Ninja University

IN THE CITY OF NEW YORK

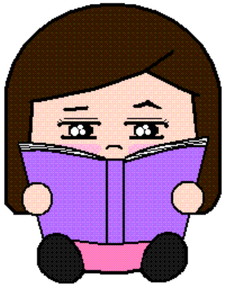
Kshitij Bhardwaj kb2673
Van Bui vb2363
Vinti Vinti vv2236
Kuangya Zhai kz2219

Overview

- Wiimote controlled object slicing game on SoCKit board
- Motivated by Fruit Ninja game
- **Storyline**: To become a Ninja, you must be very diligent and fulfill program requirements by slicing your assignments, exams, write your thesis, etc
- Strategy to become a Ninja
    - Slice objects to increase your score
    - Avoid slicing an F object
    - Slice objects before they disappear from the screen

Image Preprocessing

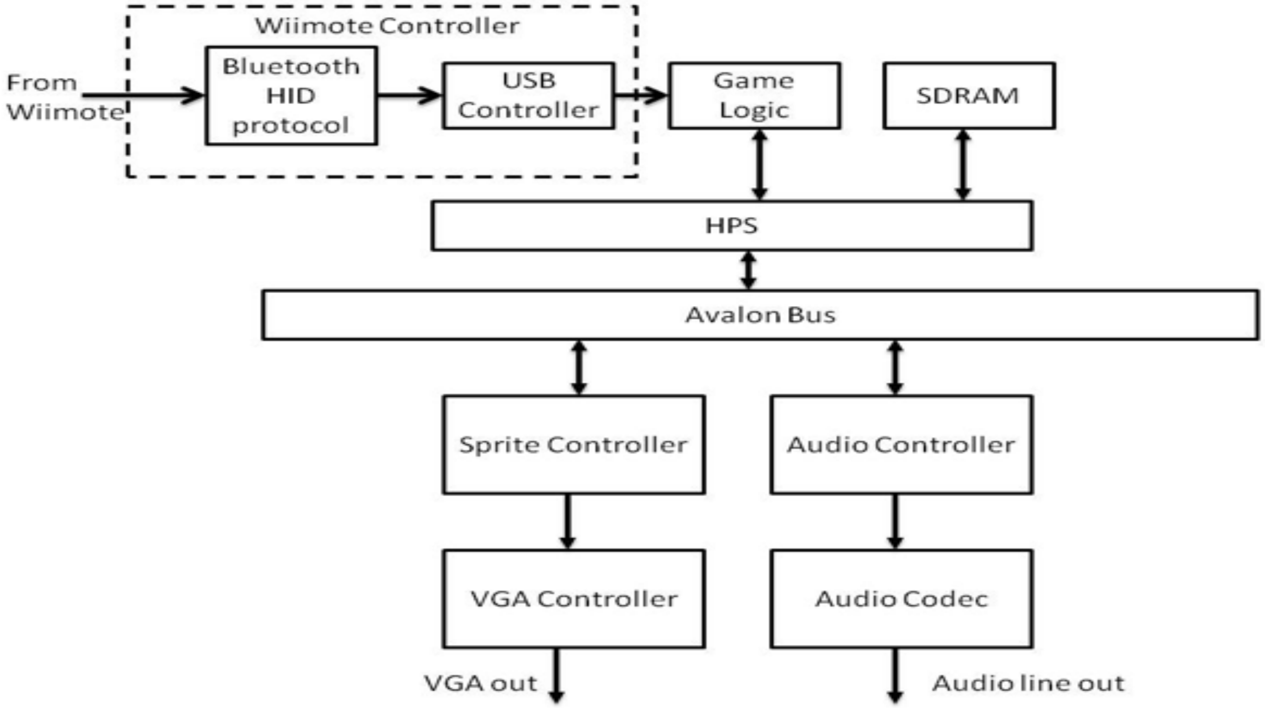
- Images for stationary and moving objects
- Generate a memory initialization file for each image
- Single-port ROM memory blocks
- 12-bit index color, i.e. 4096 colors



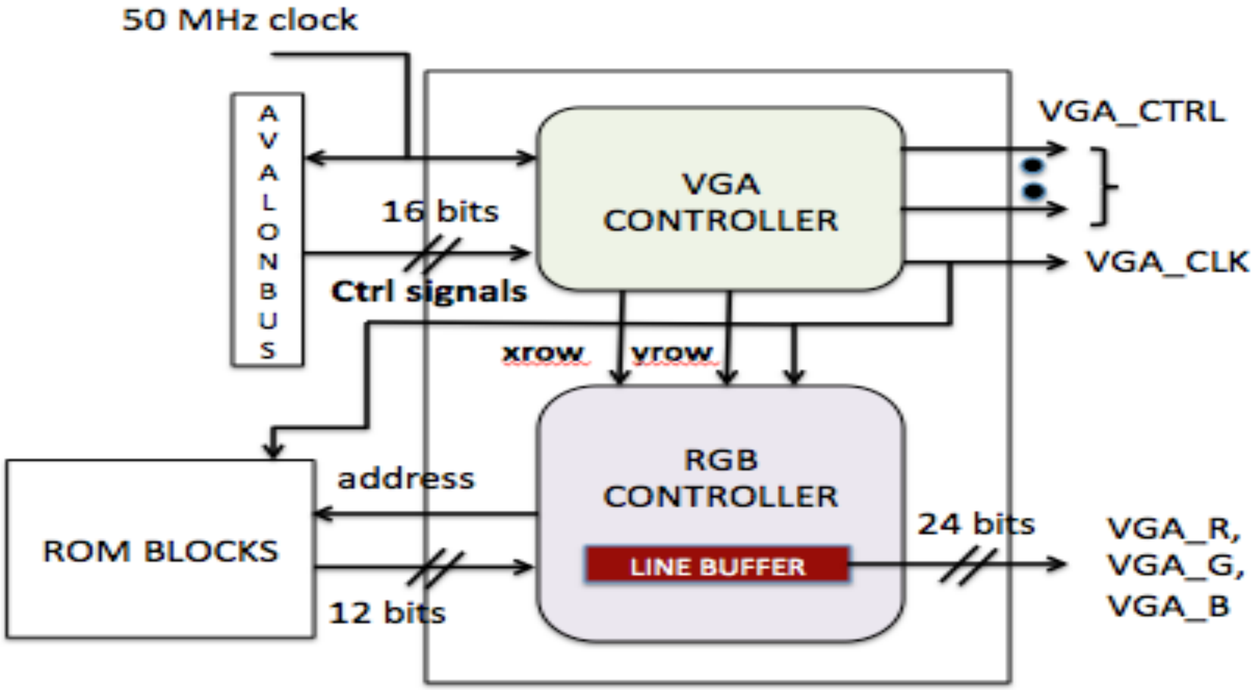
Audio Preprocessing

- Background music and sound effects
- Ogg Vorbis decoding - conversion to MIF format
- Single-port ROM memory blocks
- Sampling rate: 44100 Hz
- Quantization bits: 16 bits
- Edit audio files for length, channels, and sampling rate

Hardware Design

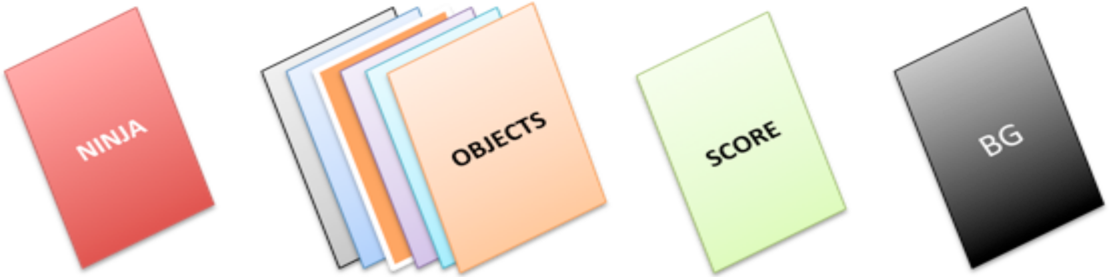


VGA DISPLAY MODULES

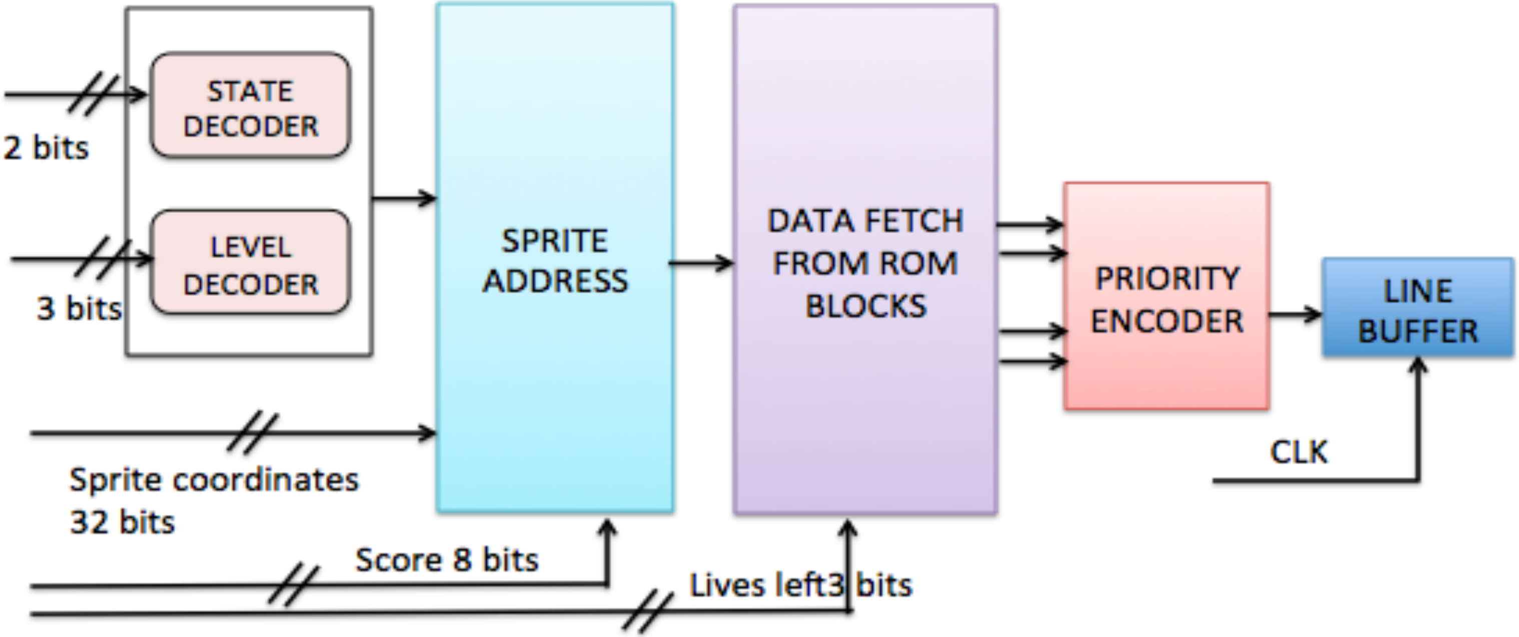


RGB Controller

- Sprite selection and movement controlled by software
- Hardware generates images
- Line buffer write operation
- Priority encoder for sprites



Line Buffer Write Operation



- Two line buffers used for reading and writing
- Write at alternate rows

Sprite selection logic

- Sprite selection and position based on control input (on/off flag and coordinates) from game logic.
- Flag checking, calculation of address, data fetching done using combinational logic, in parallel for all layers (to ensure no timing issues).
- Priority encoder used for selecting the pixel to be written into line buffer.
- Writing into line buffer using sequential logic at 25MHz clock frequency.
- Used combinational logic to simplify design, other options could be pipelining/ interleaving.

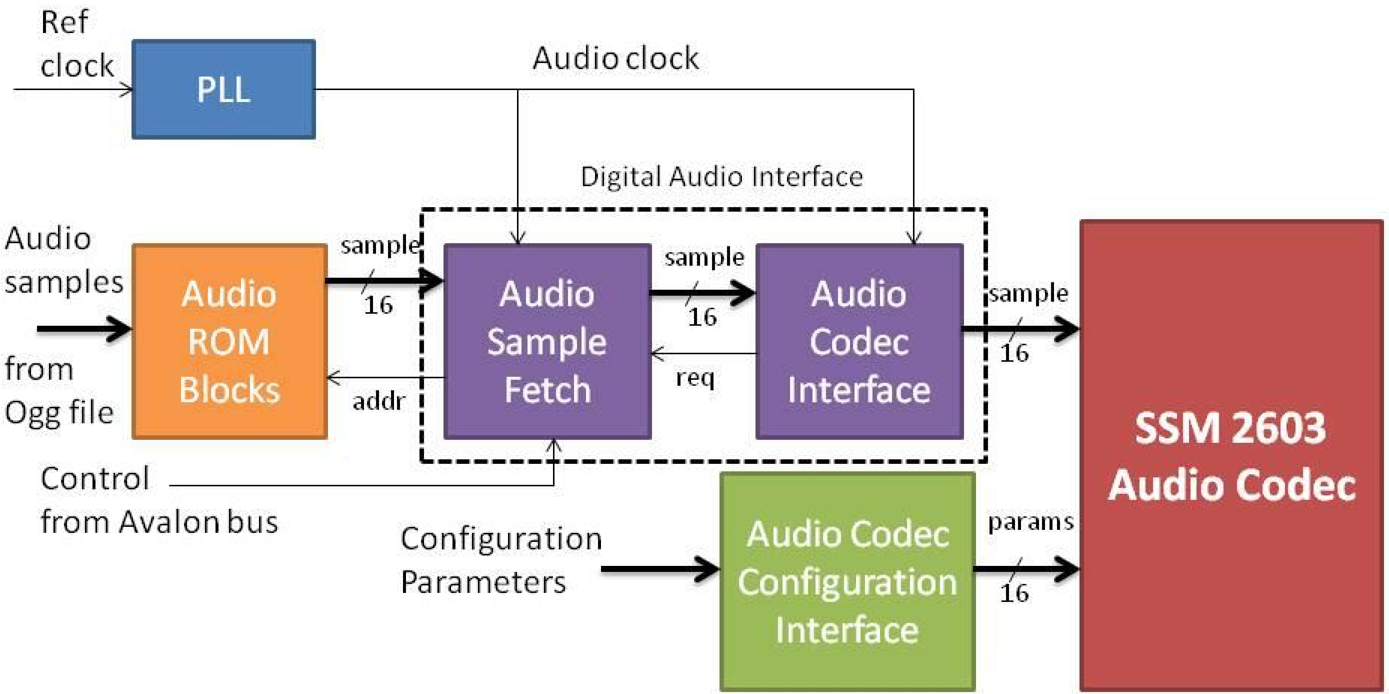
List of Sprites

Block	**Number of Sprites**	**Size of Images (pixels)**	**Total ROM size (bytes)**
Numbers	10	32x32	61440
Lives	1	32x32	1536
Ninja	3	64x64	18432
Weather	3	64x64	18432
Slicing Objects	6	64x64	36864
Level Selection	3	64x64	18432
Try Again	1	64x64	6144
Diploma	1	64x64	6144
NYC Skyline	3	200x160	144000
Pass/Fail	2	64x64	96000
Total	33		401.28 KB

Audio Controller: Major Components

- Audio Data
    - Audio Samples stored in ROM blocks
- Audio Codec Configuration Interface
    - Configure audio codec SSM2603
- Digital Audio Interface
    - Send audio samples from ROM to audio codec at audio clock rate

Audio Controller: Block Diagram



Audio ROM blocks

- Two sounds converted from ogg file format to mifs:
  - city.mif : Background music
  - sword.mif: Ninja striking an object sound
- Both sounds stored in ROM blocks
  - city: 16 bit samples, 16537 words
  - sword: 16 bit samples, 22049 words
  - total size: 77 KB

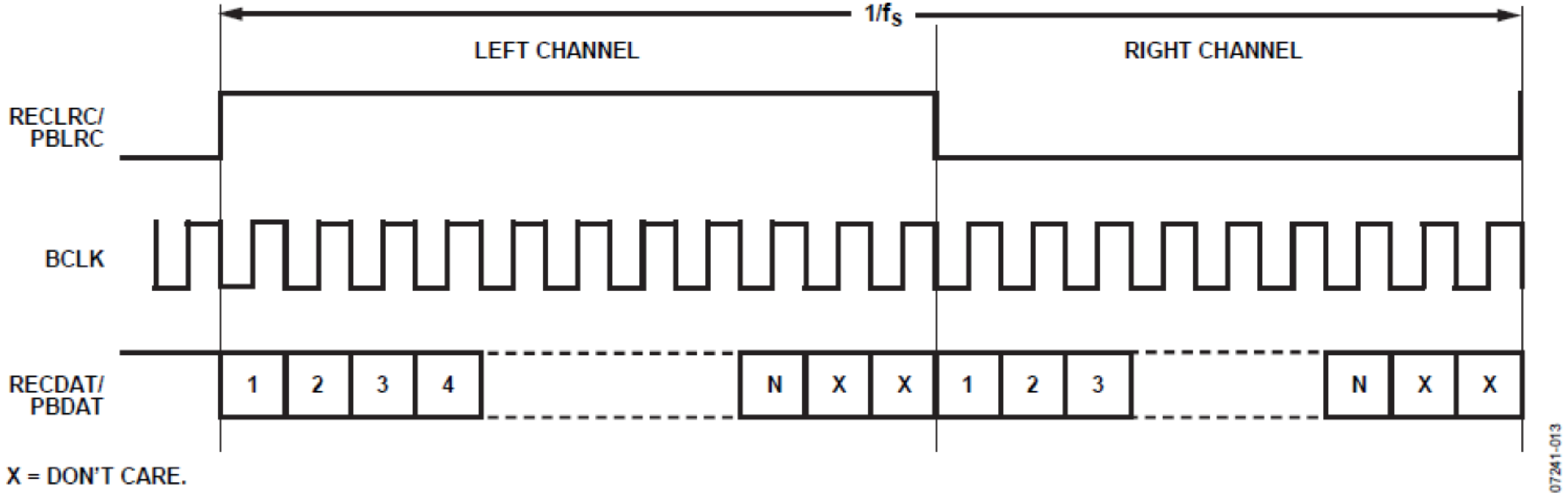
Audio Codec Configuration Interface

- Uses I2C protocol to configure 16 9-bit registers in audio codec SSM 2603
- Configured parameters include
  - Volume (0 db)
  - Mode (slave)
  - Sampling rate (44.1 khz)
  - Power on/off

Audio Codec Interface

- Operates at audio clock (11.3 Mhz)
- Implemented as Shift registers that send audio samples to audio codec
- Two clocks derived from audio clock
  - Channel clock: Time multiplexed, send sample on one channel (left or right) at a time
  - Bit clock: send a bit of each sample

Audio Codec Interface: Operation



Kernel Device Driver Modules

- VGA device driver
  - Ioctl calls to write positions (x,y) of sprites, scores, remaining lives, select screens, select levels
  - VGA peripheral memory: 4-bit address, 16-bit words
- Audio device driver
  - Ioctl calls to control (on/off) of sword sound
  - Audio peripheral memory: 1-bit address, 16-bit word
  - Can be easily extended to control other sounds...

Debugging Methods

- System console scripts to test hardware
  - Audio sound
  - Sprites display
- Modelsim
- Modular design coding

Wiimote Controller

- Peripherals
  - wiimote, infrared sensor light, bluetooth USB dongle
- Software
  - Libwiimote (C-library)
  - Linux Device Driver: BlueZ, libwiimote-dev
- Recognize the infrared source on the screen
- Cast the screen size from 1784 x 1272 to 640 x 480
- Vibrate when cutting the bomb

Game Logic

- Implemented in the software world by C
- Interaction between user and hardware
    - User: bluetooth dongle connected to USB
    - Hardware: VGA and audio device driver
- Do the computation and control the game...
    - Input: infrared source position from wiimote
    - Output: current screen, position of sprites, ninja, enabling the sound and vibration, score, life...

Experiences and Issues

- Wiimote connection takes longer than expected
- Codesign by contract in the favor of the hardware
- Interfacing with audio codec were the most difficult
- Audio buffers and interrupts
- Limited on-chip memory space

Lessons Learned

- Architecture design of SoCKit board
- Software and hardware co-design
- Collaborative coding
- Time management
- SoCKit tutorials by Howard Mao were very helpful
- Simple implementation first, then optimize as needed

Demo