ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1979

Seria: AUTOMATYKA z. 49 Nr kol. 612

Wiesław TRACZYK

Instytut Automatyki
Politechnika Warszawska

UOGÓLNIONE OPERATORY CYFROWE

Streszczenie. Przedstawiono propozycję zapisu operatorowego funkcji, realizowanych przez elementy i bloki cyfrowe. Zapis ten obejmuje poziom układów logicznych i przesłań międzyrejestrowych, wiążąc opis sprzętu z formułami algebraicznymi i oprogramowaniem. Od języków RTL i HDL różni się tym, że zamiast sztywnych ograniczeń proponuje ogólne zasady ułatwiające dostosowanie opisu do aktualnych potrzeb. Podano klasyfikację i przykłady operatorów oraz przykłady ich zastosowań.

1. WPROWADZENIE

W procesie syntezy układów cyfrowych przechodzi się przez różne formy opisu tych układów, poczynając od opisu słownego lub sieci działań, a kończąc na schemacie montażowym lub liście połączeń. Jeśli pominie się czysto techniczne opisy montażowe, to okaże się, że końcowymi, pełnymi opisami układów są zazwyczaj schematy. Powstają one niekiedy wprost z opisów słownych i wtedy w procesie syntezy ulegają licznym transformacjom, a jeśli (w najlepszym przypadku) powstają na podstawie formuł logicznych lub arytmetycznych, to postać schematu rzadko jest prostym odpowiednikiem formuły. Dzieje się tak dlatego, że tradycyjne formuły boolowskie i arytmetyczne nie uwzględniają specyficznych cech nowoczesnych układów cyfrowych. W tej sytuacji łatwiej jest narysować schemat, w którym multiplekser realizuje funkcję przełączającą, niż zapisać tę funkcję w postaci przydatnej do realizacji z multiplekserem; łatwiej narysować prostokąt z liczbą 74178 niż wyjaśniać rolę każdego sygnału w tym rejestrze. Ten dominujący "obrazkowy" system przedstawiania układów cyfrowych, obok takich zalet, jak duża czytelność rozwiązań (w przypadku prostych schematów), skrócenie drogi od zadania do realizacji itp., ma jednak istotne wady:

- jest mało czytelny przy złożonych układach i ich składnikach,
- utrudnia wprowadzanie i stosowanie formalnych metod syntezy i optymalizacji rozwiązań,
- nie nadaje się do automatycznego projektowania i maszynowej symulacji układów.

Dla uniknięcia tych wad, a zwłaszcza ostatniej, opracowano liczne wersje innej metody opisu układów cyfrowych - języki przesłań międzyrejestrowych i opisu sprzętu (RTL - register transfer languages, HDL - hardware description languages, itp.) [1, 2, 3, 4].

Niektóre z tych języków bardzo dobrze opisują nie tylko strukturę układu lecz i jego zachowanie, umożliwiają więc zarówno symulację jak i automatyczne projektowanie, nie są jednak wolne od dwóch pozostałych, wymienionych wyżej, wad. Wynika to z faktu, że opisy te, jak na dobrze zdefiniowane języki przystało, są wyposażone w zestaw reguł, symboli i przepisów, które zmniejszają czytelność zapisów, a równocześnie ograniczają zakres stosowalności języka do ram przewidzianych przez autora. Skutek jest taki, że żaden z licznych znanych języków nie znalazł szerszego zastosowania i żaden z nich nie nadaje się wprost do opisu takich na przykład problemów jak synteza układów kombinacyjnych z bloków średniej skali integracji i wyznaczanie sterowań w układach mikroprogramowanych.

Wydaje się, że języki z grupy HDL i RTL dobrze spełniają rolę j ę z y k ó w s t r u k t u r a l n y c h [1], precyzyjnie opisując szczegóły budowy, synchronizacji itp., a znacznie gorzej wywiązują się z zadań j ę z y k ó w f u n k c j o n a l n y c h, w których ważniejsza od szczegółów jest zdolność do opisania czynności na różnym poziomie złożoności układu.

Ponieważ opisy funkcjonalne są bardzo potrzebne w publikacjach, dydaktyce i na etapie projektów koncepcyjnych, a utworzenie nowego, idealnego języka wydaje się mało prawdopodobne, pozostaje rozwiązanie pośrednie: zamiast precyzyjnego języka trzeba określić zespół prostych reguł, w ramach których każdy mógłby sam sobie definiować język potrzebny mu do określonego celu. Z pewnością nie to jest ważne, by wszyscy posługiwali się jednym językiem, lecz aby jedni rozumieli język drugich bez specjalnego wysiłku. Należy więc tak określić ogólne przepisy, aby reguły tworzenia języków były oczywiste, a definicje proste. Niżej przedstawiona jest próba określenia takich przepisów.

2. UOGÓLNIONE OPERATORY CYFROWE

Sprecyzujmy pojęcie układów cyfrowych i funkcje opisu. M o d u ł e m l o g i c z n y m (lub cyfrowym) będziemy nazywali kostkę scaloną, czyli podstawowy element konstrukcyjny układu. E l e m e n t l o g i c z n y (lub funkcjonalny) to układ realizujący jedną prostą funkcję logiczną jednej lub kilku zmiennych) albo elementarną funkcję pamięciową (dotyczącą jednego bitu). B l o k f u n k c j o n a l n y (cyfrowy) to układ realizujący złożone funkcje logiczne albo funkcje logiczne, arytmetyczne, komutacyjne lub pamięciowe nad zmiennymi o postaci słów binarnych, przy czym ustalony stan wejść narzuca wykonanie jednej operacji lub sekwencji

identycznych operacji. Zazwyczaj moduł małej skali integracji zawiera jeden lub kilka elementów logicznych, a blok funkcjonalny złożony jest z wielu modułów małej skali integracji albo z jednego lub kilku modułów średniej skali integracji albo z jednego modułu dużej skali integracji. Połączenie elementów logicznych daje u k ł a d l o g i c z n y, natomiast połączenie bloków funkcjonalnych i ewentualnie elementów logicznych tworzy u k ł a d c y f r o w y. Jak widać - wg tej definicji w skład układów cyfrowych nie wchodzą moduły o rozbudowanych, wielotaktowych operacjach (np. mikroprocesory); dla takich układów rezerwujemy nazwę s y s t e m y c y f r o w e. Układy logiczne będziemy uważali za szczególny przypadek układów cyfrowych.

Funkcjonalny opis układu cyfrowego powinien dotyczyć nie modułów lecz elementów i bloków funkcjonalnych, przy czym pożądany jest ścisły związek między postacią opisu a schematem funkcjonalnym układu. Będzie to spełnione, jeśli elementowi lub blokowi będzie odpowiadał jeden, identyfikujący go o p e r a t o r w języku opisu.

Formuły boolowskie są bardzo wygodne w przypadku elementów AND, OR i NOT, gdyż ściśle wiążę się ze schematem, a ich postać umożliwia łatwe przekształcenia formalne. Byłoby dobrze, aby również po wprowadzeniu bardziej złożonych operatorów niż boolowskie zachowana została w jakimś stopniu postać formuł. A oto inne pożądane cechy opisu:

- działania powinny dotyczyć zarówno zmiennych binarnych jak i ciągów takich zmiennych (słów, wektorów), co wynika z konieczności opisywania zarówno elementów jak i bloków,
- należy zachować jednolity opis dla różnych poziomów układu i różnych jego składników (prostych elementów i złożonych bloków), gdyż często są one łączone,
- symbole operacji powinny łatwo identyfikować operację, a więc wobec dużej liczby funkcji) zalecane są skróty literowe,
- przy dużej liczbie operandów i operacji notacja infiksowa jest niewygodna, natomiast najbardziej czytelna jest notacja prefiksowa,
- opis nie powinien uwzględniać tych cech elementów i bloków, które nie są istotne na poziomie, którego język dotyczy.

Oznaczmy przez u, v, q zmienne binarne, odpowiadające informacyjnym sygnałom wejściowym, wyjściowym i pamięciowym elementów i bloków. Niech - odpowiednio - symbole U, V, Q oznaczają ciągi (wektory) zmiennych binarnych, odpowiadające równocześnie występującym sygnałom, a $\underline{U}$, $\underline{V}$, $\underline{Q}$ - zbiory wartości tych ciągów. Ciągi mogę zawierać kilka składników o postaci wektorów, np. jeśli:

$$U_1 = (u_{10}, u_{11}, u_{12}) \quad \text{i} \quad U_2 = (u_{20}, u_{21}, u_{22}, u_{23})$$

to

$$U = (u_{10}, u_{11}, u_{12}, u_{20}, u_{21}, u_{22}, u_{23}) = \langle U_1, U_2 \rangle.$$

Gdy porządek występowania składników nie musi być podkreślany, będziemy pisać: $U = (U_1, U_2)$. Liczba zmiennych tworzących ciąg U będzie oznaczana przez $l(U)$.

Element lub blok kombinacyjny realizujący funkcję:

$$\varphi : \underline{U} \rightarrow \underline{V}$$

można, w myśl powyższych zaleceń, opisać najogólniej przez:

$$V = OPR\ (U),$$

przy czym dla konkretnych funkcji φ miejsce operatora OPR zajmą inne symbole, np. $v = LUB\ (u_0, u_1)$. Dla wielu bloków ich złożone funkcje dają się łatwo zdekomponować na proste funkcje φ_i za pomocą pewnej grupy selekcyjnych (sterujących) zmiennych wejściowych, które wydzielimy z U i będziemy oznaczać przez S. Dekompozycja ma postać:

$$\varphi = \sum_{i=0}^{2^r-1} \varphi_i \cdot S^{(i)},$$

w której występują operacje logiczne, $S = (s_0, s_1, \ldots, s_{r-1})$ i $S^{(i)} = 1$, gdy wartościowanie binarne S odpowiada całkowitej liczbie dodatniej i (w naturalnym kodzie dwójkowym), natomiast $S^{(i)} = 0$ w pozostałych przypadkach.

Uwzględniając zmienne sterujące można blok kombinacyjny opisać operatorem:

$$V = OPR\ (U:S).$$

Opis układów sekwencyjnych musi obejmować dwie funkcje:

$$\delta : \underline{Q} \times \underline{U} \rightarrow \underline{Q} \quad \text{i} \quad \lambda : \underline{Q} \times \underline{U} \rightarrow \underline{V},$$

a ponieważ i tu można zastosować dekompozycję, zapis rozszerzy się do postaci:

$$(Q; V) = OPR\ (U; Q; S).$$

Rezultatem wykonywania operacji na wektorach mogą być albo wektory (np. przy operacjach arytmetycznych i pamięciowych), albo pojedyncze zmienne binarne, odpowiadające predykatom (np. przy badaniu relacji). Ponieważ operandami predykatów mogą być wyniki o postaci wektorów, więc rezultaty działań predykatowych wygodnie będzie wyłączyć z ciągu V, tworząc ciąg zmiennych kontrolnych P, z których każda jest określona funkcją:

$$\mathcal{K} : \underline{Q} \times \underline{U} \times \underline{V} \rightarrow \{0, 1\}.$$

Po tych zabiegach ogólny opis elementów i bloków funkcjonalnych przybrał postać u o g ó l n i o n e g o o p e r a t o r a c y f r o w e g o:

$$(P; Q; V) = OPR\ (U; Q; S). \tag{1}$$

Jest to ogólny model zapisu, na którym można dokonywać następujących działań:

1) rugowania operandów (np. Q - w układach kombinacyjnych),
2) podziału operandów na wektory składowe (np. V_1, V_2 zamiast V),
3) przypisania nazw operacjom lub zmiennym, np.:
   OPR - ADD; U_1,U_2 - WE1, WE2; s_0,s_1 - Clear, Preset.
4) przypisania wartości operandom.

Zwarta postać (1) uprości relacje między formułę (wykorzystującą takie operatory) a schematem, lecz skomplikuje niekiedy zdefiniowanie samego operatora, który odpowiada rozbudowanej funkcji:

$$\mathcal{T} : \underline{U} \times \underline{Q} \times \underline{S} \rightarrow \underline{P} \times \underline{Q} \times \underline{V} \tag{2}$$

i dlatego przy określeniu operatora będziemy często korzystać z prostszych funkcji $\mathcal{K}$, δ, λ i φ oraz dekompozycji. Konkretne wartości S zamienią opis (1) w zbiór odpowiednio prostszych operatorów albo w formuły arytmetyczne, logiczne, sterujące itp.

Układ cyfrowy o sygnałach wejściowych x, tworzących zbiór $\underline{x}$ i wyjściowych y, tworzących zbiór $\underline{y}$, może być opisany formułą operatorową, złożoną z operatorów cyfrowych. Oznaczmy łącznie zmienne wejściowe i wyjściowe operatora:

$$U^* = (U, S), \quad V^* = (P, V)$$

Niech $\underline{z}$ oznacza zbiór zmiennych pomocniczych, u_i^*, v_i^* - dowolne zmienne wejściowe i wyjściowe i-tego operatora.

F o r m u ł a o p e r a t o r o w a to zbiór operatorów cyfrowych, w których nazwy zmiennych spełniają następujące warunki:

1. $(\forall x \in \underline{x})(\exists i)\ (x = u_i^*)$;
2. $(\forall y \in \underline{y})(\exists i)\ (y = v_i^*)$;
3. $(\forall i)(u_i^* \in \underline{x} \cup \underline{z} \cup \{0, 1\})$
4. $(u_i^* = z) \Rightarrow (\exists j)(z = v_j^*)$

Jeśli $z = v^* = OPR\ (U^*, Q)$, to OPR może zastąpić z w formule, np. zamiast $y = NOR(z_1, x_3)$, $z_1 = NOR(x_1, x_2)$, można napisać

$$y = NOR(NOR(x_1, x_2), x_3).$$

W zbiorze $\underline{x}$ można niekiedy wyróżnić zmienne, których wartościowanie dekomponuje funkcję układu cyfrowego na zbiór funkcji prostszych, podobnie jak to było w przypadku bloków. Stan tych zmiennych sterujących oznaczmy przez $\underline{\tilde{S}}$, a stan pozostałych zmiennych wejściowych przez $\underline{\tilde{U}}$; wobec tego $X = (\underline{\tilde{U}}, \underline{\tilde{S}})$. Wyodrębniając (i oznaczając) podobnie zmienne kontrolne, dostajemy $Y = (\underline{\tilde{P}}, \underline{\tilde{V}})$. Jeśli ze złożenia wszystkich stanów Q operatorów wchodzących w skład formuły utworzy się wektor $\tilde{Q}$, to działanie układu cyfrowego można opisać funkcją:

$$\tilde{\tau}: \underline{\tilde{U}} \times \underline{\tilde{Q}} \times \underline{\tilde{S}} \rightarrow \underline{\tilde{P}} \times \underline{\tilde{Q}} \times \underline{\tilde{V}} \qquad (3)$$

o postaci jak (2), a zatem i operatorem o postaci jak (1). W ten sposób każdy układ cyfrowy, spełniający założenia definicji bloku, można opisać operatorem cyfrowym i uważać za składnik układu bardziej złożonego. Oczywiście minimalnym zbiorem operatorów generujących wszystkie inne operatory cyfrowe jest system strukturalnie pełny.

Uogólniony operator cyfrowy (1) jest tylko modelem, który może i powinien być dostosowywany (za pomocą wymienionych wyżej czterech działań) do klasy rozpatrywanych zadań. Przy wprowadzaniu nowych operatorów należy je zdefiniować za pomocą wcześniej wprowadzonych operatorów lub powszechnie stosowanych formuł arytmetycznych, logicznych itp., a także określić dziedzinę i przeciwdziedzinę.

3. KLASY OPERATORÓW CYFROWYCH I PRZYKŁADY

Wprowadzając ograniczenia do postaci (1) uogólnionego operatora cyfrowego można operatory poklasyfikować. Przypisując pewne cechy całej klasie, unika się powtórzeń przy definiowaniu operatorów klasy, a więc klasyfikacja może być pożyteczna.

W śród wielu możliwych kryteriów klasyfikacji na uwagę zasługuje sprawa podstawowych nośników informacji. W wektorze np. zmiennych wejściowych U, poszczególne bity mogą odpowiadać niezależnym sygnałom albo mogą być powiązane w grupy (słowa), łącznie reprezentujące cyfry, znaki itp. Operatory działające na zmiennych pierwszej grupy będziemy nzywali b i n a r n y m i, natomiast drugiej grupy - w e k t o r o w y m i.

Inne kryterium może wyłączać z grupy operatorów te, które odpowiadają podstawowym funkcjom logicznym (lub arytmetyczno-logicznym), podstawowym funkcjom pamięciowym itp. Niżej podane są przykłady operatorów dla kilku takich klas. Dla operatorów przyjęto skróty angielskie jako bardziej rozpowszechnione.

A. B i n a r n e o p e r a t o r y l o g i c z n e

Modelem operatorów tej klasy może być:

$$v = \underline{BOL}\ (U)$$

ale zazwyczaj różnice między v i p oraz U i S wynikają dopiero z zastosowań, więc te oznaczenia mogę się zmieniać.
Przykłady:

$$v = NOT(u) \quad gdy \quad v = \bar{u},$$

$$v = AND\ (u_1,u_2,u_3) \quad gdy \quad v = u_1 \cdot u_2 \cdot u_3.$$

Podobnie można zdefiniować OR, NOR, NAND, XOR, AOI (And-Or-Invert), a także - dla elementów z otwartym kolektorem - NAND* itd.

B. B i n a r n e o p e r a t o r y p a m i ę c i o w e

Model:

$v = \underline{BOP}\ (U; v; S)$ przy $q = v$.

Na przykład

p r z e r z u t n i k typu W-Z: $v = WZ(v; s_0,s_1)$

może być opisany formułą boolowską:

$$v = s_0 v + \bar{s}_1,$$

lub formułą operatorową:

$$v = NAND(s_1,NAND(s_0,v))$$

lub przez wartościowanie zmiennych sterujących:

$(s_0,s_1) = (10)$: $v = 1$
(01): $v = 0$
(11): $v = v$

Ponieważ zapis $v = v$ nie niesie żadnej informacji, a powinien oznaczać, że stan przerzutnika nie ulega zmianie, zazwyczaj znak równości zastępuje się tu strzałką lub symbolem := (oznaczającym, że lewa strona przyjmuje wartość prawej). Często ważne jest jeszcze podkreślenie, że czynność ta następuje natychmiast po pojawieniu się odpowiedniego stanu S (sterowanie asynchroniczne) lub dopiero w chwili narzuconej specjalnym sygnałem taktującym (sterowanie synchroniczne). Dla uproszczenia, sygnału taktującego (zegarowego) nie będziemy włączać do S, znak: = przypiszemy działaniom s y n c h r o n i c z n y m a := działaniom a s y n c h r o n i c z n y m. W ostatnim opisie zamiast = powinno więc być :=.

Definiowanie operatorów formułę boolowską wiąże operację z formalizmami algebry; formuła operatorowa pokazuje budowę wewnętrzną odpowiedniego układu, natomiast działania uzyskane przy określonych S (tzw. mikrooperacja) opisują dobrze zewnętrzne funkcje układu. W dalszych opisach będziemy stosowali tylko jedną z tych postaci.

Przerzutnik typu D: $v = D(u; v; s_0, s_1)$

może być opisany np. przez:

$(s_0, s_1) = (10)$: $v := 1$
(01): $v := 0$
(11): $v := u$

C. Binarne operatory specjalne

Model ma postać:

$V = \underline{BOS}\ (U; S)$

i umożliwia umieszczenie w tej klasie różnych układów kombinacyjnych które niepasują do klasy A, np.:

Pamięć stała: $(v_0, v_1, \ldots, v_{m-1}) = ROM(u_0, u_1, \ldots, u_{n-1}; s)$,

przy czym

$v_i = s \,.\, f_i(u_0, u_1, \ldots, u_{n-1})$, $i = 0,1,\ldots,m-1$,

a f_i jest dowolną funkcją logiczną.

D. Wektorowe operatory arytmetyczno-logiczne

Model:

$(P; \bar{V}) = \underline{WOL}\ (U_1, U_2; S)$, $l(V) = l(U_1) = l(U_2) = m$.

Przyjmując, że dla operacji OPR z klasy BOL zapis

$V = OPR\ (U_1, U_2)$ oznacza $v_i = OPR\ (u_{1i}, u_{2i})$, $i = 0,1,\ldots,m-1$

możemy wprowadzić całą grupę operacji:

$V = NOT(U)$, $V = OR(U_1, U_2)$, $V = AND(U_1, U_2)$ itd.

oraz operacje złożone:

$V = INV(U; s)$ $s = (0)$: $V = U$
(1): $V = NOT(U)$

Dla przykładów arytmetycznych załóżmy, że słowa U i V reprezentują liczby całkowite bez znaku, w naturalnym kodzie dwójkowym, a + oznacza dodawanie arytmetyczne. Oto kilka typowych operatorów:

$V = INCR(U)$, gdy $V = (U + 1) \bmod 2^m$

$(p_1,p_2) = COMP(U_1,U_2)$, gdy $<p_1,p_2> = <(U_1 > U_2), (U_1 < U_2)>$

$(p; V) = SHL(U;s_1,s_2)$, gdy $s_2 = (0)$: $<p,V> = <0,U>$

(1): $<p,V> = <U,s_1>$

$(p; V) = ADD(U_1,U_2; s)$ gdy $<p, V> = U_1 + U_2 + s$

$(p_1,p_2;V) = ALU(U_1,U_2; s_1,s_2,s_3)$ gdy $p_1 = (V=0)$ oraz

$(s_2,s_3) = (00)$: $(p_2;V) = ADD(U_1,U_2; s_1)$

(01): $(V) = U_1$

(10): $(V) = AND(U_1,U_2)$

(11): $(V) = XOR(U_1,U_2)$

Predykaty są tu zapisywane albo w jawnej postaci, np. $p_1 = (U_1 > U_2)$, albo - dla uproszczenia - jako części składowe wektorów.

E. W e k t o r o w e o p e r a t o r y p a m i ę c i o w e

Ogólny model ma postać: $P; Q; V) = \underline{WOP}(U_D,U_A; Q; S)$, przy

$Q = (Q_0,Q_1,\ldots,Q_{N-1})$, $l(V) = l(U_D) = l(Q_i)$.

Typowym przedstawicielem tej klasy jest p a m i ę ć o s w o b o d n y m d o s t ę p i e:

$(Q; V) = RAM(U_D,U_A; Q; s_0,s_1)$

określona przez

$(s_0,s_1) = (00)$: $(Q;V) := (Q; \underline{C})$

(10): $(Q_0,\ldots,Q_i,\ldots;V) := (Q_0,\ldots,U_D,\ldots,\underline{C})$

(01): $(Q; V) := (Q;Q_i)$,

przy czym $\underline{C}$ oznacza stały wektor, np. (00...0) albo (11...1), a $i = (U_A)$ oznacza dziesiętną liczbę całkowitą, powstałą po zdekodowaniu słowa U_A.

Dla dużej rodziny typowych bloków z pamięcią model $\underline{WOP}$ jest zbyt ogólny i można go uprościć, zakładając $N = 1$ oraz $V = Q$, a wówczas:

$(P; V) = \underline{WOQ}(U; V; S)$, $l(U) = l(V)$

Na przykład - r e j e s t r: $V = REG(U; V; s_0,s_1)$ przy:

$(s_0,s_1) = (00)$: $V; = V$

(01): $V; = U$

(10): $V; = C$

- r e j e s t r p r z e s u w a j ą c y: $(p; V) = SRL(U;V;s_0,s_1,s_2,s_3)$ przy

$(s_0,s_1,s_2) = (xx0)$: jak REG

(001): $(p; V) = SHL(U; s_3,1)$

- l i c z n i k d o d a j ą c y: $V = CNT(U; V; s_0, s_1, s_2)$ przy

$(s_0, s_1, s_2) = (000)$: $V: = V$
(001): $V: = INCR(V)$
(010): $V: = U$
$(1xx)$: $V: = C$

W przypadku b l o k ó w s t e r u j ą c y c h wygodnie jest opisywać operatory językiem przyjętym w określaniu sterowań. Jeśli np. U i V oznaczają różne stany układu, to zamiast mikrooperacji V := U można napisać mikroinstrukcję:

V: go to U,

zamiast:

$$V := \begin{cases} V & \text{gdy } s = 0 \\ U & \text{gdy } s = 1 \end{cases}$$

będzie:

V: if s then go to U else go to V itp.

Prosty układ wyznaczający kolejność sterowań może więc być opisany operatorem:

$$V = SEQ(U; V; s_1, s_2)$$

przy:

$s_2 = (0)$: V: if s_1 then go to U else go to INCR (V)
(1): V: go to C.

F. W e k t o r o w e o p e r a t o r y k o m u t a c j i i k o n w e r s j i

Model klasy: $V = \underline{WOK}\ (U; S)$.
Na przykład - m u l t i p l e k s e r:

$$V = MUX(U_0, U_1, \ldots, U_{N-1}; s_0 s_1, \ldots, s_{r-1}),\ l(V) = l(U_i),$$

przy

$$V = \bar{s}_0 \cdot U_i, \quad i = \xi(s_1, s_2, \ldots, s_{r-1}),$$

- u k ł a d y w s p ó ł p r a c y z s z y n ą:

$$V = BUS(U_0, U_1, \ldots, U_{N-1}; s_0, s_1, \ldots, s_{N-1}),\ l(V) = l(U_i),$$

przy

$$V = \sum_{i=0}^{N-1} s_i \cdot U_i,$$

- d e k o d e r:

$V = DEC(U; s)$ przy $[\bar{V}]_B = \bar{s} \cdot [U]_A$,

gdzie A oznacza np. naturalny kod dwójkowy, a B - kod "1 z n",

- p a m i ę ć s t a ł a:

$V = ROM(U;s)$ przy $s = (0)$: $V = \underline{C}$
(1): $V = f(U)$

i dowolnej funkcji f (jest to inny zapis operacji z klasy C).

4. ZASTOSOWANIA

Dla zilustrowanie przydatności proponowanego zapisu operatorowego przedstawiono niżej kilka przykładów zastosowań wykorzystujących operatory przedstawione w p. 3. Bardziej precyzyjne zdefiniowanie operatorów (np. uwzględniające wyzwalanie zboczem lub poziomem itp.) może wygenerować nowe obszary zastosowań.

A. R o z s z e r z e n i e p o d s t a w f o r m a l n y c h t e o r i i u k ł a d ó w c y f r o w y c h

Coraz powszechniej do budowy układów kombinacyjnych stosuje się multipleksery. Formalnym usankcjonowaniem tego może być następujące twierdzenie:

- MUX tworzy System Funkcjonalnie Pełny przy $N \geq 2$.

Najprostszy dowód dla dolnych ograniczeń, tzn. $N = 2$, $l(V) = l(U_i) = 1$ wynika z faktu, że wówczas:

$$v = MUX(u_0,u_1; s_0,s_1)$$

i dla $s_0 = 1$ przy $U = (10)$ jest $v = \bar{s}_1$
$U = (0s)$ $v = s \cdot s_1$.

B. U ł a t w i e n i e o p i s u u k ł a d ó w z m o d u ł ó w SSI i MSI

Układ kombinacyjny opisany zależnościami:

$$y_1 = f_1(x_1,x_2,x_3) = \sum (0,1,3,4)$$
$$y_2 = f_2(x_1,x_2,x_3) = \sum (0,3,4,7)$$
$$y_3 = f_3(x_1,x_2,x_3) = \sum (2,3,5,7)$$

można opisać fomułę operatorową:

$$(z_0,z_1,\ldots,z_7) = DEC(x_1,x_2,x_3; 1)$$

$$y_1 = \text{NAND}\ (z_0, z_1, z_3, z_4)$$
$$y_2 = \text{NAND}\ (z_0, z_3, z_4, z_7)$$
$$y_3 = \text{NAND}\ (z_2, z_3, z_5, z_7)$$

Formuła ta wynika wprost z postaci kanonicznej funkcji, a jednocześnie precyzyjnie opisuje realizację.

C. U j e d n o l i c e n i e o p i s u m o d u ł ó w

Wprawdzie operatory były przeznaczone do funkcjonalnego opisu elementów i bloków, ale mogą być również wykorzystane do jednolitego opisu modułów (zwłaszcza MSI i LSI), ułatwiające ich zastosowanie. Na przykład moduł UCY 74157 można opisać operatorem:

$$v_i = \text{MUX}(u_{i0}, u_{i1}; s_0, s_1), \quad i = 1,2,3,4$$

albo

$$V = \text{MUX}(U_0, U_1; s_0, s_1), \quad l(V) = 4.$$

Wprowadzając umowne oznaczenie:

$$\underset{i=0}{\overset{k}{\text{OPR}}}\ (f(i); S) = \text{OPR}\ (f(0), f(1), \ldots, f(k); S)$$

można zasadę budowania mulitpleksorów dwupoziomowych przedstawić w postaci:

$$\text{MUX}(U; S) = \underset{i=0}{\overset{2^r-1}{\text{MUX}}}\ (\text{MUX}\ (U_i; 1, s_1); S_0)$$

przy

$$U = \langle U_0, U_1, \ldots, U_{2^r-1} \rangle, \quad S = \langle S_0, S_1 \rangle$$
$$l(S_0) = r + 1, \quad l(U_i) = 2^{l(S_1)}.$$

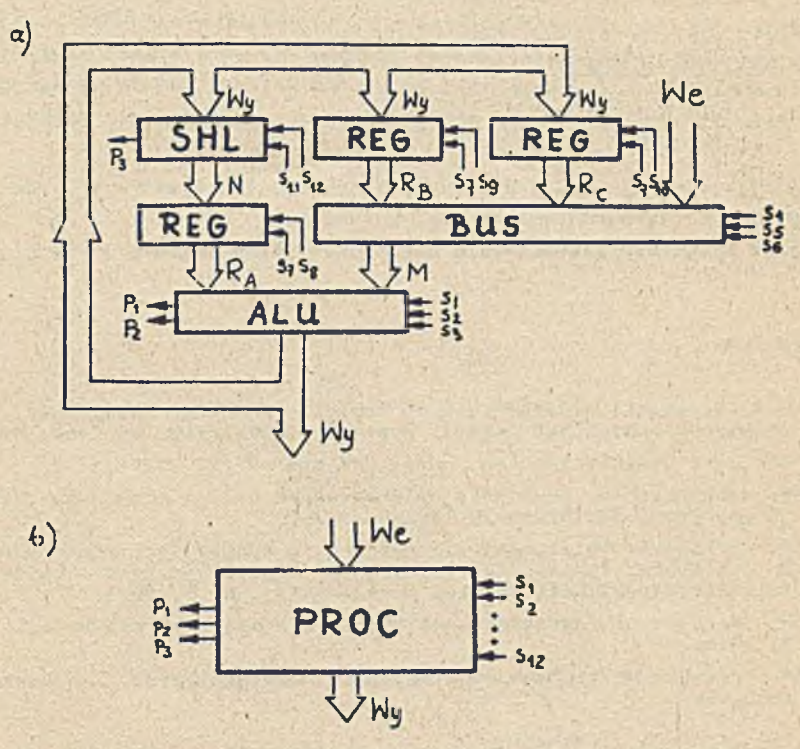
D. D e f i n i o w a n i e n o w y c h b l o k ó w f u n k c j o n a l n y c h

Składniki procesora, złożonego z trzech rejestrów, bloku arytmetyczno-logicznego i układu przesuwającego, połączonych szyną wewnętrzną M i zewnętrzną Wy, można opisać operatorami:

$$(p_1, p_2; Wy) = \text{ALU}(M, R_A; s_1, s_2, s_3)$$
$$M = \text{BUS}(R_B, R_C, We; s_4, s_5, s_6)$$
$$R_A = \text{REG}(N; R_A; s_7, s_8)$$
$$R_B = \text{REG}(Wy; R_B; s_7, s_9)$$
$$R_C = \text{REG}(Wy; R_C; s_7, s_{10})$$
$$(p_3; N) = \text{SHL}(Wy; s_{11}, s_{12})$$

Cały procesor jest więc blokiem funkcjonalnym o mikrooperacjach powstałych za złożenia mikrooperacji składników i opisie:

$$(p_1, p_2, p_3; Q; Wy) = \text{PROC}(We; Q; s_1, s_2, \ldots, s_{12})$$
$$Q = \langle R_A, R_B, R_C \rangle$$



Rys. 1. Procesor jako połączenie bloków (a) i blok (b)

E. P o w i ę z a n i e s p r z ę t u z o p r o g r a m o w a n i e m

Niektóre ciągi mikrooperacji procesora (w szczególności - pojedyncze mikrooperacje) tworzą "makrooperacje" układu cyfrowego, którego składnikiem jest procesor. Tym makrooperacjom przypisuje się zazwyczaj specjalne nazwy, tworzące język wewnętrzny układu (systemu) cyfrowego. Nazwa operacji (lub szerzej - instrukcji) jest więc również nazwą ciągu sterowań procesora. Na przykład w opisanym wyżej procesorze operacji ADD A,B - dodania zawartości akumulatora R_A i rejestru pomocniczego R_B - odpowiada wektor sterujący:

S = (000 100 0100 00),

gdyż

$(s_1, s_2, s_3) = (000)$ wymusza $Wy = M + R_A$,
$(s_4, s_5, s_6) = (100)$ daje $M = R_B$,
$(s_7, s_8, s_9, s_{10}) = (0100)$ powoduje $R_A := N$,
$(s_{11}, s_{12}) = (00)$ oznacza, że $N = Wy$,

co łącznie daje (z pominięciem sygnałów p):

$$R_A := R_B + R_A.$$

Podobnie ciąg sterowań:

$$S^1 = (001\ 001\ 0100\ 00)$$
$$S^2 = (000\ 100\ 0100\ 00)$$

realizuje operację ADD D,B - dodania zawartości rejestru R_B do danych przesyłanych szyną wejściową (np. z pamięci).

Inne przykłady zastosowania operatorów można znaleźć w [5, 6].

LITERATURA

[1] M.R. Barbacci: A Comparison of Register Transfer Languages for Describing Computer and Digital Systems, IEEE Trans. on Comp. Nº 2, 1975.

[2] Hardware Description Languages, Computer Nº 12, 1974.

[3] P. Dembiński, S. Budkowski: Microprogram Design and Description Language, Proc. EUROMICRO-78 Munich, 1978.

[4] M. Perkowski: Relational - Structure Languages and their Application in the System for Automatic Design of Digital Systems, Raport Instytutu Automatyki Politechniki Warszawskiej, Nr 9, 1975.

[5] W. Traczyk: Mikroprogramowane układy sterujące, Archiwum A i T, Nr 1, 2, 1978.

[6] W. Traczyk: Multipleksery w układach kombinacyjnych, Archiwum A i T (w druku).

ОБОБЩЕННЫЕ ЦИФРОВЫЕ ОПЕРАТОРЫ

Р е з ю м е

В работе представлено предложение операторной записи функций, реализуемых цифровыми элементами и блоками. Запись включает уровень логических схем и междурегистровых передач, связывая описание оборудования с алгебраическими формулами и программным обеспечением. От известных языков она отличается тем, что вместо жёстких ограничений предлагает общие принципы, облегающие приспособление описания к актуальным нуждам. Поданы классификация и примеры операторов, а также приведены примеры их применения.

THE GENERALIZED DIGITAL OPERATORS

S u m m a r y

A proposal of the operator function description is presented, realised by the digital elements and building blocks. The notation includes logic

circuits end register transfer levels, linking the hardware description with algebraic formulae and software. It differs from RTL and HDL because instead of fixed limitations it suggests general principles, allowing for easier adaptation of description to the current needs. The classification and the examples of operators as well as their applications are presented