DIGITAL-MIXED-SIGNAL MODELLING USING SIGNAL FLOW EQUATIONS

1ST JULY 2025

PETER GROVE
DISTINGUISHED MEMBER OF TECHNICAL STAFF
RENESAS ELECTRONICS CORPORATION

Distinguished Engineer

RENESAS

AGENDA

- Digital-Mixed-Signal (DMS) modelling styles
- S-Domain to Z-Domain
- DMS Capacitor model example
- Known Issues / Corner Cases
- Python simplification for adoption
- Questions

WHAT IS DMS METHODOLOGY

(Digital-Mixed-Signal) DMS Methodology directs not to structurally connect its two-terminal devices together in arbitrary networks, but instead to model those networks with signal transfer modelling. This presentation aims to explain why and how.

DMS MODELLING CHOICES

AnalogLib Style Modelling	Signal Transfer Modelling
Schematic representation using custom primitives	Higher level Mathematical
10x faster than Spice	>10,000x faster than Spice
Faster to initially create but tuning takes time.	Can be error prone in math's equations
Validation is slower	Validation is quick

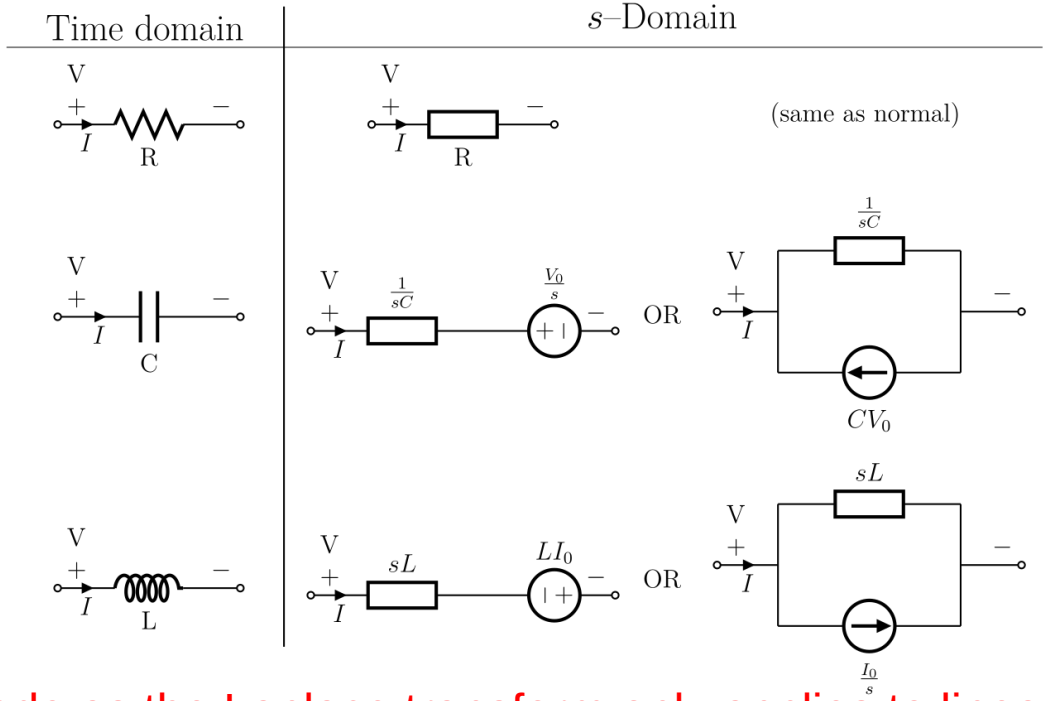
- Scientific Analog has tried to make AnalogLib style to work and still not got industry traction.
  - Presented as solution for DMS in UVM-MS and got no traction.
  - Bespoke solution that is non-transferable

- EDA companies provide a library of passive components, but they are slower than Signal transfer ones.
  - Bespoke solution where code of library components can't be shared, and some is encrypted.
  - Hard to debug when things go wrong.

OVERVIEW

TIME->FREQUENCY DOMAIN

Laplace transform - Useful for converting differentiation and integration in the continuous-time domain into much easier multiplication and division in the Laplace domain (frequency domain).



No Diode as the Laplace transform only applies to linear differential equation

Z-transform – Can be considered a discrete-time equivalent for the Laplace transform. That is take a discrete-time domain into the frequency domain. No differentiation and integration though, so not easy to go from passive networks.

Z-TRANSFORM

In signal processing, one of the means of designing **digital filters** is to take analog designs, subject them to a **bilinear transform** which maps them from the s-domain to the z-domain, and then produce the digital filter by inspection, manipulation, or numerical approximation.

Such methods tend not to be accurate except in the vicinity of the complex unity, i.e. at low frequencies.

Bilinear transform can be used to convert continuous-time filters into discrete-time filters, and vice versa.

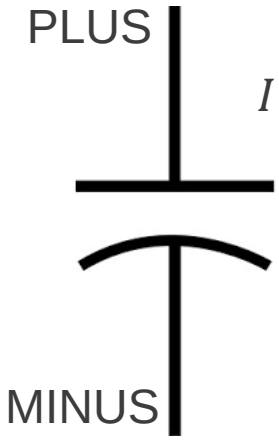
$$s = \frac{2}{T}\frac{(z-1)}{(z+1)}$$

When you have an energy storage element these are the steps needed to model them in event-based simulation.

SIGNAL TRANSFER MODELLING

CAPACITOR EXAMPLE

Time domain



$$I = C\frac{dV}{dt}$$

Laplace Transform (S domain)

$$I = sCV$$

Bilinear Transform (Z domain)

$$s = \frac{2\,(Z-1)}{T(Z+1)} \qquad f = \frac{1}{T}$$

$$s = \frac{2f\,(1-Z^{-1})}{(1+Z^{-1})}$$ *Trapezoid Rule*

$$I(t) = 2f * CV(t) * \frac{(1-Z^{-1})}{(1+Z^{-1})}$$

Re-arrange for V(t) = ….

$$I(t) + \mathrm{I(t-1)} = 2f * C(V(t) - V(t-1))$$

$$\frac{I(t)+\mathrm{I(t-1)}}{2fC} = (V(t) - V(t-1))$$

$$\frac{I(t) + \mathrm{I(t-1)}}{2fC} + V(t-1) = V(t)$$

Other Methods

$s = \frac{1-Z^{-1}}{TZ^{-1}}$ *Forward Euler Rule*

$s = \frac{1-Z^{-1}}{T}$ Backwards Euler Rule

VERILOG CODE

```
real I_1, I, V_1, V, Iprobe;
real TS  = 4e-9;
real C   = 100e-9;
assign f = 1.0/TS;
always begin
       #(TS*1s);     //Sample Time
       I_1 = I;      //Sample Shift  i[t-1] = i[t]
       I   = Iprobe;//Get New value i[t]   = next value
       V_1 = V;      //Sample Shift  v[t-1] = v[t]
       //Calculate new Voltage across Capacitor
       V = V_1 + (I + I_1)/(2.0 * f * C);
end
```

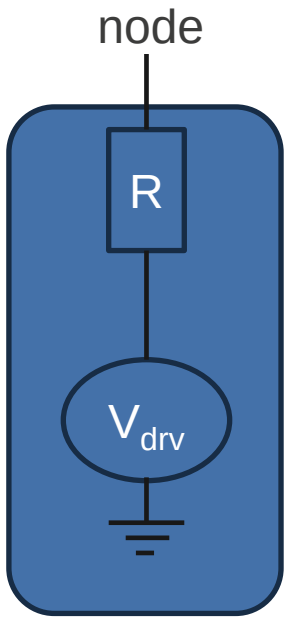
$$\frac{I(t) + \mathrm{I}(\mathrm{t}-1)}{2fC} + V(t-1) = V(t)$$

- Using variables like I_1 and not a packed array is quicker. (Or an unpacked array could be used).
- ‘C’ could be a derated cap and some teams have used that. (Tiny losses as it doesn’t preserve charge)

- Trying to dynamically change TS like an analog solver creates more issue than it solves as there is no generic solution. Digital simulators cannot roll back time.

DMS MODEL REPRESENTATION

GROUND REFERENCED



R is set to Capacitor ESR value. (Equivalent Series Resistance shown later)

Iprobe is (node.V-V_{drv})/R, sampled at TS intervals.

What about a *2 Terminal Device*?

- Use one node as a reference, then it's simply $V_{drv} = V + V_{minus}$
- Use impedance to balance out
  - Requires recursive function calls that *may not* converge.
    - Many ways to try and solve this but none are generic!
    - Remember DMS is not a matrix solver but event driven solver.

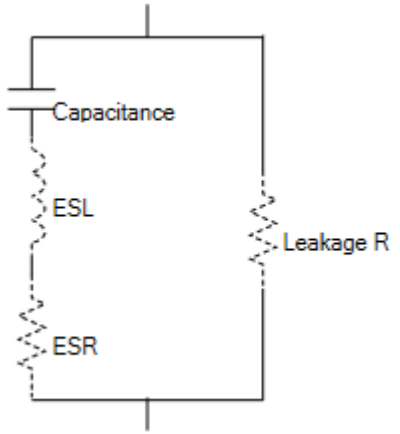
OTHER CAPACITOR MODELS

- Add series R_s and parallel R_p to get this equation

$$V(t)= V(t-1)\left(\frac{2R_P fC+2R_P R_S fC-1}{2R_P fC+2R_P R_S fC+1}\right) + i(t)\left(\frac{R_P+2R_P R_S fC}{2R_P fC+2R_P R_S fC+1}\right) + i(t-1)\left(\frac{R_P-2R_P R_S fC}{2R_P fC+2R_P R_S fC+1}\right)$$

- Add series R_s+ L and parallel R_p to get this equation *(Untested!)*

$$V(t)= \left(\frac{1}{2R_P fC+2R_S fC+2^2 f^2 CL+1}\right) * (\quad i(t) \quad *(2fR_P\ R_S C + 2^2 f^2 CLR_P + R_P)$$
$$+i(t-1)*(-\ 2^3 f^2 CLR_P + 2\ R_P\ )$$
$$+i(t-2)*(2\ f\ R_P\ R_S C + 2^2 f^2 CLR_P + R_P)$$
$$-v(t-1)*(-\ 2^3 f^2\ CL+2)$$
$$-v(t-2)*(-2\ f\ C\ R_P\ -2\ f\ C\ R_S + 2^2 f^2 CL+1))$$



ESR - Equivalent Series Resistance
ESL - Equivalent Series Inductance

- Optimization done in Renesas components
  - Only update the UDN output if Vdelta > 1uV. (Spice solver abstol on Voltages)
    - Internally the Capacitor model needs to track the real V and not the filtered.
      - Requires Driver-Receiver-Segragation technology borrowed from Verilog-AMS.
    - Real device testing has shown a game change in runtime improvement.
  - Coefficients calculated in different assign block so computed only when they change values.

KNOWN ISSUES

- Using a series L causes a change in the frequency response!

$$Y(s) = \frac{I(s)}{V(s)} = \frac{s}{L\left(s^2 + \frac{R}{L}s + \frac{1}{LC}\right)}$$

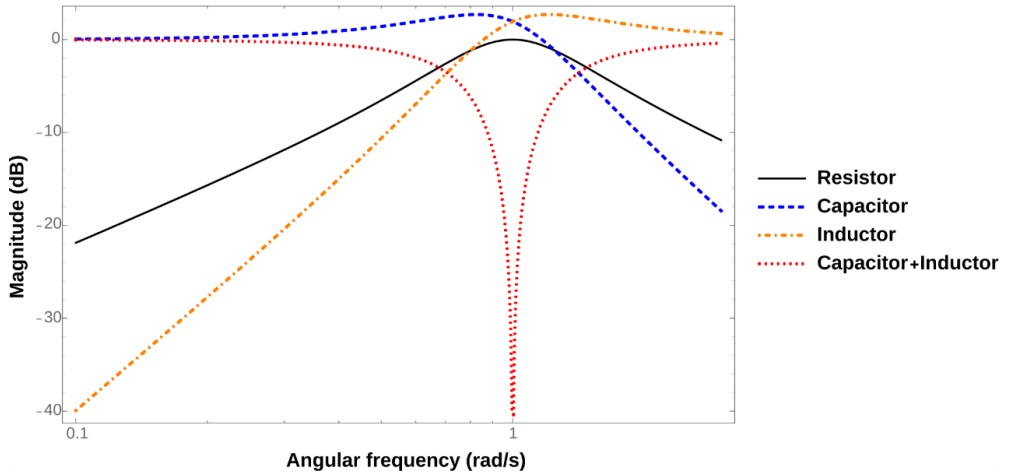
  - Impedance decreases with frequency in a capacitor (attenuation)

$$\mathrm{R} = \frac{1}{2\pi f c}$$

  - Impedance increase with frequence in an inductor (gain)

$$\mathrm{R} = 2\pi f L$$

- Notch at $\mathrm{f} = \frac{1}{2\pi\sqrt{LC}}$



- **<u>Possible Solutions (untested)</u>**
  - Set Sampling Period 1/f
  - Pre or post filter, but why model the inductor
  - Add Bandwidth filters to all DMS models so matches analog.
  - DMS is for functional simulations. Don't use DMS to check parasitic effects. Let analog simulations deal with attenuations/gain.
  - Some limiter to dampen any gain.

CORNER CASES

- Large external changes causes the sampled *I = (node.V-*V_{drv}*)/R* value to generate oscillations.
  - If node.V changes enough it can make I very large, which pushed the calculation of V to change too much, say negative.
  - System then has a feedback loop (IIR Filter) that may have gain or attenuation. Gain will cause the node to go to NaN!
- **Solution** is to use a clamp, but the clamp must equalize the Z domain equation so it stable. (Requires DRS)

$$V(t)= V(t-1)\left(\frac{2R_PfC+2R_PR_SfC-1}{2R_PfC+2R_PR_SfC+1}\right) +i(t)\left(\frac{R_P+2R_PR_SfC}{2R_PfC+2R_PR_SfC+1}\right)+i(t-1)\left(\frac{R_P-2R_PR_SfC}{2R_PfC+2R_PR_SfC+1}\right)$$

Kv_1 Ki Ki_1

I(t) == I(t-1) == *(node.V - vdrv)/R*
V(t) == V(t-1)

Resolved value of PLUS node's UDN excluding the Cap driver. (DRS)

Re-Arrange

V(t) = ((Ki + Ki_1) * (P_ext_drv.V - MINUS.V))/((1-Kv_1)*(P_ext_drv.R + R_s)+(Ki + Ki_1));
I(t) = (P_ext_drv.V – V(t) - MINUS.V)/(P_ext_drv.R + R_s);
I(t-1) = I(t);
V(t-1) = V(t);

How to decide when to clamp is up to the audience to work out!

OPTIMIZATIONS

- Signal Transfer Modelling
  - Disable the sampling when that node is not driven. (Minimal improvement.)
  - Code refactoring and profiling. (Big runtime improvement when vdelta/idelta features added in.)
  - Tune the sample frequency for the Bandwidth of interest. Why sample at GHz when the interest is at MHz.
  - Don’t probe everything or introduce Tool specific waveform absdelta on reals been dumped.
  - Merge passives networks into a single transfer function. Even Switched system can be done via state space modelling.

- General DMS Modelling
  - Use tran/tranif1/tranif0 statement now supports UDN as well as logic for an ideal Open/Close switch. (SV-2023 LRM)
  - If one port is meant to be a reference disable the bidirectionality.
  - Avoid complex passive networks as it will slow down simulations.
  - Try and avoid multiple updates in delta cycles.

PYTHON - SYMBOLIC COMPUTING

- Use Python's Sympy library to take the S-Domain representation into the Z-Domain.

```
from sympy import *
#Setup some variables used by equation
var('V, I, Rp, Rs, L, C, s, z, f', positive=True)
init_printing()
#S-Domain V = ... equation
equation = Eq(V, (I - V/Rp)*(Rs + s*L + 1/(s*C)))
#Solve for V/I = ... equation
H_s = solve(equation, V)[0]/I
#Define Z transform to use
s_z = 2*f*(z-1)/(z+1)
#Apply transform
H_z = H_s.subs(s, s_z)
#Get numerator terms
num = poly(fraction(together(H_z))[0], z)
num.all_coeffs()
#Get denomintor terms
den = poly(fraction(together(H_z))[1], z)
den.all_coeffs()
```

Other transforms could be used and tried more easily now!

$$\frac{Numerator}{Denominator}$$

PYTHON - SYMBOLIC COMPUTING

- From the Python Script we now have the *numerators* and *denominators*, now how to use them in an SV model

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{a_m Z^m + \ldots + a_2 Z^2 + a_1 Z^1 + a_0}{b_m Z^m + \ldots + b_2 Z^2 + b_1 Z^1 + b_0}$$

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{a_m + \ldots + a_2 Z^{2-m} + a_1 Z^{1-m} + a_2 Z^{-m}}{b_m + \ldots + b_2 Z^{2-m} + b_1 Z^{1-m} + b_0 Z^{-m}}$$

Note no change to numerators/denominators

$$y[n] = \frac{b_{m-1}}{b_m} y[n-1] + \frac{b_{m-2}}{b_m} y[n-2] + \cdots + \frac{a_m}{b_m} x[n] + \frac{a_{m-1}}{b_m} x[n-1] + \cdots$$

Calculated in other always@ block as they will not change often

PYTHON EXAMPLE – 2ND ORDER LPF

$$V = V_i \left(\frac{1}{1 + sCR} \right)^2$$

```
var('V, Vi, R, C, s, z, f', positive=True)
init_printing()
#S-Domain V = ... equation
equation = Eq(V, Vi*(1 /(1 + (s*C*R)) )**2)
```

Numerator a2, a1, a0 $[1, 2, 1]$

Denomerator b2, b1, b0 $[4C^2R^2f^2 + 4CRf + 1, -8C^2R^2f^2 + 2, 4C^2R^2f^2 - 4CRf + 1]$

$$y[n] = \frac{b_1}{b_2} y[n-1] + \frac{b_0}{b_2} y[n-2] + \frac{a_2}{b_2} x[n] + \frac{a_1}{b_2} x[n-1] + \frac{a_0}{b_2} x[n-1]$$

Renesas.com