- The correlated thermal noise model was added. (IBM, Cadence)
  - To enable, uncomment the following line in the code
    //`define _TNOIMOD3_
    and set **TNOIMOD** parameter to 3.
  - References:
    1. tnoiMod=2 in BSIM4.7.0
    2. C. McAndrew, G. Coram, A. Blaum and O. Pilloud, ''Correlated Noise Modeling and Simulation,'' Proc. 2005 Workshop on Compact Modeling (WCM 2005), Anaheim, CA, May 8-12, 2005, pp. 40-45.
  - Model Parameters introduced:

```
parameter real TNOIC    = 3.5;
parameter real RNOIC    = 0.395;
parameter real SCALEN   = 1e5;
```

  - Node introduced:

```
// Internal node controlled by Correlated Thermal Noise Switch
`ifdef  _TNOIMOD3_
   electrical N;
`endif
```

  - Variables introduced:

```
//Variables controlled by Correlated Thermal Noise Switch
`ifdef  _TNOIMOD3_
   real B4SOItnoic;
   real B4SOIrnoic;
`endif

`ifdef  _TNOIMOD3_
   real npart_c;
   real eta, gamma, delta, epsilon;
   real Lvsat;
   real sid, sf;
   real ctnoi, B4SOInoiGd0, GammaGd0, C0;
`endif
```

  - Code added:

```
//Assignments controlled by Correlated Thermal Noise Switch   (~Line  2081)
`ifdef  _TNOIMOD3_
   B4SOItnoic = TNOIC;
   B4SOIrnoic = RNOIC;
`endif

//Correlated Thermal Noise by Navid, July 2013    (~Line 7481)
`ifdef _TNOIMOD3_
    eta   = 1.0 - B4SOIVdseff*B4SOIAbovVgst2Vtm ;
    T0    = 1.0 - eta;
    T1    = 1.0 + eta;
    T2    = T1 + 2.0*B4SOIAbulk*B4SOIvtm/(B4SOIVgsteff+1.0e-10);
    Lvsat = Leff*(1.0 + B4SOIVdseff /EsatL);
    T6    = Leff / Lvsat;
    gamma = T6*(0.5*T1 + T0*T0/(6.0*T2));
    T3    = T2*T2;
    T4    = T0*T0;
    T5    = T3*T3;
    delta   = ((T1/T3)-(5.0*T1 + T2)*T4/(15.0*T5) + T4*T4/(9.0*T5*T2))/(6.0*T6*T6*T6);
    T7    = T0/T2;
    epsilon = (T7 + T7*T7*T7/3.0)/(6.0*T6);
    T8    = B4SOIVgsteff / (EsatL);
    T8    = T8 * T8;
    npart_c = B4SOIrnoic * (1.0 + T8 * B4SOItnoic * Leff);
    ctnoi   = epsilon / sqrt( gamma * delta) * (2.5316 * npart_c);
    if (ctnoi > 1)          ctnoi=1;
    if (ctnoi < 0)          ctnoi=0;
```

```
    npart_beta  = B4SOIrnoia * (1.0 + T8 * B4SOItnoia * Leff);
    npart_theta = B4SOIrnoib * (1.0 + T8 * B4SOItnoib * Leff);
    gamma       = gamma * (3.0  * npart_beta  * npart_beta);
    delta       = delta * (3.75 * npart_theta * npart_theta);
    B4SOInoiGd0 = B4SOInf * beta * B4SOIVgsteff / (1.0 + gche * Rds);
    GammaGd0    = gamma * B4SOInoiGd0;
    sid         = fourkt * GammaGd0;
    C0          = B4SOInf * B4SOIcox * pParam_B4SOIweffCV * pParam_B4SOIleffCV;
    sf = (B4SOInoiGd0+1e-15)/sqrt(delta/gamma);
    I(di,si) <+  white_noise(sid*abs(1.0-ctnoi * ctnoi));
    I(N)     <+  V(N) * sf * SCALEN;
    I(N)     <+  white_noise(sid/(sf*sf*SCALEN*SCALEN));
    I(di,si) <+  ctnoi * V(N)*sf*SCALEN ;
    I(gi,si) <+  ddt(0.5 * C0 * SCALEN * V(N));
    I(gi,di) <+  ddt(0.5 * C0 * SCALEN * V(N));
`else
    $strobe("[BSIMSOI] Although the model selector TNOIMOD is set to 3, the new correlated thermal noise
             model is not activated in the Verilog-A   code.  Please uncomment \"`define _TNOIMOD3_\" in the
             beginning of the Verilog-A code.");
`endif
```

- dc and ac *DIBL* parameters have been decupled. (IBM)
  - Following parameters were introduced in CV model to decouple the DIBL effect for IV and CV:

```
parameter real ETA0CV        = ETA0;        // Subthreshold region DIBL coefficient for C-V
parameter real ETABCV        = ETAB;        // Subthreshold region DIBL coefficient for C-
parameter real STETA0CV      = STETA0;      // eta0cv shift factor related to stress effect on vth
parameter real LODETA0CV = LODETA0;   // eta0cv shift modification factor for stress effect
parameter real LETA0CV       = LETA0;       // Length dependence of eta0cv
parameter real LETABCV       = LETAB;       // Length dependence of etabcv
parameter real WETA0CV       = WETA0;       // Width dependence of eta0cv
parameter real WETABCV     = WETAB;     // Width dependence of etabcv
parameter real PETA0CV       = PETA0;       // Cross-term dependence of eta0cv
parameter real PETABCV      = PETAB;       // Cross-term dependence of etabcv
```

  - Variables introduced:

```
real B4SOIeta0cv;
real B4SOIetabcv;
real B4SOIsteta0cv;
real B4SOIlodeta0cv;
real pParam_B4SOIeta0cv;
real pParam_B4SOIetabcv;
real here_B4SOIeta0cv;

real deta0cv_lod;
real Vbseff_CV;
real Vbsh_CV;
real Vbsitf_CV, Vbs0_CV,Vbsmos_CV;
real PhiON_CV, PhiFD_CV, Vbs0t_CV, VthFD_CV;
real Phis_CV, sqrtPhis_CV, Xdep_CV;
real lt1_CV, ltw_CV;
real Theta0_CV, n_CV;
real VtgsFD_CV, ExpVtgsFD_CV, VgstFD_CV, ExpVgstFD_CV;
real VtgseffFD_CV, VgsteffFD_CV;
real tmp2_CV;
real Delt_vth_CV, DeltVthw_CV, DeltVthtemp_CV, DIBL_Sft_CV, DITS_Sft_CV;
real sqrtPhisExt_CV;
real Vth_CV, VTH_CV;
real Abulk0_CV;
```

  - Code added:

```
B4SOIeta0cv = ETA0CV;          (~Line  2004)
B4SOIetabcv = ETABCV;          (~Line  2005)
B4SOIsteta0cv = STETA0CV;      (~Line  2105)
B4SOIlodeta0cv = LODETA0CV; (~Line  2106)

(~Line  2655-2656):
```

```
pParam_B4SOIeta0cv   = B4SOIeta0cv   + LETA0CV   * Inv_L + WETA0CV   * Inv_W + PETA0CV   * Inv_LW;
pParam_B4SOIetabcv   = B4SOIetabcv   + LETABCV   * Inv_L + WETABCV   * Inv_W + PETABCV   * Inv_LW;

deta0cv_lod = B4SOIsteta0cv / pow(pParam_B4SOIkvth0, B4SOIlodeta0cv) * OD_offset;   (~Line  3215)

here_B4SOIeta0cv = pParam_B4SOIeta0cv + deta0cv_lod;    (~Line  3220)

here_B4SOIeta0cv = pParam_B4SOIeta0cv;        (~Line  3227)

if (B4SOIlodeta0cv <= 0.0)            (~Line  3791)
    $strobe("Warning: LODETA0CV = %g is not positive.",B4SOIlodeta0cv);

if (pParam_B4SOIeta0cv < 0.0)      (~Line  3916)
    $strobe("Warning: Eta0CV = %g is negative.", pParam_B4SOIeta0cv);

if (B4SOIsoiMod == 0) begin
        Vbsmos = Vbs;
        Vbsmos_CV = Vbs;     (~Line  4719)
end else begin …

/* DIBL_Sft_CV due to introduction of ETA0CV and ETABCV */       (~Lines  4868-75)
T3 = here_B4SOIeta0cv + pParam_B4SOIetabcv * Vbs0mos;
if (T3 < 1.0e-4) begin /* avoid  discontinuity problems caused by etabcv */
    T9 = 1.0 / (3.0 - 2.0e4 * T3);
    T3 = (2.0e-4 - T3) * T9;
end
DIBL_Sft_CV = T3 * theta0vb0 * Vds;

VthFD_CV = B4SOItype * here_B4SOIvth0+ (pParam_B4SOIk1ox * sqrtPhis  - pParam_B4SOIk1eff * sqrtPhi) *
Lpe_Vb- here_B4SOIk2ox * Vbs0mos- Delt_vth - DeltVthw  + (pParam_B4SOIk3 + pParam_B4SOIk3b *
Vbs0mos) * tmp2 + DeltVthtemp - DIBL_Sft_CV - DITS_Sft - DITS_Sft2;    (~Line  4890)

/* VtgseffFD_CV, PhiON_CV, PhiFD_CV, Vbs0_CV, Vbsitf_CV, and Vbsmos_CV calculation */   (~Lines  4969-
5037)

  /* VtgseffFD_CV calculation for PhiFD_CV */
  VtgsFD_CV = VthFD_CV - Vgs_eff;
  T10 = B4SOInofffd * Vtm;
  `DEXP((VtgsFD_CV - B4SOIvofffd)/ T10, ExpVtgsFD_CV)
   VtgseffFD_CV = T10 * ln(1.0 + ExpVtgsFD_CV);

  /* surface potential modeling at strong inversion: PhiON_CV */
  VgstFD_CV = Vgs_eff - VthFD_CV;
  `DEXP((VgstFD_CV - B4SOIvofffd)/ T10, ExpVgstFD_CV)
  VgsteffFD_CV = T10 * ln(1.0 + ExpVgstFD_CV);
   /* T1 = B4SOImoinFD*pParam_B4SOIk1eff*Vtm*Vtm;*/
  T1 = B4SOImoinFD*pParam_B4SOIk1ox*Vtm*Vtm;
  T2 = VgsteffFD_CV + 2*pParam_B4SOIk1eff*sqrt(phi);
  T0 = 1 + VgsteffFD_CV * T2 / T1;
  PhiON_CV = phi + Vtm * ln(T0) ;

  /* surface potential from subthreshold to inversion: PhiFD_CV */
  T0 = B4SOIcox / (B4SOIcox + 1.0/(1.0/B4SOIcsi + 1.0/Cbox));
  PhiFD_CV = PhiON_CV - T0 * VtgseffFD_CV;

  /* built-in potential lowering: Vbs0_CV */
  if (B4SOIfdMod == 0) begin
      T0 = -B4SOIdvbd1 * pParam_B4SOIleff / pParam_B4SOIlitl;
      T1 = B4SOIdvbd0 * (exp(0.5*T0) + 2*exp(T0));
      T2 = T1 * (vbi - phi);
      T3 = 0.5 * pParam_B4SOIqsi / B4SOIcsi;
      Vbs0t_CV=PhiFD_CV - T3 + B4SOIvbsa + T2;
      T0 = 1 + B4SOIcsi / Cbox;
      T3 = -B4SOIdk2b * pParam_B4SOIleff / pParam_B4SOIlitl;
      T5 = B4SOIk2b * (exp(0.5*T3) + 2*exp(T3));
      T1 = (B4SOIk1b - T5) / T0;
      T2 = T1 * Vesfb;
      T0 = 1.0/(1 + Cbox / B4SOIcsi);
```

```
        Vbs0_CV = T0 * Vbs0t_CV + T2;
    end else begin
        T0 = 1.0/(B4SOIcsi + Cbox + B4SOIcdsbs);
        T1 = -B4SOIdvbd1 * pParam_B4SOIleff / pParam_B4SOIlitl;
        T2 = B4SOIdvbd0 * (exp(0.5*T1) + 2*exp(T1));
        T3 = T2 * (Vds + B4SOIvsce);
        T4 = 0.5 * pParam_B4SOIqsi / B4SOIcsi;
        T5 = B4SOIcsi * T0 * (PhiFD_CV - T4 + B4SOIvbsa);
        T6 = B4SOIcdsbs * T0 * T3;
        Vbs0t_CV = T5 + T6;
        T7 = Cbox * T0 * Vesfb;
        Vbs0_CV = Vbs0t_CV + T7;
    end

    /* set lower bound of Vbs (from SPICE) to Vbs0_CV: Vbsitf_CV (Vbs at back interface) */
    if (B4SOIsoiMod == 2) begin
        Vbsitf_CV = Vbs0_CV + `OFF_Vbsitf;
        Vbs = Vbs0_CV + `OFF_Vbsitf;
    end else begin
        /* soiMod = 1 */
        T1 = Vbs - (Vbs0_CV + `OFF_Vbsitf) - 0.01;
        T2 = sqrt(T1*T1 + 0.0001);
        Vbsitf_CV = (Vbs0_CV + `OFF_Vbsitf) + 0.5 * (T1 + T2);
    end

    /* Based on Vbsitf_CV, calculate zero-field body potential for MOS: Vbsmos_CV */
    T1 = Vbs0t_CV - Vbsitf_CV - 0.005;
    T2 = sqrt(T1 * T1 + (2.5e-5));
    T3 = 0.5 * (T1 + T2);
    T4 = T3 * B4SOIcsi / pParam_B4SOIqsi; /* v3.2 */
    Vbsmos_CV = Vbsitf_CV - 0.5 * T3 * T4;
```

/* Vbsmos_CV, Vbsh_CV, and Vbseff_CV calculation */ **(~Lines 5063-80)**

```
    /* T2 is Vbsmos_CV limited above Vbsc=-5 */
    T0 = Vbsmos_CV + 5 - 0.001;
    T1 = sqrt(T0 * T0 - 0.004 * (-5));
    T2 = (-5) + 0.5 * (T0 + T1);
    /* Vbsh_CV is T2 limited below 1.5 */
    T0 = 1.5;
    T1 = T0 - T2 - 0.002;
    T3 = sqrt(T1 * T1 + 0.008 * T0);
    Vbsh_CV = T0 - 0.5 * (T1 + T3);
    /* Vbseff_CV is Vbsh_CV limited to 0.95*phi */
    T0 = 0.95 * phi;
    T1 = T0 - Vbsh_CV - 0.002;
    T2 = sqrt(T1 * T1 + 0.008 * T0);
    Vbseff_CV = T0 - 0.5 * (T1 + T2);
```

(~Line 5361):

```
- Vthzb = B4SOItype * here_B4SOIvth0 - Delt_vthzb - DeltVthwzb + pParam_B4SOIk3 * tmp2  + DeltVthtempzb;
+ Vthzb = B4SOItype * here_B4SOIvth0 - Delt_vthzb - DeltVthwzb + pParam_B4SOIk3 * tmp2_CV  +
DeltVthtempzb;
```

/* Calculation of Abulk0_CV by Pankaj in May 2012*/ **(~Lines 5521-62)**

```
if (pParam_B4SOIa0 == 0.0) begin // {
    Abulk0_CV = 1.0;
end else begin // }{
    T10 = pParam_B4SOIketa * Vbsh_CV;
    if (T10 >=-0.5) begin
        T11 = 1.0 / (1.0 + T10);
    end else begin /* added to avoid the problems caused by Keta */
    T12=-1.0/((1.0 - 0.5)*(1.0 - 0.5));
    T13=1.0/((1.0 - 0.5))+T12*0.5;
    T11=T12*T10+T13;
end

T10 = phi + pParam_B4SOIketas;
T13 = (Vbsh_CV * T11) / T10;
```

```
if (T13 < 0.50) begin
    T14 = 1.0 / sqrt(1.0-T13);
end else begin
    T11=1.0/(2*(1-0.50)*sqrt(1-0.50));
    T12=(1/sqrt(1.0 - 0.50))-T11*0.50;
    T14=T11*T13+T12;
end

    T10 = 0.5 * pParam_B4SOIk1ox * Lpe_Vb/ sqrt(phi + pParam_B4SOIketas);     /* v4.0 */
     T1 = T10 * T14;
    T9 = sqrt(pParam_B4SOIxj * Xdep_CV);
     tmp1 = Leff + 2.0 * T9;
    T5 = Leff / tmp1;
    tmp2 = pParam_B4SOIa0 * T5;
    tmp3 = pParam_B4SOIweff + pParam_B4SOIb1;
    tmp4 = pParam_B4SOIb0 / tmp3;
    T2 = tmp2 + tmp4;
    T6 = T5 * T5;
    T7 = T5 * T6;
    Abulk0_CV = 1 + T1 * T2;
end
if (Abulk0_CV < 0.01) begin
    T9 = 1.0 / (3.0 - 200.0 * Abulk0_CV);
     Abulk0_CV = (0.02 - Abulk0_CV) * T9;
end


/* v3.2 Separate VgsteffCV with noff */     (~Lines  6620-31)
/* New Vgst(Vgs_eff -Vth_CV) and n_CV */
Vgst=Vgs_eff-Vth_CV;
T10 = n_CV*Vtm;
VgstNVt = pParam_B4SOImstar * Vgst / T10;
- noff = n * pParam_B4SOInoff;
+ noff = n_CV * pParam_B4SOInoff;

/* New Vth (Vth_CV), sqrtPhis (sqrtPhis_CV), Vbseff (Vbseff_CV) */   (~Lines  6708-11)
Vth=Vth_CV;
sqrtPhis=sqrtPhis_CV;
Vbseff=Vbseff_CV;
```

- Some hard-coded material parameters in BSIMSOI4.4.0 now are model parameters.

Variable	B4SOImtrlMod=0	B4SOImtrlMod=1
eggbcp2	1.12	**EGGBCP2**
eggdep	1.12	**EGGDEP**
agb1	3.7622E-07	**AGB1**
bgb1	-3.1051E+10	**BGB1**
agb2	4.9758E-07	**AGB2**
bgb2	-2.357E+10	**BGB2**
agbc2n	3.4254E-07	**AGBC2N**
agbc2p	4.9723E-07	**AGBC2P**
bgbc2n	1.1665E+12	**BGBC2N**
bgbc2p	7.4567E+11	**BGBC2P**
Vtm00	0.026	**VTM00**

New model parameters

```
/* New parameters added corresponding to the various material properties for mtrlMod=1
parameter real EGGBCP2  = 1.12;              // Bandgap in Agbcp2 region
parameter real EGGDEP    = 1.12;             // Bandgap for gate depletion effect
parameter real AGB1       = 3.7622e-7;      // 'A' for Igb1 Tunneling current model
parameter real BGB1       = -3.1051e10;     // 'B' for Igb1 Tunneling current model
parameter real AGB2       = 4.9758e-7;      // 'A' for Igb2 Tunneling current model
parameter real BGB2       = -2.357e10;      // 'B' for Igb2 Tunneling current model
parameter real AGBC2N    = 3.4254e-7;      // NMOS 'A' for tunneling current model
parameter real AGBC2P     = 4.9723e-7;     // PMOS 'A' for tunneling current model
parameter real BGBC2N    = 1.1665e12;      // NMOS 'B' for tunneling current model
parameter real BGBC2P     = 7.4567e11;     // PMOS 'B' for tunneling current model
parameter real VTM00      = 0.026;          // Hard coded 25 degC thermal voltage

/* New variables added corresponding to the various material properties for mtrlMod=1
real B4SOIEGGBCP2;
real B4SOIEGGDEP;
real B4SOIAGB1;
real B4SOIBGB1;
real B4SOIAGB2;
real B4SOIBGB2;
real B4SOIAGBC2N;
real B4SOIAGBC2P;
real B4SOIBGBC2N;
real B4SOIBGBC2P;
real B4SOIVTM00;

B4SOIEGGBCP2=EGGBCP2;    (~Lines  2254-64)
B4SOIEGGDEP=EGGDEP;
B4SOIAGB1=AGB1;
B4SOIBGB1=BGB1;
B4SOIAGB2=AGB2;
B4SOIBGB2=BGB2;
B4SOIAGBC2N=AGBC2N;
B4SOIAGBC2P=AGBC2P;
B4SOIBGBC2N=BGBC2N;
B4SOIBGBC2P=BGBC2P;
B4SOIVTM00=VTM00;

/*These constants are replaced with model parameters */
- eggbcp2 = 1.12;
- eggdep = 1.12;
- agb1 = 3.7622e-7;
- bgb1 = -3.1051e10;
- agb2 = 4.9758e-7;
- bgb2 = -2.357e10;
- agbc2n = 3.42537e-7;
- agbc2p = 4.97232e-7;
- bgbc2n = 1.16645e12;
 - bgbc2p = 7.45669e11;
 - Vtm00 = 0.026;

 + eggbcp2 = B4SOIEGGBCP2;  (~Lines  2294-2303)
 + eggdep = B4SOIEGGDEP;
 + agb1 = B4SOIAGB1;
 + bgb1 = B4SOIBGB1;
 + agb2 = B4SOIAGB2;
 + bgb2 = B4SOIBGB2;
 + agbc2n = B4SOIAGBC2N;
 + agbc2p = B4SOIAGBC2P;
 + bgbc2n = B4SOIBGBC2N;
 + bgbc2p = B4SOIBGBC2P;

 + Vtm00=B4SOIVTM00;    (~Line  2482)
```

- When values of NRS and NRD are zero, the source/drain conductance is set to 1.0e3 instead of zero. (Proplus) **(~Lines 3308-19)**

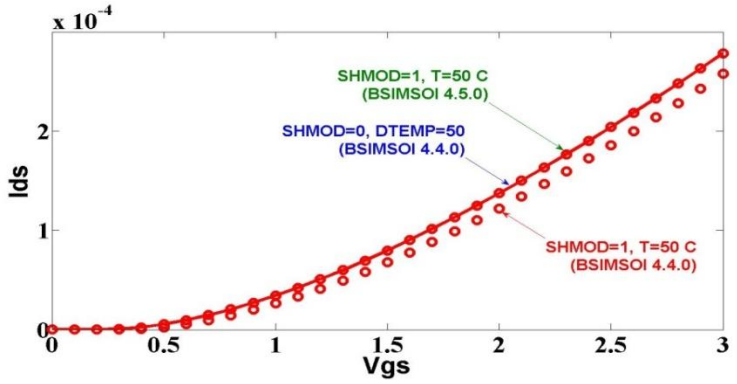
BSIMSOI4.5.0	BSIMSOI4.4.0

real B4SOIsourceResistance;
real B4SOIdrainResistance;
 	
/* process source/drain series resistance */
B4SOIdrainResistance = B4SOIsheetResistance * B4SOIdrainSquares;
if (B4SOIdrainResistance > 0.0)
B4SOIdrainConductance = 1.0 / B4SOIdrainResistance;
else
B4SOIdrainConductance = 1.0e3;
 	
B4SOIsourceResistance = B4SOIsheetResistance * B4SOIsourceSquares;
if (B4SOIsourceResistance > 0.0)
B4SOIsourceConductance = 1.0 / B4SOIsourceResistance;
else
B4SOIsourceConductance = 1.0e3;
	/* process source/drain series resistance */
	B4SOIdrainConductance = B4SOIsheetResistance * B4SOIdrainSquares;
	if (B4SOIdrainConductance > 0.0)
	B4SOIdrainConductance = 1.0 / B4SOIdrainResistance;
	else
	B4SOIdrainConductance = 0;
	
	B4SOIsourceConductance= B4SOIsheetResistance * B4SOIsourceSquares;
	if (B4SOIsourceConductance > 0.0)
	B4SOIsourceConductance = 1.0 / B4SOIsourceResistance;
	else
	B4SOIsourceConductance = 0;

- BSIMSOI4.5.0 limits "ni" for SHMOD=1 to avoid numerical problems at extremely low temperatures. (Agilent) **(~Line 4311)**

BSIMSOI4.5.0	BSIMSOI4.4.0
T6=21.5565981 - Eg / (2.0 * Vtm); if (T6 >-`EXPL_THRESHOLD) T4 = exp(T6); else T4=exp(-`EXPL_THRESHOLD); ni = T3 * T4;	T4 = exp(21.5565981 - Eg / (2.0 * Vtm)); ni = T3 * T4;

- Inconsistency in drain current when self-heating is on has been addresses. (Agilent)
  - Solution: in BSIMSOI4.4.0 for SHMOD=1 (self heating on), some variables like *Vtm*, *ni*, *vbi*, and *phi* are recalculated. To be consistent, in BSIMSOI4.5.0, **here_B4SOIvfb** and **here_B4SOIvth0** are recalculated as well. This fixes the problem.



- **here_B4SOIvfb and here_B4SOIvth0 Recalculation Code (~Lines 4501-67)**

```
if ($param_given(K1) || $param_given(K2)) begin // {
    if (!$param_given(K1)) begin
        $strobe( "Warning: k1 should be specified with k2.");
        pParam_B4SOIk1 = 0.53;
    end
    if (!$param_given(K2)) begin
        $strobe( "Warning: k2 should be specified with k1.");
        pParam_B4SOIk2 = -0.0186;
    end
    if ($param_given(XT))
        $strobe( "Warning: xt is ignored because k1 or k2 is given.");
    if ($param_given(VBX))
        $strobe( "Warning: vbx is ignored because k1 or k2 is given.");
    if ($param_given(VBM))
        $strobe( "Warning: vbm is ignored because k1 or k2 is given.");
    if ($param_given(GAMMA1))
        $strobe( "Warning: gamma1 is ignored because k1 or k2 is given.");
    if ($param_given(GAMMA2))
        $strobe( "Warning: gamma2 is ignored because k1 or k2 is given.");
end else begin // }{
    if (!$param_given(VBX)) begin
        if (B4SOImtrlMod)
            T0 = `Charge_q / (2.0 * epssub) * 1.0e6;
        else
            T0 = 7.7348e-4; /* constant from v4.3.0 and earlier */
        pParam_B4SOIvbx = phi - T0 * pParam_B4SOInpeak * pParam_B4SOIxt * pParam_B4SOIxt;
    end
    if (pParam_B4SOIvbx > 0.0)
        pParam_B4SOIvbx = -pParam_B4SOIvbx;
    if (pParam_B4SOIvbm > 0.0)
        pParam_B4SOIvbm = -pParam_B4SOIvbm;
    if (!$param_given(GAMMA1))
        pParam_B4SOIgamma1 = sqrt2qeps * sqrt(pParam_B4SOInpeak) / B4SOIcox;
    if (!$param_given(GAMMA2))
        pParam_B4SOIgamma2 = sqrt2qeps * sqrt(pParam_B4SOInsub) / B4SOIcox;
    T0 = pParam_B4SOIgamma1 - pParam_B4SOIgamma2;
    T1 = sqrt(phi - pParam_B4SOIvbx) - sqrtPhi;
    T2 = sqrtPhi * (sqrt(phi - pParam_B4SOIvbm)- sqrtPhi);
    T3= T0 * T1 / (2.0 * T2 + pParam_B4SOIvbm);
    here_B4SOIk2=here_B4SOIk2-pParam_B4SOIk2 + T3;
    pParam_B4SOIk1 = pParam_B4SOIgamma2 - 2.0 * here_B4SOIk2 * sqrt(phi - pParam_B4SOIvbm);
end // }

T0 = pParam_B4SOIweff + pParam_B4SOIk1w2;
if (T0 < 1e-8)
    T0 = 1e-8;
pParam_B4SOIk1eff = pParam_B4SOIk1 * (1 + pParam_B4SOIk1w1/T0);
/* v4.1 */
if (!$param_given(VFB)) begin
    if ($param_given(VTH0) || $param_given(VTHO)) begin
        here_B4SOIvfb = here_B4SOIvfb - pParam_B4SOIvfb + B4SOItype * here_B4SOIvth0 - phi -
                pParam_B4SOIk1eff* sqrtPhi;
    end else begin
        here_B4SOIvfb=here_B4SOIvfb;
    end
end

if (!$param_given(VTH0)) begin
    here_B4SOIvth0= B4SOItype * (here_B4SOIvfb + phi + pParam_B4SOIk1eff*sqrtPhi);
end
```

- GISL/GIDL Model for gidlMod=0 has been modified. (Synopsys) **(~Line 5795-5838)**

BSIMSOI4.5.0

```
/* GISL */
if ((agisl <= 0.0) || (bgisl <= 0.0) ||   (cgisl < 0.0))
begin
            Igisl = 0.0;
end else begin
            T1 = hypsmooth(T1, 1.0E-2);                    T2
= bgisl / (T1+1.0E-3);
      Igisl = wdios * agisl * T1 * exp(-T2);
            T4 = Vbs * Vbs;
              T5 = -Vbs * T4;
              T6 = cgisl + abs(T5) + 1.0E-9;
                        T7 = hypsmooth(T5 / T6, 1.0E-6) - 1.0E-6;
                        Igisl = Igisl * T7;
        end
/* End of GISL */
```

BSIMSOI4.4.0

```
/* GISL */
if ((agisl <= 0.0) || (bgisl <= 0.0) ||  (T1 <= 0.0) || (cgisl < 0.0) ||
(Vbs > 0.0) )
            begin
                        Igisl = 0.0;
          end else begin
          T2 = bgisl / T1;
          if (T2 < `EXPL_THRESHOLD) begin
                Igisl = wdios * agisl * T1 * exp(-T2);
           end else begin
                T3 = wdios * agisl * `MIN_EXPL;
                Igisl = T3 * T1 ;
           end
           if (cgisl >= `MIN_EXPL) begin
                T4 = Vbs * Vbs;
                T5 = -Vbs * T4;
                T6 = cgisl + T5;
                T7 = T5 / T6;
                Igisl = Igisl * T7;
              end
  end
/* End of GISL */
```

  - Similar changes for GIDL.

- GISL/GIDL Model for gidlMod=1 has been modified. (Agilent) **(~Lines 5847-87)**

BSIMSOI4.5.0

```
/* GISL */
if ((agisl <= 0.0) || (bgisl <= 0.0)||   (cgisl < 0.0)  ) begin
      Igisl = 0.0;
end else begin
      T1 = hypsmooth(T1, 1.0E-2);
      T2 = bgisl / (T1+1.0E-3);
      Igisl = wdios * agisl * T1 * exp(-T2);
      T4 = Vbs - fgisl;
      if (T4 >= -1.0/`EXPL_THRESHOLD)
            T5 = -kgisl * `EXPL_THRESHOLD;
     else
           T5 = kgisl/T4;
     T6 = exp(T5);
     Igisl = Igisl * T6;'
end
/* End of GISL */
```

BSIMSOI4.4.0

```
/* GISL */
if ((agisl <= 0.0) || (bgisl <= 0.0)|| (cgisl < 0.0)) || (T1 <= 0.0) ||
begin
      Igisl = 0.0;
end else begin
      T2 = bgisl / T1;
       if (T2 < `EXPL_THRESHOLD) begin
            Igisl = wdios * agisl * T1 * exp(-T2);
        end else begin
            T3 = wdios * agisl * `MIN_EXPL;
            Igisl = T3 * T1 ;
        end
        T4 = Vbs - fgisl;
        if (T4 == 0)
            T5 = `EXPL_THRESHOLD;
        else
            T5 = kgisl/T4;
         if (T5<`EXPL_THRESHOLD)
             T6 = exp(T5);
         else
             T6 = `MAX_EXPL;
          Igisl = Igisl * T6;
end
/* End of GISL */
```

  - Similar changes for GIDL.

- Overflow in exponential has been avoided in several places. (Agilent)

BSIMSOI4.4.0	BSIMSOI4.5.0	~Line

if ((pParam_B4SOIvrec0 - vsbs) < 1e-3) begin
/* v2.2.3 bug fix */
T1 = 1e3;
T0 = -vsbs / NVtmr * pParam_B4SOIvrec0 * T1;
T11 = -exp(T0);
end
	if ((pParam_B4SOIvrec0 - vsbs) < 1e-3) begin
	/* v2.2.3 bug fix */
	T1 = 1e3;
	T0 = -vsbs / NVtmr * pParam_B4SOIvrec0 * T1;
	`DEXP(T0,T11); //SDM fix
	T11 = -T11;
	end
		5939

if ((pParam_B4SOIvrec0d - vdbd) < 1e-3) begin /* v2.2.3 bug fix */ T1 = 1e3; T0 = -vdbd / NVtmr * pParam_B4SOIvrec0d * T1; T11 = -exp(T0); end	if ((pParam_B4SOIvrec0d - vdbd) < 1e-3) begin /* v2.2.3 bug fix */ T1 = 1e3; T0 = -vdbd / NVtmr * pParam_B4SOIvrec0d * T1; `DEXP(T0,T11); T11 = -T11; end	5967
if ((pParam_B4SOIvtun0 - vsbs) < 1e-3) begin /* v2.2.3 bug fix */ T1 = 1e3; T0 = -vsbs / NVtm2 * pParam_B4SOIvtun0 * T1; T1 = exp(T0); T3 = WsTsi * jtuns; Ibs4 = T3 * (1- T1); end	if ((pParam_B4SOIvtun0 - vsbs) < 1e-3) begin /* v2.2.3 bug fix */ T1 = 1e3; T0 = -vsbs / NVtm2 * pParam_B4SOIvtun0 * T1; `DEXP(T0,T1); T3 = WsTsi * jtuns; Ibs4 = T3 * (1- T1); end	6056
if ((pParam_B4SOIvtun0d - vdbd) < 1e-3) begin /* v2.2.3 bug fix */ T1 = 1e3; T0 = -vdbd / NVtm2 * pParam_B4SOIvtun0d * T1; T1 = exp(T0); T3 = WdTsi * jtund; Ibd4 = T3 * (1- T1); end	if ((pParam_B4SOIvtun0d - vdbd) < 1e-3) begin /* v2.2.3 bug fix */ T1 = 1e3; T0 = -vdbd / NVtm2 * pParam_B4SOIvtun0d * T1; `DEXP(T0,T1); T3 = WdTsi * jtund; Ibd4 = T3 * (1- T1); end	6074

- In Capmod=3, the XDC calculation has been modified according to the technical manual: (Cadence) **(~Lines 7084-87)**

(ii) X_{DC} of inversion charge

The inversion charge layer thickness can be formulated as

$$X_{DC} = \frac{ADOS \times 1.9 \times 10^{-9}\ \mathrm{m}}{1 + \left(\frac{V_{gsteff} + 4(VTH0 - VFB - \Phi_s)}{2TOXP} \right)^{0.7 \times BDOS}}$$

BSIMSOI4.5.0	BSIMSOI4.4.0
tmp = exp(B4SOIbdos*0.7 * ln(T0)); T1 = 1.0 + tmp; Tcen = B4SOIados*1.9e-9 / T1;	tmp = exp(0.7 * ln(T0)); T1 = 1.0 + tmp; Tcen = 1.9e-9 / T1;

- BSIM4.5.0 makes sure that "pwr" is positive in white_noise(pwr,name) in thermal noise model implementation. (Agilent)

Line 7631
```
-  I(d, di) <+ white_noise( fourkt * gdnoise, "rd");
+ I(d, di) <+ white_noise( abs(fourkt * gdnoise), "rd");
```

Line 7637
```
-  I(s, si) <+ white_noise( fourkt * gsnoise, "rs");
+ I(s, si) <+ white_noise( abs(fourkt * gsnoise), "rs");
```

Line 7682
```
-  I(db,di) <+ white_noise( 2 * `Charge_q * B4SOInoif * Ibd, "ibd");
+ I(db,di) <+ white_noise( 2 * `Charge_q * B4SOInoif * abs(Ibd), "ibd");
```

Line 7683
```
-  I(sb,si) <+ white_noise( 2 * `Charge_q * B4SOInoif * Ibs, "ibs");
+ I(sb,si) <+ white_noise( 2 * `Charge_q * B4SOInoif * abs(Ibs), "ibs");
```

Line:7686
```
- I(gi,di) <+ white_noise( 2 * `Charge_q * (B4SOIIgd + B4SOIIgcd), "igd");
+ I(gi,di) <+ white_noise( 2 * `Charge_q * abs(B4SOIIgd + B4SOIIgcd), "igd");
```

Line 7687
```
- I(gi,si) <+ white_noise( 2 * `Charge_q * (B4SOIIgs + B4SOIIgcs), "igs");
+ I(gi,si) <+ white_noise( 2 * `Charge_q * abs(B4SOIIgs + B4SOIIgcs), "igs");
```

Line 7688
```
- I(gi,b)  <+ white_noise( 2 * `Charge_q * B4SOIig, "igb");
+ I(gi,b)  <+ white_noise( 2 * `Charge_q * abs(B4SOIig), "igb");
```

Line 7716
```
- I(g, gm) <+ white_noise( fourkt * B4SOIgrgeltd, "rg");
+ I(g, gm) <+ white_noise( abs(fourkt * B4SOIgrgeltd), "rg");
```

Line 7728
```
- I(gm,gi) <+ white_noise(fourkt * B4SOIgrgeltd / (T0 * T0), "rg");
+ I(gm,gi) <+ white_noise( abs(fourkt * B4SOIgrgeltd / (T0 * T0)), "rg");
```

Line 7736
```
- I(b, db) <+ white_noise(fourkt * B4SOIgrbdb, "rbdb");
+ I(b, db) <+ white_noise( abs(fourkt * B4SOIgrbdb), "rbdb");
```

Line 7737
```
- I(b, sb) <+ white_noise(fourkt * B4SOIgrbsb, "rbsb");
+ I(b, sb) <+ white_noise( abs(fourkt * B4SOIgrbsb), "rbsb");
```

- Parameter limiting
  - NTNOI is now limited to positive values only. NTNOI affects "thernalNoiseContrib" in this equation:
    Lines 7446, 7476, and 7536: I(di,si) <+ white_noise(fourkt * thermalNoiseContrib, "id");
  - NOIF is now limited to positive values only. NOIF appears as B4SOInoif in these equations:
    Line 7682: I(db,di) <+ white_noise(2 * `Charge_q * B4SOInoif * abs(Ibd), "ibd");
    Line 7683: I(sb,si) <+ white_noise(2 * `Charge_q * B4SOInoif * abs(Ibs), "ibs");

- The thermal noise contribution due to rbody has been included. (Cadence) **(~Line 7676)**

```
if ((B4SOIbodyMod == 0) || (B4SOIbodyMod == 2))
        V(b, p)  <+ 0;
else    begin
        I(b, p)  <+ B4SOItype * Ibp;
        I(b, p)  <+ white_noise(fourkt*abs(Ibp)/(abs(vbp)+1.0e-9));
end
```

- "ExpVgst" calculation is now protected against overflows in two places: (Agilent, Cadence)

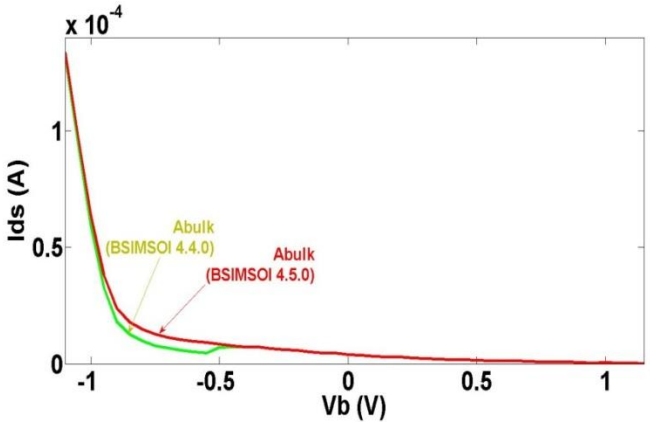
1)
```
if (B4SOIvgstcvMod == 0)  (
begin
        if ((VgstNVt > -`EXPL_THRESHOLD) && (VgstNVt < `EXPL_THRESHOLD))
        begin
                - ExpVgst = ExpVgst * ExpVgst;              (~Line 6635)
                + ExpVgst = exp(VgstNVt) * exp(VgstNVt);  (~Line 6636)
```

2)
```
  + if ((VgstNVt > -`EXPL_THRESHOLD)&&(VgstNVt<`EXPL_THRESHOLD)) (~Line 6648)
       ExpVgst =exp(VgstNVt/(pParam_B4SOImstar*pParam_B4SOInoff));
      …
  …

  + end    (~Line 6658)
```

- The calculation of A_{bulk} has been updated to avoid non-monotoinc behavior at high body-bias. **(~Line 5448)**

$$A_{bulk} = 1 + \left(\frac{K_{1ox} \cdot \sqrt{1 + LPEB / L_{eff}}}{2\sqrt{(\phi_s + Ketas) - \frac{V_{bsh}}{1 + Keta \cdot V_{bsh}}}} \left(\frac{A_0 L_{eff}}{L_{eff} + 2\sqrt{T_{si} X_{dep}}} \left(1 - A_{gs} V_{gsteff} \left(\frac{L_{eff}}{L_{eff} + 2\sqrt{T_{si} X_{dep}}} \right)^2 \right) + \frac{B_0}{W_{eff}' + B_1} \right) \right)$$



BSIMSOI4.5.0 (~Line 5448)	BSIMSOI4.4.0

if (T10 >=-0.5) begin
T11 = 1.0 / (1.0 + T10);
end else begin // added to avoid the problems caused by Keta
T12=-1.0/((1.0 - 0.5)*(1.0 - 0.5));
T13=1.0/((1.0 - 0.5))+T12*0.5;
T11=T12*T10+T13;
end
	if (T10 >= -0.9) begin
	T11 = 1.0 / (1.0 + T10);
	end else begin // added to avoid the problems caused by Keta
	T12 = 1.0 / (0.8 + T10);
	T11 = (17.0 + 20.0 * T10) * T12;
	end

BSIMSOI4.5.0 (~Line 5479)	BSIMSOI4.4.0
if (T13 < 0.50) begin T14 = 1.0 / sqrt(1.0-T13); end else begin T11=1.0/(2*(1-0.50)*sqrt(1-0.50)); T12=(1/sqrt(1.0 - 0.50))-T11*0.50; T14=T11*T13+T12; end	if (T13 < 0.96) begin T14 = 1 / sqrt(1-T13); end else begin T11 = 1.0 / (1.0 - 1.0593220339 * T13); T14 = (6.0169491525 - 6.3559322034 * T13) * T11; end

- Bug Fixes:

1) **(~Line 3095)**

```
-  T2 = sqrt(pParam_B4SOIphi * (pParam_B4SOIphi - pParam_B4SOIvbm)) - pParam_B4SOIphi;
+ T2 = pParam_B4SOIsqrtPhi * (sqrt(pParam_B4SOIphi - pParam_B4SOIvbm)- pParam_B4SOIsqrtPhi);
```

2) **(~Line 4028)**

```
-  if (B4SOIsii2 < 0.0)          $strobe("Warning:  Sii2 = %g is negative.", B4SOIsii1);
+ if (B4SOIsii2 < 0.0)          $strobe("Warning:  Sii2 = %g is negative.", B4SOIsii2);
```

3) NF dependence in calculation of Ig_agbcp2 (Agilent) **(~Line 6358)**

```
- T11 = T11 * B4SOIagbcp2 * pParam_B4SOIoxideRatio/B4SOInf;
+ T11 = T11 * B4SOIagbcp2 * pParam_B4SOIoxideRatio;
```

Navid Paydavosi
November 2013