**SystemVerilog 第三次实验报告**

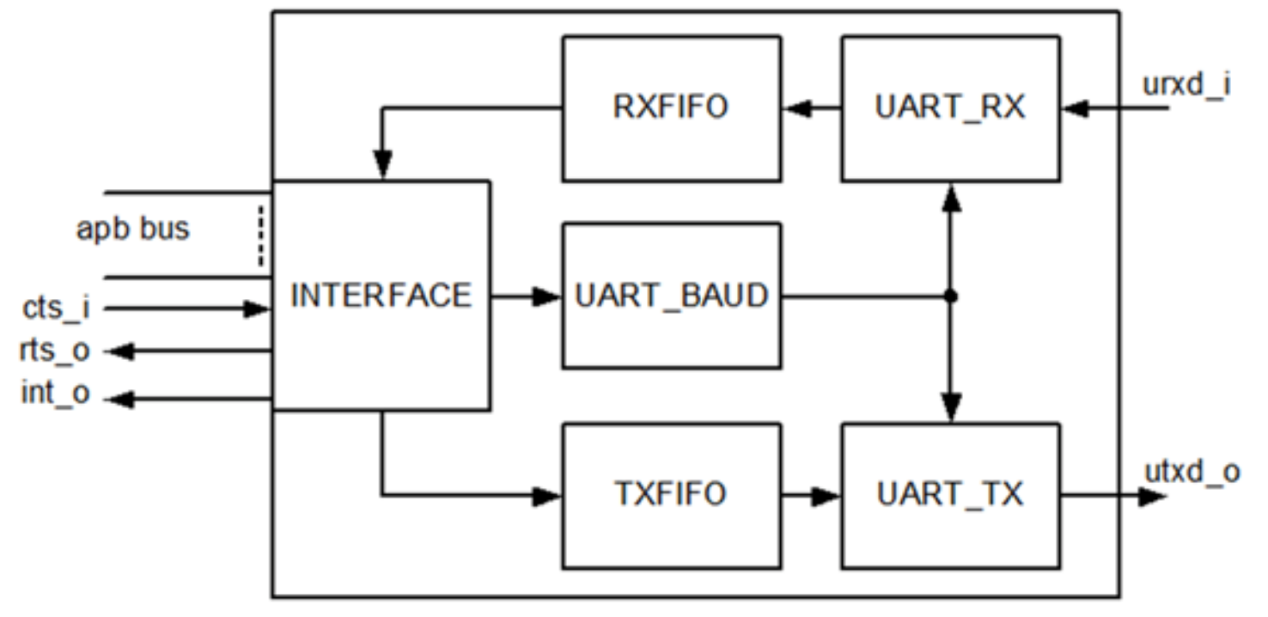
**赵拯基**

**B2103994 121039910125**

**2022.1.25**

# 实验概览

## DUT 设计描述



DUT将UART模块连接到APB总线上，通过配置DUT内部的寄存器可以设置UART TX端口的输出模式和 UART RX端口的输入模式。

DUT内部分为四个部分：寄存器配置（UART\_REG\_IF）、波特率时钟产生（UART\_BAUD）、TX输出（UART\_TX）、RX输入（UART\_RX），其中TX输出模块和RX输入模块各自包含了一个FIFO（UART\_FIFO）。依据AMBA 2.0协议，可以对DUT内部的寄存器进行读写，包括写入TX端口要输出的数据、读取RX端口要输入的数据、功能设置以及状态读取。

关于DUT有一下几点值得注意：

* **数据位宽**。APB的地址、数据总线都是32位。模块内部进行寄存器读写时仅截取了APB总线输入地址的一部分，如要进行非法地址需注意；APB总线输入数据往往仅有部分有效。
* **连续数据发送**。根据DUT的设计，其FIFO的写使能拉高后在一个时钟周期后才能再次拉高。意味着，当APB总线向地址0x00连续写数据时，不能在ENABLE阶段后再次进入到SETUP阶段。两次连续的向地址0x00写数据之间必须间隔一个IDLE阶段。
* **奇偶校验位。**文档中的奇偶检验位说明反了，按位异或得到的是偶校验的结果。

## TESTBENCH 设计描述

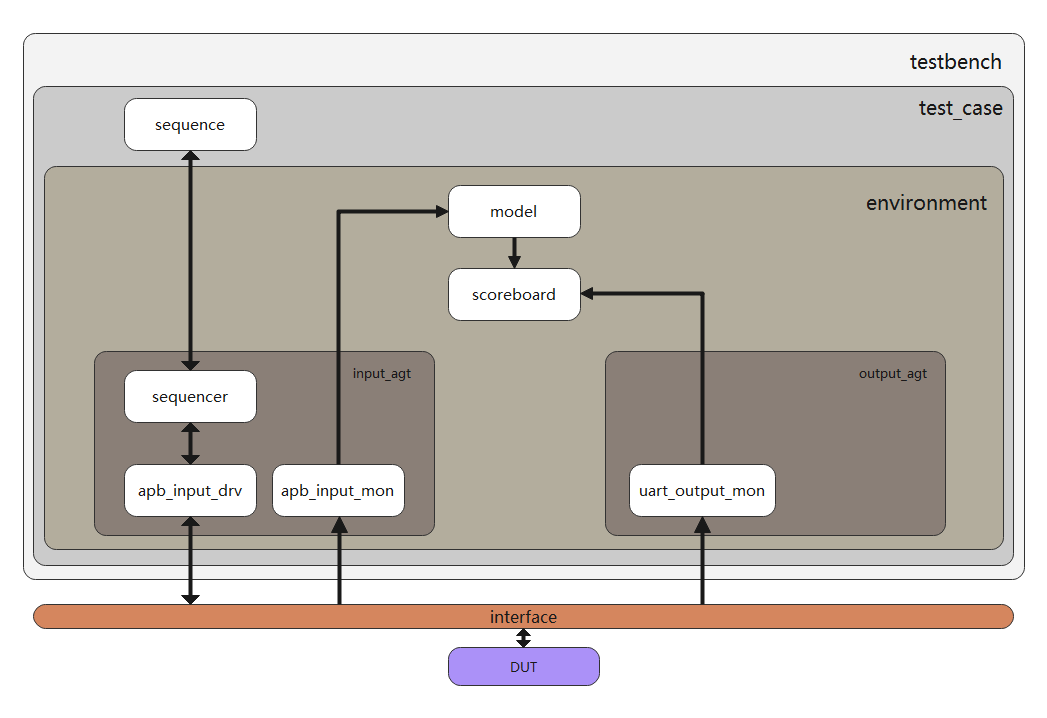


图 1 验证平台的结构

使用UVM搭建测试平台，UVM包含的主要component如图1所示，验证DUT内部寄存器的读写以及APB总线到UART TX的数据通路。整个测试平台包含多个不同的test\_case，由QuestaSim的启动脚本决定进行测试的test\_case。启动脚本通过-sv\_seed参数设置了初始化的随机数种子，使得每次运行相同的test\_case时可以生成不同的随机数序列，避免了systemverilog伪随机数的缺点。

除了上述基本的UVM component，还加入了**寄存器模型**，通过**后门访问**的方式监控内部寄存器的值（后门访问只读不写）；加入了driver和sequence的**response机制**，用于更好的处理中断，当driver检测到中断时告知sequence，使sequence产生用于处理中断的激励；加入了两种不同的component之间的沟通机制，用于更好的传递信息。更多具体细节请参考后续内容。

# 实验内容

## TESTBENCH COMPONENT具体描述

### INPUT\_AGENT



图 2 input\_agt 定义

如图2所示，input\_agt内部包含了apb\_input\_drv、apb\_input\_mon和sequencer，还包含一个uvm\_analysis\_port 端口指针，指向apb\_input\_mon到model的端口（其目的是在env保持同样的层级关系）。所有的类都在build\_phase中进行初始化，所有的端口都在connect\_phase中进行连接。

#### APB\_INPUT\_DRIVER APB总线输入驱动

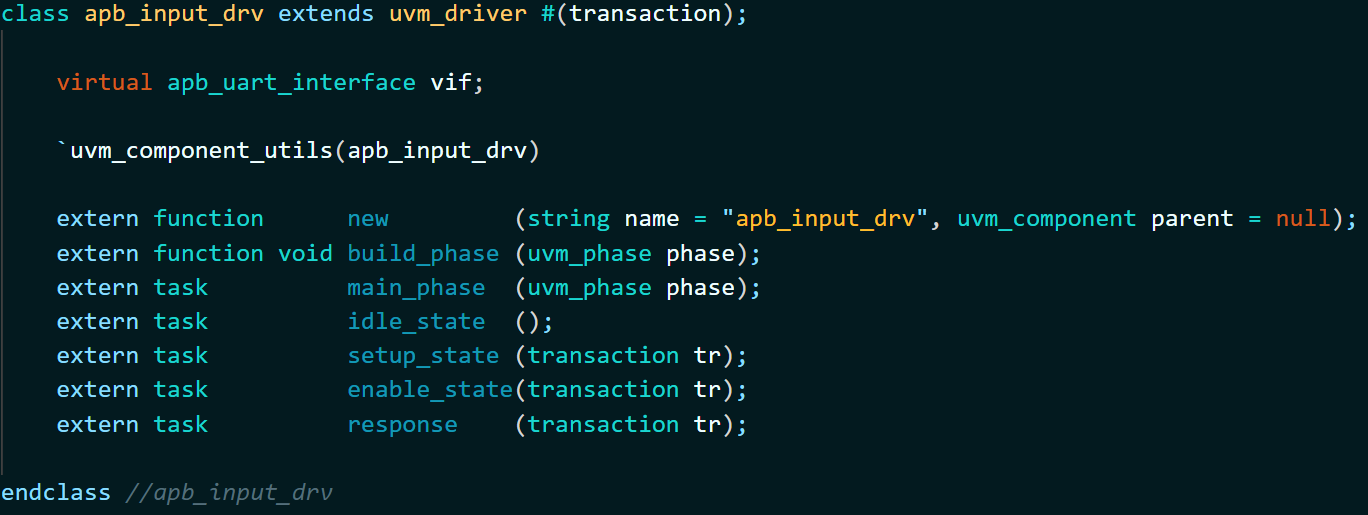


图 3 apb\_input\_drv定义

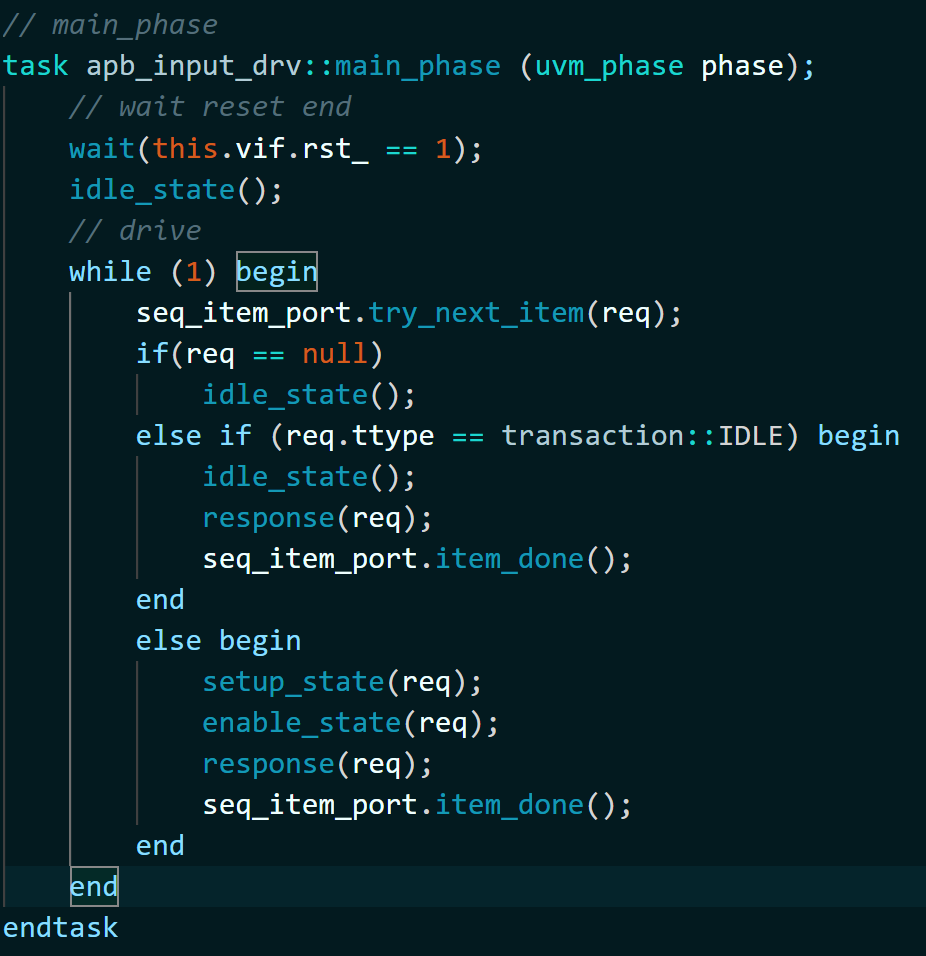


图 4 任务main\_phase

apb\_input\_drv定义如图3所示。vif为实例化的interface，在build\_phase阶段通过uvm\_config\_db获取。任务idle\_state、setup\_state、enable\_state、response为自定义任务，前三者用于模拟APB总线的行为，response用于向sequence发送信息。

main\_phase 为主要的执行任务，如图4所示。当系统复位后，获取来自sequence的激励，采用非阻塞的方式。若可以获取激励，并且激励有效则驱动，同时监测中断是否拉高，最后返回sequence消息，由此循环往复。

#### APB\_INPUT\_MONITOR APB总线输入监控

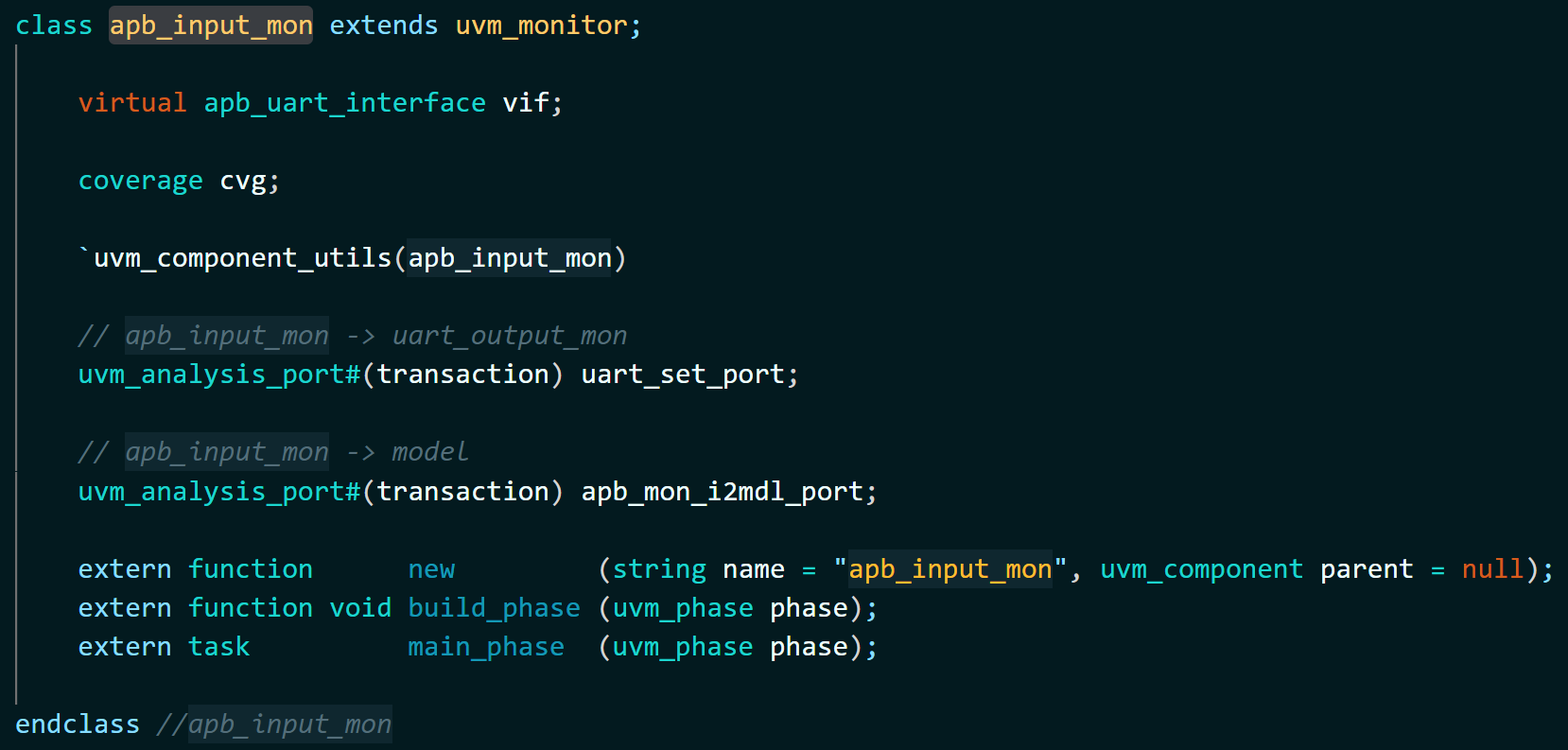


图 5 apb\_input\_mon定义

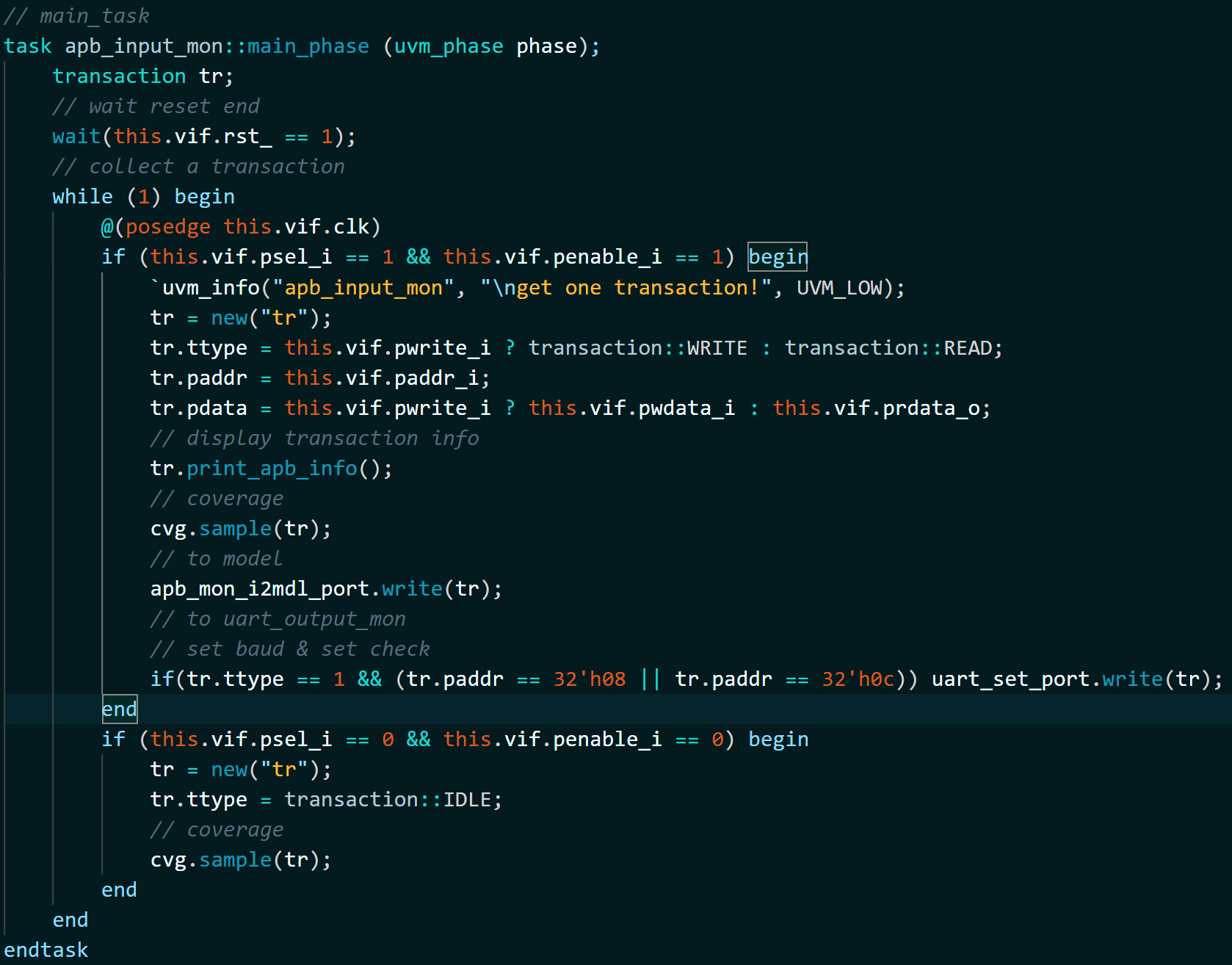


图 6 main\_phase

如图5所示，覆盖率类在apb\_input\_mon 中实例化，apb\_input\_mon检测APB总线的状态，所有的激励（transaction）都会被apb\_input\_mon获取，因此可以在此采样到所有的激励用于覆盖检测。同样的，vif为实例化的interface。其内部还声明了两个端口，uart\_set\_port用于发送UART TX的输出设置给uart\_output\_mon，apb\_mon\_i2mdl\_port用于将所有的激励发送给model。

main\_phase 为主要的执行任务，如图6所示。时钟上升沿时继续采样，主要关注IDLE阶段（对应的激励状态为idle）和ENABLE阶段（可获取有效的激励，对应状态为READ和WRITE）。把所有的有效激励发送给model，同时根据地址判断是否为UART TX输出设置，若是则将激励信息发送给uart\_output\_mon，对所有的有效激励进行覆盖率采样。

#### SEQUENCER

最基本的sequencer组件设计，仅继承原有UVM的uvm\_sequencer类。

### OUTPUT\_AGENT

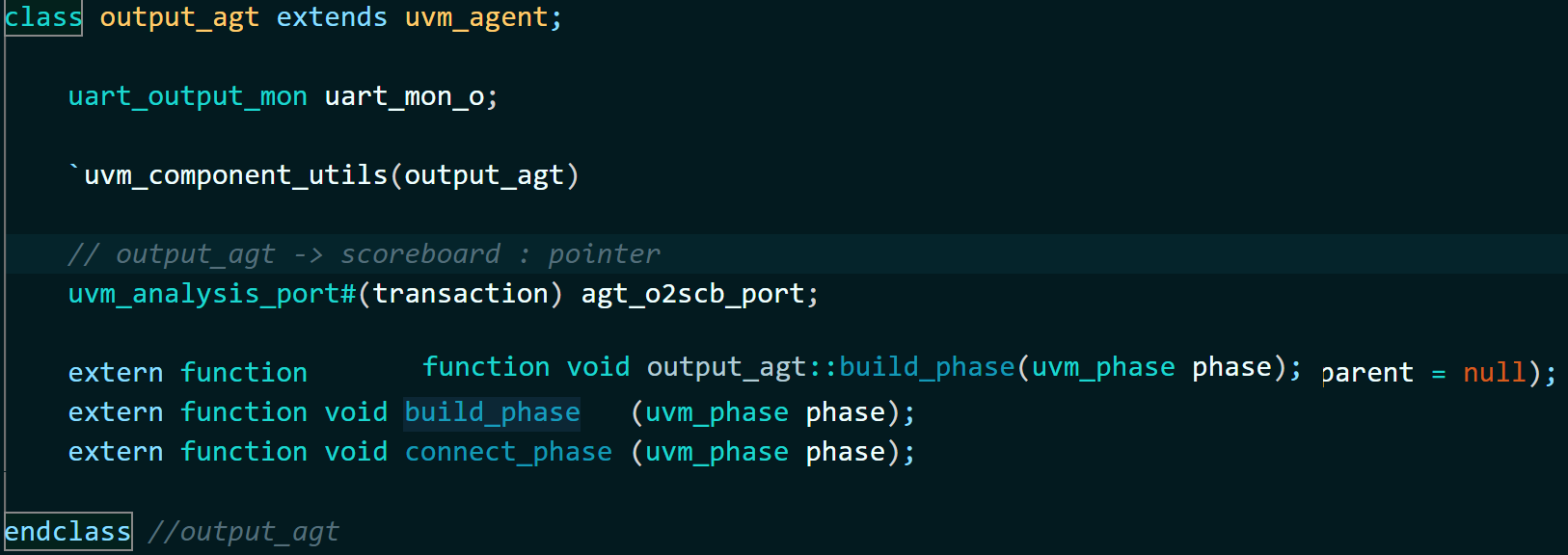


图 7 output\_agt 定义

如图7所示，output\_agt内部仅包含uart\_output\_mon。和input\_agt，其也有一个uvm\_analysis\_port端口指针，指向uart\_output\_mon到scoreboard的端口。

#### UART\_OUTPUT\_MONITOR UART输出监控

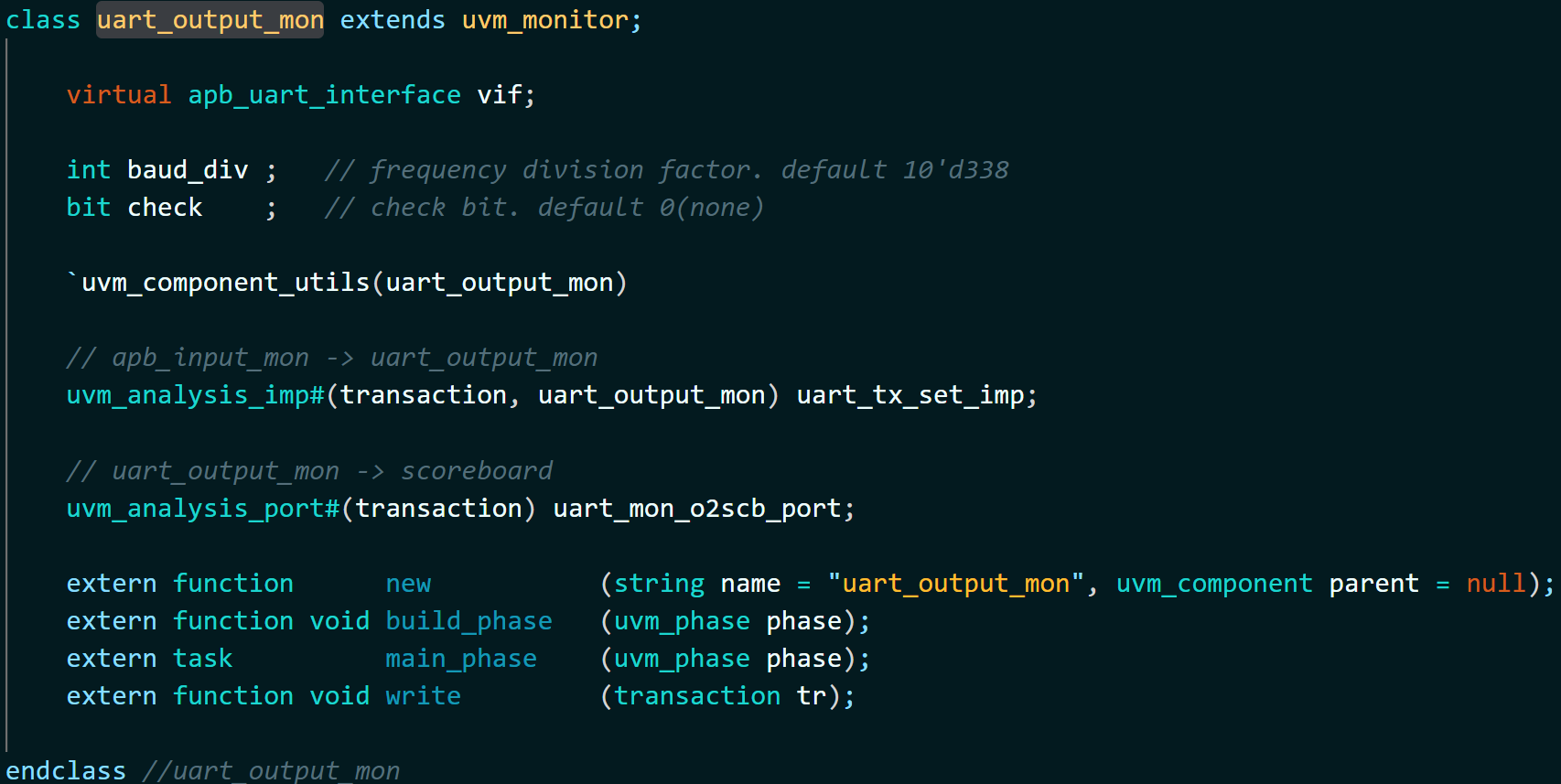


图 8 uart\_output\_mon定义

如图8所示，uart\_output\_mon内部包含两个端口，uart\_tx\_set\_imp接受UART TX设置端口，uart\_mon\_o2scb\_port用于将收到的URAT数据发送给scoreboard。baud\_div为波特率的分频系数， check为是否设置校验位，初始值与DUT保持一致，通过write函数进行更新。vif为实例化的interface。

值得一提的是，apb\_input\_mon与uart\_output\_mon的通信方式不同其他模块，采用IMP而不是FIFO，uart\_tx\_set\_imp完全从属于uart\_set\_port，当uart\_set\_port发送数据时，uart\_tx\_set\_imp必须要接受，以此保证功能设置的实时性。

main\_phase 为主要的执行任务。检测UART TX的下降沿，当检测到时意味着数据传输的开始，除去第一位的起始位总共有8位的有效数据（如果有校验位则是9位）。在数据传输的中间进行采样（根据波特率的分频系数进行计数），有效数据采集完成后，则等待下一个下降沿的到来。同时将数据发送给scoreboard。

### MODEL



图 9 model定义

如图9所示，model内部仍需要记录当前DUT是否设置校验位以及校验方式（奇校验、偶校验）用于模仿DUT的行为。对于数据发送激励，即UART TX端口发送数据，model则要根据当前的设置计算数据的校验位，对于配置寄存器激励（包括寄存器的读写），如果为设置UART TX端口是否有校验位以及校验方式，model则需要记录。

APB总线与UART分属两个时钟域，时间跨度很大，因此，将数据发送激励和配置寄存器激励通过两个端口发向两个通向scoreboard的FIFO。

### SCOREBOARD

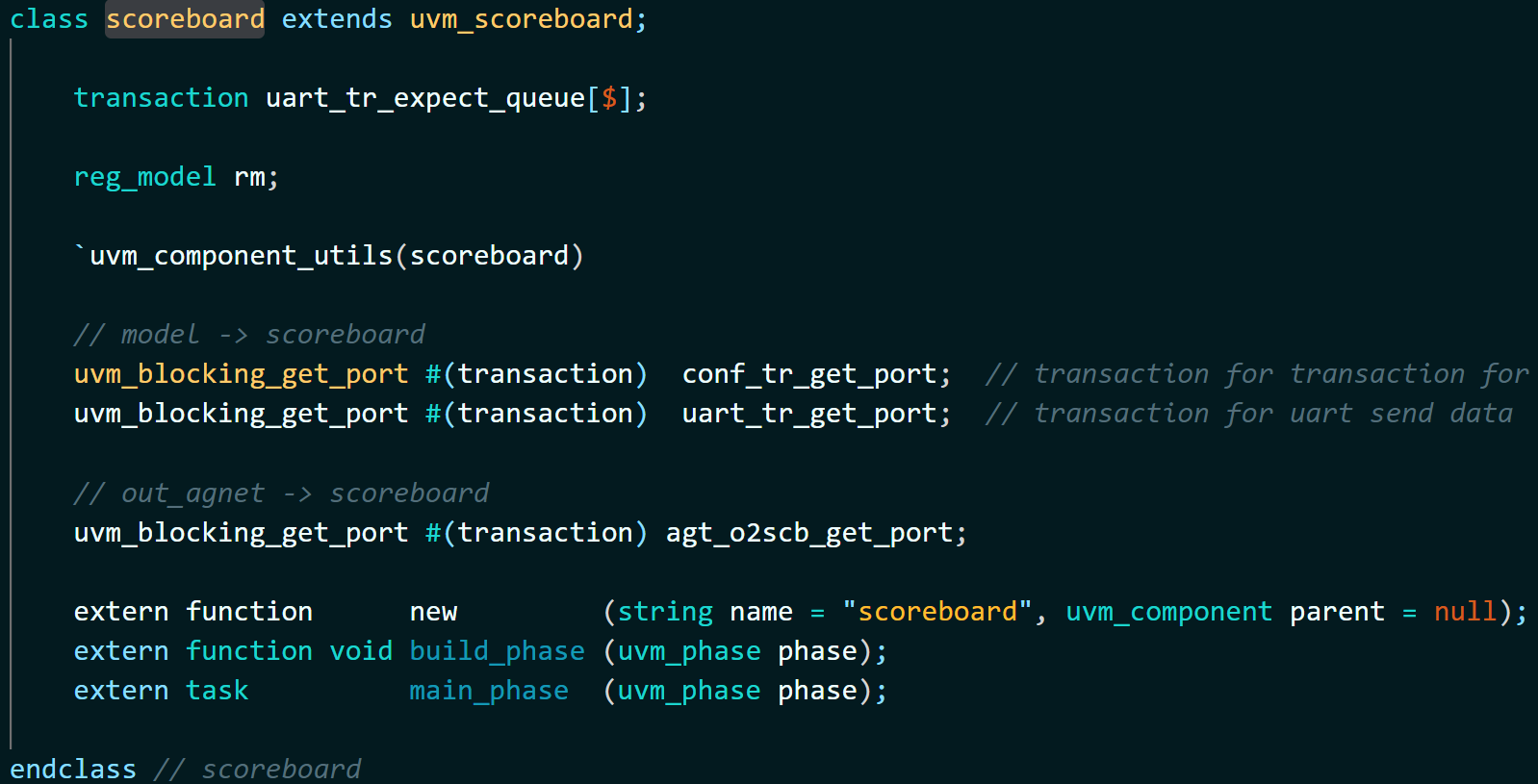


图 10 scoreboard 定义

如图10所示，scoreboard有三个用于接受数据的端口；从uart\_tr\_get\_port、agt\_o2scb\_get\_port获取的数据进行比较，比较数据位和校验位。从conf\_tr\_get\_port获取的数据通过寄存器模型进行判断，判断该激励对应的操作是否正常。main\_phase 为主要的执行任务，并行获取数据并比较（uart\_tr\_get\_port的数据时间跨度很大，保证conf\_tr\_get\_port的数据执行比对不会被阻塞）。

### ENVIRONMENT

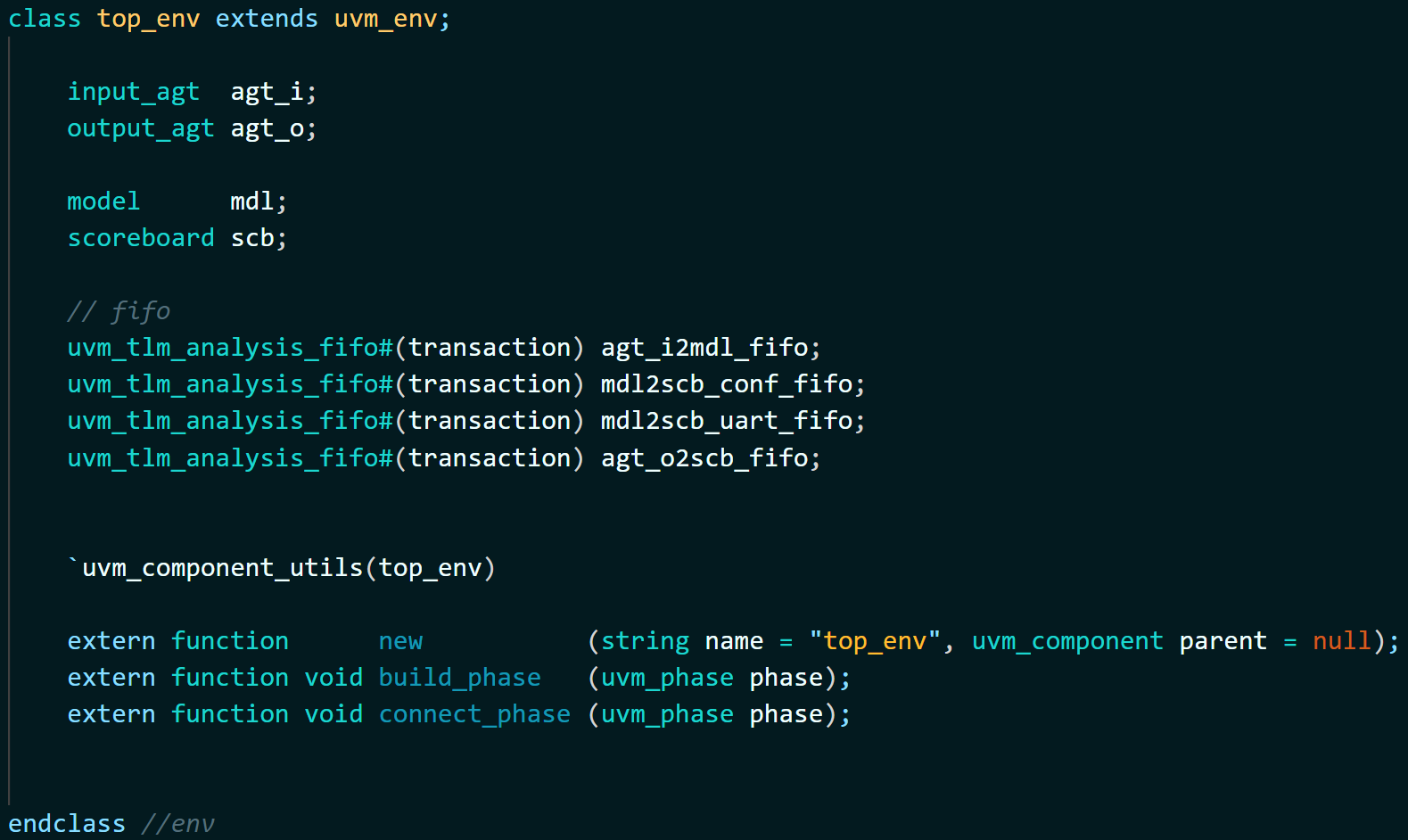


图 11 env 定义



图 12 组件实例化，端口连接

对基本模块进行实例化，对端口进行连接。

## TESTBENCH OBJECT具体描述及其他

### INTERFACE 接口定义

将DUT顶层的端口按照传输协议分为两组clocking port：apb\_port，包含addr\_i, pwdata\_i, psel\_i, penable\_i, pwrite\_i，uart\_int\_o, prdata\_o，采样时钟clk；uart\_port，包含urxd\_i，utxd\_o，采样时钟clk26m。

### TRANSACTION 激励定义

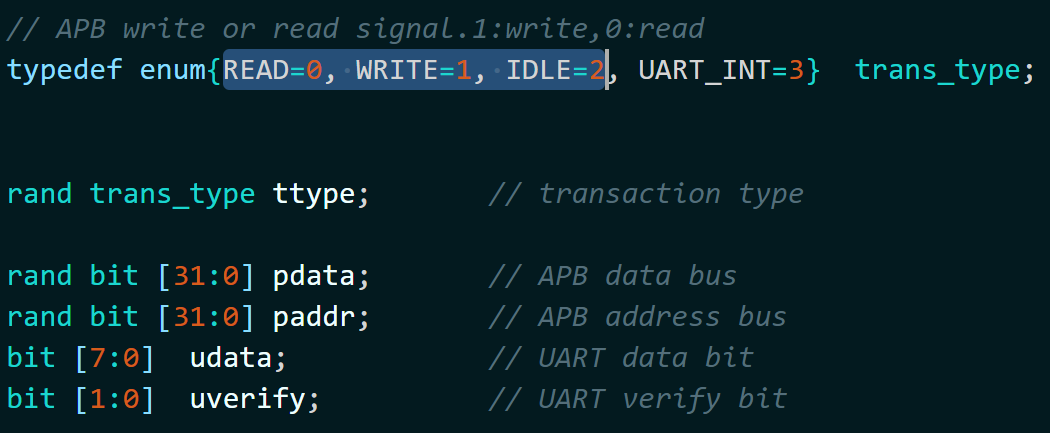


图 13 transaction 变量

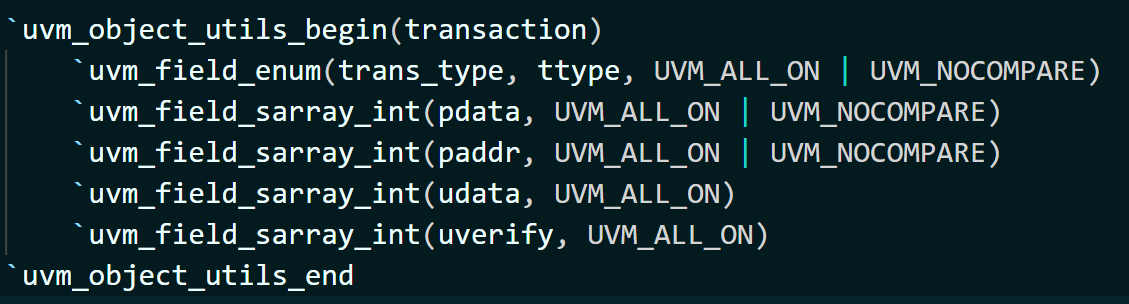


图 14 transaction filed\_automation注册

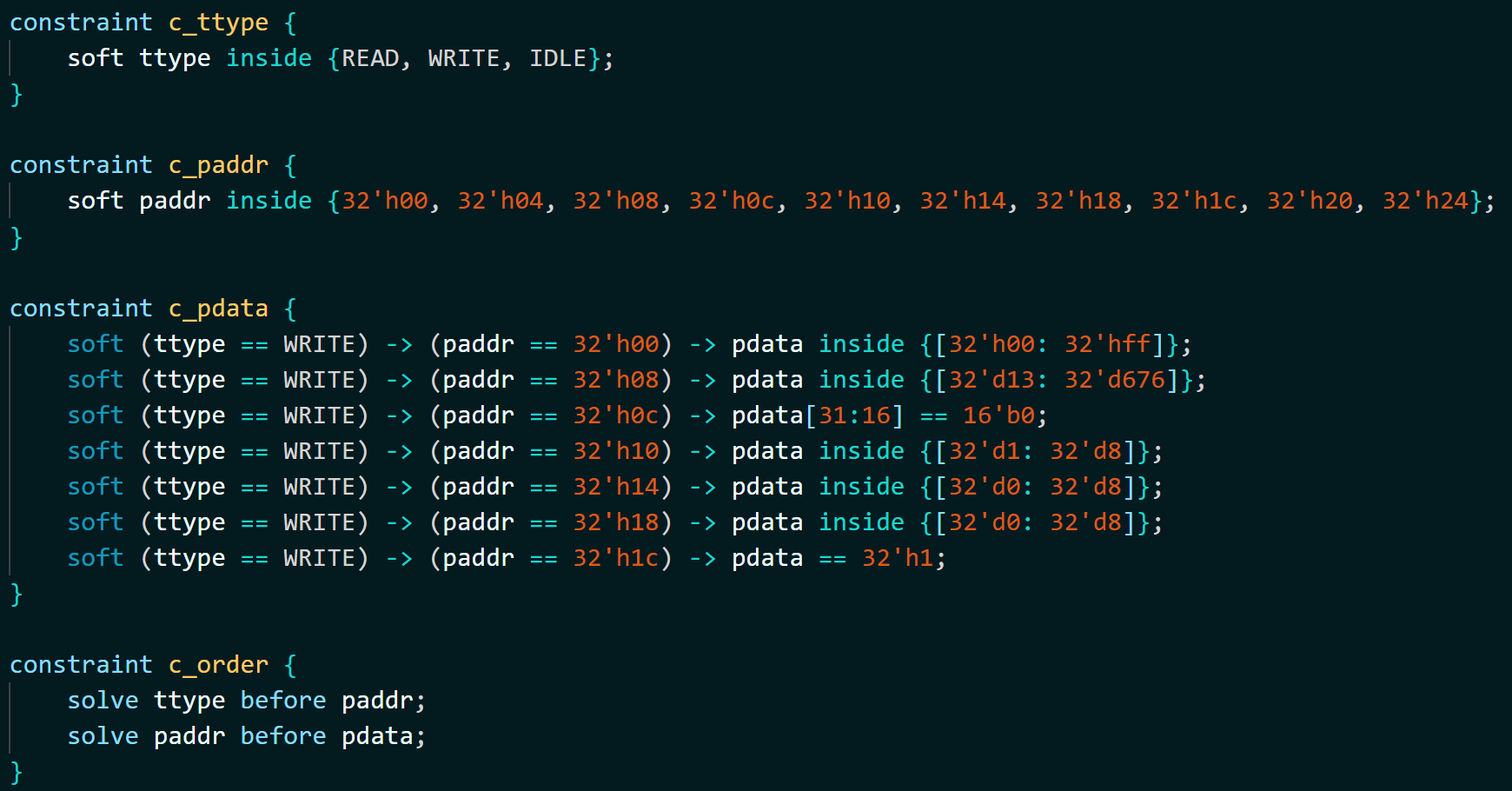


图 15 transaction 约束

一个transaction实例包含了一个完整的APB总线操作所需要的信息，即总线状态（READ, WRITE, IDLE）、地址、数据。如图13所示，ttype表示激励的类型；paddr表示地址；pdata表示数据，若是写激励则为随机的写数据，若是读激励则为从总线上读到的数据。udata表示从UART TX接收的数据（不是所有激励都会设置访问这个字段），uverify表示验证状态，高位表示是否有校验位，高有效；低位表示校验位数据，仅当高位为1时此位有效。

scoreboard进行比较时仅比对udata和uverify，故如图14所示，其他字段filed\_automation注册时增加UVM\_NOCOMPARE宏定义。

随机产生激励时，激励的状态只能是READ, WRITE, IDLE三者之一，UART\_INT状态用于response机制。所有的地址均是合法的。当激励为WRITE时，对特定地址的写数据进行约束。如图15所示。

### REG\_MODEL 寄存器模型

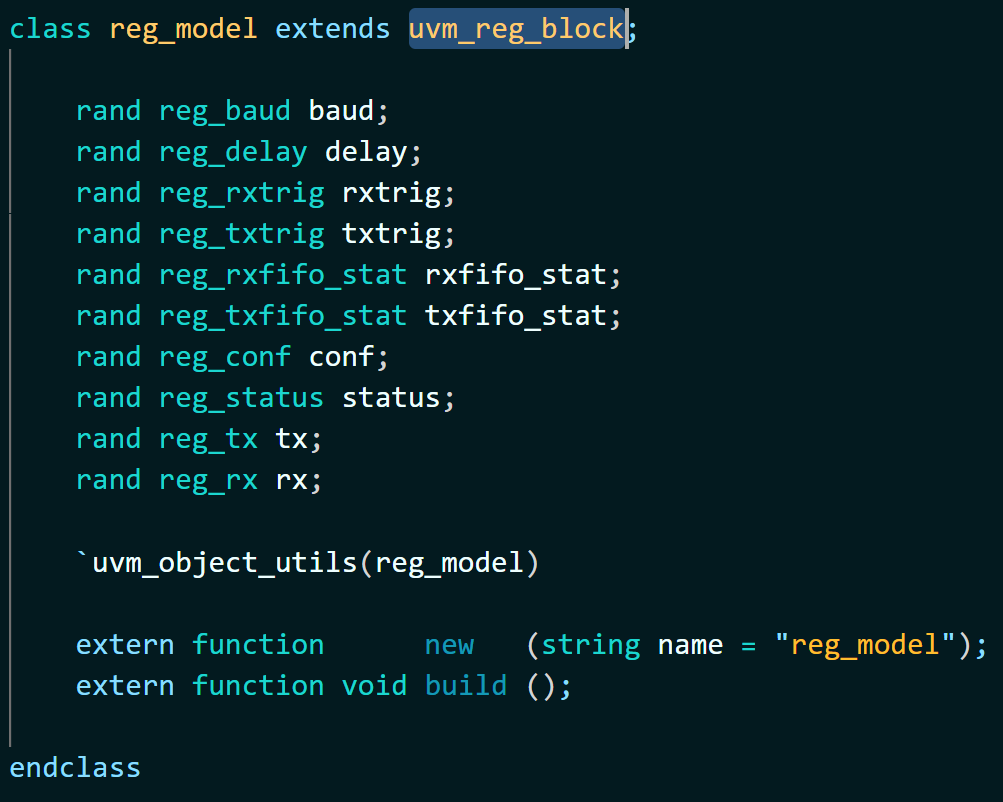


图 16 reg\_model 定义

每一个合法地址都对应一个uvm\_reg。每一个合法地址内部可能有多个uvm\_reg\_field，例如功能模式寄存器和状态寄存器。所有的uvm\_reg组合起来成为uvm\_reg\_block，如图16所示。采用后门访问，不占用APB总线，不消耗仿真时间，但需要设置寄存器的访问路径。

## TEST\_CASE 具体描述

### 断言

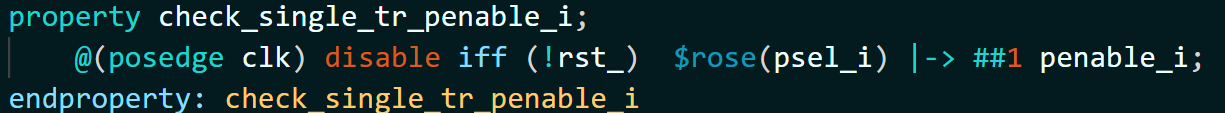


图 17 单个激励

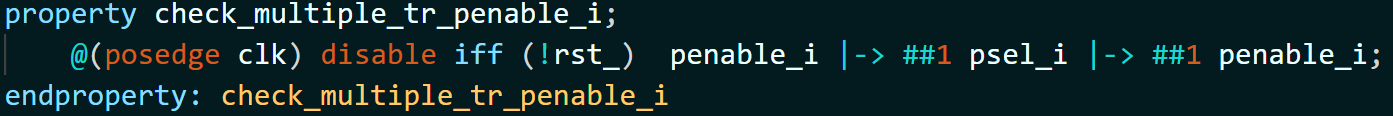


图 18 连续两个激励

* 对单个激励（单个数据传输）进行断言，如图17所示。
* 对连续两个激励（burst2）进行断言，如图18所示。

#### 结果

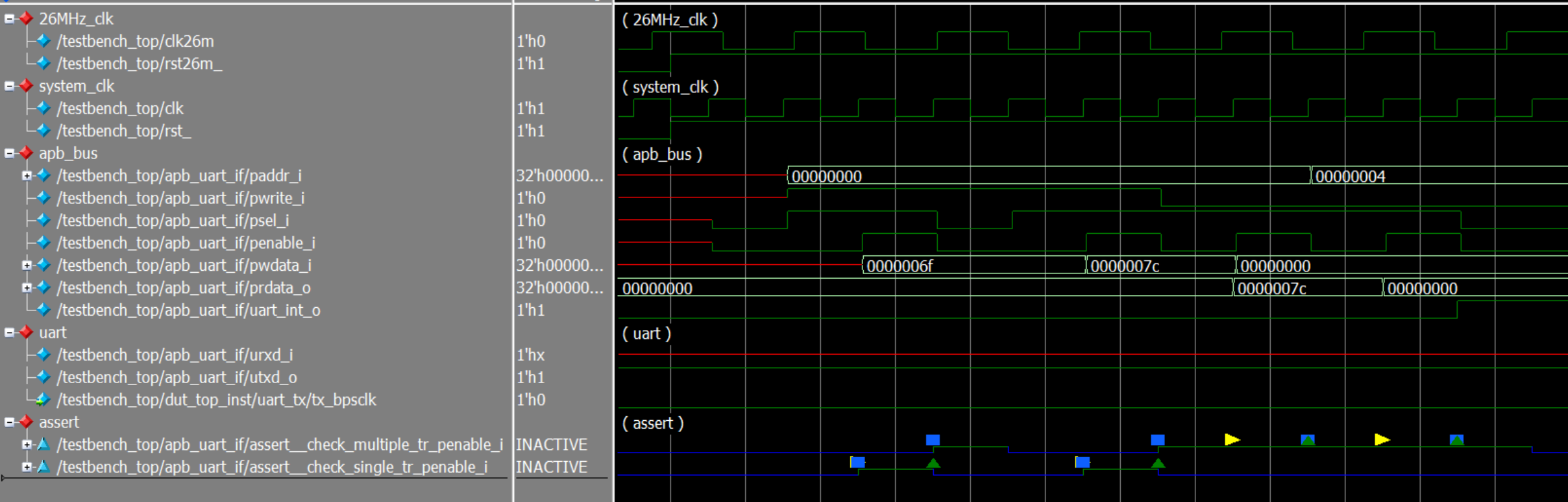


图 19 断言结果

### COVERAGE 覆盖率



图 20 覆盖率定义

如图20所示的覆盖率定义，测试内容包括总线状态、波特率设置、是否设置校验位、校验类型、是否有停止位、TX的触发深度、帧间隔以及是否进行中断操作。所有的测试点都覆盖了全部的初始化随机值。因为仅验证APB总线到UART TX端的数据通路，所以没有对UART RX端相关的寄存器进行覆盖率测试。

#### 结果

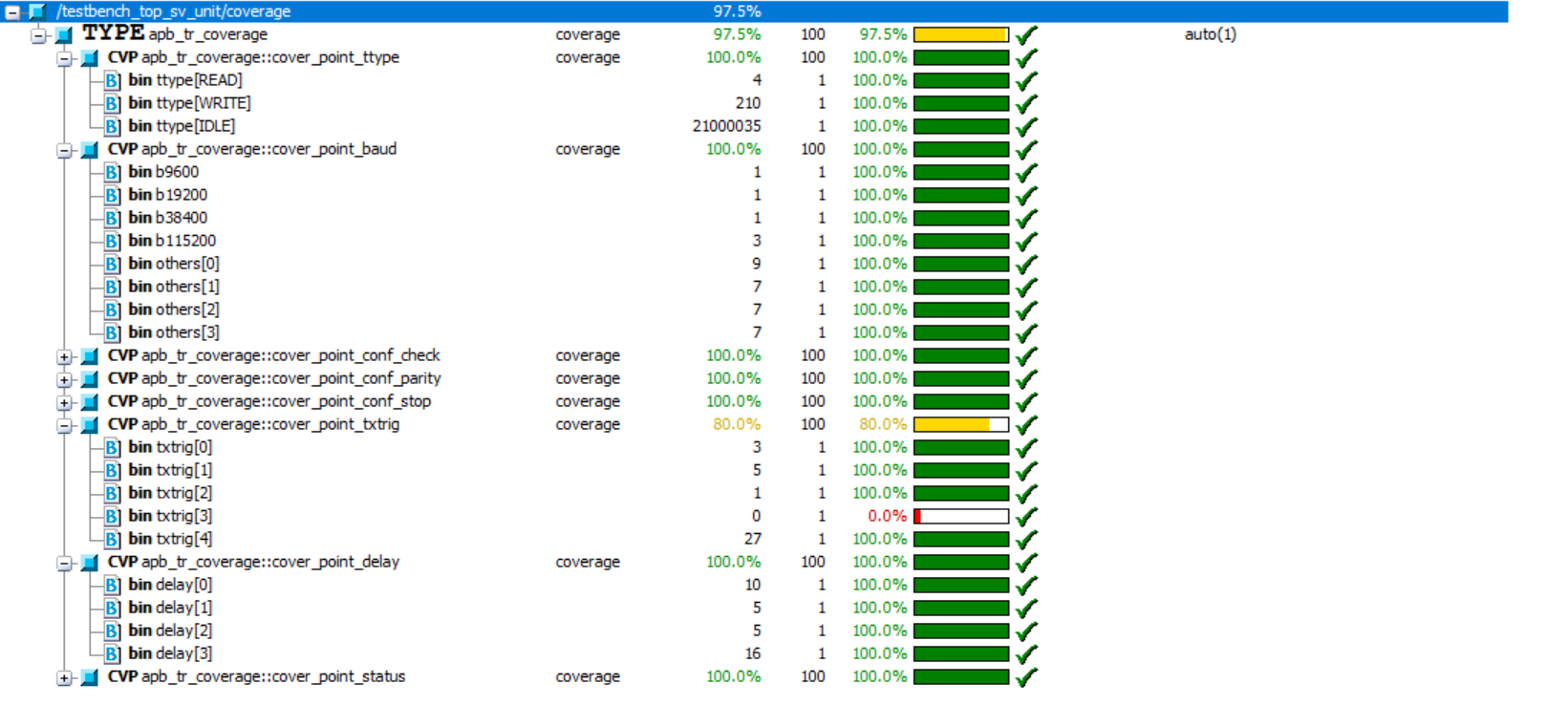


图 21 覆盖率结果

### TEST\_CASE\_0

寄存器读写测试

**测试启动**：测试脚本设置set TEST test\_case0

**测试内容**：通过 APB 总线配置 UART 模块寄存器，所有控制寄存器的读写测试、所有状态寄存器的读写测试。所有寄存器先写后读，包含对只读寄存器的非法写操作

**测试结果**：读写值正确，测试通过

#### 结果

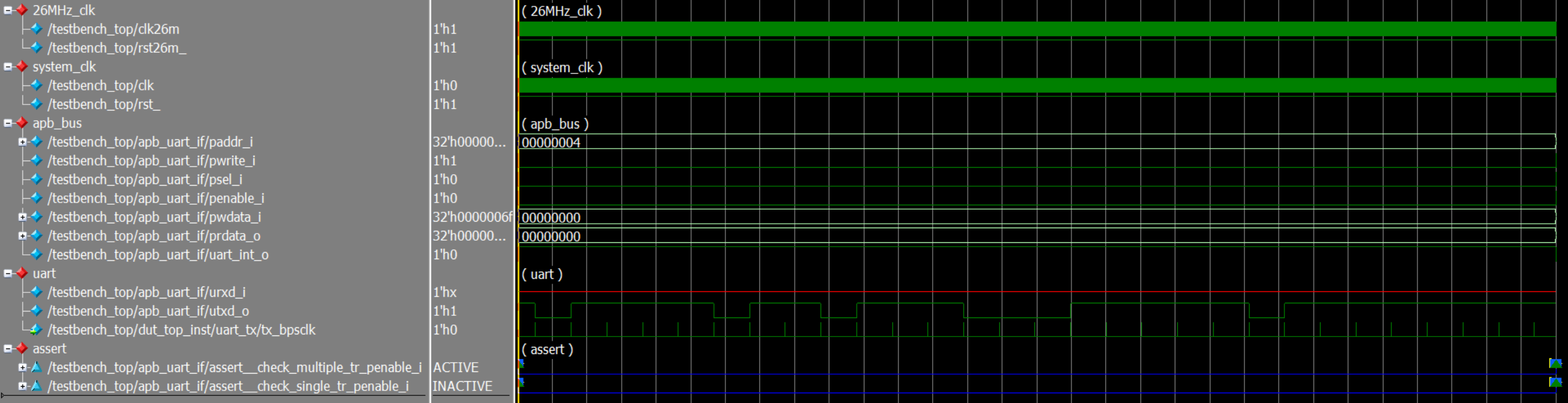


图 22 test\_case0结果

### TEST\_CASE\_1

寄存器稳定性测试

**测试启动**：测试脚本设置set TEST test\_case1

**测试内容**：对读写寄存器保留域进行读写，对只读寄存器进行写操作

**测试结果**：通过写入和读取，读写寄存器值是预期值不紊乱，同时非法操作不会影响 UART功能，测试通过

**注：跟据DUT内部逻辑，其仅截取部分总线地址作为内部寄存器的索引，非法地址的写入操作必定会引起内部值的变化（例如0x01非法地址，对于DUT来说其等价于0x00）**

#### 结果

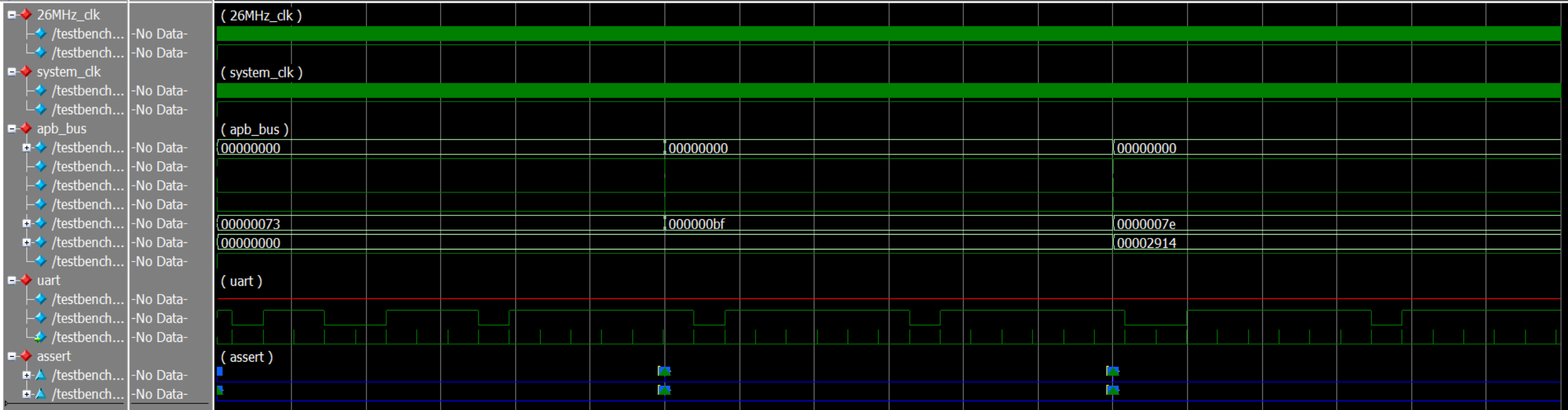


图 23 test\_case1结果

### TEST\_CASE\_2

UART模块基础功能测试

**测试启动**：测试脚本设置set TEST test\_case2

**测试内容**：配置UART基础功能寄存器，波特率设置115200，无奇偶校验位，有停止位，TX触发深度为0，帧间隔为2，之后向 TX FIFO 写入发送的数据

**测试结果**：UART模块TX端信号线发送的字符格式与配置相同，测试通过

#### 结果

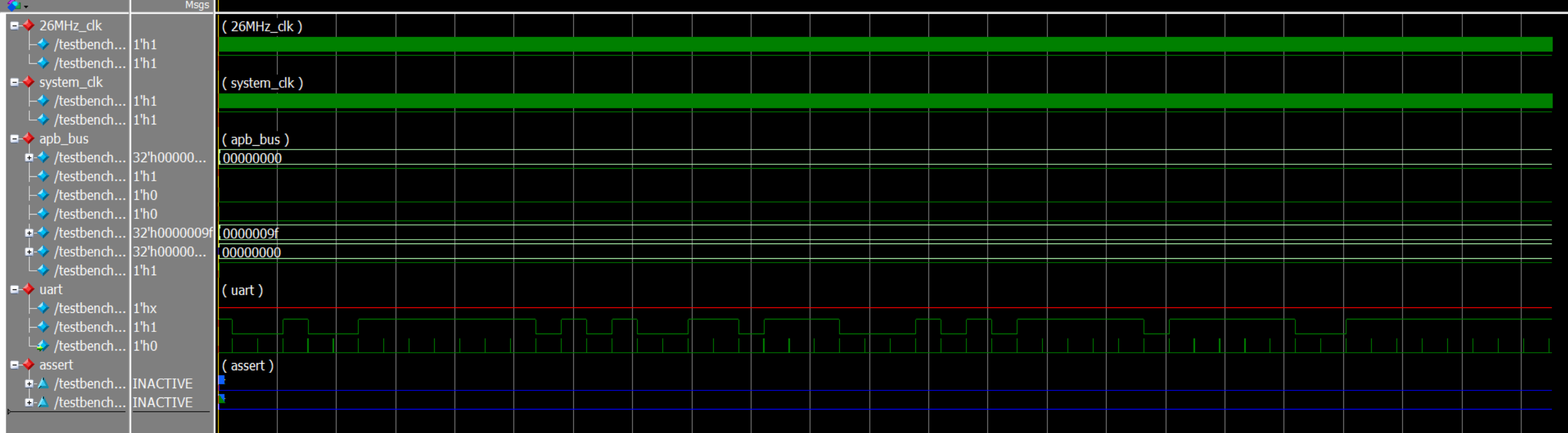


图 24 test\_case2 结果

### TEST\_CASE\_3

UART特殊功能测试

**测试启动**：测试脚本设置set TEST test\_case3

**测试内容**：随机化配置UART寄存器不同的工作模式，包括波特率、奇偶校验、停止位、触发深度等多种情况，之后向TX FIFO 写入发送的数据

**测试结果**：UART模块TX端信号线发送的字符格式与配置相同，测试通过

#### 结果

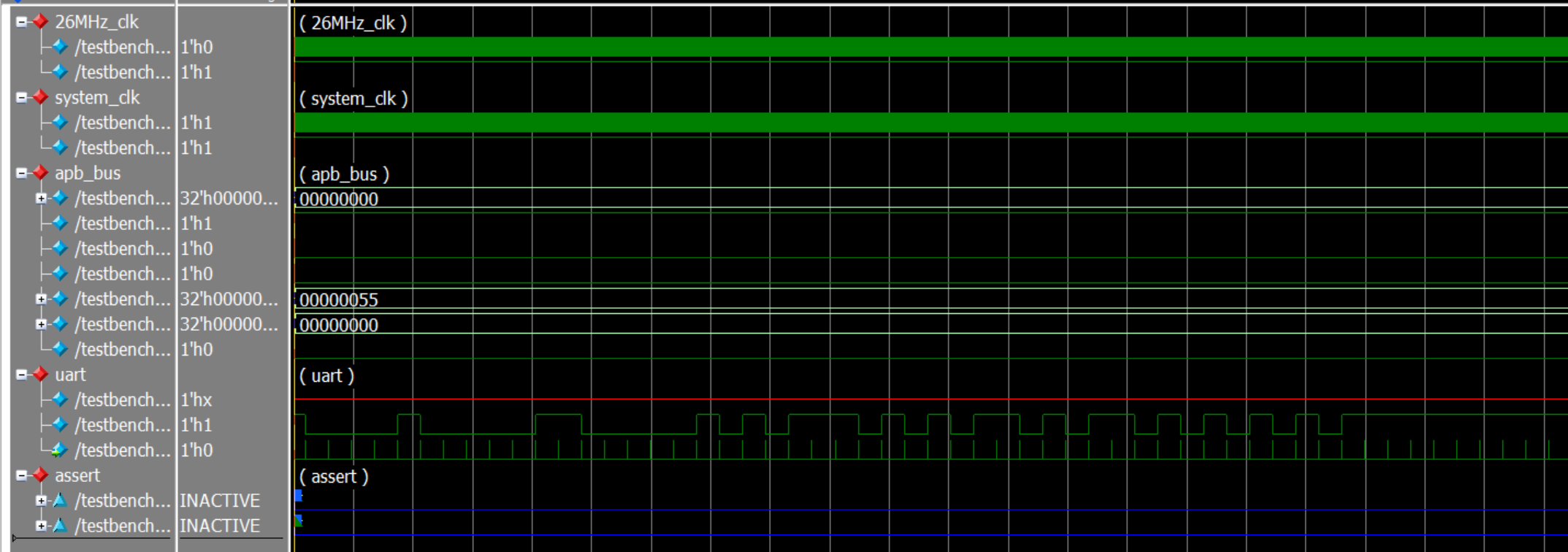


图 25 test\_case3 结果

### TEST\_CASE\_4

UART 中断测试

**测试启动**：测试脚本设置set TEST test\_case4

**测试内容**：配置TX FIFO触发深度，向TX FIFO中连续写入16个数据，等待UART发送一定数量数据后触发中断

**测试结果**：中断信号拉高，此时读取状态寄存器，判断中断情况，之后清除状态寄存器，中断信号拉低，测试通过

#### 结果

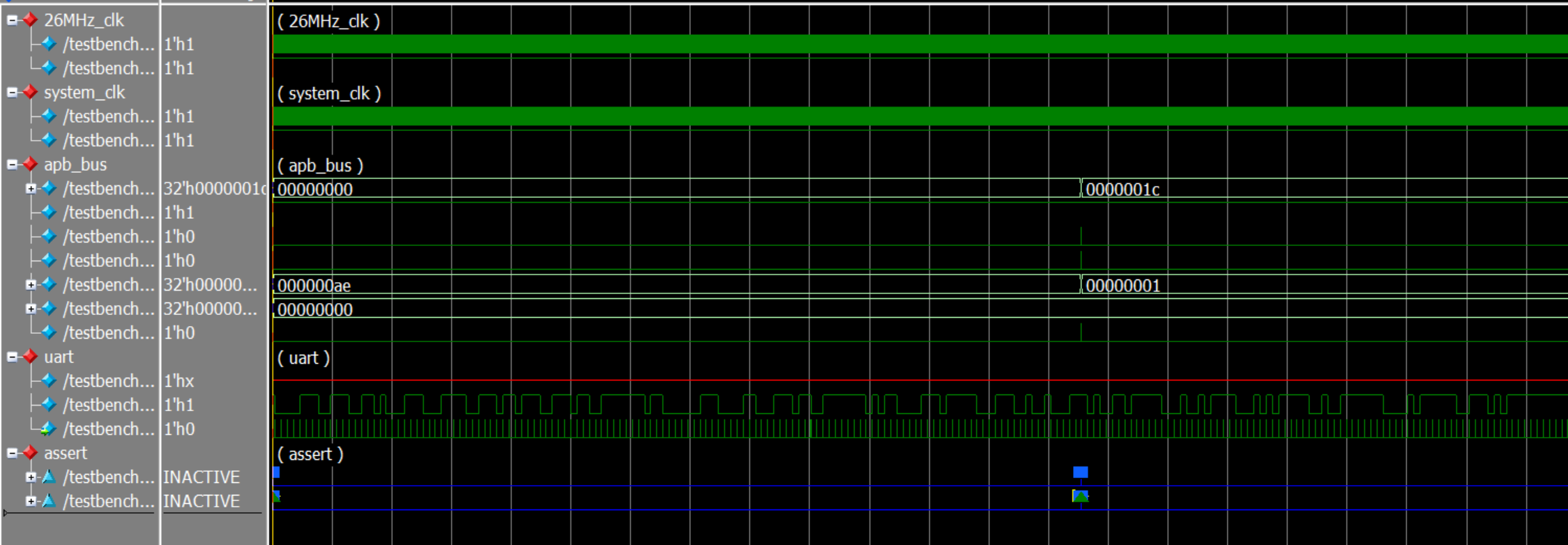


图 26 test\_case4 结果

### TEST\_CASE\_5

覆盖率测试



图 27 断言

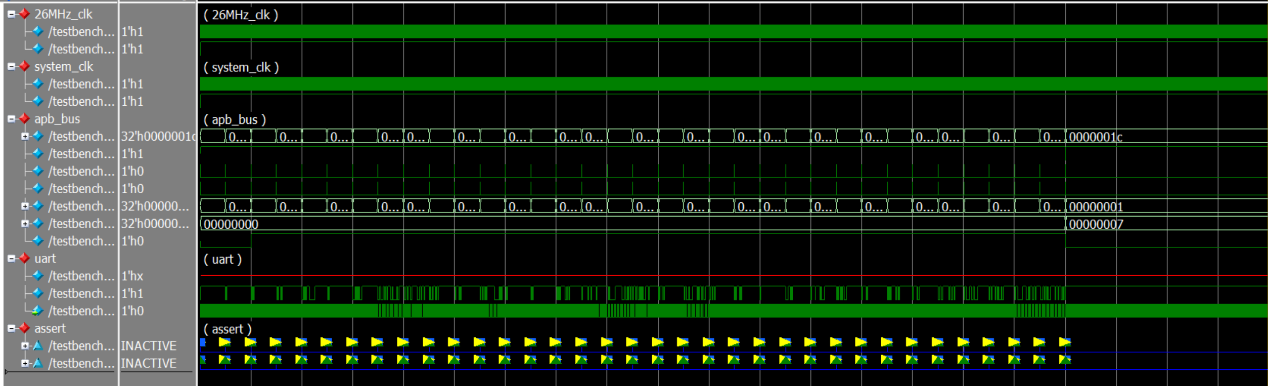


图 28 覆盖率测试波形