

Remaining Points to Earn

Remaining Points to Earn

Lab 4 : 12 points + bonus points (partially graded)

Lab 5 : 8 points + bonus points

Lab 6 : 4 points + bonus points

Lab 5 Exercises: 1.6 points

Final Exam: 25 points

Homework 1: 1.5 points

Homework 2: 1.5 points

ACTIVITY points: up to 5 bonus points

Total: 53.6 points



Lecture 19

Lab 5 FPro System

Source code of the FPro system to be used as a starting point for Lab 5

Posted at

http://ece.gmu.edu/coursewebpages/ECE/ECE448/S20/labs/448_lab5.htm

under

Examples:

Source code of the FPro system targeting Basys 3
with the frame buffer (one bit per pixel) and support for buttons.

Steps to Implement the Basic FPro System for Lab 5

1. Create a new project for Lab 5
2. Create cpu.xci using the procedure described in Appendix A.4.3. Generate a MicroBlaze MCS Component of the main textbook.
3. Add all files from the subfolders
   - sys
   - mmio, and
   - video

   of the download folder video (including any subfolders!).
4. Add

   video/constraint/basys3_chu.xdc

   as a constraint file for your project.

Basic FPro System for Lab 5
Major Features

1. Support for the Frame Buffer Core with 1 bit per pixel

2. Support for Switches, Buttons, and LEDs

Basic FPro System for Lab 5
Necessary Extensions

1. **Julia Set Fractal MMIO Core**
   for performing calculations
   (e.g., by replacing a dummy MMIO core in slot 4, S4_USER).

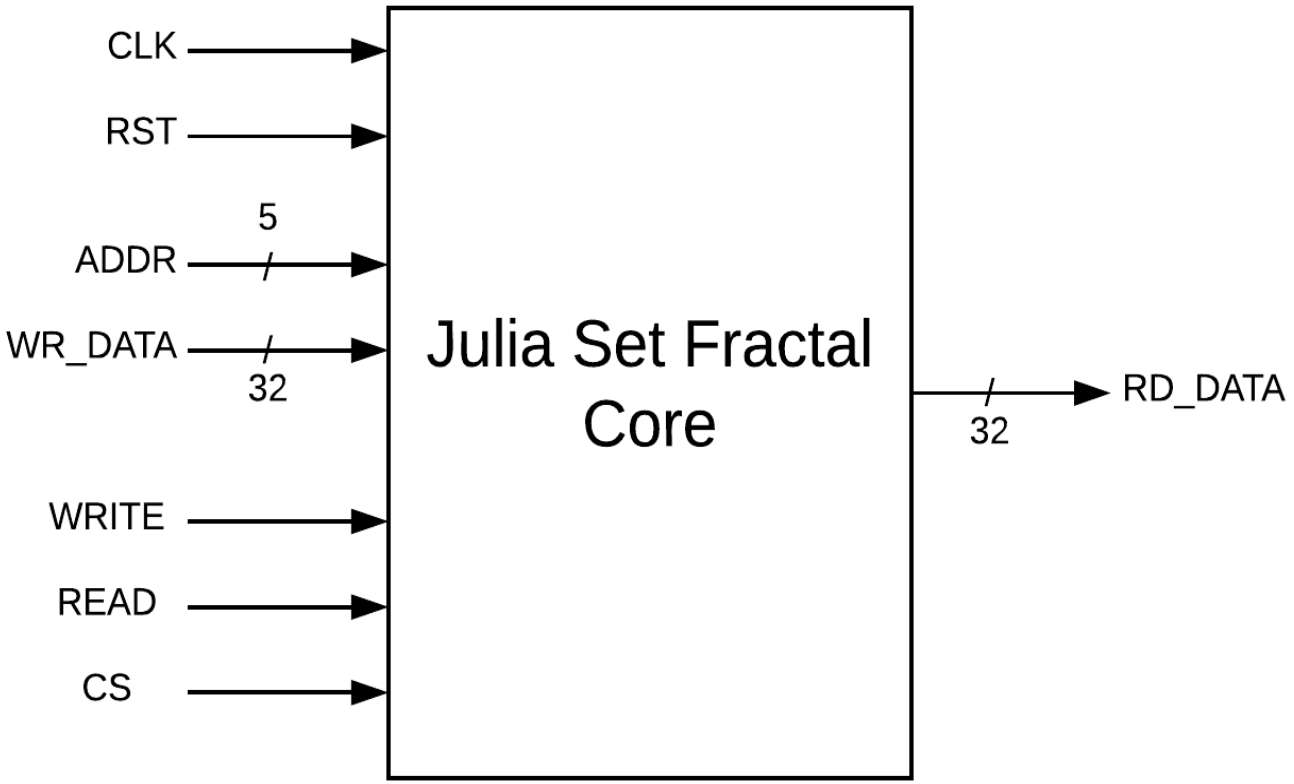
2. **Video Core**
   allowing you to change the colors of
   a fractal and background
   (e.g., by replacing a dummy video core in slot 4, V4_USER4).



Lab 5 MMIO Core

Julia Set Fractal Core

MMIO Core!!!

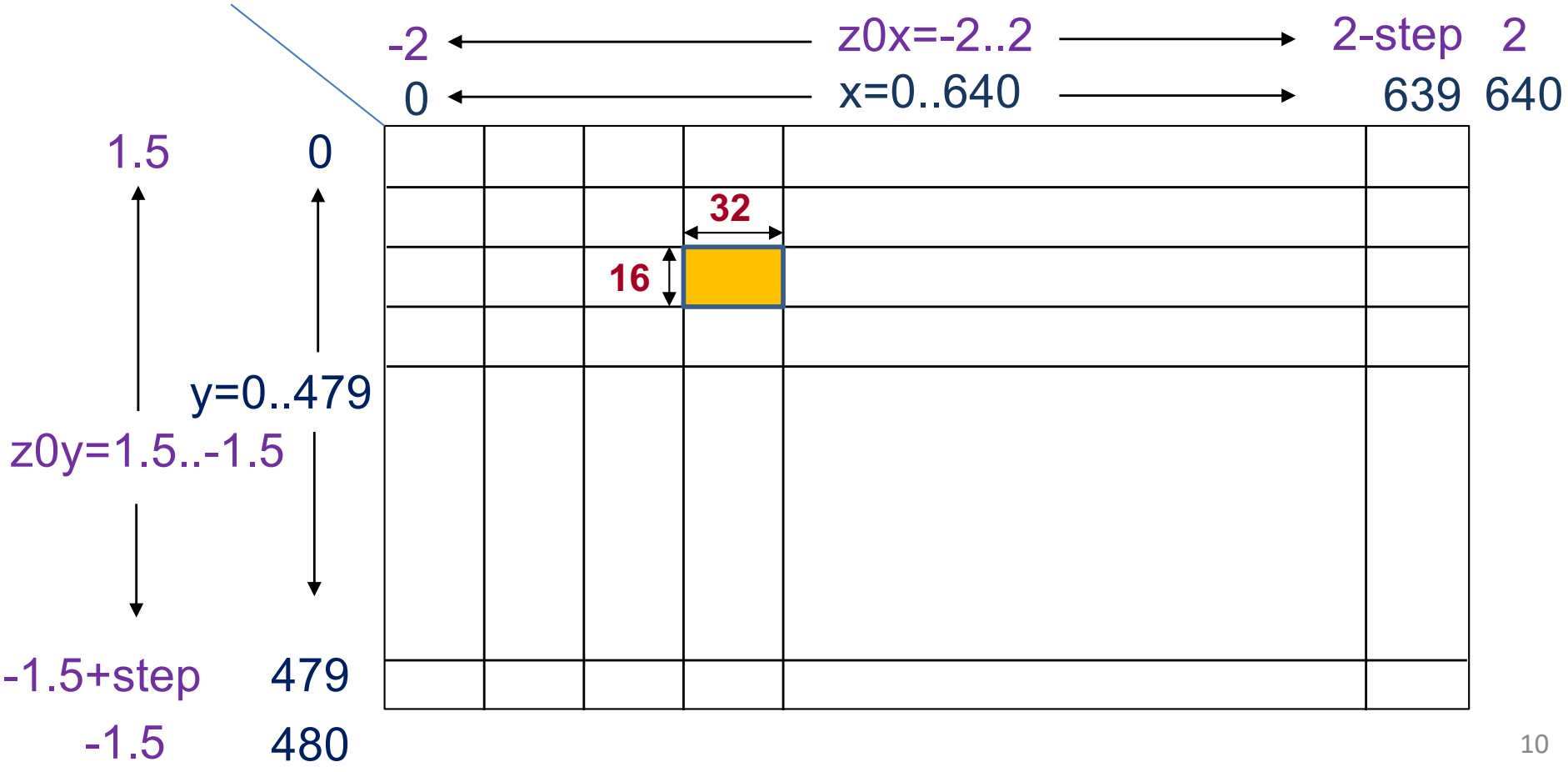


Inputs and Outputs

$-2 \le z0x = \mathrm{Re}(z_0) < 2$

$-1.5 < z0y = \mathrm{Im}(z_0) \le 1.5$

$(x,y) \Longleftrightarrow (z0x, z0y)$



Software/Hardware Partitioning: Lab 5

Software

Application [main_fractal.cpp]:

Custom Driver [fractal_core.h, fractal_core.cpp]:

- FractalCore class

Hardware

Custom Core [fractal_core.vhd]:

- Julia Set Fractal in Hardware

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

*** datapath * controller * top-level * functional verification**

H2. A wrapper matching the interface of an MMIO core

*** design adjustments * coding * functional verification**

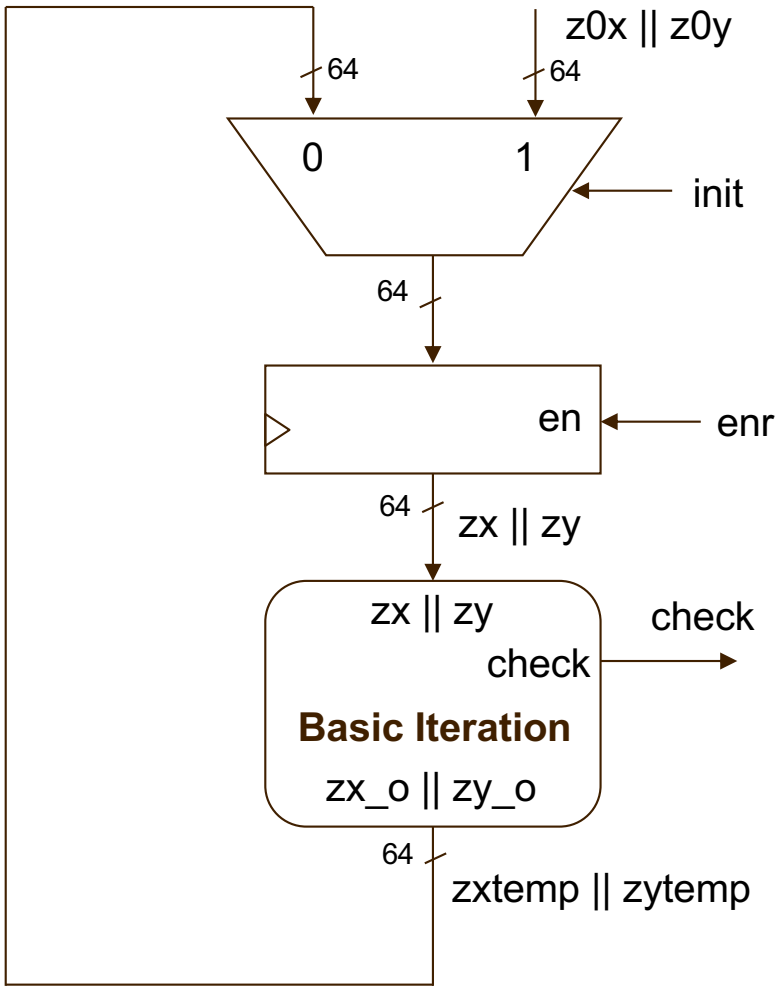
Software Design:

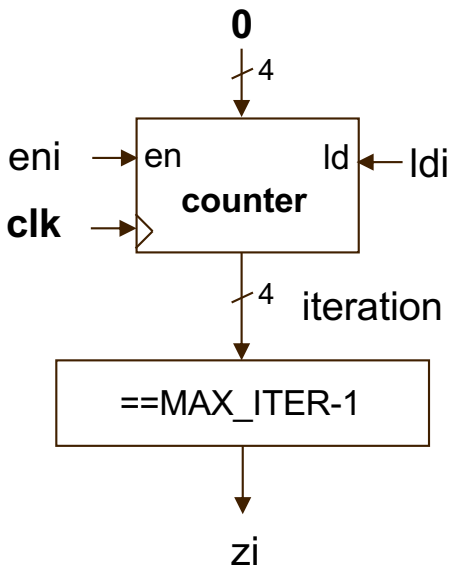
S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

Block Diagram





User Interface

- The higher the maximum iteration count, the more accurate the picture but the longer the computation time.
- Use switches SW15..SW13 to increase the value of MAX_ITER by a multiple of 50 beyond the default value of 100, based on the table below:

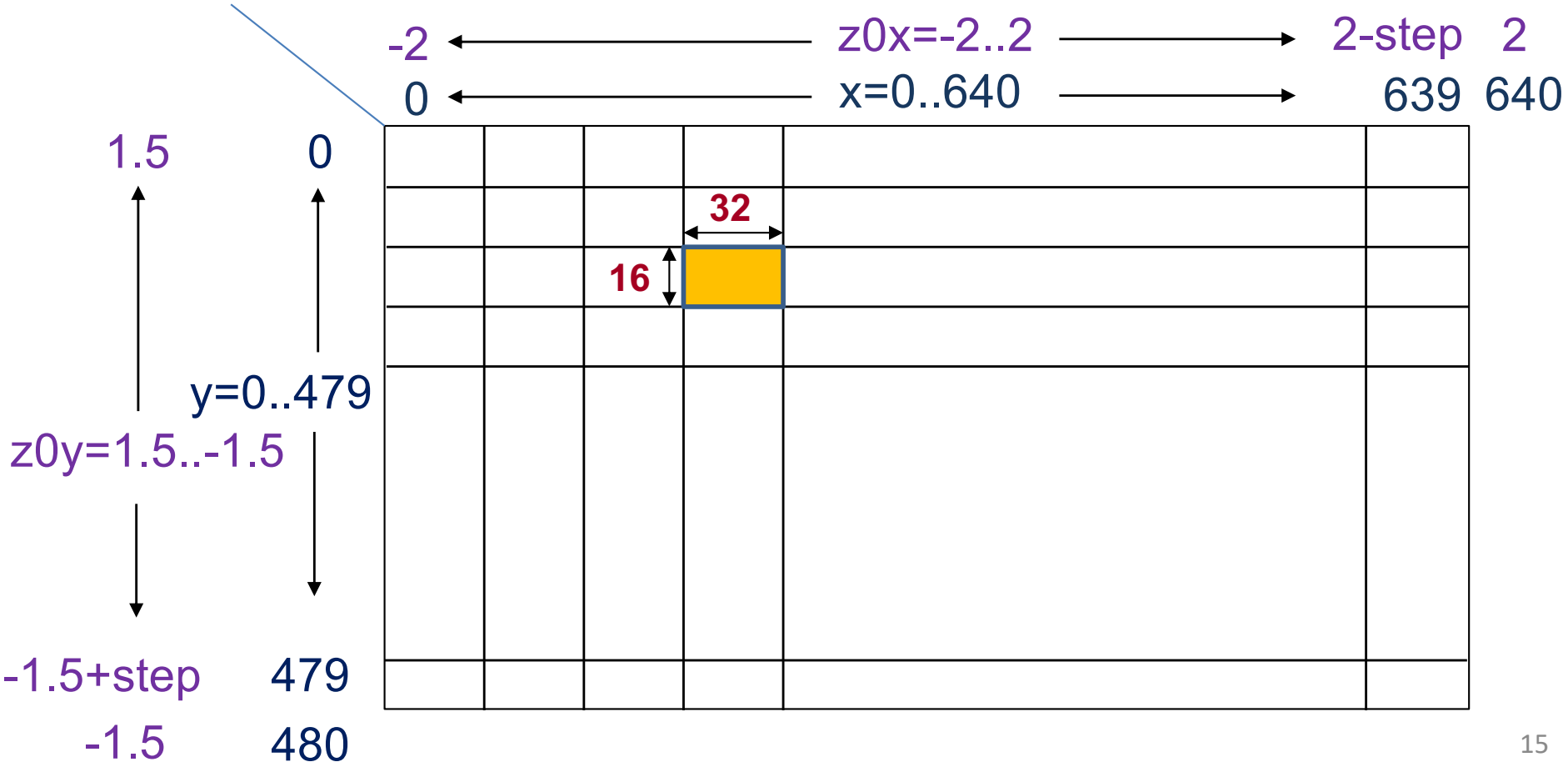
SW15..SW13	MAX_ITER
000	100
001	150
010	200
011	250
100	300
101	350
110	400
111	450

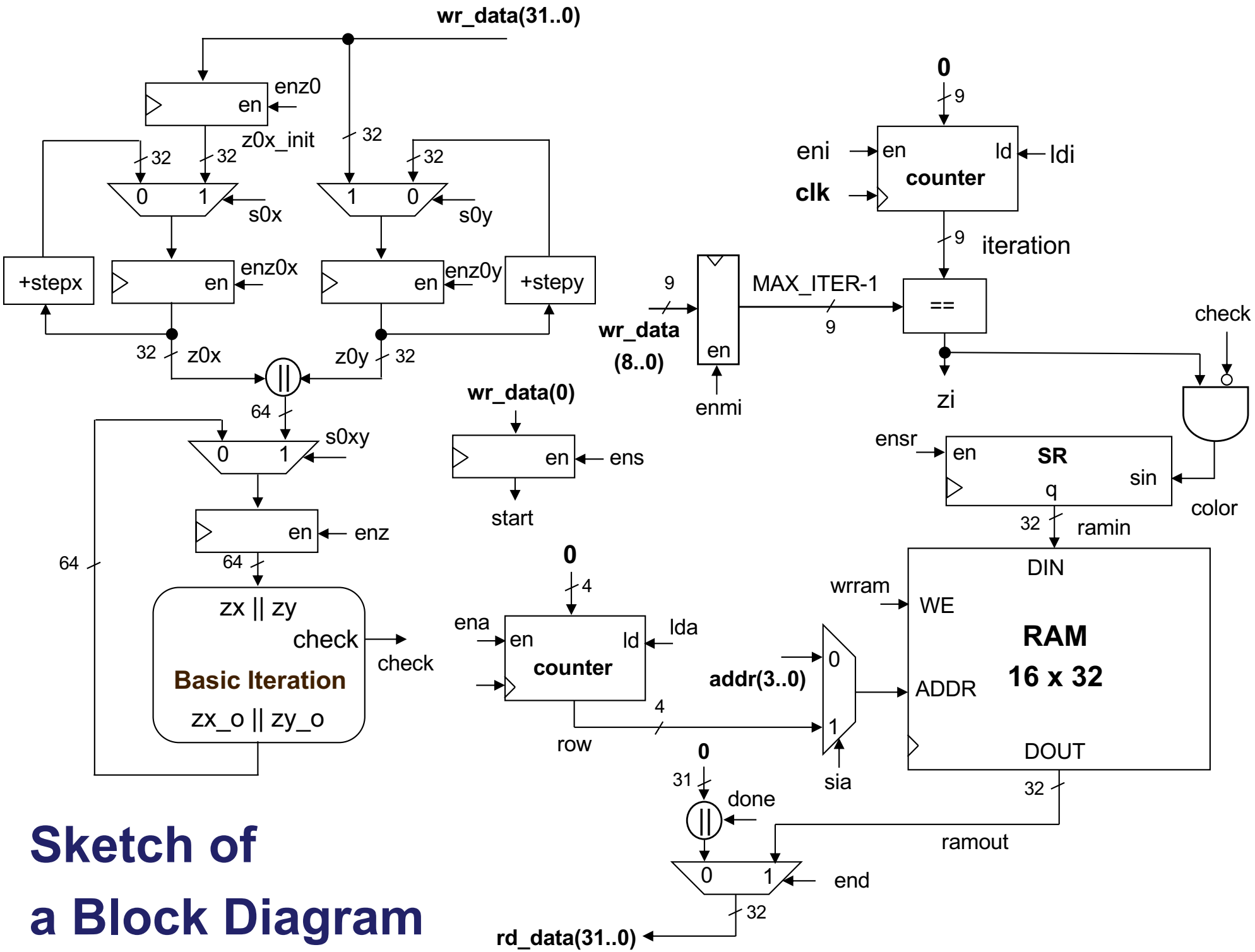
Screen Divided into Tiles

$-2 \le z0x = \mathrm{Re}(z_0) < 2$

$-1.5 < z0y = \mathrm{Im}(z_0) \le 1.5$

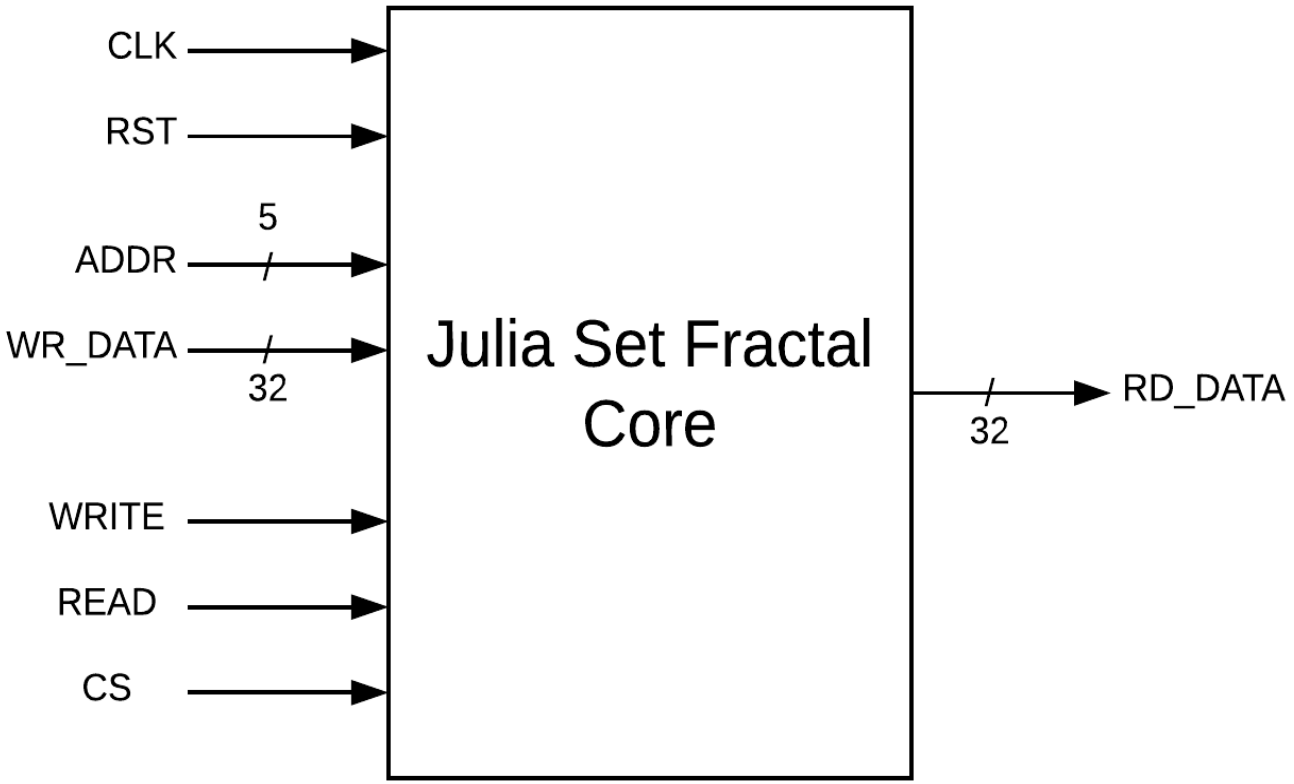
$(x,y) \longleftrightarrow (z0x, z0y)$





Sketch of a Block Diagram

Julia Set Fractal Core



Constants Representing Slot Numbers in VHDL

video/sys/chu_io_map.vhd

```
constant S0_SYS_TIMER : integer := 0;
constant S1_UART1     : integer := 1;
constant S2_LED       : integer := 2;
constant S3_SW        : integer := 3;
constant S4_USER      : integer := 4;
constant S5_XADC      : integer := 5;
constant S6_PWM       : integer := 6;
constant S7_BTN       : integer := 7;
constant S8_SSEG      : integer := 8;
constant S9_SPI       : integer := 9;
constant S10_I2C      : integer := 10;
constant S11_PS2      : integer := 11;
constant S12_DDFS     : integer := 12;
constant S13_ADSR     : integer := 13;
```

Constants Representing Slot Numbers in C/C++

fpga_mcs_vhdl_src/cpp/drv/chu_io_map.h

```
#define S0_SYS_TIMER  0
#define S1_UART1      1
#define S2_LED        2
#define S3_SW         3
#define S4_USER       4
#define S5_XDAC       5
#define S6_PWM        6
#define S7_BTN        7
#define S8_SSEG       8
#define S9_SPI        9
#define S10_I2C      10
#define S11_PS2      11
#define S12_DDFS     12
#define S13_ADSR     13
```

Choosing Slot no. 4 for the sorting core in VHDL

video/sys/chu_io_map.vhd

```
constant S0_SYS_TIMER : integer := 0;
constant S1_UART1     : integer := 1;
constant S2_LED       : integer := 2;
constant S3_SW        : integer := 3;
constant S4_FRACTAL   : integer := 4;
constant S5_XADC      : integer := 5;
constant S6_PWM       : integer := 6;
constant S7_BTN       : integer := 7;
constant S8_SSEG      : integer := 8;
constant S9_SPI       : integer := 9;
constant S10_I2C      : integer := 10;
constant S11_PS2      : integer := 11;
constant S12_DDFS     : integer := 12;
constant S13_ADSR     : integer := 13;
```

Choosing Slot no. 4 for the sorting core in C/C++

fpga_mcs_vhdl_src/cpp/drv/chu_io_map.h

```
#define S0_SYS_TIMER  0
#define S1_UART1      1
#define S2_LED        2
#define S3_SW         3
#define S4_FRACTAL    4
#define S5_XDAC       5
#define S6_PWM        6
#define S7_BTN        7
#define S8_SSEG       8
#define S9_SPI        9
#define S10_I2C      10
#define S11_PS2      11
#define S12_DDFS     12
#define S13_ADSR     13
```

Choosing Slot no. 4 for the sorting core in VHDL

video/sys/mmio_sys_vanilla.vhd

```
slot 4: reserved for user defined
  user_slot4 : entity work.fractal_core
    port map(
       clk      => clk,
       reset    => reset,
       cs       => cs_array(S4_USER),
       read     => mem_rd_array(S4_USER),
       write    => mem_wr_array(S4_USER),
       addr     => reg_addr_array(S4_USER),
       rd_data  => rd_data_array(S4_USER),
       wr_data  => wr_data_array(S4_USER)
    );
 --  rd_data_array(4) <= (others => '0');
```

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

* design adjustments * coding * functional verification

Software Design:

S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

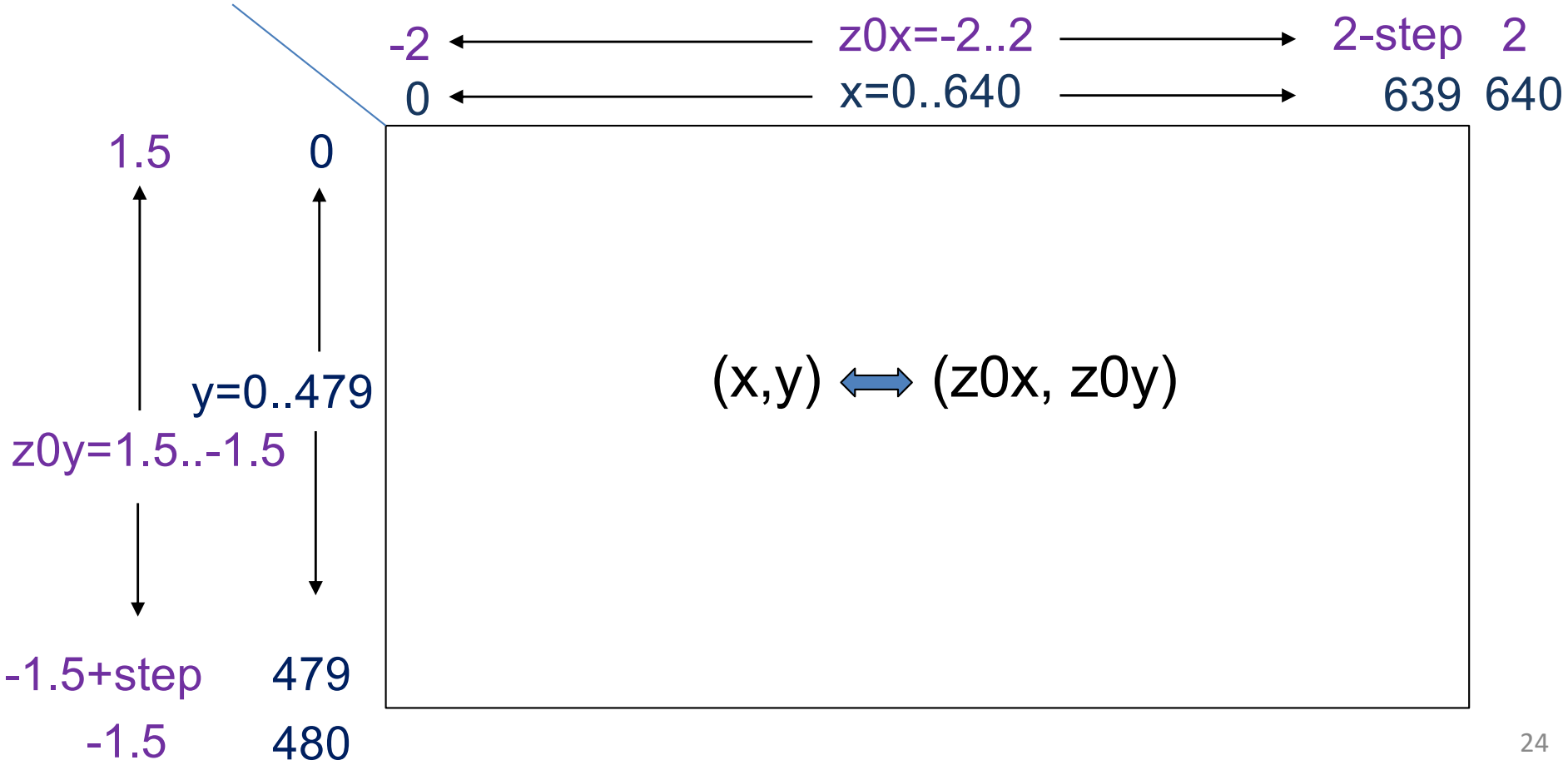
S2. Application based on functions of custom and standard cores

Pseudocode

In the pseudocode below, z_0=z0x+i·z0y, corresponds to one pixel of the display region.

$$-2 \le z0x = \mathrm{Re}(z_0) < 2$$

$$-1.5 < z0y = \mathrm{Im}(z_0) \le 1.5$$



Converting z0x & z0y to x and y

The functions x_conv() and y_conv() are used to convert the coordinates of the complex number z0, z0x & z0y, into x & y coordinates of the pixel.

x = x_conv(z0x) = (z0x-(-2))*(640/4) = 160*(z0x+2)

y = y_conv(z0y) = 480 - (z0y-(-1.5))*(480/3)= 480 – 160*(z0y+1.5)

Frame Core Class Definition (vga_core.h) (1)

```
class FrameCore {
public:
   enum {
      BYPASS_REG = 0x7ffff  /**< bypass control register */
   };
   enum {
    HMAX = 640,  /* 640 pixels per row */
    VMAX = 480   /* 480 pixels per column */
   };
 /* methods */
   FrameCore(uint32_t frame_base_addr);
   ~FrameCore();
```

Frame Core Class Definition (vga_core.h) (2)

```
/**
   * write a pxiel to frame buffer
   * @param x x-coordinate of the pxixel (between 0 and HMAX)
   * @param y y-coordinate of the pxixel (between 0 and VMAX)
   * @param color pixel color
   *
   */
  void wr_pix(int x, int y, int color);

/**
   * clear frame buffer (fill the frame with a specific color)
   * @param color color to fill the frame
   *
   */
  void clr_screen(int color);
```



Lab 5 Video Core

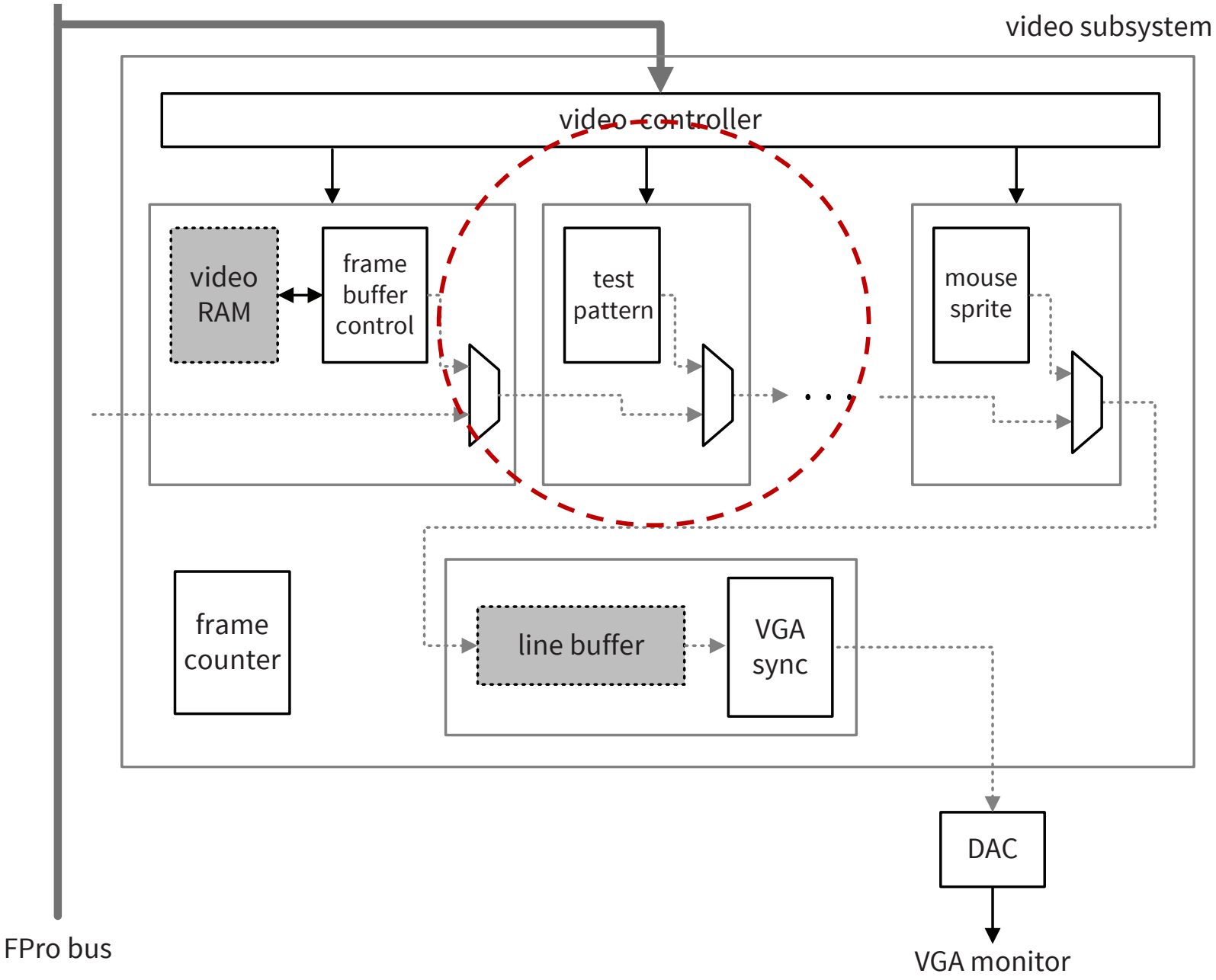
User Interface

- Use BTNS to start/pause the computations.
- Provide support for changing the color of the fractal and background depending on the settings of switches, as described in the table below:

Switches	Background Color	Fractal Color
SW0	White	Black
SW1	Yellow	Red
SW2	Blue	White
SW3	Black	Red
SW4	White	Cyan
SW5	Black	Green
SW6	Red	Blue
SW7	Blue	Yellow

A simple user video core required!

FPro Video Subsystem



Slot Assignment of the Daisy Video Subsystem

Slot	Symbolic name	Description
0	V0_SYNC0	synchronization core
1	V1_MOUSE1	mouse sprite core
2	V2_OSD2	OSD (on screen display) core
3	V3_GHOST3	ghost sprite core
4	V4_USER4	dummy core
5	V5_USER5	dummy core
6	V6_GRAY6	color-to-grayscale core
7	V7_BAR7	bar test-pattern generator core

FPro Video Subsystem

22-bit word space, 24-bit byte space

`11 aaaa aaaa aaaa aaaa aaaa`

1 Frame Buffer Core, with 2^{20}-word address space

`10 xxxv vvaa aaaa aaaa aaaa`

8 Video Cores, each with 2^{14}-word address space

vvv : slot number **aa aaaa aaaa aaaa** : 14-bit address within a slot