Logické funkce ve výpočetní technice

Michal Košek

NMAG166 2018/2019

1 Úvod

1.1 CPU [1]

Počítačový procesor (central processing unit, CPU) je součástka, která má na starost zpracování strojového kódu. Mezi jednotlivé instrukce patří především posun paměti a aritmetické a logické operace, každý procesor má ovšem vlastní sadu instrukcí, které zvládne vykonat. Tyto operace provádí aritmeticko-logická jednotka (ALU), která na vstupu dostane operaci a zpravidla dva operandy a na výstupu výsledek operace. Pro posun paměti je operandem úsek paměti, na který chceme posun aplikovat a výstupem je tento úsek po aplikaci posunu. [2]
V této práci se budeme zabývat výhradně logickými operacemi na dvou operandech. Ty jsou realizovány logickými hradly.

1.2 Logická hradla

Logická hradla jsou elektronické součástky, jejichž jedinou prací je realizace logických operací s elektrickým proudem jako operandy. Ke konstrukci se dříve používaly diody nebo kombinace rezistorů a tranzistorů, případně diod a tranzistorů. V dnešní době převládá využití výhradně tranzistorových součástek. [3]
Z jednotlivých logických hradel je pak sestaven složitější obvod, jehož prací je celkové vyhodnocování logických funkcí v počítači. Mezi nejdůležitější parametry takového obvodu patří jeho průměrná rychlost (resp. čas na jednu operaci), velikost (fyzický objem) a konstrukční cena. My se nyní budeme snažit o minimalizaci počtu různých hradel, která se v obvodu vyskytují. To má za efekt snížení průměrné rychlosti a zvýšení objemu za cenu jednodušší a levnější konstrukce. Čas na operaci i objem mají jednoduchou, přibližně lineární závislost na počtu použitých tranzistorů, tedy už konstrukce jednoho hradla ze dvou menších má za následek zdvojnásobení objemu a potřebného času.
Problém je tedy zajímavý ve chvíli, kdy nám záleží na ceně a jednoduchosti (tedy i spolehlivosti) systému více než na rychlosti a objemu. Vedle toho nám objasnění vztahu mezi logickými spojkami umožní zjednodušit konstrukci procesoru a zefektivnit implementaci logických operací v překladačích a interpretech i v případě, že nebudeme omezeni počtem různých hradel.

2 Predikátová logika

K popisu funkce logických hradel se s výhodou využívá poznatků predikátové logiky a Booleovy algebry. Pro zjednodušení zde ovšem Booleovu algebru využívat nebudeme, případné její myšlenky budou uvedeny v jazyce matematické logiky.

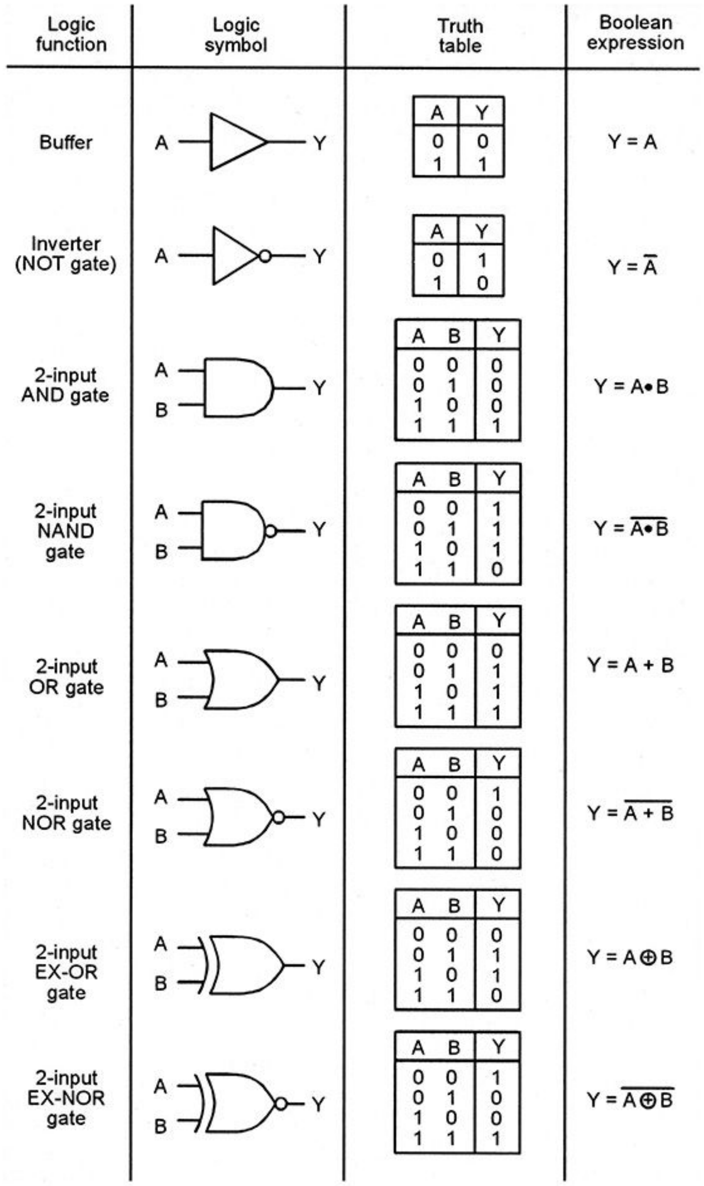
2.1 Logické operátory

Abychom mohli problém vůbec formulovat, potřebujeme nejprve několik pojmů z oblasti matematické logiky. Především je třeba ujasnit si, co je myšleno pojmem *logický operátor*.

Definice: *Unárním logickým operátorem* rozumíme každé zobrazení $\{0,1\}\rightarrow\{0,1\}$.
Značení: Obraz prvku P při zobrazení A značíme P A.
Definice: *Operátorem NOT* rozumíme unární logický operátor daný výčtem NOT 0 = 1, NOT 1 = 0.
Definice: *Binárním logickým operátorem* (také *spojkou* nebo *funkcí*) rozumíme každé zobrazení $\{0,1\}\times\{0,1\}\rightarrow\{0,1\}$, kde $\times$ značí kartézský součin.
Značení: Obraz vektoru (P, Q) při zobrazení A značíme P A Q.
Poznámka: Unární a binární logické operátory budeme souhrnně nazývat *logické operátory*, *spojky*, případně *funkce.* Bude-li nás zajímat struktura operátoru (jak vznikl složením jiných operátorů), budeme o něm mluvit jako o *(výrokové) formuli*.
Definice: *Úplným systémem logických funkcí* (dále také jen *úplným systémem*) [4] rozumíme množinu operátorů, z nichž lze spolu s operací skládání (ve smyslu běžných funkcí) vytvořit všechny logické operátory.

2.1.1 Seznam operátorů

V textu budou operátory běžně uváděny v podobě anglických názvů jim odpovídajících logických hradel. Pro úplnost následuje osm běžných spojek i s jejich grafickým a booleovským zápisem [5].



Obrázek 1: Seznam vybraných logických operátorů

2.2 Formulace problému

Na závěr kapitoly formulujeme pomocí právě zavedených pojmů problém, jímž se budeme zabývat ve třetí kapitole a jehož řešení [6] významně přispělo k pokroku IT ve 20. století (viz čtvrtá kapitola): Jaké jsou všechny jednoprvkové úplné systémy logických funkcí?

3 Úplné systémy logických funkcí

3.1 AND, OR, NOT

K důkazu úplnosti systému přímo z definice je potřeba nejprve zkonstruovat všech 16 logických spojek pomocí tohoto systému. Především pro svou intuitivní srozumitelnost se k tomu nejlépe hodí systém AND, OR, NOT. Přestože je tříprvkový, je tudíž vhodné začít s ním.

Věta: Operátory AND, OR a NOT tvoří úplný systém.
Důkaz: Každá logická spojka je jednoznačně určena výčtem hodnot všech prvků. Nechť F je logická funkce. Popíšeme algoritmus, který z daných operátorů vytvoří výrokovou formuli s totožným výčtem hodnot, jako má funkce F.
Nejprve vytvoříme dílčí operátory $A_{0,0}$, $A_{0,1}$, $A_{1,0}$, $A_{1,1}$ podle následujících předpisů:
P $A_{0,0}$ Q = (NOT P) AND (NOT Q),
P $A_{0,1}$ Q = (NOT P) AND Q,
P $A_{1,0}$ Q = P AND (NOT Q),
P $A_{1,1}$ Q = P AND Q.
Z těchto operátorů pak vybereme ty $A_{P,Q}$, pro které P F Q = 1. Takové označíme $B_1,\ldots, B_n$. Konečně C ≡ B_1 OR … OR B_n je hledaná formule.
Tento algoritmus zřejmě vždy vytvoří platnou spojku. Navíc vektory (P, Q), pro které platí P F Q = 1, splňují i P C Q = 1, protože P $A_{P,Q}$ Q = 1, a tedy pro nějaké *i* platí B_i = 1.
Uvažujme vektory (P, Q), pro které P F Q = 0. Z definice dílčích operátorů $A_{0,0}$, $A_{0,1}$, $A_{1,0}$, $A_{1,1}$ plyne, že P $A_{R,S}$ Q = 0 pro všechny vektory (R, S) ≠ (P, Q). Dále protože P F Q ≠ 1, neexistuje *i* takové, že B_i ≡ $A_{P,Q}$. Celkem pro každé *i* platí P B_i Q = 0. Odtud P C Q = 0.

□

Poznámka: K důkazu úplnosti systému logických spojek dále postačí zkonstruovat z těchto funkcí nějaký systém, o kterém již víme, že je úplný (zatím jen AND, OR, NOT).
Poznámka: Spojka AND v předchozí větě je nadbytečná. Z De Morganových zákonů víme $\forall P, Q \in \{0,1\} : P\ AND\ Q\ =\ NOT\ (NOT\ P\ OR\ (NOT\ Q))$, tedy z předchozí poznámky plyne úplnost systému OR, NOT.

3.2 NOR, NAND

Dále ukážeme existenci prvního jednoprvkového úplného systému logických funkcí. Vhodným kandidátem bude spojka NOR. Ukázat její úplnost je díky předchozímu oddílu snadné.

Věta: Operátor NOR tvoří úplný systém.
Důkaz: Stačí ukázat (viz poznámky), že ze spojky NOR lze vytvořit výrokovou formuli odpovídající spojkám OR a NOT. Z lingvistického významu operátoru NOR (ani, ani) lze nahlédnout konstrukci spojky NOT. Ta odpovídá formuli P NOR P (lze snadno ověřit například tabulkou pravdivostních

hodnot). Jelikož NOR je negací funkce OR a negace negace výroku dává původní výrok, lze formuli P OR Q zkonstruovat jako NOT (P NOR Q), tedy elementárně jako (P NOR Q) NOR (P NOR Q).

□

Druhým kandidátem je spojka NAND. Důkaz lze provést podobně jako u operátoru NOR, ukážeme však obecněji aplikovatelný postup využívající definici duálního operátoru.

Definice: *Duálním operátorem* (dále také *duálem*) k binárnímu operátoru A nazveme zobrazení B splňující: $\forall P, Q \in \{0,1\} : P\ B\ Q\ =\ NOT\ ((NOT\ P)\ A\ (NOT\ Q))$.
Značení: Operátor B z předchozí definice značíme A^d.
Poznámky: Vztah duality je symetrický, tedy $(A^d)^d \equiv A$. Duál k danému operátoru vždy existuje právě jeden. Intuitivně získáme duální operátor záměnou všech 0 a 1 v tabulce pravdivostních hodnot spojky.

Věta: Nechť $A_1, \ldots, A_n$ je úplný systém logických funkcí. Pak $A_1^d, \ldots, A_n^d$ také tvoří úplný systém.
Důkaz: Ze symetrie duality a existence duálu víme, že každý operátor je duální k nějakému jinému, tedy patří do množiny všech duálních operátorů. Odtud zřejmě množina duálních operátorů je právě množina všech operátorů.
Dále nechť F je operátor, ten lze z předpokladu zkonstruovat použitím systému $A_1, \ldots, A_n$. Definujeme operátor G, který vznikl z předchozí konstrukce nahrazením všech elementárních operátorů $A_1, \ldots, A_n$ jejich duálem. Předpokládejme, že A je poslední aplikovanou funkcí konstrukce. Aplikací definice duálního operátoru na A_1 dostaneme:
$\forall P, Q \in \{0,1\} : P\ F\ Q\ =\ (P\ B\ \ Q)\ A\ (P\ C\ Q)\ =\ NOT\ ((NOT\ (P\ B\ Q))\ A^d\ (NOT\ (P\ C\ Q)))$ pro nějaké spojky B, C.
Opakovanou aplikací definice duálu pak dostaneme:
$\forall P, Q \in \{0,1\} : P\ F\ Q\ =\ NOT\ ((NOT\ P)\ G\ (NOT\ Q))$, tedy $G = F^d$.
Spojením obou částí dokážeme zkonstruovat množinu všech duálních operátorů, tedy množinu všech operátorů pomocí funkcí $A_1^d, \ldots, A_n^d$.

□

Z věty bezprostředně vyplývá, že $NOR^d \equiv NAND$ tvoří úplný systém.

3.3 Ostatní

Nakonec ukážeme, že jiné jednoprvkové úplné systémy neexistují. Odpovědí na otázku, jaké jsou všechny jednoprvkové úplné systémy logických funkcí, jsou tedy systémy NOR a NAND.

Věta: NOR a NAND jsou jediné jednoprvkové úplné systémy.
Důkaz: Z předchozího oddílu již víme, že NOR i NAND tvoří úplné systémy. Stačí tedy dokázat, že jiné neexistují. Unární funkce zřejmě úplné systémy tvořit nemohou, zbývá tedy ověřit zbývajících 14 binárních funkcí.
Funkce F, pro které 0 F 0 = 0, nemohou samy tvořit jednoprvkový úplný systém, jelikož z nich zřejmě skládáním nelze vytvořit funkci, pro kterou 0 G 0 = 1. Podobně funkce F, pro které 1 F 1 = 1, nemohou samy tvořit úplný systém. Zbývají funkce NOR, NAND, NOT P a NOT Q (respektive jejich binární

verze). Z funkce NOT P nelze vytvořit funkci (Q) a naopak. Zbývají tedy pouze funkce NOR a NAND, které tvoří úplné systémy.

□

4 Apollo Guidance Computer [7]

Významným využitím úplnosti systému NOR je řídící počítač všech modulů Apollo zvaný AGC. Ten se skládal ze 4100 samostatných hradel NOR, pozdější verze až z 5600 hradel [8]. Všechny ostatní logické funkce byly tedy realizovány skládáním funkce NOR. Toto řešení se ukázalo jako dostatečné a přestože počítač při přistání modulu Apollo 11 selhal [9], na vině nebylo řešení logiky.
Postupem času zvítězil požadavek rychlosti procesoru nad jednoduchostí výroby, proto jsou moderní procesory složeny z řady různých logických hradel a jedna logická operace tak odpovídá jednomu průchodu hradlem. Přesto hrají NOR a NAND významnou roli ve vývoji procesorů.

5 Seznam použité literatury

[1] Přispěvatelé Wikipedie. *Central processing unit, Wikipedie: Otevřená encyklopedie* [online]. [cit. 10. 9. 2019]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Central_processing_unit&oldid=914957182.

[2] Přispěvatelé Wikipedie. *Arithmetic logic unit, Wikipedie: Otevřená encyklopedie* [online]. [cit. 10. 9. 2019]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Arithmetic_logic_unit&oldid=911245694.

[3] *Logic Families, Digital Byte* [online]. [cit. 10. 9. 2019]. Dostupné z: http://digitalbyte.weebly.com/logic-families.html.

[4] *Základní logické funkce, ELUC* [online]. [cit. 10. 9. 2019]. Dostupné z: https://eluc.kr-olomoucky.cz/verejne/lekce/475.

[5] Ray Marston. *Understanding Digital Buffer, Gate, and Logic IC Circuits - PART 1 , Nuts and Volts* [online]. [cit. 10. 9. 2019]. Dostupné z: https://www.nutsvolts.com/magazine/article/understanding_digital_buffer_gate_and_ic_circuits_part_1.

[6] THOMAS W. SCHARLE. *Axiomatization of propositional calculus with Sheffer functors*. Notre Dame J. Formal Logic 6 (1965), no. 3, 209–217, doi:10.1305/ndjfl/1093958259.

[7] *Evolution of the hardware: Old technology versus new block I and Block I designs, NASA* [online]. [cit. 10. 9. 2019]. Dostupné z: https://history.nasa.gov/computers/Ch2-4.html.

[8] Přispěvatelé Wikipedie. *Apollo Guidance Computer, Wikipedie: Otevřená encyklopedie* [online]. [cit. 10. 9. 2019]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Apollo_Guidance_Computer&oldid=914924983.

[9] Peter Adler. *Apollo 11 Program Alarms, NASA* [online]. [cit. 10. 9. 2019]. Dostupné z: https://www.hq.nasa.gov/alsj/a11/a11.1201-pa.html.