# Summary of changes from BSIM-CMG 111.1.1 to BSIM-CMG_112.0.0

BSIM Group, UC Berkeley

A. Pampori (pampori@berkeley.edu), D. Rajasekharan, and Shivendra Singh Parihar

## A. Summary of bug fixes

1. 2021bug1(IBM): Technical manual update
2. 2021bug2(ADI): Change BETA OP type to `OPM
3. 2021bug3(ADI, UCB): Appropriate bound for parameters
4. 2021bug4(Cadence, ADI, UCB): Protection for Junction Capacitance Grading Coefficient
5. 2021bug5(ADI): Incorrect derivative of ln_one_plus_exp function for x < -37
6. 2021bug6(UCB): Corrections in Teff implementation in cryogenic model (CRYOMOD=1 and 2)
7. 2021bug7(UCB): Non-zero threshold shift at T=Tnom for Tnom < 210K in CRYOMOD=2
8. 2021bug8(UCB): Impact ionization current missing in ISEFF/IDEFF ops for BULKMOD=0
9. 2021bug9(UCB): Default value of model parameter 'LSP' depends on parameters 'L' and 'XL'
10. 2021bug10(ADI): Conditional assignment of OP variables
11. 2021bug11(Silvaco): Correction in technical manual
12. 2021bug12(UCB): S/D velocity saturation model corrections: a) correct for voltage polarity reversal and b) make delta factor a function of voltage
13. 2021bug13(UCB): Units and typos correction in code/manual
14. 2021bug14(ADI): Remove unused macros
15. 2021bug15(ADI): Limit parameter MVSRSD's range
16. 2022bug1(Cadence): Allow minr = 0
17. 2022bug2(Cadence, ADI): Reverse temperature sweep's order in QA

**18.** 2022bug3(Silvaco, ADI): No key for SmartSpice in QA perl script

**19.** 2022bug4(UCB): GIDLMOD=2 parameters missing in parameter table of manual

**20.** 2022bug5(Keysight): Typo in GIDLMOD=2

**21.** 2022bug6(Keysight, ADI, UCB): Divide by zero case in CRYOMOD=2

**22.** 2022bug7(Cadence): Code implementation issue in junction_cap macro

**23.** 2022bug8(ADI): Use local variables in macros

**24.** 2022bug9(Cadence, ADI): Bound DTLOW and DTLOW1 to be > 0

**25.** 2022bug10(UCB): Typos in manual

**26.** 2022bug12(UCB, ADI, Cadence): Inaccuracy in exp(x) – 1 calculation in Verilog-A for very small x

**27.** 2022bug14(ADI): Change terminal “b” in QA with “e” for consistency with the code

**28.** 2023bug1(Cadence): Discontinuity in Cds at Vds =0 for small Vgs (accumulation region)

**29.** 2023bug2(ADI): Use bias-independent if-else conditions for warnings

**30.** 2023bug3(Cadence, UCB): Discontinuity in Cds at Vds=0 and a large negative gate voltage

**31.** 2023bug4(UCB): Incorrect units of binning parameters

**32.** 2024bug1(ADI): Correct the range of parameter TYPE.

**33.** 2024bug2(UCB): Gummel symmetry issue in Cryogenic Coulomb scattering model.

**34.** 2024bug3(UCB): Sign correction for variable xdpart in NQSMOD=2

**35.** 2024bug4(ADI): Order-of-operations problem in macros gammafunc and smoothmaxx2.

**36.** 2024bug5(ADI): Superfluous assignment to qm_ln.

**37.** 2024bug6(ADI): Use updated hypsmooth function.

**38.** 2024bug7(ADI): Initialize gate current variables.

**39.** 2024bug8(IBM): Set a lower limit of zero to phib to prevent negative square root evaluation.

**40.** 2024bug9(IBM,ADI): Check for WGAAeff and Leff bounds right after their evaluation to prevent a divide-by-zero error.

## B. Summary of Enhancements

1. 2021enh1(IBM): Max NGAA allowed from 3 to 6
2. 2021enh2(IBM, UCB): Enhance Gate to Substrate Capacitance model for GAA
3. 2021enh3(ADI): ln(1.0 + `lexp(x)) replacement with new function "ln_one_plus_exp"
4. 2021enh4(TSMC): Width binning parameters due to etch in GAA
5. 2021enh5(UCB): Cryogenic Temperature Model
6. 2020enh6(UCB): Improve accuracy of C-V fitting for high Vd and Vg
7. 2021enh7(UCB): Improve the accuracy of I-V near Vdsat at high Vg: S/D velocity saturation model
8. 2021enh8(UCB): Parasitic substrate GIDL/GISL model
9. 2022enh1(ADI): Add QA test to check any missing parameters
10. 2022enh2(UCB): Trap-assisted tunneling (TAT) GIDL model
11. 2022enh3(UCB): Add cryogenic model parameter extraction procedure in manual.
12. 2023enh1(IBM): Improve cryogenic model parameter extraction procedure in the manual.
13. 2023enh1: Reinstate the GAA functionality.
14. 2024enh2(UCB): Thin Film Transistor Model.
15. 2024enh3(ADI, UCB): Add QA tests for all parameters.
16. 2024enh4(Rapidus): Improved gds fitting in low gate overdrive.
17. 2024enh5(ADI,UCB): Change parameter case according to VA standard.
18. 2024enh6(UCB): Flexibility for capacitance tuning in the saturation region.
19. 2024enh7(ADI): Removing NF from parameters/nmosDefaultParameters and parameters/pmosDefaultParameters in the QA package, and redefining the parameter check with NF as an instance parameter.
20. 2024enh8(IBM): Improved Self-Heating model for GAAFETs.
21. 2024enh9(Intel): Additional parameters to model mobility degradation due to hot carriers.
22. 2024enh10(IBM): Improved Self-Heating model for GAAFETs
23. 2024enh12(CMC Policy): Removed TFT feature from 112.0.0beta3
24. 2024enh13(ADI, QA subcommittee): Extended QA to evaluate non-default

parameter values.

## C. Description of bug fixes

1. **2021bug1(IBM): Technical manual update**

   Description of GEOMOD=4 was missing in the manual. We have added them in the 111.2.0_beta0_1.

   GEOMOD = 4: Unified model

2. **2021bug2(ADI): Change BETA OP type to `OPM**

   In BSIM-CMG111.1.0, BETA op variable was declared as `OPP. Since BETA op should scale with $mfactor, it was changed to `OPM in BSIM-CMG 111.2.0beta0_1.

```
`define OPM(nam,uni,des)    (* units=uni, desc=des, multiplicity="multiply" *)  real    nam;
```

3. **2021bug3(ADI, UCB): Appropriate bound for parameters**

   Parameters NI0SUB, BG0SUB and NC0SUB that should physically be positive were allowed to be both positive and negative. These have been restricted to positive values in BSIMCMG111.2.0beta0_1.

```
`MPRoz(NI0SUB, 1.1e16, "/m^3", "Intrinsic carrier constant at 300.15K")
`MPRoz(BG0SUB, 1.12, "eV", "Bandgap of substrate at 300.15K")
`MPRoz(NC0SUB, 2.86e25, "/m^3", "Conduction band density of states")
```

4. **2021bug4(Cadence, UCB): Protection for Junction Capacitance Grading Coefficient**

   Some junction capacitance grading coefficients of BSIMCMG 111.1.0 like “MJS, MJD, MJSWS, MJSWD, MJSWGS, MJSWGD, MJS2, MJD2, MJSWS2, MJSWD2, MJSWGS2, MJSWGD2” had no protection mechanism when they were set to one. If they are set to one, numerical issues will occur due to the presence of terms like 1/(1-MJ) and 1/(1 - MJ2) in the Qej calculation in junction_cap macro.

   There are two types of cases where 1 / (1 - MJ) (or 1/(1 - MJ2)) appears in the Qej calculation depending on the term T1 = vex/PB.

   1. For T1 < 0.9, Qej involves the terms like (1.0 - arg * sarg) / (1.0 - MJ) where sarg = pow(arg, -MJ)

For MJ = 1, (1.0 - arg * sarg) / (1.0 - MJ) takes 0/0 form. Using the L'Hopital's rule, the limit of this term is a finite number as

$$\lim_{MJ \to 1} \frac{1 - \arg^{1-MJ}}{1 - MJ} = -\ln(\arg)$$

Therefore, (1.0 - arg * sarg) / (1.0 - MJ) terms are replaced with - 'lln(arg) for MJ=1. Similarly, for MJ2=1.

```
if (vex > vec) begin \
    arg = 1.0 - T1; \
    if (MJ == 0.5) begin \
        sarg = 1.0 / sqrt(arg); \
    end else begin \
        sarg = pow(arg, -MJ); \
    end \
    if (MJ != 1) begin \
        Qej = PB * Cz * (1.0 - arg * sarg) / (1.0 - MJ); \
    end else begin \
        Qej = -PB * Cz * `lln(arg); \
    end \
end else begin \
    arg = 1.0 - vec / PB; \
    if (MJ == 0.5) begin \
        sarg = 1.0 / sqrt(arg); \
    end else begin \
        sarg = pow(arg, -MJ); \
    end \
    if (MJ != 1) begin \
        Qec = PB * Cz * (1.0 - arg * sarg) / (1.0 - MJ); \
    end else begin \
        Qec = -PB * Cz * `lln(arg); \
    end \
    arg = 1.0 - (vex - vec) / pb2; \
    if (MJ2 == 0.5) begin \
        sarg = 1.0 / sqrt(arg); \
    end else begin \
        sarg = pow(arg, -MJ2); \
    end \
    if (MJ2 != 1) begin \
        Qej = Qec + SJ * pb2 * Cz * (1.0 - arg * sarg) / (1.0 - MJ2); \
    end else begin \
        Qej = Qec - SJ * pb2 * Cz * `lln(arg); \
    end \
end \
```

2. For T1 >= 0.9

**BSIMCMG111.1.0**

```
T2 = pow(0.1, -MJ); \
T3 = 1.0 / (1.0 - MJ); \
T4 = T2 * (T1 - 1.0) * (5.0 * MJ * (T1 - 1.0) + (1.0 + MJ)); \
T5 = T3 * (1.0 - 0.05 * MJ * (1.0 + MJ) * T2); \
Qej = PB * Cz * (T4 + T5); \
```

Here, for MJ=1, the term T5 takes the form of 0/0. T5 for the limit MJ tends to 1 is a constant.

$$\lim MJ \to 1 \frac{1 - 0.05MJ(1+MJ)0.1^{-MJ}}{1-MJ} = 1.5 - \ln(0.1)$$

The code is then modified as

```
end else begin \
    T2 = pow(0.1, -MJ); \
    T3 = 1.0 / (1.0 - MJ); \
    T4 = T2 * (T1 - 1.0) * (5.0 * MJ * (T1 - 1.0) + (1.0 + MJ)); \
    if (MJ != 1) begin \
        T5 = T3 * (1.0 - 0.05 * MJ * (1.0 + MJ) * T2); \
    end else begin \
        T5 = 1.5 - ln(0.1); \
    end \
    Qej = PB * Cz * (T4 + T5); \
end \
```

Note:
This bug was addressed in BSIM-CMG111.2.0beta0_1 release. However, there was a term left that could take 1/0 form for MJ=1 in the junction_cap macro.

```
T2 = pow(0.1, -MJ); \
T3 = 1.0 / (1.0 - MJ); \
T4 = T2 * (T1 - 1.0) * (5.0 * MJ * (T1 - 1.0) + (1.0 + MJ)); \
if (MJ != 1) begin \
    T5 = T3 * (1.0 - 0.05 * MJ * (1.0 + MJ) * T2); \
end else begin \
    T5 = 1.5 - ln(0.1); \
end \
Qej = PB * Cz * (T4 + T5); \
```

Further, T2 is not required in Qej calculations when MJ=1 and sqrt function can be used in T2 in place of pow function for MJ=0.5.

In BSIM-CMG111.2.0beta0_2, the above issues have been corrected

```
T4 = T2 * (T1 - 1.0) * (5.0 * MJ * (T1 - 1.0) + (1.0 + MJ)); \
if (MJ != 1) begin \
    if (MJ == 0.5) begin \
        T2 = 1.0 / sqrt(0.1); \
    end else begin \
        T2 = pow(0.1, -MJ); \
    end \
    T3 = 1.0 / (1.0 - MJ); \
    T5 = T3 * (1.0 - 0.05 * MJ * (1.0 + MJ) * T2); \
end else begin \
    T5 = 1.5 - ln(0.1); \
end \
Qej = PB * Cz * (T4 + T5); \
```

Similar rearrangements are made at other places in the junction_cap macro.

5. **2021bug5(ADI): Incorrect derivative of ln_one_plus_exp function for x < -37**

ln_one_plus_exp function in BSIM-CMG 111.1.0beta0_1 is defined as

**BSIM-CMG 111.1.0beta0_1**

```
// ln(1 + exp(x)) function
analog function real ln_one_plus_exp;
    input x; real x;
    begin
        if (x > 37) begin
            ln_one_plus_exp = x;
        end else if (x < -37) begin
            ln_one_plus_exp = 0.0;
        end else begin
            ln_one_plus_exp = ln(1.0 + exp(x));
        end
    end
endfunction
```

Since 0 is assigned to the function for x < -37 (neglecting exp(x) in comparison to 1), its derivative is also 0 in this range. However, considering the actual derivative expression of ln(1 + exp(x))

$$\frac{dln(1+exp\,(x))}{dx} = \frac{exp\,(x)}{1+exp\,(x)} \neq 0\ for\ x < -37$$

In the revised BSIM-CMG 111.2.0beta0_2 code, ln_one_plus_exp function uses exp(x) for x < -37 which is also consistent with its Taylor series expansion.

**BSIM-CMG 111.1.0beta0_2**

```
// ln(1 + exp(x)) function
analog function real ln_one_plus_exp;
    input x; real x;
    begin
        if (x > 37) begin
            ln_one_plus_exp = x;
        end else if (x < -37) begin
            ln_one_plus_exp = exp(x);
        end else begin
            ln_one_plus_exp = ln(1.0 + exp(x));
        end
    end
endfunction
```

6. **2021bug6(UCB): Corrections in $T_{eff}$ implementation in cryogenic model**

A. The present $T_{eff}$ formulation for CRYOMOD=1 results in an unphysical behavior in two cases

i) For $T_{nom}$ < TLOW, $T_{eff}$ saturates to $T_{nom}$ below TLOW instead of saturating to TLOW (see red curve in Fig. 1). Also, a large difference between $T_{eff}$ and T can be seen for T > $T_{nom}$..

ii) When smoothing factor DTLOW is large, $T_{eff}$ can become significantly smaller than T for T > Tnom (see red curve in Fig. 2).

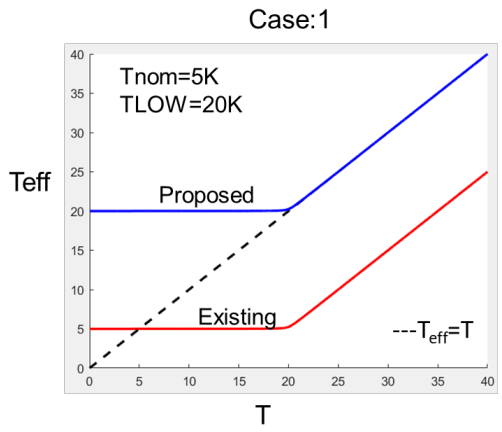



Fig. 1

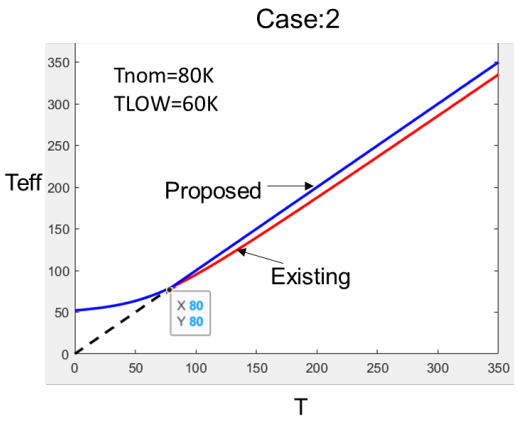



Fig. 2

**BSIM-CMG 111.2.0beta0_1 CRYOMOD=1**

$$T_{low0}(T) = \frac{T + TLOW + \sqrt{(T - TLOW)^2 + 0.25 \cdot DTLOW^2}}{2}$$

$$T_{low1}(T) = \frac{KLOW \cdot (TLOW1 - T) + \sqrt{[KLOW \cdot (TLOW1 - T)]^2 + 0.25 \cdot DTLOW1^2}}{2}$$

$$T_{eff}(T) = T_{low0}(T) + T_{low1}(T) - T_{low0}(T_{nom}) - T_{low1}(T_{nom}) + T_{nom}$$

**Code Implementation**

```
DevTemplow0 = `smoothminx(DevTemp, TLOW, DTLOW);
DevTemplow1 = `smoothminx((-KLOW1 * (DevTemp - TLOW1)), 0.0, DTLOW1);
if (CRYOMOD == 1) begin
    DevTempeff = DevTemplow0 + DevTemplow1 - `smoothminx(Tnom, TLOW, DTLOW) - `smoothminx((-KLOW1 * (
    Tnom - TLOW1)), 0.0, DTLOW1) + Tnom;
end else begin
    .
    .
```

In BSIM-CMG 111.2.0beta0_2, we correct this behavior by calculating $T_{eff}$ as

**BSIM-IMG 111.2.0beta0_2 CRYOMOD=1**

If $T_{nom} > TLOW$ then

$$T1 = T_{low0}(T) + T_{low1}(T) - T_{low0}(T_{nom}) - T_{low1}(T_{nom}) + T_{nom}$$

end else

$$T1 = T_{low0}(T) + T_{low1}(T) - T_{low0}(T_{nom}) - T_{low1}(T_{nom}) + TLOW$$

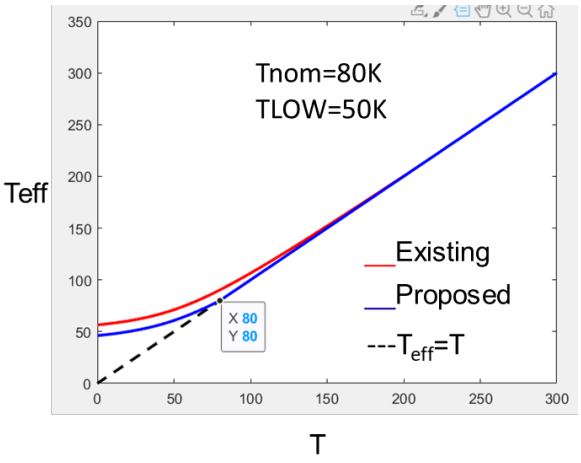
end

$$T_{eff} = \frac{T + T1 + \sqrt{(T - T1)^2 + 0.25 \cdot 0.04}}{2}$$

**Code Implementation**

$T_{eff}$ using the proposed implementation shows correct behavior as evident in Fig. 1 and 2 (blue curves).

B. $T_{eff}$ formulation for CRYOMOD=2 does not ensure $T_{eff}$=$T_{nom}$ at T=$T_{nom}$ for $T_{nom}$ < 210K (see red curve in Fig below). For a multi-temperature fitting, the fitting at T=$T_{nom}$ must not get affected while tuning temperature model parameters only.




```
Tlow = TLOW;
if (CRYOMOD != 0) begin
    // Effective temperature for core model charge calculation at low temperatures
    DevTemplow0 = `smoothminx(DevTemp, Tlow, DTLOW);
    DevTemplow1 = `smoothminx((-KLOW1 * (DevTemp - TLOW1)), 0.0, DTLOW1);
    if (CRYOMOD == 1) begin
        T1 = `smoothminx(Tnom, Tlow, DTLOW);
        T2 = `smoothminx((-KLOW1 * (Tnom - TLOW1)), 0.0, DTLOW1);
        if (Tnom > Tlow) begin
            T3 = DevTemplow0 + DevTemplow1 - T1 - T2 + Tnom;
        end else begin
            T3 = DevTemplow0 + DevTemplow1 - T1 - T2 + Tlow;
        end
        DevTempeff = `smoothminx(DevTemp, T3, 0.2);
    end else begin
        .
        .
        .
```

**BSIM-CMG 111.2.0beta0_1 CRYOMOD=2**

$$T_{low0}(T) = \frac{T + TLOW + \sqrt{(T - TLOW)^2 + 0.25 \cdot DTLOW^2}}{2}$$

$$T_{low1}(T) = \frac{KLOW \cdot (TLOW1 - T) + \sqrt{[KLOW \cdot (TLOW1 - T)]^2 + 0.25 \cdot DTLOW1^2}}{2}$$

$$T1 = T_{low0}(T) + T_{low1}(T) - T_{low0}(210) - T_{low1}(210) + 210$$

$$T_{eff} = \frac{T + T1 + \sqrt{(T - T1)^2 + 0.25 \cdot 0.01}}{2}$$

### Code Implementation

```
end else begin
    T1 = DevTemplow0 + DevTemplow1 - `smoothminx(210, TLOW, DTLOW) - `smoothminx((-KLOW1 * (210 - TLOW1
    )), 0.0, DTLOW1) + 210;
    DevTempeff = `smoothminx(DevTemp, T1, 0.1);
    DevTemp1 = `smoothmaxx2(DevTemp, 210, 0.1);
    delTemp1 = DevTemp1 - `smoothmaxx2(Tnom, 210, 0.1);
    delTRatio1 = (DevTemp1 - 210) / Tnom;
end
```

$T_{eff}$ is therefore modified in BSIM-CMG 111.2.0beta0_2 for $T_{nom} < 210$ K to correct for this behavior. The new formulation also takes care of the issues mentioned in A for CRYOMOD=1.

## BSIM-CMG 111.2.0beta0_2 CRYOMOD=2

If $T_{nom} > 210$ then

$$T1 = T_{low0}(T) + T_{low1}(T) - T_{low0}(210) - T_{low1}(210) + 210$$

$$T_{eff} = \frac{T + T1 + \sqrt{(T - T1)^2 + 0.25 \cdot 0.04}}{2}$$

end else

If $T_{nom} > TLOW$ then

$$T1 = T_{low0}(T) + T_{low1}(T) - T_{low0}(T_{nom}) - T_{low1}(T_{nom}) + T_{nom}$$

end else

$$T1 = T_{low0}(T) + T_{low1}(T) - T_{low0}(T_{nom}) - T_{low1}(T_{nom}) + TLOW$$

end

$$T2 = \frac{T + T1 + \sqrt{(T - T1)^2 + 0.25 \cdot 0.04}}{2}$$

$$w_h = 0.5 + 0.5 \cdot \tanh[0.5 \cdot (T - 210)]$$

$$w_l = 1 - w_h$$

$$T_{eff} = w_l \cdot T2 + w_h \cdot T$$

end

end

### Code Implementation

```
    end else begin
        if (Tlow > 210) begin
            $strobe("Warning: TLOW = %e is greater than 210 K. Set to 210 K", TLOW);
            Tlow = 210;
        end
        wh = 0.5 + 0.5 * tanh(0.5 * (DevTemp - 210));
        wl = 1.0 - wh;
        if (Tnom > 210) begin
            T1 = `smoothminx(210, Tlow, DTLOW);
            T2 = `smoothminx((-KLOW1 * (210 - TLOW1)), 0.0, DTLOW1);
            T3 = DevTemplow0 + DevTemplow1 - T1 - T2 + 210;
            DevTempeff = `smoothminx(DevTemp, T3, 0.2);
        end else begin
            T1 = `smoothminx(Tnom, Tlow, DTLOW);
            T2 = `smoothminx((-KLOW1 * (Tnom - TLOW1)), 0.0, DTLOW1);
            if (Tnom > Tlow) begin
                T3 = DevTemplow0 + DevTemplow1 - T1 - T2 + Tnom;
            end else begin
                T3 = DevTemplow0 + DevTemplow1 - T1 - T2 + Tlow;
            end
            T4 = `smoothminx(DevTemp, T3, 0.2);
            DevTempeff = wl * T4 + wh * DevTemp;
        end
        DevTemp1 = `smoothmaxx2(DevTemp, 210, 0.2);
        delTemp1 = DevTemp1 - `smoothmaxx2(Tnom, 210, 0.2);
        delTRatio1 = (DevTemp1 - 210) / Tnom;
    end
```

7. **2021bug7(UCB): Non-zero threshold shift at T=Tnom for Tnom < 210K in CRYOMOD=2**

   The CRYOMOD=2 expression for temperature threshold voltage shift in BSIM-CMG 111.2.0beta0_1 does not become zero at T = $T_{nom}$ where $T_{nom}$ < 210K.

   **BSIM-CMG 111.2.0beta0_1**

```
T4 = KT11 / (1 + `lexp(KT12 * (DevTemp - TVTH))) - KT11 / (1 + `lexp(KT12 * (210 - TVTH)));
if (KT11 * KT12 > 0) begin
    dvth_temp = (KT1_i + KT1L / Leff) * TRatio_m1 + `smoothminx(T4, 0, 1.0e-4);
end else begin
    dvth_temp = (KT1_i + KT1L / Leff) * TRatio_m1 + `smoothmaxx2(T4, 0, 1.0e-4);
end
```

   To ensure this, the threshold shift formulation in BSIM-CMG 111.2.0beta0_2 is modified as

   **BSIM-CMG 111.2.0beta0_2**

```
DevTemp1 = `smoothmaxx2(DevTemp, 210, 0.2);

T4 = `smoothmaxx2(Tnom, 210, 0.2);
dvth_temp = (KT1_i + KT1L / Leff) * TRatio_m1 + KT11 / (1 + `lexp(KT12 * (DevTemp1 - TVTH))) - KT11 / (1
 + `lexp(KT12 * (T4 - TVTH)));
```

8. **2021bug8(UCB): Impact ionization current missing in ISEFF/IDEFF OPs for BULKMOD=0**

For BULKMOD=0 (SOI), BSIM-CMG111.2.0beta0_1 implementation does not consider Iii (impact ionization current)

a) from drain to source in ISEFF for sigvds >0

b) from source to drain in IDEFF for sigvds < 0

**BSIM-CMG111.2.0beta0_1**

```
    // Total drain and source currents
    if (sigvds > 0.0) begin
        if (BULKMOD != 0) begin
            IDEFF = IDS + devsign * idsgen - devsign * (igd + igcd) + devsign * (Iii + igidl) - devsign * Ied;
            ISEFF = -IDS - devsign * idsgen - devsign * (igs + igcs) + devsign * (igisl) - devsign * Ies;
        end else begin
            IDEFF = IDS + devsign * idsgen - devsign * (igd + igcd + igbd) + devsign * (Iii + igidl - igisl);
            ISEFF = -IDS - devsign * idsgen - devsign * (igs + igcs + igbs) + devsign * (igisl - igidl);
        end
    end else begin
        if (BULKMOD != 0) begin
            IDEFF = -IDS - devsign * idsgen - devsign * (igs + igcs) + devsign * (igisl) - devsign * Ied;
            ISEFF = IDS + devsign * idsgen - devsign * (igd + igcd) + devsign * (Iii + igidl) - devsign * Ies;
        end else begin
            IDEFF = -IDS - devsign * idsgen - devsign * (igs + igcs + igbd) + devsign * (igisl - igidl);
            ISEFF = IDS + devsign * idsgen - devsign * (igd + igcd + igbs) + devsign * (Iii + igidl - igisl);
        end
    end
```

In BSIM-CMG 111.2.0beta0_2, we correct this for BULKMOD=0. In addition, BULKMOD!=0, the contribution of parasitic substrate GIDL/GISL is also accounted for GIDLMOD=2 and GEOMOD=2, 3 or 5.

**BSIM-CMG111.2.0beta0_2**

```
    if (sigvds > 0.0) begin
        if (BULKMOD != 0) begin
            if (GIDLMOD == 2 && (GEOMOD == 2 || GEOMOD == 3 || GEOMOD == 5)) begin
                IDEFF = IDS + devsign * idsgen - devsign * (igd + igcd) + devsign * (Iii + igidl + igidlb) - devsign * Ied;
                ISEFF = -IDS - devsign * idsgen - devsign * (igs + igcs) + devsign * (igisl + igislb) - devsign * Ies;
            end else begin
                IDEFF = IDS + devsign * idsgen - devsign * (igd + igcd) + devsign * (Iii + igidl) - devsign * Ied;
                ISEFF = -IDS - devsign * idsgen - devsign * (igs + igcs) + devsign * igisl - devsign * Ies;
            end
        end else begin
            IDEFF = IDS + devsign * idsgen - devsign * (igd + igcd + igbd) + devsign * (Iii + igidl - igisl);
            ISEFF = -IDS - devsign * idsgen - devsign * (igs + igcs + igbs) + devsign * (igisl - igidl - Iii);
        end
    end else begin
        if (BULKMOD != 0) begin
            if (GIDLMOD == 2 && (GEOMOD == 2 || GEOMOD == 3 || GEOMOD == 5)) begin
                IDEFF = -IDS - devsign * idsgen - devsign * (igs + igcs) + devsign * (igisl + igislb) - devsign * Ied;
                ISEFF = IDS + devsign * idsgen - devsign * (igd + igcd) + devsign * (Iii + igidl + igidlb) - devsign * Ies;
            end else begin
                IDEFF = -IDS - devsign * idsgen - devsign * (igs + igcs) + devsign * igisl - devsign * Ied;
                ISEFF = IDS + devsign * idsgen - devsign * (igd + igcd) + devsign * (Iii + igidl) - devsign * Ies;
            end
        end else begin
            IDEFF = -IDS - devsign * idsgen - devsign * (igs + igcs + igbd) + devsign * (igisl - igidl - Iii);
            ISEFF = IDS + devsign * idsgen - devsign * (igd + igcd + igbs) + devsign * (Iii + igidl - igisl);
        end
    end
```

9. **2021bug9(UCB): Vampyre error: Default value of model parameter 'LSP' depends on parameters 'L' and 'XL'**

LSP is a model parameter defined as
`MPRoz(LSP, 0.2 * (L + XL), "m", "Thickness of the gate sidewall spacer")

The default of LSP depends on L and XL that can be both model and instance parameters.

After workgroup discussion it has been decided to make the default LSP independent of L and XL in BSIM-CMG 111.2.0beta0_1 as
`MPRoz(LSP, 6.0e-9, "m", "Thickness of the gate sidewall spacer")

10. **2021bug10(ADI): Conditional assignment of OP variables**

    In BSIM-CMG 111.2.0beta0_1, the OP variables IGBS, IGBD, IGBINV and IGBACC are only conditionally assigned a value.

    **BSIM-CMG 111.2.0beta0_1**

```
if (BULKMOD == 0) begin
    IGBS = devsign * igbs;
    IGBD = devsign * igbd;
end else begin
    IGBINV = devsign * igbinv;
    IGBACC = devsign * igbacc;
end
```

    This has been corrected as

    **BSIM-CMG 111.2.0beta0_2**

```
if (BULKMOD == 0) begin
    IGBS = devsign * igbs;
    IGBD = devsign * igbd;
    IGBINV = 0.0;
    IGBACC = 0.0;
end else begin
    IGBINV = devsign * igbinv;
    IGBACC = devsign * igbacc;
    IGBS = 0.0;
    IGBD = 0.0;
end
```

11. **2021bug11(Silvaco): Correction in technical manual**

    In the technical manual NQSMOD was defined twice in Table 2

| Parameter | Function | Controlled Components |
|---|---|---|
| RGATEMOD | Parasitics | Rg: parasitic gate resistance |
| RDSMOD | Parasitics | Rd: parasitic drain resistance Rs: parasitic source resistance |
| NQSMOD | Non-quasi static | Rii: Intrinsic input resistance |
| IGCMOD | Gate leakage | Igs: gate-to-source tunneling current Igd: gate-to-drain tunneling current Igcs: gate-to-channel tunneling current at source side Igcd: gate-to-channel tunneling current at drain side |
| NQSMOD | Non-quasi static | Rii: Intrinsic input resistance |
| IGBMOD | Gate leakage | For BULKMOD = 1, Igbinv: gate-to-substrate tunneling current at inversion Igbacc: gate-to- |

This has been corrected in BSIM-CMG 111.2.0_beta02.

12. **2021bug12(UCB): S/D velocity saturation model corrections: a) correct for voltage polarity reversal and b) make delta factor a function of voltage**

In BSIM-CMG 111.2.0beta0_2, the voltage dependent resistance at the drain side is expressed as

$$R_{vs,d} = R_0 \left[ 1 + \left( \frac{\delta_{vs,rd}^{\frac{1}{MVSRSD}} \cdot devsign \cdot V(di1,\ di)}{I_{sat,rd} \cdot R_0} \right)^{MVSRSD} \right]^{\frac{1}{MVSRSD}}$$

with

$$\delta_{vs,rd} = \frac{I_{DS}^{4-MVSRSD}}{I_{DS}^{4-MVSRSD} + VSRDFACTOR \cdot I_{sat,rd}^{4-MVSRSD}}$$

a) In the resistance $R_{vs,d}$ expression above, V(di1,di) can become negative. For this reason, we smoothly clamped the (second) term in brackets to a small number for its negative values in beta0_2. However, according to physics, $R_{vs,d}$ should not depend on the sign of the voltage across it. For example, in nmos, if the source is at a higher voltage w.r.t the drain, the beta0_2 expression will clamp it to $R_0$ and underestimate $R_{vs,d}$, and thereby overestimate the current. Therefore, in beta1 release we now use |V(di1,di)| instead of V(di1,di).

b) $\delta_{vs,rd}$ factor in beta1 is replaced with a voltage dependent factor. This makes $R_{vs,d}$ completely a function of the voltage V(di1,di).

The updated $R_{vs,d}$ expression in BSIM-CMG 111.2.0 beta1is

$$R_{vs,d} = R_0 \left[ 1 + \left( \frac{\delta_{vs,rd} \cdot |V(di1,\ di)|}{I_{sat,rd} \cdot R_0} \right)^{MVSRSD} \right]^{\frac{1}{MVSRSD}}$$

$$\delta_{vs,rd} = \frac{|V(di1,\ di)|^{4-MVSRSD}}{|V(di1,\ di)|^{4-MVSRSD} + VSRDFACTOR \cdot (R_0 I_{sat,rd})^{4-MVSRSD}}$$

Similar, changes are done for the source side $R_{vs,s}$.

## Code Changes

### BSIM-CMG 111.2.0 beta0_2

```
// Velocity saturation in source/drain resistances
if (RDSMOD == 1) begin
    if (RDLCW > 0) begin
        rdstempvs = `hypsmooth((1.0 + PRTVSRSD * delTemp - 1.0e-6), 1.0e-3);
        `tempdep(VSATRSD_t, VSATRSD, -ATVSRSD)
        T0 = qis - PTWG1VSRSD;
        T0 = `smoothminx(T0, 0.1, 2.0);
        T1  = 10.0 * PSATXVSRSD * T0 / (10.0 * PSATXVSRSD + T0);
        vsatrsd_eff = VSATRSD_t * (1.0 + PTWGVSRSD * T1);
        vsatrsd_eff = `hypsmooth(vsatrsd_eff, 10.0);
        T2 = NFINtotal * Weff0 * `q * vsatrsd_eff;
        if (GAVSRD == 0) begin
            T3 = 1.0;
        end else begin
            T3 = `smoothminx((devsign * V(di1, di) - RDVDS), 0, 0.5);
            T3 = 1.0 + T3 * GAVSRD;
        end
        isat_rd = T2 * NVSRD * T3;
        delta_vsrd = pow(ids, 4 - MVSRSD) / (pow(ids, 4 - MVSRSD) + VSRDFACTOR * pow(isat_rd, 4 - MVSRSD));
        T4 = rdstempvs * RDLCW * WeffWRFactor;
        T5 = pow(delta_vsrd, 1.0 / MVSRSD) * devsign * V(di1, di) / (isat_rd * T4);
        T5 = `smoothminx(T5, 0.0, 1.0e-3);
        Rvs_d = T4 * pow((1.0 + pow(T5, MVSRSD)), 1 / MVSRSD);
    end
    if (RSLCW > 0) begin
        if (RDLCW == 0.0) begin
            rdstempvs = `hypsmooth((1.0 + PRTVSRSD * delTemp - 1.0e-6), 1.0e-3);
            `tempdep(VSATRSD_t, VSATRSD, -ATVSRSD)
            T0 = qis - PTWG1VSRSD;
            T0 = `smoothminx(T0, 0.1, 2.0);
            T1  = 10.0 * PSATXVSRSD * T0 / (10.0 * PSATXVSRSD + T0);
            vsatrsd_eff = VSATRSD_t * (1.0 + PTWGVSRSD * T1);
            vsatrsd_eff = `hypsmooth(vsatrsd_eff, 10.0);
            T2 = NFINtotal * Weff0 * `q * vsatrsd_eff;
        end
        isat_rs = T2 * NVSRS;
        delta_vsrs = pow(ids, 4 - MVSRSD) / (pow(ids, 4 - MVSRSD) + VSRSFACTOR * pow(isat_rs, 4 - MVSRSD));
        T4 = rdstempvs * RSLCW * WeffWRFactor;
        T5 = pow(delta_vsrs, 1.0 / MVSRSD) * devsign * V(si, si1) / (isat_rs * T4);
        T5S = V(si1, si);
        T5 = `smoothminx(T5, 0.0, 1.0e-3);
        Rvs_s = T4 * pow((1.0 + pow(T5, MVSRSD)), 1 / MVSRSD);
    end
end
```

**BSIM-CMG 111.2.0 beta1**

```
// Velocity saturation in source/drain resistances
if (RDSMOD == 1) begin
    if (RDLCW > 0) begin
        rdstempvs = `hypsmooth((1.0 + PRTVSRSD * delTemp - 1.0e-6), 1.0e-3);
        `tempdep(VSATRSD_t, VSATRSD, -ATVSRSD)
        T0 = qis - PTWG1VSRSD;
        T0 = `smoothminx(T0, 0.1, 2.0);
        T1  = 10.0 * PSATXVSRSD * T0 / (10.0 * PSATXVSRSD + T0);
        vsatrsd_eff = VSATRSD_t * (1.0 + PTWGVSRSD * T1);
        vsatrsd_eff = `hypsmooth(vsatrsd_eff, 10.0);
        T2 = NFINtotal * Weff0 * `q * vsatrsd_eff;
        T5 = abs(V(di1, di));
        if (GAVSRD == 0) begin
            T3 = 1.0;
        end else begin
            T3 = `smoothminx((T5 - RDVDS), 0, 0.5);
            T3 = 1.0 + T3 * GAVSRD;
        end
        isat_rd = T2 * NVSRD * T3;
        T4 = rdstempvs * RDLCW * WeffWRFactor;
        vsat_rd = isat_rd * T4;
        delta_vsrd = pow(T5, 4 - MVSRSD) / (pow(T5, 4 - MVSRSD) + VSRDFACTOR * pow(vsat_rd, 4 - MVSRSD));
        T6 = pow(delta_vsrd, 1.0 / MVSRSD) * T5 / vsat_rd;
        Rvs_d = T4 * pow((1.0 + pow(T6, MVSRSD)), 1 / MVSRSD);
    end
    if (RSLCW > 0) begin
        if (RDLCW == 0.0) begin
            rdstempvs = `hypsmooth((1.0 + PRTVSRSD * delTemp - 1.0e-6), 1.0e-3);
            `tempdep(VSATRSD_t, VSATRSD, -ATVSRSD)
            T0 = qis - PTWG1VSRSD;
            T0 = `smoothminx(T0, 0.1, 2.0);
            T1  = 10.0 * PSATXVSRSD * T0 / (10.0 * PSATXVSRSD + T0);
            vsatrsd_eff = VSATRSD_t * (1.0 + PTWGVSRSD * T1);
            vsatrsd_eff = `hypsmooth(vsatrsd_eff, 10.0);
            T2 = NFINtotal * Weff0 * `q * vsatrsd_eff;
        end
        isat_rs = T2 * NVSRS;
        T4 = rdstempvs * RSLCW * WeffWRFactor;
        vsat_rs = isat_rs * T4;
        T5 = abs(V(si, si1));
        delta_vsrs = pow(T5, 4 - MVSRSD) / (pow(T5, 4 - MVSRSD) + VSRSFACTOR * pow(vsat_rs, 4 - MVSRSD));
        T6 = pow(delta_vsrs, 1.0 / MVSRSD) * T5 / vsat_rs;
        Rvs_s = T4 * pow((1.0 + pow(T6, MVSRSD)), 1 / MVSRSD);
    end
```

13. **2021bug13(UCB): Units and typos correction in code/manual**

a) Units of SPRT and the corresponding binning parameters

**BSIM-CMG 111.2.0beta0_2:**

```
`MPRcz(SPRT, 1.0e-2, "/K", "CRYOMOD != 0 parameter for corner temperature smoothing in dual-slope temperature model of series resistance")
`MPRnb(LSPRT, 0.0, "m/K", "L-term of SPRT")
`MPRnb(NSPRT, 0.0, "/K", "N-term of SPRT")
`MPRnb(PSPRT, 0.0, "m/K", "P-term of SPRT")
`MPRnb(WSPRT, 0.0, "m/K", "W-term of SPRT")
`MPRnb(P2SPRT, 0.0, "(m^2)/K", "WL-term of SPRT")
```

SPRT appears in terms like `smoothminx(rdstemp0, rdstemp1, SPRT_i) and should be a unitless parameter (since rdstemp0 and rdstemp1 are unitless).

**Corrected units in BSIM-CMG 111.2.0beta1:**

```
`MPRcz(SPRT, 1.0e-2, "", "CRYOMOD != 0 parameter for corner temperature smoothing in dual-slope temperature model of series resistance")
`MPRnb(LSPRT, 0.0, "m", "L-term of SPRT")
`MPRnb(NSPRT, 0.0, "", "N-term of SPRT")
`MPRnb(PSPRT, 0.0, "m", "P-term of SPRT")
`MPRnb(WSPRT, 0.0, "m", "W-term of SPRT")
`MPRnb(P2SPRT, 0.0, "m^2", "WL-term of SPRT")
```

b) Units of binning parameters of TR0

**BSIM-CMG 111.2.0beta0_2:**

```
`MPRnb(LTR0, 0.0, "m/K", "L-term of TR0")
`MPRnb(NTR0, 0.0, "/K", "N-term of TR0")
`MPRnb(PTR0, 0.0, "m/K", "P-term of TR0")
`MPRnb(WTR0, 0.0, "m/K", "W-term of TR0")
`MPRnb(P2TR0, 0.0, "(m^2)/K", "WL-term of TR0")
```

**Corrected units in BSIM-CMG 111.2.0beta1:**

```
`MPRnb(LTR0, 0.0, "m*K", "L-term of TR0")
`MPRnb(NTR0, 0.0, "K", "N-term of TR0")
`MPRnb(PTR0, 0.0, "m*K", "P-term of TR0")
`MPRnb(WTR0, 0.0, "m*K", "W-term of TR0")
`MPRnb(P2TR0, 0.0, "(m^2)*K", "WL-term of TR0")
```

c) Typo in eq 3.201 on page 24 of manual

**BSIM-CMG 111.2.0beta0_2**

$$T_{low1}(T) = \frac{KLOW \cdot (TLOW1 - T) + \sqrt{[KLOW \cdot (TLOW1 - T)]^2 + 0.25 \cdot DTLOW1^2}}{2}$$

KLOW in the above expression should be KLOW1

**BSIM-CMG 111.2.0beta1**

$$T_{low1}(T) = \frac{KLOW1 \cdot (TLOW1 - T) + \sqrt{[KLOW1 \cdot (TLOW1 - T)]^2 + 0.25 \cdot DTLOW1^2}}{2}$$

14. **2021bug14(ADI): Remove unused macros**

In BSIM-CMG 111.2.0beta1, the following macros have been defined but are never used in the code.

NOTICE in file bsimcmg.va, line 36: Macro 'MEXPQM' is never used
NOTICE in file bsimcmg.va, line 36: Macro 'DELTA_ASYMM' is never used

They have been removed in BSIM-CMG 111.2.0beta2.

15. **2021bug15(ADI): Limit parameter MVSRSD's range**

The drain/source velocity saturation model in BSIM-CMG 111.2.0 beta1is given by

$$R_{vs,d} = R_0 \left[ 1 + \left( \frac{\delta_{vs,rd} \cdot |V(di1,\ di)|}{I_{sat,rd} \cdot R_0} \right)^{MVSRSD} \right]^{\frac{1}{MVSRSD}}$$

with

$$\delta_{vs,rd} = \frac{|V(di1,\ di)|^{4-MVSRSD}}{|V(di1,\ di)|^{4-MVSRSD} + VSRDFACTOR \cdot (R_0 I_{sat,rd})^{4-MVSRSD}}$$

In $\delta_{vs,rd}$, MVSRSD >= 4 can result in a divide by zero situation when $V(di1,\ di) = 0$. We have therefore limited its range to (0, 4) in BSIM-CMG 111.2.0beta2.

```
`MPRoo(MVSRSD, 1.0, "", 0, 4, "Non-linear resistance parameter")
```

16. **2022bug1(Cadence): Allow minr = 0**

In **BSIM-CMG 111.2.0beta1**, minr=0 was not allowed due to the use of `MPRoz macro

```
// Minimum resistance value
`MPRoz(minr, $simparam("minr", 1m), "Ohm", "minr is the value below which the simulator expects elimination of
source/drain resitance and it will improve simulation efficiency without significantly altering the results")
```

It has been a corrected in **BSIM-CMG 111.2.0beta2** replacing `MPRoz by `MPRcz

```
`MPRcz(minr, $simparam("minr", 1m), "Ohm", "minr is the value below which the simulator expects elimination of
source/drain resitance and it will improve simulation efficiency without significantly altering the results")
```

17. **2022bug2(Cadence, ADI): Reverse temperature sweep's order in QA**

The temperature sweep in **BSIM-CMG 111.2.0beta1 QA** is performed from -273 C to 125 C as

```
tempSweep ..... -273, 125, 1
```

However, Spectre simulates for a dummy point first and starts from -274 C which is not allowed.

In **BSIM-CMG 111.2.0beta2 QA,** we have reversed the order of the temperature sweep to avoid the aforementioned error as

```
tempSweep ..... 125, -273, -1
```

18. **2022bug3(Silvaco, ADI): No key for SmartSpice in QA perl script**

The CMC QA suit 3.0.0 perl script does not have key for SmartSpice simulator. The following statement is added in the qaSpec file of BSIM-CMG 111.2.0beta2 as a workaround

```
nTypeSelectionArguments   bsimcmg_va
pins                      d g s b
verilogaFile              ../code/bsimcmg.va
keyLetter                 x
checkPolarity             no
symmetricPins             d s
scaleParameters           m
```

Note: The latest qa release 3.1.0 takes care of the above issue. Therefore, BSIM-CMG_111.2.0 standard qaSpec file does not need keyletter x.

19. **2022bug4(UCB): GIDLMOD=2 parameters missing in parameter table of manual**

In **BSIMCMG-111.2.0beta1**, GIDLMOD=2 parameters, AGIDLB, BGIDLB, CGIDLB, EGIDLB, PGIDLB, AGISLB, BGISLB, CGISLB, EGISLB and PGISLB were not listed in the parameters table (6.4) in the manual.

They have been added in **BSIMCMG-111.2.0beta2** as

| | | | | | |
|---|---|---|---|---|---|
| AGIDLB$^{(b)}$ | $\Omega^{-1}$ | 6.055e-12 | - | - | Pre-exponetial coeff. for GIDL (GIDLMOD=2) |
| BGIDLB$^{(b)}$ | $Vm^{-1}$ | 0.3e9 | - | - | Exponential coeff. for GIDL (GIDLMOD=2) |
| CGIDLB$^{(b)}$ | $V^3$ | 0.2 | - | - | Parameter for body bias effect of GIDL (GIDLMOD=2) |
| EGIDLB$^{(b)}$ | $V$ | 0.2 | - | - | Band bending parameter for GIDL (GIDLMOD=2) |
| PGIDLB$^{(b)}$ | - | 1.0 | - | - | Exponent of electric field for GIDL (GIDLMOD=2) |
| AGISLB$^{(b)}$ | $\Omega^{-1}$ | AIGDLB | - | - | Pre-exponetial coeff for GISL (GIDLMOD=2) |
| BGISLB$^{(b)}$ | $Vm^{-1}$ | BGIDLB | - | - | Exponential coeff. for GISL (GIDLMOD=2) |
| CGISLB$^{(b)}$ | $V^3$ | CGIDLB | - | - | Parameter for body bias effect of GIS (GIDLMOD=2)L |
| EGISLB$^{(b)}$ | $V$ | EGIDLB | - | - | Band bending parameter for GISL (GIDLMOD=2) |
| PGISLB$^{(b)}$ | - | PGIDLB | - | - | Exponent of electric field for GISL (GIDLMOD=2) |

20. **2022bug5(Keysight): Typo in GIDLMOD=2**

BSIM-CMG 111.2.0beta2 has a typo in the following code snippet

```
if (sigvds > 0.0) begin
    igisl = T6;
    igislb = T7;
end else begin
    igidl = T6;
    igislb = T7;
end
```

igislb in the else condition should be igidlb.

Corrected code in BSIM-CMG 111.2.0beta3

```
if (sigvds > 0.0) begin
    igisl = T6;
    igislb = T7;
end else begin
    igidl = T6;
    igidlb = T7;
end
```

21. **2022bug6(Keysight, ADI, UCB): Divide by zero case in CRYOMOD=2**

In the following condition for UDS_t in CRYOMOD=2 of BSIMCMG111.2.0beta2, UDS_t will blow up if UDS1_i*(Tnom - 210)/Tnom is zero (or very close to zero). Similarly for UDD_t.

```
if (Tnom == 210) begin
    UDS_t = UDS_i * (`lexp(UDS1_i * delTRatio1) - 1);
    UDD_t = UDD_i * (`lexp(UDD1_i * delTRatio1) - 1);
end else begin
    UDS_t = UDS_i * (`lexp(UDS1_i * (DevTemp1 - 210) / Tnom) - 1) / abs(`lexp(UDS1_i * (Tnom - 210) / Tnom) - 1);
    UDD_t = UDD_i * (`lexp(UDD1_i * (DevTemp1 - 210) / Tnom) - 1) / abs(`lexp(UDD1_i * (Tnom - 210) / Tnom) - 1);
end
```

We provide protection for the above case in BSIMCMG111.2.0beta3 as

```
if (abs(UDS1_i * (Tnom - 210) / Tnom) < 1e-6) begin
    UDS_t = UDS_i * (`lexp(UDS1_i * delTRatio1) - 1);
end else begin
    UDS_t = UDS_i * (`lexp(UDS1_i * delTRatio1)  - 1) / abs(`lexp(UDS1_i * (Tnom - 210) / Tnom) - 1);
end
if (abs(UDD1_i * (Tnom - 210) / Tnom) < 1e-6) begin
    UDD_t = UDD_i * (`lexp(UDD1_i * delTRatio1) - 1);
end else begin
    UDD_t = UDD_i * (`lexp(UDD1_i * delTRatio1)  - 1) / abs(`lexp(UDD1_i * (Tnom - 210) / Tnom) - 1);
end
```

22. **2022bug7(Cadence): Code implementation issue in junction_cap macro**

The following portion of the code of junction_cap macro in **BSIM-CMG 111.2.0** has an implementation issue.

For T1=vex/PB > =0.9

```
T4 = T2 * (T1 - 1.0) * (5.0 * MJ * (T1 - 1.0) + (1.0 + MJ)); \
if (MJ != 1) begin \
    if (MJ == 0.5) begin \
        T2 = 1.0 / sqrt(0.1); \
    end else begin \
        T2 = pow(0.1, -MJ); \
    end \
    T3 = 1.0 / (1.0 - MJ); \
    T5 = T3 * (1.0 - 0.05 * MJ * (1.0 + MJ) * T2); \
end else begin \
    T5 = 1.5 - ln(0.1); \
end \
Qej = PB * Cz * (T4 + T5); \
```

Here, T4 requires T2 but appears before T2 calculation. Further, T2 is missing for MJ=1.

The following shows the correct implementation in **BSIM-CMG 111.2.1**

```
if (MJ != 1) begin \
    if (MJ == 0.5) begin \
        T2 = 1.0 / sqrt(0.1); \
    end else begin \
        T2 = pow(0.1, -MJ); \
    end \
    T3 = 1.0 / (1.0 - MJ); \
    T5 = T3 * (1.0 - 0.05 * MJ * (1.0 + MJ) * T2); \
end else begin \
    T2 = 10.0; \
    T5 = 1.5 - ln(0.1); \
end \
T4 = T2 * (T1 - 1.0) * (5.0 * MJ * (T1 - 1.0) + (1.0 + MJ)); \
Qej = PB * Cz * (T4 + T5); \
```

23. **2022bug8(ADI): Use local variables in macros**

The junction capacitance macro in BSIM-CMG 111.2.1 uses global variables T1, T2, T3, etc. In BSIM-CMG 111.3.0beta0_1 these have now been defined locally in the macro itself as

**BSIM-CMG 111.3.0beta0_1**

```
`define junction_cap(block_name, vex, vec, pb2, Cz, PB, SJ, MJ, MJ2, Qej) \
begin: block_name \
    real T1, T2, T3, T4, T5; \
    real arg, sarg, Qec; \
    \
    if (Cz > 0.0) begin \
        T1 = vex / PB; \
        if (T1 < 0.9) begin \
            if (SJ > 0.0) begin \
                if (vex > vec) begin \
                    arg = 1.0 - T1; \
```

24. **2022bug9(Cadence, ADI): Bound DTLOW and DTLOW1 to be > 0**

DTLOW and DTLOW1 are smoothing parameters used in the cryogenic model effective temperature formulation as

$$T_{low0}(T) = \frac{T + TLOW + \sqrt{(T - TLOW)^2 + 0.25 \cdot DTLOW^2}}{2}$$

$$T_{low1}(T) = \frac{KLOW1 \cdot (TLOW1 - T) + \sqrt{[KLOW1 \cdot (TLOW1 - T)]^2 + 0.25 \cdot DTLOW1^2}}{2}$$

**BSIM-CMG 111.2.1** defines DTLOW and DTLOW1 as real numbers

```
`MPRnb(DTLOW, 1.0, "K", "CRYOMOD != 0 smoothing parameter for TLOW")
`MPRnb(DTLOW1, 0.0, "K", "CRYOMOD != 0 smoothing parameter for TLOW1")
```

DTLOW and DTLOW1 are also allowed to be zero, which can result in temperature derivative discontinuity.

In **BSIM-CMG 111.3.0beta0_1**, these parameters have been restricted to be positive numbers, and the default value of DTLOW1 has also been changed to 1.0e-3 as

```
`MPRoz(DTLOW, 1.0, "K", "CRYOMOD != 0 smoothing parameter for TLOW")
`MPRoz(DTLOW1, 1.0e-3, "K", "CRYOMOD != 0 smoothing parameter for TLOW1")
```

25. **2022bug10(UCB): Typos in manual**

Min/max values of a few parameters were incorrectly specified in the **BSIM-CMG 111.2.1** manual

| Name | Unit | Default | Min | Max | Description |
|---|---|---|---|---|---|
| TLOW | $K$ | 50.0 | - | 0 | Transition temperature of SS at low temperatures (CRYOMOD $\neq$ 0) |
| DTLOW | $K$ | 1.0 | - | 0 | Smoothing parameter for TLOW (CRYOMOD $\neq$ 0) |
| TLOW1 | $K$ | 0.0 | - | 0 | Transition temperature of SS at low temperatures (CRYOMOD $\neq$ 0) |
| DTLOW1 | $K$ | 0.0 | - | 0 | Smoothing parameter for TLOW1 (CRYOMOD $\neq$ 0) |
| KLOW1 | - | 0.0 | - | 0 | Slope magnitude of effective low temperature below TLOW1 (CRYOMOD $\neq$ 0) |

These typos have been corrected in **BSIM-CMG 111 .3.0beta0_1** manual.

| | | | | | |
|---|---|---|---|---|---|
| TLOW | $K$ | 50.0 | 0.0 | - | Transition temperature of SS at low temperatures (CRYOMOD $\neq$ 0) |
| DTLOW | $K$ | 1.0 | $> 0$ | - | Smoothing parameter for TLOW (CRYOMOD $\neq$ 0) |
| TLOW1 | $K$ | 0.0 | 0.0 | - | Transition temperature of SS at low temperatures (CRYOMOD $\neq$ 0) |
| DTLOW1 | $K$ | 1.0e-3 | $> 0$ | - | Smoothing parameter for TLOW1 (CRYOMOD $\neq$ 0) |
| KLOW1 | - | 0.0 | 0.0 | - | Slope magnitude of effective low temperature below TLOW1 (CRYOMOD $\neq$ 0) |

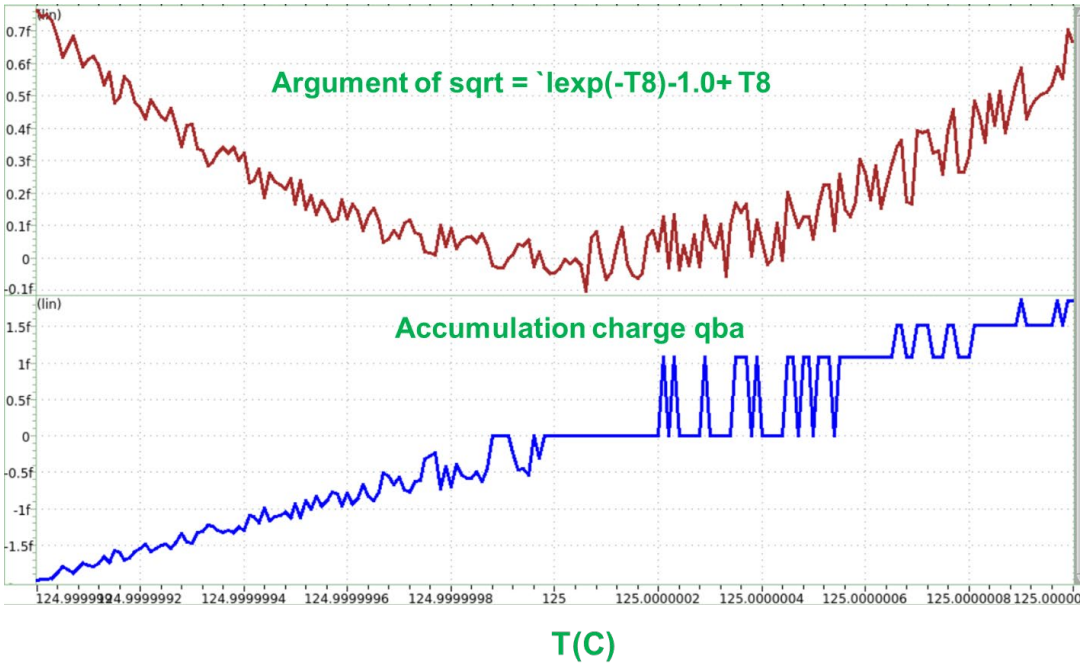
26. **2022bug12(UCB, ADI, Cadence): Inaccuracy in exp(x) – 1 calculation in Verilog-A for very small x**

Cadence provided us with a test case where they observed a floating point error in accumulation charge at T very close to 125 °C. After investigation, the problem was found in the calculation of `lexp(x) – 1 terms (highlighted) when x becomes close to zero, which was happening in this test case.

```
T9 = K1_t / (2.0 * nVtm) * sqrt(Vtm);
T0 = T9 / 2.0;
T2 = (vge - (deltaPhi - Eg - Vtm * `lln(NBODY_i / Nc) + DELVFBACC)) / Vtm;
if ((T2 * Vtm) > phib + T9 * sqrt(phib * Vtm)) begin
    T1 = sqrt(T2 - 1.0 + T0 * T0) - T0;
    T10 = 1.0 + T1 * T1;
    T6 = `lexp(-T10) - 1.0;
end else begin
    T3 = T2 * 0.5 - 3.0 * (1.0 + T9 / sqrt(2.0));
    T10 = T3 + sqrt(T3 * T3 + 6.0 * T2);
    if (T2 < 0.0) begin
        T4 = (T2 - T10) / T9;
        T6 = -T10 + T4 * T4;
        T10 = -`lln(1.0 - T10 + T4 * T4);
    end else begin
        T11 = `lexp(-T10);
        T4 = sqrt(T2 - 1.0 + T11 + T0 * T0) - T0;
        T10 = 1.0 - T11 + T4 * T4;
        T6 = `lexp(-T10) - 1.0;
    end
end
T7 = sqrt(T6 + T10);
if (T10 > 1.0e-15) begin
    e0 = -(T2 - T10) + T9 * T7;
    e1 = 1.0 - T9 * 0.5 * T6 / T7;
    T8 = T10 - (e0 / e1);
    T11 = `lexp(-T8) - 1.0;
    T12 = sqrt(T11 + T8);
    qba = -T9 * T12 * Vtm;
end else begin
    if (T10 < -1.0e-15) begin
        e0 = -(T2 - T10) - T9 * T7;
        e1 = 1.0 + T9 * 0.5 * T6 / T7;
        T8 = T10 - e0 / e1;
        T12a = `lexp(-T8) + T8 - 1.0;
        if (T12a <= 0) begin
            T12 = 0.0;
        end else begin
            T12 = T9 * sqrt(T12a);
        end

    end else begin
        T8 = 0.0;
        T12 = 0.0;
    end
    qba = T12 * Vtm;
```

It was found that the value of `lexp(x) – 1 using Verilog-A exp(x) function was inaccurate for very small x and resulted in a numerical noise, e.g., as shown in the figure below the calculation of the term `lexp(-T8) – 1.0 + T8. Note that mathematically, this term must >=0 for any T8 value, however, due to numerical noise it sometimes becomes negative and when used in the sqrt function, results in a floating-point error.

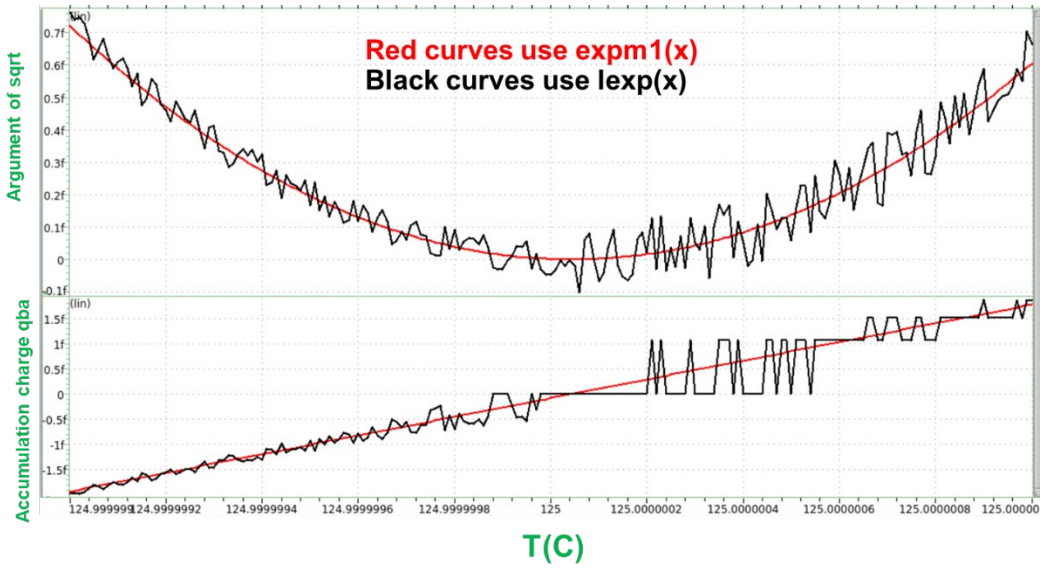



In **BSIM-CMG 112 .0.0beta0_1** we have replaced the above highlighted `lexp(x) - 1 terms with a new function called expm1(x)

```
// exp(x) - 1 function
analog function real expm1;
    input x; real x;
    begin
        if (abs(x) < 1e-7) begin
            expm1 = x + 0.5 * x * x;
        end else begin
            expm1 = `lexp(x) - 1.0;
        end
    end
endfunction
```

This function uses a Taylor expansion up to the second order and ensures that the value of exp(x) -1 + x remains always positive (higher order Taylor terms are numerically lost when x is small). Also, note that for |x|>=1e-7, expm1 becomes equal to `lexp(x) – 1. With expm1 used in accumulation charge (qba) calculation, we

do not see any numerical noise or floating-point errors since the argument of the sqrt function always remains positive as shown in the figure below.

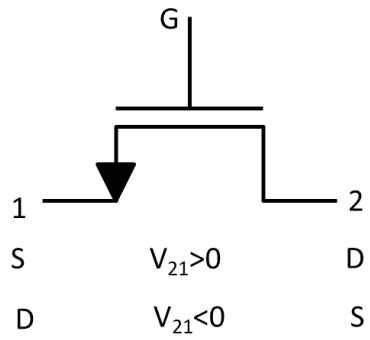



27. **2022bug14(ADI): Change terminal "b" in QA with "e" for consistency with the code**

    The body terminal in the BSIM-CMG code is denoted by "e". For consistency, the body terminal named "b" in QA files (qaspec & reference files) is replaced with the letter "e".

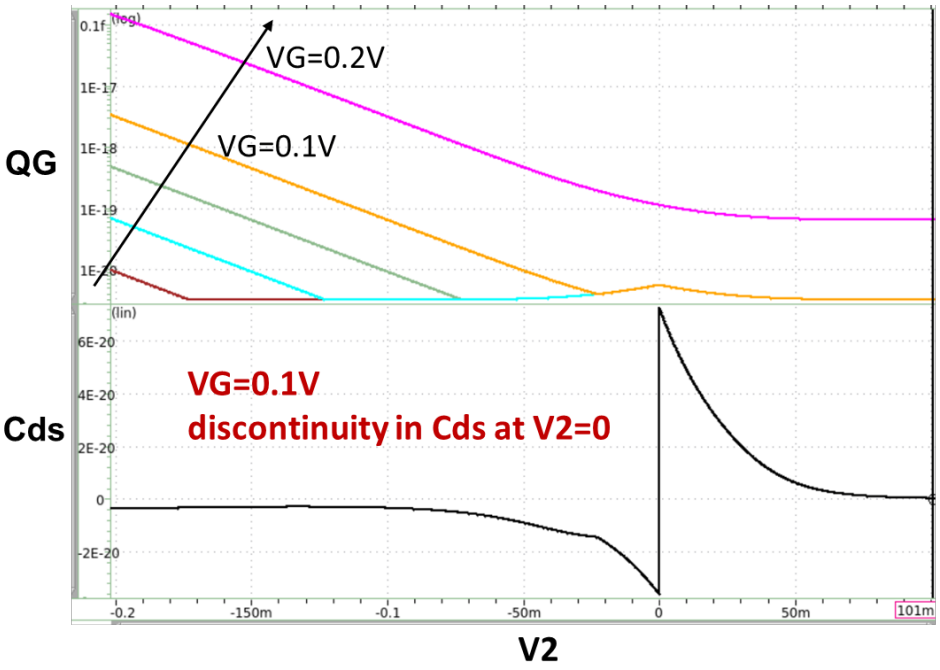
28. **2023bug1(Cadence): Discontinuity in Cds at Vds =0 for small Vgs (accumulation region)**

    A discontinuity in Cds capacitance is observed when Vgs is small and Vds is swept from a positive value to a negative value as shown in the case below.



VG=V1=VB=0, V2 is swept across 0

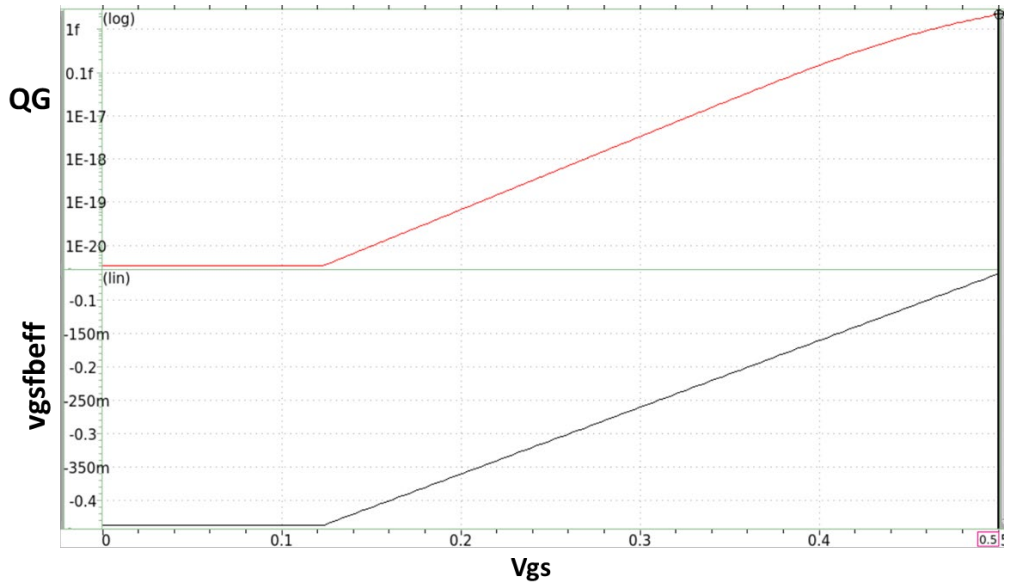
The following figure shows the gate terminal charge and Cds as a function of V2 for different Vg values.



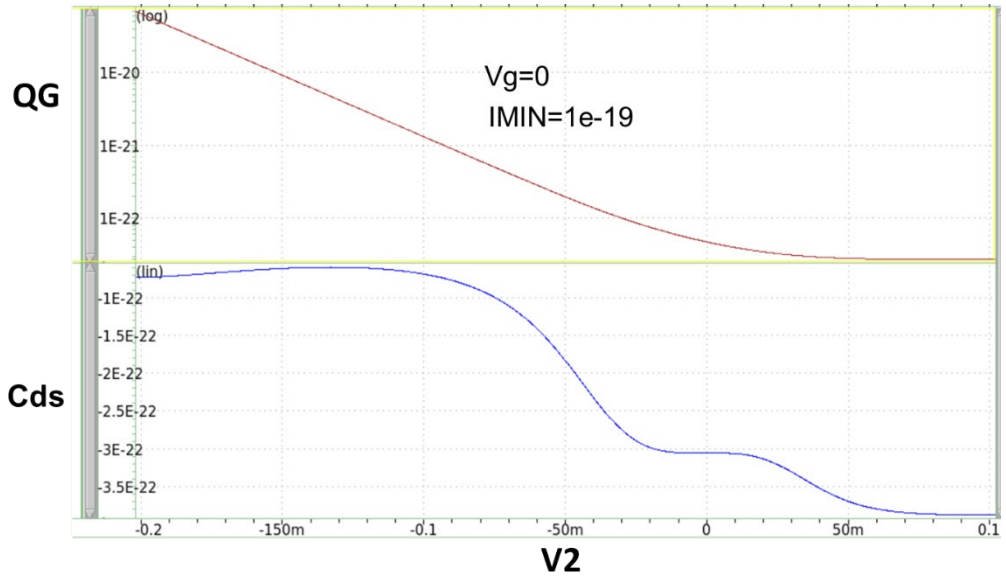

As shown for low Vg, QG exhibits a change in slope V2=0, and Cds is discontinuous. For $V_{21}$>0, terminal 2 acts as the drain. As $V_{21}$=Vds increases, the drain draws up electrons from the channel and QG decreases as expected from physics. However, for $V_{21}$ < 0, the source and drain are swapped. Now Vds still increases but Vgs also increases simultaneously. Physically, Vgs should take over and Qg should increase for $V_2$<0. For Vg=0.2V this is the case, but for Vg=0.1V (or lower), Qg starts to decrease first and then increases. This reduction of Qg causes a discontinuity in capacitance at Vds=0. It indicates an initial Qg desenstivity w.r.t Vgs for $V_2$<0. We find this is due to the clamping used for calculating effective Vgs in accumulation region in BSIM-CMG as shown in the code snippet below.

```
beta0 = u0_a * cox * Weff0 / Leff;
T0 = -(dvch_qm + nVtm * `lln(2.0 * cox * IMIN / (beta0 * nVtm * `q * Nc * TFIN)));
T1 = vgsfb + T0 + DELVTRAND;
vgsfbeff = `hypsmooth(T1, 1.0e-4) - T0;
```

Vgs is clamped when device reaches accumulation region, and the current becomes very small. The lower limit of current can be set by the parameter IMIN. For small Vgs, QG is also clamped and does not change w.r.t Vgs as shown in the figure below.




This desensitivity of QG results in the decrease in QG observed earlier for $V_{21} < 0$ since the effect of Vds takes over. To avoid this issue, the current limit IMIN can be reduced from the default value of 1e-15A/m^2. As shown below, for IMIN=1e-19, the QG behavior is physical and discontinuity in capacitance for low Vgs goes away.




We have added a precautionary note in the technical manual of **BSIM-CMG 112.0.0beta0_1** as

"Vgsfbeff limiting for a small Vgs and Vds being swept from positive to negative values can sometimes result in discontinuities in terminal capacitances. Such issues if

arise can be resolved by lowering the value of IMIN parameter."

29. **2023bug2(ADI): Use bias-independent if-else conditions for warnings**

Several instances of bias-dependent if-else conditions in BSIM-CMG 112.0.0beta0_1 were using $strobe conditions. These can be categorized into two types.

Type 1:

**BSIM-CMG 112.0.0beta0_1**

```
if (VSAT_t < 1000) begin
    $strobe("Warning: VSAT(%g) = %g is less than 1K, setting it to 1K.", DevTemp, VSAT_t);
    VSAT_t = 1000;
end
```

VSAT_t is device temperature (DevTemp) dependent i.e., it is also bias dependent. In total, there were 11 VAMPyRE Warnings that applied to VSAT_t, VSAT1_t, VSATCV_t, VSATR_t, and VSAT1R_t in CRYOMOD=0, 1, and 2, and TEMPMOD=0 and 1.

Since the if-else condition is bias-dependent here, the simulator must save the message during evaluation and then print the saved messages when the iteration converges. To avoid this, we have removed the $strobe statement message as

**BSIM-CMG 112.0.0beta0_2**

```
if (VSAT_t < 1000) begin
    VSAT_t = 1000;
end
```

Type 2:

There were also if-else statements with bias-dependent charge density-based conditions.

**BSIM-CMG 112.0.0beta0_1**

```
    if (qis < 0.0) begin
        $strobe("Warning: Negative source-side inversion carrier density. Vgs = %g Vds = %g Vbs = %g qis
= %g", V(g, s), V(d, s), V(e, s), qis);
    end
    if (qid < 0.0) begin
        $strobe("Warning: Negative drain-side inversion carrier density. Vgs = %g Vds = %g Vbs = %g qid =
%g", V(g, s), V(d, s), V(e, s), qid);
    end
    if (qis_cv < 0.0) begin
        $strobe("Warning: Negative source-side inversion carrier density for C-V. Vgs = %g Vds = %g
Vbs = %g qis_cv = %g", V(g, s), V(d, s), V(e, s), qis_cv);
    end
    if (qid_cv < 0.0) begin
        $strobe("Warning: Negative drain-side inversion carrier density for C-V. Vgs = %g Vds = %g
Vbs = %g qid_cv = %g", V(g, s), V(d, s), V(e, s), qid_cv);
    end
end
```

**BSIM-CMG 112.0.0beta0_2**

The above warnings have now been removed from the code since they don't affect the code calculations in any way.

30. **2023bug3(Cadence, UCB): Discontinuity in Cds at Vds=0 and a large negative gate voltage**

For test cases involving $V_d$ sweeping from positive to negative values (or vice versa) and large negative $V_{gs}$ (Fig. 1), it was found that the Cds and Csd capacitances are discontinuous at Vds=0 as shown in Fig. 2.

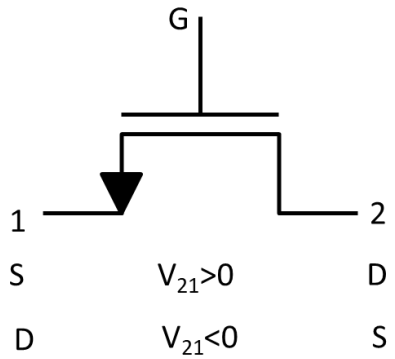
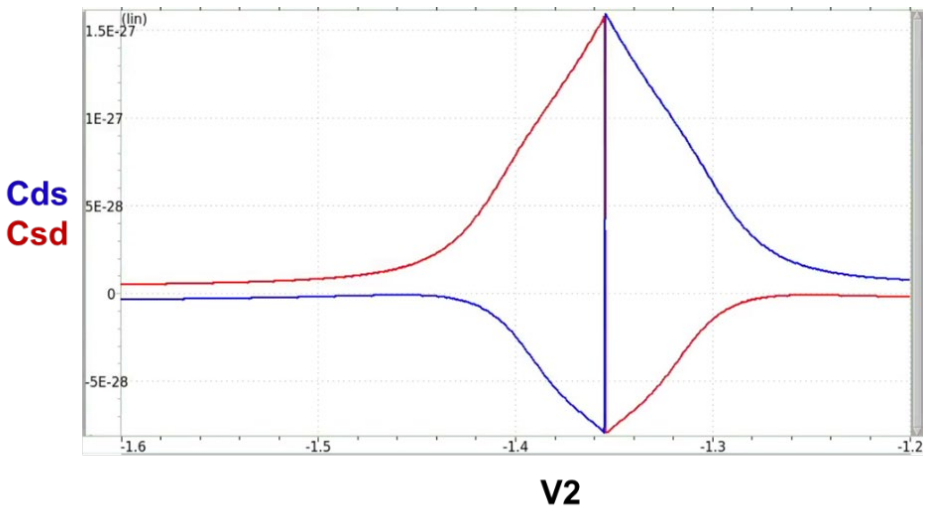



Fig. 1 Test setup for capacitance discontinuity check. V1=-1.35424 V, Vg=-5.5566 V, Vb = 0 V, and V2 is swept from a positive to a negative value. Temperature = 125 C.



The fundamental reason for the discontinuity was found to be the Vgs clamping to a minimum level in the large negative Vgs region. The minimum Vgs level can be controlled using the parameter IMIN. It was suggested in the BSIM-CMG 112.0.0beta0_1 manual that for such cases, the IMIN be reduced from its default value of 1e-15 to a smaller number to reduce the minimum Vgs and thereby get rid of the capacitance discontinuity. However, the Vgs clamping formulation shown below still put a limit to the minimum Vgs value achieved.

**BSIM-CMG 112.0.0beta0_1**

```
T0 = -(dvch_qm + nVtm * `lln(2.0 * cox * IMIN / (beta0 * nVtm * `q * Nc * TFIN)));
T1 = vgsfb + T0 + DELVTRAND;
vgsfbeff = `hypsmooth(T1, 1.0e-4) - T0;
```

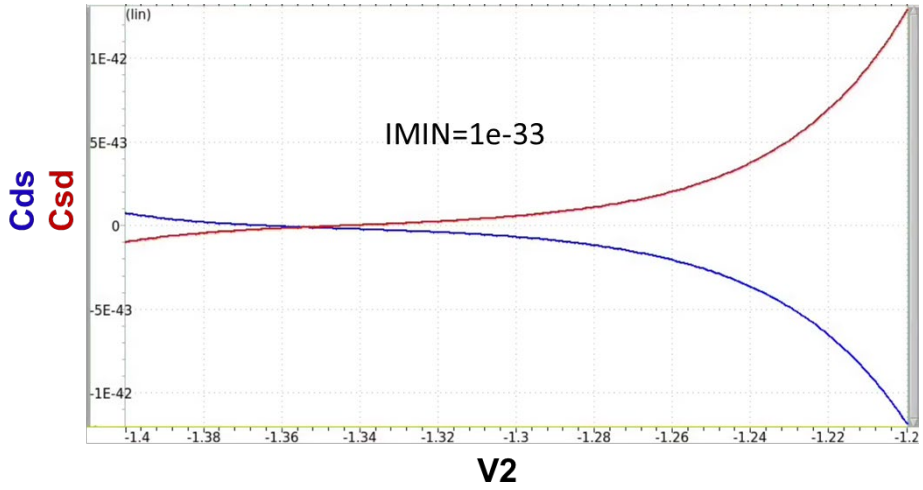
Here the Vgs clamp is limited by `lln(x), which gets clamped to LN_N_MINLOG =-87.498233534 for x < N_MINLOG 1.0e-38, which results in the minimum Vgs limit of ~ -3V (at T=125C), no matter how small IMIN becomes.

We have now rearranged the terms in the same formulation as follows, which can achieve a much smaller value of the minimum Vgs clamp.

**BSIM-CMG 112.0.0beta0_2**

```
T3 = pow((2.0 * cox * IMIN / (beta0 * nVtm * `q * Nc * TFIN)), nVtm);
T0 = -(dvch_qm + `lln(T3));
T1 = vgsfb + T0 + DELVTRAND;
vgsfbeff = `hypsmooth(T1, 1.0e-4) - T0;
```

The above formulation can achieve a minimum Vgs clamp value of -88.498V at T = 125 C and solves the capacitance discontinuity issue with IMIN=1e-33 as shown below.




31. **2023bug4(UCB): Incorrect units of binning parameters**

    The VAMPyRE testing of BSIM-CMG 112.0.0beta0_1 flagged incorrect format of the units m*V(^-1) of two parameters, PBTATD and PBTATS. The unit formats have now been corrected in BSIM-CMG 112.0.0beta0_2 as V^-1*m for both parameters.

32. **2024bug1(ADI): Correct the range of parameter TYPE.**

    In BSIM-CMG 112.0.0beta0_3, TYPE=0 is also an allowed value due to the following definition

```
`MPIcc(TYPE, `ntype, "", `ptype, `ntype, "1: NMOS; -1: PMOS")
```

    We have corrected it in BSIM-CMG 112.0.0beta0_3 by defining a new macro MPIty and using it for defining TYPE as

```
`define MPIty(nam, def, uni, des) (* units = uni, desc = des *) parameter integer nam = def from[ -1: 1] exclude 0;

`MPIty(TYPE, `ntype, "", "1: NMOS; -1: PMOS")
```

33. **2024bug2(UCB): Gummel symmetry issue in Cryogenic Coulomb scattering model**

For non-zero values UDS and UDD, the CRYOMOD=1 or 2 exhibits the Gummel symmetry issue. The issue is resolved in BSIM-CMG 112.0.0beta0_4 by introducing a Vds dependent factor in Dmob as:

For CRYOMOD $\neq 0$ or TFTMOD $= 1$

$$T1 = \begin{cases} q_{is} & \text{TFTMOD=0} \\ q_{frees} & \text{TFTMOD=1} \end{cases} \qquad T2 = \begin{cases} q_{id} & \text{TFTMOD=0} \\ q_{freed} & \text{TFTMOD=1} \end{cases}$$

$$D_{mob} = \begin{cases} 1 + UA(T) \cdot (E_{effa})^{EU} + \dfrac{UD(T)}{\left(\frac{1}{2}\cdot\left(1+\frac{UDS_{eff}(T)\cdot T1 + UDD_{eff}(T)\cdot T2}{1E-2/Cox}\cdot T3\right)\right)^{UCS(T)}} & \text{BULKMOD=0} \\ 1 + (UA(T) + UC(T)\cdot V_{eseff})\cdot (E_{effa})^{EU} + \dfrac{UD(T)}{\left(\frac{1}{2}\cdot\left(1+\frac{UDS_{eff}(T)\cdot T1 + UDD_{eff}(T)\cdot T2}{1E-2/Cox}\cdot T3\right)\right)^{UCS(T)}} & \text{BULKMOD=1} \end{cases}$$

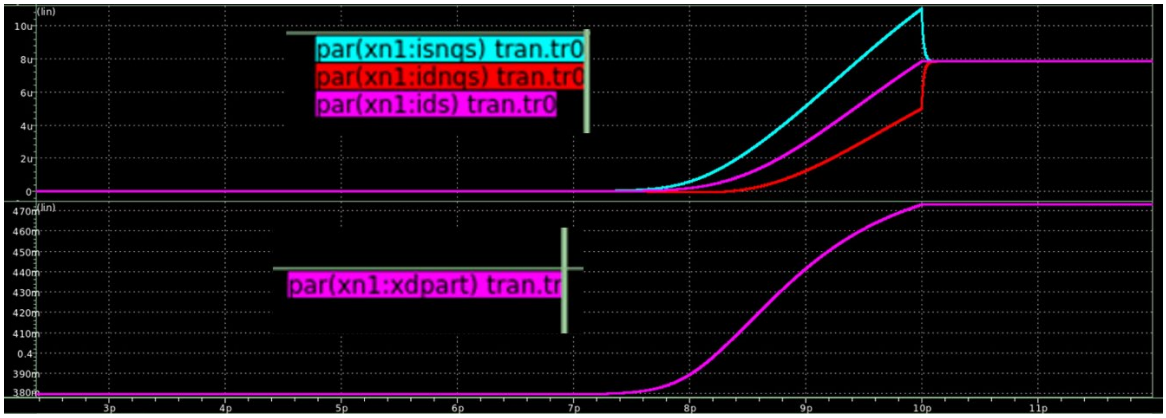
where $T3 = 1 - \exp(-V_{dseff}^2/6.25 \times 10^{-4})$ is a factor to preserve Gummel symmetry.

34. **2024bug3(UCB): Sign correction for variable xdpart in NQSMOD=2**

The existing model for NQSMOD=2 behaved incorrectly, with both the source and drain current components leading the drain-source current. This has been corrected by inverting the sign of the variable `xdpart`.

```
2649      if (NQSMOD == 2) begin
2650          xdpart = - qd / qg;
2651      end else begin
2652          xdpart = 0.0;
2653      end
```

With this change in place, the drain and source current components lag and lead, respectively, in transient simulations (as shown below):




35. **2024bug4(ADI): Order-of-operations problem in macros gammafunc and smoothmaxx2.**

In BSIM-CMG 112.0.0beta0_4 and older versions, the definitions of macros in general, and gammafunc and smoothmaxx2 in particular, did not conserve the order of operations. To correct this, all arguments after a minus, multiplication, or division sign in the macro are surrounded by a bracket. E.g.

The macro smoothmaxx2 was changed from:

```
`define smoothmaxx2(x,x0,deltax) (0.5*(x+x0-sqrt((x-x0)*(x-x0)+0.25*deltax*deltax)))
```

To:

```
`define smoothmaxx2(x,x0,deltax) (0.5*(x+x0-sqrt((x-(x0))*(x-(x0))+0.25*deltax*deltax)))
```

36. **2024bug5(ADI): Superfluous assignment to qm_ln.**

The variable qm_ln was being defined but never used. All instances of this variable were removed.

37. **2024bug6(ADI): Use updated hypsmooth function.**

The hypsmooth macro was improved following the improvements in the hypsmooth function in BSIM-BULK, aimed at preventing divide-by-zero errors caused by large negative arguments to hypsmooth. The existing macro reads as:

`define hypsmooth(x, c) (0.5 * (x + sqrt((x) * (x) + 4.0 * c * c)))

In the case that |x| is large, we can use a Taylor series:

sqrt(1 + ε) = 1 + ε/2 – ε^2/8 + ...

sqrt(x * x + 4.0 * c * c) = |x| * sqrt(1.0 + 4.0*c*c/(x*x))

≈ |x| * (1 + 2*(c/x)^2 – 2*(c/x)^4 + ...)

When x < 0, the leading 1 cancels the x in hypsmooth (x + |x| = 0), and the hypsmooth macro evaluates to zero. This can create problems if the output of the hypsmooth macro is used as a denominator – leading to divide-by-zero errors. To prevent this, large negative cases are handled by an updated definition as:

`define hypsmooth(x, c) ((x < -1e4 * (c)) ? (-(c) * (c) / (x)) : (0.5 * (x + sqrt((x) * (x) + 4.0 * (c) * (c)))))

38. **2024bug7(ADI): Initialize gate current variables.**

The variables igbacc_v, igbd_v, igbinv_v, igbs_v, igcd_v, igcs_v, igd_v, igidl_v, igisl_v, and igs_v were not initialized. This has been updated in the latest code, with these variables set to zero.

39. **2024bug8(IBM): Set a lower limit of zero to phib to prevent negative square root**

**evaluation.**

A lower limit of zero was introduced for the variable phib to prevent negative square root evaluations:

```
if (cryomod == 0) begin
        phib = Vtm * `lln(NBODY_i / ni);
        phib = `smoothminx(phib, 0, 1e-10);
        vbi = Vtm * `lln(NBODY_i * nsd / (ni * ni));
    end else begin
        phib = Vtm * (`lln(NBODY_i) - niln);
        phib = `smoothminx(phib, 0, 1e-10);
        vbi = Vtm * (`lln(NBODY_i * nsd) - 2 * niln);
    end
```

40. **2024bug9(IBM,ADI): Check for WGAAeff and Leff bounds right after their evaluation to prevent a divide-by-zero error**

The evaluation of WGAAeff involves an unbounded term XW_i, which can lead to WGAAeff being zero.

```
WGAAeff = wgaa + XW_i;
WGAAeff1 = WGAAeff + DWBIN_i;
```

This can cause a divide-by-zero error in the following lines:

```
if (geomod == 5) begin
       Inv_W = 1.0e-6 / WGAAeff1;
       Inv_WL = 1.0e-12 / (WGAAeff1 * Leff1);
```

Similarly, Lg = l + XL_i; and Lg can take a value of zero. This can cause Leff to be zero and a divide-by-zero error when evaluationg Inv_L.

To prevent these conditions, the following check was introduced right after the assignments for WGAAeff and Leff:

```
if (geomod == 5) begin
        if (WGAAeff1 <= 0.0) begin
            $error("Fatal: WGAAeff1 = %g is not positive.", WGAAeff1);
        end else if (WGAAeff1 <= 1.0e-9) begin
            $strobe("Warning: WGAAeff1 = %g <= 1.0e-9.", WGAAeff1);
        end
    end


if (Leff <= 0.0) begin
        $error("Fatal: Leff = %g is not positive.", Leff);
    end else if (Leff <= 1.0e-9) begin
```

```
        $strobe("Warning: Leff = %g <= 1.0e-9.", Leff);
    end

    if (Leff1 <= 0.0) begin
        $error("Fatal: Leff1 = %g is not positive.", Leff1);
    end else if (Leff1 <= 1.0e-9) begin
        $strobe("Warning: Leff1 = %g <= 1.0e-9.", Leff1);
    end
  if (LeffCV <= 1.0e-9) begin
        $strobe("Warning: LeffCV = %g <= 1.0e-9.", LeffCV);
    end
    if (bulkmod != 0) begin
        if (LeffCV_acc <= 1.0e-9) begin
            $strobe("Warning: LeffCV_acc = %g <= 1.0e-9.", LeffCV_acc);
        end
   end
```

## D. Description of Enhancements

1. **2021enh3(ADI): ln(1.0 + `lexp(x)) replacement with new function "ln_one_plus_exp"**

   BSIMCMG 111.1.0 uses ln(1.0 + `lexp(x)) terms at a few places.

   ln(1.0 + `lexp(x)) becomes

   - numerically indistinguishable from x for x > 37
   - numerically indistinguishable from 0 for x < -37

   For these cases it is not required to compute ln and `lexp.

   Therefore, ln(1.0 + `lexp(x)) is replaced with a new function ln_one_plus_exp defined as

```
analog function real ln_one_plus_exp;
    input x; real x;
    begin
        if (x > 37) begin
            ln_one_plus_exp = x;
        end else if (x < -37) begin
            ln_one_plus_exp = 0.0;
        end else begin
            ln_one_plus_exp = ln(1.0 + exp(x));
        end
    end
endfunction
```

2. **2021enh5(UCB): Cryogenic Temperature Model**

These models are introduced in BSIM-CMG111.2.0beta0_1 to capture the device physics and temperature effects down to low cryogenic temperatures, such as for the CMOS circuits used in quantum computing applications.

Cryogenic models can be enabled by CRYOMOD switch which can have the following three settings

CRYOMOD = 0 Turns off cryogenic models (same temperature models as BSIM-CMG111.1.0)

CRYOMOD = 1 Most physical cryogenic models

CRYOMOD = 2 Cryogenic temperature expressions converge to BSIM-CMG111.1.0 temperature expressions for T > 210 K (-63.15 C)

In summary, the cryogenic mode introduces new temperature models for the following effects

- Mobility power laws
- Drain bias dependence of Coulomb scattering
- Velocity Saturation and Pinch-Off Voltage
- Band Tail States and Fermi-Dirac Effects for SS and $V_{th}$
- Source/Drain Resistances

For details of the above models, please refer to the BSIMCMG_111.2.0 technical manual.

3. **2020enh6(UCB): Improve accuracy of C-V fitting for high Vd and Vg**

Existing BSIM-CMG C-V model scheme may not be sufficient to fit the CV data for different Vd and high Vg (see Fig. 4)

**Fitting using BSIM-CMG111.2.0beta0_1**

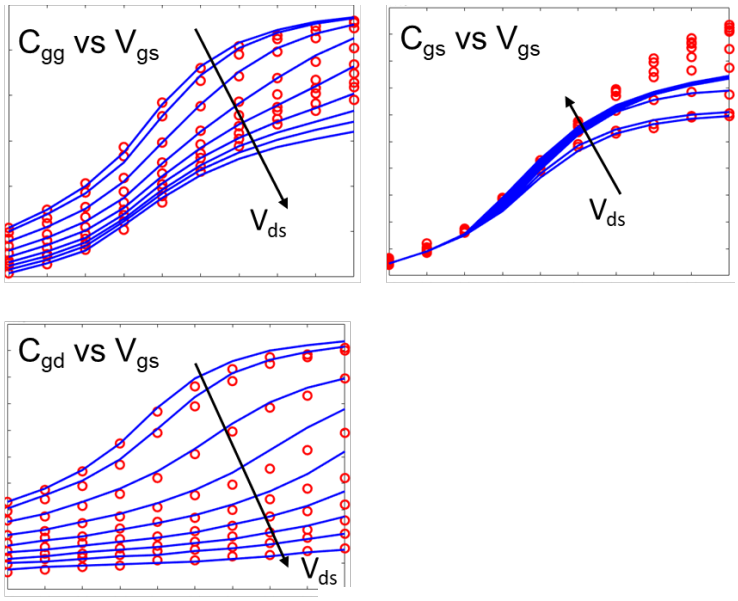



Fig. 4

To improve capacitance fitting flexibility additional, I-V and C-V parameters are decoupled

ETA0 → ETA0CV

U0 → U0CV

UA →UACV

UD → UDCV

UC → UCCV

CVMOD=1 is used to enable this feature.

Fitting using the updated scheme is shown in Fig. 5. As can be seen the decoupling provides a much better fit to the C-V data.

**Fitting using BSIM-CMG111.2.0beta0_2**

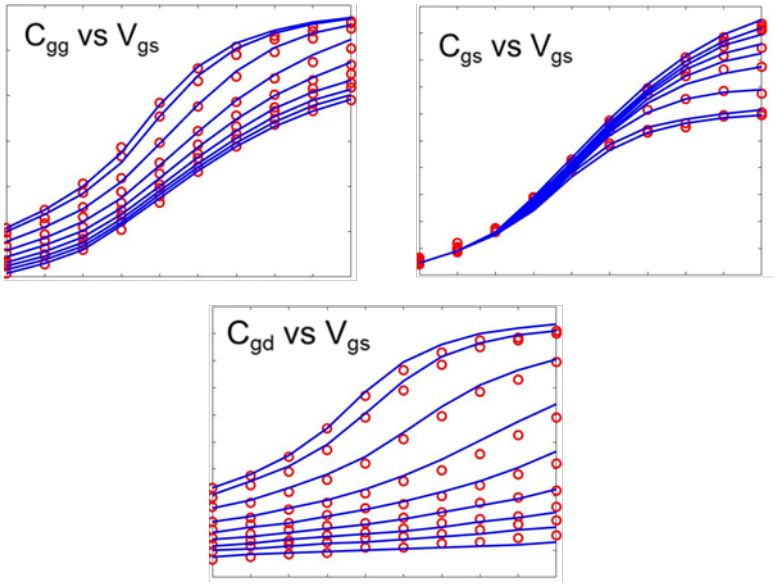



Fig. 5

4. **2021enh7(UCB): Improve the accuracy of I-V near Vdsat at high Vg: S/D velocity saturation model**

To capture velocity saturation effect in drain/ source regions, two bias dependent resistances, $R_{vs,d}$ and $R_{vs,s}$ are introduced in BSIM-CMG 111.2.0beta0_2 with the help of two extra nodes di1 and si1. These resistances are added between di1 and di, and si1 and si, respectively. The regular (velocity independent) resistances $R_{drain}$ and $R_{source}$, are now placed between d and di1, and s and si1, respectively. Nodes di1 and di are shorted when $R_{vs,d}$ and $R_{vs,s}$ are zero.

**Code implementation:**

$R_{drain}$ and $R_{source}$ for RDSMOD=1

```
  //Both bias-dependent and bias-independent parts of parasitic resistances are external
  1: begin
      Rdsi = 0.0;
      Dr = 1.0;
      T2 = devsign * V(gi, si1) - vfbsd;
      T3 = sqrt(T2 * T2 + 0.1);
      vgs_eff = 0.5 * (T2 + T3);
      T4 = 1.0 + PRWGS_i * vgs_eff;
      T1 = 1.0 / T4;
      T0 = 0.5 * (T1 + sqrt(T1 * T1 + 0.01));
      T5 = RSW_i * (1.0 + RSDR_a * pow((V(s, si1) * V(s, si1) + 1.0e-6), (0.5 * PRSDR)));
      Rsource = rdstemp * (RSourceGeo + (RSWMIN + T5 * T0) * WeffWRFactor);
      T2 = devsign * V(gi, di1) - vfbsd;
      T3 = sqrt(T2 * T2 + 0.1);
      vgd_eff = 0.5 * (T2 + T3);
      T4 = 1.0 + PRWGD_i * vgd_eff;
      T1 = 1.0 / T4;
      T0 = 0.5 * (T1 + sqrt(T1 * T1 + 0.01));
      T5 = RDW_i * (1.0 + RDDR_a * pow((V(d, di1) * V(d, di1) + 1.0e-6), (0.5 * PRDDR)));
      Rdrain = rdstemp * (RDrainGeo + (RDWMIN + T5 * T0) * WeffWRFactor);
  end
```

```
// Velocity saturation in source/drain resistances
if (RDSMOD == 1) begin
    if (RDLCW > 0) begin
        rdstempvs = `hypsmooth((1.0 + PRTVSRSD * delTemp - 1.0e-6), 1.0e-3);
        `tempdep(VSATRSD_t, VSATRSD, -ATVSRSD)
        T0 = qis - PTWG1VSRSD;
        T0 = `smoothminx(T0, 0.1, 2.0);
        T1  = 10.0 * PSATXVSRSD * T0 / (10.0 * PSATXVSRSD + T0);
        vsatrsd_eff = VSATRSD_t * (1.0 + PTWGVSRSD * T1);
        vsatrsd_eff = `hypsmooth(vsatrsd_eff, 10.0);
        T2 = NFINtotal * Weff0 * `q * vsatrsd_eff;
        if (GAVSRD == 0) begin
            T3 = 1.0;
        end else begin
            T3 = `smoothminx((devsign * V(di1, di) - RDVDS), 0, 0.5);
            T3 = 1.0 + T3 * GAVSRD;
        end
        isat_rd = T2 * NVSRD * T3;
        delta_vsrd = pow(ids, 4 - MVSRSD) / (pow(ids, 4 - MVSRSD) + VSRDFACTOR * pow(isat_rd, 4 - MVSRSD));
        T4 = rdstempvs * RDLCW * WeffWRFactor;
        T5 = pow(delta_vsrd, 1.0 / MVSRSD) * devsign * V(di1, di) / (isat_rd * T4);
        T5 = `smoothminx(T5, 0.0, 1.0e-3);
        Rvs_d = T4 * pow((1.0 + pow(T5, MVSRSD)), 1 / MVSRSD);
    end
    if (RSLCW > 0) begin
        if (RDLCW == 0.0) begin
            rdstempvs = `hypsmooth((1.0 + PRTVSRSD * delTemp - 1.0e-6), 1.0e-3);
            `tempdep(VSATRSD_t, VSATRSD, -ATVSRSD)
            T0 = qis - PTWG1VSRSD;
            T0 = `smoothminx(T0, 0.1, 2.0);
            T1  = 10.0 * PSATXVSRSD * T0 / (10.0 * PSATXVSRSD + T0);
            vsatrsd_eff = VSATRSD_t * (1.0 + PTWGVSRSD * T1);
            vsatrsd_eff = `hypsmooth(vsatrsd_eff, 10.0);
            T2 = NFINtotal * Weff0 * `q * vsatrsd_eff;
        end
        isat_rs = T2 * NVSRS;
        delta_vsrs = pow(ids, 4 - MVSRSD) / (pow(ids, 4 - MVSRSD) + VSRSFACTOR * pow(isat_rs, 4 - MVSRSD));
        T4 = rdstempvs * RSLCW * WeffWRFactor;
        T5 = pow(delta_vsrs, 1.0 / MVSRSD) * devsign * V(si, si1) / (isat_rs * T4);
        T5S = V(si1, si);
        T5 = `smoothminx(T5, 0.0, 1.0e-3);
        Rvs_s = T4 * pow((1.0 + pow(T5, MVSRSD)), 1 / MVSRSD);
    end
end
```

```
// External source/drain resistance
if (RDSMOD != 2 && RDrainGeo > 0) begin
    gdpr = 1.0 / Rdrain;
    I(d, di1) <+ V(d, di1) * gdpr;
    if (RDSMOD == 1 && RDLCW > 0) begin
        gvs_d = 1.0 / Rvs_d;
        I(di1, di) <+ V(di1, di) * gvs_d;
    end else begin
        V(di1, di) <+ 0.0;
    end
end else begin
    V(d, di1) <+ 0.0;
    V(di1, di) <+ 0.0;
end
```

```
if (RDSMOD != 2 && RSourceGeo > 0) begin
    gspr = 1.0 / Rsource;
    I(s, si1) <+ V(s, si1) * gspr;
    if (RDSMOD == 1 && RSLCW > 0) begin
        gvs_s = 1.0 / Rvs_s;
        I(si1, si) <+ V(si1, si) * gvs_s;
    end else begin
        V(si1, si) <+ 0.0;
    end
end else begin
    V(s, si1) <+ 0.0;
    V(si1, si) <+ 0.0;
end
```

```
// Thermal noise for parasitics
if (RDSMOD != 2 && RDrainGeo > 0) begin
    I(d, di1) <+ white_noise(4.0 * Vtm * `q * gdpr, "rd");
    if (RDSMOD == 1 && RDLCW > 0) begin
        I(di1, di) <+ white_noise(4.0 * Vtm * `q * gvs_d, "rd");
    end
end
if (RDSMOD != 2 && RSourceGeo > 0) begin
    I(s, si1) <+ white_noise(4.0 * Vtm * `q * gspr, "rs");
    if (RDSMOD == 1 && RSLCW > 0) begin
        I(si1, si) <+ white_noise(4.0 * Vtm * `q * gvs_s, "rs");
    end
end
```

```
// Self-heating
if (SHMOD != 0 && RTH0 > 0.0) begin
    Pwr(t) <+ -(devsign * sigvds * V(di, si) * ids);
    if (RDSMOD != 2 && RDrainGeo > 0) begin
        Pwr(t) <+ -V(d, di1) * V(d, di1) * gdpr;
        if (RDSMOD == 1 && RDLCW > 0) begin
            Pwr(t) <+ -V(di1, di) * V(di1, di) * gvs_d;
        end
    end
    if (RDSMOD != 2 && RSourceGeo > 0) begin
        Pwr(t) <+ -V(s, si1) * V(s, si1) * gspr;
        if (RDSMOD == 1 && RSLCW > 0) begin
            Pwr(t) <+ -V(si1, si) * V(si1, si) * gvs_s;
        end
    end
    Pwr(t) <+ Temp(t) * gth;
    Pwr(t) <+ ddt(Temp(t) * cth);
end else begin
    Temp(t) <+ 0.0;
end
```

```
    if (RDSMOD != 2) begin
        if (RDSMOD == 1 && (RDLCW > 0 || RSLCW > 0)) begin
            ITH = V(di, si) * ids + V(di1, di) * V(di1, di) * gvs_d + V(d, di1) * V(d, di1) * gdpr + V(si1, si) * V(si1, si) * gvs_s + V(s, si1)
* V(s, si1) * gspr;
        end else begin
            ITH = V(di, si) * ids + V(d, di1) * V(d, di1) * gdpr + V(s, si1) * V(s, si1) * gspr;
        end
    end else begin
        ITH = V(di, si) * ids;
    end
```

5. **2021enh8(UCB): Parasitic substrate GIDL/GISL model**

In geometries such as quadruple gate, the GIDL consists of two components (see Fig. below)

a) D to S band to band tunneling
b) D to E (substrate) band to band tunneling: Parasitic Substrate GIDL

Similarly, GISL consists of S to D BTBT tunneling and S to E BTBT tunneling.

BSIM-CMG 111.2.0beta0_1 model, considers the GIDL/GISL current to be flowing from D to E/S to E for BULKMOD!=0, which underestimates the source current and overestimates the substrate current. For example, for sigvds > 0

```
  if (BULKMOD != 0) begin
      I(di, e) <+ devsign * (igidl + Iii);
      I(si, e) <+ devsign * igisl;
      I(gi, e) <+ devsign * (igbinv + igbacc);
  end else begin
      I(di, si) <+ devsign * (igidl + Iii);
      I(si, di) <+ devsign * igisl;
  end
```

In BSIM-CMG 111. 2.0beta0_2, we separate out D to S/S to D and D to E/S to E components of GIDL/GISL for the following geometries

GEOMOD = 2 (Quadruple gate)

GEOMOD = 3 (Cylindrical gate)

GIDLMOD = 2 is used to enable this feature.

**BSIM-CMG 111.2.0beta0_2 GIDLMOD=2 model**

Drain to Source component of GIDL

$$T0 = AGIDL_i \cdot W_{eff0} \cdot \left( \frac{V_{ds} - V_{gs} - EGIDL_i + V_{fbsd}}{\epsilon_{ratio} \cdot EOT} \right)^{PGIDL_i} \times \exp\left( - \frac{\epsilon_{ratio} \cdot EOT \cdot BGIDL(T)}{V_{ds} - V_{gs} - EGIDL_i + V_{fbsd}} \right) \times NFIN_{total}$$

$$I_{gidl} = T0 \cdot \frac{V_{de}^3}{CGIDL_i + V_{de}^3}$$

Drain to Substrate component of GIDL

$$T0 = AGIDLB_i \cdot W_{effB} \cdot \left( \frac{V_{ds} - V_{gs} - EGIDLB_i + V_{fbsd}}{\epsilon_{ratio} \cdot EOT} \right)^{PGIDLB_i} \times \exp\left( - \frac{\epsilon_{ratio} \cdot EOT \cdot BGIDLB(T)}{V_{ds} - V_{gs} - EGIDLB_i + V_{fbsd}} \right) \times NFIN_{total}$$

$$I_{gidlb} = T0 \cdot \frac{V_{de}^3}{CGIDLB_i + V_{de}^3}$$

where

$$W_{effB} = \begin{cases} TFIN\_BASE & \text{for } GEOMOD = 2 \\ D & \text{for } GEOMOD = 3 \end{cases}$$

**Code Implementation**

```
// Parasitic substrate GIDL
if (BULKMOD != 0 && (GIDLMOD == 2 || GIDLMOD == 3) && (GEOMOD == 2 || GEOMOD == 3)) begin
    BGIDLB_t = BGIDLB_i * `hypsmooth((1.0 + TGIDL_i * delTemp - 1.0e-6), 1.0e-3);
    if ((AGIDLB_i <= 0.0) || (BGIDLB_t <= 0.0)) begin
        T7 = 0.0;
    end else begin
        T1 = (-vgd_noswap - EGIDLB_i + vfbsd) / T0;
        T1 = `hypsmooth(T1, 1.0e-2);
        T2 = BGIDLB_t / (T1 + 1.0e-3);
        T3 = pow(T1, PGIDLB_i);
        T4 = -ved_jct * ved_jct * ved_jct;
        T4a = CGIDLB_i + abs(T4) + 1.0e-5;
        T5 = `hypsmooth((T4 / T4a), 1.0e-6) - 1.0e-6;
        T7 = AGIDLB_i * WeffB * T3 * `lexp(-T2) * T5;
    end
end

// Parasitic substrate GISL
if (BULKMOD != 0 && (GIDLMOD == 2 || GIDLMOD == 3) && (GEOMOD == 2 || GEOMOD == 3)) begin
    BGISLB_t = BGISLB_i * `hypsmooth((1.0 + TGIDL_i * delTemp - 1.0e-6), 1.0e-3);
    if ((AGISLB_i <= 0.0) || (BGISLB_t <= 0.0)) begin
        T7 = 0.0;
    end else begin
        T1 = (-vgs_noswap - EGISLB_i + vfbsd) / T0;
        T1 = `hypsmooth(T1, 1.0e-2);
        T2 = BGISLB_t / (T1 + 1.0e-3);
        T3 = pow(T1, PGISLB_i);
        T4 = -ves_jct * ves_jct * ves_jct;
        T4a = CGISLB_i + abs(T4) + 1.0e-5;
        T5 = `hypsmooth((T4 / T4a), 1.0e-6) - 1.0e-6;
        T7 = AGISLB_i * WeffB * T3 * `lexp(-T2) * T5;
    end
end
```

9. **2022enh1(ADI): Add QA test to check any missing parameters**

A test with all the default parameters specified is added to QA of BSIMCMG111.2.0beta3. The purpose of this test is to catch any accidentally deleted parameters during the model development. For nmos this test is

```
test.....765_ParamCheck
temperature.....27
biases.....V(s)=0 V(b)=0 V(d)=1
biasSweep.....V(g)=0.0,1.0,0.5
outputs.....I(d) I(g) I(s) I(b)
modelParameters.....parameters/nmosDefaultParameters
```

The file nmosDefaultParameters contains all the BSIMCMG parameters with their default values.

A similar test is added for the pmos.

10. **2022enh2(UCB): Trap-assisted tunneling (TAT) GIDL model**

A new trap-assisted tunneling model has been added in BSIM-CMG 111.3.0beta0_1 to capture the GIDL leakage in the advanced technology nodes. A new GIDLMOD = 3 can be used to enable this model. Note that GIDLMOD=3 retains all the features of GIDLMOD=2 as well. The relevant model equations are

$$T0 = AGIDL_i \cdot W_{eff0} \cdot \left( \frac{V_{ds} - V_{gs} - EGIDL_i + V_{fbsd}}{\epsilon_{ratio} \cdot EOT} \right)^{PGIDL_i} \times \exp\left( - \frac{\epsilon_{ratio} \cdot EOT \cdot BGIDL(T)}{V_{ds} - V_{gs} - EGIDL_i + V_{fbsd}} \right) \times NFIN_{total}$$

$$T1 = ATATD_i \cdot W_{eff0} \cdot n_i \times NFIN_{total} \times \exp\left( \frac{BTATD_i \cdot (V_{ds} - V_{gs})^2 - CTATD(T) \cdot (V_{ds} - V_{gs}) - DTATD_i + V_{fbsd}}{V_{tm}} \right)$$

$$I_{gidl} = \begin{cases} (T0 + T1) \cdot \frac{V_{de}^3}{CGIDL_i + V_{de}^3} & \text{for } BULKMOD = 1 \\ (T0 + T1) \cdot V_{ds} & \text{for } BULKMOD = 0 \end{cases}$$

$$T0 = AGISL_i \cdot W_{eff0} \cdot \left( \frac{-V_{ds} - V_{gd} - EGISL_i + V_{fbsd}}{\epsilon_{ratio} \cdot EOT} \right)^{PGISL_i}$$
$$\times \exp\left( -\frac{\epsilon_{ratio} \cdot EOT \cdot BGISL(T)}{-V_{ds} - V_{gd} - EGISL_i + V_{fbsd}} \right) \times NFIN_{total}$$

$$T1 = ATATS_i \cdot W_{eff0} \cdot n_i \times NFIN_{total}$$
$$\times \exp\left( \frac{BTATS_i \cdot (-V_{ds} - V_{gd})^2 - CTATS(T) \cdot (-V_{ds} - V_{gd}) - DTATS_i + V_{fbsd}}{V_{tm}} \right)$$

$$I_{gisl} = \begin{cases} (T0 + T1) \cdot \frac{V_{se}^3}{CGISL_i + V_{se}^3} & \text{for } BULKMOD = 1 \\ (T0 + T1) \cdot V_{sd} & \text{for } BULKMOD = 0 \end{cases}$$

## Code Changes:

```
// GIDL current
if ((AGIDL_i <= 0.0) || (BGIDL_t <= 0.0)) begin
    T6 = 0.0;
end else begin
    T1 = (-vgd_noswap - EGIDL_i + vfbsd) / T0;
    T1 = `hypsmooth(T1, 1.0e-2);
    T2 = BGIDL_t / (T1 + 1.0e-3);
    T3 = pow(T1, PGIDL_i);
    if (BULKMOD != 0) begin
        T4 = -ved_jct * ved_jct * ved_jct;
        T4a = CGIDL_i + abs(T4) + 1.0e-5;
        T5 = `hypsmooth((T4 / T4a), 1.0e-6) - 1.0e-6;
        T6 = AGIDL_i * Weff0 * T3 * `lexp(-T2) * T5;
    end else begin
        T6 = AGIDL_i * Weff0 * T3 * `lexp(-T2) * vds_noswap;
    end
end
// Trap Assisted Tunneling GIDL
if (GIDLMOD == 3 && ATATD_i > 0.0) begin
    if (BULKMOD != 0) begin
        CTATD_t = CTATD_i * `hypsmooth((1.0 + TTAT_i * delTemp - 1.0e-6), 1.0e-3);
        T1 = (BTATD_i * vgd_noswap * vgd_noswap - CTATD_t * vgd_noswap - DTATD_i + vfbsd) / Vtm;
        T2 = ATATD_i * Weff0 * ni * `lexp(T1);
        T4 = -ved_jct * ved_jct * ved_jct;
        T4a = CGIDL_i + abs(T4) + 1.0e-5;
        T5 = `hypsmooth((T4 / T4a), 1.0e-6) - 1.0e-6;
        T6 = T6 + T2 * T5;
    end else begin
        CTATD_t = CTATD_i * `hypsmooth((1.0 + TTAT_i * delTemp - 1.0e-6), 1.0e-3);
        T1 = (BTATD_i * vgd_noswap * vgd_noswap - CTATD_t * vgd_noswap - DTATD_i + vfbsd) / Vtm;
        T2 = ATATD_i * Weff0 * ni * `lexp(T1);
        T6 = T6 + T2 * vds_noswap;
    end
end

// GISL current
if ((AGISL_i <= 0.0) || (BGISL_t <= 0.0)) begin
    T6 = 0.0;
end else begin
    T1 = (-vgs_noswap - EGISL_i + vfbsd) / T0;
    T1 = `hypsmooth(T1, 1.0e-2);
    T2 = BGISL_t / (T1 + 1.0e-3);
    T3 = pow(T1, PGISL_i);
    if (BULKMOD != 0) begin
        T4 = -ves_jct * ves_jct * ves_jct;
        T4a = CGISL_i + abs(T4) + 1.0e-5;
        T5 = `hypsmooth((T4 / T4a), 1.0e-6) - 1.0e-6;
        T6 = AGISL_i * Weff0 * T3 * `lexp(-T2) * T5;
    end else begin
        T6 = AGISL_i * Weff0 * T3 * `lexp(-T2) * (-vds_noswap);
    end
end
// Trap Assisted Tunneling GISL
if (GIDLMOD == 3 && ATATS_i > 0.0) begin
    if (BULKMOD != 0) begin
        CTATS_t= CTATS_i*`hypsmooth((1.0 + TTAT_i * delTemp - 1.0e-6), 1.0e-3);
        T1 = (BTATS_i * vgs_noswap * vgs_noswap - CTATS_t * vgs_noswap - DTATS_i + vfbsd) / Vtm;
        T2 = ATATS_i * Weff0 * ni * `lexp(T1);
        T4 = -ves_jct * ves_jct * ves_jct;
        T4a = CGISL_i + abs(T4) + 1.0e-5;
        T5 = `hypsmooth((T4 / T4a), 1.0e-6) - 1.0e-6;
        T6 = T6 + T2 * T5;
        itat = T2 * T5;
    end else begin
        CTATS_t= CTATS_i*`hypsmooth((1.0 + TTAT_i * delTemp - 1.0e-6), 1.0e-3);
        T1 = (BTATS_i * vgs_noswap * vgs_noswap - CTATS_t * vgs_noswap - DTATS_i + vfbsd) / Vtm;
        T2 = ATATS_i * Weff0 * ni * `lexp(T1);
        T6 = T6 + T2 * (-vds_noswap);
    end
end
```

11. **2022enh3(UCB): Add cryogenic model parameter extraction procedure in manual.**

    Parameter extraction procedure for a multiple temperature fitting using the cryogenic model (CRYOMOD=1 or 2) has been added as section 6 in the BSIM-CMG 111 .3.0beta0_1 manual.

12. **2023enh1(IBM): Improve cryogenic model parameter extraction procedure in the manual.**

    The cryogenic model extraction procedure has been updated in the manual of **BSIM-CMG 112.0.0beta0_2** in a tabular form to make it easy to follow. Notes are also provided below each table for more details. For example:

### 6.1 Drain Current Fitting in Linear Region

**Step 2.** Subhreshold region fitting: In this step, the $V_{GS}$ dependence of the drain current $I_{DS}$ in low $V_{GS}$ region is extracted.

| Extracted Parameters | Device & Experimental Data | Extraction Methodology |
|---|---|---|
| TLOW and DTLOW | $I_{ds}$ v.s. $V_{gs}$ @ $V_{ds} = 50mV$ @ multiple temperatures. | Observe Subthreshold swing (SS) @ different temperatures and tune TLOW and DTLOW to capture SS saturation. |
| TLOW1, KLOW1, DTLOW1 | $I_{ds}$ v.s. $V_{gs}$ @ $V_{ds} = 50mV$ @ multiple temperatures. | Observe sub-threshold swing (SS) @ different temperatures in cryogenic range for a rise in SS with T reduction. Tune TLOW1, KLOW1, DTLOW1 to capture the SS rise if any. |
| TVTH, KT11, KT12, KT1 | $I_{ds}$ v.s. $V_{gs}$ @ $V_{ds} = 50mV$ @ multiple temperatures. | Observe threshold voltage off-set @ different temperatures and tune TVTH, KT11, KT12, KT1. |

Notes:

- TLOW is used to extract the temperature at which SS saturates w.r.t temperature in the cryogenic range. DTLOW is used to smoothly transition to the SS saturation region @ low T from the usual SS behavior governed by the Boltzmann law ($\ln(10)kT/q$) @ high T. At very low temperatures, the SS may again start to rise with T and TLOW1 is the temperature below which this happens. DTLOW1 is used to capture the smoothness of this SS transition and KLOW1 is used to capture the rate at which the SS rises w.r.t T (assuming linear dependency on T).

13. **2023enh1: Reinstate the GAA functionality.**

We have reinstated the GAA functionality i.e. GEOMOD=5 (GAAFET geometry and related effects selector) and SUBBANDMOD (GAA quantum subband model

selector) in the BSIM-CMG_112.0.0beta0_3 release. The technical manual has also been updated accordingly.

14. **2024enh2(UCB): Thin Film Transistor Model.**

We have introduced compact models for capturing the device characteristics of Thin Film Transistors. The models can be enabled with TFTMOD = 1.
The full description of the models and an extraction procedure of the related parameters is available in the technical manual. Here is the summary of the models we introduced:

- Core model of charge density including the impact of trap charges
- Channel current model including trap limited conduction and percolation transport
- Source/Drain parasitic Schottky contacts in all RDSMODs
- Terminal charges qg, qd and qs
- Temperature dependence of traps

Other than the above models, the models for mobility, velocity saturation, non-saturation effect, lateral non-uniform doping effect, body effect, CLM and DIBL effects, Quantum mechanical effects, gat tunneling current, NQS model, and bias dependency of parasitic Source/Drain resistances are also modified for TFTMOD = 1.

15. **2024enh3(ADI, UCB): Add QA tests for all parameters.**

Added QA tests 885 -1038 for TFTMOD=1 and addressing the following issues:

- No test for parameter NGCON = 2
- No test of values [0, 2, 3, 4, 5, 6] for parameter NGAA
- No test for parameter MOBSCMOD = 1
- No test for parameter TYPE = 0
- No test for parameter BULKMOD = 2

- No test of values [1, 2, 3, 4] for parameter GEOMOD
- No test for parameter CGEO1SW = 1
- No test for parameter RDSMOD = 1
- No test for parameter ASYMMOD = 1
- No test for parameter CVMOD = 1
- No test of values [2, 3] for parameter GIDLMOD
- No test of values [1, 2] for parameter IIMOD
- No test for parameter TNOIMOD = 1
- No test of values [1, 2] for parameter NQSMOD
- No test for parameter TEMPMOD = 1
- No test for parameter RGATEMOD = 1
- No test for parameter RGEOMOD = 1
- No test of values [1, 2, 3] for parameter CGEOMOD
- No test for parameter FNMOD = 1
- No test for parameter SH_WARN = 1
- No test for parameter IGCLAMP = 0
- No test for parameter SDTERM = 1

16. **2024enh4(Rapidus): Improved gds fitting in low gate overdrive.**

Short channel devices tend to have the output conductance (gds) less sensitive to the drain voltage under low gate overdrive. This behavior is captured using an updated $\Delta V_{th,DIBL}$ defined using a new parameter ETA1 as:

$$\Delta V_{th,DIBL} = \Theta_{DIBL} ETA0_i \cdot \left(V_{dsx} + ETA1\sqrt{V_{dsx} + 0.01}\right) + DVTP0 \cdot \Theta_{DITS} \cdot (V_{dsx} + 0.01)^{DVTP1}$$

This expression follows a similar treatment as 2017enh6 for BSIM-IMG 102.9.1.

17. **2024enh5(ADI,UCB): Change parameter case according to VA standard.**
Changed the parameters in the code to lowercase to follow the latest Verilog-A standards. All variables with conflicting names were suffixed with _v. The parameter D, when converted to lowercase, conflicts with the drain node d. This

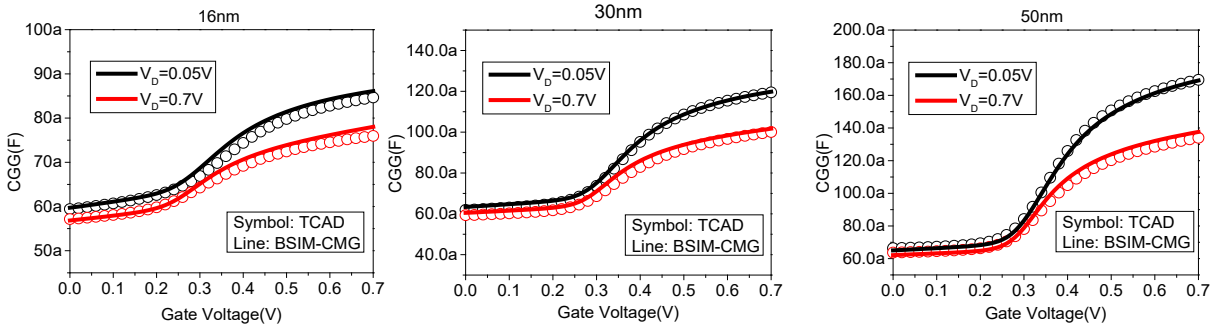
parameter was renamed to `dia`.

18. **2024enh6(UCB): Flexibility for capacitance tuning in the saturation region.**

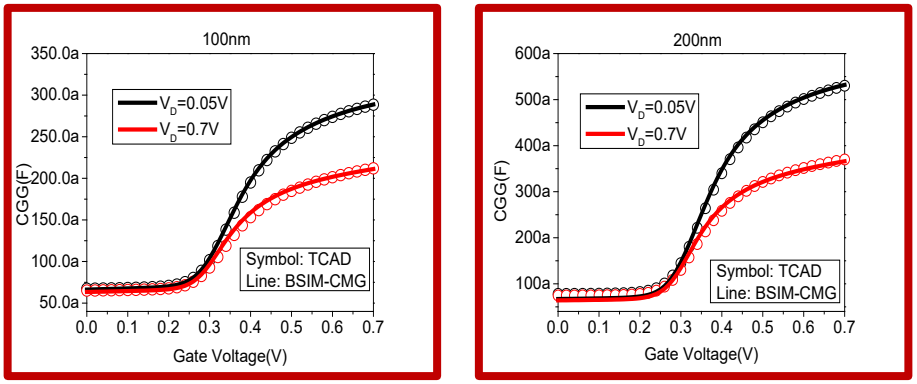
ASAT (CV fitting parameter for high $V_D$) and its corresponding binning parameters have been added. The updated expression for gate charge now reads:

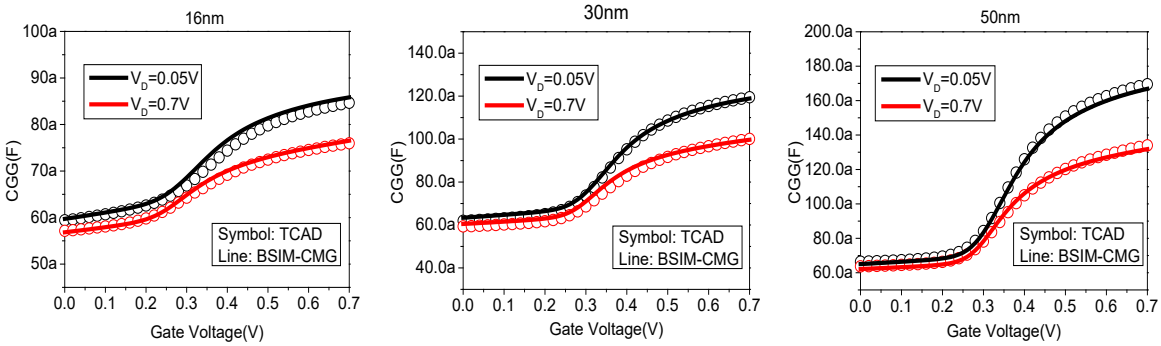
$$qg = qia + dqi * \frac{dqi}{6(2qia + nVtm)} * \boldsymbol{ASAT} * DvsatCV$$

Before ASAT:




After ASAT:






Further, binning has been introduced for the parameter PQM, with an additional length scaling parameter PQML for fitting flexibility.

19. **2024enh7(ADI): Removing NF from parameters/nmosDefaultParameters and parameters/pmosDefaultParameters in the QA package, and redefining the parameter check with NF as an instance parameter.**

    NF is now defined as an instance parameter for the default parameter check in QA. The updated qaSpec now reads:

```
test                1038_ParamCheck
temperature         27
biases              V(s)=0 V(e)=0 V(d)=1
biasSweep           V(g)=0.0,1.0,0.5
outputs             I(d) I(g) I(s) I(e)
instanceParameters  NF=1
modelParameters     parameters/nmosDefaultParameters
```

    NF has been removed from the files parameters/nmosDefaultParameters and parameters/pmosDefaultParameters.

20. **2024enh8(IBM): Improved Self-Heating model for GAAFETs.**

    An improved self-heating model is implemented to include the impact of WGAA and NGAA on self-heating in GAAFETs. The model activates only when GEOMOD=5, and defines the thermal resistance and capacitance as:

$$\frac{1}{R_{th}} = G_{th} = \frac{\left(WTH0 \cdot NF^{BSHEXP} + FPITCH \cdot NFINtotal^{\text{ASHEXP}} + \frac{WGAA}{WGAANOM} \cdot NGAA^{CSHEXP}\right)}{RTH0}$$

$$C_{th} = CTH0 \cdot \left(WTH0 \cdot NF^{BSHEXP} + FPITCH \cdot NFINtotal^{\text{ASHEXP}} + \frac{WGAA}{WGAANOM} \cdot NGAA^{CSHEXP}\right)$$

21. **2024enh9(Intel): Additional parameters to model mobility degradation due to hot carriers.**

In the source and drain regions, there is a chance for a concentration of hot-carrier defects which can be impacted by hot carriers and lead to interface states to appear at the Si surface. This can lead to a shift in the threshold voltage and a degradation in the mobility because of scattering. The mobility degradation is modeled with two model parameters ***MUHC0*** and ***MUHC1*** through the following equation:

$$u0mult\_v = U0MULT \cdot \left(1 - MUHC0 \cdot exp\left(-MUHC1 \cdot V_{dseff}\right)\right)$$

- Before stress: MUHC0/MUHC1=0 (default)
- After stress: MUHC0 is [0, 1) to model mobility degradation
  After stress: MUHC1 is [0, 3) to model the Vds dependence of mobility degradation.

22. **2024enh10(IBM): Improved Self-Heating model for GAAFETs**

Linear scaling terms ASH and CSH have been introduced to improve flexibility.

When GEOMOD=5, thermal resistance ($R_{th}$) and capacitance ($C_{th}$) are defined as:

$$\frac{1}{R_{th}} = G_{th} = \frac{(WTH0 \cdot NF^{BSHEXP} + ASH \cdot FPITCH \cdot NFINtotal^{ASHEXP})}{RTH0} + \frac{(CSH \cdot WGAA \cdot NF \cdot NGAA^{CSHEXP})}{RTH0}$$

$$C_{th} = CTH0 \cdot (WTH0 \cdot NF^{BSHEXP} + ASH \cdot FPITCH \cdot NFINtotal^{ASHEXP} + CSH \cdot WGAA \cdot NF \cdot NGAA^{CSHEXP})$$

Otherwise:

$$\frac{1}{R_{th}} = G_{th} = \frac{(WTH0 \cdot NF^{BSHEXP} + ASH \cdot FPITCH \cdot NFINtotal^{ASHEXP})}{RTH0}$$

$$C_{th} = CTH0 \cdot (WTH0 \cdot NF^{BSHEXP} + ASH \cdot FPITCH \cdot NFINtotal^{ASHEXP})$$

23. **2024enh12(CMC Policy): Removed TFT feature from 112.0.0beta3**

Following CMC recommendations, the TFT feature was removed from BSIM-CMG 112.0.0beta3.

24. **2024enh13(ADI, QA subcommittee): Extended QA to evaluate non-default parameter values.**

The following tests were added to the QA, including 3 modelcards with randomly assigned non-default parameters. The tests include:

918_ParamCheck_DC_NDefault
919_ParamCheck_AC_NDefault
920_ParamCheck_AC_Freq_NDefault
921_ParamCheck_Noise_NDefault
922_ParamCheck_DC_AllMods_NDefault
923_ParamCheck_AC_AllMods_NDefault
924_ParamCheck_AC_Freq_AllMods_NDefault
925_ParamCheck_Noise_AllMods_NDefault
926_ParamCheck_DC_AllMods_NDefault
927_ParamCheck_AC_AllMods_NDefault
928_ParamCheck_AC_Freq_AllMods_NDefault
929_ParamCheck_Noise_AllMods_NDefault
930_ParamCheck_DC_AllMods_NDefault
931_ParamCheck_AC_AllMods_NDefault
932_ParamCheck_AC_Freq_AllMods_NDefault
933_ParamCheck_Noise_AllMods_NDefault
934_ParamCheck_DC_AllMods_NDefault
935_ParamCheck_AC_AllMods_NDefault
936_ParamCheck_AC_Freq_AllMods_NDefault
937_ParamCheck_Noise_AllMods_NDefault
938_ParamCheck_DC_AllMods_NDefault
939_ParamCheck_AC_AllMods_NDefault
940_ParamCheck_AC_Freq_AllMods_NDefault
941_ParamCheck_Noise_AllMods_NDefault