Listing of the FBH HBT Model

Matthias Rudolph
Ferdinand-Braun-Institut für Höchstfrequenztechnik (FBH),
Gustav-Kirchhoff-Str. 4, D-12489 Berlin, Germany
rudolph@fbh-berlin.de, http://www.fbh-berlin.de/modeling.html

ver 2.1.20050728

```
/*
  FBH_HBT model version 2.1.20050728

  Copyright (C) 2005 Ferdinand-Braun-Institut
              im Forschungsverbund Berlin (FBH)
              Gustav-Kirchhoff-Str. 4
              D-12489 Berlin

  All rights reserved.

  By downloading this code, you agree that FBH shall not be held
  to any liability with respect to any claim by you or from any
  third party arising from or on account of the use of this code,
  regardless of the form of action, including negligence. In no event
  will FBH be liable for consequential or incidental damages of
  any nature whatsoever.

  Model documentation:

     www.fbh-berlin.de/modeling.html
     rudolph@fbh-berlin.de

*/

‘include "disciplines.vams"
‘include "constants.vams"
‘include "compact.vams"

‘define STDTEMP  20.0
‘define KDURCHQ  0.861708692e-4

‘define FOUR_K (4 * 1.3806226e-23)
‘define TWO_Q (2 * 1.6021918e-19)

‘define sqr(x)  (x*x)

// begin of FBH HBT model
module HBT_Xb(c,b,e,t);

//external nodes
inout e,b,c,t;
electrical e,b,c,t;

//internal nodes
electrical ei, bi, bii, ci, ti, ex, exx, cx, ni, nii;

//model parameters
```

```
parameter integer Mode = 1  from [0:4];              // Ignored
parameter integer Noise = 1  from [0:4];             // Ignored
parameter integer Debug = 0  from [0:inf);           // Ignored
parameter integer DebugPlus = 0  from [0:inf);       // Ignored

parameter real Temp = 25.0    from [-273.15:inf);
  // Device operating temperature, Celsius
parameter real Rth = 0.1    from [0.0:inf);
  // Thermal resistance, K/W
parameter real Cth = 700n     from [0.0:inf);
  // Thermal capacitance

parameter integer N = 1      from (0:inf);
  // Scaling factor, number of emitter fingers
parameter real L = 30u     from (0.0:inf);
  // Length of emitter finger, m
parameter real W = 3u     from (0.0:inf);
  // Width of emitter finger, m

parameter real Jsf = 20e-24  from [0.0:inf);
  // Forward saturation current density, A/um^2
parameter real nf = 1.0    from [0.0:inf);
  // Forward current emission coefficient
parameter real Vg = 1.3    from [-2.0:inf);
   // Forward thermal activation energy, V,
   // (0 == disables temperature dependence)

parameter real Jse = 0.0    from [0.0:inf);
  // B-E leakage saturation current density, A/um^2
parameter real ne = 0.0    from [0.0:inf);
   // B-E leakage emission coefficient
parameter real Rbxx = 1e6    from (0.0:inf);
  // Limiting resistor of B-E leakage diode, Ohm
parameter real Vgb = 0.0    from [0.0:inf);
  // B-E leakage thermal activation energy, V,
  // (0 == disables temperature dependence)

parameter real Jsee = 0.0    from [0.0:inf);
  // 2nd B-E leakage saturation current density, A/um^2
parameter real nee = 0.0    from [0.0:inf);
  // 2nd B-E leakage emission coefficient
parameter real Rbbxx= 1e6    from (0.0:inf);
  // 2nd Limiting resistor of B-E leakage diode, Ohm
parameter real Vgbb = 0.0    from [0.0:inf);
  // 2nd B-E leakage thermal activation energy, V,
  // (0 == disables temperature dependence)

parameter real Jsr = 20e-18  from [0.0:inf);
  // Reverse saturation current density, A/um^2
parameter real nr = 1.0    from [0.0:inf);
  // Reverse current emission coefficient
parameter real Vgr = 0.0    from [0.0:inf);
  // Reverse thermal activation energy, V,
  // (0 == disables temperature dependence)
parameter real XCjc = 0.5    from [0.0:1.0);
  // Fraction of Cjc that goes to internal base node

parameter real Jsc = 0.0    from [0.0:inf);
  // B-C leakage saturation current density, A/um^2
  // (0. switches off diode)
parameter real nc = 0.0    from [0.0:inf);
  // B-C leakage emission coefficient (0. switches off diode)
parameter real Rcxx = 1e6    from (0.0:inf);
```

```
    // Limiting resistor of B-C leakage diode, Ohm
  parameter real Vgc = 0.0    from [0.0:inf);
    // B-C leakage thermal activation energy, V,
    // (0 == disables temperature dependence)

  parameter real Bf = 100.0    from [0.0:inf);
    // Ideal forward beta
  parameter real kBeta= 0.0    from [0.0:inf);
    // Temperature coefficient of forward current gain, -1/K,
    // (0 == disables temperature dependence)
  parameter real Br = 1.0    from [0.0:inf);
    // Ideal reverse beta

  parameter real VAF = 0.0    from [0.0:inf);
    // Forward Early voltage, V,  (0 == disables Early Effect)
  parameter real VAR = 0.0    from [0.0:inf);
    // Reverse Early voltage, V, (0 == disables Early Effect)

  parameter real IKF = 0.0    from [0.0:inf);
    // Forward high-injection knee current, A,
    // (0 == disables Webster Effect)
  parameter real IKR = 0.0    from [0.0:inf);
    // Reverse high-injection knee current, A,
    // (0 == disables Webster Effect)

  parameter real Mc = 0.0    from [0.0:inf);
    // C-E breakdown exponent, (0 == disables collector break-down)
  parameter real BVceo= 0.0    from [0.0:inf);
    // C-E breakdown voltage, V, (0 == disables collector break-down)
  parameter real kc = 0.0    from [0.0:inf);
    // C-E breakdown factor, (0 == disables collector break-down)

  parameter real BVebo= 0.0    from [0.0:inf);
    // B-E breakdown voltage, V, (0 == disables emitter break-down)

  parameter real Tr = 1f     from [0.0:inf);
    // Ideal reverse transit time, s
  parameter real Trx = 1f     from [0.0:inf);
    // Extrinsic BC diffusion capacitance, s
  parameter real Tf = 1p     from [0.0:inf);
    // Ideal forward transit time, s
  parameter real Tft = 0.0    from [0.0:inf);
    // Temperature coefficient of forward transit time
  parameter real Thcs = 0.0    from [0.0:inf);
    // Excess transit time coefficient at base push-out
  parameter real Ahc = 0.0    from [0.0:inf);
    // Smoothing parameter for Thcs

  parameter real Cje = 1f     from [0.0:inf);
    // B-E zero-bias depletion capacitance, F/um^2
  parameter real mje = 0.5    from [0.0:1);
    // B-E junction exponential factor
  parameter real Vje = 1.3    from [0.0:inf);
    // B-E junction built-in potential, V

  parameter real Cjc = 1f     from [0.0:inf);
    // B-C zero-bias depletion capacitance, F/um^2
  parameter real mjc = 0.5    from [0.0:inf);
    // B-C junction exponential factor
  parameter real Vjc = 1.3    from [0.0:inf);
    // B-C junction built-in potential, V
  parameter real kjc = 1.0    from (-inf:inf);
    // not used
```

```
parameter real Cmin = 0.1f   from [0.0:inf);
  // Minimum B-C depletion capacitance (Vbc dependence), F/um^2

parameter real J0   = 1e-3   from [0.0:inf);
  // Collector current where Cbc reaches Cmin, A/um^2
  // (0 == disables Cbc reduction)
parameter real XJ0  = 1.0    from [0.0:1.0];
  // Fraction of Cmin, lower limit of BC capacitance (Ic dependence)
parameter real Rci0 = 1e-3   from (0.0:inf);
  // Onset of base push-out at low voltages, Ohm*um^2
  // (0 == disables base push-out)
parameter real Jk   = 4e-4   from [0.0:inf);
  // Onset of base push-out at high voltages, A/um^2,
  // (0 == disables base push-out)
parameter real RJk  = 1e-3   from [0.0:inf);
  // Slope of Jk at high currents , Ohm*um^2
parameter real Vces = 1e-3   from [0.0:inf);
  // Voltage shift of base push-out onset, V

parameter real Rc = 1.0    from (0.0:inf);
  // Collector resistance, Ohm/finger
parameter real Re = 1.0    from (0.0:inf);
  // Emitter resistance, Ohm/finger
parameter real Rb = 1.0    from (0.0:inf);
  // Extrinsic base resistance, Ohm/finger
parameter real Rb2 = 1.0    from (0.0:inf);
  // Inner Base ohmic resistance, Ohm/finger

parameter real Lc = 0.0    from [0.0:inf);
  // Collector inductance, H --- not yet implemented
parameter real Le = 0.0    from [0.0:inf);
  // Emitter inductance, H   --- not yet implemented
parameter real Lb = 0.0    from [0.0:inf);
  // Base inductance, H      --- not yet implemented

parameter real Cq = 0.0    from [0.0:inf);
  // Extrinsic B-C capacitance, F
parameter real Cpb = 0.0    from [0.0:inf);
  // Extrinsic base capacitance, F
parameter real Cpc = 0.0    from [0.0:inf);
  // Extrinsic collector capacitance, F

parameter real Kfb = 0.0    from [0.0:inf);
  // Flicker-noise coefficient
parameter real Afb = 0.0    from [0.0:inf);
  // Flicker-noise exponent
parameter real Ffeb = 0.0    from [0.0:inf);
  // Flicker-noise frequency exponent
parameter real Kb = 0.0    from [0.0:inf);
  // Burst noise coefficient
parameter real Ab = 0.0    from [0.0:inf);
  // Burst noise exponent
parameter real Fb = 0.0    from (0.0:inf);
  // Burst noise corner frequency, Hz
parameter real Kfe = 0.0    from [0.0:inf);
  // Flicker-noise coefficient
parameter real Afe = 0.0    from [0.0:inf);
  // Flicker-noise exponent
parameter real Ffee = 0.0    from [0.0:inf);
  // Flicker-noise frequency exponent

parameter real Tnom = 20.0    from [-273.15:inf);
  // Ambient temperature at which the parameters were determined
```

```
// general functions
//

// kT/Q
analog function real Vth;
    input TT;
    real TT, KDURCHQ;
    begin
        KDURCHQ=0.861708692e-4;

        Vth  = KDURCHQ*(TT +273.15);

    end
endfunction

// safe exponential function
analog function real exp_soft;
    input x;
    real x, maxexp, maxarg;
    begin

        maxexp = 1.0e25;
        maxarg = ln(maxexp);
        if (x < maxarg) begin
            exp_soft = exp(x);
        end
        else begin
            exp_soft = (x+1.0-maxarg)*(maxexp);
        end
    end
endfunction

 // limited internal Voltage
 analog function real Vt;
    input U, Ud;
    real  U, Ud, Vch, VF;
    begin
       Vch = 0.1 * Ud;
       VF  = 0.9 * Ud; // we fix this value for simplicity.

       if (U < VF)
            Vt = U  - Vch * ln(1.0 + exp((U-VF)/Vch));
       else
            Vt = VF - Vch * ln(1.0 + exp((VF-U)/Vch));
    end
 endfunction

// diode function
analog function real diode;
    input U, Is, Ug, N, AREA, TJ, TNOM;
    real  U, Is, Ug, N, AREA, TJ, TNOM, VTH0, VTHJ, VTHNOM, maxi,
          Tmax, TJM, KDURCHQ, lnIs;
    begin

        VTH0=Vth(20.0);
        VTHNOM=Vth(TNOM);
        KDURCHQ = 0.861708692e-4;
        lnIs=ln(Is*AREA);

        maxi=ln(1e6);
        if ((maxi<(Ug/VTHNOM)) && (U < 0.0))
           begin
```

```
               Tmax= Ug*VTHNOM/((Ug - maxi*VTHNOM)*KDURCHQ) - 273.15;
               TJM=Vt(TJ,Tmax);
            end
         else
            begin
               TJM=TJ;
            end
         VTHJ = Vth(TJM);

         if (Ug > 0.0) begin
              diode = exp_soft(U/(N*VTHJ) + Ug/VTHNOM - Ug/VTHJ + lnIs) -
                      exp_soft(Ug/VTHNOM - Ug/VTHJ + lnIs);
         end
         else begin
             diode = exp_soft(U/(N*(VTH0)) + lnIs) - Is*AREA;
         end
     end
 endfunction

 // CE-breakdown function
analog function real MM;
     input VBCI, VCBO, MC, VCBLIN, BF, KC;
     real  VBCI, VCBO, MC, VCBLIN, BF, KC;
     real FBD, vcbi;
     begin

         if((KC > 0.0) && (MC > 0.0) && (VCBO > 0.0)) begin
             vcbi = VBCI;
             FBD = VCBLIN/VCBO;
             if(VBCI > 0.0)
                 MM = 1.0;
             else if(VBCI > (-VCBLIN)) begin
                 if (MC==1)
                     MM = 1.0/(1.0 - (vcbi/(-VCBO)));
                 else
                     MM = 1.0/(1.0 - pow(vcbi/(-VCBO),MC));
             end
             else if(VBCI <= (-VCBLIN)) begin
                 if (MC==1) begin
                     MM = 1.0/(1.0 - FBD) - 1.0/VCBO *
                         1.0/pow(1.0 - FBD,2.0) * (vcbi + FBD*VCBO);
                 end
                 else begin
                     MM = 1.0/(1.0 - pow(FBD,MC)) - MC/VCBO *
                         pow(FBD,MC-1.0)/pow(1.0 -
                         pow(FBD,MC),2.0) * (vcbi + FBD*VCBO);
                 end
             end
         end
         else
             MM = 1.0;
     end
 endfunction


  // Depletion Charge
  analog function real charge;
      input U, C0, Ud, m, Area;
      real U, C0, Ud, m, Area, Vj, Vjo, VF;
      begin
         Vj  = Vt(U,Ud);
         Vjo = Vt(0.0,Ud);
         VF  = 0.9 * Ud; // we fix this value for simplicity.
```

```
        if(m==1.0) begin
            charge = Area*(C0)*
                   ( Ud*( ln(1.0 - Vjo/Ud) -
                          ln(1.0 - Vj/Ud)
                        ) +
                     1.0/(1.0 - VF/Ud) * (U - Vj + Vjo));
         end
         else begin
          charge = Area*(C0)*
                 ( (Ud/(1.0-m))*( pow(1.0 - Vjo/Ud , 1.0-m) -
                                  pow(1.0 - Vj/Ud , 1.0-m)
                                ) +
                   pow(1.0 - VF/Ud,-m) * (U - Vj + Vjo) -
                   Ud*(1.0/(1.0-m)));
         end
      end
 endfunction


// limited internal Voltage
analog function real Vceff;
    input U, VCES;
    real  U, VCES, Vth0;
    begin
       Vth0 = 0.025;

       if (U < VCES)
          Vceff = Vth0  + Vth0 * ln(1.0 + exp((U-VCES)/Vth0 - 1.0));
       else
          Vceff = (U-VCES) + Vth0 * ln(1.0 + exp(1.0-(U-VCES)/Vth0));
    end
endfunction

// Current for Onset of Kirk effect
analog function real ICK;
   input U, RCI0, VLIM, InvVPT, VCES;
   real  U, RCI0, VLIM, InvVPT, VCES, VC, x;
   begin
      VC  = Vceff(U,VCES);
      x   = (VC - VLIM)*InvVPT;
      ICK = VC/RCI0 * (1.0/sqrt(1.0 + (VC/VLIM)*(VC/VLIM)))*(
                     1.0 + (x + sqrt((x*x)+0.001))/2.0);
   end
endfunction



//local variables
real vbcx, vbci, vbei, vxe, vxxe, vxc, vcei;
real Ic0, Ic, Ic1, Ic1r, Ib2, Ibx,
     Ib0, Ibdx, Icdx, Ibdxx, Ib1, Ic0a, Ic1ra,
     Ipdiss, Ik, eps, IcIk;
real qb2;
real qb2x, qb2med, qb1, xtff, qbe, qbtr,
     qbtra, qbtf;
real EdBeta, mm;
real epsi, Vbclin;
real Texi, Tex, Tj, TjK, Area;
real RCIO, AHC, Ih, Wh, Vlim, InvVpt, q1, q2, qb, I00;
real xix;
real FOUR_K,TWO_Q ;
```

```
analog begin

  //
  // begin of model equations
  //
  // Port Voltages
  vbcx = V(bi,ci);
  vbci = V(bii,ci);
  vbei = V(bii,ei);
  vxe = V(ex,ei);
  vxc = V(cx,ci);
  vxxe = V(exx,ei);
  vcei = V(ci,ei);

  Texi = V(ti);
  Tj = Texi + Temp;  // Junction temperature
  TjK = Tj+273.15;   // Junction temperature in K
  Tex = Tj - Tnom;    // Temperature difference to reference

  Area =  L*W*(1.0e12) * N;     // Transistor area in um^2

  FOUR_K = 4 * 1.3806226e-23; // 4 k for noise
  TWO_Q  = 2 * 1.6021918e-19; // 2 q for noise

  //
  // Nonlinear Part --- Current Sources
  //
  // Collector Currents

  Ic0a = diode(vbei,Jsf,Vg,nf,Area,Tj,Tnom);
  Ic1ra = diode(vbci,XCjc*Jsr,Vgr,nr,Area,Tj,Tnom);

  // Early-Effect borrowed from VBIC
  if((VAF >0.0) && (VAR >0.0)) begin
      q1 = (1.0 + (charge(vbei,1.0,Vje,mje,1.0)-
                   charge(0.0,1.0,Vje,mje,1.0))/VAR +
                  (charge(vbci,1.0,Vjc,mjc,1.0)-
                   charge(0.0,1.0,Vjc,mjc,1.0))/VAF);
  end
  else if((VAF >0.0) && (VAR == 0.0)) begin
      q1 = (1.0 + (charge(vbci,1.0,Vjc,mjc,1.0)-
                   charge(0.0,1.0,Vjc,mjc,1.0))/VAF);
  end
  else if((VAF ==0.0) && (VAR > 0.0)) begin
      q1 = (1.0 + (charge(vbei,1.0,Vje,mje,1.0)-
                   charge(0.0,1.0,Vje,mje,1.0))/VAR);
  end
  else begin
      q1 = 1.0;
  end

  // Webster Effect borrowed from VBIC
  if((IKF > 0.0) && (IKR > 0.0)) begin
      q2 = Ic0a/(Area*IKF) + Ic1ra/(Area*IKR);
  end
  else if((IKF > 0.0) && (IKR == 0.0)) begin
      q2 = Ic0a/(Area*IKF);
  end
  else if((IKF == 0.0) && (IKR > 0.0)) begin
      q2 = Ic1ra/(Area*IKR);
  end
  else begin
```

```
    q2 = 0.0;
end

qb = (q1 + sqrt((q1*q1) + 4.0 * q2))/2.0;

Ic0 = Ic0a/qb;
Ic1r= Ic1ra/qb;
Ic1 = (Ic0 - Ic1r);

Ib2 = diode(vbci,XCjc*Jsr,Vgr,nr,Area,Tj,Tnom)/(Br);
Ibx = diode(vbcx,(1.0-XCjc)*Jsr,Vgr,nr,Area,Tj,Tnom)/(Br);

// Base Currents

epsi = 1.0e-6;
Vbclin =  BVceo * pow(1.0 - epsi , 1/Mc);

mm = MM(vbci, BVceo, Mc, Vbclin, Bf, kc);

if(mm >1.0) begin
    if(kBeta > 0.0) begin
        if((Bf - kBeta*Tex) > 1e-6) begin
            EdBeta = (1/(Bf - kBeta*Tex) -
                      kc*(mm - 1)) / (kc*(mm - 1) + 1);
        end
        else begin
            EdBeta = (1e6 - kc*(mm - 1))/(kc*(mm - 1)+1);
        end
    end
    else begin
        EdBeta = (1/(Bf) - kc*(mm - 1))/(kc*(mm - 1)+1);
    end
end
else begin
    if(kBeta > 0.0) begin
        if((Bf - kBeta*Tex) > 1e-6) begin
            EdBeta = (1/(Bf - kBeta*Tex));
        end
        else begin
            EdBeta = (1e6 );
        end
    end
    else begin
        EdBeta = (1/(Bf) );
    end
end

Ib0 = Ic0a * EdBeta;

// no Break-Down
if (BVebo>0) begin
    Ib1 = Ib0 -
        diode((-BVebo - vbei), Jsf, 0.0, 1.0, Area, 0.0, 0.0);
end else
    Ib1 = Ib0;

// Emitter Currents
if((Jse>0.0) && (ne>0))
    Ibdx = diode(vxe,Jse,Vgb,ne,Area,Tj,Tnom);
else
    Ibdx = vxe*1e-12;

if((Jsee>0.0) && (nee>0))
```

```
        Ibdxx = diode(vxxe,Jsee,Vgbb,nee,Area,Tj,Tnom);
    else
        Ibdxx = vxxe*1e-12;

    if((Jsc>0.0) && (nc>0))
        Icdx = diode(vxc,Jsc,Vgc,nc,Area,Tj,Tnom);
    else
        Icdx = vxc * 1e-12;

    // Dissipated Power
    Ipdiss = (Ic1 * (vcei)) + (Ib1 * (vbei)) +
             (Ib2 * vbci) + (Ibx * vbcx);

    if (Ipdiss < 0.0)
        Ipdiss = 0;

    //
    // Nonlinear Part --- Charge Sources
    //

    // qb2med: Base-Collector-Capacitance at medium currents

    I00=(J0*Area);

    // qb2med: Base-Collector-Capacitance at medium currents
    if ((XCjc < 1.0) && (XCjc > 0.0))  begin
        if ((J0<=0.0) || (Ic0<0.0)) begin
            // Qbc independent of current  C = Cjc
            qb2med = XCjc * charge(vbci,(Cjc-Cmin),Vjc,mjc,Area) +
                XCjc * Area * Cmin * vbci;
        end
        else begin
            // C = (1-(2 Ic/I0)/(1+(Ic0/Ia00)^2))*Cjc

                xix = Ic0/I00;

                qb2med = XCjc * (1.0 - tanh( xix )) *
                    (charge(vbci,(Cjc-Cmin),Vjc,mjc,Area) +
                     (1.0-XJ0) * Area * Cmin*vbci) +
                     XJ0 * XCjc * Area * Cmin*vbci;
        end
    end
    else begin
        // if XCjc not within (0,1), sets extrinsic capacitance to zero
        if ((J0<0.0) || (Ic0<0.0)) begin
            // Qbc independent of current  C = Cjc
            qb2med = charge(vbci,(Cjc-Cmin),Vjc,mjc,Area) +
                Area * Cmin*vbci;
        end
        else begin
            // C = (1-(2 Ic/I0)/(1+(Ic0/Ia00)^2))*Cjc

                xix = Ic0/I00;

                qb2med =  (1.0 - tanh( xix  )) *
                    (charge(vbci,(Cjc-Cmin),Vjc,mjc,Area) +
                    (1.0 - XJ0)*Area * Cmin*vbci) +
                    XJ0*Area * Cmin*vbci;

         end
    end

    // qb1: Cex
```

```
if ((XCjc < 1.0) && (XCjc > 0.0)) begin
    qb1 = (1.0-XCjc) * charge(vbcx,(Cjc-Cmin),Vjc,mjc,Area) +
        (1.0-XCjc) * Area * Cmin* vbcx;
end
else begin
    qb1 = 0.0;
end

// qbtr: Tfr*Ic
   qbtr = Tr * Ic1r;
   qbtra = Trx * Ibx;

// qb2: Cbc
qb2 = qb2med + qbtr;

// Base push-out borrowed from HICUM

if ((Jk > 0.0) && (Rci0 > 0.0)) begin
   if (RJk > 0.0) begin
     Vlim   = Jk * Rci0  / (1.0 -  Rci0/RJk);
     InvVpt = (1.0 -  Rci0/RJk)/(Jk*RJk);
   end
   else begin
      Vlim   = Jk * Rci0 / (1.016);
      InvVpt = 0.0;
   end
end

if ((Thcs>0.0) && (Ahc>0.0)  && (Jk>0.0) && (Ic0>0.0)) begin
    RCIO = Rci0/Area;
    AHC  = Area*Ahc;
    if ((Rci0<RJk) || (RJk <= 0.0))
       begin
         Ih   = 1.0 - ICK(vcei, RCIO, Vlim, InvVpt, Vces)/Ic0;
       end
    else
       begin
         Ih   = 1.0 - Vceff(vcei,Vces)/(RCIO*Ic0);
       end
    Wh   = ((Ih + sqrt((Ih*Ih)+AHC)))/(1.0 + sqrt(1.0+AHC));
    xtff = Thcs * Ic0 *(Wh*Wh);
end
else begin
   xtff = 0;
end

// diffusion capacitance
qbtf = (Tf + Tft * Tex) * Ic0;

// total capacitance
qbe = xtff + qbtf + charge(vbei, Cje, Vje, mje,  Area);

//
// Deliver Branch currents
//

// nonlinear part
I(bi, ci) <+ Ibx + ddt(qb1 + qbtra);
I(bii,ci) <+ Ib2 + ddt(qb2);
I(bii,ei) <+ Ib1 + ddt(qbe);
I(ci, ei) <+ Ic1;

I(ex ,ei) <+ Ibdx;
```

```
    I(exx,ei) <+ Ibdxx;
    I(cx ,ci) <+ Icdx;

    // shot noise
    I(bii,ei) <+ white_noise( (TWO_Q *Ib1), "Ib");
    I(ni)     <+ V(ni) + white_noise( (TWO_Q *Ic0), "Ic");
              // collector noise; dummy node to generate correlation
    I(bii,ei) <+ V(ni);
    I(bii,ci) <+ (-absdelay(V(ni),Tf));

    // low-frequency noise
    I(e,  ei) <+ flicker_noise(Kfe* pow(Ib1,Afe) , Ffee,
                               "Hooge␣noise␣of␣emitter␣resistance");
    I(nii)    <+ V(nii) + ddt(V(nii)/(2.0*3.1415*Fb)) +
                 white_noise( Kb*pow(Ib1,Ab));
                 // be-noise; dummy node to generate Lorentz spectrum
    I(bii,ei) <+ V(nii) + flicker_noise(Kfb* pow(Ib1,Afb) , Ffeb,
                            "Flicker␣noise␣base-emitter␣junction␣(a)");

    // linear part
    I(b,  bi) <+ V(b,   bi)/(Rb/N)  +
                 white_noise( (FOUR_K*TjK)/(Rb/N), "thermal") ;
    I(e,  ei) <+ V(e,   ei)/(Re/N)  +
                 white_noise( (FOUR_K*TjK)/(Re/N), "thermal") ;
    I(c,  ci) <+ V(c,   ci)/(Rc/N)  +
                 white_noise( (FOUR_K*TjK)/(Rc/N), "thermal") ;
    I(bii,bi)  <+ V(bii, bi)/(Rb2/N)+
                  white_noise( (FOUR_K*TjK)/(Rb2/N), "thermal");

    if((Jse>0.0) && (ne>0)) begin
       I(ex, bii)  <+ V(ex,  bii)/(Rbxx/N) +
                      white_noise( (FOUR_K*TjK)/(Rbxx/N), "thermal");
    end
    else begin
       I(ex, bii)  <+ V(ex,  bii)*1e-12;
    end

    if((Jsee>0.0) && (nee>0)) begin
       I(exx,bii)  <+ V(exx, bii)/(Rbbxx/N) +
                      white_noise( (FOUR_K*TjK)/(Rbbxx/N), "thermal");
    end
    else begin
       I(exx, bii)  <+ V(exx,  bii)*1e-12;
    end

    if((Jsc>0.0) && (nc>0)) begin
       I(cx, bii)  <+ V(cx,  bii)/(Rcxx/N) +
                      white_noise( (FOUR_K*TjK)/(Rcxx/N), "thermal");
    end
    else begin
       I(cx, bii)  <+ V(cx,  bii)*1e-12;
    end

    I(b)   <+ ddt(Cpb * V(b));
    I(c)   <+ ddt(Cpc * V(c));
    I(b,c) <+ ddt(Cq * V(b,c));

    I(ti) <+ -Ipdiss;
    if (Rth) begin
        I(t,ti) <+ V(t,ti) / Rth;
        I(t,ti) <+ Cth * ddt(V(t,ti));
    end
    else begin
```



```
      I(t,ti) <+ V(t,ti) * 1e12;
  end

end
//
// end of model equations
//

endmodule
```