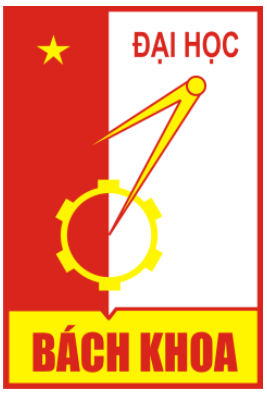
VIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI



BÁO CÁO ĐỒ ÁN HỆ NHÚNG

Đề tài:

TÌM HIỂU PCIE VÀ TRIỂN KHAI IP CORE XỬ LÝ ẢNH TRÊN KIT DE2i-150

Giảng viên hướng dẫn: ThS. Nguyễn Đức Tiến

Sinh viên thực hiện: Lê Thanh Tùng - MSSV: 20145095

Hà Nội 12/2018

LỜI GIỚI THIỆU

FPGA (Field Programmable Gate Array) hiện nay đang được sử dụng rộng rãi trong rất nhiều lĩnh vực, có thể kể đến như Multimedia, Networking, DSP, Video/Image Processing, Machine learning… Việc nghiên cứu, tìm hiểu FPGA ngày nay đã trở nên dễ dàng hơn đối với tất cả mọi người. FPGA không còn chỉ được sử dụng trong các phòng thí nghiệm mà đã trở nên phổ biến trong nhiều sản phẩm được ứng dụng trong thực tế và giá thành ngày càng rẻ. Các board thí nghiệm FPGA được cung cấp bởi rất nhiều hãng trên thế giới, có thể kể đến như Terasic, AVNET… cùng với đó là các chip FPGA được sản xuất bởi các hãng như Xilinx, Intel (Altera)… với rất nhiều mức giá và các tính năng khác nhau.

Trong môn học đồ án hệ nhúng, em đã có cơ hội được tìm hiểu về board DE2i-150 của hãng Terasic sử dụng chip FPGA Cyclone IV do Intel (Altera) sản xuất, xây dựng device driver và triển khai ứng dụng sử dụng PCIe link truyền dữ liệu qua lại giữa CPU Intel Atom và FPGA và xây dựng IP Core thực hiện phép nhân chập (convolution) với số fixed point, thử nghiệm kết quả với một số bộ lọc như: bộ lọc trung bình, bộ lọc Gaussian, bộ lọc Sobel, bộ lọc Prewitt… IP Core được xây dựng với đầu vào và đầu ra tương thích Avalon Video Streaming Interface.

Trong quá trình tìm hiểu và thực hiện, em có thể sẽ gặp phải nhiều thiếu sót, em mong nhận được phản hồi và góp ý từ thầy để bài làm được trở nên đầy đủ và hoàn thiện hơn.

MỤC LỤC

DANH MỤC HÌNH VẼ

1. AVALON INTERFACES

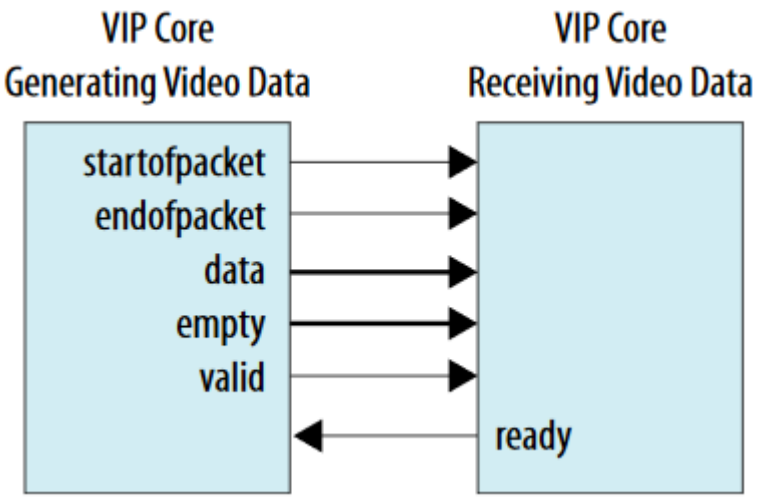
a. Giới thiệu Avalon Interfaces

Tài liệu *Avalon Interface Specifications* của Intel cung cấp thông tin chi tiết về 7 interface bao gồm:

- **Avalon Streaming Interface (Avalon-ST)**: truyền dữ liệu theo một chiều, không quan tâm đến địa chỉ dữ liệu, thường là dữ liệu ảnh, video, các tín hiệu số…
- **Avalon Memory Mapped Interface (Avalon-MM)**: đọc/ghi dữ liệu dựa trên địa chỉ, thường dùng để đọc ghi các thanh ghi điều khiển/trạng thái hoặc các vùng nhớ như RAM, ROM…
- **Avalon Interrupt Interface**: tín hiệu ngắt để điều khiển các module khác.
- **Avalon Clock Interface**: tín hiệu clock.
- **Avalon Reset Interface**: tín hiệu reset.
- **Avalon Conduit Interface**: tín hiệu hoặc nhóm các tín hiệu không nằm trong các interface bên trên, thường là các tín hiệu riêng biệt dùng để ghép nối với các tín hiệu từ IP Core khác.
- **Avalon Tri-state Conduit Interface**: tín hiệu kết nối tới các ngoại vi bên ngoài FPGA (input hoặc output).

b. Avalon Streaming/Video Streaming Interface

- Avalon Video Streaming là interface được sử dụng trong các IP Core xử lý dữ liệu ảnh/video được cung cấp sẵn bởi Intel. Do vậy để sử dụng cũng như thiết kế các IP Core xử lý số cho FPGA Intel cần tìm hiểu rõ interface này.
- Các tín hiệu được sử dụng trong Avalon Video Streaming Interface:



Hình 1: Avalon Video Streaming Interface

- **Nhận xét:**

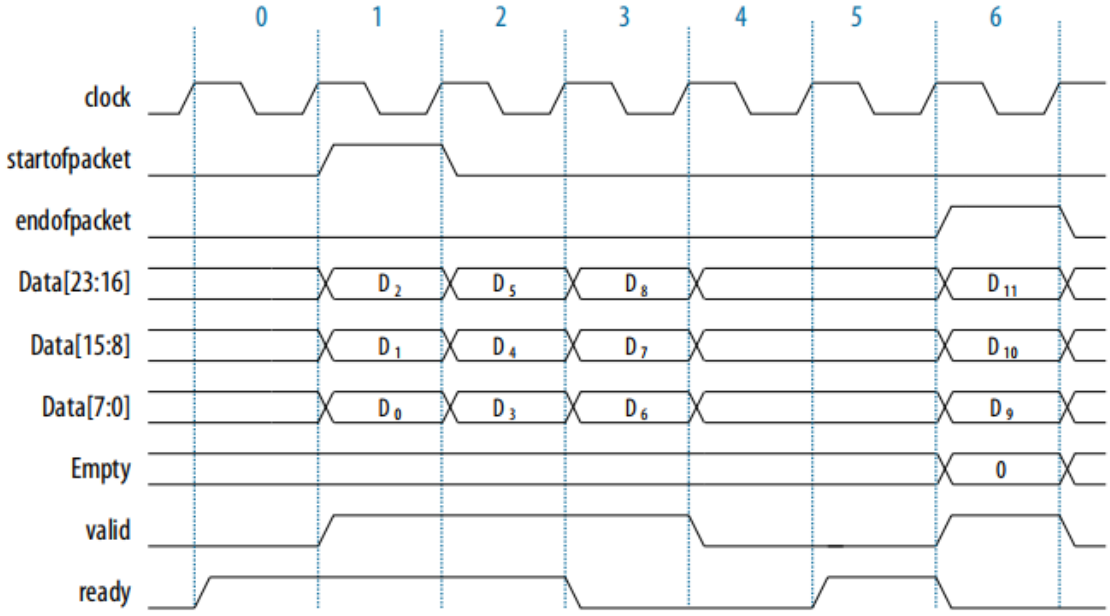
  + Dữ liệu truyền theo một chiều từ bên Source -> Sink

  + Không có bus thông tin địa chỉ (address), luồng dữ liệu được truyền một cách liên tục giữa 2 bên Source và Sink

- Ý nghĩa của các tín hiệu:

Tín hiệu	Ý nghĩa
startofpacket	Báo hiệu bắt đầu của 1 packet (ví dụ với dữ liệu ảnh là pixel đầu tiên của 1 frame)
endofpacket	Báo hiệu kết thúc của 1 packet (ví dụ với dữ liệu ảnh là pixel cuối cùng của 1 frame)
data	Dữ liệu được truyền/nhận giữa 2 bên (ví dụ với dữ liệu ảnh là 24 bit RGB tương ứng với 1 pixel của ảnh)
empty	Cho biết số lượng symbol trong gói dữ liệu đang truyền cuối cùng (tương ứng với endofpacket) là không hợp lệ
valid	Báo hiệu dữ liệu đang truyền là hợp lệ
ready	Báo hiệu trạng thái có thể nhận dữ liệu hay không

- Ví dụ về quá trình truyền dữ liệu theo đặc tả Avalon Video Streaming:



Hình 2: Avalon Streaming Interface Signals Waveform

- **<u>Nhận xét</u>:**

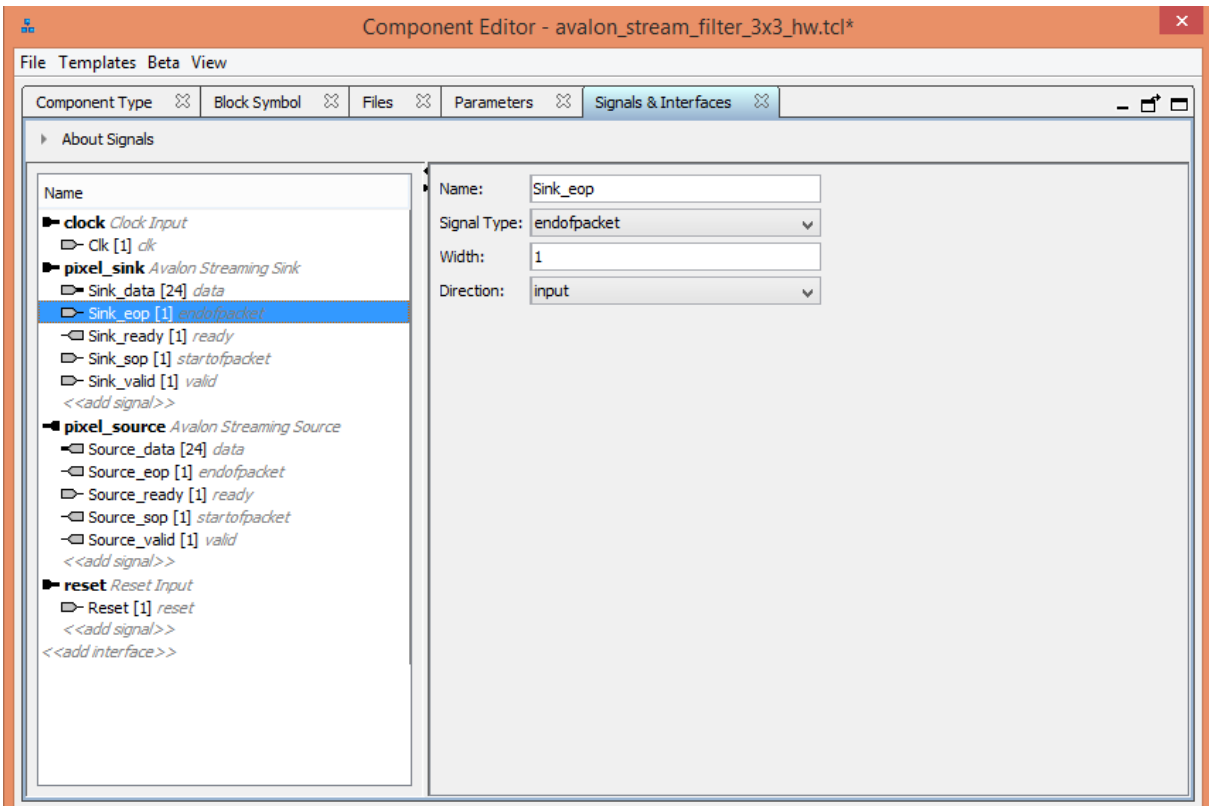
  + Tín hiệu *startofpacket* ở mức cao trong 1 chu kỳ duy nhất ứng với gói dữ liệu đầu tiên

  + Tín hiệu *endofpacket* ở mức cao trong 1 chu kỳ duy nhất ứng với gói dữ liệu cuối cùng

  + Tín hiệu *valid* ở mức cao trong khi bên gửi (Source) đang gửi ra dữ liệu

  + Tín hiệu *ready* ở mức cao trong khi bên nhận (Sink) có thể nhận dữ liệu, tín hiệu này có độ trễ là 1 chu kỳ, nên sau khi bên nhận (Sink) có *ready* ở mức cao, thì 1 chu kỳ sau tín hiệu *valid* từ bên gửi (Source) mới ở mức cao, tương tự khi bên nhận (Sink) chuyển trạng thái ready sang mức thấp.

- Trong công cụ Qsys, khi tự tạo thêm các IP Core tự thiết kế, ta có thể sắp xếp, định nghĩa các port tương ứng với ý nghĩa của nó:



Hình 3: Avalon Streaming Interface IP Design Example

- Với IP Core được thiết kế như trên, có thể thấy bao gồm các loại Avalon Interface sau đây:

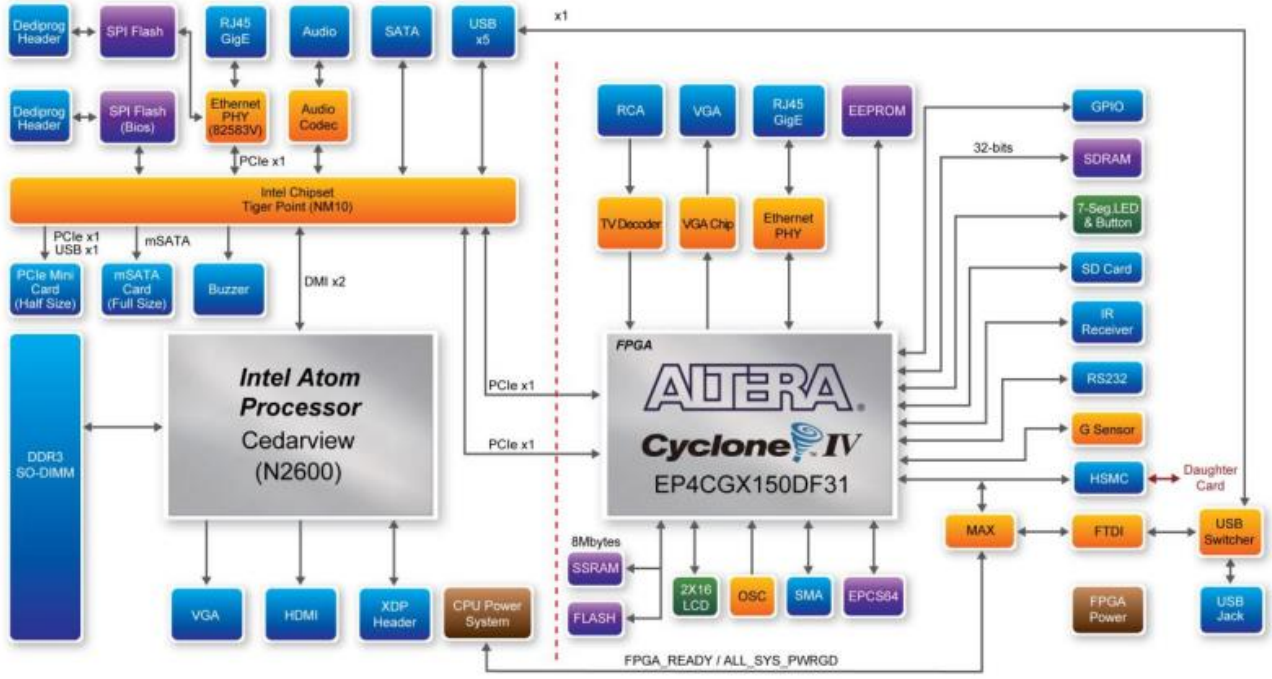
  + *Avalon Streaming Interface*: **pixel_sink** (đầu vào) và **pixel_source** (đầu ra)

  + *Avalon Clock Interface*: **clock**

  + *Avalon Reset Interface*: **reset**

  ➔ IP Core thực hiện xử lý tính toán với đầu vào là *Avalon Video Streaming* và đầu ra cũng là *Avalon Video Streaming*. Ở mục 3 sẽ nói chi tiết về thuật toán nằm bên trong IP Core này.

2. PCIE DE2i-150



Hình 4: DE2i-150 Block Diagram

a. IP Compiler for PCI Express

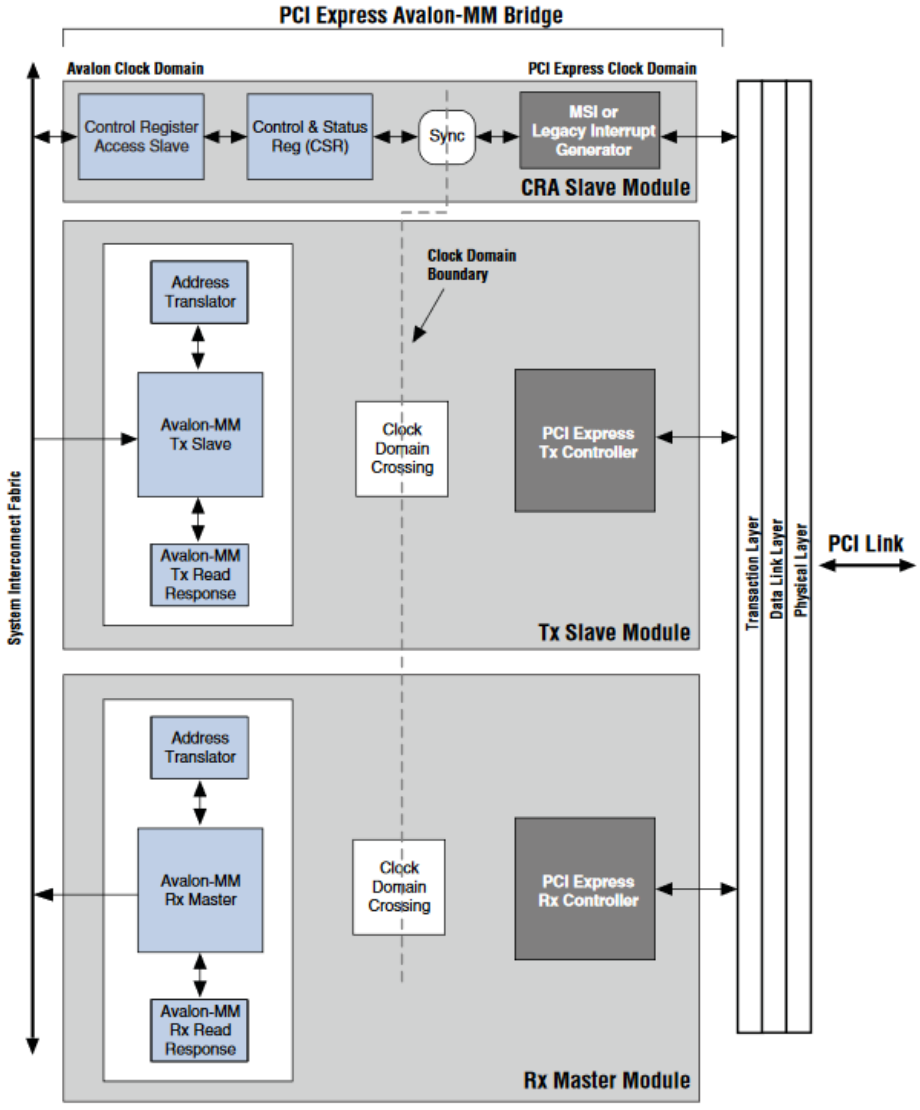
- **PCI Express** là giao thức kết nối tốc độ cao được sử dụng trong nhiều thiết bị phần cứng như: network adapter, graphic accelerator board, audio-video processor… PCIe tương thích ngược với các giao thức trước đó như PCI và PCI-X, nhưng có nhiều điểm khác biệt, đó là cơ chế truyền dựa theo gói (packet), nối tiếp (serial) và khả năng kết nối point-to-point giữa hai thiết bị. Hiệu năng hay tốc độ PCIe có thể tăng lên bằng cách tăng số lượng lane và các thế hệ (generation) PCIe được cải tiến tiếp theo.
- **IP Compiler for PCI Express** IP Core hỗ trợ việc triển khai PCIe protocol stack, bao gồm *transaction*, *data link* và *physical layer*. IP Core dễ sử dụng, dễ cấu hình và không yêu cầu license hay trả phí để sử dụng.
- **IP Compiler for PCI Express** sử dụng module *PCI Express Avalon-MM Bridge* làm cầu nối giữa PCIe link và hệ thống bus Avalon-MM trên FPGA. Điều này đơn giản hóa việc kết nối các IP Core khác vào hệ thống bus PCIe trong thiết kế với *Qsys Platform Designer*, bao gồm 3 thành phần:

+ **TX Slave Module:** chuyển đổi các request đọc/ghi bursting từ phía FPGA sang các request đọc/ghi theo gói trên hệ thống bus PCIe link.

+ **RX Slave Module:** chuyển đổi các request đọc/ghi theo gói nhận từ phía PCIe link sang các request đọc/ghi bursting trên hệ thống bus Avalon-MM trên FPGA.

+ **CRA Slave Module:** dùng để hỗ trợ việc đọc/ghi vào các thanh ghi điều khiển, trạng thái bên trong FPGA từ phía CPU, ví dụ như cấu hình *dynamic address translation* trên PCIe IP Core.

- **PCI Express Avalon MM-Bridge** hỗ trợ cơ chế chuyển đổi giữa 2 không gian địa chỉ Avalon-MM sang PCIe link và ngược lại. Việc chuyển đổi dựa trên *Base Address Registers* và *Andress Translation Table*.



Hình 5: PCI Express Avalon-MM Bridge

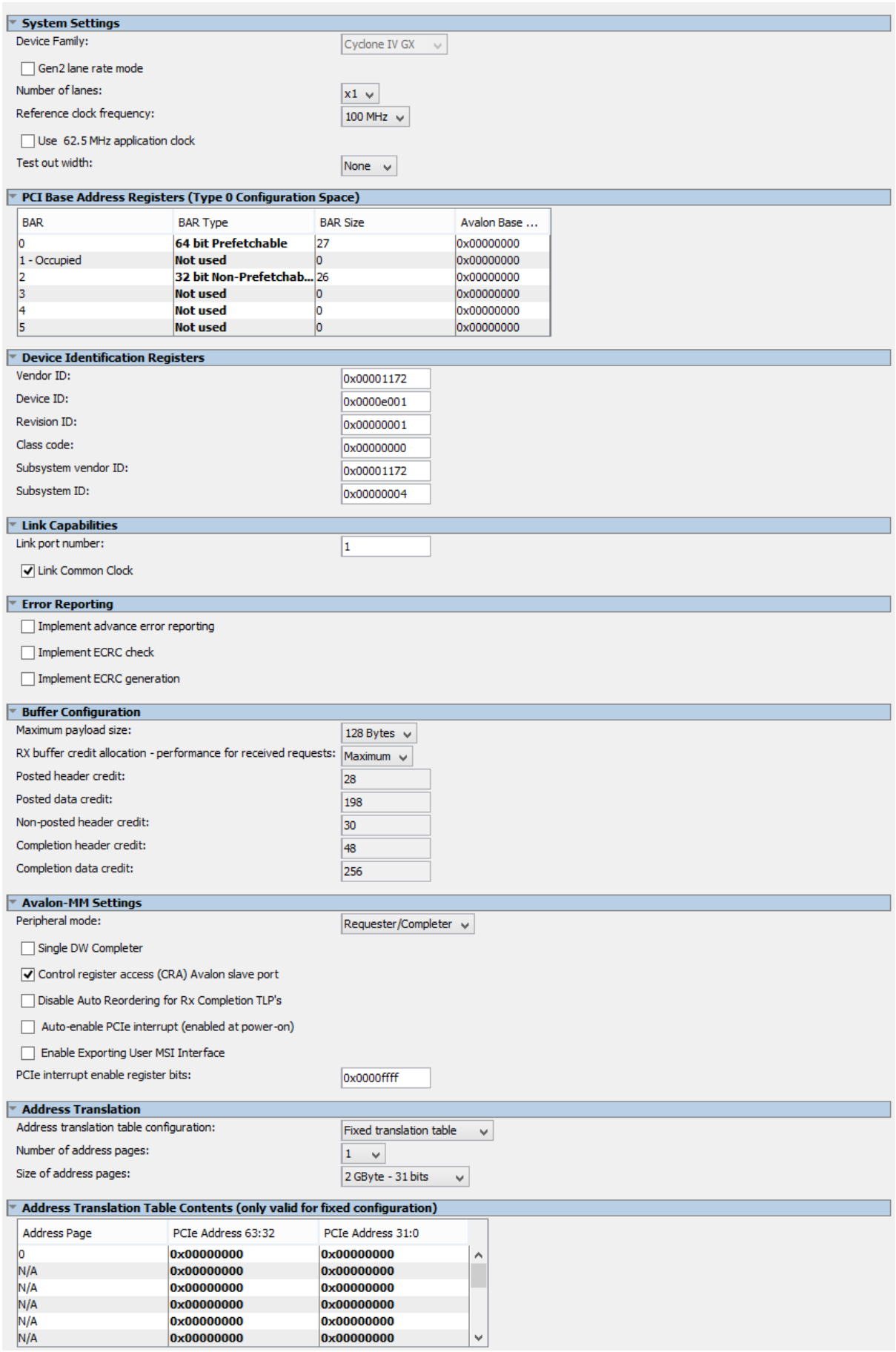
***** Cấu hình IP Core trong Qsys:**

- **Number of lanes:** x1, theo *DE2i-150 System Manual*, board có 2 đường kết nối PCIe Gen1 x1, do vậy số lượng lane tối đa chỉ là 1.
- **Test out width:** none, không sử dụng tín hiệu kiểm tra đầu ra.
- **Reference clock frequency:** tần số clock đưa vào IP Core là 100 MHz.
- **PCI Base Address Registers:** interface theo chuẩn *Avalon-MM Master*, kết nối với các *Avalon-MM Slave*, dùng để đọc/ghi các thanh ghi điều khiển, trạng thái các IP Core khác dùng trong thiết kế. BARs cho phép xác định địa chỉ cho các thanh ghi đó, qua đó người dùng biết địa chỉ truy cập các thanh ghi để điều khiển các IP Core qua PCIe link. Ở đây cấu hình bar0 là 64 bit Prefetchable và bar2 là 32 bit Non-Prefetchable.
- **Control register access (CRA) avalon slave port:** cần enable nếu sử dụng *dynamic translation table*.
- **Address Translation:** cấu hình mode chuyển đổi địa chỉ:

  + **Fixed Translation Table**: cấu hình cố định dải địa chỉ, có thể thay đổi địa chỉ bắt đầu của các trang nhớ trong bảng **Address Translation Table** bên dưới.

  + **Dynamic Translation Table**: cần cấu hình dải địa chỉ chuyển đổi bằng phần mềm, thường trực tiếp trong device driver, bảng **Address Translation Table** không có tác dụng.

  + Ở đây, có thể cấu hình *Fixed Translation Table*, 1 trang địa chỉ, kích thước trang địa chỉ là 2 GB, vì vùng nhớ vật lý (DRAM phía Intel Atom) có kích thước là 2GB.



Hình 6: IP Compiler for PCI Express Configurations

b. Scatter-Gather DMA

- Scatter-Gather DMA (SG-DMA) làm nhiệm vụ chuyển dữ liệu qua lại giữa các vùng nhớ, hoặc đọc dữ liệu từ vùng nhớ đưa ra Streaming interface và ngược lại. Trong thiết kế này, SG-DMA được sử dụng để trao đổi dữ liệu giữa DRAM CPU và DRAM FPGA.
- SG-DMA hoạt động theo các descriptor được lưu trữ trong buffer, tùy vào kích thước buffer có thể lưu trữ một hay nhiều descriptor.
- Định dạng của mỗi descriptor cơ bản bao gồm: địa chỉ đọc (nguồn), địa chỉ ghi (đích), kích thước và trường *control* cho phép một số thiết lập khác như: ngắt khi descriptor đã được xử lý xong, ngắt khi xảy ra lỗi…

	Byte Lanes			
Offset	**3**	**2**	**1**	**0**
0x0	Read Address[31..0]			
0x4	Write Address[31..0]			
0x8	Length[31..0]			
0xC	Control[31..0]			

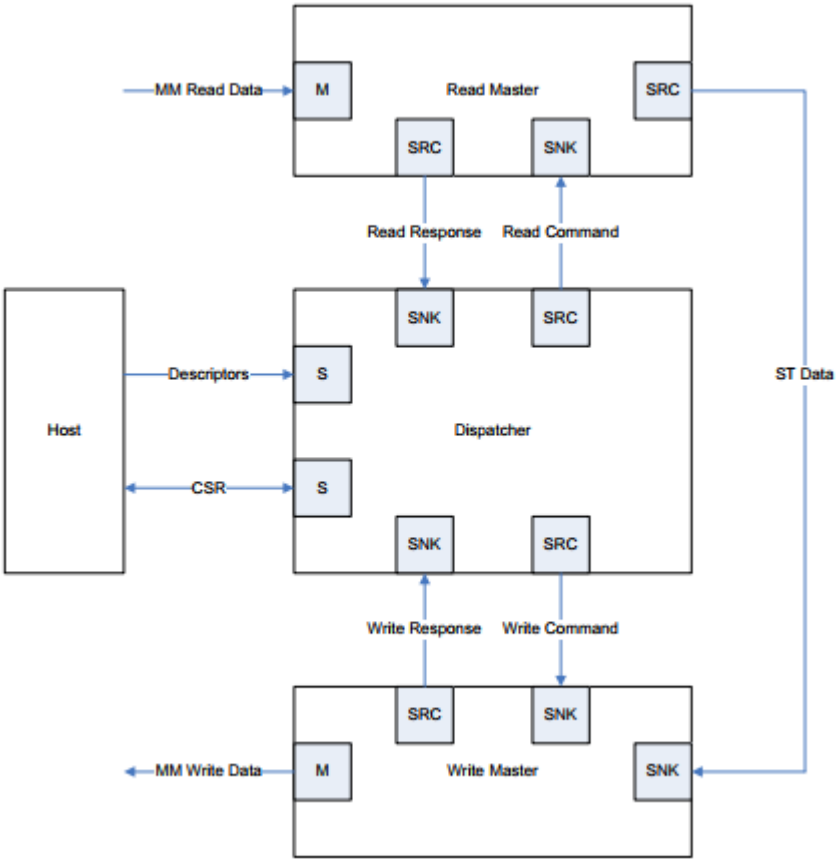
Hình 7: Standard descriptor format

- Scatter-Gather DMA (SG-DMA) có 3 mode hoạt động:

  + **Memory-Mapped to Streaming**: đọc dữ liệu từ một vùng nhớ (DRAM, On Chip Memory RAM/ROM) và đầu ra là Streaming interface.

  + **Streaming to Memory-Mapped**: đọc dữ liệu đầu vào từ Streaming interface và đầu ra là Memory-Mapped interface.

  + **Memory-Mapped to Memory-Mapped**: đọc dữ liệu từ vùng nhớ này và ghi sang vùng nhớ khác.
- CPU ghi vào buffer lưu trữ các descriptor, và sau đó SG-DMA sẽ xử lý theo các descriptor đó.
- Việc khởi tạo SG-DMA và các descriptor được thực hiện trong device driver, datasheet của IP Core SG-DMA có nêu rõ cấu trúc descriptor, địa chỉ offset của các thanh ghi cần quan tâm để sử dụng SG-DMA.



Hình 8: Memory-Mapped to Memory-Mapped SG-DMA Architecture

c. Linux device driver

- Trong Linux kernel, dù chưa có device driver chức năng cụ thể nào cho PCI/PCIe device, các thông tin theo khuôn dạng quy định của PCI/PCIe device vẫn được thu thập và đưa vào PCI Core trong tầng nhân.
- PCI Core trong tầng nhân cung cấp một số function, macro và struct được xây dựng sẵn để làm việc với PCI/PCIe, để sử dụng cần include header file *linux/pci.h*.
- Các struct cần sử dụng trong device driver:

  struct pci_dev;

  ➔ Cấu trúc đại diện cho PCI device.

  struct pci_driver;

  ➔ Cấu trúc đại diện cho PCI driver, bao gồm thông tin về tên, về các device được hỗ trợ, thông tin về các hàm probe, remove, suspend, resume…

  struct pci_device_id;

➔ Cấu trúc thông tin mô tả PCI device, bao gồm VENDOR ID và DEVICE ID.

- Một số macro được sử dụng trong device driver:

  PCI_DEVICE

  ➔ Tạo một cấu trúc pci_device_id với tham số VENDOR ID và DEVICE ID.

  MODULE_DEVICE_TABLE

  ➔ Liên kết driver (kernel module) với PCI/PCIe device có trong hệ thống với tham số truyền vào là struct/mảng struct pci_device_id.

- Struct pci_driver cần có thông tin của hai hàm sau:

  + probe: được gọi bởi PCI Core khi có liên kết giữa PCI/PCIe device và driver

  + remove: được gọi bởi PCI Core khi struct pci_dev bị remove khỏi hệ thống driver bị remove khỏi kernel

- Một số function được sử dụng trong device driver:

  pci_enable_device

  ➔ Gọi hàm này cho phép thiết bị PCI/PCIe hoạt động

  pci_disable_device

  ➔ Gọi hàm này dừng hoạt động của PCI/PCIe device

  pci_resource_start

  ➔ Trả về địa chỉ của BARs (từ BAR0 đến BAR5)

  pci_resource_len

  ➔ Trả về kích thước theo byte của BARs (từ BAR0 đến BAR5)

- Ví dụ khai báo các cấu trúc với DE2i-150:

```
#define DIR_BAR_NR                      0   //bar 0
#define CSR_BAR_NR                      2   //bar 2
...
static struct pci_device_id pci_ids[] = {
        { PCI_DEVICE(0x1172, 0xe001), },
        { 0, }
};
```

```
MODULE_DEVICE_TABLE(pci, pci_ids);

static int pci_probe(struct pci_dev *dev, const struct pci_device_id *id);
static void pci_remove(struct pci_dev *dev);
static struct pci_driver pci_driver = {
        .name     = "alterahello",
        .id_table = pci_ids,
        .probe    = pci_probe,
        .remove   = pci_remove,
};
static int pci_probe(struct pci_dev *dev, const struct pci_device_id *id) {
        int err;
        printk(KERN_DEBUG "altera : Device 0x%x has been found at"
                " bus %d dev %d func %d\n",
                dev->device, dev->bus->number, PCI_SLOT(dev->devfn),
                PCI_FUNC(dev->devfn));

        err = pci_enable_device (dev);
        printk("pci_enable_device = %d rev=%d\n",err,dev->revision);
        if( err ) {
                printk(" pci device enable failed \n");
                return err;
        }
        printk(" pci/fpga device rev id = 0x%x\n",dev->revision);
        printk(KERN_DEBUG "bar0 start at: %llx\n", pci_resource_start(dev,
DIR_BAR_NR));
        printk(KERN_DEBUG "bar0 len: %lld\n", pci_resource_len(dev,
DIR_BAR_NR));
        bar0 = ioremap(pci_resource_start(dev, DIR_BAR_NR),
pci_resource_len(dev, DIR_BAR_NR));
        printk(KERN_DEBUG "bar2 start at: %llx\n", pci_resource_start(dev,
CSR_BAR_NR));
        printk(KERN_DEBUG "bar2 len: %lld\n", pci_resource_len(dev,
CSR_BAR_NR));
```

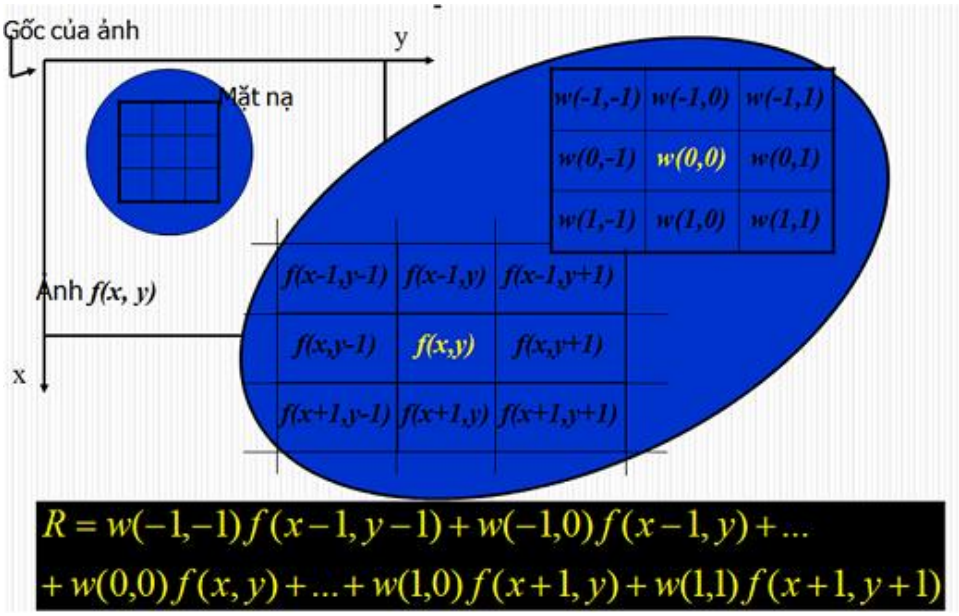
```
        bar2 = ioremap(pci_resource_start(dev, CSR_BAR_NR),
pci_resource_len(dev, CSR_BAR_NR));
        return 0;
}
static void pci_remove(struct pci_dev *dev) {
        iounmap(bar0);
        iounmap(bar2);
        pci_disable_device(dev);
```

- Phần khai báo đã ánh xạ địa chỉ vật lý của BARs vào kernel virtual address bằng hàm ioremap. Sử dụng địa chỉ offset trong thiết kế Qsys Platform Designer, ta có thể truy cập để điều khiển SG-DMA, Video DMA Controller… và các IP Core khác nếu muốn. Ví dụ tạo descriptor để SG-DMA đọc từ vùng nhớ SDRAM FPGA:

```
// ADDR_DES là địa chỉ offset của descriptors fifo, liên kết với bar2 trong Qsys
#define ADDR_DES                0x02000020
write( bar2, ADDR_DES, ADDR_RM);  //source address
write( bar2, ADDR_DES + 4, __pa(target_to) ); // dest address
write( bar2, ADDR_DES + 8, len); // size to transfer
write( bar2, ADDR_DES + 0xC, control_bits | (1<<31) ); //commit descriptor to fifo
```

3. IP CORE AVALON STREAM FILTER

a. Khái niệm convolution



Hình 9: Convolution overview

- Các thao tác biến đổi trên ảnh hay dò tìm đặc trưng của ảnh được thực hiện thông qua việc lọc ảnh, bản chất chính là thực hiện các phép nhân chập (convolution) trên ảnh.
- Khi đó, để thực hiện một phép biến đổi hay dò tìm đặc trưng trên ảnh, chúng ta chỉ cần tiến hành "nhân chập" ảnh đầu vào với một ma trận hay cửa sổ nhân chập gọi là "kernel". Toàn bộ các pixel trên ảnh sẽ được tiến hành nhân chập với cửa sổ nhân chập (tâm của cửa sổ nhân chập sẽ được đặt trùng vào vị trí của pixel đang được tính nhân chập), làm thay đổi giá trị của pixel ban đầu.
- Dưới đây là công thức tính nhân chập:

$$\boldsymbol{Y(m,n) = X(m,n) * H(k,l) = \sum_{k=-r}^{r} \sum_{l=-r}^{r} X(m-k, n-l) H(k,l)}$$

Trong đó:

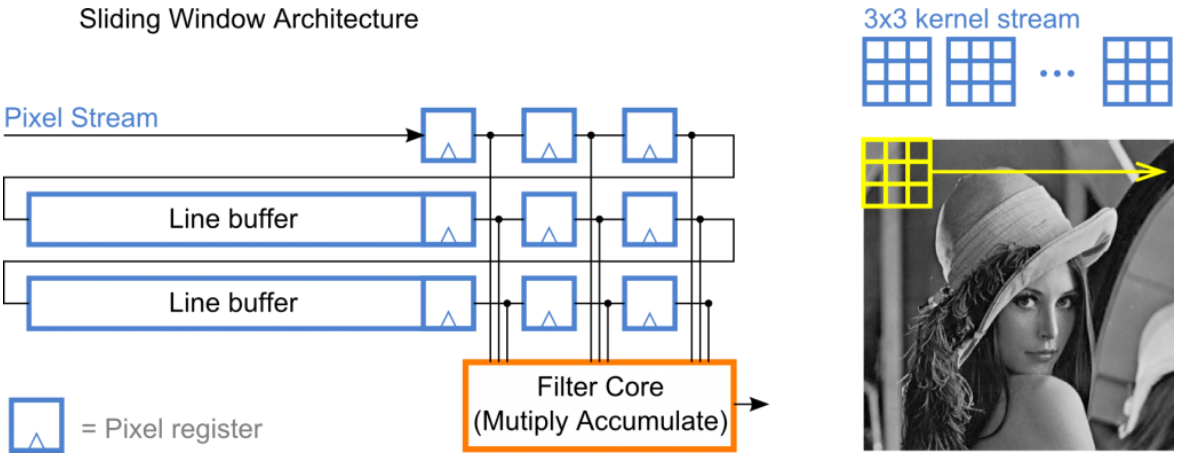
+ X(m,n) là ma trận ban đầu của ảnh kích thước mxn

+ H(k,l) là ma trận hạt nhân của phép nhân chập hay còn gọi là mặt nạ

+ Y(m,n) là ma trận đầu ra của phép nhân chập giữa X và H

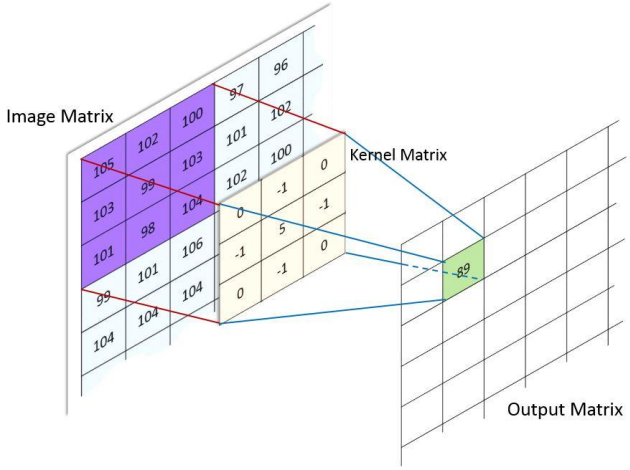
b. Xây dựng IP Core nhân chập

- Line buffer:



Hình 10: Line buffer

- Các pixel từ ảnh được đưa vào lần lượt theo chiều từ trái sang phải, từ trên xuống dưới, line buffer đóng vai trò lưu trữ lại các pixel và tạo số chu kỳ trễ phù hợp (latency) để thực hiện nhân chập, ví dụ: ảnh đầu vào có kích thước 640x480 và kernel có kích thước 3x3, đến khi pixel hàng 3 cột 3 được đưa vào thì phép nhân chập mới hoàn thành xong cho pixel đầu tiên ảnh đầu ra.

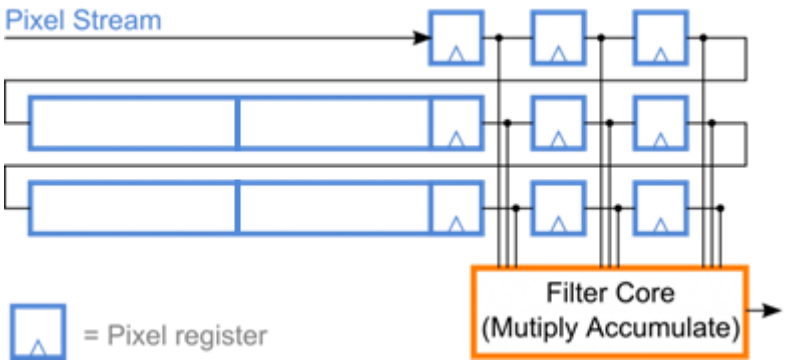


Hình 11: Image input and kernel matrix

- Ví dụ line buffer với ảnh đầu vào 5x5 như dưới đây:

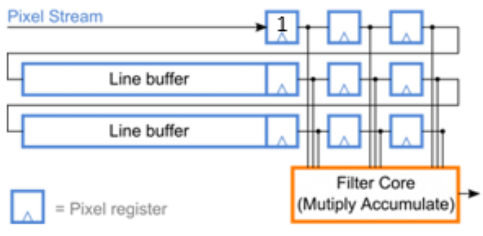
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

- Với kernel kích thước 3x3, line buffer cần có kích thước như dưới đây:

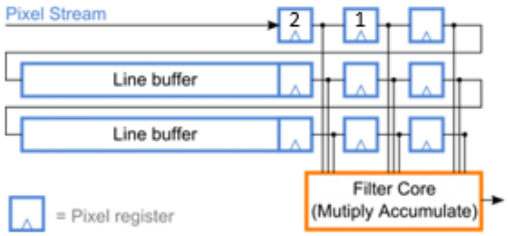


Hình 12: Line buffer example

**** Trạng thái line buffer lần lượt từ pixel đầu tiên:*

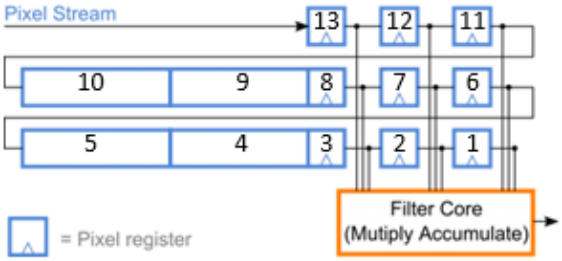


(Pixel đầu vào đầu tiên)

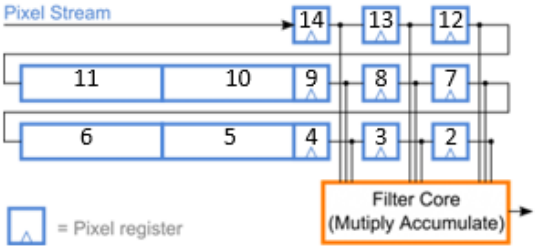


(Pixel đầu vào thứ hai)

...



(Tính xong pixel đầu ra đầu tiên)



(Tính xong pixel đầu ra thứ hai)

Hình 13: Line buffer progress states

c. Số fixed-point

- Thay vì các phép toán với số floating-point, IP Core được thiết kế sử dụng số fixed-point, trong MATLAB có hai hàm *sfi()* và *ufi()* cho phép tạo số fixed-point từ một số biết trước, ta có thể dùng hai hàm này để thu được các hệ số kernel khi cấu hình IP Core. Ví dụ lấy hệ số Gaussian kernel kích thước 3x3 với tham số $\sigma = 1$, số fixedpoint có word length là 32 bit và fractional length là 15 bit:

  >> **H = sfi(fspecial('Gaussian', 3, 1), 32, 15);** *% H is Gaussian kernel*

>> **H.bin** *% show all values with binary format*

- Với số fixed-point, thao tác cộng, nhân diễn ra tương tự như với số nguyên (integer), vì số lượng bit dành cho phần thập phân đã biết trước. Ví dụ nhân số fixed-point (unsigned) với số nguyên (unsigned):

  5.625 = 101.101

  5 = 101

  101101

  000000

  101101

  11100.001 = 28.125

d. Cấu hình IP Core

- IP Core được xây dựng tương thích với *Avalon Video Streaming Interface*, bao gồm một đầu vào *Avalon Video Streaming Sink Interface* và một đầu ra *Avalon Video Streaming Source Interface*
- Bên trong Qsys, các tham số cho phép cấu hình bởi IP Core:

  + IMAGE_HEIGHT: chiều cao của ảnh

  + IMAGE_WIDTH: độ rộng của ảnh

  + KERNEL_SIZE: kích thước kernel đầu vào là 3x3 (cố định 3x3)

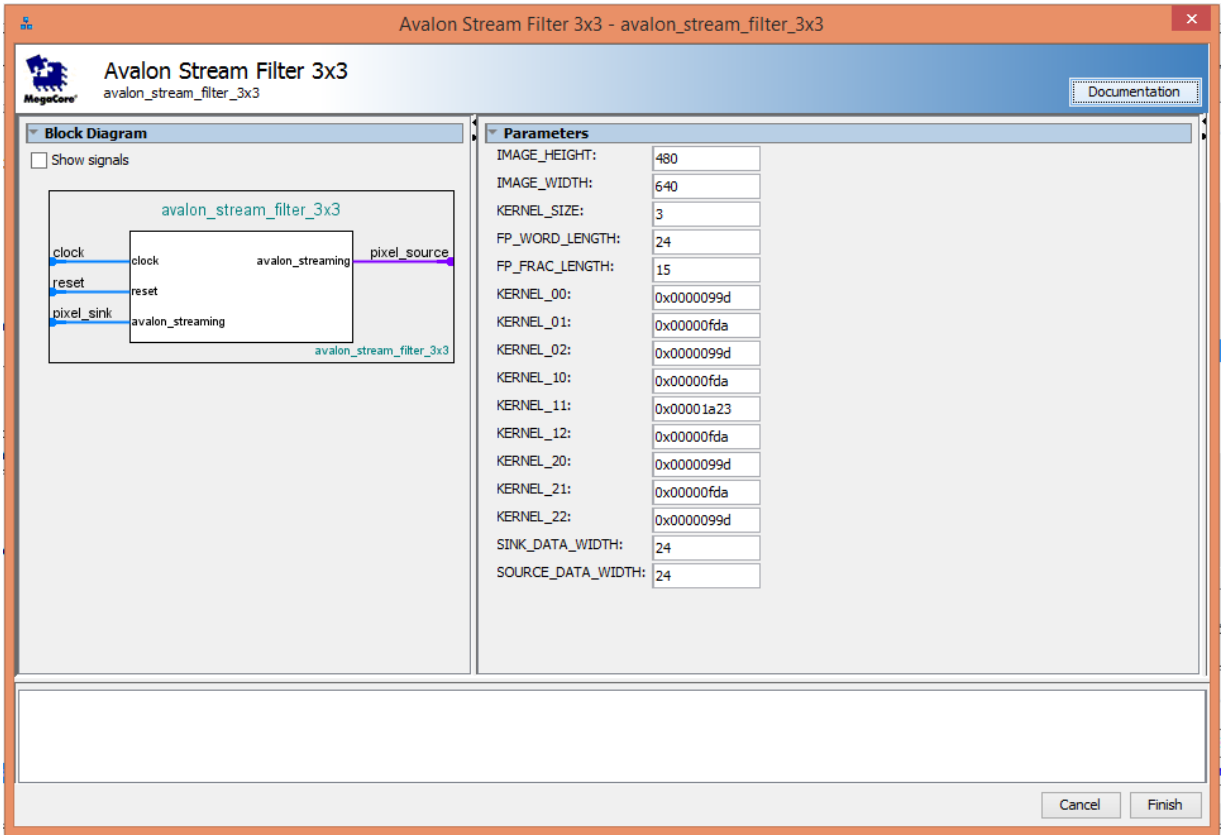
  + FP_WORD_LENGTH: word length của số fixed point

  + FP_FRAC_LENGTH: fractional length của số fixed pont

  + KERNEL_xx: các giá trị tương ứng của kernel (dùng MATLAB để tính)

  + SINK_DATA_WIDTH: độ rộng bus dữ liệu đầu vào

  + SOURCE_DATA_WIDTH: độ rộng bus dữ liệu đầu ra



Hình 14: Avalon Stream Filter 3x3 Configurations

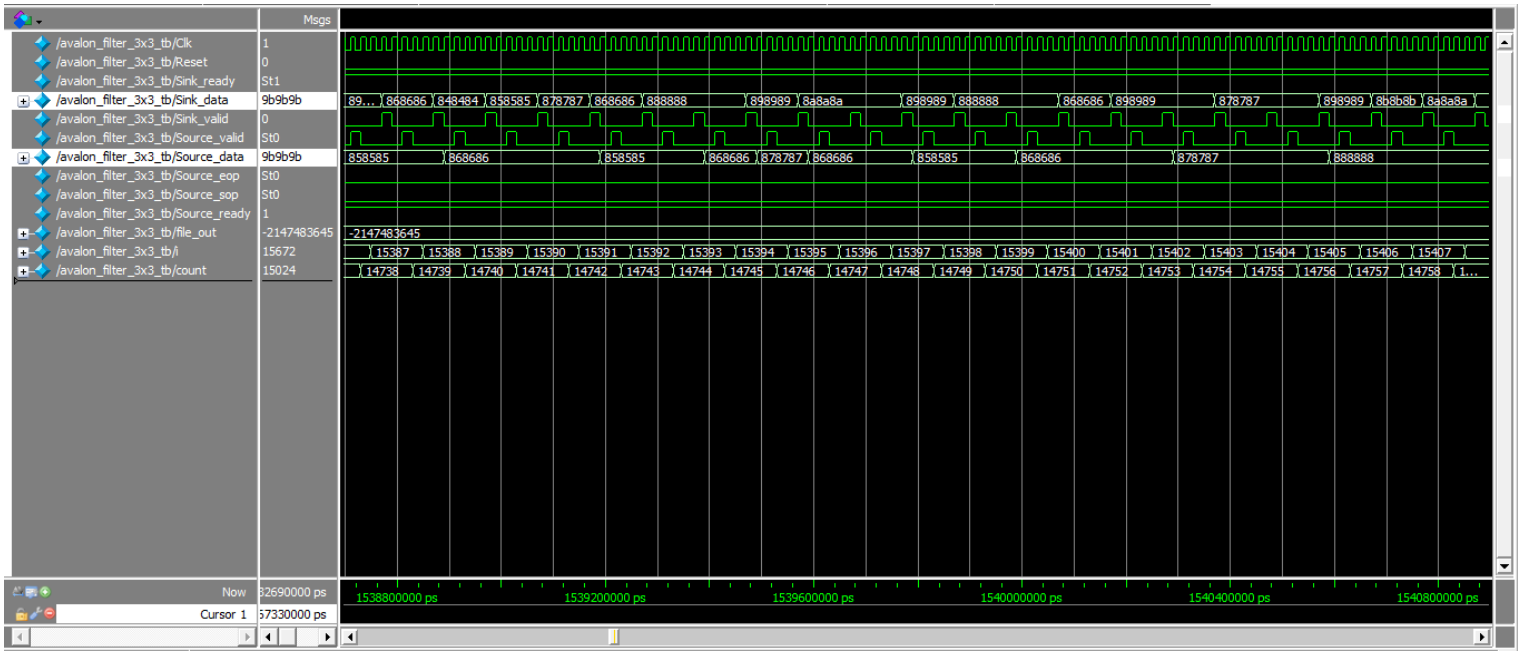
e. Testbench

- Xây dựng testbench kiểm tra kết quả trên ModelSim:

  + **Input**: *image_in.txt*, chứa các pixel của ảnh đầu vào 640x480, mỗi dòng là giá trị binary của một pixel (24 bit)

  + **Output**: *image_out.txt*, chứa các pixel của ảnh đầu ra 640x480, mỗi dòng là giá trị binary của một pixel đầu ra tương ứng (24 bit)

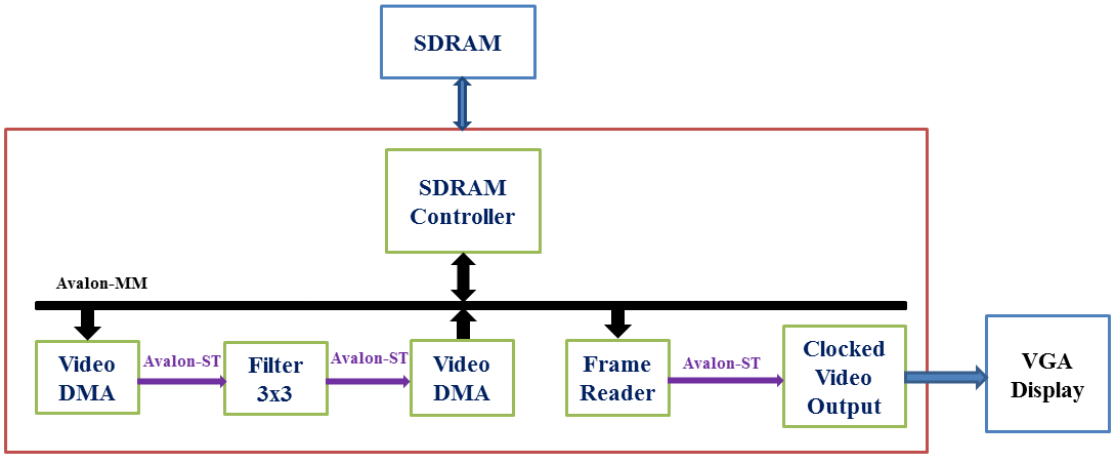
  + **Waveform**:



Hình 15: Waveform of testbench

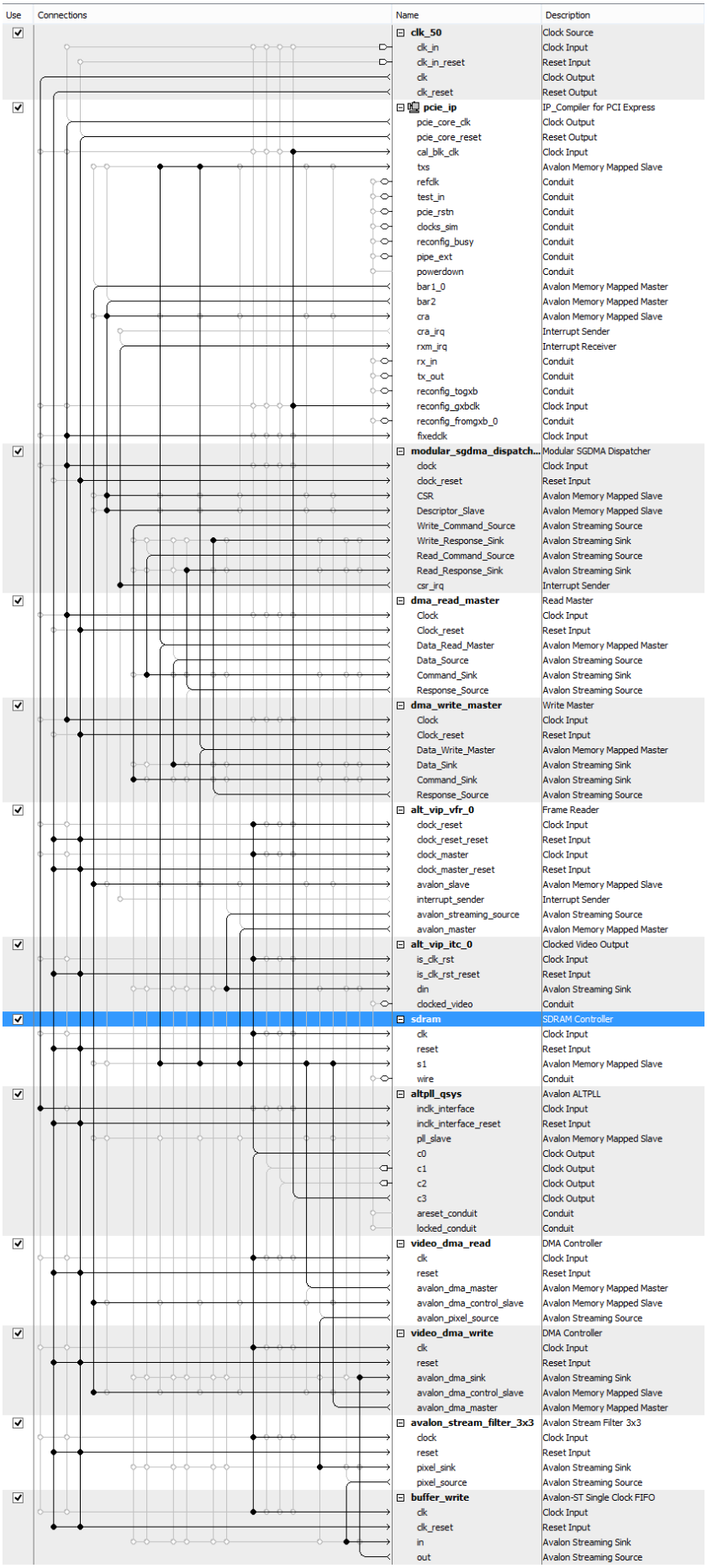
4. KẾT QUẢ THỬ NGHIỆM

a. Block diagram



Hình 16: Block diagram of the convolution and display system

- **SDRAM Controller**: thực hiện việc đọc/ghi vào SDRAM, cung cấp interface Avalon-Memory Mapped cho phép ghép nối với các IP Core khác.
- **Video DMA**: có 2 mode, mode *From Memory To Stream* đọc dữ liệu từ DRAM và đưa dữ liệu ra theo định dạng *Avalon Video Streaming*, mode *From Stream To Memory* lấy dữ liệu Avalon Video Streaming và ghi vào DRAM.
- **Frame Reader**: đọc dữ liệu từ DRAM và chuyển dữ liệu cho **Clocked Video Output IP**, cho phép thiết lập 2 frame buffer và lựa chọn hiển thị 1 trong 2.
- **Clocked Video Output**: nhận dữ liệu từ **Frame Reader IP** và tạo ra các tín hiệu RGB, HSync, VSync phù hợp để hiển thị ra cổng VGA.
- **Filter 3x3**: thực hiện phép toán nhân chập (convolution) với kernel kích thước 3x3 với đầu vào và đầu ra đều là Avalon Video Streaming (đã trình bày thuật toán chi tiết ở mục 3).



Hình 17: System diagram in Qsys Platform Designer

b. Kết quả thử nghiệm

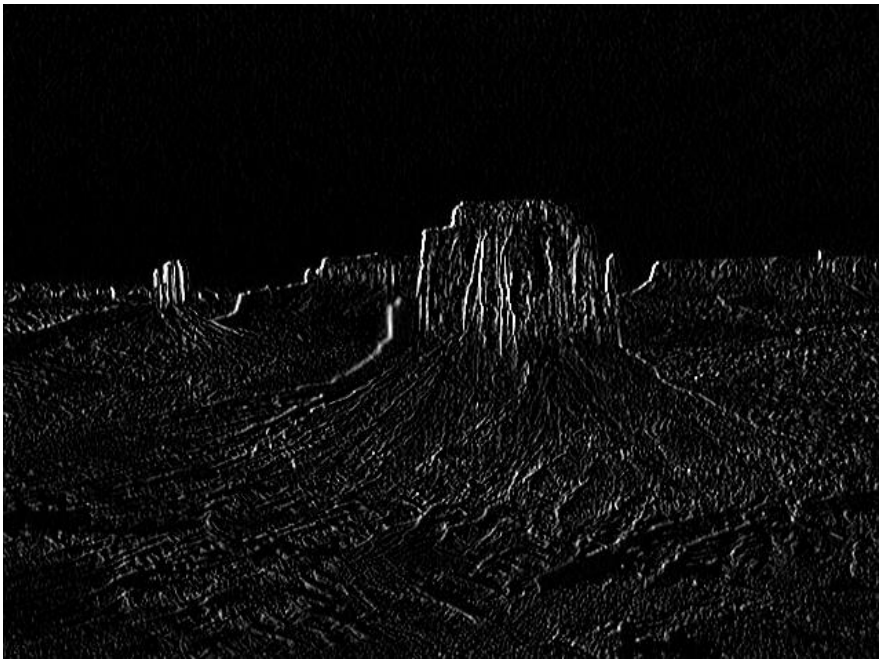
- Ví dụ thử nghiệm với ảnh đầu vào:

Hình 18: Input Image

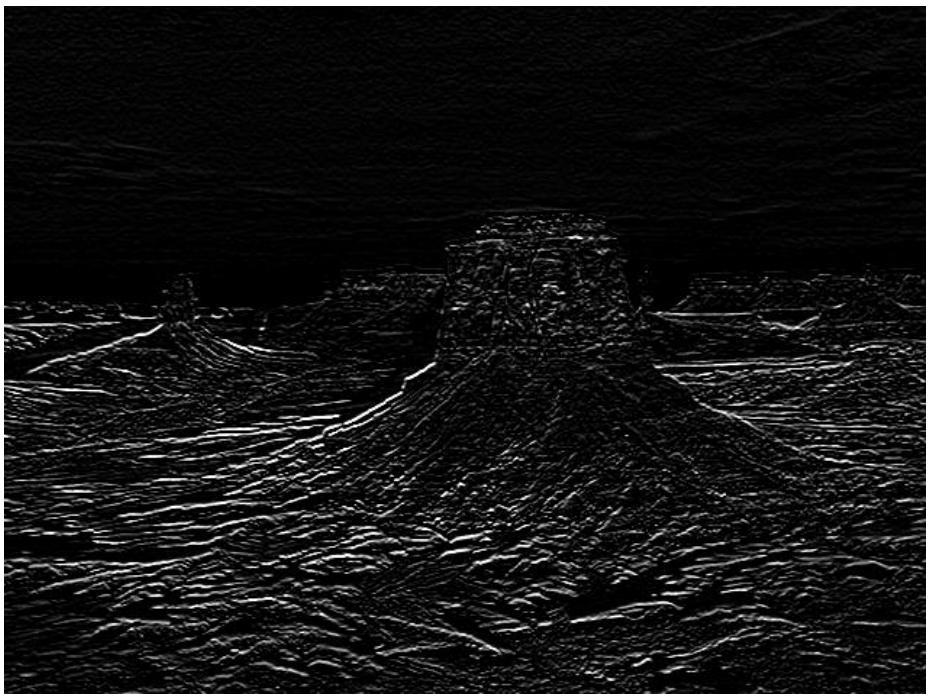
- Với IP Core đã thiết kế, ta có thể thử nghiệm với nhiều bộ lọc 3x3 khác nhau, một số bộ lọc đã thử nghiệm:

Hình 19.a: Gaussian 3x3, б=1.0

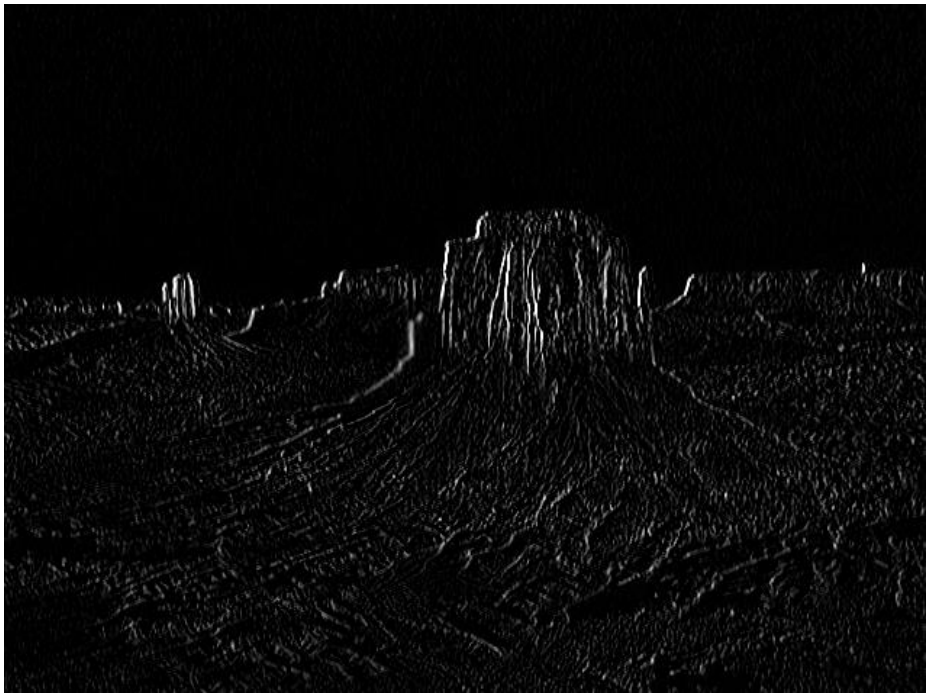
Hình 19.b: Gaussian 3x3, б=2.0



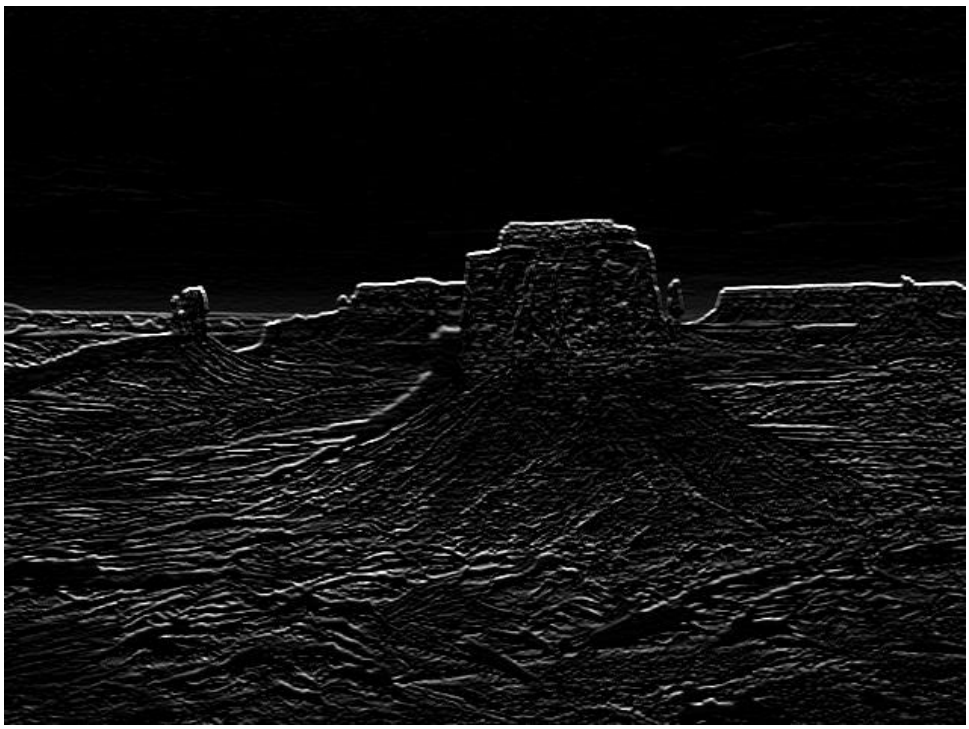
Hình 19.c: Sobel X



Hình 19.d: Sobel Y



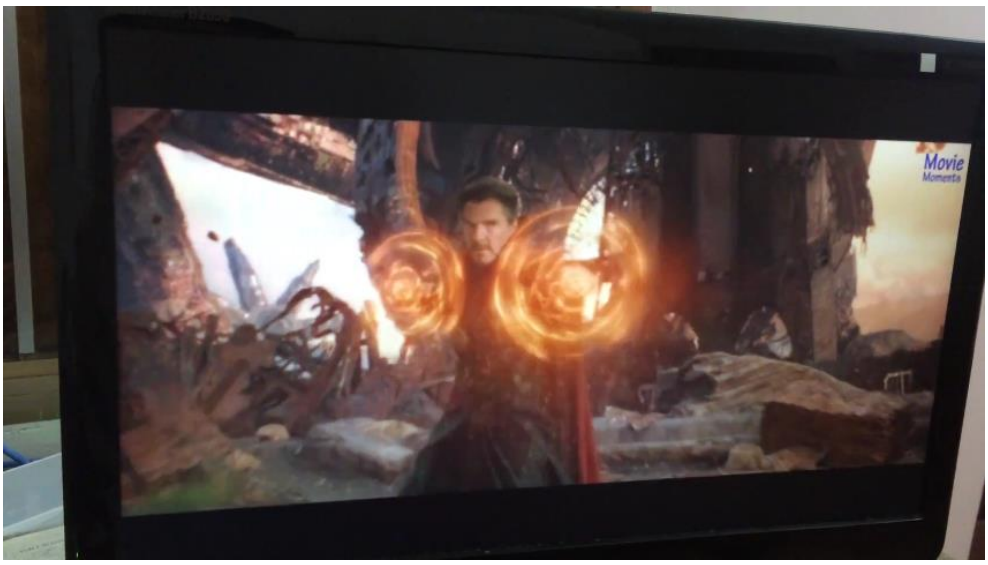
Hình 19.e: Prewitt X



Hình 19.f: Prewitt Y

Hình 19.g: Mean 3x3

- Kết quả tương tự đạt được khi áp dụng với video streaming:



Hình 20: Video streaming input

c. So sánh

- Bảng bên dưới đây so sánh một vài thông số của việc triển khai thuật toán nhân chập giữa FPGA và CPU với mục đích tham khảo:

	Cyclone IV FPGA	**CPU (i5 4590 3.3 GHz)**
Cài đặt	Phức tạp	Đơn giản
Thời gian	6 ms (150 MHz)	16 ms (C++)
Năng lượng	~ 1.8 W	~ 84 W

KẾT LUẬN

Tích chập và lọc ảnh là thuật toán cơ bản thường xuyên được dùng trong tiền xử lý ảnh. Trong đồ án môn học này, em đã xây dựng IP Core theo chuẩn vào ra Avalon Video Streaming Interface, thực hiện phép nhân chập (convolution) với kernel có kích thước 3x3 và thực hiện các thử nghiệm với ảnh 640x480. Bên cạnh đó, để có thể ghép nối IP Core đã xây dựng và thử nghiệm hiển thị ra VGA trên board DE2i-150, em cũng đã trình bày phần tìm hiểu của mình về PCIe - hệ thống bus được sử dụng để trao đổi thông tin qua lại giữa CPU và FPGA, Scatter-Gather DMA IP Core đóng vai trò truyền dữ liệu từ DRAM phía CPU sang DRAM phía FPGA và ngược lại. Để hiển thị ra VGA cần có một số module khác như Frame Reader IP, dùng để đọc dữ liệu từ DRAM (FPGA) và gửi sang Clocked Video Output IP để tạo ra các tín hiệu RGB, HSync, VSync phù hợp. Em đã thử nghiệm với ảnh và với dữ liệu video streaming, kết quả thử nghiệm thu được có độ chính xác tương tự với tính toán trên phần mềm MATLAB khi có cùng tham số cho số fixed-point. Ngoài ra, để thực hiện chạy thử nghiệm trên board cần có device driver trên Ubuntu Linux. Phần device driver và application được xây dựng tương đối đơn giản nhằm mục đích thử nghiệm kết quả thiết kế. Cuối cùng, một vài thông số về thời gian, năng lượng được so sánh khi thực hiện giữa CPU và FPGA nhằm mục đích tham khảo.

Em xin chân thành cảm ơn thầy Nguyễn Đức Tiến đã tạo điều kiện về board thí nghiệm DE2i-150, về tài liệu thiết kế để giúp em hoàn thành đồ án môn học này. Em rất mong nhận được phản hồi và góp ý từ thầy để bài làm và báo cáo được hoàn thiện hơn.

TÀI LIỆU THAM KHẢO

1. Avalon Interface Specifications, *Intel*: *https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf*
2. Video and Image Processing Suite User Guide, *Intel*: *https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_vip.pdf*
3. An FPGA Based Processor For Convolutional Networks, *Clément Farabet, Cyril Poulet, Jefferson Y. Han, Yann LeCun* *http://yann.lecun.com/exdb/publis/pdf/farabet-fpl-09.pdf*
4. Linux device driver, 3rd edition, *Jonathan Corbet, AlessandroRubini, and Greg Kroah-Hartman* *https://bootlin.com/doc/books/ldd3.pdf*
5. Một số tài liệu tham khảo khác:

   Qsys System Design Tutorial

   DE2i-150_v.2.2.0_SystemCD

   DE2i-150_Getting Started Guide.pdf

   DE2i-150_FPGA_System_manual.pdf

   IP Compiler for PCI Express User Guide