(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114787775 A
(43)申请公布日 2022.07.22

(21)申请号 202180007057.2
(22)申请日 2021.01.27
(30)优先权数据
2020-016144 2020.02.03 JP
(85)PCT国际申请进入国家阶段日
2022.06.09
(86)PCT国际申请的申请数据
PCT/JP2021/002880 2021.01.27
(87)PCT国际申请的公布数据
WO2021/157448 JA 2021.08.12
(71)申请人 索尼互动娱乐股份有限公司
地址 日本东京都
(72)发明人 大塚活志
(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 金兰
(51)Int.Cl.
G06F 9/48(2006.01)
G06F 15/177(2006.01)
G06F 15/80(2006.01)

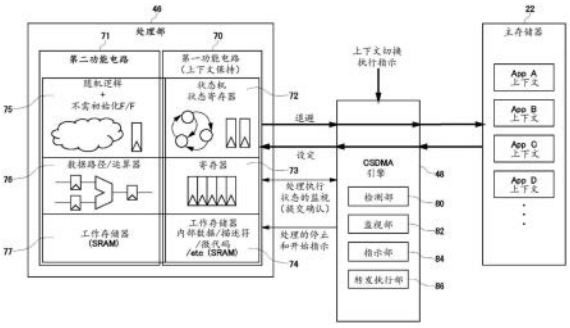
权利要求书2页 说明书15页 附图15页

(54)发明名称

数据处理系统、数据转发装置以及上下文切换方法

(57)摘要

处理部(46)以时分方式执行与多个应用有关的处理。CSDMA引擎(48)检测处理部(46)中的执行对象的应用的切换定时。在检测到切换定时时，CSDMA引擎(48)不经由管理上述多个应用的软件的处理，而使由处理部(46)所执行的应用的上下文从处理部(46)退避到主存储器(22)，将由处理部(46)接下来要执行的应用的上下文从所述存储器设定到处理部(46)。



CN 114787775 A

1.一种数据处理系统,其特征在于,具备:

处理部,以时分方式执行与多个应用有关的处理;以及

数据转发部,在所述处理部和存储器之间转发数据,

所述数据转发部包含:

检测部,检测所述处理部中的执行对象的应用的切换定时;以及

转发执行部,在检测到所述切换定时的情况下,执行如下的转发处理:不经由管理所述多个应用的软件的处理,而使由所述处理部所执行的应用的上下文从所述处理部退避到所述存储器,并将由所述处理部接下来要执行的应用的上下文从所述存储器设定到所述处理部。

2.根据权利要求1所述的数据处理系统,其特征在于,

被退避在所述存储器中的应用的上下文包含在所述处理部中未完成的处理的当前状态,

在检测到所述切换定时的情况下,即使所述处理部中的与应用有关的处理未完成,所述转发执行部也执行所述转发处理。

3.根据权利要求1所述的数据处理系统,其特征在于,

在检测到所述切换定时的情况下,所述转发执行部在所述处理部中的与应用有关的处理完成后执行所述转发处理。

4.根据权利要求3所述的数据处理系统,其特征在于,

所述处理部中的与应用有关的处理包含该应用中的小粒度的多个内部处理,

所述转发执行部在所述处理部中的执行对象的应用的内部处理完成后执行所述转发处理。

5.根据权利要求1至4中任一项所述的数据处理系统,其特征在于,

所述处理部从与所述多个应用对应的多个队列、即被存储指示与各应用有关的处理内容的命令的多个队列中,读出所述命令,

所述处理部在切换执行对象的应用时,切换读出命令的队列。

6.根据权利要求1至5中任一项所述的数据处理系统,其特征在于,

所述检测部将垂直回扫期间或水平回扫期间的开始检测为所述切换定时。

7.根据权利要求1至6中任一项所述的数据处理系统,其特征在于,

所述转发执行部使用DFT用的扫描链即可测性设计用的扫描链和存储器BIST用的电路即存储器内建自测用的电路中的至少一方来执行所述转发处理。

8.根据权利要求1所述的数据处理系统,其特征在于,

所述处理部包含多个保持应用的上下文的功能部,

在所述处理部使用被保持在第一功能部的应用的上下文来执行与该应用有关的处理的期间,所述转发执行部将被保持在所述处理部的第二功能部的其他应用的上下文作为对象来执行所述转发处理。

9.根据权利要求1所述的数据处理系统,其特征在于,

所述处理部包含多个保持应用的上下文的功能部,

所述处理部使用被保持在第一功能部中的第一应用的上下文来执行与所述第一应用有关的处理,使用被保持在第二功能部中的第二应用的上下文来执行与所述第二应用有关

的处理，

在所述处理部中的执行对象从所述第一应用切换到所述第二应用的情况下，所述处理部在与所述第一应用有关的处理中的不能与所述第二应用所相关的处理混流的处理完成后，并行执行与所述第一应用有关的处理中的能够与所述第二应用所相关的处理混流的处理、和与所述第二应用有关的处理。

10.根据权利要求1至9中任一项所述的数据处理系统，其特征在于，

由所述转发执行部进行的转发处理的对象包含被保持在所述处理部的状态机中的上下文。

11.根据权利要求1至10中任一项所述的数据处理系统，其特征在于，

由所述转发执行部进行的转发处理的对象包含被保持在所述处理部中的上下文、即从软件不能访问的上下文。

12.根据权利要求1至11中任一项所述的数据处理系统，其特征在于，

所述上下文是被中断的时间点的应用的处理的再现所需的数据。

13.一种数据转发装置，其特征在于，具备：

检测部，检测以时分方式执行与多个应用有关的处理的处理部中的执行对象的应用的切换定时；以及

转发执行部，在检测到所述切换定时的情况下，执行如下的转发处理：不经由管理所述多个应用的软件的处理，而使由所述处理部所执行的应用的上下文从所述处理部退避到存储器，并将由所述处理部接下来要执行的应用的上下文从所述存储器设定到所述处理部。

14.一种上下文切换方法，其特征在于，计算机执行：

检测以时分方式执行与多个应用有关的处理的处理部中的执行对象的应用的切换定时的步骤；以及

在检测到所述切换定时的情况下，执行如下的转发处理的步骤：不经由管理所述多个应用的软件的处理，而使由所述处理部所执行的应用的上下文从所述处理部退避到存储器，并将由所述处理部接下来要执行的应用的上下文从所述存储器设定到所述处理部。

数据处理系统、数据转发装置以及上下文切换方法

技术领域

[0001] 本公开涉及数据处理技术，特别涉及数据处理系统、数据转发装置和上下文切换(context switch)方法。

背景技术

[0002] 在以时分方式执行与多个应用有关的处理的处理部中，在切换执行对象的应用时需要进行上下文切换。

发明内容

[0003] 发明要解决的课题

[0004] 近年来，处理部(例如GPU等)有时具备数千个以上的寄存器，另外，由处理部执行的应用的上下文有时成为数十MB以上的大小。在该情况下，处理部的上下文切换有可能需要长时间。

[0005] 本公开是鉴于这样的状况而完成的，一个目的在于抑制伴随上下文切换的处理的延迟。

[0006] 用于解决课题的手段

[0007] 为了解决上述课题，本发明的某一方式的数据处理系统包括：处理部，以时分方式执行与多个应用有关的处理；以及数据转发部，在处理部和存储器之间转发数据。数据转发部包括：检测部，检测处理部中的执行对象的应用的切换定时；以及转发执行部，在检测到切换定时的情况下，执行如下的转发处理：不经由管理多个应用的软件的处理，而使由处理部所执行的应用的上下文从处理部退避到存储器，将由处理部接下来要执行的应用的上下文从存储器设定到处理部。

[0008] 本发明的另一方式是数据转发装置。该装置包括：检测部，检测以时分方式执行与多个应用有关的处理的处理部中的执行对象的应用的切换定时；以及转发执行部，在检测到切换定时的情况下，执行如下的转发处理：不经由管理多个应用的软件的处理，而使由处理部所执行的应用的上下文从处理部退避到存储器，将由处理部接下来要执行的应用的上下文从存储器设定到处理部。

[0009] 本发明的又一方式是上下文切换方法。该方法由计算机执行：检测在以时分方式执行与多个应用有关的处理的处理部中的执行对象的应用的切换定时的步骤；以及在检测到切换定时的情况下，执行如下的转发处理的步骤：不经由管理多个应用的软件的处理，而使由处理部所执行的应用的上下文从处理部退避到存储器，将由处理部接下来要执行的应用的上下文从存储器设定到处理部。

[0010] 另外，通过在计算机程序、记录有计算机程序的记录介质等之间对以上的结构要素的任意组合、本公开的表现进行变换而得到的方案，作为本公开的方式也是有效的。

[0011] 技术效果

[0012] 根据本公开，能够抑制伴随上下文切换的处理的延迟。

附图说明

[0013] 图1是表示第一实施例的计算机的结构的图。

[0014] 图2是表示处理部和CSDMA引擎的详细结构的框图。

[0015] 图3是按时序表示处理部中的视频绘制处理的图。

[0016] 图4是表示视频定时的例子的图。

[0017] 图5是表示SOC的结构的图。

[0018] 图6是表示SOC的结构的图。

[0019] 图7是表示SOC的结构的图。

[0020] 图8是表示SOC的结构的图。

[0021] 图9A是表示第一实施例中的上下文切换时的内部处理的例子的图,图9B是表示变形例中的上下文切换时的内部处理的例子的图。

[0022] 图10是表示第二实施例的处理部的结构的图。

[0023] 图11是表示第三实施例中的上下文切换时的内部处理的例子的图。

[0024] 图12A、图12B、图12C是表示上下文切换中的处理部和CSDMA引擎的动作的图。

[0025] 图13是表示第四实施例的处理部的结构的图。

[0026] 图14是表示第五实施例的处理部的结构的图。

[0027] 图15A和图15B是表示应用的分配例的图。

具体实施方式

[0028] 在实施例中,提出了以时分方式执行与多个应用有关的处理的处理部,换言之,提出了在由多个应用以时分方式所共享的处理部中高速地执行上下文切换的技术。实施例的技术对于某个系统在挂起(suspend)或休眠(hibernation)等保持状态的状态下暂时停止,之后从保持的状态再次开始执行的动作也是有效的,能够使该动作高速化。

[0029] 处理部也可以是SOC(System-On-a-Chip:片上系统)等中的硬件功能块。另外,处理部可以是(1)CPU(Central Processing Unit:中央处理单元)、DSP(Digital Signal Processor:数字信号处理器)、GPU(Graphics Processing Unit:图形处理单元)、NPU(Network Processing Unit:网络处理单元)等处理器,也可以是(2)执行视频的压缩以及解压缩的视频编解码块。将在视频编解码块中执行视频压缩的块在以下也称为“视频编码器”。另外,处理部可以是(3)执行视频的分辨率变换或画质变换、复用等、并以规定的定时处理像素数据的视频(显示器)管道(pipeline)块(以下也称为“视频管道”。),也可以是(4)向外围设备的输入输出接口块。

[0030] 〈背景〉

[0031] 一个硬件功能块(在实施例中称为“处理部”。)有时以时分方式执行多个应用任务线程(以下称为“应用”。)。处理部为了按每个应用执行专用的处理,保持指定执行对象的应用的数据、或指定设定和动作的数据等。处理部在开始处理时,在建立了这些数据之后开始处理。处理部在执行与应用有关的处理的过程中,随时生成应用专用的数据或设定、动作状态等。将与这些各个应用有关的处理中所需的数据或所生成的数据称为“上下文”。即,上下文是在相同的处理部再次执行相同的处理时,为了被确保得到相同的结果的再现性而需要的数据。

[0032] 在一个处理部以时分方式执行多个应用的情况下,需要该处理部中的切换处理。具体而言,通过使此前由处理部执行的应用的上下文退避到外部,并在处理部中设定接下来要执行的应用的上下文,从而进行基于时分的应用切换。将该切换在以下称为“上下文切换”。在进行上下文的退避和设定的期间,需要停止该处理部的处理。为了提高处理部的利用效率,并且降低处理部中的处理的延迟(换言之,提高实时性),要求使上下文切换高速化,缩短处理部的处理停止时间。

[0033] 一般而言,上下文切换大多由软件访问处理部,通过基于软件的逐次处理,经由寄存器访问而被执行上下文的退避和设定。另一方面,存在近年来的处理部的高功能化不断发展,设定用寄存器数达到数千个以上,并且处理中保持的上下文的数据量达到数十MB以上的情况。因此,若通过基于软件的逐次处理进行上下文切换,则需要数十毫秒级的时间等,有可能大幅损害实时性。

[0034] 在CPU中,有的具有支持上下文切换的高速化的硬件功能。但是,该功能只不过是硬件对可从软件访问的一部分寄存器自动执行上下文的退避和设定,而不是覆盖被保持在内部的全部上下文的退避和设定。因此,通过基于软件的逐次处理进行其他上下文的退避和设定。这可以说是在一般的x86或ARM架构的CPU中执行OS(Windows(注册商标)或Linux(注册商标))和多个应用的情况中等。

[0035] <实施例的概要>

[0036] 实施例的数据处理系统具有执行用于处理部的上下文切换的上下文转发的专用功能块。由此,在利用庞大的上下文的处理部和应用的组合中,也高速地执行上下文切换。换言之,抑制伴随上下文切换的处理的延迟。以下,将该专用功能块称为“上下文切换DMA(Direct Memory Access:直接存储器访问)引擎”,标记为“CSDMA引擎”。

[0037] CSDMA引擎在被通知应用的切换定时或者自己检测到应用的切换定时时,(1)选择被保持在存储器上的各应用的上下文,不经由软件而转发给处理部来进行设定。或者,(2)不经由软件而使被保持在处理部中的上下文转发、退避到存储器中。

[0038] 上下文切换中的转发对象包含寄存器内容、处理部内存储器的内容、状态机的状态状态(状态迁移的内部状态)、在高速数据传输错误纠正(on the fly)中执行中的待处理传输(out standing)的处理的当前状态。这些转发对象不限于可从软件访问的对象,除此之外,还包括处理部内置并保持上下文的对象。即,实施例中的上下文切换中的转发对象也包含通常不能从软件访问的上下文。

[0039] CSDMA引擎在被通知应用的切换定时或者自己检测到应用的切换定时时,向处理部发出动作停止指示,并且对于在高速数据传输错误纠正(on the fly)中执行中的处理、特别是在待处理传输(out standing)或乱序传输(out of order)中需要提交(commit)处理的处理,只要确认其完成,就可以开始上下文的退避和设定。CSDMA引擎可以在完成上下文的退避和设定之后指示开始下一应用的处理。这一系列的处理不限于CSDMA引擎执行的情况,也可以是保持最新的上下文的处理部和CSDMA引擎被协作执行。

[0040] 在处理部中也可以设置多个应用用的多个队列(queue)。在各个队列中,也可以从对应的应用随时写入表示对处理部的指示的数据。处理部也可以在发生上下文切换时,切换读出数据的队列。另外,数据被写入到与某一应用对应的队列中的定时也可以不限于该应用占有处理部的期间。此时,在上下文切换执行中,各应用也能够将处理的指示持续积累

在队列中,能够缩短伴随上下文切换的处理的停止时间。另外,也可以应对在队列中写入数据的CPU上的软件与处理部(例如GPU)非同步地动作的情况。

[0041] CSDMA引擎也可以将垂直回扫期间或水平回扫期间的开始作为触发,执行上下文切换。在视频处理系统等中,在垂直回扫期间以及水平回扫期间,能够暂时停止GPU或视频管道、视频编码器等。因此,通过在短时间的垂直回扫期间中或水平回扫期间中执行上下文切换,能够隐藏上下文切换的开销。

[0042] 在从处理部退避上下文时、以及对处理部设定上下文时,也可以将DFT(Design For Test:可测性设计)用的扫描链、或者存储器BIST(Built-In Self-Test:内建自测)用作路径。由此,经由CSDMA引擎连接处理部和存储器的系统不经由通常的数据转发所使用的芯片内总线(bus)(结构(fabric)),因此不会阻碍通常的数据转发。另外,不需要设置用于上下文切换的专用接口。另外,在处理部内,由于在保持上下文信息的电路上连接有DFT电路,所以也不需要使用通常的数据转发用的系统或专用系统。

[0043] 在处理部的功能块中,也可以仅对保持上下文的功能块(例如电路)的一部分或全部进行复用。在这种情况下,可以在以下步骤中执行上下文切换。(1)不使处理部暂时停止,对被复用的其他电路预先设定上下文。(2)使处理部的处理停止(也可以等待紧前面处理的提交),选择在上述(1)中设定了上下文的电路,开始执行下一应用。(3)使在上述(1)中正在执行的应用的上下文退避。由此,使上下文切换的处理和处理部的处理在时间上重叠,能够进一步削减处理的停止时间。

[0044] 此外,也可以在处理部的功能块中,将用于保持上下文的电路进行复用,并且处理部开始下一应用的处理,而不执行用于等待提交处理的暂时停止。但是,也可以在上下文切换前的应用和上下文切换后的应用中,判定是否发生因以不同的设定使同一电路动作等、两个应用的设定不是共同的而导致的处理的失败,在必要的情况下,暂时停止直到提交完成为止。另外,也可以将一部分功能电路并行化,一边避免处理失败,一边同时执行多个应用。由此,能够并行地执行多个应用的处理(也可以是待处理传输(out standing)或乱序传输(out of order)的处理),能够缩短基于上下文切换的停止时间。另外,能够提高电路利用效率、性能可扩展性以及灵活性。

[0045] <第一实施例>

[0046] 图1表示第一实施例的计算机10的结构。计算机10可以是根据多个客户端装置的请求而并行处理多个应用(游戏等)并且向各个客户端装置提供各个应用的处理结果(例如,图像数据)的服务器。另外,计算机10也可以是PC或游戏机(视频游戏控制台等),该PC或游戏机以时分方式并行执行多个应用,使各应用的处理结果(例如图像数据)显示在显示装置上。

[0047] 计算机10包括SOC(System On a Chip:片上系统)11和主存储器22。SOC11是在一个芯片上安装了数据处理系统的功能的集成电路产品。SOC11包括CPU12、GPU14、视频编码器16、视频管道18和视频定时发生器20。

[0048] CPU12以时分方式执行成为并行处理的对象的多个应用。另外,CPU12执行用于管理成为并行处理的对象的多个应用的软件(以下也称为“管理软件”。)。管理软件也可以是OS或中间件等位于应用之下的层的软件,在实施例中,管理多个应用的执行顺序或上下文切换。另外,管理软件的一部分功能也可以由GPU14或视频编码器16等(后述的处理部46)执

行。

[0049] GPU14根据在CPU12中以时分方式执行的多个应用的指示，以时分方式执行与多个应用有关的图像处理或通用计算处理。例如，GPU14生成多个应用各自的图像数据，并将生成的图像数据写入与各应用对应的帧缓冲器。

[0050] 视频编码器16以时分方式执行并行处理对象的多个应用的图像的压缩处理。例如，视频编码器16从与多个应用对应的多个帧缓冲器中依次读出各应用的图像数据，并对读出的图像数据执行压缩编码处理。

[0051] 视频管道18是由图像供给源的其他块(例如GPU14或未图示的视频解码器等)提供的图像，换言之，以时分方式对并行处理对象的多个应用的图像执行分辨率变换、画质变换或复用。

[0052] GPU14、视频编码器16、视频管道18之间的图像交接的情形包括以下三种模式。

[0053] (1)GPU14→视频管道18→视频编码器16。

[0054] (2)GPU14→视频编码器16。

[0055] (3)未图示的其他块(视频解码器等)→视频管道18→视频编码器16。其中，视频管道18中还被输入GPU14的处理结果。

[0056] 即，视频编码器16(A)直接参照CPU12生成的图像的模式、和(B)参照视频管道18加工后的图像的模式中的哪一个都有。

[0057] 这样，GPU14、视频编码器16、视频管道18以时分方式执行与成为并行处理的对象的多个应用有关的处理，以下，将GPU14、视频编码器16、视频管道18统称为“处理部46”。

[0058] 视频定时发生器20将与图像的显示有关的各种定时(也可称为视频定时)通知给各设备。在实施例中，视频定时发生器20向CPU12、GPU14、视频编码器16、视频管道18发送用于通知显示装置的垂直回扫期间(或者与垂直回扫期间对应的期间)的开始的信号。

[0059] 另外，SOC11还具备CSDMA引擎40、CSDMA引擎42、CSDMA引擎44(统称时称为“CSDMA引擎48”。)。CSDMA引擎48是通过DMA执行用于上下文切换的上下文转发的数据转发部。

[0060] CSDMA引擎40伴随GPU14中的上下文切换，在GPU14和主存储器22之间转发上下文。CSDMA引擎42伴随视频编码器16中的上下文切换在视频编码器16和主存储器22之间转发上下文。CSDMA引擎44伴随视频管道18中的上下文切换在视频管道18和主存储器22之间转发上下文。

[0061] 路径50是基于GPU14的通常的存储器访问用(包括命令的读入和图像数据的存储)的路径。路径52是从由CPU12执行的软件(例如管理软件)向GPU14以及CSDMA引擎40的访问路径(设定、确认、通知、控制用)。路径52以同样的目的也与其他块连接。另外，在图中的线的交点有黑圈的部位表示连接了线的状态，在线的交点没有黑圈的部位表示没有连接线的状态。路径54是从主存储器22到GPU14的上下文设定路径。路径56是从GPU14到主存储器22的上下文退避路径。

[0062] 路径58是CSDMA引擎40监视GPU14的处理状态的路径。路径60是CSDMA引擎40向GPU14指示停止和开始(再次开始)应用处理的路径。路径62是从视频定时发生器20向GPU14及CSDMA引擎40通知垂直回扫期间的开始的路径。路径62以同样的目的也与其他块连接。视频编码器16与CSDMA引擎42之间的各路径、视频管道18与CSDMA引擎44之间的各路径也相同。

[0063] 主存储器22存储由SOC11参照或更新的数据。主存储器22存储GPU命令缓冲器30、GPU上下文32、视频编码器上下文34和视频管道上下文36。

[0064] GPU上下文32是与在SOC11中被并行执行的多个应用(例如App A、App B等)有关的在GPU14中使用的上下文。同样地,视频编码器上下文34是与在SOC11中被并行执行的多个应用有关的在视频编码器16中使用的上下文。视频管道上下文36是与SOC11中被并行执行的多个应用有关的在视频管道18中使用的上下文。

[0065] GPU命令缓冲器30包括与由SOC11并行处理的多个应用对应的多个队列(例如,用于App A用、App B用等)。在GPU命令缓冲器30的各队列中,通过CPU12写入并存储用于指示与各应用有关的处理内容的命令。换言之,由CPU12所执行的各应用在与本应用对应的GPU命令缓冲器30的队列中存储描绘指示等命令。GPU14从与执行对象的应用对应的GPU命令缓冲器30的队列中读出命令。GPU14在切换执行对象的应用时,对读出命令的队列进行切换。

[0066] 这样,通过设置GPU命令缓冲器30,CPU12能够与处理部46(例如GPU14、视频编码器16、视频管道18)非同步地动作。即,在CPU12中被并行(时分等)执行的各应用在处理部46进行其他应用的处理的期间,也能够将与本应用有关的命令随时存储在GPU命令缓冲器30中。换言之,也能够应对对GPU命令缓冲器30进行写入的CPU12上的软件与处理部46非同步地动作的情况。由此,在处理部46中的上下文切换执行中,CPU12上的各应用也能够将处理的指示堆积在GPU命令缓冲器30中,因此能够抑制处理的延迟。

[0067] 另外,在实施例的SOC11中,由CPU12执行的管理软件进行上下文切换的初始设定。例如,管理软件针对成为并行处理的对象的多个应用的每一个,在主存储器22中确保GPU上下文32、视频编码上下文34和视频管道上下文36的区域。另外,管理软件将各应用的上下文存储位置(地址等)通知给CSDMA引擎48(CSDMA引擎40、CSDMA引擎42、CSDMA引擎44)。

[0068] 另外,管理软件将成为并行处理的对象的多个应用的执行顺序通知给处理部46和CSDMA引擎48。处理部46和CSDMA引擎48基于来自管理软件的通知或设定,掌握成为本次执行对象的应用或成为下次执行对象的应用。

[0069] 虽然在图1中未图示,但作为变形例,也可以在CPU12中设置用于上下文切换的CSDMA引擎48。在这种情况下,CPU12也可以与处理部46同步动作。

[0070] 图2是表示图1的处理部46(具体而言是GPU14、视频编码器16、视频管道18)和CSDMA引擎48的详细结构的框图。图2的处理部46和CSDMA引擎48的组分别对应于图1的GPU14和CSDMA引擎40的组、视频编码器16和CSDMA引擎42的组、以及视频管道18和CSDMA引擎44的组。即,图2的处理部46的结构可以应用于GPU14、视频编码器16、视频管道18中的至少一个。另外,图2的CSDMA引擎48的结构能够应用于CSDMA引擎40、CSDMA引擎42、CSDMA引擎44中的至少一个。

[0071] 在本公开的框图中示出的各个块在硬件方面能够由以计算机的CPU存储器为首的元件或机械装置来实现,并且在软件方面由计算机程序等来实现,但是这里描述了通过它们的协作来实现的功能块。本领域技术人员能够理解这些功能块能够通过硬件、软件的组合以各种形式实现。

[0072] 处理部46具备第一功能电路70、第二功能电路71。第一功能电路70和第二功能电路71执行与执行对象的应用有关的数据处理(例如图像生成处理或压缩解压缩处理)。处理部46基于来自CPU12上的管理软件的通知或设定等,识别执行对象的应用,对执行对象的应

用分配第一功能电路70以及第二功能电路71。

[0073] 处理部46从主存储器22读出与执行对象的应用有关的各种数据，并输入到第一功能电路70和第二功能电路71。例如，作为处理部46的GPU14从与执行对象的应用对应的GPU命令缓冲器30的队列中读出与该应用有关的命令。另外，GPU14还从主存储器22读出图像描绘所需的其他数据。

[0074] 第一功能电路70是保持上下文的电路，该上下文是在应用的执行中被更新的数据，是处理的再现所需的数据。另外，第一功能电路70是需要根据执行对象的应用来替换上下文的电路。第一功能电路70包括状态机72、寄存器73和工作存储器74。状态机72包含状态寄存器，保持由处理部46执行中的各处理的当前状态，换言之，保持状态迁移的状况。寄存器73保持设定及处理中的数据或处理的结果。工作存储器74是在处理部46的存储器中保持根据处理而被更新的内部数据或描述符以及微代码的区域。

[0075] 第二功能电路71是不保持上下文的电路，该上下文是在应用的执行中被更新的数据，是处理的再现所需的数据。另外，第二功能电路71是不需要初始化的电路或者能够一并复位的电路，例如是进行运算处理等的电路。第二功能电路71包括随机逻辑75、运算器76和工作存储器77。随机逻辑75包含基于硬接线的功能，并包含不需要初始化或可进行一定初始化的触发电路(锁存器)。运算器76包括触发电路(锁存器)，该触发电路不需要数据路径和初始化、或者可进行一定初始化。工作存储器77是在处理部46的存储器中工作存储器74以外的区域。

[0076] CSDMA引擎48包括检测部80、监视部82、指示部84、转发执行部86。检测部80检测处理部46中的执行对象的应用的切换定时(以下也称为“App切换定时”。)。实施例的检测部80将垂直回扫期间的开始定时检测为App切换定时，具体而言，根据从视频定时发生器20被通知了垂直回扫期间的开始定时的情况，判断为是App切换定时。

[0077] 作为变形例，检测部80也可以将水平回扫期间的开始定时检测为App切换定时。具体而言，检测部80也可以根据从视频定时发生器20被通知了水平回扫期间的开始定时的情况，判断为是App切换定时。作为另一变形例，CPU12上的管理软件也可以将App切换定时通知给处理部46和CSDMA引擎48。检测部80也可以基于来自CPU12上的管理软件的通知来检测App切换定时。

[0078] 在检测到App切换定时的情况下，监视部82监视处理部46中的与应用有关的处理的执行状态。处理部46中的与应用有关的处理包括粒度小的多个内部处理，多个内部处理被并行执行。监视部82确认各个内部处理的执行状态(例如各个内部处理是否完成)。

[0079] 指示部84指示处理部46停止和开始(再次开始)与应用有关的处理。也可以在检测到App切换定时的情况下，指示部84指示处理部46停止与应用有关的处理。处理部46可以基于来自CSDMA引擎48的指示部84的指示来检测App切换定时，也可以基于来自CPU12上的管理软件或视频定时发生器20的通知来检测App切换定时。另外，监视部82和指示部84是可选的结构，CSDMA引擎48也可以是不包含监视部82和指示部84的一方或双方的结构。

[0080] 当检测到App切换定时时，转发执行部86执行上下文转发处理。具体而言，作为上下文转发处理，转发执行部86不经由管理多个应用的软件(在实施例中为CPU12上的管理软件)的处理，而使由处理部46执行的应用的上下文从处理部46(第一功能电路70)退避到主存储器22(该应用的上下文存储区域)。

[0081] 此外，作为上下文转发处理，转发执行部86从主存储器22(应用的上下文存储区域)读出由处理部46接下来要执行的该应用的上下文，而不经由管理多个应用的软件(在本实施例中，CPU12上的管理软件)的处理。转发执行部86在处理部46(第一功能电路70)中设定所读出的上下文。当检测到由转发执行部86进行的上下文设定处理的完成时，指示部84也可以指示处理部46开始(再次开始)与应用有关的处理。

[0082] 成为由转发执行部86进行的转发的对象的上下文是再现伴随处理部46中的上下文切换而被中断的时间点的应用的处理(上述的各个内部处理)所需的数据。在实施例中，成为由转发执行部86进行的转发的对象的上下文是被保持在处理部46的第一功能电路70中的上下文的全部，包含不能从软件(例如CPU12上的软件以及处理部46上的软件)访问的上下文。不能从软件访问的上下文例如可以是被保持在处理部46的状态机72中的上下文，也可以是表示被包含在应用的处理中的各个内部处理的状态迁移状况的上下文。

[0083] 图3按时序表示了处理部46(例如，GPU14)中的视频绘制处理。该图表示以时分方式执行与4个应用(App A、App B等)有关的视频描绘处理的例子。在本实施例中，各应用需要以60fps(frames per second:帧每秒)生成图像。处理部46通过分别以1/240秒生成4个应用的图像，实现每一个应用60fps的帧速率。

[0084] 但是，各应用的处理时间基于描绘内容而始终变动，未必能够在1/240秒以内完成处理。此时，如上所述，在实施例中，垂直回扫期间的开始定时(图3的定时90)被检测为App切换定时(换言之，切换图像写入目的地的帧缓冲器的翻转(flip)定时)。这里，在图像的生成还未完成的情况下，通过将处理部46交给下一应用，能够使各应用能够使用处理部46的时间均等。另外，在定时90之前不能生成图像的情况下，成为帧丢失(コマ落ち)，该应用的前图像被反复交给后级处理。

[0085] 图3的期间92相当于垂直回扫期间，通过使用CSDMA引擎48，在该期间以内高速地执行处理的再现所需的全部上下文的退避、设定。这样，箭头所示的图像生成期间和包含上下文切换的期间92(垂直回扫期间)的合计被控制为成为1/240秒以内。

[0086] 另外，在应用比垂直回扫期间的开始定时较早地完成图像的生成时，应用也可以向CPU12上的管理软件通知能够执行切换帧缓冲器的翻转。并且，CPU12上的管理软件也可以代替垂直回扫期间的开始定时，自己将App切换定时通知给处理部46。由此，该应用的后级处理或后续的应用提前开始处理，能够削减延迟，或提高处理部的利用效率。

[0087] 另外，在变形例中，水平回扫期间的开始定时可以被检测为App切换定时，也可以在水平回扫期间内进行上下文切换。

[0088] 图4表示视频定时的例子。该图表示垂直回扫期间94(Vblank)和水平回扫期间96(Hblank)。另外，在服务器描绘视频并将该视频向客户端分发的情况下，有时使用视频定时的外部显示器不与服务器连接。在该情况下，为了维持与以往的应用的兼容性，有时也生成视频定时，在实施例中利用该视频定时来检测App切换定时。

[0089] 在垂直回扫期间以及水平回扫期间，GPU14或视频编码器16、视频管道18也能够暂时停止处理。在实施例中，通过在短时间的垂直回扫期间或水平回扫期间中执行上下文切换，能够隐藏上下文切换的开销。

[0090] 接着，说明上下文的转发路径。

[0091] 图5表示SOC11的结构。SOC11内的各个电路通常经由总线结构100和存储器接口

102访问主存储器22。在SOC11(即半导体芯片)中内置有用于测试功能是否正确动作的DFT(Design For Test:可测性设计)功能(DFT控制电路104等)。另外,BIST(Built-In Self Test:内建自测)具有自我诊断功能,设有逻辑电路用和存储电路用。扫描链106网罗性地连接到SOC11内的各电路内部,用于基于BIST或芯片外部的指示测试各电路。通过使用扫描链106,能够进行SOC11内部的各状态值的设定以及读出。

[0092] 图6也表示SOC11的结构。如该图所示,CSDMA引擎48能够使用通常的数据转发用的总线(总线结构100)和路径114,进行上下文的退避和设定。路径114是用于访问保持不能从软件访问的上下文的电路的路径。但是,在该结构中,由于在上下文的转发中,与其他电路(CPU12等)共享总线结构100,所以存在妨碍通常的数据转发的缺点。

[0093] 图7也表示SOC11的结构。如该图所示,CSDMA引擎48也可以使用上下文转发用的专用总线(专用路径108)和路径114,进行上下文的退避和设定。但是,在该结构中,需要在SOC11上新安装专用路径108,存在SOC11的成本增大的缺点。

[0094] 图8也表示SOC11的结构。如该图所示,实施例的CSDMA引擎48使用DFT用的电路(包含扫描链106)及BIST用的电路(BIST电路110),进行上下文的退避及设定。例如,CSDMA引擎48经由扫描链106从第一功能电路70读出与本次被执行的应用有关的上下文,并经由BIST电路110将所读出的上下文保存在主存储器22中。另外,CSDMA引擎48经由BIST电路110从主存储器22读出与接下来要被执行的应用有关的上下文,并经由扫描链106对第一功能电路70设定所读出的上下文。

[0095] 这样,通过使用扫描链106,还能够访问没有软件进行访问的路径的内部电路的上下文,能够进行处理的再现所需要的上下文的退避和设定。另外,总线结构100、专用路径108、路径114都不需要新设置,能够以低成本实现上下文的转发。此外,可以避免因上下文的转发而阻碍SOC11中的通常的数据转发。

[0096] 接着,说明上下文切换的开始定时。

[0097] CSDMA引擎48的转发执行部86使被保持在处理部46的第一功能电路70(状态机72等)中的应用的上下文退避到主存储器22中。从处理部46被退避到主存储器22的应用的上下文、以及从主存储器22被设定到处理部46的应用上下文是被保持在状态机72等中的数据,包含表示在上下文切换开始时间点在处理部46中未完成的处理(小粒度的内部处理)的当前状态的数据。

[0098] 在检测到App切换定时的情况下,处理部46的第一功能电路70和第二功能电路71与此前作为执行对象的应用的内部处理的状态无关,换言之,即使内部处理未完成,也立即停止内部处理。在检测到App切换定时的情况下,CSDMA引擎48的转发执行部86与此前作为执行对象的应用的内部处理的状态无关,换言之,即使内部处理未完成,也开始上下文的转发处理。

[0099] 图9A表示第一实施例中的上下文切换时的内部处理的例子。该图中,横方向是时间经过轴,一个箭头表示与一个应用有关的处理中包含的小粒度的内部处理。各箭头的左端表示内部处理的开始定时,右端表示内部处理的结束定时。

[0100] 如图9A所示,实施例的处理部46在接受到上下文切换执行指示的时间点(即检测到App切换定时的时间点),立即停止所有的内部处理,开始上下文切换。另外,在该时间点未完成(即高速数据传输错误纠正(on the fly)和待处理传输(out standing))的内部处

理(图中的虚线)也立即中断,将该时间点的上下文保存在主存储器22中。另外,图中的单点划线所示的内部处理是预定在接受到上下文切换执行指示之后开始的处理,因此不开始。CSDMA引擎48的转发执行部86在接受到上下文切换执行指示的时间点(即,检测到App切换定时的时间点)立即开始上下文转发处理(退避和设定)。

[0101] 在实施例的SOC11中,使处理部46中的各内部处理的状态,即包含从软件不能确认的数据(例如状态机72中被保持的数据等)的上下文退避。由此,在上下文切换执行指示接受时间点处于执行途中的内部处理,在下一次执行时,能够复原其状态而再次开始。因此,在实施例的SOC11中,能够在上下文切换执行指示的接受时间点(App切换定时的检测时间点)立即开始上下文切换,能够抑制处理的延迟。

[0102] 作为变形例,也可以在检测到App切换定时的情况下,CSDMA引擎48的转发执行部86待机至由监视部82确认处理部46中的应用的内部处理完成了为止,在确认处理部46中的应用的内部处理完成后,开始上下文的转发处理。

[0103] 图9B表示变形例中的上下文切换时的内部处理的例子。在该图中,用虚线表示在上下文切换执行指示时已开始但未完成(即高速数据传输错误纠正(on the fly)和待处理传输(out standing))的内部处理。在本变形例中,在等待到上下文切换执行指示时已开始但未完成的内部处理结束之后,开始上下文切换。

[0104] 通过不中断地完成在上下文切换执行指示时已开始但未完成的处理,在要求与上下文切换对象块外的处理协作的情况下,或要求实时性的情况下,能够防止处理失败(不匹配的发生等)。此外,可以降低成为转发对象的上下文的数据量。

[0105] <第二实施例>

[0106] 关于本实施例,以下以与此前的实施例的不同点为中心进行说明,省略共同点的说明。在本变形例的结构要素中,对与此前的实施例的结构要素相同或对应的结构要素标注相同的标号进行说明。

[0107] 图10表示第二实施例的处理部46的结构。图10的处理部46和CSDMA引擎48的组分别对应于图1的GPU14和CSDMA引擎40的组、视频编码器16和CSDMA引擎42的组、以及视频管道18和CSDMA引擎44的组。即,图10的处理部46的结构能够应用于GPU14、视频编码器16、视频管道18中的至少一个。另外,图10的CSDMA引擎48的结构能够应用于CSDMA引擎40、CSDMA引擎42、CSDMA引擎44中的至少一个。

[0108] 第二实施例的处理部46包括多个保持应用的上下文的第一功能电路70。图10的处理部46具备2个第一功能电路70(第一功能电路70a和第一功能电路70b),但也可以具备3个以上的第一功能电路70。

[0109] 第二实施例的CSDMA引擎48的转发执行部86,在处理部46使用被保持在某第一功能电路70中的应用的上下文来执行该应用的处理的期间,将其他第一功能电路70中所保持的其他应用的上下文作为对象来执行上下文转发处理。

[0110] 例如,在处理部46使用被保持在第一功能电路70a中的App A的上下文来执行App A的处理的期间,转发执行部86(1)使被保持在第一功能电路70b中的App D(即上次执行的App)的上下文退避到主存储器22中。同时,转发执行部86(2)读出被保持在主存储器22中的App B(即,接下来成为执行对象的App)的上下文,并且在第一功能电路70b中设定该上下文。

[0111] 处理部46在使用被保持第一功能电路70a中的应用的上下文来执行该应用的处理的过程中检测到App切换定时时，立即中止该应用的处理。然后，处理部46使用被保持在第一功能电路70b中的其他应用的上下文，开始该其他应用的处理。由此，能够使伴随上下文切换的待机时间(即等待上下文的退避和设定的时间)大致为零。

[0112] <第三实施例>

[0113] 关于本实施例，以下以与此前的实施例的不同点为中心进行说明，省略共同点的说明。在本变形例的结构要素中，对与此前的实施例的结构要素相同或对应的结构要素标注相同的标号进行说明。

[0114] 第三实施例的处理部46的结构与图10所示的第二实施例的处理部46的结构相同。处理部46使用被保持在第一功能电路70a中的第一应用(例如App A)的上下文来执行与第一应用有关的处理。另外，处理部46使用被保持在第一功能电路70b中的第二应用(例如App B)的上下文来执行与第二应用有关的处理。

[0115] 在检测到App切换定时时，处理部46基于由CPU12上的管理软件进行的事先设定等，识别接下来应该执行的应用。在此，假设为处理部46中的执行对象从第一应用(例如App A)切换为第二应用(例如App B)。处理部46在与第一应用有关的处理中的不能与第二应用所相关的处理混流(换言之，并行执行或同时执行)的处理完成后，并行执行与第一应用有关的处理中的能够与第二应用所相关的处理混流的处理和与第二应用有关的处理。

[0116] 图11表示第三实施例中的上下文切换时的内部处理的例子。该图表示上述第一应用(切换前的应用)的内部处理。该图的虚线和双点划线表示在上下文切换执行指示时已开始但未完成(高速数据传输错误纠正(on the fly)和待处理传输(out standing))的内部处理。其中，虚线表示不能与第二应用所相关的处理混流的内部处理，双点划线表示能够与第二应用所相关的处理混流的内部处理。

[0117] 能够混流的处理例如是使用数据路径或运算器的内部处理，是在时间上与前后的处理内容没有依赖关系，并且在空间上与作为电路连接在周边的功能的处理内容没有依赖关系的内部处理。不能混流的处理例如是具有上述依赖关系的内部处理，另外，是如果与第二应用所相关的处理并行执行则有可能失败的内部处理。处理部46也可以预先存储表示各内部处理是否能够与其他应用的内部处理混流的数据，另外，CPU12上的管理软件也可以将该数据预先存储在处理部46中。

[0118] 处理部46即使接受上下文切换执行指示，也继续与第一应用有关的和待处理传输(out standing)的内部处理。处理部46等待至高速数据传输错误纠正(on the fly)和待处理传输(out standing)的内部处理中不能与第二应用(换言之，其他应用)混流的处理(图11的虚线)结束之后，开始上下文切换。换言之，处理部46在高速数据传输错误纠正(on the fly)和待处理传输(out standing)的内部处理中的不能与第二应用混流的处理全部结束的情况下，不管能够与第二应用混流的处理(图11的双点划线)是否结束，都开始上下文切换。

[0119] 在执行上下文切换后，处理部46(例如第一功能电路70a)继续执行与第一应用有关的高速数据传输错误纠正(on the fly)和待处理传输(out standing)的内部处理中的能够与第二应用混流的内部处理。与此并行地，处理部46(例如第一功能电路70b)执行与第二应用有关的内部处理。当能够混流的内部处理全部结束时，在保持第一应用的上下文的

电路(例如第一功能电路70a)中记录执行结果的状态。

[0120] CSDMA引擎48的转发执行部86在由监视部82检测到与第一应用有关的高速数据传输错误纠正(on the fly)和待处理传输(out standing)的内部处理全部完成了时,使第一应用的上下文从保持第一应用的上下文的电路(例如第一功能电路70a)退避到主存储器22。此外,转发执行部86从主存储器22向上述电路(例如,第一功能电路70a)设定在处理部46中预定接下来执行的第三应用的上下文。

[0121] 根据第三实施例的SOC11,能够防止与第一应用有关的处理的失败,并且能够提前开始与第二应用有关的处理,能够抑制处理部46中的处理的延迟。

[0122] 图12A、图12B、图12C表示上下文切换中的处理部46和CSDMA引擎48的动作。图12A表示第一实施例的处理部46的动作(立即上下文切换)。处理部46执行前一App处理和下一App处理,CSDMA引擎48执行前一App的上下文退避处理和下一App的上下文设定处理。图12B表示第一实施例的变形例中记载的处理部46的动作(全部内部处理完成后的上下文切换)。上下文退避处理的开始延迟了等待前一App处理结束的时间。

[0123] 在如第二实施例中记载的那样保持上下文的第一功能电路70被复用的情况下,在图12A和图12B的任一个所示的上下文切换方法中,都能够在使用一方的上下文保持电路执行应用的过程中,在另一方的上下文保持电路中事先设定预定接下来执行的应用的上下文。由此,能够使等待上下文的退避和设定的完成的时间(图12A和图12B的期间112)大致为零。

[0124] 图12C表示第三实施例的处理部46的动作(不能混流的内部处理完成后的上下文切换)。图12C中的“上下文切换开始和结束”表示将参考目的地切换到与此前不同的上下文保持电路以执行下一App。例如,表示将参考目的地从第一功能电路70a切换为第一功能电路70b。如上所述,当前一App能够混流的内部处理结束时,CSDMA引擎48使前一App的上下文从前一App的上下文保持电路(例如,第一功能电路70a)退避到主存储器22。

[0125] <第四实施例>

[0126] 关于本实施例,以下以与此前的实施例的不同点为中心进行说明,省略共同点的说明。在本变形例的结构要素中,对与此前的实施例的结构要素相同或对应的结构要素标注相同的标号进行说明。

[0127] 图13表示第四实施例的处理部46的结构。第四实施例的处理部46除了图10所示的第三实施例的结构以外,还具备多个执行在多个应用间不能混流的内部处理的功能块。执行在多个应用之间不能混流的内部处理的功能块例如(1)也可以是执行与“在时间上前后的处理内容、或在空间上作为电路连接到周边的功能的处理内容”存在依赖关系的处理的电路。或者,(2)也可以是当投入了以各应用的不同设定为前提的内部处理时,发生由于两个应用的设定不是共同的而引起内处理的失败的电路。

[0128] 图13所示的处理部46具有多个随机逻辑75,作为执行在多个应用之间不能混流的内部处理的功能块的例子。图13的处理部46具备2个随机逻辑75(随机逻辑75a和随机逻辑75b),但也可以具备3个以上的随机逻辑75。

[0129] 在检测到App切换定时、应该将处理部46中的处理对象从第一应用(例如App A)切换到第二应用(例如App B)时,处理部46将与第一应用有关的高速数据传输错误纠正(on the fly)和待处理传输(out standing)的内部处理中的不能与第二应用所相关的处理混

流的内部处理，使用一方的随机逻辑75(例如随机逻辑75a)继续执行。另外，处理部46不等待上述不能混流的内部处理的完成，立即使用另一方的随机逻辑75(例如随机逻辑75b)，开始与第二应用有关的处理。

[0130] 根据第四实施例的SOC11，不需要等待与第一应用有关的高速数据传输错误纠正(on the fly)和和待处理传输(out standing)的内部处理中的不能与第二应用所相关的处理混流的内部处理的完成。由此，在上下文切换执行指示时，能够提前开始与第二应用有关的处理，能够更高速地实现上下文切换。

[0131] 说明第四实施例的变形例。处理部46可以将多个应用同时分配给多个被复用的功能电路，并且处理部46可以并行执行多个应用的内部处理。这里，在功能电路的复用是2路复用、并且并行处理对象的应用是4个(例如，App A～D)的情况下，也可以将所选择的2个应用(例如，App A和App B)分配给2个被复用的功能电路。在上下文切换时，也可以将剩余的2个应用(例如App C和App D)作为新的执行对象，与此前执行的两个应用(例如App A和App B)进行替换。

[0132] 例如，在图13所示的结构中，在并行执行App A的内部处理和App B的内部处理的情况下，处理部46也可以使第一功能电路70a保持App A的上下文，使第一功能电路70b保持App B的上下文。另外，处理部46也可以将与App A有关的内部处理中的与App B所相关的内部处理不能混流的内部处理分配给随机逻辑75a，将与App B有关的内部处理中与App A所相关的内部处理不能混流的内部处理分配给随机逻辑75b。另外，处理部46也可以将与App A有关的内部处理中的与App B所相关的内部处理能够混流的内部处理、和与App B有关的内部处理中的与App A所相关的内部处理能够混流的内部处理这两者分配给共同的运算器76和工作存储器77。

[0133] 根据该变形例的结构，处理部46不是以时分方式执行多个应用，而是同时执行多个应用，因此能够进一步抑制处理的延迟。另外，还能够抑制上下文切换的发生次数。另外，处理部46具有的电路的active率(换言之，激活率)提高，能够提高数据处理的性能。

[0134] <第五实施例>

[0135] 关于本实施例，以下以与此前的实施例的不同点为中心进行说明，省略共同点的说明。在本变形例的结构要素中，对与此前的实施例的结构要素相同或对应的结构要素标注相同的标号进行说明。

[0136] 图14表示第五实施例的处理部46的结构。在第五实施例的处理部46中，对保持上下文的第一功能电路70和不保持上下文的第二功能电路71这两者进行复用。图14的处理部46具备4个第一功能电路70和4个第二功能电路71，但第一功能电路70和第二功能电路71的复用不限于4路复用。

[0137] 在本实施例的结构中，在各应用的性能要件不同的情况下，处理部46在每次上下文切换时决定对一个应用分配几个第一功能电路70以及第二功能电路71。然后，处理部46对成为执行对象的各应用分配一个以上的第一功能电路70和一个以上的第二功能电路71。各应用的性能要件、或者各应用所需的第一功能电路70和第二功能电路71的个数也可以由CPU12上的管理软件设定在处理部46中。

[0138] 图15A和图15B表示应用的分配例。如图15A所示，处理部46可以将4个应用(例如，App A至App D)分别分配给1个第一功能电路70和1个第二功能电路71的组。另外，在与App

E有关的数据处理中需要与App A有关的数据处理的4倍的性能的情况下，如图15B所示，处理部46也可以对App E分配4个第二功能电路71和1个第一功能电路70。在执行App E时，3个第一功能电路70也可以未使用。

[0139] 处理部46也可以以时分方式执行图15A所示的分配中的处理和图15B所示的分配中的处理。在该情况下，通过应用在第一实施例至第四实施例中说明的上下文切换方法中的任一个，能够高速地实现图15A的状态和图15B的状态之间的迁移。

[0140] 另外，在复用了的第一功能电路70之间，也可以设置共享当前是否处于工作中(保持上下文的App是否处于执行中)、以及是怎样的上下文(App的识别信息等)的接口。另外，与处理部46连接的CSDMA引擎48也可以代替处理部46，集中多个第一功能电路70，管理在各第一功能电路70中保持哪个App的上下文。

[0141] 另外，在切换图15A的状态和图15B的状态的情况下，需要同时执行全部第一功能电路70的上下文切换。此时，也可以对各App执行第一实施例至第四实施例中说明的上下文切换方法中的任一个。并且，也可以在全部App对齐的定时(例如全部App的内部处理完成了的定时)执行上下文切换。另外，在图15A所示的状态下仅将App A切换为其他App(例如App F)的情况下，且App F的性能要件为App A的性能要件以下，则也可以仅在与App A对应的第一功能电路70和第二功能电路71中执行上下文切换。

[0142] 另外，对于第一功能电路70以及第二功能电路71，当然可以是图15A、图15B所示的方式以外的应用分配的方式。例如，在处理部46中，占有2个或3个第二功能电路71的应用也可以与其他应用同时执行。

[0143] 在上述的第二实施例～第五实施例中，对保持上下文的第一功能电路70进行了复用。根据该结构，在应用的处理中重叠，能够进行上下文的退避和设定。由此，能够简化CSDMA引擎48以及输入输出数据的系统(例如总线结构100、扫描链106、专用路径108、路径114)的转发能力。

[0144] 例如，在图3所示的视频描绘处理中，当各应用以每1/240秒(＝期间4.2毫秒)动作时，假设为垂直回扫期间为0.5毫秒，各应用的处理时间为3.7毫秒。此时，在不进行复用的结构中，需要在小于0.5毫秒的时间内执行上下文切换。另一方面，在进行了复用的可能性方面，以小于3.7毫秒来执行即可。这里，在上下文的数据量为50MB时，为了在垂直回扫期间内进行转发，需要100GB/秒以上的转发能力。另一方面，如果是应用处理时间内的转发，则只要具有13.5GB/秒以上的转发能力即可。

[0145] 以上，基于第一实施例～第五实施例说明了本公开。这些实施例是例示，本领域技术人员能够理解各结构要素或各处理过程的组合可以有各种变形例，并且这样的变形例也在本公开的范围内。

[0146] 在上述实施例中，将上下文的退避目的地设为主存储器22，但成为上下文的退避目的地的存储器(换言之，进行上下文保存的存储器)也可以是与主存储器22不同的存储器。另外，成为上下文的退避目的地的存储器的数据保持特性可以是易失性的，也可以是非易失性的。

[0147] 在成为上下文的退避目的地的存储器是非易失性的情况下，即使在挂起、休眠等处理部46的暂时停止、再次开始中切断对该存储器的电源供给，被退避至该存储器的上下文也被保持。由此，能够更有效地利用上述实施例中记载的高速的上下文切换。

[0148] 在SOC11中，在单独切断对未使用的处理部46的电源供给来实现低功率化的情况下，需要等待处理完成的划分，或者，如果上下文的退避、设定的时间长，则有可能不耐用。如果使用上述实施例中记载的高速上下文切换，则在实现低功率化的情况下也能够进行间歇动作，能够提高功率效率。

[0149] 另外，在上述实施例中，CPU12、处理部46(GPU14、视频编码器16等)和CSDMA引擎48安装在了一个硬件(SOC11)上。作为变形例，这些功能块也可以分散安装在多个硬件上。例如，各实施例的CSDMA引擎48也可以作为与安装有处理部46的装置独立的数据转发装置来实现。

[0150] 上述实施例和变形例的任意组合作为本公开的实施例也是有用的。通过组合而产生的新的实施方式同时具有组合的实施例及变形例各自的效果。另外，本领域技术人员会理解权利要求书中记载的各结构要件应起到的作用是通过实施例和变形例中所示的各结构要素的单体或它们的协作来实现的。

[0151] 产业上的可利用性

[0152] 本公开的技术可应用于处理数据的系统或装置。

[0153] 标号说明

[0154] 10计算机、11 SOC、46处理部、48 CSDMA引擎、80检测部、86转发执行部、106扫描链。

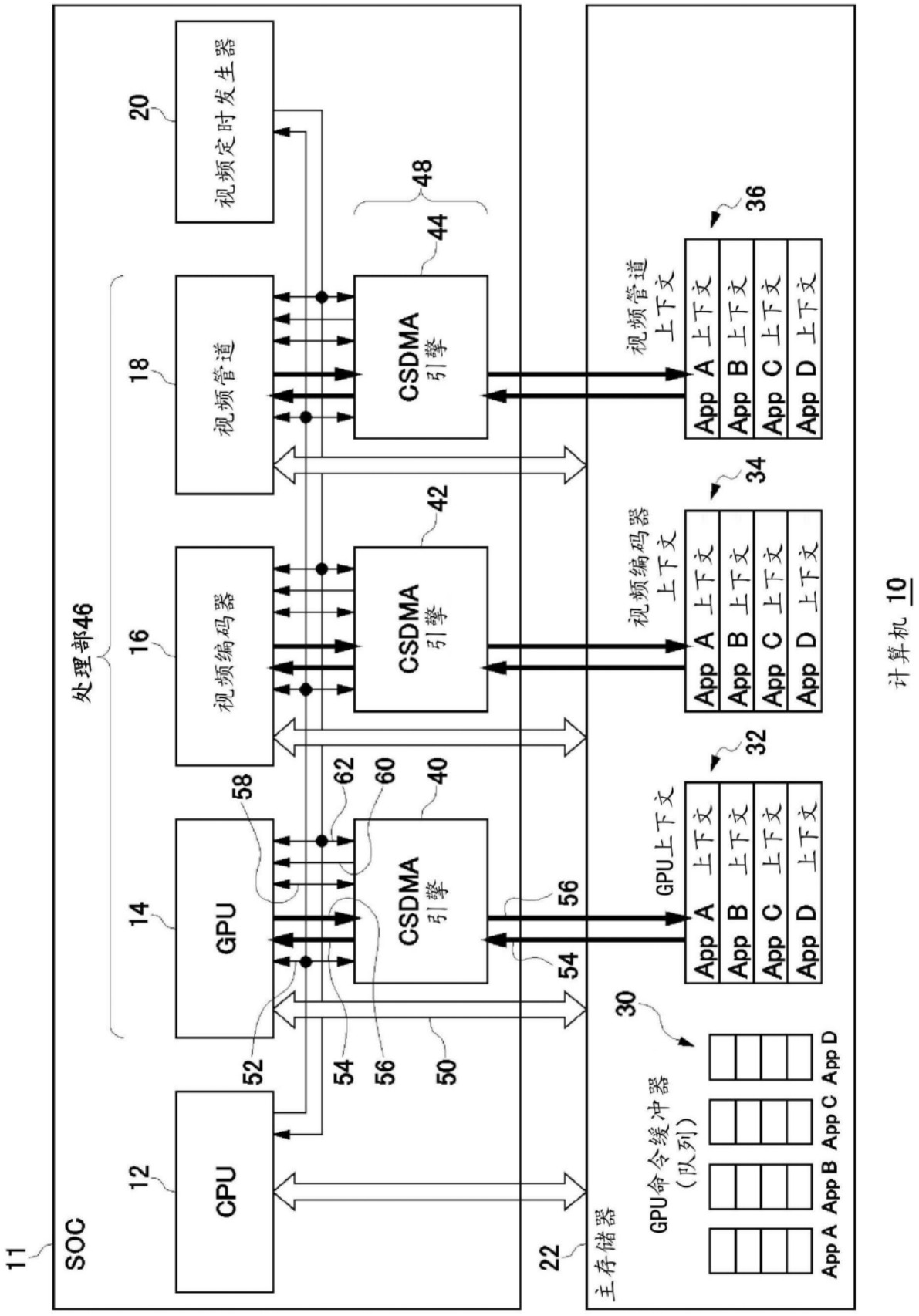

计算机 10
11
SOC
处理部46
12
CPU
14
GPU
16
视频编码器
18
视频管道
20
视频定时发生器
22
主存储器
30
GPU命令缓冲器
(队列)
App A
App B
App C
App D
32
GPU上下文
34
视频编码器
上下文
36
视频管道
上下文
App A 上下文
App B 上下文
App C 上下文
App D 上下文
40
42
44
48
CSDMA
引擎
50
52
54
56
58
60
62

图1

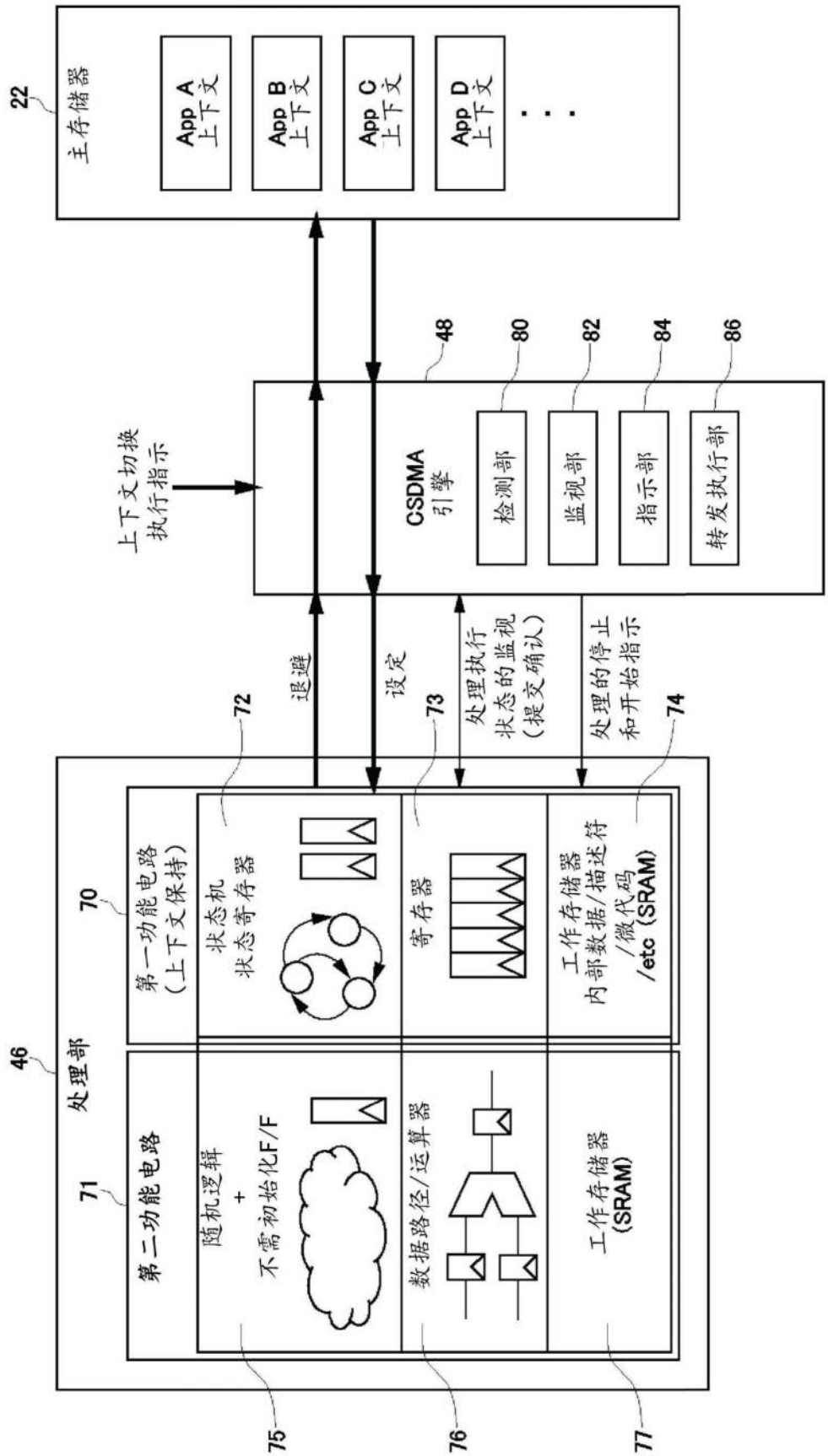

22
主存储器
App A 上下文
App B 上下文
App C 上下文
App D 上下文
48
80
82
84
86
CSDMA 引擎
检测部
监视部
指示部
转发执行部
上下文切换执行指示
退避
设定
处理执行状态的监视（提交确认）
处理的停止和开始指示
72
73
74
46
处理部
70
第一功能电路（上下文保持）
状态机 状态寄存器
寄存器
工作存储器 内部数据/描述符/微代码 /etc（SRAM）
71
第二功能电路
随机逻辑 + 不需初始化F/F
数据路径/运算器
工作存储器（SRAM）
75
76
77

图2

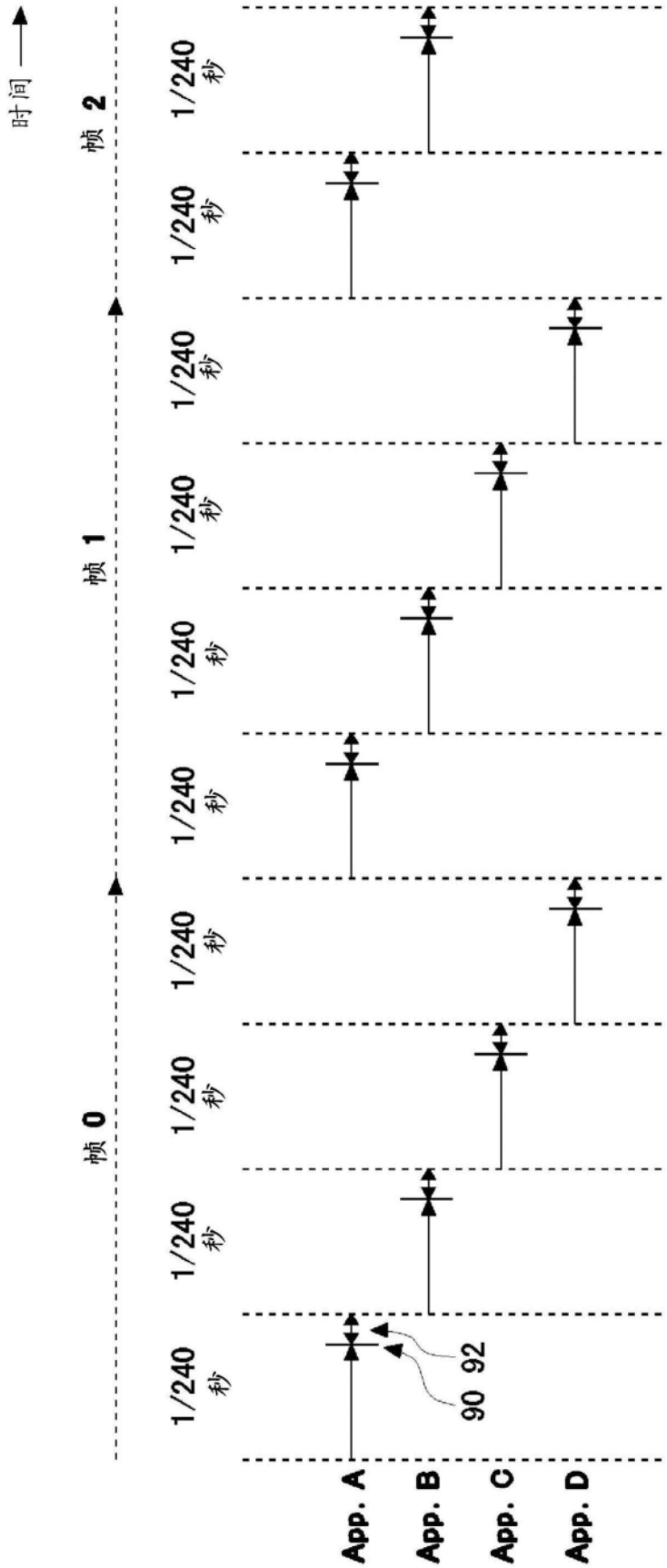


图3

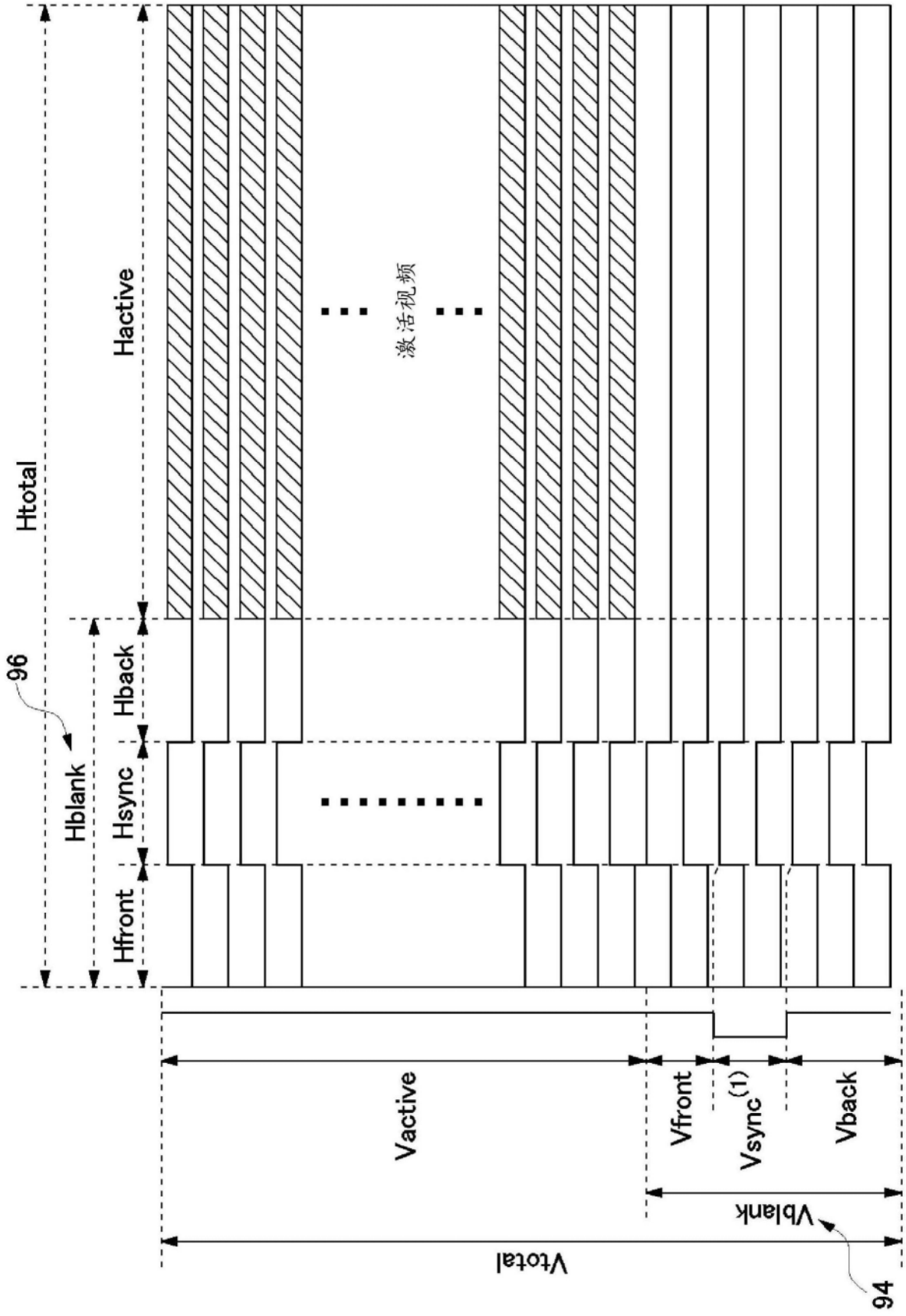

Htotal
Hactive
激活视频
96
Hblank
Hback
Hsync
Hfront
Vactive
Vfront
Vsync(1)
Vback
Vblank
Vtotal
94

图4

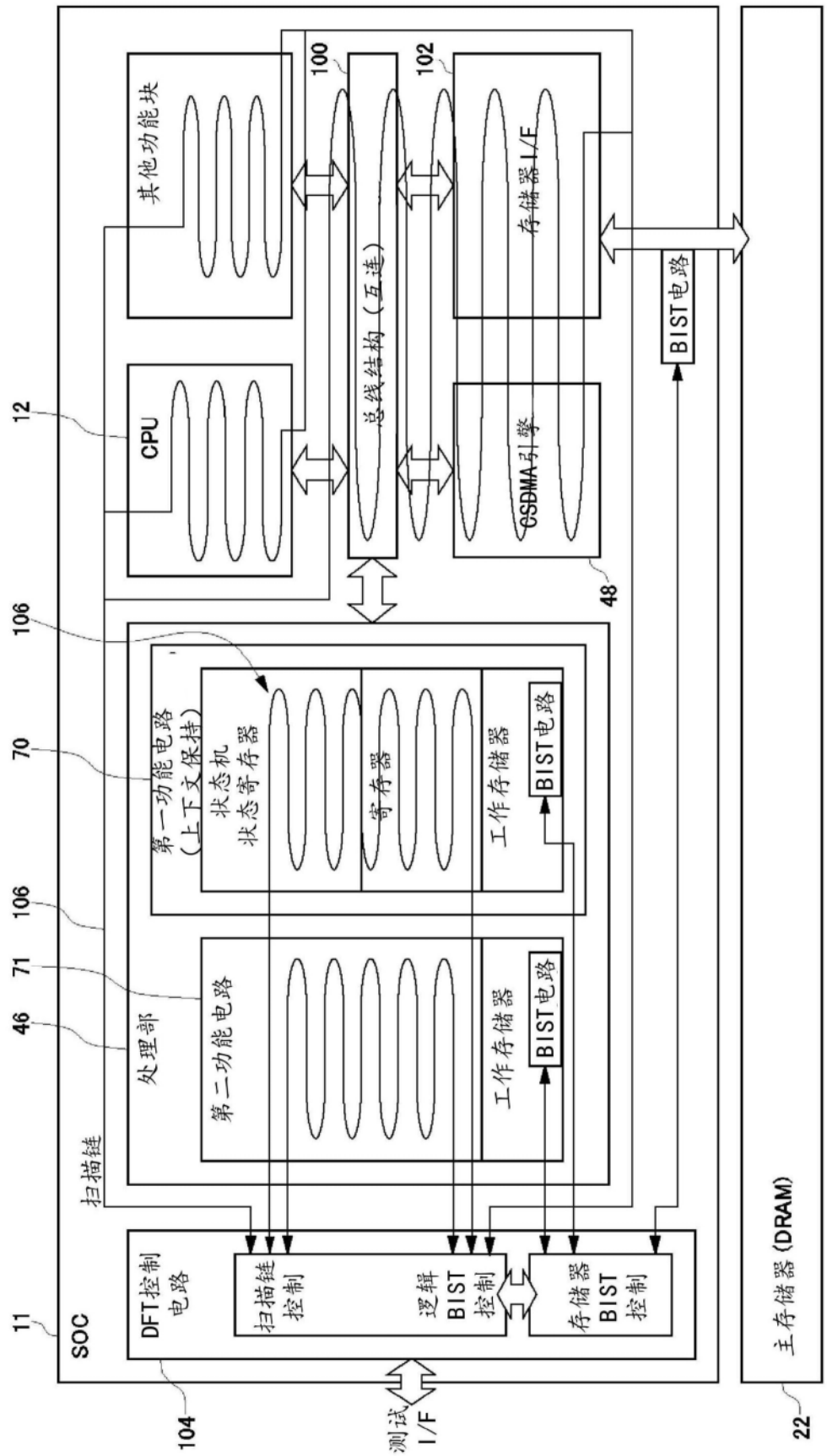

SOC
11
其他功能块
CPU
12
总线结构（互连）
100
102
存储器I/F
CSDMA引擎
48
BIST电路
106
第一功能电路
（上下文保持）
70
状态机
状态寄存器
寄存器
工作存储器
BIST电路
106
71
第二功能电路
工作存储器
BIST电路
46
处理部
扫描链
DFT控制
电路
104
扫描链
控制
逻辑
BIST
控制
存储器
BIST
控制
测试
I/F
主存储器（DRAM）
22

图5

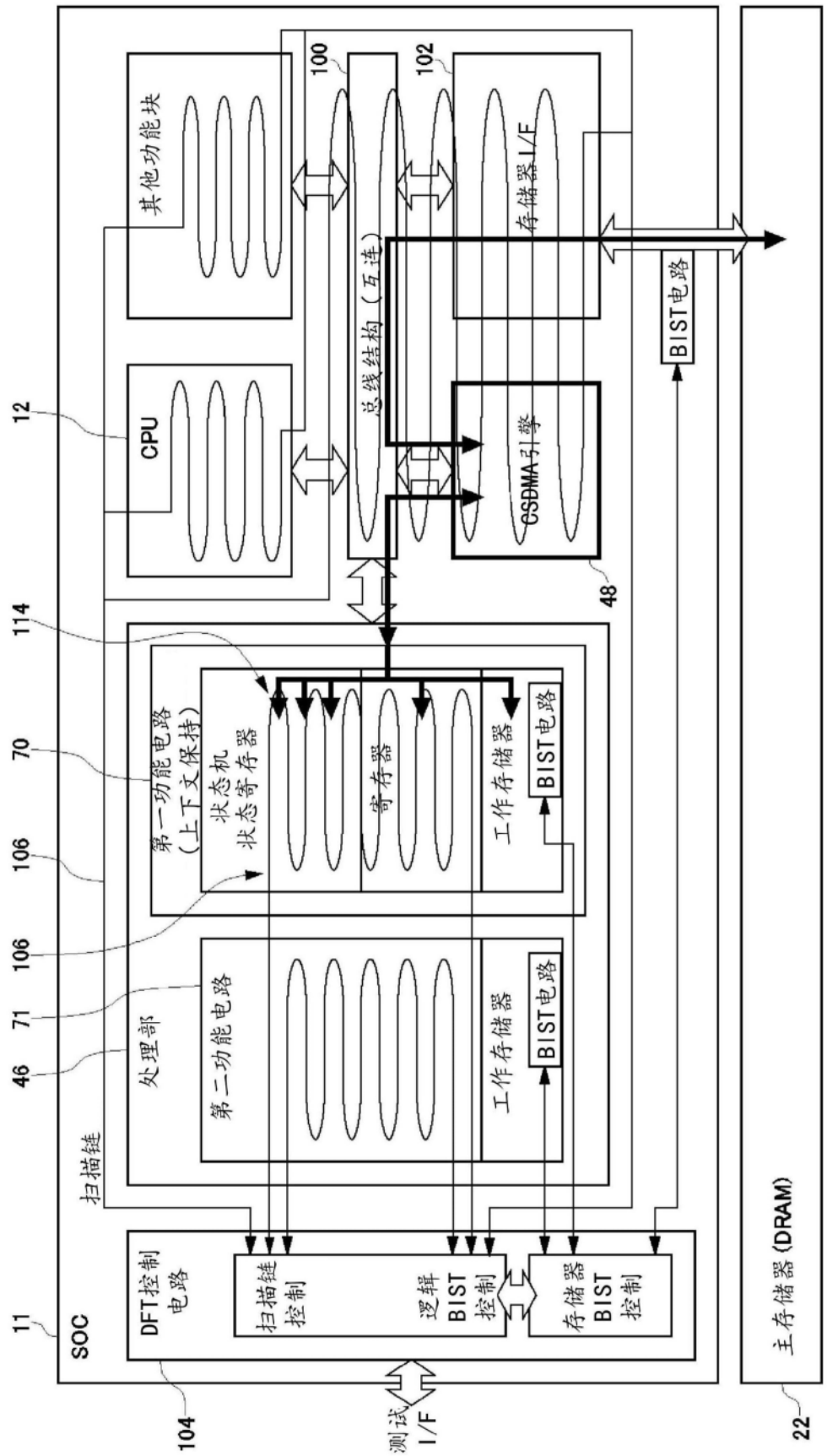

SOC
11
其他功能块
CPU
12
100
总线结构（互连）
102
存储器I/F
BIST电路
CSDMA引擎
48
114
第一功能电路
（上下文保持）
状态机
状态寄存器
70
寄存器
工作存储器
BIST电路
106
106
第二功能电路
71
处理部
46
工作存储器
BIST电路
扫描链
DFT控制
电路
104
扫描链
控制
逻辑
BIST
控制
存储器
BIST
控制
测试
I/F
主存储器(DRAM)
22

图6

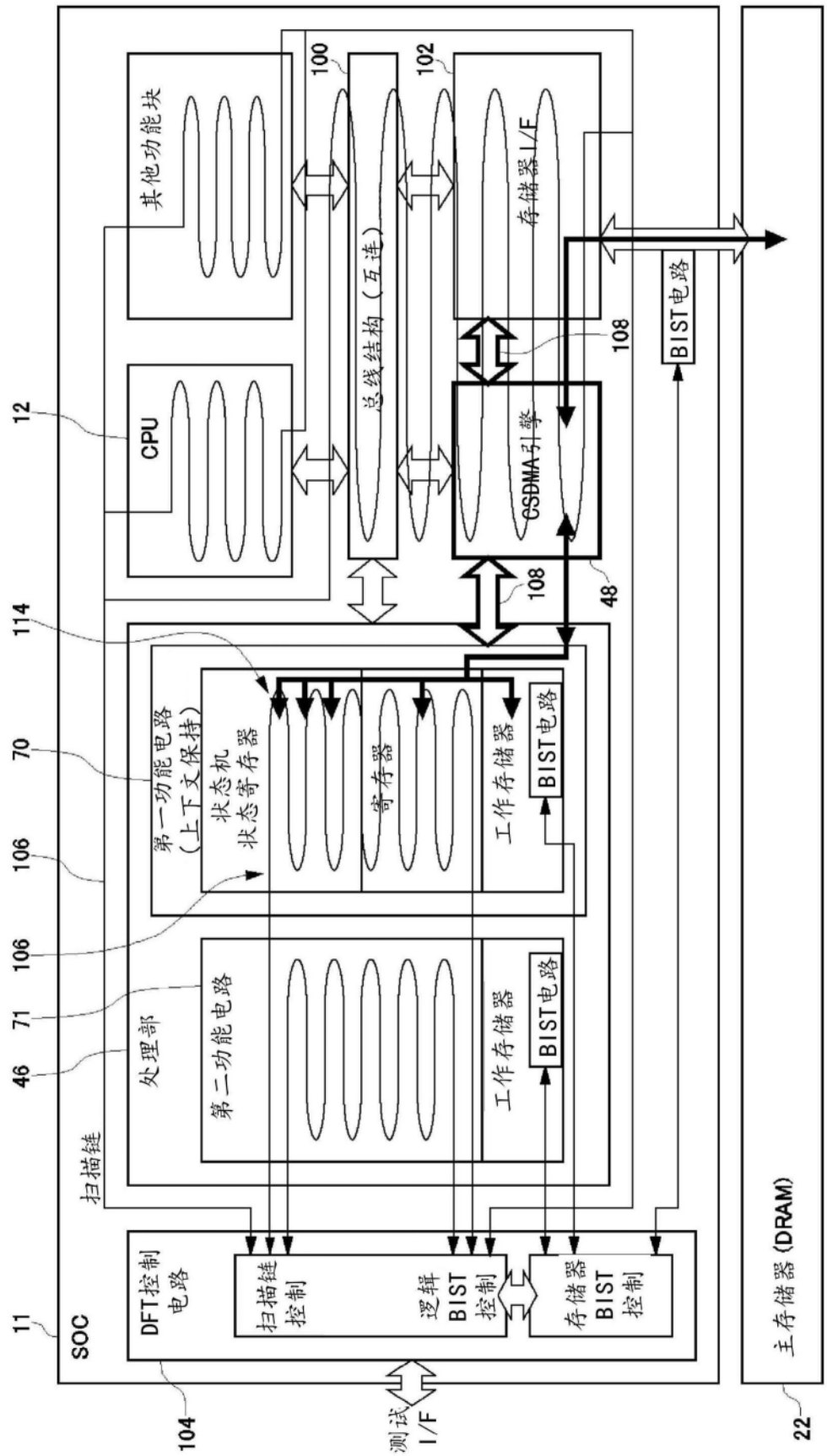

SOC
11
CPU
12
其他功能块
总线结构（互连）
100
102
存储器I/F
CSDMA引擎
108
108
48
BIST电路
114
第一功能电路
（上下文保持）
状态机
状态寄存器
70
寄存器
工作存储器
BIST电路
106
106
处理部
46
第二功能电路
71
工作存储器
BIST电路
扫描链
DFT控制
电路
104
扫描链
控制
逻辑
BIST
控制
存储器
BIST
控制
测试
I/F
主存储器（DRAM）
22

图7

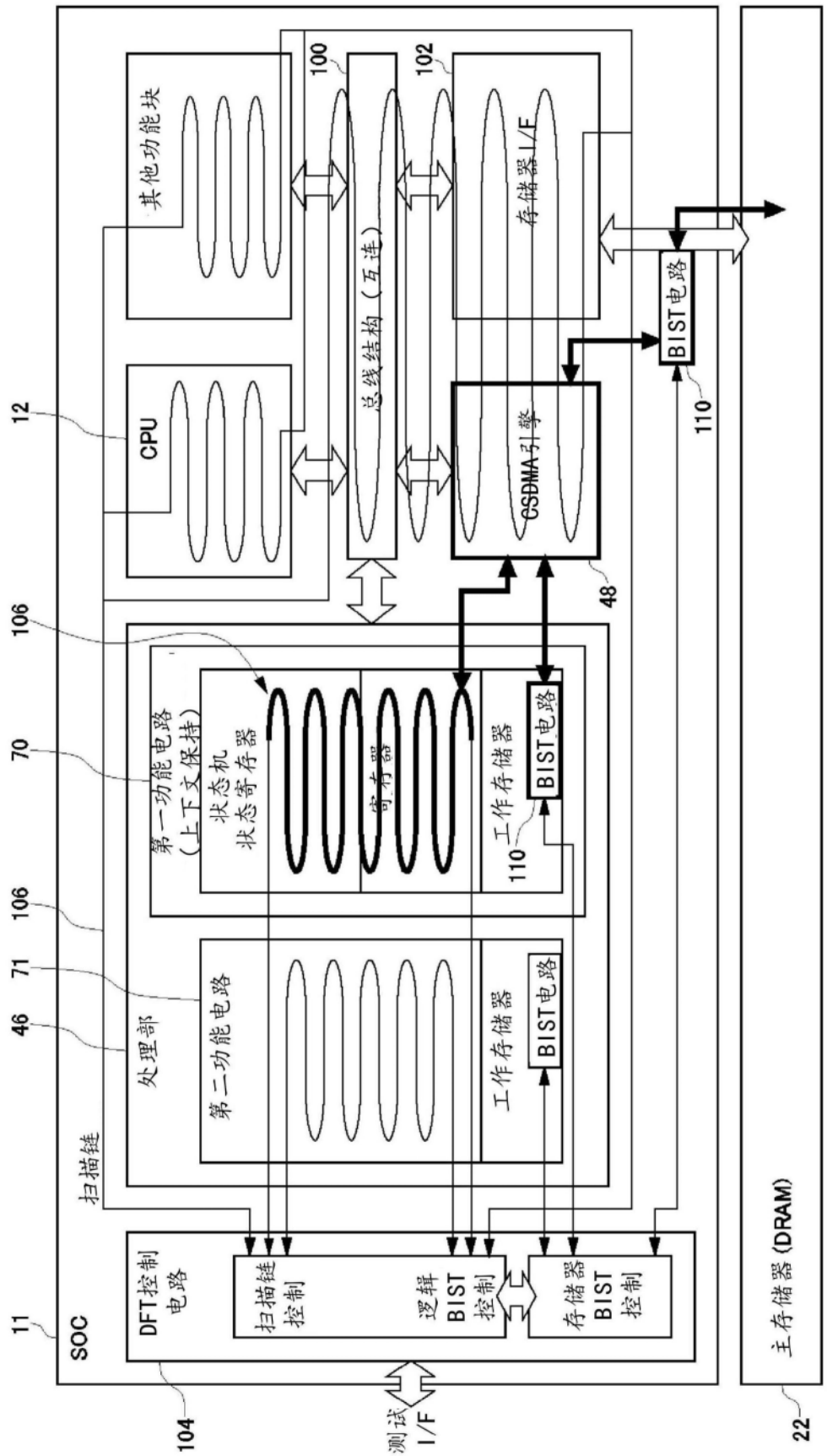

SOC
11
其他功能块
CPU
12
总线结构（互连）
100
102
存储器I/F
BIST电路
110
CSDMA引擎
48
106
第一功能电路
（上下文保持）
状态机
状态寄存器
寄存器
工作存储器
BIST电路
70
106
71
第二功能电路
工作存储器
BIST电路
处理部
46
扫描链
DFT控制
电路
104
扫描链
控制
逻辑
BIST
控制
存储器
BIST
控制
测试
I/F
主存储器（DRAM）
22

图8

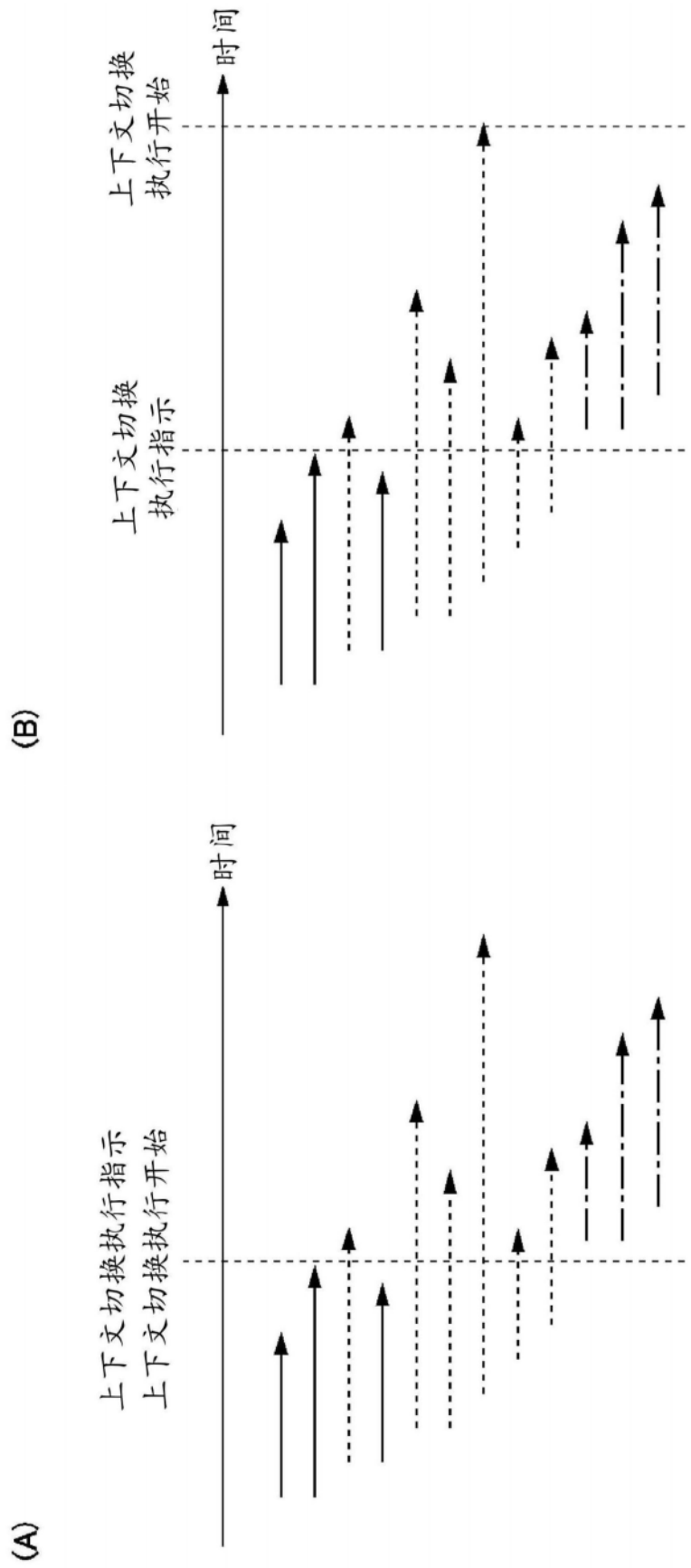


图9

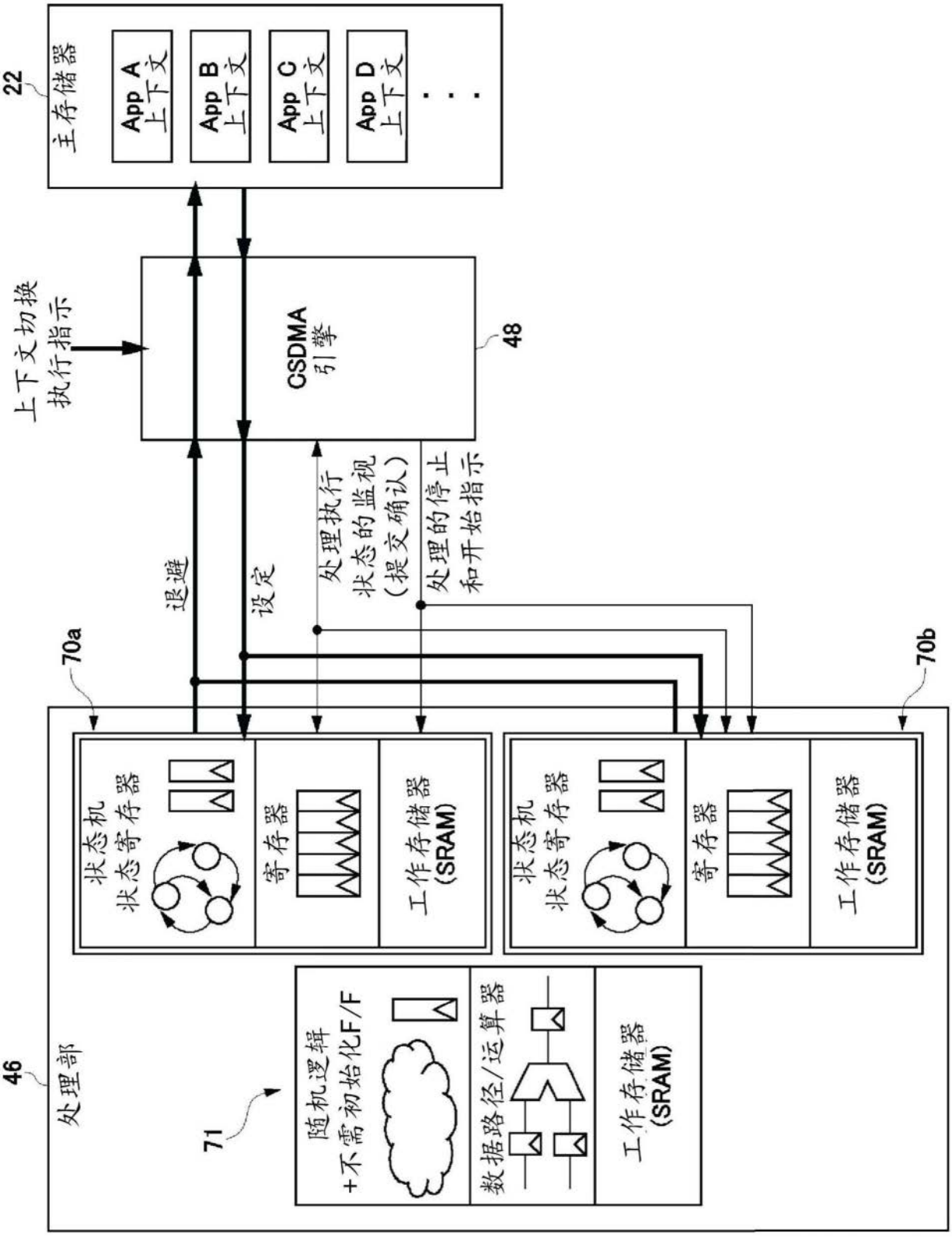

22
主存储器
App A 上下文
App B 上下文
App C 上下文
App D 上下文
上下文切换
执行指示
CSDMA 引擎
48
退避
设定
处理执行状态的监视（提交确认）
处理的停止和开始指示
70a
70b
状态机 状态寄存器
寄存器
工作存储器 (SRAM)
状态机 状态寄存器
寄存器
工作存储器 (SRAM)
46
处理部
71
随机逻辑 +不需初始化F/F
数据路径/运算器
工作存储器 (SRAM)

图10

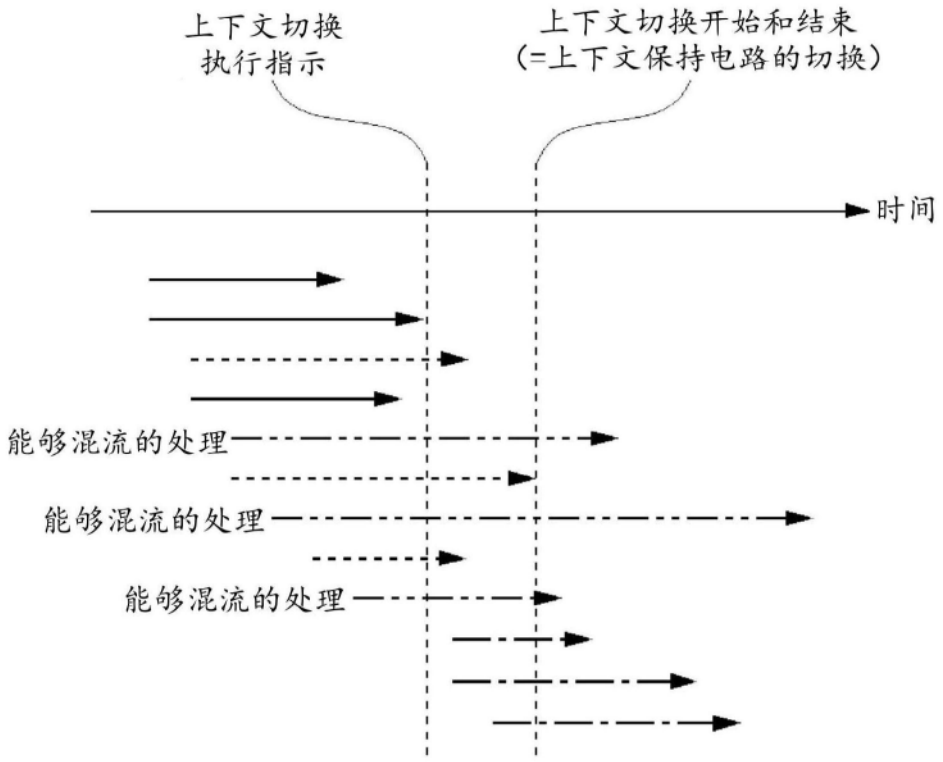

上下文切换
执行指示
上下文切换开始和结束
(=上下文保持电路的切换)
时间
能够混流的处理
能够混流的处理
能够混流的处理

图11

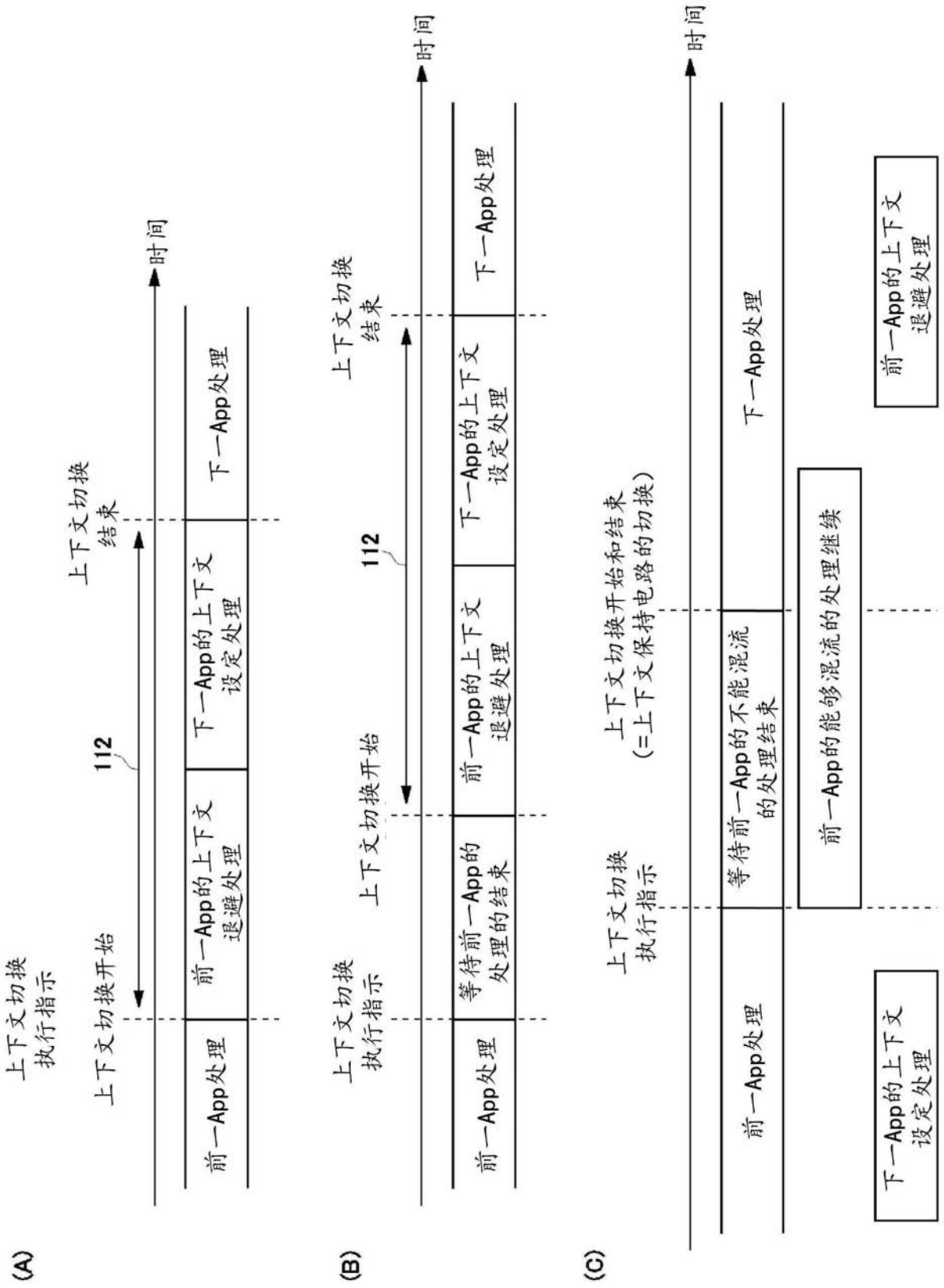

(A)
上下文切换
执行指示
上下文切换开始
112
上下文切换
结束
时间
前一App处理
前一App的上下文
退避处理
下一App的上下文
设定处理
下一App处理
(B)
上下文切换
执行指示
上下文切换开始
112
上下文切换
结束
时间
前一App处理
等待前一App的
处理的结束
前一App的上下文
退避处理
下一App的上下文
设定处理
下一App处理
(C)
上下文切换
执行指示
上下文切换开始和结束
(=上下文保持电路的切换)
时间
前一App处理
等待前一App的不能混流
的处理结束
前一App的能够混流的处理继续
下一App处理
下一App的上下文
设定处理
前一App的上下文
退避处理

图12

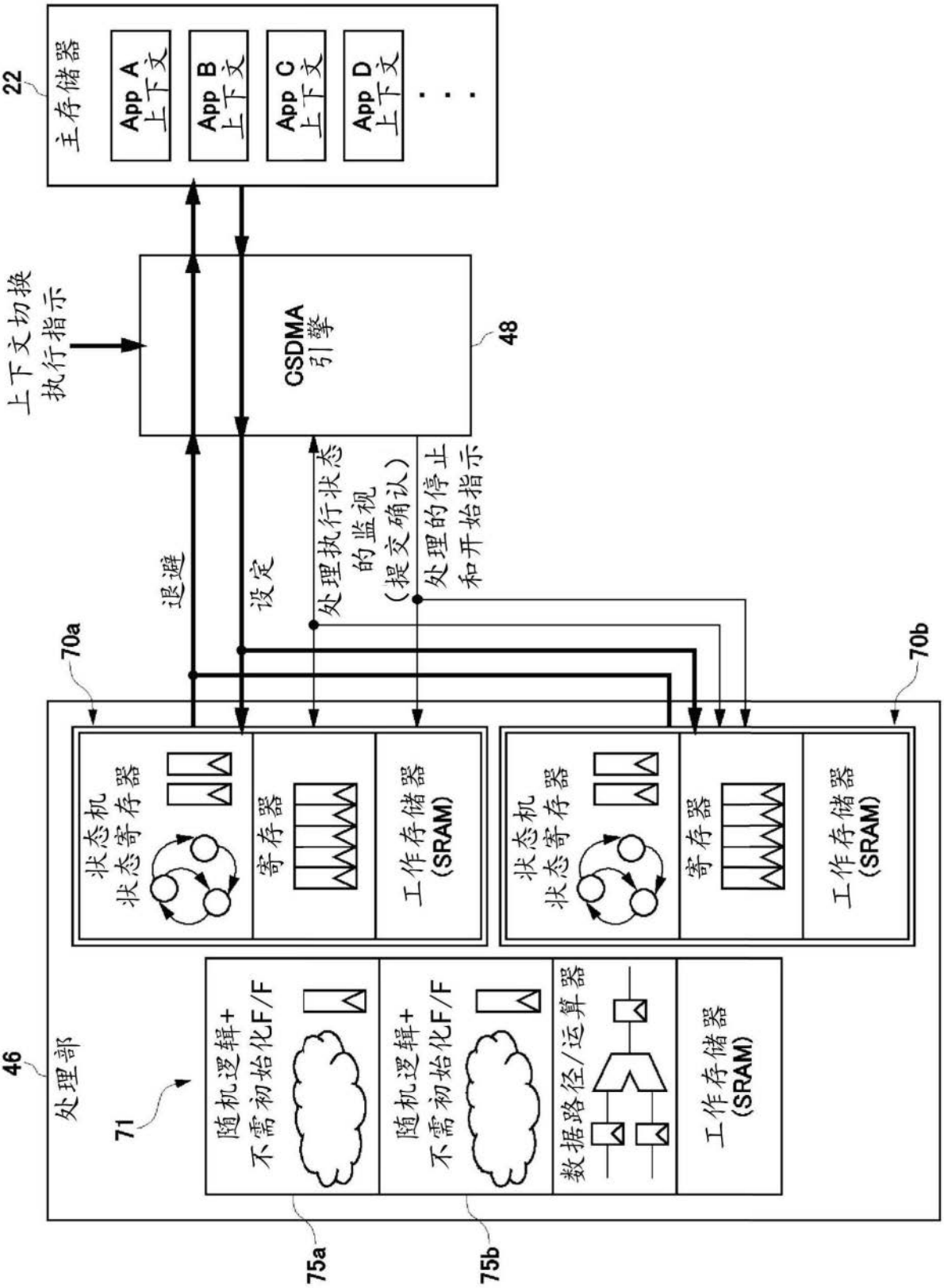

22
主存储器
App A 上下文
App B 上下文
App C 上下文
App D 上下文
上下文切换执行指示
CSDMA 引擎
48
退避
设定
处理执行状态的监视（提交确认）
处理的停止和开始指示
70a
70b
状态机 状态寄存器
寄存器
工作存储器 (SRAM)
46
处理部
71
随机逻辑+不需初始化F/F
数据路径/运算器
75a
75b

图13

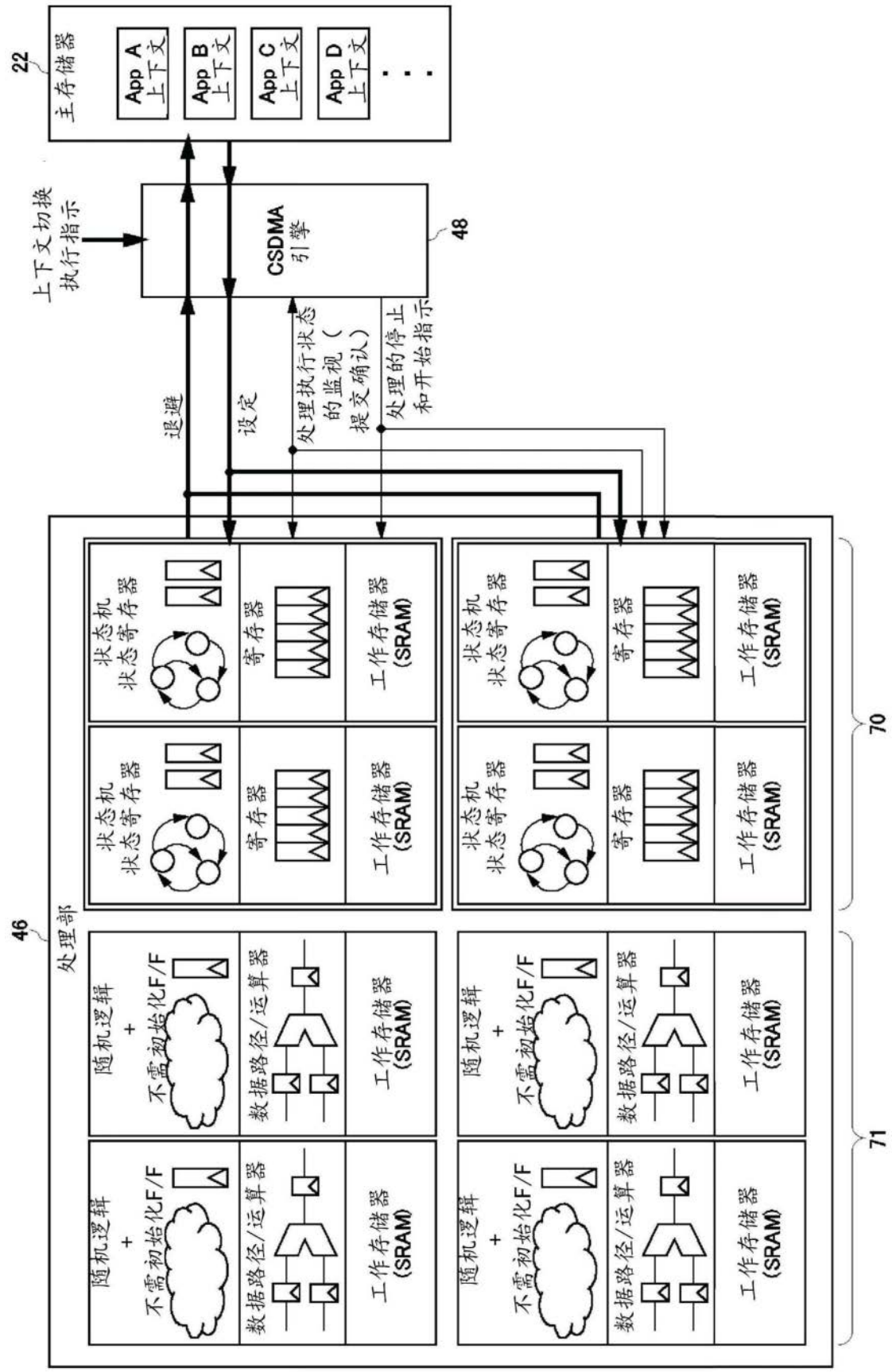

22
主存储器
App A 上下文
App B 上下文
App C 上下文
App D 上下文
上下文切换执行指示
CSDMA 引擎
48
退避
设定
处理执行状态的监视（提交确认）
处理的停止和开始指示
46
处理部
状态机 状态寄存器
寄存器
工作存储器 (SRAM)
随机逻辑 + 不需初始化F/F
数据路径/运算器
70
71

图14

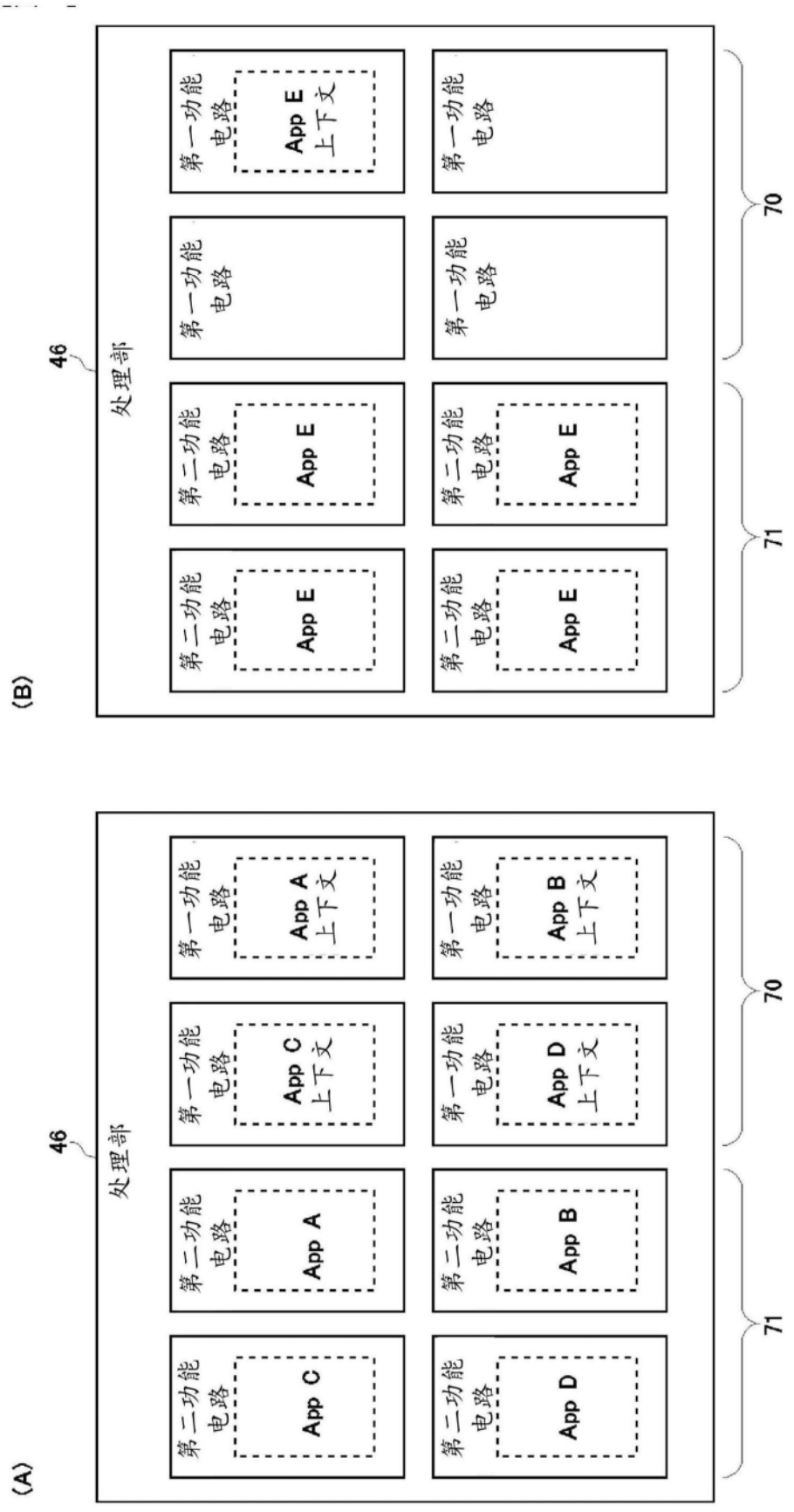

(A)
46
处理部
第一功能电路
App A 上下文
第一功能电路
App C 上下文
第一功能电路
App B 上下文
第一功能电路
App D 上下文
第二功能电路
App A
第二功能电路
App C
第二功能电路
App B
第二功能电路
App D
70
71
(B)
46
处理部
第一功能电路
App E 上下文
第一功能电路
第一功能电路
第一功能电路
第二功能电路
App E
第二功能电路
App E
第二功能电路
App E
第二功能电路
App E
70
71

图15