SYNOPSYS®
Accelerating Innovation

利用DDR控制器读取重排序缓冲器，将DRAM带宽提高十倍

February 2014

作者：
Marc Greenberg
Synopsys（新思科技）
DDR 控制器部产品营销总监

概述

DDR DRAM内存控制器要满足众多市场竞争的需求。一款出色的内存控制器必须能够增加存储器接口的带宽，满足CPU、图形处理、系统实时DRAM的延迟需求，同时符合存储总线和片上总线标准的规定。

读取重排序缓冲器（RRB）是DesignWare uMCTL和uMCTL2 DDR内存控制器IP产品上可用的一项硅验证的架构增强功能，是对DDR内存控制器架构的进一步完善。本白皮书将解释读取重排序缓冲器的概念，并对其如何提升存储带宽加以说明。此外，本文还总结了测试结果，展示了不同架构的DRAM控制器（根据该控制器是带RRB、带外部调度的RRB、或是带内容可寻址内存（CAM）调度的RRB的架构）可从相同输入数据流获得10%、66%或100%的截然不同的DRAM总线利用率。

DRAM控制器上事务重排序

每个存储子系统必须符合与之相连的片上总线的DDR DRAM总线标准和数据一致性需求。

将系统片上总线事务转换成存储事务，最简单的方法是使用一个协议控制器。该协议控制器将按从片上总线收到指令的顺序将指令转换成存储事务，同时遵循DRAM标准的规范。典型的协议控制器还将对输入的DRAM维护事务（如与存储读写操作相关的激活和预充电指令）进行调度。

很少有系统能够自然地生成高效的存储流量；例如，数据采集系统生成较长的顺序存取存储器。大多数系统都有CPU缓存填充、视频编/解码或网络分组数据流，这些数据流包含发射到存储器中随机位置的短事务。短事务指令对DRAM内部数据 bank结构的利用率较低，而且通常效率低下。甚至是在包含能够生成高效存储流量的存储请求者（主控）的系统（多请求者系统）中，来自多请求者的数据流也可能产生低效率的存储总线序列。

由于协议控制器以从片上总线收到指令的顺序来执行指令，按照高效DRAM执行的顺序为协议控制器提供指令是非常重要的。如果不能按此顺序，存储带宽可能受影响，因为协议控制器可能需要延迟一些事务来防止违反DRAM协议。

为处理系统内存储流量生成低效的问题，通常的方法是对存储流量重新排序，从而获得更高效的操作。可以在流量到达协议控制器之前通过一个存储调度器，或通过一个带重排序功能的内存控制器，对存储流量进行重排序。

重排序有多种策略可用；这些策略通常都会尽力避免DRAM协议限制所需的指令对指令延迟。有效的内存控制器不仅会防止不正确的指令序列，还会尽力实现正确的指令序列，如有利于高效DRAM执行的顺序页面点击。

高效的内存控制器将通过输入指令进行搜索，并设定一个可尽力防止DRAM标准中指定的指令对指令延迟的顺序。性能更佳的内存控制器（例如：带CAM设计的内存控制器）通常能对更大的输入指令集进行排序，从而提高发现更优化事务的概率。表1所示为可能影响符合DDR3-1600K（11-11-11）DRAM标准设备性能的一些主要的DRAM时间参数。

内存延迟	说明	时间（时钟）
RL	读取延迟（CAS）	11
tRCD	激活X bank以读取X bank	11
tRP	预充电X bank以激活X bank	11
tRC	激活X bank以激活X bank	39
tRAS (min)	激活X bank以预充电X bank	28
tRRD	激活X bank以激活Y bank	5
tCCD (BL)	读取（任意 bank）以读取（任意 bank）	4
tFAW	四个激活窗口——在该滚动窗口中激活的 bank不超过四个	24
tWTR	写入（任意 bank）以读取（任意 bank）延迟	6

表1：限制性能的DDR3-1600K（11-11-11）DDR DRAM主要时间参数

表1中的某些延迟不可避免。例如，内存控制器必须在X bank读取之前激活X bank中的一个页面，因此始终存在tRCD延迟。可以通过流量排序从而同时错过多个tRCD延迟，或通过向已满足tRCD延迟的其它 bank执行读写指令，将tRCD效应降至最低。

图1显示了一个简单的实例，在该实例中，表2所示的A、B、C三个指令按顺序到达，而由于片上总线排序原则、系统指定的指令优先级和重排序功能不可用（如在采用协议控制器的情况下）的原因，内存控制器不能对这三个指令重新排序。在本例中，在54个时钟周期内仅向控制器发射了三个读取指令。这表明在该指令序列期间，存储接口效率非常低。

指令	到达时间	特点
A	0	0 bank
B	1	0行
C	2	0 bank
		1行
		1 bank
		0行

表2：图1和图2到达内存控制器的指令顺序和类型

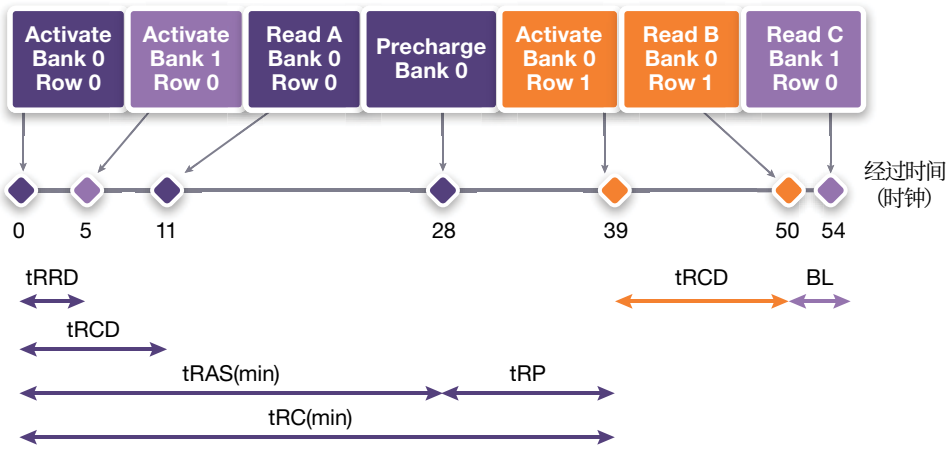

图1：显示表2所示之有序执行指令的时间轴以及防止过早执行指令的DRAM时间参数

图2显示了一个重排序案例，在该案例中，内存控制器收到了表2所示的A、B、C指令，且内存控制器可以对这些事务重新排序。图2中的控制器在50个时钟周期内发出了3个存储读取事务（与图1中的54个时钟周期不同），在该时段期间，存储带宽增长了8%。在读取指令C与B之间，有6个读取指令的空间。如果在读取指令C之后还有其它可重排序的指令且这些指令是

读入到 bank1行0或其它 bank的读取指令，则20、24、29、35、40和44等时钟周期全都可用于其它读取指令，从而大幅提升总线效率。对读取指令C重排序的好处在于，与指令C相关的数据返回系统的时间比图1所示早38个时钟周期。

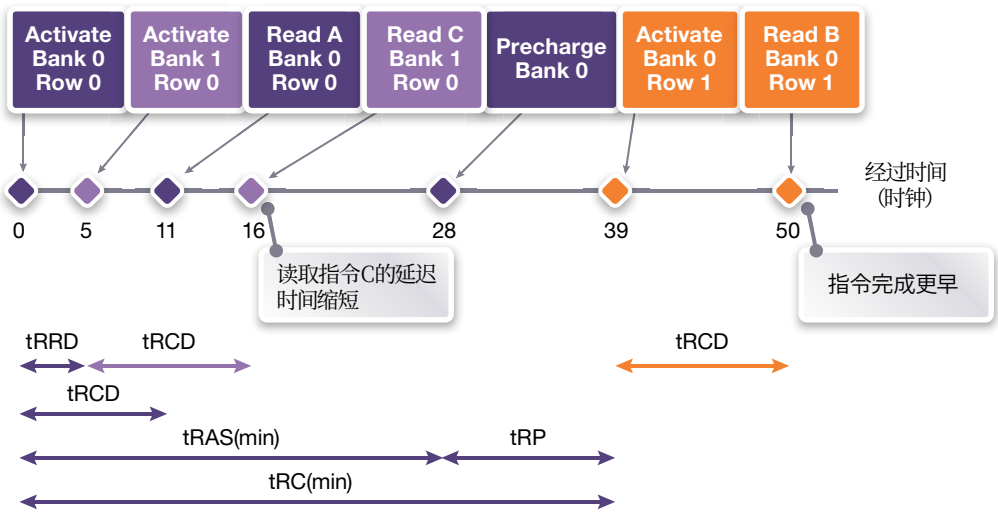

图2：显示表2所示的乱序执行指令的时间轴以及防止过早执行指令的DRAM时间参数

对事务重排序的限制

许多生成DRAM命令流的系统主控预计将以指令发出时的相同顺序从存储器收到读取数据和写入响应，而不管主控发出指令时对流量的排序是否高效。

系统中主控和片上总线的特点，决定了内存控制器中可能进行的排序数。像Synopsys主机接口（HIF）标准等一些片上总线标准，允许事务能够在片上总线接口上进行自由重排序。对于可接受无限制重排序的内存请求方，类似HIF等内存控制器接口是个理想的选择，而且无需读取重排序缓冲器。

大多数片上总线，包括符合ARM AXI标准的总线，都设立了一定的规则，这些规则不仅支持更简单的系统级一致性，而且在没有读取重排序缓冲器的情况下，也限制了内存控制器对DRAM事务重排序的能力。

片上总线通常需要在每个总线端口或总线段上进行有序的操作。可以在由不同总线主控生成流量时（如不同边带ID信号、线程ID或总线端口的信令所示）放松该限制条件，允许在总线端口内进行重排序。在多主控系统中，如果主控或总线有顺序执行指令的要求，有时可能会在不同总线主控间对事务重新排序，并显著提高带宽。

要求顺序执行指令的系统可能产生临时的情况，即只有一个端口和一个ID很活跃，而从其它端口没有可重排序的指令。有时，可能会有多个端口很活跃，但端口中所有队列头的事务可能执行效率很低。如果每个端口对 bank中不同行都请求一个指令，而这个 bank最近刚被激活且未能满足tRAS(min)时间参数，则可能发生上述情况。

能够将流量重排序至DRAM的内存控制器可能仍必须在这些实例中以不正确的顺序执行事务，除非系统带有一个读取重排序缓冲器。

图3显示了一个多主控系统，该系统要求每个主控的事务按顺序返回。在该例中，内存控制器可能在序列开始时选择A、D或G指令中的任何一个指令，而且必须遵照每个主控的指令序列执行。

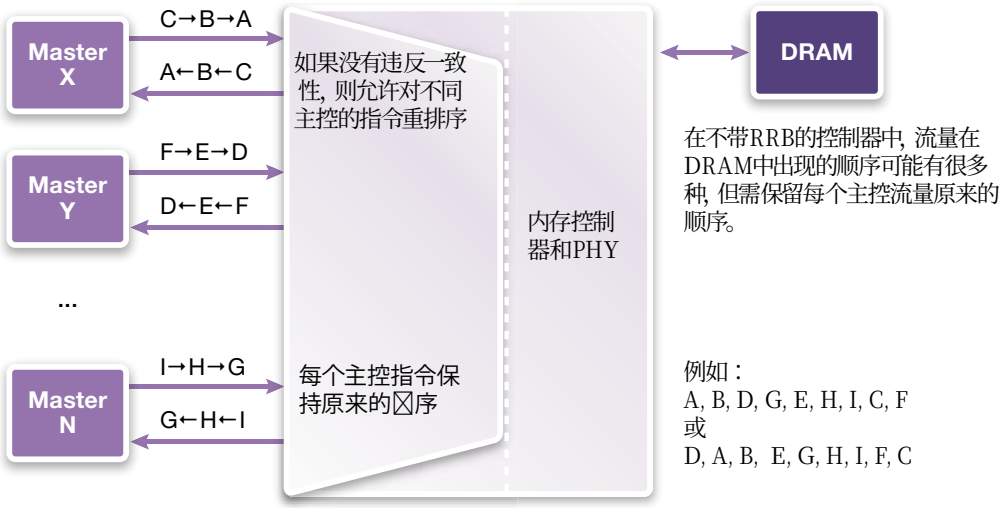

图3：在典型的多主控系统中，每个主控将以设定的顺序向DRAM控制器发出指令，并将预计以发出指令的顺序返回这些指令的响应

如图3所示，不带读取重排序缓冲器的控制器所存在的问题是，重排序的选项有限。如果首先向控制器发出指令A，然后开始执行，则第二个指令的选项为指令B、D和G。如果采用单面DDR3存储而流量在DRAM中随机分配，则第二个指令B、D和G中各个指令与指令A同 bank不同行的概率为1比8。此外，所有这三个指令与A指令同 bank不同行的概率为1比256。

尽管概率相对较低，但向同一个 bank中不同行发出背对背事务的代价很大，会造成tRC-tCCD延迟或上述DDR3-1600K内存延迟35个时钟周期。当在一个不带读取重排序缓冲器的3端口系统中以每tCCD一个事务的速率发出随机分配的存储读取事务时，由于所有输入的三个指令迫使存储总线上出现tRC延迟，系统的存储带宽损失统计为3.4%（即每1024个时钟会有35个时钟延迟）。

这种情况只考虑了向同一个 bank中不同行发出连续的背对背事务指令。由于tRC时间参数跨越多个指令时段，因此有机会遇到发生一个或多个“好”事务的序列，而且余下的事务将向tRC未到期的 bank中的不同行发出。

有种情况是，首先向内存控制器发出指令A，然后向不同 bank发出第二个三选一的指令（可能在发出指令A后立即执行）。该序列中的第三个指令有三种可能性，且第三个三选一指令中每个指令与指令A同 bank不同行的概率是1比8。由于执行了一个“好”指令，与“好-坏”序列相关的延迟是tRC-2tCCD或31个时钟周期。“好-坏”序列发生的概率为1/256，造成的损失率统计为31/1024或带宽的3%。

第四个事务指令会产生“好-好-坏”的序列和tRC-3tCCD的延迟，发生概率同样为1/256，造成的损失率统计为27/1024或带宽的2.6%。

在第四个事务指令之外，设计师们必须考虑tFAW效应。tFAW效应计算的复杂度超过了本文的范围，但根据所述情形，如果内存控制器不带读取重排序缓冲器，在DDR3-1600K DRAM 中读取事务传输长度随机的3端口存储系统中，“坏，好-坏，好-好-坏”事务序列相关的带宽损失至少为9%。

被迫接受向同 bank不同行发出的“坏”事务的概率，取决于当时需进行流量重排序的端口的数量。上述例子考虑了3端口上的随机流量，在每个决策点仅出现“坏”指令的概率为1/512（即1/8*1/8*1/8）。仅有2个端口出现“坏”指令的概率为1/64，仅有1个端口出现“坏”指令的概率为1/8。

为每个主控的读取数据路径添加一个读取重排序缓冲器，可以扩大指令的选择范围。图4显示了每个读取数据路径中带读取重排序缓冲器功能的系统实例，使内存控制器能够完全灵活地以最佳的顺序为DRAM进行事务重排序，而无需考虑每个主控的顺序。

按此前讨论的同一个例子，如图4所示，该内存控制器带读取重排序缓冲器功能，如果首先选择指令A，则可用的第二个指令包括B、C、D、E、F、G、H和I。第二个指令中每个指令与指令A同 bank不同行的概率约为1/8；但所有8个第二指令均为“坏”指令的概率仅为(1/8)8或1/16,277,216。因此，通常情况下，设计师预计向同 bank不同行连续发出指令的内存控制器的概率几乎为零。

图4：带读取重排序缓冲器的多主控系统能够以最优化的顺序在存储总线上执行事务，同时能够以与向内存控制器发出事务的顺序重新调整数据并向每个主控或ID发回响应

对事务排序限制的另一种看法，是考虑图3和图4所示案例中有多少种不同的事务顺序。图5显示了图3和图4 中所示内存控制器以指令A开始发出四个连续事务指令的不同顺序。这些序列可能在向另一个 bank发出长序列指令后在控制器中出现，也可以在内存总线离线进行刷新、初始化、培训或从低功耗模式退出时出现。图5左侧显示，在不带读取重排序缓冲器功能的3端口DDR控制器中，前4个指令仅有20种序列；但在添加读取重排序缓冲器之后，由于消除了同一个端口指令间的排序限制，将序列数量增至336种。

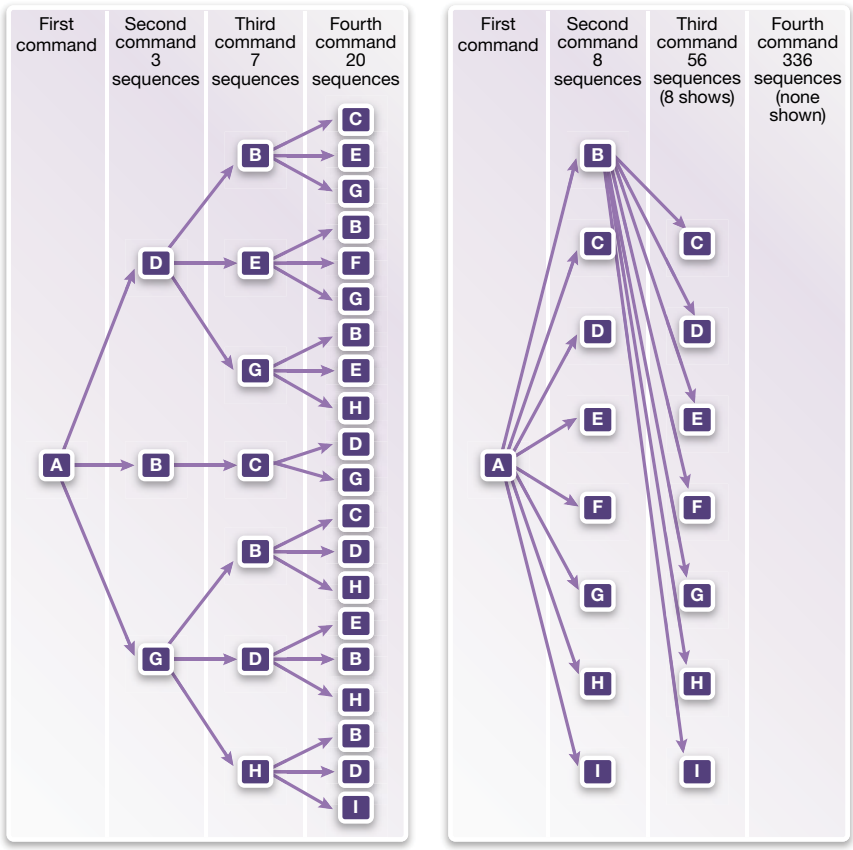

图5：不带RRB（左图）和带RRB（右图）的如图4所示控制器发出的前4个指令的序列数量

读取重排序缓冲器可以改善系统中的带宽，通过图6所示的外部调度单元，将来自系统中多个主控的流量融入内存控制器的单条指令流中。有些调度器要求，所有返回调度器的数据和响应以发出指令时的顺序返回。即使采用了高效的外部调度器，读取重排序缓冲器也可在漏预测或偏好与调度器初始设定不同的顺序的系统条件做出改变时，帮助实现调度器指令的重新排序。

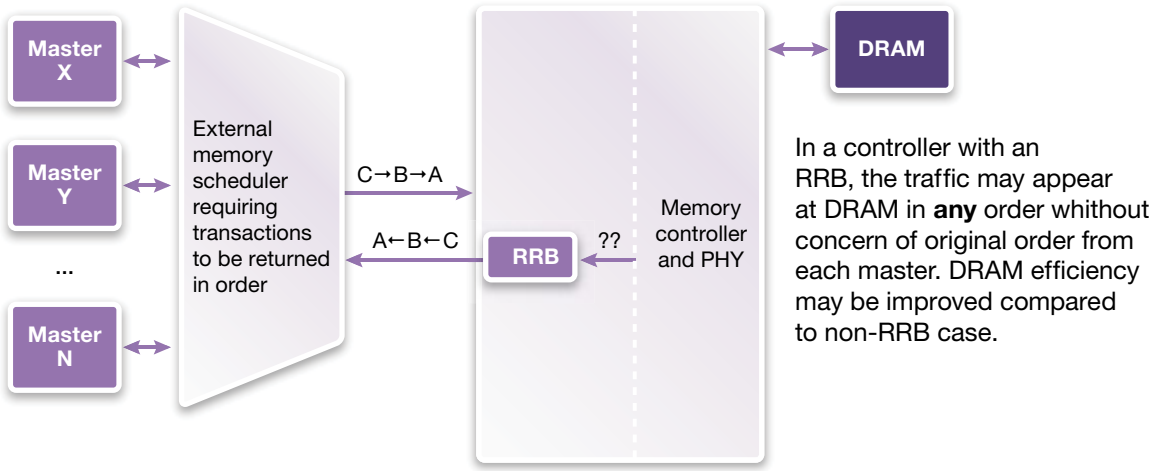

图6：带一个需要顺序响应的外部内存调度器的多主控系统。在本例中，读取重排序缓冲器可通过内存控制器提升带宽来实现重排序，同时仍需满足外部内存调度器的顺序要求

写入重排序和读/写重排序

内存控制器允许对写入事务进行重排序，而无需特别执行写入重排序缓冲器。通过将写入数据缓冲器作为RAM而非FIFO执行，系统可以对写入指令进行重排序，并根据一致性原则，从写入缓冲器中乱序提取写入数据。

当总线类型包含一个写入响应渠道（如：ARM AXI总线）且要求按照收到写入指令的顺序向主控发出写入响应，则写入响应仍需保存在一个写入响应重排序缓冲器中。

通常情况下，读取和写入还可进行相应的重排序，以改善内存带宽并防止较长的读写（tWTR）总线周转时间。读/写重排序无需特定的缓冲器，但必须遵循系统一致性原则。

测试配置

为显示读取重排序缓冲器的好处，现对三个不同的缓存控制器配置进行对比：

- 严格在线的控制器（Synopsys uPCTL Controller，图7）
- 带调度器和读取重排序缓冲器的控制器，该读取重排序缓冲器可以在调度阶段对指令进行重排序（Synopsys uMCTL Controller，图8）
- 带调度器、读取重排序缓冲器和CAM存储调度器的控制器，该CAM内存调度器可以在初始调度阶段后对DRAM的指令进行重排序（Synopsys uMCTL2 Controller，图9）

在所有情况下，均采用了单输入指令流（单端口），所有从测试平台到内存控制器的事务都将作为AXI ID相同的AXI事务发出，以限制在片上总线接口上允许的重排序。需考虑包含所有读取指令的数据流，因为在多数情况下，可以对写入重排序，而无需使用写入重排序缓冲器。存储接口为32比特，采用了带11-11-11配时的DDR3设备模型。

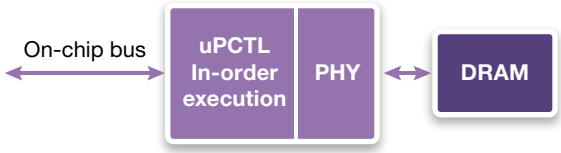

图7：Synopsys DesignWare uPCTL Controller的框图，用以表示不带读取重排序缓冲器的在线控制器

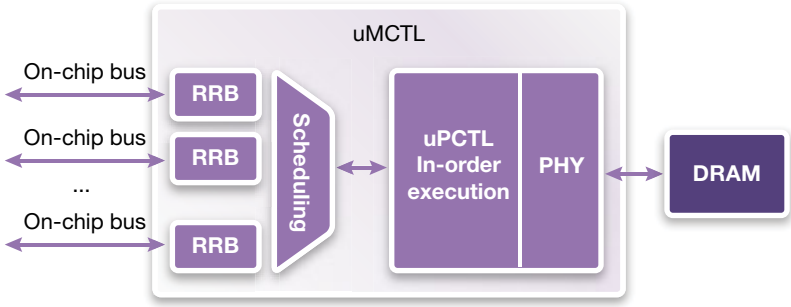

图8： Synopsys DesignWare uMCTL Controller框图，用以表示带读取重排序缓冲器的在线控制器。对于测试而言，仅使用了一个端口，且事务必须按顺序返回片上总线，所有事务均带有相同的AXI读取ID

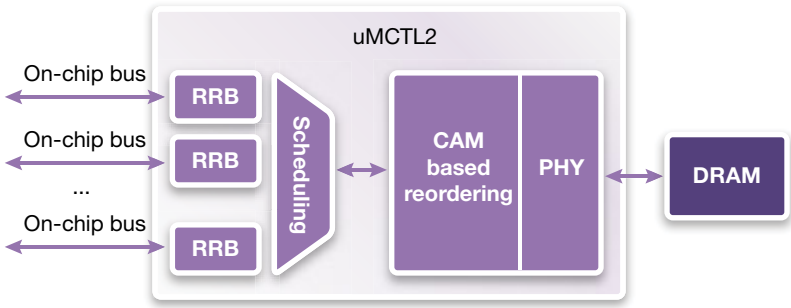

图9：Synopsys DesignWare uMCTL2 Controller框图，用以表示带读取重排序缓冲器的重排序控制器。对于测试而言，仅使用了一个端口，且事务必须按顺序返回片上总线，所有事务均带有相同的AXI读取ID

测试结果

将多个输入数据流进行对比，所有这些数据流均作为不可重排序的流量，从带有相同ID的主控器发送到控制器上。如果在接口上发生重排序，则通过RRB以正确的顺序将数据返回到接口上。

第一个测试是将一个事务序列发射到控制器上，控制器已经包含最优化的DRAM流量（发送至同行同 bank的事务长序列），以表明读取重排序缓冲器不会造成DRAM带宽的损失。第二个控制测试是将流量发射至不可能优化排序的控制器（突发长度增加到同 bank的行存取），以表明读取重排序缓冲器不会违反存储排序规则。两项测试的结果如图10所示，结果表明读取重排序缓冲器不会在事务达到最优化顺序时造成存储带宽的损失。

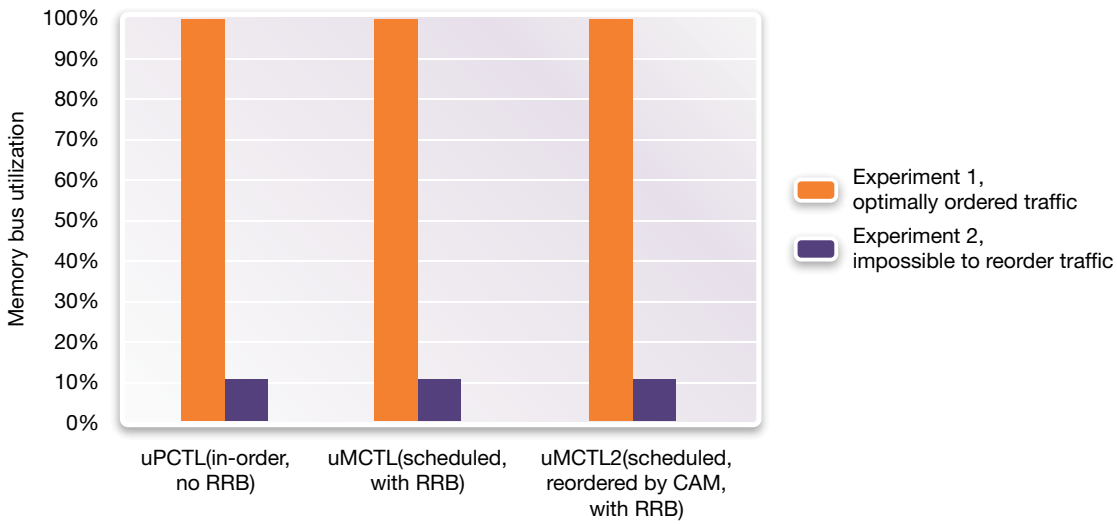

图10：控制测试表明，RRB不会改善无法重排序的流量，也不会破坏已经优化排序的流量

For a final test, experiment 3, constructed worst-case traffic for a system without a read reorder buffer was used. In the constructed worst-case traffic, transactions were issued that consisted of alternating back-to- back 32-byte transactions to different rows in the same bank, as shown in Table 3. The transactions were issued to the memory controllers from the on-chip bus interface in the order A, B, C, D, E, F, G, H, … and all with the same ID and all the minimum size of transaction (1 burst).

最后一个测试中，为不带读取重排序缓冲器的系统构建了一个最差情况的流量。在这个最差流量中，发出的事务包含向同bank不同行交替发出背对背32字节的事务指令（如表3所示）。这些事务以A、B、C、D、E、F、G、H……的顺序从片上及总线接口向内存控制器发出事务，所有事务都具有相同的ID，且保持最小流量（1个突发）

Transaction reference	Transaction ID (AXI ARID parameter)	Bank	Row	Column
A	0	0	0	0
B	0	0	1	0
C	0	0	0	80
D	0	0	1	80
E	0	1	0	0
F	0	1	1	0
G	0	1	0	80
H	0	1	1	80
…	0	…	…	…

表3：测试3的事务指令序列

正如预期，不带读取重排序缓冲器的控制器也以A、B、C、D、E、F、G、H……的顺序向DRAM发出事务指令。这种流量的问题在于，每个新事务都与前一个事务指令存在 bank冲突；而且当这些事务按顺序执行时，在测试中使用的DRAM协议将需要间隔39个时钟周期的读取指令，允许DRAM的行周期（tRC）时间参数。在测试3中，不带读取重排序缓冲器的内存控制器的存储总线利用率仅为10.1%。

在测试3中，在控制器中添加读取重排序缓冲器和调度器（uMCTL）后，大幅改善了流量，并以 bank循环的顺序调度事务，从而将总线周期的存储利用率提高到66.4%。调度器单元在所有8个 bank中周期循环，但已经开始达到tFAW存储时间参数（四个存取窗口）所设定的时间限制，这个参数限制了新 bank激活的速率。

在测试3 中，带读取重排序缓冲器的控制器和CAM存储调度单元（Synopsys uMCTL2 Controller）通过以A、C、E、G……B、D、F、H……顺序执行内存接口上的事务，将流量带宽增加了一倍，从而在循环到一个无tRC时间限制的新 bank之前，在DRAM上创建了2个背对背页面点击。在将读取数据返回到片上总线之前，读取重排序缓冲器按照片上总线协议的要求，将数据重新返回到A、B、C、D、E、F、G、H……的顺序。

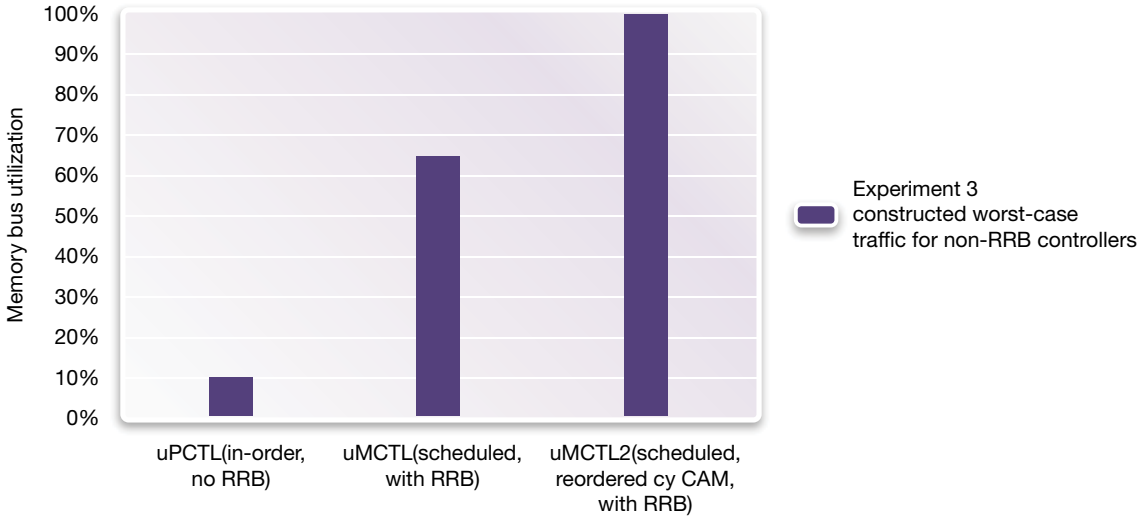

图11：通过添加读取重排序缓冲器和CAM，提高最差存储流量的存储带宽

潜在问题

在内存控制器上添加读取重排序缓冲器的潜在问题是造成更大的延迟。这种延迟可分为两部分：固定延迟和基于流量的可变延迟。

对于Synopsys uMCTL2 Controller而言，读取重排序缓冲器作为SRAM执行时的固定延迟为1个时钟周期，而且通常情况下，这种固定延迟的增加被视为可以接受的，因为可以大幅提高带宽。

由于引入了读取重排序缓冲器，大多数系统的平均延迟将减少。指令到达速率、系统中指令数量与稳定系统中的指令延迟之间的关系可通过利特尔法则L=λW来表示，其中L表示系统中平均指令数量，λ表示指令平均到达速率（等于指令平均执行速率，否则队列将无限延长），W表示指令在系统中的平均时间。在固定最大队列长度为L的内存控制器中，预计一个指令在系统中的平均时长W与指令执行速率λ（带宽）成反比。

读取重排序缓冲器的可变延迟很大程度上取决于流量；尽管平均指令延迟可能减少，但与不带读取重排序缓冲器的情况相比，一些指令的延迟有所增加。如果对某些特定事务的延迟存在疑问，这些事务可以采用更高的服务质量（QoS）或更高的优先级，以表明这些指令比系统中的其它指令优先执行。用户无需担心读取重排序缓冲器会造成拥堵问题，因为Synopsys的uMCTL2 Controller可以动态地将主控分配到RRB的分区中，从而防止一个主控流量会造成另一个主控不必要的流量延迟。

结论

读取重排序缓冲器对于内存控制器是一个很有用的补充，能够实现重排序，甚至在片上总线需要按原来顺序返回事务时也可以实现重排序，从而将特定类型流量的带宽提高10倍。

鸣谢：感谢Dara Hurley和Synopsys DDR Controller工程师队伍创建了DDR Controller产品并运行展示了测试结果的数据。