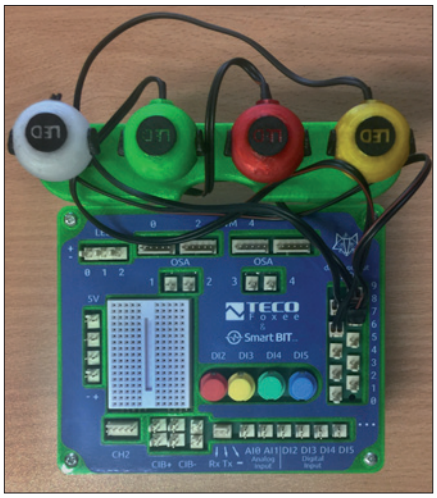
Inspiromat pro výuku a Tecomat: logika (nejenom) pro programátory (část 2)

(dokončení z minulého čísla)

Robot Foxee a jeho řídicí systém

Mobilní výukový robot, označovaný jako Foxee, vznikl v rámci robotické větve výukového programu EDUtec firmy Teco. Program byl založen před více než dvaceti lety pro podporu výuky automatizace na odborných školách. Výukový robot Foxee byl vyvinut firmou SmartBit a je dodáván firmou Teco (*obr. 1* v první části článku). Jeho (oddělitelnou) součástí je kompaktní řídicí systém – pracovně pojmenovaný kostka Foxee

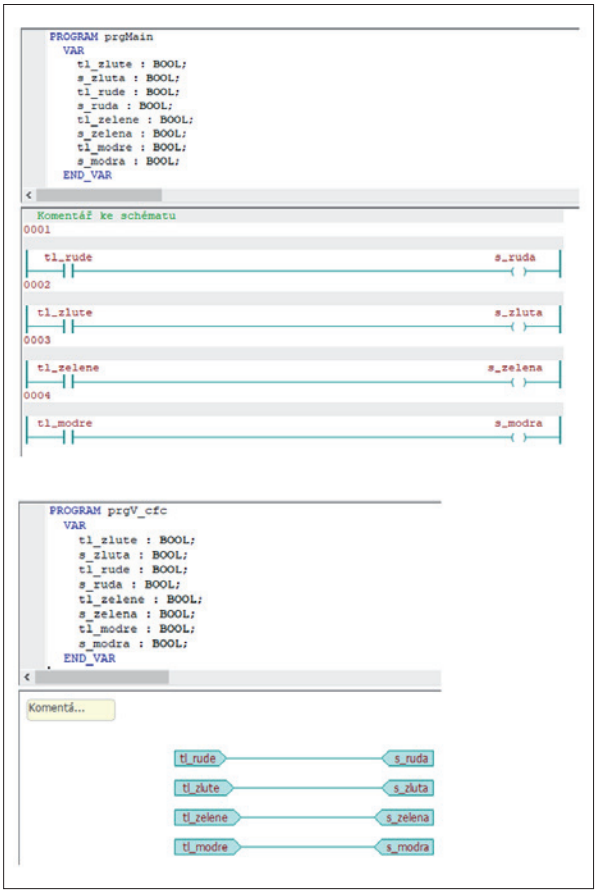


Obr. 4. Kostka Foxee s barevnými signálkami připravená pro řešení příkladů

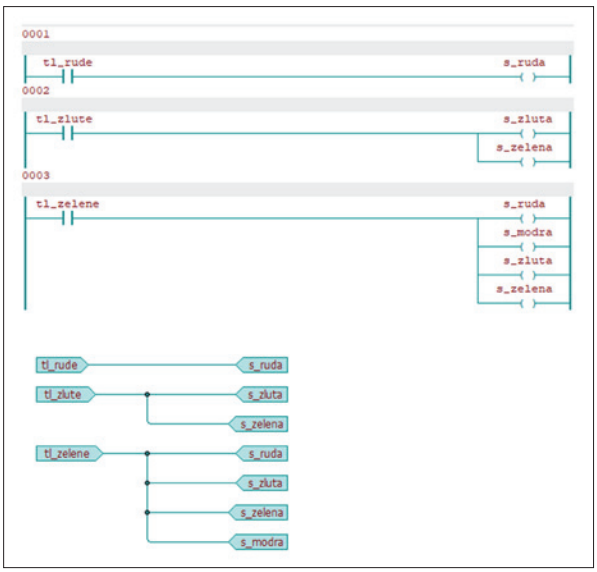
(*obr. 2* v první části článku). Může řídit i jiné mechanismy nebo fungovat samostatně. Tak je využíván pro příklady z počátku tohoto seriálu. Je v něm zabudován programovatelný automat Tecomat Foxtrot verze CP 1972 a je přizpůsoben pro komunikaci WiFi. Na panelu jsou umístěny konektory pro:

- dva analogové vstupy (A0, A1), které mohou být proudové 0 až 20 mA nebo napěťové v rozsahu ±0,3 V,
- čtyři dvouhodnotové vstupy (D2 až D5), zapojené paralelně s kontakty tlačítek (DI2 až DI5),
- deset dvouhodnotových výstupů (DO0 až DO9) 12 V,
- sériovou sběrnici CIB a TL2 pro rozšiřovací moduly vstupů a výstupů,
- komunikační porty CH1 (rozhraní RS-232) a CH2 (RS-232 nebo RS-485),
- až tři signální LED,
- až čtyři řízené.

Podrobnosti jsou uvedeny v [13] (viz seznam literatury v předchozí části). Vzhledem ke komunikačním možnostem systému Foxtrot



Obr. 5. Programy k příkladu 1 s deklaracemi proměnných – horní část obsahuje program v jazyce LD, pod ním je program v jazyce CFC (sledování)



Obr. 6. Grafické programy k příkladu 2 (sledování s kolizí adres přiřazení)

lze koordinovat činnost několika kostek Foxee, např. při řízení několika samostatných mechanismů nebo součástí složitějšího mechanismu (*obr. 3* v první části článku). Tak mohou studenti nenásilnou formou získat zkušenosti s distribuovaným řízením výrobních procesů, které je charakteristické pro koncepci průmyslu 4.0.

Pro příklady z úvodní části seriálu bude prozatím využívána jen minimální konfigurace: čtveřice barevných tlačítek a čtveřice barevných signálek, které lze ke kostce připevnit a připojit ke konektorům pro dvouhodnotové vstupy (*obr. 4*). Pro řešené příklady je účelné barevné signálky uspořádat v pořadí shodném s pořadím barev tlačítek.

Sledování a negace

Výklad začne od nejjednodušších příkladů úloh, kdy signálky kopírují stav tlačítek (logická funkce sledování) nebo svítí inverzně k nim (logická funkce negace – NOT).

Příklad 1. shodné barvy

Tlačítky ovládejte signálky shodné barvy – např. stiskem a uvolněním červeného tlačítka ovládejte svit červené signálky, žlutým tlačítkem ovládejte žlutou atd.

Deklarace proměnných

Nejprve je nutné deklarovat vstupní a výstupní proměnné. Všechny jsou dvouhodnotové, tedy typu BOOL. Jejich jména je možné zvolit libovolně podle zásad normy pro tvorbu identifikátorů – mohou obsahovat číslice, malá a velká písmena z anglické abecedy (bez háčků a čárek), znak „_“ (podtržítko), nesmí obsahovat me-

zery a musí začínat písmenem nebo podtržítkem. Pro tyto příklady byla zvolena jména co možná nejkratší tak, aby vystihovala význam proměnných. Proměnné musí být definovány dříve, než budou použity. Oba programy uvedené v *obr. 5* začínají blokem deklarací proměnných. V programech dalších příkladů již deklarace nebudou uváděny, ale budou předpokládány. Protože signálky mohou být připojeny ke konektorům libovolných binárních výstupů DO0 až DO9, bylo by třeba deklarace vždy volit podle této konkrétní konfigurace. V *obr. 5* jsou proto deklarace voleny tak, jako by šlo o vnitřní proměnné v roli vstupů – pro konkrétní situaci je potom nutné deklarace upravit podle skutečné konfigurace.

Grafické jazyky LD a CFC

V horní části *obr. 5* je uveden program v jazyce LD, pod ním je stejný program v jazyce CFC. Pro informaci: v jazyce LD značí prvek se dvěma svislými čárkami spínací kontakt a prvek s dvojicí závorek značí binární výstup („cívku relé“).

Program v jazyce v ST vypadá takto:

```
s_ruda := tl_rude;
s_zluta := tl_zlute;
s_zelena := tl_zelene;
s_modra := tl_modre;
```

Deklarace nejsou uvedeny už ani před programem v textovém jazyce ST, ale předpokládá se, že již byly uvedeny dříve. V programu ST jsou použity příkazy přiřazení, kdy proměnné vlevo od symbolu přiřazení := („pascalského rovnítka“) je přiřazena hodnota výrazu vpravo od něj – zde to jsou jen jednoduché proměnné. Každý příkaz musí být zakončený znakem; (středníkem). V dalších příkladech bude program v jazyce ST sloužit zároveň jako názorná forma zadání logické funkce v podobě logického výrazu – a současně jako jedno z řešení. Proto bude uváděn na začátku příkladů.

Všechny tři programy jsou rovnocenným řešením zadání příkladu. Program bude správně fungovat jak pro jednotlivě tisknutá tlačítka, tak pro několik tlačítek tisknutých současně.

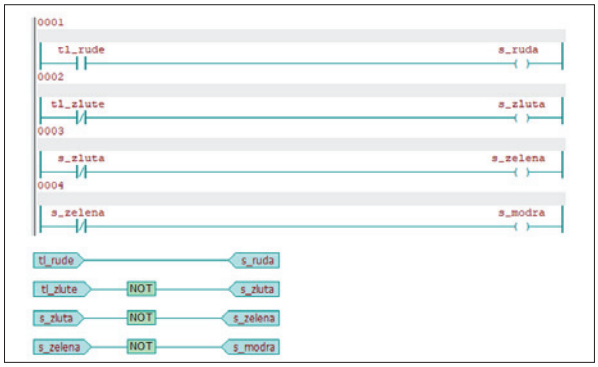
Příklad 2. na pořadí záleží

Červeným tlačítkem ovládejte červenou signálku jako v předchozím případě, shodně se stavem žlutého tlačítka ovládejte žlutou a zelenou signálku, shodně se stavem zeleného tlačítka ovládejte všechny signálky podle přiřazení:

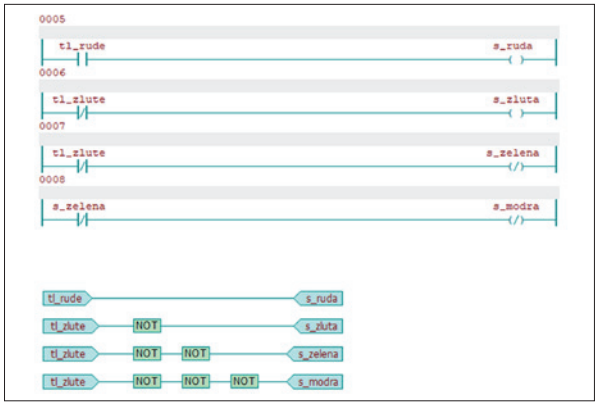
```
s_ruda := tl_rude;
s_zluta := tl_zlute;
s_zelena := tl_zlute;
s_ruda := tl_zelene;
s_zluta := tl_zelene;
s_zelena := tl_zelene;
s_modra := tl_zelene;
```

Řešení v LD a CFC

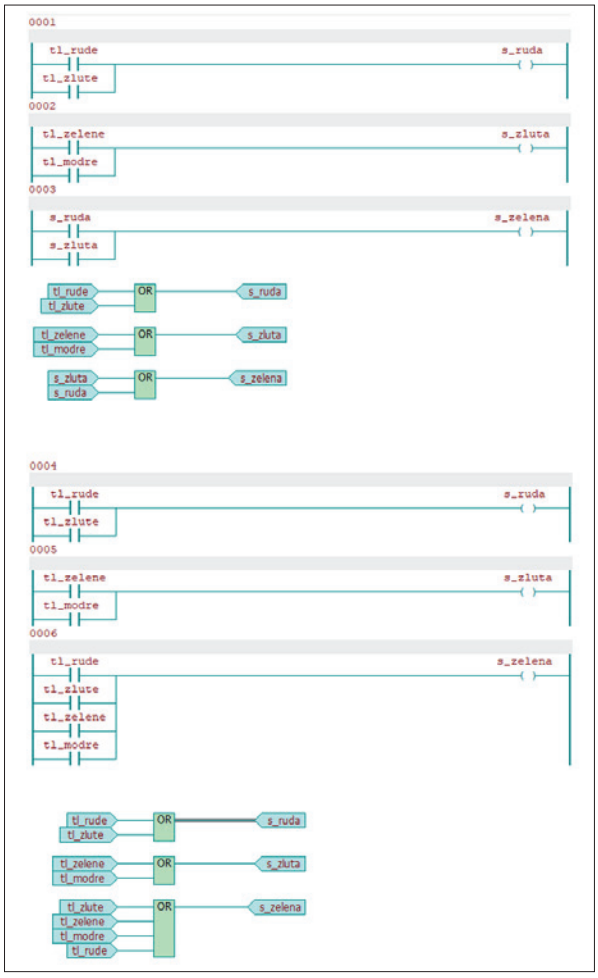
Program v grafických jazycích je uveden v *obr. 6*. Jeho poslední příkaz duplicitně provádí duplicitní zápis do proměnných s_ruda, s_zluta, s_zelena. Dochází ke kolizi požadavků na zápis. Stisk modrého tlačítka ovlivní stav signálek v rozporu s požadavky předchozích příkazů podle zásady „poslední má pravdu“. Při změně pořadí příkazů se situace změní. Správně bude program fungovat jen při tisknutí tlačítek jednotlivě. Doporučuje se při programování se podobných situací vyvarovat.



Obr. 7. K prvnímu řešení příkladu 3 (negace)



Obr. 8. K druhému řešení příkladu 3 (negace)



Obr. 9. K příkladu 4 – obě varianty řešení (logický součet OR)

Příklad 3. negace

Zadáním je červenou signálku ovládat shodně se stavem červeného tlačítka, žlutou signálku opačně ke stavu žlutého tlačítka, zelenou signálku opačně ke stavu žluté signálky a modrou opačně ke stavu zelené:

```
s_ruda := tl_rude;
s_zluta := NOT tl_zlute;
s_zelena := NOT s_zluta;
s_modra := NOT s_zelena;
```

První řešení

Operátor NOT provádí negaci logické proměnné, která za ním následuje – mění její pravdivostní hodnotu na opačnou. Způsobí, že žlutá signálka svítí opačně ke stavu žlutého tlačítka. Zelená signálka pak svítí opačně než žlutá a modrá opět opačně než zelená. To znamená, že zelená signálka svítí shodně se stavem žlutého tlačítka a modrá k němu opačně, tedy shodně se zelenou signálkou. Platí, že negace negované proměnné je rovna původní hodnotě této proměnné – negace negace (obecně sudý počet negací) se navzájem ruší. Zadání rovnocenně odpovídají oba programy v *obr. 7*.

Druhé řešení

Program je možné rovnocenně přepsat:

```
s_ruda := tl_rude;
s_zluta := NOT tl_zlute;
s_zelena := NOT (NOT tl_zlute);
s_modra := NOT (NOT (NOT tl_zlute));
```

Závorky jsou uvedeny jen pro přehlednost, pro funkci programu nejsou nutné.

Řešení jsou v *obr. 8*. V kontaktním schématu LD lze negaci řešit dvojím způsobem – negací vstupní proměnné (znázorněné lomítkem mezi svislými čárkami) nebo negací výstupní proměnné (znázorněné lomítkem mezi závorkami). Třetí negaci zde již nelze provést a je řešena negací výstupní proměnné s_zelena. Program v CFC je přesným přepisem programu ve ST.

Logický součet a součin

Příklad 4. logický součet OR

Červená signálka má svítit, jestliže je stisknuto červené nebo žluté tlačítko (nebo obě současně – to je vyjádřeno tím, že se před nebo nepíše čárka), a je zhasnutá, není-li stisknuto ani jedno z obou tlačítek. Obdobně se požaduje, aby žlutá signálka svítila, jestliže je stisknuto zelené nebo modré tlačítko. Zelená signálka má svítit, jestliže svítí červená nebo žlutá signálka.

První řešení

s_ruda := tl_rude OR tl_zlute;
s_zluta := tl_zelene OR tl_modre;
s_zelena := s_ruda OR s_zluta;

Operátor OR provádí operaci logického součtu (inkluzivního) proměnných, které spojuje. V češtině jej lze interpretovat jako spojku *nebo* (nikoliv *buď – nebo*, kdy má spojka *nebo* význam vylučovací a píše se před ní čárka). Výsledek je pravdivý, jestliže je pravdivý alespoň jeden z operandů. Program lze rovnocenně přepsat do tvaru v horní části *obr. 9*.

Druhé řešení

Program lze rovnocenně upravit na:
s_ruda := tl_rude OR tl_zlute;
s_zluta := tl_zelene OR tl_modre;
s_zelena := (tl_rude OR tl_zlute) OR (tl_zelene OR tl_modre);

Výraz pro zelenou signálku je důsledkem asociativnosti operace logického součtu OR. Závorky v posledním příkazu nejsou nutné, jsou uvedeny jen pro zdůraznění skutečnosti, že logický součet dílčích součtů je shodný se součtem všech operandů. Grafické verze programu jsou v dolní části *obr. 9*.

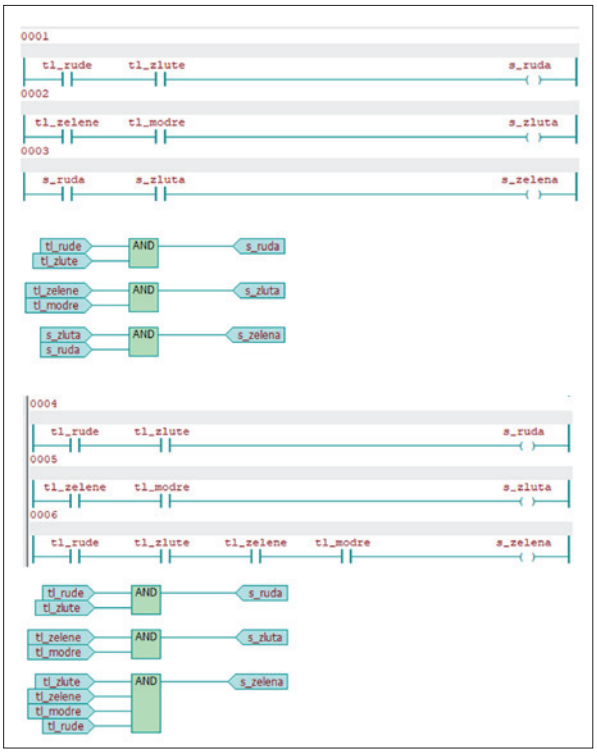
Příklad 5. logický součin AND

Červená signálka má svítit jenom tehdy, je-li stisknuto červené tlačítko současně se žlutým, a je zhasnutá, jestliže některé z tlačítek není stisknuto. Obdobně se požaduje, aby žlutá signálka svítila, jestliže je tisknuto zelené a současně modré tlačítko. Zelená signálka má svítit, svítí-li červená signálka současně se žlutou.

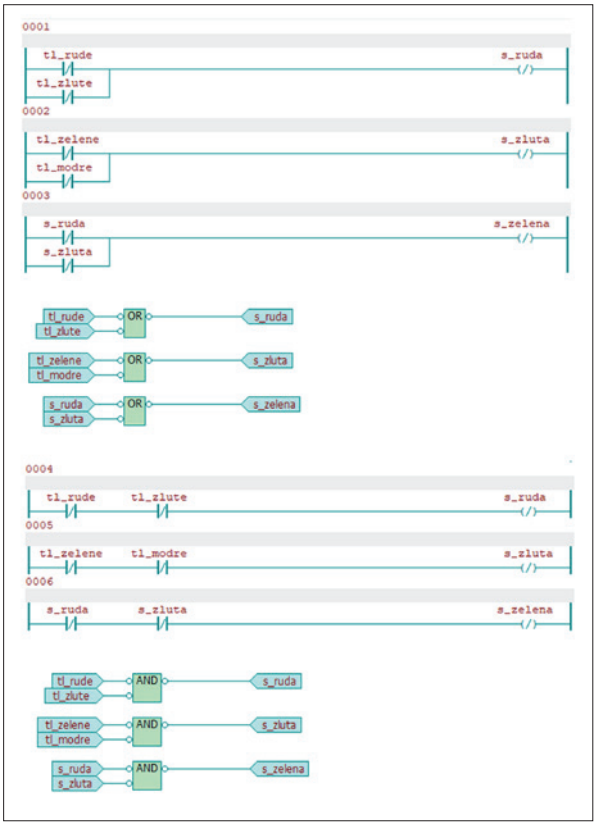
První řešení

s_ruda := tl_rude AND tl_zlute;
s_zluta := tl_zelene AND tl_modre;
s_zelena := s_ruda AND s_zluta;

Operátor AND provádí operaci logického součinu proměnných, které spojuje. V češtině jej lze interpretovat spojkou *a* ve významu *současně*. Výsledek je pravdivý, jestliže jsou pravdivé všechny z operandů. Je-li alespoň jeden z operandů nepravdivý, je nepravdivý i výsledek. Místo symbolu AND lze v logických výrazech jazyka ST rovnocenně používat znak „&“.



Obr. 10. K příkladu 5 (logický součin AND)



Obr. 11. De Morganova pravidla

Druhé řešení

Program lze přepsat do tvaru:
s_ruda := tl_rude AND tl_zlute;
s_zluta := tl_zelene AND tl_modre;
s_zelena := tl_rude AND tl_zlute AND tl_zelene AND tl_modre;
nebo do tvaru:
s_ruda := tl_rude & tl_zlute;
s_zluta := tl_zelene & tl_modre;
s_zelena := tl_rude & tl_zlute & tl_zelene & tl_modre;

Oběma variantám řešení odpovídají grafické programy v *obr. 10*.

Ověření De Morganových pravidel

Ověřte, že program
s_ruda := NOT ((NOT tl_rude) OR (NOT tl_zlute));
s_zluta := NOT (NOT tl_zelene OR NOT tl_modre);
s_zelena := NOT (NOT s_ruda OR NOT s_zluta);
se chová shodně s řešením předchozího příkladu (*obr. 10*) a naopak, že program:
s_ruda := NOT ((NOT tl_rude) & (NOT tl_zlute));
s_zluta := NOT (NOT tl_zelene & NOT tl_modre);
s_zelena := NOT (NOT s_ruda & NOT s_zluta);
se chová shodně s řešením příkladu 4 (*obr. 9*). Tato (snad) překvapivá souvislost mezi operacemi logického součtu OR a součinu AND je důsledkem De Morganových pravidel, která jsou součástí Booleovy algebry. K programu v ST uveďme, že vnořené závorky v prvním řádku obou programů jsou uvedeny jen pro přehlednost a nejsou nutné pro správné vykonání programu jako u ostatních příkazů. Operátor NOT má při vykonávání programu přednost před operátory OR a AND – nejprve se tedy provede negace odpovídající proměnné a teprve potom operace OR nebo AND. Nezbytné jsou jen závorky za prvním operátorem NOT. Programy obou verzí v grafických jazycích jsou v *obr. 11*. V jazyce CFC negaci znázorňují kroužky u vstupu nebo výstupu značky logické funkce.

Příklad 6. výlučný logický součet XOR, ekvivalence a parita

Červená signálka má svítit, jestliže je stisknuto buď (jen) červené tlačítko, nebo (jen) žluté tlačítko (nikoliv obě současně), a je zhasnutá, jestliže není tisknuto žádné z tlačítek nebo jsou stisknuta obě tlačítka současně. Dále požadujeme, aby žlutá signálka svítila, jestliže jsou červené a žluté tlačítko ve shodném stavu (obě tisknutá nebo obě uvolněná). Zelená signálka má svítit, jestliže je stisknuto buď zelené, nebo modré tlačítko (právě jedno). Modrá signálka má svítit, je-li počet stisknutých tlačítek liché číslo.

První řešení:

Bez důkazů a odvozování uvádíme program, který řeší požadavky zadání:

s_ruda := (tl_rude & NOT tl_zlute) OR (NOT tl_rude & tl_zlute);

s_zluta := (tl_rude & tl_zlute) OR (NOT tl_rude & NOT tl_zlute);

s_zelena := tl_zelene XOR tl_modre;

s_modra := tl_rude XOR tl_zlute XOR tl_zelene XOR tl_modre;

Program pro červenou signálku vychází přesně z požadavku zadání – signálka svítí, jestliže je stisknuto rudé tlačítko a není stisknuto žluté (tl_rude & NOT tl_zlute) nebo je stisknuto žluté a není stisknuto rudé (NOT tl_rude & tl_zlute). To je definice logické funkce výlučného součtu, anglicky pojmenovaného *exclusive OR*, zkráceně XOR. Stejně jsou definovány i další logické funkce dvou proměnných:

- neshoda (*nonekvivalence* – NEQ): výstup pravdivý, jestliže oba operandy mají odlišné hodnoty (neshodují se, tlačítka jsou v odlišných stavech),
- právě jeden ze dvou (S1_2): výstup je pravdivý, jestliže je právě jeden z operandů pravdivý (je stisknuto právě jedno tlačítko),
- lichá parita (*parity odd*, PO): výstup je pravdivý, jestliže je lichý počet operandů pravdivých (zde je podmínka splněna právě pro jediné tisknuté tlačítko),
- modulo 2 (M2): pravdivost výstupu je shodná s výsledkem sčítání binárních číslic bez přenosu (součtu modulo 2),
- funkce schodišťového ovladače.

Tyto logické funkce lze zobecnit pro větší počet operandů, ale pak již nejsou všechny shodné – shodují se pouze funkce liché parity, součtu modulo 2 a schodišťového ovladače.

Podmínkou pro svit žluté signálky je shodný stav obou tlačítek – logická funkce shody (ekvivalence, EQ), který je negací funkce neshody – svítí tedy opačně (inverzně) ke stavu červené signálky. Svit zelené signálky je ovládán shodnou logickou funkcí, která je zde realizována s použitím operátoru XOR. Podmínkou pro svit modré signálky je lichý počet stisknutých tlačítek. Stejným způsobem by bylo ovládáno osvětlení schodiště nebo jiného společného prostoru (chodby, haly) ze čtyř míst.

Druhé řešení:

Zadání vyhovuje i další varianta programu:

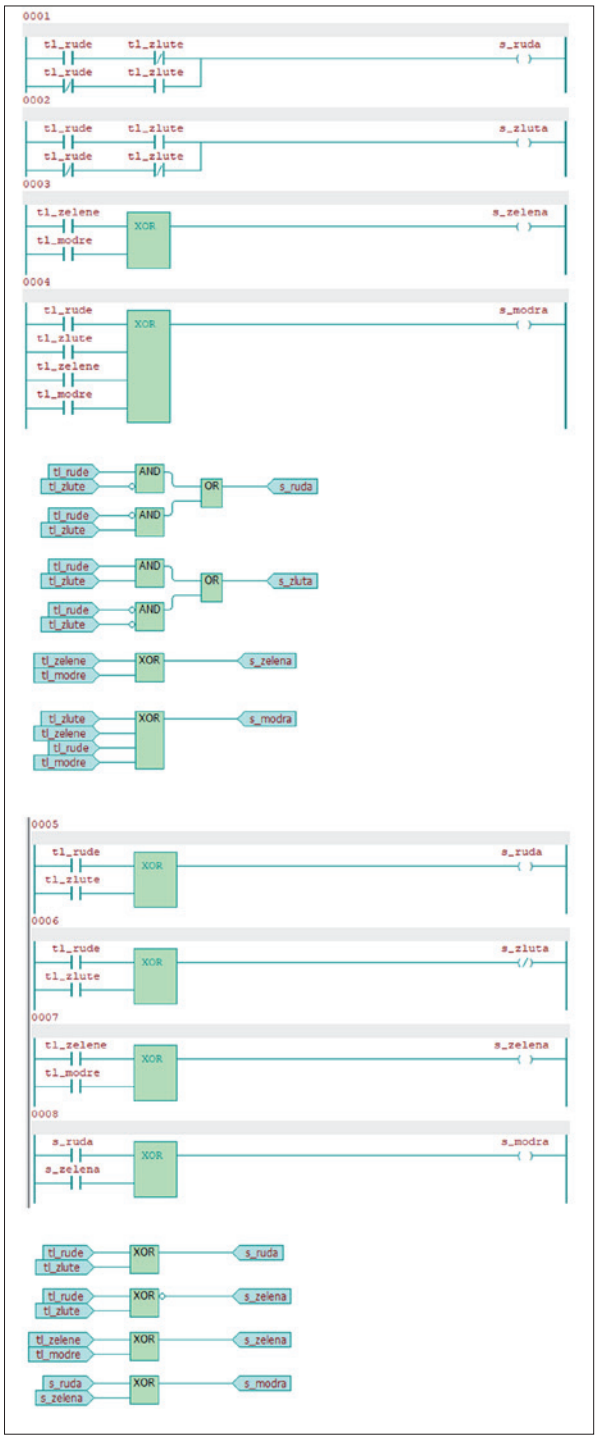
s_ruda := tl_rude XOR tl_zlute;

s_zluta := NOT(tl_rude XOR tl_zlute);

s_zelena := tl_zelene XOR tl_modre;

s_modra := s_ruda XOR s_zelena;

Programy pro obě varianty řešení v grafických jazycích jsou v *obr. 12*. Funkce shody (EQ) a její negace (neshody NEQ) jsou využívány při řešení bezpečných systémů s dvojitou nadbytečností (redundancí), kdy důležité proměnné (vstupní a výstupní) jsou zdvojené a jejich hodnoty se průběžně porovnávají – zjištěná neshoda je vyhodnocena jako chyba. Jsou využívány i při vyhodnocování sledu dvou časových vzorků stejného signálu (současný a minulý) a jejich neshoda signalizuje, že se změnila logická proměnná, popř. lze vyhodnotit výskyt náběžné nebo sestupné hrany signálu.



Obr. 12. K příkladu 6 (výlučný součet XOR a příbuzné funkce)

Příklad 7. majorita ze tří a prahové funkce

Červená signálka má svítit, jestliže je stisknuta většina z prvních tří tlačítek (červené, žluté, zelené), tedy alespoň dvě (dvě nebo tři). Žlutá signálka má svítit, jsou-li stisknuta alespoň dvě ze čtyř tlačítek (tedy kterákoliv dvě, tři nebo čtyři tlačítka).

Řešení

Bez důkazů uvádíme program, který řeší požadavky zadání:

s_ruda := (tl_rude & tl_zlute) OR (tl_rude & tl_zelene) OR (tl_zlute & tl_zelene);

s_zluta := s_ruda OR (tl_rude & tl_modre) OR (tl_zlute & tl_modre) OR (tl_zelene & modre); // toto je řádkový komentář – může obsahovat libovolné znaky až do konce řádku

V jazyce ST se znak „nový řádek“ interpretuje stejně jako znak mezera –, obecný komentář začíná dvojicí znaků (*, může obsahovat libovolné znaky, mít libovolnou délku a končí dvojicí znaků *).

Programy grafických jazycích již nejsou uvedeny. Přepis z jazyka ST je rutinní operace, zřejmá z předchozích příkladů. První příkaz (pro červenou signálku) definuje funkci „majorita ze tří“ (M3). Je pravdivá, jestliže je většina ze tří operandů pravdivých (tedy alespoň dva). K řešení lze dojít úvahou: má-li být výsledek pravdivý, stačí, aby alespoň dva z některých operandů byly pravdivé. Řešením je tedy logický součet dvoumístných součinových členů, ve kterých se postupně vystřídají všechny operandy – pro tři proměnné to jsou tři součinové členy. Druhý příkaz (pro žlutou signálku) definuje prahovou funkci „alespoň dva ze čtyř“ (P2_4). K řešení můžeme dospět stejnou úvahou s tím, že sčítat budeme šest dvoumíst-

ných součinových členů, v nichž se vystřídají všechny operandy – první tři jsou shodné s definicí funkce M3, stačí tedy k výstupu s_ruda přičíst zbývající tři.

Majorita ze tří se využívá při řešení bezpečných systémů s trojitou nadbytečností (redundancí). Důležité proměnné (vstupní a výstupní) jsou ztrojené a jejich hodnoty se průběžně porovnávají. Zjištěná neshoda je vyhodnocena jako chyba (kterou je nutné později opravit), ale pro současnou aktivitu je využívána hodnota, na které se shodne většina z trojice proměnných (M3). Obě funkce (M3 a P2_4) patří do skupiny prahových funkcí, jejichž výsledek je pravdivý, jestliže je pravdivý alespoň prahový počet operandů. V obou případech je práh roven dvěma. Tyto a další prahové funkce lze opět využít ke zvýšení bezpečnosti, např. při vyhodnocení souboru zabezpečovacích senzorů (např. požárních) – omezí se tak závislost na případné poruše některého ze senzorů a současně lze zabránit falešným poplachům, popř. stanovit naléhavost poplachu.

Ing. Ladislav Šmejkal, CSc., Teco, a. s.
a externí redaktor Automa,
Ing. Josef Kovář a Ing. Zuzana Prokopová,
učitelé automatizace na SPŠ Zlín

Augustus De Morgan (1806–1871)

Augustus De Morgan se narodil 27. června 1806 v Madrásu v Indii jako pátý potomek britského důstojníka. Rodina se vrátila do Británie, když bylo Augustovi sedm měsíců.

Augustus měl chabou fyzickou konstituci a již jako dítě oslepl na jedno oko. Ve škole byl pro svou odlišnost často šikanován a jeho výsledky byly nevalné. V šestnácti ale vstoupil na Trinity College Cambridge, kde učitelé poznali jeho matematický talent. Později se vrátil do Londýna, kde byl přijat jako vedoucí katedry matematiky na právě založené University College London – ve svých 21 letech a bez toho, že by měl jakékoliv předchozí publikace v matematice. O dva roky později se stal profesorem. Kromě logiky vynikal v algebře, zabýval se komplexními čísly a jejich geometrickou interpretací a rozvinul matematickou indukci.

Zákony, které jsou po něm pojmenovány, byly známy již Aristotelovi. Slovně je ve 14. století formuloval William z Ockhamu, De Morgan je první formuloval matematicky.

Augustus De Morgan je spolu s jiným britským matematikem Georgem Boolem považován za zakladatele renesance zájmu o matematickou logiku v 19. století.

Augustus De Morgan měl pedantickou povahu a byl velmi zásadový. Z Trinity College odešel jen s nižším titulem, protože odmítl složit zkoušku z teologie; později z principiálních důvodů na čas opustil post vedoucího katedry matematiky na University College London; odmítal čestné vědecké tituly i nabídku členství v britské Královské společnosti; z principu se neúčastnil ani veřejného a politického života. Spoluzaložil Londýnskou společnost pro matematiku, jejímž byl prvním prezidentem.

(Zdroj: Encyclopedia Britannica, obrázek: Wikipedia) *(Bk)*

Fórum Automa, MSV Brno 2018

Na základě úspěchu a zkušeností z organizace Fóra automatizace na veletrhu Amper byla redakce našeho časopisu oslovena realizačním týmem brněnského Mezinárodního strojírenského veletrhu s nabídkou uspořádat podobnou akci.

Po tři dny se tak v prostorách pavilonu C, který byl tradičně věnován řídicím systémům, měření a regulaci, speciálním senzorům a mnoha dalším produktům a službám, většinou přímo spojeným s problematikou řízení průmyslu, konalo Fórum Automa. Každý den byl věnován jiné oblasti – digitální továrně, komunikacím, robotům a automatizaci. Vzhledem k jednotlivým okruhům bylo jasné, že nelze vytyčit pevné hranice a mnohé přednášky se nutně musely dotknout několika témat současně. Provázanost celé problematiky digitální továrny ani jinou možnost nedává.

Velmi pozitivní byla účast posluchačů na přednáškách. Mnoho z nich se dostavilo cíleně na konkrétní prezentaci, velmi často v diskusi řešili s přednášejícím konkrétní problémy a odborné rozhovory pokračovaly i mimo prezentační prostor. Tuto pozitivní

Obr. 1. Fórum Automa v průběhu MSV v Brně

zkušenost si odnesla valná většina přednášejících. Ti mnohdy ocenili možnost vystoupit na MSV, přestože sami nevystavovali. Mohli se tak přesvědčit o zájmu návštěvníků strojírenského veletrhu o automatizaci a možná je to přesvědčí i o tom, aby se příští rok zúčastnili i jako vystavovatelé.

Během prvního dne Fóra Automa byly předvedeny prezentace, které se především týkaly digitalizace průmyslu a komplexního přístupu k otázce sběru dat a komunikacím, jejich archivaci a vyhodnocení a dalšímu využití pro řízení podniku. Tady se představily společnosti COMPAS automatizace, mySCADA, ifm electronic, Amtek a WAGO-Elektro.

Druhý den byl věnován robotům a robotizaci. Zde využily příležitost oslovit posluchače společnosti Tiesse Praha, Kuka, Humusoft, Axiomtech a PEKAT Vision.

V závěrečný den Fóra Automa představily své projekty Národní centrum Průmyslu 4.0, FANUC, SMC Industrial Automation CZ, Advanced Risk Management, Intemac Solutions a ServisControl.

Téměř dvě desítky přednášejících, několik stovek posluchačů, konkrétní diskuse o několika obchodních případech a spousta technických rad, tak je možné shrnout letošní Fórum Automa na MSV v Brně.

Radim Adam