**VLSI Technology**

**X2X Synchronous Bridge**

**Lê Quang Hưng**

**Nguyễn Hùng Quân**

**2020.09.22**

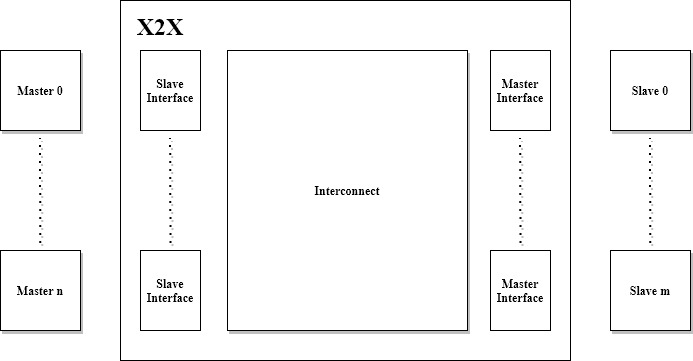
1 ĐẶC ĐIỂM THIẾT KẾ

* 1. Giới thiệu
     1. Đặc điểm hỗ trợ

Cầu AXI được mô tả trong tài liệu này hỗ trợ các đặc điểm sau:

* Cấu hình số lượng Master và Slave
* Cấu hình vùng địa chỉ cho từng Slave
* Cấu hình độ rộng bus dữ liệu
* Hỗ trợ 3 loại burst: FIXED, INCR và WRAP
* Hỗ trợ các loại phân xử:
  + Absolute Fixed Priority Arbiter.
  + Dynamic Priority Arbiter
  + Round Robin with Priority logic
  + Balance Round Robin
  + Least Recently Used
* Hỗ trợ Outstanding cho cả read và write transaction.
* Cấu hình được độ sâu FIFO trên các kênh ở cả phía master và slave.
* Hỗ trợ Out-of-order completion cho cả read và write transaction
* Cấu hình quyền truy cập của mỗi master đến các slave: master chỉ có thể truy cập đến các slave được cho phép.
  + - 1. Sơ đồ khối chức năng X2X Synchronous Bridge

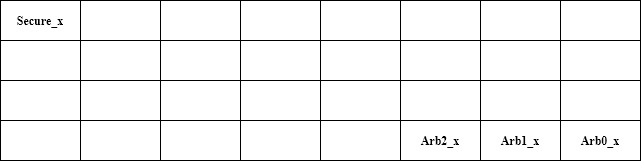
Số lượng master và slave có thể khác nhau. Độ rộng bus dữ liệu phải thống nhất giữa Master, Slave và X2X. Slave interface sẽ được tạo ra tương ứng với số lượng master được hỗ trợ, tương tự cho số Master interface và số Slave. Slave 0 và Master interface 0 là hai khối chức năng mặc định của X2X, điều khiển và chứa các thanh ghi cấu hình hoạt động của X2X. Do đó, khi người dùng cấu hình số Slave là x, mặc định số Master interface được sinh ra sẽ là x + 1.



**Hình 1: X2X Synchronous Bridge functional block diagram**

* 1. Tóm tắt tín hiệu giao tiếp X2X
  2. Các thanh ghi cấu hình X2X
     1. X2X Master interface configure register x (X2XMICRx)

Số lượng thanh ghi X2XMICR được tạo ra bằng với số lượng Slave được hỗ trợ.



**Hình 2: X2XMICRx**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bit | Name | Default | Attribute | Description |
| 31:3 | Reversed | 29’h0 | RO | Các bit không sử dụng |
| 2:0 | Arb2\_x – Arb0\_x | 3’b0 | RW | *Arbiter scheme*  Chọn cấu hình arbiter được sử dụng cho mỗi Slave interface:  3’b000: Absolute Fixed Priority Arbiter  3’b001: Dynamic Priority Arbiter  3’b010: Round Robin with Priority logic  3’b011: Balance Round Robin  3’b100: Least Recently Used |

**Bảng 2: Bảng mô tả thanh ghi X2XMICRx**

* 1. Các parameter cấu hình X2X

Các parameter đánh dấu “\*”, là các parameter cấu hình chỉ nhận 2 giá trị là 0 hoặc 1, với mức phép là mức 1.

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Default** | **Description** |
| MASTER\_NUM | 1 | Số lượng Master kết nối vào X2X |
| SLAVE\_NUM | 1 | Số lượng Slave kết nối vào X2X |
| RAW\_ID\_WIDTH | 4 | Độ rộng lớn nhất của ID mà một Master có thể tạo ra |
| SLV\_BASE\_ADDR\_x | 0x0000\_0000 | Base address cho vùng địa chỉ của mỗi Slave |
| SLV\_RANGE\_x | 0 | Số lượng Byte trong vùng địa chỉ của mỗi Slave |
| CDAS\_SEL | SP | S: Single Slave CDAS  SP: Single Slave per ID CDAS |
| ID\_NUM\_MASTER\_x | 1 | Số lượng ID mà mỗi Master có thể tạo ra |
| QUEUE\_WIDTH\_MASTER\_x | 1 | Số lượng outstanding write transaction mà một Master có thể gửi đi |
| MI\_WRITE\_ACCEPTANCE | 1 | Số lượng outstanding write transaction mà một Slave có thể nhận được. |
| MASTER\_x\_SLAVE\_y\_ENA\* | 0 | Cho phép kết nối Master \_x vơi Slave \_y |
| SI\_AR\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh AR ở phía SI |
| SI\_AR\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh AR ở phía SI |
| MI\_AR\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh AR ở phía MI |
| MI\_AR\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh AR ở phía MI |
| SI\_AW\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh AW ở phía SI |
| SI\_AW\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh AW ở phía SI |
| MI\_AW\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh AW ở phía MI |
| MI\_AW\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh AW ở phía MI |
| SI\_W\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh W ở phía SI |
| SI\_W\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh W ở phía SI |
| MI\_W\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh W ở phía MI |
| MI\_W\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh W ở phía MI |
| SI\_R\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh R ở phía SI |
| SI\_R\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh R ở phía SI |
| MI\_R\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh R ở phía MI |
| MI\_R\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh R ở phía MI |
| SI\_B\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh B ở phía SI |
| SI\_B\_FIFO\_DEPTH | 0 | Cấu hình độ sâu FIFO cho kênh B ở phía SI |
| MI\_B\_FIFO\_ENA\* | 1 | Cấu hình một FIFO cho kênh B ở phía MI |
| MI\_B\_FIFO\_DEPTH | 1 | Cấu hình độ sâu FIFO cho kênh B ở phía MI |
| SI\_REG\_SLICE\_x\* | 1 | Chèn thêm các Flip Flop trên các đường critical path bên phía SI |
| MI\_REG\_SLICE\_x\* | 1 | Chèn thêm các Flip Flop trên các đường critical path bên phía MI |

**Hình 3: Bảng parameter cấu hình của X2X**

* 1. Hoạt động X2X
     1. Outstanding transaction

Master có thể gửi địa chỉ liên tục ngay cả khi transaction trước nó vẫn chưa hoàn thành. Để đơn giản thiết kế và đảm bảo hoạt động chính xác của hệ thống bus, X2X không cho phép các outstanding transaction hiện hành có cùng ID đi đến các Slave khác nhau. Nếu X2X phát hiện ra một outstanding transaction đang đi đến một Slave khác với Slave đang hiện hành, X2X sẽ yêu cầu dừng outstanding transaction đó đến khi các outstanding transaction trước nó được hoàn thành.

* + 1. Out-of-order transaction

Thiết kế AXI này hỗ trỡ out-of-order completion của cả read transaction và write transaction. Khi Master gửi nhiều transaction đến các Slave khác nhau, các Slave hoạt động với tốc độ nhanh hơn có thể hoàn thành transaction trước các Slave chậm hơn. Với hỗ trợ Out-of-order, thứ tự hoàn thành các transaction có thể khác với thứ tự mà Master issue địa chỉ khởi động transaction. Các transaction thỏa mãn các điều kiện ordering có thể được hoàn thành trước các transaction được issue trước nó, nhờ đó tốc độ của hệ thống có thể được cải thiện.

* + 1. Cyclic Dependency Avoidance Schemes (CDAS)

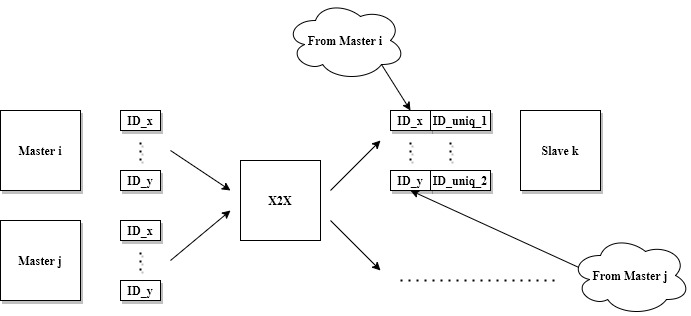
X2X cho phép các transaction được hoàn thành với thứ tự khác với thứ tự khởi tạo, và Slave có thể trả về các re-ordering transaction. Nếu như không có một cơ chế phù hợp điều khiển quá trình này, hiện tượng deadlock có thể xảy ra. Do đó, khi một Master gửi các transaction có cùng ID đến các Slave khác nhau, X2X phải hiện thực một số quy tắc để đảm bảo hoạt động chính xác của hệ thống bus. Để đơn giản, thiết kế X2X này hỗ trợ 2 scheme của ARM là:

* Single Slave:
  + Tất cả các outstanding read transaction hiện hành phải đi đến cùng một Slave.
  + Tất cả các outstanding write transaction hiện hành phải đi đến cùng một Master.
    - * Nếu Slave interface phát hiện một transaction được gửi đến một Slave khác với Slave hiện hành của transaction cùng loại (các transaction của cùng một kênh), nó sẽ yêu cầu dừng transaction này cho đến khi các outstanding transaction trước đó của nó được hoàn thành.
* Single Slave per ID
  + Tất cả các outstanding read transaction hiện hành có cùng ID phải đi đến cùng một Slave
  + Tất cả các outstanding write transaction hiện hành có cùng ID phải đi đến cùng một Slave.
    - * Slave interface cho phép các outstanding transaction khác ID có thể đi đến các Slave khác nhau, tuy nhiên các outstanding transaction cùng ID (và cùng đến từ một kênh) phải đi đến cùng một Slave. Các outstanding transaction có cùng ID gửi đến Slave khác Slave hiện hành sẽ bị dừng cho đến khi các outstanding transaction trước nó được hoàn thành.
    1. Ordering

Các yêu cầu về ordering cần được thõa mãn:

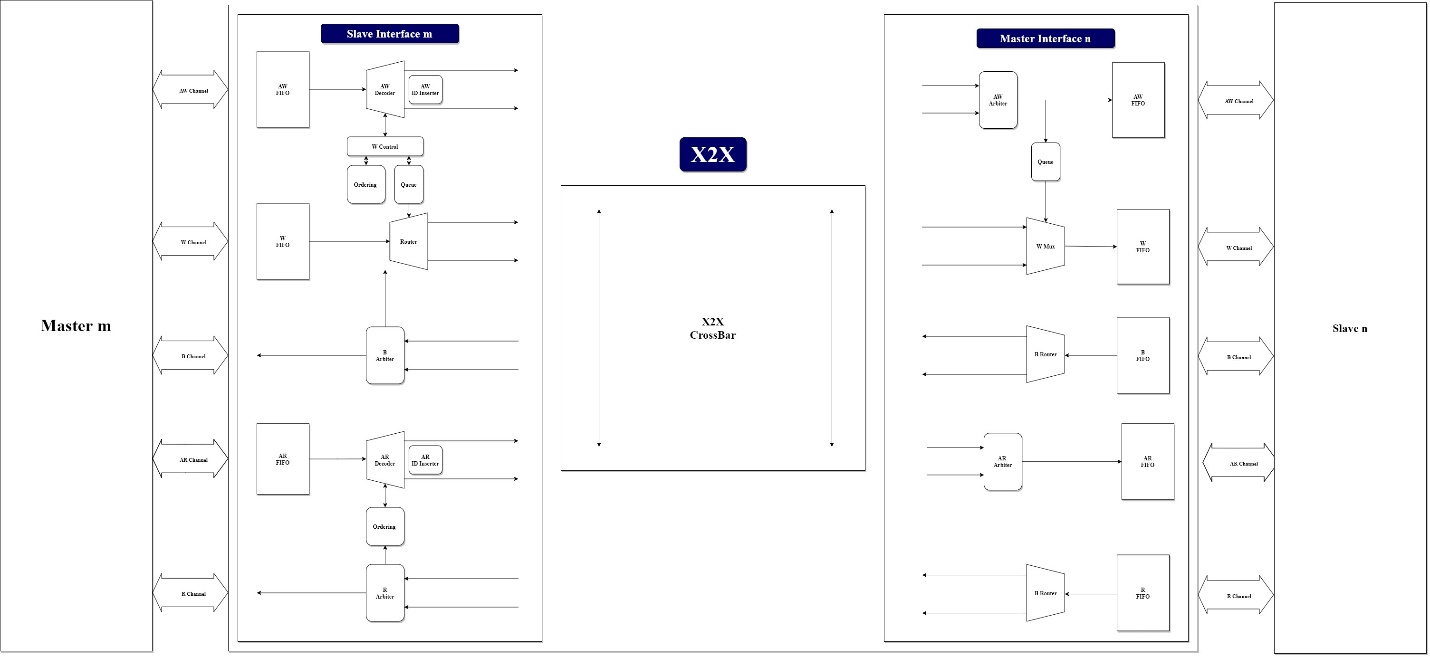
* Các Read transaction với cùng ID phải được hoàn thành theo đúng thứ tự mà Master issue địa chỉ.
* Các Write transaction với cùng ID phải được hoàn thành theo đúng thứ tự mà Master issue địa chỉ.
* Các Transaction của kênh W phải được đưa đến Slave đích theo đúng thứ tự mà Master issue địa chỉ. (do AXI4 không hỗ trợ WID)
  + 1. ID transaction

Các ID được hỗ trợ ở mỗi Master có thể trùng nhau. X2X sẽ thêm các trường phân biệt sau mỗi ID trước khi đưa đến Slave đích. Điều này giúp ID nhận được ở phía Slave là độc nhất cho mỗi Master nguồn.



**Hình 4: ID identify**

1. CẤU TRÚC X2X SYNCHRONOUS BRIDGE
   1. Tổng quát



**Hình 5: Hình minh họa một cấu hình X2X**

Bên trong lõi X2X, khối X2X Crossbar làm nhiệm vụ kết nối các SI (Slave Interface) và MI (Master Interface). Tùy vào các parameter được cài đặt, các bộ FIFO, Register Slice, Arbiter, Decoder, … sẽ được tạo ra và kết nối và Crossbar.

* 1. Decoder (Address Decoder)

Dựa vào tín hiệu địa chỉ trên các kênh AW và AR, các bộ **Decoder** sẽ so sánh các tín hiệu này với các vùng địa chỉ được cài đặt để tìm ra Slave đích. Nếu địa chỉ gửi tới không thuộc vùng địa chỉ của bất kỳ Slave nào, **Decoder** sẽ trả về tín hiệu decode error (DECERR). Vùng địa chỉ của mỗi Slave được quy định thông qua các parameter sau:

* SLV\_BASE\_ADDR\_x
* SLV\_RANGE\_x

Công thức tính vùng địa chỉ của mỗi Slave như sau:

* 1. ID identify

Để đảm bảo ID nhận được ở các Slave là duy nhất cho mỗi Master, các bộ **ID Inserter** sẽ chèn thêm các bit phân biệt cho mỗi ID và các bit này là duy nhất tương ứng với mỗi Master.



**Hình 6: ID identify nhận được ở các Slave**

Độ rộng của ID\_x phải đủ lớn để có thể chứa tất cả các bit của ID được tạo ra từ các Master, được cài đặt thông qua parameter RAW\_ID\_WIDTH. Độ rộng của ID\_uniq\_1 phải đủ để có thể phân biệt được các Master được kết nối vào X2X, thông số này được tự động tạo ra sau khi người dùng cài đặt parameter MASTER\_NUM.

* 1. CDAS và ordering

X2X cho phép out-of-order completion cho các transaction không có cùng ID, tuy nhiên các transaction có cùng ID phải được hoàn thành theo đúng thứ tự mà Master tạo ra nó. Để đảm bảo các quy định về ordering, X2X hiện thực 2 cấu hình CDAS: Single Slave và Single Slave per ID.

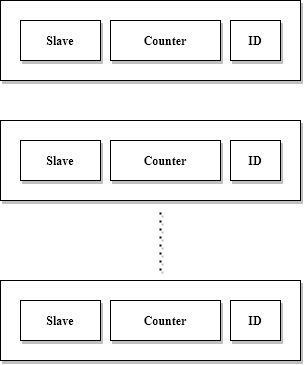
**Ordering:** các bộ ordering lưu thông tin về ID đang có các outstanding transaction, số outstanding transaction hiện hành và Slave đích của các transaction này.

* Cấu hình Single Slave chỉ lưu các giá trị cho duy nhất một ID, do đó tại mỗi thời điểm chỉ có một Slave được truy cập bởi các outstanding transaction đến từ ID này. Các transaction đến các Slave khác của ID này sẽ bị dừng cho đến khi các outstanding transaction trước nó được hoàn thành.

****

**Hình 7: Ordering cho cấu hình Single Slave**

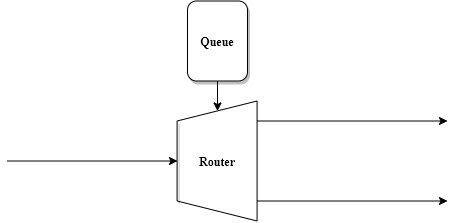
* Cấu hình Single Slave per ID: cấu hình này cho phép một Master có thể gửi multiple outstanding transaction đến các Slave khác nhau. Miễn là tại một thời điểm, mỗi ID chỉ được truy cập đến một Slave duy nhất. Số lượng ID được hỗ trợ được cài đặt thông qua parameter ID\_NUM\_MASTER\_x.



**Hình 8: Ordering cho cấu hình Single Slave per ID**

* 1. Write transaction

Do AXI4 không hỗ trợ WID, dữ liệu trên kênh W phải được đưa đến các Slave theo đúng thứ tự mà Master tạo ra địa chỉ. X2X sử dụng các bộ queue để đảm bảo thứ tự trên. Các queue này lưu thông tin về Slave đích của mỗi write transaction. Độ sâu của mỗi bộ queue được cấu hình thông qua các parameter QUEUE\_WIDTH\_MASTER\_x.



**Hình 9: Các bộ Queue cho W transaction**

**Write threshold:** SI sẽ không nhận thêm write transaction nếu như các outstanding write transaction cho ID đó đã đạt giá trị tối đa , hoặc số phần từ chờ trong queue đã đầy.

**Read threshold**: SI sẽ không nhận thêm read transaction nếu như các outstanding read transaction cho ID đó đã đạt giá trị tối đa.

* 1. Arbiter

X2X phân biệt các bộ phân xử ở phía SI và MI:

* Phía SI: SI chỉ sử dụng một loại phân xử là Balance Round Robin để lựa chọn các đường trả về cho các kênh R và B. Do các slave có thể trả về các out-of-order transaction và các transaction này có thể xảy ra cùng lúc nên bộ phân xử cho các kênh R và B là cần thiết.
* Phía MI: MI hỗ trợ nhiều loại phân xử khác nhau cho các kênh AR và AW. Các bộ phân xử này sẽ được lựa chọn thông qua các bit của thanh ghi X2XMICRx. Các bộ Arbiter này sẽ bỏ qua các request từ các SI đạt đến trạng thái Read Threshold (AR Arbiter) hoặc Write Thresholh (AW Arbiter).
  1. FIFO

Các bộ FIFO sẽ được thêm vào ở các kênh AXI, ở một trong 2 đầu SI, MI hoặc ở cả hai tùy thuộc vào các parameter được cài đặt.

Các bộ FIFO này được dùng làm bộ nhớ đệm giúp giảm thời gian truy cập của các Master hay Slave hoạt động với tần số thấp. Chẳng hạn khi các Master khác nhau muốn ghi dữ liệu vào một Slave hoạt động với tốc độ thấp, hoạt động của toàn bộ hệ thống sẽ chậm lại. Để cải thiện vấn đề này, X2X sẽ ghi các dữ liệu này vào một FIFO rồi cho phép các Master hoạt động tiếp. X2X sẽ từ từ đọc dữ liệu từ FIFO này và ghi vào Slave đích.

* 1. Register Slice

Khi các parameter này được cài đặt, X2X sẽ chèn các Flip Flop vào các đường Critical Path giúp cải thiện timing hệ thống.

* 1. Error Detection

X2X hỗ trợ phát hiện 2 lỗi sau:

* Decode Error (DECERR): địa chỉ gửi tới không thuộc vùng địa chỉ của bất kỳ Slave nào.
* Secure Error (DECERR): một un-secure transaction truy cập xuống một secure Slave (Một secure Slave sẽ có bit secure trên thanh ghi X2XMICRx, tương ứng với MI mà nó kết nối, được bật lên 1).