AUTOMATING DIGITAL SIGNAL PROCESSING, DSP, DESIGN USING AUGMENTED SPICE SIMULATION

Lawrence Meares, Intusoft, U.S.A.

Abstract

Coding for real time signal processing is tedious and error prone. Even when you get it “right”, errors creep in. Clearly there is a need for computer automation to free the engineer from detailed book keeping so that better algorithms can be implemented. Simulation, using SPICE, partially solves the problem. Substituting a SPICE delay line for the z transform, z^{-1}, is the usual approach. But as complexity increases, the substitution breaks down, with instabilities above the sampling frequency that are clearly impossible and sap computational efficiency. Even more troublesome is the increased complexity when algebraic “feedback” occurs. Take, for example, a simple Buck regulator plant model. The output voltage feeds back to the input, making the current through the inductor proportional to the difference in output and input voltage. But the output voltage is proportion to this difference, making a circular dependency. The simplest approximation is to selectively use backward Euler integration to break this nasty feedback; while a “proper” solution is to solve the algebraic equation. But the detailed implementation is so error prone, that the designer accepts loss in performance, just to get the job done. This paper explores a new SPICE primitive model, Z-Delay that acts as a delay line in the frequency domain; but in time domain looks like a sample and hold - no more high frequency oscillation! Classically it’s really a z transform model, but when viewed from the outside, it looks like a zero-order hold. Next, the system of algebraic equations can be solved using matrix reduction. Combining this matrix solution in SPICE with automated code generation does the trick.

1. Summary

A target design was chosen to illustrate the power of DSP automation. Using a plant model for a Buck regulator, the current feedback is replaced by an estimated current, that is, a virtual current, from the model. This hidden variable requires extra calculation and there is a complex interaction between the output of the plant model and the duty ratio of the controller. It will be shown that a matrix reduction gives up a solution to this problem and is easily automated for each target digital signal processor, DSP.

1.1. A Close Look at the Z-Delay model

The Z-Delay model is really quite simple. It samples the data every T seconds and outputs its results after a TCOMP delay, where TCOMP is the computational delay. When viewed with a “probe”, it appears to be a sample and hold circuit. Unlike the SPICE transmission line, it has no continuous output. It behaves just like a DSP difference equation, so that oscillation above its sampling frequency is impossible. For AC analysis, it has the same properties as a transmission line. Figure 1 shows an example of its behavior.

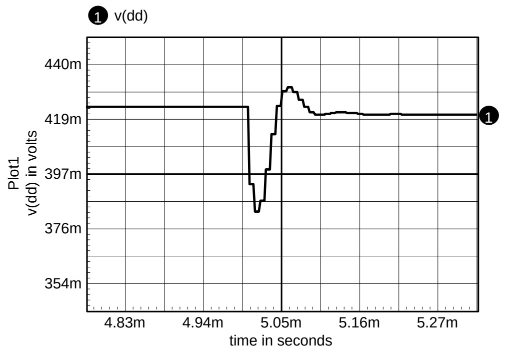

Fig. 1. Duty ratio control illustrates Z-Delay quantizing

This model is easilly added to SPICE simulators using code models provided with the the Georgia Tech. XSPICE [1] extensions.

1.2. A Target Design

The approach shown in Figure 2 uses a modified Kalman filter [2]. The load current is estimated using an extra control loop that makes the plant model output equal to the actual buck regulator output. This current estimate removes the need for current sense components, thereby reducing hardware cost. This is far too complex for an analog controller, but fits nicely within the capability of a DSP. It turns out that noise is predominantly systematic, caused by A/D quantizing, so that the usual Kalman filter noise measuring procedure can be replaced by beforehand knowledge using simulation results.

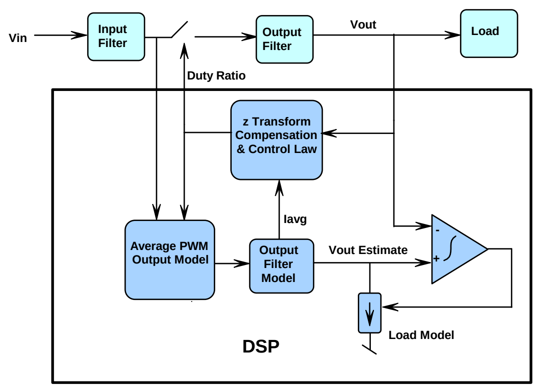

Fig. 2. Target design block diagram

This illustrates that most of the control law complexity is hidden within the DSP, hence the need for improving both design and test procedures.

1.3. Matrix Solution

These DSP control equations can be expressed using matrix algebra as shown in Figure 3. Assume there are j+k states that need to be evaluated, with some of the k states having a delay history expressed using the z-delay model. The equations can be arranged as shown with all trivial solutions at the bottom of the matrix (these are the z-delay outputs and A/D inputs).

$$\begin{bmatrix} 1 & a12 & a13 & \dots & \dots & a1j & \dots & & a1(j+k) \\ 0 & 1 & a23 & & & a2j & \dots & & \dots \\ 0 & 0 & 1 & a34 & & a3j & \dots & & \dots \\ 0 & 0 & 0 & 1 & & a4j & \dots & & \dots \\ \dots & & & & \dots & ajj & \dots & & a(j+k)(j+k) \\ \dots & & & & & \dots & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} V1 \\ V2 \\ \dots \\ \dots \\ Vj \\ Hk \\ H3 \\ \dots \\ H1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ \dots \\ 0 \\ Vkp \\ V3 \\ \dots \\ V1p \end{bmatrix}$$

Fig. 3. A matrix solution has RHS(0thru j)=0

The solutions are trivial because they are all of the form :

Var = constant,

whether the constant value is an A/D input or a z-delay output. Each of these constant values are accumulated prior to perfoming a new iteration. For z-delays they are the the matrix solutions propogated from the z-delay input state to its output. For the A/D inputs, they are measured at the beginning of each computational cycle.

Then the j+k by j sub matrix at the top will have its right hand side, RHS, equal to zero. The zero RHS occurs because these equations are formed by summing states and histories for each state variable ; that is,

Var = sum(state*coeff)

then,

Var – sum(state*coef) = 0

The importance of a zero RHS for these equations lies in the re-usability of the state matrix. With a constant state matrix the same equations are solved over and over with the computed histories placed in the RHS of the z-delay outputs for each sample. This matrix approach is the key to removing the algebraic dependancy from the DSP difference equations.

2. Z Transform Basics

Design and analysis of control systems are usually performed in the frequency domain; whereby the time domain process of convolution is replaced by a simple process of multiplication of complex polynomials in the frequency domain. Sampled data systems use a similar concept with a unit delay as the basic building block. The analog s-plane maps into the sampled data z-plane by substitution of variables where z=e^{sT} or more importantly by

$$z^{-1} = e^{-sT}$$

The later representation is seen to be identical to a transmission line, with z^{-n} representing a delay of nT seconds.

2.1. Direct Programming

Transfer functions, including impedance and admittance functions, are described as polynomial ratios of the form $G = \frac{N}{D}$, where

$N = a_0 + a_1 z^{-1} + \dots\ a_n z^{-n}$ and $D = 1 + b_1 z^{-1} + \dots b_n z^{-n}$ are the numerator and denominator polynomials respectively. Notice that $b_0 = 1$. Then rearranging the following equation with D' = D-1 yields:

$$\frac{Vo}{Vi} = \frac{N}{D}$$
$$Vo(1 + D') = ViN$$
$$Vo = ViN - VoD'$$

This is the "Direct" programming method that is more rigorously derived in [3] pp 475,477. This equation can also be implemented in the s-plane using the following block diagram with the unit time delay (UTD) replaced by a SPICE transmission line. This implementation allows for SPICE analysis of the time domain difference equations, including both transient and ac analysis, and of course Bode plots.

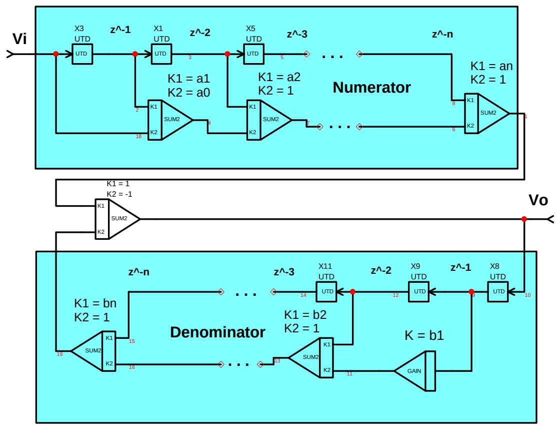

Fig. 4. Direct Programming Method.

2.2 Finding s as a Function of z

Solving for s as a function of z in the z transform yields

$$s = \frac{1}{T}\ln(z)$$

The ln(z) function can be broken down into 3 common approximations. Let's first do this by using the first term of the series expansion yielding the bilinear transformation:

$$\ln(z) = 2\frac{z-1}{z+1}$$

For $w << 1/T,\ z+1 \approx 2z$ further simplifies to:

$$\ln(z) = \frac{z-1}{z}$$

Alternatively $z+1 \approx 2$, and

$$\ln(z) = z-1$$

Then, the 3 approximations are:

$$s = \frac{1}{T}(z-1)$$
$$s = \frac{1}{T}\left(\frac{z-1}{z}\right)$$
$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

The second representation is the one commonly used [3] pg 741 in the z-transform tables. Mathematically it is common to let T = 1 and omit it from the tables, leaving it to the user to scale the result for other sample frequencies. If you fail to account for T, then you will get the wrong answer! Restating the above equations to represent integration and delays yields:

Rectangular, backward Euler, integration

$$\frac{1}{s} = T\left(\frac{z^{-1}}{1-z^{-1}}\right) \qquad \text{eq 1}$$

Rectangular, forward Euler, integration (z Transform)

$$\frac{1}{s} = T\left(\frac{1}{1-z^{-1}}\right) \qquad \text{eq 2}$$

Trapezoidal integration (Bilinear transform)

$$\frac{1}{s} = \frac{T}{2}\left(\frac{1+z^{-1}}{1-z^{-1}}\right) \qquad \text{eq 3}$$

There are 3 interpretations to these equations in terms of integration method, although they were derived here from a series expansion; they could have also been derived in time domain using rectangular and trapezoidal integration methods. These representations have several interesting properties.

First, eq 1, has a step response of z^{-1}, which is interpreted as the dead-beat response, that is, the error to a step response will be reduced to zero in one sample time. That's the holy grail of digital control system design, so the loop compensation should be designed to replicate that behavior. Unfortunately, adding 6dB of gain makes it unstable.

The second equation gives good results for control system compensation as shown in figure 5; the phase is leading near the nyquist frequency, providing added margin.

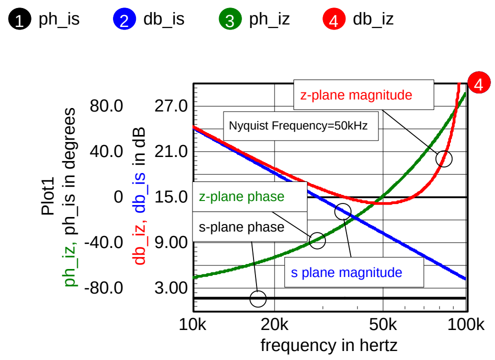

Fig. 5. Z Forward Euler integration compared to s-plane integration shows massive phase lead.

Finally, eq 3 inserts a term that averages the previous sample with the current sample. The average of 2 samples at ½ the nyquist frequency is exactly zero for any signal, at Fs/2, regardless of its phase because the 90 degree phase shifted signal is always equal in magnitude and opposite in sign at Fs/2. The phase stays at 90 degrees up to Fs/2. This property is useful for constructing algorithmic sine/cosine generators.

3. Making the Plant Model

Using the elementary z-transforms discussed previously, the DSP equations for a plant model can be developed from the block diagram shown in Figure 6. Notice that polynomial functions are not combined. There are several reasons behind this. First, the inductor current must be available as the hidden control system variable. But importantly, combining z-transforms into higher order polynomials may lead to coefficient scaling that either overflows or underflows the DSP word length.

The admittance, $Y_L = \dfrac{1}{Ls + R_L}$

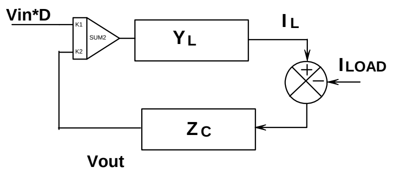

Fig. 6. DSP plant model

and the impedance, $Z_C = R_C + \dfrac{1}{Cs}$ are formed using z transform integrators and the direct programming method discussed earlier.

3.1 Plant Model Difference Equations

Both Vin and I_{LOAD} are variables; however, they are taken as constants here in order to simplify the discussion. The equations can be written by inspection from Figure 6;

$$I_L = Y_L(V_{IN}D - V_{OUT})$$

$$V_{OUT} = Z_C(I_L - I_{Load})$$

First, Y_L is converted into a z-transform in a form that can be easily coded using the direct programming method.

$$Y_L = \frac{K_1}{1 - K_2 z^{-1}}$$

where $K_1 = \dfrac{T}{L + R_L T}$ and $K_2 = \dfrac{L}{L + R_L T}$

Next, Z_C is similarly programmed.

$$Z_C = R_C + \frac{T}{C}\left(\frac{1}{1 - z^{-1}}\right)$$

The schematic representation, from which DSP equations can be written by inspection, is shown in Figure 7.

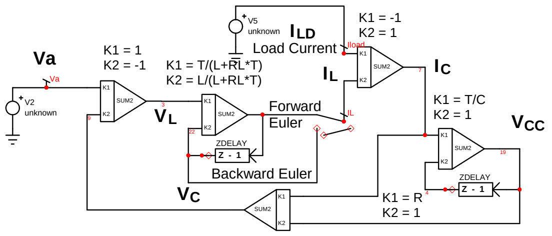

Fig. 7. DSP z-transform plant model

If I_L is connected to the "Backward Euler" node, then the algebraic feedback is broken and the difference equations can be written directly from the schematic. But if the "Forward Euler" node is used, there is an algebraic dependency that must be removed by solving the simultaneous set of equations. Overall control system phase margin is better using forward Euler integration because of its phase lead at high frequency (Figure 5). Two controllers are added to complete the control system design. First, the plant model output voltage is matched to the measured output by setting the load current to the integral of the error. Lead-lag compensation is added. Next, the Output error is integrated and summed with the hidden average current variable to control Duty ratio. Again, lead lag compensation is used to get acceptable gain and phase margins. The number of equations for this complete solution increases dramatically and there are several algebraic feedback loops, requiring computer automation for implementation.

4. The SPICE Matrix

The schematic of Figure 7 can be used to run a SPICE simulation. SPICE enters the circuit netlist into a modified nodal admittance, MNA, matrix. It then re-arranges the matrix to precondition it for solution by LU decomposition. The general problem SPICE solves has non-zero terms in the RHS. LU decomposition can use the matrix over and over to iterate to a solution. Unfortunately the non-zero RHS requires both forward and backward substitution. The matrix shown in Figure 3 doesn't need forward substitution because the RHS in the upper portion is zero. So it is necessary to extract just the parts of the matrix that are needed and re-arrange them according to the previously described algorithm. This is accomplished by expanding the SPICE simulations to include a .DSP simulation type. This is basically an operating point solution that outputs the DSP specialized matrix to a SPICE .out file. Then a separate software program reads in the matrix and forms the DSP code for each target processor. All of this can be integrated into a simulation environment, producing source files that are added to the DSP development environment.

4.1. Eliminating Intermediate States

The only states needing a solution for each iteration are the z-delay inputs and DSP outputs. Other states need not be evaluated. The trick to eliminating these states is to place them at the top of the matrix and then use gauss elimination to make all matrix entries below the main diagonal zero. Then, when performing backward substitution, DO NOT evaluate these variables in the top rows!

4.2. Conditioning the Matrix

As the number of variable increase, the matrix tends to become sparse; that is, most entries are zero. Reduced instruction set, RISC, architectures may require extra instructions because the next state index may not be close enough to the previous index. This situation can be remedied by moving the trivial equations upward in the matrix until they are just below a row that uses the state variable. Notice that all values below the moved row are zero because that's how the trivial solutions are defined. It is then possible to move the column left until it resides on the main diagonal. No fill-ins below the main diagonal are added by doing this; however, the non-zero matrix values will cluster near the main diagonal, eliminating the need to reload the state index.

4.3 Building the Matrix

The matrix equations for the circuit Figure 7 have been solved as shown in Figure 8. That was accomplished by placing unwanted variables in the top rows and using Gauss elimination to triangularize the matrix. Space does not permit illustrating the complete Gauss elimination procedure.

$$\left|\begin{array}{ccccccccc}
\cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{-1} & \cancel{0} & \cancel{0} & \cancel{0} \\
\cancel{0} & \cancel{1} & \cancel{0} & \cancel{-1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} \\
\cancel{0} & \cancel{0} & 1 & -R & -1 & 0 & R & 0 & 0 \\
\cancel{0} & \cancel{0} & 0 & 1 & a & -a & b & c & 0 \\
\cancel{0} & \cancel{0} & 0 & 0 & 1 & d & e & f & -1 \\
\cancel{0} & \cancel{0} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\cancel{0} & \cancel{0} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\cancel{0} & \cancel{0} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\cancel{0} & \cancel{0} & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right| \text{X} \left|\begin{array}{c}
\cancel{VL} \\ \cancel{IC} \\ VC \\ IL \\ Vcc \\ Va \\ ILd \\ ILP \\ VccP
\end{array}\right| = \left|\begin{array}{c}
\cancel{0} \\ \cancel{0} \\ 0 \\ 0 \\ 0 \\ Vain \\ ILdin \\ ILPi \\ VccPi
\end{array}\right|$$

Parameters	Parameters
K1=T/(L+RL*T)	b=-R*K1/(1+R*K1)
K2=L/(L+RL*T)	c=-k2/(1+R*K1)
K3=T/Cf	d= -a*k3/(1+a*k3)
R=RC	e=(1+b)*k3/(1+a*k3)
a=k1/(1+R*K1)	f=c*k3/(1+a*k3)

Fig. 8. Gauss elimination solves the matrix

The strikeouts in Figure 8 represent the elimination of VL (voltage across the inductor) and IC (capacitor current). The solution has 11 non-zero matrix entries requiring multiply-accumulate operations. The matrix before solution had 10 non-zero entries; so that eliminating unneeded variables produces a more efficient operation than a solution using LU decomposition which would require at least one fill-in. The parameters block in Figure 8 can be evaluated using a SPICE pre-processor.

4.4. Writing Difference Equations

First, snake a line from the topmost equation to be solved down to the equation for the first non-trivial state as shown in Figure 9 (Do not include equations that are to be discarded, V1 in this case).

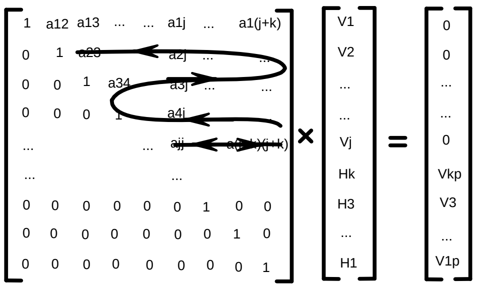

Fig. 9. Ordering of coefficients.

Then accumulate the coefficients in the order of use as shown by the directional arrows. Notice that the jth row may proceed in either direction depending on whether there are an odd or even number of solutions. That becomes the coefficient array. The states are indexed from the bottom to the top, Vj to V2. For now, divide each row by its main diagonal value so that no division (or multiplication) is necessary when evaluating a state variable. Then for each variable to be evaluated, initialize the multiply accumulate (MAC) register and perform the MAC operations needed for the row. Alternate between 2 MAC accumulators so that saving the result can be delayed until the appropriate state variable is in range. If a state variable index is too far away, it must be reinitialized, adding an instruction each time the situation is encountered. After accumulating the result, it must be scaled and saved and the MAC accumulator must be reinitialized. That requires two instructions for each state plus a MAC instruction for each coefficient. After solving for the state variables, the solution for z-delay inputs is propagated to the z-delay output in the RHS, and The PWM output is sent to the DSP modulator. Then the A/D inputs are sampled to begin the next iteration.

4.5. DSP Architectures

Code generation for different DSP architectures place different demands on the conditioning of the matrix. RISC architecture, as used in Microchip DSP's, demand more attention to indexing. While CISC architectures, typical of Texas Instruments DSP's, will have more efficient indexing but they use more instruction cycles.

4.6. A SPICE Subset

The entire family of SPICE primitive parts is not required to describe the DSP operation. It is sufficient to consider only the following:

- Voltage Source for inputs
- Z-delay code models for time delays
- Behavioral elements for mathematical operations
- Nodes beginning with alphabetical characters identify states for the required solution.

Therefore, a SPICE net configuration conforming to these rules forms the netlist for the .DSP simulation.

5. Conclusion

Using a DSP opens new horizons for PWM algorithm development, but exposes weakness in computer automation. As complexity increases, the use of a transmission line to simulate a delay breaks down. Moreover, the difference equations become more complex than can be handled by hand calculation. A method of automation has been disclosed that uses a SPICE simulator supporting a new delay model and uses the simulator to extract the basic matrix, which is further manipulated using matrix algebra to yield an efficient formation of difference equations. Future work is needed to make the state variables in the hardware available in support of SPICE AC, DC and TRAN simulations; perhaps using an embedded real-time operating system.

6. Literature

[1] F. L. Cox, et al.: Code-level modeling in XSPICE: *Proceedings* IEEE International Symposium on Circuits and Systems, (ISCAS 92), vol. 2, pp. 871-874: 10-13 May, 1992

[2] Lawrence Meares: SPICE Simulation Provides Significan Advantages For a DSP Based SMPS: Power Electronics Technology Magazine: Part1&2: March-April 2009

[3] BENJAMIN C. KUO: DIGITAL CONTROL SYSTEMS: 2nd EDITION, OXFORD UNIVERSITY PRESS, 1992