River-Raid III (CU Edition)

W4840 Embedded System Design

Rojan, Xinhao

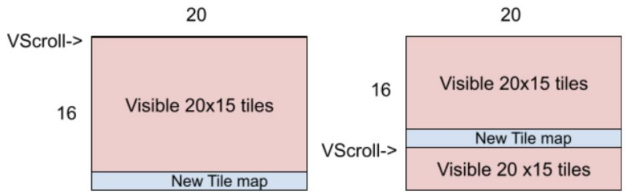
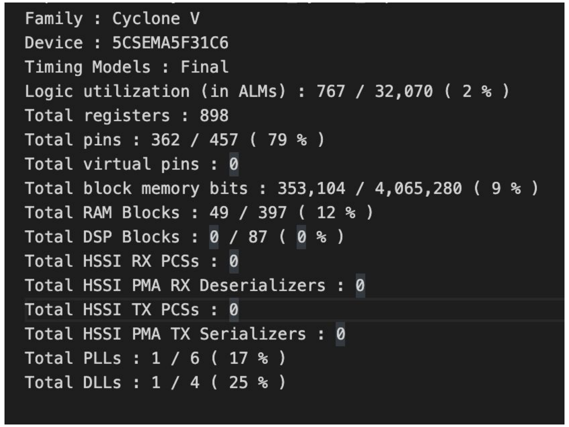


Figure 3 : Vertical Scrolling of background

Overview

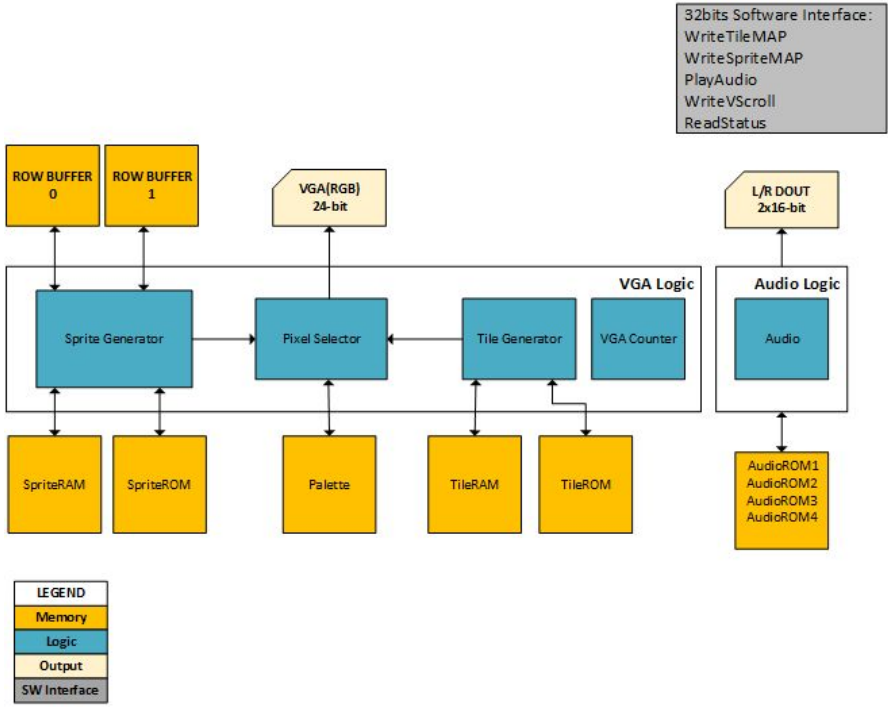
- DE1-SOC board (Cyclone V FPGA + ARM Cortex HPS).
- Resolution: 320 x 240 -> 640 x 480 @60Hz
- RAM
  - Background Tile: Mapping for 20 x 15, plus 1 extra row of hidden tiles (for scrolling)
  - Sprite: 16 total sprites on screen
- ROM
  - 16 x 16 pixel per sprite/tile
  - 6 bit color index for per pixel
  - Both support at most 32 artwork
- # of Colors Support: 4 Color Palette x 64 colors
- Audio
  - Sample Rate: 8 KHz
  - Sample Word: 8 bit
- Original Atari Controller



```
Family : Cyclone V
Device : 5CSEMA5F31C6
Timing Models : Final
Logic utilization (in ALMs) : 767 / 32,070 ( 2 % )
Total registers : 898
Total pins : 362 / 457 ( 79 % )
Total virtual pins : 0
Total block memory bits : 353,104 / 4,065,280 ( 9 % )
Total RAM Blocks : 49 / 397 ( 12 % )
Total DSP Blocks : 0 / 87 ( 0 % )
Total HSSI RX PCSs : 0
Total HSSI PMA RX Deserializers : 0
Total HSSI TX PCSs : 0
Total HSSI PMA TX Serializers : 0
Total PLLs : 1 / 6 ( 17 % )
Total DLLs : 1 / 4 ( 25 % )
```

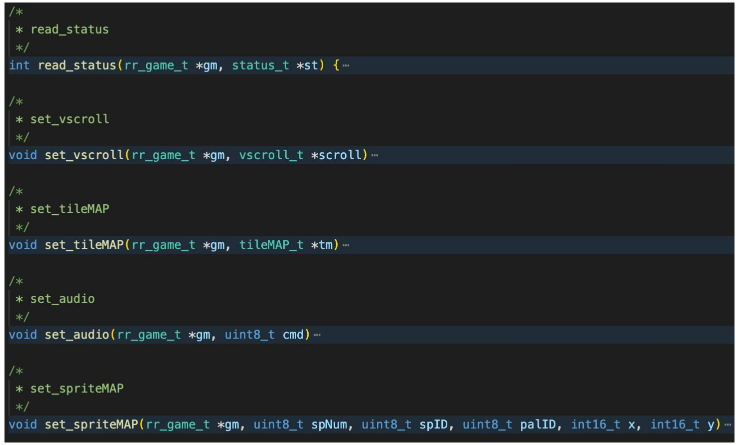
Hardware Overview

- Sprite and Tile artwork stored in Tile and Sprite ROMs
- VGA Counter
- Tile Generator:
  - TileRAM: Tile id, Pal id of tiles to draw
  - [hcount, vcount,Vscroll]->TileRAM_address
  - [TileRAMData, hcoun,vcount,VScroll]->TileROM_address
  - [TileROMData,Palette] -> TilePixelVal
- Sprite Generator:
  - *SpriteRAM: X, Y, Sprite id, Pal id of sprite to draw*
  - ***16 total sprites.***
  - *Sprite order Tile,Sprite_0,Sprite_1..Sprite_15*
  - Sprite Row Buffer (Double buffering)
- Pixel Selector: Color Pallette return the actual ***RGB*** color
- 4x Audio ROMS, 8bit-8Khz Mono, simultaneous playback



Software and Hardware Interface

- **read_status**: reads the joystick status and update frame signal, by ***polling***, controlling the aircraft
- **set_vscroll**: Writes the vertical scroll value
- **set_audio**: Selects the audio sample by ***audio id*** for different events (e.g. Crash, Fire, Fly)
- **set_tileMAP**: Writes the ***tile id*** and ***color palette id*** in Tile RAM at given ***"slot"***
- **set_spriteMAP**: Writes the information of given sprite into the Sprite RAM



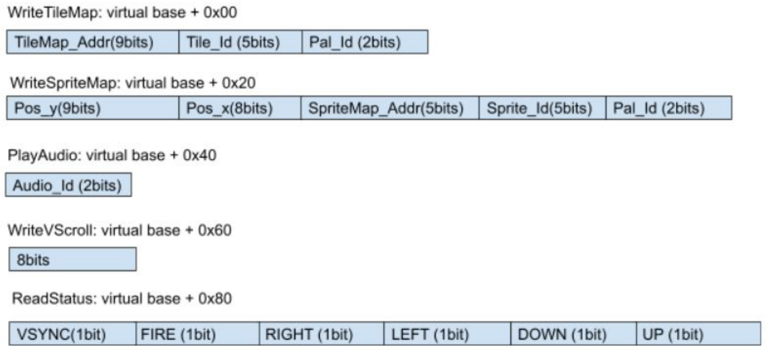
```
/*
 * read_status
 */
int read_status(rr_game_t *gm, status_t *st) {⋯

/*
 * set_vscroll
 */
void set_vscroll(rr_game_t *gm, vscroll_t *scroll)⋯

/*
 * set_tileMAP
 */
void set_tileMAP(rr_game_t *gm, tileMAP_t *tm)⋯

/*
 * set_audio
 */
void set_audio(rr_game_t *gm, uint8_t cmd)⋯

/*
 * set_spriteMAP
 */
void set_spriteMAP(rr_game_t *gm, uint8_t spNum, uint8_t spID, uint8_t palID, int16_t x, int16_t y)⋯
```



WriteTileMap: virtual base + 0x00

TileMap_Addr(9bits)	Tile_Id (5bits)	Pal_Id (2bits)

WriteSpriteMap: virtual base + 0x20

Pos_y(9bits)	Pos_x(8bits)	SpriteMap_Addr(5bits)	Sprite_Id(5bits)	Pal_Id (2bits)

PlayAudio: virtual base + 0x40

Audio_Id (2bits)

WriteVScroll: virtual base + 0x60

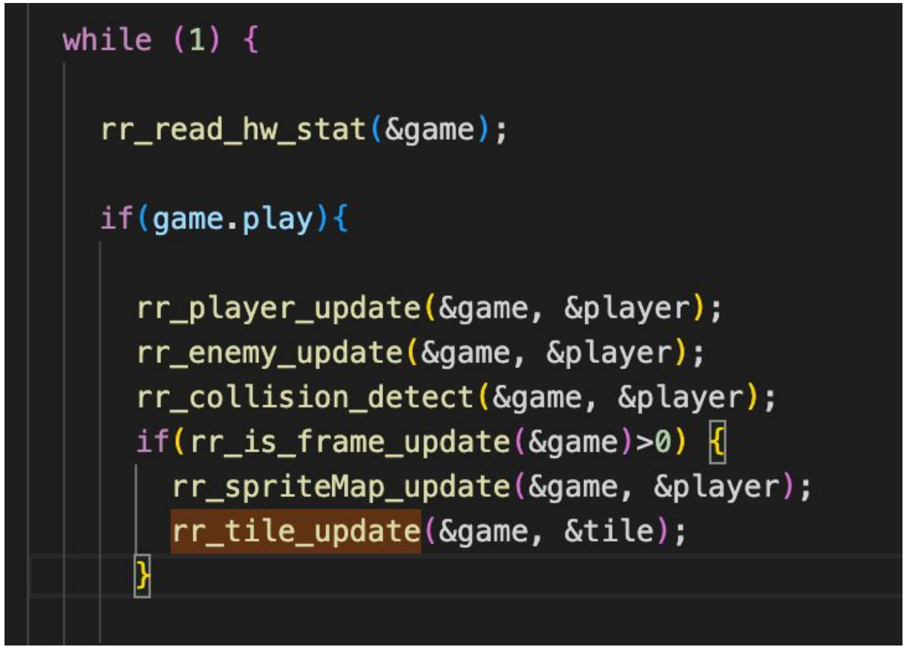
8bits

ReadStatus: virtual base + 0x80

VSYNC(1bit)	FIRE (1bit)	RIGHT (1bit)	LEFT (1bit)	DOWN (1bit)	UP (1bit)

Software (Game Loop)

- **rr_read_hw_stat**: Reads joystick and update_frame flag from the hardware.
- **rr_player_update**: Updates player's position and fuel level.
- **rr_enemy_update**: Creates new enemy ship (total of 5 enemies can exist at a time). Updates enemy position. Sets enemy attack mode.
- **rr_collision_detect**: Collision if two sprite overlap (16 x 16 boundary)
- **rr_spriteMap_update**(): Collect all the updates and update them at once
- **rr_tile_update**(): Increments Vscroll register and writes Tile MAP RAM via tile map RAM register.
    - tiles are reading from a pre-define txt file
-



```
while (1) {

  rr_read_hw_stat(&game);

  if(game.play){

    rr_player_update(&game, &player);
    rr_enemy_update(&game, &player);
    rr_collision_detect(&game, &player);
    if(rr_is_frame_update(&game)>0) {
      rr_spriteMap_update(&game, &player);
      rr_tile_update(&game, &tile);
    }
```

Question:
How often do we update sprite/tiles?

Improvements

- Explore and incorporate other available on chip and on board peripherals. Eg. SDRAM.
- Graphics - higher resolution (Too conservative on the resource budget).
- Audio - improve quality, length, add effects.
- Make some sprite, tile and audio data loadable from software.

Lessons Learned

- Programming game is a never ending task
- Timing required for games
- Importance of testbench
- Make the process as fun as possible

DEMO

Hope you like our adaptation of the classic Atari River Raid game!!