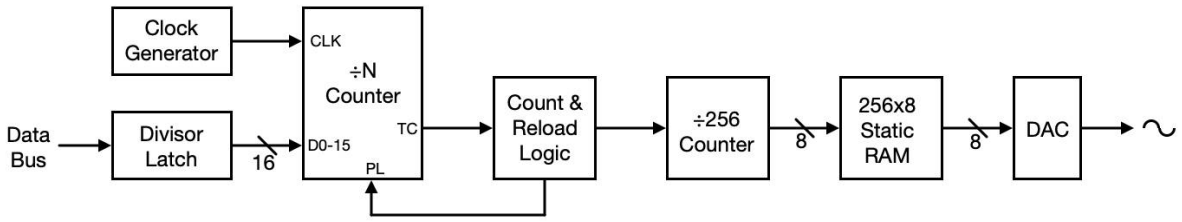
Digital Waveshape Generator

by Phillip L Harbison

Digital Waveshape Generator

I graduated from the University of Alabama in Huntsville (UAH) in 1981 with a degree in Electrical Engineering. For my senior design project I designed and implemented what I called a Digital Waveshape Generator (DWG). I had a strong interest in music synthesizers as I was a trained classical pianist and a fan of progressive rock music. I intended this DWG to be the core of a single voice in a synthesizer. Here is a block diagram of the DWG.



The DWG is purely digital up to the output of the Digital-to-Analog Converter (DAC). It begins with a master clock signal with a frequency that is approximately 256 times the Least Common Multiple of the 12 frequencies of the top octave (C8-A#8). The chosen frequency is 1802240 Hertz. If it was not possible to source a clock oscillator with that exact frequency it could be generated using a Phase-Locked Loop (PLL).

The master clock is then divided by a 16-bit integer to generate a clock that is 256 times the desired output frequency of the sound to be generated. This divisor is saved in a 16-bit latch implemented using a pair of 74LS374 8-bit latches which can be individually written to using a microprocessor. This allows the divisor to be saved in two 8-bit writes in the case of an 8-bit data bus. The ÷N counter (where N is the 16-bit divisor) is implemented using four 74LS161 4-bit synchronous binary counters. The 74LS161 has a parallel load feature which is used to load the counters with the divisor. It then counts down to zero and reloads.

Each time the ÷N counter reaches zero it clocks a ÷256 counter which is implemented using an additional pair of 74LS161 counters. The 8-bit output of the ÷256 counter is used as the address to a 256x8 static RAM. The 8-bit output of this RAM is used as the input to a fast 8-bit Digital-to-Analog Converter (DAC).. This output is the analog wave of the sound being .

The analog wave was fed to an additional 8-bit multiplying DAC not shown in the block diagram. The intent was to use the multiplying DAC to implement the features of an envelope generator and voltage controlled amplifier of a synthesizer voice in software.

Choosing The Master Clock Frequency

The master clock frequency needs to be the Least Common Multiple (LCM) of the frequencies of the twelve notes in the top octave (C8-B8). A true LCM is not possible since eleven of the exact frequencies of the top octave are not integers. Only A8 is an integer which is 7040 Hertz. However, it is possible to choose a frequency that can be divided by a set of 12 integers producing the top octave frequencies with an acceptable error. Table 1 shows the top octave frequencies, the necessary divisor, the resulting frequency, and the error relative to the top octave frequency when using a master clock frequency of 14417920 Hz.

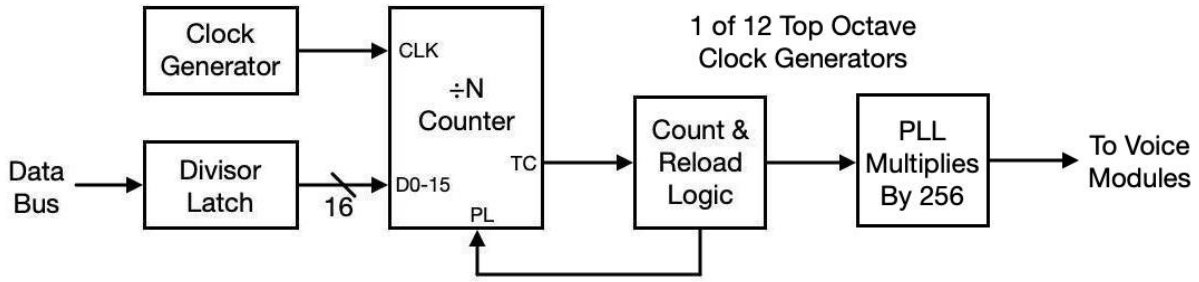
Table 1

Note	Frequency	Divisor	Generated	Error
C8	4186.00904480958	3444	4186.38792102207	0.0091%
C8#	4434.92209562995	3251	4434.91848661950	0.0001%
D8	4698.63628667852	3068	4699.45241199478	0.0174%
D8#	4978.03173955329	2896	4978.5635359116	0.0107%
E8	5274.04091060592	2734	5273.56254572056	0.0091%
F8	5587.65170292806	2580	5588.34108527132	0.0123%
F8#	5919.91076338615	2435	5921.11704312115	0.0204%
G8	6271.92697570799	2299	6271.38755980861	0.0086%
G8#	6644.87516127915	2170	6644.20276497696	0.0001%
A8	7040.00000000000	2048	7040.00000000000	0
A8#	7458.62018428947	1933	7458.83083290222	0.0028%
B8	7902.13282009806	1825	7900.23013698630	0.0240%

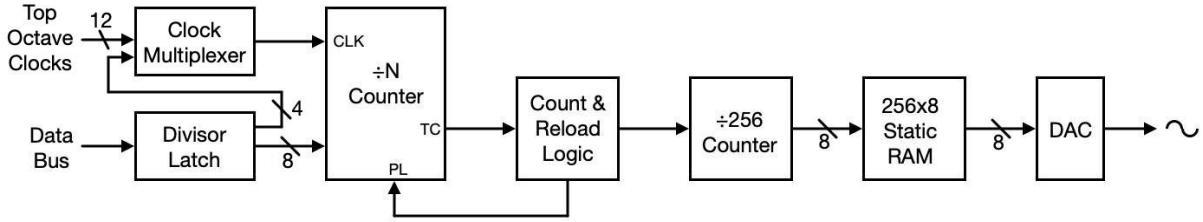
When using equal temperament tuning, the notes on the scale are separated by 100 cents. The largest error when using a master clock frequency of 14417920 Hertz is 0.0204% or 2 cents. A tuning error of less than 3-5 cents is generally imperceptible top the human ear.

There is one glaring problem. To sequence through the waveshape RAM we need 256 times the fundamental frequency. That pushes the master clock frequency to 3.9 GHz which is impractical. One possible solution is to create 12 top octave clocks, each 256 times the fundamental frequency. The individual voices would then choose the appropriate top octave clock and divide it by a power of two to obtain a clock that is 256 times the desired frequency.

The following diagram shows a possible implementation of a top octave clock generator. It produces a clock at 256 times the frequency of a not in the top octave (C8-B8). Twelve of these are required.



Each voice module would then use a 12-to-1 multiplexer to choose the appropriate top octave clock. The following illustrates this.



One might ask why choose octave 8 for the top octave. On a standard 88-key piano keyboard, only one note (C8) of octave 8 is used. This decision was arbitrary and suitable results could be achieve using octave 7 as the top octave. This design adheres to the philosophy that overkill is better than risking problems.

Choosing Resolutions

The choice to implement the wave shape with 256 points with 8-bit resolution was driven primarily by the parts available on a poor college student's budget and any samples available. For the waves shape RAM, a sample of a 1024x8 RAM with 100 nanosecond access time was available. The 75% extra RAM could be used to quickly switch between 4 wave shapes, and 100 nanosecond access time was more than sufficient. A sample of a fast 8-bit DAC was also available.

Implementation

The DWG was prototyped using wire wrapping with standard protoboard and wire wrap sockets. At the frequencies involved, noise was not a major consideration but a ground grid was used with decoupling capacitors for each device. The design was tested and demonstrated using a modular industrial control computer based on the 6502 microprocessor and a bus extension board on loan from my employer.

I received a grade of A+ for the course and my faculty advisor commented that the design was much more complicated than most submissions for the course. I was able to produce wave shapes the would have been impossible with analog synthesizers available in 1980.

Modern Considerations

The DWG was prototyped using technology available at reasonable prices in 1980. At that time, the 74LS (Low-power Shottky) series was a good trade-off of speed, power consumption, and cost. Today those would probably be replaced with 74HC (High-speed CMOS) series for significant power reduction. The speed of both 74LS and 74HC series are more than adequate for this application.

A modern implementation of the DWG could implement all but the DAC in a Field Programmable Gate Array (FPGA). Modern gate arrays could fit as many as 16 DWGs,

Playability

Two features not considered when designing the DWG were portamento (glide) and pitch bending. Both could be implemented in software which could vary the divisor for pitch bending and sequencing through intermediate divisors for portamento.

The DWG design assumes a single wave shape can be used to model an instrument. This is not true for many instruments. For example, string instruments sound brighter as pitch increases, while the lower notes on a piano have a more complex timbre. A larger RAM could be used to store multiple wave shapes and switch between them in software. The current design uses a 1024x8 SRAM which can stored 4 wave shapes. Modern SRAMs range in size up to 8 megabytes. A 32Kx8 SRAM could store a different wave shape for every note on an 88-key keyboard with room to spare.

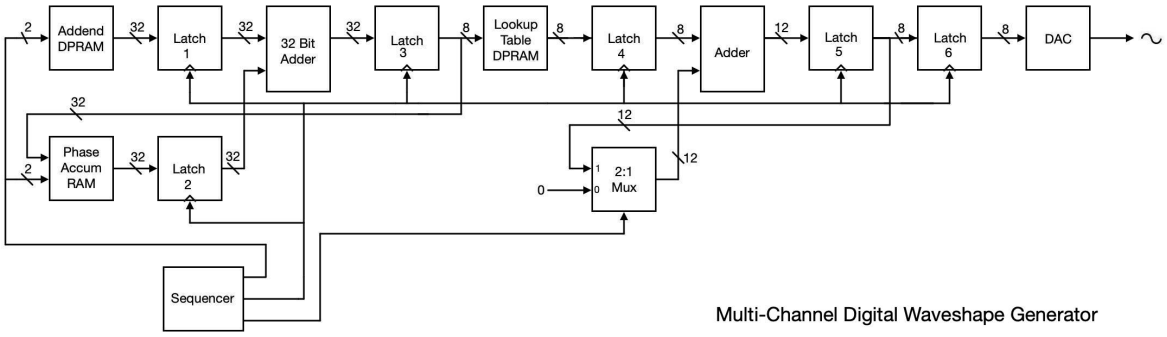
A Different Approach

A different approach to the divider concept is Direct Digital Synthesis (DDS). A Numerically Controlled Oscillator (NCO) uses a phase accumulator to add a fixed phase increment each clock cycle. The master clock frequencies required for accurate tuning is much more practical than the 3.9 GHz required for the divider approach. In DDS the output of the phase register is used to index a phase-to-amplitude lookup table (LUT).

There are many integrated circuits that implement DDS but most synthesize simple waveforms such as a sine or sawtooth. If the DDS is implemented discretely, the LUT is exposed. the wave shape memory from the divider approach can be used as the LUT for a DDS system.

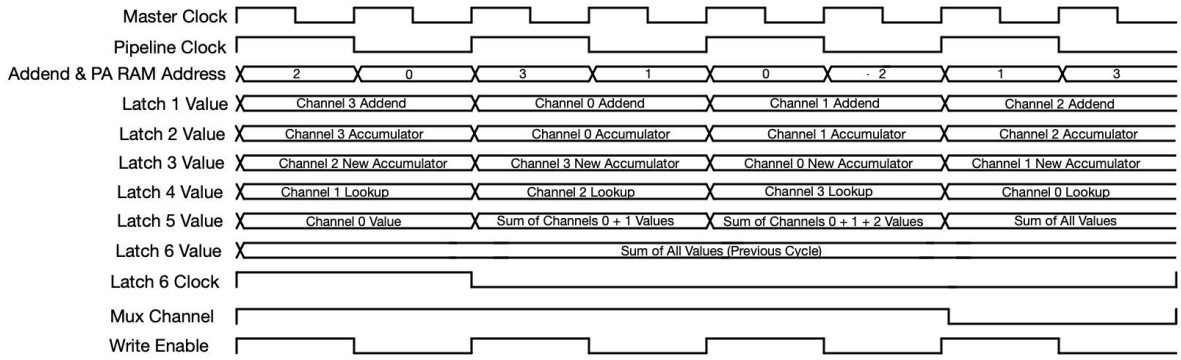
Unfortunately, a discrete implementation requires a lot of devices. A DDS with a 32-bit phase accumulator would require 4 8-bit latches (e.g. 74HC374) and 8 4-bit adders (e.g. 74HC283) to implement the phase accumulator. Between 2 and 4 additional 8-bit latches are needed to store the addend depending on the maximum phase adjustment per clock cycle.. Additional devices are needed to implement the LUT and control logic as well as a DAC.

Available logic devices such as the 74HC family and memory devices are much faster than necessary to implement a single channel DDS. The following design implements a multi-channel DDS where the devices are used in *time slices*. The phase adjustment and phase accumulator latches are replaced by memory devices. An additional set of adder devices are used to sum the LUT output for each channel to drive a single DAC.

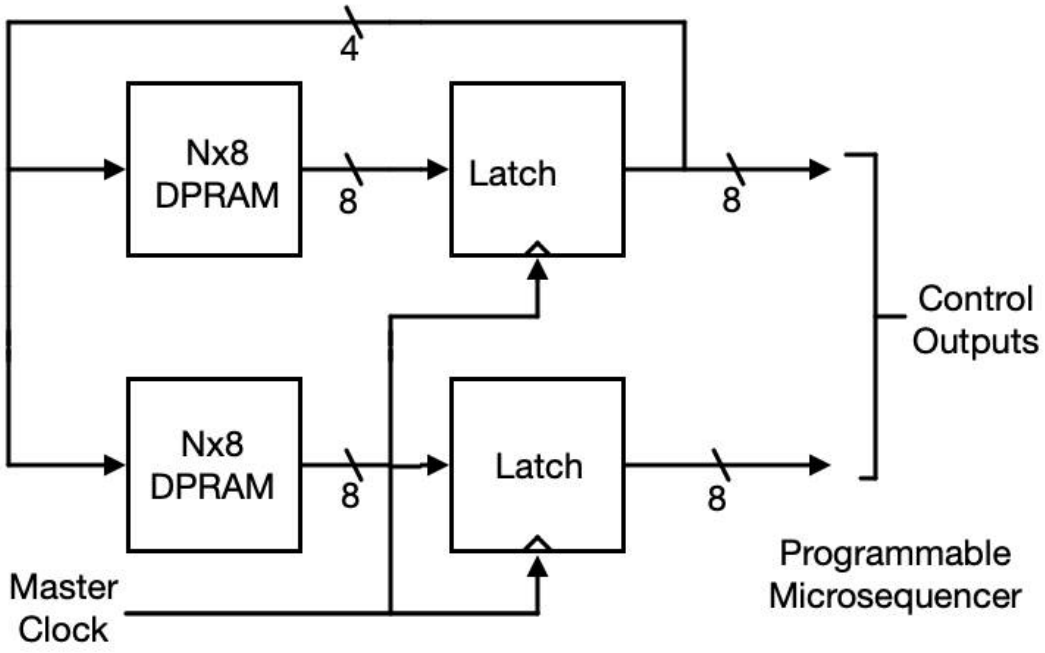


Multi-Channel Digital Waveshape Generator

This design is pipelined. Here is a timing diagram.



The design is controlled by a programmable microsequencer implemented with two Nx8 Dual-Port RAMs (where N >= 16) and two 8-bit latches (e.g. 74HC374). Four of the latch outputs are fed back to the DPRAM as address 0-3. The remaining address lines are not used. The sequencer controls whether the system generates 4 channels with 256 samples per cycle, 8 channels with 128 samples, or 16 channels with 64 samples. Here is a diagram of the sequencer.



Compromises

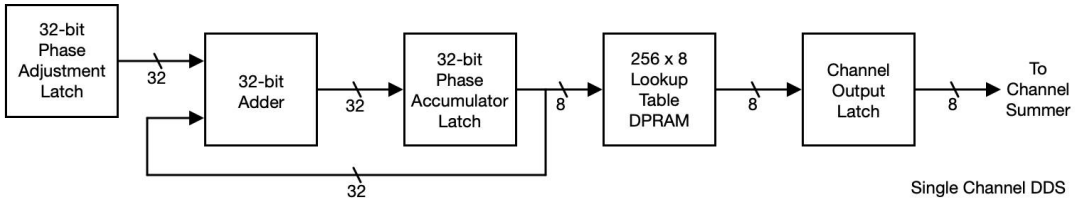
While the DDS design solves the problem of matching the chromatic scale, it does so by sacrificing resolution at higher frequencies. For example, at C8 (the highest key on an 88-key keyboard) the phase adjustment (addend) is 1797877 yielding only 5.6 samples per cycle, far less than the goal of 256 samples per cycle. I have not determined the impact this will have on the sound.

With a sample rate of 10 Mhz, the highest note with a sample rate greater than or equal to 256 is F2. Each doubling of the clock frequency raises this by an octave; however, the master clock is the sample rate multiplied by the number of channels, so this strategy quickly becomes untenable using 74HC series devices. If 8 74HCT283 devices are used for the 32-bit adder, the design probably could not support a master clock frequency higher than 20 MHz (50 nanosecond cycle time).

The devices used throughout the design are dual-port RAMs (DPRAMs) except the phase accumulator RAM. This is for convenience to allow a microprocessor to modify the memories without a lot of bus control logic.

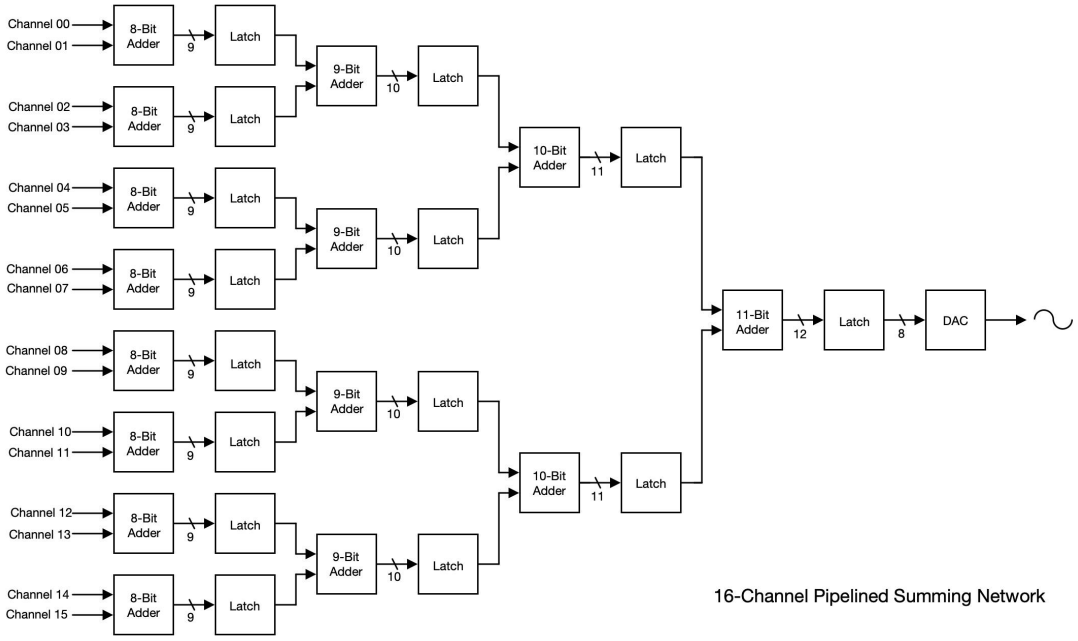
The phase adjustment (addend) memory and phase accumulator memory could be as small as 16xN but no such devices are currently in production. The 74LS219 was a 16x4 RAM but an HC/HCT version of this device was never produced. The 74HCT670 4x4 register file could be used to support a minimum of 4 channels.

One alternative would be to implement the entire design excluding the DAC in a Field Programmable Gate Array (FPGA). For example, the Lattice iCE40UP5K family prices range from $5 to $8 and have enough internal RAM to support as many as 16 channels. There is probably enough RAM to allow multiple waveshapes in memory and select a different waveshape based on the desired pitch Here is a design of a single channel DDS.



Single Channel DDS

A summing network would need to be pipelined to account for the delays in a tree of adders. Here is a design of a 16-channel pipelined summing network.



16-Channel Pipelined Summing Network

Future Plans

It is possible this device will be available in the future either as a complete product or as a printed circuit board for hobbyists. You can follow the progress on the following web site.

`http://www.xavax.com/alvitar/dwg/`