ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA TECHNOLOGIÍ A MĚŘENÍ

DIPLOMOVÁ PRÁCE

Návrh software pro řešení elektrických obvodů - parser

Bc. Pavel Kupilík **2013**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel KUPILÍK**
Osobní číslo:	**E09N0030P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Komerční elektrotechnika**
Název tématu:	**Návrh software pro řešení elektrických obvodů - parser**
Zadávající katedra:	**Katedra technologií a měření**

Zásady pro vypracování:

1. Prostudujte teorii formálních jazyků a seznamte se základy konstrukce překladačů.
2. Prostudujte podrobně syntaxi jazyka PSpice.
3. V jazyce C++ vytvořte modul "Parser", který bude zpracovávat vstupní textový soubor ve formátu PSpice, výstupem budou vhodně navržené třídy připravené pro solver.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce: **Ing. David Pánek, Ph.D.**
Katedra teoretické elektrotechniky

Datum zadání diplomové práce: **15. října 2012**
Termín odevzdání diplomové práce: **9. května 2013**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Ing. Vlastimil Skočil, CSc.
vedoucí katedry

V Plzni dne 15. října 2012

Abstrakt

Předkládaná diplomová práce je zaměřena na základní část software pro řešení elektrických obvodů – paseru.

Klíčová slova

Parser, překladač, scanner, třída

Abstract

The master theses presents the principles of the basic part software for solving electrical circuit – parser.

Key words

Parser, compilator, scanner, class

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 9.5.2012 Bc. Pavel Kupilík

……..

Seznam symbolů a zkratek

ε……. prázdný řetězec

α_i ……řetězce

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu práce Ing. Davidu Pánkovi Ph.D za trpělivost, při zpracování této práce

OBSAH

ÚVOD **11**

1 TEORIE FORMÁLNÍCH JAZYKŮ **12**

1.1 ZÁKLADNÍ POJMY 12
1.2 GRAMATIKA 12
1.3 ZÁPIS SYNTAXE 13
1.4 KONSTRUKCE PŘEKLADAČE 14
1.4.1 Popis přední části kompilátoru *16*
1.4.2 Lexikální analyzátor - scanner *16*

2 POPIS JAZYKA PSPICE **18**

2.1 VSTUPNÍ SOUBOR .CIR 18
2.2 TOPOLOGIE OBVODU 18
2.3 PŘÍKAZY 19
2.3.1 Příkaz .AC - AC analýza *19*
2.3.2 Příkaz .DC – Stejnosměrná analýza *19*
2.3.3 Příkaz .OP - pracovní bod *19*
2.3.4 Příkaz .FOUR - Fourierova analýza *19*
2.3.5 Příkaz .SENS - citlivostní analýza *20*
2.3.6 Příkaz .TF – přenosová funkce *20*
2.3.7 Příkaz .TRAN – přechodová analýza *20*
2.3.8 Příkaz .IC – nastavení počáteční hodnoty *20*
2.3.9 Příkaz .NODESET – nastavení hodnot v uzlech *20*
2.3.10 Příkaz .SUBCKT a .ENDS - podobvody *21*
2.3.11 Příkaz .MODEL – definice modelu *21*
2.3.12 Příkaz .FUNC – funkce *21*
2.3.13 Příkaz .INC – přidání souboru *21*
2.3.14 Příkaz .LIB – přidání knihovního souboru *21*
2.3.15 Komentáře *21*
2.4 ČÍSELNÉ KONSTANTY 22
2.5 FUNKCE 22
2.6 DEFINICE SOUČÁSTEK 24
2.6.1 Kapacitor C *24*
2.6.2 Rezistor R *25*
2.6.3 Indukčnost L *25*
2.6.4 Transformátor *26*
2.6.5 Dioda D *26*
2.6.6 Nezávislý proudový zdroj I *26*
2.6.7 Nezávislý zdroj napětí V *27*
2.6.8 Napěťový zdroj řízený napěťovým zdrojem E a Proudový zdroj řízený napěťovým zdrojem G *27*
2.6.9 Bipolární tranzistor *28*
2.6.10 Unipolární tranzistory *28*
2.6.10.1 JFET 28
2.6.10.2 MOSFET 28
2.6.11 Přenosová linka T *29*
2.6.12 Uživatelsky definovaná součástka X *29*
2.6.13 Napěťově řízený spínač S *29*

3 KONSTRUKCE PARSERU **30**

3.1 TŘÍDA MODEL 30
3.2 METODA CONNECTMODELTODEVICE() 30
3.3 METODA DEVICES_CNT() 31
3.4 METODA DESTROY() 31
3.5 METODA PRINT_SOUC() 31
3.6 TŘÍDA DEVICE 31
3.7 POSTUP PARSOVÁNÍ VSTUPNÍHO SOUBORU 31

3.8 IMTREPRETACE SOUČÁSTEK....32
ZÁVĚR....33
SEZNAM LITERATURY A INFORMAČNÍCH ZDROJŮ....1

Úvod

Předkládaná práce je zaměřena návrh software pro řešení elektrických obvodů konkrétně na část, která se zabývá dekódováním zdrojového kódu jazyka PSpice a převodem do formátu použitelného v modulu řešiče.

Text práce je rozdělen do tří částí. První se zabývá teorií formálního jazyka, konstrukcí a popisem překladače. Ve druhé části je popsána syntaxe jazky PSpice. Třetí část popisuje samotný návrh modulu parseru pro analýzu vstupních dat v jazyce PSpice, které budou dále využívána pro modul solver neboli řešič. Pro naprogramování modulu parseru byl zvolen programovací jazyk C++.

1 Teorie formálních jazyků

Teorie formálních jazyků a gramatik je významnou součástí informatiky, která je využita především při tvorbě překladačů a kompilátorů.

1.1 Základní pojmy

Stejně jako u přirozených jazyků, se v teorii formálních jazyků používají pojmy jako písmeno, slovo, věta.

Abeceda A je konečná množina symbolů, ze kterých se v daném jazyku vytvářejí vyšší významové celky.

Např. písmena ……[a-z][A-Z]

číslice [0-9]

speciální znaky[+ - * =...]

vyhrazená slova [if, while, case, int, double…]

Řetězec w je libovolná konečná posloupnost symbolů dané abecedy A.

Délka řetězce |w| je definovaná jako počet symbolů v řetězci w.

Teoreticky lze v jazyku odvodit řetězce různé délky od prázdných až po nekonečné. Významově jsou důležité řetězce konečné délky.

Prázdný řetězec ε je řetězec s nulovou délkou.

Zřetězení - jedná se o operace. Tato operace je definována na množině řetězců dané abecedy následovně:

$u = a_1 a_2 ... a_k$, $v = b_1 b_2 ... b_k$, u a v jsou řetězce nad danou abecedou, pak

$u \bullet v = a_1 a_2 ... a_k b_1 b_2 ... b_k$ je nový řetězec uv . Zápisy u,v a uv jsou ekvivalentní.

Množina všech řetězců nad abecedou A je označena A*. Nechť A je abeceda. Potom formální jazyk nad abecedou A definujeme jako libovolnou podmnožinu množiny A*.

1.2 Gramatika

Výše uvedené není vhodné pro popis programovacího jazyka, vyžaduje totiž vyjmenování všech možných řetězců. Mechanismus, který popisuje daný jazyk je gramatika. V gramatice se používají následující pojmy:

Terminál nebo terminální symbol – prvotní symbol daného jazyka.

Neterminál nebo neterminální symbol – symbol, který je definovaný pomocí pravidel jazyka.

Začátečný symbol (startovací) je jeden z neterminálních symbolů, ze kterého začíná generování vět jazyka, případně analýza věty.

Přepisovací pravidlo nebo *vytvářecí pravidlo*, produkce je předpis, který určuje způsob náhrady neterminálu jiným podřetězcem.
Gramatika je určena syntaxí jazyka, tj. přípustnou strukturou vět. Sémantika se zabývá správným významem věty.
Gramatika je definovaná jako čtveřice $G = (N, T, P, S)$ **,** kde

N – konečná množina neterminálních symbolů

T – konečná množina terminálních symbolů, $N \cap T = \varnothing$

S – začáteční symbol, $S \in N$, (musí být pouze jeden!)

P – konečná množina přepisovacích pravidel z $(N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$

Pro odlišení terminálů a neterminálů se používají lomené závorky např. <identifikátor>. V případě, že je identifikátor uveden v lomených závorkách jedná se o neterminální symbol, o terminál který zastupuje množinu pojmů splňujících určité vlastnosti. To, o který pojem se jedná, určují pravidla z množiny P.

1.3 Zápis Syntaxe

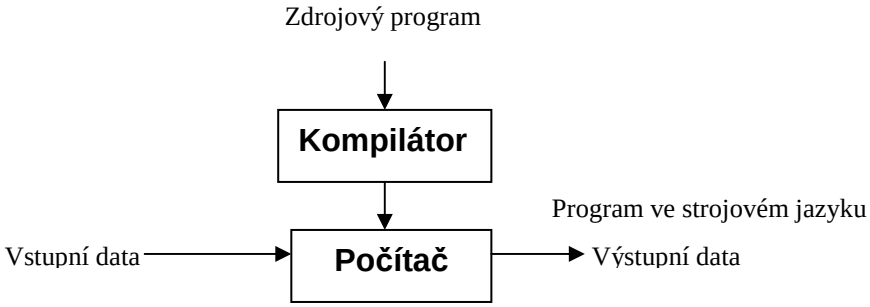
Pro zápis syntaxe pravidel se používá Backusova – Naurova forma. Tato forma je charakteristická následujícím tvarem :
<neterminální symbol> ::=|α_1| α_2|…| α_n
Neterminální symbol na levé straně je definován řetězcem na pravé straně, kde α_i jsou řetězce terminálních a neterminálních symbolů nebo prázdný řetězec. Tento systém popisuje pravidla gramatiky. Jako příklad je uveden zápis celého čísla
*<celé číslo> ::= (<znaménko> | ε) . <číslice>.<číslice>**
Tento systém zápisu je sám o sobě jazyk, který popisuje gramatiku jiného jazyka, je proto nazýván metajazyk. Symboly se nazývají metasymboly a příkazy se nazývají metapříkazy.

Metasymbol	Význam metasymbolu
::=	*definice* symbol nahrazení levé strany (LHS) pravou stranou pravidla (RHS) při derivaci (= „definuj jako…“)
ı	*alternativa* metaoperátor výběru jedné varianty z uvedených pravých stran (= „nebo“)
.	*sekvence* metaoperátor zřetězení operandů v zápise pravidla (= „následuje“)
*	*mocnina* metaoperátor opakování výskytů metaoperandu, (..) prázdného počtu výskytů (= „opakuj“)
()	změna priority metaoperátorů
<>	Čitelné vyjádření symbolu, který nazýváme neterminál
+	Mocnina typu „alespoň jedenkrát“ Metaoperátor opakování metaoperandů s povinným výskytem (= „ opakuj alespoň jednou“)

Tab. 1.1 přehled metasymbolů a metaznaků

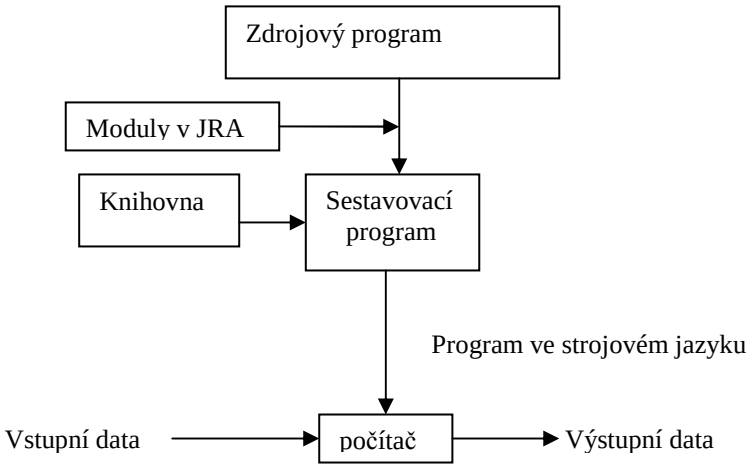
1.4 Konstrukce překladače

Překladač neboli kompilátor slouží k překladu zdrojového kódu vytvořeného programátorem do formy kterou využívá procesor. Překladač převádí kód z vyššího programovacího jazyka do strojového kódu procesoru. Na následujícím schématu je blokově zobrazen překlad zdrojového programu.



Obr. 1.1 Zjednodušené schéma překladu programu

Z pravidla kompilátory nepřekládají zdrojový program do strojového jazyka, program překládají do jazyka relativních adres. Tento jazyk je specifický pro každý procesor a vychází z jeho architektury.Program vznikne sestavením modulů jazyka relativních adres a knihovních modulů pomocí sestavovacího programu (Obr.:1.2).

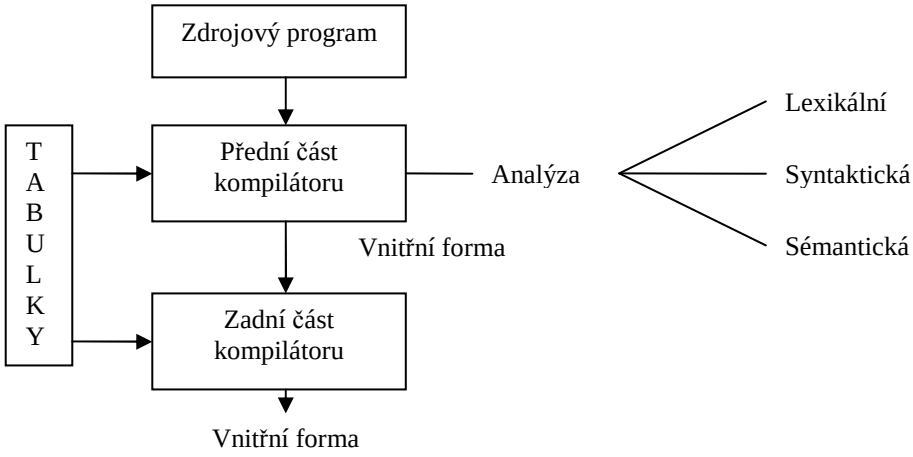


Obr.1.2

Překlad programovacího jazyka do strojově orientovaného jazyka je složitý problém, z toho důvodů se pro zjednodušení se kompilace rozděluje do dvou částí analýzy a syntézy.

Analýza se provádí v přední části kompilátoru a lze ji rozdělit do tří částí, na analýzu lexikální, syntaktickou a sémantickou.

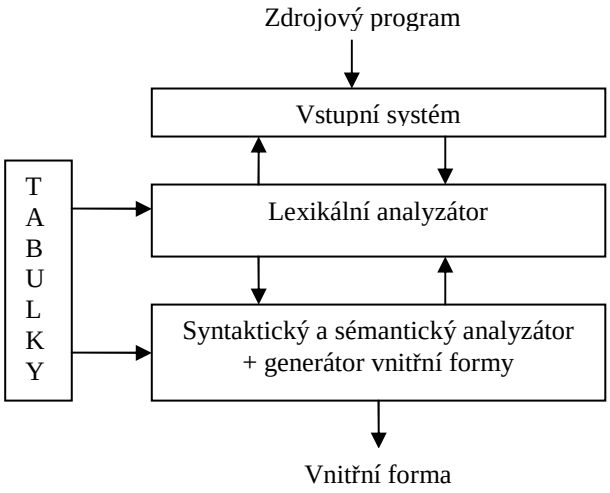
Výsledek analýzy je převeden na vnitřní formu programu a je předán zadní části kompilátoru, kde se generuje cílový program. V této části kompilátoru lze provádět i optimalizace. Do obou částí kompilátorů mohou vstupovat různé tabulky ve kterých jsou např. deklarace identifikátorů (Obr.: 1.3).



Obr. 1.3

1.4.1 Popis přední části kompilátoru

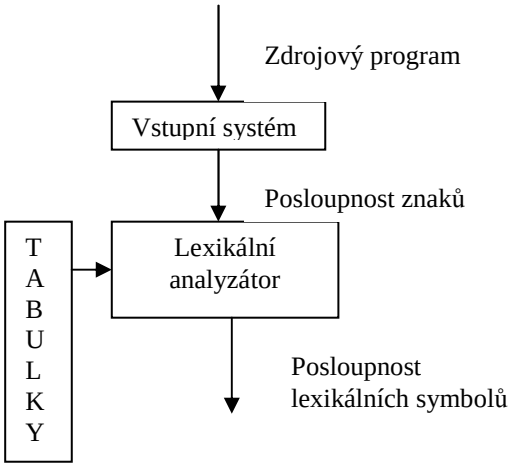
Lexikální analyzátor získává vstupní znaky od vstupního systému a sestavuje z nich lexikální symboly (tokens). Tokens představují identifikátory, klíčová slova, literály, operátory apod. Tyto symboly jsou terminální symboly pro syntaktický analyzátor. Lexikální analyzátor využívá konstrukci regulárních výrazů a konečných automatů. Přední část kompilátoru je na následujícím obrázku (obr. : 1.4.).



Obr. 1.4 Schéma přední části kompilátoru

1.4.2 Lexikální analyzátor - scanner

Lexikální analyzátor čte zdrojový program za pomocí vstupního systému od kterého získává posloupnost znaků, kterou transformuje na posloupnost lexikálních symbolů. Tyto symboly jsou terminálními symboly pro syntaktickou analýzu.



Obr. 1.5 Schéma lexikálního analyzátoru

Lexikální symboly odpovídají množině řetězců, které lze popsat regulárními výrazy a nazývají se lexikální elementy. Kromě syntaxe lexikálních elementů je zavedena též kategorie oddělovačů lexikálních elementů. Mezi tyto elementy řadíme např. mezery, komentáře a oddělovače. Oddělovače zpravidla nebývají pro syntaktickou analýzu důležité, proto je lexikální analyzátor vynecháván. Lexikální analyzátor je realizován jako podprogram, který je volán syntaktickým analyzátorem. Při každém zavolání čte vstupní text a rozpoznává v něm lexikální elementy. Výstupem této operace je lexikální symbol, který odpovídá lexikálnímu elementu.

Příklad syntaxe lexikálních elementů:
Lexikální element = identifikátor |číslo| specifický symbol
Specifický symbol = +|-|<|*| klíčové slovo
Klíčové slovo = if | while | else
Identifikátor = písmeno {písmeno | číslice}
Číslo = číslice {číslice}
Písmeno = A | B |… | Z | a | b | … | z
Číslice = 0 | 1 | … | 9

Dalším úkolem scanneru je:

- Přeskakovat konce řádků a prázdné znaky
- Rozpoznávat rezervované výrazy jako je např. if, else
- Sekvence písmen a číslic které začínají písmenem a zároveň nejsou rezervovaným výrazem jsou rozpoznány jako identifikátor
- Sekvence číslic je rozpoznána jako číslo
- Kombinace speciálních znaků např. = < > jsou rozpoznány jako symboly
- Komentáře programátora jsou přeskočeny
- Pokud scanner přečte neplatný znak vrací hodnotu null

2 Popis jazyka PSpice

2.1 Vstupní soubor .cir

Vstupní soubor .cir slouží k popisu obvodu a začíná uvozovací řádkou, která slouží pro informaci, je možné ji i vynechat. Následuje netlist, což je popis součástek, zapojení v obvodu a popis vstupních parametrů, popřípadě odkaz do knihovny součástek nebo na model součástek, který následuje za netlistm. Pro popis k programu je vhodné používat komentáře. Uvozovací znak komentáře pokud začíná na novém řádku je *(hvězdička). V případě, že komentář nezačíná na začátku řádky ale například za příkazem je použit středník. Za netlustém pak dále pokračují příkazy simulátoru které jsou uvozeny tečkou.Konec definice obvodu je ukončen příkazem .END. V jednom souboru *.cir je možno použít více definic obvodů ale je nutné každou definici ukončit příkazem .END.

2.2 Topologie obvodu

Obvod je složen ze součástek a uzlů, uzle jsou propojeny součástkami. Obvod musí splňovat určité podmínky:

-v obvodu musí být referenční uzel 0.

- všechny uzly v obvodu musí být pojmenovány jedinečným jménem
- do uzlu musí být zapojeny, alespoň dvě součástky.
- v obvodu musí být všechny součástky a vývody zapojeny. Nezapojené vývody lze ošetřit spojením uzlu 0 přes rezistor o vysoké hodnotě.
- od každého uzlu musí existovat cesta pro stejnosměrný proud k uzlu 0.
- zapojení dvou nebo více paralelně napěťových zdrojů je nepřípustné.
- zapojení dvou nebo více sériově proudových zdrojů je nepřípustné.
- zapojení induktoru k paralelně k napěťovému zdroji je možné pouze s využitím seriového rezistoru.
- součástky, které mají nadefinované stejná jména uzlů jsou těmito uzly spojeny

Uzel může mít libovolné jméno složené z písmen a číslic délky maximálně 131 znaků.Některá jména uzlů jsou rezervována např. referenční uzel 0.

- jméno součástky musí být uvozeno typem (např. R, L, C) . Za typem součástky bez mezery pokračuje volitelný název složený s čísel, písmen a znaků $ _ * / % maximální délky 131 znaků.

2.3 Příkazy

2.3.1 Příkaz .AC - AC analýza

Příkaz .AC analyzuje chování obvodu v daném frekvenčním pásu.

Obecná forma: *.AC <typ> <počet bodů> <počáteční frekvence>*

+ <konečná frekvence>

Střídavá analýza počítá chování obvodu v určitém frekvenčním intervalu, okolo pracovního bodu, kde je obvod linearizován. Tuto linearizaci určuje typ analýzy.

Typy jsou: lineární(LIN), oktávová(OCT) a linearizace po dekádách(DEC)

<počet bodů> je celé číslo a určuje počet bodů v intervalu dle typu linearizace.

2.3.2 Příkaz .DC – Stejnosměrná analýza

Obecná forma:*.DC [LIN] <proměnná > <počáteční hodnota> <konečná hodnota>*

+<inkrement> [specifikace vnější smyčky]

Obecná forma:*.DC < typ> <proměnná> <počáteční hodnota> <konečná hodnota>*

+<počet bodů> [specifikace vnější smyčky]

Obecná forma:*.DC <proměnná> LIST <hodnota>* [specifikace vnější smyčky]*

Typ analýzy jako v předchozím případě, určuje počet bodů v oktávě nebo dekádě.

2.3.3 Příkaz .OP - pracovní bod

Obecná forma: .OP

Příkaz .OP podá detailní informace o pracovním bodu obvodu.

2.3.4 Příkaz .FOUR - Fourierova analýza

Obecná forma:*.FOUR <frekvence 1. harmonické> [počet harmonických]*

+ <výstupní proměnná>

Příkaz .FOUR počítá amplitudu a fázi jednotlivých vyšších harmonických. Pokud není udán počet harmonických, počítá se počtem 9. Výstup je uveden v souboru *.out .

2.3.5 Příkaz .SENS - citlivostní analýza

Obecná forma:.*SENS <výstupní proměnná>**

Výstup analýzy je proveden do souboru *.out .

2.3.6 Příkaz .TF – přenosová funkce

Obecná forma: .*TF < výstupní proměnná > <vstupní proměnná>*

Linearizuje obvod okolo pracovního bodu a výstup uvádí do souboru *.out.

2.3.7 Příkaz .TRAN – přechodová analýza

Obecná forma: .*TRAN[/OP] <časový krok pro výpis> <konečný čas>*

+[počátek výpisu[mez časového kroku]][SKIPBP]

Při přechodové analýze se je využita proměnná TIME, která se mění od nuly do hodnoty konečného času. Hodnota časového kroku je využita při výpisu do výstupního souboru. Hodnota počátek výpisu udává od kdy se začnou výstupní hodnoty zapisovat do výstupního souboru.

2.3.8 Příkaz .IC – nastavení počáteční hodnoty

Obecná forma: *.IC < V(<uzel> [,<uzel>])=<hodnota> >**

*.IC <I(<induktor>)=<hodnota>>**

Příkaz .IC slouží k nastavení počátečních hodnot, jako napětí na kapacitoru nebo proud induktorem.

2.3.9 Příkaz .NODESET – nastavení hodnot v uzlech

Obecná forma: *.NODESET < V(<uzel> [,<uzel>])=<hodnota> >**

*.NODESET <I(<induktor>)=<hodnota>>**

Stejně jako příkaz .IC nastavuje hodnoty napětí v uzlech a proudů součástkami. Příkaz .IC je nadřazen příkazu .NODESET, v případě použití příkazu .IC je .NODESET ignorován.

2.3.10 Příkaz .SUBCKT a .ENDS - podobvody

Obecná forma: *.SUBCKT <název> [uzly]**

...

.ENDS

Příkazy. SUBCKT a .ENDS se používají pro definování podobvodů příkazem .SUBCKT je podobvod uvozen a příkazem .ENDS definice podobvodů končí.

2.3.11 Příkaz .MODEL – definice modelu

Obecná forma: *.MODEL <název modelu> [AKO: <jméno referenčního modelu>]*

+ <typ modelu>

*+ ([<název parametru> = <hodnota> [specifikace tolerancí]]**

+ [T_MEASURED=<hodnota>] [[T_ABS=<hodnota>]

+ [T_REL_GLOBAL=<hodnota>] [T_REL_LOCAL=<hodnota>]])

Jméno referenčního modelu udává od kterého modelu je tento model odvozen.

2.3.12 Příkaz .FUNC – funkce

Obecná forma: *.FUNC <název> ([arg]*) {<tělo funkce>}*

Příkaz .FUNC se používá pro definování vlastních funkcí.

2.3.13 Příkaz .INC – přidání souboru

Obecná forma: *.INC <název souboru>*

2.3.14 Příkaz .LIB – přidání knihovního souboru

Obecná forma: *.LIB [název souboru]*

2.3.15 Komentáře

Pro okomentování celého řádku se používá hvězdička (*). Komentáře jsou ignorováni. Pro komentář na konci řádky se používá středník (;), pro pokračování příkazu na další řádce je použito znaménko plus (+).

2.4 Číselné konstanty

Číselné konstanty mohou být vyjádřeny jako celá čísla, čísla s plovoucí desetinou tečkou nebo jako čísla v exponenciálním tvaru. Dále je možno využít standardních symbolů:

Hodnota	Symbol	Název
10^{-15}	F	femto
10^{-12}	P	piko
10^{-9}	N	nano
10^{-6}	U	mikro
$25.4*10^{-6}$	MIL	--
10^{-3}	M	mili
	C	hodinový cyklus*
10^{+3}	K	kilo
10^{+6}	MEG	mega
10^{+9}	G	giga
10^{+12}	T	tera

Tab.: 2.1. Tabulka standardních symbolů pro číselné konstanty

U číselných konstant se nerozlišují velká a malá písmena (U i u znamenají mikro). Další znaky za symbolem jsou ignorovány až do oddělovače. Bezprostředně za konstantou je vhodné pro lepší orientaci uvádět fyzikální jednotky (1e3, 1.0e3, 1KHz reprezentují stejné číselné hodnoty).

2.5 Funkce

Pro numerické výrazy lze použít následující funkce:

Funkce	**Význam**	**Popis**
ABS(x)	|x|	Absolutní hodnota
ACOS(x)	arcuscosinus x	-1.0 ≤ x ≤ +1.0
ARCTAN(x)	tan-1(x)	výsledek v radiánech
ASIN (x)	arcussinus x	-1.0 <= x <= +1.0
ATAN(x)	tan-1(x)	výsledek v radiánech

ATAN2(y,x)	arctan(y/x)	Výsledek v radiánech
COS(x)	cos(x)	x v radiánech
COSH(x)	hyperbolický cosinus x	x v radiánech
EXP(x)	e^x	
IF(t, x, y)	x když t=TRUE y když t=FALSE	
IMG(x)	Imaginární část x	Hodnota 0.0 pro reálné číslo
LIMIT(x,min,max)		x=min - pokud x<min x=max – pokud x>max x=x v ostatních případech
LOG(x)	Ln(x)	Logaritmus se základem e
LOG10(x)	log(x)	Logaritmus se základem10
MAX(x,y)		Vrací větší z x nebo y
MIN(x,y)		Vrací menší z x nebo y
P(x)	Fáze x	Vrací 0.0 pro reálné číslo
PWR(x,y)	Mocnina	$\|x\|^y$
R(x)		Reálná část x
SIN(x)		x v radiánech
SINH(x)	Hyperbolický sin	x v radiánech
STP(x)	přepínač	x=1 pro x>0.0, x=0 pro x<0.0
SQRT(x)	$\sqrt{x}$	
TAN(x)	tan(x)	x v radiánech
TANH(x)	Hyperbolický tan	x v radiánech
TABLE(x,x1,y1, xn,yn)		definuje tabulku
funkce	význam	popis
ABS(x)	|x|	Absolutní hodnota
ACOS(x)	arcuscosinus x	-1.0 ≤ x ≤ +1.0
ARCTAN(x)	tan-1(x)	výsledek v radiánech
ASIN (x)	arcussinus x	-1.0 <= x <= +1.0
ATAN(x)	tan-1(x)	výsledek v radiánech
ATAN2(y,x)	arctan(y/x)	Výsledek v radiánech
COS(x)	cos(x)	x v radiánech
COSH(x)	hyperbolický cosinus x	x v radiánech

EXP(x)	e^x	
IF(t, x, y)	x když t=TRUE y když t=FALSE	
IMG(x)	Imaginární část x	Hodnota 0.0 pro reálné číslo
LIMIT(x,min,max)		x=min - pokud x<min x=max – pokud x>max x=x v ostatních případech
LOG(x)	Ln(x)	Logaritmus se základem e
LOG10(x)	log(x)	Logaritmus se základem10
MAX(x,y)		Vrací větší z x nebo y
MIN(x,y)		Vrací menší z x nebo y
P(x)	Fáze x	Vrací 0.0 pro reálné číslo
PWR(x,y)	Mocnina	$\|x\|^y$
R(x)		Reálná část x
SIN(x)		x v radiánech
SINH(x)	Hyperbolický sin	x v radiánech
STP(x)	přepínač	x = 1 pro x>0.0 x = 0 pro x<0.0
SQRT(x)	$\sqrt{x}$	
TAN(x)	tan(x)	x v radiánech
TANH(x)	Hyperbolický tan	x v radiánech
TABLE(x,x1,y1,xn,yn)		definuje tabulku

Tab. 2.2 Seznam Funkcí

2.6 Definice součástek

2.6.1 Kapacitor C

Obecná forma:*C<název> <(+) uzel> <(-) uzel> [název modelu] <hodnota> +[IC=<počáteční hodnota>]*

Definice kapacitoru musí začínat písmenem C a dále bez mezery je uveden název součástky. Další argumenty určují zapojení součástky v obvodu. První uzel je definován jako kladný. Napětí na součástce je pak rozdíl napětí v jednotlivých uzlech. V případě, že je název modelu vynechán,hodnota kapacity je uvedena ve faradech.

Pokud model definován je hodnota kapacity určují parametry uvedené v modelu. U součástek je rovněž možné nastavit počáteční hodnotu pomocí parametru IC.

Model kapacitoru CAP

Model kapacitou je definován následovně.

Obecná forma: *.MODEL <název modelu> CAP [parametry modelu]*

Model kapacitou je definován jako lineární, teplotně závislý, ale tyto parametry v modelu jsou nastaveny na nulovou hodnotu. V případě, že je definován název modelu výpočet hodnoty kapacity je určen následujícím vzorcem:

$<hodnota> \cdot C \cdot (1+\mathbf{VC1} \cdot V+\mathbf{VC2} \cdot V^2) \cdot (1+\mathbf{TC1} \cdot (T-Tnom)+\mathbf{TC2} \cdot (T-Tnom)^2)$

C - kapacitní násobitel

TC1 - lineární teplotní koeficient

TC2 – kvadratický teplotní koeficient

VC1 – lineární napěťový koeficient

VC2 – kvadratický napěťový koeficient

2.6.2 Rezistor R

Obecná forma:*R<název> <(+)uzel> <(-) uzel> [název modelu] <hodnota>*

+[TC = <TC1> [,<TC2>]]

Rezistor je definován obdobně jako kapacitor, definici uvozuje označení součástky R, následuje zapojení v obvodu, název modelu pokud je nadefinován a hodnota součástky.

Model odpotu RES

Obecná forma: *.MODEL <název modelu> RES [parametry modelu]*

Model odporu je stejně jako model kapacitou nadefinován jako teplotně závislí, lze nastavit parametry TC1 a TC2. Tyto parametry jsou opět nadefinovány na nulovou hodnotu. Hodnota odporu je stanovena následovně

$<hodnota> \cdot R \cdot (1+\mathbf{TC1} \cdot (T-Tnom)+\mathbf{TC2} \cdot (T-Tnom)^2)$

2.6.3 Indukčnost L

Obecná forma: *L<název> <(+) uzel> <(-) uzel> [název modelu] <hodnota>*

+[IC=<počáteční hodnota>]

Model indukčnosti IND

Obecná forma:*.MODEL <název modelu> IND [parametry modelu]*

$$<hodnota>\cdot L\cdot(1+\mathbf{IL1}\cdot I+\mathbf{IL2}\cdot I^2)\cdot(1+\mathbf{TC1}\cdot(T-Tnom)+\mathbf{TC2}\cdot(T-Tnom)^2)$$

2.6.4 Transformátor

Obecná forma**:** *K<název> L<jméno induktoru> <L<jméno induktoru>>**
+<vzájemná indukčnost>
K<název> <L<jméno induktoru>> <vzájemná indukčnost hodnota>*
+ <název modelu> [parametry]

Součástka transformátoru se skládá z několika induktorů. Každý z těchto induktorů musí být nadefinován odděleně a poté je možného použit v definici transformátoru.

Model transformátoru CORE

Obecná forma**:***.MODEL <název modelu> CORE [parametry modelu]*

2.6.5 Dioda D

Obecná forma**:** *D<název> <(+) uzel> <(-) uzel> <název modelu> [další parametry]*

Model diody D

Obecná forma**:***.MODEL <název modelu> D [parametry modelu]*

2.6.6 Nezávislý proudový zdroj I

Obecná forma**:** *I<název> <+ uzel> <- uzel> [[DC] <hodnota>] [AC <hodnota>*
[fázový posun]] +[přenosové funkce]
I<název> <(+) uzel> <(-) uzel>
+ [[DC] <hodnota>]
+ [AC <hodnota> [velikost fáze]]
+ [STIMULUS=<název>]
+ [transient specifikace]

Zdroj proudu I lze definovat jako stejnosměrný, potom není nutné označení DC uvádět, nebo jako střídavý s nastavitelnou velikostí amplitudy. Dále je možné nastavovat přenosovou funkci a volit tak tvar proudu zdroje.

[přenosové funkce]

EXP (<parametry>) exponenciální funkce

EXP (<i1> <i2> <td1> <tc1> <td2> <tc2>)

PULSE (<parametry>) pulsní funkce

PULSE (<i1> <i2> <td> <tr> <tf> <pw> <per>)

PWL (<parametry >) pulsně-šířková funkce

PWL [TIME_SCALE_FACTOR=<hodnota>]

*+ [VALUE_SCALE_FACTOR=<hodnota>] (bod)**

SFFM (<parametry >) frekvenčně modulovaná funkce

SFFM (<ioff> <iampl> <fc> <mod> <fm>)

SIN (<parametry >) sinusová funkce

SIN (<ioff> <iampl> <freq> <td> <df> <fáze>)

Parametry v přenosových funkcích jsou číselného typu.

2.6.7 Nezávislý zdroj napětí V

Obecná forma: *V<název> <+ uzel> <- uzel> [[DC] <hodnota>] [AC <hodnota>*
+[fázový posuv]] [přenosová funkce]

Obdobně jako u proudového zdroje lze volit stejnosměrný nebo střídavý zdroj. Lze nastavit také parametr přenosové funkce.

2.6.8 Napěťový zdroj řízený napěťovým zdrojem E a Proudový zdroj řízený napěťovým zdrojem G

Obecná forma: *E<název> <+ svorka> <- svorka> <+ řídicí svorka> <- řídicí svorka>*
+ <svorka substrát>
E<název> <(+) uzel> <(-) uzel> <(+) řídicí uzel>
+<(-) řídicí uzel><uzel substrát>
E<název> <(+) uzel> <(-) uzel> POLY(<hodnota>)
+ < <(+) řídicí uzel > <(-) řídicí uzel > > < <hodnota> >*
E<název> <(+) <uzel> <(-) uzel> VALUE = { <výraz> }
E<název> <(+) <uzel> <(-) uzel> TABLE { <výraz> } =
+ < <vstupní hodnota>,<výstupní hodnota> >
E<název> <(+) uzel> <(-)uzel> LAPLACE { <výraz> } =
+ { <transformace> }
E<název> <(+) uzel> <(-) uzel> FREQ { <výraz> } = [KEYWORD]

*+ < <frekvence>,<rozsah >,<fáze> >**

+ [DELAY = <prodleva>]

Syntaxe je pro obě součástky napěťový zdroj řízený napěťovým zdrojem (E) proudový zdroj řízený napěťovým zdrojem (G) stejná, liší se pouze označením E nebo G.

2.6.9 Bipolární tranzistor

Obecná forma**:** *Q<název> < uzel kolektor > < uzel báze> < uzel emitor>*

+ [uzel substrát] <název modelu> [další parametry]

Bipolární tranzistor je definován jako součástka typu Q. Součástka je definována jako čtyřvývodová báze, emitoir, kolektor a může být dále definován vývod substrát. Za názvem modelu lze nadefinovat další parametry. Bipolární tranzistor lze nadefinovat jako PNP nebo NPN.

Model bipolárního tranzistoru NPN, PNP

.MODEL <název modelu> NPN [parametry modelu]

.MODEL <název modelu> PNP [parametry modelu]

2.6.10 Unipolární tranzistory

2.6.10.1 JFET

Obecná forma**:** *J<název> <drain uzel> <gate uzel> <source uzel> <název modelu>*

+[další parametry]

Součástka tranzistoru JFET je definována jako čtyřvývodová. Za definicí vývodů je uveden

Model Unipolárního tranzistoru NJF, PJF

.MODEL <název modelu> NJF [parametry modelu]

.MODEL <název modelu> PJF [parametry modelu]

2.6.10.2 MOSFET

Obecná forma**:** *M<název> <uzel drain> < uzel gate> < uzel source>*

+ <bulk/substrát uzel> <název modelu>

+ [L=<hodnota>] [W=<hodnota>]

+ [AD=<hodnota>] [AS=<hodnota>]

+ [PD=<hodnota>] [PS=<hodnota>]

+ [NRD=<hodnota>] [NRS=<hodnota>]

+ [NRG=<hodnota>] [NRB=<hodnota>]

+ [M=<hodnota>] [N=<hodnota>]

Součástka typu MOSFET je definována jako čtyřvývodová. Má definovány vývody gate, drain, source a substrát. Dále je třeba definovat model součástky. Model součástky může být definován jako typ NMOS nebo PMOS.

Model MOSFET NMOS, PMOS

.MODEL <název modelu> NMOS [parametry modelu]

.MODEL <název modelu> PMOS [parametry modelu]

2.6.11 Přenosová linka T

Obecná forma**:** *T<název> <A port (+) uzel> <A port (-) uzel> <B port (+) uzel>*

+ <B port (-) uzel>

+ [<název modelu>] LEN=<hodnota> R=<hodnota> L=<hodnota>

+G=<hodnota> C=<hodnota>

Model vedení je definován jako čtyřpól, který je zapojen mezi svorky A port a B port. V modelu jsou definovány součástky odpor, indukčnost, svod a kapacita.

Model vedení TRN

.MODEL <název modelu> TRN [parametry modelu]

2.6.12 Uživatelsky definovaná součástka X

X<název> [seznam vývodů] <název obvodu >[parametry:<<název>=<hodnota>>*]*

V PSpice je možné také nadefinovat vlastní typ součástky typu X.

2.6.13 Napěťově řízený spínač S

Obecná forma**:** *S<název> <+ přepínaná svorka> <- přepínaná svorka> <+ řízená svorka> <- řízená svorka>*

3 Konstrukce parseru

Parser je navržen pro zpracování vstupního souboru *.cir, ze kterého získává data. Pro každou součástku je navržena samostatná třída která obsahuje datové složky a metody pro práci s těmito datovými složkami.

3.1 Třída model

Třída model má datové složky, které slouží k uchovávání dat o modelu který je načten ze vstupního cir souboru. Její datové složky jsou jméno modelu, typ modelu a seznam parametrů. Pokud dojde při načítání vstupního souboru k načtení klíčového slova .model bude vytvořena instance třídy model a do této instance budou načteny příslušné hodnoty.

3.2 Metoda connectModelToDevice()

Tato metoda projde pole součástek, které jsou načteny ze vstupního souboru následujícím způsobem:

```
string dev = mod->get_device();
  if(dev.compare("GAFSET") == 0){
     int i = 0;
     for(; i < dev_counter; i++){
        if(souc[i]->get_id() != 'B')
           continue;
        Gaasfet *pom = (Gaasfet *)souc[i];
    if(pom->get_id() == 'B' && pom->get_model_name().compare(mod->get_name())== 0){
       pom->setModel(mod);
       return;
       }
     }
   }
```

Do proměnné dev se získá název typu součástky, poté je prohledáno pole souc[], ve kterém jsou uloženy všechny součástky ze vstupního souboru. V případě, že se shoduje typ modelu s typem součástky, je vyhodnoceno zda odpovídá název modelu s názvem která je uložen v poli součástek.Tento postup je aplikován na všechny součástky z pole součástek a pokud je nalezena shoda názvů modelů, potom jsou přiřazeny parametry modelu této součástce.

3.3 Metoda devices_cnt()

Tato metoda vyvrací počet součástek ve vstupním souboru. počet součástek je využit například při vytváření výstupního pole pro uložení všech součástek v obvodu.

3.4 Metoda destroy()

Metoda destroy() je využita k vyčištění paměti. Tato metoda je volána nakonec, kdy modul parseru doběhl dokonce.

3.5 Metoda print_souc()

Metoda slouží pro informativní výpisy.

3.6 Třída device

Tato třída slouží pro oddědění jednotlivých typů součástek, má dvě virtuální metody které jsou společné pro všechny typy součástek. V každí třídě odvozené od této třídy budou pak tyto virtuální metody překryty.

Výstupní pole potom bude rovněž typu device, to z toho důvodu, aby bylo možné do něj vložit jakýkoli typ součástky.

3.7 Postup parsování vstupního souboru

Vstupní soubor je čten po řádcích, pokud je zachyceno klíčové slovo .model je vytvořena instance modelu, která je poté přiřazena k příslušné součástce. Součástky lze od příkazů rozlišit tak, že nezačínají tečkou(.), jsou tedy rozlišovány na základě prvního písmena podle pravidel, které jsou popsány v předchozí kapitole o syntaxi jazyka PSpice. Při zachycení typu součástky je vytvořena příslušná instance.
V každé třídě je definována metoda parse, která se zavolá nad každou instancí a rozparuje řetězec podle pravidel v metodě parse. Rozparované prvky jsou pak uloženy do jednotlivých proměnných v instanci. Tato instance je poté uložena v poli součástek a je současně inkrementována proměnná *dev_counter* která udává pozici součástky v poli součástek.

3.8 Imtrepretace součástek

Následující tabulka ukazuje přiřazení použitých tříd a názvů modelů

Typ součástky	Instance
B	vytvoří instanci třídy Gaasfet() s modelem typu GASFET
C	vytvoří instanci třídy Capacitor() s modelem typu CAP
D	vytvoří instanci třídy Diode() s modelem typu D
E	vytvoří instanci třídy VCVS()
F	vytvoří instanci třídy CCCS()
I	vytvoří instanci třídy Independent()
J	vytvoří instanci třídy Junction() s modelem typu NJF, PJF
K	vytvoří instanci třídy Inductor() S modelem typu CORE
L	vytvoří instanci třídy InductorL() s modelem typu IND
M	vytvoří instanci třídy Mosfet() s modelem typu NMOS, PMOS
N	vytvoří instanci třídy Input()
O	vytvoří instanci třídy Output()
Q	vytvoří instanci třídy Bipolar() s modelem typu NPN, PNP, LPNP
R	vytvoří instanci třídy Resistor() s modelem typu RES
S	vytvoří instanci třídy Switch() s modelem typu VSWITCH
W	vytvoří instanci třídy CCSwitch() s modelem typu ISWITCH
X	vytvoří instanci třídy Subcircuit()
Z	vytvoří instanci třídy Igbt() s modelem typu NIGBT

Tab. 3.1 tabulka instancí

Závěr

Tato práce je rozdělena do tří kapitol. V první části je popsána konstrukce překladače rozdělení na jednotlivé části a jejich popis. Kompilátor je rozdělen do dvou částí, na přední část a zadní část. V přední části je prováděna analýza vstupního souboru. Touto částí se rovněž zabývá tato práce.

Druhá část je popsán jazyk PSpice. Je zde popsána syntaxe jazyka a jednotlivých součástek a příkazů jazyk PSpice.

Třetí kapitola pak popisuje vlastní konstrukci samotného parseru. V tomto modulu je naprogramována vstupní část, která čte zadaný soubor. Následně jsou vstupní data rozparsována a jsou podle popisu syntaxe uložena do pole součástek. modul parser má naprogramovány všechny analogové součástky a načítání modulů součástek a jejich přiřazení dané součástce podle jména modelu.

Seznam literatury a informačních zdrojů

[1] WIRTH, Niklaus. Compiler Construction. Zürich, 2005, ISBN 0-201-40353-6

[2] Müler, Karel. Programovací jazyky. [online] Dostupné http://service.felk.cvut.cz/courses/X36PJP/Skripta_prednasky.pdf

[3] Úvod do kompilátorů.[online].Dostupné http://kifri.fri.uniza.sk/~bene/vyuka/kompilatory/prednasky-prezentace/Predn-01-text.pdf

[4] Referenc manual. [online]. Dostupné http://www.electronics-lab.com/downloads/schematic/013/tutorial/PSPCREF.pdf