1. Monitor
2. Interfaces

Monitor thu thập dữ liệu thông qua interface giao tiếp giữa Driver & DUT. Trong môi trường này, monitor kết nối với 2 interface:

* APB interface có tên instance là *vifApbMaster*để giám sát tất cả các transaction đọc/ghi.
* Interrupt interface có tên instance là *vifInterrupt* để giám sát tất cả các tín hiệu interrupt.

2 interfaces này được khai báo là pointer và được kiểm tra trong build\_phase.

//Declare the monitored interfaces

ifApbMaster \*vifApbMaster;

ifInterrupt \*vifInterrupt;

void build\_phase(uvm::uvm\_phase& phase)

{

uvm::uvm\_monitor::build\_phase(phase);

//Check the APB connection

if (!uvm::uvm\_config\_db<ifApbMaster\*>::get(this, "\*", "vifApbMaster", vifApbMaster))

UVM\_ERROR("cApbMasterDriver", "Can NOT get vifApbMaster!!!");

//Check the interrupt connection

if (!uvm::uvm\_config\_db<ifInterrupt\*>::get(this, "\*", "vifInterrupt", vifInterrupt))

UVM\_ERROR("cVSequencer", "Can NOT get vifInterrupt!!!");

}

1. Ports

Dữ liệu thu thập được sẽ được gửi đến Scoreboard thông qua analysis port của Monitor. Trong môi trường này, Monitor có 2 analysis port:

* Analysis port có tên instance là *ap\_toScoreboard*để gửi các transaction có kiểu dữ liệu là *cApbTransaction*trên APB interface đến Scoreboard.
* Analysis port khác tên *preset\_toScoreboard* được dùng để gửi thông tin reset cho Scoreboard. Kiểu dữ liệu gửi trên port này là **bool**.

2 ports này được khai báo như sau:

//Declare analysis ports

uvm::uvm\_analysis\_port<bool> preset\_toScoreboard;

uvm::uvm\_analysis\_port<cApbTransaction> ap\_toScoreboard;

Note: 2 interfaces và 2 ports của Monitor được miêu tả trong contructor của class.

//Constructor

cApbMasterMonitor(uvm::uvm\_component\_name name)

: uvm\_monitor(name),

preset\_toScoreboard("preset\_toScoreboard"),

ap\_toScoreboard("ap\_toScoreboard"),

vifApbMaster(0),

vifInterrupt(0),

{}

Note: khác biệt với UVM system verilog

* UVM system verilog, analysis port của Monitor kết nối trực tiếp với Scoreboard qua analysis implement ports.
* UVM system C, Monitor kết nối gián tiếp với Scoreboard thông Subriber. Subriber sẽ gọi function của Scoreboard khi có thay đổi từ Monitor.

1. Variables

So với UVM system C, transaction object và analysis port không cần phải được tạo trong build phase. Hiện tại nếu khai báo sẽ có error khi compile (có khả năng là hạn chế của UVM systemC).

Các biến nội khác của Monitor không cần phải được khai báo toàn cục, mà được khai báo cục bộ trong run\_phase. Lý do là các biến nội này chỉ xuất hiện trong run\_phase mà không xuất hiện trong các hàm khác, do đó người thiết kế có thể chọn khai báo các biến này là biến cục bộ (local) của 1 hàm duy nhất để ngăn chặn truy xuất từ bên ngoài class, thay vì là biến toàn cục (global) để chia sẻ giữa các hàm với nhau.

Dưới đây là các biến được sử dụng trong Monitor:

* Transaction: gói transaction được gửi tới Scoreboard thông qua Analysis port.
* String format: phương pháp của C++ dùng để tạo format message cho Error, Warning.
* APB interface: các biến để lưu giá trị từ APB interface.
* Interrupt interface: các biến để lưu giá trị từ Interrupt interface.

// transaction

cApbTransaction coApbTransaction;

// string format

char char\_string[];

std::string warning;

//Internal variables for vifApbMaster

bool preset\_n;

bool psel;

bool penable;

bool pready;

bool pwrite;

sc\_uint<32> paddr;

sc\_uint<32> pwdata;

sc\_uint<32> prdata;

sc\_uint<4> pstrb;

//Detect interrupt toggle

//Internal variables for vifInterrupt

sc\_uint<5> ifEn;

#ifdef INTERRUPT\_COM

sc\_uint<1> ifSta;

bool ctrl\_if;

#else

sc\_uint<5> ifSta;

bool ctrl\_tif;

bool ctrl\_rif;

bool ctrl\_pif;

bool ctrl\_oif;

bool ctrl\_fif;

#endif

1. Cấu trúc và chức năng
2. build\_phase(): Kiếm tra kết nối của Monitor trong môi trường. Cụ thể, hai interface *vifApbMaster*và *vifInterrupt*được kiểm tra.
3. run\_phase():

- Khai báo các biến nội (variables) và transaction object.

- Vòng lặp vô hạn thực hiện các hàm sau khi có cạnh lên xung clock:

1. **read\_interface()**giám sát interface, lưu giá trị cần thiết trên interface vào các biến nội của Monitor.

2. **collect\_data()**phát hiện read/write transfer và lưu lại trong đối tượng coApbTransaction. Sau đó, method write() được sử dụng để gửi gói coApbTransaction đến TLM analysis port ap\_toScoreboard. Phía Scoreboard sẽ nhận và xử lý.

3. **detect\_reset()** gửi giá trị *preset\_n* qua analysis port *preset\_toScoreboard*bằng method *write()*.

4. **monitor\_ifEn()**phát hiện transaction ghi vào thanh ghi interrtupt enable của DUT (UART-TX và UART-RX), địa chỉ offset *16'h0010*.

Giá trị ghi vào các bit interrupt enable sẽ được lưu lại trong biến *ifEn[4:0]*. Biến này sẽ được dùng để điều khiển task *detect\_intf()* với mục đích xác định xem một tín hiệu interrupt tích cực, chuyển từ mức 0 sang mức 1, có đúng hay không.

5.**detect\_intf()**

Nếu một interrupt tích cực nhưng không được enable, UVM\_ERROR sẽ thông báo.

Nếu một interrupt tích cực và được enable, một UVM\_WARNING sẽ thông báo cho người test biết để kiểm tra lại xem có đúng như mong muốn hay không.

Một biến *ifSta*được sử dụng để đảm bảo các thông điệp UVM\_ERROR và UVM\_WARNING chỉ in ra một lần khi interrupt bắt đầu tích cực. Nếu không có biến này, các thông điệp sẽ được in liên tục trong suốt quá trình tín hiệu interrupt tích cực sau mỗi cạnh lên xung clock.

build\_phase

Kiểm tra kết nối với interface

run\_phase

Khai báo transaction & variable

Wait posedge clock

read\_interface()

collect\_data()

detect\_reset()

monitor\_ifEn()

detect\_intf()

APB interface (vifApbMaster, vifInterrupt)

Internal variables

ap\_toScoreboard

preset\_toScoreboard

sc\_uint<5> ifEn

Information message (ERROR, WARNING)

1. Scoreboard
2. Ports

Scoreboard nhận dữ liệu gửi từ Monitor qua analysis export.

Trong môi trường này, Scoreboard có 3 analysis export:

* frmMonitorTX\_listener\_imp: kết nối với analysis port *ap\_toScoreboard* của Monitor trong agent *coApbMasterAgentTx*
* frmMonitorRX\_listener\_imp: kết nối với analysis port *ap\_toScoreboard* của Monitor trong agent *coApbMasterAgentRx*
* resetfrmTX\_listener\_imp: kết nối với analysis port *preset\_toScoreboard* của Monitor trong agent *coApbMasterAgentTx.* Do reset của hệ thống là chung nên chỉ cần kết nối một port để giám sát trạng thái reset, không cần kết nối port đến agent *coApbMasterAgentRx.*

Note: Khác với UVM system verilog, UVM system C không khai báo suffix tương ứng với các task "analysis implementation" (dành cho việc xử lý transaction nhận trên analysis implementation port), mà UVM system C có các Subcriber đóng vai trò cầu nối trung gian giữa Monitor và Scoreboard, Subcriber nhận data từ analysis export của Scoreboard, và sẽ kích hoạt các hàm tương ứng của Scoreboard để xử lý transaction.

Vì vậy Scoreboard có các Subcriber sau:

* resetfrmTX\_subscriber\* resetfrmTX\_listener;
* frmMonitorTX\_subscriber\* frmMonitorTX\_listener;
* frmMonitorRX\_subscriber\* frmMonitorRX\_listener;

1. Variables

Scoreboard có các hàm xử lý transaction khác nhau, nên việc khai báo các biến toàn cục để chia sẻ dữ liệu là cần thiết. Các biến được sử dụng trong Scoreboard:

* Fifo Rx,Tx để lưu dữ liệu được truyền và biến truy xuất Fifo để so sánh.
* UART enable status.
* Reset flag.
* String format: phương pháp của C++ dùng để tạo format message cho Error, Warning.

// Data queue for storing the transmit data

// of UARTs

std::queue<int> queueTransTX;

std::queue<int> queueTransRX;

// Compared data

int queueCompTX; //from queueCompTX

int queueCompRX; //from queueCompRX

// UART-TX and UART-RX enable status

sc\_uint<1> uartEnTX;

sc\_uint<1> uartEnRX;

//Reset flag

sc\_uint<1> rst\_flg;

// string format

char char\_string[];

std::string info;

std::string warning;

std::string error;

1. Cấu trúc và chức năng
2. build\_phase(): tạo các Subcriber.

void build\_phase(uvm::uvm\_phase& phase)

{

uvm::uvm\_scoreboard::build\_phase(phase);

uvm::uvm\_config\_db<uvm\_object\*>::set(this, "resetfrmTX\_listener", "coScoreboard", this);

uvm::uvm\_config\_db<uvm\_object\*>::set(this, "frmMonitorTX\_listener", "coScoreboard", this);

uvm::uvm\_config\_db<uvm\_object\*>::set(this, "frmMonitorRX\_listener", "coScoreboard", this);

resetfrmTX\_listener = xmt\_subscriber::type\_id::create("resetfrmTX\_listener", this);

assert(resetfrmTX\_listener);

frmMonitorTX\_listener = rcv\_subscriber::type\_id::create("frmMonitorTX\_listener", this);

assert(frmMonitorTX\_listener);

frmMonitorRX\_listener = rcv\_subscriber::type\_id::create("frmMonitorRX\_listener", this);

assert(frmMonitorRX\_listener);

}

1. connect\_phase(): kết nối analasys export của Scoreboard đến các Subcriber.

void connect\_phase(uvm::uvm\_phase& phase)

{

resetfrmTX\_listener\_imp.connect(resetfrmTX\_listener->analysis\_export);

frmMonitorTX\_listener\_imp.connect(frmMonitorTX\_listener->analysis\_export);

frmMonitorRX\_listener\_imp.connect(frmMonitorRX\_listener->analysis\_export);

}

1. run\_phase(): thực thi các task “analysis implementation” – Subcriber sẽ gọi các task này khi nó nhận được transaction trên analasys export.

* *write\_resetfrmTX()*giám sát trạng thái reset được Monitor gửi đến là tích cực cờ trạng thái *rst\_flg=1* nếu reset tích cực mức 0.
* *write\_frmMonitorTX()* giám sát việc đọc dữ liệu từ thanh ghi dữ liệu của UART-TX (uart\_0), địa chỉ offset *16'h000C.* Nếu có một tranasaction đọc thanh ghi dữ liệu, dữ liệu đọc trên *prdata[7:0]* sẽ so sánh với dữ liệu truyền phía UART-RX (uart\_1). Nếu giá trị dữ liệu đọc trùng khớp thì thông điệp báo SUCCESS sẽ được in ra, nếu dữ liệu đọc bị sai thì thông điệp báo lỗi FAIL sẽ được in ra.
* *write\_frmMonitorRX()* giám sát việc đọc dữ liệu từ thanh ghi dữ liệu của UART-RX (uart\_1), địa chỉ offset *16'h000C.* Nếu có một tranasaction đọc thanh ghi dữ liệu, dữ liệu đọc trên *prdata[7:0]* sẽ so sánh với dữ liệu truyền phía UART-TX (uart\_0). Nếu giá trị dữ liệu đọc trùng khớp thì thông điệp báo SUCCESS sẽ được in ra, nếu dữ liệu đọc bị sai thì thông điệp báo lỗi FAIL sẽ được in ra.

1. report\_phase() sẽ kiểm tra lại số lượng dữ liệu mong muốn truyền trên UART-TX và UART-RX. Nếu vẫn còn dữ liệu cần truyền ở UART-TX nhưng chưa được đọc và kiểm tra trên UART-RX hoặc ngược lại thì Scoreboard sẽ cảnh báo với UVM\_WARNING. Chú ý, đây không phải là một lỗi (error) vì nó phụ thuộc vào mục đích test của người viết testbench (testcase).