Accurate Charge-pump Regulator Modeling using SV *EEnet*

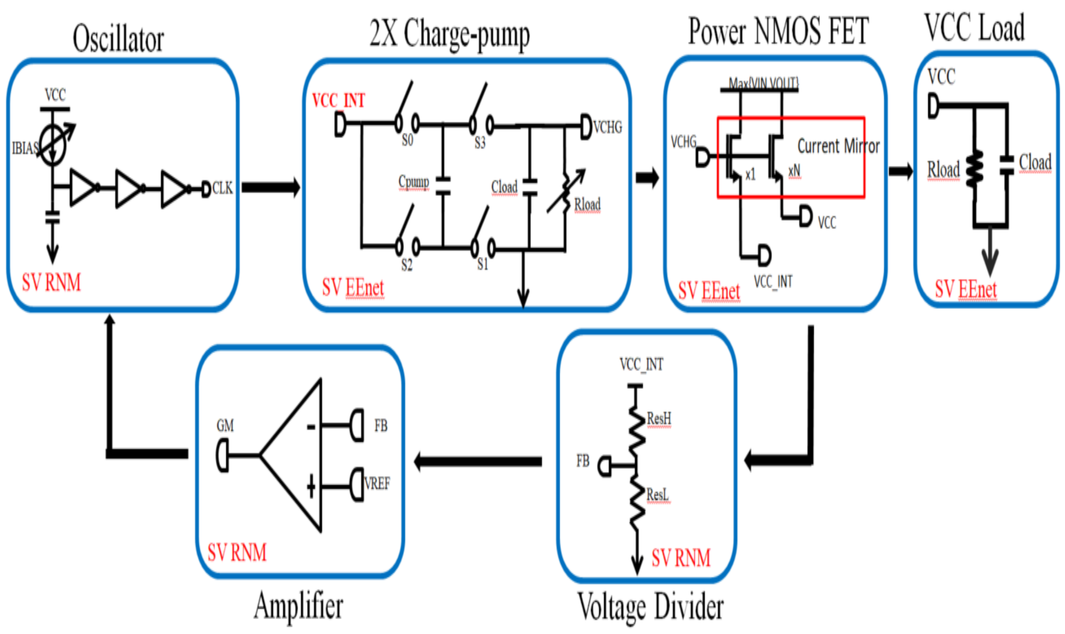
Sherry Li, Texas Instruments, Shanghai, China (sherry_li@ti.com), Tulong Yang, Texas Instruments, Dallas, USA (t-yang1@ti.com),
Ayesha Huq, Texas Instruments, Dallas, USA (*a-huq1@ti.com*)

MOTIVATION

- **To model analog blocks with high accuracy**
- **To balance simulation speed and accuracy for SOC functional verification**
- **Extend functional coverage in top-level verification and reduce effort and time for AMS**

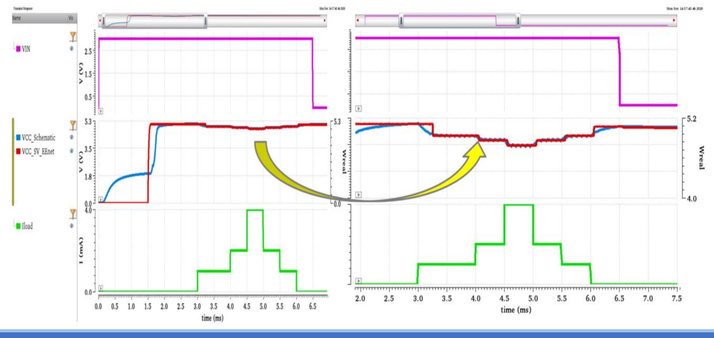
CHARGE-PUMP REGULATOR

- Internal power supply used in boost converter products
- Consists of a current-controlled oscillator, 2x cross-coupled charge-pump, a high gain amplifier, a resistor-divider and a Power FET



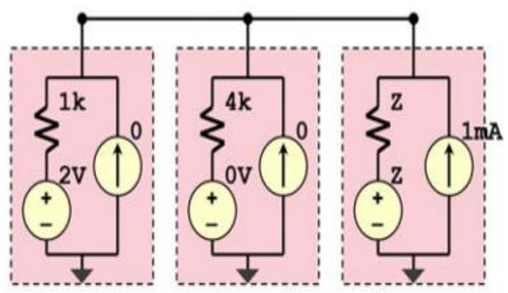
MODEL VERIFICATION

- Model matches with schematic with load current steps up to 1mA, 2mA, 3mA, 4mA
- Model sim runs 10.36 seconds, 10x speed up compared to schematic



SYSTEM VERILOG *EENET*

- Built-in SV nettype supported by Cadence with 3 fields, V, I and R
- Include analog impedance-based interactions

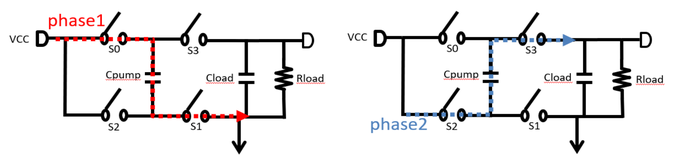


Output Evaluation:
ΣI = 2mA+0mA+1mA = 3mA
ΣG = 1m + 0.25m + 0 = 1.25m
Vout = ΣI / ΣG = 3m/1.25m = 2.4V
Rout = 1/ΣG = 800 Ω
Resulting EEnet value: '{2.4, 0, 800}

MODELING

CHARGE-PUMP MODEL

- Two-phase switch capacitor circuits

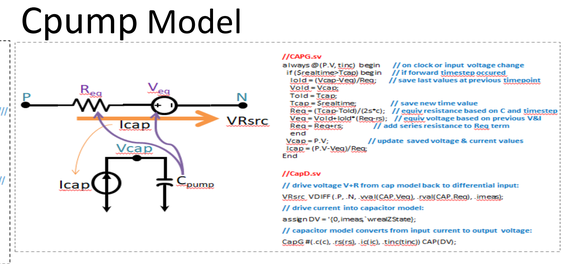


Switch Model

```
///Switch.sv///
initial sie = $SIE_input(P,P_new);  ///SIE_input system task to get pin value///
initial sie = $SIE_input(N,N_new);
...
always @(P_new.V, P_new.R, Vd, Rd) begin   /// if port P changes, update port N ///
begin
  VNdrv <= P_new.V - Vd;
  RNdrv <= P_new.R + Rd;
end

always @(N_new.V, N_new.R, Vd, Rd) begin   ///if port N changes,  update port P///
  VPdrv  <= Next.V + Vd;
  RPdrv <= Next.R + Rd;
end
.....
assign P = '{VPdrv, 0, RPdrv};     /// drive output pins differentially///
assign N = '{VNdrv, 0, RNdrv};
```

Cpump Model

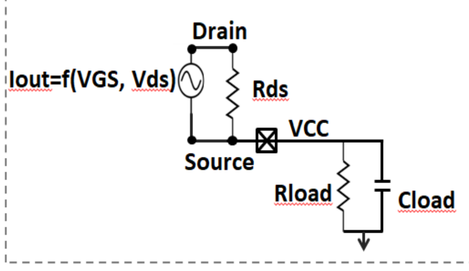



```
//CAPG.sv
always @(P.V, tinc)  begin      // on clock or input  voltage change
 if (Srealtime>Tcap) begin      // if forward timestep occured
  Iold = (Vcap-Veq)/Req;        // save last values at previous timepoint
  Vold = Vcap;
  Told = Tcap;
  Tcap = $realtime;             // save new time value
  Req = (Tcap-Told)/(2*C);      // equiv resistance based on C and timestep
  Veq = Vold+Iold*(Req-rs);     // equiv voltage based on previous V&I
  Req = Req+rs;                 // add series resistance to Req term
 end
 Vcap = P.V;                    // update  saved voltage & current values
 Icap = (P.V-Veq)/Req;
End

//CapD.sv
// drive voltage V+R from cap model back to differential input:
VRsrc  VDIFF (.P, .N, .vval(CAP.Veq), .rval(CAP.Req), .imeas);
// drive current into capacitor model:
assign DV = '{0,imeas, vreal'State};
// capacitor model converts from input current to output  voltage:
CapG #(.c(c), .rs(rs), .ic(ic), .tinc(tinc)) CAP(DV);
```

NMOS FET MODEL

- Nonlinear function of gate voltage
- Composes of a dependent current source and an internal resistor in parallel.

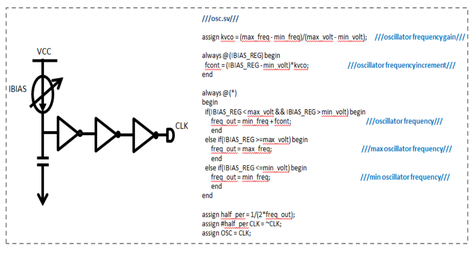



```
///sv_nmos.sv///

always @(G.V, D.V, S.V) begin
 if ((G.V-S.V)>vth) Vgs = G.V-S.V-vth;      /// active gate-source
 else  Vgsf = 0;
 if ((G.V-D.V)>vth)  Vgd = G.V-D.V-vth;     /// active gate-drain
 else  Vgd = 0;
 if ((D.V-S.V)>0)
   Klamda = kn*(1+lambda*(D.V-S.V));        /// Scalar for Output current
 else  Klamda = kn;
 Trm = Vgsf**2 - Vgdf**2;                   /// primary voltage-dependence term
 Is = Klamda/2*Trm+goff*(D.V-S.V);          /// computed Source current
end
```

OSCILLATOR MODEL

- RC delay circuit and 3 invertor buffers
- Oscillator frequency is proportional to its controlled bias voltage




```
///osc.sv///

assign kvco = (max_freq - min_freq)/(max_volt - min_volt);   ///oscillator frequency gain///

always @(IBIAS_REG) begin
 fcont = (IBIAS_REG - min_volt)*kvco;          ///oscillator frequency increment///
end

always @(*)
begin
 if(IBIAS_REG < max_volt && IBIAS_REG > min_volt) begin
   freq_out = min_freq + fcont;                ///oscillator frequency///
   end
 else if(IBIAS_REG >=max_volt) begin
   freq_out = max_freq;                        ///max oscillator frequency///
   end
 else if(IBIAS_REG <=min_volt) begin
   freq_out = min_freq;                        ///min oscillator frequency///
   end
end

assign half_per = 1/(2*freq_out);
assign #half_per CLK = ~CLK;
assign OSC = CLK;
```

REFERENCES

[1] Jess Chen, M.Henrie, M.F.Mar, Mladen Nizic, B.Bailey – Mixed-Signal Methodology Guide, Cadence Design Systems, CA, USA, 2012;
[2] Cadence Rapid Adoption Kits, "Using EEnet to perform Electrical Equivalent Modeling in SystemVerilog", 28 Sep 2017
[3] Cadence Training, *"Real Modeling with SystemVerilog"*, 06 Aug 2016
[4] R. Sanborn, R. Mitra, Z. Fan, "Best Practices for Verifying Mixed-Signal Systems", Cadence Application Notes, USA and Canada, 2018.
[5] El-Ebiary, Mohamed Dessouky, and Hassan Ghitani, "Behavioral modeling of a charge pump voltage converter." in Proceedings of the 2007 IEEE International Behavioral Modeling and Simulation Workshop, 2007, pp. 84-89.

ACKNOWLEDGEMENT

Highly appreciate Ayesha Huq for her fully guidance and support.

Many Thanks to Tulong Yang for his great help and suggestions.