ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1992

Seria: INFORMATYKA z. 20 Nr kol. 1190

Lech ZNAMIROWSKI

KOMPUTEROWE WSPOMAGANIE KREŚLENIA MASEK STRUKTUR ASIC

Streszczenie. W pracy przedstawiono narzędzia programowe otwartego systemu oprogramowania do komputerowego wspomagania kreślenia masek cyfrowych struktur ASIC/VLSI, pozwalające generować opis struktury chipu na podstawie biblioteki standardowych, sprawdzonych komórek struktur elementarnych VLSI.

Przyjęto standardowy format opisu struktury CIF (Caltech Intermediate Form).

Wykorzystując cechy formatu CIF przedstawiono zasady i wymagania dla edytora graficznego korzystającego pośrednio z opisu struktur w wymienionym formacie, dokonując wyboru edytora niespecjalizowanego. Przyjęto zasadę wykorzystania ogólnie dostępnych edytorów współpracujących z translatorami formatów graficznych w środowisku systemu operacyjnego MS-DOS.

THE CAD SYSTEM TOOLS FOR ASIC STRUCTURES LAYOUT DRAWING

Summary. The CAD system tools for ASIC digital structures layout generation using well checked standard cells library are presented. The system generates standard format chip description. The format applied to structure description is CIF (Caltech Intermediate Form).

The modular structure of software system is covered with wide used text editors and standard graphics editors (DXF - Drawing eXchange file Format) cooperating with format translators in the MS-DOS environment.

CAO POUR DESSINER DES MASQUES DES STRUCTURES ASIC

Résumé. Dans ce travail, il a été presenté un groupe d'outils du système logiciel ouvert constituant le CAO, pour la génération et le dessin des masques des

structures d'integration à grande échelle/ASIC avec application de la librairie des cellules bien verifiées.

Le CIF (Caltech Intermediate Form) est langage intermédiare appliqué pour la description des structures VLSI/ASIC.

La conception du système modulaire de la programmation, donne une possibilité de l'usage des éditeurs de textes et des éditeurs graphiques standard (langage intermédiare DXF - Drawing eXchange file Format) coopérant avec le traducteur de la description des structures, dans l'environnement du système d'exploitation MS-DOS.

1. Wprowadzenie

Projektowanie struktur ASIC (Application Specific Integrated Circuits) przy aktualnych możliwościach technologii wymaga wyboru właściwego narzędzia wspomagania projektowania. Na to narzędzie składa się system komputerowy wraz z dedykowanym oprogramowaniem pracującym w pewnym środowisku programowym.

Systemy wspomagania projektowania, jeśli przyjąć kryterium porównania polegające na ocenie elastyczności, można podzielić na dwie grupy: systemy otwarte i systemy "turn-key" - zamknięte. W pierwszym przypadku użytkownik ma możliwość ingerowania w strukturę oprogramowania, organizowania "systemu operacyjnego" oprogramowania, może wymieniać i modyfikować moduły programowe systemu na niskim poziomie oprogramowania. W systemach "pod klucz", oprogramowanie jest zwykle wydajniejsze, a cena, jaką płaci użytkownik, jest ograniczeniem jego możliwości ingerencji w system, co jest niekorzystne w przypadku prac nad wykorzystywaniem nowych modeli, nowych interfejsów technologii, testowania czy włączania w system nowych modułów programowych (wymaga to ciągłych kontaktów użytkownik - producent systemu).

Przedstawione w pracy narzędzia programowe zakładają wykorzystanie procesorów rodziny Intel 80XXX, pracę w środowisku powszechnie stosowanego systemu operacyjnego MS-DOS, powszechnie stosowanych edytorów, bazy danych oraz ilustrują sposób wykorzystania specjalizowanych modułów do realizacji zadań "nowych" dla środowiska, co składa się na otwarty system CAD problemowo zorientowany na projektowanie struktur VLSI typu ASIC. Należy podkreślić, że nie przedstawiono tu narzędzi programowych do symulacji projektowanych układów i weryfikacji projektów masek, chociaż są to istotne elementy systemu wspomagania projektowania, stanowią one bowiem same w sobie obszerne zagadnienia wymagające odrębnego rozważenia.

2. Projektowanie układów scalonych ASIC

2.1. System wspomagania projektowania

Ogólne podejście związane z projektowaniem [NEWT1, MEAD1, FILI1, KUZM1, RUBI1, GOTO1, SHER1, KOZM1] układów scalonych, a w tym i struktur, od początku (1960 rok) wykazywało tendencję rozdzielenia projektu od technologii, w jakiej jest realizowany, co doprowadziło do opracowania i zaakceptowania interfejsu stanowiącego translator pomiędzy projektem a jego realizacją. Stanowią go reguły określające postać (formaty CIF, GDSII, LUCIE) i warunki, jakie spełnia projekt (reguły projektowania konstrukcyjne i elektryczne) i jakie akceptuje producent (często jest to grupa producentów działająca w ramach usługowych organizacji, zdolnych do akceptacji takich standardów - MOSIS, EUROCHIP).

Zwykle więc, kończące projekt "Formaty transferowe", stanowią dane wejściowe do fabrykacji układu scalonego.

Na metodykę projektowania struktur VLSI składają się cztery [GOTO1] fazy:

a) ocena wykonalności,

b) projektowanie,

c) weryfikacja,

d) wykonanie.

W rzeczywistości poszczególne fazy są znacznie rozbudowane, szczególnie w zakresie wspomagania projektowania na etapie projektu symulacją i weryfikacją na różnych poziomach opisu oraz generacją danych dla testów, natomiast w zakresie wykonania, badaniami wyrobu z wykorzystaniem testów pośrednich i końcowych [JAKU1, KOLO1, ZIEL1, FEUG1].

Blokowy schemat systemu automatyzacji (przyjęty jako ramowy dla niniejszej pracy) projektowania, a ściśle systemu CAD wspomagania projektowania struktur ASIC (zawierający także elementy kompilatora krzemowego) [WONG1, GAJS1, ZNAM2, ZNAM3] przedstawiony na rys. 1, ilustruje wzajemne powiązania zadań projektowych na tle specjalizowanych narzędzi CAD [NEWT1, GAJS1, KUZM3].

System korzysta z dużej biblioteki struktur, współpracując ciągle z bazą danych i edytorami, realizując zadania związane z projektowaniem struktury schematu, weryfikacji logicznej i układowej, projektowaniem struktury fizycznej, weryfikacji struktury.

Wynikiem projektu jest wygenerowana dokumentacja projektu, wydruki symulacji oraz projekt w postaci plików wykorzystujących formaty transferowe (interfejs projektu).

Baza danych wykorzystywana w procesie projektowania dostarcza danych potrzebnych dla różnych poziomów opisu abstrakcyjnego struktur [DUTT1, DUVA1].

2.2. Fabrykacja (Silicon Foundry)

Proces fabrykacji obejmuje [WONG1, SZE1, PROC1, DUTT2]] sekwencje operacji związane z wytwarzaniem masek do naświetlania rezystu (warstwy kopiowej) albo litografii wiązką elektronów (lub innych bezpośrednich metod litografii), procesy obróbki płytki, testy ostrzowe, montaż struktury (realizacja systemu połączeń chipu z obudową - packaging) i testy końcowe. Zwykle z danym typem technologii związane są wymagania technologiczne, które na poziomie projektu uwzględnia się przez realizację reguł projektowych konstrukcyjnych i elektrycznych [MEAD1, MALY1, NEWK1, STRO1].

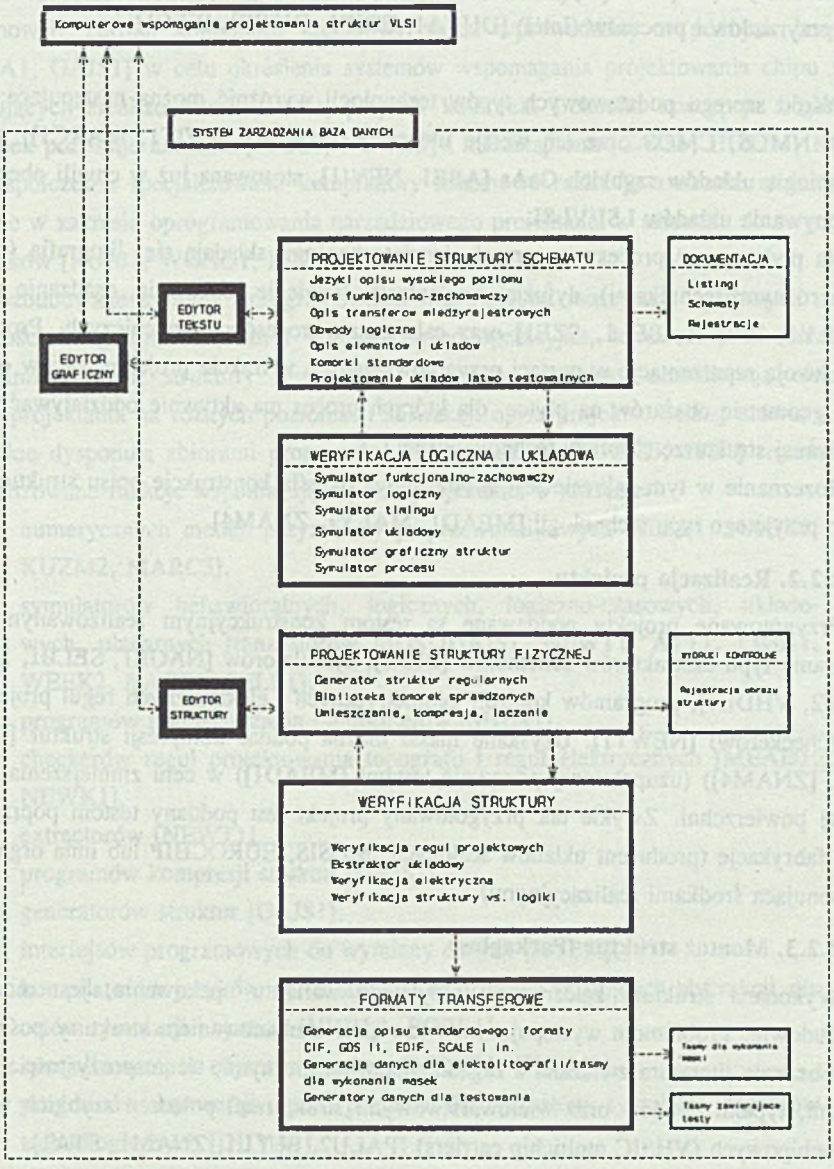
2.2.1. Procesy technologiczne

Procesy technologiczne w fabrykacji struktur składają się z ciągu operacji, uporządkowanych w zależności od typu procesu w sekwencję prowadzącą do uzyskania "technologicznych obrazów" masek, ciąg zaś procesów ma prowadzić do wytworzenia działającej struktury [MALY1, MEAD1, ZNAM4].

W trakcie fabrykacji realizowane muszą być wymagania związane z powtarzalnością i dokładnością geometrii, właściwości fizykochemicznych materiałów, stabilności parametrów procesów.

Skala rodzaju urządzeń realizujących aktualnie procesy jest bardzo szeroka, obejmuje stosowanie specjalizowanych pieców, reaktorów plazmowych, urządzeń litografii optycznej, litografii wiązką elektronową i litografii rentgenowskiej, źródeł promieniowania ultrafioletowego i rentgenowskiego, systemów próżniowych, laserów i złożonych systemów optycznych.

Typy technologii można przedstawić na podstawie zbioru powtarzalnych dla danej technologii procesów. W tym zakresie rysuje się silna tendencja budowy modeli procesów [ANTO1, DUTT1, DUTT2, STRO1, ITE1, BLOD1], opracowania programów kompilujących modele procesów w model technologii i na tej podstawie prowadzenia złożonych badań symulacyjnych nad uzyskiem, kompresją komórek struktur elementarnych [BARK1, FERR1] weryfikowaną symulacją oraz modeli ciągów procesów "realizujących" zaprojektowane struktury [STRO1]. Opracowano także propozycje interfejsów pozwalających przedstawiać, podobnie jak dla warstw geometrię (maski), tak w przypadku procesów,



Rys. 1. Typowy system automatyzacji procesu projektowania struktur ASIC (elementy kompilatora krzemowego)

Fig. 1. Typical design automation system for ASIC (elements of a Silicon Compiler)

hierarchiczny opis profili wytwarzanych struktur półprzewodnikowych dla celów symulacji przyrządów i procesów (Intel) [DUVA1, EDIF1, DUTT2, IEEE1].

Wśród szeregu podstawowych typów technologii wyróżnić można następujące: bipolarną, NMOS, CMOS oraz ich wersje ulepszone [MALY1, SWIT1, MARC1], a także technologię układów szybkich GaAs [ABE1, NEVI1], stosowaną już w chwili obecnej do wykonywania układów LSI/VLSI.

Na podstawowe procesy w ramach danej technologii składają się: litografia (realizowana różnymi technikami), dyfuzja, implantacja, trawienie, utlenianie, osadzanie warstw [MALY1, PROC1, BEL1, SZE1] oraz cała grupa procesów pomocniczych. Procesy te mają swoją reprezentację w postaci przygotowywanych w trakcie projektu warstw opisujących geometrię obszarów na płytce, dla których proces ma aktywnie oddziaływać w fabrykowanej strukturze ("obrazy technologiczne").

Rozeznanie w tym zakresie jest istotne, gdyż określa konstrukcję opisu struktur w ramach przyjętego typu technologii [MEAD1, MALY1, ZNAM4].

2.2.2. Realizacja projektu

Przygotowane projekty poddawane są testom konstrukcyjnym realizowanym przez programy typu ekstraktorów schematów [BEL1], symulatorów [NAGE1, SELB1, LIGH1, ROSI2, VHDL1], programów kontroli geometrycznych i elektrycznych reguł projektowania (checkerów) [NEWT1]. Uzyskane maski można poddać kompresji struktur [GAJS1] (por. [ZNAM4]) (uzupełnionej ponownie testami [MEAD1]) w celu zmniejszenia zajmowanej powierzchni. Zwykle tak przygotowany projekt jest poddany testom poprzedzającym fabrykację (producent układów scalonych, MOSIS, EUROCHIP lub inna organizacja dysponująca środkami realizacyjnymi).

2.2.3. Montaż struktur (Packaging)

Wykonana struktura, zależnie od przyjętego wariantu upakowania, jest mocowana w obudowie. Problemom występującym w związku z obudowaniem struktury poświęcona jest obszerna literatura związana z zagadnieniami termicznymi, termosprężystością, wieloliniami, typami obudów oraz wielowarstwowymi strukturami połączeń szybkich układów wielochipowych (VHSIC multichip carriers) [PALU2, BULL1, ZNAM1, STA1].

2.3. Środowisko programowe kompilatora krzemowego

Wśród szeregu stosowanych strategii projektowania układów VLSI, jeśli jako kryterium porównania przyjmiemy stopień automatyzacji projektowania, strategia zmierzająca

do możliwie zautomatyzowanych operacji związanych z generacją topografii struktury na podstawie opisu struktury na wysokim poziomie abstrakcji opisu nosi nazwę kompilacji krzemowej. Termin kompilator krzemowy został wprowadzony w późnych latach 70 [JOHA1, GAJS1] w celu określenia systemów wspomagania projektowania chipu wykorzystujących zrealizowane wcześniej projekty komórek. I chociaż koncepcja sięgania do bibliotek pozostaje aktualna [IEEE1, NEWK1] i niezastąpiona [GAJS1, HOLD1, KOZM-1], współczesne specjalizowane kompilatory krzemowe realizują znacznie rozbudowane funkcje w zakresie oprogramowania narzędziowego projektanta w stosunku do swych poprzedników [RUBI1, WONG1, EINS1, FEUE1].

Rozbudowane systemy oprogramowania pracujące na maszynach cyfrowych o dużej szybkości i wydajności (duża i szybka pamięć operacyjna, hardware'owe zarządzanie zasobami pamięci, struktury równolegle, procesory graficzne) umożliwiają efektywną pracę projektanta na różnych poziomach abstrakcji opisu projektu. Ściślej mówiąc, systemy takie dysponują zbiorami programów [IEEE1, BEL1, KUZM3, ITE1] realizującymi specjalizowane funkcje wspomagające pracę projektanta w zakresie:

- numerycznych modeli przyrządów półprzewodnikowych [SELB1, NAGE1, KUZM2, MARC3],
- symulatorów behawioralnych, logicznych, logiczno-czasowych, układowych, planarnych tranzystorów MOS [GAJS1, NEWT1, ATT1, LIGH1, WEEK2, NAGE1, SELB1],
- programów rozmieszczania i autoruterów [SHER1],
- checkerów reguł projektowania topografii i reguł elektrycznych [MEAD1, NEWK1],
- extractorów [NEWT1],
- programów kompresji struktur [GAJS1],
- generatorów struktur [GAJS1],
- interfejsów programowych do wymiany danych [RUBI1],
- kompilatorów języków opisu struktur na różnych poziomach abstrakcji dla celów symulacji i syntezy [VHDL1, BORR1],
- specjalizowanych edytorów struktur [MEAD1, KUZM3],
- programów konwersji opisu projektu do formatów symulacji procesów technologicznych [DUVA1],
- programów obsługi bibliotek zapisanych w bazie danych dla celów projektowania [DUTT1, DUVA1].

Aby efektywnie wykorzystać oprogramowanie, które zresztą zwykle jest tworzone w różnych środowiskach i w różnym czasie, a dla celów integracji wymaga możliwości

wymiany informacji najczęściej dwustronnej, zachodzi potrzeba dysponowania oprogramowaniem uzupełniającym, spełniającym rolę interfejsów programowych dla wymiany danych.

Do grupy takich programów można zaliczyć przedstawione dalej w pracy translatory formatów [PAWL1, ZNAM2] opisu struktur ASIC w formatach CIF oraz DXF [MEAD1, AUTO2].

3. Opis struktur VLSI (ASIC)

Przyjęty tu opis struktur VLSI (ASIC) jest opisem jawnym, tzn. przyjmuje się, że struktury wyrażane są obrazami je przedstawiającymi. Jest to sposób nie dający oszczędności pamięci zajmowanej przez projekt (chociaż w porównaniu z innymi formatami, format CIF jest oszczędny) ma natomiast zaletę dużej poglądowości i stwarza projektantowi możliwość ingerencji w projekt w każdym elemencie struktury. Ponadto konieczne środki programowe są proste: sprowadzają się bowiem do edytorów tekstowego i graficznego, translatorów formatów, a także dla niewielkich projektów, prostej bazy danych.

3.1. Formaty transferowe opisu struktur

Przyjęte w praktyce standardowe formaty stosowane w dziedzinie projektowania struktur VLSI (ASIC) można zaliczyć do grup: formatów transferowych (Maski/Obrazy struktur) - CIF [MEAD1] (The Caltech Intermediate Form), GDS II [RUBI1] (Calma & General Electric), LUCIE [PAIL1]] (Langage Universitaire de Conception de circuits Intégré pour l'Enseignement), DXF [AUTO2] (Drawing eXchange file Format) oraz formatów opisu schematów elektrycznych (pozwalających również opisywać elementy topografii struktur) EDIF [EDIF1] (The Electronic Design Interchange Format), SPICE [NAGE1] (Simulation Program with Integrated Circuits Emphasis - Berkeley).

Przedstawione formaty nie wyczerpują innych sposobów reprezentowania obrazów wykorzystywanych do opisu struktur [ASIC1], przyjęte są jednak powszechnie.

Zaklasyfikowanie danego formatu do odpowiedniej grupy jest w zasadzie umowne, gdyż zakresy stosowania wzajemnie przenikają się.

Szczególnie rozpowszechniony jest w zakresie projektowania struktur VLSI format CIF, który dalej zostanie przedstawiony szczegółowo.

Należy zwrócić uwagę na to, że właściwy wybór formatów transferowych projektów jest bardzo istotny ze względu na ich techniczną realizację, którą w przypadku wykony-

wania krótkich serii czy pojedynczych układów dla celów badawczych lub dydaktycznych, mogą zapewnić międzynarodowe stowarzyszenia bazujące na korporacji producentów układów scalonych. W Stanach Zjednoczonych jest to MOSIS [SMIT1, RUBI1] MOS Implementation Service (formaty CIF oraz GDS II), a w Europie EUROCHIP [EURO1, DELO1] (CIF, LUCIE oraz GDS II).

3.2. Format CIF opisu struktur

Tekstowy format opisu struktur CIF charakteryzuje się tym, że zapisane struktury zajmują stosunkowo mało pamięci (porównując z innymi formatami). Mechanizm formatu jest tak pomyślany, że stanowi rodzaj programu generującego obraz, wykorzystując struktury programowe o nazwie transformacji, które działając na wcześniej definiowanych makrostrukturach, w sposób przejrzysty generują złożone obrazy. Przyjęty zbiór prymitywów graficznych, ukierunkowany na potrzeby projektów struktur, daje możliwość efektywnej edycji zarówno "ręcznej", jak i opartej na przetwarzaniu plików CIF.

Wymiary i odległości przedstawiane są jako liczby całkowite. Jednostką jest jedna setna mikrona (por. DODATEK 1; [α] = 10[nm]).

Podobnie jak w procesie technologicznym, integralną częścią opisu są warstwy, w ramach których opisuje się lokalizację prymitywów.

Dalej przedstawiono ścisły formalnie opis formatu CIF. Składnia dopuszcza blanki przed i po komendzie, blanki lub inny rodzaj separatora (wyjątki w definicji upperChar i blank). Składnia odzwierciedla fakt, że definicje symboli nie mogą być zagnieżdżone.

3.2.1. Syntaktyka formatu CIF

Plik CIF składa się z sekwencji znaków stanowiących listę komend zakończonych znacznikiem (komendą) końca. Komendy oddziela się średnikami.

Przy definiowaniu składni przyjęto standardową notację Wirtha:

=	zdefiniowano jako
|	lub (pionowa kreska)
' '	oznaczenie symbolu terminalnego (w obrębie " ")
{ }	nawias wskazujący powtórzenie dowolną ilość razy włączając zero razy
[ ]	nawias wskazujący, że zawartość może być powtórzona zero lub jednokrotnie
( )	nawias określający grupę
.	kropka kończąca definicję.

Poniżej, dla tak określonej konwencji oznaczeń, przedstawiono syntaktykę [MEAD1] formatu opisu CIF.

```
cifFile              = {{blank} [command] semi} endCommand {blank}.
command              = primCommand | defDeleteCommand | defStartCommand
                       semi {{blank} [primCommand] semi} defFinishCommand.
primCommand          = polygonCommand | boxCommand | roundFlashCommand
                       | wireCommand | layerCommand | callCommand |
                       userExtensionCommand | commentCommand.
polygonCommand       = "P" path.
boxCommand           = "B" integer sep integer sep point [sep point].
roundFlashCommand    = "R" integer sep point.
wireCommand          = "W" integer sep path.
layerCommand         = "L" {blank} shortname.
defStartCommand      = "D" {blank} "S" integer [sep integer sep integer].
defFinishCommand     = "D" {blank} "F".
defDeleteCommand     = "D" {blank} "D" integer.
callCommand          = "C" integer transformation.
userExtensionCommand = digit userText.
commentCommand       = "(" commentText ")".
endCommand           = "E".

transformation       = {{blank} ("T" point | "M" {blank} "X" | "M" {blank}
                       "Y" | "R" point)}.
path                 = point {sep point}.
point                = sInteger sep sInteger.

sInteger             = {sep} ["-"] integerD.
integer              = {sep} integerD.
integerD             = digit {digit}.
shortname            = c [c] [c] [c].
c                    = digit | upperChar.
userText             = {userChar}.
commentText          = {commentChar} | commentText "(" commentText ")" com-
                       mentText.

semi                 = {blank} ";" {blank}.
sep                  = upperChar | blank.
digit                = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
```

upperChar = "A" | "B" | "C" | ... | "Y" | "Z".
blank = dowolny znak ASCII z wyjątkiem digit, upperChar, "-", "(", ")", lub ";".
userChar = dowolny znak ASCII z wyjątkiem ";".
commentChar = dowolny znak ASCII z wyjątkiem "(" lub ")".

3.2.2. Semantyka formatu CIF

Założeniem konstrukcji formatu jest możliwość tworzenia plików, które nie stosują zbyt dużej liczby prymitywów, w związku z czym ograniczono się do kilku prostych struktur: wielokąta, koła, ścieżki i prostokąta kreślonych przez komendy "rysujące": P (polygonCommand), R (roundFlashCommand), W (wireCommand) i B (boxCommand). Ponieważ maski zawierają wiele takich prymitywów, konieczne było wprowadzenie makrostruktur i wywołań tych makrostruktur, co radykalnie nie tylko oszczędza pamięć, zmniejsza objętość plików, ale także czyni pliki czytelniejszymi.

Prymitywy rozmieszczane są na warstwach, które ustawiane są specyfikacjami L (layerCommand) poprzedzającymi listę prymitywów. Przyjęto konwencję, zgodnie z którą pierwsza litera nazwy warstwy oznacza technologię (np. NP - nazwa maski warstwy polikrzemu w procesie nMOS, NI - nazwa maski warstwy implantacji w procesie nMOS, CPOL - nazwa maski warstwy polikrzemu w procesie CMOS czy CME1 - nazwa maski warstwy metalu 1 w procesie CMOS). Makrostruktury definiowane są w obrębie symboli komend DS (defStartCommand) ... DF (defFinishCommand), posiadają swój numer (oznaczany dalej symbolem NumerStruktury), skalę (a/b), a z ciała definicji mogą być wywoływane inne makrostruktury. Wprowadzona komenda DD (defDeleteCommand) może być wykorzystana do deaktywowania od numeru będącego argumentem komendy i numerów większych wprowadzonych wcześniej definicji symboli, co w prosty sposób umożliwia realizację multiprojektów (komenda DD poniżej numeru argumentu umożliwia wykorzystywanie innych makrostruktur, natomiast pozwala "zapomnieć" wprowadzone wcześniej definicje określone numerami równymi i większymi od swego argumentu).

Komendy wywołania C (callCommand) i transformacji (transformation) - (T, MX, MY, R) umożliwiają w prosty sposób umieszczanie w realizowanym rysunku istniejących struktur po ich przetransformowaniu.

Praktyka generacji biblioteki podstawowej VLSI wskazuje na fundamentalne znaczenie prostego prymitywu Box w budowie, sprawdzaniu, stosowaniu w procesie technologicznym, przetwarzaniu, translacji struktur. Omówimy ten prymityw.

Zgodnie z syntaktyką komendy, mamy jej postać tekstową:

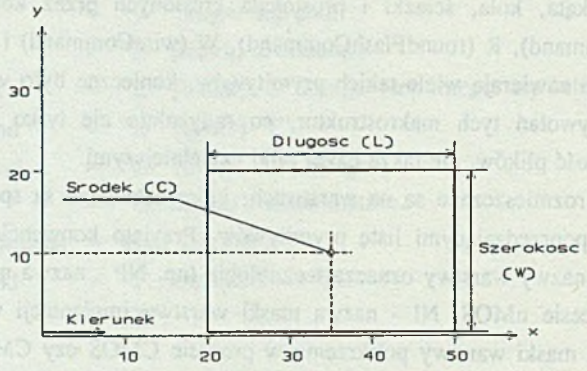
boxCommand = "B" integer sep integer sep point [sep point],

co można w sensie jej znaczenia (w pełnym rozwinięciu) zapisać w postaci:

B długość(L) szerokość(W) x_c(C) y_c(C) a_x a_y,

gdzie przez x_c oznaczono współrzędną x środka Boxu, y_c - współrzędną y środka Boxu, a_x - współrzędną rzutu wektora kierukowego Boxu na oś x oraz a_y -współrzędną rzutu wektora kierunkowego Boxu na oś y.

W przypadku braku w aktualnej komendzie Box parametrów a_x i a_y, przyjmuje się wektor kierunkowy o współrzędnych $a_x=1$, $a_y=0$.



Rys. 2. Struktura Box
Fig. 2. Box representation

Przykład

Strukturę Box, którą zapiszemy w postaci:

B 30 20 35,10;

w postaci graficznej przedstawiono na rys. 2.

Wywołania symboli i transformacje stosowane w formacie CIF oraz przeliczanie wartości liczbowej odległości opisywanych w tym formacie przedstawiono w DODATKU 1.

3.2.3. Struktura pliku CIF

Struktura pliku spełniać musi podstawowe wymaganie polegające na tym, aby definicje symboli poprzedzały ich wywołania.

Początek definicji symbolu nie może pojawić się w definicji innego symbolu. Format CIF nie dopuszcza zagnieżdżania definicji symboli. Po zdefiniowaniu makrostruktur, ich wywołaniach wraz z transformacjami, bezpośrednio zdefiniowanych prymitywach, plik CIF kończy się komendą E (endCommand).

4. Edytor graficzny rysunku struktur

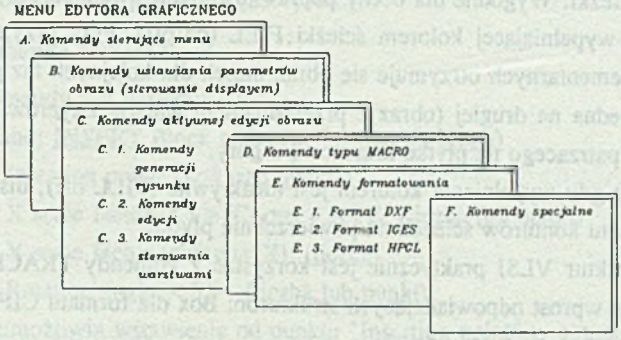
Konstrukcja warstwowa rysunków struktur, stosowane prymitywy obrazowe i konieczne operacje na nich pozwalają sprecyzować wymagania, jakie powinien spełniać edytor graficzny.

Przyjęto rozwiązanie polegające na wykorzystaniu uniwersalnego edytora graficznego, eksponując potrzebne komendy w wydzielonym "Menu".

Takim edytorem może być edytor uniwersalnego programu AutoCAD [AUTO2].

Komendy związane z edycją obrazów masek można zebrać w następujące grupy:

A. Komendy sterujące menu [Otwarcie plików w celu utworzenia nowego rysunku, Edyc-



Rys. 3. "Menu" dla edycji opisu struktur VLSI
Fig. 3. ."Menu" of VLSI structures description editor

ja istniejącego rysunku, Wyprowadzenie rysunku na plotter, ...], B. Komendy ustawiania parametrów obrazu [Powiększ/pomniejsz (ZOOM), Przeglądaj duże rysunki bez powiększania (PAN), Rysuj wypełnianie kolorem (FILL),...], C. 1. Komendy generacji rysunku [Nanieś ścieżkę (TRACE), Nanieś tekst (TEXT), Ustaw parametry skoku kursora (SNAP), ...], C. 2. Komendy edycji [Usuń element (ERASE), Przesuń (MOVE), ...], C. 3. Komendy sterowania warstwami [Ustaw warstwę (LAYER), Ustaw kolor (COLOR), ...], D. Komendy typu MACRO [Utwórz substrukturę (BLOCK), Wstaw substrukturę (INSERT), ...], E. Komendy formatowania [Format DXF (Wyprowadź/Czytaj rysunek w formacie DXF) .....], F. Komendy specjalne [Wprowadź własne menu (MENU), Zapisz projekt na dysk (SAVE), Kończ edycję bez zapisu na dysk (QUIT), ...].

W powyższym menu (rys. 3) zaznaczono najistotniejsze komendy dla założonych zadań edycji. W nawiasach dla porównania przytoczono nazwy komend uniwersalnego edytora graficznego programu AutoCAD [AUTO2].

Wybór jest podyktowany wieloma możliwościami realizacji menu z jednej strony oraz możliwością wykorzystania wymiennych plików DXF opisu rysunku do wymiany z innymi programami z drugiej. W dalszym ciągu przedstawiono komendy najczęściej wykorzystywane w edycji obrazów struktur.

4.1. Komenda kreślenia ścieżki TRACE

Komenda TRACE pozwala kreślić ścieżki o określonej parametrem komendy szerokości. Jako punkt początkowy określa się punkt o współrzędnych środka ścieżki, podobnie jest dla końca ścieżki. Wygodne dla oceny poprawności realizowanego projektu jest stosowanie komendy wypełniającej kolorem ścieżki FILL (on/off). Przy przyjętej konstrukcji opisu struktur elementarnych otrzymuje się obraz masek dla kolejnych faz procesu, masek umieszczonych jedna na drugiej (obraz z przesłanianiem warstw Layerów leżących niżej) dla obserwatora patrzącego na płytkę krzemową z góry.

W przypadku gdy wypełnianie kolorem jest nieaktywne (FILL off), display prezentuje barwny obraz rzutu konturów ścieżek na powierzchnię płytki.

W edycji struktur VLSI praktycznie jest korzystać z komendy TRACE w celu generacji prostokątów wprost odpowiadających strukturom Box dla formatu CIF.

Format komendy jest następujący:

Command: TRACE Trace width <current>: (podać szerokość ścieżki)
From point: (określić punkt)
To point: (określić punkt)
To point: (określić punkt)
To point: (<Enter> zakończenie ścieżki)

Podkreśleniem oznaczono dane dialogu wprowadzane przez operatora. Wiersze dialogu kończone są naciśnięciem klawisza <Enter>.

4.2. Komendy struktury BLOCK

4.2.1. Komenda BLOCK

Komenda BLOCK tworzy blok (strukturę mogącą zawierać wiele prymitywów) dla opracowywanego rysunku i może być na tym rysunku wielokrotnie wstawiana.

Format komendy:

Command: BLOCK Block name(or ?): (nazwa bloku)

Podając parametr komendy (nazwa bloku), można wprowadzić nazwę nowo definiowanego bloku lub znakiem <?> uzyskać wydruk listy bloków dla aktualnie realizowanego rysunku. W trakcie definiowania bloku określić należy punkt bazowy bloku, który przy wstawianiu bloku komendą INSERT jest zrównywany (pokrywa się) z punktem "Insertion point" komendy INSERT.

Przykład:

Command: BLOCK Block name (or ?): CELL

Insertion base point: 20,26

Select objects: (określić go)

W wyniku wykonania komendy utworzony zostaje blok o nazwie CELL, przy czym jego punkt bazowy ma współrzędne (20.0,26.0).

4.2.2. Komenda INSERT

Format komendy jest następujący:

Command: INSERT Block name (or ?): (nazwa bloku)

Insertion point: (wskazać punkt)

X scale factor <1>/Corner / XYZ: (liczba lub punkt)

Y scale factor (default=X): (liczba)

Rotation angle <0>: (liczba lub punkt)

Komenda umożliwia wstawienie od punktu "Insertion point" w aktualnie realizowanym rysunku, obrazu zawartego w uprzednio zdefiniowanym bloku o nazwie (nazwa bloku) względem punktu bazowego bloku, przy czym jest przyjęta zasada, że jeśli nie definiowano wyraźnie współrzędnych punktu bazy to przyjmuje się je jako (0.0,0.0).

Komenda INSERT może być także wykorzystana do wstawiania rysunku (nie deklarowanego jako struktura bloku) w rysunek. W tej sytuacji jako nazwę bloku wstawianego należy podać nazwę pliku zawierającego wstawiany rysunek.

Wielokrotne wstawianie realizować można wygodnymi komendami MINSERT lub ARRAY, natomiast komendą WBLOCK w trakcie edycji większej struktury możemy zapisywać fragmenty struktury jako osobne rysunki.

Uzupełniając listę najczęściej wykorzystywanych komend, należy wymienić komendy: obrotu ROTATE, odbicia MIRROR, MOVE, SAVE, QUIT, LIMITS, STATUS, ZOOM, GRID, LAYER, a także komendy konwersji formatu zapisu rysunku DXFIN i DXFOUT.

4.3. Format DXF opisu struktur

Obrazy przyjmowane przez edytor AutoCADa przechowywane są w formacie wewnętrznej bazy danych (pliki z rozszerzeniem .DWG).

Dla celów wymiany danych z otoczeniem przyjęty jest format DXF (Drawing eXchange file Format).

Format ten określa pliki tekstowe o czytelnej dla człowieka postaci, jest zorganizowany przejrzyście i pisanie translatorów formatów lub wykorzystywanie go przez inne specjalizowane programy wspomagania projektowania nie nastręcza trudności.

Oceniając sytuację z punktu widzenia użytkownika, plik DXF odzwierciedla elementy rysunku (prymitywy obrazowe) i bloki, czyli ujęte jako całości grupy elementów rysunku. Ma to swoją reprezentację w strukturze pliku DXF.

Plik DXF złożony jest z części wyszczególnionych niżej jako sekcje i koniec pliku:

- **sekcja NAGŁÓWKA (HEADER section).**

W sekcji zgromadzone są parametry określone wartościami zmiennych związanych z prezentowanym na displayu rysunkiem. Są to dane zbliżone do postaci informacji generowanych komendą STATUS.

- **sekcja TABLIC (TABLES section).**

W sekcji zgromadzone są deklaracje dotyczące struktury rysunku (nazwy warstw - Layer, rodzaje czcionki - Style, rodzaje linii - Ltype, tablice VIEW).

- **sekcja BLOKÓW (BLOCKS section).**

W sekcji zebrano definicje bloków (Block Definition) wykorzystywanych na rysunku. Mogą wystąpić odwołania do bloków z sekcji ENTITIES.

- **sekcja ELEMENTÓW (ENTITIES section).**

W sekcji występują prymitywy i ewentualne odwołania do bloków. Ten segment stanowi rodzaj interpretera generującego obraz na displayu edytora.

- **KONIEC PLIKU (END OF FILE).**

Format DXF realizowany jest z zestawienia mniejszych elementów zwanych *grupami* w strukturę opisaną wyżej. Grupa składa się z dwu linii. Pierwsza linia zawiera tzw. *kod grupy* - liczbę typu integer (np. dla FORTRANU odpowiada formatowi I3), natomiast druga linia grupy określa *typ wartości grupy* zależny od kodu grupy (np. kod 0-9 wyznacza typ wartości jako napis, kod grupy 10-59 wyznacza typ wartości jako liczbę zmiennoprzecinkową, ... [AUTO2]).

Format DXF szczegółowo definiuje związek kodów grupy i typów wartości w rozbiciu na te typy, które podnoszą cechy parametrów rysunku (np. Typ linii, nazwa warstwy, współrzędna x, y, z, czcionka, ... [AUTO2]).

Prymitywem formatu DXF odpowiadającym prymitywowi Box w formacie CIF jest TRACE. Sposób tworzenia TRACE w pliku DXF oraz strukturę pliku DXF przedstawiono w DODATKU 2.

5. Translacja formatów (przykład)

Translacje formatu DXF na format CIF i odwrotnie realizowane w trakcie edycji obrazów masek wnoszą pewne problemy [PAWL1, ZNAM2], które należy rozstrzygnąć przy opracowywaniu programów translacji.

Problemy te wynikają z faktu, że różne formaty nie zawsze mają odpowiadające sobie listy elementów oraz komend. To pociąga za sobą konieczność wprowadzenia wirtualnych elementów dopasowujących elementy i komendy w taki sposób, aby zwolnienie translacji było możliwie małe i zapewniało możliwie oszczędne gospodarowanie pamięcią. Podobne problemy występują przy operowaniu nazwami symboli. Typy liczb występujące w formatach wymagają wprowadzenia procedur konwersji liczb.

Uwzględniając fakt, że interpretacja pliku DXF w edytorze graficznym wnosi w konsekwencji sekwencyjną realizację obrazu, co w pewnych przypadkach (stan pracy zapewniający wypełnianie kolorem obrazów na warstwach) powoduje niebezpieczeństwo (mimo bezbłędnej struktury danych) generacji obrazów paradoksalnych (niewłaściwe przenikanie warstw), zachodzi konieczność sortowania elementów zapisanych na warstwach na potrzeby ekspozycji zawartości pliku DXF na displayu edytora.

Ostatnim z ważniejszych problemów jest przeliczanie odległości związanych ze skalą struktur dla różnych reprezentacji. Zagadnienie to przedstawiono w DODATKU 1.

Powyższe problemy szczegółowo rozpatrzono w pracy [PAWL1] w trakcie opracowywania translatorów formatów, pozostawiając użytkownikowi możliwość świadomego wyboru przypadków alternatywnych poprzez wprowadzenie parametrów translacji.

Przykład

Przedstawimy opis struktury TRACE z rys. 5 w formacie CIF, przyjmując $[\lambda]=2.5\text{-}[\mu]$ (przy $[\alpha]=0.01[\mu]$).

Korzystając z zależności (7.1) do (7.4) (DODATEK 1), otrzymujemy wprost:

$(dist_{CIF})_i = 250(b/a)(D_{DXF})_i$.

Przyjmując b=1, a=125, nadając tworzonemu symbolowi nazwę (NumerStruktury) "100", otrzymamy opis w formacie CIF w postaci następującej:

DS 100 125/1;L NP;B 4 12 26,58;.

6. Podsumowanie

Przedstawiony w pracy wybór narzędzi programowych oraz formatów opisu masek struktur ASIC składa się na graficzny system ich generacji i edycji.

Ze względu na to, iż niektóre biblioteki standardowych, sprawdzonych komórek struktur elementarnych VLSI zostały częściowo opublikowane [MEAD1, NEWK1], przy odpowiednim doborze oprogramowania podstawowego oraz opracowania modułów specjalizowanych (moduły te z praktycznego punktu widzenia ich funkcji mogą być bardzo proste [ZNAM2]), projektowanie struktur ASIC można sprowadzić do zagadnienia wykonalnego środkami niekoniecznie najwyżej specjalizowanymi, w rozsądnym czasie.

Oczywiście, w przypadku projektowania struktur FCD (Full Custom Design), system oprogramowania musi być wyposażony w narzędzia wyszczególnione w pkt. 2.3, tym niemniej projektowanie w "technologii" VLSI struktur ASIC (biblioteka standardowych komórek sprawdzonych) nie musi być domeną wysoce profesjonalnego środowiska. To, niestety, z drugiej strony automatycznie wymusza pewien brak projektantów układów ASIC [SMIT1, EURO1, MEAD1].

W chwili obecnej można powiedzieć, że style projektowania struktur ASIC narzuca zinformatyzowane otoczenie projektanta. Projektowanie układów scalonych w początkowym natomiast okresie (lata sześćdziesiąte [NEWT1] i później), wraz z rozwojem technologii, wymuszało rozwój podstawowych i specjalizowanych narzędzi programowych.

Z drugiej strony zarysowała się logiczna tendencja do pełnego rozgraniczenia projektu od technologii, poprzez precyzyjne zdefiniowanie interfejsu pomiędzy zakończonym projektem a jego technologiczną realizacją. Kompilacja krzemowa, a także stosowanie symulatorów procesów technologicznych są tego najlepszym przykładem, wyraża to również tendencja do tworzenia organizacji typu MOSIS czy EUROCHIP realizujących projekty.

Przedstawiony w pracy system generacji i edycji masek struktur ASIC wymaga wzbogacenia go o kolejne moduły programowe, wśród których należy na pierwszym miejscu wymienić program autorutera realizującego automatyczne lub półautomatyczne ciągnięcie połączeń pomiędzy strukturami umieszczanymi w projekcie oraz programy kontroli geo-

metrycznych reguł projektowania i ekstrakcji schematu elektrycznego. Algorytm działania autorutera, uwzględniając zachowanie geometrycznych reguł projektowania, realizuje poprawne połączenia, jednak w praktyce, w rozbudowanych projektach zwykle niezbędna jest dodatkowa, "ręczna" edycja maski, co podkreśla praktyczny aspekt konieczności stosowania drugiej grupy z wymienionych wyżej programów w celu weryfikacji zmodyfikowanej maski. Programy te usprawniają również pracę projektanta przy tworzeniu rozbudowanych masek nowych struktur nie występujących w bibliotece.

7. DODATEK 1 - Format CIF

7.1. Wywołania i transformacje w formacie CIF

Odwołując się do definicji syntaktycznej komendy wywołania C, mamy:

```
callCommand    = "C" integer transformation.
transformation = {{blank} ("T" point | "M" {blank} "X" |
                 "M" {blank} "Y" | "R" point)}.

point      = sInteger sep sInteger.
sInteger   = {sep} ["-"] integerD.
integer    = {sep} integerD.
integerD     = digit {digit}.
sep        = upperChar | blank.
digit      = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
upperChar    = "A" | "B" | "C" | ... | "Y" | "Z".
blank      = dowolny znak ASCII z wyjątkiem digit,
             upperChar, "-", "(", ")", lub ";".
```

Interpretacja semantyczna jest następująca: wywołanie C sprowadza się do wstawienia w realizowany rysunek makrostruktury o numerze określonym parametrem typu integer występującym za symbolem wywołania C, przy czym makrostrukturę poddaje się transformacji "transformation".

Przykład

Komenda C 217 T 0,0;

poleca wstawić symbol (makrostrukturę) 217 jej bazą w punkt (0.0, 0.0) rysunku bez żadnych operacji na symbolu.

Określona w definicji wywołania transformacja dopuszcza cztery możliwe sytuacje, które przedstawiają elementarne przekształcenia na elementach zawartych w wywoływanym symbolu:

T point - Przesunięcie. Umieścić punkt bazowy (0,0) symbolu w punkcie o współrzędnych "point",

MX - Odbicie lustrzane. Przekształcić współrzędne x umieszczanej struktury mnożąc je przez -1,

MY - Odbicie lustrzane. Przekształcić współrzędne y umieszczanej struktury mnożąc je przez -1,

R point - Obrót. Dokonać obrotu symbolu ("jego osią x") w kierunku wyznaczonym wektorem kierunkowym (czyli współrzędnymi rzutów wektora kierunkowego "point").

Jeśli wartości współrzędnych punktów elementów struktury przed transformacją oznaczyć przez x oraz y, a po transformacji x' oraz y', ogólnie transformację można zapisać relacją:

$$[x', y', 1] = [x, y, 1]\, T,$$

gdzie T jest macierzą transformacji.

W przypadku kilku transformacji w komendzie wywołania, wyliczenie współrzędnych punktów po transformacji sprowadza się do obliczenia macierzy:

$$T = T_1 T_2 T_3 T_4,$$

gdzie przez T_1, T_2, T_3, T_4 oznaczono kolejne elementarne transformacje.

7.1.1. Przesunięcie i odbicia lustrzane

Dla przesunięcia T mamy przekształcenie postaci następującej:

$$x' = x + a$$
$$y' = y + b,$$

co daje macierz transformacji T_{ab}:

$$T_{ab} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{bmatrix}.$$

Ponieważ odbicie lustrzane MX zdefiniowane jest zależnością:

$x' = -x$

$y' = y,$

natomiast odbicie lustrzane MY:

$x' = x$

$y' = -y,$

daje to macierze transformacji:

$$M_x = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

oraz odpowiednio

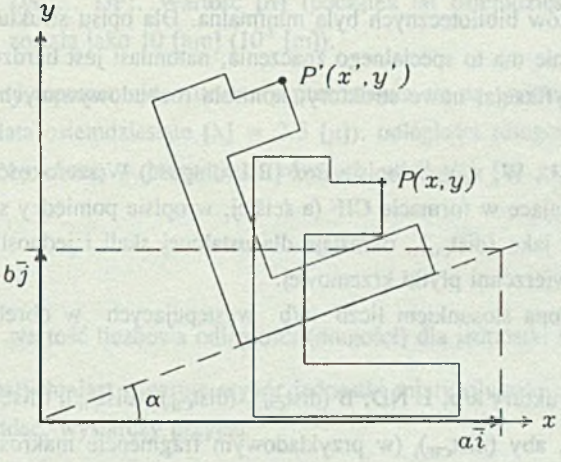
$$M_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

7.1.2. Obrót

Obrót R przedstawiono na rys. 4. Na podstawie rysunku możemy napisać:

$x = cos\alpha\ x' + sin\alpha\ y'$

$y' = tg\alpha\ x' + (1/cos\alpha)\ y,$



Rys. 4. Transformacja R_{ab}

Fig. 4. R_{ab} Transformation

co daje:

$$x' = \cos\alpha\, x - \sin\alpha\, y$$
$$y' = \sin\alpha\, x + \cos\alpha\, y.$$

Jeśli oznaczyć:

$$c=\sqrt{a^2+b^2},$$

to możemy zapisać:

$$\cos\alpha = a/c, \qquad \sin\alpha = b/c,$$

a zatem zachodzi:

$$[x',\ y',\ 1] = [x,\ y,\ 1] \begin{bmatrix} a/c & b/c & 0 \\ -b/c & a/c & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x,\ y,\ 1]\ R_{ab},$$

gdzie R_{ab} jest macierzą transformacji.

7.2. Przeliczanie wartości liczbowej odległości

Opis topografii struktury pojawiający się w różnych formatach, a reprezentujący sytuację na powierzchni płytki krzemowej, stwarza konieczność wyboru odpowiednich współczynników skali z jednej strony i wygodnych jednostek miary z drugiej. Związane jest to z oczywistym wymaganiem, aby przy zmianie parametrów technologicznych ilość zmian w opisie elementów bibliotecznych była minimalna. Dla opisu struktur przechowywanych w bazie danych nie ma to specjalnego znaczenia, natomiast jest bardzo istotne dla działań projektanta (weryfikacja, nowe struktury, kontrola rozbudowywanych struktur bibliotecznych).

Liczby {np.: L, W, x_c,y_c w opisie Box [B L(długość) W(szerokość) x_c,y_c (współrzędne środka)]} występujące w formacie CIF (a ściślej, w opisie pomiędzy symbolami DS i DF) jeśli oznaczyć je jako $(\text{dist}_{CIF})_i$, określają dla ustalonej skali i jednostki miary, odległości i długości na powierzchni płytki krzemowej.

Skala wyrażona stosunkiem liczb a/b występujących w obrębie symboli komend DS ..... DF:

DS NumerStruktury a/b; L ND; B $(\text{dist}_{CIF})_1$ $(\text{dist}_{CIF})_2$ $(\text{dist}_{CIF})_3$ $(\text{dist}_{CIF})_4$; ..... DF;,

jest tak dobrana, aby $(\text{dist}_{CIF})_i$ (w przykładowym fragmencie makrostruktury są to tylko parametry prymitywu Box, umieszczonego na warstwie ND) były liczbami całkowitymi możliwie małymi, możliwie bez zer na najmniej znaczących pozycjach.

Diagramy reprezentujące struktury przedstawia się na wykresach o jednostce długości [λ]. Przez [λ] oznaczono jednostkę długości wyrażoną w [μ] (mikronach) związaną z możliwościami technologicznymi procesu i urządzeń przy produkcji struktur. Przedstawianie struktur w formie takich diagramów ma istotne znaczenie przy zmianie [λ], gdyż opis struktury może pozostać nie zmieniony, podobnie jak diagram, a zmianie ulega tylko skala a/b.

Związek odległości (długości) na powierzchni płytki krzemowej z liczbami określonymi w ramach formatu CIF (DS ... DF;) może być określony arbitralnie [MEAD1, NEWK1].

Zdefiniowano go następującą relacją:

$$d_i = (dist_{CIF})_i[\beta] \tag{7.1}$$

gdzie:

d_i - mające wymiar w [μ] przedstawia odległość lub długość na powierzchni płytki krzemu,

[β] - jest jednostką miary długości dla opisu pomiędzy symbolami DS ... DF;.

Wybór [β] wynika z możliwości technologicznych i został zdefiniowany relacją:

$$[\beta] = (a/b)[\alpha] \tag{7.2}$$

gdzie:

a,b - liczby określające współczynnik skali w opisie CIF,

[α] - jednostka miary długości przyjęta dla opisu "na zewnątrz" definicji DS ... DF;. Wartość [α] (początek lat osiemdziesiątych) przyjęta została jako 10 [nm] (10^{-8} [m]).

Ponieważ diagramy reprezentujące struktury przedstawia się na wykresach o jednostce miary długości [λ] (lata osiemdziesiąte [λ] = 2.5 [μ]). odległości (długości) na powierzchni płytki krzemu wyrażone w jednostkach równych wielkości [λ] można wyznaczyć z zależności:

$$d_i = (D)_i[\lambda] \tag{7.3}$$

gdzie:

$(D)_i$ - wartość liczbowa odległości (długości) dla jednostki miary [λ].

Ostatnia relacja natychmiast sugeruje wybór jednostki miary długości dla edytora graficznego, gdyż jak widać, wystarczy przyjąć

$$(D)_i = (D_{DXF})_i \tag{7.4}$$

gdzie:

$(D_{DXF})_i$ - wartość liczbowa odległości (długości) dla jednostki miary [λ], czyli wprost liczby na displayu edytora graficznego.

Widać stąd, że zmiana [λ] w celu zachowania nie zmienionego diagramu opisującego strukturę (diagram taki nosi też nazwę rzutu zbiorczego lub zbiorczego wydruku struktury) oraz zachowania nie zmienionego opisu CIF (a ściśle wartości odległości/długości pomiędzy symbolami DS i DF), a więc dla nie zmienionych $(D_{DXF})_i$ oraz $(dist_{CIF})_i$ ([α] - stała), wymaga jedynie w opisach struktury w formacie CIF właściwego wyboru skali, określonego stosunkiem liczb a/b.

8. DODATEK 2 - Format DXF

8.1. Struktura pliku DXF

Struktura pliku w formacie DXF, w której oznaczono grupy wyróżniające poszczególne sekcje, jest następująca:

```
  0              {początek sekcji NAGŁÓWKA
SECTION
  2
HEADER
  .....
Lista grup sekcji NAGŁÓWKA
  .....
  0
ENDSEC           koniec sekcji NAGŁÓWKA}
  0              {początek sekcji TABLIC
SECTION
  2
TABLES
  .....
Lista grup sekcji TABLIC
  .....
  0
```

```
    ENDTAB
     0
    ENDSEC              koniec sekcji TABLIC}
     0                  {początek sekcji BLOKÓW
    SECTION
     2
    BLOCKS
    .....
  Lista grup sekcji BLOKÓW
    .....
     0
    ENDSEC              koniec sekcji BLOKÓW}
     0                  {początek sekcji ELEMENTÓW
    SECTION
     2
    ENTITIES
    .....
  Lista grup sekcji ELEMENTÓW
    .....
     0
    ENDSEC              koniec sekcji ELEMENTÓW}
     0
    EOF                 {KONIEC PLIKU}.
```

Najistotniejsza informacja o rysunku przechowywana w formacie DXF zawarta jest w sekcjach bloków i elementów.

8.2. Postać TRACE w pliku DXF

Edycja pliku DXF może być realizowana wprost edytorem tekstowym na odpowiednich grupach w sekcjach BLOKÓW i ELEMENTÓW z zachowaniem zasad opisu elementów obrazu zgodnych ze strukturą pliku DXF. Ponieważ uniwersalny edytor AutoCADa wykorzystujemy do edycji struktur VLSI, ze względu na przyjętą zasadę budowy elementów bibliotecznych, wykorzystującą tylko prymitywy Box w formacie CIF, można przyjąć, że wystarczy w ramach formatu DXF operować tylko prymitywami TRACE w ich najprostszej postaci (prostokąty). Przykładowo przedstawimy postać opisu ścieżki

przedstawiającej prostokąt na warstwie NP, co zilustruje reprezentację tej ścieżki w pliku DXF i na displayu edytora.

Załóżmy, że w celu naniesienia TRACE wykonano komendę:

Command: trace<Enter>
Trace width <0.1000>:2<Enter>
From point:13.26<Enter> {punkt P (rys. 5)}
To point:13.32<Enter> {punkt K (rys. 5)}
To point:<Enter>

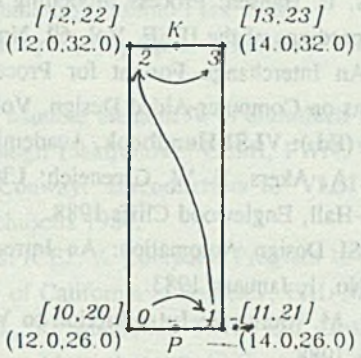
Wykonanie tej komendy wygenerowało ścieżkę, która w pliku DXF ma postać:

```
  0
TRACE
  8
NP
 10
12.0
 20
26.0
 11
14.0
 21
26.0
 12
12.0
 22
32.0
 13
14.0
 23.
32.0
```

Sposób zapisu TRACE w pliku wprowadza numerację wierzchołków prostokąta (od 0 do 3), przyporządkowując odpowiednim współrzędnym odpowiednie grupy i tak:

Numer punktu	Współrzędna	Kod grupy	Wartość grupy
0	x	10	12.0
	y	20	26.0
1	x	11	14.0
	y	21	26.0
2	x	12	12.0
	y	22	32.0
3	x	13	14.0
	y	23	32.0

Obraz graficzny utworzonej w ten sposób TRACE przedstawiono na rys. 5.



Rys. 5. Konstrukcja opisu TRACE
Fig. 5. TRACE description

LITERATURA

[ABE1] M. Abe, T. Mimura, N. Yokoyama, H. Ishikawa: New Technology Towards GaAs LSI/VLSI for Computer Application, IEEE Transactions of Microwave Theory and Techniques, Vol. MTT-30, No. 7, July 1982.

[ANTO1] P. Antognetti, D. A. Antoniadis, R. W. Dutton, W. G. Oldham (Eds.): **Process and Device Simulation for MOS-VLSI Circuits**, Martinus Nijhoff Publishers, Boston-Dordrecht 1983.

[BARK1] J. R. Barker, D. K. Ferry: On Physics and Modeling of Small Semiconductor Devices - I. Solid-State Electronics, Vol. 23, No. 6-A, pp. 519-530, Pergamon Press 1980.

[BLOD1] J. Blödel, R. W. Hartenstein, W. Nebel, M. Ryba: A Technology Description Method for Generalized Layout/Circuit Relations, Microprocessing and Microprogramming, No. 23, pp. 15-20, North-Holland 1988.

[BORR1] D. Borrione (Ed.): **From HDL Descriptions to Guaranteed Correct Circuit Designs**, Proceedings of the IFIP WG 10.2 Working Conference on From HDL Descriptions to Guaranteed Correct Circuit Designs, Grenoble, North-Holland, September 1986.

[DELO1] H. Delori, A. Guyot, J. F. Paillotin: French MPC Activity Report, CMP, Grenoble 1989.

[DUTT1] R. W. Dutton: Data Requirements and Program Interfaces for Simulating Integrated-Circuit Technology, IEEE Electro-Technology Review 1984.

[DUTT2] R. W. Dutton, S. E. Hansen: Process Modelling of Integrated Circuit Device Technology, Proceedings of the IEEE, Vol. 69, No. 10, October 1981.

[DUVA1] S. G. Duvall: An Interchange Format for Process and Device Simulation, IEEE Transactions on Computer-Aided Design, Vol. 7, No. 7, July 1988.

[EINS1] N. G. Einspruch (Ed.): **VLSI Handbook**, Academic Press, New York 1985.

[FERR1] D. K. Ferry, L. A. Akers, E. W. Greeneich: **Ultra Large Scale Microelectronics**, Prentice-Hall, Englewood Cliffs 1988.

[FEUE1] M. Feuer: VLSI Design Automation: An Introduction, Proceedings of the IEEE, Vol. 71, No. 1, January 1983.

[FEUG1] R. J. Feugate, S. M. McIntyre: **Introduction to VLSI Testing**, Prentice Hall, Englewood Cliffs 1988.

[FILI1] A. Filipkowski: **Projektowanie układów scalonych**, Wydawnictwa Politechniki Warszawskiej, Warszawa 1983.

[GAJS1] D. D. Gajski: **Silicon Compilation**, Addison-Wesley, Reading, Massachusetts 1988.

[GOTO1] S. Goto (Ed.): **Design Methodologies**, Advances in CAD for VLSI, Vol. 6, North-Holland, Amsterdam 1986.

[HOLD1] T. Holden: **Knowledge Based CAD and Microelectronics**, North-Holland, Amsterdam 1987.

[JAKU1] A. Jakubowski, W. Marciniak, H. Przewłocki: **Pomiary elektryczne w diagnostyce produkcji układów scalonych LSI i VLSI**, WNT, Warszawa 1991.

[JOHA1] D. Johannsen: Bristle blocks: A Silicon Compiler, Proc. 16th Design Automation Conf., pp. 310-313, June 1985.

[KOLO1] J. F. Kolodziejski: Problemy projektowania łatwo testowalnych układów scalonych, Prace Instytutu Technologii Elektronowej, z. 3, Warszawa 1985.

[KOZM1] K. A. Koźmiński, A. Świt: Automatyzacja projektowania układów scalonych VLSI, Cz. I. Style i metody projektowania, Elektronika, Rok XXIX, nr 1, styczeń 1988.

[KUZM1] W. Kuźmicz: **Projektowanie analogowych układów scalonych**, WNT, Warszawa 1981.

[KUZM2] W. Kuźmicz: **Modelowanie numeryczne przyrządów półprzewodnikowych**, Biblioteka Elektroniki (20), z. 3, VLSI kierunki, bariery i granice wzrostu, PWN, Warszawa 1988.

[KUZM3] W. Kuźmicz: Training in Custom VLSI Design on Apple Macintosh Computers, The Proceedings of the Second Eurochip Workshop on VLSI Design Training, 30 Sept.- 2 Oct., pp. 58-62, Grenoble 1991.

[LIGH1] M. R. Lightner: Modeling and Simulation of VLSI Digital Systems, Proceedings of the IEEE, Vol. 75, No. 6, June 1987.

[MALY1] W. Maly: **Atlas of IC Technologies: An Introduction to VLSI Processes**, The Benjamin/Cummings, Menlo Park 1987.

[MARC1] W. Marciniak: **Przyrządy półprzewodnikowe i układy scalone**, WNT, Warszawa 1987.

[MARC3] W. Marciniak: **Modele elementów w układach scalonych MOS ITE**, Prace Instytutu Technologii Elektronowej CEMI, PWN, Warszawa 1985.

[MEAD1] C. Mead, L. Conway: **Introduction to VLSI Systems**, Addison-Wesley, Reading, Massachusetts 1980.

[NAGE1] L. W. Nagel: SPICE2: A Computer Program to Simulate Semiconductor Circuits, University of California at Berkeley, ERL-M520, May 1975.

[NEWK1] J. A. Newkirk, R. Mathews: **The VLSI Designer's Library**, Addison - Wesley, Reading, Massachussetts 1983.

[NEWT1] A. R. Newton, A. L. Sangiovanni-Vincentelli: CAD Tools for ASIC Design, Proceedings of the IEEE, Vol. 75, No. 6, June 1987.

[NEVI1] L. J. Nevin: GaAs High Speed Digital ICs, Intern. Solid-State Conf. Digest of Techn. Papers, Vol. 26, pp. 31-37, February 1983.

[PAIL1] J-P. Paillotin: Le Système LUCIE, Institute IMAG, Grenoble 1985.

[PALU2] O. A.Palusinski, M. Scheinfein, L. Znamirowski, J. C. Liao, F. Quintero, P. Teschan: Electrical Modeling/Simulation, VLSI Packaging and Interconnection Research, SRC Ann. Project Rev., University of Arizona, Tucson 1986.

[PAWL1] R. Pawłowski: Opracować i uruchomić program translacji formatów opisu struktur układów VLSI dla celów komputerowego wspomagania projektowania, Praca dyplomowa, Instytut Informatyki Politechniki Śląskiej, Gliwice 1991.

[ROSI2] A. T. Rosiński: Symulacja na poziomie przesłań międzyrejestrowych jako metoda badania i weryfikacji układów scalonych VLSI, Biblioteka Elektroniki (20), z. 3, VLSI kierunki, bariery i granice wzrostu, PWN, Warszawa 1988.

[RUBI1] S. M. Rubin: **Computer Aids for VLSI Design**, Addison - Wesley, Reading, Massachusetts 1987.

[SELB1] S. Selberherr: **Analysis and Simulation of Semicoductor Devices**, Springer-Verlag, Wien 1984.

[SHER1] A. T. Sherman: **VLSI Placement and Routing: The PI Project**, Springer-Verlag, New York 1989.

[SMIT1] M. J. S. Smith, C. Portmann, R. Jorgenson, D. W. Mattos, R. Halverson, J. K. Chung, P. Tsachang, C. Anagnostopoulos, R. Rao, P. Valdenaire, H. Ching: Analog CMOS Integrated Circuit Design: Research and Undergraduate Teaching, IEEE Transactions on Education, Vol 32, No. 3, August 1989.

[STA1] C.J. Stangham, B. M. Macdonald: Electrical Characterization of Packages for High Speed Integrated Circuits, 35th Electronic Comp. Conf., Proceedings, pp. 356-364, Washington 1985.

[STRO1] A. J. Strojwas, S. W. Director: VLSI: Linking Design and Manufacturing, IEEE Spectrum, October 1988.

[SWIT1] A. Świt, J. Półtorak: **Przyrządy półprzewodnikowe**, WNT, Warszawa 1979.

[SZE1] S. M. Sze (Ed.): **VLSI Technology**, McGraw-Hill, New York 1983.

[WEEK2] W. T. Weeks: A. J. Jiminez, G, W. Mahoney, D. Mehta, H. Quassemradeh, T. R. Scott: Algorithms for ASTAP - A Network Analysis Program, IEEE Trans. on Circuit Theory, Vol. CT - 20, pp. 628-634, November 1973.

[WONG1] D. G. Wong: **Digital Systems Design**, Edward Arnold, London 1985.

[ZIEL1] R. Zielonko, A. Królikowski: **Metody pomiarowo-diagnostyczne analogowych układów elektronicznych**, WNT, Warszawa 1988.

[ZNAM1] L. Znamirowski: Computing Line Parameters from the Capacitance Measurements. Error Propagation Studies, in: O. A.Palusinski, M. Scheinfein, L. Znamirowski, J. C. Liao, F. Quintero, P. Teschan: Electrical Modeling/ Simulation, VLSI Packaging and Interconnection Research, SRC Ann. Project Rev., University of Arizona, Tucson 1986.

[ZNAM2] L. Znamirowski: Komputerowe wspomaganie projektowania struktur VLSI, Laboratorium 5, Laboratorium Komp. Systemów Aut. Prac. Inż., Instytut Informatyki, Pol. Śląska, Gliwice 1988.

[ZNAM3] L. Znamirowski: System oprogramowania wspomagającego graficzne prace projektowe w mikroelektronice, Laboratorium 4, Laboratorium Komp. Systemów Aut. Prac Inż., Instytut Informatyki, Pol. Śląska, Gliwice 1990.

[ZNAM4] L. Znamirowski: Komputerowe wspomaganie generacji złożonych masek struktur ASIC (w tym Zeszycie).

[MAVO1] J. Mavor, M. A. Jack, P. B. Denyer: **Introduction to MOS LSI Design**, Addison-Wesley 1983.

[GLAS1] L. A. Glasser, D. W. Dobberpuhl: **The Design and Analysis of VLSI Circuits**, Addison-Wesley 1985.

[WEST1] N. Weste, K. Eshraghian: **Principles of CMOS VLSI Design - A Systems Perspective**, Addison-Wesley 1985.

[MUKH1] A. Mukherjee: **Introduction to nMOS & CMOS VLSI Systems Design**, Prentice-Hall 1986.

[ASIC1] ASIC Layout Systems/Designers' Buying Guide, Computer Design, Vol. 27, No. 11, June 1, 1988.

[AUTO2] AutoCAD Release 10, Reference Manual, Autodesk, Inc., Oakland 1989.
[BEL1] Biblioteka Elektroniki (20) Zeszyty 1 do 6, VLSI kierunki, bariery i granice wzrostu, PWN, Warszawa od 1986.
[EDIF1] EDIF - Electronic Design Interchange Format, Version 2 0 0, EDIF Steering Committee, 1987.
[EURO1] EUROCHIP Information, Service Organisation of the CEC VLSI Design Action, S. Augustin, March 1991.
[IEEE1] IEEE Transactions on Computer-Aided Design [of Integrated Circuits and Systems], Roczniki.
[ITE1] Prace Instytutu Technologii Elektronowej CEMI, PWN, Roczniki.
[PROC1] Praca zbiorowa: **Procesy technologiczne w elektronice półprzewodnikowej**, WNT, Warszawa 1980.
[VHDL1] **Electronic Hardware Description "the VHSIC Hardware Description Language (VHDL)"**, US Government Printing Office, Washington 1988.

Recenzent: Doc. dr hab. inż. Wiesław Kuźmicz

Wpłynęło do Redakcji 3 stycznia 1992 r.

Abstract

Set of programming tools and standard formats for ASIC structures description presented in the paper composes a graphics system for masks generation and edition.

Considering published standard cell library, with cells which are well checked, and working cell for VLSI elementary structures [MEAD1, NEWK1], when the proper choice of a standard software takes place, with worked out specialized programming tools (these modules can be very simple ones [ZNAM1]), the problem of layout drawing for ASIC structures can be reduced to a feasible task in a sensible time with unnecessary high advanced tools.

Of course, in the case of design the Full Custom chips, the CAD system has to be equipped with tools specified in section 2.3 (simulators, placement and autorouters, extractors, ERC and DRC checkers, ... etc.), nontheless, the design of the ASIC structures in VLSI "technology" not neccesarily has to be the domain of a very professional environment. But on the other hand this forces a lack of an ASIC chip designers [SMIT1, EURO1, MEAD1].

For this moment it can be said, that the design methodology is forced by CAD tools environment, at the beginning of the Integrated Circuits design (nineteen sixties years

[NEWT1] and later) the state-of-the-art was opposite, it was design necessities which stimulated the development of the specialized design tools.

On the other hand, the logical tendency towards full separation of the design and the Silicon Foundry, with precision definition of interface between the finished design and its technological realization. The silicon compilation and technology processes simulators are the best example of this tendency, this being expressed as well with the trends to organize such consortiums as MOSIS and EUROCHIP for chip fabrication.

For better efficiency, the presented system for ASIC structures layout generation and edition requires the extension with tool modules, and the autorouter for interconnections between library cell routing and geometrical rule checker are of the first importance ones among them. The autorouter's algorithm keeps the design rules but in massive designs there is a neccesity of a little "manual" edition and then, the role of a second mentioned above program is distinguished in practical sense of the layout verification. This is also important in the case of creation of the new, extended structures not appearing in the standard cell library.