**VLSI Technology**

**X2X Synchronous Bridge**

**Lê Quang Hưng**

**Nguyễn Hùng Quân**

**2020.09.22**

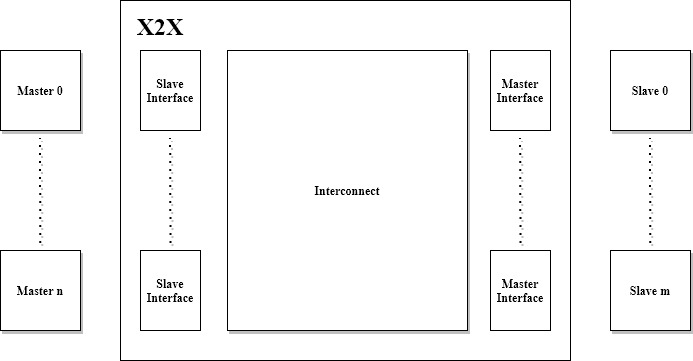
1 ĐẶC ĐIỂM THIẾT KẾ

* 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

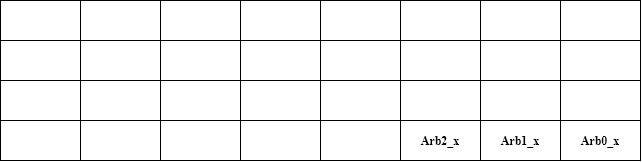
Số lượng master và slave có thể khác nhau. Độ rộng bus dữ liệu phải thống nhất giữa Master, Slave và X2X. Slave interface sẽ được tạo ra tương ứng với số lượng master được hỗ trợ, tương tự cho số Master interface và số Slave. Slave 0 và Master interface 0 là hai khối chức năng mặc định của X2X, điều 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
     1. Các thanh ghi cấu hìnhX2X Slave interface configure register x (X2XSICRx)

Số lượng thanh ghi X2XSICR được tạo ra bằng với số lượng master được hỗ trợ.



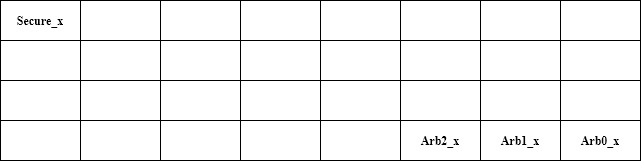
**Hình 2: X2XSICRx**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 1:Bảng mô tả thanh ghi X2XSICRx**

* + 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 3: 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. 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)

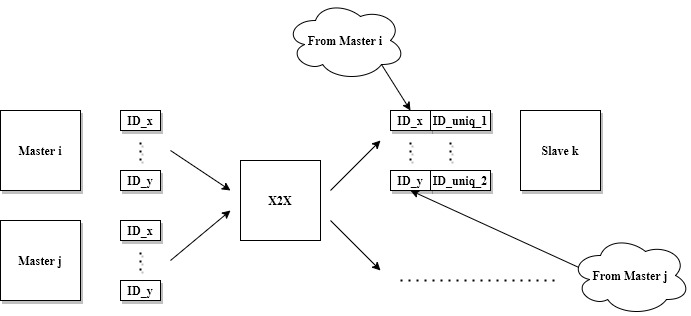
X2X cho phép các transaction được hoàn thành với thứ tự khác với thứ tự khởi tạo, và Slave có thể trả về các re-ordering transaction. Nếu như không có một cơ chế phù hợp điều khiển quá trình này, hiện tượng deadlock có thể xảy ra. Do đó, khi một Master gửi các transaction có cùng ID đến các Slave khác 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:

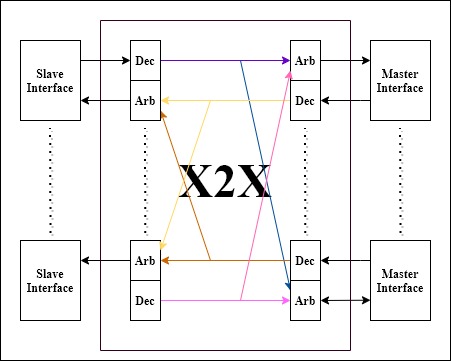
* Các Read transaction với cùng ID phải được hoàn thành theo đúng thứ tự mà Master issue địa chỉ.
* Các Write transaction với cùng ID phải được hoàn thành theo đúng thứ tự mà Master issue địa chỉ.
* Các Transaction của kênh W phải được đưa đến Slave đích theo đúng thứ tự mà Master issue địa chỉ. (do 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 : ID identify

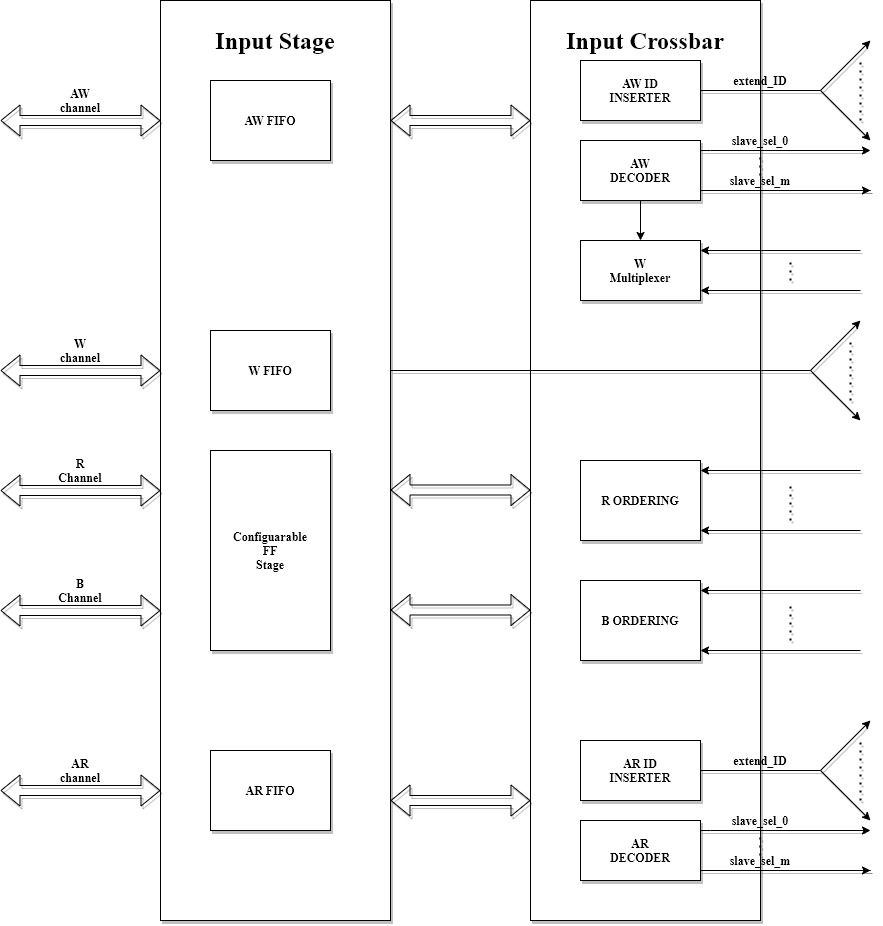
1. CẤU TRÚC X2X SYNCHRONOUS BRIDGE
   1. Tổng quát



Hình : X2X Connection

Thành phần chính của X2X là các bộ Decoder (Dec) và Arbiter (Arb). Dữ liệu đi từ chiều Master xuống Slave cần thông qua các bộ Decoder trước khi đi đến Slave đích, tương tự cho dữ liệu từ Slave lên Master. Bộ Arb ở phía Master Interface là cần thiết vì tại một thời điểm có thể có nhiều Master cùng yêu cầu một Slave. Tương tự ở phía Slave interface, tại mỗi thời điểm có thể có nhiều Slave cùng trả dữ liệu về cho một Master, đồng thời X2X hỗ trợ out-of-order completion do đó một bộ Arb cho mỗi Master là cần thiết.

* 1. Slave Interface



Hình : Slave interface functional block diagram

* + 1. Input Stage

Khối Input Stage giúp X2X giao tiếp với các AXI Master, chứa các khối chức năng chính sau:

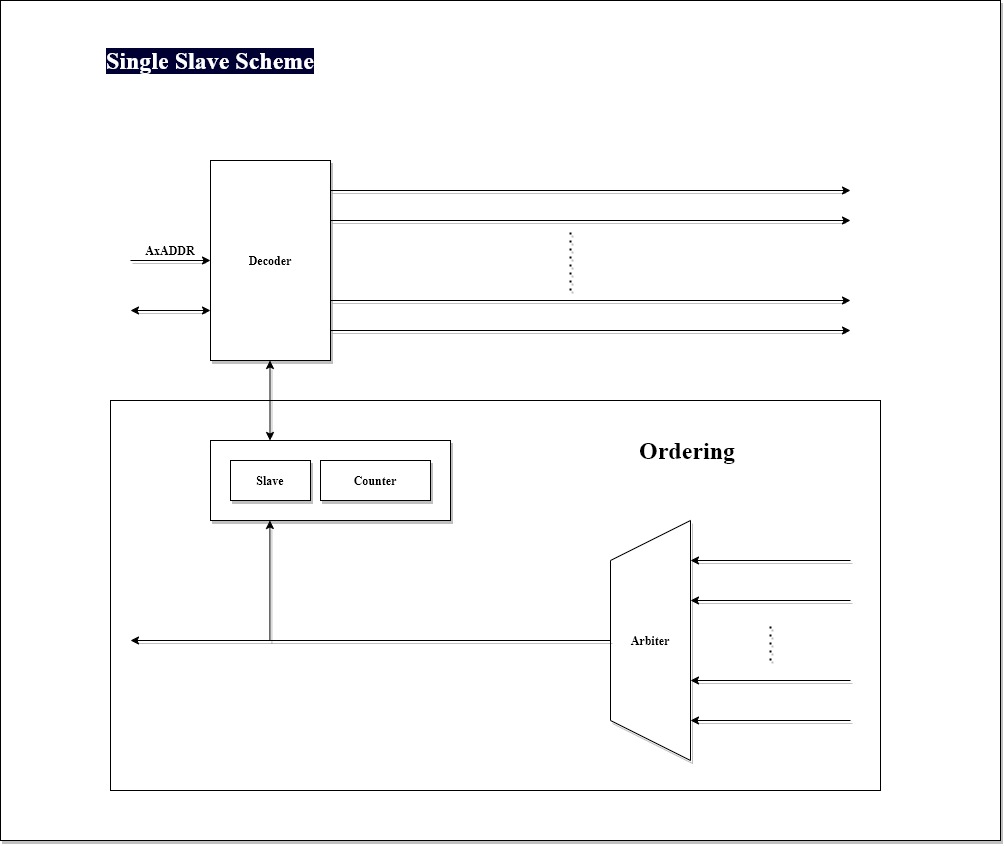
* AR FIFO: chứa tất cả dữ liệu của một transaction từ AR channel, nhờ đó Master có thể liên tục gửi các AR transaction. Con trỏ đọc của AR FIFO được đếm lên ngay khi dữ liệu được đưa đến Slave tương ứng
* AW FIFO: chứa tất cả dữ liệu của một transaction từ AW channel, nhờ đó Master có thể liên tục gửi các AW transaction. Con trỏ đọc của AW FIFO chỉ được đếm lên khi tất cả các phần tử trong một burst dữ liệu, được lưu trong W FIFO được đưa đến Slave tương ứng.
* W FIFO: chứa tất cả dữ liệu của một transaction từ W channel, nhờ đó Master có thể liên tục gửi các W transactions. Con trỏ đọc của W FIFO được đếm lên khi tín hiệu output từ W Multiplexer được bật lên mức cao.
* Configurable FF Stage: người dùng có thể cấu hình thêm môt tầng Flip Flop cho các **AXI channel,** giúp cải thiện timing của hệ thống.

Các Master có thể liên tục gửi các transaction trên các kênh AR, AW và W cho đến khi các FIFO tương ứng đầy. Con trỏ đọc của AR FIFO được đếm lên ngay khi dữ liệu từ nó được đưa đến Slave tương ứng. Tuy nhiên, con trỏ đọc của

* + 1. Input Crossbar

Khối Input Crossbar làm nhiệm vụ phân bổ dữ liệu từ các kênh AR, AW và W đến các Slave đích, đồng thời tiến hành phân xử và nhận dữ liệu từ các kênh R, B. Input Crossbar chứa các khối chức năng chính sau:

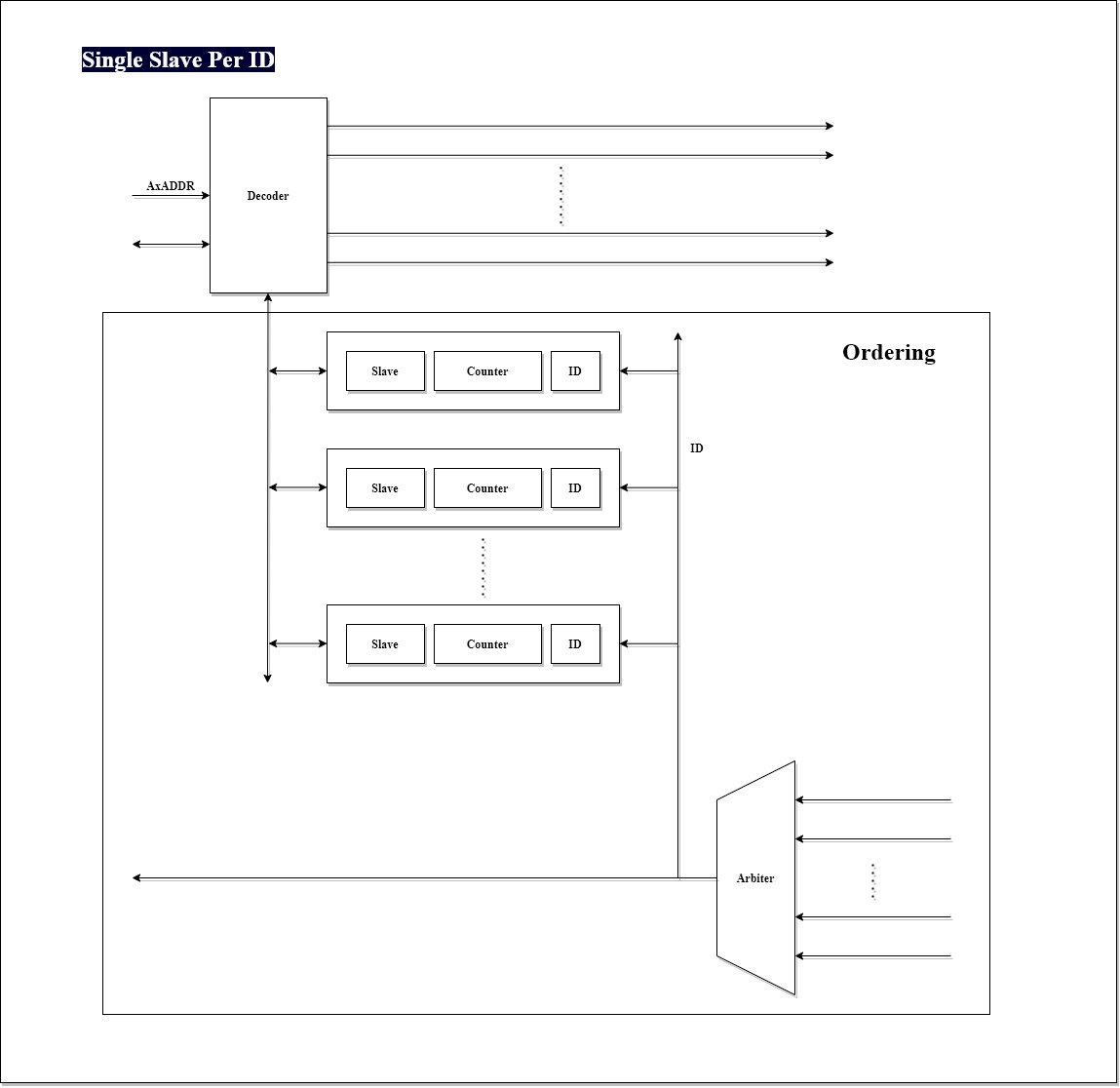
* AW ID Inserter: chèn thêm các trường phân biệt vào AWID trước khi đưa đến Slave, giúp đảm bảo mỗi AWID nhận được ở các Slave là “duy nhất” (unique) cho từng Master khác nhau.
* AW Decoder: decode trường địa chỉ từ AWADDR, giúp X2X biết được Master đang muốn truy cập đến Slave nào cho Write transaction.
* AR ID Inserter: chèn thêm các trường phân biệt vào ARID trước khi đưa đến Slave, giúp đảm bảo mỗi ARID nhận được ở các Slave là “duy nhất” (unique) cho từng Master khác nhau.
* AR Decoder: decode trường địa chỉ từ ARADDR, giúp X2X biết được Master đang muốn khi cập đến Slave nào cho Read transaction.
* W Multiplexer: chọn tín hiệu response tương ứng từ các Slave trước khi đưa đến W FIFO.
* R Ordering: chứa các bộ phân xử và khối chức năng đảm bảo order của các read transaction.
* W Ordering: chứa các bộ phân xử và khối chức năng đảm bảo order của các write transaction.
  + - 1. Ordering
* Single Slave Scheme:



Hình : Ordering with Single Slave scheme

Các bộ Ordering phải kiểm tra các điều kiện của CDAS trước khi gửi yêu cầu đến phía slave. Ordering kiểm tra địa chỉ Slave đích hiện tại có trùng với địa chỉ của các outstanding transaction đang hiện hành hay không:

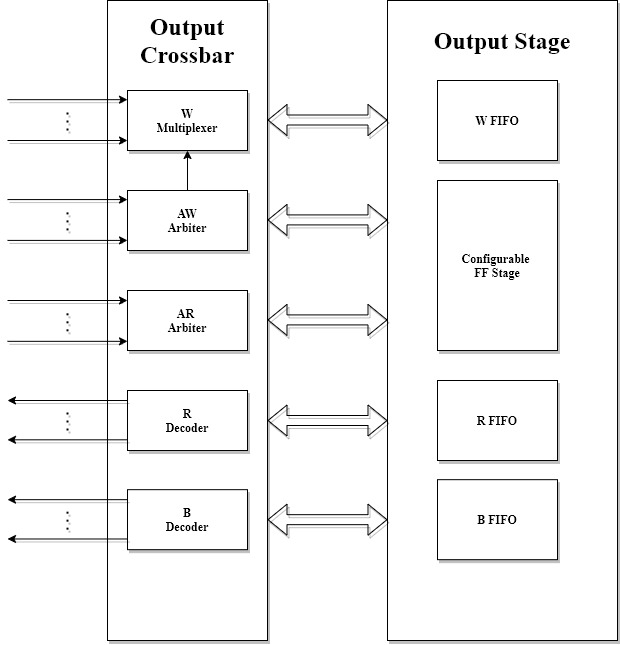
* + Trùng: cho phép transaction được đưa đến Slave, đồng thời tăng biến đếm Counter. Sau khi có một transaction được hoàn thành, Counter sẽ giảm giá trị.
  + Không trùng: khi Master muốn gửi transaction đến một Slave khác, việc gửi dữ liệu sẽ bị buộc dừng lại cho đến khi tất cả các outstanding transaction hiện hành được hoàn thành.
* Single Slave Per ID:



Hình : Ordering with Single Slave Per ID scheme

Các bộ Ordering phải kiểm tra các điều kiện của CDAS trước khi gửi yêu cầu đến phía slave. Ordering kiểm tra các outstanding transaction có cùng ID có cùng truy cập đến một Slave hay không:

* + Trùng: cho phép transaction được đưa đến Slave, đồng thời tăng biến đếm Counter của ID tương ứng. Sau khi có một transaction được hoàn thành, Counter của ID tương ứng sẽ giảm giá trị.
  + Không trùng: khi Master muốn gửi transaction đến một Slave khác, việc gửi dữ liệu sẽ bị buộc dừng lại cho đến khi tất cả các outstanding transaction hiện hành được hoàn thành.
  1. Master Interface



Hình : Master Interface functional block diagram

* + 1. Output Crossbar

Khối Output Crossbar làm nhiệm vụ phân xử và nhận dữ liệu từ các kênh AR, AW và W từ các Master, đồng thời phân bổ dữ liệu từ các kênh R, B đến Master đích. Output Crossbar chứa các khối chức năng chính sau:

* AW Arbiter: phân xử dữ liệu từ các kênh AW, AW Arbiter sẽ không trả tín hiệu grant cho Decoder yêu cầu nó cho đến khi phần tử cuối cùng trong Burst dữ liệu hiện hành của kênh W được đưa đến phía Slave (hoặc W FIFO – khác với W FIFO ở Slave interface).
* W Multiplexer: tại mỗi thời điểm, mỗi Master chỉ có thể yêu cầu một W Multiplexer hoạt động, nhờ đó chỉ có các Slave đích mới đọc được dữ liệu từ kênh W tương ứng.
* AR Arbiter: phân xử dữ liệu từ các kênh AR, AW Arbiter sẽ trả tín hiệu grant cho Decoder yêu cầu nó ngay sau khi Slave tương ứng nhận được AR transaction.
* R Decoder: phân bổ dữ liệu nhận được từ kênh R đến Master đích.
* B Decoder: phân bổ dữ liệu nhận được từ kênh B đến Master đích.
  + 1. Output Stage

Khối Output Stage giúp X2X giao tiếp với các AXI Slave. Output Stage chứa các khối chức năng chính sau:

* W FIFO: chứa dữ liệu từ kênh W. Khi giao tiếp với các Slave chạy với tốc độ thấp, Master chỉ việc ghi dữ liệu vào W FIFO và tiếp tục hoạt động. Đồng thời W FIFO cũng giúp hỗ trợ outstanding write transaction.
* R FIFO: chứa dữ liệu đọc được từ kênh R của Slave. Khi kênh R của Master đích đang bận, Slave chỉ việc ghi dữ liệu vào R FIFO và tiếp tục xử lý các transaction khác.
* B FIFO: chứa dữ liệu đọc được từ kênh B của Slave. Khi kênh B của Master đích đang bận, Slave chỉ việc ghi dữ liệu vào B FIFO và tiếp tục xử lý các transaction khác.
* Configurable FF Stage: người dùng có thể cấu hình thêm môt tầng Flip Flop cho các **AXI channel,** giúp cải thiện timing của hệ thống.