ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

ꕥꕥꕥ

Trường Đại Học Công Nghệ Thông Tin

BÁO CÁO ĐỒ ÁN 1

※※※

THIẾT KẾ IP NHÂN 2 SỐ CHẤM ĐỘNG THEO CHUẨN IEEE- 754

GVHD: Trương Văn Cương

NHÓM SVTH:

Đào Công Hậu - 22520408

Phạm Ngọc Hải- 22520389

TP.Hồ Chí Minh, ngày 05 tháng 07 năm 2025

Contents

LỜI NÓI ĐẦU

Trong thời đại cách mạng công nghiệp 4.0, các ứng dụng trí tuệ nhân tạo, xử lý tín hiệu số (DSP), đồ họa máy tính và tính toán khoa học phức tạp ngày càng phát triển mạnh mẽ. Các hệ thống này yêu cầu các phép toán số thực dấu phẩy động với độ chính xác cao và hiệu năng tốt. Chuẩn IEEE-754 single-precision (32 bit) đã trở thành tiêu chuẩn phổ biến trong biểu diễn và tính toán số thực dấu phẩy động trên các thiết bị phần cứng như CPU, GPU, FPGA và bộ xử lý nhúng. Việc thiết kế phần cứng thực hiện phép toán nhân dấu phẩy động đúng chuẩn, hiệu quả về mặt tài nguyên và tốc độ là bài toán quan trọng trong thiết kế vi mạch số hiện nay.

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1. Mục tiêu

Nắm vững kiến thức chuẩn IEEE-754 về định dạng, quy trình nhân, chuẩn hoá và làm tròn. Thiết kế một mô-đun IP thực thi phép nhân dấu phẩy động đơn chính xác, đáp ứng đầy đủ các bước: nhân mantissa, cộng số mũ, chuẩn hoá và round-to-nearest-even. Tối ưu hóa kiến trúc theo phương pháp iterative (vòng lặp nội bộ) để tiết kiệm diện tích phần cứng so với pipelining sâu, đồng thời đạt được hiệu năng đủ dùng trên FPGA. Thực hiện mô phỏng và kiểm thử toàn diện, so sánh kết quả với thư viện tham chiếu, đánh giá chu kỳ xử lý và tiêu hao tài nguyên

1.2. Phạm vi nghiên cứu:

- Thiết kế cho chuẩn single-precision (32 bit) theo IEEE-754.
- Hỗ trợ quy tắc làm tròn round-to-nearest-even.
- Chưa xử lý các trường hợp đặc biệt như NaN, vô cực và số denormal trong phiên bản hiện tại.
- Sử dụng thiết kế iterative (vòng lặp nội bộ) thay vì pipeline sâu.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Định dạng số chấm động (Floating-point)

Chuẩn IEEE-754 single-precision (32 bit) mô tả cách biểu diễn số thực dấu phẩy động với cấu trúc ba trường:

Sign	**Exponent**	**Mantisa**
1 bit	**8 bit**	**23 bit**

Sign:1 bit; 0: biểu diễn số dương, 1: biểu diễn số âm

Exponent: 8 bit, biểu diễn mũ có bias là 127

Mantissa: 23 bit, phần trị sau dấu ".", với bit ẩn (implicit leading 1) cho số chuẩn hoá

Biểu diễn số thực dấu chấm động:

$$(-1)^S * (1 + Mantisa) * 2^{(Exponent-Bias)}$$

2.2. Các bước trong phép nhân số dấu chấm động

2.2.1. Xác định dấu của kết quả (Sign)

Dấu của kết quả phép nhân hai số dấu chấm động được xác định bằng phép XOR của hai bit dấu:

$$S_{out} = S_A \oplus S_B$$

2.2.2. Cộng số mũ (Exponent)

Tổng của số mũ mới được tính theo công thức:

$$E_{normaliz} = E_A + E_B - bias$$

- Trong đó, E_A và E_B là các trường exponent của hai số.
- Phép trừ bias(127) nhằm loại bỏ bias đã được cộng trước đó cho mỗi số.

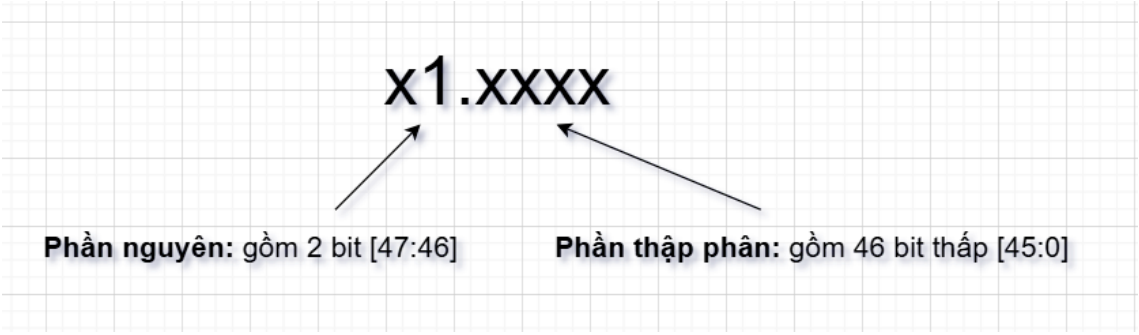
2.2.3. Nhân phần trị (Mantissa)

Hai mantissa được mở rộng thêm bit ẩn thành 24 bit mỗi số, sau đó nhân với nhau:

$$1.M_A \times 1.M_B$$

Kết quả có độ rộng 48 bit, gồm:

- Phần nguyên: (Bit 47-46)
- Phần phân số: (Bits 45–0).



2.3. Chuẩn hoá và làm tròn

2.3.1. Chuẩn hoá

Mục đích của chuẩn hoá là đưa mantissa về dạng chuẩn (1.xxxx), kết quả phép nhân ở dạng "x1.xxxx...":

- Nếu bit 47 = 0: *dạng "01.xxxx"* → đã chuẩn hoá, giữ nguyên số mũ và không cần dịch.
- Nếu bit 47 = 1 *dạng "10.xxxx"* hoặc "11,xxxx" → chưa chuẩn hoá, dịch phải 1 bit đồng thời tăng số mũ thêm 1 : $E_{normalize} = E + 1$.

Chuẩn hoá đảm bảo phần mantissa có dạng chuẩn hoá và exponent tương ứng.

2.3.2. Làm tròn

Để đảm bảo độ chính xác và tuân thủ chuẩn IEEE-754, kết quả mantissa sau chuẩn hoá được làm tròn dựa trên ba bit phụ trợ:

- Sử dụng **guard** bit (G): bit ngay sau bit cuối cùng của mantissa (bit 23)
- **round** bit (R): bit tiếp theo sau **guard**
- **sticky** bit (S): OR của tất cả các bit sau **round**

Quy tắc làm tròn:

Nếu G = 0 → không làm tròn, giữ nguyên mantissa.

Nếu G = 1 và làm tròn lên nếu ít nhất một trong các bit R, S hoặc LSB của mantissa hiện tại là **1**

Kết quả:

Result [31:0] ={$S_{normaliz}$, E [7:0], Mantissa[22:0]}

CHƯƠNG 3: KIẾN TRÚC HỆ THỐNG

3.1. Tổng quan kiến trúc

Hệ thống thực hiện phép nhân hai số dấu phẩy động 32-bit theo chuẩn IEEE-754 được thiết kế theo mô hình tách biệt rõ ràng giữa datapath và control unit, đảm bảo tính mô đun, dễ kiểm soát và mở rộng.

Cấu trúc tổng quát gồm 2 khối chính:

Datapath: Thực hiện các phép toán: nhân mantissa, cộng số mũ, chuẩn hóa, làm tròn, kết hợp kết quả.

Control Unit: Sử dụng FSM để điều phối các bước xử lý theo đúng trình tự.**3.2.**

3.2. Cấu trúc chi tiết khối Datapath:

Datapath được cấu tạo bởi các thành phần chính sau:

1. **Thanh ghi đầu vào** (Registers A và B) và **MUX2to1**: Lưu trữ các toán hạng đầu vào 32 bit (gồm dấu, exponent và mantissa) và lưu giá trị của khối cộng exponent và khối nhân mantissa. Mux chọn giá trị đầu vào phù hợp với từng chu kì.
2. **Khối XOR dấu**: Thực hiện phép XOR giữa bit dấu của hai số đầu vào để xác định dấu kết quả.
3. **Khối cộng exponent** (Adder): Cộng hai số mũ 8 bit của hai toán hạng và trừ đi bias 127.
4. **Khối nhân mantissa** (Multiplier): Nhân hai mantissa 24 bit (bao gồm bit ẩn) với nhau, tạo ra kết quả 48 bit.
5. **Khối dịch phải** (Shift-Right): Dịch kết quả mantissa nếu bit cao nhất của tích bằng 1 (chuẩn hoá mantissa thành dạng 1.xxxx).

6. **Khối tăng số mũ** (Incrementer): Tăng số mũ lên 1 khi mantissa đã được dịch phải 1 bit.

7. **Khối làm tròn** (Rounding Unit): Thực hiện làm tròn mantissa theo quy tắc round-to-nearest-even dựa trên các bit guard, round và sticky; chứa 2 thanh ghi lưu các giá trị output.

+ **Thanh ghi kết quả** (Output Register): Ghép dấu, số mũ và mantissa sau làm tròn thành số 32 bit kết quả cuối cùng, một thanh ghi giữ tín hiệu cờ báo tràn overflow_flag.

3.3. Khối điều khiển (Control Unit)

Khối điều khiển được mô hình hóa bởi FSM (Finite State Machine), điều phối quá trình xử lý theo trình tự các bước:

1. **RESET:** Trạng thái reset, chờ tín hiệu start.
2. **IDLE:** Nhận giá trị input, chọn ngõ vào thanh ghi và thực hiện các khối cộng, nhân, xor.
3. **CHECK_SHIFT_INCREMENT:** Chọn ngõ vào thanh ghi, kiểm tra bit MSB của kết quả phép nhân mantissa để chọn tín hiệu cho bộ Increment và bộ Shift.
4. **ROUNDING:** xử lý các tín hiệu trong khối rounding thông qua tín hiệu đầu vào bit_check_overflow.

Tín hiệu điều khiển chính:

start: tín hiệu bắt đầu.

inc_shift_en: bật/tắt dịch phải mantissa và tăng số mũ.

mux_en_rounding: chọn dữ liệu đầu ra cho khối làm tròn, overflow hoặc không overflow.

enable_rounding: tín hiệu cập nhật ngõ ra cho chu kì cuối.

enable_reg và mux_en_reg: điều khiển ghi dữ liệu vào thanh ghi.

no_start: tín hiệu phụ reset ngõ ra khi chưa có tín hiện bắt đầu (start).

3.4. Luồng dữ liệu điều khiển

Quá trình nhân được thực hiện tuần tự theo luồng:

1. Nạp hai toán hạng A, B vào thanh ghi.
2. Tính dấu kết quả bằng XOR dấu của A và B.

3. Cộng exponent hai số, trừ bias.

4. Nhân mantissa mở rộng, tạo kết quả 48 bit.

5. FSM kiểm tra bit 47 (cao nhất) của mantissa:

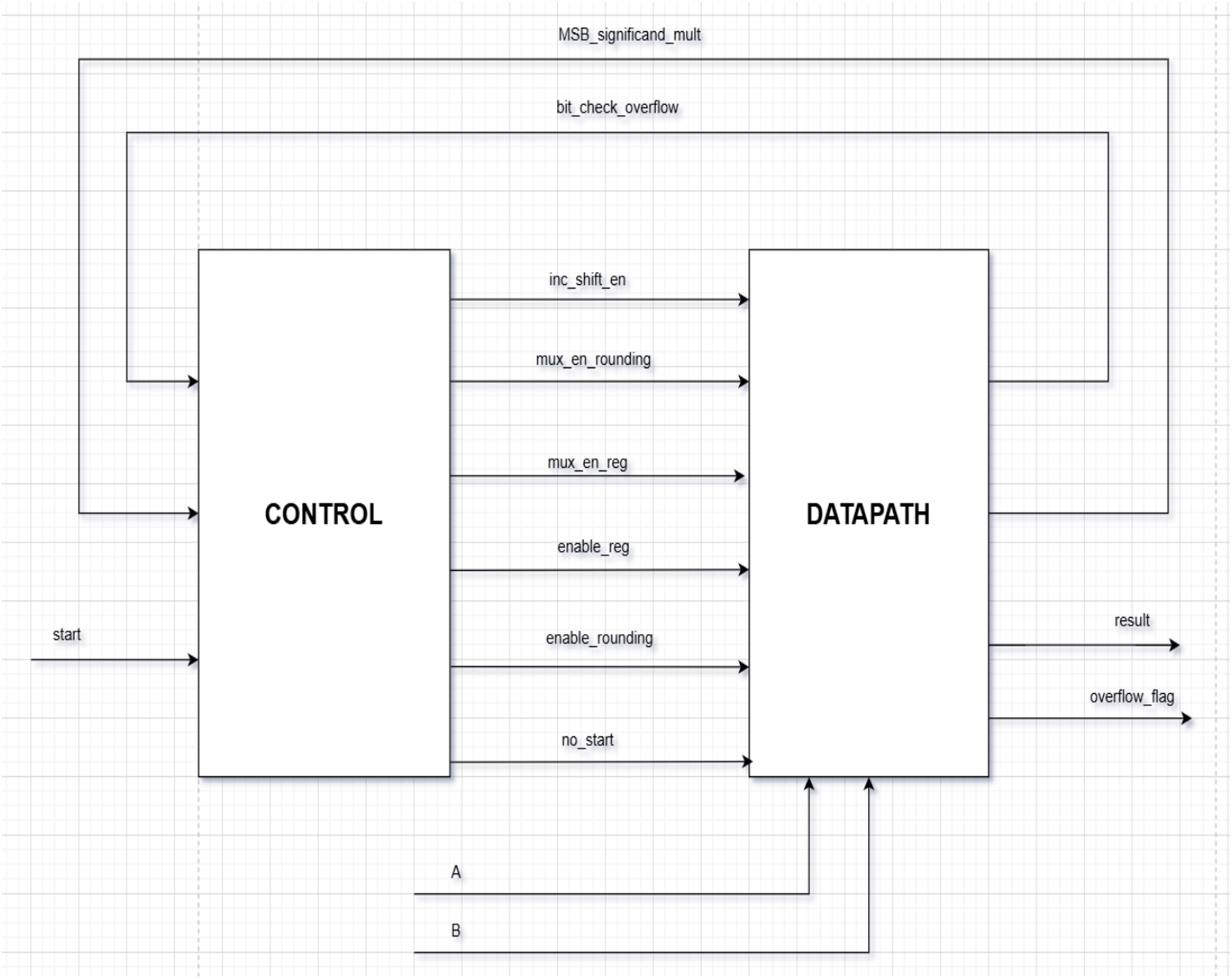
6. Nếu bit 47 = 1, bật inc_shift_en dịch phải mantissa và tăng exponent và ngược lại bit [47] = 0 => in_shift_en = 0, không làm gì.

7. Kiểm tra bit báo tràn.

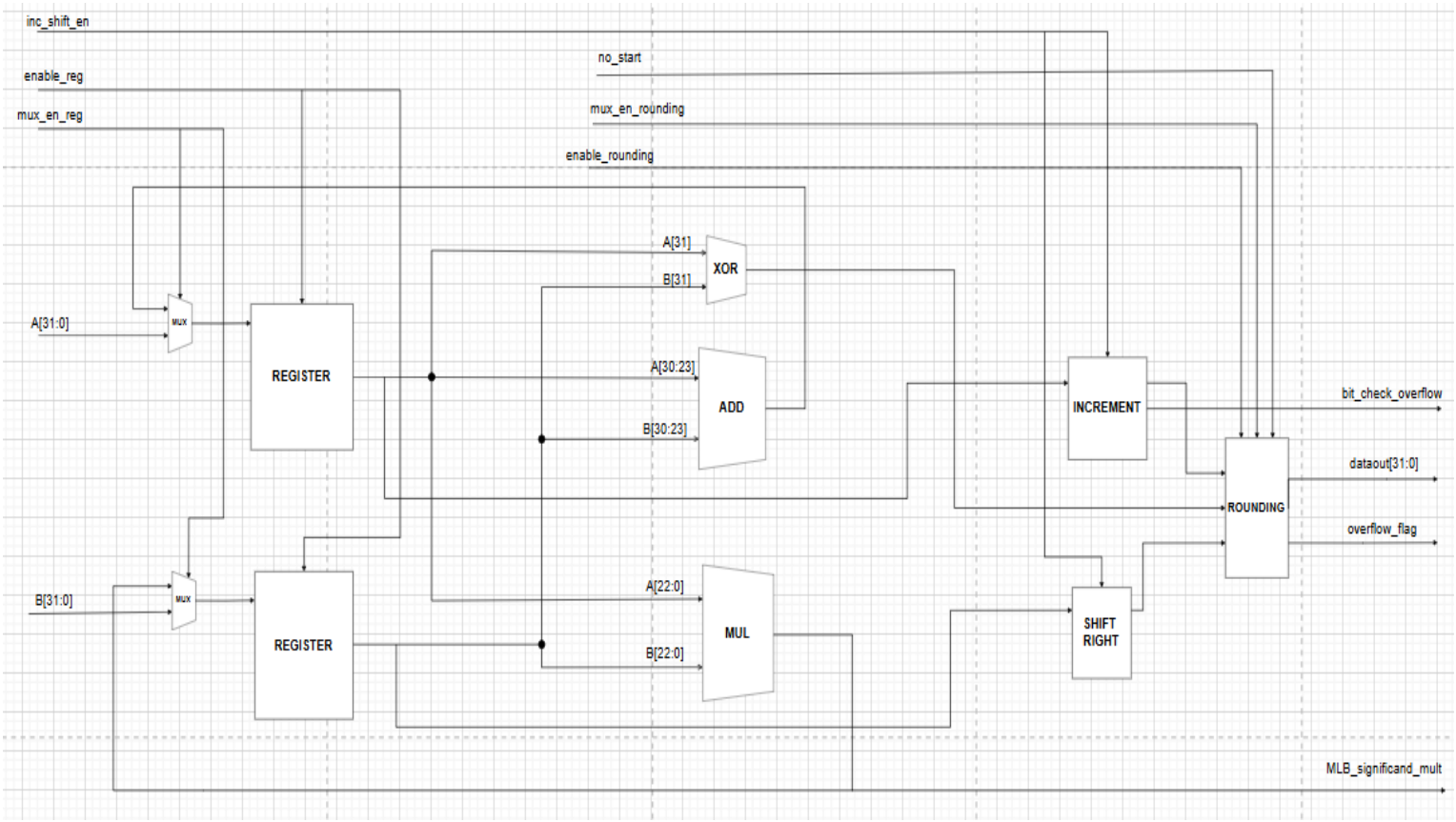
8. Khi mantissa ở dạng chuẩn, chuyển sang làm tròn theo round-to-nearest-even.

9. Ghép dấu, số mũ và phần trị thành kết quả và xuất ra dataout[31:0].

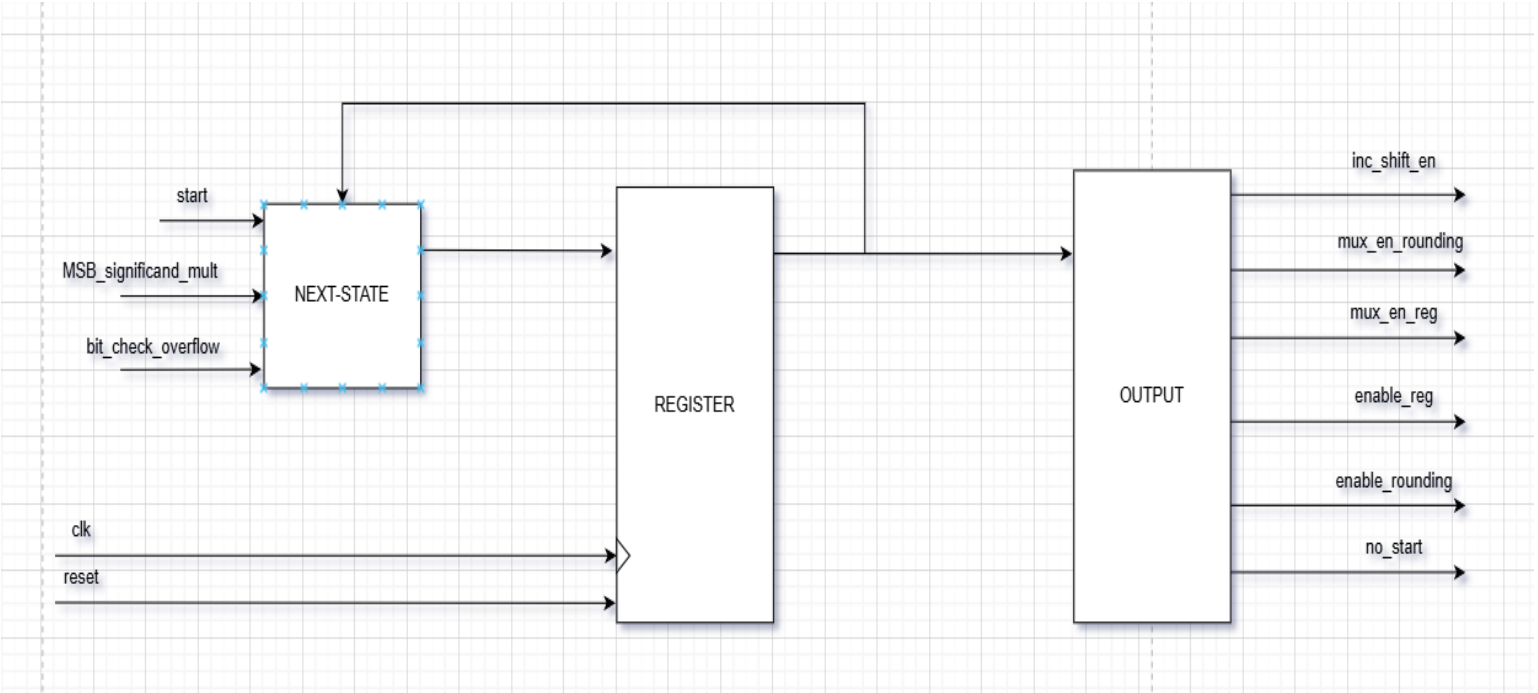
3.5. Sơ đồ khối tổng quát



Datapath:



Control Unit:

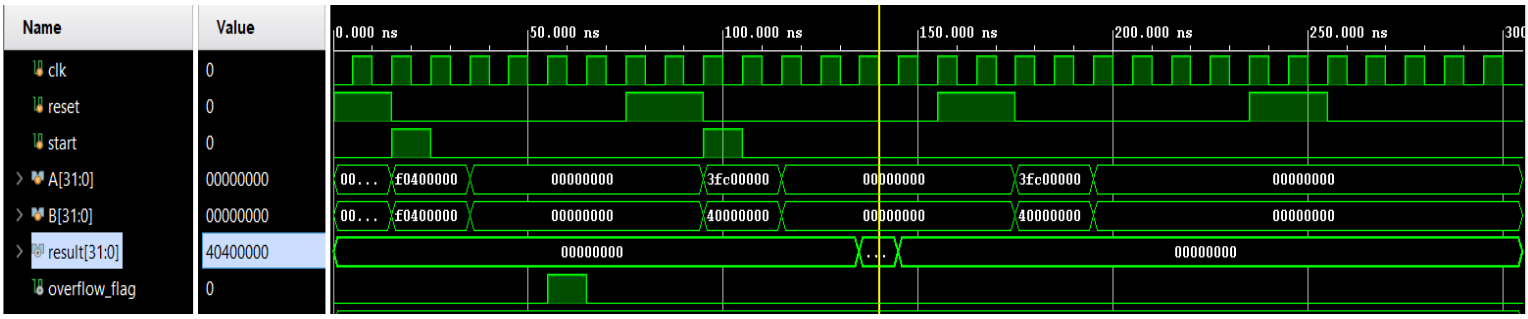


CHƯƠNG 4: HIỆN THỰC THIẾT KẾ VÀ TESTBENCH

4.1. Hiện thực thiết kế:

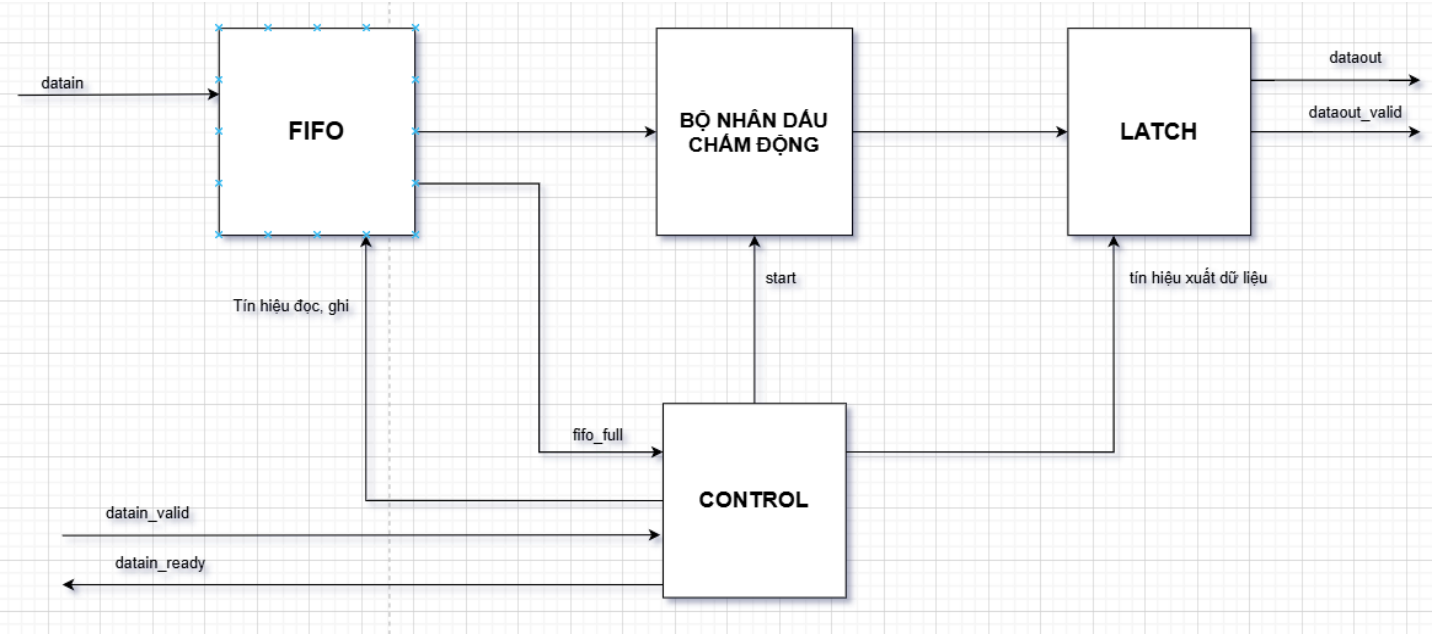
***Source code:**

***Mô phỏng:**



4.2. Kết nối các tín hiệu valid thành top-module:

***Tổng quan kết nối:**



***Mô tả hoạt động:**

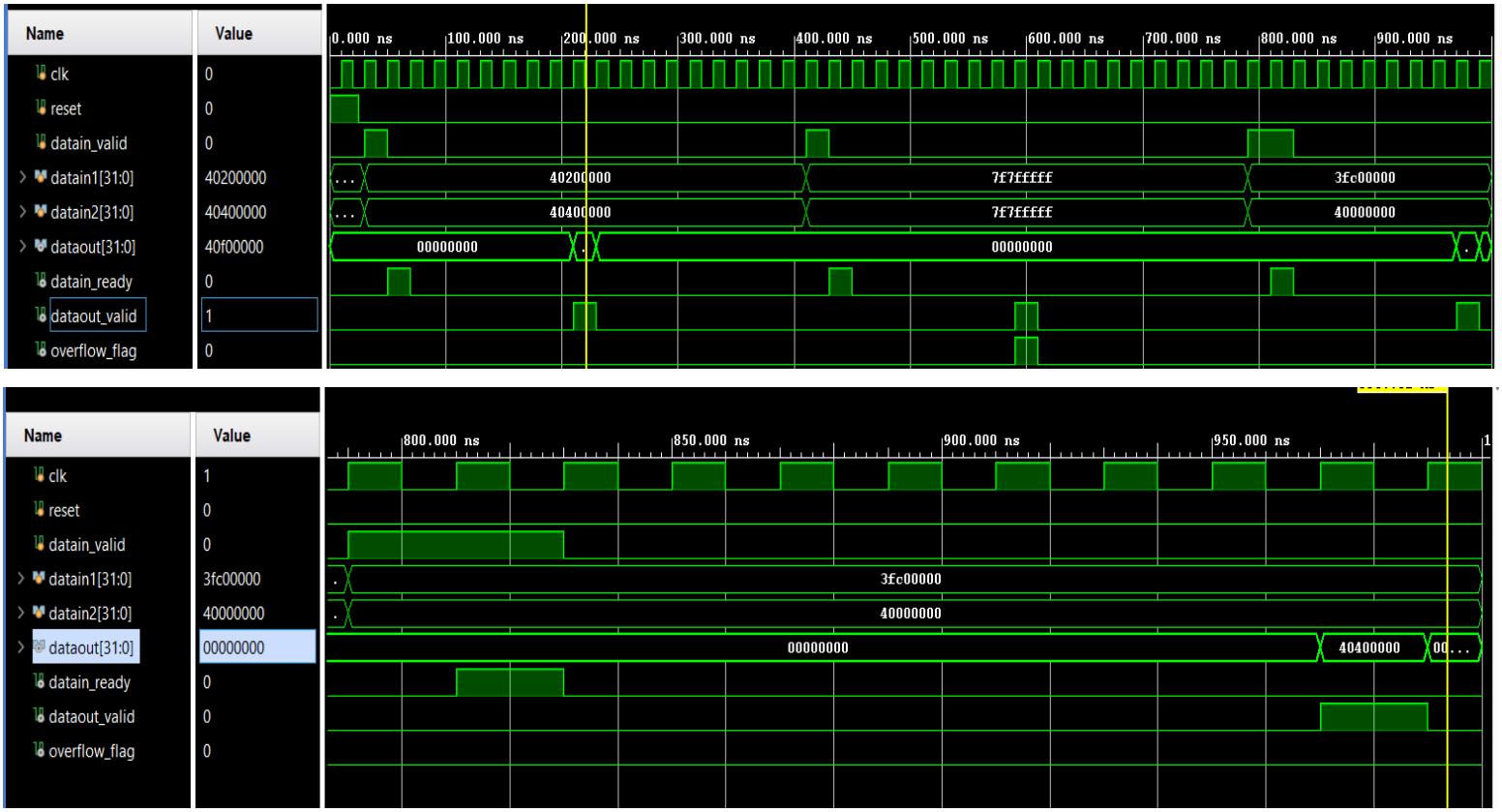
- Các thành phần: FIFO (do nhà phát hành cung cấp), Bộ nhân tự thiết kế, Latch để chốt dữ liệu đầu ra.

- Control:

- **IDLE**: Trạng thái chờ, hệ thống đứng yên cho đến khi nhận được tín hiệu `datain_valid = 1` và kiểm tra FIFO full hay không.
- **READY_ONE_CYCLE**: Hệ thống phát tín hiệu `datain_ready = 1` trong đúng một chu kỳ để báo hiệu sẵn sàng nhận dữ liệu đầu vào.
- **WRITE_FIFO**: Ghi cặp dữ liệu đầu vào vào FIFO bằng cách kích hoạt tín hiệu `fifo_write_en`.
- **READ_FIFO**: Đọc một cặp dữ liệu từ FIFO để chuẩn bị cho xử lý, kích hoạt `fifo_read_en`.
- **WAIT_READ_DELAY**: Giá trị từ FIFO đọc ra, đồng thời phát tín hiệu `start = 1` để khởi động khối nhân dấu chấm động. (bộ nhân sẽ hoạt động sau tín hiệu start)
- **WAIT_MUL_RESULT1**: Chờ kết quả xử lý nhân trong chu kỳ đầu tiên (stage 1).
- **WAIT_MUL_RESULT2**: Chờ kết quả xử lý nhân trong chu kỳ thứ hai (stage 2).
- **WAIT_MUL_RESULT3**: Chờ kết quả xử lý nhân trong chu kỳ thứ ba (stage 3).

- **OUT_VALID**: Hoàn thành bộ nhân, kết quả được đưa qua latch, phát tín hiệu `out_valid = 1` trong một chu kỳ để báo hiệu dữ liệu đầu ra đã sẵn sàng, sau đó quay trở lại trạng thái `IDLE`.

***Mô phỏng:**



Đánh giá kết quả đạt được

***Ưu điểm:** Đã hoàn thành yêu cầu, xử lý kết quả và tín hiệu.

***Hạn chế**: thiết kế chưa tối ưu hoàn toàn, chưa pipeline.

TÀI LIỆU THAM KHẢO

- Giáo trình ĐHQGTP.HCM

- Và một số nguồn khác…..

KẾT LUẬN