Group minimization occurs in the first pass, and minimization of each output function takes place in the second pass. The final table contains sets of vectors corresponding to product terms, which you can easily translate into logic equations. **Listing 1** shows the results obtained for a sample input table comprising 16,384 vectors of 14 inputs and eight outputs. **Listing 2**, a batch file, calls up the executable program heco.com for converting portions of the hex file xmpl.hex, which contains an input table (image of a 16-kbyte memory in hex format). The batch file also calls up heco.com, which converts portions of the hex file into Berkeley format, and glue.com, which reformats the merged results before the second processing pass. **Listings 3** and **4** show the source code for heco.com and glue.com, respectively. You can download the executable and source-code files from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the files from DI-SIG, #2042. (DI #2042) EDN

To Vote For This Design, Circle No. 437

LISTING 4—REFORMATTING MERGED RESULTS

```
/*---------- glue.c -------------*/
/* GLUE - merged Espresso gluer   */
/* ver. 2.2              JCh 1997 */
/*--------------------------------*/

#include  <string.h>
#include  <stdio.h>


void   bin(unsigned int val, int pos)
{
    while (--pos >= 0)
    {
    if ((val>>pos) & 1)
       printf("1");
    else
       printf("0");
    }
    return;
}


int main(int argc, char *argv[])
{
    char  c, *str, *adr, *val, *format, flag = 0;
    unsigned int   inputs = 16, outputs = 8;
    unsigned int   bits = 4, count = 0;

    if (argc > 1)
    {
       argv[1] = strlwr(argv[1]);
       flag = ((strstr(argv[1],"?") != NULL)
       ||  (strstr(argv[1],"/h") != NULL)
       ||  (strstr(argv[1],"-h") != NULL));
    }
    if (argc == 1 || flag)
    {
       printf("\nGLUE: merged espresso -> espresso format converter (C) JCh 1997");
       printf("\nGLUE  inp_cnt out_cnt bit_cnt <input_file >output_file\n");
       return 0;
    }
    sscanf(argv[1],"%2d",&inputs);
    if (argc > 2)
    {
       sscanf(argv[2],"%1d",&outputs);
       if (argc > 3)
          sscanf(argv[3],"%d",&bits);
    }
    printf(".i %d\n.o %d\n",inputs,outputs);
    while ((c = getchar()) != EOF)
    {
       gets(str);
       if (c == '.' && str[0] == 'e')
          count++;
       if (c == '-')
       {
          bin(count,bits);
          puts(&str[bits-1]);
       }
    }
    puts(".e");
    return 0;
}
```

Spice subcircuit models thermistors

LUTZ WANGENHEIM, HOCHSCHULE, BREMEN, GERMANY

The subcircuit in **Figure 1** is a simple and efficient behavioral model for a thermistor, implemented in PSpice (Microsim, Irvine, CA). The model simulates realistic thermistor parameters for all standard analyses (transient, ac, and dc). You can set all relevant thermistor parameters—nominal resistance, nominal and ambient temperatures, and material and thermal constants—during the call of the subcircuit. **Listing 1** gives the parameter definitions for the subcircuit. The model reflects **Equation 1**, traditionally used to describe the resistance-temperature characteristic of thermistors:

$$R_T = R_{NOM} \bullet \exp(B/T_B - B/T_{NOM}), \qquad (1)$$

where R_T is the resistance at thermistor-body temperature T_B, R_{NOM} is the nominal resistance at nominal (standard reference) temperature T_{NOM}, and B is the material constant (in Kelvin) that describes the temperature sensitivity.

Note that **Equation 1** is valid only within a limited temperature range, because it does not reflect the actual temperature dependence of the material constant, B. However, with respect to manufacturing tolerances of B, the **equation** provides a useful approximation. The subcircuit establishes thermal feedback by using a conventional ohmic resistor, representing R_{NOM}, in series with a voltage source E_{TEMP} that represents the temperature-dependent term. The independent zero-voltage source, V_{SENSE}, senses the current, I_T, through the circuit.

The voltage, $V_T = I_T R_T$, generated between thermistor nodes 1 and 2 is the sum of the voltage across R_{NOM} and the output, V_{TEMP}, of the voltage source, E_{TEMP}:

$$I_T \bullet R_T = I_T \bullet R_{NOM} + V_{TEMP}.$$

Replacing R_T by **Equation 1** and solving for V_{TEMP} yields the control function for E_{TEMP}. You can derive the function by applying the analog-behavioral model of PSpice in **Listing 1**:

$$V_{TEMP} = I_T \bullet R_{NOM}[\exp(B/T_B - B/T_{NOM}) - 1].$$

Because all control parameters of the dependent source must be constants, currents, or voltages, you need some extra circuitry to compute the body temperature, T_B, and convert it to a corresponding voltage. Therefore, you use a dependent current source, G_{PWR}, controlled by the voltage-current product, V_TI_T, to generate a voltage across resistor R_{TH} at Node 5. This voltage represents the power-dependent portion of the body temperature, T_B, if you set the value of the thermal resistance, R_{TH}, to the reciprocal of the thermistor's dissipation factor, D. Capacitor C_{TH} in parallel with R_{TH} models the thermal time constant, τ, of the thermistor body.

For ambient temperature-to-voltage conversion, a constant current, I_{AMB}=1A, must produce a voltage at Node 6 that is numerically equal to the ambient temperature, T_{AMB}, in Kelvin. This temperature constitutes the second portion of T_B. To satisfy this equivalency, you use a resistor model for R_{AMB} with a nominal value of 300.15Ω and a linear temperature coefficient of TC=1/300.15=$3.33\times10_3$. During simulation runs, you can set the ambient temperature (for PSpice, nominally 27°C, or 300.15K) with a global .TEMP statement.

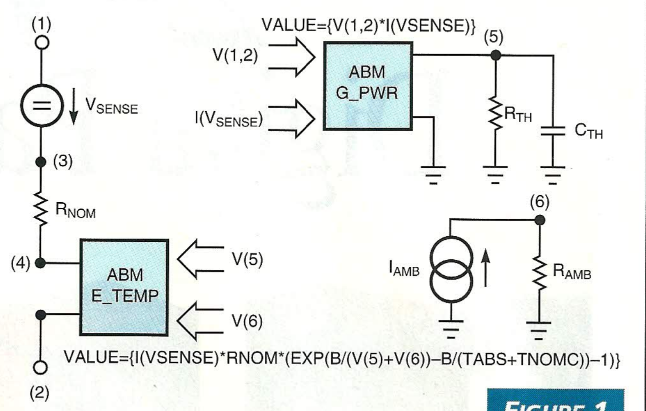

FIGURE 1

This thermistor model incorporates thermal feedback to accurately model current-voltage characteristics for transient analysis as well as for dc and ac simulations.

LISTING 1—SUBCIRCUIT DESCRIPTION FOR THERMISTOR MACRO MODEL

```
*LISTING MACRO MODEL "THERMISTOR"
.SUBCKT THERM 1 2
+ PARAMS: TABS=273.15  TNOMC=25  RNOM=1k  B=3000  D=1E-3  TAU=1
*
*DEFAULT SETTINGS TO BE MODIFIED DURING SUBCIRCUIT CALL
*TNOMC=STANDARD REF. TEMPERATURE IN DEG. CELSIUS
*RNOM=NOMINAL RESISTANCE VALUE AT TNOMC
*B=THERMISTOR CONSTANT In K
*D=THERMISTOR DISSIPATION FACTOR IN W/K
*TAU=THERMAL TIME CONSTANT IN SEC.
*>>> HINT FOR AC ANALYSIS:  SET TAU>100/FMIN <<<
**************************************************
* BASIC THERMISTOR MODEL
*
VSENSE  1 3 DC 0V
RNOM    3 4 {RNOM}
E_TEMP 4 2 VALUE=
+{I(VSENSE)*RNOM*(EXP(B/(V(5)+V(6))-B/(TABS+TNOMC))-1)}
*****************************************************
*POWER-TO-TEMPERATURE-TO-VOLTAGE CONVERSION
*
G_PWR 0 5 VALUE={V(1,2)*I(VSENSE)}
RTH     5 0 {1/D}
CTH     5 0 {TAU*D}
************************************************
*AMBIENT TEMPERATURE-TO-VOLTAGE CONVERSION
*
IAMB  0 6 1A
RAMB 6 0 300.15  TC=3.33E-3
.ENDS THERM
************
```

Note that ac simulation, which is a linear (small-signal) analysis, must not cause any change in the thermistor's electrical resistance. Therefore, the voltage at Node 5, a measure of internal heating, must not contain any ac portion that may result from the nonlinear control operation (current-voltage multiplication) of the behavioral-modeling block, G_{PWR}. Unfortunately, if any dc flows through the thermistor during bias-point calculation (before the ac analysis), the output of G_{PWR} will always contain such an unwanted ac portion. Therefore, to make the ac voltage at Node 5 negligible, you should increase the value of C_{TH} accordingly.

During ac simulations, you should set the time constant, $\tau=R_{TH}C_{TH}$, to a much larger value than $1/f_{MIN}$, the start frequency of the ac analysis. Thus, the current-voltage characteristic of the thermistor becomes a function of only its nominal value, R_{NOM}; the ambient temperature, T_{AMB}; and the dc bias flowing through the device. You can download the PSpice file from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the files from DI-SIG, #2043. (DI #2043) EDN

To Vote For This Design, Circle No. 438

Battery monitor emulates auto dashboard

TG BARNETT AND J MILLAR, UNIVERSITY OF LONDON, UK

When you use battery-powered scientific-recording equipment, you must know the state of the battery, because an unnoticed battery run-down can lead to loss of data. Several types of warning indicators are available, but all have their drawbacks. Moving-coil indicators, test terminals, and digital flags, for example, all are wanting in some respect—they can be inaccurate or unreliable, have an unacceptable battery drain, or have some other operational problem. Ideally,