Structura Sistemelor de Calcul

S.L. Dr. Ing. Mircea Paul Muresan

Msc. Ing. Gabriel Cireap

Ing. Sopterean Andrei

Ing. Corpodean Darius

Ing. Darius Stan

Obiectivul General

Cunoașterea structurii și proiectarea unor componente optimizate ale sistemelor de calcul moderne

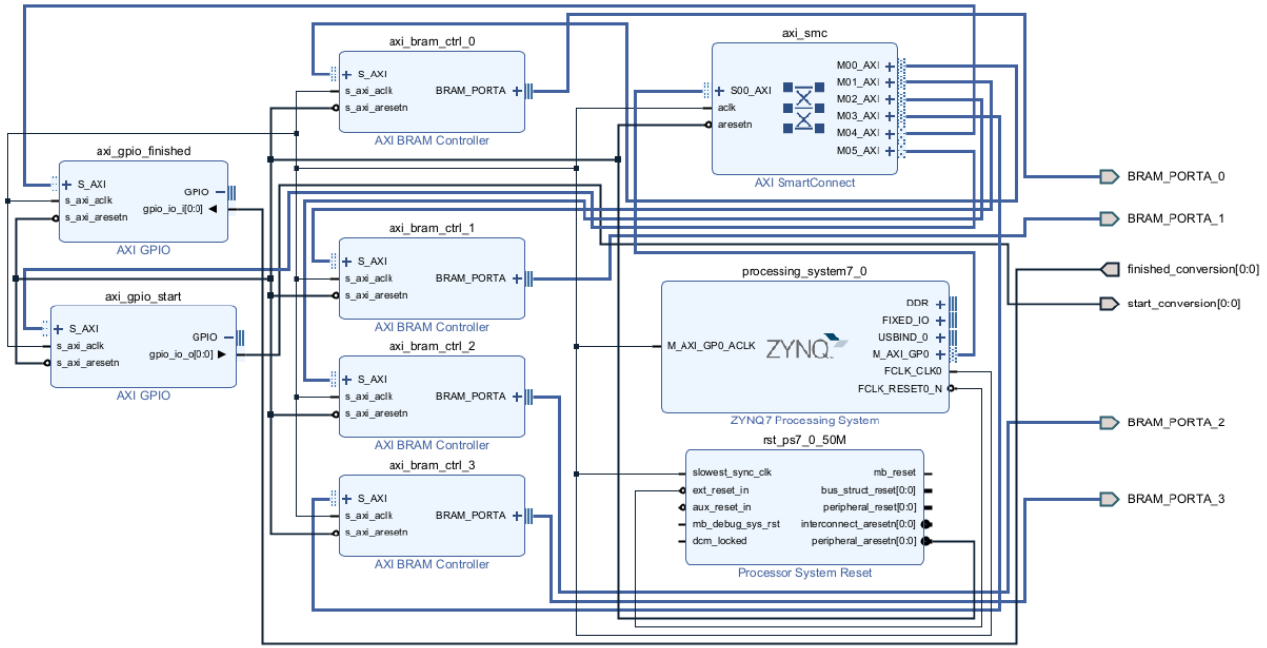
Obiective Teoretice

- Cunoașterea unor indicatori de performanță
- Cunoașterea diferitelor metode de implementare a operațiilor aritmetice
- Cunoașterea diferitelor tehnologii și tipuri de memorii: asociativă, *cache*, virtuală
- Cunoașterea unor arhitecturi paralele

Obiective Practice

- Proiectarea unor sisteme de tip SoC care combina atat partea hardware cat si partea software pentru a crea o aplicatii embedded robuste
- Demonstrarea taskurilor de accelerare hardware prin optimizarea unor operații aritmetice fundamentale
- Proiectarea și implementarea unor module hardware utilizând limbajul VHDL si C și mediul de dezvoltare Xilinx VivadoDesign Suite si Vitis IDE
- Utilizarea de functii SIMD pentru optimizarea unor algoritmi
- Utilizarea CPU+GPU (pe jetson) pentru optimizare d.p.d.v. al timpului al unor algoritmi intensivi computational

Zynq PS-PL Optimization (CPU + FPGA)



proiect4-Copy1 - Jupyter Notebook

New Tab

68.2.99:9090/notebooks/proiect4-Copy1.ipynb

jupyter proiect4-Copy1 Last Checkpoint: 10/23/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code Voilà

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image  # Pillow pentru manipularea imaginilor
from pynq import Overlay, allocate
from math import ceil

# Configurări
IMAGE_PATH = '/boot/poza/sample1.bmp'
BITSTREAM_PATH = '/home/xilinx/pynq/overlays/thresholdaah/design_1_wrapper.bit'

# Citirea i maginii
original_image = Image.open(IMAGE_PATH).convert('L') # Convertim la grayscale
image_matrix = np.array(original_image, dtype=np.uint8)

height, width = image_matrix.shape

# Afișează imaginea originală
plt.imshow(image_matrix, cmap='gray')
plt.title("Imaginea originală")
plt.axis("off")
plt.show()

print(f"Dimensiuni imagine: {width}x{height}")
```

Imaginea originală

```
Dimensiuni imagine: 1920x1080
```

In [2]:
```
# Încarcă bitstream-ul
overlay = Overlay(BITSTREAM_PATH)

# Alocare buffer pentru DMA
input_buffer = allocate(shape=(height, width), dtype=np.uint8)  # Alocăm buffer de dimensiune exactă
output_buffer = allocate(shape=(height, width), dtype=np.uint8)

# Copiază imaginea în buffer-ul DMA
```

proiect4-Copy1 - Jupyter Notebook

New Tab

090/notebooks/proiect4-Copy1.ipynb

jupyter proiect4-Copy1 Last Checkpoint: 10/23/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipyk

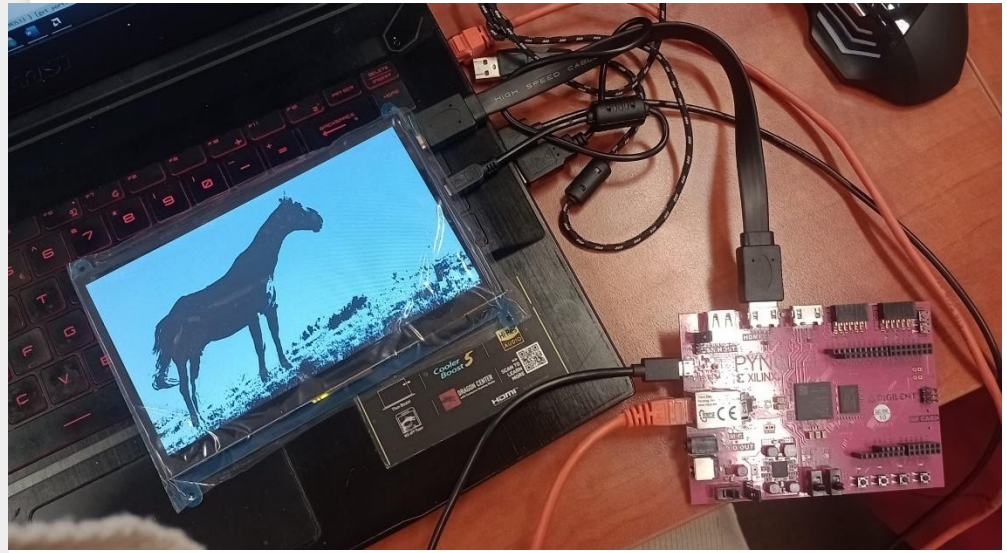
Run Code Voilà

Imaginea cu rânduri shiftate

```
Imaginea cu rânduri shiftate a fost salvată.
```

In [5]:
```
from pynq.overlays.base import BaseOverlay
from pynq.lib.video import VideoMode
import numpy as np
from PIL import Image

# Configurează overlay-ul de bază
base = BaseOverlay("base.bit")

# Configurează ieșirea HDMI la o rezoluție dorită
hdmi_out = base.video.hdmi_out
hdmi_out.configure(VideoMode(1280, 720, 24))  # Setează rezoluția la 720p, 24-bit color
hdmi_out.start()

# Presupunem că `processed_image` este imaginea finală (numpy array, grayscale)
# Extindem imaginea procesată pentru a se potrivi cu rezoluția HDMI (convertim la RGB)
processed_image_rgb = np.stack((processed_image,)*3, axis=-1)  # Convertim la RGB

# Redimensionăm imaginea procesată la rezoluția HDMI (1280x720)
image_pil = Image.fromarray(processed_image_rgb).resize((1280, 720))
frame = np.array(image_pil, dtype=np.uint8)

# Creăm un nou frame pentru HDMI Out
outframe = hdmi_out.newframe()

# Copiem imaginea în buffer-ul HDMI
outframe[:] = frame

# Afișăm frame-ul pe ieșirea HDMI
hdmi_out.writeframe(outframe)

print("Imaginea procesată a fost afișată pe HDMI.")
```

```
Imaginea procesată a fost afișată pe HDMI.
```



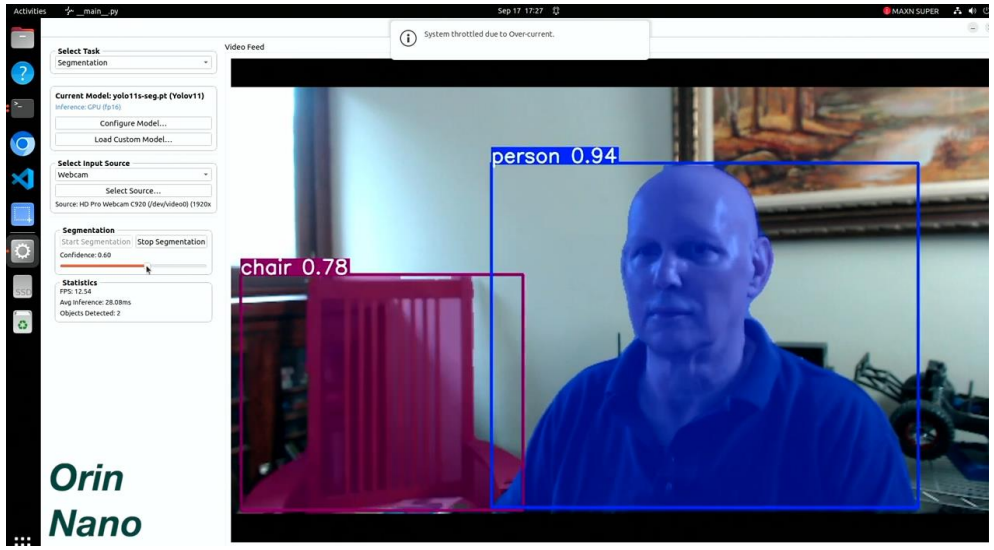
CPU (SIMD) Optimization

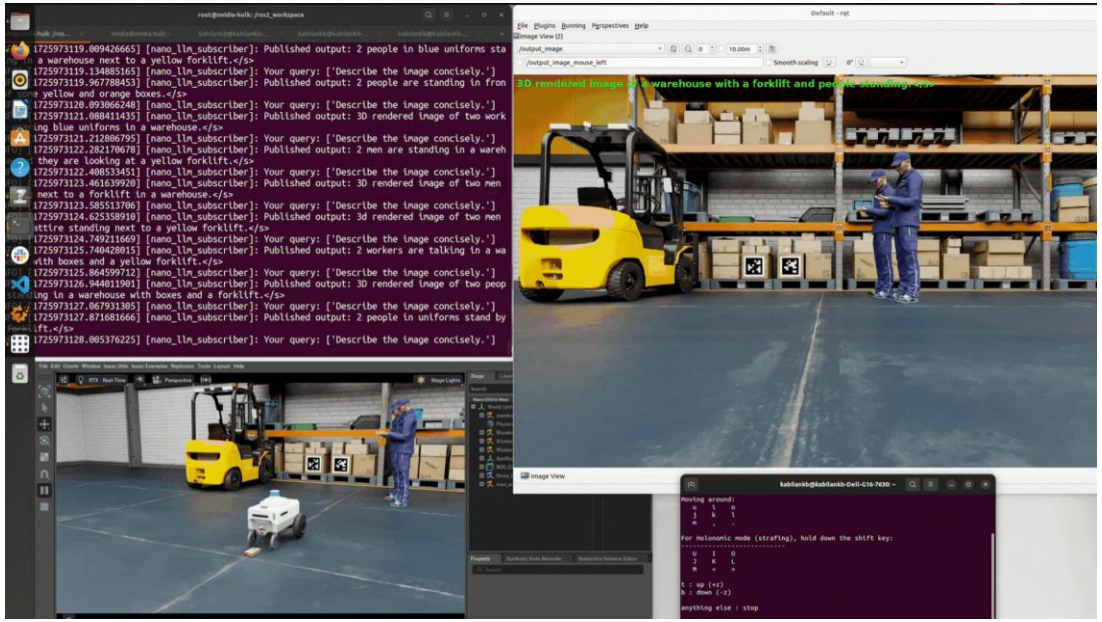
```
long convert_to_grayscale_otimized(FIL* imageFile) {
    while (remainingPixels > 0) {

        FRESULT imageResult = f_read(imageFile, pixelBuffer, currentChunkSize, &nrOfBytes);
        if (imageResult != FR_OK || nrOfBytes != currentChunkSize) {
            free(pixelBuffer);
            xil_printf("Failed to read image pixel data\n");
            return XST_FAILURE;
        }
        //  XTime_GetTime(&start);

        for (u32 i = 0; i < currentChunkSize; i += 16) {
            uint8x16_t pixelData = vld1q_u8(&pixelBuffer[i]);

            uint8x16x4_t bgra = vld4q_u8(&pixelBuffer[i]);

            float32x4_t blue = vcvtq_f32_u32(vmovl_u16(vget_low_u16(vmovl_u8(vget_low_u8(bgra.val[0])))));
            float32x4_t green = vcvtq_f32_u32(vmovl_u16(vget_low_u16(vmovl_u8(vget_low_u8(bgra.val[1])))));
            float32x4_t red = vcvtq_f32_u32(vmovl_u16(vget_low_u16(vmovl_u8(vget_low_u8(bgra.val[2])))));

            float32x4_t grayFloat = vmlaq_f32(vmlaq_f32(vmulq_f32(red, redFactor), green, greenFactor), blue, blueFactor);

            uint16x4_t gray16 = vqmovn_u32(vcvtq_u32_f32(grayFloat));
            uint8x8_t gray8 = vqmovn_u16(vcombine_u16(gray16, gray16));

            uint8x8x4_t grayPixel;
            grayPixel.val[0] = gray8;
            grayPixel.val[1] = gray8;
            grayPixel.val[2] = gray8;
            grayPixel.val[3] = vget_low_u8(bgra.val[3]);
```

(a) Imaginea originală RGB.

(b) Imaginea după aplicarea algoritmului de grayscaling.

Jetson Orin Nano (CPU + GPU)





Organizare

1. Plăcile de lucru, cât și componentele adiționale vor fi distribuite de către cadrul didactic. După primire, studentul va pune placa pe ON și va verifica aprinderea led-ului de POWER. Dacă acesta nu funcționează, se va raporta imediat cadrului didactic.
2. Toate resursele folosite în laborator vor fi predate cadrului didactic. Împrumutul acestora în afara laboratorului nu este permis. În cazul dispariției materialelor didactice, se vor aplica proceduri legale.
3. Pe stațiile de lucru existente, se va lucra în **D:**\, într-un director cu numele grupei, respectiv într-un sub-director cu numele studentului.
4. Munca se poate salva pe USB-stick sau online, nu se garantează persistența datelor de la un laborator la altul pe stațiile de lucru din laborator. O recomandare este ca studenții sa vina cu calculatoarele personale si sa lucreze pe acestea.
5. Din cauza resurselor limitate, schimbarea grupei de laborator nu este permisă. Situațiile excepționale se analizează de către profesorul de curs.

Progresul activitatii si prezenta

1. Este obligatorie citirea în prealabil a lucrării de laborator.
2. Activitățile de laborator se parcurg în ordinea indicată în suportul de laborator. La final, studentul va comunica cadrului didactic ultima activitate finalizată.
3. Pentru ca un laborator să fie considerat finalizat, studentul trebuie să finalizeze tutorialul corespunzător din lucrare. Dacă nu finalizează activitatea obligatorie în cadrul laboratorului curent, studentul are la dispoziție încă două săptămâni, începând din momentul predării laboratorului, pentru a demonstra cadrului didactic parcurgerea tutorialului. Pentru a primi punctul din oficiu la nota finala(conform formulei prezentate in secțiunea următoare), trebuie realizate activitățile obligatorii din laboratoare. Fiecare lucrare are o pondere egală în calculul punctului din oficiu.
4. Dacă studentul prezintă un cod pe care nu îl poate explica, acesta va fi considerat copiat (sau generat de un LLM) și va fi sancționat corespunzător.
5. Prezența este obligatorie. Se permit maximum 4 recuperări în cadrul oricărei semi-grupe din seria studentului, în orice săptămână a semestrului.
6. Recuperarea unei absențe este posibilă doar prin notificarea prealabilă a cadrului didactic și cu confirmarea acestuia.
7. Studentul poate obține puncte suplimentare la nota finală a laboratorului prin rezolvarea exercițiilor propuse. Pentru a primi punctajul la nota finala de laborator, studentul trebuie să prezinte cadrului didactic ceea ce a realizat. Pentru fiecare laborator se pot obține maximum 0,25 puncte suplimentare.
8. Studenții trebuie sa se prezinte strict cu grupa proprie (la laborator si proiect).
9. Se pot efectua schimburi intre semigrupe doar cu acordul cadrelor didactice.

Evaluare

Evaluare

Laborator

1. **Primul test (T1):** acoperă primele 5 lucrări de laborator.
2. **Al doilea test (T2):** acoperă restul lucrărilor de laborator.
3. **Punctaj din oficiu (PO):** acordat pe baza realizării exercițiilor obligatorii (1 punct).
4. **Puncte suplimentare (PP):** pentru exercițiile propuse (maximum 0,25 puncte/laborator).

$$\mathbf{N} = 0.35*\mathbf{T1} + 0.55*\mathbf{T2} + \mathbf{PO} \geq \mathbf{5}$$

$$\text{Nota laborator } (\mathbf{NL}) = \mathbf{N} + \mathbf{PP}$$

5. Pentru a promova laboratorul și a primi **PP**, **N** trebuie să fie minim 5.

Proiect

1. Evaluarea pe parcurs a progresului studentului în realizarea proiectului – **EP**
2. Evaluarea variantei finale a proiectului – **EF**

$$\text{Nota proiectului } (\mathbf{NP}) = 0.2 * EP + 0.8\ EF \geq \mathbf{5}$$

3. Pentru a promova proiectul, nota acestuia trebuie să fie minim 5.

Evaluare

Examen

1. Pentru a putea susține examenul atât nota de pe proiect cat si cea de pe laborator trebuie sa fie mai mari de 5. In caz contrar nu se va putea susține examenul pana când nu va fi promovata atât activitatea de proiect cat si cea de laborator.
2. Pentru a promova materia, nota la examenul scris trebuie să fie minim 5.
3. Nota finală la disciplină se calculează astfel:
   - Examen scris (E) – 60%
   - Nota laborator (NL) – 20%
   - Nota proiect (NP) – 20%

Pentru neclaritati sau alte intrebari puteti sa scrieti un email pe Mircea.Muresan@cs.utcluj.ro


Next ...