Fakulta informačních technologií, VUT v Brně
Ústav počítačových systémů

Generování grafů celulárními automaty

(Projekt EVD)

Michal Bidlo, 2005

Obsah

1 Úvod

Biologií inspirované algoritmy se v poslední době stále více uplatňují při řešení mnoha komplexních úloh v různých oblastech vědy. Evoluční návrh patří k relativně mladému a rozvíjejícímu se odvětví, jenž nám dosud umožnilo získat mnoho zajímavých výsledků. Jako příklad uveďme kreativní evoluční návrh nebo nebo formy umělého života [7], číslicové obvody [8], neuronové sítě [9], analogové elektronické obvody nebo počítačové programy [10]. Přestože bylo aplikováno několik odlišných metod evolučních algoritmů, bylo možné takto získat pouze relativně jednoduchá řešení.

Problém škálovatelnosti patří pravděpodobně k nejvýznamnějším aspektům, které ztěžují proces evolučního návrhu s využitím současných technologií. Se zvyšující se složitostí cílového systému (např. nárůst počtu vstupních a výstupních veličin nebo počet komponent potřebných k jeho implementaci) vzrůstá délka chromozomu reprezentujícího kandidátní řešení během evolučního procesu, což má za následek zvětšení vyhledávacího prostoru a je tudíž velmi náročné navrhnout efektivní prohledávací (návrhový) algoritmus.

Dosud bylo vyvinuto několik odlišných přístupů snažících se problém škálovatelnosti překonat. Tyto techniky je možné rozdělit do následujících tříd:

- evoluce na úrovni funkčních bloků (složitějších celků tvořících stavební kameny cílového systému), viz např. [11],
- inkrementální evoluce (viz např. Toressenova metoda *rozděl a panuj* [12]) a
- development (vývoj cílového systému z triviální instance daného problému podle určitých pravidel – inspirováno biologickými principy ontogeneze), viz např. [13, 14, 15].

Cílem tohoto projektu je návrh metody založené na vývoji celulárního automatu (CA) pro konstrukci systémů, které je možné reprezentovat pomocí grafů. Celulární automat je v tomto případě rozšířen o schopnost generování entit dané problémové domény, které tvoří stavební bloky cílového systému. Tato generativní činnost je řízena lokálními přechodovými pravidly popisujícími chování (vývoj) celulárního automatu a představuje postup konstrukce cílového systému. Kritériem úspěšnosti vytvořeného řešení bude v tomto případě funkčnost kombinačních logických obvodů, na než jsou vytvořené grafové struktury mapovány. Navržený přístup využívá jednorozměrných uniformních celulárních automatů jako prostředků pro realizaci vývojového procesu (development).

Text je členěn do následujících částí: Kapitola 2 obsahuje popis základních principů ontogeneze spolu s přehledem nejznámějších biologií inspirovaných technik a modelů, které se v oblasti počítačového inženýrství v současné době používají. Kapitola 3 uvádí formální definici celulárního automatu a poskytuje základní popis jeho činnosti. Charakteristika vlastní návrhové metody je uvedena v kapitole 4. Popis evolučního algoritmu použitého při návrhu je uveden v kapitole 5 a přehled nejzajímavějších experimentálních výsledků poskytuje kapitola 6. V kapitolách 7 a 8 je uvedena diskuze k dosaženým výsledkům, jejich zhodnocení a závěr.

2 Development

Ontogeneze (development) je biologický proces, během kterého dochází k vývoji zárodečné buňky (zygoty) ve vícebuněčný organismus. Základním principem developmentu

je stavba organismu a sebeorganizace. Podstatnou částí ontogeneze je proces konstrukce, který je dán souhrou mezi geny, proteiny, buňkami a prostředím buňky obklopujícím. Výsledkem tohoto procesu je vyspělý organismus. V průběhu vývoje zygoty můžeme pozorovat tyto základní fáze [6] (poznamenejme, že tyto dílčí procesy neprobíhají zcela odděleně, ale obvykle se překrývají):

- *zárodečné dělení* – zárodečná buňka (tzv. zygota) podstoupí v této fázi rapidní dělení, jehož výsledkem je utvoření shluku buněk (tzv. blastula). Nedochází zde však k žádnému nárůstu objemu jednotlivých buněk.
- *organizace buněk* – v této fázi dochází k prostorové a časové organizaci buněk v embryu. V prvním stupni této fáze je vytvořen prostorový koncept organismu. Během druhého stupně jsou vytvořeny základní zárodečné vrstvy (vnější, střední a vnitřní).
- *morfogeneze* – buněčné migrace, dochází k přemísťování buněk, utváří se základ vnitřností. Tento proces se nazývá gastrulace, z blastuly se utvoří tzv. gastrula.
- *odlišení buněk* – jednotlivé buňky získají vlastní strukturu a funkci, vzniknou různě typy buněk, např. nervové, kožní atd.
- *růst* – organismus nabývá na velikosti (rozmnožování buněk, růst buněčné hmoty).

V oboru počítačového inženýrství existuje mnoho technik, které jsou inspirovány právě biologickými procesy. Pravděpodobně nejznámější a nejrozšířenější jsou evoluční algoritmy. Principy ontogeneze bývají mnohdy využívány buď samostatně nebo v kombinaci s jinými biologií inspirovanými metodami. Jelikož je development ve skutečnosti velice komplikovaný proces, bývají jeho modely podstatně zjednodušeny, případně simulují pouze některé jeho části. V oblasti počítačového inženýrství je development primárně chápán jako mapovací proces mezi genotypem a fenotypem v evolučním algoritmu. Genotyp v takovém případě obsahuje *předpis* pro konstrukci cílového systému (fenotypu).

Mezi nejznámější biologií inspirované modely umožňující simulaci vývoje mnohobuněčných organismů patří celulární automaty [2] a L-systémy [1]. V této práci se zaměříme na aplikace celulárních automatů pro konstrukci systémů, které je možné reprezentovat pomocí grafů.

3 Celulárních automaty

Celulární automaty (CA) původně zavedli Ulam a von Neumann ve čtyřicátých letech minulého století jako formální model pro vyšetřování chování složitých systémů [2]. CA jsou dynamické systémy tvořené diskrétní soustavou buněk, z nichž každá se může nacházet v jednom stavu z konečné množiny stavů. Stavy buněk jsou aktualizovány synchronně v diskrétních časových krocích v závislosti na lokálním přechodovém pravidle [3].

Prakticky se používá 1-, 2- nebo 3-dimensionální uspořádání buněk automatu. V tomto projektu se zaměříme na jednorozměrné uniformní celulární automaty, tj. CA s lineárním uspořádáním buněk, kde všechny buňky obsahují identické přechodové pravidlo.

Přechodové pravidlo obsažené v každé buňce je v podstatě konečný stavový automat, který je obvykle specifikován tabulkou přechodů (tzv. přechodová funkce). Tabulka přechodů udává následující stav dané buňky pro všechny možné kombinace stavů buněk v jejím sousedství.

Sousedství buňky sestává z buněk obklopujících danou buňku a této buňky samotné. V případě jednorozměrného CA tvoří buněčné sousedství r buněk obklopujících danou buňku z každé strany, včetně samotné vyšetřované buňky. Parametr r se nazývá *rádius*. V sousedství každé buňky v 1D CA je tedy $2r+1$ buněk. Uvažujeme-li konečný počet buněk celulárního automatu, je potřeba přesně vymezit sousedství pro dvě buňky nacházející se na okrajích celulární struktury, tzv. okrajové (hraniční) podmínky celulárního automatu. Mezi nejčastěji používané okrajové podmínky patří tzv. *konstantní podmínky*, kdy je pomyslným (chybějícím) sousedům buněk na okrajích CA přiřazen vybraný stav z konečné množiny stavů, který se v průběhu vývoje CA nemění (speciálním případem bývají *nulové okrajové podmínky*). Cyklické okrajové podmínky představují další variantu vymezení sousedství buněk na okraji CA, které přiřazují levému, resp. pravému sousedovi nejlevější, resp. nejpravější buňky 1D CA hodnotu aktuálního stavu nejpravější, resp. nejlevější buňky CA. V takovém případě je buněčný prostor celulárního automatu uspořádán do tvaru kružnice.

V následujícím odstavci uvedeme formální popis uniformního celulárního automatu, jehož definice je inspirována formalismy z [5]).

Definice 1: Pětici $A = (d, Q, N, \delta, c_0)$ nazveme *uniformním celulárním automatem*, kde

- $d \in \mathbf{N}$ je dimenze,
- $Q \neq \emptyset$ je konečná množina stavů,
- $N \subset \mathbf{Z}^d$ je sousedství,
- $\delta : Q^N \to Q$ je lokální přechodové pravidlo a
- $c_0 : \mathbf{Z}^d \to Q$ je přiřazení iniciálních stavů jednotlivým buňkám (počáteční konfigurace celulárního automatu).

Definici 1 nyní zůžíme na popis jednorozměrného uniformního celulárního automatu s konečným počtem buněk, kterého využijeme k získání experimentálních výsledků dále v této práci.

Definice 2: *Jednorozměrný uniformní celulární automat s konečným počtem buněk* je osmice $A = (d, Q, N, \delta, z, b_1, b_2, c_0)$, kde

- dimenze $d = 1$ a prvky Q, N, δ mají stejný význam jako v definici 1,
- $z \in \mathbf{N}$ udává počet buněk,
- $b_1, b_2 \in Q$ jsou hodnoty okrajových podmínek pro nejlevější a nejpravější buňku celulárního automatu,
- c_0 je počáteční konfigurace přiřazující každé buňce z $C = \{1, 2, \ldots, z\}$ nějaký stav z Q, tj. $c_o : C \to Q$.

Pro účely tohoto projektu uvažujeme jednotné sousedství N ve tvaru $N = \{-1, 0, 1\}$, což znamená, že následující stav každé buňky závisí na aktuálním stavu této buňky a na stavech jejích dvou nejbližších sousedů. Za těchto podmínek lze formulovat globální přechodovou funkci $G : Q^C \to Q^C$ ve tvaru

$$G(c_t) = \begin{cases} \delta(b_1, c(1), c(2)), & i = 1, \\ \delta(c(i-1), c(i), c(i+1)), & i = 2, \ldots, z-1, \\ \delta(c(z-1), c(z), b_2), & i = z, \end{cases}$$

kde δ je lokální přechodové pravidlo. Buňky CA jsou identifikovány pomocí indexu $i = 1, 2, \ldots, z$. Globální přechodová funkce definuje posloupnost konfigurací celulárního automatu $c_0, c_1, c_2 \ldots$ tak, že $c_{t+1} = G(c_t)$ pro $t \geq 0$. Tato sekvence reprezentuje *vývoj* (výpočet) celulárního automatu.

4 Charakteristika návrhové metody

Cílem projektu je návrh metody umožňující generování grafových struktur s využitím celulárních automatů. Celulární automaty (CA) představují jeden z možných způsobů realizace vývoje (developmentu) daného systému. Důraz je kladen na aplikace evolučního návrhu systémů, které lze v tomto případě reprezentovat pomocí grafů. Typickými třídami problémů, jejichž entity lze reprezentovat grafy, jsou například logické obvody nebo neuronové sítě. Za účelem nalezení vhodných pravidel celulárního automatu pro konstrukci cílového systému aplikujeme genetický algoritmus.

4.1 Formalizace problému

Uvažujme jednorozměrný uniformní celulární automat odpovídající definici 2. Navrhněme vhodné rozšíření CA, jehož smyslem bude schopnost buněk tohoto CA generovat určitou strukturu dané problémové domény po každém kroku jeho vývoje. Nechť proces vývoje takto modifikovaného CA představuje postup konstrukce cílového systému, jehož stavebními bloky budou entity generované jednotlivými buňkami CA.

Z uvedeného popisu je zřejmé, že se jedná z velké části o experimentální práci. Klíčovou roli v tomto procesu hraje především volba vhodné třídy problémů, na které bude navržená metoda aplikovatelná a reprezentace entit dané problémové domény s ohledem na efektivní činnost evolučního algoritmu, charakter použité techniky pro development a řešitelnost problémů zvolené třídy s použitím navrženého přístupu. Jak již bylo uvedeno, zaměříme se v této práci na návrh kombinačních logických obvodů.

Definice 3 představuje formální popis ceulárního automatu s generativními schopnostmi, jenž je využit pro získání experimentálních výsledků. Pro jednoduchost budeme uvažovat sousedství ve tvaru $N = \{-1, 0, 1\}$ uvedené v předcházející kapitole.

Definice 3: *Celulární automat s generativními schopnostmi* je algebraická struktura $A = (d, Q, N, \Sigma, \delta, \lambda, z, b_1, b_2, c_0)$, kde

- dimenze $d = 1$ a prvky Q, N, δ mají stejný význam jako v definici 1,
- $\Sigma \neq \emptyset$ je konečná množina symbolů (abeceda),
- $\lambda : Q^N \to \Sigma$ je funkce generující pro každé přechodové pravidlo tvaru $q_{-1}q_0q_{+1} \to q\prime$ nějaký symbol ze Σ,

- $z \in \mathbf{N}$ udává počet buněk celulárního automatu,
- $b_1, b_2 \in Q$ jsou hodnoty okrajových podmínek pro nejlevější a nejpravější buňku celulárního automatu,
- c_0 je počáteční konfigurace přiřazující každé buňce z $C = \{1, 2, \ldots, z\}$ nějaký stav z Q, tj. $c_o : C \rightarrow Q$.

Uvedená definice není omezena na žádnou konkrétní doménu. Volba vhodných struktur (symbolů) abecedy Σ a generativní činnost CA v podobě symbolů této abecedy umožňuje libovolnou interpretaci entit vytvořených vývojem celulárního automatu. Následující podkapitola je zaměřena na podrobný popis aplikace generativních celulárních automatů pro návrh kombinačních logických obvodů na úrovni hradel jako vzorové třídy systémů reprezentovatelných pomocí grafů.

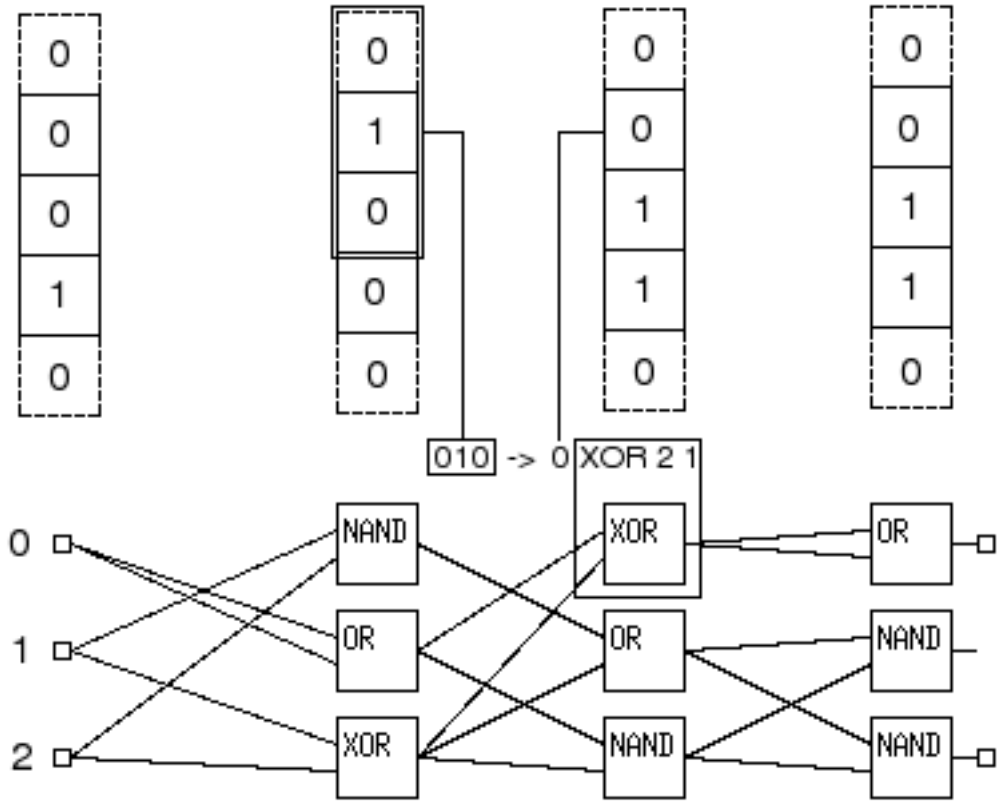
4.2 Návrh kombinačních logických obvodů

Dosud bylo uvedeno několik metod, pomocí kterých je možné realizovat evoluční návrh číslicových obvodů. Mezi nejznámější patří například tzv. *kartézské genetické programování (CGP)* [16], které pracuje s maticí programovatelných elementů o předem stanovených rozměrech, která je považována za indexovaný graf reprezentující například právě logický obvod. Evoluční algoritmus je v tomto případě využit pro nalezení vhodné konfigurace prvků této matice (uzlů grafu) a jejich propojení k realizaci požadované funkce. Typické příklady obvodů nalezených pomocí této metody jsou uvedeny v [8].

Později aplikoval Miller development založený na programu uvnitř buněk, jehož opakovaná aplikace umožňuje modifikaci stávajícího propojení a konfigurace buňky, případně její duplikaci [17].

Návrhová metoda, která tvoří jádro této práce, využívá development založený na vývoji celulárního automatu. Narozdíl od kartézského genetického programování však nepracuje přímo s grafovou reprezentací obvodu, ta je generována v závislosti na vývoji CA po jednotlivých (paralelně proveditelných) stupních obvodu. Každým vývojovým krokem celulárního automatu je generován právě jeden stupeň obvodu. Pravidla celulárního automatu zde tvoří předpis (program), podle kterého je výsledný obvod generován. Podobně jako v CGP jsou i zde vstupy a výstupy jednotlivých elementů obvodu identifikovány indexy.

Před spuštěním vývojového procesu je známa funkce, která bude navrženým obvodem implementována. Dále je stanoven počet buněk celulárního automatu, hodnoty okrajových podmínek a počet iterací (kroků) CA, po kterém bude navržený obvod vyhodnocen. Činnost algoritmu je následující: Celulární automat je inicializován do počáteční konfigurace c_0. Provede se jeden krok výpočtu CA podle daného přechodového pravidla a na základě dílčích přechodových pravidel aplikovaných na jednotlivé buňky jsou vygenerovány příslušné obvodové struktury (v tomto případě logická hradla). Každé vygenerované hradlo má stanoveno indexy pozic hlavních vstupů obvodu, případně hradel vygenerovaných v předcházejícím kroku CA, ke kterým jsou připojeny vstupy tohoto hradla. Z pohledu definice 3 je hradlo společně s konkrétními indexy jeho vstupních signálů považováno za jeden symbol, tj. například hradla [AND 0, 1] a [AND 1, 2] představují dva různé symboly. Tento postup se opakuje, dokud není dosaženo stanoveného počtu kroků CA. Takto vytvořený výsledný obvod je vyhodnocen v simulátoru. Názorná ukázka postupu konstrukce obvodu s využitím tohoto přístupu je na obrázku 1, který představuje příklad konstrukce úplné sčítačky. Pravidla celulárního automatu použitého v příkladu jsou shrnuty v tabulce 1.



Obrázek 1: Princip konstrukce obvodu pomocí generativního celulárního automatu. Horní polovina obrázku znázorňuje konfiguraci CA v každém kroku jeho vývoje. Čárkovaně označené buňky reprezentují okrajové podmínky CA. Ohraničená část CA v druhém kroku vývoje označuje příklad generace komponenty: dle tabulky 1 je vybráno pravidlo č. 2, jehož aplikace má za následek přechod příslušné buňky ze stavu 1 do stavu 0 a vygenerování logického hradla XOR, jehož vstupy jsou připojeny na pozice 1 a 2 předešlého stupně obvodu (zobrazeného ve spodní polovině obrázku). Stejným způsobem probíhá konstrukce ostatních částí obvodu.

Číslo pravidla	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Schéma pravidla v CA	000	001	010	011	100	101	110	111
Nový stav buňky	1	0	0	1	1	1	1	1
Generovaný symbol	NAND 1 2	OR 0 0	XOR 2 1	NAND 1 2	OR 2 0	OR 2 1	NAND 1 2	NAND 2 1

Tabulka 1: Pravidla celulárního automatu použitého v příkladu na obrázku 1. Pravidla pro výpočet následujícího stavu buňky jsou označena hodnotou v desítkové soustavě, která odpovídá hodnotě šísla v soustavě o základu $|Q|$ sestaveného z cifer (symbolů) stavů buněk v definovaném sousedství.

5 Evoluční algoritmus

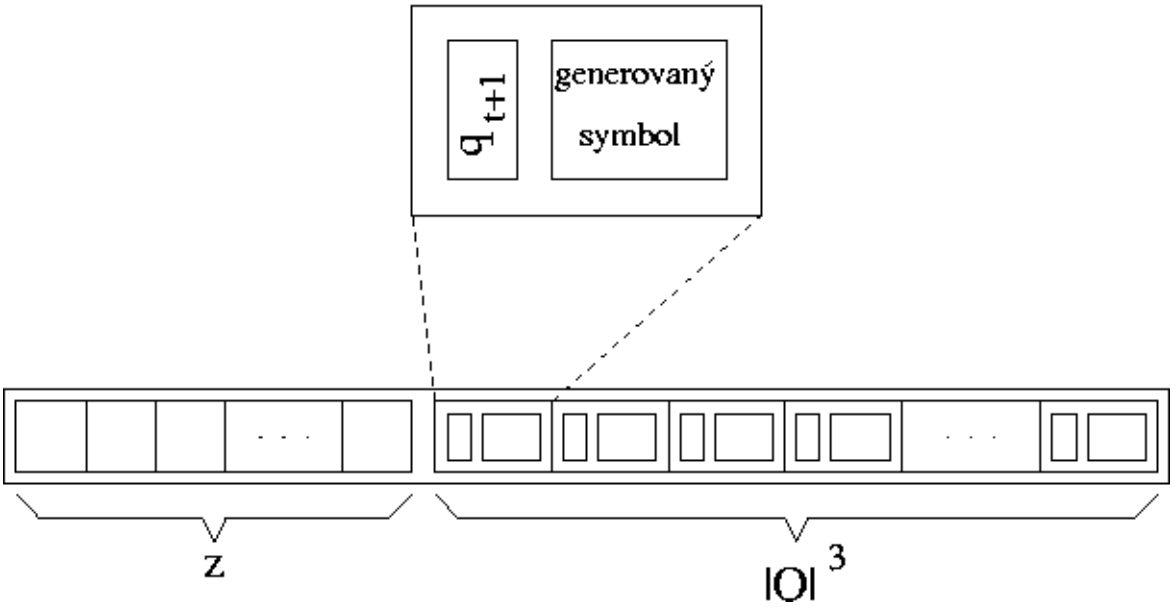
V této sekci uvedeme popis evolučního algoritmu použitého pro získání výsledků s využitím návrhové metody představené v předcházející kapitole a přehled vybraných obvodů, které byly pomocí této techniky vytvořeny.

Vývoj (výpočet) celulárního automatu probíhá podle předem známých pravidel počínaje iniciálním stavem (konfigurací) synchronně v diskrétních časových krocích, přičemž každá buňka automatu generuje přechodem do následujícího stavu jistou strukturu dané problémové domény (v tomto případě logické hradlo). Jelikož přechody buněk z aktuálních do následujících stavů probíhají paralelně, lze návrhový proces přizpůsobit tak, že je po každém vývojovém kroku CA vytvořen právě jeden stupeň obvodu (tj. soustava obvodových elementů – logických hradel, jejichž funkci lze provést paralelně). Volbou vhodné počáteční konfigurace a přechodových pravidel je třeba zajistit takové chování CA, aby byl výsledkem jeho generativní činnosti po předem stanoveném počtu vývojových kroků funkční kombinační logický obvod. Nalezení těchto nastavení celulárního automatu je předmětem evolučního algoritmu. Pro tento účel byl použit genetický algoritmus.

Součástí evolučního algoritmu je vhodná reprezentace jendotlivých řešení. Řešením jsou v tomto případě myšleny počáteční konfigurace a pravidla celulárního automatu. Sousedství každé buňky je ve všech experimentech této práce tvořeno danou buňkou a jejími dvěma bezprostředními sousedy. Uvažujeme-li $|Q|$ možných stavů buňky, je možné zakódovat jednotlivá pravidla (udávající přechod buňky do následujícího stavu v závislosti na libovolné kombinaci stavů buněk v definovaném sousedství) indexy vypočtenými jako hodnota čísla v soustavě o základu $|Q|$ složeného z cifer (symbolů) jednotlivých stavů buněk v sousedství. Prostřednictvím těchto indexů (levá strana přechodového pravidla) je pak možné odkazovat se jednoduše na pravé strany přechodových pravidel pro všechny možné kombinace stavů buněk v sousedství. V chromozomu genetického algoritmu jsou tak fyzicky uchovány pouze pravé strany přechodových pravidel, jejich levé strany jsou jednoznačně určeny polohou (indexem) v poli, pomocí kterého je chromozom implementován. Uvažujme například binární množinu stavů $Q = \{0, 1\}$ a přechodové pravidlo $110 \rightarrow 0$. Vyšetřovaná buňka přejde v případě této kombinace stavů buněk v sousedství ze stavu 1 do stavu 0. Přechodové pravidlo pro tuto kombinaci bude mít index 6, jelikož binární kombinace stavů 110 odpovídá vyjádřením v desítkové soustavě hodnotě 6. Na 6. pozici v chromozomu se tedy bude nacházet pravá strana pravidla, tj. stav 0. Úplné schéma chromozomu zahrnující počáteční konfiguraci celulárního automatu a strukturu generovanou po aplikaci každého pravidla je na obrázku 2.

Fitness funkce nejprve inicializuje celulární automat počáteční konfigurací z chromozomu, dále provede předepsaný počet kroků CA s použitím pravidel uvedených v chromozomu a nakonec vyhodnotí funkčnost výsledného obvodu. Kritériem úspěšnosti je počet korektně vypočtených bitů výsledku pro všechny možné vstupní posloupnosti obvodu.

Genetický algoritmus pracoval při experimentech s populací čítající 100 jedinců. Pravděpodobnost mutace byla nastavena na 98%, operátor křížení nebyl aplikován. Selekční operátor byl implementován jako turnajový výběr o základu 4.



Obrázek 2: Schéma chromozomu genetického algoritmu: první šást obsahuje počáteční konfiguraci CA (z buněk), druhá část pak následující stav buňky pro všechny možné kombinace stavů buněk v definovaném sousedství (v tomto případě $|Q|^3$ možností) včetně symbolů generovaných každým pravidlem.

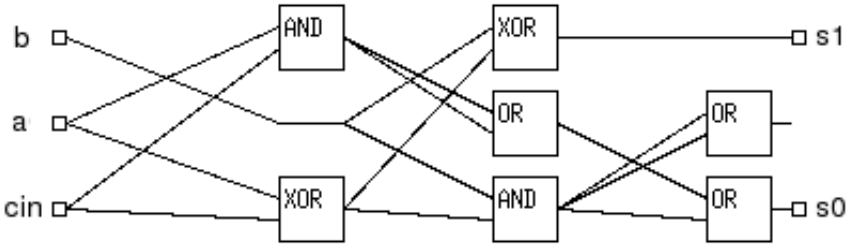
6 Experimentální výsledky

Uvedená metoda umožňuje s využitím výše popsaného genetického algoritmu návrh obvodů mnoha tříd (testováno především na aritmetických obvodech). Složitost obvodů, které se podařilo úspešně vytvořit, závisí na typu obvodu. Počet vstupů takto získaných obvodů se pohybuje od čtyř (násobičky) až po čtrnáct (paritní obvody).

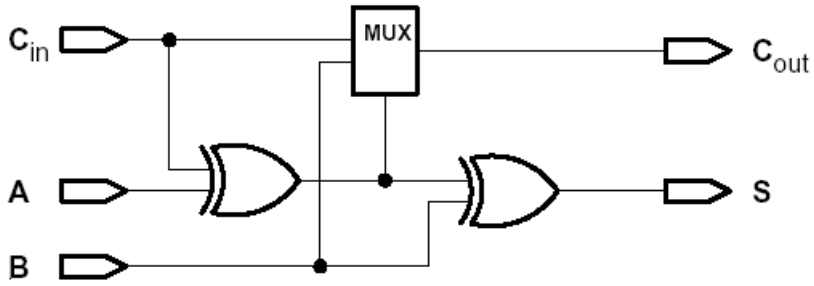
Pro každou třídu obvodů bylo provedeno 1000 nezávislých experimentů s různě nastavenými parametry vývojového procesu (bylo-li to možné). Parametry vývoje zahrnují počet možných stavů buňky celulárního automatu, počet kroků jeho výpočtu a počet buněk. Z hlediska snadné simulace obvodu a jednotných pravidel celulárního automatu ve všech krocích vývoje byl ve všech experimentech počet buněk CA roven počtu primárních vstupů konstruovaného obvodu.

V souvislosti s parametry vývojového procesu se mění délka chromozomu potřebná pro zakódování konfiguračních informací CA: Zvýšení poštu vstupů obvodu má za následek nárůst délky chromozomu o odpovídající počet prvků umístěných na počátku chromozomu, které určují iniciální konfiguraci CA. U obvodů s vyšším stupněm různorodosti vnitřní struktury má však tato modifikace nepříznivý dopad na úspěšnost evoluce, jelikož pro takto složité obvody se nepodařilo nalézt vhodná pravidla CA, která by umožňovala jejich konstrukci. Nejvíce patrné bylo omezení počtu vstupů u sčítaček a násobiček. Rozšíření množiny pravidel CA a tím i zvýšení pravděpodobnosti pro nalezení řešení lze docílit zvýšením počtu stavů celulárního automatu. V závislosti na počtu stavů však kubicky roste délka chromozomu (velikost vyhledávacího prostoru GA) a stává se tak téměř nemožné nalezení funkčního řešení s využitím současných výpočetních prostředků.

Uvažujme jednoduchý příklad generativního celulárního automatu obsahující tři buňky, sousedství v podobě zavedené definicí 3 a dva možné stavy buněk. Pro úplný popis chování tohoto automatu je potřeba osm pravidel. Dále uvažujme tři dvouvstupová logická hradla, která mohou být automatem generována. Jejich každý vstup je



Obrázek 3: Jednobitová sčítačka s přenosem



Obrázek 4: Jednobitová sčítačka s přenosem navržená v [8]

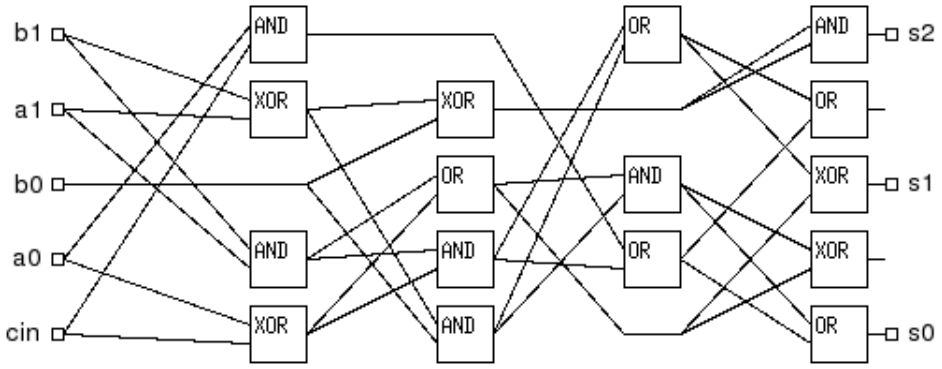
určen indexem 0, 1 nebo 2. Chromozom obsahuje ve své první části tři prvky určující počáteční konfiguraci CA, druhá část obsahuje pravé strany osmi přechodových pravidel, kde každá udává následující stav buňky, logickou funkci a indexy jejích dvou vstupů (tedy čtyři prvky na každé pravidlo). Celková délka chromozomu je tedy 35 prvků, z nichž 11 z nich může nabývat dvou různých symbolů (stavy 0 a 1), zbylých 24 prvků může nabývat tří různých symbolů (prvky udávající jednu ze tří logických funkcí a prvky udávající index vstupu 0, 1 nebo 2). Prostor genotypů pro tento jednoduchý případ tedy obsahuje celkem $2^{11} + 3^{24} \approx 3 \cdot 10^{79}$ prvků, což je v dnešní době hrubou silou v rozumném čase neprohledatelné.

V následujících odstavcích jsou uvedeny ukázky vybraných obvodů různých tříd, které se podařilo navrhnout s využitím popsaného přístupu. Každý obvod má označeny hlavní vstupy a výstupy malým čtverečkem s případným popisem významu daného bitu. Výstupní bity neobsahující označující čtvereček jsou bezvýznamné a případná hradla k nim připojená mohou být odstraněna bez vlivu na funkci obvodu. Rovněž je možné hradla AND a OR se stejnými vstupy nahradit spojem, hradla NAND a NOR se stejnými vstupy lze nahradit negací.

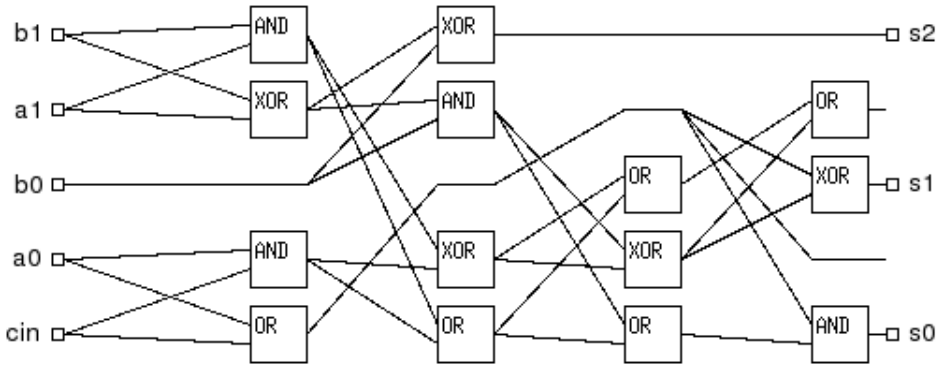
6.1 Sčítačky

Nejjednodušším případem sčítaček, se kterým bylo při experimentech počítáno, byly jednobitové sčítačky s přenosem. Obrázek 3 ukazuje typickou strukturu tohoto aritmetického obvodu tak, jak ji navrhnul evoluční algoritmus s použitím výše uvedeného přístupu. Při bližším pohledu na logickou strukturu této sčítačky zjistíme, že je velice podobná obvodu vytvořenému v Millerově práci [8] (viz obrázek 4). Náš obvod se liší pouze absencí komponenty multiplexoru, který je v podstatě implementován na úrovni logických hradel AND a OR v obvodu obsažených.

Obrázky 5 a 6 ukazují příklady dvoubitových sčítaček s přenosem vytvořených výše popsanou metodou. V kategorii sčítašek jsou to nejsložitější obvody, které se podařilo navrhnout. Množina stavů celulárního automatu obsahovala 3 stavy, z 1000 nezávislých



Obrázek 5: Dvoubitová sčítačka s přenosem



Obrázek 6: Jiná varianta dvoubitové sčítačky s přenosem

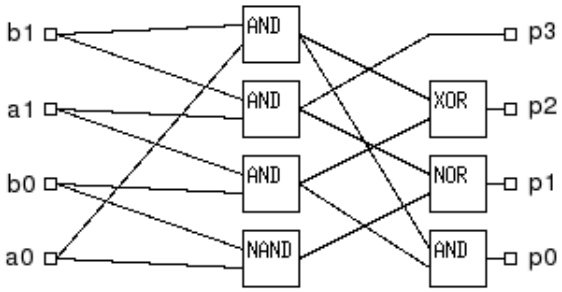
experimentů bylo úspěšně nalezeno 20 plně funkčních řešení. Při dvouprvkové množině stavů CA bylo z 1000 nezávislých experimentů nalezeno pouze jediné řešení. Příčinou neúspěchu při návrhu vícebitových sčítaček je pravděpodobně již zmíněný kubický nárůst délky chromozomu se zvyšujícím se počtem možných stavů buňek, který je při návrhu složitějších obvodů nezbytný.

6.2 Násobičky

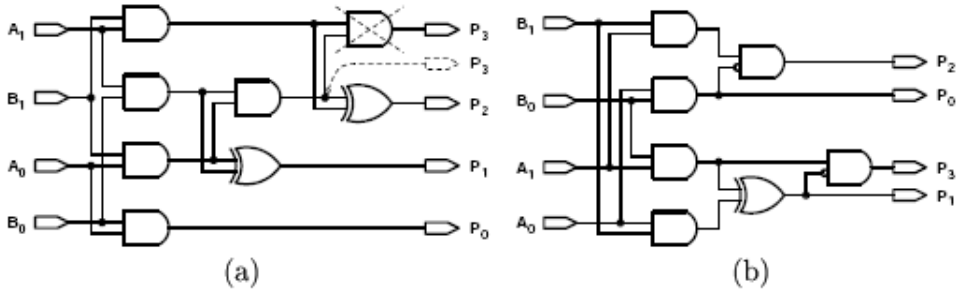
Vícebitové kombinační násobičky patří mezi obvody s relativně komplikovanou logickou strukturou. Důkazem toho je fakt, že narozdíl od jiných obvodů (např. řadicí sítě nebo sčítačky, viz [18]) se v případě násobiček nepodařilo nalézt s využitím evolučních technik obecný konstruktor. S využitím výše uvedeného přístupu, který sice není orientován na návrh obecných konstrucktorů, bylo nalezeno několik řešení kombinačních násobiček 2x2 bity.

Obrázek 7 ukazuje násobičku tak, jak ji navrhnul evoluční algoritmus s využitím vývojové metody popsané v kapitole 4. Díky povolení užití hradel NAND a NOR generovanými spolu s ostatními hradly celulárním automatem má obvod o jednotku nižší zpoždění v porovnání s nejlepší známou konvenční, respektive evolučně navrženou násobičkou v Millerově práci [8] (viz obr. 8).

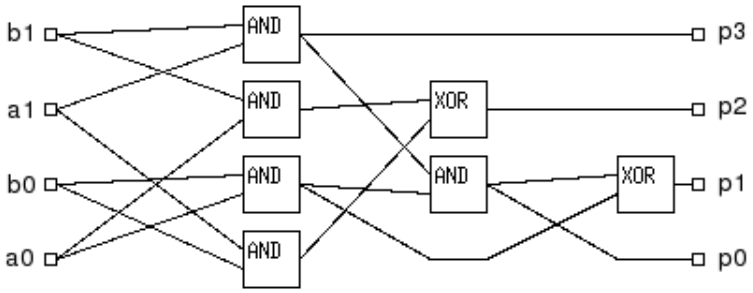
Omezíme-li implementační prostředky pouze na základní logická gradla (AND, OR, XOR a NOT), je pro plně funkční obvod potřeba zpoždění 3τ. Bližší pohled na logickou strukturu této násobičky (obr. 9) odhalí vysoký stupeň podobnosti s nejlepším konvenčním řešením uvedeném na obrázku 8a.



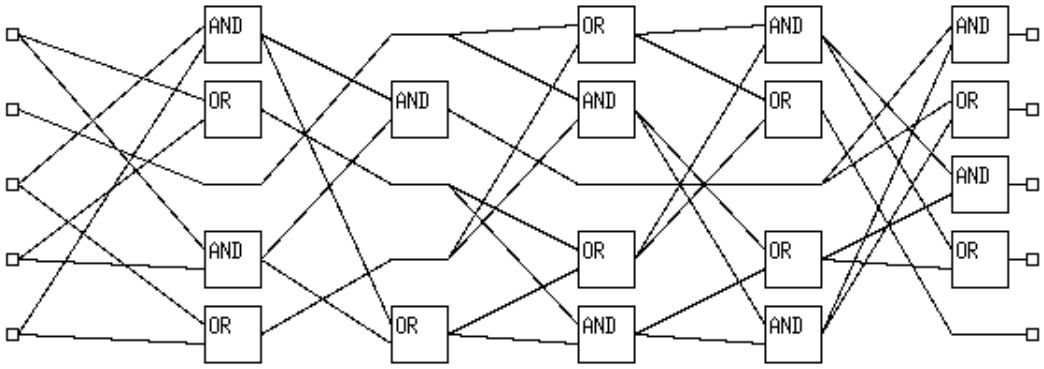
Obrázek 7: Dvoubitová násobička s hodnotou zpoždění 2τ (kromě základních logických hradel využívá též NAND a NOR)



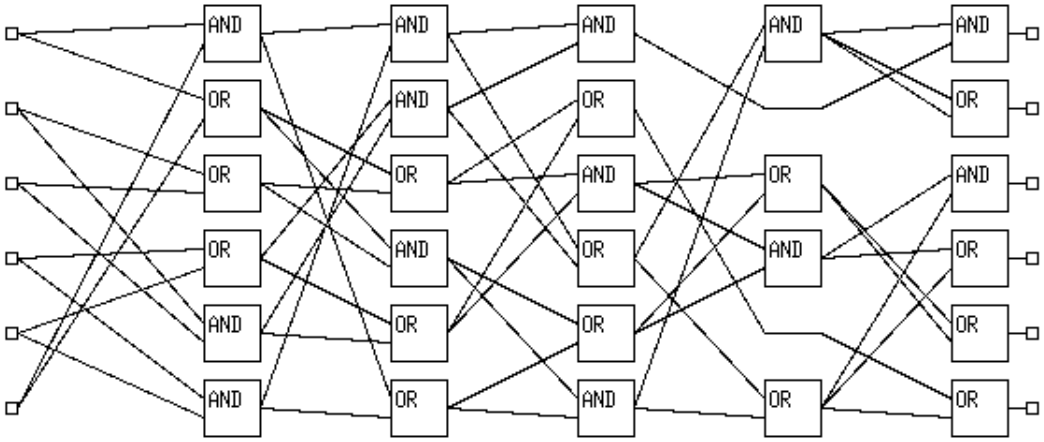
Obrázek 8: Dvoubitové násobičky [8]: (a) nejlepší konvenční řešení, (b) řešení navržené evoluční technikou



Obrázek 9: Dvoubitová násobička implementovaná pouze s využitím základních hradel



Obrázek 10: Řadicí síť o pěti vstupech



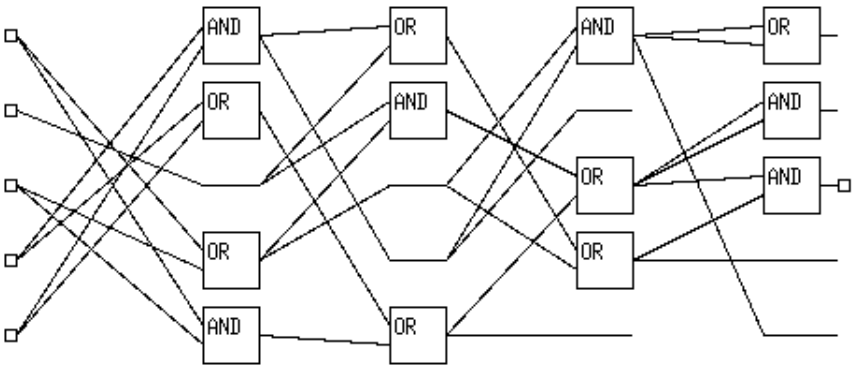
Obrázek 11: Řadicí síť o šesti vstupech

6.3 Řadicí sítě

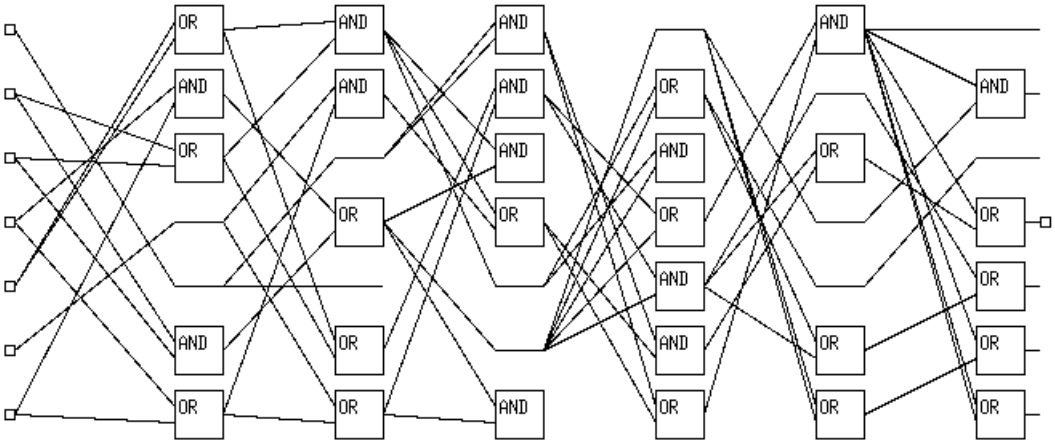
Narozdíl od obvodů představených v předcházejících odstavcích se podařilo ve třídě řadicích sítí navrhnout poměrně složité struktury. Obrázek 10 ukazuje příklad nejlepší pětivstupové řadicí sítě, která byla pomocí celulárního automatu zkonstruována. Pro usnadnění bávrhu byla uvažována pouze hradla AND a OR, která se obvykle pro výpočet minima a maxima používají. Dvojice těchto hradel se společnými dvěma vstupy mohou být považovány za komparátor a je tak možné objektivní porovnání s již existujícími obvody. Dle reference nejlepších známých řadicích sítí [19] je obvod na obrázku 10 kvalitativně shodný (počet komparátorů a zpoždění obvodu) s nejlepším řešením. Z tisíce nezávisle provedených experimentů bylo nalezeno 63 platných řešení. Příklad obvodu nalezeného pro 6 vstupů je na obrázku 11. V tomto případě je shoda s nejlepším řešením uvedeném v [19] pouze u zpoždění sítě, počet párů AND, OR je o jednotku vyšší. Zde byla úspěšnost nižší, pouze 23 platných exemplářů z tisíce nezávislých procesů.

6.4 Mediánové obvody

Mediánový obvod lze zkonstruovat z platné řadicí sítě o daném počtu vstupů odebráním některých komponent, což je dáno tím, že se u mediánové sítě zajímame pouze o prostřední bit seřazené posloupnosti. Tato skutečnost je patrná i z obvodů, které se v této třídě podařilo navrhnout. Obrázek 12 ukazuje nejlepší pětivstupový obvod pro výpočet mediánu posloupnosti, který má (narozdíl od stejně velké řadicí sítě) jak nižší



Obrázek 12: Pětivstupový mediánový obvod



Obrázek 13: Sedmivstupový mediánový obvod

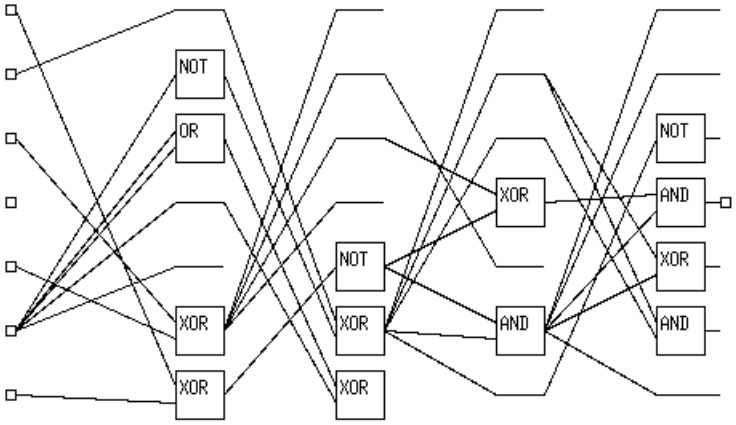
počet komponent AND, OR, tak i nižší hodnotu zpoždění. Jiným příkladem je sedmivstupová mediánová síť uvedená na obrázku 13. Pro pětivstupovou variantu obvodu byla úspešnost 798/1000, ve druhém případě pak 321/1000.

6.5 Paritní obvody

Experimenty ukázaly, že u této třídy obvodů nebyl problém navrhnout vícevstupovou strukturu. Se zvyšujícím se počtem vstupů narůstá čas potřebný pro nalezení řešení především kvůli nárůstu počtu testovacích posloupností. Logická stavba takto vytvořených obvodů není překvapující v porovnání s konvenční metodou konstrukce paritních obvodů. Výsledné obvody získané odstraněním redundantních hradel obsahují ve většině případů pouze hradla XOR uspořádaná do stromové struktury tak, jak je obvyklé u konvenčních paritních obvodů. Návrh paritních obvodů byl testován až do počtu vstupů 14, po této hodnotě již bylo testování obvodů časově velmi náročné.

6.6 Booleovská symetrie

Funkce booleovské symetrie vrací hodnotu '1' v případech, kdy je uspořádání binárních hodnot ve vstupním vektoru symetrické podle jeho středu, jinak je výsledek této funkce nulový. Obrázek 14 ukazuje příklad nejlepšího sedmivstupového obvodu booleovské symetrie, který byl s využitím celulárního automatu navržen. Jak je z obrázku patrné, algoritmus výpočtu neuvažuje prostřední bit, jelikož tento nemá na výslednou



Obrázek 14: Sedmivstupový obvod booleovské symetrie

hodnotu vliv. Po odstranění redundantních komponent obsahuje obvod celkem 8 hradel z výpočetně úplné množiny AND, OR, XOR a NOT. Experimenty potvrdily, že rozšířením této množiny o hradla NAND, NOR a XNOR lze vytvožit obvod, jehož zpoždění je o jednotku nižší než v případě na obrázku 14 (podobně jako například u násobičky).

7 Diskuze

Uvedená metoda evolučního návrhu využívá development založený na vývoji celulárního automatu. V průběhu každého vývojového kroku CA je každou jeho buňkou generována logická funkce (obecně symbol dané abecedy), která závisí na pravidle aplikovaném pro přechod dané buňky do následujícího stavu v závislosti na stavech buněk v jejím sousedství. Cílem práce byl návrh metody pro generování grafových struktur. Jako vzorová aplikační doména byla zvolena třída kombinačních logických obvodů, které lze pomocí grafů reprezentovat.

Jak plyne z předcházející kapitoly, lze s využitím tohoto přístupu navrhovat širokou škálu kombinačních obvodů. Úspěšnost návrhu konkrétní struktury závisí především na typu cílového obvodu. Pro obvody s vysokým stupněm různorodosti logické struktury, případně pro obvody s vyšším počtem významných výstupních bitů, byl návrh náročnější. Mezi takové obvody patří například sčítačky nebo násobičky. Ve třídě kombinačních děliček se dokonce nepodařilo nalézt ani jediný plně funkční obvod. Opačným případem jsou obvody s pouze jedním významným bitem výsledku (mediám, parita, booleovská symetrie), s jejichž rostoucí složitostí (počtem vstupů) se nejvíce projevovala hlavně časová náročnost testování obvodů vlivem rostoucího počtu testovacích vektorů. Výjimku členění složitosti obvodů v závislosti na úspěšnosti návrhu, případně schopnosti návrhové metody nalézt řešení pro vyšší počet vstupů obvodu, tvoří řadicí sítě, které mají počet významných výstupních bitů stejný jako počet vstupů. V případě řadicích sítí bylo úspěšně navrženo i několik šestivstupo-vých obvodů, které logicky vyžadují vyšší počet stupňů obvodu (zpoždění) a vykazují poměrně složitou vnitřní strukturu. Úspěch při návrhu těchto obvodů by mohl být zapříčiněn redukcí množiny hradel pouze na členy AND a OR, které se obvykle pro implementaci komparátorů řadicích sítí používají. Obvody s vyšším stupněm různorodosti (např. sčítačky a násobičky) však takovou redukci implementačních prostředků neumožňují, což může být příčinou neúspšchu při návrhu rozsáhlejších obvodů těchto tříd.

Ačkoliv bylo v některých případech dosaženo relativně vysoké složitosti vnitřní struktury obvodů, mohou být příčinou obtížného návrhu rozsáhlejších struktur také různá omezení vlastního modelu vývojového procesu. Jedním z nich je například skutečnost, že veškeré experimenty byly provedeny s využitím uniformního celulárního automatu. Z pohledu výpočetní síly jsou uniformní celulární automaty slabší než automaty neuniformní (viz např. [20]). V případě neuniformního CA se zachováním stávající architektury modelu vývoje by bylo nutné hledat pomocí evolučního algoritmu několik předpisů různých přechodových funkcí, což by vedlo k enormnímu nárůstu vyhledávacího prostoru. Otázkou v takovém případě zůstává, zda by uplatněním neuniformního přístupu byla zachována (alespoň přibližně) evolvovatelnost uvažovaných struktur. Dalším možným rozšířením by mohla být parametrizace počtu buněk celulárního automatu nezávisle na počtu vstupů cílového systému. V současné verzi vývojového modelu je počet vstupů obvodu vždy shodný s počtem buněk CA z důvodu jednoduchosti implementace přechodových pravidel. V neposlední řadě by mohly být omezujícím faktorem také zvolené entity (symboly) generované celulárním automatem, případně jejich interpretace. Velké množství těchto variant ovlivňujících chování CA a tím i proces vývoje (development) však komplikuje celý model a staví problematiku návrhu s využitím tohoto přístupu na úroveň čistě experimentální práce.

8 Závěr

V práci byl představen nový model pro tzv. development založený na vývoji jednorozměrného uniformního celulárního automatu. Tento celulární automat byl rozšířen o generativní schopnosti realizované při každém jeho vývojovém kroku k zajištění konstrukce entit dané problémové domény. Předmětem evolučního procesu byl návrh přechodových pravidel a počáteční konfigurace CA pro předem specifikovanou třídu problémů (v tomto případě kombinačních logických obvodů). Generativní schopnosti automatu nejsou omezeny na žádnou konkrétní doménu a mohou být přizpůsobeny pro konkrétní aplikaci. Entity generované celulárním automatem jsou pak chápány jako obecné symboly jisté abecedy, jejichž interpretace je závislá na povaze řešeného problému (viz definice 3).

S využitím vývojového modelu výše uvedených vlastností bylo úspěšně navrženo mnoho kombinačních logických obvodů z několika tříd, konkrétně: sčítačky (2-bitové se vstupním i výstupním přenosem), násobičky (2-bitové), mediánové sítě (až 7-vstupové), řadicí sítě (až 6-vstupové), obvody booleovské symetrie (až 7-vstupové) a paritní obvody (až 14-vstupové).

Výčet tříd úspěšne navržených obvodů dokazuje schopnost uvedeného přístupu řešit tento typ problémů (kombinační logické obvody). Možnými směry dalšího výzkumu mohou být například návrh polymorfních obvodů, neuronových sítí a obecně i entit jiných domén, jež lze reprezentovat grafovými strukturami, případně vhodně interpretovanými posloupnostmi generovaných symbolů (např. výzkum tohoto modelu v souvislosti s formálními jazyky, výpočetní silou apod.). Zcela odlišným směrem výzkumu by mohl být vývoj celulárních automatů za účelem studia tzv. emergentního chování, kdy dochází z různých počátečních konfigurací k vytvoření jedné určité struktury (konfigurace), případně chování generující různé struktury v závislosti na počáteční konfiguraci bez změny přechodových pravidel automatu. Jak již bylo uvedeno v předcházející kapitole, může být však stávající model vývoje v jistých směrech omezující a bude tedy nutné přikročit k jeho zobecnění či modifikaci.

Reference

[1] Lindenmayer, A.: Mathematical models for cellular interaction in development, Parts I and II. Journal of Theoretical Biology, 18:280–315, 1968

[2] von Neumann, J.: The Theory of Self-Reproducing Automata (edited by A. W. Burks). University of Illinois Press, 1966

[3] Wolfram, S.: Universality and Complexity in Cellular Automata. Physica D, Vol. 10, pp 1–35, 1948

[4] Wolpert, L.: The Principles of Development. Oxford University Press, 1998

[5] Sipper, M.: Evolution of Parallel Cellular Machines. Springer–Verlag Berlin Heidelberg, LNCS vol. 1194, p. 203, 1997

[6] Kumar, S., Bentley, P. J. (eds.): On Growth, Form and Computers. Elsevier Academic Press, 2003

[7] Bentley, P. J. (ed.): Evolutionary Design by Computers. Morgan Kaufmann Publisher, San Francisco, 1999

[8] Miller, J. F., Job, D.: Principles in the evolutionary design of digital circuits – part I. Genetic Programming and Evolvable Machines, 1(1), pp 8–35, 2000

[9] Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE, 87(9), pp 1423–1447, 1999

[10] Koza, J. R. et al.: Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publisher, San Francisco, 1999

[11] Murakawa, M. et al.: Evolvable Hardware at Function Level. In: Proc. of the Parallel Problem Solving from Nature IV, LNCS vol. 1141, pp 62–71, Springer–Verlag Berlin Heidelberg New York, 1996

[12] Toressen, J.: A scalable approach to evolvable hardware. Genetic Programming and Evolvable Machines, 3(3), pp 259–282, 2002

[13] Gordon, T. G. W., Bentley, P. J.: Towards development in evolvable hardware. In: Proc. of the 2002 NASA/DoD Conference on Evolvable Hardware, pp 241–250, IEEE Computer Society Press, 2002

[14] Haddow, P., Tufte, G.: Bridging the genotype–phenotype mapping for digital FPGAs. In: Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware, pp 109–115, IEEE Computer Society Press, 2001

[15] Sekanina, L., Bidlo, M.: Evolutionary Design of Arbitrarily Large Sorting Networks Using Development. Genetic Programming and Evolvable Machines, vol. 6, 2005 (in print)

[16] Miller, J. F., Thomson, P.: Cartesian genetic programming. In: Proc. of the 3rd European Conference on Genetic Porgramming, Lecture Notes in Computer Science, vol 1802, pp 121–132, Springer–Verlag Berlin Heidelberg New York, 2002

[17] Miller, J. F., Thomson, P.: A developmental method for growing graphs and circuits. In: Proc. of the 5th Conf. on Evolvable Systems: From Biology to Hardware (ICES 2003), Lecture Notes in Computer Science, vol. 2606, pp 93–104, Springer–Verlag Berlin, DE, 2003

[18] Bidlo, M.: A Developmental Method for Construction of Arbitrarily Large Sorting Networks and Adders. Brno, CZ, 2005

[19] Zeno, R.: A Reference of the Best-Known Sorting Setworks for Up To 16 Inputs. 2003

[20] Codd, E. F.: Cellular Automata. Academic Press, New York, 1968 http://www.angelfire.com/blog/ronz/Articles/999SortingNetworksReferen.html