DEKOMPOZIČNÍ TECHNIKY PRO APLIKAČNĚ SPECIFICKÉ SYSTÉMY

Petr Mikušek
Výpočetní technika a informatika, 2. ročník, prezenční studium
Školitel: Václav Dvořák

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno

imikusek@fit.vutbr.cz

Abstrakt. Jednorozměrná kaskáda náhledových tabulek (LUT kaskáda) se jeví jako vhodná struktura pro realizaci logických funkcí v hardware, firmware a software. LUT kaskádu můžeme snadno získat z vícevýstupových binárních rozhodovacích diagramů (MTBDD), ale bohužel konstrukce MTBDD je poměrně složitý úkol, obzvláště pokud chceme získat optimální diagram. Cílem této disertační práce je navrhnout metodu syntézy LUT kaskád založenou na iterativní disjunktní dekompozici.

Klíčová slova. neúplně zadané funkce, vícevýstupové BDD, LUT kaskády, iterativní disjunktní dekompozice, dekompozice funkcí

1 Úvod

Dva z nejpalčivějších problémů moderních VLSI obvodů lze spatřovat v časové náročnosti vývoje a relativně krátké životnosti výsledného obvodu. Jedním z možných řešení těchto problémů mohou být rekonfigurovatelné architektury, např. komplexní programovatelné obvody CPLD a programovatelná hradlová pole FPGA. Základní strukturu těchto obvodů si můžeme představit jako dvourozměrnou matici základních logických bloků ve formě malých pamětí implementujících logickou funkci a propojovací sítě, která je obzvlášť u FPGA velmi komplexní a zabírá velkou část plochy čipu. Komplexní propojení představuje taky největší potíž syntézních metod pro FPGA. Komplexní propojení má taktéž negativní vliv na výkonnost FPGA návrhů, protože největší část zpoždění kombinační části obvodu připadá na propojení a nikoliv na užitečný logický výpočet.

Řešením je mít strukturu s určitou pravidelností nejen ve fyzickém umístění subsystémů, ale také v jejich propojení. Jednorozměrná kaskáda náhledových tabulek LUT je právě takovou pravidelnou strukturou. Tabulky LUT jsou vlastně vícevstupové, vícevýstupové univerzální logické bloky. Realizace tabulek LUT pomocí blokových pamětí RAM umožňuje podporu rekonfigurovatelných architektur, asynchronních kaskád nebo synchronních zřetězených linek; rychlost je srovnatelná s ostatními návrhy založených na FPGA [11], rozmístění a propojení je velice jednoduché. Výhody LUT kaskád oproti FPGA jsou následující: lepší využití plochy čipu, snadná a přesná předvídatelnost zpoždění a snadná testovatelnost.

Struktura této práce je následující. Kapitola 2 se zabývá dekompozicí booleovských funkcí. Kapitola 3 představuje podrobněji LUT kaskády, reprezentaci booleovských funkcí v podobě binárních rozhodovacích diagramů a stávající syntézní metody LUT kaskád. Disertaci zaměřené na návrh nové syntézní metody LUT kaskád je věnována kapitola 4.

2 Dekompozice booleovských funkcí

Dekompozice je základní problém v moderní logické syntéze. Jejím cílem je rozložit logický obvod na množinu menších spolupracujících komponent. Matematicky je dekompozice proces vyjádření funkce o n proměnných jako funkce funkcí o méně proměnných [2]. Například funkce $F(X)$ je dekomponovatelná, jestliže může být vyjádřena jako $F = H(U, G(V))$, kde U a V jsou vlastní podmnožiny množiny X vstupních proměnných a G a H mají méně vstupních proměnných než F.

Bylo navrženo mnoho dekompozičních algoritmů. Ashenhurst ve svém stěžejním článku [2] uvedl větu o disjunktivní dekompozici založenou na dekompozičních diagramech. Curtis ve svém díle [5] rozšířil výsledky Ashenhursta na vícenásobnou dekompozici ve tvaru $F = H(U, G_1(V), \ldots, G_k(V))$. Použití diagramů pro dekompozici logických sítí je použitelné pouze na omezenou třídu funkcí. O vylepšení se pokusili Roth a Karp použitím kompaktnější reprezentace funkce ve formě pokrytí on-set a pokrytí off-set [13]. Bohužel jejich metoda nepracuje přímo s vícevýstupovými funkcemi.

Když výzkumníci zjistili, že jejich dřívější algebraické metody nemohou být snadno adaptovány na LUT model, přešli k metodám založených na funkčních závislostech.Bohužel navzdory zásadní povaze problému funkční dekompozice a jeho častého využití, neexistuje mnoho uniformních přístupů k dekompozici, které mohou být aplikovány na úplně nebo neúplně zadané vícevýstupové booleovské funkce reprezentované kompaktně pokrytím booleovskými krychlemi.

Jedním z takovýchto přístupů je metoda Brzozowského a Luby [4] založená na ternární algebře a určitém zobecnění systémů množin, které autoři nazývají „pokrývky“ (blankets).

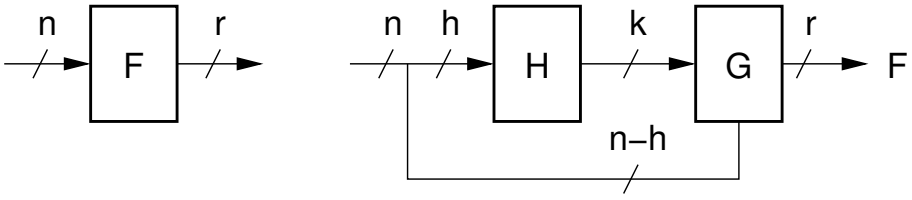
3 LUT kaskády

Implementace kombinačních logických obvodů ve formě jednorozměrného uspořádání modulů (buněk) přitahovala pozornost výzkumníků již před hodně lety. Důvodem může být jednoduchost, pravidelnost a skrytý potenciál pro budoucí VLSI technologie. Brzo se přišlo na to, že nejjednodušší jednovodičové (Maitrovy) kaskády horizontálně propojených buněk s jednotlivými (postranními) vstupy nejsou postačující k realizaci každé booleovské funkce o $n > 2$ proměnných, i když připouštíme, že každá proměnná může být připojená k více buňkám. Nicméně pokud se pro horizontální vodiče použije k-hodnotový signál a m-hodnotový signál pro postranní vstupy, každá k-hodnotová celočíselná funkce s m-hodnotovými proměnnými je pak realizovatelná uniformní redundatní kaskádou (tj. s opakovaným použitím některých vstupních proměnných) [22]. Celočíselné hodnoty mohou být nahrazeny za binárně kódované celočíselné hodnoty, pokud jsou požadovány vícevýstupové booleovské funkce. Buňky kaskády mohou být realizovány nepravidelnou logikou z hradel, multiplexorů/demultiplexorů nebo náhledovými tabulkami (Look-Up Table, LUT) uloženými v ROM nebo RAM.

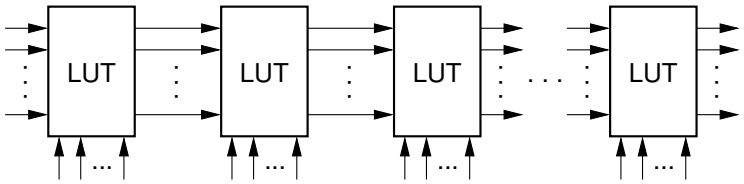
Redundatní kaskády tohoto typu byly navrhovány s použitím algebraického přístupu [22], což vedlo k nadměrnému počtu buněk, jelikož si tento přístup nevšímal komplexnosti syntetizovaných funkcí. Dlouhé kaskády byly nepraktické kvůli vysoké ceně a velkému zpoždění. Z tohoto důvodu byly redundatní kaskády víceméně zavrženy a byly žádány nejkratší možné neredundatní kaskády.

3.1 Reprezentace booleovských funkcí

Pro strojovou reprezentaci booleovských funkcí můžeme použít binární rozhodovací diagramy (Binary Decision Diagram, BDD) [3]. Redukované uspořádané binární rozhodovací diagramy (Reduced Ordered BDD, ROBDD) jsou kanonickou (unikátní) reprezentací pro danou funkci a uspořádání proměnných. Důležitým parametrem je velikost BDD, tj. celkový počet rozhodovacích uzlů, jelikož určuje velikost datové struktury potřebné k uložení BDD. Konstrukce minimálních ROBDD patří do skupiny NP-těžkých problémů [3]: velikost ROBDD závisí na uspořádání proměnných a máme $n!$ možných uspořádání n proměnných.



Obrázek 1: Disjunktní dekompozice vícevýstupové booleovské funkce F o n proměnných



Obrázek 2: LUT kaskáda

Pro reprezentaci systému booleovských funkcí pomocí rozhodovacích diagramů můžeme použít m bitově orientovaných BDD, jeden pro každou z m booleovských funkcí (eventuálně můžeme sdílet některé poddiagramy, pak se jedná o tzv. sdílené BDD – Shared BDD, SBDD) nebo jedním slovně orientovaným BDD (Word-Level BDD, WLBDD) s n booleovskými rozhodovacími proměnnými a R celočíselnými koncovými hodnotami. Druhá forma je výstižnější, ale není lehké ji získat z bitově orientovaného BDD. Existuje mnoho typů WLBDD. Vícekoncové BDD (Multi-Terminal BDD, MTBDD) mají celočíselné listy a tudíž reprezentují funkce z booleovských hodnot na celočíselné.

Existuje několik možností, jak slovně orientovaný rozhodovací diagram reprezentovat pomocí jednoho bitově orientovaného rozhodovacího diagramu. Zakódovaná charakteristická funkce nenulových výstupů (Encoded Characteristic Function of Non-zero outputs, ECFN) [15] je další reprezentací vícevýstupových funkcí, která používá nejkratší zakódování výstupních vektorů s použitím pomocných proměnných. Pomocné proměnné mohou být libovolně promíchány s normálními proměnnými [15]. Podobným způsobem jsou konstruovány i diagramy SBDD+ a MTBDD+ z diagramů SBDD a MTBDD [20, 17].

Další reprezentací k popisu vícevýstupových funkcí používá bitově orientované BDD je BDD pro charakteristickou funkci (characteristic function, CF) – BDD pro CF [1]. Vstupní a výstupní proměnné jsou seřazeny tak, že výstupní proměnné se nacházejí pod její podporou. Podpora funkce je množina proměnných, na kterých daná funkce závisí.

3.2 Syntéza LUT kaskád

Libovolná logická funkce může být realizována pomocí jediné LUT. Tento přístup je ve vestavěných systémech přijatelný jen do zhruba desítky proměnných, protože velikost paměti roste s počtem proměnných exponenciálně. Pro desítky proměnných musíme použít kompaktnější datovou strukturu a jedna z cest jak ji získat, je použít disjunktní dekompozici na původní funkci F a získat tak dvě funkce G a H, jejichž celková velikost může být menší než velikost funkce F. Základní ideu můžete vidět na obrázku 1. Opakovaným (iterativním) použitím disjunktní dekompozice na danou funkci získáme LUT kaskádu zobrazenou na obrázku 2.

Výše uvedená kaskáda má k horizontálních vodičů přenášejících booleovské hodnoty a každá buňka má m vertikálních (bočních) vstupů. Poslední buňka v kaskádě může mít $r \neq k$ výstupů. Kaskáda se označuje jako neredundantní pokud každá proměnná vstupuje pouze do jediné buňky. Jinak je kaskáda redundantní.

Metoda syntézy zdola nahoru

V [6] byla prezentována metoda souběžné syntézy LUT kaskád/MTBDD pomocí iterativní disjunktní dekompozice. Metoda pracuje nad úplně nebo částečně zadanými funkcemi celočíselných hodnot zadaných pomocí tabulky (mapy) funkce. Základním principem metody je počítání podfunkcí, které generuje daná m-ární proměnná. Syntéza kaskády probíhá zdola nahoru tj. od poslední buňky k první a zároveň se jako vedlejší produkt získává MTBDD od listů ke kořenu, jelikož jedna úroveň LUT kaskády odpovídá jedné úrovni MTBDD. V každém dekompozičním kroku se vyzkouší všechny doposud nepoužité proměnné a pro každou proměnnou se určí počet podfunkcí generovaných danou proměnnou. Pro výběr proměnné, podle které se provede dekompozice, se používá heuristika, která se snaží minimalizovat šířku MTBDD a tím pádem i počet propojovacích horizontálních vodičů.

Metoda syntézy shora dolů

Myšlenka použít LUT kaskády pro realizaci logických funkcí byla po dlouhé době znovu objevena v [20] a dále rozpracována [16, 17, 10]. V [18] byla prezentována metoda návrhu LUT kaskády pomocí BDD pro CF. Pro danou vícevýstupovou funkci se vygeneruje BDD pro CF s výchozím uspořádáním podle vzrůstající velikosti podpory proměnných a minimalizuje pomocí tzv. sifting algoritmu [14], kde se jako cenová funkce používá šířka BDD. Následně se vybere několik proměnných poblíž kořenového uzlu, přiřadí se binární kódy a provede se odejmutí uzlů a rekonstrukce BDD pro CF. Tento postup se opakuje do té doby, než jsou vybrány všechny proměnné.

Metoda byla dále rozšířena pro neúplně zadané vícevýstupové funkce [19]. Byl navržen nový způsob reprezentace neúplně zadaných vícevýstupových funkcí pomocí BDD pro CF. V případě neúplně zadaných funkcí se může dále zredukovat šířka BDD pro CF nalezením vhodného přiřazení konstant don't care hodnotám. Postup dekompozice je stejný jako v [18] s tím rozdílem, že po minimalizaci BDD pro CF pomocí sifting algoritmu je zařazen krok dále minimalizují šířku pomocí navržené heuristiky.

4 Disertační práce

Pro uniformní reprezentaci LUT kaskád v hardware, software a firmware se jeví jako ideální MTBDD [6], protože LUT kaskádu získáme přímo jako řezy MTBDD. Bohužel je velikost MTBDD ve srovnání s bitově orientovanými BDD typicky větší a mnohdy u složitejších funkcí příliš veliká pro konstrukci pomocí stávajících metod slovně orientovaných BDD [17]. Řešením může být použití bitově orientovaných BDD, které reprezentují danou funkci nepřímo. Jako nejvhodnější se jeví BDD pro CF [19]. Jejich nevýhodou je to, že obsahují promíchaně vstupní a výstupní proměnné a mají větší výšku než MTBDD, což je nevýhodou při hledání optimálního uspořádání proměnných. Taktéž získání LUT kaskád z BDD pro CF je nepřímé a náročnější, protože po odejmutí několika proměnných z vrcholu diagramu se musí celý BDD pro CF rekonstruovat [19]. Navíc celý postup je téměř nemožné paralelizovat, protože se výhradně pracuje nad grafy a stěží se hledají nezávislé podproblémy, u kterých by šlo paralelně překrýt jejich zpracování.

Efektivní přístupem k získání MTBDD je nesestavovat MTBDD přímo a pak provádět jeho optimalizaci globálně, ale získávat ho postupně po jednotlivých proměnných již lokálně optimalizovaný, což by mělo vést na celkově suboptimální MTBDD [6]. Důležitým aspektem tohoto přístupu je heuristika pro výběr proměnných při postupném sestavování. Ve stávající metodě [6] se provádí syntéza zdola nahoru, tj. od koncových uzlů ke kořenu MTBDD, protože je to výhodnější u použitého dekompozičního postupu. Použitá heuristika totiž prochází všechny možné kombinace nepoužitých proměnných a se vzrůstajícím počtem naráz uvažovaných proměnných vzrůstá velikost prohledávaného prostoru kombinatoricky. Proto se jeví jako nejschůdnější prohledávání po jedné proměnné. Dostaneme tak sice dlouhé kaskády, ale kratší lze snadno získat sloučením více buněk do jedné, a máme tak možnost volby kompromisu mezi výkonem

a paměťovou náročností výsledné kaskády. Výhodou tohoto přístupu je, že prohledávání prostoru se dá snadno paralelizovat, protože se jedná o nezávislé úlohy. Zásadním nedostatkem metody [6] je, že nepodporuje zpracování neúplně zadaných vícevýstupových funkcí.

Cílem této disertační práce je navrhnout metodu syntézy LUT kaskád postupným sestavováním suboptimálního MTBDD částečně a neúplně zadaných vícevýstupových funkcí, která vychází z [6]. Hlavní myšlenkou je použití dekompoziční metody booleovských funkcí zadaných krychlemi [4]. Výhodou této dekompoziční metody je použití kompaktní reprezentace funkcí ve formě krychlí (formát Espresso PLA) na vstupu i výstupu metody. Implementace dekompoziční metody [4] je poměrně jednoduchá, protože se redukuje na dvě hlavní procedury: práci s pokrývkami (bitově orientované instrukce nad proudem dat) a hledání obarvení grafu (NP-úplný problém, ale existují rychlé heuristické postupy [12]).

4.1 Cíle disertační práce

1. Navrhnout metodu pro souběžnou syntézu MTBDD/LUT kaskád neúplně zadaných vícevýstupových funkcí založenou na iterativní dekompozici a lokální heuristice výběru proměnných pro minimalizaci velikosti/šířky MTBDD/LUT kaskády.
2. Navrženou metodu implementovat ve formě programu (sady programů) s přihlédnutím k možné paralelizaci na SMP systémech.
3. Navrženou metodu a program ověřit na sadě standardních benchmarkových obvodů a výsledky porovnat se stávajícími metodami.
4. Navrhnout a implementovat platformu pro realizaci LUT kaskád v hardware, software a firmware.
5. Implementovat vybrané aplikačně specifické systémy na základě paradigmatu LUT kaskád a zhodnotit, které aplikace lze takto výhodněji implementovat.

4.2 Dosavadní výsledky

Byla navržena nová heuristická technika konstrukce suboptimální LUT kaskády [8], která vychází z [6], a jejímž základem je iterativní disjunktní dekompozice celočíselných funkcí. Hlavním přínosem této metody je to, že vícevýstupová funkce může být zadána ve formě neúplně zadané vstupní matice a tak efektivně reprezentovat řídké funkce s velkým počtem DC hodnot.

Pro usnadnění syntézy LUT kaskád byl implementován program HIDET (Heuristic Iterative DEcomposition Tool) [8]. HIDET provádí dekompozici nad vícevýstupovými funkcemi zadaných ve formě PLA matice (funkce typu *fr*). Bohužel stávající verze programu HIDET zpracovává jenom určitou podmnožinu funkcí typu *fr*, neumožňuje zpracovávat vícevýstupové booleovské funkce s překrývajícími se vstupními termy. Z tohoto důvodu nešlo spustit program HIDET nad standardní sadou benchmarků MCNC [21] a výsledky porovnat s ostatními metodami. Přesto bylo možné otestovat navrženou metodu na určitém typu v praxi použitelných funkcí, které se dají zadat bez překrývajících se vstupních termů. Mezi tyto funkce patří např. různé druhy arbitrážních obvodů [8] a alokátorů [7].

Nedostatek předchozí metody byl odstraněn aplikováním nově objevené dekompoziční metody [4] a byla představena vylepšená metoda [9], která zpracovává všechny neúplně zadané celočíselné funkce typu *fr*. Do budoucna se počítá s vylepšením metody rozšířením podpory o obecné *fr* funkce, tj. že výstupní term může být zadán formou krychle.

5 Závěr

Navržená metoda syntézy LUT kaskád vícevýstupových booleovských funkcí se ukazuje jako vhodná metoda pro syntézu kombinačních a sekvenčních obvodů s desítkami vstupně/výstupních a stavových proměnných. Arbitrážní obvody a alokátory, ale i jiné číslicové obvody často používané v praxi, mají relativně nízkou složitost, což umožňuje jejich úspornou kaskádní implementaci.

Poděkování

Tento výzkum byl uskutečněn díky finanční podpoře výzkumných grantů „Návrh a obvodová realizace zařízení pro automatické generování patentovatelných invencí“ GA102/07/0850, „Bezpečnost a zabezpečení aplikací sítí vestavěných systémů“ GA102/08/1429, „Matematické a inženýrské metody pro vývoj spolehlivých a bezpečných paralelních a distribuovaných počítačových systémů“ GD102/09/H042 a „Výzkum informačních technologií z hlediska bezpečnosti“ MSM0021630528.

Reference

[1] Ashar, P.; Malik, S.: Fast Functional Simulation Using Branching Programs. In *Proceedings of International Conference on Computer-Aided Design*, 1995, s. 408–412.

[2] Ashenhurst, R. L.: The Decomposition of Switching Functions. In *Proceedings of an International Symposium on the Theory of Switching*, 1959, s. 74–116.

[3] Bryant, R. E.: Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, ročník C-35, č. 8, 1986: s. 677–691.

[4] Brzozowski, J. A.; Luba, T.: Decomposition of Boolean Functions Specified by Cubes. Technická zpráva, University of Waterloo, 1997.

[5] Curtis, H. A.: *A New Approach to the Design of Switching Circuits*. Princeton, NJ, USA: Van Nostrand, 1962, 635 s.

[6] Dvořák, V.: LUT Cascade-Based Architectures for High Productivity Embedded Systems. *International Review on Computers and Software*, ročník 2, č. 4, 2007: s. 357–365, ISSN 1828-600X.

[7] Dvořák, V.; Mikušek, P.: LUT Cascade-Based Implementations of Allocators. In *Proceedings of the 25th Convention of EEE in Israel*, New York, NY, USA: IEEE Computer Society, 2008, ISBN 978-1-4244-2482-5, s. 85–89.

[8] Mikušek, P.; Dvořák, V.: On Lookup Table Cascade-Based Realizations of Arbiters. In *11th EUROMICRO Conference on Digital System Design DSD 2008*, IEEE Computer Society, 2008, ISBN 978-0-7695-3277-6, s. 795–802.

[9] Mikušek, P.; Dvořák, V.: Heuristic Synthesis of Multi-Terminal BDDs Based on Local Width/Cost Minimization. In *12th EUROMICRO Conference on Digital System Design DSD 2009*, IEEE Computer Society, 2009, str. 4.

[10] Mishchenko, A.; Sasao, T.: Logic Synthesis of LUT Cascades with Limited Rails: A Direct Implementatin of Multi-Output Function. *IEICE Technical Report*, ročník 102, č. 479, 2002: s. 103–108, ISSN 0913-5685.

[11] Nakamura, K.; Sasao, T.; Matsuura, M.; aj.: Programmable logic device with an 8-stage cascade of 64K-bit asynchronous SRAMs. In *Cool Chips VIII, IEEE Symposium on Low-Power and High-Speed Chips*, 2005, str. 1.

[12] Perkowski, M.; Malvi, R.; Grygiel, S.; aj.: Graph coloring algorithms for fast evaluation of Curtis decompositions. In *Proceedings of the 36th ACM/IEEE conference on Design automation*, Ney York, NY, USA: ACM, 1999, ISBN 1-58133-109-7, s. 225–230.

[13] Roth, J. P.; Karp, R. M.: Minimization over Boolean Graphs. *IBM Journal of Research and Development*, ročník 6, 1962: s. 227–238.

[14] Rudell, R.: Dynamic variable ordering for ordered binary decision diagrams. In *ICCAD '93: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, Los Alamitos, CA, USA: IEEE Computer Society Press, 1993, ISBN 0-8186-4490-7, s. 42–47.

[15] Sasao, T.: Compact SOP representations for multiple-output functions: An encoding method using multiple-valued logic. In *31th International Symposium on Multiple-Valued Logic*, 2001, s. 207–212.

[16] Sasao, T.: Design Methods for Multi-Rail Cascades. In *International Workshop on Boolean Problems*, 2002, s. 123–132.

[17] Sasao, T.; Iguchi, Y.; Matsuura, M.: Comparison of Decision Diagrams for Multiple-Output Logic Functions. In *International Workshop on Logic and Synthesis*, 2002, s. 379–384.

[18] Sasao, T.; Matsuura, M.: A Method to Decompose Multiple-Output Logic Functions. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, New York, NY, USA: ACM, 2004, ISBN 1-58113-828-8, s. 428–433.

[19] Sasao, T.; Matsuura, M.: BDD Representation for Incompletely Specified Multiple-Output Logic Functions and Its Applications to Functional Decomposition. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, New York, NY, USA: ACM, 2005, ISBN 1-59593-058-2, s. 373–378.

[20] Sasao, T.; Matsuura, M.; Iguchi, Y.: A Cascade Realization of Multiple-Output Function for Reconfigurable Hardware. In *International Workshop on Logic and Synthesis*, 2001, s. 12–15.

[21] Yang, S.: Logic Synthesis and Optimization Benchmark User Guide Version 3.0. In *MCNC International Workshop on Logic Synthesis*, 1991.

[22] Yoeli, M.: The Synthesis of Multivalued Cellular Cascades. *IEEE Transactions on Computers*, ročník 19, č. 11, 1970: s. 1089–1090, ISSN 0018-9340.