www.aplac.com

APLAC RF Design Tool

Release Notes & Installation Guide

for Windows and Unix

Version 8.10

Circuit simulation and design tool

APLAC®

APLAC Version 8.10 Installation Guide for Windows and UNIX

APLAC documentation assumes that you have a working knowledge of your operating system and any non-**APLAC** CAD framework you choose to use, as well as related conventions. For additional information, please refer to the documentation that came with your computer system or your CAD framework. Procedures and applications are presented for their instructional value. They have been carefully tested, but are not guaranteed for any specific computer system application. Complete manuals are delivered in PDF format. **Release Notes** supplement each **APLAC** software revision and manual update.

Sales, distribution and support:

APLAC Solutions Corporation
P.O. Box 284
FIN-02600 Espoo
Finland
Tel. +358-9-5404 5000 (GMT +2)
Fax. +358-9-5404 5040

APLAC Solutions, Inc.
320 Decker Drive, Suite 100
Irving, Texas 75062
USA
Tel. +1 (972) 719-2562 (Central Time, GMT-6)
Fax. +1 (972) 719-2568

Email: **sales@aplac.com**
support@aplac.com
publications@aplac.com

For a list of international distributors, please see **http://www.aplac.com**

This manual was typeset April 2005

APLAC includes a rich collection of basic linear and nonlinear models, semiconductor models, and much more. **APLAC** modules Fast RF-IC and RF Board are empowered by application-optimized algorithms. A versatile collection of system-level blocks are available for the simulation and design of analog and digital communication systems, including Micro-Electromechanical Systems. An FDTD-based electromagnetic simulator and radio access modules Bluetooth and WLAN contribute further simulation functionality.

New and customized models can be created, by the user or by the **APLAC** support team, using the C-model Interface or the Artificial Neural Network model generator.

Contents

1. What's New in APLAC 8.10

1.1 Introduction

The APLAC RF Design Tool version 8.10 is now released. The new version includes several improvements and enhancements that will benefit RF design engineers.

New features include

- Memory Consumption Reduced in RFIC Simulations
  - VCCS parallel sampling method improved
  - Reduced memory requirement for many transistor models
  - Reduced internal variable memory usage

- New and Improved Circuit Models
  - PindiodeRC improved
  - HICUML0 model implemented
  - BSIM3SOI model implemented
  - ANN improved

- Philips® SiMKit Model Support
  - Several SiMKit models supported

- DC Convergence Improved
  - New Dog-Leg iteration method implemented
  - DC convergence strategy control improved

- New Discrete Libraries
  - Murata capacitor library
  - Freescale LDMOS library

These and all other improvements are explained in more detail in the following sections.

1.2 New Features in APLAC Simulator

Analysis

- New convergence strategy #6 in DC analysis.
- Sampling of a group of VCCSs in HB analysis requires considerably less memory
- Norm reduction has been implemented in the transient analysis
- Internal update of device currents in DC and TRAN analyses

Functions

- PolyFit-function for polynomial fitting

Graphics

- Dot-type MARKER
- Pasterecolor (Edit menu)

Models

- Public DCOP has been added to IBIS model
- Models BSIM3, MOS9, BJT, and STBJT require less memory in HB analysis
- Improvements in STBJT model
- Internal model specific defaults
- New parameter DIOMOD in BSIM3
- New model level for PinDiodeRC
- Support for Philips SiMKit models
- Periodical saving of the best ANN-model in ANN training
- HICUM Level 0 implemented
- BSIM3SOI implemented

EM Simulation

- FDTD-lumped element cosimulation implemented

Miscellaneous

- Functional Vars that are changed to CONST require less memory
- Parameter IBIS_WARNING_LEVEL added to Prepare
- TeXMode switchable at runtime

1.3 Changes in APLAC Simulator

Analysis

- Default DC strategy after strategy change

- The nonlinear equation solver improved
- Speed-up of ANN training with Gradient optimization
- Further speed-up of ANN training with Gradient optimization
- Speed-up of automatic ANN-model generation

Models

- Faster automatic ANN-model generation in ANNModelGenerator
- BSIM3 VERSION/LEVEL warning removed
- HICUM convergence has improved
- HICUM creates less internal nodes in electrothermal analysis
- Some parameter restrictions removed from VBIC
- The NQS-model of HICUM has been rewritten
- RTH=0 ignores thermal network and all other self-heating parameters
- ANNModelGenerator and ANN training using Gradient
- Added parameter for PinDiodeRC junction parallel resistance

System Simulation

- New parameter TIMESTEP added for WLANOffsetCorrector

EM Simulation

- The MUR1 ABC code partly rewritten

Miscellaneous

- Preprocessor directives can be defined more than once

1.4 Bug Fixes in APLAC Simulator

Analysis

- Loading of old HB guess file failed
- Analyze REDU sometimes caused memory fault
- Analyze REDU + user-defined Gyrators or IdealTransformers
- Analyze REDU caused floating point exception in HP-UX 11
- Internal device currents and DogLeg algorithm
- DC, TRAN, or HB iteration was sometimes terminated prematurely

Functions

- HBIndex gives internal error when used after TRANsient analysis
- Interpol1Dxxxx functions with bad data
- MDLPolyFit did not find solution

Graphics

- Identical PEN assigned automatically
- Pen 0 incorrectly changed to Pen 15

Models

- ModelSet bug fixes
- Initial condition for a dynamic VCCS/VCVS/CCCS/CCVS in case of multiple controlling voltages
- ANNModelGenerator and unspecified optimization method

- ANNModelGenerator and INIT_FILE with zero-valued ANN weight(s)
- MOS9 noise was sometimes incorrect
- Differential IBIS model ignored [Model Selector] data
- Complex CSource gave sometimes incorrect results
- ANNModelGenerator and global activation-function definition
- ModelSet with binning caused sometimes an error
- BSIM4 gave an unnecessary warning
- Some parameters of VBIC had incorrect AREA and temperature scaling
- HICUM caused sometimes memory fault
- Default MOSFET .MODELs were sometimes missing
- Hicum caused sometimes a floating point error
- dynamic VCVS, CCCS, and CCVS and initial condition (UO/IO)
- ANNModelGenerator with NEURONS and AUTO_GENER specified
- BSIM3 noise was sometimes incorrect
- Curr and NoiseSource between the same nodes caused an error
- Sharing a variable for VCCS or NoiseSource caused sometimes an error
- Incorrectly loaded dynamic library caused sometimes a memory fault

EM Simulation

- EM simulation sometimes got stuck
- MUR1 ABC corrected the Ey field erroneously

Miscellaneous

- Disabling statistical variable from optimization caused sometimes a memory fault
- Selecting ”Report|Info|Variables” caused memory fault
- Saving contents of text window
- Functional Var sometimes produces an internal error
- eldo2a gave an error when there was blank space after ’=’
- spi2a/eldo2a/hspice2a did not work with vswitch

2. APLAC Simulator 8.10 Enhancements

2.1 New Features

Analysis

- New convergence strategy #6 in DC analysis.

  New convergence strategy #6 uses the Dog-Leg iteration method for nonlinear DC analysis. It is combined with the source-stepping and model-damping methods as in strategy #1.

  The strategy order is such that strategy #5 (piecewise-linear solution algorithm DC refinement) is called only when all other strategies fail, e.g.:

  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5$

  or

  $3 \rightarrow 4 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 5$.

- Sampling of a group of VCCSs in HB analysis requires considerably less memory

  A method internally known as ”sampling of a group parallel VCCSs inside one nonlinear device” was initially introduced in version 7.51i. This method speeds up the analysis such that it stores during the HB analysis sampling the current and derivatives (or charge and derivatives) of all VCCSs belonging into a special group, and later the data is fetched from the memory when the other VCCSs of this group are sampled. Typically the group is equivalent to all, or most, nonlinear VCCSs inside one nonlinear model, such as BSIM3, or BJT. All temporary data, i.e., current/charges and derivatives, needed during HB sampling are stored into memory simultaneously.

  Version 8.00d improves this ”parallel sampling” - method such that only temporary working data is kept in memory at a time - this causes considerable savings in memory consumption, when the

number of harmonics is large. This new method is available for both HB_MODE=0 and HB_MODE=1, and is supported also with multi-thread HB analysis.

The new method is the new default (can also be selected using command-line argument -newpv), the old method is selectable using command-line argument -oldpv, and both these methods can be disabled, if -npv is specified on the command line.

- Norm reduction has been implemented in the transient analysis

  Norm reduction technique to improve convergence has been implemented in the transient analysis. The method may be invoked by specifying

```
Prepare TRANNORMREDUCTION=1
```

  Other options are 0 (no norm reduction as previously) and 2 (use norm reduction in the case of poor convergence). The default value is 2. The property may also be invoked by command-line argument -tnr=(0|1|2) or -trannormreduction=(0|1|2). The command-line argument overrides the specification in Prepare.

- Internal update of device currents in DC and TRAN analyses

  In the previous versions, after each iteration cycle in the DC and TRANsient analyses, the internal device currents were automatically updated. Now, the device currents are updated only, when the solution is a good candidate for accepting the analysis result. In general, this results in faster computation in DC and TRANsient analyses. If the status of the internal currents are found to be inconsistent, then a warning is issued (this actually means a bug in the iteration code, and users should contact support whenever they see this warning message).

  Users can change the method for updating the internal currents in DC and TRANsient analyses as follows using command line option '-upc' (long form is '-updatecurrents'):

  – upc=0: compute the internal device currents only when the result of that iteration cycle is good enough (this is the new default). A warning message is issued, when the internal device current data is found to be inconsistent

- upc=1: compute the internal device currents at every iteration cycle, this was the behaviour in the previous versions
- upc=2: the same as '-upc=0', but terminate the simulation with an error instead of issuing a warning message

Functions

- PolyFit-function for polynomial fitting

  A new PolyFit has been created similar to the MDLPolyFit function. PolyFit will fit the nth degree polynomial to the given set of x-y data, where n is a user given parameter. Other parameters to PolyFit function are similar to MDLPolyFit (See the manual page of MDLPolyFit). Example:

```
#define SAMPLES 7

Declare VECTOR k REAL SAMPLES
+       VECTOR r REAL 2
+       VAR myx myf

$ data to be modelled (7 samples)
Vector x 2 4 6 10 14 17 18
Vector y 0 4 5 0 -5 -3 0
Vector s 0 0 0 0 0 0 0

$ find a polynomial of degree 3 that fits the data
Call r=PolyFit(x,y,SAMPLES,s,0,1,3,k)

PRINT S "The coefficients of 3rd order polynomial fit" LF
For i 0 r[0]-1
  PRINT S "a_" INT i S " = " REAL k[i] LF
EndFor

Sweep "Polynomial fit" NO_ANALYSIS
+ LOOP 101 VAR myx LIN 0 20
```

```
$ draw the fitting function
  Call myf=k[0]+k[1]*myx+k[2]*pow(myx,2)+k[3]*pow(myx,3)
  Show W 0 XY myx myf NAME="model"

$ draw the specified data points
  If (LoopIndex==0) Then
    Show W 0 VECTOR SAMPLES XY x y NAME="data" MARKER_ONLY=3
  EndIf
EndSweep
```

Graphics

- Dot-type MARKER

  A new dot-type MARKER, number 20, has been added. It is very small and should be useful in cases where the other, larger markers would obscure graphic data.

- Pasterecolor (Edit menu)

  A new Edit menu item has been added. PASTERECOLOR is like PASTE, but an attempt is made to recolor curves pasted with pens different from those already in use.

Models

- Public DCOP has been added to IBIS model

  A new public, DCOP, has been added to IBIS model. It contains the values of the die capacitance(s), package paracitics C_pkg, L_pkg, and R_pkg, and the values of the reference voltage sources.

- Models BSIM3, MOS9, BJT, and STBJT require less memory in HB analysis

  Models BSIM3, MOS9, BJT, and STBJT require less memory in HB analysis. The amount of saved memory is dependent on the number of sampling points in HB analysis.

- Improvements in STBJT model

  Parameter MODEL_LEVEL can be used to select the behavior of STBJT as follows.

  - MODEL_LEVEL=1: specifies the model as it is in versions $\leq$ 8.00b
  - MODEL_LEVEL=2: two BE (BC) static diodes are internally combined into one BE (BC) diode
  - MODEL_LEVEL=3: in addition to MODEL_LEVEL=2, the BE (BC) dynamic diode capacitances are internally combined with nonlinear QDE (QDC) charge sources

  MODEL_LEVEL=3 is the new default, as it results in smaller memory requirements and faster computation than MODEL_LEVEL=1.

  A test with a real simulation example (2200 nodes, about 200 BJTs and MOS9s, HB TONE2=3 MULTIDIM) using new memory saving improvements introduced in versions 8.00b and 8.00c resulted in 11% and 30% savings in memory consumption, and function calls, respectively, when compared with version 8.00a.

- Internal model specific defaults

  Memory requirement has been reduced such that all instances of a model use the same global default variables instead of private copy of a default variable.

- New parameter DIOMOD in BSIM3

  Parameter DIOMOD can be specified to BSIM3. DIOMOD can have two values, 0 and 1 (default). When DIOMOD is set to 0, BSIM3 type of internal Source-Bulk and Drain-Bulk diodes will not be created (neither current nor charge). DIOMOD=0 has no effect if any of the common mosfet parameters (ALEV, DIOLEV, RLEV, DCAPLEV, TLEVI, TLEVR, TLEVC, TLEV, or ACM) have been defined as in this case the diodes will not be those of the original BSIM3 model.

  Default setting DIOMOD=1 has no effect to the model operation.

- New model level for PinDiodeRC

  New MODEL_LEVEL=2 has been added for PinDiodeRC. This level brings frequency and bias dependent junction capacitance to this model. New parameters WD for depletion area width, EPS

for diode permittivity and RHO for i-region resistivity have been added for this level.

- Support for Philips SiMKit models

  SiMKit is a library of nonlinear semiconductor device models developed and maintained by Philips. The source code, and the documentation of the device models can be found at http://www.semiconductors.philips.com/Philips_Models.

  SiMKit models can be used, if either ”RFIC” or ”simkit” licensing option is available. Version 8.10a supports the following Philips SiMKit models:

– simkit_mos3100	MOS31
– simkit_mos3100et	MOS31 (electrothermal)
– simkit_mos1100elect	MOS11, electrical model
– simkit_mos1100geom	MOS11, geometrical model
– simkit_mos1101elect	MOS1101, electrical model
– simkit_mos1101electet	MOS1101, electrical model (electrothermal)
– simkit_mos1101geom	MOS1101, geometrical model
– simkit_mos1101geomet	MOS1101, geometrical model (electrothermal)
– simkit_mos1101geombinn	MOS1101, geometrical (binning) model
– simkit_mos1101geombinnet	MOS1101, geometrical (binning) model (electrothermal)
– simkit_mos1102elect	MOS1102, electrical model
– simkit_mos1102electet	MOS1102, electrical model (electrothermal)
– simkit_mos1102geom	MOS1102, geometrical model
– simkit_mos1102geomet	MOS1102, geometrical model (electrothermal)

– simkit_mos1102geombinn MOS1102, geometrical (binning) model
– simkit_mos1102geombinnet MOS1102, geometrical (binning) model (electrothermal)

These models are based on SiMKit version 1.3. General issues:

- the models accept only parameters listed in the original documentation available from the Philips (www.semiconductors.philips.com/Philips_Models).
- MOS models have no support for parasitic resistors, diodes etc.
- reference temperature should be specified using model-dependent parameter, usually named as TR or TREF. APLACs TNOM/TNOMC does not change the nominal temperature of the SiMKit model

- Periodical saving of the best ANN-model in ANN training

  During ANN training, ANNModelGenerator saves the best ANN-model file (”params.ann”) obtained so far at every 100th optimization cycle. If VALID_FILE (ANN validation file) has been specified, the saving of the best ANN-model file is done at every 10th cycle; this way a ”back-to-history kind of” cross validation can be performed to detect overlearning (a situation where the training error, Etr, still, decreases, while the validation error, Eva, starts to increase).

  In both cases, without or with VALID_FILE, when ANN training terminates normally, a relevant message is printed on the screen if the best ANN-model file (with the lowest Etr or Max(Etr,Eva), respectively) was obtained before the last optimization cycle.

- HICUM Level 0 implemented

  A new model, HICUML0, has been implemented. HICUML0 is a simplified version of the HICUM BJT model. When compared to HICUM, the main differences are the lack of the BE tunneling current source, the NQS model, the substrate transistor and the substrate coupling network.

- BSIM3SOI implemented

  BSIM3SOI model implemented. Available versions are 3.1, 3.11, and 3.2 (default).

EM Simulation

- FDTD-lumped element cosimulation implemented

  Lumped element-FDTD cosimulation has been implemented. The lumped-element circuit must be specified before the ElectroMagnetics block. The interface between the circuit simulator and the FDTD simulator is EMConnector, specified within the ElectroMagnetics block. Cosimulation currently works only with a fixed time step, i.e., FIXED must be specified in Sweep.

  The following netlist example shows how to run a cosimulation:

```
Volt Eg 1 0 TRAN=sin(2*PI*freq*(t)) R=50
Res R1 2 0 50

ElectroMagnetics RES_TEST
+ HARMONIC freq
+ MARGIN 99.9%
+ DIV 64 50 12
+ CELLSIZE 0.4064mm 0.4233mm 0.265mm
+ LOOPS 1500
+ MUR1
+ ABCTRIM X 1.8697 1.8697
+ ABCTRIM Y 1.8697 1.8697

  EMConnector le_volt 0 1
  + SPAN 32 10 0 32 10 3
  + DIR Z

  Slab substrate1
  + SPAN 0 0 0 64 50 3
  + ER 2.2

  EMConnector le_res 0 2
  + SPAN 32 25 0 32 25 3
  + DIR Z

  Patch ground_plane
  + SPAN 0 0 0 64 50 0
```

```
  Patch y_directed_strip
  + SPAN 29 10 3 35 40 3
EndElectroMagnetics

Call TT = EMSimTime("RES_TEST")
Call NL = EMSimLoops("RES_TEST")

Var dt TT/NL

Sweep "Vin and Vres in time domain"
+ FIXED
+ LOOP NL TIME LIN 0 TT

  Show Y  Vem("RES_TEST", 32, 10, 0, 32, 10, 3) NAME="Vg"
  Show Y2 Vem("RES_TEST", 32, 25, 0, 32, 25, 3) NAME="Vr"
  Show Y2 Vtran(2)

EndSweep
```

The output voltage at node ”2” is the same as the voltage calculated across points (32,25,3) and (32,25,0) using the FDTD output function Vem.

Miscellaneous

- Functional Vars that are changed to CONST require less memory

  Functional variables, that are automatically changed to CONSTant, require less memory when compared to the situation where the functionality is preserved.

- Parameter IBIS_WARNING_LEVEL added to Prepare

  Parameter IBIS_WARNING_LEVEL, which controls the output of the warnings generated by the IBIS parser, has been added to Prepare. Possible values are:

  0: The warnings are suppressed 1: The warnings are printed to the text output (default setting) 2: The warnings are printed into a

file 'ibis_file.log', where ibis_file is the name of the parsed IBIS file without the .ibs extension

- TeXMode switchable at runtime

  A checkbox was added to the Edit|Title/axes dialog to allow switching TeXMode on/off at runtime.

2.2 Changes

Analysis

- Default DC strategy after strategy change

  If, during a DC sweep, APLAC is forced to change the convergence strategy, say from #1 to #4, then strategy #4 is used as a first strategy to the next sweep point. However, if strategy #5 (PWL DC analysis and refinement) has found the convergence, then APLAC continues with the original default or user-defined strategy. Previously in such a case DC sweep continued with PWL DC analysis.

- The nonlinear equation solver improved

  The nonlinear equation solver has been improved. It uses now less norm reduction steps and causes less problems with convergence.

- Speed-up of ANN training with Gradient optimization

  The well-known Error Back Propagation (EBP) algorithm has been implemented to speed up ANN training using the (conjugate) Gradient optimization method. In short, the EBP algorithm replaces both the slow numerical gradient evaluations and the fast gradient updates (with Broyden's updating formula) by very fast analytical evaluations. In ANN training using Gradient, the EBP algorithm is automatically invoked. Furthermore, the parameter DERIV has no more meaning.

  The following table demonstrates the speed-up obtained. In the table, Nw is the number of ANN weights (optimization variables),

Ntr that of training-set samples (resulting in Ntr*OUTPUTS optimization goals), and Etr is the value of MAX_ERR that was used to stop ANN training. In the table, each number of optimization cycles, CPU/s, and speed-up is the average of 10 successive runs (with weights randomly initialized).

NEURONS	Nw	Ntr	Etr/%	OLD cycles CPU/s	NEW cycles CPU/s	speed-up
[3,5,2]	33	306	1.0	589 2.76	163 0.62	4.45
			0.5	1526 6.97	392 1.43	4.87
[3,10,2]	62	50	1.0	1687 1.75	459 0.39	4.49
			0.5	8353 8.18	1327 1.08	7.57
[5,10,5]	115	486	1.0	1506 62.21	784 8.77	7.09
			0.5	- -	- -	-

- Further speed-up of ANN training with Gradient optimization

  About 1.5X speed-up of ANN training has been obtained by reorganizing and fine tuning the internal ANNModelGenerator ↔ Gradient C-code interface.

  Below, an attempt has been made to demonstrate the total improvement of ANN training, which results from the speed and convergence improvements in APLAC versions 8.00d, 8.00e, and 8.10a. First, consider the following three ANN-modeling problems:

ANN	Ni	Nh	No	Ntr	Nw	Ng
1	3	10	2	50	62	100
2	3	5	2	306	33	612
3	5	10	5	486	115	2430

  where

  ANN = ANN-modeling problem
  Ni = # of ANN inputs
  Nh = # of hidden-layer neurons
  No = # of ANN outputs
  Ntr = # of training-set samples
  Nw = # of ANN weights (opt. vars)
  Ng = # of optimization goals

  For each problem, the following values of ANN training error (Etr)

were used (by setting MAX_ERR) to stop ANN training: 4.0, 3.0, 2.0, 1.0, 0.5 %. The table below demonstrates the speed-up obtained. In the table, each ANN test error (Ete), the number of optimization cycles, CPU/s, and speed-up is the average of 10 successive runs (with weights randomly initialized). In the table, ”8.00c” and ”8.10a” mean APLAC versions 8.00c and 8.10a, while ”-” and ”oo” mean ”Etr not reached” and ”infinite”, respectively.

ANN	Etr/%	Ete/% 8.00c 8.10a	cycles 8.00c 8.10a	CPU/s 8.00c 8.10a	speed-up
1	4.0	2.643 2.600	144 40	0.23 0.04	6
	3.0	1.958 2.040	294 57	0.43 0.05	9
	2.0	1.704 1.524	14127 164	16.81 0.09	187
	1.0	- 0.946	- 577	- 0.27	oo
	0.5	- 0.794	- 3280	- 1.45	oo
2	4.0	3.839 3.907	245 26	1.41 0.08	18
	3.0	2.868 2.879	981 65	5.43 0.15	36
	2.0	1.929 1.902	1576 87	8.38 0.19	44
	1.0	0.979 0.984	24535 217	125.31 0.43	291
	0.5	- 0.485	- 634	- 1.19	oo
3	4.0	2.436 2.552	305 42	16.85 0.30	56
	3.0	1.892 1.730	5404 56	304.80 0.39	782
	2.0	- 1.269	- 122	- 0.81	oo
	1.0	- 0.766	- 504	- 3.15	oo
	0.5	- -	- -	- -	-

Based on the table above, we can safely say that a 1...2 order(s) of magnitude speed-up of ANN training has been obtained between APLAC versions 8.00c and 8.10a.

- Speed-up of automatic ANN-model generation

  The automatic ANN-model generation algorithm of ANNModelGenerator (invoked by parameter AUTO_GENER) has been speeded up by adopting the accelerated Gradient-based ANN training and, especially, by changing the number and calling order of optimization methods tried.

Models

- Faster automatic ANN-model generation in ANNModelGenerator

The automatic ANN-model generation (specified by parameter AUTO_GENER) has been speeded up. The speed-up is based on a more intelligent control of internal operations. Practically speaking, the message ”Reinitializing ANN weights” should not appear on the screen as frequently as before.

- BSIM3 VERSION/LEVEL warning removed

  Since version 7.92e, BSIM3 produced a warning if VERSION and LEVEL were both specified. This warning has been removed.

- HICUM convergence has improved

  HICUM convergence has improved in transient analysis

- HICUM creates less internal nodes in electrothermal analysis

  HICUM creates less internal nodes in electrothermal analysis, i.e., when self-heating is activated.

- Some parameter restrictions removed from VBIC

  The following error messages, which resulted in termination of the simulation, have been removed from VBIC:

  ”NEN must be greater than NEI” ”NCN must be greater than NCI” ”NCNP must be greater than NCIP”

- The NQS-model of HICUM has been rewritten

  The NQS-model of HICUM has been rewritten such that it creates less internal nodes in AC and HB analyses.

- RTH=0 ignores thermal network and all other self-heating parameters

  If RTH=0 is specified to a self-heating model then the thermal network is not created and all other thermal parameters specified will be ignored. Previously RTH=0 created a thermal network and used RTH=RMin.

- ANNModelGenerator and ANN training using Gradient

  The range used for random initialization of ANN weights has been changed from [-1,1] to [-0.5,0.5]. Also, the internal optimization-variable windowing method, which is used for ANN training using Gradient, has been changed from ”ContinuousWindowing” to ”NoWindowing”. These changes resulted, using OPT_CYCLES=1000

	OLD	NEW
problem	Etr/% Ete/%	Etr/% Ete/%
1	3.386 2.163	1.864 1.314
2	2.540 1.795	1.413 1.244
3	2.076 2.008	0.770 0.757

with three relevant ANN-training problems, in the following improvements:

In the table, each training error (Etr) and test error (Ete) shown is the average of 10 successive runs.

- Added parameter for PinDiodeRC junction parallel resistance

  New parameter RP for the parallel resistance of PinDiodeRC's junction capacitance has been added. Previously this value was constant.

System Simulation

- New parameter TIMESTEP added for WLANOffsetCorrector

  The input vector sampling time step of DT-system block WLANOffsetCorrector can now be changed via optional parameter TIMESTEP. Default value is 50ns.

EM Simulation

- The MUR1 ABC code partly rewritten

  The code for the MUR1 type ABC is partly written using macros that automatically decide whether to use vectorized data structure and looping or not.

Miscellaneous

- Preprocessor directives can be defined more than once

  If a preprocessor directive is defined twice as in the netlist below

```
#define xyz       1.25
print TEXT "xyz = " REAL xyz LF
#define xyz      -20.8
print TEXT "xyz = " REAL xyz LF
```

a warning is issued:

```
APLAC 8.10 WARNING: xyz already defined at line 1, using
new definition starting from line 3
```

The latest definition is always used - this behavior is the same as with C-compilers. Previous versions gave an error, when a preprocessor directive was defined more than once.

2.3 Bug Fixes

Analysis

- Loading of old HB guess file failed

  An internal change of components DCBlock/DCFeed, introduced in version 7.92d, made useless the HB guess files which were created with any version $\leq$ 7.92c.

  The guess file loading code has been changed such, that even if the number of nodes in the circuit is the same as in the guess file, the node names are matched between the circuit and the guess file. This may sometimes cause a minor slowdown when a large file is loaded, but also makes the guess file insensitive for future internal changes.

- Analyze REDU sometimes caused memory fault

  Analyze REDU caused memory fault if the file in which the poles and residues were tried to be stored could not be opened for writing. This could have resulted from the user specifying a non-

existing path or the working directory or an existing file with the same name being write protected.

- Analyze REDU + user-defined Gyrators or IdealTransformers

  Analyze REDU sometimes produced incorrect results, if the circuit contained user-defined Gyrators (with G specified) or IdealTransformers. In some cases, when the previously mentioned components were used, the simulation terminated in a strange error message: ”ERROR: NaN or Inf prevents convergence.”

- Analyze REDU caused floating point exception in HP-UX 11

  Analyze REDU caused floating point exception in HP-UX11.

- Internal device currents and DogLeg algorithm

  If the DC iteration converged with the DogLeg method, then the internal device DC currents were not automatically updated.

- DC, TRAN, or HB iteration was sometimes terminated prematurely

  DC, TRAN, or HB iteration was sometimes terminated before the convergence had been reached. This happened if the change in the controlling voltages of nonlinear VCCSes was below RELERR, even if the change in all voltages was not.

Functions

- HBIndex gives internal error when used after TRANsient analysis

  Function HBIndex caused internal error if called after TRANsient analysis.

- Interpol1Dxxxx functions with bad data

  Interpol1Dxxxx with several data points having the same x coordinate resulted in a floating-point error. Now APLAC either issues a warning and sets the interpolation values and derivatives to zero or results in an error message.

- MDLPolyFit did not find solution

  If the x-axis values for MDLPolyFit fitting tool where larger than one, the function could not find the correct solution. Now the x-axis values are internally scaled so that the solution will be found more precisely.

Graphics

- Identical PEN assigned automatically

  If more than 16 curves were displayed in one sweep step and PENs were assigned automatically, curves numbered above 16 were assigned the same PEN.

- Pen 0 incorrectly changed to Pen 15

  If a pen number of 0 was assigned to a curve in the input file, the number was incorrectly changed to 15.

Models

- ModelSet bug fixes

  The following problems with ModelSet have been fixed:

  – The Models passed to ModelSet can now have a different number of parameters. This property can be used, for example, to initialize all Models with some default values and override the parameters within the same Model statement as needed. Note, that all Models must specify all parameters at least once.

  – ModelSet with BINNING property could not be used inside DefModel.

  – The memory requirements of ModelSet has been reduced in the case, when the binning and corner variables are constant, i.e., the ModelSet actually uses one fixed Model.

- Initial condition for a dynamic VCCS/VCVS/CCCS/CCVS in case of multiple controlling voltages

  An attempt to specify the initial condition using parameter UO (or IO) for a dynamic VCCS/VCVS/CCCS/CCVS resulted always in an error, when there were more than one controlling voltage (or branch). The syntax of UO (and IO) has been changed to

```
VCCS ... UO=initialcondition
```

where initialconditional is an n-dimensional Var and n is the number of controlling voltages (or branches). Note, that the new syntax above is compatible with old versions, when there is only one controlling voltage (or branch). For example,

```
VCCS v1 1 0 2 2 0 3 0 [ cv(0)^3+cv(1), 3*cv(0)^2, 1 ]
+ C NONLINEAR DERIV
+ UO=[2,5]
```

defines a dynamic VCCS with two controlling voltages having initial conditions 2V and 5V, respectively.

- ANNModelGenerator and unspecified optimization method

  The internal default optimization-method specification of ANNModelGenerator, being equal to

```
OptimMethod Gradient OPT_CYCLES=10000 OPT_XTOL=0.0
+ OPT_FTOL=0.0
```

  did not work; if ”OptimMethod ...” was not specified, the following odd error message was shown on the screen:

```
APLAC 8.00 ERROR: STACK2 is invalid argument
for BuildStack
```

- ANNModelGenerator and INIT_FILE with zero-valued ANN weight(s)

  If INIT_FILE was specified and one or more ANN weights in the file were equal to 0.0, an error occurred.

- MOS9 noise was sometimes incorrect

  MOS9 noise was incorrect if the model parameters were constant, model level 903 was used, and the noise equation selection parameter NFMOD=1.

  This bug exists only in version 8.00a.

- Differential IBIS model ignored [Model Selector] data

  If the model for the inverting pin of a differential IBIS model corresponded to a [Model Selector] entry, the model was not found and the simulation terminated in an error.

- Complex CSource gave sometimes incorrect results

  If both the value and the scaling factor of the CSource were complex constants, then the CSource gave incorrect results.

- ANNModelGenerator and global activation-function definition

  If NEURONS was specified to an ANN with more than three layers, the global hidden-layer activation-function definition (e.g., "MLP_ATAN = 0 0.9") only worked for the first hidden layer (that is, for the second ANN layer). For example, the definition

```
ANNModelGenerator "JfetModGen"
+ ...
+ NEURONS  = [2,6,4,3]
+ MLP_ATAN = 0 0.9
```

  resulted in the following ANN structure

layer	neurons	act. func.	slope param.
1	2	NO_ACT	0.000
2	6	MLP_ATAN	0.900
3	4	MLP_TANH	0.500
4	3	LINEAR	1.000

  while the "act. func." and "slope param." for ANN layer 3 (with 4 neurons) should be "MLP_ATAN" and "0.900", respectively.

- ModelSet with binning caused sometimes an error

  If binning limits did not obey a regular rectangular grid, an error occurred. One such situation was with definition like

```
Model M1 WMIN a WMAX b ...
Model M2 WMIN b WMAX c ...
Model M3 WMIN a WMAX c ...
```

```
ModelSet Mx MODEL=M1 ...
```

- BSIM4 gave an unnecessary warning

  BSIM4 gave occasionally a warning about setting source and drain conductances to 1.0E3 mho even though the resistors were not included in the equivalent circuit.

- Some parameters of VBIC had incorrect AREA and temperature scaling

  VBIC parameters IKR, IKP, and ITF were not scaled by AREA. Temperature scaling of parameters PC, PE, and PS was incorrect.

- HICUM caused sometimes memory fault

  HICUM caused memory fault in HB analysis if LSSS and NOISE were specified in Prepare and the NQS parameter ALIT was specified to HICUM.

- Default MOSFET .MODELs were sometimes missing

  The definitions of automatically added default MOSFET .MODELs for levels 1-3 (named as MosfetLevelOne,...) were sometimes missing from the converted netlist.

- Hicum caused sometimes a floating point error

  Hicum caused sometimes a floating point error in HB analysis.

- dynamic VCVS, CCCS, and CCVS and initial condition (UO/IO)

  Dynamic VCVS, CCCS, and CCVS caused an error, when there were more than one controlling voltage (or current), and the initial condition was specified using UO (IO).

- ANNModelGenerator with NEURONS and AUTO_GENER specified

  If automatic ANN-model generation (”AUTO_GENER”) and more than three ANN layers (e.g., four layers by ”NEURONS [3,8,6,2]”) were specified, automatic ANN-model generation was carried out with this large number of ANN layers, instead of the desired ANN structure with three layers (and internally varied number of hidden-layer neurons).

- BSIM3 noise was sometimes incorrect

BSIM3 noise was calculated erronously if device multiplier (parameter M) was specified and it was not equal to 1.

- Curr and NoiseSource between the same nodes caused an error

  Current source Curr, and noise source defined using Curr, Volt, or NoiseSource caused an error, if the external nodes n1 and n2 were same. Now a warning is issued and the component is automatically eliminated from the circuit.

- Sharing a variable for VCCS or NoiseSource caused sometimes an error

  In the following netlist

```
Var x 3 const
vccs v1 1 2 1 1 2 x                 $ G=x=3 (linear)
vccs v2 3 0 1 3 0 x g nonlinear     $ J=x=3 (nonlinear)
vccs v3 1 2 1 1 2 x                 $ J=x=3 (bug: this
                                    $ was nonlinear also)
```

  VCCS v2 is a nonlinear element because of keyword 'nonlinear'. The simulator internally marks variable 'x' ('x' computes the value of v2) to be a voltage-dependent variable. This causes VCCS v3 to be created as a nonlinear VCCS because 'x' now depends on voltages. The code has been fixed such that in certain cases variable 'x' is not marked to be voltage dependent, and VCCS v3 is created as a linear VCCS.

- Incorrectly loaded dynamic library caused sometimes a memory fault

  If a dynamic library was loaded using command #LOAD such that the type was incorrect (i.e., TYPE=FUNCTION when loading a model library, or TYPE=MODEL when loading a function library), then a memory fault occurred sometimes. The C-interface API file, aplacexternapi.h, has been changed to check, if the type of the loaded library is incorrect. This change has no effect on the binary compatibility between old and new versions, but the checking code is included in the library only if compiled with the newest version of aplacexternapi.h. Also only simulator versions $\geq$ 8.10a include this check.

EM Simulation

- EM simulation sometimes got stuck

  EM simulations got ”stuck” sometimes when there were more than one EM output functions used in conjunction with one or more Show statements within a Sweep statement. This happened due to the following reason.

  The internal function controlling an EM simulation was invoked from the EM output functions. In some situations, when invoked more than once at the same time point, the internal controlling function would decide (for numerical reasons) to end the ongoing simulation, reset all field values to zero, and restart the simulation. This gave the impression that the simulation was ”stuck”.

  The simulation is no longer triggerred from the EM output functions. It now starts (and continues to the end if no output function is specified in the Sweep statement) at the end of the (dummy) transient analysis performed at each time point.

- MUR1 ABC corrected the Ey field erroneously

  The Ey field correction made by the MUR1 ABC was wrong at planes x=0 and x=xmax. The for looping was done upto ymax instead of ymax-1 when correcting for Ey.

Miscellaneous

- Disabling statistical variable from optimization caused sometimes a memory fault

  If statistical variable x was disabled from optimization using command

```
SetVar x OPT=0
```

  a memory fault occurred, if the statistical variable had never been included in the optimization.

- Selecting ”Report|Info|Variables” caused memory fault

  Selecting ”Report|Info|Variables” in a graphics window caused memory fault in version 8.00c.

- Saving contents of text window

  If text windows were used and an attempt was made to save such a window’s contents in an existing file, nothing happened on HP-UX, while there was a segfault on Linux.

- Functional Var sometimes produces an internal error

  The following file caused an internal error:

```
*function func1
*+ 3.0

Var Var1 FUNCN func1 1 TIME_DEPENDENT
```

  If TIME_DEPENDENT was missing, then the error message was correct saying

```
APLAC 8.00 ERROR: Function "func1" not found,
                  if this is an expression,
                  please add parentheses around it
```

- eldo2a gave an error when there was blank space after ’=’

  spi2a/eldo2a/hspice2a gave an error for the following netlist:

```
D1 n1 n2 dxx area= ’3*101/1k’ tc=0
.model dxx d(cjo=2p is=1n)
```

  The error was caused by a space character after device parameter ”area=”. Fixed in spi2a/eldo2a/hspice2a version 1.47.

- spi2a/eldo2a/hspice2a did not work with vswitch

  Voltage-controlled switch was not translated correctly (”MODEL=” was not added before model statement name). spi2a/eldo2a/

hspice2a has been changed such that LEVEL=1 or LEVEL=2 is added automatically to the translated Switch statement, because the default Switch (without LEVEL) is not compatible with the spice/eldo switch.

3. The APLAC Software Package

*The **APLAC** Installation CD-ROM contains installation packages for the **APLAC Simulator** and **APLAC Editor**, for Windows and UNIX platforms. The installation packages also include simulation examples, installation scripts, utility programs and on-line manuals.*

*This Installation Guide gives detailed instructions for **APLAC** installations on Windows and UNIX systems, with illustrated examples.*

The **APLAC** Installation CD contains installation media for all supported licensing methods and platforms. **APLAC** user guides are provided along with Acrobat Reader software for each platform.

3.1 APLAC License Types

Floating license - **APLAC** Solutions Corporation uses the popular FLEXlm license management framework developed by GLOBEtrotter Software Inc. (http://www.globetrotter.com). The FLEXlm License Server manages the usage of **APLAC Simulator** and **APLAC Editor** licenses. License Server software is used to administer licenses centrally for multiple clients, concurrently, with valid license files. Clients are permanently connected to the server and have access to the license file, which contains host identification and is read by the license server. A new license file can be combined with existing license files.

Typically a stable network node such as UNIX workstation or NT Server are chosen for license management. FLEXlm works transparently on both UNIX and Windows platforms and the components of the framework are essentially the same for both platforms.

We recommend that you nominate a license administrator. Software licenses are usually in high demand, and technical support can be useful at the site to resolve error conditions and disputes over license usage. **APLAC** licensing makes this as effortless as possible.

Portable license - Only supported in Windows machines, the portable license is tied to a Hardlock key that is connected to the computer's parallel port. This commercial portable license and Hardlock can be moved from one computer to another. No network connection is needed.

3.2 License File and HostID

You need to have access to a valid license file or a FLEXlm License Server with valid **APLAC** licenses before using **APLAC**, because the full-featured **APLAC** version will not run without a license.

A machine-dependent license file can be ordered from **APLAC** Solutions Corporation, using the hostID of your computer. You can order

the license file using the following license application form:

Commercial License Order Form - com/docs/licorder.txt
(send to *licenses@aplac.com* for Portable and Floating Licenses)

You can get the internal hostIDs of your computer by using following commands:

Platform	Type this command	Example HostID

Windows NT/2000/XP
Physical address of ethernet adapter
Pentium III+ hostID
	ipconfig /all or
	lmhostid or
	lmhostid -CPU
		`01-20-5a-b8-6f-b7`
		`01205ab86fb7`
		`95D2-1D3D`

HP-UX 32-bit hostID	**uname -i** or **lmhostid**	`2005771344` `778DA450`
Linux HW address of network card	**/sbin/ifconfig eth0** or **lmhostid**	`00:40:53:34:e4:35` `00405334e435`
Sun 32-bit hostid	**hostid** or **lmhostid**	`170a3472` `170a3472`

License data will be sent to you by email. Copy the license data from the email message to a file and save the file as license.dat in **ascii** (text only) format.

With the portable license, the hostID appears in the hardlock key.

NOTE: If you need to transfer your license service to a new computer, contact **support@aplac.com**. New license creation and handling may require a written certificate and handling fee.

4. Installing APLAC in Windows

APLAC can be installed in Microsoft operating systems Windows 2000, NT 4.0 and Windows XP, using all supported forms of licensing. Network based FLEXlm licensing requires that the computer running the License Server must have an ethernet card or adapter installed for network access.

4.1 Floating License Installation

When installing the centrally administered Floating License to a Windows computer, you need to be logged into the machine as an administrator, and you must have access to the license file, and know its **name** and **location** before installing or know a server and the port that is used to offer licenses at your site.

The centrally administered licenses are typically provided by a Windows or UNIX server, permanently accessible through your network. Your system administrator can give you more details.

For best results, exit all other applications before running the installation.

APLAC Client Software

1. Start the **APLAC** Installation Launcher and select the **APLAC Full Version** installation, or start the installation from **com\windows\aplacflx\setup.exe**.

   You will see the standard Welcome and License Information dialogs. Press **Next** and **Yes** to proceed.

2. Select the directory where **APLAC** is to be installed *(Default:* ***C:\Program Files\APLAC\APLAC*** *<**version + type**>)* and press **Next**.

3. Enter the name of the license file *(Default:* ***license.dat***).

   You can define the 'license file' as **port@host**, where **port** is the number of the TCP/IP port at which the license server runs, and **host** is the DNS name of the computer, such as **27000@licserver**

   NOTE: It is possible to store the license file itself at the client, using only the server name and port number, as shown in the first lines of the license file. This approach is not recommended.

4. Select the directory in which the license file is stored, and press **Next**.

   This directory can be in a local drive, or in a network volume.

5. Confirm the name of the license file. Make sure that the filename ends in .dat, such as **license.dat**, not **license.dat.txt**.

6. The installation proceeds with files copied and configured. You may be asked to rename old INI files for backup purposes, and you will be given an opportunity to create associations for .i (**APLAC Simulator**) and .n (**APLAC Editor**) files.

   Persistent setup-related data is stored in **APLAC32.INI** (**APLAC Simulator**), **APLACED.INI** (**APLAC Editor**) and **APLACSRV.INI** (License Server information), located in the Windows system directory, such as **C:\WINNT**.

7. Reboot your computer.

After you reboot, you should be able to get started using **APLAC** from the **Start** menu. **APLAC** will be able to checkout the license if you have a network connection to the license server, and the server is running with valid **APLAC** licenses.

FLEXlm License Server

1. For best results, exit all other applications before running the installation.

2. Start the FLEXlm License Server installer, using the **APLAC** Installation Launcher, or from **com\windows\flexlsi\setup.exe**.
3. If you already have FLEXlm-based licensing installed, the Installation Wizard detects this and stops. In these cases you must install the FLEXlm license server with the LMTools utility or use the existing license server by combining APLAC's license file with the existing one. You must also install the **APLAC** vendor daemon. For more information, see the FLEXlm End User Manual.
4. You will see standard Welcome and Information dialogs, containing useful information. Read the information carefully and press **Next** to proceed.
5. Select the directory where the License Server is to be installed, such as **C:\flexlm** and press **Next**.
6. Again, choose the name of the license file *(Default: license.dat)* and press **Next**.
7. Select the directory in which the license file is stored *(Default: **C:\flexlm**)* and press **OK**.
8. Before the system starts copying files, you will see a summary of the installation.

   Contents of the summary may vary according to the parameters of the installation and the target operating system. This data is saved as an **instlog.txt** file *(Default: **C:\flexlm\instlog.txt**)*, and is very important in case of installation problems.
9. The installation proceeds with files copied.
10. In Windows NT/2000/XP, the License Server runs as a Service.

    - To start it,

      *From the **Start** menu, choose **Settings** ⇒ **Control Panel** ⇒ **Administrative Tools in Win2000/XP** ⇒ **Services**. Its startup status should be **Automatic** so that it will start every time the computer is rebooted.*
11. Reboot your computer.

After you reboot, the computer should be usable as License Server for Floating Licenses immediately. The **flexlog.txt** file contains useful

startup information for the license server at its installation directory as well as information on license checkouts - in case of problems, consult this file.

APLAC Vendor Daemon

If you already run the FLEXlm system on Windows in your organization, and the license files use the same hostID, then you only need the vendor daemon (**lm_aplac.exe**) and the license file.

The vendor daemon can be installed using the Installation Launcher, or from **com\windows\vendordm**. Before the new licenses take effect, you should restart the license service, or perform a license file reread.

4.2 Portable License Installation

Portable Licensing is also based on FLEXlm technology starting from release 7.70. The hardlock key (usually a green hardlock device, FLEXID6 dongle) is connected to the parallel port of the computer and acts as a FLEXlm host identifier (hostID) that validates the **license file**. From version 8.00 onwards, FLEXID9 dongle, an USB-device, can be used also to tie the portable license.

Before starting the installation, save the license file somewhere. We suggest the usage of the **C:\flexlm** directory, but any folder that is always accessible is suitable.

If you have not obtained the license file, please contact support@aplac.com or licenses@aplac.com.

1. Connect the hardlock key into the parallel port.

2. Start the **APLAC** Installation Launcher and select the **APLAC Full Version** installation.

   You will see the standard ”Welcome” and ”License” Information dialogs. Press **Next** and **Yes** to proceed.

3. Select the directory where **APLAC** is to be installed *(Default:* ***C:\program files\aplac\aplac*** *<**version + type**>)* and press **Next**.

4. Indicate the name and the location of the license file.

5. You will be given an opportunity to create associations for .i (**APLAC Simulator** input) and .n (**APLAC Editor** simulation schematic) files.

6. Hardlock drivers are automatically installed after Portable Installation by default. You may also access the driver installer via the Installation Launcher.

   (In case you choose to postpone hardlock driver installation, the hardlock installer is also available at **com\windows\flexid6installer**.)

7. When you have completed installation you should be able to get started using **APLAC** immediately, from the **Start** menu.

5. Running APLAC in Windows

5.1 Launching APLAC in Windows

After installation, to start **APLAC** From the **Start** menu, choose **Programs** ⇒ **APLAC 8.10 FLEXlm** ⇒ **APLAC**

You will find menu items for **APLAC** software and user documentation. In FLEXlm licensed versions, the FLEXlm User Guide and LMTOOLS application can be accessed through this menu.

6. Installing APLAC in UNIX

APLAC can be installed on the following UNIX platforms:

Processor	Operating System
PA-RISC	HP-UX 11
SPARC	SUN Solaris 8 and above
Intel x86	LINUX/ELF 2.x.x kernel, glibc-version (e.g. recent Linux distributions such as Redhat 8.0 or Debian 3.0)

APLAC binaries are installed in a tree structure, created in a base directory. This base directory can be located at any point in the directory hierarchy accessible to all users. The **APLACPATH** environment variable must be set to point at this directory.

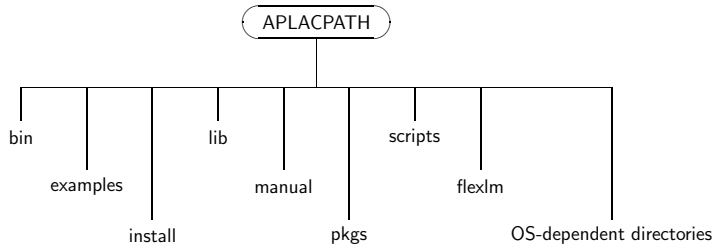

Figure 6.1: ***APLAC** directory structure, UNIX*

The **APLAC** software tree includes the following directories.

bin contains symbolic links to the **APLAC** launch scripts. This directory should be in the **PATH** of **APLAC** users.

examples contains **APLAC** examples, which can be used in building simulations. The **measurements** subdirectory contains **APLAC Editor** measurement templates.

install contains scripts used during the installation process.

lib is dedicated to the model libraries.

manual contains manuals and other documentation in PDF format.

pkgs contains utility files for various software packages, including the **aplaced** directory for platform-independent **APLAC Editor** files.

scripts contains the platform-independent scripts (such as runall.aplac) for launching the platform-specific binaries as well as the example aplac.conf file for setting environment variables for the license server. Application-specific scripts are also included.

flexlm contains license server and vendor daemon binaries for each supported UNIX operating system.

OS-dependent directories contain **APLAC** binaries and resource files for each supported UNIX operating system.

Example: HPUX_HPPA contains binaries for PA-RISC machines using the HP-UX operating system.

6.1 Installing and Updating APLAC

APLAC software can be installed either using an installation script **setup.sh** or manually. The full installation of the **APLAC Simulator** and **APLAC Editor** takes approximately 250 MB with binaries for all supported platforms. On-line user documentation is included.

Installation using setup.sh

1. Mount the CD-ROM device (on HP-UX systems, use mount option **-o cdcase**)
2. Run the setup script **setup.sh** found on the **APLAC** Installation CD-ROM:
   - **com/unix/setup.sh**
3. Select the directory where **APLAC** is to be installed (Default: **/usr/local/aplac-**<**version**>).
4. The installation script starts to copy program files. The script will ask you which platforms are to be used. The installation script uses the following codes for different platforms:

Code	Processor	Operating System
HPUX_HPPA	PA-RISC	HP-UX 11
SOLARIS_SPARC	SPARC	SUN Solaris 8 and above
LINUX_X86_GLIBC	Intel x86	LINUX/ELF 2.x.x kernel, glibc (eg. Redhat 8.0, Debian 3.0)

5. The FLEXlm License Server software is installed in the $APLACPATH/flexlm directory. The license file should be placed in this directory and named as **license.dat**.

6. **APLAC** uses environment variables to locate installed binaries and utility programs. Set the **APLACPATH** variable and add the directory **$APLACPATH/bin** into the application search path.

   Example: In the **/etc/profile** file specify:

   ```
   APLACPATH=/usr/local/aplac-8.10a
   PATH="$PATH:$APLACPATH/bin"
   export APLACPATH PATH
   ```

   and in the /etc/csh.cshrc file:

   ```
   setenv APLACPATH /usr/local/aplac-8.10a
   setenv PATH ${PATH}:$APLACPATH/bin
   ```

7. If you have installed the Floating License version, you can now start the License Server and set the license file path.

Updating using setup.sh

You can use the same installation script **setup.sh** to update your current **APLAC** installation. Use the following checklist to assure the smoothest transition.

- The **APLAC** installation script will ask whether you want update **APLAC** in the existing **APLAC** tree. If you are installing the updated version in another directory, indicate the appropriate directory name.

  NOTE: The structure of the examples directory **$APLACPATH/examples** has changed in **APLAC 8.10**. If you install over an earlier **APLAC**

version, the installation script will rename your previous examples directory **$APLACPATH/examples_old**. If you do not need the old example files, the examples_old directory can be removed.

- Many users find it advantageous to keep past versions of **APLAC** software for optional use.
- A number of **APLAC** binaries are named according the version number, so that old executables remain.

  Example: **APLAC** release 7.92a is installed and the simulator binary is named **aplac.run-7.92a**. The symbolic link **aplac.run** points to this file. When **APLAC** release 8.10a is installed, the symbolic link is updated to point to **aplac.run-8.10a**, the new binary.
- Other binaries (**spi2a** etc.) and scripts are updated to new versions. If you have edited **APLAC** scripts, rename them before updating. The installation program will overwrite all scripts except install/aplac.conf and **APLAC** resourcefiles (**Aplac**) in binary directories.
- If **setup.sh** locates binaries which do not follow the new version's naming convention, it will change the original name of binaries to **original_name-old**, such as **aplac.run** to **aplac.run-old**. You can rename old binaries according their version number, such as **aplac.run-7.50**.

  You can check the version number of renamed binaries using the command **original_name-old -ver** or **aplac -rel old -ver**.

Installing APLAC Manually

1. Create the base directory for the **APLAC** software tree.
   mkdir /usr/local/aplac
2. Set the **APLACPATH** environment variable.
   setenv APLACPATH /usr/local/aplac *(**csh** shell)*
3. Copy **base.tar** into the base directory.
   cp /mnt/cdrom/unix/base.tar /usr/local/aplac

4. Untar the **base.tar** file.
   cd /usr/local/aplac
   tar xvf base.tar

5. Uncompress and untar the **os.*.tar.Z** files in the base directory.
   uncompress os.*.tar.Z
   tar xvf os.*.tar
   If you do not want to install support for all platforms, uncompress only the relevant files.

6. Add the directory **$APLACPATH/bin** in the **PATH** variable.

7. Run the script $APLACPATH/install/install.aplac.
   cd install
   ./install.aplac

8. If you are installing the commercial version (with Floating or Node-locked licensing), install **FLEXlm** License Server software, found in the **server.tar** archive:
   cd $APLACPATH
   tar xvf server.tar

   The License Server software will be placed in the directory **$APLAC-PATH/flexlm**.

9. After installing License Server software, you can start the License Server and set the license file path.

6.2 Starting The License Server

The default location for License Server software is **$APLACPATH/flexlm**. This directory is divided into the following platform-specific subdirectories, which contain license manager daemons, vendor daemons and utility software:

Code	Processor	Operating System
HPUX_HPPA	PA-RISC	HP-UX 10.20 & 11
SOLARIS_SPARC	SPARC	SUN Solaris 8 and above
LINUX_X86_GLIBC	Intel x86	LINUX/ELF 2.x.x kernel, glibc (libc6)

If the license file is stored in another directory, the location must be indicated using using one of two environment variables:

- **APLAC_LM_LICENSE_FILE** is APLAC-specific and can be defined in $APLACPATH/scripts/aplac.conf.
- **LM_LICENSE_FILE** is FLEXlm-specific and recognized by all clients using FLEXlm licensing.

The location can also be **port@host** setting, where port and host define the network location of the license server in network.

Example: setenv APLAC_LM_LICENSE_FILE 27000@host (C shell)

The License Server **lmgrd** can be started using the command: <lmgrd_path>/lmgrd -c license_file -l log_file.

cd $APLACPATH/flexlm/v9.2/LINUX_X86_GLIBC/
lmgrd -c ../license.dat -l ../log.txt

- The License Server can also be started at system start-up.
- You can combine a new license file with existing ones.
- With an Options file for the Vendor daemon, you can fine-tune the license checking process (enable/disable the access for the licenses for some user, etc.).
- For more information concerning license management, see the *FLEXlm End User Manual* (**com/docs/flexlm.pdf** in the Installation CD).

6.3 Configuring APLAC Program Resources

After installing the **APLAC Simulator** and **APLAC Editor** client software or License Server software, characteristics of the interface and functionality can be configured by manipulating parameters within resource files.

The APLAC Simulator

APLAC Simulator's interface parameters are defined using an application resource file **Aplac**, located in the **APLAC** directory tree.

The application resource file's search priority is:

1. $HOME/Aplac
2. $APLACRESDIR/Aplac
3. /usr/lib/X11/app-defaults/Aplac

The search location can be changed with the **$APLACRESDIR** variable.

Some resources are platform-dependent, so **APLAC** resource files are located in platform-specific binary directories.

Example: The platform-dependent resource file for **Linux/glibc** is located in the directory **APLACPATH/LINUX_X86_GLIBC**.

Usually, only the printing commands **APLAC*printHpglToPrinter:** and **APLAC*printPsToPrinter:** need editing.

The following parameters in the resource file correspond to command-line parameters listed in the User's Manual:

```
APLAC*defsize:  geom
APLAC*font:     font
APLAC*fontMath: font
APLAC*bg: color
APLAC*fg: color
APLAC*cursor:   cn
APLAC*cursorfg: color
APLAC*cursorbg: color
APLAC*prompt:   string
APLAC*epsOptions:   string
APLAC*blackAndWhiteMode: boolean
```

NOTE: Command-line arguments always override resource parameters.

The following entries are only supported in the resource file:

Dialog background color ***NOTE:*** Default dialog background color is white

```
APLAC*dialogBackground: color
```

Printing Commands

```
APLAC*messagePsToPrinter:   MessageString
APLAC*printPsToPrinter:     CommandString
APLAC*messageHpglToPrinter: MessageString
APLAC*printHpglToPrinter:   CommandString
```

Message strings appear in the File/Print dialog box as button texts. If you select an option (PostScript or HPGL), **APLAC** patches the accompanying command string and runs it. Patching means that '%src' in the commands is replaced by the name of a temporary output file that is automatically created and deleted by **APLAC**.

NOTE: Platform-specific configurations are defined using the environment variable $APLACRESDIR, which is handled by the **runall.aplac** script. If **$APLACRESDIR** is set in system files, **runall.aplac** will not reset it.

The APLAC Editor

APLAC Editor resource parameters are not system-wide, but user-controlled using the Options menu. These user-specific parameters are saved in the file **$HOME/.aplacedini**.

The APLAC Editor and HP-UX

Users running the **APLAC Editor** on HP-UX have found that zooming and other view editing commands can behave slowly. Performance can be improved by editing the Xserver's font path, according to the following considerations:

- Keep font paths as short in length as possible.
- List often-used fonts directories/servers first in the font path
- List non-standard fonts directories/servers last

If you are running the HP-UX version of the **APLAC Editor** from an X-terminal, using XFree86 server, then you will also need to set up a font server in an HP-UX machine and add it to the X-terminal's font path.

7. Running APLAC in UNIX

The pd_ctrls.dat file
On startup you may encounter a dialog box informing you about errors in the pd_ctrls.dat file. This file contains the **APLAC Editor** control object data. Since users can customize the control object data, the user-specific file is stored as .pd_ctrls.dat in the user home directory. The **APLAC Editor** launch script checks whether the user-specific file exists or not. If it doesn't, it is created as a copy of the master file **$APLACPATH/pkgs/aplaced/pd_ctrls.dat.base**. If you haven't edited the **.pd_ctrls.dat** file in your home directory, the most straightforward way to eliminate the errors is to simply delete it and restart **APLAC Editor**.

7.1 Launching APLAC in UNIX

After installation the environment variable **APLACPATH** should point to the directory under which **APLAC** software is installed and the directory **$APLACPATH/bin** should be on the search path pointed out by the environment variable **PATH**; all the executable files of **APLAC** are located in the **bin** directory. You can set up a Bourne or Korn shell as follows (the installation directory is assumed to be **/usr/local/aplac**):

```
APLACPATH=/usr/local/aplac

PATH=$APLACPATH/bin:$PATH

export APLACPATH PATH
```

For C shell, the syntax is:

```
setenv APLACPATH /usr/local/aplac
set path=($APLACPATH/bin $path)
```

You may also want to set the **PATH** environment variable in your .profile, **.login**, .cshrc or .kshrc file to include **$APLACPATH**.

APLAC is started via the script file called **aplac**. If you do not give any file as an argument for the **aplac** script then the **APLAC Editor** is started. If you give a file as the argument it is assumed to be **APLAC** netlist and **APLAC Simulator** is invoked. This enables batch mode **APLAC** simulations from some other software.

The **APLAC Editor** window is open constantly when using **APLAC** in a UNIX environment. The **APLAC Editor** invokes a batch mode **APLAC Simulator** process each time you do a new simulation. The simulator can be closed by the user when simulation and graphic post-processing are done.

Text output (**stdout**) and errors and warnings (**stderr**) are, by default, directed to text output window that is opened automatically.

After successful execution the program returns the exit code 0; on error, the exit code is 1. Note that graphic output files are created relatively starting from the same directory where the input file is unless an absolute pathname is specified.

7.2 Resource files: .Xresources and APLAC

The X11 resource files define several parameters that affect **APLAC** program behavior and appearance. The basic system resource files should not contain any **APLAC**-related parameters. If the system reads the file **$HOME/.Xresources**, the **APLAC** parameters can be defined in that file. If it is desired to keep **APLAC**'s resources separate, they can be put in the file **Aplac**.

If a specified resource is not found in **$HOME/.Xresources**, an attempt is made to find it in the application defaults file **Aplac**. This file can be in several places, the precedence being

1. $HOME/Aplac
2. $APLACRESDIR/Aplac
3. /usr/lib/X11/app-defaults/Aplac

The following **APLAC**-related parameters in the resource file(s) correspond to the command-line parameters listed in the preceding section (note that the command-line arguments always override the resource settings)

```
APLAC*defsize:    geom
APLAC*font:       font
APLAC*fontMath:   font
APLAC*fontTextOutput: font
APLAC*bg:         color
APLAC*fg:         color
APLAC*cursor:     cn
APLAC*cursorfg:   color
APLAC*cursorbg:   color
APLAC*prompt:     string
APLAC*epsOptions: string
APLAC*blackAndWhiteMode: boolean
```

The following entries are only supported in the resource file(s):

Dialog backgrounds

APLAC*dialogBackground: Color

where **Color** is an X11 color name. The default is **white**.

Printing commands

APLAC*messagePsToPrinter: MessageString

APLAC*printPsToPrinter: CommandString

APLAC*messageHpglToPrinter: MessageString

APLAC*printHpglToPrinter: CommandString

The message strings appear in the File/Print dialog box as button texts. When you select an option (PostScript or HPGL), **APLAC** patches the accompanying command string and runs it. Patching simply means that %src in the commands is replaced by the name of a temporary file that is created (and deleted) by **APLAC** to store the output; you normally do not specify it.

A typical **.Xresources** or **Aplac** file might look as follows:

```
.
.
APLAC*defsize:  600x400
APLAC*fg:       red
APLAC*bg:       cyan
APLAC*font:     6x10
# (comment) cursor number 62 = heart
APLAC*cursor:   62
APLAC*epsOptions: landscape
APLAC*messagePsToPrinter: Print plotter output with PS printer
APLAC*printPsToPrinter:   /usr/bin/lp%src
APLAC*messageHpglToPrinter: Print plotter output with HPGL printer
APLAC*printHpglToPrinter:   /usr/bin/lp -ohpgl2%src
.
.
```

7.3 User defaults: .aplac

Information about **APLAC** graphics windows is stored in the file $HOME/.aplac. These are user defaults - there is no system-wide $HOME/.aplac file. If the file does not exist, it will be automatically created. The following entries are supported in the .aplac file:

Window geometry information

Window <**n**>**:** <**width**> **x**<**height**> **+**<**xoffset**> **+**<**yoffset**>

where

<n>	window number, starting from 0
<width>	window width in pixels
<height>	window height in pixels
<xoffset>	x coordinate in pixels
<yoffset>	y coordinate in pixels

The upper left corner of the window is located at (<xoffset>,<yoffset>) relative to the screen coordinates. The origin (0,0) is at the upper left corner of the screen; x coordinates increase to the right, y coordinates, downwards. Negative offsets are allowed. Window width and height must be non-negative.

WindowText <**n**>**:** <**width**> **x**<**height**> **+**<**xoffset**> **+**<**yoffset**>

WindowVerbose <**n**>**:** <**width**> **x**<**height**> **+**<**xoffset**> **+**<**yoffset**>

These apply to the text output and verbose windows.

Enabling/Disabling saving of configuration

APLAC*SaveOnExit: Boolean

where **Boolean** is either **true** or **false**. Setting **SaveOnExit** to **false** will disable saving of the configuration, otherwise .aplac is updated before exiting **APLAC**. **SaveOnExit** is **true** by default.

Hard vs. Soft limits

APLAC*hardLimitsX: Boolean
APLAC*hardLimitsY: Boolean
APLAC*hardLimitsY2: Boolean

where **Boolean** is either **true** or **false**. If any of these is **true**, **APLAC** will use exact limits on the corresponding axis if autoscaling is requested. Otherwise, soft, that is, rounded, limits will be used.

Clipping markers

APLAC*clipMarkers: Boolean

where **Boolean** is either **true** or **false**. If this item is **true**, **APLAC** will clip markers at the edges of the drawing area. If the item is **false**, markers are not clipped. The default is **false**.

Removing trailing zeroes

APLAC*removeTrailingZeroes: Boolean

where **Boolean** is either **true** or **false**. If this item is **true** (the default), **APLAC** will remove trailing zeroes from axis texts, that is, instead of 4.00', '4 will be written.

Dense log grid

APLAC*denseLogGrid: Boolean

where **Boolean** is either **true** or **false**. If this item is **true**, **APLAC** will use long lines inside the drawing area for logarithmic axes if, additionally, the window grid is on. If the item is **false**, short ticks outside the drawing area are drawn. The default is **true**.

Tuning window size

APLAC*tuneVisible: Integer

where **Integer** is between 2 and 20. This is the maximum number of tuning variables that will be shown in the dialog box. The default is 5.

Differential clipboard

APLAC*differentialClipboard: Boolean

where **Boolean** is either **true** or **false**. If this item is **true**, **APLAC** will write curves to the clipboard as midpoints and differences ($+/-$). Thus, the number of digits saved is greater than if the curve numbers themselves were stored. However, importing/exporting curves is harder. It should be noted that not all digits saved are necessarily significant. The default is **false**.

Cursors

APLAC*startWithCursors: Boolean

where **Boolean** is either **true** or **false**. If this item is **true**, **APLAC** will put cursors on initially (they can be turned off later). If the item is **false**, cursors are only turned on through the Options/Cursors menu item. The default is **false**.

Vertical-axis text direction

APLAC*axisTextYDirection: Setting

where **Setting** is either **automatic**, **vertical** or **horizontal**. If the setting is **automatic**, **APLAC** will display short texts horizontally,

while longer texts are displayed vertically. Setting the direction to **vertical** or **horizontal** turns off the automatism. The default is **automatic**.

TeX mode

APLAC*TeXmode: Boolean

where **Boolean** is either **true** or **false**. If this item is **true**, **APLAC** will interpret text to be output to a window - say, curve names - according to (basic) TeX syntax. In other words, an underscore starts a subscript; an up-arrow, a superscript. Math symbols, too, are supported (**SIGMA** produces a Σ, etc.). The default is **false**.

MonteCarlo averaging

APLAC*monteCarloAverage: Boolean

where **Boolean** is either **true** or **false**. If this item is **true**, all MonteCarlo curves will be, by default, averaged at the end of the analysis and only the maximum, minimum and average points will be plotted. This can be circumvented for any window by specifying **MC_DRAW_ALL** for that window in the input file. The default for MonteCarlo averaging is **false**, that is, **MC_DRAW_ALL** holds for every window unless otherwise specified.

XOR

APLAC*xorDiamond: Boolean

where **Boolean** is either **true** or **false**. This item may help if cursors are invisible. The background is that X11 servers appear to implement XOR drawing in slightly different ways and the XFree86 Diamond server's implementation is different from other XFree86 servers. Thus, only Linux users with Diamond display adapters may actually need this resource. The default value is **false**.

Version-specific warnings

APLAC*versionWarning: Boolean major.minor

where **Boolean** is either **true** or **false**. This item's function is to enable/disable version-specific warnings. **true** enables, **false** disables. The version (major.minor, an example: 7.62) sets the limit. true 7.62' would enable warnings specific to 7.62 and later versions, whereas false 7.92 would disable warnings specific to 7.92 and earlier versions. The default is true x.yy', where x.yy is

APLAC's current version, that is, all warnings specific to the current version (or above, if any) are printed. Warnings specific to earlier versions are disabled.

Saving memory

APLAC*saveMemory: Boolean

where **Boolean** is either **true** or **false**. To speed up redraws, **APLAC** normally draws graphics both on screen and into memory (RAM). This limits the number of number windows available if there is little RAM, as on some X11 terminals. Setting this item **true** disables drawing into RAM, that is, allows more windows, but results in slower redraws.

Marking Phase/Y2 curves

APLAC*markPY2Curves: Boolean

where **Boolean** is either **true** or **false**. This option makes **APLAC** mark Phase/Y2 curves in the legend (below the drawing area) with a small dot.

Text buffer sizes

APLAC*textBufferSize: 2853
APLAC*verboseBufferSize: 144

Size and location of text windows

WindowText 672x350+5+2
WindowVerbose 672x350+4+391

Before **APLAC** graphics windows are created, geometry information is read from the file .aplac. If no information is available, settings in **.Xresources** will be used instead. If no settings are found, behavior will depend on the window manager used.

Here is an example of an .aplac file:

```
APLAC*hardLimitsY:      false
Window0:     400x400+600+1
Window1:     600x400+600+400
APLAC*saveOnExit:  false
```

7.4 Font Scaling

The text fonts in **APLAC**'s graphics windows can be scaled automatically according to the window's size. In other words, the smaller the window, the smaller the font, and vice versa. This feature is based on X11's ability to use font *aliases*.

A regular X11 font name is long and convoluted, as it is also a shorthand description of the font. Consider an example:

```
-adobe-new century
schoolbook-medium-r-normal--14-100-100-100-p-82-iso8859-1
```

This font name includes the foundry (Adobe), the typeface name (New Century Schoolbook), the point size (14), the weight, the slant, the encoding (ISO8859-1), and various other specifications.

A font name like the above is, obviously, not really usable by humans, a problem that is addressed by aliases. Aliasing a font means attaching another, shorter, name. X11 looks for these in a file called fonts.alias. This file will be found somewhere in X11's font path; unfortunately, the exact location depends on the particular X11 implementation.

Assuming the fonts.alias file has been found, making an alias means adding lines to the file by hand:

```
ncenr14 -adobe-new century
schoolbook-medium-r-normal--14-100-100-100-p-82-iso8859-1
```

After X11 is restarted, the new, short alias **ncenr14** can be used anywhere the long name can.

APLAC's ability to rescale a font depends on a number of hard-coded font aliases. These aliases consist of two parts; the first, letter, part identifies the typeface together with weight (normal/bold) and slant (normal/italic), the second (2 digits), the size in points. One example:

```
timbi08
```

is the Times-Roman, bold, italic, font; the size is 8 points.

The letter parts of the aliases are (the corresponding typefaces are Charter, Courier, Helvetica, Lucida Bright, New Century Schoolbook, Times Roman, and Symbol, respectively):

```
charb, charbi, chari, charr
courb, courbo, couro, courr
helvb, helvbo, helvo, helvr
lubb, lubbi, lubi, lubis, lubr, lubs
ncenb, ncenbi, nceni, ncenr
timb, timbi, timi, timr
symb
```

The point-size parts are

```
08,10,12,14,18,24
```

Now, let us assume you want to use the New Century Schoolbook bold font. The corresponding ncenb aliases might already be available; xlsfonts will tell you whether they are.

If they are not, you must edit the fonts.alias file and add rows for “ncenb08”, “ncenb10”, and so forth (see above). Then, restart X11.

Once the ncenb aliases are available, one can be selected from **APLAC**’s menu (Edit—Font), or **APLAC** can be started with a command line:

```
aplac -font ncenb file.i
```

Depending on the window size, **APLAC** will then make up a font alias by appending a likely point size (08, 10, etc.) to the **ncenb** specification and use the combination **ncenb14** to load a font.

7.5 Examples

Request of the **APLAC** version:

```
aplac -ver
```

Quick syntax reference of **Res**:

```
aplac -syntax Res
```

Start **APLAC** file chebysh.i (note that extension .i may be omitted) and redirect the text output to the file **chebysh.out** (the -ntw flag suppresses the normal text output window):

```
aplac -ntw chebysh > chebysh.out
```

Run **APLAC** file tlsweep.i and create the window on the display defined by the **DISPLAY** environment variable. All other graphics related parameters are system defaults unless they are specified in the file **.Xresources**. In addition all displayed results are printed to text output window. An example command line follows:

```
aplac tlsweep -p
```

The graphics output is redirected host **wks.domain.com** (if the user is allowed to do so !). In the UNIX system a blue window appears on the top left corner of the display (horizontal size 480 pixels and vertical size 400 pixels).

```
aplac tlsweep -geometry 480x400+1+1 -bg blue -display wks.domain.com:0
```

NOTE: The following command-line arguments are not used in the Windows version:

- -bg
- -blackandwhite
- -cursor
- -cursorbg
- -cursorfg
- -display
- -fg
- -font
- -fontmath
- -geometry
- -help
- -loadsweep
- -mcassemble
- -nodisplay
- -option
- -prompt
- -silent
- -users
- -verbose
- -windowplaces

8. FLEXlm License Administration

Using a License Server, FLEXlm licensing technology makes multiple licenses available to member workstations within a network. Typically a stable network node such as a UNIX workstation or NT Server is used for this task. Each workstation can **check out** an available license, making that license unavailable to other workstations. A standalone installation makes it possible to create licenses for computers not permanently connected to the network.

Sites using electronic design automation software are typically equipped with FLEXlm license management. **APLAC** conforms closely to standard conventions of the FLEXlm system. Following the detailed instructions given in this manual for Windows and UNIX installation will help assure a smooth beginning.

The FLEXlm license management system incorporates the following components:

License Manager daemon (lmgrd) - background process requiring no direct intervention, the License Manager daemon handles the initial contact with **APLAC** and the Vendor daemon. The License Manager binary is provided by Globetrotter Inc.

Vendor daemon - Another background process, the application-specific Vendor daemon grants licenses for the licensed application, keeping track of licenses as they are checked out, and who has them. **APLAC** Solutions Vendor daemon is called **lm_aplac**.

Vendor Applications - (**APLAC Simulator**, **APLAC Editor**, ...)

The Vendor application software may be located on the same network node as the vendor daemon, or it may be located elsewhere, as it communicates through a network communications protocol. Usage of a standard network protocol assures that **APLAC** software and License Server software can be installed in completely

different operating systems, guaranteeing a completely heterogeneous design environment across a range of workstations.

License file - The License file holds all licensing data in a human-readable format. It contains data about server nodes and vendor daemons, with a feature line for each licensed product, including software version data. Feature lines are designed so that editing the contents of the License file will make the license invalid. For more information concerning license management, see the *FLEXlm End User Manual* (**com/docs/flexlm.pdf** in the Installation CD).

8.1 License File Location

The license file appears as a plain text file that resides in a default location. If the license file is stored in another directory, the location must be indicated.

Windows versions search first for a reference to the license file from the aplacsrv.ini file. This reference variable is defined at installation and defaults to C:\flexlm\license.dat. If this link variable fails, Windows will look at **C:\flexlm\license.dat** to find a license.

UNIX versions search first for a link to the license file using the APLAC_LM_LICENSE_FILE environment variable. This environment variable may be set in **$APLACPATH/scripts/aplac.conf**, in system wide configuration files or in personal configuration files.

If the **APLAC_LM_LICENSE_FILE** is not set or the license file is not found from the location indicated, then the search priority is:

1. `$APLACPATH/flexlm/license.dat`
2. `LM_LICENSE_FILE`
3. `/usr/local/flexlm/licenses/license.dat`

NOTE: **APLAC_LM_LICENSE_FILE** and/or **LM_LICENSE_FILE** can contain a colon-separated list of License Servers in **port@host** format. If

the License Server running at one host becomes unavailable, the License Server at the next host takes over, if it has valid **APLAC** licenses available.

This is a very important feature enabling **APLAC** software to enhance the fault tolerance of long simulations by changing License Servers on the fly.

A port number can also be assigned for the license file, telling the License Server to assign license calls to that port.

8.2 License Management Tools

FLEXlm provides several utility programs, some for the end user of the licensed product, and others used only by the license administrator of the site. In UNIX, license management tools are installed in platform binary directories. In Windows, you can also use the **LMTools** interface accessible from the **Start** menu, to manage the functionality of the License Server.

These tools are valuable for both end users and license administrators:

lmdiag - This tool will diagnose the problem when you cannot check out a suitable license. You can specify a license file explicitly with command line option **-c**.

lmstat - This tool helps you monitor the entire network's licensing system.

These tools are valuable for license administrators:

lmcksum - This tool can be used to verify that the license file format is correct.

lmdown - This is the primary method for shutting down the entire FLEXlm license system (**lmgrd** and vendor daemons).

lmhostid - This tool reports the hostID of the system.

lmremove - This tool allows you to remove a single user's license for a specified feature. This may be needed if an individual client node crashes or when **APLAC** has swapped the License Server because of a fault in the first License Server.

lmreread - This tool causes the license daemon to reread the license file and start any new vendor daemons that have been added. With this utility, you do not have to shut down the entire License Server framework to add software products (indicating a new vendor daemon).

lmver - This tool determines the version of the FLEXlm system used, inside any binary that supports FLEXlm.

For more information concerning license management tools, see the *FLEXlm End User Manual* (**com/docs/flexlm.pdf** in the Installation CD).

8.3 Advanced Licensing Features

Options File

The **Options File** allows the network's license administrator to manage a variety of FLEXlm parameters. The administrator can reserve licenses for specific designers, control access to features, control the flow of information to log files, etc. The Options File can be used without Overdraft Access, but it makes Overdraft Access possible, by launching report logging. For more information concerning license administration, see the *FLEXlm End User Manual* (**com/docs/flexlm.pdf** in the Installation CD).

To implement the Options File,

1. Create the file, typically **C:\flexlm\options\lm_aplac.opt** (Windows) or **/usr/local/flexlm/options/lm_aplac.opt** (UNIX)

2. Add the Options File's pathname to the license file, as the fourth field in the **APLAC** vendor daemon.

Troubleshooting and Diagnostics

If you are trying to use **APLAC** client software but no license seems to be available, the following techniques can be used to locate the problem and offer a solution:

- Make sure that the license file is accessible, at the location defined using the installation instructions given.
- If you are using a Floating license, make sure that your workstation's connection to the License Server is active. To test this, use **ping [servername]** at the command prompt.
- Make sure that processes **lmgrd** and **lm_aplac** are running (at the License Server node):
  - In Windows NT, use the **Task Manager** to assure that processes **lmgrd.exe** and **lm_aplac.exe** are running.
  - In UNIX, use the **ps** command to print all running processes and make sure that **lmgrd** is running.

    Example:

    ps -e | **grep lmgrd** in HP-UX, Solaris
    ps a | **grep lmgrd** in Linux

In case you are a license administrator, **APLAC** recommends the following preventive and diagnostic techniques for use with FLEXlm:

- Always use a log file when running the License Server. The contents of a log file can be very helpful in diagnosing a problem, and if you contact **APLAC** or FLEXlm support, you will be asked about the contents of this file.

  The log file used in Windows installations is **flexlog.txt**, typically located in **C:\flexlm**.

- If the log file **flexlog.txt** does not exist in the License Server, then the **lmgrd** process has not started. Most probably you have a faulty or tampered license file. Check especially that the **SERVER** and **VENDOR** lines are intact.
- If **lmgrd** has started correctly (indicated by a log file with no alarming entries), see if **lmstat -a** or **lmdiag** offer any clues.
- You can also set the FLEXlm **FLEXLM_DIAGNOSTICS** environment variable to launch diagnostics for the entire licensing system.
- If the FLEXlm license server is running on the local host, and the host is not connected to a network, make sure the path to your local license file comes before any other license paths in the registry key
  HKEY_LOCAL_MACHINE\SOFTWARE
  \FLEXlm_License_Manager\LM_APLAC_LICENSE_FILE.
- Be sure that you have implemented the correct version of the FLEXlm vendor daemon.

For more information concerning license management, see the *FLEXlm End User Manual*
(**com/docs/flexlm.pdf** in the Installation CD).