EE-334 Digital System Design Presentation

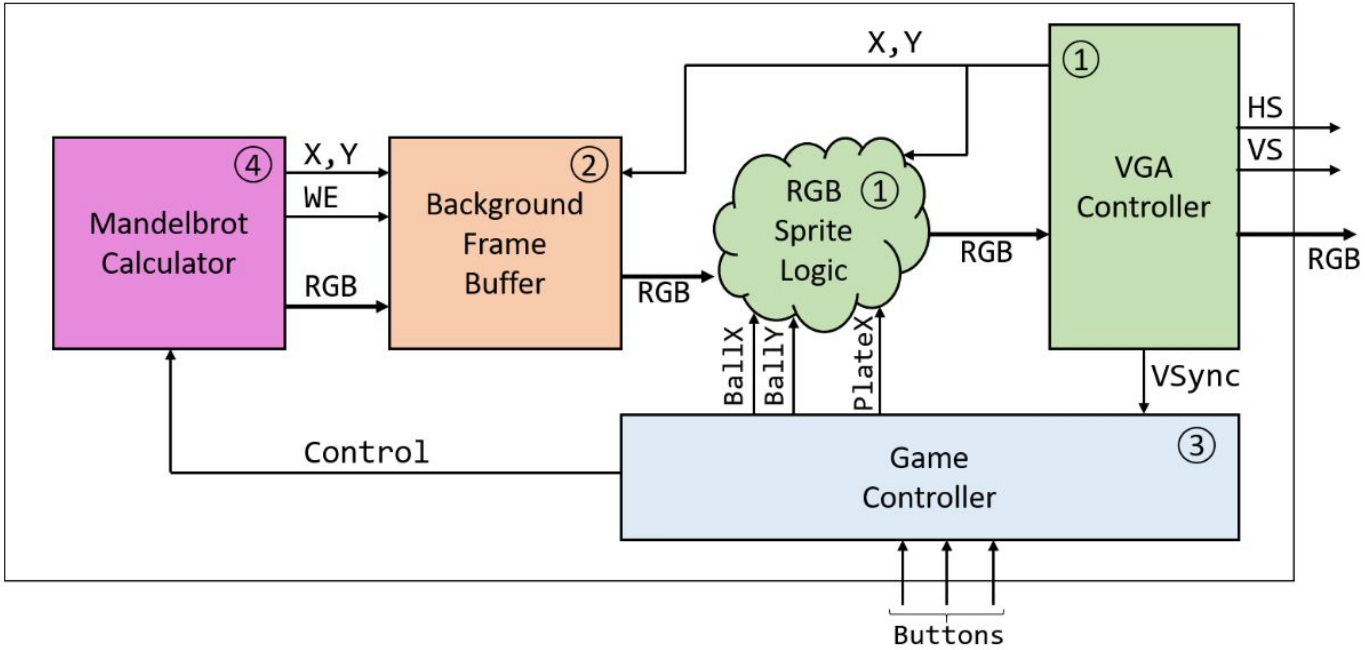
Eliot Abramo,
Emma Gaia Poggiolini,
Alexandre Moy

Introduction - ROBOTO

Outline:

- Overall Architecture
- Mandatory features from lab05 to 08
- Additional Features:
  - *Multiple balls* spawn as the game evolves (ranging from 1 to 3 balls)
  - *Plate shrinks* in size with every collision of the ball
  - *Plate speed increases* as collisions increase
  - *Colored* balls and plate
  - *ROBOTO logo* displayed

Architecture Overview

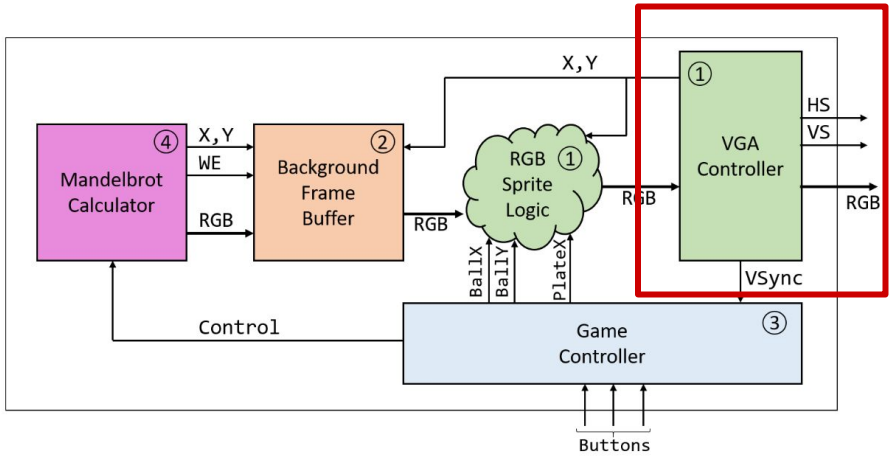


VGA Controller – Overview

- Generates signals to display images on a VGA-compatible monitor

VGA Standard:

- **Analog** signals for Red, Green, and Blue (**RGB**) colors
- **Digital** signals for synchronization (**HS**, **VS**)
  - **Horizontal Sync (HS):** marks the end of each row
  - **Vertical Sync (VS):** marks the end of each frame



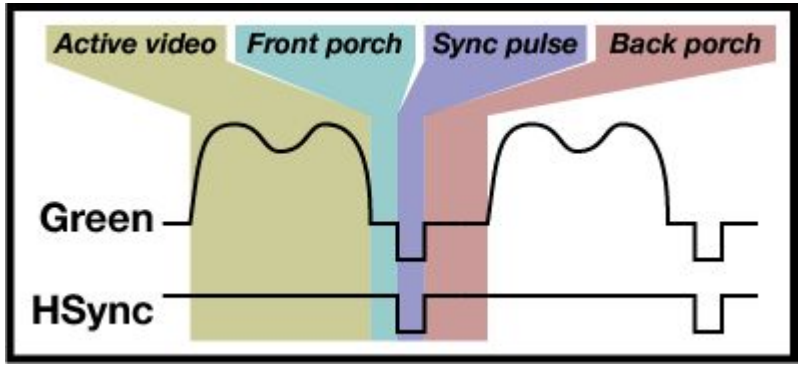
Example application: display pong game

VGA Controller - Architecture

Main Components:

1. **Pixel Counter:** Tracks the current pixel within a row
2. **Line Counter:** Tracks the current line within a frame
3. **Sync Generators:** Generate HS and VS based on counters
4. **RGB Logic:**
    a. Outputs RGB signals based on current pixel
    b. Outputs black during blanking periods

Blanking periods = {Front porch, Sync pulse, Back porch}

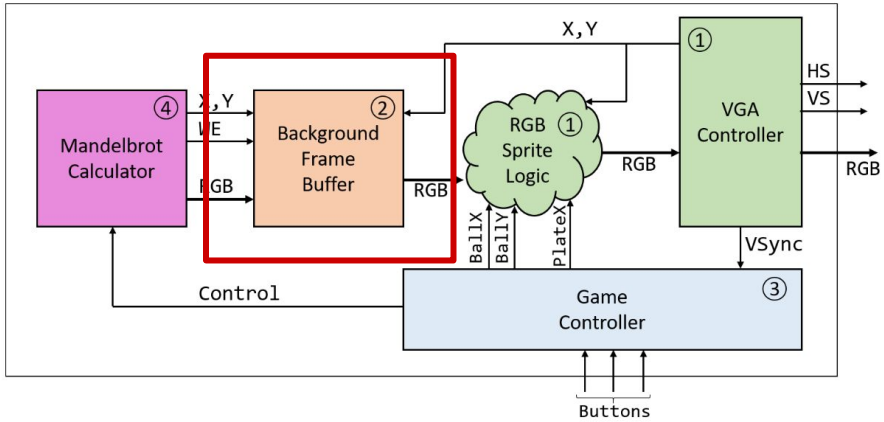


Frame Buffer – Overview

- Memory structure that **stores pixel data** for an entire **frame**

- 2D image mapped to **1D memory addresses**
- Map X, Y coordinates from VGA Controller to memory addresses

- EXTRA: **ROBOTO logo** displayed in top-left corner



Example application: load/store background

Sprite Logic

Track Positions:

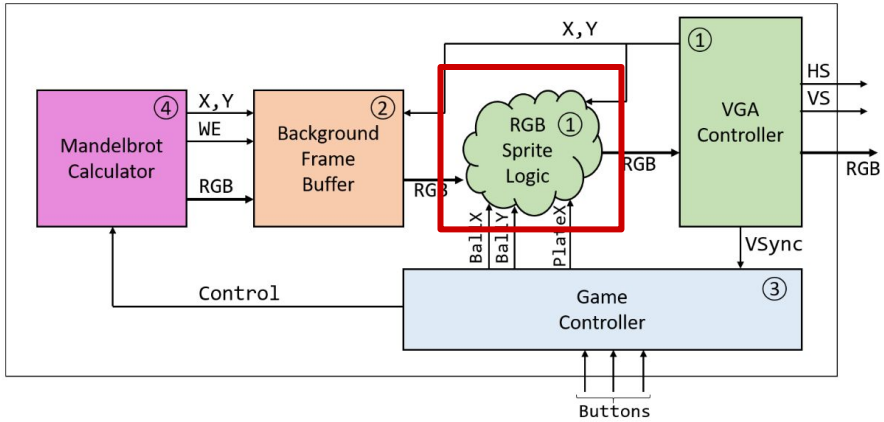
- **Ball Positions** and **Paddle Position**

Overlay Sprites:

- The ball and paddle are *overlaid* onto the background frame (during the active display period of the VGA signal)

Synchronization:

- Sprite positions are updated once per frame
- Frame sync signal (`VSEdgexS0`) for timing updates



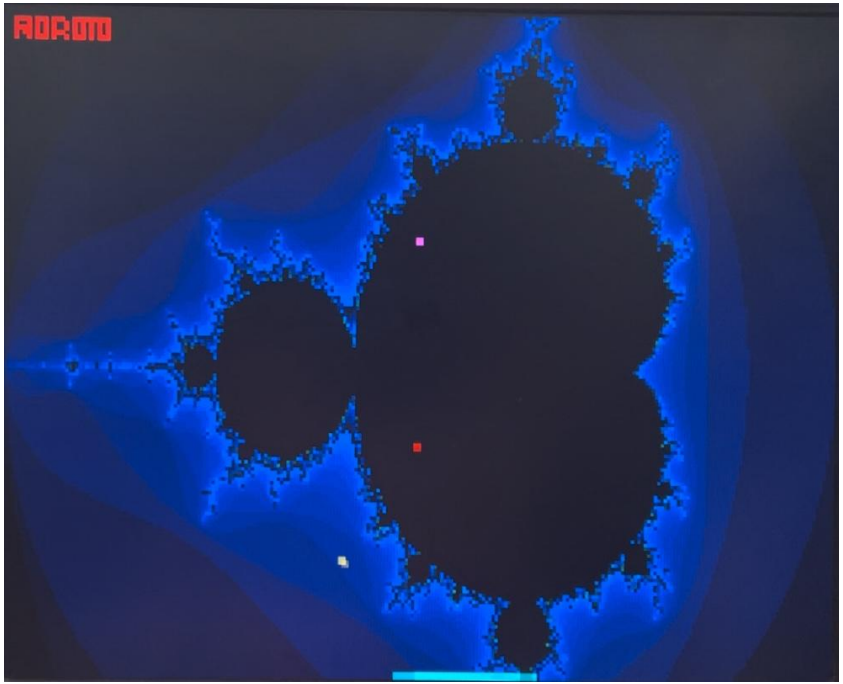
Pong Game – Game Logic

Main Elements:

- **Ball:** Moves across the screen and bounces off walls
- **Paddle (Plate):** Controlled by buttons to move left or right
- **Game State:**
  - Running or Game Over

Control Inputs:

- **Two push-buttons** on the FPGA for paddle movement



Pong Game - Game Logic

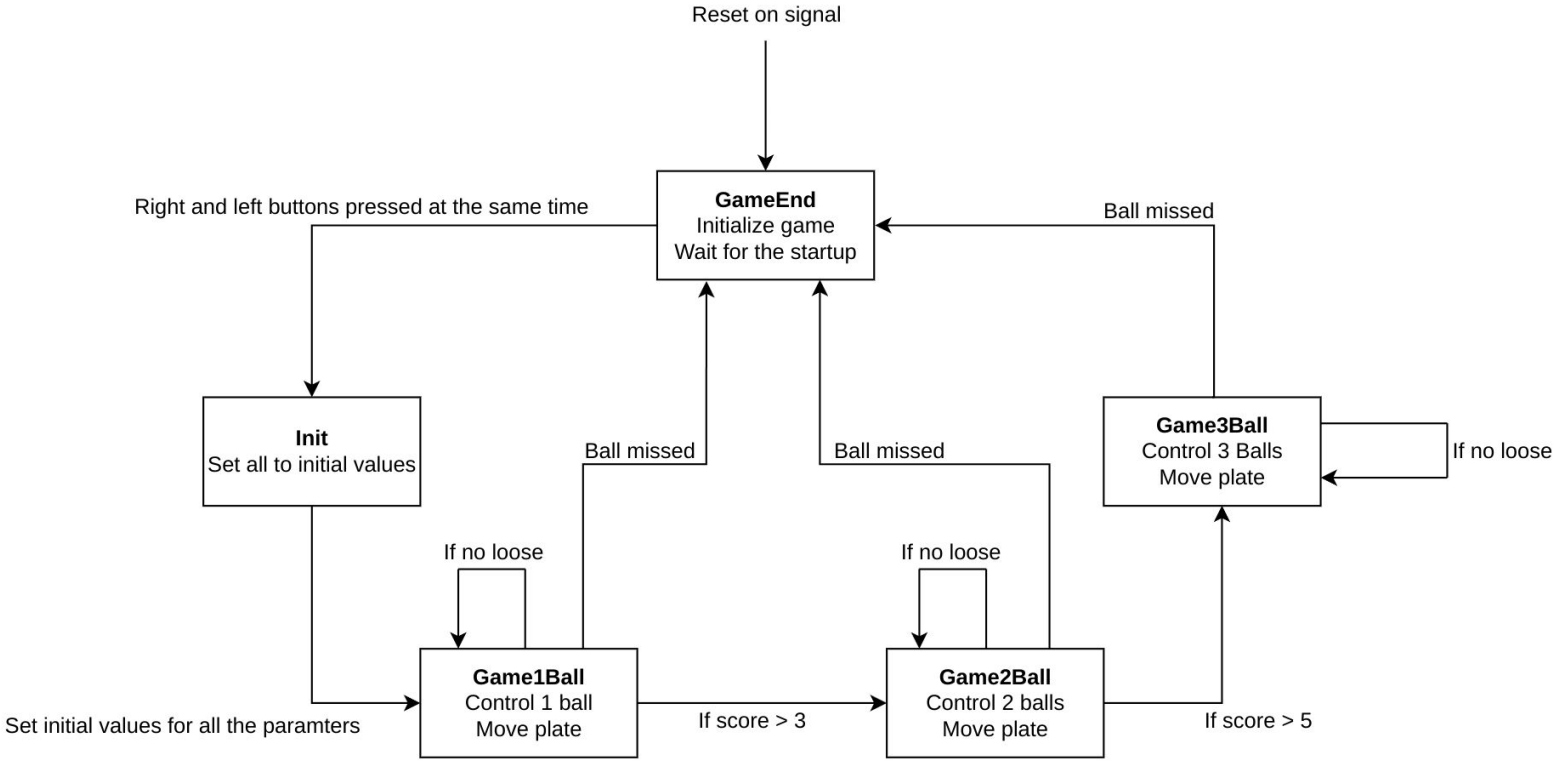
Ball Movement:

- Ball moves diagonally
- **Changes direction** when hitting walls or paddle
- Game Over if ball hits the bottom edge without touching the paddle
- EXTRA: **new ball spawns** after a fixed amount of collisions with the plate (up to 3 balls)
- EXTRA: ball **changes color** at every collision with the plate

Paddle Movement:

- Paddle moves left or right based on button inputs
- EXTRA: plate **shrinks in size** at every collision of the ball
- EXTRA: plate **speed increases** at every collision of the ball

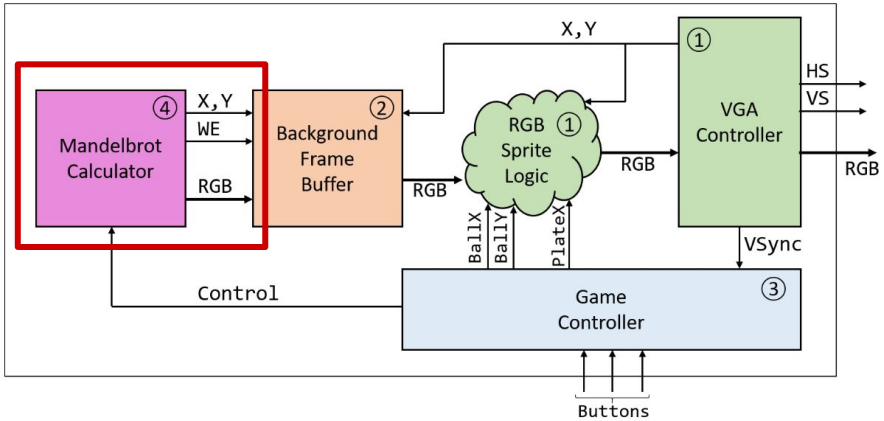
Pong Game – FSM



Mandelbrot

Maths:

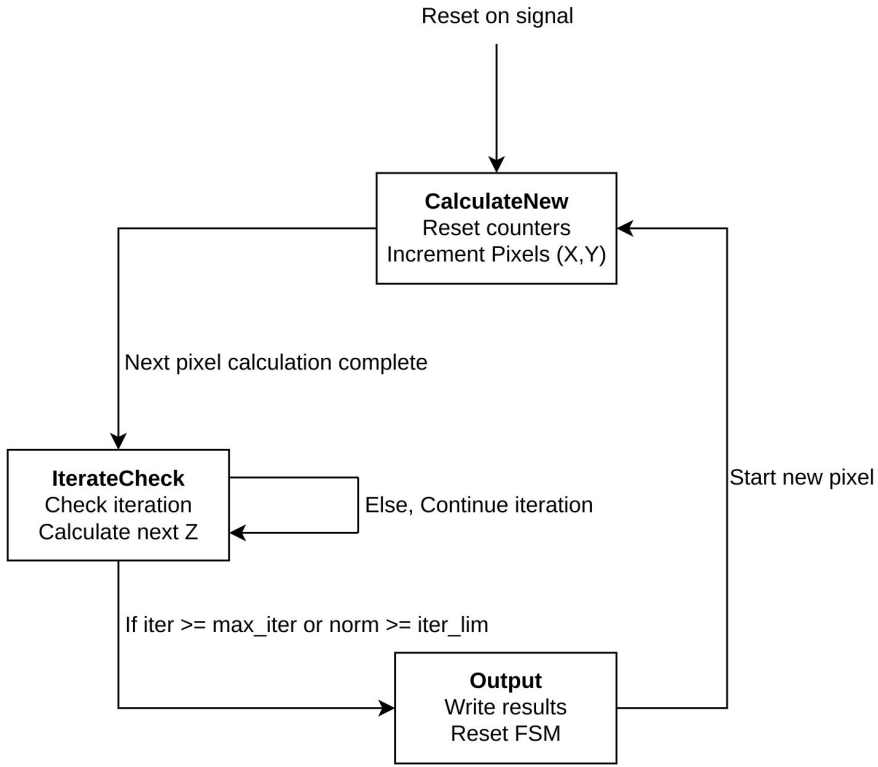
- The Mandelbrot algorithm is processed as seen in the Fixed-point Arithmetic lecture
- The **iteration limit** and the **norm of the complex number** associated to a pixel are checked



Mandelbrot

FSM:

- In the process : all the states are defined (**CalculateNew**, **IterateCheck**, **Output**)
- The limit of the screen is checked in **CalculateNew**
- Iteration limit is checked in **IterateCheck**
- We set the next state of our FSM in the **Output**
- All intermediate signal are updated to process the mandelbrot algorithm



Thank You

Merry Christmas !