Nové prístupy k syntéze logických obvodov

Anna Príkopová
Fedor Kállay

V príspevku je opísaná metóda zjednodušenej realizácie logického obvodu, ktorá je založená na využití rozšírených operačných možností programovateľných systémov. Prínosom tejto metódy je možnosť kontroly automatu v nepredvídaných stavoch.

Úvod do problematiky

Implementácia logických obvodov v súčasnosti patrí k štandardným úlohám riadiacej techniky. Štandardnou je najmä ich realizácia, prakticky výlučne formou programovateľných procesorových systémov v najrôznejších podobách (ako inteligentné relé, špecializované modulárne systémy), na rozdiel od pôvodných realizácií pomocou diskrétnych prvkov. Hoci zmena technickej bázy predstavuje významný kvalitatívny posun aj v procese syntézy, zreteľne sa ukazujú aj mnohé spoločné črty. V záujme porovnania uvedieme niektoré základné črty tvorby a nasadzovania logických obvodov z diskrétnych prvkov. Pri ich formálnom opise sa používajú výlučne dvojhodnotové algebry, najmä také, ktoré umožňujú konštrukciu štruktúrnych foriem. Tieto hľadiská sú celkom prirodzené, pretože stavebné elementy principiálne pracujú ako diskrétne a dvojhodnotové, takže realizácia výsledného obvodu je možná po určení ich vhodného zapojenia, t. j. štruktúry. Uvedený spôsob návrhu logických obvodov sa ukázal natoľko efektívny, že sa touto metódou postupuje aj v súčasnosti, v zmenených podmienkach realizácie programovateľnými systémami. Dôkazom tohto tvrdenia sú vyššie programovacie jazyky, založené na tzv. priečkových diagramoch. Programy v priečkových diagramoch možno jednoznačne vnímať ako ekvivalenty reléových schém. Ich charakteristickým znakom je tvorba priechodných, resp. nepriechodných priečok z tzv. bitových inštrukcií (ekvivalentov kontaktov relé), ktoré tvoria základ štruktúry v tvare sériovo-paralelných dvojpólov. Tieto postupy umožňujú pomerne jednoduchú a intuitívnu tvorbu logických obvodov, či už v podobe kombinačných logických obvodov alebo sekvenčných obvodov, ktoré sú označované ako automaty.

Programovateľné systémy ponúkajú ďaleko rozsiahlejšie operačné možnosti než je tvorba logických reťazcov. Tieto možnosti sa dajú využiť na zracionalizovanie práce projektanta či programátora, a to najmä v prípade zložitých vzťahov, kde zlyháva intuícia a rutina.

1. Zložitý logický obvod, spôsoby jeho opisu, etapy návrhu

Bez ujmy na všeobecnosti úvah budeme pod pojmom zložitý logický obvod rozumieť sekvenčný automat A, ktorý je v termínoch abstraktnej teórie definovaný ako pätica

$$A = (X, Y, S, F, \Phi) \tag{1}$$

kde X, Y, S sú konečné, neprázdne množiny symbolov

vstupných $X = \{X_1, X_2, \ldots\}$
výstupných $Y = \{Y_1, Y_2, \ldots\}$
vnútorných stavov $S = \{S_1, S_2, \ldots\}$ (2)

F, Φ sú funkcie

výstupná $Y(t) = F\left[S(t), X(t)\right]$ (3)

stavová $S(t+1) = \Phi\left[S(t), X(t)\right]$ (4)

Pre praktické aplikácie býva explicitne vyznačený počiatočný stav, do ktorého musí byť automat nastavený vždy pred začiatkom činnosti. Ku konkretizácii výstupnej funkcie automatu (3) sa môžeme dostať z používateľského zadania, ktoré formuluje priradenie postupnosti výstupov k zadanej postupnosti vstupov. Prirodzeným nedostatkom používateľského zadania býva jeho neúplnosť, a predovšetkým rozpornosť. Na druhej strane súčasťou tohto zadania bývajú dodatočné informácie – vzhľadom na očakávané diskrétne dvojhodnotové procesy sú vstupy X_i a výstupy Y_j zadané v zložkovom tvare formou dvojhodnotových signálov.

Spomínaná rozpornosť používateľského zadania, ktorú možno charakterizovať ako nejednoznačnosť priradenia vstupných a výstupných symbolov, tvorí jeden zo základných problémov syntézy. V rozpornej forme sa totiž automat nedá realizovať. Tento nedostatok sa principiálne odstraňuje zavedením stavových premenných, ktorých generovanie zaisťuje stavová funkcia (4). Zdôraznime, že podmienky pre konštrukciu tejto funkcie nebývajú explicitne definované. Buď sú celkom ponechané na projektantovi, alebo sú ovplyvňované implicitne dodatočnými požiadavkami (podmienka dobrej činnosti a pod.). Realizáciu stavového zobrazenia možno na úrovni abstraktnej syntézy uskutočniť viacerými metódami, (napr. Ginsburgova, Ajzermanova atď.).

V očakávaní praktickej realizácie dvojhodnotovými elementmi, rozpisujeme vektorové rovnice (3) a (4) v zložkovom tvare

$$\begin{aligned} y_m(t) &= f_m\left[s_j(t), s_{j-1}(t), \ldots s_1(t), x_k(t), x_{k-1}(t), \ldots x_1(t)\right] \\ y_{m-1}(t) &= f_{m-1}\left[s_j(t), s_{j-1}(t), \ldots s_1(t), x_k(t), x_{k-1}(t), \ldots x_1(t)\right] \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ y_1(t) &= f_1\left[s_j(t), s_{j-1}(t), \ldots s_1(t), x_k(t), x_{k-1}(t), \ldots x_1(t)\right] \end{aligned} \tag{5}$$

$$\begin{aligned} s_j(t+1) &= \varphi_j\left[s_j(t), s_{j-1}(t), \ldots s_1(t), x_k(t), x_{k-1}(t), \ldots x_1(t)\right] \\ s_{j-1}(t+1) &= \varphi_{j-1}\left[s_j(t), s_{j-1}(t), \ldots s_1(t), x_k(t), x_{k-1}(t), \ldots x_1(t)\right] \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ s_1(t+1) &= \varphi_1\left[s_j(t), s_{j-1}(t), \ldots s_1(t), x_k(t), x_{k-1}(t), \ldots x_1(t)\right] \end{aligned} \tag{6}$$

kde $X_p = (x_k, x_{k-1}, \ldots x_1)$ je vektor vstupných premenných, ktorý pozostáva z „k“ dvojhodnotových zložiek a $S_q = (s_j, s_{j-1}, \ldots S_1)$ je vektor stavových premenných, ktorý pozostáva z „j“ dvojhodnotových zložiek, pričom $t \in N$ je diskrétny čas.

Rozpis vektorových rovníc do zložkového tvaru umožňuje vhodnú reprezentáciu automatu analógiou pravdivostnej tabuľky, ktorá dovoľuje bezprostredný zápis pracovných funkcií, resp. štruktúrnych vzťahov. Na obr. 1 je znázornená tabuľka relatívne jednoduchého automatu, pre ktorý množina vstupných symbolov je $X = \{X_1, X_2, X_3\}$. Každý z týchto symbolov je zapísaný v zložkovom tvare pomocou dvoch dvojhodnotových zložiek $X_p = (x_2, x_1)$, t. j. vstupy reprezentujú dva dvojhodnotové signály. Analogicky pre výstupy $Y = \{Y_1, Y_2, Y_3\}$, kde $Y_q = (y_2, y_1)$ a pre stavové symboly $S = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$ je definovaných sedem dvojhodnotových stavových zložiek $S_r = (s_7, s_6, s_5, s_4, s_3, s_2, s_1)$.

Pri použití Booleovej algebry možno z tejto tabuľky získať pracovné funkcie typu (5) a (6) v obvyklej polynomickej forme – úplnej, normálnej a disjunktnej forme (ÚNDF). Tieto funkcie obsahujú dve výstupné rovnice pre generovanie zložiek $y_2(t)$, $y_1(t)$ a sedem stavových rovníc pre generovanie zložiek $s_7(t+1)$, $s_6(t+1)$, $s_5(t+1)$, $s_4(t+1)$, $s_3(t+1)$, $s_2(t+1)$, $s_1(t+1)$. Takto získané polynomické formy sú štruktúrnymi vzťahmi, ktoré sú predpisom na vytvorenie dvoj-

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
t	ED(t)	$s_7(t)$	$s_6(t)$	$s_5(t)$	$s_4(t)$	$s_3(t)$	$s_2(t)$	$s_1(t)$	$x_2(t)$	$x_1(t)$	$s_7(t+1)$	$s_6(t+1)$	$s_5(t+1)$	$s_4(t+1)$	$s_3(t+1)$	$s_2(t+1)$	$s_1(t+1)$	$y_2(t)$	$y_1(t)$
0	28	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0
1	29	0	0	0	0	1	1	1	0	1	0	0	0	1	0	1	0	0	1
2	41	0	0	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	1
3	40	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0	1
4	168	0	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0	1
5	169	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	1
6	170	0	1	0	1	0	1	0	1	0	0	1	0	0	0	0	1	0	1
7	134	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	1
8	132	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0
9	30	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0	1	0
10	82	0	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	1	0
11	80	0	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0	1	0
12	336	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0	1	0
13	338	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	1	0
14	337	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0	1	1	0
15	261	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0
16	260	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0

Obr.1

pólov pri realizácii diskrétnymi prvkami, resp. na vytvorenie priečok pri programovom vyjadrení. Ako sa čitateľ môže presvedčiť, takto získaná štruktúra je dosť zložitá a tvoria ju dvojpóly typu π, s veľkým počtom paralelných vetiev sériových reťazcov. Bývajú preto obvyklé ďalšie etapy, ktoré zahŕňajú napr. minimalizáciu, aby sme získali prijateľnú formu na realizovanie.

2. Nové prístupy k štruktúrnej syntéze zložitých logických obvodov

V predchádzajúcej časti sme poukázali na zložitosť dvojpólov, ktoré realizujú logický obvod, získaný v procese štruktúrnej syntézy, a to pomocou pravdivostnej tabuľky a z nej vytvorenej polynomickej formy ÚNDF. Bezprostredným dôsledkom tejto skutočnosti je takmer úplná nereálnosť aplikácie výsledkov štruktúrnej syntézy pri realizácii zložitých logických obvodov z diskrétnych prvkov. Avšak aj pri programovej realizácii možno považovať získané priečky za neprimerane zložité. Až na celkom výnimočné prípady možno tvrdiť, že podstatné zjednodušenie nezískame ani obvyklou následnou etapou – bezprostrednou minimalizáciou ÚNDF. Jestvujú exaktné algebraické, ale aj empirické metódy predĺženia minimalizačného procesu, ktoré vedú k prijateľným výsledkom realizácie. Tieto postupy však treba charakterizovať ako umelé a náročné, ktorých dopad na funkčné vlastnosti získaného automatu v prípadoch poruchového vývoja nedokážeme odhadnúť. Naznačené problémy vedú k odmietaniu exaktného návrhu logických obvodov a k príklonu k intuitívnemu návrhu, ktorý poskytuje prijateľnú štruktúru.

Iným možným postupom je využívanie rozšírených operačných možností programovateľných systémov, ktoré môžu celkom prirodzeným spôsobom riešiť naznačenú stránku zložitosti návrhu. Jedna z možností sa opiera o odlišnú interpretáciu pravdivostnej tabuľky automatu. Jej prvá – zadávacia časť – totiž definuje usporiadanú n-ticu ($n = k + j$) argumentov funkcií v zložkovom tvare (5) a (6), t. j. stĺpce $s_j(t)$, $s_{j-1}(t)$,… $s_1(t)$, $x_k(t)$, $x_{k-1}(t)$,… $x_1(t)$. Každú takúto n-ticu v jednotlivých riadkoch možno interpretovať ako číslo v dvojkovom vyjadrení, ktorému prislúcha celočíselný ekvivalent vo vhodnej číselnej sústave, napr. dekadický $ED(t)$. Pre uvažovaný automat sú hodnoty dekadických ekvivalentov vyznačené v stĺpci 1 pravdivostnej tabuľky. Obsah tohto stĺpca možno označiť ako definičnú množinu dekadických ekvivalentov automatu ED_{def}. S použitím naznačenej číselnej transformácie možno každú dvojhodnotovú zložku z druhej, výsledkovej časti tabuľky, a to stĺpce $s_j(t+1)$, $s_{j-1}(t+1)$,… $s_1(t+1)$, $y_m(t)$, $y_{m+1}(t)$,… $y_1(t)$, vyjadriť vo forme zovšeobecnenej ÚNDF, ktorá v tomto vyjadrení nadobúda tvar disjunkcie dekadických ekvivalentov tých n-tíc, pre ktoré má príslušná zložka hodnotu 1. Pre jednotlivé zložky tak získame tzv. zložkové podmnožiny dekadických ekvivalentov ED_{zym}, resp. ED_{zsj}. Pre automat zadaný tabuľkou sú zovšeobecnené ÚNDF

$$y_2(t) = \cup(30, 80, 82, 261, 336, 337, 338) = \cup\{ED_{zy2}\}$$

$$y_1(t) = \cup(29, 40, 41, 134, 168, 168, 170) = \cup\{ED_{zy1}\}$$

$$s_7(t+1) = \cup(80, 261, 336, 337) = \cup\{ED_{zs7}\}$$

$$s_6(t+1) = \cup(40, 134, 168, 170) = \cup\{ED_{zs6}\}$$

$$s_5(t+1) = \cup(30, 80, 82, 336, 338) = \cup\{ED_{zs5}\}$$

$$s_4(t+1) = \cup(29, 40, 41, 168, 169) = \cup\{ED_{zs4}\}$$

$$s_3(t+1) = \cup(28, 30, 80, 82, 132, 260, 336, 338) = \cup\{ED_{zs3}\}$$

$$s_2(t+1) = \cup(28, 29, 40, 41, 132, 168, 169, 260) = \cup\{ED_{zs2}\}$$

$$s_1(t+1) = \cup(28, 132, 134, 170, 260, 261, 337) = \cup\{ED_{zs1}\}$$

Vzhľadom na logickú povahu zápisu tieto vzťahy možno čítať tak, že v ľubovoľnom pracovnom takte t (resp. $t+1$) prislúcha vybranej zložke, napr. $y_2(t)$ hodnota 1 vtedy a len vtedy, ak aktuálna zadávacia n-tica v takte t zodpovedá dekadickému ekvivalentu 30, alebo 80, alebo 82, 261, 336, 337, 338. Z hľadiska zostavenia programu pre procesorový systém ide o generovanie nenulovej hodnoty vybranej dvojhodnotovej zložky na základe rozhodnutia, či okamžitá zadávacia n-tica pochádza z jej zložkovej podmnožiny.

Naznačený formálny postup má niekoľko významných dôsledkov. V prvom rade ide o skrátenie a sprehľadnenie procesu realizácie. Aj takáto programová realizácia sa opiera o klasický spôsob syntézy, založený na tvorbe pravdivostnej tabuľky, ktorú možno chápať ako exaktný prepis používateľského zadania. Z pravdivostnej tabuľky pre jednotlivé zložky jej výsledkovej časti získame zovšeobecnené ÚNDF, ktoré definujú ich zložkové podmnožiny dekadických ekvivalentov. Týmto krokom je syntéza ukončená. Pri tvorbe programovej realizácie nehľadáme štruktúru minimalizovaných dvojpólov typu π z bitových premenných, ale realizujeme porovnávanie okamžitého dekadického ekvivalentu zadávacej n-tice s položkami zložkových podmnožín formou jednoduchého príkazu, napr.

$$y_2(t) = \begin{pmatrix} ED = 30\ OR\ ED = 80\ OR\ ED = 82\ OR\ ED = 261 \\ OR\ ED = 336\ OR\ ED = 337\ OR\ ED = 338 \end{pmatrix}$$

Medzi nesporné výhody naznačeného postupu patrí možnosť prípadného ošetrenia nekorektných, poruchových alebo nebezpečných situácií. Tieto sa pri činnosti prejavia tým, že pri uvedenom porovnávaní zistíme, že okamžitý dekadický ekvivalent zadávacej n-tice nepochádza z definičnej množiny dekadických ekvivalentov automatu. Takémuto stavu možno dodatočne predpísať špecifickú akciu.

Záver

V predloženom článku je zhodnotený klasický spôsob syntézy sekvenčného automatu najmä z hľadiska zložitosti, či už teoretického návrhu, alebo programovej realizácie pre procesorové riadiace systémy, a to v podmienkach veľkého počtu vstupných a výstupných zložiek. Fázu teoretického návrhu treba považovať za nevyhnutnú súčasť syntézy, ktorú nemožno obísť. Možno ju vnímať ako etapu prepisu intuitívneho používateľského zadania do formálnych výrazových prostriedkov, ktoré dovoľujú posúdenie jeho úplnosti a najmä bezospornosti a získanie podkladov na vyjadrenie pracovných funkcií. Fázu realizácie automatu modernými prostriedkami však možno upraviť účelovým využitím rozsiahlejších operačných možností programovateľných systémov, než iba tvorbou logických reťazcov. Ide predovšetkým o aritmetiku celých čísel vo vybraných sústavách (binárna, dekadická) s možnosťou jednoduchej zmeny základu. Vhodnou aplikáciou týchto možností sa dá získať odlišná a jednoduchšia štruktúra realizácie, a to bez nutnosti obvyklých krokov, ako je minimalizácia apod. Navyše takýto spôsob realizácie dáva projektantovi do rúk rozšírené možnosti, ako napr. kontrolu činnosti automatu aj v nepredvídaných či poruchových situáciách.

Ing. Anna Príkopová
doc. Ing. Fedor Kállay, PhD.

Katedra merania a automatizácie
Strojnícka fakulta, Žilinská univerzita
Veľký diel, 010 26 Žilina
Tel.: 041/513 28 00
e-mail: Fedor_Kallay@kma.utc.sk
Anna_Prikopova@kma.utc.sk