关于提高 SoC 设计时钟域交叉验证

GUPTA Vrinda[1], KASIM Mohammad[1], JEBIN Mohandas[2]*

1.国立技术学院 印度 古鲁格舍德拉 136119

2.Intel Technology India Private Limited 印度 古鲁格舍德拉 136119

【摘 要】：摩尔定律一直推动半导体行业大量生产多时钟（大多不相关的）复杂的系统级芯片（SoC）设计。跨越这种不相关或异步时钟域的数据/信号更有可能在稳定之前被采样，从而导致亚稳态等问题。这种时钟域交叉（CDC）信号必须使用同步器有效地在域之间同步，且必须在质量验收前完全验证。如果没有得到适当验证或在设计周期后期才被检测到，这些交叉将导致芯片旋转并增加成本。在基于平面网表的 CDC 验证检查中，最为典型的困难是超长的运行时间、内存消耗以及数百万次违反调试计数。因此，针对系统级芯片集成电路进行了分层 CDC 验证方法。这种分层方法在不影响质量的前提下，在内存消耗、签收时间方面改进了 SoC 的 CDC 验证。

【关键词】：SoC 亚稳定性；时钟域交叉；同步；重置域交叉；故障

Towards Improving Clock Domain Crossing Verification for SoCs

GUPTA Vrinda[1], KASIM Mohammad[1], EBIN Mohandas[2]*

1. National Institute of technology India Kurukshetra 136119

2. Intel Technology India Private Limited India Kurukshetra 136119

Abstract: Moore‘s law has been motivating the semiconductor industry to churn out multi-clock (mostly unrelated) complex system on chip (SoC) designs. Data/signals that crosses such unrelated or asynchronous clock domains are more likely to be sampled before they are stable, and can cause issues like metastability. Such clock domain crossing (CDC) signals must be synchronized between the domains using a valid synchronizer, and must be verified in some way exhaustively before quality sign-off. If not verified appropriately or detected late in the design cycle, these crossings will result into chip re-spins and prove too costly, as sometimes the product itself will be out of market. The typical challenge in flat netlistbased CDC verification checks is huge run time, memory consumption & millions of violations to debug counts. In this paper, therefore, a hierarchical CDC verification methodology has been implemented for system-on-chip integrated circuits. This hierarchical approach improves the CDC verification for SoC in terms of memory consumption, time taken for sign-off without compromising quality.

Keywords: SoC; Metastability; Clock Domain Crossing; Synchronization; Reset Domain Crossing; Glitch

1 引言

早期的 ASIC/定制芯片设计通常只有一两个时钟来驱动整个芯片。在相同的硅尺寸上遵循摩尔定律的路线图来放置更多数量的晶体管的尝试仍在继续。通过在单个芯片上放置更多数据，人们可以放置更多由不同时钟驱动的不相关的东西，这便导致数据跨时钟域。这是现在非常常见的情况。跨时钟域的数据还会导致不同的问题，如亚稳态、数据丢失等问题。这可以统称为跨时钟域交叉（CDC）问题[1]。随着人们制造更复杂的系统，（CDC）问题在现代 SoC 芯片上持续增多。因此，随着设计复杂性的增加，需要解决方案来缩小问题的复杂性增加。传统的寄存器传输级（RTL）验证技术，如功能仿真、时序分析和许多其他复杂的形式技术，不能保证制造的芯片没有 CDC 和跨复位域交叉（RDC）错误。如果未经验证，导致芯片旋转的一些 CDC 和 RDC 相关故障如下[1,2]。

（1）CDC 路径中的设置和保持时间违规导致的亚稳态

（2）异步 RDC 导致的亚稳态

（3）交叉路径的收敛和发散导致的毛刺传播

（4）时钟抖动

对于像 SoC 这样的复杂设计，拥有多个独立时钟域是很常见的，因此在设计中存在大量 CDC 交叉路径，数据可能会面临亚稳态、数据丢失、数据不一致等 CDC 问题。跨时钟域的数据易受 CDC 问题的影响，并可能导致芯片功能故障。很难或不可能在模拟级别检测到此类问题[3]虽然。没有办法确定这个问题可以通过某种方法来防御，但是人们可以在同步器的帮助下尝试减少这种问题对芯片的有害影响。针对跨时钟域的不同数据场景，提出了不同的同步器，因此验证同步器的正确性变得非常关键[4]。有几种设计自动化工具可以帮助找到丢失的同步器、同步信号的重新收敛、交叉中的发散等[5,6]。此外，报告 System Verilog Assertions（SVA）的研究工作可用于在模拟中查找 CDC 故障[7,8]。SoC 芯片的 CDC 验证面临着多方面的挑

战，如巨大的运行时间、内存消耗和百万违规计数。考虑到所有的挑战，我们应该努力改进针对 SOC 的 CDC 验证技术。

本文采用了一种分层方法来改进 SOC 的 CDC 验证。分级方法利用了可用的 CDC 验证技术的优势。通过减少运行时间，它可以降低内存利用率，减少签收时间。

2 研究相关回顾

在超大规模集成电路工业中，已经进行了多种努力来对设计进行 CDC 验证。在文献资源中，存在关于 CDC 问题验证的重要著作[9,10]。作者在此描述了两种验证 CDC 接口的方法。其中，基于规则的模型检测器被应用于可编程门阵列（FPGA）实现及其在几个综合基准电路中的应用。该方法的结果显示，与本文中讨论的其他现有推测方法相比，平均节省 135%和 204%的面积和大约 100%的电力成本。在另一项工作中，设计了一种新的方法来分析同步器的行为和性[11]。在此，同步器的电压增益的时间已经被公式化，并且结果已经证明了在同步器电路中增益作为来自再生反馈的时间的函数而指数地增加。

许多影响同步器性能测量的因素已经被确定，如工艺技术、操作条件和电路设计[12]。本文试图证明这种全局约束适用于整个集成电路，而其他因素可以在设计中针对单个同步器进行调整。此外，本文提出了一种改进同步器的方法，即选择最小尺寸的触发器单元、避免扫描和复位、最小化布线、选择高性能风格和最小阈值电压（VTH）以及通过减少相干时钟域交叉中的抖动。该文献还包含基于 SAL 模型的检查器，用于验证简单 CDC 接口电路的正确性[13]。本文采用基于 SAL 的模型对 CDC 接口进行建模。本文还提供了证明电路符合基本不变量的证明。除此之外，文献中的一些其他工作已经建议使用 SMV 模型检查器来验证多个时钟设计[14,15]。

3 CDC 问题

如果跨时钟域的数据/信号没有正确同步，则可能导致许多 CDC 问题之一，具体取决于设计架构。以下小节将讨论一些常见的 CDC 问题。基于亚稳定性通过属性说明语言生成的规则（PSL）信号转换图（STG）的特性。在第二种方法中，验证数据传输、重复数据或缺失数据的正确性。另一项重要工作提出了一种新的基于 SAT 的方法来验证 CDC 方案[16]。这里的重点是通过为每个时钟分配一个状态变量来模拟多个时钟，该变量在每个验证时钟周期取值 0 或 1。这里假设任何顺序触发器的建立和保持时间都为零。

3.1 亚稳态

现代 SOC 在不同的接口上有许多时钟域，所以它使用不同的同步器。在一项研究工作中描述了一种解决方案，通过将同步延迟与计算周期重叠来防止同步延迟[17]。本文通过现场试验验证了该方法的正确性电子学中的亚稳态是指数字电子系统在无限时间或有限时间内保持不稳定平衡或亚稳态的能力[18-21]。电路可能无法在电路正常工作所需的时间内稳定在稳定的“0”或“1”逻辑电平。在数字设计中，除了逻辑“0”或逻辑“1”之外的任何值，比如 0.8VDD、0.5VDD 或 0.2VDD，在应该采样的时间内都不会解析为逻辑“0”或逻辑“1”，那么该系统的操作就会变得不稳定/不可预测。如图 1 所示，CLK1=C1 和 CLK2=C2 是异步时钟，A 是 TX 触发器的输出。这里，信号“A”的变化非常接近 C2 的有效边沿，并且它落在 RX- flop 的建立保持窗口之间，这导致 RX-flop 的输出在不确定的时间段内不稳定，或者 RX-flop 的输出处于亚稳态。在不确定的时间后，根据不稳定状态的解决方式，输出将稳定为逻辑“1”或逻辑“0”。

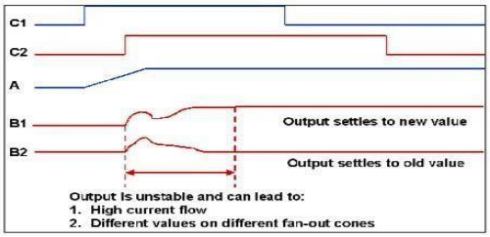

图 1 具有亚稳态的 CDC 信号[9]

因此，亚稳态是设计人员试图避免的问题，因为它会导致与电源相关的故障，以及取决于架构的功能错误。在亚稳态结束时，触发器将最终稳定到逻辑“0”或逻辑“1”。建立所需的时间取决于接收触发器的技术，即晶体管增益、保持器尺寸和电容等因素。因此，可以计算稳定时间（td）并将其用于找出平均故障间隔时间（MTBF）。决定 MTBF 的等式如下所示。

$$MTBF \quad \alpha \quad 1/(td * fin * fclk) \qquad (1)$$

其中，td 是触发器进入亚稳态后建立到稳定值所需的建立时间，fin 是输入数据的频率，fclk 是对输入数据进行采样的时钟频率。MTBF 表示 RX_flop 进入亚稳态之前的估计时间。

3.2 数据丢失

如果 TX 触发器的时钟和数据频率与 RX 触发器的时钟频率相比非常高，那么如果没有正确同步，可能会导致数据丢失。为了避免这种数据丢失，TX-flop 的输出数据应该在某个最小时间周期内保持稳定，以便相对于目的时钟的至少一个有效边沿满足建立和保持时间要求。图 2（a）显示了当 C1 和 C2 的有效时钟边沿彼此靠近（绿色）和不太靠近（粉色）时的亚稳定效应。当有效边沿彼此接近时，如果输入数据在最短时间内保持恒定，使得时序要求与目标时钟的至少一个有效边沿匹配，则数据将在第二个目标时钟周期被捕获。但是如果时钟边沿不接近，则数据在第一个目的时钟周期被捕获。这里重要的一点是，发生在源位置的每个转换都应该在目标位置捕获，以避免数据丢失。考虑数据总线跨域的情况，源时钟的频率比目的时钟快两倍时钟。两者相位相同。在图 2（b）中，0101111 序列是 TX 触发器在“C1””有效沿的输出，由 RX 触发器采样。

接收触发器的输出将是“0011”。这里，在源时钟“C1”的第三个有效边沿上从“0”到“1”的转换没有被目的地触发器捕获，因此数据丢失。

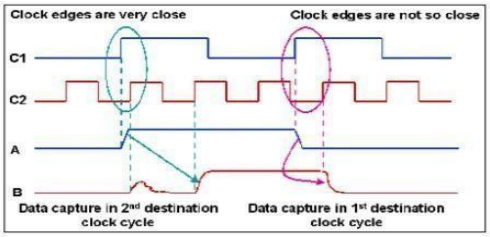

图 2（a） 输出和数据丢失时的亚稳态效应[9]

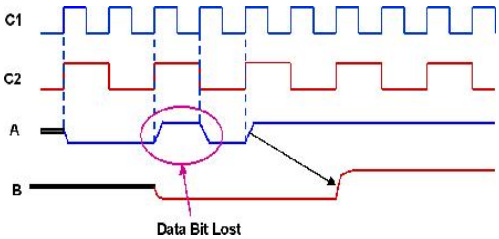

图 2（b） 跨域总线的数据丢失[9]

3.3 数据不一致

为了理解数据不一致问题，考虑一个例子，其中数据总线跨越时钟域，数据总线的每一位分别与双同步（背靠背连接的 2 D FF）同步。现在，如果数据总线上的位落在接收器域时钟的建立保持窗口内，则根据双同步的解决方式，数据总线每个接收位的输出可能在 N 或 N-1 或 N+1 个周期后出现，因此可能导致接收器端的数据不正确。在这种情况下，会失去数据一致性，并可能导致功能错误。数据不一致的情况如图 3 所示。X [0:1]是由“C1”计时的源的 2 位输出数据总线，Y [0:1]是由“C2”计时的目的的输出数据总线。 这里，将“00”和“11”作为由源时钟“C1”为数据总线 X 产生的两个有效值。现在，让我们假设在 X 的两个位上的 $1 \rightarrow 0$ 的转变，并且目标时钟 C2 在第一个周期中完美地捕获了这两个值，使得 Y 变成“00”。接下来，在信号 X 的两个位上从 $0 \rightarrow 1$ 的转变期间，时钟 C2 的上升沿接近信号 X 上的转变，由此在 Y[0:1]上获得中间值“10”,这是无效状态，因此在这种情况下丢失了数据一致性。

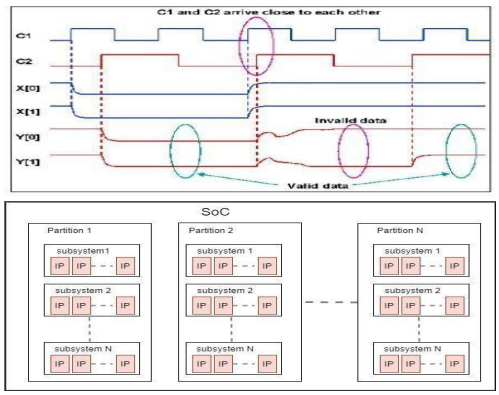

图 3 数据不一致[9]

4 分层 CDC 验证流程

CDC 层次分析非常适合基于 IP 的 SOC 和大型复杂设计。它是分析多时钟、复位和多时钟域交叉设计的最佳选择。通常，基于 IP 的 SoC 流程中的 IP 在顶层设计和设置之前可用。IP 组运行块级 CDC 分析并生成抽象块模型，然后将这些模型交付给集成团队，后者在顶级 CDC 分析中使用这些抽象模型。这种抽象模型是以一种基于 Tcl 的文件格式规定的，这种文件格式称为 SDC（综合设计约束）。这种抽象模型应在 SoC 级别选择和使用，以避免重新读取和合成 IP 的需要，从而大大节省运行时间，因为其抽象可用的 IP 可直接用于 CDC 检查。该 SDC 文件被导入分层的自下而上的分析流程。CDC 工具的流程可以分为三个不同的检查阶段，如下所述。

（1）验证设计设置。该阶段通过编译检查 RTL 设计文件，以及与时钟和复位相关的所有顶层设计约束信息的正确性。

（2）验证模块设置并分析模块。此阶段指定并验证数据块设置，并生成数据块级别的时钟报告，以确保指定所有约束。该阶段还为该模块生成一个抽象模型，该模型将在下一阶段用于 SOC 级别的 CDC 验证。它包含了 SOC 中模块集成期间验证和调试问题所需的所有模块信息。

（3）顶层的层次分析。在块级别审查违规并修复任何问题后，将在顶层执行分层分析。这种 CDC 分层顶级分析包括在块级分析期间为每个块生成的抽象信息。

5 结果和讨论

表 1 显示了通过使用中使用的所有 IP 的抽象模型在 SoC 子系统上运行平面 CDC 和分层 CDC 检查获得的结果。可以看出，与平面 CDC 相比，分层 CDC 消耗非常少的运行时间、更少的存储器利用（资源）以及更少的违规计数。总的来说，分级 CDC 提高了结果的质量，从而在设计中捕捉到实际的 CDC 问题。

表 1 扁平化 CDC 与层级化 CDC 的比较

参数	CDC 核查方法		
	扁平 CDC	层级化 CDC	% 改进
抽象 IP/块的数量	不适用	20	不适用
设置	X （K）	X/2 （K）	~ 50%
CDC 验证	~ Y Lacs	~ Y/32 lac	~ 96%
故障检查	Z （K）	Z/4 （K）	25%
内存	P （GB）	P/2 （GB）	50%
运行时间	Q（hrs.）	Q/6（hrs.）	~ 83%

6 小结与展望

本文重点对改进现代复杂 SoC 设计中出现的时钟域交叉问题进行了验证。与平面 CDC 流相比，分层 CDC 流的实施在运行时间、内存利用和违规计数方面显示出改进的结果。这种分级方法将有助于设计工程师将设计按时交付到设计周期的

下一阶段。将来，人们可以将注意力集中在大型复杂 SOC 芯片的 CDC 分析上，这些 SOC 具有超过 100 个子系统实例，这使得 CDC 分析对于分析来说更加复杂。通过将整个 SoC 划分为逻辑分区，可以在每个逻辑分区执行顶级 CDC 分析，并生成在 SoC 级别使用的分区摘要。通过在不同的部分水平上并行执行 CDC 分析，它将具有减少运行时间的优势。

参考文献：

[1] N. Karimi and K. Chakrabarty, “Detection,Diagnosis, and Recovery from Clock-Domain Crossing Failures in Multi clock SoCs”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 9, pp. 1395-1408, September 2013.

[2] Y. Feng, Z. Zhou, D. Tong, X. Cheng, “Clock domain crossing fault model and coverage metric for validation of SoC design”, Proc. Design, Automation & Test in Eur. Conf. &Exhibition, Nice, France, pp. 1-6, 2007.

[3] P. Ashar and V. Viswanath, “Closing the Verification Gap with Static Sign-off”, International Symposium on Quality Electronic Design, CA, USA, pp. 343-347, 2019.

[4] C. Szàsz, and R. Şinca, “The Nontrivial Problem of Matching in Redundant Digital Systems”, Journal of Electrical and Electronics Engineering, vol. 12, no.1, pp. 51-56, May 2019.

[5] S. Yang and M. Greenstreet, “Computing Synchronizer Failure Probabilities”, Proc. Design, Automation & Test in Europe Conf. & Exhibition, Nice, France, pp. 1-6, 2007.

[6] I.W. Jones, S. Yang, and M. Greenstreet, “Synchronizer Behavior and Analysis”, Int’l Symp. on Asynchronous Circuits and Systems, NC, USA, pp. 117-126, 2009.

[7] K. R. Talupuru and S. Athi, “Achieving Glitch- Free Clock Domain Crossing Signals Using Formal Verification, Static Timing Analysis, and Sequential Equivalence Checking”, 12th Intl. Workshop on Microprocessor Test and Verification, Austin, pp.5-9, 2011.

[8] Y.Peng, I.W. Jones and M. Greenstreet, “Finding Glitches Using Formal Methods”, Intl. Symp. on Asynchronous Circuits and Systems, pp. 45-46, 2016.

[9] T. Kapschitz and R. Ginosar, “Formal verification of synchronizers”, Proc. 13th IFIP Correct Hardware Design and Verification Methods, Germany, pp. 359-362, Oct 2005.

[10] R. Dobkin, T. Kapshitz, S. Flur and R.Ginosar, “Assertion Based Verification of Multiple-Clock GALS Systems”, Proc. IFIP/IEEE Int. Conference on Very Large- Scale Integration (VLSI-SoC), Greece, pp. 1-6, Oct. 2008.

[11] E. Clarke, D. Kroening, and K. Yorav, “Specifying and Verifying Systems with Multiple clocks”, P roc. Intl. Conf. on Computer Design, CA, USA, pp. 48-55, 2003.

[12] G.Tarawneh, A.Yakovlev, and T.Mak, “Eliminating Synchronization Latency Using Sequenced Latching”, IEEE Transactions on Very Large-Scale Integration Systems, vol.22, no.2, pp.408–419, 2014.

[13] S.Beer, J. Cox, R. Ginosar, T. Chaney, and D. M. Zar, “Variability in Multistage Synchronizers”, IEEE Transactions on Very Large Scale Integration Systems, vol. 23, no. 12, pp. 2957–2969, 2015.

[14] S. Beer and R. Ginosar, “Eleven Ways to Boost Your Synchronizer”, IEEE Transactions on Very Large Scale Integration Systems, vol. 23, no. 6, pp. 1040–1049, 2015.

[15] G. M. Brown, “Verification of a Data Synchronization Circuit for All Time”, Intl. Conf. on Application of Concurrency to Sys. Design, Finland, pp. 217-228, 2006.

[16] A. Smrcka, V. Rehak, T. Vojnar, D. Safranek, P. Matousek, and Z. Rehak, “ Verifying VHDL Design with Multiple Clocksin SMV”, FMICS, , pp. 148-164, 2007.

[17] A. Smrcka, “Verification of Asynchronous and Parametrized Hardware Designs”, Information Sciences and Technologies Bulletin of the ACM Slovakia, vol. 2, no. 2, pp. 60-69, 2010.

[18] R. Ginosar, “Metastability and Synchronizer: ATutorial”, IEEE Design and Test of Computers, pp. 23-35, Oct. 2011.

[19] C. Portmann, and T. Meng, “Metastability in CMOS library elements in reduced supply and technology scaled applications”, IEEE Journal of Solid-State Circuits, vol. 30, no. 1, pp. 39-46, 1995.

[20] J. Reiher, M. Greenstreet, I. W. Jones, “ Explaining Metastability in Real Synchronizers”, Intl. Symp. on Asynchronous Circuits and Systems, Austria, pp. 59-67, 2018.

[21] M. Thakur, B. B. Soni, P. Gaur and P. Yadav, “Analysis of metastability performance in digital circuits on flip-flop”, International Conference on Communication and Network Technologies, Sivakasi, India, pp. 265-269, 2014.