SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

Ing. Ondrej Gallo

Autoreferát dizertačnej práce

APLIKOVANIE PETRIHO SIETÍ NA NÁVRH ASYNCHRÓNNYCH ČÍSLICOVÝCH OBVODOV

na získanie akademickej hodnosti philosophiae doctor, PhD.
v doktorandskom študijnom programe
9.2.9 aplikovaná informatika

Bratislava 2014

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia
na Ústave informatiky a matematiky FEI STU v Bratislave

Predkladateľ: Ing. Ondrej Gallo
ÚIM FEI STU, Ilkovičova 3, 812 19 Bratislava
Ilkovičova 3
812 19 Bratislava

Školiteľ: Prof. RNDr. Gabriel Juhás, PhD.
FEI STU Bratislava

Oponenti: doc. Ing. Pavel Čičák, PhD.
Ústav počítačových systémov a sietí
Fakulta informatiky a informačných technológií
Slovenskej technickej univerzity v Bratislave
Ilkovičova 2
842 16 Bratislava

doc. Ing. Zoltán Balogh, PhD.
Katedra informatiky
Fakulta prírodných vied
Univerzity Konštantína Filozofa v Nitre
Tr. A. Hlinku 1
949 74 Nitra

Autoreferát bol rozoslaný dňa: 30.7.2014

Obhajoba dizertačnej práce sa koná: 26. 08. 2014 o 12h
na Fakulte elektrotechniky a informatiky STU, Ilkovičova 3, 812 19 Bratislava, v miestnosti C502.

Prof. RNDr. Gabriel Juhás, PhD.
Dekan FEI STU v Bratislave

Obsah

Úvod 2
1 Ciele dizertačnej práce 3
2 Dosiahnuté výsledky dizertačnej práce 4
3 Literatúra 9
4 Zoznam publikácií 10
4.1 Publikované výsledky dizertačnej práce a príspevky na konferenciách 10
4.2 Ostatné 10
Summary 11

Úvod

Na návrh sekvenčných obvodov sa ako modelovací nástroj používali konečné stavové automaty [1]. Je to modelovací nástroj založený na zmene stavov. Začne sa z určitého stavu, prečítajú sa príslušné vstupy, následne sa nastavia príslušné výstupy a prejde sa do ďalšieho stavu. Čas hrá dôležitú rolu, pretože práve on mení stav celého systému. Čas je pre potreby modelovania diskretizovaný. Konečné automaty sú vhodné na modelovanie obvodu synchronizovaného periodickým hodinovým signálom (CLK), ktorý určuje časový okamih zmeny stavu systému. Jednotlivé zmeny stavu sú týmto signálom podmienené, to znamená, že sa môžu vykonať paralelne. Dve operácie sú paralelné, ak sa spúšťajú v tom istom cykle hodinového signálu CLK. Oneskorenie cyklu je dané ako najväčšie možné oneskorenie obvodu, v ktorom sa má vykonať naplánovaná operácia. Synchronizácia medzi jednotlivými operáciami je implicitná, t.j. operácia sa začne vždy s prichádzajúcou hranou hodinového signálu a dokončí sa až po príchode ďalšej hrany. Hrany hodinového signálu teda indikujú začiatok a koniec niekoľkých akcii, ktoré nastali súčasne.

Asynchrónne obvody sa dostali do popredia až s nástupom technológie výroby integrovaných obvodov s veľmi vysokou hustotou integrácie, VLSI (Very-large-scale integration). Synchrónne obvody predstavovali problém v procese návrhu pri použití technológie VLSI [2]. Tento problém spočíval v rovnomernej distribúcii taktovacieho signálu po celom čipe integrovaného obvodu. Synchrónny systém je odkázaný na externý hodinový signál, ktorý slúži ako spoločná časová referencia pre všetky systémové komponenty. Rýchlosť, s ktorou integrovaný obvod pracuje, sa líši podľa použitej technológie, zmeny teploty či stability napájacieho napätia. Tieto parametre majú nezanedbateľný vplyv na nežiadúce zväčšenie periódy taktovacieho signálu. V niektorých prípadoch je toto nechcené predĺženie periódy až o 100% väčšie, ako je uvažovaná perióda. Z týchto dôvodov sa musí privádzať hodinový signál interne, ku každému systému zvlášť a regulovať fázu týchto čiastkových hodinových signálov. Na tento proces je potrebný špeciálny obvod vo vnútri čipu, ktorý zabezpečí, aby bol hodinový signál v každom kľúčovom systéme rovnaký (alebo len s veľmi malou relatívnou odchýlkou). Tento prídavný obvod ale zaberá na čipe pomerne veľa miesta (okolo 10%) a spotrebováva veľký výkon (približne 40% z celkového príkonu čipu). Cena takýchto integrovaných obvodov je potom vysoká a obvod energeticky náročný.

Pri asynchrónnych obvodoch už nie je potrebný globálny hodinový signál, pretože správanie takéhoto obvodu je podmienené udalosťami, čiže začatím a dokončením individuálnej operácie (udalosti). Na základe tohto faktu sa extrémne zníži spotreba obvodu [3]. Generovanie hodinového signálu v synchrónnych obvodoch sa vo všeobecnosti realizuje CMOS obvodom, ktorý spotrebováva, alebo lepšie povedané, nespotrebováva skoro žiaden výkon, pokiaľ je v tzv. IDLE režime (neaktívny režim). Nízkopríkonové synchrónne obvody vyžadujú, aby sa hodinový signál synchronizujúci jednotlivé podsystémy občas vypol, keď tieto podsystémy nie sú potrebné. Ale napájanie generátora hodinového signálu musí byť neustále aktívne. Výsledok je, že výkon sa spotrebováva aj počas IDLE režimu. Veľkou výhodou asynchrónneho obvodu je, že sa prirodzene dostane do IDLE režimu, pokiaľ nie je potrebná žiadna akcia alebo nie je žiadna komunikácia na zbernici. A preto je jeho spotreba oveľa nižšia.

Korektné správanie asynchrónneho obvodu závisí nielen na jeho štruktúre, ale aj na správaní sa jednotlivých hradiel v čase a vzájomnej interakcii. Obvod sa môže modelovať ako množina procesov (hradiel), ktoré komunikujú cez kanály (vodiče) a menia stavy systému, ktorý je reprezentovaný množinou logických premenných (signálov). Hradlo je aktivované, keď je jeho výstupná hodnota odlišná od hodnoty jeho logickej funkcie. Aktivované hradlo môže zmenou hodnoty vyprodukovať prechod (udalosť) na výstupný signál.

V súčasnosti je návrh asynchrónnych obvodov stále zložitý proces, zložitejší ako návrh synchrónnych obvodov. Je to spôsobené absenciou CAD riešení, ktoré by pomohli návrhárovi s touto neľahkou úlohou. Samozrejme, existujú niektoré softvérové riešenia, ktoré by vedeli poskytnúť potrebnú pomoc, no stále v nedostatočnej miere. Môžeme spomenúť niektoré, napr. TANGRAM alebo nástroj Petrify [4]. Bohužial, tieto nástroje ani zďaleka nedosahujú kvalitu komerčných CAD softvérov. Ďalšou prekážkou pre návrhára je potreba jeho preškolenia, pretože návrh asynchrónnych obvodov predstavuje odlišný prístup. Metódy, ktoré sa používajú sú príliš odlišné od metód používaných pre synchrónny návrh, v ktorom prevláda používanie HDL jazykov (VHDL, Verilog). Z týchto informácií by sa zdalo, že vývoj asynchrónnych obvodov sa nebude rozvíjať, alebo na jeho presadenie je potrebný dlhý čas. Nie je tomu celkom tak. Existujú riešenia, ktoré vedia využiť komerčné CAD programy a pomerne rýchlo navrhnúť asynchrónny obvod. Jeden prístup využíva pri návrhu tzv. IP bloky [5]. Ďalší prístup dokáže skombinovať návrhové postupy, ktoré využívajú Petriho siete s postupmi využívajúce HDL jazyky. U synchrónneho návrhu sa najprv definuje špecifikácia správania v HDL jazyku, z neho sa syntézou vytvorí realizácia, ktorá sa následne odsimuluje pomocou tzv. TestBench-ov. Po následnej optimalizácii a úprav sa na záver otestuje pomocou reálnej časovej simulácie a spustí sa proces optimálneho rozmiestnenia komponentov na čipe (Place&Route). Všetky tieto kroky sú v súčasných CAD softvéroch zvládnuté na veľmi dobrej úrovni. Vývojový postup využívajúci Petriho siete môžeme relatívne jednoducho skombinovať zo zaužívaným spôsobom a to tak, že na všetky kroky okrem syntézy použijeme tradičný CAD softvér. Návrhár teda pohodlne vytvorí špecifikáciu v HDL jazyku, ktorú jednoducho, za pomoci externých programov, pretransformuje do Petriho siete. Tá potom vstupuje do už existujúcich programov na syntézu asynchrónnych obvodov. Ich výstupom je výsledný obvod, ktorý môžeme následne odsimulovať a spustiť Place&Route proces opäť v zaužívanom prostredí. Nie je teda potrebné drahé preškolenie návrhárov na nový CAD program. Takýto postup bol popísaný napr. v publikáciách [6], [7], kde sa na špecifikáciu činnosti obvodu využil jazyk Verilog a na syntézu logickej funkcii z Petriho siete zase akademický program Petrify.

Petriho siete sú modelovací nástroj, ktorý je zvlášť vhodný na návrh asynchrónnych číslicových obvodov a to kvôli tomu, že obsahujú formalizmus na popísanie paralelných udalostí. Paralelizmus sa v tomto prípade chápe ako vykonanie udalostí nezávisle na sebe. Dizertačná práca popisuje celý postup pri návrhu obvodov, od špecifikácia až po samotnú syntézu obvodu. Práca sa nezaoberá metódami na optimálne rozsadenie komponentov na čipe, tzv. Place&Route.

1 Ciele dizertačnej práce

Prvým cieľom dizertačnej práce bolo analyzovať možnosti využitia Petriho sietí v procese návrhu asynchrónnych číslicových obvodov. Keďže sa nejedná o štandardný spôsob návrhu takýchto obvodov, bolo potrebné zvážiť, kedy je vhodné použiť Petriho siete a kedy je zase výhodnejšie využiť štandardné spôsoby. Aplikovanie Petriho sietí v tomto procese vyžaduje mierne odlišný spôsob v špecifikácii správania. Práve tento bod bol hlavným cieľom dizertačnej práce. Problém by sa dal zhrnúť do týchto bodov:

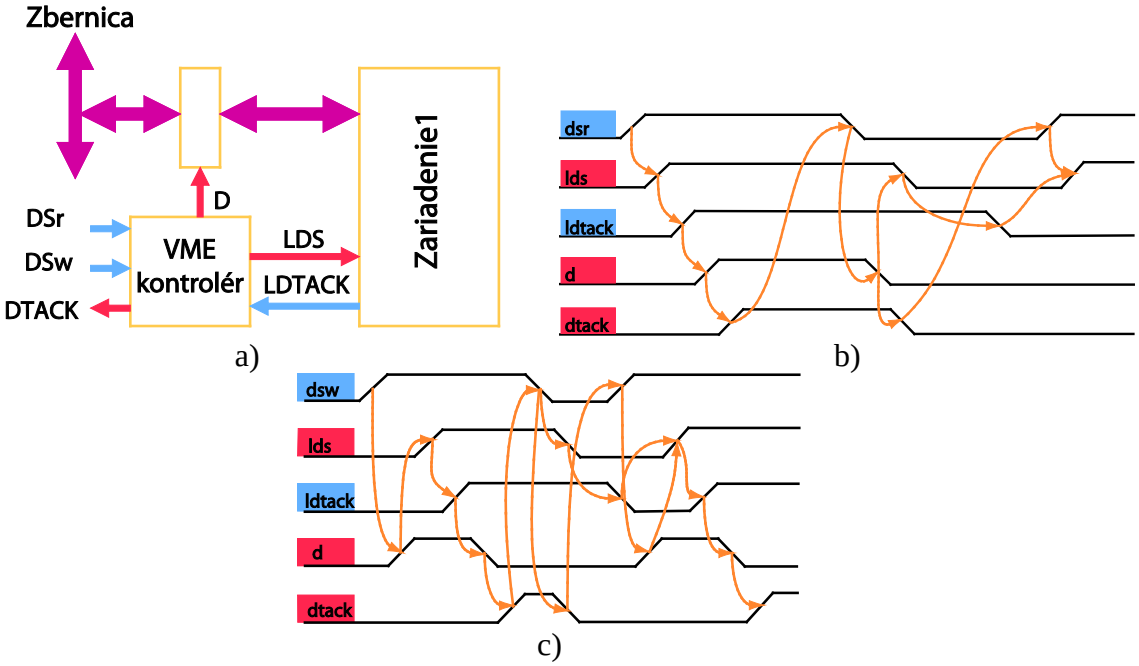
1. Zvoliť vhodnú formu špecifikácie správania obvodu pre potreby syntézy Petriho siete.
2. Vytvoriť algoritmus na syntézu Petriho siete, ktorá generuje požadované správanie.
3. Minimalizovať Petriho sieť a overiť správnosť generovaného modelu.

Možností, ako vytvoriť špecifikáciu správania obvodu pre potreby nášho prístupu je viacero. Keďže sme chceli na získanie modelu správania využiť tzv. syntézu Petriho siete z nesekvenčného správania, ako najrozumnejšia voľba špecifikácie bolo využitie časových diagramov. V časovom diagrame sú zachytené informácie nie len o časových závislostiach,

ale hlavne o kauzalite medzi jednotlivými signálmi. Práve tieto kauzality sú vstupným parametrom pre tvorbu grafu signálov (STG), ktorý je východiskovým modelom pre tvorbu logickej funkcie. Najzávažnejším problém, ktorý v tomto procese vznikol, bolo zabezpečenie cyklického správania v STG. Časové diagramy totižto neznázorňujú časové správanie s nekonečnou dĺžkou, ale len krátky periodický úsek, ktorý sa môže opakovať nekonečne veľa krát. Takýchto diagramov môžeme mať veľký počet a nemusí byť jednoduché vytvoriť z nich jeden model v podobe STG.

2 Dosiahnuté výsledky dizertačnej práce

Vytvorenie STG z časových diagramov tvorí ťažisko práce. Na vytvorenie STG sme využili metodológiu syntézy Petriho sietí z nesekvenčných scenárov popísaných v publikáciách [8] a [9]. Je v nich popísaný princíp generovania Petriho siete, ako aj relevantné teoretické vedomosti. Vstupom pre spomínanú syntézu Petriho sietí bolo tzv. označkované čiastočné usporiadanie (LPO). Pomocou neho dokážeme vyjadriť, veľmi presne a jednoducho, nesekvenčné správanie ľubovoľného systému. Nesekvenčné správanie je pre číslicové obvody prirodzené. Takto dokážeme zachytiť nie len udalosti vykonávajúce sa za sebou, ale aj udalosti, ktoré môžu nastať nezávisle na sebe. Ako príklad spomeňme správanie VME kontroléra (obr. 2.1), ktorý riadi komunikáciu medzi jednotlivými zariadeniami (čítanie/zápis do pamäte).



Obr. 2.1 a) Bloková schéma VME kontroléra b) časový diagram pre cyklus čítania c) časový diagram pre cyklus zápisu

Operácie, týkajúce sa požiadavky resp. odpovede na čítanie alebo zápis, sa vykonávajú v určitom poradí (sekvenčne). Ale napr. požiadavka na začiatok komunikácie medzi zariadením 1 (LDS) a druhým zariadením (DTACK) už nemusí byť sekvenčná udalosť (obr. 2.1b). Podľa špecifikácie by signály mali zmeniť stav naraz, ale kvôli oneskoreniam sa zmenia v rôznom čase. Toto správanie je teda na sebe nezávislé od momentu, keď požiadavka na čítanie (DSR) nadobudne logickú 0. Štruktúra, ktorá je vhodná pre definovanie takéhoto správania sa volá označkované čiastočné usporiadanie (*Labeled Paprtial Order*), LPO.

Definícia 2.1: Čiastočné usporiadanie $po(V, <)$ je binárna relácia $<$ na množine prvkov V, ktorá vyhovuje nasledujúcim podmienkam:

4. Ireflexivita: $\forall v \in V, \neg(v < v)$;
5. Asymetria: $\forall v_1, v_2 \in V, (v_1 < v_2) \Rightarrow \neg(v_2 < v_1)$;
6. Transitivita: $\forall v_1, v_2, v_3 \in V, (v_1 < v_2) \wedge (v_2 < v_3) \Rightarrow (v_1 < v_3)$.

Definícia 2.2: Označkované čiastočné usporiadanie (LPO) je trojica $lpo = (V, <, l)$, kde $(V, <)$ je čiastočné usporiadanie po a $l: V \rightarrow T$ je značkovacia funkcia zo značkami z množiny T.

Syntéza Petriho siete zo zadaného nesekvenčného správania popísaného označkovaným čiastočným usporiadaním (LPO) pozostáva z týchto krokov:

a) **Prvý krok** - vytvorenie tzv. nultého rozšírenia zadaného LPO o začiatočný a koncový vrchol.
b) **Druhý krok** – vytvorenie sústavy lineárnych rovníc reprezentujúce tzv. regióny.
c) **Tretí krok** – pretransformovanie problému riešenia sústavy lineárnych rovníc na sústavu lineárnych nerovníc.
d) **Štvrtý krok** – nájsť celočíselné nezáporné riešenie sústavy lineárnych nerovníc.
e) **Piaty krok** – vygenerovanie Petriho siete. Riešenie sústavy sa reprezentuje ako násobnosť vstupných a výstupných hrán spájajúcich konkrétne miesta a prechody, a takisto aj počiatočné značkovanie Petriho siete.

Modifikácia pôvodnej verzie syntézy, tak aby výsledný STG bola cyklická Petriho sieť, spočíva v úprave prvého a druhého kroku.

Prvý krok – nulté rozšírenie LPO

V pôvodnej syntéze z [20] sme potrebovali rozšíriť množinu vrcholov V o množinu $\{v_0, v_{max}\}$, kde v_o je začiatočný vrchol a v_{max} zase koncový vrchol v LPO. Výsledná množina vrcholov V^* potom bola definovaná ako zjednotenie týchto dvoch množín:

$$V^* = (V \cup \{v_0, v_{max}\}).$$

Následne bolo ešte potrebné dodefinovať aj vzťah nanovo pridaných vrcholov $\{v_0, v_{max}\}$ k pôvodnému LPO, tj. binárnu reláciu $<^*$, ktorá je reflexívna a tranzitívna. V novom prípade budeme rozširovať množinu V len o koncové vrcholy v_{max}. Teda začiatočné v_0 už nie. Nová definícia nultého rozšírenia LPO znie:

Definícia 2.3 (**nulté rozšírenie**). Pre množinu označkovaných čiastočných usporiadaní L označujeme $W_L = \bigcup_{(V,<,l)\in L} V$, $E_L = \bigcup_{(V,<,l)\in L} <$ a $l_L = \bigcup_{(V,<,l)\in L} l$. Nulté rozšírenie označkovaného čiastočného usporiadania $\text{lpo}^* = (V^*, <^*, l^*)$ pôvodného lpo je definované:

1) $V^* = \left(V \cup \left\{v_{max}^{lpo}\right\}\right)$ kde $v_{max}^{lpo} \notin V$,
2) $<^* = < \cup \left(V \times \left\{v_{max}^{lpo}\right\}\right)$,
3) $l^*\left(v_{max}^{lpo}\right) \in l(V), l^*\left(v_{max}^{lpo}\right) = l^*\left(v_1^{lpo}\right)$.

v_1^{lpo} je začiatočný vrchol v príslušnom LPO a v_{max}^{lpo} pridaný koncový vrchol v príslušnom LPO. Nech $\text{lpo}^* = (V^*, <^*, l^*)$ je nulté rozšírenie každého $\text{lpo} \in L$ takého, že platí:

4) Pre každé dva označkované čiastočné usporiadania $(V, <, l)$, $(V', <', l') \in L$ platí: $l\left(v_1^{lpo}\right) = l'\left(v_1^{lpo'}\right)$. Alebo slovne vyjadrené, každé LPO musí mať začiatočný vrchol v_1 rovnako pomenovaný.
5) Pre každé dva rozdielne označkované čiastočné usporiadania $(V, <, l)$, $(V', <', l') \in L$ platí: $l\left(v_{max}^{lpo}\right) = l\left(v_1^{lpo}\right) \wedge l'\left(v_{max}^{lpo'}\right) = l'\left(v_1^{lpo'}\right)$. Slovne vyjadrené, každé LPO musí mať koncový vrchol v_{max} s rovnakým názvom ako je začiatočný vrchol v_1.

Potom nazveme množinu $L^* = \{\text{lpo}^* | \text{lpo} \in L\}$ nultým rozšírením množiny L.

Podobne označujeme aj množinu vrcholov rozšírenej množiny L^* $W_L^* = W_{L^*}$, množinu relácii medzi všetkými, teda aj pridanými vrcholmi $E_L^* = E_{L^*}$ a nakoniec aj množinu označkovania všetkých vrcholov $l_L^* = l_{L^*}$. Tá je totožná s pôvodnou množinou označení l_L.

Druhý krok – sústava lineárnych rovníc

Tento krok sa od pôvodnej syntézy líši zásadným spôsobom. Rozdielnosť je úzko naviazaná na prvý krok, kedy bol pozmenený zmysel nultého rozšírenia LPO. V nasledujúcom texte budú vysvetlené všetky tieto zmeny. Tak ako aj v originálnej syntéze aj tu potrebujeme vyrátať regióny, z ktorých je možné zostaviť STG. Regióny vieme vyrátať z lineárnej sústavy rovníc $A_L \cdot x_r = 0$, kde vektor x_r reprezentuje premenné a matica A_L zase koeficienty sústavy rovníc. Veľkosť vektora x_r je totožná s počtom hrán v LPO vytvorenom v kroku 1.

$$x_r = (x_1, x_2, \cdots, x_n),\ n = |E_L^*|.$$

Je vhodné uviesť, že riešenie sústavy rovníc (vektor x_r) sa nachádza v obore reálnych čísel R, ale pre syntézu Petriho siete sa potrebujeme pohybovať iba v prirodzených číslach vrátane 0. Dôvodom je fakt, že značkovanie ako aj váhy hrán v Petriho sieti sú podľa definície tiež prirodzené čísla. Tomu musíme prispôsobiť aj metódu na riešenie sústavy rovníc. Hodnoty riadkov A_L sú definované aj teraz rovnakým spôsobom ako v [8], vstupnou, výstupnou a iniciálnou tokovou funkciou. Rozdiel je ale v iniciálnej tokovej funkcii. Pre každý pár vrcholov s rovnakým názvom sa priradia hodnoty vstupnej a výstupnej tokovej funkcie. To sa uskutoční identicky s pôvodnou syntézou. Presný postup vytvorenia tejto časti matice A_L je rozobratý v [8]. Ako už bolo spomenuté, pre potreby syntézy STG je potrebné zmeniť predpis pre iniciálnu tokovú funkciu, pretože v nultom rozšírení sa nevložili 2 nové vrcholy pre jedno LPO ale iba jeden a pozmenil sa význam začiatočného v_0, ktorý teraz nazývame v_1. Preto je potrebné vytvoriť rovnicu pre začiatočný v_1 ale aj koncový v_{max} vrchol. Zadefinujme si presnú formuláciu predpisu pre všetky tri tokové funkcie + jednu navyše, vznik ktorej súvisí s modifikáciou nultého rozšírenia.

Vstupná toková funkcia

$$a_{m,j}^t = \begin{cases} 1 & \text{ak } e_j \text{ je vstupná hrana vrcholu } v_m^t, \\ -1 & \text{ak } e_j \text{ je vstupná hrana vrcholu } v_{m+1}^t, \\ 0 & \text{inak} \end{cases}$$

$$a_m^t = \left(a_{m,1}^t, \cdots, a_{m,n}^t\right)$$

a_m^t je m-tý riadok matice A_L. Každý riadok reprezentuje dva rôzne vrcholy v_m^t a v_{m+1}^t s rovnakým označením $t \in T$. Každý riadok má n stĺpcov, ktoré reprezentujú hrany e_j vstupujúce do už spomínaného vrcholu v_m^t a v_{m+1}^t. Jednotlivé hrany nazývame regióny r. Ak hrana smeruje do v_m^t, tak je hodnota $a_{m,j}^t = 1$ ak vstupuje do v_{m+1}^t, tak $a_{m,j}^t = -1$

a nakoniec, ak nevstupuje do týchto vrcholov žiadna hrana, ohodnotíme $a_{m,j}^t = 0$. Platí, že $a_m^t \cdot x_r = 0$ ak $In_{lpo}(v_m^t, r) = In(v_{m+1}^t, r)$ pre dvojicu LPO lpo $= (V, <, l)$ a lpo$' = (V', <', l')$ s $v_m^t \in V$ a $v_{m+1}^t \in V'$. Počet riadkov *m* závisí od veľkosti množiny W_t nasledovne $1 \leq m \leq |W_t| - 1$, kde $W_t = \{v \in W_L^* | \, l_L^*(v) = t\}$ je množina vrcholov, ktoré majú označenie *t*. Podobne môžeme popísať aj výstupnú tokovú funkciu.

Výstupná toková funkcia

$$b_{m,j}^t = \begin{cases} 1 & \text{ak e}_\text{j} \text{ je výstupná hrana vrcholu } v_m^t, \\ -1 & \text{ak e}_\text{j} \text{ je výstupná hrana vrcholu } v_{m+1}^t, \\ 0 & \text{inak} \end{cases}$$

$$b_m^t = \left(b_{m,1}^t, \cdots, b_{m,n}^t\right)$$

b_m^t je *m*-tý riadok matice A_L. Každý riadok reprezentuje dva rôzne vrcholy v_m^t a v_{m+1}^t s rovnakým označením $t \in T$. Každý riadok má *n* stĺpcov, ktoré reprezentujú hrany e_j vystupujúce zo spomínaného vrcholu v_m^t a v_{m+1}^t. Ak hrana smeruje z v_m^t, tak je hodnota $b_{m,j}^t = 1$ ak vystupuje z v_{m+1}^t, tak $b_{m,j}^t = -1$ a nakoniec, ak nevystupuje z týchto vrcholov žiadna hrana, ohodnotíme $b_{m,j}^t = 0$. Platí, že $b_m^t \cdot x_r = 0$ ak $Out_{lpo}(v_m^t, r) = Out(v_{m+1}^t, r)$ pre dvojicu LPO lpo $= (V, <, l)$ a lpo$' = (V', <', l')$ s $v_m^t \in V$ a $v_{m+1}^t \in V'$. Počet riadkov *m* závisí tiež od veľkosti množiny, $1 \leq m \leq |W_t| - 1$, ktorá je zadefinovaná rovnako ako pri definovaní vstupnej tokovej funkcie.

Vstupná iniciálna toková funkcia

Táto funkcia nebola definovaná v pôvodnej definícii iniciálnej funkcie. Je to spôsobené tým, že pridaný koncový vrchol plní inú funkciu a teda má rovnaký názov ako začiatočný. Pôvodne mal koncový vrchol vždy iný názov od ostatných označení vrcholov.

$$c_{m,j} = \begin{cases} 1 & \text{ak e}_\text{j} \text{ je vstupná hrana vrcholu } v_{max}^{lpo_m}, \\ -1 & \text{ak e}_\text{j} \text{ je vstupná hrana vrcholu } v_{max}^{lpo_{m+1}}, \\ 0 & \text{inak} \end{cases}$$

$$c_m = \left(c_{m,1}, \cdots, c_{m,n}\right)$$

c_m je *m*-tý riadok matice A_L. Každý riadok reprezentuje dva rôzne koncové vrcholy $v_{max}^{lpo_m}$ a $v_{max}^{lpo_{m+1}}$ (z rozdielnych LPO) ale s rovnakým označením $t \in T$. Každý riadok má *n* stĺpcov, ktoré reprezentujú hrany e_j vstupujúce do spomínaného vrcholu $v_{max}^{lpo_m}$ a $v_{max}^{lpo_{m+1}}$. Ak hrana smeruje do $v_{max}^{lpo_m}$, tak je hodnota $c_{m,j} = 1$ ak vstupuje do $v_{max}^{lpo_{m+1}}$, tak $c_{m,j} = -1$ a nakoniec, ohodnotíme $c_{m,j} = 0$, ak nevstupuje do týchto vrcholov žiadna hrana. Platí, že $c_m \cdot x_r = 0$ ak $In_{lpo_m}\left(v_{max}^{lpo_m}, r\right) = In_{lpo_{m+1}}\left(v_{max}^{lpo_{m+1}}, r\right)$. Počet riadkov *m* závisí od veľkosti množiny *L* nasledovne $1 \leq m \leq |L| - 1$, kde $L = \{lpo_1, lpo_2, \cdots\}$ je množina všetkých LPO.

Výstupná iniciálna toková funkcia

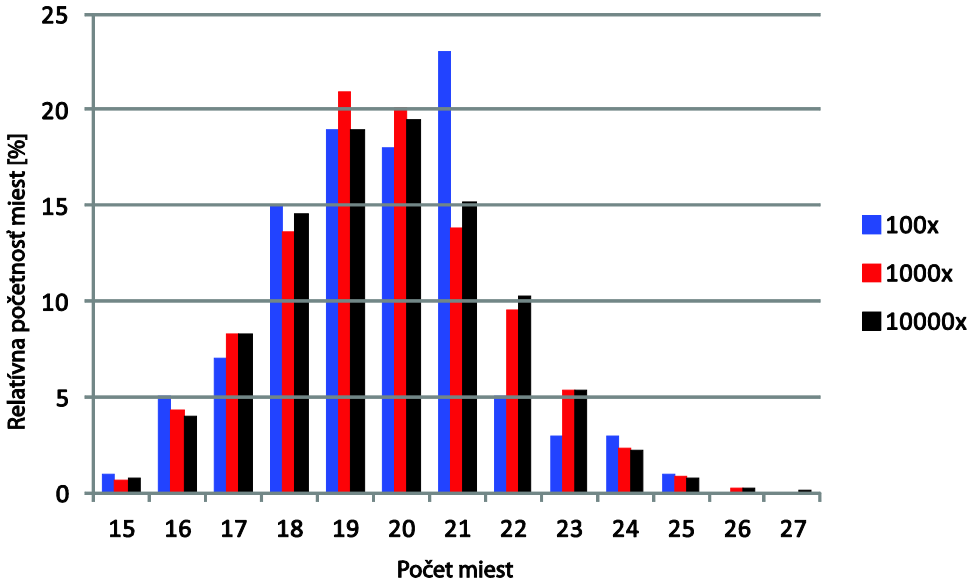
V tejto funkcii musela byť tiež vykonaná zmena (i keď iba malá), pretože počiatočný vrchol nebol pridávaný v nultom rozšírení. Začiatočný vrchol je vlastne tvorený prvým vrcholom v LPO.

$$d_{m,j} = \begin{cases} 1 & \text{ak e}_\text{j} \text{ je výstupná hrana vrcholu } v_1^{lpo_m}, \\ -1 & \text{ak e}_\text{j} \text{ je výstupná hrana vrcholu } v_1^{lpo_{m+1}}, \\ 0 & \text{inak} \end{cases}$$

$d_m = (d_{m,1}, \cdots, d_{m,n})$

d_m je m-tý riadok matice A_L. Každý riadok reprezentuje dva rôzne začiatočné vrcholy $v_1^{lpo_m}$ a $v_1^{lpo_{m+1}}$ (z rozdielnych LPO) ale s rovnakým označením $t \in T$. Každý riadok má n stĺpcov, ktoré reprezentujú hrany e_j vystupujúce zo spomínaných vrcholov $v_1^{lpo_m}$ a $v_1^{lpo_{m+1}}$. Ak hrana smeruje z $v_1^{lpo_m}$, tak je hodnota $d_{m,j} = 1$ ak vystupuje z $v_1^{lpo_{m+1}}$, tak $d_{m,j} = -1$ a nakoniec, ohodnotíme $c_{m,j} = 0$, ak nevystupuje z týchto vrcholov žiadna hrana. Platí, že $d_m \cdot x_r = 0$ ak $Out_{lpo_m}(v_1^{lpo_m}, r) = Out_{lpo_{m+1}}(v_1^{lpo_{m+1}}, r)$. Počet riadkov m závisí od veľkosti množiny L nasledovne $1 \leq m \leq |L| - 1$, kde $L = \{lpo_1, lpo_2, \cdots\}$ je množina všetkých LPO.

Ukážme si teraz na konkrétnom príklade aplikovanie modifikovanej syntézy na vytvorenie STG. Aby bol výsledný STG použiteľný v ďalšom procese návrhu obvodu, je potrebné zredukovať jeho veľkosť. Každá syntéza, či už Petriho siete alebo STG, vytvorí sieť s veľkým počtom miest. Väčšina týchto miest sú redundantné. Preto bolo potrebné navrhnúť spôsob minimalizácie počtu miest výslednej siete. Tento bod tvorí ďalší krok, ktorý doplňuje pôvodnú syntézu Petriho sietí. Princíp redukcie je založený na náhodnom odstraňovaní miest. Následne sa kontroluje, či dané miesto porušilo spustiteľnosť pôvodného neredukovaného STG. Ak je spustiteľnosť porušená, miesto sa vráti a náhodne sa vyberie ďalšie. Tento proces sa opakuje a skončí vtedy, ak už neexistuje žiadne miesto vhodné na odstránenie. Z charakteru tohto heuristického algoritmu vyplýva, že výsledný STG sa už nebude môcť ďalej redukovať. No na druhej strane nezaručuje redukciu miest na najmenšiu možnú mieru. Teda výsledné STG nemusí byť minimálne. Stupeň redukcie STG silne závisí od poradia odstraňovania miest, a keďže je toto poradie určené náhodným generátorom, každým spustením algoritmu môžeme dostať rôzny výsledok. Na overenie algoritmu bola použitá testovacia sekvencia, ktorá zabezpečila opakované spustenie redukcie pre ten istý STG. Počet opakovaných spustení sme stanovili na 100, 1000 a 10000. Pri každom spustení redukcie bol zaznamenávaný počet miest v STG. Výsledky testov sú zobrazené na obr. 2.2. Pri 100-násobnom opakovaní algoritmu boli spozorované STG s najmenej 15 a najviac s 25 miestami. Pri zväčšení počtu opakovaní napr. na 10000 sme zaznamenali STG aj s väčším počtom miest (27 miest). Je to pochopiteľné, pretože algoritmus mohol vyskúšať aj iné poradie pri odstraňovaní miest, ktoré pri menšom počte pokusov už nemohol uskutočniť. Ak by sme chceli zmenšiť STG na najmenšiu možnú mieru, museli by sme redukciu spustiť viac krát, čo pri STG s veľkým počtom miest nie je možné kvôli výpočtovej náročnosti ekvivalencie. Z histogramu je vidieť, že početnosť výskytu najmenšieho počtu miest (15) je malá, pod 1%. Napríklad, ak sme opakovali redukciu 10000-krát dostali sme 10000 STG a z toho len 73 mali počet miest 15 (minimálny počet). Pre tento prípad je relatívna početnosť iba 0,73%. Najviac STG je s počtom miest okolo 20, t.j. približne v strede histogramu. Z toho dôvodu nám algoritmus s najväčšou pravdepodobnosťou vráti STG, ktoré nebude minimálne, ale ani maximálne.



Obr.2.2 Histogram pre výsledný počet výskytov miest v STG VME kontroléra z obr. 2.1

Tento konkrétny histogram bol vytvorený pre modelový príklade VME kontroléra s operáciou read/write z obr. 2.1. Po vložení neredukovaného STG so 450 miestami algoritmus dokázal minimalizovať sieť až o 96,6% v najlepšom prípade, a o 94,4% v najhoršom prípade. V prepočte na absolútne čísla sme dokázali zredukovať sieť maximálne o 435 a minimálne o 425 miest.

Algoritmus bol podrobený skúškam na viacerých príkladoch a vykazoval vynikajúce výsledky v redukcii miest. Pre príklad čítania zo zariadenia prostredníctvom VME kontroléra (obr. 2.1b) mal STG vygenerovaný syntézou 99 miest a po redukcii iba 11 miest (88,9%) a navyše bol úplne zhodný s STG vytvoreným ručne z časového diagramu z článku [10]. Tento príklad je zaujímavý aj preto, lebo pri redukovaní miest ich výsledný počet nezávisel od poradia odstraňovania. T.j. vždy sme dostali ten istý STG s 11 miestami. Úspešnosť redukcie miest teda závisí od poradia výberu miesta, t.j. od samotnej štruktúry STG.

3 Literatúra

1. **Myers, C. J.** s.l. : JohnWiley&Sons, July 2004. Asynchronous Circuit Design. 0-471-41543-X.

2. **Davis, A. and Nowick, Steven M.** [ed.] A. Kent and J. G. Williams. *An introduction to asynchronous circuit design.* New York : The Encyclopedia of Computer Science and Technology, February 1998. Vol. 38.

3. **Sparsø, J.** 2006. Asynchronous Circuit Design. A Tutorial. [cit. 2013-03-07]. Dostupné na internete: <http://www.ee.technion.ac.il/courses/048878/book.pdf>.

4. Petrify. *lsi.upc.edu.* [Online] [Cited: July 9, 2013.]

5. **Kondratyev, A. and Lwin, K.** *Design of asynchronous circuits using synchronous CAD tools.* s.l. : Journal IEEE Design & Test, August 2002. Vol. 19, No. 4, pp. 107-117.

6. **Blunno, I. and Lavagno, L.** Deriving Signal Transition Graphs from Behavioral Verilog HDL. 2 [ed.] A. Yakovlev, L. Gomes and L. Lavagno. *Hardware Design and Petri Nets.* Boston : Kluwer Academic Publishers, 2003. 0-7923-7791-5.

7. **Wang, A., Keutzer, K. and Vanbekbergen, P.** *A Design and Validation System for Asynchronous Circuits.* s.l. : 32nd ACM/IEEE Conference on Design Automation Conference (DAC'95), 1995. pp. 725-730.

8. **Lorenz, R., et al.** Application of Concurrency to System Design. *Synthesis of Petri Nets from Finite Partial Languages.* July 2007. pp. 157-166.

9. **Lorenz, R. and Juhás, G.** In Proceedings of the 27th international conference on Applications and Theory of Petri Nets and Other Models of Concurrency (ICATPN'06). *Towards synthesis of petri nets from scenarios.* Berlin : Springer-Verlag, 2006. pp. 302-321.
10. **Cortadella, J., et al.** Application and Theory of Petri Nets 2000. *Hardware and Petri Nets: Application to Asynchronous Circuit Design.* s.l. : pringer Verlag, June 2000. Vol. 1825, pp. 1-15.

4 Zoznam publikácií

4.1 Publikované výsledky dizertačnej práce a príspevky na konferenciách

Gallo, Ondrej - Nečas, Tomáš - Lehocki, Fedor: A Tool for the Synthesis of Asynchronous Speed Independent Circuits.
In: Application of Region Theory (ART) : 1st Workshop. Braga, Portugal, 21.-22.6.2010. - : Springer Verlag, 2010. - ISBN 978-972-8692-54-4. - S. 61-65.

Gallo, Ondrej - Nečas, Tomáš - Juhás, Gabriel: Syntéza asynchrónnych číslicových obvodov pomocou Petriho sietí.
In: SMART S2AI : Workshop SMART systémov a služieb v oblasti aplikovanej informatiky. Bratislava, 18. apríla 2011. - Bratislava : STU v Bratislave FEI, 2011. - ISBN 978-80-227-3513-1. - S. 6-8.

Gallo, Ondrej - Savka, Andrej - Juhás, Gabriel: How to Synthesize Reduced STG from LPOs.
In: Application of Region Theory (ART) : 3rd Workshop Proceedings; Barcelona, Spain, 9 July 2013. - Barcelona : Universitat Politecnica de Catalunya, 2013. - S. 55-61.

4.2 Ostatné

Gallo, Ondrej - Marček, Stanislav: Application of optical sensors in biomedical engineering.
In: Meditech - Proceedings of the ESF Project Conference : Innovative Program of Modern Biomedical Technologies. Project No. SORO/JPD-26/2005. Bratislava, Slovakia, 26.5.2008. - Bratislava : STU v Bratislave, 2008. - ISBN 978-80-227-2881-2. - S. 159-164.

Marček, Stanislav - Gallo, Ondrej: Non-Invasive Patient Monitoring. In: Meditech - Proceedings of the ESF Project Conference : Innovative Program of Modern Biomedical Technologies. Project No. SORO/JPD-26/2005. Bratislava, Slovakia, 26.5.2008. - Bratislava : STU v Bratislave, 2008. - ISBN 978-80-227-2881-2. - S. 153-158.

Lehocki, Fedor - Stuchlíková, Ľubica - Gallo, Ondrej - Kékešiová, Gabriela: e-Learning in Professional Education for Medical High-School Students.
In: Virtual University 2008 : 9th International Conference. Bratislava, Slovak Republic, 11.-12.12.2008. - Bratislava : STU v Bratislave, 2008. - ISBN 978-80-89316-10-6. - CD-Rom.

Gallo, Ondrej - Zajac, Pavol: Simple Power Analysis Demonstration on Arduino Platform. In: EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie. - ISSN 1335-2547. - Roč. 19, mimoriadne č. : konferencia ELOSYS, Trenčín, 15.-18.10.2013 (2013), s. 15-19.

Summary

The main goals of the thesis were:

- to analyze the various styles of asynchronous circuits design, which is currently used.
- to analyze the methods for the synthesis a logic function from the input model defined by Petri nets.
- to design a methodology for the synthesis of Petri nets, which is suitable for the design of asynchronous circuits.
- to evaluate the advantages and disadvantages of using Petri nets in the design of asynchronous circuits.

In the first and second chapter of thesis were compared properties of synchronous and asynchronous circuits. They contained also motivation for using Petri nets in hardware design. The comparison showed different design techniques for asynchronous and synchronous circuits. The third chapter described asynchronous hardware design with various hardware components and quantitative and qualitative analysis. The fourth and fifth chapter interested in theoretical knowledge of Petri nets and Signal Transition Graph (STG). The most important part of the work represents chapter six, which was explained a methodology of STG synthesis from non-sequential behavior. Non-sequential behavior could be express by time diagrams of circuit behavior. From this model was necessary to create a STG, which is the key component for next design steps of asynchronous circuits. For synthesis of STG was used labeled partial order (LPO). The LPO was made direct from the time diagram. The modification of synthesis STG was based on technique which provides a cycle in STG. In the original synthesis from [8], this was not possible. Anyway, this synthesis produced a lot of places in STG. Many of them were redundant and could be removed. The thesis contained also algorithm of place reduction which was based on random removing. Correct removing process was evaluated by compare the behavior with original STG and reduced STG. It was achieved a high place reduction rate (80-90%) by successive random elimination of places. Since the algorithm used stochastic approach for reduction of places, every execution generated STG with different number of places. The reduced STG did not contain the minimal number of places, but the number was still significantly minimized. The hardware design of asynchronous circuit with using STG is suitable just for control circuit (memory controllers). It is not appropriate for complex circuits which contain data paths (data buses). The STG for those circuits should provide too many transitions and places. Nevertheless, this is not a significant problem, because asynchronous circuit design is due to its operating principle (controlled through a special protocol) modular. In principle, its mean that designer made only one logical block (controller) which control data registers. Registers may have a large bus width and they would be designed by other suitable method. In the last seven chapter were analyzed a methods for generating logic function from STG. The resulting logic function was derived as a hazard free. This property is very important especially for asynchronous circuits.