TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HCM
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

ĐỒ ÁN THIẾT KẾ
KỸ THUẬT MÁY TÍNH

Thiết kế SoC RISC-V tích hợp EdgeAI cho ứng dụng IoT

Học kỳ 251

GVHD: PGS. TS. Trần Ngọc Thịnh
ThS. Huỳnh Phúc Nghị

STT	Họ và tên	MSSV	Ghi chú
1	Lâm Nữ Uyển Nhi	2212429	
2	Vũ Đức Lâm	2211824	

TP. Hồ Chí Minh, Tháng 12/2025

Mục lục

Danh sách hình vẽ

Danh sách bảng biểu

Danh mục Ký hiệu và Chữ viết tắt

Ký hiệu	**Ý nghĩa**
N	Kích thước lô (Batch size)
C	Số lượng kênh đầu vào (Input Channels)
M	Số lượng kênh đầu ra (Output Channels/Filters)
H_{in}, W_{in}	Chiều cao và chiều rộng của đặc trưng đầu vào (Input Feature Map)
H_{out}, W_{out}	Chiều cao và chiều rộng của đặc trưng đầu ra (Output Feature Map)
R, S	Chiều cao và chiều rộng của bộ lọc (Kernel Height, Kernel Width)
U	Bước trượt (Stride)
P	Kích thước vùng đệm (Padding)
I	Tensor dữ liệu đầu vào
O	Tensor dữ liệu đầu ra
W	Tensor trọng số (Weights)
B	Vector hệ số chệch (Bias)
O_{dw}	Đầu ra của lớp Depthwise Convolution
O_{pw}	Đầu ra của lớp Pointwise Convolution

Ký hiệu	**Ý nghĩa**
μ	Giá trị trung bình (Mean) trong Batch Normalization
σ	Phương sai (Variance) trong Batch Normalization
γ	Tham số tỉ lệ (Scale factor)
β	Tham số dịch chuyển (Shift factor)
ϵ	Hằng số Epsilon

Viết tắt	**Ý nghĩa**
AI	Trí tuệ nhân tạo (Artificial Intelligence)
SoC	Hệ thống trên chip (System-on-Chip)
FPGA	Mảng cổng lập trình được dạng trường (Field-Programmable Gate Array)
CNN	Mạng nơ-ron tích chập (Convolutional Neural Network)
DNN	Mạng nơ-ron sâu (Deep Neural Network)
RTL	Mức chuyển giao thanh ghi (Register Transfer Level)
IP	Sở hữu trí tuệ (Intellectual Property - Khối thiết kế phần cứng)
PE	Phần tử xử lý (Processing Element)
MAC	Phép tính Nhân-Cộng tích lũy (Multiply-Accumulate)
DMA	Truy cập bộ nhớ trực tiếp (Direct Memory Access)
AXI	Giao diện mở rộng nâng cao (Advanced eXtensible Interface - chuẩn AMBA)
BRAM	Block RAM (Bộ nhớ nội trên FPGA)
DSP	Digital Signal Processing (Khối xử lý tín hiệu số trên FPGA)
LUT	Bảng tra (Look-Up Table)
FF	Flip-Flop

Chapter 1

Giới thiệu đề tài

Chương này trình bày tổng quan về bối cảnh nghiên cứu, xác định mục tiêu cụ thể, phạm vi thực hiện và phân công nhiệm vụ giữa các thành viên trong nhóm.

1.1 Đặt vấn đề

1.2 Mục tiêu đề tài

1.3 Phạm vi đề tài

1.4 Phân chia công việc

1.5 Cấu trúc báo cáo

Chapter 2

Cơ sở lý thuyết

Chương này cung cấp các kiến thức nền tảng về Mạng nơ-ron tích chập (CNN), các kỹ thuật thiết kế phần cứng cho AI và kiến trúc System-on-Chip trên FPGA.

2.1 Tổng quan về Mạng nơ-ron tích chập (CNN)

2.2 Kỹ thuật thiết kế bộ tăng tốc phần cứng (AI Accelerator)

2.3 Kiến trúc System-on-Chip trên nền tảng FPGA

Chapter 3

Phân tích yêu cầu và Kiến trúc tổng quan

Chương này phân tích các ràng buộc thiết kế từ đó đề xuất kiến trúc tổng thể của hệ thống SoC tích hợp AI Accelerator.

3.1 Phân tích yêu cầu thiết kế

3.2 Kiến trúc hệ thống tổng quan

3.3 Đặc tả giao diện kết nối

Chapter 4

Thiết kế kiến trúc Accelerator và Chiến lược dòng dữ liệu

Chương này trình bày chi tiết thiết kế của lõi IP Accelerator, bao gồm phân tích toán học, chiến lược tối ưu dòng dữ liệu và kiến trúc vi mô.

4.1 Phân tích toán học của phép tính tích chập

Để đảm bảo tính linh hoạt cho kiến trúc phần cứng, giúp hệ thống có khả năng hỗ trợ đa dạng các mô hình mạng nơ-ron từ kinh điển (như VGG16) đến các mô hình tối ưu cho thiết bị biên (như MobileNet), nhóm thực hiện đề tài đã tập trung phân tích đặc tả toán học của hai loại phép tính cốt lõi: **Standard Convolution** và **Depthwise Separable Convolution**. Việc hiểu rõ bản chất toán học và cấu trúc dữ liệu của các phép tính này (bao gồm cả cơ chế xử lý biên - Padding) là cơ sở quan trọng để chúng tôi thiết kế nên một kiến trúc thống nhất (Unified Architecture).

4.1.1 Standard Convolution (Tích chập tiêu chuẩn)

Đây là phép tính nền tảng trong hầu hết các mạng CNN truyền thống. Về mặt toán học, tích chập tiêu chuẩn thực hiện việc trượt bộ lọc (filter) trên không gian đầu vào (H, W), đồng thời tích lũy giá trị qua toàn bộ chiều sâu của kênh (Channels).

4.1.1.1 Công thức toán học tổng quát

Xét một lớp tích chập với đầu vào I có kích thước $C \times H_{in} \times W_{in}$ và bộ trọng số W có kích thước $M \times C \times R \times S$. Tham số Padding (P) được sử dụng để giữ nguyên kích thước không gian hoặc kiểm soát việc giảm kích thước. Giá trị đầu ra O tại kênh m, vị trí (h, w) được xác định bởi:

$$O[m][h][w] = B[m] + \sum_{c=0}^{C-1} \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} I[c][h \cdot U + r - P][w \cdot U + s - P] \times W[m][c][r][s] \tag{4.1}$$

Trong đó:

- U: Bước trượt (Stride).
- P: Số lượng điểm ảnh đệm thêm vào mỗi cạnh (Padding).
- Điều kiện biên: Nếu chỉ số truy cập I nằm ngoài phạm vi $[0, H_{in} - 1]$ hoặc $[0, W_{in} - 1]$, giá trị trả về là 0 (Zero-padding).

4.1.1.2 Cấu trúc vòng lặp (Loop Nest)

Với giả thiết kích thước batch $N = 1$, chúng tôi mô hình hóa phép tính này dưới dạng 6 vòng lặp lồng nhau. Việc xử lý Padding thường được thực hiện bằng phần cứng chuyên dụng (Padding Logic) để tránh truy cập bộ nhớ ngoài vùng cho phép.

Algorithm 1: Standard Convolution (Standard Conv2D)

Input: $I[C][H_{in}][W_{in}]$, $W[M][C][R][S]$, Padding P, Stride U

Output: $O[M][H_{out}][W_{out}]$

for $m = 0$ ***to*** $M - 1$ **do**
  for $c = 0$ ***to*** $C - 1$ **do**
    for $h = 0$ ***to*** $H_{out} - 1$ **do**
      for $w = 0$ ***to*** $W_{out} - 1$ **do**
        for $r = 0$ ***to*** $R - 1$ **do**
          for $s = 0$ ***to*** $S - 1$ **do**
            $h_{in} = h \cdot U + r - P$
            $w_{in} = w \cdot U + s - P$
            if $h_{in} \geq 0 \wedge h_{in} < H_{in} \wedge w_{in} \geq 0 \wedge w_{in} < W_{in}$ **then**
              $val = I[c][h_{in}][w_{in}]$
            else
              $val = 0$ `/* Zero Padding */`
            end
            $O[m][h][w] \leftarrow O[m][h][w] + val \times W[m][c][r][s]$
          end
        end
      end
    end
  end
end

4.1.2 Depthwise Separable Convolution

Để giảm chi phí tính toán cho các thiết bị biên, các mô hình như MobileNet sử dụng kỹ thuật **Depthwise Separable Convolution**, tách phép chập chuẩn thành hai bước: **Depthwise (DW)** và **Pointwise (PW)**.

4.1.2.1 Depthwise Convolution (DW)

Phép tính này áp dụng bộ lọc riêng cho từng kênh đầu vào. Công thức tính toán bao gồm tham số Padding như sau:

$$O_{dw}[c][h][w] = \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} I[c][h \cdot U + r - P][w \cdot U + s - P] \times W_{dw}[c][r][s] \quad (4.2)$$

Nhận xét: Việc xử lý Padding trong Depthwise cũng tương tự như Standard Conv, tuy nhiên do tính độc lập giữa các kênh, bộ điều khiển (Controller) cần đảm bảo logic Padding hoạt động chính xác cho từng luồng tính toán song song.

Algorithm 2: Depthwise Convolution (với Padding)

for $c = 0$ ***to*** $C - 1$ `/* Parallel Channels */`
do
  for $h = 0$ ***to*** $H_{out} - 1$ **do**
    for $w = 0$ ***to*** $W_{out} - 1$ **do**
      for $r = 0$ ***to*** $R - 1$ **do**
        for $s = 0$ ***to*** $S - 1$ **do**
          $h_{in} = h \cdot U + r - P$
          $w_{in} = w \cdot U + s - P$
          if $h_{in} \in [0, H_{in}) \wedge w_{in} \in [0, W_{in})$ **then**
            $O_{dw}[c][h][w] += I[c][h_{in}][w_{in}] \times W_{dw}[c][r][s]$
          end
        end
      end
    end
  end
end

4.1.2.2 Pointwise Convolution (PW)

Pointwise Convolution là tích chập chuẩn với kernel 1×1. Do kích thước kernel là 1×1, tham số Padding thường được đặt bằng 0 ($P = 0$) và Stride $U = 1$ để giữ nguyên kích thước không gian (H, W), chỉ thay đổi số kênh từ C sang M.

$$O_{pw}[m][h][w] = \sum_{c=0}^{C-1} I[c][h][w] \times W_{pw}[m][c] \tag{4.3}$$

4.1.3 Yêu cầu đối với Kiến trúc thống nhất (Unified Architecture)

Từ các phân tích trên, nhóm nhận thấy rằng để bộ tăng tốc hoạt động hiệu quả cho cả hai trường hợp, kiến trúc phần cứng cần giải quyết được bài toán "kép":

1. **Cơ chế xử lý Padding động:** Phần cứng cần có khối logic để tự động chèn giá trị 0 khi chỉ số tính toán $(h \cdot U + r - P)$ bị âm hoặc vượt quá kích thước ảnh, thay vì phải tốn tài nguyên bộ nhớ để lưu trữ các viền số 0 thực tế.

2. **Tính linh hoạt của Mảng PE:** Các đơn vị tính toán cần có khả năng chuyển đổi chế độ giữa tích lũy theo không gian (Standard/Pointwise) và tính toán độc lập theo kênh (Depthwise).

4.1.4 Kỹ thuật Gập Batch Normalization (BN Folding)

Trong các mạng CNN hiện đại như MobileNet, lớp Batch Normalization (BN) thường được đặt ngay sau lớp Convolution để chuẩn hóa phân phối dữ liệu, giúp mạng hội tụ nhanh hơn. Công thức tính toán của lớp BN trong quá trình suy luận (Inference) cho một kênh m là:

$$y = \frac{x - \mu_m}{\sqrt{\sigma_m^2 + \epsilon}} \cdot \gamma_m + \beta_m \tag{4.4}$$

Trong đó:

- x: Giá trị đầu ra từ lớp Convolution (trước khi qua hàm kích hoạt).
- μ_m, σ_m: Giá trị trung bình (mean) và phương sai (variance) động (running statistics) của kênh m.
- γ_m, β_m: Tham số tỉ lệ (scale) và dịch chuyển (shift) được học trong quá trình huấn luyện.
- ϵ: Hằng số nhỏ để tránh chia cho 0.

Việc thực hiện trực tiếp công thức này trên phần cứng rất tốn kém do yêu cầu các phép toán phức tạp như căn bậc hai và phép chia. Tuy nhiên, do tại thời điểm suy luận, các tham số $\mu, \sigma, \gamma, \beta$ đều là hằng số, chúng tôi áp dụng kỹ thuật **BN Folding** để gộp toàn bộ phép tính BN vào trong trọng số (W) và bias (B) của lớp Convolution đi trước nó.

4.1.4.1 Công thức biến đổi trọng số

Giả sử đầu ra của lớp Convolution là $x = W_{orig} \cdot Input + B_{orig}$. Khi thay vào công thức BN, ta có:

$$y = \frac{(W_{orig} \cdot Input + B_{orig}) - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta \tag{4.5}$$

Phương trình trên có thể được viết lại dưới dạng một phép Convolution mới với trọng số W' và bias B':

$$y = W' \cdot Input + B' \tag{4.6}$$

Trong đó, các tham số mới được tính toán trước (offline) bởi phần mềm

(driver) trước khi nạp xuống phần cứng:

$$W'[m][c][r][s] = W_{orig}[m][c][r][s] \cdot \frac{\gamma_m}{\sqrt{\sigma_m^2 + \epsilon}} \tag{4.7}$$

$$B'[m] = \left(B_{orig}[m] - \mu_m\right) \cdot \frac{\gamma_m}{\sqrt{\sigma_m^2 + \epsilon}} + \beta_m \tag{4.8}$$

Kết luận thiết kế: Nhờ kỹ thuật BN Folding, kiến trúc phần cứng của chúng tôi **không cần** thiết kế khối chức năng riêng cho Batch Normalization. Accelerator chỉ cần thực hiện phép tính Convolution bình thường với bộ trọng số (W', B') đã được tinh chỉnh, giúp tiết kiệm đáng kể tài nguyên DSP và giảm độ trễ xử lý.

4.2 Chiến lược phân mảnh và Dòng dữ liệu đề xuất

4.3 Chiến lược phân mảnh và Dòng dữ liệu đề xuất

Dựa trên phân tích về tài nguyên phần cứng và đặc thù của các lớp tích chập (Convolution), nhóm đề xuất chiến lược quản lý dữ liệu dựa trên kỹ thuật **Ping-Pong Buffering** (Double Buffering) mở rộng. Chiến lược này không chỉ giúp che giấu độ trễ truy cập bộ nhớ mà còn giải quyết hiệu quả bài toán tính toán tại biên (boundary computation) mà không cần bộ nhớ đệm dòng (Line Buffer) riêng biệt.

4.3.1 Chiến lược quản lý bộ nhớ Ping-Pong và Xử lý biên

Thay vì sử dụng một bộ đệm dòng chuyên dụng để lưu trữ dữ liệu đầu vào chồng lấn, hệ thống tận dụng chính cơ chế Ping-Pong của Output Buffer

để lưu trữ và tích lũy các kết quả trung gian tại biên (Boundary Partial Sums).

Hệ thống sử dụng hai bộ đệm đầu ra (Output Buffers): **Buffer A** và **Buffer B** hoạt động luân phiên:

- **Cơ chế Tích lũy chéo (Cross-Accumulation):** Khi hệ thống đang tính toán cho vùng dữ liệu hiện tại (Tile k) và lưu kết quả vào Buffer A, các giá trị tính toán thuộc vùng biên dưới ("giá trị dôi ra" do kích thước bộ lọc R) sẽ được lưu trực tiếp vào các hàng đầu tiên của Buffer B.

- **Trạng thái khởi tạo tự động:** Khi hệ thống chuyển sang xử lý vùng dữ liệu tiếp theo (Tile $k+1$) với Buffer B làm bộ đệm chính, các hàng đầu tiên của Buffer B đã chứa sẵn các giá trị tổng riêng (Partial Sums) cần thiết từ bước trước đó. Việc tính toán tiếp tục được cộng dồn (accumulate) vào các giá trị này.

- **Điều kiện chuyển đổi:** Buffer A chỉ được giải phóng (ghi xuống DRAM) khi đã hoàn thành tích lũy đủ T_h hàng của toàn bộ C kênh đầu vào (với Standard Conv) hoặc T_c kênh (với Depthwise Conv). Hệ thống phải đảm bảo Buffer A đã được giải phóng (trống) trước khi Buffer B cần ghi đè vùng biên ("dôi ra") của Tile $k+1$ vào Buffer A (để chuẩn bị cho Tile $k+2$).

4.3.2 Tham số phân mảnh dữ liệu (Tiling Parameters)

Không gian tính toán được chia nhỏ dựa trên các tham số cấu hình phần cứng như sau:

- W, H: Chiều rộng và chiều cao của Input Feature Map (IFM).

- T_h: Chiều cao của một mảnh IFM (Input Tile Height) được xử lý trong một lần nạp. Mảnh cuối cùng có chiều cao $H\%T_h$.
- T_c: Số kênh đầu vào (Input Channels) được xử lý song song trong 1 pass (k_i).
- T_m: Số kênh đầu ra (Output Channels) được tính toán song song (m_i).
- R, S: Kích thước nhân chập (Kernel Height, Width).

4.3.3 Cấu trúc Dòng dữ liệu (Dataflow Description)

Quy trình thực hiện được mô tả chi tiết cho hai loại tích chập chính:

4.3.3.1 Standard Convolution

Trong Standard Convolution, mỗi kênh đầu ra là tổng hợp từ tất cả C kênh đầu vào.

- **Đơn vị xử lý (1 Pass):** Trong mỗi pass, hệ thống nạp T_c kênh đầu vào và thực hiện tích chập với T_m bộ lọc tương ứng để tạo ra T_m mảnh Output Feature Map (OFM).
- **Tích lũy kênh (Channel Accumulation):** Để hoàn thành một mảnh OFM (gồm T_m kênh), hệ thống cần thực hiện $\lceil C/T_c \rceil$ passes. Các kết quả từ các pass này được cộng dồn vào Buffer hiện tại (A hoặc B).
- **Hoàn thành toàn bộ OFM:**
  - Để hoàn thành T_m kênh OFM với chiều cao H: Cần $\lceil C/T_c \rceil \times \lceil H/T_h \rceil$ passes.
  - Để hoàn thành toàn bộ M kênh OFM: Cần $\lceil C/T_c \rceil \times \lceil H/T_h \rceil \times \lceil M/T_m \rceil$ passes.

- **Quản lý trọng số (Weight Management):** Trọng số không được nạp theo từng pass mà được nạp và tích lũy trong bộ nhớ on-chip (Weight Buffer) cho đến khi đầy giới hạn cho phép. Trọng số chỉ bị xóa (evicted) khi toàn bộ các tính toán liên quan đến bộ lọc đó đã hoàn tất, giúp tối đa hóa tái sử dụng trọng số (Weight Reuse).

4.3.3.2 Depthwise Convolution

Trong Depthwise Convolution, mỗi kênh đầu vào tương ứng duy nhất với một kênh đầu ra ($C = M$).

- **Đơn vị xử lý (1 Pass):** Một pass tính toán xong sẽ hoàn thành luôn T_m mảnh OFM (tương ứng với T_c mảnh IFM, với $T_m = T_c$). Không cần vòng lặp tích lũy kênh đầu vào.
- **Hoàn thành toàn bộ OFM:**
  - Để hoàn thành T_m kênh OFM với chiều cao H: Cần $\lceil H/T_h \rceil$ passes.
  - Để hoàn thành toàn bộ M kênh OFM: Cần $\lceil H/T_h \rceil \times \lceil M/T_m \rceil$ passes.

4.3.4 Thuật toán và Loop Nest

Mã giả dưới đây mô tả chi tiết hoạt động của bộ điều khiển (Controller) và cơ chế Ping-Pong Buffer:

Algorithm 3: Dataflow với cơ chế Ping-Pong Accumulation

Input: Input Tiles, Weights

Output: Output Feature Map

Initialize $Buffer\_Current = A$, $Buffer\_Next = B$

for $m_{blk} = 0$ ***to*** $\lceil M/T_m \rceil$ **do**

  Load Weights into On-chip Memory

  for $h_{blk} = 0$ ***to*** $\lceil H/T_h \rceil$ **do**

    for $c_{blk} = 0$ ***to*** $\lceil C/T_c \rceil$ **do**

      Load Input Tile ($T_c \times T_h$)

      Compute Convolution

      if *pixel* $\in$ *Main Body* (T_h *rows)* **then**

        Accumulate to $Buffer\_Current$

      else

        Accumulate to $Buffer\_Next$ (start rows)

      end

    end

    if *Standard Convolution OR* (h_{blk} *done in Depthwise)* **then**

      Drain $Buffer\_Current$ to DRAM (Off-chip)

      Swap($Buffer\_Current$, $Buffer\_Next$)

      Clear $Buffer\_Next$ (prepare for future overlap)

    end

  end

end

Chiến lược này giúp giảm thiểu dung lượng bộ nhớ on-chip cần thiết vì mỗi thời điểm chỉ cần chứa tối đa dữ liệu cho 2 passes (Current và Next), đồng thời loại bỏ hoàn toàn việc nạp lại dữ liệu IFM chồng lấn từ bộ nhớ ngoài.

4.4 Thiết kế kiến trúc vi mô (Micro-architecture)

4.4.1 Kiến trúc Mảng tính toán (PE Array Architecture)

4.4.1.1 Cấu hình PE cho Tích chập Không gian (Spatial Convolution)

Mảng tính toán được tổ chức để ánh xạ trực tiếp chiều cao R của bộ lọc lên phần cứng. Hệ thống sử dụng một nhóm gồm R phần tử xử lý (PE) hoạt động phối hợp (Cooperative PEs) để tính toán cho một hàng đầu ra (Output Row).

- **Phân bố dữ liệu:** Mỗi PE trong nhóm R này chịu trách nhiệm lưu trữ và tính toán cho một hàng trọng số (Weight Row) của bộ lọc kích thước $R \times S$.
- **Cơ chế trượt (Sliding Mechanism):** Các PE cùng trượt trên các hàng dữ liệu đầu vào tương ứng. Kết quả từ R PE này được cộng dồn (Reduction) để tạo ra một điểm pixel đầu ra hoàn chỉnh.
- **Tổng số PE:** Để hỗ trợ tính toán song song cho T_c kênh đầu vào và T_m kênh đầu ra, tổng số PE của hệ thống là:

$$N_{PE} = R \times T_c \times T_m \tag{4.9}$$

4.4.1.2 Hỗ trợ Depthwise Convolution

Trong chế độ Depthwise, vì mỗi kênh đầu vào chỉ tương tác với một bộ lọc duy nhất (T_c input channels $\leftrightarrow$ T_c output channels), kiến trúc phần cứng tự động cấu hình lại đường dẫn dữ liệu:

- Nhóm R PE vẫn hoạt động như cũ để xử lý không gian.

- Tuy nhiên, T_c nhóm PE (vốn dùng để cộng dồn kênh trong Standard Conv) giờ đây sẽ hoạt động độc lập, mỗi nhóm xử lý một kênh riêng biệt (Channel-Parallelism).
- Điều này đảm bảo không có PE nào bị lãng phí (idle) khi chuyển từ Standard sang Depthwise, giải quyết triệt để vấn đề hiệu suất thấp thường gặp ở các kiến trúc systolic array truyền thống.

Chapter 5

Hiện thực nền tảng SoC

Chương này trình bày quá trình xây dựng hệ thống SoC cơ sở trên FPGA, bao gồm cấu hình vi xử lý, hệ thống bus và tích hợp các ngoại vi.

5.1 Môi trường và Công cụ hiện thực

5.2 Cấu hình hệ thống xử lý (Processing System)

5.3 Thiết kế hệ thống kết nối (Interconnect Subsystem)

5.4 Tích hợp và Kiểm thử nền tảng cơ sở

Chapter 6

Đánh giá hiệu năng lý thuyết

Chương này sử dụng các mô hình giải tích để ước lượng hiệu năng, độ trễ và tài nguyên tiêu thụ của kiến trúc đề xuất.

6.1 Phương pháp đánh giá: Mô hình Roofline

6.2 Ước lượng độ trễ và Tài nguyên

6.3 So sánh với các nghiên cứu liên quan

Chapter 7

Kế hoạch phát triển

Chương này tổng kết các kết quả đạt được trong Giai đoạn 1 và đề ra kế hoạch chi tiết cho việc hiện thực và kiểm thử trong Giai đoạn 2.

7.1 Đánh giá mức độ hoàn thành Giai đoạn 1

7.2 Kế hoạch thực hiện Giai đoạn 2

7.3 Tiến độ dự kiến