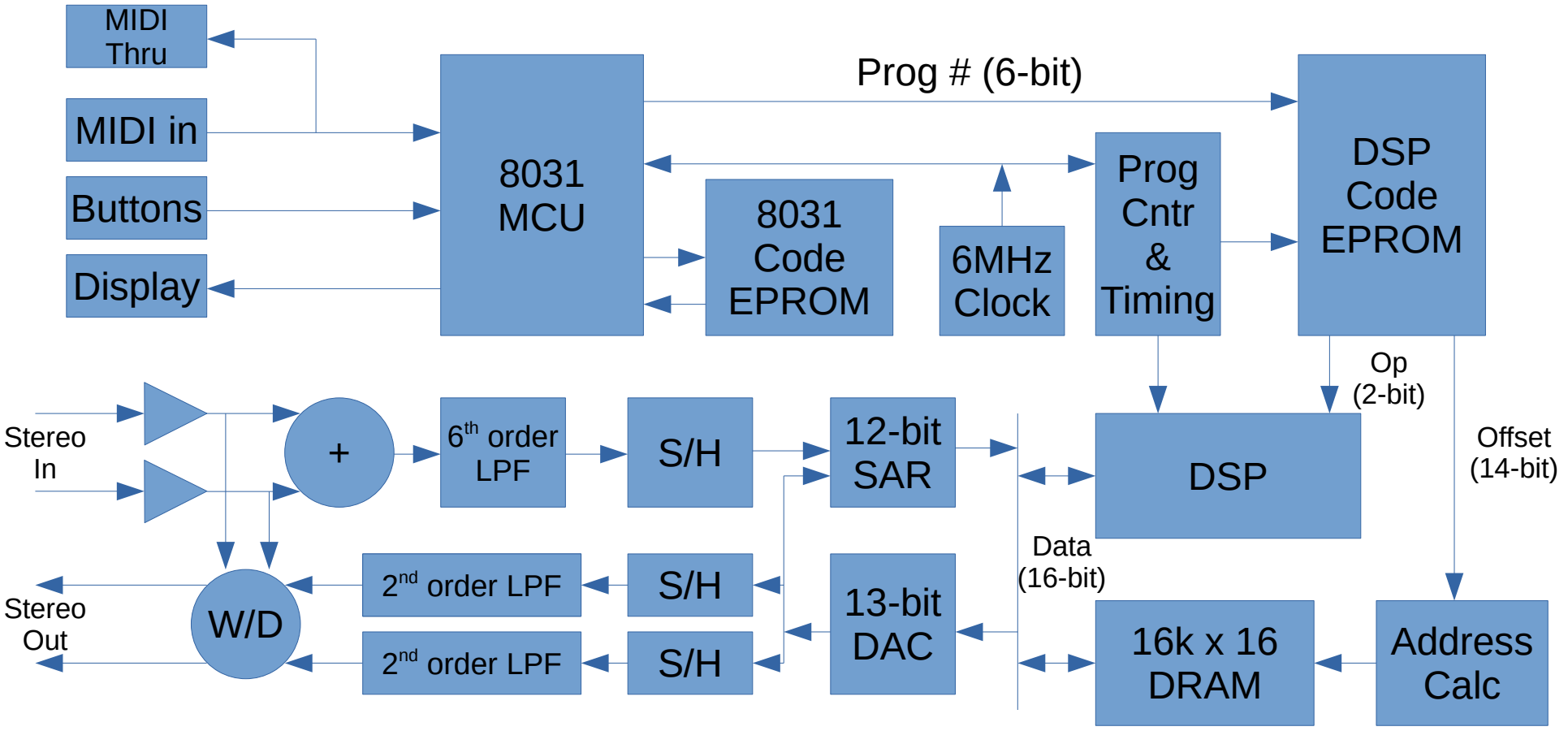
Midiverb Analysis

- Designed in mid-1980s with commonly available components:
  - LSTTL
  - Dynamic RAM
  - 27-series EPROM
  - 8031 MCU
  - 12-bit DAC
  - Jellybean linear op-amps

Midiverb Top Level



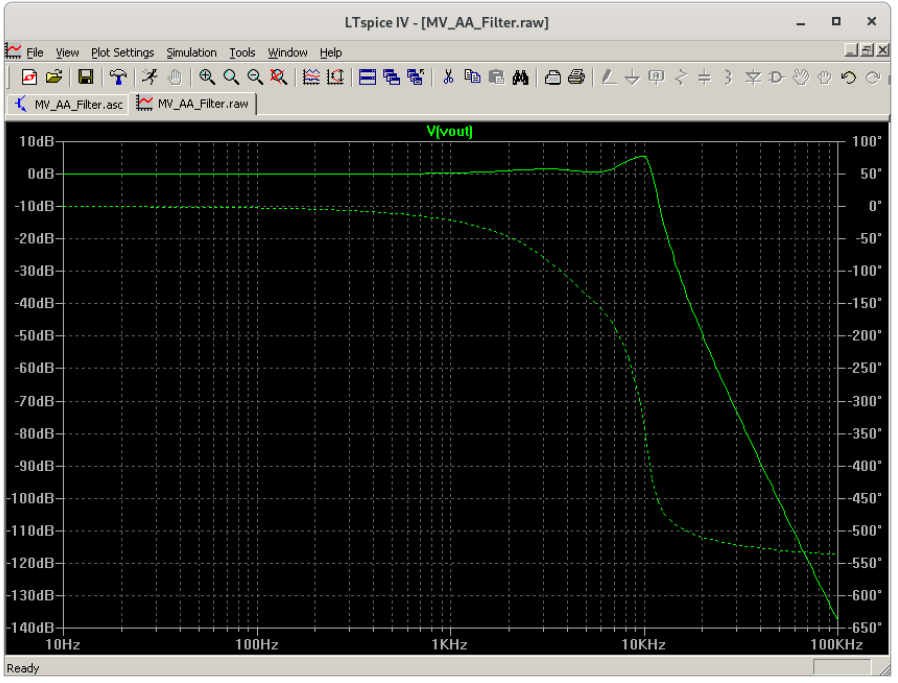
8031 Operation

- Very simple program that just manages the MIDI input, buttons & LED display to select the reverb program
  - Final output is the 6 MS-bits of the DSP microcode address.

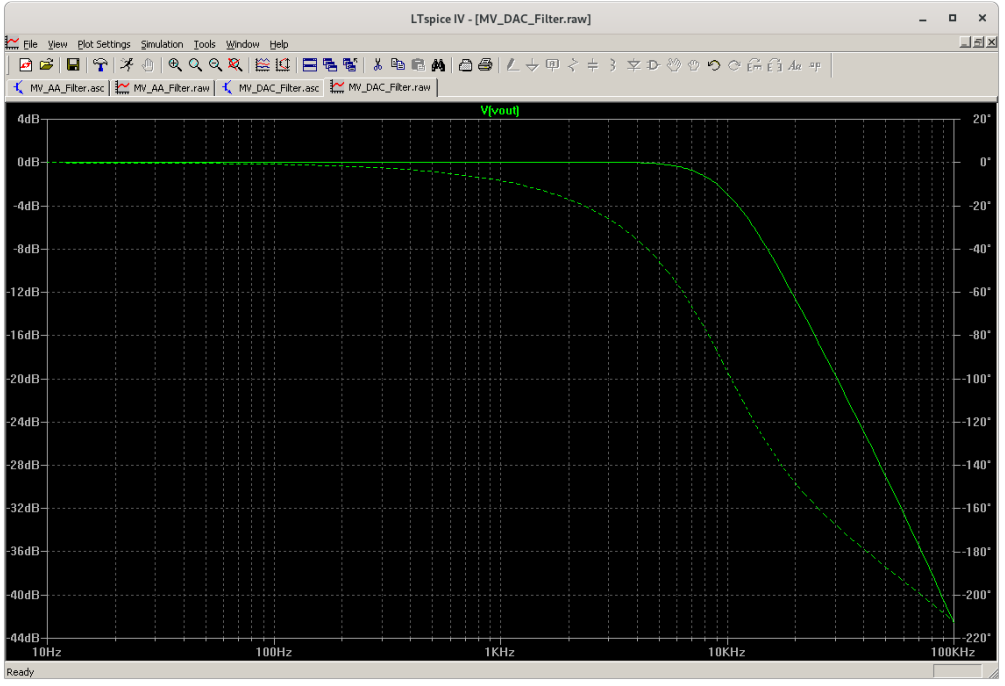
Timing

- 6MHz system clock
- Sample rate: 6MHz/256 = 23.4kSPS
- DSP Instruction rate: 3MIPS
  - Microcode read @ 6MBps
  - 2 bytes / instruction
    - 1st byte: 8-bit row address offset
    - 2nd byte: 2-bit OP + 6-bit column address offset
  - DSP Accum clocked @ every instruction
  - DRAM Reads or Writes @ every instruction
  - DRAM Address is adjusted @ every instruction
  - 128 instructions per audio sample
    - Mono input
    - Stereo output

Input Anti-Alias Filter Response



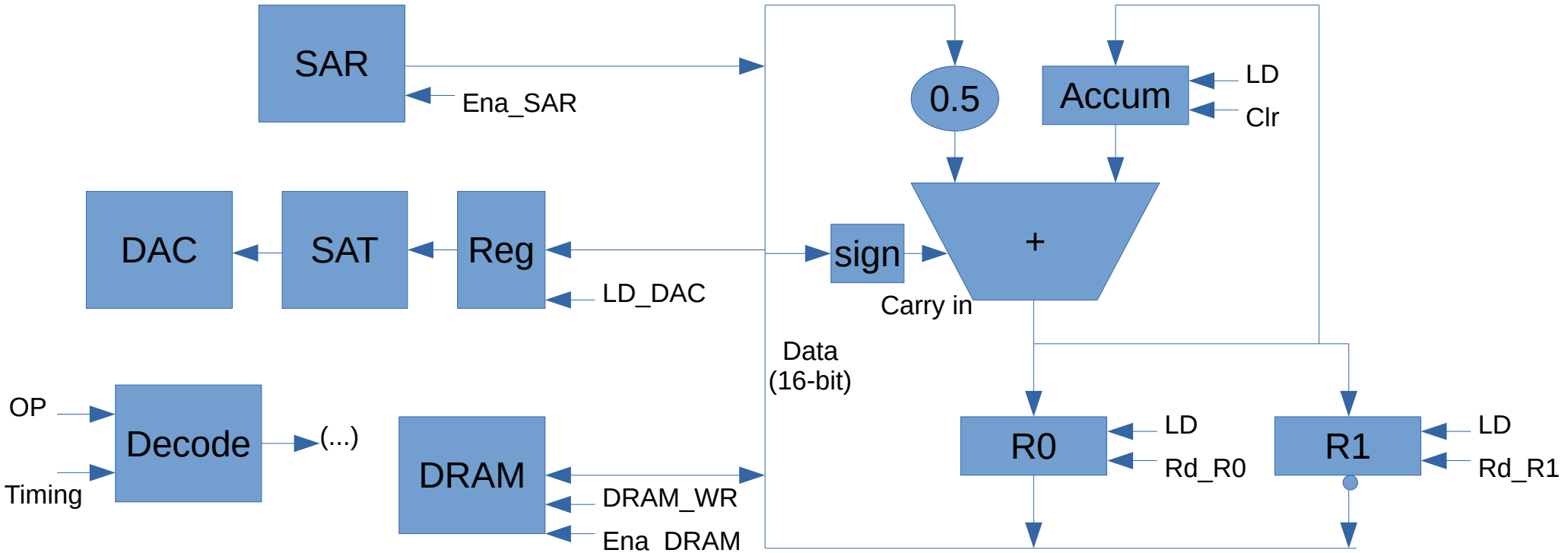
Output Reconstruction Filter Response



Instruction Format

- 16-bits per instruction
  - 2 bits operation, 14 bits address offset
- Low-endian, byte addressed
  - 8-bits row address in byte 0
  - 2-bit operation + 6 bits column address in byte 1
- Pipelined
  - Operation bits lead address bits by one cycle

DSP Detail


SAR
Ena_SAR
0.5
Accum
LD
Clr
DAC
SAT
Reg
LD_DAC
sign
+
Carry in
Data
(16-bit)
OP
Decode
(...)
Timing
DRAM
DRAM_WR
Ena_DRAM
R0
LD
Rd_R0
R1
LD
Rd_R1

DSP Operations

OP	/Clr	/Rd_R0	/Rd_R1	/DRAM_W	Function
00	1	1	1	1	Acc = Acc + DRAM[addr]/2 + sgn
01	0	1	1	1	Acc = DRAM[addr]/2 + sgn
10	1	0	1	0	DRAM[addr] = Acc, Acc = Acc + Acc/2 + sgn
11	0	1	0	0	DRAM[addr] = ~Acc, Acc = ~Acc/2 + sgn

- Four total instructions:
  - 00 - Sumhlf
  - 01 - Ldhlf
  - 10 - Strpos
  - 11 - Strneg
- Accumulator is either summed with or loaded with (Data/2 + sgn)
- Writes to DRAM are either Accumulator or binary inverse of Accumulator

Program Execution

- No “hard reset”
  - Program execution begins at arbitrary address
    - After power-on
    - After program change
  - DSP contents must “settle out”
    - Wait for DRAM buffer to fill with valid data
    - Noise burst at start up / change
    - No muting in hardware
- No branching
  - Fixed deterministic execution path
  - “Big Loop” of 128 instructions

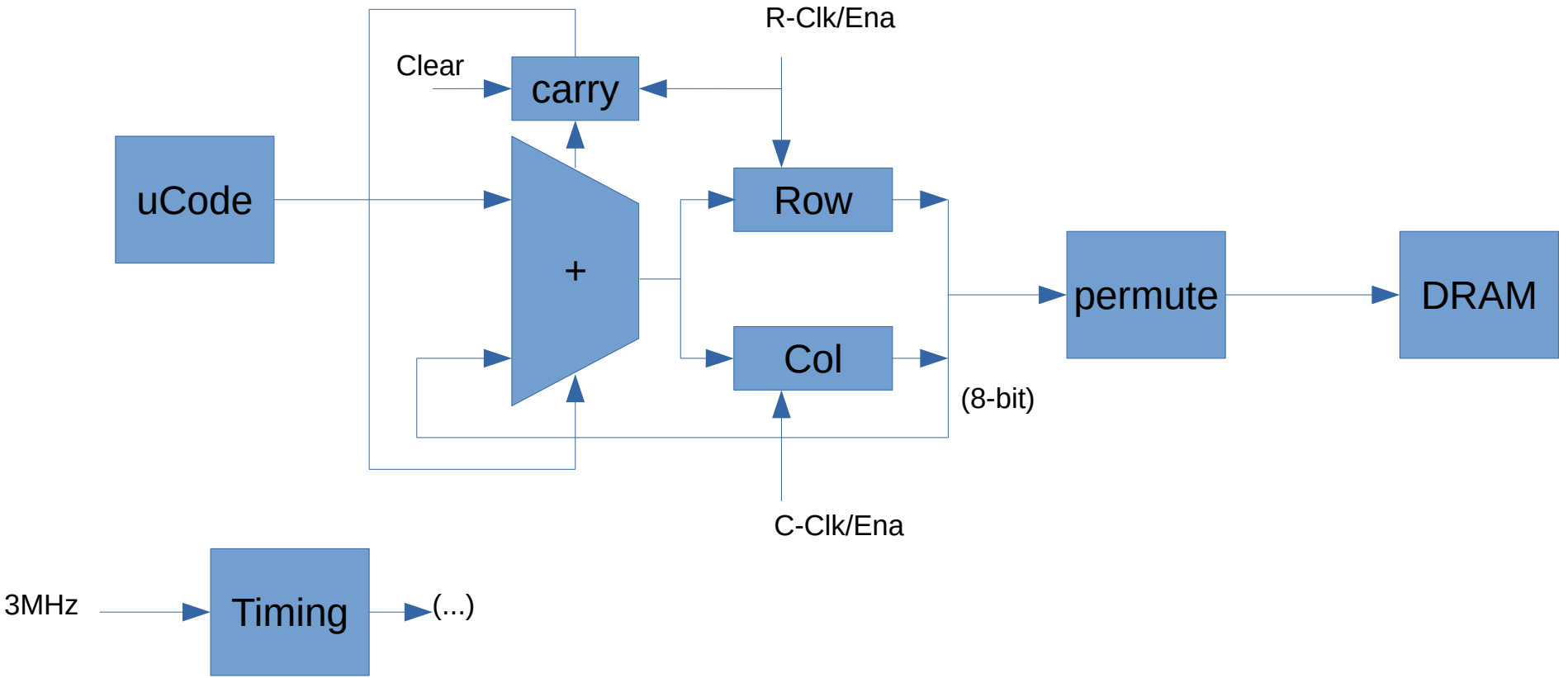
Fixed Address Functions

- Certain pre-defined data moves are hard-coded in the timing:
  - Addr 0x00 – Read from SAR
    - ADC result is forced onto data bus and DRAM write is asserted, writing new sample data into the DRAM address. Accumulator is NOT updated.
  - Addr 0x60 – Write to DAC + S/H R channel
    - Usually this coincides with a DRAM read and is often followed by an instruction which adjusts the address register back to what it had been. Accumulator is NOT updated.
  - Addr 0x70 – Write to DAC + S/H L channel
    - Usually this coincides with a DRAM read and is often followed by an instruction which adjusts the address register back to what it had been. Accumulator is NOT updated.

Digital/Analog interface

- “13-bit” Input
  - 12-bit SAR
  - Left shifted 1 bit & Sign-extended to 16-bits DSP data
    - Range = -4096 to + 4094
    - Left shift since 1st operation in DSP is to right shift 1 bit.
- “13-bit” Output
  - 12-bit DAC + “faked out ls-bit” with 10M resistor pulling on DAC output
  - 16-bit DSP data is saturated to 13-bits
    - Range = -4096 to +4095
    - Uses Tri-state disable and resistor pulls to force saturation values
      - Only top 8 bits of 13 are pulled – low 5 bits float (undriven TTL inputs usually resolve to ‘1’)

Address Detail



Address Calcs

- Two 8-bit registers for Row/Column address
- Common 8-bit adder with pipelined carry
  - Carry bit is cleared prior to Row address calculation
- Address bits permuted so 2 unused bits in Column address match up with OP bits in microcode.
- 8-bit row address comes first, then 6-bit column address. OP bits come from previous instruction.
- All address offsets in program must sum up to a value that walks thru the total number of RAM locations in the buffer when accumulated modulo 16k.
  - System strides through 16k DRAM at increments of 1, ie: contents of Row/Column registers at end of program is one greater than at start of program.
  - Allows simple tapped delay structures – data written into offset N will appear X samples later at offset N-X.

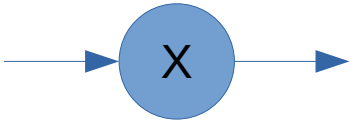
Programming

- Limited instruction set requires use of “tricks” to do basic operations.
  - Multiply by coeff : requires repeated instructions to ramp contents of accumulator
  - Allpass filter : specific sequence of load/store to buffer addresses.
  - Some operations require “dummy” DRAM addresses – places to put data that won’t actually be used.
- Cannot alter DSP characteristics in realtime
  - Changing coefficients / delays requires changing code. Must load whole new program.

Multiplication by constant > 1.0

```
0x0E : 1 0x1E8F   LDHALF  RD 0x2173 -> ACC        ;Acc = src/2 + sgn
0x0F : 2 0x0000   STRPOS  WR  ACC -> 0x0002      ;dst = Acc, Acc = Acc + Acc/2 + sgn
0x10 : 2 0x0000   STRPOS  WR  ACC -> 0x0002      ;dst = Acc, Acc = Acc + Acc/2 + sgn
0x11 : 2 0x0000   STRPOS  WR  ACC -> 0x0002      ;dst = Acc, Acc = Acc + Acc/2 + sgn
0x12 : 2 0x3F5B   STRPOS  WR  ACC -> 0x0002      ;dst = Acc, Acc = Acc + Acc/2 + sgn
```

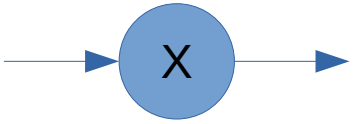
- Contents of DRAM location 0x2173 is loaded into Accumulator and multiplied by 2.53125
- Uses “dummy” location 0x0002 which has final value of (0x2173) * 1.6875



Multiplication by constant < 1.0

```
0x2C : 1 0x08EF   LDHALF  RD 0x3712 -> ACC        ;Acc = src/2 + sgn
0x2D : 3 0x0000   STRNEG  WR ~ACC -> 0x0001       ;dst = ~Acc, Acc = ~Acc/2 + sgn
0x2E : 3 0x3711   STRNEG  WR ~ACC -> 0x0001       ;dst = ~Acc, Acc = ~Acc/2 + sgn
0x2F : 0 0x3C1B   SUMHALF RD 0x3712 -> ACC        ;Acc = Acc + src/2 + sgn
```

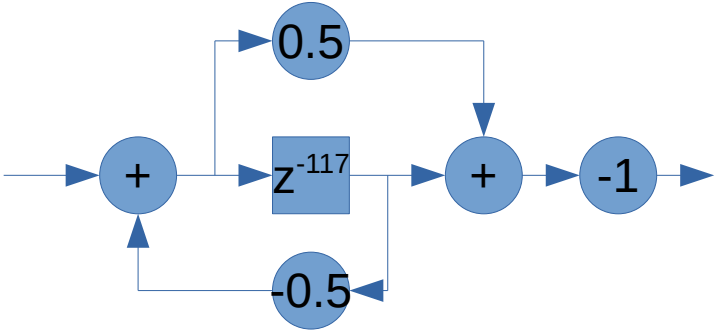
- Contents of DRAM location 0x3712 is loaded into Accumulator and multiplied by 5/8 (0.625)
- Uses “dummy” location 0x0001 which has final value of (0x3712) * 1/4



Allpass Filter

```
0x23 : 0 0x0075   SUMHALF RD 0x3A2A -> ACC          ;Acc = Acc + src/2 + sgn
0x24 : 3 0x3F8B   STRNEG  WR ~ACC -> 0x3A9F         ;dst = ~Acc, Acc = ~Acc/2 + sgn
0x25 : 0 0x0000   SUMHALF RD 0x3A2A -> ACC          ;Acc = Acc + src/2 + sgn
0x26 : 0 0x3FB2   SUMHALF RD 0x3A2A -> ACC          ;Acc = Acc + src/2 + sgn
```

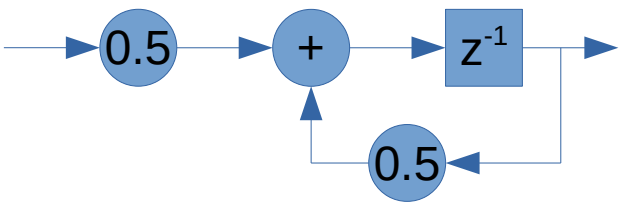
- Contents of Accumulator is processed through an allpass filter with coeff = 0.5 and delay of 117 samples
  - (0x3A9F – 0x3A2A) = 117
- Allpass filters are a crucial ingredient of reverb algorithms and typically there are many required.
  - Program 22 uses 9 allpass filters with lengths ranging from 76 to 1306 samples
- This allpass has output inverted as side effect



Lowpass Filter

```
0x42 : 2 0x216F   STRPOS  WR  ACC -> 0x0001        ;dst = Acc, Acc = Acc + Acc/2 + sgn
0x43 : 1 0x1E91   LDHALF  RD 0x2170 -> ACC         ;Acc = src/2 + sgn
0x44 : 0 0x2170   SUMHALF RD 0x0001 -> ACC         ;Acc = Acc + src/2 + sgn
0x45 : 2 0x0000   STRPOS  WR  ACC -> 0x2171        ;dst = Acc, Acc = Acc + Acc/2 + sgn
```

- Contents of Accumulator is processed through a first-order lowpass filter with coeff = 0.5
- DRAM location 0x2170 stores state of the filter, location 0x0001 is used as ‘dummy’



Typical MV I Reverb Program