Clock Domain Crossing (CDC)

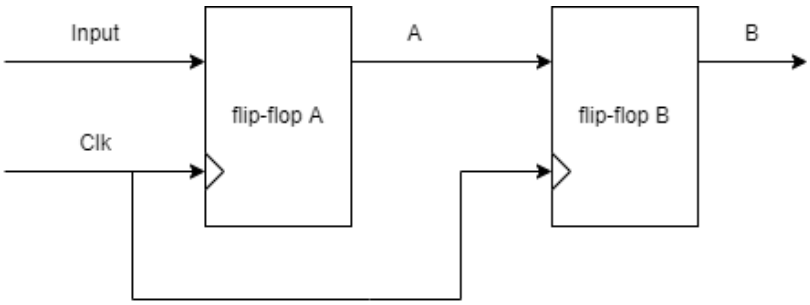
Bùi Anh Khoa

October 2022

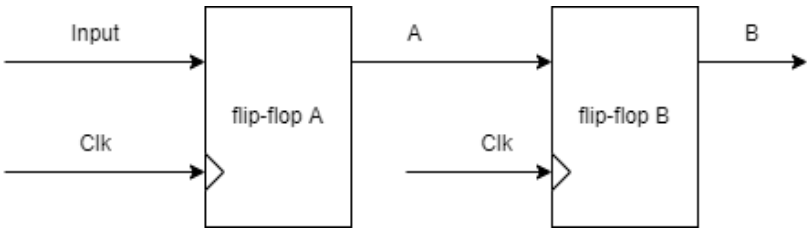
1 Clock Domain Crossing (CDC) là gì?

Clock Domain Crossing được định nghĩa là việc truyền một tín hiệu, hay dữ liệu giữa hai vùng mạch số bất đồng bộ. Các mạch số sử dụng flip-flop luôn cần tín hiệu xung clock để chốt dữ liệu đầu ra. Trong những chip SoCs, với việc tích hợp nhiều module trong cùng một con chip đã dẫn đến việc chuyển đổi dữ liệu giữa các vùng có xung clock với tần số khác nhau, hoặc từ các nguồn khác nhau là điều không thể tránh khỏi. Nhưng khi ta truyền dữ liệu trực tiếp giữa 2 vùng sử dụng xung clock khác nhau thì việc bất đồng bộ sẽ xảy ra, và gây ra xung đột.

Hai vùng được xem là đồng bộ (synchronous) khi và chỉ khi tất cả các flip-flop được điều khiển bởi một xung clock từ một nguồn duy nhất. Hai vùng được xem là bất đồng bộ (asynchronous) khi các flip-flop ở vùng đó sử dụng 2 xung clock khác nhau về tần số, ngay cả khi chúng sử dụng xung clock có cùng tần số nhưng lấy từ 2 nguồn khác nhau cũng được coi là bất đồng bộ.



Hình 1: hai flip-flop đồng bộ



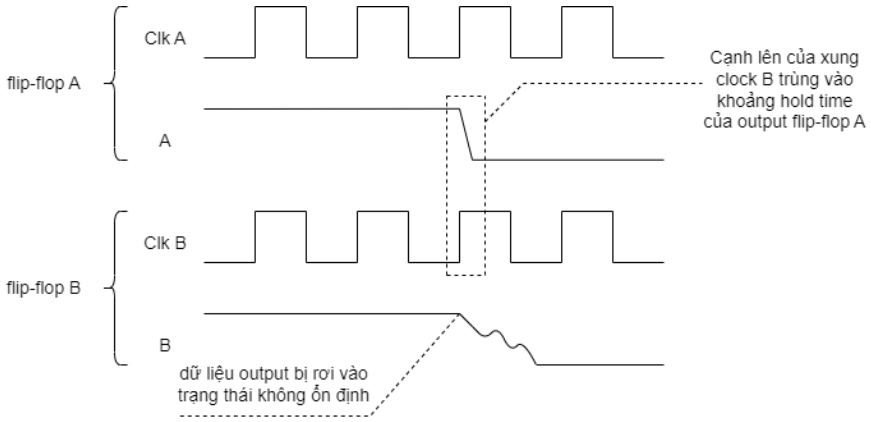
Hình 2: hai flip-flop bất đồng bộ

2 Vì sao cần sử dụng các phương pháp Clock Domain Crossing (CDC)?

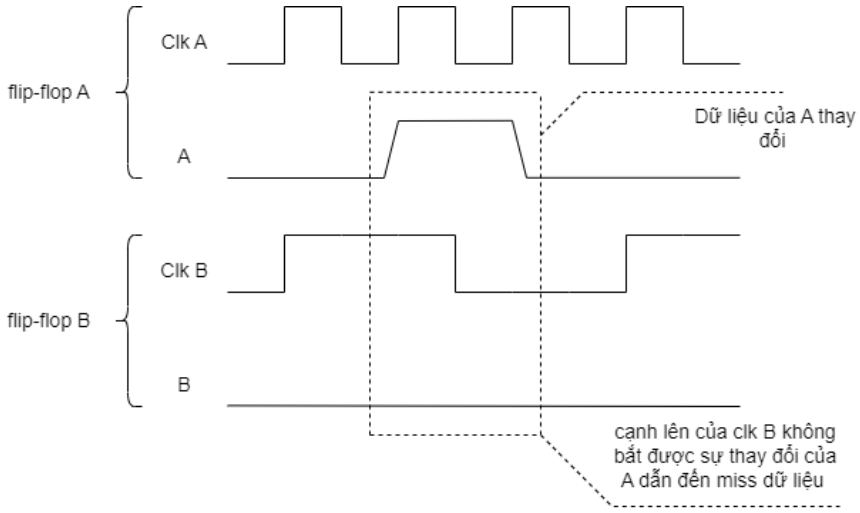
Trước tiên ta cần tìm hiểu về định nghĩa "metastability". "Metastability" được định nghĩa là trạng thái mà dữ liệu không ổn định, dữ liệu "lơ lửng" nghĩa là output đầu ra của dữ liệu không đạt được tới 1 mà cũng không hoàn toàn là 0. "Metastability" không thể tránh khi thiết kế phần cứng sử dụng nhiều xung clock, nhưng có thể kiểm soát chúng bằng các phương pháp Clock Domain Crossing (CDC).

"Metastability" xuất hiện bởi sự ảnh hưởng của "setup time" và "hold time". "Setup time" là thời gian mà dữ liệu đưa vào flip-flop cần được giữ ổn định và không được thay đổi trước cạnh lên của clock. "Hold time" là thời gian mà dữ liệu cần được giữ ổn định và không được thay đổi sau cạnh lên của clock.

Ở ví dụ dưới đây, ta có 2 flip-flop A và B sử dụng 2 xung clock từ 2 nguồn khác nhau. Khi output của flip-flop A chuyển trạng thái từ 1 xuống 0, sẽ cần một khoảng thời gian "hold time" để ổn định, và khi này xung clock của flip-flop B ngay lập tức kích hoạt cạnh lên khi output của flip-flop A vẫn còn đang "lơ lửng", làm cho flip-flop B chốt dữ liệu output là "Metastability". Và phải đợi sau đó 1 chu kì xung clock, flip-flop B mới cập nhật lại được đúng dữ liệu 0. Ngoài ra, khi truyền dữ liệu từ vùng có tần số xung clock nhanh đến cùng có tần số xung clock chậm hơn cũng dễ gây ra lỗi miss dữ liệu.



Hình 3: setup time/ hold time dẫn đến "metastability"

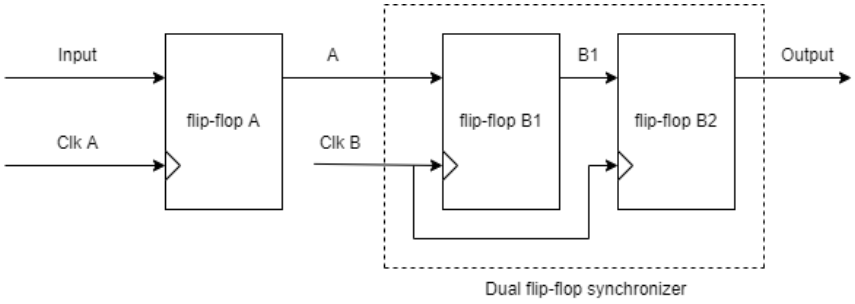


Hình 4: lỗi miss dữ liệu

3 Một số kĩ thuật Clock Domain Crossing (CDC)

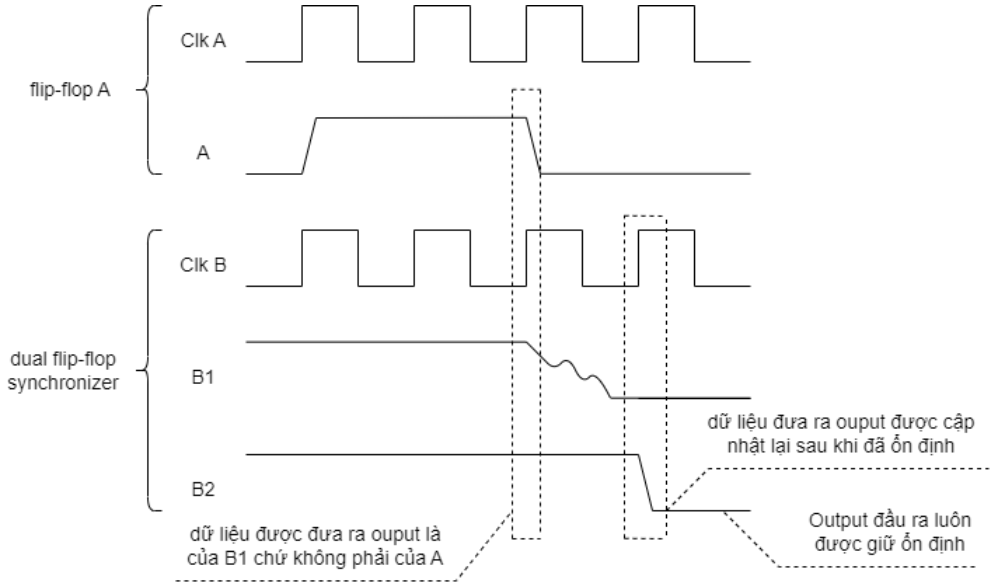
3.1 Đầu ra xuất hiện “metastability”.

Giải quyết vấn đề dữ liệu đầu ra xuất hiện “metastability” là ta giải tuyết vấn đề của setup/hold time. Một trong những kĩ thuật phổ biến nhất là sử dụng bộ đồng bộ dùng 2 flip-flop (dual flip-flop synchronizer hoặc 2-FF synchronizer). Ở kỹ thuật này, ta sẽ ghép thêm một flip-flop đồng bộ với flip-flop B



Hình 5: CDC sử dụng phương pháp dual flip-flop synchronizer

Ở ví dụ này, khi có cạnh lên của xung clock B, ngay lập tức flip-flop B2 sẽ lấy dữ liệu ouput của flip-flop B1 và đưa vào input B2 rồi trả ra output B2. Trong trường hợp flip-flop B1 nhận input là dữ liệu “metastability”, khi dữ liệu “metastability” của B1 truyền đến output B1, thì flip-flop B2 đã chốt tín hiệu trước đó của B1 và tín hiệu“metastability” bị loại bỏ. Đến chu kì xung clock tiếp theo, khi B1 đã ổn định, B2 sẽ chốt lại tín hiệu của output B1 ra ngoài.



Hình 6: mô tả hoạt động của phương pháp dual flip-flop synchronizer

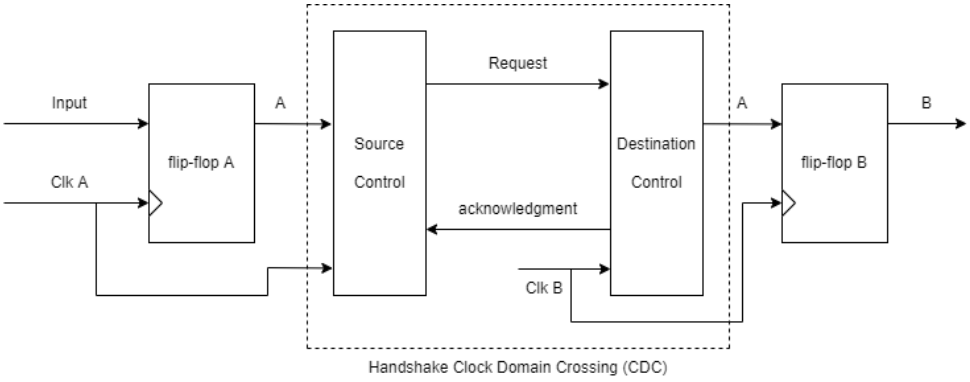
Để tránh rủi ro xuất hiện “metastability”, một số module ghép thêm không chỉ 2, mà có thể 3, 4 flip-flop vào trước output. Và các flip-flop thêm vào phải đặt càng gần flip-flop gặp “metastability” càng tốt, để giảm thời gian delay trên đường truyền.

3.2 Truyền một chuỗi data liên tục

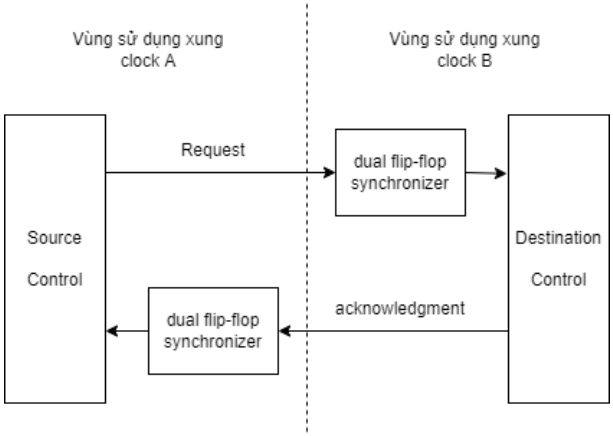
Truyền 1 chuỗi dữ liệu liên tục giữa 2 miền xung clock khác nhau với phương pháp dual flip-flop synchronizer là bất khả thi. Vì với việc "metastability" xuất hiện vào thời điểm nào và kết thúc ở thời điểm nào là không thể đoán trước; khi sử dụng bộ dual flip-flop synchronizer trong CDC, để "metastability" trở về trạng thái ổn định ta sẽ tốn một hoặc một vài chu kì xung clock. Do vậy ta không thể đảm bảo một bên gửi dữ liệu liên tục và một bên nhận dữ liệu truyền đến liên tục.

Do vậy nếu chuỗi dữ liệu được truyền liên tục (mỗi chu kì sẽ truyền đi 1 bit), khi gặp "metastability", dữ liệu tiếp theo của chuỗi vẫn tiếp tục được truyền đi, nhưng ở vùng nhận sẽ phải dành ra một vài chu kì xung clock để ổn định "metastability". Việc này sẽ làm sai lệch các bit trong chuỗi cần truyền. Và sự sai lệch này vùng truyền và vùng nhận không hề nhận ra, gây nên lỗi dữ liệu. Vậy nên ta phải cần một giao thức để vùng truyền và vùng nhận giao tiếp với nhau, để nhận biết liệu dữ liệu đã được nhận hay chưa. Và đó là lúc cần đến "handshake Clock Domain Crossing (CDC)".

Để hiện thực bộ "handshake Clock Domain Crossing (CDC)", ta sẽ xây dựng 2 bộ điều khiển gọi là "source control" và "destination control". "Source control" gửi yêu cầu thay đổi dữ liệu đến "destination control", tiếp đó "destination control" sẽ gửi dữ liệu, ghi vào flip-flop B, đồng thời sẽ phản hồi lại cho "source control". Khi nhận được tín hiệu phản hồi, "source control" mới tiến hành gửi tiếp dữ liệu tiếp theo. Và để tránh xuất hiện "metastability" khi truyền tín hiệu "request" và "ack" giữa "source control" và "destination control" ta chèn vào đường truyền bộ dual flip-flop synchronizer



Hình 7: kỹ thuật Handshake Clock Domain Crossing

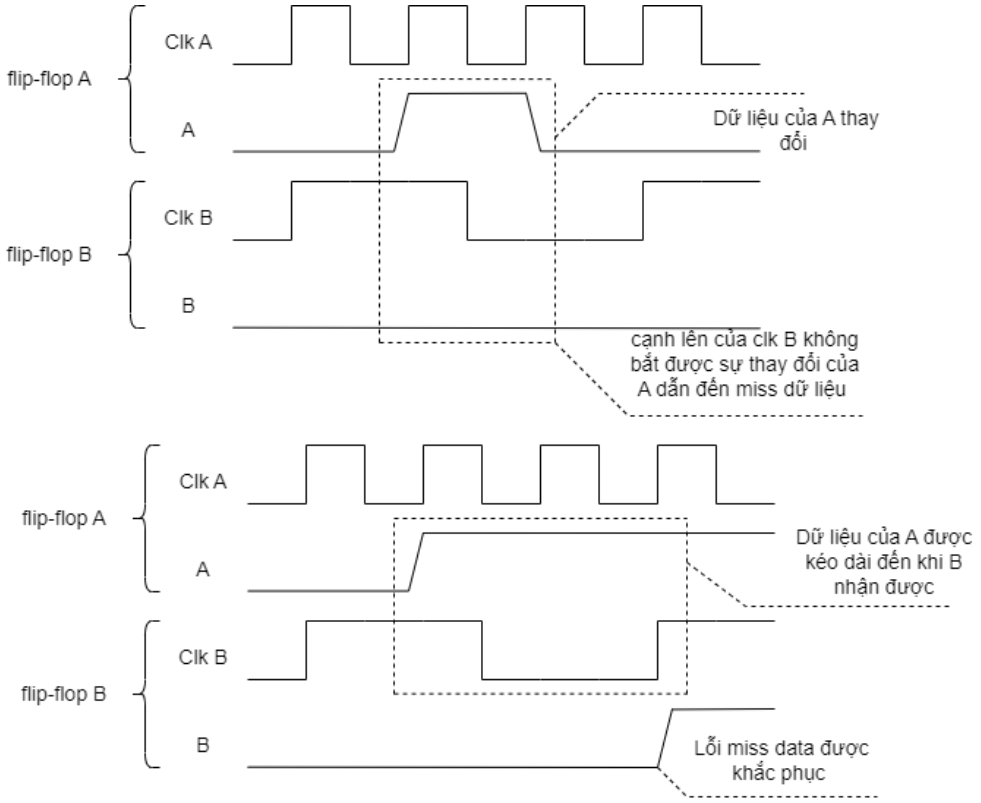


Hình 8: bộ Handshake Clock Domain Crossing hoàn chỉnh

3.3 Xuất hiện lỗi miss dữ liệu

Khi truyền dữ liệu từ vùng có tần số xung clock nhanh đến cùng có tần số xung clock chậm hơn sẽ dễ gây ra lỗi miss dữ liệu. Để tránh việc mất dữ liệu này, "handshake Clock Domain Crossing (CDC)" lại là một giải pháp.

Ở phương pháp này, ta sẽ kéo dài tín hiệu cần truyền đến khi vùng nhận nhận được dữ liệu và gửi phản hồi về vùng truyền, khi nhận được tín hiệu phản hồi về, vùng truyền mới bắt đầu truyền dữ liệu khác.



Hình 9: khắc phục lỗi miss data

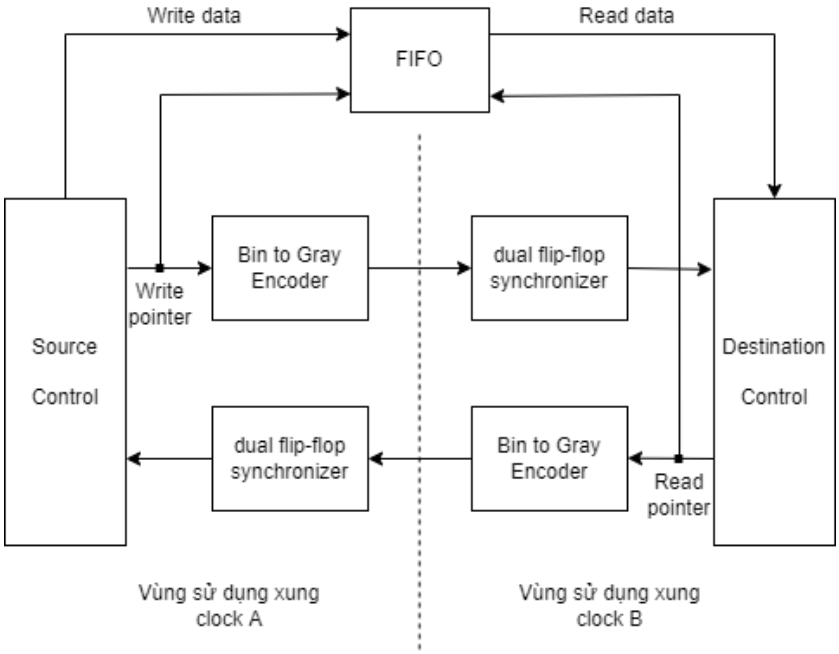
3.4 Tăng tốc độ truyền dữ liệu giữa 2 vùng xung clock khác nhau

Đối với phương pháp Handshake Clock Domain Crossing, đầu ra ở miền "truyền" phải đợi đến khi nhận được tín hiệu "ack" trả về thì mới có thể cập nhật dữ liệu mới, và dữ liệu truyền được sau mỗi chu kì "request" và "ack" chỉ là 1 bit. Điều này làm giảm tốc độ truyền dữ liệu giữa 2 vùng. Một phương pháp phổ biến để giải quyết vấn đề tốc độ đó chính là sử dụng Dual Clock Asynchronous FIFO (FIFO sử dụng 2 miền xung clock bất đồng bộ).

Với phương pháp này ta sẽ cần 1 FIFO để lưu dữ liệu. Khi này vùng "truyền" có thể thoải mái ghi dữ liệu vào FIFO, rồi cập nhật dữ liệu mới mà không cần phải đợi tín hiệu phản hồi "ack". Với việc sử dụng FIFO, ta phải tuân theo nguyên tắc cơ bản nhất của FIFO, đó chính là không được ghi dữ liệu vào một FIFO đang ở trạng thái "full" (để tránh việc ghi đè, mất dữ liệu) và không được đọc dữ liệu từ một FIFO đang ở trạng thái "empty" (để tránh việc đọc dữ liệu không mong muốn, dữ liệu bị sai).

Phương pháp này hoạt động như sau:

- Khối “source control” sẽ chịu trách nhiệm quản lý “write pointer” (địa chỉ ghi dữ liệu vào FIFO).
- Khối “destination control” sẽ chịu trách nhiệm quản lý “read poiner” (địa chỉ đọc dữ liệu ra khỏi FIFO).
- Khi dữ liệu ra khỏi vùng “truyền”, dữ liệu sẽ được ghi vào FIFO ở vị trị trí mà “write pointer” chỉ định. Có thể ghi liên tục vào FIFO, miễn là FIFO chưa đạt đến trạng thái “full”.
- Khi vùng “nhận” đọc dữ liệu, dữ liệu sẽ được đọc ra khỏi FIFO ở vị trí mà “read pointer” chỉ định. Có thể đọc liên tục từ FIFO, miễn là FIFO không ở trạng thái “empty”
- Kiểm soát trạng thái “empty” của FIFO: tín hiệu “write pointer” ở “source control” sau khi được cập nhật sẽ được truyền đến “destination control” và “destination control” sẽ sử dụng tín hiệu “write poiner” nhận được và tín hiệu “read pointer” để tính toán ra liệu FIFO có ở trạng thái “empty” hay không.
- Kiểm soát trạng thái “full” của FIFO: tín hiệu “read pointer” ở “destination control” sau khi được cập nhật sẽ được truyền đến “source control” và “source control” sẽ sử dụng tín hiệu “read poiner” nhận được và tín hiệu “write pointer” để tính toán ra liệu FIFO có ở trạng thái “full” hay không.
- Ta không truyền trực tiếp “read pointer” và “write pointer” giữa 2 miền xung clock vì không thể truyền một lần nhiều hơn 1 bit dữ liệu qua bộ dual flip-flop synchronizer. Vậy nên giải pháp là sử dụng bộ chuyển từ binary sang gray code. Với việc sử dụng gray code, khi truyền dữ liệu, ta chỉ cần cập nhật 1 bit, điều này sẽ giúp ta giảm bớt rủi ro gặp meta trên đường truyền, và dễ dàng nhận ra lỗi khi dữ liệu vào/ra FIFO có vấn đề.



Hình 10: CDC sử dụng Dual Clock Asynchronous FIFO

Ở phương pháp này, dữ liệu ta thực sự truyền giữa 2 vùng A và B là dữ liệu “write pointer” và “read pointer” để kiểm soát trạng thái “full” và “empty” của FIFO. Còn dữ liệu cần truyền giữa 2 vùng A và B được lưu vào FIFO, vùng A sẽ lưu dữ liệu cần truyền vào FIFO, và vùng B sẽ đọc dữ liệu cần nhận ra khỏi FIFO. Đây cũng chính là phương pháp phổ biến nhất khi ta thực hiện truyền dữ liệu giữa 2 vùng sử dụng xung clock khác nhau.

4 Kết luận

Việc truyền dữ liệu giữa 2 vùng có xung clock khác nhau thông thường sẽ gây ra xung đột về thời gian và gây ra "metastability". Việc hiểu và sử dụng tốt các phương pháp CDC là rất cần thiết để loại bỏ các rủi ro do vấn đề thời gian gây ra.

Tùy vào từng nhu cầu sẽ có cách thiết kế và sử dụng các phương pháp CDC khác nhau, trên đây chỉ trình bày những phương pháp CDC phổ biến, thường gặp chứ không phải toàn bộ.

Tài liệu

[1] Anysilicon, *Clock Domain Crossing (CDC)*. Truy cập từ: https://anysilicon.com/clock-domain-crossing-cdc/

[2] Hardwarebee, *Clock Domain Crossing Techniques for FPGA*. Truy cập từ https://hardwarebee.com/clock-domain-crossing-techniques-for-fpga/

[3] Nandland, *Crossing Clock Domains in an FPGA*. Truy cập từ https://nandland.com/lesson-14-crossing-clock-domains/