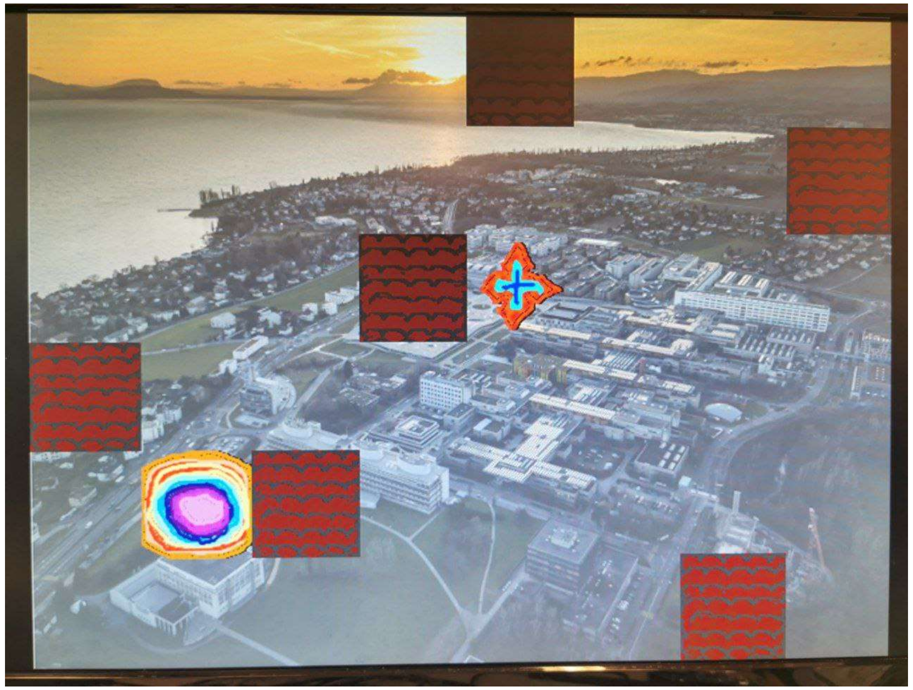


Rocket Rider

Ismael Frei
SCIPER: 301225

EE-390(a) TP de conception de systèmes numériques

Design Features: Checklist (1/3)

- **ARM Processors**
  - Game dynamics:
    - Switching between levels
    - Determining if player won
    - Initialization and Restart of game
    - Console Input/Output
  - Graphics:
    - Writing background in Backbuffer
    - Copying Backbuffer into Framebuffer
- **DRAM memory**
  - Storage of Backbuffer and Framebuffer
  - Storage of Sprite positions for HW Sprite Drawing
- **Memory coherence**
  - Backbuffer and Sprite positions have non-cacheable memory

Design Features: Checklist (2/3)

- **AXI slave**
  - Register file for:
    - Storing signals for Sprite generation (SpriteList Start, SpriteList Length, Backbuffer Start, Backbuffer Length Start, Done)
    - Storing Player and Obstacle coordinates for collision detection (is_collision, PlayerPos, PlayerDir, ObstacleList1, ObstacleList2)
- **AXI master**
  - Memory Reader/Writer File for:
    - Reading Sprite Locations one after the others
    - Writing into the Backbuffer

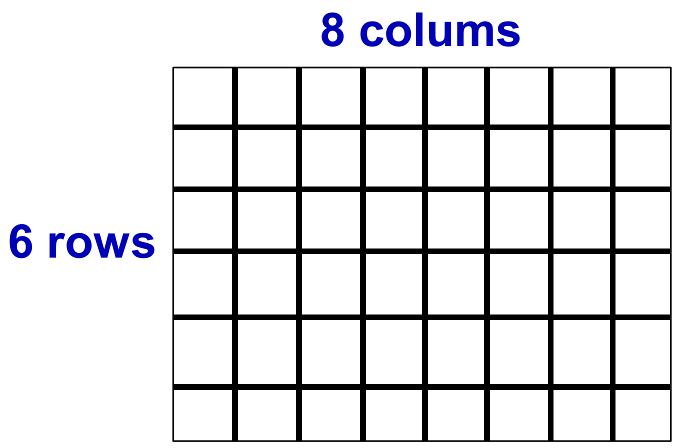
EPFL

Design Features: Checklist (3/3)

- **Interrupts -** none
- **Linux device driver -** none
- **Video subsystem**
  - Bavigap HDMI Output from Lab Session 9
- **Additional peripherals -** none

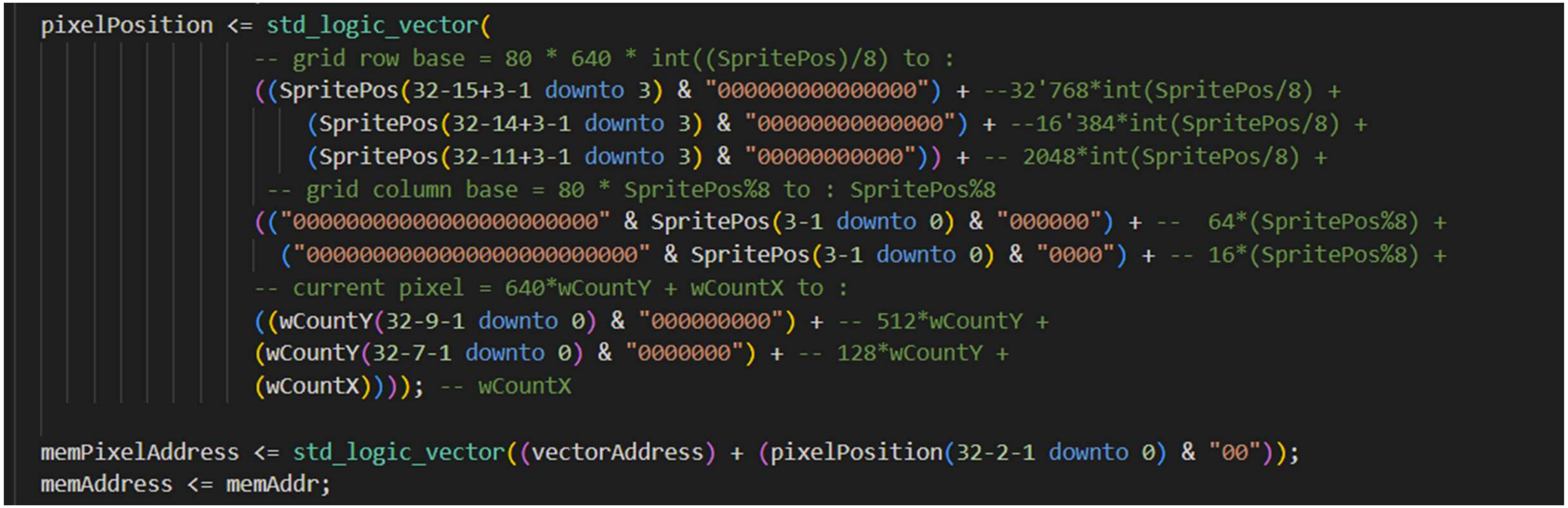
Coordinate systems

- Screen : 640x480
- NDS game: 256x192
- Grid positions: 48 (squares of 80*80 pixels)
  - Get screen coordinates from Grid positions:
    - X = GridPos%8*80
    - Y = Gridpos\8 * 80 * 640
  - Individual pixels of a Sprite in backbuffer memory:
    - Define xCount and yCount both from 0 to 79, yCount increases for every overflow of xCount
    - startingAddres is the addres of the first element in the backbuffer
    - **memAddr = startingAddress + X + Y + 640*yCount + xCount**



Problems and solutions

- Not a single multiplication operator was used in the HDL part. Only bitshifts and additions



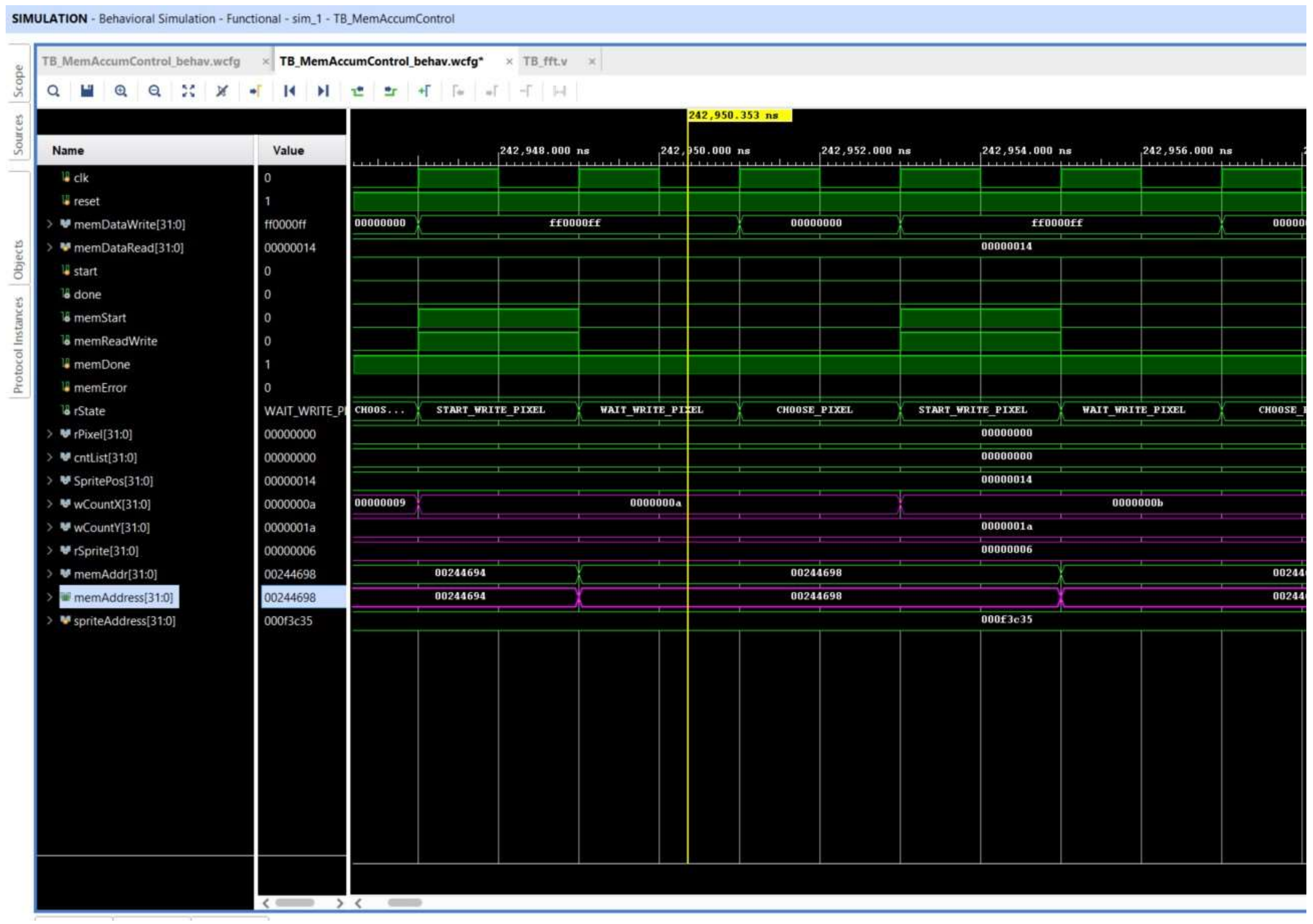
```
pixelPosition <= std_logic_vector(
                -- grid row base = 80 * 640 * int((SpritePos)/8) to :
                ((SpritePos(32-15+3-1 downto 3) & "000000000000000") + --32'768*int(SpritePos/8) +
                    (SpritePos(32-14+3-1 downto 3) & "00000000000000") + --16'384*int(SpritePos/8) +
                    (SpritePos(32-11+3-1 downto 3) & "00000000000")) + -- 2048*int(SpritePos/8) +
                 -- grid column base = 80 * SpritePos%8 to : SpritePos%8
                (("00000000000000000000000" & SpritePos(3-1 downto 0) & "000000") + --  64*(SpritePos%8) +
                  ("0000000000000000000000000" & SpritePos(3-1 downto 0) & "0000") + -- 16*(SpritePos%8) +
                -- current pixel = 640*wCountY + wCountX to :
                ((wCountY(32-9-1 downto 0) & "000000000") + -- 512*wCountY +
                (wCountY(32-7-1 downto 0) & "0000000") + -- 128*wCountY +
                (wCountX)))); -- wCountX

memPixelAddress <= std_logic_vector((vectorAddress) + (pixelPosition(32-2-1 downto 0) & "00"));
memAddress <= memAddr;
```

Testbenching

2 Testbenches (Verilog) for :

a) testing Sprite Addresses and control State machine for sprite generation
b) testing collision detection

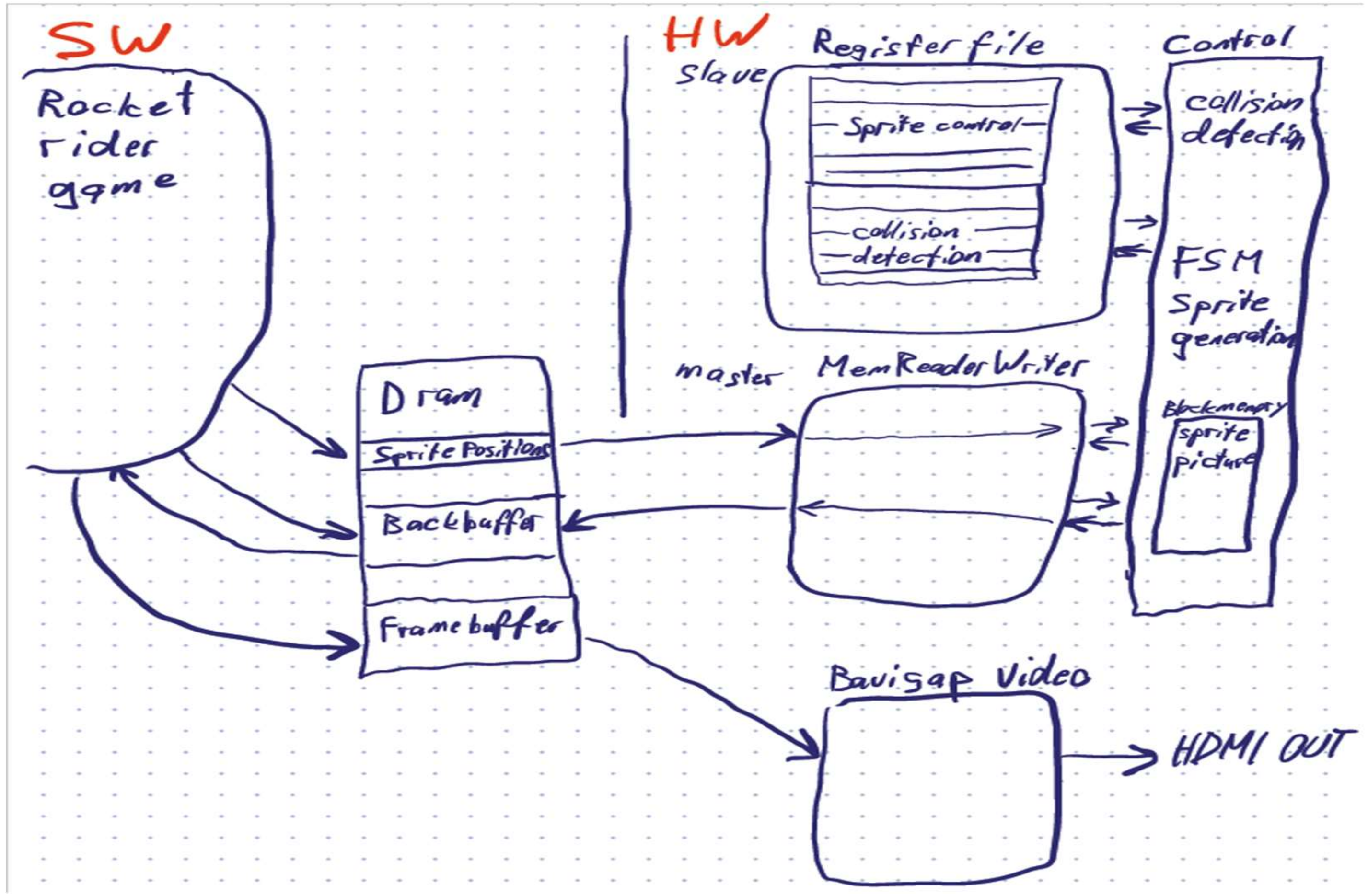


SIMULATION - Behavioral Simulation - Functional - sim_1 - TB_MemAccumControl

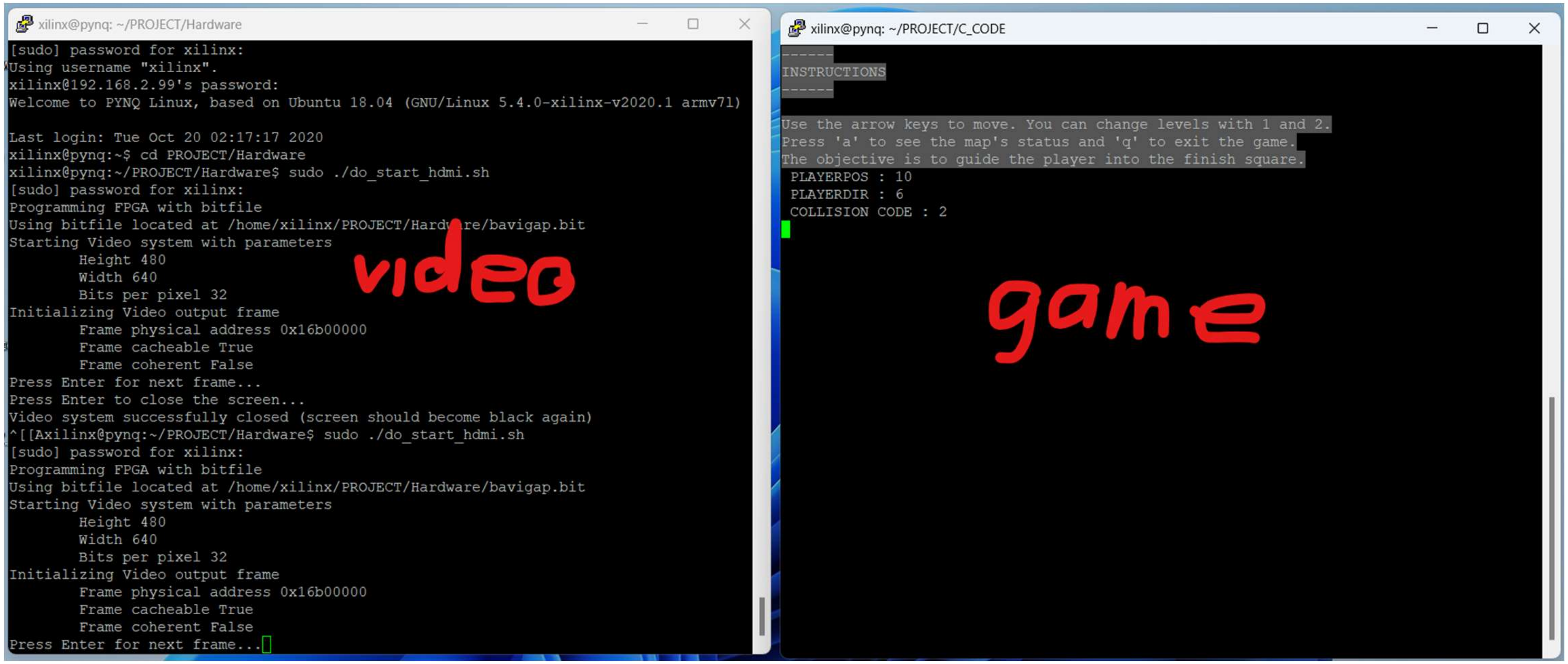
TB_MemAccumControl_behav.wcfg × TB_MemAccumControl_behav.wcfg* × TB_fft.v ×

Name	Value
clk	0
reset	1
memDataWrite[31:0]	ff0000ff
memDataRead[31:0]	00000014
start	0
done	0
memStart	0
memReadWrite	0
memDone	1
memError	0
rState	WAIT_WRITE_PI
rPixel[31:0]	00000000
cntList[31:0]	00000000
SpritePos[31:0]	00000014
wCountX[31:0]	0000000a
wCountY[31:0]	0000001a
rSprite[31:0]	00000006
memAddr[31:0]	00244698
memAddress[31:0]	00244698
spriteAddress[31:0]	000f3c35

Simplified Blockdiagram of the Project



Screenshot of the 2 open terminals through ssh


xilinx@pynq: ~/PROJECT/Hardware
video
xilinx@pynq: ~/PROJECT/C_CODE
game

```
[sudo] password for xilinx:
Using username "xilinx".
xilinx@192.168.2.99's password:
Welcome to PYNQ Linux, based on Ubuntu 18.04 (GNU/Linux 5.4.0-xilinx-v2020.1 armv7l)

Last login: Tue Oct 20 02:17:17 2020
xilinx@pynq:~$ cd PROJECT/Hardware
xilinx@pynq:~/PROJECT/Hardware$ sudo ./do_start_hdmi.sh
[sudo] password for xilinx:
Programming FPGA with bitfile
Using bitfile located at /home/xilinx/PROJECT/Hardware/bavigap.bit
Starting Video system with parameters
        Height 480
        Width 640
        Bits per pixel 32
Initializing Video output frame
        Frame physical address 0x16b00000
        Frame cacheable True
        Frame coherent False
Press Enter for next frame...
Press Enter to close the screen...
Video system successfully closed (screen should become black again)
^[[Axilinx@pynq:~/PROJECT/Hardware$ sudo ./do_start_hdmi.sh
[sudo] password for xilinx:
Programming FPGA with bitfile
Using bitfile located at /home/xilinx/PROJECT/Hardware/bavigap.bit
Starting Video system with parameters
        Height 480
        Width 640
        Bits per pixel 32
Initializing Video output frame
        Frame physical address 0x16b00000
        Frame cacheable True
        Frame coherent False
Press Enter for next frame...
```

```
------
INSTRUCTIONS
------

Use the arrow keys to move. You can change levels with 1 and 2.
Press 'a' to see the map's status and 'q' to exit the game.
The objective is to guide the player into the finish square.
 PLAYERPOS : 10
 PLAYERDIR : 6
 COLLISION CODE : 2
```

Pictures of the game