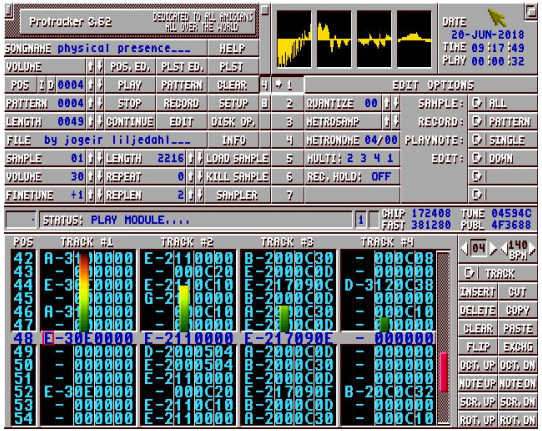
WaveSURFER

Wave Synthesis Using Real FPGA EngineeRing

Sunny Fang, Opalina Khanna, Harry Minsky

Background

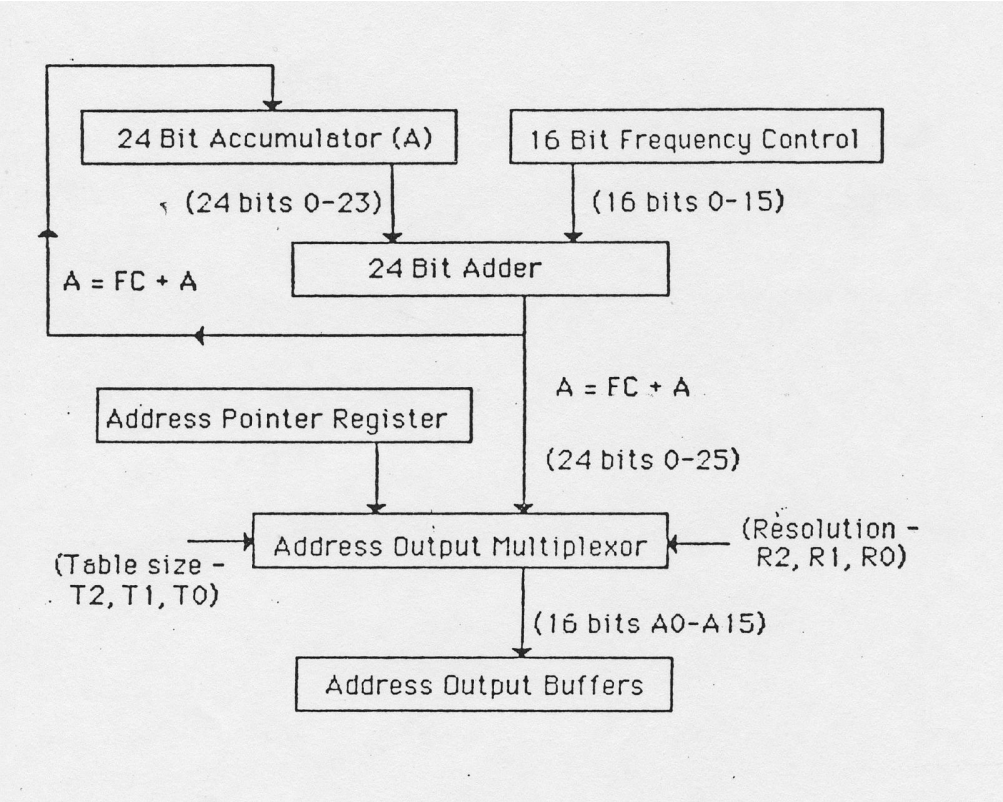
Yamaha GS1
(1980)



Amiga
ProTracker
(1990)

Synth I found
on the
internet
(today)

Ensoniq 5503 Digital Oscillator Chip



Ensoniq 5503 Digital Oscillator Chip

There are 32 oscillators in the chip. Each oscillator is controlled by a set of seven registers:

1. Frequency Low
2. Frequency High
3. Volume
4. Data Sample
5. Address Pointer
6. Control Register
7. Resolution/Table Size Registers

Environment

- Lab 3 was main jumping off point, used the configs + make files + soc_system_top module as our base

- Used linux socfpga 4.19.04 – rebuilt with ALSA drivers, otherwise about the same (as far as we know)

Originally using AKAI USB-MIDI Controller

Switched to M-Audio USB-MIDI keyboard

High-level Overview

1

loads the FPGA bitstream and Linux kernel from the SD card, & a kernel module (wave_synth.ko)

exposes synth peripheral as **/dev/wave_synth** for mmap

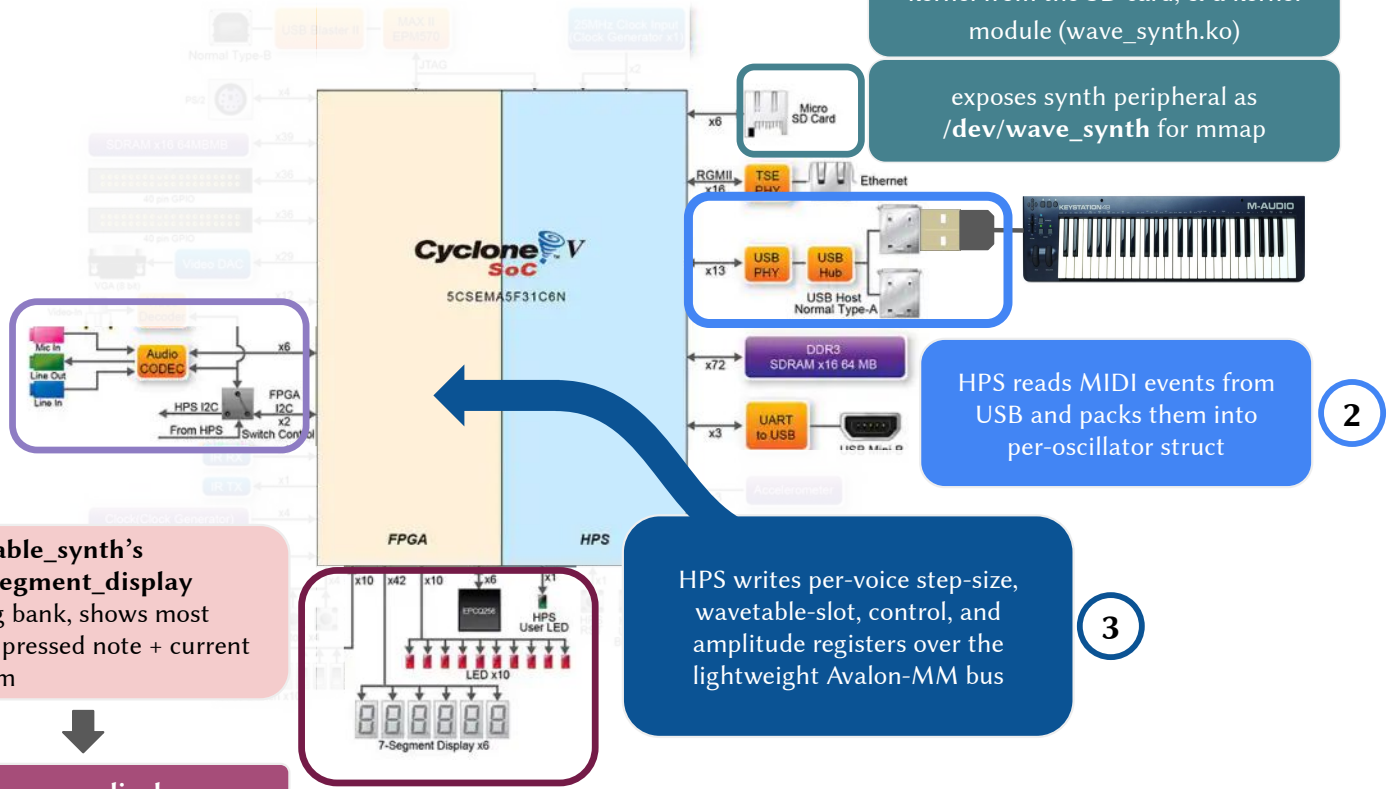
wave_table_synth's oscillator
looks up samples from a 4-slot BRAM wavetable, scales by per-voice amplitude, and sums into 48 kHz stereo stream

Wolfson WM8731 Audio CODEC

4

wave_table_synth's seven_segment_display
same reg bank, shows most recently pressed note + current waveform

seven-seg display



HPS reads MIDI events from USB and packs them into per-oscillator struct

2

HPS writes per-voice step-size, wavetable-slot, control, and amplitude registers over the lightweight Avalon-MM bus

3

LW AXI Bus Map

Region	Base Address	Address start -> end	Size	Purpose
audio_and_video_config_0	0xFF20_0000	0xFF20_0000 - 0xFF20_0010	16B	Auto-initializes the WM8731 codec
Wave_table_synth	0xFF28_0000	0xFF28_0000 - 0xFF2F_FFFF	512KB	Entire FPGA peripheral

HPS to FPGA Register Map

All data carried using mmap-ed lw axi bus

Region	Base Address	Address start -> end	Size	Purpose
Wavetable BRAM	0xFF28_0000	0xFF28_0000 - 0xFF2B_FFFF	256KB	Storing single audio wave periods
Per-Oscillator Ctrl	0xFF2C_0000	0xFF2C_0000 - 0xFF2C_00FF	512B	Controlling oscillators
Amp Ctrl	0xFF2C_0100	0xFF2C_0100 - 0xFF2C_0101	2B	Setting amplitude
Hex Ctrl	0xFF2C_0200	0xFF2C_0200 - 0xFF2C_020B	12B	Setting hex display

0x10000

Wavetable Register Map

32768*16/8/1024

Region	Base Address	Address start -> end	Size
Wavetable slot 1	`0xFF28_0000`	`0xFF28_0000 - 0xFF28_FFFF`	64KB
Wavetable slot 2	`0xFF29_0000`	`0xFF29_0000 - 0xFF29_FFFF`	64KB
Wavetable slot 3	`0xFF2A_0000`	`0xFF2A_0000 - 0xFF2A_FFFF`	64KB
Wavetable slot 4	`0xFF2B_0000`	`0xFF2B_8000 - 0xFF2B_FFFF`	64KB

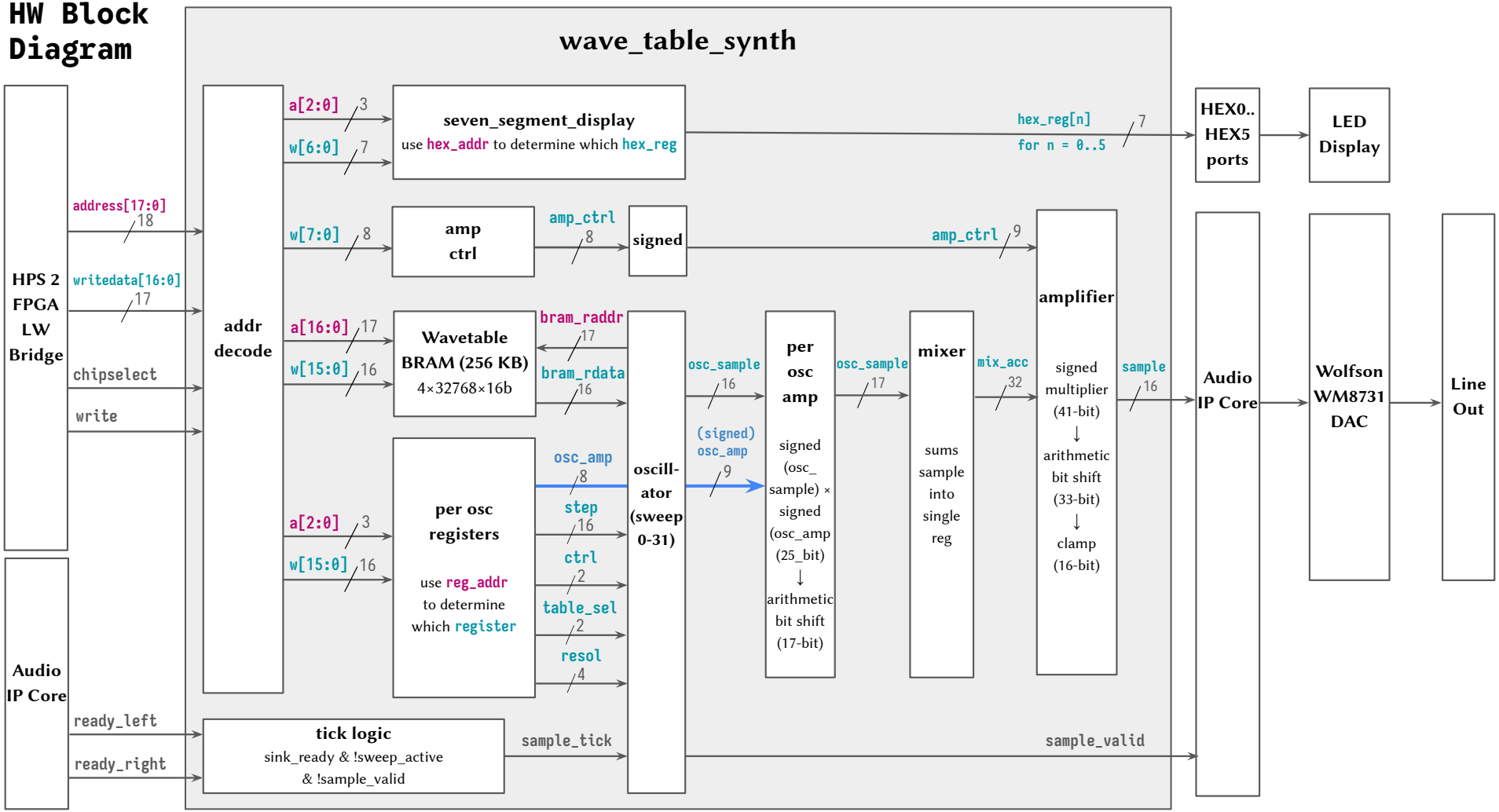
Quartus Specs

Flow Summary

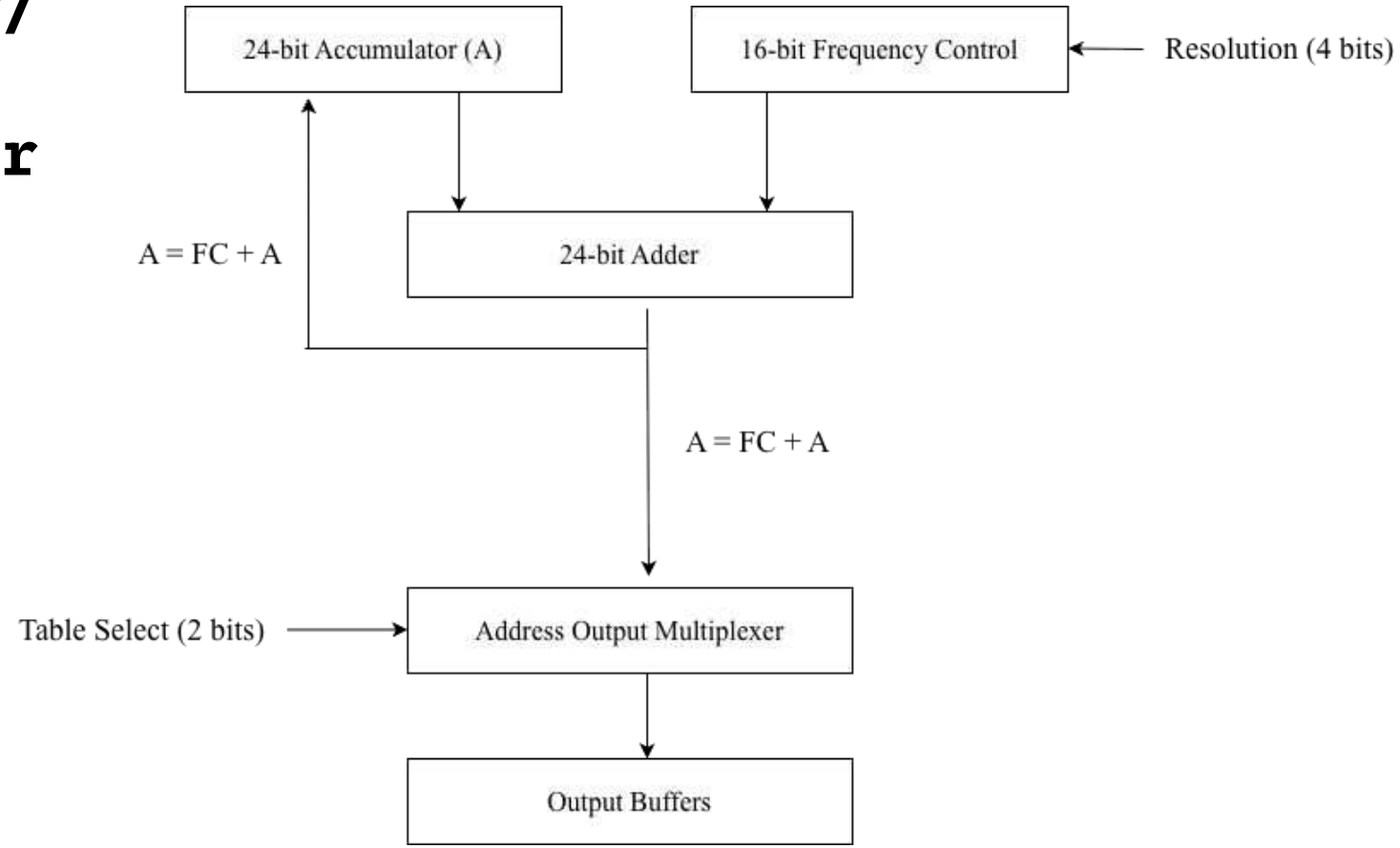
<<Filter>>

Flow Status	Successful - Tue May 12 22:30:36 2026
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	soc_system
Top-level Entity Name	soc_system_top
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	1,858 / 32,070 (6 %)
Total registers	2896
Total pins	362 / 457 (79 %)
Total virtual pins	0
Total block memory bits	2,101,248 / 4,065,280 (52 %)
Total DSP Blocks	3 / 87 (3 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 6 (17 %)
Total DLLs	1 / 4 (25 %)

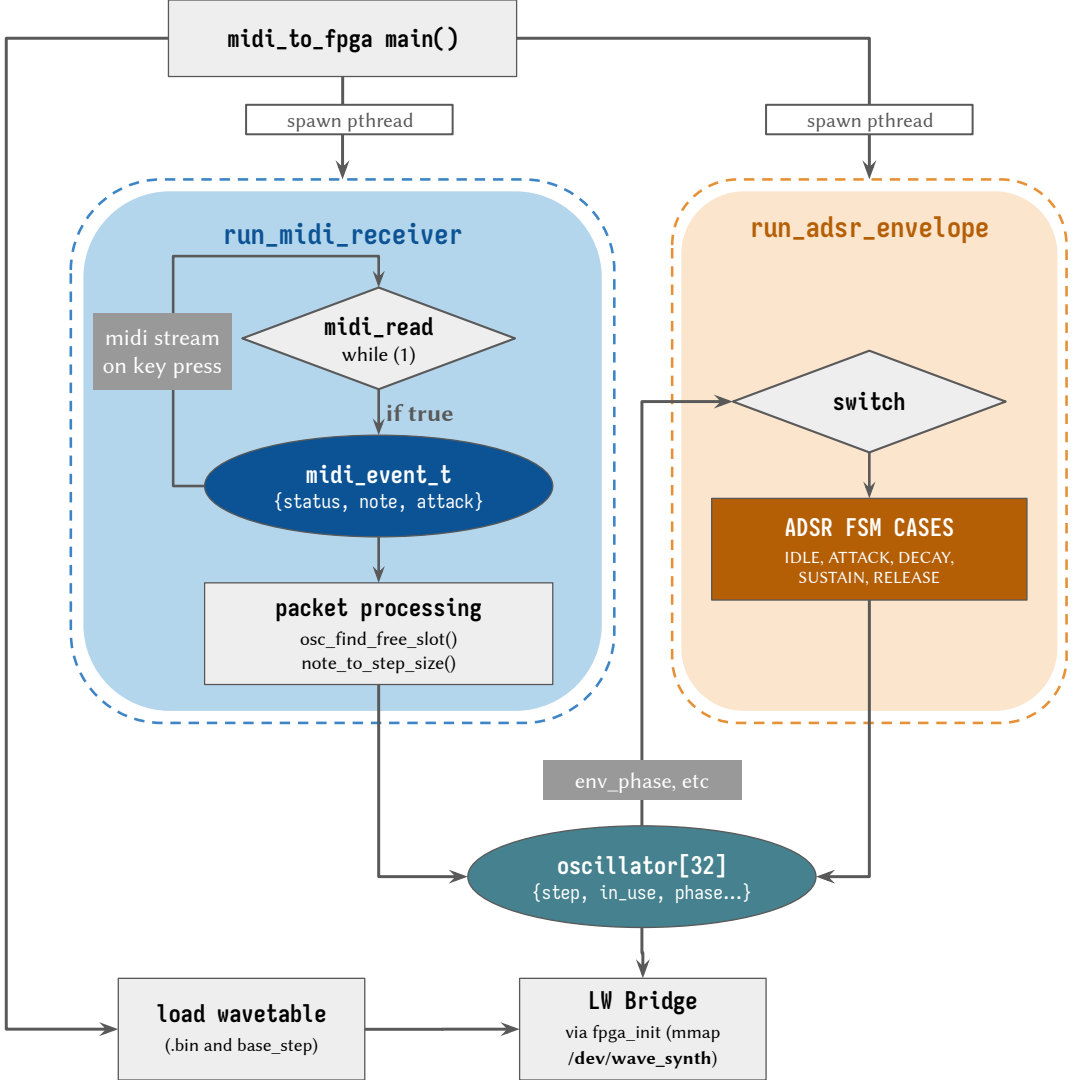
HW Block Diagram



Oscillator/ Phase Accumulator



SW Data Flow

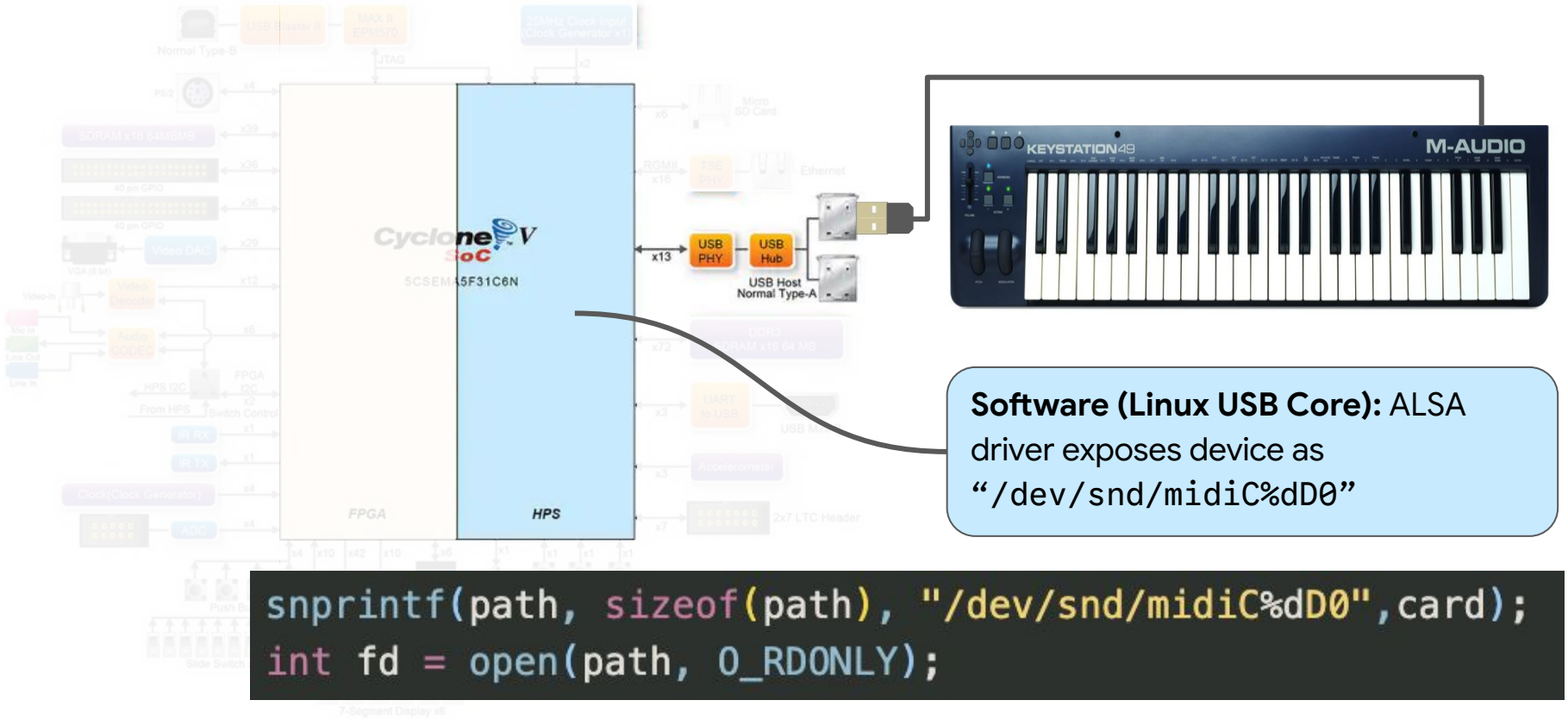

midi_to_fpga main()
spawn pthread
spawn pthread
run_midi_receiver
midi stream on key press
midi_read
while (1)
if true
midi_event_t
{status, note, attack}
packet processing
osc_find_free_slot()
note_to_step_size()
run_adsr_envelope
switch
ADSR FSM CASES
IDLE, ATTACK, DECAY, SUSTAIN, RELEASE
env_phase, etc
oscillator[32]
{step, in_use, phase...}
load wavetable
(.bin and base_step)
LW Bridge
via fpga_init (mmap /dev/wave_synth)

Custom Device Driver

```
static int wave_synth_mmap(struct file *filp, struct vm_area_struct *vma){
    unsigned long len = vma->vm_end - vma->vm_start;
    unsigned long dev_size = resource_size(&dev.res);

    if(vma->vm_pgoff != 0){
        return -EINVAL;
    }

    if(len > dev_size){
        return -EINVAL;
    }

    // necessary so that read/writes to this portion of memory are never using
    // cache
    vma->vm_page_prot = pgprot_noncached(vma->vm_page_prot);

    if(io_remap_pfn_range(vma, vma->vm_start, dev.res.start >> PAGE_SHIFT,len, vma->vm_page_prot)){
        return -EAGAIN;
    }
    return 0;

}
```

```
    wave_table_synth_0: synth@0x100080000 {
        compatible = "csee4840,wave_table_synth-1.0";
        reg = <0x00000001 0x00080000 0x00080000>;
        clocks = <&clk_0>;
    }; //end synth@0x100080000 (wave_table_synth_0)
}; //end bridge@0xc0000000 (hps_0_bridges)
```

ALSA Driver



Software (Linux USB Core): ALSA driver exposes device as “`/dev/snd/midiC%dD0`”

```
snprintf(path, sizeof(path), "/dev/snd/midiC%dD0",card);
int fd = open(path, O_RDONLY);
```

wavetable.bin

4 B magic "WTSY"
2 B version (=1)
2 B num_metadata_slots (=4)
4 B per slot × 4 slots:
  2 B base_step
  1 B resolution
  1 B reserved (=0)
	24-byte header

32768 × int16 (LE) — slot 0 audio	wave tables
32768 × int16 (LE) — slot 1 audio	
32768 × int16 (LE) — slot 2 audio	
32768 × int16 (LE) — slot 3 audio	

Core Formula: **Note to Frequency**

Algorithm 1 Note to frequency conversion

1: **input:** note n `// midi_packet.note`

2: **output:** frequency s

3: $f = 440.0 * pow(2.0, (note - 69) / 12.0);$ `// used to find step_size`

equal temperament tuning formula

Core Formula: **Frequency to step size**

Algorithm 2 Frequency to Step_size Conversion

1: **input:** frequency f, table_size t, sample_rate f_s, phase_acc_size p
2: **output:** step_size s
3: $m \leftarrow log2(t)$ ▷ num of bits needed to represent all samples in the table
4: $x \leftarrow p - m$ ▷ fractional bits
5: $s \leftarrow \text{round}((f \times 2^t \times 2^x)/f_s)$ ▷ fixed-point representation of step size in Qm.x format

(the actual code abstracts this away from the calculation; e.g., f × 65536 / 48000)

```
// in sw, we send a 16-bit step_size s to hw
// in hw, we concatenate 8-bits (≡ multiply by 2^8)
```

Attack, Decay, Sustain, Release