

This simple example provides virtually all the functionality of the SPICE diode implementation.

```
//
// Copyright 2002 Tiburon Design
// Author: 2002 Marek Mierzwiński
//
// $RCSfile: diode.va,v $
// $Revision: 1.3 $
// $Date: 2002/10/12 04:33:29 $
//
```

```
`include "std.va"
`include "const.va"
```

```
module diode(anode,cathode);
```

```
inout anode, cathode;
electrical anode, cathode, internal;
```

```
parameter real area = 1 from [0:inf];
parameter real is=1e-14 from [0:inf];
parameter real n = 1 from [0:inf];
parameter real cjo=0 from [0:inf];
parameter real vj=1.0 exclude 0;
parameter real fc=0.5 from [0:1];
parameter real tt=0 from [0:inf];
parameter real bv=1e10 from [0:inf];
parameter real rs=0 from [0:inf];
parameter real m=0.5 from [0:inf];
parameter real ibv = 0.001 from [0:inf];
parameter real eg=1.11;
parameter real xti=3;
parameter real tnom = 27;
parameter real af = 1.0;
parameter real kf = 0.0;
```

```
real Vd, Id, Qd, idiode;
real f1, f2, f3, fcp;
real ibv_calc, vth;
real temp, is_t, temp_nom;
```

```
// Use analog function for temperature dependence
```

```
analog function real IS_of_T;
input IS, T, T_NOM, EG, Vth, XTI, N;
real IS, T, T_NOM, EG, Vth, XTI, N;
begin
    IS_of_T = IS * exp((T / T_NOM - 1) * EG / (N * Vth)) * pow(T / T_NOM, XTI / N);
end
endfunction // IS_T
analog
```

Parameter definition and range checking is simple

C-like procedural language

Analog functions simplify coding

begin

```
vth = $vt(); // Get system value of Vthermal

if (ibv != 0)
    ibv_calc = ibv;
else
    ibv_calc = is*bv/vth;
Vd = V(anode, internal); // Get V and I
Id = I(anode, internal);
// Temperature dependence
temp = $temperature(); // Get circuit temperature
temp_nom = tnom + `P_CELSIUS0;
is_t = IS_of_T(is, temp, temp_nom, eg, vth, xti, n);
```

System functions return simulator info

```
// intrinsic diode.
if (Vd < -5*n*vth)
begin
    if (Vd == -bv)
        idiode = -area*ibv_calc;
    else
        if (Vd > -bv)
            idiode = -area*is_t;
        else
            idiode = -area*is_t*(limexp(-(bv + Vd)/vth) - 1 + bv/vth);
end
else
    idiode = area * is_t * (limexp((Vd)/(n * vth)) - 1);

I(anode, internal) <+ idiode + white_noise(2 * `P_Q * Id, "shot")
I(anode, internal) <+ flicker_noise(kf * pow(abs(Id), af), "flicker");
I(internal, cathode) <+ V(internal, cathode)/rs + white_noise(4 * `P_K * temp / rs, "thermal");
```

Currents and noise contribution is easy

```
// capacitance (junction and diffusion).
f1 = (vj/(1 - m))*(1 - pow((1 - fc), m));
f2 = pow((1 - fc), (1 + m));
f3 = 1 - fc*(1 + m);
fcp = fc*vj;
if (Vd <= fcp)
    Qd = tt*Id + area*cjo*vj*(1 - pow((1 - Vd/vj), (1 - m)))/(1 - m);
else
    Qd = tt*Id + area*cjo*(f1 + (1/f2)*(f3*(Vd - fcp) + (0.5*m/vj)*(Vd*Vd - fcp*fcp)));

I(anode, internal) <+ ddt(Qd);
```

Just calculate the charge - no integration, derivatives needed

end

endmodule