[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 7/00 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200380104145.6

[43] 公开日 2006 年 9 月 6 日

[11] 公开号 CN 1830173A

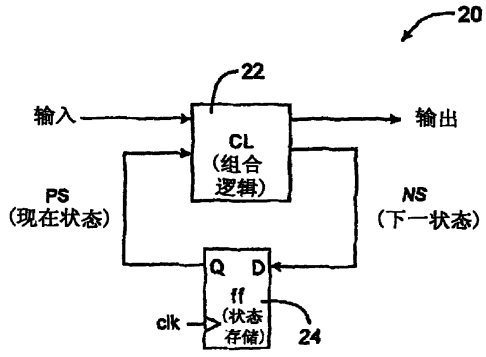
[22] 申请日 2003.11.26
[21] 申请号 200380104145.6
[30] 优先权
[32] 2002.11.27 [33] US [31] 60/429,736
[86] 国际申请 PCT/US2003/038010 2003.11.26
[87] 国际公布 WO2004/051907 英 2004.6.17
[85] 进入国家阶段日期 2005.5.26
[71] 申请人 罗得岛及普罗维登斯属地高等教育管理委员会
地址 美国罗得岛
[72] 发明人 A·K·尤特
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 李 玲
权利要求书 1 页 说明书 11 页 附图 11 页

[54] 发明名称

数字系统性能增强的系统和方法

[57] 摘要

本发明利用低于最坏情况需要的时钟周期(即一个高速时钟)实现数字计算，同时在具有相同硬件的第二系统中，利用更大的假定最坏情况时钟周期实现相同的计算(即一个低速时钟)。 对来自计算的输出进行比较以确定是否有错误发生。 如果在两个应答之间有不同，高速计算一定是错误的(即发生了计算错误)，系统使用来自低速系统的应答。在一个实施方案中，本发明利用两个复制的低速系统，其中每个以主系统的半速率运行。 然而两个复制系统以相同速率生成的结果之和与主系统相同，该主系统在本发明之外以比其可能更快的速率运行。 因此虽然使用了更多的硬件，本发明改进了性能(例如，速度)。 有利地，本发明动态适应以在实际操作条件下达到可能的最好性能。



1、 一种用于数字系统性能增强的系统，其接收输入信号和第一时钟信号,所述系统包括：

时钟控制逻辑，接收第一时钟信号并且生成第二时钟信号和第三时钟信号，其中所述第二和第三时钟信号具有所述第一时钟信号频率的整分数的频率；

第一数字同步网络,响应第一时钟信号和输入信号，并提供第一输出信号；

第二数字同步网络，基本上与所述第一数字同步网络相同，其中所述第二数字同步网络接收所述第二时钟信号和输入信号，并且提供第二输出信号；

第三数字同步网络，基本上与上述第一数字同步网络相同，其中所述第三数字同步网络接收所述第三时钟信号和输入信号，并且提供第三输出信号；和

比较和选择逻辑，响应所述第一、第二和第三输出信号，以确定所述第一输出信号是否发生计算错误，其中，如果没有错误发生，所述比较和计算逻辑提供一个指示所述第一输出信号的系统输出信号，其中，如果发生错误，所述比较和选择逻辑提供指示所述第二输出信号的所述系统输出信号。

数字系统性能增强的系统和方法

相关申请的交叉参考

本申请要求2002年11月27日申请的序号为60/429,736的美国专利申请的优先权，并且本申请是2000年9月27日申请的序号为09/672，128的美国专利申请的连续部分申请，所述美国申请要求1999年9月27日申请的序号为60/156,219的美国专利申请的优先权，所有前述的申请公开的全部内容在这里做为参考。

发明背景

本发明涉及数字电子系统领域，尤其涉及同步数字电子系统。

包括中央处理单元（例如Intel Pentium）的计算机、蜂窝电话、微波炉以及几乎现在制造的每个电子器件都利用数字硬件来进行工作。所述仅仅根据电路的当前输入状态计算结果的数字电路由组合逻辑构成。组合系统能够用于许多应用中，但是对于意识到的任何有趣的数字系统，该系统必须将其输出建立在当前输入和系统的在先输入或状态的基础上。

具有两种“状态”类型的数字系统，它们的状态保持在存储器设备中，因此这些系统经常被称为具有记忆的系统。第一种类型，异步数字系统，一旦改变其输入值就改变状态。即使使用现代异步技术，模拟、设计以及验证异步系统在实践中是及其困难的。数字系统的一个优点是能够在逻辑延迟允许的范围内尽快的操作。

第二种数字系统类型是同步系统，其中状态仅在全球系统时钟（也就是与时钟同步）确定的时刻改变。例如设想一具有每秒振动500亿次（也就是500MHz）的基础芯片（CPU）时钟的Intel Pentium III处理器；该处理器只在这些振动的一次或多次的开始时改变它的状态。同步方法使得数字系统的设计、构造和使用更容易。

然而，同步系统的固有难点和性能损失是时钟的持续时间/周期必须足够大以能够处理最坏情况的操作条件和制造的允许误差。该周期典型的至少是标准（普通）操作和制造允许误差标称要求长度的两倍。因此，假如不是在最坏情

况下，上述数字系统的性能是其可实现性能的一半或者更少。

数字同步系统20可以由图1中示出的框图模型表示。该系统的组件包括组合逻辑22（CL）、触发器或锁存器（FF）24。锁存器24维持系统的当前/现在状态。每个锁存器一般存储一位信息。众所周知的，只有当时钟信号转变时，触发器才改变它的内容或状态。相同的时钟到达所有的触发器。组合逻辑22没有时钟输入或反馈回路，仅由于电子电路和光速的制约，其输入的变化传播到一个或多个输出就会有延迟。但是从时钟转变到输出的改变，锁存器24也具有传播延迟 。

系统20利用逻辑22进行操作以从系统现在状态和输入到系统的当前值计算系统的下一状态（NS）。当时钟转变时下一状态存储在锁存器24中，处理过程重复进行。为了使该系统完全起作用，计算结果必须通过组合逻辑传播并在锁存器发生时钟的相关转变之前出现在锁存器的输入。

如果通过逻辑和锁存器的精确延迟是已知的，时钟频率将设置为延迟总和的倒数，系统将以最高性能运行（每秒的计算做为度量）。然而延迟不是恒量，而是随制造过程的差异、供电电压的变化、工作温度和湿度的变化以及其它因素而变化。由于这些变化，需要保证数字系统在最坏情况下的工作（例如温度极限），时钟速度设置到比在多数典型的情况下需要的更低、更保守的数值。结果使平均用户感受到明显低于实际需要的低性能。

因此，需要更快的同步系统结构。

发明内容

简单的，根据本发明的一方面，系统时钟的频率自动增加直到检测到系统错误，然后时钟的频率会自动轻微地减少到一个不会引起系统错误的值。

简单的，根据本发明的另一个方面，系统时钟的频率会自动增加直到检测到不可接受数目的错误，然后时钟频率自动轻微地减少，使得检测到的错误数目少于不可接受的错误数目。

本发明实现使用低于最坏情况需要的时钟周期（也就是高速时钟）的数字计算，同时，使用大的、假定最坏情况的时钟周期（也就是低速时钟）在具有相同硬件的第二系统实现相同的计算。通过比较计算的输出来确定是否有错误发生。如果两个应答有差别，高速计算一定是错误的（也就是发生计算误差），系统使用来自低速系统的应答。

在一个具体实施例中，本发明利用两个低速系统的复制系统，每个以主系统一半的速度运行。然而，以相同速率运行的两个复制系统产生的结果总和同主系统一样，该主系统在本发明之外以比其可能更快的速率运行。因此虽然本发明使用了更多的硬件，但改进了性能（例如速度）。

有利地，本发明动态适应以在实际操作条件下达到可能的最好性能。

另一方面，本发明包括一个时序误差避免（TEA 时间）系统，该系统包括使用具有流水线寄存器之间最长路径延迟的附加逻辑来基于循环周期测试该系统时钟是否太快或太慢。如果应用到延迟测试逻辑输入的信号在机器最慢路径的时间内出现在测试逻辑的输出，系统会提供一个信号以提高系统时钟速度。可选地，如果应用到延迟测试逻辑输入的信号在多于机器最慢路径的时间出现在测试逻辑的输出，系统将提供一个信号减去安全边界来降低系统时钟速度。因为延迟测试逻辑的特性（延迟等）反映了主逻辑的特性（它们一起密封在相同的芯片里），系统时钟既适用于动态环境条件，包括温度和工作电压，也适用于静态变化的制造条件。

本发明的这些以及其他的目的、特性和优点将由以下附图中示出的优选实施方案的详细描述而更加清楚。

附图的简要说明

图 1 示出了一个数字同步系统的功能框图；

图 2 示出了根据本发明的数字系统的第一个实施方案；

图 3 示出了一个用于图 2 所示系统的时序图；

图 4 示出了本发明第一可选实施方案；

图 5 示出了用于图 4 中所示第一可选实施方案的时序图；

图 6 示出了本发明第二可选实施方案；

图 7 示出了用于图 5 中所示实施方案的时序图；

图 8 示出了一个时序误差避免系统的框图；

图 9 是一个图 8 所示时序误差避免系统更加详细的框图示意图；

图 10 是一个图 9 所示时序检查电路的框图示意图；和

图 11 示出了用于图 10 所示实施方案的时序图。

发明的详细描述

图 2 示出了根据本发明的数字系统 30 的第一实施方案。数字系统 30 包括

图 1 所示的主数字同步系统 20 的两个复制系统 20a、20b，其中每个复制系统 20a、20b 以主数字同步系统 20 的半速率工作。一个复制系统在奇周期复制主数字同步系统 20 的结果，而另一个在偶周期复制结果。两个半速率系统 20a、20b 在互相同步输出的主系统周期工作。分别在线路 32、34 上的两个半速率系统 20a、20b 的输出在交替周期利用比较器 38、40 与在线路 36 输出的主系统输出进行比较。如果两个输出之间有不同（例如在线路 32 和 36 上的信号之间），检测到一个错误，自然假定错误是由高速系统引起的（即系统 20），选择逻辑 42 选择来自半速率系统的输出信号。每次校正都会丢失一个工作周期，这就是所涉及的算错损失。

图 3 示出了一个用于图 2 所示系统 30 的时序图。工作的第一个三个周期（即 0－2）用于没有错误产生的情况。单独信号时序图内的数字表示哪一个计算信号在该时刻是继续工作还是保持。在周期三 52 的末端（在星号的地方），线路 34 的信号 CL.0（半速）与线路 36 上的信号 Q-sys 的比较指示出在计算 3 有错误。系统 30 停顿一周期，使得下一状态在周期 3 停留在 3（见 54），它从 CL.0 获得的具有计算 3 的正确译本，系统利用该正确结果重新工作。在周期 3 及以后，理想的计算数字不加括号显示，实际（具有延迟）计算数字加括号显示。

半速率系统 20a、20b 不能比初始最坏情况系统速率更快，以确保同高速主计算比较的无错计算。这种解决方案要求大约三倍的原始系统硬件。

修改该解决方案以允许性能增加大于二分之一因子。因子的每个增量增加（例如从 2x 增加到 3x），必须使用另一个硬件的复制。此外，低速比较系统使用具有低速增加因子的时钟，例如在 3x 性能增加的情况下，第三时钟系统（未示出）在主系统时钟的一个三分之一频率工作。因子的每个增量增加，以周期计算的错误计算损失增加（例如 3x 情况下的损失是两个周期）。其它情况据此相应处理。注意整个系统的所有时钟是同步的。

随着频率增加，系统的基本性能增加，但是在一些点由于来自增加的误码率的算错损失导致的性能降低，抵消基本性能（时钟速率）、降低整体性能。因此，系统 30 确定最佳性能点，适合于改变条件以确定通过实际系统工作条件和组件的制造误差所能得到的最好性能。系统利用控制技术实时调整系统时钟频率。该系统的基本操作也许偏重增加时钟速率，并且接收来自时序误差检测电路中的比较器的信息。系统时钟驱动具有时钟使能功能的计数器。当检测到有

错误时，该计数器不能启动（在我们的性能双倍实施例的情况下，这是每错误一周期）。因此，计数器整个绝对的平均计数速率能够直接度量系统性能。如错误增加，计数减少，尽管在较高速率，同发明的性能一样动态变化。

计数器的平滑输出反馈回系统的时钟发生器，适当的调整时钟频率。如果平均计数器输出低，增加时钟频率（计数器输出也增加）直到平均计数器输出开始下降；然后频率逐渐的降低，增加计数器输出，直到输出再次开始降低，频率在上述点恢复一次。也就是，当性能的变量（综合的计数器输出）增加时时钟频率增加；当性能的变量降低时，时钟频率降低；当性能再次开始增加，时钟频率也再次增加。

基本数字同步系统20（图2）是可复制的，两个复制系统20a、20b（图2）的输出在每个周期进行比较。时钟频率增加，直到复制系统 20a、20b（图 2）的结果不同。系统30返回到已知的好状态，并重新工作。

该技术假定给出的统计上的波动在两个复制系统20a、20b的制造中，一个复制系统以较低频率败于另一个。如果它们以相同的方式在相同时间失败，则不会检测到错误，系统30出错。实际情况中在上述系统中有限的误差率是可以容许的。例如如果一个DSP设备正在进行图像处理，一旦在成像期间失败，可能总共约一百万象素中有一个象素有不正确的值，而没有人会注意到。

对于任务的关键应用，该技术是不适合的。然而，仅通过继续向修改的系统增加原始数字系统的复制系统以及比较所有的输出可将该技术改变为在任何可能性下起作用的。在上述方法中可以使用投票技术。这些系统已经被应用（例如在航天飞机中），但是提高可靠性胜于改进性能。因此现有系统可以根据本发明的一个方面，利用控制系统技术增加它们的时钟速率到一个可接受误差（容许）率级别来改进它们的性能。

本发明的实施方案利用与可变频率时钟发生器结合的可编程硬件进行测试以提供一组合逻辑。特别的，32 位加法器由商用的现有的现场可编程门阵列（FPGA）构成。32 位加法器的输入来自使用相同时钟的寄存器。该加法器的输出也有两个寄存器，第一个在加法器的输入寄存器加载测试数据后加载一个周期，第二个在输入被加载后加载两个时钟周期。比较器比较第一和第二输出寄存器的输出，因此有一个周期的时差。两个一位寄存器位于比较器的输出，以保存（即取样）不同时间的计算输出。因此该发明的主要基本要素模型化。

对任一情况，两个随机数同时应用到加法器的输入。加法器的输出在一和两个时钟周期后锁存。通过调整时钟频率和监视输出寄存器的结果，以及比较器的结果，确定当加法器产生正确结果时，正确/不正确操作是否被低速系统检测到（即，第二寄存器，其给加法器两次机会去计算它的结果）。整个系统由进一步验证该加法的主计算机驱动和检查。

主要的试验确定该系统能够无错误操作的频率，确切的说，是非常少的错误（所有可以容许的）。我们使用指示加法器（在该系统中，也就是，包括寄存器延迟）能在假设的最坏情况下的大约 33MHz（每秒 33 百万加法）工作的设计工具的结果做为基准频率。相应的时钟周期是大约 30 毫微秒。

试验要执行很多次。每一遍包括在随机数字上以一种工作频率实现二十次不同的加法。系统以低频率启动。时钟振荡器可从 369KHz 变化到 120MHz。主计算机设置频率，并且使用平分算法在二十次无误差的加法中很快的找到最高工作频率。

第一次运行后，工作频率确定在大约 60+MHz。然而，数据的某些方面指示出系统可能实际上工作更快；比较器实际上太慢。重新执行试验，允许比较器多次工作（但是仍然监视在最初时期定时的两个输出寄存器）。工作频率增加到大约 95MHz。因此在加法器性能中的近似三个改进中的一个因子被获得。

实现快速加法的一个问题是需要考虑在最坏情况下运载从最小有效比特（LSB）到最大有效比特（MSB）的传播。然而带有随机数的最坏情况很少发生。此外尽管运载任何特定比特的可能是 1/2，所有运载传播超过多比特的可能性随比特的数目按指数的减少。典型的 8—9 比特“最大”传播长度是估计的，我们发现试验中加数和被加数（加法器输入）的 20（20）个随机对几乎是准确的。因此最坏情况 32 比特的传播长度（在该情况下）并不是刚才所见的典型数据，并且加法器的输出实际上在比所认为的时间更少的时间内进行调整。

图 4 示出了本发明的第一可选实施方案 70。该实施例可能在门和锁存电平或寄存器电平实现。图 4 所示的系统具有硬件成本，该成本以与性能相同的速率增加（例如，大约 2x 硬件成本用于 2x 性能增加，而功率也增加两个因子）。该解决方案易于构建并且不增加在主要路径上的逻辑数量（门延迟）。这种方案应用在流水线系统功能级上。图 5 示出了一个用于图 4 所示的第一可选实施方案系统 70 的时序图。

为了描述该实施方案，当然并不仅限于此，假设系统是流水线的。在流水线系统(通常用于现在的处理器中)，原始组合逻辑的工作被分成几个部分/阶段。每一阶段在不同的时刻做部分计算工作。众所周知的，典型的流水线系统同装配线一样工作在相同的模式，即在任何给定时间许多产品正在流水线上构造，但是每个都处于不同的构造过程。尽管实现一个单独计算需要的时间相同，许多的计算能够同时进行处理，实现一种并行类型，因此能够改进性能。

参考图 4，系统 70 包括两个相同的原始系统复制，增加的比较器以及在交替系统时钟周期的定时相邻阶段。两个复制系统在相应的阶段使用互补的时钟。两个半速率时钟被一个系统时钟周期偏移，如图 5 所示。参考图 5，时序图示出没有误差发生的时序，以及部分 A 中当在 R1 输出检测到一个错误时的系统时序。(图 4)。

再次参考图 4，假设图中示出的硬件是系统整个流水线的一部分（例如，一个 Intel Pentium Π 微处理器在它的流水线包括大约十二阶段)。与其它的相比，流水线的操作允许使用较高速率的时钟。基本硬件是非基本（顶部）硬件的复制。

系统 70 的操作如下。到整个系统的输入以系统时钟速率出现。注意至少就硬件而言，没有以全速率工作的实际时钟。在交替周期输入到达每个流水线 72、74。在时间 0，通过线路 78 上的信号 clk.0 把输入锁存到锁存器 R0 76。第一个计算发生在组合逻辑块 CL1 80，并且在一个系统周期后在时间 1 被线路 84 上的信号 clk.1 锁存到 R1 82。信号 clk.0 和 clk.1 以整个系统时钟的半速率运行。

因此，在组合逻辑块 CL1 80 的计算如锁存在锁存器 R1 82 中一样需要一个系统周期。然而，块 CL1 80 直到时间 2 才改变它的输入（见图 5)。在第二周期的末端，锁存器 R1 的输出（一个周期计算时间）同组合逻辑块 CL1 的当前输出进行比较（两周期比较时间，因此保证正确答案)。如果两个结果，慢的和快的都等于线路 88 上的信号 good.1，表明快速计算是正确的，并且不需要校正。在时间 2（见图 5）来自组合逻辑块 CL2 90 的第二计算输出被锁存到锁存器 R2 92。相似的操作发生在流水线 A 阶段的其余部分，在流水线 B 中也是一样的。结果导致流水线 A（和 B）以整个系统时钟速率的半速率运行，系统时钟速率是本发明外的系统时钟速率的两倍。

然而，有两条流水线，因此在原始系统速率的 0.5*2*2=2 倍产生结果。

如果发生算错误差，我们就使用图5示出的低速时序图94。在该情况下，锁存器R1 82锁存由组合逻辑块CL1 80产生的不正确结果。这种情况在时间2的末端被比较器 87 检测到，该比较器在线路 88 上提供一个信号值指示信号good.1是错误的。因此组合逻辑块CL2 90同样有一不正确的应答，线路78上的信号clk.0在时间2中禁止用于流水线A（见图5）。组合逻辑块CL1 80仍然为输入IA1计算同样的结果，在时间3（见图5）锁存器R1 82锁存来自组合逻辑块CL1 80的正确结果。组合逻辑块CL1 80有超过两周期来计算它的结果，该结果是正确的。这个正确的结果现在用于流水线上，并且正常的高吞吐量操作重新开始。因此，流水线A遭受两个系统周期的算错损失。总的说，这也可能导致一个周期的系统算错损失，但是如果我们要求两条流水线的输出是有序的，则流水线B必须停止两个周期，因此我们假定在该实施方案中用于算错损失的是两个周期。

如果典型的延迟是三分之一原始系统的最坏情况延迟，我们将通过三分之一因子改进性能，将需要该系统的第三个复制系统，有三个运行在三分之一系统时钟速率的时钟，该系统时钟运行速率是原始系统时钟的三倍。注意运行新系统所需的功率也同样按性能增加的比例增加。算错损失也按比例增加到三个周期。

图4所示系统与图2所示系统相比，优点是图4中的实施方案不需要允许更快时钟的选择逻辑（见图2种的42），确切的说不需要增加通过一个阶段的延迟。

再次参考图4，每个周期用于新计算的输入交替进入流水线A和流水线B。相似的，整个系统的输出在每个周期交替来自流水线A和流水线B。如上描述可以注意到，两个流水线是独立的（即一个线路中的计算并不依赖于另一个线路的计算）。

图6示出了本发明的另一个实施方案100。值得注意的是，图6所示实施方案100实现用少于2x的硬件成本增加而实现2x性能增加，而功率增加四分之一因子。通过图 1 所示的普通数字系统模型，该实施方案的主要特点可以适用于所有数字系统。

参考图 6，该实施方案提前创造一个最小文本的比率管道，以不同的方式构造阶段的组合逻辑。假设图2所示的原始组合逻辑块CL平分为两个相等延迟

的组合逻辑部分 CLa102 和 CLb104（即通过二分之一因子增加流水线化）。这就允许时钟频率变为双倍的，以及利用一个二相位定时系统，隐式系统频率能够增加另外二分之一因子。然而由于我们在通过流水线的每个完整遍仅得到一个结果（即每两个隐式系统时钟周期），整个性能增加二分之一因子。

该实施方案将图 2 所示的组合逻辑块 22 分为两个块 102、104，其中每个块分别包括各自的分段寄存器 106、108，如在流水线中一样，除了在交替系统周期定时的阶段外。系统 100 也包括比较器 110、112。隐式系统时钟频率是原始系统的 4x，该解决方案的显式（物理存在）阶段时钟频率是原始系统时钟频率的 2x，与新的显式系统时钟频率相同。

系统 100 也包括误差处理逻辑 120 用于控制单元和处理误差。误差处理逻辑在线路 122 生成信号 LDR.a，线路 122 是用于寄存器 Ra106 的同步加载使能线路。当线路 122 上的信号 LDR.a 是正确的，并且寄存器 Ra 的时钟从 0 到 1，寄存器 Ra 106 被加载。因此当逻辑块 CLa 102 的输出有错误并且逻辑块 Cla 需要更多的时间去计算该结果，或者当较早阶段没有额外延迟的生成有效的结果时，该寄存器被加载。在线路 124 利用误差处理逻辑生成信号 LDRb 的技术是相似的。

图 7 示出了一个用于图 5 所示实施方案的时序图。值得注意的，两个半速时钟 clk.a 和 clk.b 被偏移了一个隐式系统的时钟周期。显式系统时钟同信号 clk.a 相同。上部的图 126（图 7）示出了没有错误发生的时序，而底部的图 128 示出了在锁存器 Ra 106（图 5）的输出检测到错误时的时序。术语“sla”指示状态 1 ，部分 a（第一个半初始状态）正在计算。

根据图 4 示出的实施方案，图 6 所示的系统性能通过增加系统部分的数目得以增加。例如，通过增加三分之一因子性能，组合逻辑将分为三个部分，每个结束于根据三相位时钟的不同相位定时的一个寄存器。

基本 32 位加法器需要相同的全部组合逻辑（组合加法器本身）和至少两个用于输入的 32 位寄存器（总共 64 位寄存器），有时候还有一个用于输出的附加 32 位寄存器，尽管在流水线系统中输出寄存器也计为下一阶段的一部分。根据图 6 中实施方式的加法器使用 92 位寄存器和三个十或十一位的比较器。粗略假定一位比较器的成本与一位寄存器的成本相同，用于图 6 实施方式的总体硬件成本等同于 125 寄存器位。

在图 8 示出时序误差避免系统 800，包括标准逻辑和模拟元件，一个用于驱动数字模拟转换器（DAC）804 的向上/向下计数器 802，数字模拟转换器 804 依次生成模拟电压以驱动在线路 808 上设置系统时钟频率的 VCO 806。在实施例系统中，计数器 802 经常变化，向上加一或向下减一。随着 VLSI 技术的进步，所有这些元件都可以用与系统相同的芯片实现。注意既然有一个来自系统时钟的显式反馈回路反馈到计数器的设置，计数器 802 的绝对值不是重要的，仅在于它能够响应来自时序检查器 810 的命令上升和下降。

时序误差避免系统 800 通过确定数字机器内部寄存器元件之间的主要路径来构造。例如在流水线 CPU，包括确定最低阶段（时钟周期确定）、通过逻辑的主要路径（最长的，时间方式），构造逻辑的一位宽文本，一位文本的输入可以从逻辑 0 到 1 变化，或者从逻辑 1 到 0 变化，并通过逻辑的结束传播到所有的路径。这个延迟测试逻辑并未连接到该机器的任何有规则的逻辑上。然而延迟测试逻辑名义上与通过机器的最坏情况路径有相同的延迟。利用交替的 1's 和 0's 来驱动延迟测试逻辑 814，后者与线路 802 上的系统时钟同步。该测试输入位置相应于在 CPU 中最慢流水线阶段的开始流水线寄存器的输出。在每个周期的末端，如果测试数据在系统时钟边缘之前没有到达流水线阶段的输出寄存器，系统运行较慢，系统时钟频率是增加的。如果测试数据到达输出寄存器，系统时钟频率接近系统的极限，因此系统时钟频率降低。

为了示出主时序误差避免电路系统的简单性，我们在图 9 中给出其实现的低电平的详细资料。交替的 1's 和 0's 通过触发器 902 的有线触发操作实现。延迟测试逻辑 814 包括通过地址多路开关的一位片、CPU 注册文件、用于 CPU 前向操作数以减少数据开销的旁路转换开关、以及通过数据路径宽度的零检测比较器。

异或门 906 使线路 908 上的延迟信号标准化以在线路 910 上出现一个到具有相同极性的时序检查器 810 的信号，而不考虑反转触发器 902 的输出。延迟测试逻辑 814 的延迟在系统设计时间内被调整为稍微大于在先前提到的给出合适安全边缘的主要路径。当高品质的逻辑模拟器用于设计处理时，这是一个相对简单的过程。在我们的实施例 CPU 系统中，在 CPU 上运行测试程序执行一个结构模拟。从该模拟中，我们获得用于无时序误差避免（基线）CPU 的最坏情况的工作频率，和检查时序误差避免逻辑性能以保证在规则 CPU 逻辑的时序

约束是违规的之前减少系统时钟频率。这保证时序误差避免。

在图 9 所示时序误差避免系统中，有一个地方可能会发生系统故障一这就是时序检查器 810 的开始，延迟的信号在这里被锁存到触发器。因为延迟信号可以随时位于任何地方，不会与线路 914 上的系统时钟同步，还有一种可能就是在信号被锁存到时序检查器 810 的同时，延迟测试信号会改变值。这可能导致在时序检查器 810 输出的亚稳定性，此时，时序检查触发器的线路 916 上的逻辑输出信号的物理值既不是 0 也不是 1。众所周知亚稳定信号会无限的停留在该状态，通过系统逻辑的其余部分指向该值。

图 10 示出了在前面的段落中增加的定址的时序检查电路 810 的实施方案。时序检查电路 810 在两个不同的时刻取样线路 910 上的延迟测试信号 D1。然后对于单周期，只有一个触发器 Q1 1002 或 Q2 1004 可能处于亚稳定状态。也就是，因为线路 910 上的延迟测试信号在一个周期中最多只改变一次值，触发器 1002、1004 不能在同一周期都处于亚稳定状态。只要频率变化增加保持合适的微值，监视触发器 Q1 1002 和 Q2 1004 以确定增加或减少时钟变化的逻辑输出在亚稳定情况开始很久后才进行取样。时序检查器逻辑确保没有亚稳定状态传播通过取样点。例如，参见图 11 情况 3，处理亚稳定情况的例子，情况 1 和 2 示出了更多典型的频率分别增加和降低的情况。

如图 10 所示，来自 VCO 的信号输入到线路 918 上。该信号输入到两个串联门延迟以在线路 920 上生成系统时钟。

时序误差避免逻辑相对便宜。例如对于 32 位 CPU，延迟测试逻辑的硬件成本少于最慢流水线阶段成本的 1/32。可变频率振荡器的增加只花费很少成本。

如果 CPU 或其它数字系统有两个或多个具有相似延迟的流水线阶段，这些都可以做为如上描述的单个阶段情况来对待，其中单个阶段情况具有来自其中具有设置时钟频率优先级的“降低时钟频率”信号。

尽管已经示出和描述了本发明有关的几个优选实施方案，其中在形式和详细内容上的各种变化、删除和增加都不脱离本发明的精神和本发明的范围。

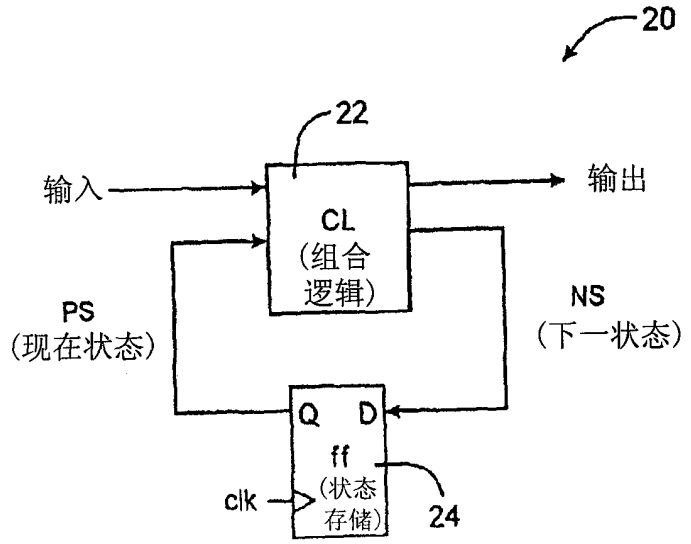


图 1

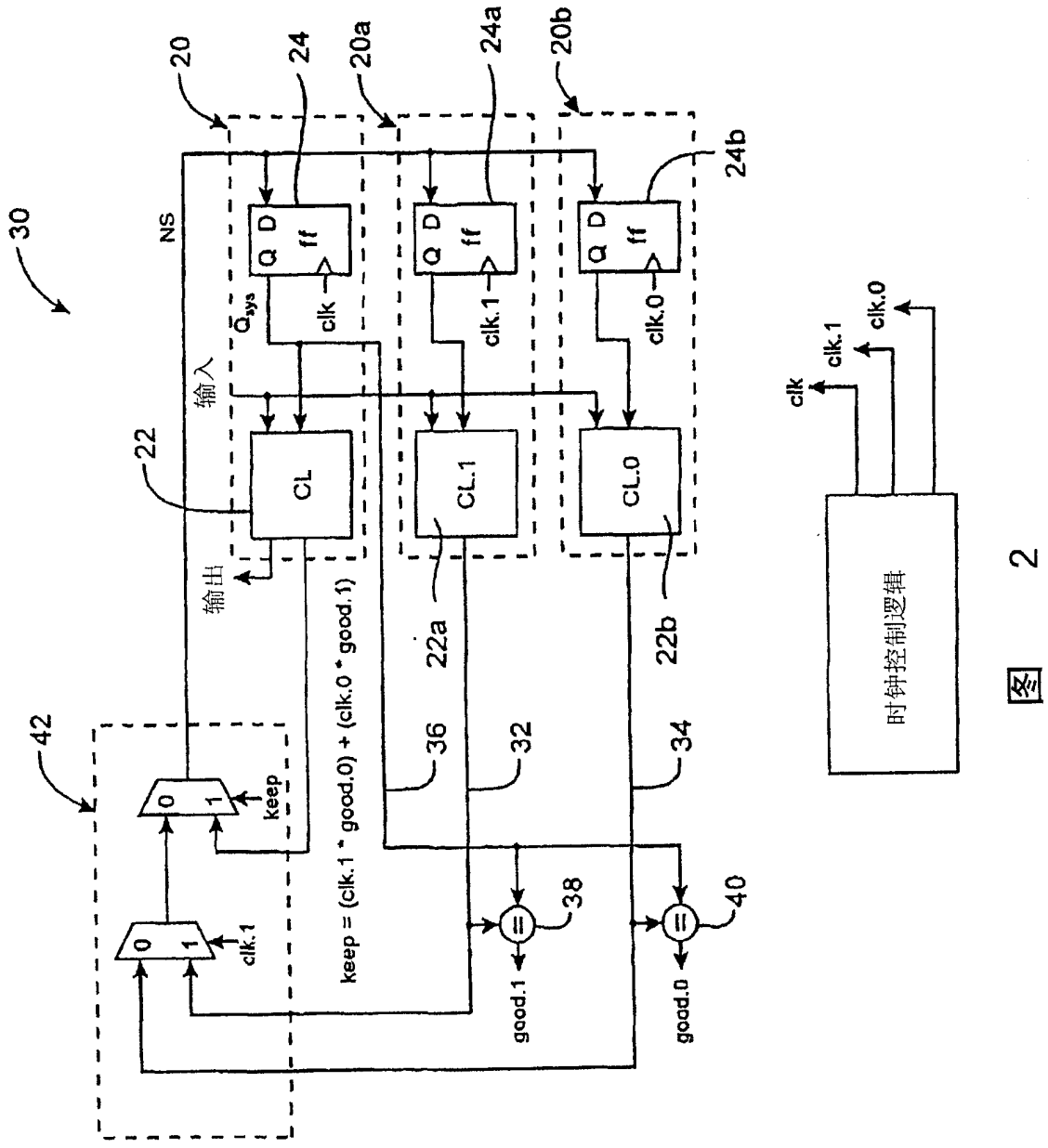

30
20
20a
20b
24
24a
24b
NS
输入
输出
22
22a
22b
CL
CL.1
CL.0
Q D ff
clk
clk.1
clk.0
Q_{sys}
keep = (clk.1 * good.0) + (clk.0 * good.1)
42
keep
36
32
34
38
40
good.1
good.0
clk
clk.1
clk.0
时钟控制逻辑

图 2

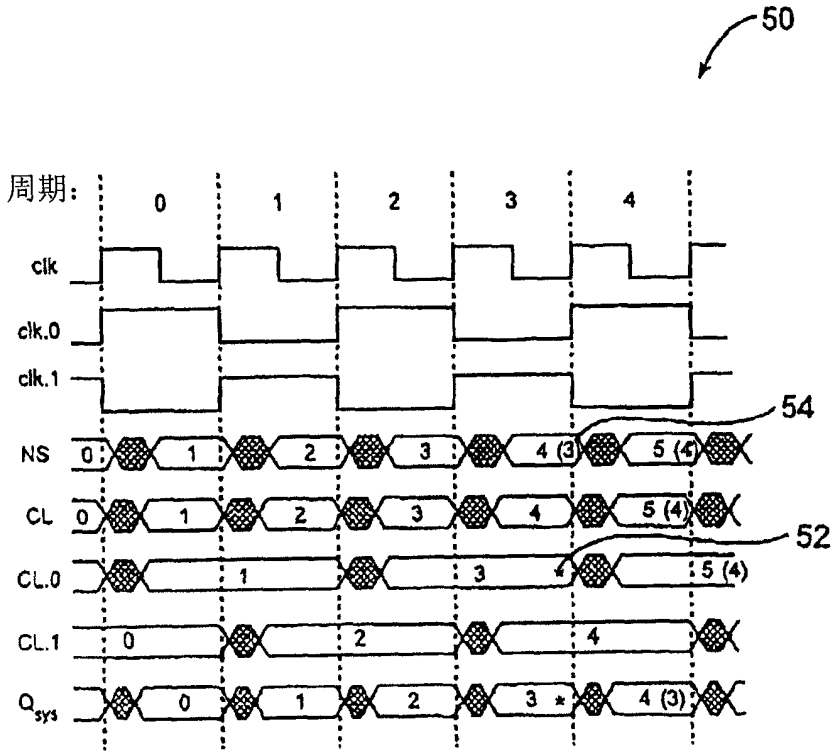


图 3

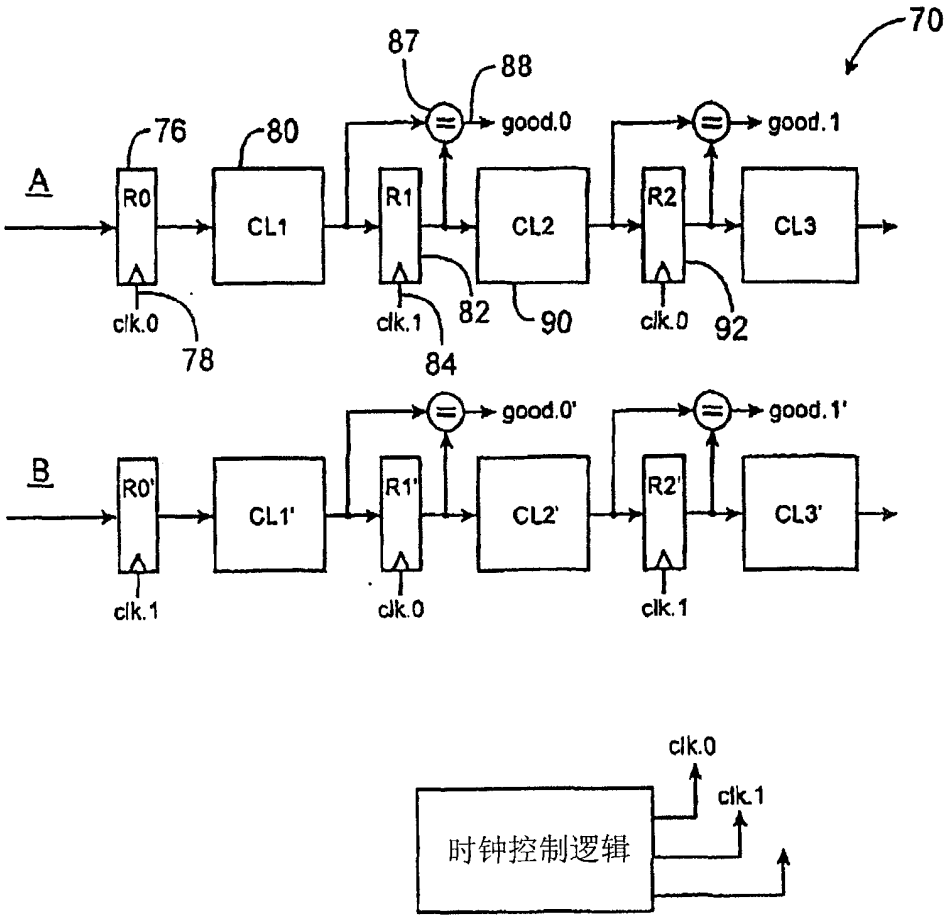

70
87
88
good.0
good.1
76
80
A
R0
CL1
R1
CL2
R2
CL3
clk.0
78
clk.1
82
84
90
clk.0
92
good.0'
good.1'
B
R0'
CL1'
R1'
CL2'
R2'
CL3'
clk.1
clk.0
clk.1
clk.0
clk.1
时钟控制逻辑

图 4

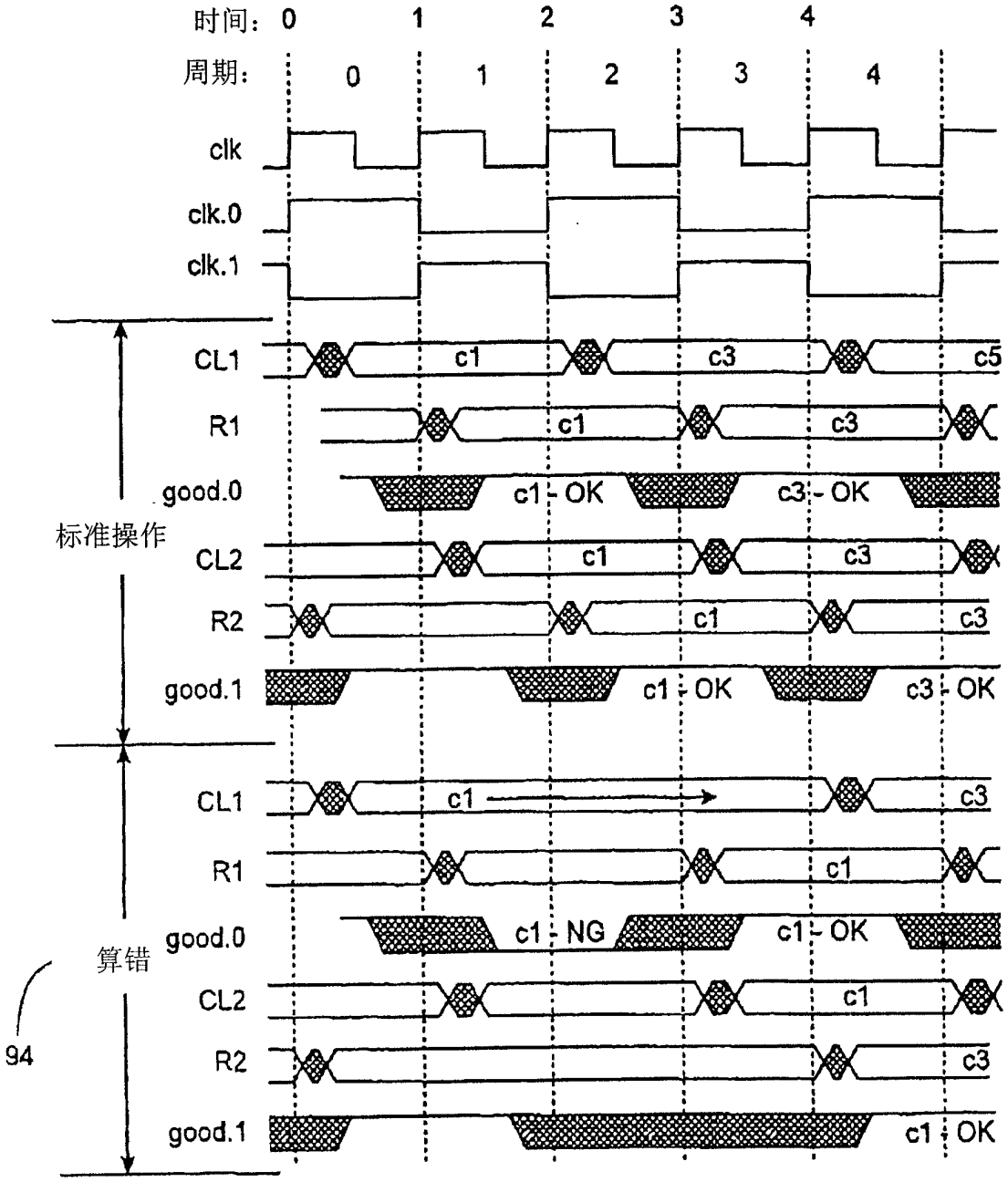

时间：
0
1
2
3
4
周期：
clk
clk.0
clk.1
CL1
R1
good.0
c1 - OK
c3 - OK
标准操作
CL2
R2
good.1
c1
c3
c5
算错
c1 - NG
94

图 5

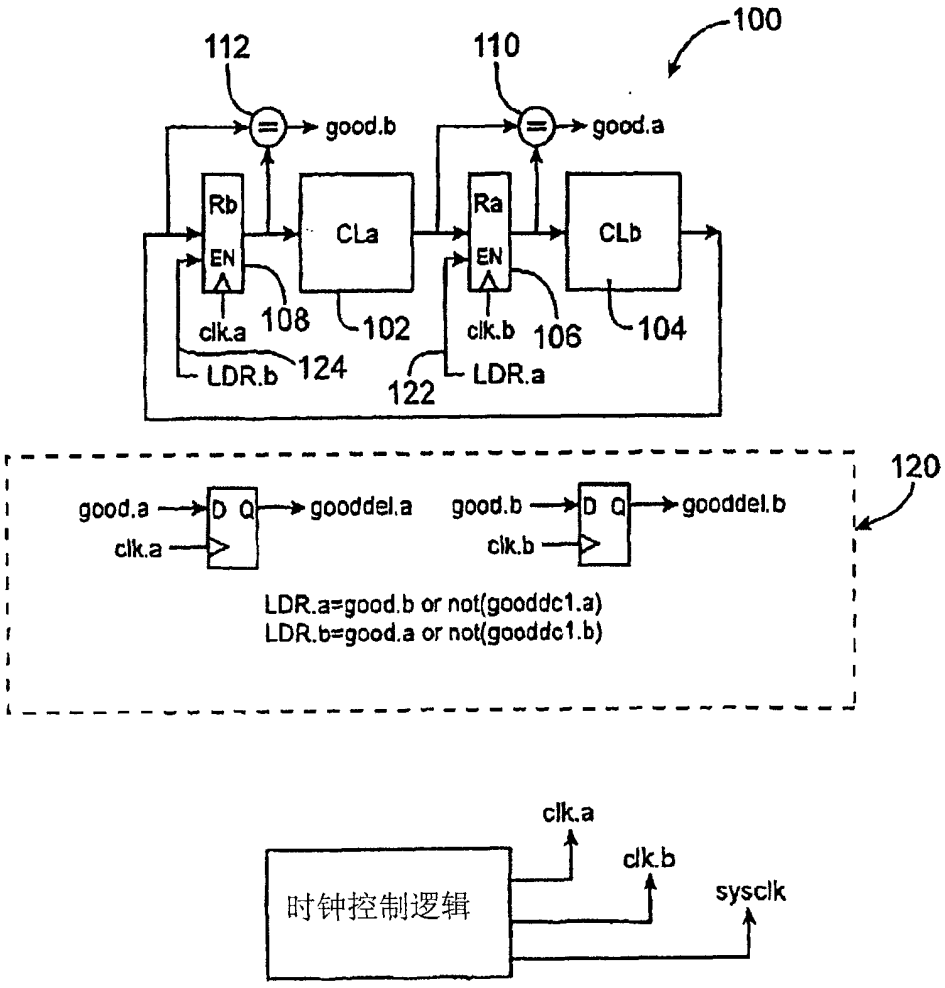

100
112
110
good.b
good.a
Rb
EN
CLa
Ra
EN
CLb
clk.a
108
102
clk.b
106
104
LDR.b
124
122
LDR.a
120
good.a
D Q
gooddel.a
clk.a
good.b
D Q
gooddel.b
clk.b
LDR.a=good.b or not(gooddc1.a)
LDR.b=good.a or not(gooddc1.b)
clk.a
clk.b
sysclk
时钟控制逻辑

图 6

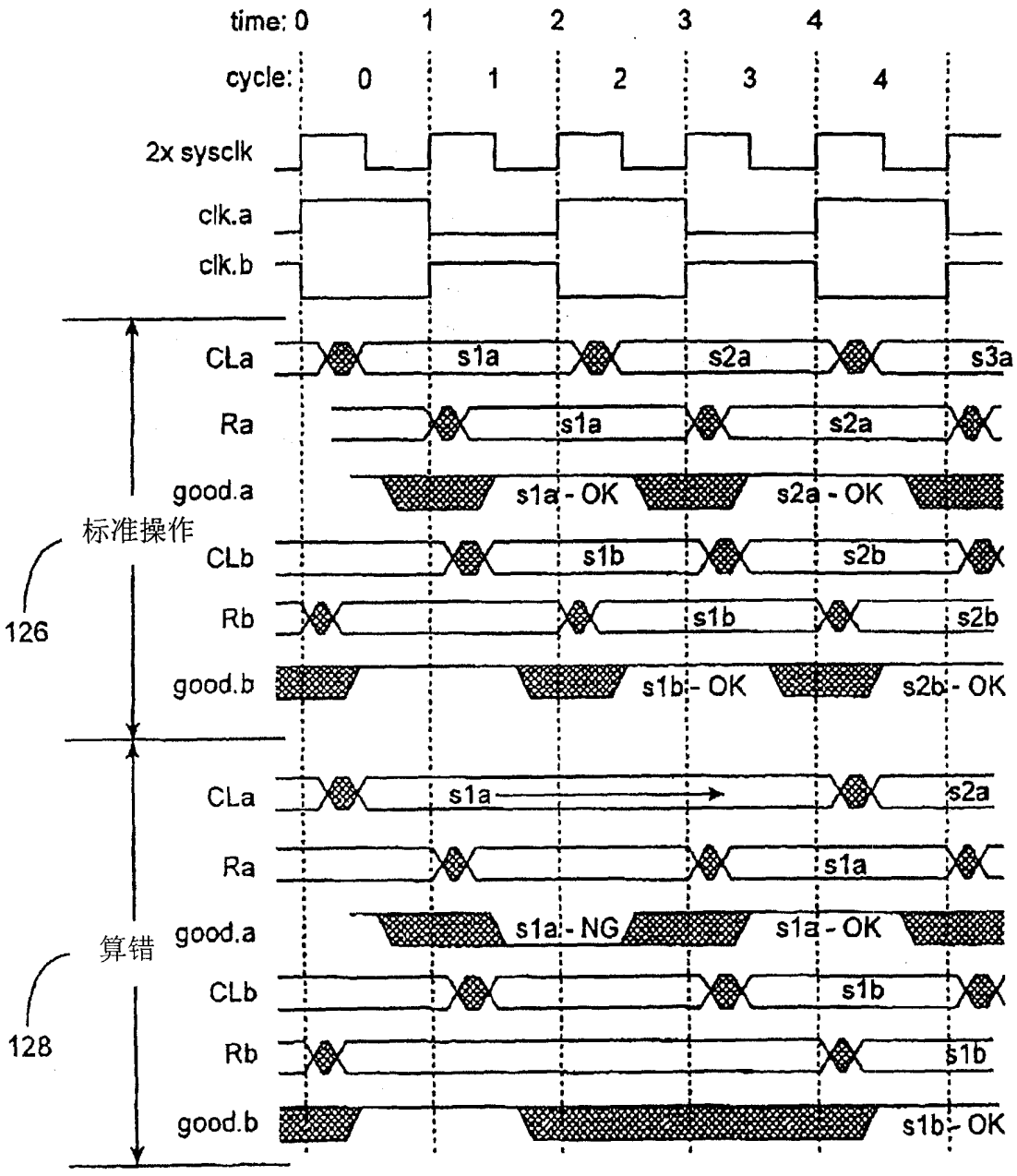

time: 0
1
2
3
4
cycle:
0
1
2
3
4
2x sysclk
clk.a
clk.b
CLa
s1a
s2a
s3a
Ra
s1a
s2a
good.a
s1a - OK
s2a - OK
标准操作
CLb
s1b
s2b
126
Rb
s1b
s2b
good.b
s1b - OK
s2b - OK
CLa
s1a
s2a
Ra
s1a
good.a
s1a - NG
s1a - OK
算错
CLb
s1b
128
Rb
s1b
good.b
s1b - OK

图 7

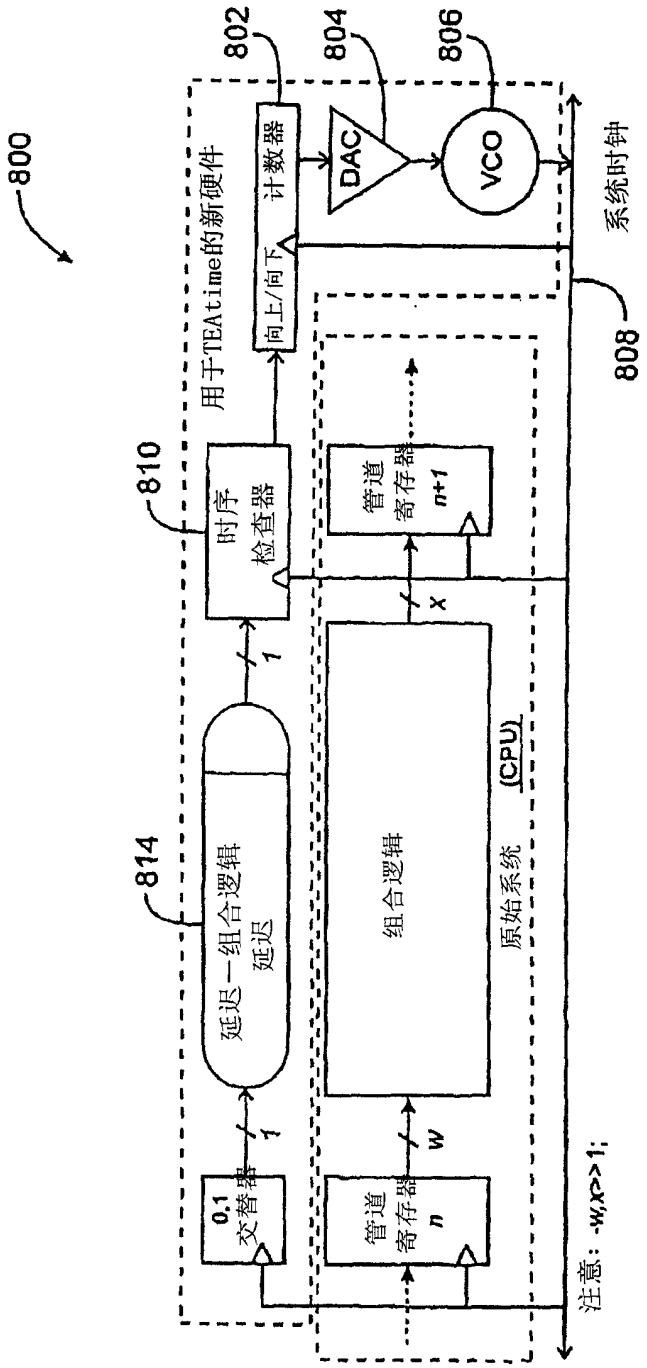

800
802
804
806
808
810
814
用于TEAtime的新硬件
计数器
向上/向下
DAC
VCO
系统时钟
时序
检查器
管道
寄存器
n+1
x
1
组合逻辑
原始系统 (CPU)
延迟一组合逻辑
延迟
0,1
交替器
管道
寄存器
n
w
注意：-w,x>>1;

图 8

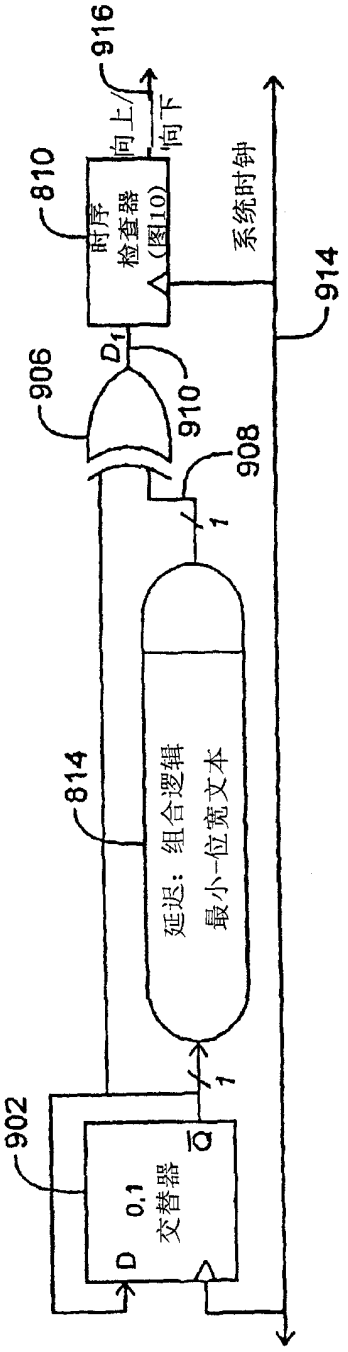

916
向上/向下
810
时序检查器(图10)
系统时钟
914
906
D_1
910
908
1
814
延迟：组合逻辑最小-位宽文本
1
902
0,1
交替器
D
Q

图 9

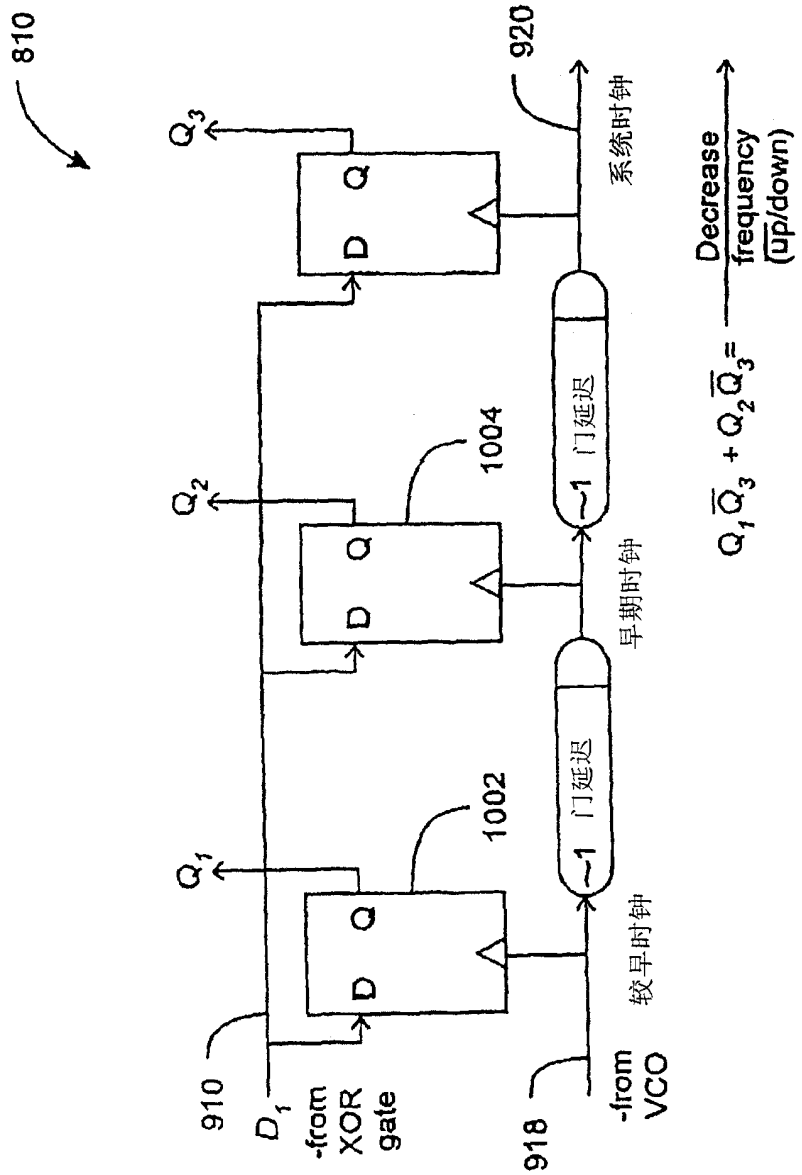

810
Q_3
Q_2
Q_1
D
Q
1002
1004
920
910
D_1
-from XOR gate
-from VCO
918
较早时钟
早期时钟
系统时钟
~1 门延迟
~1 门延迟
$Q_1\overline{Q}_3 + Q_2\overline{Q}_3 =$
Decrease frequency (up/down)

图 10

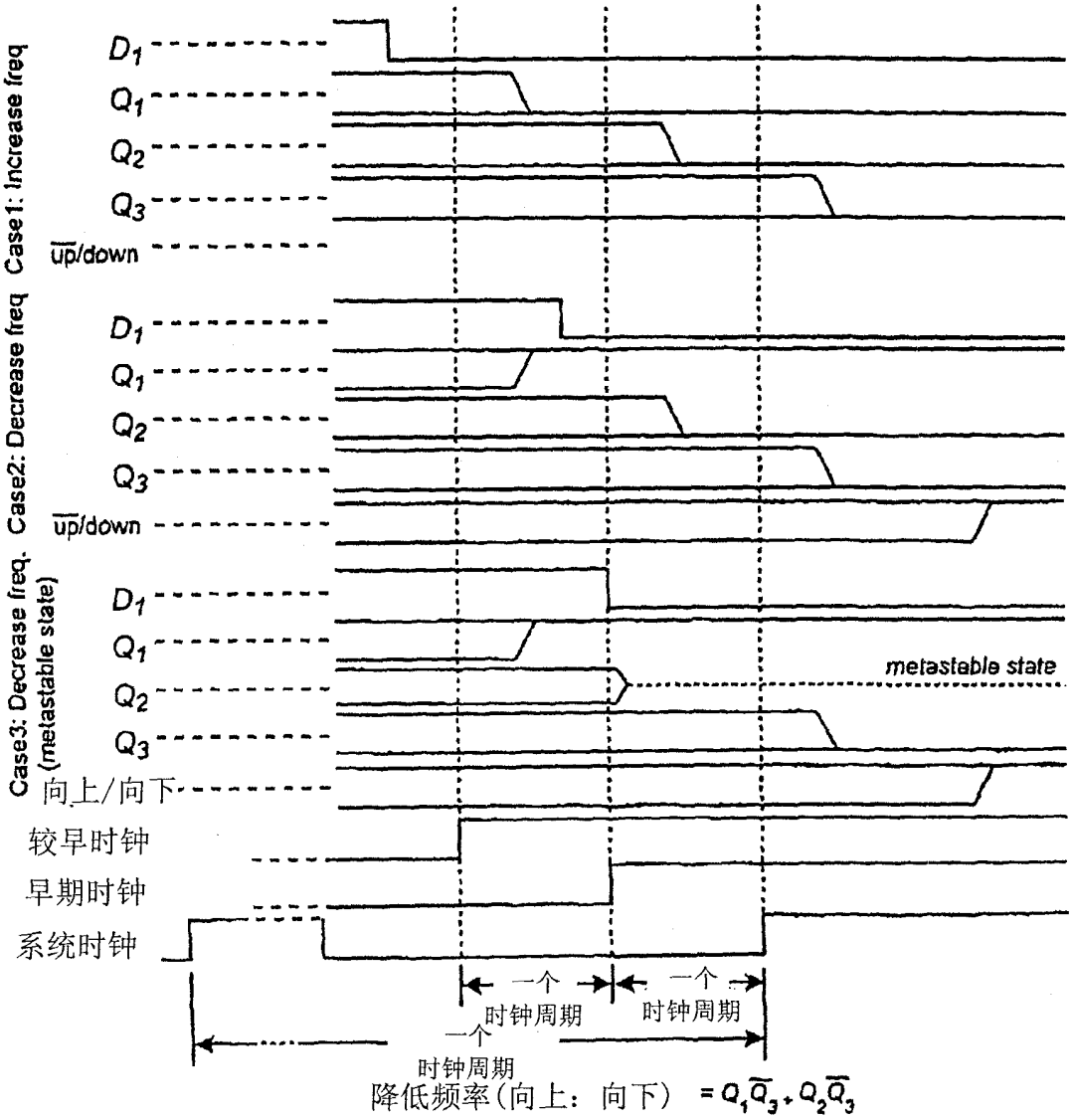

Case3: Decrease freq. (metastable state)
Case2: Decrease freq
Case1: Increase freq
D1
Q1
Q2
Q3
up/down
metastable state
向上/向下
较早时钟
早期时钟
系统时钟
一个
时钟周期
一个
时钟周期
一个
时钟周期
降低频率(向上：向下) = Q1Q3 + Q2Q3

图 11