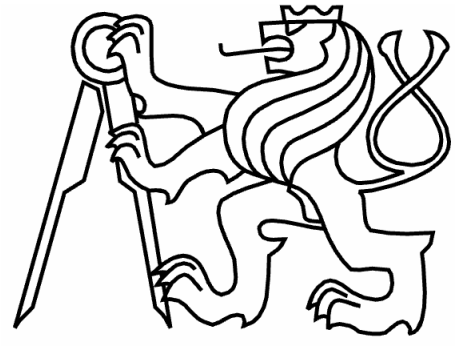
**České vysoké učení technické v Praze**

**Fakulta elektrotechnická**

**Katedra počítačů**



**Bakalářská práce**

# Rychlý simulátor kombinačních obvodů

Karel Houžvička

Vedoucí práce: Ing. Petr Fišer, Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, bakalářský

Obor: Výpočetní technika

19. květen 2010

## Poděkování

Hlavní poděkování patří panu Ing. Petru Fišerovi, Ph.D. za skvělé vedení, rady a spolupráci. Dále bych chtěl poděkovat mé rodině, přátelům a především mé přítelkyni za podporu a důvěru.

## Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů·(autorský zákon).

V Praze dne 20. 5. 2010 ……..……..

## Abstract

The aim of this project is to implement an application for simulation of two-level logic circuits described in PLA format. Another goal is to extend this application to be able to simulate multi-level circuits described in BLIF, considering combinational and even sequential types of BLIF. In order to achieve fast simulation, standard techniques are used - e.g. machine-word parallel simulation or preprocessing.

## Abstrakt

Cílem projektu je naprogramovat simulátor dvouúrovňových kombinačních obvodů zadaných ve formátu PLA a dále je rozšířit na simulaci víceúrovňových obvodů zadaných ve formátu BLIF kombinačního i sekvenčního typu. Pro dosažení vyšší rychlosti simulace využiji standardních technik jako je paralelní simulace na úrovni strojového slova a předzpracování.

# Obsah

# Seznam obrázků

# Seznam tabulek

## Seznam grafů

# 1 Úvod

Cílem práce bylo naprogramovat simulátor kombinačních obvodů zadaných pomocí formátu PLA a BLIF. Tyto formáty se využívají pro zápis obvodů různé složitosti, ale jejich výhoda a praktičnost se projeví až u větších obvodů. Díky zmíněným vlastnostem se tyto formáty využívají pro přímé programování logických kontrolérů. Tato zařízení jsou cenově velice zajímavá a není problém si pomocí nich vytvořit vlastní číslicový obvod bez znalosti programování, jak je tomu u mikroprocesorů.

Možnost simulace je zde velice potřebná, můžeme pomocí ní otestovat jednotlivé návrhy logických obvodů. Pokud nemáme jistotu, že návrh funguje, není ho vhodné testovat až po konkrétní implementaci do zařízení, ale řešení odzkoušet softwarově. Tím se urychlí čas i případné náklady spojené s chybou nebo fyzickým testováním. Ještě více důležité je využití simulace při vlastním vytváření logické reprezentace.

V této práci se zaměřím na diskrétní simulaci, to znamená, že se nebudu zabývat zpožděním jednotlivých obvodů. Zaměřím se pouze na vztah mezi vstupem s výstupem a jejich závislostí. To znamená, že množinu vstupních hodnot převedu na množinu výstupních hodnot.

Program bude také umět simulovat neurčené stavy na vstupu. Můžeme nastavit některé vstupy do obvodu jako neurčené a po simulaci analyzovat jejich vztah na jednotlivé výstupní signály. Toto se může hodit mimo jiné k hledání poruch a jejich odolnosti vůči jim.

Nemalým parametrem v mé práci je rychlost samotné simulace, aby bylo možné pracovat s velkým množstvím dat v reálném čase. K tomu využiji standardních technik jako je paralelizace na úrovni strojového slova a předzpracování.

Zaměřím se také na simulaci sekvenčních obvodů, které se ve formátu BLIF také vyskytují.

Posledním úkolem na samý závěr práce je zjistit závislost počtu vstupů LUTu na rychlosti simulace. Toto zjištění může vést k doporučení pro zrychlení simulace, protože při přemapování na určitý počet vstupů může docházet ke změnám celkové rychlosti.

Program je postaven na jádře BOOM. Toto jádro poskytuje již objekty simulovaných standardů PLA a BLIF. Nejprve bylo nutné seznámit se s formátem PLA, na který jsem vytvořil triviální, ale plně funkční simulaci, kde jsem si osvojil základní znalosti této problematiky. Poté následovala simulace formátu BLIF, jenž jsem dále rozvíjel a vylepšoval. V této práci se budu zabývat především druhým jmenovaným formátem, na němž jsou všechny níže popsané algoritmy implementovány.

# 2 Základní pojmy

Zde si povíme něco o formátech, se kterými se v bakalářské práci setkáme. V této kapitole je jen základní shrnutí, podrobnější popis se nachází v Příloze D Základní definice formátů.

## 2.1 Formát PLA

Tento formát umožňuje zápis dvouúrovňového kombinačního obvodu.

V jeho hlavičce je deklarován počet vstupů, výstupů a nepovinně i jejich jména. Dalším důležitým parametrem je typ PLA určující, co definuje vnitřní tabulka termů. Tabulka lze definovat kombinací tří různých množin. Množina ON-SET definuje na výstupu jedničku, OFF-SET nulu a DC-SET neučený stav. Pokud se jedná o definici pomocí ON-SETu, pak je za `.type` písmeno `f`. OFF-SET a DC-SET označuje písmeno `r` respektive `d`. Tyto typy se mohou kombinovat a jednotlivé PLA může určovat více množin, například `f`, `r`, `fr`, `fd`, `dr`.

Tělo PLA se skládá z tabulky definující vazbu mezi specifickým vstupním termem a výstupem. Pokud počet takových řádků není stanoven v hlavičce, může jich být velké množství, ale obecně by jich měl být minimální počet, se kterým se dá jednotlivá funkcionalita vytvořit.

```
.i 4
.o 2
.type fd
0100 00
0101 10
-111 1~
```

**Obr. 2.1.1 Ukázka souboru PLA**

V hlavičce (**Obr. 2.1.1**) je určen počet vstupů na čtyři a počet výstupů na dvě, dále je uveden typ `fd`, jenž říká, že se jedná o definici pomocí ON-SETu a DC-SETu. Poté následuje tabulka jednoznačně určující vazbu mezi vstupem a výstupem.

Při řešení procházíme postupně vstupní termy a pokud se některý shoduje se vstupním vektorem, máme výsledek nebo jeho část, protože ve výstupní části může být znak „~“ doplňující jiný řádek, nebo je určen na konci podle typu jednotlivého PLA.

## 2.2 Formát BLIF

Druhým formátem umožňujícím zápis víceúrovňového kombinačního i sekvenčního obvodu je BLIF. V hlavičce souboru je uveden název modelu a jména vstupních a výstupních formátů.

Tělo BLIF se skládá z více jednovýstupových PLA, která jsou definována pomocí hlavičky se jmény vstupních signálů a jedním výstupním signálem, oddělených navzájem mezerou definující název a hodnotu signálu sloužící buď jako další vstup nebo jako výstup celého BLIFu. Těmto tabulkám budu říkat v celé práci Look Up Table, zkráceně LUT.

```
.model C17.iscas
.inputs 1GAT(0) 2GAT(1) 3GAT(2) 6GAT(3) 7GAT(4)
.outputs 22GAT(10) 23GAT(9)
.names 1GAT(0) 2GAT(1) 3GAT(2) [2] 22GAT(10)
1-1- 1
-1-1 1
.names 2GAT(1) 7GAT(4) [2] 23GAT(9)
1-1 1
-11 1
.names 3GAT(2) 6GAT(3) [2]
0- 1
-0 1
.end
```

**Obr. 2.2.1 Ukázka souboru BLIF**

Na obrázku je BLIF model `C17.iscas` se vstupy `1GAT(0)`, `2GAT(1)`, `3GAT(2)`, `6GAT(3)`, `7GAT(4)` a výstupy `22GAT(10)`, `23GAT(9)`. Soubor obsahuje tři LUTy. Je zde vidět jejich provázanost pomocí názvů signálů. První LUT nelze simulovat, protože neznáme jeden vstup, tj. `[2]`. Ten zjistíme až po simulaci třetího.

# 3 Analýza problémů a jejich řešení

## 3.1 Simulace PLA

Při simulaci PLA procházíme množinu termů a porovnáváme vstupní vektor s jednotlivými termy. Pokud term odpovídá vstupu, doplní se výstup o definované hodnoty, ty mohou obsahovat nedefinovaný stav „~“, jenž je později doplněn jiným termem nebo nahrazen na konci simulace podle typu jednotlivého PLA doplňkem.

Z tohoto jednoduchého popisu vycházejí jednotlivé problémy.

První problém je porovnávání vstupu s termem. Na každý jednotlivý term budeme pohlížet jako na vektor, s nímž porovnáváme vstupní vektory. Pro využití logických funkcí musíme převést term na bitové pole, které ovšem nabývá tří hodnot {1,0,-}. Proto je nutné provést rozklad na datovou část ($TD$) a na část neurčitých stavů($TN$). Stav „0“ na $\langle 0,1 \rangle$, „1“ na $\langle 1,1 \rangle$ a „–“ na $\langle 0,0 \rangle$. Této problematice je věnována kapitola 4.2.2 Reprezentace základních dat.

$$vstup \times TN_i = TD_i \tag{3.1}$$

Vstup, který si převedeme na bitové pole, vynásobíme neurčenou částí termu. Vznikne nám upravená hodnota, v níž jsou vymaskovány hodnoty signálů, jenž nedefinuje jednotlivý term. Poté stačí již jen porovnat.

Druhým problémem je skládání jednotlivých výstupů termu. Zde se vychází ze specifikace formátu PLA pro program Espreso [1].

- Nahrazují se zde neurčené stavy hodnotou.
- Pokud se setkají stejné hodnoty, nic se neděje.
- Při setkání „1“ a „–“ u typu `fd` je výsledný hodnota „–“.

Posledním problémem je závěrečné doplnění nedefinovaných výstupních hodnot. Ty se nahradí znakem reprezentujícím množinu, která je doplňkem definující množiny jednotlivého PLA.

| Typ PLA | Doplněk |
|---|---|
| f | 0 |
| r | 1 |
| fr | - |
| fd | 0 |
| rd | 1 |

**Tabulka 3-1 Definice doplňků PLA**

U typu `frd`, který je definovaným všemi množinami nedojde ke stavu, kdy po projití celé množiny termů je ve výstupu nedefinovaná hodnota.

## 3.2 Simulace LUTu

Nyní si řekneme o simulaci základní části BLIF, o LUTu. Jak již bylo výše uvedeno, jedná se o tabulku součinových termů, která má nadefinovány vstupy a jeden výstup. Zde jako v předešlé části si převedeme jednotlivé termy na dva bitové vektory, které definují datovou část a část neučených stavů.

Zkratky a znaky použité ve vzorcích:

**Vstup**
- Datová část — *vstD*
- Neurčitá část — *vstN*

**Term**
- Datová část — *trmD*
- Neurčitá část — *trmN*

**Operace #**
- Logický součin přes všechny položky bitového pole, je zprava asociativní.

Za předpokladu simulace vstupů skládajících se pouze z jedniček a nul, je simulace jedné tabulky celkem triviální.

$$((vstD \times trmN_i) \otimes trmD)\#$$

(3.2)

Tento výpočet provede simulaci jednoho vektoru postupně přes všechny řádky tabulky. Část $vstD \times trmN$ vymaskuje neurčené hodnoty vstupního vektoru termem tabulky a následná operace $\otimes trm$ porovná hodnotu s datovým vektorem termu. Pokud se jedná o shodu, logický součin přes všechny položky bitového pole bude jedna. Výpočet můžeme v tomto okamžiku ukončit, protože máme jednoznačně určen výstup.

Mnohem složitější nastává situace, jestliže chceme simulovat neurčité stavy na straně vstupních vektorů. Zde je nutné opět zachovat stejný výstupní formát, jenž se skládá z části datové a části určující neurčené stavy. Pokud hovoříme o jednovýstupových tabulkách,

výsledná hodnota pro každý vektor se skládá právě ze dvou bitů. Datovou část určíme pomocí hledání možnosti výpočtu, při kterém zanedbáváme neurčité stavy, jenž doplní až druhá část.

| trmN | trmD | vstN | vstD | term | vstup | data |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | - | - | 1 |
| 0 | 0 | 0 | 1 | - | x | x |
| 0 | 0 | 1 | 0 | - | 0 | 1 |
| 0 | 0 | 1 | 1 | - | 1 | 1 |
| 0 | 1 | 0 | 0 | x | - | x |
| 0 | 1 | 0 | 1 | x | x | x |
| 0 | 1 | 1 | 0 | x | 0 | x |
| 0 | 1 | 1 | 1 | x | 1 | x |
| 1 | 0 | 0 | 0 | 0 | - | 0 |
| 1 | 0 | 0 | 1 | 0 | x | x |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | - | 0 |
| 1 | 1 | 0 | 1 | 1 | x | x |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Tabulka 3-2 Určení datové části**

V tabulce jsou zahrnuty nedefinované stavy určené kombinací termu nebo vstupu $\langle data, neurcene \rangle = \langle 1,0 \rangle \Rightarrow X$ . Tyto stavy můžeme pro účely minimalizace zahrnout, protože se nikdy nevyskytnou.

$$D = (\overline{trmN}_i + vstD \times trmD_i + \overline{vstD} \times vstN \times \overline{trmD}_i)\#$$

(3.3)

Výsledná funkce pro datovou část určí, kde je výsledek jednoznačně určen vstupním vektorem a kde ne.

Funkce určující druhou část výsledku zjišťuje řešení s nedefinovanými stavy na vstupu. Bitový vektor pro datovou část vstupu a termu vynásobíme částí určující neurčité stavy a ty porovnáme mezi sebou. Využíváme znalosti, že $xD \times xN = xD$ , pro $x$ určující název bitového vektoru, která vyplývá z určení reprezentace stavů pomocí dvou vektorů.

$$DC = ((vstD \times trmN_i) \otimes (trmD_i \times vstN))\# \otimes D$$

(3.4)

Výsledek porovnáme s již vypočítanou datovou částí pomocí operace nonekvivalence, která nám při shodném výsledku potvrdí hodnotu z datové části $D$ jedničkou. Při neshodě se nastaví neurčitý stav pomocí nuly.

Získali jsme výsledek porovnání jednoho termu se vstupem. Simulovaný LUT se ale většinou skládá z více termů. Proto je ještě nutné stanovit funkci, jenž nám bude spojovat výsledky postupně ze všech termů.

| V2N | V2D | V1N | V1D | V2 | V1 | vysl |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | - | - | - |
| 0 | 0 | 0 | 1 | - | x | x |
| 0 | 0 | 1 | 0 | - | 0 | - |
| 0 | 0 | 1 | 1 | - | 1 | 1 |
| 0 | 1 | 0 | 0 | x | - | x |
| 0 | 1 | 0 | 1 | x | x | x |
| 0 | 1 | 1 | 0 | x | 0 | x |
| 0 | 1 | 1 | 1 | x | 1 | x |
| 1 | 0 | 0 | 0 | 0 | - | - |
| 1 | 0 | 0 | 1 | 0 | x | x |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | - | 1 |
| 1 | 1 | 0 | 1 | 1 | x | x |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Tabulka 3-3 Určení spojování výsledků z jednotlivých termů**

V tabulce je vidět slučování výsledků z různých řádků tabulky termů $v1$ a $v2$. Priorita výsledků je v tomto pořadí definována od nejnižší po nejvyšší následovně $\{0,-,1\}$. Pomocí minimalizace se získají dvě rovnice.

Datová část výsledku:

$$vyslD = v1D + v2D \tag{3.5}$$

Neurčitá část výsledku:

$$vyslN = v1D + v2D + v1N \times v2N \tag{3.6}$$

Tyto rovnice zabezpečí správné sčítání výsledků z jednotlivých termů.

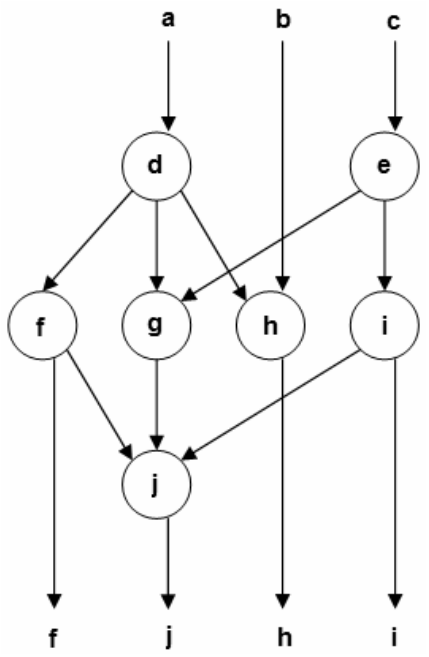
Celý výpočet se provádí pro typ LUTu `f`, definovaným ON-SETem. Proto je nutné v případě, že se jedná o typ `r`, definovaný OFF-SETem, provést na konci simulace LUTu inverzi. $0 \to 1$, $1 \to 0$, neurčitý stav zůstává, tím se myslí převod $- \to -$ i nezměnění části výsledku definujícího neurčité stavy.

Inverzní rovnice:

$$vyslD = \overline{vyslD} \times vyslN$$
$$vyslN \text{ zůstává} \qquad (3.7)$$

## 3.3 Předzpracování a simulace struktury BLIF

BLIF má vícerozměrovou strukturu, jejíž vztahy jsou navzájem definovány pomocí názvů signálů.




**Obr. 3.3.1 Struktura formátu BLIF**

Na obrázku je názorná ukázka. Vstupní signály jsou **a**, **b**, **c** a výstupní signály **f**, **j**, **h**, **i**. LUTy jsou reprezentovány kolečky s názvy výstupní proměnné. Jedná se o orientovanou síť uzlů, jejich průchodem získáme výslednou množinu. Každý uzel lze projít (odsimulovat) pouze tehdy, pokud jsou známy všechny jeho vstupní hrany.

Na začátku známe pouze množinu signálů, která obsahuje vstupy $A = \{a,b,c\}$ a potřebujeme dosáhnout stavu, kdy bude platit $A \supseteq X, X = \{f, j, h, i\}$. To znamená, že budou známy výstupní hrany. Množina $A$ vždy obsahuje dostatečné množství prvků pro splnění podmínek pro vstup do nějakého uzlu. Zde to splňují hned uzly dva $D = \{a\}, E = \{c\}, A \supseteq D, A \supseteq E$, po jejich průchodu se množina $A$ rozšíří o jejich výstupy $A = \{a,b,c,d,e\}$. Takto to pokračuje dokud $A \supseteq X$ .

Na ukázce je vidět, že nelze simulaci provádět v náhodném pořadí a formát BLIF nijak nenormalizuje pořadí LUTů. Program díky tomu pracuje se seznamem LUTů v náhodném pořadí.

Předzpracování spočívá v seřazení seznamu LUTů podle možnosti jejich řešitelnosti, tak aby každý LUT byl signálově závislý jen na svých předchůdcích v seznamu. To značně zjednoduší a urychlí řešící algoritmus. Seřazení zajistí platnost signálů potřebných k další simulaci. Bez této metody by se musel seznam procházet a pro každý LUT zkoušet sestavit vstup. Pokud by se nalezl signál, který není ještě definovaný, skládání vstupu by se muselo přerušit a pokračovalo by se dále podle seznamu. Po simulaci každého LUTu by bylo nutné jej označit jako odsimulovaný, jinak by došlo k zacyklení. Další částí, která úplně zmizí, je test, zda máme celý BLIF odsimulovaný a je nutné ukončit cyklus procházení seznamem. V předzpracované verzi stačí pouze jeden průchod seznamem a máme jistotu, že všechny výstupní signály byly odsimulovány.

## 3.4 Paralelizace výpočtu

### 3.4.1 Základní princip

Paralelizace výpočtu představuje největší zrychlení. Využívá se v ní základní vlastnost mikroprocesoru, že za jeden takt můžeme provést logickou operaci nejen nad jedním bitem informace, ale i nad celými 32 bity, které reprezentují procesorové slovo. Logický součin dvou čísel typu `integer`, trvá stejnou dobu jako součin dvou proměnných typu `boolean`. Pokud si `integer` nepředstavíme jako pouhé číslo, ale jako bitové pole o délce 32, je hned vše jasné. Dvaatřicetinásobného teoretické zrychlení lze dosáhnout právě s 32 vektory o velikosti simulovaného modelu blížícího se nekonečnu, protože do času nutného na simulaci se započítává konstantní doba úpravy vstupních dat.

$$\lim_{x\to\infty}\left(\frac{32\times x+K}{x}\right)=\lim_{x\to\infty}\left(\frac{32\times x}{x}+\frac{K}{x}\right)=\lim_{x\to\infty}\left(32+\frac{K}{x}\right)=32+\frac{K}{+\infty}=32 \tag{3.8}$$

K – Časová konstanta potřebná k převodu vektorů na paralelní reprezentaci.

x – Čas potřebný k simulaci.

Algoritmus paralelizace dat zajišťuje změnu 32bitových vektorů délky $n$ na 32bitové číslo v počtu $n$.

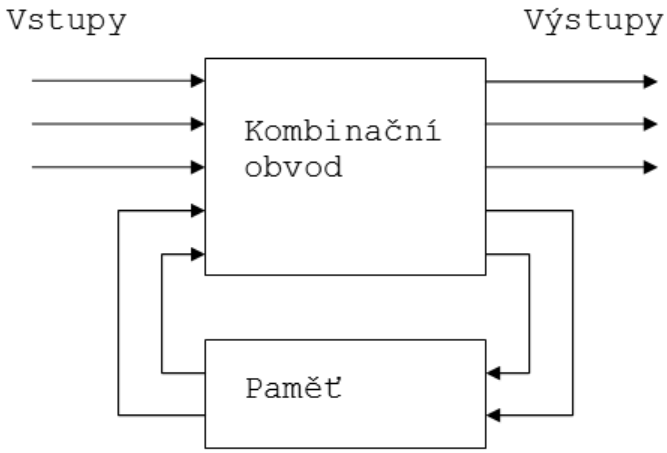
$$\begin{array}{rcccccc} & & & & & & b_0 & b_1 & b_2 & b_3 & b_4 \\ & & & & & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ a_0 \rightarrow & \langle 1 & 0 & 1 & 1 & 1\rangle & 1 & 0 & 1 & 1 & 1 \\ a_1 \rightarrow & \langle 0 & 1 & 0 & 0 & 0\rangle & \Rightarrow 0 & 1 & 0 & 0 & 0 \\ a_2 \rightarrow & \langle 0 & 1 & 1 & - & 0\rangle & 0 & 1 & 1 & - & 0 \\ a_3 \rightarrow & \langle 1 & 0 & 0 & 0 & 0\rangle & 1 & 0 & 0 & 0 & 0 \\ a_{31} \rightarrow & \langle - & 1 & 1 & 0 & 1\rangle & - & 1 & 1 & 0 & 1 \end{array}$$

**Obr. 3.4.1 Paralelizace vektorů**

Na Obr. 3.4.1 je ukázána situace, kde máme 32 vstupních vektorů $a_0 \ldots a_{31}$ velikosti 5. Velikost všech vektorů je konstantní, je dána počtem vstupních signálů do simulovaného modelu. Výstup metody je 5 vektorů `integer`, každý reprezentuje jeden vstupní signál do modelu.

## 3.5 Simulace sekvenčních obvodů BLIF

Pro simulaci sekvenčních obvodů není potřeba udělat velké změny v zápise obvodu ani v jeho simulaci.




**Obr. 3.5.1 Teoretické schéma sekvenčního obvodu**

Jak ukazuje teoretické schéma obvodu Obr. 3.5.1 , sekvenční obvod se skládá z kombinační části a paměti. Paměť slouží k zachování minimálně jednoho výstupního signálu, který slouží jako vstup do kombinační části v příštím časovém intervalu. My se nebudeme zabývat hodinami a jejich závislostmi, proto pro nás paměť reprezentuje vstupní signály, jenž byly nastaveny při simulaci minulého vektoru.

Díky této vlastnosti, že každý simulovaný vstupní vektor je závislý na výsledku řešení předchozího, není bohužel možné provádět paralelní simulaci 32vektorů najednou. To znamená, že se celkový algoritmus 32krát zpomalí.

Formát BLIF umožňuje mnoho způsobů jak reprezentovat sekvenční obvod. V této práci se zaměřím na zápis pomocí struktury `.latch`, která přímo reprezentuje onu paměť neboli klopný obvod.

# 4 Implementace

## 4.1 Programovací jazyk

V mém případě jsem si nemohl určit jaký programovací jazyk použiji pro tuto bakalářskou práci, protože stavím na jádře pana Ing. Petra Fišera Ph.D. , které je naprogramované v C++. Tento jazyk je nejlepší volbou, neboť se zde zabýváme prací nad základními daty skládajících se převážně z jedniček a nul. C++. Jako jediný jazyk vyšší úrovně má velice malou režii zpracování a co nejvíce se přibližuje strojovému kódu. Díky této vlastnosti se hodí pro psaní programů majících velkou vazbu na hardware, jako jsou například ovladače nebo v mém případě řešící programy, ve kterých hraje velkou úlohu rychlost.

## 4.2 Datové struktury

### 4.2.1 Program jako celek

Hlavní myšlenka a zadání programu byly jasné. Simulace formátu BLIF a PLA. To znamená z určité množiny vstupních vektorů vytvořit množinu výstupních vektorů odpovídajících simulované struktuře. Mnou navrhnuté řešení programu je procedurální, nevytvářel jsem žádnou novou třídu či strukturu pro simulaci, protože celkový smysl není vhodný pro objektové ztvárnění. Samozřejmě jsem se neobešel bez objektů. Ty mi poskytlo jednak jádro BOOM s reprezentací formátů PLA a BLIF, a jednak základní knihovna C++ STL (Standard Template Library), která usnadňuje práci s velkými objemy dat pomocí kontejnerů.

### 4.2.2 Reprezentace základních dat

Základní jednotkou pro simulaci je vstupní vektor skládající se z jedniček a nul. Každá položka by měla být samostatně adresovatelná a zároveň musí být implementovány logické bitové operace (AND, OR, XOR, NEG) pro celek. Nejvhodnějším objektem pro ukládaní vektoru je `Bitvector` z jádra BOOM, který výše zmíněné podporuje.

Pro podporu simulace i neurčených vstupů se situace komplikuje tím, že již nemáme dvoustavovou logiku `{1, 0}`, ale třístavovou `{1, 0, -}`. K jedničce a nule se přidal neurčený stav. Přesto chceme nad daty pracovat bitovými operacemi. Nejvhodnější řešení je tedy rozdělit vektor na dvě části. Část datovou a část neurčených stavů.

| Vstup | Datová část | Část neurčených stavů |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |
| - | 0 | 0 |

**Tabulka 4-1 Rozklad vstupního vektoru**

Například:

```
Vstup:              1001-10
Data:               1001010
Neurčené stavy:     1111011
```

### 4.2.3 Základní práce s vektory

Pro práci s velkým počtem vstupních a výstupních vektorů, jejichž počet je zdvojnásoben díky rozdělení na dvě části, je využit kontejner z knihovny STL List. Tento objekt má výhodné vlastnosti díky konstantnímu přístupu k položkám přes iterátor a umožňuje vkládání. V Listu jsou vektory uloženy v pořadí vektor dat, jenž je následován vektorem neurčených stavů, což zaručuje jednoduchost ukládání a zpracování, které iteračně postupuje od první položky k poslední.

## 4.3 Simulace PLA

Základní program na simulaci PLA je vytvořen z jednoduchých struktur a z velké části se zde pracuje pouze se řetězci `string`. Jeho implementace se skládá z jednoduchých bloků.

### 4.3.1 Pseudokód simulace PLA

```
ALGORITMUS PLASIMUL (PLA VSTPLA, STRING VSTUP)
    VSTBITV = VYTVOŘ_BITVEKTOR(VSTUP);
    STRING OUT = PRÁZDNÝ ŘETĚZEC;
    FOR(ITERATOREM PROCHAZEJ SEZNAM TERMŮ V VSTPLA)
        BEGIN
        IF(VSTBITV&BITVEKTRONEUR(ITER->TERM)==BITVECTORDAT(ITER->TERM))
            OUT = SLOUČI(OUT,ITER->OUT);
        END;
 RETURN DOPLŇVÝSTUP(OUT);
END-ALGORITMU;
```

V první části se prochází množina termů a porovnávají se se vstupní hodnotou. Pokud je podmínka splněna, sloučí se jednotlivé výstupy. Po projití všech termů se doplní neurčité stavy, které zbyly ve výstupu.

## 4.4 Paralelizace dat k výpočtu BLIFu

### 4.4.1 Použité struktury

Jako výstup paralelizace je skupina 32bitových proměnných typu `unsigned int`. Pro přehledný přístup v simulaci jsem zvolil uložení dvou čísel reprezentujících 32 vstupů jednoho signálu do datového typu `pair <unsigned int, unsigned int>.` První z páru čísel je vždy reprezentace datové části a druhý představuje část neurčených stavů.

Paralelizace přímo předchází simulaci celého BLIFu. Proto je vhodné data rovnou přizpůsobit následujícímu algoritmu. Protože simulace formátu BLIF je přímo spjatá s názvy signálů, k nimž náhodně přistupujeme, není vhodné ukládat data do struktury, která má časovou složitost pro hledání $O(n)$ , kde $n$ je počet položek. Vhodnější pro reprezentaci dat

je asociativní pole, neboli mapa. Datový typ STL Map má složitost pro vyhledávání $O(\log(n))$ a přímý přístup pomocí názvu signálu. Výsledný datový typ pro ukládání dat po paralelizaci je `map<string,pair<unsigned int,unsigned int>>`.

### 4.4.2 Pseudokód metody upravující data do paralelní formy

```
ALGORITMUS PARALELIZACE (LV = LIST(BITVEKTOR),VJ=VECTOR(JMÉNA SIGNÁLŮ) )
    VÝSTUP = VYTVOŘ ASOCIATIVNÍ MAPU TYPU <UNSIGNED INT, UNSIGNED INT>;
    MASKA = 1;
    FOR (I=0; I<32; I++)
            BEGIN
            IF(KONEC LV) BREAK;
            VDATA = VEZMI BITVEKTOR Z VL;
            VDC = VEZMI BITVEKTOR Z VL
            FOR (J=0; J<VELIKOST BITVEKTORU; J++)
                    BEGIN
                    POMOCNÁ = 0XFFFF * VDATA[J] ;
                    POMOCNÁDC = 0XFFFF * VDC[J] ;
                    VÝSTUP[VJ[J]].FIRST = (MASKA&POMOCNÁ)| VÝSTUP[VJ[J]].FIRST;
                    VÝSTUP[VJ[J]].SECOND= (MASKA&POMOCNÁ)| VÝSTUP[VJ[J]].SECOND;
                    END;
            ZAROTUJ MASKU O JEDNA DOLEVA;
            END;
    RETURN VÝSTUP;
END-ALGORITMU;
```

Vstupem do metody je List bitových vektorů, které chceme odsimulovat. Ty postupně procházíme a upravujeme výsledný seznam vektorů `integer`, jenž byl na začátku inicializován na nulu.

### 4.4.3 Kompletní obsluha vstupních dat při paralelizaci

```
ALGORITMUS STARTSIMUL (LVIN = LIST(BITVEKTOR),LVOUT =* LIST(BITVEKTOR), BLIF B)
    WHILE(JSOU NEODSIMOLOVANÉ SIGNÁLY)
            BEGIN
            MAP_IN = PARALELIZACE(VLIN,B.SEZNAM SIGNÁLŮ) ;
            MAP_OUT = SIMUL(MAP_IN) ;
            LVOUT.VLOŽ(DEPARALELIZACE(MAP_OUT));
            END;
END-ALGORITMU;
```

Po výpočtu je opět nutné převést hodnoty zpět na výstupní formát bitových vektorů. Tato metoda je obdobná. Změna spočívá pouze v tom, že vstup je `unsigned integer` a výstup je bitový vektor. Toto se opakuje do té doby, dokud jsou na vstupu neodsimulované vektory.

Celková časová náročnost úpravy je $\Theta(n \times m)$, kde $n$ je velikost vstupního vektoru a $m$ je počet vstupních vektorů.

### 4.4.4 Paralelizace u sekvenčního odvodu

V případě simulace sekvenčního typu souboru se paralelizace také provede, ale skupina hodnot `unsigned integer` bude reprezentovat pouze jeden vektor. Pro $1 \rightarrow 0xFFFF$ a $0 \rightarrow 0x0$. Tím se zajistí, že algoritmus pro simulaci struktury BLIFu bude stejný jak pro kombinační obvod, tak pro sekvenční obvod.

Dále zde přibude inicializace mapy reprezentující klopný obvod. Ukazatel této mapy se předává jako další parametr do metody simulace. Tím se zajistí pamatování stavu do následující simulace.

Celková časová náročnost úpravy sekvenčního obvodu je nezměněných $\Theta(n \times m)$.

## 4.5 BLIF předzpracování

Algoritmus předzpracování používá asociativní mapu platnosti jednotlivých signálů a k nim přiřazených `boolean` proměnných určujících jejich platnost. Datový typ vypadá přesně takto `map<string, bool>`. Na počátku tato mapa obsahuje platné vstupní a neplatné výstupní signály. Při řazení se mapa postupně doplní o vnitřně definované proměnné určené výstupem jednotlivého LUTu.

### 4.5.1 Pseudokód metody předzpracování

```
ALGORITMUS PŘEZPRACOVÁNÍ (BLIF &IN) )
    MAPVALID = VYTVOŘÍ SE PRÁZDNÁ MAPA;
    MAPVALID.VLOŽIT VSTUPNÍ SIGNÁLY Z IN A NASTAVIT NA PLATNÉ;
    MAPVALID.VLOŽIT VÝSTUPNÍ SIGNÁLY Z IN A NASTAVIT JAKO NEPLATNÉ;
    LIST NOVÝLIST = VYTVOŘIT NOVÝ PRÁZDNÝ SEZNAM LUTŮ ;
    INT NEZAŘAZENO NASTAVIT NA CELKOVÝ POČET LUTŮ Z IN;
    WHILE(NEZAŘAZENO>0)
        BEGIN
        FOR(ITERATOREM PROCHAZEJ SEZNAM LUTŮ V IN)
            BEGIN
            IF(ITER->JE SEŘAZEN) CONTINUE;
            IF(ITER->JDE SLOŽIT VSTUP)
                BEGIN
                ITER->OZNACIT JAKO SEŘAZENÝ;
                NAZAŘAZENO - - ;
                NOVÝLIST.VLOŽIT NA KONEC  ITER;
                IF(EXISTUJE V MAPVALID ITER.VYSTUPNÍ SIGNÁL)
                    MAPVALID[ITER.VYSTUPNÍ SIGNÁL]=PLATNÝ;
                ELSE
                    MAPVALID.VLOŽIT ITER.VÝSTUPNÍ SIGNÁL JAKO PLATNÝ;
            END;
        END;
    UPRAV SEZNAM LUTŮ V IN NA NOVÝLIST;
    END-ALGORITMU;
```

Algoritmus spočívá v procházení seznamu LUTů a zkoušení sestavení vstupu do jednotlivé položky z mapy platnosti signálů. Pokud je možné LUT odsimulovat, označí se a

vloží se do nového seznamu. V jednom průchodu seznamem lze minimálně jednou složit vstup a daný LUT odsimulovat, čímž se rozšíří množina platných signálů. Na konci algoritmu je nový seznam již seřazený a nahradí se jím stávající seznam v objektu BLIF.

Změna pro sekvenční obvody je velice malá. Pouze se před hlavním cyklem projde zvlášť seznam LUTů a všechny klopné obvody, které jsou zde také obsaženy, se jako první vloží na konec do nového seznamu. To znamená, že jsou v předzpracovaném BLIFu na začátku, čímž jsme také získali jednoduchý test, zda se jedná o sekvenční obvod či nikoliv. Otestujeme první ze seznamu LUTů, zda se jedná o klopný obvod.

Tato úprava struktury BLIFu se provádí na začátku programu a pouze jednou časovou náročností $O(n^2)$, kde $n$ je počet LUTů. Díky tomu se zefektivní proces skládání vstupů do jednotlivých LUTů z $O(n^2)$ na $\Theta(n)$.

## 4.6 Simulace struktury BLIFu

Simulace struktury zajišťuje pouze skládání vstupů do jednotlivých LUTů. Díky předzpracování se nemusíme zabývat platností signálů a iteračně procházíme seznam LUTů. Pro každý LUT složíme množinu vstupů a zavoláme metodu řešící samostatný LUT, jenž nám vrátí hodnotu výstupního signálu, kterou musíme opět vložit do vhodné mapy.

### 4.6.1 Pseudokód simulace struktury

```
ALGORITMUS SIMUL (BLIF B, MAP MAP_IN )
    MAP_OUT = INICIALIZACE VÝSTUPNÍ MAPY;
    MAP_OUT = VLOŽENÍ VŠECH VÝSTUPNÍCH SIGNÁLŮ B;
    MAP_OTHER = INICIALIZACE MAPY VNITŘNÍCH SIGNÁLŮ;
    FOR(ITERÁTOREM PROCHÁZEJ SEZNAM LUTŮ V B)
        BEGIN
        VEKTOR_IN = INICIALIZACE;
        FOR(PROCHÁZEJ SEZNAM VSTUPŮ DO AKTUÁLNÍHO LUTU)
            BEGIN
            IF(MAP_OTHER.FIND(VSTUP)) VEKTOR_IN.VLOŽ(MAP_OTHER[VSTUP])
            ELSE IF(MAP_IN.FIND(VSTUP)) VEKTOR_IN.VLOŽ(MAP_IN[VSTUP])
                ELSE VEKTOR_IN.VLOŽ(MAP_OUT[VSTUP]);
            END;
        VÝSLEDEK = SIMULACELUTU(VEKTOR_IN, LUT);
        IF(MAP_OUT.FIND(VÝSTUPLUTU)) MAP_OUT[VÝSTUPLUTU] = VÝSLEDEK
            ELSE MAP_OTHER[VÝSTUPLUTU] = VÝSLEDEK;
        END;
RETURN MAP_OUT;
END-ALGORITMU;
```

Vektor, který se předává simulační metodě nižší vrstvy BLIFu, je reprezentován kontejnerem `STL vector<pair<unsigned int, unsigned int>>`. Tato struktura se může již odprostit od názvů signálů, protože je sestavená a seřazená přesně na míru jednotlivého LUTu.

Časová náročnost simulace struktury BLIFu je vzhledem k předzpracování $\Theta(n)$, kde $n$ je počet LUTů.

U sekvenčního typu přibude pouze mapa reprezentující klopný obvod, která se předá této metodě jako parametr a pracuje se s ní stejně jako s ostatními mapami. Hledají se v ní signály pro složení vstupu a ukládají se výsledky simulace jednotlivých LUTů.

Jádro BOOM bohužel nepodporuje načítání a práci s přednastavenou hodnotou klopného obvodu, proto je v programu nastavena napevno hodnota 0.

## 4.7 Simulace LUTu

Simulace LUTu probíhá podle vzorců uvedených v analýze kapitola 3.2 Simulace LUTu. Před procházením termu se musí rozložit na datovou a neurčitou část, ze které se při iteraci vytvářejí `unsigned integer` se samými nulami nebo jedničkami. S těmito hodnotami se posléze pracuje ve výpočtu.

### 4.7.1 Pseudokód simulace LUTu

```
ALGORITMUS SIMULACE LUTU (VECTOR VEKTOR_IN ,LUT L)
    IF(LUT JE DEFINICE KONSTANTY) RETURN  PÁR PODLE TYPU(F,R);
    FOR(PROCHÁZEJ TERMY)
        BEGIN
            FOR(PROJDI TERM)
                BEGIN
                VYSLD = VYPOČITEJ DATOVOU ČÁST;
                VYSLN = VYPOČÍTEJ NEURČENÉ STAVY;
                END;
        VYSLDC = VYSLD ⊗ VYSLN; //DOPOČÍTÁNÍ
        SLOUČI S MINULÝM VÝSLEDKEM;
        END;
    IF(JEDNÁ SE O TYP R) PROVEĎ INVERZI VÝSLEDKU;
RETURN VYSL<VYSLD,VYSLN>;
END-ALGORITMU;
```

Výsledná časová náročnost tohoto algoritmu je $\Theta(n \times m)$, $n$ reprezentuje počet termů LUTu a $m$ jejich velikost.

Pouze ještě doplním, že inicializace výsledku se nemůže provést na $\langle 0,0 \rangle$, protože by se tím zadal jako prvotní stav neurčitý stav. Inicializace se musí provádět na $0$, tedy na $\langle 0,0xffffffff \rangle$.

## 4.8 Formát ukládání vstupů a výstupů BLIFu

V průběhu implementace jsem byl postaven před nutnost definovat formát pro vstup a výstup vektorů. Stanovil jsem dva typy ukládání, které mohou složit jako vstupní i výstupní. Jako symbol pro neurčený stav jsem zvolil „-“.

### 4.8.1 Jednoduchý formát

První formát má jednoduchou strukturu skládající se z dvou hodnot, jenž určí velikost ukládaných vektorů a druhé číslo reprezentující počet vektorů. Dále již následují samotné vektory.

```
10
5
0-10-0-11-
1001011--1
------00--
-0110-----
010--0----
```

**Obr. 4.8.1 Jednoduchý typ ukládání vektorů**

Ukázka jednoduchého typu ukládá vektory o velikosti 10 a v počtu 5. Do tohoto typu lze uložit jak vygenerované vstupní soubory, tak výstupní.

### 4.8.2 Podrobnější formát

Podrobnější formát má první dva řádky definující velikost a počet vektorů stejné, dále však informuje o názvu modelu a jménech vstupních a výstupních signálů. Obsahuje také vstupní a výstupní vektory napsané vedle sebe a oddělené mezerou. Tento formát lze zpětně použít i jako vstupní, ale načte se zněj opět jen množina vstupů.

```
10
5
.model x2
.inputs a b c d e f g h i j
.outputs k l m n o p q
0-10-0-11- --01---
1001011--1 --01---
------00-- ----111
-0110----- ---1---
010--0---- ---1---
.end
```

**Obr. 4.8.2 Jednoduchý typ ukládání vektorů**

Na ukázce je znovu ten samý soubor, jen v podrobnějším zápise. Na prvních řádcích máme opět informace o vstupních vektorech o velikosti 10 a v počtu 5. Dále zde máme název simulovaného modelu `x2` , názvy vstupů `a, b, c, d, e, f, g, h, i, j` a výstupů `k, l, m, n, o, p, q`. Vše je zakončeno `.end`.

Tento formát lze použít pro přehledné uložení výsledků simulace.

# 5 Testování

Testování proběhlo na MCNC [2] testovacích kombinačních i sekvenčních obvodech. Obvody, které se jádru nepodařilo načíst z důvodů poškození: k2.blif, ile.blif, tbk.blif.

## 5.1 Porovnání s existujícím řešením

Jako referenční řešení pro moji bakalářskou práci byla zvolena diplomová práce Simulace logických obvodů v SystemC pana Miroslava Maněny [5]. Konkrétně se jednalo o simulační program Cirsim, který umí řešit kombinační obvody BLIF.

Při prvních testech jsem zjistil různé výsledky simulace stejných vektorů v mém a jeho podání. Po krátkém zkoumání jsem došel k závěru, že program Cirsim má špatné výsledky. Toto jsem dokázal na malém triviálním příkladě, který jsem poté poslal mému vedoucímu práce. Ten mi mé závěry potvrdil.

Z těchto důvodů nemám kompletní srovnání s exitujícím řešením, ale domnívám se, že moje práce je dobře, neboť jsem program, výsledky i postupy řešení podrobně konzultoval s mým vedoucím práce.

## 5.2 Podmínky testování

Testování se konalo na notebooku s procesorem Mobile DualCore AMD Turion 64 X2 TL-58, 1900 MHz (9.5 x 200) s operační pamětí 2048 MB (DDR2-667 DDR2 SDRAM). Na notebooku běžel operační systém Windows Vista Home premium 32b SP2, plně naběhnutý. Všechny aplikace byly vypnuty a frekvence procesoru byla pevně nastavena na 1900MHz.

Měření času probíhá přímo prostředky v kódu, měří se pouze čas simulace. Nejsou zahrnuty časové nároky na generování vstupních vektorů ani na načtení a předzpracování souboru.

Testovalo se na větším množství vektorů, čímž se snížila chyba měření.

Následující dvě kapitoly obsahují tabulky s výsledky měření simulace jednotlivých souborů.

## 5.3 Testy kombinačních obvodů

Testy byly provedeny náhodnými vektory v počtu **10000**.

| Soubor | Počet vstupů | Počet výstupů | Počet LUTů | Celkový čas [ms] | Čas na jeden vektor [µs] |
|---|---|---|---|---|---|
| 9symml.blif | 9 | 1 | 44 | 312,00 | 31,20 |
| alu2.blif | 10 | 6 | 59 | 702,00 | 70,20 |
| alu4.blif | 14 | 8 | 112 | 1294,81 | 129,48 |
| apex6.blif | 135 | 99 | 238 | 3447,62 | 344,76 |
| apex7.blif | 49 | 37 | 59 | 967,21 | 96,72 |
| b1.blif | 3 | 4 | 6 | 46,80 | 4,68 |
| b9.blif | 41 | 21 | 117 | 748,81 | 74,88 |
| C1355.blif | 41 | 32 | 546 | 2215,21 | 221,52 |
| C17.blif | 5 | 2 | 6 | 46,80 | 4,68 |
| C1908.blif | 33 | 25 | 880 | 3229,22 | 322,92 |
| C2670.blif | 233 | 140 | 1193 | 8611,26 | 861,13 |
| C3540.blif | 50 | 22 | 1669 | 6240,04 | 624,00 |
| C432.blif | 36 | 7 | 160 | 764,41 | 76,44 |
| C499.blif | 41 | 32 | 202 | 1185,61 | 118,56 |
| C5315.blif | 178 | 123 | 2307 | 11684,50 | 1168,45 |
| C6288.blif | 32 | 32 | 2416 | 9235,26 | 923,53 |
| C7552.blif | 207 | 108 | 3512 | 16239,70 | 1623,97 |
| C8.blif | 28 | 18 | 48 | 514,80 | 51,48 |
| C880.blif | 60 | 26 | 383 | 1809,61 | 180,96 |
| cc.blif | 21 | 20 | 33 | 343,20 | 34,32 |
| cm138a.blif | 6 | 8 | 9 | 124,80 | 12,48 |
| cm150a.blif | 21 | 1 | 16 | 187,20 | 18,72 |

**Tabulka 5-1 Měření simulace kombinačních obvodů (1)**

| Soubor | Počet vstupů | Počet výstupů | Počet LUTů | Celkový čas [ms] | Čas na jeden vektor [µs] |
|---|---|---|---|---|---|
| cm151a.blif | 12 | 2 | 9 | 109,20 | 10,92 |
| cm152a.blif | 11 | 1 | 1 | 78,00 | 7,80 |
| cm162a.blif | 14 | 5 | 19 | 171,60 | 17,16 |
| cm163a.blif | 16 | 5 | 16 | 171,60 | 17,16 |
| cm42a.blif | 4 | 10 | 13 | 93,60 | 9,36 |
| cm82a.blif | 5 | 3 | 6 | 62,40 | 6,24 |
| cm85a.blif | 11 | 3 | 24 | 140,40 | 14,04 |
| cmb.blif | 16 | 4 | 14 | 171,60 | 17,16 |
| comp.blif | 32 | 3 | 55 | 421,20 | 42,12 |
| cordic.blif | 23 | 2 | 102 | 452,40 | 45,24 |
| count.blif | 35 | 16 | 47 | 452,40 | 45,24 |
| cu.blif | 14 | 11 | 23 | 234,00 | 23,40 |
| dalu.blif | 75 | 16 | 1131 | 6130,84 | 613,08 |
| decod.blif | 5 | 16 | 18 | 156,00 | 15,60 |
| des.blif | 256 | 245 | 926 | 22308,10 | 2230,81 |
| example2.blif | 85 | 66 | 90 | 1528,81 | 152,88 |
| f51m.blif | 8 | 8 | 16 | 202,80 | 20,28 |
| frg1.blif | 28 | 3 | 3 | 421,20 | 42,12 |
| frg2.blif | 143 | 139 | 526 | 5475,64 | 547,56 |
| cht.blif | 47 | 36 | 36 | 702,01 | 70,20 |
| i1.blif | 25 | 16 | 33 | 327,60 | 32,76 |
| i10.blif | 257 | 224 | 2497 | 16972,90 | 1697,29 |

**Tabulka 5-2 Měření simulace kombinačních obvodů (2)**

| Soubor | Počet vstupů | Počet výstupů | Počet LUTů | Celkový čas [ms] | Čas na jeden vektor [µs] |
|---|---|---|---|---|---|
| i2.blif | 201 | 1 | 36 | 3198,02 | 319,80 |
| i3.blif | 132 | 6 | 70 | 1716,01 | 171,60 |
| i4.blif | 192 | 6 | 94 | 2839,22 | 283,92 |
| i5.blif | 133 | 66 | 199 | 2636,42 | 263,64 |
| i6.blif | 138 | 67 | 344 | 3338,42 | 333,84 |
| i7.blif | 199 | 67 | 406 | 4804,83 | 480,48 |
| i8.blif | 133 | 81 | 1183 | 8580,05 | 858,01 |
| i9.blif | 88 | 63 | 353 | 3307,22 | 330,72 |
| lal.blif | 26 | 19 | 71 | 514,80 | 51,48 |
| majority.blif | 5 | 1 | 2 | 46,80 | 4,68 |
| mux.blif | 21 | 1 | 6 | 171,60 | 17,16 |
| my_adder.blif | 33 | 17 | 49 | 468,00 | 46,80 |
| pair.blif | 173 | 137 | 830 | 6645,64 | 664,56 |
| parity.blif | 16 | 1 | 15 | 124,80 | 12,48 |
| pcle.blif | 19 | 9 | 16 | 202,80 | 20,28 |
| pcler8.blif | 27 | 17 | 24 | 343,20 | 34,32 |
| pm1.blif | 16 | 13 | 31 | 265,20 | 26,52 |
| rot.blif | 135 | 107 | 243 | 3276,02 | 327,60 |
| sct.blif | 19 | 15 | 40 | 374,40 | 37,44 |
| t.blif | 5 | 2 | 3 | 62,40 | 6,24 |
| t481.blif | 16 | 1 | 2072 | 10530,10 | 1053,01 |
| tcon.blif | 17 | 16 | 16 | 202,80 | 20,28 |

**Tabulka 5-3 Měření simulace kombinačních obvodů (3)**

| Soubor | Počet vstupů | Počet výstupů | Počet LUTů | Celkový čas [ms] | Čas na jeden vektor [µs] |
|---|---|---|---|---|---|
| term1.blif | 34 | 10 | 147 | 1045,21 | 104,52 |
| too_large.blif | 38 | 3 | 43 | 4789,23 | 478,92 |
| ttt2.blif | 24 | 21 | 67 | 670,80 | 67,08 |
| unreg.blif | 36 | 16 | 32 | 405,60 | 40,56 |
| vda.blif | 17 | 39 | 123 | 3556,82 | 355,68 |
| x1.blif | 51 | 35 | 35 | 1326,01 | 132,60 |
| x2.blif | 10 | 7 | 12 | 187,20 | 18,72 |
| x3.blif | 135 | 99 | 332 | 3697,22 | 369,72 |
| x4.blif | 94 | 71 | 136 | 2106,01 | 210,60 |
| z4ml.blif | 7 | 4 | 8 | 140,40 | 14,04 |

**Tabulka 5-4 Měření simulace kombinačních obvodů (4)**

## 5.4 Testy sekvenčních obvodů

Testy byly provedeny náhodnými vektory také v počtu **10000**.

| Soubor | Počet vstupů | Počet výstupů | Počet LUTů | Celkový čas [s] | Čas na jeden vektor [µs] |
|---|---|---|---|---|---|
| a.blif | 1 | 1 | 13 | 1,15 | 115,44 |
| bbara.blif | 4 | 2 | 37 | 4,82 | 482,04 |
| bbsse.blif | 7 | 7 | 52 | 8,31 | 831,49 |
| bbtas.blif | 2 | 2 | 22 | 2,29 | 229,32 |
| beecount.blif | 3 | 4 | 26 | 3,01 | 301,08 |
| cse.blif | 7 | 7 | 69 | 14,70 | 1469,53 |
| dk14.blif | 3 | 5 | 43 | 6,27 | 627,12 |
| dk15.blif | 3 | 5 | 32 | 4,09 | 408,72 |
| dk16.blif | 2 | 3 | 87 | 17,69 | 1769,05 |
| dk17.blif | 2 | 3 | 31 | 3,99 | 399,36 |
| dk27.blif | 1 | 2 | 19 | 1,90 | 190,32 |
| dk512.blif | 1 | 3 | 33 | 4,13 | 413,40 |
| ex1.blif | 9 | 19 | 99 | 19,94 | 1993,69 |
| ex2.blif | 2 | 2 | 47 | 8,69 | 868,93 |
| ex3.blif | 2 | 2 | 30 | 3,96 | 396,24 |
| ex4.blif | 6 | 9 | 45 | 6,19 | 619,32 |
| ex5.blif | 2 | 2 | 28 | 3,85 | 385,32 |
| ex6.blif | 5 | 8 | 47 | 7,53 | 753,49 |
| ex7.blif | 2 | 2 | 29 | 4,02 | 402,48 |
| keyb.blif | 7 | 2 | 69 | 20,81 | 2081,05 |
| kirkman.blif | 12 | 6 | 97 | 17,33 | 1733,17 |
| lion.blif | 2 | 1 | 12 | 1,20 | 120,12 |

**Tabulka 5-5 Měření simulace sekvenčních obvodů (1)**

| Soubor | Počet vstupů | Počet výstupů | Počet LUTů | Celkový čas [s] | Čas na jeden vektor [µs] |
|---|---|---|---|---|---|
| lion9.blif | 2 | 1 | 19 | 2,03 | 202,80 |
| mark1.blif | 5 | 16 | 59 | 7,43 | 742,57 |
| mc.blif | 3 | 5 | 23 | 2,39 | 238,68 |
| modulo12.blif | 1 | 1 | 22 | 2,17 | 216,84 |
| opus.blif | 5 | 6 | 36 | 5,34 | 533,52 |
| planet.blif | 7 | 19 | 142 | 51,34 | 5133,99 |
| planet1.blif | 7 | 19 | 142 | 51,36 | 5135,55 |
| s1.blif | 8 | 6 | 105 | 35,30 | 3530,30 |
| s1a.blif | 8 | 6 | 103 | 25,07 | 2506,94 |
| s8.blif | 4 | 1 | 17 | 1,76 | 176,28 |
| sand.blif | 11 | 9 | 133 | 46,91 | 4690,95 |
| scf.blif | 27 | 56 | 272 | 98,56 | 9856,14 |
| shiftreg.blif | 1 | 1 | 14 | 1,20 | 120,12 |
| sse.blif | 7 | 7 | 52 | 8,35 | 834,61 |
| styr.blif | 9 | 10 | 129 | 38,70 | 3870,38 |
| tav.blif | 4 | 4 | 23 | 2,39 | 238,68 |
| train11.blif | 2 | 1 | 22 | 2,56 | 255,84 |
| train4.blif | 2 | 1 | 12 | 1,22 | 121,68 |

**Tabulka 5-6 Měření simulace sekvenčních obvodů (2)**

## 5.5 Zhodnocení testů

Rozdíl mezi časy kombinačních a sekvenčních obvodů je opravdu velký. Potvrdilo se, že simulace sekvenčního obvodu je zhruba 32krát pomalejší.

U kombinačních obvodů se čas simulace pohybuje u středně malých souborů kolem 100µs. U sekvenčních se číslo zvedá na řádově ms.

# 6 Analýza vztahu počtu vstupů LUTu na rychlost simulace

## 6.1 Úvod

Hlavní myšlenkou je otázka, zda existuje nějaký vztah mezi rychlostí simulace mnou navrženým programem a počtem vstupů LUTu. Pokud nějaká závislost existuje, bude možné doporučit úpravu na určitý počet vstupů LUTu pro rychlejší simulaci.

## 6.2 Teoretický rozbor

Vztah mezi počtem vstupů LUTu a samotným počtem LUTů je nepřímá úměra. Čím menší počet vstupů, tím může LUT provádět menší rozhodovací úlohu a tudíž jich musí být v BLIFu o stejné funkčnosti více. To samozřejmě platí i opačně. S velkým počtem vstupů se snižuje počet LUTů až k extrému, kdy se dá celý logický problém popsat několika rozhodovacími tabulkami. Musíme brát v potaz, že každý LUT definuje jen jeden výstup, z toho vyplývá, že onen extrémně malý počet LUTů se rovná počtu výstupů jednotlivého BLIFu.

Podíváme-li se na základní algoritmy pro simulaci a pokusíme se nalézt místa, která by teoreticky mohla být závislá na počtu vstupů LUTu, zjistíme následující.

Algoritmus předzpracování není zahrnut v měření času simulace. I přesto, pokud se na něj podíváme zjistíme, že jednoznačně odpovědět na otázku změny času simulace není jednoduché. Když vezmeme v potaz extrémní případ, kdy jsou namapovány tři vstupy do jednotlivých LUTů, zvýší se nám jejich počet. Toto zvýšení má za následek větší množství času potřebného k opakovanému procházení jejich seznamem. Na druhou stranu nejsou takové ztráty při zkoušení skládání vstupu, protože v případě nemožnosti ztratíme čas rovný maximálně vyhledávání tří vstupů. Z toho důvodu nemůžeme jednoznačně říci, zda se předzpracování zrychlí, či nikoliv.

V simulaci struktury se skládají vstupy do jednotlivých LUTů, díky předzpracování zde nejsou žádné časové ztráty. Zde se vyhledává z celkového počtu signálů uložených v mapách s časovou složitostí $O(\log n)$, což je zde nejvyšší časová závislost. Proto rychlost této simulace je $O(x \times \log(n_{in}) + y \times \log(n_{out}) + z \times \log(n_{other}))$, kde $x$ je počet hledání ve struktuře `Map_in`, $y$ v `Map_out` a $z$ v `Map_other`. Hodnoty $n_{in}$ a $n_{out}$ nejsou závislé na změně struktury BLIFu, ostatní se zvýší s větším počtem LUTů, protože je nutné vytvářet vnitřní proměnné a častěji vyhledávat ve všech mapách. Opačný extrém, kdy existuje pouze minimální počet LUTů, má za následek úplné zmizení vnitřních proměnných, které jsou potřeba pro uložení mezivýsledku. Díky těmto skutečnostem bude tento algoritmus pomalejší při nižším počtu vstupů LUTu.

V řešícím algoritmu je situace jednoznačnější. Zde je složitost výpočtu $\Theta(n \times m)$, $n$ reprezentuje počet termů LUTu a $m$ jejich velikost. Celkem $n \times m$ se provede obtížný výpočet konstantní složitosti. Při zvýšení počtu vstupů LUTu se zvýší jeho složitost a tím i počet termů. Časová náročnost zde roste kvadraticky s růstem vstupů. Tady je nejspíše ono úzké hrdlo, které bude při měření nejvíce znát.

Algoritmy pro paralelizaci a deparalelizaci vstupních vektorů se samozřejmě nemění a závisí na počtu vstupů.

## 6.3 Testování

Za účelem získání dat jsem použil některé kombinační soubory z MCNC [2], které jsem před měřením upravil pomocí programu Mapper [5], umožňující přemapovat vstupní soubor Bench na výstupní soubor BLIF s definovaným počtem vstupů LUTu. Protože Mapper umožňuje jako vstup pouze soubory typu Bench, musel jsem použít pro převod BLIFu program p. Ing. Fišera Ph.D. Blif2Bench.

Testovací metodika byla stejná jako v předešlém měření kapitola 5.2 Podmínky testování

Testovací soubory jsem volil, aby byly zahrnuty všechny různé velikosti a větší počet velkých souborů. Tyto soubory jsem přemapoval na 3 až 12 vstupů. To sice změnilo jejich velikost, ale ta není důležitá, podle ní jsem si orientačně volil soubory.

Testování se provedlo na vektorech v počtu 1000 pro každý soubor. V tabulkách je uveden čas simulace jednoho vektoru.

| Soubor | Velikost [kB] | Počet LUTů | Edge [1)] | Cube [2)] | Level [3)] |
|---|---|---|---|---|---|
| 9symml.blif | 2,1 | 44 | 219 | 114 | 6 |
| apex6.blif | 15,5 | 238 | 860 | 480 | 8 |
| C3540.blif | 69,1 | 1669 | 2939 | 1669 | 47 |
| C432.blif | 8,3 | 160 | 336 | 178 | 17 |
| C5315.blif | 101,8 | 2307 | 4386 | 2307 | 49 |
| C7552.blif | 149,8 | 3512 | 6144 | 3512 | 43 |
| cmb.blif | 0,604 | 14 | 53 | 32 | 5 |
| dalu.blif | 52,4 | 1131 | 3035 | 2276 | 24 |
| des.blif | 157,3 | 926 | 5104 | 2620 | 5 |
| example2.blif | 4 | 90 | 329 | 176 | 6 |
| lal.blif | 2,3 | 71 | 222 | 138 | 5 |
| mux.blif | 0,671 | 6 | 32 | 46 | 3 |
| pair.blif | 28,1 | 830 | 2020 | 1636 | 18 |
| t481.blif | 100,7 | 2072 | 6823 | 4414 | 10 |
| too_large.blif | 68,4 | 43 | 603 | 1115 | 2 |
| unreg.blif | 1,3 | 32 | 112 | 80 | 2 |
| x2.blif | 0,792 | 12 | 63 | 40 | 2 |

**Tabulka 6-1 Analýza - testované soubory**

[1)] Edge - Určuje počet hran/vztahů signálů s LUTy. V našem případě toto číslo definuje počet hledání hodnot jednotlivých signálů při skládání vstupů do LUTů.

[2)] Cube - Určuje, kolik je v BLIFu termů v součtu přes všechny uzly.

[3)] Level - Maximální počet uzlů/LUTů při průchodu BLIFem. Nejdelší cesta.

Informace byly zjištěny pomocí programu ABC [7].

## 6.4 Výsledky testů

Výsledky testů potvrdily úvahy z minulé kapitoly. V některých případech bylo znát omezení ze simulace struktury, kdy snižování počtu vstupů LUTu zvyšuje jejich počet a také počet vnitřních proměnných.

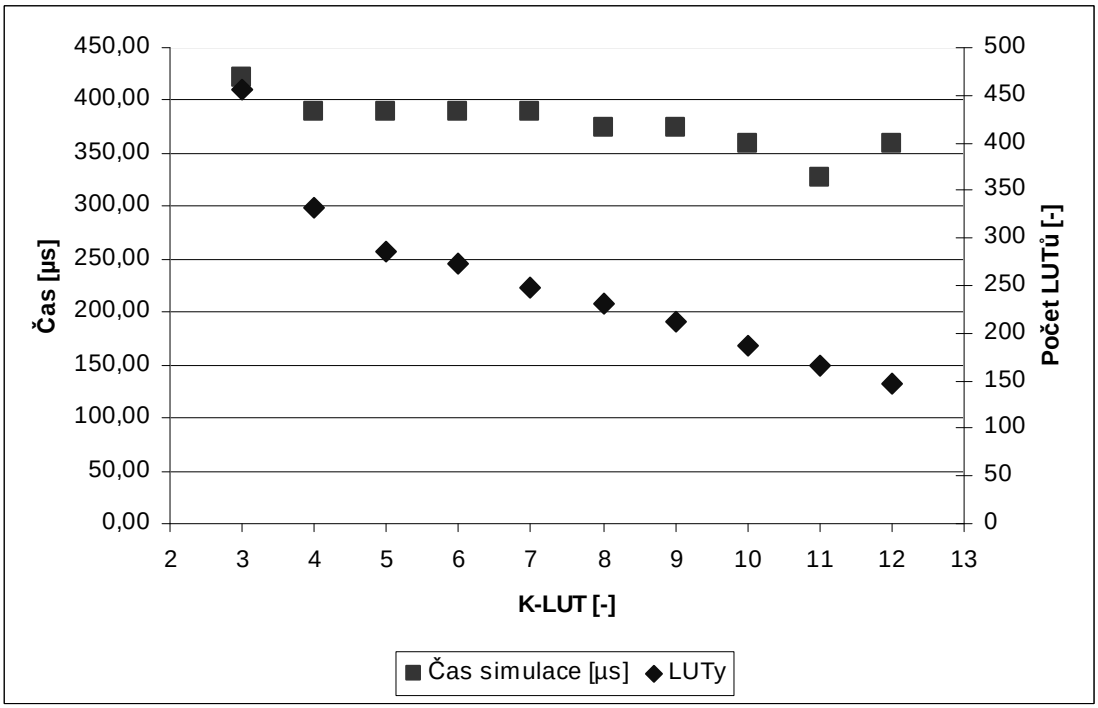
V tabulkách je obsažena hodnota určující počet vstupů LUTu, která se v jednotlivém BLIFU nejvíce vyskytuje a její procentuální zastoupení. Není to vždy počet na který se BLIF přemapovával, ale záleží na kvalitě úpravy.

### 6.4.1 Výsledky apex6.blif

Konkrétně v tomto případě. Přemapovaný soubor `apex6.blif` mající 135 vstupů a 99 výstupů.

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 356 | 455 | 421,20 | 3 | 83,08 |
| 4 | 232 | 331 | 390,00 | 4 | 68,88 |
| 5 | 187 | 286 | 390,00 | 5 | 43,71 |
| 6 | 175 | 274 | 390,00 | 5 | 34,67 |
| 7 | 149 | 248 | 390,00 | 5 | 35,08 |
| 8 | 132 | 231 | 374,40 | 5 | 29,44 |
| 9 | 113 | 212 | 374,40 | 5 | 28,30 |
| 10 | 89 | 188 | 358,80 | 5 | 31,38 |
| 11 | 67 | 166 | 327,60 | 5 | 33,73 |
| 12 | 49 | 148 | 358,80 | 5 | 37,16 |

**Tabulka 6-2 Analýza naměřených dat – apex6.blif**




**Graf 6-1 Závislost rychlosti simulace na počtu vstupů do LUTu – apex6.blif**

Z grafu je patrné zpomalení simulace při mapování na nižší počet vstupů. To je způsobeno několikanásobným zvýšením počtu LUTů a vnitřních proměnných, které musí kompenzovat vyšší složitost s poměrně velkým počtem výstupů.

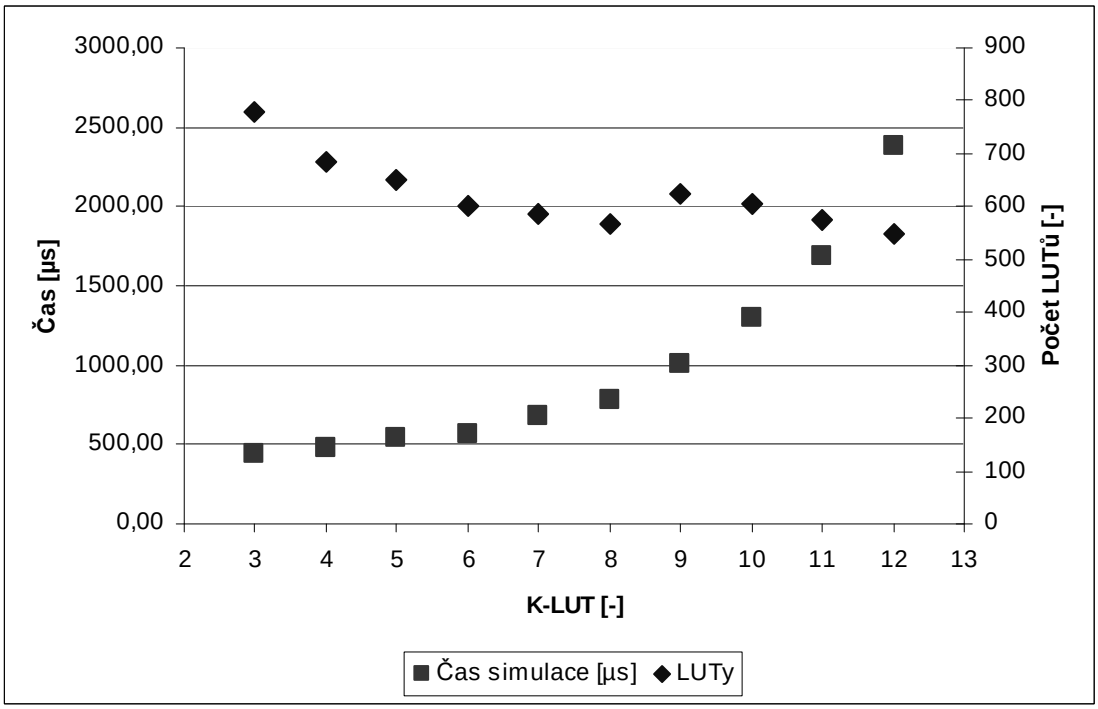
Minimálního času na simulaci bylo dosaženo při počtu vstupů 10 a 11, poté křivka opět stoupá vlivem zátěže simulace LUTu.

### 6.4.2 Výsledky C3540.blif

Další soubor `C3540.blif` s 50 vstupy a 22 výstupy.

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 758 | 780 | 436,80 | 3 | 92,69 |
| 4 | 663 | 685 | 483,60 | 4 | 84,82 |
| 5 | 629 | 651 | 546,00 | 5 | 74,04 |
| 6 | 578 | 600 | 561,60 | 6 | 56,17 |
| 7 | 566 | 588 | 686,40 | 7 | 46,26 |
| 8 | 546 | 568 | 780,01 | 8 | 37,68 |
| 9 | 603 | 625 | 1014,01 | 9 | 31,84 |
| 10 | 584 | 606 | 1294,81 | 10 | 30,36 |
| 11 | 552 | 574 | 1684,81 | 11 | 28,75 |
| 12 | 526 | 548 | 2386,82 | 12 | 27,55 |

**Tabulka 6-3 Analýza naměřených dat – C3540.blif**




**Graf 6-2 Závislost rychlosti simulace na počtu vstupů do LUTu – C3540.blif**

Zde se projeví zvyšující se časová náročnost simulace LUTu a celkový čas kvadraticky začíná růst od nejnižšího počtu vstupů LUTu. V tomto případě se neprojeví zvyšování časové

náročnosti na nižších hodnotách vstupů, protože nárůst počtu LUTů a vnitřních proměnných není tam markantní.

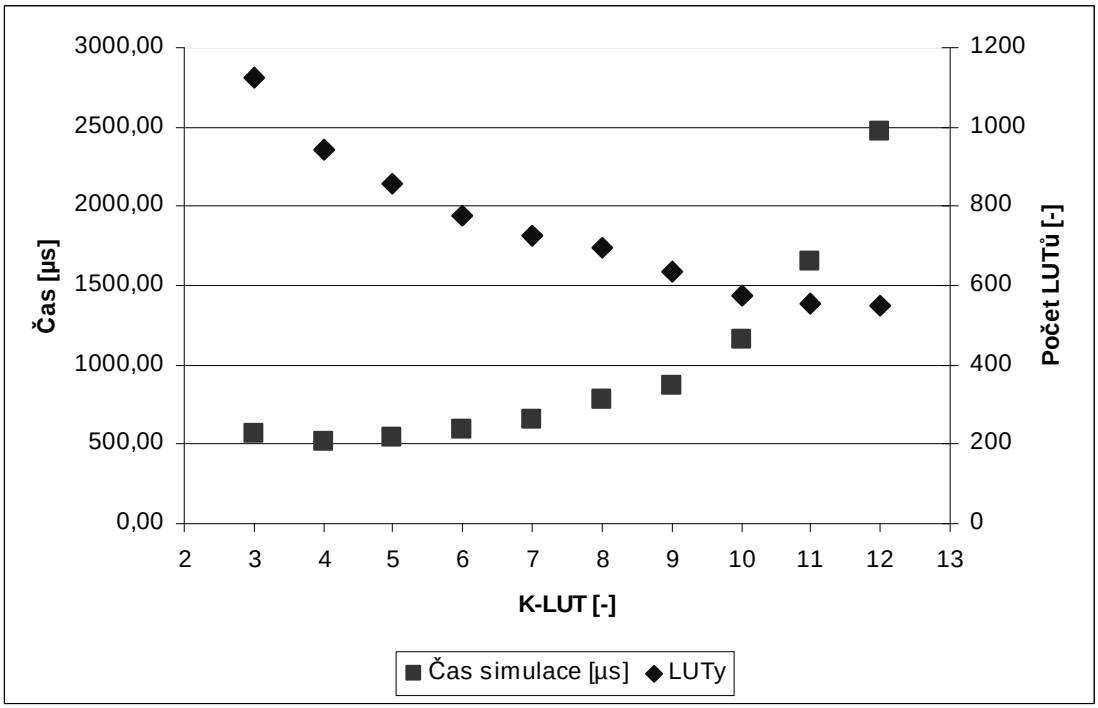
Simulace byla nejrychlejší při namapování na 3 vstupy LUTu.

### 6.4.3 Výsledky t481.blif

Soubor `t481.blif` s 50 vstupy a 22 výstupy.

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 1124 | 1125 | 561,60 | 3 | 83,11 |
| 4 | 944 | 945 | 514,80 | 4 | 72,59 |
| 5 | 858 | 859 | 546,00 | 5 | 63,80 |
| 6 | 774 | 775 | 592,80 | 6 | 55,35 |
| 7 | 725 | 726 | 655,20 | 7 | 49,17 |
| 8 | 697 | 698 | 780,01 | 8 | 50,72 |
| 9 | 632 | 633 | 873,61 | 9 | 49,92 |
| 10 | 576 | 577 | 1154,41 | 10 | 37,61 |
| 11 | 552 | 553 | 1653,61 | 11 | 22,24 |
| 12 | 549 | 550 | 2464,82 | 12 | 22,73 |

**Tabulka 6-4 Analýza naměřených dat – t481.blif**




**Graf 6-3 Závislost rychlosti simulace na počtu vstupů do LUTu – t481.blif**

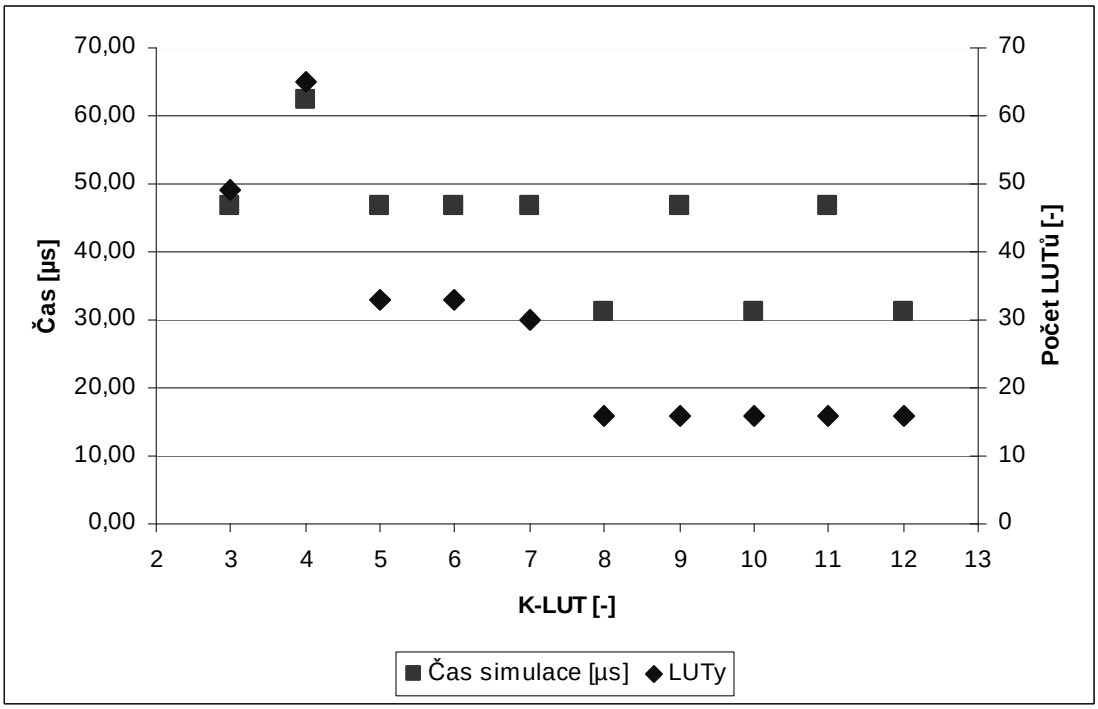
Zde je opět stejný jev jako v předcházejícím případě. Simulace se začíná zpomalovat od nejnižšího počtu vstupů a kolem počtu 9-ti začne strmý nárůst potřebného času k simulaci.

### 6.4.4 Výsledky unreg.blif

Soubor `unreg.blif` s 36 vstupy a 16 výstupy.

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 33 | 49 | 46,80 | 3 | 65,31 |
| 4 | 49 | 65 | 62,40 | 2 | 75,38 |
| 5 | 17 | 33 | 46,80 | 2 | 51,52 |
| 6 | 17 | 33 | 46,80 | 2 | 51,52 |
| 7 | 14 | 30 | 46,80 | 6 | 53,33 |
| 8 | 0 | 16 | 31,20 | 6 | 100,00 |
| 9 | 0 | 16 | 46,80 | 6 | 100,00 |
| 10 | 0 | 16 | 31,20 | 6 | 100,00 |
| 11 | 0 | 16 | 46,80 | 6 | 100,00 |
| 12 | 0 | 16 | 31,20 | 6 | 100,00 |

**Tabulka 6-5 Analýza naměřených dat – unreg.blif**




**Graf 6-4 Závislost rychlosti simulace na počtu vstupů do LUTu – unreg.blif**

`unreg.blif` reprezentuje případ, kdy si zvolíme moc jednoduchý soubor a při mapování už nelze zvýšit počet vstupů, protože jsme dosáhli stejného počtu LUTů jako je

výstupů BLIFu, což je optimální čas, a proto je simulace po dosažení tohoto bodu nejefektivnější.

Zbytek naměřených dat obsahuje Příloha C.

## 6.5 Zhodnocení výsledků

Při testech mé aplikace jsem nalezl její limity vyplývající z hlavních algoritmů. Pokud se díváme na omezení z pohledu počtu vstupů LUTu, můžeme jejich počet stanovit na dvě. První je, když se s počtem přibližujeme k minimu, které je tři. Přiblížení k tomuto extrému se projeví v zpomalení simulace. Druhé omezení přichází se zvětšováním počtu vstupů, kde náročnost simulace stoupá kvadraticky a má opravdu velký vliv na celkovou rychlost.

Není jednoduché určit obecnou pravdu, která stanoví při kolika vstupech LUTu bude simulace nejrychlejší. Záleží to opravdu na konkrétních případech. Přesto se dá obecně tvrdit, že pro soubory BLIF s menší velikostí do 20kB je vhodné použít mapování na větší počet. Pro soubory větších objemů naopak není vhodné upravovat soubor BLIF na více než šest vstupů. Všeobecně zde platí zlatá střední cesta, že při počtu od čtyř do šesti bude simulace ve většině případů rychlejší.

# 7 Závěr

Cílů stanovených v zadání bakalářské práce jsem dosáhl. Dokázal jsem naimplementovat rychlý simulátor kombinačních obvodů, který používá pro urychlení standardních technik. Paralelizace na úrovni strojového slova se ukázala jako největší způsob zrychlení, jenž však kód velice znepřehlednila. Předzpracování samo o sobě nevedlo k velkému zrychlení, ale díky němu se kód velice zjednodušil, čímž se poměrně dost zlepšil přínos předzpracování.

Bakalářská práce je schopna řešit kombinační a sekvenční obvody BLIF i s neurčenými stavy na vstupu. Úprava na řešení těchto stavů byla jedna z nejkomplikovanějších problémů, na něž jsem narazil, protože přidala do celé implementace práci s třetím stavem. Nakonec jsem ale přišel na způsob, který vše umožní.

Konečná rychlost mého řešení je na velice slušné úrovni pohybující se řádově ve stovkách mikrosekund pro menší obvody a dle složitosti roste.

V analýze vztahu počtu vstupů LUTu na rychlosti simulace jsem došel ke dvěma limitujícím faktorům, jenž limitují rychlost v extrémních počtech vstupů. Proto jsem doporučil přemapování souboru na hodnotu od čtyř do šesti, ale urychlení je u všech souborů velice individuální.

Velmi se mi osvědčil způsob implementace, kdy jsem začínal s velice jednoduchým, ale již funkčním řešením, které jsem postupně vylepšoval a přidával nové funkčnosti. Každá novější verze se dala jednoduše otestovat vůči verzi předchozí. Rozdíly mezi verzemi jsem se snažil dělat co nejmenší, aby se v případě nečekané události způsobené nevhodným zásahem do kódu bylo možné bez větších ztrát vrátit na funkční verzi. Díky tomu jsem nemusel řešit větší problémy s chybami v programu.

# 8 Použité zdroje

[1] *Service - K336 Community Web Server* [online]. 1992 [cit. 2010-04-24]. Espresso - input file format for espresso. Dostupné z WWW: <http://service.felk.cvut.cz/vlsi/prj/BoomBench/pla_c.html>.

[2] *The Donald O. Pederson Center for Electronic Systems Design* [online]. 1992 [cit. 2010-04-24]. Blif_spec_92. Dostupné z WWW: <http://embedded.eecs.berkeley.edu/Alumni/teh/research/papers/blif_spec_92.pdf>.

[3] *Cplusplus.com : The C++ Resources Network* [online]. 2.3. c2009 [cit. 2010-04-24]. Dostupné z WWW: <http://cplusplus.com/>.

[4] S. Yang. Logic Synthesis and Optimization Benchmarks User Guide, Technical Report 1991-IWLS-UG-Saeyang, MCNC, Research Triangle Park, NC, January, 1991

[5] MANĚNA, Miroslav. *Simulace logických obvodu v SystemC*. [s.l.], 2008. 53 s. Diplomová práce. ČVUT. Dostupné z WWW: <https://service.felk.cvut.cz/vlsi/dip/Manena08/>.

[6] KARŠULÍN, Miroslav. *Mapování logických obvodů do FPGA*. [s.l.], 2008. 53 s. Bakalářská práce. ČVUT. Dostupné z WWW: <https://service.felk.cvut.cz/vlsi/dip/Karsulin08/>.

[7] *Electrical Engineering & Computer Sciences | EECS at UC Berkeley* [online]. July 29, 2005, October 4, 2007 [cit. 2010-04-27]. ABC : A System for Sequential Synthesis and Verification. Dostupné z WWW: <http://www.eecs.berkeley.edu/~alanmi/abc/>.

# A. Uživatelská příručka

## a. KSim (PLA)

Při spuštění KSim.exe je význa k načtení souboru PLA, poté se již zadává ručně vstupní vektor potvrzený klávesou ester. Okamžitě se zobrazí k němu náležící výstupní vektor.

## b. BlifSim (BLIF)

Program BlifSim.exe se ovládá dvěma způsoby. První způsob jsou parametry programu.

```
Parametry:
1. soubor Blif
   "?" nápověda
2. vstupní soubor
   jen vstupní soubor s vektory
   "-gf" pro generování kompletní tabulky stavů
   "-g 500" generovaní určitého počtu vektorů bez DC
   "-gdc 500" generování s neurčitymi stavy
3. výstupní soubor
   "-sf out.txt" uloží vstupy a výstupy proti sobě
   "-s out.txt" uloží jen výstupy
   "-t" testovací mod. Bez výpisu podrobností o blifu a bez
uložení výsledku.
 Pokud není zadán vystupní soubor, tak out.txt
```

Druhý způsob nastane tehdy, pokud program neobsahuje parametry nebo pouze jeden, tj. vstupní soubor Blifu, zobrazí se menu s těmito možnostmi:

```
a) Vygenerovat nove vstupy.
b) Nacist vstupy ze souboru.
c) Zadat vstup rucne.
d) Napoveda.
x) Konec.
```

Po hlavním výběru je uživatel dále prováděn různými nastaveními simulace a ukládáním dat.

# B. Seznam a popis metod implementace

## a. KSim (PLA)

### Hlavní metody

| Název metody | Parametry | Návratová hodnota | Význam |
|---|---|---|---|
| bitSimul | PLA & vstupPLA<br>string iStr | string | Hlavní metoda simulace PLA. |
| appendStringOut | string a<br>string b<br>PLA_type type | string | Skládání dvou stringů, jeden hlavní výstup a druhý výstup termu. |
| setOut | string in<br>PLA_type type<br>int length | string | Závěrečné doplnění nedefinovaných hodnot. |

**Tabulka B-1 Popis hlavních metod KSim (PLA)**

### Vedlejší metody

| Název metody | Parametry | Návratová hodnota | Význam |
|---|---|---|---|
| iTermToBit | string IM | bitvector | Převede string na bitvector obsahující datovou část. |
| iTermToBitDC | string IM | bitvector | Také převede, ale výsledný vektor bude obsahovat neurčenou část. |

**Tabulka B-2 Popis vedlejších metod KSim (PLA)**

## b. BlifSim (BLIF)

### Hlavní řešící algoritmy

| Název: | startBlifSimul |
|---|---|
| Vstupní parametry: | Blif & bl, list<bitvector> * inList |
| Výstupní parametry: | list<bitvector> * |
| Význam: | Provádí paralelizaci vstupních dat a volá simulaci struktury BLIF. |

**Tabulka B-3 Popis metody startBlifSimul**

| Název: | blifSimul |
|---|---|
| Vstupní parametry: | Blif * bl<br>std::map<string,pair<unsigned int,unsigned int>> *mapIn<br>std::map<string,pair<unsigned int,unsigned int>> *mapLatch=0 |
| Výstupní parametry: | std::map<string,pair<unsigned int,unsigned int>> * |
| Význam: | Simulace struktury BLIF. Skládá vstup do jednotlivých LUTů a volá metodu pro jejich řešení. |

**Tabulka B-4 Popis metody blifSimul**

| Název: | SoPlaSimul |
|---|---|
| Vstupní parametry: | std::vector <pair<unsigned int,unsigned int>> * iBitV<br>SoPLA vstupPLA |
| Výstupní parametry: | std::pair<unsigned int,unsigned int> |
| Význam: | Simulace LUTu. |

**Tabulka B-5 Popis metody SoPlaSimul**

| Název: | predzpracovani |
|---|---|
| Vstupní parametry: | Blif * bl |
| Výstupní parametry: | void |
| Význam: | Předzpracuje BLIF pro rychlejší simulaci. |

**Tabulka B-6 Popis metody predzpracovani**

### Vedlejší obslužné metody

| Název metody | Parametry | Návratová hodnota | Význam |
|---|---|---|---|
| iTermToBit | string IM | bitvector | Převede string na bitvector obsahující datovou část. |
| iTermToBitDC | string IM | bitvector | Také převede, ale výsledný vektor bude obsahovat neurčenou část. |

**Tabulka B-7 Popis vedlejších obslužných metod**

### Metody pro ukládání s načítání dat

| Název metody | Parametry | Návratová hodnota | Význam |
|---|---|---|---|
| saveListBitvector | list<bitvector> * in<br>string fileName | void | Uloží výsledky simulace. |
| saveListBitvector | list<bitvector> * in<br>list<bitvector> * out<br>string fileName<br>Blif * b | void | Uloží výsledky simulace vedle vstupů. Jedná se o přehledný výpis. |
| loadListBitvector | string fileName,<br>unsigned int inputs | list<bitvector> | Načte vstupní vektory do simulace. |

**Tabulka B-8 Popis metod pro ukládání**

### Metody pro generování náhodných vektorů

| Název metody | Parametry | Návratová hodnota | Význam |
|---|---|---|---|
| generateInput | int inputs<br>int count<br>bool dc | list<bitvector> | Vygeneruje náhodné vektory, vstup dc určuje, jestli bude obsahovat neurčité stavy. |
| generateInputMax | int inputs | list<bitvector> | Vygeneruje úplnou tabulku vstupů s velikostí $2^{inputs}$. |

**Tabulka B-9 Popis metod pro generování vektorů**

### Pomocné metody

| Název metody | Parametry | Návratová hodnota | Význam |
|---|---|---|---|
| Napoveda | void | void | Vypíše na konzoli nápovědu k programu. |
| vypisMnozinu | list<bitvector>*mnozina | void | Vypíše list bitvektorů na konzoli. |
| vypisVector | bitvector in | void | Na konzoli vypíše jeden bitvector. |
| strToInt | string in | int | Převede string na int. Pro účely zpracování parametrů programu. |

**Tabulka B-10 Popis pomocných metod**

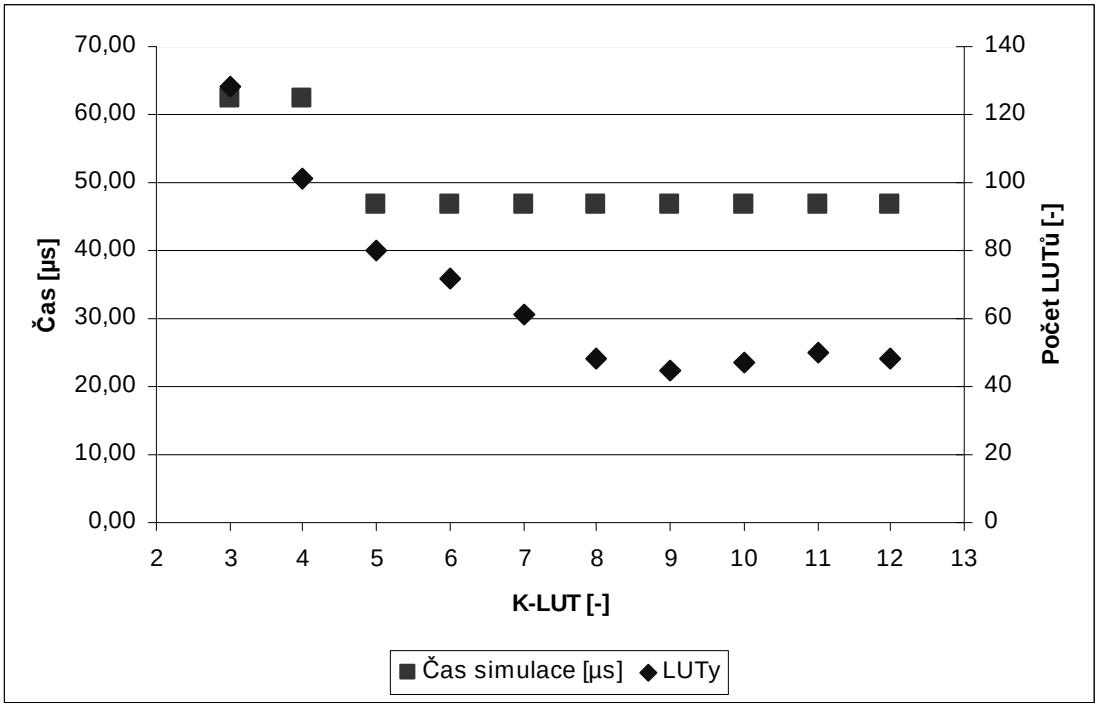
# C. Kompletní výsledky z analýzy - Kapitola 6.

Zde se nacházejí všechny naměřené hodnoty s grafy z kapitoly 6.

```
Soubor: 9symml.blif
Počet vstupů: 9
Počet výstupů: 1
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 127 | 128 | 62,40 | 3 | 74,22 |
| 4 | 100 | 101 | 62,40 | 4 | 58,42 |
| 5 | 79 | 80 | 46,80 | 5 | 43,75 |
| 6 | 71 | 72 | 46,80 | 6 | 31,94 |
| 7 | 60 | 61 | 46,80 | 2 | 24,59 |
| 8 | 47 | 48 | 46,80 | 5 | 20,83 |
| 9 | 44 | 45 | 46,80 | 5 | 28,89 |
| 10 | 46 | 47 | 46,80 | 5 | 29,79 |
| 11 | 49 | 50 | 46,80 | 4 | 28,00 |
| 12 | 47 | 48 | 46,80 | 4 | 29,17 |

**Tabulka C-1 Analýza naměřených dat – 9symml.blif**

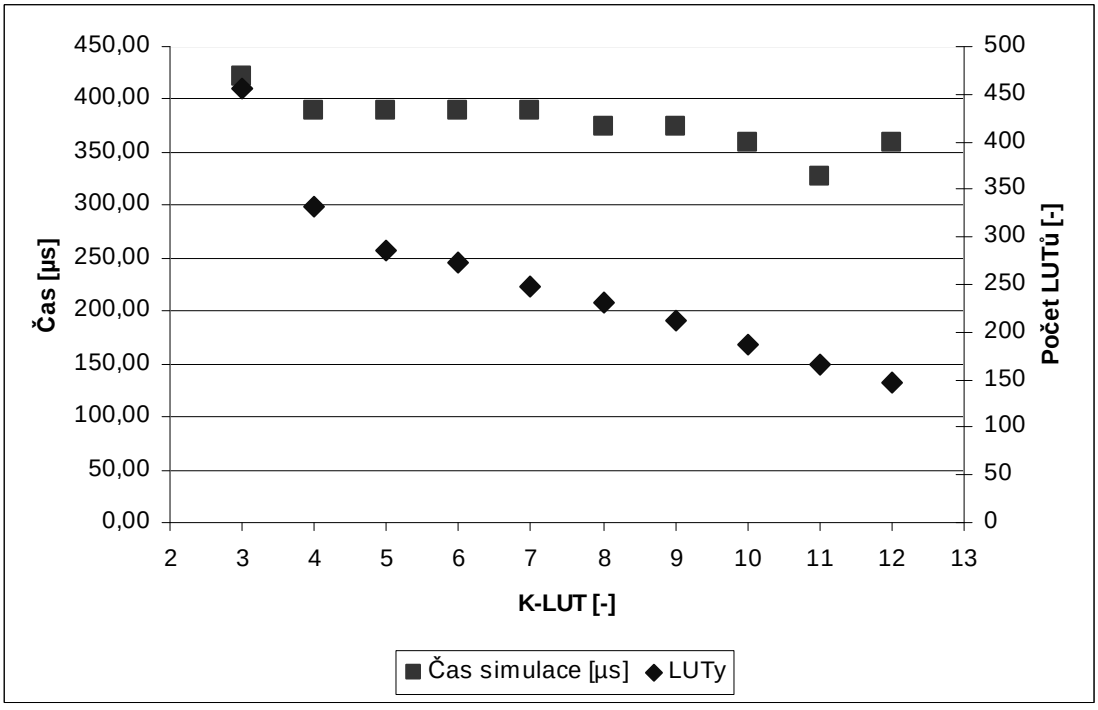



**Graf C-1 Závislost rychlosti simulace na počtu vstupů do LUTu – 9symml.blif**

```
Soubor: apex6.blif
Počet vstupů: 135
Počet výstupů: 99
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 356 | 455 | 421,20 | 3 | 83,08 |
| 4 | 232 | 331 | 390,00 | 4 | 68,88 |
| 5 | 187 | 286 | 390,00 | 5 | 43,71 |
| 6 | 175 | 274 | 390,00 | 5 | 34,67 |
| 7 | 149 | 248 | 390,00 | 5 | 35,08 |
| 8 | 132 | 231 | 374,40 | 5 | 29,44 |
| 9 | 113 | 212 | 374,40 | 5 | 28,30 |
| 10 | 89 | 188 | 358,80 | 5 | 31,38 |
| 11 | 67 | 166 | 327,60 | 5 | 33,73 |
| 12 | 49 | 148 | 358,80 | 5 | 37,16 |

**Tabulka C-2 Analýza naměřených dat – apex6.blif**

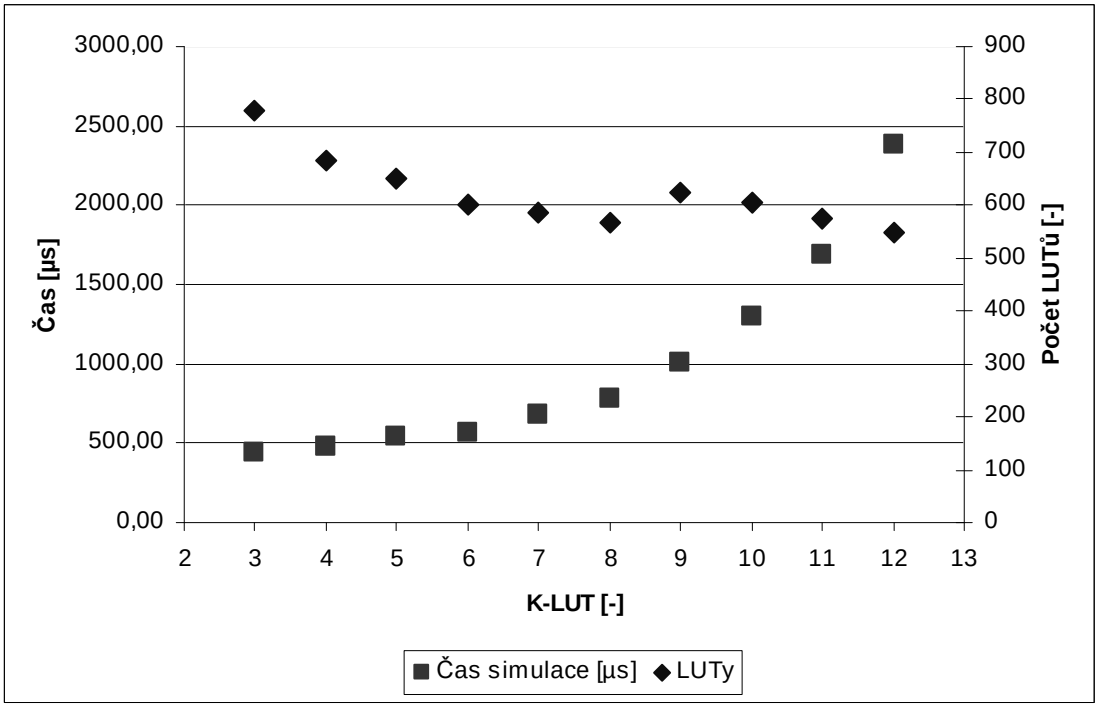



**Graf C-2 Závislost rychlosti simulace na počtu vstupů do LUTu – apex6.blif**

```
Soubor: C3540.blif
Počet vstupů: 50
Počet výstupů: 22
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 758 | 780 | 436,80 | 3 | 92,69 |
| 4 | 663 | 685 | 483,60 | 4 | 84,82 |
| 5 | 629 | 651 | 546,00 | 5 | 74,04 |
| 6 | 578 | 600 | 561,60 | 6 | 56,17 |
| 7 | 566 | 588 | 686,40 | 7 | 46,26 |
| 8 | 546 | 568 | 780,01 | 8 | 37,68 |
| 9 | 603 | 625 | 1014,01 | 9 | 31,84 |
| 10 | 584 | 606 | 1294,81 | 10 | 30,36 |
| 11 | 552 | 574 | 1684,81 | 11 | 28,75 |
| 12 | 526 | 548 | 2386,82 | 12 | 27,55 |

**Tabulka C-3 Analýza naměřených dat – C3540.blif**

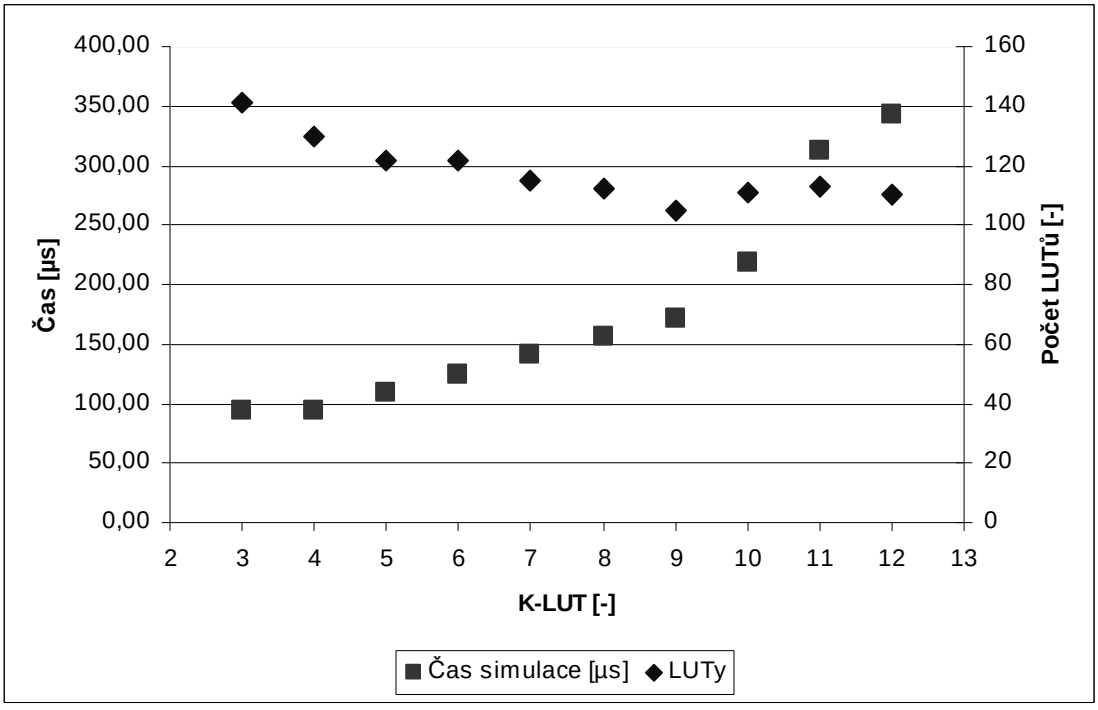



**Graf C-3 Závislost rychlosti simulace na počtu vstupů do LUTu – C3540.blif**

```
Soubor: C432.blif
Počet vstupů: 36
Počet výstupů: 7
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 134 | 141 | 93,60 | 3 | 85,82 |
| 4 | 123 | 130 | 93,60 | 4 | 84,62 |
| 5 | 115 | 122 | 109,20 | 5 | 80,33 |
| 6 | 115 | 122 | 124,80 | 6 | 77,05 |
| 7 | 108 | 115 | 140,40 | 7 | 71,30 |
| 8 | 105 | 112 | 156,00 | 8 | 64,29 |
| 9 | 98 | 105 | 171,60 | 9 | 54,29 |
| 10 | 104 | 111 | 218,40 | 10 | 60,36 |
| 11 | 106 | 113 | 312,00 | 11 | 55,75 |
| 12 | 103 | 110 | 343,20 | 12 | 61,82 |

**Tabulka C-4 Analýza naměřených dat – C432.blif**

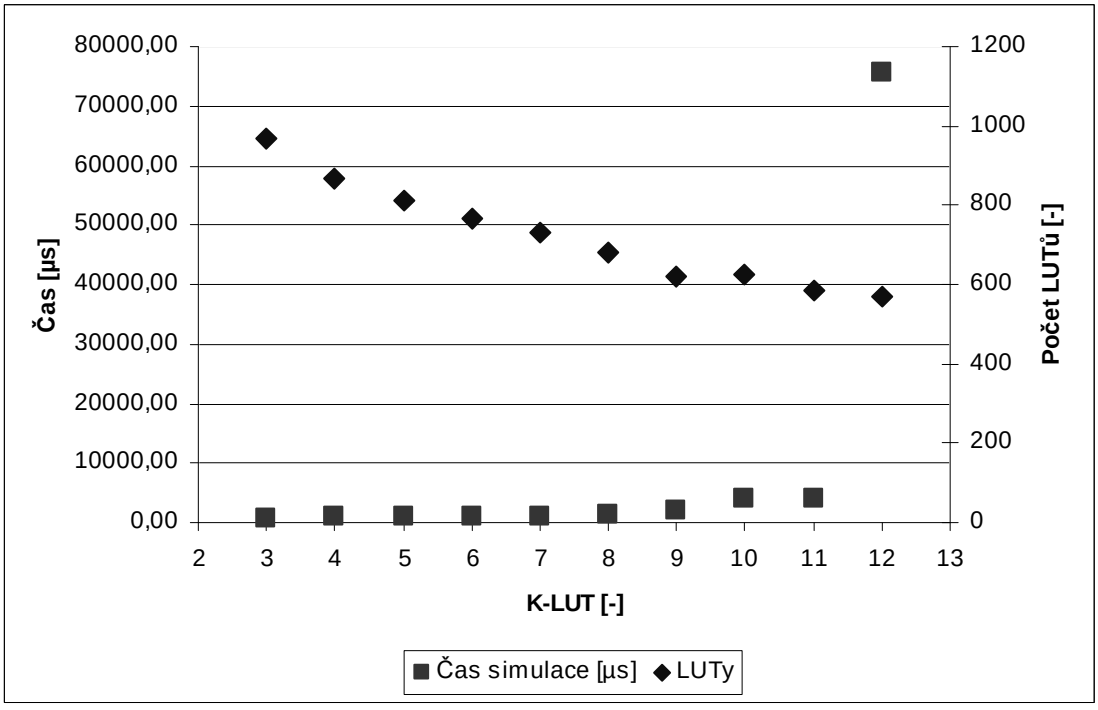



**Graf C-4 Závislost rychlosti simulace na počtu vstupů do LUTu – C432.blif**

```
Soubor: C5315.blif
Počet vstupů: 178
Počet výstupů: 123
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 846 | 969 | 826,81 | 3 | 83,59 |
| 4 | 745 | 868 | 858,01 | 4 | 62,33 |
| 5 | 689 | 812 | 858,01 | 5 | 49,51 |
| 6 | 643 | 766 | 998,41 | 6 | 45,56 |
| 7 | 606 | 729 | 1045,21 | 7 | 34,98 |
| 8 | 556 | 679 | 1185,61 | 8 | 30,04 |
| 9 | 498 | 621 | 2090,41 | 9 | 26,09 |
| 10 | 502 | 625 | 4102,83 | 3 | 23,04 |
| 11 | 462 | 585 | 3962,43 | 3 | 21,71 |
| 12 | 446 | 569 | 75613,70 | 3 | 21,62 |

**Tabulka C-5 Analýza naměřených dat – C5315.blif**

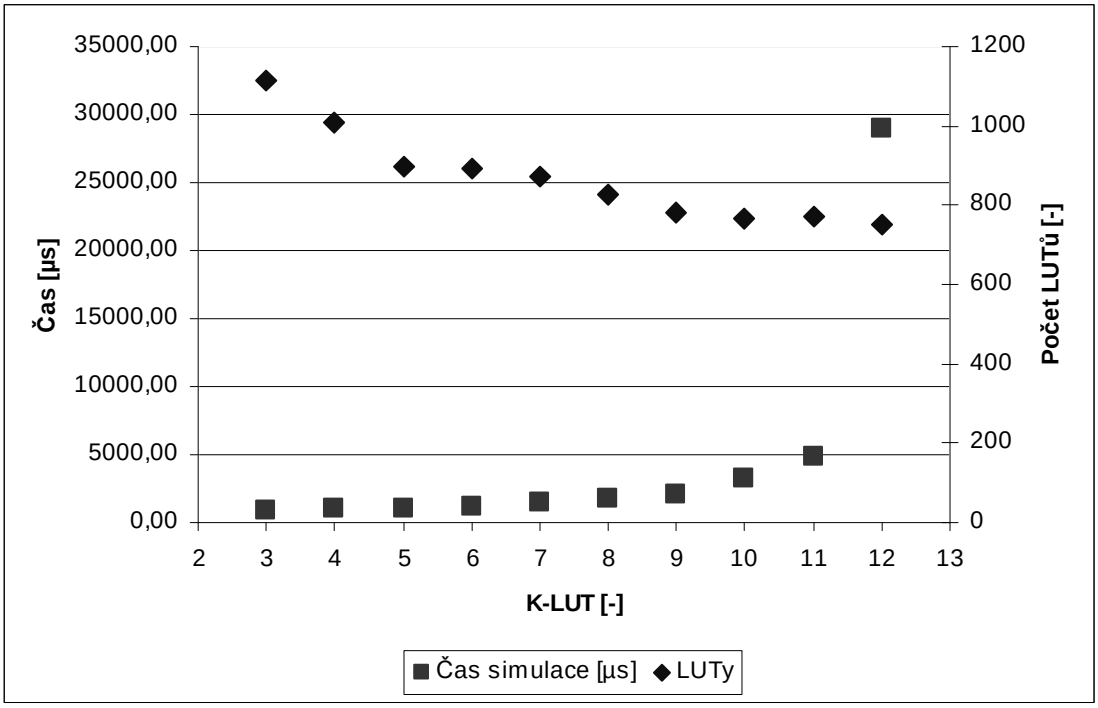



**Graf C-5 Závislost rychlosti simulace na počtu vstupů do LUTu – C5315.blif**

```
Soubor: C7552.blif
Počet vstupů: 207
Počet výstupů: 108
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 1007 | 1115 | 889,21 | 3 | 82,24 |
| 4 | 898 | 1006 | 982,81 | 4 | 67,20 |
| 5 | 792 | 900 | 1045,21 | 5 | 58,33 |
| 6 | 782 | 890 | 1170,01 | 6 | 45,17 |
| 7 | 766 | 874 | 1404,01 | 7 | 38,90 |
| 8 | 719 | 827 | 1731,61 | 8 | 32,53 |
| 9 | 675 | 783 | 2106,01 | 9 | 30,01 |
| 10 | 659 | 767 | 3198,02 | 10 | 27,38 |
| 11 | 665 | 773 | 4882,83 | 11 | 23,16 |
| 12 | 643 | 751 | 29000,60 | 2 | 21,84 |

**Tabulka C-6 Analýza naměřených dat – C7552.blif**

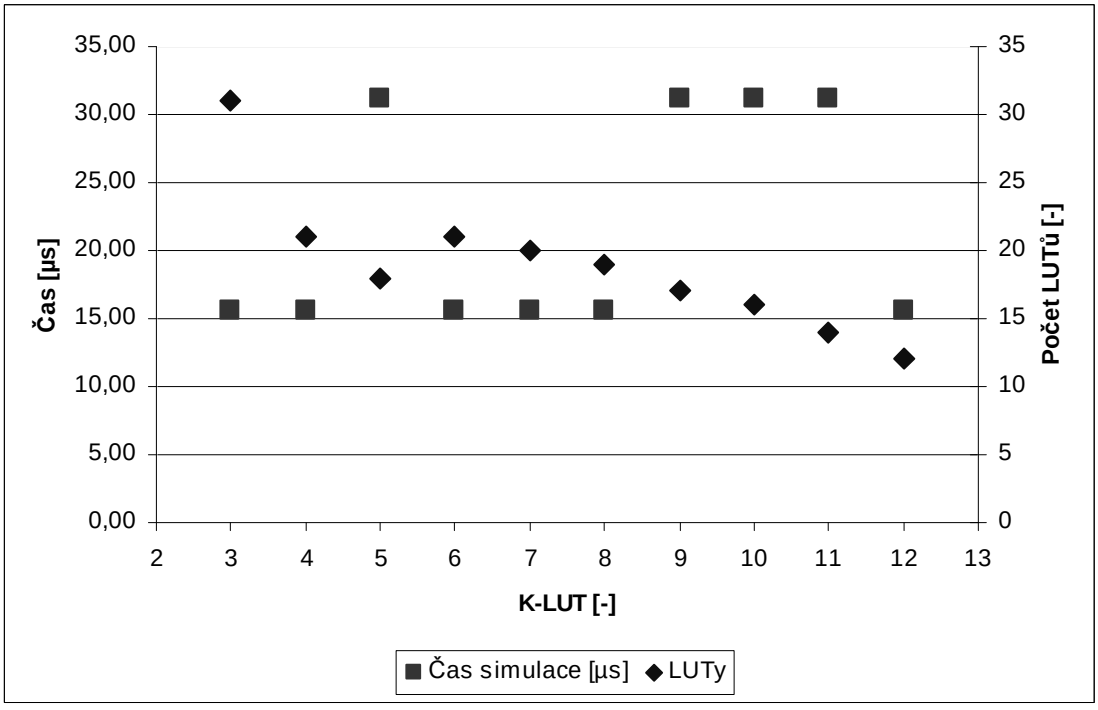



**Graf C-6 Závislost rychlosti simulace na počtu vstupů do LUTu – C7552.blif**

```
Soubor: cmb.blif
Počet vstupů: 16
Počet výstupů: 4
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 27 | 31 | 15,60 | 3 | 58,06 |
| 4 | 17 | 21 | 15,60 | 4 | 57,14 |
| 5 | 14 | 18 | 31,20 | 2 | 44,44 |
| 6 | 17 | 21 | 15,60 | 2 | 57,14 |
| 7 | 16 | 20 | 15,60 | 2 | 55,00 |
| 8 | 15 | 19 | 15,60 | 2 | 47,37 |
| 9 | 13 | 17 | 31,20 | 2 | 41,18 |
| 10 | 12 | 16 | 31,20 | 2 | 43,75 |
| 11 | 10 | 14 | 31,20 | 2 | 35,71 |
| 12 | 8 | 12 | 15,60 | 2 | 33,33 |

**Tabulka C-7 Analýza naměřených dat – cmb.blif**

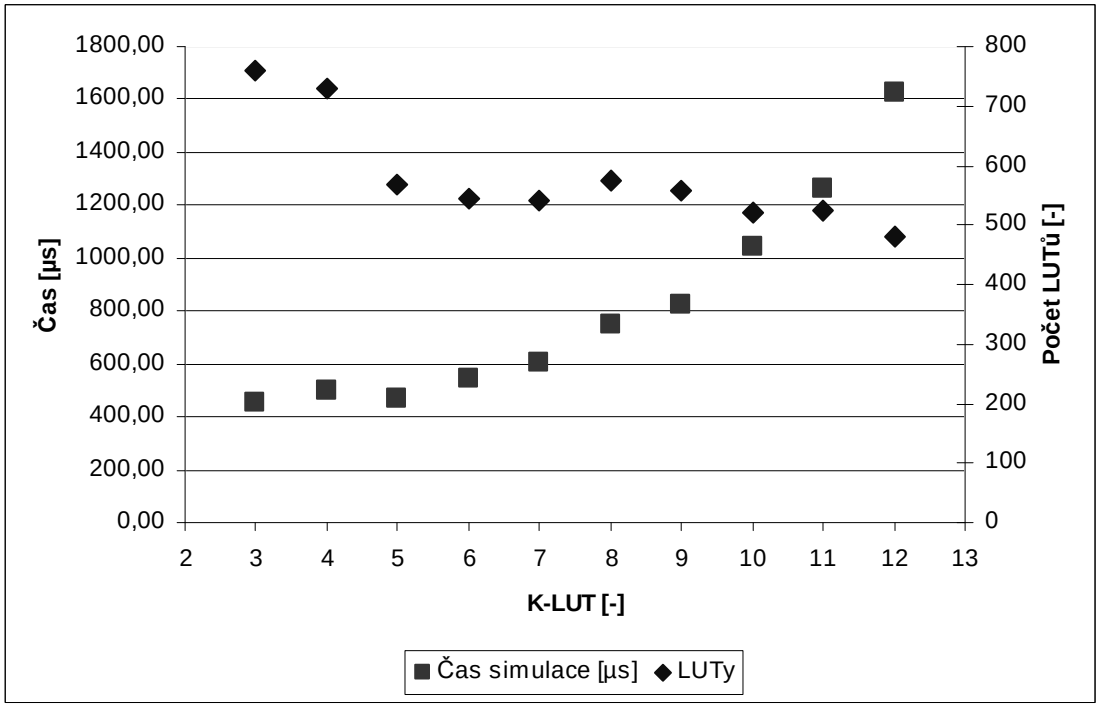



**Graf C-7 Závislost rychlosti simulace na počtu vstupů do LUTu – cmb.blif**

```
Soubor: dalu.blif
Počet vstupů: 75
Počet výstupů: 16
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 744 | 760 | 452,40 | 3 | 95,66 |
| 4 | 715 | 731 | 499,20 | 4 | 76,74 |
| 5 | 551 | 567 | 468,00 | 5 | 61,02 |
| 6 | 527 | 543 | 546,00 | 6 | 46,59 |
| 7 | 524 | 540 | 608,40 | 7 | 36,85 |
| 8 | 560 | 576 | 748,81 | 8 | 28,13 |
| 9 | 542 | 558 | 826,81 | 9 | 20,61 |
| 10 | 506 | 522 | 1045,21 | 10 | 20,69 |
| 11 | 509 | 525 | 1263,61 | 3 | 20,00 |
| 12 | 463 | 479 | 1622,41 | 11 | 20,04 |

**Tabulka C-8 Analýza naměřených dat – dalu.blif**

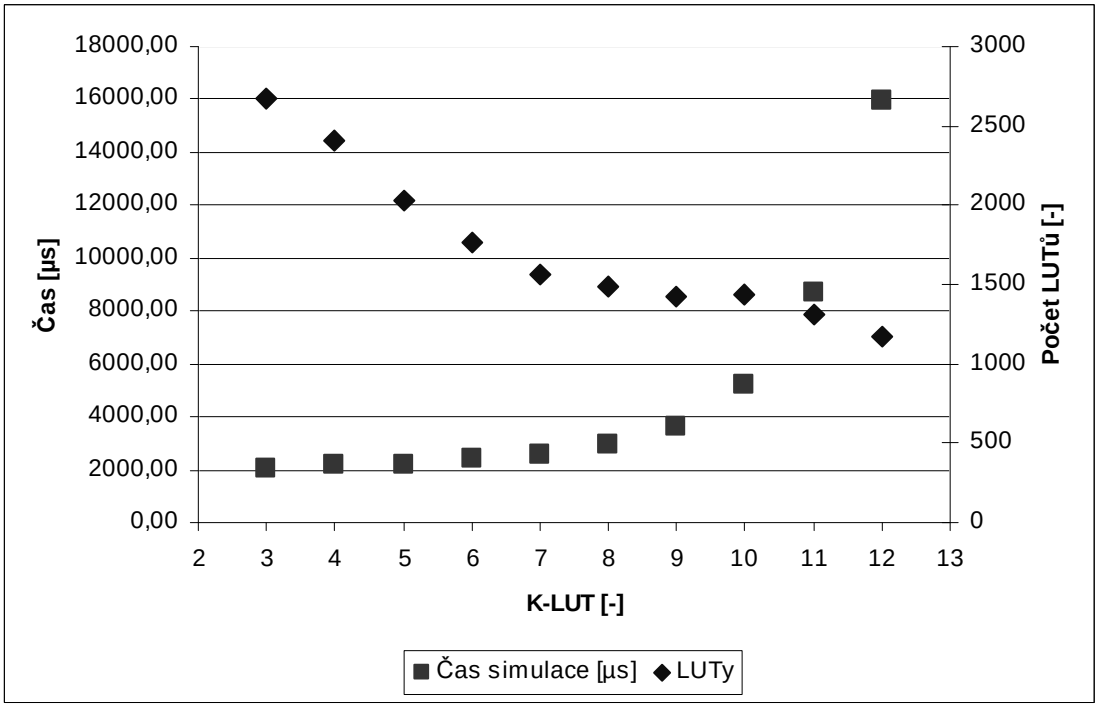



**Graf C-8 Závislost rychlosti simulace na počtu vstupů do LUTu – dalu.blif**

```
Soubor: des.blif
Počet vstupů: 256
Počet výstupů: 245
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 2423 | 2668 | 2028,01 | 3 | 96,10 |
| 4 | 2157 | 2402 | 2215,21 | 4 | 82,72 |
| 5 | 1785 | 2030 | 2199,61 | 5 | 65,27 |
| 6 | 1514 | 1759 | 2402,42 | 6 | 72,37 |
| 7 | 1319 | 1564 | 2558,42 | 7 | 60,04 |
| 8 | 1243 | 1488 | 2917,22 | 8 | 48,86 |
| 9 | 1178 | 1423 | 3666,02 | 9 | 27,55 |
| 10 | 1191 | 1436 | 5210,43 | 6 | 23,12 |
| 11 | 1071 | 1316 | 8720,46 | 6 | 18,84 |
| 12 | 922 | 1167 | 15958,90 | 6 | 19,71 |

**Tabulka C-9 Analýza naměřených dat – des.blif**

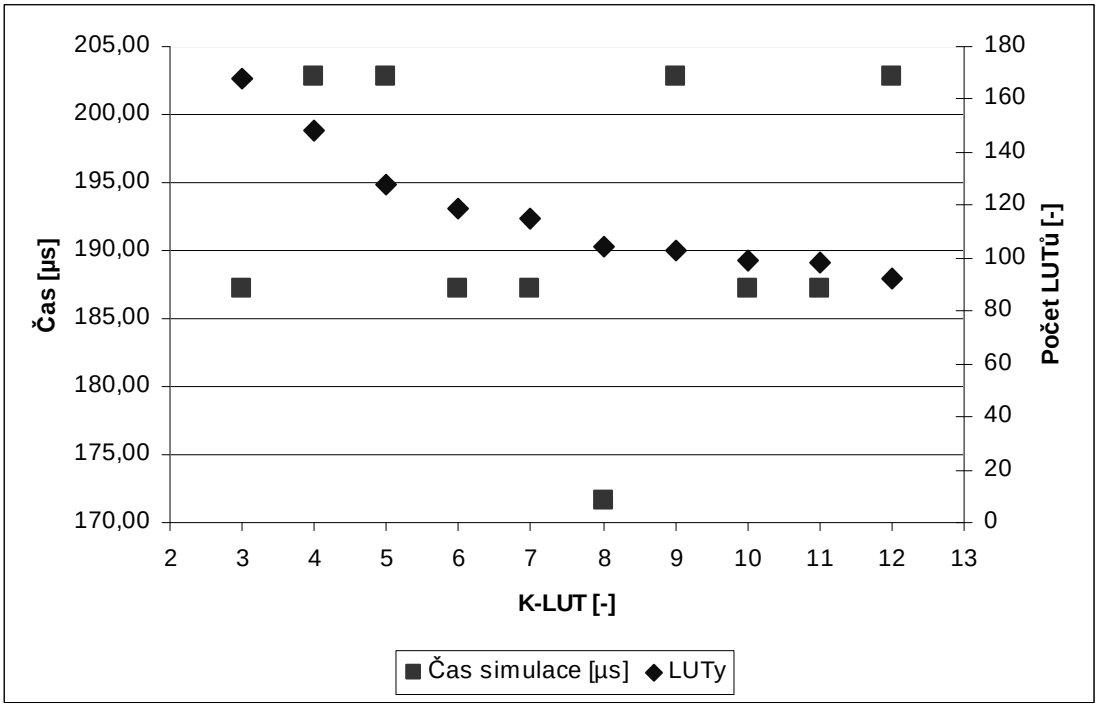



**Graf C-9 Závislost rychlosti simulace na počtu vstupů do LUTu – des.blif**

```
Soubor: example2.blif
Počet vstupů: 85
Počet výstupů: 66
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 102 | 168 | 187,20 | 3 | 87,50 |
| 4 | 82 | 148 | 202,80 | 4 | 63,51 |
| 5 | 62 | 128 | 202,80 | 5 | 57,03 |
| 6 | 53 | 119 | 187,20 | 6 | 36,13 |
| 7 | 49 | 115 | 187,20 | 7 | 34,78 |
| 8 | 38 | 104 | 171,60 | 3 | 24,04 |
| 9 | 37 | 103 | 202,80 | 3 | 28,16 |
| 10 | 33 | 99 | 187,20 | 3 | 27,27 |
| 11 | 32 | 98 | 187,20 | 3 | 26,53 |
| 12 | 26 | 92 | 202,80 | 3 | 27,17 |

**Tabulka C-10 Analýza naměřených dat – example2.blif**

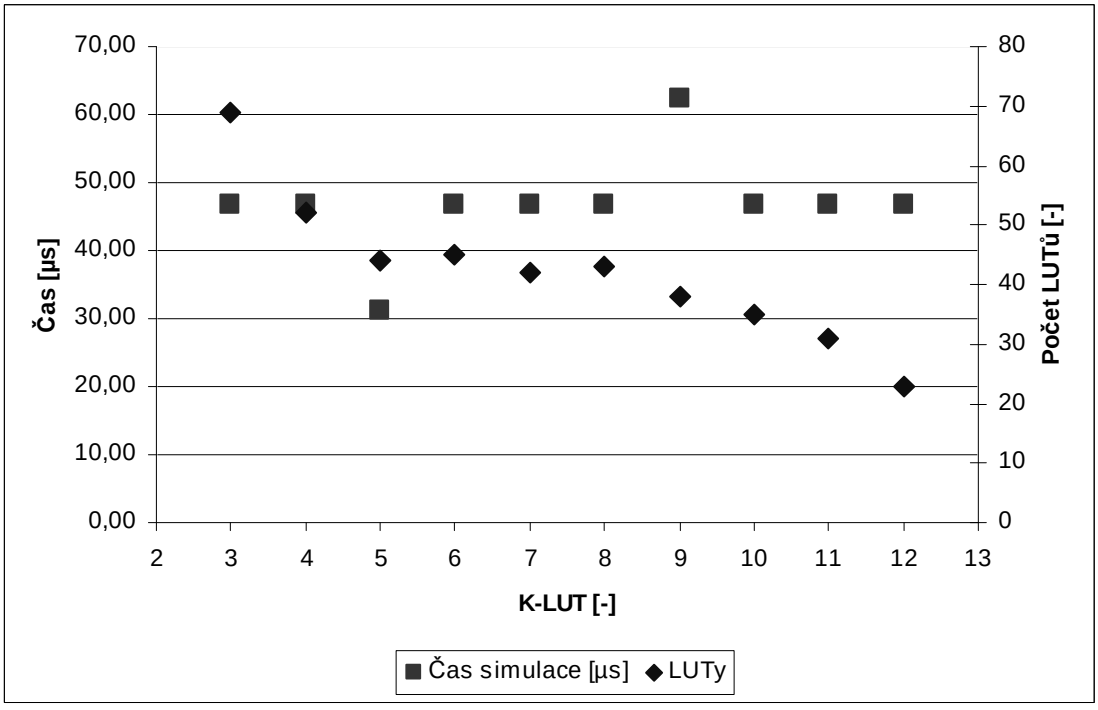



**Graf C-10 Závislost rychlosti simulace na počtu vstupů do LUTu – example2.blif**

```
Soubor: lal.blif
Počet vstupů: 26
Počet výstupů: 19
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 50 | 69 | 46,80 | 3 | 63,77 |
| 4 | 33 | 52 | 46,80 | 4 | 51,92 |
| 5 | 25 | 44 | 31,20 | 5 | 38,64 |
| 6 | 26 | 45 | 46,80 | 2 | 33,33 |
| 7 | 23 | 42 | 46,80 | 2 | 42,86 |
| 8 | 24 | 43 | 46,80 | 2 | 53,49 |
| 9 | 19 | 38 | 62,40 | 2 | 52,63 |
| 10 | 16 | 35 | 46,80 | 2 | 51,43 |
| 11 | 12 | 31 | 46,80 | 2 | 48,39 |
| 12 | 4 | 23 | 46,80 | 2 | 30,43 |

**Tabulka C-11 Analýza naměřených dat – lal.blif**

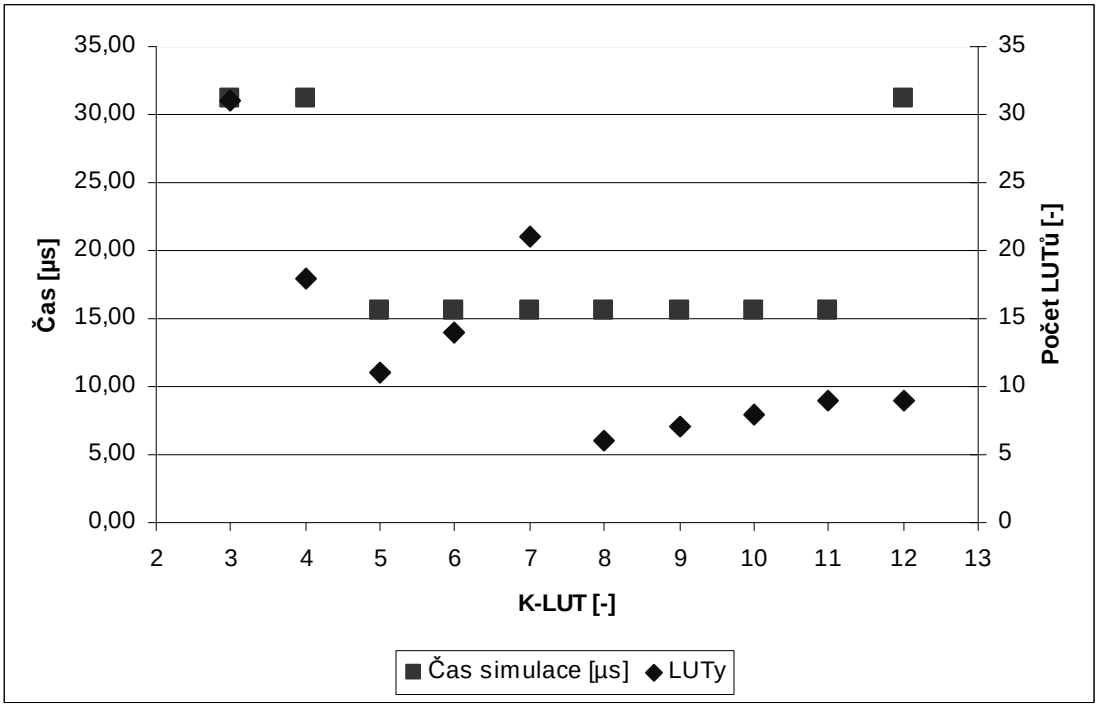



**Graf C-11 Závislost rychlosti simulace na počtu vstupů do LUTu – lal.blif**

```
Soubor: mux.blif
Počet vstupů: 21
Počet výstupů: 1
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 30 | 31 | 31,20 | 2 | 51,61 |
| 4 | 17 | 18 | 31,20 | 4 | 55,56 |
| 5 | 10 | 11 | 15,60 | 4 | 63,64 |
| 6 | 13 | 14 | 15,60 | 2 | 28,57 |
| 7 | 20 | 21 | 15,60 | 2 | 76,19 |
| 8 | 5 | 6 | 15,60 | 7 | 66,67 |
| 9 | 6 | 7 | 15,60 | 4 | 28,57 |
| 10 | 7 | 8 | 15,60 | 4 | 37,50 |
| 11 | 8 | 9 | 15,60 | 4 | 44,44 |
| 12 | 8 | 9 | 31,20 | 4 | 66,67 |

**Tabulka C-12 Analýza naměřených dat – mux.blif**

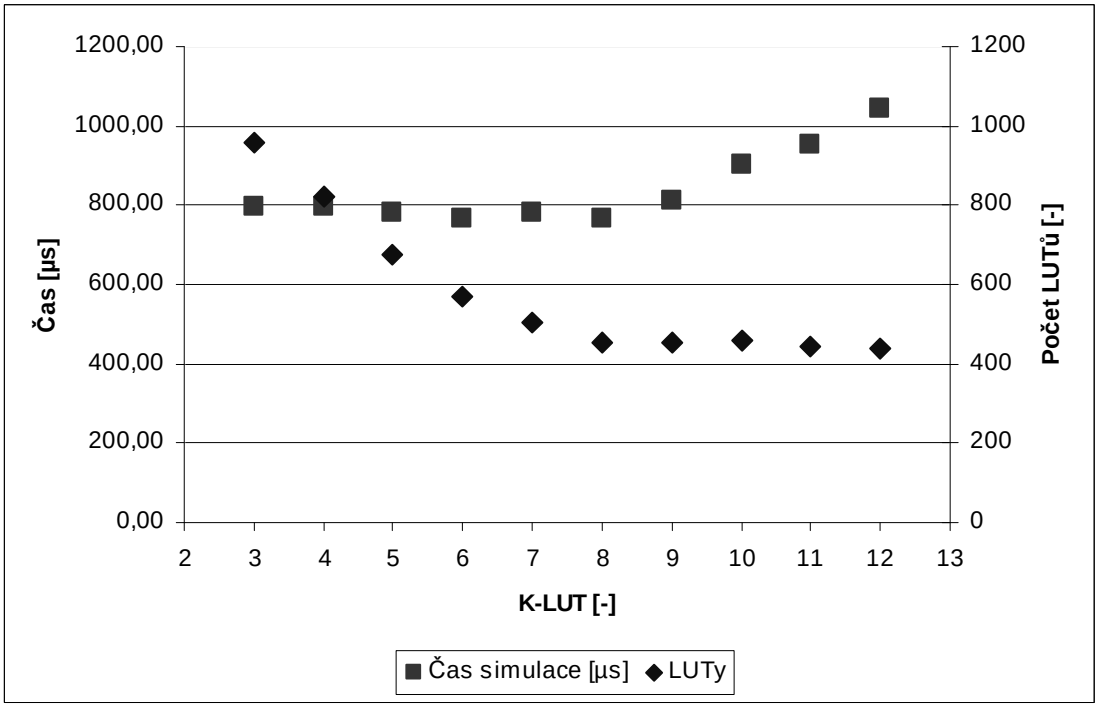



**Graf C-12 Závislost rychlosti simulace na počtu vstupů do LUTu – mux.blif**

```
Soubor: pair.blif
Počet vstupů: 173
Počet výstupů: 137
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 819 | 956 | 795,61 | 3 | 84,52 |
| 4 | 683 | 820 | 795,61 | 4 | 74,51 |
| 5 | 538 | 675 | 780,01 | 5 | 66,52 |
| 6 | 434 | 571 | 764,41 | 6 | 58,67 |
| 7 | 367 | 504 | 780,01 | 7 | 49,60 |
| 8 | 319 | 456 | 764,41 | 8 | 37,06 |
| 9 | 316 | 453 | 811,21 | 9 | 27,59 |
| 10 | 321 | 458 | 904,81 | 2 | 26,20 |
| 11 | 309 | 446 | 951,61 | 2 | 30,27 |
| 12 | 304 | 441 | 1045,21 | 2 | 30,61 |

**Tabulka C-13 Analýza naměřených dat – pair.blif**

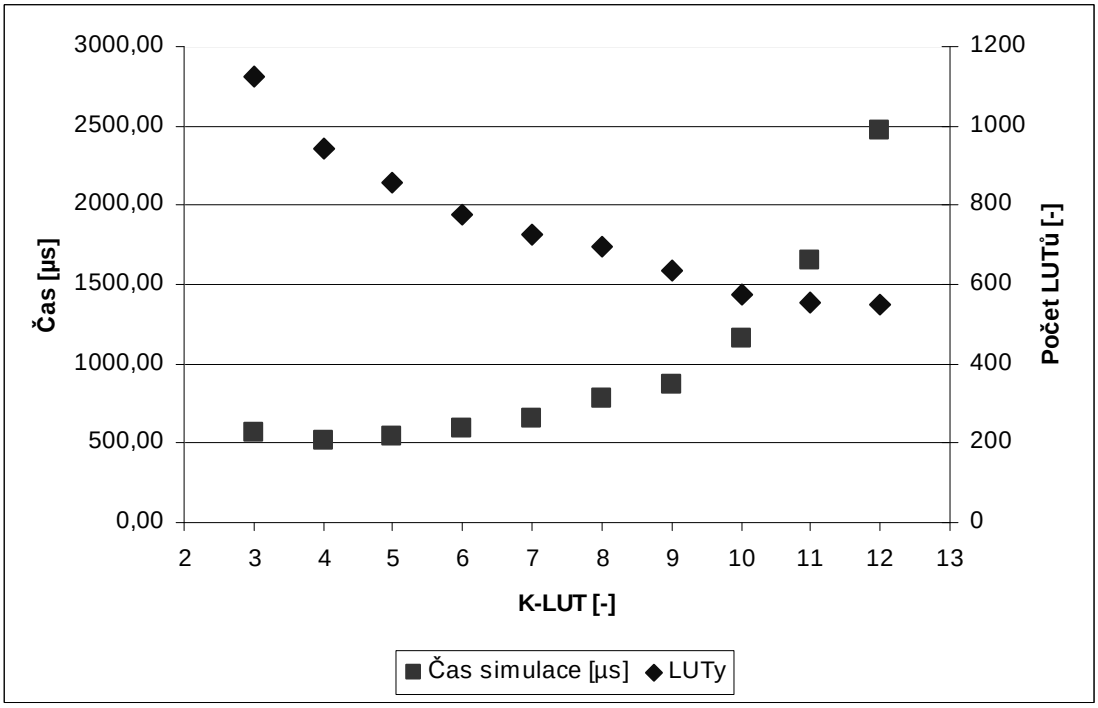



**Graf C-13 Závislost rychlosti simulace na počtu vstupů do LUTu – pair.blif**

```
Soubor: t481.blif
Počet vstupů: 16
Počet výstupů: 1
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 1124 | 1125 | 561,60 | 3 | 83,11 |
| 4 | 944 | 945 | 514,80 | 4 | 72,59 |
| 5 | 858 | 859 | 546,00 | 5 | 63,80 |
| 6 | 774 | 775 | 592,80 | 6 | 55,35 |
| 7 | 725 | 726 | 655,20 | 7 | 49,17 |
| 8 | 697 | 698 | 780,01 | 8 | 50,72 |
| 9 | 632 | 633 | 873,61 | 9 | 49,92 |
| 10 | 576 | 577 | 1154,41 | 10 | 37,61 |
| 11 | 552 | 553 | 1653,61 | 11 | 22,24 |
| 12 | 549 | 550 | 2464,82 | 12 | 22,73 |

**Tabulka C-14 Analýza naměřených dat – t481.blif**

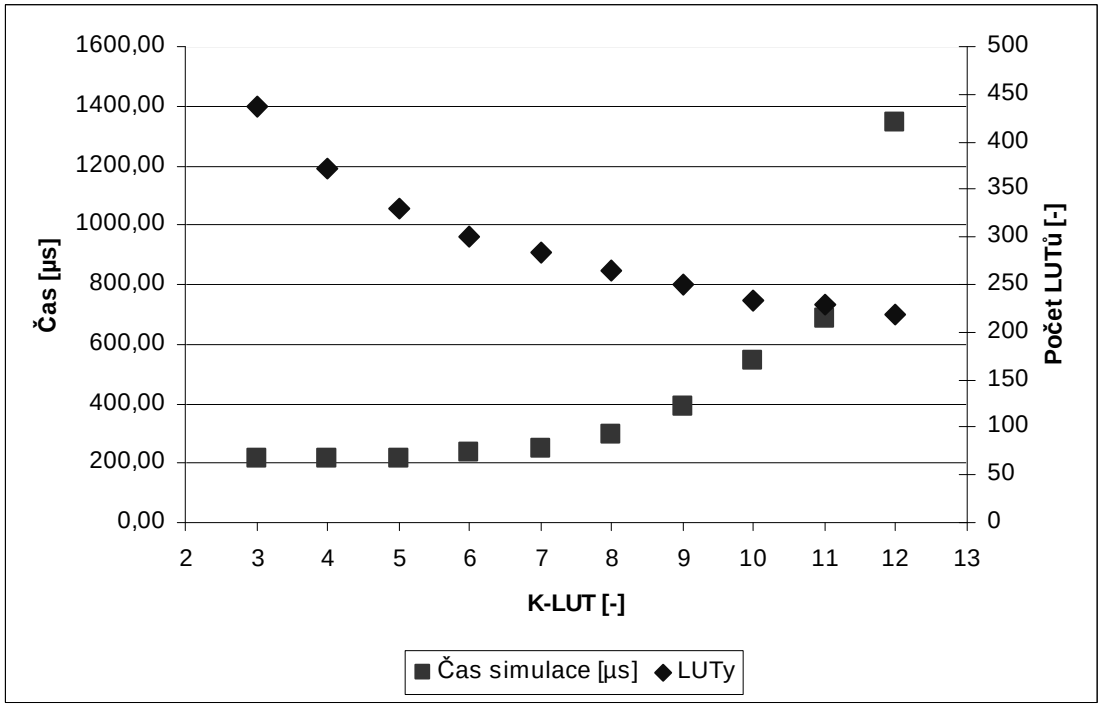



**Graf C-14 Závislost rychlosti simulace na počtu vstupů do LUTu – t481.blif**

```
Soubor: too_large.blif
Počet vstupů: 38
Počet výstupů: 3
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 435 | 438 | 218,40 | 3 | 83,33 |
| 4 | 369 | 372 | 218,40 | 4 | 63,71 |
| 5 | 327 | 330 | 218,40 | 5 | 53,33 |
| 6 | 298 | 301 | 234,00 | 6 | 41,86 |
| 7 | 281 | 284 | 249,60 | 7 | 39,79 |
| 8 | 261 | 264 | 296,40 | 8 | 32,20 |
| 9 | 247 | 250 | 390,00 | 9 | 26,40 |
| 10 | 230 | 233 | 546,00 | 10 | 24,89 |
| 11 | 226 | 229 | 686,40 | 2 | 24,02 |
| 12 | 216 | 219 | 1341,61 | 2 | 24,20 |

**Tabulka C-15 Analýza naměřených dat – too_large.blif**

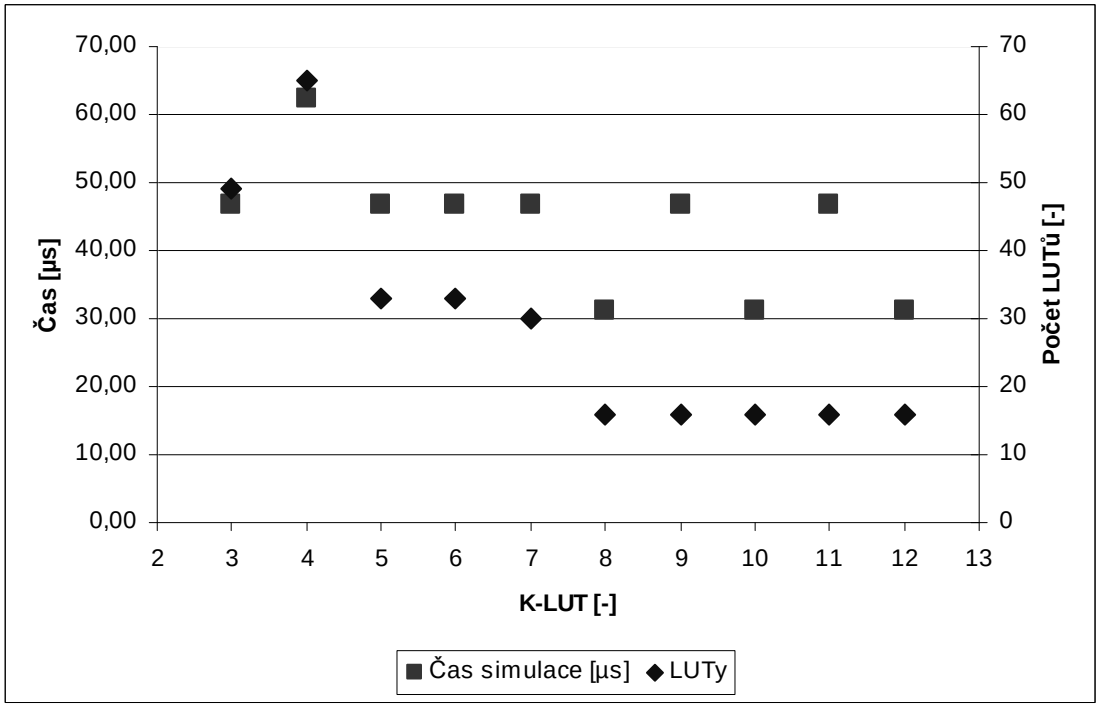



**Graf C-15 Závislost rychlosti simulace na počtu vstupů do LUTu – too_large.blif**

```
Soubor: unreg.blif
Počet vstupů: 36
Počet výstupů: 16
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 33 | 49 | 46,80 | 3 | 65,31 |
| 4 | 49 | 65 | 62,40 | 2 | 75,38 |
| 5 | 17 | 33 | 46,80 | 2 | 51,52 |
| 6 | 17 | 33 | 46,80 | 2 | 51,52 |
| 7 | 14 | 30 | 46,80 | 6 | 53,33 |
| 8 | 0 | 16 | 31,20 | 6 | 100,00 |
| 9 | 0 | 16 | 46,80 | 6 | 100,00 |
| 10 | 0 | 16 | 31,20 | 6 | 100,00 |
| 11 | 0 | 16 | 46,80 | 6 | 100,00 |
| 12 | 0 | 16 | 31,20 | 6 | 100,00 |

**Tabulka C-16 Analýza naměřených dat – unreg.blif**



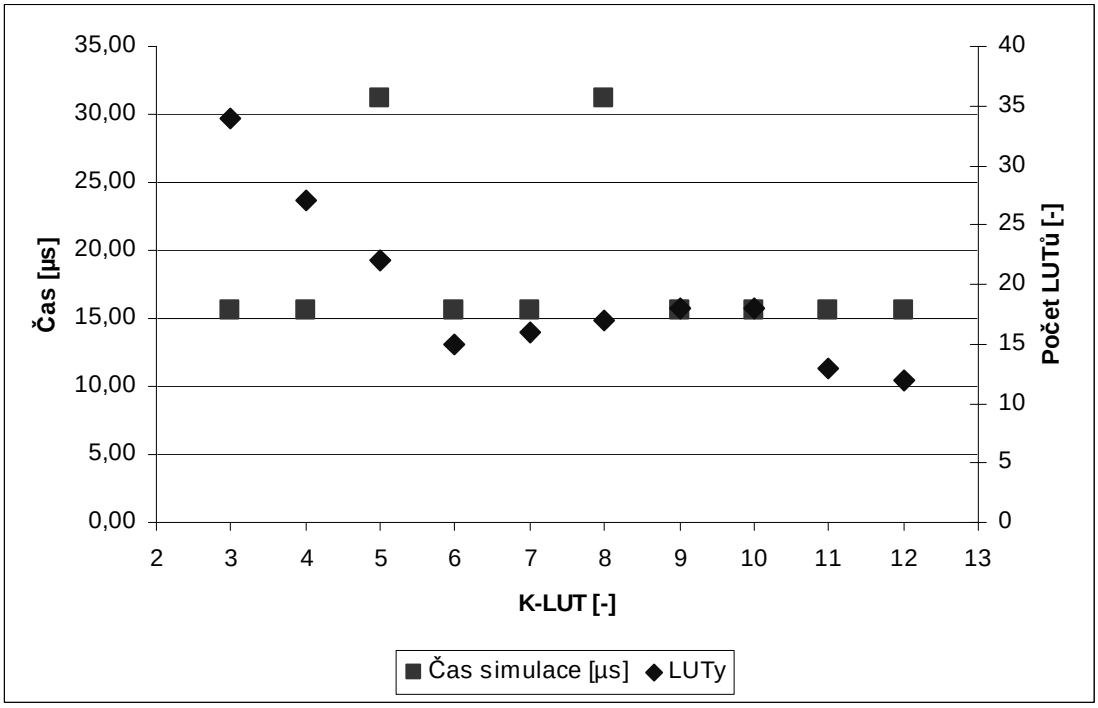

**Graf C-16 Závislost rychlosti simulace na počtu vstupů do LUTu – unreg.blif**

```
Soubor: x2.blif
```

```
Počet vstupů: 10
Počet výstupů: 7
```

| K-LUT | Počet vnitřních proměnných | Počet LUTů | Čas simulace [µs] | Nejpočetnější namapování | Procentuální podíl [%] |
|---|---|---|---|---|---|
| 3 | 27 | 34 | 15,60 | 3 | 64,71 |
| 4 | 20 | 27 | 15,60 | 2 | 40,74 |
| 5 | 15 | 22 | 31,20 | 2 | 45,45 |
| 6 | 8 | 15 | 15,60 | 3 | 26,67 |
| 7 | 9 | 16 | 15,60 | 4 | 31,25 |
| 8 | 10 | 17 | 31,20 | 2 | 29,41 |
| 9 | 11 | 18 | 15,60 | 2 | 38,89 |
| 10 | 11 | 18 | 15,60 | 2 | 50,00 |
| 11 | 6 | 13 | 15,60 | 2 | 30,77 |
| 12 | 5 | 12 | 15,60 | 2 | 33,33 |

**Tabulka C-17 Analýza naměřených dat – x2.blif**




**Graf C-17 Závislost rychlosti simulace na počtu vstupů do LUTu – x2.blif**

# D. Základní definice formátů

## a. Formát PLA

Tento formát umožňuje zápis dvouúrovňového kombinačního obvodu, respektive jeho reprezentaci, pomocí logického výrazu zadaného disjunktní formou součtu součinů. Například $(x_0 \times \bar{x}_1 \times \bar{x}_2 \times \bar{x}_3) + (x_0 \times \bar{x}_1 \times x_2 \times \bar{x}_3) + (x_0 \times \bar{x}_1 \times x_3)$.

### Neúplně definované funkce

K pochopení PLA je nutná znalost definování funkcí pomocí množin ON-set, OFF-set, DC-set, které společně definují úplnou funkci.

$$F = (ON-set, OFF-set, DC-set) : B^n \rightarrow \{1,0,x\}$$

- `ON-set` – určuje množinu termů, která generuje na výstup logickou jedničku. $f(x) = 1 \leftrightarrow F(x) = 1$
- `OFF-set` – Množina, jenž určuje na výstup logickou nulu. $f(x) = 1 \leftrightarrow F(x) = 0$
- `DC-set` – Množina určující neurčené stavy. $f(x) = 1 \leftrightarrow F(x) = x$

K úplné definici všech funkcí je zapotřebí znát pouze dvě množiny, třetí neznámá množina je doplněk součtu známých množin k maximální možné $B^n$.

### Hlavička PLA

V hlavičce se nachází klíčová slova definující vlastnosti celého PLA. Uvádím jen několik nejdůležitějších, kde `[d]` označuje celočíselnou hodnotu a `[s]` řetězec znaků bez mezer.

- `.i [d]` – Počet vstupních proměnných.
- `.o [d]` – Počet výstupních proměnných.
- `.ilb [s1] . . . [sn]` – Nepovinné. Určuje jména vstupních proměnných, jména jsou oddělena minimálně jednou mezerou a jejich počet musí odpovídat počtu vstupů.
- `.ob [s1]. . . [sn]` – Nepovinné. Tento parametr definuje obdobně jména výstupních proměnných. Opět jsou odděleny minimálně jednou mezerou a jejich počet musí odpovídat hodnotě definované v `.o [d]`.
- `.p [d]` – Nepovinné. Určuje počet termů v PLA.
- `.type [s]` – Nejdůležitější parametr, definuje způsob reprezentace tabulky termů. Existují tyto typy.

- Typ `f` určuje množinu `ON-set`. OFF-set se získá jako doplněk `ON-setu`. `DC-set` je zde prázdná množina.
- Typ `r` definuje `OFF-set`. `ON-set` se získá jako doplněk. `DC-set` je zde prázdná množina.
- Typ `fd` definuje dvě množiny `ON-set` a `DC-set`. `OFF-set` je doplněk k součtu množin `ON-set` a `DC-set`. Tento typ se používá jako defaultní nastavení při nepřítomnosti parametru `.type`.
- Typ `fr` definuje `ON-setu` a `OFF-set`. `DC-set` je doplněk k součtu množin `ON-set` a `OFF-set.`
- Typ `dr` definuje `DC-set` a `OFF-set`. `ON-set` je doplněk k součtu množin `OFF-set` a `DC-set`
- Typ `fdr` je úplný typ, který definuje jak `ON-set`, `OFF-set,` tak i `DC-set`.

## Tělo PLA

Jádrem PLA je jeho tělo, což je tabulka definující vazbu mezi vstupy a výstupy, která se skládá z jednotlivých řádků. Každý řádek definuje jeden součinový term vstupních signálů a k němu náležící výstupní vektor definující výstup, oddělený mezerou. Počet znaků vstupní a výstupní části musí být shodný s definovanou hodnotou počtu vstupních a výstupních signálů z hlavičky PLA.

Například: **`010-01 1~00`**

Tento řádek definuje term $\bar{x}_0 \times x_1 \times \bar{x}_2 \times \bar{x}_4 \times x_5$ a jeho výstupní vektor $\langle 1,\sim,0,0 \rangle$. Znak ~ znamená, že je nedůležitý a bude později doplněn jiných termem nebo se jedná o zástupce nedefinované množiny a bude po ukončení průchodu celého PLA nahrazena správným znakem.

### Patička PLA

Celé PLA může být ukončeno znaky `.e` nebo `.end`.

```
.i 9
.o 2
.ibl a0 a1 a2 a3
.ob x0 x1
.type fd
.p 4
-100 11
1011 1~
0111 11
0101 10
.end
```

**Obr. D.1 Ukázka souboru PLA**

## b. Formát BLIF (Berkeley Logic Interchange Format)

Tento formát vychází z předcházejícího formátu PLA. Oproti němu umožňuje zápis víceúrovňového kombinačního i sekvenčního obvodu. BLIF se skládá z více (může být i jen jedna) tabulek, tzv. K-LUTů (Look Up Table). Každý K-LUT je jednovýstupové PLA s K vstupy, které může být typu f nebo r. Vazby mezi jednotlivými LUTy jsou jasně definovány pomocí názvů signálů. A to jak vstupních a výstupních, tak i vnitřních, které mohou být definovány názvem výstupu z jednotlivého LUTu nebo definováním konstanty.

### Hlavička BLIF

Zde si řekneme pár základních klíčových slov, které jsou důležité pro zápis kombinačního obvodu. Znak `[d]` označuje celočíselnou hodnotu a `[s]` řetězec znaků bez mezer.

- .model `[s]` – Řetězec určující název modelu, souboru BLIF.
- .inputs `[s1] . . . [sn]` – Řetězce určují názvy vstupních signálů a tím i definují jejich počet.
- .outputs `[s1] . . . [sn]` – Řetězce definují názvy výstupních formátů a tím i definují jejich počet.

## Tělo BLIF

Tělo se skládá z většího počtu LUTů (tabulek) majících vlastní hlavičku.

- Hlavička LUTu obsahuje `.names [s1] . . . [sn] [sx]`, která určuje vstupní signály a jeden výstupní signál. Výstupní signál může nabývat pouze dvou hodnot, jedničky (typ f) a nuly (typ r), a je v celé jedné tabulce stejná.

- Tělo LUTu se skládá z řádků definujících součinové termy obdobně jako u PLA. Řádek **`1101-1 1`** určuje součinový term $x_0 \times x_1 \times \overline{x}_2 \times x_3 \times x_5$.

Specifický zápis LUTu se používá při specifikaci vnitřní konstanty. Hlavička obsahuje pouze jeden název a vnitřek je definován hodnotou konstanty. Například takto:

```
.names konstanta
1
```

## Patička BLIF

Soubor BLIF je ukončen klíčovým slovem `.end` .

```
.model traffic_c1
.inputs a b c d e
.outputs f
.name h f
0 1
.names d a b c e h
000-- 1
00-0- 1
0-00- 1
00--0 1
0-0-0 1
0--00 1
.end
```

**Obr. D.2 Ukázka souboru BLIF**

# E. Obsah přiloženého CD

| | |
|---|---|
| **index.html** | - Průvodce obsahem CD. |
| Adresář **Benchmarky** | |
| Adresář **MCNC** | - skupina testovacích benchmarků |
| Adresář **Analýza** | - přemapované BLIF použité v Kapitole 6 |
| Adresář **Execute** | - spustitelné soubory |
| Adresář **Literatura** | - literatura ze které jsem čerpal |
| Adresář **Source** | - obsahuje zdrojové kódy jádra BOOM a aplikace BlifSim a KSim (PLA) |
| Adresář **Apps** | - zdrojové kódy aplikací |
| Adresář **Kernel** | - zdrojové kódy jádra BOOM |
| Adresář **Multilevel** | - zdrojové kódy jádra BOOM |
| Adresář **Releases\MSVC** | - zdrojové kódy ve formě projektu pro Microsoft Visual Studio 2008 |
| Adresář **Text** | |
| **Bc.doc** | - text bakalářské práce ve formátu doc |
| **Bc.pdf** | - text bakalářské práce ve formátu pdf |