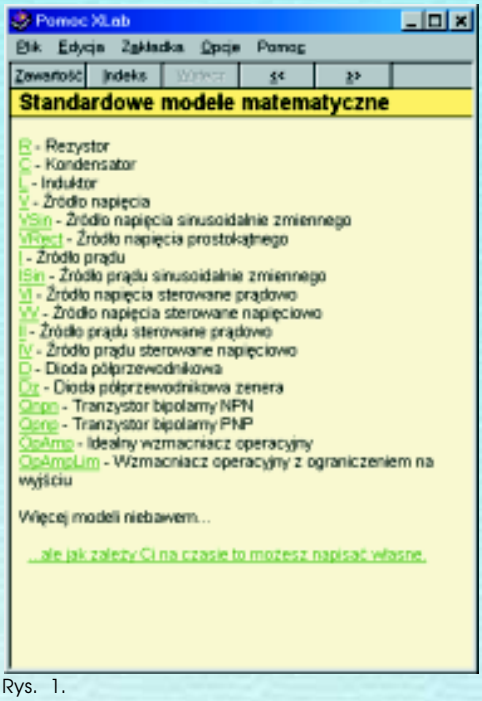
XLab

Symulator układów analogowych

Ostatnio, ku uciesze użytkowników komputerów osobistych, nastała moda na pisanie darmowego oprogramowania. Weźmy pod uwagę chociażby system operacyjny Linux, który zdobywa sobie coraz większą popularność oraz całą masę różnych aplikacji typu freeware ze Star Officem na czele. Dlaczego by nie zrobić na podobnych zasadach bezpłatnego, polskiego programu CAD dla elektroników hobbystów? Właśnie ta myśl przyświecała mi, gdy zacząłem tworzyć XLab - program do symulacji układów elektronicznych. Starałem się nie powielać rozwiązań stosowanych w komercyjnych aplikacjach tego typu, bowiem nie chodzi mi o napisanie jeszcze jednego Protela czy EdWina (bardzo dobre programy). Czy mi się to udało, ocenią użytkownicy.

Instalacja

Pierwszą różnicę widać już po rozpakowaniu archiwum. Program składa się właściwie tylko z jednego pliku wykonywalnego, będącego zintegrowanym edytorem schematów, symulatorem i przeglądarką wyników. Napisanie tych modułów jako trzech oddzielnych aplikacji wiązałoby się ze spadkiem wydajności systemu oraz prawie dwukrotnym wzrostem objętości plików (nawet stosując biblioteki *.dll albo *.bpl). Właściwie nie trzeba instalować pakietu. Jest gotowy do pracy od razu po umieszczeniu plików w docelowym folderze na twardym dysku użytkownika. Specjalnie dla początkujących, oprócz pomocy dołączonych jest kilka przykładów prezentujących jego możliwości. Dlatego pierwszą symulację można przeprowadzić już w kilka minut po zapoznaniu się z podstawowymi funkcjami.



Rys. 1.

Wprowadzanie schematu

Opis układu umożliwia specjalny, obiektowy język programowania. Nie należy się go bać - jest naprawdę prosty, zdecydowanie łatwiejszy od BASIC-a. Najlepiej najpierw narysować sobie szkic schematu na kartce papieru. Nadawszy wszystkim węzłom odpowiednie nazwy umieszcza się ich deklaracje na początku opisu topologii. Nagłówek wraz z deklaracją masy i zasilania jest już podany, więc nie trzeba zaczynać „od zera". Węzły mogą mieć określony stały potencjał. Nie spełniają wtedy pierwszego prawa Kirchhoffa - działają raczej jak połączenia zewnętrzne (np. masa o zawsze zerowym potencjale). Musi być zdefiniowany co najmniej jeden taki węzeł.

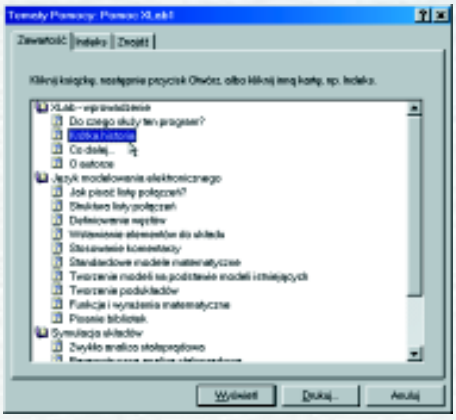
Następnie użytkownik musi wyszczególnić kolejno elementy występujące w układzie, podając przy każdym, jaki ma być wykorzystany model matematyczny. Określa on zachowanie się elementu w zależności od stanu potencjałów i prądów na jego końcówkach. Może to być układ równań (dokładniej: zestaw funkcji wielowymiarowych przyrównanych do zera), określony bezpośrednio (np. dla opornika jest to prawo Ohma) albo poprzez użycie podukładu - wtedy równania uzyskuje się poprzez kompilację. Przeniesienie części układu do podukładu nie spowalnia obliczeń. Podukłady mogą zawierać inne podukłady - liczba zagnieżdżeń, podobnie jak liczba elementów w projekcie, jest praktycznie nieograniczona, chociaż bardzo duże układy zajmują odpowiednio dużo pamięci i długo są przetwarzane. Definicje, które będą wykorzystane w wielu projektach, należy umieścić w oddzielnych bibliotekach, dołączanych później jednym poleceniem.

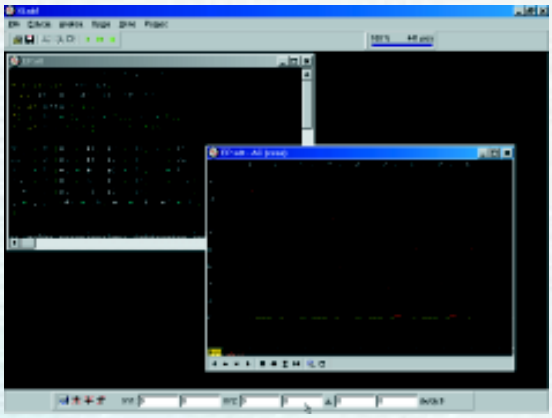
W programie znajduje się kilkanaście podstawowych modeli, takich jak: źródła prądu i napięcia, włączniki sterowane, elementy RLC, diody, tranzystory i proste wzmacniacze operacyjne. Nie jest to na razie liczba zawrotna (**rys. 1**), ale w przyszłych wersjach programu znacznie się zwiększy.

Po zadecydowaniu o rodzaju elementu, należy podać, do czego ma być podłączony. Tu przydają się wcześniej zdefiniowane nazwy węzłów. Składnia wygląda następująco: *{końcówka1 = węzeł1, końcówka2 = węzeł2,...}*. Dzięki temu od razu przejrzyście widać, co i do czego zostało podłączone. Nie jest wymagane zachowanie kolejności ani wykorzystywanie wszystkich wyprowadzeń.

W dalszej kolejności trzeba ustalić wartości parametrów, stosując dokładnie ten sam schemat, co wcześniej z łączeniem końcówek. Po lewej stronie znaku równości występuje nazwa parametru, a po prawej jego wartość. Zależnie od modelu, parametry mogą posiadać wartość domyślną, wtedy określanie ich nie jest konieczne.

Wbrew pozorom, przy pewnej wprawie tą metodą układy wprowadza się nawet nieco szybciej niż korzystając z edytorów graficznych. Szczególnie sprawnie dokonuje się poprawek w istniejącym projekcie. Przyczyną jest to, że nie traci się czasu na prowadzenie przewodów i „kosmetykę" schematu. Dodatkowo sprawę ułatwia kolorowy edytor tekstu podświetlający odpowiednio różne fragmenty kodu. Użytkownik ma pełną kontrolę nad wprowadzanym projektem. Odpada też problem niezgodności formatów plików. Listę połączeń można odczytać nawet na komputerze bez zainstalowanego symulatora, używając notatnika albo WordPada. W przypadku ja-


Rys. 2.



Rys. 3.

kichkolwiek trudności przydaje się interaktywna pomoc, w której zamieściłem dokładny opis składni języka oraz wszystkich modeli standardowych (**rys. 2**). Pominąłem natomiast wiele informacji podstawowych (takich jak np. „Jak zapisać projekt na dysk?“), które nawet mało obeznany z obsługą komputera użytkownik zdobędzie w ciągu paru minut pracy z XLabem.

Symulacja

Gdy wpisana lista połączeń jest gotowa, należy ją skompilować, czyli przetłumaczyć do postaci zrozumiałej przez komputer, z której bezpośrednio można wygenerować odpowiedni układ równań. Dla dużych schematów może być ich nawet kilkaset. W przypadku napotkania błędu wyświetlony zostanie odpowiedni komunikat, a kursor ustawi się w miejscu, które trzeba poprawić. Kiedy wprowadzone zostaną wszystkie niezbędne poprawki, a topologia ponownie skompilowana, można rozpocząć symulację. Ważne jest, że przed kompilacją projekt zostaje każdorazowo automatycznie zapisany na dysku. Dzięki temu, gdyby nastąpiła jakaś usterka (proces tworzenia w pamięci relacyjnej bazy danych o elementach i powiązaniach między nimi nie należy do prostych), użytkownik nie utraci wyników swojej pracy.

XLab oferuje pięć rodzajów analiz: stałoprądową, stałoprądową parametryczną, czasową (**rys. 3**), zespoloną częstotliwościową (**rys. 4**) oraz zespoloną parametryczną (począwszy od wersji 1.09).

Analiza stałoprądowa polega na obliczeniu punktu pracy układu. Elementy takie jak kondensatory, indukcyjności i źródła sygnałów zmiennych są pomijane. Z tego powodu np. w zwykłym prostowniku program ustali, że napięcie wyjściowe wynosi dokładnie zero, co zupełnie odbiega od rzeczywistości. Mimo to jest ona bardzo użyteczna przy ustawianiu polaryzacji tranzystorów we wszelkiego rodzaju wzmacniaczach. Nadaje się zwłaszcza do tego jej rozszerzona wersja, w której można wykonać całą serię pomiarów dla różnych wartości jakiegoś parametru, np. napięcia zasilania.

Niewątpliwie jednak najczęściej używana jest symulacja czasowa. Pozwala zamienić komputer w cyfrowy, wielokanałowy oscyloskop (**rys. 5**). Z tego powodu do jej uruchomienia wymagane jest podanie większej liczby parametrów. Szczególnie duże są możliwości dobierania dokładności obliczeń. XLab stosuje zmienną długość kroku symulacyjnego. W momencie, gdy przebiegi czasowe napięć i prądów stają się bardziej skomplikowane, krok ten jest zawężany. Gdy symulator natrafi na strome zbocze sygnału, odległość między próbkami może zostać zredukowana nawet miliard razy. Zwiększa to znacznie szczegółowość wyników, a zarazem skraca czas oczekiwania na zakończenie symulacji.

Ostatnie dwa rodzaje analiz operują na podwójnej liczbie równań - wykorzystywane są liczby zespolone. Oznacza to, że każdy potencjał i prąd zapisywane są za pomocą dwóch wartości - rzeczywistej i urojonej. Można je łatwo przełożyć na kąt przesunięcia fazowego i amplitudę. Analiza zespolona przydatna jest przede wszystkim przy projektowaniu wzmacniaczy oraz filtrów, ponieważ umożliwia wyznaczanie charakterystyki częstotliwościowej oraz fazowej.

Następne wersje symulatora planuję rozbudować jeszcze o analizę temperaturową, szumów, FFT, a może nawet i MonteCarlo. Wtedy jego zakres zastosowań na pewno rozszerzy się jeszcze bardziej.

XLab pozwala także na powtarzanie tej samej analizy dla różnych wartości tego samego parametru. Tą metodą można np. wykreślić rodzinę charakterystyk wyjściowych tranzystora albo badać układ korektora graficznego audio. Również nie ma żadnych ograniczeń na liczbę przeprowadzanych symulacji tego samego układu. Możliwa jest obserwacja np. odpowiedzi impulsowej filtru w jednym, a jego charakterystyki częstotliwościowej w drugim oknie.

Bardzo użyteczną właściwością symulatora jest jego wielowątkowość. Oznacza to, że wszystkie obliczenia przeprowadzane są w tle i nie przeszkadzają w pracy np. nad innym projektem (można otworzyć kilka projektów równocześnie). Program składuje „zadania symulacyjne“ na specjalnym stosie. Gwarantuje to, że żadna komenda wysłana przez użytkownika nie zostanie zignorowana, nawet jeśli nie da się jej wykonać od razu. W trakcie symulacji wyświetlana jest szybkość pracy (liczba próbek na sekundę) oraz procentowy postęp w obliczeniach. Jeśli z jakichś powodów konieczne jest opuszczenie aplikacji przed zakończeniem symulacji, nie należy obawiać się o utratę danych. Po ponownym uruchomieniu XLab będzie kontynuował obliczenia od miejsca, w którym zostały przerwane. Podobnie, jeśli przez pomyłkę ustalony został zbyt krótki czas dla analizy czasowej, możliwe jest obliczenie brakujących danych bez zaczynania wszystkiego od nowa.

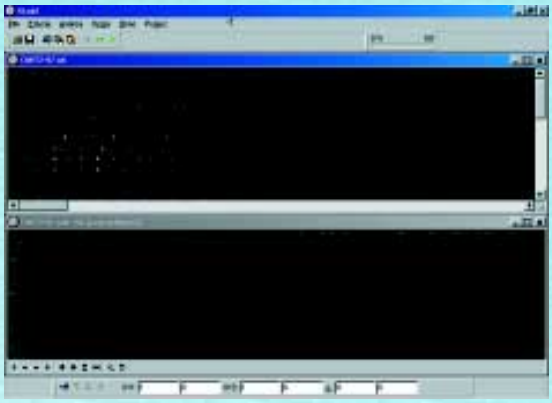
Przeglądanie wyników

Wykonana symulacja nie miałaby dużej wartości, gdyby wyniki nie zostały przedstawione w czytelny sposób. Można je obserwować już w trakcie analizy układu. Są odświeżane co kilka sekund. Jeżeli w liście połączeń dokonamy jakichś zmian, to po ponownej kompilacji projektu XLab zaktualizuje zawartość okien z wykresami. W oknach tych są wyświetlane nie tylko przebiegi napięć i prądów. Można wprowadzić dowolną formułę matematyczną, dzięki czemu znalezienie takich wielkości jak moc, sprawność, wzmocnienie, dobroć czy przesunięcie fazowe nie nastręcza trudności. Rozbudowany edytor formuł pomaga w tym użytkownikowi, wyświetlając listę wszystkich węzłów. Dostępne są podstawowe funkcje matematyczne (pierwiastki, logarytmy, funkcje trygonometryczne i inne). Sterowanie obrazem na ekranie odbywa się podobnie jak w oscyloskopie. Na dole okna znajdują się przyciski do zmiany podstawy pionowej i poziomej, przesuwania oraz powiększania wybranych fragmentów. Wydaje mi się, że dla elektroników jest to wygodniejsze rozwiązanie, niż standardowe paski przesuwu i powiększenia proporcjonalnego („+“ i „-“), znane z typowych programów graficznych. Dla wymagających wysokiej precyzji, XLab oferuje pionowe i poziome kursory pomiarowe, które umożliwiają znalezienie wartości szczytowych oraz pomiar nachylenia zbocza sygnału. Sterowanie nimi odbywa się nie tylko za pomocą myszki, ale również poprzez wprowadzenie dokładnych wartości liczbowych w pasku współrzędnych. Używając funkcji śledzenia można np. sprawdzić, w którym momencie jakieś napięcie przyjmuje konkretną wartość.

Jak to działa?

Jak już wcześniej napisałem, na podstawie listy połączeń w wyniku kompilacji zostaje utworzona w pamięci komputera odpowiednia baza danych, zawierająca informacje o wszystkich użytych elementach. Węzły też są traktowane jako oddzielne elementy. W rzeczywistości w programie każdy element jest obiektem (podobnie jak model - klasą). Na podstawie tej bazy danych wiadomo, które równania zostaną wykorzystane oraz jaka jest ich kolejność. Wszystkie wzorce równań zostały już wcześniej przeze mnie napisane - stanowią integralną część każdego modelu. To w powiązaniach między nimi, realizowanych poprzez wspólne niewiadome, których ustalenie następuje tuż przed uruchomieniem symulacji, zakodowany jest układ. Jednak prawidłowy układ równań nie wystarcza do znalezienia szukanych wartości poszczególnych niewiadomych (potencjałów i prądów). Potrzebne są dodatkowe informacje mówiące w jaki sposób ten układ należy rozwiązać. Komputer musi wiedzieć, jak zmieni się stan równania, tj. różnica wartości strony lewej i prawej (dla uproszczenia prawa jest zawsze równa zero), w przypadku zmiany wartości niewiadomych występujących w tym równaniu. W praktyce oznacza to konieczność podania wzorów na pochodne cząstkowe wszystkich zmiennych użytych w definicji modelu, ponieważ obliczanie pochodnej z ilorazu różnicowego jest niedokładne. Łatwo sobie wyobrazić, co stanie się, gdy w takiej pochodnej jest błąd.

Załóżmy, że zmienna określająca prąd płynący przez opornik ma zbyt dużą wartość i nie jest adekwatna do różnicy potencjałów na końcówkach elementu. Logicznie rozumując należy albo zmniejszyć prąd, albo zwięk-



Rys. 4.

szyć napięcie, by prawo Ohma było spełnione. Gdy podaliśmy nieprawidłowy wzór na pochodną cząstkową, to komputer wykonuje operację przeciwną, czyli zmniejsza napięcie i zwiększa prąd. Równanie staje się jeszcze bardziej „niespełnione" - przyjęte rozwiązanie jest nieprawidłowe.

Prześledźmy dokładnie proces rozwiązywania układu równań nieliniowych. Zastosowałem znany algorytm Newtona. Polega on na iteracyjnym znajdywaniu kolejnych przybliżeń dokładnego rozwiązania zgodnie ze wzorem:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \boldsymbol{d}^{(i)}$$

$$\boldsymbol{d}^{(i)} = [\ \boldsymbol{F}'\ (\boldsymbol{x}^{(i)})\ ]^{-1}\ \boldsymbol{F}\ (\boldsymbol{x}^{(i)})$$

gdzie:

$\boldsymbol{x}^{(i)}$ - wektor aktualnego rozwiązania;

$\boldsymbol{x}^{(i+1)}$ - wektor lepszego, szukanego rozwiązania;

$\boldsymbol{F}\ (\boldsymbol{x}^{(i)})$ - wektor wyrazów wolnych wyliczony dla $\boldsymbol{x}^{(i)}$;

$\boldsymbol{F}'\ (\boldsymbol{x}^{(i)})$ - pierwsza pochodna wektora wyrazów wolnych, czyli inaczej macierz główna układu dla $\boldsymbol{x}^{(i)}$;

$\boldsymbol{d}^{(i)}$ - wektor poprawki do obliczenia.

Zastosowanie tego wzoru sprowadza się do wyznaczenia macierzy odwrotnej układu i przemnożenia jej przez wektor wyrazów wolnych. W praktyce te dwie operacje wykonuje się jednocześnie rozwiązując układ równań liniowych:

$$\boldsymbol{F}'\ (\boldsymbol{x}^{(i)})\ \boldsymbol{d}^{(i)} = \boldsymbol{F}\ (\boldsymbol{x}^{(i)})$$

W pierwszym kroku wyliczona zostaje macierz wyrazów wolnych (n) oraz macierz główna (n^2). Jest ona zbudowana z wartości odpowiednich pochodnych cząstkowych. Każdy wiersz odpowiada jednemu równaniu, każda kolumna - niewiadomej. Natomiast wyrazy wolne stanowią wartość lewych stron równań (strony prawe równe 0) dla aktualnych wartości niewiadomych. Na lekcjach matematyki w szkole średniej omawia się metodę wyznacznikową, jednak ze względu na zbyt dużą złożoność (proporcjonalną do $n!$) nie może być ona zastosowana do rozwiązywania dużych układów. Dlatego wykorzystałem metodę eliminacji Gaussa z częściowym wyborem wyrazu podstawowego, zwaną w skrócie GCW. Polega ona na przekształceniu macierzy kwadratowej w macierz trójkątną górną, na której diagonali znajdują się same jedynki. Przypomina to trochę metodę przeciwnych współczynników (również znaną ze szkoły średniej).

Eliminację rozpoczyna się od pierwszego wiersza. Jeżeli pierwszy wyraz [1, 1] jest równy zero, to poszukuje się takiego wiersza, w którym pierwszy wyraz jest niezerowy. Następnie zamienia się te wiersze miejscami (razem z wyrazami wolnymi). Teraz dzieli się cały pierwszy wiersz przez wyraz [1, 1] i w wyniku wyraz ten przyjmuje wartość równą 1. Tak przygotowany wiersz mnożymy przez [2, 1] i odejmujemy od wiersza drugiego. W efekcie współczynnik [2, 1] zostaje wyzerowany. To samo robi się z pozostałymi równaniami, tj. trzecim, czwartym, itd. Gdy dochodzi się do ostatniego, okazuje się, że cała pierwsza kolumna z wyjątkiem wyrazu [1, 1] została wyzerowana. Operację powtarza się kolejno dla pozostałych kolumn poczynając od [2, 2], [3, 3] i kończąc na [n - 1, n - 1]. Zamiana wierszy miejscami wykonywana jest przez zamianę wskaźników, a nie na danych (3 operacje zamiast 3n).

Gdy macierz zostanie przygotowana, można przystąpić do wyliczenia poprawek. Jeśli początkowe wartości niewiadomych wynosiły 0, to poprawki będą dobrym przybliżeniem rozwiązania. Wyznacza się je począwszy od końca tj. w kierunku od ostatniego wiersza do pierwszego. Wartości poprawek dodaje się do wartości niewiadomych otrzymując dokładniejsze przybliżenie. Cały proces powtarza się kilka razy dla uzyskania wystarczającej precyzji. Złożoność takiego algorytmu jest zdecydowanie mniejsza, zwłaszcza że macierz układu jest macierzą rzadką - większość współczynników to zera.

Obsługa błędów

Czasami niestety zdarza się, że podczas eliminacji jakiś wiersz zostanie całkowicie wypełniony zerami. Wtedy mamy do czynienia z układem nieoznaczonym albo sprzecznym. Świadczy to o błędzie popełnionym przez użytkownika. Sytuacja taka jest zwykle następstwem zwarcia, braku uziemienia albo „wiszącego" węzła i zostaje zasygnalizowana wyświetleniem okienka dialogowego z komunikatem. Trzeba wprowadzić odpowiednie zmiany i skompilować topologię jeszcze raz.

To nie koniec niespodzianek czyhających na użytkownika. W pewnych układach mogą wystąpić problemy ze zbieżnością. Oznacza to, że kolejne iteracje nie dają coraz lepszych rozwiązań, a wartości niewiadomych oscylują w sposób zupełnie przypadkowy. Przyczyną może być silne dodatnie sprzężenie zwrotne albo „pechowa" konfiguracja elementów nieliniowych. Wprawdzie zbieżność metody Newtona jest gwarantowana dla wektora $\boldsymbol{x}$ dostatecznie bliskiego rzeczywistemu rozwiązaniu, jednak nigdy nie da się dokładnie określić, co to znaczy „dostatecznie blisko". Dlatego zastosowałem kilka ciekawych trików niwelujących ten problem. Pierwszym sposobem obrony przed tego typu sytuacjami jest wspomniany wcześniej mechanizm dynamicznego sterowania krokiem obliczeniowym. Jest to uzasadnione domniemaną ciągłością otrzymywanych w symulacji przebiegów. Jeśli przyjmie się dostatecznie mały krok, kolejne próbki nie powinny się od siebie różnić. Z tego powodu rozwiązanie początkowe dla próbki następnej nie powinno leżeć zbyt daleko od rozwiązania prawidłowego i możliwe jest, że algorytm „zaskoczy".

Jeżeli to jednak nie pomoże, to istnieje podejrzenie, iż algorytm „zapętlił się" na lokalnym minimum funkcji $\boldsymbol{F}(\boldsymbol{x})$. Dzieje się tak najczęściej w układach wszelkiego rodzaju przerzutników tuż przed przełączeniem (analiza czasowa). Znika wtedy jedno rozwiązanie, a pojawia się drugie. W miejscu starego zostaje właśnie taki „dołek", z którego ciężko się wydostać. Po paru bezskutecznych iteracjach komputer przypisuje wszystkim niewiadomym wartości losowe i ponawia próbę. Czasami taką operację trzeba powtórzyć dwa albo trzy razy. Ten sposób niestety czasami też zawodzi.

Pozostaje zastosowanie ostatniej deski ratunku - tłumienia. Polega ono na przyjmowaniu mniejszej o 50% poprawki niż wyliczona z eliminacji GCW. Rząd metody gwałtownie maleje, ale szanse na uzyskanie zbieżności rosną.

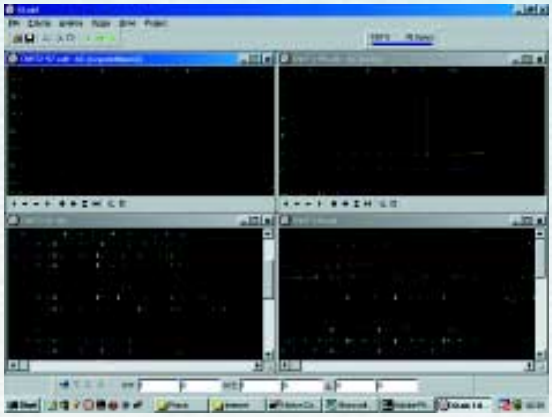
Gdy już wszystkie opisane wyżej metody zawiodą, pozostaje jedynie wyświetlenie przykrego dla użytkownika komunikatu w stylu „Niestety nie potrafię tego rozwiązać... Spróbuj zmodyfikować układ." Należy sprawdzić wtedy przede wszystkim, czy w ustawieniach parametrów analizy nie został popełniony błąd, np. zbyt mała maksymalna liczba próbek lub zbyt wygórowane żądanie dotyczące precyzji ich wyznaczania.

Innych błędów nie należy się zbytnio obawiać - XLab posiada wbudowaną obsługę nieprzewidzianych wyjątków i utracenie danych albo zawieszenie systemu operacyjnego jest prawie niemożliwe.

Jak widać, XLab jest aplikacją o całkiem dużych możliwościach, chociaż ma też i wady. Przede wszystkim brakuje jeszcze wielu funkcji spotykanych w pakietach firm zagranicznych. Na przykład przydałyby się modele elementów cyfrowych, symulacja wpływu temperatury, transformata FFT, import plików symulatora SPICE, analiza szumów czy edytor graficzny dla „nieprogramistów". Program jest w końcu dosyć młody - ma niecałe półtora roku. Mam jednak nadzieję, że uzupełnienie go o te dodatkowe elementy nie zajmie dużo czasu. Może któryś z Czytelników znających Delphi albo C++ chciałby przyłączyć się do pisania? Razem poszłoby szybciej...

Piotr Kołaczkowski
xlab@free.polbox.pl

XLab jest dostępny na stronach:
- *www.elektronika.basnet.pl/programy/programy.html*
- *www.ep.com.pl*
- *oraz na płycie CD-EP5/2000.*



Rys. 5.