Zaproszenie do współpracy dla wszystkich Czytelników EP!

XLAB

Polski symulator układów analogowych

Inicjatywa, którą przedstawiamy w artykule, jest zjawiskiem niespotykanym na naszym rynku elektronicznym: twórca najlepszego polskiego programu do symulacji układów nieliniowych, znanego już z łamów EP - XLAB - proponuje za naszym pośrednictwem wszystkim polskim elektronikom i programistom możliwość współpracy na zasadach zbliżonych do OpenSource.

XLAB jest jedynym zaawansowanym programem do symulacji układów nieliniowych opracowanym w Polsce. Niezwykle istotny jest fakt, że jest to oprogramowanie freeware, a „zespół" twórców składał się zazwyczaj z... jednej osoby. To właśnie jest przyczyną naszego wspólnego apelu - dalszy rozwój tego oprogramowania nie jest możliwy bez powiększenia zespołu twórców. Ponieważ przedsięwzięcie jest z założenia niekomercyjne, autor postanowił złożyć za naszym pośrednictwem propozycję wszystkim programistom i elektronikom, którzy czują się na siłach podjąć wysiłek dalszego rozwijania XLAB-a na zasadach *OpenSource* (przedstawiamy je w dalszej części artykułu).

Naturalnymi adresatami naszej propozycji są przede wszystkim studenci uczelni technicznych, dla których praca nad XLAB-em może być wielką przygodą przygotowującą do profesjonalnego życia po studiach, zapewniającą przy okazji kontakt z nowoczesnymi technikami symulacji układów analogowych.

EP obejmuje patronat medialny nad przedsięwzięciem. Jeżeli znajdą się chętni do współpracy, rozważymy możliwość udzielania także innych form wsparcia osób pracujących nad projektem.

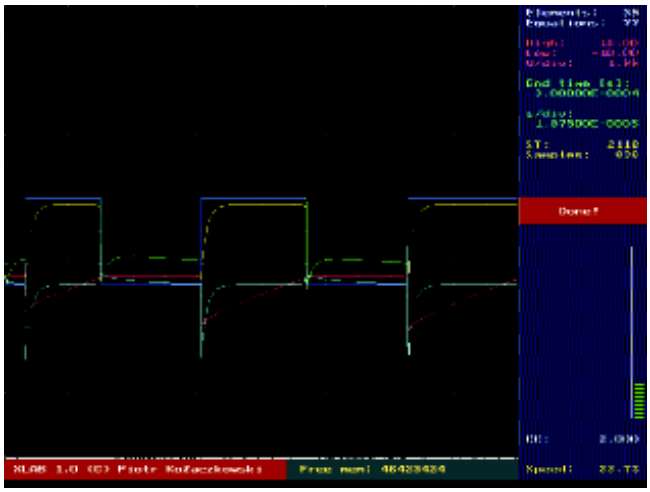
Zaczynamy od przedstawienia historii XLAB-a. Oddajemy głos autorowi programu.

XLAB - historia

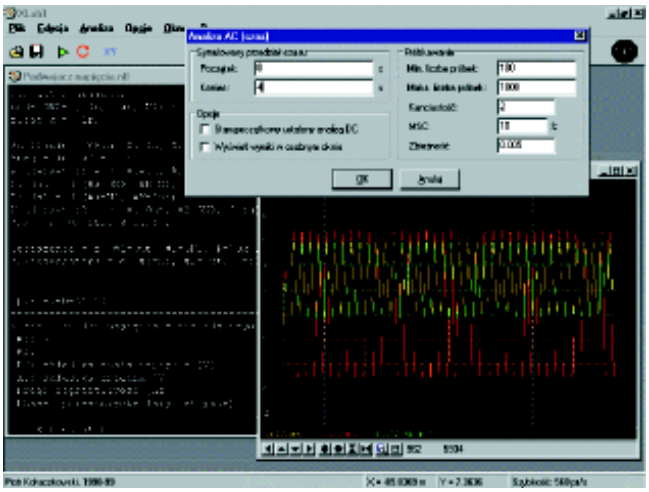
Bardzo często dzieje się tak, że zabierając się za tworzenie „czegoś", po miesiącach pracy otrzymujemy „coś" zupełnie innego niż zamierzaliśmy. Czasem popełniamy błąd, a czasem ciężko jest coś skończyć. Na przykład chcemy zbudować sobie wzmacniacz. Przesadzamy nieco ze wzmocnieniem i otrzymujemy... bardzo dobry generator. Albo montujemy generator i okazuje się, że wystarczy dodać niewiele elementów, aby zrobić syrenę alarmową. Podobna historia przytrafiła się mi, kiedy pisałem „programik" do rozwiązywania równań.

Miał on być z założenia bardzo prosty - podawał rozwiązania równania wprowadzonego bezpośrednio w kodzie, więc jego przydatność do innych zastosowań była znikoma. Każda modyfikacja równania powodowała konieczność rekompilacji całego kodu (ok. 50 linijek w Pascalu). Jednak zagadnienie było na tyle interesujące, że szybko dodałem nową funkcję: rozwiązywanie całych układów.

I wtedy zacząłem się zastanawiać nad możliwymi zastosowa-



Rys. 1



Rys. 2

niami tego „wynalazku". Co można zrobić z kodem, który jest w stanie rozwiązać numerycznie układ ok. 500 równań nieliniowych w mniej niż sekundę? Okazało się, że można i to nawet sporo. Tak powstał...

XLAB 1.00 (listopad 1998)

Wystarczyło dopisać moduł z gotowymi równaniami prostych elementów elektronicznych, żeby powstał symulator układów XLAB 1.0. Pracował w DOS-ie (**rys. 1**), nie posiadał interfejsu wejściowego (definicja układu była „ukryta" w kodzie), rysował wykresy w trybie VGA i rozwijał „zawrotną" prędkość 35 próbek/s przy symulacji multiwibratora astabilnego złożonego z dwóch tranzystorów i kilku elementów dodatkowych. Dla porównania, ten sam układ z dokładniejszymi modelami tranzystorów programy PSPICE i XLAB 2.4 liczą z szybkością ok. 200 próbek/s (testy były wykonywane na Pentium 150MHz).

Były dwie przyczyny tego stanu rzeczy: brak optymalizacji kodu i słaby algorytm. Po przeniesieniu programu na platformę Win32 i rekompilacji w Delphi 2 udało się wyeliminować tę pierwszą i prędkość wzrosła aż 4 razy! A po wprowadzeniu pewnych ulepszeń do algorytmu udało się jego efektywność jeszcze trochę poprawić. Niestety, brak jakiegokolwiek interfejsu użytkownika uniemożliwiał publikację programu.

XLAB 1.04 (lipiec, sierpień 1999)

Konieczne było stworzenie jakiegoś sposobu definiowania układów elektronicznych bez konieczności każdorazowego kompilowania programu. Najprostszy do realizacji okazał się sposób wykorzystany w symulatorach SPICE, czyli definiowanie układów za pomocą opisów w odpowiednim języku. Język opisu układów symulatora XLAB 1.04 różnił się gramatyką od SPICE, ale idea była ta sama. Podawało się listę elementów, określało do jakich węzłów są dołączone oraz jakie są ich parametry. Teraz program nadawał się do analizy niewielkich układów. Niestety, możliwości obliczeniowe symulatora nadal pozostawiały wiele do życzenia. Brakowało przede wszystkim analiz częstotliwościowej i parametrycznej, dobrego algorytmu analizy czasowej i modeli elementów półprzewodnikowych.

XLAB 1.05...1.08 (wrzesień...grudzień 1999)

W tym okresie dodałem brakujące analizy, poprawiłem nieco algorytm analizy czasowej (jak się później okazało - nadal nie działał najlepiej), usprawniłem sposób prezentacji wyników i powiększyłem bibliotekę elementów m.in. o tranzystory JFET oraz włączniki sterowane. Dorzuciłem też gotowy edytor tekstowy z kolorowaniem słów kluczowych, który ściągnąłem z sieci jako darmowy komponent do Delphi. Nie miałem zbyt wiele czasu na pracę nad pakietem, gdyż musiałem równocześnie przygotować się do matury. Dlatego XLAB 1.08 (**rys. 2**) nie był zbyt udanym programem - niezbyt szybki, mało stabilny i niewygodny w obsłudze. Jego opis znalazł się w EP5/2000.

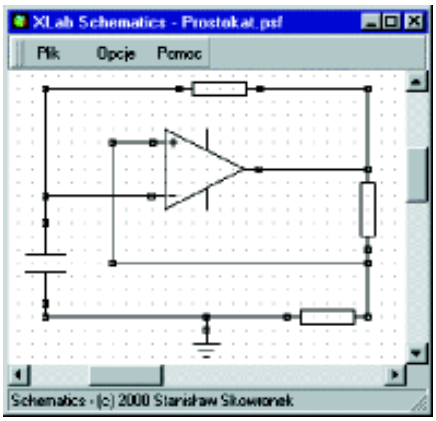
Do największych wad należał z pewnością brak edytora graficznego. Niewielkie układy dość łatwo definiowało się ręcznie, czasem nawet szybciej niż za pomocą niektórych edytorów graficznych, jednak gdy liczba elementów była zbyt wielka, kod stawał się nieczytelny i bardzo podatny na błędy.

XLAB 1.10 i XLab Schematics (styczeń...lipiec 2000)

Dlatego szybko powstał program wspomagający: XLab Schematics (**rys. 3**), którego autorem jest Stanisław Skowronek. Z narysowanego schematu generował listy połączeń wczytywane przez symulator. Mimo swojej prostoty, był bardzo użyteczny. XLAB w tym czasie nie zyskał zbyt wielu nowych funkcji, poprawiłem natomiast kilka drobnych błędów. Wersja 1.10 jest ostatnią z serii XLAB 1.

XLAB 2.0 (styczeń 2001)

W drugiej połowie 2000 r. zacząłem prace nad własnym wektorowym edytorem graficznym. Dostosowałem język opisu układów, aby nadawał się do zapisu grafiki. W ten sposób XLAB zyskał wbudowany, przyjazny użytkownikowi interfejs (**rys. 4**).



Rys. 3

Jest to najbardziej udana i zarazem najbardziej złożona część pakietu. Edytor posiada m.in. następujące właściwości:

- Automatycznie dostosowuje tryb pracy do położenia kursora myszki. Jeśli kursor znajduje się nad końcówką symbolu, to program jest w stanie gotowości do rysowania połączeń. Jeśli nad środkiem symbolu - kliknięcie domyślnie powoduje jego zaznaczenie i ewentualne rozpoczęcie procesu przesuwania. Tryb pracy jest sygnalizowany przez kształt kursora.
- Można zaznaczyć wiele elementów naraz i za jednym zamachem zmodyfikować jakąś wspólną właściwość np. rezystancję. To się na ogół przydaje, gdy w schemacie mamy kilka takich samych elementów.
- Jeśli stwierdziliśmy, że każdy z 30 jednakowych tranzystorów projektowanego układu trzeba wymienić na inny, to wystarczy je wszystkie zaznaczyć i wykonać odpowiednie polecenie (jedno a nie 30!). Podałem przykład ekstremalny, ale funkcja ta jest użyteczna, nawet gdy mamy do czynienia tylko z parą elementów.
- Przy przesuwaniu symboli połączenia są zachowywane. Napisanie tej procedury to była katorga, ale o dziwo działa nad wyraz dobrze.
- Nie trzeba się martwić o kropki na skrzyżowaniu połączeń. Program sam je wstawia i usuwa. Spotkałem edytor, w którym musiał to robić użytkownik!
- Dla tych, co lubią projektować metodą prób i błędów, istnieje wielopoziomowe polecenie „Cofnij/Ponów".
- Nie ma spotykanej w niektórych edytorach „kartki papieru". Prze-

strzeń rysunkowa jest praktycznie nieograniczona. Dlatego nie czeka Cię przesuwanie całego schematu tylko dlatego, że chcesz dostawić element, który wychodzi za margines.
- Wyniki analizy punktu pracy mogą być przedstawiane na schemacie.
- Symbole nie są na stałe zintegrowane z programem, a umieszczone w oddzielnej bibliotece. Możesz tworzyć własne i modyfikować istniejące.

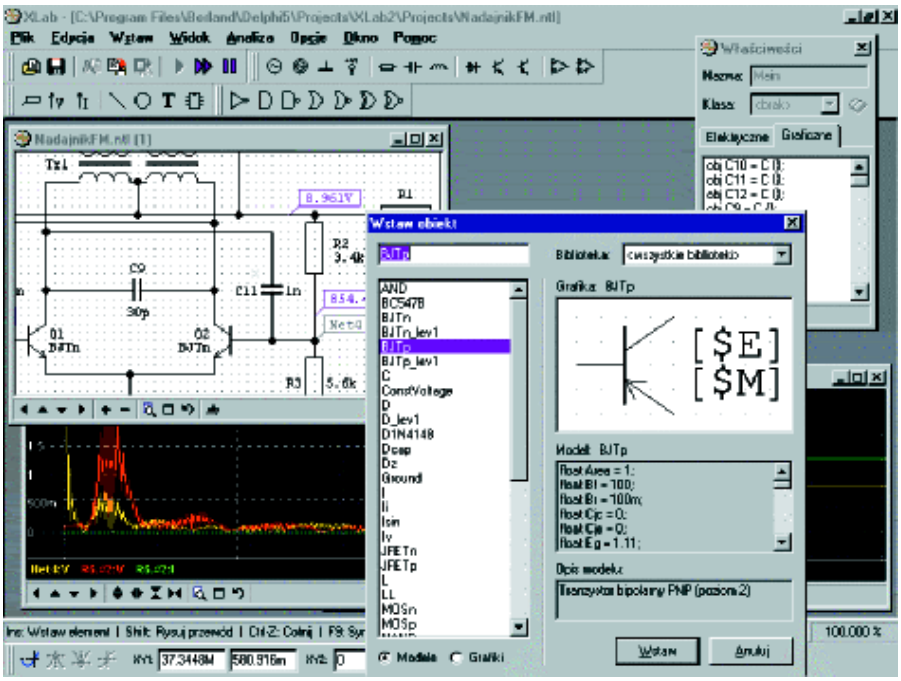
Głównie z tych względów pakiet był przez pewien okres rozpowszechniany przez firmę DGN jako shareware (czyli wypróbuj i ewentualnie kup).

Na początku 2001 r. praca pt. „XLAB - the non-linear circuit simulator" zdobyła III miejsce w Krajowych Eliminacjach Konkursu Prac Młodych Naukowców Unii Europejskiej. Przy okazji tego konkursu kod został wnikliwie oceniony przez Jury i zostały wychwycone pewne merytoryczne usterki:
- Zbyt uproszczony i nieskuteczny algorytm doboru kroku analizy czasowej.
- Niedokładne modele elementów półprzewodnikowych.
- Niewykorzystane wszystkie możliwości przyspieszenia programu.

XLAB 2.2...2.4 (lipiec...wrzesień 2001)

To tylko trochę wzmocniona wersja XLAB-a 2.0. Nie wyeliminowałem wszystkich błędów, dodałem natomiast udogodnienia przydatne na laboratorium z ELIU (Elementy i Układy Elektroniczne).

W międzyczasie zauważyłem również inne wady pakietu:
- Zbyt zawiły sposób definiowania modeli skutecznie spowalnia tworzenie bibliotek. Program nie potrafi sam zlinearyzować modelu (musi mieć gotowe wzory) ani zredukować liczby równań, eliminując równania liniowe.
- Brak możliwości wczytywania plików **.cir* (w formacie PSPICE).
- Program źle radzi sobie z błędami nadmiaru maszynowego, generowanymi przez modele w trakcie symulacji. Ogólnie mechanizmy ochrony przed błędami dosyć często zawodzą, co może być irytujące.
- Istnieje zbyt wiele niejasnych zależności między modułami pakietu. Brak planowania i spontaniczne wprowadzanie zmian spowodowało, że cały projekt stał się bardzo nieczytelny. Utrudnia to wykrywanie błędów i rozbudowę.
- Programu nie da się łatwo przystosować do pracy w systemie Linux, ponieważ został napisany w Pascalu, a dodatkowo występuje w nim dużo wywołań funkcji systemu MS Windows.



Rys. 4

Z tych powodów kontynuowanie rozwoju wersji 2 straciło sens, a całość wraz z kodem źródłowym udostępniłem w Internecie.

XLAB 4.0

To mógłby być koniec historii XLAB-a. Tyle, że licznik ściągnięć na mojej stronie nadal intensywnie „cyka", a dotychczasowi użytkownicy wysyłają e-maile z pytaniami, kiedy będzie następna wersja. Do tego w trakcie 2 lat studiów informatyki nabyłem umiejętności wystarczające do napisania dużo lepszego programu. Dlaczego by ich nie wykorzystać? Z drugiej strony tworzenie takiego oprogramowania pochłania sporo czasu. To nie jest już prosty programik do rozwiązywania zwykłych równań.

Dlatego chciałbym, aby XLAB 4.0 powstawał na podobnych zasadach, na jakich obecnie tworzone jest darmowe oprogramowanie dla systemu Linux (GNU). Po pierwsze, kod będzie pisany równolegle przez kilka osób. Wiele modułów mało zależy od siebie, więc nic nie powinno kolidować. Zespół nie jest ustalony na „sztywno" - każdy może dołączyć w dowolnym momencie. Po drugie, językiem będzie C++, a nie Pascal. Ten język jest bardziej elastyczny, lepiej wspiera zespołowe programowanie i istnieją dobre darmowe kompilatory na wszystkie platformy systemowe. XLAB 4.0 będzie działał zarówno w systemie Windows, jak i Linux, zwłaszcza że na ten drugi bardzo brakuje aplikacji tego typu. Po trzecie, przedsięwzięcie jest niekomercyjne, więc przyszli użytkownicy nie będą musieli płacić za licencję.

Istnieje wstępny projekt architektury nowego pakietu. Cała część symulacyjna zostanie całkowicie przebudowana i wyposażona w nowe, niestosowane dotąd algorytmy. Przede wszystkim ulegnie zmianie sposób definiowania modeli, tak by wszystkie dało się umieścić poza kodem źródłowym. Wymusza to wprowadzenie zmian do języka opisu modeli. Będzie akceptować równania różniczkowo-całkowe dowolnego rzędu i dowolne uzależnianie zmiennych od siebie - np. będzie można uzależnić opór rezystora od dowolnej funkcji napięcia występującego w innym miejscu układu. To rozszerzy zastosowania symulatora na inne dziedziny niż elektronika.

Algorytmy numeryczne będą wspomagane algorytmami symbolicznego upraszczania układu, co powinno zdecydowanie przyspieszyć obliczenia. Wstępne testy z prostym układem LC sugerują bardzo duże zyski efektywności (MicroSim PSPICE: 800 próbek/s, XLAB 2.4: 5000 próbek/s, algorytm optymalny: 250000 próbek/s).

Program ma także lepiej wspierać projektowanie układów cyfrowych i analogowo-cyfrowych. W tym celu zastosowane będą techniki symulacji sterowanej zdarzeniowo.

Do symulatora będzie dołączony optymalizator wielowymiarowy - narzędzie bardzo użyteczne, bo pozwala na automatyczne dobieranie parametrów układu metodą prób i błędów w celu spełnienia odpowiednich ograniczeń. Proces ten trwa niestety dość długo, ale choć komputer nie dysponuje intuicją niejednego inżyniera, to zwykle znajduje lepsze rozwiązania. A inżynier może w tym czasie wyjść na kawę...

Jeśli chodzi o grafikę, to rewolucji nie będzie. Obecny edytor jest bardzo dobry i wszystkie zmiany ograniczą się do poprawek kosmetycznych. Trzeba będzie jedynie dopisać konwertery znanych formatów rysunkowych oraz umożliwić współpracę z edytorami płytek drukowanych. Możliwe, że uda się też napisać własny autorouter. Spore zmiany natomiast pojawią się w części prezentacji wyników. XLAB 2.4 wypada pod tym względem bardzo słabo na tle np. popularnego PROBE-a.

To jest złożony projekt i obejmuje wiele dziedzin naraz, ale mam nadzieję, że znajdą się chętni do podjęcia tego wyzwania. Nie jest potrzebna nie wiadomo jaka znajomość C++ i matematyki wyższej (choć to na pewno by pomogło). Liczy się zapał, kreatywność i chęć poznawania rzeczy nowych.

Wszelkie szczegóły znajdują się na stronie *www.xlab.prv.pl*, a zgłoszenia należy przesyłać na adres *pkolaczk@elka.pw.edu.pl*.

Piotr Kołaczkowski

Definicja oprogramowania *open source*

(za www.opensource.org)

Wprowadzenie

Open source nie znaczy tylko dostępu do kodu źródłowego. Warunki dystrybucji oprogramowania open source muszą być zgodne z następującymi kryteriami:

1. **Swoboda redystrybucji**
   Licencja nie może ograniczać swobody którejkolwiek ze stron do sprzedawania lub rozdawania oprogramowania jako elementu szerszej dystrybucji zawierającej programy z różnych źródeł. Licencja nie może wymagać pobierania honorariów lub innych opłat od takiej sprzedaży.
2. **Kod źródłowy**
   Do programu musi być dołączony kod źródłowy, a licencja musi zezwalać na dystrybucję zarówno w postaci kodu źródłowego, jak i skompilowanej. Jeśli któryś produkt nie jest rozprowadzany wraz z kodem źródłowym, musi istnieć dobrze udokumentowany sposób uzyskania tego kodu źródłowego za cenę nieprzekraczającą rozsądnych kosztów wykonania kopii - najlepiej poprzez darmowe pobranie z Internetu. Kod źródłowy musi być dostępny w zalecanej postaci, pozwalającej na prostą modyfikację. Nie jest dozwolone celowe gmatwanie kodu źródłowego. Formaty pośrednie, takie jak wynik działania preprocesora lub translatora, nie są dozwolone.
3. **Dzieła pochodne**
   Licencja musi zezwalać na dokonywanie zmian oraz tworzenie dzieł pochodnych. Musi również umożliwiać dystrybucję takich dzieł na tych samych warunkach, jakie opisuje licencja oryginalnego oprogramowania.
4. **Spójność kodu źródłowego autora**
   Licencja może ograniczać dystrybucję kodu źródłowego w zmodyfikowanej postaci tylko wtedy, jeśli dozwolona jest przy tym dystrybucja "poprawek" (ang. patch) wraz z kodem źródłowym, za pomocą których program jest potem modyfikowany w trakcie kompilacji. Licencja musi jawnie zezwalać na dystrybucję oprogramowania skompilowanego ze zmodyfikowanego kodu źródłowego. Licencja może wymagać, aby dzieła pochodne nosiły inną nazwę lub numer wersji niż oprogramowanie oryginalne.
5. **Niedozwolona dyskryminacja osób i grup**
   Licencja nie może dyskryminować jakichkolwiek osób czy grup.
6. **Niedozwolona dyskryminacja obszarów zastosowań.**
   Licencja nie może zabraniać wykorzystywania programu w jakimś konkretnym obszarze zastosowań. Na przykład, nie może zabraniać wykorzystania programu w sposób komercyjny lub używania go do badań genetycznych.
7. **Dystrybucja licencji**
   Określenie praw dołączone do programu musi obowiązywać wszystkich, którzy otrzymują oprogramowanie bez konieczności przestrzegania przez te osoby dodatkowych licencji.
8. **Licencja nie może obejmować konkretnego produktu**
   Określenie praw dołączone do programu nie może zależeć od tego, że dany program stanowi część określonej dystrybucji oprogramowania. Jeśli program został pobrany z takiej dystrybucji i wykorzystywany lub rozprowadzany jest zgodnie z warunkami licencji, wszystkie osoby, do których program trafia, powinny posiadać te same prawa, które określone są dla oryginalnej dystrybucji oprogramowania.
9. **Licencja nie może ograniczać stosowania innego oprogramowania**
   Licencja nie może nakładać ograniczeń na inne oprogramowanie rozprowadzane wraz z oprogramowaniem objętym licencją. Na przykład, nie może wymagać, aby wszystkie inne programy rozprowadzane na tym samym nośniku były programami open source.