[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
H04J 3/06
H04L 5/24

[12]发明专利说明书

[21] ZL 专利号 94104632.X

[45]授权公告日 2001 年 4 月 25 日

[11]授权公告号 CN 1065090C

[22]申请日 1994.4.25 [24]颁证日 2001.1.13
[21]申请号 94104632.X
[30]优先权
[32]1993.4.28 [33]US [31]054,332
[73]专利权人 美国电话电报公司
地址 美国纽约
[72]发明人 罗伯特·L·利恩
[56]参考文献

US 3754098	1973. 8.21	H04J3/06
US 4759041	1988. 7.19	H04L7/00
US 5142529	1992. 8.25	H04J14/08

审查员 秦力军

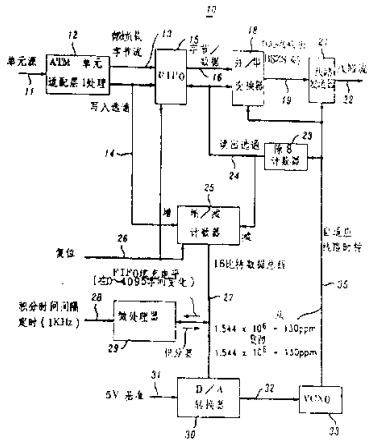
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 付建军

权利要求书 4 页 说明书 35 页 附图页数 10 页

[54]发明名称 自适应时钟恢复的方法和装置

[57]摘要

自适应时钟恢复电路从诸如异步传递方式(ATM)单元流之类一个异步的数 据包流中导出同步时钟。连续监测存储在 FIFO 存储器中的信息量值的偏差 ,在微处理器的控制下以多种方式调整称为自适应线路时钟频率的同步时钟频 率。调整是根据监测到的偏差增加状态作出的。调整是开环式的,且不需根据 监测到的偏差来连续调整自适应线路时钟频率。与常规的锁相环路相比,阻尼 效应好,可达到正确的频率后无频率摆动现象。



ISSN1008-4274

权　利　要　求　书

1．一种在一个电路配置中使用的方法，该电路配置包括：

接收装置，用以接收异步的、数据包化的信息；

存储装置，用以存储上述接收的信息；

发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送上述存储的信息；

该方法，其特征在于包括以下步骤：

连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；

检测所述监测到的偏差增加的状态；

根据所述检测到的增加状态，调整所述自适应线路时钟频率，使所述自适应线路时钟频率过校正，直至所述监测到的偏差开始减小。

2．根据权利要求 1 所述的方法，其特征在于，所述过校正包括：

使所述自适应线路时钟频率线性变化，但并不在所述监测到的偏差基础上连续地调整所述自适应线路时钟频率。

3．根据权利要求 1 所述的方法，其特征在于，所述调整还包括：

在所述过校正之后，使所述自适应线路时钟频率保持恒定，直至所述监测到的偏差减小到一个预定的阈值。

4．根据权利要求 3 所述的方法，其特征在于，所述保持恒定包括：

使所述自适应线路时钟频率保持恒定，而并不在所述监测到的偏差基础上连续地调整所述自适应线路时钟频率。

5. 根据权利要求 3 所述的方法，其特征在于，还包括以下步骤：

连续地确定视在源频率，其中所述的调整还包括：

按所述多种模式的第三种方式，改变所述自适应线路时钟频率，直至所述自适应线路时钟频率等于所述视在源频率。

6. 根据权利要求 5 所述的方法，其特征在于，使所述自适应线路时钟频率线性变化，但并不在所述监测到的偏差基础上连续地调整所述自适应线路时钟频率。

7. 根据权利要求 1 所述的方法，其特征在于，还包括以下步骤：

对所述监测到的偏差进行积分，以滤除掉数据抖动。

8. 一种在一个电路配置中使用的方法，该电路配置包括：

接收装置，用以接收异步的、数据包化的信息；

存储装置，用以存储所述接收的信息；

发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送所述存储的信息；

该方法其特征在于包括以下步骤：

连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；

对所述监测到的偏差增加状态进行检测；

根据所述检测到的增加状态，调整所述自适应线路时钟频率，但并不依据所述监测到的偏差连续地调整所述自适应线路时钟频率。

9. 一种在一个电路配置中使用的方法，该电路配置包括：

接收装置，用以接收异步的、数据包化的信息；

存储装置，用以存储所述接收的信息；

发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送所述存储的信息；

该方法其特征在于包括以下步骤:

连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差;

连续地确定视在源频率;

对所述监测到的偏差增加状态进行检测;

响应所述检测到的增加状态,部分地根据所述连续地确定的视在源频率来调整所述自适应线路时钟频率。

10. 根据权利要求 9 所述的方法,其特征在于,所述连续地确定视在源频率包括:

在所述监测到的偏差和所述自适应线路时钟频率的基础上,连续地确定所述视在源频率。

11. 一种在一个电路配置中使用的方法,该电路配置包括:

接收装置,用以接收异步的、数据包化的信息;

存储装置,用以存储所述接收的信息;

发送装置,响应一个自适应线路时钟频率,在一个同步的电路上发送所述存储的信息;

该方法其特征在于,包括以下步骤:

连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差;

根据所述监测到的偏差和所述自适应线路时钟频率,连续地确定视在源频率以用于调整所述自适应线路时钟频率。

12. 一种在一个电路配置中使用的方法,该电路配置包括:

接收装置,用以接收异步的、数据包化的信息;

存储装置,用以存储所述接收的信息;

发送装置,响应一个自适应线路时钟频率,在一个同步的电路上发送所述存储的信息;

该方法其特征在于,包括以下步骤:

连续监测存储在所述存储装置中的信息的量值与一个标称

值的偏差;

对所述监测得的偏差增加状态进行检测;

响应所述检测到的增加状态,对所述自适应线路时钟频率实行开环调整。

13. 根据权利要求 12 所述的方法，其特征在于，所述实行的调整包括:

对所述自适应线路时钟频率进行过校正,直至所述监测到的偏差开始减小。

14. 根据权利要求 15 所述的方法，其特征在于，所述实行的调整还包括:

使所述自适应线路时钟频率保持恒定,直至所述监测到的偏差减小到一个预定的阈值。

15. 根据权利要求 14 所述的方法，其特征在于，还包括:

连续地确定视在源频率,并且,其中所述实行的调整还包括:

改变所述自适应线路时钟频率,直至所述自适应线路时钟频率等于所述视在源频率。

16. 根据权利要求 15 所述的方法，其特征在于，还包括:

在所述改变时钟频率之后,对所述自适应线路时钟频率实行闭环校正。

17. 一种自适应时钟恢复装置，其特征在于包括:

接收装置，用以接收异步的、数据包化的信息;

存储装置，用以存储所述接收的信息;

发送装置，响应一个适应线路时钟频率，在一个同步的电路上发送所述存储的信息;

微处理器装置，用于（a）对所述存储装置中存储的信息的量值与一个标称值的偏差连续地进行监测;（b）对所述监测到的偏差增加状态进行检测;（c）响应所述检测到的增加状态，以多种模式调整所述自适应线路时钟频率。

说　明　书

自适应时钟恢复的方法和装置

本发明涉及通信系统。

近年来，业已研制出许多电话、图像和数据通信系统，将数字数据流编码成短的包或单元(*cell*)来取代采用同步传输。对于这种包式传输和交换技术派生出的世界性标准 。称为“异步传递方式(*ATM*)”。虽然网络正在向 *ATM* 传输发展，但对于现今的同步交换和传输系统以及对于端点终端设备来说，都需要有接口。话音和图像信道总是需要恒定比特率的 、同步的接口。从 *ATM* 或别的包式传输转换到恒定比特率同步系统的处理基本上需要两个步骤。第一步是提取出携载同步比特流的单元有效负载数据，将它存入一个先进先出(*FIFO*)存储器，该 *FIFO* 起缓冲存储器的作用，对突发单元的到达予以平滑。第二步是根据平均的数据到达比特率来恢复或导出一个时钟，并利用导出的时钟来对 *FIFO* 的数据输出确定节拍，再按节拍进入一个传输接口电路进行传输 。从到达的单元/包流的数据速率中导出一个精确时钟速率的过程，称之为自适应时钟恢复。

ATM 单元流在其单元速率方面经常具有短期变动突发性 ，

对于某些 *ATM* 系统来说，变动期的时间量级为1毫秒。所导出的(自适应的)时钟速率必须稳定，典型的稳定值必须是百万分之几(*ppm*)秒；而就长时期来说，必须精确地跟踪(数据)源速率。对于不同的系统和不同的应用场合，在要求上有较大的差异。在确定平均进钟速率方面，基础性的技术是在一段时期内对到达的 *ATM* 单元数目进行积分。长的积分时间可用于产生低抖动、窄频带的时钟输出。然而，如果在这种场合下采用"通常的"锁相环路(*PLL*)，例如这里所描述的电路配置210(图10)，则比较长的积分时间将导致许多稳定性问题。积分时间与 *PLL* 控制环路中的反馈延时直接有关系，该反馈延时会使闭环控制系统有不稳定的趋势。此外，自适应时钟的转换速率(*slew rate*)会导致更多的反馈延时，对它必须加以限制。阻尼因数将显著地造成慢响应和不稳定运行。在"通常的"*PLL* 环路中应用相位超前或多极点电路来控制阻尼是不实际的，因为对于在短时间上导出所需的相位超前信息来说，单元流有着太多的抖动。另有一条途径来说明这一点，即输入信号相位/频率信息的信噪比使得"通常的"*PLL* 中不便于应用双极点滤波器。在这种场合下，一个(极其)窄带的 *PLL* 中若不采用多极点滤波器，便不能够将阻尼因数有效地调整到能提供稳定的工作。已经发现，具有足够增益和窄带特性的"通常的"*PLL* 会振荡。

除了要解决上面所述的阻尼响应问题之外，自适应时钟恢复电路还应当：(1) 从突发的 *ATM* 输入流中得出一个低抖动(窄

带）的时钟；（2）具有良好的或近乎理想的阻尼稳定性；（3）具有快速响应，它仅受防抖动所必需的积分的限制；（4）具有足够增益（FIFO 电平控制）来满足时钟跟踪和漂移规范；（5）具有受控的时钟转换率；（6）保持一个精确的抖动补偿延时（缓冲 FIFO 存储器排队电平）；（7）可调整参数以适合于某种应用范围和需求范围。

鉴于上面的情况，本技术领域内需求一种改进的电路结构，用于从异步数据包流中恢复出同步时钟，它不象“通常的”锁相环路（PLL）结构中那样，并不依赖于用闭环调整来恢复出同步时钟频率。

为了实现以上目的，本发明提供了一种在一个电路配置中使用的方法，该电路配置包括：接收装置，用以接收异步的、数据包化的信息；存储装置，用以存储上述接收的信息；发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送上述存储的信息；该方法，其特征在于包括以下步骤：连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；检测所述监测到的偏差增加的状态；根据所述检测到的增加状态，调整所述自适应线路时钟频率，使所述自适应线路时钟频率过校正，直至所述监测到的偏差开始减小。

本发明还提供了一种在一个电路配置中使用的方法，该电路配置包括：接收装置，用以接收异步的、数据包化的信息；存储装置，用以存储所述接收的信息；发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送所述存储的信息；

该方法其特征在于包括以下步骤：连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；对所述监测到的偏差增加状态进行检测；根据所述检测到的增加状态，调整所述自适应线路时钟频率，但并不依据所述监测到的偏差连续地调整所述自适应线路时钟频率。

本发明还提供了一种在一个电路配置中使用的方法，该电路配置包括：接收装置，用以接收异步的、数据包化的信息；存储装置，用以存储所述接收的信息；发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送所述存储的信息；该方法其特征在于包括以下步骤：连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；连续地确定视在源频率；对所述监测到的偏差增加状态进行检测；响应所述检测到的增加状态，部分地根据所述连续地确定的视在源频率来调整所述自适应线路时钟频率。

本发明还提供了一种在一个电路配置中使用的方法，该电路配置包括：接收装置，用以接收异步的、数据包化的信息；存储装置，用以存储所述接收的信息；发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送所述存储的信息；该方法其特征在于，包括以下步骤：连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；根据所述监测到的偏差和所述自适应线路时钟频率，连续地确定视在源频率以用于调整所述自适应线路时钟频率。

本发明还提供了一种在一个电路配置中使用的方法，该电路配置包括：接收装置，用以接收异步的、数据包化的信息；存储装置，用以存储所述接收的信息；发送装置，响应一个自适应线路时钟频率，在一个同步的电路上发送所述存储的信息；该方法其特征在于，包括以下步骤：连续监测存储在所述存储装置中的信息的量值与一个标称值的偏差；对所述监测得的偏差增加状态进行检测；响应所述检测到的增加状态，对所述自适应线路时钟频率实行开环调整。

本发明还提供了一种自适应时钟恢复装置，其特征在于包括：接收装置，用以接收异步的、数据包化的信息；存储装置，用以存储所述接收的信息；发送装置，响应一个适应线路时钟

频率，在一个同步的电路上发送所述存储的信息；微处理器装置，用于（a）对所述存储装置中存储的信息的量值与一个标称值的偏差连续地进行监测；（b）对所述监测到的偏差增加状态进行检测；（c）响应所述检测到的增加状态，以多种模式调整所述自适应线路时钟频率。

按照本发明的一个典型实施例，这种需求得以满足，并达到技术先进；这里，存储在先进先出存储器例如是 FIFO（图 1）中的信息量值的偏离被连续地监测，并且这里的同步时钟频率（在此称作自适应线路时钟频率）受到调整，在一个处理器例如微处理器 29（图 1）的控制下可用多种模式作方便的调整。该调整是根据所监测的偏离中检测到的偏离增加状态作出的。重要点在于，所作的调整都是开环调整，不必以所监测的偏离为基础来连续地调整自适应时钟频率。与“通常的”PLL 结构相比较，阻尼实质上是增加了，这是因为，开环调整可得到具有完善的或近乎完善的非周期性阻尼也即无振荡状态的快速频率校正；而在闭环结构中，频率校正之后会跟随有阻尼振荡。

本发明的方法被应用在一种包括如下设备的一种电路安排一个：一个异步的、数据包式信息接收器，一个诸如 *FIFO* 之类的存储器来存储接收到的数据包式信息，以及一个发送器，用以根据自适应线路时钟频率在一个同步电路上发送出存储的信息。该方法包括连续地监测存储器中存储的信息量值相对于一个标称值的偏离。当检测到所监测的偏离处于增加状态时，电路结构按多种模式来调整自适应线路时钟频率。

示例性地说明，在第一模式(斜坡—模式 1)中，自适应线路时钟频率被过校正，直至所监测到的偏离开始减小。在第二模式(斜坡—模式 2 或 4)中，自适应线路时钟频率保持恒定，直至所监测到的偏离减小到一个预定的阈值。该示例性的方法还包括连续地确定一个视在源频率 。在第三模式(斜坡—模式 3 或 5)中，使自适应线路时钟频率改变，直至自适应线路时钟频率等于视在源频率。该视在源频率是根据所监测到的偏离和自适应线路时钟频率确定的。在斜坡—模式 1 到斜坡—模式 5 中所作的调整 ，都是开环调整，也即并不是基于所 监 测到的偏离来连续地调整自适应线路时钟频率而实现调整的。将监测到的偏离进行积分以平滑掉数据抖动。为使处理时间最小，在算法中不采用乘法。大部分的除法是以 2 的幂作除数，它们被汇编为左移操作。

图 1 示出本发明的一个示例性自适应时钟恢复电路图；

图 2 和图 3 示出图 1 电路配置的响应曲线图；

图 4 至图 6 示出图 1 电路配置中所包括的微处理器执行程序的软件流程图；

图 7 至图 9 示出图 1 电路配置的附加响应曲线图；

图 10 示出现有技术中“常规的”锁相环电路图。

图 1 是一个示例的自适应时钟恢复电路 10 的电路图，该电路在一个接口上使用，它从线路 11 上的 155*Mb/s*(兆比特/秒)的异步传递模式（*ATM*)单元流接口到线路 22 上同步的 *DS*1、1.544*Mb/s* 的恒定比特率电路中 。(其它实例的同步数据率包括 *DS*3＝44.736*Mb/s*、*CEPT*1＝2.048*Mb/s* 、*CEPT*3＝34.368*Mb/s*。)如图 1 所示，电路 10 包括有用于将信息从 *ATM* 单元流传送到同步电路的硬件。在另一方向上—从同步电路 去 往 *ATM* 单元流—传送信息所需的硬件在当前的叙述中并不重要，所以图 1 中未示出。由于线路 11 上输入(*incoming*)单元流的突发和异步性质，故而在线路 22 上同步地传送信息所需的时钟想要应用线路 11 单元流中的边沿信息或过渡信息来导出是做不到的。线路 11 上的每个 *ATM* 单元是 53 字节的数据包，包括 5 字节的首标、1 字节的适配层和 47 字节的可用信息有效负载。每个 *ATM* 单元代表 155*Mb/s* 上的一个 53 字节字符组；各单元异步地到达，一般有比较长的开区间例如 8 至 243 毫秒间隔着。电路 12 执行 *CCITT* 适配层 1 处理，包括去除 5 字节的单元首标和 1 字节的适配层，并应用线路 14 上的写入选通径由字节母线 13 来控制 47 字节有效负载去写入

一个先进先出(*FIFO*)存储器 15 。*FIFO*15 例如是集成器件技术公司(*IDT*)的存储量达 4096 个 8 比特字节的 72241，该存储量完全足以存储在线路 22 上积累着等候传输的全部字节数目。*FIFO* 15 与一个例如 *Fairchild* 公司的 *F*579 增/ 减计数器 25 协同工作，该增/减计数器 25 在任一时刻计数 *FIFO* 15 中存储的字节数目。从电路 12 上每次给 *FIFO* 15 写入一个字节时，线路 14 上的写入选通使计数器 25 增加 1。从 *FIFO* 15 中每次读出一个字节时，线路 24 上的读出选通使计数器 25 减少 1。

在上述作了说明的、从线路 35 上导出的自适应线路时钟，用来控制从 *FIFO* 15 向 *DS*1 也即线路 22 上的同步电路传输字节。在本实施例中，线路 35 上的时钟频率可在 1.544×10^6 *b*/s—130*ppm* 到 1.544×10^6 *b*/s+130*ppm* 之间变化 。对于同步电路 *DS*1 来说，这个变动范围是容许的。线路 35 上的这一自适应线路时钟频率用来操作一个例如仙童公司的 *F*323 并/串变换器 18 和一个线路发送器 21 。该线路时钟频率由一个例如 *Fairchild* 公司的 *F*161*A* 计数器 23 除以 8，在线路 24 上得到的字节时钟用作读出选通，以便经由字节母线 16、并/串变换器 18 和线路 19 从 *FIFO* 15 实现有效负载信息的字节读出，以便由线路发送器 21 在线路 22 上作为一个 *DS*1 电路进行传输 。发送器 21 将一个 *B*8*ZS* 码插入线路流中，以防止有 7 个以上接连的 0 比特在线路 22 上传送出。如前面所说明，线路 24 上的读出选通还用于使计数器 25 递减计数。

微处理器29例如是*Motorola*公司的68070，对于在线路35上导出自适应线路时钟来说，它是电路结构10中一个重要的部分。根据线路28上接收到的1*KHz*积分时间定时，微处理器29每毫秒一次地执行一个程序（图4至图6的流程图）。程序的执行要用250毫秒左右的时间。程序输入数据是经由16比特数据母线27从计数器25上读出的*FIFO*填充电平。该*FIFO*填充电平是在0与4095之间的一个12比特数目，它代表在*FIFO* 15中存储的字节数目。程序输出数据是一个"积分器(*integrator*)变量，它经由母线27传送到一个数/模(*D/A*)转换器30，例如模拟器件公司的8412，该*D/A*转换器30通过线路31连接5*V*基准电压；线路32上的*D/A*转换器输出电压在0*V*至5*V*间变化。*D/A*转换器30通过线路32向一个压控晶体振荡器(*VCXO*)33提供控制用输入信号，该*VCXO*33例如是*AT&T*公司的*S*型*VCXO*。如果*D/A*转换器30在线路32上产生出5*V*电压作为对*VCXO* 33的控制输入，则*VCXO*33会在线路35上传送出一个频率为1.544*Mb/s*+200*ppm*的自适应线路时钟。当*D/A*转换器30在线路32上产生出2.5*V*电压时，*VCXO*33传送出1.544*Mb/s*的频率。而若*D/A*转换器在线路32上给出*OV*电压时，*VCXO* 33将传送出1.544*Mb/s*−200*ppm*的频率。在本实施例中，线路32上的控制用输入信号并非在0*V*与5*V*之间变化；其电压的变化是使线路35上的自适应时钟频率被控制在1.544*Mb/s*−130*ppm*与1.544*Mb/s*

+130*ppm* 之间。线路 32 包括一个防混叠滤波器(图 1 中未示出)，用来除去线路 32 上小的阶梯函数效应。由于不需在线路 35 上以高速率转换时钟频率 ，所以防混叠滤波器具有较大的 *RC* 时间常数，例如 $R=2.2K\Omega$，$C=33\mu F$。

时基抖动通常是模拟通信线路失真的一种类型，它是由信号相对于基准定时位置有变动造成的，将会导致数据传输误差，尤其是在高速传输时。这种变动可以是幅度上的、时间上的、频率上的或相位上的。在当前的应用中，抖动更具体地指的是期望的(定期的)单元流到达时间与实际的单元流到达时间之间的差别。电路结构 10 是按线路 11 上的入局单元流有最坏情况的抖动设计的 ，即从 *DS*3 速率时的 0.3 毫秒到 *DS*1 速率时的 3 毫秒。在 *DS*3 之类高速率上的限制因素 ，是 *FIFO* 的规模。抖动产生的原因有：1)在不同的等时 *ATM* 源之间由速率差拍造成的群聚效应 ；和/或 2)在一个 *ATM* 交换系统中由多径造成的统计排队延时，它们被其它业务暂时地中断。一般，群聚效应比统计排队引起的延时量小；群聚延时是有规律地发生的。

图 2 和图 3 是用来帮助理解电路配置 10 的工作的响应曲线图。图 2 和图 3 中，在 90 秒时间段上画出了三个程序变量"*fill—level—err*"(填充—电平—误差)、"*integrator*"(积分器)和"*integrator—float*"(积分器—浮动)。变量"填充—电平—误差"代表 *FIFO* 中存储的信息量值对一个标称值的偏离。图 2 是对初始源

时钟误差的响应(*FIFO* 过填充状态)，初始状态是自适应时钟频率(“积分器”)比源时钟慢 60*ppm* 。所以，*FIFO*15 中存储的字节数目在标称值之上增加到最大值 30 字节(*FIFO* 过填充状态)。当 *FIFO* 15 中的字节数目开始增加时，电路 10 被置于斜坡—模式 1，自适应时钟频率(“积分器”)增加(过校正) 到 超过源时钟 50*ppm* 的一个点上 。当 *FIFO* 填充电平误差(“*fill—level—err* ”)增加到 30 字节时，便减少一个规定的量值(“*DEF—pole*(极点)—2*D*”=5 字节)，电路 10 被置于 *ramp—mode*(斜坡—模式)4 ，自适应时钟频率(“积分器”)恒定地保持于源时钟之上 50*ppm* ，以对 *FIFO* 15 给出时间来清除些＋30 字节。当 *FIFO* 填充电平误差减少到一个预定的阈值(“*DEF— ramp —db*”=10 字节)时，电路配置 10 被置于斜坡—模式 5，自适应时钟频率(“积分器”)线性地下降(其速率比例于前面所需的过校正量值)到一个值，它等于视在源频率 (“*integrator—float*”积分器—浮动”)，然后电路配置 10 被置于 *ramp—mode* (斜坡—模式)0。快速的线性下降产生出减小了长度的校正周期 。自适应时钟频率(“积分器”)的下降到 0 是在离坐标源点正好不到 8 秒时发生的。(程序还包括防护性校验，如果视在源频率 (“积分器—浮动”)并非可靠地估值，则程序将停止自适应时钟频率 (“积分器”)的下降。)需要指出，这里不存在进一步的振荡—这可称为完善的非周期性阻尼 。“填充—电平—误差”和“积分器—浮动”两条曲线均随时间逐渐下降(图 2 中示出 90 秒的

总时间)。在斜坡—模式0中，程序包括了很慢的校正，它能在1.0—1.5分钟内将*FIFO*填充电平误差校正为几个字节，使得自适应时种频率(“积分器”)能准确地等于源频率。

图3示出对初始源时钟误差的响应(*FIFO*欠填充状态)，初始状态是自适应时钟频率(“积分器”)比源时钟快60*ppm*。所以，*FIFO* 15中存储的字节数目减少到比标称值低达30字节处(*FIFO*欠填充状态)。当*FIFO* 15中的字节数目开始减少时，电路配置10被置于斜坡—模式1，自适应时钟频率(“积分器”)下降到低于源时钟50*ppm*的一个点上。当*FIFO*填充电平误差(“填充—电平—误差”)下降到—30字节时，便增加一个规定的量值(“*DEF*—*pole*(极点)—2*D*”=5字节)，电路配置10被置于斜坡—模式2，自适应时钟频率保持恒定，保持在源时钟之下50*ppm*，以对*FIFO* 15给出时间来存储附加的字节。当*FIFO*填充电平误差改变到标称值之下一个预定的阈值(“*DEF*—斜坡—*db*”=10字节)时，电路配置10被置于斜坡—模式3，自适应时钟频率(“积分器”)线性地增加(其速率比例于前面所需的过校正量值)到一个值，它等于视在源频率(“积分器—浮动”)，然后电路配置10置于斜坡—模式0。快速的线性增加产生出减小了长度的校正周期。自适应时钟频率(“积分器”)的增加到0是在离坐标原点正好不到8秒时发生的。(程序还包括防护性校验，如果视在源频率(“积分器—浮动”)并非可靠地估值，则程序停止自适应时钟频率(“积分

器”)的增加。)需要指出，这里不存在进一步的振荡，即具有完善的非周期性阻尼 。“填充－电平－误差”和“积分器－浮动”两条曲线随时间逐渐上升 (图 3 中示出 90 秒的总时间)。在斜坡－模式 0 中，程序包括了很慢地校正，它能在 1.0—1.5 分钟内将 *FIFO* 填充电平误差校正为几个字节，使得自适应时钟频率(“积分器”)能准确地等于源频率。

FIFO 15 的标称填充电平为 7 个 *ATM* 单元 ，或即 7×47＝329 字节。*FIFO* 15 中 存 储的 *ATM* 单元最大数目是 60 单元或即 60×47＝2820 字节 。标称填充电平是在对付抖动上和对付起动所致偏差(初始时钟捕捉)上所需的最小值，因而有着最小延时。对视在源频率(“积分器－浮动”)的估值应用了自适应时钟频率(“积分器”)和 *FIFO* 填充电平误差(“填充－电平－误差”)两者的加权组合。实现这估值主要是去控制自适应时钟频率(“积分器”)的阻尼。需要指出，视在源频率 (“积分器－浮动”)比之自适应时钟频率(“积分器”)变动得更慢。这部分地是基于如此的假定，即源时钟频率的变动恰是十分平缓的。自适应时钟频率(“积分器”)用来校正 *FIFO* 偏差，此类偏差起因于 ：1) 与源频率的失配；2) 在 *ATM* 网络中数据的得益或损失 。*FIFO* 填充电平误差 (“填充－电平－误差”)与自适应时钟频率(“积分器”)协同地起响应。

自适应时钟恢复方法是在微处理器 29 (图 1)中执行的一个数字信号处理程序，去控制可变的晶体振荡器(*VCXO*)33。该程序以

1KHz 的取样频率执行 。自适应时钟提供“通常的”PLL 功能，但具有不能由“通常的”PLL 精确跟踪的输入信号条件。在自适应时钟恢复电路配置 10 中 ，FIFO 15 的填充电平对程序提供输入。如果对突发的数据流能施加以充分的积分(截止频率为 1Hz 左右的低通滤波器)，则能 从突发的数据流中确定出准确的源时钟速率 。采用数字积分法 ，容易做出这低通滤波器。滤波器/积分时间由输入抖动、锁定时间、时钟抖动和漂移要求等来确定。

实现自适应时钟恢复时的困难点在于，以良好的阻尼和稳定性来作出窄带滤波器 。“通常的”PLL 在时钟校正与检测得的响应之间有很大的相位/时间滞后 。FIFO 在电路中的作用象一个弹簧，对校正 用反馈管加以延时 。FIFO 中的抖动使电路不能识别出小的校正。长时间常数的积分滤波器缓解了这种环路延时问题。另外，不象“通常的”PLL，电路结构 10 中的 FIFO 15 能记忆过去的时钟误差和丢失的数据。这些过去的状态必须由时钟摆动的过校正来予以校正，以使 FIFO 正常化。在这种类型的应用中，“通常的”PLL 超前/滞后阻尼滤波器是无效的。

本示例性实施例的算法，开发出来用于自适应时钟恢复和解决上面说明的环路阻尼与稳定性问题 。首先，确定出三个输入信号，它们是视在源频率 、FIFO 填充电平误差和误差方向(增加/减少)。然后，对当前的时钟频率误差和 FIFO 电平状态计算校正量。这种校正是按照一个开环的、按比例地斜坡升和斜坡降的

VCXO33 控制信号来执行。对校正的速率(斜坡斜率)、幅度和时间作出计算以便校正误差，而不需要自 FIFO 电平误差来的连续的反馈；这称作开环调整。(斜坡一模式 1，2，3，4，5 对应于开环调整。)在校正周期的终端，FIFO 的电平和时钟一般处在它们的静态零点上，即使时钟在校正周期中坡斜得过陡也是这样。当精确地确定出误差时，总的阻尼响应是非周期性的(无下冲或上冲)。

对 FIFO 电平误差信号进行积分，以提供出对数据抖动附加的滤波。这一积分时间是个重要的参数，决定了自适应时钟的部分阻尼响应。由于自适应时钟斜坡升/降的控制判决是在 FIFO 电平误差数据的模糊逻辑上作出的，所以在自适应时钟恢复电路配置 10 变到失锁之前，应用若干个探索性检验和校正来检测和防止错误的时钟斜坡变动。

在模拟运行中，对于 1.5 至 45MHz 的时钟速率已经达到了优良的时钟稳定性、阻尼响应和 FIFO 电平控制。在时钟输出中完全不存在抖动。

对于有着大的抖动的输入数据流，要想从其中直接导出具有极小抖动的精确时钟时，示例的自适应时钟恢复方法对那样的系统是很有用的。对于输入信号中伴随着噪声，在控制响应上有大的滞后的其它一些系统，该方法也很有用。对此方法可加以比例化和协调化，以在宽广的范围内应用。

图 4 至图 6 示出由微处理器 29 执行的自适应时钟程序所用的

软件流程图，程序的作用和算法将在这里说明。请注意，图 4 至图 6 中的程序框参考在后面列出的程序文本中的具体代码行号。

自适应时钟程序在 1 毫秒时间间隔上执行 。这一速率选择得能提供出匀滑的、4096 步控制的 *VCXO*，因而接近于模拟控制电路。程序可以用 0.9 至 1.1 毫秒范围内的平均执行间隔来运行，在这样的速率上对抖动变化并不敏感。重要的是应该知道，图 4 至图 6 的流程图及这里相应的叙述是说明在一段时间上程序的运行；它们示明了单个程序以上的程序执行。

可变的“*clk—tic*”(图 4，框 101)是个 32 比特计数器，在每个执行时间间隔计数值加 1，在整个程序中它用作一个定时器。由定时时标的一个二—十进制变换来触发事件和过程算法，而定时时标是按一种协调参数作出规范的。例如，一个 1 秒事件的定时以二—十进制时标参数 1024—1 来实施，它给出 1024 个 1*ms*“*clk—tic*”的时间。

程序只有一个输入量(框 102)，它是外部缓冲存储器 *FIFO* 15 中的字节数目 。从外部硬件增/减计数器 25 来的含有 *FIFO* 填充电平的一个读出值装入可变的“*FIFO*”。*FIFO* 15 必须容量足够大，能缓冲吸收最坏情况的单元延时，并加上时钟捕获时间期间供排队用的宽裕量。

由程序流程中的第一算法在变量“积分器—浮动”中产生一个值，依据积分器计数，该变量“积分器—浮动”总包括含有一个估

值的原始频率，或者视在源频率。这一算法也可以在程序的终端放置和执行。

理想上，原始频率应当等于*FIFO*数据到达比特率，它即是视在源频率 。在程序的静态或锁定状态(斜坡一模式 0)中，“积分器一浮动 ”的值等于变量“积分器”。“积分器”变量提供出主(极点 1)积分累加器功能，并直接控制*VCXO* 33 的频率。“积分器一浮动”的值被显现为“积分器”和“填充一电平一误差”中两值的加权和比例组合 。参见程序文本中确切的逻辑。变量“积分器一浮动”是另一个积分累加器 ，它的积分时间常数大约是极点 1 变量“积分器”的两倍。在一个开环校正周期的终端，它达到其最终值，从而应等于“积分器 ”中的值。变量“积分器一浮动”不对“积分器”的较快的变化起响应。变量“积分器”较快地响应于源频率变化的超前校正 ，并且就这一个或者就数据损失使*FIFO* 电平正常化 。由“积分器 ”控制的*VCXO* 33 时钟也被过校正一小段时间，以在频率变化之后使*FIFO* 15 正常化。变量“积分器一浮动”不被过校正，但就在校正周期结束时 ，它达到其新的基线电平。主*FIFO* 和时钟校正环路用“积分器一浮动”来确定在*FIFO* 电平校正作出之后它返回到(或者向下坡斜到)何处。当时钟发生变化而仅使*FIFO* 电平正常化之后，这个算法又使“积分器一浮动”返回到原来的原始源频率上。从图 8 中可以看到这一点 。(其它传输电路中的误差状态会使得有数据加上或丢失，而源时钟频率方面没有变化。)

对产生出变量“积分器—浮动”的诸参数，是由画出“积分器”、“积分器—浮动”和“填充—电平—误差”的响应曲线，并将它们调整到得出填充—电平—误差和积分器（*VCXO* 33 频率）变量的最大阻尼（下冲/上冲最小或没有）来确定的。“积分器—浮动”达到其新的原始值，与“填充—电平—误差”的达到零相一致。示出这种交点的曲线示于图 2 和图 3。

框 103 的算法连续地对视在源频率估值，以便在作出任一个开环时钟或 *FIFO* 电平调整之后，能使电路返回到正确的频率上。该算法用作整个电路方面阻尼因数的主要控制。

流程图中的下一个算法（框 104）实现一种预积分或抖动平滑操作。它滤除掉出现在变量“*FIFO*”之内大部分的数据到达（*FIFO* 填充电平）抖动。该算法的输出是一个称为“*FIFO*—浮动”的新变量。在每一个由参数“*DEF*—浮动—惯性（*inertia*）—时标（*mask*）”规定的周期内，“*FIFO*—浮动”跟踪“*FIFO*”1 字节的数目。对于 *DS*1 速率的时钟实现，该参数适定于八进制 37，每 32 毫秒它给出一个事件。“*FIFO*—浮动”中每 32 毫秒内 1 个数据字节的最大移动等同于 *DS*1 时钟的变化为 162*ppm*/s。在这一处理中，超过 162*ppm* 的所有抖动变动都被平滑掉或弃置不顾。此种操作对实际的 *FIFO* 电平没有影响，而只影响由程序的其余部分处理的视在电平。该参数必须设定到这样的值，它能使视在的 *FIFO* 速率变化大于 *VCXO* 33 的最大转换率，例如是该转换率的两倍。

框104的方法可预滤掉数据到达的抖动，并跟踪*FIFO*数据电平变动于额定的速率变化之下，单位为*ppm/s*。这方法免除了用前端*FIFO*来匀滑数据到达的抖动的需求。

下一个重要的程序语句（框105)使变量"填充—电平—误差"初始化，这是从"*FIFO*—浮动"中减去参数"*DEF*—正常(*normal*)—填充—电平"。变量"填充—电平—误差"是一个带符号的值，表明*FIFO*电平对预定标称值的偏离。在程序的其余部分中它用作开环误差信号。正的值指明*FIFO*电平在增加，*VCXO* 33必须变化到一个较高的频率上来使之正常化。

下面的算法(框106和框107)通过计算出的时钟校正来产生主要控制状态以供该开环应用。

由框106的算法判定，将视行的"填充—电平—误差"与"旧(*old*)—填充—误差"中先前的误差状态相比较时，填充电平误差是否在减小。依靠仅对*FIFO*填充电平中的变化起响应以在这一检测中将滞迟作用组合进来，而这类变化是指超过参数"*DEF*—极点—2*D*"中额定字节的数目。这种滞迟作用减小了由余留在变量"填充—电平—误差"中的数据到达的抖动造成的错误检测之数目。然后，由填充电平误差是否在减小的检验（框106)判定自适应时钟变化的方向。对于*FIFO*欠填充状态，自适应时钟将处在比源时钟低的频率上(积分器值在中心频率点之下)；对于过填充(正)状态，自适应时钟将处在比源时钟高的频率上(积分器值在中

心频率点之上)。如果在前面的执行期间填充电平误差在减小，则除了将“旧—填充—误差”更新为“填充—电平—误差”中的当前误差之外，不发生其它作用。然而，如果“斜坡—模式”中先前的状态等于1(增加模式)，则这一事件变为增加—减少的变化。

“增加—减少”事件有重要意义，因为它判定，自适应时钟*VCXO* 33已超过“视在”源时钟频率。这一事件用来启动斜坡—模式2或4，它将会停止进一步(增加)的时钟校正，并保持住“积分器”中的视行频率。对于欠填充的*FIFO* 15，变量“斜坡—模式”设定于状态2；对于过填充的*FIFO* 15，变量“斜坡—模式”设定于状态4。变量“斜坡(*ramp*)—斜率(*slope*)”被初始化以控制这样的速率，即在此速率下时钟返回(斜坡降)到原始频率。(要回想到，时钟已过校正得使*FIFO* 15正常化。)对存储于变量“斜坡—斜率”和“填充—误差—*dx*”中的斜坡斜率速率进行计算，成为由参数“*DEF*—*div*(除法)—*X*”确定的“填充—电平—误差”的一个分数率。斜坡降的斜率是与这一事件起始处*FIFO*电平误差成比例的。

框106的方法指出，*VCXO*时钟校正在何时等于视在源频率(数据所达速率)。框106的方法还计算出这样的速率，在此速率上时钟返回或即斜坡降(斜坡—斜率)到原始频率，并当达到原始频率的时候，*FIFO*填充误差也将达到零。

当填充误差减小的检验完成之后，程序进入填充误差增加的检验(框107)。增加的方向决定于旧—填充—误差与视行填充误差

的比较。这个工作与减小的检验是一样的，只是旧的与新的倒换。对于增加的检验，旧的与新的填充－电平－误差之间的差别必须超过参数 *DEF*－极点－2*I* 。在对一个增加的事件作出响应之前，先要进行进一步的检验。如果斜坡－模式表明时钟斜坡－下降(状态 3 或 5)，或是填充误差处在空 (*null*)误差(停滞带段(*dead band*))区域，则对增加的事件置之不顾。让这些检验通过，将斜坡－模式置于 1，并将斜坡－斜率置于值“填充－误差－*dx*”。

参数“*DEF*－极点－2*D* ”除了为增加－减少的检验提供滞迟作用外，还控制自适应时钟过冲的量值。过冲(过校正)的受控量用来在源时钟速率满足之后使 *FIFO* 15 填充电平正常化 。数值的确定依靠画出响应曲线，并调整到约 75% 的时钟过冲(源时钟频率与自适应时钟频率之间初始差别的 75%过校正)。由过冲量确定使 *FIFO* 15 正常化的时间。

框 106 和框 107 的方法能可靠地判定，在存在高的数据到达抖动的情况下，*FIFO* 15 电平误差是增加的或是减小的。框 106 的方法还控制使 *FIFO* 填充电平正常化所需的时钟过校正量值。

一旦处于增加的填充误差模式(斜坡－模式 1)，将每一时间段内斜坡－斜率中的校正量叠加到“积分器 ”的当前值上。需要指出，由于所需的时钟校正会是趋向较高的频率或是较低的频率(图 6 的框 114)，所以斜坡－斜率可能包含正的值或是负的值。这个代数和使积分器值斜升 ，将 *VCXO* 33 频率驱动到这样的值，它满足

并随后超过源频率。当超过视在源频率时，*FIFO* 填充电平误差开始减小。它不象大多数连续反馈的闭环系统中那样，当误差减小时斜坡—斜率的速率并不逐渐减小。这种方法使误差收敛时间最小。过冲受斜坡—模式 2 至 5 的运行的控制。

框 114(图 6)的方法实现主要(极点 1)积分作用。

框 108 (图 5)是转移(斜坡—模式)语句，行号代码#208。时钟斜坡模式提供出保持、过校正和斜坡降等几种功能。模式 2(转移语句中的事例 2)保持于负的填充电平误差，它引出模式 3 以使斜坡降。模式 4 保持于正的填充电平误差，相应的斜坡降是模式 5。斜坡—模式 2 和 4 由减小的填充误差检验算法调用 (框 106)。在减小的事件的开始 (增加的误差转到减小的误差)，自适应时钟 *VCXO* 33 满足源时钟速率，然后超过源时钟速率一个受控的量值。初始时，*FIFO* 15 填充电平将脱离开其标称值，这是因为，当时钟速率正收敛时，在增加的误差模式时段内 *FIFO* 15 填充电平获得或丢失数据。斜坡模式 2 和 4 (图 5 中的框 109 和框 111)可使时钟过冲状态保持恒定，直至 *FIFO* 15 填充电平向标称值收敛，满足参数“*DEF—斜坡—db*”中规定的阈值。这一参数设定于一个 *FIFO* 电平校正点，它给出时间供自适应时钟积分器值得以斜坡下降，并恰好当“填充—电平—误差”达到零时下降到原始(源)频率。各参数均设定正确后，所得结果是个非周期性响应 (时钟过冲或下冲很小或者不存在)，而“通常的”*PLL* 其情况一般不是这样的。对于在

源时钟与自适应时钟 *VCXO* 33 之间有大范围初始频率差的情况，一般也能达到上面的结果。

如图 5 中所示，斜坡下降的“斜坡—模式”是斜坡—模式 3（框 110）和斜坡—模式 5（框 112）。当斜坡—模式 3 和 5 完成了斜坡—下降而到达“积分器—浮动”中的原始积分器值时，它们将斜坡—模式设定于 0，并脱离开转移语句。斜坡—模式 0 是开环控制算法（斜坡—模式 1—5）的静止状态。如果有完善的非周期性阻尼，自适应频率将与源频率协配，且 *FIFO* 将在标称电平上。如果余留的空（*null*）误差大于参数“*DEF*—停滞（*dead*）—带段（*band*）”，则由增加的误差检测算法启动一个新序列（框 107）。当时钟不相同且 *FIFO* 填充电平脱离开标称值时，在电路启动期间内需要有一个以上的周期。这一点，可从图 9 中 40 秒上的二次校正周期看出。

框 109 和框 111 的方法是在 *FIFO* 电平调整好之前保持住过校正的自适应时钟状态。框 110 和框 112 的方法是使自适应时钟频率坡斜地下降，当 *FIFO* 填充误差趋近零时达到源频率。

斜坡—模式 0 是静止状态，当“填充—电平—误差”处于一个小的停滞带段范围内时，进入该斜坡—模式 0。参数“*DEF*—停滞—带段”规定了斜坡—模式 1—5 的停滞带段。在 *DS*1 速率的场合下设定于 8，它允许＋/—8 字节的空误 差范围。斜坡—模式 0（框 113）对积分器值加上一个直接（闭环）反馈校正值，它在由参数“*DEF*—事例 0—速率”规定的速率上加 1 或减 1。校正速率非常低

，该环路的自然频率约为1/60Hz。环路在自然环路频率上振荡，在短时期内“填充—电平—误差”的偏离通常不超过+/—2字节，而在60秒内平均为0字节。应当指出，自适应时钟频率的跟踪/漂移误差是与空误差上的*FIFO*填充电平误差有关联的。

框103(图4)的方法中在空误差点上加入一个量十分小的闭环反馈，以在其环路频率的一个周期内使*FIFO*填充—电平空误差减小到零。

自适应时钟程序的最后作用是将变量“积分器”的内容写入*D/A*转换器30，它控制*VCXO* 33的频率(图6中框115)。这一作用将16比特的变量“积分器”标度给参数“*DEF—VCXO*—范围—因数”中规定的控制范围，然后将所标度的值偏称到*VCXO* 33的中心频率点，它在参数“*DS1—VCXO—DAC*—中心”内规定出。这个被标度的和定中心的值写入到*D/A*转换器30的地址中，并且自适应程序返回到呼叫程序或中断电平。

对于有着大的抖动的输入数据流，要想从其中直接导出具有极小抖动的精确时钟时，该自适应时钟技术和算法对那样的系统是很有用的。对于输入信号中伴随着噪声，在控制响应上有大的滞后的系统，该方法也很有用。对此方法可加以比例化和协调化，以在宽广的范围内应用。

本实施例中所叙述的程序对于小规模式处理器中的执行速度来说是最佳的。一个约1*MIP*容量的16比特处理器可以并行地处

理4个自适应时钟电路。它不使用乘法操作，许多算法产生出近似值来代替精确的计算。不过，可以加上一些计算来使协调的建立减至最小，并在宽广的运用状态下改善阻尼效应来达到非周期性响应。该程序表如下：

```
30    /* t1_adc.c source file */
31
32  1 {
33  1
34  1 /* local DEFinitions arranged as variables for tuning */
35  1
36  1 /* Tuning Notes : DS1
37  1 Pole2I = 6 for low jitter, 8 for moderate cell jitter
38  1 Pole2D = 4 for low jitter, 6 for moderate jitter
39  1 dead_band = 2 for low to moderate jitter, 8 for high jitter >> must be pwr of 2
40  1 case0_db = 0-2 for low to moderate jitter, try 2-4 for high jitter
41  1 case0_rate = none or 0177, use 037 for more null control
42  1
43  1 The remaining variables are for large transient or offset response
44  1 and should not require adjustment.
45  1 */
46  1
47  1 /* The VCXO must be set to +/- 130 ppm for DS1 -- or --  use following factor */
48  1 /* VCXO range factor = (16 * (
                              actual VCXO range divided by 130 for DS1 and /40 for DS3).
49  1
50  1 #define  DS1_VCXO_DAC_center  1900
51  1 #define  DS1_integrator_center  (DS1_VCXO_DAC_center * 16)
52  1
53  1 static short DEF_VCXO_range_factor = 22;        /* DS1 = (16 * (180/130)) = 22 */
54  1 static short DEF_VCXO_DAC_center =   DS1_VCXO_DAC_center;
55  1 static short DEF_integrator_center = DS1_integrator_center;
56  1 static short DEF_normal_fill_level = LAM_FIFO_NORMAL_LEVEL_DS1;
57  1 static short DEF_pole2D    = 5;                 /* pole 2 integration (dec err diff
58  1 static short DEF_pole2I    = 7;                /* pole 2 integration (inc err diff
59  1 static short DEF_min_dec_rate = 2;             /* low-end dec rate boost *
60  1 static short DEF_div_x     = 8;                /* divide by x  for fill_level_err
61  1 static short DEF_dead_band = 8;                /* large response dead-band */
```

```
 62  1 static short DEF_ramp_db  = 10;                  /* ramp-down threshold */
 63  1 static short DEF_float_inerita_mask = 037;       /* 037 (octal) = 162 ppm/sec  */
 64  1 static short DEF_integ_float_rate   = 03;        /* 03 = 250/sec -- DS1 value */
 65  1 static short DEF_integ_adj_ratio    = 4096;
 66  1 static short DEF_fill_adj_ratio     = 8;
 67  1 static short DEF_adj_lim            = 3;
 68  1 static short DEF_case0_rate         = 07;
 69  1 static short DEF_case0_db           = 1;         /* null dead-band */
 70  1
 71  1 /* local static variables */
 72  1 static long old_fill_err;
 73  1 static long damp_err;
 74  1 static long fill_level_err;
 75  1 static long fill_err_unf;
 76  1 static long integrator = DS1_integrator_center;
 77  1 static long integrator_float = DS1_integrator_center;
 78  1 static long scaled_integrator;
 79  1 static long clk_tic;
 80  1 static long ramp_mode;
 81  1 static long ramp_slope;
 82  1 static long fill_err_dx;
 83  1 static long integ_float_adj;
 84  1 static long fill_adj;
 85  1 static long FIFO;
 86  1 static long FIFO_float;
 87  1 static u_short dac_loc;
 88  1 static long dead_band;
 89  1
 90  1
 91  1 clk_tic = clk_tic + 1;
 92  1
 93  1 FIFO = *ptr_FIFO_level;          /* Read FIFO level register - qty in bytes */
 94  1
 95  1 /********  Adaptive clk initialization ************/
 96  1 if (adclk_ckt_init == TRUE)
 97  2         {
 98  2         adclk_ckt_init = FALSE;
 99  2
100  2         integrator = DEF_integrator_center;
101  2         integrator_float = integrator;
102  2         damp_err = 0;
103  2         ramp_mode = 0;
104  2         clk_tic = 0;
105  2         old_fill_err = 0;
106  2         fill_level_err = 0;
107  2         FIFO_float = FIFO;
108  1         }
109  1
110  1 /************** Start of adaptive clock algorithm ******************/
111  1
112  1 /*  the rate of integrator_float change must NOT exceed the integrator ramp rate */
113  1 if ( (clk_tic & DEF_integ_float_rate) == 0)      /* rate of change of integrator_flo

114  2 {
115  2 integ_float_adj = (integrator - integrator_float) / DEF_integ_adj_ratio;
116  2 fill_adj = fill_level_err / DEF_fill_adj_ratio;
117  2
118  2 if (integ_float_adj > DEF_adj_lim)
119  2         integ_float_adj = DEF_adj_lim;
120  2
121  2 if (integ_float_adj < -DEF_adj_lim)
122  2         integ_float_adj = -DEF_adj_lim;
```

```
123  2
124  2 if (fill_adj > DEF_adj_lim)
125  2         fill_adj = DEF_adj_lim;
126  2
127  2 if (fill_adj < -DEF_adj_lim)
128  2         fill_adj = -DEF_adj_lim;
129  2
130  2 integrator_float = integrator_float + integ_float_adj + fill_adj;
131  1 }
132  1
133  1 if ((clk_tic & DEF_float_inerita_mask) == 0)
134  2 {
135  2 if (FIFO > FIFO_float)
136  2         FIFO_float = FIFO_float + 1;
137  2
138  2 if (FIFO < FIFO_float)
139  2         FIFO_float = FIFO_float - 1;
140  1 }
141  1
142  1
143  1 fill_level_err = FIFO_float - DEF_normal_fill_level;
144  1 fill_err_unf = FIFO - DEF_normal_fill_level;
145  1
146  1 *ptr_DAC_3 = (u_short)(fill_level_err + 2048);          /* output for test access *
147  1
148  1 fill_err_dx = fill_level_err / DEF_div_x;
149  1 dead_band = fill_level_err / DEF_dead_band;
150  1
151  1
152  1 /* decreasing error */
153  1 if ((fill_level_err < 0) && (fill_level_err - old_fill_err >= DEF_pole2D))
154  2 {
155  2
156  2         old_fill_err = fill_level_err;
157  2
158  2 /* Increasing turns to decreasing */
159  2         if (ramp_mode == 1)
160  3         {
161  3         ramp_mode = 2;
162  3         ramp_slope = fill_err_dx - DEF_min_dec_rate;     /* low-end dec rate boost *
163  2         }
164  2
165  1 }
166  1
167  1
168  1 if ((fill_level_err > 0) && (old_fill_err - fill_level_err >= DEF_pole2D))
169  2 {
170  2
171  2         old_fill_err = fill_level_err;
172  2
173  2 /* Increasing turns to decreasing */
174  2         if (ramp_mode == 1)
175  3         {
176  3         ramp_mode = 4;
177  3         ramp_slope = fill_err_dx + DEF_min_dec_rate;     /* low-end dec rate boost *
178  2         }
179  2
180  1 }
181  1
182  1 /* increasing error */
183  1 if ((fill_level_err < 0) &&  (old_fill_err -  fill_level_err >= DEF_pole2I))
184  2 {
```

```
185  2
186  2          old_fill_err = fill_level_err;
187  2
188  2          if ((ramp_mode != 3) && (ramp_mode != 5) && (dead_band != 0))
189  3          {
190  3          damp_err = fill_err_dx;
191  3          ramp_mode = 1;
192  2          }
193  1 }
194  1
195  1
196  1 if ((fill_level_err > 0) &&  (fill_level_err - old_fill_err >= DEF_pole2I))
197  2 {
198  2
199  2          old_fill_err = fill_level_err;
200  2
201  2          if ((ramp_mode != 3) && (ramp_mode != 5) && (dead_band != 0))
202  3          {
203  3          damp_err = fill_err_dx;
204  3          ramp_mode = 1;
205  2          }
206  1 }
207  1
208  1 switch (ramp_mode)
209  2 {
210  2 case 0:
211  2 case 1:
212  2
213  2 if ( (clk_tic & DEF_case0_rate) == 0)
214  3 {
215  3
216  3 if (fill_err_unf > DEF_case0_db)
217  3 integrator = integrator + 1;
218  3
219  3 if (fill_err_unf < -DEF_case0_db)
220  3 integrator = integrator - 1;
221  2 }
222  2
223  2 break;
224  2
225  2 case 2:
226  2          damp_err = 0;
227  2
228  2          if (fill_level_err > -DEF_ramp_db)
229  3          {
230  3                    ramp_mode = 3;
231  2          }
232  2 break;
233  2
234  2 case 3:
235  2          damp_err = -ramp_slope;
236  2
237  2          if ((integrator > integrator_float) || (fill_level_err > DEF_ramp_db))
238  3          {
239  3                    ramp_mode = 0;
240  3                    damp_err = 0;
241  3                    old_fill_err = fill_level_err;
242  2          }
243  2 break;
244  2
245  2 case 4:
246  2          damp_err = 0;
```

```
247  2
246  2          if (fill_level_err < DEF_ramp_db)
249  3          {
250  3                  ramp_mode = 5;
251  2          }
252  2 break;
253  2
254  2 case 5:
255  2          damp_err = -ramp_slope;
256  2
257  2          if ((integrator < integrator_float) || (fill_level_err < -DEF_ramp_db))
258  3          {
259  3                  ramp_mode = 0;
260  3                  damp_err = 0;
261  3                  old_fill_err = fill_level_err;
262  2          }
263  2 break;
264  1 }
265  1
266  1 integrator = integrator + damp_err;
267  1
268  1 if (integrator < 0)
269  1          integrator = 0;
270  1
271  1 else if (integrator > 0xFFF0)          /* avoid rollover to 0 */
272  1          integrator =  0xFFF0;          /* masking will not work here */ -
273  1
274  1 /* normalize and scale >> the 16 bit integrator to 12 bits << and */
275  1 /* scale the >> DAC value << (
                                   both scales are included in the VCXO_ranger_factor valu
276  1 scaled_integrator = (integrator - DEF_integrator_center) / DEF_VCXO_range_factor;
277  1
278  1 dac_loc = (u_short)(scaled_integrator +  DEF_VCXO_DAC_center);
                                                          /* add it DAC center or nul.
279  1
280  1 *ptr_DAC =  *ptr_DAC_global = dac_loc;                  /* write to DAC for ckt. x
281  1
282    }
283    /************** End of adaptive clock algorithm *****************/
```

图 7 示出对瞬态误差的响应曲线 *FIFO* 15 丢失数据 1 个单元有效负载或 47 字节)，表明自适应时钟频率(“积分器”)相对于源时钟的时间曲线如 *FIFO* 填充电平误差 (“填充一电平一误差”)的时间曲线 。曲线表明电路配置 10(图 1)对一个瞬态误差的响应，该瞬态误差是 *FIFO* 丢失一个单元有效负载 (47 字节或 47 个 8 比特组)。填充电平误 差曲线表明暂时的单元丢失及自适应时钟对此的响应一比源时钟低下 60*ppm* 。需要指出，这里有半个周期的过冲时间或阻尼稳定时间。电路配置 10 已作出最佳设计，使它对于跟踪源时钟变化比之对于瞬态数据误差提供出更好的阻尼特性。

图 8 示出初始 *FIFO* 误差(1 单元有效负载或 47 字节)对初始源时钟无误差的响应曲线，表明自适应时钟频率(“积分器”)相对于源时钟的时间曲线、*FIFO* 填充电平误差(“填充一电平一误差”)的时间曲线和视在源频率 (“积分器一浮动”)的时间曲线。图 8 的情况与图 7 的类同 ，与图 7 的差别仅在于单元丢失在这里发生于起始的填充电平误差 。此外，这里也画出了变量“积分器一浮动”的曲线。起始期或即时钟捕捉期得以大为减短，其方法是在自适应时钟算法启动之前，在起始时将 *FIFO* 电平初始化于“*DEF*一正常一填充一电平”值。这个 工作也由微处理器 29 按下面的程序来完成 。当电路不起作用时，读出选通 24 阻塞，线路发送器 21 传送出全 1 的置闲码(*DS*1 的 *AIS* 信号)。当单元流的到达被检测到并稳

定后，*FIFO* 复位 26，在一个紧密环路中立即监测填充一电平值(查询)。当填充电平等于“*DEF*一正常一填充一电平”减去超前偏置时，读出选通 24 启动，*AIS* 信号被阻塞。超前偏置使微处理器 29 对读出选通 24 的启动提供出时间。可以调整到使该启动与填充电平达到“*DEF*一正常一填充一电平”相一致。

图 9 示出初始 *FIFO* 误差(1 单元有效负载或 47 字节)对初始源时钟误差的响应曲线，表明自适应时钟频率(“积分器”)相对于源时钟的时间曲线、*FIFO* 填充电平误差(“填充一电平一误差”)的时间曲线和视在源频率 (“积分器一浮动”)的时间曲线。曲线表明了电路结构 10 对初始 *FIFO* 15 1 单元有效负载 (47 字节)误差的响应以及对初始源时钟一70*ppm* 误差的响应 。图 9 中还示出在 40 秒上的一个二次校正周 期 。初始校正周期使时钟坡斜地下降到比实际源频率低几个 *ppm* 。这使得“填充一电平一误差”慢慢地向上滑移，并在 40 秒上有一个校正周期 。斜坡一模式 0 也可以在空误差区内进行慢漂移校正，这可以从大约 35 秒处自适应时钟积分器收敛于近乎 0 来看出 。在这种情况下，斜坡一模式 0 的校正显得太小和太迟。

在“通常的”*PLL* 中，目的时钟速率由平均的单元流到达速率单独确定。相位/频率信息来自 *FIFO* 填充电平，如图 1 中所示，这是 1988 年 10 月 10 日理查德 (*Richard c. Lau*)对 *T*1S1. 1 提出的 *Bellcore* 论文“供 *ATM* 电路模拟用的一种时钟恢复方案”中提出

的。由于这一技术不需要源电路中来的网络基准时钟和编码的时钟信息，所以一般公认是很有希望的。然而，正常也公认，当尝试进行实际设计时，就时钟收敛时间、抖动和漂移而言其性能并不良好或是不能接受的。

除了上面所说的性能问题之外，电路分析和实验表明，在这种应用场合下，当相位/环路增益足以控制时钟漂移时，“通常的”*PLL* 在阻尼和稳定性的控制上存在着固有的问题。

10 示出“常规的”*PLL* 电路的一个具体例子图电路 210。电路 210 用作一个接口，从线路 211 上 155*Mb/s* 的导步传递模式(*ATM*)单元流接口到线路 222 上同步的 *DS*1 的 1.544*Mb/s* 恒定比特率电路。由于线路 211 上入局单元流的突发和导步性质，线路 222 上同步地传送信息所需的时钟并不能利用线路 211 中的边沿信息或过渡信息来导出，而要依靠线路 211 上长时期的平均单元流速率来得到。线路 211 上的每个 *ATM* 单元为 53 字节的数据包，包括 5 字节的首标、1 字节的适配层和 47 字节的可用信息有效负载。电路 212 实行 *CCITT* 适配层 1 处理，包括去除单元首标和适配层，并应用线路 214 上的写入选通经由字节母线 213 来控制 47 字节有效负载的写入一个同步的先进先出(*FIFO*)存储器 215。*FIFO* 215 可存储到 4096 个 8 比特字节，它足以充分地存储在线路 222 上积累着等候传输的全部字节数目。*FIFO* 215 还在线路 227 上传送模拟的 *FIFO* 填充电平信号，它在空的 *FIFO* 时为 0*V*，

满的 *FIFO* 时变化到 5*V*。

线路 235 上的自适应线路时钟用来控制从 *FIFO* 215 向 *DS*1 也即线路 222 上同步的电路传输字节。这个线路 235 上的自适应线路时钟频率用来运行并/串变换器 218 和线路发送器 221 。该线路时钟频率由一个计 数器 223 除以 8 ，在线路 224 上得到的字节时钟用作读出选通，以便经由字节母线 216 、并/串变换器 218 和线路 219 从 *FIFO* 15 中实现有效负载信息的字节读出，用于由线路 221 在线路 222 上作为一个 *DS*1 电路进行传输。线路发送器 221 将一个 *B*8*ZS* 码插入线路流中，以防止有 7 个以上接连的 0 比特在线路 222 上传送出。

线路 227 上的模拟电压通过一个由电阻 232 和电容 234 组成的 *PLL* 环路 滤 波 器 传 输 。滤波得到的电压对一个压控晶体振荡器(*VCXO*233)提供控制用输入信号，线路 235 上的 *VCXO* 233 自适应线路时钟频率是在闭环状态下调整的。

CCITT 研究组 *XVIII* 已规定了标准化的 *SRTS* (先有技术的同步残余时间标记)结构，以在 *ATM* 网络中重建时钟速率。在这种结构中，将时钟速率信息以时间标记(*time*—*stamp*)形式编码在源电路中。该时间标记基本上是源电路时钟与网络基准时钟之间的差值。这个编码的 时 间 标记以 *ATM* 单元首标的倒置比特形式传送到目的时钟电路。在目的时钟电路上，时间标记和网络基准时钟根本上用来重建原来 的源频率 。这方面的图解示于 1992 年 3 月 9

日 *CCITT* 的注释图 5 中。

本方法提供出性能优良的目的时钟电路，它满足全部传输要求。然而，只当有网络基准可供应用时，才可能应用。其它的缺点是需要从源时钟中得到编码的信息，以及在目的时钟电路上缺乏对调节或维持 *FIFO* 填充电平的控制。(*FIFO* 填充电平影响电路中总的传输延时。)

说　明　书　附　图

图1

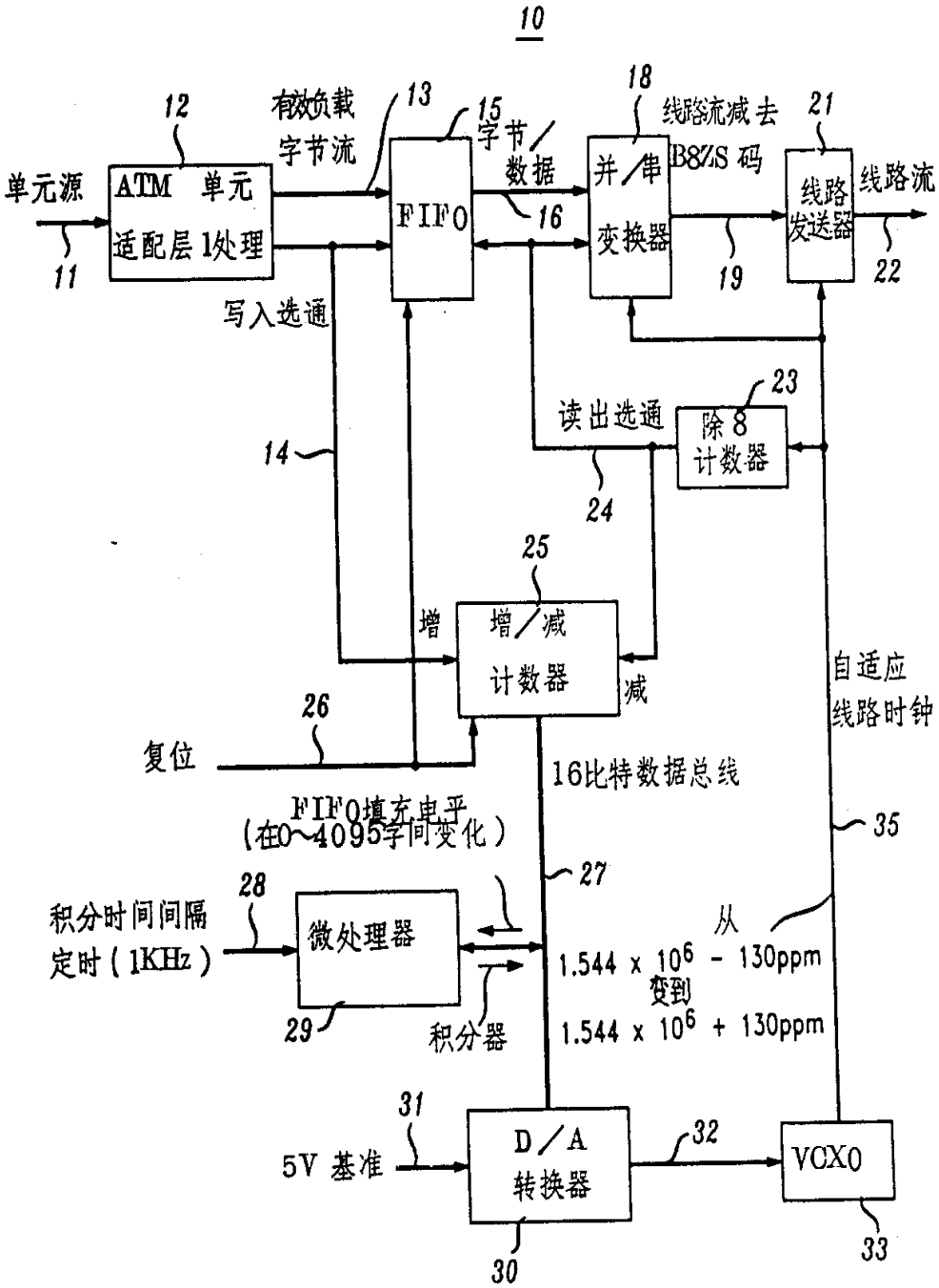

10
12
有效负载
字节流
13
15
字节/
数据
18
线路流减去
B8ZS 码
21
单元源
ATM 单元
适配层1处理
FIFO
并/串
变换器
线路
发送器
线路流
11
16
19
22
写入选通
读出选通
23
除8
计数器
14
24
25
增
增/减
计数器
减
自适应
线路时钟
26
复位
16比特数据总线
FIFO填充电平
(在0~4095字间变化)
35
27
28
积分时间间隔
定时(1KHz)
微处理器
从
1.544 x 10^6 - 130ppm
变到
1.544 x 10^6 + 130ppm
29
积分器
31
5V 基准
D/A
转换器
32
VCXO
30
33

图2

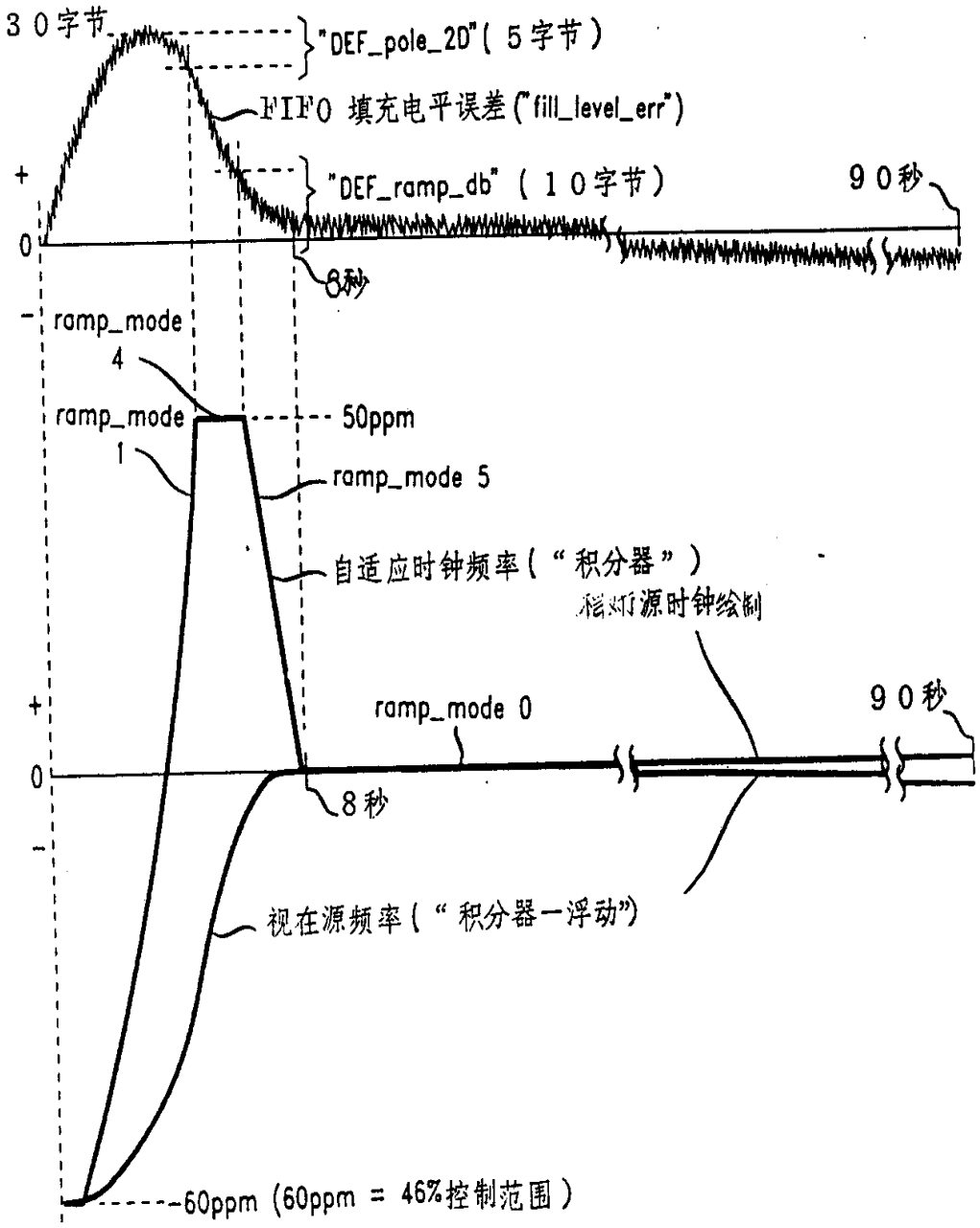

30字节
"DEF_pole_2D"（5字节）
FIFO 填充电平误差("fill_level_err")
+
0
-
"DEF_ramp_db"（10字节）
90秒
8秒
ramp_mode
4
ramp_mode
1
50ppm
ramp_mode 5
自适应时钟频率（"积分器"）
标称源时钟绘制
+
0
-
ramp_mode 0
90秒
8秒
视在源频率（"积分器一浮动"）
-60ppm (60ppm = 46%控制范围)

图3

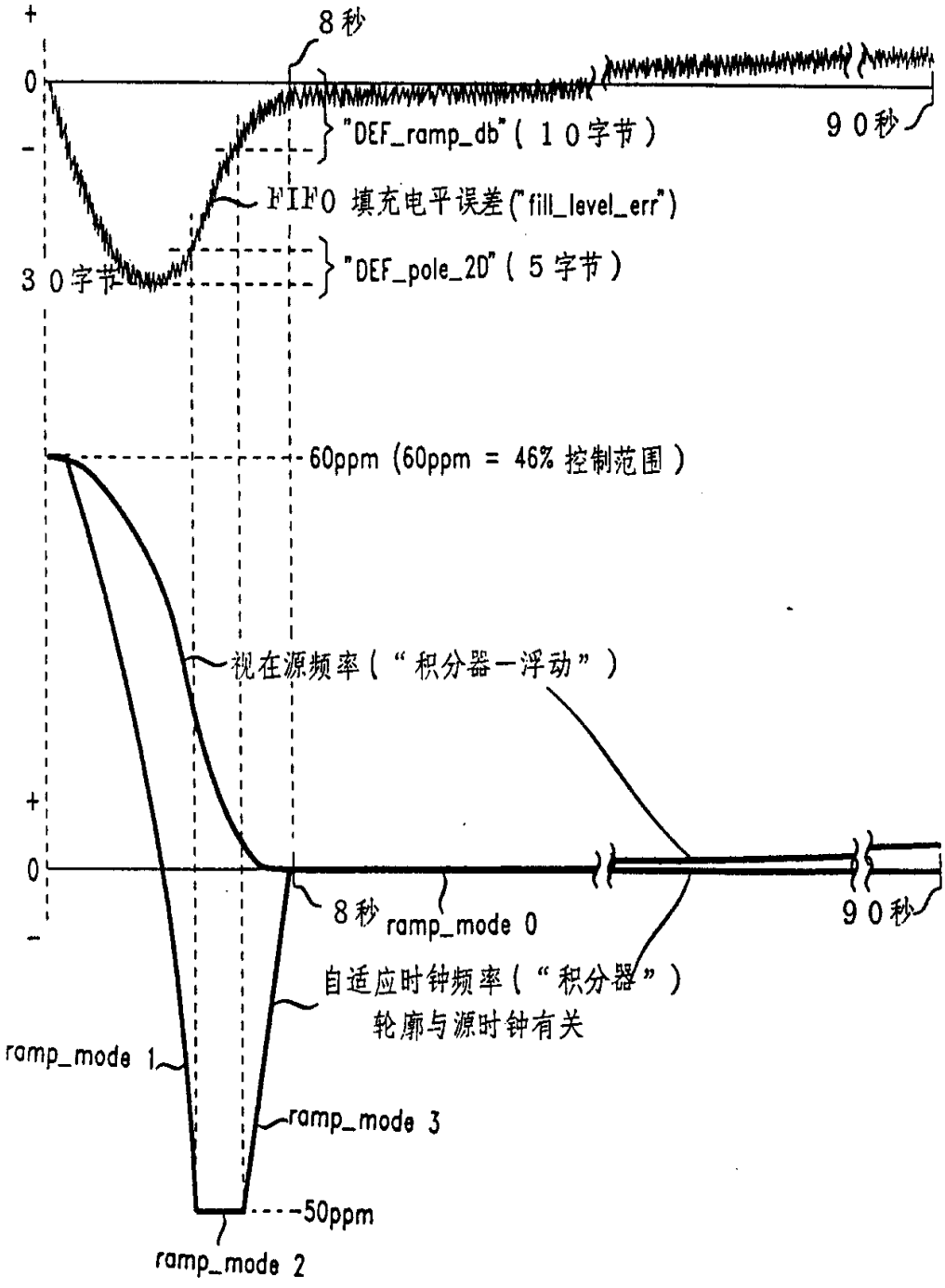
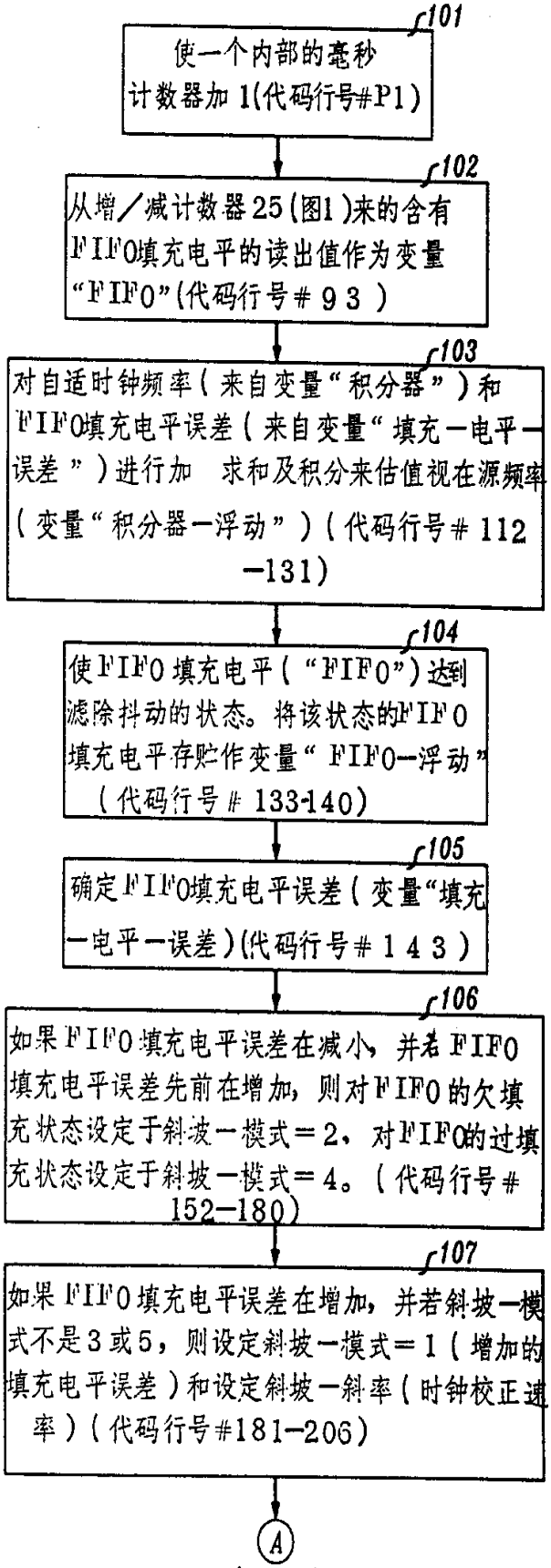
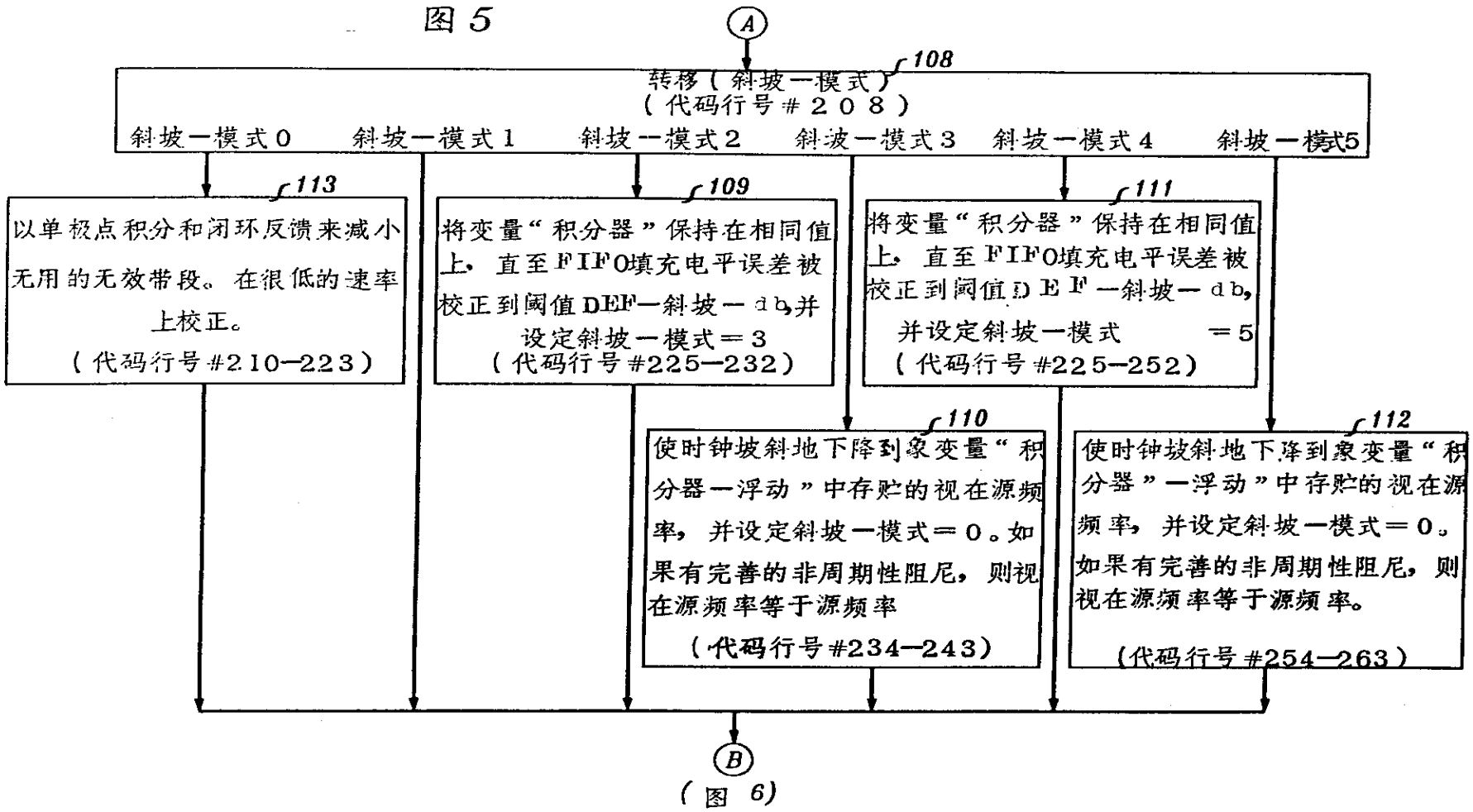

+
0
-
8秒
90秒
"DEF_ramp_db"（10字节）
FIFO 填充电平误差("fill_level_err")
30字节
"DEF_pole_2D"（5字节）
60ppm (60ppm = 46% 控制范围)
视在源频率（“积分器—浮动”）
+
0
-
8秒
ramp_mode 0
90秒
自适应时钟频率（“积分器”）
轮廓与源时钟有关
ramp_mode 1
ramp_mode 3
-50ppm
ramp_mode 2

图4


101
使一个内部的毫秒计数器加1(代码行号#P1)
102
从增／减计数器25(图1)来的含有FIFO填充电平的读出值作为变量"FIFO"(代码行号#93)
103
对自适时钟频率(来自变量"积分器")和FIFO填充电平误差(来自变量"填充一电平一误差")进行加 求和及积分来估值视在源频率(变量"积分器一浮动")(代码行号#112-131)
104
使FIFO填充电平("FIFO")达到滤除抖动的状态。将该状态的FIFO填充电平存贮作变量"FIFO一浮动"(代码行号#133-140)
105
确定FIFO填充电平误差(变量"填充一电平一误差")(代码行号#143)
106
如果FIFO填充电平误差在减小，并若FIFO填充电平误差先前在增加，则对FIFO的欠填充状态设定于斜波一模式=2，对FIFO的过填充状态设定于斜坡一模式=4。(代码行号#152-180)
107
如果FIFO填充电平误差在增加，并若斜坡一模式不是3或5，则设定斜坡一模式=1(增加的填充电平误差)和设定斜坡一斜率(时钟校正速率)(代码行号#181-206)
A

(图.5)

图 5


A
转移（斜坡一模式）
（代码行号＃２０８）
108
斜坡一模式０
斜坡一模式１
斜坡一模式２
斜坡一模式３
斜坡一模式４
斜坡一模式5
113
以单极点积分和闭环反馈来减小无用的无效带段。在很低的速率上校正。
（代码行号#210—223）
109
将变量“积分器”保持在相同值上，直至FIFO填充电平误差被校正到阈值DEF—斜坡—db，并设定斜坡一模式＝3
（代码行号#225—232）
111
将变量“积分器”保持在相同值上，直至FIFO填充电平误差被校正到阈值DEF—斜坡—db，并设定斜坡一模式＝5
（代码行号#225—252）
110
使时钟坡斜地下降到象变量“积分器一浮动”中存贮的视在源频率，并设定斜坡一模式＝０。如果有完善的非周期性阻尼，则视在源频率等于源频率
（代码行号#234—243）
112
使时钟坡斜地下降到象变量“积分器”一浮动”中存贮的视在源频率，并设定斜坡一模式＝０。如果有完善的非周期性阻尼，则视在源频率等于源频率。
（代码行号#254—263）
B

（图 6）

图6

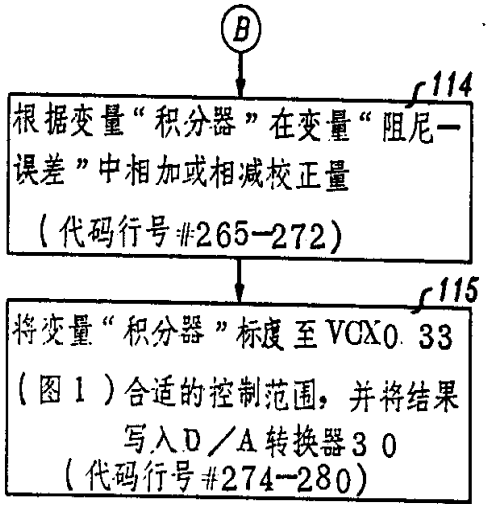

B
114
根据变量“积分器”在变量“阻尼—误差”中相加或相减校正量
（代码行号#265—272）
115
将变量“积分器”标度至VCXO. 33（图1）合适的控制范围，并将结果写入D／A转换器30
（代码行号#274—280）

图 7

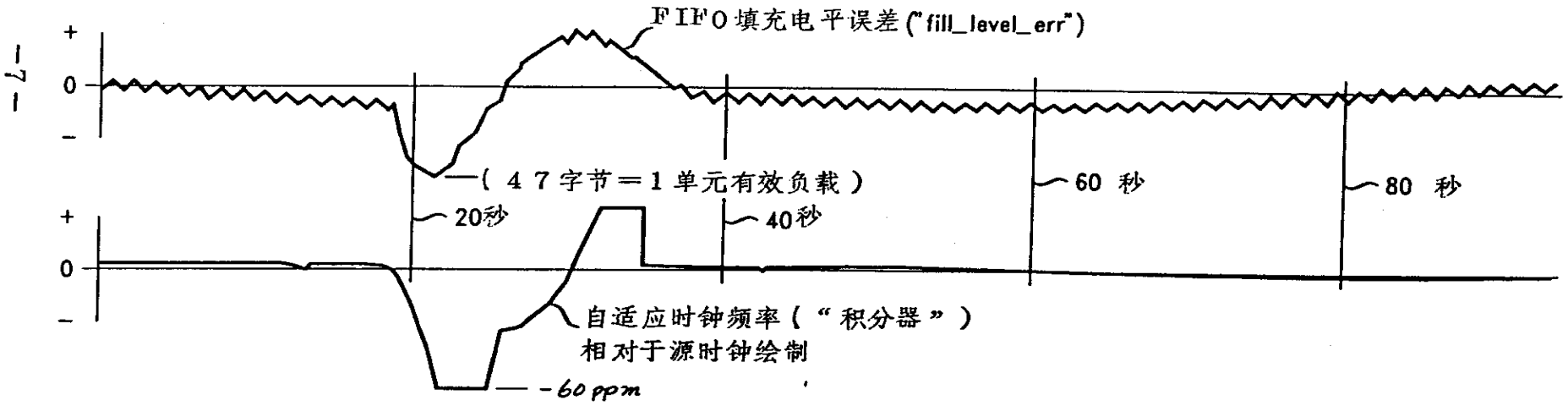

FIFO填充电平误差("fill_level_err")
+
0
−
(47字节=1单元有效负载)
20秒
40秒
60 秒
80 秒
+
0
−
自适应时钟频率(“积分器”)
相对于源时钟绘制
-60 ppm

图 8

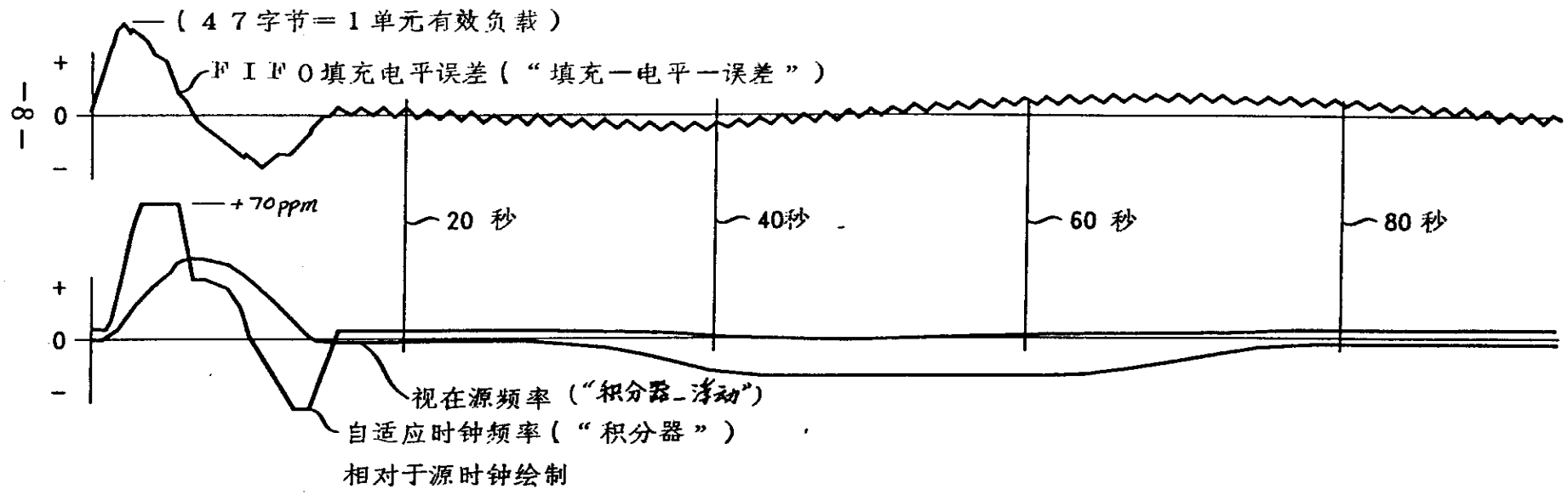

—（47字节＝1单元有效负载）
+
0
–
FIFO填充电平误差（“填充—电平—误差”）
—+70ppm
20 秒
40秒
60 秒
80 秒
+
0
–
视在源频率（“积分器_浮动”）
自适应时钟频率（“积分器”）
相对于源时钟绘制

图 9

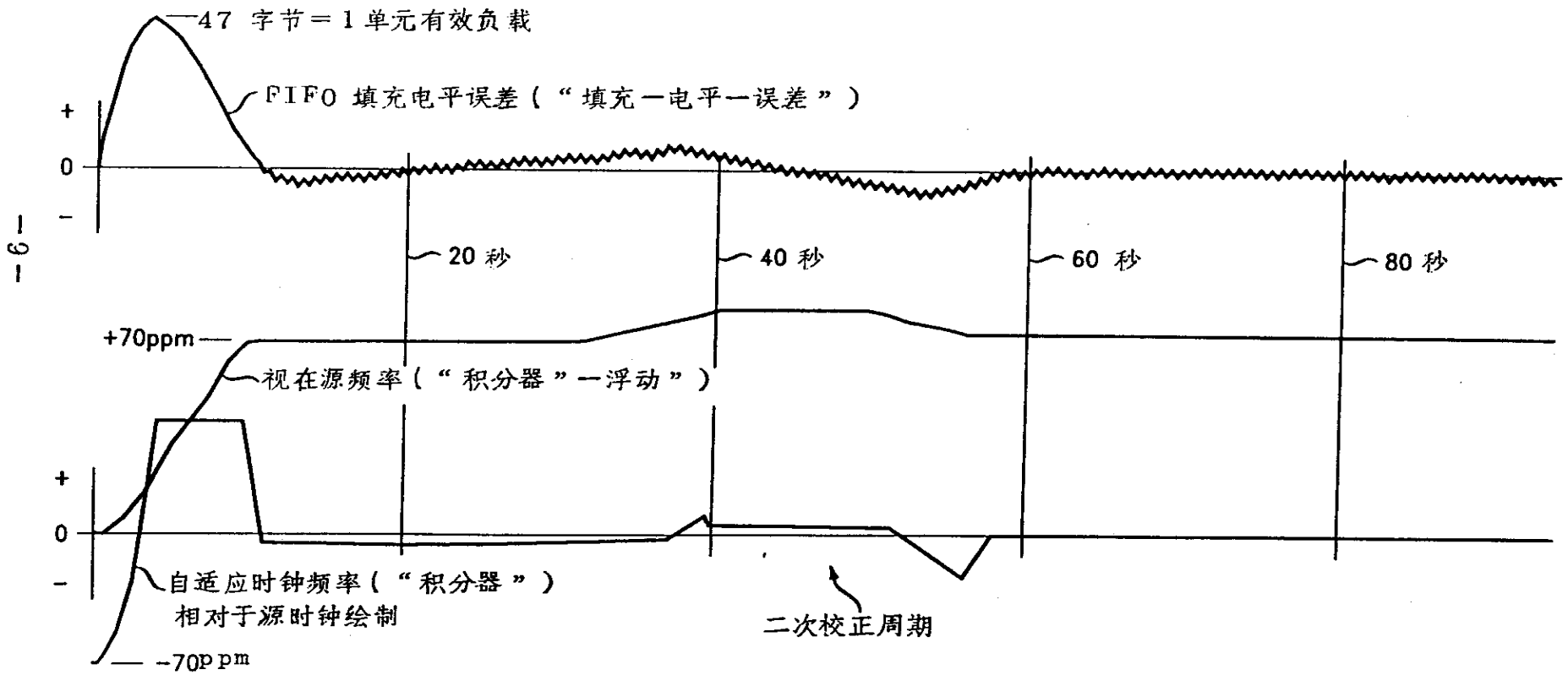
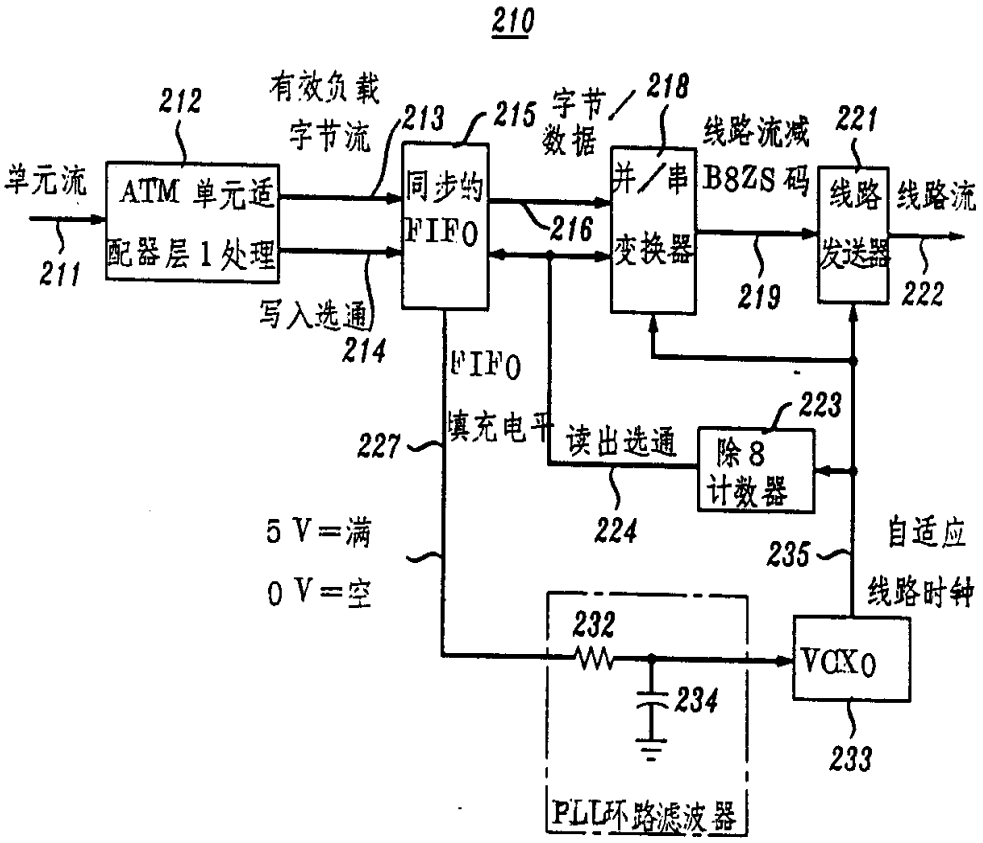

47 字节＝1 单元有效负载
FIFO 填充电平误差（“填充—电平—误差”）
+
0
–
20 秒
40 秒
60 秒
80 秒
+70ppm
视在源频率（“积分器”—浮动”）
+
0
–
自适应时钟频率（“积分器”）
相对于源时钟绘制
-70ppm
二次校正周期

图 10


210
212
有效负载
字节流
213
215
字节
数据
218
线路流减
221
单元流
ATM 单元适
配器层 1 处理
同步的
FIFO
并/串
变换器
B8ZS 码
线路
发送器
线路流
211
216
219
222
写入选通
214
FIFO
填充电平
227
读出选通
223
除 8
计数器
224
5 V＝满
0 V＝空
235
自适应
线路时钟
232
VCXO
234
233
PLL环路滤波器