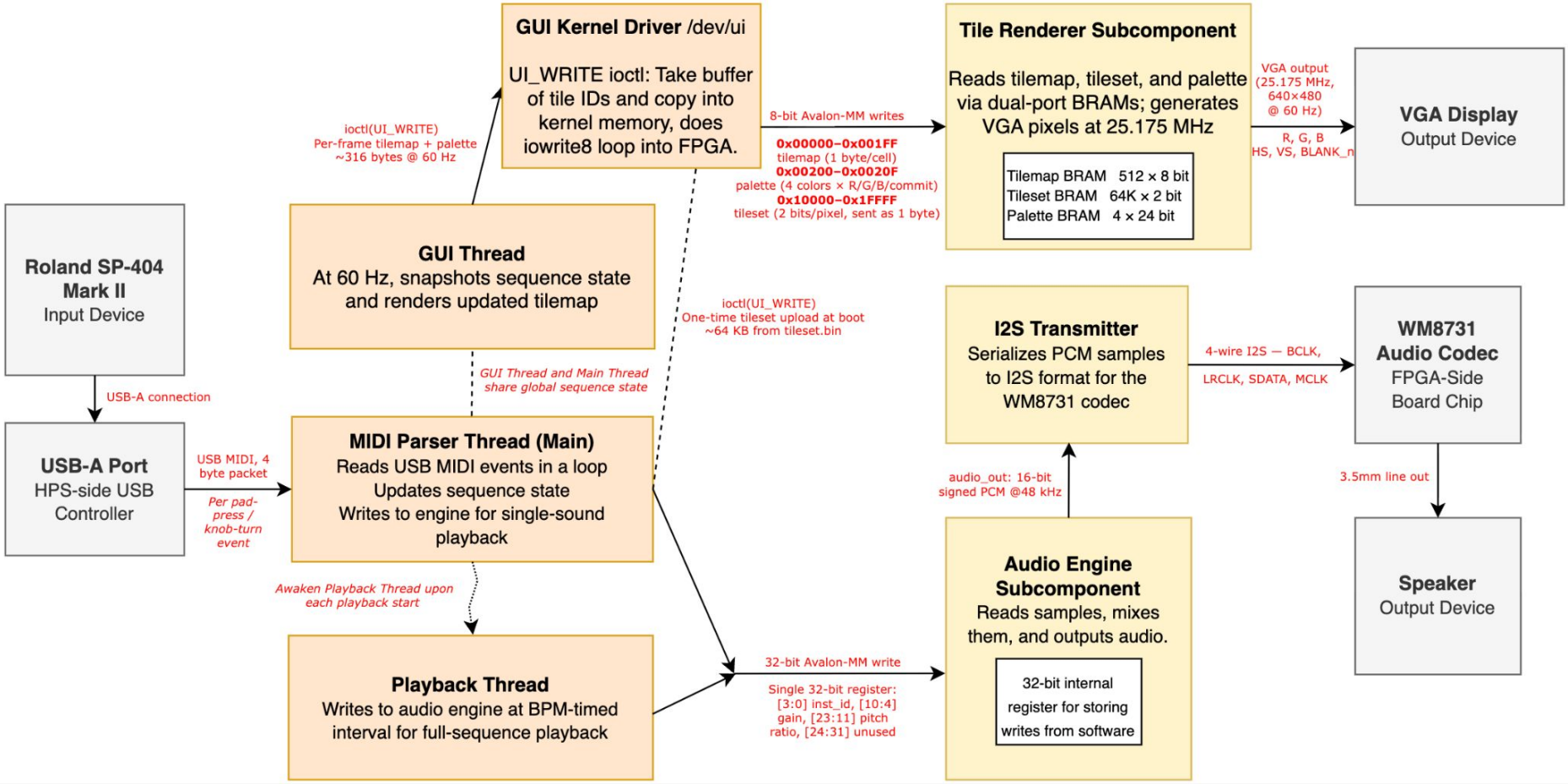

GUI Kernel Driver /dev/ui
UI_WRITE ioctl: Take buffer of tile IDs and copy into kernel memory, does iowrite8 loop into FPGA.
Tile Renderer Subcomponent
Reads tilemap, tileset, and palette via dual-port BRAMs; generates VGA pixels at 25.175 MHz
Tilemap BRAM 512 × 8 bit
Tileset BRAM 64K × 2 bit
Palette BRAM 4 × 24 bit
VGA output (25.175 MHz, 640×480 @ 60 Hz)
R, G, B
HS, VS, BLANK_n
VGA Display
Output Device
ioctl(UI_WRITE)
Per-frame tilemap + palette
~316 bytes @ 60 Hz
8-bit Avalon-MM writes
0x00000–0x001FF
tilemap (1 byte/cell)
0x00200–0x0020F
palette (4 colors × R/G/B/commit)
0x10000–0x1FFFF
tileset (2 bits/pixel, sent as 1 byte)
GUI Thread
At 60 Hz, snapshots sequence state and renders updated tilemap
ioctl(UI_WRITE)
One-time tileset upload at boot
~64 KB from tileset.bin
Roland SP-404 Mark II
Input Device
USB-A connection
GUI Thread and Main Thread share global sequence state
I2S Transmitter
Serializes PCM samples to I2S format for the WM8731 codec
4-wire I2S — BCLK, LRCLK, SDATA, MCLK
WM8731 Audio Codec
FPGA-Side Board Chip
3.5mm line out
USB-A Port
HPS-side USB Controller
USB MIDI, 4 byte packet
Per pad-press / knob-turn event
MIDI Parser Thread (Main)
Reads USB MIDI events in a loop
Updates sequence state
Writes to engine for single-sound playback
audio_out: 16-bit signed PCM @48 kHz
Speaker
Output Device
Awaken Playback Thread upon each playback start
Audio Engine Subcomponent
Reads samples, mixes them, and outputs audio.
32-bit internal register for storing writes from software
32-bit Avalon-MM write
Single 32-bit register:
[3:0] inst_id, [10:4] gain, [23:11] pitch ratio, [24:31] unused
Playback Thread
Writes to audio engine at BPM-timed interval for full-sequence playback

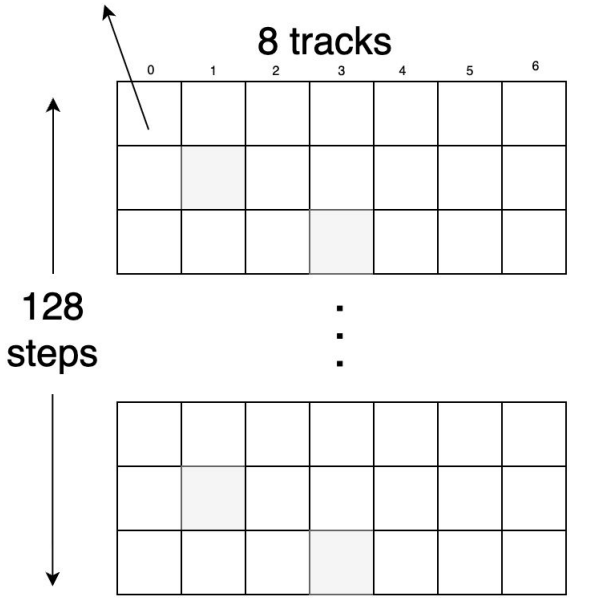

Roland
SP-404MKII
VOLUME
CTRL 1
CTRL 2
CTRL 3
MIN
MAX
CUTOFF
RESONANCE
DRIVE
FILTER +DRIVE
RESONATOR
DELAY
ISOLATOR
DJFX LOOPER
MFX
C1:START
C2:OFF
C3:END
ENC:200M(1x)
A-3
M:[S]
MENU
PUSH ENTER
PATTERN SEQUENCER
SAMPLE EDIT
UNDO
CHOP
ENVELOPE
SAMPLING
SAMPLE MODE
PING-PONG
BANK
ROLL SET
PATTERN STOP
CURRENT PAD
DJ MODE
BUS FX
MUTE BUS
HOLD
PAUSE
EXT SOURCE
INPUT SETTING
SUB PAD
PROJECT
FIXED VELOCITY
16 VELOCITY
CUE
CHROMATIC
EXCHANGE
INIT PARAM
PAD LINK
MUTE GROUPS
METRONOME
COUNT-IN
TAP TEMPO
GAIN
UTILITY
IMPORT/EXPORT
PAD SETTING
EFX SETTING
PHONES
GAIN
MIC/GUITAR
INPUT

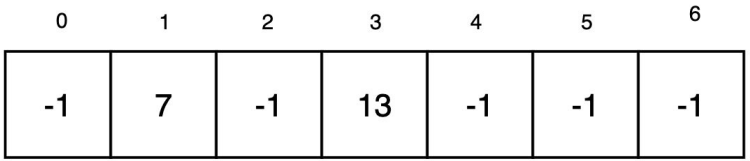
Global Sequencer State

```c
/* Complete sequencer state. The UI calls get_sequencer_state() to pull a snapshot
   and render; drum.c never pushes to the UI. */
typedef struct {
    cell       sequence[NUM_STEPS][MAX_TRACKS];   /* [step][track] */
    int          tracks[MAX_TRACKS];              /* instrument_id per slot, or -1 if unassigned */
    unsigned int measure;                         /* [1, 8]: current measure */
    unsigned int global_bpm;                      /* [64, 192] */
    unsigned int cursor_step;                     /* [0, 15]: step offset within current measure */
    unsigned int cursor_track;                    /* [0, 7]: current track slot */
    unsigned int playback_state;                  /* 0 = editing, 1 = playing back */
    unsigned int playback_step;                   /* [0, 15]: step within current measure during playback */
    unsigned int edit_mode;                       /* NAVIGATION, INSTRUMENT_SELECT */
} sequencer_state;
```

Sequence Array + Active Track Array

```c
typedef struct {
    uint8_t active;  /* 0 or 1, default 0 */
    int8_t  pitch;   /* -24 to 24 semitones, default 0 */
    uint8_t volume;  /* 0-99, default 50 */
} cell;
```

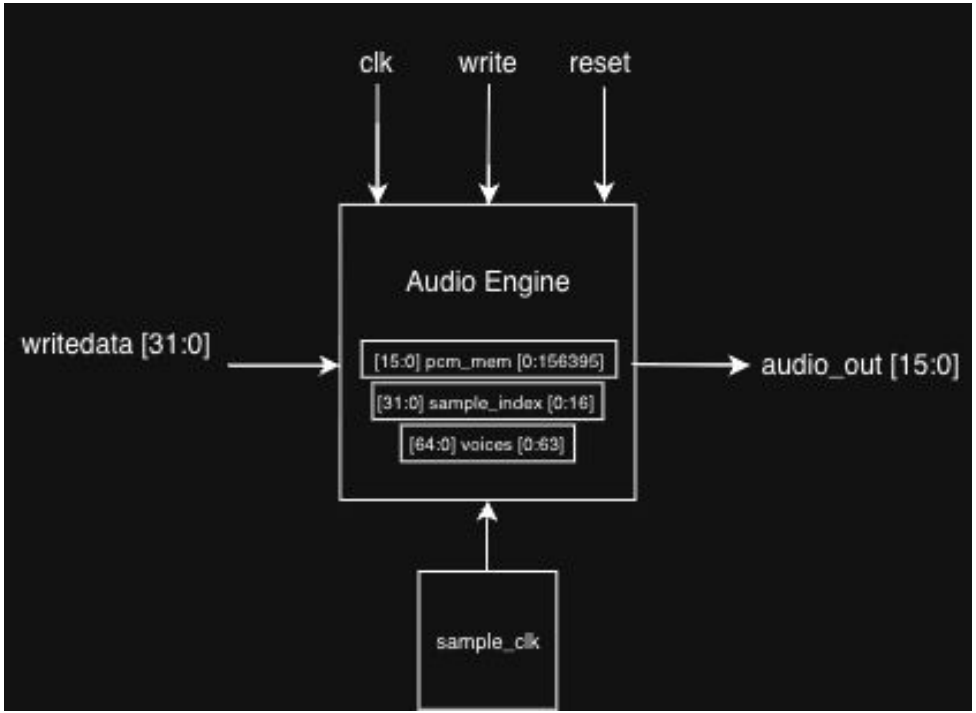




Playback Register Map

- Avalon-MM Slave: audio engine module
- Single register
- Writes over LWH2F bridge (0xFF200000 base address)

FPGA Audio Engine Peripheral

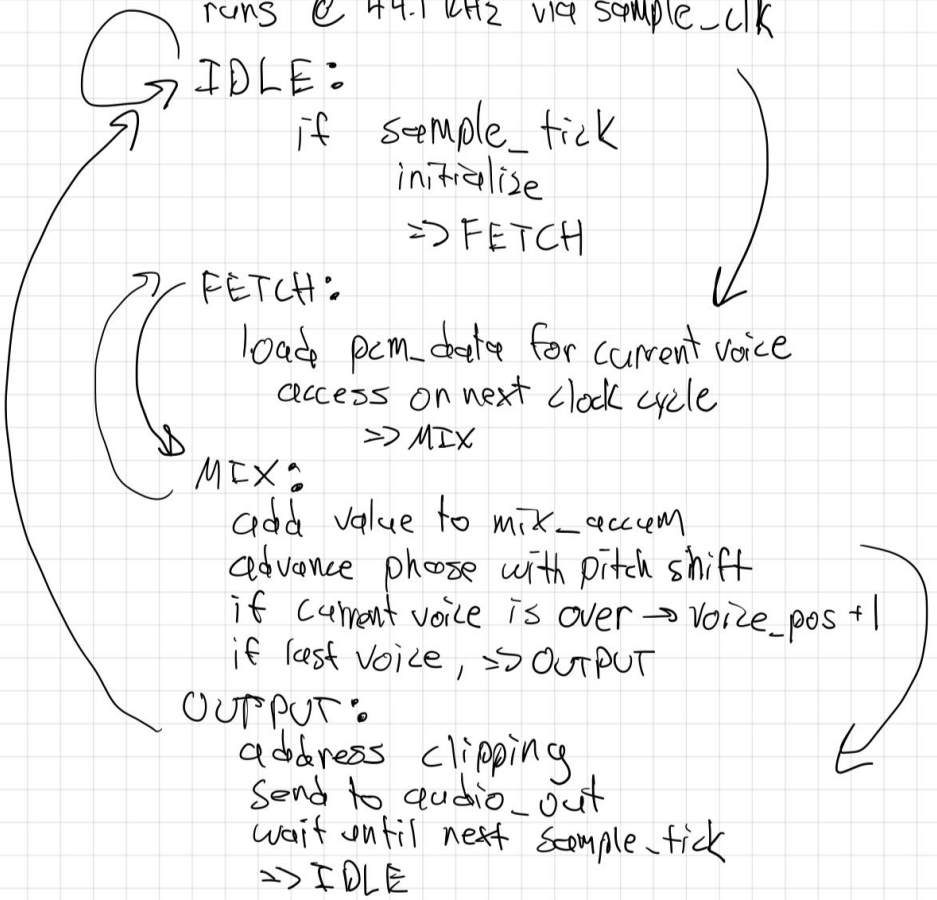


voice_t struct
For storing
writedata

```
typedef struct packed {

    logic active;
    logic [27:0] phase;
    logic [17:0] end_addr;
    logic [12:0] ratio;
    logic [6:0] gain;

} voice_t;

voice_t voices[0:NUM_VOICES-1];
```

Mixer State Machine

runs @ 44.1 kHz via sample_clk

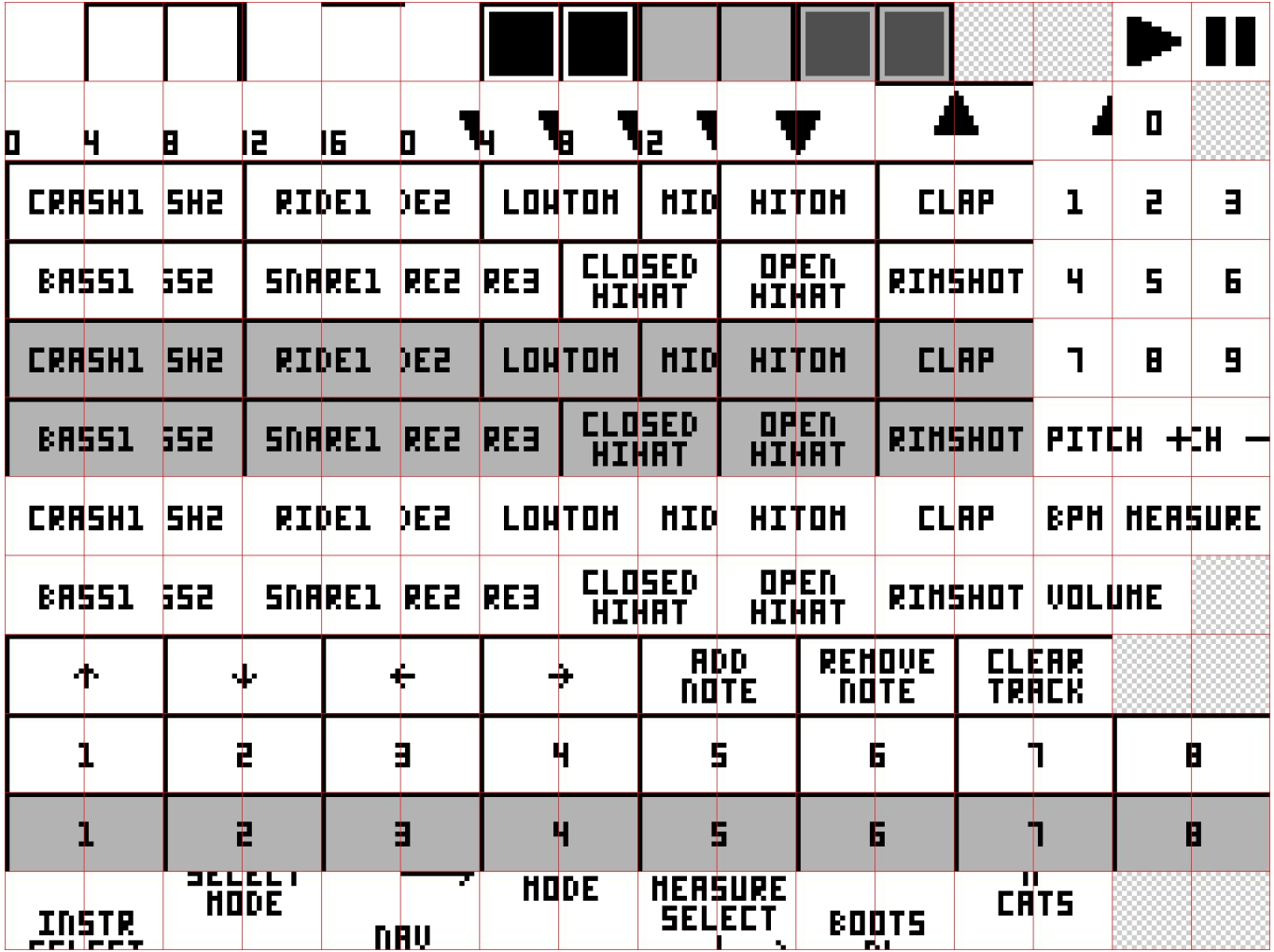


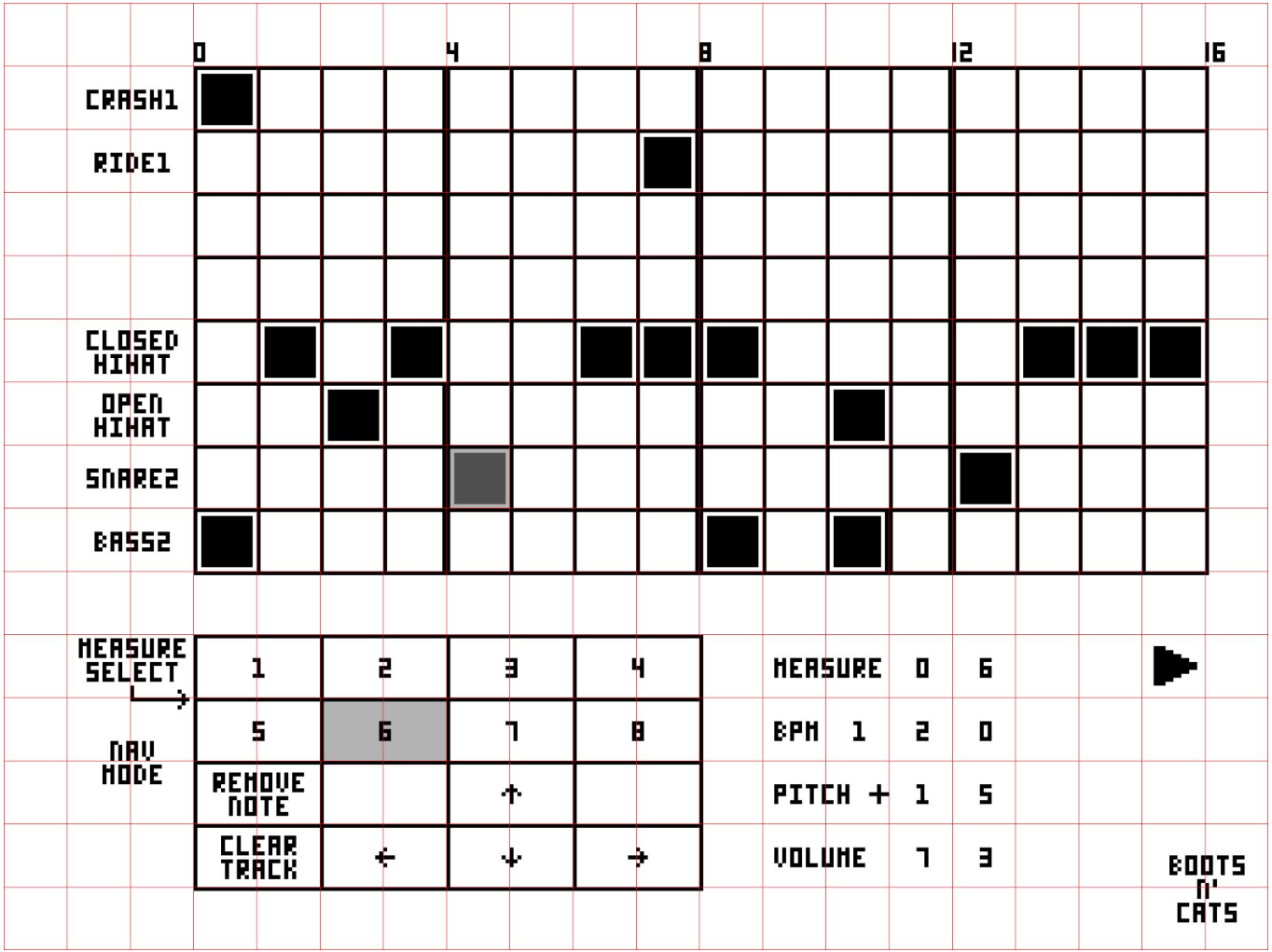
Tileset

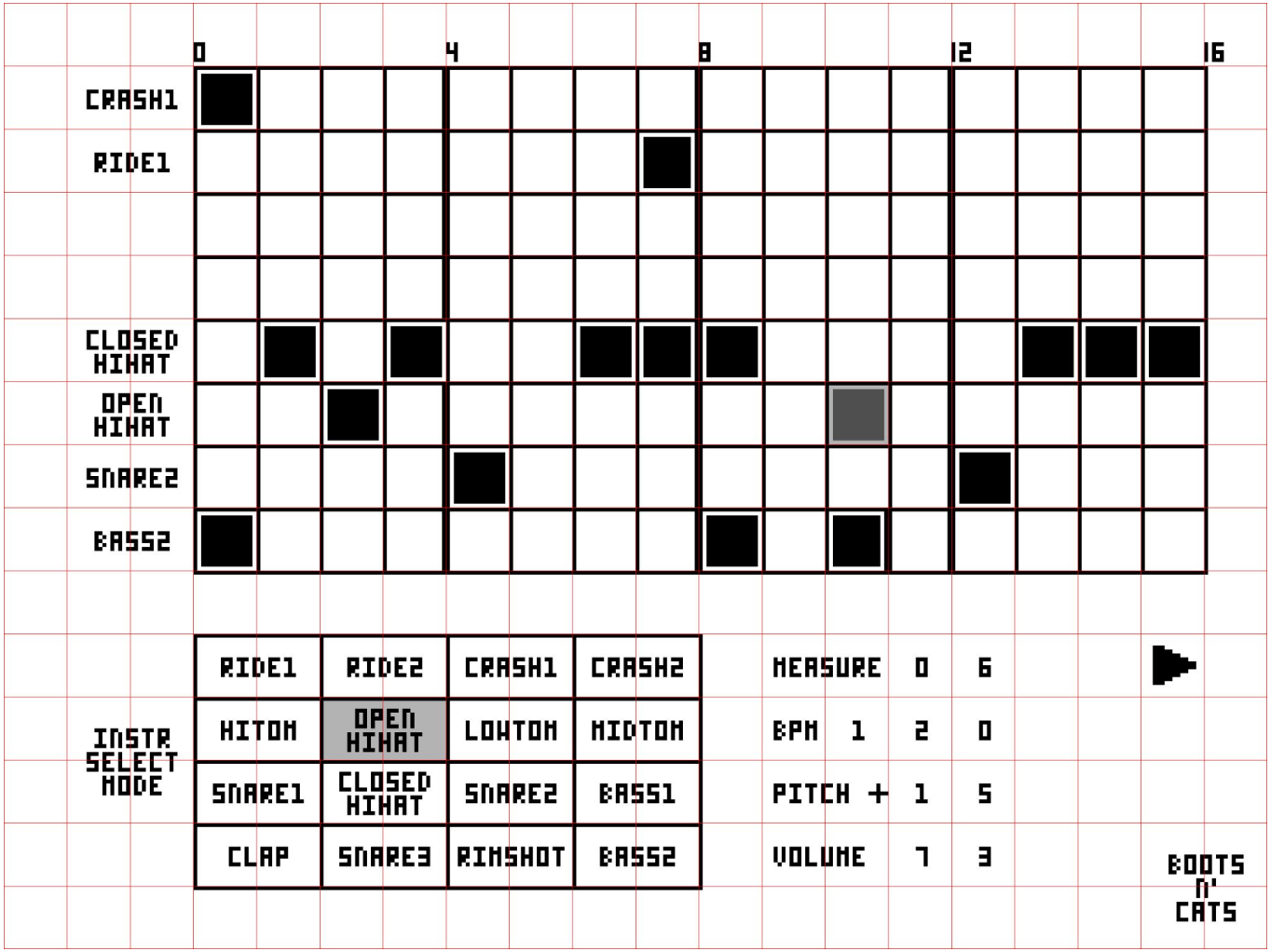
256 tile slots in total, but we are only using 192 of them

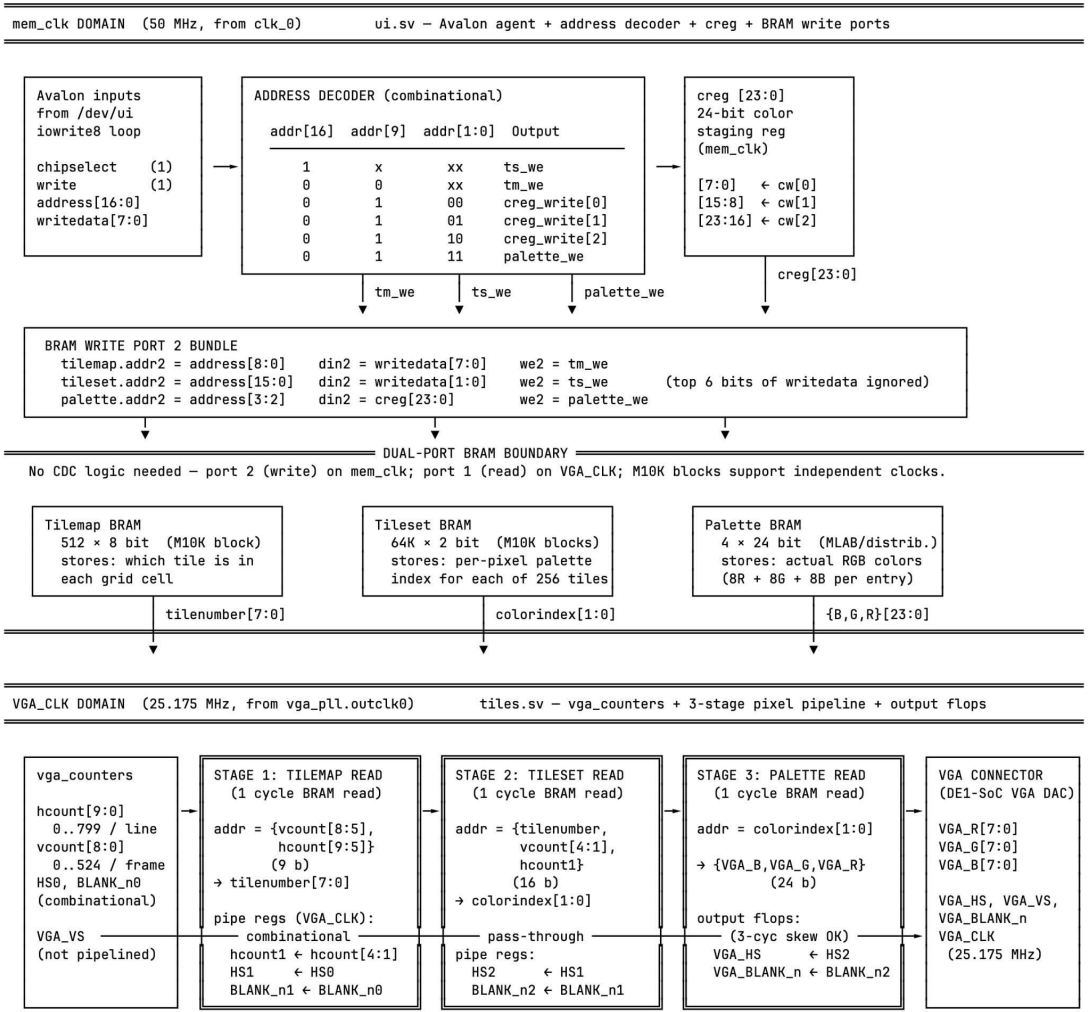
Each tile is 16×16 pixels, four possible colors (black, dark gray, light gray, white), so 2 bits per pixel

We display each pixel on each tile as a 2×2 macropixel on the VGA display, so effective resolution is 320×240 just “blown up” 2× (VGA full resolution is 640×480)




0
4
8
12
16
CRASH1
RIDE1
CLOSED HIHAT
OPEN HIHAT
SNARE2
BASS2
MEASURE SELECT
1
2
3
4
5
6
7
8
NAV MODE
REMOVE NOTE
CLEAR TRACK
MEASURE 0 6
BPM 1 2 0
PITCH + 1 5
VOLUME 7 3
BOOTS N' CATS


0
4
8
12
16
CRASH1
RIDE1
CLOSED HIHAT
OPEN HIHAT
SNARE2
BASS2
INSTR SELECT MODE
RIDE1
RIDE2
CRASH1
CRASH2
HITOM
OPEN HIHAT
LOWTOM
MIDTOM
SNARE1
CLOSED HIHAT
SNARE2
BASS1
CLAP
SNARE3
RIMSHOT
BASS2
MEASURE 0 6
BPM 1 2 0
PITCH + 1 5
VOLUME 7 3
BOOTS N' CATS


mem_clk DOMAIN (50 MHz, from clk_0)
ui.sv — Avalon agent + address decoder + creg + BRAM write ports
Avalon inputs
from /dev/ui
iowrite8 loop
chipselect (1)
write (1)
address[16:0]
writedata[7:0]
ADDRESS DECODER (combinational)
addr[16] addr[9] addr[1:0] Output
1 x xx ts_we
0 0 xx tm_we
0 1 00 creg_write[0]
0 1 01 creg_write[1]
0 1 10 creg_write[2]
0 1 11 palette_we
creg [23:0]
24-bit color
staging reg
(mem_clk)
[7:0] ← cw[0]
[15:8] ← cw[1]
[23:16] ← cw[2]
tm_we
ts_we
palette_we
creg[23:0]
BRAM WRITE PORT 2 BUNDLE
tilemap.addr2 = address[8:0] din2 = writedata[7:0] we2 = tm_we
tileset.addr2 = address[15:0] din2 = writedata[1:0] we2 = ts_we (top 6 bits of writedata ignored)
palette.addr2 = address[3:2] din2 = creg[23:0] we2 = palette_we
DUAL-PORT BRAM BOUNDARY
No CDC logic needed — port 2 (write) on mem_clk; port 1 (read) on VGA_CLK; M10K blocks support independent clocks.
Tilemap BRAM
512 × 8 bit (M10K block)
stores: which tile is in
each grid cell
Tileset BRAM
64K × 2 bit (M10K blocks)
stores: per-pixel palette
index for each of 256 tiles
Palette BRAM
4 × 24 bit (MLAB/distrib.)
stores: actual RGB colors
(8R + 8G + 8B per entry)
tilenumber[7:0]
colorindex[1:0]
{B,G,R}[23:0]
VGA_CLK DOMAIN (25.175 MHz, from vga_pll.outclk0)
tiles.sv — vga_counters + 3-stage pixel pipeline + output flops
vga_counters
hcount[9:0]
0..799 / line
vcount[8:0]
0..524 / frame
HS0, BLANK_n0
(combinational)
VGA_VS
(not pipelined)
STAGE 1: TILEMAP READ
(1 cycle BRAM read)
addr = {vcount[8:5],
hcount[9:5]}
(9 b)
→ tilenumber[7:0]
pipe regs (VGA_CLK):
combinational
hcount1 ← hcount[4:1]
HS1 ← HS0
BLANK_n1 ← BLANK_n0
STAGE 2: TILESET READ
(1 cycle BRAM read)
addr = {tilenumber,
vcount[4:1],
hcount1}
(16 b)
→ colorindex[1:0]
pass-through
pipe regs:
HS2 ← HS1
BLANK_n2 ← BLANK_n1
STAGE 3: PALETTE READ
(1 cycle BRAM read)
addr = colorindex[1:0]
→ {VGA_B,VGA_G,VGA_R}
(24 b)
output flops:
(3-cyc skew OK)
VGA_HS ← HS2
VGA_BLANK_n ← BLANK_n2
VGA CONNECTOR
(DE1-SoC VGA DAC)
VGA_R[7:0]
VGA_G[7:0]
VGA_B[7:0]
VGA_HS, VGA_VS,
VGA_BLANK_n
VGA_CLK
(25.175 MHz)

Name	Notes	Memory (bit)
pcm_samples.hex	3.546s, 44.1kHz, loaded in 16b pcm_mem registers	156,395
[31:0] sample_index [0:16]	Indexes for the starting samples of each instrument and final index of pcm_mem	544
[64:0] voices [0:63]	Registers to store data and parameters for each playing note. Up to 64 at a time. logic active; logic [27:0] phase; logic [17:0] end_addr; logic [12:0] ratio; logic [6:0] gain;	4160
Mixing regs	Keeping track of logic for the main mixer FSM logic [5:0] voice_pos; logic signed [31:0] pcm_data; logic signed [31:0] mix_accum; mixer_state_t state [1:0];	62
Tileset	256 tile slots, 16×16 pixels per slot, 2 bits per pixel (4 colors total) The actual tileset.bin uses 8 bits (1 byte) each **only at load time** for ease of addressing with (only slightly slows down startup), actual FPGA BRAM usage is 2 bits each with ts_din(writedata[1:0]) In practice we only use about 192 tile slots	131,072
Palette	8 bits per R/G/B (24 bits total), 4 colors total	96
(Live) Tilemap	32×16 tiles, 8 bits per tile (to address all 256 possible tile slots) Only 20×15 tiles are visible. (Actual displayed resolution is 320×240, we display each pixel as a 2×2 pixel on screen.) The extra tiles allow easy address conversion by bit-slicing VGA counters	4096
	Total memory (bit)	296425