SUPER MARIO

Bo Kizildag, Brandon Khadan, Nico De la Cruz

idea

Iconic Mario boss fight against Bowser

=>

Everyone else does the first level, for novelty’s sake we just had to go with the final level

=>

ACTUAL GAME SCREENSHOT OF BOWSER

design

Every particular build choice was made with memory safety as a priority. Each design module, from the sprites to the audio is optimized to consume as little resources as possible

controls

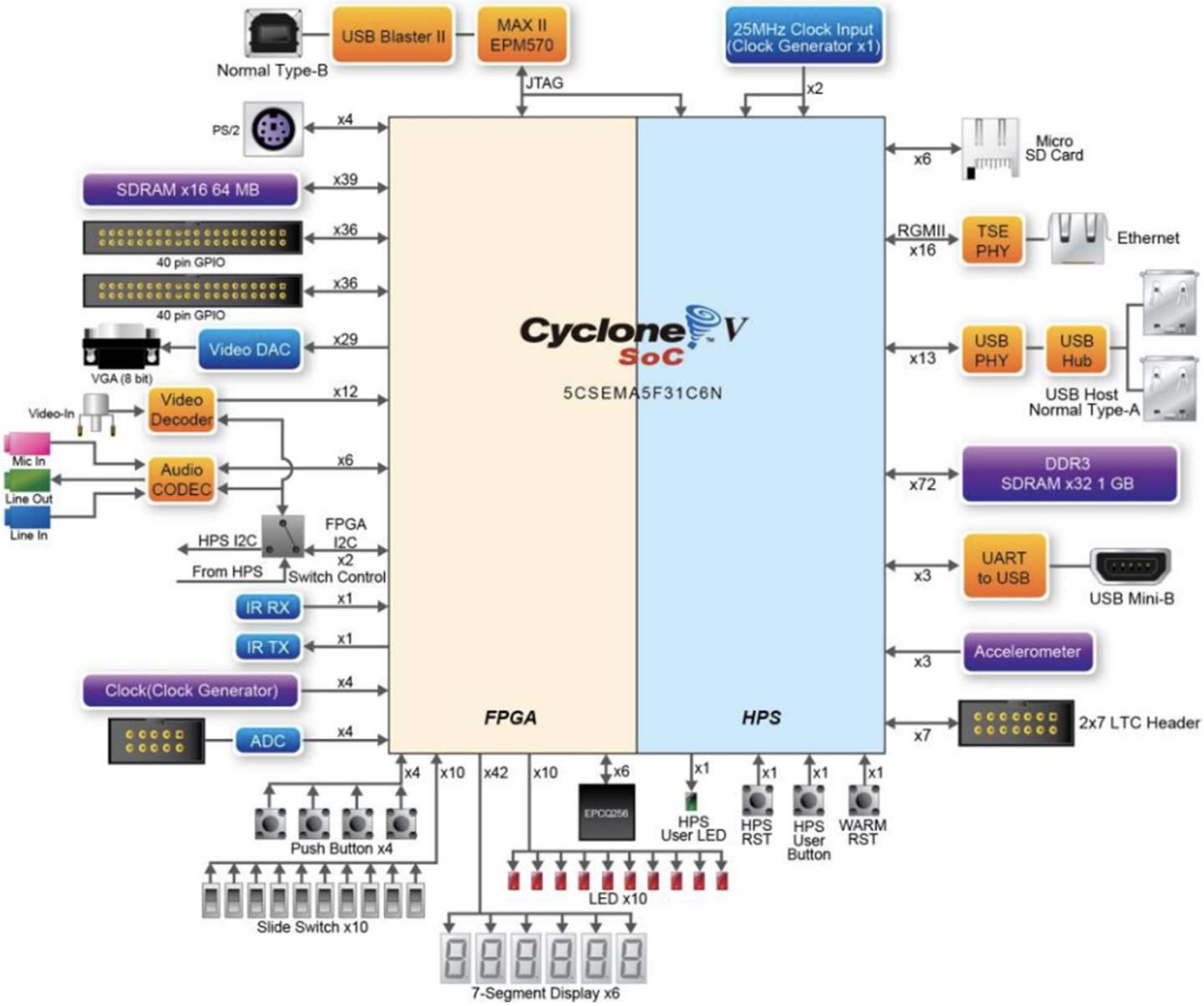
Keyboard scheme

A	- left
D	- right
SPACE	- jump


peeking under
the hood

DE1-SoC board

connections



hardware

- Sprites

Each sprite has its own memory file and module to handle access

- Pixel processing unit

camera & hitbox

sound engineering

The octave is divided into 12 logarithmically equal steps, each step being a semitone. This division means that the frequency ratio between any two adjacent notes (like C and C#, or E and F) is the twelfth root of two (21/1221/12), approximately 1.05946. This system allows for consistent intervals across keys, which is essential for the flexibility in modulation and transposition in modern music composition and performance.

Calculating frequencies:

To calculate the frequencies of the other notes from the reference pitch A4 = 440 Hz, gotta use the formula: Frequency of Note=440×2(*n*12)Frequency of Note=440×2(12*n*) where nn is the number of semitones away from A4. If nn is positive, the note is higher than A4; if nn is negative, the note is lower.

Example:

- **C4 (Middle C)** is 9 semitones below A4. Hence its frequency is: 440×2−(912)≈261.63 Hz440×2−(129)≈261.63 Hz
- **D4** is 7 semitones below A4, so: 440×2−(712)≈293.66 Hz440×2−(127)≈293.66 Hz
- **E4** is 5 semitones below A4, so: 440×2−(512)≈329.63 Hz440×2−(125)≈329.63 Hz
- **F4** is 4 semitones below A4, so: 440×2−(412)≈349.23 Hz440×2−(124)≈349.23 Hz
- **G4** is 2 semitones below A4, so: 440×2−(212)≈392.00 Hz440×2−(122)≈392.00 Hz

audio

NoteGenerator

Responsible for generating individual musical tones

I2S_Controller

1. Manages the I2S protocol to transmit audio data to the WM8731 codec

AudioGenerator

Orchestrates the overall audio generation, managing the sequence of notes (it can also control when they play)

tones

Tone generation via a counter to create a square wave at a specific frequency determined by the half_period input

Utilizes an internal counter that increments on every clock cycle when note_enable is high

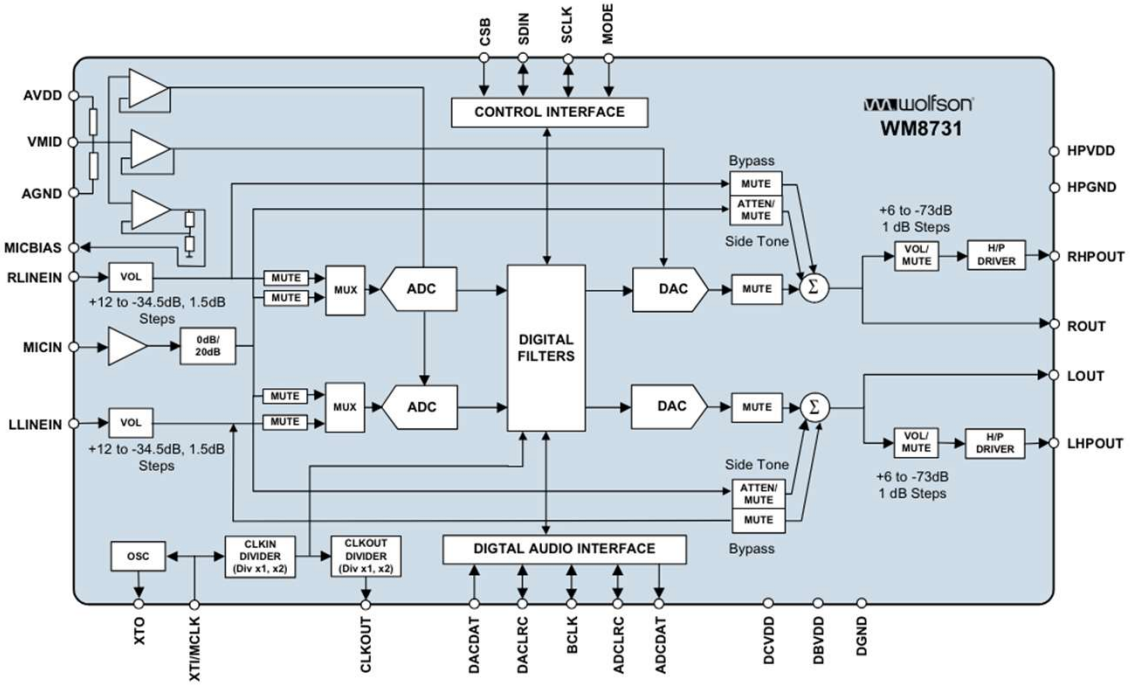
When the counter reaches the half_period value, the output (note_out) toggles, creating a square wave

The frequency of the square wave is determined by how quickly the counter reaches the half_period value, setting the tone's pitch

Wolfson DAC

wm8731

BLOCK DIAGRAM



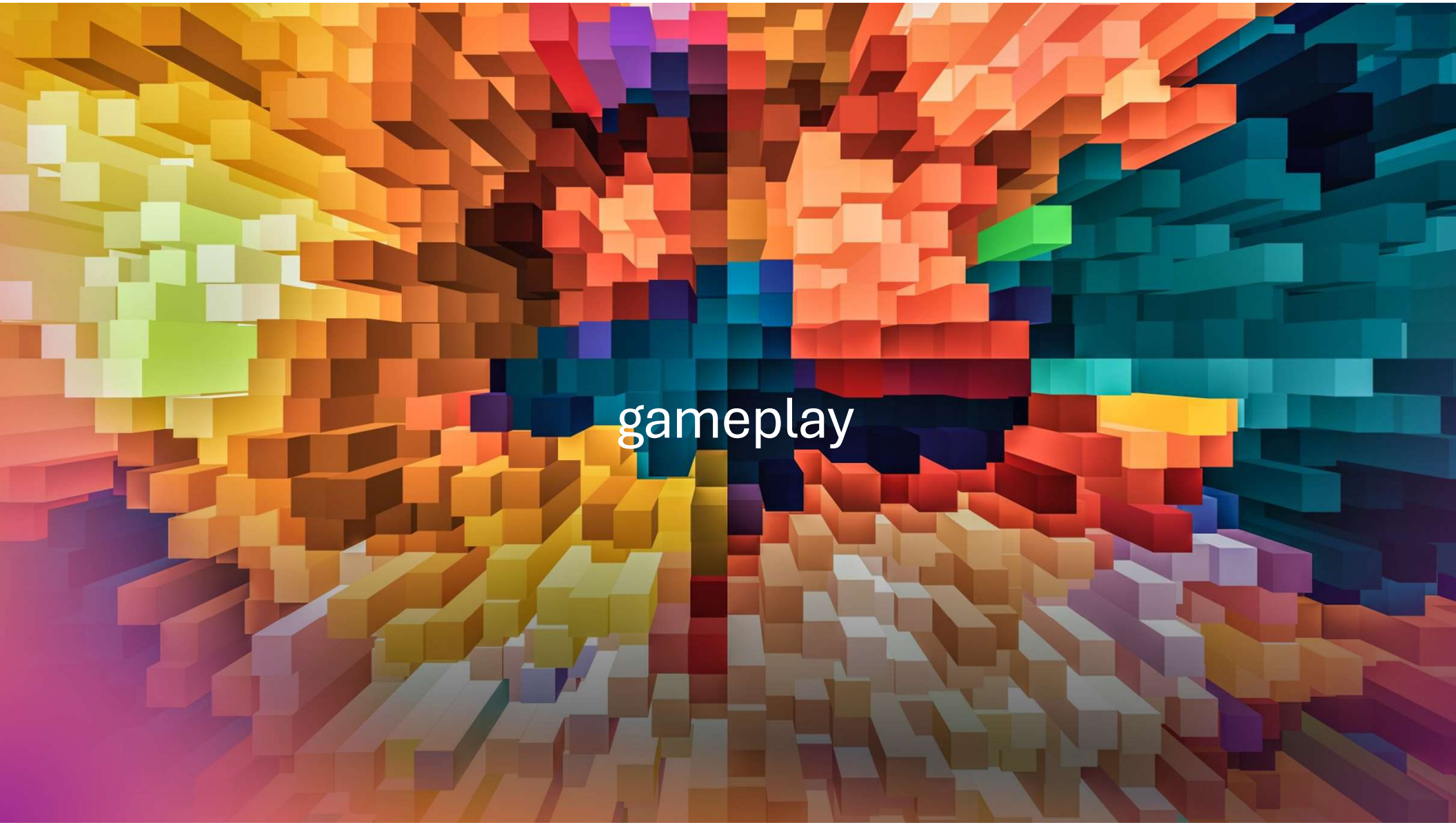
software utilization

Platform and in-game struct generation

DAC configuration

Everything else was built in SystemVerilog!

(Thank you Dennis Ritchie!)


gameplay