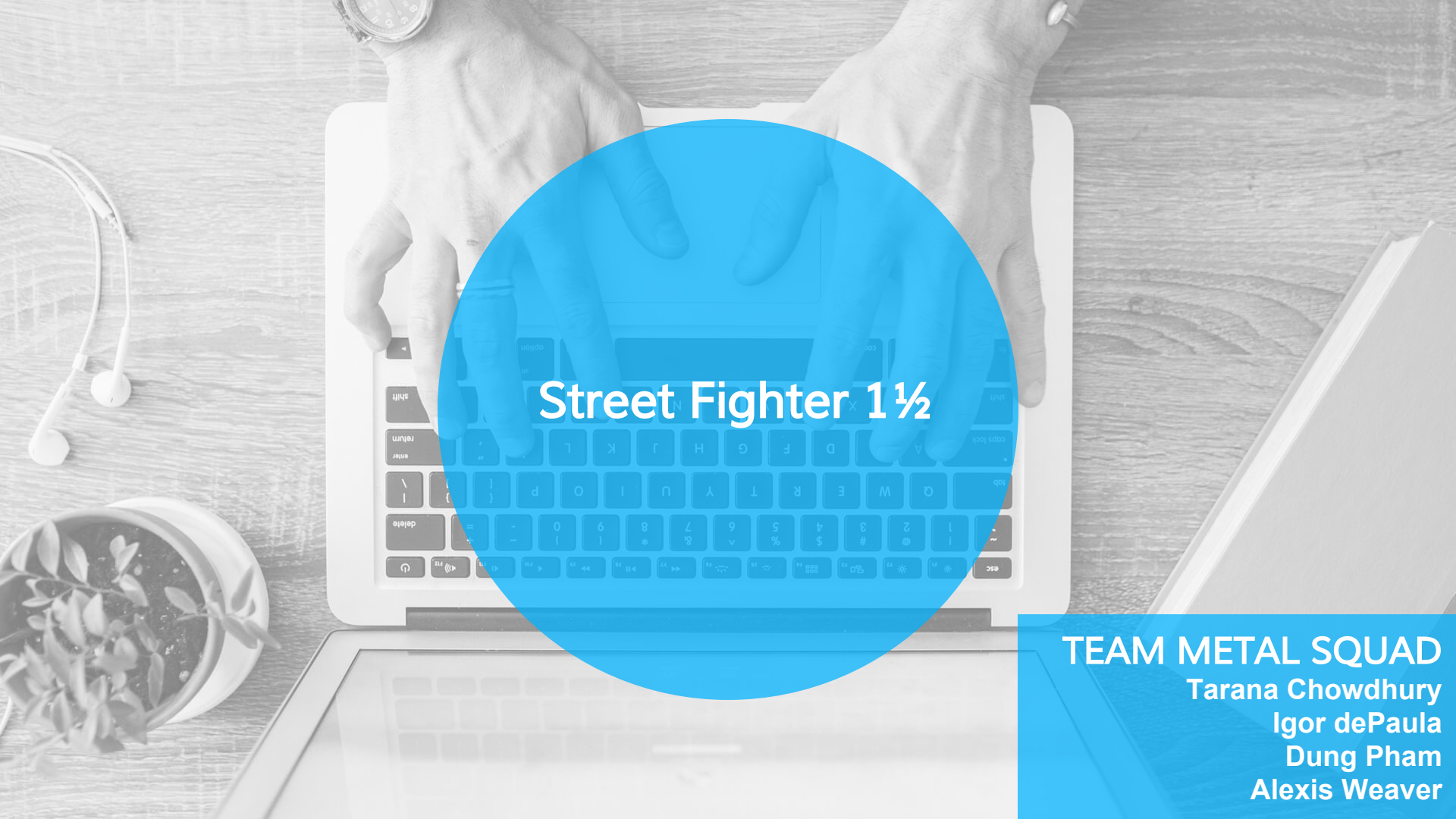


Street Fighter 1½

TEAM METAL SQUAD

Tarana Chowdhury

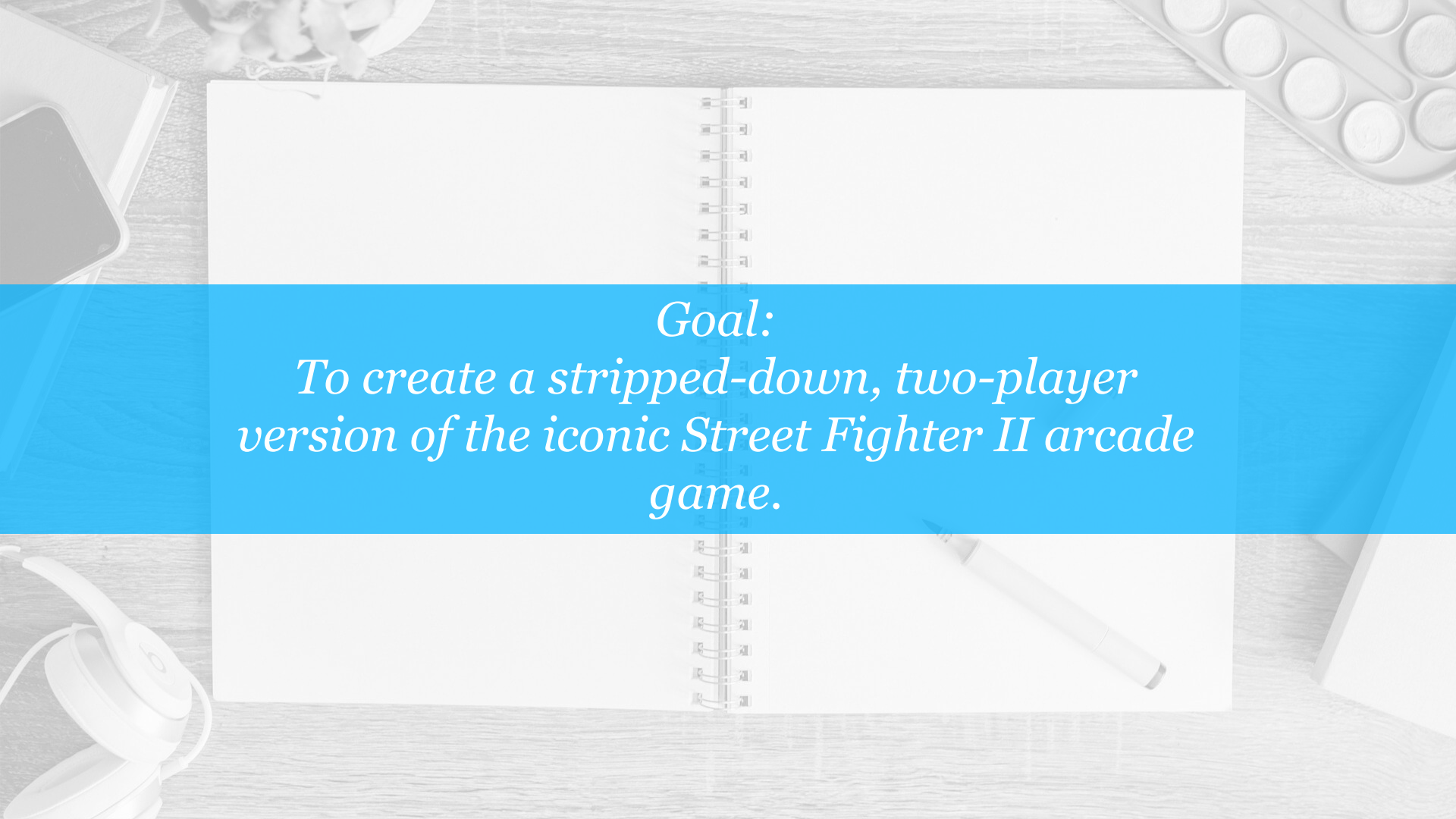
Igor dePaula

Dung Pham

Alexis Weaver

Goal & Motivation

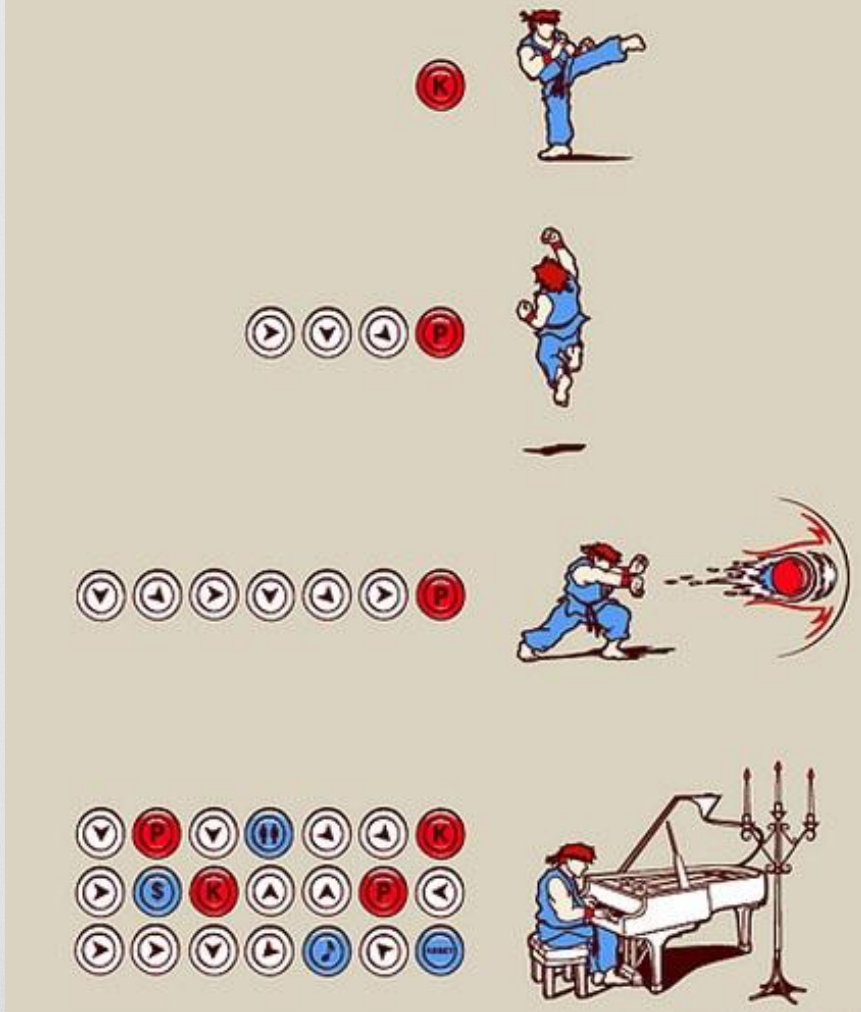
What did we set out to accomplish, and why?



Goal:
To create a stripped-down, two-player version of the iconic Street Fighter II arcade game.

Motivation

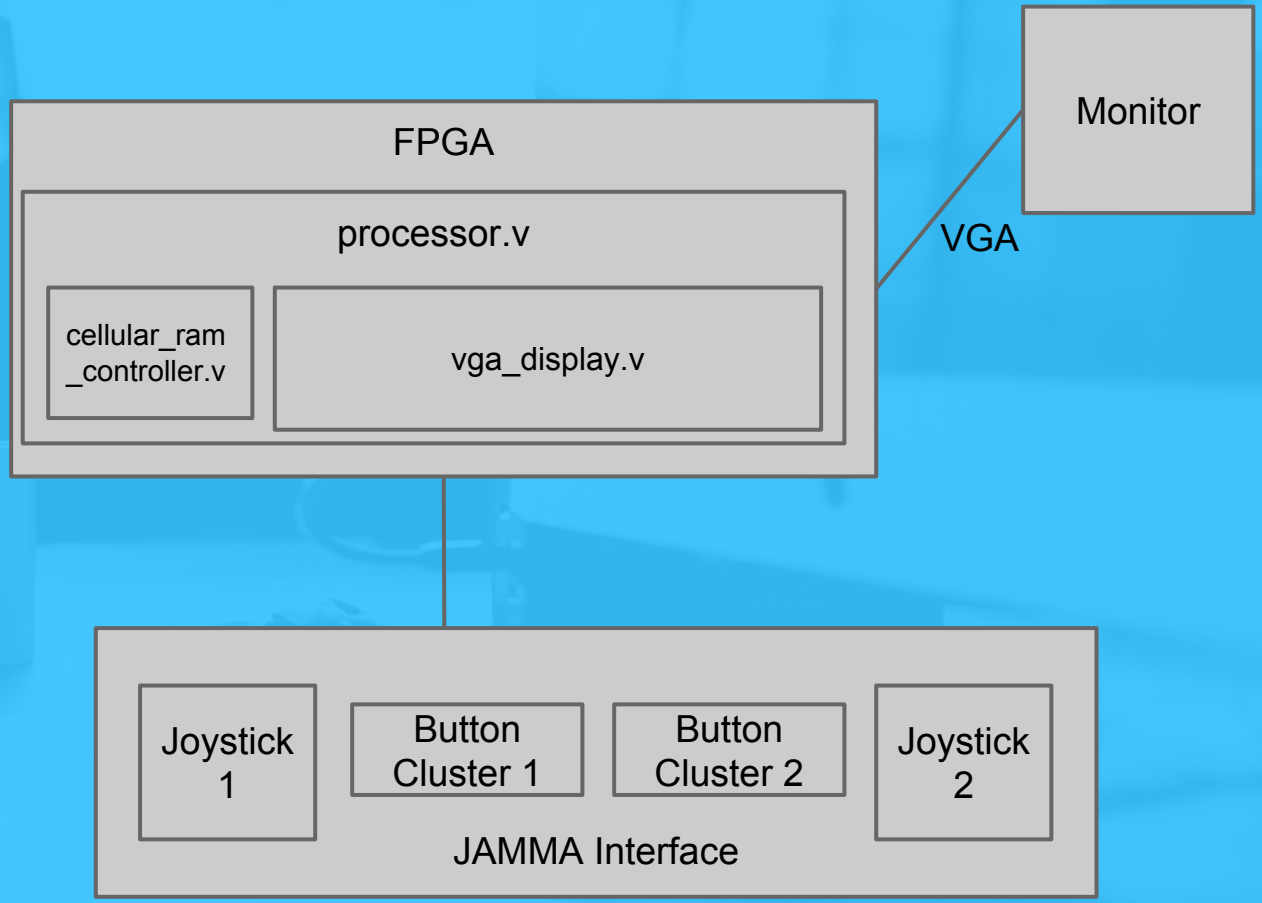
1. Challenging
2. Modular
3. Expandability
4. Fun!



Short Functionality

High-level project overview

High-Level System Block Diagram



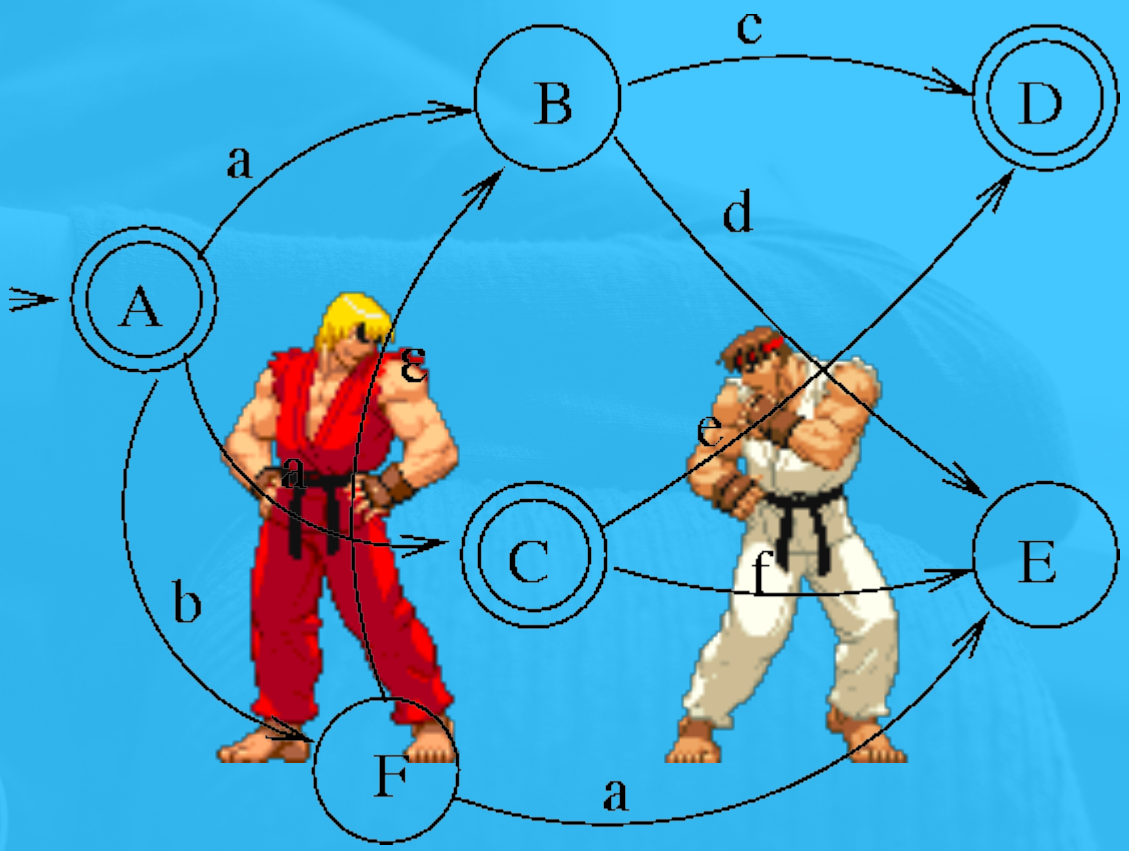
High-Level Gameplay Implementation

vga_display.v

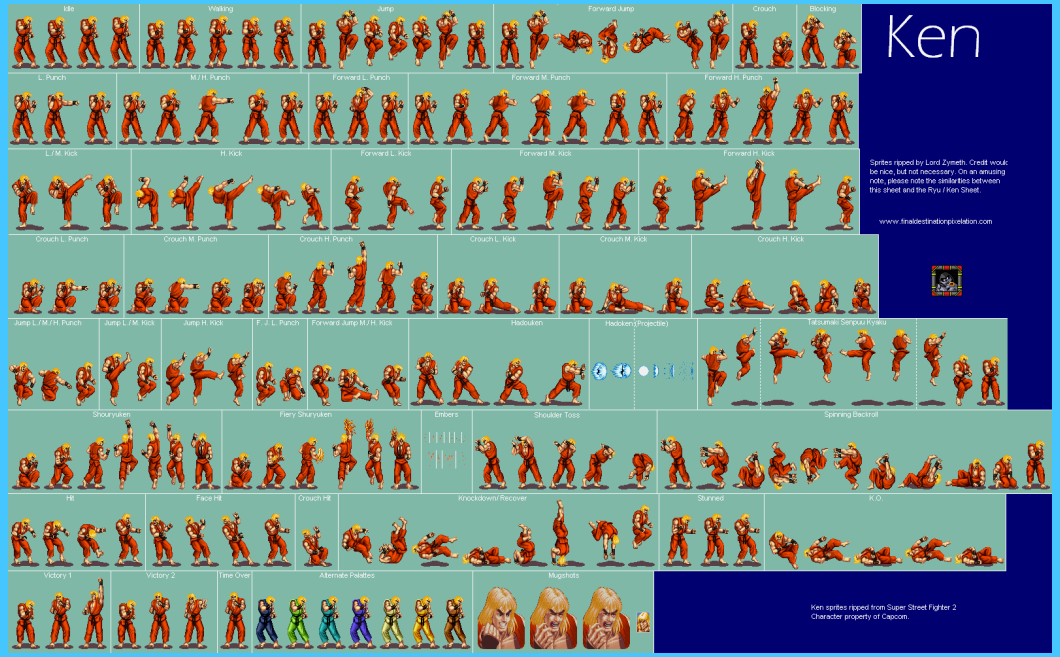
ken:
moves

ryu:
moves

control:
state & move



Sprite Sheets and Memory

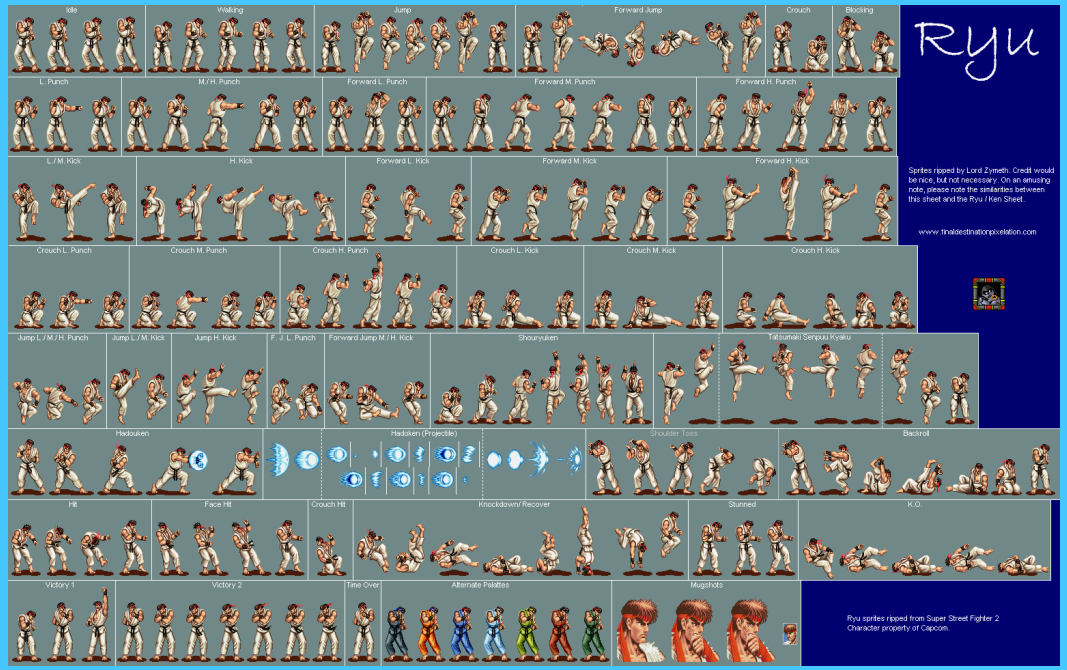


194 KB

We used the following techniques to load the sprite sheets onto the FPGA:

- .png → .coe (6 MB) → IP Core → BRAM (max. ~576 Kbit)
- .png → .bin (1 MB) → Adept → DRAM (max. ~128Mbit)

Sprite Sheets and Memory

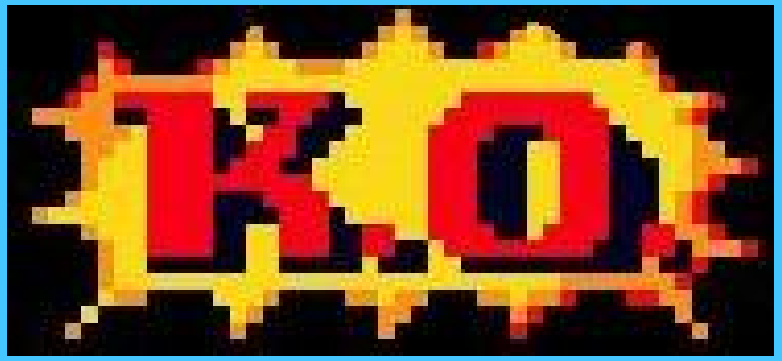


198 KB

We used the following techniques to load the sprite sheets onto the FPGA:

- .png → .coe (6 MB) → IP Core → BRAM (max. ~576 Kbit)
- .png → .bin (1 MB) → Adept → DRAM (max. ~128Mbit)

Sprite Sheets and Memory



+

We used the following techniques to load the sprite sheets onto the FPGA:

- .png → .coe (6 MB) → IP Core → BRAM (max. ~576 Kbit)
- .png → .bin (1 MB) → Adept → DRAM (max. ~128Mbit)

Device Utilization

Number of RAMB16BWERs	28	32	87%
Number of RAMB8BWERs	0	64	0%
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%
Number of BUFG/BUFGMUXs	5	16	31%
Number used as BUFGs	5		
Number used as BUFGMUX	0		
Number of DCM/DCM_CLKGENs	0	4	0%
Number of ILOGIC2/ISERDES2s	0	248	0%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	248	0%
Number of OLOGIC2/OSERDES2s	0	248	0%
Number of BSCANs	0	4	0%
Number of BUFHs	0	128	0%
Number of BUFPLLs	0	8	0%
Number of BUFPLL_MCBs	0	4	0%
Number of DSP48A1s	30	32	93%

Device Utilization

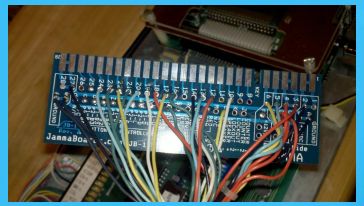
Number of Slice LUTs	5,345	9,112	58%
Number used as logic	5,294	9,112	58%
Number using O6 output only	2,459		
Number using O5 output only	878		
Number using O5 and O6	1,957		
Number used as ROM	0		
Number used as Memory	0	2,176	0%
Number used exclusively as route-thrus	51		
Number with same-slice register load	1		
Number with same-slice carry load	50		
Number with other load	0		
Number of occupied Slices	1,741	2,278	76%
Nummber of MUXCYs used	3,508	4,556	76%
Number of LUT Flip Flop pairs used	5,352		
Number with an unused Flip Flop	3,794	5,352	70%
Number with an unused LUT	7	5,352	1%
Number of fully used LUT-FF pairs	1,551	5,352	28%
Number of unique control sets	37		
Number of slice register sites lost to control set restrictions	133	18,224	1%
Number of bonded IOBs	76	232	32%
Number of LOCed IOBs	75	76	98%

Short Specification

What were the project requirements?

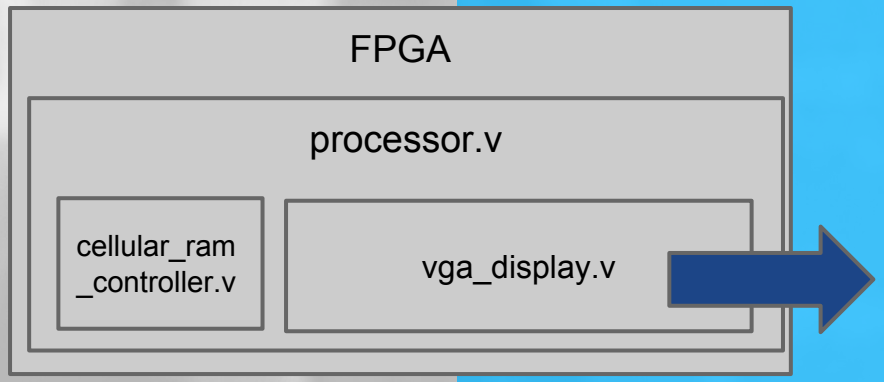
Project Requirements

- VGA Display
- Nexys 3 Spartan 6 FPGA
- JAMMA control interface




Detailed
Functionality
Low-level project details

Detailed System Block Diagram



- **Registers: stores current status of game**
  - location (horizontal and vertical)
  - health (point system)
  - move
  - start, pause, knocked out

- **Move of Ken and Move Ryu as FSM**
- **State of Ken and Ryu as FSM** (includes collision detection)
- **Memory Address Calculation**
- **Logic for VGA display output**

Verilog Code Snippet - Ken Jump

```
6'b11: begin // jump
    case (state_ken)
    6'd0: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450);
          player_1_region = (hcount <= (16'd50 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    6'd1: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450) + 2
          player_1_region = (hcount <= (16'd40 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    6'd2: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450) + 2
          player_1_region = (hcount <= (16'd38 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    6'd3: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450) + 2
          player_1_region = (hcount <= (16'd36 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    6'd4: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450) + 2
          player_1_region = (hcount <= (16'd38 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    6'd5: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450) + 2
          player_1_region = (hcount <= (16'd40 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    6'd6: begin
          pixel_addr_palyer_1 = (vcount - character_y_loc + ken_height) * ken_img_size + ((hcount - location_1) + 25'd450) + 2
          player_1_region = (hcount <= (16'd50 + location_1)) && (hcount >= (location_1)) && ((vcount >= character_y_loc - ken
          end
    endcase
```

Verilog Code Snippet

Logic to switch between different moves (Ken)

```
always @ (posedge clk) begin
   if(reset_reg)begin
      move_ken<=6'b0;
   end
   else if(health1==10'd0)begin
         move_ken<=10'b1001;//knock_out
      end
   else if(health2==10'd0) begin
         move_ken<=10'b1000;//victory
      end
   else if (start) begin
   if ((move_ryu == 6'b1 || move_ryu == 6'b101) && (location_2 - location_1 == 10'd50) && ken_height==8'd0) begin
      move_ken <= 6'b100; // got hit!
   end
   else if (punch_1) begin
      if (move_ken != 6'b101) move_ken <= 6'b1; //punch move
      else move_ken <= 6'b111; // special!
      end
   else if (down_1) begin
      move_ken <= 6'b10; // crouch
      end
   else if (up_1) begin
      move_ken <= 6'b11; // jump
      end
   else if (kick_1) begin
      if (move_ken != 6'b1) move_ken <= 6'b101; //kick
      else move_ken <= 6'b111; // special!
      end
   else if (move_ken == 6'b1 && state_ken == 6'b101) begin
      move_ken <= 6'b0;
      end
   else if (move_ken == 6'b10 && state_ken == 6'b10) begin
      move_ken <= 6'b0;
      end
   else if (move_ken == 6'b101 && state_ken == 6'b11) begin
      move_ken <= 6'b0;
      end
   else if (move_ken == 6'b11 && state_ken == 6'b110) begin
      move_ken <= 6'b0;
      end
   else if (move_ken == 6'b100 && state_ken == 6'b100) begin
      move_ken <= 6'b0;
      end
```

PRIDE & JOY

The Verilog code we are most proud of.

Verilog Code Snippet - DRAM Controller

```
module cellular_ram_controller (clk, OE, WE, ADV, CLK, CE, CRE, UB, LB, A,
                                WAIT, DQ, ram_addr);
//asynchronous read
input WAIT, clk;
input [15:0] DQ;
input [25:0] ram_addr;
output OE, WE, ADV, CLK, CE, CRE, UB, LB;
output [25:0] A;


assign A = ram_addr;//for testing

assign OE=0; // asynchronous reading
assign WE=1;
assign ADV=0;
assign CLK=0;
assign CE=0;
assign CRE=0;
assign UB=0;
assign LB=0;



endmodule
```

Success Analysis

How and why our project is successful.

The Game SUCCESS

Players:

- Ryu
- Ken

Moves

- Idle
- Jump
- Crouch
- Kick*
- Punch
- Special (different for each character)
- Victory
- Knockout

Illustrations

- Street Fighter Logo
- KO
- Health Bars

*Ken has an additional punch, instead of kick.

Failure Analysis

What didn’t go as planned.

The Game failures

Features that were planned, but did not make the final cut:

- Only two characters
- No midi music
- No character select
- No game maps
- Not all moves implements

*All these obstacles are due to limited on-board memory

Remaining Work

What's left to be done.

Finishing Touches

- Implement walking for each player

- Add ability to reset/restart gameplay
- Clean-up noisy images

Thanks!