SCM设计文档

国防科技大学计算机学院

2018年9月

**文档过程记录表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **时间** | **修正内容** | **校对人** | **版本号** | **备注** |
| **2018.9.14** | **设计文档** | **蒋越** | **V1.0** | **设计文档初稿** |
| **2018.9.17** | **状态机细化** | **蒋越** | **V1.1** | **状态机进一步细化** |
| **2018.9.26** | **状态机修改** | **蒋越** | **V1.2** | **状态机重新修改**  **接口信号更新** |
| **2018.9.28** | **测试** | **蒋越** | **V1.3** | **测试SCM设计功能正确性** |
|  |  |  |  |  |

# 设计概述

## 需求分析

支持对FAST-ANT收到的报文进行相关的统计操作，从而获取测试方所感兴趣的的数据，如：报文数量、时延等。并且根据具体的需求来决定对收到的报文是进行丢弃操作，还是上送软件端处理。

## 功能概述

1. 支持对报文数据字段或对FAST提供的metadata进行解析，提取与数据统计有关的信息，如：时间戳等；
2. 支持计数操作，包括对总接收bit数计数、总接收报文数计数等；
3. 支持将统计结果写入指定寄存器，并允许软件端进行读取；
4. 支持根据不同需求对报文进行丢弃或上送软件的操作。

# SCM详细设计

## SCM整体框架

SCM位于GME和UDA之间，如图2.1所示。



图2.1 SCM位于FAST UM五级流水线中的位置

SCM负责获取相关统计信息，对保存测试结果的寄存器（计数器）进行更新，以及对报文进行丢弃或上送软件操作。

SCM相关的计数器如表1所示。

表1 SCM中相关计数器信息

|  |  |  |  |
| --- | --- | --- | --- |
| 计数器名称 | 计数器大小 | 计数器初始值 | 计数器描述 |
| scm\_bit\_num\_cnt | 64 | 64’b0 | 记录总接收bit数量 |
| scm\_pkt\_num\_cnt | 64 | 64’b0 | 记录总接收报文数量 |
| scm\_time\_cnt | 64 | 64’b0 | 记录总接收时间 |

FAST-ANT将SCM置于GME和GAC之间，以借助FAST所提供的Metadata和PHV数据消息来获取对相关字段（域）的解析结果。并通过软件端写寄存器（protocol\_type）操作来确定测试所需获取的报文类型，减少不必要的解析工作，从而进一步提升测试性能。

由于FAST-ANT提供主动测量，即：自定义测试报文并发送，因此需要在SCM中确定对报文是进行直接丢弃操作，还是上送软件端。这样，便可以确保主动测量和测试统计在FAST-ANT中同时进行。

## SCM接口信号关系图

SCM位于GME与GAC之间，其信号接口与FAST流水线中原GME对GAC的输出信号与GAC来自GME的输入信号相匹配，从而减少对前后两个模块的更新与修改，同时也符合FAST UM流水线中模块间的接口定义规范。图2.2为SCM接口信号关系图。



图2.2 SCM接口信号关系图

## SCM接口信号定义

SCM接口信号定义如表2所示。

表2 SCM接口信号定义

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名称 | 方向 | 宽度 | 信号描述 |
| clk | input | 1 | 时钟信号 |
| rst\_n | input | 1 | 复位信号 |
| in\_scm\_md | input | 256 | GME传递给SCM的MD |
| in\_scm\_md\_wr | input | 1 | GME传递给SCM的MD的写使能信号 |
| in\_scm\_phv | input | 1024 | GME传递给SCM的PHV |
| in\_scm\_phv\_wr | input | 1 | GME传递给SCM的PHV的写使能信号 |
| in\_scm\_md\_alf | input | 1 | GAC能够接收MD的信号 |
| in\_scm\_phv\_alf | input | 1 | GAC能够接收PHV的信号 |
| gac2scm\_sent\_start | input | 1 | GAC传递给SCM的测试开始信号 |
| gac2scm\_sent\_end | input | 1 | GAC传递给SCM的测试结束信号 |
| cin\_scm\_data | input | 134 | GME传递给SCM的控制通路数据 |
| cin\_scm\_data\_wr | input | 1 | GME传递给SCM的控制通路数据的写使能信号 |
| cin\_scm\_ready | input | 1 | GAC准备接收SCM传递的控制通路数据的信号 |
| out\_scm\_md\_alf | output | 1 | SCM能够接收MD的信号 |
| out\_scm\_phv\_alf | output | 1 | SCM能够接收PHV的信号 |
| out\_scm\_md | output | 256 | SCM传递给GAC的MD |
| out\_scm\_md\_wr | output | 1 | SCM传递给GAC的MD的写使能信号 |
| out\_scm\_phv | output | 1024 | SCM传递给GAC的PHV |
| out\_scm\_phv\_wr | output | 1 | SCM传递给GAC的PHV的写使能信号 |
| out\_scm\_ready | output | 1 | SCM准备接收GME传递的控制通路数据的信号 |
| out\_scm\_data | output | 134 | SCM传递给GAC的控制通路数据 |
| out\_scm\_data\_wr | output | 1 | SCM传递给GAC的控制通路数据的写使能信号 |

## SCM状态转换

SCM状态机包含五种状态，分别为IDLE\_S、SEND\_S、CNT\_S、WAIT\_S和FETCH\_S。其具体状态转换如图2.3所示。



图2.3 SCM状态转换

1. IDLE\_S：

SCM初始状态，此时SCM内数据信号和使能信号置为0。

当FAST-ANT未启动时，SCM作为FAST流水线的一个模块，将收到的数据正常转发到下一模块。此时状态跳转到SEND\_S状态。

当FAST-ANT启动时，接收到gac2scm\_sent\_start信号跳转至CNT\_S状态。

1. SEND\_S：

SCM作为FAST流水线的一个模块时的转发状态，此时将收到的数据正常转发到下一模块。

当转发结束时，跳转到IDLE\_S状态。

1. CNT\_S：

SCM测试统计状态，此时SCM根据对收到的测试报文进行解析和统计来修改相应计数器。对于非测试报文则正常转发。转发时跳转到SEND\_S状态。

当接收到gac2scm\_sent\_end信号时，跳转至WAIT\_S状态。

1. WAIT\_S：

SCM等待接收统计终止状态。在此状态中，SCM在接收到gac2scm\_sent\_end信号后n个RTT（n由探针报文确定）后停止接收测试报文，即超时，跳转到FETCH\_S状态。

如果在n个RTT内仍有测试报文，那么在WAIT\_S状态中仍然进行解析和统计操作，并负责跳转到SEND\_S状态进行转发。

1. FETCH\_S：

软件获取相关寄存器的值，并在确定新的测试开始时，接收statistic\_reset信号，相关寄存器清零，并跳转至IDLE\_S状态。

## 地址空间划分

SCM内部地址空间划分如表3所示。

表3 SCM内部地址空间划分

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 寄存器名称 | 寄存器大小 | 寄存器描述 |
| 0x7000 0000 | protocol\_type | 8 | 测试报文类型 |
| 0x7000 0001 | statistic\_reset | 1 | 测试重置信号 |
| 0x7000 0002 | n\_RTT | 32 | 超时间隔 |
| 0x7000 0008 0x7000 0009 | scm\_bit\_num\_cnt | 64 | 记录总接收bit数量 |
| 0x7000 000A 0x7000 000B | scm\_pkt\_num\_cnt | 64 | 记录总接收报文数量 |
| 0x7000 000C 0x7000 000D | scm\_time\_cnt | 64 | 记录总接收时间 |

# 测试案例

## 软件下发信号正确性验证

SCM接收从软件下发的信号，即读取控制报文中的特殊字段来获取相关测试信息。

在SCM中，定义了三个接收软件下发信息的寄存器，分别是protocol\_type、statistic\_reset和n\_RTT，其大小、虚拟地址以及描述见表2。核心代码实现如图3.1所示。

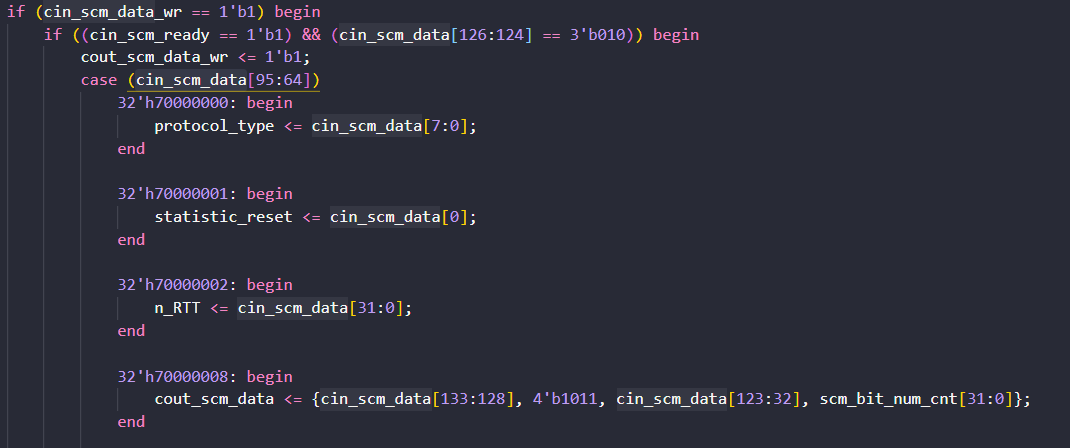


图3.1 接收软件端下发信息的具体代码实现

我们编写了相应的Testbench，如图3.2所示。并在Vivado Simulator中完成仿真，其结果如图3.3所示。

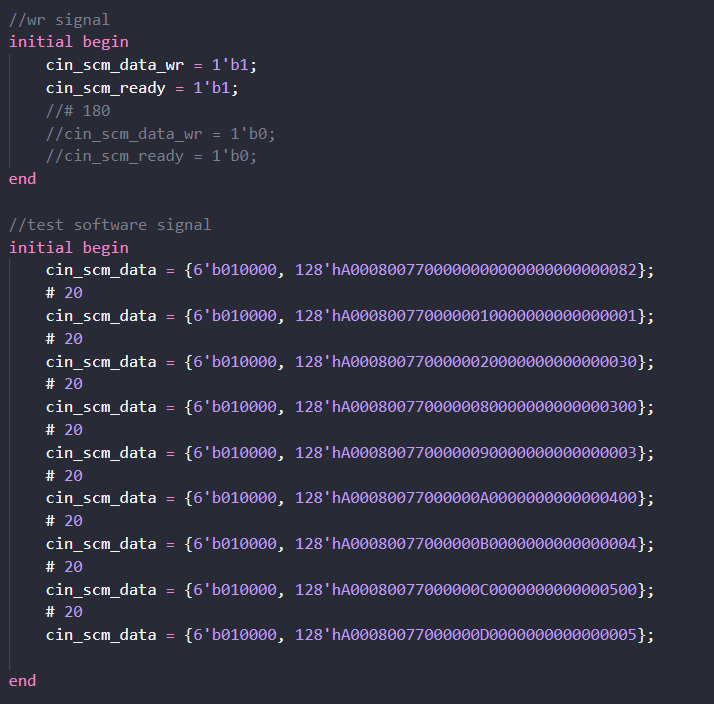


图3.2 Testbench具体内容

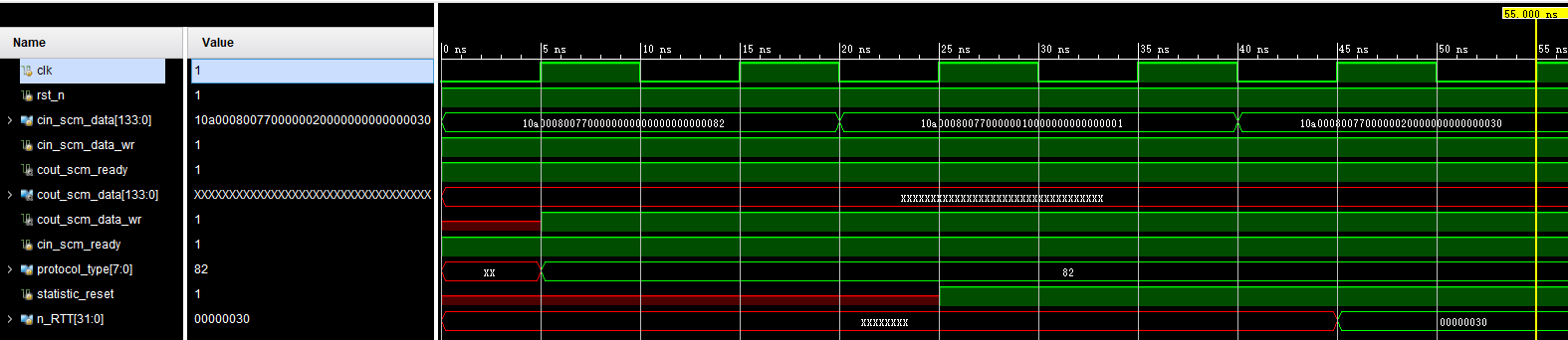


图3.3 仿真结果

在Testbench中，每间隔20，发送一个控制报文，共发送三个控制报文。报文的95：64位为虚拟地址，分别对应三个寄存器的虚拟地址。低32位为写入的数据内容，提取相应位数的内容放到对应的寄存器中。可以对比图3.2和图3.3，发现仿真结果符合期望实现功能，从而验证了该部分设计功能的正确性。

## 软件读取计数器正确性验证

SCM内定义了三个不同的计数器scm\_bit\_num\_cnt、scm\_pkt\_num\_cnt以及scm\_time\_cnt，分别负责保存不同的测试结果。当软件需要读取这些寄存器的值时，便通过发送控制报文，将寄存器的值写入低32位的数据字段中，并修改126：124位为011，即此报文为读响应帧，上送给软件端。

由于控制报文的数据字段只有32位宽，而SCM内的计数器为64位宽，因此对于一个计数器需要两个控制报文才能将数据读完。我们对于每一个计数器分配两个地址，通过两个控制报文将数据部分读取。核心代码实现如图3.4所示。

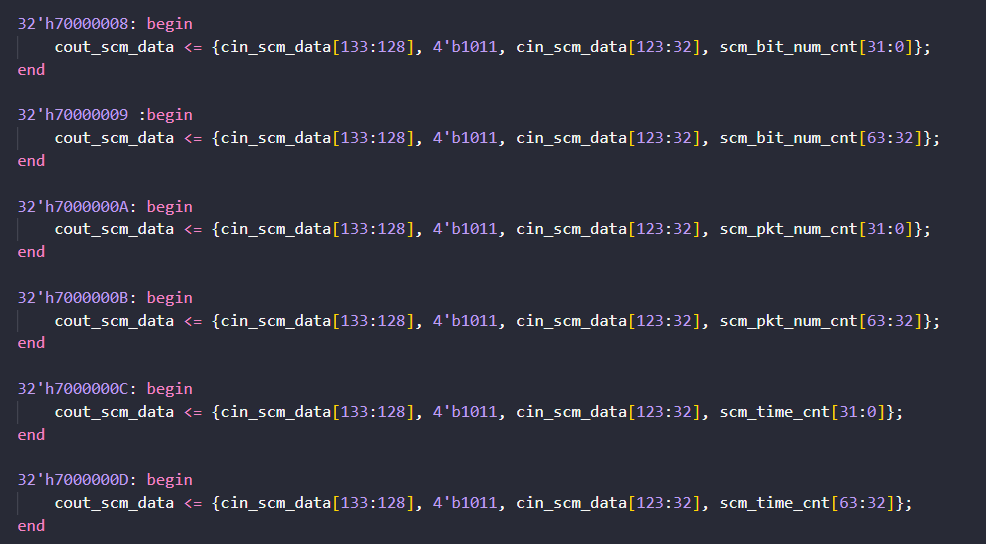


图3.4 软件端读取计数器的具体代码实现

我们编写了相应的Testbench，如图3.5所示。并在Vivado Simulator中完成仿真，其结果如图3.6所示。

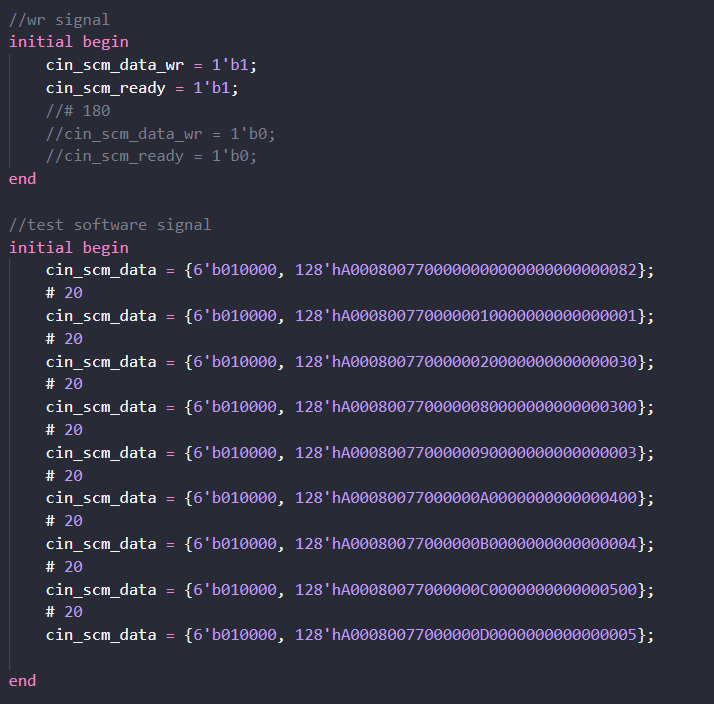


图3.5 Testbench具体内容

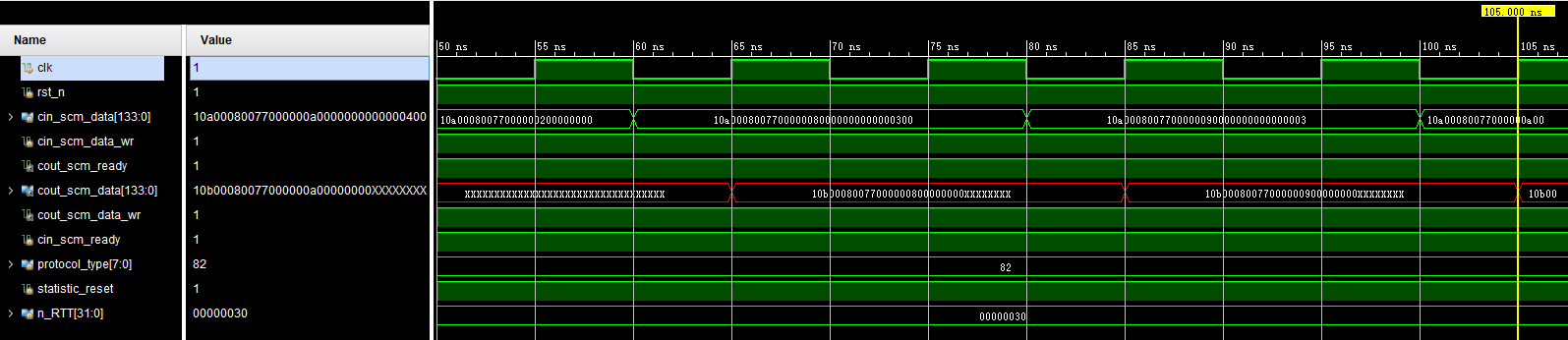


图3.6 仿真结果

通过观察仿真结果，我们发现两个测试报文保存一个计数器的内容，并上送软件部分。由于没有对计数器进行赋值，因此低32位为未知值。

## 状态机正确性验证

SCM内部状态机共有五个状态，分别是IDLE\_S、SEND\_S、CNT\_S、WAIT\_S以及FETCH\_S。下面将依次验证这五个状态的正确性。

### 3.3.1 IDLE\_S：

1. IDLE\_S -> IDLE\_S

当MD的FIFO和PHV的FIFO都没有数据进入时，反馈信号MD\_fifo\_empty和PHV\_fifo\_empty均是1，SCM状态仍然保持IDLE\_S。仿真结果如图3.7所示。

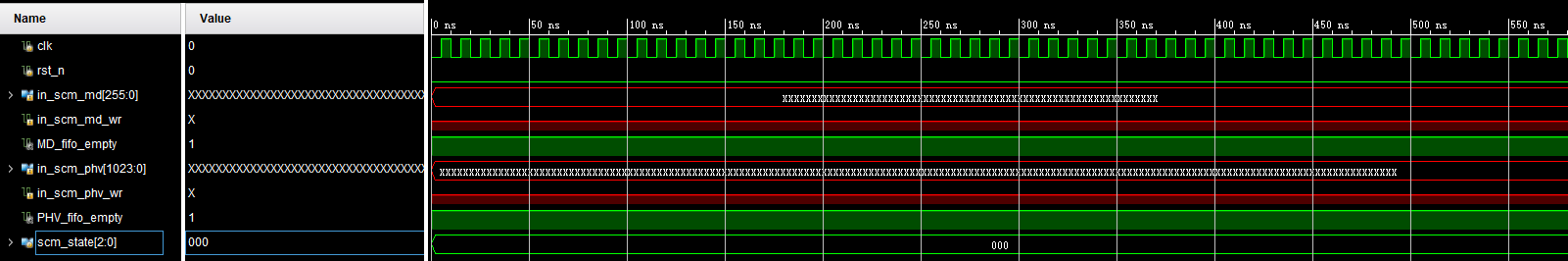


图3.7 仿真结果

1. IDLE\_S -> SEND\_S

当MD的FIFO和PHV的FIFO有数据进入，并且其写使能信号有效时，FIFO的empty信号为0，表示FIFO不为空。获取FIFO中的数据到MD\_fifo\_rdata，并判断其87：80位是否为LMID。

如果满足LMID，由于未收到GAC模块传来的测试开始信号，则修改NMID（7->5）后进行转发，仿真结果如图3.8所示。

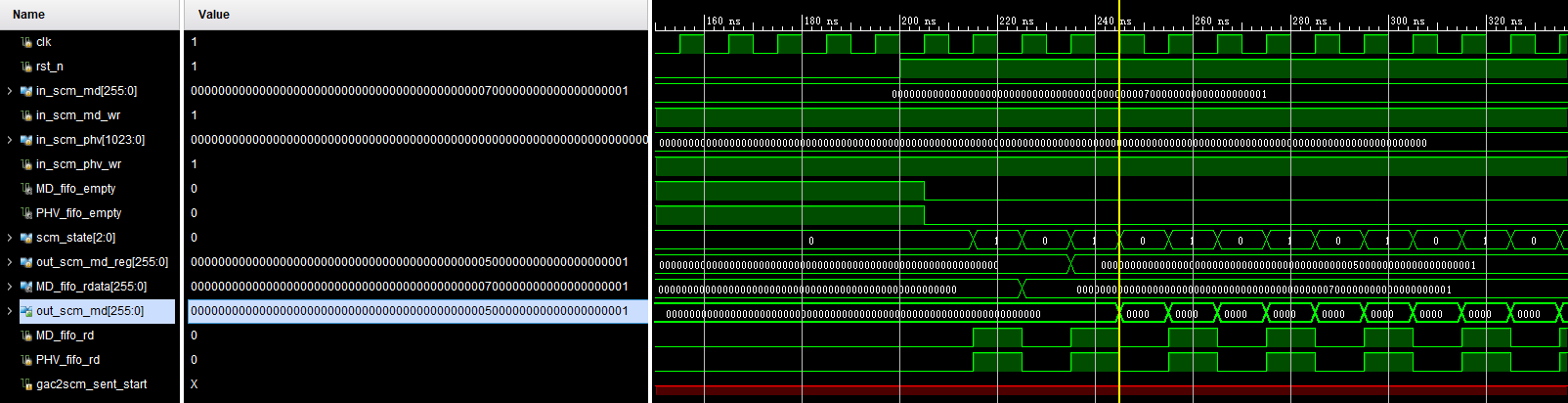


图3.8 符合LMID时修改NMID后转发的仿真结果

可以看出out\_scm\_md中将NMID修改为5。

如果不满足LMID，则是单纯的转发，跳转到SEND\_S状态，仿真结果如图3.9所示。

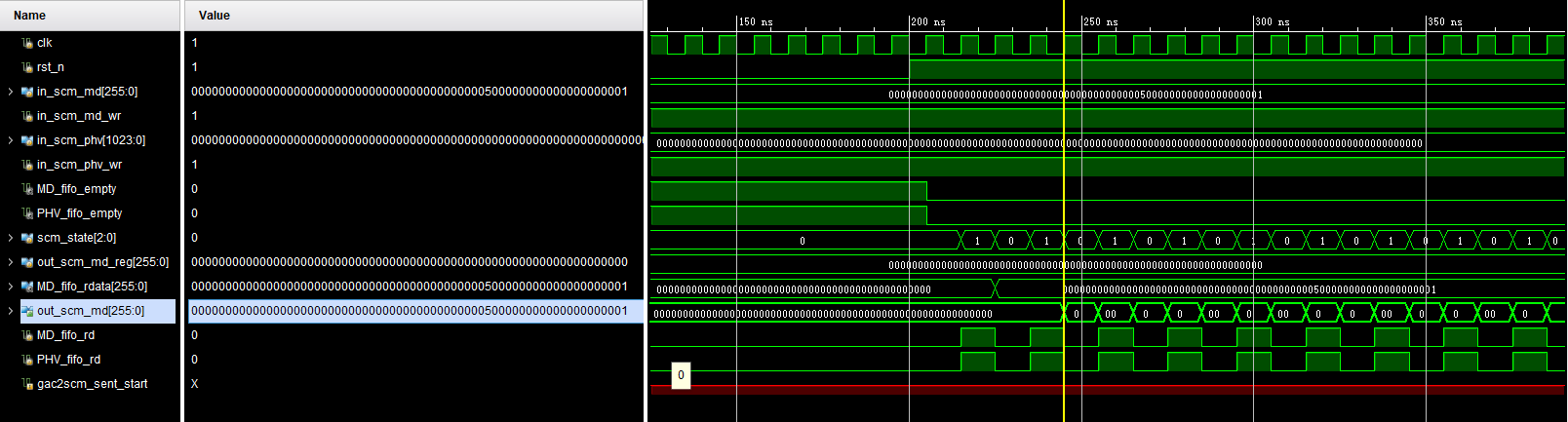


图3.9 不符合LMID时转发的仿真结果

1. IDLE\_S -> CNT\_S

当SCM接收到GAC发送的测试开始信号gac2scm\_sent\_start有效时，跳转到CNT\_S状态进行统计操作。仿真结果如图3.10所示。

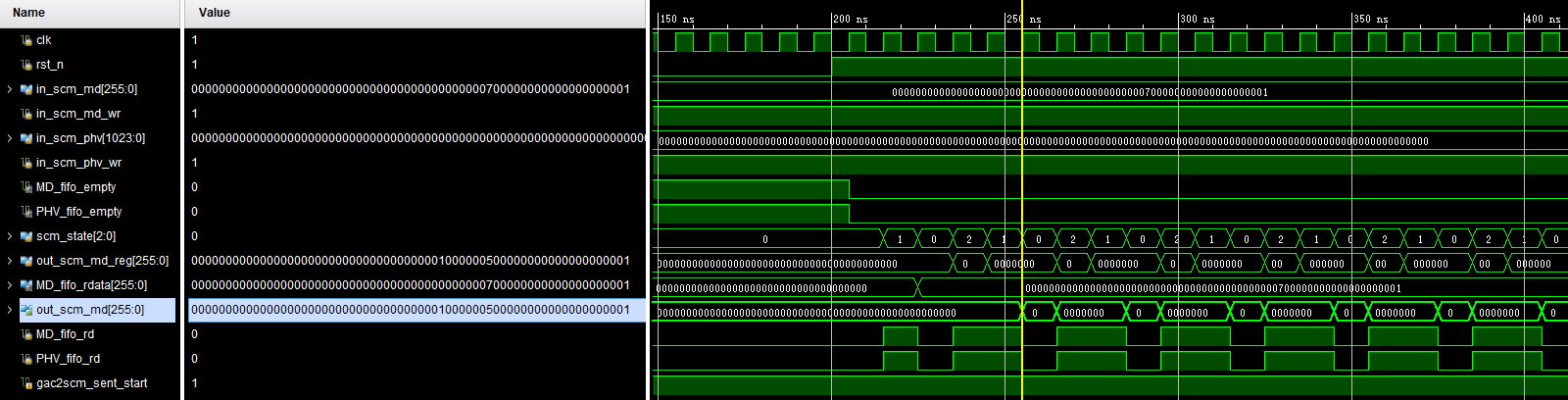


图3.10 IDLE\_S跳转到CNT\_S的仿真结果

从图3.10可以看出，状态从0 -> 2 -> 1，即从IDLE\_S -> CNT\_S -> SEND\_S，状态转换正确。

### 3.3.2 SEND\_S：

1. SEND\_S -> IDLE\_S

在发送状态结束后，将会跳转到空闲状态，等待新的状态转换。仿真结果如图3.11所示。

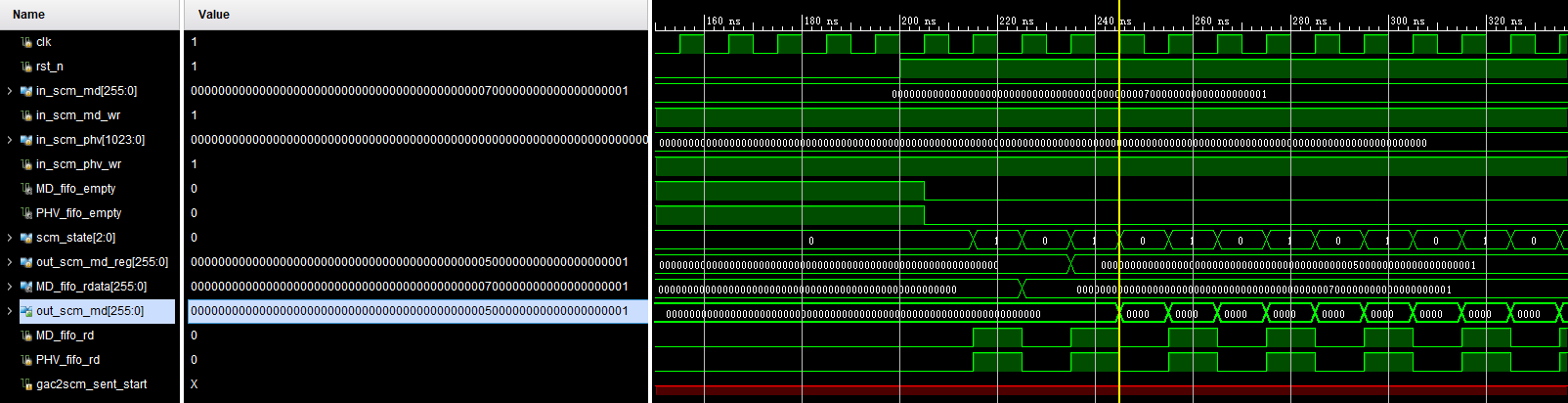


图3.11 SEND\_S跳转到IDLE\_S的仿真结果

从图3.11可以发现，状态从0 -> 1 -> 0，即从IDLE\_S -> SEND\_S -> IDLE\_S，状态转换正确。

### 3.3.3 CNT\_S：

1. CNT\_S -> WAIT\_S

当接收到GAC传递的测试结束信号gac2scm\_sent\_end时，跳转至WAIT\_S状态。仿真结果如图3.12所示。

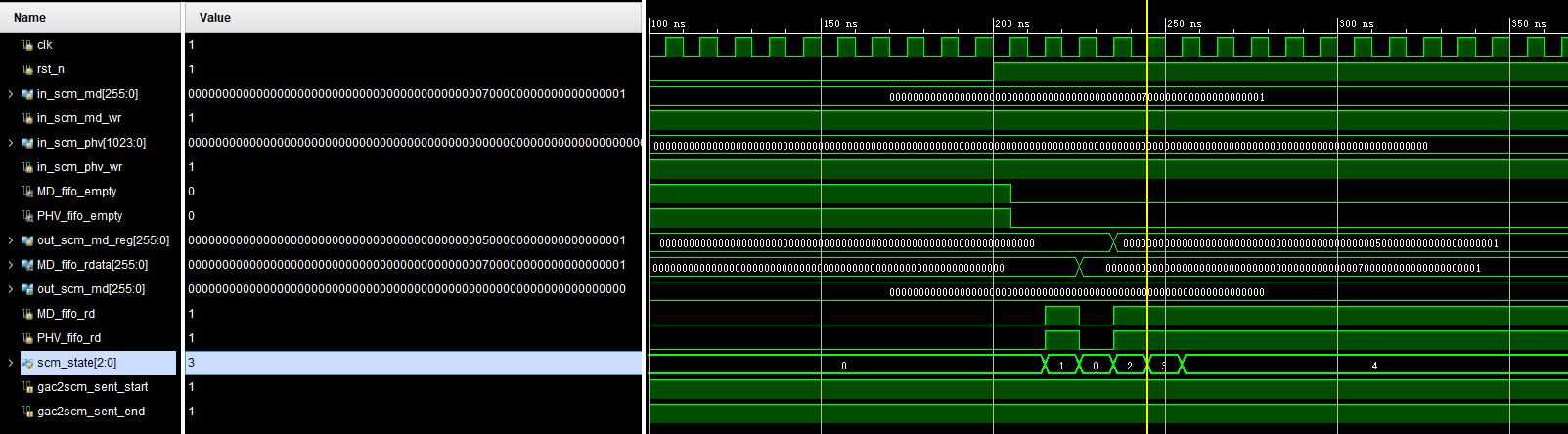


图3.12 CNT\_S跳转到WAIT\_S的仿真结果

从图3.12可以发现，状态从2 -> 3，即从CNT\_S -> WAIT\_S，状态转换正确。

1. CNT\_S -> SEND\_S

当未接收到测试停止的信号时，判断收到的报文是否符合测试需求，即判断MD中79：72位是否与软件端下发的protocol\_type相同。如果相同，则更新计数器，并将MD中第108位置为1，在GOE模块中丢弃（符合之前功能设计要求），跳转到SEND\_S，仿真结果如图3.13所示；否则，正常转发，跳转到SEND\_S，仿真结果如图3.14所示。

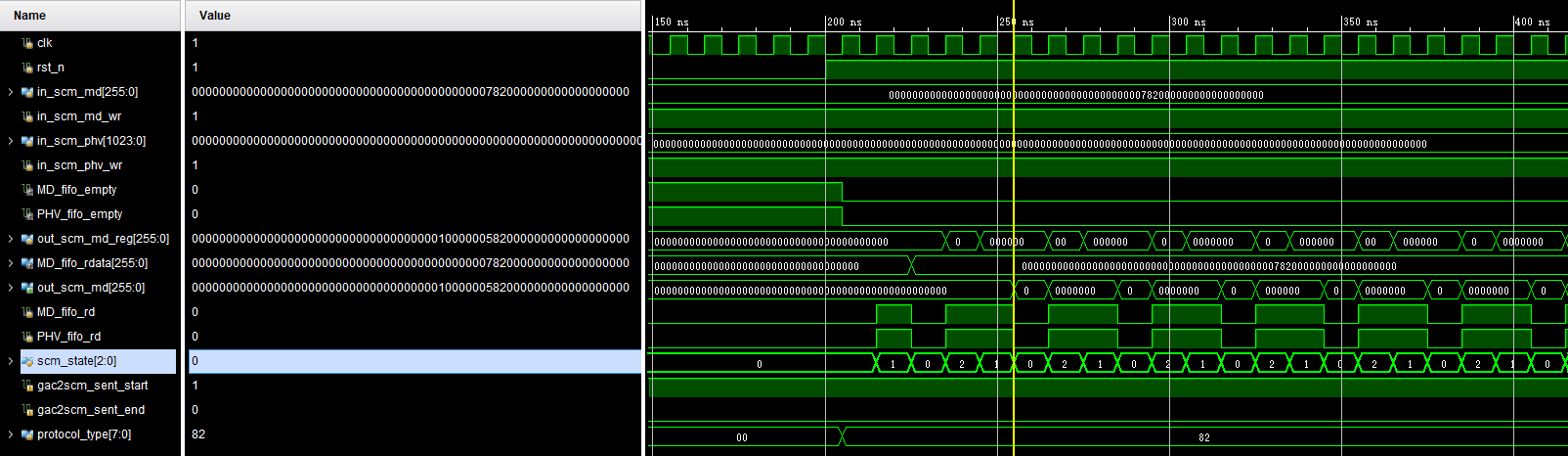


图3.13 符合测试需求时CNT\_S跳转到SEND\_S的仿真结果

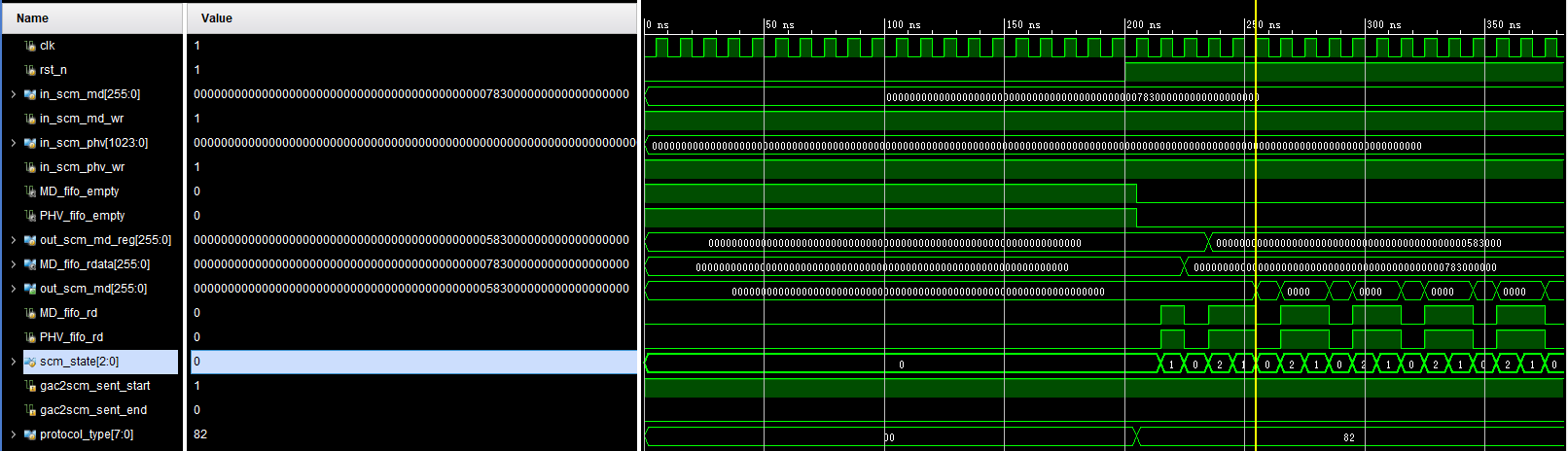


图3.14 不符合测试需求时CNT\_S跳转到SEND\_S的仿真结果

从图3.13可以发现，状态从2 -> 1，即从CNT\_S -> SEND\_S，状态转换正确。同时，收到的报文符合测试所需报文，并将MD的108位置1，在GOE中丢弃，符合预期设计。

从图3.14可以发现，状态从2 -> 1，即从CNT\_S -> SEND\_S，状态转换正确。但收到的报文不符合测试所需报文，正常转发，未将MD的108位置1，同样符合预期设计。

### 3.3.4 WAIT\_S：

1. WAIT\_S -> FETCH\_S

当在WAIT\_S内出现超时情况，即out\_scm\_md\_reg[31:0] > (end\_time + n\_RTT)时，状态跳转到FETCH\_S。仿真结果如图3.15所示。

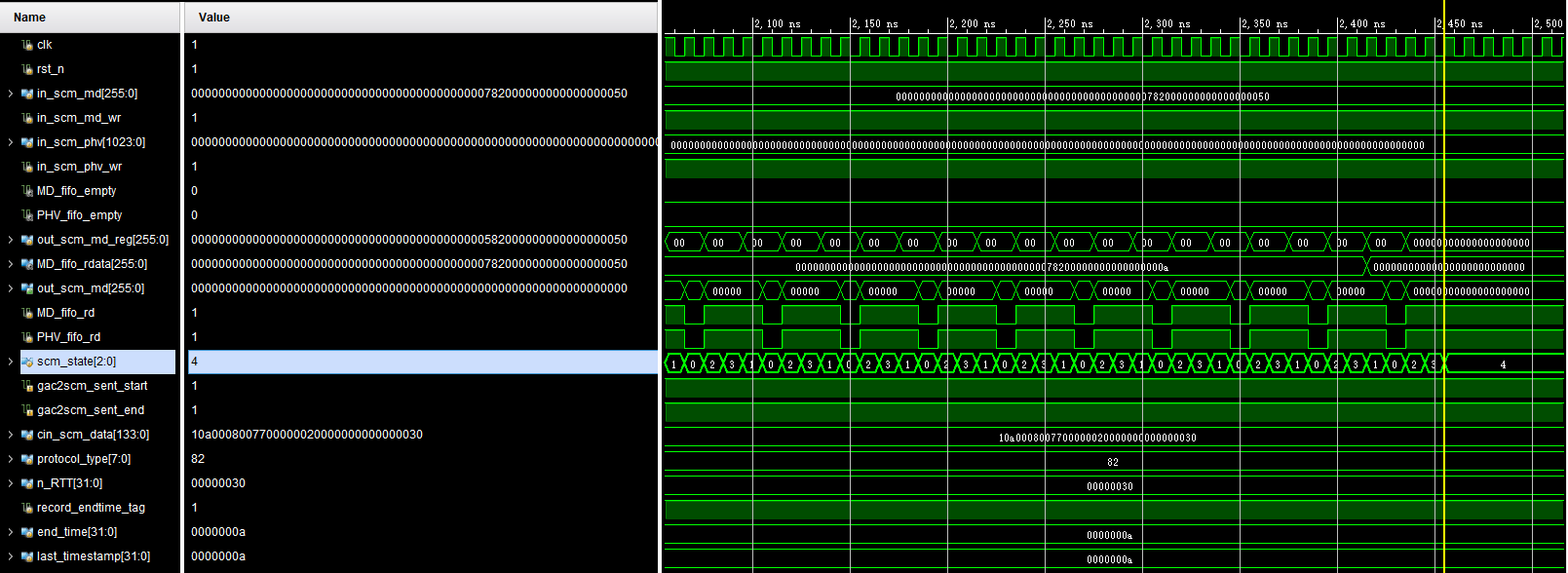


图3.15 WAIT\_S跳转到FETCH\_S的仿真结果

从图3.15可以发现，测试终止时间（end\_time）为8’h0000000a，等待时间间隔（n\_RTT）为8’h00000030，此时报文自身携带的时间戳为8’h00000050，所以未超时。状态从3 -> 4，即从WAIT\_S -> FETCH\_S。符合预期功能设计和状态转换。

1. WAIT\_S -> SEND\_S

当在WAIT\_S内未出现超时情况，即out\_scm\_md\_reg[31:0] < (end\_time + n\_RTT)时，判断接收到的数据包是否符合测试所需报文类型。

如果报文类型符合，则修改相关计数器值，并将输出MD的第108位置为1，跳转到SEND\_S。仿真结果如图3.16所示。

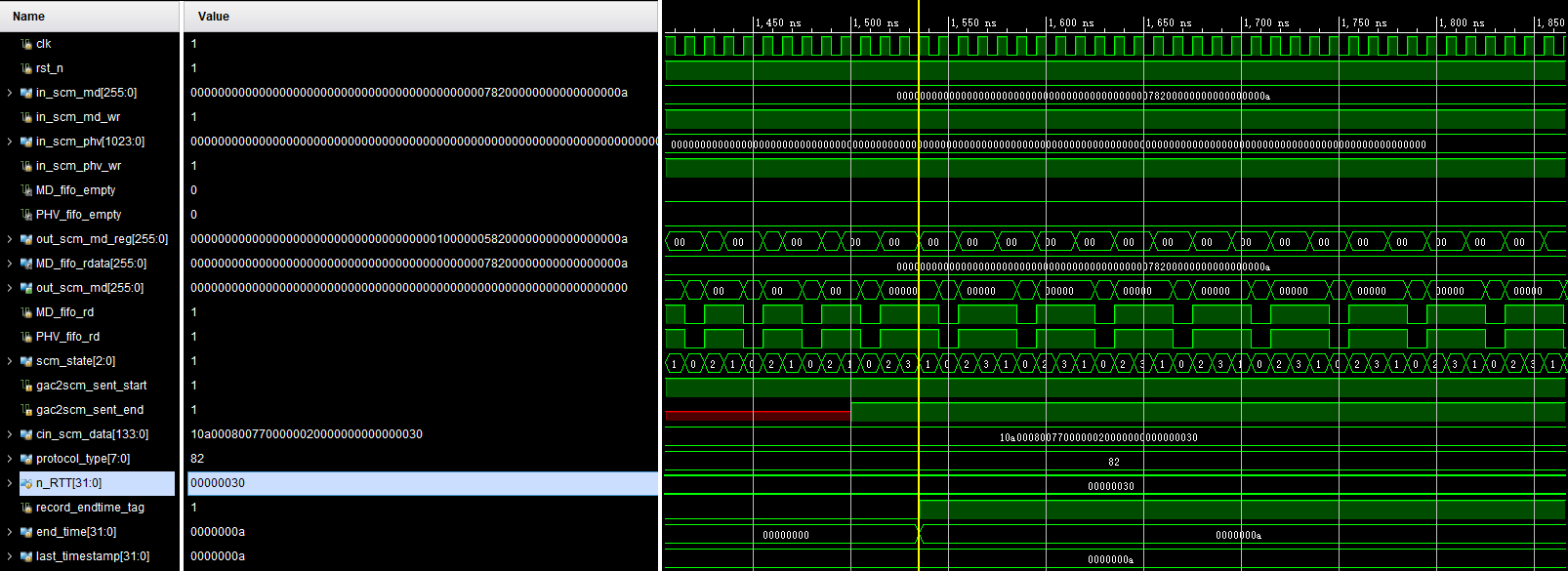


图3.16 WAIT\_S跳转到SEND\_S的仿真结果

从图3.16可以发现，状态从3 -> 1，即：WAIT\_S -> SEND\_S。同时，测试终止时间（end\_time）为8’h0000000a，等待时间间隔（n\_RTT）为8’h00000030，此时报文自身携带的时间戳为8’h0000000a，所以未超时，继续进行测量统计。符合预期功能设计和状态转换。

如果报文类型不符合，则进行转发，此处不再赘述。

### 3.3.5 FECH\_S：

1. FETCH\_S -> IDLE\_S

当收到软件端下发的测试重置信号statistic\_reset时，跳转到IDLE\_S状态。仿真结果如图3.17所示。

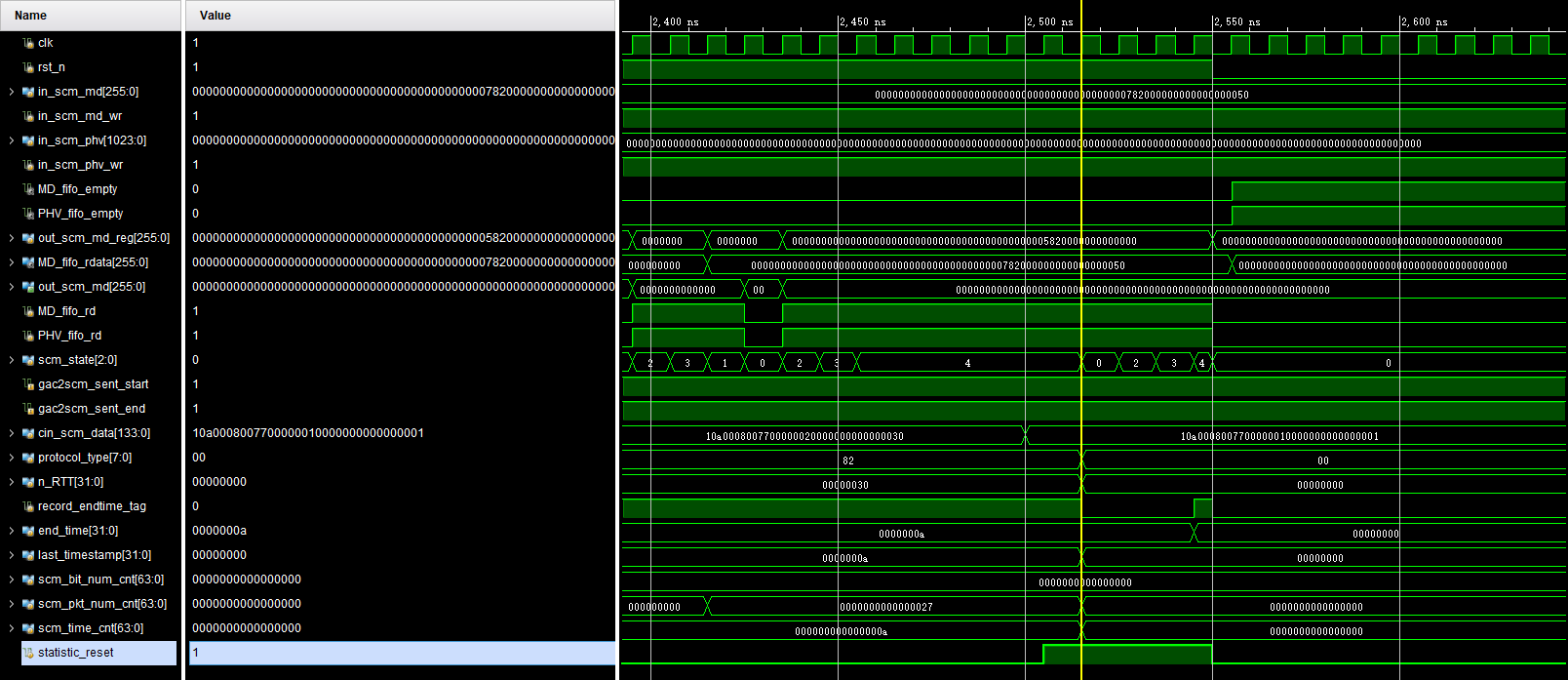


图3.17 FETCH\_S跳转到IDLE\_S的仿真结果

从图3.17可以发现，状态从4 -> 0，即从FETCH\_S -> IDLE\_S。收到重置信号后，所有测试相关的计数器均置0。符合预期功能设计和状态转换。

1. FETCH\_S -> FETCH\_S

当未收到软件端下发的测试重置信号statistic\_reset，则一直处于FETCH\_S。仿真结果如图3.18所示。

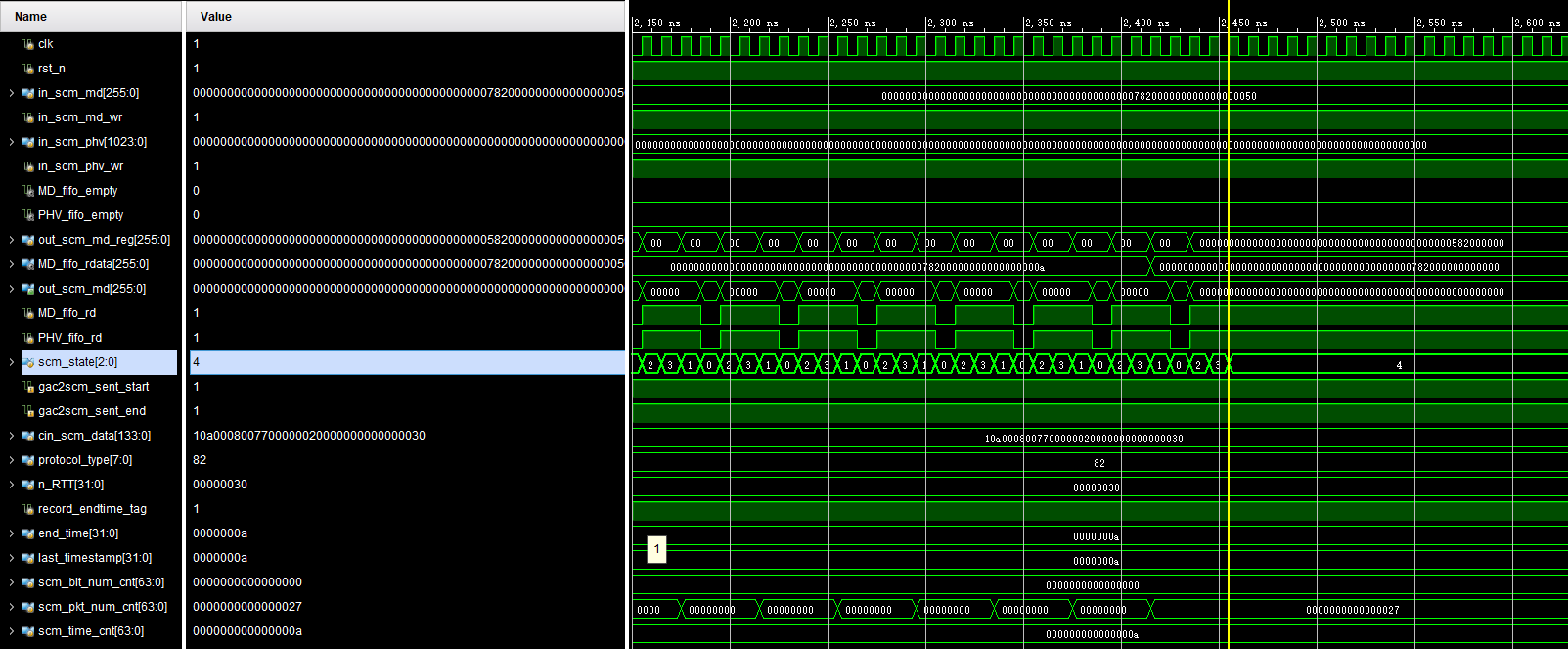


图3.18 FETCH\_S跳转到FETCH\_S的仿真结果

从图3.18可以发现，一直处于状态4，即：FETCH\_S。同时，各计数器的值保持不变，等待软件端读取。符合预期功能设计和状态跳转。