ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1987

Seria: AUTOMATYKA z. 83 Nr kol. 888

Wojciech SAKOWSKI

WYKORZYSTANIE JĘZYKA OPISU SPRZĘTU MODLAN W DIAGNOSTYCE SYSTEMÓW MIKROPROCESOROWYCH

Streszczenie. W artykule zaproponowano metodę gromadzenia danych testowych w trakcie symulacji systemu cyfrowego. Przedstawiono opis analizatora sygnatur w języku MODLAN i sposób jego wykorzystania do symulacji pomiaru sygnatur w typowym systemie mikroprocesorowym.

1. Wpływ wzrostu złożoności systemów cyfrowych na technikę ich projektowania i testowania

1.1. Wzrost niezawodności i obniżka kosztów - te dwie podstawowe korzyści płynące z rozwoju technologii układów scalonych umożliwiającego stałe zwiększanie stopnia ich integracji - pozwalały na budowę coraz bardziej skomplikowanych systemów cyfrowych. Dostępność układów MSI i LSI stanowiących bloki funkcjonalne o znacznym stopniu złożoności ograniczyła mocno użyteczność tradycyjnych metod syntezy układów cyfrowych opartych na algebrze Boole'a i teorii automatów. Zarówno w procesie projektowania systemów cyfrowych, jak i cyfrowych układów scalonych zaczęto na szeroką skalę stosować komputerowe wspomaganie projektowania. Możliwość wykorzystania komputera do obliczeń (np. minimalizacji formuł boolowskich), do tworzenia bazy danych projektu integrującej etapy procesu projektowego i ułatwiającej prowadzenie dokumentacji, a przede wszystkim możliwość weryfikacji projektu w procesie symulacji znacznie usprawniła przebieg projektowania.

Złożoność projektów i związana z nią liczebność zespołów projektowych oraz posługiwanie się komputerem stworzyły potrzebę posługiwania się formalnymi metodami specyfikacji projektu. Tak powstały języki opisu sprzętu (ang. - hardware description languages) w początkowym okresie stanowiące jedynie konwencje notacyjne, aby później stać się językami modelowania. Opis sprzętu w języku modelowania stanowi definicję źródłową dla procesu symulacji komputerowej. Jednym z takich języków jest język MODLAN [1, 3]. Umożliwia on, wraz ze związanym z nim systemem modelowania, komputerowe wspomaganie procesu projektowania (zarówno metodą zstępującą jak i wstępującą) poprzez weryfikację kolejnych decyzji projektowych w procesie symulacji projektowanego systemu, którego bloki mogą być zdefiniowane na różnych poziomach opisu: behawioralnym, funkcjonalnym lub strukturalnym.

System modelowania oparty na MODLANIE jest szczególnie wygodny przy projektowaniu systemów mikroprocesorowych.

1.2. Równolegle do problemów, jakie wzrost złożoności realizowanych systemów cyfrowych stawiał przed projektantami, rosły problemy związane z testowaniem tych systemów. I w tej dziedzinie komputer okazał się pomocnym narzędziem. Ułatwia generację wzorów testowych na podstawie wyników komputerowej symulacji uszkodzeń oraz umożliwia gromadzenie danych testowych i ich porównanie z odpowiednimi wzorcami. Obecnie istnieje już oprogramowanie zapewniające kompatybilność danych testowych uzyskanych z określonych urządzeń (np. z analizatora cyfrowego DAS 9000 firmy Tektronix czy testera Sentry Series 20 firmy Fairchild [11]) z uzyskanymi w czasie symulacji.

Ilość danych testowych dla złożonych układów utrudnia bądź uniemożliwia korzystanie z tych metod w serwisie. Przełomem stała się wprowadzona przez firmę Hewlett-Packard metoda analizy sygnatur [4, 9, 10]. Umożliwia ona kompresję dowolnej długości ciągu danych pomiarowych w sygnaturę złożoną z czterech cyfr heksadecymalnych kosztem rezygnacji ze 100% pewności wykrycia błędu. Jest bardzo wygodna przy testowaniu systemów mikroprocesorowych, choć dla jej pełnego wykorzystania konieczne są pewne środki sprzętowe i programowe, które należy opracować jeszcze na etapie projektowania systemu, aby ułatwić jego testowanie. To oddziaływanie problemów testowania i narzędzi testujących na proces projektowy nie jest zresztą jednokierunkowe i nie dotyczy jedynie analizy sygnatur. Zdolność urządzenia czy układu scalonego do łatwego testowania (ang. testability) staje się jednym z ważnych wymagań projektowych. Jednocześnie narzędzia projektowania takie jak języki opisu sprzętu w kategoriach funkcjonalnych - zaczyna się wykorzystywać do modelowania uszkodzeń (dotyczy to głównie układów scalonych), choć do tej pory opierało się ono na opisie struktury bramkowej układu. Wszystko wskazuje na to, że dalszy wzrost złożoności systemów cyfrowych będzie stymulował zazębianie problemów projektowania i testowania układów. Zalety analizy sygnatur każą natomiast oczekiwać jej rosnącego znaczenia w testowaniu urządzeń cyfrowych i cyfrowych układów scalonych. Obecnie stosowana jest ona szeroko w diagnostyce systemów mikroprocesorowych.

2. Możliwości opracowania słownika sygnatur systemu cyfrowego w procesie projektowania wspomaganego komputerem

2.1. Obecne metody gromadzenia dokumentacji diagnostycznej

Jakkolwiek dokumentację diagnostyczną systemu mikroprocesorowego przewidzianego do testowania metodą analizy sygnatur opracowuje się w trakcie projektowania urządzenia, gdyż - jak już wspomniano - w projekcie uwzględ-

nić należy odpowiednie środki sprzętowe i programowe ułatwiające testowanie, to opracowanie słownika sygnatur możliwe jest dopiero po zbudowaniu i uruchomieniu (np. przy użyciu analizatora logicznego) kilku prototypów. Przy stosowaniu bardziej wyrafinowanego sprzętu uruchomieniowego - jak emulator diagnostyczny - sygnatury można zebrać w trakcie uruchamiania urządzenia [6]. Zarówno w jednym, jak i w drugim wypadku na etapie projektowania sygnatury nie są znane.

2.2. Wykorzystanie symulacji do opracowania słownika sygnatur

Modelowanie systemu cyfrowego umożliwia dokładne śledzenie jego pracy jeszcze przed wykonaniem prototypu. Odpowiednie zlecenia sterujące procesem symulacji [1, 2] umożliwiają obserwację sygnałów w różnych punktach systemu i stanów jego rejestrów wewnętrznych w dowolnych momentach czasu.

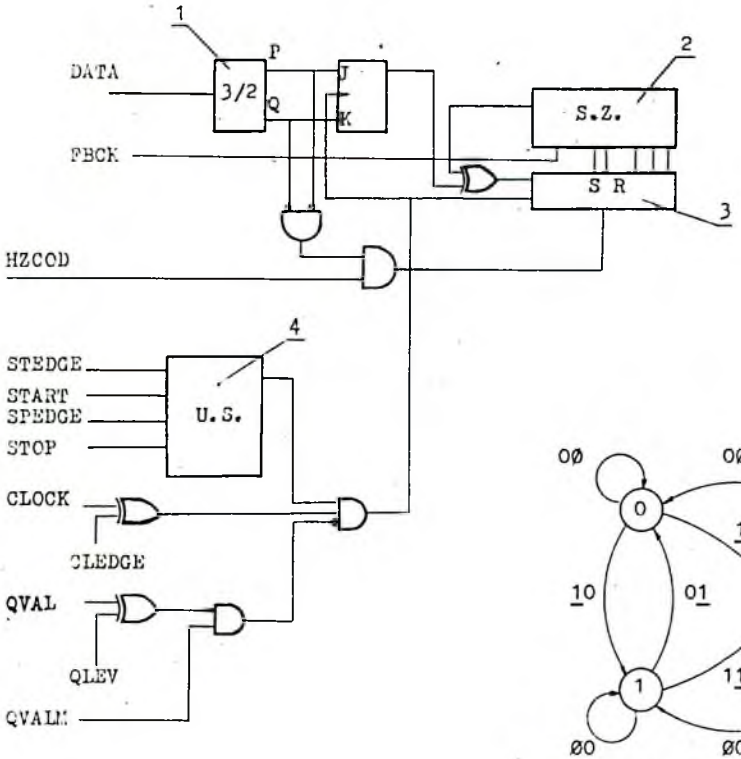
Idea proponowanej w tym artykule metody jest następująca: do modelowanego systemu należy dołączyć model analizatora sygnatur i sterować procesem symulacji w sposób odpowiadający strategii testowania systemu mikroprocesorowego z wykorzystaniem analizy sygnatur. Umożliwi to uzyskanie poszukiwanych danych. Z uwagi na prostotę realizacji i efektywność (pod względem czasu symulacji) wskazane jest użycie modelu funkcjonalnego analizatora sygnatur.

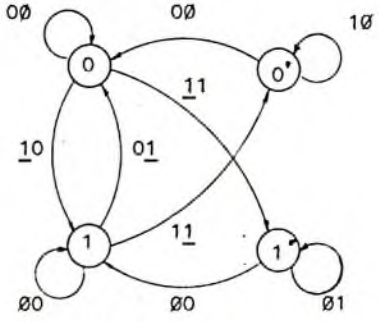
Poniżej przedstawiono opis funkcjonalny analizatora sygnatur w języku MODLAN. Jako przykład wybrany został analizator PAS-80 opracowany w Instytucie Elektroniki Politechniki Śląskiej [7]. Uproszczony schemat ideowy tego urządzenia, a ściśle rzecz biorąc jego części odpowiedzialnej za generowanie sygnatury na podstawie danych testowych przedstawia rysunek 1.

2.3. Opis analizatora sygnatury PAS-80 w języku MODLAN

Ze względów, które omówione będą niżej, analizator opisany został jako złożenie dwóch bloków funkcjonalnych DATACOMP i CONTROL. Odpowiadają one dwóm częściom układu oddzielonym od siebie na rys. 1 przerywaną linią. W poniższym opisie połączone są one ze sobą mechanizmami opisu strukturalnego MODLAN-u.

```
SMODULE SIGAN  (DATA,   %  WEJSCIE DANYCH
                HZCOD,  %  WYBOR KODU DLA STANU WYSOKIEJ IMPEDANCJI
                FBCK,   %  WYBOR STRUKTURY SPRZĘŻENIA ZWROTNEGO
                START, STOP, CLOCK,
                STEDGE, SPEDGE, CLEDGE, % Wybór aktywnego zbocza
                                        % sygnałów start, stop, clock
                qval, QLEV, QVALM)      % sygnał kwalifikatora wyboru
                                        % poziomu jego aktywności,
                                        % wyboru trybu pracy z kwali-
                                        % fikatorem
```





Graf przejść układu sterującego

Rys. 1. Fragment układu analizatora sygnatury PAS-80

1. Sonda trójstanowa: DATA=H Q=L, P=H; DATA=L Q=H, P=L; DATA=HZ P=Q=L;
2. Układ sprzężenia zwrotnego; 3. Rejestr przesuwny dwuwejściowy; 4. Układ sterujący

Fig. 1. A part of the signature analyzer PAS-80

1. 3-state probe: DATA=H Q=L, P=H; DATA=L Q=H, P=L; DATA=HZ Q=P=L;
2. A feedback circuit; 3. Two-input shift register; 4. Control circuit;

```
BEGIN
   PMODULE  DATACOMP (DATA, HZCOD, FBCK, % znaczenie jak wyżej
   BEGIN                CL)              % sygnał taktujący rejestry
    REGISTER :  JK, SHIFT   1-16 ;
    SBEGIN
     AT CL DO
      IF   DATA <> ∨ THEN
          PARBEGIN
           JK ← DATA;
```

```
        SHIFT  [1-16]←((SHIFT [6] & FBCK $' FBCK & SHIFT [7] ) #
                       SHIFT [9] # SHIFT [12] # SHIFT [16] # JK) @
                       SHIFT [1-15]

      PARAEND
                    ELSE
      SHIFT [1-16] ← ((SHIFT [6] & FBCK $ 'FBCK  SHIFT [7] ) #
                       SHIFT [9] # SHIFT [12] # SHIFT [16] # JK) @
                      (HZCOD  #  SHIFT [8] ) @ SHIFT [9-15]

  SEND
 END DATACOMP
FMODULE  CONTROL  (START, STOP, CLOCK, STEDGE, SPEDGE, CLEDGE, QVAL,
                   QLEV, QVALM : CL) % znaczenie jak wyżej
 BEGIN
   REGISTER :  STATE 2;
   SBEGIN
     AT  CLOCK # CLEDGE DO
        ON STATE  CASE 00 B DO ON  (START # STEDGE) @ (STOP # SPEDGE)
                  CASE 00B, 01B  DO NOOP
                  CASE 10B       DO STATE ← 10B
                  CASE 11B       DO STATE ← 11B
                  END

          CASE 01B  DO ON (START # STEDGE) @ (STOP # SPEDGE)
                CASE 10B, 11B  DO NOOP
                CASE 00B, 01B  DO STATE ← 00B
                END

          CASE 10B  DO ON (START # STEDGE) @ (STOP # SPEDGE)
                CASE 10B, 00B  DO NOOP
                CASE 11B       DO STATE ← 01B
                CASE 01B       DO STATE ← 00B
                END

          CASE  11B  DO ON (START # STEDGE) @ (STOP # SPEDGE)
                 CASE 10B, 00B . DO STATE ← 10B
                CASE  11B, 01B  DO NOOP
                END

          END

          CL ← STATE [0] & (CLOCK # CLEDGE) & (QVALM & (QVAL # QLEV))
```

```
    SEQEND
END  CONTROL
DATA COMP REG;  CONTROL  CTR;      % Deklaracja modułów REG i CTR o
                                   % typach DATACOMP I CONTROL odpowiednio
REG [1-3] ,  CTR [1-9]  :=SIGAN    % przypisanie wejściom bloków funkcjo-
                                   % nalnych REG i CTR wejść analizatora
END  SIGAN;
```

Komentarz umieszczony w opisie wyjaśnia znaczenie wejść analizatora sygnatur PAS-80. Wyjścia pominięto, gdyż sygnaturę można odczytać bezpośrednio z rejestru SHIFT zleceniem WRITE języka sterowania symulacją MODSIM [1, 2].

Sam opis funkcjonalny w języku MODLAN ma składnię zbliżoną do języka Pascal i nie powinien nastręczać trudności w interpretacji, jeśli uwzględnimy następujące wyjaśnienia:

\# - operator EX-OR (sumy modulo dwa)
& - operator AND (iloczyn logicznego)
$ - operator OR (sumy logicznej)
@ - operator konkatencji, czyli łączenia bitów rejestrów bądź wyrażeń binarnych w dłuższe łańcuchy bitowe
⟵ oznacza instrukcję przesłania

AT CL DO - tzw. instrukcja dynamicznego transferu powodująca wykonanie instrukcji umieszczonej po DO o ile wystąpi zmiana sygnału CL z 0 na 1. Szczegółowy opis języka MODLAN czytelnik znajdzie w literaturze [1, 3].

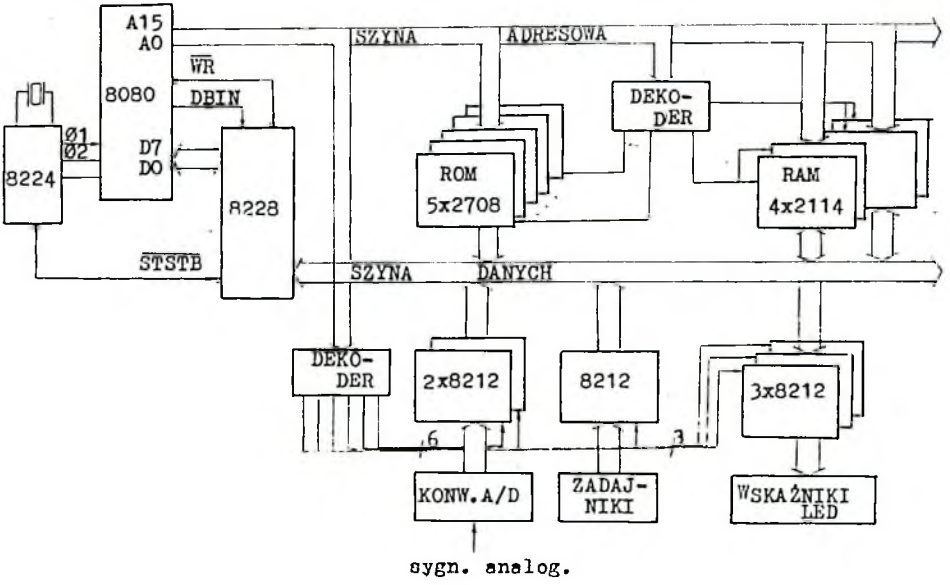
Wyróżnienie w powyższym opisie dwóch niezależnych bloków funkcjonalnych, nazwijmy je umownie blokiem kompresji danych i blokiem sterującym, zostało uczynione ze względów praktycznych.

Deklarując w opisie modelowanego systemu tablicę modułów DATACOMP można zbierać w jednym przebiegu symulacji sygnatury w wielu punktach. O ile sygnały sterujące zapisem danych są identyczne dla wszystkich punktów pomiarowych, wystarczy zadeklarować tylko jeden moduł CONTROL. Jeśli konieczne jest użycie różnych sygnałów sterujących, należy użyć jeden moduł CONTROL na każdy ich zestaw. Postępowanie takie pozwala na znacznie lepsze wykorzystanie czasu symulacji kosztem nieco większej zajętości pamięci.

3. Zastosowanie symulacji do uzyskania sygnatur w specjalizowanym systemie mikrokomputerowym CRISTALDIGRAF NC 6

Jako przykład wykorzystania modelu funkcjonalnego analizatora sygnatur przedstawiony zostanie opis systemu mikrokomputerowego CRISTALDIGRAF NC z odpowiednio włączonymi modułami DATACOMP I CONTROL.

Rysunek 2 przedstawia schemat blokowy systemu CRISTALDIGRAF NC, przy czym dla jego uproszczenia pominięto pewną liczbę bramek i przerzutników



Rys. 2. Architektura systemu mikrokomputerowego CRISTALDIGRAF NC
Fig. 2. An architecture of microcomputer system CRISTALDIGRAF NC

służących do uzyskania sygnałów sterujących magistrali MUBUS. Mikrokomputer CRISTALDIGRAF NC jest specjalizowanym systemem cyfrowym do analizy krzywych krzepnięcia odlewów. Opracowany został w Instytucie Elektroniki Politechniki Śląskiej we współpracy z Instytutem Odlewnictwa naszej Uczelni.

Szczegółowego opisu strukturalnego systemu CRISTALDIGRAF NC w języku MODLAN autor nie podaje, gdyż szczegóły konstrukcji tego systemu nie są istotne dla prezentowanej metody.

Przyjmijmy więc założenie, że opis strukturalny systemu, którego sygnatury chcemy zdejmować, został uzyskany w procesie projektowym i jest dostępny. Opis taki wykorzystamy jako moduł strukturalny wchodzący w skład większej struktury zawierającej obok modelowanego systemu pewną ilość modułów DATACOMP I CONTROL, które wykorzystane będą do "pomiaru" sygnatur. Jako terminale projektowanego systemu należy przyjąć wszystkie sygnały potrzebne do symulowania testu z wykorzystaniem analizatora sygnatur.

3.1. Diagnostyka systemu mikrokomputerowego CRISTALDIGRAF NC

Z uwagi na typową dla systemów mikroprocesorowych struktura, w której można wyróżnić procesor, pamięć ROM i RAM oraz porty wejściowe i wyjściowe, mikrokomputer CRISTALDIGRAF NC jest dobrym przykładem do zademonstrowania strategii testowania systemu mikroprocesorowego z wykorzystaniem analizy sygnatur. Testowany jest on metodą "rozszerzonego jądra" [4,9,10]. Pierwszy etap to tzw. test biegu jałowego (ang. free run). Realizuje się go przez rozłączenie magistrali danych między procesorem 8080 a kontrolerem 8228 i wymuszenie na wejściu mikroprocesora instrukcji MOV A,A (za pomocą odpowiednio dołączonych rezystorów). Powoduje ona inkrementację licznika rozkazów i systematyczny obieg przestrzeni adresowej mikroprocesora. Pozwala to na wstępne przetestowanie procesora wraz z zegarem i kontrolerem, a także szyny danych i szyny adresowej oraz pamięci ROM. Pamięci RAM oraz porty wejściowe i wyjściowe są w tym czasie odłączone od systemu.

Tabela 1

Sygnały sterujące zapisem danych w czasie testowania mikrokomputera CRISTALDIGRAF NC

	TEST		CLOCK		START		STOP	
SYMB.	Nr	Obiekt testu	źródło	zb.	źr.	zb.	źr.	zb.
TASFR	5	Kernel+ROM	DBIN	⌠	A15	⌠	A15	⌠
TASPT	6	porty wejś.	DBIN	⌠	A15	⌠	A15	⌡
D	4	dekodery	I/OW+DBIN	⌡	A15	⌠	A15	⌠
PO	3	PORTY wyjś.	I/OW	⌠	A15	⌠	A15	⌡
R	2	pamięć RAM	CS	⌠	A15	⌡	A15	⌠
K	1	konwerter A/C	ADPERLOW	⌠	A15	⌠	A15	⌡

Zbierając dane w czasie testu free run wejścia sterujące analizatora sygnatury łączy się następująco: clock - dbin, start i stop - A15, wszystkie z aktywnym zboczem narastającym. Dobór sygnałów sterujących i aktywnych zboczy do poszczególnych testów w mikrokomputerze CRISTALDIGRAF NC przedstawiony jest w tabeli 1.

W przypadku stwierdzenia uszkodzenia pamięci ROM można przy użyciu mikroprzełączników odłączyć kolejno jej bloki składowe w celu wykrycia bloku wadliwego.

Pozostałe testy realizowane są programowo (ang. software driven). Sprawdza się w nich dekodery, pamięć RAM, porty i przetwornik analogowo-cyfrowy. Dane zbierane są odpowiednio z wyjść dekoderów magistrali danych i wejść lub wyjść portów.

3.2. Opis strukturalny połączeń testowanego systemu z modułami analizatora sygnatur

System CRISTALDIGRAF NC posiada, obok programów testujących umieszczonych w pamięci ROM, pewne środki sprzętowe - w postaci mikroprzełączników i bramek logicznych - wspomagające testowanie. Uwzględnione one zostały w poniższym opisie, w którym wykorzystano instrukcję wyboru struktury języka MODLAN umożliwiającą modyfikację układu (odpowiadającą czynnościom wykonywanym w trakcie testowania) bez konieczności ponownej translacji całego opisu. Modyfikacji tej dokonuje się za pomocą instrukcji VARSTR języka sterowania symulacją MODSIM [1, 2].

```
BEGIN
SMODULE CRDGF (FI1, FI2, FITTL, % WEJSCIA ZEGAROWE SYSTEMU
               A [16] *TSL,
               DBIN, NIOW, PO1 [1-4], PO2 [1-5], PO3 [1-8],
               DEC, DEC1 [1-6], DEC2 [1-7], ADPERLOW
               : DBUS [8] * TSL)

  % SZCZEGOŁOWY OPIS STRUKTURALNY SYSTEMU MIT8O

END CRDGF;

FMODULE CONTROL (START, STOP, CLOCK, STEDGE, SPEDGE, CLEDGE,
                 QVAL, QLEV, QVALM : CL)

  % OPIS FUNKCJONALNY MODUŁU CONTROL

END CONTROL ;

FMODULE DATACOMP (DATA, HZCOD, FBCK, CL)

  % OPIS FUNKCJONALNY MODUŁU DATACOMP

END DATACOMP ;

NAND2 TESTDCL ; INV TESTFR, TESTD ;  % DEKLARACJE BRAMEK GENERUJACYCH
                                     % SYGNAŁY CLOCK DLA TESTOW 4, 5, 6

CONST TEST = 5 ;

CRDGF SYSTEM ; DATACOMP SAD [24] ;   % DEKLARACJE MODUŁOW

CONTROL SAC ;                        % ZDEFINIOWANYCH POWYŻEJ

GENERATOR ZEGAR [3] ;

SAC.START, SAC.STOP := 2(SYSTEM.A [15] );

SAD [1-24]. CL := 24 (SAC.CL);

TESTFR := SYSTEM.DBIN ; TESTDCL := TESTFR, SYSTEM.NIOW ;

TESTD := TESTDCL ;

SYSTEM.FI1, SYSTEM.FI2, SYSTEM.FITTL := ZEGAR;
```

```
SAD [1-24] .HZCOD := 24(1); SAD [1-24] .FBCK := 24(1);
ON TEST CASE  1,2,3,5,6   DO SAC.CLEDGE := 0
        CASE  1,3,4,5,6,  DO SAC.STEDGE := 0
        CASE      5       DO (SAC.CLOCK := TESTFR; SAC.SPEDGE := 0;
                              SAD [1-24] .DATA :=SYSTEM.A [1-16] ,
                                                 SYSTEM.DBUS [0-7] ).
        CASE      6       DO (SAC.CLOCK := TESTFR; SAC.SPEDGE := 1;
                              SAD [1-8] .DATA := SYSTEM.DBUS [0-7] )
        CASE      4       DO (SAC.CLOCK := TESTD;SAC.CLEDGE := 1;
                              SAC.SPEDGE :=; SAD [1-14] .DATA :=
                              SYSTEM.DEC, SYSTEM.DEC1 [1-6] , SYSTEM.DEC2
                              [1-7] )
        CASE      3       DO(SAC.CLOCK :=SYSTEM.NIOV; SAC.SPEDGE :=1;
                             SAD [1-17] .DATA := PO1 [1-4] ,PO2 [1-5] , PO3 [1-8]
        CASE      2       DO(SAC.CLOCK := SYSTEM.DEC [5] ; SAC.SPEDGE :=1;
                             SAC.STEDGE :=1; SAD [1-8] .DATA:=SYSTEM.DBUS)
        CASE      1       DO(SAC.CLOCK := SYSTEM. ADPERLOW;SAC.SPEDGE:=1;
                             SAD [1-8] := SYSTEM.DBUS)
        END
END
```

Łatwo zauważyć, że instrukcja wyboru struktury ściśle odpowiada tabeli 1, jeśli chodzi o dobór źródeł sygnałów sterujących. Dane zbierane są z różnych punktów układu w zależności od testu. Wartości parametru TEST umieszczone na tzw. listach etykiet wyboru (między słowami kluczowymi CASE i DO) odpowiadają numerom testów w tabeli 1.

Aby symulacja systemu poddanego testowaniu metodą analizy sygnatur przebiegała sprawnie, konieczne jest (obok umożliwienia modyfikacji sposobu włączenia modelu analizatora sygnatur w modelowany system) zapewnienie łatwych zmian struktury tego systemu, odpowiadających wyłączeniom niektórych podzespołów na czas trwania określonych testów (realizowanych w fizycznym modelu za pomocą mikrowyłączników). Na przykład w przypadku nieprawidłowej sygnatury na liniach danych w czasie testu free run należy sprawdzić po kolei wszystkie bloki pamięci ROM odłączając na ten czas pozostałe. W języku MODLAN można to opisać następująco:

```
ON ROMCHOICE  CASE 0 DO ROM  [1-5] . NCS : = DEC, DEC1 [1-4]
              CASE 1 DO ROM  [1] .   NCS : = DEC
              CASE 2 DO ROM  [2] .   NCS : = DEC1 [1]
              CASE 3 DO ROM  [3] .   NCS : = DEC1 [2]
```

```
CASE 4 DO ROM [4]. NCS : = DEC1 [3]
CASE 5 DO ROM [5]. NCS : = DEC1 [4]
END
```

Identyfikator DEC odpowiada w strukturze systemu CRISTALDIGRAF NC bramce logicznej dekodującej sygnały adresowe dla jednego z modułów ROM-u. Pozostałe moduły ROM-u wybierane są sygnałem z dekodera scalonego UCY 74155 oznaczonego DEC 1.
Wymuszenie instrukcji MOV A, A na wejściu procesora dla testu free run można zapisać za pomocą instrukcji połączenia warunkowego:

```
IF TESTER THEN CPU D/O-7/:= 7FH ELSE CPU.D/O-7/:= CPUBUS /O-7/.
```

CPUBUS jest tu identyfikatorem wewnętrznej magistrali danych. Parametr TESTER jest typu Boolean i może być modyfikowany zleceniem VARSTR języka MODSIM.

Sygnatury uzyskane w procesie symulacji zostaną zmierzone za pomocą analizatora sygnatur w prawidłowo działającym urządzeniu. Interesujące wydaje się wszakże zagadnienie, czy przez symulację można też uzyskać sygnatury nieprawidłowe, które umożliwiłyby lokalizację niektórych uszkodzeń.

4. Możliwości zbierania sygnatur identyfikujących typowe uszkodzenia

Symulacja uszkodzeń poszczególnych elementów systemu mikroprocesorowego jest niecelowa ze względów praktycznych - wszak każdy rodzaj uszkodzenia np. pamięci ROM dawałby inną sygnaturę nieprawidłową.

Ponieważ uszkodzenie któregokolwiek z elementów systemu mikroprocesorowego objawia się nieprawidłowymi sygnaturami w kilku miejscach układu można, analizując te sygnatury, zidentyfikować uszkodzony element. Częstym jednak powodem niesprawności systemu mikroprocesorowego są błędy montażowe i uszkodzenia połączeń. Mogą to być zwarcia ścieżek do masy lub napięcia zasilania, tzw. zimne luty, zwarcia ścieżek sąsiednich bądź przerwane ścieżki lub przewody.

Tego typu sytuacje łatwo zasymulować i znaleźć sygnatury, który będą na nie jednoznacznie (lub prawie jednoznacznie) wskazywały. Zwarcia do ścieżek zasilania bądź przerwane połączenia (np. zimne luty) powodują pojawienie się na wejściu analizatora sygnatur ciągu stanów 0, 1 lub stanu wysokiej impedancji.

W takim przypadku sygnatura zależy jedynie od wyboru źródeł sygnałów START, STOP, CLOCK. Bardziej skomplikowanym zagadnieniem jest modelowanie zwarć między sąsiednimi ścieżkami. Pojawienie się wówczas na ścieżkach przeciwnych wartości logicznych powoduje sytuację konfliktową. Napięcie, jakie ustali się wówczas na zwartych ścieżkach, może odpowiadać "0" lo-

gicznemu z uwagi na niższą rezystancję wyjściową większości układów w stanie niskim niż wysokim, może jednak odpowiadać także stanowi wysokiej impedancji.

Jakkolwiek w MODLAN-ie sygnał na magistrali może przyjmować wartość C oznaczającą, iż na magistralę kierowane są dane z więcej niż jednego nadajnika, jednak na wejściu modułu sygnał ten interpretowany jest jako stan nieokreślony, co uniemożliwia bezpośrednią interpretację zwarcia ścieżek w procesie symulacji. Interpretację taką możemy jednak uzyskać wykorzystując niżej opisany moduł SHORTCIRCUIT. Jego dwa wejścia, DATA IN i DATA SIDE łączymy w opisie strukturalnym ze ścieżkami, których zwarcie chcemy zamodelować, wyjście DATA zaś, łączymy z wejściem DATA modułu DATACOMP modelu analizatora sygnatur.

```
FMODULE SHORTCIRCUIT (DATAIN, DATASIDE: DATA)
  BEGIN
   VARIABLE : INTERP :
   SBEGIN
     ON DATAIN CASE 0 DO ON DATASIDE CASE 0 DO DATA ←"0
                                     CASE 1 DO ON INTERP
                                            CASE 1 DO DATA←0
                                            CASE 2 DO DATA←V
                                            END
                                     CASE V DO DATA←0
                                     END
               CASE 1 DO ON DATASIDE CASE 0 DO ON INTERP
                                            CASE 1 DO DATA←0
                                            CASE 2 DO DATA←V
                                            END
                                     CASE 1 DO DATA←1
                                     CASE V DO DATA←1
                                     END
               CASE V DO ON DETASIDE CASE 0 DO DATA←0
                                     CASE 1 DO DATA←1
                                     CASE V DO DATA←V
                                     END

               END

  SEND
END SHORTCIRCUIT;
```

Wartość zmiennej INTERP ustalamy instrukcją MINITIAL języka sterowania symulacją MODSIM [1, 2] w zależności od sposobu, w jaki chcemy modelować nadanie na zwarte ścieżki różnych stanów logicznych.

5. Podsumowanie

Omówione krótko w punkcie pierwszym tendencje rozwojowe techniki cyfrowej zwiększają oddziaływanie na siebie zagadnień związanych z projektowaniem i testowaniem układów.

Przedstawiona powyżej metoda pozwalająca uzyskać prawidłowe sygnatury (a także niektóre identyfikujące typowe niesprawności) już na etapie projektowania może znacznie ułatwić uruchamianie modelu urządzenia.

Symulacja uszkodzeń i oparta na niej automatyczna generacja wzorów testowych ma szczególne znaczenie w przypadku projektowania układów scalonych bardzo wielkiej skali integracji (VLSI).

Ich wzrastająca złożoność czyni zagadnienia testowania tak skomplikowanymi, że od kilku lat wzrasta nacisk na uwzględnianie wymogów testowania już na etapie projektu. Zalety analizy sygnatur i prostota związanego z tą techniką sprzętu każą przypuszczać, że wśród działań mających na celu ułatwienie testowania chipów VLSI będzie stosowane coraz szerzej umieszczanie w ich strukturze obwodów samotestujących, wykorzystujących analizę sygnatur [13]. W takim przypadku zgromadzenie słownika sygnatur w procesie symulacji umożliwiłoby umieszczenie prawidłowych sygnatur w pamięci ROM jako wzorca, z którym układ samotestujący porównywałby sygnatury zmierzone w trakcie testowania.

Jednocześnie zależy oczekiwać, że symulacja uszkodzeń na podstawie opisu funkcjonalnego będzie stosowana coraz częściej [11].

Autor zdaje sobie sprawę, że propozycja jego może pokazać swą wartość jedynie po jej weryfikacji przez symulację i pomiary. Ze względów technicznych (konieczność dopasowania systemu modelowania do pracy pod systemem operacyjnym GEORGE 3) publikuje ją przed tą weryfikacją.

Autor dziękuje dr inż. Andrzejowi Chławiczce i dr inż. Adamowi Pawlakowi z Instytutu Elektroniki Pol. Śl. za zachętę i cenne uwagi, które umożliwiły powstanie niniejszego artykułu.

LITERATURA

[1] Pawlak A.: Metoda opisu i system modelowania układów cyfrowych z mikroprocesorem. Praca doktorska Pol. Śl. Gliwice 1983.

[2] Pawlak A., Jeżewski J.: MODSIM - A Language for Control of Digital Logic Simulation. Proc. of 3rd Microelectronics Conference, Siofok 1982.

[3] Pawlak A.: MODLAN - A Hardware Module Description Language. Raport IPI PAN nr 532, Warszawa 1983.

[4] Hławiczka A.: Wykłady z przedmiotu Diagnostyka systemów mikroprocesorowych. Gliwice Pol. Śl.

[5] Hławiczka A., Nowiński M.: Programowalny analizator sygnatury PAS-80 współpracujący z mikrokomputerem. Materiały konferencji "Mikrokomputery w automatyce i technice pomiarowej". Wrocław 18-21.09.1984.

[6] Zagajewski T., Hławiczka A., Nowiński M., Jeżewski J.: sprawozdanie z pracy NB "System analizy topnienia" cz. pt. "Opracowanie koncepcji testowania systemu".

[7] Hławiczka A., Jura J. Sąkol J.: Testowanie systemów mikroprocesorowych za pomocą emulatora układowego sprzężonego z analizatorem sygnatur. Pomiary Automatyka Kontrola nr 2/1985.

[8] Hławiczka A., Huetter, Pach R.: Stymulator do analizy sygnaturowej uszkodzeń w systemach z mikroprocesorem 8080. PAK nr 6/1983.

[9] Laube J., Kałkus W.: Testowanie urządzeń mikroprocesorowych metodą analizy sygnatur. PAK nr 8/1983.

[10] Kubiś A.: Analiza sygnatur. WKŁ, "Elektronizacja" zeszyt 20.

[11] Bassak C.: Testing VLSI circuits. Job starts in design". Electronic Design nr 24/1984.

[12] Smith K.: Plessey custom chips will test themselves. Electronics Week 25 marzec 1985.

[13] Mead C., Conway L.: Introduction to VLSI systems. Addison Wesley Publisjing Company 1980.

Recenzent: Doc. dr inż. Ferdynand WAGNER

Wpłynęło do Redakcji 10.06.1985 r.

ИСПОЛЬЗОВАНИЕ ЯЗЫКА ОПИСАНИЯ ХАРДВЕРА МОДЛАН ДЛЯ ДИАГНОСТИКИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Р е з ю м е

В статье предложено применение языка для описания хардвера с целью определения сигнатур цифровой системы. Дано функциональное описание анализатора сигнатур НАС - 80 на МОДЛАН. На примере специализированной микрокомпьютерной системы КРИСТАЛЬ.ИГРАФ показан способ накопления сигнатур микрокомпьютерной системы во время её ситуляции. Рассмотрены возможности нахождения идентифицирующих некоторые монтажные ошибки.

THE USE OF THE MODLAN HARDWARE DESCRIPTION LANGUAGE IN A MICROCOMPUTER SYSTEMS DIAGNOSTICS

S u m m a r y

The use of a hardware description language in determining digital system signatures set has been proposed in this article.

The author has pointed out testing problems caused by the complexity of digital systems designed nowadays and the growing need of taking these problems into consideration early in the design cycle.

After brief description of the signature analysis concept a functional specification of the signature analyzer PAS-80 in MODLAN language has been presented.

With the dedicated microcomputer system CRISTALDIGRAF NC as an example the way of microprocessor system signatures gathering during its simulation (by means of the mentioned above functional model of PAS-80) has been showed.

Finally the method of modelling of some assembly faults has been presented and possibilities of finding signatures that identify such faults have been considered.