doi:10.3772/j.issn.1002-0470.2017.05.001

稀疏队列中的动态发射机制及电路实现①

刘 臻②[*,**,***] 王 剑[*,***] 赵鹏飞[****] 丁健平[****]

([*]计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

([****]龙芯中科技术有限公司 北京 100095)

摘 要 针对多运算部件处理器中非流水多拍指令堵塞非相关指令的问题,提出了一种动态发射机制,该机制可以在发射当拍根据空闲运算部件数量同时选中并发射多条指令,不必提前为指令分配运算部件。动态发射稀疏队列基于一种快速的 N 选 M 电路,利用电流的大小来表征指令在队列内驻留时间的长短,通过灵敏电流放大器实现快速的筛选,最后经过 RS 触发器调整波形,利用 NMOS 放电网络得到指令位置的掩码。动态发射队列解决了运算部件冲突问题,提高了每时钟周期执行指令数(IPC),最大程度发挥全局队列的效能,其中的调度电路使用 SMIC 40nm 工艺实现,通过 Hspice 仿真验证,该电路工作频率可达 8GHz。

关键词 发射队列,动态发射,稀疏发射队列,N 选 M,灵敏放大器

0 引 言

超标量微处理器[1]采用乱序发射技术来获得更高的指令级并行性(instruction level parallelism, ILP),这一技术的核心部件为发射队列(issue queue)。队列监听相关指令的状态同时调度指令的发射,提高运算部件的使用效率。增加队列的项数可以显著提高性能,如从 16 项指令发射队列扩展至到 64 项,性能可以提升 39%[2],然而,早期的紧实队列(compacting issue queue)每周期指令移位的特点[3],导致其功耗居高不下,制约队列项数的提升。为了解决这一问题,各芯片厂商开始使用稀疏队列(non-compacting issue queue),即取消了指令发射后的移位操作,增加专用的模块来维护指令顺序信息。这其中不乏一些经典的例子,如矩阵调度器(matrix scheduler),相关矩阵(dependence matrices)[4],可配置 FIFO[5]等。这些策略都是通过添加指针或者计数器,在控制功耗的前提下期望达到紧实队列的调度效率。

提升性能的另一方式是增加运算部件数量,工艺的进步使得单个处理器核内可以集成多个算术逻辑单元(arithmetic logic unit, ALU)[6],然而上述的调度逻辑单周期内只能计算出满足发射条件且驻留时间最久的一条指令,却无法得到驻留时间第二久、第三久的指令,要查询这些指令,需要花费额外的时间代价或者面积成本,对照运算部件数量复制多份相同的调度逻辑求出各自驻留时间最久的指令,分发给所对应的运算部件。这种方式需要提前为指令分配指定的运算部件,由于不同指令的执行周期不一样,会导致多拍非流水指令占用执行部件堵塞后续非相关指令,也被称为“运算部件冲突”。

① 国家“核高基”科技重大专项课题(2014ZX01030101),国家自然科学基金(61432016)和 863 计划(2013AA014301)资助项目。

② 男,1988 年生,博士生;研究方向:微处理器体系结构,高性能集成电路;联系人,E-mail: liuzhen@ict.ac.cn

(收稿日期:2016-12-16)

基于以上考虑,本文提出了一种动态发射机制,即根据当前空闲运算部件的数量,动态选择与发射指令。指令进入队列等待发射的过程中,可以被分发给任意运算部件。该机制不存在“运算部件冲突”,在队列中尚有操作数已写回的指令等待时,所有运算部件都会处于满载状态,由此提高处理器的每时钟周期执行指令数(Instruction Per Clock,IPC)。队列中的调度电路利用灵敏电流放大器判断指令的驻留时间,可配的参考电流源控制发射数量,单周期可同时选中多条指令,逻辑数量大幅降低,有效地减小队列面积。

1　紧实队列与稀疏队列

紧实队列中的指令一旦被发射出去,其后的指令需要依次向前移位以填补空洞,而新的指令则会进入队尾,指令之间不存在空项并且维持了先后顺序。紧实队列的调度仅需要一套基于位置查找的逻辑,其过程如下:为队列的每一项分配一位标志寄存器,空项初始化为“0”。每条指令时时监听结果总线,若与之相关的指令已写回,则置标志位为 1。队首至队尾所有标志寄存器中首个“1”的位置,就是当前源操作数已写回且驻留时间最长的指令所在的位置。图 1(a)显示了一个 4 项紧实队列的发射流程,A,B,C,D 是顺序进入队列的四条指令,且 B 刚刚被发射出去,指令 C 和 D 依次向下移位,新的指令 E 会被置入队列顶部。下一个周期指令 C 和 E 的操作数都已经准备就绪,采用基于位置的调度逻辑,标志寄存器中首个 1 对应于指令 C,则紧实队列会发射指令 C。

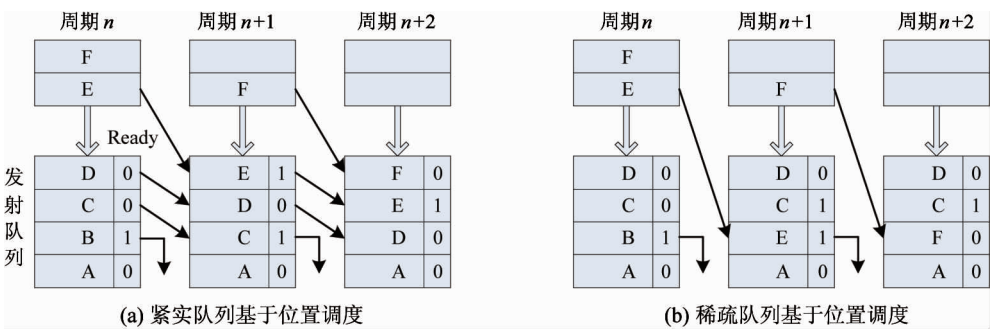

图 1　紧实队列与稀疏队列基于位置调度时的发射情况

实现这一调度策略的前导零电路,结构如图 2 所示。工作流程如下:第一步,待测信号沿树向下传播,每一个仲裁单元若是发现待测信号中有“1”存在,就将标志位(ANYREQ)置起,与此同时,其父节点的待测信号即被置起成“1”,信号一直从叶节点传播至根节点;第二步,仲裁单元依据使能信号来判断当前序列中首“1”是否有效,若有效则置起相应的授权信号,与此同时,其子节点的使能信号即被置起成“1”,信号一直从根节点传播至叶节点。

紧实队列的调度逻辑简单,但过高的功耗制约了频率的提升。仔细分析其结构特点可知,在每个时钟周期内,指令发射之后的移位刷新触发器的内容,但记录的有效信息并没有变化。更糟糕的是,队列维护了指令的顺序,所以,靠近队首的指令总是优先被发射出去,这就导致了每次移位的指令数量非常庞大。

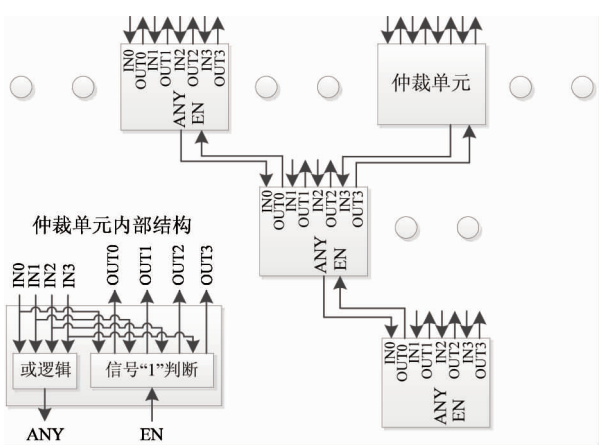

图 2　前导零电路结构

稀疏队列则取消了指令位置先后顺序的约束。指令进入队列时,可被置入任意位置,一般从队列两端依次查找空项进行分配[7]。队列内指令的分布分散,需要额外的模块来记录顺序信息,如果还是使用基于位置的调度逻辑会造成指令异常等待。图 1(b)是稀疏队列采用基于位置的调度策略时的情况。同样是指令 B 被发射出去,指令 E 会被置入刚刚发射的 B 的位置。在下一个周期,指令 C 和 E 的操作数都已经准备就绪,采用基于位置的调度逻辑,队列会发射指令 E。可预见的是,具有较高优先级的指令 C 和 D 会在队列中驻留很长的一段时间,这对于指令级并行性(IPC)的影响是非常大的。

因此,稀疏队列往往需要更加复杂的调度方法。一种解决的方案是添加记录优先级的标志位,简称 SB(sorting bit)[8]。所有未写回的指令都会存储在重排序缓存(reorder buffer,ROB)内,ROB 记录了指令正确的的执行顺序,可以将指令连同其在 ROB 内的位置存储在标志位(SB)内一起写入队列,若尾指针又回到了队列顶部,则将新指令的 SB 置为“1”,如上例中,指令 E 被分配进队列后,其 SB 位必定小于将指令 A,C 和 D。这样,只需比较 SB 值的大小,就可以找出驻留时间较长的指令。图 3 详细记录了这一过程。

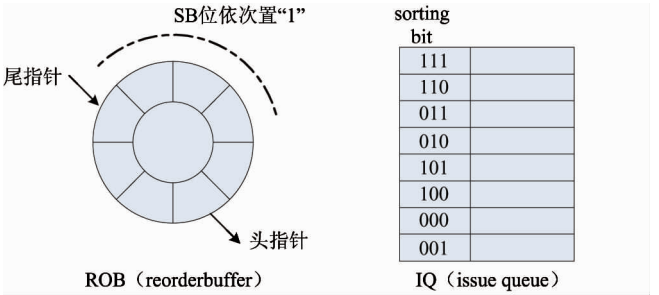

图 3　增加 sorting bit 辅助的调度策略

一旦被分配了 SB 序列,指令驻留时间的长短就可以通过 SB 序列值的大小来确定。选择最早进入的指令转化成寻找序列中值最大的数,只需从权重较高的一端依次比较即可。流程如下:先将最高位的 SB 或在一起,如果结果为 1,则最高位 SB 为 0 的指令将被排除;若结果为 0,保留所有指令;重复这一过程,直到 SB 的最低位,最终保留的指令即为在队列内驻留时间最长的指令。

图 4 所示为一个 16 项队列的双发射调度电路,每一项的 SB 共 4 位: S1,S2,S3,S4。这种电路的优势在于可以实现多个端口的并行发射,理论上可以同时发射任意 2^n 条指令。例如,如果发射端口增加至 4 个,则可以抛弃最低位 S1,仅比较前三位即可。电路的缺点是,基于多比特数据相或多电路需要从

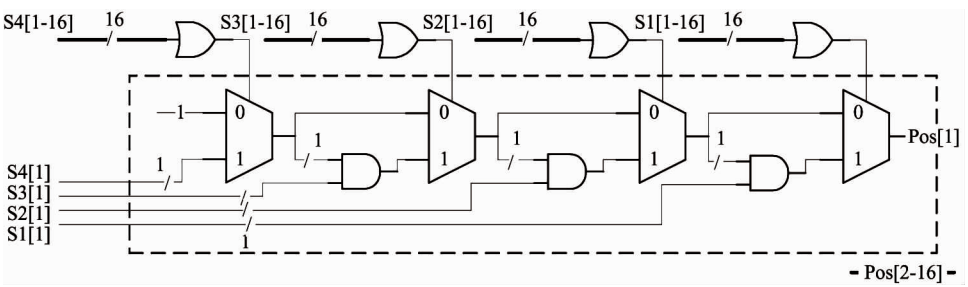

图 4　16 项队列比较标志位(SB)的电路结构

SB 高位至低位串行运算,其运算复杂度为线性的 $O(N)$,对于一个 128 项的队列,SB 的宽度为 8 位,单周期内完成 8 次判断的难度是非常大的。

Intel 的矩阵调度器(matrix scheduler)增加了年龄触发器(age cell)[2],同样用来记录指令在队列内的驻留时间,但是较大的数据并不代表更高的优先级。指令被分配至队列时,获取一串“年龄”数据,将已在队列内的指令对应位置记录为“1”,空项对应的位置记录为“0”;指令被发射出去之后,将所有标记该指令的年龄触发器重置为“0”。年龄触发器的意义在于,若其存储的值为 1,则代表该位置上的指令比当前指令驻留时间更长(指令进入时,该位置上的指令已存在),反之,则表示该位置上的指令比当前指令驻留时间更短。

队列调度时,操作数已写回的指令生成发射信号,进入 matrix scheduler 内传播,发射信号的传播路径有两条:一条传向调度器终点;一条传向其他指令。传向其他指令的发射信号,会激活年龄值为“1”的触发器,生成冲突信号,取消指令发射;传向调度器终点的发射信号,同样会被其他指令的冲突信号所取消;只有没有被取消的发射信号,可以传播到调度器终点,该指令即为当前驻留时间最长的指令。图 5 显示了这一发射过程。

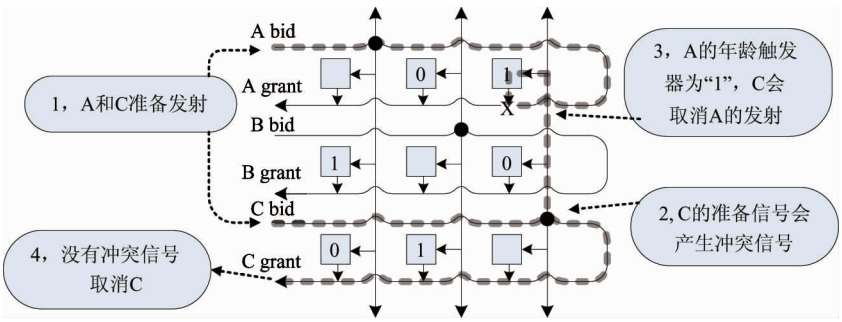

图 5　matrix scheduler 选择发射指令的过程

这种电路结构充分利用了“线与”逻辑,可以实现非常短的延时,但是矩阵的规模随队列项数成平方增长关系,当队列规模较大时,传播信号有可能因为负载太大导致变形和失真,如果增加一定数量的缓冲器来恢复信号,又增大了电路的延时。

2　动态发射机制

前文所述的紧实队列和稀疏队列,无论依据何种调度策略,都是基于 N 选 1 电路,当存在多个运算部件时,不可避免地会发生“运算部件冲突”,以下将详述该冲突发生的背景。

发射队列有独立队列、分组队列和全局队列 3 种常见的组织方式[9]。独立队列就是每一个功能部件独占一个队列,分组队列是把功能部件分组,同一个组的功能部件共用一个队列,例如定点部件和访存部件共用一个队列,多个浮点部件共用一个队列。全局队列就是所有功能部件共同使用一个队列。图 6 给出了独立队列、分组队列和全局队列的结构。

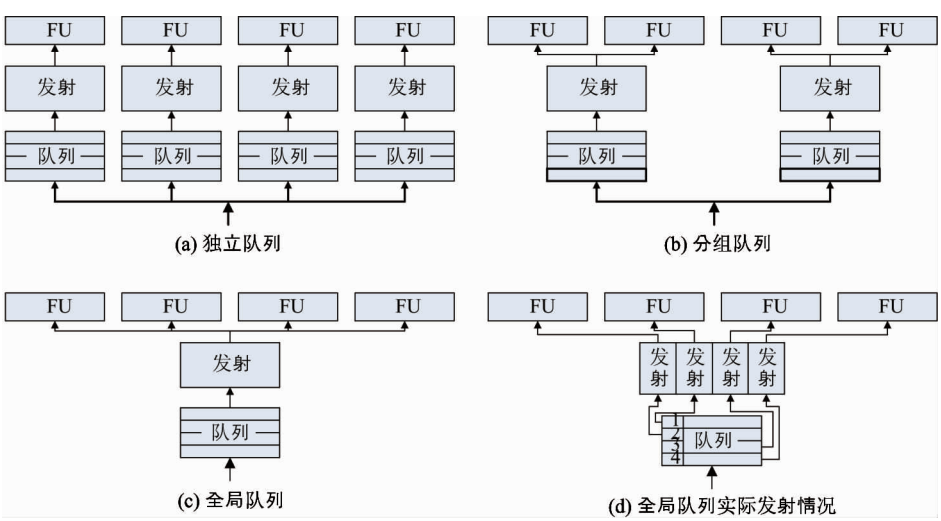

图 6　独立队列、分组队列和全局队列的结构

独立队列的结构特点是设计简单,每个队列只有一个写入端口和一个读出端口。但是这种结构的队列利用率较低,指令繁忙的队列可能已经出现堵塞,而另外一些队列还在空闲状态。分组队列和全局队列都采用了功能部件共享队列的结构,只不过共享程度不同而已。这就需要队列提供更多的读出和写入端口。

全局队列提高了队列的使用效率,但是发射过程中,仍然会出现类似独立队列中的“假堵塞”现象。考虑第 3 节中所提到的调度策略,在单周期内只能计算出满足发射条件且驻留时间最久的一条指令,这意味了队列只能提供一个读出端口[10]。为了实现多个读端口,需要在队列内部对指令进行分组,在不同的组内分别使用相同的调度算法,每组提供一个读出端口。如图 6(d)所示。

队列内部按照运算部件分组,导致全局队列退化至独立队列。考虑图 7(a)中的情况:指令 A、B、C、D 依次进入两个功能单元(function unit,FU)的全局队列,其中,A 是一条多拍非流水指令,B、C、D 为单拍指令,指令 C 等待指令 A 的结果,指令 D 等待指令 C 的结果,B 和 A、C、D 均不相关。在进入队列时,A、B、C、D 根据 FU 数量被分成两组,A 和 B 一组被指定送往 FU1,C 和 D 一组被指定送往 FU2,当 A 在执行周期内,C 和 D 等待源操作数写回,因此不能发射,指令 B 因为要等待 FU1 空闲,也不能发射,这就导致 FU2 会闲置多个周期。指令 B 因为等待指令 A 释放运算部件而不能发射的情况,就称为“运算部件冲突”。

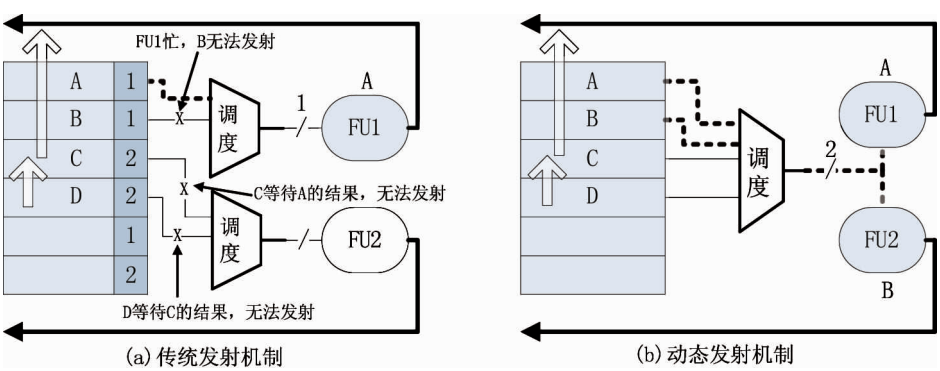

图 7　传统发射机制和动态发射机制的不同

动态发射机制取消了全局队列内部指令的分组,根据当前运算部件空闲数量,同时从队列内查找驻留时间最久的 N 条指令。图 7(b)所示,队列首先确定当前可用的运算部件有 FU1 和 FU2,从当前操作数已写回的指令(A、B)中查找 2 条指令,A、B 被选中,分别分发至 FU1 和 FU2。在此周期,并不会出现运算部件闲置的情况。而不相关的指令 A 和 B 同时进入执行阶段,没有发生运算部件冲突。

3　电路实现

上述的动态发射机制依赖于一个 N 选 M 的电路,且 M 的值在各个周期还是动态变化的,而无论是紧实队列中基于位置的调度策略,还是稀疏队列中基于时间的调度策略,其实现电路都是一个 N 选 1 编码电路。本文保留了年龄触发器来记录指令的先后顺序,用年龄触发器中“1”的个数表征指令在队列内驻留时间的长短,“1”的数量越多,驻留时间越长。乘法器中常见的压缩树可以计算序列中“1”的个数,但是仔细分析电路逻辑可知,对于指令发射数量的要求决定了“1”的个数是有限制的,这一限制在数字电路中需要多位比较器来实现,进一步增加了电路的延时。模拟电路可以提供快速的累加和比较运算,即灵敏放大器。将年龄触发器中“1”的个数和指令发射数量转化成电流或电压信号,基于基尔霍夫定律,通过串并联网络得到累加信号,再经过灵敏放大器选出待发射指令,接入分发网络实现的位置掩码得到不同指令在队列中的位置。整个电

路结构如图 8 所示。

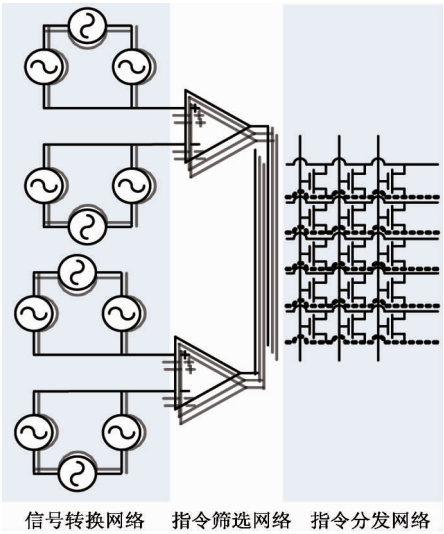

图 8　N 选 M 电路结构

作为电路核心的灵敏放大器有两种类型:电压型和电流型。电压型灵敏放大器的特点是结构较为简单、工作稳定,但是工作延时受到位线电容放电的影响,速度较慢;电流型灵敏放大器感应位线上微小的电流差分值,通过电路结构完成电流到电压的转化并进行放大,独立于位线放电,所以速度较快。为获取可比拟紧实队列的调度速度,本文采用的是基于自偏置电荷转移(self-biased charge-transfer, SBCT)的灵敏放大器,相应的累加计算采用的是电流源并联结构。

电路分成 3 个部分:信号转换网络、指令筛选网络和指令分发网络。信号转换网络通过电流源并联将年龄触发器中的值转化成离散的驻留时间电流,同时根据需要发射的指令数量得到参考电流,筛选网络对比电流值选出待发指令的集合,最后通过分发网络得到指令在队列内的地址。

◆　信号转换网络

将年龄触发器接入指令电流源开关 MP_s,根据触发器的内容有条件地打开电流源开关,由 PMOS 管的物理尺寸决定节点电流 I_t 的大小,式为

$$I_t = k_n \frac{W}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right] \qquad (1)$$

理论上节点电流越小,电路的漏电功耗越小,但会受到后一级运放的灵敏度以及噪声容限的限制。

式中,W 为沟道宽度,L 为沟道长度,V_{GS} 为栅源电压,V_{DS} 为源漏电压,V_T 为阈值电压,k_n 称为工艺跨导参数,式为 $k_n = \mu_n C_{ox}$。

在确定的工艺下,即 k_n、W 和 V_T 不变的前提下,通过调节沟道长度 L 来改变节点电流的大小,年龄触发器中记录的“时刻”决定电流支路的数量,在并联网络端口处即可获得代表该指令驻留时间的电流 I_{time}。根据发射指令数量,开启对应的参考电流开关 MP_r,参考电流开关尺寸需要和指令电流开关完全一致,并且在版图规划时采用叉指图形或者尽可能地紧贴在一起,以避免工艺偏差引入的电流噪声。

考虑到电路在芯片中的复杂环境,特别是指令队列本身就是一个高翻转率的模块,为提高电路的噪声容限,在电流源网络中增加阈值电流源。阈值电流的大小为 0.5 倍的 I_t,在当前指令符合发射条件下,指令电流与参考电流最接近的差值为 $-0.5I_t$,在当前指令不符合发射条件时,指令电流与参考电流最接近的差值为 $0.5I_t$。例如,对于一个 8 项的队列,目前需要发射 2 条指令,开启 2 个参考电流源,驻留时间最久的指令年龄触发器中共有 0 个“1”,指令电流源不打开,该指令的驻留时间电流与参考电流差值为 $-1.5I_t$,次久的指令开启 1 个电流源,该条指令的驻留时间电流与参考电流差值为 $-0.5I_t$,后一条不满足发射条件的指令开启 2 个电流源,与参考电流差值为 $0.5I_t$。

◆　指令筛选网络

筛选网络通过判别指令驻留时间的长短,决定该条指令是否可以通过。电路结构为一个标准的灵敏电流放大器。传统的灵敏放大器输入和输出共用同一个端口,容易产生串扰,造成输出结果错误,并且当该端口作为输出端时,会浪费部分时间在对位线寄生电容充电放电上。因此改进电路可添加一对隔离管,具体结构如图 9 所示。

灵敏放大器工作过程分为两个模式:预充模式和放大模式。在预充模式下,PC、SAEN 和 Ysel 为“0”,MP0,MP1,MP2 导通,平衡管 MP0 使两边的节点 init 和 nint 电压处于近似相等的状态,电流传输管 MP3 和 MP4 导通,电流通过这两管传输到 A、B

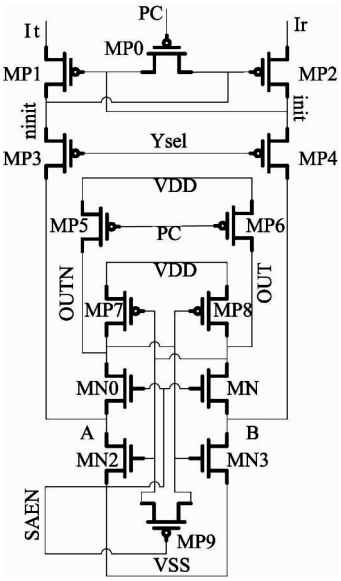

图 9　灵敏放大器结构

端,等待放大信号的到来; MN0、MN1 关断,将放大通路阻断;MP5,MP6,MP7,MP8,MP9 打开,输出端口 OUT、OUTN 预充至 VDD,并拉平两端口电压。这一过程主要是为灵敏放大器在下一阶段放大信号做准备。在放大模式下,PC、SAEN 跳变为“1”,MP5、MP6 关断,停止为输出端口充电;MP9、MN0、MN1 导通,放大通路由此形成。之后 Ysel 变为高电平,将电流传输管关闭,此时 A、B 端口之间的微小电流信号差通过正反馈被迅速放大成全摆幅的逻辑信号。当输出信号稳定后,灵敏放大器重新进入到预充模式,开始新一个周期的工作。在整个灵敏放大器电路中,MP7、MP8、MN2、MN3 管构成的交叉耦合反相器是核心,其管子的尺寸直接影响放大器的速度和精度。前文已经推导出电流信号差值的临界点为 $0.5I_t$,可以由此调整放大管尺寸,获得极快的放大速度。

◆　指令分发网络

筛选网络确定了指令是否可以发射,但是每一个发射端口都需要对应的指令地址来读取寄存器堆。对于驻留时间在筛选窗口以内的指令,会被筛选网络中的每一个灵敏放大器选中。例如表 1,当前需要发射 2 条指令,则对于驻留时间最久的指令,会被筛选网络中参考指令码为 111 和 11 的两个灵敏放大器选中,对于次久的指令,仅会被参考指令码为 11 的灵敏放大器选中,若不做修正,得到的指令位置分别为 1000 和 1100,对于位置 1 处的指令读取错误。驻留最久的指令被 111 筛选后,需要纠正 11 的结果,则指令位置分别为 1000 和 0100,两条指令都会正确读出。

利用 NMOS 连接信号端和电源地,用更靠前的指令筛选结果来控制 NMOS 的开关。一旦指令已经通过筛选,其后所有的参考指令码得到的结果,都会被泄放至低电平。

表 1　修正后得到指令的位置

指令位置	年龄寄存器值	待发指令限制	111	11	111	11
0	1111	11	1	1	1	1->0
1	1110	11	0	1	0	1
2	1100	11	0	0	0	0
3	1000	11	0	0	0	0

分发网络的另一个作用是修复波形。因为灵敏放大器会交替工作在预充模式和放大模式,而预充模式下放大器的输出会被强制拉升至高电平,这一阶段是没有任何信息的。在分发网络端口增加一个 RS 触发器,S 接端口时钟信号,即可使因放大器预充模式出现的高电平脉冲修复为维持上一个放大模式下的值。

4　电路仿真

本文基于 SMIC 40nm 工艺实现了一个 64 项队列的动态发射电路,为降低参考电流导致的漏电功耗,限制发射数量上限为 4 条。在供电电压为 1.1V 的情况下,取分辨电流为 3μA,通过 Hspice 仿真可知,电路可以稳定工作在 8.3G。作为对比,取龙芯某芯片中 64 项队列的调度电路[11],其最长路径约为 460ps,工作频率在 2.2G 左右。图 10 为 64 项队列中单项的调度电路。

电路的工作分为两个阶段:截止阶段和求值阶段。不同阶段下,需要多个控制信号配合工作。当电路工作在截止阶段时,信号转换网络通过时钟信号,关断所有的电流源,以减小电路功耗;筛选网络

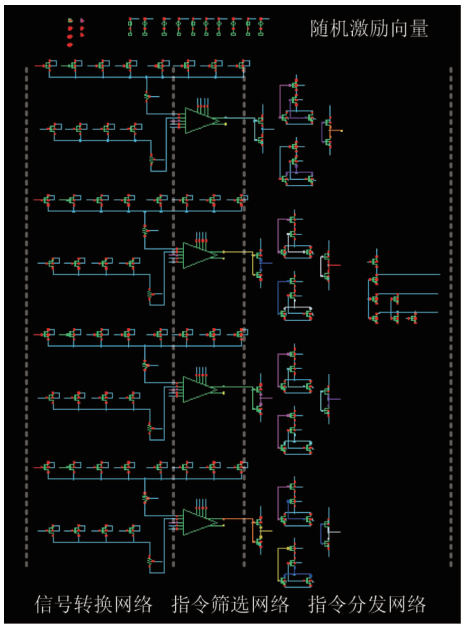

图 10　64 项队列中单项的调度电路

中的 PC、SAEN 信号为低电平，使灵敏放大器处于预充模式；分发网络中的 RS 触发器，保持端口处前半个周期的值，保证该信号可以维持整个时钟周期。

当电路工作在求值阶段时，信号转换网络中的时钟使能打开，电压信号转化成电流信号；筛选网络中，为保证端口电压被充分传播至 A、B 端，PC 信号先于 SAEN 信号跳变为高，交叉耦合反相器开始工作，灵敏放大器工作在放大模式；分发网络中 RS 触发器中的输出开始跟踪放大器的结果。

图 11 截取的是一串随机激励向量下的仿真结果片段。图中分别展示了电路三个主要部分的端口波形，显示了指令命中、低位数据纠正以及数据保持的情况，验证电路功能正确。其中指令命中部分显示的是待测指令的年龄触发器中“1”的个数由 4 条减少为 3 条时，即待测指令属于队列中驻留时间最久的四条指令之一，满足发射条件，放大器工作在放大模式，输出高电平；低位数据纠正显示了对于前三个发射端口，该条指令通过独热码的地址掩码，将输出信号从高电平泄放至低电平；数据保持显示的 RS 触发器工作，将预充阶段出现的高电平脉冲修复为上一个放大模式下的值。多次测量时钟上升沿到输出端口处的延时为 105ps ~ 121ps 之间。

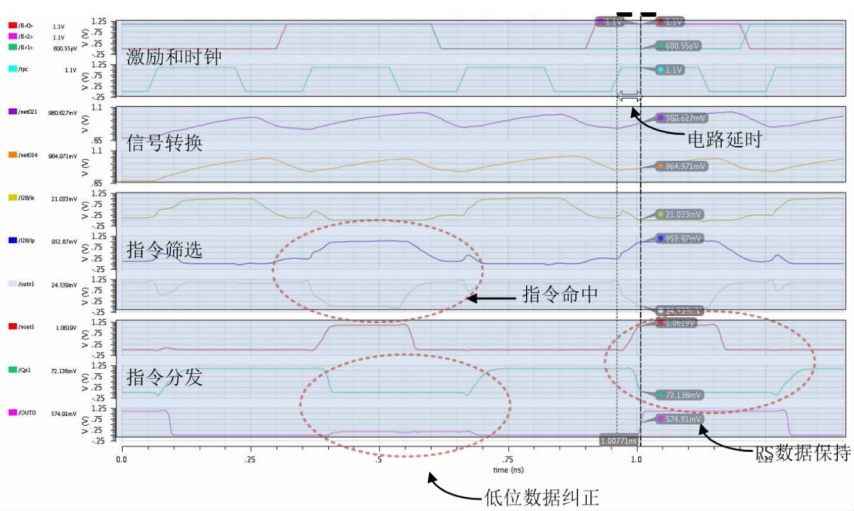

图 11　电路仿真结果

5　结　论

本文所提出的动态发射机制，从根本上解决了提前为指令分配运算部件导致的“运算部件冲突”问题，采用 SMIC 40nm 工艺实现的动态发射电路，实现了单周期内并行选取多条指令，且时序较目前商用芯片中发射单条指令的时序提高了接近 3 倍。将模拟电路中的放大器取代数字电路中的比较逻辑这一思路可以应用在其他方面，如前导零电路、判断溢出电路等。队列中引入了灵敏放大器，在应用时需要注意和数字电路的隔离，以保证电路工作的可靠性。

参考文献

[1] Kessler R E. The alpha 21264 microprocessor. *Micro IEEE*, 1999, 19(2):24-36

[2] Sassone P G, Rupley Ii J, Brekelbaum E, et al. Matrix scheduler reloaded. In: Proceedings of the 34th Annual International Symposium on Computer Architecture, San Diego, USA, 2007. 335-346

[3] Tendler J M. IBM e-server power4 system microarchitecture. *IBM Journal of Research & Development*, 2001, 46(1): 5-25

[4] Goshima M, Nishino K, Nakashima Y, et al. A high-speed dynamic instruction scheduling scheme for supersealar processors. *Proceedings of International Symposium on Microarchitecture*, 2001, 142(9):225-236

[5] Bai Y, Bahar R I. A dynamically reconfigurable mixed in-order/out-of-order issue queue for power-aware microprocessors. In: Proceedings of the IEEE Computer Society Symposium on VLSI, Tampa, USA, 2003. 139-146

[6] Hossain M, Fluhr E, Hall A, et al. Physical design and implementation of POWER8™ (P8) server class processor. In: Proceedings of the IEEE International Midwest Symposium on Circuits and Systems, Fort Collins, USA, 2015. 1-4

[7] Fayneh E, Yuffe M, Knoll E, et al. 4.1 14nm 6th-generation Core processor SoC with low power consumption and improved performance. In: Proceedings of the IEEE International Solid-State Circuits Conference, San Francisco, USA, 2016. 72-73

[8] Buyuktosunoglu A, El-Moursy A, Albonesi D H. An oldest-first selection logic implementation for non-compacting issue queues. In: Proceedings of the IEEE International ASIC/SOC Conference, Rochester, USA, 2002. 31-35

[9] 胡伟武, 陈云霁, 肖俊华等. 计算机体系结构. 北京: 清华大学出版社, 2011. 124-127

[10] Yamaguchi K, Kora Y, Ando H. Evaluation of issue queue delay: banking tag RAM and identifying correct critical path. In: Proceedings of the 29th International Conference on Computer Design (ICCD), Amherst, USA, 2011. 313-319

[11] Hu W, Zhang Y, Yang L, et al. Godson-3B1500: A 32nm 1.35GHz 40W 172.8GFLOPS 8-core processor. In: Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers, San Francisco, USA, 2013, 56:54-55

A dynamic schedule mechanism in non-compacting issue queues and its implementation

Liu Zhen[*,**,***], Wang Jian[*,***], Zhao Pengfei[****], Ding Jianping[****]

([*] Key Laboratory of Computer System and Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

([****] Loongson Technology Corporation, Beijing 100095)

Abstract

In view of the fact that non-pipelining multi-cycle instructions in high performance processors with several functional units always prevent the issue of following noncorrelation instructions, a mechanism of dynamic issue was presented. According to the numbers of vacant functional units, the system picks instructions in a non-compacting issue queue whose operand is written back instead of dispatching it in advance. The system based on multi-bit selector includes three parts. Firstly, the current source network converts the flip-flop outputs into current signals. Then, a current sense amplifier compares the output of the network and the reference current which represents the number of vacant units. Finally, as a mask, a NMOS discharge network corrects the low bit data, thus repeating signal "1s" is filtered. The circuit was implemented in SMIC 40nm and it could work at 8GHz.

Key words: issue queue, dynamic schedule, non-compacting issue queue, multi-bit selector, sense amplifier