Characterisation of Analogue Macromodels under Fault Conditions using a Probabilistic Neural Network

Mark Zwolinski and Cheng Tan
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
mz@ecs.soton.ac.uk, cht197@hotmail.com

Abstract

A technique for parameterising the macromodels of analogue circuit blocks under fault conditions is described. The technique uses a Robust Heteroscedastic Probabilistic Neural Network to classify simulation data. A large reduction in the number of fault classes can be obtained. The classification process is fast and the macromodels generated are accurate.

1. Introduction

Macromodels of analogue circuits have been used to increase the speed of circuit simulation for a number of years [1]. With appropriate characterisation, such macromodels can be extremely accurate, while increasing simulation speed significantly. Recently, research into analogue and mixed-signal testing techniques has required the use of large numbers of simulations of circuits under fault conditions. The computational cost of circuit simulation is such that exhaustive fault simulation is impractical for all but the smallest circuits.

Given that comprehensive fault simulation of analogue circuit blocks is possible, it is still necessary to evaluate the testability (the controllability and observability) of larger circuits containing those blocks. Two possibilities exist (other than simply allowing processor speeds to increase): increasing the speed of the basic circuit and fault simulation algorithms [2]; and using more abstract fault models.

It has been shown [3, 4] that macromodels can accurately represent circuits under fault conditions by varying the macromodel parameters. There are potential problems to do with loading effects if the macromodels are not sufficiently detailed. The major difficulty, however, has been to characterise the macromodels under fault conditions. Two benefits accrue from this: the macromodels simulate faster than the full circuit models and different faults produce similar macroscopic effects, allowing the number of faults modelled to be reduced. Therefore the problem of characterisation is that of matching circuit performance to model parameters aand of grouping responses. Essentially therefore it is a problem of pattern recognition. Previously this work has been done manually, which has proved extremely laborious.

The object of the work reported in this paper has therefore been to automate the characterisation of macromodel parameters. The technique used is that of the *Robust Heteroscedastic Probabilistic Neural Network* [5]. This is described briefly in the next section. In section 3, the overall methodology is described and in section 4, the technique is applied to a number of circuit examples, allowing the characterisation of SpectreHDL macromodels. The method is shown to be fast and accurate. Some proposals for further enhancements are discussed in the final section.

2. Robust Heteroscedastic Probabilistic Neural Network

The principle of the Robust Heteroscedastic Neural Network (RHPNN) has been described elsewhere. Here a brief summary of its principles will be given.

A probabilistic neural network (PNN) classifies data by estimating the probability density functions (pdfs) of the different classes. A PNN requires a training phase, followed by a validation phase before it can be used to classify real data. A PNN consists of a set of Gaussian distribution functions. The training patterns are used as the centres of the Gaussian distribution functions and a common variance or covariance is assumed (this is known as a *homoscedastic* PNN). Conversely, the Gaussian distribution functions of a *heteroscedastic* PNN are uncorrelated and separate variance parameters are assumed. Of course, this heteroscedasticity is a property of such real data as model parameters. This type of PNN is more difficult to train, because of numerical difficulties. A robust method has been proposed to solve these numerical problems, hence the term "Robust Heteroscedastic Probabilistic Neural Network" (RHPNN).

The PNN is a four layer feedforward neural network that realises the Bayes classifier given by (1).

$$g_{Bayes}(\mathbf{x}) = \arg\left(\max_{1 \le j \le K} \{\alpha_j f_j(\mathbf{x})\} \right) \qquad (1)$$

where $\mathbf{x} \in \Re^d$ is a d-dimensional pattern, $g(\mathbf{x})$ is the class index of $\mathbf{x}$, the *a priori* probability of class j ($1 \le j \le K$) is α_j and the conditional probability density function of class

j is f_j. The object of the PNN is to estimate the values of f_j. This is done using a mixture of Gaussian kernel functions.

The training algorithm for homoscedastic PNNs of an expectation process (E) followed by a maximization process (M) (EM algorithm). This algorithm converges to the Maximum Likelihood (ML) estimate. For the heteroscedastic PNN, the EM algorithm frequently fails because of numerical difficulties. These problems have been overcome by using the "Jack-Knife" which is a robust statistical method. Unlike other PNNs, the RHPNN does not need a validation phase.

It has been shown [5] that the RHPNN can be used successfully to differentiate between circuit responses and hence to classify a circuit as faulty or fault-free. Here the problem is a little different. The number of classes of faulty behaviour is not known *a priori*. The data has to be grouped dynamically. This involved some minor modifications to the original RHPNN.

3. Generation of Macromodels

The objective of this work was to generate parameters for existing macromodels, not to derive the macromodels themselves. Therefore, no attempt has been made to include models of "unusual" behaviour. The first step is therefore to write a macromodel. Fig. 1 shows a SpectreHDL model of an opamp in closed loop configuration. SpectreHDL was chosen because of its availability, but another analogue HDL would have been equally suitable.

The parameters of the fault-free macromodel were derived after HSpice simulations of full transistor circuit model. The parameters were obtained manually in this case, although it would be possible to use the RHPNN.

Fault simulations of the full transistor-level model of the circuit were then performed. We used the ANAFINS tools [6] to inject faults and to control the HSpice simulations. In addition we performed full Monte Carlo simulations in order to model the expected parametric variation.

Following the transistor-level simulations, the simulation data is extracted and classified using the RHPNN program. Hence, macromodel parameters are obtained. These parameters are used in invoking the macromodel within a Spectre file, for example:

```
xamp pin nin psu nsu out opamp \
(gain=2.64e-4 rin=4.001e5 vin_offset=0 \
rout=1.054e2 vout_offset=-4.992 \
bk_freq=2.500e7)
```

Notice that these are significantly different to the default parameters shown in Fig. 1. Unspecified parameters take default values.

4. Circuit Examples

A number of circuit examples were used to verify the methodology. Three examples are given here: an OTA configured as an inverting amplifier; a folded-cascode opamp, also configured as an inverting amplifier and a VCO in a PLL. All the simulations were run on a 167 MHz Sun UltraSPARC.

40 distinct faults were inserted into the OTA. To generate the training data, 30 Monte Carlo simulations of the fault-free circuit and 5 Monte Carlo simulations of each of the 40 faulty circuits were performed, giving 230 sets of data. For the testing (i.e. classification) phase, 30 Monte Carlo simulations were run for the fault-free circuit and for each faulty circuit giving 1230 data points. Four measurements were obtained from each simulation run and used for classification: input resistance, output resistance, gain and output offset voltage. The RHPNN, classified the 40 faults and the fault-free circuit into 11 distinct groups. The classification process took about 9 minutes. Fig. 2 compares the DC transfer characteristics, the frequency response and the time domain response for four faults at transistor and macromodel levels. As can be seen, there is relatively close correspondence between the results.

The folded cascode opamp has 17 MOSFETs (compared with 8 for the OTA). The same macromodel was used, as for the OTA. In this case, 84 distinct faults were simulated, together with the fault-free circuit. The same numbers of Monte Carlo simulations were performed, thus 450 simulations were used for training and 2550 for testing. Again four measurements were made. This time, 20 distinct fault groups were found. The RHPNN took around 53 minutes to analyse the data.

The third circuit analysed was a VCO in a PLL. 155 distinct faults were introduced and the same numbers of Monte Carlo simulations were performed, giving 805 sets of training data and 4680 sets of testing data. It may be noted that these fault simulations took around 6 days to complete! Six measurements were used to classify the data in the RHPNN. 30 distinct fault groupings were obtained. The classification took around 10 hours 15 minutes.

This final example illustrates well the kind of speed up in fault simulation that can be obtained, once a circuit block has been characterised. A transient analysis of the full netlist of the PLL takes around 50 minutes, as shown in Table 1. A simulation of the PLL as a set of macromodels takes just under 3 minutes. Simulations of the full transistor model of the PLL with the VCO macromodel and of the PLL macromodel with the VCO modelled at transistor level were also performed to compare the accuracy of simulations. Two things should be noted. First the simulation of the macromodel of the VCO with the full PLL netlist actually takes longer than the full simulation of the entire circuit. Second there is a

difference in the simulation results when expressed in terms of the capture range and lock range. Both these facts are consequences of the macromodels used (rather than of their characterisation). The increase in simulation time is possibly caused by incorrect modelling of the loading effect of the VCO on the rest of the circuit. The variations in the capture and lock range are caused by the model of the low pass filter of the PLL. It should be noted, however, that the centre frequency of the PLL was found to be the same for *all* four versions of the circuit model.

5. Conclusions

One of the hardest problems in generating macromodels of circuits under fault conditions is the parameterisation of the macromodels. It has been shown in this paper that the Robust Heteroscedastic Probabilistic Neural Network (RHPNN) can classify the results of a large number of fault simulations within an acceptable time. The accuracy of the parameterised macromodels is generally good.

The major remaining problem is to generate the appropriate macromodel structure. Under fault conditions, a circuit block may load other parts of the circuit in unusual ways. Such loading needs to be modelled. Therefore either more detailed macromodels are required or the macromodels themselves must be generated dynamically. This problem will form the basis of further research.

Acknowledgements

The authors would like to acknowledge the work of Zheng Rong Yang in developing the RHPNN program, with the support of EPSRC.

References

[1] G.R.Boyle, B.M.Cohn, D.D.Pederson, J.E. Solomon, “Macromodeling of Integrated Circuit Operational Amplifiers” IEEE Journal of Solid-State Circuits, vol. 9, No. 6, pp 353-363, 1974.

[2] Z.R. Yang and M.Zwolinski, “Fast, Robust DC And Transient Fault Simulation For Nonlinear Analogue Circuits”, Proceedings of DATE, Munich, Germany, March 1999.

[3] M. Zwolinski, C. Chalk and B.R. Wilkins “Analogue Fault Modelling and Simulation for Supply Current Monitoring”, ED&TC'96, 1996, 547-552.

[4] A.J. Perkins, M. Zwolinski, C.D. Chalk and B.R. Wilkins, “Fault Modeling And Simulation Using VHDL-AMS”, Analog Integrated Circuits and Signal Processing, 16(2), 1998, 141-155.

[5] Z.R. Yang, M. Zwolinski, C.D. Chalk, “Fault Detection and Classification in Analogue Integrated Circuits using Robust Heteroscedastic Probabilistic Neural Networks”, 4th IEEE International Mixed Signal Testing Workshop, 1998.

[6] I.M. Bell, S.J. Spinks, “Analogue Fault Simulation for the Structural Approach to Analogue and Mixed-Signal Testing",International Mixed Signal Testing Workshop, June 20-22 1995.

Circuit Simulated	Simulation time	Capture Range	Lock Range
PLL netlist	50m 13 s	1.47 MHz	2.76MHz
PLL macromodel	2m 43s	990 KHz	2.58MHz
PLL netlist with VCO macromodel	1hr 8m 49s	900KHz	3.0MHz
PLL macromodel with VCO netlist	26m 53s	1.08MHz	2.26MHz

Table 1 Results of the PLL performance of the 4 simulated combinations

```
module opamp (vin_p, vin_n, vspply_p, vspply_n, vout)
             (gain, rin, vin_offset,rout, vplimit,
              vnlimit, vout_offset, bk_freq)

node [V,I] vin_p, vin_n, vspply_p, vspply_n, vout;
parameter real gain = -3;
parameter real rin = 100e3;
parameter real vin_offset = -3.60e-4;
parameter real rout = 173;
parameter real vplimit = 4.94;
parameter real vnlimit = -4.99;
parameter real vout_offset = 0;
parameter real bk_freq = 4.251e5;
{
  node [V,I] vac_out, vac_in;
  real vin_val,vo,C = 1/(2*PI*rin*bk_freq);
  analog {
    //Input stage
    vin_val = V(vin_p,vin_n) - vin_offset;
    I(vin_p, vin_n) <- (vin_val / rin);

    //Pole1
    V(vac_in) <- vin_val;
    V(vac_out, vac_in) <- rin * I(vac_out, vac_in);
    I(vac_out) <- dot(C*V(vac_out));

    //voltage gain
    vo = V(vac_out) * gain + vout_offset;

    // Output Limiting
    if (vo > V(vspply_p)){
      vo = vplimit;
    }
    else if (vo < V(vspply_n)){
      vo = vnlimit;
    }

    V(vout) <- vo - (-I(vout)*rout);
  }
}
```

Figure 1 SpectreHDL Model of Opamp configured as Inverting Amplifier

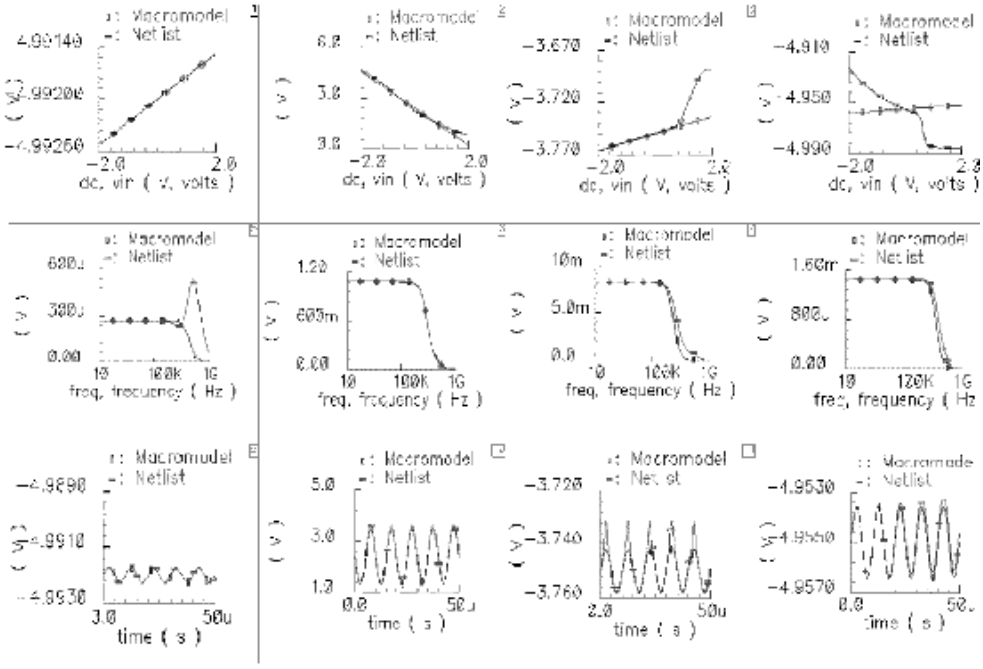

Figure 2 Comparison of Transistor-level and Macromodel Fault Simulations of OTA