Summary of updates from
BSIM-IMG 102.9.1 to BSIM-IMG 102.9.2

BSIM Group, UC Berkeley

Pragya Kushwaha (pragya@berkeley.edu)

A. Summary of enhancements

- 2017enh2: Self-Heating Temperature Clamping
- 2018enh2: Binning Equations for QME Effect
- 2018enh3: DVTP formulation in line with BSIM4

B. Summary of bug-fixes

- 2018bug1: VA code syntax (& vs &&)
- 2018bug2: division by zero in Vdsx term
- 2018bug3: Vthop is independent of PHIG1
- 2018bug4: Division by zero in dvth_dibl term
- 2018bug5: Add checks in code
- 2018bug8: gcrg term initialization in “else” block
- 2018bug10: $simparam(“gmin”) implementation for minimum conductance

C. Description of enhancements

- **2017enh2: Self-Heating Temperature Clamping**

  It is important to limit the internal temperature otherwise model can go to arbitrarily high internal temperatures (which can affect IV dramatically and cause non-convergence) during Newton-Raphson loops. We have clamped the internal temperature to a maximum device temperature i.e., TMAXC (new parameter added with default value 400C).

BSIM-IMG 102.9.2

```
`MPRnb( TMAXC          ,400.0          ,"Celsius"                          ,"Maximum Device Temperature" )

   TMAXK   = TMAXC + `P_CELSIUS0; // TMAX in Kelvin

if (DevTemp > TMAXK) begin
    $strobe("Warning: DevTemp = %e is more than TMAXK. Set to TMAXK.", DevTemp);
end

    DevTemp   = minx(DevTemp, TMAXK,  1.0e-2); //Limiting maximum temperature
```

- **2018enh2: Binning Equations for QME Effect**

  GF wants to add binning in all QME parameters. In the updated code we have added binning in QME parameters as follows:

  BSIM-IMG 102.9.1

```
// Quantum Mechanical Effects
if (QMTCENCV_i > 0.0) begin
    T4     = (qia + ETAQM * qba) / QM0;
    T5     = 1.0 + pow(T4, PQM);
    Tcen0  = TSI;
    Tcen   = Tcen0 / T5;
    coxeff = 3.9 * `EPS0 / (IMGTOXP * 3.9 / EPSROX1 + Tcen * QMTCENCV_i / epsratio);
end else begin
    coxeff = cox1P;
end
```

  BSIM-IMG 102.9.2

```
// Quantum Mechanical Effects
if (QMTCENCV_i > 0.0) begin
    T4     = (qia + ETAQM_i * qba) / QM0_i;
    T5     = 1.0 + pow(T4, PQM_i);
    Tcen0  = TSI;
    Tcen   = Tcen0 / T5;
    coxeff = 3.9 * `EPS0 / (IMGTOXP * 3.9 / EPSROX1 + Tcen * QMTCENCV_i / epsratio);
end else begin
    coxeff = cox1P;
end
```

- **2018enh3: DVTP formulation in line with BSIM4**

  We have implemented Drain-Induced-Vth Shift (DITS) module in the BSIM-IMG 102.9.2.

BSIM-IMG 102.9.2

```
// Scaling for DITS Parameters
DVTP0_i = DVTP0 + ADVTP0 * lexp(-Leff / BDVTP0);
DVTP1_i = DVTP1 + ADVTP1 * lexp(-Leff / BDVTP1);
```

```
dvth_dibl   = -(ETA0_t + ETAB_i * vbgx) * Theta_DIBL * (vdsx + ETA1_i * sqrt(vdsx + 0.01))
+ (DVTP0_i * Theta_DITS * pow(vdsx, DVTP1_i));
```

D. Description of bug-fixes

- **2018bug1: VA code syntax (& vs &&)**

  In following case, both & and && may be functionally equivalent, but ADI compiler doesn't like/support it.

  BSIM-IMG 102.9.1

```
// Front- and Back-Gate Workfunctions
if (NBG != 0.0 & !$param_given(PHIG2)) begin
    if (WELLTYPE == `ptype) begin
        PHIG2_i = PHIG2_i - 0.5 * BG0SUB + phisub;
    end else begin
        PHIG2_i = PHIG2_i + 0.5 * BG0SUB - phisub;
    end
end
```

  As per ADI request, we have used logical and (&&) instead of bitwise and (&) in the updated version.

BSIM-IMG 102.9.2

```
// Front- and Back-Gate Workfunctions
if (NBG != 0.0 && !$param_given(PHIG2)) begin
    if (WELLTYPE == `ptype) begin
        PHIG2_i = PHIG2_i - 0.5 * BG0SUB + phisub;
    end else begin
        PHIG2_i = PHIG2_i + 0.5 * BG0SUB - phisub;
    end
end
```

- **2018bug2: division by zero in Vdsx term**

The problem is coming from a sqrt(x) function that upon differentiation produces 1/sqrt(x). There is nothing that prevents x from evaluating being zero. In the updated version, we have updated dvth_dibl term to avoid divide by zero condition.

BSIM-IMG 102.9.1

```
dvth_dibl   = -(ETA0_t + ETAB_i * vbgx) * Theta_DIBL * (vdsx + ETA1_i * sqrt(vdsx));
```

BSIM-IMG 102.9.2

```
dvth_dibl   = -(ETA0_t + ETAB_i * vbgx) * Theta_DIBL * (vdsx + ETA1_i * sqrt(vdsx + 0.01))
              + (DVTP0_i * Theta_DITS * pow(vdsx, DVTP1_i));
```

- **2018bug3: Vthop is independent of PHIG1**

Vth equation implemented inside the OP section is independent of parameter PHIG1 in BSIM-IMG 102.9.2. We have added flat band voltage term in Vth equation to capture the effect of parameter PHIG1 in BSIM-IMG 102.9.2.

  - **BSIM-IMG 102.9.1**

```
// Threshold voltage operating point
A0     = (2.0 * `q * ni * TSI * TSI) / (epssi * Vtm);
k1     = cox1 / csi;
qth    = 1.0;
qsq1   = k1 * k1 * qth * qth - A0 * lexp(phib * 2.0);
qsqrt1 = sqrt(qsq1);
qcoth1 = (1.0 - qsqrt1 / 8.0) / (0.5 - qsqrt1 / 24.0);
T1     = (1.0 + lln(k1 * k1 * qth * qth + k1 * qth * qcoth1) - lln(A0)) * Vtm;

Vthop  = devsign * (T1 + dvth_all + DELVTRAND);
```

➢ **BSIM-IMG 102.9.2**

```
// Threshold voltage operating point
A0     = (2.0 * `q * ni * TSI * TSI) / (epssi * Vtm);
k1     = cox1 / csi;
qth    = 1.0;
qsq1   = k1 * k1 * qth * qth - A0 * lexp(phib * 2.0);
qsqrt1 = sqrt(qsq1);
qcoth1 = (1.0 - qsqrt1 / 8.0) / (0.5 - qsqrt1 / 24.0);
T1     = (1.0 + lln(k1 * k1 * qth * qth + k1 * qth * qcoth1) - lln(A0)) * Vtm;
Vfb    = PHIG1_i - (EASUB + Eg / 2.0);
Vthop  = Vfb + devsign * (T1 + dvth_all + DELVTRAND);
```

Results for PMOS (W=1.2e-7 L=2.48e-8**) @** Vds = - 0.1 V, Vfg from 1 to - 0.2 V.

	PHIG1=4.522872	PHIG1=4.822872
Vthop	**0.15333**	**0.45333**

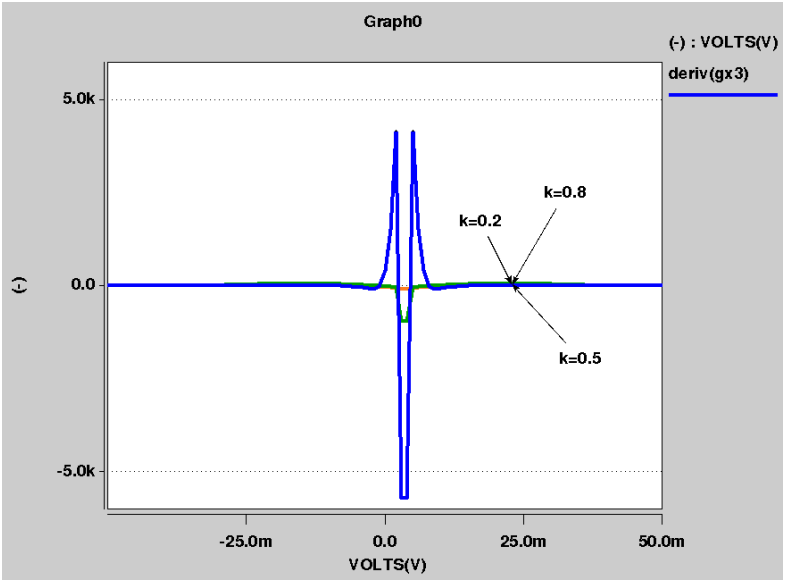
- **2018bug4: Division by zero in dvth_dibl term**

DVTP1_i = 0.5 (or any fractional value < 1), may cause issue in dvth_dibl term. In the updated version, we have modified the dvth_dibl term to avoid divide by zero condition.

➢ **BSIM-IMG 102.9.1**

$$dvth\_dibl = function(DVTP0\_i * Theta\_DITS * pow(vdsx, DVTP1\_i))$$

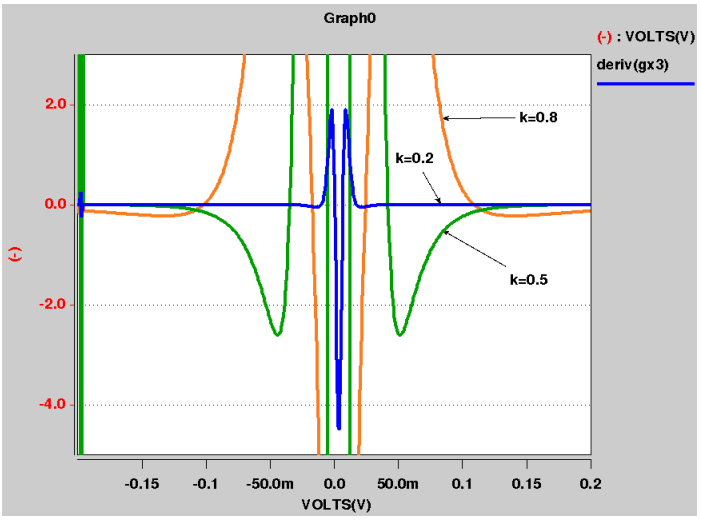
Gummel Symmetry Test: 3rd derivative of trans-conductance for DVTP1=0.2, 0.5, 0.8



- **BSIM-IMG 102.9.2**

$$dvth\_dibl = function(DVTP0\_i * Theta\_DITS * pow((vdsx + 0.01), DVTP1\_i))$$

Gummel Symmetry Test: 3rd derivative of trans-conductance for DVTP1=0.2, 0.5, 0.8



- **2018bug5: Add checks in code**

To avoid blow ups in terms “Theta_DIBL”, “Theta_SCE” and “DIBLfactor”, we have added

checks for DSUB_i, DVT1_i and DROUT_i, respectively.

➢ **BSIM-IMG 102.9.2**

```
if (DVT1_i <= 0.0) begin
    $strobe("Fatal: DVT1_i = %e is not positive.", DVT1_i);
    $finish(0);
end

if (DSUB_i <= 0.0) begin
    $strobe("Fatal: DSUB_i = %e is not positive.", DSUB_i);
    $finish(0);
end

if (DROUT_i <= 0.0) begin
    $strobe("Fatal: DROUT_i = %e is non-positive.", DROUT_i);
    $finish(0);
end
```

- **2018bug8: gcrg term initialization in "else" block**

We have initialized term gcrg to 0 in "else" block.

➢ **BSIM-IMG 102.9.1**

```
// NQS gate resistance Ref: BSIM4

if (NQSMOD == 1 && XRCRG1_i != 0.0) begin
    T0       = ueff * cox1 * Weff / Leff;
    IdsovVds = beta * qia * Moc / (Dmob * Dvsat * Dr);
    gcrg     = NF * XRCRG1_i * (IdsovVds + XRCRG2_i * Vtm * T0);


end
```

➢ **BSIM-IMG 102.9.2**

```
// NQS gate resistance Ref: BSIM4

if (NQSMOD == 1 && XRCRG1_i != 0.0) begin
    T0       = ueff * cox1 * Weff / Leff;
    IdsovVds = beta * qia * Moc / (Dmob * Dvsat * Dr);
    gcrg     = NF * XRCRG1_i * (IdsovVds + XRCRG2_i * Vtm * T0);
end else begin
    gcrg     = 0;
end
```

- **2018bug10: $simparam("gmin") implementation for minimum conductance**

  We have implemented minimum conductance as **$simparam("gmin")** instead of model parameter GDSMIN.

  - **BSIM-IMG 102.9.1**

```
if (sigvds > 0.0) begin
    I(di, si) <+ devsign * ids + (GDSMIN * V(di, si));
    I(di, si) <+ devsign * (igidl + Iii);
    I(si, di) <+ devsign * igisl;
    I(gi, si) <+ devsign * (igcs + igs);
    I(gi, di) <+ devsign * (igcd + igd);
end else begin
    I(si, di) <+ devsign * ids + (GDSMIN * V(si, di));
    I(si, di) <+ devsign * (igidl + Iii);
    I(di, si) <+ devsign * igisl;
    I(gi, di) <+ devsign * (igcs + igs);
    I(gi, si) <+ devsign * (igcd + igd);
end
```

  - **BSIM-IMG 102.9.2**

```
if (sigvds > 0.0) begin
    I(di, si) <+ devsign * ids + ($simparam("gmin",1e-12) * V(di, si));
    I(di, si) <+ devsign * (igidl + Iii);
    I(si, di) <+ devsign * igisl;
    I(gi, si) <+ devsign * (igcs + igs);
    I(gi, di) <+ devsign * (igcd + igd);
end else begin
    I(si, di) <+ devsign * ids + ($simparam("gmin",1e-12) * V(si, di));
    I(si, di) <+ devsign * (igidl + Iii);
    I(di, si) <+ devsign * igisl;
    I(gi, di) <+ devsign * (igcs + igs);
    I(gi, si) <+ devsign * (igcd + igd);
end
```