ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÀI TẬP LỚN MÔN THIẾT KẾ LUẬN LÝ

ĐỀ TÀI 1: FIFO - FIRST IN FIRST OUT

LỚP L05 -- NHÓM 6 --- HK222

NGÀY NỘP 10/05/2023

Giảng viên hướng dẫn: Nguyễn Thiên Ân

Sinh viên thực hiện	Mã số sinh viên	Điểm số
Phạm Hồng My Sa	2112173	
Võ Ngọc Tú	2213857	
Lê Đăng Huy	2211186	
Trịnh Thu Trang	2213561	

Thành phố Hồ Chí Minh – 2023

MỤC LỤC

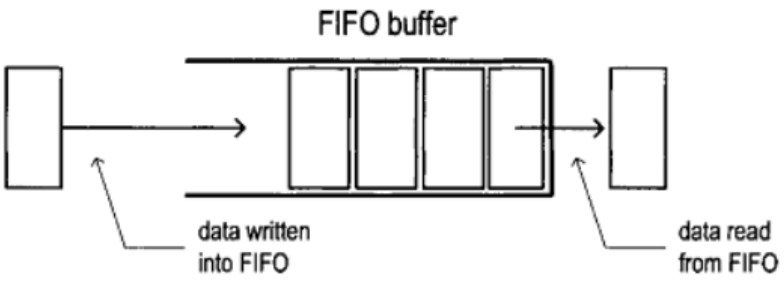
I. First In First Out (FIFO) – Theory and Applications

1. Theory – Khái niệm

Trong mọi item của các thiết bị kĩ thuật đều sẽ có sự chuyển đổi dữ liệu giữa các bảng mạch in (printed circuit boards PCBs – hay còn gọi là các bo mạch in). Bộ lưu trữ trung gian (Intermediate storage) hoặc bộ nhớ đệm (buffer) luôn cần thiết khi dữ liệu đến bảng mạch in nhận với tốc độ cao hoặc theo từng đợt nhưng lại không được xử lí một cách mượt mà hoặc tốc độ xử lí không nhanh.

Ta có thể dễ dàng bắt gặp các bộ nhớ đệm (buffer) trong cuộc sống hằng ngày, ví dụ như hàng đợi khách hàng tại điểm thanh toán tại các siêu thị hoặc hàng xe ô tô di chuyển tại các nơi có đèn giao thông. Điểm thanh toán trong siêu thị làm việc tuần tự và chậm rãi, trong khi lượng khách hàng lại đến không ổn định; vậy nên nếu như có nhiều khách hàng tiến đến thanh toán cùng một lúc, một hàng đợi sẽ được hình thành và hoạt động theo nguyên tắc ai đến trước sẽ được phục vụ trước (first come, first served). Còn đối với đèn giao thông, nó chỉ cho phép các xe ô tô đi qua theo từng đợt trong thời gian nhất định.

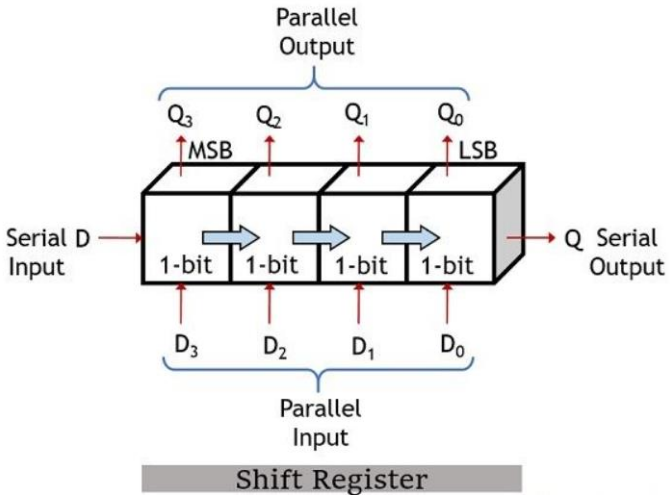
FIFO là một bộ đệm đặc biệt, FIFO viết tắt cho First In First Out, nghĩa là dữ liệu nào được ghi vào bộ đệm trước sẽ ra khỏi bộ đệm trước. Có các loại bộ đệm khác như LIFO (Last In First Out), thường được gọi là bộ nhớ stack, và bộ nhớ dùng chung (shared memory). FIFO có thể hiện thực bằng phần mềm hoặc phần cứng. Sự lựa chọn giữa giải pháp phần mềm hay phần cứng phụ thuộc vào ứng dụng đang được sử dụng và các tính năng mong muốn. Khi các yêu cầu thay đổi, một phần mềm FIFO sẽ dễ dàng thích nghi với sự thay đổi đó hơn bằng cách điều chỉnh chương trình, trong khi một phần cứng FIFO sẽ phải cần đến một board layout mới. Phần mềm thì linh hoạt hơn phần cứng rất nhiều, tuy nhiên bù lại, lợi ích của phần cứng FIFO nằm ở tốc độ của nó.



Bất kì bộ nhớ nào mà dữ liệu được viết vào đầu tiên cũng sẽ ra đầu tiên khi dữ liệu được đọc là bộ nhớ FIFO. Có 3 loại FIFO:

a. *Shift register – FIFO*

Thanh ghi dịch FIFO với số dữ liệu được phép lưu trữ là cố định, do đó, cần có sự đồng bộ giữa thao tác đọc và ghi vì một dữ liệu phải được đọc mỗi khi một dữ liệu khác được ghi vào. Về cơ bản, thanh ghi dịch là một mạch FIFO đơn hướng, nó dịch từng bit đơn của dữ liệu được đưa vào input ra output trong mỗi chu kì clock.



b. *Exclusive read/write FIFO*

FIFO chỉ đọc/ghi (trong một thời điểm xác định chỉ có thể thực hiện thao tác đọc hoặc ghi, không thực hiện đồng thời) với số lượng dữ liệu được phép lưu trữ là không cố định, bởi vì cấu trúc bên trong, cần có sự đồng bộ giữa các thao tác đọc và ghi. Trong cơ chế này, có sự phụ thuộc giữa việc đọc và ghi dữ liệu, có mối quan hệ thời gian giữa write clock và read clock, ví dụ, không được phép chồng chéo (overlap) read clock và write clock với nhau.

c. *Concurrent read/write FIFO*

FIFO đọc/ghi đồng thời, là FIFO với số lượng dữ liệu được phép lưu trữ là không cố định, và có thể có sự không đồng bộ giữa thao tác đọc và ghi. Trong cơ chế này, không có sự phụ thuộc giữa việc đọc và ghi dữ liệu, chúng ta có thể đồng thời đọc và ghi chồng chéo (overlap) nhau. Điều này đồng nghĩa với việc hai hệ thống với tần số khác nhau có thể kết nối vào cơ chế FIFO này. Người thiết kế không cần lo về việc đồng bộ hóa hai hệ thống bởi vì việc này sẽ được xủ lí trong mô hình FIFO, việc đọc/ghi đồng thời phụ thuộc vào các tín hiệu điều khiển cho việc đọc và ghi, được phân thành hai

nhóm: FIFO đồng bộ (synchronous FIFO, read write sử dụng cùng clock) và FIFO bất đồng bộ (asynchronous FIFO, read clock khác write clock).

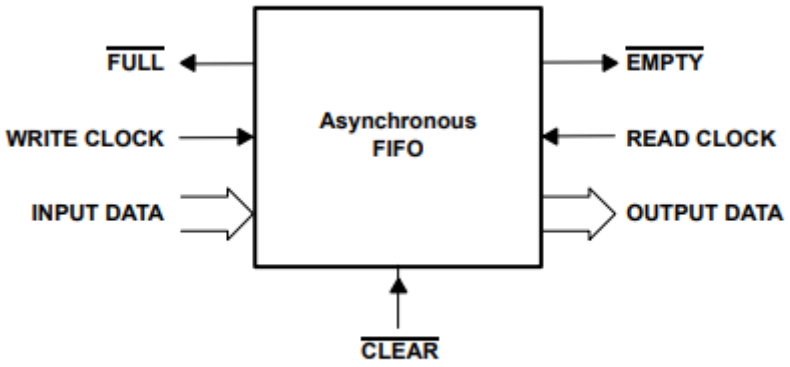


Figure 8. Connections of an Asynchronous FIFO

Minh họa cho FIFO không đồng bộ (2 clock khác nhau)

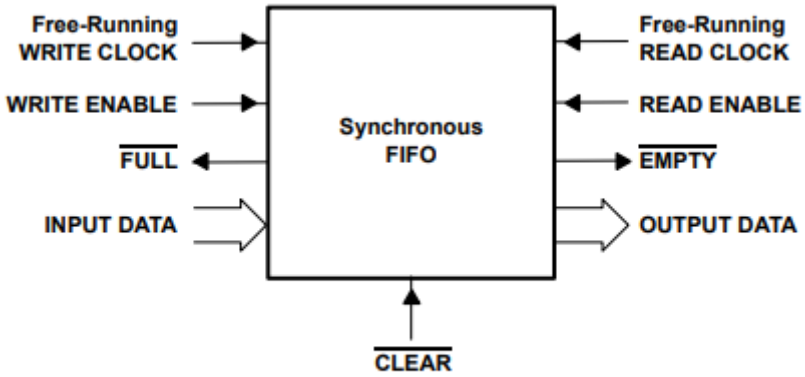


Figure 11. Connections of a Synchronous FIFO

Minh họa cho FIFO đồng bộ

Trong bài báo cáo này, nhóm sẽ nói về thiết kế và hiện thực FIFO đồng bộ. Mọi hệ thống xử lý kỹ thuật số đều hoạt động đồng bộ với tín hiệu clock trên toàn hệ thống. Hệ thống đếm này sẽ cứ tiếp tục chạy ngay cả khi không có hành động nào được thực thi. Tín hiệu kích hoạt (enable signals: bao gồm write enable và read enable), còn thường được gọi là tín hiệu chọn chip, bắt đầu thực hiện đồng bộ các thao tác và đọc trong các thiết bị khác nhau, chẳng hạn như bộ nhớ (memories) và cổng (ports).

2. *Implement – Hiện thực*

FIFO có thể hiện thực bằng thanh ghi dịch – phần cứng (hardware shift register) hoặc dùng các cấu trúc bộ nhớ khác, điển hình là FIFO vòng (circular buffer, ring buffer, array-base bufer), hoặc danh sách (list). FIFO dạng List được biểu diễn thông qua cấu trúc dữ liệu (data structure) Queue; ở đây chúng ta sẽ nghiên cứu về mô hình FIFO được hiện thực thông qua cấu trúc Ring buffer.

Ring Buffer (Circular Buffer): Trong khoa học máy tính, buffer vòng là một cấu trúc dữ liệu sử dụng một buffer đơn, có kích thước cố định và được kết nối đầu cuối với nhau (end-to-end). Như vậy, buffer vòng không có điểm cuối thực tế và nó sẽ lặp lại (loop) vòng quanh trong buffer.

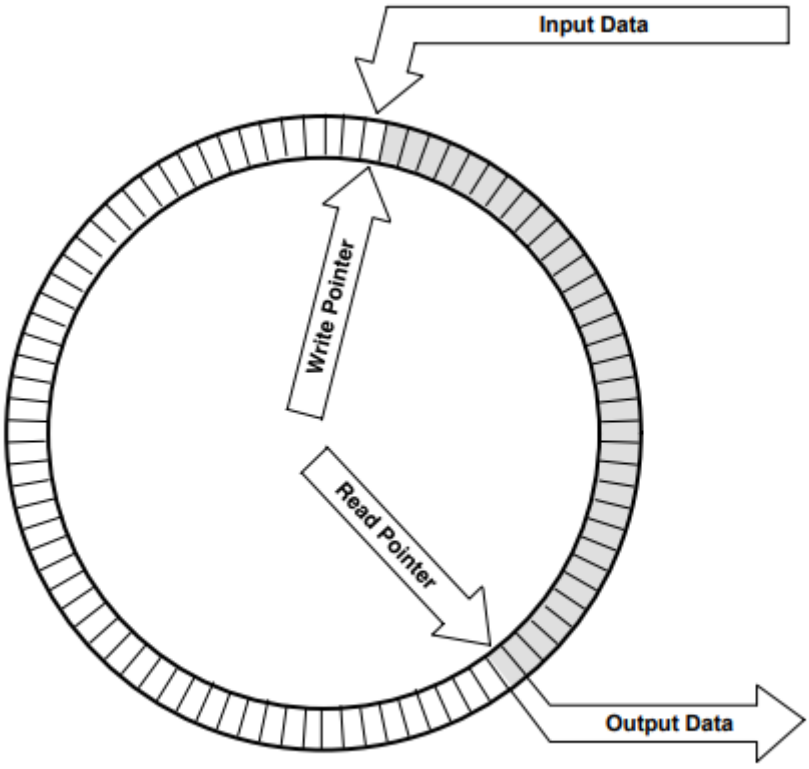
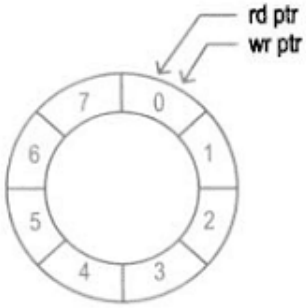
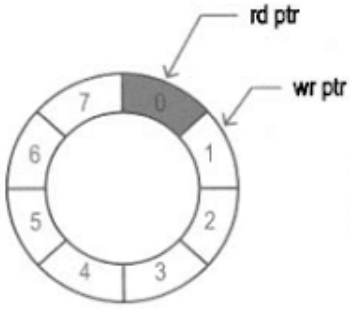


Figure 15. Circular FIFO With Two Pointers

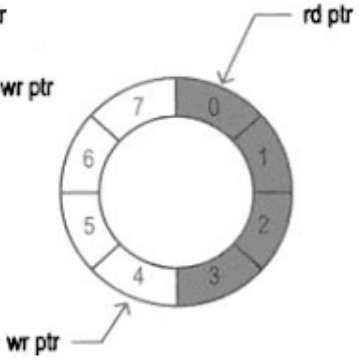
Minh họa cho Ring buffer



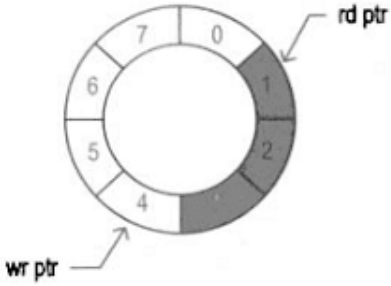
(a). initial (empty)



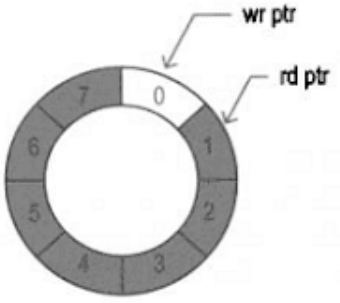
(b). after a write



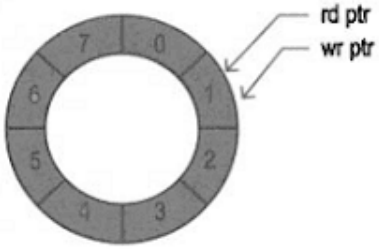
(c). 3 more writes



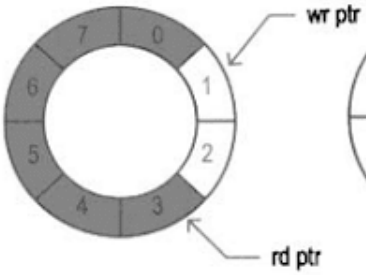
(d). after a read



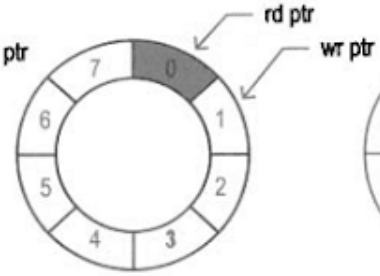
(e). 4 more writes



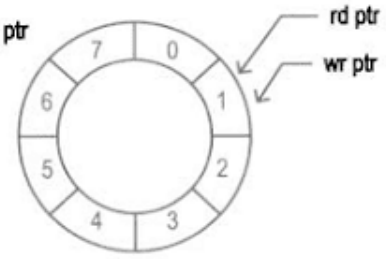
(f). 1 more write (full)



(g). 2 reads



(h). 5 more reads



(i). 1 more read (empty)

Hiện thực FIFO bằng buffer vòng: Như tên gọi của nó (array-base), bộ đệm này được thực hiện dựa trên một mảng. Kèm theo đó là 2 con trỏ Write và Read.

- Khi bắt đầu, con trỏ pWrite và pEnd đều cùng trỏ vào vị trí đầu tiên (index 0).
- Mỗi khi nhận lệnh ghi, con trỏ pWrite sẽ ghi data vào bộ đệm, sau đó sẽ tăng lên 1 đơn vị (pWrite++). Mỗi khi nhận lệnh đọc, dữ liệu được con trỏ pRead trỏ vào sẽ được đọc ra và con trỏ pRead sẽ tăng lên một (pRead++).
- Khi 1 con trỏ tới được cuối mảng, nó sẽ cuộn lại vị trí đầu tiên. Đó là lý do vì sao gọi đây là bộ đệm vòng.
- Nếu (pRead == pWrite), chứng tỏ bộ đệm đang trống (empty), không có gì để đọc.
- Nếu (pWrite + 1 == pRead), chứng tỏ bộ đệm đang đầy (full), không thể ghi thêm.

3. Applications - Ứng dụng

a. FIFO thường được sử dụng trong các mạch điện tử để đệm (buffering) và điều khiển luồng (flow) giữa phần cứng và phần mềm. Ở dạng phần cứng, FIFO chủ yếu bao gồm một tập hợp các con trỏ đọc và ghi, lưu trữ và điều khiển logic. Bộ lưu trữ (storage) có thể là bộ nhớ truy cập ngẫu nhiên tĩnh (SRAM), flip-flop, latches hoặc bất kỳ hình thức lưu trữ phù hợp nào khác. Đối với các FIFO có kích thước lớn hơn, SRAM hai cổng thường được sử dụng, trong đó một cổng được dành riêng cho việc ghi và cổng còn lại dành cho việc đọc. FIFO được biết đến đầu tiên được triển khai trong điện tử là của Peter Alfke vào năm 1969 tại Fairchild Semiconductor. Alfke sau đó là giám đốc của Xilinx.

b. FIFO cũng thường được ứng dụng trong lĩnh vực quản lý hàng hóa. Phương pháp FIFO là cách quản lý kho theo hình thức nhập trước xuất trước. Theo đó những hàng hóa (lô) được nhập vào kho đầu tiên sẽ là những hàng hóa đầu tiên đầu tiên được xuất ra khỏi kho đó. Những hàng hóa còn tồn lại sẽ là những hàng hóa mới được nhập gần đầy nhất. Với đặc thù này, FIFO sẽ được ứng dụng cho các loại hàng hóa dễ hư hỏng như thực phẩm hay các sản phẩm có vòng đời thấp như thời trang, hoặc các sản phẩm

có thể nhanh chóng trở nên lỗi thời như công nghệ. Với những mặt hàng trên, nếu doanh nghiệp của bạn lưu kho quá lâu, chắc chắn sẽ gây ra các khoản thua lỗ vì hàng hóa hết hạn hay trở nên lỗi mốt.

Về ưu điểm:

- Do hàng hóa sẽ được lưu kho trong thời gian ngắn, do đó, chi phí tồn kho trên mỗi sản phẩm sẽ giảm. Lẽ tất nhiên, điều này giúp tăng lợi nhuận cho doanh nghiệp.
- Với đặc thù luân chuyển hàng hóa liên tục, doanh nghiệp có thể kiểm soát chất lượng sản phẩm tốt hơn, tránh việc hàng hóa được sản xuất từ rất lâu nhưng thời điểm hiện nay mới xuất kho. Khi có lỗi phát sinh, khó có thể truy xuất nguồn gốc.

Về nhược điểm:

- Đây là nguyên tắc đòi hỏi cần chú trọng vào không gian lưu trữ kho lớn với nhiều thiết bị chuyên dụng. Bởi lẽ, các sản phẩm đã lưu lâu nhất trong kho cần được sắp xếp tại những vị trí dễ tiếp các công cụ xử lý.
- Doanh nghiệp bạn phải có hệ thống phần mềm theo dõi hàng tồn kho một cách kỹ càng để tránh bỏ sót hàng hóa sản xuất trước đó.

II. Design and implement a FIFO circuit using Verilog HDL

Dựa trên ý tưởng đã trình bày ở trên, nhóm đã từ đó nghiên cứu và đưa ra đoạn code hoàn chỉnh cho một FIFO cơ bản được hiện thực bằng cấu trúc ring buffer.

Source code: BTL.v

```
module BTL(
  input wire clk, // clock
  input wire rst, // reset
  input wire wr_en, // write enable
  input wire rd_en, // read enable
  input wire [7:0] data_in, // input data
  output reg [7:0] data_out, // output data
  output reg empty, // FIFO empty flag
  output reg full // FIFO full flag
    );
```

Các thành phần trong module

```
// Define the FIFO depth (FIFO buffer size)
parameter DEPTH = 8;

// Create a register to hold the FIFO data (buffer can store 8 data 8bit-words)
reg [7:0] mem [0 : DEPTH - 1];

// wr_ptr / rd_ptr
reg [2:0] rd_idx = 0;
reg [2:0] wr_idx = 0;
```

```
// Initialize FIFO empty and full flags
always @(*) begin
  empty = (wr_idx == rd_idx);
  full = ((wr_idx + 1) % DEPTH == rd_idx);
end
```

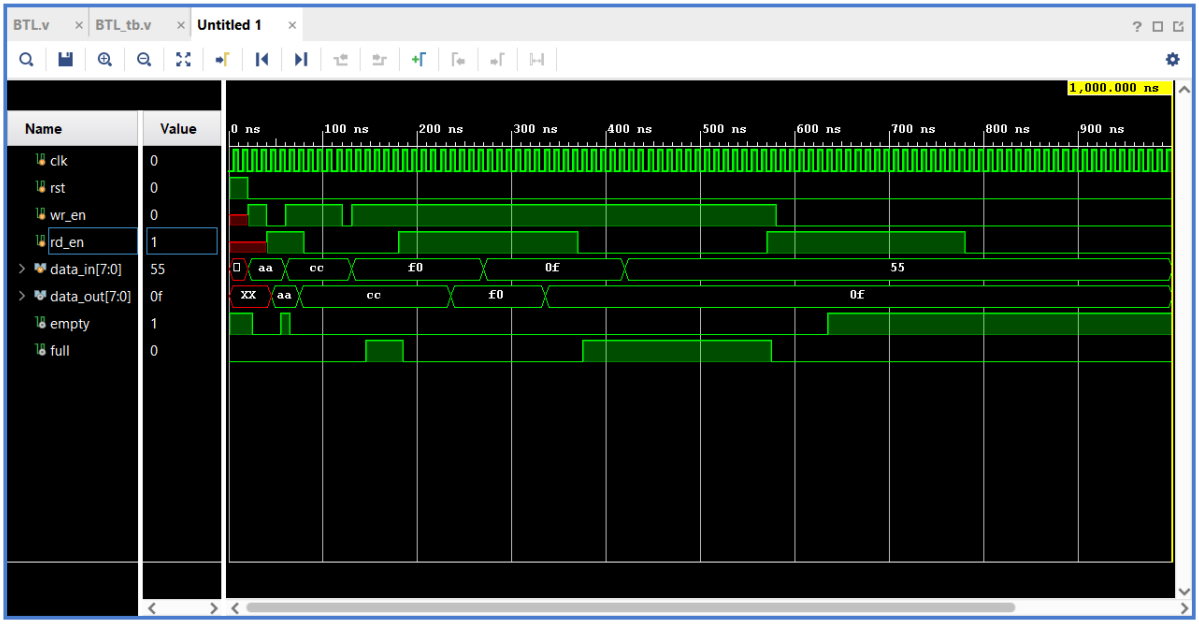
```
// Write operation
always @(posedge clk) begin
  if (rst) begin
    wr_idx <= 0;
  end
  else if (wr_en && !full) begin
    mem[wr_idx] <= data_in;
    wr_idx <= (wr_idx + 1) % DEPTH;
  end
end

// Read operation
always @(posedge clk) begin
  if (rst) begin
    rd_idx <= 0;
  end
  else if (rd_en && !empty) begin
    data_out <= mem[rd_idx];
    rd_idx <= (rd_idx + 1) % DEPTH;
  end
end
```

III. Test bench

Source code: BTL_tb.v

Kết quả:



IV. Demo on board

Source code: BTL_board.v

Code BTL_board.v dùng để hiện thực trên board thay đổi so với code BTL.v ở hai nội dung: (1) thay đổi kích thước dữ liệu data_in vào FIFO từ 8 bit thành 1 bit để đơn giản hơn trong việc hiện thực trên board để quan sát data_out (1 bit: bit 0 – đèn tắt, bit 1 – đèn sáng) và (2) clk được scale lại (thông qua temp_clk với chu kì dài hơn) để có thể dễ dàng quan sát output.

V. Tài liệu tham khảo

1. https://hgn37.wordpress.com/2017/05/20/bo-dem-fifo-vong/
2. https://en.wikipedia.org/wiki/Circular_buffer
3. https://www.ti.com/lit/an/scaa042a/scaa042a.pdf
4. https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)
5. https://en.wikipedia.org/wiki/Queueing_theory
6. https://vimach.net/threads/thiet-ke-bo-nho-dem-fifo-dung-verilog.124/
7. https://itgtechnology.vn/quan-ly-kho-fifo-lifo/
8. https://als.com.vn/phuong-phap-fifo-first-in-first-out-trong-xuat-nhap-hang