文章编号:1671-4598(2020)01-0179-05 DOI:10.16526/j.cnki.11—4762/tp.2020.01.038 中图分类号:TP391 文献标识码:A

基于反馈约束的SRAM接口时序分析方法

左丽丽，刘国斌，吴维林，陈 云
（上海航天软件测评中心，上海 201109）

摘要：FPGA验证作为保证FPGA产品功能和可靠性的重要手段已经备受关注；对接口芯片时序的验证通常通过布局布线后仿真来进行，但布局布线后仿真需要耗费大量的时间；介绍了一种基于反馈的SRAM接口时序验证的方法，将FPGA输入输出连接成一个回路，验证结果表明，与动态仿真验证相比，该种静态时序验证方法可以较早、快速、精确定位FPGA接口时序设计存在的问题；缩短了验证时间，提高了验证效率、准确性和覆盖率。

关键词：FPGA；静态时序分析；SRAM

SRAM Interface Timing Analysis Method Based on Feedback Constraint

Zuo Lili, Liu Guobin, Wu Weilin, Chen Yun
(Shanghai Aerospace Software Testing and Evaluation Center, Shanghai 201109, China)

Abstract: FPGA verification has attracted much attention as an important means of FPGA product function and reliability. Verification of interface timing is usually done by post-layout simulation, but simulation after layout and routing takes a lot of time. Introduces a method of timing verification of SRAM interface based on feedback constraint, which links the input and output of FPGA, the verification results show that compared with dynamic simulation, this static timing verification method can locate the problems in timing design of FPGA interface earlier, faster and more accurately. It shortens the verification time and improves the verification efficiency and coverage.

Keywords: FPGA; static timing analysis; SRAM

0 引言

随着军工产品向着高集成度、小型化、高速和高可靠性方向发展，FPGA和CPLD等可编程逻辑器件在军工产品中的应用数量成爆发式增长[1]。相对于传统的逻辑器件，FPGA能够很大程度缩短实验时间[2]，随着设计规模的增大和设计复杂度的提高，许多原本应用于专用集成电路（ASIC）的验证方法如STA（static timing analysis，静态时序分析）方法也逐渐应用到FPGA的设计验证中。静态时序分析不需要测试向量，即使没有仿真条件也能快速地分析电路中所有时序路径是否满足约束要求[3]。

本文使用静态时序分析工具Prime Time，针对某FPGA设计中SRAM的读写接口设计进行时序验证。采用衍生时钟约束，输入、输出延时约束，多周期路径约束相结合的方式，将FPGA的输入输出信号关联起成一个回路，成功的对FPGA与SRAM之间的交互进行环路分析。根据分析结果，快速准确发现了接口设计的缺陷和薄弱环节，高效实现设计的优化。

1 SDRAM接口设计概况

某图像处理FPGA在进行设计时采用3片SRAM接口进行图像数据的预存。硬件接口图如图1所示。

2片SRAM与FPGA的信号连接关系如表1所示。

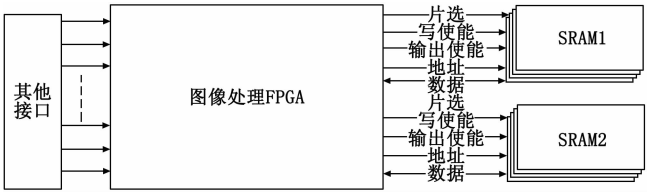


图1 FPGA设计硬件接口框图

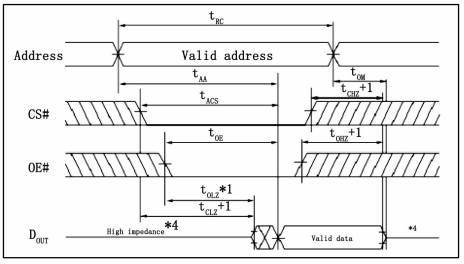
表1 SRAM与FPGA对外接口信号表

信号名	I/O	信号功能	连接关系
sram1_cs0_n	O	片选信号1,低有效	SRAM1端口CS0#
sram1_cs1_n	O	片选信号2,低有效	SRAM1端口CS1#
sram1_we_n	O	写选通信号,低有效	SRAM1端口WE#
sram1_oe_n	O	读选通信号,低有效	SRAM1端口OE#
sram1_add	O	地址线	SRAM1地址端口Address
sram1_dat	I/O	数据线	SRAM1数据端口D
sram2_cs0_n	O	片选信号1,低有效	SRAM2端口CS0#
sram2_cs1_n	O	片选信号2,低有效	SRAM2端口CS1#
sram2_we_n	O	写选通信号,低有效	SRAM2端口WE#
sram2_oe_n	O	读选通信号,低有效	SRAM2端口OE#
sram2_add	O	地址线	SRAM2地址端口Address
sram2_dat	I/O	数据线	SRAM2数据端口D

设计中采用的主处理时钟频率为70 M，在对SRAM进行读写控制时，需满足芯片读写时序要求，根据器件手册SRAM读数据时序及时序参数如图2，SRAM写数据时序及时序参数如图3。

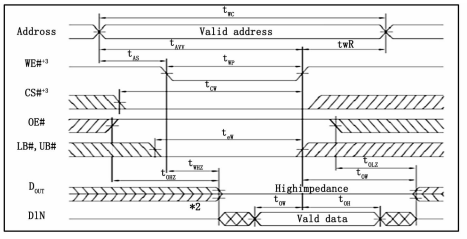
收稿日期:2019-05-13； **修回日期**:2019-07-05。

作者简介:左丽丽(1983-),女,河南三门峡人,高级工程师,主要从事航天型号FPGA产品验证方向的研究。



Parameter	Symbol	Min	Max	Unit	Notes
Read cycle time	t_{RC}	12	—	ns	
Address access time	t_{AA}	—	12	ns	
Chip select access time	t_{ACS}	—	12	ns	
Output enable to output valid	t_{OE}	—	6	ns	
Byte select to output valid	t_{BA}	—	6	ns	
Output hold from address change	t_{OH}	3	—	ns	
Chip select to output in low-Z	t_{CLZ}	3	—	ns	1
Output enable to output in low-Z	t_{OLZ}	0	—	ns	1
Byte select to output in low-Z	t_{BLZ}	0	—	ns	1
Chip deselect to output in high-Z	t_{CHZ}	—	6	ns	1
Output disable to output in high-Z	t_{OHZ}	—	6	ns	1
Byte deselect to output in high-Z	t_{BHZ}	—	6	ns	1

图 2 读 SRAM 数据时序及时序参数图



Parameter	Symbol	Min	Max	Unit	Notes
Write cycle time	t_{WC}	12	—	ns	
Address valid to end of write	t_{AW}	8	—	ns	
Chip select to end of write	t_{CW}	8	—	ns	8
Write pulse width	t_{WP}	8	—	ns	7
Byte select to end of write	t_{BW}	8	—	ns	
Address setup time	t_{AS}	0	—	ns	5
Write recovery time	t_{WR}	0	—	ns	6
Data to write time overlap	t_{DW}	6	—	ns	
Data hold from write time	t_{DH}	0	—	ns	
Write disable to output in low-Z	t_{OW}	3	—	ns	1
Output disable to output in high-Z	t_{OHZ}	—	6	ns	1
Write enable to output in high-Z	t_{WHZ}	—	6	ns	1

图 3 写 SRAM 数据时序及时序参数图

由图 2～3 可知，SRAM 读写操作由多个信号同时控制，包括 CS0＃、CS1＃、WE＃、OE＃、Address 和 Din/Dout，且相互之间需满足一定的时序关系才能保证 SRAM 的正常工作。为了便于控制，设计中将 CS0＃、CS1＃置为常有效，通过 WE＃和 OE＃控制读写逻辑的产生。

2 传统的接口验证方法

FPGA 的验证工作，在很多方面都表现出了较高的复杂性和较强的技术性[4]。按照传统的验证流程，依次进行前仿真验证、静态时序分析、后仿真验证。

2.1 前仿真验证

考虑各时序参数的要求，在进行接口设计时考虑各信号的相关性，尽量避免同时跳变，写时序下地址和数据的持续时间较长，而 WE＃在地址和数据稳定时有效，为各时序参数预留充分的余量，读时序下采用地址驱动 SRAM 数据输出，FPGA 在下个周期读取数据，前仿真结果如图 5。

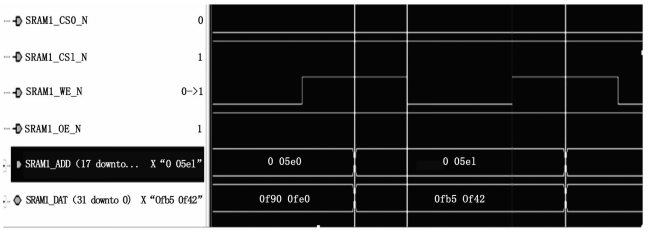


图 4 SRAM 写时序前仿真结果图

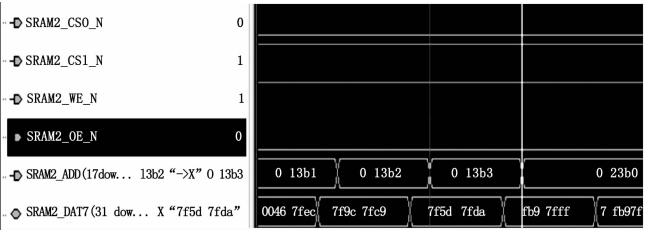


图 5 SRAM 读时序前仿真结果图

前仿真情况下，SRAM 读写时序参数测试结果如表 2 所示。

表 2 SRAM 时序参数

参数	说明	要求/ns	实测值(前仿)	结果
tRC	读周期	≥12	14.286 ns	满足
tWC	写周期	≥12	28.572 ns	满足
tAS	地址有效到写开始时间	≥0	7.143 ns	满足
tAW	地址有效到写结束时间	≥8	21.429 ns	满足
tWP	写有效时间	≥8	14.286 ns	满足
tWR	写结束到地址变化时间	≥0	7.143 ns	满足
tDW	数据有效到写结束的时间	≥6	21.429 ns	满足
tDH	写结束后数据的保持时间	≥0	7.143 ns	满足

功能前仿真结果表明读写时序均满足要求，且各时序参数均存在一定的余量，因此对设计进一步开展静态时序分析。

2.2 传统的静态时序分析方法

从信号接口表可以看出，与 SRAM 的接口中除了数据线之外，均为 FPGA 的输出信号，因此在静态时序分析时，由于输入数据 sram＊ _dat 与本地的 70 M 时钟并无固定的相位关系，按照常规处理方式，仅进行了 output delay 的约束，静态时序分析通过后进一步通过布局布线后的仿真验证接口处理功能的正确性。

2.3 后仿真验证

在后仿真验证 SRAM 接口时，进行了 3 种工况的布局布线后仿真。写时序的产生完全由 FPGA 输出控制，且写时序在静态时序分析时已添加了时序约束，后仿真的结果表明静态时序分析的结果与后仿真的结果一致且均满足时序要求。

但读时序的验证则遇到了问题，写时序的产生完全取决于 FPGA，数据线和控制线均由 FPGA 输出产生，控制相对简单。而读时序动作的完成需要 FPGA 和 SRAM 配合

完成，读时序的整个过程其实可以划分为分为 3 个子过程：1）FPGA 输出读时序控制信号 CS0＃、CS1＃、WE＃、OE＃和 Address；2）SRAM 在以上信号的控制下输出对应的数据；3）FPGA 读取 SRAM 输出的数据并在内部进行取数。在模拟 SRAM 读时序接口时，由图 2 可知，读时序下时序参数例如 t_{AA}为最大值、t_{OH}为最小值，均非典型值，因此若要考虑覆盖性，则需要进行仿真的模式非常之多：最大工况下 t_{AA}取最大值、最大工况下 t_{AA}取最小值、最大工况下 t_{AA}取典型值、最小工况下 t_{AA}取最大值、最小工况下 t_{AA}取最小值等等。

以上的仿真方案存在如下问题：

1）t_{AA}值的选取直接影响了仿真效果；

2）后仿耗时巨大，且存在多种工况多种模式，在整个 FPGA 设计规模较大的情况下，此种验证方式严重降低了验证效率；

3）一旦发现接口时序不满足，由于为后仿真网表，代码中的信号在布线后的名称被改变，追溯和定位问题困难，即便发现问题后进行修改，很可能造成参数 A 在某工况下满足要求了，而参数 B 又出错了。

基于以上的问题，到了后仿真阶段再进行接口时序的验证，不但效率低下，而且设计方案变更困难。测试投入和成本较高，不利于型号的快速研制[5]，因此项目组决定重新使用静态时序分析的方式对 SRAM 接口进行验证

3　基于反馈约束的静态时序分析方法

静态时序分析的验证方法，它可以简单的定义为：设计者提出一些特定的时序要求，或者说是添加特定的时序约束，套用特定的时序模型，针对特定的电路进行分析。时序分析的目的就是确保在 FPGA 芯片的 3 种工况下，整个设计的接口设计满足上级和下级系统接口时序需求[6]。利用分析结果给出的路径延迟可以反向的进行设计改善，指导综合和布局布线工具，从而实现优化设计。Prime Time 是 Synopsys 公司的一款功能强大的全芯片和门级静态时序分析工具，也是业界最流行的工具，常被用来分析大规模、同步、数字 ASIC。

3.1　静态时序路径

STA 的工作原理是提取数字电路的所有时序路径，计算信号在时序路径上的传播延迟，分析其中最大路径延迟和最小路径延迟[7]。静态时序分析时将设计分成 4 种时序路径，每条路径包含一个起点和终点，时序路径的起点只能是设计的输入端口或者内部寄存器的时钟输入端，终点只能是内部寄存器的数据输入端或者设计的基本输出端口。因此 STA 的 4 种时序路径包括：

1）从输入端口到寄存器的数据 D 端；

2）从寄存器的时钟 CLK 端到寄存器的数据 D 端；

3）从寄存器的时钟 CLK 端到输出端口；

4）从输入端口到输出端口。

静态时序分析主要是通过对最大路径延迟和最小路径延迟的分析，检查建立时间、保持时间、移除时间、恢复时间等是否满足要求。

结合本项目的实际情况，SRAM 接口的静态时序路径包含以上 4 种路径中的 2 种：即从输入端口到寄存器的数据 D 端的路径和从触发器的时钟 CLK 端到输出端口的路径，这两种路径在没有添加相关约束情况下并不会自动分析，因此即使 input delay 为 0，也需要进行约束，若不约束，静态时序分析工具并不会按照默认 0 进行计算，而是直接忽略。添加合理的约束是决定静态时序分析质量的决胜因素。

3.2　静态时序约束

FPGA 时序约束的工作就是模拟 FPGA 的实际工作环境，包括定义 FPGA 的时钟频率，设置数据端口的输入输出延时，设置芯片电路内部的伪路径等[8]。时序约束的作用主要是指导 FPGA 软件按照一定的原则布局布线，以控制逻辑路径端点间的布线延时。同时，时序约束还为静态时序分析器提供时序分析的标准和依据[9]。

数据的到达时间：输入数据在有效时钟沿后到达所需要的时间。通常由三部分组成：时钟到达寄存器时间，寄存器输出延时和数据传输延时。

数据要求时间：在时钟锁存的建立时间和保持时间之间数据必须保持稳定，从源时钟起点到达这种稳定状态需要的时间即为数据需求时间，默认的参考值是一个时钟周期。如果数据能够在一个要求时间内到达终点，那么这条路径符合设计规则。SLACK＝要求时间－到达时间

其中，要求时间为约束时长，到达时间为添加延时候的实际到达时刻，SLACK 为时序余量，正值表示满足时序，负值表示不满足时序，静态时序分析把上公式作为依据，分析设计中的所有时序路径。如果得到的静态时序分析报告中 SLACK 为负值，那么此时序路径存在时序问题，是一条影响整个电路工作性能的关键路径。在逻辑综合、布局布线等阶段进行静态时序分析，就能及时发现并修改关键路径上存在的时序问题，达到修正错误，优化设计的目的。

因此，正确且全面的约束是时序设计的重要环节，约束过少或过于放松实际上即是对设计的放松，也不能发挥静态时序分析应有的作用。

3.2.1　设置输入端口延时

输入端口延时的约束可以优化从输入端口到第一级寄存器之间的路径延迟，保证系统时钟采到的外部信号可靠、稳定。FPGA 端口需要跟不同的外部器件连接，通过设置输入延时来模拟端口信号在片外路径的传播情况，可使静态时序分析结果更加符合实际。设置输入端口延时的语法如下：

```
set_input_delay[－clock clock_name]
                [－clock_fall]
                [－level_sensitive]
                [－rise]
                [－fall]
                [－max]
```

[－min]
[－add_delay]
[－network_latency_included]
[－source_latency_included]
delay_value
port_pin_list

由上可知，输入端口延时约束的两大要素分别为：延时值、相关的时钟。简单的解释就是外部输入相对某时钟的延时时间，此处外部输入数据必须和时钟有固定或较固定的相位关系。根据图 2 所示，SRAM 输入到 FPGA 的数据与地址、片选、读使能均有较固定相位关系，但设计中采用片选、读使能接固定值、读地址驱动数据输出的方式产生读数据，因此可约束输入数据相对读地址的延时值。由图可知，SRAM 数据相对于读地址的延时最大值为 t_{AA}－12 ns，最小值为 t_{OH}－3ns，根据此信息进入如下约束，两组 SRAM 的约束相同：

set_input_delay 12 －max －clock sram1_adclk [get_ports sram1_dat]

set_input_delay 3 －min －clock sram1_adclk [get_ports sram1_dat]

3.2.2 时钟的创建

对于时钟的约束有两种，create _ clock 和 create _ generated _ clock。create _ clock 命令用于创建一个时钟，包括时钟名称、源、周期和波形。create _ generated _ clock 用于创建一个衍生时钟。

在 3.2.1 节中提到，设计中采用读使能驱动的方式产生读数据，因此可约束输入数据相对读使能的延时值。但根据 set _ input _ delay 的语法要求，约束的输入延时只能是相对于时钟的，而设计中的读使能信号并非时钟，此时就用到了虚拟时钟，此处可以使用衍生时钟的约束方式来实现，设置衍生时钟约束的语法如下：

create_generated_clock [－name clock_name]
－source master_pin
[－edges edge_list]
[－divide_by factor]
[－multiply_by factor]
[－duty_cycle percent]
[－invert]
[－edge_shift shift_list]
[－add]
[－master_clock clock]
source_objects

设计中 FPGA 和 SRAM 的数据流交互实际上均与 70 M 时钟 clk _ 70 M 相关，在 70 M 时钟下产生 SRAM 的地址信号 SRAM1 _ ADD，SRAM 输出数据到 FPGA 后，FPGA 再使用 70 M 的时钟读取 SRAM 数据，以 SRAM1 为例，数据的交互如图 6 所示。

基于以上的信息，首先将 70 M 时钟约束为时钟，再将

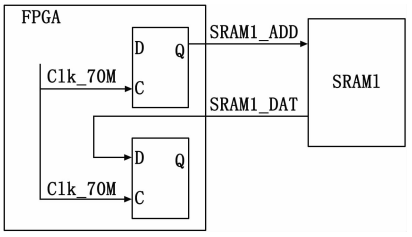


图 6 SRAM1 与 FPGA 数据交互

SRAM1 _ ADD 约束为 70 M 时钟的衍生时钟，约束方法如下：

create_clock －period 14.286 clkin －name CLK_fpga70 －waveform [list 0 [expr 7.143]]

create_generated_clock －name sram1_adclk －source [get_ports clkin] －divide_by 1 [get_ports sram1_add[5]]

3.2.3 定义多周期路径

默认情况下静态时序分析工具基于单周期进行时序的检查，set _ multicycle _ path 可以将普通的单周期电路扩展为多周期，用在同源但不同周期或者成倍频关系的时钟域之间，设置多周期约束的语法如下：

set_multicycle_path [－setup]
[－hold]
[－rise]
[－fall]
[－start]
[－end]
[－from from_list]
[－to to_list]
[－through through_list]
path_multiplier

在一定程度上讲多周期约束实际上是对时序检查的放松，因此一定要确定设计中确实采用了多周期的设计方法，才能进行多周期的约束。

set_multicycle_path －to [get_ports sram1_dat] －setup 2 －start

set_multicycle_path －to [get_ports sram1_dat] －hold 1 －start

3.3 静态时序分析

经过以上约束后，运行静态时序分析软件，发现端口 SRAM1 _ DAT 在读操作情况下建立时间不满足要求，以其中的一条路径为例进行分析，结果如图 7 所示。

由图 7 可知，整个路径的起点为 sram1 _ adclk，即输出到 SRAM 的地址信号，在此条路径计算时首先添加了 SRAM 地址线上的输出延时信息“clock network delay”，之后添加了约束的输入延时值“input external delay”，之后再添加 FPGA 内部的延时信息，与图 8 中的数据流吻合，结合到图 6 中，各延时数据分布如下：

进一步对以上的路径在后仿真时进行验证，在后仿真时将以上的数据路径分别找出，发现延时数据与静态时序

```
Startpoint: sram1_dat[0]
            (input port clocked by sram1_adclk)
Endpoint: *******/*******/*******rd_din_0/FDC_1
            (rising edge-triggered flip-flop clocked by CLK_70M')
Path Group: CLK_70M
Path Type: max

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock sram1_adclk (rise edge)                            0.00       0.00
clock network delay (ideal)                              7.94       7.94
input external delay                                    12.00      19.94 f
sram1_dat[0] (inout)                                     0.00      19.94 f
*******/*******ds2/IBUF/O (X_BUF)                        0.19 *    20.13 f
sram1_dat<0>/IFF/IMUX/O (X_BUF)                          0.63 *    20.77 f
*******/*******/*******_din_mux0001<0>1/O (X_LUT4MUX16)
                                                         5.07 *    25.83 r
*******/*******/*******rd_din<0>/DXMUX/O (X_BUF)
                                                         0.28 *    26.11 r
*******/*******/*******rd_din_0/FDC_1/I (X_FF)
                                                         0.00 *    26.11 r
data arrival time                                                  26.11

clock CLK_70M' (rise edge)                              21.40      21.40
clock network delay (propagated)                         3.30      24.70
*******/*******/*******rd_din_0/FDC_1/CLK (X_FF)        24.70 r
library setup time                                      -0.23 *    24.47
data required time                                                 24.47
-----------------------------------------------------------------------------
data required time                                                 24.47
data arrival time                                                 -26.11
-----------------------------------------------------------------------------
slack (VIOLATED)                                                   -1.64
```

图 7　SRAM1 读时序接口静态时序分析结果

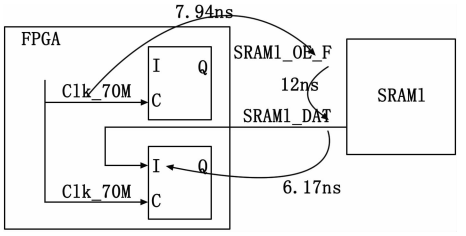


图 8　静态时序分析中的延时与数据路径对应关系

分析的数据值吻合，进一步证明该种约束方案快速且有效。

4　设计改进

由于静态时序分析时清楚的列出数据的路径及每一级的延时信息，这就方便设计人员分析导致问题发生的关键原因，从而进一步调整方案，从上述的分析结果看，造成该问题的主要原因是由于 SRAM 的地址存取时间 t_{AA}、保持时间 t_{OH}在 3～12 ns 的范围内，该范围过大。一般情况下 t_{AA}参与建立时间余量的计算，t_{OH}参与保持时间余量的计算，在调整设计后很难达到建立时间和保持时间的平衡，建立时间满足了，保持时间又出错了，或者最大工况下满足了，最小工况又出错了。

基于以上的原因，将设计方案进行调整，由地址驱动改为读使能驱动控制读时序，由图 2 可知，读使能驱动情况下的延时值在 0～6 ns 之间，处于更加可控的延时范围内，最终正确实现了 FPGA 对 SRAM 的读写控制。

静态时序约束的方式简便快捷，但最重要的时必须保证约束内容的正确性，约束的内容越准确，则分析的结果越趋近于实际值，例如：

1）以上约束时添加的数值实际上均为常温常压下的数据，而在实际使用中考虑器件在高低温条件下的延时可能变大，对 SRAM 器件资料进行进一步的解读，在高温或低温下大部分的时序参数值会增大或减小 10%，在进行约束时也需要考虑 FPGA 产品的实际使用环境，从而考核出 FPGA 设计真正的余量；

2）3.2.2 节中将地址约束为衍生时钟，而实际上地址为多 bit 信号，可以通过 set _ max _ delay 的约束方式找出延时最大和延时最小的地址线。

5　结论

本文以 SRAM 读、写接口设计的时序验证为例，采用 Prime Time 对读写接口进行时序分析，介绍了将输入输出关联的特殊约束方式，将传统的单向的时序检查变为数据环路控制的时序检查，有效的提高设计及验证效率，该种方式同样可推广应用于 ROM、MRAM 等类似接口芯片。

参考文献：

[1] 朱伟杰，周　辉，费亚南，等. 一种基于时序路径的 FPGA 接口时序测试方法 [J]. 航天控制，2017，35 (4)：79.

[2] 卫建华，刘　琪，齐　攀，等. 基于 FPGA 的可配置时序信号发生系统设计 [J]. 应用天地，2017，36 (10)：107.

[3] 谈晓婷，付宇卓，谢凯年. SOC 静态时序分析中时序约束策略的研究及实例 [J]. 微电子学与计算，2006，23 (4)：64V

[4] 于维佳，何毅波，秦　臻. SOC 数字集成电路 FPGA 验证的若干研究 [J]. 电子技术与软件工程，2016，1：133.

[5] 刘海山，乔　森，丁怀龙. 基于软、硬件协同的 FPGA 软件交联仿真验证技术 [J]. 导弹与航天运载技术，2017，4：75.

[6] 周　珊，王金波，王晓丹. 基于时序路径的 FPGA 时序分析技术研究 [J]. 微电子学与计算机，2016，33 (1)：77.

[7] 刘宁宁，田　泽，郭　蒙. 基于 H.264/AVC 解码芯片的静态时序分析约束设计 [J]. 计算机技术与发展，2014，24 (5)：91.

[8] 常迎辉，杨振学，张　勇，等. 一种基于多 FPGA 的 Soc 验证方法 [J]. 计算机与网络，2012，24：60.

[9] 刘　垚，王　维，巩玉振，等. 在 Altera 的 FPGA 中实现高速 Link 口的时序约束方法 [J]. 测控技术，2012，31 (1)：117.

[10] 周海斌，等. 静态时序分析在高速 FPGA 设计中的应用 [J]. 电子工程师，2016，31 (11).

(上接第 178 页)

[11] Lawton J R T，Beard R W，Young B J. A decentralized approach to formation maneuvers [J]. IEEE Transactions on Robotics & Automation，2004，19 (6)：933－941.

[12] Olfati－Saber R. Flocking for multi－agent dynamic systems：algorithms and theory [J]. IEEE Transactions on Automatic Control，2006，51 (3)：401－420.

[13] Su H，Wang X，Lin Z. Flocking of Multi－Agents With a Virtual Leader [J]. IEEE Transactions on Automatic Control，2009，54 (2)：293－307.

[14] 丁家如，杜昌平，赵耀，等. 基于改进人工势场法的无人机路径规划算法 [J]. 计算机应用，2016，36 (1)：287－290.

[15] 游文洋，章政，黄卫华. 基于模糊改进人工势场法的机器人避障方法研究 [J]. 传感器与微系统，2016，35 (1)：14－18.

[16] Olfati－Saber R. Flocking for multi－agent dynamic systems：algorithms and theory [J]. IEEE Transactions on Automatic Control，2006，51 (3)：401－420.