**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP HỒ CHÍ MINH**

**KHOA ĐIỆN – ĐIỆN TỬ**

---

# ĐỒ ÁN TỐT NGHIỆP

**NGÀNH CNKT ĐIỆN TỬ - VIỄN THÔNG**

# NÂNG CAO ĐỘ PHÂN GIẢI HÌNH ẢNH BẰNG TRÍ TUỆ NHÂN TẠO TRÊN XILINX ZYNQ ULTRASCALE+ MPSOC – ZCU102

**SVTH : HÀ NGUYỄN MINH PHƯỚC**

**MSSV : 20161245**

**Lớp : 20161CLVT2B**

Tp. Hồ Chí Minh, tháng 06 năm 2024

# CHƯƠNG 1: TỔNG QUAN

## 1.1 Tính cấp thiết của đề tài

Trong thời đại số hóa và phát triển công nghệ, hình ảnh đóng vai trò quan trọng trong giao tiếp và truyền tải thông tin. Chất lượng ảnh không chỉ ảnh hưởng đến trải nghiệm người dùng mà còn có ảnh hưởng đáng kể đến nhiều lĩnh vực khác nhau như y tế, an ninh, công nghiệp và nghệ thuật.

Ngoài ra, việc có được hình ảnh rõ ràng và chi tiết cũng là yếu tố quan trọng trong công nghệ, đặc biệt là trong lĩnh vực xe tự lái, an ninh và giám sát. Hình ảnh độ phân giải cao giúp hệ thống phát hiện và nhận dạng đối tượng một cách chính xác và nhanh chóng, từ đó nâng cao độ tin cậy và hiệu suất của các ứng dụng này.

Tóm lại, trong một xã hội ngày càng phụ thuộc vào công nghệ và truyền thông hình ảnh, tính cấp thiết của việc độ phân giải hình ảnh không chỉ là vấn đề của cá nhân mà còn là yếu tố quyết định sự thành công và phát triển của các ngành công nghiệp và cả xã hội.

## 1.2 Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đề tài là áp dụng các phương pháp và thuật toán học sâu (deep learning) để tăng cường chất lượng ảnh và cải thiện khả năng xử lý ảnh trên nền tảng phần cứng SoC. Nâng cao độ phân giải ảnh: Sử dụng deep learning để tăng độ phân giải của ảnh thông qua các thuật toán như Super-Resolution của các thư viện Pytorch hay Tensorflow. Mục tiêu là tạo ra các ảnh có độ chi tiết cao hơn và rõ nét hơn, đặc biệt là khi làm việc với ảnh có độ phân giải thấp.

## 1.3 Tình hình nghiên cứu hiện nay

### 1.3.1. Tình hình nghiên cứu trong nước

Bài luận văn tốt nghiệp của tác giả Nguyễn Hoàng Nhật Uyên về đề tài Triển khai bộ lọc Sobel trên Xilinx Zynq-7000 vào năm 2022 [1] là một đề tài nổi bật cho việc ứng dụng thiết kế trên SOC. Đề tài đạt được thành tựu đó là triển khai và ứng dụng được bộ lọc Sobel cho các hình ảnh trên Zynq-7000 và so sánh thông số ngõ ra của ảnh hình ảnh đã xử lý như PSRN, SSIM,... trên phần cứng Zynq-7000 và phần mềm OpenCV.

### 1.3.2. Tình hình nghiên cứu ngoài nước

Hệ thống siêu phân giải ảnh dùng mạng GAN của Keras triển khai trên Zynq Ultrascale+ MPSOC ZCU102 của tác giả Gokula Krishnan Ravi [2]. Tác giả đã sử dụng mạng GAN của Keras tiến hành huấn luyện, lượng tử hóa và biên dịch mô hình trên phần mềm Vitis AI và triển khai trên Zynq Ultrascale+ MPSOC - ZCU102. Thiết kế của tác giả có nhiều điểm nổi bật có thể phân giải ảnh ở nhiều patch size khác nhau, từ đó có thể suy ra hiệu suất của DPU trên ZCU102. Tuy nhiên, thiết

kế có một số điểm hạn chế như là mô hình siêu phân giải GAN không được tích hợp trên Vitis AI nên việc triển khai và thực thi ứng dụng rất khó khăn cho người mới bắt đầu.

## 1.4 Đối tượng và phạm vi nghiên cứu

### 1.4.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là áp dụng mạng học sâu (Deep Learning) và tận dụng khả năng tính toán của bộ xử lý học sâu (Deep Learning Processor Unit - DPU) trong SoC để nâng cao độ phân giải hình ảnh.

### 1.4.2 Phạm vi nghiên cứu

Tìm hiểu về chức năng và kết nối các IP cần thiết giúp giao tiếp giữa bộ xử lý học sâu với ZYNQ Ultrascale+ MPSoC. Xây dựng hệ điều hành Linux chạy trên phần cứng. Sử dụng mô hình mạng học sâu cho xử lý hình ảnh RCAN (Residual Channel Attention Networks) của thư viện Pytorch được tích hợp trên Vitis-AI Model Zoo. Tiến hành huấn luyện, lượng tử hóa, biên dịch model và triển khai trên Xilinx ZYNQ Ultrascale+ MPSoC-ZCU102.

## 1.5 Bố cục đồ án tốt nghiệp

**Chương 1:** Giới thiệu: Trình bày tính cấp thiết và lý do lựa chọn đề tài, ý nghĩa khoa học và thực tiễn đạt được, mục tiêu nghiên cứu, đối tượng, phạm vi và giới hạn của đề tài, phương pháp nghiên cứu, bố cục của đồ án và phân định nội dung của vấn đề được nghiên cứu.

**Chương 2:** Cơ sở lý thuyết: Trình bày lý thuyết của mạng RCAN và thông tin phần cứng ZCU102.

**Chương 3:** Thiết kế hệ thống: Trình bày sơ đồ thiết kế của hệ thống và giải thích các phương pháp xử lý dữ liệu. Cuối cùng, thực hiện chạy mạng RCAN trên ZCU102.

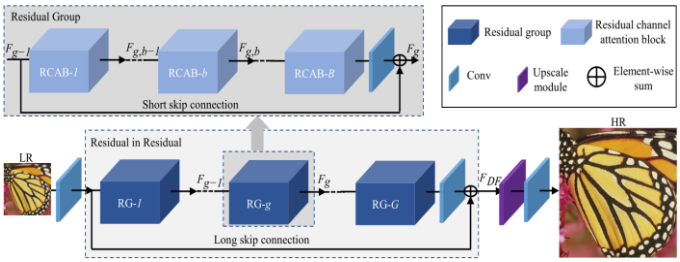
**Chương 4:** Kết quả: Trình bày chi tiết kết quả của từng khối đã được thực hiện. So sánh thông số của ảnh RCAN chạy trên ZCU102 và trên Kaggle.

**Chương 5:** Kết luận và hướng phát triển: Đưa ra kết luận về những nội dung đã đạt được và những điểm còn chưa hoàn thiện của đề tài. Từ đó, đề xuất hướng phát triển tiếp theo nhằm giúp cải thiện hệ thống một cách tốt hơn và ứng dụng vào thực tiễn hiệu quả.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1. Tổng quan về Residual Channel Attention Networks (RCAN)

### 2.1.1. Kiến trúc mô hình (Network Architecture)




Trong hình 2.1, RCAN chủ yếu bao gồm bốn phần: trích xuất đặc trưng cơ bản (shallow feature extraction), trích xuất đặc trưng sâu bằng cách dùng cấu trúc Residual in Residual (RIR), mô-đun tăng kích thước, và phần tái tạo.

***Hình 2.1.*** *Sơ đồ kiến trúc mạng RCAN*

### 2.1.2. Trích xuất các đặc trưng cơ bản

Đây là bước đầu tiên của quá trình siêu phân giải ảnh gồm một lớp tích chập có kích thước kernel là 3x3 hoặc 3x5. [6] Đây là quá trình trích xuất các đặc trưng cơ bản từ dữ liệu đầu vào một cách đơn giản, đặc trưng này thường không chứa nhiều thông tin phức tạp mà chỉ là những đặc điểm cơ bản của dữ liệu. Công thức được mô tả ở dưới đây:

$$F_0 = H_{SF}(I_{LR}) \tag{2.1}$$

$F_0$ là ngõ ra của phép chập.$H_{SF}$ biểu thị cho phép toán chập. $I_{LR}$ là ảnh ngõ vào có độ phân giải thấp.

### 2.1.3. Trích xuất các đặc trưng sâu bằng cơ chế Residual in Residual (RIR)

Ngõ ra F0 của bài toán trích xuất đặc trưng cơ bản được sử dụng cho bước trích xuất đặc trưng sâu RIR. [6] Công thức được mô tả ở dưới đây:

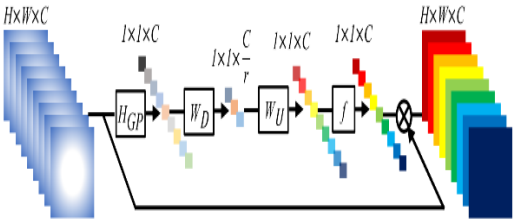
$$F_{DF} = H_{RIR}(F_0) \tag{2.2}$$

Trong đó, $H_{RIR}$ biểu thị phần dư (residual) rất sâu chứa các nhóm dư G (RG). Mỗi RG chưa B khối chú ý kênh dư RCAB (Residual Channel Attention Blocks). [6] Cấu trúc RIR cho phép huấn luyện CNN rất sâu (trên 400 lớp) cho hình ảnh có hiệu suất cao. Một RG (Residual Group) trong nhóm g-th được xây dựng như sau:

$$F_g = H_g(F_{g-1}) = H_g(H_{g-1}(\dots H_1(F_0)\dots)) \tag{2.3}$$

Hg biểu thị hàm của nhóm dư thứ g. Fg-1 và Fg lần lượt là ngõ vào và ngõ ra của nhóm dư thứ g.

### 2.1.4. Channel Attention (CA)




***Hình 2.2.*** *Kiến trúc Channel Attention của mạng RCAN*

Để làm cho mạng tập trung vào các đặc trưng nổi bật, cần phải khai thác sự tương phản giữa các kênh đặc trưng. Dùng lớp Global Average Pooling được thể hiện trong hình 2.3, giả sử X = [$x_1$, · · · , $x_c$, · · · , $x_C$ ] là ngõ đầu vào, có C bản đồ đặc trưng (feature map) với kích thước H × W.

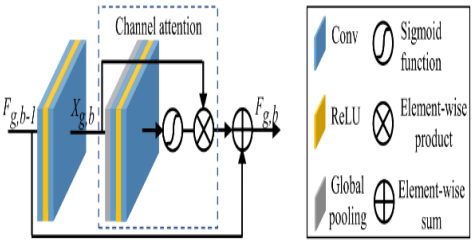
Thống kê theo kênh z ∈ $R^C$ có thể được tính được bằng cách thu gọn X qua các chiều không gian H ×W. Sau đó, phần tử thứ c của z được xác định bởi công thức: [6]

$$z_c = H_{GP}(x_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i,j) \quad (2.4)$$

Trong đó $x_c(i, j)$ là giá trị tại vị trí (i, j) của đặc trưng thứ c của $x_c$. HGP (·) biểu thị hàm tổng hợp toàn cục. Thống kê kênh như vậy có thể được coi là một tập hợp các mô tả cục bộ.

### 2.1.5. Residual Channel Attention Bock (RCAB)

Các nhóm dư thừa (residual groups) cho phép các phần chính của mạng tập trung vào các thành phần thông tin hơn của các đặc trưng của LR. Tích hợp CA vào RB ta được RCAB.




***Hình 2.3.*** *Kiến trúc RCAB của mạng RCAN*

Đối với RB thứ b trong RG thứ g: [6]

$$F_{g,b} = F_{g,b} - 1 + R_{g,b}(X_{g,b}) . X_{g,b} \quad (2.5)$$

Trong đó, $R_{g,b}$ biểu thị chức năng của CA. $F_{g,b}$ và $F_{g,b-1}$ là ngõ vào và ngõ ra của RCAB. $F_{g,b-1}$ được học từ $X_{g,b-1}$ là kết quả của 2 lớp tích chập chồng lên nhau. [6]

$$X_{g,b-1} = W^2_{g,b}\, \delta(W^{-1}_{g,b}\, F_{g,b-1}) \quad (2.6)$$

Trong đó $W^{-1}_{g,b}$ và $W^2_{g,b}$ là trọng số của 2 lớp chập chồng lên nhau ở trong RCAB.

### 2.1.6. Mô-đun tăng kích thước

Dựa vào sơ đồ kiến trúc mạng ở hình 2.4. mô-đun này nằm ở ngõ ra của RIR, sau đó được tăng kích thước bằng công thức: [6]

$$F_{UP} = H_{UP}(F_{DF}) \tag{2.7}$$

Sau khi tăng kích thước, hình ảnh sẽ được cấu trúc lại bằng một lớp tích chập: [6]

$$I_{SR} = H_{REC}(F_{UP}) = H_{RCAN}(I_{LR}) \tag{2.8}$$

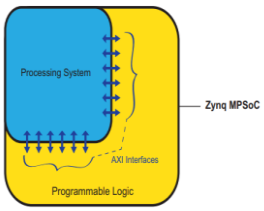
Trong đó HREC và HRCAN lần lượt là lớp tái cấu trúc và chức năng của mạng RCAN. Cuối cùng là tính mất mát của mạng: [6]

$$L(O) = \frac{1}{N}\sum_{i=1}^{N} ||H_{RCAN}(I_{LR}^{i}) - I_{HR}^{i}||_1 \tag{2.9}$$

Thông số của RG được đặt là G=10. Trong mỗi RG, thông số của RCAB được đặt là 20.

## 2.2. Tổng quan về Zynq Ultrascale+ MPSoC - ZCU102

### 2.2.1. Kiến trúc của Zynq Ultrascale+ MPSoC - ZCU102




***Hình 2.4.*** *Sơ đồ kiến trúc tổng quát của ZCU102*

ZCU102 Evaluation Kit có những đặc điểm và tính năng đáng chú ý như :

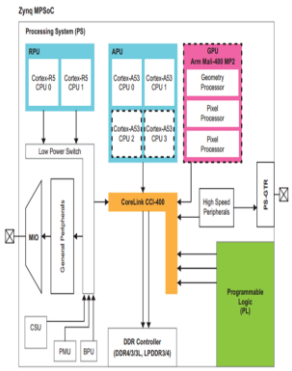
- Hệ thống xử lý (Processing System - PS).
- Logic lập trình (FPGA Programmable Logic PL)

Hệ thống xử lý [7] [8]: Đơn vị xử lý (APU: ARM Cortex-A53 và Cortex-R5), Giao diện bộ nhớ (Memory Interfaces), Thiết bị ngoại vi (I/O Peripherals - IOP), khối kết nối, …

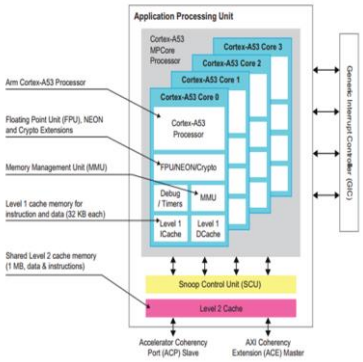
Logic lập trình [7] [8]: Configurable Logic Blocks – CLB, khối RAM, khối xử lý tín hiệu số (DSP48E1), Analog-to-Digital Converter – XADC, bộ nhớ DDR4, giao diện ngoại vi băng thông cao bao gồm 1G Ethernet, USB 3.0, DisplayPort, PCIe Gen2.

### 2.2.2. Đơn vị xử lý (PS) của Zynq Ultrascale+ MPSoC - ZCU102

Hình 2.5 dưới đây cung cấp một cái nhìn tổng quát và đơn giản hóa về kiến trúc Zynq MPSoC [11]. Các đơn vị xử lý APU, RPU, GPU, PL giao tiếp với nhau thông qua một kết nối được gọi là "Kết nối đồng bộ bộ nhớ Cache" (Cache Coherent Interconnect - CCI).




***Hình 2.5.*** *Kiến trúc của ZCU102 [7]*




***Hình 2.6.*** *Sơ đồ khối APU [7]*

CCI là một bổ sung vào kiến trúc PS, cho phép các bộ xử lý chia sẻ tác vụ và dữ liệu một cách linh hoạt. Về cơ bản, CCI được sử dụng để đạt được quá trình xử lý bất đối xứng, tạo sự kết nối, hỗ trợ tính mạch lạc giữa các tác vụ xử lý, và đảm bảo rằng mỗi lõi xử lý đang hoạt động trên dữ liệu mới nhất trong toàn bộ PS.

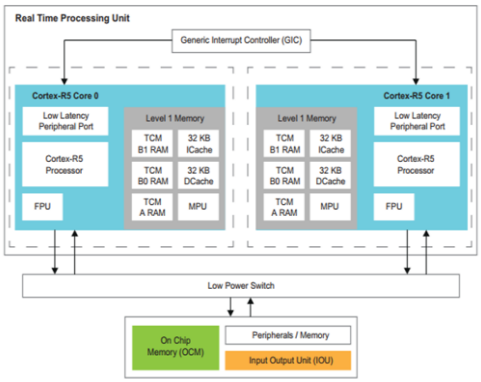
#### 2.2.2.1. Application Processing Unit (APU)

Mỗi lõi Cortex-A53 bao gồm: Đơn vị dấu chấm động (Floating Point Unit - FPU), Động cơ xử lý đa phương tiện (NEON Media Processing Engine - MPE), tiện ích mở rộng mã hóa (Cryptography Extension - Crypto), Đơn vị quản lý bộ nhớ (Memory Management Unit - MMU), và bộ nhớ cache cấp 1 riêng biệt cho mỗi lõi (Hình 2.6). Phần còn lại của APU bao gồm đơn vị kiểm soát Snoop (Snoop Control Unit - SCU) và bộ nhớ cache cấp 2 [11].

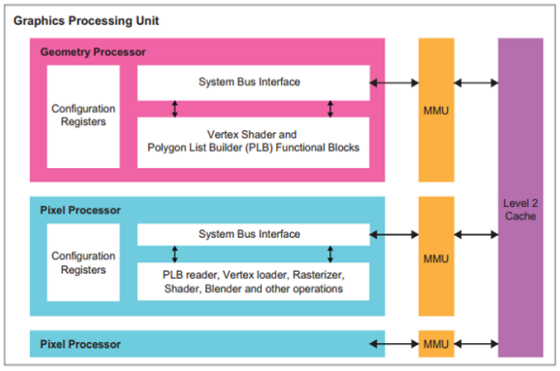
#### 2.2.2.2. Real-Time Processing Unit (RPU)

RPU chứa một bộ vi xử lý kép Arm Cortex-R5, thiết kế đặc biệt cho các ứng dụng thời gian thực. Kiến trúc của RPU mang lại hoạt động có độ trễ thấp và hiệu suất ổn định trong toàn bộ đơn vị.

Lõi Cortex-R5 bao gồm một Đơn vị Dấu chấm động (FPU) cho phép thực hiện các tính toán dấu chấm động chính xác ở cả đơn và đôi. Bộ nhớ cấp 1 của mỗi lõi Cortex-R5 có ba bộ nhớ liên kết chặt chẽ (TCMs), hai bộ nhớ cache 32 KB dành cho dữ liệu và lệnh, cùng với đơn vị bảo vệ bộ nhớ (MPU) . Các lõi Cortex-R5 trong RPU có khả năng hoạt động với tốc độ lên đến 600 MHz. Hình 2.7 minh họa kết nối giữa lõi xử lý Arm Cortex-R5 và công tắc tiết kiệm năng lượng, cùng với kết nối tới các thiết bị ngoại vi khác như Đơn vị Đầu vào/Đầu ra (IOU) và Bộ nhớ Trong Chip (OCM).




***Hình 2.7.*** *Sơ đồ khối của RPU [7]*




***Hình 2.8.*** *Sơ đồ khối của GPU [7]*

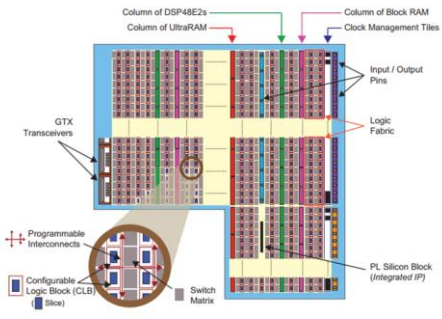
### 2.2.2.3. Graphics Processing Unit (GPU)

Đơn vị xử lý đồ họa Arm Mali-400 MP2 (GPU) bao gồm một bộ xử lý hình học (Geometry Processor - GP) và hai bộ xử lý pixel (Pixel Processors - PP). Đơn vị này sử dụng ba Đơn vị Quản lý Bộ nhớ (Memory Management Units - MMU) tích hợp (một cho mỗi bộ xử lý) và một bộ nhớ cache cấp 2 để lưu trữ dữ liệu tạm thời. [7]

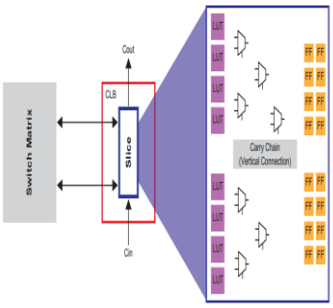
GPU đạt được hiệu suất tăng tốc đồ họa hai chiều (2D) và ba chiều (3D) lên đến 667 MHz. Giao diện lập trình ứng dụng (API) OpenGL ES 1.1/2.0 cho phép xử lý đồ họa phức tạp. Hình 2.8 minh họa bộ xử lý hình học và pixel cùng với các đường truyền thông của GPU [8].

### 2.2.3. Logic lập trình (FPGA Programmable Logic - PL) [7]

PL là một thành phần quan trọng của Zynq MPSoC, vì nó cung cấp tăng tốc phần cứng cho các phép toán số tính toán cường độ cao. PL sử dụng kiến trúc FPGA (Field-Programmable Gate Array) Kintex UltraScale+ 16nm.




***Hình 2.9.*** *Zynq MPSoC Programmable Logic*




***Hình 2.10.*** *Kết nối và các thành phần bên trong của CLB [7] [8]*

#### 2.2.3.1. Cấu trúc Logic cua Programmable Logic

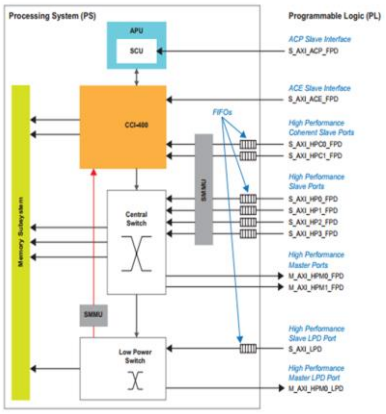
PL bao gồm cấu trúc FPGA Kintex UltraScale+ 16nm, các Slices và Configurable Logic Blocks (CLBs). Sắp xếp của CLBs trong cấu trúc FPGA là các mảng hai chiều. Mỗi CLB được đặt gần một ma trận chuyển mạch để có thể định tuyến dữ liệu đến một tài nguyên PL khác. Ngoài ra, CLBs có thể kết nối với các tài nguyên tương tự khác bằng cách sử dụng liên kết khác.

Một CLB bao gồm một slice, chứa đựng các tài nguyên cần thiết để triển khai mạch logic tuần tự và logic tổ hợp (Hình 2.10). Mỗi phần slice trong các thiết bị Zynq MPSoC được cấu thành từ 8x6-input Lookup Tables (LUTs), 16 Flip-Flops (FFs) và logic định tuyến bổ sung.

Cấu trúc logic FPGA có thể được sử dụng để xây dựng các mạch toán học khác nhau như cộng, nhân và chia. Các mạch này có thể được kết hợp để thực hiện các hàm toán học tùy ý. Lookup Tables (LUTs) có thể được sử dụng như bộ nhớ cục bộ nhỏ, hoặc được kết nối với nhau để tạo thành một mảng bộ nhớ lớn hơn. Cấu trúc này được biết đến là RAM phân tán (Distributed RAM).

### 2.2.4. Giao tiếp giữa hệ thống xử lý (PS) và Logic lập trình (PL)

Chuẩn giao diện chính được sử dụng trong PL để truyền dữ liệu giữa các khối IP là AXI. Có thể kết nối nhiều cổng AXI trong PL từ các nguồn khác nhau bằng cách sử dụng một AXI interconnect. AXI interconnect hoạt động như một công tắc trong cấu trúc logic, để PL có thể định tuyến lưu lượng từ nhiều nguồn đến các đích đến của chúng.AXI hỗ trợ chuyển dữ liệu với công suất truyền tải cao và độ trễ thấp




***Hình 2.11.*** *Sơ* đồ khối kết nối giữa PS và PL

Hình 2.13 hiển thị các tài nguyên trong PS kết nối trực tiếp với PL. PS bao gồm CCI-400, APU (và SCU), đơn vị quản lý bộ nhớ hệ thống (SMMU), công tắc trung tâm và công tắc tiết kiệm điện. Hệ thống bộ nhớ cũng đã được bao gồm để minh họa các tài nguyên bộ nhớ khác nhau trong PS mà mỗi công tắc và liên kết có thể truy cập. SMMU cung cấp chuyển đổi địa chỉ giữa PS và PL, để PL có thể sử dụng địa chỉ ảo.

# CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

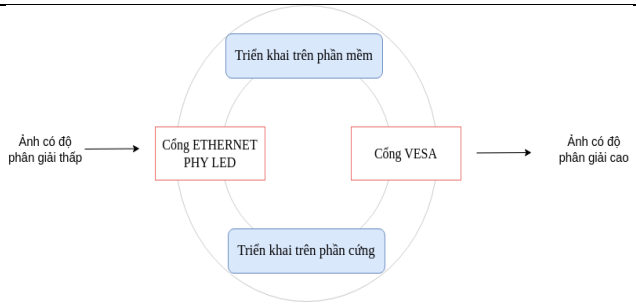
## 3.1. Yêu cầu hệ thống

Đề tài yêu cầu thực hiện mô hình mạng siêu phân giải RCAN theo cấu trúc thư viện Pytorch được triển khai trên Kaggle và Zynq Ultrascale+ MPSoC - ZCU102.

- Thiết kế các IP Cores giao tiếp giữa Zynq Ultrascale+ MPSOC với Deep Learning Processor Unit (DPU).
- Xây dựng hệ thống khởi động tích hợp thiết kế IP Cores chạy hệ điều hành Linux cho Xilinx Ultrascale+ MPSoC - ZCU102.
- Tiến hành tiền xử lý thuật toán siêu phân giải ảnh RCAN và triển khai trên Xilinx Ultrascale+ MPSoC - ZCU102.

## 3.2. Sơ đồ khối và đặc điểm kỹ thuật hệ thống

Sơ đồ khối của hệ thống siêu phân giải ảnh được mô tả qua hình 3.1. Trong đó, ảnh có độ phân giải thấp được đọc qua cổng Ethernet từ máy tính và đưa vào khối xử lý siêu phân giải hình ảnh. Các module giao tiếp với nhau bằng chuẩn AXI4-Stream và kết quả của nó sẽ được lưu vào bộ nhớ DDR trên phần cứng, có thể hiển thị bằng cổng VESA




***Hình 3.1.*** *Sơ đồ khối hệ thống siêu phân giải ảnh*

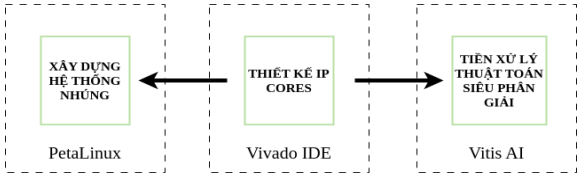
## 3.3. Thiết kế hệ thống

Để xây dựng hệ thống cần phải triển khai trên hai môi trường nền tảng:

***Nền tảng phần mềm:*** Thực hiện thiết kế IP Cores, xây dựng hệ thống nhúng, và tiền sử lý mạng siêu phân giải RCAN.

***Nền tảng phần cứng:*** Khởi chạy môi trường làm việc Linux và triển khai thuật toán siêu phân giải ảnh đã biên dịch từ nền tảng phần mềm.

### 3.3.1. Nền tảng phần mềm




***Hình 3.2.*** *Sơ đồ khối nền tảng phần mềm*

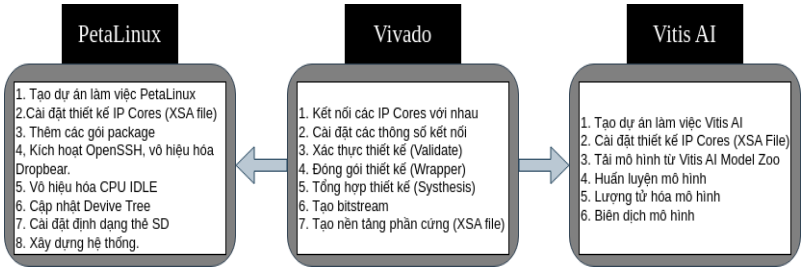
**Thiết kế IP Cores:** Thực hiện trên Vivado IDE được Xilinx phát triển. Kết nối các IP Cores đã được thiết kế như Xilinx như Zynq Ultrascale+ MPSoC, Deep Learning Processor Unit, Processor System Reset,... với nhau và tiến hành cài đặt thông số cần thiết, xác minh, đóng gói, triển khai, tạo bitstreams,... để tạo nền tảng phần cứng cho quá trình xây dựng hệ thống nhúng.

**Xây dựng hệ thống nhúng:** Thực hiện trên PetaLinux. Công cụ này có chức năng tự động hóa quá trình xây dựng (build) hệ thống chạy hệ điều hành Linux nhúng dựa trên các cấu hình và mã nguồn được tạo từ khối thiết kế IP Cores. Quá trình này bao gồm việc tạo ra hệ điều hành Linux kernel, các công cụ hệ thống như busybox, thư viện và ứng dụng cần thiết khác, cũng như việc tạo ra hệ thống tập tin rootfs (file system) để chạy trên Zynq Ultrascale+ MPSoC - ZCU102.

**Tiền xử lý thuật toán siêu phân giải:** Khối này thực hiện trên cloud (Kaggle) và phần mềm VitisAI IDE. VitisAI được tích hợp các thuật toán mạng học sâu của các thư viện như Tensorflow, Pytorch, Caffe,... trong thư mục Vitis AI Model Zoo. Ngoài ra Vitis AI còn có các công cụ tiền xử lý các thuật toán như tối ưu, lượng tử hóa, biên dịch,... Công cụ Kaggle sẽ huấn luyện mô hình mạng và VitisAI sẽ xử lý mô hình mạng đã được huấn luyện để có thể triển khai trên ZCU102.

#### 3.3.1.1. Quy trình thiết kế nền tảng phần mềm

Quy trình thiết kế gồm có ba khối chính được mô tả ở hình 3.3 dưới đây:

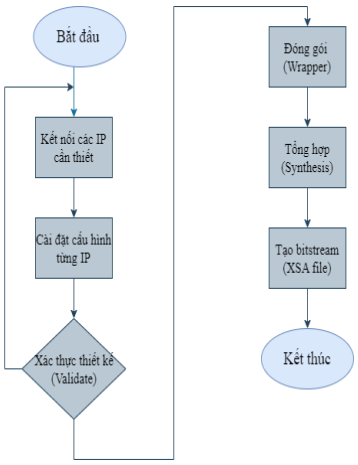



***Hình 3.3.*** *Quy trình thiết kế nền tảng phần mềm*

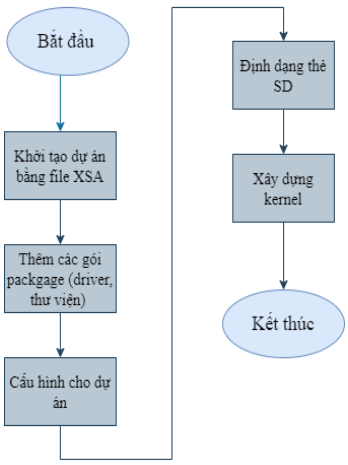
Nội dung thiết kế nền tảng phần mềm được mô tả từng bước như sau:

- **Phần mềm Vivado IDE:**

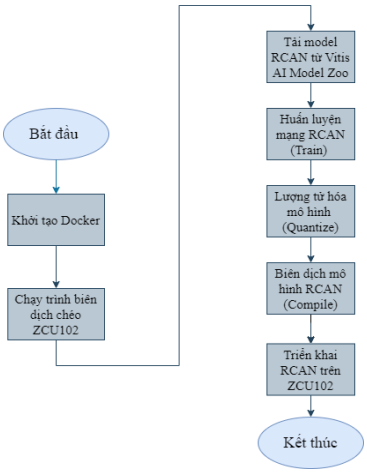
Đây là bước đầu tiên. Mục tiêu là tạo ra nền tảng phần cứng được định dạng **.sxa.** Lưu đồ thiết kế lõi IP trên Vivado được mô tả trong hình dưới đây:




***Hình 3.4.*** *Lưu đồ thiết kế IP tạo file XSA trên Vivado IDE*




***Hình 3.5.*** *Lưu đồ thiết kế dự án Linux trên công cụ PetaLinux*




***Hình 3.6.*** *Quy trình thiết kế hệ thống siêu phân giải ảnh trên công cụ Vitis AI*

- Đầu tiên, kết nối các I/O của IP Cores như Zynq Ultrascale+ MPSOC, Deep Learning Processor Unit,Clock Wi với nhau. Cài đặt các thông số kết nối như số lõi DPU, chuẩn kết nối, tần số,...
- Xác thực thiết kế (Validate Design) sẽ kiểm tra tính ràng buộc thiết kế như ràng buộc thời gian (timing constraints), ràng buộc bố trí (placement constraints), ràng buộc kết nối (routing constraints), kiểm tra các khối chức năng được kết nối đúng cách, không có lỗi kết nối (routing errors) và không có lỗi thiết kế logic nào xảy ra. Đảm bảo rằng chúng được thiết lập chính xác và đáp ứng được yêu cầu của FPGA.
- Đóng gói thiết kế (Wrapper) sẽ đóng gói các IP thành các module bậc cao để có thể dễ dàng quản lý kết nối giữa các IP Cores và các phần khác của dự án. Tổng hợp thiết kế (Synthesis) giúp tối ưu hóa cấu trúc và chức năng của IP để đảm bảo tính tương thích, hiệu suất tốt nhất.
- Bitstream là một tập hợp các dữ liệu được lập trình để cấu hình FPGA và các tài nguyên liên quan, chẳng hạn như bộ nhớ, cấu trúc logic và các kết nối. Tệp XSA mô tả toàn bộ hệ thống phần cứng, bao gồm cả cấu hình FPGA, IP cores, bộ nhớ, các kết nối và tài nguyên khác. File XSA có thể dễ dàng tích hợp vào các dự án khác và sử dụng lại trong các môi trường làm việc khác nhau như PetaLinux, Vitis AI.

❖ **Công cụ PetaLinux**

- Đầu tiên, tạo dự án PetaLinux với file XSA được tạo ra từ bước 7 ở phần mềm Vivado.Thêm các gói package hỗ trợ người dùng như *xrt, dfn, vitisai,...*
- Kích hoạt OpenSSH và vô hiệu hóa Dropbear nhằm mục đích cải thiện tốc độ truyền dữ liệu nhanh hơn trong nền tảng nhúng.

- Vô hiệu hóa CPU IDLE trong cấu hình của Kernel giúp tránh trường hợp hệ thống bị treo do các giao tiếp AXI chưa hoàn tất. Cập nhật Device Tree giúp mô tả các thành phần phần cứng hệ thống dựa trên cấu hình phần cứng từ file .**xsa.**
- Cài đặt định dạng thẻ SD. Vitis AI cần 2 phân vùng trên thẻ SD đó là FAT32 chứa các tập tin khởi động và EXT4 chứa các cấu hình của Linux chạy trên hệ thống nhúng (Phụ lục). Xây dựng PetaLinux sẽ tiến hành tạo các tập tin cần thiết để chạy trên hệ thống nhúng.

❖ **Công cụ Vitis AI**

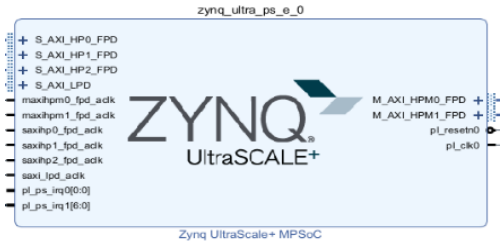
- Cài đặt môi trường làm việc Vitis AI trên docker, sau đó tiến hành chạy trình biên dịch chéo trên Zynq Ultrascale+ MPSoC-ZCU102.
- Sử dụng mô hình RCAN trong thư viện Vitis AI Model Zoo được thiết kế theo thư viện Pytorch. Mục đích nhằm giúp tiết kiệm thời gian và nâng cao độ chính xác của model.
- Tiến hành huấn luyện mô hình (Training). Mô hình mạng RCAN được huấn luyện trên công cụ Kaggle. Sử dụng tập dữ liệu có sẵn DIV2K gồm hơn 800 ảnh có độ phân giải cao khác nhau dùng để huấn luyện và 800 ảnh dùng để kiểm tra.
- Tiến hành lượng tử hóa mô hình (Quantisation) bằng công cụ Vitis AI Quantisation. Đây là một bước bắt buộc, nếu thiếu bước này thì mô hình không thể triển khai được. Bước này chuyển đổi mô hình RCAN có định dạng mô hình từ FP32 thành INT8.
- Tiến hành biên dịch mô hình (Compile) để tạo tập tin có đuôi .elf. Quá trình chuyển đổi mô hình mạng thành mã máy (ngôn ngữ bậc thấp) có thể chạy được trên phần cứng.
- Chuyển model RCAN đã được biên dịch vào ZCU102 qua cổng Ethernet. Tiến hành kiểm tra model bằng cách siêu phân giải ảnh của tập tin benhmark.

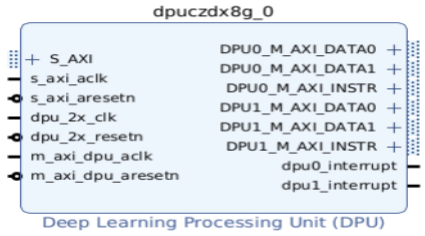
#### 3.3.1.2. Thiết kế IP Cores trên Vivado IDE

Hệ thống mô tả phần cứng gồm có các IP chính sau: Zynq Ultrascale+ MPSOC, Deep Learning Processor Unit (DPU), Processor System Reset, Clocking Wizard, AXI Interconnect.

**Khối IP Zynq Ultrascale+ MPSOC và hối IP Deep Learning Processor Unit** [10]

Khối IP Zynq Ultrascale+ MPSOC chứa các thông tin cấu trúc về thành phần PS và PL của Zynq Ultrascale+ MPSoC - ZCU102. Khối IP Deep Learning Processor unit (DPU) có thể được định cấu hình cho một số lõi DPU , kiến trúc tích chập, xếp tầng DSP, sử dụng DSP, …




***Hình 3.7.*** *IP Zynq Ultrascale+ MPSOC* ***Hình 3.8.*** *IP Deep Learning Processor Unit*

**Bảng 3.1**. *Thông tin các chân I/O của IP Zynq Ultrascale+ MPSOC* [11]

| Tên giao diện | Viết tắt | I/O | Vị trí | Mô tả |
|---|---|---|---|---|
| S_AXI_HP{0:3}_FPD | HP{0:3} | I | PL | Chân data có băng thông lớn từ PS đến PL. |
| S_AXI_LPD | PL_LPD | I | PL | Chân data có băng thông nhỏ từ PS đến PL. |
| M_AXI_HPM(0,1)_FPD | HPM(0,1) | O | PS | Cho phép PL thực hiện đọc/ghi đến bộ nhớ và tài nguyên khác trong PS với hiệu suất cao. |
| saxihp(0:3)_fpd_aclk | | I | PS | Tín hiệu clock có chức năng đồng bộ hóa truy cập (đọc) dữ liệu. |
| pl_ps_irq(0,1) | | I | PL | Tín hiệu ngắt từ PL gửi tới PS. |
| pl_resetn0 | | I | PL | Tín hiệu reset hoạt động của PL. |

***Bảng 3.2.*** *Thông tin các chân của IP DPU*

| Tên tín hiệu | Loại tín hiệu | Kích thước | I/O | Mô tả |
|---|---|---|---|---|
| S_AXI | Data (Slave) | 32 | I/O | Giao diện AXI được ánh xạ với bộ nhớ 32 bit cho các thanh ghi. |
| s_axi_aclk | Clock | 1 | I | Tín hiệu clock AXI cho S_AXI. |
| s_axi_aresetn | Reset | 1 | I | Tín hiệu reset tích cực mức thấp cho S_AXI. |

| | | | | |
|---|---|---|---|---|
| dpu_2x_clk | Clock | 1 | I | Tín hiệu clock cấp cho khối DSP của DPUCZDX8G. Tần số gấp đôi m_axi_dpu_aclk. |
| dpu_2x_resetn | Reset | 1 | I | Tín hiệu reset tích cực mức thấp cho khối DSP. |
| m_axi_dpu_aclk | Clock | 1 | I | Tín hiệu clock cấp cho các khối logic trong DPUCZDX8G. |
| m_axi_dpu_aresetn | Reset | 1 | I | Tín hiệu reset tích cực mức thấp cho khối logic. |
| DPU(0,1)_M_AXI_DATA0 | Data (Master) | 128 | I/O | 128 bit cho phiên bản Zynq Ultrascale+ MPSoC. |
| DPU(0,1)_M_AXI_DATA1 | Data (Master) | 128 | I/O | 128 bit cho phiên bản Zynq Ultrascale+ MPSoC. |
| DPU(0,1)_M_AXI_INSTR | Data (Master) | 32 | I/O | Giao diện AXI được ánh xạ với bộ nhớ 32 bit dùng để truyền tải các dòng lệnh cho DPUCZDX8G. |
| dpu(0,1)_interrupt | Ngắt | 1 | O | Ngõ ra tín hiệu ngắt tích cực mức cao cho DPUXZDX8G. |
| SFM_M_AXI (tùy chọn) | Data (Master) | 128 | I/O | Giao diện AXI được ánh xạ cho dữ liệu softmax. |

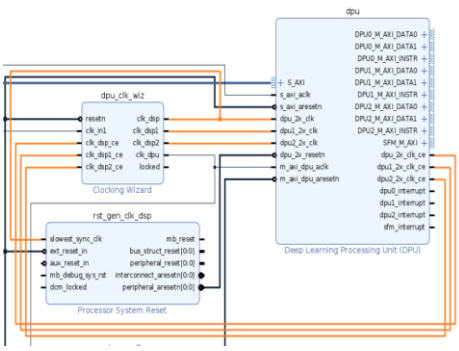
Có thể chọn tối đa 4 lõi trong DPU. Càng nhiều lõi giúp tăng hiệu suất. Ở trường hợp siêu phân giải ảnh, chỉ cần dùng 1 lõi. IP DPU có thể được cấu hình với nhiều kiến trúc tích chập khác nhau. Các kiến trúc cho IP DPU bao gồm B512, B800, B1024, B1152, B1600, B2304, B3136 và B4096. Trong báo cáo này sử dụng B4096. Mức sử dụng RAM: xác định tổng dung lượng bộ nhớ trên chip được sử dụng trong các kiến trúc DPU. Quá trình siêu phân giải ảnh chỉ chạy một mô hình CNN nên chọn Low. Tùy chọn ALU Parallel và ALU Parallel lần lượt là 4.

***Bảng 3.3.*** *Cấu hình DPU IP*

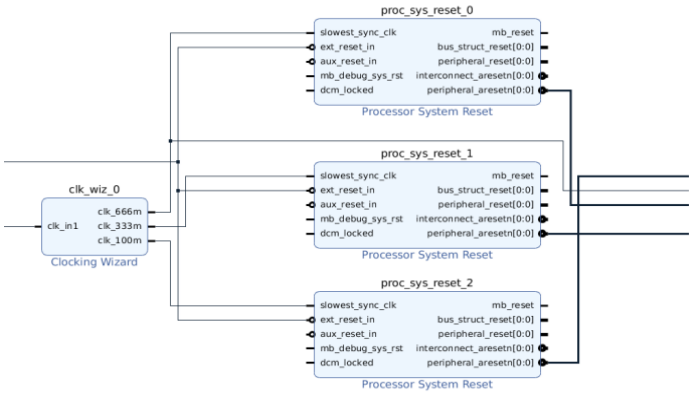
| | |
|---|---|
| Số lượng lõi | 1 |
| Kiến trúc DPU | B4096 |

| | |
|---|---|
| Mức sử dụng bộ nhớ RAM | Low |
| Khả năng tăng cường các kênh | Enabled |
| Loại hàm kích hoạt ReLU | ReLU + ReLU6 |
| Số lượng ALU song song | 4 |
| Số lõi Softmax | 1 |
| Phiên bản | 1.4.1 |
| Chuẩn truyền AXI | AXI4 |

- **Khối tạo tín hiệu clock IP Clock Wizard và khối tạo tín hiệu reset IP Processor System Reset**




***Hình 3.9.*** *Kết nối giữa IP Clocking Wizard và IP DPU*




***Hình 3.10.*** *Cấu hình IP Processor System Reset*

Trong DPU, có 3 khối cần tín hiệu clock đó là khối cấu hình thanh ghi, khối điều khiển dữ liệu và khối tín toán. Ba tín hiệu clock này có thể được cấu hình độc lập để phù hợp với các yêu cầu về hệ thống và hiệu suất. Để đáp ứng được những yêu cầu trên, IP Clock Wizard được sử dụng trong thiết kế này. Vì có ba tín hiệu clock ở ngõ vào nên mỗi tín hiệu clock sẽ có một tín hiệu reset tương ứng. [10] Mỗi tín hiệu reset phải được đồng bộ hóa với tín hiệu clock tương ứng. Khối IP Processor System Reset được sử dụng để tạo ra tín hiệu reset được đồng bộ hóa.

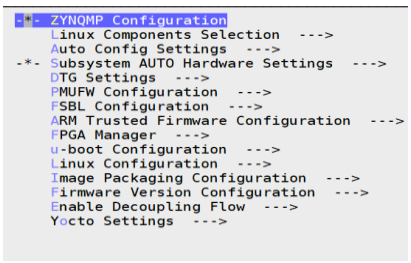
Tần số *s_axi_aclk* được đặt thành 100 MHz và *m_axi_dpu_aclk* được đặt thành 325 MHz. Ttần số của *dpu_2x_clk* được đặt thành 650 MHz . Cấu hình cho IP Clock Wizard được hiển thị trong bảng 3.4 dưới đây:

***Bảng 3.4.*** *Cấu hình IP Clock Wizard* [10]

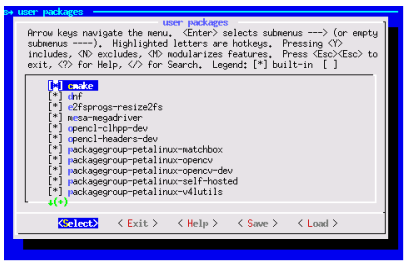
| Tên tín hiệu | Thiết lập | Mô tả |
|---|---|---|
| clk_dsp | 650MHz | Tần số của tín hiệu dpu_2x_clk. |
| clk_dpu | 325MHz | Tần số của tín hiệu m_axi_dpu_aclk. |
| Duty Cycle | 50% | Độ rộng xung của mức cao và mức thấp bằng nhau. |
| Driver | Buffer with CE | Driver là bộ đệm với cấu hình CE (Clock Enable). |

### 3.3.1.3. Xây dựng hệ thống nhúng

Mục tiêu là khởi tạo được các tập tin khởi động cho ZCU102 có IP mô tả phần cứng là file **.xsa** được tạo ra ở phần mềm Vivado. Công cụ PetaLinux do Xilinx phát triển được sử dụng trong phần này. Đầu tiên là cập nhật thông tin phần cứng ZCU102 ở phần DTG Setting trong mục cấu hình của dự án (Hình 3.11). Sau đó, kích hoạt các gói package cho dự án như *packagegroup-petalinux-vitisai, cmake, xrt_dev,...* (Hình 3.12).



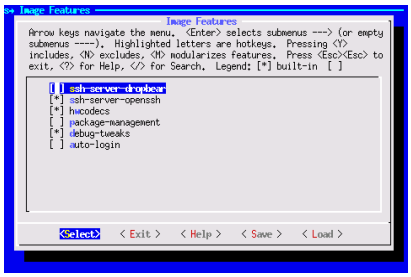

***Hình 3.11.*** *Các thông số cấu hình cho dự án PetaLinux*



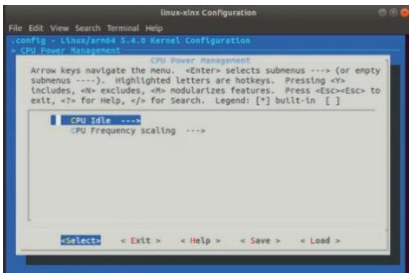

***Hình 3.12.*** *Thiết lập các gói package người dùng*

Kích hoạt OpenSSH và vô hiệu hóa Dropbear. Dropbear là công cụ SSH mặc định trong Nền tảng nhúng Vitis IDE. Nếu OpenSSH được sử dụng để thay thế Dropbear, hệ thống có thể đạt tốc độ truyền dữ liệu nhanh hơn gấp 4 lần qua ssh (ug1144). Vì các ứng dụng Vitis-AI có thể sử dụng tính

năng hiển thị từ xa để hiển thị kết quả học máy nên việc sử dụng OpenSSH có thể cải thiện trải nghiệm hiển thị (Hình 3.13).




***Hình 3.13.*** *Kích hoạt SSH và vô hiệu hóa Dropbear*




***Hình 3.14.*** *Vô hiệu hóa CPU IDLE*

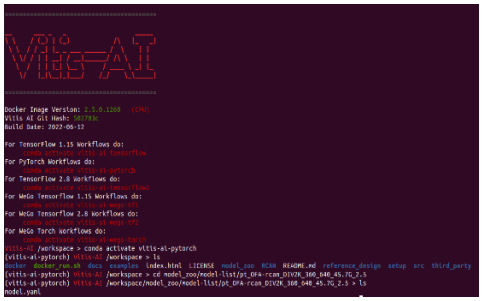
Vô hiệu hóa CPU IDLE trong Kernel. CPU IDLE sẽ khiến bộ xử lý chuyển sang trạng thái IDLE (trạng thái chờ) khi bộ xử lý không được sử dụng. Khi JTAG được kết nối, môi trường thiết kế phần cứng (hardware server) trên Vitis IDE sẽ thường xuyên trao đổi với bộ xử lý. Nếu nó giao tiếp với bộ xử lý ở trạng thái IDLE, hệ thống sẽ bị treo do các giao dịch AXI chưa hoàn tất. Do đó, nên tắt tính năng CPU IDLE trong giai đoạn cài đặt dự án (Hình 3.14).

Từ bất kỳ thư mục nào trọng dự án PetaLinux **"zcu102_platform"**, sử dụng lệnh: *petalinux-build* để xây dựng một hệ thống nhúng chạy hệ điều hành Linux hoạt động trên ZCU102

#### 3.3.1.4. Tiền xử lý hệ thống siêu phân giải ảnh

- **Tải mô hình siêu phân giải ảnh**

Model Zoo cung cấp mô hình siêu phân giải ảnh được định dạng theo cấu trúc thư viện Pytorch có tên **pt_OFA_rcan_DIK2K_360_640_45.7G_2.5.** Sau khi tải thư viện Vitis AI từ Github và truy cập bằng docker, môi trường làm việc của Vitis-AI được mô tả trong hình 3.15.




***Hình 3.15.*** *Môi trường làm việc Vitis AI trên Docker*

Trong bài báo cáo này, sử dụng phiên bản Vitis AI 2.5. Các phần mềm PetaLinux, Vivado và Vitis sử dụng phiên bản được phát hành vào năm 2022.2.

- **Huấn luyện mô hình**

Dữ liệu dùng cho huấn luyện mô hình là tập dữ liệu DIV2K. Đây là một bộ dữ liệu được sử dụng rộng rãi trong lĩnh vực xử lý ảnh và thị giác máy tính để đánh giá và so sánh hiệu suất của các thuật toán siêu phân giải hình ảnh. Tên "DIV2K" được viết tắt từ "Diverse 2K resolution dataset", tương tự như "Bộ dữ liệu đa dạng độ phân giải 2K". Bộ dữ liệu này bao gồm 2.000 hình

ảnh, được chia thành tập huấn luyện và tập kiểm tra. Mỗi hình ảnh trong DIV2K được chia thành các tập hợp riêng biệt để đảm bảo tính đa dạng và độ phong phú của dữ liệu. Việc huấn luyện mô hình cần nhiều tài nguyên của máy tính, nên trong báo cáo này, sẽ thực hiện huấn luyện mô hình trên Cloud, cụ thể là trên môi trường Kaggle. Thông số cho quá trình huấn luyện được mô tả trong bảng 3.10 dưới đây:

***Bảng 3.4.*** *Thông số quá trình huấn luyện mạng RCAN*

| Thông số | Giá trị | Mô tả |
|---|---|---|
| LR<br>(Learning Rate) | 3e-4<br>(0.003) | Tốc độ học của mô hình. Kích thước của bước điều chỉnh trọng số trong mỗi lần epoch được cập nhật. |
| BETAS | 0.9 , 0.999 | Tham số beta cho thuật toán của quá trình tối ưu hóa Adam. Đại diện cho hệ số giảm dần của trung bình động bậc nhất (Momentum) và bậc hai của gradient. |
| STEP_SIZE | 10 | Cứ mỗi sau 10 epoch thì tốc độ học sẽ được điều chỉnh theo hệ số GAMMA. |
| GAMMA | 0.67 | Đây là hệ số làm giảm tốc độ học, chỉ định tỷ lệ mà tốc độ học sẽ bị giảm sau mỗi STEP_SIZE. Cụ thể, tốc độ học sẽ giảm 67% sau mỗi 10 epoch. |
| NUM_EPOCHS | 100 | Tổng số epoch mà mô hình sẽ huấn luyện. |
| DEVICE | cuda | Chọn thiết bị mà mô hình sẽ được huấn luyện. Được đặt là cuda (GPU). Nếu không có GPU thì sẽ huấn luyện trên CPU. |

❖ **Lượng tử hóa mô hình**

Sau khi huấn luyện mô hình, các trọng số không quan trọng được loại bỏ bằng phương pháp cắt tỉa (Prune) nhưng mô hình lúc này vẫn rất phức tạp. Để mô hình hoạt động trên SoC (System on chip) thì mô hình cần được đơn giản nhưng vẫn giữ nguyên được hiệu xuất hoạt động.Lượng tử hóa là quá trình chuyển đổi trọng số từ giá trị dấu phẩy động 32 bit FP32 (Floating Point 32 bit) sang biểu diễn Số nguyên 8 bit (INT8). Đây là bước bắt buộc vì DPU chỉ chấp nhận các giá trị biểu diễn số nguyên 8 bit. Để xử lý, Vitis AI cung cấp công cụ vai_q_pytorch. Tiến hành chạy file ***run_quant.sh*** với các thông số được cài đặt như sau:

scale : được đặt là 2,  n_resgroups: được đặt là 1, epochs: Vòng lặp được đặt là 250.

Công cụ ***vai_q_pytorch*** sẽ tiến hành lượng tử mô hình có các thông số trong tập tin *run_quant.py*. Sau khi quá trình lượng tử hoàn tất sẽ tạo ra một file có tên là ***Model_int.xmodel.*** Tập tin này chứa mô hình đã được định dạng thành INT8, có thể triển khai trên DPU.

❖ **Biên dịch mô hình mạng**

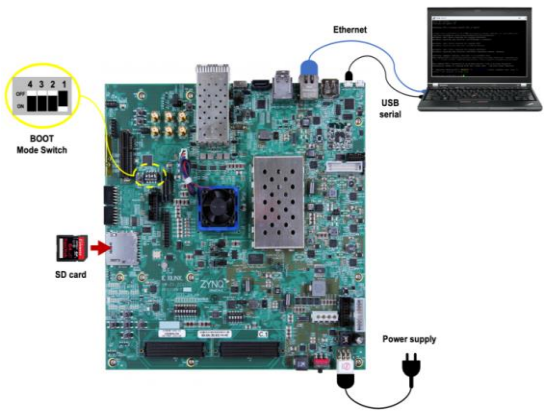
Sau khi tiến hành lượng tử hóa mô hình, bước cuối cùng là tiến hành biên dịch mô hình mạng.Trình biên dịch mô hình mạng sẽ biên dịch mô hình đã được lượng tử thành đoạn mã bitstream và các tập lệnh có thể chạy trên DPU. Tiến hành biên dịch mô hình mạng RCAN bằng dòng lệnh dưới đây:

```
vai_c_xir -x /workspace/RCAN/pt_OFA-rcan-
DIV2K_360_640_45.7G_2.5/quantized/Model_int.xmodel -a
/vitis_ai/compiler/arch/DPUCZDX8G/ZCU102/arch.json --output_dir
/workspace/RCAN/pt_OFA-rcan-DIV2K_360_640_45.7G_2.5/compile -n rcan
```

Đoạn mã trên sẽ sử dụng công cụ vai_c_xir-x để biên dịch mô hình đã được lượng tử có tên Model_int.xmodel và file .json mô tả kiến trúc phần cứng là ZCU102. Sau khi biên dịch hoàn tất, VAI_C sẽ tạo tập tin có tên rcan.xmodel để triển khai trên board mạch.

### 3.3.2. Nền tảng phần cứng

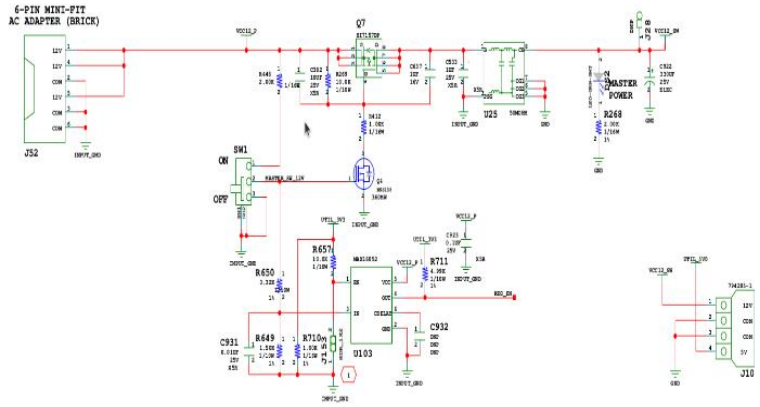
Thiết kế phần cứng áp dụng những kết quả của quá trình phát triển nền tảng phần mềm, được mô tả ở hình 3.16 bên dưới:




Xác định kết nối truyền dữ liệu giữa máy tính và phần cứng ZCU102 qua giao diện Ethernet.
Ghi các tập tin được xây dựng bởi công cụ PetaLinux vào thẻ SD để tiến hành khởi động ZCU102 ở chế độ SD Boot.
Tiến hành thiết lập các kết nối phần cứng như nguồn, USB, UART.
Triển khai mô hình siêu phân giải ảnh trên ZCU102 sử dụng mạng nơ-ron tích chập RCAN.

***Hình 3.16.*** *Quy trình thiết kế phần cứng*

#### 3.3.2.1. Khối nguồn




Zynq Ultrascale+ MPSOC - ZCU102 sử dụng nguồn điện 12V DC.

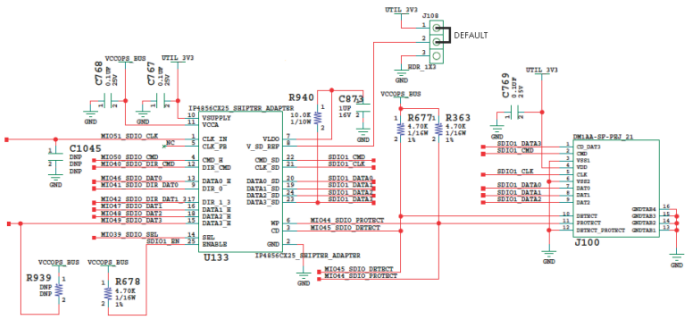
Công tắc nguồn của bo mạch ZCU102 là SW1. Trượt công tắc từ vị trí Tắt (OFF) sang Bật (ON) sẽ cấp nguồn 12V DC từ J52 là đầu nối mini 6 chân.

***Hình 3.17.*** *Kết nối nguồn cho ZCU102*

Đèn LED DS2 màu xanh lục sáng khi nguồn của ZCU102 được bật. Ngoài ra hệ thống còn có IC điều khiển MOSFET SI7157DP có chức năng chống đảo pha, bảo vệ điện áp, quá dòng và IC MAX16052 giúp quản lý nguồn điện vào ra.

### 3.3.2.2. Khối giao diện SD

Thẻ SD là nơi lưu trữ và truy cập dữ liệu cho bo mạch. Thẻ cần định dạng đúng và được kết nối chính xác. Sơ đồ kết nối giữa thẻ SD trên ZCU102 được mô tả ở hình 3.18.



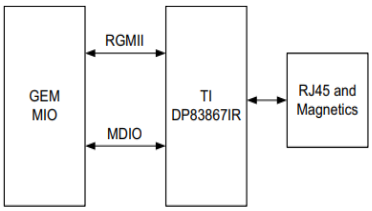

***Hình 3.18.*** *Sơ đồ kết nối thẻ SD*

Các tín hiệu SDIO được kết nối với XCZU9EG MPSoC PS Bank501 [1] có nguồn điện được cung cấp cho I/O (VCCMIO) được đặt thành 1,8V. Mỗi sáu chân MIOxx_SDIO_ đều có điện trở nối tiếp 30 Ω ở nguồn. Bộ chuyển đổi mức điện áp tương thích NXP IP4856CX25 SD 3.0 U133 nằm ở giữa bo mạch và đầu nối thẻ SD (J100). Thiết bị NXP IP4856CX25 U133 cung cấp khả năng SD3.0 với hiệu suất SDR104.

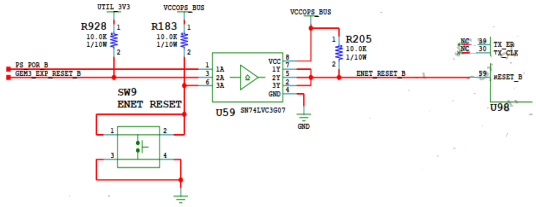
### 3.3.2.3. Khối ngõ vào ETHERNET PHY LED

Ngoài việc nhận dữ liệu cục bộ từ thẻ SD, ZCU102 có thể nhận dữ liệu trực tiếp thông qua giao diện GEM3 Ethernet. Gigabit Ethernet MAC (GEM) triển khai giao diện Ethernet có tốc độ l= 10/100/1000 Mb/s, được hiển thị trong Hình 3.21, kết nối với TI DP83867IRPAP Ethernet RGMII

PHY trước khi được định tuyến đến đầu nối Ethernet RJ45. RGMII Ethernet PHY được khởi động ở địa chỉ PHY 5'b01100 (0x0C) và Auto Negotiation được đặt thành Bật.




***Hình 3.19.*** *Sơ đồ khối kết nối Ethernet*

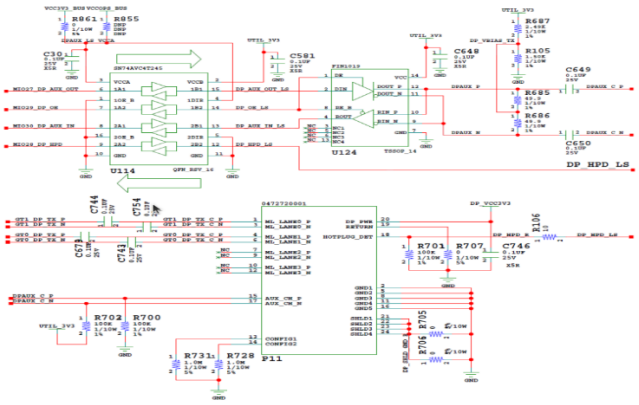



***Hình 3.20.*** *Sơ đồ kết nối mạch Reset*

Phần cứng ZCU102 sử dụng TIDP83867IRPAP tại U98 [1] để sử dụng Ethernet ở tốc độ 10Mb/s, 100Mb/s hoặc 1000Mb/s. Chế độ Reset Ethernet PHY: Mạch Reset DP83867IRPAP PHY (U98) được hiển thị trong hình 3.22. Có thể Reset DP83867IRPAP bằng nút nhấn SW9 (U59.6), thiết bị Reset POR phía PS MAX16025 (U22) MPSoC (U59.1) hoặc cổng mở rộng (U97) TCA6416A I/O được kết nối I2C0 P06 chân số 10 (U59. 3). [1]

### 3.3.2.4. Khối hiển thị VESA (DPAUX MIO 27 -30) [1]

Zynq UltraScale+ MPSoC cung cấp bộ điều khiển VESA DisplayPort 1.2 hỗ trợ tối đa hai luồng dữ liệu liên kết chính ở tốc độ 1.62Gb/s, 2.70Gb/s hoặc 5.40Gb/s. Chuẩn DisplayPort xác định kênh phụ trợ (auxiliary channel) sử dụng tín hiệu LVDS ở tốc độ dữ liệu 1 Mb/s, được dịch từ tín hiệu MIO một đầu sang kênh DisplayPort Aux khác.



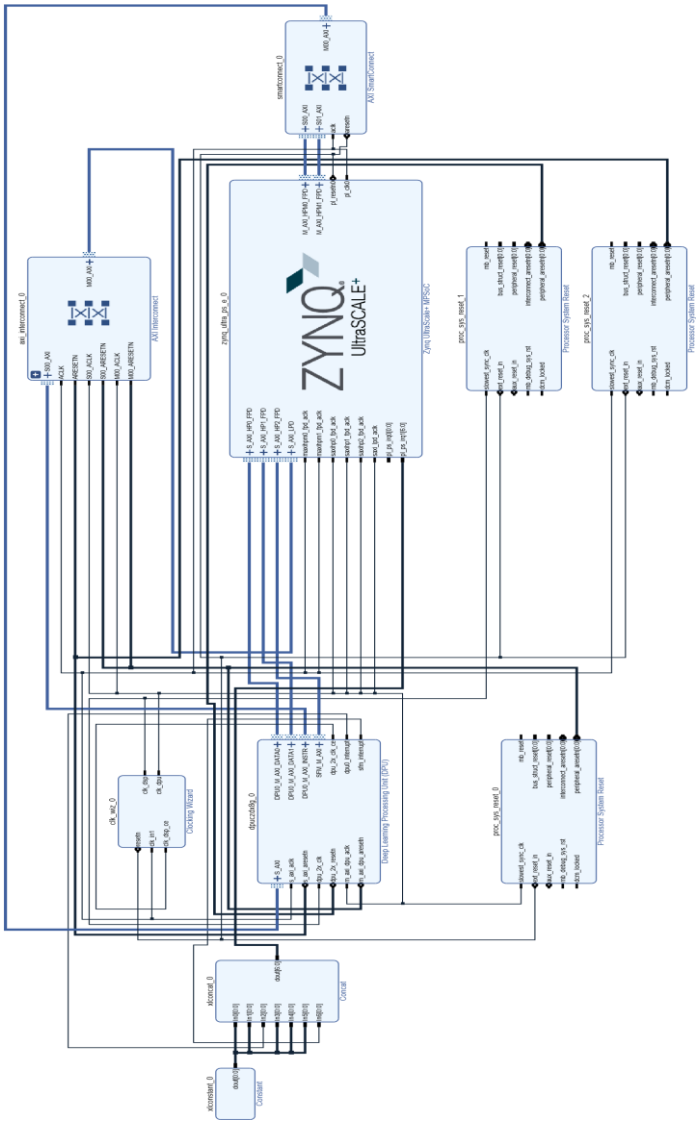

***Hình 3.21.*** *Sơ đồ kết nối của khối hiển thị VESA*

Các tín hiệu từ ZU102 được truyền lên màn hình quan sát. Tuy nhiên, mỗi màn hình quan sát đều có mức điện áp khác. Do đó, các tín hiệu sẽ đi qua U114. IC này chuyển đổi mức điện áp sao cho phù hợp. IC còn có khả năng truyền dẫn dòng điện cao lên đến 35mA nên cho phép kết nối trực tiếp với nhiều thiết bị mà không cần bộ khuếch đại. Cuối cùng, các tín hiệu này sẽ đi qua FIN1019 (U124), IC này có chức năng lọc nhiễu nguồn, lọc nhiễu tín hiệu và lọc nhiễu tín hiệu clock.

# CHƯƠNG 4: KẾT QUẢ

## 4.1. Kết nối IP đầy đủ

Hình 4.1 dưới đây mô tả toàn bộ kết nối IP của hệ thống gồm các IP chính như Zunq Ultrascale+ MPSoC - ZCU102, Deep Learning Processor Unit (DPU), Clock Wiward, Processor System Reset.



***Hình 4. 1.*** *Sơ đồ các kết nối IP đầy đủ của hệ thống*

Hệ thống các lõi IP được thiết kế với 4 lớp mạng (Network) được kết nối giữa Deep Learning Processor Unit (DPU) (Master) với Zynq Ultrascale+ MPSOC (Slave) bằng chuẩn giao diện AXI.

Sau khi quá trình tổng hợp (Synthesis) kết thúc, các lõi IP được chuyển thành các phần tử logic mà FPGA có thể hiểu được. Thông ttin về các phần tử này được mô tả chi tiết trong bảng 4.1 dưới đây:

***Bảng 4.1.*** *Tài nguyên của các khối IP khi tổng hợp*

| Constrains | LUT | FF | BRAM | URAM | DSP |
|---|---|---|---|---|---|
| DPU_zynq_ultra_ps_e_0_0 | 264 | 0 | 0 | 0 | 0 |
| DPU_clk_wiz_0_0 | 1 | 0 | 0 | 0 | 0 |
| DPU_proc_sys_reset_0_0 | 19 | 40 | 0 | 0 | 0 |
| DPU_proc_sys_reset_0_1 | 24 | 40 | 0 | 0 | 0 |
| DPU_dpuczdx8g_0_0 | 61893 | 107918 | 259 | 0 | 724 |

- Thành phần PS của ZCU102: Mức sử dụng tài nguyên là 264 LUT. Cho thấy khối IP này chủ yếu sử dụng các logic tổ hợp (combinational logic) mà không yêu cầu quá nhiều các yếu tố lưu trữ như FF hoặc bộ nhớ.
- Khối IP Clock Wizard: DPU_clk_wiz_0_0: Mức sử dụng tài nguyên là 1 LUT.
- Khối IP Processing System Reset: Mức sử dụng tài nguyên lần lượt là: 19 LUT, 40 FF và 24 LUT, 40 FF.
- Khối IP Deep Leanring Processor Unit: Mức sử dụng tài nguyên là 61893 LUT, 107918 FF, 259 BRAM, 724 DSP. IP này cần sử dụng nhiều tài nguyên để thực hiện các phép tính phức tạp liên quan đến mạng học sâu.

Bảng 4.2 dưới mô tả chi tiết các đường địa chỉ của các lớp mạng được sử dụng. Trong trường hợp này sử dụng 4 mạng lưới khác nhau (Network 0, 1, 2, 3), mỗi mạng lưới chứa các IP blocks khác nhau.

Các thông số như : Interface mô tả giao diện tương tác của IP block, Slave Segment chỉ ra chức năng của IP block (ví dụ: HP0_PCIE_LOW,HP1_LPS_OCM). Master Base Address và Range xác định vùng địa chỉ mà khối IP có thể truy cập Cuối cùng là Master High Address cho biết giới hạn trên của dải địa chỉ.

***Bảng 4.2.*** *Thông số các giao diện và đường địa chỉ thiết kế IP học sâu*

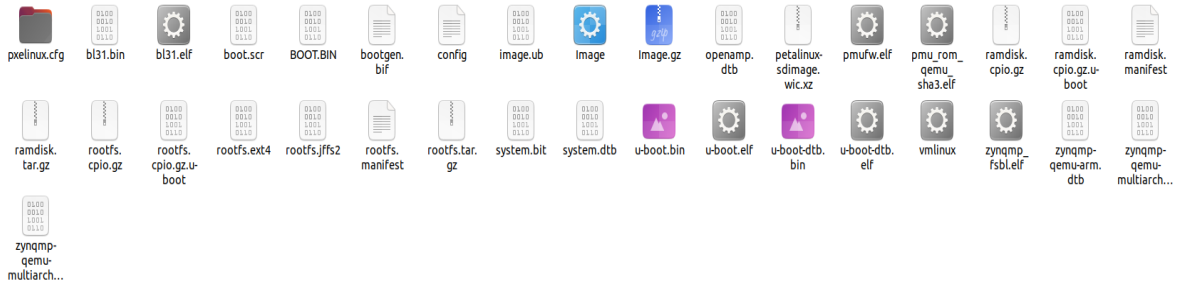
| Name | Interface | Salve Segment | Master Base Address | Phạm vi địa chỉ | Master High Address |
|---|---|---|---|---|---|
| Network 0/dpuczdx8g_0/DPU_M_AXI_DATA0 | | | | | |
| zynq_ultra_ps_e_0/SAXIGP2 | S_AXI_HP0_FPD | HP0_LPS_OCM | 0x00_FF00_0000 | 16M | 0x00_FFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP2 | S_AXI_HP0_FPD | HP0_QSPI | 0x00_C000_0000 | 512 | 0x00_DFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP2 | S_AXI_HP0_FPD | HP0_DDR_LOW | 0x00_0000_0000 | 2G | 0x00_7FFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP2 | S_AXI_HP0_FPD | HP0_PCIE_LOW | 0x00_E000_0000 | 256M | 0x00_EFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP2 | S_AXI_HP0_FPD | HP0_DDR_HIGH | 0x08_0000_0000 | 32G | 0x0F_FFFF_FFFF |
| Network 1/dpuczdx8g_0/DPU_M_AXI_DATA0 | | | | | |
| zynq_ultra_ps_e_0/SAXIGP3 | S_AXI_HP1_FPD | HP1_DDR_HIGH | 0x08_0000_0000 | 32G | 0x0F_FFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP3 | S_AXI_HP1_FPD | HP1_QSPI | 0x00_C000_0000 | 512 | 0x00_DFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP3 | S_AXI_HP1_FPD | HP1_PCIE_LOW | 0x00_E000_0000 | 256M | 0x00_EFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP3 | S_AXI_HP1_FPD | HP1_LPS_OCM | 0x00_FF00_0000 | 16M | 0x00_FFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP3 | S_AXI_HP1_FPD | HP0_DDR_LOW | 0x00_0000_0000 | 2G | 0x0F_7FFF_FFFF |

| Network 3/dpuczdx8g_0/DPU_M_AXI_INSTR | | | | | |
|---|---|---|---|---|---|
| zynq_ultra_ps_e_0/SAXIGP4 | S_AXI_HP2_FPD | HP2_DDR_LOW | 0x00_0000_0000 | 2G | 0x00_7FFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP4 | S_AXI_HP2_FPD | HP2_QSPI | 0x00_C000_0000 | 512M | 0x00_DFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP4 | S_AXI_HP2_FPD | HP2_PCIE_LOW | 0x00_E000_0000 | 256M | 0x00_EFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP4 | S_AXI_HP2_FPD | HP2_LPS_OCM | 0x00_FF00_0000 | 16M | 0x00_FFFF_FFFF |
| zynq_ultra_ps_e_0/SAXIGP4 | S_AXI_HP2_FPD | HP2_DDR_HIGH | 0x08_0000_0000 | 32G | 0X0F_FFFF_FFFF |
| Network 4/zynq_ultra_ps_e_0/Data | | | | | |
| dpuczdx8g_0/S_AXI | S_AXI | reg0 | 0x04_0000_0000 | 16M | 0x04_00FF_FFFF |
| dpuczdx8g_0/S_AXI | S_AXI | reg0 | 0x04_0000_0000 | 16M | 0x04_00FF_FFFF |

Sau khi thực hiện các bước đóng gói (Wrapper) và kiểm tra (Validate) các kết nối của hệ thống, tiến hành tổng hợp (Synthesis) , triển khai (Implementation) và tạo bitstream. Cuối cùng là tạo file nền tảng phần cứng (hardware platform) có đuôi **.xsa** chứa các thông tin vừa tạo để tiến hành cho bước tiếp theo là xây dựng hệ thống nhúng cho ZCU102.

### 4.2. Các tập tin xây dựng hệ thống nhúng

Kết quả của quá trình xây dựng nhân chạy hệ điều hành Linux (Build Kernel) chứa nền tảng phần cứng được tạo từ Vivado là các file dùng để khởi động (boot.src, BOOT.BIN,...) và các file cấu hình (pxelinux.cfg, rootfs.tar,...).




***Hình 4. 2.*** *Các tập tin được tạo sau khi build kernel*

Các tập tin này được ghi vào thẻ SD ( lớn hơn 16GB) và gắn vào J100 trên ZCU102 để khởi động. Quá trình khởi động được mô tả ở hình 4.3 và 4.4 bên dưới.

***Hình 4.3.*** *ZCU102 vừa bật nguồn*

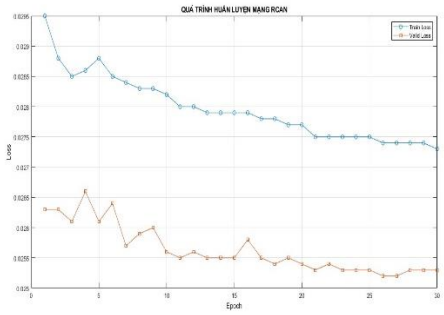
***Hình 4.4****. ZCU102 khởi động hoàn tất*

Khi bật nguồn, ZCU102 sẽ lấy các tập tin cần thiết từ thẻ SD đã được cắm vào J100 để tiến hành khởi động, lúc này đèn led báo trạng thái INT_B hiển thị màu đỏ. Sau khi khởi động thành công, led INT_B chuyển thành màu xanh, báo hiệu ZCU102 đã sẵng sàng.

### 4.3. Triển khai mô hình siêu phân giải ảnh

#### 4.3.1. Huấn luyện

Sử dụng tập dữ liệu "*benchmark*" để tiến hành chạy mô hình siêu phân giải ảnh. Tập dữ liệu này gồm có 4 tập dữ liệu con lần lượt là B100, Set14, Set5, Urban100. Tập dữ liệu Set14 gồm có 14 tấm ảnh có dung lượng từ 100KB đến 800KB và có kích thước dao động từ 300 đến 800 pixels. Tập dữu liệu Set5 gồm có 5 tấm ảnh có dung lượng từ 100KB đến 300KB và có kích thước dao động từ 280 đến 500 pixels. Cuối cùng là. Các tập dữ liệu này gồm có ảnh gốc có độ phân giải cao và ảnh có độ phân giải thấp được tạo ra bằng phép nội suy bicubic với các scale khác nhau.




***Hình 4.5.*** *Biểu đồ giá trị mất mát trên 30 epoch của quá trình huấn luyện mạng RCAN*

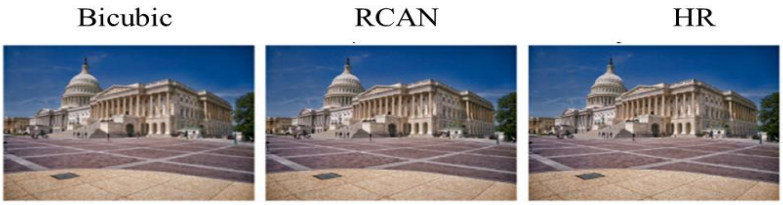
Hình 4.5 cho thấy giá trị mất mát (Loss) của tập dữ liệu Train và tập dữ liệu Valid đều giảm dần theo từng epoch. Sự chênh lệch giữa Train Loss và Valid Loss không lớn, chứng tỏ mô hình không bị quá khớp (Overfitting) với tập dữ liệu huấn luyện.

#### 4.3.2. Chạy mạng siêu phân giải ảnh trên phần cứng

Tiến hành chạy mô hình RCAN đã được huấn luyện trên trên ZCU102.


***Hình 4.6.*** *Kết quả siêu phân giải ảnh 253027.png của tập dữ liệu B100*

Ảnh 253027.png được lấy từ tập dữ liệu B100. Tập dữ liệu B100 gồm có 100 tấm ảnh có dung lượng từ 150 KB đến 300KB và có kích thước dao động từ 300 đến 500 pixels.




***Hình 4.7.*** *Kết quả siêu phân giải ảnh img070.png của tập dữ liệu Urban100*

Ảnh img070.png được lấy từ tập dữ liệu Urban100. Tập dữ liệu Urban100 gồm có 100 tấm ảnh có dung lượng từ 900KB đến 1.2MB và có kích thước dao động từ 600 đến 1200 pixels.


***Hình 4.8.*** *Kết quả siêu phân giải ảnh butterfly.png của tập dữ liệu Set5*

Ảnh butterfly được lấy từ tập dữ liệu Set5. Tập dữ liệu Set5 gồm có 5 tấm ảnh có dung lượng từ 100KB đến 300 KB và có kích thước dao động từ 256 đến 512 pixels.

Cuối cùng là ảnh coastguard được lấy từ tập dữ liệu Set14. Tập dữ liệu Set14 gồm có 14 tấm ảnh có dung lượng từ 100KB đến 800 KB và có kích thước dao động từ 256 đến 712 pixels.

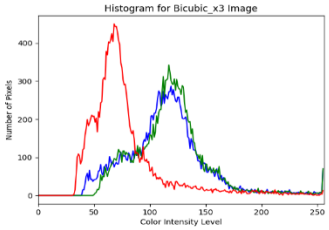

***Hình 4.9.*** *Kết quả siêu phân giải ảnh coastguard.png của tập dữ liệu Set14*

Hình 4.6, 4.7, 4.8, 4.9 là kết quả của quá trình chạy mô hình siêu phân giải RCAN trên phần cứng Zynq Ultrascale+ MPSoC – ZCU102. Ở mỗi bức hình gồm có 3 thông tin:.Bicubic là ảnh dùng phép nội suy Bicubic với scale = 3. HR là ảnh gốc có chất lượng cao. RCAN là ảnh được tạo khi chạy mạng siêu phân giải ảnh RCAN trên PL của ZCU102. Sự khác nhau về kích thước và dung lượng được mô tả trong bảng 4.3 dưới đây:

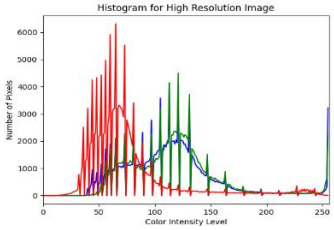
***Bảng 4.3.*** *Kích thước và dung lượng ảnh*

| | | Bicubic x3 | HR | RCAN |
|---|---|---|---|---|
| 253027 | Kích thước (pixesl) | 160×107 | 481×321 | 1280×720 |
| | Dung lượng | 34.7 KB | 290 KB | 259 KB |
| img070 | Kích thước (pixesl) | 341×256 | 1024×769 | 1280×720 |
| | Dung lượng | 160 KB | 1.47 MB | 1.02 MB |
| butterfly | Kích thước (pixesl) | 85 × 85 | 256×256 | 1280×720 |
| | Dung lượng | 16.9 KB | 124 KB | 101KB |
| coastguard | Kích thước (pixesl) | 117×96 | 352×288 | 1280×720 |
| | Dung lượng (KB) | 19.6KB | 150 KB | 89.8 KB |

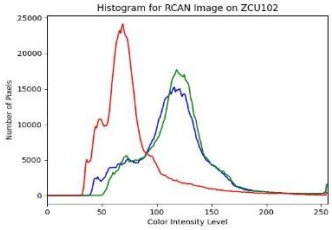
Dựa vào bảng 4.3 ở trên, kích thước của tất cả các ảnh đều được xử lý về mức 1280×720 pixel. Kích thước này là cố định, nhưng có thể điều chỉnh trong bước tiền xử lý hệ thống siêu phân giải ảnh được giới thiệu ở chương 3. Dung lượng của các bức ảnh được cải thiện gấp 5 đến 8 lần của ảnh Bicubic.




***Hình 4.10.*** *Biểu đồ histogram ảnh 253027 dùng bicubic_x3*

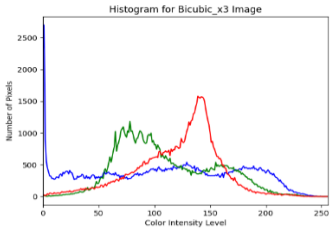



***Hình 4.11.*** *Biểu đồ histogram ảnh HR 253027*

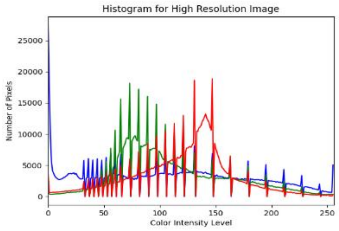



***Hình 4.12.*** *Biểu đồ histogram ảnh 253027 dùng RCAN*

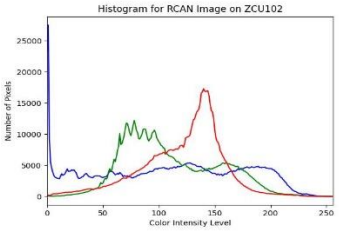
Cường độ màu kênh đỏ của 3 hình tập trung chủ yếu từ 30 – 100. Với số lượng pixels tại đỉnh lần lượt là 650, 4200, và 24600. Cường độ màu kênh xanh lá tập trung chủ yếu từ 60 – 150. Với số lượng pixels tại đỉnh lần lượt là 350, 4500, và 18000. Cường độ màu kênh xanh dương tập trung chủ yếu từ 60 – 150. Với số lượng pixels tại đỉnh lần lượt là 290, 2000, và 15000.




***Hình 4.13.*** *Biểu đồ histogram ảnh img070 dùng bicubic_x3*

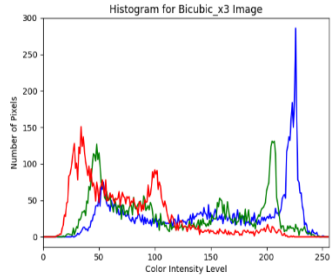



***Hình 4.14.*** *Biểu đồ histogram ảnh HR img070*

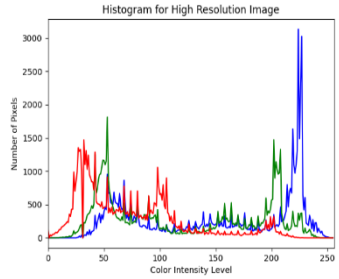



***Hình 4.15.*** *Biểu đồ histogram ảnh img070 dùng RCAN*

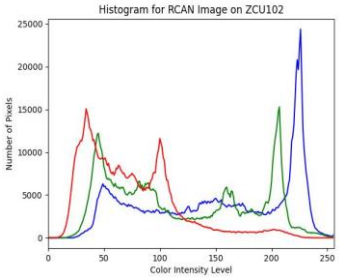
Cường độ màu kênh đỏ của 3 hình tập trung chủ yếu từ 60 – 180. Với số lượng pixels tại đỉnh lần lượt là 1500, 17000, và 16000. Cường độ màu kênh xanh lá tập trung chủ yếu từ 50 – 180. Với số lượng pixels tại đỉnh lần lượt là 1100, 18000, và 12500. Cường độ màu kênh xanh dương tập trung chủ yếu từ 0 – 225. Với số lượng pixels tại đỉnh lần lượt là 500, 7000, 4000.




***Hình 4.16.*** *Biểu đồ histogram ảnh butterfly dùng bicubic_x3*

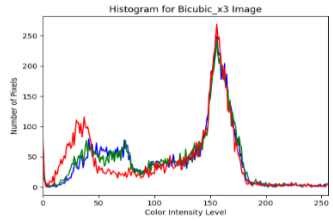



***Hình 4.17.*** *Biểu đồ histogram ảnh HR butterfly*

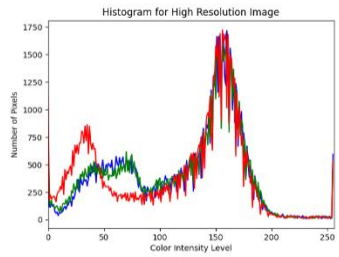



***Hình 4.18.*** *Biểu đồ histogram ảnh butterfly dùng RCAN*

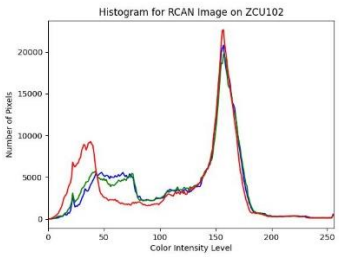
Cường độ màu kênh đỏ của 3 hình tập trung chủ yếu từ 20 – 120. Với số lượng pixels tại đỉnh lần lượt là 150, 1500, và 15000. Cường độ màu kênh xanh lá tập trung chủ yếu từ 30 – 110 và 180 - 220. Với số lượng pixels tại đỉnh lần lượt là 120, 1800, và 14000. Cường độ màu kênh xanh dương tập trung chủ yếu từ 40 – 60 và 200 - 250 . Với số pixels tại đỉnh lần lượt là 280, 3000, 24000.




***Hình 4.19.*** *Biểu đồ histogramảnh coastguard dùng bicubic_x3*




***Hình 4.20.*** *Biểu đồ histogram ảnh HR coastguard*



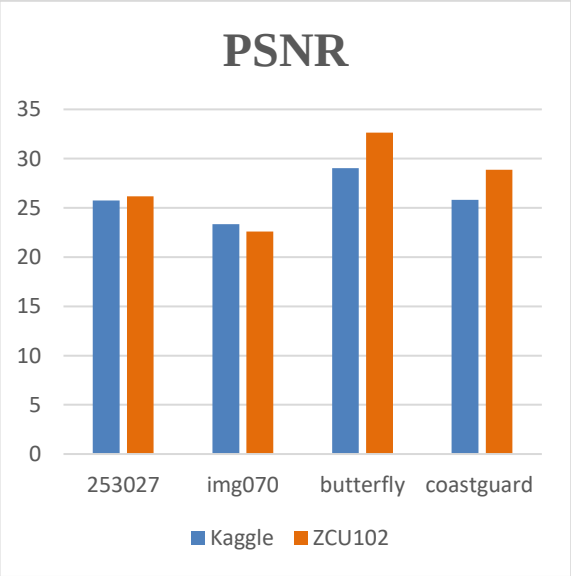

***Hình 4.21.*** *Biểu đồ histogram ảnh coastguard dùng RCAN*

Cường độ màu kênh đỏ của 3 hình tập trung chủ yếu từ 10 – 50 và 120 - 170. Với số lượng pixels tại đỉnh lần lượt là 270, 1800, 23000. Cường độ màu kênh xanh lá và xanh dương tập trung chủ yếu từ 30 – 80 và 120 - 180. Với số lượng pixels tại đỉnh lần lượt là 240, 1700, và 20000.

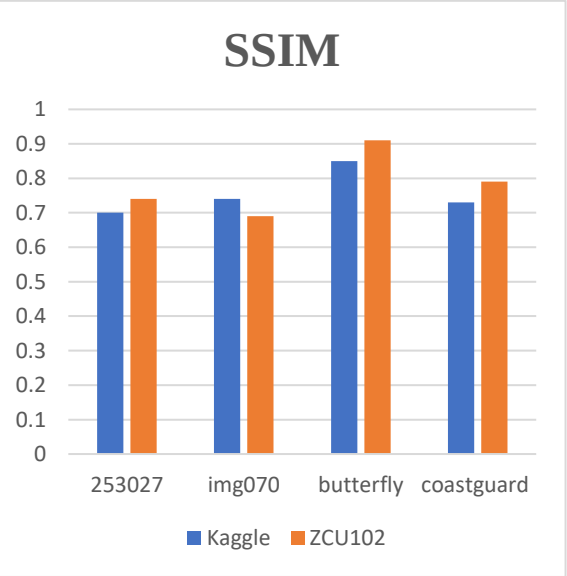
Dựa vào lược đồ histogram và so sánh kích thước và dung lượng ảnh, ảnh được tạo bằng mạng siêu phân giải RCAN trên ZCU102 có chất lượng tốt trên các tập dữ liệu Set5, Set14 và 100. Đặc điểm chung là dung lượng các ảnh trong những tập dữ liệu này đều ở mức thấp từ 100 KB đến 700KB và có kích thước nhỏ hơn 800x800 pixels. Đối với tập dữ liệu Urban100 với ảnh có dung lượng lớn (≥1MB) thì ảnh tạo ra không được hiệu suất cao.

### 4.3.3. Chạy mạng siêu phân giải ảnh trên Cloud (Kaggle)

Hình 4.22 và 4.23 dưới đây sẽ hiển thị các thông số so sánh ảnh một cách trực quan về chất lượng của ảnh chạy trên cloud (Kagggle) và ZCU102.




***Hình 4.22.*** *Biểu đồ cột giá trị PSNR khi chạy RCAN trên Kaggle và ZCU102*




***Hình 4.23.*** *Biểu đồ cột giá trị SSIM khi chạy RCAN trên Kaggle và ZCU102*

Giá trị PSNR của ảnh siêu phân giải RCAN thực hiện trên phần cứng Zynq Ultrascale+ MPSoC – ZCU102 có giá trị lớn hơn chạy trên Kaggle. Cụ thể, ảnh Butterfly (có chất lượng 124KB) được xử lý tốt nhất với giá trị PSNR lần lượt trên Kaggle và ZCU102 là 29.04 và 32.64. Giá trị SSIM lần lượt là 0.85 và 0.91. Cho thấy ảnh Butterfly dùng mạng siêu phân giải RCAN thực nghiệm trên phần cứng có chất lượng tốt hơn chạy trên Kaggle.

Tuy nhiên, với ảnh img070 của tập dữ liệu Urban100 (ảnh có chất lượng cao 1.47 MB) thì thông số PSNR và SSIM ảnh tạo ra từ ZCU102 lại thấp hơn trên Kaggle.

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

Đồ án đã đạt được một số thành công nhất định. Thành công tạo được nền tảng phần cứng từ các IP, xây dựng được hệ điều hành Linux và triển khai mạng siêu phân giải RCAN chạy trên ZCU102. Dựa vào kết quả đạt được ở chương 4, đồ án đã cung cấp các thông tin cần thiết chứng tỏ cho việc triển khai thuật toán mạng học sâu xử lý hình ảnh trên nền tảng phần cứng cụ thể là Zynq Ultrascale+ MPSOC tốt hơn trên nền tảng phần mềm. Các chỉ số thu được như PSNR, SSIM, lược đồ histogram trong chương trước có thể chứng tỏ được hình ảnh đầu ra trên nền tảng Kaggle có chất lượng thấp hơn trên ZCU102.

Tuy nhiên, đồ án vẫn còn một số điểm cần phải cải thiện. Việc huấn luyện mạng trên máy ảo Ubuntu chạy hệ điều hành Linux gặp lỗi và không thành công nên phải huấn luyện trên môi trường Kaggle. Ngoài ra, ảnh siêu phân giải RCAN chỉ tạo ra một kích thước nhất định là 720x1280. Ảnh siêu phân giải được tạo ra có chất lượng càng cao khi ảnh có dung lượng càng thấp. Do đó, mô hình chỉ hợp lý cho các hình ảnh có dung lượng thấp, khi chạy mô hình cho ảnh có dung lượng lớn, thì ảnh đầu ra sẽ không đạt được kết quả mong muốn.

## 5.2. Hướng phát triển

Hệ thống siêu phân giải ảnh RCAN cần được tối ưu và hoàn thiện hơn nữa. Đầu tiên, cần phải cải thiện chất lượng ảnh ngõ ra theo ý muốn. Giúp mô hình linh hoạt hơn trong thực tiễn. Triển khai các mạng siêu phân giải khác như GAN, VDSR, … và so sánh với RCAN, từ đó suy ra phương pháp nào tối ưu nhất.

Phần cứng Zynq Ultrascale+ MPSoC - ZCU102 là một SoC mạnh mẽ với nhiều tiềm năng phát triển. Ngoài việc ứng dụng Deep Learning cho ZCU102, có thể ứng dụng cho nhiều lĩnh vực khác nhau như điều khiển các thiết bị ngoại vi bằng các chuẩn giao tiếp CAN, SPI,...Ứng dụng trong IoT cho việc nhận các giá trị từ cảm biến, hoặc trong các bài toán siêu cao tần,....

# TÀI LIỆU THAM KHẢO


[1] N. H. N. Uyen, "Implementation Of Sobel Filter Base On The Zynq-7000 SoC Development Board," Ho Chi Minh, 2022.

[2] G. K. Ravi, "Deploying Deep learning Image Super-Resolution Models in Xilinx Zynq MPSoC ZCU102," 2020.

[3] C. C. L. K. H. X. T. Chao Dong, "Learning a Deep Convolutional Network for Image Super-Resolution," 2014.

[4] Jiwon Kim, Jung Kwon Lee,Kyoung mu Lee, "Accurate Image Super-Resolutioon Using Very Deep Convolutional Networks," 2016.

[5] Jwon Kim, Jung Kwon Lee, Kyoung Mu Lee, "Deep-Recursive Convolutional Network for Image Super-Resolution," 2016.

[6] Yulun Zhang, Kunpeng Li, Kai li, Lichen Wang, Bineng Zhong, Yun Fu, "Image Super-Resolution Using Very Deep Residual Channel Attention Networks," 2018.

[7] Louise H. Crockett, David Northcote, Craig Ramsay, Fraser D. Robinson, Robert W. Stewart, Exploring Zynq® MPSoC With PYNQ and Machine Learning Applications, Scotland, 2019.

[8] Xilinx, ZCU102 Evaluation Board User Guide (Ug1182) (v.17), 2023.

[9] ARM, Learning the architecture - An introduction to AMBA AXI (v3.0), 2022.

[10] Xilinx, DPUCZDX8G for Zynq UltraScale+ MPSoCs - PG338 (v4.1), 2023.

[11] Xilinx, Zynq Ultrascale+ MPSoC Processing System v3.5 (PG201), 2023.

[12] Xilinx, Clocking WIzard v6.0 (PG065), 2022.

[13] Xilinx, "Processsor System Reset Module v5.0 (PG164)," 2015, p. 13.