会显著增大制作和操作该设备的成本。

因此若能减少性能的降低而提供一种方法和设备则将是一个优点。如果这种方法和设备允许面向对象的程序既能有效应用于嵌入式环境又能有效应用于台式环境中，则将是另一个优点。

发明概要

本发明提供一种使面向对象的编程语言得以加速的方法和设备。本发明的最佳实施例配置成与一 Java 相容的处理器一起使用，用 Java 编程语言和标准 Java 基于对象的可重用基本设计结构。然而，本发明的另一些实施例适合于与任何面向对象编程语言一起应用。本发明的另一些实施例还适合于与其它基于对象的结构一起应用，例如，苹果计算机的“开放式步骤”结构或微软的基于对象的 AFC 库。

本发明提供一种硬件加速器，它包含执行一种或多种 Java 应用程序结构的功能。在本发明目前最佳的实施例中，将加速器构成作为一个专用集成电路 (ASIC) 的一部分。上述加速器包括一硬件对象管理系统，它管理实体以及分派给实体的数值的状态。

本发明的最佳实施例执行 Java 抽象分屏成套软件工具(AWT), Java.NET, 以及 Java 输入/输出 应用程序结构。此外，如果需要的话，本发明还能够支持另一些 Java 结构。

Java 运行系统既管理硬件对象管理系统又管理软件对象管理系统。上述软件对象管理系统可以完全用软件方式来完成，或既用软件又用硬件来完成。

由一 Java 程序执行的，支持 Java 应用程序结构分类的实体和方法被卸载到硬件对象管理系统。于是，在本发明的最佳实施例中，在上述硬件对象管理系统管理 Java.AWT 和 Java.NET 请求的同时，中央处理器 (CPU) 仍能继续处理指令。

本发明存储应用程序结构分类作为在共用存储器中的分类库。然而，每一支持应用程序结构是分别加以管理的。一项请求被分配到特定应用程序结构的硬件控制器，根据需要访问共用存储器。

一软件占位程序被提供作为硬件对象管理系统和 CPU 之间的一个接口。上述软件占位程序能够进行修改和替换，以允许本发明可与任何兼容的 OOP 语言一起应用。

硬件对象管理系统包括一个用以接收和存储对象请求的输入/输出请求队列。当任务处理器为对象指定存储器时，就把它从队列中除去。现用对象

列表追踪 Java 虚拟机中的引用计数数到一实体。当到实体的引用计数数为零时，借助一无用存储单元收集程序释放存储上述对象的存储器。

在本发明目前最佳的实施例中，Java.AWT 硬件工具包含一个窗口/观察系统和一映射驱动程序。一个图形 Java 程序构成有如一个分层的嵌套式包容文件和组元。包容文件管理称之为组元的可视对象的收集。组元表示在屏幕上显示的可视功能性，例如，选项栏，按钮，可编辑文本区和文本区段。包容文件本身可以相互间分层嵌套，并可与另一个包容文件保持父代 <-> 子代关系。对一给定的应用程序来说，在视窗层的顶部有一包容文件视窗，它是借助一视窗系统 (AWT. 图文框) 对象呈现的。AWT. 图文框把本机窗口定义为专门的计算机平台 (即 WindowsTMSolarisTM)。

此外，还能够把包容文件看成是特定分类组元，它有容纳其它一些组元的能力。这是因为从功能性来看包容文件分类为组元的子类。屏幕是在一应用程序或小应用程序范围以内可被显示在一显示屏上的包容文件。图文框是带标题和选项栏的窗口，同时还是另一种组元/包容文件的类型。共享分类特征意味着所有这些分类允许它们共享各种特性，并能很好集成在一起。所述 Java.AWT 硬件工具通过优化通信和增加包容文件显示效率，在不同的组元对象变型之间影响这些共享功能。

包容文件是能够容纳其它组元的 Java.AWT 组元。屏幕是能够被显示在显示屏上的一个包容文件，例如一个小应用程序。图文框是具有像标题和选项栏这样一类特征的窗口。

窗口/观察系统包括：一个用以建立图文框和组元，并用以把数据传送到映射驱动程序的通用图形控制器；一个用以管理图文框的窗口管理器；一

个在图文框范围内用以管理包容文件分层的布局管理器；以及一个在包容文件范围内用以管理组元分层的组元管理器。

Java.NET 硬件工具包含一个窗口/观察系统和一个连通性驱动程序。上述窗口/观察系统包括一网络控制器，它包含用于 Java.NET 可重用基本设计结构等价的微代码。一网络查询机制执行 DNS 查找并把报告结果传到上述网络控制器，一网络界面接口管理器管理由应用程序使用的网络界面，以及一数据流管理器用于通过由网络协议支持的链路往返空间输入和输出数据。

因此，本发明提供一种硬件工具，它借助支持应用程序结构的协同指令用以减少阻塞。Java 处理过程由此得以加速。

附图的简要说明

图 1 是显示按照本发明的硬件加速器 10 结构性综述的框图；

图 2 是按照本发明的 Java 加速器的示意框图；

图 3 显示按照本发明在对象实体化和删除以后的对象目录表；

图 4 是显示按照本发明的包容文件与其内部组元实体连接序列的框图；

图 5 是显示按照本发明的现用对象数据清单的框图；

图 6 是按照本发明的硬件加速器的内部功能框图；

图 7 是图 6 上示出的 Java.awt 加速器的更加详细的框图；

图 8 是显示全部应用本发明同时使用几个线程的 JVM 的框图；

图 9 是按照本发明在硬件中处理 AWT 显示对象的理想话路图；

图 10 是显示由本发明支持的 Java 抽象分屏全套软件工具对象的框图；

图 11 是按照本发明的 Java 抽象分屏全套软件工具硬件实现的功能框

图；

图 12 是按照本发明的视窗系统的框图；

图 13 显示按照本发明由一映射驱动程序支持的组元表；

图 14 是显示按照本发明结合处理实例的示意图；

图 15 显示由图 13 的映射驱动程序的图形程序节支持的映射类型表；

图 16 是显示按照本发明的 Java.AWT 加速器“指令”执行和对象管理的框图；

图 17 是按照本发明嵌入 AWT 程序节中的功能性说明；

图 18 是按照本发明的 Java 抽象分屏全套软件工具窗口/包容文件/观察实体分层结构的实施例的框图；

图 19 是由本发明支持的 Java.Net 对象的框图；

图 20 是按照本发明的 Java.Net 加速器的功能框图；以及

图 21 是按照本发明嵌入 NET 程序节中的功能性说明。

发明的详细描述

本发明提供一种用以加速面向对象的编程语言的方法和设备。本发明的最佳实施例已加以优化可与由美国 Sun 微系统公司开发的 Java 编程语言，并与 Java 应用程序接口一起使用。然而，本发明也容易适合于与任何面向对象的编程语言一起使用，例如 C++ 程序设计语言，或甚至可适用于以面向对象形式书写的程序 C 语言。因此，以下论述只是作为例子而提供的，并不能作为对本发明范围的限制。

同样地，与 Java 加速器内连的工具也可应用不同的应用程序结构，或设定应用程序结构不与 Sun 公司的 Java APIS 相连。

虽然，本发明的最佳实施例适合用于通过因特网的数据传送，但本发明等同地可应用于其他广域网或局域网。另外，尽管最佳实施例是与台式计算机一起应用的，实际上本发明也可与其他设备一起应用，包括网络服务器，个人数据助理，传真机和能与因特网联网的电话。

以下述及的显示屏和图形用户接口的布局结构是按照本发明目前最佳的实施例配置的。然而，熟悉本技术领域的人士会理解到，上述显示屏和图形用户接口为迎合本发明另一个实施例的需要，是很容易加以修改的。因此，以下论述仅仅提供作为本发明的例子，而不能作为对本发明范围的限制。

图 1 是显示按照本发明的硬件加速器 10 结构性综述的框图。靠计算机的 CPU (图上未示出) 运行的 Java 虚拟机 16，是供靠计算机系统运行的 Java 应用程序 12 用的字节码翻译器。为了 Java 小应用程序 14，将上述字节码翻译器构成可启动 Java 的万维网浏览器。

硬件对象管理

Java 加速器的存储器 400 (参见图 2) 的一部分被预先指定容纳对象描述信息和参数。这些对象空间全都是相等尺寸的，但各自的格式却取决于分配给它们的对象类型。在每个对象描述信息的起始点有一个状态字节，它指明那个特定对象空间是否是在使用中。

在启动的同时，标志符表明为正空闲着。于是软件可应用以下步骤开始建立对象：

- * 将第一项命令或任务发布到 AWT 命令寄存器 407 或 NET 命令寄存器 413。
- * 上述命令接着在 AWT 命令解码器 408 或 NET 命令解码器 414 进行

解码。

- * 一旦确定需要有待建立对象的类型，就从空闲对象定位器 403 获得下一个可用的对象 ID，并与任何其它状态信息一道通过返回操作先进先出(FIFOs) 405 或 412 回传，经由 AWT 数据寄存器 406 或 NET 数据寄存器 410 退出系统的处理器。
- * 只要上述空闲对象定位器将其对象 ID 传回到系统，它便出去并开始为下一个空闲对象空间轮询可用的存储器 400。
- * 当它发现一个时，它把对象空间的 ID 存储在所述下一个对象建立命令的预报中。
- * 万一，空闲对象定位器循环通过所有记忆存储器却没有找到一个可用的对象空间，将会把存储器超出状态返回给系统。
- * 当系统希望存取一已经建立的特定对象时，它把任务和对象的 ID 一道传到 AWT 命令寄存器 407 或 NET 命令寄存器 413。对象 ID 和任务被分列并译码，同时上述 ID 被传送到对象 ID 分解器 402。该程序块接着把对象 ID 转换成一存储器地址，可以用该地址在存储器块 400 中存取一特定的对象空间。
- * 当系统接收到对象删除请求，恰当的命令和对象的 ID 一道被传到命令寄存器。上述任务和 ID 被译码和分列，同时上述 ID 随即被传到对象分解器。Java 加速器这时开动并清除在存储器中该特定对象空间中表示使用着的状态位，这样它就使空闲对象定位器可用于下一个对象建立命令。在空闲对象定位器之前已报告存储器超出状态的情况下，上述新近释放对象的 ID 被直接存储在该空闲对象定位器中，因而消除所述定位器循环通过存储器寻

找上述定位的需要。

* 该空闲对象定位器具有存储下一个 “n” 空闲对象 IDs 的能力。将这种能力应用在每个单独对象空间尺寸很小的场合，某些对象于是就利用 “n” 对象空间使之链接在一起。在这种情况下，第一个对象空间包含一个链接域，它指明连续的对象标示符存储在哪里。这种在每一逐次对象空间中的连续直到第 “n-1” 空间为止，它包含一个到第 n 个 (“nth”) 对象空间的链路。在这种结构形式下，该空闲对象定位器响应一对象建立命令，报告反馈起始对象空间的 ID。同样地，如果确定有待建立的对象的类型需要多个对象空间，若有足够的空闲对象空间可用，那么该空闲对象定位器仅仅报告反馈起始的 ID。反之，则返回存储器超出状态信号。在这种情况下，对象空间并非必须按连续序列，而可以是随机链接的。该空闲对象定位器应当能够存储 “n” 空闲对象 IDs，此处 “n” 是有待建立的最大对象所需要的最大限度的对象空间数量。

由硬件提供的所述对象 IDs 在长度方面可以是任何比特数。

该硬件对象管理器也可在对象 IDs 由主机系统提供而不是由硬件返回的状态下运行。在这种情况下，将第一个提供的对象 ID 存储在一变址引用计数寄存器中，并与所述第一空闲对象定位相关联。每个接着发生的对象请求，其 ID 都要与所述引用计数寄存器加以比较。上述偏移量确定在存储器中的实际对象定位。倘若该特定记录已经在使用着，那么就应用下一个空闲对象定位。一预定的引用字段要与每个对象记录一起算入，以表明其关联的偏移量。偏移量是按模量计算方式完成的，为此当偏移量大于配给的存储空间时，会使存储器定位再回卷到起点。该系统也适用于系统提供的对象 ID 比

特数大于硬件对象 ID 的状态下运行。

图 3 用实例说明在几个对象已被实例化和删除以后，所述堆看来像什么情况。要注意已删除的 5 号对象当前已释放可供重新使用。应当指出包容文件对象实质上居于组元链接表的首位。当对象数值为零时上述列表终止。为此并没有 0 号对象，任何对它的引用都是无效的。

其它对象类型可显示出与上述相同的功能性。例如一按钮对象类似地引用一串对象，它本身就可能超过 30 比特。在这种情况下，按钮对象会指向一串对象，后者会依次指向每个对象按 30 比特长度量度的另一串对象。由此显示出组合对象是如何进入列表功能的。

图 4 明确显示包容文件与其内部组元实体的连接序列，在图 4 中不直接示出相同的列表。

在现用对象列表上的对象

图 5 是显示按照本发明的现用对象数据清单的框图。该现用对象数据清单包含可能有的许多 30 比特块的列表，各自代表一个特定对象类型的现用实体。每个对象类型有与其关联的结构。所有实体化的对象都包含标志，该标志描述对象类型，运行系统的方法选择程序和对象所归属的结构。边框布局实体另外还包含给包容文件用以安排一组组元的指针，有可能对象引用到其在北面，南面，东面和西面的组元，以及当绘制它们时确定在组元之间离开(在象素中)空间的数值。包容文件对象包含封装组元的数量，引用持有该包容文件的接受实体值的组元对象，以及引用组元对象链接表中第一包容文件对象。组元对象包含一“类型”值，它代表他们是否应当按压一按钮，或根据一按钮，滑动块，文本字段或其它窗口小部件类型来处理事件。它们还

包含一个通往包容文件中下一个可用组元的链路。

本发明将对象 (参照图 5) 处理在小程序块中。所述对象依据它们的大小和覆盖区看起来总是相同的。在现用对象列表上实体相互间的区分是其数据结构或实体变量。

实体可以是或简单或复杂的。它们可包含 8,16,32 位整数或单位布尔值。它们还可包含对象对其他对象的参考 IDs。

就包容文件对象的情况而言。在已经把关于包容文件对象的请求置于输入/输出请求队列以后, J 快速取得该请求并建立一固定大小的存储块以存储与该包容文件实体有关的变量。在建立几个组元对象以后, 可将它们加到该包容文件上。如果指定一布局管理程序管理分配给包容文件的对象, 这是相应地完成的。在包容文件 <-> 边框布局伴随共生的情况下, 可将五个可能的组元直接加到上述包容文件上。该边框布局对象于是就被给定最多可引用五个 (北,南,东,西和中) 组元, 这样它就知道如何在以后恰当的时间取用并更新这些对象。

现用对象列表包含可能许多个 30 字节块的目录, 其中每个代表特定对象类型的现用实体。

每个对象类型有一与以下示出的三个对象实体图中关联的结构。

所有实体化的对象都包含描述对象类型的标志, 用于运行系统的方法选择程序, 以及对象所归属的结构。边框布局实体另外还包含给包容文件用以安排一组组元的指针, 有可能对象引用到其在北, 南, 东和西的组元, 以及当绘制它们时确定在组元之间离开(在象素中)空间的数值。

包容文件对象包含封装组元的数量, 引用持有该包容文件的接受实体值

的组元对象，以及引用组元对象链接表中第一包容文件对象。组元对象包含一“类型”值，它代表他们是否应当按压按钮，或根据按钮，滑动块，文本字段或其它窗口小部件类型来处理事件。它们还包含一个通往包容文件中下一个可用组元的链路。

这些对象定义直接映射进入由上述可重用基本设计结构所管理的 C 数据类型结构中（要注意这些是初步的结构，都将会改变且进一步优化）。

类型定义 结构 ir_ 描述符_ // 对象的开头三字节。

{ j 字节 结构类型;_ // 结构的特定类型(Java.awt,NET,IFC,等等)

j 字节 对象类型;_ // 对象的特定类型(图文框,事件,包容文件,等等)

j 字节 方法类型;_ // 特定对象方法(-画图(),等等)

j 字节 使用着;_ // 找出是否有记录在使用着。

} 降序;

类型定义 结构 ir_ 包容文件_ // 提示:每个包容文件必须引用一个组元结构

{ 降序_ 对象描述符;

j 短_n 组元;_ // 包容文件中现有组元数。

矩形_ 嵌入;

_ // 布局管理程序可以是五个系统设定之一。

j 字节_ 布局类型;_ // 代表用于该包容文件的布局类型。

j 接口_ 布局对象;_ // 布局实体的特定索引。

j 接口_ 接受组元对象;_ // 我们是组元的子类所以我们需要这

_____ // 对组元结构的索引。

////////// 随变量而定的图文框-对象。

_j 布尔 是可调整大小的; __ // 与图文框对象一起使用。

_j 字节 光标类型; __ // 光标 0-13 与图文框对象一起使用。

_无符号短菜单栏对象; __ // 菜单栏往往附属于图文框对象。

//////////

接口 第一个组元节点;

} 包容文件, 对话, 图文框, 窗口; // 为方便起见指定多个名称。

_____ // 一窗口=图文框=对话=包容文件远至

_____ // 涉及 J 快递。

类型定义 结构 ir_组元

{ 降序 对象描述符;

_j 字节 同层对象类型; __ // 这是组元的选中对象值

_____ // 例如, 按钮, 标号, 文本字段, 等等。

_无符号 短 X;

_无符号 短 Y;

_无符号 短 宽度;

_无符号 短 高度;

_无符号 短 父代对象; __ // 对包容文件对象索引。

_无符号 短 字体对象; __ // 对描述由该组元使用着的字体的对象索引。

无符号 短 前景颜色对象; // 对前景颜色对象索引。
无符号 短 背景颜色对象; // 对背景颜色对象索引。
无符号 短 光标对象; // 对光标对象索引。
无符号 短 串对象; // 引用包含串的对象到
_____ // 设法在组元中上色(可能是 0)。
_j 布尔 可见的;
_j 布尔 启动;
_j 布尔 有效;
// 尺寸 最小尺寸;_ // 组元包含的最小允许尺寸。
// 尺寸 最佳尺寸;_ // 组元包含的最佳尺寸。
j 接口 下一个组元节点; // 对在包容文件中下一个已知组元素索引。
} 详细检查, 组元;_ // 详细检查是仅仅用画图()方法的组元。

类型定义 结构 ir_边框布局

{ 降序 对象描述符;

 j 短 h 间隔;

 j 短 v 间隔;

 j 短 包容文件;_ // 索引包容文件的阵列中与我们有关的。

 j 短 北组元;_ // 索引组元的阵列中指针指向之处。

 j 短 南组元;

 j 短 西组元;

 j 短 东组元;

j 短 中组元;
} 边框布局;

类型定义 结构 ir_信息流布局

{ 降序 对象描述符;
 j 短 h 间隔;
 j 短 v 间隔;
 j 短 对准;

} 信息流布局;

类型定义 结构 ir_网格布局

{ 降序 对象描述符;
 无符号 短 h 间隔;
 无符号 短 v 间隔;
 无符号 短 行;
 无符号 短 列;
} 网格布局;

类型定义 结构 ir_W 图形

{ 降序 对象描述符;
 无符号 短 X,Y;
 无符号 短 宽度, 高度;

无符号 短 前景颜色对象;_ // 对前景颜色对象索引.

无符号 短 X 偏移量, Y 偏移量;

矩形 剪辑矩形;__ // 由 W 图形:改变剪辑()设定

} W 图形;

其他概念

本文中提出的对象定义的要点 (与其它本文中未定义的一起) 被用于建立实体和保持独特的实体数值。不用这些数据结构, 上述在现用对象列表中 30 字节块就会毫无意义。应用这些对象结构来识别, 设定和存取在特定对象实体范围内的数值。所述运行系统应用对象实体的收集从而形成方法执行, 对给定数据集是很有意义的。

可以为对象规定在描述信息结构中的方法描述符, 以便将所述对象传送到低级帧缓存执行。该智能化的帧缓存接着可读出所述结构, 并以合乎理性的方式翻译一项命令 (例如, 调整大小或刷新屏幕)。在这时图形和组元对象就把它们的结构转到该智能化帧缓存。

Java 运行系统 18 包括 Java 虚拟机以及它的支持代码。该 Java 运行系统分列实体化和方法调用。在本发明的最佳实施例中, 该 Java 运行系统管理两个实体管理程序, 一个硬件对象管理系统 22 和一个软件对象管理系统 20。然而, 在本发明的另一个实施例中, 该 Java 运行系统管理两个以上实体管理程序。

上述硬件对象管理系统包含支持 Java 应用程序结构的功能。在本发明目前最佳的实施例中, 该硬件对象管理系统执行 Java 抽象分屏成套软件工

具 (AWT) 26, 和 Java.NET 24 应用程序结构。Java 输入/输出 应用程序结构 (未示出) 也执行支持 Java.NET 功能。上述配件对象管理系统当需要时可容易地适用于支持另外一些 Java 结构。为此, 本发明提供一种 Java 应用程序结构硬件工具, 它对 CPU 起到一个加速器的作用。

该硬件对象管理系统管理实体以及分配给实体的数值状态。在本发明目前最佳的实施例中, 该硬件对象管理系统全部用门加以实现, 并可外接随机存储器(RAM)。

在该实施例中, 硬件对象管理系统被构成为一专用集成电路(ASIC)的一部份。应予理解的是, 虽然本发明的最佳实施例述及与门级工具和专用集成电路(ASIC)相联系, 但对熟悉本技术领域的人士来说本发明的实际硬件实现是可考虑加以选择的。

上述软件对象管理系统可完全用软件来实现。然而, 在本发明的另一个实施例中, 该软件对象管理系统是既用软件又用硬件元件来实现的。软件对象管理系统的一个实例是计算机系统的存储管理器。

用 Java 程序执行的支持 Java 应用程序结构分类的实体和方法被卸载到硬件对象管理系统。因此, 在本发明的最佳实施例中, 在所述硬件对象管理系统管理 Java.AWT 和 Java.NET 请求的同时, CPU 能够连续处理指令。

无论什么时候在所述硬件实现的 Java 应用程序结构分类中的实体和方法正通过一 Java 程序执行时, 上述实体和方法就被卸载到硬件对象管理系统。这样, 在本发明的最佳实施例中, 该硬件对象管理系统管理靠 Java 虚拟机运行的线程的 Java.AWT 和 Java.NET 请求。因此, 在所述硬件对象管理系统管理 Java.AWT 和 Java.NET 请求的同时, CPU 仍可用于连续处理指

令和线程而不取决于卸载的实体和方法。

打算在不同程序中要反复重新使用的应用程序结构分类被存储于 Java 分类库 32(图 1)中。这些分类可包括 Java.awt 34, Java.net 36, Java.lang 42, Java.io 38 和 Java.util 44 的那些应用程序结构。

上述应用程序结构分类作为分类库被存储在共享的存储器中。尽管，每个支持应用程序结构是各别地进行管理的。将一请求分配给特定应用程序结构的硬件控制器，而需要时可访问共享的存储器。

用以完成网络运行的分类和特征，例如，建立一网络界面接口连接，是借助 Java.net 加速器 24 来完成的。Java.net 加速器配置一个接口通往网络 TCP/IP 存储栈 28。该 Java.net 加速器包括一 Java.net 36 联网对象和 Java.io 38 输入/输出相关对象两者的硬件实现。这是因为联网需要含有数据的输入和输出。把两个结构都结合进硬件内使冗余软件编码能显著减少。

在 Java.net 加速器的实现的一个实施例中，应用 Java.io 硬件工具在网络界面接口连接期间处理数据输入和输出。在 Java.net 加速器中，Java.net 和 Java.io 应用程序结构分类通过继承性被约束在一起。这样，在一个结构中的分类能接受在另一个结构中的功能并叠加于其上。

Java.awt 加速器完成所述抽象分屏成套软件工具的分类和方法（即，特征）。该 Java 抽象分屏成套软件工具 (Java.awt) 提供用以建立基于图形用户接口 (GUI) 的小应用程序和应用程序的分类和特征。这样的分类包括 Java.awt 类 32, 和任何关联的绘制工具，例如，二维绘制驱动程序 40。

Java.awt 加速器可用于绘图到显示屏上，建立窗口，按钮，选项栏，

滑动块区，以及其他用户接口元素，而且还能用于处理用户输入，例如，鼠标器单击和击键。因此，为 Java.AWT 加速器配置一接口通往上述设备 30 就可用作计算机显示，帧缓存，键盘和鼠标器。

结构上的综述

图 6 和 7 是本发明结构上的概要框图。所述支持应用程序结构，例如，Java.awt 26 (在图 7 中相当详细地示出)，Java.NET 24，以及 Java.IO (图上未示出) 都被分类整理到一 ASIC 中从而构成加速器。对以下讨论来说，Java.NET 加速器将假定包括 Java.IO 的输入/输出相关对象。

在本发明的最佳实施例中，硬件对象管理系统 22 也被分类整理到 ASIC 中去管理实体和分配给实体的数值状态。例如，该硬件对象管理系统存储数据以追踪一按钮是否被按压。该硬件对象管理系统还存储指明对象之间关系的数据，例如，按钮是特定窗口的一部份。该信息存储在共享的存储器内。

提供软件占位程序作为每个加速器和运行 Java 虚拟机的 CPU 之间的一个接口。于是，Java.NET 加速器与 Java.NET 软件占位程序 52 和 Java.IO 软件占位程序 54 相关联，而 Java.awt 加速器与 Java.awt 软件占位程序 50 相关联。本发明覆盖硬件组元的软件占位程序是能够加以修改和替代的，其允许本发明与任何可兼容的 OOP 语言一道使用。

以上述及的图 2 是按照本发明的硬件加速器 60 的内部功能框图。提供寄存器接口程序 64 (406, 407, 409, 410, 413, 415) 以监控并同步化传送在主机 CPU 62 与本发明之间的所有输入和输出。

Java 附从的主机 CPU 运转 Java 虚拟机，例如，来自只读存储器

(ROM)的存储器 88。经由物理传输 94 传送到本发明的 Java 小应用程序或应用程序，接着经由物理传输接口 86 对准主机 CPU。AWT 命令解码器 408 和 NET 命令解码器 414 用于把二进制输入信息转换成用二进制代码表示的输出元素。

上述小应用程序或应用程序以字节码形式存储在存储器 400 内。必须建立一 GUI 用以显示小应用程序和应用程序。因此，当 Java 程序靠上述存储器一开始运行时，构成一调用到存储在 ROM 中的功能库，指示生成 GUI 显示。

借助于硬件对象管理系统 402, 403 和软件占位程序，把上述调用指向相应的 Java 应用程序结构加速器。举例来说，当 Java.awt 窗口被请求时，该 Java.awt 加速器建立一窗口对象。图形命令生成器 80 接着被请求指挥图形控制器 82 在显示屏 500 上实际绘制出窗口。

Java.NET 加速器可包括电子邮件和万维网浏览器应用驱动程序 74。于是，举例来说，应用本发明可以从因特网下载 JPFG 或 GIF 图象。这时，上述图象通过网络 TCP/IP 栈 76 被引导到 JPEG/GIF 解码器 78，并接着导向由一帧缓存管理程序 84 控制的帧缓存器 92。从而，本发明用一硬件工具加速从网上下载的图形图象的显示。

存储控制器/判优器 401 担任主机 CPU 和共享主存储器 400 之间的接口。该存储控制器/判优器在本发明的各种组元之间分配和操纵系统存储器。上述存储器管理包括对存储器的分配，例如，分配给 Java.awt 加速器，或为 CPU 准备由本发明所建立的对象的存储单元。

图 8 是显示按照本发明 Java 虚拟机/运行时间的多线程操作的框图。

因为 Java 有一多线程的体系结构，靠 Java 虚拟机 16 和运行时间 18 运行的 Java 小应用程序 12 和应用程序 14 可以是用同时执行的多线程 100, 102, 104, 106 构成的。

包含一种支持 Java 应用程序结构的分类和方法的线程，经由硬件对象管理系统被引导到本发明。于是，在本发明的最佳实施例中，硬件对象管理系统管理靠上述 Java 虚拟机运行的线程 Java.awt 26 和 Java.NET 24 的请求。因此，在硬件对象管理系统管理 Java.awt 和 Java.NET 请求的同时，CPU 仍能够不依赖在其上的线程和指令连续处理。

图 9 是按照本发明的硬件对象管理系统图。本发明管理 Java.awt 和 Java.NET 请求，如上所述这些请求是由靠 Java 虚拟机运行的线程构成的。对象在专用缓冲器中被保留和去除保留。

硬件对象管理系统包括用以接收和存储对象请求的输入/输出请求队列 108。上述请求可以包括：

建立一个包容文件 110;

建立一个边框布局 112;

建立一个界面 114;

建立一个标识符 116;

建立一个文本字段 118; 以及

建立一个按钮 120。

当任务处理器 150 为一对象分配存储区时，例如，在建立一个新包容文件对象 152 之前，该对象从队列中移走。现用对象列表 122 存储实体 124 的目录。将使用着的标志 140 提供给一个实体，以指明在 Java 虚拟机中引用

计数 126 的总数。

当给一个实体的引用计数数字为零 142 时，存储该对象的存储区已通过无用单元收集程序（图上未示出）去除保留。该现用对象列表也将指明所述特定实体未在使用 128。

一个对象的状态被存储在一状态表 144 中。所述状态可包括该对象是在使用着 146，或存储该对象的存储区已被去除保留 148。

图 10 是显示由本发明支持的 Java 抽象分屏全套软件工具对象的框图。AWT 快递结构连接到基于软件的 Java.awt 占位程序。AWT 快递结构 154 本身能够分解成两部份。低层实现对于把基本的 Java 命令转换到我们的显示是必需的，并允许对鼠标/光笔/键盘事件的控制以及更新窗口和查看层面。高层实现基本上为 AWT 的余项加特征，例如把为 UI 元素(按钮，文本区，文本字段)的微化画图特征包含进去。

Java 虚拟机用一组命令（任务）156, 158 发送数据到本发明，并从本发明接收数据。Java 编程语言主要依靠低层对象。因此，低层寄存器 158 包括对于 Java 的实现是极为重要的对象。

在本发明的一个实施例中，只执行低层实现。该实施例大大减小尺寸和存储容量要求，而且对用于嵌入式系统中是很有利的。

低层的功能性包括基本分屏控制，图象绘制，字体控制，以及事件处理。由本发明最佳实施例支持的低层对象包括：

彩色模型 210;

组元 212;

包容文件 214;

图象裁剪筛选 216;
直接换色模型 218;
事件 220;
字体 222;
字体规格 224;
图文框 226;
图形 228;
网格包压缩 230;
图象 232;
图象筛选 234;
彩色模型索引 236;
选项单组元 238;
象素数据采集器 240; 以及
RGB 图象筛选 242。

高层寄存器 156 提供能够任选地在硬件中执行的补充功能。这些高层功能包括标准 GUI 窗口小部件, GUI 布局特征, 以及存储媒体控制对象。由本发明的最佳实施例支持的高层对象包括:

边框布局 160;
按钮 162;
详细检查 164;
卡片布局 166;

校验框 168;
校验框组 170;
校验框选项 172;
选择 174;
对话框 176;
信息流布局 178;
网格包布局 180;
网格布局 182;
标识符 184;
目录 186;
媒体跟踪器 188;
菜单 190;
菜单栏 192;
可选项 194;
多边形 196;
矩形 198;
滚动条 200;
文本区 202;
文本组元 204;
文本字段 206; 以及
窗口 208。
因为本发明用硬件代表完整的应用程序结构，所以本发明显著省略代码

数量的需要。例如，OOP 应用程序结构的某些部件能够通过常规应用来构成，而其它部件则由于应用程序结构的独特应用被隐蔽起来。然而，在硬件实现中，是不需要隐蔽这些结构部件的。省略冗余编码和仅执行那些可公开地访问，在硬件门阵列范围内启动任务的软件占位程序的功能是更为有用的。

从而，在本发明中可省略许多 Java 应用程序结构的对象和分类。一个被省略分类的例子是美国 Sun 微系统公司的 Java.NET. 网络界面输入数据流分类。该分类源自 Java.IO. 文件输入数据流并只在原始基类上增加极少量功能。因此，本发明能够省略这样冗余的分类，却没有导致重要功能的流失。

图 11 是按照本发明的 Java 抽象分屏全套软件工具硬件实现的功能框图。一个包括 Java.awt 应用程序结构实体和方法的线程，从 Java 虚拟机 16 卸载并经由 Java.awt 软件占位程序 50 到达硬件对象管理系统 22。寄存器接口 64 监控并同步化地传送在主机 CPU 62 与硬件对象管理系统之间的所有输入和输出。

对象请求被接收并存储，直到在 输入/输出 请求队列 108 中进行处理。接着任务处理器 150 将每一请求引导到硬件 AWT 加速器 250。该加速器包括窗口/观察系统 252 和映射驱动程序 260。

上述窗口/观察系统包括用以管理由本发明生成的图文框的一窗口管理器 254。该布局管理器 256 管理在每个图文框范围内包容文件的分层，而组元管理器 258 则管理在每一包容文件内的组元。

上述窗口/观察系统还包括通用图形控制器 82，它用于建立图文框和组元并把数据传到映射驱动程序。在该通用图形控制器中完成用于 Java.awt 应用程序结构的硬件等价微代码。由窗口/观察系统的其它组元和映射驱动

程序相应地作出并执行对象/方法 请求。

映射驱动程序 260 应用绘制工具，例如，图象再现器 262，多边形再现器 264，以及字体再现器 266 建立所请求的对象。熟悉本技术领域的人士会理解到，所述绘制工具的类型包括映射驱动程序在内，是可以按照支持应用程序结构的特定需要而加以改变的。

图象再现器 262 接收并还原压缩图象数据。于是就在显示屏的给定组元范围内再现所述图象。多边形再现器 264 为再现基于向量的图象提供加速。能够支持预定制图类型，例如，绘制弧形，线条，椭圆形，多边形和矩形。字体再现器 266 管理用以显示文本的请求。

窗口服务器和对象执行

窗口服务器/分屏系统（参照图 12）出现在离开 Java 虚拟机的最远点。它依赖于从对象执行机制（即，运行系统）转到其上的命令，而从不直接从软件层接收命令。因此，窗口服务器没有来自软件层的直接接口。

这大体上相当于 Java 在 PC 上运行的方式。一般，建立对等对象以映射从 Java 处理器到运行系统之外编译资源的请求。对 Windows '95/NTTM 或任何其它现代操作系统来说，这意味着作出对标准视窗系统不是 C 就是汇编代码的调用。

用在这些操作系统中的编译代码，就这样对实现系统级分屏请求形成基本的需要。窗口受控于显示屏上，往往利用标准按钮，滑动块和其它基于操作系统的 UI 窗口小部件来简化编码，增强性能，并提供与主机本身操作系统适配的用户接口。在美国 Sun 微系统公司的 Java 开发工具中，这是应用一对等接口程序加以调用的，使用一组基于软件的分类占位程序映射在 Java

对象与编译本机码之间的功能。

本发明不严格地应用对等的概念。无论如何，所述对等实现完全是在门级实现情况下予以提供的，而且，只有从其它基于专用集成电路功能（即，硬编码 Java 对象）才是可存取的。从许多透视系数中了解到这样做的意义。首先，可以在寄存器定义层很容易地把基于对等的绘制类型换出，允许常规或多个 UI 实现得以建立。其次，在 ASICS 中能够适当地将对等间隔成适用于硬件的不同区域。这是硬件与软件之间的差异。在软件中要建立逻辑功能组合，而在 ASICs 则要考虑到效率和减少门计数。

在软件中我们可以按顺序方式逻辑地组合基于对象的功能。所有基于观察的对象可以在一处...一个结构中运行得非常协调。在硬件中却必须把事情分开来。人们可以在窗口服务器为每个 UI 对象建立一个对象实现。然而，结果弄清楚这既不有效也不合乎需要。

需要指出的是，倘若可能，为使门计数减到最低限度应当牺牲精美，在图 12 中作出更有吸引力的方案。

应当指出在这个分屏系统的实现中，只有少许可与标准 Java ‘对等’对象相比较的对象。它们与只用软件 Java 实现中的对等有相当的差别因为它们支持更为抽象的功能组。一个软件实现可能把对等分配给每一个 UI 对象（即，按钮，滑动块，文本组元，等等），而本发明采用一个单一组元对等，处理所有与组元有关的绘制请求。

映射驱动程序是如何运行的

映射驱动程序 511 当时压缩所有现行的绘制代码，并为解释图文框 512，图形 513，组元 514 和程序串 515 实体提供再现器/处理器。该映射驱

动程序一般通过存储器引用或经由一固定大小数据流(30字节)存取这些实体。所述实体包含一命令字节，指导映射驱动程序关于该对象应执行哪一种方法。

当该映射驱动程序接收一实体时，它首先对正被发送到四个支持对象之一的对象类型解码。一旦搞明对象类型，就把适用于该特定对象类型的参数传送到解码器。上述执行方法(例如，组元：画图())被确定，同时该方法得以执行。

绘制组元实体

就一组元：画图()而言，所述对象被进一步询问以找出有待显示的组元的类型。映射驱动程序的当前组元扇区知道，该如何再现图13上示出的基于观察的窗口小部件的类型。

一旦映射驱动程序察觉到组元类型是在绘制，就询问实体关于例如窗口小部件存储单元和尺寸之类的信息。然后上述组元就在显示屏上再现。

因为对象都是相当小的，不可能在组元实体范围内传送程序串。情况正是如此，在本发明中一个实体的固定大小多半是30字节，它可能比程序串本身的大小更小。为此，在发送现行组元到映射驱动程序之前，先发送程序串或至少引用程序串。

程序串由映射驱动程序累加存储。数据流或引用固定字节对象包含串信息一般被送到映射驱动程序。该程序串可能后随另一个包含另一个程序串的固定程序块。连续传输被并置在一程序串的末端上，直到一组元实体到达。一旦所述组元到达，该累加存储的程序串与其相关联，于是该组元就与其程序串一起再现。在文本可能有非常庞大程序串的情况下，上述并置特别有必要。

要。

应当指出，在最终 ASIC 形式中，上述并置处理（参照图 14）包括把指针累加到固定程序块，其包含用以再现的程序串而不是并置的程序串本身。这是很有必要的，因为某些组元可能有很大的文本内容，例如，在传统 Java 上的文本区多半是 32 千字节或更大些。一文本区基本上是一文本编辑对象。把本发明的型式按比例缩小有可能任意分配一最大文本字符数，使之能够保证有效应用在只有小的存储覆盖区的设备上。

一旦组元与其程序串得以再现，程序串本身就被复位为零，处于为下一个程序串/组元结合的准备状态。

绘制图形实体

图形实体与组元实体有所不同。图形实体担负绘制低层图元，在这方面组元可驱动这图元或甚至驱动数以百计的图元，包括线条，实心区和文本，以便绘制某些像可编辑文本区那样。图形对象支持基本要素绘制的通用类型。图 15 示出的绘制类型是由映射驱动程序中的图形对象支持的。

图形对等实现在映射驱动程序中看起来是相当复杂的，但实际上比组元对等要简单得多。图形满足很低层次绘制，而组元通常把一组图形绘制特征组合进如前面段落所描述的单个宏指令中。

绘制图文框实体

图文框比较直截了当地说就是它们做什么。当这是图文框对等实现的一种分类时，它在一特定类型的显示屏上绘制窗口。窗口显示类型可取决于上述实现而改变，并可支持多种个性类型，情况就是概括驱动程序的全面作用。

通常，图文框有一标题栏，一图标图象和一与其关联的光标类型。图文框按组元和图形相同的方式运转，但被形成在它们包含的 UI 后面。

图文框在标准 Java 实现中是包容文件的一种类型。然而，因为这有些像帧元同位体，它与任何其它对象没有直接关系。就映射驱动程序而论，一个窗口仅仅是具有一宏指令的特殊对象，它绘制特定的线条集合，等等。它表现得与一组元在该层次上的工作方式非常相似。

图 16 是显示按照本发明的 Java.AWT 加速器对象管理的框图。对象请求在执行驱动程序 270 中排队等候 输入/输出请求队列 108 待决处理。当一个对象开始激活时，它就加入现用对象目录 122。

在图 16 中，输入/输出 队列请求中第一个对象请求是建立一个图文框对象 272。当所述图文框被建立时，就把 274 加入到现用对象目录。下一个对象请求，建立一个事件对象 276，接着就被移向上述输入/输出 请求队列的顶部。当所述事件被建立时，也把 278 加入到现用对象目录。

其余对象激活一直进行到对象 Java 虚拟机无用单元回收程序引用计数减少到零为止（参照以上图 9）。此外，需要时对象可包含相互引用。例如，窗口范围内的关系，窗口之中的一个按钮，依据所述按钮显示的文本就是由上述引用加以表明的。

图 17 是按照本发明在 Java.AWT 加速器中窗口/观察系统 252 功能性的说明。在该通用图形控制器 82 中完成用于 Java.AWT 应用程序结构的硬件等价微代码。在本发明的最佳实施例中，通过上述通用图形控制器完成的功能包括：

建立图文框（即，窗口）；

建立组元 (即, 视图);
传送图象数据到图象再现器;
传送文本到字体再现器;
传送形状到多边形再现器;
允许为图文框和组元选择 (聚焦);
从显示屏中实际删除图文框和组元;
当调用视图中画图方法时来回移动对象层; 以及
通过按下鼠标器和键盘按钮, 借助图文框和组元链操纵鼠标器和键盘事
件。

通过窗口管理器 254 完成的功能包括:
利用在多边形再现器中的多边形制图拉出到视窗上;
管理多个窗口以便它们能够重叠;
确定图文框中已发生事件, 并通知适当的布局管理器实体它可能需要进
行更新; 以及
在每一个图文框中监视各分层的包容文件。

窗口/观察系统还包括一布局管理器 256 和一组元管理器 258。上述布
局管理器属于 (被引用) 一图文框, 并完成以下功能:
在一个图文框的范围内管理各分层的包容文件; 以及
确定包容文件中已发生事件, 如有必要, 指示包容文件进行更新。

通过组元管理器完成的功能包括：

在一个包容文件的范围内管理各分层的组元；以及

确定组元中已发生事件，如有必要，指示组元进行更新。

图 18 是按照本发明的 Java 抽象分屏全套软件工具窗口/包容文件/观察实体分层 290 的实施例框图。结构 292 可包含一个或多个包容文件 294,296。每一包容文件本身又可包含一个或多个不同类型的组元。

图 19 是由本发明支持的 Java.NET 对象的框图。可提供写作工具以允许用户从现成 NET/IO 可重用基本设计结构 154 中，通过汇编先有组元建立 Java.NET 和 Java.IO 可重用基本设计结构的对象和方法。

在本发明的最佳实施例中支持的 Java.NET 分类 320 包括：

数据语法分析信息包 324;

数据语法分析网络界面接口 326;

超文本传输协议统一资源地址（HTTPURL）连接 328;

因特网地址 330;

多目标信息包网络界面接口 332;

服务器网络界面接口 334;

网络界面接口 336;

网络界面接口实现 338;

统一资源地址 (URL) 340;

URL 连接 342;

URL 编码器 344; 以及

URL 数据流处理程序 326。

在本发明的最佳实施例中支持的 Java.IO 分类 322 包括：

缓存输入数据流 348；

缓存输出数据流 350；

字节阵列输出数据流 352；

输入数据流 354；

输出数据流 356；

行输入数据流 358；

打印数据流 360；

<通用化方法> 362；以及

<通用化功能> 364。

在 Java.NET 加速器中省去许多冗余支持对象，并没有导致损失任何联网功能。

图 20 是按照本发明的 Java.NET 加速器的功能框图。将包括 Java.NET 应用程序结构实体和方法的一个线程，从 Java 虚拟机 16 并通过 Java.NET 软件占位程序 52 卸载到硬件对象管理系统 22。寄存器接口程序 64 监控并同步化地传送到在主机 CPU 62 与上述硬件对象管理系统之间的所有输入和输出。

Java.NET 对象请求被接收并存储在 输入/输出 请求队列 108 中待决处理。接着任务处理器 150 将每一请求引导到硬件 NET 加速器 370。该

加速器包括联网系统 372 和连接驱动程序 382。上述连接驱动程序利用，举例来说，如 IP 384, TCP/IP 386, 以及 UDP 388 这样的网络通信协议，执行请求以建立，保持，或终止网络连接。

在网络控制器 374 中完成用于 Java.NET 加速器的微代码。窗口/观察系统还包括网络查询机制 376, 网络界面接口管理器 378 和数据流管理器 380.

图 21 是按照本发明在 Java.NET 加速器窗口/观察系统 372 内部的功能性说明。网络控制器 374 的功能包括：

 信号 DNS 查询；

 信号网络界面形成，允许多路同时的网络界面；

 导通网络输入和输出；

 信号关闭网络界面；以及

 以抽象分层形式通过 TCP 和 UDP 协议运行。

网络控制器可同步地或非同步地运行。网络查询机制 376 完成 DNS 查阅并把结果报告到网络控制器。

网络界面管理器 378 管理正由应用程序使用着的网络界面。该项管理包括打开或终止网络界面的连接。上述网络界面管理器还确定包容文件中已发生事件，如有必要，通知包容文件进行更新。当被请求时，将网络界面状态传送到上述网络控制器。

数据流管理器 380 通过 TCP/IP 或 UDP 链路穿梭传送基于字符/数据流的输入或输出。上述数据流管理器提供一种不同于应用程序使用的输入/输出数据流的选择。此外，当被请求时，该数据流管理器把缓存区转换到网

络控制器。

虽然此中依照最佳实施例描述了本发明，但是熟悉本技术领域的人士很容易懂得，其它应用可能替代这些陈述而没有脱离本发明的实质和范围。

例如，在本发明的最佳实施例中，不管支持应用程序结构的数量有多少，在 ASIC 上只需要一个硬件对象管理系统。然而，在另外一些实施例中，提供多个硬件对象管理系统。举例来说，当每个硬件组件有一专用硬件对象管理系统时，可为每个支持应用程序结构提供单独的硬件组件，例如，一块 ASIC。

可以将本发明构成计算机系统的一部份，或者也可以构成可插入的设备，例如一 PCI 卡的一部份。

熟悉本技术领域的人士将能够很容易地应用众所周知的编程技术和设备，来构造本发明所要求的硬件和软件。

因此，应当仅用以下包含的权项来限制本发明。

说 明 书 附 图

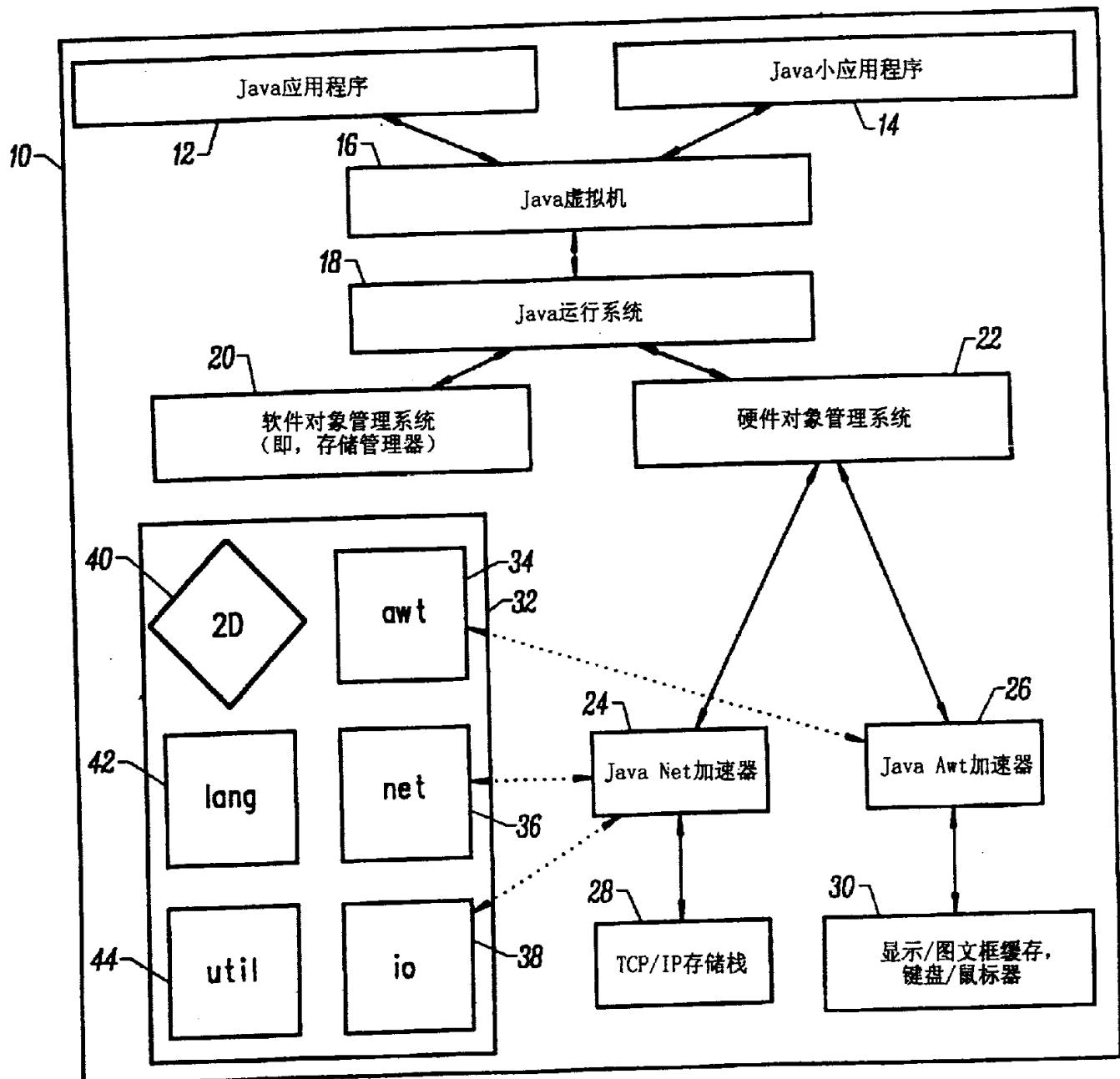


图1

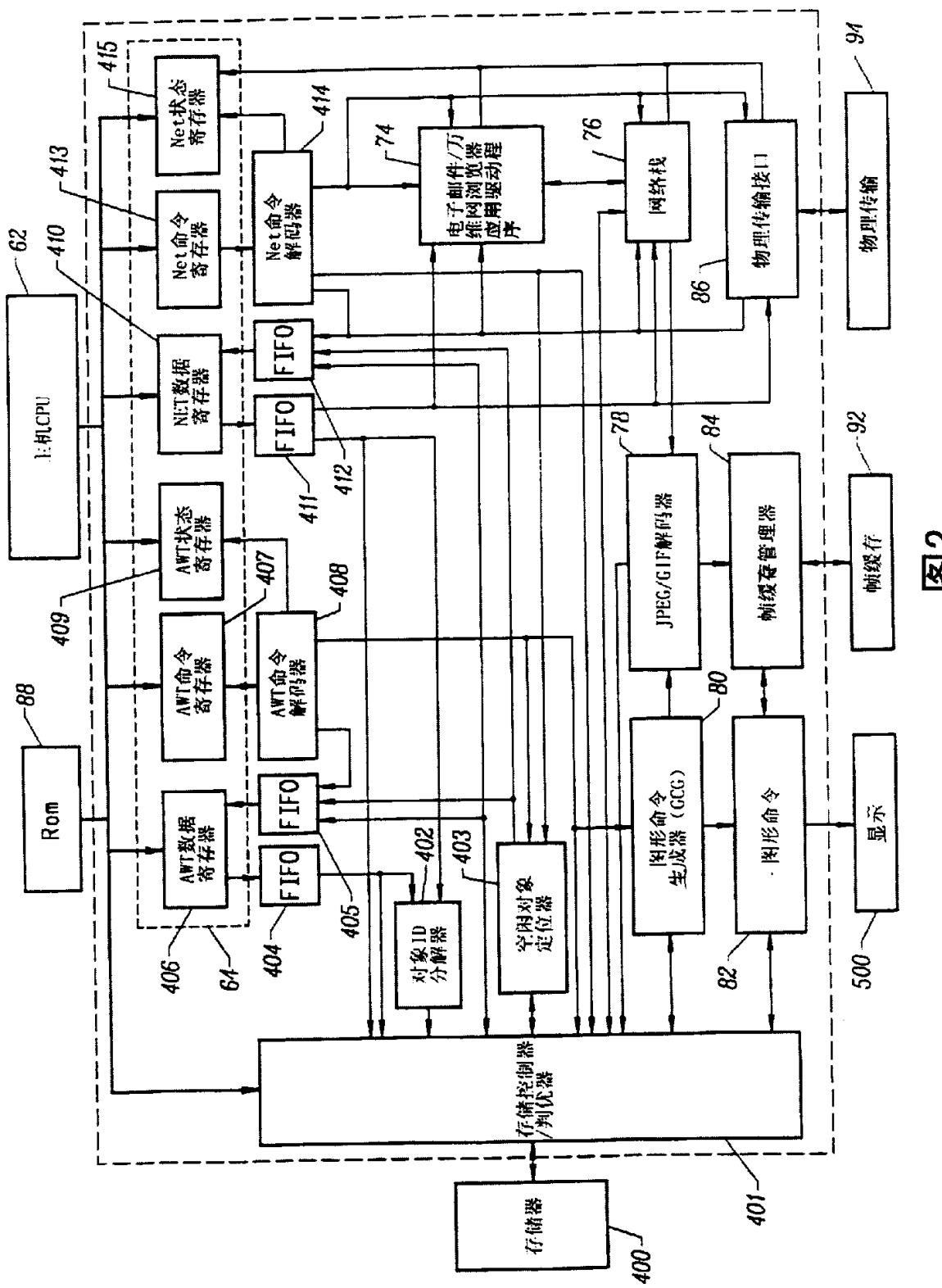
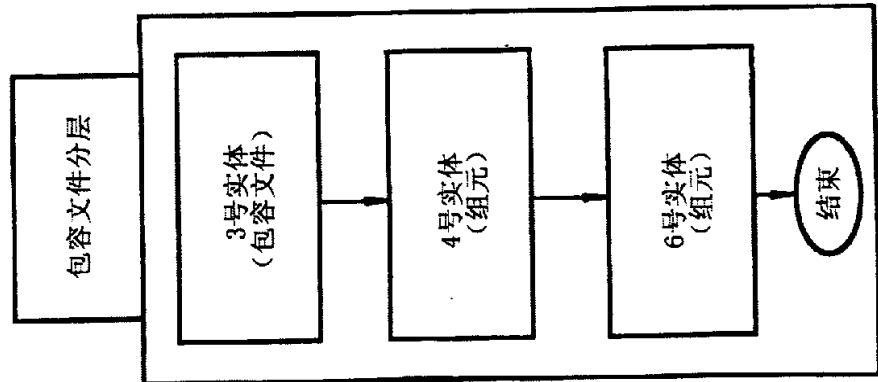
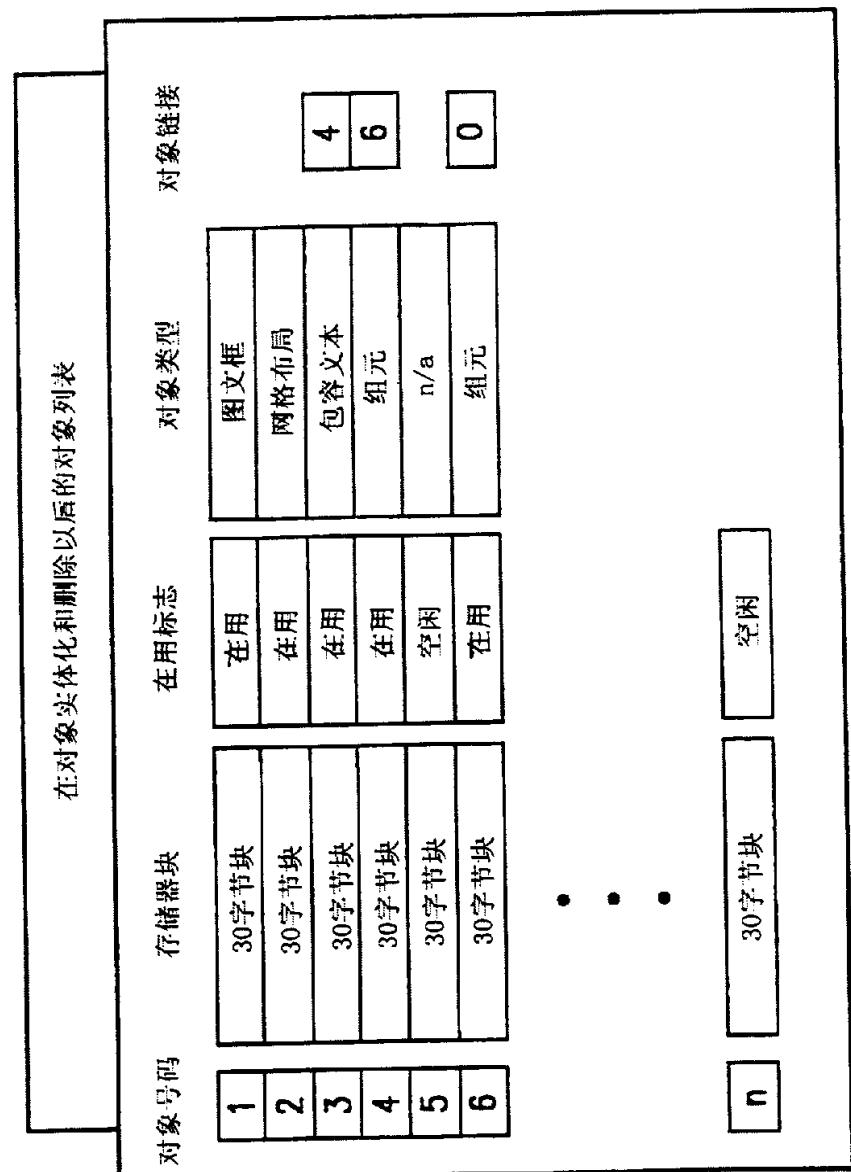


图2

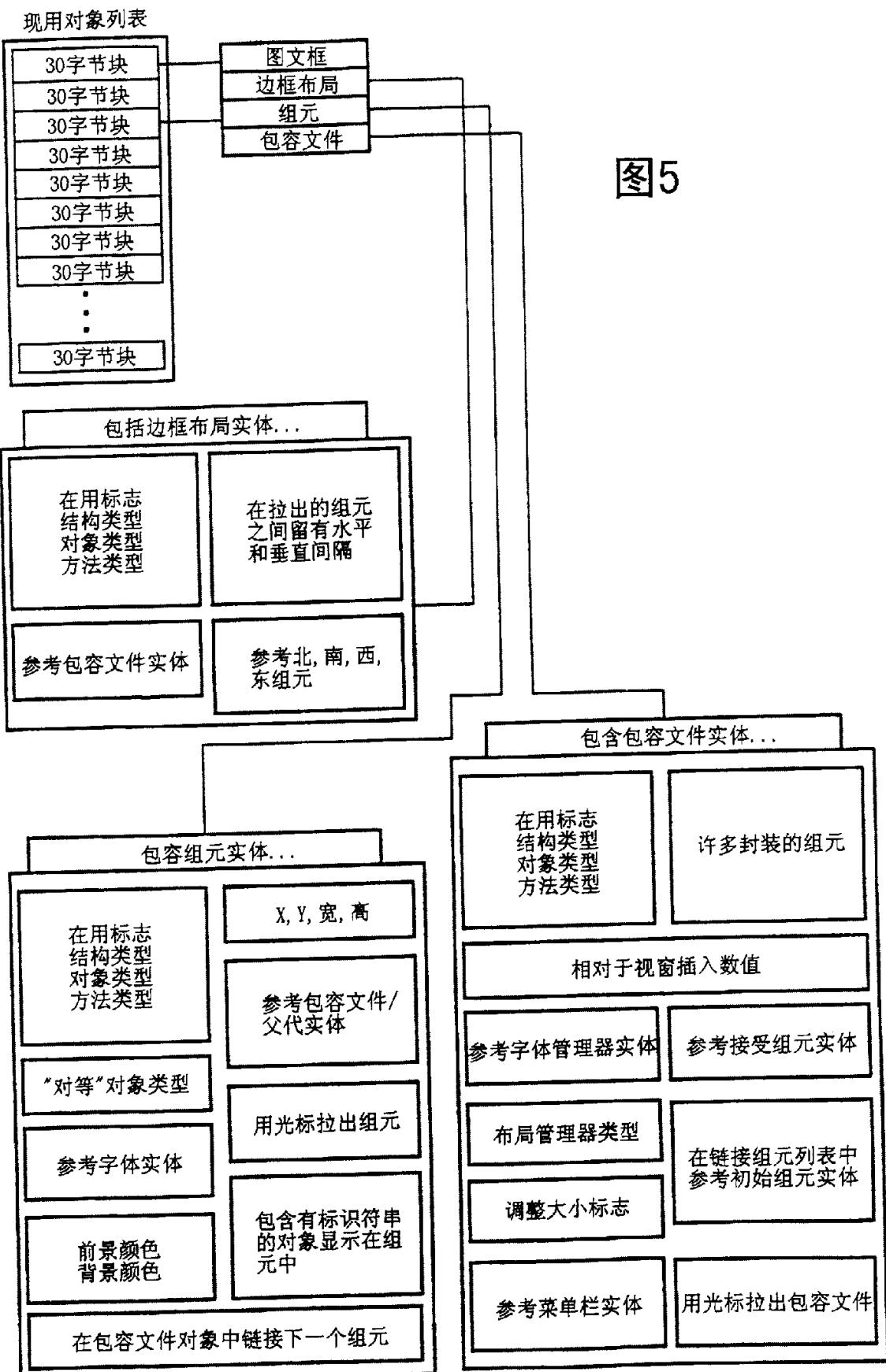


4



3

图5



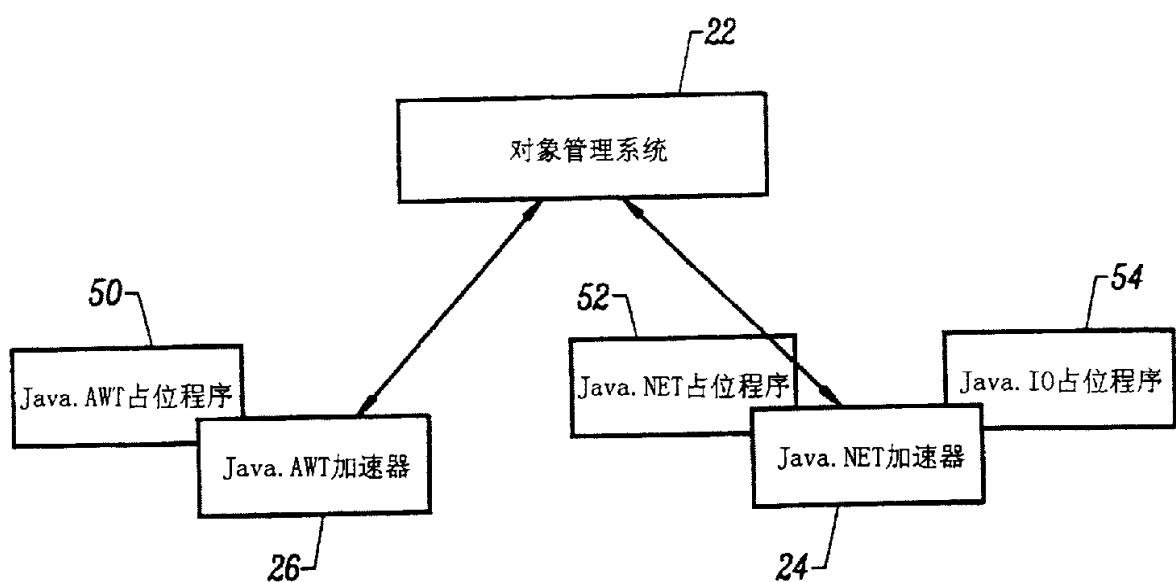


图6

图 7

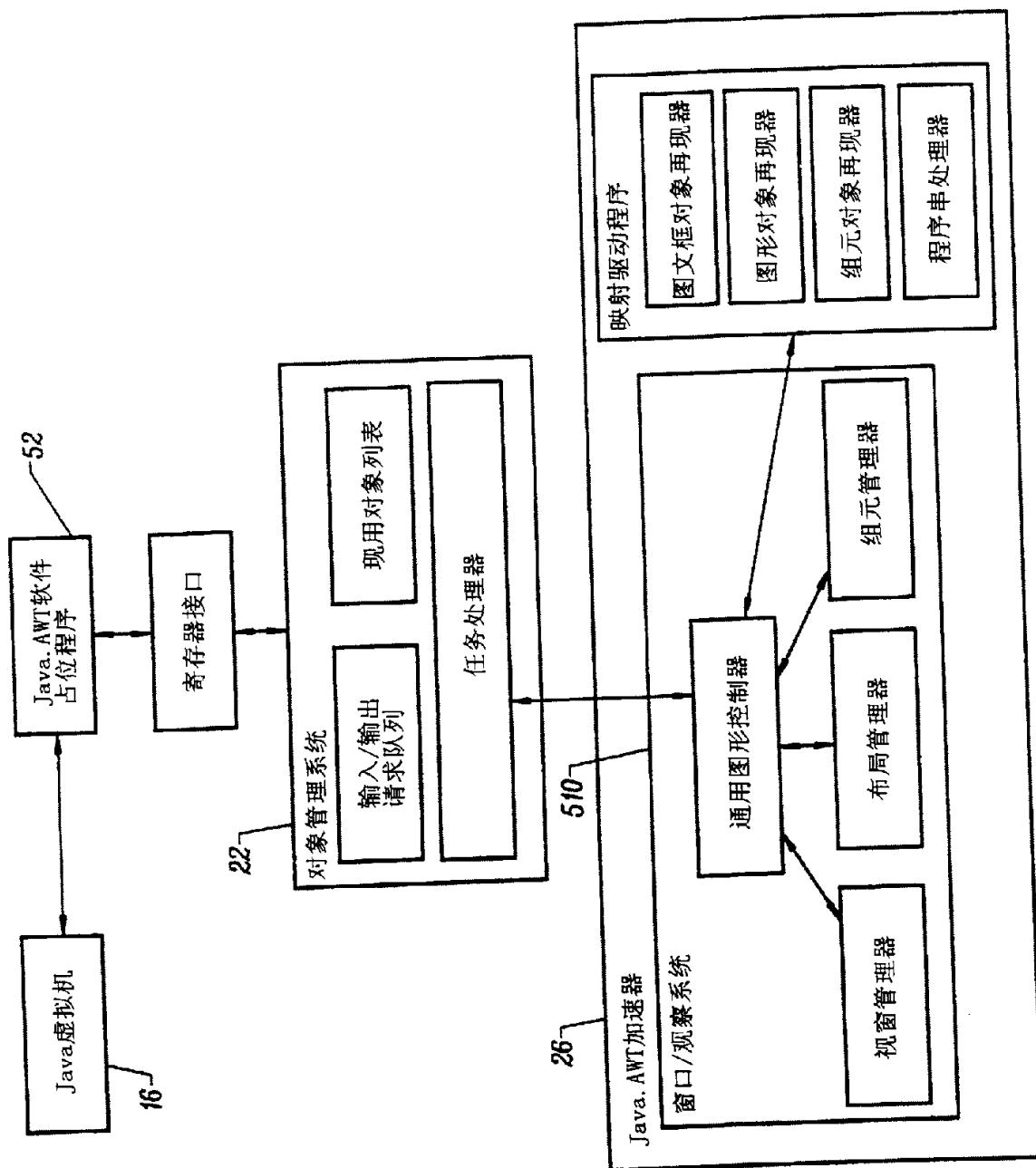
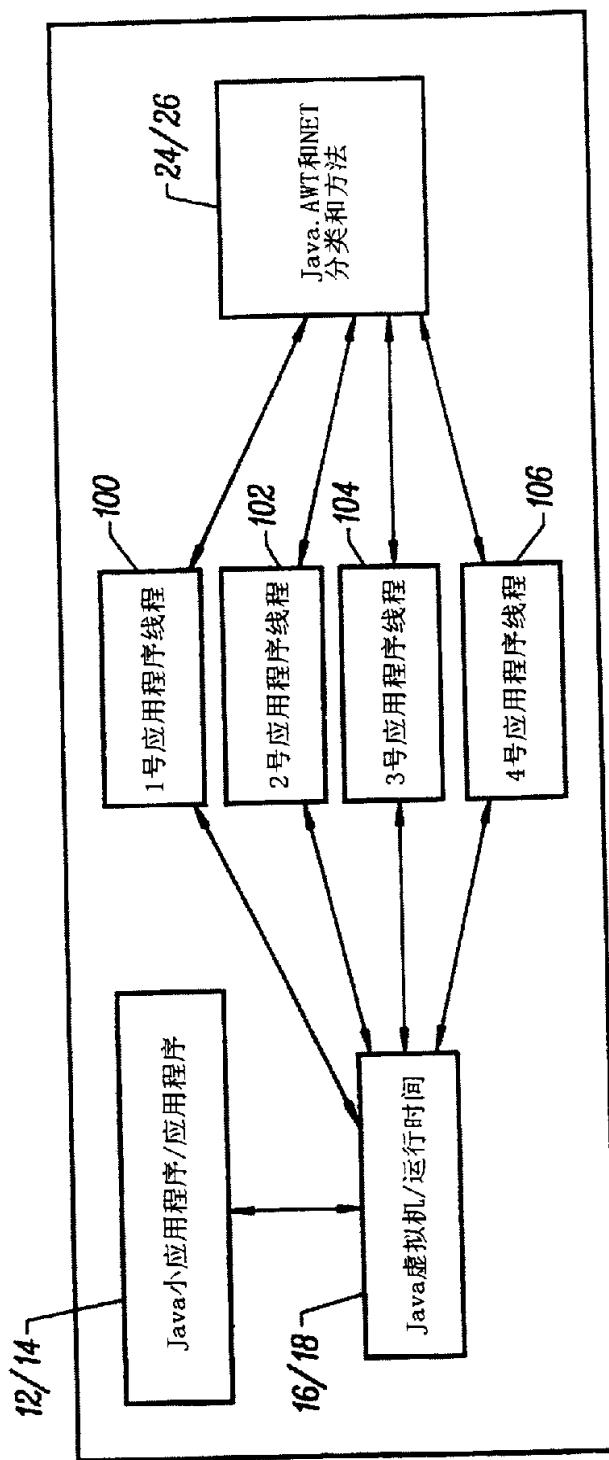
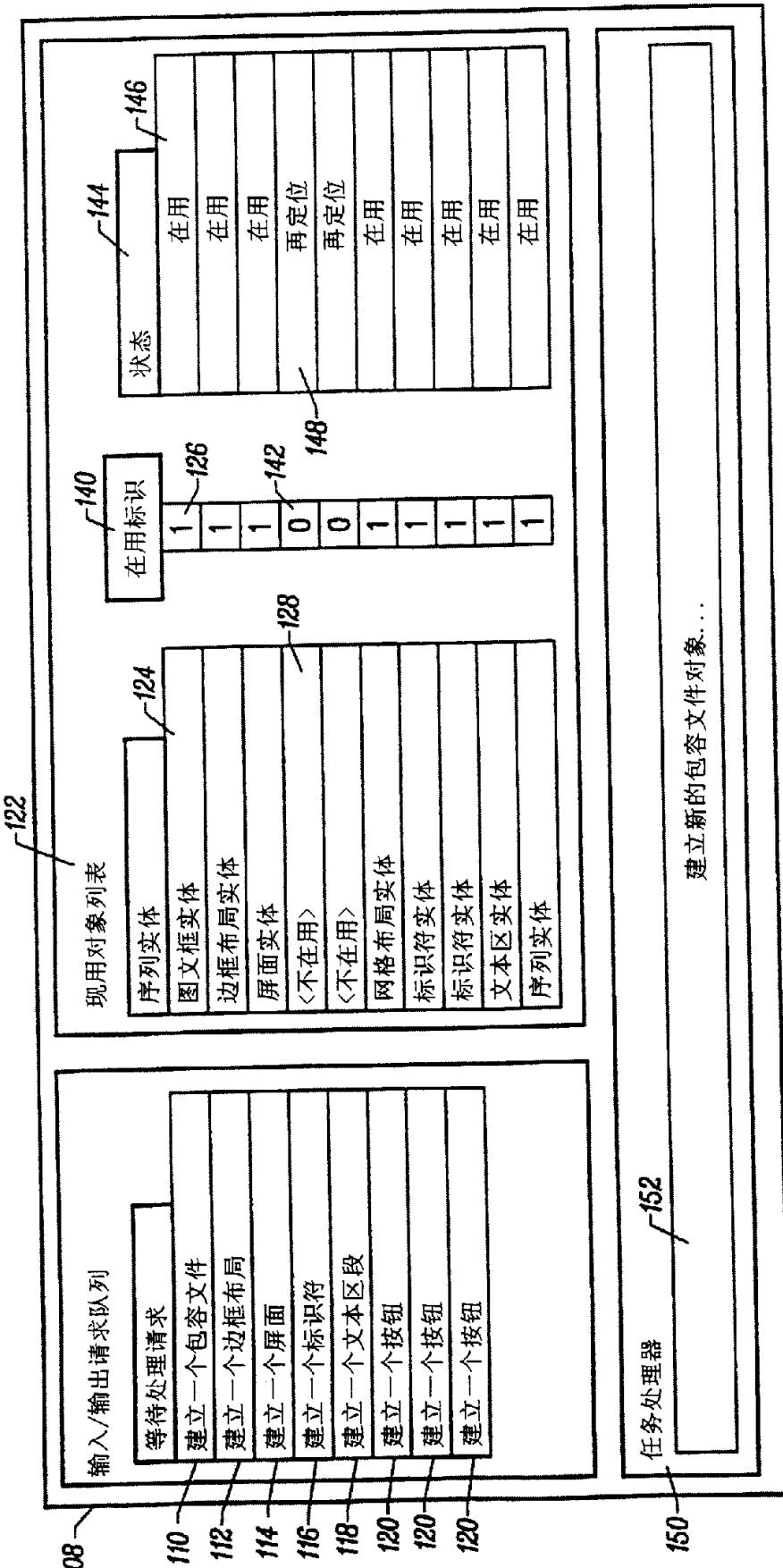


图8





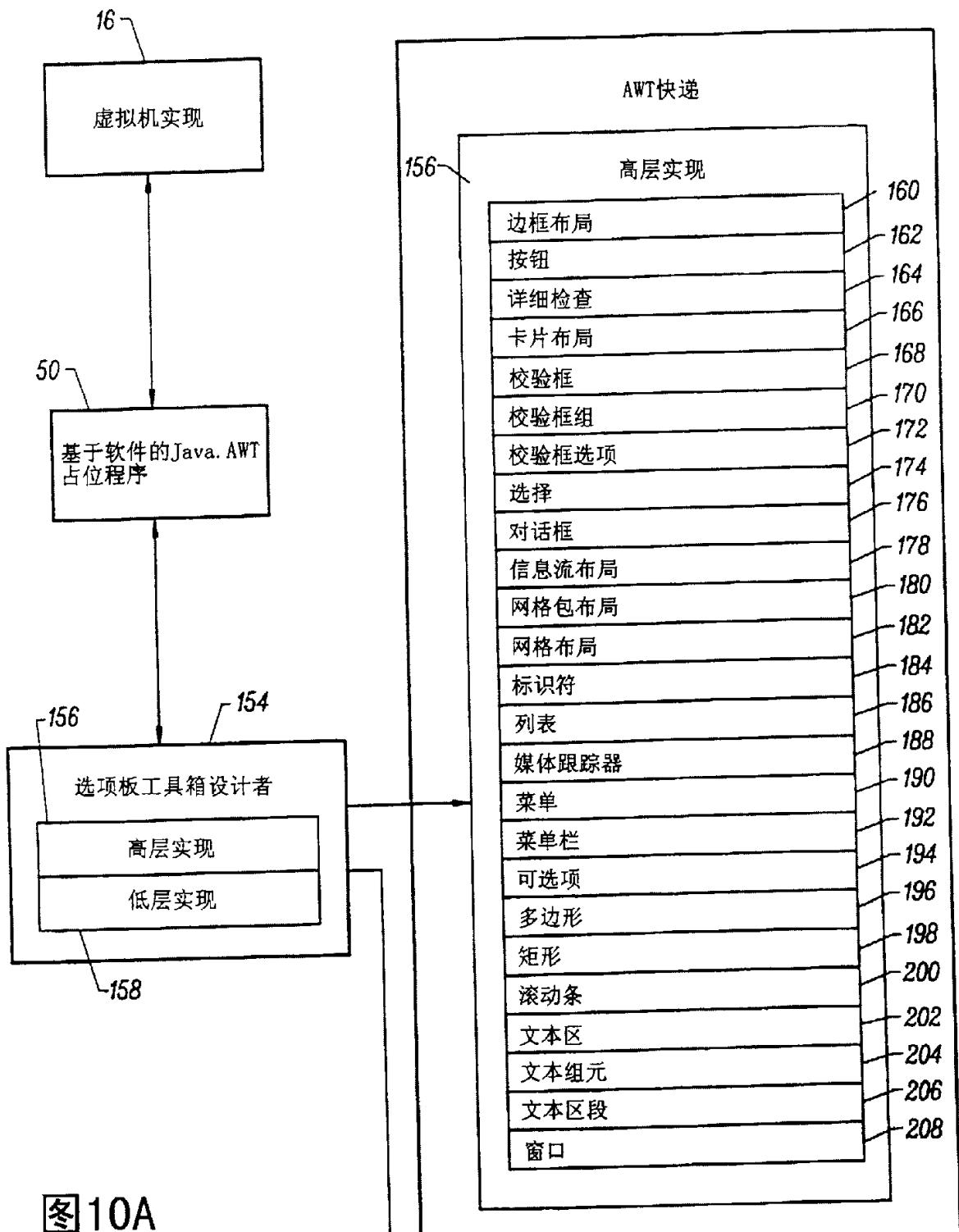


图 10A

看图 10B

看图10B

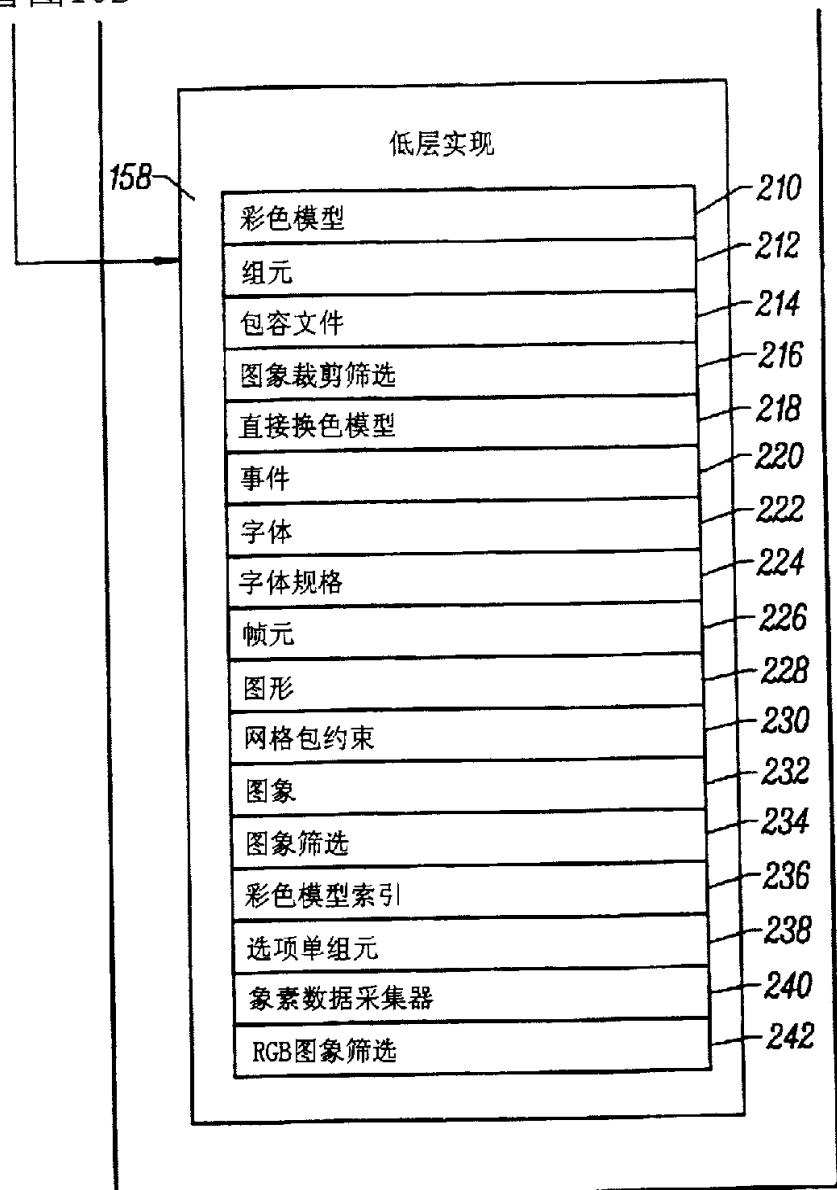
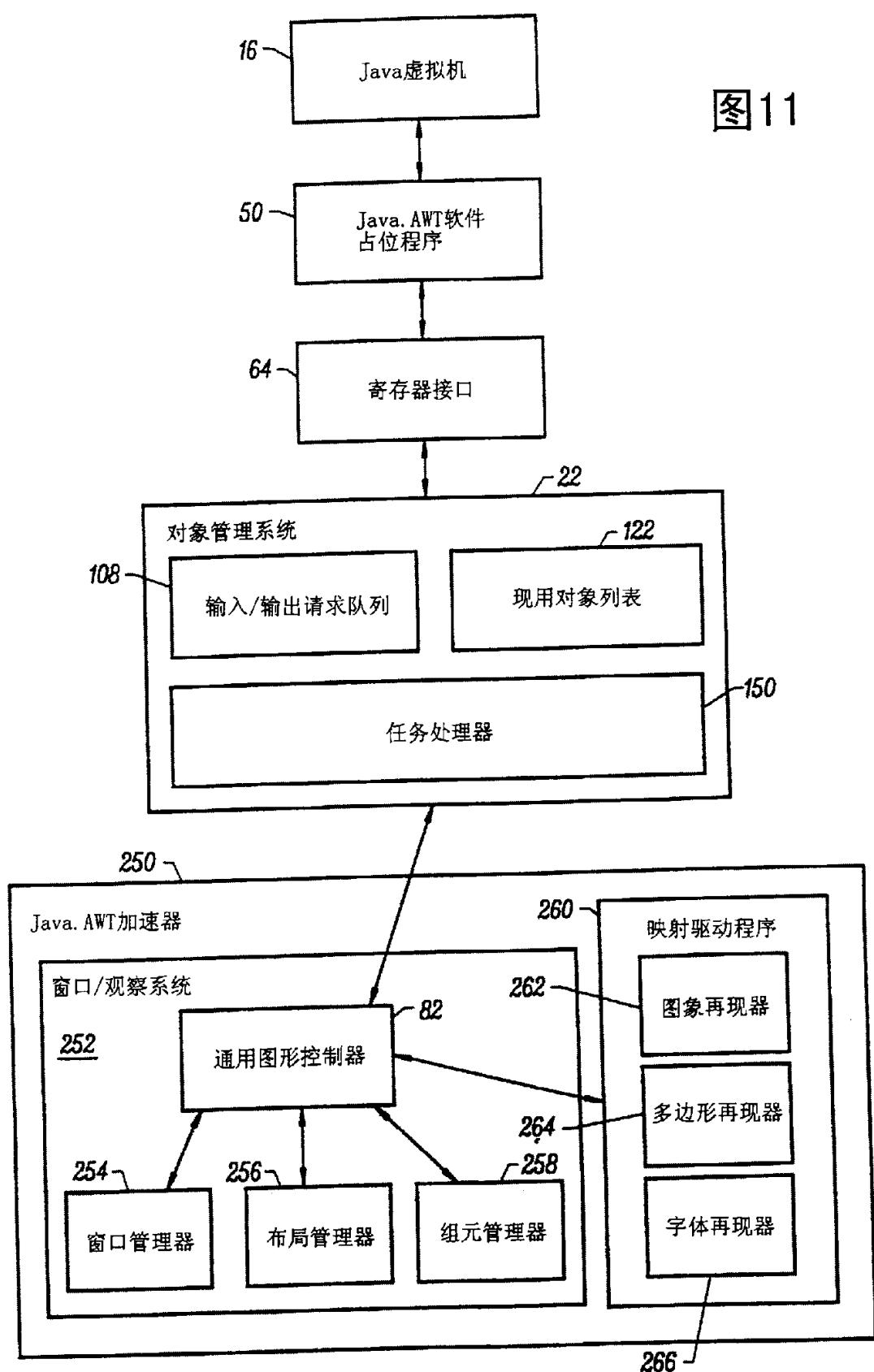
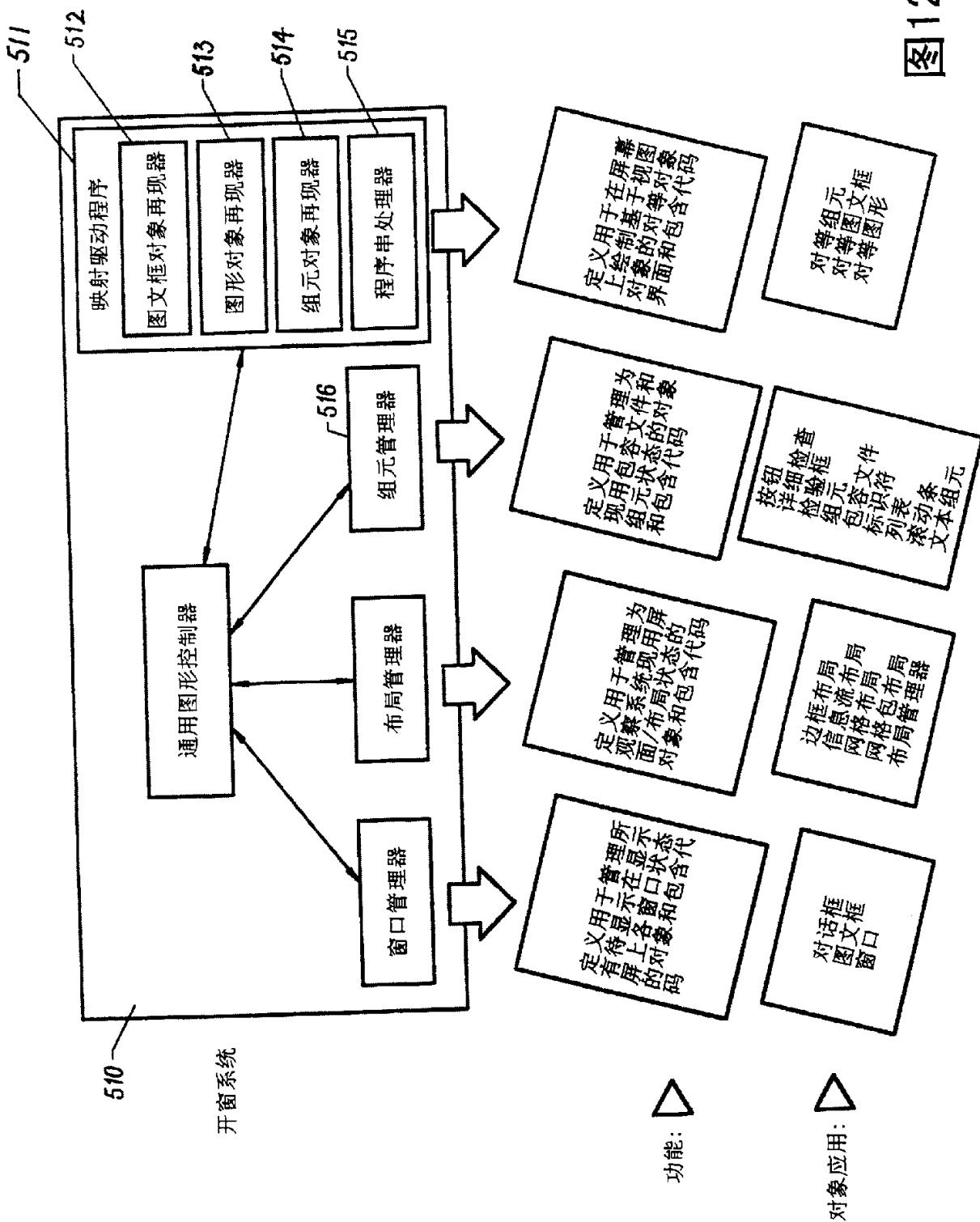


图10B

图 11





由映射驱动程序支持的组元

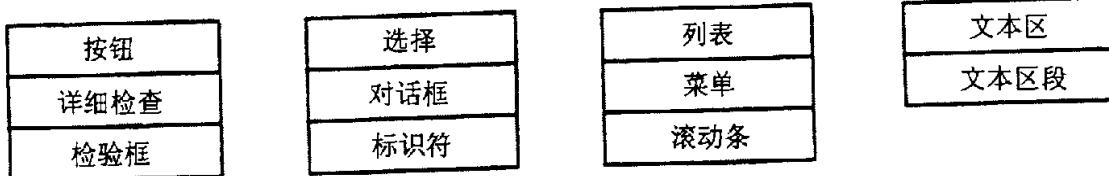
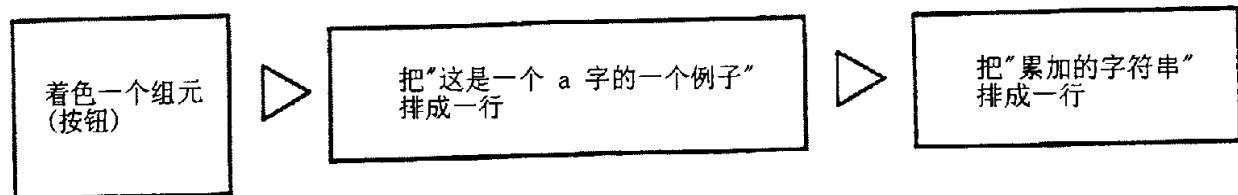


图 13

命令：



结果：

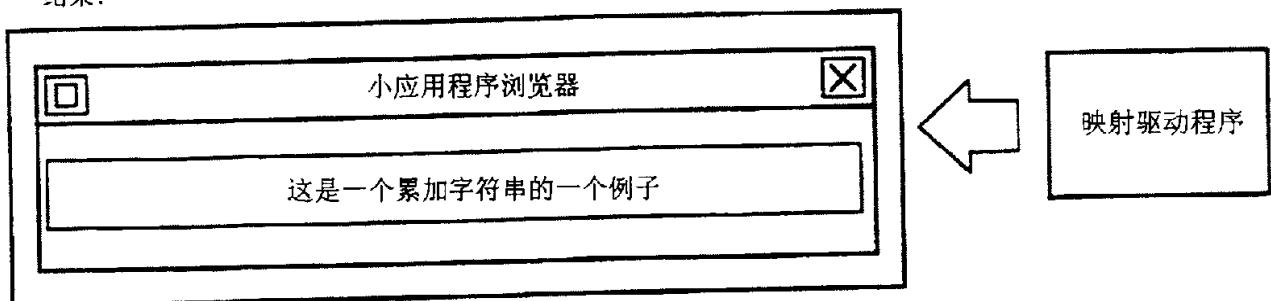


图 14

画图方法：

清除矩形
剪贴矩形
复制器
画三维矩形
画弧线
画线条
画椭圆线
画多边形

画矩形
来回画矩形
实心三维矩形
填充圆弧线
填充椭圆线
实心多边形
实心矩形
来回实心矩形

其他方法：

画图象
画字符串
设置颜色

设置字体
设置画图模式
传送

图 15

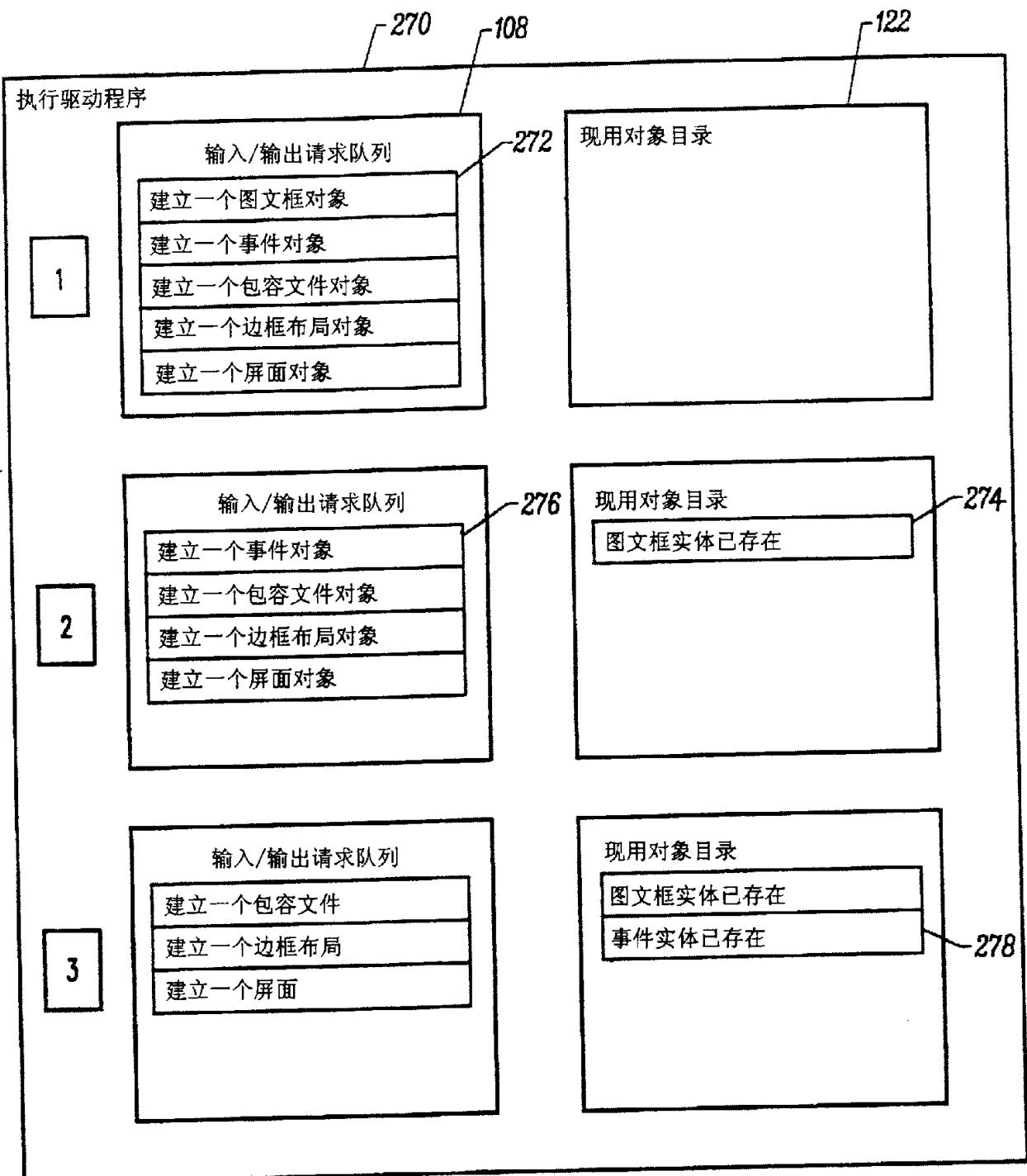


图16

252

窗口/观察系统

252

通用图形控制器

- 建立图文框(即, 窗口)
- 建立组元(即, 视图)
- 传送图象数据到图象再现器
- 传送文本到字体再现器
- 传送形状到多边形再现器
- 允许为图文框和组元选择(聚焦)
- 从显示屏中实际删除图文框和组元
- 当调用视图中的画图方法时来回移动对象层
- 通过按下鼠标器和键盘按钮借助图文框和组元链操纵鼠标器和键盘事件

254

窗口管理器

- 利用在多边形再现器中的多边形制图拉出到窗口上
- 管理多个窗口以便它们能够重叠
- 确定图文框中已发生事件就适当地通知布局管理器实体可能需要更新
- 在每一个图文框中监视各层次的包容文件

256

布局管理器(属于/是引用图文框)

- 在一个图文框的范围内管理各分层的包容文件
- 确定包容文件中已发生事件如需要就通知它更新

258

组元管理器(属于/是引用布局管理器/包容文件)

- 在一个包容文件的范围内管理各分层的组元
- 确定组元中已发生事件如需要就通知它更新

图 17

290

一个图文框的举例说明/包容文件/组元(窗口/包容文件/视图)的分层例子

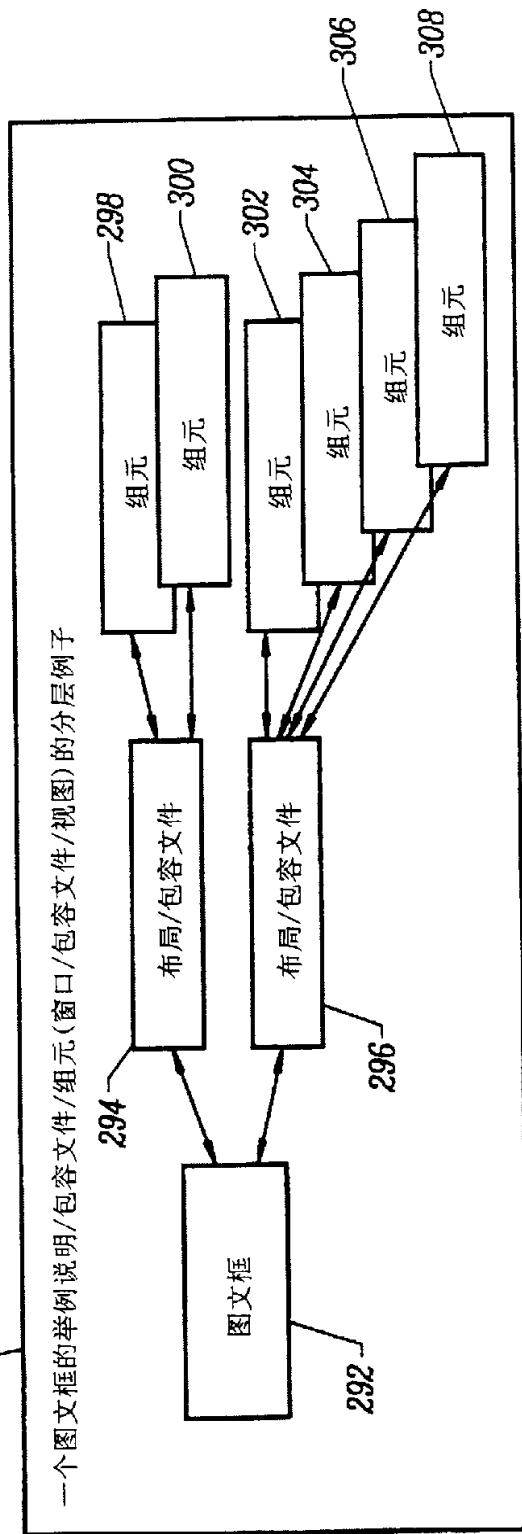


图 18

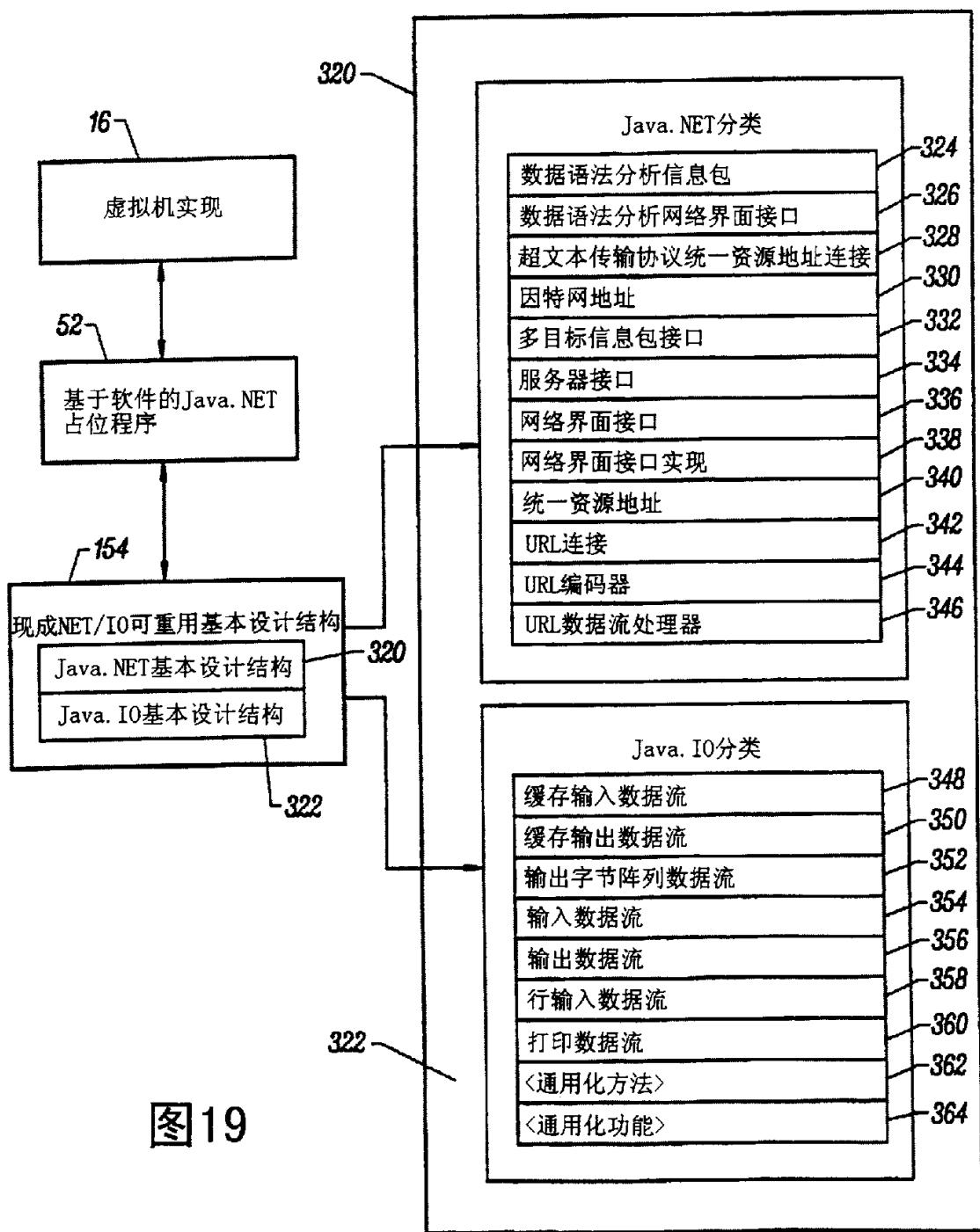
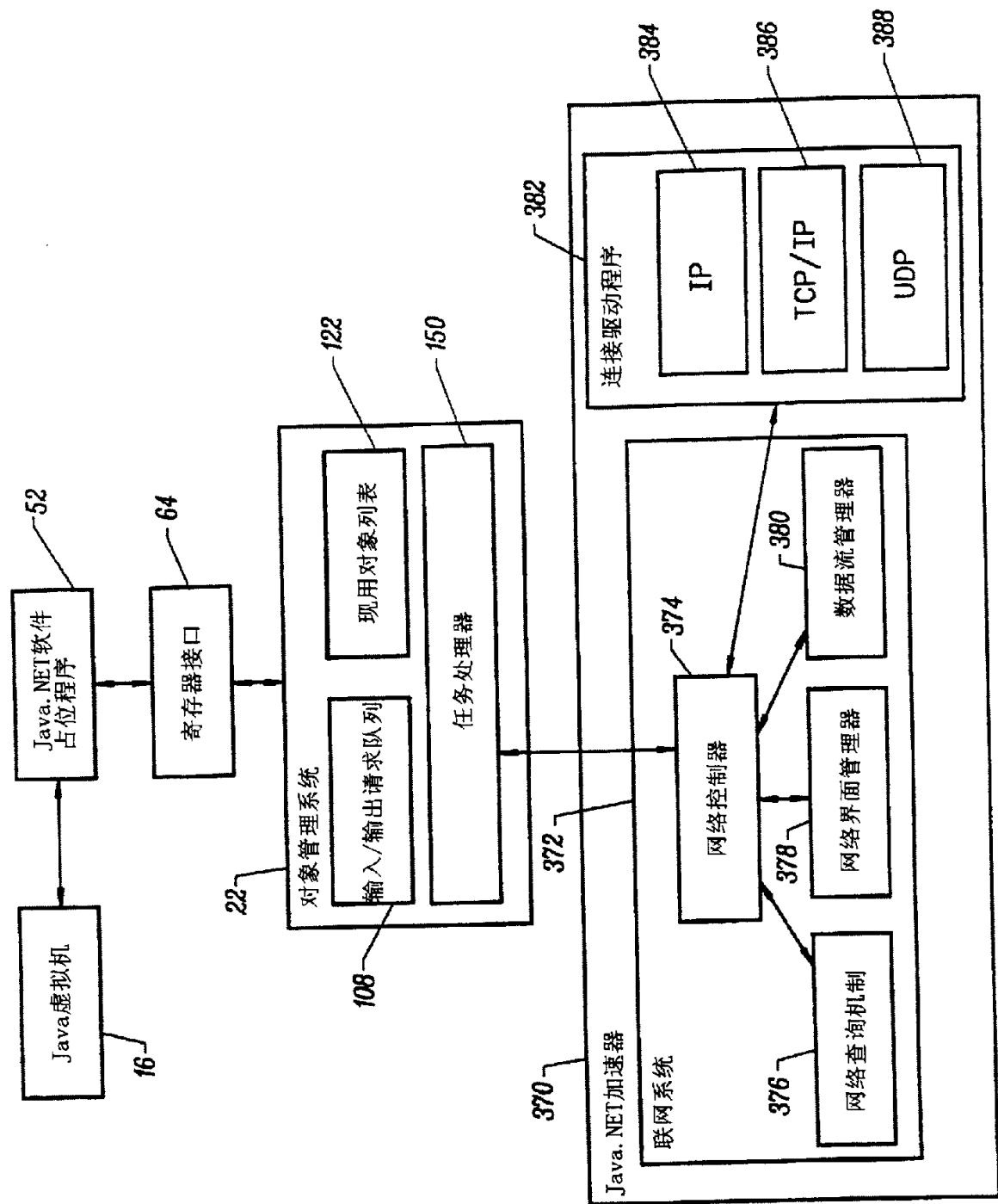


图 19



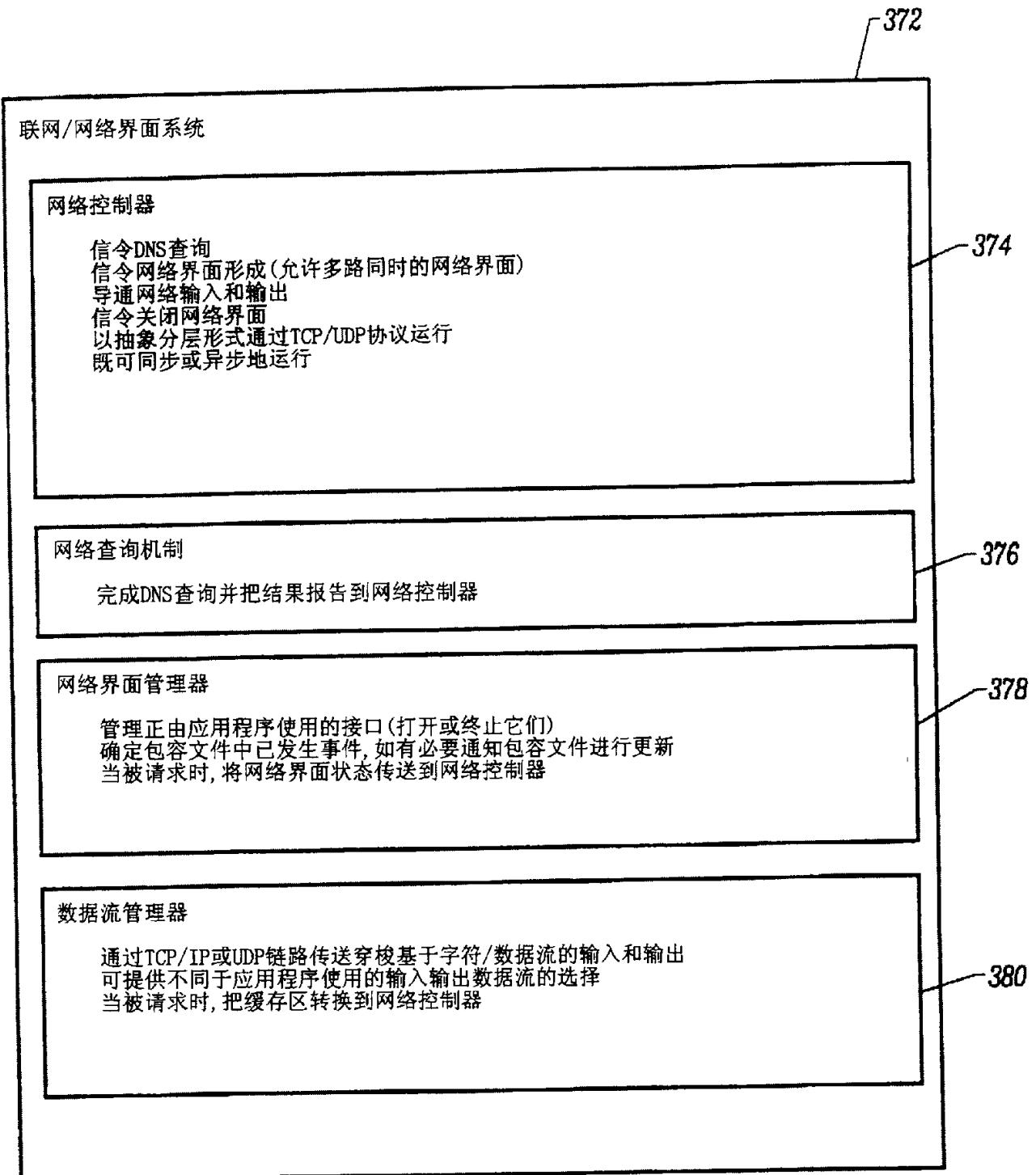


图21