INFORMATICA 3/85

LOGIČNI MODELI RAČUNALNIŠKIH STRUKTUR

MAKSIMILJAN GERKEŠ

TEHNIŠKA FAKULTETA, MARIBOR
VTO ELEKTROTEHNIKA, RAČUNALNIŠTVO IN INFORMATIKA

UDK: 681.3.517.11/.12

Koncept stanja in operacije, kot je znan iz snovanja programske opreme, je izhodiščni koncept z nekoliko širšo interpretacijo. Nabor sestavljenih operacij tvorijo selektorska, sekvenčna in paralelna operacija. Definirana je zančna operacija, ki ima poljubno mnogo ponovitev in nima izhoda. Transformacije, definirane nad sestavljenimi operacijami omogočajo njihovo preoblikovanje, tako da jih lahko modeliramo z izbranimi mikroelektronskimi komponentami male, srednje, velike, pa tudi zelo velike stopnje integracije, vključno z logičnimi mrežami. Na tej osnovi so zgrajeni modeli nekaterih značilnejših logičnih in računalniških struktur.

LOGICAL MODELS FOR COMPUTER STRUCTURES: The concept of state and operation, as it is known from software design is basic concept with extended interpretation. Collection of compound operations consists of select, sequential and parallel operation. An infinite loop operation with no exit terminal is defined. Compound operations can be changed with a set of defined transformations into a different forms which can be simply modeled with SSI, MSI, LSI, or even VLSI structures, including gate arrays. Models for some characteristic logic and computer structures are proposed on that base.

UVOD:

Snovanje računalniških struktur postaja z razvojem mikroelektronske tehnologije vedno bolj kompleksno opravilo. Ob predpostavki, da omogočajo metode programskega snovanja učinkovito reševanje nalog na področju programske opreme, se zdi vprašanje, ali je možno te metode enako učinkovito uporabljati tudi za snovanje strojne opreme povsem utemeljeno. Eksplicitен odgovor na tako zastavljeno vprašanje bi bil zaenkrat precej spekulativen. Lažje je odgovoriti tako, da je možno s smiselno prireditvijo teh postopkov, dosečі z njimi dobre rezultate tudi na področju snovanja strojne opreme.

Začetni zapis računalniške strukture, ki jo želimo realizirati običajno pojmujemo kot nekakšno amorfno strukturo, saj zaradi semantične razdalje v splošnem ni možen neposreden prehod na logično izvedbo te strukture. Izkušnje učijo, da ni smotrno vnaprej predpostavljati organizacijo takšne strukture na logičnem nivoju, ampak da jo je potrebno izpeljati iz lastnosti specifikacije, iz katere izhajamo, z upoštevanjem zunanjih parametrov-hitrost, cena, zanesljivost, ...

Ob takšnem izhodišču nas postopki snovanja strojne opreme s pomočjo modelov brez večjih neprijetnih presenečenj vodijo do želene realizacije. Koraki, ki jih pri tem izvajamo, so podobni snovanju programske opreme, le da so osnovne strukture drugačne. Upoštevati pa moramo tudi zunanje parametre, kamor sodijo tudi realne lastnosti mikroelektronskih komponent, ki jih lahko idealiziramo samo na višjih abstraktnih nivojih snovanja. Neupoštevanje zunanjih parametrov lahko povzroči, da moramo sicer korektno zasnovano in logično pravilno rešitev opustiti in poiskati novo, ki bo dovolj upoštevala te zahteve.

Koncept stanja in operacije, kot ga poznamo iz snovanja programske opreme, je izhodiščni koncept, le da ga interpretiramo nekoliko širše. Nabor sestavljenih operacij je drugačen in sestoji v osnovi iz selektorske, sekvenčne in paralelne operacije. Za izgradnjo modelov sekvenčnih krmilnih enot pa je definirana zančna operacija s poljubno mnogo ponovitvami.

Preoblikovanje sestavljenih operacij omogoča nabor transformacij, s katerimi lahko le-te preoblikujemo tako, da najdemo zanje - ali jih sami definiramo - primerne mikroelektronske gradnike, ki so lahko male, srednje, velike, pa tudi zelo velike stopnje integracije, vključno z logičnimi mrežami.

1. STANJA IN OPERACIJE

Koncept stanja in operacije smiselno prilagodimo za potrebe snovanja strojne opreme. S tem bo prehod iz formalizirane specifikacije računalniške strukture na njen logični model razmeroma tekoč. Kot iztočnico uporabimo koncept stanja in operacije, tako kot je specificiran v /1/. Formalnega dela definicije ne bomo spreminjali in

bomo stanje pojmovali samo kot nabor imenovanih vrednosti.

Pojem operacije ponazorimo kot prireditev, ki začetnemu stanju priredi končno stanje.

Formulo, ki določa pogoje za izvajanje operacije povzamemo po / 1/:

$$(\forall s \in S)(Pi(s) \rightarrow Po\ (s, ex\ (Op, s))) \ . \qquad (1.1)$$

Za vsako stanje s iz prostora stanj S, ki izpolnjuje začetno trditev določeno s predikatom Pi, se z izvajanjem operacije Op določi (izračuna) izhodno stanje ex(Op,s), ki je z začetnim stanjem povezano s končno trditvijo, ki jo specificira predikat Po.

S takšno opredelitvijo stanja in operacije se ne želimo omejiti na krmiljenje operacij, po principu ... izvedi operacijo i, izvedi opredijo i+1 ... .

Če ponovno preberemo (1.1) lahko specificiramo krmilni pogoj za izvajanje operacije tudi takole ..če je zadano stanje s trditev določena s predikatom Pi izpolnjena, tedaj se operacija Op lahko izvede ... . Zadnjo izjavo razširimo takole: ... izvedejo se lahko vse tiste operacije izmed $Op_1, Op_2, \ldots, Op_n$, za katere velja, da so pripadajoče trditve $Pi_1, Pi_2, \ldots, Pi_n$ izpolnjene nad stanji $s_1, s_2, \ldots, s_n$ ... .

Za to izjavo najdemo v praksi pogosto naslednji približek: ... izvedejo se lahko vse tiste instrukcije, za katere velja, da imajo veljavne izvorne operande ... .

Če povzamemo, lahko rečemo, da je možno operacije v obeh skrajnostih izvajati s krmilnim oz. podatkovnim pretokom. Formula (1.1) predpisuje samo pogoj, kdaj se operacija lahko izvede, ne predpisuje pa, kdo je tisti, ki ugotavlja izpolnjenost pogoja - človek oz. stroj.

1.1. MODEL PODATKA

Za logični model računalniške strukture prilagojen zapis stanj oz. njihovih komponent, izgradimo model s pomočjo izjav, s katerimi opisujemo elemente izbranih množic, katerih člani so vrednosti komponent stanj, imena komponent stanj, imena operacij, ...

Vzemimo množico elementov D in vsakemu elementu, ki je član te množice, priredimo izjavo, ki le-tega enolično opisuje. Če ima množica D m elementov, je izjav, ki te elemente opisujejo prav tako m. Izjave označimo z $D_{m-1}, D_{m-2}, \ldots, D_0$.

Sedaj pa izberimo še n izjav ob pogoju $m \leq 2^n$, ki jih označimo z $A_{n-1}, A_{n-2}, \ldots, A_0$ in zaenkrat ignorirajmo vsebino teh izjav. Če tvorimo vse možne konjunkcije teh izjav, s tem da pravilnostne vrednosti izjavam sami predpišemo dobimo:

$$\begin{aligned}
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge \bar{A}_1 \wedge \bar{A}_0 \\
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge \bar{A}_1 \wedge A_0 \\
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge A_1 \wedge \bar{A}_0 \\
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge A_1 \wedge A_0 \\
&\quad \vdots \\
&A_{n-1} \wedge A_{n-2} \wedge \ldots \wedge A_1 \wedge \bar{A}_0 \\
&A_{n-1} \wedge A_{n-2} \wedge \ldots \wedge A_1 \wedge A_0 \ .
\end{aligned} \qquad (1.1.1)$$

Sedaj pa tvorimo m ekvivalenc, tako da vsaki izjavi izmed $D_0, D_1, \ldots, D_{m-1}$ predpišemo kot ekvivalentno izjavo poljubno konjunkcijo izmed (1.1.1), vendar tako, da bo prireditev enolična.

Za zgled predpostavimo, da je $m=2^n$ in tvorimo eno izmed možnih prireditev.

$$\begin{aligned}
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge \bar{A}_1 \wedge \bar{A}_0 = D_0 \\
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge \bar{A}_1 \wedge A_0 = D_1 \\
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge A_1 \wedge \bar{A}_0 = D_2 \\
&\bar{A}_{n-1} \wedge \bar{A}_{n-2} \wedge \ldots \wedge A_1 \wedge A_0 = D_3 \\
&\quad \vdots \\
&A_{n-1} \wedge A_{n-2} \wedge \ldots \wedge A_1 \wedge \bar{A}_0 = D_{m-2} \\
&A_{n-1} \wedge A_{n-2} \wedge \ldots \wedge A_1 \wedge A_0 = D_{m-1}
\end{aligned} \qquad (1.1.2)$$

Izjavam $A_{n-1}, A_{n-2}, \ldots, A_1, A_0$ priredimo pravilnostne vrednosti glede (1.1.2) in dobimo:

A_{n-1}	A_{n-2}	...	A_1	A_0	
0	0	...	0	0	D_0
0	0	...	0	1	D_1
0	0	...	1	0	D_2
0	0	...	1	1	D_3
		⋮			
1	1	...	1	0	D_{m-2}
1	1	...	1	1	D_{m-1}

(1.1.3)

Če si zapomnimo prireditev (1.1.3) lahko z nizi pravilnostnih vrednosti izjav $A_{n-1}, A_{n-2}, \ldots, A_1, A_0$ ponazorimo elemente množice D.

Na opisan način lahko praktično elemente poljubnih mno-

žic priredimo za logični nivo pri izgradnji modelov računalniških struktur.

Doslej nas notranja zgradba izjav ni posebej zanimala. Včasih pa lahko z upoštevanjem notranje zgradbe izjav, izgradimo modele, ki imajo podobne lastnosti kot originalni podatki. Takšen pristop nam včasih olajša izgradnjo modelov operacij nad tako modeliranimi podatki.

Kot zgled uporabimo množico dvojiških števil, ki jo opišemo z izrazom:

$$a_{n-1}.2^{a_{n-1}} + \ldots + a_1.2^1 + a_0.2^0 , \qquad (1.1.4)$$

kjer je $a_i \in \{0, 1\}_2$ in $i = 0, 1, \ldots, n-1$.
Sedaj specificirajmo n izjav takole:
Koeficient a_i ima vrednost 1_2. (1.1.5)

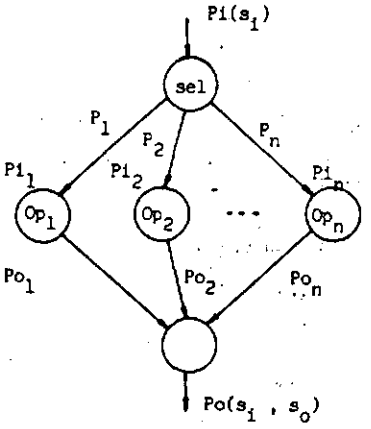
$a_i = 0, 1, \ldots, n-1$

Izjava je pravilna, če je trditev resnična, drugače je napačna. Pri takšnem modelu lahko na primer seštevalnik po mod_2 nad dvojiškimi števili ponazorimo z operacijo logične ekvivalence nad izjavami (1.1.5).

2. SESTAVLJENE OPERACIJE

2.1. SELEKTORSKA OPERACIJA

Selektorsko operacijo ponazorimo z označenim usmerjenim grafom na sliki 2.1.1.



Slika 2.1.1: Graf selektorske operacije

Selektorska operacija je sestavljena operacija, kjer se naenkrat lahko izvede samo ena izmed operacij Op_1, $Op_2, \ldots, Op_n$. Katera operacija se bo izvedla, je odvisno od pravilnosti ene izmed izjav $P_1, P_2, \ldots P_n$, ki jih definiramo takole:

$$P_1 = R_1(e_1(s_i), s_i) \wedge \overline{R_2(e_2(s_i), s_i)} \wedge \ldots \wedge \overline{R_n(e_n(s_i), s_i)}$$
$$P_2 = \overline{R_1(e_1(s_i), s_i)} \wedge R_2(e_2(s_i), s_i) \wedge \ldots \wedge \overline{R_n(e_n(s_i), s_i)}$$
$$\vdots \qquad (2.1.1)$$
$$P_n = \overline{R_1(e_1(s_i), s_i)} \wedge \overline{R_2(e_2(s_i), s_i)} \wedge \ldots \wedge R_n(e_n(s_i), s_i) .$$

V izrazih (2.1.1) so $R_1, R_2, \ldots, R_n$ predikati, s_i je začetno stanje, e_j, $j = 1, 2, \ldots, n$ pa so funkcije.

Logična pravila, s katerimi opišemo selektorsko operacijo, so:

$$Pi(s_i) \wedge P_1 \longrightarrow Pi_1(s_i)$$
$$Pi(s_i) \wedge P_2 \longrightarrow Pi_2(s_i)$$
$$\vdots$$
$$Pi(s_i) \wedge P_n \longrightarrow Pi_n(s_i) \qquad (2.1.2)$$

$$Pi(s_i') \wedge P_1 \wedge Po_1(s_i', s_0) \longrightarrow Po(s_i', s_0)$$
$$Pi(s_i') \wedge P_2 \wedge Po_2(s_i', s_0) \longrightarrow Po(s_i', s_0)$$
$$\vdots$$
$$Pi(s_i') \wedge P_n \wedge Po_n(s_i', s_0) \longrightarrow Po(s_i', s_0) .$$

Nad spodnjo polovico izrazov (2.1.2) uporabimo formulo $0 \vee \ldots \vee 0 \vee A \vee 0 \vee \ldots \vee 0 = A$ in dobimo:

$$(Pi(s_i') \wedge P_1 \wedge Po_1(s_i', s_0)) \vee$$
$$\vee Pi(s_i') \wedge P_2 \wedge Po_2(s_i', s_0) \vee \qquad (2.1.3)$$
$$\vdots$$
$$\vee Pi(s_i') \wedge P_n \wedge Po_n(s_i', s_0)) \longrightarrow Po(s_i', s_0) .$$

Če upoštevamo še: $(A \wedge B) \vee \ldots \vee (A \wedge C) = A \wedge (B \vee \ldots \vee C)$ dobimo naslednji izraz:

$$Pi(s_i') \wedge [P_1 \wedge Po_1(s_i', s_0) \vee P_2 \wedge Po_2(s_i', s_0) \vee \ldots$$
$$\ldots \vee P_n \wedge Po_n(s_i', s_0)] \longrightarrow Po(s_i', s_0). \qquad (2.1.4)$$

V izrazih (2.1.2) do (2.1.4) je s_i' spremenjeno stanje s_i, ki ga povzroči operacija Op_1 ali Op_2 ali ... .

Zapis (2.1.4) sicer s stališča snovanja programske opreme ni posebno zanimiv, vendar je njegova zgradba značilna v toliko, da nas navede na definicijo poenostavljene selektorske operacije, ki jo specificirajmo takole:

sel (P_j):
$$P_1 \longrightarrow Op_1$$
$$P_2 \longrightarrow Op_2 \qquad (2.1.5)$$
$$\vdots$$
$$P_n \longrightarrow Op_n$$

$P_1, P_2, \ldots, P_n$ so izjave, $Op_1, Op_2, \ldots, Op_n$ pa izjave ali podatki modelirani v smislu razdelka (1.1). Za izjave $P_1, P_2, \ldots, P_n$ ponovno velja pogoj, da je lahko naenkrat pravilna samo ena izmed njih.

Z izrazom $A \wedge (A \longrightarrow B)$ zožimo pravilnostni prostor implikacije, tako da je enak prostoru pravilnosti konjunk-

cije A∧B. Napravimo to za implikacije v (2.1.5).

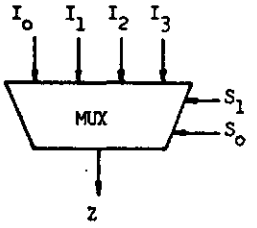
$$P_1 \wedge (P_1 \longrightarrow Op_1) = P_1 \wedge Op_1$$
$$P_2 \wedge (P_2 \longrightarrow Op_2) = P_2 \wedge Op_2 \qquad (2.1.6)$$
$$\vdots$$
$$P_n \wedge (P_n \longrightarrow Op_n) = P_n \wedge Op_n$$

Upoštevamo $0 \vee \ldots \vee 0 \vee A \vee 0 \vee \ldots \vee 0 = A$ in zapišemo:

$$P_1 \wedge Op_1 \vee P_2 \wedge Op_2 \vee \ldots \vee P_n \wedge Op_n \; . \qquad (2.1.7)$$

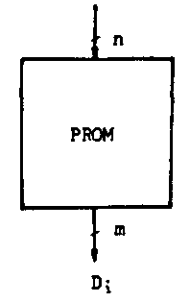
Izraz (2.1.7) sicer strogo gledano ni ekvivalenten zapisu (2.1.5), vendar se dogovorimo, da bomo (2.1.5) brali takole ... če je Pi pravilna, tedaj je pravilna tudi Op_i .... .

Na sliki 2.1.2 so podani zgledi modelov nekaterih tipičnih gradnikov s pomočjo poenostavljene selektorske operacije.



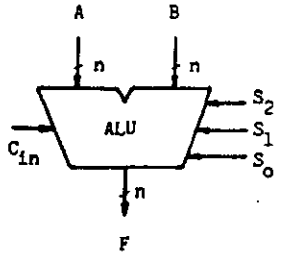
$$Z = (\bar{S}_1 \wedge \bar{S}_0) \wedge I_0 \vee (\bar{S}_1 \wedge S_0) \wedge I_1 \vee (S_1 \wedge \bar{S}_0) \wedge I_2 \vee (S_1 \wedge S_0) \wedge I_3$$

a) model multipleksirnika



$$D = (\bar{A}_{n-1} \wedge \ldots \wedge \bar{A}_1 \wedge \bar{A}_0) \wedge D_0 \vee (\bar{A}_{n-1} \wedge \ldots \wedge \bar{A}_1 \wedge A_0) \wedge D_1 \vee (\bar{A}_{n-1} \wedge \ldots \wedge A_1 \wedge \bar{A}_0) \wedge D_2 \vee \vdots \vee (A_{n-1} \wedge \ldots \wedge A_1 \wedge A_0) \wedge D_{2^n-1}$$

b) model bralnega pomnilnika

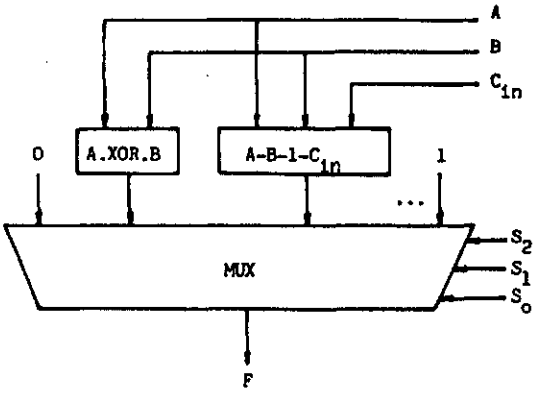


$$F = (\bar{S}_2 \wedge \bar{S}_1 \wedge \bar{S}_0) \wedge \emptyset \vee (\bar{S}_2 \wedge \bar{S}_1 \wedge S_0) \wedge (A.XOR.B) \vee (\bar{S}_2 \wedge S_1 \wedge \bar{S}_0) \wedge (A - B - 1 + C_{in}) \vee (\bar{S}_2 \wedge S_1 \wedge S_0) \wedge (A.AND.B) \vee (S_2 \wedge \bar{S}_1 \wedge \bar{S}_0) \wedge (B - A - 1 + C_{in}) \vee (S_2 \wedge \bar{S}_1 \wedge S_0) \wedge (A \; . \; OR \; . \; B) \vee (S_2 \wedge S_1 \wedge \bar{S}_0) \wedge (A + B + C_{in}) \vee (S_2 \wedge S_1 \wedge S_0) \wedge 1$$

c) model ALU operacijske enote

Slika 2.1.2: Zgledi uporabe poenostavljene selektorske operacije za izgradnjo logičnih modelov

Pri tem smo ponovno predpostavili, da so skrajno desni konjunktivni členi ponazorjeni v smislu razdelka 1.1. Zaradi lažje orientacije je na sliki 2.1.3 podan še eden izmed možnih strukturnih modelov za c) s slike 2.1.2.



Slika 2.1.3: Blokovna shema možnega strukturnega modela za c) 2.1.2

2.2. SEKVENČNA OPERACIJA

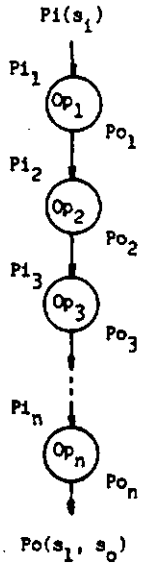
Pri specifikaciji sekvenčne operacije izhajamo iz grafa na sliki 2.2.1.

Logična pravila za sekvenčno operacijo zapišemo takole:

$$Pi(s_1) \longrightarrow Pi_1(s_1)$$
$$Pi(s_1) \wedge Po_1(s_1, s_2) \longrightarrow Pi_2(s_2) \qquad (2.2.1)$$
$$Pi(s_1) \wedge Po_1(s_1, s_2) \wedge Po_2(s_2, s_3) \longrightarrow Pi_3(s_3) \; \ldots$$
$$Pi(s_1) \wedge Po_1(s_1, s_2) \wedge Po_2(s_2, s_3) \wedge \ldots \wedge Po_{n-1}(s_{n-1}, s_n) \longrightarrow \longrightarrow Pi_n(s_n)$$

$$Pi(s_1) \wedge Po_1(s_1, s_2) \wedge Po_2(s_2, s_3) \wedge \ldots \wedge Po_n(s_n, s_0) \longrightarrow \longrightarrow Po(s_1, s_0).$$

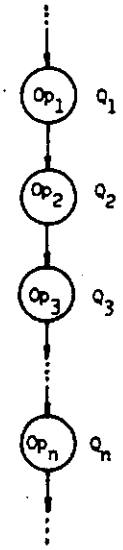
V (2.2.1) smo upoštevali, da lahko Op_1 spremeni kom-



Slika 2.2.1: Graf sekvenčne operacije

ponente v s_i, ki zato preide v s_1.

Do zanimivih zaključkov pridemo, če izgradimo sekvenčno krmiljen model sekvenčne operacije. V ta namen predpostavimo, da je sekvenčna operacija s slike 2.2.1 del sestavljene operacije in jo ponazorimo po sliki 2.2.2.

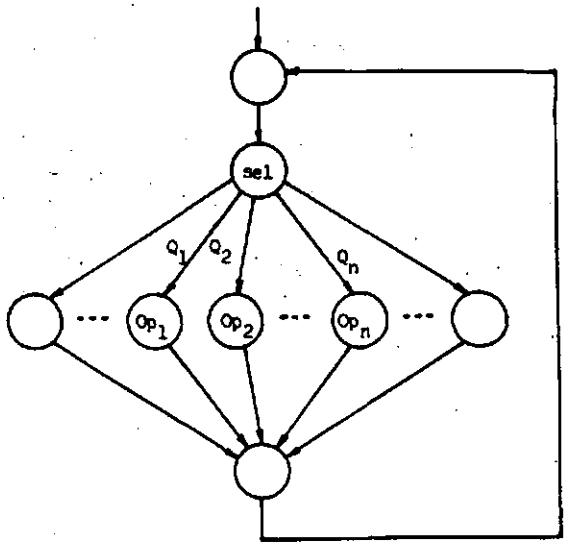


Slika 2.2.2: Prirejen graf sekvenčne operacije

Vozliščem grafa na sliki 2.2.2 smo pripisali izjave $Q_1, Q_2, \ldots, Q_n$. Izjava Q_l, $l = 1, 2, \ldots, n$ je pravilna tedaj in le tedaj, ko je glede na sekvenčno operacijo s slike 2.2.1 izpolnjena pripadajoča izjava $Pi_l(s_l)$ in se operacija Op_l še ni izvedla. Tedaj lahko graf s slike 2.2.2 preoblikujemo v selektorsko operacijo in zapišemo:

$$\underline{1}\text{: sel } (Q_l):$$
$$\begin{aligned} &Q_1 \longrightarrow Op_1 \\ &Q_2 \longrightarrow Op_2 \\ &\quad\vdots \\ &Q_n \longrightarrow Op_n \\ &\quad\vdots \end{aligned} \qquad (2.2.2)$$
$$\underline{1}\ .$$

Na sliki 2.2.3 je podan nekoliko prilagojen graf selektorske operacije, s katerim ponazorimo krmiljenje izvajanja sekvenčne operacije s slike 2.2.1.



Slika 2.2.3: Graf modela krmiljenja izvajanja sekvenčne operacije

K dosedanjim izvajanjem sekvenčnega krmilnega modela in sliki 2.2.3 pripomnimo, da bo sekvenčno krmiljenje operacij podrobneje obdelano v razdelku 2.5.

2.3. PARALELNA OPERACIJA

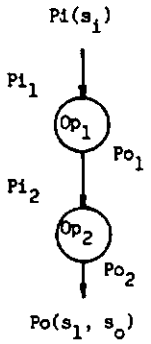
V dosedanjih izvajanjih sekvenčne operacije smo predpostavljali, da je začetna trditev $Pi_j(s_j)$ operacije Op_j pravilna šele, ko se izvedejo vse operacije $Op_1, Op_2, \ldots, Op_{j-1}$. Pri izpeljavi paralelne operacije pa sprostimo ta pogoj.

Izhajajmo iz grafa sekvenčne operacije na sliki 2.3.1.

Logična pravila za takšno sekvenčno operacijo so:

$$\begin{aligned} &Pi(s_i) \longrightarrow Pi_1(s_i) \\ &Pi(s_1) \wedge Po_1(s_1, s_2) \longrightarrow Pi_2(s_2) \\ &Pi(s_1) \wedge Po_1(s_1, s_2) \wedge Po_2(s_2, s_3) \longrightarrow Po(s_1, s_3)\ . \end{aligned} \qquad (2.3.1)$$

Sedaj pa predpostavimo, da velja:



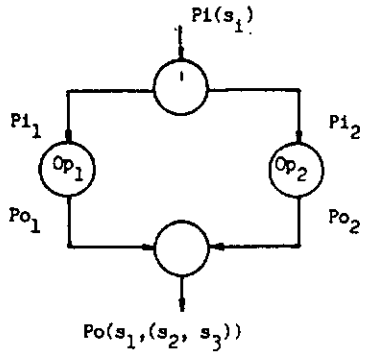
Slika 2.3.1: Graf dveh sekvenčno povezanih operacij

$$Pi(s_i) \longrightarrow Pi_1(s_i)$$
$$Pi(s_i) \longrightarrow Pi_2(s_i) \quad , \qquad (2.3.2)$$

da je $Pi_2(s_i)$ pravilen neodvisno od tega, ali se je Op_1 že izvršila ali ne. Za končni pogoj lahko tedaj zapišemo:

$$Pi_1(s_1) \wedge Po_1(s_1,s_2) \wedge Po_2(s_1,s_3) \longrightarrow$$
$$\longrightarrow Po(s_1,(s_2,s_3)). \qquad (2.3.3)$$

Z (s_2,s_3) smo označili konkatenacijo s_2 in s_3. Za ponazoritev paralelne operacije vpeljemo graf, ki ga podaja slika 2.3.2.



Slika 2.3.2: Graf paralelne operacije

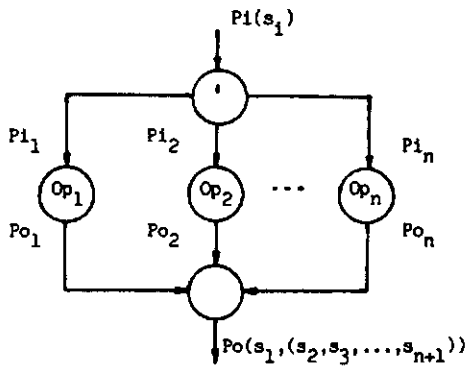
Izvedimo še posplošitev paralelne operacije na n paralelno povezanih operacij in zapišimo logična pravila:

$$Pi(s_i) \longrightarrow Pi_1(s_i)$$
$$Pi(s_i) \longrightarrow Pi_2(s_i) \qquad (2.3.4)$$
$$\vdots$$
$$Pi(s_i) \longrightarrow Pi_n(s_i)$$

$$Pi(s_1) \wedge Po_1(s_1,s_2) \wedge Po_2(s_1,s_3) \wedge \ldots \wedge Po_n(s_1,s_{n+1}) \longrightarrow$$
$$\longrightarrow Po(s_1, (s_2, s_3, \ldots, s_{n+1}) .$$

Pripadajoč graf podaja slika 2.3.3.

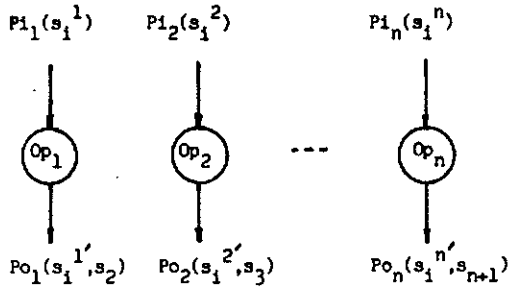
Pogoj za paralelno izvajanje operacij je v bistvu ta, da



Slika 2.3.3: Graf n paralelno povezanih operacij

operacija Op_j ne spremeni tistih komponent stanja s_i, ki so tudi začetne komponente za Op_1, Op_2,..., Op_{j-1}, Op_{j+1},..., Op_n. Enako velja tudi za vse ostale operacije. Operacija tedaj lahko spremeni samo tiste vhodne komponente v stanju s_i, ki so vhodne komponente samo te operacije, sicer se mehanizem paralelnega izvajanja poruši.

Takšno paralelno operacijo lahko tedaj razstavimo na komponente, med katerimi ni več nobene povezave. Slika 2.3.4 podaja tako razgrajeno paralelno operacijo.



Slika 2.3.4: Grafi operacij, ki se lahko izvajajo paralelno

2.4. PREOBLIKOVANJE SESTAVLJENIH OPERACIJ

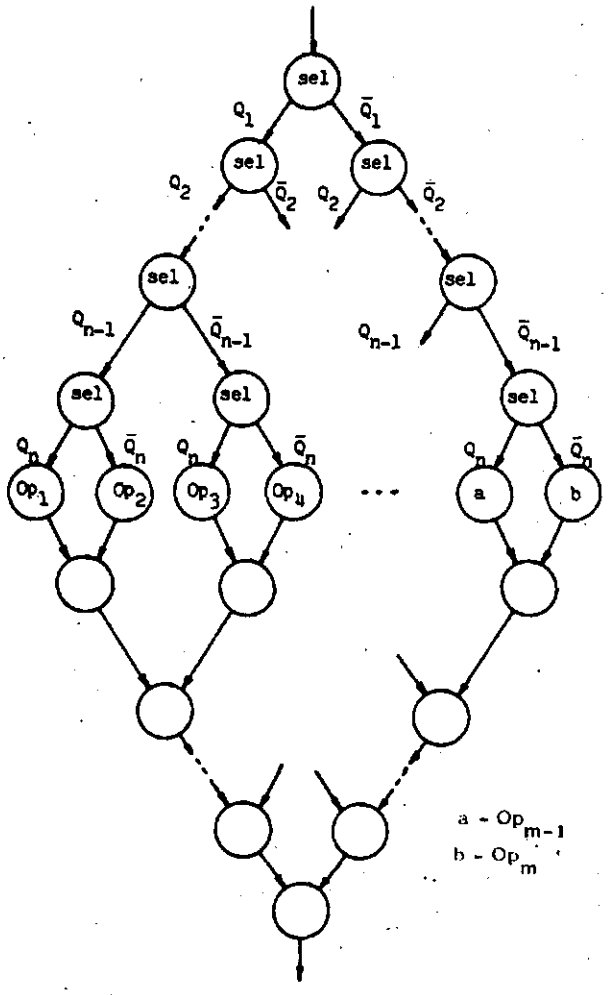
Razdelek podaja nekatere možnosti preoblikovanja sestavljenih operacij.

a) Preoblikovanje sestavljene selektorske operacije v selektorsko operacijo.

Selektorsko operacijo s slike 2.4.1 zapišimo v disjunktivni obliki.

$$(Q_1 \wedge Q_2 \wedge \ldots \wedge Q_{n-1} \wedge Q_n) \wedge Op_1 \vee$$
$$\vee (Q_1 \wedge Q_2 \wedge \ldots \wedge Q_{n-1} \wedge \bar{Q}_n) \wedge Op_2 \vee$$
$$\vee (Q_1 \wedge Q_2 \wedge \ldots \wedge \bar{Q}_{n-1} \wedge Q_n) \wedge Op_3 \vee$$
$$\vee (Q_1 \wedge Q_2 \wedge \ldots \wedge \bar{Q}_{n-1} \wedge \bar{Q}_n) \wedge Op_4 \vee \qquad (2.4.1)$$
$$\vdots$$

$$\vee(\bar{Q}_1\wedge\bar{Q}_2\wedge\ldots\wedge\bar{Q}_{n-1}\wedge Q_n)\wedge Op_{m-1}\vee$$
$$\vee(\bar{Q}_1\wedge\bar{Q}_2\wedge\ldots\wedge\bar{Q}_{n-1}\wedge\bar{Q}_n)\wedge Op_m \qquad (2.4.1)$$



Slika 2.4.1: Sestavljena selektorska operacija

Konjunkcije izjav Q_i, i = 1, 2, ..., n v (2.4.1) poimenujmo s $P_1, P_2, \ldots, P_m$, $m=2^n$ in dobimo:

$$P_1\wedge Op_1\vee P_2\wedge Op_2\vee P_3\wedge Op_3\vee P_4\wedge Op_4\vee \ldots$$
$$\ldots\vee P_{m-1}\wedge Op_{m-1}\vee P_m\wedge Op_m. \qquad (2.4.2)$$

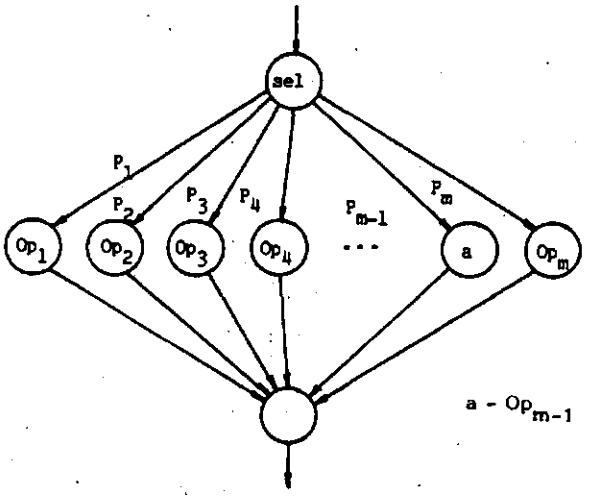
Z izrazom (2.4.2) pa lahko opišemo tudi selektorsko operacijo, katere graf podaja slika 2.4.2.

b) Preoblikovanje paralelne selektorske operacije v paralelno povezane selektorske operacije

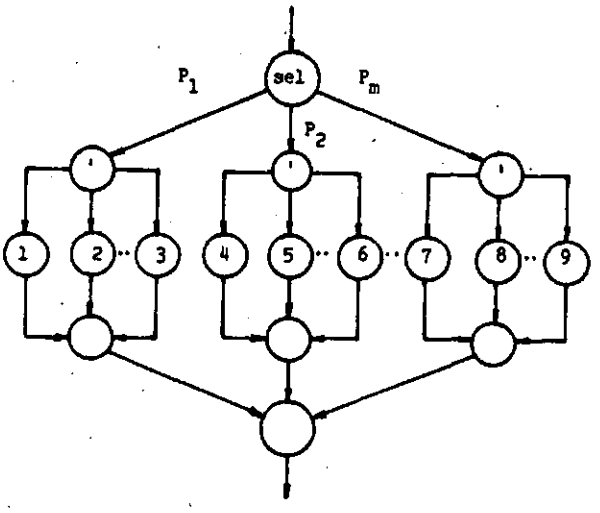
Sliko 2.4.3 opišemo s poenostavljeno selektorsko operacijo:

sel (P_i):

$$P_1 \longrightarrow (Op_{11}, Op_{12}, \ldots, Op_{1n})$$
$$P_2 \longrightarrow (Op_{21}, Op_{22}, \ldots, Op_{2n}) \qquad (2.4.3)$$
$$\vdots$$
$$P_m \longrightarrow (Op_{m1}, Op_{m2}, \ldots, Op_{mn})$$



Slika 2.4.2: Graf selektorske operacije za izraz 2.4.2



1 - Op_{11}	4 - Op_{21}	7 - Op_{m1}
2 - Op_{12}	5 - Op_{22}	8 - Op_{m2}
3 - Op_{1n}	6 - Op_{2n}	9 - Op_{mn}

Slika 2.4.3: Graf paralelne selektorske operacije

in preoblikujemo v disjunktivno obliko:

$$P_1\wedge(Op_{11}, Op_{12}, \ldots, Op_{1n})\vee P_2\wedge(Op_{21}, Op_{22}, \ldots, Op_{2n})\vee\ldots$$
$$\ldots\vee P_m\wedge(Op_{m1}, Op_{m2}, \ldots, Op_{mn}). \qquad (2.4.4)$$

Izraz (2.4.4) zapišemo po komponentah.

$$P_1\wedge Op_{11}\vee P_2\wedge Op_{21}\vee\ldots\vee P_m\wedge Op_{m1}$$
$$P_1\wedge Op_{12}\vee P_2\wedge Op_{22}\vee\ldots\vee P_m\wedge Op_{m2} \qquad (2.4.5)$$
$$\vdots$$

$$\vdots$$
$$P_1 \wedge Op_{1n} \vee P_2 \wedge Op_{2n} \vee \ldots \vee P_m \wedge Op_{mn}$$

V izraze (2.4.5) pa lahko preoblikujemo tudi naslednje selektorske operacije:

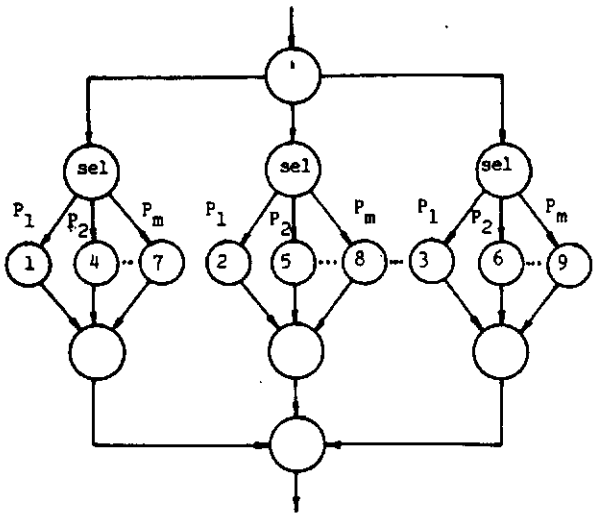
sel (P_i):
- $P_1 \longrightarrow Op_{11}$
- $P_2 \longrightarrow Op_{21}$
- $\vdots$
- $P_m \longrightarrow Op_{m1}$

sel (P_i):
- $P_1 \longrightarrow Op_{12}$
- $P_2 \longrightarrow Op_{22}$
- $\vdots$
- $P_m \longrightarrow Op_{m2}$

$\vdots$

sel (P_i): (2.4.6)
- $P_1 \longrightarrow Op_{1n}$
- $P_2 \longrightarrow Op_{2n}$
- $\vdots$
- $P_m \longrightarrow Op_{mn}$.

Graf selektorskih operacij (2.4.6) je podan na sliki 2.4.4, kot paralelno povezane selektorske operacije.



Opomba: koda operacij je na sliki 2.4.3.

Slika 2.4.4: Graf paralelno povezanih selektorskih operacij

c) Preoblikovanje paralelne selektorske operacije, kadar se operacije ponavljajo

Primerov za ponavljanje operacij v paralelni selektorski operaciji je veliko. Omenimo samo mikroprogramirano krmilno enoto, asociativni pomnilnik, razne registerske strukture, ki omogočajo paralelen dostop do podatkov, procesorje z množico funkcijskih enot, itd.

Ponazoritev takšnih sklopov s paralelno selektorsko operacijo ima svojo težo, saj jih lahko tako na višjih abstraktnih nivojih snovanja ponazorimo v koncentriranem zapisu, ki ga postopoma razgrajujemo z napredovanjem pri izgradnji modela.

Preoblikovanje paralelne selektorske operacije z lastnostjo ponavljanja operacij ponazorimo z zgledom:

sel (P_i):
- $P_0 \longrightarrow A, 1, a$
- $P_1 \longrightarrow A, 1, b$
- $P_2 \longrightarrow A, 1, c$
- $P_4 \longrightarrow A, 2, a$
- $P_5 \longrightarrow A, 2, b$
- $P_6 \longrightarrow A, 2, c$
- $P_8 \longrightarrow A, 3, a$
- $P_9 \longrightarrow A, 3, b$
- $P_{10} \longrightarrow A, 3, c$
- $P_{16} \longrightarrow B, 1, a$
- $P_{17} \longrightarrow B, 1, b$
- $P_{18} \longrightarrow B, 1, c$
- $P_{20} \longrightarrow B, 2, a$
- $P_{21} \longrightarrow B, 2, b$
- $P_{22} \longrightarrow B, 2, c$
- $P_{24} \longrightarrow B, 3, a$
- $P_{25} \longrightarrow B, 3, b$
- $P_{26} \longrightarrow B, 3, c$

(2.4.7)

P_i	Q_4	Q_3	Q_2	Q_1	Q_0
P_0	$\bar{Q}_4$	$\bar{Q}_3$	$\bar{Q}_2$	$\bar{Q}_1$	$\bar{Q}_0$
P_1	$\bar{Q}_4$	$\bar{Q}_3$	$\bar{Q}_2$	$\bar{Q}_1$	Q_0
P_2	$\bar{Q}_4$	$\bar{Q}_3$	$\bar{Q}_2$	Q_1	$\bar{Q}_0$
P_4	$\bar{Q}_4$	$\bar{Q}_3$	Q_2	$\bar{Q}_1$	$\bar{Q}_0$
P_5	$\bar{Q}_4$	$\bar{Q}_3$	Q_2	$\bar{Q}_1$	Q_0
P_6	$\bar{Q}_4$	$\bar{Q}_3$	Q_2	Q_1	$\bar{Q}_0$
P_8	$\bar{Q}_4$	Q_3	$\bar{Q}_2$	$\bar{Q}_1$	$\bar{Q}_0$
P_9	$\bar{Q}_4$	Q_3	$\bar{Q}_2$	$\bar{Q}_1$	Q_0
P_{10}	$\bar{Q}_4$	Q_3	$\bar{Q}_2$	Q_1	$\bar{Q}_0$
P_{16}	Q_4	$\bar{Q}_3$	$\bar{Q}_2$	$\bar{Q}_1$	$\bar{Q}_0$
P_{17}	Q_4	$\bar{Q}_3$	$\bar{Q}_2$	$\bar{Q}_1$	Q_0
P_{18}	Q_4	$\bar{Q}_3$	$\bar{Q}_2$	Q_1	$\bar{Q}_0$
P_{20}	Q_4	$\bar{Q}_3$	Q_2	$\bar{Q}_1$	$\bar{Q}_0$
P_{21}	Q_4	$\bar{Q}_3$	Q_2	$\bar{Q}_1$	Q_0
P_{22}	Q_4	$\bar{Q}_3$	Q_2	Q_1	$\bar{Q}_0$
P_{24}	Q_4	Q_3	$\bar{Q}_2$	$\bar{Q}_1$	$\bar{Q}_0$
P_{25}	Q_4	Q_3	$\bar{Q}_2$	$\bar{Q}_1$	Q_0
P_{26}	Q_4	Q_3	$\bar{Q}_2$	Q_1	$\bar{Q}_0$

(2.4.8)

(2.4.8) je tabela izjav, ki jih priredimo izjavam P_i. Paralelno selektorsko operacijo (2.4.7) lahko sedaj nadomestimo s tremi selektorskimi operacijami.

sel (Q_4):
- $\bar{Q}_4 \longrightarrow A$
- $Q_4 \longrightarrow B$

sel (Q_3, Q_2):
- $\bar{Q}_3 \wedge \bar{Q}_2 \longrightarrow 1$
- $\bar{Q}_3 \wedge Q_2 \longrightarrow 2$
- $Q_3 \wedge \bar{Q}_2 \longrightarrow 3$

sel (Q_1,Q_0):
- $\bar{Q}_1 \wedge \bar{Q}_0 \longrightarrow a$
- $\bar{Q}_1 \wedge Q_0 \longrightarrow b$
- $Q_1 \wedge \bar{Q}_0 \longrightarrow c$ (2.4.9)

(2.4.9) lahko glede na točko b) ponazorimo v grafu kot tri paralelno povezane selektorske operacije.

2.5. MODEL SEKVENČNEGA STROJA

Izhajamo iz implikacij:

$$\wedge p\,(I, Q) \longrightarrow Q$$
$$\wedge p\,(I, Q) \longrightarrow Z\ , \qquad (2.5.1)$$

kjer smo z I ponazorili niz izjav (i_{m-1}, i_{m-2},..., i_0) in z Q niz izjav (q_{n-1}, q_{n-2},..., q_0). Z $\wedge(I, Q)$ označimo vse možne konjunkcije sestavljenega niza (I, Q),

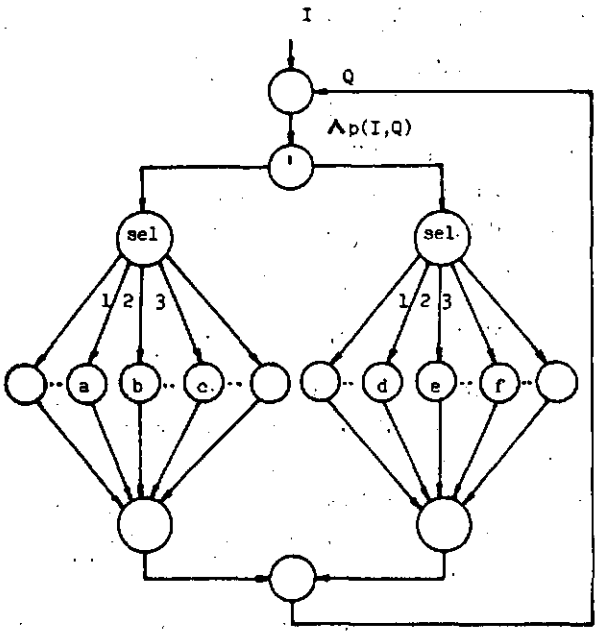
$z \bigwedge p(I, Q)$ pa izbor poljubnega števila konjunkcij iz $\bigwedge(I, Q)$, Z pa naj bo znak za niz $(z_{r-1}, z_{r-2}, \ldots, z_0)$.

(2.5.1) lahko ponazorimo s paralelno selektorsko operacijo:

$$
\begin{aligned}
&\text{sel } (P_i):\\
&\quad \vdots\\
&\quad P_i \longrightarrow Z_i, P_{i+1}\\
&\quad P_{i+1} \longrightarrow Z_{i+1}, P_{i+2}\\
&\quad \vdots\\
&\quad P_{i+v} \longrightarrow Z_{i+v}, P_{i+v+1}\\
&\quad \vdots
\end{aligned}
\qquad (2.5.2)
$$

kjer smo s ... , P_i, ... označili konjunkcije iz tabele $\bigwedge p(I,Q)$. Pri tem v splošnem ne zahtevamo enoličnosti prireditev ... Z_i ..., niti desne strani ... P_{i+1} ... v (2.5.2), oz. isti Z_i in P_i se lahko na desni pojavijo večkrat, medtem pa mora biti leva stran (2.5.2) enolična.

(2.5.2) ponazorimo z grafom paralelne selektorske operacije na sliki 2.5.1.



a - Z_i d - Q_{i+1} 1 - P_i
b - Z_{i+1} e - Q_{i+2} 2 - P_{i+1}
c - Z_{i+v} f - Q_{i+v+1} 3 - P_{i+v}

Slika 2.5.1: Graf vase zaključene paralelne selektorske operacije

Tako definiramo selektorsko operacijo lahko imenujemo model Mealyjevega stroja, če napravimo primerjavo med (2.5.1) in

$$
\begin{aligned}
f &: I \times Q \longrightarrow Q\\
g &: I \times Q \longrightarrow Z,
\end{aligned}
\qquad (2.5.3)
$$

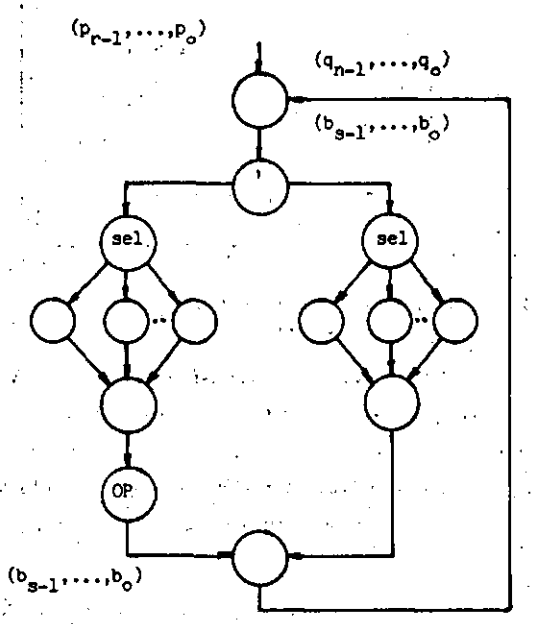
ter so I - vhodi, Q - stanja, Z - izhodi in f in g preslikavi.

Pravkar opisani model nekoliko dopolnimo, tako da ga bolj približamo obliki s kakršno imamo opravka v praksi snovanja strojne opreme. Za ta namen najprej definirajmo rep in glavo niza $I = (i_{m-1}, i_{m-2}, \ldots, i_j, i_{j-1}, \ldots, i_0)$ ter poimenujmo glavo niza I s P in rep niza I z B in ju označimo takole:

$$
\begin{aligned}
P &= (p_{r-1}, p_{r-2}, \ldots, p_0)\\
B &= (b_{s-1}, b_{s-2}, \ldots, b_0),
\end{aligned}
\qquad (2.5.4)
$$

kjer so $r-1 = m-1$, $r-2 = m-2$, ... in $s-1 = j-1$, $s-2 = j-2$, ... Niz P imenujmo zunanje vhodne izjave, niz B pa notranje vhodne izjave.

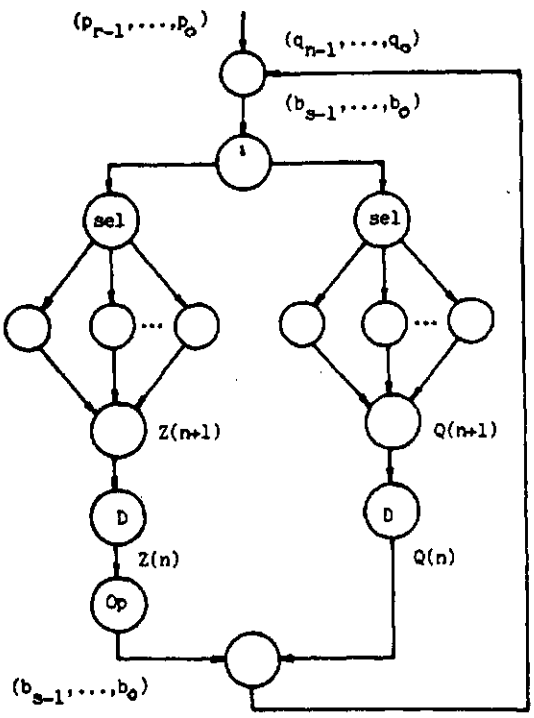
Sedaj pa narišimo nekoliko modificiran graf s slike 2.5.1 na sliki 2.5.2.



Slika 2.5.2: Modificiran graf paralelne selektorske operacije s slike 2.5.1

V sliko (2.5.2) smo vgradili operacijsko enoto OP, ki izvaja operacije določene z Z in kot rezultat daje izjave B o izvedenih operacijah. Podatkovni del operacijske enote nas zaenkrat ne zanima. $(b_{s-1}, b_{s-2}, \ldots, b_0)$ imenujemo vejitvene pogoje, ker omogočajo izvajanje vejitev v grafih operacij, ki jih modeliramo na takšnem modelu.

Sedaj pa vgradimo v naš model še mehanizem, ki omogoča časovno prekrivanje operacij med krmilno in operacijsko strukturo modela, kadar so izjave (b_{s-1}, b_{s-2}, ..., b_0) neaktivne. V ta namen oblikujmo graf našega modela po sliki 2.5.3.



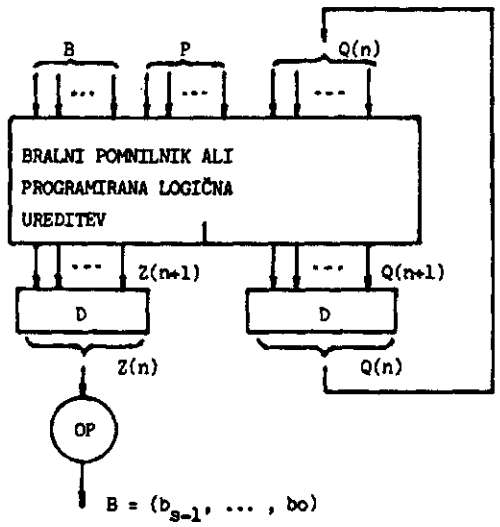
Slika 2.5.3: Model izvajanja operacij s časovnim prekrivanjem

Na sliki 2.5.3 imenujmo D zakasnilni element, ki vhodne izjave prenese na izhod s časovno zakasnitvijo, katere dolžina naj bo zaenkrat določena z zunanjimi pogoji. K grafu, na sliki 2.5.3 narišimo še pripadajočo blokovno shemo, kakršne smo pri snovanju bolj vajeni, na sliki 2.5.4.

Preden nadaljujemo z izgradnjo krmilnega modela definirajmo še modela programirane logične ureditve in bralnega pomnilnika. V ta namen izhajajmo iz poenostavljene selektorske operacije:

$$D_i = A_{m-1} \wedge D_{m-1} \vee A_{m-2} \wedge D_{m-2} \vee \ldots \ldots \vee A_0 \wedge D_0 \quad , \qquad (2.5.5)$$

kjer so A_i, i=0, 1,..., m-1 enolično prirejene vsem možnim konjunkcijam nad nizem $(a_{n-1}, a_{n-2}, \ldots, a_0)$ in D_i poljubno izbrane - v splošnem ne enolično - iz tabele izjav:



Slika 2.5.4: Blokovna shema modela sekvenčnega stroja z operacijsko enoto

$$\left.\begin{array}{l} \bar{d}_{t-1}, \ldots\ldots, \bar{d}_1, \bar{d}_0 \\ \bar{d}_{t-1}, \ldots\ldots, \bar{d}_1, d_0 \\ \bar{d}_{t-1}, \ldots\ldots, d_1, \bar{d}_0 \\ \bar{d}_{t-1}, \ldots\ldots, d_1, d_0 \\ \vdots \\ d_{t-1}, \ldots\ldots, d_1, d_0 \end{array}\right\} 2^t \qquad (2.5.6)$$

Če "preberemo" iz bralnega pomnilnika z "adrese" A_i "element" D_i lahko to zapišemo takole:

$$D_i = 0 \vee 0 \vee \ldots \vee A_i \wedge D_i \vee 0 \vee \ldots \vee 0 \qquad (2.5.7)$$
$$= A_i \wedge D_i = 1 \wedge D_i = (d_{t-1}, \ldots, d_1, d_0)_i \quad .$$

Model programirane logične ureditve je v bistvu identičen s to razliko, da (2.5.5) zapišemo po komponentah in v zapisu izpustimo vse tiste konjunkcije, ki imajo zaradi $d_i = 0$ vrednost 0. V disjunktivne zapise vstavimo tedaj samo tiste komponente d_j, ki imajo vrednost 1, glede na tabelo (2.5.6).

S stališča uporabe lahko tedaj rečemo, da v splošnem ni potrebno razlikovati med bralnim pomnilnikom in programirano logično ureditvijo, saj lahko oba gradnika po potrebi interpretiramo kot bralni pomnilnik oz. programiramo logično ureditev.

Sedaj pa nadaljujmo z izgradnjo modela mikroprogramiranega krmilnika. V ta namen definirajmo selektorsko operacijo, s katero bomo izdelali začetni približek k sekvencerju mikroprogramiranega krmilnika. Iz nizev:

$$\begin{array}{l} P = (p_{n-1}, p_{n-2}, \ldots, p_0) \\ B = (b_{s-1}, b_{s-2}, \ldots, b_0) \\ Q = (q_{n-1}, q_{n-2}, \ldots, q_0) \end{array} \qquad (2.5.8)$$

z operacijami glava, rep, delni niz, konkatenacija, glava delnega niza, rep delnega niza, konkatenacija delnega niza tvorimo nize enakih dolžin in jih poimenujmo z $A = (a_{n-1}, a_{n-2}, \ldots, a_0)$. Oglejmo si nekaj primerov tako konstruiranih nizev:

$$\begin{matrix} q_{n-1}, \ldots\ldots, q_1, q_0 \\ q_{n-1}, \ldots\ldots, b_{s-1}, b_0 \\ p_{n-1}, \ldots\ldots, p_1, p_0 \end{matrix} \qquad \begin{matrix} q_{n-1}, \ldots, b_i, b_j, b_0, b_7, \\ i,j \in \{0, 1, \ldots, s-1\} \end{matrix} \qquad (2.5.9)$$

Pri tem bomo smatrali, da lahko v splošnem vsi elementi tako definiranih nizev zavzamejo vrednosti q_a ali $\bar{q}_a$ b_e ali $\bar{b}_e$ in p_c ali $\bar{p}_c$ in a in $c \in \{0, 1, \ldots, n-1\}$ ter $e \in \{0, 1, \ldots, s-1\}$.

Sedaj pa definirajmo selektorsko operacijo s katero izbiramo nize, ki smo jih konstruirali po zgornjem pravilu.

sel (BR_j):

$BR_1 \longrightarrow$ niz 1
$BR_2 \longrightarrow$ niz 2 (2.5.10)
⋮
$BR_w \longrightarrow$ niz $_w$

Za zgled predpostavimo, da ima niz j takšno obliko:

$q_{n-1}, q_{n-2}, \ldots, q_2, b_1, b_0$.

Z BR_j tedaj izberemo enega izmed nizev

$$\begin{matrix} \bar{q}_{n-1}, \bar{q}_{n-2}, \ldots, \bar{q}_2, \bar{b}_1, \bar{b}_0 \\ \bar{b}_1, b_0 \\ b_1, \bar{b}_0 \\ \vdots \\ b_1, b_0 \end{matrix}$$

$q_{n-1}, q_{n-2}, \ldots, q_2, b_1, b_0$,

odvisno pač od trenutnih vrednosti, ki jih imajo izjave

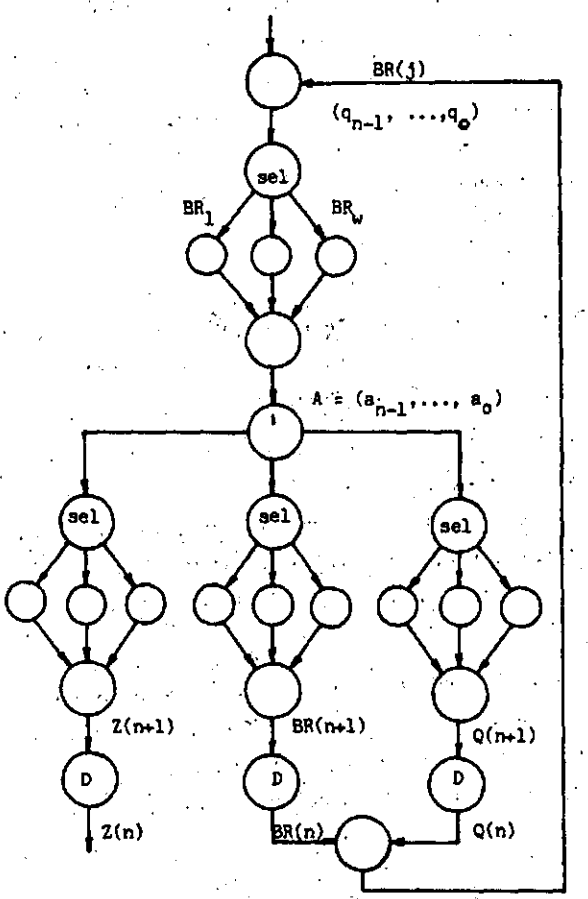
$q_{n-1}, q_{n-2}, \ldots, q_2, b_1, b_0$.

Za selektorsko operacijo (2.5.10) bomo smatrali, da opravlja nalogo sekvencerja adres v začetnem približku k modelu mikroprogramiranega krmilnika, (sl. 2.5.5).

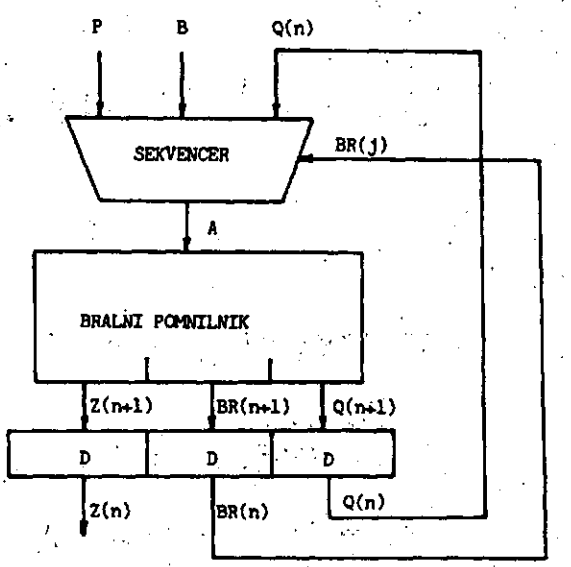
S postopno širitvijo začetnega modela sekvencerja ga lahko opremimo z mehanizmi za strežbo podprogramov, pasti itd., vendar lahko še tako kompleksen sekvencer vedno prevedemo na selektorsko operacijo. Na sliki 2.5.6 je podana blokovna shema krmilnika izdelana na podlagi grafa na sliki 2.5.5.

Omenimo še, čeprav s tem prehajamo zastavljeni okvir tega članka, da se krmilna struktura z izgradnjo mikroprogramskega krmilnika ne konča, ampak v splošnem prehaja v operacijsko enoto, ki smo jo doslej opazovali

$P = (p_{n-1}, \ldots, p_0)$, $B = (b_{s-1}, \ldots, b_0)$



Slika 2.5.5: Začetni približek k mikroprogramiranemu krmilniku



Slika 2.5.6: Blokovna shema začetnega približka mikroprogramiranega krmilnika

le kot koncentrirani gradnik strukture. Tudi izgradnjo

modela operacijske enote lahko v splošnem pričnemo s selektorsko operacijo, ki jo razgrajujemo toliko časa, dokler ne pridemo do gradnikov, ki jih s stališča izvajanja operacij lahko pojmujemo kot koncentrirane gradnike. Te gradnike obravnavamo tedaj kot elemente, ki pričnejo izvajati izbrano operacijo z nastopom izvornih operandov oz. v trenutku, ko so izpolnjeni pogoji začetne trditve Pi(s) v (1.1).

3. ZAKLJUČEK

Podani so postopki snovanja z logičnimi modeli računalniških struktur, s katerimi je možen postopen prehod na logično realizacijo izbranih struktur.

Stanja in operacije je možno postopoma razgrajevati v vedno večje detalje, postopek se konča, ko najdemo za sestavljene operacije skupek ustrezno povezanih mikroelektronskih gradnikov, katerih funkcije ustrezajo sestavljenim operacijam, s katerimi smo modelirali izhodiščno strukturo.

Pri predlaganih postopkih snovanja moramo razen notranjih značilnosti strukture upoštevati tudi zunanje parametre - hitrost, cena, zanesljivost, ..., ki v snovanje vnašajo komponento realnega okolja.

Izdelani so modeli sestavljenih operacij, transformacije med njimi in izgrajeni modeli nekaterih realnih logičnih in računalniških struktur.

4. LITERATURA

/1/ C. B. JONES: Software Design: A Rigorous Approach, Prentice-Hall International 1980.

/2/ M. GERKEŠ: Metodologija snovanja računalniških struktur in sistemov z upoštevanjem trendov v razvoju tehnologije in konceptnih rešitev, disertacija, Univerza Edvarda Kardelja v Ljubljani, Fakulteta za elektrotehniko, Ljubljana 1984.

/3/ J. VIRANT: Preklopne funkcije, strukture in sistemi, Univerza Edvarda Kardelja v Ljubljani, Fakulteta za elektrotehniko, Ljubljana 1983.

/4/ M. GERKEŠ, M. PERNEK, M. PAGLAVEC: Aplikacija bipolarnega mikroprocesorja, Poročilo o delu za leto 1984, URP/RP: Računalniška oprema 03-2570, RSS, PORS 3, Visoka tehniška šola, Maribor, 1984.