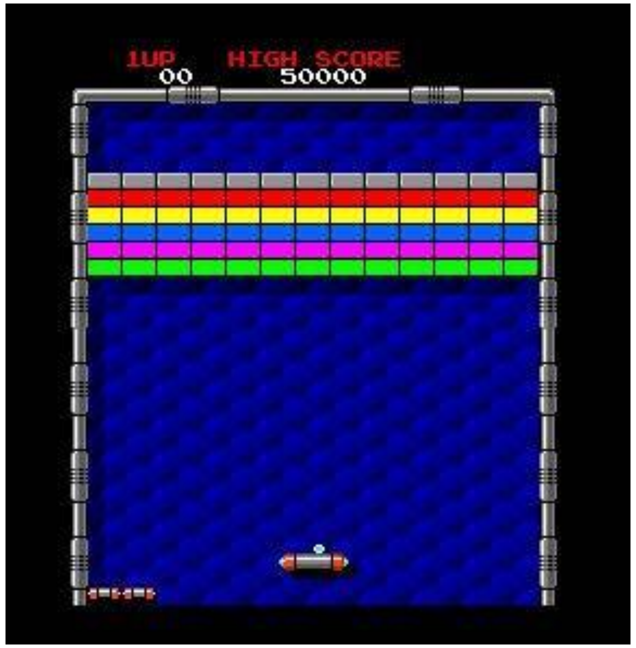
Breakout Game Remastered

Youfeng Chen (yc3999)
Angzi Xu (ax2157)
Wang Chen (wc2794)
Zheyuan Song (zs2527)

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

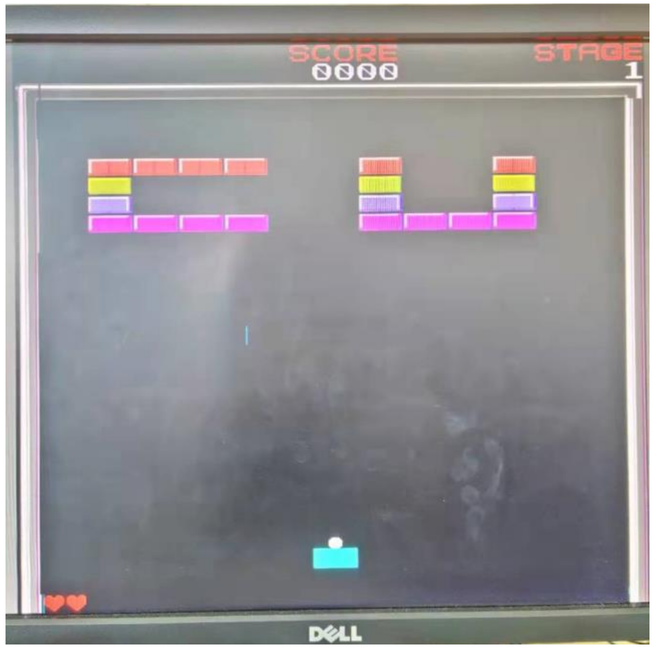
Overview

- Rebuild the classic Breakout game.
- Destroy all bricks with the ball to win the game.
- If the player failed to catch the ball for a total of three time in one stage, player will lose the game.

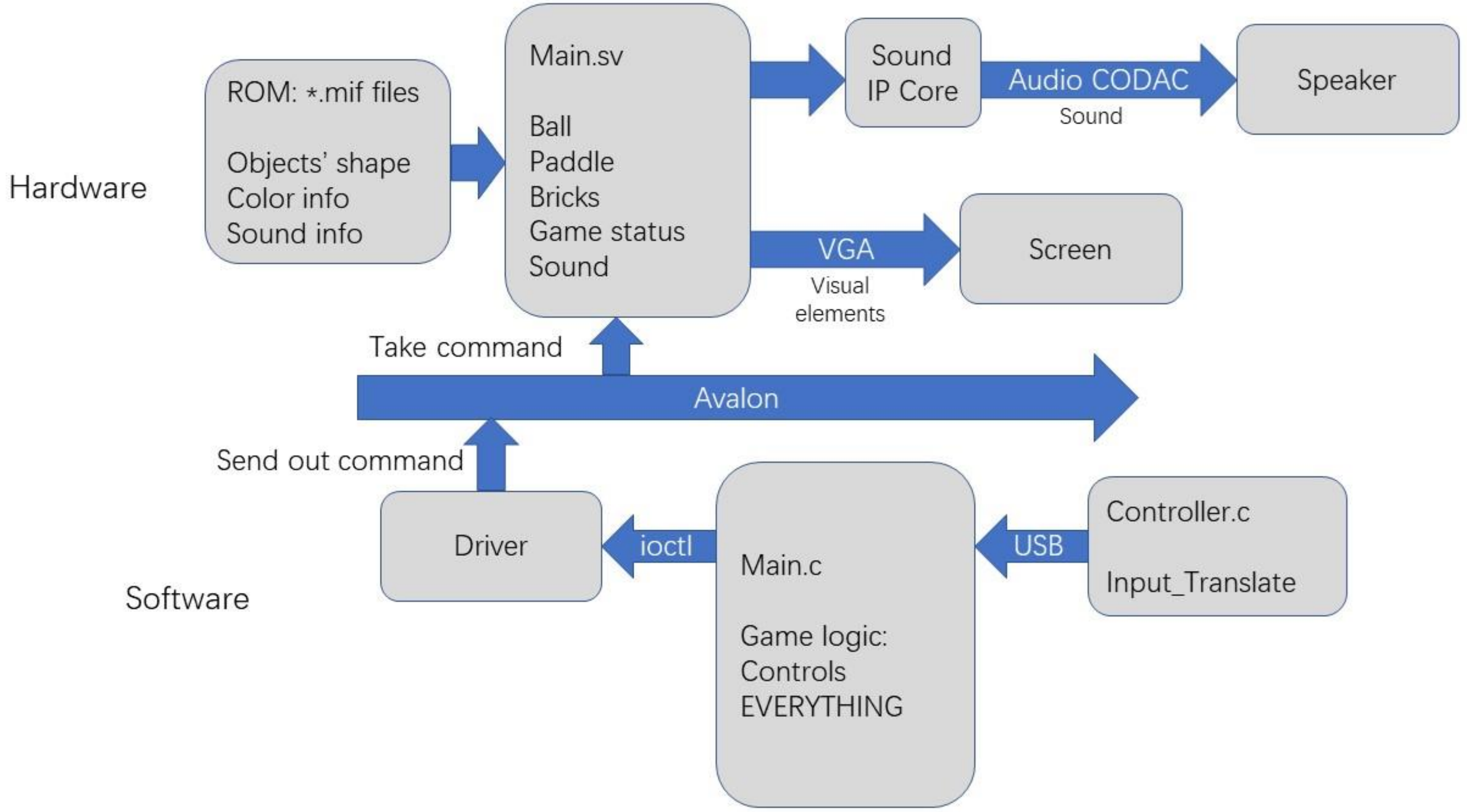


←original

ours→



Top-level Architecture



HW Design - Graphics

Tiles and sprites

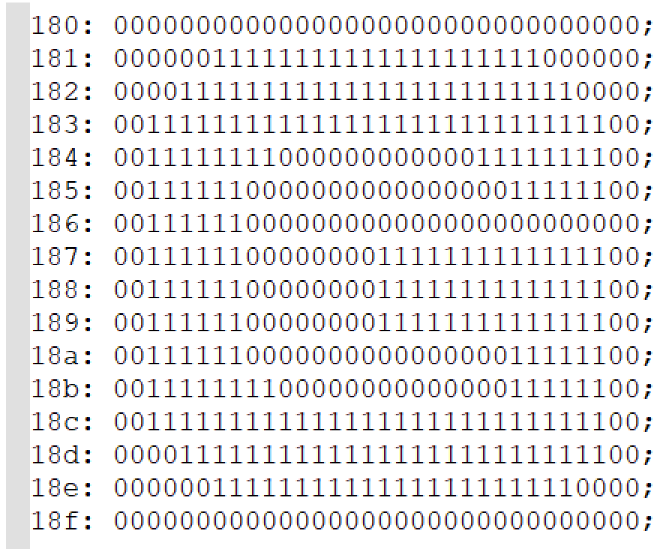
- Tiles: pre-made graphical materials - assign to certain locations on the screen in case of need.

- Sprites: place things in layers - ensure proper overlay.

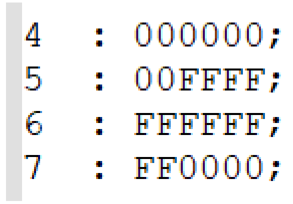
HW Design - Graphics

Tile example:

- Letter G, which utilizes two colors: “11” and “00”.
- 00 means the first color (#000000) and 11 means the forth (#FF0000).
- Each tile can use up to 4 colors.



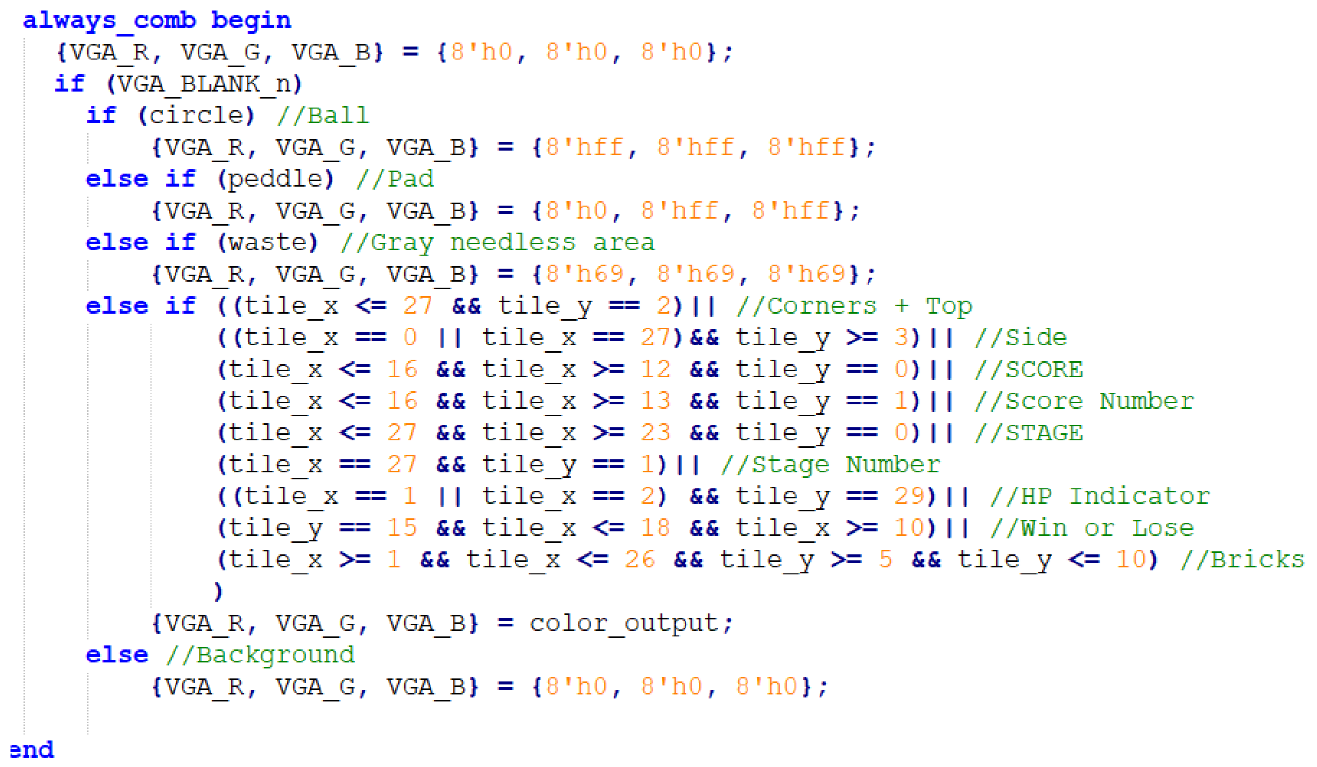
```
180: 00000000000000000000000000000000;
181: 00000011111111111111111111000000;
182: 00001111111111111111111111110000;
183: 00111111111111111111111111111100;
184: 00111111110000000000001111111100;
185: 00111111000000000000000011111100;
186: 00111111000000000000000000000000;
187: 00111111000000001111111111111100;
188: 00111111000000001111111111111100;
189: 00111111000000001111111111111100;
18a: 00111111000000000000000011111100;
18b: 00111111110000000000000011111100;
18c: 00111111111111111111111111111100;
18d: 00001111111111111111111111111100;
18e: 00000011111111111111111111110000;
18f: 00000000000000000000000000000000;
```



```
4   : 000000;
5   : 00FFFF;
6   : FFFFFF;
7   : FF0000;
```

HW Design - Graphics

Sprite:



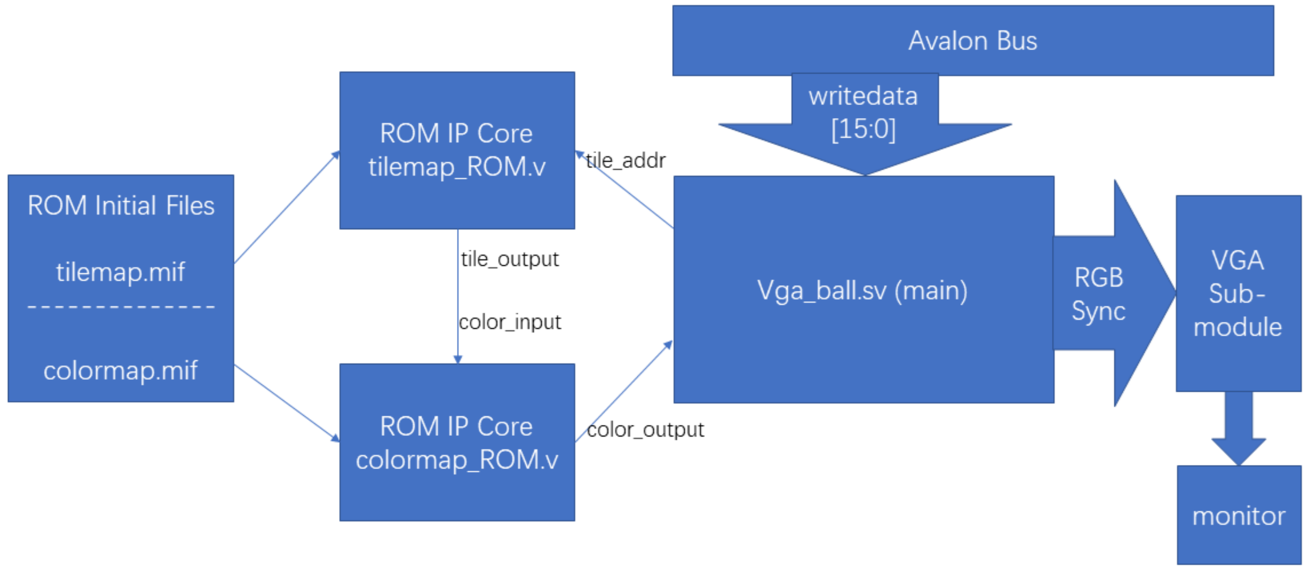
```
always_comb begin
  {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};
  if (VGA_BLANK_n)
    if (circle) //Ball
        {VGA_R, VGA_G, VGA_B} = {8'hff, 8'hff, 8'hff};
    else if (peddle) //Pad
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'hff, 8'hff};
    else if (waste) //Gray needless area
        {VGA_R, VGA_G, VGA_B} = {8'h69, 8'h69, 8'h69};
    else if ((tile_x <= 27 && tile_y == 2)|| //Corners + Top
            ((tile_x == 0 || tile_x == 27)&& tile_y >= 3)|| //Side
            (tile_x <= 16 && tile_x >= 12 && tile_y == 0)|| //SCORE
            (tile_x <= 16 && tile_x >= 13 && tile_y == 1)|| //Score Number
            (tile_x <= 27 && tile_x >= 23 && tile_y == 0)|| //STAGE
            (tile_x == 27 && tile_y == 1)|| //Stage Number
            ((tile_x == 1 || tile_x == 2) && tile_y == 29)|| //HP Indicator
            (tile_y == 15 && tile_x <= 18 && tile_x >= 10)|| //Win or Lose
            (tile_x >= 1 && tile_x <= 26 && tile_y >= 5 && tile_y <= 10) //Bricks
            )
        {VGA_R, VGA_G, VGA_B} = color_output;
    else //Background
        {VGA_R, VGA_G, VGA_B} = {8'h0, 8'h0, 8'h0};

end
```

Upper elements will display on top of lower elements in case of conflict

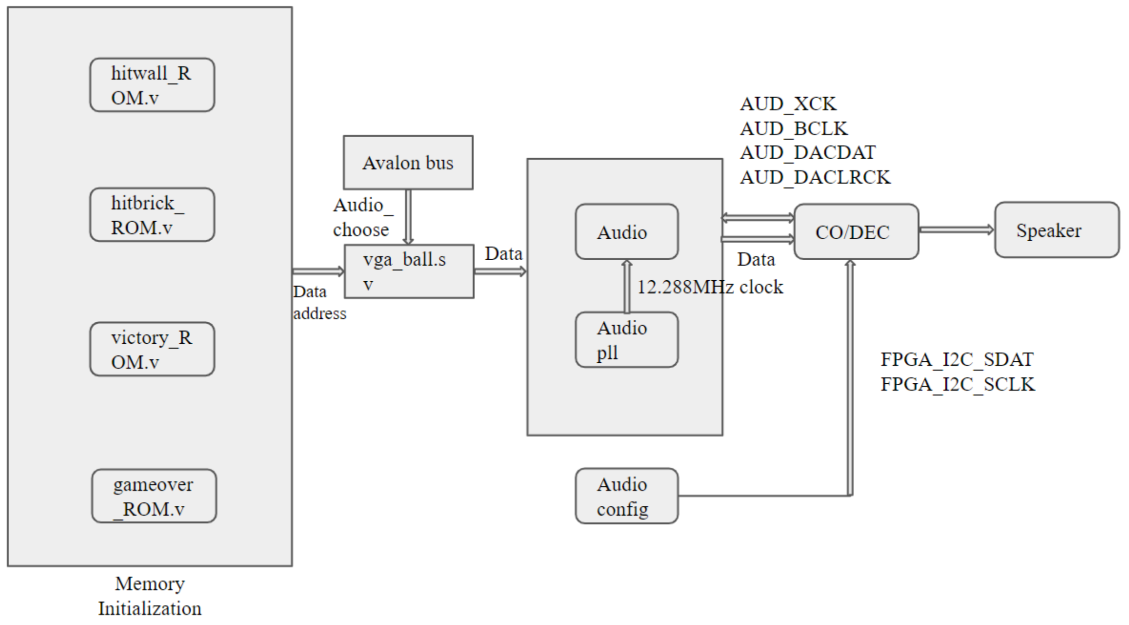
HW Design - Graphics

Block Diagram:



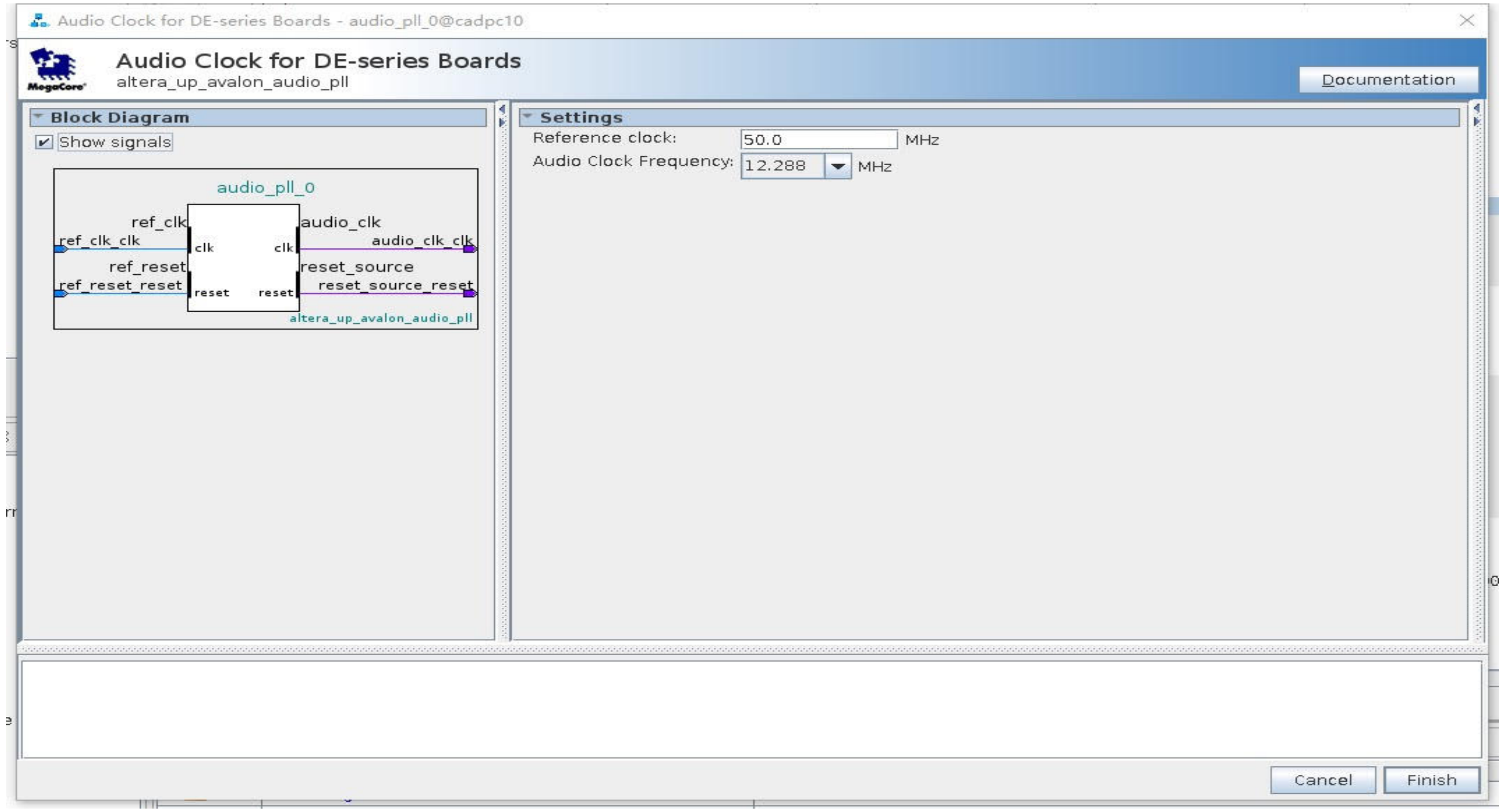
HW Design - Sound

Block Diagram:



HW Design - Sound

- Audio_ppl_0 configuration:


Audio Clock for DE-series Boards - audio_pll_0@cadpc10

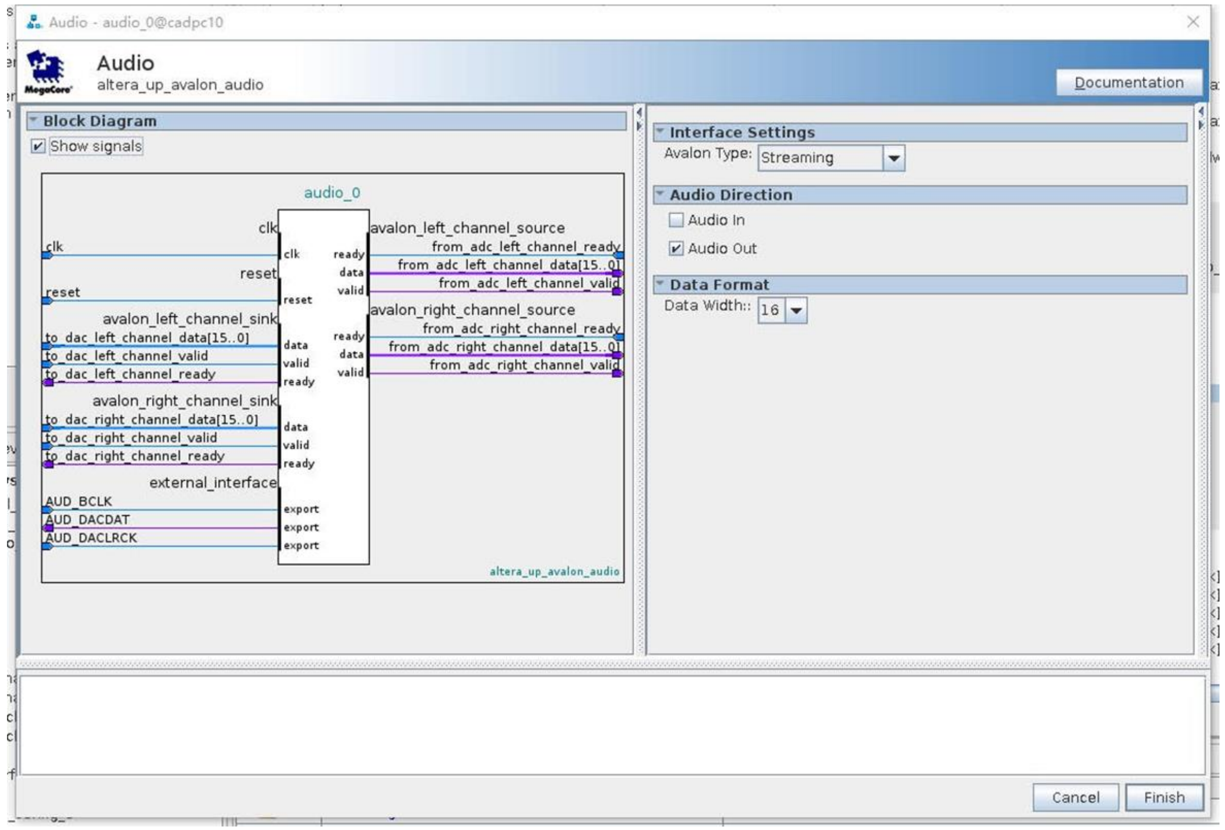
Audio Clock for DE-series Boards
altera_up_avalon_audio_pll

Block Diagram

Settings
Reference clock: 50.0 MHz
Audio Clock Frequency: 12.288 MHz

HW Design - Sound

- Audio IP Core:


Audio - audio_0@cadpc10
MegaCore
Documentation
Block Diagram
audio_0
clk
reset
avalon_left_channel_sink
to_dac_left_channel_data[15..0]
to_dac_left_channel_valid
to_dac_left_channel_ready
avalon_right_channel_sink
to_dac_right_channel_data[15..0]
to_dac_right_channel_valid
to_dac_right_channel_ready
external_interface
AUD_BCLK
AUD_DACDAT
AUD_DACLRCK
data
valid
ready
export
avalon_left_channel_source
from_adc_left_channel_ready
from_adc_left_channel_data[15..0]
from_adc_left_channel_valid
avalon_right_channel_source
from_adc_right_channel_ready
from_adc_right_channel_data[15..0]
from_adc_right_channel_valid
altera_up_avalon_audio
Cancel
Finish

Audio
altera_up_avalon_audio

[✓] Show signals

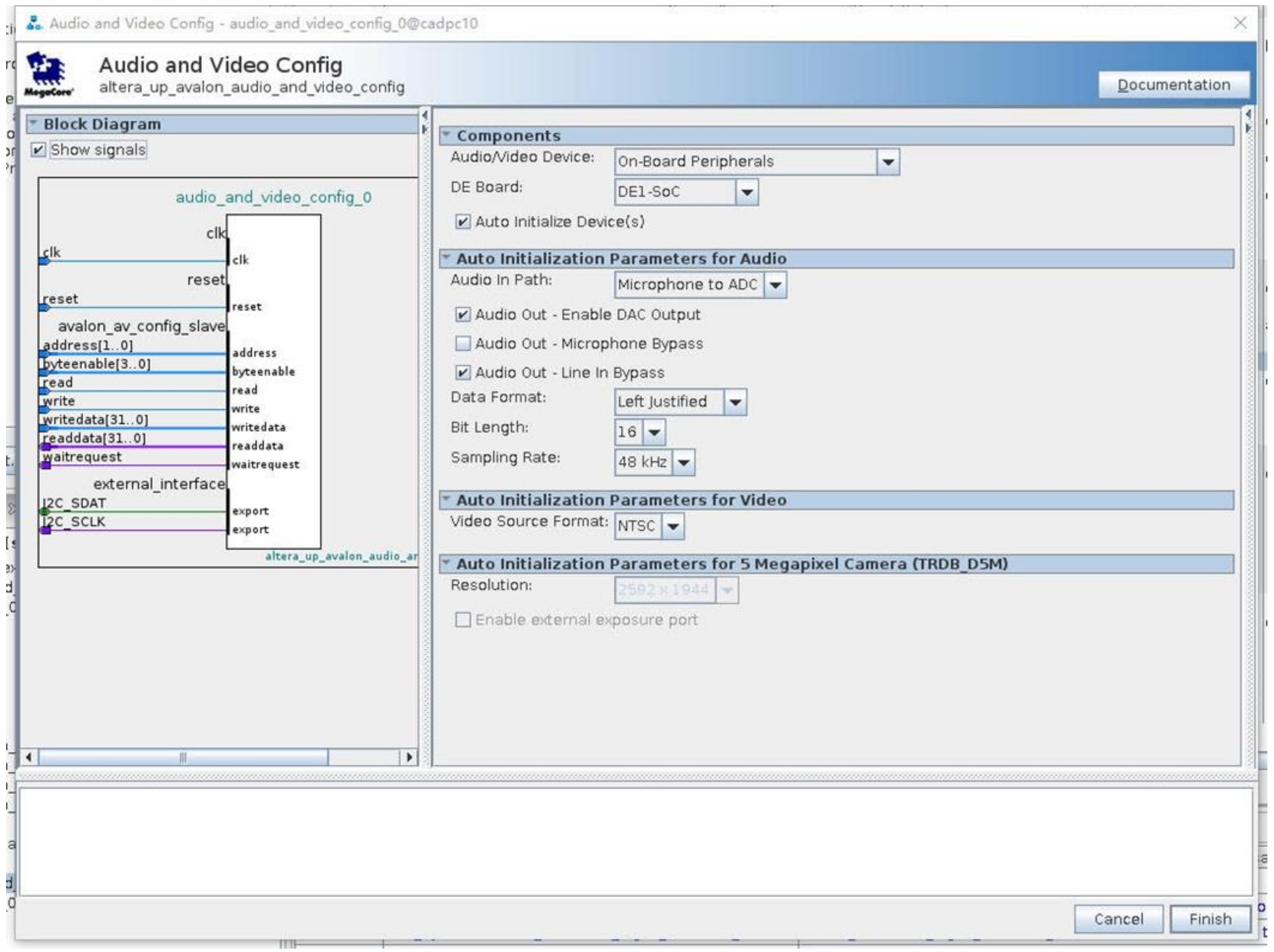
Interface Settings
Avalon Type: Streaming

Audio Direction
[ ] Audio In
[✓] Audio Out

Data Format
Data Width:: 16

HW Design - Sound

- Audio and Video Config.



Audio and Video Config
altera_up_avalon_audio_and_video_config

Components

Audio/Video Device: On-Board Peripherals
DE Board: DE1-SoC
[✓] Auto Initialize Device(s)

Auto Initialization Parameters for Audio

Audio In Path: Microphone to ADC
[✓] Audio Out - Enable DAC Output
[ ] Audio Out - Microphone Bypass
[✓] Audio Out - Line In Bypass
Data Format: Left Justified
Bit Length: 16
Sampling Rate: 48 kHz

Auto Initialization Parameters for Video

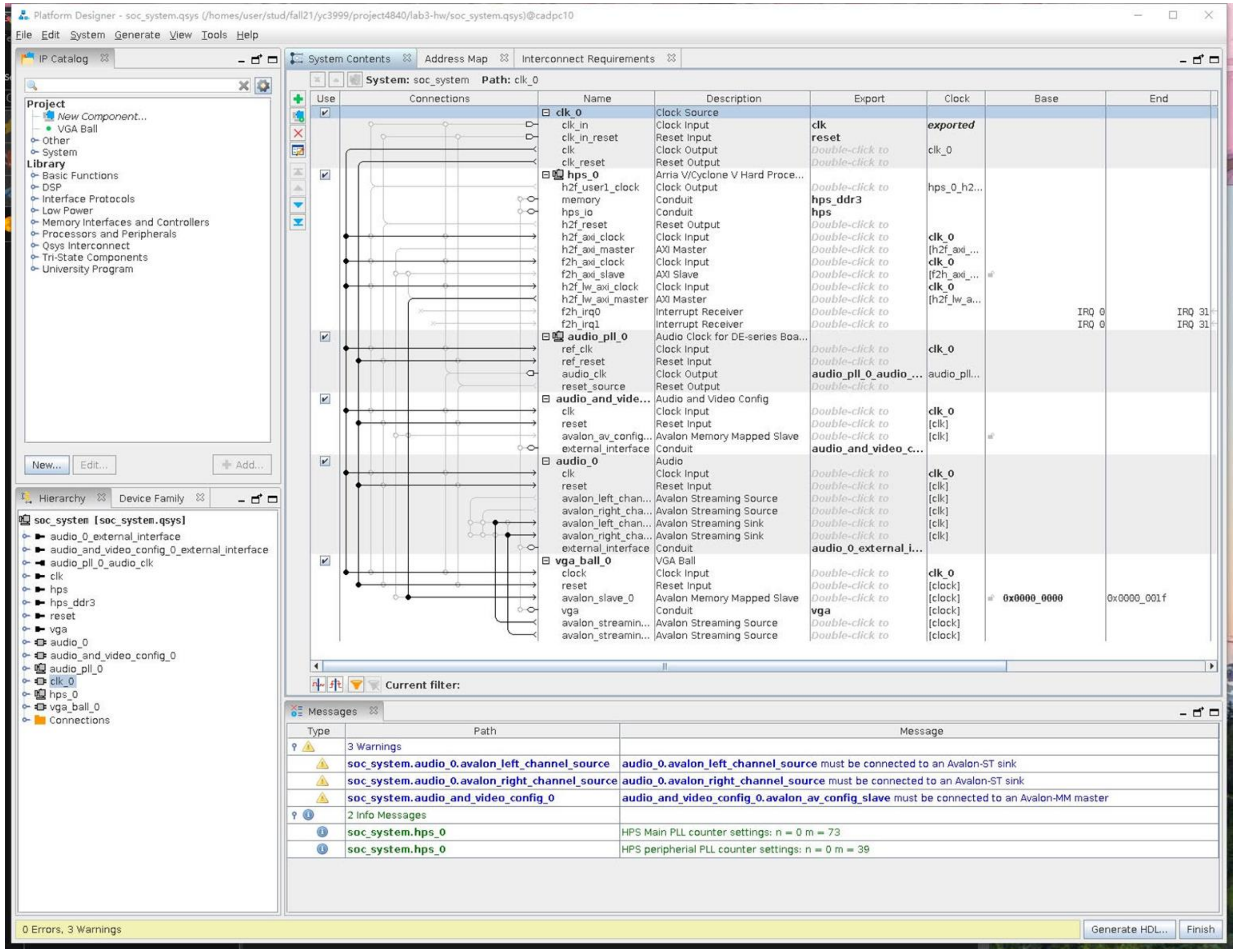
Video Source Format: NTSC

Auto Initialization Parameters for 5 Megapixel Camera (TRDB_D5M)

Resolution: 2592 x 1944
[ ] Enable external exposure port

HW Design - Sound

- Final Qsys connections



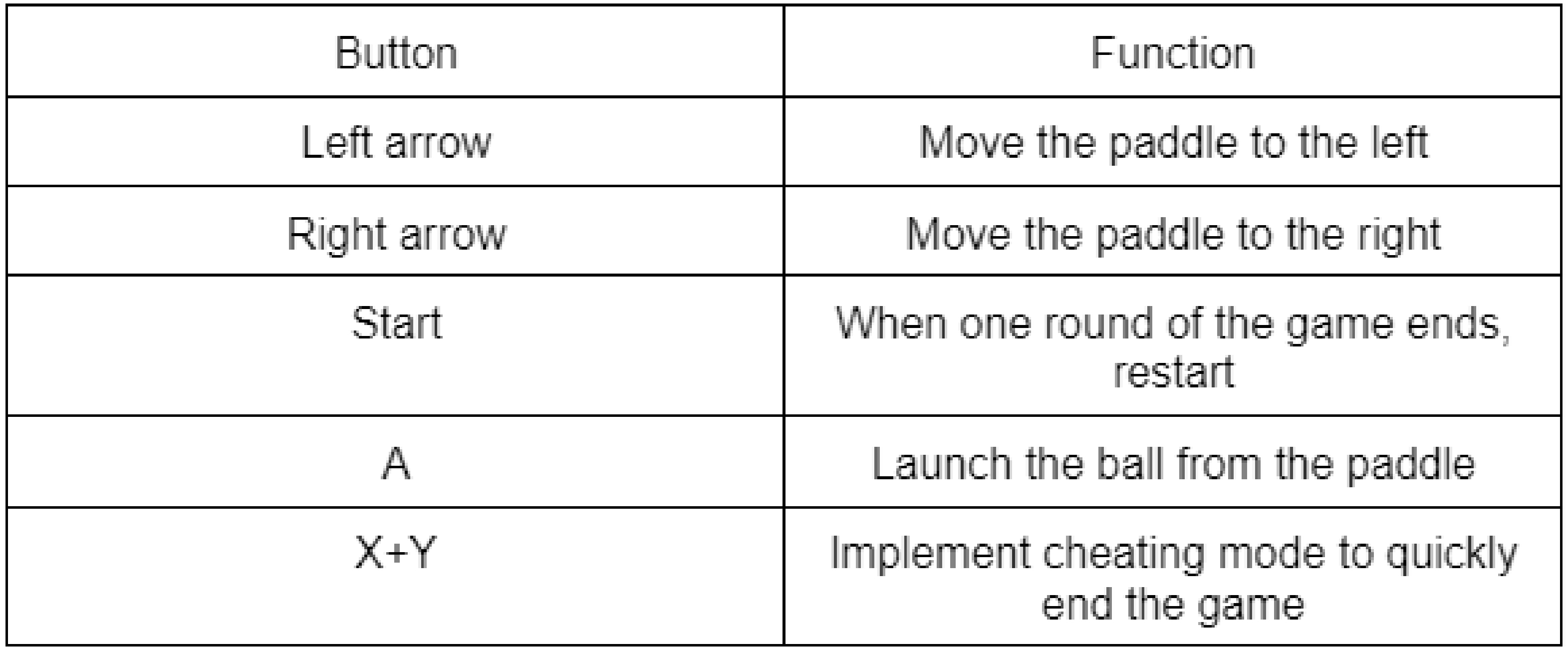
System: soc_system **Path:** clk_0

Use	Connections	Name	Description	Export	Clock	Base	End
✓		⊟ clk_0	Clock Source				
		clk_in	Clock Input	clk	exported		
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	Double-click to	clk_0		
		clk_reset	Reset Output	Double-click to			
✓		⊟ hps_0	Arria V/Cyclone V Hard Proce...				
		h2f_user1_clock	Clock Output	Double-click to	hps_0_h2...		
		memory	Conduit	hps_ddr3			
		hps_io	Conduit	hps			
		h2f_reset	Reset Output	Double-click to			
		h2f_axi_clock	Clock Input	Double-click to	clk_0		
		h2f_axi_master	AXI Master	Double-click to	[h2f_axi_...		
		f2h_axi_clock	Clock Input	Double-click to	clk_0		
		f2h_axi_slave	AXI Slave	Double-click to	[f2h_axi_...		
		h2f_lw_axi_clock	Clock Input	Double-click to	clk_0		
		h2f_lw_axi_master	AXI Master	Double-click to	[h2f_lw_a...		
		f2h_irq0	Interrupt Receiver	Double-click to		IRQ 0	IRQ 31
		f2h_irq1	Interrupt Receiver	Double-click to		IRQ 0	IRQ 31
✓		⊟ audio_pll_0	Audio Clock for DE-series Boa...				
		ref_clk	Clock Input	Double-click to	clk_0		
		ref_reset	Reset Input	Double-click to			
		audio_clk	Clock Output	audio_pll_0_audio_...	audio_pll...		
		reset_source	Reset Output	Double-click to			
✓		⊟ audio_and_vide...	Audio and Video Config				
		clk	Clock Input	Double-click to	clk_0		
		reset	Reset Input	Double-click to	[clk]		
		avalon_av_config...	Avalon Memory Mapped Slave	Double-click to	[clk]		
		external_interface	Conduit	audio_and_video_c...			
✓		⊟ audio_0	Audio				
		clk	Clock Input	Double-click to	clk_0		
		reset	Reset Input	Double-click to	[clk]		
		avalon_left_chan...	Avalon Streaming Source	Double-click to	[clk]		
		avalon_right_cha...	Avalon Streaming Source	Double-click to	[clk]		
		avalon_left_chan...	Avalon Streaming Sink	Double-click to	[clk]		
		avalon_right_cha...	Avalon Streaming Sink	Double-click to	[clk]		
		external_interface	Conduit	audio_0_external_i...			
✓		⊟ vga_ball_0	VGA Ball				
		clock	Clock Input	Double-click to	clk_0		
		reset	Reset Input	Double-click to	[clock]		
		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to	[clock]	0x0000_0000	0x0000_001f
		vga	Conduit	vga	[clock]		
		avalon_streamin...	Avalon Streaming Source	Double-click to	[clock]		
		avalon_streamin...	Avalon Streaming Source	Double-click to	[clock]		

Type	Path	Message
⚠	3 Warnings	
⚠	soc_system.audio_0.avalon_left_channel_source	audio_0.avalon_left_channel_source must be connected to an Avalon-ST sink
⚠	soc_system.audio_0.avalon_right_channel_source	audio_0.avalon_right_channel_source must be connected to an Avalon-ST sink
⚠	soc_system.audio_and_video_config_0	audio_and_video_config_0.avalon_av_config_slave must be connected to an Avalon-MM master
ⓘ	2 Info Messages	
ⓘ	soc_system.hps_0	HPS Main PLL counter settings: n = 0 m = 73
ⓘ	soc_system.hps_0	HPS peripherial PLL counter settings: n = 0 m = 39

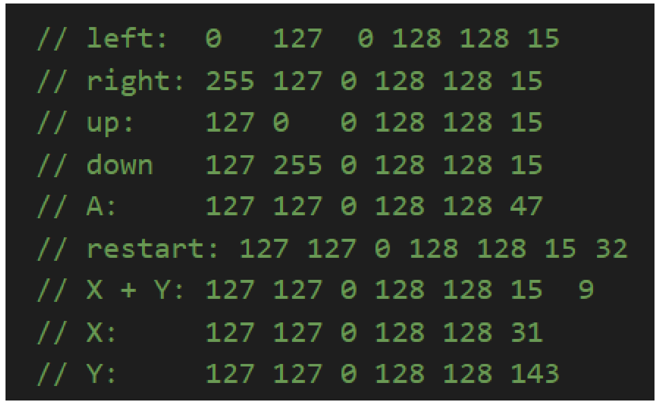
SW Design - Input

Buttons’ Function of the controller



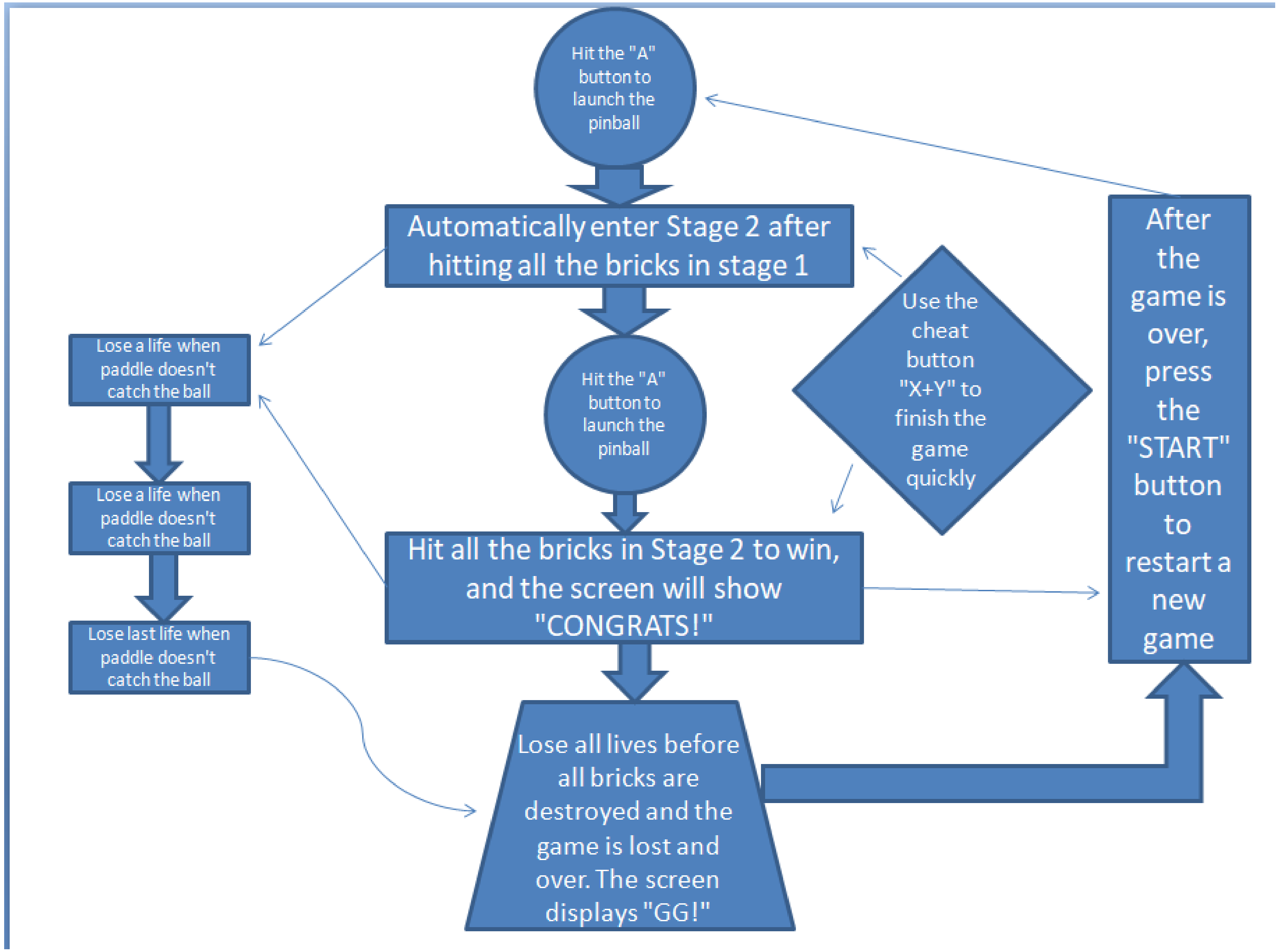
Button	Function
Left arrow	Move the paddle to the left
Right arrow	Move the paddle to the right
Start	When one round of the game ends, restart
A	Launch the ball from the paddle
X+Y	Implement cheating mode to quickly end the game

Data received for each key press:



```
// left:  0   127  0 128 128 15
// right: 255 127 0 128 128 15
// up:    127 0   0 128 128 15
// down   127 255 0 128 128 15
// A:     127 127 0 128 128 47
// restart: 127 127 0 128 128 15 32
// X + Y: 127 127 0 128 128 15  9
// X:     127 127 0 128 128 31
// Y:     127 127 0 128 128 143
```

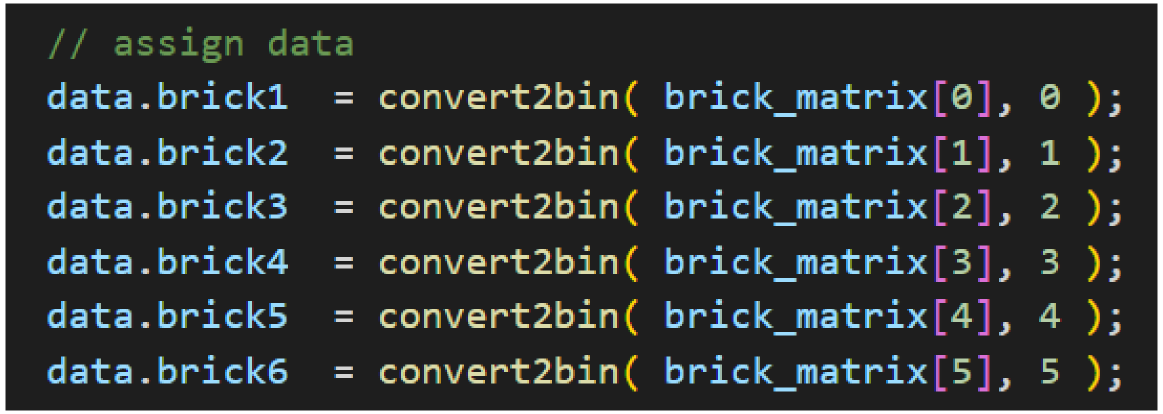
SW Design: Game Logic


Hit the "A" button to launch the pinball
Automatically enter Stage 2 after hitting all the bricks in stage 1
Hit the "A" button to launch the pinball
Hit all the bricks in Stage 2 to win, and the screen will show "CONGRATS!"
Lose all lives before all bricks are destroyed and the game is lost and over. The screen displays "GG!"
Use the cheat button "X+Y" to finish the game quickly
After the game is over, press the "START" button to restart a new game
Lose a life when paddle doesn't catch the ball
Lose a life when paddle doesn't catch the ball
Lose last life when paddle doesn't catch the ball

SW Design - Game Logic

Initialization

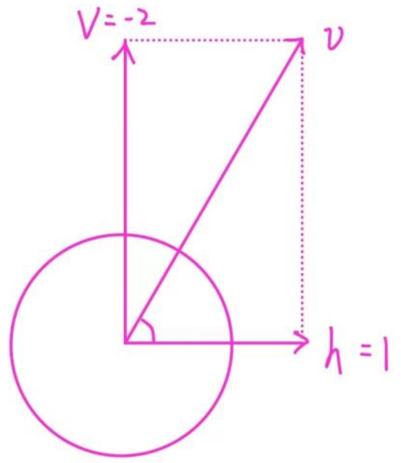
ball_h = 208; & ball_v = 425; data.x_pad = 208;
Assign brick's data:



```
// assign data
data.brick1  = convert2bin( brick_matrix[0], 0 );
data.brick2  = convert2bin( brick_matrix[1], 1 );
data.brick3  = convert2bin( brick_matrix[2], 2 );
data.brick4  = convert2bin( brick_matrix[3], 3 );
data.brick5  = convert2bin( brick_matrix[4], 4 );
data.brick6  = convert2bin( brick_matrix[5], 5 );
```

Movement logic

The moving vector of the ball is a compose of its horizontal and vertical velocity



SW Design - Hit Logic

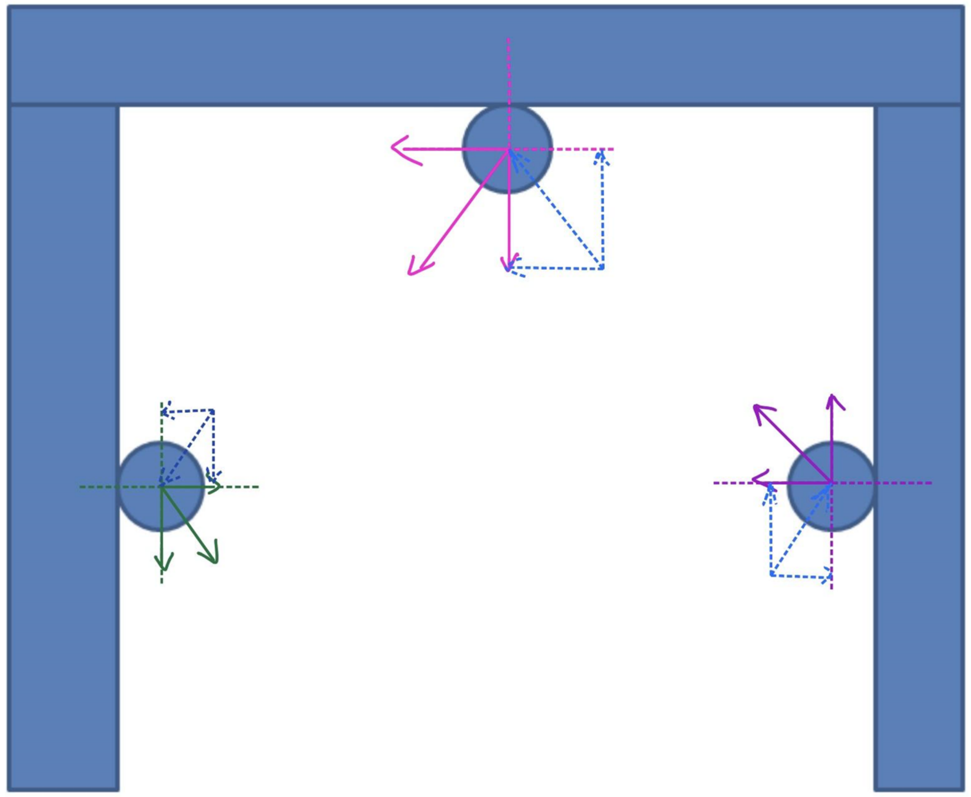
Wall:

top wall y coordinate: 53
right wall x coordinate: 411
left wall x coordinate: 5

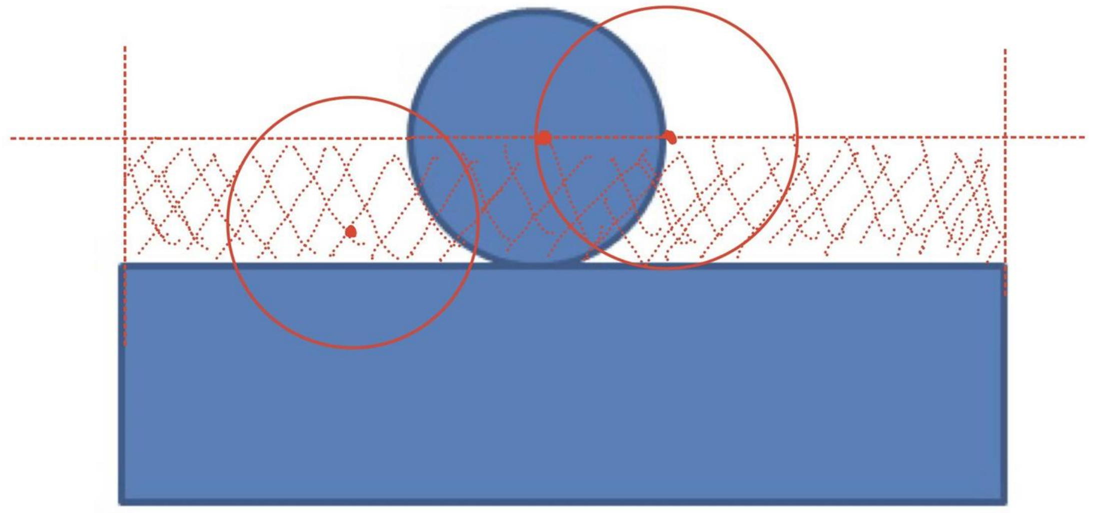
1. Hits the right wall:
   - Horizontal movement: reversed
   - Vertical movement: unchanged

2. Hits the top wall:
   - Horizontal movement: unchanged
   - Vertical movement: reversed

3. Hits the left wall:
   - Horizontal movement: reversed
   - Vertical movement: unchanged

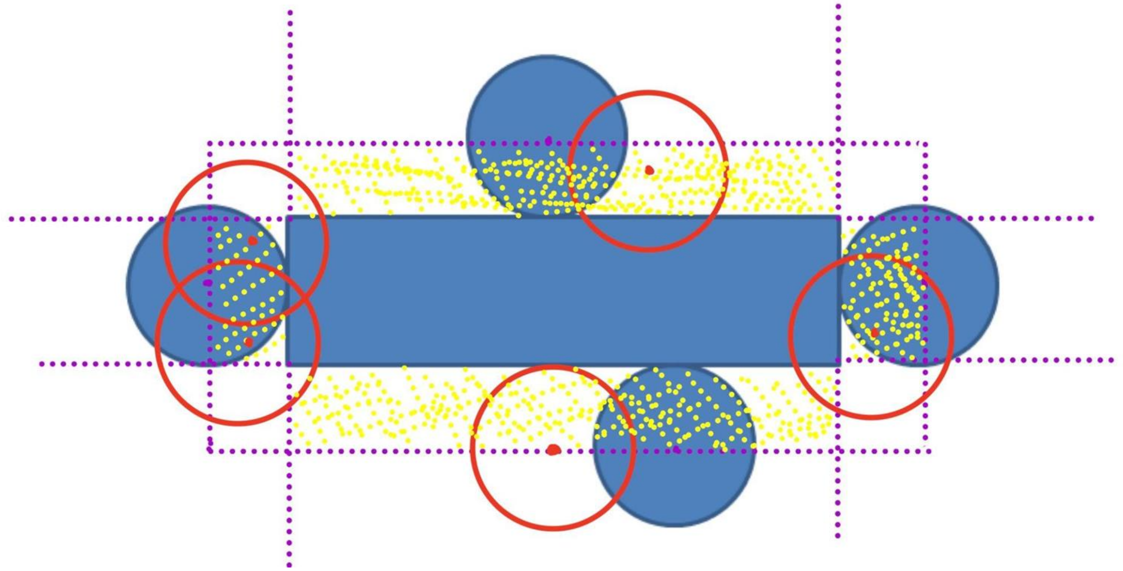


SW Design - Hit Logic



Paddle “hitbox”:

Rectangle zone on the paddle.
L = length of the paddle
H = radius of the ball



Brick “hitbox”:

Four rectangle zones at each side of a brick

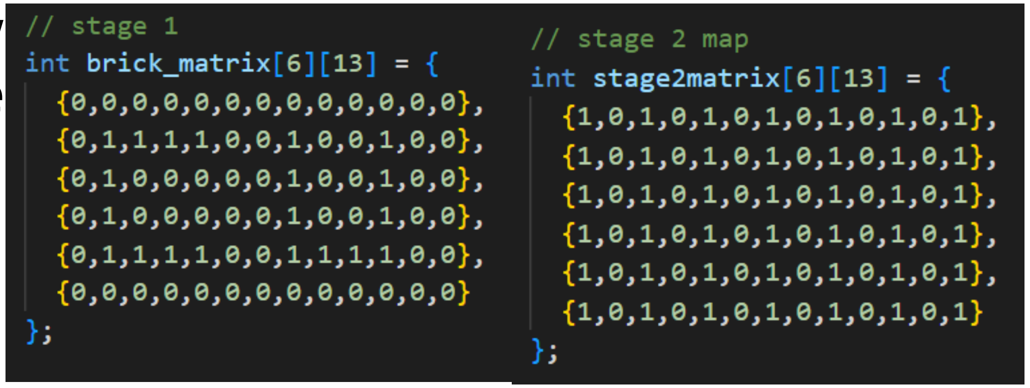
Yellow areas as shown on the right

SW Design – Other information

I. Score
A four-digit "score" at the top. Break one brick = +10 points
Capable of handling the highest number of bricks possible
(6 rows * 13 bricks per row * 2 stages * 10 pts per brick = 1560 points maximum)

II. HP indicator
When the ordinate of the ball is greater than or equal to
execute "game_hp -= 1;".
When game_hp = 0, the game ends.

III. Game stage number
Stage 1 is relatively easy: fewer bricks and a slow
Stage 2 is more difficult: more bricks with a faste



```
// stage 1
int brick_matrix[6][13] = {
  {0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,1,1,1,1,0,0,1,0,0,1,0,0},
  {0,1,0,0,0,0,0,1,0,0,1,0,0},
  {0,1,0,0,0,0,0,1,0,0,1,0,0},
  {0,1,1,1,1,0,0,1,1,1,1,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0}
};
```

```
// stage 2 map
int stage2matrix[6][13] = {
  {1,0,1,0,1,0,1,0,1,0,1,0,1},
  {1,0,1,0,1,0,1,0,1,0,1,0,1},
  {1,0,1,0,1,0,1,0,1,0,1,0,1},
  {1,0,1,0,1,0,1,0,1,0,1,0,1},
  {1,0,1,0,1,0,1,0,1,0,1,0,1},
  {1,0,1,0,1,0,1,0,1,0,1,0,1}
};
```

SW Design - Win & Loss Logic

- Clear all balls in both stages → CONGRATS!
- Fail to catch the ball for three time in one stage → GG!

- Think the game is too hard? We made a “cheating mode” button for you with love ♡

Thank you!