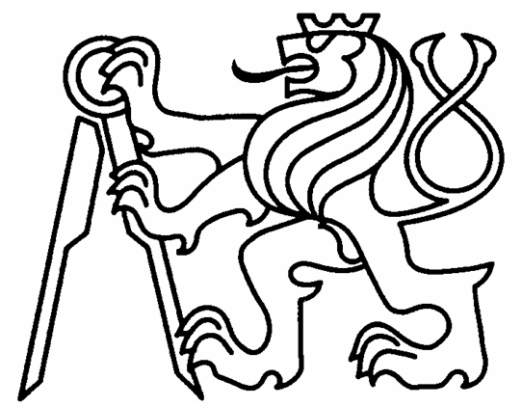
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická - Katedra počítačů



BAKALÁŘSKÁ PRÁCE

Automatické generování schémat z Netlistu
(Automatic generation of schematic diagrams from Netlist)

2005 Martin Kučera

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu. Uděluji souhlas s užitím 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

……..

Podpis

……..

Anotace

V této práci se věnuji shromáždění informací o známých technikách používaných při problému automatického generování schémat na základě obvodového netlistu. Stručně zde popisuji výhody a nevýhody některých z těchto technik a snažím se nalézt nejvhodnější techniky pro pozdější implementaci automatického generátoru schémat v rámci diplomové práce. Podařilo se mi implementovat testovací verzi programu, který bude použit při implementaci mé následné diplomové práce. Mezi předchozími pracemi jsem objevil systém SPAR, který je rozumným kompromisem všech dostupných systémů. Se souhlasem autora, jsem tento systém použil jako stavební kámen pro svůj testovací program.

Abstract

This Thesis presents the known state-of-the-art techniques used for automatic generation of schematic diagrams from a circuit netlist. The benefits and disadvantages of the techniques are described here. Then, the Thesis aims at finding optimal techniques for later implementation of a program for automatic generation of schematic diagrams in terms of my diploma thesis. I have succeeded in implementing a test version of a program that will be used as a basis in an implementation of my resulting diploma thesis. I have discovered the SPAR system among previous works. SPAR is a reasonable compromise of all the available systems and, with an agreement of its author, I have used it as a basis for my test program.

Obsah

Seznam obrázků

Kapitola 1.
Úvod

Cílem této práce bylo vytvořit automatický generátor schémat z netlistu, kde netlist je soubor definující logický obvod pomocí seznamu použitých obvodů, vstupních a výstupních portů výsledného obvodu a seznamu všech spojů v obvodu . Při výběru tohoto tématu jsem se zpočátku domníval, že se jedná o velice snadnou úlohu. Zanedlouho po důkladné a časově náročné analýze problému jsem dospěl k velice překvapivému závěru. Problém automatického generování schémat se skládá hned z několika dílčích problémů, jako je problém rozmístění součástek, jejich případné seskupování do větší funkčních celků, vedení jednotlivých spojů a v neposlední řadě uspořádání spojů vzhledem ke specifikovaným podmínkám. Nejtěžší na celé práci však bylo skloubit všechny tyto kroky a dospět k výslednému schématu, který bude estetický, lehce čitelný a bude vněm vidět přibližná funkce obvodu.

Při procházení desítek článků týkající se automatického generování schémat jsem nalezl jednu velice zdařilou práci od Steven T. Frezza z Pittsburghské Univerzity [1], týkající se přesně mého daného tématu. V této práci pan S.T.Frezza popisuje systém SPAR (Schematics Placement and Routing), který kombinuje algoritmický a heuristický přístup ke generování schématických diagramů.

Po domluvě s vedoucím práce panem P.Fišerem jsme dospěli k názoru, že by tato práce mohla být dobrým základem pro mou práci. Tématem mé práce bude tedy implementovat postupy popsané v systému SPAR. Vzhledem k veliké náročnosti všech kroků generování schématu se v této práci omezím na implementaci globálního a lokálního routeru. Problém seskupování a rozmístění součástek prozatím řešit nebudu a nahradím ho jednoduchým topologickým uspořádáním, tedy uspořádáním do pásů.

V této úvodní kapitole popíšu problémy spojené s automatickým generováním schémat V druhé kapitole uvedu jednotlivé techniky a postupy použité v systému SPAR. Ve třetí kapitole detailně popíšu, můj způsob implementace a její postup.

1.1. Přínos

Práce [1] je jistě velikým přínosem svými jedinečnými teoretickými postupy a několika novými technikami pro automatizované generování schématických diagramů. Prvním přínosem této práce je věnování pozornosti schopnosti generovat schémata taková, že je z nich patrná alespoň minimální funkčnost obvodu a dále pak generovat čitelná schémata, kde slovem čitelná myslím schopnost trasování daného obvodu. Dalšími přínosy je přítomnost postupů pro nakládání s místem ve schématu, řešeného pomocí informace o hustotě vedených čar, která je rozhodující pro globální router a techniky pro optimalizaci čitelnosti v lokálním routeru. Slovem router se rozumí soubor algoritmů řešících problém propojení jednotlivých spojů.

1.2. Problém generování schémat

Schématické diagramy jsou jedním z prostředků při návrhu digitálních obvodů, poskytující zřetelnou grafickou reprezentaci, která určuje obvod v různých úrovních. Jsou nepostradatelnou pomůckou ve všech stupních návrhu obvodů. Schéma se skládá z množiny modulů (součástek), spojených spoji, pomocí jejich vývodů. Existují různé grafické reprezentace modulů pro reprezentaci spojení, mezi vývody každého spoje se však používají čáry. Pomůcky pro kreslení schématu jsou základním stavebním kamenem hlavně u programů sloužících pro počítačový návrh (CAD) obvodů. V těchto programech jsou použity jako vstupní médium. Nicméně přechod ke generování obvodů na základě tex-

tového popisu posunul použití schématických diagramů od vstupního média k výstupnímu nebo zobrazovacímu médiu.

Naneštěstí mají jazyky pro textový popis obvodů tendenci být nesrozumitelné a oproti grafickému vyjádření postrádají jasný a názorný pohled. Odtud plyne potřeba generování schémat z textového popisu hardwaru. Při návrhu, kde je primárním vstupním médiem nakreslené schéma, muže být použita úprava automatickou optimalizací základní struktury návrhu a tím dochází k znehodnocení vstupního schématu. Návrháři však potřebují grafické podklady pro výsledky těchto úprav k uchování dosaženého návrhu. Grafická dokumentace slouží k ověření aplikovaných změn, umožňuje zjistit výsledky úprav po procesu optimalizace a sledovat návrh po celou dobu jeho iterativního vývoje. Schématický generátor musí být schopný oddělit základní rysy implicitní pro vstupní popis obvodu, vytvořit řádné a čitelné schéma. Naneštěstí pravidla platná pro vytvoření schémat s těmito kvalitami jsou těžko kvantifikovatelná a často potřebují speciální chování a výjimky k jejich zachycení.

1.2.1. Formulace problému

Cílem této práce je vytvořit algoritmický systém, generující „použitelné“ schématické diagramy z textového popisu obvodu (netlist). Budu řešit zejména problém generování schématu, pro obvody sestávající se z různých standardních logických obvodů (AND,OR,NAND, …). Generování předpokládá :

- Schopnost pracovat bez informací o hierarchii obvodu
- Modularitu schématu podle funkce
- Vytvoření čitelného schématu
- Efektivní a flexibilní využití místa ve schématu

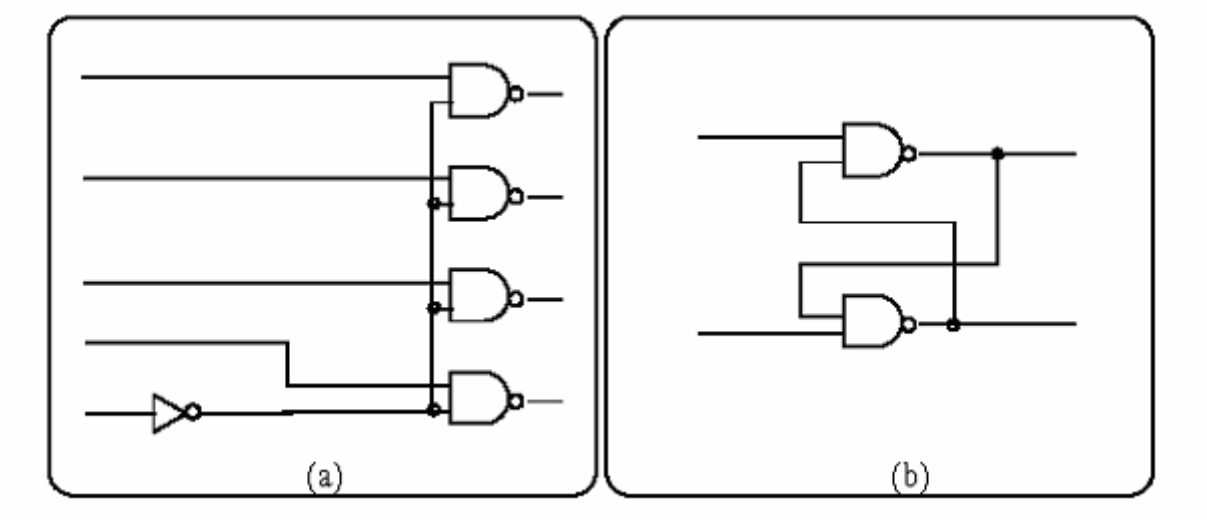
Na základě těchto předpokladů musíme nejprve určit vhodné charakteristiky pro schématické diagramy. Zadruhé musíme určit podmínky těchto charakteristik a zatřetí musíme tyto podmínky začlenit do algoritmů pro generování diagramů.

1.3. Charakteristické potřeby při generování schémat

Schéma je užitečná zpětná vazba pro návrháře nebo pro dokumentaci, schéma musí být schopné sdělovat „tok informací, času nebo součástek bez ohledu na fyzické rozložení“. Cílem schématu je podávat informaci o funkci návrhu. Při pouhém umísťování konstrukčních modulů do návrhu pomáhá tyto informace sdělovat. Pro člověka, který návrh provádí, schéma znamená porozumění celkovému návrhu. Proto musí být jakýkoliv pomocný program pro automatické generování schémat posuzován podle toho, jak výslednému schématu porozumí návrhář. V našem systému používáme k zjištění čitelnosti schématu koncept *trasovatelnosti* a *identifikace funkce*.

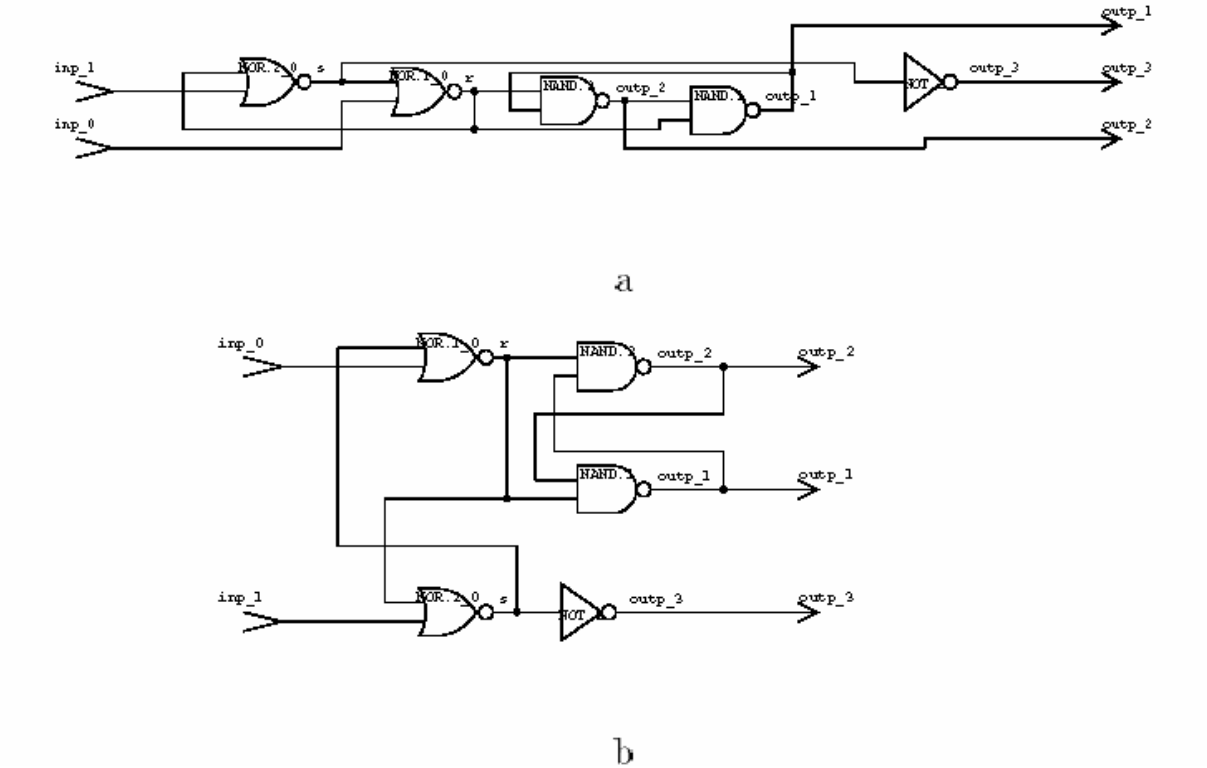
Trasovatelnost znamená ulehčení, pomocí kterého můžeme sledovat signály od jejich logických zdrojů k jejich cílům, a soustředí se na vedení cest, které nejsou komplexní a nevedou hustými oblastmi. *Funkční identifikace* se zaměřuje na umístění modulů, které mají funkční souvislost, a na oddělení skupin modulů, které mají funkci odlišnou. Trasovatelnost a funkční identifikace jsou dvě nepostradatelné charakteristiky, které nám pomáhají porozumět schématu, z tohoto důvodu je cílem systémů pro automatické generování schémat(dále jen ASG systémy) produkovat schémata s těmito dvěma charakteristickými rysy.

V ideálním případě musí být systém schopen rozpoznat funkčnost reprezentovanou schématem a trasovatelnost spojů ve schématu. Funkční identifikace vychází ze schopnosti rozpoznat běžné struktury. Obrázek 1 ukazuje dva základní příklady : jediné hradlo spouští několik dalších a identifikaci dvouhradlového klopného obvodu.



Obrázek 1 - Základní ukázky funkční identifikace ve schématu

Funkční identifikace a trasovatelnost jsou vzájemně závislé, při daném rozmístění nemohou být jednoduše optimalizovány buďto směrem k rozpoznání na základě vzorové struktury, nebo směrem k trasovatelnosti dané struktury. Pro příklad použijeme základní představu o trasovatelnosti, budeme tedy umísťovat jednotlivá hradla od zdroje signálu zleva doprava s použitím topologického uspořádání. Obrázek 2a ukazuje výsledek tohoto pohledu, použitím jednoduchého pravidla toku signálu zleva doprava při dodržení topologického uspořádání, při tomto pravidlu je hlavní signální cesta snadno lokalizovatelná. Obvod ukazující několik klopných obvodů je na obrázku 2b. Obrázek 2b používá sdružování do funkčních skupin, ale není tak striktní při dodržování pravidla toku signálu zleva doprava při topologickém uspořádání, ale oproti obrázku 2a je patrné zvýšení čitelnosti schématu. Tato ukázka ukazuje, že dva cíle, funkční identifikace a trasovatelnost, jsou provázány a nemohou být odděleny jednoduchými optimalizačními kritérii.



Obrázek 2 - Spojitost trasovatelnosti a funkční identifikace

Obojí, identifikace struktur a trasovatelnost schémat mají "přísné standardy" pro estetické předpoklady. Obecně platí, že je velice snadné říci co je na schématu *ošklivé*, jako na obrázku 2a. Zde je rozmístění takové, že jednu cestu lze snadno vystopovat. Funkční bloky a sledování dalších cest v tomto schématu jsou však nezřetelné. Na druhé straně je tedy více obtížné říci co je *dobré*.

Pro shrnutí, ASG systémy musí zobrazit netlisty jako fyzické uspořádání ikon identifikovatelnou cestou, a vést spoje zřetelně. Mimoto, jeho uspořádání musí být trasovatelné, ukazující souvislost každé cesty signálu rozmístěných modulů takovým způsobem, že každá cesta bude trasovatelná. Seskupené moduly na základě funkce mají často několik spojů, které ovlivňující vyhrazenou oblast, tyto dva cíle funkční seskupování a trasovatelnost signálů jsou často konfliktní, a tvořit jádro toho, co dělá generování dobrých schémat tak zajímavé ale zároveň obtížné.

1.4. Návrhové požadavky

Funkce ASG systému je použít informaci vloženou v netlistu pro generování blokového schématu. Protože návrhové systémy, s kterými pracujeme, používají optimalizační nástroje, očekáváme, že se ASG systémy nebudou spoléhat na hierarchické informace uvnitř netlistu, ale budou pro generování schémat používat *doprovodný* netlist(bez hierarchických informací). Očekáváme systém, který rozpozná funkcionalitu návrhu a odhalí struktury vložené do návrhu. Také očekáváme systém, který uspořádá ikony reprezentující moduly a spojí je tak, aby bylo výsledné schéma trasovatelné.

Jak jsem již zmínil, ASG systémy jsou určené pro práci s návrhy vytvořenými pomocí popisovacích jazyků. Malá ukázka hardwarového popisovacího jazyka je na obrázku 3. Toto je hierarchický popis hladinového klopného obvodu typu D, který používá konstrukční komponenty. Návrh klopného obvo-

du typu D obsahuje dva klopné obvody typu RS, jehož VHDL popis je uveden na obrázku 4. Tak jak byl tento klopný obvod navržen, tak také očekáváme že bude rozpoznán.

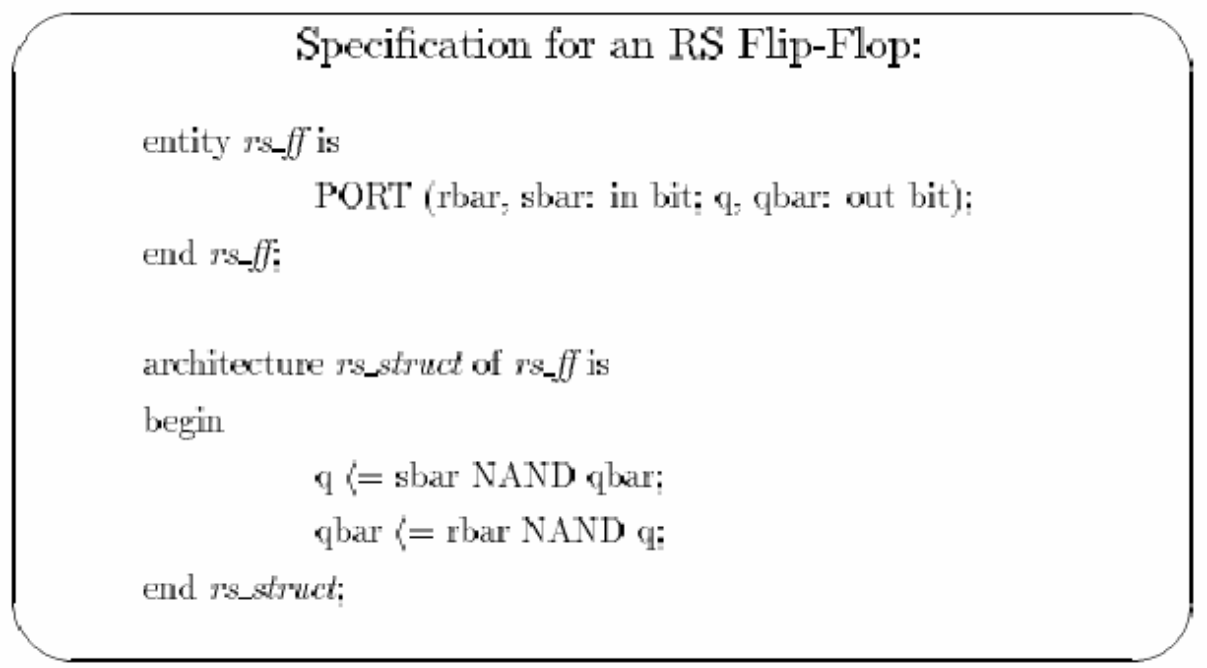
Tento popis je zkompilován do netlistu, který je použit pro vytvoření schématu. Netlist obsahuje názvy jednotlivých modulů a spojů, které jsou použity pro popis schématu a obsahuje jednotlivé spoje v návrhu. Tyto informace o spojení jsou vloženy uvnitř deklarací modulů tak, že každá deklarace obsahuje seznam signálů, které jsou připojeny k modulu. Toto je ukázáno na obrázku 5.

U ASG systému se očekává, že bude pracovat s doprovodnými popisy uvedenými v netlistu, jako ty, které jsou uvedeny na obrázku 5. Tento netlist neobsahuje hierarchické informace. Modul dvou klopných obvodů je obsažen v dalších modulech, pouze jejich jména jsou rozdílná, tato hierarchie je původně navržena ve VHDL popisu, ale už není dostupná pro ASG systém, při generování schématu. Pro splnění očekávané reprezentace klopných obvodů musíme strukturu, kterou reprezentují objevit skrze jejich spojení, protože to je jediná informace, která je dostupná pro ASG systém. Toto odhalení struktury uvnitř návrhu je klíčem k funkční identifikaci.

Specification for the D-Latch:

```
entity d_ff is
        port (d, c: in bit; q1, qbar1: out bit);
end d_ff;
architecture d_struct of d_ff is
        signal dbar, cbar: bit;
        signal rbar0, sbar0, q0, qbar0: bit;
        signal rbar1, sbar1: bit;
        label rs_0, rs_1;
        component rs_ff
                port (rbar, sbar: in bit; q, qbar: out bit);
        end component;
        for rs_0, rs_1: rs_ff
                use entity rs_ff(rs_struct);
begin
        dbar <= NOT d;
        rbar0 <= dbar NAND c;
        sbar0 <= d NAND c;
        cbar <= not c;
        rs_0: rs_ff port map (rbar0, sbar0, q0, qbar0);
        rbar1 <= qbar0 NAND cbar;
        sbar1 <= q0 NAND cbar;
        rs_1: rs_ff port map (rbar1, sbar1, q1, qbar1);
end d_struct;
```

Obrázek 3 -Hierarchický VHDL návrh jednoduchého obvodu typu D



Obrázek 4 - VHDL kód definující RS obvod použitý při definici obvodu typu D

Na obrázku 6 je schéma obvodu typu D vygenerované pomocí našeho systému. Dva RS klopné obvody jsou lehce rozeznatelné, protože byly vhodným způsobem umístěny. Algoritmus rozdělení a rozmístění kombinuje dva RS klopné obvody, které jsou součástí schématu a jsou charakteristické svou křížovou vazbou. Tyto moduly, které představují klopné obvody jsou označeny rs_0/NAND.1, rs_0/NAND.2 a rs_1/NAND.1, rs_1/NAND.2 jsou specifikovány na obrázku 5. Tok signálu v návrhu je také vhodný, vychází přímo zleva doprava od vstupu k výstupu.

Ačkoli je to jen malý příklad, obvod D na obrázku 6 dokládá úkol, který musí ASG systém dobře zvládnout. Toto vygenerované schéma je typickým vzorem identifikace funkce a trasovatelnosti.

Netlist information for the D-Latch:

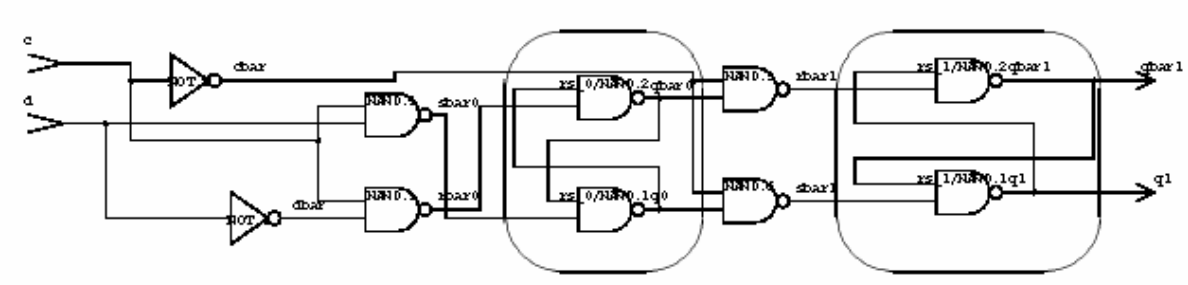
```
MX d_ff.d_struct
{
  MP {d IN;c IN;q1 OUT;qbar1 OUT;}
  FS {cbar dbar q0 qbar0 rbar0 rbar1 sbar0 sbar1}
  FG NOT.1 {NOT 1 d 1 dbar }
  FG NAND.3 {NAND 2 dbar c 1 rbar0 }
  FG NAND.4 {NAND 2 d c 1 sbar0 }
  FG NOT.2 {NOT 1 c 1 cbar }
  FG NAND.5 {NAND 2 qbar0 cbar 1 rbar1 }
  FG NAND.6 {NAND 2 q0 cbar 1 sbar1 }
  FG rs_1/NAND.1 {NAND 2 sbar1 qbar1 1 q1 }
  FG rs_1/NAND.2 {NAND 2 rbar1 q1 1 qbar1 }
  FG rs_0/NAND.1 {NAND 2 sbar0 qbar0 1 q0 }
  FG rs_0/NAND.2 {NAND 2 rbar0 q0 1 qbar0 }
}
```

Example module definition:

FG NAND.5 {NAND 2 qbar0 cbar 1 rbar1 }

is a NAND module having two inputs, nets *qbar0* and *cbar* and one output, *rbar1*.

Obrázek 5 - Netlist obvodu D



Obrázek 6 - Vygenerované schéma obvodu D

Kapitola 2.
Předchozí práce

Tato kapitola představuje ucelený přehled a zhodnocení předešlých prací zaměřených na systémy automatického generování schémat, diskuzi o běžných problémech spojených s tímto tématem a vysvětlím zde čím mě upoutal systém SPAR.

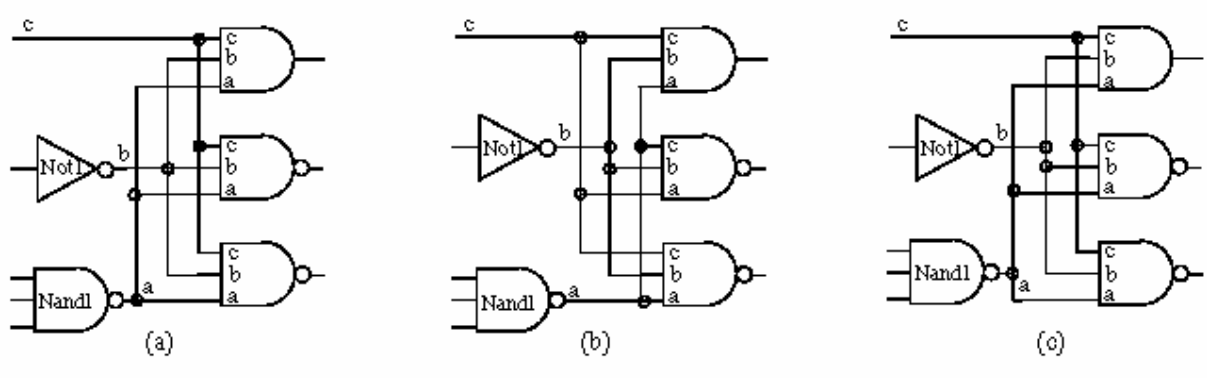
2.1. Minulost vývoje

Bylo mnoho ASG systémů zaměřených na generování schémat z netlistu. Některé řešily obecný problém generování schémat, zatímco jiné se specializovaly na problém generování logických schémat [2] [5], nebo vývojových diagramů [6] [9]. Cílem bylo reprodukovat to, co vytvořily projektanti a v nějaké úrovni modelovat pravidla, které tito experti používají. Jsou tu dva základní typy modelů : algoritmický a expertní systém. Nejprve se zaměřím na algoritmické modely a poté se budu věnovat různým expertním přístupům.

Nejvíce předchozích prací přistupují k problému cestou přirozených algoritmů, používajících kombinaci specializovaných heuristických algoritmů. Některé z prací jsou více názornější jako například R. J, Brennan, který představuje [4] dobrý heuristický přístup k problému propojení. May [8] [9] popisuje aplikaci techniky minimalizace grafu s některými působivými výsledky rozmístění. VISION [5] a jeho pozdější modifikace [11] docílily, při zaměření na problém rozmístění logických hradel, vynikajících výsledků pomocí technik kombinování informací o propojení se seskupováním a simulovaným ochlazováním. Ostatní [3] [12] používaly ke zjednodušení problému rozdělování. Práce popsaná v [2] byla jednou z prvních, která se pokoušela kombinovat rozmístění s propojením, použitím iteračních vylepšeních v algoritmickém přístupu.

Jako v systému SPAR zmíněné algoritmické přístupy došly k závěru, že rozmístění a propojování můžeme řešit jako oddělené problémy, a že vzájemné působení mezi rozmístěním a propojením může být efektivně zvládnuto bez přímého modelování návrhářských pravidel, podobně jako v expertních systémech. Vzájemné ovlivňování mezi rozmístěním a propojením je velice důležitá otázka, jedna z těch, které byly užívány k obhajobě použití expertních pravidel [13] a přidružení ceny vědomostí o zisku a splnění.

Vzájemné ovlivňování mezi rozmístěním a propojením se rozděluje do dvou kategorií, jasné a důmyslné. Jasné ovlivnění je založené na celkovém rozmístění modulů, to znamená, že pro moduly, které mají dvě hradla spojené křížovou vazbou, je lepší překřížení spojů mezi sebou(obrázek 1). Důmyslná ovlivnění jsou ty které ovlivní kudy povede spoj a zpravidla ovlivní kvalitu spoje. Obrázek 7 ukazuje příklad složitosti důmyslného ovlivnění. Tento příklad ukazuje, jak se musí při rozmístění modulů předvídat kde se bude vyskytovat spoj a objasňuje fakt, že oddělení úkolů rozmístění a propojení musí být provedeno opatrně.



Obrázek 7 - Vliv rozmístění na kvalitu propojení

Důmyslnost vzájemného ovlivnění mezi rozmístěním a propojením můžete vidět na obrázku 7, který ukazuje tři různá rozmístění jednoho schématu. Nepatrné změny v umístění modulů *Nand1* a *Not1* dokázaly změnit jednotlivé uspořádání spojů *a*, *b* a *c* . Na obrázku 7a mohou být spoje *a*, *b* a *c* přiděleny do jednoho ze tří sloupců mezi dvěmi sady modulů. Na obrázku 7b musí být spoj c umístěn na pravou stranu spoje b. Nakonec na obrázku 7c je situace, kde je celé propojení omezené, protože spoj *a* musí být vlevo od spoje *b*, který musí být postupně umístěn vlevo od spoje *c* . Změny rozmístění modulů *Nand1* a *Not1*, které jsem ukázal jsou velice malé, přesto omezují schopnost vytvořit nejlepší propojení. Tato důmyslná ovlivnění poukazují na to, že rozmístění a propojování spolu velice úzce souvisí a každé prostředky k jejich oddělení musí předvídat toto vzájemné ovlivňování.

Další práce zabývající se touto problematikou argumentují, že algoritmický přístup se nemůže zabývat vícenásobnými cíli a rozpoznáváním vzorů [13] tak, jak je měl návrhář na mysli. Tvrdí se v nich, že díky umělecké povaze úlohy generování schémat je obtížné ji zachytit použitím klasických programovacích technik [6]. Tito výzkumníci aplikovali na problém ASG systémů techniky Umělé Inteligence(UI) [6] [13], ty jsou však velice náročné na získávání znalostí. Expertní systém popsaný v [13] je nejdůležitější prací popisující problémy, které dělají UI přitažlivou, jsou tu komplexní pravidla, které používají lidé k rozlišení čitelnosti spojů. V popisu jejich systému,založeném na znalostech(KBS), také poskytují návod, jak ošetřit vzájemné ovlivňování rozmístění a propojení. Tyto systémy typicky svá pravidla rozdělují na kategorii hlavního rozmístění a kategorii propojování, a dále, pravidla pro propojování v KBS mohou také použity pro změnu rozmístění. Toto není běžné při algoritmickém přístupu.

Význam oddělení rozmístění od propojování je rozhodující pro problém generování. Ve skutečnosti předpoklad oddělení fáze rozmístění od fáze propojení je základem mnoha KBS přístupů a všech přístupů algoritmických.

Rozdělení rozmísťovacího problému do zajímavých pod-problémů umožňuje použití sady rozmísťovacích heuristik pro vytvoření dobrého schématu. Zaměřením rozmísťovacích algoritmů na omezený počet pevně daných zapojených modulů můžeme použít heuristiku, která se bude soustředit na řádné propojení modulů v lokální úrovni a díky tomu dosáhneme dobrých výsledků rozmístění tohoto podproblému. Například HALS [6] používá expertní pravidla k vytvoření shluků modulů v rozdělení, tímto se minimalizuje složitost výsledného úkolu. Systém Pablo [3] [12] rozděluje schéma , ještě před vlastním rozmístěním modulů, do bloků s pevně danou velikostí. Častým prvkem této myšlenky rozdělování je redukce složitosti, vyplívající z podproblému, s následným zvýšením rychlosti a dovoluje použít heuristiku na problém omezené velikosti a rozsahu.

2.2. Společné problémy

Už byly použity nejenom různé algoritmické a KBS techniky, ale také různá kritéria stanovující míru kvality schématu. Ty se odvodily od "ad hoc" pravidel, které používají jak projektanti, tak návrháři při vytváření schémat. V algoritmech se využívají pravidla rozdělená na kritéria pro dobré rozmístění versus kritéria pro dobré propojení, pravidla jsou shrnuty na obrázku 8.

Schéma musí ...

- Být vizuálně jednoduché, ale funkčně identifikovatelné.
- Být seskupováno do funkčních skupin.
- Minimalizovat shluky.
- Využívat a minimalizovat plochu pro kreslení.
- Být uspořádané tak, že co nejvíce signálových toků bude jednosměrných.
- Udržovat a ukazovat směr toku.
- Mít krátké spoje mezi logickými jednotkami.

Cesty by měly ...

- Používat malý počet uzlů, pro odbočky čar, a křížení.
- Vykazovat nízkou složitost protínání.
- Nebýt příliš dlouhé a nebo mít příliš mnoho ohybů.
- Minimalizovat počet ostrých ohybů.
- Mít minimální počet uspořádaných segmentů čar.
- Minimalizovat celkový počet segmentů, překříženích a celkovou délku cesty.

Obrázek 8 – Pravidla dobrého rozmístění a propojení

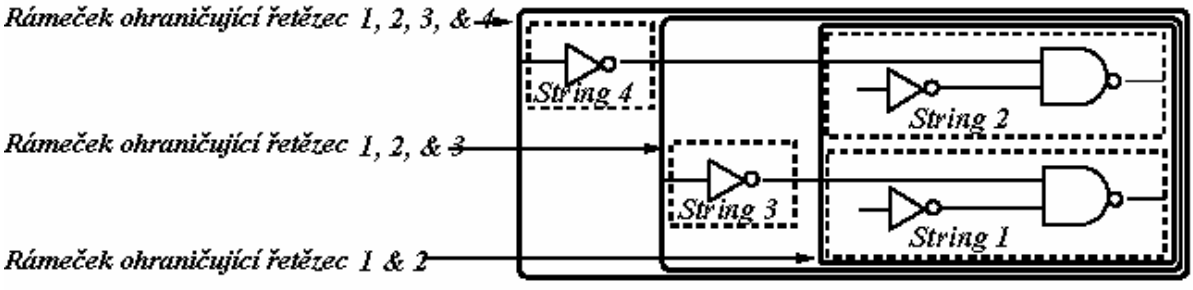
Tento hrubý přehled estetických pravidel pro schémata uvedený na obrázku 8 je důležitý pro dodržení přísných standardů, které jsou dány inženýrskou komunitou, ale není v nich jasně dáno, jak tyto standardy implementovat. Docházíme tedy k závěru, že všechna tato kritéria jsou užitečná a uspokojují dříve zmíněné potřeby uživatelů generovaných schémat.

Jasným cílem uživatelů schémat je trasovatelnost a identifikace. Pravidla shrnutá na obrázku 8 se pouze blíží k těmto dvěma způsobům. Mnoho předešlých systému selže, protože řeší tyto problémy buď na úrovni, při které nejsou schopni celkového řešení, nebo jsou příliš přísní při rozeznávání skupin modulů.Věřím, že jeden z důvodů těchto omezení je vysoce nepřirozený způsob, jakým předchozí systémy řídí fyzické rozvržení stránky. V systému SPAR jsou tyto problémy řešeny pomocí správy prostoru.

Slabou stránkou ostatních systému je používání pevně daný souřadnicový systém (mřížka), který je charakteristický pro mnoho ASG systémů. Běžně užívaná konvence je uspořádat mřížku tak, že každá ikona nebo cesta zabírá jedno pole v mřížce. Měřítko mřížky a umístění modulu jsou pak přizpůsobe-

ny estetickému vylepšení kompletního schématu. Tento způsob má výhodu v menší výpočetní náročnosti, protože „proces rozhodování je omezen na konečnou množinu možností“. Toto zjednodušení výpočetní složitosti problému je výhodné speciálně u systému používající techniku simulovaného ochlazování, nebo systémů založených na pravidlech. Bohužel spoléhání na uspořádání do mřížky nás omezuje při využívání volného prostoru. Mřížka také zhoršuje schopnost čistě modelovat představy *lokálních vztahů*. Tyto funkční vztahy dané množiny modulů jsou interpretovány dvěma způsoby : blízkostí souvisejících modulů a jejich vzdáleností od nesouvisejících modulů. Je těžké obsluhovat proměnné množství volného prostoru mezi moduly, nebo skupinou modulů.

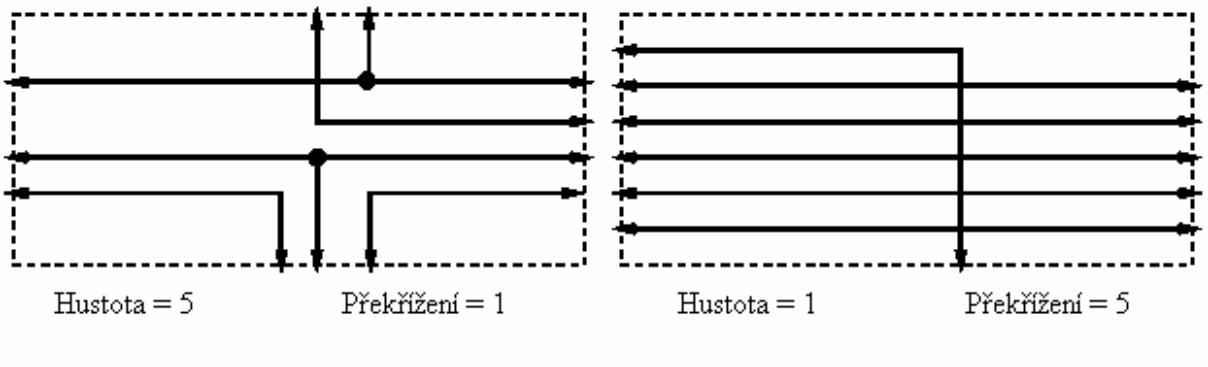
Ostatní práce, které nejsou limitovány použitím mřížky, jsou například *Pablo* a *HALS*. Na nich je zajímavé, že jsou určeny na řešení více obecnějšího problému automatického generování schémat, při kterém se očekává nepravidelný tvar a velikost hradel. Oba dva systémy používají pevně dané velikosti rozdělení a proto nejsou schopny odhalit funkční skupinu modulů, které jsou žádoucí.



Obrázek 9 - Nejstarší pokusy o správu prostoru

Oba tyto systémy používají *řetězce*, které jsou seznamem modulů spojených vazbou výstup→vstup mezi jejich vývody. Řetězec se skládá z jednoho a více modulů a je obklopen hraničním rámečkem v kterém jsou moduly umístěny. Oba systémy jsou schopné spojovat řetězce podle jejich hraničních rámečků. Pro jednoduché řetězce je to docela užitečné, ale vede to k problému s volným místem, tak jak ukazuje obrázek 9. Řetězce mohou pouze sousedit, ale ne se překrývat a to vede k tomu, že na schématu vznikají prázdná místa, která nemohou být využita. Na problému s volným prostorem muže být tato technika dobře aplikována pomocí rozmístění řetězců, které musí tvořit skupiny, jejichž vzájemným ovlivněním nebude docházet k problému s nevyužitým místem. Systém SPAR je schopný řešit tyto problémy a je prvotřídní při řešení obecného problému generování schémat. Tyto techniky jsou detailněji popsány v části 3.2.

Další oblast, kde se musí efektivně nakládat s volným místem, je trasovatelnost schématu. Všechny kvality používané při definování dobrého schématu (obrázek 8) jsou pravdivé, ale zakrývají závislost trasovatelnosti na blízkosti spoje s ostatními spoji a moduly. Blízkost spojů, nebo spíše blízkost rysů spoje brání trasovatelnosti daného spoje. Systém SPAR tyto rysy výslovně přiřazuje rohům a T uspořádáním čar, které tvoří cestu. Kdekoliv se tyto rysy vyskytují na malém prostoru jsou čáry vedoucí skrze tento prostor hůře trasovatelné. Tady si všimli, že se jedná o jakýsi prostorový jev takový, že počet rohů v dané oblasti (rohová hustota) stoupá a trasovatelnost čar vedoucích touto oblastí klesá. Obrázek 10 ukazuje dvě oblasti se stejným množstvím křížících se spojů. Počet překřížení a hustota jsou na obrázcích obráceny. Zastávám názor, že oblast s vysokým počtem hustoty se obtížněji sleduje, než oblast s vysokým počtem překřížení. To ukazuje, že pro docílení trasovatelnosti je lepší měřit hustotu rohů, raději než počet překřížení, které obsahuje.



Obrázek 10 - Počet překřížení versus hustota

Všechny předchozí ASG systémy používaly pravidla pro propojování schématu, která jednoduše minimalizovaly počet překřížení, nebo používaly specifická pravidla řazení, která používají lidští návrháři. Já tvrdím, že je to nedostatečné pro trasovatelnost spoje a trasovatelnost není ve všech případech závislá na počtu překřížení, ale spíše na zahlcení oblasti, přes kterou čára vede. Řešení s nejmenším počtem překřížení nemusí být nevyhnutelně nejlépe trasovatelné.

2.3. Co mě zaujalo na systému SPAR

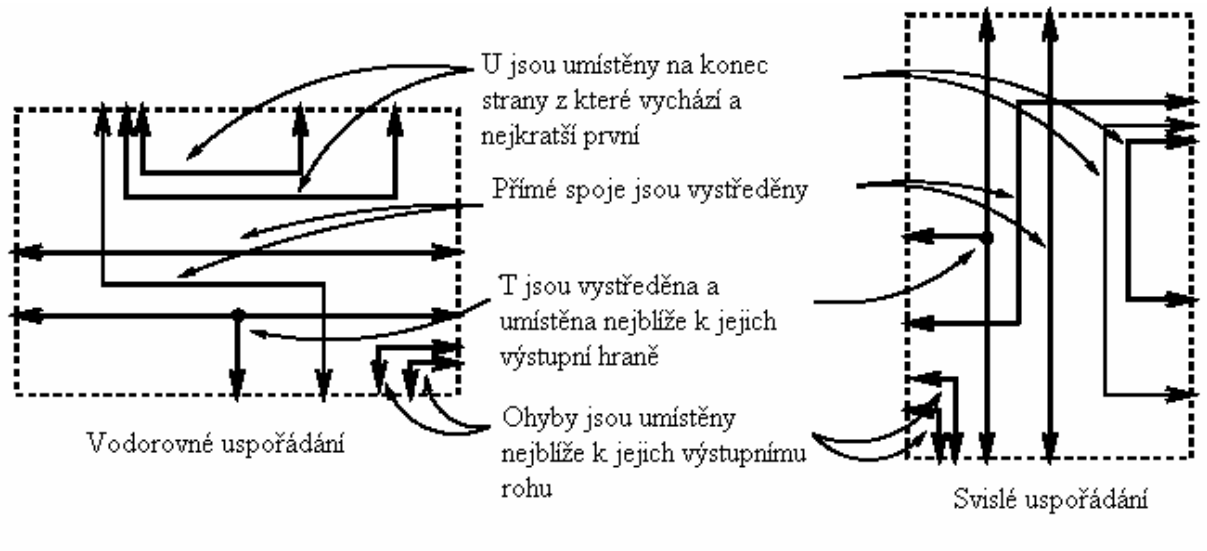
V této části bych se chtěl jen krátce zmínit o zajímavostech systému SPAR, kvůli kterým jsem na jeho základě zpracoval svou práci.

Základem celého systému SPAR je *identifikování funkčních vztahů* mezi jednotlivými moduly a snadná *trasovatelnost* jednotlivých spojů. Tohoto cíle dosáhli použitím předchozích technik, které zkombinovali s vlastní technikou správy volného prostoru.

Správa volného prostoru je založena na používání *virtuálního místa* jako mapy, které se můžeme dotazovat na informace. To nám dává výhodu nad více tradičními metodami. K dosažení výhodného rozmístění je netlist rozdělen a umístěn na virtuální stránky. Stránky jsou poté sloučeny dohromady a vytvoří kompletní rozmístění.

Tento postup je implementován pomocí metody rohově svázaných dlaždic [12]. Tato technika poskytuje čisté a poddajné řešení při správě nepravidelných, dynamicky vytvářených, dvourozměrných ploch. Užitečnost a použití této techniky je popsáno v části 3.2. Nejdůležitější charakteristikou této techniky je schopnost slučování sousedních dlaždic.

Globální propojování využívá informaci o zaplnění, která je důležitá pro trasovatelnost. Při lokální propojování se klade důraz na rozmístění čar pro jednotlivé rohy, které jsou výsledkem globálního propojení. Pravidla pro rozmístění jednotlivých typů spojení v rohu jsou ukázány na obrázku 11.



Obrázek 11 - Pravidla řazení pro snadné čtení

Všechny nejdůležitější myšlenky použité v systému SPAR jsou uvedeny v následující kapitole.

Kapitola 3.
Popis systému „SPAR“

Tato kapitola představuje přehled systému SPAR. Představuje jakým způsobem přistupujeme k rozsáhlým úkolům, jako je správa prostoru, rozdělení, rozmístění modulů, globální a lokální propojování. Detaily implementace jednotlivých problémů budou vysvětleny v kapitole 3.

3.1. Oddělení úlohy rozmístění a propojení

SPAR kombinuje algoritmický a heuristický přístup k automatickému generování schémat. Tento přístup začíná tvrzením, které říká, že problémy rozmístění a propojení jsou vzájemně závislé, ale mohou být odděleny. Daná povaha rozmístění může znemožnit vytvoření trasovatelného propojení, SPAR rozděluje rozmístění modulů do systémových skupin založený na propojení. To znamená, že se snažíme zjistit rozdělení a rozmístění, které vykazuje rozumnou reprezentaci funkční povahy netlistu. Předpokládáme, že dobře rozdělené schéma bude i dobře rozmístěné a bude dodržovat tok signálu, který existuje uvnitř jednotlivých rozdělení a bude dosaženo i dobrého propojení. Tento předpoklad je docela rozumný, protože toto rozmístění dodržuje první ze dvou požadavků na generování - Funkční identifikaci.

3.2. Správa prostoru

SPAR explicitně řídí prostor pro rozmístění a propojení. Tento prostor je tvořen pomocí množin rohově spojených dlaždic, každá z nich má svůj specifický účel. Rozmístění všech vnitřních modulů(ne systémových portů) vznikne na oddělené množině dlaždic, každé rozdělení je umístěno v jeho vlastním dlaždicovém prostoru. Tyto "stránky" jsou pak sloučeny do jedné dlaždice. Relativní informace o rozmístění je pak použita pro převod všech vnitřních modulů do prostoru dlaždic, které jsou použity pro propojování. Výhoda tohoto převodu je ve stanovení ohraničujícího rámečku při rozmístění a jeho použití při ohraničení oblasti uvnitř které se bude vyskytovat očekávané propojení.

Propojení je vytvořeno použitím dvou vrstev dlaždicového prostoru pootočených o 90°, tak, aby mezi sebou mohly jednotlivé propojení soupeřit o prostor a tak jsou vygenerovány. Použití dvou vrstev prostoru dlaždic využívá jedné z charakteristik algoritmu rohového spojení, to jsou volné dlaždice(ty které nejsou obsazeny) roztáhlé po horizontálním, nebo vertikálním směru, až tak daleko jak je to možné. Díky použití dvou o 90° pootočených prostorů lze rychle vyhledat volné horizontální, nebo vertikální oblasti pro jakýkoliv bod.

Prostor dlaždic také používáme pro lokalizování modulů a spojů. Dlaždice obsahují ukazatele na moduly a spoje, které obsahují a jsou použity pro řízení algoritmů. To je obzvláště důležité pro inkrementální rozmístění systémových výstupů, které jsou přidány až do dlaždicového prostoru, který obsahuje moduly a spoje. Když je přidán nový modul, tak je zkontrolována platnost informací, které obsahují ovlivněné dlaždice.

3.3. Problém rozdělení

Vhodné rozdělení vytváříme objevováním a zvýrazňováním vnitřních struktur uvnitř návrhu, abychom dosáhli funkční identifikace mezi podstatnými moduly. Důležité vztahy jsou dědičnost, globální zpětná vazba a křížová vazba. Dědičnost je do schématu aplikována jako sled spojů, které spojují moduly, je-li výstup modulu spojen s vstupem druhého modulu, pak říkáme, že je dědicem prvního modulu. Křížová vazba je tam, kde je jedna množina modulů dědí a zároveň je pro další množinu modulů, od

které dědí, dědičná. Každá množina má výstup, který je připojen na vstup další množiny. Globální zpětná vazba se od křížové vazby odlišuje v tom, že je výstup celé množiny modulů připojen ke vstupu stejné množiny modulů.

Našli jsme efektivní rozdělování, kterého může být dosaženo pomocí informací o spojeních daných v netlistu. Tato rozdělení budou mít dopad na proces rozmístění a propojení. Moduly jsou umístěny uvnitř rozdělení tak, že při seskupování se zahrnou jen lokální zpětné vazby a zpětné vazby globální se vynechají pro algoritmy mezi rozděleními.

To se provede použitím techniky seskupování, založené na zkoumání spojů mezi dvěmi danými množinami modulů. Matice spojů je sestavena z modulů, které obsahuje schéma a seskupením pevně spojených modulů jsou vytvořena seskupení. Seskupování pokračuje do té doby, dokud bude navrhované sloučení seskupených skupin produkovat dobré rozdělení.

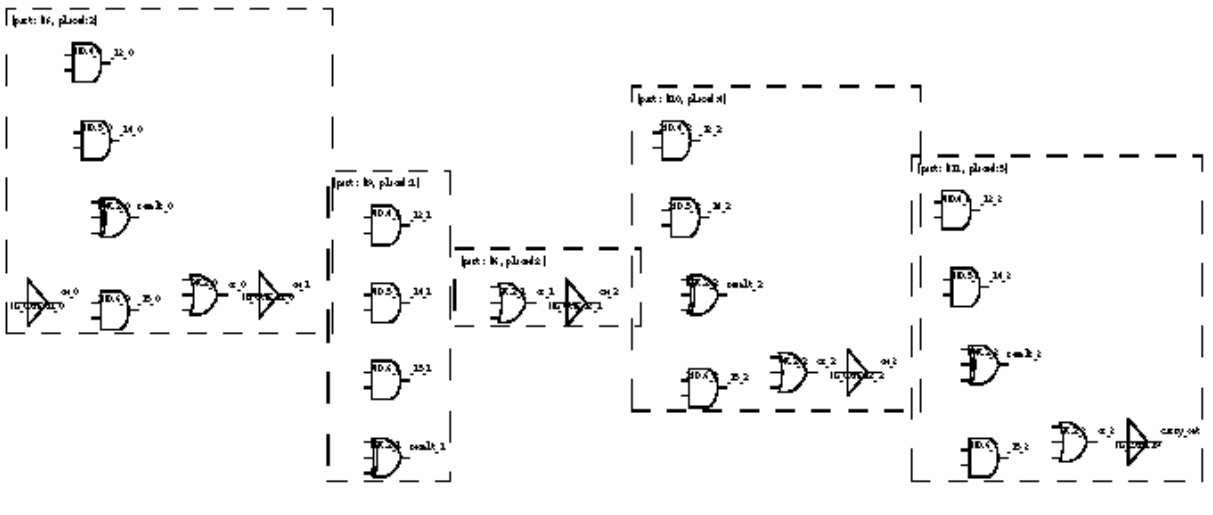
3.4. Rozmístění modulů

Po daném specifickém rozdělení se provádí rozmístění jednotlivých rozdělení odděleně. Skupina modulů uvnitř rozdělení se používá pro vytvoření virtuální stránky a obsažené moduly jsou rozmístěny bez ohledu na spojení vně rozdělení. Tyto stránky budou rozmístěny samostatně, mezirozdělovací rozmístění vysvětlím později. Během rozdělování jsou moduly zformovány do řetězců a na toto řetězcové formování jsou "přísné standardy". Pro rozmístění modulů uvnitř rozdělení je velice důležitý výběr a uspořádání modulů. To je děláno na základě jejich lokálních spojů a uvažuje přitom s jejich délkou, hustotou a potřebami pro ohyby. Uvnitř rozdělení se pro mnoho spojů předpokládají krátké vzdálenosti, aby byly vedeny přímo.

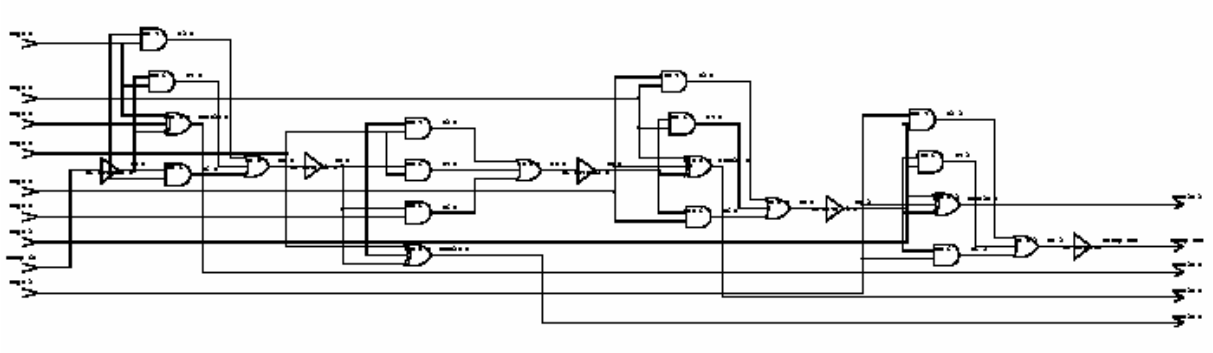
Rozmísťovací algoritmus pro moduly uvnitř rozdělení je série vylepšení algoritmu použitého v systému Pablo. Na dané virtuální stránce jsou sady řetězců vytvářeny tak, aby dodržovaly tok informací : vstup→výstup zleva→doprava. Tyto řetězce jsou slučovány dohromady dokud nejsou přidány všechny moduly v rozdělení do stránky.

Zlepšení Pablova algoritmu pro vytváření řetězců spočívá ve formování komplexních řetězců. To dovoluje řádnou identifikaci důležitých rysů, jako je například lokální zpětná vazba mezi moduly. Dalším důležitým rozdílem je použití informací o pozici signálu uvnitř řetězce při slučování řetězců dohromady. Při použití dlaždicového prostoru si nemusíme dělat starosti s komplexními tvary řetězců a jsme schopni využít volný prostor, který může zbýt, jak ukazuje obrázek 9. Heuristika, která řídí slučování řetězců, používá propojení neumístěných řetězců a polohu, ke které je řetězec připojen a používá to pro určení pozice nového řetězce v rozdělení.

Jakmile jsou moduly uvnitř rozdělení rozmístěny na individuální stránky, je po sloučení těchto stránek rozmístění kompletní. Toto rozmístění musí udržovat nízký počet dlouhých spojů, aby nebylo schéma plné dlouhých propojeních, které ruší funkční identifikaci. Pak jsou rozdělení sloučeny dohromady použitím souboru heuristik, podobných těm, které se použily při slučování řetězců uvnitř rozdělení. Tyto stránky jsou sloučeny jedna po druhé dokud nejsou rozmístěny všechny moduly obsahující vývody. Na obrázku 12 je ukázka rozdělení a rozmístění pro 4bitovou sčítačku a na obrázku 13 je konečná podoba sčítačky po propojení.



Obrázek 12 - Rozdělení a rozmístění pro 4bitovou sčítačku.



Obrázek 13 - Výsledné zapojení 4bitové sčítačky po propojení.

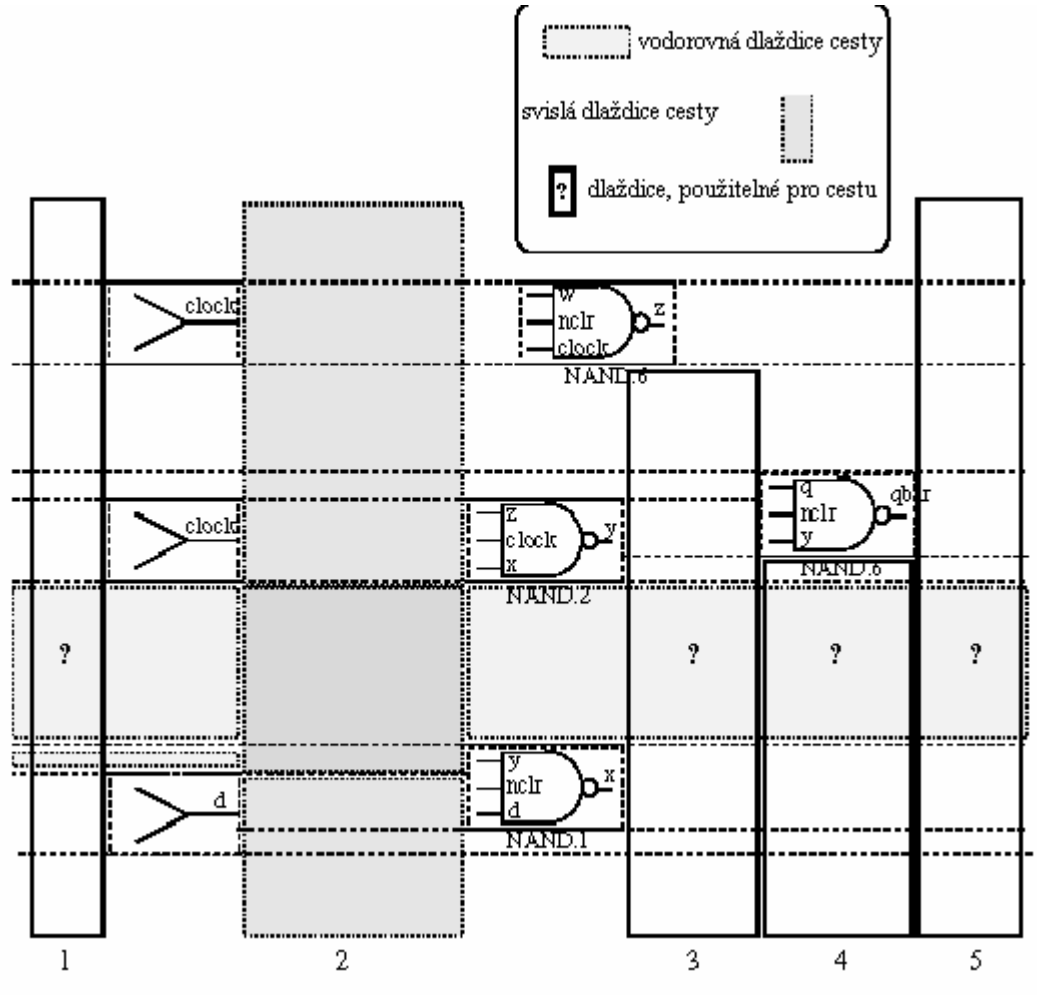
Koncové porty jsou zpracovávány samostatně, protože jsou kladeny velice specifické požadavky na jejich umístění. Vstupní porty se očekávají na levé straně a výstupní porty na straně pravé, všechny uspořádané do sloupců. Mimoto se porty očekávají zarovnané, kdekoliv je to možné, na jejich zdroji, nebo na levém kraji cílového vývodu. Toto je hotové po prvním rozmístění modulů do schématu a jejich propojení. Tyto propojovací informace jsou uloženy do prostoru dlaždic, který je použit při propojování a tyto informace jsou pak použity při rozmísťování portů. Porty jsou inkrementálně přidány do propojovaného prostoru a propojení tím je kompletní. Detailní popis tohoto problému je uveden v části 4.5.

3.5. Globální propojení

Propojování v systému SPAR je rozděleno do dvou fází: globální propojení, kde se určí hlavní místa spojů a jejich ohybů, a lokální propojení, ve kterém se na tyto překrývající oblasti dívá, jako na kombinaci omezení, kterým se musí vyhovět. Globální propojení určuje počet a umístění ohybů, zatímco lokální router rozmísťuje rohy a vytváří mezi nimi spojení. Tyto dvě techniky maximalizují čitelnost výsledného schématu.

Globální router vykonává heuristické hledání, vytvořením vícebodových expanzí při použití cenové funkce založené na složitosti cesty (např. počet ohybů) a na zaplnění cesty (to znamená počet dalších spojů a rohů, které se vyskytují v oblastech kudy spoj vede). Tímto paralelním prohledávání do hloub-

ky chceme pro každý spoj nalézt nejnižší hodnotu složitosti/zaplnění. Příklad expanze jedné cesty spoje je na obrázku 14.



Obrázek 14 - Expanze jedné cesty spoje *y*.

Zaplnění je funkcí ostatních spojů, kromě spoje hledaného, a lze vyhodnotit pouze pokud je navržena sada globálních propojení. Míra zaplnění může být ohodnocena až po dosažení propojení, a spoje jsou tedy přeneseny až na základě existujících informací o zaplnění. Metrika pro určení ceny cesty je dána následujícím vzorcem :

$$cena\ cesty = cena\ zaplnění\ +\ počet\ rohů \times cw + délka\ cesty \times lw \qquad (1)$$

kde :

$$délkacesty\ =\ \overbrace{\sum_{i=1}^{počet\ dlaždic} průměrná\ vzdálenost\left(dlaždice_{\,i}, dlaždice_{\,i+1}\right)}^{Průměrná\ délka\ cesty} + \overbrace{\left(\left|termX - \bar{x}\right| + \left|termY - \bar{y}\right| \times f\right)}^{\substack{Odhadovaná\ vzdálenost\ k \\ nejbližším\ u\ portu}} \qquad (2)$$

Kde $\bar{x}$ a $\bar{y}$ je pozice posledního známého rohu v předpokládané cestě a *termX* a *termY* je pozice portu nejblíže k ($\bar{x}$, $\bar{y}$).

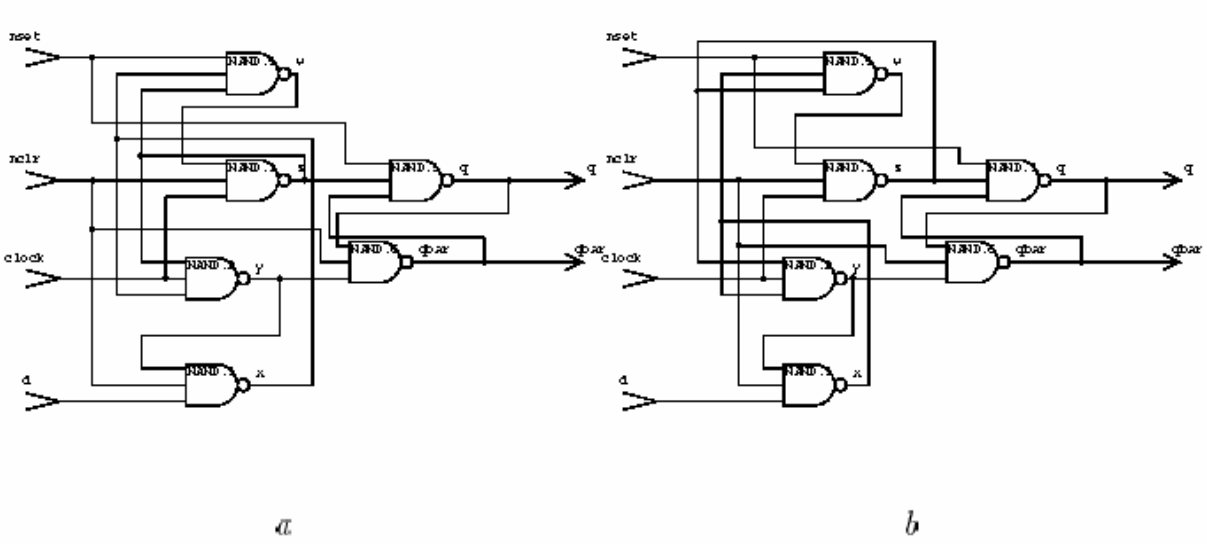
Pokud platí, že *Použité dráhy* < *Dostupné dráhy*, pak :

$$Cena\ zaplnění\ = \frac{Použité\ dráhy}{Dostupné\ dráhy} \times w \qquad (3)$$

‚jinak platí :

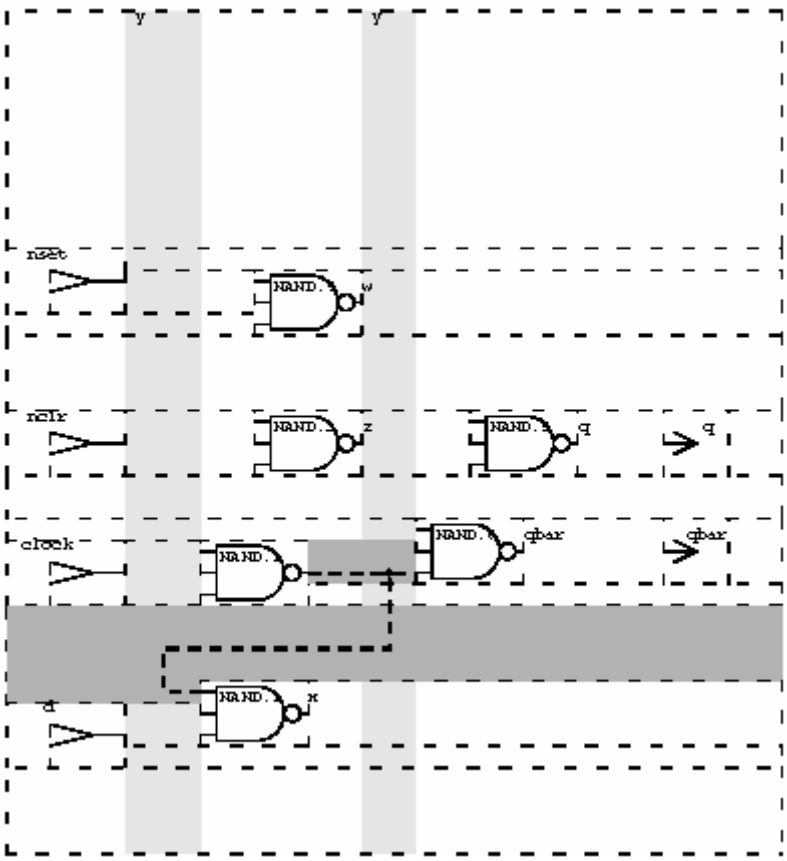
$$Cena\ \ zaplnění\ \ = 2w \times \frac{Použité\ \ dráhy}{Dostupné\ \ dráhy} + c \qquad (4)$$

Kde *w* je konstanta, která je měřítkem hodnoty zaplnění vzhledem k ostatním hodnotám odhadované ceny cesty. Na obrázku 15 je zobrazeno výsledné schéma bez použití oceňování zaplnění (a) a s použitím oceňování zaplnění (b).



Obrázek 15 - Propojení obvodu SN7474 s vypnutým oceňováním zaplnění (a) a se zapnutým (b).

Globální cesta je pouhým plánem, kudy dané spoje povedou, a není přesným vyjádřením cesty. Spíše je důležitý počet a hlavní umístění ohybů spoje, které jsou uvedeny v dlaždicích po kterých propojení vede. Proces specifického umístění rohů se nechává na rutině pro lokální propojení. Příklad výběru dlaždic pro určitou cestu spoje je uveden na obrázku 16.



Obrázek 16- Globální cesta pro spoj *y* vybranými dlaždicemi.

3.6. Lokální propojení

Lokální propojování je založeno na nové technice propagování omezení a řídí rozmístění rohů, které vytvoří propojení pro daný spoj. Spoje byly při globálním propojení umístěny do prostoru dlaždic a byl jim přidělen seznam dlaždic, po kterých budou vedeny. Dlaždice přiřazené při globálním propojení vymezují minimální oblasti, kde budou vedeny spoje při lokálním propojení. Globální cesta je tedy užívána k omezení cesty lokální. Tato omezení se nazývají rozsahy. Omezující rozsahy se mohou rozšířit přes několik dlaždic. Několik spojů může používat stejnou propojovací dlaždici, tak se mezi sebou budou nutně překrývat jejich rozsahy. Překrývání musí být vyřešeno dříve, než se začnou spoje rozmísťovat a to je hlavní úloha pro lokální rozmístění.

Takto je stanoven konečný počet rozsahů tvořící spoj a také konečná množina spojů, vedoucí přes nějakou stanovenou oblast, které musejí být propojeny a my tím redukujeme problém lokálního propojení na problém vyřešení omezení . Problém je nalézt nejlepší řešení pro umístění rohů pro každý spoj v každé dlaždici, počínající omezeními z globálního propojení. Během tohoto řešení musí být použit problém křížení čar zmíněný v části 2.2. V každém případě existuje uspořádání, řešící problém umístění rohů, jehož výsledkem bude dobře vedená cesta ve schématu.

Dlaždice z globální cesty jsou použity pro určení počátečních rozsahů které definují, kde se budou vyskytovat rohy. Lokální router expanduje rozsahy dané globální cestou do všech přilehlých dlaždic, které neucpou tok. Kompletní rozsah, v kterém se může nacházet nějaký daný roh, se může rozšířit na několik přilehlých dlaždic.

Vedle jednoduchého výběru pozic rohů pro každý spoj nám vyvstává otázka, jak sdělovat informaci o závislosti umístění jednoho rohu při rozhodování o dalším. Například umístění rohu X ovlivní umístění (nebo bude ovlivněn) rohu připojeného k X. Podstatou tohoto problému je rozhodnutí, který roh

bude umístěn jako první. Výběr jedné pozice bude totiž omezovat možnosti dalším rohům, které ještě nebyly rozmístěny. Další komplikací je odlehlost dlaždic, v kterých se rohy vyskytují. Přesto že je jeden roh pevně umístěn, změna se musí rozšířit ke všem dalším rohům, které jsou ze stejného spoje připojeny buďto vodorovně, nebo svisle.

Řešení těchto problémů spočívá ve využití informací o zaplnění, získaných ve fázi globálního propojování. Jako první se ohodnotí pozice všech rohů v nejhorší dlaždici pomocí hladového algoritmu. Toto ohodnocení je vytvořeno odděleně, pro vodorovné a svislé části, a zahrnuje vytvoření seznamu rozsahů použitých v dané dlaždici. Rozsahy jsou pak odděleny tak, že vyhovují našim *Pravidlům jednoduché čitelnosti* zmíněných v části 2.2. Každé oddělení omezí rozsah, v kterém se může roh vyskytovat. Pozice rohu jsou omezeny a všechny spojené rohy jsou také omezeny. Touto cestou je ohodnoceno více dlaždic, tím roste omezení umístění rohů, a předává se dalším dosud neohodnoceným dlaždicím. Tento postup pokračuje doté doby, dokud všechny rohy nemají svůj jedinečný rozsah pro své umístění. Nejvíce ucpané dlaždice jsou ohodnocovány jako první a tím budou nejvíce ucpané (nejméně trasovatelné) množiny rohů nejvíce čitelné.

Výhodou tohoto přístupu je čisté řešení rozmísťovacího problému tak, aby v každé oblasti bylo uděláno jen malé množství rozhodnutí. Typické ucpané oblasti obsahují několik spojů, které mají přečnívající omezení, umístěním těchto spojů jako první a předáním informací o rozhodnutí svázaným rohům dosáhneme většího omezení dalších dlaždic. Postupem tohoto algoritmu se postupně dlaždice stávají snadnější pro rozhodování. V podstatě tato technika identifikuje důležité rozsahy, které mají být rozhodnuty, a odkládá řešení nekritických (ucpaných) rozsahů, dokud není upevněno více kritických rozsahů.

Nyní máme přehled o klíčových myšlenkách algoritmů použitých v systému SPAR a jsme připraveni diskutovat o dalších detailech jejich implementace.

Kapitola 4.
Vlastní implementace

V této kapitole se věnuji svým ranným představám problému automatického generování schémat. Krátce zde popisuji mou předchozí implementaci ASG systému a důvody proč jsem přešel na systém SPAR.

4.1. Předchozí implementace

Půl roku před objevením systému SPAR jsem se pokoušel implementovat svůj vlastní systém. Bohužel se mi v té době nepodařilo získat informace o problému automatického generování schémat, ale získal jsem práci doc. Servíta [11], která se zabývá implementací problému propojení na deskách s plošnými spoji. Tento problém ve své práci řeší vytvořením prostoru mřížky a nad tímto prostorem používá vlnový algoritmus, který využívá cenovou funkci a prostorovou expanzi. Jako základ prostorové reprezentace jsem tedy použil mřížku. Jak jsem postupem vývoje zjistil, tato reprezentace je opravdu těžkopádná a velice těžce se vyrovnává s problémy s volným prostorem, zmíněnými v části 2.2. Navíc nalezení vhodné cenové funkce by vydalo na celou diplomovou práci, protože problém propojení řešený doc. Servítem je velice rozdílný od problému automatického generování schémat. Hlavní rozdíl těchto dvou problému je v čitelnosti. U problému propojení desky plošného spoje se neklade důraz na estetické hledisko, které je stěžejní pro schémata. Nicméně jsem se díky této práci získal matriály zmíněné v 2. kapitole a k vlastnímu systému SPAR.

Dále jsem se snažil implementovat takový systém, který bude nezávislý na operačním systému. Při tomto požadavku jsem narazil na problém, jaký zvolit typ výstupního souboru. To mě přivedlo k myšlence vytvořit abstraktní třídu, která bude sloužit jako rozhraní pro kreslení schématu. To se mi také povedlo a výstupem mého programu mohl být jakýkoliv typ souboru, pokud byl implementován jako syn abstraktní třídy tohoto rozhraní, později jsem přešel na Postskript.

4.2. Přechod na systém SPAR

Zhruba po půl roce vývoje jsem objevil popis systému SPAR a ten mě tak uchvátil, že jsem obratem navázal komunikaci s autorem tohoto systému Ing. Frezzou [8]. Ten mi ochotně předal část zdrojových textů, které implementují celou funkčnost systému popsanou v kapitole 3. Po konzultaci s vedoucím mé práce Ing. Fišerem jsme usoudili, že by byla škoda získané zdrojové texty zahodit a dohodli jsme se na jejich použití.

4.2.1. Implementace hlavní funkce

Prvním krokem bylo napsání vstupní funkce, která provede vlastní generování schémat. Jako nápovědu jsem měl k dispozici tělo vstupní funkce programu XCircuit, který v sobě má zakomponován ASG systém SPAR. Níže uvedený text je výňatek z konečné verze těla této funkce:

```
/* načteni vstupního souboru */

read_bench(input_file);
partition_count = partition();

if (collapseNullModules == TRUE) clip_null_gates();

erase_systerms();
```

```
place_first_strings();
box_placement();
placement_prep();
modulesJustPlaced = partition_placement();
/* Nastaveni hraničního rámečku */
systerms = make_room_for_systerm_placement(&systermNets
               ,&x1, &y1, &x2, &y2);

/* Propojení */
currentPartition = 0;
for(ml = systerms; ml != NULL; ml = ml->next)
     pull_terms_from_nets(ml->object);
routingRoot[VERT] = (tile **)calloc(partition_count + 2
                                , sizeof(tile *));
routingRoot[HORZ] = (tile **)calloc(partition_count + 2
                                , sizeof(tile *));

create_routing_space(modulesJustPlaced, currentPartition
     , x_sizes[currentPartition], y_sizes[currentPartition]
     , xfloor, yfloor);

nl = first_global_route(modulesJustPlaced, currentPartition
          ,FALSE, congestion_rule);

/* Vytvoření lokálního propojení pro lokalizaci portů */
local_route(systermNets, currentPartition, FALSE);

/* Přidání portů do schématu */
modulesJustPlaced = fine_systerm_placement(&x1, &y1
                          , &x2, &y2);

/* Přidání portů zpět k jejich vlastním spojům */
for(ml = systerms; ml != NULL; ml = ml->next)
     add_terms_to_nets(ml->object);

/* Dokončení propojení */
reset_boundaries(routingRoot[HORZ][currentPartition], x1, y1
     ,x2, y2);
reset_boundaries(routingRoot[VERT][currentPartition], y1, x1
, y2, x2);
xfloor = x1;    x_sizes[currentPartition] = x2 - x1;
yfloor = y1;    y_sizes[currentPartition] = y2 - y1;

nl = incremental_global_route(modulesJustPlaced, modules, nl
          , currentPartition);

local_route(nets, currentPartition, TRUE);

print_local_route(lroutputFile, nets);
```

4.2.2. Netlist ve formátu BENCH

Dále bylo potřeba přepracovat syntaktický analyzátor vstupního souboru netlistu. My používáme netlist v bench formátu, který je velice jednoduchý a obsahuje pouze seznam vstupních a výstupních portů a dále seznam jednotlivých modulů a zapojení jejich vstupů a výstupů. Příklad takového netlist souboru je na obrázku 17. Netlist je v programu reprezentován seznamem spojů, portů a modulů.

```
#Definice vstupních portů
INPUT(G1gat)
INPUT(G2gat)
INPUT(G3gat)
INPUT(G6gat)
INPUT(G7gat)

#Definice výstupních portů
OUTPUT(G22gat)
OUTPUT(G23gat)

#Definice modulů a spojů
G10gat = nand(G1gat, G3gat)
G11gat = nand(G3gat, G6gat)
G16gat = nand(G2gat, G11gat)
G19gat = nand(G11gat, G7gat)
G22gat = nand(G10gat, G16gat)
G23gat = nand(G16gat, G19gat)
```

Obrázek 17 - Příklad netlistu ve formátu BENCH.

4.2.3. Problémy řešené při vývoji

Vzhledem k tomu, že program SPAR byl vyvinut pro systém UNIX a mým úkolem byl vývoj programu pro systém Windows (později i pro ostatní operační systémy), musel jsem ze zdrojových textů odstranit konfliktní kód, jako například volání systémových funkcí. Po odstranění konfliktního kódu jsem přepsal prototypy veškerých funkcí, protože byly napsány podle staré normy. Tato norma definovala syntaxi pro zápis prototypů funkcí a to následovně :

```
návratová_hodnota název_funkce();
deklarace_parametrů;
{
      „tělo funkce“
}
```

Tyto prototypy jsem odstranil, protože zabraňovaly typové kontrole při předávání parametrů.

Další problém se vyskytl po úspěšném překladu. Tento problém se týkal tzv. memory leaků, což je vlastně neuvolněná paměť alokovaná během vykonávání programu. Díky softwaru Rational Purify Plus se mi podařilo odstranit většinu chyb spojených se správou paměti. Bohužel se mi nepodařilo všechny tyto chyby odstranit, protože některé se projeví pouze při generování velkého schématu,

zhruba o velikosti 150 modulů. Takže program je omezen na generování menších schémat o maximální velikosti 100 modulů.

4.2.4. Použití programu

Můj program SPAR(přiložený jako příloha na CD) je konzolová aplikace spustitelná pod operačním systémem Windows. Jeho jediným parametrem je název vstupního netlist souboru, na jehož základě bude vygenerován výstupní soubor se schématem, který bude mít stejný název jako vstupní soubor a příponu PS. Výstupní soubor je ve formátu Postskript a je zobrazitelný např. programem GSView.

Kapitola 5.
Závěr a budoucí práce

5.1. Závěr

V této práci představuji soubor algoritmů s jejichž pomocí lze generovat obecné schémata logických obvodů z obvodového netlistu a ukazuje jakým způsobem jsou užitečné pro obvodové návrháře svou identifikací funkce a trasovatelností. Důraz je kladen na manipulaci s volným prostorem tak, aby splňoval estetické požadavky, více než při použití pevně daných sloupců a řádků. Úspěchy lze shrnout následujícím způsobem :

- Vytvářet schémata z jednoduchého netlistu
- Produkovat schémata s funkčně uspořádanými moduly
- Vytvářet schémata, v kterých lze vystopovat jednotlivé spoje
- Využívat prostor účinným a flexibilním způsobem

Výsledky v této práci ukazují, že algoritmický přístup k automatickému generování schémat je vhodný a přináší dobré výsledky zejména pro menší schémata. Moje zkušenosti s vývojem programu SPAR ukazují, že generování schémat není triviální problém, a aby přinesl dobré výsledky, vyžaduje jednoúčelové algoritmy a datové struktury. Program je zatím testovací platformou pro popsané algoritmy generování a důraz není kladen na generování velkých schémat, které budou vyžadovat ještě drobné úpravy.

Nejslabším místem tohoto systému je to, že schéma rozdělování není efektivně navrženo pro objevování funkčně identifikovatelných závislostí, tam kde je jejich struktura ukryta hluboko ve schématu. Kombinace rozdělení a rozmístění je velice efektivní pro malá schémata, ale není efektivní při objevování kombinovaných nebo hluboce zakomponovaných struktur. Je několik cest jak zlepšit tyto slabá místa, nejvýznamnějším je vývoj nového schématu rozdělení.

5.2. Budoucí práce

Je mnoho oblastí, kde může být systém SPAR vylepšen. Čtyři nejdůležitější oblasti jsou v použitých metodách rozdělení, rozšíření na inkrementální algoritmy, odstranění chyb spojených se správou paměti a kompletní přechod na objektové programování. Těchto zlepšeních bych chtěl dosáhnout ve své navazující diplomové práci, kterou budu věnovat rozšíření systému SPAR.

5.2.1. Zlepšení rozdělení

Zmíněné metody rozdělování nejsou moc efektivní při manipulaci se sdruženými schématy. Nové metody pro rozdělení schématu by měly být schopny objevit hluboce skrytou hierarchii a rozšířit tak třídu problémů, na které může být SPAR nasazen. To doprovází schopnost automaticky vybrat nejlepší rozdělovací schéma, pokud jsou počáteční výsledky nedostačující.

5.2.2. Inkrementální rozmístění

Zmíněné techniky rozmístění a propojování by měly být rozšířeny, a to tak, že sloučení virtuálních stránek modulů bude řízeno poskytovanými propojovacími informacemi. Tyto propojovací informace mohou být vodítkem pro určení, jak budou moduly nejlépe přidány do rostoucího schématu. To je použitelné ve dvou případech : na nízké úrovni, podobně jako klopný obvod, a na širší úrovni při roz-

mísťování dvou nezávislých struktur. Klopné obvody by měly být propojovány speciálně a komplexnější struktury by měly být vytvářeny odděleně tak, že jejich nezávislé funkce budou identifikovatelné.

5.2.3. Správa paměti

Systém SPAR ještě stále obsahuje chyby vzniklé chybným uvolňování alokované paměti, které by jsem chtěl odstranit v rámci diplomové práce. Jednou cestou jak zmenšit pravděpodobnost výskytu chybného uvolňování alokované paměti je kompletní přechod na objektový model, který však může být doprovázen zvýšením výpočetní složitosti celého systému. Řešení těchto problému bude náplní následné diplomové práce.

Obsah přiloženého CD

Na přiloženém CD jsou umístěny následující adresáře :

- Kořenový adresář – obsahuje soubor readme.txt popisující mapu CD.
- \bin – obsahuje spustitelný soubor testovacího programu.
- \documentation – obsahuje tuto bakalářskou práci ve formátu PDF.
- \sources – obsahuje kompletní zdrojové soubory.

Bibliografie

[1] Baer, Charles J. and Ottaway, John R. „Electrical and Electronics Drawing“, kapitola 5, str. 141 -150. McGraw-Hill, fourth edition, 1980. “Flow Diagrams and Logic Diagrams”.

[2] Arya, A., Kumar, A., Swaminathan, V., and Misra, A. „Automatic Generation of Digital System Schematic Diagrams“. 22nd Design Automation Conf., str. 388 - 395, 1985.

[3] Stok, L. and Koster, G.J.P. „From Network to Artwork“. 26th Design Automation Conf., str. 686 - 689, June 1989.

[4] Brennan, R. J. „An Algorithm for Automatic Line Routing on Schematic Diagrams“. 12th Design Automation Conf., str. 324 - 330, June 1975.

[5] Chun, R. K., Chang, K., and McNamee, L. P. „VISION: VHDL Induced Schematic Imaging On Net-lists“. 24th Design Automation Conf., str. 436 - 442, 1987.

[6] Ahlstrom, M. L., Hadden, G. D., and Stroick, G. R. „HAL: A Heuristic Approach to Schematic Generation“. IEEE International Conf. on Comp. Aided Design, str. 83 - 86, 1983.

[7] Frezza, S.T. „SPAR: A Schematic Place and Route System“. Technical Report TR-CE-91-02, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA, April 1991.

[8] May, M., Iwainsky, A., and Mennecke, P. „Placement and Routing for Logic Schematics“. IEEE Transactions on Comp. Aided Design, Vol. 15 No. 3, str. 89 - 101, May 1983.

[9] May, M. „Computer-Generated Multi-Row Schematics“. IEEE Transactions on Comp. Aided Design, Vol. 17 No. 1, str. 25 - 29, January 1985.

[10] Ousterhout, John K. „Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools“. IEEE Transactions on Comp. Aided Design, Vol. CAD-3 No. 1, str. 87 - 100, 1984.

[11] Servít, Jan, Schmidt Jan, Friš Zdeněk „Automatizace konstrukčního návrhu elektronických zařízení“. ČSVTS –FEL – ČVUT, Praha 1982.

[12] Ousterhout, John K. „Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools“. IEEE Transactions on Comp. Aided Design, Vol. CAD-3 No. 1, str. 87 - 100, 1984.