

一种同步输出的稀疏树前导零检测电路^①

刘臻^② * * * * * 王剑^③ * * * * * 王昊 * * * * *

(* 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 为了降低前导零检测电路的延时和提高端口输出的同步性,提出了一种稀疏树前导零检测结构及动态电路的实现方法。通过递归前导零算法表达式定义了两种新的布尔运算逻辑,以构成稀疏树中的结点。精确控制动态电路中预充时钟的偏移量,在抑制电路漏电的同时控制输出端口延时差。该结构能够以最少的逻辑级数和均衡的运算单元负载实现检测前导零功能,对于不同数量的待测信号,通过扩展“点操作”和“块操作”单元互连网络来生成新电路。采用伪随机激励向量验证了电路功能的正确性,大幅缩短了验证时间,在 SMIC 40nm 工艺下仿真显示,各端口输出延时差小于 1ps,数据路径长度可降低 20%。

关键词 前导零检测器,点操作,块操作,稀疏树,同步输出,动态电路

0 引 言

前导零检测(leading-one-detector, LOD)电路在数字集成电路中应用十分广泛。现代数字信号处理器(DSP)处理的指令中超过半数都是加法和乘法指令^[1],各种 DSP 算法都在尽力优化运算部件,其中一个重要步骤是尾数操作,需要提前检测尾数加、减结果的首个“1”的位置,从而在结果计算出来后立即进行移位操作,使首个“1”移至最左端^[2]。超标量处理器中的乱序发射技术主要依托于指令发射队列,判断队列中指令的状态以及选择待发指令都是通过查找状态寄存器中的前导零来实现的^[3],这两条数据通路是发射队列的时序瓶颈^[4]、查询速度决定了发射队列的性能。

LOD 电路是一种可以检测出二进制序列中前导零的个数或者首个“1”的位置的电路。LOD 是常见的数字电路,为节省设计时间,目前多采用商业工具综合的手段来生成门级网表,随机的逻辑门组合造成输入端口到不同输出端口的路径延时会有一定的差异,当时序收敛过程中较长的路径满足了建立时间约束时,较短的路径往往会出现保持时间违例,尤其是在不同工艺角下不同逻辑门的偏差并不能保证一致,这种输出延时差导致的保持时间违例会变得更加严重。

本文利用动态电路实现了一种稀疏树^[5] LOD 电路结构,有效降低了检测信号的逐级传播效应,以此来提高时序。电路复用两种自定义的运算逻辑:“点操作”和“块操作”,简化了设计复杂度,提高了设计效率。同时,每一位待测信号到达输出端口所

① 国家“核高基”科技重大专项课题(2014ZX01020201, 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;E-mail: liuzhen@ict.ac.cn
③ 通讯作者,E-mail: jw@ict.ac.cn
(收稿日期:2016-10-17)

经过的运算单元数量相同,且每级单元都通过同一时钟控制,保证了各端口同步输出。构建伪随机激励向量对电路进行功能验证,极大地缩减了验证时间。在 SMIC 40nm 工艺下,通过 Hspice 仿真显示,本研究所设计电路结构较目前主流结构路径长度可缩短 20%,输出延时差几乎为零。

本文介绍了 LOD 电路的设计目标以及目前主流的电路结构,详述了构建稀疏树的过程,进而介绍了如何利用动态电路实现该结构,最后通过 Hspice 仿真,比较了不同电路的性能,验证了结构正确性。

1 LOD 电路相关工作

高性能处理器中经过 LOD 电路的路径时序紧张,但该电路所占面积很小,因此目前的电路结构都是在牺牲部分面积和功耗的前提下,提升其性能,以适应处理器越来越高的主频^[6]。LOD 逻辑上具有一定的规整性,但又不像 RAM 一样包含完全相同的 BIT 单元,全定制的方法会极大地增加设计成本,因此在量产芯片中很少被应用。同时,随着 LOD 输入端口数的增长,单元数量和路径数量以指数级膨胀,带来的新问题是满足所有路径的时序约束。触发器是时序路径上的起止点,其采样方式决定了每条路径都有最长延时约束和最短延时约束,芯片的运行周期是时序路径长度的上限,触发器的保持时间是时序路径长度的下限。芯片主频的提高要求降低上限,片上工艺波动和信号串扰会抬高下限^[7],因此,留给时序路径的设计空间在不断压缩,若是相同起止点的两条路径间延时差超过上下限之差,则这两条路径是没有办法被修复的。总结以上介绍,设计 LOD 电路时需要关注以下三个方面:(1)提高电路频率;(2)缩短设计时间;(3)降低输出延时差。

遗传算法^[8]能很好地优化组合电路,在短时间内得到一个满足设计约束的极优解,商业综合工具大多采用该算法来生成门级网表。算法依据搜索空间、市值函数和遗传操作实现最终的收敛。搜索空间是设计中可以使用的所有标准单元,如 AND、OR、NOT 门,适值函数是 LOD 电路的真值表达式,表 1

为一个 4 位 LOD 电路的适值函数。然而,遗传算法得到的电路具有一定的随机性,各路径上的逻辑级数不尽相同,因此输出的延时差非常大,一定程度上增加了时序收敛的难度。

表 1 4 位 LOD 电路的真值表达式

输入				输出		标志位
I ₃	I ₂	I ₁	I ₀	Y ₁	Y ₀	V
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	X	0	1	0
0	1	X	X	1	0	0
1	X	X	X	1	1	0

文献[9,10]重复调用相同结构的电路模块,以此来保证不同输入到输出路径之间的级数相同。图 1 显示了如何利用一个 8bit LOD 来搭建 32bit LOD 电路。该方案选取的不同器件间延时差本身就很大,例如选择器的延时接近 2 倍于反相器,因此信号传播通路是否经过选择器,不仅影响该电路的最高频率,还会影响各个输出端口间的延时差。

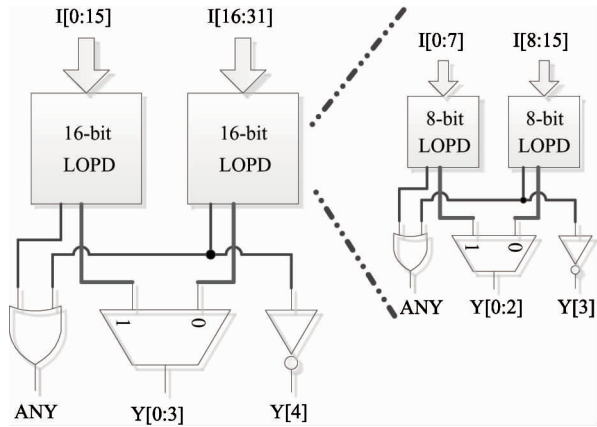


图 1 K. Kunaraj 结构递归调用子模块搭建新电路

Buyuktosunoglu 等人从信号传播的角度,提出了一种超前计算式的电路结构^[11],如图 2 所示。该结构由若干个仲裁单元构成,最底部是根节点,向上与四个子节点相连,而每个子节点又向上生长出新的子节点,最上层是叶节点,负责信号的采集和结果的输出。父节点生长出新节点的数量,决定了这个系统的规模,生长数量越多,级数越少,相应的仲裁单

元越复杂,反之,生长数量越少,级数越多,仲裁单元越简单。该结构工作流程如下:

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。

图中显示所有路径的传播长度都是相同的,仲裁单元也是相同的电路结构,因此可以精确地控制延时差,该结构被大量应用在一些商业处理器中的发射队里模块。然而,随着队列容量的不断增长,待测信号数量已经超过上百位,而该结构是不具备压缩编码能力的,输出端口会消耗大量的总线资源,金属布线延时会更加明显。

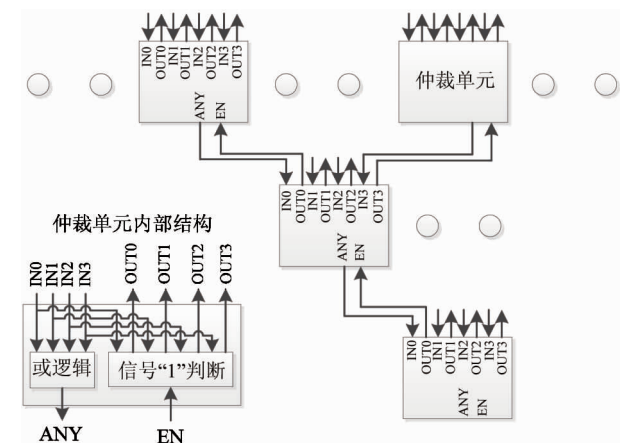


图2 超前计算式电路结构

2 稀疏树 LOD 结构介绍

研究 LOD 电路细节时就会很清楚地发现,并没有一种与位数无关的 LOD 电路,真正的延时至少随位数(更好的情况是位数的 N 次方根)呈线性增加。这一电路的大扇入使它在求值时极慢,如果用简单的逻辑门组合来实现,则又要求有多个逻辑层次。这两种情况都会使传播延时增加。此外,某些信号的扇出过于庞大,也会使整个电路变得更慢,例如,最低位的输入信号 I_0 ,出现在后续的每一位表达式

中,因此这根线上的电容是很大的,同样增加了传播延时。另外,考虑前文所介绍的几种电路结构,不难看出,单元库数量越小、复用度越高的设计,其输出延时差越小。

基于以上的分析,在设计更快速的 LOD 电路时,需要考虑以下三点:(1)避免逐级传播效应;(2)相同的逻辑级数;(3)相似的功能模块。

为了避免逐级传播效应,首先分析 LOD 逻辑表达式。在 N 位输入的 LOD 逻辑中存在下列关系^[12]:

$$\overline{Y_0} = I_0 + \overline{I_1} \cdot I_2 + \overline{I_1 + I_3} \cdot I_4 + \cdots + \overline{I_1 + I_3 + \cdots + I_{2n-1}} \cdot I_{2n} \quad (1)$$

$$\overline{Y_1} = (I_0 + I_1) + (I_2 + I_3) \cdot (I_4 + I_5) + \cdots + (I_2 + I_3) + \cdots + (I_{4n-2} + I_{4n-1}) \cdot (I_{4n} + I_{4n+1}) \quad (2)$$

$$\overline{Y_m} = (I_0 + I_1 + \cdots + I_{2^{m-1}}) + \cdots + \overline{(I_{2^n} + I_{2^n+1} + \cdots + I_{2^{n+1}-1})} + \cdots + \overline{(I_{2^{n+1}-2^n} + \cdots + I_{2^{n+1}-1})} \cdot (I_{2^{m+1}n} + \cdots + I_{2^{m+1}n+2^m-1}) \quad (3)$$

其中 $m = \log_2 N$ 。

如果把上式中每个括号内的或运算视为一个因子的话, $\overline{Y_m}$ 的表达式就可以得到简化。将所有的待测信号根据其位置进行分类并放入不同的集合里,同一个集合内所有元素的或值称之为传播信号 $P_{i,j}$ ($P_{i,j} = \underbrace{I_{2^i} + I_{2^i+1} + \cdots + I_{2^{i+1}-1}}_{2^i \text{个数相加}}$),集合之间的运算会在下文详述。将 $P_{i,j}$ 代入上述 $\overline{Y_m}$ 的表达式,得到

$$\overline{Y_m} = P_{m,0} + \overline{P_{m,1}} \cdot P_{m,2} + \overline{P_{m,1} + P_{m,3}} \cdot P_{m,4} + \cdots + \overline{P_{m,1} + P_{m,3} + \cdots + P_{m,2n-1}} \cdot P_{m,2n} \quad (4)$$

将上式中的正逻辑定义为“生成信号” G , 反逻辑定为“取消信号” D , 得到新的表达式:

$$\begin{aligned} \overline{Y_m} &= G_{m,0} + \overline{D_{m,1}} \cdot G_{m,2} + \overline{D_{m,1}} \cdot \overline{D_{m,3}} \cdot G_{m,4} \\ &+ \cdots + \overline{D_{m,1}} \cdot \overline{D_{m,3}} \cdot \cdots \cdot \overline{D_{m,2n-1}} \cdot G_{m,2n} \\ &= G_{m,0} + \overline{D_{m,1}} \cdot G_{m,2} + \overline{D_{m,1}} \cdot \overline{D_{m,3}} \cdot (G_{m,4} \\ &+ \overline{D_{m,5}} \cdot G_{m,6}) + \cdots + \overline{D_{m,1}} \cdot \overline{D_{m,3}} \\ &\cdot \cdots \cdot \overline{D_{m,2n-3}} \cdot (G_{m,2n-2} + \overline{D_{m,2n-1}} \cdot G_{m,2n}) \end{aligned} \quad (5)$$

通过不断地提取公共项 $\overline{D_m}$, 最终可得

$$\overline{Y} = G + \overline{D} \cdot G' \quad (6)$$

这一高层次的模型包含了某些隐含的依赖关系,对于一个 N 位输入的 LOD 逻辑,其输出结果受两种情况的影响:生成信号若为“1”,则置结果为“1”;取消信号若为“1”,则取消之后所有信号的影响。不同输出位的“生成信号”和“取消信号”由各自的“传播信号”产生,例如,对于输出结果的最低位,其生成信号为偶数输入位,取消信号为奇数输入位(首位默认为第零位)。

由此,整个编码逻辑可以分为两个步骤,参照图 3,不同形状图型对应不同的输出端口。

步骤 1:信号传播,产生不同输出位的“生成信号” G 和“取消信号” D , 见图中虚线部分;

步骤 2:逻辑计算,通过 G 、 D 求出最终值,见图中实线部分。

◆ 信号传播

观察 $P_{i,j}$ 的表达式可知,集合内的元素进行的是或运算,若是两个集合完全互斥,则这两个集合同样可以进行或运算,特别地,若是该两个集合内的元素完全相同于第三个集合,则它们之间存在如下等价关系:

$$\begin{aligned} P_{i,j} &= I_{2^i} + I_{2^i+1} + \cdots + I_{2^{i+1}-1} \\ &= I_{2^{2^i-1}} + I_{2^{2^i-1}+1} + \cdots + I_{2^{2^i}-1} \\ &= P_{i-1,2^i} + P_{i-1,2^i+1} \end{aligned} \quad (7)$$

下标 j 代表了集合内元素的位置,下标 i 代表了集合内元素的个数 (2^i 个),对比图 3 中虚线表示的信号传播过程, i 也可以认为是该传播信号的逻辑级数,本文将这种两个集合间的或运算称为“块操作”。

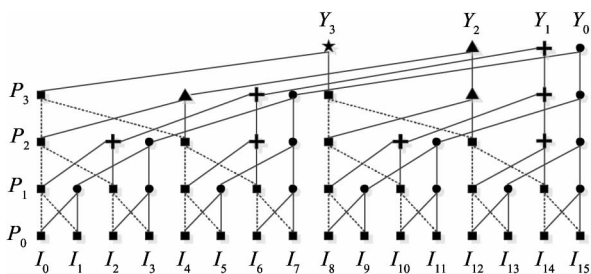


图 3 稀疏树结构示意图

◆ 逻辑计算

在推导 Y_m 的过程中,其实已经得到了 G 和 D 的表达式:

$$G_k = G_{k-1} + \overline{D_k} \cdot G'_{k-1} \quad (8)$$

$$D_k = \overline{D_{k-1}} \cdot \overline{D'_{k-1}} \quad (9)$$

可见,生成信号 G 和取消信号 D 仅与自身相关,可以把 G 和 D 也组织成树形结构。 $G_{i,j}, D_{i,j}$ 分别表示一组位(从第 i 位到第 j 位)的生成信号函数和取消信号函数,本文称之为“结生成信号”和“结取消信号”。值得注意的是,新的表达式和原表达式完全等价,只是产生和取消信号由“结产生”和“结取消”信号所代替。现在可以引入一个新的布尔运算,称为点操作($\cdot$),这一运算仅针对 $(G_{i,j}, D_{i,j})$ 并且有不同的运算对象:(1)传播信号或者(2)结信号。这是由于在逻辑计算开始时,运算对象为上一步信号传播的结果,逻辑计算开始后,运算对象为点操作的结果。以下是两种情况下的实例:

$$\begin{aligned} (G_{3,2}, D_{3,2}) &= (G_3, D_3) \cdot (G_2, D_2) \\ &= (P_3, \overline{P_3}) \cdot (P_2, \overline{P_2}) \end{aligned}$$

$$(G_{3,0}, D_{3,0}) = (G_{3,2}, D_{3,2}) \cdot (G_{1,0}, D_{1,0})$$

点操作服从结合律,不符合交换律。利用点操作的结合律,可以构成一个树形结构来计算所有位置上的结信号,但大部分的结信号并非是必须的,图 3 中实线部分是一个基数为 2 的稀疏树,不同形状图型代表了不同输出端口逻辑运算过程,从图中可以看出,对于一个 16 位的 LOD 最低输出位,仅仅需要 15 个结信号,而最高输出位甚至只需要一个结信号。省去的运算降低了内部扇出,继而降低了整个电路的延时。

以上就是稀疏树 LOD 电路的编码过程,步骤 1 中信号传播为步骤 2 中的逻辑计算提供操作数,也可以看做是两只同时生长的二叉树,信号传播的终值为 LOD 电路的标志信号,逻辑计算的终值为电路的输出信号。该结构包含以下几个优点:

(1) 逻辑级数少

N 位稀疏树 LOD 电路只需要 $\log_2 N$ 级逻辑门就可以计算出结果,少于遗传算法得到的电路。对于递归结构和超前计算结构,表面上同样可以达到 $\log_2 N$ 的逻辑级数,然而,由于包含选择器或者多输入与非门,每级的逻辑延时都会大于稀疏树结构。若是以 FO4 (fanout-4 inverter, 扇出为 4 的反相器) 来评估,则递归结构和超前计算结构逻辑级数实际

上大于 $\log_2 N$ 。

(2)相连度低

每一个“点操作”的扇出均为 1, 每一个“块操作”需要同时为逻辑计算提供操作数, 因此扇出为 2, 所有运算扇入都不大于 3, 可以用最基本的逻辑门来搭建, 极大地降低了负载电容引起的延时。

(3)计算量相同

仔细分析图 3 可知, 对于不同的输出端口, 其信号传播和逻辑计算所花费的时间比例是不同的, 但是总的逻辑级数是相同的。例如, 输入信号和 Y_0 之间, 进行了一次块操作和四次点操作, 共五次运算; 输入信号和 Y_3 之间, 进行了四次块操作和一次点操作, 同样是五次运算。相同的计算量对降低输出延时差起到了关键作用。

3 LOD 电路实现

为了获取更高的性能, 本文采用动态电路来实现该结构中的点操作和块操作。图 3 中方块代表了

计算传播信号的或门, 圆点代表计算结生成信号和结取消信号的两个门, 电路如图 4 所示。

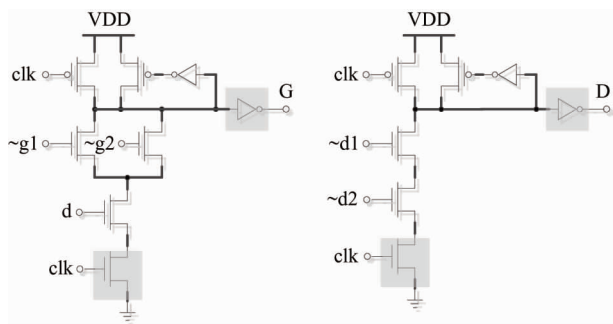


图 4 “点操作”的电路实现

本设计为预充电电路额外增加了一个保持器, 其作用有两个: (1) 抵抗漏电, 避免漏电导致的非稳态电压, 提高电路可靠性; (2) 降低电荷共享效应的影响^[13]。该效应是指, 求值电路出现了多个晶体管级联的情况, 在某些激励下, 求值晶体管的源漏电容会共享输出端的电荷, 导致输出发生错误翻转, 如图 5 所示。

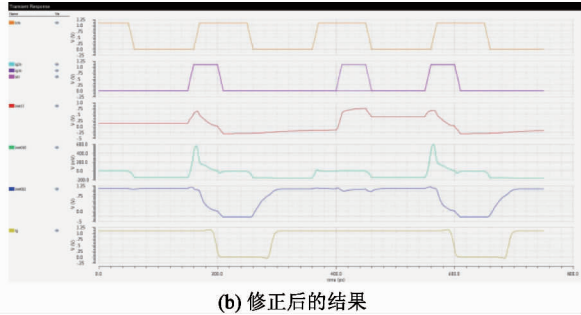
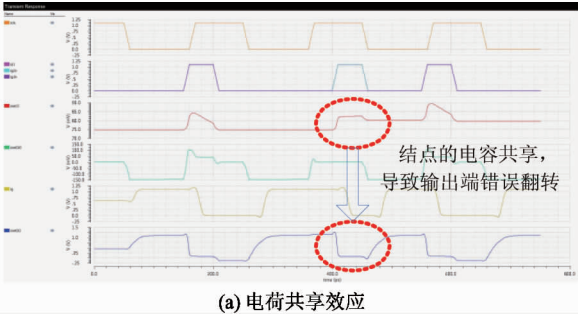


图 5 电荷共享效应对“点操作”的影响

动态电路在输出端引入的静态反相器, 可以在预充电期间, 将输出泄放至 0, 有效地避免因驱动下一级的 NMOS 管而出现的预充阶段电路漏电, 在求值阶段动态门有条件的放电, 则反相器有条件的完成 $0 \rightarrow 1$ 的翻转。但是引入静态反相器会额外地增加电路延时, 为了避免这个问题, 本文采用 np-CMOS 动态电路, 即在 LOD 电路的相邻两级分别采用 N 型动态逻辑和 P 型动态逻辑。前文提到结生成信号和结取消信号的两个门, 用 P 型逻辑实现的电路如图 6 所示。

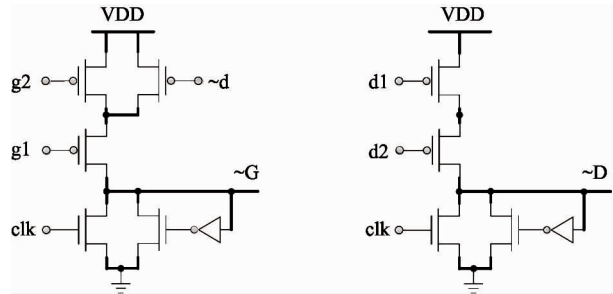


图 6 “点操作”的 P 型动态电路

为了减少电路延时, 可以将求值晶体管和电源之间的开关省去, 这是因为每一级的 N 型或 P 型动

态电路,在预充阶段,其下拉或上拉网络都是关闭的(前一级的预充结果所致)。这种方法有一个缺点,就是在预充阶段的开始时刻,前一级的预充还未达到求值网络内 MOS 管的阈值,存在一条漏电通路。为了避免这一短路电流,每一级的驱动时钟 CLK 需要比前一级延时一些时间,这就需要在时钟网络上额外增加一个 DELAY 单元,来控制两级时钟间的偏差。相邻两级之间的点操作是如图 7 所示级联在一起的。

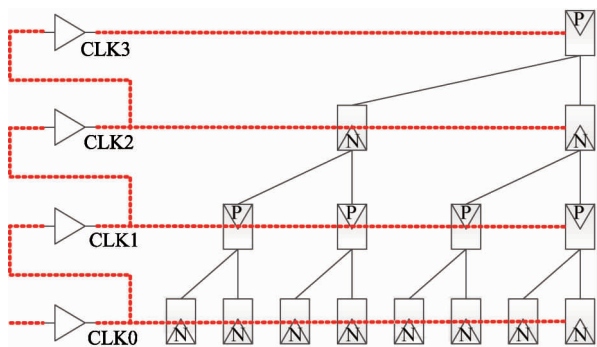


图 7 NP 逻辑连接示意图

考虑到在网络起始端时钟扇出比较多,时钟负载大,需要建立时钟树,因此可以直接将驱动不同层级单元的时钟树,改造成 DELAY 单元。增加的 DELAY 单元,除了能够避免短路,还有一条额外的好处,就是严格控制每一级的输出时间,彻底消除了因“点操作”和“块操作”逻辑不同而引入的延时不同。具体来说,不管输入数量、逻辑级数是多少,在逻辑计算的最后一步,所有的信号受同一个时钟控制开始运算,且输出的每一位最后一步进行的都是“点操作”,因此,所有信号的输出是完全同步的。

4 电路仿真与验证

本文采用 SMIC 40nm 工艺搭建了不同输入端口数的 LOD 电路,使用仿真工具 Hspice 进行前仿。其中,128 端口 LOD 电路的激励向量高达 1.8×10^{19} 个,无法遍历结果对比真值表,于是将验证过程分为两部分:第一部分,仿真动态电路搭建的各个门,验证“点操作”和“块操作”功能正确;第二部分,采用

伪随机激励向量来验证稀疏树网络。由前文所述 Y_m 的表达式可知, Y_m 的值仅与 P 的正反逻辑相关,而 P 是按其表达式进行分类的不同待测信号的集合,因此,在保证 P, G, D 逻辑运算正确的前提下,只要观察首个“1”落入不同集合时输出结果的变化,就能验证网络连接是否正确。对于输出端口 Y_2 , 其 P 值的集合分别为 $\{I_0, I_1, I_2, I_3\}, \{I_4, I_5, I_6, I_7\} \dots \{I_{4i}, I_{4i+1}, I_{4i+2}, I_{4i+3}\}$ (其中 $i \in N$), 只要保证首个“1”落在不同的集合内至少一次即可,例如:“0100 0110 1000 0001 ...”为一组满足条件的激励向量。同理,对于端口 Y_1 , 其 P 值集合为 $\{I_0, I_1\}, \{I_2, I_3\}, \{I_4, I_5\} \dots$, 激励向量可选择“01 01 10 ...”。

本设计中引入的 DELAY 单元,是控制整体电路静态功耗和性能的重要部分。若是 DELAY 单元的延时过短,后一级预充上升沿和前一级重叠,则出现漏电通路,波形稳定时间增长,降低电路的鲁棒性;若是 DELAY 单元的延时过长,两级间的冗余时间,会降低电路频率。多次实验结果显示,前后级预充上升沿在同一时刻到达 PMOS 和 NMOS 阈值电压时,可在提高电路性能的前提下,最大程度保证电路的可靠性。

根据仿真结果绘制图 8,横轴代表输入端口数量,纵轴代表延时,单位是皮秒(ps)。首先观察各结构的最大输出延时,在待测信号较少的情况下,稀疏树结构的性能提升并不明显,但是,随着待测信号的增多,遗传算法和超前计算中的冗余逻辑增加,器件扇出增大,线负载加重,时序迅速恶化。而稀疏树结构,仅“点操作”和“块操作”数量增加,电路内部扇出并没有发生任何改变。在待测信号为 128 位时,稀疏树电路的最长路径比其他两种电路少了 20%。

再观察图 8 中的折线图部分,折线图反映的是不同电路输出端口延时差,其纵坐标为右侧辅助坐标。从图中可以看出,超前计算式结构相比于递归结构的输出延时差有很大的降低,但是仍然随着待测信号的数量在不断增长。稀疏树电路受同步时钟控制,延时差不随待测信号数量变化。实际仿真中,最长路径和最短路径间差值小于 1ps。

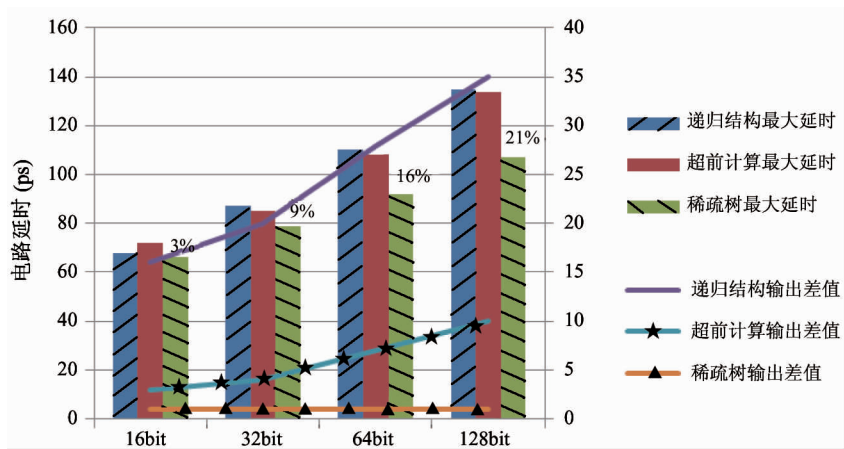


图 8 针对不同输入位数的仿真结果

5 结 论

本文所提出的电路结构,通过商业工具仿真证明了其性能的优越性,在解决大规模电路输出不同步方面,表现尤为突出。电路中的几种功能单元皆可复用,极大地缩短了设计周期。此树形结构不仅适用于 LOD 电路,对于存在信号传递性质的电路,如加法器、计数器等,都可以通过递归表达式,构建稀疏树。对于一些 P 值、 D 值、 G 值运算差别较大的情况,通过合并 P 值集合,即增加“块操作”的运算量,降低“点操作”的运算量,来建立分支更多的树形结构,可以进一步提高性能。

参考文献

- [1] Pappalardo F, Visalli G, Scarana M. An application-oriented analysis of power/precision trade-off in fixed and floating-point arithmetic units for VLSI processors. In: Proceedings of the LASTED International Conference on Circuits, Signals, and Systems, Clearwater Beach, USA, 2004. 416-421
- [2] Suzuki H, Morinaka H, Makino H, et al. Leading-zero anticipatory logic for high-speed floating point addition. *IEEE Journal of Solid-State Circuits*, 1996, 31(8):1157-1164
- [3] Tendler J M, Dodson J S, Fields J S, et al. POWER4 system microarchitecture. *IBM Journal of Research & Development*, 2002, 46(1):5-25
- [4] Brown M D, Stark J, Patt Y N. Select-free instruction scheduling logic. In: Proceedings of the 34th ACM/IEEE International Symposium on Microarchitecture, Austin, USA, 2001. 204-213
- [5] Wijeratne S, Siddaiah N, Mathew S, et al. A 9GHz 65nm Intel Pentium 4 processor integer execution core. In: Proceedings of the IEEE International Solid-State Circuits Conference, Digest of Technical Papers. San Francisco, USA, 2006. 353-365
- [6] Mooseop K, Jong W H. Compact hardware architecture for first one detector using priority-based uniform partition. In: Proceedings of International Conference on Ict Convergence (ICTC), Jeju Island, Korea, 2012
- [7] Choi B, Walker D M H. Timing analysis of combinational circuits including capacitive coupling and statistical process variation. In: Proceedings of VLSI Test Symposium, 2000. 49-54
- [8] Kunaraj K, Seshasayanan R. Leading one detectors and leading one position detectors-an evolutionary design methodology. *Canadian Journal of Electrical & Computer Engineering*, 2013, 36(3):103-110
- [9] Kogge P M, Stone H S. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, 1973, C-22(8):786-793
- [10] Dimitrakopoulos G, Galanopoulos K, Mavrokefalidis C, et al. Low-power leading-zero counting and anticipation logic for high-speed floating point units. *IEEE Transactions on Very Large Scale Integration Systems*, 2008, 16(7):837-850
- [11] Buyuktosunoglu A, El-Moursy A, Albonesi D H. An ol-

- dest-first selection logic implementation for non-compacting issue queues. In: Proceedings of the 15th Asic/soc Conference, Rochester, USA, 2002. 31-35
- [12] Dimitrakopoulos G, Galanopoulos K, Mavrokefalidis C, et al. Low-power leading-zero counting and anticipation logic for high-speed floating point units. *IEEE Transactions on Very Large Scale Integration Systems*, 2008, 16 (7):837-850
- [13] Rabaey J M. Digital Integrated Circuits: A Design Perspective. Prentice Hall, 1996. 274-278

A synchronous output leading-one-detector circuit based on sparse tree

Liu Zhen * * * * *, Wang Jian * * * * * , Wang Hao * * * * *

(* 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 Limited , Beijing 100095)

Abstract

To reduce the delay and the arrival time difference of a leading-one-detector circuit, a sparse-tree based leading-one-detector structure and its dynamic circuit implementation method were presented. Two new kinds of logical Boolean operations were defined by recursive expression to form the nodes of the sparse tree. The precalculated clock's skew in the pre-charge stage can be accurately controlled, so the leakage power and the arrival time difference can be simultaneously controlled. In consideration of the various numbers of input ports, the extending network by connecting same spot units shortens the time cost on designing. There is less logic level and fan-out in the structure presented. The whole circuit was verified by the pseudo random test vector. This technique could reduce the critical path length by 20% and keep the arrival time in 1 picosecond.

Key words: leading one detector, spot operation, sparse tree, synchronism output, dynamic circuit