Engineering Technology and Development, 工程技术与发展 (20)2022,4
ISSN:2661-3506 (Online) 2661-3492(Print)

利用半径实现低功率分配器技术

尤达·布姆内特
西班牙

摘　要：本文描述了一种分配技术的设计，将低功耗技术应用于机组的设计，并考虑了能量延迟权衡。与没有对能量进行优化的标准实现相比，除频器中的能量耗散可以减少高达70%，而不惩罚延迟。在这种划分技术中，我们比较了半径x-8分频器与通过重叠三个半径-2级和一个半径-4分频器获得的分频器进行了比较。结果表明，分频器的延迟与具有重叠级的分频器相似，但面积较小。半径-8在半径x-4上的加速约为23.58%，完成分割所耗散的能量几乎相同，尽管半径-8的面积大67.58%。

IMPLEMENTATION OF LOW POWER DIVIDER TECHNIQUES USING RADIX

Yoda Bumnet
Spain

Abstract: This work describes the design of a divder technique Low-power techniques are applied in the design of the unit, and energy-delay tradeoffs considered. The energy dissipation in the divider can be reduced by up to 70% with respect to a standard implementation not optimized for energy, without penalizing the latency. In this dividing technique we compare the radix-8 divider is compared with one obtained by overlapping three radix-2 stages and with a radix-4 divider. Results show that the latency of our divider is similar to that of the divider with overlapped stages, but the area is smaller. The speed-up of the radix-8 over the radix-4 is about 23.58% and the energy dissipated to complete a division is almost the same, although the area of the radix-8 is 67.58% larger.

1 介绍

除法是四个基本算术运算中最复杂的一个，一般来说，并不能产生一个精确的答案，因为股息不一定是除数的倍数。因此，正确的商和余数通常通过执行一系列的迭代得到，直到达到所期望的精度。这个过程称为顺序除法，是许多实际实现[1,3,4]的基本原理。

本文描述了一种双精度半径-8分频器的设计。该装置的设计目的是为了减少能量耗散而不惩罚延迟。

所使用的算法是在[3]中详细描述的数字递归划分算法。数字递归算法在每次迭代中都有一个结果位的固定数量，由基数决定。较高的雷达线减少了完成操作的迭代次数，但增加了循环时间和电路的复杂性。高半径分频器的实现并不多，本文的目的是确定一个半径-8单元的性能和能量耗散。在[3]中对半径-8分频器的面积和性能进行了评估，但没有实际实现。在[4]中，实现了一种基于共享硬件的基数-8除法和平方根算法。在[11]中，提出了一个具有三个半径-2重叠级的分频器的实现。在[5]中，由于数字选择函数的复杂性增加，导根大于4的阶段并不方便。我们在这里提出了我们的低功率半径x-8分频器的实现，将其性能与[11]中提出的实现进行了比较，并评估了相对于更常见的半径x-4单元的加速和能源消耗。此外，还考虑了一些能量延迟的权衡。

该设计的主要目标是在最短的时间内完成操作。然后采用低功耗设计技术，以减少机组内的能量耗散。面积并没有被最小化，但一些能量减少技术也减少了面积。

分隔器的实现是使用护照0.56m标准细胞库[2]完成的。该结构模型通过人工设计和功能块的合成得到，并采用自动楼层规划和路由的方式进行布局。通过在算法中选择同时影响关键路径和迭代次数的适当参数，减少了划分的延迟，如第2节所述。第3节描述了低功率系统的设计。

在第4节中，将分频器与通过重叠三个半径-2级获得的半径-8进行比较，使用类似于在SunUltraSPARC处理器[11]中实现的方案，并使用半径-4分频器[8]。

结果表明，采用适当的低功率设计技术，半径-8单元的能量耗散可降低70%。

2 算法和实现

在[3]中详细描述的radix-8分割算法是通过残差递推来实现的

$$w[j+1] = 8w[j]-qj+1d$$
$$j = 0,1.....m$$

初始值为w[0] = x，其中x是被除数，d是除数，qj+1是第j次迭代时的商数。d和x在[0.5,1)中都被归一化了。商数字用有符号数字表示{-a，.....，-101.....，a}，冗余因子p = a/7。残余w存储为存储表示（wS和wC）。在每次迭代中，商数由选择函数决定

$$qi = SEL(ds,y)$$

其中，d在第一个分数位后被截断，[y] = 8wS+8wC在t个分数位后被截断。

递归通过一个选择函数(SEL)、一个多个生成器、一个携带存加法器(CSA)和两个寄存器来实现，以存储残差的携带存表示。

为了避免实现复杂的多重生成器，将商数分为权重为4的qH和权重为1的qL(qj = 4qH+qL)，将每个部分的数字集简化为{-2、-1、0、1、2}。四个信号(M2、M1、P1、P2)用于表示四分之一的代码中的这五个值（零被编码为（0,0,0,0））。这种表示方式使多重生成器变得简单。

由于选择函数(SEL)处于关键路径中，因此要获得最小的延迟，我们必须最小化其延迟。我们探索了a的三个可能值的实现：6、7和10（使用上述表示法可能达到的最大值）。表1显示了结果的摘要。利用指南针ASIC合成器合成了选择函数的VHDL描述，得到了门级实现。这包括[ˆy]的携带保存表示的同化和实际的数字选择函数。

表1 汇总选择功能

Step	HW	LW	Explanation
initialize	0000	0111	set HW to 0, set LW to dividend
shift-left	0000	111x	
subtract	-0011		HW - divisor = -0011
	-0011	1110	result is < 0, set LSB of LW to 0, restore HW
restore	0000	1110	HW + divisor = 0000
shift-left	0001	110x	
subtract	-0011		HW - divisor = -0010
	-0010	1100	result is < 0, set LSB of LW to 0, restore HW
restore	0001	1100	HW + divisor = 0001
shift-left	0011	100x	
subtract	-0011		HW - divisor = 0000
	0000	1001	result is ≥ 0, set LSB of LW to 1, no restoring
shift-left	0001	001x	
subtract	-0011		HW - divisor = -0010
	-0010	0010	result is < 0, set LSB of LW to 0, restore HW
restore	0001	0010	HW + divisor = 0001

从表1中，我们可以看到，a = 7的SEL和a = 6一样快，但它的面积更小。令人惊讶的是，过度冗余情况下的 = 10的延迟更大。因此，选择a = 7的SEL，这导致一个冗余因子p = 1。

除法器的第一个实现如图1所示。该方案由一个控制器完成（图中未描述）。转换块执行转换和舍入。在最后一次迭代中，商根据最后残差的符号和检测它是否为零的信号进行四舍五入，这是由符号-零检测块(SZD)产生的。

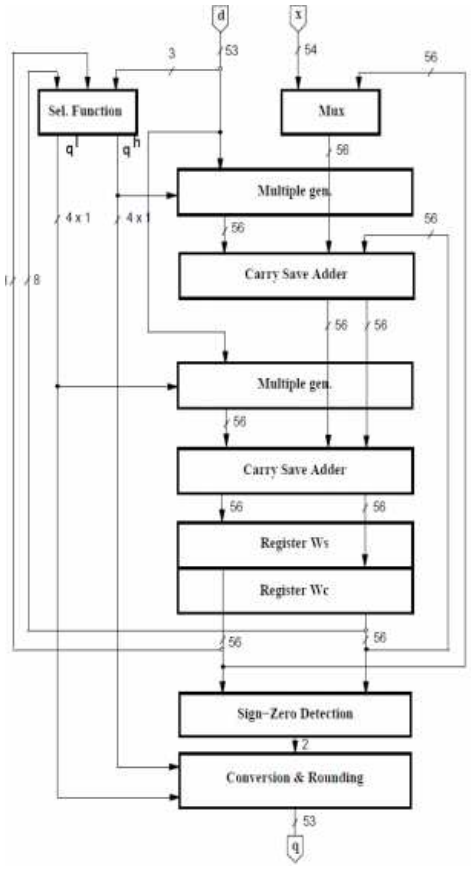

图1 基数为8的分法器实现。

为了在使用分值操作时符合IEEE的双精度标准（53位显著性和[1,2）中的标准化），在操作数上执行1位移位。此外，为了在第一次迭代中有一个界残差(w[0] = x-d)，当x > d时，我们向右移x1位，得到一个分数商。为了计算商的53位和一个额外的位来执行舍入，需要54/3次 = 18次迭代。需要一个额外的循环来加载值x作为第一个剩余的w[0]。但是，对于所提出的架构和选择功能，实现这一点的最简单的方法是这样做：

清除w的寄存器（这是在前一个除法的结尾完成的）。通过我们已经实现的选择函数，这会产生一个q1 = 1。

计算w[1] = x-d使用硬件进行递归。这需要一个多路复用器，它不在关键路径上。对于q1是1，我们将被除数向右移三位。因此，有必要相应地改变最终的商，

总之，加载循环被一个额外的迭代所取代，总共有20次迭代：19次用于计算数字，1次用于四舍入。最后，在[1,2)中，通过向左移动四个位置进行归一化。请注意，所有的转移都是通过接线来完成的，并且不会影响操作的延迟。在递归(w[j])中，我们需要54个分数位和2个整数位：一个是为了保持符号，另一个是为了避免cs表示中的溢出(即p = 1)。有两种可能的关键路径，一种通过qH，另一种通过qL。由于qH的延迟小于qL的延迟，但要遍历的加法器的数量更大，所以一个好的设计试图平衡两条路径的延迟。由此产生的关键路径，即预布局。由于时钟分布树造成的延迟和互连电容导

致布局后的关键路径为10.5ns。

3 低功耗实现

在本节中，我们在图1的单元中应用了减少能量耗散的技术。这里应用的一些技术在[8]中描述了一个半径-4分隔器的情况。这些都适用于第3.1-3.4节中的radix-8。此外，还介绍了针对radix-8的技术：3.5节描述了选择函数的划分，3.6节将修改的转换算法（完整描述参见[8]）扩展到p = 1。最后，在第3.7节中给出了双电压对低功率实现的影响。

图2显示了低功耗radix-8分频器的实现，表2总结了通过应用第3.1-3.7节中描述的低功耗技术所获得的结果。在表中，条目std是标准实现，针对速度进行了优化，条目l-p是低功耗实现。我们不可能用我们的单元库来实现双电压，所以条目d-v是一个可能实现的估计。

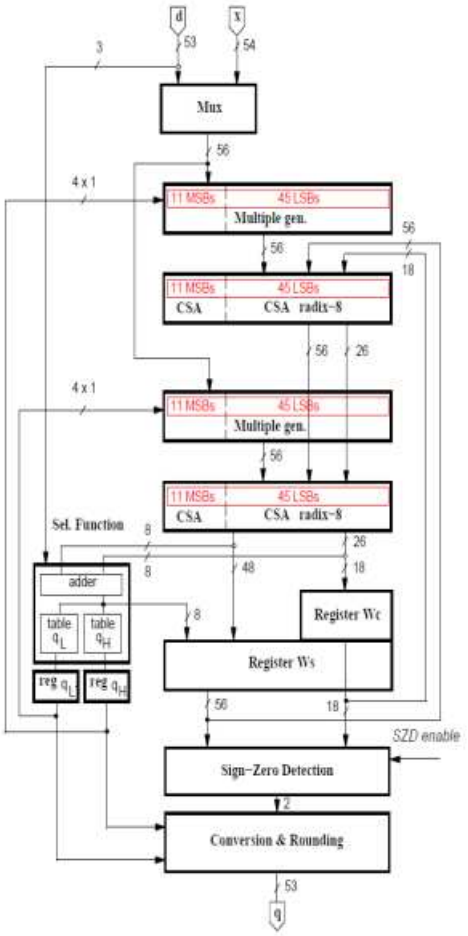

图2 低功耗的实现。

3.1 关闭非活动块

符号零检测块(SZD)只在舍入步骤中使用，通过在重复步骤中在其输入处强制执行一个恒定的逻辑值来关闭。减少的幅度约为8%。

3.2 重复再计时

寄存器在顺序系统中的位置会影响能量耗散。重定时是一种转换，包括在一个顺序电路中重新定位寄存器，而不修改其外部行为[6]。

对于递归，通过将图1的选择函数从周期的第一部分移到前一个周期的最后一部分来完成重新计时(参见图3.a和图3.b)。需要两个新的寄存器(qH和qL)来存储商数数字。

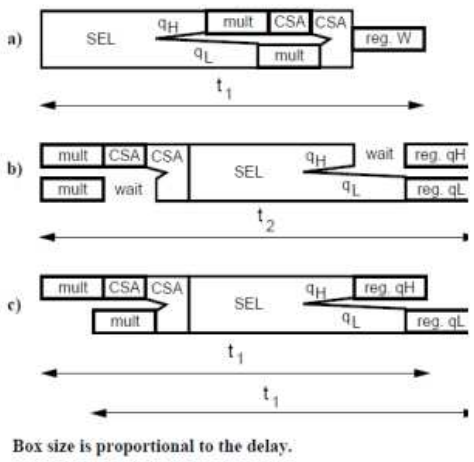

图3 重新定时和关键路径。a）重新计时前，b）重新计时后，c）重新计时后，并使时钟倾斜。

通过重新计时递归时间，我们减少了多个生成器和csa中的切换活动，并改变了现在被限制在8个最重要的位的关键路径。这允许在复发中的其余位被重新设计以适应低功耗。

3.3 减少多路复用器中的转换

图1中的多路复用器用于选择第一次迭代中的x或其他迭代中的残差。通过将其移出递推点，可以减少mux中的转换数量，如图2所示。因此，通过将寄存器分别将寄存器qH和qL重置为0和-1，并在w[0] = 0-(-x)中存储x来修改第一个循环中的操作。

3.4 改变冗余的表示方式

通过使用radix-8携带保存表示，每次重复使用三个和位和一个携带位，如图4所示，我们只需要为每个数字存储一个携带位，而不是三个。这可以对45个重新计时后不在关键路径上的45个lsb完成。此外，在重新计时后，在选择功能块内的加法器中同化的8个msb（图2），可以存储在w_S中，消除了w_C中的另外8个触发器。

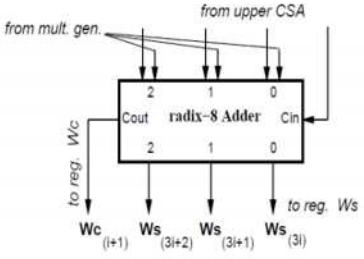

图4 基数8携带存储加法器（低）。

通过重新计时，移动mux，并改变表示，l-p相对于std的减少约为14%。

3.5 划分和禁用选择功能

函数商数选择是除数的3位和残差的8位的函数。

在radix-8的情况下，图5显示了对较高部分和较低部分

的八个部分(d3的所有可能值)的划分。解复用器将[^y]的同化值传输到所选的选择表对中，并强制将其他选择表的输出设为零。最后，一个OR门的数组组合了部分值。

实验结果表明，配分选择函数耗散的能量较少，但临界路径增加。较小的选择函数比整体的实现要快，但解复用器和OR门的延迟抵消了改进。

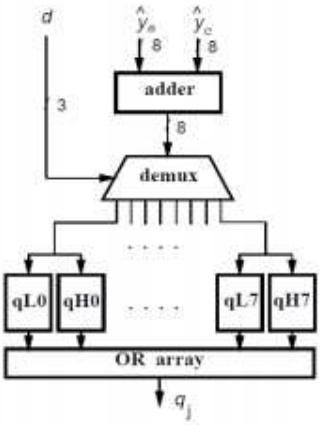

图5 分区选择功能

3.6 转换和舍入中的修改

动态转换算法[3]执行2的补中从符号数字表示到传统表示的转换。转换是在产生数字时完成的，不需要携带传播加法器。该算法，在其最初的公式中，没有考虑能量耗散，导致约占整个除法器的30%。

对于p = 1，部分商存储在三个寄存器(Q、QM、QP)中，通过移位加载操作更新，并在舍入期间在这些寄存器中选择最终商。在单元中耗散的大量能量主要是由于每次迭代过程中的移动和用于实现寄存器的触发器的数量。

作为减少能量耗散的第一步，我们加载每个数字在其最终位置[9]。这样一来，我们就避免了沿着寄存器移位数字。为了确定负载位置，我们使用一个18位环计数器C，每个数字加载一位。

3.7 使用双极电压

在电池中耗散的功率取决于电压供应(VDD)的平方，因此通过降低这个电压[1]可以节省大量的能量。然而，通过降低电压，延迟增加，因此为了保持性能，这种技术只应用于不在关键路径上的电池。不同的电源电压需要电平转换电路来消耗能量。然而，通过使用两个电压，我们只需要在从低电压[13]到高电压[13]时进行电平偏移。在我们的例子中，可以重新设计45个复发的最不重要的位，如图7所示。电压级移动器是不需要的，直到一个特定的数字移动到11个msb，通过移动跨迭代和进入关键路径。通过将电压电平移位器（总共3个）放置在11个msb之前的数字中，周期时间不会增加，并且在电平移位电路中消耗的能量很小。

我们也可以将双电压技术应用于不在临界路径上的转换和圆单元。所需的电平移位器数量为53个，与双精度显著性表示一样多，但由于新算法，每个位开关最多两次，电平移位器中的能量耗散不能抵消由于较低电压而产生的减少。

我们估计，如果有低压门可用，d-v相对于l-p减少了约50%。

4 与 radix-4 实现的比较

为了评估能量和延迟之间的权衡，我们将radix-8分配器与之前在[8]中提出的radix-4单元进行了比较。我们选择了一个半径-4分频器进行比较，因为算法与半径的唯一变化是相同的，并且减少能量的技术是相似的。radix-4分法器的缺点是需要更多的周期来计算商（30个周期），但具有更短的迭代周期（更快的时钟）和更小的面积。所使用的性能指标是每个操作所经过的时间，这是tdiv = Tcycle×(no。的周期的)。能量测量是每除法的能量Ediv，我们还包括每周期的能量Epc。表5总结了这两个分隔器的特征。

表5 基数4和基数8的分隔器

	radix-4	radix-8	[unit]
T_{cycle}	9.3	11.5	ns
t_{div}	260	215	ns
E_{div}	26.0	26.6	nJ
E_{pc}	1.9	2.4	nJ
Area	2.2	2.8	mm^2

radix-8超过radix-4的加速约为20%，而每分的能量增加不到2%。另一方面，半径-8的能量比每个周期大50%。我们的设计表明，面积的增加（约50%）并没有反映出完成操作的能量消耗，由于循环次数的减少，这几乎是相同的。radix-4分频器更小，但速度更慢，每次操作消耗的能量几乎相同。

REFERENCES

[1]A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design.* Kluwer Academic Publishers, 1995.

[2]Compass Design Automation. *Passport - 0.6-Micron, 3-Volt, High-Performance Standard Cell Library*. Compass Design Automation, Inc., 1994.

[3]M. Ergegovac and T. Lang. *Division and Square Root:Digit-Recurrence Algorithms and Implementations.* Kluwer Academic Publisher, 1994.

[4]J. Fandrianto. Algorithm for high-speed shared radix-8 division and radix-8 square root. *Proc. of 9th Symposium on Computer Arithmetic*, pages 68-75, Sept. 1989.

[5]D. Harris, S. Oberman, and M. Horowitz. SRT division architectures and implementations. *Proc. of 13th Symposium on Computer Arithmetic*, pages 18-25, July 1997.

[6]J. Monteiro, S. Devadas, and A. Ghosh. Retiming sequential circuits for low power. *Proc. of 1993 International Conference on Computer-Aided Design (ICCAD)*, pages 398-402, Nov. 1993.

[7]A. Nannarelli. PET: Power evaluation tool, Aug. 1996. http://www.eng.uci.edu/numlab/PET/.

[8]A. Nannarelli and T. Lang. Low-power radix-4 divider. *Proc. of International Symposium on Low Power Electronics and Design*, pages 205-208, Aug. 1996.

[9]A. Nannarelli and T. Lang. Low-Power Convert-and-Round Unit. *Technical Report*, Jan 1997. Available on the WWW at http://www.eng.uci.edu/numlab/archive/pub/nl97p02/.

[10]W. Nebel and J. Mermet editors. *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, 1997

[11]A. Prabhu and G. Zyner. 167 MHz radix-8 divide and square root using overlapped radix-2 stages. *Proc. of 12th Symposium on Computer Arithmetic*, pages 155-162, July 1995.

[12]J. M. Rabaey, M. Pedram, et al. *Low Power Design Methodologies*. Kluwer Academic Publishers, 1996.

[13]K. Usami and M. Horowitz. Clustered voltage scaling technique for low-power design. *Proc. of International Symposium on Low Power Design*, pages 3-8, Apr. 1995.