Motion Controlled Fighting Game

Yao Sun
Syed Mohammad Adnan Karim
Syed Talal Ashraf

Initial Concept

- 2- Player fighting game
- Gesture controlled using 4 coloured patches, two on each person
- One patch on shoulder, on patch on fist
- Damage calculation based on velocity of distances between the two patches placed on each player.
- Players can: punch, stand, crouch.

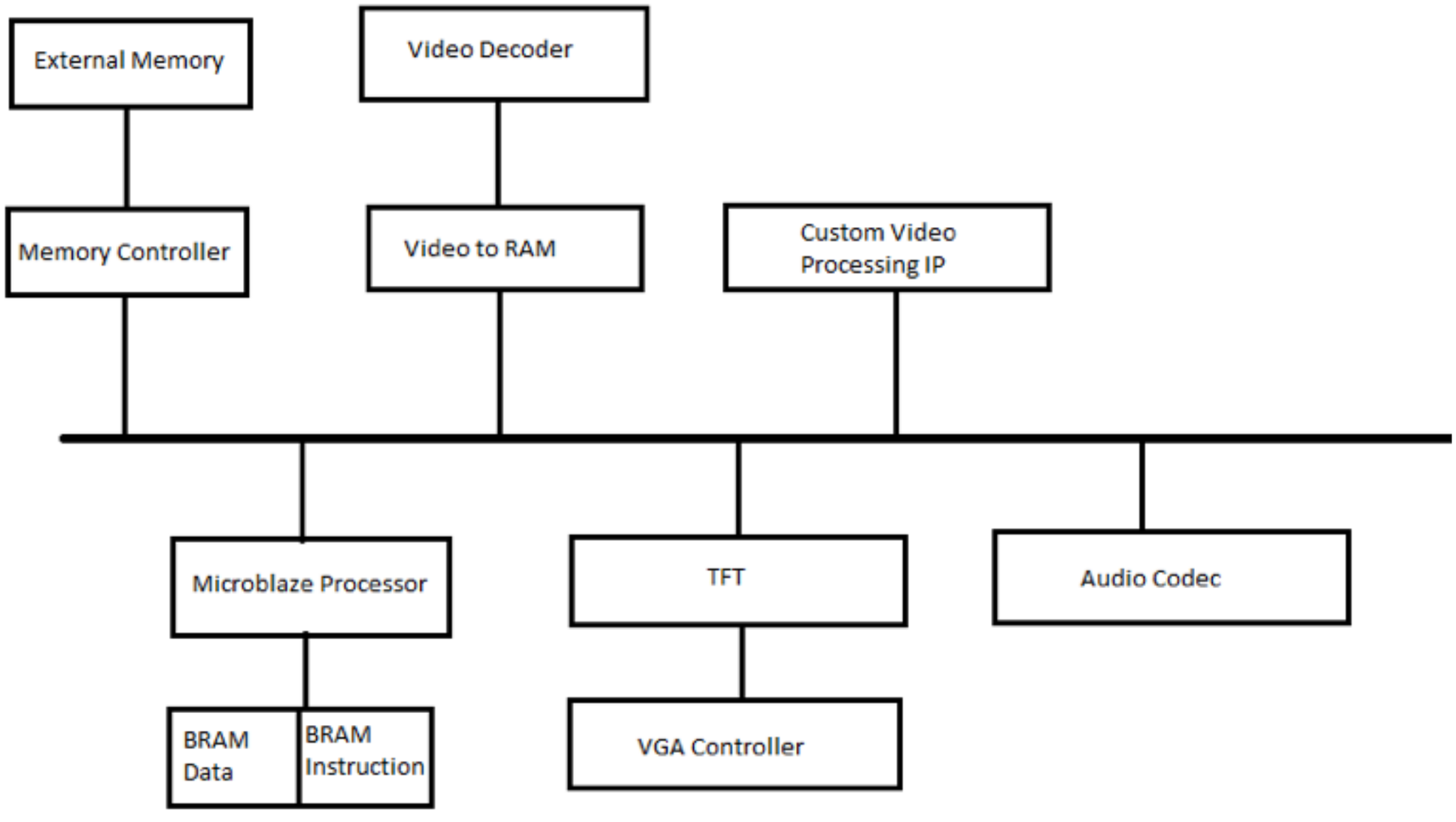
Results

- 1-Player controlled fighting game, second player controlled by AI.

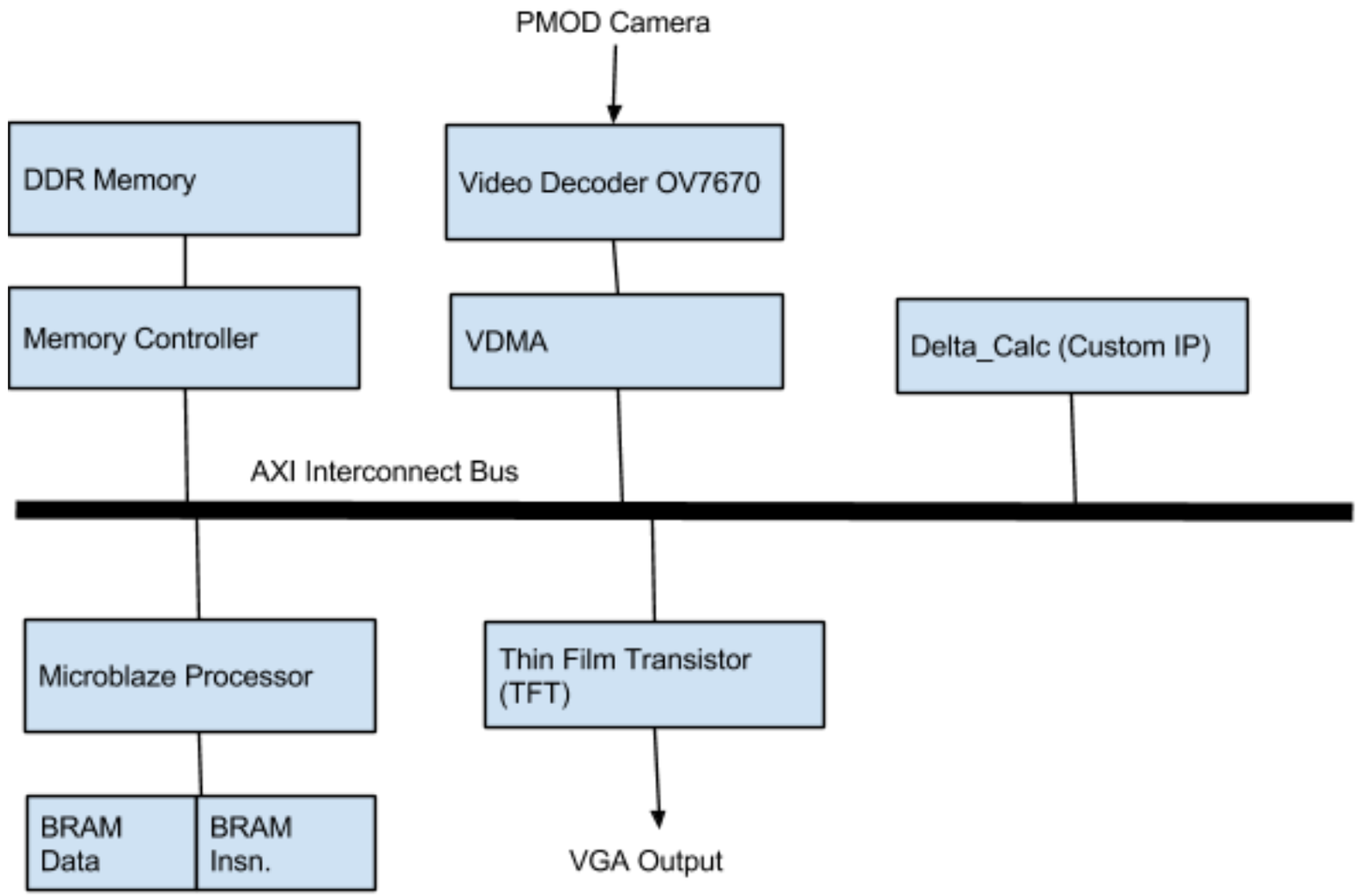
- Simpler colour detection algorithm than initially planned: less robust

- Damage based on position instead of velocity

Hardware Original

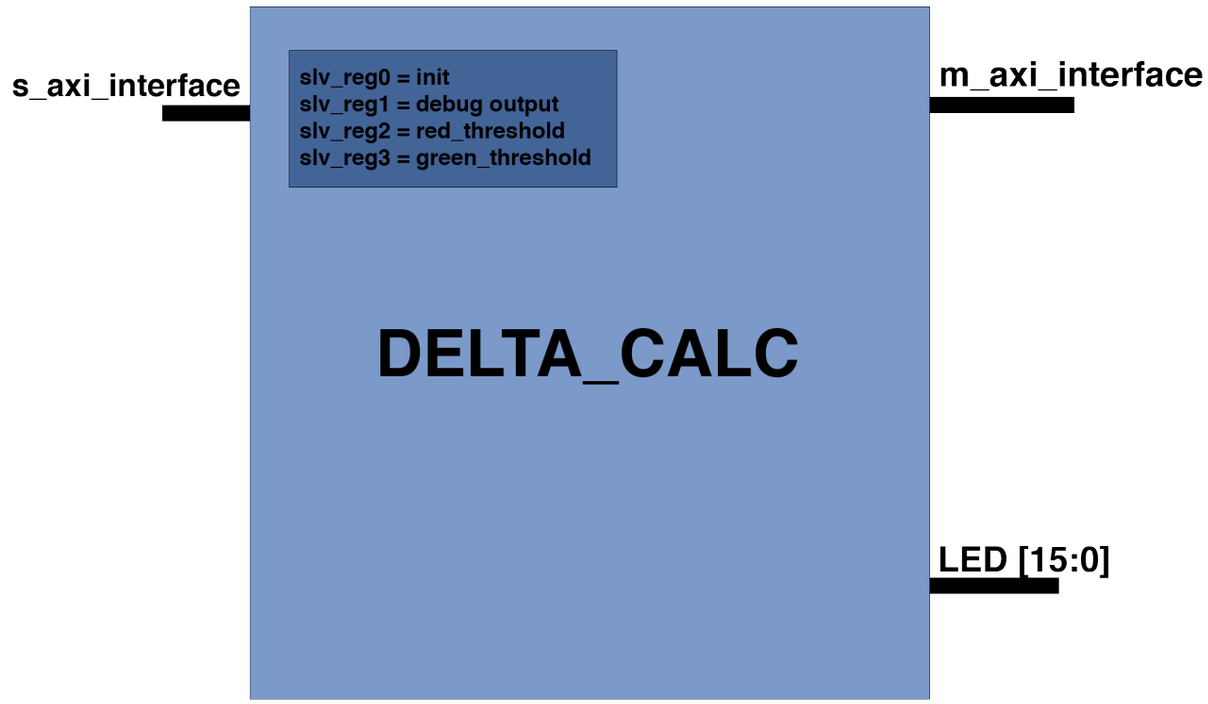


Hardware Final



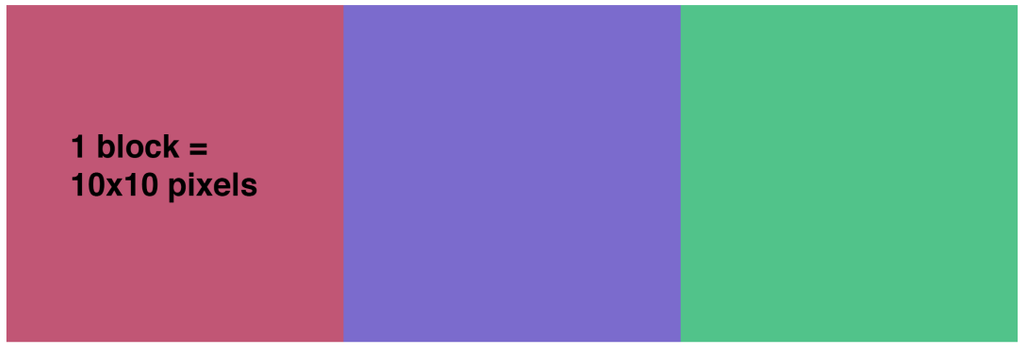
Delta Calc Module

- Works based off of axi_lite interface
- Input output controlled by slave interface
- LEDs linked to debug FSM
- slv_reg0 = init (generate pulse)
- slv_reg1 = debug output
- slv_reg2 = red_threshold
- slv_reg3 = green_threshold
- slv_reg4 = red_xy {0x0000, red_y, red_x}
- slv_reg5 = green_xy 0x0000 0, green_y, green_x}
- slv_reg6 = system_done {bit[0] = 1}
- m_axi: responsible for reading from DDR memory at fixed address (0x81000000)

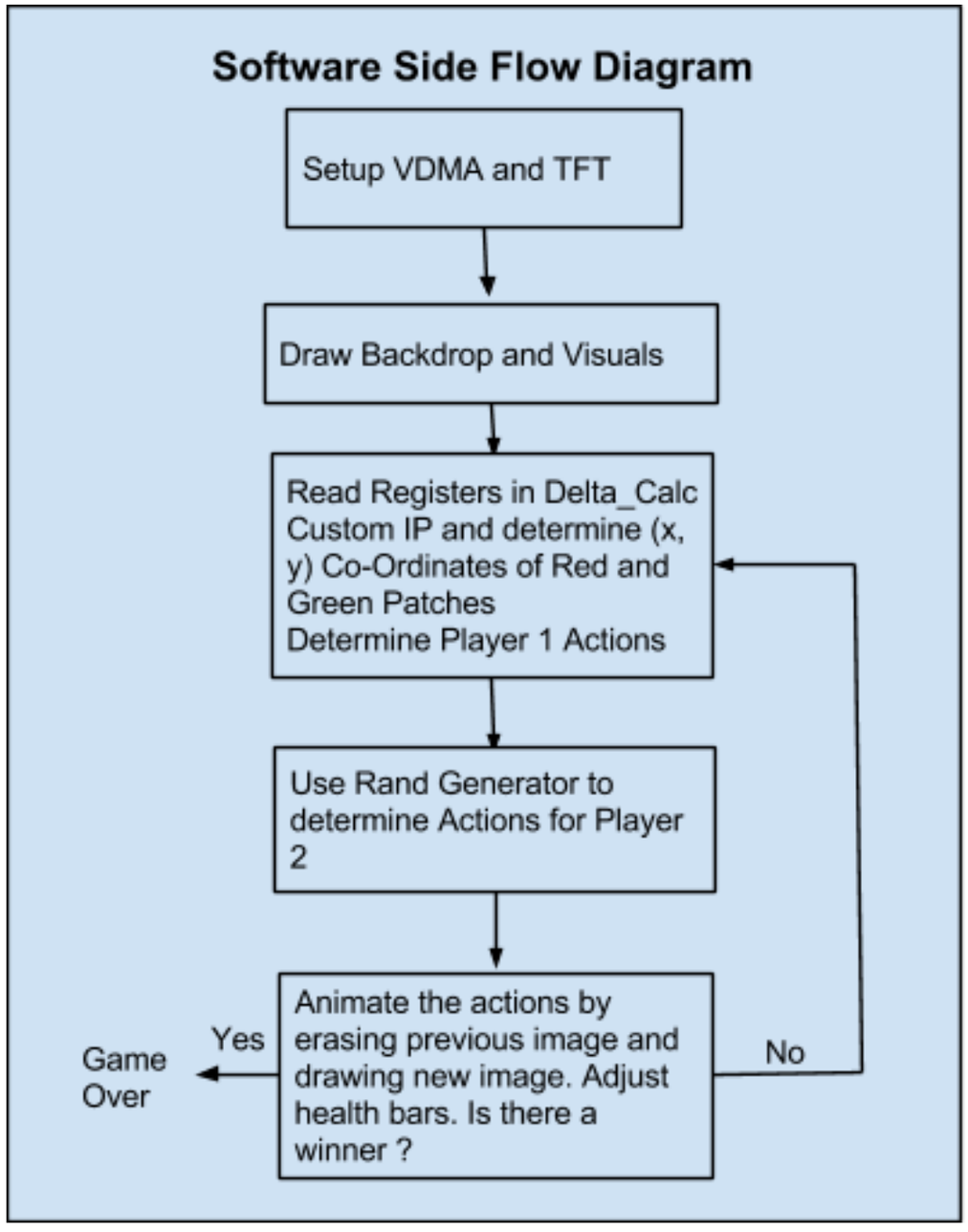


Detection Algorithm

- Algorithm works on “blocks” concept, screen is split up into 32x24 (10x10 pixel) blocks.
- Outer loop iterates through 32x24 blocks.
- Inner loop calculates the address that each block starts at.
- Iterates through 10x10 pixel in DRAM.
- Each pixel is detected through two thresholds one for red, one for green. red = 0x808080, green = 0xC0C0C0.
- If threshold met, red or green counter is incremented. If half the pixels are red or green, block is considered red or green.
- Finishes when we detect the first red and green block. Output data through slv_reg4, slv_reg5.



Software


Software Side Flow Diagram
Setup VDMA and TFT
Draw Backdrop and Visuals
Read Registers in Delta_Calc Custom IP and determine (x, y) Co-Ordinates of Red and Green Patches
Determine Player 1 Actions
Use Rand Generator to determine Actions for Player 2
Animate the actions by erasing previous image and drawing new image. Adjust health bars. Is there a winner ?
Yes
Game Over
No

Output

Design Process

- Up until we got camera working, all of us worked on both hardware and software.
- Split up project into Hardware and Software
- Specified input output ports between Hardware and Software

Borrowed Code

- Borrowed VDMA setup (.c) and interrupt code from Xilinx examples

- Borrowed TFT setup (.c) example from Xilinx examples. Modified to our needs

- Borrowed example code for AXI slave and master interface from Xilinx examples

- Borrowed OV7670 to VDMA code (with some modifications done by us) from code from similar OV7670 setup on ZYBO (Zynq) setup:

  - http://lauri.võsandi.com/hdl/zynq/zybo-ov7670-to-vga.html

What we learned

- Vivado synthesis -> implementation -> bitstream can be slow
- Move to software where possible!
- Writing single always block FSM worked out better
- OV7670 Camera output is noisy and inaccurate.
- Prototyping algorithm in software helps to formulate algorithm. However implementing in Hardware can still be challenging.
- Writing hardware requires more time than we previously anticipated