Final Report

Hierarchy

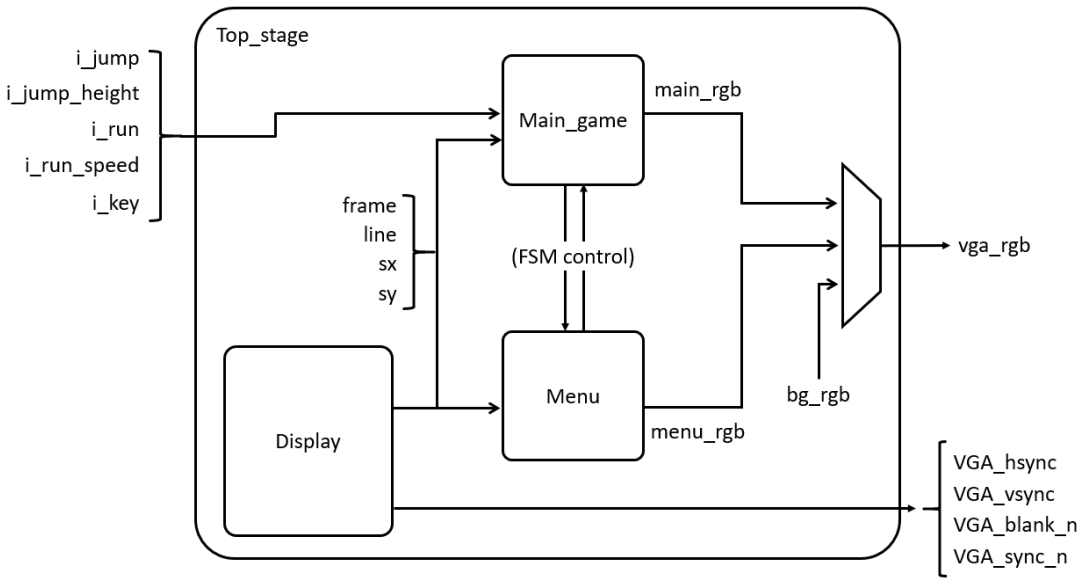
-DE2_115.sv
  -Debounce.sv
  -SevenHexDecoder.sv
  -Altpll.v
  -TopControl.sv
    -I2cInitializer.sv
    -AudRecorder.sv
    -average2048.sv
  -TopStage.sv
    -main_game.sv
    -menu_sprite.sv
    -menu.sv
    -sprite_position.sv
    -moving_sprite.sv
    -sprite.sv
    -ROM_async.sv
    -display.sv

Block Diagram

Top Control

Top Stage

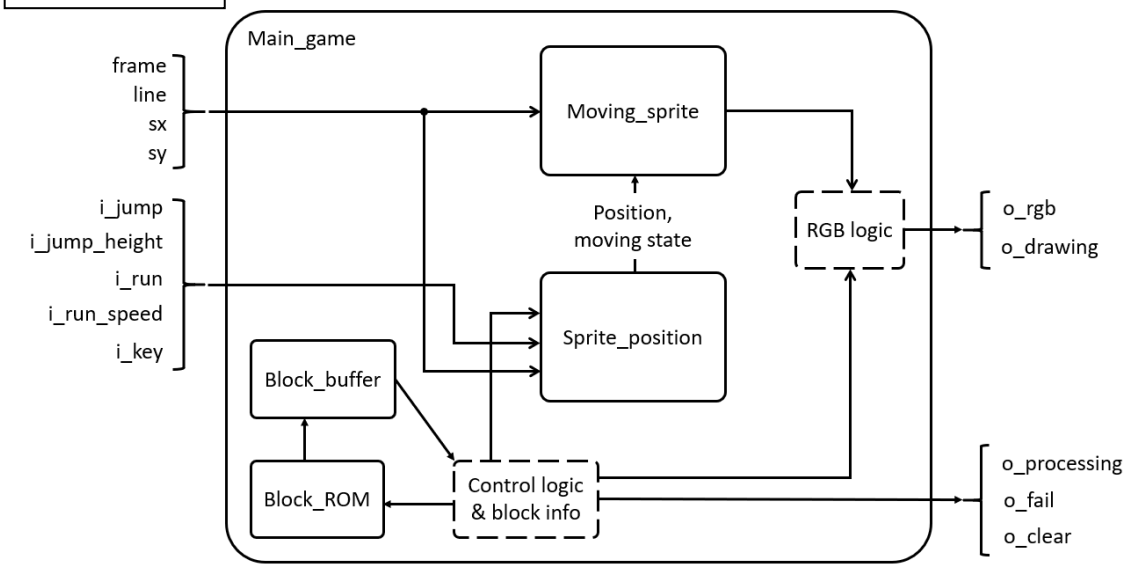
Input control signal and output VGA signal. Handle the main part of the game.



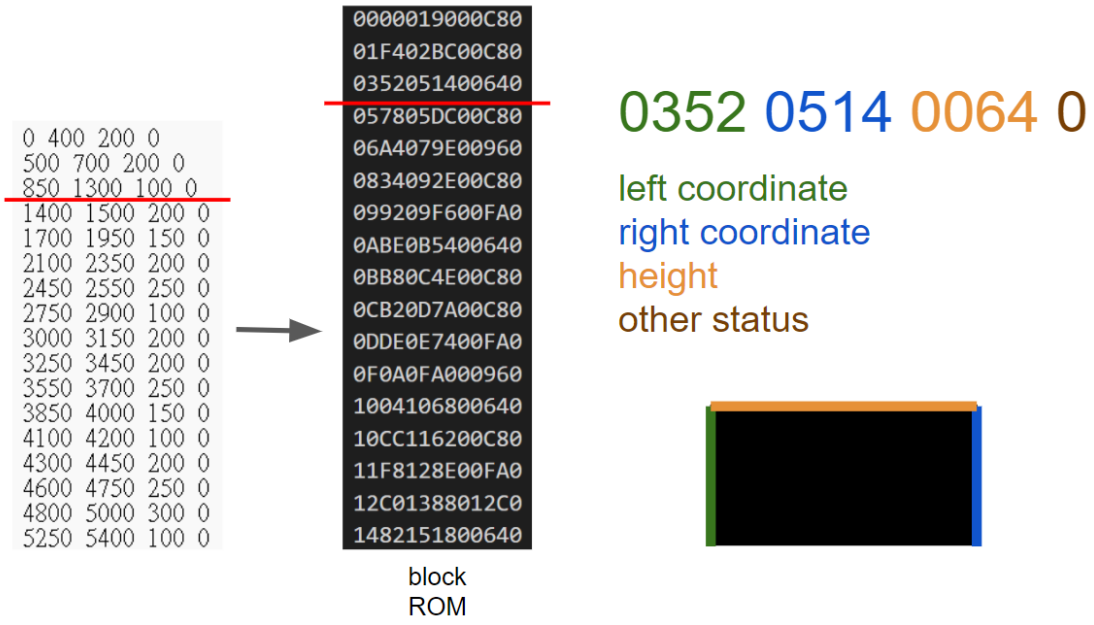
Display module generates control signal for VGA output (800*600 @60Hz, with 40MHz clock), including hsync, vsync for screen synchronize; blank, sync for DAC; frame, line, sx, sy for display logic.

Main_game and **Menu** modules handle the display. They form an FSM to tell in which state the game is currently, control the rgb output. The FSM is simple: the game starts at menu. Push the first button to enter the main game. Return to menu if fall off the block or reach the goal.

Main game



Block ROM is used to store the whole map. The map consists of several blocks, and is represented by their left, right, and top boarders. (see the figure below)



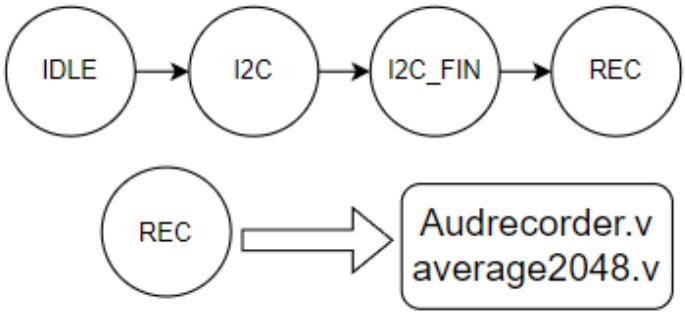
An observation we made here is, to tell where to draw the block we need to check whether the currently drawing pixel is inside the block, which may use lots of logics. So instead of using the whole map at a time, we use a **block buffer** to store a rather few blocks. It loads block information when the game starts or the last block in the buffer shows up on the screen (by players progress).

Sprite position module calculates the coordinates of the character. With the control inputs, it handles the map`s rolling, character`s jumping and collisions with blocks. With an register indicating the current block the character is on, the floor height and the block wall position is used in this calculation.

Moving sprite module displays the main character. The coordinate from the position module and the frame, line signal from display module tells it where to start drawing. A ROM is used to store the character`s moving animations (figure below). The data stored in the sprite file is the index of 256 colors, which are read out to index another ROM storing the color look up table to minimize the storage usage.

FSM

Top Control



Fitter Summary

Fitter Summary	
Fitter Status	Successful - Sun Jan 16 15:23:27 2022
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	DE2_115
Top-level Entity Name	DE2_115
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	2,565 / 114,480 (2 %)
Total combinational functions	2,530 / 114,480 (2 %)
Dedicated logic registers	780 / 114,480 (< 1 %)
Total registers	780
Total pins	480 / 529 (91 %)
Total virtual pins	0
Total memory bits	24,624 / 3,981,312 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	1 / 4 (25 %)

Timing Analyzer

Unconstrained Paths

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	1	1
3	Unconstrained Input Ports	4	4
4	Unconstrained Input Port Paths	747	747
5	Unconstrained Output Ports	51	51
6	Unconstrained Output Port Paths	111	111

遇到的問題與解決辦法

1. FFT We initially attempt to use frequency to decide whether the character should jump and the height that the character should jump. Unfortunately, we fail to reach an effective algorithm and parameters to control the character. In our code, we have four parameters:
   (a) the number of samples going through FFT together which related to the precision of frequency detection
   (b) the threshold of noise filter which related to whether a frequency should view as a noise
   (c) the threshold which we justify the signal to be over the jumping threshold, and
   (d) the number of clocks we should accumulate the data from DSP before we output a 'jump signal'.
   The main jeopardy we face is that the algorithm we use to output a 'jump signal' is: if the frequency with the biggest magnitude is greater than parameter (c) continuously over parameter (d) clocks, than we will pull up the 'jump signal' for one clock, and then simultaneously output the 'jump height' which represents the biggest frequency accumulated in parameter (d) clocks. Nevertheless, the signal is not stable enough, so it is hard to balance between parameter (b) and (d), which may cause false activate and inactivate