[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 1/00 (2006.01)
H04L 29/06 (2006.01)

[12] 发明专利说明书

专利号 ZL 200380106164.2

[45] 授权公告日 2007 年 12 月 26 日

[11] 授权公告号 CN 100357846C

[22] 申请日 2003.10.3
[21] 申请号 200380106164.2
[30] 优先权
[32] 2002.10.29 [33] US [31] 60/421,773
[32] 2002.10.29 [33] US [31] 60/421,774
[32] 2002.10.29 [33] US [31] 60/421,775
[32] 2002.12.31 [33] US [31] 10/331,879
[86] 国际申请 PCT/US2003/031313 2003.10.3
[87] 国际公布 WO2004/040427 英 2004.5.13
[85] 进入国家阶段日期 2005.6.15
[73] 专利权人 洛克希德马丁公司
地址 美国马里兰州
[72] 发明人 迈克尔·C·达普
埃里克·C·莱特
[56] 参考文献
CN 1310539 A 2001.8.29
US 5319776 A 1994.6.7
CA 2307529 A1 2001.9.29
US 5414833 A 1995.5.9
审查员 王 欢
[74] 专利代理机构 北京英赛嘉华知识产权代理有限责任公司
代理人 余 朦 方 挺

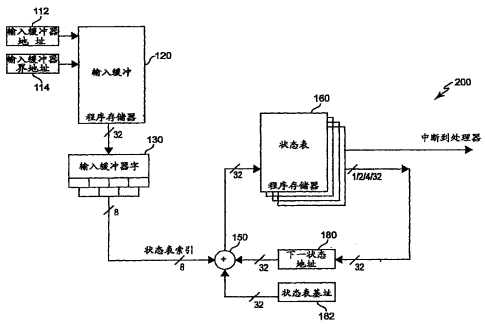
权利要求书 5 页 说明书 16 页 附图 9 页

[54] 发明名称

入侵检测加速器

[57] 摘要

文件中字符的标志能够使用硬件加速器在硬件解析器加速器的环境中被高速检测到，所述标志表示被连接到网络中的计算机系统或其节点的可能入侵或攻击，以及表示其它的安全破坏。 在可以构建上述安全破坏、入侵或攻击的命令被变得可执行之前，通过解析文件中断或异常能够被发送到主机 CPU。 CPU 能够发起网络控制措施来防止或限制入侵。



1. 一种入侵检测系统，包括:

字符缓冲器，用于存储文件的多个字节;

状态表，能够根据所述文件的字节并结合状态来寻址，以从所述状态表中访问中断、异常或用于存储令牌的命令以及下一状态数据中的至少一个，其中，当在所述状态表中解析出表示有效的令牌时，访问所述用于存储令牌的命令;

寄存器，用于存储所述下一状态数据;

用于将所述寄存器的内容与所述文件的后续字节合并起来以形成进入所述状态表的进一步地址的装置;

令牌缓冲器，用于存储多个令牌，其中所述多个令牌可被主机处理器用于进一步处理; 以及

总线，用于将所述中断或所述异常传递到所述主机处理器;

其中，所述状态表、所述用于合并的装置、以及所述令牌缓冲器可并行地同时进行操作。

2. 如权利要求 1 所述的入侵检测系统，其中，所述入侵检测系统由解析器提供。

3. 如权利要求 1 所述的入侵检测系统，其中，所述状态表由存储器提供，所述存储器与所述寄存器和所述用于合并的装置中的至少一个处于相同的芯片中。

4. 如权利要求 2 所述的入侵检测系统，其中，所述状态表由外部存储器提供。

5. 如权利要求 4 所述的入侵检测系统，还包括存储器，所述存储器与所述寄存器和所述用于合并的装置中的至少一个处于相同的芯片中，用于在当所述状态表无需在所述外部存储器中实现时存储所述状态表。

6. 如权利要求 1 所述的入侵检测系统，其中，所述状态表被以大于网络数据包传输速度的速度访问。

7. 如权利要求 1 所述的入侵检测系统，还包括用于响应检测到输入序列的出现而将模式匹配报警送到所述主机处理器的装置，所述输入序列与在所述状态表中被编码的一个或多个序列的标志相匹配。

8. 如权利要求 7 所述的入侵检测系统，其中，响应于所述中断或所述异常的入侵报警被传递到所述主机处理器来启动入侵防护操作，以防止或限制入侵攻击。

9. 如权利要求 1 所述的入侵检测系统，其中，所述状态表被与网络数据包传输速度相等的速度访问。

10. 一种入侵检测方法，包括:

访问能够根据文件的字节和当前状态来进行寻址的状态表;

如果中断或异常可用，则从所述状态表中取得所述中断或所述异常中的至少一个;

如果确定没有中断或异常可用、并且已经对有效令牌进行了分析，则从所述状态表中取得令牌存储命令;

响应于所述令牌存储命令在令牌缓冲器中存储令牌;

从所述状态表中取得下一状态数据;

存储所述下一状态数据; 以及

将所述被存储的下一状态数据与所述文件的后续字节合并起来以形成进入所述状态表的进一步地址。

11. 如权利要求 10 所述的入侵检测方法，其中，所述入侵检测方法由解析器提供。

12. 如权利要求 11 所述的入侵检测方法，其中，所述状态表由外部存储器提供。

13. 如权利要求 10 所述的入侵检测方法，其中，所述状态表被以大于网络数据包传输速度的速度访问。

14. 如权利要求 10 所述的入侵检测方法，还包括当检测到与在所述状态表中编码的一个或多个序列的标志相匹配的输入序列时，将模式匹配报警送到所述主机处理器。

15. 如权利要求 14 所述的入侵检测方法，其中，相应于所述中断或所述异常的入侵报警被通信到所述主机处理器以启动入侵防护操作，从而防止或限制入侵攻击。

16. 如权利要求 10 所述的入侵检测方法，其中，所述状态表被与网络数据包传输速度相等的速度访问。

17. 一种使得计算机可以加速入侵检测的方法，包括如下步骤：

访问能够根据文件的字节和在先状态来进行寻址的状态表；

如果中断或异常可用，则从所述状态表中取得所述中断或所述异常中的至少一个；

如果命令可用、并且已经对所述令牌进行了全面分析，则从所述状态表中取得令牌存储命令，并且响应于所述令牌存储命令存储所述令牌；

从所述状态表中取得下一状态数据；

存储所述下一状态数据；

将所述被存储的下一状态数据与所述文件的后续字节合并起来以形成进入所述状态表的进一步地址；以及

在已经分析和存储了所述令牌之后，使得所述令牌可用于不同目的的后续处理。

18. 如权利要求 17 所述的方法，其中所述不同的目的是进行上下文分析以检测文件水平上的入侵。

19. 如权利要求 17 所述的方法，其中所述不同的目的是所述文件的最终用途。

20. 如权利要求 17 所述的方法，其中所述不同的目的与入侵检测无关。

21. 一种入侵检测方法，包括:

访问状态表，所述状态表能够根据文件的第一部分和当前状态来进行寻址;

在命令可用时，从所述状态表中取得所述命令，并且响应于存储令牌的命令存储令牌;

从所述状态表中取得下一状态数据;

存储所述下一状态数据;

将所述被存储的下一状态数据与所述文件的第二部分合并起来，以形成进入将所述状态表的进一步地址;

同时地并行执行以下功能，即访问所述状态表、存储所述令牌、以及将所述被存储的下一状态数据与所述文件的第二部分合并；以及

将所述中断或所述异常传送到所述主机处理器。

22. 如权利要求 21 所述的入侵检测方法，其中所述入侵检测方法由解析器提供。

23. 如权利要求 21 所述的入侵检测方法，还包括当检测到与在所述状态表中编码的一个或多个序列标志相匹配的输入序列时，将待被提出的模式匹配报警送到所述主机处理器，以增加响应速度。

24. 如权利要求 21 所述的入侵检测方法，其中，相应于所述中断或异常的入侵报警被通信到所述主机处理器以启动入侵防护操作，从而防止或限制入侵攻击。

25. 如权利要求 21 所述的入侵检测方法，其中，所述第一部分和所述第二部分代表字符。

入侵检测加速器

说明

发明背景

发明领域

本发明一般涉及对诸如 XML™ 文件的文件解析（parsing），尤其涉及对网络数据包文件或其它逻辑序列进行解析以用于检测对网络节点的潜在入侵或攻击。

现有技术的描述

近几年来，在计算机和将计算机连接到网络中的链路之间的数字通信领域得到了迅速的发展，这与几年前个人计算机的迅速增长在许多方面都是相类似的。这种在远程处理的互连性和可能性方面的增长极大地增加了在上述网络系统中的单个计算机的有效性能和功能。然而，单个计算机和系统的用途以及当计算机被置入服务时它们的用户的喜好和技术状态这些方面的多样性已经造成了个人机器及它们的操作系统的性能和配置产生了基本程度的变化，个人机器及它们的操作系统共同地被称为“平台”，这些平台通常在某些程度上相互不兼容，尤其是在操作系统和程序语言的层面上。

平台特性和对通信能力和远程处理能力以及为了支持它的足够程度的兼容性的同时需求的这种不兼容已经导致了面向对象程序设计的发展（其接纳了这样一种概念，即，通过实体、属性和关系的参考系统将应用程序以及数据集合为一组或多或少通用的模块）和大量体现这种概念的程序设计语言。可扩展标记语言 ™（XML™）就是这样的语言，它其已经得到广泛的使用，并且能够作为文件而在任意构造和体系结构的网络上传输。

在这样的语言中，某些字符串与某些指令或标识符相对应，包括特

殊字符和其它重要的数据（它们集合起来被称作控制字），其允许数据或操作有效地进行自我识别以便它们可以在随后被当作“对象”，这样，相关的数据和命令可被以不同语言翻译成不同应用程序的适当格式和命令，从而产生各个相连的平台的足以支持在给定机器处的期望的处理的一定程度的兼容性。这些字符串的检测通过称为解析（parsing）的操作来实现，这与将表达式（例如句子）的语法分解为其组成部分以及对其进行语法描述的更常规的应用类似。

当解析 XML™ 文件时，大部分和可能主要的中央处理器（CPU）执行时间被用来遍历该文件搜索控制字、特殊字符和其它被定义作为正在被处理的具体 XML™ 的规范的重要数据。此举典型地通过软件来完成，该软件查询每个字符并确定它是否属于预定义的所关心字符串的集合，例如，一组包括“<command>”、“<data type = dataword>”、“</command>”等的字符串。如果目标字符串的任意一个被检测到，则令牌被保存，该令牌具有在文件中指向用于令牌的起点和长度的位置的指针。这些令牌被积累直到全部文件被解析。

传统的实现方式是执行基于表格（table）的有限状态机（FSM）来搜寻所要关心的那些字符串。状态表驻留于存储器中并被设计以在文件中搜寻特殊的模式。当前状态被用作状态表中的基址，输入字符的 ASCII 表示（representation）是该表的索引。例如，假定上述状态机处于状态 0（零）并且第一输入字符的 ASCII 值为 02，则状态条目的绝对地址将是基址（状态 0）和索引/ASCII 字符（02）的和/级联（concatenation）。FSM 从 CPU 从存储器中取出输入文件的第一个字符开始。接着 CPU 在存储器中的状态表中构造相应于初始化的/当前状态和输入字符的绝对地址，接着从该状态表中获取状态数据。基于被返回的状态数据，如果不同（表示该字符相应于所关心的字符串的第一字符），CPU 更新当前状态到新的值，并执行在状态数据中指示的其它操作（例如，如果基于前述的进一步重复，单个字符是特殊字符或如果当前的字符被发现是所要关心的串的最后字符，则发出令牌或中断）。

随着所要关心的串的连续字符被找到，上述的处理被重复，状态被改变。也就是，如果初始的字符是所关心的串的初始字符，则上述 FSM

的状态能够被提升为新的状态（例如，从初始状态 0 到状态 1）。如果该字符不是所要关心的，则通过在从状态表地址返回的状态表条目中指定同样的状态（例如，状态 0）或不命令状态更新）而将（通常）状态机保持不变。可能的操作包括但不限制于设置中断、存储令牌和更新指针。对后来的字符随后将重复执行上述处理。应该注意到，在所要关心的串正被跟随以及上述 FSM 处于非状态 0 的状态（或其它表示所要关心的串还没有发现为当前正在跟随的字符的其它状态）时，与当前串不一致但是所要关心的其它串的初始字符的字符可被发现。在这样一种情况下，状态表条目将指示适当的操作以指示和识别先前被跟随的串片段或部分，并跟随所要关心的可能的新串直到该新串被彻底识别或被发现为不是所要关心的串为止。换句话说，所关心的串可以是嵌套的，状态机必须能够在所要关心的另一个串内检测出所要关心的串，等等。这就要求 CPU 遍历 XML™ 文件很多次以彻底解析该 XML™ 文件。

全部的 XML™ 文件或其它语言文件通过上述方式被一个字符接一个字符地解析。随着潜在的目标串被认出，上述 FSM 一个字符接一个字符地逐步通过各种状态，直到所关心的串被完全识别出或遇到与所关心的可能的串不一致的字符（例如，在上述串被完全/完整地匹配时，或字符偏离目标串时）为止。在后一种的情况下，除了返回到初始的状态或与其它目标串的初始字符的检测相应的状态外，通常不会发生操作。在前一种情况下，令牌与输入文件中的开始地址以及令牌的长度一起被存储到存储器中。在解析完成时，所有的对象将会被识别出，并且基于本地平台或给定平台的处理可以开始。

由于上述搜寻通常是针对所关心的多个串来执行的，所以状态表能够提供从任意给定状态的多种转换。这种实现方式在便利地适应嵌套串的同时，允许当前的字符被解析以同时用于多个目标串。

从前述说明可以看出，对例如 XML™ 文件的文件解析需要许多重复和用于各次重复的许多存储器存取。因此，在通用 CPU 中的处理时间是最基本必需的。而且，处理多个串的另一个主要的复杂性在于，它需要产生大的状态表，并且是在脱离实时数据包处理的情况下处理。然而，这就需要大量的 CPU 周期获取输入的字符数据、获取状态数据、更新用

于文件中各个字符的各种指针和状态地址。这样，解析例如 XML™ 文件的文件以完全抢先（pre-empt）CPU 或平台中的其它处理，并基本地延迟所需的处理是相对普通（common）的。

在本领域中应该认识到，通过程序编制能够使得通用的硬件模仿（emulate）专用硬件的功能，专用的数据处理硬件通常比被程序编制的通用硬件运行更快，即便是它们的结构和程序精确地相互一致，这是因为需要更少的开销来管理和控制专用硬件。然而，专用硬件的需要用于特定处理的硬件资源可以大的惊人（prohibitively），尤其是在处理速度的增长可能较小的情况下。而且，专用硬件必需具有功能限制，并为特定的应用提供足够的灵活性，例如，提供搜寻字符的任意数目的任意组合的能力也可以是有限制的。这样，为了切实可行，在提供非常充分的硬件节约的同时专用硬件在处理速度中必须提供大的增长；在所需的处理功能中，由于增长的功能灵活性或可程序编制性而愈加难以同步适应（accommodate）的需求是需要的。

在这点上，互相连接能力和需要用于解析例如 XML™ 文件的文件处理时间量还引出了系统安全问题。一方面，以相对高的优先级进行的、需要非常多的处理时间的任何处理在某些方面相似于在系统或其节点中的拒绝服务（DOS）攻击，或能够是在这样一种攻击中使用的工具。

为了有敌意地消耗并最终使可用资源过载，频繁的 DOS 攻击表现为琐碎或不良的系统服务请求。硬件加速器的适当配置能够极大地减少或消除可用资源过载的潜在性。此外，在过载时，系统经常发生故障或暴露出脆弱的安全性。这样，消除过载是重要的安全性所要考虑的事情。

而且，由于状态表必须能够在基础层面上（basic levels）包含 CPU 命令，因此，完成解析之前，开始一些处理和执行一些命令是可能的，这在没有对系统的性能进行严格折中（compromise）的情况下是困难或不可能的。简而言之，用于折中安全的潜在性将通过减少用于处理例如 XML™ 解析的处理时间而必要地减少，但用于显著地减少用于上述解析的处理时间的技术尚不能获得。

许多安全系统依靠在非常早的阶段检测有企图的安全破坏（breach），一旦这些安全破坏开始，快速地或通过程序编制的干涉来中断安全破坏

是困难或不可能的。例如，一种安全性能高的系统已被提出，并在美国第 09/973,769 和 09/973,776 号专利申请中被公开，这两个专利都被转让给本申请的受让人。上述申请公开了一种具有两级节间（internodal）通信的系统，一级具有非常高的速度，以此，被检测到的受到攻击或入侵的节点能够被隔开（compartmentalize），并在被重新连接到网络之前被自动地修复（如果有必要）。因此，解析的加速早期地支持对潜在攻击的响应，并在系统（例如上述并入的发明申请中描述的系统）中尤其具有有益的效果，这是因为网络的适当控制可以作为解析事件而被启动，并且如果解析被显著地加速的话就能够在较早的时间被启动。除了入侵检测外，响应检测报警而及时启动的正确网络控制还能够实现入侵保护。

发明内容

本发明提供了一种硬件解析器加速器，它可用于潜在的实时入侵检测和防护操作，其提供了对文件的解析的极大加速以用于检测网络计算机系统中的可能入侵、攻击或其它的安全破坏的标志（signature），所述对文件的解析以能够适应网络传输数据包的速度的速度来进行。

为了完成本发明的这些目的和其它目的，提供了一种可在文件解析器内实现的入侵检测系统，包括：字符缓冲器，用于文件的多个字节；状态表，能够根据文件的字节和状态来寻址以从所述状态表中存取中断或异常和下一状态数据中的至少一个；寄存器，用于存储所述下一状态数据；累加装置，将所述寄存器的内容与文件的随后数据合并起来以形成所述状态存储器的进一步地址；以及用于将所述中断或异常传输到主机 CPU 的总线。

附图的简要说明

前述和其它目标，方面和有益效果在参照附图对本发明的优选实施方式进行详细的描述后将会被更好地理解，其中：

图 1 显示了在文件解析中使用的状态表的一部分；

图 2A 是根据同时提交的相关临时专利申请所述的解析器加速器的高层次示意图；

图 2B 是根据本发明的解析器加速器的高层次示意图;

图 2C 描述了带有主处理器和主存储器的一个本发明的实现方案;

图 3 描述了如图 2A 所描述的优选的字符调色板（palette）的格式;

图 4A 和 4B 描述了如图 2A 描述的本发明优选形式所述的状态表格式和与状态表一起使用的状态表控制寄存器;

图 5A 描述了如图 2A 所示的优选的下一状态调色板格式;

图 5B 描述了与如图 2B 所示的本发明一起使用的优选的状态表条目格式，以及

图 6 是如图 5 所描述的优选令牌格式。

本发明优选实施方案的详细描述

现在参考附图，更具体地是参照图 1，其中显示了对理解本发明有用的状态表的部分表示。应该理解在图 1 中显示的状态表是可用于解析 XML™ 文件的非常小的部分，并在本质上是用于示例性的目的。尽管全部状态表没有物理地存在（至少在本发明显示的表格中），图 1 还是能够被用来便于对公知软件解析器的操作的理解，图 1 中没有部分是属于与本发明相关的现有技术。

应该理解，XML™ 文件在这里被用作能够使用根据本发明的加速器来处理的一种类型的逻辑数据序列的实施例。其它的逻辑数据序列也能够从网络数据包内容构建，例如用于共享服务器计算机执行的用户终端命令串。这些命令串被恶意的用户频繁地产生，并作为长期入侵攻击企图的一部分被发送到共享（shatred）的服务器。根据本发明的加速器适于处理许多这样的逻辑数据序列。

观察到在图 1 中描述的状态表部分的许多条目是复制性的也是有帮助的，对于本发明的理解而言，不再需要使硬件适应于图 1 中表示出的整个状态表是重要的。相反，尽管本发明能够在软件中实现（可能使用专用的处理器），但是根据本发明的硬件需求被充分地限制，用于通过软件来解析的增加处理时间的性能损失（penalty）不会通过任意可能的硬件节约（hardware economy）来评判（justify）。

然而，应该认识到，上述入侵检测系统可应用到任意类型的数字文

档，并且不仅限于可被用来以数据包传输速度或超出数据包传输速度来表述具体应用或数据结构的文本文档或特殊语言，上述传输速度能够适应网络上的实时数据传输，而攻击通常是通过该传输来进行的。这样，本发明还可实现为一种用于仅提供入侵检测的配置结构；在这种情况下将期望以最低的成本实现基本上最适宜的性能。然而，以信号传输速度实现入侵检测的目标还能够被作为解析器加速器操作的特殊模式而获得，在这种模式下，一些操作被忽略以提供更进一步的加速，这些操作通过如下所述的备选状态表存储器结构还可能被增加，该模式在目前被认为是优选的。因此，为了完整性和为了传递本发明提供的有益效果范围的更通透的理解，本发明将在解析器加速器的上下文中描述，即便是上下文比本发明起到用于实时、高速的入侵检测作用的所需装置更复杂。

在图 1 中，状态表被分割成任意数目的行，每一行具有与一种状态相应的基址。基址的行被分割成与被用来表示待被解析的文件中的字符的编码的个数一致的列；在该实施例中，有二百五十六（256）列与用于作为状态表的索引的字符的基础 8 位字节相对应。

注意到显示的状态表条目的几个方面是有帮助的，尤其在传递在图 1 中描述的示例性的状态表的一小部分是如何支持许多字的检测的理解方面：

1. 在显示的状态表中，在用于状态 0 的行中只有两个条目包括不是“位于状态 0”的条目，在被测试的字节与所关心的任意字符串的初始字符不匹配时，该条目保持了初始的状态。提供前进到状态 1 的单个条目对应于这样一种特殊情况下，其中全部所关心的串都以相同的字符开始。可以提供前进到其它状态的任意其它字符将通常但不是必须地前进到除状态 1 以外的状态，但是，对通过其它字符能够被达到的相同状态的进一步参考可被用来例如检测嵌套的串。在{状态 0，FD}处示出的命令（例如，“特殊的中断”）与“位于状态 0”的结合可被用来检测和操作特殊的单个字符。

2. 在高于状态 0 的状态中，“保持于状态 n”的条目是为待要通过一个或多个字符的可能的长时间运行而得到保持的状态而规定的，像通常遇到的情况那样，这种状态可能在例如命令的数值增加中遇到。本发

明提供了这种类型的字符串的特殊处理以提供增强的加速，如在下面将详细描述的那样。

3. 在高于状态 0 的状态，“到达状态 0”表示检测到了从所关心的任意串中区分该串的字符，而不管有多少匹配的字符在先前被检测到，并将解析处理返回到初始/缺省状态以开始搜寻所关心的其它串。（因此，到目前为止，“到达状态 0”条目通常是状态表中出现最频繁和最多的条目）。返回到状态 0 可要求解析操作返回到文件中的在区分字符被检测到时、开始被跟随的字符的后面的字符。

4. 包括“到达状态 0 的命令的条目表示完成了对所关心的完整串的检测。通常，该命令将会是存储一个令牌（带有该令牌的地址和长度），之后，该令牌使得串被作为一个对象来处理。然而，具有“到达状态 n”的命令规定在继续跟随能够潜在地与所关心的串匹配的串的同时，在中间点（intermediate point）发起操作。

5. 为了避免在其上搜寻在所关心的两个串（例如，具有 n-1 个相同的初始字符但第 n 个字符不同的串，或具有不同的初始字符的串）之间出现分枝的任意点处出现不确定性（ambiguity），通常需要向不同的（例如，非连续性的）状态行进，如在{状态 1，01}和{状态 1，FD}处所示的状态。除了被包括的特殊字符的串和所关心的具有共同初始字符的串的特殊情况外，对任意长度为 n 的串的完整识别需要 n-1 个状态。为此，状态和状态表行的数目通常必须非常大，即便是对于所关心的相对适度数目的串也如此。

7. 与前段内容相反，大部分的状态能够通过一个或两个唯一的条目和缺省的“到达状态 0”而得到完全表征。图 1 的状态表的这些特性在本发明中被用来产生高程度的硬件节约性并大幅度地加速解析处理以用于所关心的串的通常情况。

如上所暗示的一样，在解析处理的中，如传统地实现一样，系统开始于给定的缺省/初始状态，在图 1 中被描述为状态 0，接着，在重复处理后随着匹配字符被发现而前进到较高序号的状态。在所关心的串被完全识别时，或者在具有潜在匹配性的串的中间位置处指定一个特定的操作时，例如存储令牌或发出中断的操作被执行。然而，在用于文件的各

个字符的每次重复中，字符必须从 CPU 存储器获取，状态表必须被（再次从 CPU 存储器）获取，并且各种指针（例如，指向文件的字符和状态表中的基址）和寄存器（例如，指向串的初始匹配字符地址和累加的长度）必须在顺序的操作中被更新。因此，很容易认识到上述解析处理耗费大量的处理时间。

根据本发明的解析器加速器 100 的高层次示例性的方框图如图 2A 所示。如本领域的普通技术人员可以认识到的那样，图 2A 还可以理解为根据本发明执行解析的执行步骤的流程图。如下面将连同图 3、4A、4B、5A 和 6 进行详细论述的那样，本发明在表示状态表时使用了一些硬件节约措施，以使得多个硬件流水线（pipelines）被开发出来，这些硬件流水线尽管在时间上有轻微的不对称（skewed）但基本上以并行的方式操作。这样，指针和寄存器的更新能够被基本上并行地和与其它操作并发地执行，同时通过较快地访问以并行方式操作的硬件以及从关于状态表和文件的 CPU 存储器中预取（prefetching），存取存储器所需的时间被大量降低。

作为一般的总体观点，例如 XML™ 文件的文件被外部地存储在由寄存器 112、114 编索引的 DRAM 120 中，并优选地通过 32 位字传输到为上述流水线起到多路复用器的作用的输入缓冲区 130。各个流水线包括字符调色板（palette）140、状态表 160 和下一状态调色板 170 的拷贝；其每一个都适应于压缩形式的状态表的部分。下一状态调色板 170 的输出包含进入状态表 160 中的条目的地址的下一状态地址部分和待被存储的令牌值（如果有的话）。字符调色板 140 和下一状态调色板 170 中的操作是对高速内部 SRAM 简单的存储器访问，内部 SRAM 可以相互并行地运行，并且与对形成状态表 160 的外部高速 DRAM（其还可实现为高速缓冲器）的简单存储器访问并行。因此，只需要 CPU 初始地控制这些硬件元件（但是，其一旦启动，仅能够用偶然的 CPU 操作调用来自发地起作用以更新文件数据和存储令牌）的相对少的时钟周期以用于文件中的各个字符的评估。基础的加速增益是在 CPU 中各个字符的所有操作持续时间加上在高速 SRAM 或 DRAM 中单一自发地执行的存储器操作的持续时间的总和的减少。

应该理解，在这里所称为“外部的”存储器构造是用来暗示存储器120、140的配置，考虑到所需的存储的量和从上述硬件解析器加速器和/或主CPU的存取，其目前是本发明人所优选的。换句话说，令牌处理和一些其它操作提供根据本发明的解析器加速器的体系结构以便利存储器共享，或至少便利由主机CPU以及硬件加速器存取是具有有益效果的。根据这些论述，本领域的普通技术人员应该认识到没有其它的预期内含和更广范围的例如同步DRAM（SDRAM）的硬件替代物是适合的。

现在参照图3到图6以支持图2A中优选的实现的硬件节约措施为例对字符调色板140、状态表160、下一状态表170和下一状态和令牌的格式进行讨论。可以使用其它的技术/格式，同样，上述描述的格式可以理解为示例性的但目前是优选的。

图3描述了的字符调色板优选形式，该调色板与被包括在或可以被包括在所关心的串中的字符对应。相对应于图1中的状态表中列的数目，这些格式优选地提供了编号为0-255的条目。（术语“调色板（palette）”与包含用于支持各颜色的数据并且被集合性地称为全色域（gamut）的术语“调色板（color palette）”以相同的意思使用，调色板的使用减少了状态表中的条目/列）。例如，被称为“空字符”的不会导致任何状态变化的字符能够在状态表中的单列中表述，而不是在许多列中表述。在144处为空字符输出进行测试是所期望的，这能够基本地加速用于解析的处理，这是因为它允许对对下一字符立刻进行处理，而无需对状态表存取的进一步的存储器操作。这种格式能够被单个寄存器所适应，或者被由（例如）在指向特殊字符的调色板（被图2中用重叠的存储板示意地描述）的基址寄存器142中的数据进行配置的存储单元所适应。来自文件（例如XML™文件）的当前8位字符（四个中的一个从输入缓冲器130提供，和从外部DRAM 120接收的四字节字一样）寻址调色板中的条目，该调色板接着输出地址作为状态存储器的索引或局部指针。这样，通过以上述格式提供调色板，图1的功能的一部分能够以相对被限制容量的单个寄存器的形式提供；这样，允许它们中的多个被形成并以并行的方式操作，同时保持充分的硬件节约并支持状态表160中的其它功能。

图4A显示了优选的状态表格式，其以与字符调色板（例如，基本上

和寄存器一样）相似的方式构成或配置。与图 3 中的字符调色板的主要不同在于，寄存器的长度依赖于对所期望字符的响应次数以及所关心字符串的个数和长度。因此，如果能够被经济地提供的内部存储器的数量在具体的情况中是不充足的，则提供在 CPU 或其它外部 DRAM（可能具有内部或外部的高速缓冲）中实现这种存储器的可能性是考虑期望得到的。尽管如此，由于在图 1 的状态表中的具有较高复制性的条目能够被减少到单个条目，所以提供充分的硬件节约是很清楚的；单个条目的地址通过如根据图 3 的字符调色板如上所述而提供的数据而被接纳。状态表 160 的输出优选地是一位、两位或四位，但是规定多达 32 位的位数可以提供增加的灵活性，如在下面参照图 4B 论述的那样。在任何情况下，状态表的输出提供了访问下一状态调色板 170 的地址或指针。

现在参照图 4B，如在后面关注的本发明的完美特征一样，本发明优选的实现特征包括状态表控制寄存器 162，其允许更进一步的充分硬件节约，尤其是在如果状态表 160 的 32 位输出将被提供的情况下。实质上，状态表控制寄存器通过允许变长字被存储到状态表并将其从其中读出而提供状态表信息的压缩。

更具体地说，状态表控制寄存器 162 存储和提供图 4A 的状态表 160 中的各个条目的长度。由于图 1 中的一些状态表条目是高度复制性的（例如，“到达状态 0”、“保持于状态 n”），所以这些条目不仅能够通过状态表 160 中的单个条目或至少远少于图 1 中的条目来表示，而且还可被较少的位表示，这些较少的位可以少到即便是大部分或所有的复制性条目被包括在状态表时仍能够产生充分的硬件节约，就像在一些状态表中被发现是方便的那样。本领域的普通技术人员应该认识到这种减少的原理相似于所谓的熵（entropy）编码。

现在参照图 5A 对下一状态调色板 170 的优选格式进行讨论。下一状态调色板 170 优选地以与上述讨论过的字符调色板 140 非常相同的方式实现。然而，由于具有状态存储器 160，可能需要的条目的个数不会被事先知道，并且各自条目的长度优选地非常长（例如，两个 32 位的字节）。另一方面，由于只有相对小和可预见的地址范围需要包含在任意给定的时间，所以下一状态调色板 170 能够作为高速缓冲来操作（例如，使用

下一状态调色板基址寄存器 172）。而且，如果状态表 160 的 32 位输出被提供，一些上述数据能够被用来补充下一状态调色板 170 的条目中的数据，可能在后者中允许较短的条目或可能完全地跳过下一状态调色板，如短划线 175 所示。

如图 5A 中所示，从下一状态调色板 170 输出的低地址 32 位字是待被保存的令牌。这些令牌优选地被形成为具有 16 位令牌值、8 位令牌标记和 8 位控制标记，其中，令牌值和令牌标记被存储在令牌缓冲器 190 的由指向串的开头的指针 192 和通过计算成功的字符比较而积累的长度共同提供的地址处。控制标记设置中断到主 CPU 或控制解析器加速器中的处理。上述后者控制标记中的一个被优选地用来设置能够跳过字符的功能，这不会导致在除了状态 0 之外的状态处的状态变化，例如在所关心的串中出现的任意长度的相同或相关字符的串，如上所能推导出的一样。在这样一种情况下，下一状态表条目无需从 SRAM/SDRAM 获取就能够被重新使用。输入缓冲地址 112 无需额外的处理就可被递增；从而允许用于字符的确定串的解析的充分附加的加速。第二个 32 位字是反馈到寄存器 180 和加法器 150 的地址偏移，其待与来自字符调色板的索引输出连接（concatenate）以形成指向用于下一字符的状态表指针。相应于状态 0 的初始地址通过寄存器 182 提供。

这样可以看出，字符调色板的使用、简化形式的状态存储器和下一状态存储器将传统的状态存储器的操作的功能清楚地表达为单独的阶段；各个阶段能够用相对稍高速度的存储器来极快地执行，这样，各阶段可被复制以依次形成与令牌的其它操作和存储并行的对文件各自字符操作的并行流水线。因此，解析处理能够被极大地加速，即便是相对于在其它的字符处理能够被开始之前必须顺次执行所有上述功能的专用处理器。

总之，该加速器存取主机 CPU 的字符数据（有时被称作暗含网络传输的数据包）和状态表所定位的程序存储器。加速器 100 经由存储器映射寄存器而处于主 CPU 的控制下。该加速器能够中断主 CPU 以指出例外、报警和终止，在入侵检测的上下文中，其可一般地被称为模式匹配报警。入侵事件报警等。在解析开始时，指针（112，114）被设置到待被解析

的输入缓冲器 130 的数据的开端和结尾，待被使用的状态表（如基址 182 所示和其它控制信息（例如，142）被建立在加速器内。

为了启动加速器的操作，CPU 发出命令到加速器，作为对该命令的响应，加速器从 CPU 程序存储器（例如，120 或高速缓冲）获取第一个 32 位字，并将其置入到输入缓冲器 130，第一字节/ASCII 字符从输入缓冲 130 中选择。加速器获取相应于输入字符（也就是，图 4A 对应图 1 中的完整状态表的单个字符或单个列）的状态信息和当前状态。该状态信息包括下一状态地址和待执行的例如中断 CPU 或终止处理的任一特定操作。

加速器接下来从输入缓冲器 130 中选择待被分析的下一字节，并用加法器 150 利用可得到的新状态信息重复上述处理。上述操作或令牌信息的存储能够被并发地执行。这种执行在输入字的所有四个字符被分析之前继续。接着（或与解析预获取的第四个字符并发地），缓冲器 112、114 被比较以确定是否到达了文件缓冲器 120 的末尾，如果达到了文件缓冲器的末尾，则中断被发送回 CPU。否则，获取新的字，缓冲器 112 被更新，上述处理被重复。

由于指针和计数器在专用的硬件中实现，所以它们能够被并行地更新，而不是像如果以软件实现时所需的串行更新。这就将解析数据的字节的时间减少至执行以下动作所需的时间，即，从局部输入缓冲器中获取字符、从高速局部字符调色板存储器中产生状态表地址、从存储器中获取相应的状态表条目和再次从局部高速缓冲器中取出下一状态信息。上述操作的一些操作能够在单独的并行流水线中并行地执行，在状态表信息（部分地或全部地通过下一状态调色板来提供）中指定的其它操作可在继续更进一步的字符解析的同时被执行。

因此可以很清楚地看出，本发明通过小而经济数量的专用硬件来提供解析处理的充分加速。在解析器加速器能够中断 CPU 时候，上述处理的操作在初始命令后被完全从 CPU 移到解析器加速器。然而，由于即便是与其它的解析操作并发地操作时，需要有充分的时间来用于令牌的处理，所以如上所述提供的加速对于可能的入侵或安全破坏的检测不是最佳的，尤其鉴于如下事实，即，通过在解析过程期间发出命令，就可以

启动很难或者不可能获得的操作。

现在参照图 2B，其中显示了用于硬件解析器加速器的结构配置，其将解析的处理速度极大地提高，从而超出如上所述但为了可能的入侵和安全破坏的标志（signature）的检测的有限目的的图 2A 中的结构配置的处理速度，而且是完全与其兼容的。通过比较图 2A 和图 2B，本领域的普通技术人员应该认识到，图 2B 中的结构配置大体上是图 2A 中的结构配置的子集（sub-set），并且提供了相同的能够搜寻可以是入侵的标志的全部串的效果（例如，匹配一个或多个表达式或表达式的部分，以此，调色板匹配报警能够在状态表中被编码的完整表达式匹配之前被发给 CPU，这样增加了响应速度），但在由于只有保护系统的中断或异常的发出需要被作为处理的结果被发出，所以其通过省略令牌处理而同时提供更进一步的加速。因为可能的安全破坏标志的含入被完成，所以如上所述的用于文件的完整解析能够在该文件被筛选（screen）后被执行。由于在上述筛选处理的过程当中令牌处理被省略，所以对存储器访问的次数被减少。也就是，对于根据本发明的入侵检测加速器（和与上述的硬件解析器加速器比较）而言，令牌处理和字符调色板的使用被省略，这导致了较低的存储器资源需求和在处理时间上有一些减少。然而，由于很多这样的处理是以并行方式执行的，所以处理速度的增加通常约为 25％或稍微低于 25％，这部分地依赖于被用于与速度、逻辑速度等有关的各种资源的特殊设备。然而，也许比速度更重要的是如下事实，即，任何的可能存在的安全破坏标志将被检测到，并且可在作为攻击的一部分的相应命令被 CPU 执行之前发出补救性中断或异常（exception）。

图 2A 中表示了图 2B 中的结构配置的所有功能单元，并且相应的单元使用相同的标号。因此，显而易见，根据本发明的入侵检测解析器加速器 200 完全与上述的解析器加速器兼容，并且，其结构配置的改变能够通过程序编制而在很大程度上被完成，以使得上述入侵检测处理是图 2A 的解析器加速器操作的特殊模式。

具体地说，输入缓冲器 120 和输入字缓冲器 130 以及地址寄存器 112，114、加法器 150 和状态表基址寄存器 182 与上述的相应单元一致，并以相同的方式来访问状态表 160。不同点主要在于省略了字符调色板、下一

状态调色板存储器、状态表中的数据以及数据的内部格式。该状态表基本上具有与图 2A 所示的实施方式相同的 256 个字符的宽度。应该理解，为其进行搜寻的标志可能比任意数目的字符组成的单个字符串更复杂。该标志更通常地被描述为比字符串更复杂的“正则表达式（regular expressions）”，如 Vikram Vaswani 等人（开放的源代码 Web 开发网站（Developer shed））在“So what's a $#!%% Regular Expression Anyway?!（$#!%%正则表达式究竟是什么）”（曼乐范（Melonfire）公司版权 2000－2002）中论述的那样，其全部的内容并入本文作为参考。这样，与正则表达式相应的状态表会远大于单个字符串的状态表。另外，若干个正则表达式能够使用同一状态表表被并发地搜寻，这将导致非常大的状态表。然而，在实践中，64 个状态通常是足够的。然而，如果不够，则将需要将状态表条目扩展成超过 8 位。因此，如上所述提供的极端压缩通常不是必需的，并且如上所述能够提供足够部分的状态表 160 的硬件不是为了减少所需的存储器访问的次数而被需要用来检测攻击标志的状态表的全部。

如图 2A 所示的实施方式中，状态表中的数据的格式优选地包括 n＝256 个条目以接纳可通过字节来表述的字符的个数。然而，在如图 2B 所示的实施方式的情况下，对状态表的访问是直接从缓冲在字缓冲器 130 中的字符位来执行。状态表中的各条目的内容只需包括下一状态或待被装载的状态表的行，这就限定了所关心的字符串，并允许字符串被跟随，和/或允许用于将被发出的中断或排除的标记或其它编码用于支持将上述串识别为标志的串的字符（其并不一定是构成上述标志的串的最后字符），如图 5B 所示。下一状态通常能够以少于 8 位的方式描述，在检测到所关心的串后的将被产生的异常中断可被描述为一位。

这样，字符被顺次地测试，在所关心的串的第一个字符被遇到之前，除了寄存器 112，114 之外的其它寄存器都不会更新。也就是，在上述检测之前，即便是状态都不会改变以及在寄存器 180 中的下一状态不会被更新。因此，文件能够用极快的速度被筛选以用于最初的字符。在所关心的串的最初字符被遇到时，下一状态数据被从状态表中读取，寄存器 180 被更新，新的状态表数据被载入状态存储器中（如果还不存在的话），

并且下一字符被以相同的形式处理。该状态表存储器远远小于如上所述的用于 XML™ 解析器的存储器。这就允许该状态表存储器在具有图 2A 或图 2B 的其它逻辑和元件的芯片的板载实现，这就将处理周期时间尽可能减少到用于使用外部存储器的设计中所需时间的 25％。然而，如果加速器被简单地设计为 XML™ 加速器操作的特殊模式，则状态表将在外部存储器中实现，上述的速度增长不会实现。因此，在上述情况下，提供根据状态表的大小交替地使用的板载和外部存储器是划算的。这样，各个字符只需相对少的时钟周期来筛选用于攻击标志的文件。在足够数量的字符被处理以识别所关心的串时，从状态表中读取的数据将会包括被作为命令发送到主机 CPU 以用于保护系统的中断或异常。

尽管包括如图 2A 或图 2B 中所示的本发明的系统的体系结构对于本发明的实行不是关键的，但是如图 2C 显示的体系结构对于图 2B 中的入侵检测加速器而言是优选的。具体地说，主机 CPU 230 和其主存储器 210 通过总线 220 连接，本发明的硬件解析器加速器通过总线 220 与主存储器 210 和主机 CPU 230 进行通信。在 CPU 230 能够监控主存储器 210 和加速器 100/200 之间的通信时，令牌还没有被定义或建立，执行用于实现攻击的编码是不可能的。因此，在包括在攻击中的任何操作执行之前，本发明就可发出补救的中断和异常。

根据前述的内容可以看出，本发明提供了非常快速的文件筛选速度以用于寻找一些标志，这些标志可以表明在硬件加速器的上下文（context）和环境内企图攻击的可能性，此举可将用于解析例如 XML™ 文件的时间大大降低到本发明之前所需要的时间的一小部分。本发明的入侵检测解析器不需要超出根据本发明的解析器加速器的附加元件或硬件，并能够在任何入侵处理变为可执行之前发出中断和/或异常。

尽管本发明根据单独的优选实施方式来描述，本领域的普通技术人员将会认识到在本发明的精神和所附的权利要求的范围内，也可对本发明做出修改。

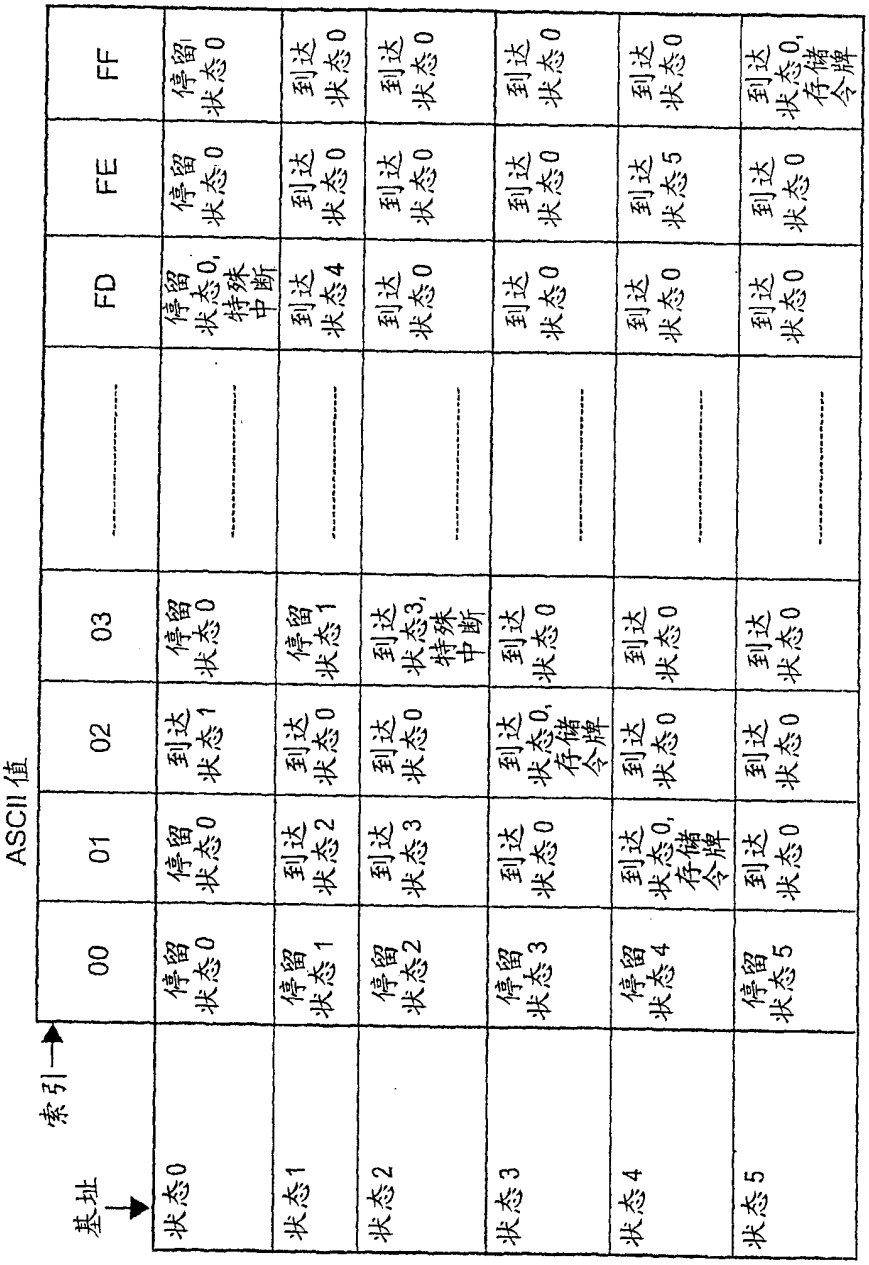

基址
索引
ASCII值
00
01
02
03
FD
FE
FF
状态0
停留状态0
停留状态0
到达状态1
停留状态0
.....
停留状态0,特殊中断
停留状态0
停留状态0
状态1
停留状态1
到达状态2
到达状态0
停留状态1
.....
到达状态4
到达状态0
到达状态0
状态2
停留状态2
到达状态3
到达状态0
到达状态3,特殊中断
.....
到达状态0
到达状态0
到达状态0
状态3
停留状态3
到达状态0
到达状态0,存储令牌
到达状态0
.....
到达状态0
到达状态0
到达状态0
状态4
停留状态4
到达状态0,存储令牌
到达状态0
到达状态0
.....
到达状态0
到达状态5
到达状态0
状态5
停留状态5
到达状态0
到达状态0
到达状态0
.....
到达状态0
到达状态0
到达状态0,存储令牌

图 1

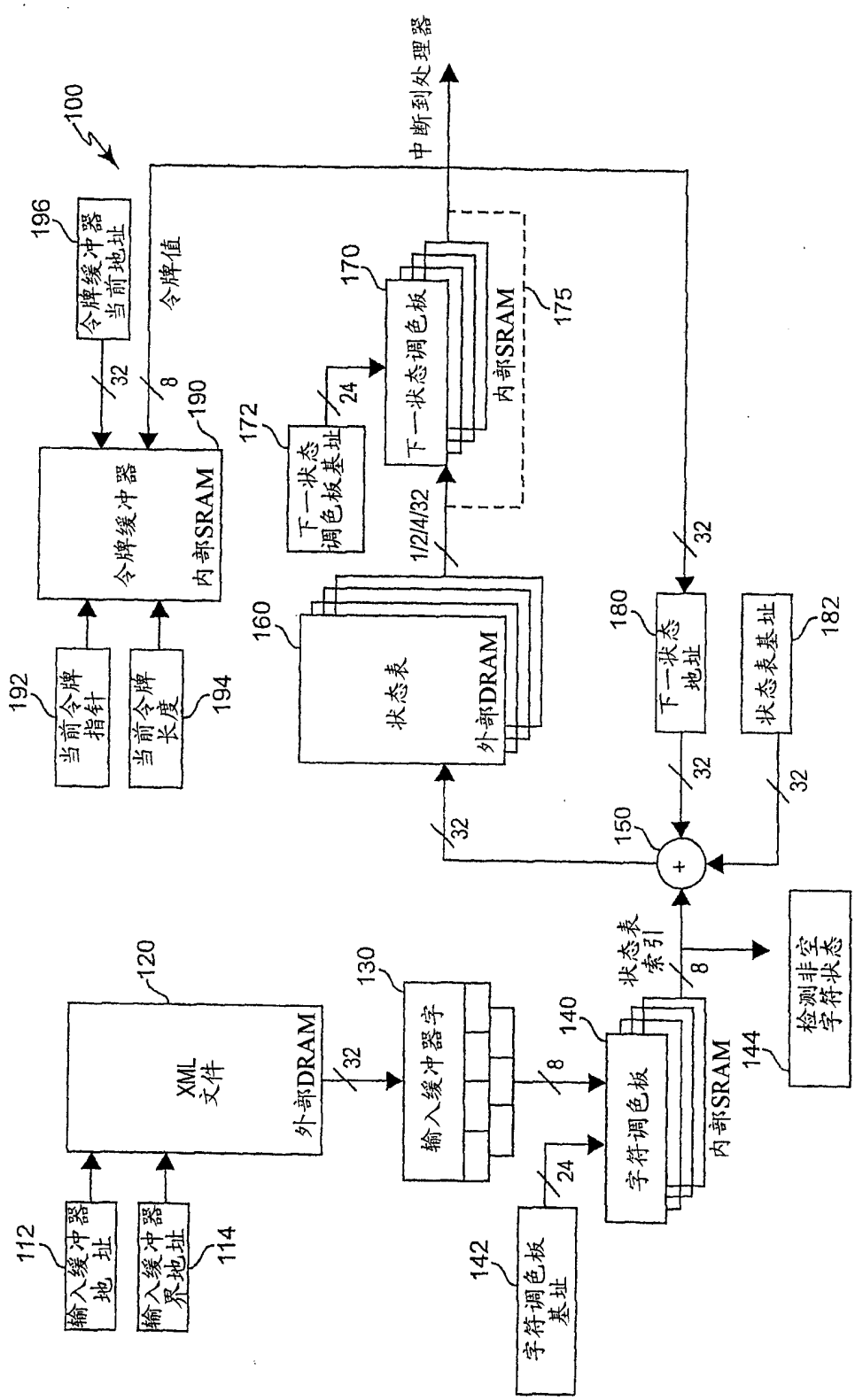

100
中断到处理器
196
令牌缓冲器当前地址
令牌值
32
8
190
令牌缓冲器
内部SRAM
192
当前令牌指针
194
当前令牌长度
172
下一状态调色板基址
24
170
下一状态调色板
内部SRAM
175
1/2/4/32
160
状态表
外部DRAM
32
150
+
180
下一状态地址
32
状态表基址
182
32
32
120
XML文件
外部DRAM
32
130
输入缓冲器字
8
140
字符调色板
内部SRAM
144
状态表索引
8
检测非空字符状态
112
输入缓冲器地址
114
输入缓冲器界地址
142
字符调色板基址
24

图 2A

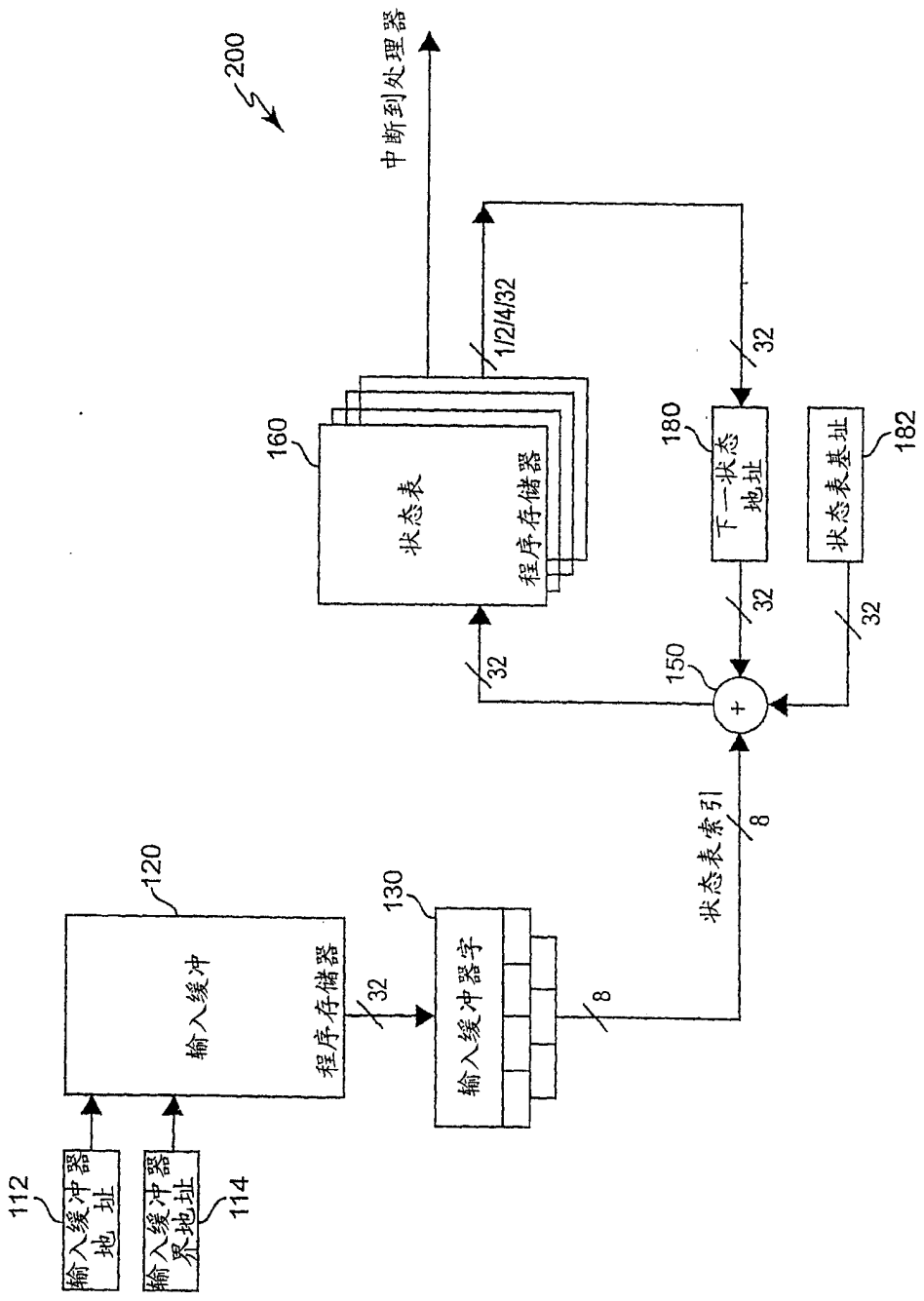

200
中断到处理器
1/2/4/32
160
状态表
程序存储器
32
180
下一状态地址
32
182
状态表基址
32
150
+
32
状态表索引
8
130
输入缓冲器字
8
120
输入缓冲
程序存储器
32
112
输入缓冲器地址
114
输入缓冲器界地址

图 2B

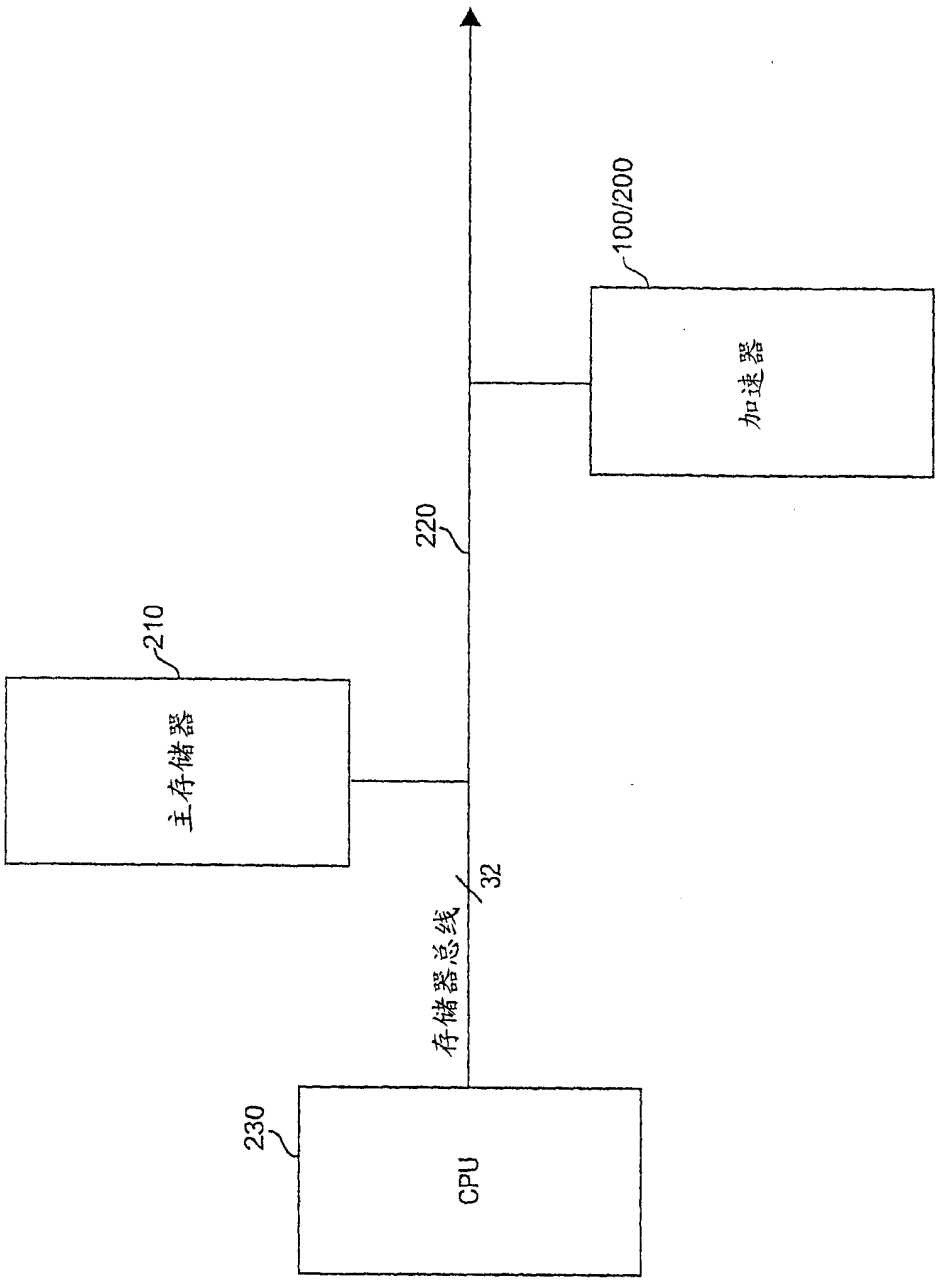


图 2C

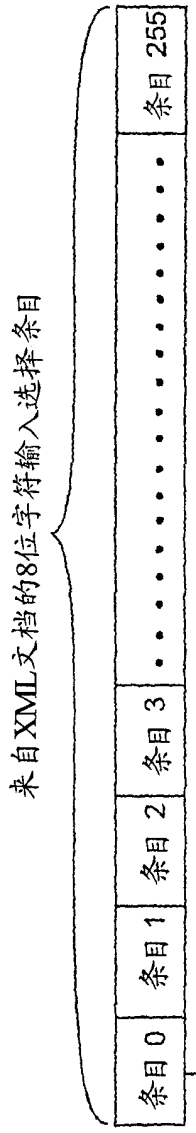


图 3

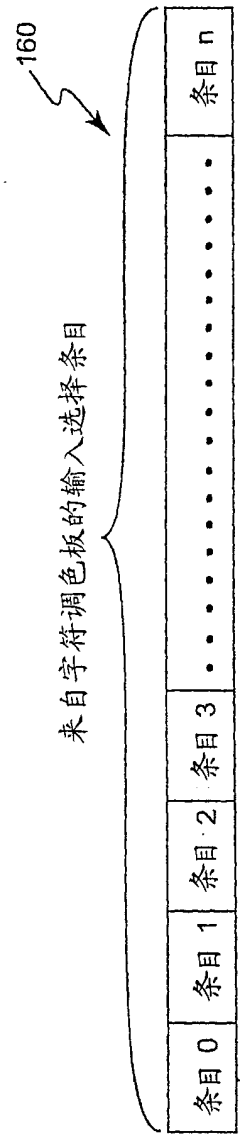

160
来自字符调色板的输入选择条目
条目 0
条目 1
条目 2
条目 3
条目 n
进入下一状态调色板的条目

图 4A

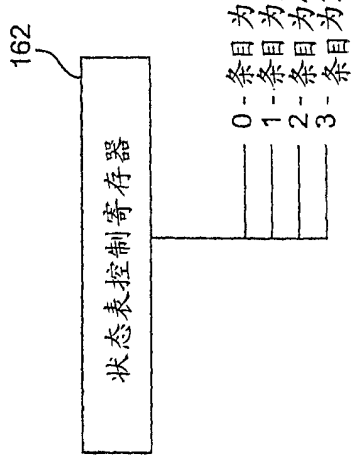

162
状态表控制寄存器
0 - 条目为1位
1 - 条目为2位
2 - 条目为4位
3 - 条目为32位

图 4B

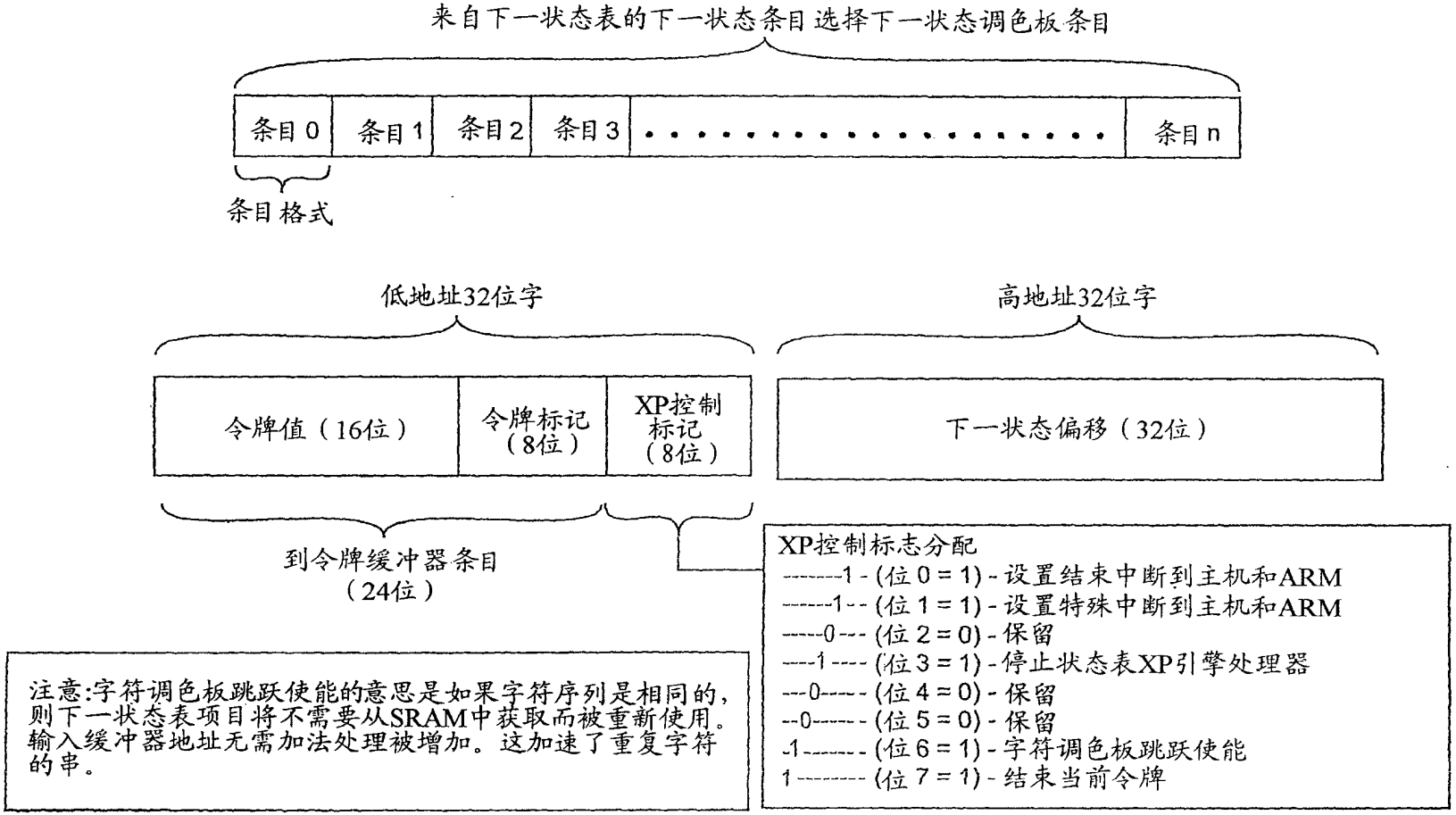

来自下一状态表的下一状态条目选择下一状态调色板条目
条目0
条目1
条目2
条目3
条目n
条目格式
低地址32位字
高地址32位字
令牌值（16位）
令牌标记（8位）
XP控制标记（8位）
下一状态偏移（32位）
到令牌缓冲器条目（24位）
XP控制标志分配
-------1 - (位0 = 1) - 设置结束中断到主机和ARM
------1- - (位1 = 1) - 设置特殊中断到主机和ARM
-----0-- - (位2 = 0) - 保留
----1--- - (位3 = 1) - 停止状态表XP引擎处理器
---0---- - (位4 = 0) - 保留
--0----- - (位5 = 0) - 保留
-1------ - (位6 = 1) - 字符调色板跳跃使能
1------- (位7 = 1) - 结束当前令牌
注意:字符调色板跳跃使能的意思是如果字符序列是相同的，则下一状态表项目将不需要从SRAM中获取而被重新使用。输入缓冲器地址无需加法处理被增加。这加速了重复字符的串。

图 5A

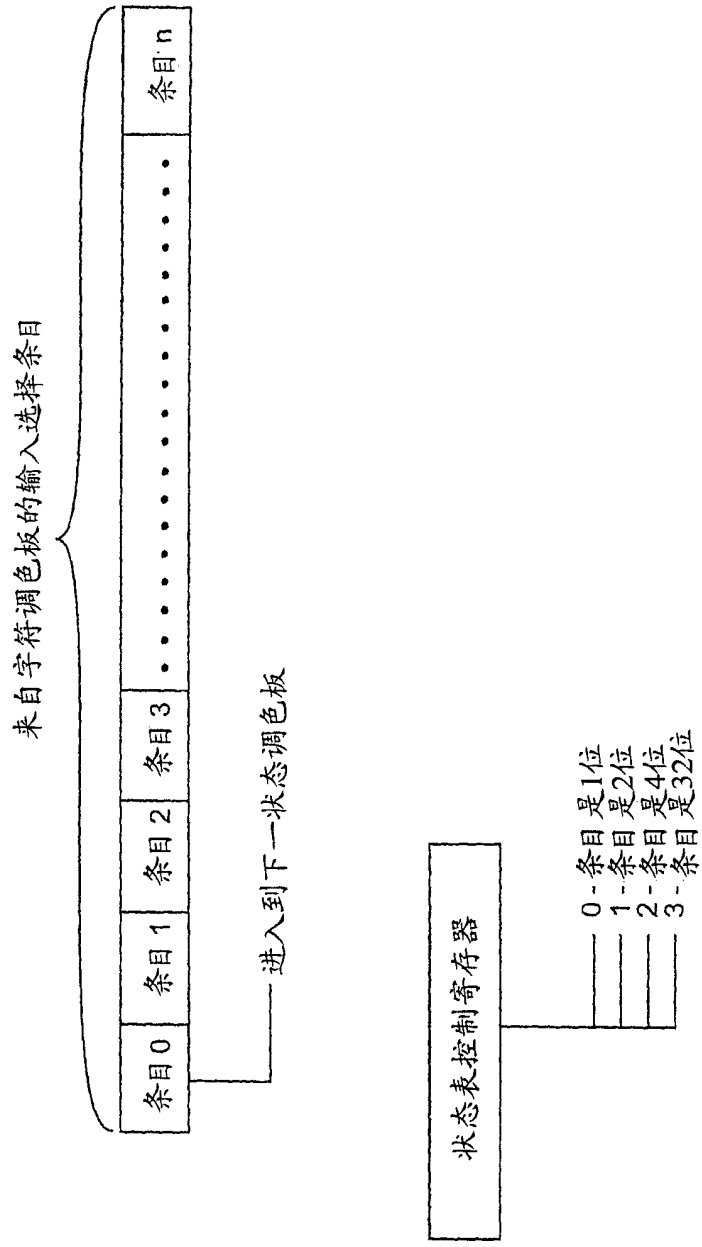

来自字符调色板的输入选择条目
条目0
条目1
条目2
条目3
条目n
进入到下一状态调色板
状态表控制寄存器
0-条目是1位
1-条目是2位
2-条目是4位
3-条目是32位

图 5B

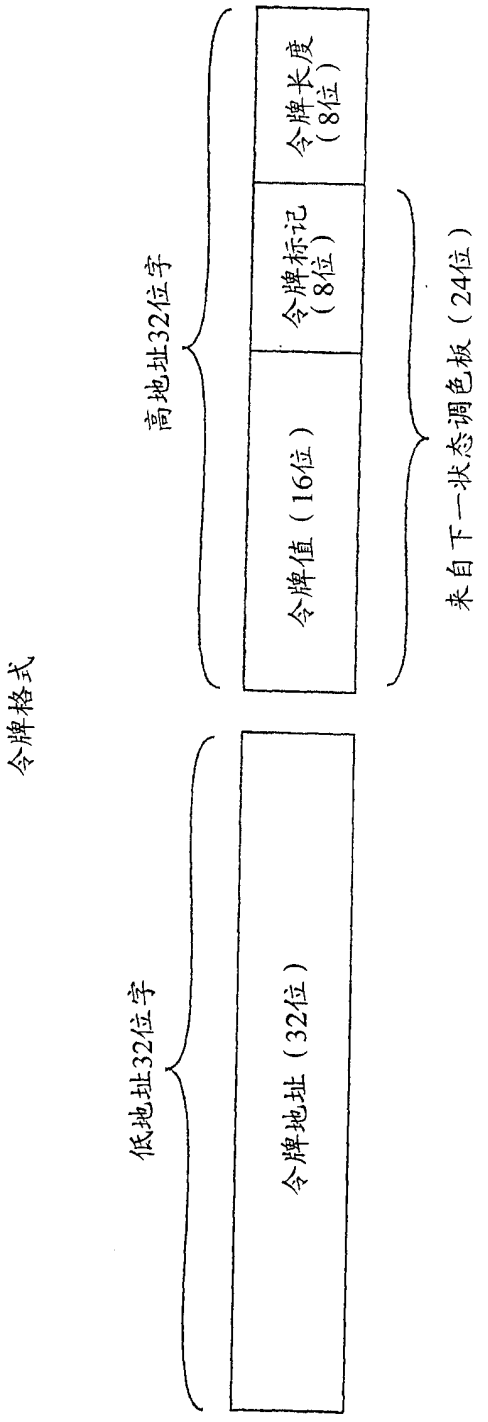

令牌格式
低地址32位字
令牌地址（32位）
高地址32位字
令牌值（16位）
令牌标记（8位）
令牌长度（8位）
来自下一状态调色板（24位）

图 6