一种提高芯片良率的时序电路缓冲器插入算法

戢小亮[1],佟星元[1],吴睿振[2],杜 鸣[3]
(1. 西安邮电大学电子工程学院,陕西西安 710121;2. 西安电子科技大学博士后流动站,陕西西安 710071;
3. 西安电子科技大学微电子学院,陕西西安 710071)

摘 要: 针对集成电路工艺参数波动影响芯片良率的问题,提出一种提高芯片良率的时序电路缓冲器插入算法. 该算法通过蒙特卡罗仿真模拟流片后的芯片,确定时序电路中可插入缓冲器的最佳位置,在保证良率的前提下,降低了面积及成本损耗. 算法经过 ISCAS89 的基准电路和 TAU2013 的电路进行仿真验证,结果表明插入缓冲器的数量小于等于触发器数量的 1%,良率提高高达 35. 98%.

关键词: 工艺参数波动; 芯片良率; 缓冲器
中图分类号: TP391. 41 **文献标识码:** A **文章编号:** 0372-2112 (2018)12-2964-06
电子学报 URL: http://www. ejournal. org. cn **DOI:** 10. 3969/j. issn. 0372-2112. 2018. 12. 020

A Sequential Circuit Buffer Insertion Algorithm for Yield Improvement of Chips

JI Xiao-liang[1], TONG Xing-yuan[1], WU Rui-zhen[2], DU Ming[3]
(1. *School of Electronic Engineering, Xi' an University of Posts & Telecommunications, Xi' an, Shaanxi* 710121, *China*;
(2. *Postdoctoral Station of Xidian University, Xi' an, Shaanxi* 710071, *China*;
(3. *School of Microelectronics, Xidian University, Xi' an, Shaanxi* 710071, *China*)

Abstract: Chip Process variations cause yield degradation after manufacturing. To improve yield, the sequential circuit buffer insertion algorithm for yield improvement of chips is proposed. The locations of buffers are determined by sequential circuit simulating chips after manufacturing based on Monte Carlo simulations. The proposed method not only maintains a good yield improvement, but also reduces area cost. By using ISCAS89 benchmarks and TAU 2013 circuits, the simulation results show that the number of inserted buffers is no larger than 1% of the number of flip-flops in the circuits, and the yield is improved up to 35. 98%.

Key words: process variations; yield; buffer

1 引言

在摩尔定律的指引下,芯片特征尺寸不断减小,芯片上可集成的功能模块增多,设计的复杂度也逐渐上升. 工艺参数波动对芯片良率的影响越来越明显,因此有时不得不进行过度设计,以增加时序阈度提高芯片良率. 流片后,工艺参数的波动固定,针对参数的波动和具体的芯片工作需求,对特殊延时器件进行调整,即可提高芯片良率.

图 1 是一种广泛应用的插入延迟缓冲器调整时序以提高流片后芯片良率的技术[1~4]. 这种技术是利用插入带有触发器的延迟缓冲器来实现时序阈度调整的. 以增加一定面积为代价,实现芯片良率的提高. 这种技术是时序电路中插入缓冲器以提高芯片良率的经典方法.

近年来,出现了很多针对具有可配置缓冲器的电路的时序分析和优化方法. 文献[5]提出了针对芯片良率分析,基于蒙特卡罗算法的框架模型. 文献[6]分析了不同存储器在电路流片中对延时、面积和功耗等方面的影响. 文献[7]在文献[6]的基础上建立了静态时序电路的分析模型,并基于蒙特卡罗算法与实际电路比较. 文献[8]提出了一种插入缓冲器以提高电路时序稳定性的方法,最高可以减少 87. 3% 的时钟抖动.

收稿日期:2017-11-21;修回日期:2018-04-13;责任编辑:马兰英
基金项目:国家自然科学基金(No. 61674122);新一代宽带无线移动通信网科技重大专项(No. 2016ZX03001003-006);陕西省创新人才推进计划项目(No. 2017KJXX-46)

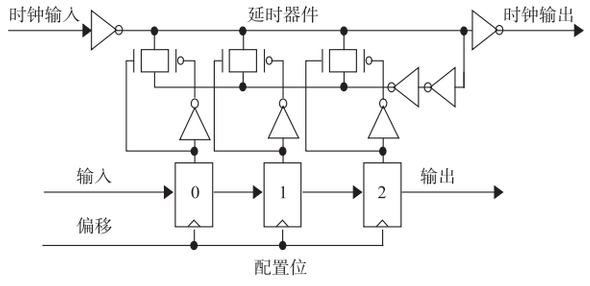


图1　文献[1]延迟缓冲器

文献[9]提出了一种流片后延迟缓冲器插入方法,以面积损耗1% ~2%为代价降低了电路频率迭代对时序的影响. 文献[10]提出了一种在时钟树上插入时钟游标装置(CVD)的算法.

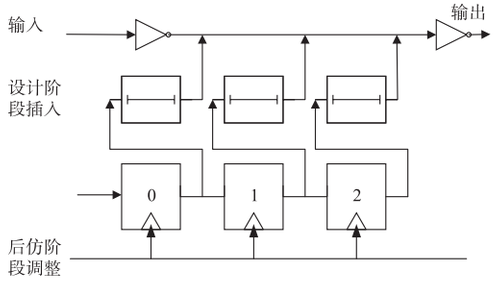


图2　文献[10]时钟游标装置

如图2所示,文献[10]在构建时钟游标装置时,首先使用固定延迟的缓冲器件在设计阶段插入电路. 然后利用触发器,基于得到的实际工作情况数据配置触发器,以提高芯片良率.

文献[11]中,提出了一种通过搜索配置树并结合图论的配置延迟缓冲器算法. 算法将电路的时序构建为树形结构. 算法寻找需要优化的时序路径,插入延迟缓冲器以提高良率,平均良率提高23%.

本文提出了一种在电路设计阶段为了提高芯片良率的时序电路延迟缓冲器插入算法. 该算法中,我们利用蒙特卡罗仿真的方法模拟流片后的芯片,在每一次采样中使用尽可能少的可配置的延迟缓冲器. 采样结束后,只把对芯片良率起关键作用的延迟缓冲器保留下来. 这种方法可以得到最佳的可配置延迟缓冲器大小及其插入位置,而不用再进行额外的中间步骤建模. 避免了在统计优化里的启发式算法.

2　具有延迟缓冲器电路时序约束分析和建模

图3解释了带有配置缓冲器的时序约束条件,图中两个触发器的中间部分为组合逻辑.

假设时钟信号在参考时间0时刻变化,那么时钟信号在触发器 i 和 j 上改变的时间分别是 x_i 和 x_j. 为了满足建立时间和保持时间,因此需满足以下公式:

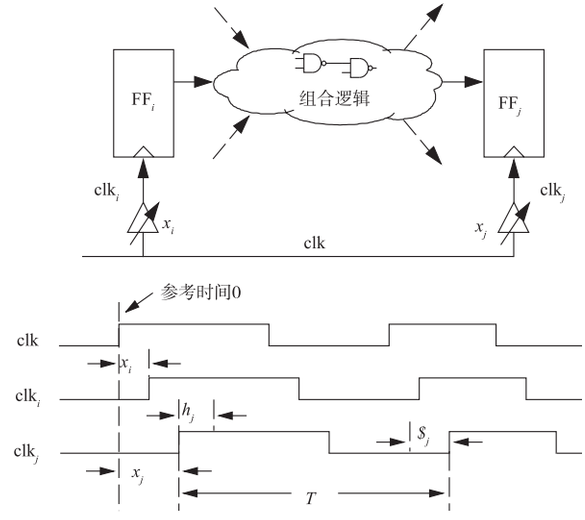


图3　插入缓冲器的电路时序

$$x_i + \overline{d}_{ij} \le x_j + T - s_j \tag{1}$$

$$x_j + \underline{d}_{ij} \ge x_i + h_j \tag{2}$$

其中 $\overline{d}_{ij}$ 和 $\underline{d}_{ij}$ 是触发器 i 和 j 之间组合逻辑的最大和最小延迟. s_j 是触发器 j 的建立时间,h_j 是保持时间,T 是时钟周期. 可配置缓冲器在公式(1)和(2)中引入了两个延迟变量. 如果没有它们,这两个不等式就是数字电路经典的时序约束条件[12].

为降低面积损耗,可配置的缓冲器延迟越小越好. 假设这个可调整缓冲器 i 的调整值的下限是 r_i,缓冲器的延迟范围是 τ_i. 延迟缓冲器的延迟范围是:

$$r_i \le x_i \le r_i + \tau_i \tag{3}$$

我们对可配置缓冲器的延迟范围设置为相对于0不对称的,这样可以达到最大的灵活度. 此外,基于45nm Nangate工艺库实际参数情况,得到这些延迟缓冲器的延迟离散值.

这些变量可以使用文献[12]中的标准形式进行分解并与 $\overline{d}_{ij}$,$\underline{d}_{ij}$,s_j 和 h_j 合并. 因此我们假设延迟缓冲器的延迟可以通过公式(3)设置为固定值.

在实际电路中,可配置缓冲器的数量是有限制的,为了实现一个合理的良率提升,插入延迟缓冲器的方法可以描述为:选择一定数量的触发器,在其周围插入尽可能少的可配置缓冲器以提高芯片的良率,并保证面积损耗最小.

除此之外,可调整缓冲器的延迟范围每个应该不同,并且可为负. 在本文中,我们利用基于整数线性规划的蒙塔卡罗仿真得到延迟缓冲器的位置和范围.

3 基于采样的延迟缓冲器插入算法

为了得到应该插入延迟缓冲器的位置,我们利用一种基于采样的方法去解决参数波动带来的问题. 整体流程如图4所示. 算法需要输入的变量有:电路结构,统计的门延迟以及可配置缓冲器的规格,如缓冲器的延迟范围和离散大小. 我们提出的方法基于给定的时钟周期 T,在提高芯片良率的前提下,得到尽可能准确的缓冲器位置和尽可能小的插入数量.

在所提出的方法中,我们假设每一个触发器都有一个可以配置的延迟缓冲器插在其周围. 首先我们对不等式(1)和(2)中的统计变量进行采样,在保证芯片工作频率的前提下,最小化其插入延迟缓冲器的数量. 这里的采样可以理解为流片后的每一个芯片. 如果有足够的采样,那么延迟缓冲器的位置便可以通过采样得到.

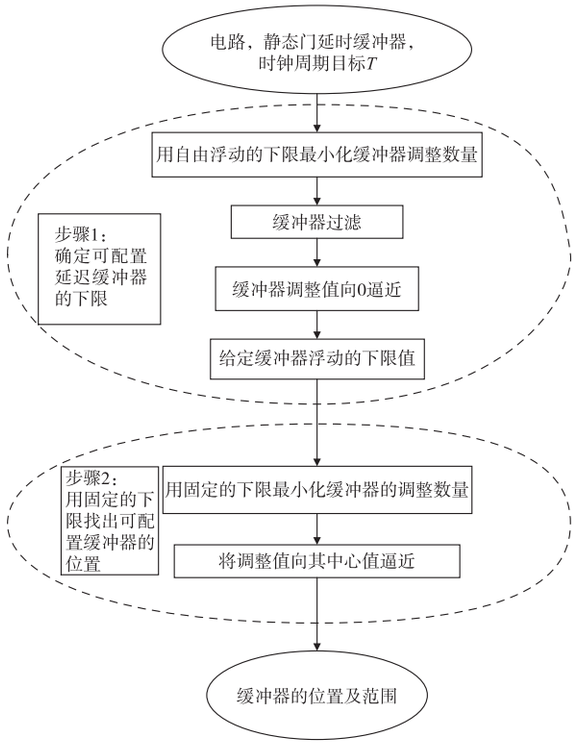


图4 基于采样的延迟缓冲器插入算法

如图4所示算法包括两个主要步骤. 步骤1,我们允许延迟缓冲器的延迟下限自由变化. 因为在(3)中 r_i 的限制是未知的. 例如,每一次采样中,我们最小化所需要的延迟缓冲器的数量并把可调整的值向0逼近. 压缩这些可调整值以缩小缓冲器延迟的范围. 最终选择包括最多延迟调整的范围以决定可配置延迟缓冲器的下界.

步骤2,延迟缓冲器的下界确定后,我们重新进行采样. 对于每一次采样,最小化需要的延迟缓冲器的数量,并且使它们的调整值向中心值逼近. 这一步,因为下限已经确定,取调整值的均值就可以反映出调整值的趋势. 因此,压缩调整值以接近其均值的方法可以缩小延迟缓冲器的面积损耗. 最后,延迟缓冲器的面积由最小和最大调整值决定.

3.1 确定可配置延迟缓冲器的下限

电路结构和延迟分布决定了延迟缓冲器的位置,我们基于采样的仿真得到插入延迟缓冲器的位置及其下界的初步信息. 具体方法为,基于每一次采样得到的电路中的延迟信息,建立一个整数线性规划模型,在可配置范围的下界可自由变化前提下,最小化可以配置的延迟缓冲器数量. 如果采样数目足够多,可以更准确得到对提高良率最关键的延迟缓冲器的位置.

为了对延迟缓冲器调整与否的问题进行建模,我们对第 i 个缓冲器利用一个二进制变量 c_i,$c_i=1$ 表示第 i 个缓冲器被调整了,因此 $x_i\neq 0$;反之 $c_i=0$ 时 $x_i=0$. 根据文献[13],这个约束可以转化为:

$$x_i \le c_i M \tag{4}$$

$$-x_i \le c_i M \tag{5}$$

其中,M 是一个很大的数值. 如果 x_i 大于0,那么 c_i 必须等于1. 如果 x_i 小于0,那么 c_i 也设置为1. c_i 设置为0的情况仅仅在 x_i 为0的情况下,即是说无需调整.

从上面的分析可见,c_i 是第 i 个缓冲器的调整数量的上界,因为 c_i 在 $x_i=0$ 的时候也可以取1. 相应的,所有 c_i 的和是对于每个采样得到的调整数量的上界. 这个上界可以表示为:

$$c_{sum}=\sum_i c_i,\quad i\in I \tag{6}$$

其中 I 是所有可调整缓冲器标号的集合. 如果我们最小化这个上界,那么相应的调整值也会被最小化.

除了最小化缓冲器总数量 c_{sum},我们也希望调整值不要太分散,这样的话它们就可以被同一个范围窗口 τ_i 涵盖. 为了达到这个目标,我们最小化调整值和0之间的距离. 因此,第二个优化目标是 $\sum_{i=1}^{I}|x_i|$,即是调整值与0距离的总和.

这个优化问题现在有两个目标,可以通过一个权重表达式结合起来. 在我们的建模中,缓冲器的数量优化的优先级最高,因为越少的缓冲器数量意味着越小的面积以及更简单的布局设计. 因此,我们把这个优化问题分为两个. 第一个优化目标 c_{sum},即每次采样需要的最小缓冲器数量. 得到的数值作为最小化 $\sum_{i=1}^{I}|x_i|$ 的一个约束条件. 在完成第一次优化后,我们删去那些很少需要调整的缓冲器,这样的话第二次的优化时间将会大大降低.

为得到可配置延迟缓冲器的下限,具体的算法可分为四步:

(1)用自由浮动的下限最小化缓冲器调整数量.

第一次优化问题可以描述如下:

$$\text{对于每一次采样:}m_k \in M \tag{7}$$

$$\text{最小化:}c_{sum} \tag{8}$$

$$\text{约束:建立时间和保持时间约束(1),(2)} \tag{9}$$

$$\text{调整缓冲器约束:(5),(6)} \tag{10}$$

$$\text{缓冲器范围约束:(3)} \tag{11}$$

$$r_i \leq 0 \text{ 且 } r_i + \tau_i \geq 0 \tag{12}$$

其中 m_k 代表了第 k 个采样的芯片,M 是采样的总和,其基数是仿真中采样的数量. 在上述模型中,缓冲器延迟下限 r_i 是变量,并由解算器决定. 约束式(12)要求延迟范围包括 0 在内. 在每一次解决完优化问题式(7)~(12)以后,我们标记第 m_k 采样的缓冲器调整数目是 n_k.

(2)缓冲器过滤.

因为上述优化问题试图尽量减小需要的缓冲器的数量,而许多缓冲器调整次数很少甚至没有调整. 如图 5 所示,其中节点代表了缓冲器,连接线代表了触发器之间的逻辑连接.

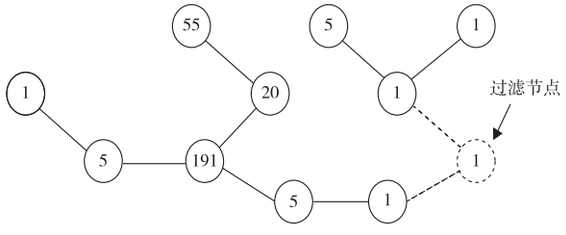


图5　最小化调整缓冲器数量

图中的数目代表了在 $|M|$ 次采样中对应缓冲器的调整次数. 我们在算法中去掉那些调整次数不大于 1 的以及与重要节点没有相连的缓冲器. 重要节点定义为调整次数大于一定数量的节点,在实验中我们设置为 5,总共采样次数为 10000. 例如,在图 5 中,我们去掉那些虚线描述的节点. 这种过滤不仅仅可以加速算法,也通过图的分解降低了问题的空间.

(3)缓冲器调整值向 0 逼近.

优化问题式(7)~(12)仅仅降低了在每一次采样中调整的数目. 因为在不同采样中,优化问题有不同解,所以所有采样中缓冲器的调整值将会分布在一个很宽的范围,图 6 给出了这种范围,其中 x 轴代表了在所有采样中缓冲器的调整大小,y 轴代表了这次调整值出现的次数.

为了压缩这些分散的调整值的范围,我们最小化它们的绝对值. 最后的这个约束保证了在每次采样中

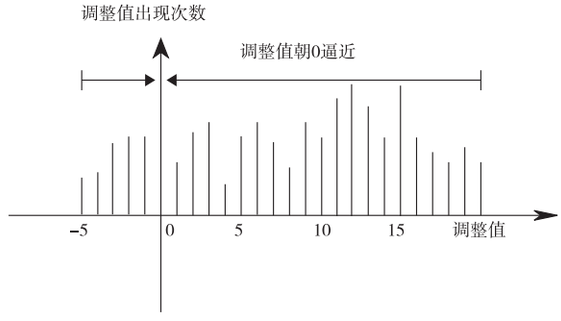


图6　所有采样下的缓冲器调整值

调整数目都是最优的.

$$\text{对于每次取样 } m_k \in M \tag{13}$$

$$\text{最小化 } \sum_{i=1}^{I} |x_i| \tag{14}$$

$$\text{约束条件为(9)~(12)} \tag{15}$$

$$c_{sum} \leq n_k \tag{16}$$

其中,目标方程(14)可以利用文献[14]中方法转化为线性的.

(4)给定缓冲器浮动下限值.

在调整值尽量向 0 逼近以后,我们将用一个最大的范围窗口 τ_i 涵盖所有的调整值,图 6 描述了这个过程,范围窗口在 x 轴上滑动. 因为 y 轴代表了在所有采样中相应的调整值,范围窗口涵盖的调整值就是在这个窗口内所有调整值的总和. 为了提高芯片的良率,我们选用可以涵盖最大数量调整值的范围窗口. 这样的话,调整值范围的下限由最小值决定.

3.2　用固定的下限找出可配置延迟缓冲器

在缓冲器延迟下限确定后,缓冲器的位置和大小应该基于下限重新估计. 具体分为两步:

(1)用固定的下限最小化缓冲器的调整数量.

在缓冲器下限固定后,式(3)和式(12)中的 τ_i 变量将是固定值. 因此,我们重新执行最小化问题式(8)~式(12)去找到更为准确的缓冲器调整值. 在算法实现中,设置其阈值为 0.1%.

(2)将调整值向其中心值逼近.

在用固定下限运行完所有的仿真后,调整值将固定在一个范围窗口内. 这跟之前的缓冲器调整值向 0 逼近是一个问题,同样的,我们试图将缓冲器调整值向其均值 $x_{avg,i}$ 逼近,如图 7 所示.

这一步的优化问题描述如下:

$$\text{对于每次采样 } m_k \in M \tag{17}$$

$$\text{最小化 } \sum_{i=1}^{I} |x_i - x_{\text{avg},i}| \tag{18}$$

$$\text{约束条件为式(9)~(12)} \tag{19}$$

$$c_{sum} \leq n_k \tag{20}$$

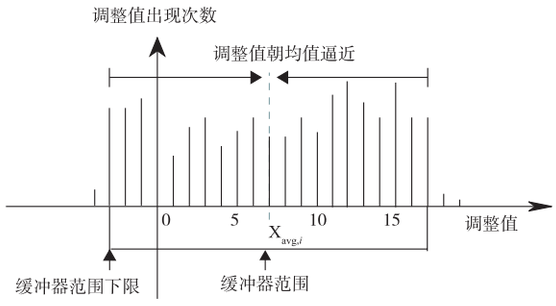


图7 所有采样下的缓冲器调整值均值

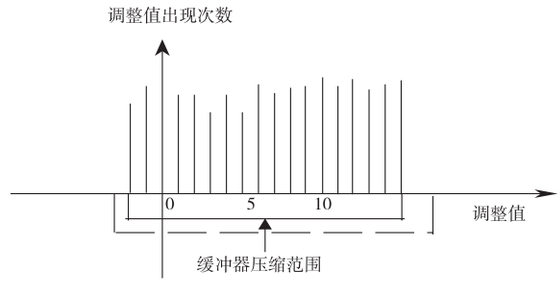


图8 所有采样下的缓冲器调整值压缩范围

在这里我们也用最小的调整次数 n_k 去保证在新一轮的仿真中解算器不会增加缓冲器的数量. 在式(17)~式(20)中,下限已经固定,所以我们仅仅压缩这些调整值向其均值逼近即可降低缓冲器调整值的大小.

最后,我们得到了缓冲器调整值的大小,图 8 描述了这个过程.

4 实验结果

我们用c ++ 实现了该算法,并在3. 20GHzCPU上的一个线程运行该算法. 采用 ISCAS89 的基准电路和 TAU2013 的电路进行验证. 关于这些电路的信息在表 1 中给出,其中 N_s 代表了触发器的数量,N_g 是逻辑门的数量,用于解优化问题整数线性规划的解算器是 Gurobi[13]. 假设最大允许的缓冲器大小是时钟周期的 1/8[15]. 所有的调整值都有 20 个分离值. 晶体管长度,氧化层厚度和阈值电路的标准偏差分别是标准值的 15. 7%,5. 3% 和 4. 4%. 在提出的算法中,我们采样 10000 次. 仿真结果如表 1,表 2 所示.

表 1 ISCAS89 电路仿真结果

电路			μ_T					$\mu_T+\sigma_T$					$\mu_T+2\sigma_T$				
	N_s	N_g	N_b	A_b	$Y(\%)$	$Y_i(\%)$	$T(s)$	N_b	A_b	$Y(\%)$	$Y_i(\%)$	$T(s)$	N_b	A_b	$Y(\%)$	$Y_i(\%)$	$T(s)$
S9234	211	5597	2	12. 60	77. 11	27. 11	52. 23	2	12. 00	95. 95	11. 82	47. 12	2	11. 01	99. 18	1. 46	7. 78
S13207	638	7951	5	9. 70	72. 36	22. 36	156. 08	5	14. 30	96. 43	12. 30	92. 85	6	17. 30	99. 52	1. 80	24. 17
S15850	534	9772	5	19. 83	69. 35	19. 35	223. 07	5	19. 50	94. 32	10. 19	90. 88	5	15. 19	99. 12	1. 40	23. 41
S38584	1426	19253	11	9. 75	85. 98	35. 98	1800. 18	7	13. 17	98. 46	14. 33	683. 66	7	13. 55	98. 98	1. 26	223. 96

表 2 TAU2013 电路仿真结果

电路			μ_T					$\mu_T+\sigma_T$					$\mu_T+2\sigma_T$				
	N_s	N_g	N_b	A_b	$Y(\%)$	$Y_i(\%)$	$T(s)$	N_b	A_b	$Y(\%)$	$Y_i(\%)$	$T(s)$	N_b	A_b	$Y(\%)$	$Y_i(\%)$	$T(s)$
Mem_ctrl	1065	10328	10	11. 91	67. 12	17. 12	1206. 55	10	11. 71	94. 51	10. 38	531. 77	10	8. 70	98. 92	1. 20	147. 90
Usb_funct	1746	14385	17	17. 17	71. 77	21. 77	2203. 67	17	16. 82	96. 55	12. 42	670. 64	9	4. 00	98. 74	1. 02	145. 78
Ac97_ctrl	2199	9207	21	15. 11	75. 02	25. 02	2225. 55	21	15. 44	94. 93.	10. 80	800. 32	8	13. 00	97. 73	0. 01	111. 39
Pci_bridge32	3322	12495	32	13. 85	73. 64	23. 64	5124. 25	32	9. 42	96. 76	12. 63	2594. 27	8	9. 50	98. 68	0. 96	586. 75

为了测试一个电路,首先执行蒙特卡罗仿真去得到有可配置缓冲器电路的时钟周期的均值 μ_T 和标准差 σ_T. 芯片对应于 μ_T,$\mu_T+\sigma_T$ 和 $\mu_T+2\sigma_T$ 的初始的良率 Y_o 分别为 50%,84. 13% 和 97. 72%[15]. 如表 1 所示,Y(%)代表了有可配置缓冲器的芯片的良率. Y_i(%)是相对于没有缓冲器调整初始良率的增加良率,也就是 $Y-Y_o$. 从对比中我们可以得到,芯片良率的增加非常显著(最高可达 35. 98%).

为了提高芯片良率,插入电路中缓冲器的数量为 N_b,其值小于触发器数量的 1%. 除了缓冲器的数量,缓冲器的平均大小在列 A_b 给出. 文献[10,11]在本文相同环境下实现后缓冲器面积增加为 20. 本文缓冲器面积增加的平均大小在列 A_b 给出,可知在同样的 ISCAS89 仿真环境下,相比文献[10,11],本文以更小的面积损耗实现了良率提高. 此外在 TAU2013 的仿真环境下也有同样效果. 通过缓冲器使用量的压缩,最终得到缓冲器的面积小于文献[10,11]中的面积.

$T(s)$代表了执行时间. 如表 2 中所示,对于最大的电路 Pci_bridge32 需要 5124. 25s 完成整个计算. 这些计算时间在电路设计的后期仅运行几次,因此可以接受.

5 结论

本文提出了一种基于时序电路,利用蒙特卡罗算法模拟采样决定可配置缓冲器的位置和数量,最终提高芯片良率的时序电路缓冲器插入算法. 实验结果表明本文提出的算法能够提高芯片良率,而插入的缓冲器数量非常少. 未来的工作将包括流片后的测试和配置.

参考文献

[1] NAFFZIGER S, STACKHOUSE B, GRUTKOWSKI T, et al. The implementation of a 2-core, multi-threaded itanium family processor[J]. IEEE Journal of Solid-State Circuits, 2006, 41(1): 197 – 209.

[2] HUANG Juihung, Lin Yucheng, CHENG Weikai, et al. Unified approach for simultaneous functional and timing ECO[J]. Iet Circuits Devices & Systems, 2016, 10(6): 514 – 521.

[3] WINTERSTEIN Felix, FLEMING K, YANG Hsinjung, et al. Custom multicache architectures for heap manipulating programs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 36(5): 761 – 774.

[4] LIN Tayjyi, SHYU Tingyu. Speculative lookahead for energy-efficient microprocessors [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016, 24(1): 50 – 57.

[5] 蓝帆,潘赟,严晓浪. 用于容错片上网络的可工作性评估框架[J]. 浙江大学学报, 2017, 51(7): 1437 – 1445.
LAN Fan, PAN Yun, YAN Xiaolang. Workability evaluation framework for fault-tolerant network-on-chip[J]. Journal of Zhejiang University, 2017, 51(7): 1437 – 1445. (in Chinese)

[6] XU Cong, ZHENG Yang, NIU Dimin et al. Impact of write pulse and process variation on 22 nm FINFET-based STT-RAM design: A Device-Architecture Co-Optimization Approach[J]. IEEE Transactions on Multi-Scale Computing Systems, 2015, 1(4): 195 – 206.

[7] ABOLMAALI Sheis, MANSOURI-GHIASI Nika, KAMAL Mehdi, et al. Efficient critical path identification based on viability analysis method considering process variations [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25(9): 2668 – 2672.

[8] CAI Yici, DENG Chao, ZHOU Qiang, et al. Obstacle-avoiding and slew-constrained clock tree synthesis with efficient buffer insertion[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015, 23(1): 142 – 155.

[9] ZHANG Graceli, LI Bing, SCHLICHTMANN Ulf. EffiTest: efficient delay test and statistical prediction for configuring post-silicon tunable buffers[A]. Design Automation Conference[C]. Austin: IEEE, 2016. 61 – 66.

[10] LAK Z, NICOLICI N. A novel algorithmic approach to aid post-silicon delay measurement and clock tuning[J]. IEEE Transactions on Computers, 2014, 63(5): 1074 – 1084.

[11] KHANDELWAL V, SRIVASTAVA A. Variability-driven formulation for simultaneous gate sizing and post-silicon tunability allocation[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(4): 610 – 620.

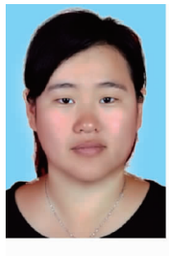
[12] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual[DB/OL]. http://www.gurobi.com/documentation/6.5/refman, 2016. 12.

[13] LAK Z, NICOLICI N. On using on-chip clock tuning elements to address delay degradation due to circuit aging [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(12): 1845 – 1856.

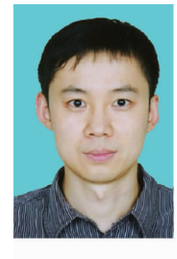
[14] LI B, CHEN N, SCHLICHTMANN U. Fast statistical timing analysis for circuits with post-silicon tunable clock buffers[A]. International Conference on Computer-Aided Design[C]. San Jose: IEEE, 2011. 111 – 117.

[15] ZHANG Graceli, LI Bing, LIU Jinglan, et al. Design-phase buffer allocation for post-silicon clock binning by iterative learning[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, (99): 1 – 1.

作者简介



戢小亮　女,1981 年生,讲师,研究方向:数字集成电路设计,信号与信息处理.
E-mail: jiliangzi@126.com



佟星元　男,1984 年生,博士/博士后,副教授,研究方向:集成电路设计.
E-mail: mayxt@126.com