**X2P IP core**

**VLSI Technology**

**Môi trường mô phỏng UVM**

**kiểm tra X2P IP Core**

**2019.11.07**

# REVISION HISTORY

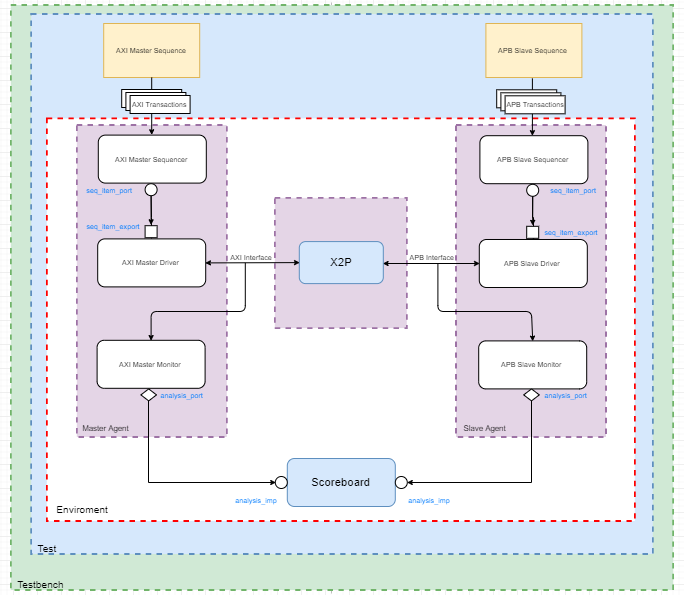
|  |  |  |
| --- | --- | --- |
| **REVISON** | **CONTENT** | **DATE/PIC** |
| **1.0** | New created | …  Quân Nguyễn  Trâm |
| **1.1** | …. | …. |
| **1.2** | Update structure for AXI agent | 7/12/2019  Quang Nguyễn |
| **1.3** | Thêm X2PScoreboard mô tả | 04/01/2019  Quang Nguyễn |

# TỔNG QUAN

Nội dung chương này sẽ trình bày tổng quan về môi trường sử dụng để kiểm tra IP X2P

* Đặc điểm của môi trường
* Kết nối giữa các thành phần

## Đặc điểm của môi trường

**Hình 1.Tổng quan của môi trường**

Các thành phần hỗ trợ:

* **AXI Master Sequence**: Là đối tượng tạo ra các AXI Transaction và cung cấp chung cho AXI Master Sequencer để đưa đến AXI Master Driver.
* **APB Slave Sequence:** Là đối tượng tạo ra các APB Transaction và cung cấp chung cho APB Slave Sequencer để đưa đến APB Slave Driver.
* **AXI Transaction:** Chứa các tín hiệu để lái X2P và dữ liệu để kiểm tra hoạt động của giao thức AXI.

Môi trường kiểm tra IP X2P bao gồm các thành phần:

1. **Master Agent** (Sử dụng giao thức AXI).

* **AXI Master Sequencer**: Là đối tượng điều khiển việc truyền các AXI Transaction do AXI Master Sequencetạo ra để chuyển đến AXI Master Driver.
* **AXI Master Driver:**

+ Kiểm tra kết nối giữa Master vs X2P Core thông qua AXI Interface

+ Nhận các AXI Transaction từ AXI Master Sequence và lần lượt chuyển chúng thành các giá trị tương ứng để lái các tín hiệu trên giao tiếp với X2P

+ **AXI Interface**: Chứa tất cả input, output của X2P có chức năng giao tiếp với Master

* **AXI Master Monitor:** Giám sát và lấy các tín hiệu và dữ liệu giao tiếp giữa AXI Master Driver vs X2P cần để chuyển đến Scoreboard.

1. **Slave Agent** (Sử dụng giao thức APB)

* **APB Slave Sequencer**: Là đối tượng điều khiển việc truyền các APB Transaction do APB Slave Sequencetạo ra để chuyển đến APB Slave Driver.
* **APB Slave Driver:**

+ Kiểm tra kết nối giữa Slave vs X2P Core thông qua APB Interface

+ Nhận các APB Transaction từ APB Slave Sequence và lần lượt chuyển chúng thành các giá trị tương ứng để lái các tín hiệu trên giao tiếp với X2P

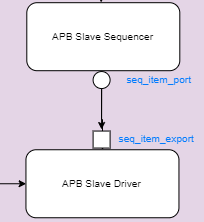
+ APB Interface: chứa tất cả input, output của X2P có chức năng giao tiếp với Slave

* **APB Slave Monitor:** Giám sát và lấy các tín hiệu và dữ liệu giao tiếp giữa APB Slave Driver vs X2P cần để chuyển đến Scoreboard.

## Giao tiếp giữa các thành phần

1. **Master Agent**
2. **Slave Agent**

* **APB Slave Sequencer vs APB Slave Driver:**

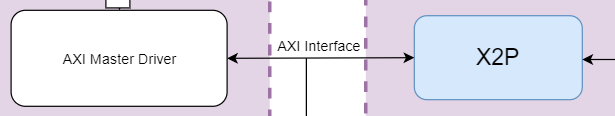


**Hình 2: Giao tiếp giữa APB Slave Sequencer vs APB Slave Driver**

Hai đối tượng này giao tiếp với nhau thông qua bộ port seq\_item\_export và seq\_item\_port, 2 port này được gọi từ thư viện UVM.

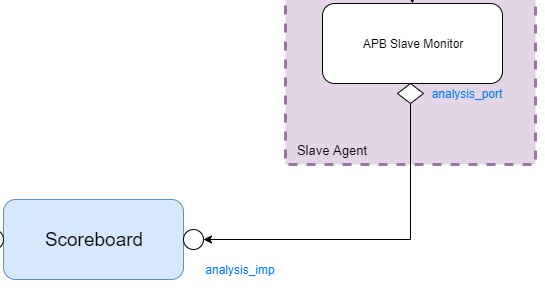
+ Port của APB Slave Sequencer là seq\_item port

+ Port của APB Slave Driver là seq\_item\_export

* **APB Slave Driver vs X2P:**

Hai đối tượng này giao tiếp với nhau thông qua APB Interface.

* **APB Slave Monitor vs Scoreboard:**



Hai đối tượng này giao tiếp với nhau thông qua bộ port analysis\_port và analysis\_imp, 2 port này được gọi từ thư viện UVM.

+ Port của APB Slave Monitor là analysis\_port

+ Port của Scoreboard là analysis\_imp

# PHÂN TÍCH MÔI TRƯỜNG

Nội dung chương này trình bày cấu trúc của các Transaction và các thành phần khác.

* Cấu trúc AXI Interface
* Cấu trúc AXI Transaction
* Cấu trúc APB Interface
* Cấu trúc APB Transaction

## 2.1 AXI AGENT

### 2.1.1 Cấu trúc AXI Interface

*// System*

*logic aclk;*

*logic reset\_n;*

*// Kênh địa chỉ đọc*

*logic arvalid;*

*logic arready;*

*logic [31:0] araddr;*

*logic [2:0] arsize;*

*logic [7:0] arlen;*

*logic [1:0] arburst;*

*logic [7:0] arid;*

*logic [2:0] arprot;*

*// Kênh dữ liệu đọc*

*logic rvalid;*

*logic rready;*

*logic rlast;*

*logic [7:0] rid;*

*logic [1:0] rresp;*

*logic [31:0] rdata;*

*// Kênh địa chỉ ghi*

*logic awvalid;*

*logic awready;*

*logic [31:0] awaddr;*

*logic [2:0] awsize;*

*logic [7:0] awarlen;*

*logic [7:0] awid;*

*logic [1:0] awburst;*

*logic [2:0] awprot;*

*// Kênh dữ liệu ghi*

*logic wready;*

*logic wvalid;*

*logic wlast;*

*logic [7:0] wid;*

*logic [31:0] wdata;*

*logic [3:0] wstrb;*

*// Kênh phản hồi ghi*

*logic bready;*

*logic bvalid;*

*logic [1:0] bresp;*

*logic [7:0] bid;*

### 2.1.2 Cấu trúc AXI Transaction

Đây là một class được mở rộng từ uvm\_sequence\_item. Nó sẽ chứa tất cả các thành phần dữ liệu và thông tin cần thiết để AXI Master Driver có thể sử dụng để lái AXI Interface cho X2P.

#### 2.1.2.1 Transaction

*rand enum trans\_type // dữ liệu chứa thông tin đây là Transaction đọc hay ghi.*

*rand bit [ADDR\_WIDTH-1:0] addr; // dữ liệu chứa thông tin địa chị đọc/ghi*

*rand bit [ID\_WIDTH-1:0] rid; // dữ liệu chứa thông tin ID của Transaction.*

*rand bit [PROT\_WIDTH-1:0] prot // dữ liệu chứa thông tin về quyền truy cập*

*rand bit [BURST\_LENGTH\_WIDTH-1:0] burst\_length // dữ liệu chứa thông tin số burst trong một Transaction.*

*rand enum burst\_size // dữ liệu chứa thông tin số byte trong một burst*

*rand enum burst\_type // dữ liệu chứa thông tin kiểu truyền của Transaction.*

*rand bit [DATA\_WIDTH-1:0] data [] // dữ liệu đọc/ghi*

*rand bit [DATA\_WIDTH/8-1:0] wr\_strobe // dữ liệu chứa thông tin data có nghĩa trong một write Transaction.*

*rand enum response // dữ liệu chứa trạng thái của Transaction đọc/ghi*

*rand bit [31:0] addr\_ready\_delay // khoản thời gian đường địa chỉ sẵn sàng hay không giữa các Transaction*

*rand bit [31:0] addr\_valid\_delay // khoản thời gian đường địa chỉ bắt đầu truyền hay không giữa các Transaction*

*rand bit [31:0] rready\_delay // khoản thời gian đường data đọc sẵn sàng hay không giữa các burst/Transaction*

*rand bit [31:0] rvalid\_delay // khoản thời gian đường data đọc bắt đầu truyền hay không giữa các burst/Transaction*

*rand bit [31:0] wready\_delay // khoản thời gian đường data ghi sẵn sàng hay không giữa các burst/Transaction*

*rand bit [31:0] wvalid\_delay // khoản thời gian đường data ghi bắt đầu truyền hay không giữa các burst/Transaction*

*rand bit [31:0] bready\_delay // khoản thời gian đường phản hồi data ghi sẵn sàng hay không giữa các burst/Transaction*

*rand bit [31:0] bvalid\_delay // khoản thời gian đường phản hồi data ghi bắt đầu truyền hay không giữa các burst/Transaction*

*rand bit data\_before\_addr // tín hiệu cho phép kiểu đường data ghi trước đường địa chỉ ghi*

*ranb bit suspend\_response // tín hiệu báo rằng không phản hồi cho trong ghi Transaction.*

#### 2.1.2.2 Cấu trúc

- *parameter*:

+ ADDR\_WIDTH = 32: X2P hỗ trợ 32bit đường địa chỉ

+ ID\_WIDTH = 7: X2P hỗ trợ 8bit đường ID.

+ BURST\_LENGTH\_WIDTH = 7: X2P hỗ trợ 8bit đường A[R,W]RLEN.

+ DATA\_WIDTH = 32: X2P hỗ trợ 32bit đường dữ liệu.

+ PROT\_WIDTH = 3: X2P hỗ trợ 3bit đường cho phép truy cập.

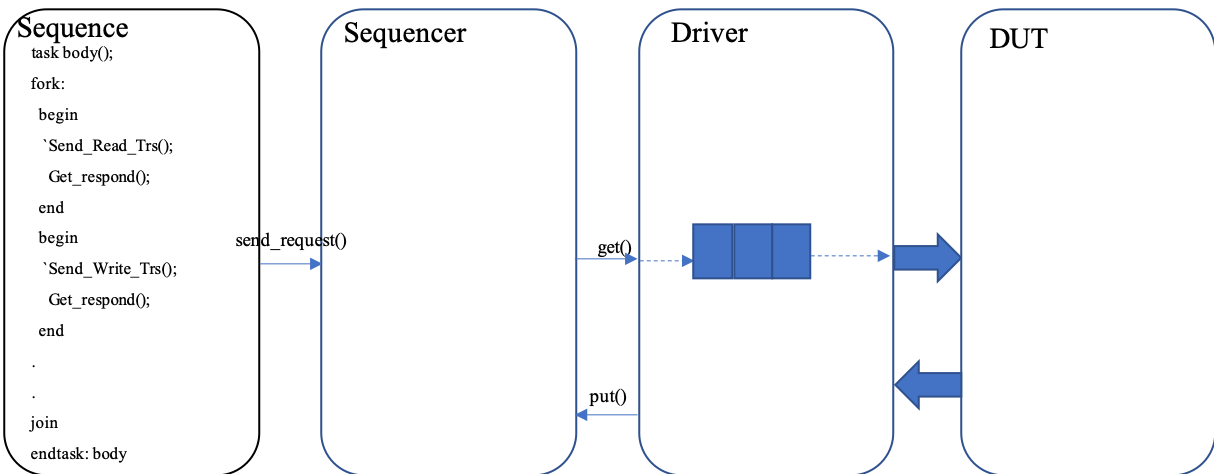
- *burst\_size*: vì X2P module chỉ hỗ trợ truyền 4byte nên cố định giá trị 2 cho biến này.

- *enable\_interleave*: cố định giá trị 0, chưa support trong môi trường hiện tai.

- *enbale\_reorder*: cố định giá trị 0, chưa support trong môi trường hiện tai.

- *data*: độ dài của mảng *data* sẽ phụ thuộc vào biến *burst\_length*, độ dài của mảng bằng với *burst\_length*+1

### 2.1.3 Giao tiếp giữa AXI Master Sequence, AXI Master Sequencer, AXI Master Driver và DUT



Sequence sẽ gửi những transaction vào Sequencer thông qua cú pháp fork join. Tất cả những transaction đó sẽ được lưu vào FIFO bên trong Sequencer(FIFO hệ thống, user không thể thấy được)

Driver sẽ lấy tất cả những transaction có trong FIFO Sequencer thông qua method *get()* và lưu nó vào FIFO bên trong Driver. FIFO bên trong Driver do người xây dựng môi trường khai báo.

BFM sẽ nhận input từ Driver FIFO và bắt đầu lái transaction đến DUT.

Sau khi nhận được tín hiệu respond từ DUT, BFM sẽ xử lý và báo cho Sequence rằng transaction đã hoàn tất thông qua method *put().*

### 2.1.4 Chi tiết AXI Agent

### 2.1.5 AXI Interface Checker

Kiểm tra BUS tín hiệu không thay đổi khi READY chưa bằng 1’b1.

Kiểm tra BUS tín hiệu không bằng ‘hX hoặc ‘hZ khi VALID bằng 1’b1.

Kiểm tra BUS tín hiệu phải bằng ‘h0 sau RESET.

## 2.2 APB AGENT

### 2.2.1 Cấu trúc APB Interface

*interface ifApbSlave;*

*//input DUT*

*logic pready;*

*logic [31:0] prdata;*

*logic pslverr;*

*//output DUT*

*logic [31:0] paddr;*

*logic [31:0] pwdata;*

*logic [SLAVE\_NUM-1:0]psel;*

*logic [2:0] pprot;*

*logic [3:0] pstrb;*

*logic pwrite;*

*endinterface: ifApbSlave*

### 2.2.2 Cấu trúc APB Transaction

Đây là một class được mở rộng từ uvm\_sequence\_item. Nó sẽ chứa tất cả các thành phần dữ liệu và thông tin cần thiết để APB Slave Driver có thể sử dụng để lái APB Interface của X2P.

Cấu trúc như sau:

*class cApbTransaction extends uvm\_sequence\_item;*

*rand bit pready;*

*rand bit [31: 0] prdata;*

*rand bit pslverr;*

*rand bit [7:0] preadyDelay;*

*//*

*logic [31:0] paddr;*

*logic [31: 0] pwdata;*

*logic [SLAVE\_NUM-1:0] psel;*

*logic [2:0] pprot;*

*logic [3:0] pstrb;*

*logic pwrite;*

*`uvm\_object\_utils\_begin (cApbTransaction)*

*`uvm\_field\_int(pready, UVM\_ALL\_ON)*

*`uvm\_field\_int(prdata, UVM\_ALL\_ON)*

*`uvm\_field\_int(pslverr, UVM\_ALL\_ON)*

*`uvm\_field\_int(preadyDelay, UVM\_ALL\_ON)*

*`uvm\_field\_int(paddr, UVM\_ALL\_ON)*

*`uvm\_field\_int(pwdata, UVM\_ALL\_ON)*

*`uvm\_field\_int(psel, UVM\_ALL\_ON)*

*`uvm\_field\_int(pprot, UVM\_ALL\_ON)*

*`uvm\_field\_int(pstrb, UVM\_ALL\_ON)*

*`uvm\_object\_utils\_end*

*//Constructor*

*function new (string name = "cApbTransaction");*

*super.new(name);*

*endfunction: new*

*endclass: cApbTransaction*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên biến | Độ rộng | Mô tả | Sử dụng |
| *1* | *pready* | *1* | *Khi tín hiệu này tích cực thì slave sẽ thông báo cho master biết slave đã sẵn sàng nhận dữ liệu.* | *Điều khiển tín hiệu pready ở Interface* |
| *2* | *prdata* | *32* | *Dữ liệu đọc được gửi từ slave APB* | *Dữ liệu truyền đến prdata ở Interface* |
| *3* | *pslverr* | *1* | *Tín hiệu này có chức năng thông báo cho APB master biết trạng thái của transfer có lỗi hay không lỗi.* | *Điều khiển tín hiệu pslverr trên Interface* |
| *4* | *preadyDelay* | *8* | *Tạo delay cho pready* | *Để Driver tạo delay sau mỗi lần thực hiện truyền nhận dữ liệu qua Interface* |
| *5* | *paddr* | *32* | *Tín hiệu địa chỉ*  *Tín hiệu này chứa thông tin địa chỉ đọc hoặc ghi* | *Dữ liệu truyền đến paddr ở Interface* |
| *6* | *pwdata* | *32* | *Tín hiệu dữ liệu*  *Tín hiệu này chứa dữ liệu cần truyền.* | *Dữ liệu truyền đến pwdata ở Interface* |
| *7* | *psel* | *SLAVE\_NUM-1:0* | *Mỗi tín hiệu psel sẽ chỉ tới 1 slave APB.*  *Tín hiệu này sẽ cho phép slave APB đó hoạt động hay không hoạt động.* | *Điều khiển tín hiệu psel ở Interface* |
| *8* | *pprot* | *3* | *Tín hiệu này thể hiện quyền truy cập của transfer.* | *Điều khiển tín hiệu pprot ở Interface* |
| *9* | *pstrb* | *4* | *Tín hiệu này có chức năng điều khiển các byte trong 1 transfer.* | *Điều khiển tín hiệu pstrb ở Interface* |
| *10* | *pwrite* | *1* | *Tín hiệu này có chức năng phân biệt truy nhập.*  *pwrite = 1 là truy nhập ghi*  *pwrite = 0 là truy nhập đọc* | *Điều khiển tín hiệu pwrite ở Interface* |

## 2.3 X2P SCOREBOARD

### 2.3.1 Block diagram

A screenshot of a cell phone

Description automatically generatedAxiRdAddr: Analysis port từ AXI Agent báo hiệu kênh Read Address từ AXI I/F.

AxiRdData: Analysis port từ AXI Agent báo hiệu kênh Read Data từ AXI I/F.

AxiWrAddr: Analysis port từ AXI Agent báo hiệu kênh Write Address từ AXI I/F.

AxiWrData: Analysis port từ AXI Agent báo hiệu kênh Write Data từ AXI I/F.

AxiWrResp: Analysis port từ AXI Agent báo hiệu kênh Write Respond từ AXI I/F.

ApbReadTransfer: Analysis port từ APB Agent báo hiệu APB đang ở trạng thái SETUP của Read Trans.

ApbReadRespond: Analysis port từ APB Agent báo hiệu trạng thái ACCESS của READ Trans kết thúc.

ApbWriteTransfer: Analysis port từ APB Agent báo hiệu APB đang ở trạng thái SETUP của Write Trans.

ApbWriteRespond: Analysis port từ APB Agent báo hiệu trạng thái ACCESS của Write Trans kết thúc.

### 2.3.2 Chức năng

X2P Model sẽ được thiết kế đặt phía trong Scoreboard để mô phỏng hoạt động, thanh ghi của X2P.

Scoreboard sẽ giám sát AXI giao thức và APB giao thức thông qua analysis port. Sau đó sẽ so sánh 2 giao thức với model đã được đề cập ở trên.

### 2.3.3 Checking point

Đối với transaction bình thường, Scoreboard sẽ kiểm tra input và output của X2P.

Đối với transaction lỗi từ AXI master, Scoreboard có thể đoán được hoạt động của X2P để:

+ có thể kiểm tra được AXI kênh RESPOND.

+ có thể kiểm tra được không có APB Trans tới SLAVE.

Scoreboard dựa vào địa chỉ của transaction để có thể biết được APB Slave nào sẽ nhận được request từ X2P.

Kiểm tra thanh ghi của X2P.