REPORT

Thiết kế và thực hiện khối tính ảnh tích phân 2-D

Ver 1.0

07/06/2025

|  |  |  |  |
| --- | --- | --- | --- |
|  | Họ và tên (Full name) | Mã SV (ID) | Đóng góp (Contribution) |
| Thành viên 1 (Member 1) | Bùi Anh Đức | 21020900 | Thiết kế fsmd, tổng hợp thành fsm + datapath |
| Thành viên 2  (Member 2) | Phạm Gia Khánh | 22022174 | Viết mã vhdl |
| Tên/Địa chỉ Repo trên Github hoặc Google Drive | https://github.com/baduck29/integral2d.git | | |

|  |
| --- |
| Tóm tắt (Abstract - from 5 to 10 lines) |
| Đề tài này trình bày thiết kế và mô phỏng bộ tính ảnh tích phân sử dụng ngôn ngữ mô tả phần cứng VHDL. Ảnh tích phân là kỹ thuật quan trọng trong xử lý ảnh, giúp tăng tốc các phép tính tổng vùng ảnh. Hệ thống được chia thành hai khối chính: điều khiển và datapath, thực hiện truy xuất và tính toán dữ liệu ảnh đầu vào. Quá trình mô phỏng được thực hiện trên phần mềm ModelSim nhằm kiểm tra tính đúng đắn của thiết kế. |

|  |
| --- |
| Từ khóa (Keywords) |
| Integral 2d, fsmd, datapath, vhdl. |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Time | Revised by | Description |
| V0.1 | 27/04/2021 | Nguyễn Kiêm Hùng | Original Version |
| V0.2 | 07/06/2025 | Bùi Anh Đức  Phạm Gia Khánh | Final Version |

Contents

[Document History 3](#_Toc200202635)

[MỤC LỤC 5](#_Toc200202636)

[1. Giới thiệu 7](#_Toc200202637)

[2. Yêu cầu 8](#_Toc200202638)

[2.1. Yêu cầu đối với thiết kế: 8](#_Toc200202639)

[2.2. Định nghĩa giao diện vào/ra 8](#_Toc200202640)

[3. Thuật toán 9](#_Toc200202641)

[4. Thiết kế mức RTL 9](#_Toc200202642)

[4.1. Mô hình máy FSMD 9](#_Toc200202643)

[4.2. Đơn vị xử lý dữ liệu (Datapath) 9](#_Toc200202644)

[4.3. Đơn vị điều khiển (Control Unit) 9](#_Toc200202645)

[4.4. Sơ đồ khối tổng thể 10](#_Toc200202646)

[5. Mô hình hóa bằng VHDL 10](#_Toc200202647)

[6. Mô phỏng/thực thi và đánh giá 10](#_Toc200202648)

[7. Kết luận 10](#_Toc200202649)

[Appendix A: Reference 11](#_Toc200202650)

[[1]. https://www.mathworks.com/help/images/ref/integralimage.html 11](#_Toc200202651)

[Appendix B: VHDL Code 12](#_Toc200202652)

[(đóng gói thành tệp nén và gửi kèm báo cáo) 12](#_Toc200202653)

[Appendix C: 13](#_Toc200202654)

[List of Figures 14](#_Toc200202655)

[List of Tables 15](#_Toc200202656)

[References 16](#_Toc200202657)

|  |
| --- |
| Hướng dẫn (Instructions) |
| Sinh viên điền vào báo cáo theo mẫu đính kèm. Sinh viên điền các mục:   * Thông tin sinh viên, mã số sinh viên * Mục *Đóng Góp* điền các công việc đã làm tương ứng của từng sinh viên. * Tên/Địa chỉ Repo trên Github   Ngoại trừ phần thông tin sinh viên, mã số sinh viên và tên/địa chỉ Repo trên Github ở đầu, sinh viên cần hoàn thành các phần nội dung (theo các mục đã được gợi ý – nhưng không hạn chế) trong phần báo cáo để mô tả các công việc nhóm đã thực hiện và các kết quả đã đạt được.  Sinh viên làm theo nhóm chỉ cần 1 sinh viên đại diện nộp 1 bản báo cáo, sửa tên file thành tên của các thành viên trong nhóm (viết có dấu).  Sinh viên nộp lại báo cáo này trước khi tới trình bày kết quả, muộn nhất trước ngày thi hết môn một ngày. Ngày thi, SV cần mang máy tính laptop và sản phẩm để chạy demo!  Lưu ý: Nghiêm cấm mọi hình thức copy bài (bao gồm cả report và mã nguồn) của nhau. Nếu phát hiện sự giống nhau giữa 2 bài thì tùy mức độ mà có thể sẽ bị trừ điểm hoặc chia lấy điểm trung bình làm điểm của project. |

MỤC LỤC

[Document History 3](#_Toc197372625)

[MỤC LỤC 5](#_Toc197372626)

[1. Giới thiệu 7](#_Toc197372627)

[2. Yêu cầu 8](#_Toc197372628)

[2.1. Yêu cầu đối với thiết kế: 8](#_Toc197372629)

[2.2. Định nghĩa giao diện vào/ra 8](#_Toc197372630)

[3. Thuật toán 9](#_Toc197372631)

[4. Thiết kế mức RTL 9](#_Toc197372632)

[4.1. Mô hình máy FSMD 9](#_Toc197372633)

[4.2. Đơn vị xử lý dữ liệu (Datapath) 9](#_Toc197372634)

[4.3. Đơn vị điều khiển (Control Unit) 9](#_Toc197372635)

[4.4. Sơ đồ khối tổng thể 10](#_Toc197372636)

[5. Mô hình hóa bằng VHDL 10](#_Toc197372637)

[6. Mô phỏng/thực thi và đánh giá 10](#_Toc197372638)

[7. Kết luận 10](#_Toc197372639)

[Appendix A: Reference 11](#_Toc197372640)

[[1]. https://www.mathworks.com/help/images/ref/integralimage.html 11](#_Toc197372641)

[Appendix B: VHDL Code 12](#_Toc197372642)

[(đóng gói thành tệp nén và gửi kèm báo cáo) 12](#_Toc197372643)

[Appendix C: 13](#_Toc197372644)

[List of Figures 14](#_Toc197372645)

[List of Tables 15](#_Toc197372646)

[References 16](#_Toc197372647)

# Giới thiệu*.*

Mục tiêu: Vận dụng các kiến thức, kỹ năng đã được học để thiết kế, mô phỏng và thực thi một mô-đun phần cứng thực hiện tính ảnh tích phân J = IntegralImage(I) (tương tự chức năng hàm integralImage() trong Matlab [1]) từ một ảnh gốc I. Trong đó, mỗi pixel trong hình ảnh tích phân J đại diện cho tổng tích lũy của một pixel đầu vào với tất cả các pixel ở phía trên và bên trái của pixel đầu vào trong ảnh gốc I. Phép chuyển đổi hình ảnh được mô tả bằng ví dụ sau.

*Ví dụ:* nếu hình ảnh đầu vào I là ma trận có kích thước 5×5 như sau:

I =

thì kết quả tính toán của khối chuyển đổi IntegralImage trả về hình ảnh J có kích thước 6×6 như sau:

J =

Hình 1. Ví dụ minh họa phép chuyển đổi hình ảnh.

Lưu ý rằng pixel có tọa độ (r,c) – (hàng, cột) – tronh ảnh gốc được ánh xạ tới pixel có tọa độ (r + 1, c + 1) trong ảnh tích phân. Các bước tính hình ảnh tích phân của hình ảnh đầu vào I được minh họa như sau:

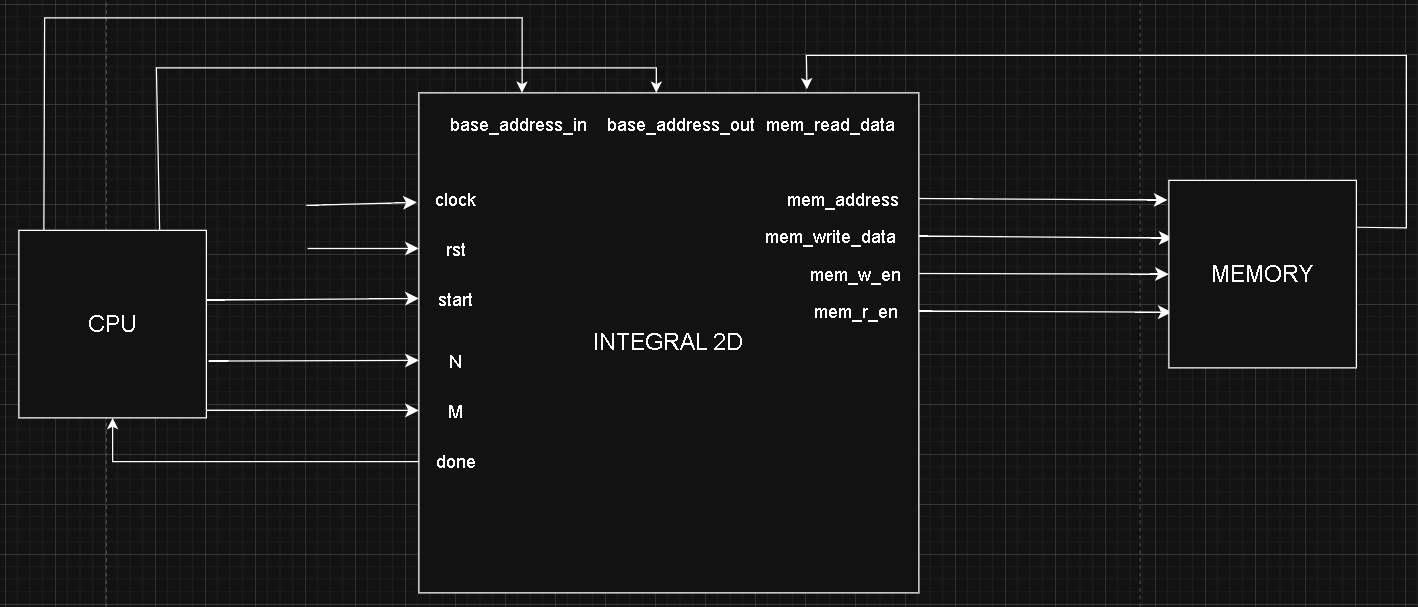
* Các pixel trong hàng và cột đầu tiên trong ảnh tích phân đều là 0
* Pixel trong ma trận ban đầu tại tọa độ (1, 1) có giá trị 17 không thay đổi trong ảnh tích phân vì không có pixel nào khác trong tổng. Do đó, pixel trong ảnh tích phân tại tọa độ (2, 2) có giá trị 17.
* Pixel trong ma trận ban đầu tại tọa độ (1, 2) ánh xạ tới pixel (2, 3) trong ảnh tích phân. Giá trị là tổng của giá trị pixel ban đầu (24), các pixel phía trên nó (0) và các pixel bên trái của nó (17): 24 + 17 + 0 = 41.
* Pixel trong ma trận ban đầu tại tọa độ (1, 3) ánh xạ tới pixel (2, 4) trong hình ảnh tích phân. Giá trị là tổng của giá trị pixel ban đầu (1), pixel phía trên nó (0) và các pixel bên trái của nó (đã được cộng lại bằng 41). Do đó, giá trị tại pixel (2,4) trong ảnh tích phân là 1 + 41 + 0 = 42.

# Yêu cầu

## Yêu cầu đối với thiết kế:

* Khối IntegralImage có giao diện ghép nối tới CPU sao cho CPU có thể thiết lập địa chỉ cơ sở của hình ảnh gốc, kích thước N×M (5×5 - 256×256)của hình ảnh, địa chỉ cơ sở của vùng nhớ lưu hình ảnh tích phân trả về
* CPU kích hoạt quá trình tính toán của khối IntegralImage bằng các đặt tín hiệu Start = ‘1’.
* Sau khi quá trình tính hình ảnh tích phân hoàn thành, khối IntegralImage sẽ báo cho CPU biết bằng cách đặt tín hiệu Done = ‘1’;

Hình 2. Giao diện ghép nối I/O.



## Định nghĩa giao diện vào/ra

Bảng 1: Mô tả các tín hiệu vào ra.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Port | Direction | Width | Meaning |
| 1 | clk | In | 1 bit | Xung clock |
| 2 | rst | In | 1 bit | Tín hiệu reset |
| 3 | start | In | 1 bit | Tín hiệu bắt đầu gửi từ cpu |
| 4 | done | In | 1 bit | Tín hiệu hoàn thành tính toán gửi đến cpu |
| 5 | N | In | 32 bit | Kích thước ảnh gửi từ cpu |
| 6 | M | In | 32 bit | Kích thước ảnh gửi từ cpu |
| 7 | Base\_address\_in | In | 32 bit | Vị trí ảnh đầu vào gửi từ cpu |
| 8 | Base\_address\_out | In | 32 bit | Vị trí ảnh đầu ra gửi từ cpu |
| 9 | Mem\_read\_data | In | 32 bit | Dữ liệu đọc được từ memory |
| 10 | Mem\_address | Out | 32 bit | Vị trí cần lấy dữ liệu trong memory |
| 11 | Mem\_write\_data | Out | 32 bit | Dữ liệu ghi vào trong memory |
| 12 | Mem\_r\_en | Out | 1 bit | Cho phép đọc từ memory |
| 13 | Mem\_w\_en | Out | 1 bit | Cho phép ghi vào memory |

# Thuật toán

Thuật toán tính ảnh tích phân:

Các bước chính:

1. Khởi tạo ma trận ảnh tích phân:
   * Gán giá trị 0 cho toàn bộ hàng đầu tiên (i,0) và cột đầu tiên (0,j) của ảnh tích phân.
2. Tính giá trị từng pixel trong ảnh tích phân:
   * Với mỗi pixel tại vị trí (i,j) (bắt đầu từ i = 2, j = 2):
     + Đọc giá trị gốc từ ảnh đầu vào tại (i−1, j−1).
     + Đọc các giá trị đã tính trước từ ảnh tích phân:
       - Bên trái: (i, j−1)
       - Phía trên: (i−1, j)
       - Chéo trên trái: (i−1, j−1)
     + Tính giá trị ảnh tích phân tại (i,j) theo công thức:

integral(i,j) = input(i-1,j-1) + integral(i,j-1) + integral(i-1,j) - integral(i-1,j-1)

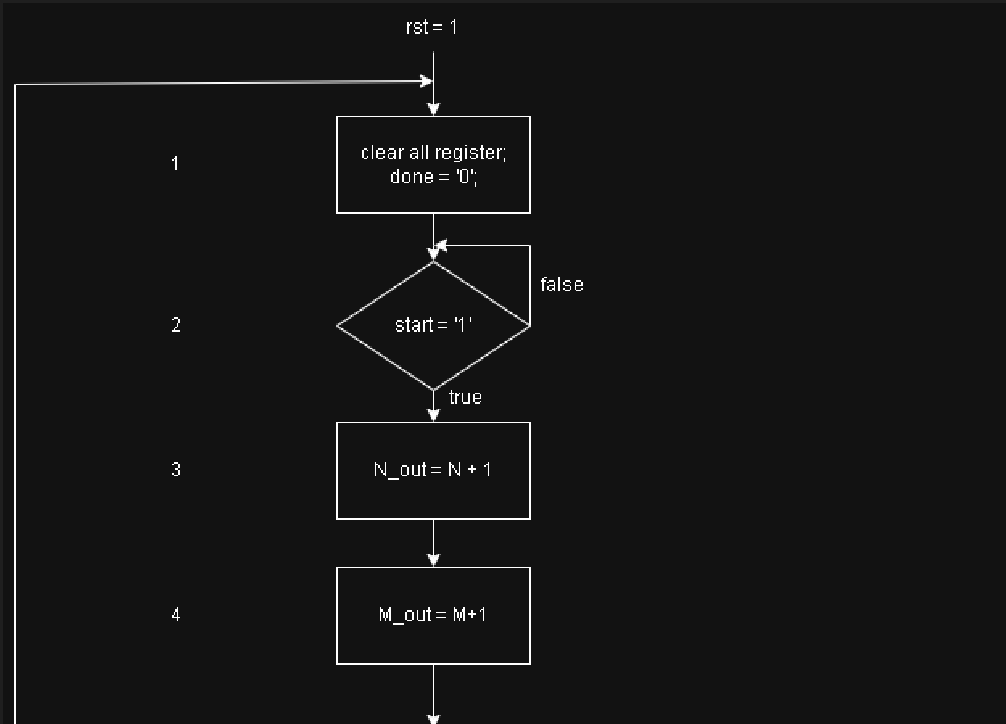
* + - Ghi giá trị này vào bộ nhớ đầu ra.

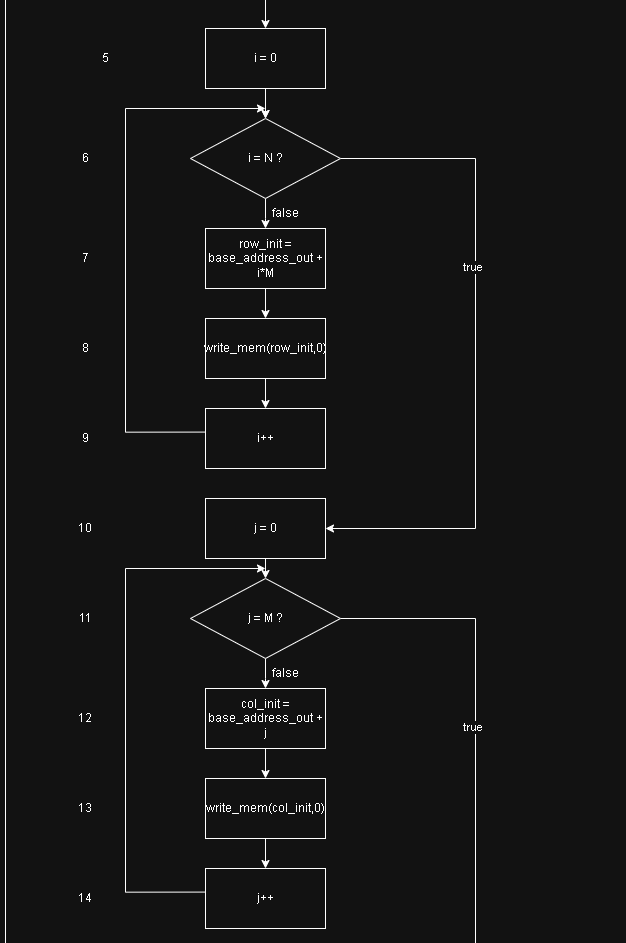
1. Kết thúc:
   * Đặt cờ done = 1 để báo hiệu hoàn thành việc tính ảnh tích phân.

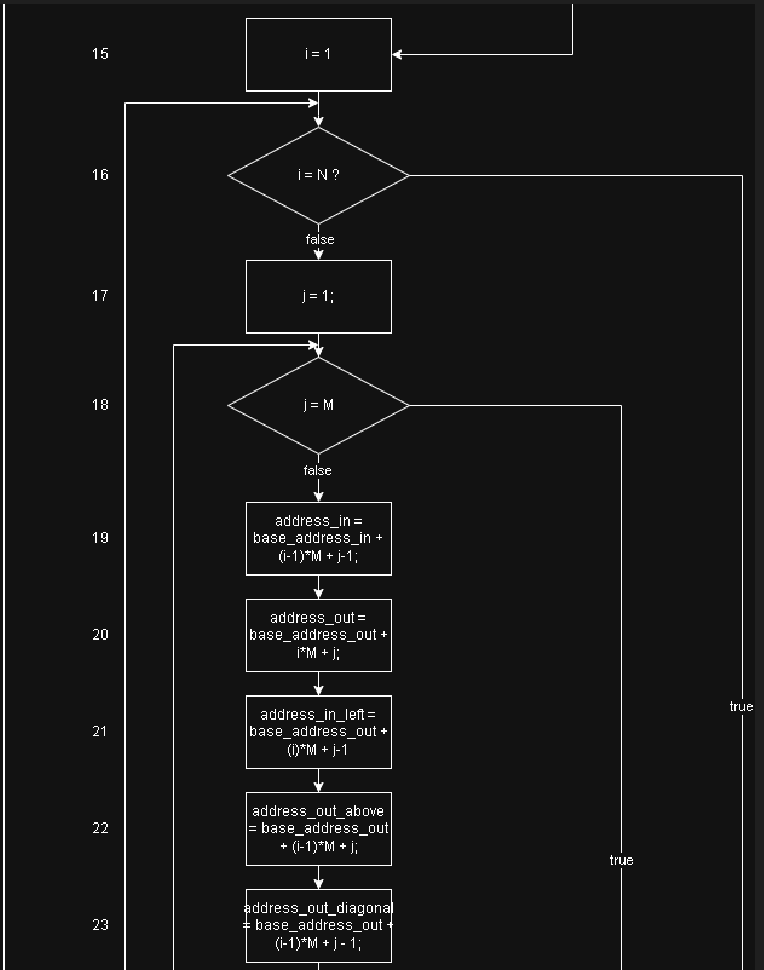
|  |
| --- |
| Mã giả cho thuật toán |
| int base\_address\_in, base\_address\_out, N, M;  int i, j;  int address\_in, address\_out;  int addr\_left, addr\_above, addr\_diag;  int pixel\_in, sum\_left, sum\_above, sum\_diag, pixel\_out;  int start, done;  // Tăng kích thước để thêm hàng và cột biên (giá trị 0)  N++;  M++;  while (1) {  if (start) {  // Khởi tạo hàng đầu (i, 0)  for (i = 1; i <= N; i++) {  write\_mem(base\_address\_out + i \* M, 0);  }  // Khởi tạo cột đầu (0, j)  for (j = 1; j <= M; j++) {  write\_mem(base\_address\_out + j, 0);  }  // Tính toán ảnh tích phân từ (2,2) đến (N,M)  for (i = 2; i <= N; i++) {  for (j = 2; j <= M; j++) {  address\_in = base\_address\_in + (i - 1) \* M + (j - 1);  address\_out = base\_address\_out + i \* M + j;  addr\_left = base\_address\_out + i \* M + (j - 1);  addr\_above = base\_address\_out + (i - 1) \* M + j;  addr\_diag = base\_address\_out + (i - 1) \* M + (j - 1);  pixel\_in = read\_mem(address\_in);  sum\_left = read\_mem(addr\_left);  sum\_above = read\_mem(addr\_above);  sum\_diag = read\_mem(addr\_diag);  pixel\_out = pixel\_in + sum\_left + sum\_above - sum\_diag;  write\_mem(address\_out, pixel\_out);  }  }  done = 1;  start = 0;  }  } |

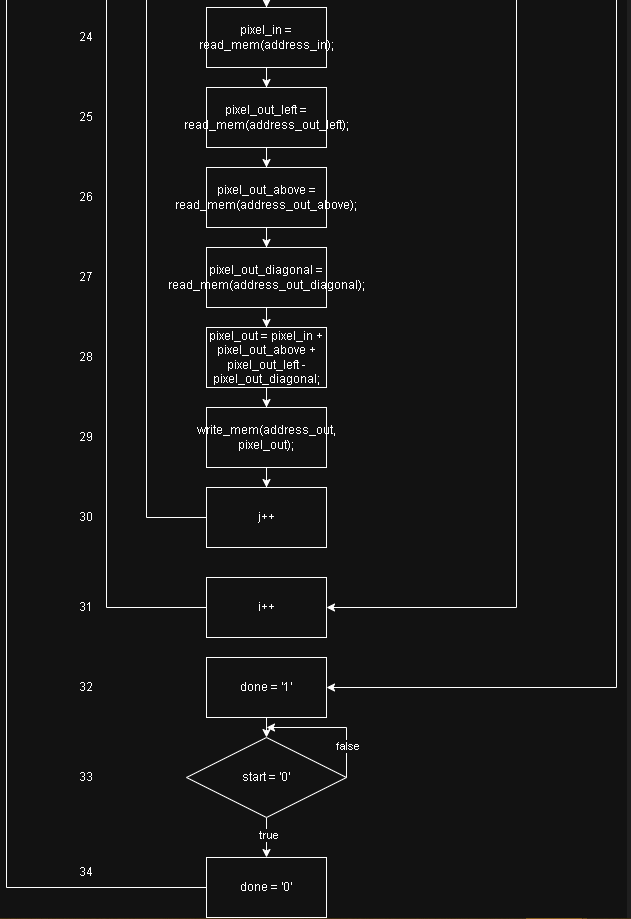
# Thiết kế mức RTL

## Mô hình máy FSMD



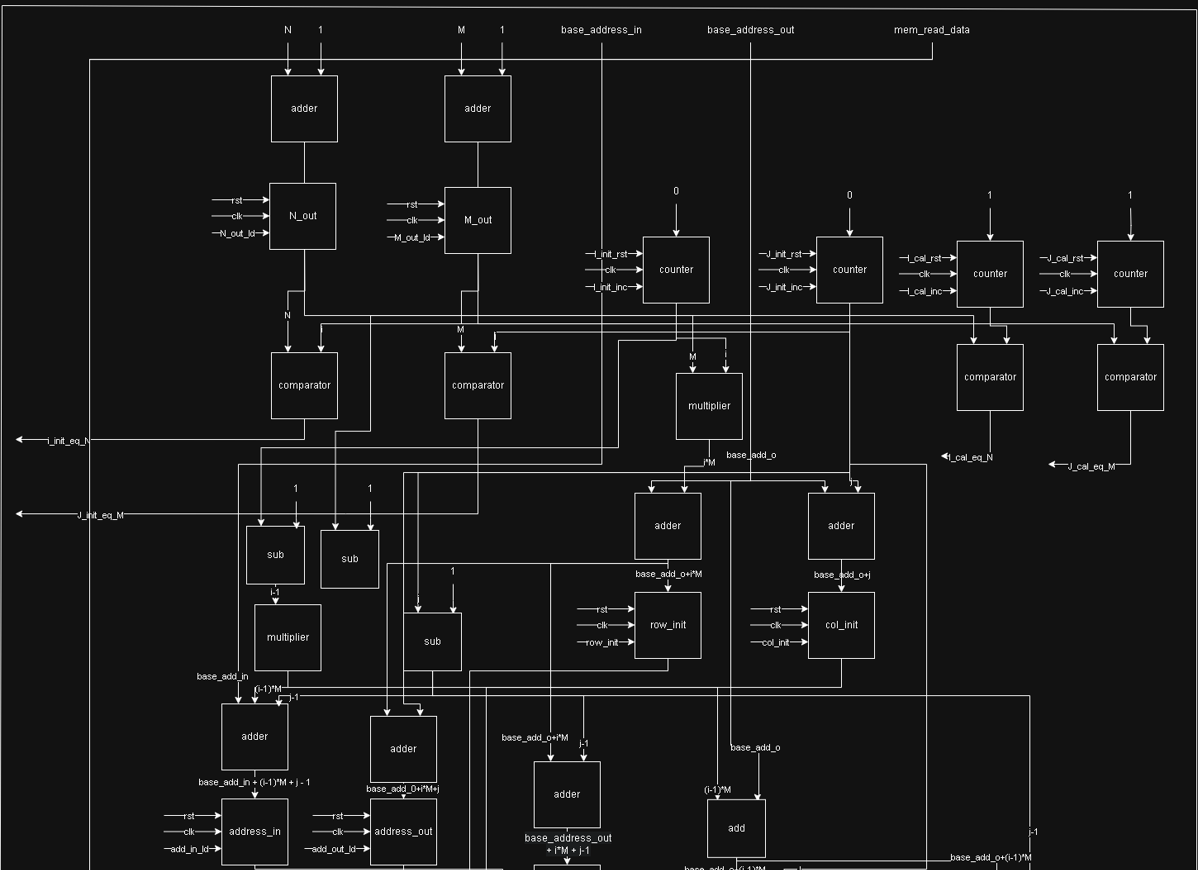


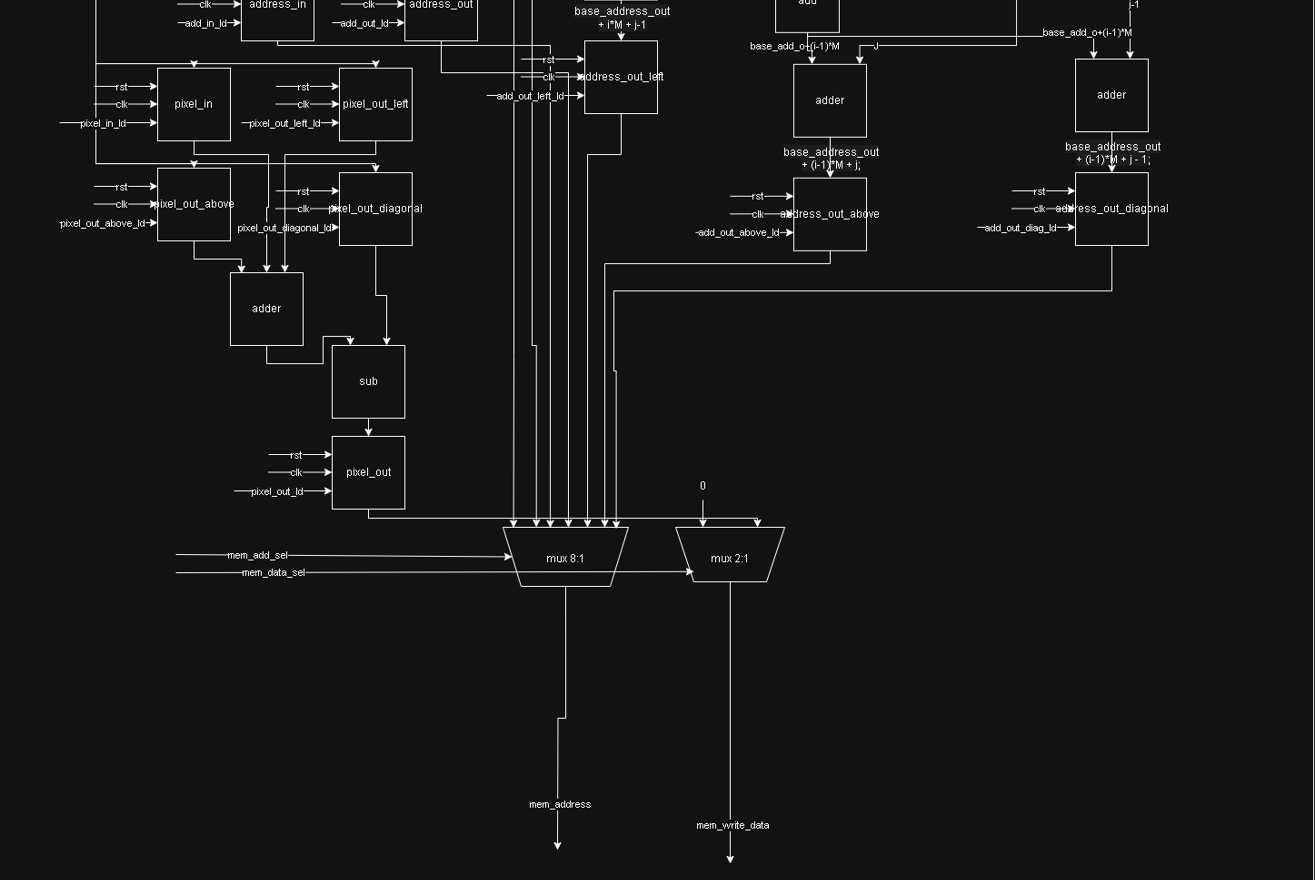




Hình 3: Mô hình máy FSMD.

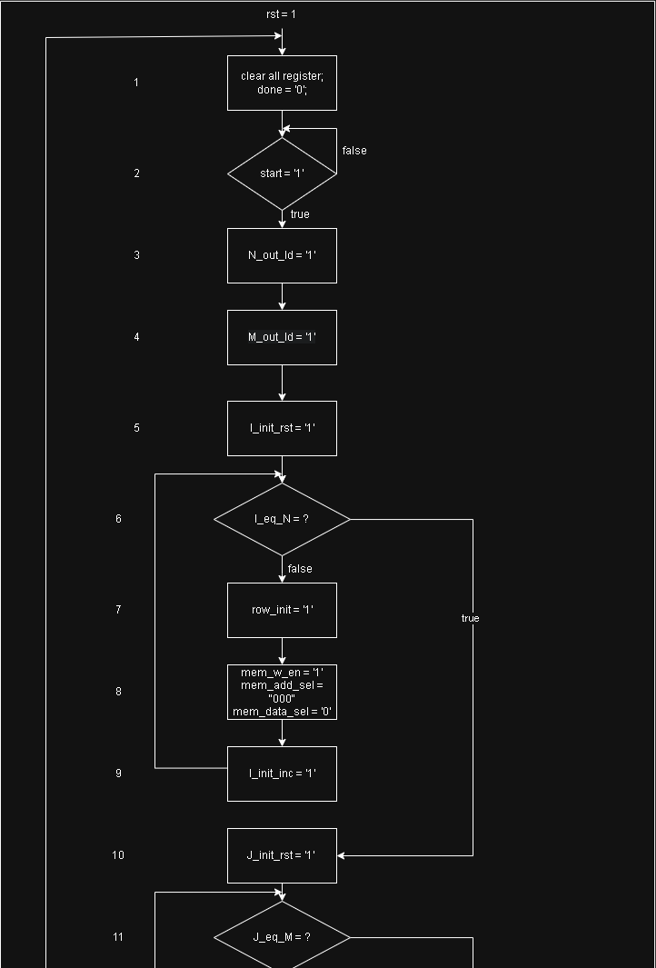
## Đơn vị xử lý dữ liệu (Datapath)

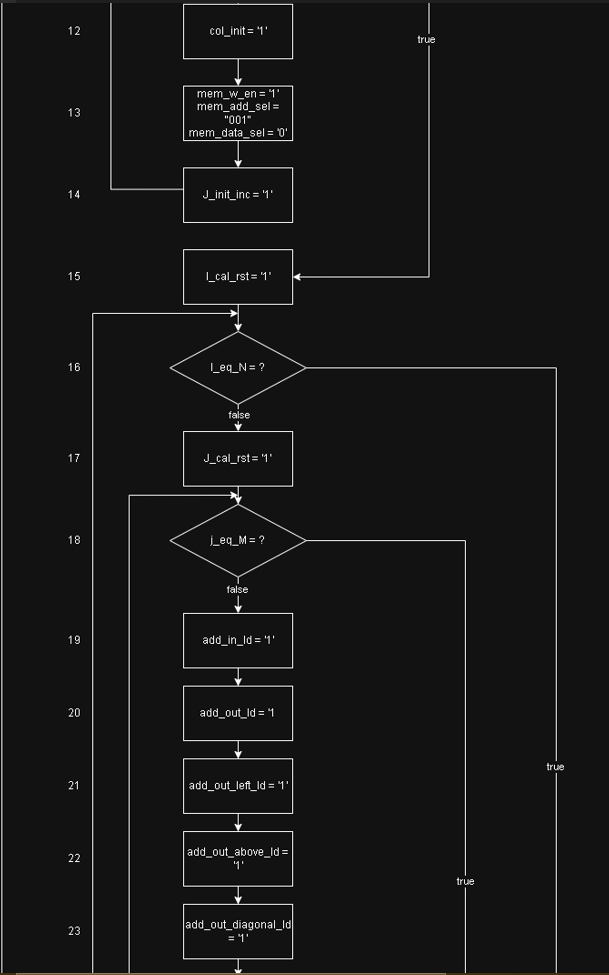
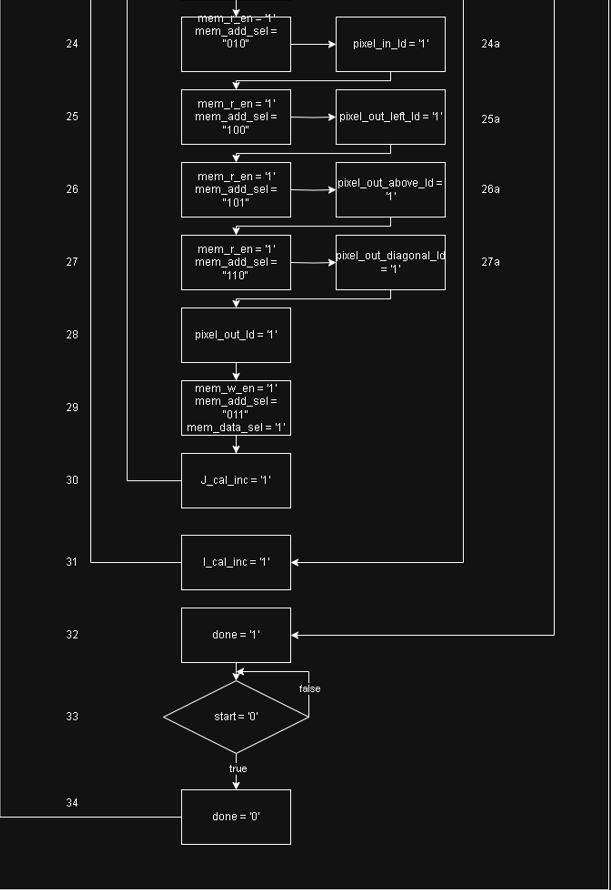




Hình 4: Cấu trúc của đơn vị xử lý dữ liệu Datapath.

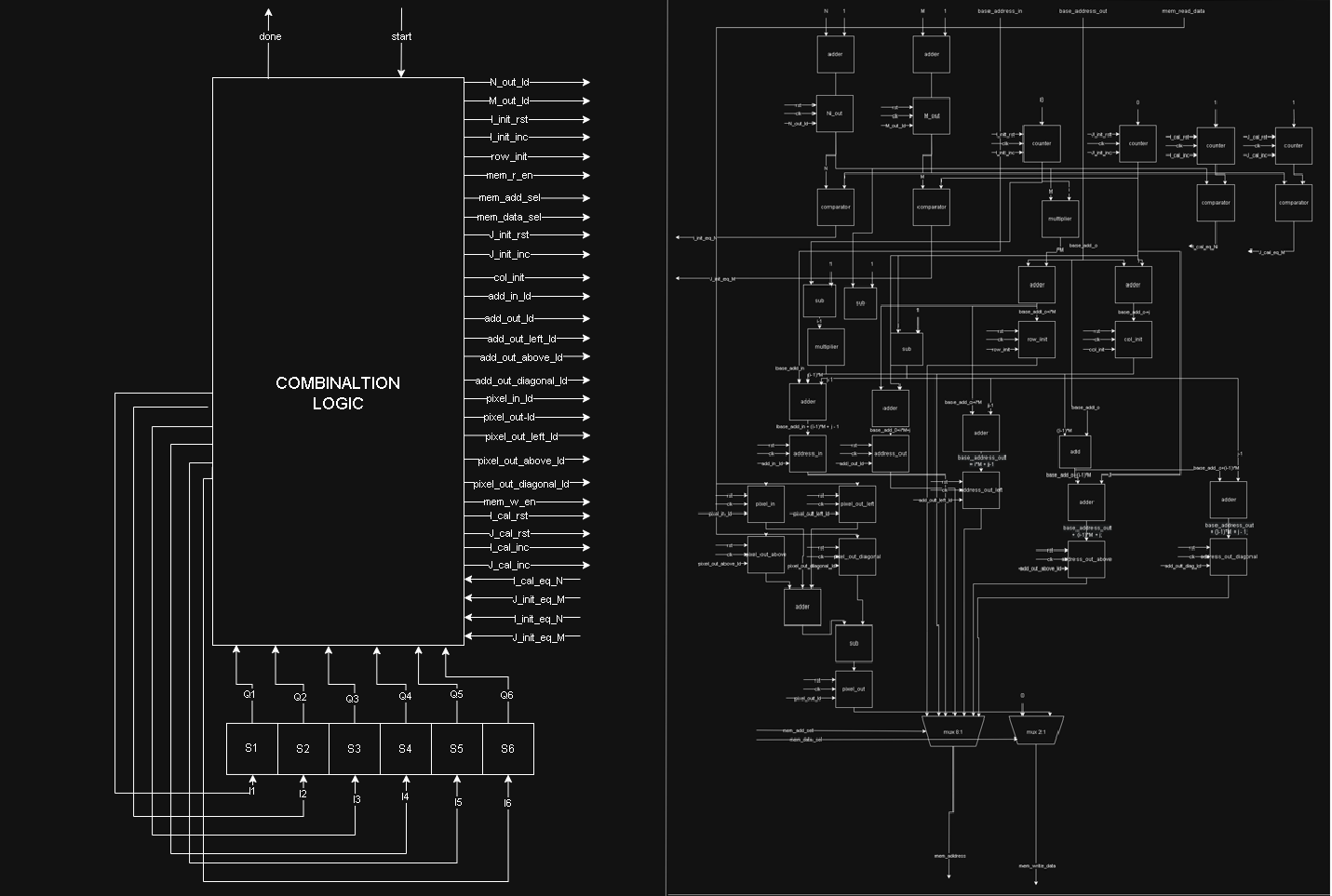
## Đơn vị điều khiển (Control Unit)



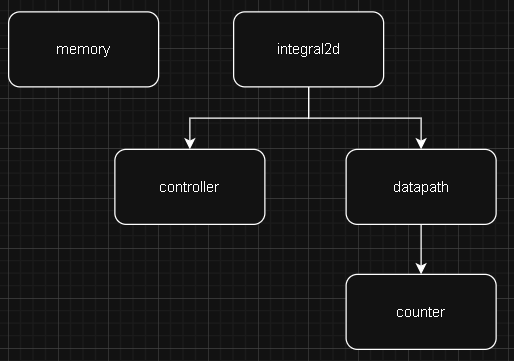
Hình 5: Máy FSM của đơn vị điều khiển.

## Sơ đồ khối tổng thể



Hình 5: Sơ đồ khối tổng thể của bộ tính ảnh tích phân.

# Mô hình hóa bằng VHDL



# Mô phỏng/thực thi và đánh giá

## Controller

### **Tín hiệu điều khiển:**

1. **Đầu vào**:
   * rst: Xóa trạng thái về S1.
   * clk: Xung nhịp đồng bộ.
   * start: Khởi động quá trình xử lý.
   * I\_init\_eq\_N, J\_init\_eq\_M, I\_cal\_eq\_N, J\_cal\_eq\_M: Tín hiệu so sánh từ bộ đếm của datapath, báo hiệu khi đạt giá trị tối đa.
2. **Đầu ra cho datapath**:
   * **Thanh ghi**:
     + N\_out\_ld, M\_out\_ld: Nạp kích thước ảnh đầu ra.
     + row\_init, col\_init: Nạp địa chỉ hàng/cột khởi tạo.
     + add\_in\_ld, add\_out\_ld, add\_out\_left\_ld, add\_out\_above\_ld, add\_out\_diagonal\_ld: Nạp địa chỉ dữ liệu vào/ra và lân cận.
     + pixel\_in\_ld, pixel\_out\_ld, pixel\_out\_left\_ld, pixel\_out\_above\_ld, pixel\_out\_diagonal\_ld: Nạp giá trị pixel vào/ra và lân cận.
   * **Bộ đếm**:
     + I\_init\_rst, J\_init\_rst, I\_cal\_rst, J\_cal\_rst: Xóa bộ đếm khởi tạo và tính toán.
     + I\_init\_inc, J\_init\_inc, I\_cal\_inc, J\_cal\_inc: Tăng giá trị bộ đếm.
3. **Đầu ra cho bộ nhớ**:
   * mem\_w\_en, mem\_r\_en: Kích hoạt ghi/đọc bộ nhớ.
   * mem\_add\_sel (3-bit): Chọn nguồn địa chỉ bộ nhớ (row\_init, col\_init, add\_in, add\_out, add\_out\_left, add\_out\_above, add\_out\_diagonal).
   * mem\_data\_sel: Chọn dữ liệu ghi vào bộ nhớ (pixel\_out hoặc 0).
4. **Tín hiệu trạng thái**:
   * done: Báo hiệu hoàn tất quá trình.

### **Hoạt động chính:**

* **Máy trạng thái hữu hạn (FSM)**: Gồm 38 trạng thái, điều khiển tuần tự:
  + **Khởi tạo (S1–S5)**:
    - Đợi start (S2), nạp kích thước ảnh (N\_out\_ld, M\_out\_ld), xóa bộ đếm khởi tạo (I\_init\_rst, J\_init\_rst).
  + **Vòng lặp khởi tạo (S6–S14)**:
    - Tăng I\_init (S9) đến khi I\_init\_eq\_N = '1' (S6), nạp row\_init (S7).
    - Tăng J\_init (S14) đến khi J\_init\_eq\_M = '1' (S11), nạp col\_init (S12).
    - Ghi địa chỉ vào bộ nhớ (S8, S13).
  + **Vòng lặp tính toán (S15–S31)**:
    - Xóa bộ đếm tính toán (I\_cal\_rst, J\_cal\_rst) (S15, S17).
    - Tăng I\_cal (S31) đến khi I\_cal\_eq\_N = '1' (S16), tăng J\_cal (S30) đến khi J\_cal\_eq\_M = '1' (S18).
    - Nạp địa chỉ (add\_in\_ld, add\_out\_ld, add\_out\_left\_ld, add\_out\_above\_ld, add\_out\_diagonal\_ld) (S19–S23).
    - Đọc pixel từ bộ nhớ (pixel\_in\_ld, pixel\_out\_left\_ld, pixel\_out\_above\_ld, pixel\_out\_diagonal\_ld) (S24–S27, S24a–S27a).
    - Tính toán và nạp pixel đầu ra (pixel\_out\_ld) (S28), ghi vào bộ nhớ (S29).
  + **Hoàn tất (S32–S34)**:
    - Đặt done = '1' (S32), trở về S1.
* **Điều khiển bộ nhớ**:
  + Kích hoạt ghi (mem\_w\_en) ở S8, S13, S29.
  + Kích hoạt đọc (mem\_r\_en) ở S24–S27.
  + Chọn địa chỉ qua mem\_add\_sel tương ứng với trạng thái.
  + Chọn dữ liệu ghi (mem\_data\_sel) ở S29.

## DATAPATH

### **Tín hiệu điều khiển:**

1. **Clock và Reset**:
   * clk: Xung nhịp đồng bộ hóa.
   * rst: Xóa các thanh ghi về trạng thái ban đầu.
2. **Điều khiển bộ đếm**:
   * I\_init\_rst, J\_init\_rst, I\_cal\_rst, J\_cal\_rst: Xóa các bộ đếm khởi tạo (I\_init, J\_init) và tính toán (I\_cal, J\_cal).
   * I\_init\_inc, J\_init\_inc, I\_cal\_inc, J\_cal\_inc: Tăng giá trị các bộ đếm.
3. **Điều khiển thanh ghi**:
   * M\_out\_ld, N\_out\_ld: Nạp kích thước ảnh đầu ra (M\_out, N\_out).
   * row\_init, col\_init: Nạp địa chỉ hàng/cột khởi tạo.
   * add\_in\_ld, add\_out\_ld, add\_out\_above\_ld, add\_out\_left\_ld, add\_out\_diagonal\_ld: Nạp địa chỉ dữ liệu vào/ra và các vị trí lân cận (trái, trên, chéo).
   * pixel\_in\_ld, pixel\_out\_ld, pixel\_out\_left\_ld, pixel\_out\_above\_ld, pixel\_out\_diagonal\_ld: Nạp giá trị pixel vào/ra và các pixel lân cận.
4. **Điều khiển bộ nhớ**:
   * mem\_add\_sel (3-bit): Chọn địa chỉ bộ nhớ từ các nguồn (row\_init\_add, col\_init\_add, add\_in\_out,...).
   * mem\_data\_sel: Chọn dữ liệu ghi vào bộ nhớ (pixel\_out\_out hoặc 0).

### **Hoạt động chính:**

* **Khởi tạo**:
  + Lưu kích thước ảnh đầu ra (M\_out = M+1, N\_out = N+1) vào thanh ghi.
  + Tính toán và lưu địa chỉ khởi tạo (row\_init\_add, col\_init\_add) dựa trên base\_address\_out, I\_init, J\_init.
* **Xử lý địa chỉ và pixel**:
  + Tính toán địa chỉ đầu vào (add\_in\_in) và đầu ra (add\_out\_in, add\_out\_left\_in, add\_out\_above\_in, add\_out\_diagonal\_in) dựa trên base\_address\_in/out, I\_cal, J\_cal, và M\_out.
  + Lưu trữ các địa chỉ này vào các thanh ghi tương ứng.
  + Đọc dữ liệu pixel từ bộ nhớ (mem\_read\_data) và lưu vào các thanh ghi pixel (pixel\_in, pixel\_out\_left, pixel\_out\_above, pixel\_out\_diagonal).
  + Tính toán giá trị pixel đầu ra (pixel\_out\_in) bằng công thức: pixel\_in + pixel\_out\_left + pixel\_out\_above - pixel\_out\_diagonal.
* **Bộ đếm**:
  + Bốn bộ đếm (I\_init, J\_init, I\_cal, J\_cal) đếm từ 0/1 đến N\_out/M\_out, với tín hiệu z (I\_init\_eq\_N, J\_init\_eq\_M, I\_cal\_eq\_N, J\_cal\_eq\_M) báo hiệu khi đạt giá trị tối đa.
* **Giao tiếp bộ nhớ**:
  + Địa chỉ bộ nhớ (mem\_address) được chọn qua mem\_add\_sel.
  + Dữ liệu ghi vào bộ nhớ (mem\_write\_data) được chọn từ pixel\_out\_out hoặc 0 dựa trên mem\_data\_sel.

## REGISTER

### Tín hiệu điều khiển:

* Đầu vào:
  + clr: Xóa thanh ghi về 0 khi clr = '1'.
  + clk: Xung nhịp đồng bộ hóa.
  + en: Kích hoạt nạp dữ liệu khi en = '1'.
  + d: Dữ liệu đầu vào kiểu integer.
* Đầu ra:
  + q: Giá trị đầu ra của thanh ghi, kiểu integer.

### Hoạt động chính:

* Khi clr = '1', đầu ra q được đặt về 0.
* Khi có cạnh lên của clk và en = '1', giá trị d được nạp vào q.
* Nếu en = '0', q giữ giá trị hiện tại.
* Quá trình này được đồng bộ với cạnh lên của clk.

## COUNTER

### Tín hiệu điều khiển:

* Đầu vào:
  + floor\_value: Giá trị tối thiểu của bộ đếm.
  + ceil\_value: Giá trị tối đa của bộ đếm.
  + clk: Xung nhịp đồng bộ hóa.
  + inc: Tăng giá trị bộ đếm khi inc = '1'.
  + clr: Xóa bộ đếm về floor\_value khi clr = '1'.
* Đầu ra:
  + z: Tín hiệu báo hiệu khi bộ đếm đạt ceil\_value (z = '1').
  + count: Giá trị hiện tại của bộ đếm, kiểu integer.

### Hoạt động chính:

* Khi clr = '1', bộ đếm (temp\_counter) được đặt về floor\_value.
* Khi có cạnh lên của clk và inc = '1', nếu temp\_counter < ceil\_value, giá trị temp\_counter tăng thêm 1.
* Tín hiệu z được đặt thành '1' khi temp\_counter = ceil\_value, ngược lại là '0'.
* Giá trị count xuất ra giá trị hiện tại của temp\_counter.

## MEMORY

### **Tín hiệu điều khiển:**

* **Đầu vào**:
  + clk: Xung nhịp đồng bộ hóa.
  + we\_in: Kích hoạt ghi dữ liệu vào bộ nhớ khi we\_in = '1'.
  + re\_in: Kích hoạt đọc dữ liệu từ bộ nhớ khi re\_in = '1'.
  + addr: Địa chỉ bộ nhớ (phạm vi từ 0 đến 60).
  + d\_in: Dữ liệu đầu vào kiểu integer để ghi vào bộ nhớ.
* **Đầu ra**:
  + d\_out: Dữ liệu đọc từ bộ nhớ, kiểu integer.

### **Hoạt động chính:**

* **Khởi tạo bộ nhớ**:
  + Bộ nhớ được mô phỏng bằng một mảng DATA\_ARRAY (61 phần tử, từ 0 đến 60).
  + 25 phần tử đầu (0 đến 24) chứa giá trị của một ảnh đầu vào 5x5 với các giá trị:
  + 36 phần tử sau (25 đến 60) dành cho ảnh đầu ra 6x6, khởi tạo bằng 0.
* **Quá trình đọc/ghi**:
  + Đồng bộ với cạnh lên của clk:
    - Nếu we\_in = '1', ghi dữ liệu d\_in vào địa chỉ addr của mảng memory.
    - Nếu re\_in = '1', xuất dữ liệu từ địa chỉ addr của memory ra d\_out.
    - Nếu re\_in = '0', đặt d\_out = 0.
* **Cơ chế hoạt động**:
  + Ghi và đọc được thực hiện độc lập trong cùng một chu kỳ xung nhịp.
  + Bộ nhớ lưu trữ dữ liệu tạm thời cho quá trình xử lý hình ảnh (ví dụ, tích phân 2D).

## Integral 2d

### Tín hiệu điều khiển:

* Đầu vào:
  + M, N: Kích thước ảnh (từ 5 đến 256).
  + rst: Xóa hệ thống về trạng thái ban đầu.
  + clk: Xung nhịp đồng bộ hóa.
  + start: Khởi động quá trình xử lý.
  + base\_address\_in, base\_address\_out: Địa chỉ cơ sở cho dữ liệu vào/ra (0 đến 255).
  + mem\_read\_data: Dữ liệu đọc từ bộ nhớ, kiểu integer.
* Đầu ra:
  + mem\_w\_en, mem\_r\_en: Kích hoạt ghi/đọc bộ nhớ.
  + mem\_write\_data: Dữ liệu ghi vào bộ nhớ, kiểu integer.
  + mem\_address: Địa chỉ bộ nhớ (0 đến 255).
  + done: Báo hiệu hoàn tất quá trình xử lý.
* Tín hiệu nội bộ (kết nối giữa controller và datapath):
  + Điều khiển thanh ghi: M\_out\_ld, N\_out\_ld, row\_init, col\_init, add\_in\_ld, add\_out\_ld, add\_out\_left\_ld, add\_out\_above\_ld, add\_out\_diagonal\_ld, pixel\_in\_ld, pixel\_out\_ld, pixel\_out\_left\_ld, pixel\_out\_above\_ld, pixel\_out\_diagonal\_ld.
  + Điều khiển bộ đếm: I\_init\_rst, J\_init\_rst, I\_cal\_rst, J\_cal\_rst, I\_init\_inc, J\_init\_inc, I\_cal\_inc, J\_cal\_inc.
  + Tín hiệu so sánh: I\_init\_eq\_N, J\_init\_eq\_M, I\_cal\_eq\_N, J\_cal\_eq\_M.
  + Điều khiển bộ nhớ: mem\_data\_sel, mem\_add\_sel (3-bit).

### Hoạt động chính:

* Tích hợp:
  + Controller: Điều khiển tuần tự các bước xử lý thông qua FSM (38 trạng thái), tạo tín hiệu điều khiển cho datapath và bộ nhớ, đặt done = '1' khi hoàn tất.
  + Datapath: Thực hiện tính toán địa chỉ và dữ liệu pixel, sử dụng các thanh ghi và bộ đếm để xử lý hình ảnh dựa trên các tín hiệu điều khiển từ controller.
* Quy trình xử lý:
  + Khởi tạo kích thước ảnh (M, N) và địa chỉ cơ sở (base\_address\_in, base\_address\_out).
  + Tính toán địa chỉ và pixel theo thuật toán tích phân 2D, lưu trữ tạm thời trong các thanh ghi.
  + Đọc dữ liệu pixel từ bộ nhớ (mem\_read\_data), thực hiện phép tính tích phân (pixel\_out = pixel\_in + pixel\_out\_left + pixel\_out\_above - pixel\_out\_diagonal), và ghi kết quả vào bộ nhớ (mem\_write\_data).
  + Quản lý địa chỉ bộ nhớ (mem\_address) thông qua mem\_add\_sel.
* Giao tiếp bộ nhớ:
  + Kích hoạt đọc (mem\_r\_en) để lấy pixel, ghi (mem\_w\_en) để lưu kết quả.
  + Chọn dữ liệu ghi (mem\_data\_sel) và địa chỉ (mem\_add\_sel) dựa trên trạng thái FSM.

## TEST BENCH cho mạch.

Entity integral2d\_tb là một **testbench** để mô phỏng và kiểm tra hoạt động của module integral2d kết hợp với memory\_block, mô phỏng xử lý hình ảnh tích phân 2D với kích thước ảnh 5x5.

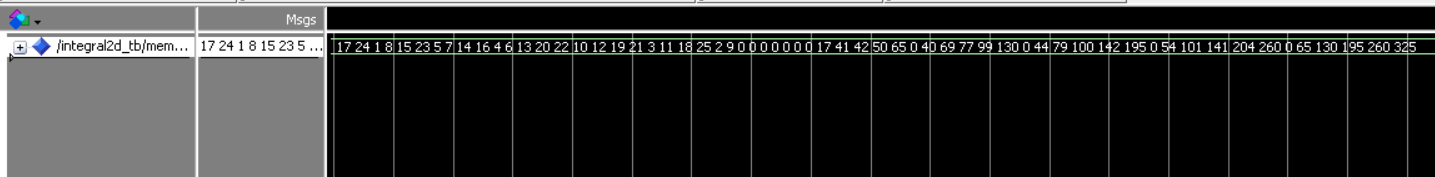
### **Tín hiệu điều khiển:**

* **Hằng số**:
  + clk\_period: Chu kỳ xung nhịp (50 ns).
  + N, M: Kích thước ảnh đầu vào (5x5).
  + base\_address\_out\_max: Địa chỉ tối đa cho bộ nhớ đầu ra (60, tính từ 25 đến 60 cho ảnh đầu ra 6x6).
* **Tín hiệu**:
  + clk: Xung nhịp, đảo trạng thái mỗi clk\_period.
  + rst: Tín hiệu reset hệ thống.
  + start: Khởi động quá trình xử lý.
  + done: Báo hiệu hoàn tất từ integral2d.
  + mem\_address: Địa chỉ bộ nhớ (0 đến 60).
  + mem\_read\_data, mem\_write\_data: Dữ liệu đọc/ghi từ bộ nhớ.
  + mem\_w\_en, mem\_r\_en: Kích hoạt ghi/đọc bộ nhớ.
  + base\_address\_in: Địa chỉ cơ sở cho ảnh đầu vào (0 đến 24).
  + base\_address\_out: Địa chỉ cơ sở cho ảnh đầu ra (25 đến 60).

### **Hoạt động chính:**

* **Khởi tạo**:
  + Tạo xung nhịp clk với chu kỳ 50 ns bằng cách đảo trạng thái sau mỗi clk\_period.
  + Kết nối module integral2d với memory\_block thông qua các tín hiệu điều khiển và dữ liệu.
* **Quá trình kiểm tra (testing)**:
  + Đặt base\_address\_in = 0 (địa chỉ bắt đầu của ảnh đầu vào 5x5).
  + Đặt base\_address\_out = 25 (địa chỉ bắt đầu của ảnh đầu ra 6x6).
  + Kích hoạt rst = '1' trong 3 chu kỳ xung nhịp để xóa hệ thống.
  + Đặt rst = '0' và start = '1' để bắt đầu xử lý.
  + Đợi đến khi done = '1' (tín hiệu hoàn tất từ integral2d), sau đó chờ thêm một chu kỳ xung nhịp trước khi dừng mô phỏng.
* **Mô phỏng**:
  + integral2d thực hiện thuật toán tích phân 2D trên ảnh đầu vào 5x5 (lưu trong memory\_block).
  + memory\_block cung cấp dữ liệu đầu vào (mem\_read\_data) và lưu trữ kết quả đầu ra (mem\_write\_data) tại các địa chỉ được chỉ định bởi mem\_address.

## Kết quả mô phỏng.



* Với đầu vào:
* Kết quả mong muốn đạt được là:
* Có thể thấy memory sau khi chạy xong có lối ra chính xác.

# Kết luận.

Trong bài báo cáo này, chúng tôi đã trình bày quá trình thiết kế và hiện thực một hệ thống tính ảnh tích phân sử dụng ngôn ngữ mô tả phần cứng VHDL. Hệ thống được chia thành ba khối chính: **controller**, **datapath** và **bộ nhớ**, trong đó controller đảm nhiệm điều khiển luồng hoạt động, datapath thực hiện các phép tính địa chỉ và giá trị pixel, còn bộ nhớ đảm bảo lưu trữ và truy xuất dữ liệu chính xác.

Thông qua mô phỏng và kiểm thử, hệ thống cho kết quả chính xác, đảm bảo tính đúng đắn của thuật toán ảnh tích phân. Thiết kế sử dụng kiến trúc tuần tự điều khiển bởi FSM, đảm bảo tính mở rộng và dễ dàng tích hợp trong các hệ thống xử lý ảnh thời gian thực.

Kết quả đạt được là cơ sở cho các nghiên cứu và ứng dụng tiếp theo trong lĩnh vực thị giác máy và xử lý ảnh số, đặc biệt là các bài toán như phát hiện vật thể, nhận diện đặc trưng và trích xuất vùng quan tâm (ROI) trong ảnh.