The Design Document for CSEE 4840 Embedded System Design

2048 Game

Jingtian Lin (jl6589)

Kanghui Lin (kl3521)

Yunhao Xing (yx2812)

Contents

2. System Block Diagram

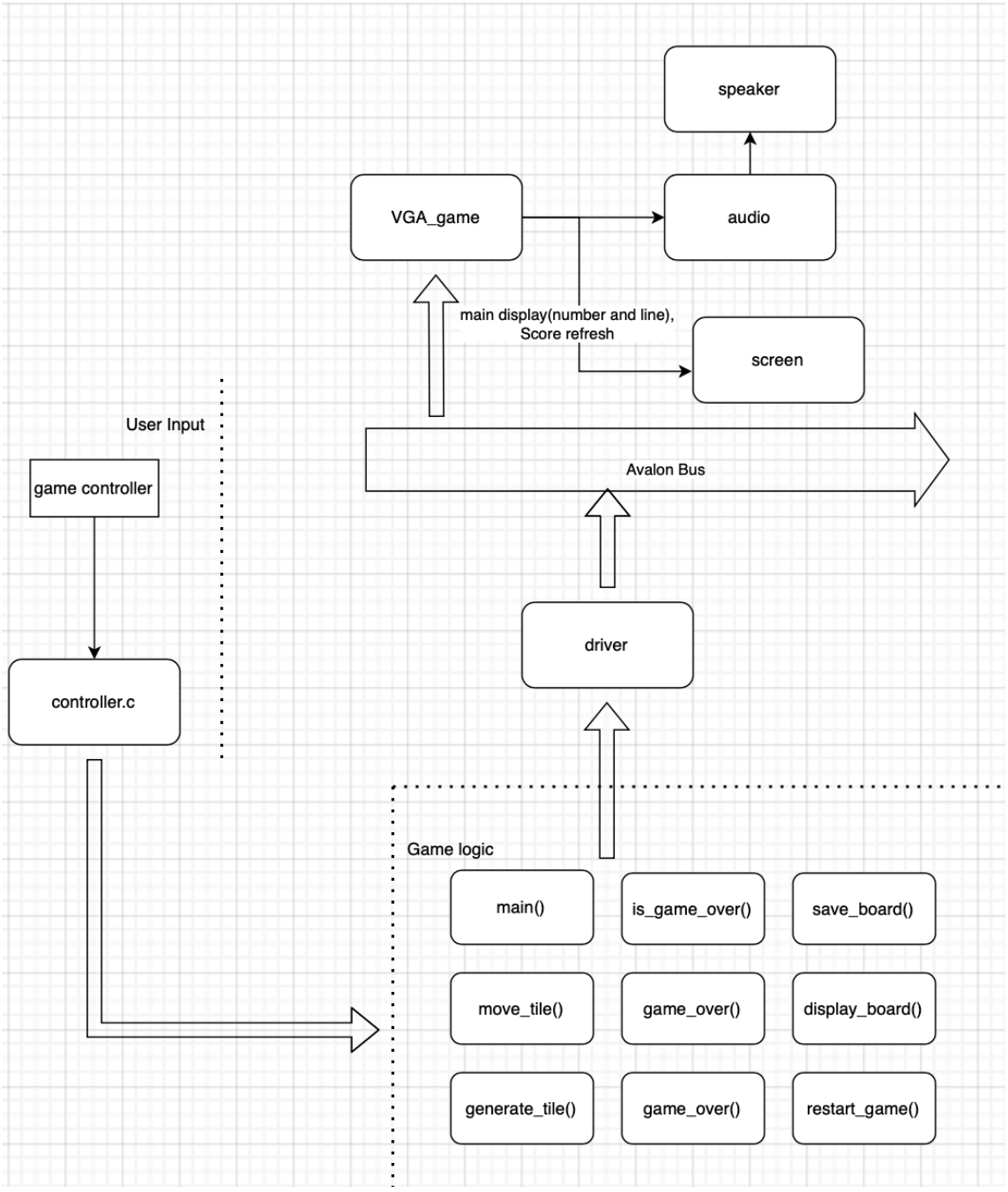

Figure 1: System diagram of the 2048

3.Algorithm

To implement the game mechanism we will be implement the following functions:

- move_tile(dir): moving the tiles in the specified directions upon detecting inputs from the joystick; merging tiles on the direction specified, and compute the new board layout; then display the new board after tile move on the display using display_board();
- generate_tile(): void function, randomly generate a new number on empty position of the board; set 75% of chance to generate a 2 and 25% of chance to generate a 4
- is_game_over(): return a boolean checking if there are still mergeable tiles on board
- game_over(): void function, display the current score, update the scoreboard and restart game
- save_board(): void function, store the current board layout in state for future use (pause and resume the game)
- display_board(): void function, compute the stored board layout to the display (default empty board)
- restart_game(): void function, empty the board and the score

We will keep track of the following states to implement the game mechanism:

Board_state - the board_state is a length 16 int array. The ith place in the array symbolizes the current values on the ith position of the board

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Tile number of 2048 board

Upon moving command in each direction dir we detect if the numbers on the columns (for move up and down) and rows (for move left and right) can be merged and update the board_state accordingly. Then generate the new random tile to the board_state and check if the game is over; if the game continues, display the new board, otherwise end the game with current score

Current_score - current score stores the current score of the play, upon each player move, compute the new score based on the values on the tiles merged; the score will be computed as the sum of the values on tiles merged square to reward larger number tile merge. Eg. if a player merged two 16 tiles, and four 8 tiles upon single move, then his score gain for the move will be 16^2 + 2*8^2 = 384.

Score_board - scoreboard keep tracks highest 5 scores player gets from all plays

4. Resource Budgets

4.1. Video

For this part, we will use some tricks to reduce the size of our images. For tiles, we will use 2 bits to represent each pixel like 2'b00 means it's transparent so we can implement the rounded rectangle. For the background grid, we will use a smaller image than its actual size (1/4), and since it will use just 2 different colors, we use 1 bit to represent each pixel.

Category	Size	# of images	Total size (bits)
Tiles	100 * 100	14	280,000
Background	480 * 480	1	57,600
Logo	100 * 100	1	20,000
Number	20 * 20	10	4,000
Score	100 * 20	1	2,000
Best	100 * 20	1	2,000
Memory Budget (bits)			365,600

4.2. Audio

	BGM	Tile move	Tile merge	Game over
Time (s)	5	0.25	0.25	1
F_s (kHZ)	8	8	8	8
Memory	655,360	32,768	32,768	131,072
Memory Budget (bits)				851,968

4.3. Total bits

The total memory budget adds up to 1,217,568 bits.

5. Hardware/Software Interface

Register 1-16 (50 bits): Tiles. 10 bits for x position, 9 bits for y position, 24 bits for colors, 3 bits for size, 4 bits for type.

Register 17 (7 bits): Current score. 3 bits for radical, 4 bits for digit.

Register 18 (7 bits): Best score. 3 bits for radical, 4 bits for digit.

Register 19 (4 bits): Game Status & Control 2 bits for game status, 2 bits for audio to play