ДОДАТОК А

А.1 Реалізація алгоритмів обробки

А.1.2 Файл comp_node.sv

```
`timescale 1ns / 1ps

module comp_node #(
    parameter low_enabled = 1,
    parameter high_enabled = 1
)(
    input [7:0] data_a,
    input [7:0] data_b,

    output reg [7:0] data_hi,
    output reg [7:0] data_lo
);

    reg sel0;

    always @(*) begin
        if(data_a < data_b)
            sel0 = 1'b0;
        else
            sel0 = 1'b1;
    end

    always @(*) begin
        case (sel0)
            1'b0 :
            begin
                if(low_enabled)
                    data_lo = data_a;
                if(high_enabled)
                    data_hi = data_b;
            end
            1'b1 :
            begin
                if(low_enabled)
                    data_lo = data_b;
                if(high_enabled)
                    data_hi = data_a;
            end
            default :
            begin
```

```
                data_lo = 7'b0;
                data_hi = 7'b0;
            end
        endcase
    end

endmodule
```

A.1.2 Файл pixel_net.sv

```
`timescale 1ns / 1ps

module pixel_net (
    input [7:0] c3h,
    input [7:0] c3m,
    input [7:0] c3l,
    input [7:0] c2h,
    input [7:0] c2m,
    input [7:0] c2l,
    input [7:0] c1h,
    input [7:0] c1m,
    input [7:0] c1l,

    output [7:0] median
);

    wire [7:0] comp_node_u0_lo;
    wire [7:0] comp_node_u1_hi;
    wire [7:0] comp_node_u1_lo;
    wire [7:0] comp_node_u2_hi;
    wire [7:0] comp_node_u3_lo;
    wire [7:0] comp_node_u4_hi;
    wire [7:0] comp_node_u5_hi;
    wire [7:0] comp_node_u6_lo;
    wire [7:0] comp_node_u7_hi;
    wire [7:0] comp_node_u7_lo;
    wire [7:0] comp_node_u8_hi;

    comp_node #(
        .low_enabled(1),
        .high_enabled(0)
    ) comp_node_u0 (
        .data_a(c3h),
        .data_b(c2h),
        .data_hi(), // unconnected
        .data_lo(comp_node_u0_lo)
    );

    comp_node #(
        .low_enabled(1),
```

```
    .high_enabled(1)
) comp_node_u1 (
    .data_a(c3m),
    .data_b(c2m),
    .data_hi(comp_node_u1_hi),
    .data_lo(comp_node_u1_lo)
);

comp_node #(
    .low_enabled(0),
    .high_enabled(1)
) comp_node_u2 (
    .data_a(c2l),
    .data_b(c1l),
    .data_hi(comp_node_u2_hi),
    .data_lo() // unconnected
);

comp_node #(
    .low_enabled(1),
    .high_enabled(0)
) comp_node_u3 (
    .data_a(comp_node_u0_lo),
    .data_b(c1h),
    .data_hi(), // unconnected
    .data_lo(comp_node_u3_lo)
);

comp_node #(
    .low_enabled(0),
    .high_enabled(1)
) comp_node_u4 (
    .data_a(comp_node_u1_lo),
    .data_b(c1m),
    .data_hi(comp_node_u4_hi),
    .data_lo() // unconnected
);

comp_node #(
    .low_enabled(0),
    .high_enabled(1)
) comp_node_u5 (
    .data_a(c3l),
    .data_b(comp_node_u2_hi),
    .data_hi(comp_node_u5_hi),
    .data_lo() // unconnected
);

comp_node #(
    .low_enabled(1),
    .high_enabled(0)
) comp_node_u6 (
```

```
        .data_a(comp_node_u1_hi),
        .data_b(comp_node_u4_hi),
        .data_hi(), // unconnected
        .data_lo(comp_node_u6_lo)
    );

    comp_node #(
        .low_enabled(1),
        .high_enabled(1)
    ) comp_node_u7 (
        .data_a(comp_node_u3_lo),
        .data_b(comp_node_u6_lo),
        .data_hi(comp_node_u7_hi),
        .data_lo(comp_node_u7_lo)
    );

    comp_node #(
        .low_enabled(0),
        .high_enabled(1)
    ) comp_node_u8 (
        .data_a(comp_node_u7_lo),
        .data_b(comp_node_u5_hi),
        .data_hi(comp_node_u8_hi),
        .data_lo() // unconnected
    );

    comp_node #(
        .low_enabled(1),
        .high_enabled(0)
    ) comp_node_u9 (
        .data_a(comp_node_u7_hi),
        .data_b(comp_node_u8_hi),
        .data_hi(), // unconnected
        .data_lo(median)
    );

endmodule
```

A.1.3 Файл median_wrap.sv

```
`timescale 1ns / 1ps

module median_unit(
    input         clk_i,
    input         rst_i,
    input         en_i,
    input  [31:0]data_i00,
    input  [31:0]data_i01,
    input  [31:0]data_i02,
    input  [31:0]data_i10,
```

```
        input  [31:0]data_i11,
        input  [31:0]data_i12,
        input  [31:0]data_i20,
        input  [31:0]data_i21,
        input  [31:0]data_i22,
        output [31:0]data_o
    );

        wire [7:0]median_red;
        wire [7:0]median_green;
        wire [7:0]median_blue;

        reg [31:0]data_o_r;
        assign data_o = data_o_r;

        always @(posedge clk_i or posedge rst_i) begin
            if (rst_i) begin
                data_o_r <= 32'd0;
            end
            else if (en_i) begin
                        data_o_r  <=  {median_red,  median_green,
median_blue, 8'd0};
            end
        end

        pixel_net pixel_net_u0 (
            .c3h(data_i00[31:24]),
            .c3m(data_i01[31:24]),
            .c3l(data_i02[31:24]),
            .c2h(data_i10[31:24]),
            .c2m(data_i11[31:24]),
            .c2l(data_i12[31:24]),
            .c1h(data_i20[31:24]),
            .c1m(data_i21[31:24]),
            .c1l(data_i22[31:24]),
            .median(median_red)
        );

        pixel_net pixel_net_u1 (
            .c3h(data_i00[23:16]),
            .c3m(data_i01[23:16]),
            .c3l(data_i02[23:16]),
            .c2h(data_i10[23:16]),
            .c2m(data_i11[23:16]),
            .c2l(data_i12[23:16]),
            .c1h(data_i20[23:16]),
            .c1m(data_i21[23:16]),
            .c1l(data_i22[23:16]),
            .median(median_green)
        );

        pixel_net pixel_net_u2 (
```

```
        .c3h(data_i00[15:8]),
        .c3m(data_i01[15:8]),
        .c3l(data_i02[15:8]),
        .c2h(data_i10[15:8]),
        .c2m(data_i11[15:8]),
        .c2l(data_i12[15:8]),
        .c1h(data_i20[15:8]),
        .c1m(data_i21[15:8]),
        .c1l(data_i22[15:8]),
        .median(median_blue)
    );

endmodule
```

A.1.4 Файл grayscale.sv

```
`timescale 1ns / 1ps

module grayscale_unit(
    input        clk_i,
    input        rst_i,
    input        en_i,
    input  [31:0]data_i,
    output [31:0]data_o
);

    wire [7:0]red_t;
    wire [7:0]green_t;
    wire [7:0]blue_t;

    reg [31:0]data_o_r;
    reg [7:0]avg_r;

    assign red_t = data_i[31:24];
    assign green_t = data_i[23:16];
    assign blue_t = data_i[15:8];

    always @(posedge clk_i or posedge rst_i) begin
        if (rst_i) begin
            data_o_r <= 32'd0;
        end
        else if (en_i) begin
            avg_r <= (red_t + green_t + blue_t) / 'd3;
            data_o_r <= {avg_r, avg_r, avg_r, 8'd0};
        end
    end

    assign data_o = data_o_r;

endmodule
```

A.1.5 Файл gray2bin.sv

```
`timescale 1ns / 1ps

module gray2bin_unit(
    input        clk_i,
    input        rst_i,
    input        en_i,
    input  [31:0]data_i,
    output [31:0]data_o
);

    localparam [7:0] Threshold_High = 8'd160;
    localparam [7:0] Threshold_Low  = 8'd60;

    wire [7:0] gray_input;
    assign gray_input = data_i[31:24];

    reg [31:0]data_o_r;

    always @(posedge clk_i or posedge rst_i) begin
        if (rst_i) begin
            data_o_r <= 32'd0;
        end
        else if (en_i) begin
            if (gray_input > Threshold_Low
             && gray_input < Threshold_High)
                data_o_r <= {8'd255, 8'd255, 8'd255, 8'd0};
            else
                data_o_r <= {8'd0, 8'd0, 8'd0, 8'd0};
        end
    end

    assign data_o = data_o_r;

endmodule
```

A.2 Код роботи з трансмітером

A.2.1 Файл hdmi_data.sv

```
`timescale 1ns / 1ps

module hdmi_data(
     i_clk,
     i_rst_n,
     data_in,
```

```
     addr_i,
     data_out
    );

input    wire    i_clk;
input    wire    i_rst_n;
input    wire    [31:0] data_in;
input    wire    [18:0] addr_i;
output   wire    [15:0] data_out;

reg [15:0] data_out_r;

always@(posedge i_clk or negedge i_rst_n)begin
      if(i_rst_n==1'b0)
        data_out_r <=  16'd0;
      else
        data_out_r <= ((data_in[31:24] & 'b11111000) << 8)
                    | ((data_in[23:16] & 'b11111100) << 3)
                    | (data_in[15:8] >> 3);
   end

  assign   data_out =   data_out_r;

endmodule
```

A.2.2 Файл hdmi_control.sv

```
`timescale 1ns / 1ps

module hdmi_control (
                  i_clk,
                  i_rst_n,
                  VDEN,
                  HSYNC,
                  VSYNC,
                  addr_out
                );

input i_clk;
input i_rst_n;
output VDEN;
output HSYNC;
output VSYNC;
output [18:0] addr_out;


parameter H_SyncPulse=44;
parameter H_BackPorch=148;
parameter H_ActivePix=1920;
parameter H_FrontPorch=88;
```

```
    parameter Hde_start=H_SyncPulse+H_BackPorch;
    parameter Hde_end=H_SyncPulse+H_BackPorch+H_ActivePix;
    parameter                                          LinePeriod
=H_SyncPulse+H_BackPorch+H_ActivePix+H_FrontPorch;

    parameter V_SyncPulse=5;
    parameter V_BackPorch=36;
    parameter V_ActivePix=1080;
    parameter V_FrontPorch=4;
    parameter Vde_start=V_SyncPulse+V_BackPorch;
    parameter Vde_end=V_SyncPulse+V_BackPorch+V_ActivePix;
    parameter                                         FramePeriod
=V_SyncPulse+V_BackPorch+V_ActivePix+V_FrontPorch;

    reg [11:0]x_cnt;
    reg [11:0]y_cnt;
    reg hsync;
    reg vsync;
    reg hsync_r;
    reg vsync_r;
    wire [11:0] x_pos_r;
    wire [11:0] y_pos_r;
    wire [18:0] addr_data;

    reg VDEN_r;
    reg HSYNC_r;
    reg VSYNC_r;
    reg [11:0] addr_out_r;

    always@(posedge i_clk or negedge i_rst_n)begin
          if(i_rst_n==1'b0)
                  x_cnt <= 12'd0;
          else if(x_cnt==(LinePeriod-1'b1))
                  x_cnt <= 12'd0;
          else
                  x_cnt <= x_cnt + 11'd1;
      end

    always@(posedge i_clk or negedge i_rst_n)begin
          if(i_rst_n==1'b0)
                  hsync <= 1'b1;
          else if(x_cnt<(H_SyncPulse-1'b1))
                  hsync <= 1'b0;
          else
                  hsync <= 1'b1;
      end

    always@(posedge i_clk or negedge i_rst_n)begin
          if(i_rst_n==1'b0)
                  hsync_r <=1'b0;
          else if(x_cnt>=(Hde_start-1'b1)&&x_cnt<(Hde_end-1'b1))
                  hsync_r <=1'b1;
```

```
        else
                hsync_r <=1'b0;
    end

    always@(posedge i_clk or negedge i_rst_n)begin
        if(i_rst_n==1'b0)
                y_cnt <= 12'd0;
        else if(y_cnt==(FramePeriod-1'b1))
                y_cnt <= 12'd0;
        else if(x_cnt==(LinePeriod-1'b1))
                y_cnt <= y_cnt + 1'b1;
    end

    always@(posedge i_clk or negedge i_rst_n)begin
        if(i_rst_n==1'b0)
                vsync <= 1'b1;
        else if(y_cnt<(V_SyncPulse-1'b1))
                vsync <= 1'b0;
        else
                vsync <= 1'b1;
    end

    always@(posedge i_clk or negedge i_rst_n)begin
        if(i_rst_n==1'b0)
                vsync_r <= 1'b0;
        else if(y_cnt>=(Vde_start-1'b1)&&y_cnt<(Vde_end-1'b1))
                vsync_r <= 1'b1;
        else
                vsync_r <= 1'b0;
    end

    assign x_pos_r = (hsync_r&&vsync_r)?
        (x_cnt-H_SyncPulse-H_BackPorch+12'd1):12'd0;
    assign y_pos_r = (hsync_r&&vsync_r)?
        (y_cnt-V_SyncPulse-V_BackPorch+12'd1):12'd0;

    assign addr_data = x_pos_r+y_pos_r*H_ActivePix;

    always@(posedge i_clk or negedge i_rst_n)begin
        if (i_rst_n == 1'b0) begin
                VDEN_r      <= 1'b0;
                HSYNC_r     <= 1'b0;
                VSYNC_r     <= 1'b0;
                addr_out_r  <= 1'b0;
                end
         else begin
                VDEN_r      <= {hsync_r && vsync_r};
                HSYNC_r     <= hsync;
                VSYNC_r     <= vsync;
                addr_out_r  <= addr_data;
         end
    end
```

```
assign HSYNC    =   hsync;
assign VSYNC    =   vsync;
assign VDEN     =   {hsync_r&&vsync_r};
assign addr_out =   addr_out_r;

endmodule
```

А.3 Програмний код роботи с Block RAM

```
#include "opencv2/opencv.hpp"
#include "opencv2/highgui/highgui.hpp"

#include <iostream>
#include <stdexcept>
#include <cstdint>
#include <cstring>
#include <string>
#include <memory>

#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>

using namespace cv;

namespace {
    constexpr const char* device_name = "/dev/mem";
    constexpr std::size_t bram_size = 0x4EC000;
    constexpr u_int64_t bram_base = 0x40000000;
}


void HelpStringOutput() {
     std::cerr << "Usage: " << argv[0] << " <video_file>" <<
std::endl;
}


void TerminateWithError(const std::string& error_str) {
    std::cerr << error_str.c_str() << std::endl;
    std::exit(EXIT_FAILURE); // Stop whole program running
}


int main(int argc, char* argv[]) {
    if (argc != 2) {
        HelpStringOutput();
        TerminateWithError("Wrong arguments provided");
    }
```

```
        // Open input video with OpenCV
                std::unique_ptr<cv::VideoCapture>   srcCapture(new
cv::VideoCapture);
        srcCapture->open(argv[1]);
        if (!srcCapture || !srcCapture->isOpened()) {
            TerminateWithError("Couldn't open the vide file");
        }

        // Open the emory device in synchronous  mode
        int fd = open(device_name, O_RDWR | O_SYNC);
        if (fd == -1) {
            TerminateWithError("Internal error occured");
        }

        u_int8_t* bram8_ptr = static_cast<u_int8_t*>(mmap(nullptr,
                        bram_size, // BRAM buffer size
                        PROT_READ | PROT_WRITE, // access flags
                        MAP_SHARED, // shared for other processes
                              fd, bram_base // descriptor and base
address
        ));

        while (true) {
            Mat inputImg;
            if (!srcCapture->read(inputImg)) {
                // Assume the file ended and restart playback
                srcCapture->set(cv::CAP_PROP_POS_FRAMES, 0);
                srcCapture->read(inputImg);
            }

            auto img_ptr = inputImg.ptr();
             auto img_size = inputImg.channels() * inputImg.rows *
inputImg.cols;
            memcpy(bram8_ptr, img_ptr, img_size);
            waitKey(30);
        }

        return 0;
    }
```

ДОДАТОК Б

https://doi.org/10.1007/s10479-021-04417-1

Development of coroutines usage model for cooperative multitasking implementation on the systems with limited resources

Amin Salih Mohammed[1,2] · Inna Filippenko[3] · B. Saravana Balaji[4] · Olesia Barkovska[3] · Ivan Semenenko[3] · Valentyn Korniienko[3]

Accepted: 8 November 2021

Abstract

According to the growing complexity of tasks, which embedded systems should solve, there arises a need to use Real-Time Operation Systems (RTOS). However, RTOS usage and implementation require a set of restrictions for a system, especially on the memory usage and time delays occurrence. The available implementations of coroutines neither cross-platform solutions nor accessible without a third-party library. The article proposes a model of cross- platform coroutine usage from C 20 for embedded systems programming and lightweight cooperative multitasking implementation. The proposed model contains the improved minimal scheduler without priorities mechanism and tasks descriptors. The model of coroutines usage for resource-constrained systems is developed. The task inside the model is presented as a coroutine, which can communicate with available coroutines, call them, or be suspended. Also, the minimal primitives of coroutines usage for cooperative multitasking, architecture decisions, and non-blocking data transmission were implemented. For the experimental research, the following list of program components was implemented in the proposed model: cooperative coroutines scheduler, tasks for data transmission, and awaitable type prototypes. The testing and analysis were completed on the chosen microcontroller (MCU). The perspective of coroutines usage on resource-constrained systems was shown in the research based on obtained results of FLASH and RAM consumption.

Keywords Coroutines · Embedded systems · Memory consumption reducing · Embedded interfaces · Cooperative multitasking

✉ B. Saravana Balaji saravanabalaji.b@gmail.com

[1] Department of Computer Engineering, Lebanese French University, Erbil, Iraq

[2] Department of Software and Informatics, Salahaddin University, Erbil, Iraq

[3] Department of Automation and Computer Engineering Design, Kharkiv National University of Radio Electronics, Kharkiv Oblast, Ukraine

[4] Department of Information Technology, Lebanese French University, Erbil, Iraq

Published online: 25 November 2021

Adaptation of FPGA Architecture for Accelerated Image Preprocessing

Olesia Barkovska [1][0000-0001-7496-4353], Inna Filippenko [2][0000-0002-3584-2107],
Ivan Semenenko [3][0000-0002-6498-2440], Valentyn Korniienko[4][0000-0001-7070-5127]

[1,2,3,4,] Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine
olesia.barkovska@nure.ua

Abstract

The work is devoted to the topical problem at the intersection of the communications theory, digital electronics and numerical analysis, namely the study of image processing methods implementation time on different architectures of computational devices, which provide for software and hardware acceleration. The result of this work is the development of a resilient SoC Zynq7000–based computing system with programmable logic and the possibility to load images to FPGA RAM using the resources of ARM core for further processing and output via HDMI video interface, which enables to change PL configuration at any time during the processing process. The efficiency of the FPGA approach was compared with a parallel image processing methods implementation with OpenMP and CUDA. The analysis of algorithm speed testing findings based on various outputs proved the advantage (of over 60 times) of hardware acceleration of image processing to software analogues. The obtained results may be used in the development of embedded SoC-based solutions, which require acceleration of big data processing.

Keywords: Speedup · FPGA · Performance · Acceleration · Parallel System · Image Processing

ДОДАТОК В

Харківський національний університет радіоелектроніки
Кафедра автоматизації проектування обчислювальної техніки

Кваліфікаційна робота на тему:

Методи обробки зображень у вбудованих системах

Виконав:
ст. гр. СКСм-20-1
Семененко Іван Георгійович

Керівник:
доц. каф. АПОТ
Філіппенко І.В.

Харків 2021

МЕТА РОБОТИ

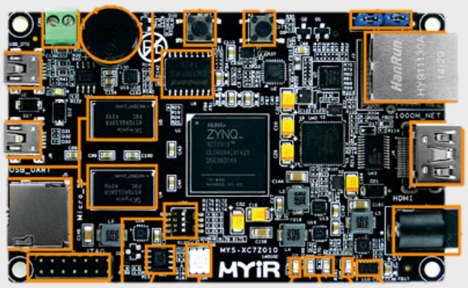
- ознайомлення із особливостями структури вбудованих систем
- огляд існуючих методів цифрової обробки
- розробка моделі на основі ARM
- розробка моделі з використанням FPGA
- порівняльний аналіз швидкодії алгоритмів

АКТУАЛЬНІСТЬ ПРОБЛЕМИ

- зростання об'єму даних, що потребує обробки
- стрімкий розвиток популярності вбудованих систем
- актуальність технологій обробки зображень
- використання в складних комп'ютерних системах
- потреба в швидкісній обробці (real-time)

ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА

Z-turn Board на базі Xilinx Zynq7020



1. ARM Cortex A9 (PS)
   - частота роботи 866 МГц
   - 1Гб DDR SDRAM
2. Xilinx Kintex-7 (PL)
   - 53200 LUTs
   - 106400 flip-flops
   - 140x36Kb RAM
   - 220 DSP блоків

ВИКОРИСТАННЯ OPENCV

OpenCV— бібліотека функцій та алгоритмів комп'ютерного зору обробки зображень і чисельних алгоритмів загального призначення звідкритим кодом

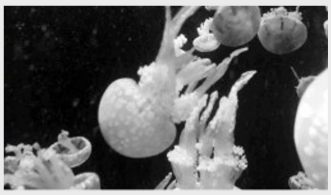
- Microsoft Windows
- Windows RT
- Linux
- Mac OS X
- iOS.

РЕАЛІЗАЦІЯ МЕДІАННОГО ФІЛЬТРУ

```
int ImgProc::MedianDenoise(const cv::Mat& src, cv::Mat& dst) {
    return CpuTimer::MeasureTime([&src, &dst]() {
        cv::medianBlur(src, dst, 3);});
}
```

ВІДТІНКИ СІРОГО ТА БІНАРИЗАЦІЯ

```
int ImgProc::Grayscale(const cv::Mat& src, cv::Mat& dst) {
    return CpuTimer::MeasureTime([&src, &dst]() {
        cv::cvtColor(src, dst, cv::COLOR_BGR2GRAY);
    });
}
```



```
int ImgProc::Binarization(const cv::Mat& src, cv::Mat& dst) {
    return CpuTimer::MeasureTime([&src, &dst]() {
        cv::threshold(src, dst, 50, 255, cv::THRESH_BINARY);
    });
}
```

ЗГОРТКОВА НЕЙРОННА МЕРЕЖА

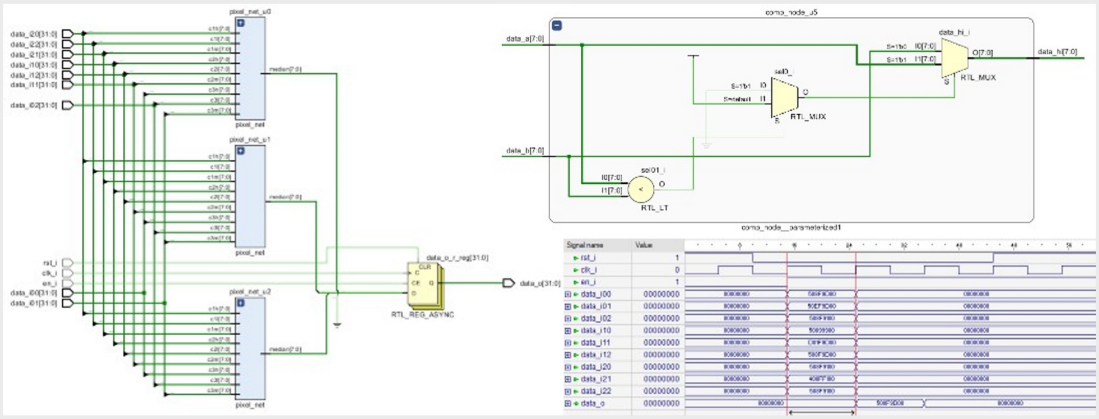
```
void maxpool(const cv::Mat& src, cv::Mat& dst) {
    int newHeight = (src.size().height - 2) / 2 + 1;
    int newWidth = (src.size().width - 2) / 2 + 1;

    cv::Mat downscaled = cv::Mat::zeros(newHeight,
                                        newWidth, cv::CV_32FC);

    for (int i = 0; i < newHeight; i+=2) {
        for (int j = 0; j < newWidth; j += 2) {
            double f_val1 = std::max(src[i][j], src[i][j + 1]);
            double f_val2 = std::max(src[i + 1][j], src[i + 1][j + 1]);
            downscaled[i/2][ /2] = std::max(f_val1, f_val2);
        }
    }

    downscaled.copyTo(dst);
}
```

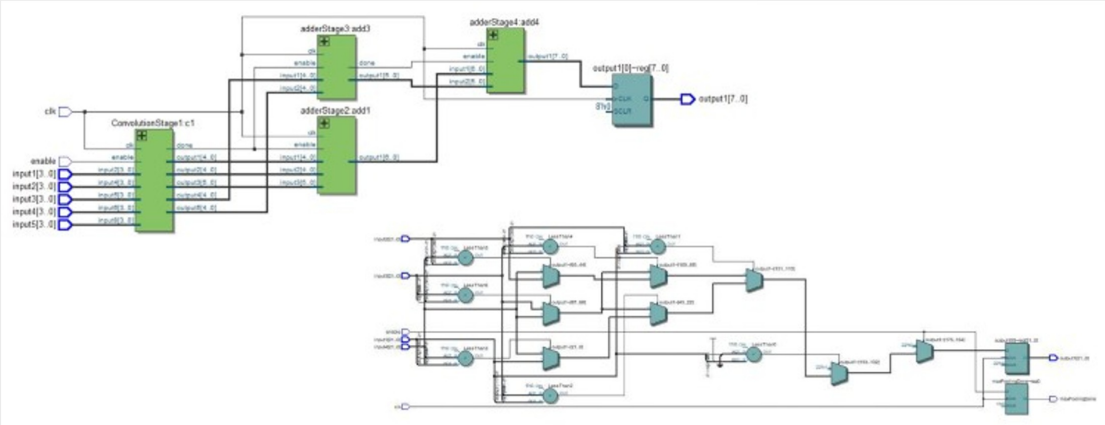

РЕАЛІЗАЦІЯ МЕДІАННОГО ФІЛЬТРУ



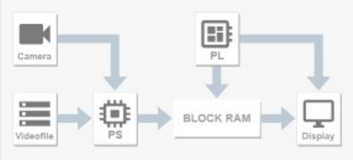
ВІДТІНКИ СІРОГО ТА БІНАРИЗАЦІЯ

```
always @(posedge clk_i or posedge rst_i) begin
    if (rst_i) begin
      data_o_r <= 32'd0;
    end
    else if (en_i) begin
      if (gray_input > Threshold_Low  &&
        gray_input < Threshold_High )
        data_o_r <= {8'd255, 8'd255, 8'd255, 8'd0};
      else
        data_o_r <= {8'd0, 8'd0, 8'd0, 8'd0};
    end
  end
```

ЗГОРТКОВА НЕЙРОННА МЕРЕЖА

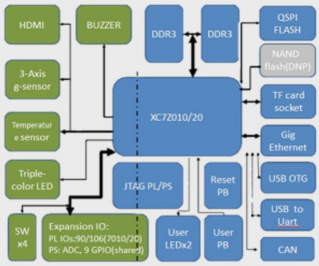


ТЕСТУВАННЯ АПАРАТНОЇ ПЛАТФОРМИ



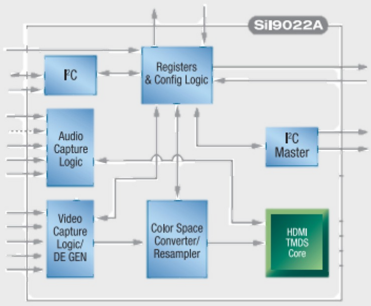
- AXI SmartConnect
- Processor System Reset
- Clocking Wizard

- ZYNQ7 Processing System
- Block Memory Generator
- AXI BRAM Controller



ВИВЕДЕННЯ ДО ВІДЕОІНТЕРФЕСУ HDMI

- формат даних RGB565
- роздільна здатність 1920x1080
- частота синхронізації 148.5 МГц
- можливість роботи з VDMA
- апаратна підтримка переривань



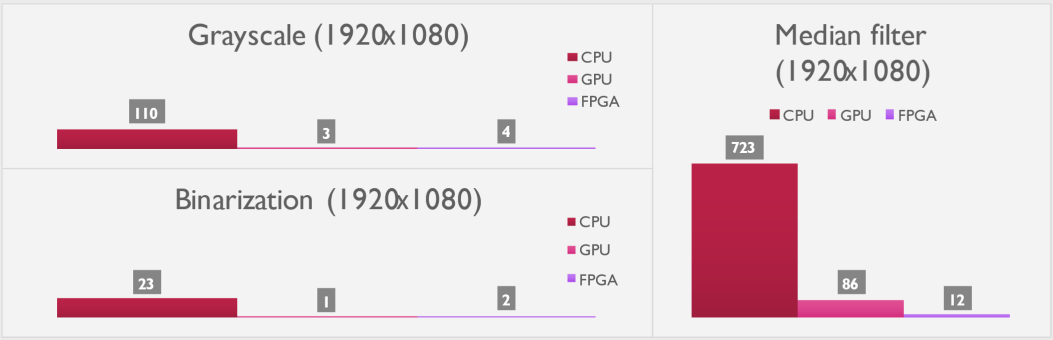
НАЛАГОДЖЕННЯ ПЗ ДЛЯ РОБОТИ FPGA

- робота з камерою та відеофайлами
- синхронний доступ до Block RAM
- базується на бібліотеці OpenCV

```
constexpr const char* device_name = "/dev/mem";
constexpr std::size_t bram_size = 0x4EC000;
constexpr u_int64_t bram_base = 0x40000000;

int fd = open(device_name, O_RDWR | O_SYNC);
if (fd != -1) {
    u_int8_t* bram8_ptr = static_cast<u_int8_t*>(
                mmap(nullptr, bram_size,
                        PROT_READ | PROT_WRITE,
                        MAP_SHARED
                        fd, bram_base address
                )
    );
}
```

АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ



ВИСНОВКИ

- досліджено різні методи цифрової обробки зображень у вбудованих системах
- розроблено алгоритми для швидкої обробки зображень в системах ARM та FPGA
- налагоджено інтерфейси взаємодії мікроконтролера з програмованою логікою
- Спроектована та протестована архітектура згорткової нейронної мережі
- виконано порівняльний аналіз отриманих результатів тестування швидкодії алгоритмів