Sample-Based MIDI Synthesizer

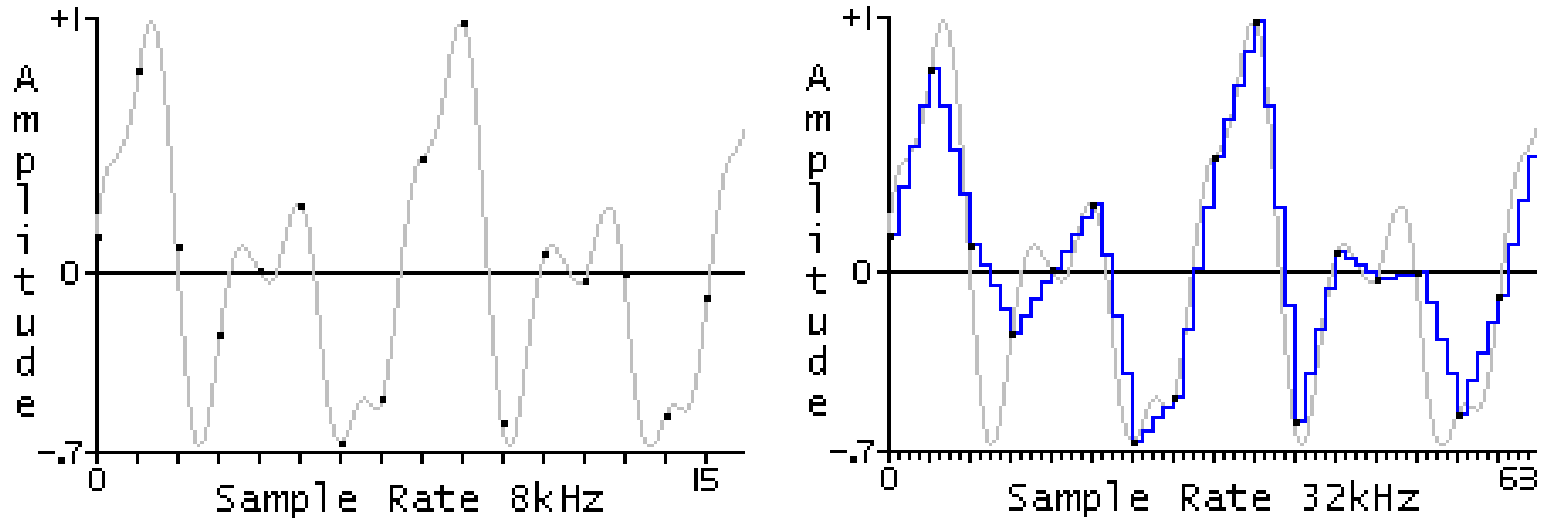
Andy Leiserson and Amir Hirsch

Sample Based Synthesis

- Sound Source is recorded sounds
- Playback at different rates to change pitch
  - rate ratio = $f_{playback} / f_{record}$
  - causes the entire spectrum to shift
- Filters to alter timbre

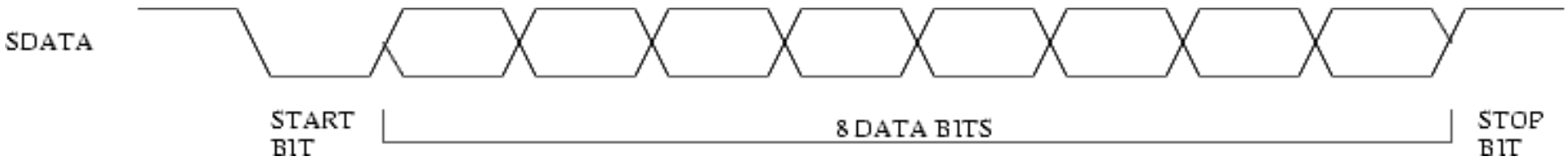
Interpolation

- Calculate Continuous Time Values of Discrete Time Signals
- Aliasing Results from Interpolation Error
- Band Limited Interpolation (sin x / x)
- Linear Interpolation
  - Weighted Sum of Two Surrounding samples

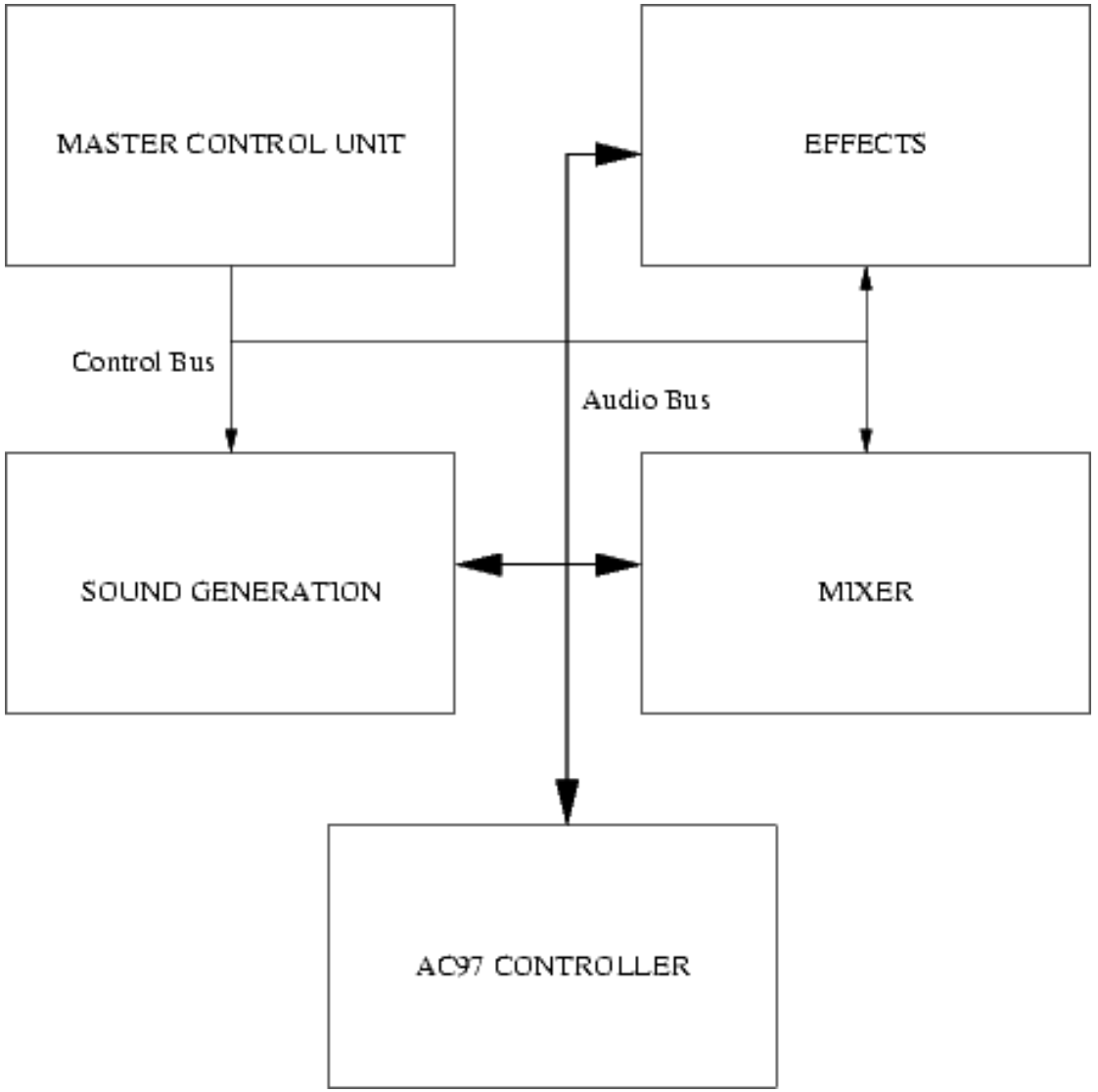


MIDI

- Musical Instrument Device Interface
- Standard interface for electronic musical instruments
- Messages such as Note On and Note Off
- Information is transmitted serially at 31.25 kilobits per second

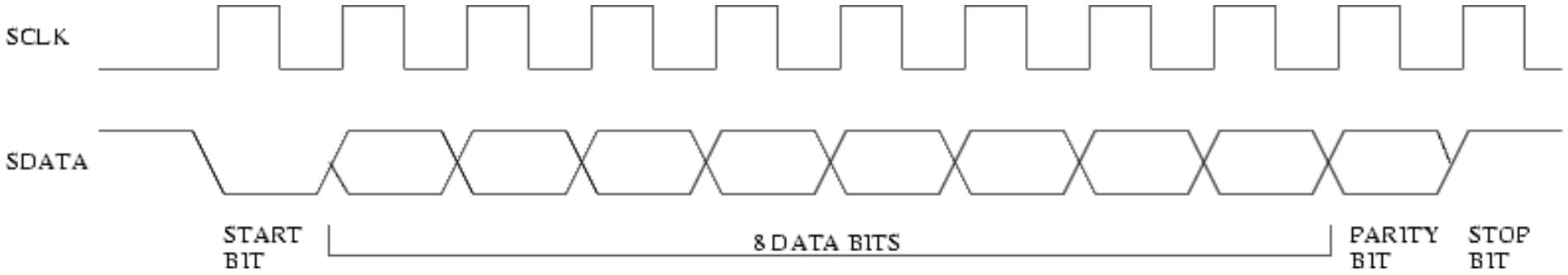


Our Implementation



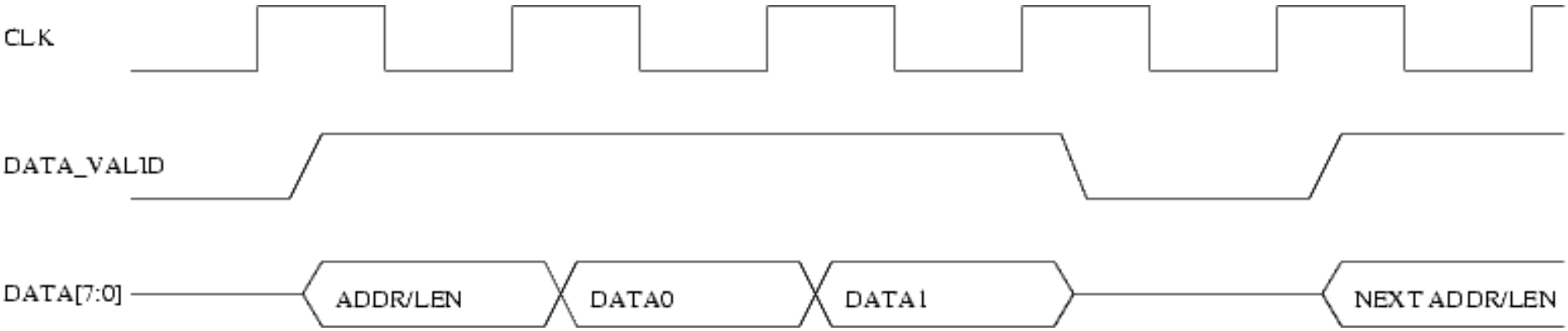
Master Control Unit

- Receives and processes MIDI messages
- Contains the Keymap (maps notes to a sound and frequency)
- Loads patch data from ROM
- Controls all other units
- Receives PS/2 input from keyboard
  - Sampled on falling edge
  - Data transmitted at 10 – 16 kilobits per second



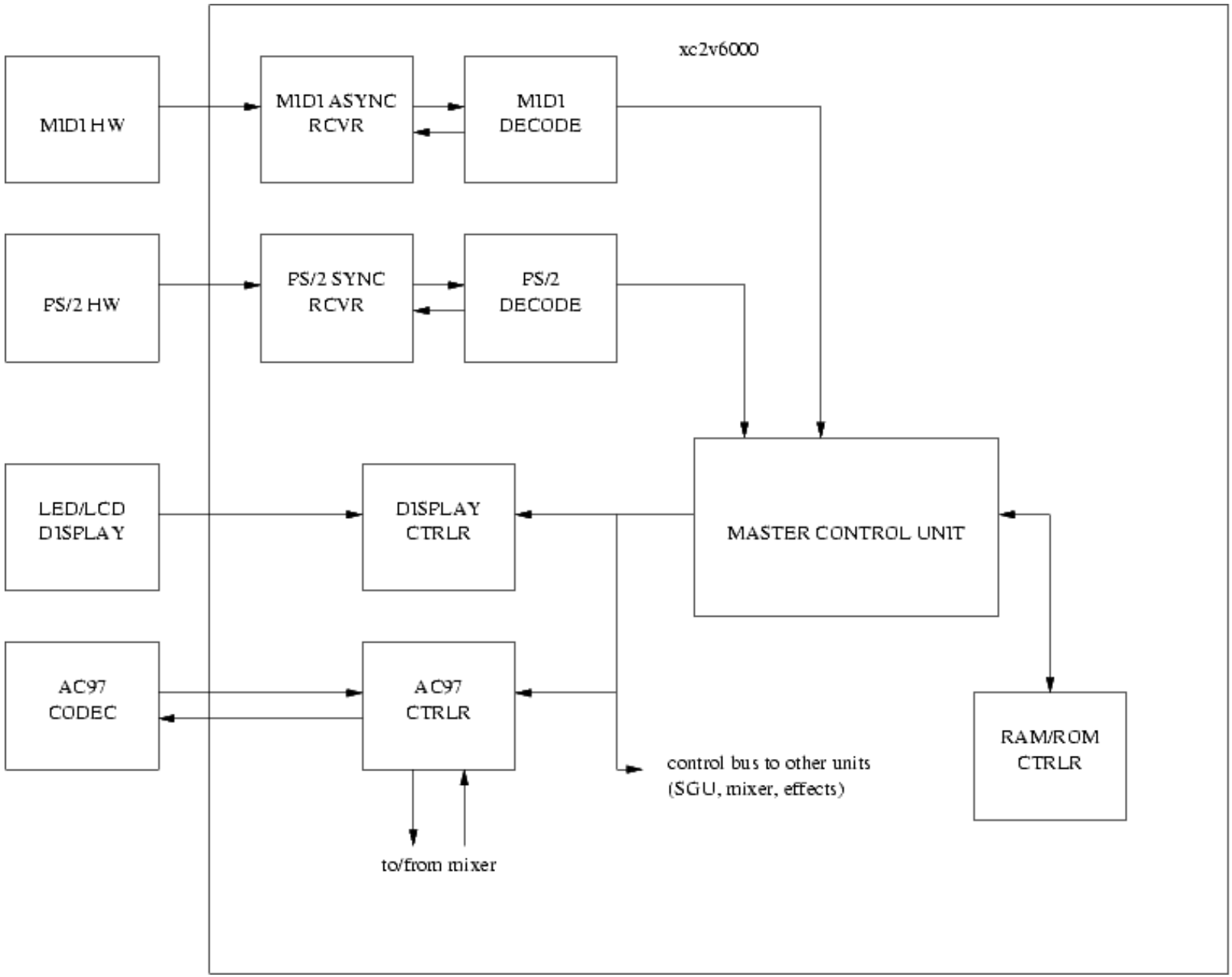
Message Passing

- Control Unit Generates Messages



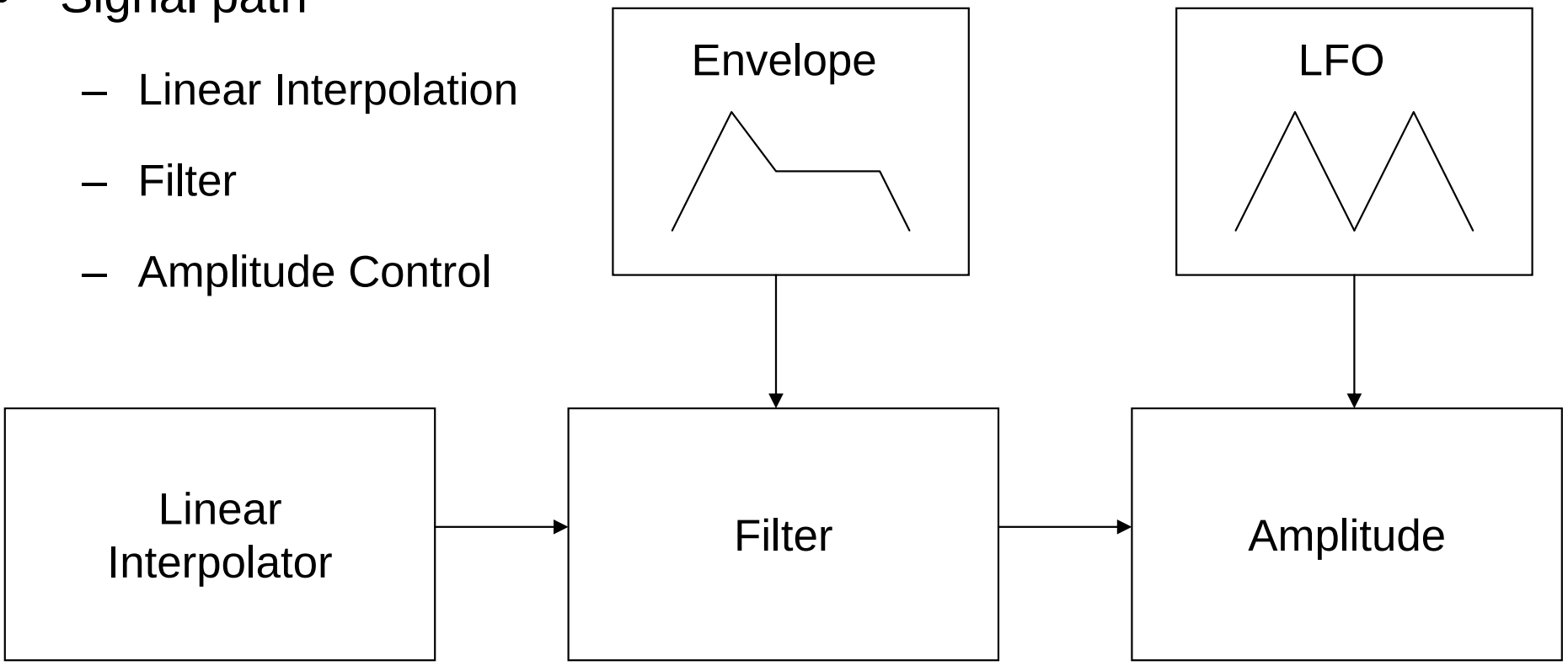
- First byte contains the recipient ID and the message length
- Data_Valid stays high for the duration of the message
  - When it goes high all units need to listen to the first byte
  - If the ID does not match the unit ID they can ignore the rest
  - Multiple bytes are sent Big Endian (determined by a coin flip)

MCU Block Diagram



Sound Generation Unit

- Controls the processing of audio
- Signal path
  - Linear Interpolation
  - Filter
  - Amplitude Control

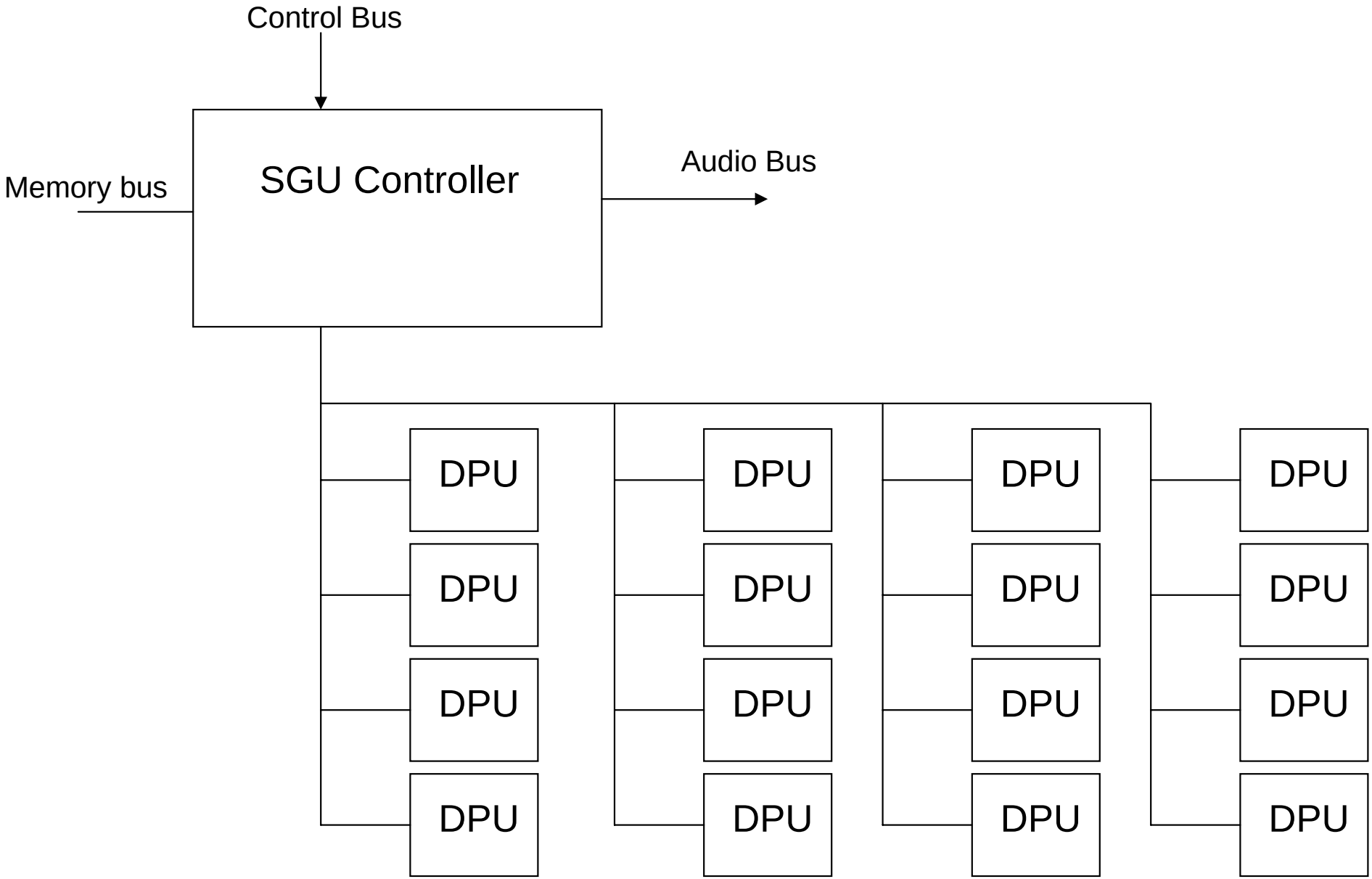


Filter

- Four Cascaded 6 dB/oct IIR filters

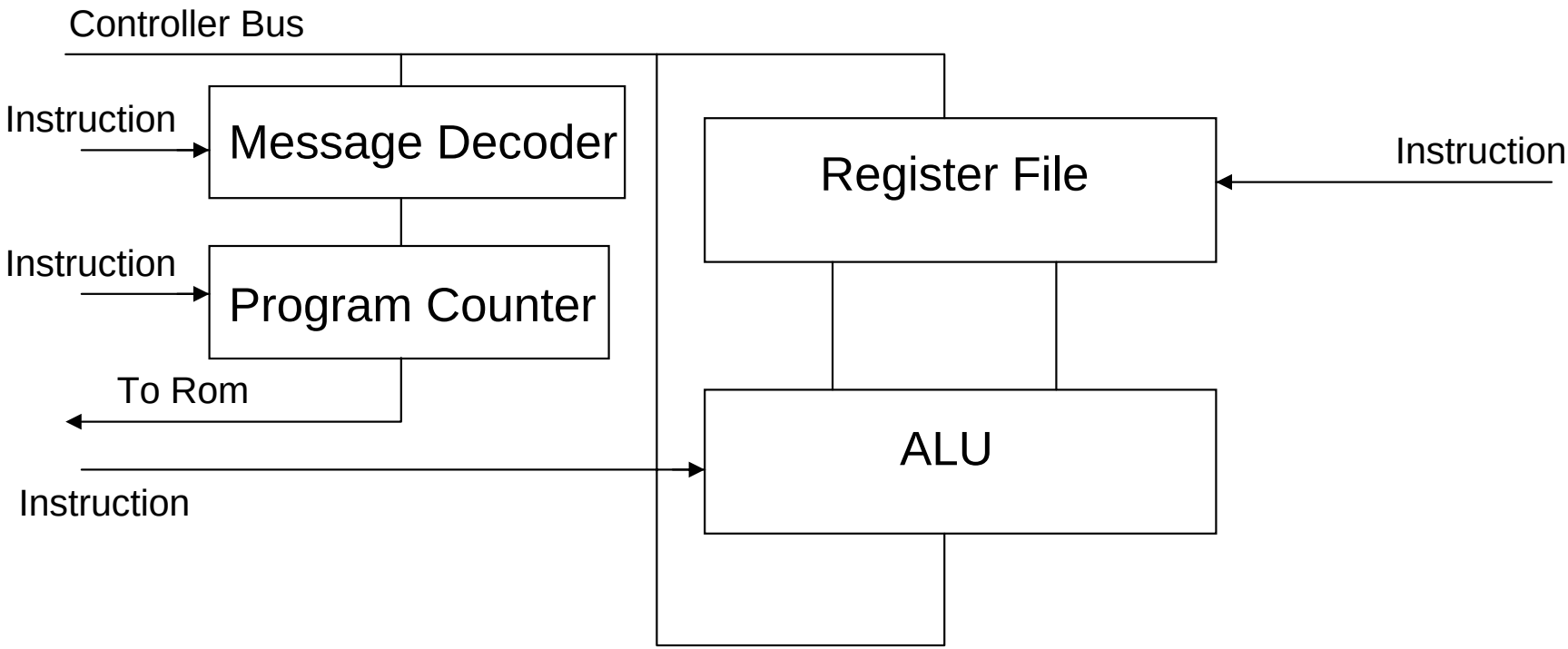
```
// Set coefficients given frequency & resonance [0.0...1.0]

  q = 1.0 - frequency;
  p = frequency + 0.8 * frequency * q;
  f = p + p - 1.0;
  q = resonance * (1.0 + 0.5 * q * (1.0 - q + 5.6 * q * q));

// Filter (in [-1.0...+1.0])

  in -= q * b4;                                  //feedback
  t1 = b1;    b1 = (in + b0) * p - b1 * f;
  t2 = b2;    b2 = (b1 + t1) * p - b2 * f;
  t1 = b3;    b3 = (b2 + t2) * p - b3 * f;
                   b4 = (b3 + t1) * p - b4 * f;
              b4 = b4 - b4 * b4 * b4 * 0.166667;    //clipping
  b0 = in;

// Lowpass  output:  b4
// Highpass output:  in - b4;
// Bandpass output:  3.0 * (b3 - b4);
```

SGU Block Diagram



Data Processing Unit

- Datapath and a ROM
  - ROM contains microcode for Filters, Envelopes, Interpolation, etc.
- Receives instructions from controller to perform function



SGU Processing Order

- Dump previous outputs to mixer (does not require DPU)
- Compute envelopes and LFO’s
- Perform interpolation for every note
- Filter every note
- Get samples for next cycle from memory
- Process all messages from MCU

Mixer

- Inputs for each MIDI Channel, Line In and Effects Return
- Controls volume level and Pan for each channel
- Auxiliary send to effects units for each channel
- Sub-output to record audio
- Mixer Controller
  - Receives control signals from MCU and audio signals
  - Determines input values is for each channel
- DPU computes the Auxilary, Sub and Main outputs