2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# Modeling Analog Systems Using Full Digital Simulations (A State Space Approach)

Rajat K Mitra

Cadence Design Systems

cadence®

# Problem Description

- Traditional approach to simulating mixed signal designs is to use a mixed mode simulator such as Cadence® AMS solutions
- Digital behavior is simulated using C++, Verilog or System Verilog
- Analog behavior is simulated using Verilog AMS or SPICE
- Simulation has very high fidelity
- Suffers runtime efficiencies and thus is a bottle neck to a Metric Driven Approach

# Possible Solution (The State Space Method)

- Certain types of analog circuitry can be modeled using a popular technique used to model control systems in the subject of Control Theory
- Control Theory seeks to identify two types of properties of a Control System
- Controllability – To manipulate the behavior of the system in a desired way one seeks to identify relevant parameters
- Observability – To identify the effect of ones manipulations these parameters must be visible

# Possible Solution (The State Space Method )

- A convenient representation of the system's dynamics can be formulated mathematically as follows –

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ . \\ \frac{dx_n}{dt} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & . & a_{1n} \\ a_{21} & a_{22} & . & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & . & b_{1n} \\ b_{21} & b_{22} & . & b_{2n} \\ . & . & . & . \\ b_{n1} & b_{n2} & & b_{nn} \end{bmatrix} * \begin{bmatrix} u_1 \\ u_2 \\ . \\ u_n \end{bmatrix}$$

- Or succinctly :

$$X' = A * X + B * U$$

$$X' \rightarrow \frac{dX}{dt}$$

**X**' is the derivative of the state variable X
**A** is the state transition matrix
**B** is the input matrix
**U** is the input vector

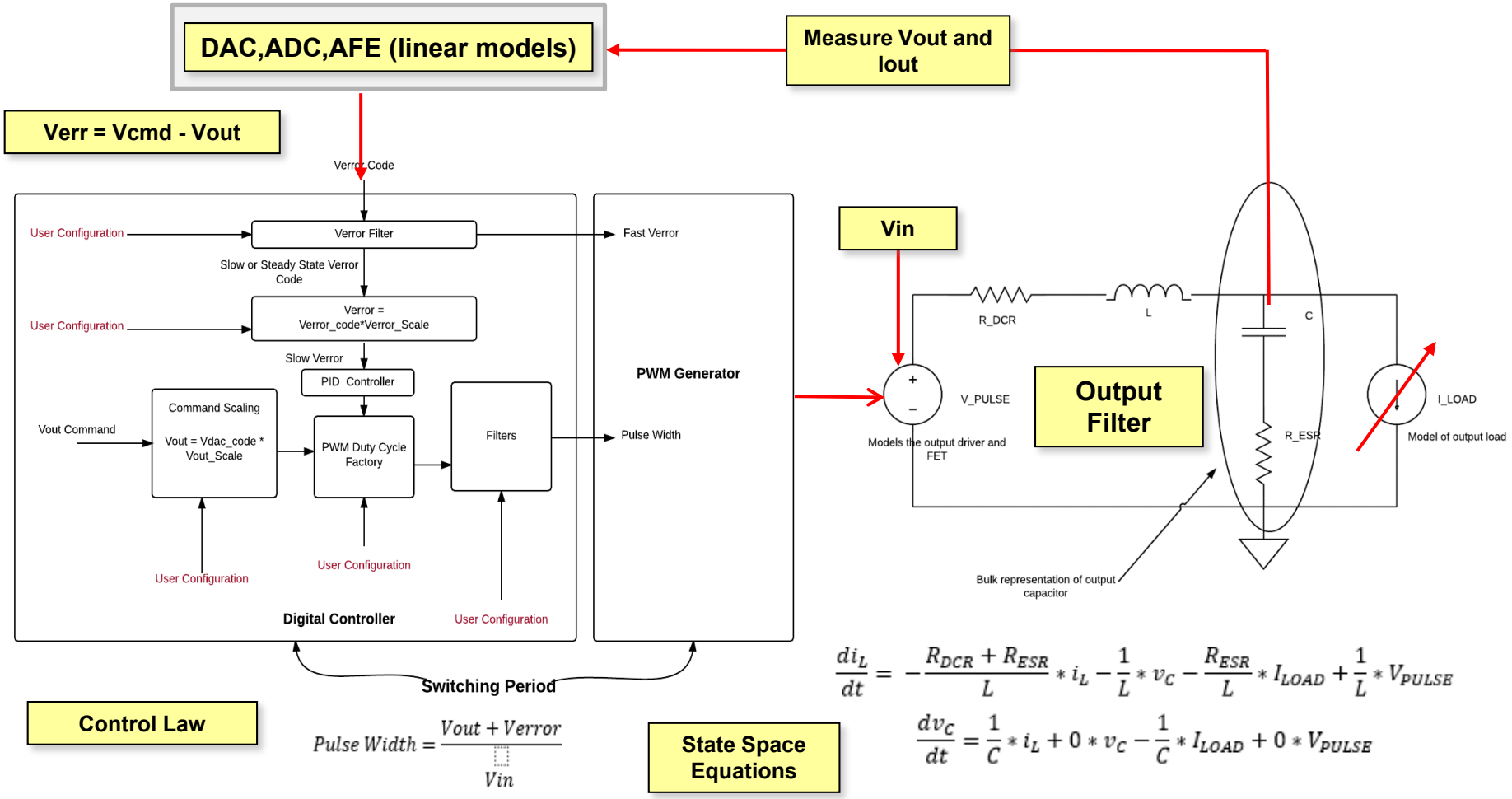
# Possible Solution (The State Space Method )

- The state variables (and in this case the set of state variables) are quantified properties of the system that one seeks to control and observe
- When the derivative vector (or time rate of change) of these variable are know then given the current values (at time t) of these variables one can attempt to model subsequent values (at time t+dt) as follows (f(x,t,dt) is an estimation function) –

$$X(t+dt) = X(t) + dt * f(X,t,dt);$$

- The state transition matrix A and the input matrix B are determined from physical laws governing the system

# Example of a Voltage Converter



# State Space Model Of Output Filter

- $$i_L(t+dt) = i_L(t) + dt * \frac{di_L(t)}{dt}$$

$$v_c(t+dt) = v_C(t) + dt * \frac{dv_C(t)}{dt}$$

- Here we have used a Forward Euler method to determine the next sext step value of the inductor current and the capacitor voltage; the estimation function is simply the rate of change of the state variable

# Model of Output Filter

- Initialization of State Transition and Input Matrix Coefficients

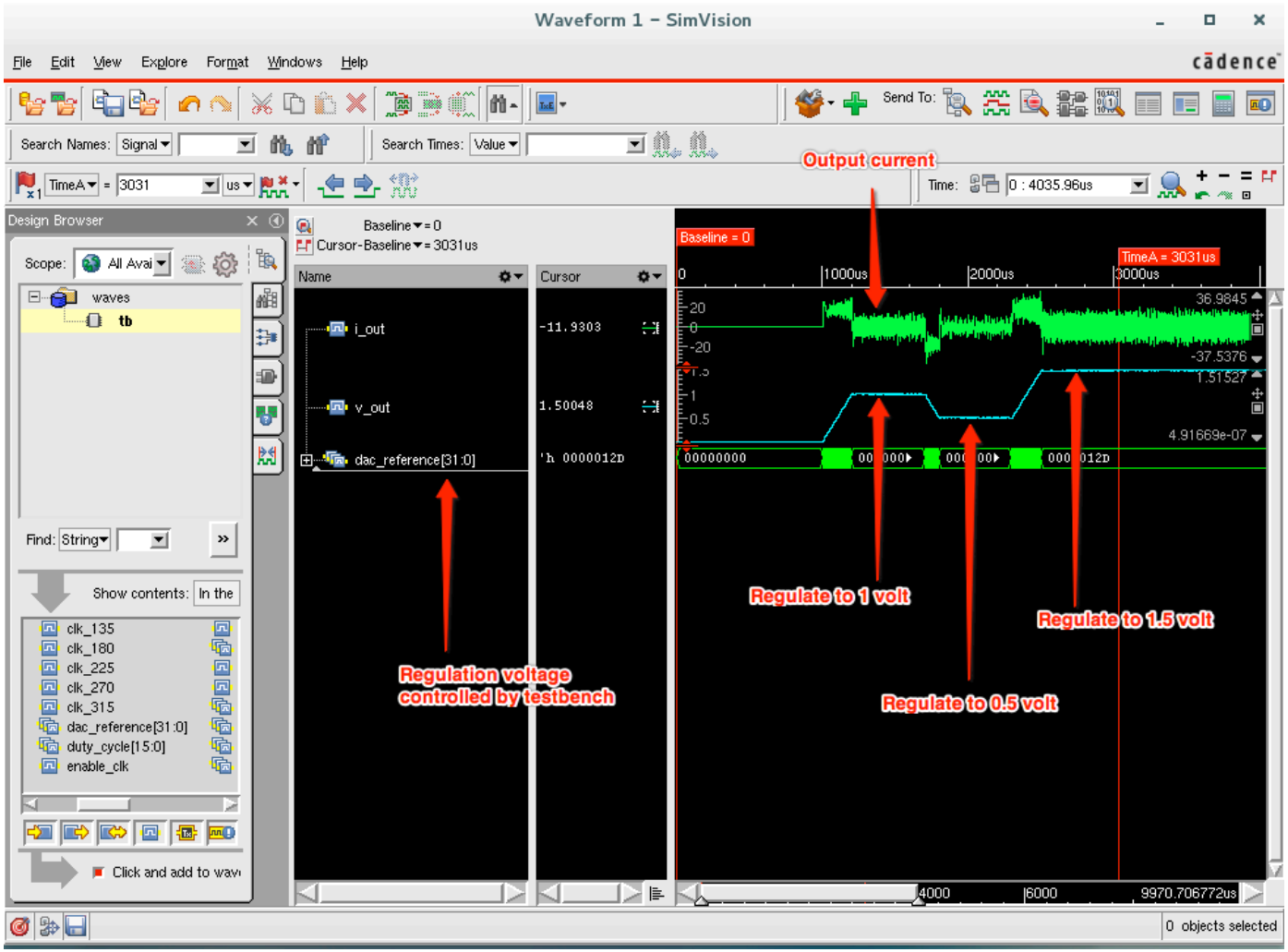
```
initial begin
     a00 = -1.00*(DCR+ESR)/L;
     a01 = -1.00/L;
     a10 = 1.00/C;
     a11 = 0.00;
     b00 = 1.00/L;
     b01 = -1.00*ESR/L;
     b10 = 0.00;
     b11 = -1.00/C;
     dt_r = dt*1.00e-9;
  end
```

# Model of Output Filter

- Sample and Update Loop

```
always begin
      //evaluate loop every time step...
      //this method will sample vpulse and iload
      //sampling errors will be introduced.
       #(dt);

      //compute the time derivatives of the voltage and current
      d_i_inductor  = a00*i_inductor[0] + a01*v_capacitor[0] +
                       b00*vpulse       + b01*iload;
      d_v_capacitor = a10*i_inductor[0] + b11*v_capacitor[0] +
                      b10*vpulse        + b11*iload;

      //compute the next inductor current step and the capacitor
      // voltage step
      i_inductor[1]  = i_inductor[0]  + dt_r*d_i_inductor;
      v_capacitor[1] = v_capacitor[0] + dt_r*d_v_capacitor;

      //update
      i_inductor[0]  = i_inductor[1];
      v_capacitor[0] = v_capacitor[1];
   end // always begin
```

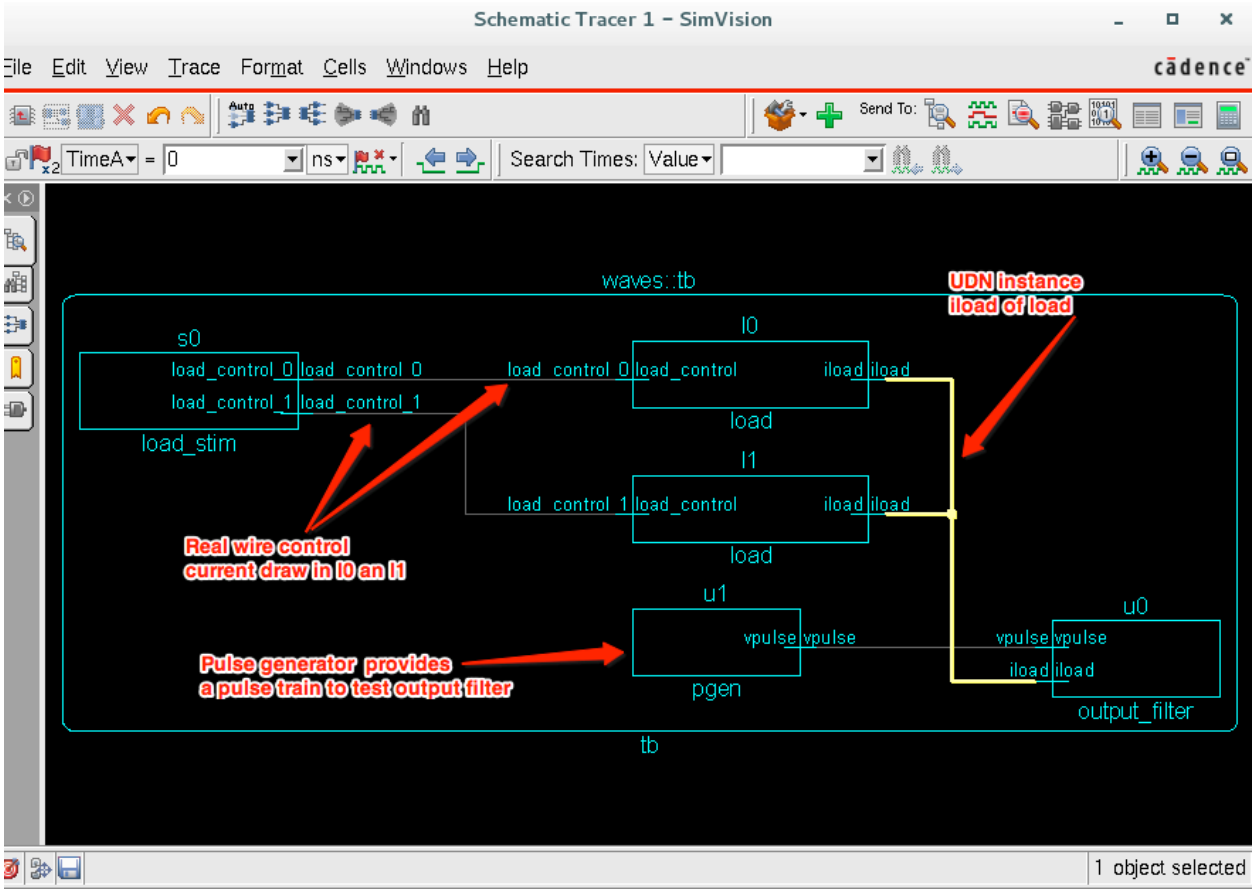
# Steady State Behavior



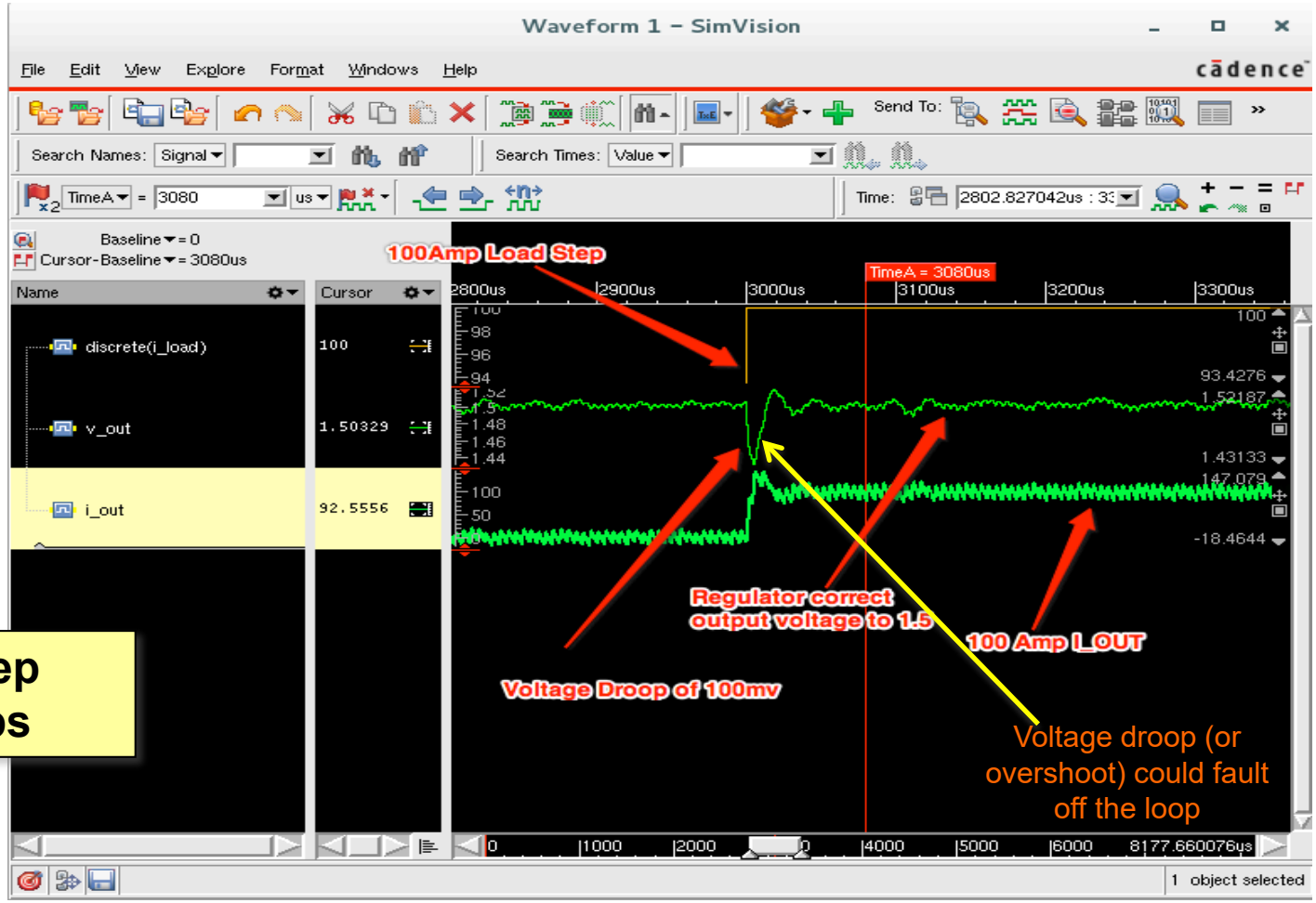
# Load Control Using Nettype

```
////////////////////////////////////////
//Definition of a current carrying wire
////////////////////////////////////////
typedef struct{
           real I;
           } load;

//resolve_TC is a resolution function
//that computes the total current of the
//network connected by an instance of "load"

function automatic load resolve_TC(input load driver[]);


   //sum over all drivers
   foreach(driver[i])begin
      resolve_TC.I += driver[i].load;
   end

endfunction

//Declare Nettype of type load with current resolution
function resolve_TC
nettype load current_net with resolve_TC
```

# Load Control Using Nettype

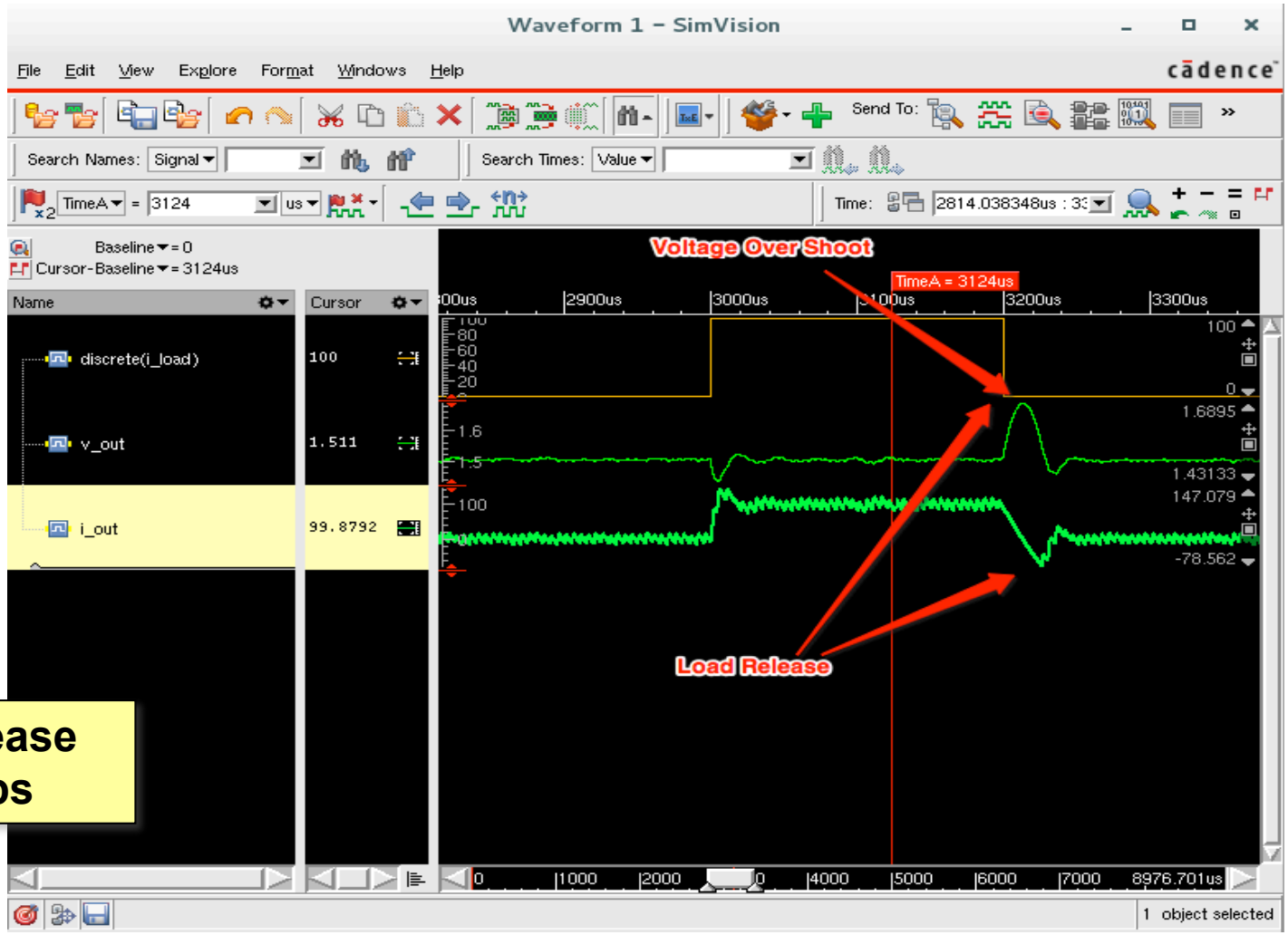


# Load Control Using Nettype




**Load Step 100Amps**

# Load Control Using Nettype




**Load Release 100Amps**

# Conclusion

- Methodology is digital centric
- Allow a MDV approach to verify digital controller
- Limitations – Requires manually crafting analog model
- Limitations – Current approach is mostly valid for Linear Control, modeling Non Linear Control is work in progress
- Fidelity of simulation is dependent on modeling accuracy
- Requires an insight of the smallest time constant of the analog circuitry to determine simulator time step
- Very effective in verifying
  - Controller's fault behavior ( OV, UV, OC, Over Temperature, Over Power )
  - For Multi Phase systems, successfully used to verify Low Power / High Power Operations