ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG
BỘ MÔN MÁY TÍNH – HỆ THỐNG NHÚNG

BÁO CÁO BÀI TẬP LỚN

Môn học: Thực hành Thiết kế SoC

Đề tài: Thiết kế bộ lọc FIR

Giảng viên hướng dẫn:..... Trần Tuấn Kiệt
Lớp:..... 20Nhung
Nhóm sinh viên thực hiện:.....
Lê Hoàng Việt Quốc..... 20200323

Thành phố Hồ Chí Minh, ngày 09 tháng 01 năm 2024

MỤC LỤC

ĐỀ BÀI:

Thiết kế bộ lọc FIR có n = 8 theo công thức bên dưới và tích hợp vào hệ thống SoC.

$$\boldsymbol{y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]}$$

$$\boldsymbol{= \sum_{i=0}^{N} b_i \cdot x[n-i]}$$

Trong đó:

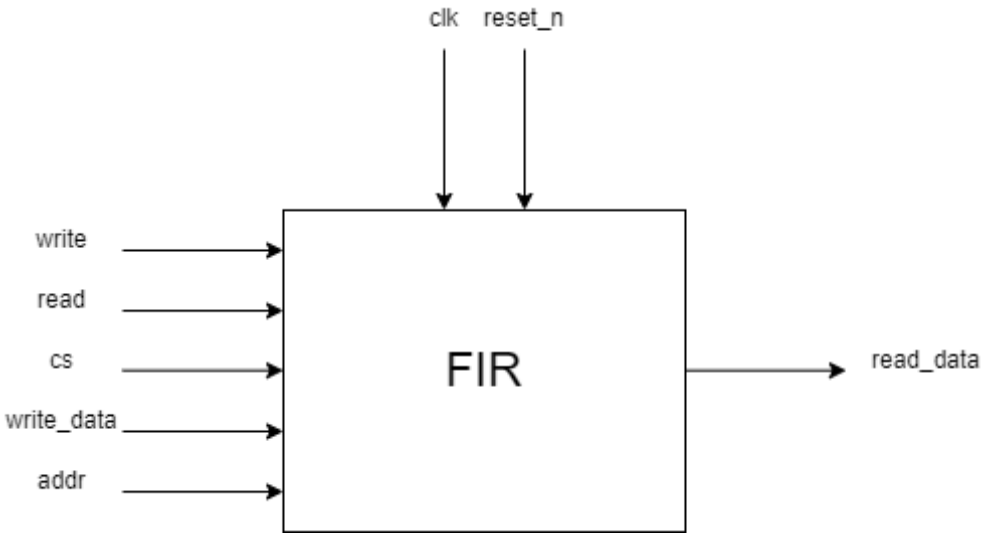
- $x[n]$ là tín hiệu vào.
- $y[n]$ là tín hiệu ra.
- N là bậc lọc; một bộ lọc bậc N có $N + 1$ thành phần ở phía bên phải của chương trình miêu tả nó.
- b_i là giá trị phản ứng xung tại khoảng thời gian thứ i với $0 \leq i \leq N$ của một bộ lọc FIR bậc N. Nếu bộ lọc biểu diễn dưới dạng trực tiếp thì b_i thực sự là hệ số của bộ lọc.

Các giá trị $x[i], y[i],$ và $b[i]$ được ghi/đọc xuống bởi CPU NIOS.

Sản phẩm nộp là file báo cáo bao gồm: chi tiết thiết kế phần cứng, tích hợp hệ thống SoC, flow code C và kết quả mô phỏng (mô phỏng ở mức system) + giải thích.

Thiết kế phần cứng:

Module FIR:



Mô tả tín hiệu của module:

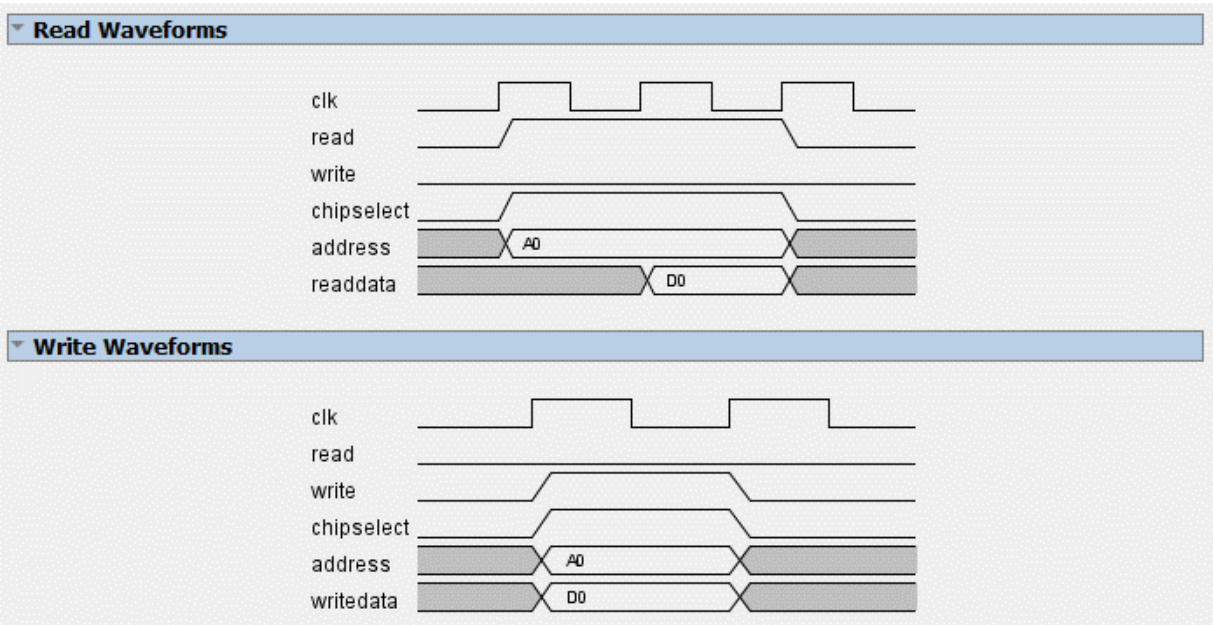
STT	Tên tín hiệu	Độ rộng (bits)	Hướng	Mô tả
1	clk	1	input	Cấp xung cho module hoạt động
2	reset_n	1	input	Cấp tín hiệu reset mức thấp cho module.
3	write	1	input	Nếu write = 0, module được cho phép dữ liệu vào. Ngược lại, write = 1, module bỏ qua tín hiệu đưa vào.
4	read	1	input	Nếu read = 0, module cho phép đọc dữ liệu ra ngoài. Ngược lại, read = 1, module không cho phép đọc dữ liệu ra ngoài.
5	cs	1	input	Nếu cs = 0, module được phép hoạt động. Ngược lại, cs = 1, module không được hoạt động

6	addr	2	input	Địa chỉ cần cho việc đọc hoặc ghi.
7	write_data	32	input	Dữ liệu ghi vào.
8	read_data	32	output	Dữ liệu đọc ra ngoài.

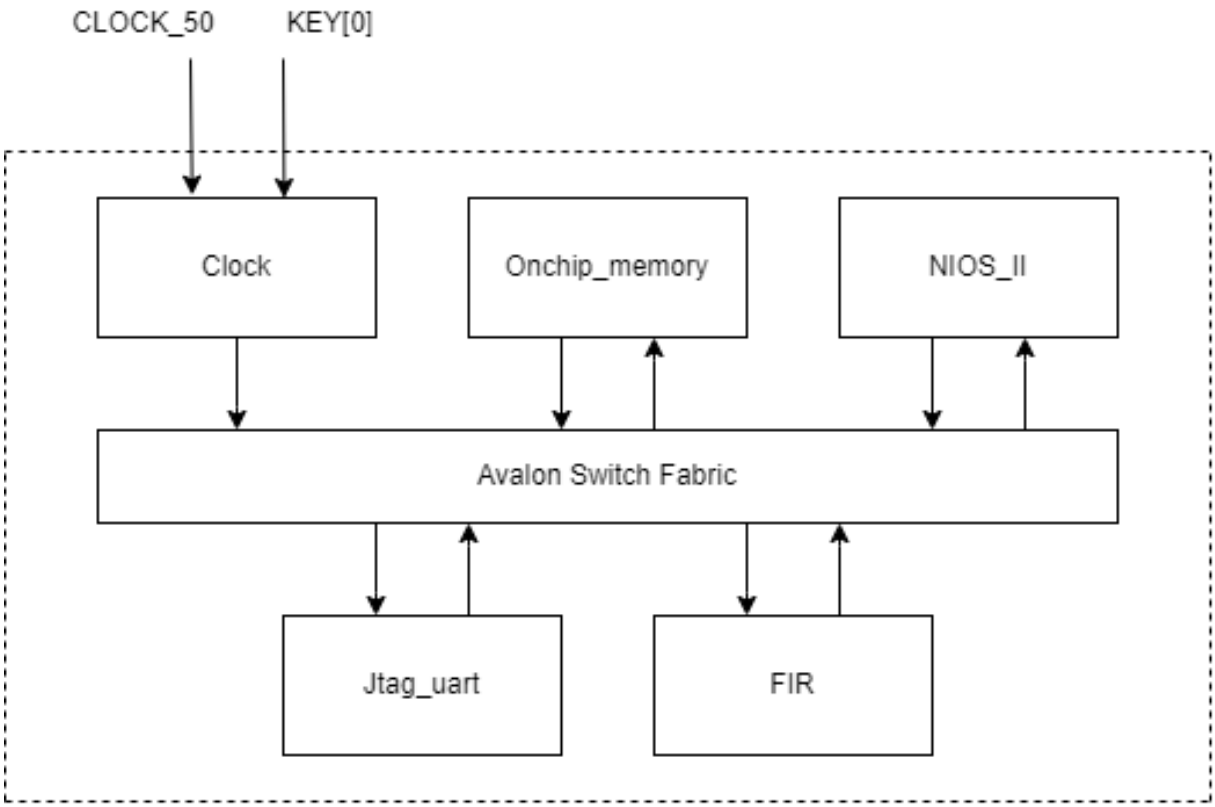
Các thanh ghi trong module:

Offset	Tên thanh ghi	Đọc/Ghi	Mô tả	
			[31:0]	[0]
0	control	Đọc/Ghi		Giá trị control
1	b	Đọc/Ghi	Giá trị b	
2	x	Đọc/Ghi	Giá trị x	
3	data_out	Đọc	Giá trị data_out	

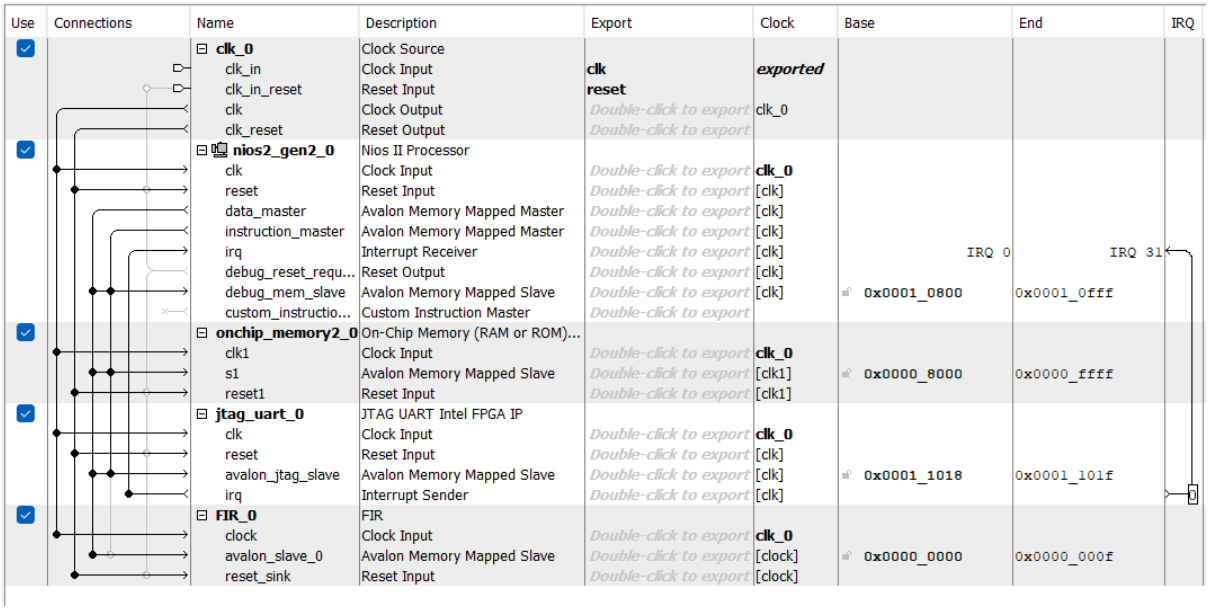
Dạng sóng đọc ghi:



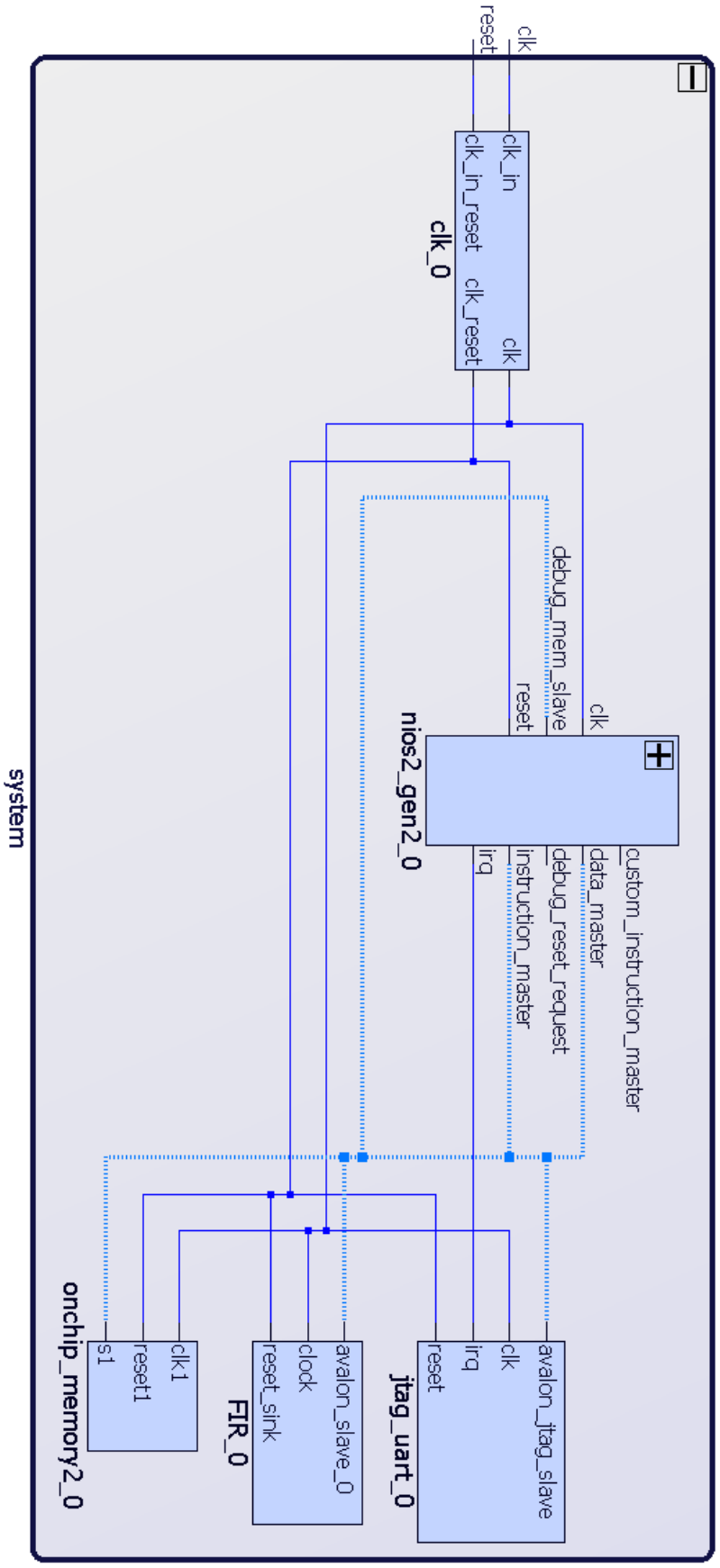
Tổng quan hệ thống:



Xây dựng phần cứng trên Platform design, thêm module FIR vào hệ thống, cấu hình tín hiệu clock, reset, kết nối tín hiệu. Hệ thống hoàn chỉnh như sau:



System schematic



Tích hợp hệ thống SoC:

Code verilog mô tả module FIR_slave:

```
module Fir_slave
#(
    parameter DATA_WIDTH = 32
)(
    input                           clk,
    input                           reset_n,
    input                           write,
    input                           read,
    input                           cs,
    input       [DATA_WIDTH - 1 : 0]    write_data,
    input       [1:0]                   addr,

    output  reg [DATA_WIDTH - 1 : 0]    read_data
);
    reg                         control;
    reg     [DATA_WIDTH - 1: 0]     b;
    reg     [DATA_WIDTH - 1: 0]     x;
    wire    [DATA_WIDTH - 1 : 0]    data_out;
        always @(posedge clk, negedge reset_n) begin
        if(~reset_n) begin
            read_data <= 32'd0;
        end else begin
```

```
            if(cs & write) begin
                case (addr)
                    2'd0: b <= write_data;
                    2'd1: x <= write_data;
                    2'd2: control <= write_data[0];
                endcase
            end
            if(cs & read) begin
                case (addr)
                    2'd0: read_data <= 32'd15;
                    2'd1: read_data <= data_out;
                endcase
            end
        end
    end
Fir f(
    .clk(clk),
    .reset_n(reset_n),
    .control(control),
    .b(b),
    .x(x),
    .data_out(data_out)
);
endmodule
```

Module Fir_slave là một thành phần của hệ thống.

Trong đó:

Tham số (Parameter):

– DATA_WIDTH: Đây là một tham số có giá trị mặc định là 32, chỉ định độ rộng của dữ liệu (data width) trong module.

Cổng vào (Input Ports):

– clk: clock.

– reset_n: tín hiệu reset.

– write: tín hiệu ghi.

– read: tín hiệu đọc.

– cs: tín hiệu chọn chip (chip_select).

– write_data: dữ liệu cần ghi vào.

– addr: địa chỉ cần truy cập.

Cổng ra (Output Ports):

– read_data: dữ liệu được đọc từ FIR slave.

Biến và dây kết nối:

– control: biến lưu trạng thái điều khiển.

– enable: biến lưu trạng thái cho phép (enable).

– b: biến lưu dữ liệu b.

– x: biến lưu dữ liệu x.

– data_out: dây kết nối với FIR master để truyền dữ liệu ra khỏi FIR slave.

Khối **always** xảy ra khi ở sườn lên xung clock hoặc ở sườn xuống reset_n.

– Nếu nút reset_n được nhấn read_data được reset về 0.

– Nếu nút reset_n không được nhấn:

+ Nếu trạng thái chip_select và write được chọn, xét giá trị của address:

Nếu address = 2’d0 thì gán giá trị của write_data vào biến b.

Nếu addres = 2’d1 thì gán giá trị của write_data vào biến x.

Nếu addres = 2’d2 thì gán giá trị của write_data[0] vào biến control.

+ Nếu trạng thái chip_select và read được chọn, xét giá trị của address:

Nếu address = 2’d0 thì gán giá trị read_data = 15 (đọc giá trị 16 lần bằng kích thước của 2 mảng b, x).

Nếu addres = 2’d1 thì gán giá trị của read_data = giá trị của data_out.

Kết nối với FIR master (Fir):

- Module này kết nối với một Module khác có tên là Fir thông qua các cổng và dây kết nối. Các cổng này bao gồm clk, reset_n, control, enable, b, x, và data_out.

Code verilog mô tả module FIR:

```
module Fir#(
        parameter    DATA_WIDTH = 32
)(
        input                           clk,
        input                           reset_n,

        input                           control,
        input  [DATA_WIDTH - 1: 0]   b,
        input  [DATA_WIDTH - 1: 0]   x,
        input                           enable,
        output reg    [DATA_WIDTH - 1: 0]   data_out
);
```

```
    reg             [DATA_WIDTH - 1 : 0]  b_array [8 : 0];
    reg             [DATA_WIDTH - 1 : 0]  x_array [8 : 0];
    reg             [DATA_WIDTH - 1 : 0]  h [8 : 0];
    reg             [DATA_WIDTH - 1 : 0]  s;
    integer i;


    always @(posedge clk) begin
            if ((control == 0) & enable) begin
        for (i = 0; i < 8; i = i + 1) begin
           b_array[i] <= b_array[i + 1];
        end
        b_array[8] <= b;
            end

    end

    always @(posedge clk) begin
            if ((control == 1) & enable) begin
     for (i = 8; i > 0; i = i - 1) begin
        x_array[i] <= x_array[i - 1];
     end
     x_array[0] <= x;
   end
end
```

```
    always @* begin
        for (i = 0; i < 9; i = i + 1) begin
            h[i] = x_array[i] * b_array[i];
        end
        s = h[0] + h[1] + h[2] + h[3] + h[4] + h[5] + h[6] + h[7] + h[8];
    end

    always @(posedge clk) begin
        data_out <= s;
    end
endmodule
```

Đây là một module được thiết kế để thực hiện chức năng của một bộ lọc hồi quy có n=8 (FIR filter).

Trong đó:

Tham số (Parameter):

– DATA_WIDTH: Đây là một tham số có giá trị mặc định là 32, chỉ định độ rộng của dữ liệu (data width) trong mô-đun.

Cổng vào (Input Ports):

– clk: clock.

– reset_n: tín hiệu reset.

– control: tín hiệu điều khiển, quyết định xem module sẽ thực hiện đọc mảng b hay x.

– b: dữ liệu đầu vào cho bộ lọc FIR.

– x: dữ liệu đầu vào khác cho bộ lọc FIR.

– enable: tín hiệu cho phép, quyết định xem module có thực hiện hoạt động hay không.

Cổng ra (Output Ports):

– data_out: dữ liệu đầu ra sau khi đi qua bộ lọc FIR.

Biến và mảng:

– b_array, x_array: mảng 1 chiều để lưu trữ các giá trị b và x tương ứng.

– h: mảng 1 chiều để lưu trữ kết quả tính kết quả từ x_array và b_array.

– s: biến để lưu tổng của các giá trị trong mảng h.

– i: biến kiểu integer sử dụng trong vòng lặp.

Quá trình lọc FIR:

– Có hai khối always @(posedge clk): Mỗi khối này kích hoạt bởi mỗi cạnh lên của xung clock.

– Khối đầu tiên thực hiện việc dịch chuyển giá trị trong b_array để tạo một bộ lưu trữ truyền dịch (shift register) cho dữ liệu b.

– Khối thứ hai thực hiện việc dịch chuyển giá trị trong x_array để tạo một bộ lưu trữ truyền dịch (shift register) cho dữ liệu x.

Quá trình tính toán FIR:

– Có một khối always @*. Nó thực hiện tính toán các giá trị h bằng cách nhân tương ứng từng cặp giá trị trong x_array và b_array.

– Sau đó, giá trị của s được tính bằng cách cộng tổng các giá trị trong h.

Gán giá trị đầu ra:

– Có một khối always @(posedge clk) cuối cùng để gán giá trị của s vào data_out như là giá trị đầu ra của module.

Flow code C:

```c
#include "system.h"
#include "sys/alt_stdio.h"

int main()
{
  int b_arr [9] = {1, 2, 1, 2, 3, 4, 1, 3, 4};
  int x_arr [9] = {1, 2, 3, 2, 5, 4, 1, 2, 2};
  int s = 0;
  int* c_ptr = (int*)FIR_0_BASE;
  *(c_ptr + 2) = 0;
  for (int i = 0; i < 9; i++) {

      *(c_ptr + 0) = b_arr[i];
  }
  *(c_ptr + 2) = 1;

  for (int i = 0; i < 9; i++) {

      *(c_ptr + 1) = x_arr[i];
  }

  s = *(c_ptr + 1);
  printf("done");
  return 0;
}
```

Trong đó:

– Khởi tạo mảng b_arr và x_arr: Tạo hai mảng b_arr và x_arr để chứa các giá trị của b và x tương ứng.

– Khởi tạo biến s: Biến này được sử dụng để lưu giữ giá trị đầu ra từ FIR.

– Khởi tạo con trỏ c_ptr: Tạo một con trỏ c_ptr và gán địa chỉ của FIR_0_BASE cho nó. Con trỏ này được sử dụng để giao tiếp với bộ lọc FIR.

– Khi c_ptr + 2 bằng 0 thực hiện vòng lặp for đọc vào mảng b_arr lưu tại con trỏ c_ptr.

– Khi c_ptr + 2 bằng 1 thực hiện vòng lặp for đọc vào mảng x_arr lưu tại con trỏ c_ptr + 1.

– Đọc giá trị đầu ra từ FIR thông qua con trỏ c_ptr và gán giá trị vào biến s.

Kết quả mô phỏng:

Thực hiện lọc fir với 2 mảng:

b_arr [9] = {1, 2, 1, 2, 3, 4, 1, 3, 4};

x_arr [9] = {1, 2, 3, 2, 5, 4, 1, 2, 2};

Theo công thức:

$$\boldsymbol{y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]}$$

$$\boldsymbol{= 1 \times 2 + 2 \times 2 + 1 \times 1 + 2 \times 4 + 3 \times 5 + 4 \times 2 + 1 \times 3 + 3 \times 2 + 4 \times 1}$$

$$\boldsymbol{= 51}$$

Thu được kết quả mô phỏng: s = 51.