Optical Input Targeting Game

By:

Daniel Southern

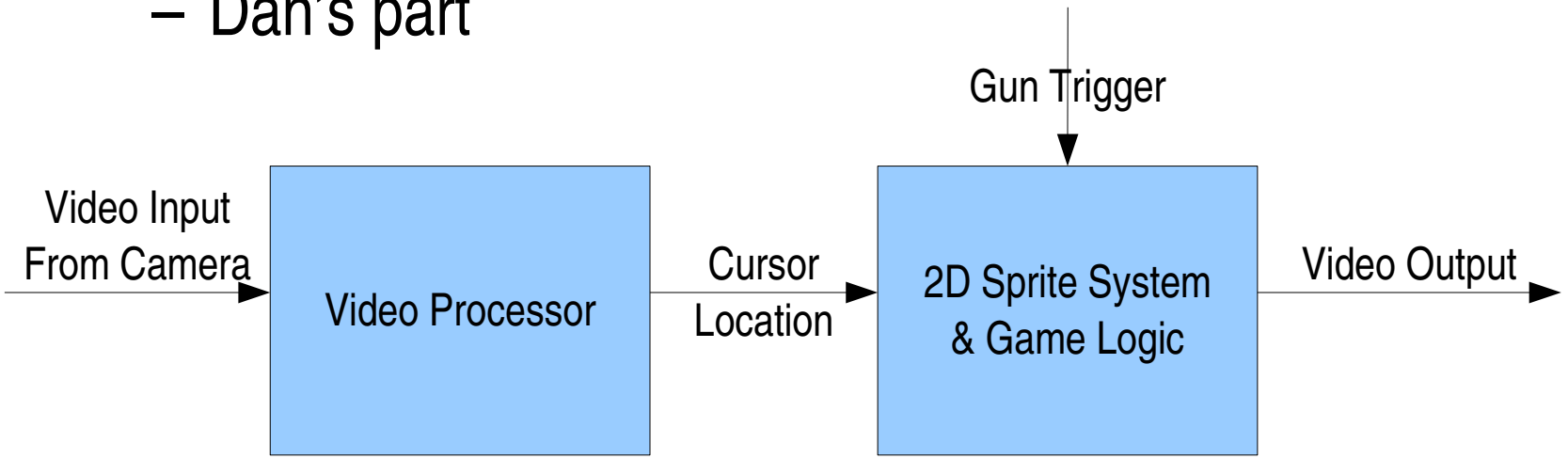
Rachel Bainbridge

Overview

- Sprite-Based 2D game system running a game similar to Duckhunt
- Detects position of laser pointer light in image
- Gun is fired when trigger is pulled, game checks if cursor position overlaps duck position
- Game logic keeps track of score, other game parameters

Components

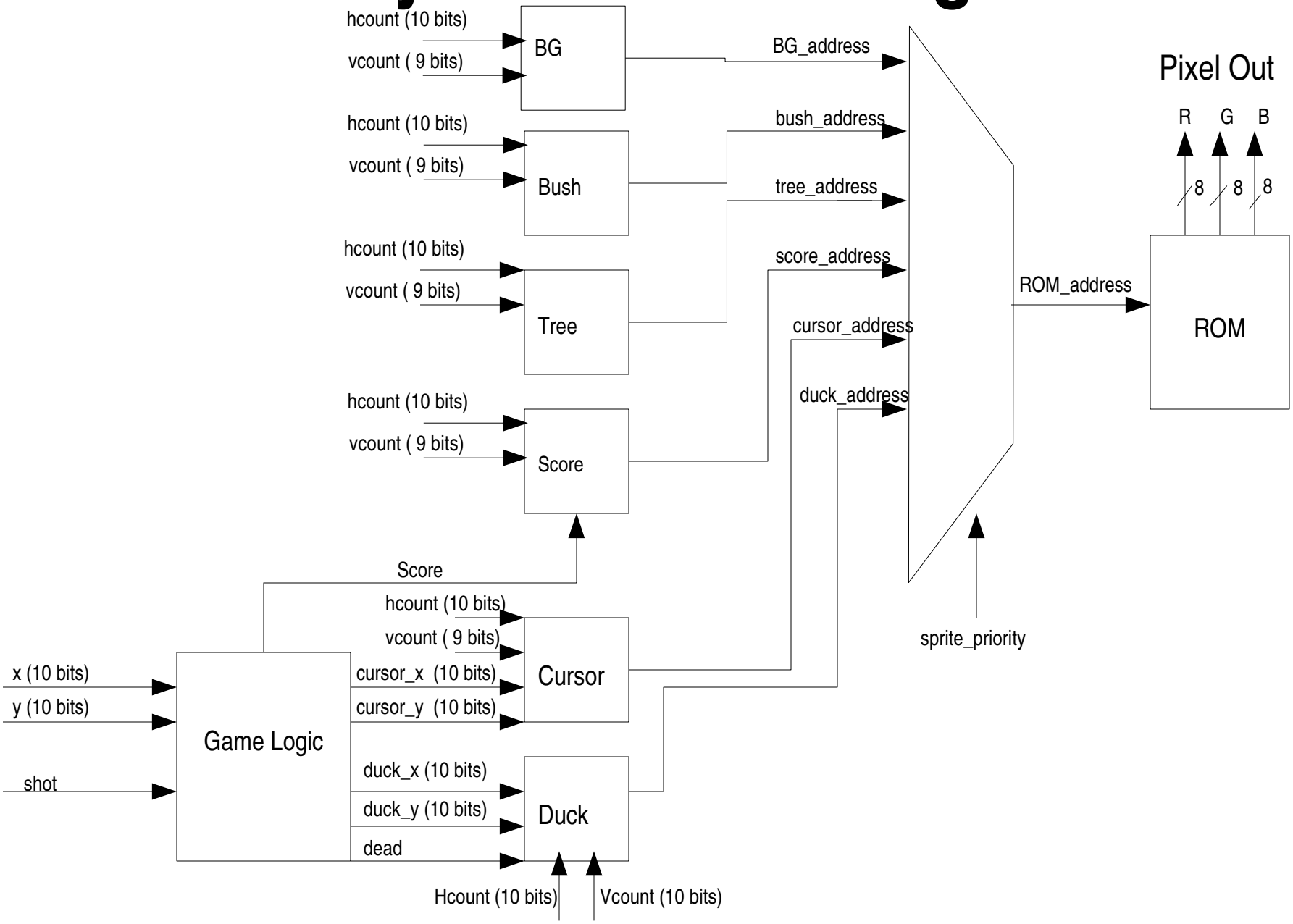
- Sprite-Based 2D Game System
  - Rachel's part
- Video Processing Cursor Locator
  - Dan's part



Sprite-based Game System

- Modules:
  - One module for each type of sprite
  - Sprite's addresses put into muxes to make sure ducks are drawn on top of the background and that trees are drawn on top of the sky
  - Game logic keeps track of score and duck movement/death and outputs score in an ASCII display

Game System Block Diagram



Sprite Module

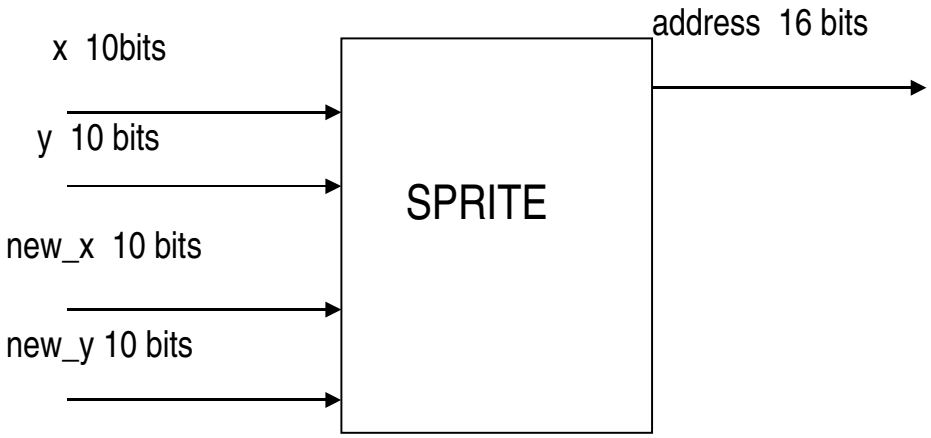
•Inputs are the x and y coordinates for the current pixel being drawn on the screen, and changes of location for the sprite (for background sprites always 0)

•Duck module requires a special input indicating whether the duck is alive or dead

•Sprite module figures out if it has output for the pixel, and outputs the memory address at which it can be found

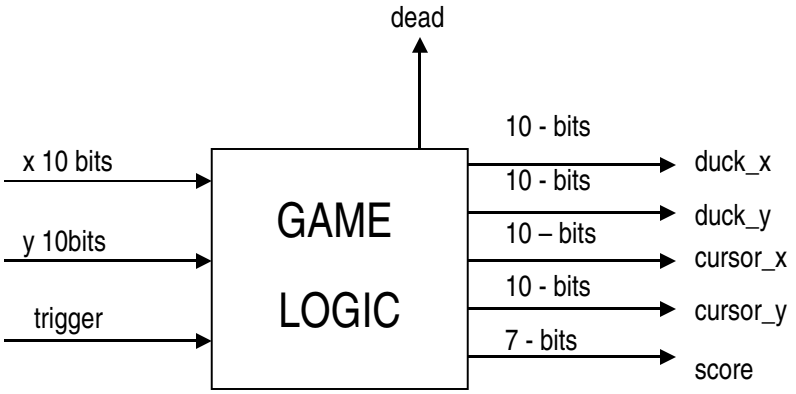
•Module will be duplicated and given the correct parameters depending on which sprite it is handling

•Addresses from the modules go into a series of muxes in order to figure out which address should be used



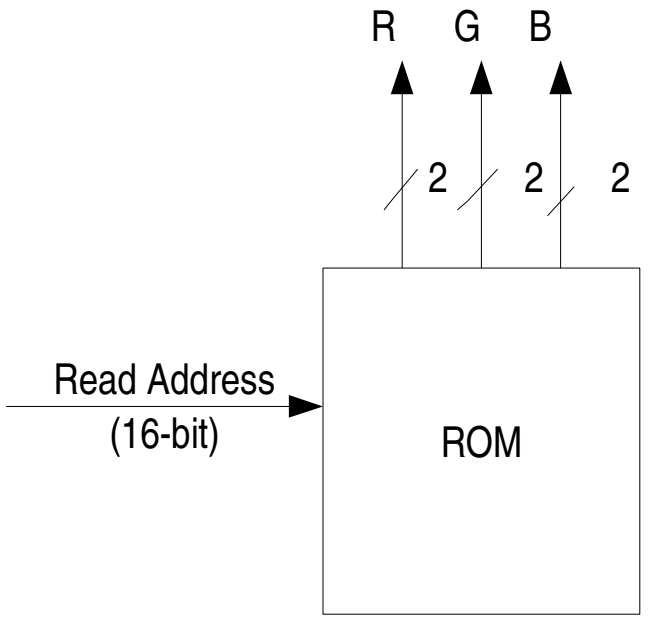
Game Logic

- Inputs are x and y coordinates from cursor locator and the button used for the trigger
- If trigger is pressed compares duck coordinates to cursor coordinates and finds if duck was hit
- Keeps track of duck movement and status (alive or dead) and outputs the location and status to the duck sprite module so it knows what to draw
- Keeps track of score and outputs it to module in charge of ASCII display



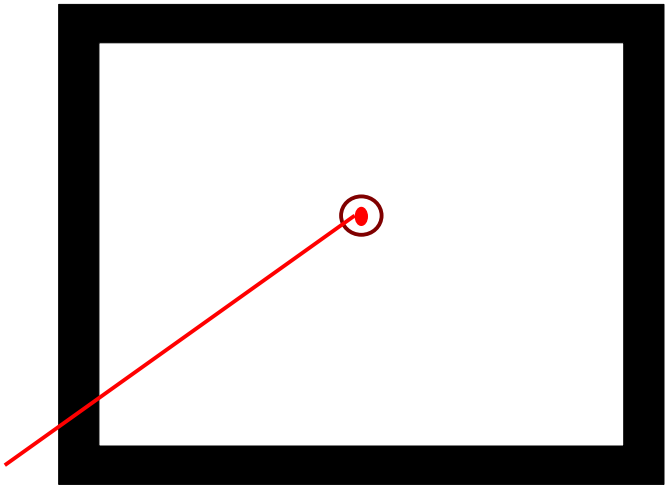
Sprite ROM

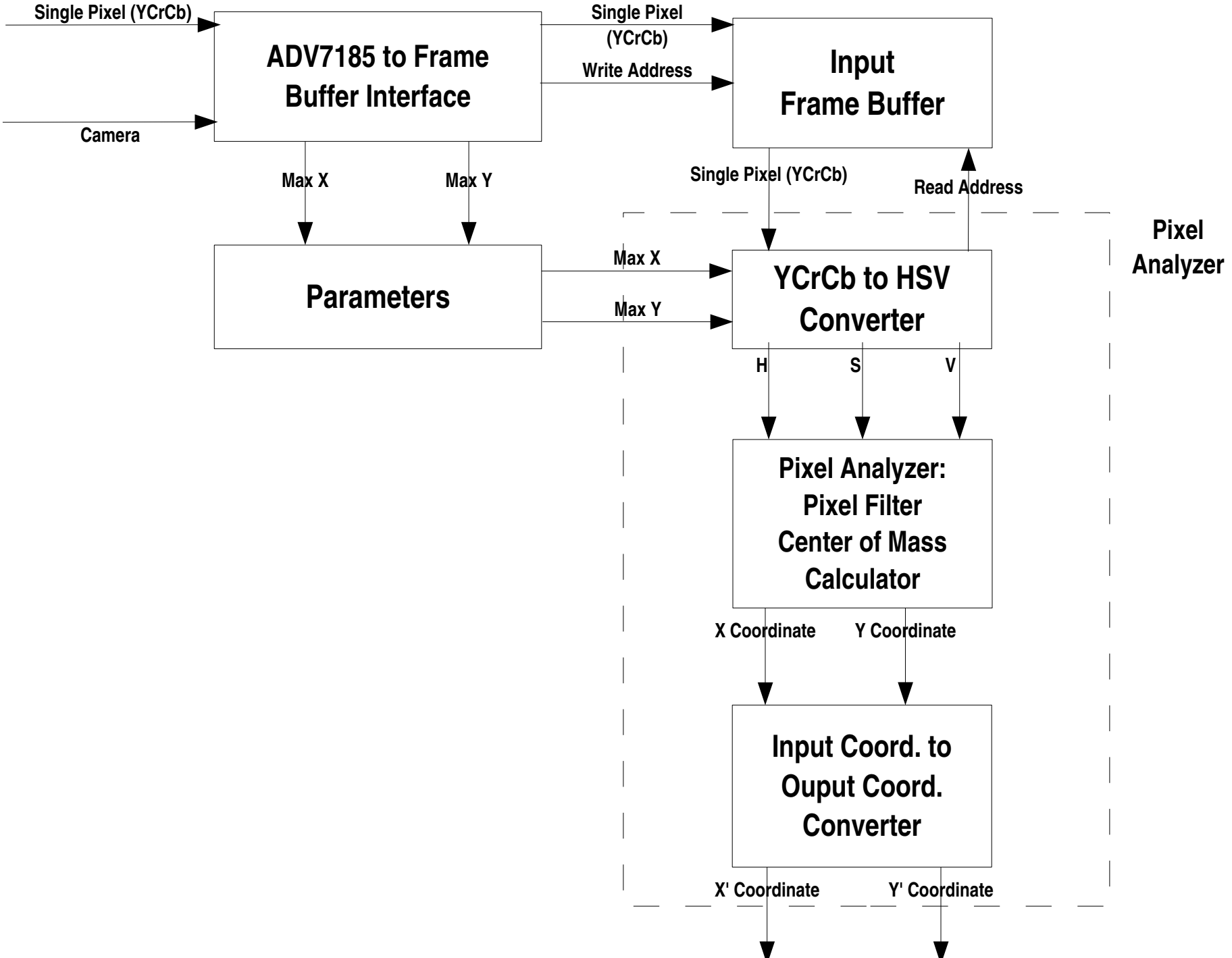
- Rom contains all sprite images
- 3 2-bit wide B-Rams
  - One for each color
  - 2-bit R, G, B colors
- Need approximately 2^16 locations



Video Processing Module

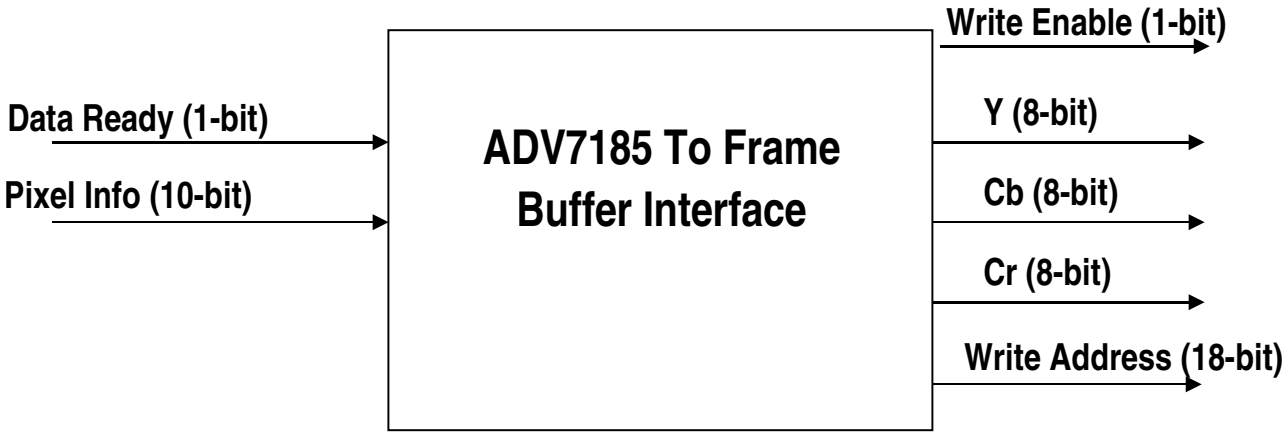
- Locates cursor position in video input frame
- Outputs the coordinates of the cursor in the video output frame




Single Pixel (YCrCb)
Camera
ADV7185 to Frame Buffer Interface
Single Pixel (YCrCb)
Write Address
Input Frame Buffer
Max X
Max Y
Single Pixel (YCrCb)
Read Address
Pixel Analyzer
Parameters
Max X
Max Y
YCrCb to HSV Converter
H
S
V
Pixel Analyzer: Pixel Filter Center of Mass Calculator
X Coordinate
Y Coordinate
Input Coord. to Ouput Coord. Converter
X' Coordinate
Y' Coordinate

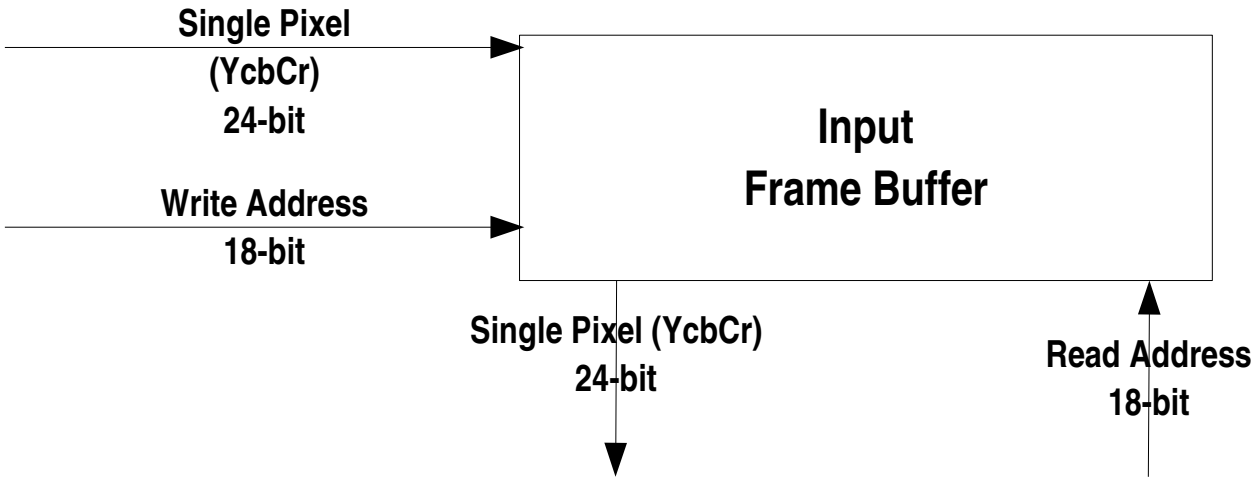
ADV7185 to Frame Buffer Write Interface

- Reads new pixel information from the ADV7185
  - Cb, Cr, Y data is read serially in a specific order
  - Module stores them until a set of three values that describe one pixel is ready
- Generates write addresses, write enable signals for Frame Buffer module



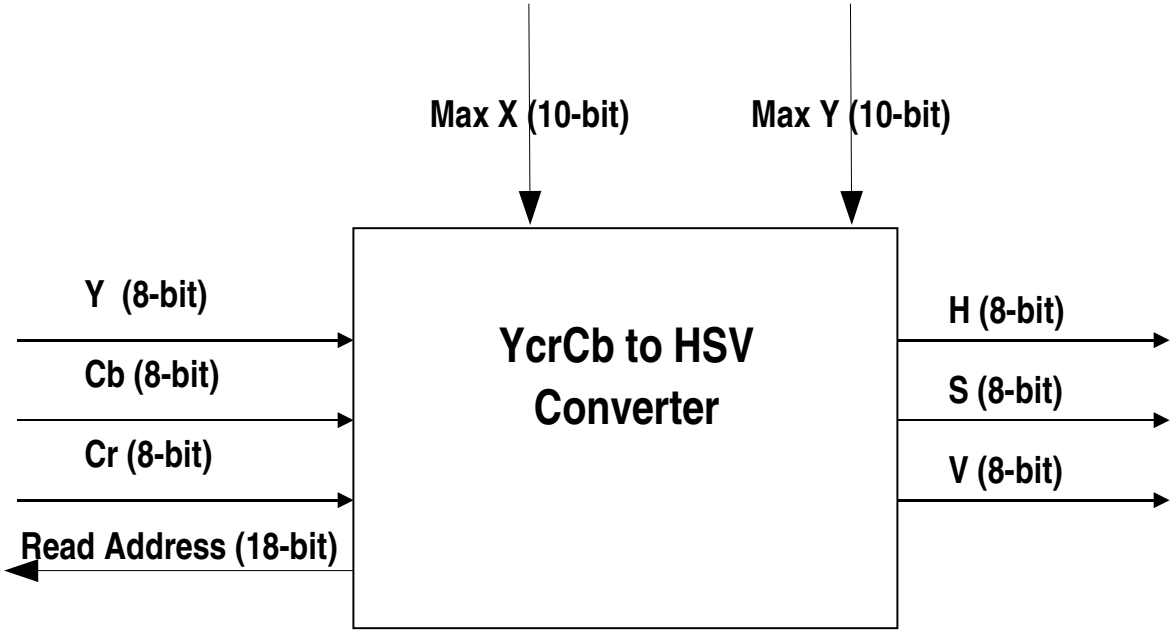
Input Frame Buffer

- Implemented in ZBT memory
- Capacity for 1 Frame at ~ 640x480 pixels, 24-bits/pixel
  - Size: .9MB (Labkit has 4MB)
  - Bandwidth : .9MB * 60Hz = 54MB/s write rate
  - Read rate will be <= write rate
- Frame buffer fits easily on one 512K x 36 ZBT memory



YCrCb to HSV Pixel Converter

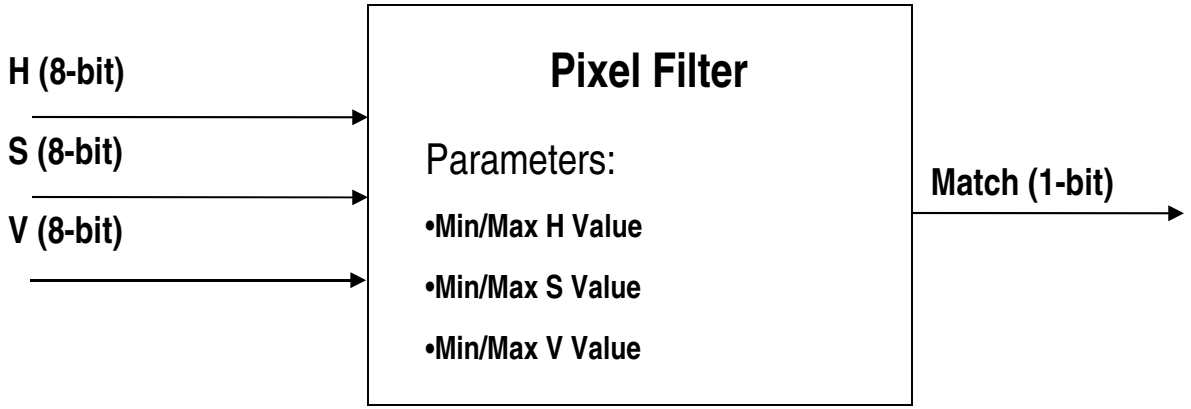
- Pixel processing is inherently easier in the HSV color space
- ADV7185 on Labkit only outputs video in YCrCb format
- Iterates through pixels from frame buffer
- Converts to the HSV color space



Pixel Filter

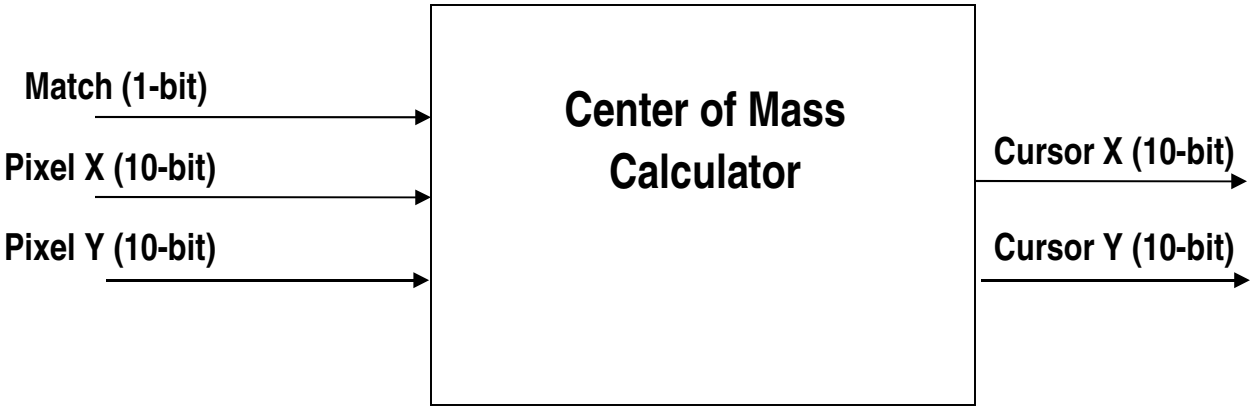
Determines whether a pixel is part of the cursor image

- Parameters will be hard-coded if possible
- Otherwise, implement a calibration functionality
  - Change these values while the system is running



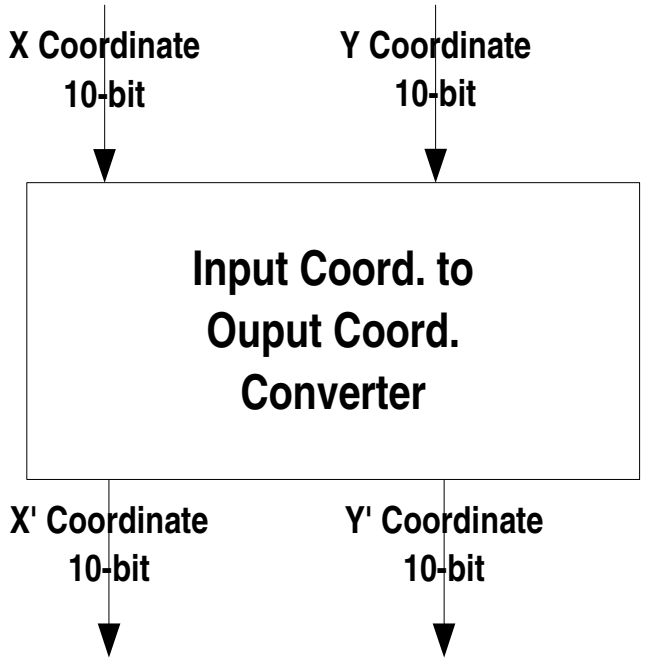
Center of Mass Calculator

- Takes a weighted average over the pixels that match
- Output changes after all pixels in frame are analyzed.
  - Reset calculation upon processing pixel at (0, 0)
  - A new cursor position appears at the output of the module for every frame of video (60Hz)



Input Coordinate to Output Coordinate Converter

- Transforms between pixel locations in the input and output image
- Complexity/Sophistication level will depend on the progress of the project
  - Which factors to take into account?
    - Rotation
    - Scale
    - Skew
    - etc.



Additional Features to be Implemented as Time Allows

- More sophisticated Input / Output transformation
- More complex game behavior
  - Levels, multiple ducks, limited number of shots
- More complex Pixel Filtering
  - Dynamic relative luminosity detector
  - Averaging over several frames