ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: ELEKTRONIKA z.5 Nr kol. 1341

Jerzy **DĄBROWSKI**
Instytut Elektroniki
Politechniki Śląskiej

ALGORYTMY SYMULACJI HYBRYDOWEJ

Streszczenie. Praca stanowi przegląd algorytmów różnych rodzajów symulacji znajdujących zastosowanie w symulacji hybrydowej. Techniki symulacji dyskretnej i ciągłej zostały przedstawione pod kątem ich implementacji w symulatorach hybrydowych. Zamieszczone zostały przykłady dwu konkretnych rozwiązań symulatorów hybrydowych. Praca bazuje częściowo na doświadczeniach wyniesionych z prac w zakresie symulacji układów, prowadzonych od kilku lat w Instytucie Elektroniki Politechniki Śląskiej.

ALGORITHMS OF MIXED-MODE SIMULATION

Summary. In the paper an overview of different simulation techniques exploited in mixed-mode simulation is presented. Discrete- and continuous simulation are discussed in context of their use in the mixed-mode simulation. Detailed examples of two particular mixed-mode simulators are included. Some comments in this paper are based on the experience gained while dealing with network simulation at the Institute of Electronics of Silesian Technical University since a last few years.

ALGORITHMEN VON HYBRIDSIMULATION

Zusammenfassung. In diesem Aufsatz eine Übersicht von verschiedenen Simulationalgorithmen, die in Hybridsimulation verwendbar sind, wird angegeben. Diskrete und stetige Simulation werden diskutiert als Bauteile für eine Hybridsimulatorkonstruktion. Die Beschreibung von zwei praktischen Hybridsimulatoren wird angeschlossen als ein Beispiel. Zum Teil basiert der Aufsatz auf einer Erfahrung mit der Simulation, gewonnen in letzten Jahren während der Arbeit im Institut für Elektronik an der Technische-Universität von Oberschlesien.

1. Wprowadzenie

Technika symulacji jest aktualnie powszechnie wykorzystywana w procesie projektowania układów elektronicznych. Programy symulacyjne jako narzędzia weryfikacji są w stanie dostarczyć szczegółowych informacji o czasowych efektach wewnątrz struktury układu, pozwalając jednocześnie przewidywać jego ewentualnie błędne działanie. Szczególna rola przypada symulacji podczas projektowania skomplikowanych układów scalonych, gdy realizacja kosztownych procesów technologicznych powinna być poprzedzona weryfikacją na etapie projektu. Istotne znaczenie posiada wówczas nie tylko złożoność samej struktury i stopień jej skomplikowania, ale również wpływ elementów pasożytniczych. To ostatnie zjawisko w odniesieniu do technologii submikronowej może mieć nawet dominujący wpływ na czasy propagacji sygnałów.

Ze względu na rozwój scalonych układów dużej skali integracji symulacja obwodowa (elektryczna), której silny rozwój przypada na początek lat siedemdziesiątych, stosunkowo szybko przestała być wystarczająca. Na bazie klasycznej symulacji elektrycznej powstały, bardziej oszczędne obliczeniowo, nowe techniki, określane jako symulacja czasowa. Wzrosło także zapotrzebowanie na narzędzia symulacyjne, obejmujące wyższe poziomy abstrakcji opisu układów: poziom bramkowy, poziom kluczy (charakterystyczny dla technologii MOS, ang.: switch-level), poziom przesłań międzyrejestrowych (ang.: RTL) oraz poziom architektury (zwany również behawioralnym). Tendencja ta odzwierciedlała równocześnie rozwój hierarchicznej strategii w projektowaniu układów (ang.: top down), w którym punkt wyjścia stanowi określenie architektury i jej opis behawioralny. Symulatory wyższych poziomów abstrakcji [6] pozwalają skutecznie wspomagać proces weryfikacji projektu w jego kolejnych etapach, a względny koszt symulacji przy ich użyciu rośnie wykładniczo ze wzrostem szczegółowości opisu. Z drugiej strony jednak, elektryczna symulacja obwodowa (dla której nieformalnym standardem jest program SPICE2) i pokrewne jej techniki czasowe są jedynymi narzędziami, które dostarczają wystarczająco dokładnych charakterystyk, aby móc zweryfikować wymagania projektowe w pełnym zakresie warunków pracy układu.

Zestawienie cech charakterystycznych poszczególnych rodzajów symulacji przedstawia tab.1 [1]. Na tym tle symulacja techniką mieszaną (ang.: mixed-mode

simulation), czyli symulacja hybrydowa [1,2], może być widziana dwojako. W odniesieniu do układów A/C symulacja hybrydowa stanowi podejście w pełni naturalne, w którym dla części cyfrowej wykorzystuje się techniki symulacji logicznej (np. bramkowej lub RTL), a dla analogowej - techniki symulacji elektrycznej (obwodowej lub tzw. czasowej). Natomiast wykorzystanie symulacji hybrydowej przy projektowaniu układów cyfrowych jest wyrazem kompromisu, polegającego na tym, że krytyczne fragmenty struktury są symulowane na wysokim poziomie szczegółowości (najczęściej elektrycznym), a pozostałe jedynie bardziej zgrubnie. Dotyczy to projektów, w których np. na poziomie bramkowym lub kluczy trzeba uwzględnić takie zjawiska jak: dwukierunkowość tranzystorów transmisyjnych, podział ładunku czy wpływ elementów pasożytniczych, bez konieczności kosztownego symulowania całego układu na poziomie elektrycznym. Podobnie, podczas projektowania, w którym poszczególne fragmenty układu powstają niezależnie, na różnych poziomach abstrakcji, ich projektanci w momencie złożenia całości często wolą weryfikować go bez zmiany poziomu opisu. Symulacja taka jest wówczas określana dodatkowo jako wielopoziomowa (ang.: multi-level simulation lub mixed-level simulation).

Tabela 1

Rodzaj symulacji	Względny koszt	Możliwości i dokładność
Behawioralna (B)	1	Weryfikacja algorytmów, uproszczone zależności czasowe
RTL ®	10	Weryfikacja funkcjonalna, uproszczone zależności czasowe
Bramkowa (G)	100	Weryfikacja funkcjonalna, przybliżone zależności czasowe
Kluczowa (S)	1000	Weryfikacja funkcjonalna, przybliżone zależności czasowe
Czasowa (T)	10000	Szczegółowe przebiegi czasowe sygnałów o ograniczonej dokładności
Elektryczna (E)	1000000	Szczegółowe przebiegi czasowe sygnałów, najdokładniejszy rodzaj symulacji

W niniejszej pracy uwaga jest skupiona na technikach symulacji związanych z niższymi poziomami abstrakcji, tj. elektrycznym, czasowym, bramkowym oraz kluczy. W symulatorach hybrydowych, obejmujących wymienione poziomy, ciągła z

natury symulacja elektryczna lub czasowa wymaga połączenia z dyskretną symulacją logiczną. Symulatory takie dostarczają przede wszystkim szczegółowych zależności czasowych między sygnałami w układzie. Równocześnie jednak stopień skomplikowania i skala trudności dotyczące ich realizacji są znacząco większe niż dla symulatorów obejmujących poziomy architektury, przesłań międzyrejestrowych lub poziom bramkowy.

Praca posiada charakter przeglądowy, ale nie pretenduje, aby uznać ją za w pełni wyczerpującą poruszany temat. Część przedstawionych uwag wynika bezpośrednio z doświadczeń zdobytych podczas prac w zakresie symulacji, prowadzonych w ostatnich latach w Instytucie Elektroniki Politechniki Śląskiej [55-67]. Ze względu na ramy pracy w większości opisów prezentowanych technik pominięto zagadnienia dotyczące modeli, ograniczając się do niezbędnego minimum, wymaganego jedynie dla zrozumienia mechanizmów symulacji.

2. Geneza symulacji hybrydowej

Symulacja logiczna i elektryczna zostały zapoczątkowane w latach pięćdziesiątych wraz z upowszechnieniem pierwszych komputerów. Przez wiele lat symulacja logiczna układów cyfrowych [6,17] i symulacja elektryczna układów analogowych rozwijały się oddzielnie. Rozwój technologii układów scalonych w latach sześćdziesiątych i siedemdziesiątych przyspieszył rozwój technik symulacyjnych, szczególnie przeznaczonych do analizy układów analogowych zawierających nieliniowości [20].

W roku 1967 udostępniony został, opracowany przez IBM, symulator SCEPTRE, umożliwiający analizę obwodów do 200 gałęzi. W latach 1973-75 na Uniwersytecie Berkeley opracowano symulator SPICE [21]. Zaletą tego programu były dokładne modele diod i tranzystorów, a także dostępność nowych rodzajów analizy: wrażliwościowej i temperaturowej. U podstaw rozwoju symulatorów elektrycznych w latach siedemdziesiątych stanęły początkowo stabilne numerycznie metody niejawnego całkowania oraz technika macierzy rzadkich, a następnie metody dekompozycyjne, wśród których największą rolę odegrały techniki relaksacyjne. Pierwszy symulator relaksacyjny, MOTIS, nazwany symulatorem czasowym, powstał w roku 1975 w Bell Labs. Zorientowany na silnie rozwijające się cyfrowe układy MOS, pozwalał ok. 100-

krotnie przyspieszyć analizę w porównaniu do klasycznej symulacji elektrycznej (chociaż w sposób przybliżony).

Rozwój systemów cyfrowych i wzrost skali integracji cyfrowej układów scalonych sprawiły, że proste bramkowe symulatory logiczne przestały być w projektowaniu wystarczające. Niezbędne stało się precyzyjniejsze odwzorowanie zależności czasowych poprzez zróżnicowanie opóźnień oraz uwzględnienie pośrednich stanów logicznych, a także stan wysokiej impedancji. Synchroniczną symulację kompilowaną [17] zastąpiła asynchroniczna symulacja sterowania zdarzeniami [41] (ang.: event-driven simulation). Idea sterowania zdarzeniami wkrótce została zaszczepiona na grunt innych rodzajów symulacji [25].

Specyfika cyfrowych układów MOS, polegajaca m.in. na tym, że rolę najprostszego elementu logicznego, tj. klucza, może spełniać pojedynczy tranzystor, sprawiła, że na początku lat osiemdziesiątych powstały odrębne techniki, zwane symulacją na poziomie kluczy (ang.: switch-level simulation). Pierwsze symulatory tej grupy były zorientowane na symulację logiczną [28,29]. Wysiłek późniejszych prac z tego zakresu był skupiony na uwzględnieniu dokładniejszych relacji czasowych w strukturach MOS, przez co symulatory te były zorientowane elektrycznie lub czasowo [22,42].

Patrząc nieco ogólniej, związane z różnymi poziomami abstrakcji techniki symulacyjne dyskretna i ciągła w toku swego rozwoju wpływały na siebie wzajemnie. W dążeniu do poprawy dokładności symulatory logiczne upodobniły się częściowo do analogowych przez zwiększenie liczby stanów (logiki wielowartościowe) oraz możliwość odwzorowania dowolnych opóźnień, także w funkcji obciążenia i sterowania. Podobnie, niektórym symulatorom ciągłym zaszczepiono cechy dyskretnej symulacji logicznej przez zdefiniowanie tzw. zdarzeń analogowych. Inspiracja taka powstała pod koniec lat siedemdziesiątych przy pracy nad symulatorem SPLICE, pierwszym symulatorem hybrydowym [25]. Obejmował on technikę symulacji elektrycznej, czasowej i bramkowej. Późniejsze wersje tego symulatora obejmowały także poziom kluczy i przesłań międzyrejestrowych.

Obecnie właściwie we wszystkich symulatorach hybrydowych obejmujących niższe poziomy abstrakcji opisu, a także w symulatorach dużych struktur wykorzystujących jedną technikę istotną rolę odgrywa metoda pośredniej dekompozycji oparta na

relaksacji [23,4]. W latach dziewięćdziesiątych tendencja stosowania relaksacyjnych algorytmów w symulacji (Gaussa-Jacobiego) zbiegła się z możliwością wykonywania obliczeń równoległych przy pomocy komputerów wieloprocesowych.

Równocześnie, silny rozwój w dziedzinie scalonych układów A/C, w tym również układów specjalizowanych (ASIC), ugruntował pozycję symulatorów hybrydowych wśród narzędzi projektowych. Przykładowo, w USA w 1994 r. istniało ok. 20 firm oferujących symulatory dla układów A/C lub układów cyfrowych w oparciu o technikę mieszaną [14].

Pracom w zakresie symulacji towarzyszyły prace nad językami HDL, tj. opisu sprzętu. Obecną tendencją w tym zakresie jest dążność do stworzenia standardu dla opisu struktur analogowych i analogowo-cyfrowych w oparciu o specjalistyczny język A-HDL [47]. Chociaż prace te są w toku, niektórzy ze wspomnianych dostawców oprogramowania już zaproponowali gotowe rozwiązania.

3. Charakterystyka symulacji ciągłej i dyskretnej

Konwencjonalna, ciągła symulacja analogowa posiada cztery zasadnicze aspekty: układanie równań, numeryczne całkowanie, linearyzację równań oraz ich rozwiązanie. Szczegółowy opis tych metod jest dostępny w literaturze (np.[8,9]). Tutaj zasygnalizowane zostaną jedynie aspekty tej metodologii. Interesujące w symulacji analogowej równania związane są z prawami Kirchhoffa oraz modelami elementów obwodu takich jak: elementy RLC, diody i tranzystory, źródła sygnałów i zasilania, a także linie transmisyjne. Stosując jedną z metod: analizy węzłowej, zmodyfikowanej analizy węzłowej lub technikę macierzy, można dla obwodu uzyskać układ nieliniowych równań różniczkowych w postaci:

$$F\left[\frac{dx(t)}{dt},\ x(t),\ u(t)\right] = 0 \qquad x(0) = x_0 \tag{1}$$

gdzie $x(t)$ jest wektorem niewiadomych (napięć, prądów, ładunków lub strumieni), $u(t)$ jest wektorem niezależnych źródeł wymuszających (prądowych lub napięciowych), a x_0 oznacza wektor warunków początkowych.

Rozwiązania tych równań, uzyskane w procesie dyskretyzacji czasu, stanowią przybliżenie dokładnych rozwiązań ciągłych. Dokładność tego procesu jest sterowana długością czasowego kroku całkowania poprzez kontrolę tzw. lokalnego błędu obcięcia. Długość kroku całkowania jest związana z chwilowymi zmianami aktywności obwodu, tzn. że podczas wolnych zmian sygnałów może być ona zwiększona i odwrotnie - podczas szybkich zredukowana do niezbędnego minimum.

Z kolei, układy cyfrowe ze względu na ustalony zakres zmian swoich sygnałów i relatywnie krótkie czasy odpowiadające przełączaniu mogą być opisane równaniami dyskretnymi w postaci:

$$l_{n+1} = Q(l_n + e_n) \tag{2}$$

gdzie l_{n+1} i l_n są wektorami logicznych stanów układu w chwilach czasu t_{n+1} i t_n, natomiast e_n stanowi wektor logicznych pobudzeń w chwili t_n. Dyskretny krok czasu $h_n=t_{n+1}-t_n$ jest w najprostszym przypadku stały. Wśród elementów wektora l można dokonać ścisłego podziału na sygnały wejściowe i wyjściowe, odniesione do wewnętrznych elementów struktury (np. bramek). Oznacza to także, że przepływ sygnałów logicznych jest ściśle jednokierunkowy. Ponadto sygnały wyjściowe są opóźnione w stosunku do wejściowych, co wyraża związek przyczynowo-skutkowy między nimi.

Ze względu na dążność do precyzyjnego odwzorowania zależności czasowych w układzie boolowski zbiór wartości stanów logicznych jest często poszerzony o stany pośrednie, np. narastania, opadania, stany wysokiej impedancji lub stany nieokreślone (związane z inicjalizacją układu lub konfliktami sygnałów).

Większość wymienionych cech symulacji logicznej przeciwstawia ją symulacji analogowej, w której, po pierwsze, uzyskane sygnały trzeba uznać za ciągłe w czasie, po drugie sygnałów w obwodzie, w większości przypadków, nie można podzielić na wejściowe i wyjściowe, a przepływ uznać za jednokierunkowy (z powodu zjawiska obciążenia). Trudniejsze są także do formalnego wyrażenia związki przyczynowo-skutkowe, gdyż typowe układy ciągłe działają bez zwłoki.

Mimo znaczących różnic wzrost skali integracji projektowanych obwodów elektronicznych z jednej strony, a z drugiej duża efektywność obliczeniowa symulacji logicznej stały się inspiracją do modyfikacji metod ciągłych. Dodatkowy impuls sta-

nowił także rozwój układów MOS, których elementy można w przybliżeniu uznać za wzajemnie się nie obciążające. Innym, istotnym czynnikiem stojącym u podstaw nowej generacji ciągłych i hybrydowych algorytmów symulacyjnych było przyjęcie z symulacji dyskretnej techniki tzw. śladu selektywnego i połączenie jej z dekompozycją opartą na metodach relaksacyjnych. Najbardziej znaczące efekty tych prac przedstawione są w kolejnych rozdziałach.

4. Technika i algorytm symulacji dyskretnej

Dyskretna symulacja logiczna jest historycznie związana z symulacją bramkową oraz symulacją na poziomie przesłań międzyrejestrowych. Symulatorom dyskretnym tego typu, wchodzącym w skład symulatora hybrydowego, stawia się szczególnie duże wymagania odnośnie do ich dokładności, jeśli obejmuje on niższe poziomy abstrakcji. Wymagania te w istocie dotyczą przyjętej logiki (liczby stanów) oraz modelu opóźnień (uzależnień czasowych). Połączenie symulacji logicznej, dającej bardzo „zgrubne" zależności czasowe w układzie ze znacząco dokładniejszą symulacją czasową lub elektryczną, może być niecelowe.

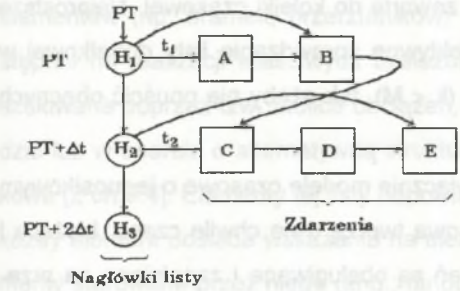
W niniejszym rozdziale omówione zostaną zasadnicze mechanizmy i organizacja symulatorów logicznych. Jak już wspomniano, zagadnienia te stoją także u podstaw symulatorów ciągłych nowej generacji oraz symulatorów hybrydowych.

Symulacja logiczna opiera się aktualnie na kilku charakterystycznych koncepcjach [41,7], [2 ch.9], [1 ch.5, ch.7]. Pierwsza z nich polega na sterowaniu procesem symulacji w oparciu o wspomniane już zdarzenia (ang. event-driven). Zdarzenie w symulacji oznacza zmianę stanu sygnału w danym węźle układu, która może wpłynąć na wywołanie zmian w sąsiednich węzłach. Innymi słowy, zdarzenie wywołuje pobudzenie elementów logicznych sterowanych z danego węzła (ang. fanout).

Drugą ideą jest analizowanie jedynie aktywnych fragmentów układu uzyskane przez śledzenie toru rozprzestrzeniania się zdarzeń. Ta cecha symulacji znana jest pod nazwą metody śladu selektywnego (ang. selective-trace).

Realizacja efektywnego symulatora posiadającego wymienione cechy opiera się z reguły na dwóch podstawowych mechanizmach: kolejce czasowej (ang. time-queue) oraz harmonogramowaniu zdarzeń (ang. event scheduling).

4.1. Kolejka czasowa i technika harmonogramowania



Rys. 1. Kolejka czasowa jako lista indeksowa
Fig. 1. Time queue as an indexed list

Kolejka czasowa, przedstawiona na rys.1, przyjmuje najczęściej postać indeksowanej listy lub tablicy [1ch.7.2], której elementy (nagłówki) wskazują na właściwe listy zdarzeń dla kolejnych chwil czasu. Każdy element kolejki odpowiada dokładnie jednej chwili czasowej. Harmonogramowanie danego zdarzenia, aktywnego w chwili t_i, polega na wprowadzeniu go na listę zdarzeń wskazywaną przez element kolejki, określony przez t_i. Wprowadzenie na listę jest ułatwione dzięki dodatkowemu wskazaniu na ostatni element listy. Lista elementów może być zorganizowana zarówno w postaci stosu (LIFO), jak i kolejki (FIFO) na ogół bez istotnego wpływu na efektywność symulacji. Jeśli zdarzenia są harmonogramowane tylko dla całkowitych wielokrotności podstawowego kroku czasowego Δt, to ze względu na dostęp bardziej efektywna od listy jest struktura tablicy. Jeśli PT odpowiada bieżącej chwili symulacji, wówczas zdarzenie aktywne w chwili PT+iΔt będzie wprowadzone na listę dostępną przez wskaźnik czasu odległy od PT o i jednostek. Długość kolejki czasowej musi być ze względów praktycznych ograniczona. Klasyczne rozwiązanie stanowi w tym przypadku zamknięcie kolejki czasowej w pętlę, zwaną kołem czasowym [41]. Istota tego rozwiązania polega na tym, że elementy kolejki czasowej mogą być wykorzystane ponownie, gdy wszystkie zdarzenia odpowiadające danej chwili zostaną obsłużone, a wskaźnik PT zwiększony o jeden. Wskaźnik ten może być wówczas uaktualniony w oparciu o następujący wzór:

$$PT = (PT + 1) \bmod M,$$

gdzie M oznacza liczbę chwil na kole czasowym. M wybiera się tak, aby w toku symulacji móc harmonogramować także odleglejsze w czasie zdarzenia, odpowiadające maksymalnemu czasowi propagacji dla elementów w układzie.

Zdarzenia bardziej odległe (na ogół pobudzenia zewnętrzne) są umieszczone na dodatkowej liście (ang. remote list). Lista ta wymaga takiej obsługi, aby w odpowiedniej chwili przenieść zdarzenia w niej zawarte do kolejki czasowej. Najprostsze rozwiązanie stanowi w tym przypadku efektywne sprawdzanie listy dodatkowej w każdym kroku czasowym lub co k kroków (k < M), tak ażeby nie opuścić obecnych tam zdarzeń.

Jeśli w symulacji wykorzystywane są wyłącznie modele czasowe o jednostkowym opóźnieniu, to wystarczającą kolejkę czasową tworzą dwie chwile czasu: bieżąca i następna, a odpowiadające im listy zdarzeń są obsługiwane i zapełniane na przemian. Jednak wykorzystywany w tym przypadku czysto logiczny model układu nie jest przydatny w symulacji hybrydowej.

W istocie rzeczy sterowaną zdarzeniami symulację można prowadzić wg jednego z dwu trybów [2 ch.9.5]. Są nimi:

* tryb synchroniczny, w którym dyskretny czas symulacji jest inkrementowany ze stałym krokiem, zdarzenia są harmonogramowane tylko w tak określonych chwilach czasu,
* tryb asynchroniczny, w którym zdarzenia mogą pojawiać się i być harmonogramowane w dowolnych chwilach czasu, zależnych wyłącznie od opóźnień charakteryzujących modele bramek.

Dla symulacji w trybie synchronicznym najodpowiedniejsza wydaje się być tablicowa struktura kolejki czasowej zorganizowanej w koło. Tryb asynchroniczny natomiast najkorzystniej jest implementować w strukturze listy, której elementy (rekordy) określają uporządkowane, aktywne chwile czasu. Zaharmonogramowanie nowego zdarzenia aktywnego E_j w chwili t_k oznacza w tym przypadku dodanie nowego elementu do kolejki czasowej (uporządkowanej listy), a następnie wpisanie E_j na wskazywaną (nową) listę zdarzeń lub jeśli element odpowiadający t_k już istnieje, to jedynie dopisanie E_j do właściwej listy. Ponieważ w symulacji asynchronicznej następna chwila nie jest z góry znana, jest to więc tzw. symulacja „następnego zdarzenia" (ang. next-event simulation), w której na ogół kolejne chwile nie są równo odległe.

4.2. Obsługa zdarzeń i algorytm symulacji na poziomie bramek

Obsługa danego zdarzenia polega w istocie na wyznaczeniu logicznego stanu elementów (np. bramek, przerzutników) uaktywnionych przez to zdarzenie, a następnie na realizacji właściwych uzależnień czasowych. Aktywne elementy są odszukiwane poprzez tzw. tablice obciążeń, wskazujące na strukturę połączeń w układzie lub w oparciu o alternatywną strukturę danych, jaką może być lista dwukierunkowa [2 ch.9.4]. Elementy tej listy odpowiadają elementom logicznym w układzie lub każdy element posiada wskazanie na elementy sterujące go (ang.: fanins) oraz elementy sterowane przez niego (ang.: fanouts). Struktura danych w postaci listy może służyć równocześnie do przechwytywania informacji o aktualnym stanie poszczególnych elementów oraz o stanach poprzednich (tj. historii). Znajomość stanów poprzednich danego elementu logicznego jest istotna co najmniej z dwu powodów: aby móc wykryć zaistnienie zdarzenia oraz aby stwierdzić, czy zostały naruszone warunki czasowe, np. określające czas trzymania, czas ustalania lub zbyt wąskie dla propagacji impulsy czasowe. Ta ostatnia sytuacja wiąże się z procesem przeciwnym do harmonogramowania, czyli usunięciem „pretendującego zdarzenia" (ang.: pending event) z odpowiedniej listy oraz przygotowaniem właściwego komunikatu. W podobny sposób mogą być także przygotowywane komunikaty, np. o wyścigu na wyjściach.

Generacja zdarzenia jest poprzedzona pobraniem stanu wejść elementu w bieżącej chwili, określeniem stanu wyjścia, a następnie opóźnieniem go zgodnie z modelem. Jeśli opóźnienia wnoszone od poszczególnych wejść są różne, to sygnały wejściowe muszą być opóźnione przed wyznaczeniem stanu wyjścia.

Dla kombinacyjnych elementów o niewielkiej liczbie wejść stan wyjścia dogodnie jest wyznaczać w oparciu o tablice prawdy. Jednak w przypadku bardziej skomplikowanych elementów (nawet przerzutników) znacznie efektywniejsze jest opisanie działania elementu za pomocą procedury.

Typową strukturę algorytmu symulacji opartej na wymienionych mechanizmach można w uproszczony sposób przedstawić następująco [1]:

```
tn←0;
while (tn ≤ Tstop) {
  tn← następne _ zdarzenie. czas (tn);
  foreach (wejście k dla tn)
    if (ek jest aktywne)
      forall (j ∈ Fanout (k) harmonogramuj (węzeł j dla tn);
  /* przetwarzanie bloku i */
  foreach (węzeł i dla tn) {
    pobierz _ stany _ wejść;
    oblicz _ nowy _ stan _ wyjścia;
    if (węzeł i zmienił stan) {
      wyznacz _ opóźnienie Δti;
      if (aktualny _ czas < poprzednie _ zdarzenie. czas(i) + Δti) {
        wypowiedź _ komunikat glitch;
        usuń _ oczekujące _ zdarzenia;
      }
      else {/* normalne zdarzenie, harmonogramowanie bloków sterowanych */
        forall (węzeł k ∈ Fanout(i)) {
          oblicz _ czas _ harmonogramowania Δtk;
          harmonogramuj (węzeł k, tn + Δtk);
        }
      }
    } else {/* brak akcji (wykorzystanie nieaktywności) */}
  }
```

Pierwszą czynnością w pętli czasowej jest określenie chwili t_n wystąpienia kolejnego zdarzenia. Wszystkie zewnętrzne węzły pobudzające **k**, które zmieniają stan w tej chwili (sygnały e_k są aktywne), harmonogramują węzły przez siebie sterowane (ang.: fanout). Następnie obsługiwane są kolejno wszystkie węzły (zdarzenia) zaharmonogramowane dla t_n, aż do opróżnienia listy. Stan kolejnego węzła jest określony w oparciu o model logiczno-czasowy elementu, dla którego węzeł ten jest wyjściem, oraz w oparciu o stany wejść w chwili t_n. Jeśli nowy stan wyjścia jest różny od poprzedniego, to wyznaczane jest opóźnienie Δt_i. Gdy tak powstałe zdarzenie pojawia

się przed zakończeniem poprzedniego przełączenia w tym węźle (do przeciwnego stanu), to poprzednie zdarzenie zostaje unieważnione (usunięte z listy wynikowej) i dodatkowo przygotowany zostaje komunikat o „pochłoniętym" impulsie czasowym (ang.: glitch). Jeśli ostatnie zdarzenie nie wywołuje tej sytuacji, to oblicza się czasy propagacji Δt_k do wszystkich węzłów sterowanych z węzła i. Węzły te są następnie harmonogramowane (zapisywane do kolejki) dla $t_n + \Delta t_k$. Natomiast, nie zmieniające stanu węzły nie są analizowane (tzw. stan „uśpienia", ang.: latency), zgodnie z zasadą śladu selektywnego.

5. Ewolucja algorytmów symulacji ciągłej

Postęp w dziedzinie ciągłej symulacji elektrycznej jest związany z metodami dekompozycji. Kluczowym aspektem metod opartych na dekompozycji jest to, że ich nakład obliczeniowy rośnie w przybliżeniu liniowo z rozmiarem obwodu, podczas gdy dla metod konwencjonalnych zależność ta jest wykładnicza.

Pierwsze próby wykorzystania dekompozycji w symulacji elektrycznej dotyczyły technik bezpośrednich, takich jak blokowa faktoryzacja LU [15] czy też dwupoziomowy algorytm Newtona-Raphsona [37]. Zasadniczo jednak metody pośredniej dekompozycji (relaksacyjne) zadecydowały o dalszym rozwoju tej dziedziny, a w szczególności symulacji hybrydowej.

Z matematycznego punktu widzenia [11] technika relaksacyjna może być stosowana, ogólnie rzecz biorąc, na wszystkich poziomach przetwarzania równań obwodowych [23], chociaż w symulacji obwodów jest wykorzystywana tylko na dwu. W bezpośrednim odniesieniu do układu obwodowych równań różniczkowych znana jest ona pod nazwą relaksacji przebiegów (ang.: waveform relaxation). Zastosowana po dyskretyzacji tych równań, określana jest jako symulacja czasowa (ang.: timing simulation). Natomiast na poziomie linearyzacji tych równań, podczas iteracyjnego ich rozwiązywania np. metodą Newtona-Raphsona (NR), technika relaksacyjna z reguły nie jest stosowana [47]. U podstaw tego faktu leży kilka przyczyn. Po pierwsze, trudne do określenia warunki zbieżności. Po drugie, stosunkowo duży nakład obliczeń w porównaniu do skutecznych metod bezpośrednich (np. techniki macierzy rzadkiej) - wynikający z konieczności prowadzenia iteracji. Ponadto, ponieważ inte-

resujące równania liniowe są rozwiązane wewnątrz pętli NR, istnieje wciąż problem wyznaczenia pełnej macierzy Jacobiego, co z reguły znacznie zwiększa nakład obliczeń.

W dalszym toku scharakteryzowane zostaną algorytmy symulacji czasowej oraz relaksacji przebiegów.

5.1. Algorytmy symulacji czasowej

Zastosowanie wybranego algorytmu dyskretyzacji (np. wsteczny Eulera, trapezów, Geara) w celu numerycznego scałkowania układu równań obwodowych (1) prowadzi do układu nieliniowych równań algebraicznych, które rozwiązuje się wielokrotnie dla kolejnych chwil czasu. Układ tych równań można przedstawić ogólnie w postaci:

$$g_i(x_1,\ldots, x_m) = 0 \qquad i = 1,2,\ldots, m \tag{3}$$

Relaksacyjna metoda rozwiązania równań (3) opiera się na jednym z dwu schematów: Gaussa-Seidela (GS) lub Gaussa-Jacobiego (GJ). W poszczególnych iteracjach każde spośród m równań jest rozwiązywane względem jednej zmiennej. Podstawowy algorytm relaksacyjny można przedstawić następująco:

$k \leftarrow 0;$

repeat

$k \leftarrow k + 1;\ i \leftarrow 0;$

repeat

$i \leftarrow i + 1;$

GS: oblicz x_i^k $\quad g_i(x_1^k,\ldots,x_{i-1}^k, x_i^k, x_{i+1}^{k-1},\ldots,x_m^{k-1}) = 0$

GJ: oblicz x_i^k $\quad g_i\left(x_1^{k-1},\ldots,x_{i-1}^{k-1}, x_i^k, x_{i+1}^{k-1},\ldots,x_m^{k-1}\right) = 0$

until $i = m$;

until $\|x^k - x^{k-1}\| \le \varepsilon_1$, $\|g_i(x^k)\| \le \varepsilon_2$, $i = 1,\ldots, m$.

gdzie ε_1, ε_2 oznaczają tolerancje błędu, natomiast

$$x^k = \left[x_1^k, \ldots, x_{i-1}^k, x_i^k, x_{i+1}^k, \ldots, x_m^k\right]^T$$

Każde z nieliniowych równań jest w praktyce rozwiązywane w oparciu o algorytm NR. Ponieważ w tym przypadku z i-tego równania wyznaczona jest zmienna x_i, to elementy leżące na przekątnej macierzy Jacobiego $\partial g_i/\partial x_i$ muszą być niezerowe. Ponadto elementy te powinny zapewniać diagonalną dominację w tej macierzy. W odniesieniu do analizowanego obwodu oznacza to, że każdy węzeł musi posiadać pojemność do masy lub do innego węzła o stałym potencjale.

Połączenie techniki relaksacji z algorytmem NR spowodowało, że stosowane określenie łączne: algorytm Gaussa-Seidela/Newtona-Raphsona (GSNR) lub Gaussa-Jacobiego/Newtona-Raphsona (GJNR). Każdy z nich obejmuje dwie pętle iteracyjne: zewnętrzną - relaksacyjną i wewnętrzną - linearyzacyjną. Można wykazać, że dla uzyskania asymptotycznie liniowej zbieżności pętli relaksacyjnej wystarczająca jest pojedyncza iteracja NR. Natomiast doprowadzenie do zbieżności w pętli wewnętrznej poprawia dodatkowo zbieżność pętli zewnętrznej [11]. Kryterium zbieżności dla przypadku liniowego jest diagonalna dominacja wspomnianej macierzy Jacobiego.

Łatwo zauważyć, że różnica między algorytmami GS i GJ polega na tym, że pierwszy z nich uaktualnia wartości zmiennych podczas każdej iteracji. Drugi natomiast uaktualnia je dopiero w następnym kroku iteracyjnym. Dlatego też algorytm GSNR jest naturalnie dopasowany do połączenia z techniką śladu selektywnego, co zapewnia mu szybszą zbieżność od algorytmu GJNR. Ten ostatni natomiast jest korzystny w implementacji równoległej przy użyciu komputera wieloprocesowego. Jeśli jakobian $\partial g_i/\partial x_j$ jest dolną macierzą trójkątną, to relaksacyjna pętla GS daje rozwiązanie dokładne w pojedynczej iteracji. Jest to oczywiście przypadek jednokierunkowego rozchodzenia się sygnału w obwodzie (brak obciążenia i sprzężeń zwrotnych), w praktyce realizowany tylko w przybliżeniu, np. przez cyfrowe struktury MOS bez bramek transmisyjnych.

Pierwszy symulator czasowy MOTIS [22], oparty na przedstawionej technice relaksacyjnej, wykorzystywał do dyskretyzacji metodę Eulera z krokiem wstecz oraz algorytm GJNR z pojedynczą iteracją w pętli zewnętrznej, a także wewnętrznej (co

ten był jednak ograniczony w zastosowaniu do struktur MOS z nie obciążającymi się tranzystorami. Pojemności międzywęzłowe nie były dozwolone i mogły być jedynie zastąpione pojemnościami równoważnymi do masy. Jego stabilność zależała ponadto silnie od długości kroku całkowania (pomimo użycia wstecznej metody Eulera) i obecności sprzężeń międzywęzłowych, występujących np. w postaci bramek transmisyjnych, które wymagały oddzielnej analizy techniką bezpośrednią.

Przyspieszenie obliczeń o dwa rzędy wielkości w stosunku do bezpośredniej symulacji elektrycznej stało się zachętą do rozwinięcia technik symulacji czasowej. W symulatorze SPLICE1 (pierwszym hybrydowym) [25] stosującym symulację czasową wprowadzono pojęcie zdarzenia analogowego, co silnie zaważyło na dalszym rozwoju w tej dziedzinie. Progam był oparty na algorytmie GSNR, a zastosowane techniki sterowania zdarzeniami oraz śladu selektywnego znacznie poprawiły efektywność symulacji. Algorytm śladu selektywnego porządkował układ równań zgodnie z drogą rozprzestrzeniania się sygnałów w obwodzie. Konsekwentnie więc nieaktywne („śpiące") węzły struktury nie były analizowane.

Koncepcja ta, zaczerpnięta z symulacji logicznej, wymagała jednak odrębnego potraktowania i posiadała nowe konsekwencje. Wprowadzono formalne kryteria aktywności węzła (stan „uśpienia") w postaci warunków określających przyrost potencjału oraz jego szybkość (istotną w przypadku stosowania zmiennego kroku całkowania). W najprostszym przypadku warunki te przyjmują postać:

$$|v_{n+1} - v_n| < \varepsilon_\Delta, \qquad |v_{n+1} - v_n| / h_n < \varepsilon_s.$$

Odpowiednie kryteria powrotu do stanu aktywności (warunek „przebudzenia") można natomiast wyrazić następująco:

$$|v_{n+1} - v_n|\, h_{n+1} / h_n < \varepsilon_\Delta, \qquad t_a = t_{n+1} + h_{n+1},$$

gdzie t_a oznacza chwilę „przebudzenia". Występujący tutaj parametr ε_Δ może przyjmować w szczególności wartość dopuszczalną lokalnego błędu obcięcia dla stosowanego algorytmu całkowania. Przyjmuje się, że dany węzeł (element) jest aktywny, tzn. poddany analizie, jeśli nie spełnia warunków „uśpienia". Wówczas w wyniku

każdej zmiany potencjału generuje on zdarzenie. Z kolei, węzeł pozostaje w stanie „uśpienia", jeśli spełnia wyszczególnione warunki formalne oraz węzły sterujące go (ang.: fanins) także są w stanie uśpienia.

Harmonogramowanie obejmuje w tym przypadku nie tylko węzły sterowane przez dany węzeł aktywny, ale także węzeł sterujący. Różnica polega jedynie na tym, że węzły sterowane zapisywane są do kolejki czasowej jeszcze w bieżącej chwili czasu t_n, a węzeł sterujący harmonogramuje sam siebie w kolejnym kroku czasowym $t_n + h_n$.

W celu analizy obwodów z silnymi sprzężeniami międzywęzłowymi w późniejszej wersji symulatora SPLICE zmodyfikowano metodę całkowania (dwukrokowa technika Implicit-Implicit-Explicit [25]), a iteracje NR w wewnętrznej pętli doprowadzono do zbieżności, zachowując jednak pojedynczą iterację w pętli zewnętrznej. Krok całkowania był stały. Rezultat ten musiał być jednak okupiony wydłużeniem czasu obliczeń.

W dalszych pracach nad symulacją czasową [1] (Iterated Timing Analysis) powrócono do koncepcji pojedynczej iteracji w pętli wewnętrznej, doprowadzając jednocześnie do zbieżności w zewnętrznej pętli relaksacyjnej. Wykorzystano w tym przypadku dodatkowo tzw. technikę nadrelaksacji (ang.: Successive Overrelaxation - SOR). Zgodnie z tą zasadą obliczone w pojedynczej iteracji NR wartości zmiennych są uaktualniane następująco:

$$x_i^k \leftarrow (1-\omega)\, x_{x_i}^{k-1} + \omega x_i^k$$

gdzie ω stanowi parametr relaksacyjny dobierany na ogół doświadczalnie. Jego wartość może mieć silny wpływ na szybkość zbieżności w zewnętrznej pętli.

Jednym z głównych spostrzeżeń w zakresie zastosowań technik relaksacyjnych było to, że obwody o silnie sprzężonych węzłach efektywniej można symulować metodami bezpośrednimi [23]. Algorytmy relaksacyjne charakteryzuje bowiem w tych przypadkach bardzo wolna zbieżność, a nawet niestabilność. Dają natomiast bardzo dobre rezultaty w zastosowaniu do obwodów ze słabymi sprzężeniami oraz bez „ciasnych" sprzężeń zwrotnych. Dlatego za kompromisowe rozwiązanie można

wodu w celu wyodrębnienia fragmentów o silnie sprzężonych węzłach [1]. Dla tak wyodrębnionych podobwodów można prowadzić iteracje w obu pętlach do uzyskania zbieżności lub stosować lokalnie techniki bezpośrednie. Stosowane metody relaksacyjne obejmują wtedy zamiast pojedynczych węzłów bloki węzłów silnie sprzężonych. Podejście to znalazło także zastosowanie w technice relaksacji przebiegów.

5.2. Algorytmy relaksacji przebiegów

W symulacji obwodów, w których sygnały elektryczne zmieniają się w zróżnicowany sposób (ang.: multirate behaviour), bardzo efektywne okazały się algorytmy relaksacji przebiegów (ang.: waveform relaxation) [23,4]. Operują one na całych sygnałach w przeciwieństwie do symulacji czasowej, w której iteracyjnie wyznacza się jedynie ich chwilowe wartości. Podstawową zaletą i cechą wyróżniającą algorytmów relaksacji przebiegów jest możliwość analizowania poszczególnych fragmentów obwodu z indywidualnym, zmiennym krokiem czasowym. Natomiast, podobnie jak w symulacji czasowej, algorytmy te mogą wykorzystywać relaksację GS łącznie z techniką śladu selektywnego lub w oparciu o relaksację GJ być implementowane w postaci równoległej. Czas obliczeń przy wykorzystaniu tych algorytmów rośnie w przybliżeniu liniowo z rozmiarem analizowanego obwodu.

W celu zobrazowania tej techniki układ równań różniczkowych (1) opisujący obwód zostanie przepisany w postaci:

$$F_i\left[\frac{dx_i(t)}{dt}, \frac{dr_i(t)}{dt}, x_i(t), r_i(t), u(t)\right] = 0 \quad i = 1,\ldots,m \quad x(0) = x_0 \tag{4}$$

gdzie:

$$r_i(t) = [x_1(t),\ldots, x_{i-1}(t), x_{i+1}(t),\ldots, x_m(t)]^T \qquad t \in \langle 0, t_{max}\rangle$$

Algorytm relaksacyjny można wówczas przedstawić następująco:

k ← 0;

repeat

k ← k +1: i ← 0:

repeat

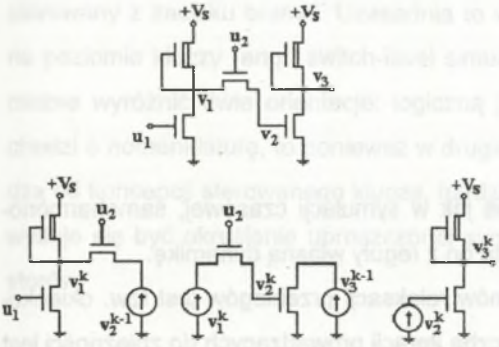
$i \leftarrow i+1$;

oblicz $x_i^k(t)$: $F_i\left[\frac{dx_i^k(t)}{dt}, \frac{dr_i^k(t)}{dt}, x_i^k(t), r_i^k(t), u(t)\right] = 0$

until $i = m$

untli $\left\| x^k(t) - x^{k-1}(t) \right\| < \varepsilon, \quad t \in \langle 0, t_{max} \rangle$.

W przypadku relaksacji GS elementy wektora $r_i(t)$ są uaktualnione podczas każdej iteracji, natomiast przy wykorzystaniu algorytmu GJ - dla następnego k. W praktycznych zastosowaniach algorytm relaksacji przebiegów posiada łagodne warunki zbieżności dla dowolnych warunków początkowych [4], jednak w wersji GS jego szybkość zbieżności zależy silnie od kolejności rozwiązywania równań (4). Dlatego też stosowany jest łącznie z techniką śladu selektywnego, działającą porządkująco. Uporządkowanie to ma zasadniczo charakter „przestrzenny" w odniesieniu do struktury, a nie jak dotąd czasowy. W procesie harmonogramowania do kolejki (lub na stos) wprowadzone są węzły lub elementy pobudzane z zewnątrz w kolejności wynikającej z analizy propagujących sygnałów. Poszczególne chwile czasu nie odgrywają roli w harmonogramowaniu.



Rys. 2. Ilustracja algorytmu relaksacji przebiegów
Fig. 2. Illustration of waveform relaxation algorithm

Na rys. 2 w celu zilustrowania działania algorytmu relaksacji przebiegów GS przedstawiono prosty obwód nMOS i sposób jego iteracyjnej analizy. Kolejność analizy wynika z drogi propagacji sygnału w obwodzie (k oznacza wskaźnik iteracji).

W ogólniejszym przypadku mamy do czynienia z tzw. blokową relaksacją przebiegów. Wówczas w kolejnych krokach rozwiązaniu podlegają nie pojedyncze równa-

nia, ale małe podukłady równań (odpowiadające poszczególnym podobwodom), na które można podzielić układ równań (4). Proces uporządkowania (dekompozycji) tego układu posiada najczęściej charakter statyczny. Poprzedza on wtedy etap relaksacji. Ideą tej dekompozycji jest łączne rozwiązywanie równań dla węzłów silnie sprzężonych (podobwodów) (ang.: strongly connected componenets - SCC), co bardzo wyraźnie poprawia zbieżność całej analizy. Granice podziału topologicznego struktury na bloki (podobwody) wyznaczają na ogół węzły sterujące o niskiej impedancji wyjściowej (np. wyjście wzmacniacza operacyjnego objętego sprzężeniem zwrotnym) lub węzły sterowane o wysokiej impedancji wejściowej (np. bramka tranzystora MOS, nie obciążona innymi elementami) [4,55]. Jeśli ponadto w tak zdekomponowanym obwodzie nie występują sprzężenia zwrotne, to sygnał rozprzestrzenia się jednokierunkowo i rozwiązanie można uzyskać w pojedynczej iteracji.

Przykład wykorzystania techniki śladu selektywnego w symulacji opartej na relaksacji przebiegów GS przedstawia poniższy algorytm [59]:

```
repeat {
  foreach {{ blok B oczekujący w kolejce} do {analiza B}
    foreach {{ zewnętrzny węzeł N należący do B} do
      if {pierwsza analiza B} then {harmonogramuj B i bloki sterowane przez N}
        else if {potencjał N zmienił się w stosunku do poprzedniej iteracji}
          then {harmonogramuj B i bloki sterowane przez N}
            else {pomiń N}
  } } }
until {kolejka jest pusta}.
```

Charakterystyczne jest tu, podobnie jak w symulacji czasowej, samoharmonogramowanie danego bloku, gdyż posiada on z reguły własną dynamikę.

Drugim istotnym aspektem algorytmów relaksacji przebiegów jest tzw. okienkowanie. Ponieważ zaobserwowano, że liczba iteracji prowadzących do zbieżności jest wprost proporcjonalna do czasu analizy t_{max}, dlatego typowym zabiegiem jest podział interwału $\langle 0, t_{max}\rangle$ na „okna czasowe": $\langle 0, t_1\rangle$, $\langle t_1, t_2\rangle$,..., $\langle t_{n-1}, t_{max}\rangle$. Iteracje prowadzone są wtedy aż do uzyskania zbieżności oddzielnie w kolejnych oknach. Po

uzyskaniu zbieżności następuje harmonogramowanie bloków dla analizy w następnym oknie w oparciu o zdarzenia zewnętrzne oraz zdarzenia w węzłach granicznych (sterujących) poszczególnych bloków.

Dodatkową zaletą okienkowania jest oszczędniejsze wykorzystanie pamięci operacyjnej komputera, gdyż przebiegi czasowe muszą być zapamiętywane na okres jednej iteracji. Zarówno problem podziału obwodu na bloki, jak i podziału czasu pełnej analizy na okna stanowi zagadnienie optymalizacyjne, w którym wskaźnikiem jakości jest całkowity czas obliczeń [59]. Nadmierny podział obwodu na bloki lub interwału symulacji na okna wydłuża obliczenia. Zagadnienie to nie doczekało się jednak jak dotąd w pełni racjonalnego rozwiązania, a istniejące algorytmy mają charakter w znacznej mierze heurystyczny.

6. Symulacja na poziomie kluczy

W strukturach cyfrowych MOS pojedyncze tranzystory odgrywają bezpośrednio większą rolę niż ich odpowiedniki w strukturach bipolarnych. Jako przykład można tu wymienić bramki transmisyjne, rejestry „zatrzaskowe" z dodatnim sprzężeniem zwrotnym czy też magistrale z tranzystorami „podciągającymi" [12, 13]. Dlatego też symulacja logiczna układów MOS na poziomie bramek z reguły nie jest wystarczająca i wymaga przejścia na poziom tranzystorów.

Najprostszą reprezentacją tranzystora MOS jest w tym przypadku idealny klucz sterowany z zacisku bramki. Uzasadnia to w istocie przyjęte określenie - symulacji na poziomie kluczy (ang.: switch-level simulation). W symulacji na poziomie kluczy można wyróżnić dwie orientacje: logiczną [28,29] i elektryczną [42,31,3,34]. Jeśli chodzi o nomenklaturę, to ponieważ w drugim przypadku stosowane modele odchodzą od koncepcji sterowanego klucza, bardziej racjonalne dla tej grupy symulatorów wydaje się być określenie uproszczonej symulacji elektrycznej na poziomie tranzystorów.

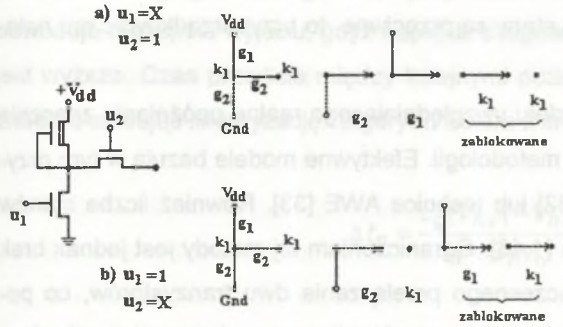
6.1. Symulacja na poziomie kluczy zorientowana logicznie

W symulacji tej największą popularność zyskała technika opracowana przez Bryanta [29]. Wykorzystuje ona logikę trójwartościową (0,X,1), a przewodzące tranzystory są uporządkowane symbolicznie wg posiadanej przewodności (ang.: strength). Stan X, w zależności od potrzeb, jest uważany za stan pośredni albo (w przypadku konfliktu) nieokreślony. Również symbolicznie uporządkowane są węzły, tzn. wg zdolności dostarczania ładunku (ang.: size). Najwyżej w uporządkowaniu tym znajdują się węzły odpowiadające zaciskom zasilania lub masy. Pozostałe węzły klasyfikowane są wg pojemności (największą wagę posiadają magistrale). W uporządkowaniu łącznym węzły występują przed przewodzącymi tranzystorami.

Symulacja tą techniką polega na analizie niewielkich na ogół podobwodów złożonych z tranzystorów połączonych wzajemnie drenami lub źródłami (tzw. silnie sprzężone elementy). Granice tych struktur wytyczają węzły, do których dołączone są bramki tranzystorów zewnętrznych (należących do sąsiednich podobwodów). Wewnątrz podobwodów wyróżnia się ścieżki sygnałowe, prowadzące od poszczególnych węzłów do masy i zasilania. Każdej ścieżce przyporządkowuje się wagę, która jest równa symbolicznej wartości „najsłabszego" tranzystora w tej ścieżce.

„Słabsze" ścieżki blokowane są w węzłach pośrednich, jeśli dochodzą do nich ścieżki „silniejsze". Te ostatnie działają blokująco tylko wtedy, gdy nie zawierają tranzystorów w stanie X. Najsilniejsza z niezablokowanych ścieżek dochodzących do danego węzła określa jego stan logiczny. W przypadku gdy w węźle spotykają się dwie ścieżki o jednakowej wadze (sile), lecz wymuszają przeciwne stany logiczne, to węzeł ten uzyskuje stan X (nieokreślony).

Zaletą tej metody jest możliwość uwzględnienia zjawiska podziału ładunku lub dwukierunkowego przepływu sygnału, np. przez bramki transmisyjne. Przykład analizy typowego podobwodu nMOS w oparciu o graf przedstawia rys. 3. W obu przypadkach zaznaczono po dwie ścieżki zablokowane i niezablokowane (są wśród nich także ścieżki zdegenerowane do węzła). Symbolami g_i i k_j są wagi odpowiednio gałęzi (ścieżek) oraz węzłów (ścieżek). W przypadku a) tranzystor sterujący znajduje się w stanie X, a niezablokowane ścieżki wymuszają w węźle i przeciwne stany. Dlatego węzeł ten przyjmuje stan X.



Rys. 3. Ilustracja algorytmu symulacji Bryanta
Fig. 3. Illustration of Bryant's simulation algorithm

Natomiast w przypadku b) w stanie X znajduje się tranzystor transmisyjny i gdy stan początkowy węzła i wynosi 0, to węzeł ten zachowa swój stan. Jeśli jednak stan ten przyjmuje wartość początkową 1, to ulegnie ona zmianie na X.

Należy przy tym zauważyć, że wynik ten posiada związek z nieokreślonym stanem tranzystora transmisyjnego. Gdyby bowiem był on określony jako w pełni przewodzący, to ścieżka do masy blokowałaby ścieżkę stanowiącą wyizolowany węzeł i. Istniałaby więc tylko jedna ścieżka niezablokowana, a zatem węzeł i uzyskałby stan 0 niezależnie od swojej wartości początkowej.

Analiza obwodu tą techniką odbywa się przy wykorzystaniu techniki śladu selektywnego. W większości praktycznych przypadków wynik tej logicznej analizy uzyskuje się na drodze iteracji przy założeniu jednostkowych opóźnień. Jej efektywność obliczeniowa jest zbliżona do efektywności logicznej symulacji bramkowej.

Podczas inicjalizacji pewien problem stwarzają tranzystory z początkowym stanem X na swoich bramkach. Zastąpienie X odpowiednio przez 1 oraz 0 dla wszystkich kombinacji prowadzi niewątpliwie w prosty sposób do oceny stanu początkowego w obwodzie, ale za cenę ogromnego nakładu obliczeniowego (rosnącego wykładniczo). W [30] zaproponowano technikę, która w dwu krokach iteracyjnych pozwala ocenić stan węzłów określonych początkowo jako X. W tym celu w pierwszym kroku przez tranzystory w stanie X propaguje się sygnały 1 (lub X), w drugim nato-

miast sygnały 0 (lub X). Jeśli dany węzeł w obu analizach uzyska ten sam stan, to jest on określony. Jeśli obliczone stany są przeciwne, to przyporządkować mu należy stan X.

Wprowadzenie dodatkowo modelu uwzględniającego realne opóźnienia znacznie rozszerza zakres zastosowań tej metodologii. Efektywne modele bazują w tym przypadku na koncepcji drzew RC [32] lub technice AWE [33]. Również liczba stanów logicznych może być zwiększona [1,46]. Ograniczeniem tej metody jest jednak brak możliwości odwzorowania równoczesnego przełączenia dwu tranzystorów, co powoduje, że jest ona stosowana głównie do weryfikacji czasowej pozwalającej odnaleźć krytyczne ścieżki sygnałowe w projekcie [38,39,40].

Dokładniejsza symulacja układów na poziomie kluczy wymaga orientacji elektrycznej.

6.2. Symulacja na poziomie kluczy zorientowana elektrycznie

W tym rodzaju symulacji operuje się na dyskretnych poziomach napięć, odpowiadających stanom logicznym w algebrze wielowartościowej, a mechanizm opóźnień jest w naturalny sposób związany z przyjętymi modelami tranzystorów. Powszechne jest także wykorzystanie techniki relaksacyjnej.

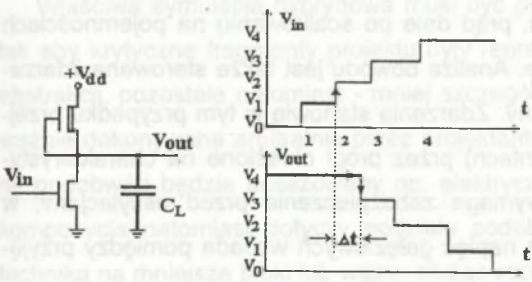
Jednym z najbardziej znanych podejść w tej grupie algorytmów jest technika symulacji elektryczno-logicznej (ang.: E-logic) [42,1]. W przeciwieństwie do klasycznej symulacji elektrycznej, przyjmuje się tam najpierw krok napięciowy (niekoniecznie stały), a następnie wylicza czas niezbędny do dokonania tej zmiany. Liczba poziomów napięciowych wpływa na dokładność uzyskanych przebiegów i może być ustalona arbitralnie. Patrząc nieco ogólniej, podczas symulacji cecha ta stwarza możliwość dokonywania elastycznego kompromisu między szybkością obliczeń a ich dokładnością.

Algorytm ten można uważać za uogólnienie podejścia zaprezentowanego w [31], gdzie wykorzystano trzy poziomy napięciowe dla określenia sygnału, a analizę opóźnień (z dokładnością 30%) oparto na uproszczonej analizie ekwiwalentnego obwodu RC. Tranzystory w stanie włączenia były reprezentowane przez równoważne opory i pojemności do masy.

Działanie algorytmu E-logic ilustruje rys. 4. Pierwsze przełączenie na wejściu nie powoduje reakcji na wyjściu, gdyż napięcie progowe bramki tranzystora sterującego jest wyższe. Czas przejścia między kolejnymi poziomami napięciowymi można wyznaczyć stosując linearyzację i algorytm Eulera z krokiem wprzód ze wzoru:

$$\Delta t_n = \frac{c_g(V_{n+1} - V_n)}{I_n - G_n V_n}$$

gdzie C_g oznacza całkowitą zastępczą pojemność do masy tego węzła, a I_n i G_n odpowiednio: sumaryczną siłę prądomotoryczną i sumaryczną przewodność związane z tym węzłem (po linearyzacji charakterystyk tranzystorów). Jeśli kolejne zdarzenie na wejściu nadchodzi przed zakończeniem stanu przejściowego do nowego stanu V_{n+1} na wyjściu, to algorytm arbitralnie zaokrągla wartość napięcia wyjściowego w dół do V_n lub w górę do V_{n+1}. Ściślej mówiąc, gdy aktualna wartość napięcia jest bliska V_n, to poprzednie zdarzenie jest unieważniane i Δt_n jest obliczane na nowo. Jeśli aktualne napięcie jest bliskie V_{n+1}, to następuje zaokrąglenie do tego stanu, przy czym nowe zdarzenie jest harmonogramowane, pod warunkiem że wywoła pełne przełączenie do stanu sąsiedniego V_{n+2}. Natomiast dla napięcia wyjściowego, leżącego w strefie środkowej pomiędzy V_n i V_{n+1}, stare zdarzenie jest unieważniane, a czas Δt_n jest obliczany ponownie i jako ostateczną jego wartość przyjmuje się średnią z nowo obliczonej oraz poprzedniej.



Rys. 4. Generacja przebiegu w logarytmie E-logic
Fig. 4. Signal generation in E-logic algorithm

Podstawowym problemem symulacji elektryczno-logicznej jest możliwość powstania oscylacji. Ich pochodzenie wynika z cechy samego algorytmu (deskretyzacja amplitudy) lub z obecności silnych sprzężeń międzywęzłowych. Dlatego też opisany tu w uproszczeniu algorytm wymaga dodatkowo pewnych zabezpieczeń [1].

Algorytm E-logic został zaimplementowany z modyfikacjami w co najmniej kilku znanych symulatorach (w oryginalnej postaci pojemności międzywęzłowe nie były dozwolone).

W symulatorze MOTIS3 [43] zastosowano zmienny krok napięciowy, a także wprowadzono koncepcję „superbloku" w celu odrębnego rozwiązania równań dla węzłów silnie sprzężonych. Umożliwiło to poprawę dokładności oraz uniknięcie oscylacji.

Inne rozwiązanie problemu węzłów silnie sprzężonych w strukturach MOS zastosowano w symulatorze ADEPT [44]. Podczas analizy każdego węzła sprawdzane jest jego „otoczenie". Jeśli estymaty prądów płynących w kierunku węzłów sąsiednich są relatywnie duże w stosunku do prądu płynącego do masy, to węzły te są analizowane w lokalnym procesie relaksacyjnym aż do osiągnięcia zbieżności.

Przykładem odrębnego podejścia do symulacji obwodów na poziomie tranzystorów (kluczy) jest program SPECS [34]. Równania są tam układane nie metodą węzłową, a w oparciu o klasyczny graf obwodu i koncepcję drzewa [8,9]. Dyskretyzacja dotyczy zasadniczo nieliniowych charakterystyk elementów obwodu, dla których przyjmuje się odcinkowo stałe wartości prądu w funkcji napięcia. Stąd też, stały w poszczególnych przedziałach czasu, prąd daje po scałkowaniu na pojemnościach odcinkowo liniowe w czasie napięcie. Analiza obwodu jest także sterowana zdarzeniami i posiada charakter relaksacyjny. Zdarzenia stanowią w tym przypadku przejścia napięć gałęziowych (na elementach) przez progi określone na charakterystykach I-U. Algorytm ten również wymaga zabezpieczenia przed oscylacjami, w szczególności gdy stan ustalony dla napięć gałęziowych wypada pomiędzy przyjętymi progami na charakterystykach.

Jako ostatni, charakterystyczny dla tej grupy, można wymienić symulator iDSIM [3]. W odróżnieniu od poprzednich wykorzystuje on dodatkowo technikę relaksacji przebiegów. Analiza struktury przebiega w kilku etapach. Pierwszy stanowi dekom-

pozycja struktury na bloki tranzystorów sprzężonych. Dla każdego bloku wylicza się następne parametry, określając blok jako makromodel. Kolejnym krokiem jest uproszczona symulacja struktury na poziomie kluczy w oparciu o trzy poziomy napięcia (L,I,H) oraz mechanizm opóźnień wykorzystujący koncepcję drzew RC [32]. Symulacja ta prowadzi poprzez analizę zdarzeń do wyznaczenia okien czasowych (dynamiczne okienkowanie) dla ostatecznej analizy - blokowej relaksacji przebiegów, sterowanej zdarzeniami. Mimo zwiększonego nakładu obliczeń czasy symulacji przy użyciu programu iDSIM są dwa, a dla struktur bez sprzężeń zwrotnych nawet o trzy rzędy wielkości mniejsze niż przy użyciu klasycznego symulatora SPICE.

7. Algorytmy symulacji hybrydowej

Symulacja hybrydowa, jak wspomniano na wstępie, łączy zalety różnych rodzajów symulacji, obejmujących poziomy abstrakcji: od behawioralnego i przesłań międzyrejestrowych aż po poziom najniższy - symulacji elektrycznej. Z drugiej natomiast strony, integracja odmiennych technik symulacyjnych jest wyrazem kompromisu między szybkością obliczeń a szczegółowością uzyskanych przebiegów czasowych. Przeznaczenie symulatora decyduje na ogół o tym, czy obejmuje on głównie poziomy wyższe, jak np. dla złożonych projektów czysto cyfrowych, czy też poziomy niższe, jak np. dla struktur A/C. W doborze technik symulacji istotną rolę może odgrywać także technologia. Przykładem tego są cyfrowe układy MOS, które najczęściej wymagają symulacji w poziomie kluczy.

Właściwa symulacja hybrydowa musi być poprzedzona dekompozycją obwodu, tak aby krytyczne fragmenty projektu były reprezentowane na niższych poziomach abstrakcji, pozostałe natomiast - mniej szczegółowo. Dekompozycja taka jest przeważnie dokonywana arbitralnie przez projektanta, który musi zadecydować, czy dany podobwód będzie analizowany np. elektrycznie czy tylko czasowo. Wtórna dekompozycja natomiast dotyczy podziału podobwodu analizowanego ustaloną już techniką na mniejsze bloki lub węzły, służąc zapewnieniu optymalnych warunków tej analizy (np. ukryciu lokalnych sprzężeń zwrotnych wewnątrz bloków). Proces ten

jest najczęściej wykonywany automatycznie i może być realizowany statycznie (przed analizą) lub dynamicznie (w trakcie analizy), jak np. we wspomnianym symulatorze ADEPT [44].

Nowoczesne symulatory hybrydowe [1] są oparte na technikach sterowania zdarzeniami i śladu selektywnego. Wpływa to nie tylko na efektywność obliczeń, ale również działa unifikująco na poszczególne subsymulatory tworzące łącznie symulator hybrydowy. Dlatego też na wszystkich wykorzystywanych poziomach abstrakcji musi być zdefiniowane pojęcie zdarzenia oraz sposób harmonogramowania. Bezpośrednio z tymi problemami związana jest organizacja czasu i sposób synchronizacji, a także konwersja sygnału podczas jego propagacji pomiędzy podobwodami reprezentowanymi na różnych poziomach.

W dalszym toku rozważań nasza uwaga będzie nadal skupiona na niższych pozimach abstrakcji, gdzie współdziałają algorytmy symulacji ciągłej i dyskretnej.

7.1. Problem synchronizacji czasu

W konwencjonalnej symulacji hybrydowej subsymulatory dyskretny i ciągły posiadały znaczną autonomię (np. [26]). Stosownie do propagacji sygnału komunikacja pomiędzy nimi odbywała się poprzez dwa oddzielne kanały. Pracując niezależnie, w momencie komunikacji wymagały one synchronizacji czasu. Algorytm określający w uproszczeniu taką synchronizację (ang.: lock-step) można przedstawić następująco [47]:

$t_n \leftarrow 0$;

while ($t_n \leq T_{stop}$) {

repeat {

analiza ciągła;

propagacja sygnału do części dyskretnej, gdy przekroczony próg logiczny;

analiza dyskretna;

propagacja sygnału logicznego do części ciągłej po zmianie jego stanu;

} **until** (brak aktywności sygnałów);

$t_n \leftarrow$ min (następny_krok _ sym _ ciągłej, następny _ krok _ sym _ dyskretnej);

}.

Ten prosty mechanizm synchronizacji jest jednak mało optymalny. Ponieważ kolejny krok symulacji jest wybierany jako minimum kroku symulatora dyskretnego i ciągłego, to jeśli zmiany sygnału w symulatorze ciągłym będą znacznie wolniejsze niż w dyskretnym, ten pierwszy będzie zmuszony wielokrotnie wykonywać czasochłonne, a właściwie zbędne obliczenia. Algorytm ten jest ponadto potencjalnie niedokładny, biorąc pod uwagę, że propagujący do części dyskretnej sygnał, przekraczając próg logiczny, staje się aktywny dopiero w następnym kroku czasowym. W celu uzyskania pożądanej dokładności niezbędny jest wówczas mechanizm odrzucenia tego kroku i powtórzenia procesu całkowania z krokiem skróconym (ang.: backtracking). W najprostszej realizacji mogą być konieczne iteracje. Dla poprawienia efektywności tego procesu wykorzystuje się metodę interpolacji. Jeśli stopień wielomianu interpolacyjnego jest równy rzędowi całkowania metody, wówczas dokładność tego procesu jest określona przez lokalny błąd obcięcia.

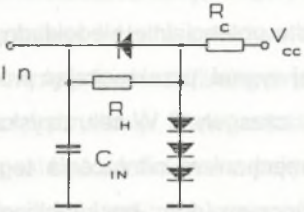
Również wszystkie nieciągłe zmiany na wejściu symulatora ciągłego, pochodzące od symulatora dyskretnego, wymagają synchronizacji i powodują, że proces całkowania musi zostać wstrzymany i powtórzony dokładnie od momentu nadejścia tego zdarzenia.

W symulatorach hybrydowych nowszej generacji wymagana jest podobna kontrola kroku całkowania (ang.: rollback), chociaż czas dla części dyskretnej i ciągłej jest na ogół zunifikowany (np. [16,55]).

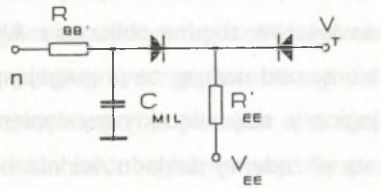
7.2. Konwersja sygnałów między symulatorem ciągłym i dyskretnym

Poprawna konwersja sygnałów podczas przejścia między analizowanymi blokami, które są reprezentowane na różnych poziomach abstrakcji, posiada zasadnicze znaczenie dla dokładności całej symulacji hybrydowej. Można zauważyć, że na ogół prostszy jest w realizacji przypadek przejścia z niższego poziomu na wyższy, gdyż wystarczy wówczas odrzucić zbędne w tym sygnale szczegóły. Jednak sterujący stopień wyjściowy powinien uwzględniać zjawisko obciążenia przez blok sterowany, analizowany dalej mniej szczegółowo (np. gdy tranzystor analizowany elektrycznie steruje wejściem bramki, analizowanej logicznie). Na rys. 5 i 6 przedstawiono jako przykład ekwiwalentne obwody wejściowe bramek wykonanych w technologii TTL i ECL umożliwiające odwzorowanie zjawiska obciążenia przy konwersji sygnałów z

postaci elektrycznej na logiczną [62,58]. Zjawisko obciążenia przez bramki MOS można modelować z reguły znacznie prościej, poprzez równoważną pojemność do masy.

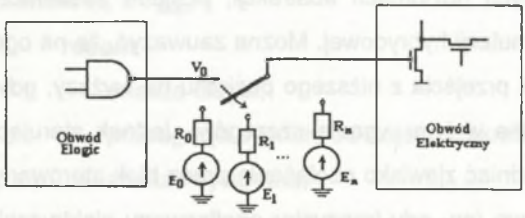


Rys. 5. Model wejściowego obwodu TTL
Fig. 5. Equivalent TTL input circuit



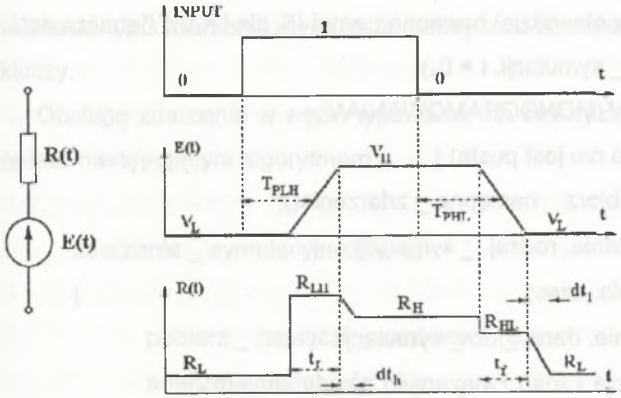
Rys. 6. Model wejściowego obwodu ECL
Fig. 6. Equivalent ECL input circuit

Znacznie trudniejsza jest konwersja w kierunku przeciwnym. Pierwsze symulatory hybrydowe rozwiązywały ten problem poprzez logicznie sterowane klucze z rezystorami i siłami elektromotorycznymi o wartościach odpowiadających poziomom logicznym 0 i 1 (ang.: boolean controlled elements) [24,26] lub poprzez proste konwertery logiczno-napięciowe albo logiczno-prądowe (ang.: LTV, LTI) [25] o ustalonych zboczach sygnałów. Niewielka dokładność tej konwersji spowodowała, że w późniejszych symulatorach dwustanowo sterowane elementy zastąpiono sterowanymi wielostanowo. Przykładowo, blok analizowany techniką E-logic może sterować wielostanowym konwerterem (ang.: voltage controlled switch) przedstawionym na rys.7 [1]. Wartości rezystancji umieszczonych w poszczególnych gałęziach mogą być



Rys. 7. Konwersja sygnału E-L na elektryczny
Fig. 7. Elogic to electrical signal conversion

wyznaczone w oparciu o charakterystyki wyjściowe R-V bramek, uzyskane za pomocą programu SPICE. Inne rozwiązanie dokładnego konwertera logiczno-elektrycznego pokazano na rys. 8 [62]. Po wyskalowaniu parametrów można implementować go dla różnych technologii



Rys. 8. Konwerter logiczno-napięciowy
Fig. 8. Logic to voltage converter

7.3. Przykłady implementacji algorytmów symulacji hybrydowej

Opisane zostaną pokrótce rozwiązania zastosowane w dwóch odmiennych symulatorach iSPLICE3 [16,1] oraz HYBRID [56,58,61]. Pierwszy z nich obejmuje trzy rodzaje symulacji: elektryczno-czasową, logiczną i na poziomie kluczy. Drugi symulację elektryczną i logiczną.

W symulatorze iSPLICE3 każde zdarzenie posiada trzy atrybuty dotyczące: funkcji, czasu oraz danych. Obsługa zdarzenia obejmuje realizację funkcji na danych w określonym czasie (rodzaj analizy, model, chwila czasu). Harmonogramowanie nowych zdarzeń jest objęte funkcją obsługującą zdarzenie i jest wykonywane w oparciu o kolejkę czasową zorganizowaną w postaci cyklicznej listy indeksowej (koło czasowe).

Główny algorytm symulacji w programie iSPLICE3 można przedstawić następująco:

```
main()
{
    wczytaj ();
    twórz _ podobwody ();
    harmonogramuj (przygotowanie _ analizy _ dc, t = 0 _);
    forall (podobwody Si w obwodzie) harmonogramuj (Si dla t = 0); /*analiza dc*/
    harmonogramuj (start _ symulacji, t = 0+);
    /*GŁÓWNA PĘTLA HARMONOGRAMOWANIA*/
    while (kolejka czasowa nie jest pusta) {
        zdarzenie ← Pobierz _ następne _ zdarzenie();
        funkcja ← zdarzenie. rodzaj _ symulacji;
        czas ← zdarzenie. czas;
        dane ← zdarzenie. dane _ do _ symulacji:
            /*Realizacja zadań związanych ze zdarzeniem*/
            funkcja (dane, czas);
        }
}
```

Po wczytaniu struktury następuje podział na podobwody. Każdy podobwód może być sklasyfikowany jako jeden z trzech typów: elektryczny (obsługa algorytmem ITA), logiczny (obsługa algorytmem bramkowej symulacji logicznej) albo typu klucz (obsługa algorytmem E-logic). Typ powinien być przydzielony podobwodom przez użytkownika w pierwotnym opisie struktury. Następnie etykietowane są węzły. Etykieta przyporządkowana węzłowi odpowiada najwyższemu poziomowi szczegółowości opisu elementów dołączonych do tego węzła. Np. jeśli do węzła dołączone są elementy elektryczne i typu klucz, to węzeł uzyskuje etykietę ELECTRICAL. Po tym przyporządkowaniu węzły ELECTRICAL są grupowane wtórnie, tworząc podobwody w taki sposób, aby objąć wszystkie lokalne pętle sprzężeń zwrotnych. Ostatecznie informacje te są ujęte w tablicach porządkujących podobwody jako sterujące i sterowane (ang.: fanin, fanout).

Następne czynności programu dotyczą przygotowania i analizy punktu pracy układu oraz uaktualnienia listy zdarzeń bezpośrednio przed rozpoczęciem symulacji.

Dalej symulator wchodzi w główną pętlę obsługi zdarzeń. Zakończenie następuje w momencie, gdy lista zdarzeń jest pusta, czyli gdy wcześniej zakończyło się także harmonogramowanie.

W czasie regularnej pracy w tej pętli najpierw pobrane zostaje zdarzenie, a następnie odbywa się jego obsługa wg odczytanego typu funkcji. Może być nią, stosowanie do typu zdarzenia, symulacja elektryczno-czasowa, logiczna albo na poziomie kluczy.

Obsługę zdarzenia w węźle typu klucz lub elektrycznym można w uproszczeniu opisać następującym algorytmem:

zdarzenie _ symulacyjne (S_i, t_n)

{

pobierz _ napięcia _ wejściowe (S_i);

analizuj _ podobwód (S_i);

if (S_i jest aktywny) harmonogramuj (S_i, t_{n+1});

foreach (węzeł j w S_i)

foreach (podobwód S_j sterowany z węzła j)

if (napięcie w j przekroczyło próg krytyczny dla S_j)

harmonogramuj (S_j, t_n);

}.

Jeśli po analizie obwodu dla t_n jego napięcia uległy zmianie, to jest on przewidziany ponownie do analizy w chwili t_{n+1}. Natomiast inne podobwody, sterowane przez analizowany obwód, są harmonogramowane dla bieżącej chwili czasu, jeśli przekroczone zostaną określone progi napięciowe.

Tak funkcjonuje mechanizm harmonogramowania dla bloków jednego typu. Natomiast, jeśli sygnał propaguje z bloku elektrycznego do bloków innego typu, to harmonogramowanie tych bloków następuje dopiero po stwierdzeniu zbieżności iteracji dla analizowanych węzłów elektrycznych, aby uniknąć zbędnych analiz bloków przez niego sterowanych.

Blok typu ELOGIC harmonogramuje sterowany przez siebie blok elektryczny zawsze po osiągnięciu nowego poziomu napięcia na wyjściu, ale nie dla t_n, lecz dla chwili czasu określonej przez warunek „przebudzenia" dla tego bloku (patrz punkt 5).

Bloki (elementy) typu LOGIC, sterowane przez bloki innego typu, są harmonogramowane zawsze po przekroczeniu napięciowych progów logicznych na swoich wejściach. Jak wspomniano już wcześniej, dla dokładności tej konwersji niezbędna jest z reguły korekcja kroku czasowego (ang.: rollback).

Opisana w punkcie 4. struktura kolejki czasowej wymaga w tym przypadku pewnej modyfikacji. Polega ona na tym, że kolejne chwile czasu t_0, t_1, t_2,... definiujące kolejkę, określają w istocie przedziały, wewnątrz których lokowane są w uporządkowaniu czasowym poszczególne zdarzenia. Tak więc tutaj zdarzenie nie jest zapisywane na koniec listy, ale na ściśle określonej pozycji, co wymaga przeszukania danej listy zdarzeń.

Analiza DC w symulatorze iSPLICE3, prowadząca w istocie do wyznaczenia warunków początkowych w obwodzie, jest realizowana także relaksacyjnie. Dla bloków logicznych stosuje prostą symulację logiczną z zerowymi opóźnieniami, a dla pozostałych metodę Newtona. Po uzyskaniu zbieżności potencjały węzłów typu ELOGIC przyjmują wartości progowe, najbliższe obliczonym. Niestety, w niektórych przypadkach proces ten może być niezbieżny.

Dla zaradzenia tej sytuacji można także przeprowadzić wstępną analizę bloków typu ELECTRICAL i ELOGIC techniką symulacji na poziomie kluczy, zorientowaną logicznie (dotyczy cyfrowych struktur MOS), a wyznaczone stany logiczne przyjąć do określenia warunków startowych dla wspomnianej analizy Newtona. Metoda ta znacznie skraca czas analizy DC, chociaż obecność nieokreślonych stanów X wymaga specjalnych rozwiązań [1]. Ostatecznie jednak, jeśli algorytm nie jest w stanie określić stanu logicznego danego węzła w strukturze MOS, to jako warunek startowy w analizie Newtona przyjmuje się 0V. Jeśli dotyczy to bramki tranzystora, to iteracje rozpoczynają się dogodnie, bowiem od stanu jego odcięcia.

Symulator HYBRID oparty jest na algorytmie blokowej relaksacji przebiegów (por. punkt 5), który obejmuje wspólną pętlą dwa rodzaje symulacji - logiczną i elektryczną. Wykorzystanie tego algorytmu jako techniki unifikującej powoduje, że HYBRID jest zorientowany głównie na analizę układów analogowo-cyfrowych. W budowie

tego symulatora wykorzystano doświadczenia zdobyte podczas implementacji algorytmu relaksacji przebiegów zarówno w symulacji ciągłej, jak i dyskretniej [57,59].

Analizowana struktura wymaga na wstępie podziału na bloki typu logicznego i elektrycznego. W przypadku bloków elektrycznych, dla uzyskania dobrej zbieżności iteracji, lokalne sprzężenia zwrotne powinny być ukryte wewnątrz bloków, a granice między nimi powinny przechodzić przez węzły określone jako „silne". Są nimi węzły wyjściowe, sterowane przez źródła o niskiej impedancji wewnętrznej, lub węzły wejściowe bloków o wysokiej impedancji. Pełny czas symulacji jest dzielony na okna czasowe, których granice są wyznaczane automatycznie podczas wstępnej analizy cyfrowych pobudzeń zewnętrznych, tak aby w każdym oknie wystąpiło możliwie pojedyncze przełączenie. Drugim czynnikiem branym tu pod uwagę jest oszacowanie wolnej pamięci RAM komputera niezbędnej do przechowywania poszczególnych przebiegów czasowych podczas iteracji. Nadzorowane są ze względów praktycznych nie wszystkie napięcia, a jedynie tylko dotyczące węzłów zewnętrznych, w których poszczególne bloki są połączone.

Organizację symulatora HYBRID przedstawia w uproszczeniu następujący algorytm:

```
repeat {
    pobierz _ granice _ okna;
    pobierz _ pobudzenia;
    utwórz _ kolejkę;
    /*pętla relaksacyjna*/
    while (kolejka niepusta) {
        pobierz _ element _ z _ kolejki;
        if (element. typ = LOGIC) then logiczna _ analiza (element)
            else elektryczna _ analiza (element); /*typ ELECTRICAL*/
    }
    przygotuj _ dane _ dla _ następnego _ okna;
    wyprowadź _ obliczone _ przebiegi;
}
until (koniec _ okna = t_stop).
```

Na początku analizy w danym oknie czasowym uaktualniane są jego granice. Pobrane, aktywne pobudzenia pozwalają utworzyć pierwotną kolejkę bloków i rozpocząć proces relaksacyjny, który wykorzystuje technikę śladu selektywnego. Podczas analizy kolejka jest uzupełniana stosownie do propagacji sygnałów w układzie (stosowany jest algorytm Gaussa-Seidela). Aktywne bloki sterowane wprowadzane są na koniec kolejki, o ile nie są już w kolejce obecne. Harmonogramowanie to oparte jest na utworzonych automatycznie przed analizą tablicach bloków sterowanych i sterujących. Należy zaznaczyć, że kolejka nie posiada tu uporządkowania czasowego, gdyż zdarzenia (w szczególności elektryczne) dotyczą nie poszczególnych chwil czasu, ale całych przebiegów w ramach okna.

Aktywne bloki zaprzestają harmonogramowania bloków przez siebie sterowanych z chwilą stwierdzenia zbieżności iteracji (po porównaniu z poprzednimi przebiegami). Dla układów bez sprzężeń zwrotnych między blokami, z jednokierunkowo rozchodzącymi się sygnałami, rozwiązanie uzyskuje się w pojedynczej iteracji. Wówczas druga iteracja jest potrzebna tylko dla stwierdzenia zbieżności. Ostatnią czynnością w ramach obsługi okna jest wyprowadzenie zadeklarowanych przebiegów czasowych na ekran.

Technika okienkowania, obok wspomnianych wcześniej zalet, stwarza w tym przypadku także problemy. Logiczne zdarzenia na wejściach, występujące blisko końca okna czasowego, wymagają specjalnej uwagi, gdyż reakcja, którą mogą wywołać, pojawia się dopiero w następnym oknie. Dlatego też niezbędne jest dwojakie działanie. Po pierwsze, zdarzenia takie harmonogramują sterowane bloki w oddzielnej kolejce, która będzie obsługiwana w następnym oknie. Konsekwentnie także następne okno musi wtedy posiadać „nakładkę" wstecz, aby wspomniane sygnały mogły być pobrane do analizy tych bloków.

Z kolei dla wąskich impulsów typu „glitch", pochłanianych na wyjściu elementów logicznych, wymagana jest nakładka „w przód" (jeśli tworzy się on na samej granicy okien), aby można było stwierdzić obecność takiego impulsu, a następnie go odrzucić.

Natomiast w przypadku bloków elektrycznych krok całkowania przy zbliżaniu się do końca okna (stosowana jest metoda wsteczna Eulera) także wymaga specjalnej kontroli. Jest ona tak dobierany, aby zakończyć obliczenia precyzyjnie w końcu okna i uzyskać w ten sposób dokładne warunki początkowe dla rozpoczęcia całkowania w oknie następnym.

Blok elektryczny, który był aktywny w danym oknie, powinien być analizowany także w oknie następnym, jeśli napięcia na jego zaciskach kontrolnych nie ustaliły się, mimo że pobudzenia mogą być już stałe (samoharmonogramowanie). Blok taki jest bowiem nadal w stanie reakcji na sterowanie zewnętrzne (dochodzi do stanu ustalonego). Dlatego przed końcem okna kontrolowana jest ich pochodna podług czasu dla sygnałów na zaciskach tych bloków.

Do wyznaczenia warunków początkowych wykorzystywany jest zasadniczo główny algorytm analizy. Początkowo wszystkim logicznym węzłom wewnętrznym struktury przyporządkowuje się stan X, a elektrycznym 0V. W chwili zerowej wejścia niezależne stają się aktywne, uzyskując określony stan początkowy. Sygnały te propagują swobodnie przez układ, tzn. jeśli np. na którymś z wejść bramki AND stwierdzono stan X (węzeł niezainicjalizowany), to przyjmuje się natychmiast, że jest on równy 1. Stwierdzenie stanu różnego od X (w następnych iteracjach) oczywiście nie upoważnia do takiego podstawienia. W układach z wieloma rozwiązaniami (przerzutniki) otrzymuje się rozwiązanie jednoznaczne, które zależy tylko od kolejności analizy elementów, tzn. praktycznie od sposobu wprowadzania danych. Kryterium zbieżności wymaga tu jednak modyfikacji, bowiem w układach sekwencyjnych istnieje tendencja do „przesuwania" stanu ustalonego (o czas propagacji) w kolejnych iteracjach do coraz odleglejszych chwil czasu. Jednak stany logiczne pozostają nie zmienione. Zbieżność wykrywana jest wówczas w oparciu o zgodność ostatniego stanu logicznego w oknie, w kolejnych dwu iteracjach, co nie jest już trudne. Ostatecznie wszystkie węzły uzyskują stan określony.

Wyznaczenie warunków początkowych odbywa się zasadniczo w jednym arbitralnie ustalonym oknie czasowym, ale dla bloków o dużych stałych czasowych niezbędne może być przejście do okien następnych, aż do uzyskania stanu ustalonego w zdefiniowanych węzłach skrajnych.

8. Związek z językami opisu sprzętu

Aktualnie symulatory układów elektronicznych (w tym hybrydowe) odgrywają szczególnie dużą rolę, jeśli stanowią integralną część określonego środowiska projektowego. Podstawową rolę w tym przypadku spełniają tzw. języki opisu sprzętu (ang.: HDL). Stoją one u podstaw oprogramowania wspomagającego lub automatycznie realizującego proces projektowy w ramach tego środowiska. Rola języków HDL nie ogranicza się do opisu struktury układów (tj. specyfikacji), ale obejmuje ściśle także aspekty modelowania i weryfikacji i symulacji [50,10], tworząc w ten sposób podstawy kompletnego środowiska projektowego.

W latach osiemdziesiątych szczególną rolę zaczął odgrywać język VHDL, zyskując stopniowo miano standardu światowego [52,53,10]. Początkowo był on przeznaczony wyłącznie do projektowania struktur cyfrowych, dla których podstawowe pojęcia służące ich modelowaniu i symulacji można było stosunkowo łatwo zunifikować (np. zdarzenia logiczne). Jednak na początku lat dziewięćdziesiątych postęp w technologii złożonych układów analogowych i analogowo-cyfrowych rzucił nowe wyzwanie opracowania standardu języka opisu struktur analogowych i mieszanych (A/C). Mimo znaczących doświadczeń i osiągnięć w zakresie języków opisu układów analogowych (np.[54]), realizacja tego zadania nie jest łatwa. Prowadzone w Europie i USA od 1992 roku prace nad nowym standardem VHDL-A są w toku, a jako termin ich zakończenia wyznaczono rok 1997.

W niniejszym punkcie przedstawione zostaną jedynie najważniejsze aspekty i założenia stojące u podstaw prac nad standardem VHDL-A [47].

Przyjęto, że nowy standard VHDL-A musi być kompatybilny z istniejącym dla struktur cyfrowych standardem VHDL (IEEE 1076). Standard ten nie może stwarzać ograniczeń ze względu na klasę modeli analogowych i poziomy abstrakcji, ani ze względu na algorytmy symulacyjne. Język musi posiadać charakter behawioralny, dając możliwość definiowania układów w oparciu o relacje lub procedury, począwszy od elementów RLC, a skończywszy na makromodelach struktur złożonych (np. wzmacniaczy operacyjnych, przetworników A/C). Zakłada się, że VHDL-A oprócz symulacji (wyznaczenie warunków początkowych i przebiegów w stanie nieustalonym) powinien realizować co najmniej: małosygnałową analizę AC, analizę szumową i analizę zniekształceń nieliniowych.

Z wymienionych tu ogólnych założeń wynika, że VHDL-A nie może definiować żadnych algorytmów symulacji ciągłej (analogowej), a jedynie umożliwiać ich implementację w ramach standardu. Równocześnie wszystkie podstawowe rodzaje symulacji ciągłej powinny być akceptowane bez ograniczeń. Wynikają stąd wymagania dla standaryzacji przekazywania i pobierania danych z tych symulatorów, kontroli dokładności obliczeń, a dodatkowo i zapewnienie zunifikowanego sprzęgu między subsymulatorami ciągłym i dyskretnym.

W określonych przypadkach rodzaj symulacji powinien być wybierany automatycznie.

W nadzorowaniu procesu symulacji można wyróżnić następujące zadania:

* wyznaczenie niezbędnych tolerancji błędu dla stosowanych algorytmów iteracyjnych,
* wybór właściwej techniki symulacyjnej dla całego projektu lub poszczególnych bloków,
* określenie formatu dla ciągłych przebiegów wyjściowych (w języku VHDL-A powinien być określony jedynie mechanizm wprowadzania i wyprowadzania danych z symulatora).

Jak stąd wynika, symulatory implementowane w nowym środowisku VHDL-A nie powinny tracić swojej tożsamości, a jedynie być efektywnie ekspolatowane. Symulatory hybrydowe stanowią tu niewątpliwie najbardziej skomplikowany przypadek, m.in. ze względu na omówione w p.7 problemy synchronizacji czasu i konwersji sygnałów.

9. Podsumowanie

W symulacji hybrydowej od samego początku starano się wykorzystać najlepsze cechy dyskretnej symulacji logicznej i ciągłej symulacji elektrycznej. Pierwsza z nich posłużyła jako wzorzec w zakresie sterowania zdarzeniami, śladu selektywnego, jednokierunkowej propagacji sygnału i modularyzacji struktury. Druga natomiast wyznaczała możliwe do osiągnięcia granice dokładności obliczeń. Przeszczepienie cech symulacji dyskretnej na grunt symulacji elektrycznej stało się możliwe dzięki dekompozycji układu, technice relaksacji oraz zdefiniowaniu pojęcia zdarzenia analogowego. Dekompozycja spełnia tu co najmniej dwie role. Po pierwsze, pozwala

reprezentować poszczególne podobwody na różnych poziomach abstrakcji, co jest wyrazem kompromisu między wymaganym czasem symulacji a dokładnością uzyskanych przebiegów. Po drugie, w odniesieniu do podobwodów przedstawianych na tym samym poziomie, służy przyspieszeniu obliczeń, jak np. w blokowej relaksacji przebiegów. W przypadku układów A/C dekompozycja posiada charakter naturalny. Podobna uwaga dotyczy także złożonych projektów, których części składowe powstają oddzielnie na różnych poziomach abstrakcji. Wówczas korzyści wynikające z bezpośredniego wykorzystania symulacji hybrydowej są ewidentne.

Technika symulacji hybrydowej może być również z powodzeniem stosowana do weryfikacji czasowej (np.[60]). Nowsze prace z dziedziny symulacji hybrydowej proponują zwiększenie liczby poziomów abstrakcji (mixed-level), obejmując dziedzinę transformacji Laplace'a oraz transformacji Z (np.[48]).

Wśród problemów nie w pełni jeszcze rozwiązanych wymienić trzeba zagadnienie symulacji błędów w zastosowaniu do testowania układów A/C. Symulacja błędów analogowych dotyczy w znacznej mierze również struktur cyfrowych, dla których szczegółowe badania wykazały, że 80% błędów ma swoje źródło w zjawiskach elektrycznych [1][(nie są to proste błędy typu sklejenia z 0 lub z 1).

Ostatecznie, należy także podkreślić, że aktualnym stymulatorem dalszego postępu w zakresie implementacji technik symulacyjnych są języki opisu sprzętu, a w szczególności opracowywany VHDL-A.

LITERATURA

1. Saleh R.A., Newton A.R.: Mixed-Mode Simulation, Kluwer Academic Pub. 1990.
2. Schwarz A.F.: Computer-Aided Design of Microelectronic Circuits & Systems Vol.2, Academic Press, 1987.
3. Rao V., Overhauser D., Hajj I., Trick T.: Switch-Level Timing Simulation of MOS VLSI Circuits, Kluwer Academic Pub., Boston, MA., 1989.
4. White J., Sangiovanni-Vincentelli A.L.: Relaxation Techniques for the Simulation of VLSI Circuits, Kluwer Academic Pub., Boston, Mass., 1986.
5. Antognetti P., Pederson D.O., De. Man H., (Eds): Computer Design Aids for VLSI Circuits, NATO adv.Study, Noordhoff & Sijithoff Gronningen 1981.

6. Breuer M.A., Ed.: Digital Design System Automation: Languages, Simulation and Data Base, Computer Science Press, 1975.
7. Breuer M.A., Friedman A.D.: Diagnosis and Reliable Design of Digital Systems, Pitman Pub.Ltd. 1976, Computer Science Press Inc.
8. Chua L.O., Lin P.M.: Komputerowa analiza układów elektronicznych, WNT, Warszawa 1981.
9. Ogrodzki J.: Komputerowa analiza układów elektronicznych, PWN, Warszawa 1994.
10. Mermet J.P., Ed.: Fundamentals and Standarts in Hardware Description Languages, Kluwer Academic Pub., 1993.
11. Ortega J.M., Rheinbold W.C.: Itrative Solution of Nonlinear Equations in Several Variables, New York, Akademic, 1970.
12. Mead C.A., Conway L.A.: Introduction to VLSI Systems, Addison-Wesley Pub, Comp. 1980.
13. Weste N., Eshraghian K.: Principles of CMOS VLSI Design, Addison-Wesley Pub.Comp., 1985.
14. Maliniak L.: A/D Simulator an Expanding Array of Choises, Electronic Design, Vol.42, No.25, Dec.1994.
15. Hatchel G.D., Sangiovanni-Vincentelli A.L.: A Survey of Third-Generation Simulation Techniques, Proc.IEEE, Vol.69, No.10, pp.1264-1280, Oct.1981.
16. Acuna E. et al.: iSPLICE3: A New Simulator for Mixed Analog/Digital Circuits, Proc.of CICC, pp.1311-1314, May 1989.
17. Breuer M.A.: Recent Developments in the Design and Analysis of Digital Systems, Proc.IEEE, vol.60, No.1, pp.12-27, Jan.1972.
18. Wilcox P.: Digital Logic Simulation at the Gate and Functional Level, Proc.of 16th DAC, pp.242-248, 1979.
19. White J., Sangiovanni-Vincentelli A.L.: RELAX2: A New Waveform Relaxation Approach for the Analysis of LSI MOS Circuits, Proc.of ISCAS, 1983.
20. Pederson D.O.: A Historical Review of Circuit Simulation, IEEE Trans. Vol.CAS 31, No.1, pp.103-111, Jan.1984.
21. Nagel L.W.: SPICE2: A Computer Program to Simulate Semiconductor Circuits, Memo. No.ERL-M520. Electr.Research Labs., Univ.of California, Berkeley 1975.

22. Chawla B.R. et al.: MOTIS - An MOS Timing Simulator, IEEE Trans. Vol.CAS-22, pp.901-909, 1975.
23. Newton A.R., Sangiowanni-Vincentelli: Relaxation-Based Electrical Simulation, IEEE Trans. VolCAD-3, No.4, pp.308-330, Oct.1984.
24. Arnout G., DE Man H.: The Use of the Threshold Functions and Boolean-Controlled Network Elements for Macromodeling of LSI Circuits, IEEE J.of Solid St.Cir., Vol.SC-13, No.3, June 1978, pp.326-332.
25. Newton A.R.: Timing, Logic and Mixed-Mode Simulation for Large MOS IC in [5] pp. 175-240.
26. De Man H. et al.: Mixed-Mode Simulation Techniques and Their Implementation in Diana in [5] pp.113-174.
27. Fan S.P., Hssueh M.Y., Newton A.R.: MOTIS-C: a New Circuit Simulator for MOS LSI Circuits Proc. ISCAS New York 1978, pp.700-703.
28. Hayes J.P.: A Unified Switching Theory with Applications to VLSI Design, Proc.IEEE, Oct.1982, pp.1140-1151.
29. Bryant R.E.: A Schwitch-Level Model and Simulator for MOS Digital Systems, Trans. IEEE, Vol. C-33, Feb.1984.
30. Bryant R.E.: A Survey of Switch-Level Algorithms, IEEE Design and Test, Aug.1987 pp.26-38.
31. Terman C.: RSIM - A logic-Level Timing Simulator, Proc. of ICCD, Port Chester, NY, 1983.
32. Rubinstein J. et al.: Signal Delay in RC Tree Networks, IEEE Trans. Vol.CAD-2, No.3, July 1981, pp.202-211.
33. Pillage L.T. et al.: Asymptotic Waveform Evoluation for Timing Analysis, IEEE Trans, Vol.CAD-9, No.4, Apr.1990, pp.352-366.
34. Visweswariah Ch. et al.: Piecewise Approximate Circuit Simulation, IEEE Trans. Vol.CAD-10, No.7, July 1991, pp.861-870.
35. Rao V.B., Trick T.N.: Accurate Multiple Delay Calculations for MOS Circuits, Proc of ICCC, pp.439-442, 1982.
36. Rao V.B., Trick T.N., Hajj I.N.: A Table Driven Delay-Operator Approach to Timing Simulation of MOS VLSI Circuits, Proc. of ICCD, pp.445-448, 1983.

37. Rabbat N.B., et al: A Multilevel Newton Algorithm with Macromodeling and Latency for the Analysis of Large-Scale Nonlinear Networks in the Time Domain, IEEE Trans. Vol.CAS-26, No.9, pp.733-741, Sept.1979.
38. Ousterhaut J.K.: CRYSTAL: A Timing Analyzer for nMOS VLSI Circuits, Proc.of 3rd Caltech VLSI Conf., 1983, pp.57-70.
39. Joupii N.P.: Timing Analysis for nMOS VLSI, Proc.of 20th DAC, 1983, pp.411-418.
40. Hwang S.H. et al.: An Accurate Delay Modeling Technique for Switch-Level Timing Verification, Proc.of 23rd DAC, 1986, pp.227-233.
41. Szygenda S.A., Thompson E.W.: Modeling and Digital Simulation for Design Verification and Diagnosis, IEEE Trans., Vol.C-25, pp.1242-1253, Dec.1976.
42. Kim Y.H. et al.: Electrical-Logic Simulation, Proc. of Int. Conf. on CAD, pp.7-10, Nov. 1984.
43. Tsao D. et al.: A Fast Timing Simulation for Digital Mos Circuits, Proc. of ICCAD, Nov.1985, pp.185-187.
44. Odryna P. et al.: A Workstation-based Mixed-Mode Circuit Simulator, Proc.of 23rd DAC, pp.186-191, June 1986.
45. Kim Y.H. et al.: Electrical-Logic Simulation and its Applications, IEEE Trans. Vol.CAD-8, pp.8-22, Jan. 1988.
46. Heydemann M.E.: Implementation Issues in Multiple Delay Switch Level Simulation, Proc. of ICCC, pp.46-49, 1982.
47. Vachoux A., Nolan K.: Analog and Mixed-Level Simulation with Implications to VHDL, in [10] pp.281-329.
48. Chadha R. et.al.: M^3- A Multilevel Mixed-Mode Mixed A/D Simulator, Proc. IEEE, Vol.CAD-11, No.5, May 1992, pp.575-585.
49. Ruan G. et al.: Analog Functional Simulator for Multilevel Systems, IEEE Trans. Vol.CAD-10, No.5, May 1991.
50. Pawlak A.: A Tutorial Guide to Modern Hardware Description and Design Languages in „Microcomputers, Usage & Design" K.Waldschmidt, B.Myrhraug (Eds.), Elsevier Science Pub. EuroMicro 1985.
51. Barbacci M.R.: Instriction Set Processor Specification (ISPS): The notation and its applications, IEEE Trans. C-30, pp.24-40, Jan, 1981.

52. Special Issue on VHDL: IEEE Design & Test, April 1986.
53. Gizdoń H. i in.: VHDL-ADA języków opisu i projektowania sprzętu, Informatyka 10/1988.
54. Bukat D. et al.: Enhanced Network Description Language for Behavioral Circuit Description Simulators Including External Models, Proc. of 14th KKTOiUE, Waplewo, Poland, Oct. 1991, pp.160-165.
55. Dąbrowski J.: Design of Multilevel Mixed-Mode Simulator for LSI/VLSI Circuits, Proc. of ISCAS'88, pp.1685-1688, 1988.
56. Dąbrowski J.: Mixed-Mode Timing Verification for VLSI Designs, AMSE Press, Vol.4, pp.129-138, 1989.
57. Dąbrowski J.: LOGSIM1: Symulator logiczny układów bramkowych Mat. XIII KKTOiUE Bielsko-Biała 1990, str.259-264.
58. Dąbrowski J., Konopacki J.: Symulator hybrydowy dla złożonych układów A/C, Raport Instytutu Elektroniki, Pol.Śl. 1991 (grant MEN nr DNS-T/05/054/90-2).
59. Dąbrowski J., Konopacki J.: Experience with Block Waveform Relaxation for Analog Networks, Proc.of 14th KKTOiUE, Waplewo, Poland, Oct.1991, pp.145-150.
60. Dąbrowski J.: Efficient Timing Verification via Mixed-Mode Technique, Microprocessing and Microprogramming 34, 1992, pp.183-186, North-Holland.
61. Dąbrowski J., Konopacki J.: HYBRID: A Mixed-Mode Simulator for A/D Networks Proc. of 15th KKTOiUE, Szczyrk, Poland, Oct. 1992, pp.103-108.
62. Dąbrowski J., Konopacki J.: Delay Models for Mixed-Mode Timing Verification of Bipolar Circuits, Proc. of 16th KKTOiUE, Kołobrzeg, Poland, Oct. 1993, pp.193-198.
63. Dąbrowski J., Konopacki J.: A/D Networks Functional-Level Macromodelling with Piecewise Linear Signals, Proc. of 17th KKTOiUE, Polanica, Poland, Oct.1994, pp.353-358.
64. Dąbrowski J.: Functional-Level Analog Macromodelling with Piecewise Linear Signals, Proc. of EURO-DAC'95, Brighton, England, Sept.1995.
65. Sobaszek Z.: Symulator dla układów PLD, praca dyplomowa, Instytut Elektroniki, Pol.Śl. 1992.

66. Jeziorowski M.: Symulator logiczny typu „switch-level" współpracujący z pakietem ORCAD-STD, praca dyplomowa, Instytut Elektroniki, Pol.Śl. 1993.

67. Olczyk K.: Analogowy symulator funkcyjny dla układów A/C współpracujący z pakietem ORCAD-STD, praca dyplomowa, Instytut Elektroniki, Pol.Śl. 1993.

Recenzent: Prof.dr hab.inż. Stanisław Kuta

Wpłynęło do Redakcji 15.10.1994 r.

Abstract

In the paper an overview of different simulation techniques exploited in mixed-mode simulation is presented. Discrete- and continuous simulation are discussed in context of their use in the mixed-mode simulation. Efficient simulation of large networks is shown to be based on event-driven, selective-trace algorithms. The network is usually partitioned and analyzed within an iterated, relaxation loop. Detailed examples of two particular mixed-mode simulators iSPLICE3 [1] and HYBRID [55] are included. Some comments in this paper are based on the experience gained while dealing with network simulation at the Institute of Electronics of Silesian Technical University in the last few years.