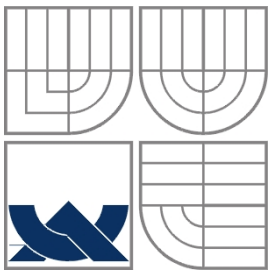
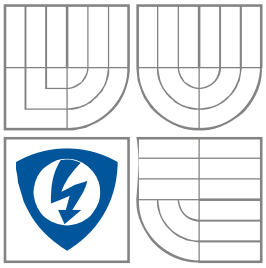


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

ANALÝZA MOŽNOSTÍ SIMULÁCIE A IMPLEMENTÁCIE AUTOSYNCHRÓNNYCH SUBSYSTÉMOV V OBVODOCH VLSI

SIMULATION AND IMPLEMENTATION ANALYSIS OF THE AUTOSYNCHRONOUS SUBSYSTEMS IN VLSI DEVICES

DIZERTAČNÁ PRÁCA
DISSERTATION THESIS

AUTOR PRÁCE — Ing. Michal Kováč
AUTHOR

VEDÚCI PRÁCE — doc. Ing. Jaromír Kolouch, CSc.
SUPERVISOR

BRNO, 2009

Abstrakt

Práca sa zaoberá analýzou riešenia problémov synchrónnych číslicových obvodov. Výsledkom je vytvorenie metodiky pre návrh autosynchrónnych obvodov, definovanie ich časových parametrov na základe simulačných modelov a stanovenie podmienok pre ich správnu funkciu. Pre jednoduchý návrh týchto obvodov bola vytvorená metodika transformácie RTL popisu synchrónneho stavového automatu vo VHDL na autosynchrónny automat pri minimálnych zmenách. Nasleduje porovnanie transformovaných stavových automatov s ich originálnymi synchrónnymi predlohami v parametroch ako sú plocha čipu, prúdová spotreba a časovanie. Na záver práce je uvedené teoretické porovnanie rôznych typov synchronizácií (synchrónny, autosynchrónny, asynchrónny vo fundamentálnom režime, EAIC, Bundled-data, Dual-rail) na jednom príklade stavového automatu pri rovnakých technologických parametroch.

Kľúčové slová

Asynchrónne systémy, autosynchrónny obvod, stavový automat, VHDL, RTL transformácia, kódovanie 1 z N, Grayove kódovanie, bundled-data, dual-rail, detekcia ustáleného stavu, detekcia rozdielnych stavov, generátor hodinového signálu.

Abstract

This thesis focuses on problem-solution analysis of synchronous digital circuits; the results of which are autosynchronous circuit design methodology, timing parameter definitions based on simulation models and constraint settings. The RTL transformation of the synchronous state machine in VHDL language to an autosynchronous state machine was created with minimal modifications for the simple design of these circuits. Following this, a comparison of the transformed state machines with their synchronous originals in parameters such as chip area, current consumption and timing specification domain is introduced. The summation of this thesis displays a theoretical comparison of several types of synchronization (synchronous, autosynchronous, fundamental asynchronous, EAIC, Bundled-data, Dual-rail) which are presented on the single state machine example with the same technology parameters.

Keywords

Asynchronous systems, autosynchronous circuit, state machine, VHDL, RTL transformation, one-hot encoding, Gray encoding, bundled-data, dual-rail, stable state detection, different state detection, clock generator.

Bibliografická citácia

KOVÁČ, M. *Analýza možností simulácie a implementácie autosynchrónnych subsystémov v obvodoch VLSI*. Dizertačná práca. Brno: FEKT VUT v Brne, 2009. 69 strán.

Prehlásenie

Prehlasujem, že svoju dizertačnú prácu na tému Analýza možností simulácie a implementácie autosynchrónnych subsystémov v obvodoch VLSI som vypracoval samostatne pod vedením školiteľa a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej dizertačnej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto dizertačnej práce som neporušil autorské práva tretích osôb, hlavne som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia § 152 trestného zákona č. 140/1961 Sb.

V Brne dňa 31. augusta 2009

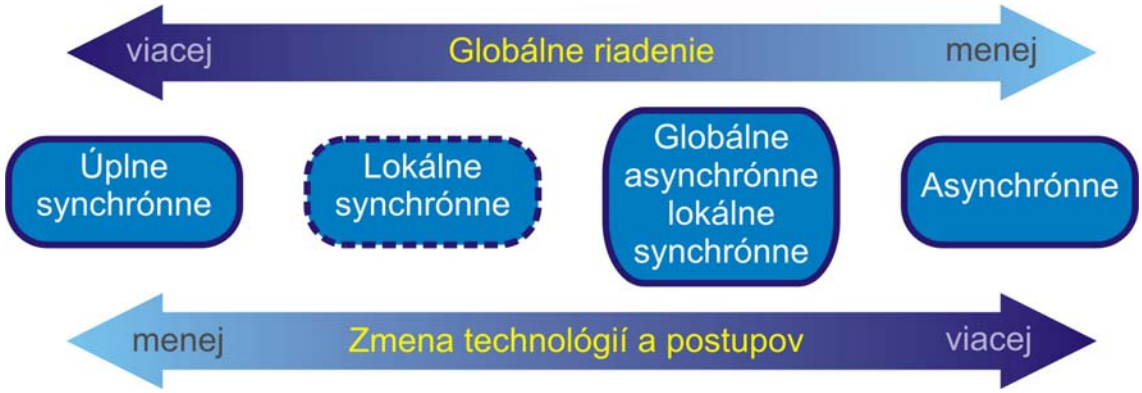
_____

autor práce

Obsah

1 Úvod

Hlavným cieľom výrobcov je okrem zvyšovania miery integrácie aj zvyšovanie rýchlosti obvodov v podobe zvyšovania hodinovej frekvencie. Hodinová frekvencia je závislá jednak na technologickej stránke, na rozvode spoločného hodinového signálu v obvode a na aplikácii. Konkrétne pri rozvode spoločného hodinového signálu nastáva problém s distribúciou hrany hodín, ktorá prichádza k jednotlivým registrom v rozdielnom čase. Tento problém sa nazýva „skew“. Nastáva situácia, keď sa obvody dostávajú k limitom rýchlosti. Výpočet maximálnej frekvencie je závislý na „najpomalšom“ prvku konštrukcie (prvok s najvyšším oneskorením). To znamená, že časová analýza uvažuje najhorší prípad, a preto nie je technológia využitá maximálne. Takéto taktované obvody sa dostávajú k svojim limitom aj kvôli problémom s nadmernou spotrebou a elektromagnetickým rušením. Preto sa v súčasnej dobe výrobcovia vracajú naspäť k asynchrónnym systémom, ktoré majú výhodnejšie vlastnosti. Dôvod, prečo sa masovo nepoužívali, bol v problémoch so zložitým návrhom, chýbajúcimi návrhovými nástrojmi a postupmi. Až s rozvojom výpočtovej techniky a návrhových nástrojov bolo možné prenikať do tejto problematiky podrobnejšie, a to umožnilo pochopiť aj komplexnejšie asynchrónne systémy.



Obr. 1.1 Kategórie systémov

Hlavným rozdielom medzi synchrónnymi a asynchrónnymi systémami je v pojatí času. Asynchrónne systémy nemajú žiadnu predstavu o spoločnom diskrétnom čase – synchronizácii. Asynchrónne obvody používajú medzi svojimi komponentami handshaking na vytvorenie potrebnej synchronizácie, komunikácie a zoradenia operácií.

Hlavnou výhodou asynchrónnych systémov je nízka spotreba, ktorá hrá v dnešnej dobe veľkú úlohu. Stratový výkon, ktorý vzniká v číslicových štruktúrach je statický a dynamický. Ten statický je spôsobený leakage prúdom a je závislý na technológii. Dynamický stratový výkon je definovaný:

$$P_D = C_{ekv}.V_{CC}^2.F \quad , \tag{1.1}$$

kde C_{ekv} je ekvivalentná spínacia kapacita, V_{CC} je napájacie napätie a F je hodinová frekvencia. Je vidieť, že dynamický stratový výkon je závislý na frekvencii

hodín. Asynchrónne systémy nemajú taký problém so stratovým dynamickým výkonom, ich synchronizácia prebieha len lokálne. Majú nulový odber v pohotovosti.

Ďalšou výhodou je operačná rýchlosť. Ako bolo spomenuté, u synchrónnych systémov je počítané s najhorším prípadom. Hodinová perióda musí byť taká dlhá, aby vybavila aj najpomalšiu operáciu. U asynchrónnych systémov sa rýchlosť mení dynamicky v závislosti na aktuálnej lokálnej latencii. Výkon je potom stanovený podľa priemernej latencie (average-case).

Menší elektromagnetický šum je ďalšou výhodnou vlastnosťou asynchrónnych systémov, pretože lokálne „hodiny“ sú spúšťané v náhodných momentoch času. Na rozdiel od synchrónnych obvodov, kde sú hodiny na jednej presnej frekvencii, všetka energia je potom koncentrovaná do veľmi úzkeho spektra na hodinovej frekvencii a jej harmonických. Aktivita u asynchrónnych obvodov je nekorelovaná, preto má viac rozložené šumové spektrum a nižšie šumové špičky.

Jednou z ďalších výhod je robustnosť k variáciám napájacieho napätia, teploty a parametrov vo výrobnom procese. Časovanie u asynchrónnych obvodov je založené na prispôsobených oneskoreniach a môže byť dokonca necitlivé na obvodové a vodičové oneskorenie u niektorých typov asynchrónnych konštrukcií.

Asynchrónne systémy majú aj svoje nevýhody. Preto boli pokusy o vytváranie hybridných systémov, ktoré mali výhodné vlastnosti oboch, aj synchrónnych aj asynchrónnych. Príkladom takého systému pre rozsiahle konštrukcie je GALS (Globally Asynchronous Locally Synchronous). Ako je zo skratky zrejmé, systém komunikuje globálne asynchrónne a lokálne synchrónne. Lokálne moduly sú taktované hodinami a majú výhody jednoduchého synchrónneho návrhu. Medzi sebou komunikujú tieto moduly na väčšie vzdialenosti na čipe asynchrónne a nie je treba dávať pozor na distribúciu hodín.

Iným druhom systémov sú autosynchrónne obvody. Obvody sú z vnútorného pohľadu takmer podobné synchrónnym, ale hodinový signál si generujú sami na základe informácie o preklopení klopných obvodov. Navonok sa ale tvária ako asynchrónne systémy, preto sa nazývajú aj ako „Externe asynchrónne, interne taktované“ (angl. skr. EAIC). U obidvoch prípadoch GALS i EAIC vzniká problém pri komunikácii medzi jednotlivými blokmi, kde pri prechode z asynchrónneho do synchrónneho prostredia a opačne môže vzniknúť metastabilita.

Nasledujúca podkapitola 2.1 v druhej kapitole rozoberá vlastnosti synchrónnych systémov a elimináciu ich nevýhod v podobe hradlovania alebo fázovania hodinového signálu, ďalej v podobe prechodu k hybridným asynchrónno-synchrónnym systémom až k prechodu k čisto asynchrónnym systémom. Podkapitola 2.2 vlastnosti asynchrónnych systémov je hlbšie rozobraná v podobe klasifikácie, základných vlastností a prvkov asynchrónnych systémov až k štýlom popisu asynchrónnych stavových automatov. Nasledujúce dve podkapitoly 2.3 a 2.4 rozoberajú možnosti návrhu asynchrónnych systémov buď už v špecializovaných nástrojoch pre návrh asynchrónnych obvodov alebo pomocou transformácie zo synchrónneho popisu. Z tejto analýzy problematiky sú stanovené ciele práce v tretej kapitole. Štvrtá kapitola práce sa bude venovať analýze a návrhu autosynchrónnych stavových automatov, definovaním a vymedzením ich parametrov. So zavádzaním takýchto systémov do výroby vzniká problém so zautomatizovaním ich návrhu. Táto práca sa v piatej kapitole snaží riešiť tento problém transformáciou z podobnej synchrónnej konštrukcie s čo najmenšími zmenami. Obsahom šiestej kapitoly je porovnanie parametrov autosynchrónnych

obvodov a overenie navrhnutého postupu pre transformáciu. Siedma kapitola teoreticky porovnáva výkonnostné časové parametre medzi stavovými automatmi na rôznej úrovni synchronizácie. V záverečnej ôsmej kapitole sú zhrnuté a zhodnotené dosiahnuté výsledky a prínos práce.

2 Súčasný stav

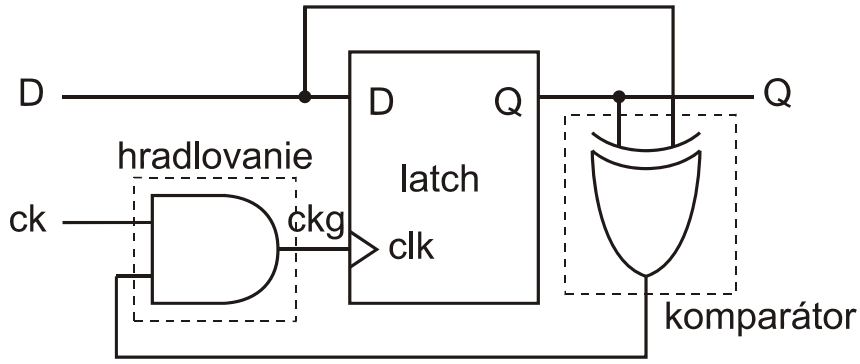
2.1 Vlastnosti a eliminácia nevýhod synchrónnych systémov

U synchrónnych systémov vplyvom rastúcej zložitosti, rýchlosti systémov a zmenšujúcim sa rozmerom spôsobuje hodinový signál nemalé problémy, ktoré nútia návrhárov hľadať iné riešenia od obvodovej až po systémovú úroveň. Najväčším problémom je spotreba, distribúcia hodín a elektromagnetická kompatibilita.

Pre znižovanie spotreby a elektromagnetického rušenia u synchrónnych obvodov boli vytvorené rôzne metódy a postupy. Pre znižovanie spotreby to bolo vypínanie hodín u klopných obvodov, ktorých vstupy boli dlhšie neaktívne, tzv. hradlovanie hodín [12], [13]. Metódou pre znižovanie elektromagnetického rušenia bolo zase riadenie „skew“ (známe aj ako tvarovanie prúdu) a špeciálna syntéza [14], [15], [16].

2.1.1. Hradlovanie hodinového signálu

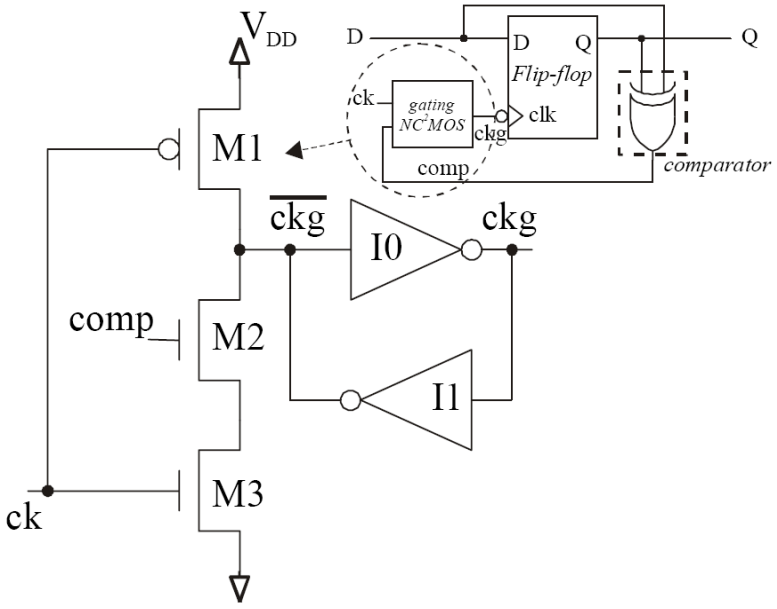
Zdroj [12] sa zaoberal myšlienkou hradlovania hodín na hradlovej a tranzistorovej úrovni v podobe vytvorenia nových klopných obvodov. Jedna technika je nazvaná „Double gating“, kde sa hradlujú časti master a slave separátne. Zapojenie je na Obr. 2.1.



Obr. 2.1 Hradlovanie hodín „double gating“

Komparátor realizovaný členom XOR porovnáva zmeny na portoch klopného obvodu. Jeho výstup hradluje pomocou členu AND hodinový vstup klopného obvodu. Ak sa zmení vstupná hodnota, hodinový signál je spustený. Toto zapojenie má problémy s hodinovým signálom so striedou 50 %, pre správnu činnosť potrebuje asymetrický hodinový signál.

Druhá technika využíva hradlovania pomocou NC2MOS obvodu, z ktorého je zostavený prvok hradlovania a komparátor. Zapojenie zobrazuje Obr. 2.2 [12].



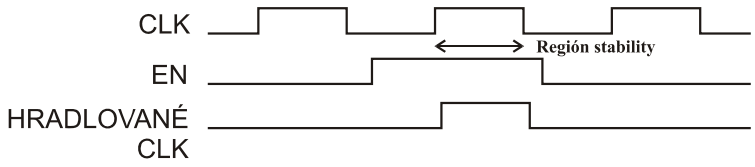
Obr. 2.2 Hradlovanie hodín NC2MOS technológiou

Táto technika je odolná voči 50 % striede. Celkovo má lepšie vlastnosti technika NC2MOS kvôli redukovanej zložitosti a spotrebe hradla s komparátorom.

Zdroj [13] popisuje postup pri redukcii spotreby obvodu ASIC použitím funkcie pre hradlovenie hodín na RTL úrovni návrhu v návrhovom nástroji Synopsis Power Compiler. Jedná sa o vkladanie hradlovacej logiky do prvkov riadených hodinami pomocou skriptu v návrhovom nástroji. Ak sa ich vstupné hodnoty nemenia, sú neaktívne, vypnú sa. Principiálne sa používajú dva postupy, ktoré sa odlišujú obvodovým zapojením a oblasťou stability.

Prvou technikou je hradlovanie bez latchu , kde sa vkladá len jednoduché logické hradlo AND ku klopnému obvodu, ako je vidieť na Obr. 2.3. Táto technika je síce jednoduchá, ale pri bližšom pozorovaní priebehov je možné zistiť, že v hradlovacích hodinách môže vzniknúť zákmit (angl. glitch) alebo môžu mať hradlovacie hodiny kratší impulz za určitých okolností. Preto tu je definovaný región stability, v ktorom sa nemôže meniť hradlovací signál. V tomto prípade je určený ako šírka hornej úrovne hodín, viď. Obr. 2.4.

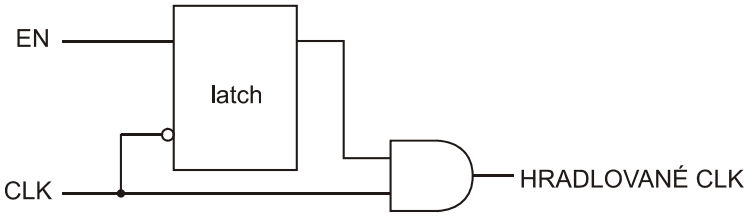
Obr. 2.3 Hradlovanie AND hradlom



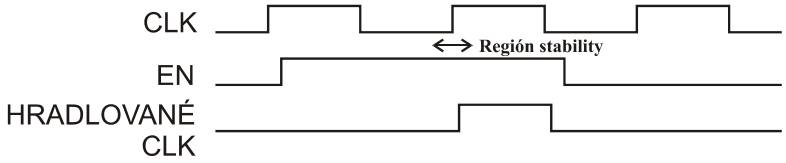
Obr. 2.4 Časovanie hradlovania s AND hradlom

Druhý prípad je tvorený zložitejším obvodovým zapojením s latchom na Obr. 2.5. Pri aktívnej hrane hodín pridrží latch hradlovací signál po celú dobu hodinovej periódy.

Jediným kritériom stability je dodržať časovanie v okolí aktívnej hrany hodinového impulzu (t_{set}, t_{hold}). Ako je vidieť na Obr. 2.6, oblasť stability sa výrazne zmenšila.



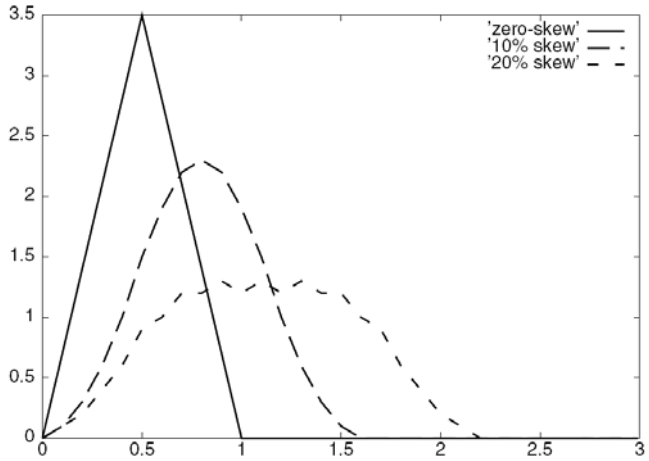
Obr. 2.5 Hradlovanie s latchom



Obr. 2.6 Časovanie hradlovania s latchom

2.1.2. Riadenie „skew“

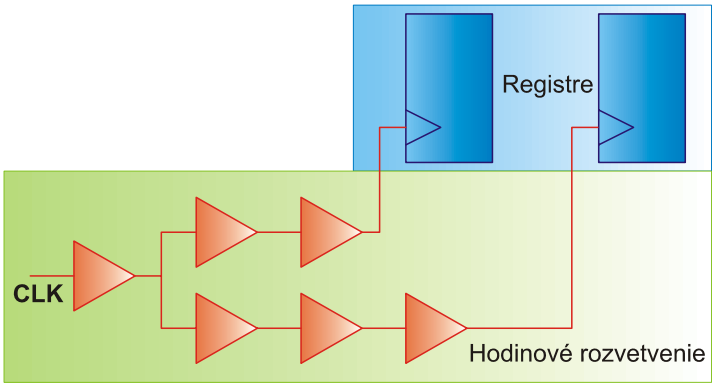
Ďalší problém synchrónnych obvodov vzniká so súčasným spínaním logických hradiel. To vedie k úzkym prúdovým impulzom na napájacích a zemniacich vývodoch. Vzniká elektromagnetické rušenie na frekvencii hodín a jej harmonických s úzkym spektrom a veľkou amplitúdou, ktoré pôsobí rušivo na ostatné obvody a okolie. Riešilo sa to filtrovaním s premosťovacími a blokovacími kondenzátormi, ale tieto ochrany boli buď drahé alebo neúčinné, alebo zvyšovali problém vyžarovania. Ďalšou možnosťou bolo modulovanie hodinového signálu pre zníženie vyšších harmonických. Zdroj [15] sa zmieňuje tiež o rozprestieraní hodinového signálu. Táto technika je však pri normálnom použití vnútorne generovaných hodín pomocou PLL príliš komplikovaná. V [14] je popísaná redukcia elektromagnetických emisií (EME) optimalizáciou hodinového stromu. Je tu riadené „skew“ pamäťových prvkov, čo dovoľuje tvarovať napájacie prúdy a tým redukciu EME. „Skew“ je riadený tak, aby bol rozložený prúdový odber do širšieho a hladšieho prúdového impulzu, ako je vidieť na Obr. 2.7.



Obr. 2.7 Rozloženie prúdových impulzov v závislosti na rozprestrení „skew“

Tento postup sa vykonáva pomocou rozdelenia logiky do viacerých hodinových domén po návrhovom procese Place&Route. Je tu komplikovaná časová analýza.

Podobným štýlom znižovania EME sa uberá aj [15], ktorý odvodzuje úroveň radiácie od parametrov ako sú veľkosť hodinovej frekvencie, strmosť hrán, obvodové riešenie a tienenie. Na základe týchto predpokladov uvádza dve techniky. Prvou je zvýšenie času nástupnej alebo zostupnej hrany hodinového signálu a druhou je riadenie „skew“ spomenuté v [14]. Princíp riadenia „skew“ je zobrazený na Obr. 2.8. Je to podobné ako u vyrovnávania „skew“, ale s opačným zámerom. Kvôli nevyrovnaným vlastnostiam je najvhodnejšie tieto dve techniky kombinovať.

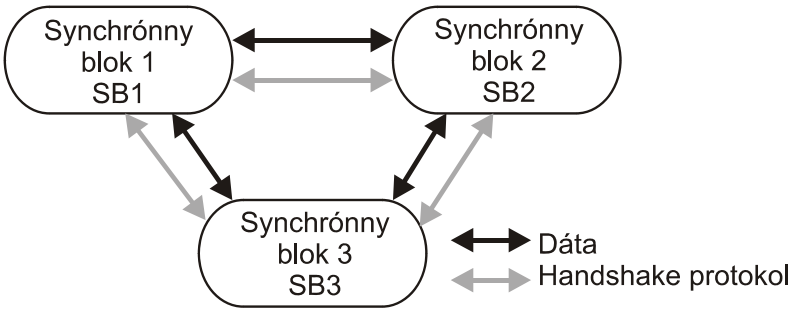


Obr. 2.8 Princíp riadenia „skew“

Zdroj [16] predstavuje dve techniky: tvarovanie prúdu kombinačných blokov správnou syntézou a rozprestrením hrán hodín sekvenčných blokov ako to bolo v [14]. Prvá technika vychádza z toho, že šírka impulzu je úmerná „worst-case delay“. Syntézne nástroje sa snažia o čo najmenšie oneskorenie a tým aj o najužší impulz, čo je zlé kvôli EME.

2.1.3. Globálne asynchrónne lokálne synchrónne systémy GALS

Prameň [18] skúma techniku znižovania spotreby hodinového signálu použitím technológie GALS. Táto technika predstavuje elimináciu globálneho hodinového signálu a rozdelenie celého synchrónneho systému na menšie synchrónne bloky (ďalej SB), ktoré medzi sebou komunikujú asynchrónne, pričom vo vnútri sú taktované synchrónne podľa lokálnych hodín, viď Obr. 2.9.



Obr. 2.9 Komunikácia synchrónnych blokov v GALS

Zrušením globálnych hodín sa zruší hlavný zdroj spotreby a stagnácie návrhu. Spotreba je závislá na napájacom napätí, spínacej kapacite a frekvencii spínania. Prvé

dva parametre sú technologicky a obvodovo závislé. Tretí parameter, spínacia frekvencia, má potenciál k variabilite vo fáze logického návrhu.

Najdôležitejšími časťami návrhu GALS oproti čisto synchrónnemu návrhu je efektívne rozdelenie synchrónnych blokov pri maximalizácii výhod GALS oproti ich nákladom. Tento proces sa nazýva predrozdelenie (angl. pre-partitioning). Druhou odlišnosťou návrhu je syntéza asynchrónneho rozhrania medzi synchrónnymi blokmi. Štandardne sa používa štvorfázový handshaking. Na konci návrhovej fázy prichádza k vyčísleniu úspory spotreby. Ak je úspora nevyhovujúca, celý proces sa znovu opakuje s iným prerozdelením synchrónnych blokov.

Spotrebu hodín GALS s N blokmi je možné vyčísliť ako:

$$P_{clock}(N) = k_1.N.a.\sqrt{a}\,, \tag{2.1}$$

kde A je celková plocha, $a=A/N$ je priemerná plocha synchrónneho bloku a k_1 obsahuje konštanty proporcionality, technológie a návrhových faktorov. Spotreba hodín je daná dvoma faktormi: záťažou sekvenčných prvkov ($N.A$) a kapacitou vedenia, ktorá je úmerná jeho dĺžke ($\sqrt{a}$).

Úspora spotreby globálneho synchrónneho systému oproti GALS je potom:

$$\frac{P_{clock}(1)}{P_{clock}(N)} = \frac{k_1.A.\sqrt{A}}{N.k_1.\frac{A}{N}.\sqrt{\frac{A}{N}}} = \sqrt{N}\,. \tag{2.2}$$

Hodnota $\sqrt{N}$ je horná hranica úspory spotreby použitím GALS. Spotreba monotónne rastie s N a pri veľmi veľkom počte N presiahnu náklady výhody GALS.

Problémy s komunikáciou medzi modulmi SB sú závislé od módu komunikácie, od frekvencie s ktorou medzi sebou moduly komunikujú, od počtu komunikujúcich modulov SB a od dĺžky vodičov. Pri návrhu rozdeľovania treba dodržať minimalizáciu externej komunikácie a blízkosť spolu komunikujúcich SB. Náklady na komunikáciu je možné zapísať ako:

$$P_{com} = \left(\sum_{b}^{B} \sum_{i}^{X_b} \left(n_{sig} C_w l_{b,i} + n_{reg} C_{reg} \right) \right) f_{b,i} V_{dd}^2\,, \tag{2.3}$$

kde B je súbor komunikujúcich párov SB, X_b je súbor komunikujúcich inštancií pre jednotlivý pár b komunikujúcich SB, n_{sig} je počet signálov v komunikačnom protokole, C_w je kapacita vodiča na jednotku dĺžky, $l_{b,i}$ je dĺžka vodiča pre komunikačný signál, n_{reg} je počet registrov pripojených na riadiace signály, C_{reg} je elektrická kapacita registrov a $f_{b,i}$ je frekvencia, s ktorou komunikujú SB moduly.

Dôležitou časťou je generovanie lokálnych hodín. Používajú sa buď nezávislé generátory pre každý SB alebo globálny referenčný signál, kde má každý SB násobičku pre svoju požadovanú frekvenciu. Pre distribúciu globálneho referenčného signálu sú dané požiadavky ako rozkmit signálu v zlomkoch napájacieho napätia a distribúcia na oveľa nižšej frekvencii, aké potrebujú SB, ktoré si ju vynásobia. Výhodou je, že sa neberie do úvahy „skew“.

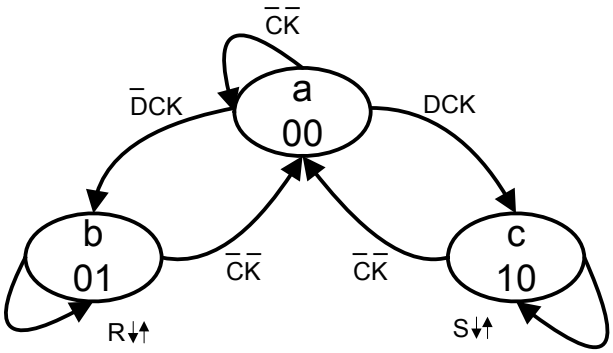
Pre násobenie v SB sa používali analógové násobičky s PLL. Ich integrácia v rušivom digitálnom prostredí je náročná a taktiež sú citlivé aj na variácie procesov. Lepšie vlastnosti má v súčasnosti kruhový oscilátor riadený globálnym hodinovým signálom. Je malý, odolný a má malú spotrebu. Frekvencia sa dá meniť zmenou

transportného oneskorenia a to zmenou počtu invertorov. Iná možnosť je použitie prúdovo ochudobnených invertorov alebo oneskorovacej linky riadených kapacít. Kruhové oscilátory sú perspektívne a lacné riešenia v GALS.

2.1.4. Externe asynchrónne vnútorne taktované systémy (EAIC)

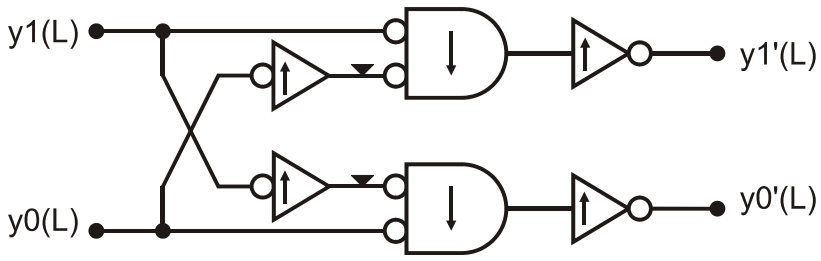
Systémy EAIC sú kompromisom medzi asynchrónnymi a synchrónnymi návrhovými postupmi. Tieto systémy komunikujú asynchrónne vzhľadom k vonkajšiemu prostrediu, zatiaľ čo sú riadené vnútorne generovanými hodinami, ktoré sú generované pri platných výstupoch z pamäťových prvkov. Pre vnútorné taktovanie je nutné, aby tieto systémy boli bez kritických súbehov a hazardov. Pamäťové prvky sú navyše chránené pred chybami spôsobenými metastabilitou. Rýchlosť vnútorných hodín je limitovaná len aktuálnym logickým oneskorením systému oproti najväčšiemu oneskoreniu v synchrónnych systémoch.

Najdôležitejšou časťou je pamäťový prvok, nazvaný DFLOP, ktorý synchronizuje vytváranie vnútorných hodín signálom *R*. Oproti hranou riadenému klopnému obvodu D obsahuje navyše výstup *R*, ktorý signalizuje, kedy sa DFLOP preklopil a je pripravený na zostupnú hranu hodín. Obsahuje tri časti. Prvou časťou je rozhodovací obvod (angl. resolver) v podobe troj-stavového automatu zobrazenom v stavovom diagrame na Obr. 2.10 [19]. Dvojité šípky pri výstupných signáloch v stavoch „01“ a „10“ znamenajú aktivovanie výstupu pri prechode do tohoto stavu a deaktivovanie výstupu pri opustení tohoto stavu.



Obr. 2.10 Rozhodovací obvod „resolver“

Ďalšou časťou je obvod vzájomnej výlučnosti, ktorý predchádza možnému metastabilnému stavu z rozhodovacieho obvodu „resolvera“. Z Obr. 2.11 [19] je zrejmá konštrukcia z krížovo zapojených hradiel AND.



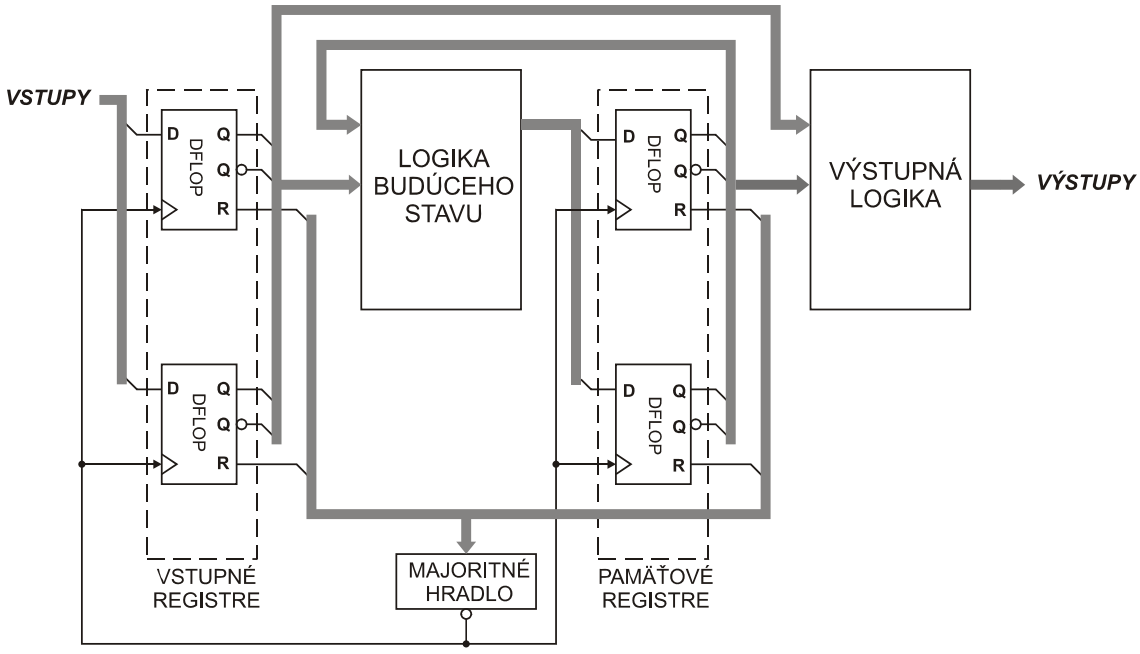
Obr. 2.11 Obvod vzájomnej výlučnosti

Hradlá a invertory označené šípkami na Obr. 2.11 označujú posun rozhodovacej úrovne podľa smeru šípky hore alebo dolu. Je to preto, že pri metastabilnom stave je stredná hodnota výstupného napätia hradla približne polovica z napájacieho napätia. Pomocou posunu prahu rozhodovania hradiel mimo tuto úroveň sa obmedzí možnosť dostať sa do stavu metastability.

Poslednou časťou modulu DFLOP je základná pamäťová bunka (SR latch), ktorá ukladá nastavovací alebo nulovací výstup z rozhodovacieho obvodu „resolvera“ cez vzájomne výlučný obvod.

Modul DFLOP je možné realizovať v statickej CMOS logike alebo v dynamickej domino CMOS logike. Dynamická domino logika má výhody v podobe zvýšenej rýchlosti a výkonu modulu DFLOP.

Všeobecná štruktúra EAIC systému je zobrazená na Obr. 2.12 [19]. Systém sa skladá z dvoch typov DFLOP pamäťových modulov: zo vstupných synchronizačných a pamäťových registrov. Ďalšími časťami sú logika budúceho stavu a obvod generujúci hodiny. Obvod pracuje tak, že pri nástupnej hrane každého hodinového cyklu sú vstupy uložené vo vstupnom registri a nový stav je uložený v pamäťovom registri závislý od logiky budúceho stavu v predchádzajúcom takte. Keď sa každý klopný obvod preklopí, nastaví výstupný riadiaci signál *R* pre majoritné hradlo, ktoré spôsobí zostupnú hranu hodín, keď sú preklopené všetky moduly DFLOP. Po zostupnej hrane sa vracajú DFLOP moduly do nerozhodnutého stavu a nulujú signál *R*.



Obr. 2.12 Štruktúra EAIC

Majoritné hradlo vlastne robí úlohu kompletovacieho detektora, ktoré sa aktivuje pri pripravenosti všetkých DFLOP modulov. Býva zostavené z Mullerových elementov C alebo logickým súčinom hradiel NOR.

Z výkonnostného hľadiska je dôležité, aby aktualizovaný výstup zo vstupného registra prešiel cez logiku budúceho stavu pred ďalšou hranou hodín. Je potrebné, aby:

$$\delta_{NS} \le \left(2\delta_{maj.gate} + \delta_{DFLOP}\right), \tag{2.4}$$

kde δ_{NS} je transportné oneskorenie cez logiku budúceho stavu, $\delta_{maj.\ gate}$ je transportné oneskorenie majoritného hradla a δ_{DFLOP} je transportné oneskorenie modulu DFLOP. Obmedzená je aj šírka vstupného impulzu, ktorá musí byť väčšia ako perióda vnútorných hodín. Frekvencia vnútorných hodín je kombináciou transportného oneskorenia registrov s majoritným hradlom vyjadrená ako:

$$f_{CK} = \left(2\delta_{DFLOP} + 2\delta_{maj.gate}\right)^{-1}. \tag{2.5}$$

Výkon tohto systému je definovaný ako čas uplynutý medzi zmenou vonkajších vstupov a tomu vyplývajúcej zmeny výstupov z pamäťového registra. Výkon systému EAIC leží v rozmedzí:

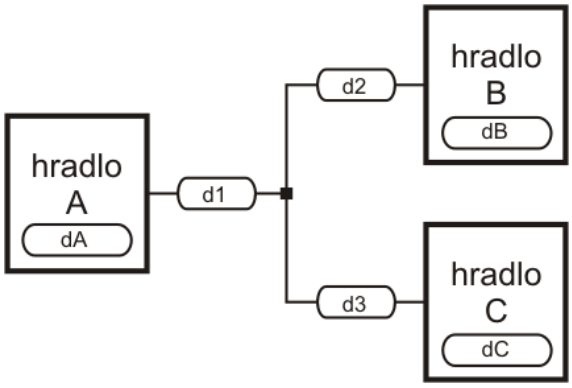
$$\left(f_{CK}^{-1} + 3\delta_{DFLOP} + 2\delta_{maj.gate}\right) \geq \delta_{throughput} \geq \left(3\delta_{DFLOP} + 2\delta_{maj.gate}\right). \tag{2.6}$$

2.2 Vlastnosti asynchrónnych systémov

2.2.1. Klasifikácia asynchrónnych obvodov

Dôležitým aspektom pri analýze asynchrónnych systémov je vedomosť o ich rozdelení podľa kritérií, ktorá je dôležitá pri zvolení správnej testovacej metódy. Najdôležitejšie kritérium je posudzovanie na základe oneskorenia v hradlách a vodičoch na hradlovej úrovni. V [1] a [7] rozdeľujú asynchrónne obvody na necitlivé na oneskorenie (delay-insensitive DI), nezávislé na rýchlosti (speed-independent SI) a samospúšťané (self-timed ST).

Obvody v triede obvodov necitlivých na oneskorenie (DI) pracujú korektne nezávisle na ľubovoľnom nenulovom oneskorení v hradlách a vodičoch. Obr. 2.13 zobrazuje názorne model oneskorenia, kde symboly *dA*, *dB*, *dC* značia transportné oneskorenia v hradlách a symboly *d1*, *d2*, *d3* značia oneskorenia vo vodičoch. V tejto triede platí potom, že *dA*, *dB*, *dC*, *d1*, *d2*, *d3* sú ľubovoľne veľké. U týchto obvodov musí byť každá zmena signálu potvrdzovaná. To platí aj u rozvetvení, kde musí prísť potvrdenie z každej vetvy pred príchodom nového signálu. Pretože je tu každá zmena potvrdzovaná, každá porucha môže spôsobiť zastavenie obvodu. Tieto obvody majú preto plnú samotestovateľnosť. Do tejto skupinky patrí len veľmi málo obvodov, pretože musia byť zostavené výhradne z Mullerových elementov C a invertorov. Obvody sú robustné a tým pádom nie príliš praktické.



Obr. 2.13 Model oneskorenia hradiel a vodičov s vetvením

Ďalšou skupinou sú obvody nezávislé na rýchlosti (SI). Tieto obvody predpokladajú ľubovoľné nenulové oneskorenie v hradlách a zanedbateľné nulové

oneskorenie vo vodičoch. Podľa Obr. 2.13 sú to ľubovoľné *dA*, *dB*, *dC* a *d1 = d2 = d3 = 0*. Keďže je predpokladané oneskorenie vo vodičoch nulové, pri rozvetveniach vodičov stačí ak bola zmena na vstupe rozvetvenia potvrdená len z jednej vetvy. Tento predpoklad je ekvivalentný s predpokladom, že všetky rozvetvenia sú izochronické.

Medzitriedou medzi posledne spomínanými triedami DI a SI je trieda obvodov takmer necitlivých na oneskorení (quasi delay-insensitive QDI). V tejto triede sú predpokladané len niektoré rozvetvenia za izochronické, teda podľa Obr. 2.13 platí: *d2 = d3*.

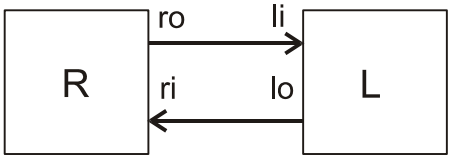
Ako to vyplýva z [1], posledná trieda samospúšťaných (ST) obvodov zahrňuje všeobecne všetky asynchrónne obvody, u ktorých nie sú presne definované vlastnosti oneskorení hradiel a vodičov ako u predchádzajúcich tried.

V súčasnosti rozsiahle systémy na čipe (SoC) potrebujú asynchrónne riadenie, pretože je veľmi zložité pri veľkej variácii parametrov kontrolovať oneskorenia pre hodinový signál (časová analýza).

2.2.2. Základné prvky a vlastnosti asynchrónnych systémov

Ucelený prehľad vlastností asynchrónnych obvodov je možné nájsť v [17]. Tento zdroj udáva prehľad o používaných komunikačných protokoloch, základných stavebných prvkoch a návrhových štýloch.

Na rozdiel od synchrónnych obvodov prebieha u asynchrónnych obvodov lokálna komunikácia na bázy tzv. handshakingu. Čistý handshaking protokol bez dát sa používa pre synchronizáciu medzi dvoma procesmi, viď. Obr. 2.14. Kanál je implementovaný dvoma vodičmi pre požiadavok (request) a pre potvrdenie (acknowledge). Podľa možnosti návratu k nule môže byť protokol dvojfázový alebo štvorfázový.



Obr. 2.14 Samostatný handshaking

Dvojfázový protokol sa nazýva aj protokol bez návratu k nule (NRZ). Je možné ho zapísať pomocou CHP (Communicating Hardware Processes) zápisom (2.7). CHP je jazyk s vysokou úrovňou abstrakcie, ktorý popisuje komunikáciu hardvéru pomocou procesov, kap.2.3.1. Za účelom zjednodušenia a názornosti popisu komunikácie asynchrónnych prvkov bude použitá táto forma zápisu aj ďalej v tejto kapitole.

$$\begin{aligned} Ru &: ro\uparrow;[ri] \\ Lu &: [li];lo\uparrow . \end{aligned} \tag{2.7}$$

Zápis (2.7) znázorňuje najjednoduchšiu komunikáciu – handshaking. Symboly *Ru* a *Lu* označujú procesy pre jednotlivé strany komunikácie. Šípka smerom hore znamená nastavenie signálu do hornej úrovne, šípka smerom dole znamená opačne nastavenie na spodnú úroveň. Signál v hranatých zátvorkách je očakávaný signál v hornej úrovni, tzn. až keď bude signál v hornej úrovni, môže proces pokračovať ďalej. Signál v hranatých zátvorkách so symbolom $\neg$ označuje čakanie na signál s dolnou úrovňou. Vysvetlenie

zápisu je potom nasledovné: strana *Ru* pošle požiadavku *ro*, a čaká na potvrdenie *ri*. Strana *Lu* zase čaká na nastavenie *li* a potom vysiela *lo*.

Zápis (2.7) je nekompletný, vyjadruje len prechod z inicializovaného stavu (false). Keďže sa protokol nevracia k nule, je treba zápis pre ďalší protokol, ktorý zase nuluje signály:

$$\begin{aligned} Rd &: ro\downarrow;[\neg ri] \\ Ld &: [\neg li];lo\downarrow. \end{aligned} \tag{2.8}$$

Alebo je možné zapísať bez rozlíšenia pre obidve fázy:

$$\begin{aligned} R &: ro := \neg ro;[ro = ri] \\ L &: [lo \neq li];lo := \neg lo. \end{aligned} \tag{2.9}$$

Pri podrobnejšom skúmaní sa dá povedať, že tento protokol komunikuje vlastne podľa hrán. Je tu komplikovaná obvodová implementácia, sú nutné aritmetické a logické operácie na transportovaných dátach v oboch smeroch. Používa sa len zriedkavo.

Štvorfázový protokol sa nazýva inak aj protokol s návratom k nule. Preto sa vždy nulujú všetky premenné do východiskových hodnôt. Štvorfázový protokol je možné zapísať:

$$\begin{aligned} R &: ro\uparrow;[ri];ro\downarrow;[\neg ri] \\ L &: [li];lo\uparrow;[\neg li];lo\downarrow. \end{aligned} \tag{2.10}$$

Tento protokol je vlastne poskladaný z dvoch dvojfázových operácií za sebou. Strana *R* nastaví požiadavku *ro* do hornej úrovne, čaká na nastavenie potvrdenia *ri* do hornej úrovne. Po prijatí *ri* nastaví *ro* do spodnej úrovne a čaká na nastavenie *ri* do spodnej úrovne. Strana *L* čaká na *li* v hornej úrovni, potom nastaví signál *lo* do hornej úrovne. Strana *L* ďalej čaká na signál *li* v dolnej úrovni a potom nastavuje signál *lo* do dolnej úrovne.

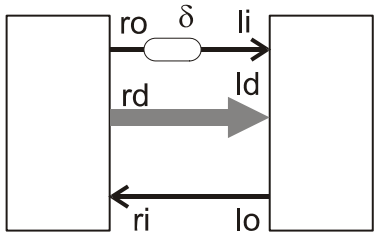
Okrem funkcie synchronizácie slúžia protokoly aj na prenos dát. Dáta sa buď prenášajú vo zväzku (bundled data) podobne ako u synchrónnych obvodov alebo sú kódované spolu s riadiacimi signálmi (dual-rail). Rozdiel medzi nimi je v predpoklade o platnosti dát. Dáta vo zväzku neprechádzajú komunikačným kanálom všetky rovnako rýchlo ako riadiaci signál, a preto je treba vytvoriť prispôsobené oneskorenie v ceste riadiacich signálov, ktoré je dostatočne veľké na to, aby boli dáta v momente požiadavku platné. Inou cestou sa vydávajú protokoly s kódovaním dát. V týchto protokoloch nie sú potrebné prispôsobené oneskorenia, pretože dáta sú kódované spolu s riadiacimi signálmi. Dáta si sami určujú svoju platnosť a neplatnosť.

Protokol pre prenos dát vo zväzku sa nazýva bundled-data. Jeho zápis vyjadruje vzťah (2.11) a princíp komunikácie je na Obr. 2.15.

$$\begin{aligned} R!x &: rd := x;ro\uparrow;[ri];ro\downarrow;[\neg ri] \\ L?y &: [li];y := ld;lo\uparrow;[\neg li];lo\downarrow \end{aligned}. \tag{2.11}$$

Odosielateľ *R!x* posiela na linku *rd* dáta *x* a nastavuje požiadavku *ro* do hornej úrovne. Ak mu príde potvrdenie *ri* v hornej úrovni, tak nastavuje požiadavku *ro* do dolnej úrovne a čaká na nastavené potvrdenie *ri* v dolnej úrovni. Príjemca *L?y* najprv čaká na požiadavku odosielateľa *li* v hornej úrovni, prijme dáta z *ld* a nastaví potvrdenie

prijatia *lo* do hornej úrovne. Príjemca čaká na nastavenie požiadavku *li* do dolnej úrovne, aby mohol nastaviť potvrdenie *lo* na dolnú úroveň.



Obr. 2.15 Bundled-data protokol

Ako je vidieť na Obr. 2.15, u bundled-data protokolu sa musí použiť prispôsobené oneskorenie δ v riadiacej ceste, aby boli všetky bity dát na vstupe *ld* platné pred príchodom požiadavku *li*.

V protokoloch s dátami sa platnosť dát dá určiť prispôsobeným oneskorením, ako to bolo uvedené vyššie, alebo sa dá platnosť dát určiť ich kódovaním s riadiacimi signálmi. Najpoužívanejšie kódy sú dual-rail a kód 1 z N. U protokolu dual-rail sa používajú dva vodiče (*d.t* a *d.f*) na jeden bit dát. Kódovanie je uvedené v Tab. 2.1. Vysoká úroveň na jednom vodiči *d.f* signalizuje hodnotu logickej nuly, resp. na druhom vodiči *d.t* logickej jednotky. Nízka úroveň na obidvoch vodičoch signalizuje neutrálny stav, tzv. stav neplatnosti dát.

Tab. 2.1. Kódovanie dual-rail

d.t	d.f	stav
0	0	bez dát
0	1	platná 0
1	0	platná 1
1	1	nepoužitý

Praktické požiadavky pre kódy sú jednoduchosť testovania, jednoduchosť kódovania a dekódovania dát a malý pomer kódovaného slova k dátovému. Z tohto dôvodu sa v praxi používa len kódovanie dual-rail a zo skupiny 1 z N len kód 1 zo 4, a to len pre dvojice bitov. Pre prenos N-bitového slova je potreba 2×N vodičov u kódovania dual-rail. Pre kódovanie 1 z N by bolo potreba 2^N vodičov, ale pri rozdelení po dvojiciach bitov a kódovaní každej dvojice štyrmi vodičmi dostávame rovnaký počet 2×N vodičov.

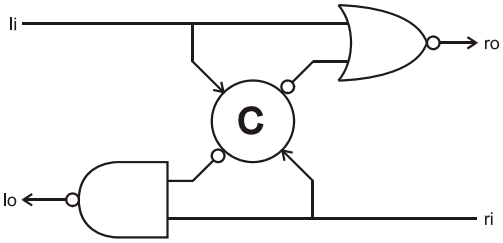
Protokoly je možné podľa aktivity rozdeliť na symetrické alebo nesymetrické. Najčastejšie sa používajú nesymetrické a to: aktívny-pasívny protokol alebo pasívny-aktívny protokol. Symetrické protokoly (aktívny-aktívny) sú komplikované.

Pre asynchrónne obvody sú dôležité tri základné stavebné bloky. Prvým je obvod, ktorý posúva komunikáciu ďalej zľava doprava, prípadne opačne (opakovač). Druhým je obvod, ktorý číta a zapisuje jednobitovú informáciu (register). Tretím je obvod, ktorý počíta Booleovské funkcie malého počtu bitov.

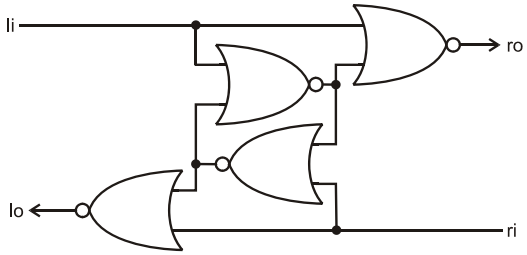
Opakovač je obvod, ktorý presúva ďalej komunikáciu, je možné ho nazvať aj ako opakovač zľava doprava. Aktívny-aktívny opakovač je možné zapísať ako:

$$*[lo\uparrow;[li];x\uparrow;lo\downarrow;[\neg li];ro\uparrow;[ri];x\downarrow;ro\downarrow;[\neg ri]]. \qquad (2.12)$$

Jeho možné implementácie sú zobrazené na Obr. 2.16 s Mullerovým elementom C alebo na Obr. 2.17 s hradlami NOR.



Obr. 2.16 Opakovač s elementom C

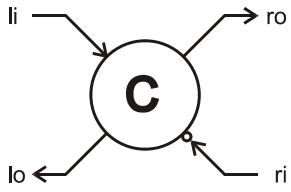


Obr. 2.17 Opakovač s NOR hradlami

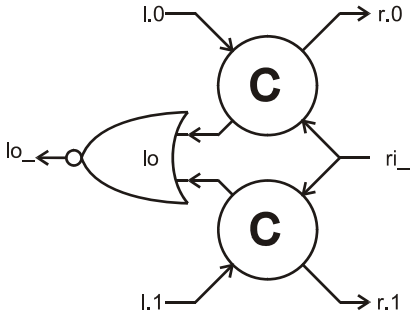
Transformáciou zápisu môže vzniknúť jednoduchý polovičný opakovač (half-buffer), ktorý má zápis:

$$*[[\neg ri];[li];lo\uparrow;ro\uparrow;[\neg li];[ri];lo\downarrow;ro\downarrow]. \tag{2.13}$$

Ako je vidieť na Obr. 2.18, tento zápis sa dá implementovať ako jednoduchý Mullerov element C, ktorý sa používa na konštrukciu jednoduchých registrov FIFO. Na Obr. 2.19 vedľa je vidieť polovičný opakovač prenášajúci jeden bit dát.



Obr. 2.18 Polovičný opakovač bez dát

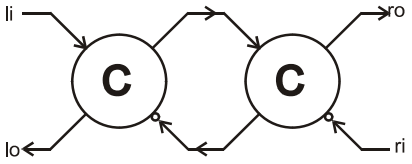


Obr. 2.19 Polovičný opakovač s 1 bitom dát

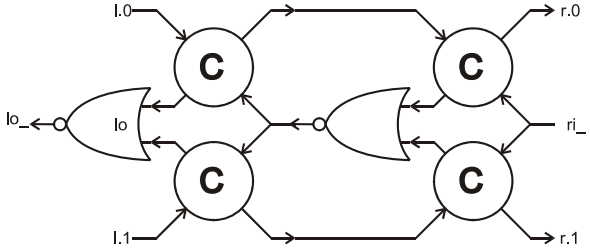
Lineárnou kompozíciou dvoch Mullerových elementov C vznikne celý opakovač, ako je vidieť na Obr. 2.20 bez dát, resp. na Obr. 2.21 s prenosom 1 bitu dát. Zápis celého opakovača bez dát je:

$$*[[li];lo\uparrow;[\neg ri];ro\uparrow;[\neg li];lo\downarrow;[ri];ro\downarrow]. \tag{2.14}$$

Medzi ľavou a pravou stranou celého opakovača vzniká mŕtva zóna (angl. slack).



Obr. 2.20 Opakovač bez dát



Obr. 2.21 Opakovač s 1 bitom dát

Na zápisy CHP je možné použiť preskupovaciu transformáciu (angl. reshuffling), ktorá presúva časť handshake sekvencie. Preskupovacia transformácia sa používa k zjednodušeniu implementácie, redukuje počet stavov pri prekrývajúcich sa handshake sekvenciách. Táto transformácia môže redukovať aj mŕtvu zónu (slack) v registri FIFO. Musí sa ale používať kompromis medzi redukciou zložitosti a redukciou mŕtvej zóny (slack), ktorý zase súvisí s výkonom.

N-bitový register je poskladaný z paralelných 1-bitových registrov. Potvrdzovacie signály (*ack*) zo všetkých registrov sú poskladané do jedného spoločného Mullerovho elementu C s *n* vstupmi. Popis jeho chovania je daný ako:

$$*[(x_1\uparrow, x_2\uparrow, ...x_n\uparrow); y\uparrow; (x_1\downarrow, x_2\downarrow, ...x_n\downarrow); y\downarrow], \tag{2.15}$$

kde x_1 až x_n sú vstupy a y je výstup.

Kvôli obvodovej zložitosti viacvstupových Mullerových elementov C je ho možné rozložiť na viac dvojvstupových Mullerových elementov C. Tento prvok má funkciu kompletovacieho detektora, čo je vlastne najneefektívnejšia časť v návrhu QDI obvodov. Pri viacbitových riešeniach nastáva problém, že oneskorenie kompletovacieho detektora je omnoho väčšie ako čas potrebný k zápisu dát do registra. Preto sa vedú diskusie, či nie je efektívnejšie použiť bundled-data štruktúru a pridať namiesto detektoru oneskorovaciu linku, ktorá odzrkadľuje oneskorenia potrebné k zápisu dát do registra. Pred návrhárom stojí otázka, či vyberie variantu s obvodovou zložitosťou a zvýšením plochy na čipe v podobe kompletovacieho detektora, alebo variantu kde je časová rezerva v oneskorovacej linke, ktorá zas znižuje rýchlosť obvodu. Pri súčasnej modernej technológii integrovaných obvodov a rozdelení dátovej cesty na menšie časti s jednoduchšími kompletovacími detektormi vychádza perspektívnejšie varianta s kompletovacím detektorom.

Pri použití funkčného bloku v asynchrónnom systéme pre aritmetické a logické operácie je dôležité testovať platnosť a neutralitu vstupov. Niekedy sa to dá urobiť priamo vo vyhodnotení funkcie. Pre zložitejšie výpočty sa to dá spočítať ako samostatná funkcia, ktorú je možno implementovať kompletovacím stromom a jej hodnotami riadiť výpočet požadovanej funkcie. Takémuto systému sa hovorí „precharge“.

Pre návrh asynchrónnych pipeline sú možné dva štýly. Prvou možnosťou je hrubozrnná štruktúra, kde sú oddelené riadenie a dátová cesta. Druhou možnosťou je jemnozrnná štruktúra, tzv. integrované pipeline, kde je dátová cesta rozdelená na menšie časti kvôli nákladom s kompletovacím detektorom. Riadenie a dátová cesta je integrovaná spolu v každej časti. Výhodou prvej varianty je jej jednoduchosť, všeobecnosť, rýchly návrh a syntéza obvodu. Naopak nevýhodou je čas cyklu, dopredná latencia a energia cyklu.

Ďalšími konštrukčnými prvkami asynchrónnych systémov pre konštrukciu zložitejších systémov sú asynchrónne zbernice založené na jednoduchších prvkoch „merge“ a „split“, ktoré sa dajú prirovnať k multiplexoru a demultiplexoru v synchrónnych systémoch. U prvku „merge“ sa zlučuje dva alebo viac signálov na vstupe do jedného výstupu v závislosti na riadiacom vstupe. U prvku „split“ je to zas analogicky opačne.

Ďalšou dôležitou funkciou v asynchrónnych systémoch a ich rozhraniach so synchrónnymi systémami je funkcia rozhodovania. Na vstupe nie sú navzájom výlučné podmienky ako u prvku „merge“. Nastáva tu nedeterministický výber, ktorý nedovoľuje

stabilnú a neinterferujúcu implementáciu, preto je treba špeciálne nedigitálne riešenie v podobe prvku arbiter a synchronizér.

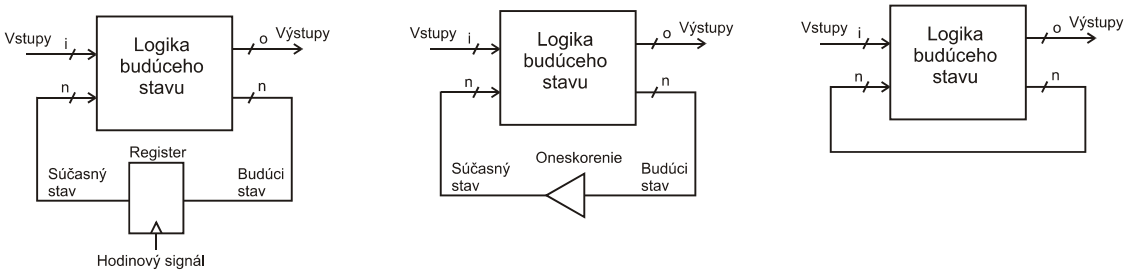
Popis základného arbitra je uvedený ako:

$$arb = *[[x \to u\uparrow;[\neg x];u\downarrow \\ \left|y \to v\uparrow;[\neg y];v\downarrow\right]'] \tag{2.16}$$

kde tenká zvislá čiara znamená činnosť v rovnaký čas. Aby arbiter správne pracoval, musia byť jeho vstupy stabilné, pokiaľ nebude aktivovaný jeden z jeho výstupov. Preto nie je možné použiť arbiter pre nesynchronizovaný signál. Pre tento účel sa používa synchronizér.

2.2.3. Sekvenčné asynchrónne obvody

Medzi najjednoduchšie sekvenčné obvody patria stavové automaty. Táto podkapitola rozoberá možné štýly návrhu asynchrónnych stavových automatov. Literatúra [1] rozdeľuje tieto asynchrónne sekvenčné obvody na Huffmanove vo fundamentálnom režime a Mullerove vo vstupno-výstupnom režime. Obr. 2.22 zachytáva porovnanie týchto sekvenčných obvodov so synchrónnou verziou stavového automatu. Je vidieť, že sa líšia len v prvku spätnej väzby. U synchrónneho automatu na Obr. 2.22a je v spätnej väzbe pamäťový prvok (klopný obvod) taktovaný hodinovým synchronizačným signálom, ktorý preklápa budúci stav do súčasného stavu. V kombinačnej logike môžu vznikať hazardy, preto sú výstupy z kombinačnej logiky filtrované preklápaním len v čase hrany hodinového signálu.



Obr. 2.22 a.) Synchrónny stavový automat, b.) Huffmanov a c.) Mullerov asynchrónny stavový automat

Obr. 2.22b ukazuje Huffmanov štýl asynchrónneho sekvenčného obvodu, ktorý obsahuje oneskorovacie prvky v ceste spätnej väzby. Oneskorenie musí byť také veľké, aby sa stihli stabilizovať signály stavových premenných na výstupe kombinačnej logiky budúceho stavu. Tomuto režimu sa hovorí fundamentálny režim, pretože dovoľuje, aby sa zmenil len jeden vstupu v čase, keď sú všetky vstupné, vnútorné a výstupné signály stabilné. Ďalší vstupný signál sa môže zmeniť, až keď bude obvod zase stabilný. Najmenší čas medzi dvoma zmenami zodpovedá najväčšiemu oneskoreniu v obvode. V literatúre [2] sa hovorí ešte o fundamentálnom režime zmien viacerých vstupov (multiple-input changes MIC), kde sa môžu zmeniť viaceré vstupy naraz, ale až po stabilizácii obvodu, tzv. burst-mode režim. Návrh týchto obvodov začína tabuľkou prechodov (flow table). Účelom návrhu je odvodiť správne zapojenie, ktoré bude optimalizované podľa návrhových kritérií pre plochu čipu, rýchlosť alebo spotrebu. Postup syntézy je podobný ako u synchrónnych stavových automatov: minimalizácia stavov, priradenie stavov a logická minimalizácia.

Mullerov vstupno-výstupný štýl je zobrazený na Obr. 2.22c. Je vidieť, že spätná väzba je spojená priamo so vstupom vodičmi. Zdroj [2] to označuje aj ako schopnosť korektného chovania obvodu bez ohľadu na oneskorenia hradiel a so zanedbateľnými oneskoreniami vodičov. Zo stabilného stavu je možné zmeniť vstupy. Po zmene výstupov je možné ďalej meniť vstupy bez ohľadu na vnútorné signály obvodu a teda bez ohľadu na to, či sa obvod dostal od poslednej zmeny do stabilného stavu. Jediné obmedzenie alebo predpoklad pre správne fungovanie je popísanie kauzálnych vzťahov medzi zmenami vstupných a výstupných signálov. Preto sa používajú metódy založené na sledovaní, ktoré špecifikujú všetky možné kombinácie vstupných a výstupných signálov. Takouto technikou sú napr. grafy signálových prechodov STG (Signal Transition Graph).

V obidvoch asynchrónnych štýloch musia byť signály platné po celý čas na rozdiel od synchrónnej verzie, preto je nutné predchádzať hazardom. Existuje ešte skupina samospúšťaných stavových automatov, ktoré pracujú synchrónne lokálne, ale z vonkajšieho prostredia nepotrebujú žiadny synchronizačný signál. Ich vnútorná synchronizácia je založená na vlastnostiach kódovania stavov a známosti súčasného a budúceho stavu. Takýmto obvodom sa hovorí aj autosynchrónne.

2.3 Návrhové systémy pre návrh, syntézu, simuláciu a implementáciu asynchrónnych systémov

Jadrom syntézy číslicových systémov je problém súbežnosti operácií čítania a zapisovania premenných a synchronizácie medzi prijímacími a vysielacími príkazmi. Synchrónna logika to rieši jednoducho priradením operácií na sekvenciu globálnych udalostí (hodín) ako napr. pri priradení konfliktných operácií čítania a zápisu. U asynchrónnej logiky pri absencii globálnej časovej referencie sa rieši otázka súbežnosti v celom jej rozsahu. Syntéza asynchrónnej logiky spočíva na metódach súbežných výpočtov [17].

Pre popis asynchrónnych obvodov bolo nutné vyvinúť jazyky umožňujúce popis súbežných procesov a komunikáciu pomocou handshakingu medzi nimi. Pre tento účel boli vytvorené rôzne procesné algebry určené pre popis asynchrónneho hardvéru: CHP (Communicating Hardware Processes, Balsa, Haste, Tangram). Boli založené na podobných princípoch ako štandardné procesné algebry: ACP, CCS, CSP, LOTOS, μCRL.. [42].

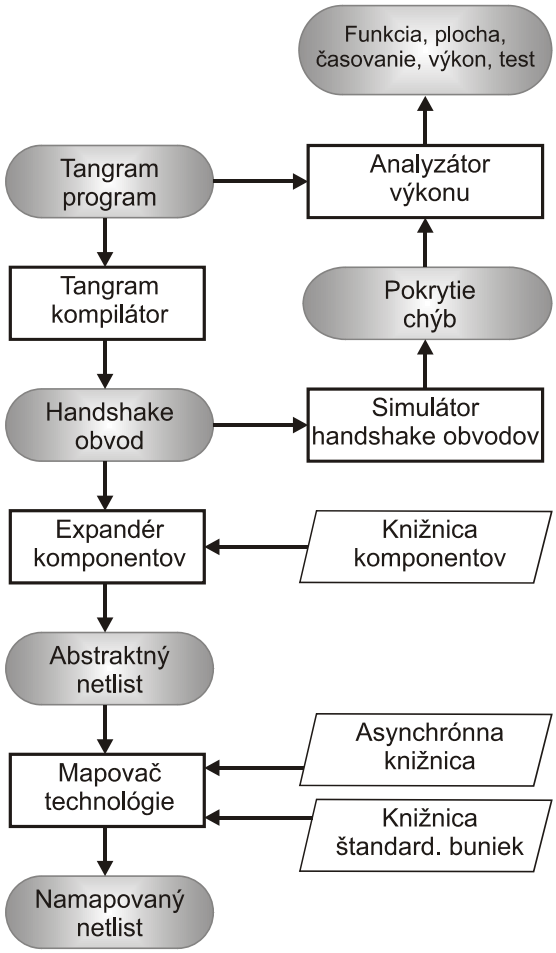
2.3.1. CHP

Jazyk CHP bol inšpirovaný Hoarovým CSP [56]. Tento jazyk je množina $\left\langle C, X, \hat{B}_0 \| ... \| \hat{B}_n \right\rangle$ konštruktov zložená z konečného súboru kanálov C pre handshake komunikáciu, konečného súboru typových premenných X a konečného súboru súbežných procesov $\hat{B}_i$ komunikujúcich prostredníctvom kanálov [17] , [42], [43].

2.3.2. Tangram

Tangram je konvenčný programovací jazyk rozšírený o konštrukty pre popis súbežnosti a komunikácie podobnej jazyku CSP. Navyše sú tu konštrukty pre popis hardvérovo špecifických problémov ako je zdieľanie blokov a čakanie na hranu hodín. Na Obr. 2.23 je zobrazený proces návrhu. Kompilátor preloží program v Tangrame do netlistu handshakingových obvodov, ktorý je zložený z knižnice handshakingových

prvkov. Simulátor handshakingových obvodov a príslušný analyzátor výkonu dávajú spätný obraz návrhárovi o aspektoch ako sú funkcia, zabraná plocha, časovanie, výkon a testabilita. Mapovanie do štandardnej (synchrónnej) knižnice sa deje v dvoch krokoch. V prvom kroku blok „Component expander“ používa knižnicu komponent ku generovaniu abstraktného netlistu skladajúceho sa z kombinačnej logiky, registrov a asynchrónnych buniek ako sú Mullerove elementy C. V druhom kroku štandardné komerčné syntézne nástroje a mapovače technológií generujú netlist zo štandardných buniek. Pri mapovaní nie sú treba asynchrónne bunky, pretože tie sú rozložené v bunkách štandardných knižníc [44].



Obr. 2.23 Návrhový proces pre konštrukcie popísané v jazyku Tangram

Tangram na rozdiel od jazyka CHP je jazyk, ktorý sa podobá skôr tradičnému kompletnému programovaciemu jazyku.

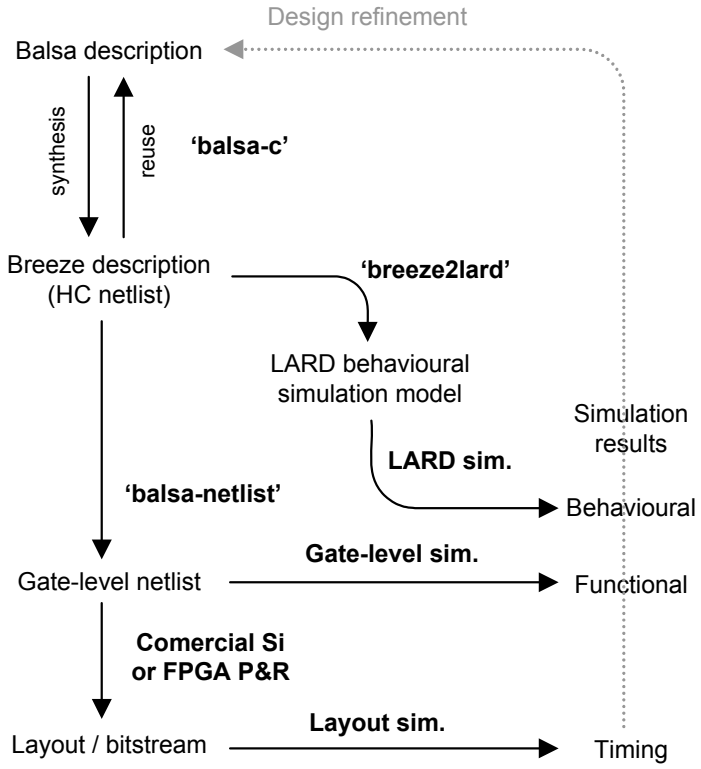
2.3.3. Balsa

Balsa je systém pre popis asynchrónnych obvodov založený na syntaxovo orientovanej kompilácii pre obvody komunikujúce pomocou handshakingu bez nutnosti optimalizácie na hradlovej úrovni. Tento systém pochádza z podobného jazyka Tangram od Philipsu, ale predstavuje viac hrubozrnný prístup k handshake obvodom použitím komponent, ktoré sú parametrizované chovaním. Výhodou tohto jazyka je, že kompilácia je transparentná. Mapovanie jazykových konštruktov prebieha priamočiaro

k výslednej obvodovej implementácii, na rozdiel od VHDL, kde malá zmena kódu dokáže vytvoriť radikálnu zmenu vo výslednom obvode.

Obvod popísaný v Balse je kompilovaný do komunikačnej siete zloženej z malých handshake komponent. Komponenty sú spojené kanálmi, cez ktoré sa uskutočňuje komunikácia alebo handshake. Kanály môžu obsahovať dátový spoj, alebo môžu byť čisto riadiace. Každý kanál spája jeden aktívny (inicializuje komunikáciu) a jeden pasívny (reaguje na komunikáciu) port určitého handshake komponentu [45].

Návrhový proces v Balse je zobrazený na Obr. 2.24 [46]. Nástroje v systéme Balsa pracujú s „Breeze“ handshake prechodnými súbormi. Tieto súbory používajú koncové nástroje pre odvodenie implementácie z popisov v jazyku Balsa. Popis konštrukcie v jazyku Balsa je kompilovaný pomocou kompilátora „balsa-c“. Tento kompilátor vytvára spomínané súbory Breeze. Nástroj „balsa-netlist“ generuje netlist z Breeze súboru vo forméte EDIF, Compass alebo Verilog. Vykonáva mapovanie do technológie a optimalizuje handshaking. Nástroj „breeze2lard“ konvertuje Breeze súbor na behaviorálny model LARD. Ten sa využíva pre behaviorálnu simuláciu. Jazyk LARD sa rovnako používa na popis asynchrónnych obvodov. Nástroj „Breeze-cost“ odhaduje plochu konštrukcie z netlistu súboru Breeze. Na základe netlistu sa v komerčných nástrojoch robí implementácia. Ostatné nástroje systému Balsa slúžia na export do koncových implementačných nástrojov alebo ako grafické výstupy vo formáte postscript [36], [52].

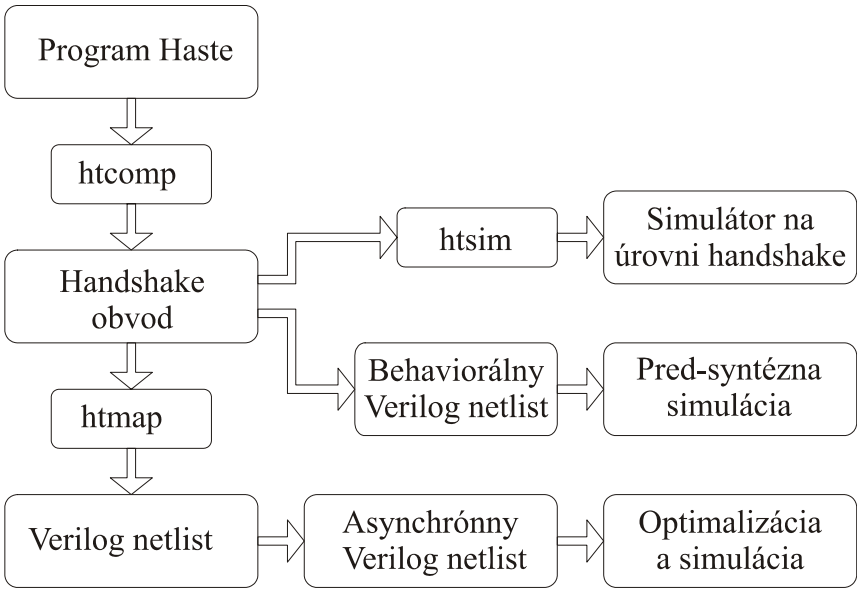


Obr. 2.24 Návrhový proces v systéme Balsa

Balsa bola vyvinutá z jazyka Tangram, je menej prepracovaná, neobsahuje toľko nástrojov. Je určená pre malé a rýchle obvody, ale i zložité obvody sa dajú robiť pomerne jednoducho. Na rozdiel od Tangramu je voľne šíriteľná.

2.3.4. Haste

Jazyk definovaný pre návrh a syntézu asynchrónnych VLSI obvodov. Bol inšpirovaný Hoarovým CSP [56] a Dijkstrovým GCL, odlúčil sa z bývalého Tangramu. Od roku 1998 boli pridané na žiadosť návrhárov do Haste nehandshakingové kanály pre rozhrania do synchrónnych domén. Tento systém obsahuje súbor nástrojov pre kompiláciu, simuláciu a analýzu. Kompilácia je transparentná, návrhár vie približne ako bude vyzerať implementácia v závislosti na použitých jazykových konštruktoch. Proces návrhu je zobrazený na Obr. 2.25. Kompilátor „htcomp“ prekladá zdrojový kód Haste na handshakingový obvod. Nástroj „htsim“ verifikuje funkciu návrhu generovaním abstraktných modelov handshakingových komponentov dovoľujúcich simuláciu na handshakingovej úrovni. Je možné rovnako simulovať aj na behaviorálnej úrovni kompilovaním programu Haste na behaviorálny Verilog a použitím štandardných simulačných nástrojov. Výhodou je použitie rovnakého testbenchu pre predsyntéznu a posyntéznu simuláciu. Simulátor „htsim“ používa odlišný testbench. Nástroj „htmap“ optimalizuje riadenie komunikácie každého handhsakingového kanálu a robí technologické mapovanie z handshakingového obvodu do Verilog netlistu. Ďalej nástroje „htlog“ a „htpost“ optimalizujú logiku a prispôsobujú oneskorovacie prvky, aby sa zhodovali s optimalizovanou logikou. Verilog netlist je následne optimalizovaný externým syntéznym nástrojom spolu so skriptami v predchádzajúcom kroku. Správnosť návrhu je kontrolovaná štandardným simulačným nástrojom [46], [47].



Obr. 2.25 Návrhový proces v systéme Haste

2.3.5. Petriho siete

Petriho sieť je matematická reprezentácia diskrétnych distribuovaných systémov. Petriho sieť graficky reprezentuje štruktúru distribuovaného systému ako orientovaný biparitný graf s ohodnotením. Petriho siete sú nástrojom pre modelovanie a simulovanie diskrétnych systémov charakterizované súbežnými, asynchrónnymi, paralelnými, nedeterministickými a stochastickými vlastnosťami. Je to grafová štruktúra, ktorá obsahuje dva druhy uzlov - miesta a prechody, ďalej obsahuje hrany a značky, ktoré sa

vyskytujú v miestach. Simulácia v Petriho sieti prebieha pomocou premiestňovania značiek z miesta do miesta, ak to dovoľuje prechod [49].

Petriho sieť je pätica $N = (P, T, pre, post, M_0)$, kde

$P = p_1, p_2, ...p_m$ je konečná množina miest,

$T = t_1, t_2, ...t_n$ je konečná množina prechodov,

$pre = P\,x\,T \rightarrow N$ je násobnosť vstupných hrán,

$post = T\,x\,P \rightarrow N$ je násobnosť výstupných hrán,

$M_0 : P \rightarrow N$ je počiatočné označenie.

Grafy signálových prechodov STG (Signal Transition Graphs) používané často pre grafické znázornenie distribuovaných systémov sú vlastne podmnožinou Petriho sietí.

2.3.6. Petrify

Petrify je nástroj pre manipuláciu súbežných špecifikácií, syntézu a optimalizáciu asynchrónnych riadiacich obvodov. Zo zdrojových Petriho sietí PN (Petri Nets), grafov signálových prechodov STG a prechodových systémov TS (Transition Systems) generuje optimalizovaný netlist asynchrónneho kontroléra v cieľovej knižnici hradiel [54].

2.3.7. LARD

LARD je jazyk pre popis hardvéru založený na CSP kanálovej komunikácii vytvorený pre behaviorálne modelovanie asynchrónnych VLSI systémov. Tento jazyk má výrazne abstraktný štýl modelovania, preto je produktívnejší ako klasické jazyky pre popis hardvéru. Pre vysoký stupeň abstrakcie sa používa hlavne na simulácie asynchrónnych obvodov (napr. z jazyka Balsa), ako zdroj pre implementáciu sa nepoužíva [55].

2.3.8. Verilog CSP

Jedná sa o knižnicu makier pre podporu CSP v štandardnom Verilogu. CSP dodáva klasickému HDL jazyku možnosti pre modelovanie asynchrónnych obvodov prostredníctvom komunikácie založenej na kanáloch a súbežnosti [53].

2.3.9. VHDL

Hlavnými vlastnosťami asynchrónnych obvodov je súbežnosť a komunikácia založená na handshake kanáloch. Tieto dve charakteristiky spĺňa jazyk CSP, na ktorom je založená väčšina jazykov pre modelovanie a syntézu asynchrónnych obvodov na vysokej úrovni abstrakcie. Štandardné jazyky pre popis hardvéru ako sú VHDL a Verilog tieto vlastnosti nemajú, aspoň nie na takej vysokej úrovni abstrakcie.

Proces v jazyku CSP sa síce podobá procesu vo VHDL, avšak CSP dovoľuje paralelnú kompozíciu príkazov v procese na rozdiel od VHDL. Vo Verilogu existujú niektoré konštrukty (fork, join), ktoré dovoľujú určitú úroveň súbežnosti v procese. VHDL a Verilog neobsahujú vyššie abstrakcie, ktoré popisujú handshaking protokol

a kódovanie dát. Je tam možné jedine napísať kód na nízkej úrovni, ktorý implementuje handshake. Je dosť nežiadúce kombinovať takéto detaily s kódom, ktorého účel je zachytiť chovanie obvodu na vyššej úrovni abstrakcie.

Pre realizáciu vyššej úrovne abstrakcie je treba rozšíriť VHDL o prvky kanálov ako je tomu v CSP. Realizuje sa to pomocou vytvorenia balíčkov (packages) s prvkami pre synchrónne[1] predávanie správ medzi procesmi a balíčkov kanálov. Vo VHDL prebieha komunikácia medzi procesmi prostredníctvom signálov. Kanály sa preto deklarujú ako signály. Kanál je deklarovaný ako typ „record“ a môže byť popísaný dvoma úrovňami abstrakcie. Prvý je reálny kanál, ktorý obsahuje pole dát a riadiacich signálov (request, acknowledge). K nemu je definovaná rozlišovacia funkcia, ktorá určuje hodnotu kanálu. Takto popísaný kanál je vhodný pre všetky úrovne abstrakcie. Na druhej strane abstraktný kanál obsahuje len dve polia: jedno popisuje stav handshakingu a druhé obsahuje dáta. Je vhodný len pre modely vysokej úrovne abstrakcie.

Na Obr. 2.26 je zobrazený návrhový proces asynchrónnych obvodov pre VHDL. Na začiatku je popísaná konštrukcia pomocou abstraktných kanálov. Je možné overiť funkciu obvodu behaviorálnou simuláciou, ale nie je možné takúto úroveň abstrakcie syntetizovať. Preto sa musí návrh rozložiť na jemnejšiu štruktúru, prvky nižšej úrovne. Tieto prvky je možné potom popísať pre reálny kanál. Ďalej sa prvky rozdelia na riadiace časti a dátové časti. Takto rozdelené prvky sa namapujú a implementujú do cieľovej technológie.

Obr. 2.26 Návrhový proces vo VHDL pre asynchrónne obvody

[1] Synchrónne predávanie správ znamená, že moduly navzájom medzi sebou komunikujúce čakajú vždy na odozvu druhého modulu pred začatím komunikácie.

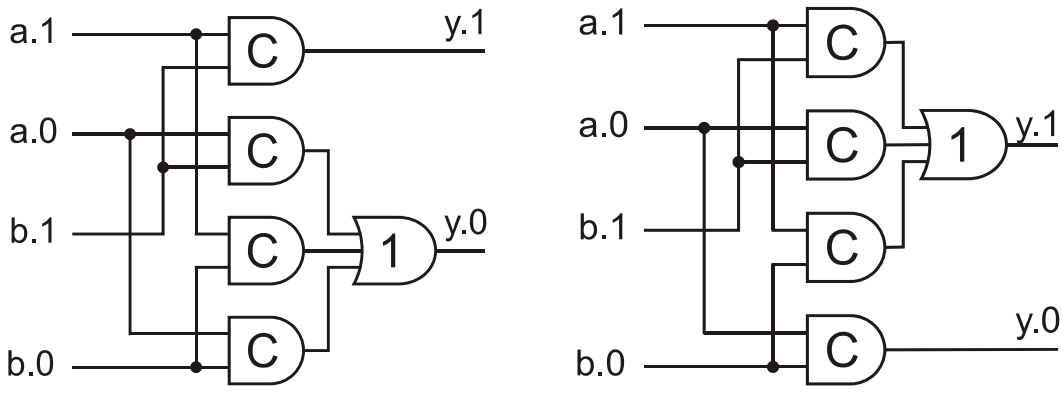
Nevýhodami VHDL sú chýbajúce vstavané prvky pre synchrónne predávanie správ v kanále a súbežnosť na úrovni príkazov v procese. Na druhej strane majú nesporné výhody. Je tu široká podpora súčasnými CAD nástrojmi pre simuláciu, syntézu a rozmiestnenie. Výhodou je rovnaký simulátor a testbench pre rôzne úrovne špecifikácie v procese. Ďalej je tu možnosť zmiešaného modelovania obvodov (napr. behaviorálny popis so štrukturálnym popisom). Takisto je výhodou, že dokážu popísať hybridné systémy (synchrónne + asynchrónne), ktoré sa v dnešnej dobe často objavujú [1].

Návrh asynchrónnych obvodov je možný kombináciou návrhových systémov, ako napríklad konverzia z grafického zobrazenia signálových prechodov STG (Signal Transition Graph) do formy VHDL [50]. Iný prípad je využitie Petriho sietí ako prechodný formát pre reprezentáciu dátovej a riadiacej cesty, ktorý ústí do formátu VHDL a jeho syntézy [51].

2.4 Transformácie sekvenčných systémov

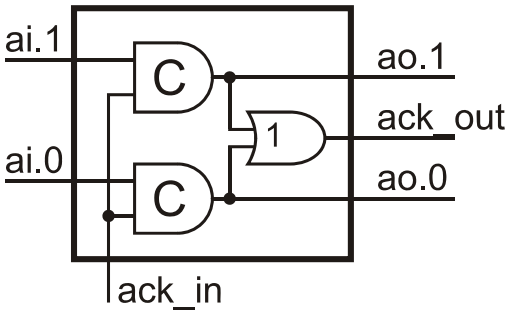
Efektívnou možnosťou, ako možno navrhnúť asynchrónny obvod bez nutnosti preučenia na nový štýl návrhového procesu a návrhových systémov je transformácia obvodu zo známej synchrónnej domény do požadovanej cieľovej asynchrónnej domény. Výhodou je, že metodika návrhu synchrónnych obvodov je zaužívaná a jednoduchá a pre asynchrónne obvody neexistuje žiadny zaužívaný návrhový postup ani návrhové nástroje (okrem pár experimentálnych).

Zdroj [32] popisuje nástroj na konverziu synchrónneho návrhu na asynchrónny použitím štvorfázového protokolu dual-rail. Základom je náhrada komponentov za ich dual-rail ekvivalenty z vhodnej knižnice. Implementácia dvojvstupových hradiel AND a OR je vytvorená pomocou štvorice Mullerových elementov C, ako ukazuje Obr. 2.27. Tie kombinujú jednotlivé vstupy podľa danej logickej funkcie. So zvyšujúcim sa počtom vstupov dual-rail hradla rastie počet Mullerových elementov C geometricky (2^n). Rovnako aj počet vstupov Mullerových elementov C rastie. Jediný prvok invertor má jednoduchú štruktúru spočívajúcu len v prehodení obidvoch dátových vodičov. Nahradzovanie kombinačnej logiky je celkom priamočiare bez žiadnych úvah.



Obr. 2.27 Dual-rail implementácia a.) hradla AND a b.) hradla OR

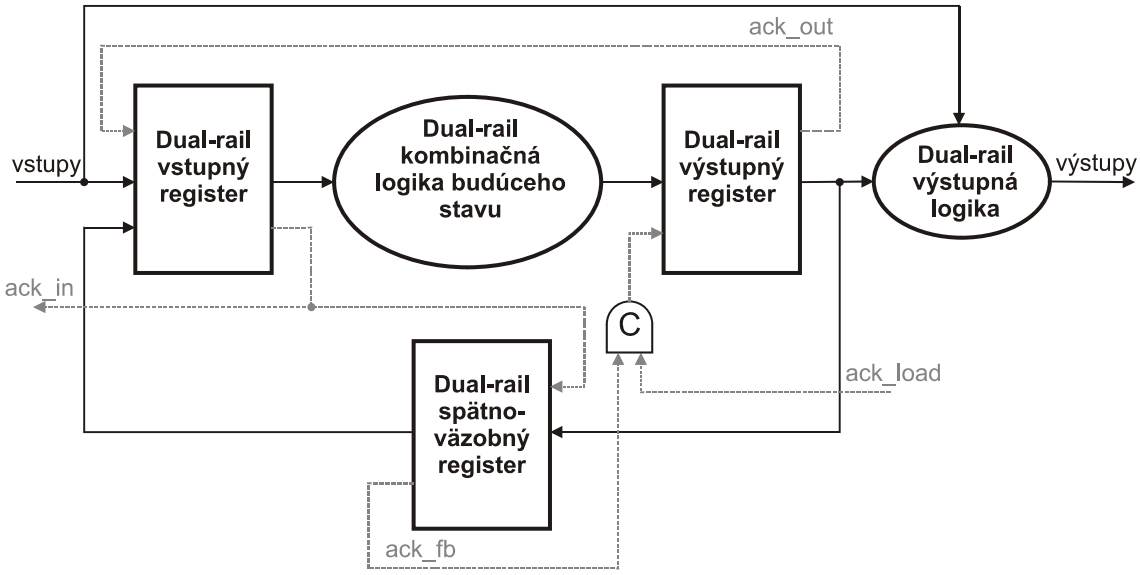
Riadenie je vytvorené pomocou potvrdzovacieho (acknowledgement) signálu, ktorý vysiela každý latch po obdržaní nových dát. Obr. 2.28 znázorňuje zapojenie latchu. Ostatné latche čakajú na potvrdzovací signál pred odstránením dát z ich výstupov. Viaceré potvrdzovacie signály ústiace do jedného bodu (latchu) sa kombinujú pomocou Mullerovho elementu C.



Obr. 2.28 Dual-rail implementácia latchu

Dual-rail latche pracujú v štvorfázovom protokole. Pri neaktívnom potvrdzovacom signále *ack_in* vo vysokej úrovni prijímajú nové dáta a ukladajú ich v Mullerovom elemente C, ktorý je ich základnou stavebnou bunkou. Ak sú dáta prijaté, vysielajú latche potvrdzovací signál *ack_out*.

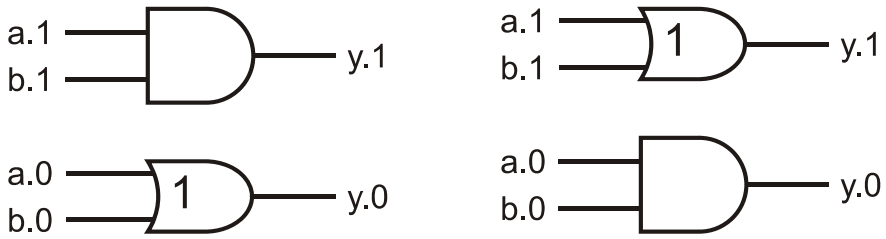
Pri implementácii pipeline musí byť aspoň dvakrát viac latchov ako dátových tokenov, pretože dátové tokeny musia byť vždy oddelené nulovým tokenom. Najjednoduchšie použitie dvoch latchov v slučke niekedy ale nestačí a systém sa môže zablokovať. Preto minimálny počet elementov v slučke sú tri, viď. Obr. 2.29 na ukážke stavového automatu dual-rail. Jeden drží dáta, ďalší drží nulový token a jeden je voľný pre posúvanie ostatných hodnôt dopredu. Na jednej strane malý počet latchov môže spôsobiť zablokovanie systému a na druhej strane nadbytočný počet latchov spôsobí zvýšené oneskorenie a spomalenie systému.



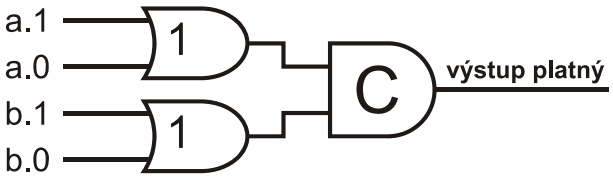
Obr. 2.29 Stavový automat dual-rail

Konštrukcia vytvorená kombináciou horeuvedeného riadenia z dual-rail latchov a dual-rail kombinačnej logiky je značne pomalá. Môže za to zložitosť dual-rail hradiel. Problém je v tom, že sa čaká na to, keď budú všetky vstupy platné. Niekedy ale nie je potrebné čakať na platnosť všetkých vstupov. Zlepšenie môže byť vytvorené separáciou riadiacich a logických častí hradiel, a tak docieliť menšiu a rýchlejšiu konštrukciu. Takéto hradlá sa nazývajú nechránené, pretože neobsahujú riadenie, viď. Obr. 2.30. To je riešené pridaním tretieho signálu platnosti, ktorý signalizuje kompletnú platnosť

vstupov, viď. Obr. 2.31. Takto riešené hradlo je dvakrát menšie a rýchlejšie oproti svojmu predchodcovi.



Obr. 2.30 Implementácia nechráneného hradla a.) AND a b.) OR



Obr. 2.31 Obvod pre signalizáciu platnosti vstupov hradla

Pri implementácii takýchto nechránených hradiel znamená príchod potvrdenia od latchu, že iba logická časť operácie bola kompletná. Niektoré zo vstupov ešte nemuseli byť platné, preto musí obvod čakať na platnosť všetkých vstupov pred vyslaním potvrdenia. Preto musia mať latche ešte signalizačný výstup, ktorý oznamuje dáta na výstupe latchu.

Implementácia veľkých logických blokov s viacerými vstupmi a výstupmi pomocou tohto postupu využíva veľký počet viacvstupových Mullerových elementov C, čo spôsobuje, že obvod môže byť príliš pomalý a má značnú spotrebu. Konvertovať pomocou tohto postupu nejde trojstavový opakovač (buffer) a už existujúce asynchrónne obvody (RS klopný obvod).

Ďalšiu podobnú možnosť transformácie popisuje [39] zo synchrónnej domény do asynchrónnej dual-rail za pomoci štandardných syntéznych nástrojov a jednoduchého transformačného skriptu. Podstatou transformácie je rovnako nahradenie kritických synchrónnych častí za ich asynchrónne náhrady. Pokiaľ budú všetky registre v asynchrónnom obvode pripojené na ten istý Mullerov element C, bude existovať jednotný referenčný bod pre celú konštrukciu. To je vlastne ekvivalencia s hodinovým signálom v synchrónnych systémoch. Tento spoločný mechanizmus umožňuje spomínanú transformáciu. Táto transformácia sa netýka bundled-data asynchrónnych systémov, pretože nie je možné navrhnúť procesne invariantný prvok prispôsobeného oneskorenia.

Štandardný synchrónny syntézny nástroj vytvorí netlist na úrovni hradiel na základe popisu RTL. Transformačný skript nahradzuje štandardný typ std_logic za ekvivalentný typ z vytvorenej knižnice pre asynchrónne obvody dual-rail vo VHDL. Modely hradiel v tejto knižnici sú syntetizované za pomoci štandardných technologických knižníc. Syntetizované asynchrónne hradlá sú uložené ako VHDL netlist. Pre transformáciu je nutné ešte syntetizovať navyše Mullerov element C za pomoci štandardných syntéznych nástrojov.

Zdroj [25] zavádza pojem desynchronizácia. Je to proces včleňovania asynchrónnych záležitostí do tradičného návrhového postupu bez preškolenia zo

synchrónneho rozmýšľania a nutnosti nových nástrojov. Tu sa snažia predstaviť plne automatický proces, ktorý nezmení úplne štruktúru synchrónnej dátovej cesty ani implementáciu riadenia, ale pôsobí skôr na synchronizačnú sieť.

Uvedený desynchronizačný model stavia na substitúcii globálnych hodín asynchrónnymi kontrolérmi, ktoré zabezpečujú ekvivalentné chovanie. Predpokladá sa obvod s kombinačnou logikou a registrami implementovanými ako klopné obvody typu D reagujúce všetky na spoločnú hranu hodín.

Postup desynchronizácie:

- Rozloženie klopných obvodov na master a slave latche, rozpojenie lokálnych hodín pre master a slave latch, prípadne optimalizovať výkon premiestňovaním latchov cez kombinačnú logiku.
- Vytvorenie prispôsobených oneskorení pre riadiacu logiku. Každé prispôsobené oneskorenie musí byť väčšie alebo rovné ako oneskorenie kritickej cesty v príslušnom kombinačnom bloku. Každé prispôsobené oneskorenie slúži ako kompletovací detektor pre kombinačnú logiku.
- Implementácia lokálnych kontrolérov pomocou značkových grafov.

Návrh začína pomocou klasickej HDL špecifikácie použitím klasických synchrónnych konštrukcií. Potom je každý prvok v dátovej ceste syntetizovaný pre cieľový čas cyklu použitím klasických syntéznych nástrojov. Pomocou klasickej statickej časovej analýzy sú odhadnuté oneskorenia pre každý prvok prispôsobeného oneskorenia, ktoré je implementované v kontroléri latchu. Bloky kontrolérov a dátovej cesty sa spoja do kompletného desynchronizovaného netlistu.

Transformácie podľa [32], [39] spočívali v náhrade synchrónnych komponent za asynchrónne na hradlovej úrovni dual-rail. Asynchrónne komponenty sa syntetizovali v štandardných syntéznych nástrojoch zo štandardných synchrónnych knižníc. Aj keď bola transformácia priamočiara a obvody si samé prispôsobovali rýchlosť podľa aktuálneho oneskorenia logickej cesty, boli tieto implementácie, čo sa plochy a spotreby týka, rozsiahle. Článok [25] priniesol desynchronizačný postup, ktorý automaticky nahradzuje hodinovú sieť sieťou asynchrónnych kontrolérov. Nevýhodou ale zostává už princíp tejto implementácie : bundled-data, ktorá uvažuje s pevnými odhadovanými oneskoreniami kombinačnej logiky pred výrobou, ako s reálnymi oneskoreniami meranými za behu (dual-rail). Tieto pevné, prispôsobené oneskorenia je zložité implementovať kvôli variáciám technologických parametrov pri výrobe. Reálnejšia cesta k lepším vlastnostiam obvodov vedie pravdepodobne kompromisom medzi asynchrónnymi a synchrónnymi systémami (GALS, autosynchrónne systémy, EAIC), pokiaľ sa nevyvinie komplexnejší postup a nástroje pre návrh asynchrónnych obvodov.

3 Ciele dizertácie

Predchádzajúce kapitoly priniesli prehľad o spôsoboch riešenia nevýhod, ktoré v poslednej dobe postihujú synchrónne číslicové systémy. Sú to hlavne distribúcia hodinového signálu, prúdová spotreba hodinového signálu a elektromagnetické rušenie číslicových synchrónnych systémov.

Prúdová spotreba bola optimalizovaná hradlovaním hodinového signálu. Elektromagnetické rušenie bolo optimalizované fázovaním hodinového signálu, tzv. umelým zvyšovaním „skew“. Inými systémovými riešeniami pre zníženie prúdovej spotreby a problémov s distribúciou hodinového signálu bolo rozdelenie systému na menšie synchrónne bloky komunikujúce medzi sebou asynchrónne – systém GALS. Takisto aj tieto bloky v systéme GALS museli byť taktované externe buď zo spoločnej alebo z viacero hodinových domén. Zlepšenie ponúkajú systémy EAIC, kde si jednotlivé bloky generujú vlastné hodiny. Základným prvkom pre synchronizáciu generovania hodín tu je prvok DFLOP, ktorý je zložitý vďaka svojej zmiešanej číslicovo-analógovej štruktúre.

Na podobnom, ale jednoduchšom princípe bude v práci vytvorený autosynchrónny stavový automat. Na rozdiel od systému EAIC tu bude vytvorená redundantná logika okolo klopného obvodu, ktorá detekuje zmenu stavu.

Samotný návrh takýchto systémov s rozdielnym synchronizačným princípom je zložitý a nie sú preň dostatočne vyvinuté návrhové nástroje. Preto je efektívnejšou cestou, kvôli jednoduchosti návrhu synchrónneho systému, transformovať synchrónny systém s rovnakou funkčnosťou na autosynchrónny.

Ciele dizertácie by sa dali zhrnúť do nasledujúcich bodov:

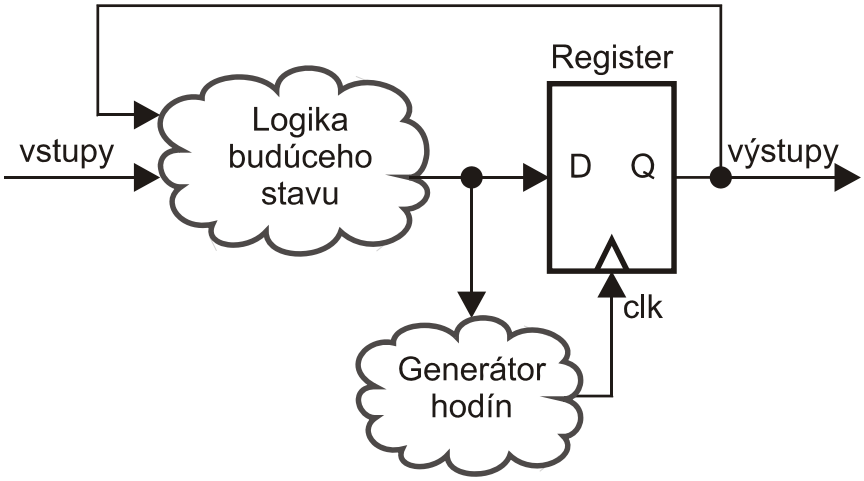
- analýza autosynchrónnych systémov, ich štruktúry, vymedzenie ich vlastností a parametrov.
- vytvorenie metodiky pre transformáciou zo synchrónnych systémov na autosynchrónne pri minimálnych zmenách.

4 Vlastnosti autosynchrónnych obvodov

V tejto kapitole bude na začiatku uvedená definícia autosynchrónnych obvodov, na základe ktorej budú stanovené ich vlastnosti a parametre. Na konci kapitoly budú analyzované časové parametre.

4.1 Autosynchrónne obvody

Autosynchrónne sekvenčné obvody patria do malej podskupiny pod asynchrónne sekvenčné obvody. Preto je aj ich hlavnou vlastnosťou schopnosť samostatného riadenia bez globálnej hodinovej autority. Z vnútra sa tieto obvody ale tvária ako synchrónne, pretože obsahujú stavový register riadený hodinami. Princíp funkcie vyplýva z Obr. 4.1. Obvod pracuje podobne ako štandardný synchrónny stavový automat, ale hodinový signál si generuje sám na základe informácie o ustálení budúceho stavu. K tomuto je nutná prídavná logika, ktorá zisťuje ustálený súčasný stav a zároveň generuje hodinový impulz definovanej dĺžky pre stavový register.



Obr. 4.1 Bloková schéma autosynchrónneho stavového automatu

4.2 Návrh autosynchrónnych stavových automatov

4.2.1. Postup návrhu a kódovanie

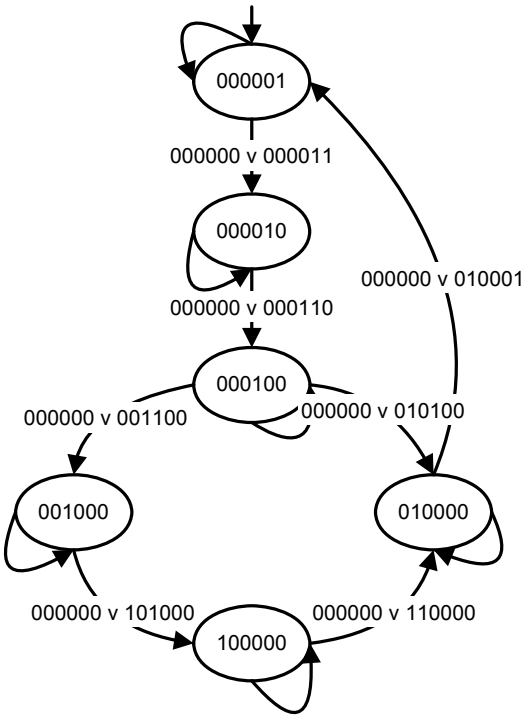
Autosynchrónne stavové automaty nie sú z veľkej časti odlišné od synchrónnych stavových automatov, preto majú časť postupu návrhu rovnakú. Zdroj [4] uvádza návrh automatu v týchto bodoch:

1. Vytvorenie stavovej tabuľky alebo stavového diagramu.
2. Minimalizovanie stavov.
3. Priradenie kódovania stavov.
4. Vytvorenie tabuľky prechodov.
5. Výber druhu klopného obvodu.

6. Vytvorenie excitačnej tabuľky s hodnotami pre požadovaný ďalší stav.
7. Odvodenie vstupných rovníc klopného obvodu z excitačných tabuliek.
8. Odvodenie výstupných rovníc obvodu.
9. Nakreslenie schematického zapojenia.
10. Prekreslenie schémy do určenej technológie.

Pre návrh autosynchrónneho obvodu je rozhodujúci bod 3. Ostatné body zostávajú rovnaké ako u synchrónnej verzie automatu. Navyše je ešte nutné pridať rozhodovaciu logiku, ktorá zisťuje ustálený stav a generátor lokálnych hodín, ktoré budú popísané v ďalších odstavcoch. Zdroj [9] udáva tri rôzne smery pre priradenie stavov. Prvým je minimálna bitová zmena, ktorej vyhovujú Grayov a Johnsonov kód. Druhým smerom je prioritné susedstvo stavov v priradení Karnaughovej mapy. Tretím smerom je kódovanie 1 z N, kde je pre každý stav jeden klopný obvod. Zároveň je treba brať do úvahy spôsob priradenia, ktoré sa vyhýba hazardom a kritickým súbehom, ako to je rozpracované v [5].

Na základe týchto poznatkov sa javili ako najvhodnejšie kódovanie 1 z N a Grayove kódovanie. Aby bolo možné vybrať tieto kódovania, bolo treba analyzovať ich na jednoduchom stavovom automate. Obr. 4.2 zobrazuje kostru jednoduchého šesť-stavového automatu s kódovaním 1 z N, ktorý bol vytvorený len pre názornú ukážku bez konkrétneho aplikačného účelu. Medzi stavmi sú zobrazené dve jediné možnosti prechodu cez medzistav. V reálnom prostredí sa totiž signály nemenia ideálne súčasne, ale je medzi nimi vždy určitý minimálny časový interval. Preto sa pri tomto kódovaní prechádza cez medzistav pri prechode medzi dvoma stavmi. Ako je vidieť z Obr. 4.2, tento medzistav je známy a má len dve možnosti. Buď obsahuje samé nuly alebo dve jedničky.



Obr. 4.2 Stavový diagram automatu s kódovaním 1 z N a zobrazenými medzistavmi

Z tejto znalosti môžeme jednoducho určiť, kedy sa nachádza stavový automat v pracovnom stave pomocou vzájomného sčítania modulo 2 jednotlivých bitov stavového vektoru budúceho stavu funkciou XOR:

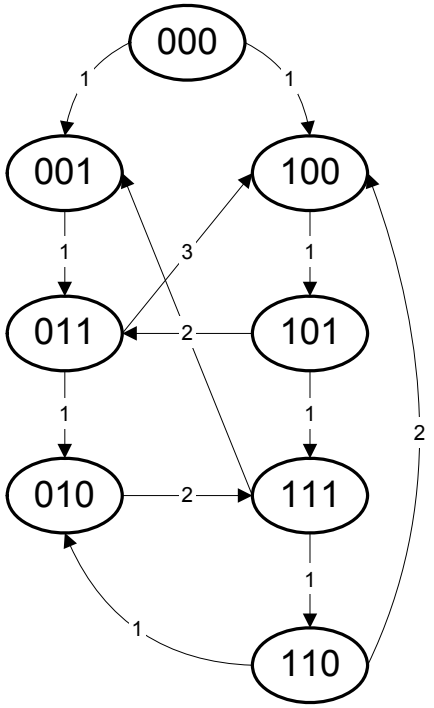
$$y = D(0) \oplus D(1) \oplus .. \oplus D(n-1), \tag{4.1}$$

kde D je stavový vektor budúceho stavu a n je bitová šírka stavového vektora. Táto funkcia by mala byť nulová v prípade prítomnosti medzistavu, inak by mala mať hodnotu jedna. Neurčitosť prechodu medzi dvoma možnosťami prechodného medzistavu eliminujú v [3] nadbytočným termom v rovnici budúceho stavu (vo vzťahu druhý člen):

$$Y_j = \sum_{k=0}^{m-1} y_k . f_{j \leftarrow k} + y_j . F_j , \tag{4.2}$$

kde F_j je Booleovská suma všetkých aktívnych premenných y v stavoch, do ktorých j-tý stav prechádza, m je počet funkcií nasledujúceho stavu.

Ešte lepšie vlastnosti má Grayove kódovanie, kde sa medzi susednými stavmi mení len jeden bit a nedochádza k parazitným impulzom na výstupe kombinačnej logiky. Ďalšou výhodou je to, že pri ideálnom priradení stavov neexistujú nevyužité stavy, ako to je u kódovania 1 z N. Ale pri zložitejších stavových diagramoch so zložitejšími prechodmi nie je možné priradiť Grayov kód tak, aby sa medzi susednými stavmi menila len jedna stavová premenná. Tento prípad je vidieť na Obr. 4.3, kde je zobrazená kostra stavového automatu – detektoru postupnosti. Čísla na hranách prechodov zobrazujú počet bitových zmien v stavovom vektore pri prechode do ďalšieho stavu.



Obr. 4.3 Počet zmien bitov na hranách v stavovom diagrame automatu s Grayovým kódovaním

Takéto priradenie je nevhodné, pretože v kombinačnej logike budúceho stavu vznikajú kritické súbehy a parazitné impulzy.

Aj keď nastávajú pri kódovaní 1 z N hazardy v kombinačnej logike budúceho stavu, dajú sa ľahko identifikovať a určiť kedy je stav ustálený. Návrh kombinačnej logiky je jednoduchý. Grayove kódovanie má výhodnejšie vlastnosti, ale v zložitejších prípadoch nejde zakódovať, tak aby bola medzi susednými stavmi zmena len jedného bitu. Preto sa pre praktické aplikácie hodí viac kódovanie 1 z N.

4.2.2. Detekcia ustáleného stavu

Ako bolo ukázané v predchádzajúcej analýze, detekcia ustáleného stavu je vhodná hlavne pre Grayove kódovanie a kódovanie 1 z N. Princípom detekcie ustáleného stavu je zisťovanie, či sa zmenil stavový vektor budúceho stavu na základe porovnania so stavovým vektorom súčasného stavu. Predpokladá sa stavový automat so všetkými stabilnými stavmi, preto je zmena stavového vektoru budúceho stavu stimulovaná len zmenou vstupných signálov.

Pre detekciu ustáleného stavu u obidvoch kódovaní (1 z N, Gray) sa používa jednoduché porovnanie súčasného a budúceho stavu logickým súčtom jednotlivých bitových súčtov modulo 2 stavových vektorov súčasného a budúceho stavu pomocou funkcie XOR:

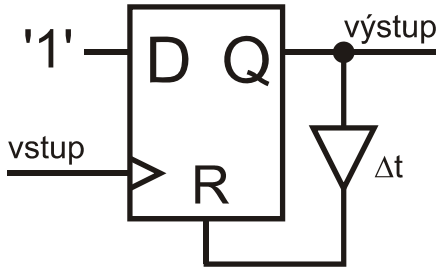
$$y = \mathrm{V}_{i=0}^{n-1} D(i) \oplus Q(i) . \tag{4.3}$$

U kódovania 1 z N je treba ešte kontrolovať moment prechodu cez medzistav, signalizovaný funkciou (4.1). Logický súčin týchto dvoch funkcií (4.1) a (4.3) určuje vhodný moment pre spustenie hodinového impulzu.

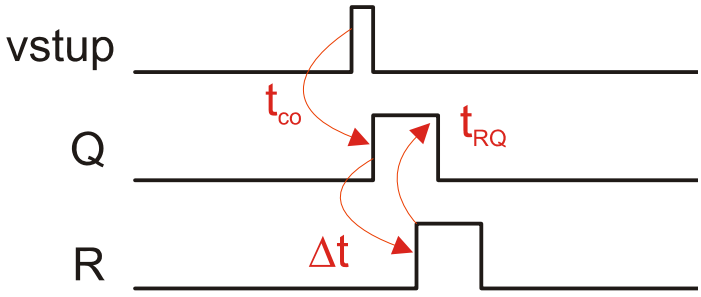
Existuje veľa ďalších možností, ako zistiť moment preklopenia klopného obvodu. Okrem základných diskrétnych obšírnych možností ako sú globálne hodiny u synchrónnych systémov a prispôsobené oneskorenie u asynchrónnych obvodov je možné použiť logiku NCL [6]. Systémy EAIC [19] používajú špeciálne klopné obvody, ktoré majú synchronizačný výstupný signál pre moment ich preklopenia.

4.2.3. Generátor hodín

Ďalším pridaným prvkom v autosynchrónnom obvode oproti synchrónnym obvodom je generátor lokálnych hodín. Jeho úlohou je generovanie impulzov definovanej šírky na základe podnetov z detektora ustáleného stavu. Generátor pracuje podobne ako monostabilný klopný obvod v analógovej technike. Obsahuje hranou riadený klopný obvod D s asynchrónnym nulovaním aktívnym vo vysokej úrovni H, ktorého výstup je cez oneskorovaciu linku privedený na spomínané asynchrónne nulovanie. Obrázok zapojenia je zobrazený na Obr. 4.4. Dĺžka generovaného impulzu je závislá na oneskorovacej ceste k nulovaciemu vstupu Δt a oneskorení výstupu klopného obvodu na základe aktivovaného nulovania t_{RQ}, ako je vidieť z Obr. 4.5.



Obr. 4.4 Schéma generátora hodín

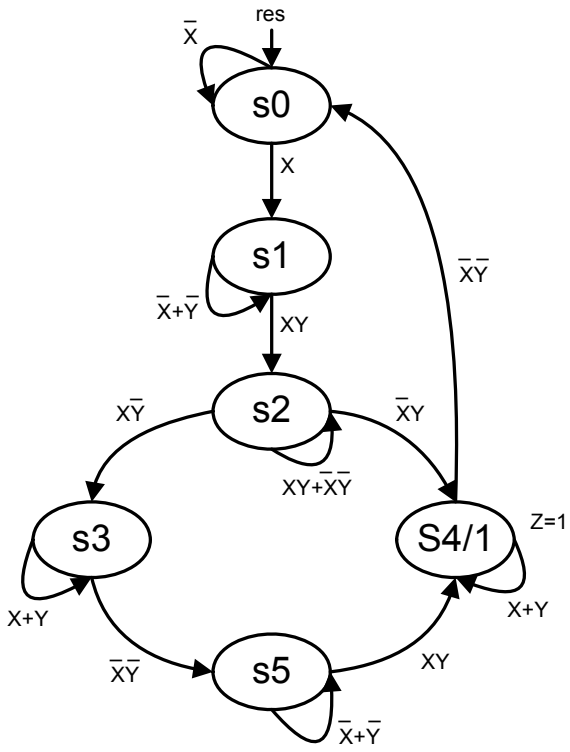


Obr. 4.5 Časový diagram generátora hodín

Pokiaľ nie je nevyhnutné generovať hodinové impulzy konštantnej šírky, je možné generátor vynechať a spoľahnúť sa na impulzy z detektora ustáleného stavu. Z pohľadu časovania sa tým skráti oneskorenie hodinového impulzu o oneskorenie t_{co} klopného obvodu. Okrem toho odpadá nutnosť implementovať oneskorovací prvok pre nulovací vstup. Preto z tohto pohľadu nebude ďalej tento prvok použitý. Pojmom generátor hodín bude ďalej uvažovaný kombinačný obvod, ktorý na základe vstupov z detektora ustáleného stavu a detektora rozdielnych stavov vytvorí hodinový signál.

4.2.4. Stavový automat s kódovaním 1 z N

Pre zistenie a odvodenie vlastností a parametrov autosynchrónneho obvodu s kódovaním 1 z N bola vytvorená ukážka návrhu jednoduchého šesť-stavového automatu bez konkrétneho aplikačného účelu, ktorého stavový diagram je na Obr. 4.6. Automat má dva vstupy a jeden výstup, ktorý má vysokú úroveň v stave *s4*, inak je v nízkej úrovni. Zo stavového diagramu bola prekreslená tabuľka stavov Tab. 4.1, kde bolo symbolickému vyjadreniu stavov priradené kódovanie 1 z N. Šedé polia označujú stabilné stavy.



Obr. 4.6 Stavový diagram automatu s kódovaním 1 z N

Tab. 4.1 Tabuľka stavov (kódovanie 1 z N)

Stav	XY				Z
	00	01	11	10	
000001 s0	s0	s0	s1	s1	0
000010 s1	s1	s1	s2	s1	0
000100 s2	s2	s4	s2	s3	0
001000 s3	s5	s3	s3	s3	0
010000 s4	s0	s4	s4	s4	1
100000 s5	s5	s5	s4	s5	0

Skombinovaním excitačných výrazov pre klopný obvod typu D s tabuľkou stavov vznikli nasledujúce logické výrazy pre vstupy klopného obvodu z kombinačnej logiky budúceho stavu a pre výstupnú kombinačnú logiku:

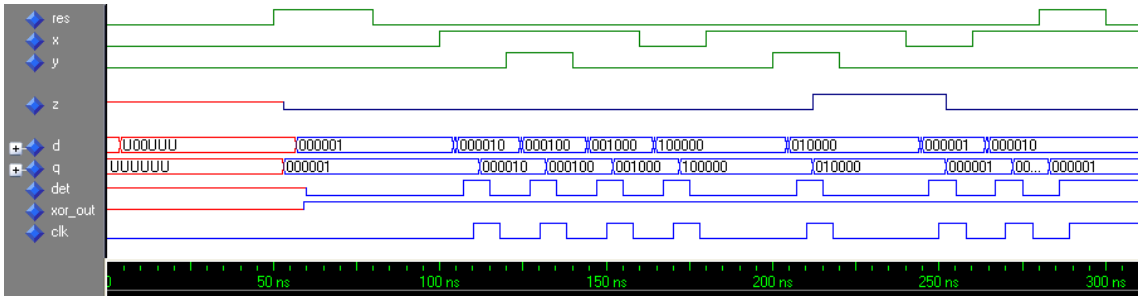
$$\begin{aligned} D_0 &= Q_0\overline{X} + Q_4\overline{X}\overline{Y}, \\ D_1 &= Q_0X + Q_1(\overline{X} + \overline{Y}), \\ D_2 &= Q_1XY + Q_2(XY + \overline{X}\overline{Y}), \\ D_3 &= Q_2X\overline{Y} + Q_3(X + Y), \\ D_4 &= Q_2\overline{X}Y + Q_5XY + Q_4(X + Y), \\ D_5 &= Q_3\overline{X}\overline{Y} + Q_5(\overline{X} + \overline{Y}), \\ Z &= Q_4. \end{aligned} \tag{4.4}$$

Symboly Q_0 až Q_5 tvoria bity stavového vektoru súčasného stavu a symboly D_0 až D_5 označujú stavový vektor budúceho stavu. Logické výrazy (4.4) boli zapísané v simulačnej syntaxi jazyka VHDL s definovaným oneskorením pre operácie AND, OR a INVERT. Synchrónna časť je rovnaká ako u synchrónnych stavových automatoch. Hodinový signál do tejto logiky však nie je privedený zo vstupu, ale ako vnútorný signál, ktorý generuje detektor ustáleného stavu. Ten je vytvorený na základe výrazov (4.1) a (4.3), kde musí spĺňať obidve podmienky naraz, a to určenie ustáleného stavu a zistenie rozdielnosti stavov. Popísaný obvod vo VHDL bol odsimulovaný vo funkčnej simulácii v nástroji Modelsim.

Simulačné priebehy sú vidieť na Obr. 4.7. V diagrame sú zvrchu zobrazené asynchrónny vstup nulovania *res* a dva vstupy *x* a *y* (zelená farba). Výstup *z* je zobrazený pod nimi tmavou farbou. V spodnej časti diagramu sú modrou farbou zobrazené vnútorné signály: budúci stav *d*, súčasný stav *q*, sledovanie rozdielnosti súčasného a budúceho stavu *det*, snímanie ustáleného stavu *xor_out* a lokálne generovaný hodinový signál *clk* z generátora hodín.

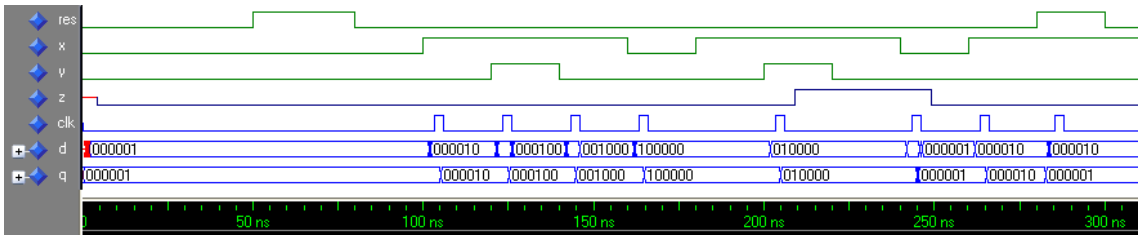
Spustenie automatu začína asynchrónnym nulovaním pri *res='1'*, nasleduje prechod do východiskového stavu. Ako je vidieť pred prvým impulzom hodín, po zmene vstupov sa ustáli výstup kombinačnej logiky budúceho stavu *d*, signál *det* signalizuje rozdiel v stavoch. Signál *xor_out* by mal generovať úzke impulzy pri ustálení stavu, ale kvôli zotrvačnosti tejto funkcie a rýchlosti ustálenia kombinačnej logiky budúceho stavu sú tieto impulzy filtrované. Hodinový signál *clk* je vytvorený ako funkcia, ktorá sleduje obdive predchádzajúce podmienky, teda *det* v logickej úrovni H a *xor_out* v logickej úrovni H. V tomto prípade generuje hornú úroveň hodinového signálu *clk*. Šírka impulzu *clk* je závislá na oneskorení kombinačnej logiky. Hodinovým

signálom *clk* sa riadi celý synchrónny proces automatu a preklápa budúci stav *d* do súčasného stavu *q*.



Obr. 4.7 Behaviorálna simulácia automatu s kódovaním 1 z N so simulačnými modelmi hradiel

Táto simulácia brala do úvahy len diskrétne oneskorenia hradiel zadané pomocou simulačnej direktívy „after“ jazyka VHDL. Pre overenie vlastností automatu v reálnom prostredí s reálnymi parametrami ako sú oneskorenia vo vodičoch boli odstránené zo zápisu VHDL oneskorovacie direktívy a bola spustená post-route simulácia. Boli použité knižnice hradiel pre obvod FPGA Spartan3 (XC3S200-5). Simulačný diagram je vidieť na Obr. 4.8.



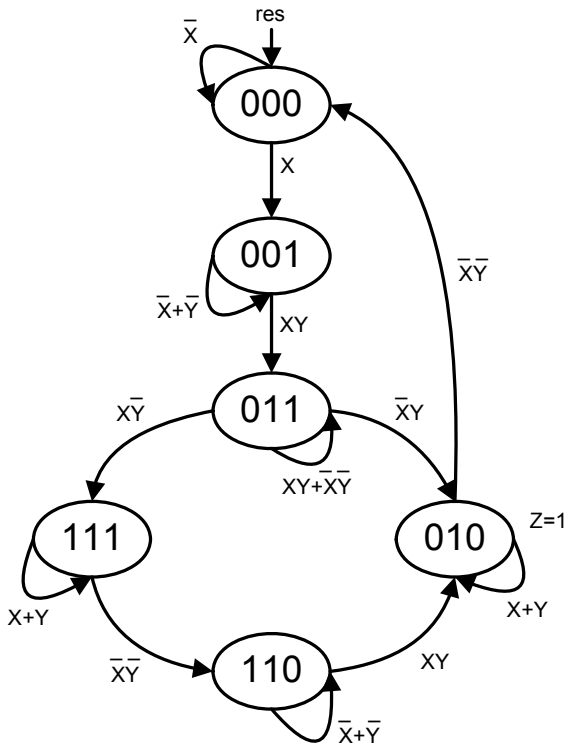
Obr. 4.8 Post-route simulácia automatu s kódovaním 1 z N s hradlami z knižnice FPGA Spartan3

Ako je vidieť z Obr. 4.8, simulácia je podobná ako behaviorálna simulácia. Automat vždy reaguje až na zmenu vstupov, pretože každý stav je stabilný. Časové otázky automatu sú preberané neskoršie v kapitole 4.3.

4.2.5. Stavový automat s Grayovým[2] kódovaním

Pre odvodenie vlastností a parametrov aj pre druhú variantu stavového automatu s Grayovým kódovaním, je ukázaný návrh rovnakého automatu v tomto kódovaní. Štruktúra stavového diagramu na Obr. 4.9 je rovnaká ako u predchádzajúceho stavového automatu s kódovaním 1 z N na Obr. 4.6. Obsahuje však už namiesto symbolického označenia stavov konkrétne Grayovo kódovanie. V tomto prípade je automat zakódovaný tak, že medzi všetkými susednými stavmi je zmena len 1 bitu stavového vektoru.

[2] Pojem Grayove kódovanie sa vzťahuje na zmenu práve jedného bitu v stavovom vektore medzi dvoma susednými stavmi. Z tabuľky stavov nie je táto skutočnosť zrejmá, ale zo stavového diagramu vyplýva, že pre susedné stavy je toto pravidlo platné.



Obr. 4.9 Stavový diagram automatu s Grayovým kódovaním

Z diagramu je vytvorená tabuľka stavov Tab. 4.2. Je vidieť, že tu sú treba len tri stavové bity na zakódovanie stavov. Logické rovnice obsahujú päť vstupov Q_2, Q_1, Q_0, X a Y.

Tab. 4.2 Tabuľka stavov s Grayovým[2] kódovaním

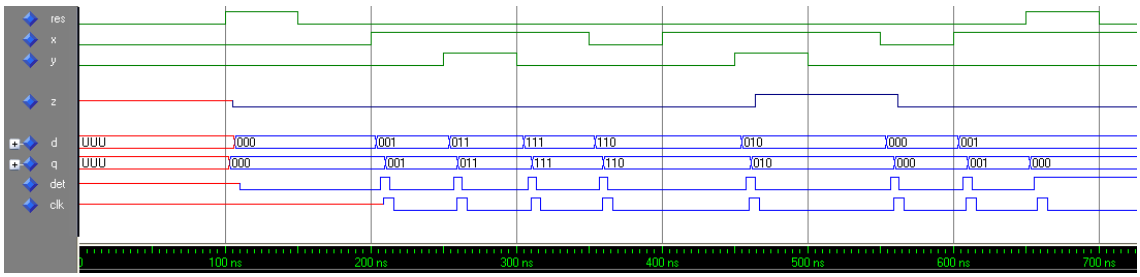
Stav	XY				Z
	00	01	11	10	
000 s0	s0	s0	s1	s1	0
001 s1	s1	s1	s2	s1	0
011 s2	s2	s4	s2	s3	0
111 s3	s5	s3	s3	s3	0
010 s4	s0	s4	s4	s4	1
110 s5	s5	s5	s4	s5	0

Po skombinovaní s excitačnou tabuľkou klopného obvodu D a úprave dostávame logické výrazy pre kombinačnú a výstupnú logiku:

$$\begin{aligned} D_0 &= \overline{Q}_2\overline{Q}_1\overline{Q}_0X + \overline{Q}_2\overline{Q}_1Q_0 + \overline{Q}_2Q_1Q_0X + \overline{Q}_2Q_1Q_0\overline{Y} + Q_2Q_1Q_0X + Q_2Q_1Q_0Y, \\ D_1 &= \overline{Q}_2\overline{Q}_1Q_0XY + Q_1Q_0 + \overline{Q}_2Q_1\overline{Q}_0X + \overline{Q}_2Q_1\overline{Q}_0Y + Q_2Q_1\overline{Q}_0, \\ D_2 &= \overline{Q}_2Q_1Q_0X\overline{Y} + Q_2Q_1Q_0 + Q_2Q_1\overline{Q}_0\overline{X} + Q_2Q_1\overline{Q}_0\overline{Y}, \\ Z &= \overline{Q}_2Q_1Q. \end{aligned} \tag{4.5}$$

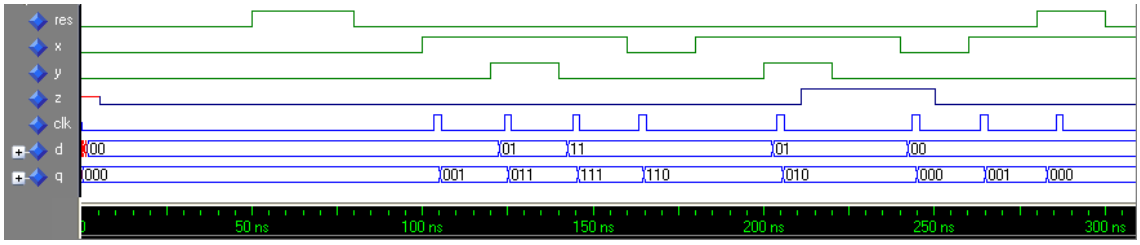
Rovnako ako v predchádzajúcom prípade bol popísaný model vo VHDL a odsimulovaný v behaviorálnej simulácii na Obr. 4.10. Ako je vidieť z obrázku, v kombinačnej logike budúceho stavu neprichádza ku zákmitom, mení sa len jeden bit

pri prechode stavu. Preto je použitý na detekciu ďalšieho stavu len signál *det*, ktorý signalizuje rozdiel medzi súčasným *q* a budúcim stavom *d*. Signál *det* spúšťa generátor hodín s výstupným signálom *clk*.



Obr. 4.10 Behaviorálna simulácia automatu s Grayovým kódovaním so simulačnými modelmi hradiel

Pre overenie reálnych podmienok bola rovnako vytvorená post-route simulácia pre hradlá z knižnice FPGA Spartan3 na Obr. 4.11.

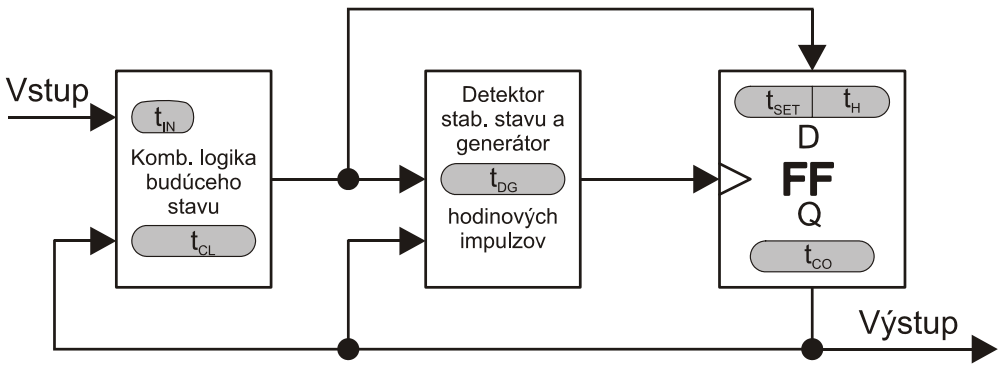


Obr. 4.11 Post-route simulácia automatu s Grayovým kódovaním s hradlami z knižníc FPGA Spartan3

Táto podkapitola analyzovala na začiatku možnosti využiť kódovanie stavov stavových automatov k osamostatneniu ich riadenia prostredníctvom pridanej kombinačnej logiky pre generovanie lokálneho hodinového signálu. Na základe tejto analýzy bola vytvorená metodika postupu návrhu na dvoch možných príkladoch kódovania 1 z N a Grayovho kódovania stavového automatu. Tieto konštrukcie boli odsimulované v behaviorálnej simulácii i v post-route simulácii. Na základe týchto časových simulácii môžeme stanoviť základné parametre a vlastnosti týchto obvodov v nasledujúcej podkapitole.

4.3 Časové vlastnosti autosynchrónnych stavových automatov

Na základe predchádzajúcich simulácií bol vytvorený časový model autosynchrónneho stavového automatu zobrazený na Obr. 4.12.

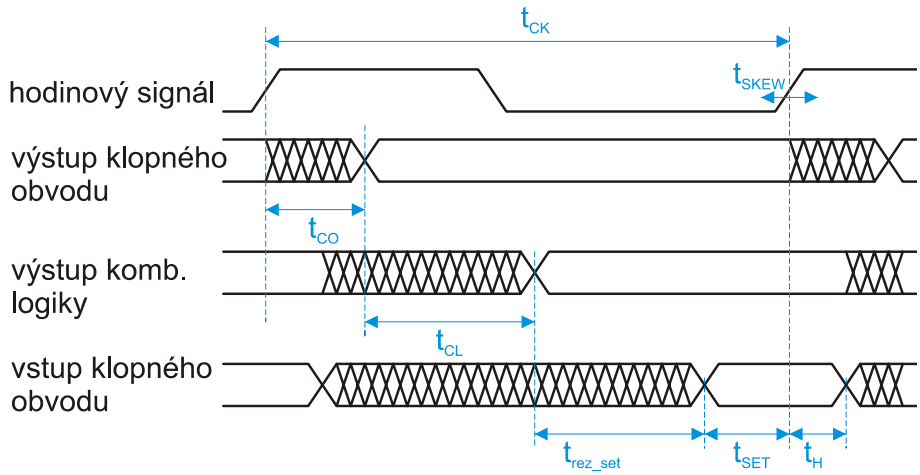


Obr. 4.12 Časový model autosynchrónneho stavového automatu

Model zahrňuje časové parametre sekvenčnej a kombinačnej logiky:

- t_{SET} – čas, po ktorý má byť vstup klopného obvodu stabilný pred aktívnou hranou hodín,
- t_H – čas, po ktorý musí zostať vstup klopného obvodu stabilný po aktívnej hrane hodín,
- t_{CO} – transportné oneskorenie klopného obvodu, teda čas za ktorý sa objaví signál na výstupe klopného obvodu po aktívnej hrane,
- t_{CL} – transportné oneskorenie kombinačnej logiky budúceho stavu,
- t_{DG} – transportné oneskorenie logiky detektora rozdielneho stavu, ustáleného stavu a generátora hodinových impulzov,
- t_{IN} – doba ustálenia vstupov, pri predpokladaní fundamentálneho chovania pri zmene vždy len jedného vstupného signálu v čase, keď sú všetky signály ustálené, tento parameter je možné zanedbať.

Aby bolo možné stanoviť objektívny vzťah medzi parametrami, vyšlo sa z porovnania časových parametrov u synchrónnych stavových automatov, ktorých časový priebeh je na Obr. 4.13.



Obr. 4.13 Časovanie synchrónneho stavového automatu

Zostávajúce parametre na Obr. 4.13 sú definované:

- t_{rez_set} – časová rezerva pre splnenie podmienky t_{SET},
- t_{CK} – minimálna hodinová perióda.

Z obrázku je vidieť, že treba dodržať čas nastavenia vstupov klopného obvodu t_{SET} prostredníctvom časovej rezervy t_{rez_set}, ktorá ešte musí zahrňovať odchýlku fázy hrany hodinového signálu t_{SKEW}, tzv. „skew“. Ďalej je nutné dodržať podmienku pre minimálnu hodinovú periódu:

$$t_{CK} \geq t_{CO} + t_{CL} + t_{SET} \pm t_{SKEW} \tag{4.6}$$

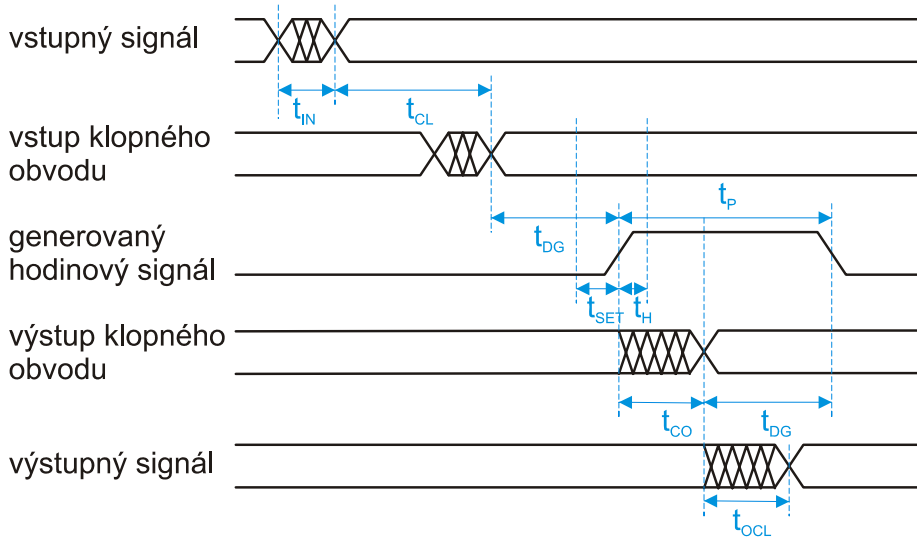
a podmienku pre držanie vstupu klopného obvodu t_H [8]:

$$t_{CO} + t_{CL} \geq t_H . \tag{4.7}$$

U časového diagramu autosynchrónneho automatu z Obr. 4.14 je situácia trochu iná. Tento diagram zachytáva časovanie pri generovaní hodinového impulzu v závislosti od zmeny vstupného signálu. Obvod reaguje na zmenu vstupov a po ich ustálení t_{IN} nastáva spracovanie kombinačnou logikou s oneskorením t_{CL}. Na základe výstupov z kombinačnej logiky reaguje kombinačná logika detektora ustáleného stavu a generátora impulzov s oneskorením t_{DG}. Toto oneskorenie musí byť väčšie ako parameter t_{SET} klopného obvodu, aby sa splnila podmienka pre nastavenie vstupov klopného obvodu a nevznikol problém metastability. Táto podmienka je vždy splnená, pretože oneskorenie kombinačnej logiky v detektore ustáleného stavu t_{DG} je vždy väčšie ako nastavovací čas klopného obvodu t_{SET}.

Po oneskorení t_{DG} je vyvolaný hodinový impulz, ktorého šírku t_P určuje transportné oneskorenie klopného obvodu t_{CO} a transportné oneskorenie detektora ustáleného stavu a generátorom hodín t_{DG}:

$$t_P = t_{CO} + t_{DG} . \tag{4.8}$$



Obr. 4.14 Časovanie autosynchrónneho stavového automatu

Ak obsahuje stavový automat ešte výstupnú kombinačnú logiku, po preklopení stavového registra sa objaví nová hodnota na výstupe za t_{OCL}, čo je transportné oneskorenie výstupnej kombinačnej logiky. V prípade Moorovho automatu sa zmena výstupu od zmeny vstupu vyvolá za čas:

$$t_{PROP} = t_{CL} + t_{DG} + t_{CO} + t_{OCL} . \tag{4.9}$$

V prípade Mealyho automatu tento vzťah udáva maximálne oneskorenie zo vstupu na výstup. Dolná hranica oneskorenia je v prípade priameho prepojenia vstupu s výstupnou kombinačnou logikou len transportné oneskorenie výstupnej kombinačnej logiky t_{OCL}. Oneskorenie ustálenia vstupov t_{IN} nebolo brané do úvahy.

4.3.1. Prechod cez stabilný stav

Pre stanovenie výkonových parametrov je potreba analyzovať časový diagram pri prechode medzi stavmi. Na Obr. 4.15 je ďalej uvedené časovanie pri prechode medzi dvoma stabilnými stavmi. Rýchlosť prechodu medzi dvoma stabilnými stavmi je určená momentom príchodu ďalšieho vstupného stimulu do stavového automatu z vonkajšieho

prostredia. Ďalej bude preto určený minimálny časový rozdiel t_{CI} medzi zmenami vstupov pre vyhodnotenie rýchlosti obvodu. Predtým je však vhodné zaviesť určité predpoklady a definovať momenty v časovom diagrame pomocou vzťahov. Pre uľahčenie sa bude predpokladať fundamentálny režim, tzn. zmena len jedného vstupu v čase za ustálených podmienok, takže doba ustálenia vstupov t_{IN} bude nulová a ďalej prvá zmena vstupov prebehne v čase $t_1=0$. Pre vyjadrenie minimálneho časového rozdielu zmien vstupov je treba určiť podmienky. Prvá je, že ďalšia zmena vstupov (d_2) sa musí prejaviť na výstupe kombinačnej logiky (vstupe klopného obvodu D) až po dobe držania t_H predchádzajúceho hodinového impulzu (cv_1). Ďalšou podmienkou je, aby nástupná hrana nasledujúceho hodinového impulzu (c_2) nepredbehla zostupnú hranu predchádzajúceho hodinového impulzu (f_1). Pre tieto podmienky sa definujú nasledujúce momenty vyznačené aj na Obr. 4.15.

- Moment druhého ustálenia výstupu kombinačnej logiky: $d_2 = t_2 + t_{CL}$, kde t_2 je čas druhej zmeny vstupov.
- Moment novej platnosti dát na vstupe klopného obvodu po prvej nástupnej hrane: $cv_1 = t_{CL} + t_{DG} + t_H$.
- Moment nástupnej hrany druhého hodinového impulzu: $c_2 = t_2 + t_{CL} + t_{DG}$.
- Moment zostupnej hrany prvého hodinového impulzu: $f_1 = t_{CL} + 2.t_{DG} + t_{CO}$.

Na základe podmienky, že ďalšia zmena vstupov (d_2) sa musí prejaviť na výstupe kombinačnej logiky (vstupe klopného obvodu D) až po dobe držania t_H predchádzajúceho hodinového impulzu (cv_1) a stanovených momentov sa vyjadria vzťahy pre odvodenie minimálneho rozpätia zmeny vstupov:

$$\begin{aligned} &d_2 > cv_1 + t_{REZ}, \\ &t_2 + t_{CL} > t_{CL} + t_{DG} + t_H + t_{REZ}, \\ &t_2 > t_{DG} + t_H + t_{REZ}. \end{aligned} \tag{4.10}$$

kde t_{REZ} je časová rezerva pre splnenie tejto podmienky.

Druhá podmienka bola, aby nástupná hrana nasledujúceho hodinového impulzu (c_2) nepredbehla zostupnú hranu predchádzajúceho hodinového impulzu (f_1):

$$\begin{aligned} &c_2 > f_1 + t_{REZ2}, \\ &t_2 + t_{CL} + t_{DG} > t_{CL} + 2.t_{DG} + t_{CO} + t_{REZ2}, \\ &t_2 > t_{DG} + t_{CO} + t_{REZ2}. \end{aligned} \tag{4.11}$$

kde t_{REZ2} je časová rezerva pre ochranný interval medzi zostupnou hranou prvého hodinového impulzu a nástupnou hranou druhého hodinového impulzu. Z týchto dvoch podmienok sa vyberie druhý výraz, ktorý obsahuje na pravej strane vyššiu hodnotu (pri zanedbaní časových rezerv), pretože t_{CO} je väčšie ako t_H pri parametroch klopných obvodov. Z toho ide určiť najmenšia možná perióda zmien vstupov:

$$\begin{aligned} &t_{CI} > t_{2\min} - t_1, \\ &t_{CI} > t_{DG} + t_{CO}, \end{aligned} \tag{4.12}$$

čo je zároveň aj minimálna hodinová perióda t_{CK}, pre porovnanie so synchrónnymi obvodmi. Musí sa brať ale do úvahy, že pri rovnosti v tomto vzťahu

nasledujú hodinové impulzy hneď za sebou (prekrýva sa zostupná hrana predchádzajúceho hodinového impulzu s nástupnou hranou nasledujúceho hodinového impulzu), preto je treba použiť časovú rezervu t_{REZ2}, ktorá zabezpečí stabilnú funkciu automatu:

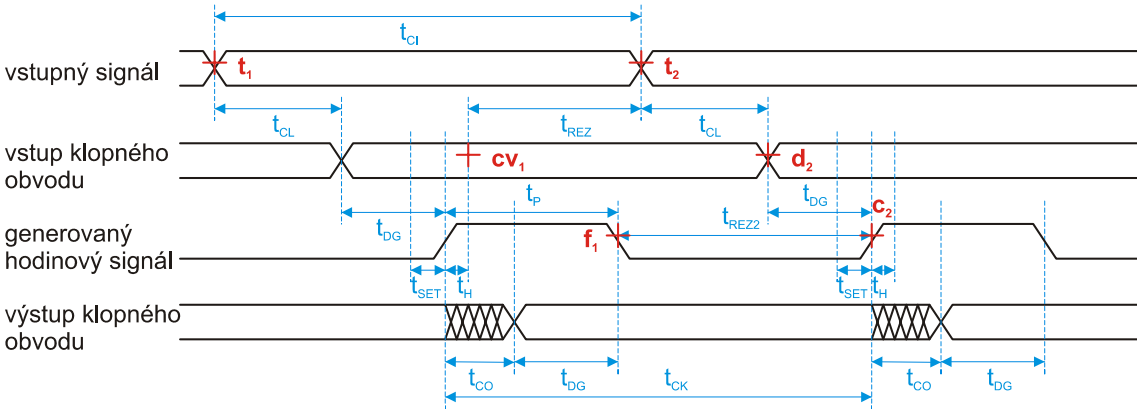
$$t_{CI} = t_{DG} + t_{CO} + t_{REZ2}\,. \tag{4.13}$$

Pre striedu 50 % by mala mať rezerva hodnotu:

$$t_{REZ2} = t_{DG} + t_{CO}\,, \tag{4.14}$$

teda perióda cyklov by bola dvojnásobná:

$$t_{CK} = t_{CI} = 2.(t_{DG} + t_{CO})\,. \tag{4.15}$$

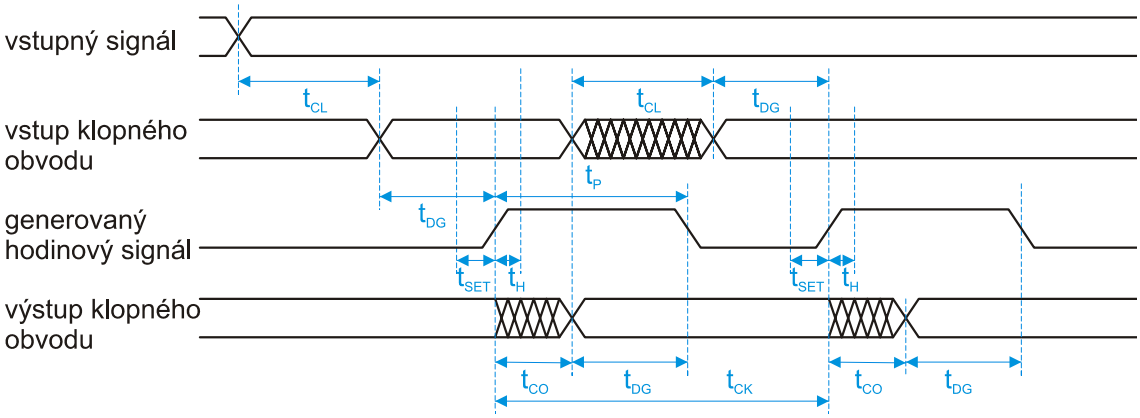


Obr. 4.15 Časovanie autosynchrónneho obvodu s prechodom cez stabilný stav

Ako je vidieť, na rozdiel od synchrónnych obvodov tu nie je treba nastavovať hodinovú periódu v závislosti od oneskorení. Hodinová perióda t_{CK}, alebo inak minimálna zmena vstupov t_{CI} je určená oneskoreniami t_{CO} a t_{DG}. Ďalej tu nie je treba dávať pozor na „skew".

4.3.2. Prechod cez nestabilný stav

Na Obr. 4.16 je druhá možnosť časovania, a to prechodu cez nestabilný stav. Tu je rýchlosť daná len oneskoreniami v kombinačnej logike bez pôsobenia vstupných stimulov z vonkajšieho prostredia.



Obr. 4.16 Časovanie autosynchrónneho obvodu s prechodom cez nestabilný stav

Po zmene vstupného signálu prichádza prvá vyvolaná hrana hodín po čase $t_{CL} + t_{DG}$. Stavový automat sa preklopí do nestabilného stavu po čase t_{CO}. Logika budúceho stavu hneď po tomto momente vygeneruje nový stavový vektor budúceho stavu za čas t_{CL}, ktorý logika detektora ustáleného stavu vyhodnotí po čase t_{DG} ako nový hodinový impulz. Rozdiel medzi nástupnými hranami oboch hodinových impulzov je:

$$t_{CK} = t_{CO} + t_{CL} + t_{DG}, \tag{4.16}$$

čo je vlastne hodinová perióda. Odčítanie šírky hodinového impulzu $t_P = t_{CO} + t_{DG}$ od hodinovej periódy t_{CK} určuje, že šírka dolnej úrovne hodinového signálu je rovná oneskoreniu kombinačnej logike budúceho stavu t_{CL}. Preto, aby bola strieda 50 %, musí byť oneskorenie kombinačnej logiky:

$$t_{CL} = t_{DG} + t_{CO}. \tag{4.17}$$

Rozdiel od predchádzajúceho Obr. 4.15 je, že tu automat nečaká na zmenu vstupov, ale hneď po preklopení do ďalšieho stavu sa začne meniť logika budúceho stavu. Preto tu už nefiguruje časový parameter medzi zmenami vstupov t_{CI}.

V tejto podkapitole boli definované časové parametre dôležité pre návrh autosynchrónnych stavových automatov na základe predchádzajúcich simulácií. Pre tieto parametre boli definované vzťahy, pre ktoré môžu pracovať tieto obvody korektne pre automaty so stabilnými aj nestabilnými stavmi. Jednalo sa o stanovenie hodinových cyklov pre automaty pri prechode cez stabilný aj nestabilný stav. Hodinový cyklus pri prechode cez stabilný stav bol určený ako (4.13). Pri prechode cez nestabilný stav je určený hodinový cyklus presne oneskoreniami (4.16). Je vidieť, že rozdiel je v určení časovej rezervy t_{REZ2} pre prechod cez stabilný stav, ktorá definuje vzdialenosť medzi hodinovými impulzmi. Pokiaľ by bola táto hodnota menšia ako t_{CL}, prechod cez stabilný stav bude mať kratší hodinový cyklus ako prechod cez nestabilný stav. Pre striedu 50 % budú mať obidve varianty rovnaký hodinový cyklus:

$$t_{CK} = 2.(t_{DG} + t_{CO}). \tag{4.18}$$

5 Transformácie sekvenčných systémov

Z dôvodov komplikovaného návrhu systémov s iným synchronizačným princípom ako je princíp taktovania globálnym hodinovým signálom sa javí ako výhodný spôsob transformácia známeho synchrónneho systému s rovnakou funkciou ako požadovaný cieľový systém s odlišnou synchronizačnou autoritou. Transformovať je možné najefektívnejšie v návrhovej fáze na RTL úrovni. Je to technologicky nezávislý medziregistrový popis na vysokej úrovni abstrakcie. Výhodou je, že až po tomto návrhovom kroku nastáva optimalizácia obvodu v podobe minimalizácie logických funkcií a konkrétne priradenie hradiel pre danú technológiu v syntéznom procese.

5.1 Transformácia synchrónneho automatu na autosynchrónny

Predpokladá sa zápis synchrónneho stavového automatu vo VHDL na RTL úrovni. Ukážky zápisu sú čerpané zo šesť-stavového automatu s kódovaním 1 z N v štvrtej kapitole. Tento štandardný doporučený zápis obsahuje deklaráciu knižníc, definíciu entity (port) a deklaráciu signálov pre stavové premenné.

```
type state_type is (s0,s1,s2,s3,s4,s5);
signal state, next_state : state_type;
```

Ďalšou hlavnou časťou je telo architektúry, zložené štandardne z troch procesov. Prvý je sekvenčný, predstavujúci stavový register.

```
SYNC_PROC: process (clk)
begin
   if (clk'event and clk = '1') then
      if (rst = '1') then
         state <= s0;
      else
         state <= next_state;
      end if;
end if;
end process;
```

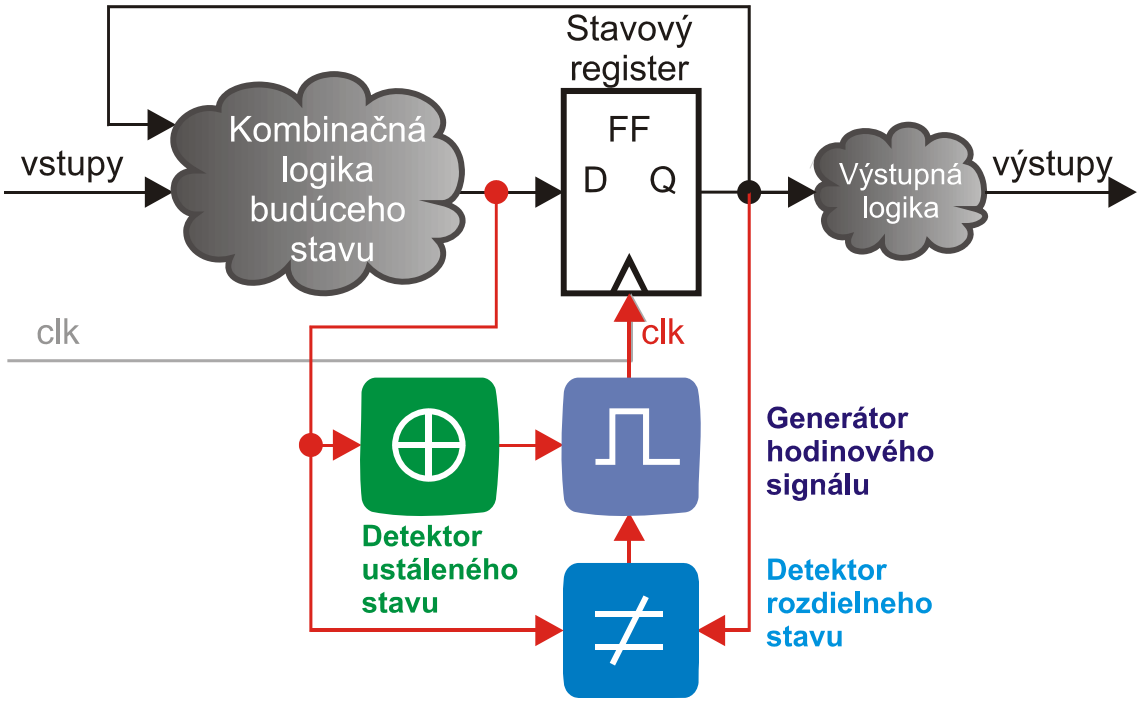
Ďalšie procesy sú kombinačné, jeden vytvára kombinačnú logiku budúceho stavu a druhý prípadne kombinačnú logiku pre výstupné signály.

```
NEXT_STATE_DECODE: process (state, x,y)
begin
next_state <= state;
case (state) is
        when s0 =>
                if x = '1' then
                        next_state <= s1;
                else
                        next_state <= s0;
                end if;
.
.
        when s4 =>
                if (x = '0' and y = '0') then
                        next_state <= s0;
```

```
                    else
                            next_state <= s4;
                    end if;
        .
        .
                when others =>
                    next_state <= s0;
        end case;
        end process;

OUTPUT_DECODE: process (state, x, y)
begin
   if state = s4 then
      z <= '1';
   else
      z <= '0';
   end if;
end process;
```

Popisovaná transformácia vlastne pridá bloky kombinačnej logiky pre generovanie lokálneho hodinového signálu, ako je vidieť z Obr. 5.1. Tieto bloky sú na obrázku vyznačené farebne. Čiernobielo je znázornené pôvodné synchrónne zapojenie. Blok „Stable State Detector“ je kombinačné priradenie pre detekciu ustáleného stavu uvedený vo štvrtom bode postupu transformácie. Blok „Different State Detector“ je proces detekcie rozdielneho stavu uvedený v treťom bode postupu transformácie. Blok „Clock Generator“ obsahuje len kombinačné priradenie vyhodnocujúce podmienky obidvoch detektorov, uvedené v piatom bode postupu transformácie.



Obr. 5.1 Bloková schéma autosynchrónneho automatu

Postup transformácie na RTL úrovni VHDL:

1. Zistenie počtu stavov sekvenčného systému v direktíve „type“ a tie nahradiť napevno definovanými stavmi s konštantne priradeným kódovaním 1 z N.

```
CONSTANT s0 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "000001";
CONSTANT s1 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "000010";
```

```
CONSTANT s2 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "000100";
CONSTANT s3 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "001000";
CONSTANT s4 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "010000";
CONSTANT s5 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "100000";
```

2. Zmeniť typ stavových vektorov z výčtového typu na bitový vektor definovanej dĺžky podľa počtu stavov a kódovania (state, next_state).

```
SIGNAL state : STD_LOGIC_VECTOR(5 DOWNTO 0);
SIGNAL next_state : STD_LOGIC_VECTOR(5 DOWNTO 0);
```

3. Pridanie procesu detektora rozdielneho stavu a deklarácie signálov s ním súvisiace.

```
signal diff_st : std_logic;

diff_state_detector: process (state, next_state)
begin
 if state /= next_state then
        diff_st <= '1';
 else
        diff_st <= '0';
 end if;
end process;
```

4. Pridanie procesu detektora ustáleného stavu (pre kódovanie 1 z N) a deklarácie signálov s ním súvisiace.

```
signal stab_st : std_logic;

stab_st<= next_state(0) xor next_state(1) xor next_state(2) xor
next_state(3) xor next_state(4) xor next_state(5);
```

5. Pridanie kombinačnej časti na spracovanie signálov z procesov z bodu 4 a 5 pre generovanie vlastného hodinového signálu.

```
clk<= diff_st and stab_st;
```

6. Predeklarovať hodinový signál z portu na vnútorný signál.

```
signal clk : std_logic;
```

Táto kapitola popisovala transformáciu autosynchrónneho stavového automatu zo synchrónnej verzie v RTL popise VHDL jazyka. Výhodou transformácie je jednoduchosť popisu synchrónneho automatu a jednoduchosť prevodu do autosynchrónnej verzie pridaním časti kombinačnej logiky.

6 Overenie navrhnutých metód

6.1 Porovnanie parametrov synchrónnych a autosynchrónnych stavových automatov

Na základe transformačného postupu uvedeného v piatej kapitole bola prevedená transformácia šesť-stavového (sa6_synchr_1zN) a pätnásť-stavového (sa15_synchr_1zN) Moorovho automatu v synchrónnej verzii s kódovaním stavov 1 z N. Zapojenie šesť-stavového automatu bolo prevzaté zo štvrtej kapitoly a zapojenie pätnásť-stavového automatu bolo pre tieto účely vytvorené. Podľa piatej kapitoly boli transformované oba automaty na autosynchrónne (sa6_asynchr_1zN) resp. (sa15_asynchr_1zN). Pre obidva automaty boli vytvorené v synchrónnej verzii ešte alternatívy s Grayovým a sekvenčným kódovaním stavov pre doplňujúce porovnanie medzi synchrónnymi verziami (sa6_synchr_bin, sa6_synchr_gray), resp. (sa15_synchr_bin, sa15_synchr_gray).

6.1.1. Úvod a postup

Pre porovnanie vlastností sa použili tri základné kritéria, ktoré sú rozhodujúcim faktorom pri voľbe číslicového systému. Kritériami sú: obsadenie plochy čipu, výkonová spotreba a rýchlosť. Vlastnosti konštrukcií sa porovnávali vo vývojovom prostredí Xilinx ISE, kde sa pre porovnávané konštrukcie využili technologické knižnice pre hradlové pole Spartan3 XC3S200-5pq208.

6.1.2. Návrhový proces a prostriedky

Vybraný návrhový systém je ISE Foundation v.10.1.03. Návrhový proces je zobrazený na Obr. 6.1.

Prvým krokom je návrh systému, kde sa špecifikujú požiadavky kladené na funkciu a vlastnosti obvodu a ich prenesenie do počítačom spracovateľnej formy popisu. Forma popisu môže byť na rôznych úrovniach abstrakcie: hradlovej, schématickej, RTL a algoritmickej.

Ďalším krokom je syntéza, ktorá sa niekde zahrňuje aj do procesu implementácie. Prichádza tu k zníženiu úrovne popisu, kde sa konverguje kód HDL na RTL úrovni zapísaný pomocou syntetizovateľnej podmnožiny jazyka do zoznamu logických prvkov a ich vzájomného prepojenia – netlistu. Dochádza tu k detekovaniu registrov, kombinačnej logiky, kódujú sa symbolické stavy stavových automatov, rozpoznávajú sa špeciálne štruktúry (násobičky, sčítačky), dochádza k odstráneniu nepoužitej logiky a k ďalším optimalizačným krokom. Proces syntézy je riadený technologickou knižnicou a tzv. vymedzeniami (angl. constraints). Technologická knižnica obsahuje popis prvkov vo zvolenom type obvodu FPGA a je dodaná od výrobcu. Pomocou vymedzení (constraints) návrhár definuje rôzne parametre (očakávaná hodinová frekvencia), alebo mapovanie vstupov a výstupov na skutočné vývody. V procese syntézy u FPGA obvodov prebieha aj proces mapovania na technológiu. Jedná sa o konverziu technologicky nezávislého netlistu do buniek danej technológie, napr. zoskupovanie

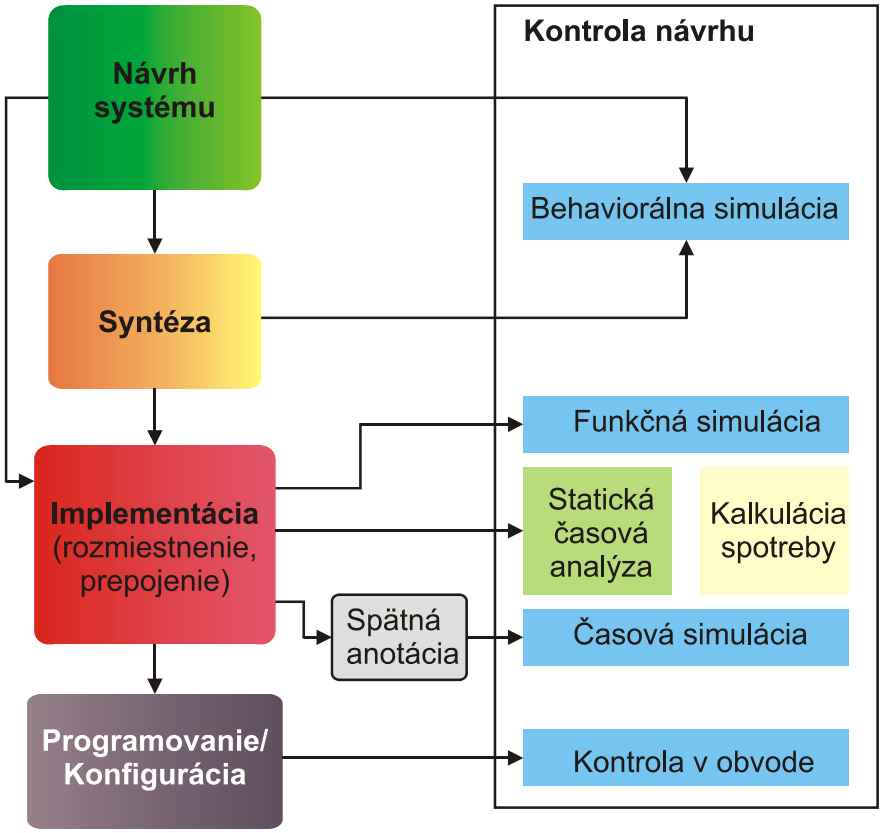
LUT a registrov do rezov, ďalšia optimalizácia, replikácia registrov. Syntéza bola prevedená integrovaným nástrojom syntézy XST.

Po syntéze nasleduje implementácia, čo je vlastne vytvorenie konfiguračných dát pre programovateľné pole. Tento proces je rozdelený na dve časti: rozmiestnenie a prepojenie. Najprv sú namapované logické prvky umiestnené do matice obvodu a zafixované. Potom pri prepojení sú preberané jednotlivé spoje a identifikované vhodné prepojovacie štruktúry (kanály, spoje, prepínacie prvky) a tie postupne zafixované.

Záverečným procesom je konfigurácia reálneho obvodu FPGA. Tu sa nahráva bitový tok konfiguračných dát do štruktúry FPGA alebo do externej konfiguračnej pamäte.

Dôležitým krokom v návrhovom procese je kontrola návrhu (verifikácia). Tá prebieha na rôznych úrovniach v závislosti od stupňa návrhu. Slúži na overenie funkčnej správnosti navrhovaného obvodu a dodržanie časových parametrov. Sledujú sa reakcie chovania obvodu vybudené zostavenými vstupnými stimulmi. Najnižším stupňom je simulácia na úrovni RTL, tzv. behaviorálna simulácia. Táto simulácia nerešpektuje reálne oneskorenia. Je ale rýchla a slúži pre overenie funkcie obvodu.

Na ďalšom stupni po syntéze je simulácia na hradlovej úrovni (post-synthesis) alebo funkčná simulácia uvedená na Obr. 6.1. Tu sa simuluje netlist po syntéze pred rozmiestnením a prepojením. Nejde tu spoľahlivo odhadnúť oneskorenie kombinačnej logiky a prepojení, a teda nejde presne simulovať reálne chovanie. Preto sa táto simulácia používa len na overenie správnej funkcie syntéznych nástrojov.



Obr. 6.1 Návrhový proces číslicových obvodov (FPGA)

Simulácia po implementácii na hradlovej úrovni (back-annotated gate level, post-place and route) simuluje reálne oneskorenia po rozmiestnení a prepojení, teda finálnu konštrukciu implementovanú vo zvolenej technológii. Proces nastavenia časových parametrov v inštanciách modelov jednotlivých buniek obvodu je uskutočnený ihneď po načítaní netlistu a nazýva sa spätná anotácia.

Pre overenie časových parametrov je možné využiť statickú časovú analýzu. Pre odhad výkonovej spotreby sa v implementačnom procese nachádza aj nástroj odhadujúci spotrebu na základe využitých logických prostriedkov a nastavených parametrov konštrukcie. Statická časová analýza bola prevedená integrovaným nástrojom Timing Analyzer a analýza výkonovej spotreby integrovaným nástrojom XPower.

Najvyšším stupňom verifikácie je kontrola v nakonfigurovanom fyzickom obvode [41].

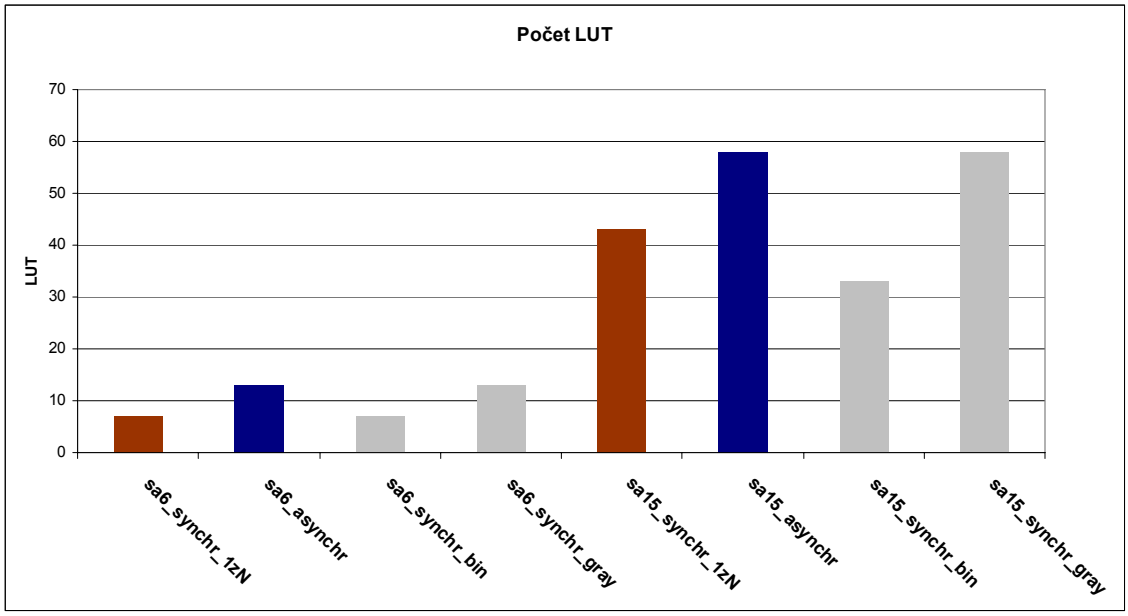
6.1.3. Porovnanie z hľadiska využitia plochy čipu

Kritérium plochy bolo skúmané kvôli zvolenej platforme FPGA kvantitou využitých logických prostriedkov v počte priraďovacích tabuliek LUT, počte klopných obvodov (FF) a počte rezov. Výsledky boli prebrané zo syntéznej správy v XST nástroji, viď Tab. 6.1.

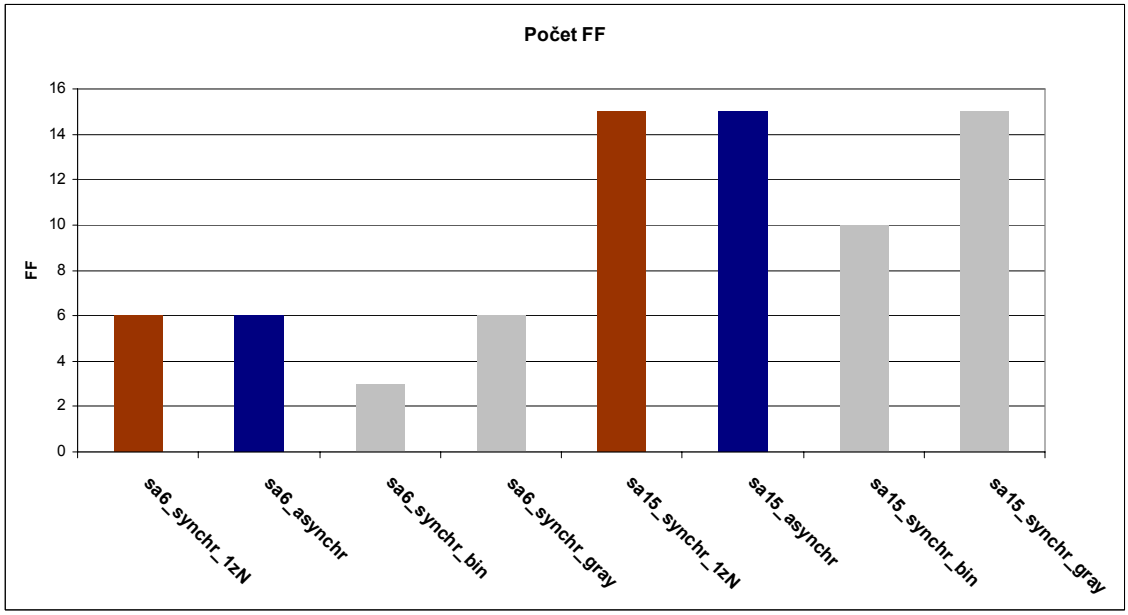
Tab. 6.1 Využitie logických prostriedkov (LUT, FF, rezy)

Stavové automaty	LUT	FF	Slices
sa6_synchr_1zN	7	6	4
sa6_asynchr	13	6	7
sa6_synchr_bin	7	3	4
sa6_synchr_gray	13	6	7
sa15_synchr_1zN	43	15	23
sa15_asynchr	58	15	31
sa15_synchr_bin	33	10	19
sa15_synchr_gray	58	15	31

Pre názornosť boli tabuľky prevedené do grafickej podoby na Obr. 6.2 podľa počtu LUT tabuliek a na Obr. 6.3 podľa počtu klopných obvodov. Farebné stĺpce zobrazujú porovnateľné automaty (hnedá – synchrónne, modrá - autosynchrónne). Šedé stĺpce dopĺňajú porovnanie o synchrónne verzie s odlišným kódovaním.



Obr. 6.2 Porovnanie využitia LUT tabuliek u rôznych stavových automatov



Obr. 6.3 Porovnanie využitia klopných obvodov u rôznych stavových automatov

Porovnanie z hľadiska plochy dopadlo podľa očakávania, pretože autosynchrónne obvody obsahujú oproti synchrónnym navyše nadbytočnú kombinačnú logiku. Porovnávané boli hlavne konštrukcie autosynchrónneho automatu (sa6_asynchr, resp. sa15_asynchr) so synchrónnym automatom s kódovaním stavov 1 z N (sa6_synchr_1zN, resp. sa15_synchr_1zN), pretože bol z tohto automatu transformovaný (má rovnaké kódovanie stavov). Pre malé konštrukcie (šesť-stavový automat) bol rozdiel v počte buniek LUT značne väčší (o 85 %). So zvyšujúcou sa zložitosťou sa rozdiely zmenšovali: pri 15-stavovom automate o 34 %. Porovnanie rezov kopírovalo situáciu s porovnaním LUT. Pri porovnaní počtu klopných obvodov FF hralo hlavnú úlohu použité kódovanie stavov. Preto boli pri porovnaní autosynchrónnych automatov so synchrónnymi automatmi rovnaké, lebo používali

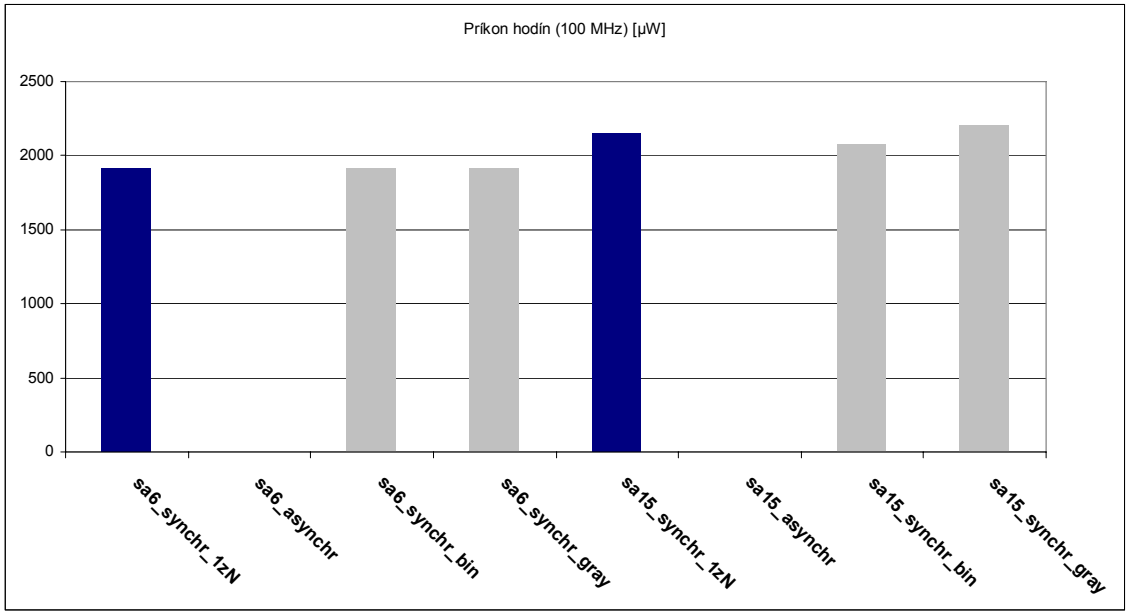
rovnaké kódovanie 1 z N. Pri transformácii totiž vzniká v autosynchrónnom obvode len prídavná kombinačná logika. Pri porovnaní rôznych druhov kódovaní synchrónnych automatov medzi sebou mali najlepšie využitie plochy automaty s binárnym kódovaním. Automaty s Grayovým kódovaním mali podobné využitie plochy ako automaty s kódovaním 1 z N.

6.1.4. Porovnanie z hľadiska výkonovej spotreby

Výkonová spotreba bola analyzovaná v nástroji XPower. Tabuľka Tab. 6.2 udáva hodnoty parametrov ako je príkon hodinového stromu (pri frekvencii 100 MHz), príkon logiky a celkový dynamický príkon.

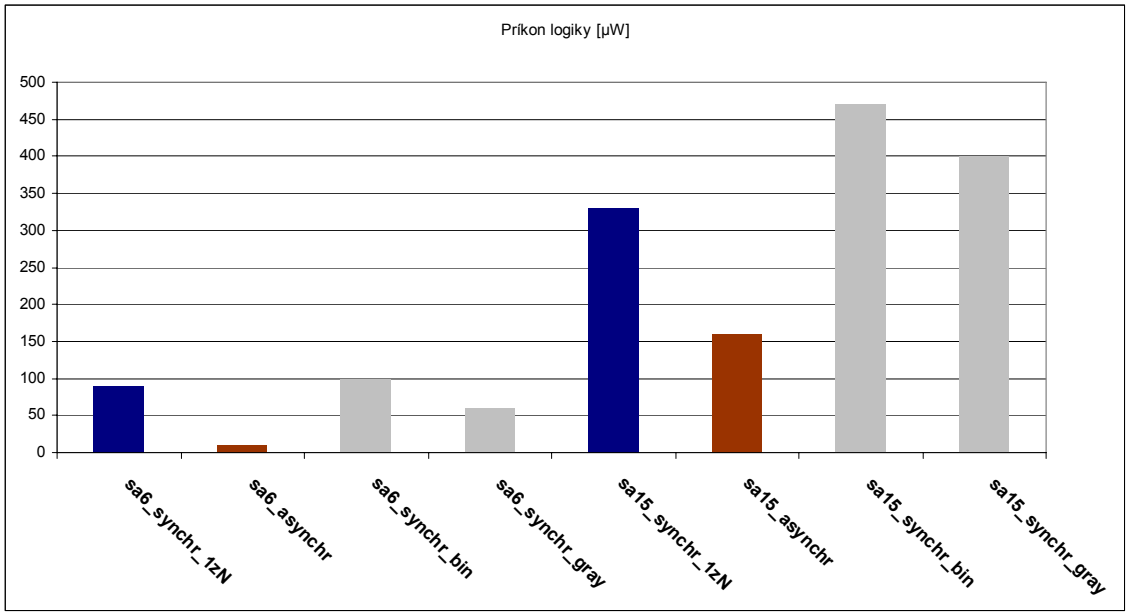
Tab. 6.2 Výkonová spotreba

Stavové automaty	Príkon hodín (100 MHz) [µW]	Príkon logiky [µW]	Celk. dynam. príkon [µW]	Celkový príkon [µW]
sa6_synchr_1zN	1 910	90	3 930	44 980
sa6_asynchr	0	10	190	41 220
sa6_synchr_bin	1 910	100	3 940	44 990
sa6_synchr_gray	1 910	60	2 620	43 660
sa15_synchr_1zN	2 150	330	5 480	46 540
sa15_asynchr	0	160	1 310	42 340
sa15_synchr_bin	2 080	470	5 580	46 640
sa15_synchr_gray	2 210	400	5 050	46 110



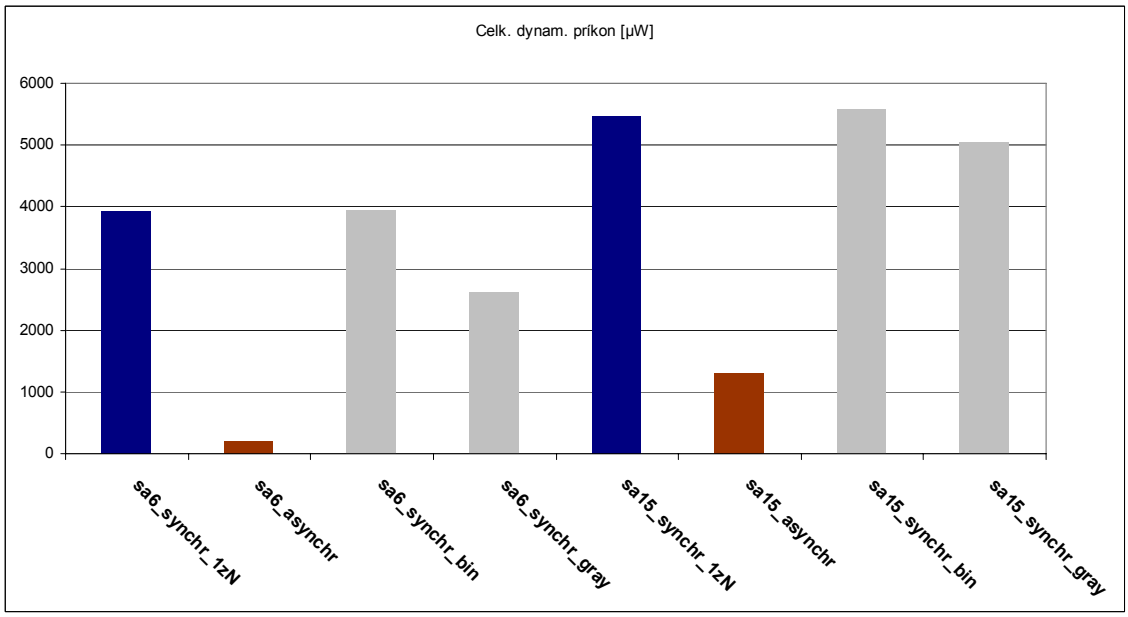
Obr. 6.4 Porovnanie výkonovej spotreby hodinových obvodov

Porovnanie na základe príkonov hodinových stromov konštrukcií stavových automatov na Obr. 6.4 ukazuje, že autosynchrónne automaty nemajú žiadnu spotrebu, pretože nevyužívajú globálny hodinový signál. Pri porovnaní synchrónnych automatov je väčšia spotreba u zložitejších (pätnásť-stavových) automatov. Spotreba hodinových stromov nie je závislá na použitom kódovaní stavov.



Obr. 6.5 Porovnanie výkonovej spotreby logických blokov

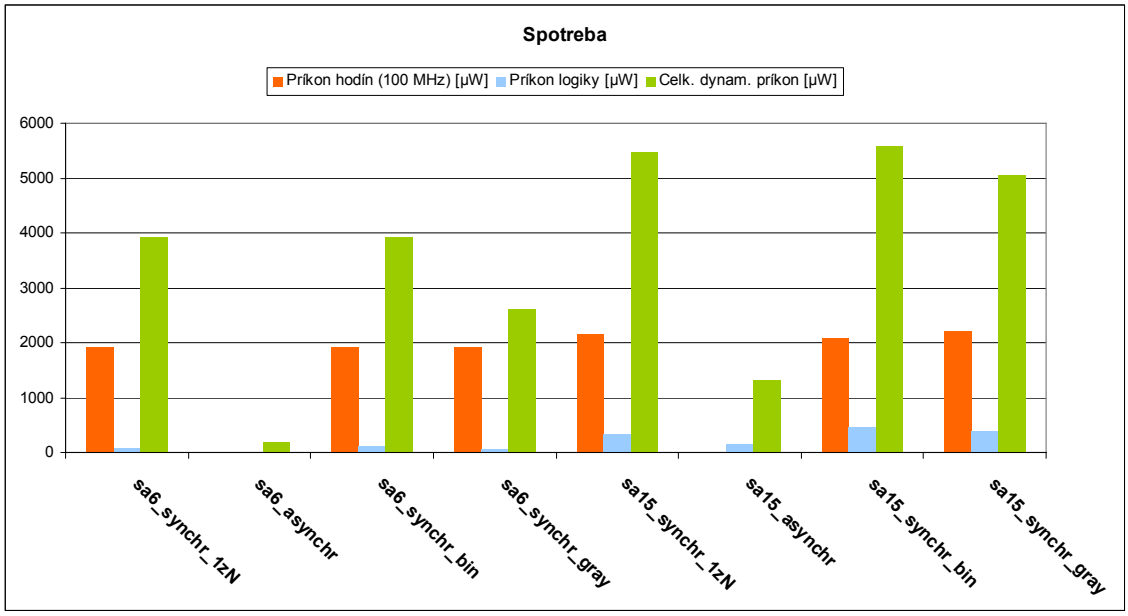
Porovnanie príkonu logiky na Obr. 6.5 opäť znázorňuje deväťkrát nižšiu spotrebu autosynchrónneho automatu oproti jeho synchrónnemu transformačnému zdroju (sa6_synchr_1zN) u šesť-stavových automatov. U zložitejších pätnásť-stavových konštrukcií sa tento rozdiel zmenšuje už len na dvojnásobok. V porovnaní medzi synchrónnymi automatmi mal najväčšiu spotrebu automat s binárnym kódovaním stavov.



Obr. 6.6 Porovnanie celkovej dynamickej výkonovej spotreby (hodiny, logika, signály, I/O)

Porovnanie celkového dynamického príkonu z Obr. 6.6 je výraznejšie. Autosynchrónny automat (sa6_asynchr) má oproti synchrónnemu automatu (sa6_synchr_1zN) dvadsaťkrát nižšiu spotrebu. Smerom k zložitejším konštrukciám tento rozdiel klesá na štvornásobne nižšiu spotrebu pre pätnásť-stavový autosynchrónny

automat. Medzi synchrónnymi automatmi má automat s Grayovým kódovaním najnižšiu spotrebu kvôli svojim vlastnostiam, kde je len jedna zmena v stavovom vektore a je menej prechodov v kombinačnej logike.



Obr. 6.7 Komplexný pohľad na výkonovú spotrebu jednotlivých zložiek výkonu

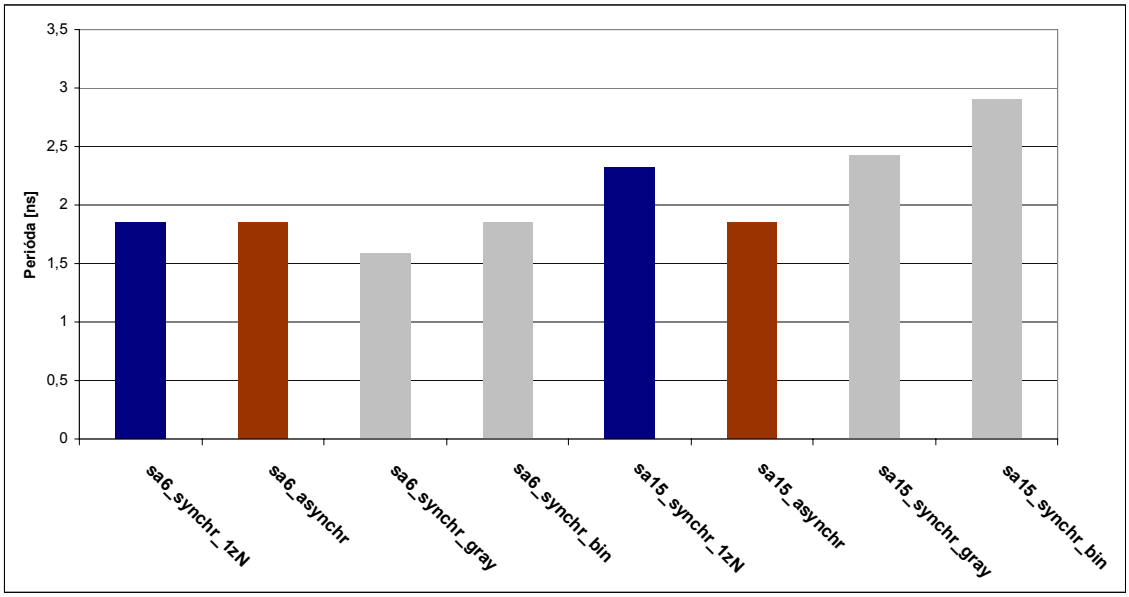
Obr. 6.7 ukazuje komplexný pohľad na porovnanie rôznych druhov príkonu autosynchrónnych automatov v porovnaní so synchrónnymi automatmi. Jednoznačne z toho vyplývajú výhodnejšie autosynchrónne obvody kvôli svojim samoriadiacim vlastnostiam.

6.1.5. Porovnanie z hľadiska časovania

Pomocou statickej časovej analýzy sa v nástroji Timing Analyzer analyzovali časové oneskorenia stavových automatov vo fáze implementačného procesu po mapovaní. V druhom stĺpci Tab. 6.3 je uvedený parameter oneskorenia od aktívnej hrany hodín až k zmene výstupu. Pre autosynchrónne automaty je začiatok počítaný od hrany vstupného signálu namiesto hrany hodín. Tretí stĺpec vyjadruje minimálnu periódu zmien v obvode stavového automatu. Pre synchrónne automaty to je hodinová perióda a pre autosynchrónne je to minimálna doba medzi dvoma nástupnými hranami vstupného signálu. Obrátenou hodnotou tejto veličiny je vo štvrtom stĺpci tabuľky maximálna frekvencia zmien.

Tab. 6.3 Časovanie rôznych stavových automatov

Stavové automaty	Oneskorenie zo vstupu na výstup [ns]	Minimálna perióda zmeny [ns]	Maximálna frekvencia [MHz]
sa6_synchr_1zN	6,407	1,855	539,08
sa6_asynchr	8,622	1,855	539,08
sa6_synchr_gray	6,986	1,590	628,93
sa6_synchr_bin	6,986	1,855	539,08
sa15_synchr_1zN	7,565	2,322	430,66
sa15_asynchr	10,838	1,855	539,08
sa15_synchr_gray	7,036	2,418	413,56
sa15_synchr_bin	6,986	2,901	344,71



Obr. 6.8 Porovnanie minimálnej periódy zmeny

Zo statickej časovej analýzy vyplýva, že oneskorenia od vstupu na výstup sú väčšie u autosynchrónnych automatov o 34 % u šesť-stavového, resp. o 43 % u pätnásť-stavového automatu. Minimálna perióda zmien je u šesť-stavových automatov medzi sebou rovnaká, na druhej strane u pätnásť-stavového autosynchrónneho automatu je nižšia o 25 % oproti synchrónnemu automatu. Medzi synchrónnymi automatmi dosahuje najnižšie časy periódy zmien automat s kódovaním 1 z N zapríčinené jednoduchou kombinačnou logikou budúceho stavu.

6.1.6. Závery porovnaní

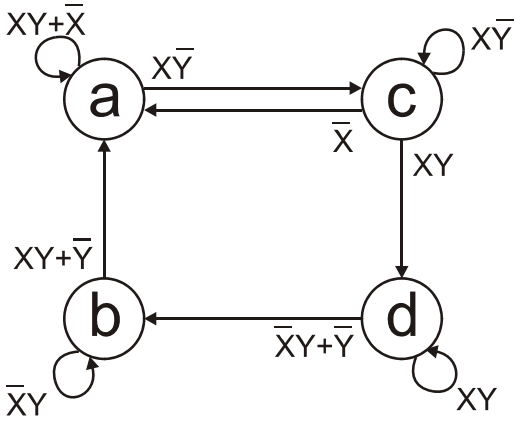
Celkovo je možné zhodnotiť, že autosynchrónne automaty majú z pohľadu obsadenej plochy na čipe horšie parametre kvôli svojej prídavnej kombinačnej logike. Z pohľadu spotreby zase majú najlepšie parametre kvôli nevyužitiu globálnych hodín. Z pohľadu časovania majú podobné, prípadne lepšie vlastnosti u zložitejších konštrukcií. Je treba ešte dodať, že rozdiel v parametroch medzi autosynchrónnymi a synchrónnymi automatmi klesá s rastúcou zložitosťou konštrukcií.

Záverečná poznámka: Udané parametre závisia od zložitosti automatu, počte prechodov, zložitosti kombinačnej logiky, dané automaty sú príklady jednoduchých ukážok konštrukcií bez zamýšľanej účelnosti.

6.2 Porovnanie časových vlastností stavového automatu s rôznou synchronizáciou

Táto podkapitola sa bude zaoberať teoretickým porovnaním výkonnostných časových parametrov rôznych stupňov synchronizácie konštrukcie jedného stavového automatu. Do úvahy boli brané dva základné parametre, a to minimálna perióda cyklu t_{CLK} a oneskorenie zo vstupu na výstup t_{PROP}. Minimálna perióda cyklu je u synchrónneho automatu považovaná za hodinovú periódu, u zvyšných automatov to je časový úsek, za ktorý sa zmení stavový vektor súčasného stavu buď preklopením pamäťového prvku v spätnej väzbe alebo prekonaním určitého oneskorenia. Pre oneskorenie zo vstupu na výstup sa odvodí rozsah oneskorenia od minimálnej hranice po maximálnu hranicu kvôli použitiu Mealyho typu automatu, ktorého vstupy sú priamo spojené aj s výstupnou kombinačnou logikou.

Štruktúra stavového automatu Mealyho typu na Obr. 6.9 bola prevzatá z [3], kde boli zároveň aj popísané jeho výkonnostné časové parametre pre štruktúru EAIC. Tieto parametre budú porovnané s parametrami ostatných synchronizačných verzií stavového automatu odvodené na základe zapojenia a popisu.



Obr. 6.9 Stavový graf porovnávaného automatu

Pre definovanie oneskorenia kombinačnej logiky boli odvodené rovnice budúceho stavu pre kombinačnú logiku. Pre jednoduché binárne kódovanie stavov to je:

$$\begin{aligned} D_0 &= Q_0\overline{X}Y + Q_0Q_1 + Q_1XY, \\ D_1 &= Q_1XY + \overline{Q}_0X\overline{Y}, \\ Z &= Q_0\overline{Q}_1\overline{X}Y. \end{aligned} \tag{6.1}$$

Toto kódovanie stavov nie je ale vhodné pre niektoré druhy asynchrónnych automatov ako je autosynchrónny automat a asynchrónny fundamentálny automat kvôli predchádzaniu hazardom. Preto sú ďalej popísané aj rovnice pre kódovanie 1 z N:

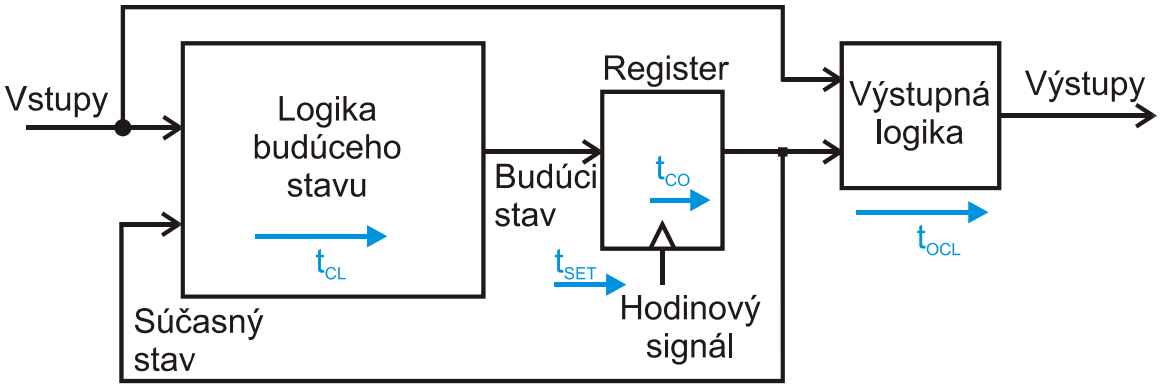
$$
\begin{aligned}
D_0 &= Q_2\overline{X} + Q_1XY + Q_1\overline{Y} + Q_0\overline{X} + Q_0XY,\\
D_1 &= Q_3\overline{X}Y + Q_3\overline{Y} + Q_1\overline{X}Y,\\
D_2 &= Q_0X\overline{Y} + Q_2X\overline{Y},\\
D_3 &= Q_2XY + Q_3XY,\\
Z &= Q_1\overline{X}Y.
\end{aligned}
\tag{6.2}
$$

Pre porovnanie výkonnostných časových parametrov uvedených automatov sa vytvorilo zjednodušenie na úrovni jednej hradlovej technológie (bola vybraná rada integrovaných obvodov 74Fxx). Na základe logických rovníc (6.1), príp. (6.2) sa definovali kombinačné logiky budúceho stavu a výstupu z diskrétnych hradiel danej technológie. Pre diskrétne hradlá sa definovalo spoločné transportné oneskorenie jedného logického hradla t_G. Transportné oneskorenie hradla v uvedenej rade 74FXX bolo $t_G = 3{,}7$ ns za štandardných podmienok ($V_{cc} = 5$ V, $T = 25$ °C). Rovnako pomocou tohto oneskorenia sa na základe zvolenej technológie definovali parametre klopných obvodov stavových automatov: $t_{CO} = \frac{5}{3}.t_G$, $t_{SET} = \frac{6}{11}.t_G$.

Pomocou týchto parametrov budú vyčíslené a porovnané časové parametre pre jednotlivé prípady stavových automatov v Tab. 6.4.

6.2.1. Synchrónny stavový automat

Na Obr. 6.10 je zapojenie synchrónneho automatu Mealyho typu. Pre určenie požadovaných parametrov je treba definovať parametre ako je transportné oneskorenie klopného obvodu t_{CO}, čas nástupu klopného obvodu t_{SET}, oneskorenie kombinačnej logiky budúceho stavu t_{CL} a kombinačnej logiky výstupu t_{OCL}.



Obr. 6.10 Synchrónny stavový automat

Potom podľa [8] je hodinová perióda určená ako:

$$t_{CLK} = t_{CO} + t_{CL} + t_{SET}. \tag{6.3}$$

Oneskorenie zo vstupu na výstup leží v intervale:

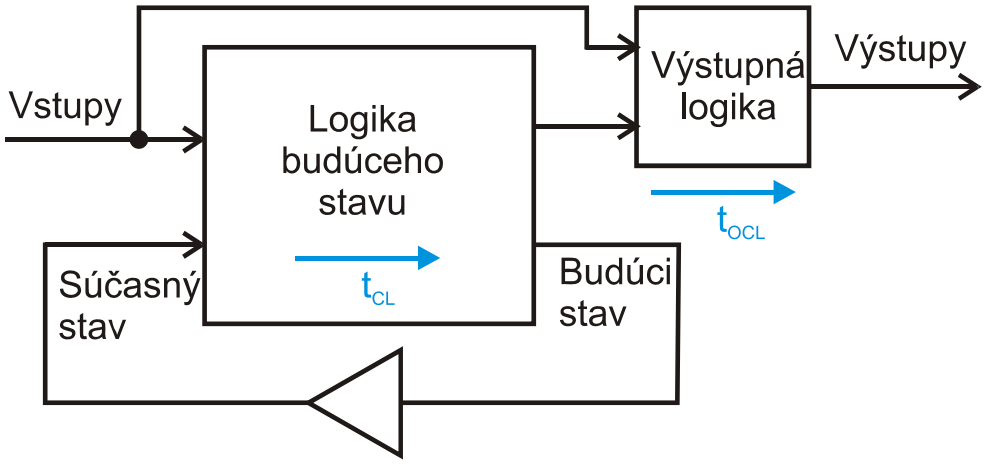
$$t_{OCL} \le t_{PROP} \le t_{CLK} + t_{CL} + t_{OCL}. \tag{6.4}$$

Pre vyčíslenie časových parametrov budú určené jednotlivé oneskorenia pomocou parametra t_G. Oneskorenie kombinačnej logiky budúceho stavu bolo na základe

dvojstupňovej štruktúry kombinačnej logiky určené ako: $t_{CL} = 2.t_G$. Výstupnú logiku tvorí len jedno hradlo, preto: $t_{OCL} = t_G$.

6.2.2. Asynchrónny Huffmanov stavový automat

Na Obr. 6.11 je zobrazený asynchrónny stavový automat vo fundamentálnom režime, ktorého spätná väzba je spojená priamo (tzv. Huffmanov štýl automatu). Využíva podmienku fundamentálneho režimu, t.j. že sa mení len jeden vstup v čase, keď je automat v ustálenom stave. Na predchádzanie hazardov využíva kódovanie 1 z N, kde sú logické rovnice doplnené o term navyše, ktorý spôsobuje prekrývanie prechodov stavového vektoru.



Obr. 6.11 Asynchrónny fundamentálny stavový automat

Pri zanedbaní oneskorenia vodičov v spätnej väzbe je hodinový cyklus určený oneskorením kombinačnej logiky [3]:

$$t_{CLK} = t_{CL}. \tag{6.5}$$

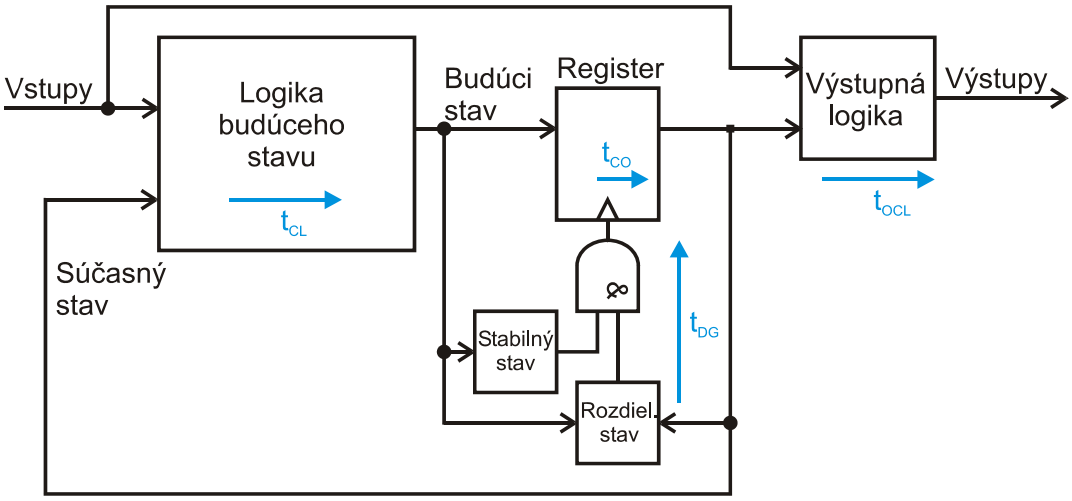
Oneskorenie kombinačnej logiky budúceho stavu je možné určiť z jej dvojstupňovej štruktúry rovnako ako v predchádzajúcom prípade: $t_{CL} = 2.t_G$. Oneskorenie výstupnej logiky je rovnaké ako v predchádzajúcom prípade: $t_{OCL} = t_G$.

Rozsah oneskorenia zo vstupu na výstup je daný:

$$t_{OCL} \le t_{PROP} \le t_{CL} + t_{OCL}. \tag{6.6}$$

6.2.3. Autosynchrónny stavový automat

Na Obr. 6.12 je zapojenie autosynchrónneho stavového automatu rozoberaného v tejto práci. Oproti predchádzajúcim automatom tu je treba definovať navyše oneskorenie detektora rozdielneho stavu a ustáleného stavu t_{DG}.



Obr. 6.12 Autosynchrónny stavový automat

V kapitole 4.3 boli definované časové parametre nasledovne:

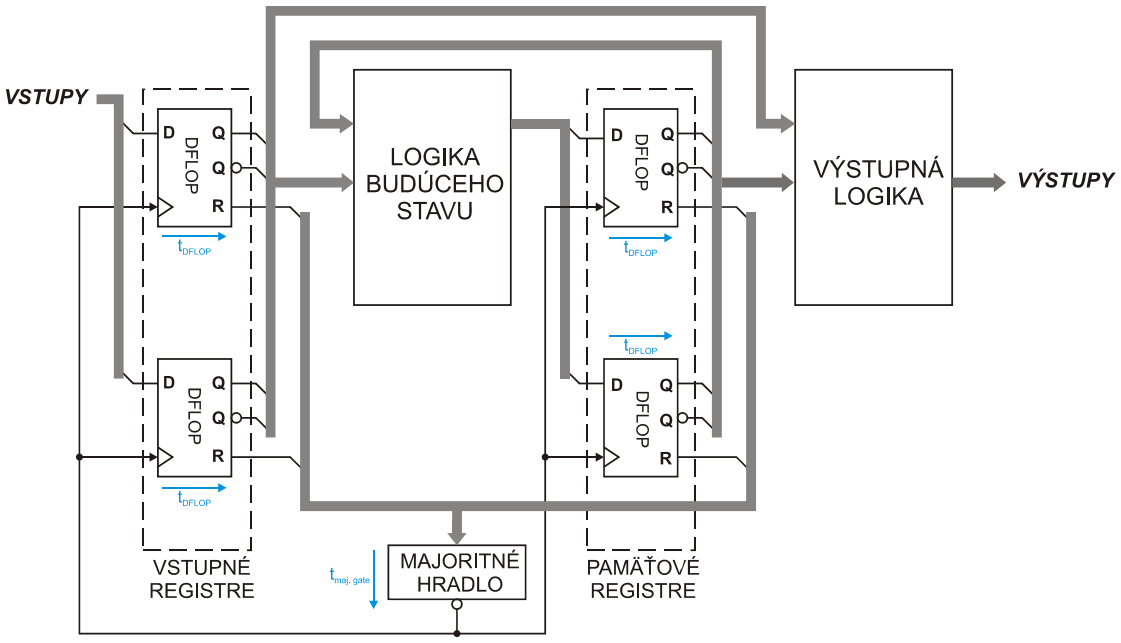
$$t_{CLK} = 2.t_{DG} + 2.t_{CO}\,, \tag{6.7}$$

$$t_{OCL} \le t_{PROP} \le t_{CL} + t_{DG} + t_{CO} + t_{OCL}\,. \tag{6.8}$$

Parameter t_{DG} je možné určiť na základe zapojenia detektora rozdielneho a ustáleného stavu automatu, ktorý pre štvorbitový stavový vektor má zapojený v najpomalšej ceste dve hradlá XOR a jedno AND hradlo za sebou. Preto je možné určiť toto oneskorenie ako $t_{DG} = 5.t_G$ pre konkrétny prípad automatu. Ostatné parametre sa dajú kvantitatívne vyčísliť rovnako ako u synchrónneho automatu.

6.2.4. EAIC stavový automat

Ďalším systémom synchronizácie stavového automatu je EAIC na Obr. 6.13. Je to pôvodné zapojenie porovnávaného automatu v tejto podkapitole [3].



Obr. 6.13 EAIC stavový automat

Pre štyri stavy je použité binárne kódovanie a teda dva stavové registre. Perióda cyklu je vyjadrená ako:

$$t_{CLK} = 2.t_{DFLOP} + 2.t_{maj.gate}, \tag{6.9}$$

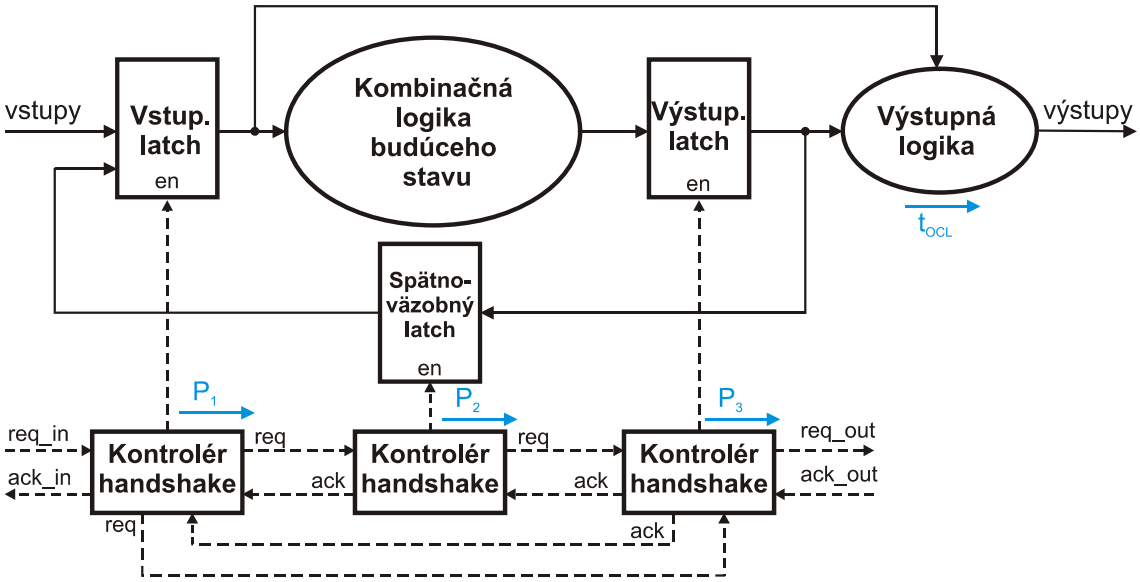
kde t_{DFLOP} je transportné oneskorenie špeciálneho registra a $t_{maj.\,gate}$ je oneskorenie majoritného hradla (kompletovacieho detektora). Oneskorenie zo vstupu na výstup je v rozmedzí:

$$3.t_{DFLOP} + 2.t_{maj.gate} \le t_{PROP} \le t_{CLK} + 3.t_{DFLOP} + 2.t_{maj.gate}. \tag{6.10}$$

Podľa štruktúry prvku DFLOP v kapitole 2.1.4 je možné jeho oneskorenie určiť ako: $t_{DFLOP} \cong 3.t_G$. Majoritné hradlo je vlastne viacvstupový Mullerov element C, ktorý je zložený z dvoch hradiel za sebou, a preto: $t_{maj.gate} \cong 2.t_G$.

6.2.5. Bundled-data stavový automat

Ďalej nasledujú len čisto asynchrónne štýly automatov založené na handshakingových štvorfázových protokoloch bundled-data a dual-rail. Na Obr. 6.14 je zapojenie stavového automatu s protokolom bundled-data. Pre zaistenie stabilnej funkcie sú tam nutné tri latche. Jeden pre dátový token, druhý pre nulový token a tretí prázdny, pre prijatie nového tokenu.



Obr. 6.14 Bundled-data stavový automat

Perióda cyklu je závislá len od oneskorení v kontroléroch handshakingu, pretože oneskorenie kombinačnej logiky je prekryté prispôsobeným oneskorením v riadiacej ceste handshakingu. Toto oneskorenie musí zodpovedať oneskoreniu v najpomalšej ceste kombinačnej logiky. Stavový automat je zapojený v kruhovej topológii latchov. Perióda cyklu sa dá vyjadriť ako súčet oneskorenia kontrolérov latchov [1]:

$$t_{CLK} = P_1 + P_2 + P_3 = 2.L_{fd} + 4.L_f + 6.L_r, \tag{6.11}$$

kde P_1 je oneskorenie kontroléra latchu s kombinačnou logikou (Input Latch), P_2 a P_3 sú oneskorenia kontrolérov bez kombinačnej logiky. V závislosti od protokolu sú tieto hodnoty závislé na doprednom oneskorení L_f (L_{fd}) a spätnom oneskorení L_r.

$$P_1 = 2.L_{fd} + 2.L_r . \tag{6.12}$$

$$P_2 = P_3 = 2.L_f + 2.L_r . \tag{6.13}$$

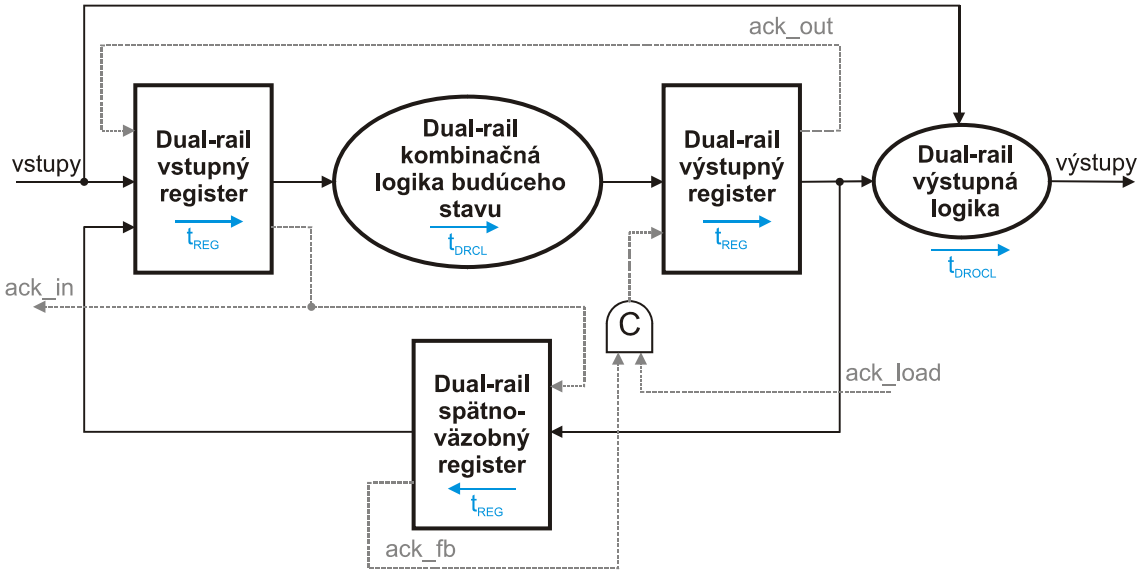
Dopredné oneskorenie je určené podľa štruktúry kontroléru v [1] ako: $L_{fd} = t_{CL} + t_C$ pre kontrolér s dátami a $L_f = t_C$ pre kontrolér bez dát. Spätné oneskorenie je $L_r = t_C$. Parameter t_C znamená oneskorenie Mullerovho elementu C, ktoré bolo predtým určené ako: $t_C = 2.t_G$.

Vyjadrenie oneskorenia zo vstupu na výstup je nasledovné:

$$t_{OCL} \le t_{PROP} \le P_1 + P_2 + P_3 + t_{OCL} . \tag{6.14}$$

6.2.6. Dual-rail stavový automat

Stavový automat s asynchrónnym protokolom dual-rail je na Obr. 6.15. Rovnako ako v predchádzajúcom prípade sú tu použité tri pamäťové prvky pre zaistenie stabilnej funkcie. Na rozdiel od predchádzajúceho prípadu je riadenie zahrnuté priamo v dátovej ceste, ako to vyplýva z vlastností dual-rail protokolu [68].



Obr. 6.15 Dual-rail stavový automat

Perióda cyklu je určená ako:

$$t_{CLK} = t_{REG} + t_{DRCL} + t_{REG} + t_{REG} , \tag{6.15}$$

kde t_{REG} je oneskorenie v dual-rail registri a t_{DRCL} je oneskorenie kombinačnej dual-rail logiky. Oneskorenie dual-rail klopného obvodu t_{REG} vyplýva z jeho štruktúry na Obr. 2.28 ako: $t_{REG} = t_C + t_G + t_C$. Oneskorenie kombinačnej logiky budúceho stavu, resp. výstupu je zložené z oneskorení základných dual-rail hradiel, ktorých štruktúra je na Obr. 2.27. Podľa toho je možné určiť oneskorenie celej kombinačnej logiky budúceho stavu a výstupu ako: $t_{DRCL} = 2.(2.t_G + t_C)$, resp. $t_{DROCL} = 2.t_G$.

Rozsah oneskorenia zo vstupu na výstup je určený ako:

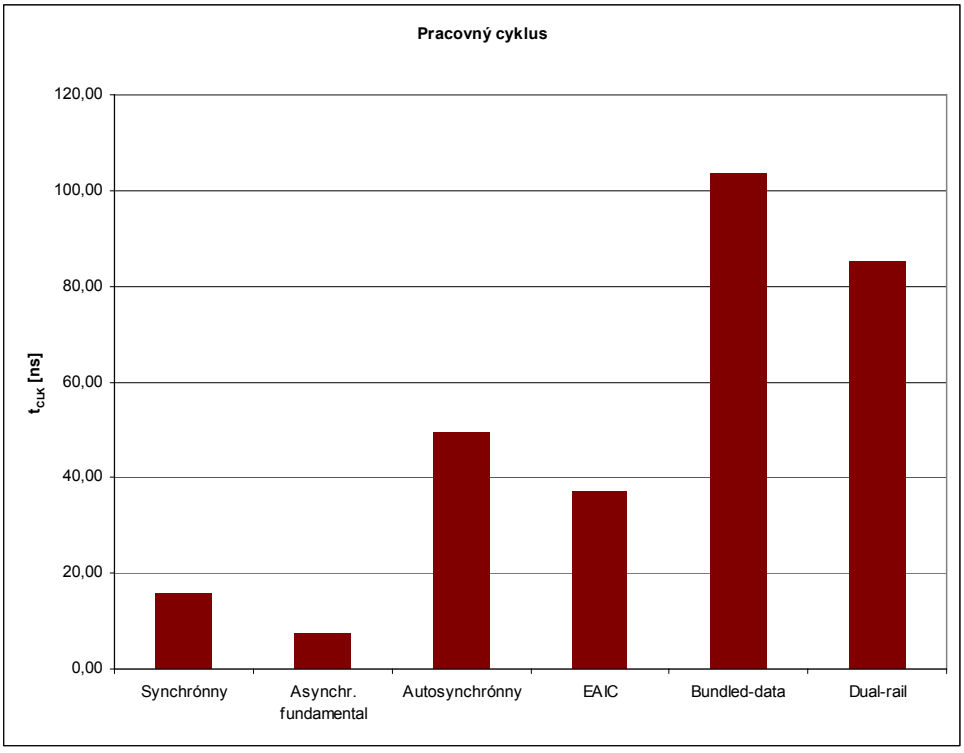
$$t_{DROCL} \le t_{PROP} \le t_{CLK} + t_{DROCL} . \tag{6.16}$$

6.2.7. Zhrnutie a porovnanie parametrov

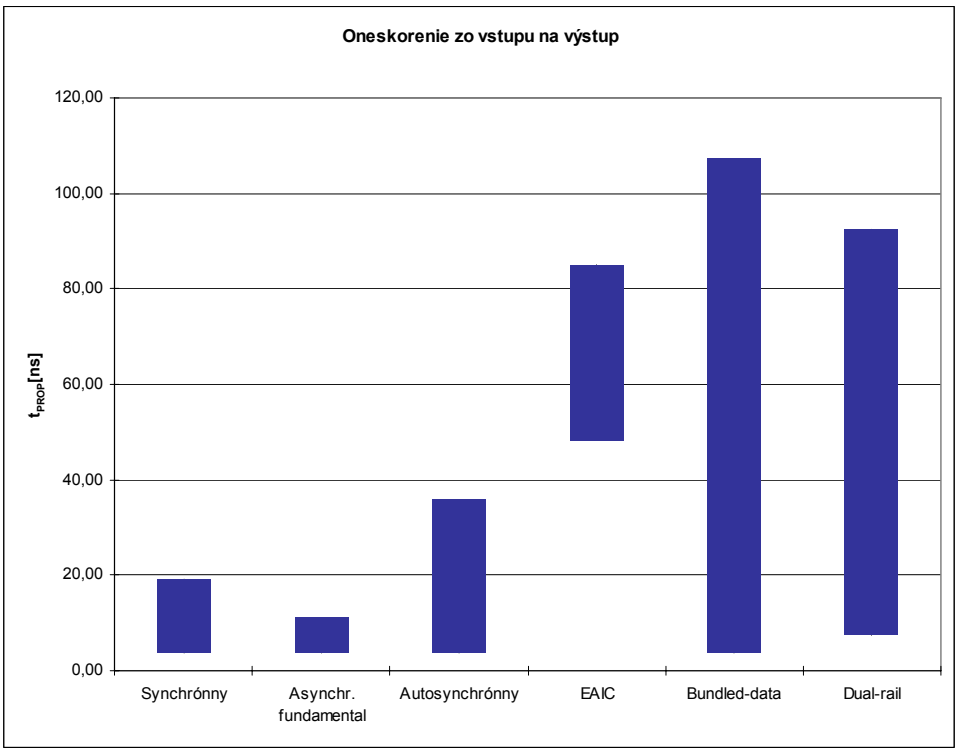
Vyčíslené časové výkonnostné parametre stavových automatov z predchádzajúcich odstavcov sú porovnané a uvedené v tabuľke Tab. 6.4. V druhom stĺpci tabuľky je hodinová perióda, v treťom stĺpci je jej obrátená hodnota v podobe frekvencie hodín, štvrtý a piaty stĺpec udávajú rozsah oneskrenia zo vstupu na výstup. Graf na Obr. 6.16 zobrazuje porovnanie pracovných cyklov jednotlivých stavových automatov a graf na Obr. 6.17 zobrazuje rozsah oneskorenia zo vstupu na výstup. Uvedené hodnoty slúžia len pre účely porovnania, pretože časové parametre boli stanovené na základe zostavenia automatov z diskrétnych hradiel bez optimalizácie.

Tab. 6.4 Porovnanie teoretických hodnôt výkonových časových parametrov pre t_G=3,7 ns

Stavový automat	t_{clk} [ns]	f_{clk} [MHz]	$t_{propmin}$ [ns]	$t_{propmax}$ [ns]
Synchrónny	15,58	64,16	3,70	19,28
Asynchr. fundamental	7,40	135,14	3,70	11,10
Autosynchrónny	49,33	20,27	3,70	35,77
EAIC	37,00	27,03	48,10	85,10
Bundled-data	103,60	9,65	3,70	107,30
Dual-rail	85,10	11,75	7,40	92,50



Obr. 6.16 Porovnanie pracovných cyklov



Obr. 6.17 Porovnanie rozsahu oneskorení automatov zo vstupu na výstup

Najdlhšia doba pracovného cyklu i rozsah oneskorenia zo vstupu na výstup u bundled-data a dual-rail stavových automatov vypovedá o ich zložitej štruktúre. Obsahujú tri registre v spätnej väzbe. Podobne aj stavový automat so štruktúrou EAIC má oneskorenia väčšie pre zložitosť špeciálnych klopných obvodov DFLOP, a pre použitie vstupných registrov navyše. Pracovný cyklus je o 288 % dlhší a maximálne oneskorenie zo vstupu na výstup je o 590 % dlhšie oproti synchrónnemu automatu. Autosynchrónny automat má hodinový cyklus dlhší o 275 % ako synchrónny automat, ale ten je stanovený zložitou kombinačnou logikou pre detekciu ďalšieho stavu. Asynchrónny automat vo fundamentálnom režime mal najkratšiu dobu cyklu (o 19 % kratšiu ako synchrónny automat) i oneskorenie zo vstupu na výstup určené len oneskorením kombinačnej logiky, prípadne výstupnej logiky (o 14 % kratšie ako synchrónny automat). Tento automat ale musí podliehať určitým predpokladom pre fundamentálny režim a všetky jeho stavy musia byť stabilné. Synchrónny automat sa približuje parametrami k asynchrónnemu fundamentálnemu automatu. Oproti ostatným porovnávaným automatom je hodinová perióda synchrónneho automatu definovaná ako minimálna pre spoľahlivú funkciu. U ostatných automatov je ich pracovná perióda cyklu pevne definovaná oneskoreniami.

7 Záver

Práca sa zaoberala analýzou riešenia problémov synchrónnych číslicových obvodov v súčasnosti. Vplyvom rastúcej zložitosti, rýchlosti systémov a zmenšujúcich sa rozmerov spôsobuje hodinový signál nemalé problémy, ktoré nútia návrhárov hľadať iné riešenia od obvodovej až po systémovú úroveň. Najväčším problémom bola spotreba, distribúcia hodín a elektromagnetická kompatibilita. Tieto problémy sa riešili hradlovaním hodinového signálu, fázovaním distribučnej siete hodinového signálu a rozdeľovaním na menšie systémové celky GALS alebo EAIC.

Tieto optimalizačné kroky viedli postupne ku globálnemu kroku – zmene na asynchrónnu doménu, ktorá riešila všetky predchádzajúce problémy. Preto boli v tejto práci zmapované vlastnosti asynchrónnych obvodov a ich návrhu. Návrh asynchrónnych obvodov nepatrí k jednoduchým a často používaným technikám. Otázkou bolo nájdenie jednoduchšieho návrhového riešenia, ktoré minimalizuje vplyvy prechádzajúcich problémov.

Na základe toho boli stanovené ciele práce, ktoré boli zhrnuté v dvoch bodoch:

- analýza autosynchrónnych systémov, ich štruktúry, vymedzenie ich vlastností a parametrov.
- vytvorenie metodiky pre transformáciou zo synchrónnych systémov na autosynchrónne pri minimálnych zmenách.

Výsledkom prvého bodu bola analýza vhodnej štruktúry s optimálnymi vlastnosťami – autosynchrónnych obvodov. Na začiatku sa analyzovali možnosti využiť kódovanie stavov stavových automatov k osamostatneniu ich riadenia prostredníctvom pridanej kombinačnej logiky pre generovanie lokálneho hodinového signálu. Na základe tejto analýzy bola vytvorená jednoduchá efektívna metodika postupu návrhu, ktorá bola ilustrovaná na dvoch možných príkladoch kódovania 1 z N a Grayovho kódovania stavového automatu. Pre overenie tejto návrhovej metodiky boli vytvorené simulačné modely a odsimulované obidve možnosti kódovania v behaviorálnej simulácii i v post-route simulácii na príklade stavového automatu.

Na základe predchádzajúcich simulácií boli ďalej definované časové parametre dôležité pre návrh autosynchrónnych stavových automatov. Pre tieto parametre boli definované vzťahy, ktorých splnenie zaručuje korektnú činnosť automatov so stabilnými aj nestabilnými stavmi. Výsledkom bolo odvodenie hodinových cyklov pre automaty pri prechode cez stabilný aj nestabilný stav. Pre 50 % striedu budú mať obidve varianty rovnaký hodinový cyklus:

$$t_{CK} = 2.(t_{DG} + t_{CO}) .$$

Hodinový cyklus pri prechode cez nestabilný stav je pevne daný oneskoreniami kombinačnej logiky, ale hodinový cyklus pri prechode cez stabilný stav môže byť o určitú časovú rezervu kratší. Táto rezerva zabezpečuje minimálny ochranný interval medzi zostupnou hranou prechádzajúceho hodinového impulzu a nástupnou hranou nasledujúceho hodinového impulzu.

Ďalej bol riešený druhý bod stanovených cieľov. Pre jednoduchý návrh týchto obvodov bola vytvorená metodika efektívnej transformácie RTL popisu synchrónneho stavového automatu vo VHDL na autosynchrónny automat pri minimálnych zmenách. Výhodou tejto transformácie je jednoduchosť popisu synchrónneho automatu a jednoduchosť prevodu do autosynchrónnej verzie pridaním časti kombinačnej logiky. Použitie transformácie šetrí čas pri návrhu, pretože využíva jednoduchosť a zvyklosť navrhovať synchrónny obvod a jednoducho ho previesť na autosynchrónny obvod.

Pre overenie transformácie a metodiky návrhu autosynchrónnych obvodov boli porovnané transformované stavové automaty s ich originálnymi synchrónnymi predlohami v parametroch ako sú plocha čipu, prúdová spotreba a časovanie. Celkovo je možné zhodnotiť, že autosynchrónne automaty majú z pohľadu obsadenej plochy na čipe horšie parametre kvôli svojej prídavnej kombinačnej logike. Počet LUT tabuliek bol u šesť-stavového, resp. pätnásť-stavového automatu väčší o 85 %, resp. 34 % oproti synchrónnemu automatu. Z pohľadu spotreby zase majú najlepšie parametre kvôli nevyužitiu globálnych hodín. U príkonu logiky mal šesť-stavový, resp. pätnásť-stavový automat deväťkrát, resp. dvakrát nižšiu spotrebu. Z pohľadu časovania majú podobné, prípadne lepšie vlastnosti u zložitejších konštrukcií. Minimálna perióda zmien je u šesť-stavových automatov rovnaká, na druhej strane u pätnásť-stavového autosynchrónneho automatu je nižšia o 25 % oproti synchrónnemu automatu. Oneskorenia od vstupu na výstup sú väčšie u autosynchrónnych automatov o 34 % u šesť-stavového, resp. o 43 % u pätnásť-stavového automatu. Na základe výsledkov porovnania je možné povedať, že rozdiel v parametroch medzi autosynchrónnymi a synchrónnymi automatmi klesá s rastúcou zložitosťou ich konštrukcií.

Na záver práce bolo vytvorené teoretické porovnanie rôznych typov synchronizácií (synchrónny, autosynchrónny, asynchrónny vo fundamentálnom režime, EAIC, Bundled-data, Dual-rail) na jednom príklade stavového automatu pri rovnakých technologických parametroch. Časové parametre boli určené na základe popisu stavových automatov kombinačnými rovnicami a vlastnosťami riadiacich a pamäťových prvkov. Najdlhšia doba pracovného cyklu i rozsah oneskorenia zo vstupu na výstup u bundled-data a dual-rail stavových automatov vypovedá o ich zložitej štruktúre. Obsahujú tri registre v spätnej väzbe. Podobne aj stavový automat so štruktúrou EAIC má oneskorenia väčšie pre zložitosť špeciálnych klopných obvodov DFLOP, a pre použitie vstupných synchronizačných registrov navyše. Autosynchrónny automat má hodinový cyklus dlhší o 275 % ako synchrónny automat, ale ten je stanovený zložitou kombinačnou logikou pre detekciu ďalšieho stavu. Asynchrónny automat vo fundamentálnom režime mal najkratšiu dobu cyklu (o 19 % kratšiu ako synchrónny automat) i oneskorenie zo vstupu na výstup určené len oneskorením kombinačnej logiky, prípadne výstupnej logiky (o 14 % kratšie ako synchrónny automat). Tento automat ale musí podliehať určitým predpokladom pre fundamentálny režim a všetky jeho stavy musia byť stabilné.

Práca priniesla výsledky v podobe vytvorenia novej návrhovej metodiky pre autosynchrónne stavové automaty, ktorej prínos je z pohľadu riešenia otázok synchrónnych stavových automatov sľubný. Ďalej práca priniesla výsledok v podobe vytvorenia metodiky transformácie pre jednoduchšie vytvorenie autosynchrónnych obvodov. Jej prínos je v efektívnom vytváraní konštrukcií s použitím štandardných vývojových postupov a návrhových nástrojov.

8 Literatúra

[1] SPARSO, J., FURBER, S. *Principles of Asynchronous Circuit Design - A System Perspective*. Boston: Kluwer Academic Publishers, 2001. ISBN 0-7923-7613-7.

[2] MYERS, J. C. *Asynchronous Circuit Design*. John Wiley & Sons, 2001. ISBN 0-471-22414-6.

[3] TINDER, F. R. *Engineering Digital Design*, Second Edition, Revised. Academic Press-Elsevier, 2000. 884 p. ISBN-13: 978-0-12-691295-1.

[4] WAKERLY, F. J. *Digital Design Principles & Practices*, Third Edition. Prentice Hall, 1999. 678 p. ISBN 0-13-769191-2.

[5] FUHRER, M. R., LIN, B., NOWICK, M. S. Algorithms for the Optimal State Assignment of Asynchronous State Machines. In *16th Conference on Advanced Research in VLSI*. Chapel Hill, North Carolina, USA, 2001, p. 59-75. ISBN-13: 9780818670473.

[6] FANT, K., BRANDT, S. Null Convention Logic, A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis, In *International Conference on Application Specific Systems, Architectures, and Processors (ASAP '96)*. Chicago, Illinois, 1996, p. 261–273. ISBN: 0-8186-7542-X.

[7] HULGAARD, H., BURNS, M. S., BORRIELLO, G. *Testing Asynchronous Circuits: A Survey: Technical report*. Seattle, Washington. University of Washington. 1994. 23 p.

[8] DAVIS, J., REESE, R. *Finite State Machine Datapath Design, Optimization, and Implementation*. Morgan & Claypool, 2008. ISBN 1598295306.

[9] REICHARDT, J., Schwarz, B. *Digital Systems: Handout*, 3rd Edition. Hamburg. University of Applied Sciences. 2004. 150 p.

[10] KOVÁČ, M., KUBÍČEK, M. Asynchronous logical system simulation in VHDL. In *Proceedings of the 18th International Conference Radioelektronika 2008*. Praha: Czechoslovakia Section IEEE, 2008. p. 199-202. ISBN 978-1-4244-2087-2.

[11] KOVÁČ, M. Asynchronous Microcontroller Simulation Model in VHDL. In *World Congress on Science, Engineering and Technology (WCSET 2008)*. 2008, vol. 35, p. 183-186. ISSN: 2070-3740.

[12] STROLLO, A. G. M., NAPOLI, E., DE CARO, D. New Clock-Gating Techniques for Low-Power Flip-flops. In *Proceedings of the 2000 international symposium on Low power electronics and design*, 2000, p. 114-119, ISBN: 1-581113-190-9.

[13] EMNETT, F., BIEGEL, M. Power Reduction Through RTL Clock Gating. SNUG, San Jose, 2000.

[14] BLUNNO, I., NARBONI, A. G., PASSERONE, C. An automated metodology for low electro-magnetic emissions digital circuits design. In *Euromicro Symposium on Digital System Design (DSD'04)*, Rennes, France, 2004, p. 540-547.

[15] PANDINI, D., REPETTO, A., SINISI, V. Clock Distribution Techniques for Low-EMI Design. In *17th International Workshop PATMOS 2007*, Gothenburg, Sweden, 2007, p. 201-210, ISBN 978-3-540-74441-2.

[16] BLUNNO, I., GREGORETTI, F., PASSERONE, C., PERETTO, D., REYNERI, L. M. Designing Low Electro Magnetic Emissions Circuits through Clock Skew Optimization. In *9th IEEE International Conference on Electronics, Circuits and Systems*, Dubrovnik, Croatia, 2002, p. 417-420.

[17] MARTIN, J. A., NYSTRÖM, M. Asynchronous Techniques for System-on-Chip Design. In *Proceedings of the IEEE*, Volume 94, Number 6, 2006, p. 1089-1120, ISSN: 0018-9219.

[18] HEMANI, A., MEINCKE, T., KUMAR, S. Lowering power consumption in clock by using Globally Asynchronous Locally Synchronous design style. In *Proceedings of the 36th ACM/IEEE conference on Design automation*, New Orleans, Louisiana, United States, 1999, p. 873-878, ISBN:1-58133-109-7.

[19] VANSCHEIK, S. W., TINDER, F. R. High Speed Externally Asynchronous/Internally Clocked Systems. In *IEEE Transactions on Computers*, Volume 46, Issue 7, 1997, p. 824-829, ISSN: 0018-9340.

[20] CORVINO, D., EPICOCO, I., FERRANDI, F., FUMMI, F., SCIUTO, D. Automatic VHDL Restructuring for RTL Synthesis Optimization and Testability Improvement. In *Proceedings IEEE ICCD*, 1998, p. 587-596.

[21] KUUSILINNA, K., LAHTINEN, V., HAMALAINEN, T., SAARINEN, J. Finite state machine encoding for VHDL synthesis. In *IEEE Computers and Digital Techniques*, Volume 148, Issue 1, 2001, p. 23-30.

[22] GOLSON, S. State machine design techniques for Verilog and VHDL. In *Synopsys Users Group Conference (SNUG)*, San Jose, 1994, p. 1-48.

[23] BRANOVER, A., KOL, R., GINOSAR, R. Asynchronous Design by Conversion: Converting Synchronous Circuits into Asynchronous Ones. In *IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE'04)*, Volume 2, 2004, p. 870-875.

[24] NIELSEN, F. S., SPARSO, J., MADSEN, J. Behavioral Synthesis of Asynchronous Circuits Using Syntax Directed Translation as Backend. In *IEEE Transactions on VLSI Systems*, Volume 17, Issue 2, 2009, ISSN: 1063-8210.

[25] CORTADELLA, J., KONDRATYEV, A., LAVAGNO, L., SOTIRIOU, P. CH. Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications. In *IEEE Transactions on computer-aided design of integrated circuits and systems*, Sonoma, CA, USA. Volume 25, Issue 10, 2006, p. 1904-1921, ISSN: 0278-0070.

[26] CARMONA, J., COLOM, J-M., CORTADELLA, J., GARCIA-VALLES, F. Synthesis of Asynchronous Controllers Using Integer Linear Programming. In *IEEE Transactions on computer-aided design of integrated circuits and systems*, Sonoma, CA, USA. Volume 25, Issue 9, 2006, p. 1637-1651, ISSN: 0278-0070.

[27] RUTTEN, J.W.J.M., VAN EIJK, C.A.J. Asynchronous Counters for Low Power Applications. In *Proceedings of the ProRISC Workshop on Circuits, Systems and Signal Processing*, 1997.

[28] SMITH, R., LIGTHART, M. High-Level Design for Asynchronous Logic. In *Proceedings of the 2001 Asia and South Pacific Design Automation Konference*, Yokohama, Japan, 2001, p. 431-436, ISBN:0-7803-6634-4.

[29] AL-ASSADI, K. W., KAKARLA, S. Design for Test of Asynchronous NULL Convention Logic (NCL) Circuits. In *Journal of Electronic Testing: Theory and Applications*, Volume 25, Issue 1, 2009, p. 117-126. ISSN:0923-8174.

[30] SACKER, M., BROWN, D. A., WILSON, R. P., RUSHTON, J. A. A General Purpose Behavioural Asynchronous Synthesis System. In: *International Symposium on Asynchronous Circuits and Systems*, 2004, Greece.

[31] MOORE, S. W., TAYLOR, G. S., CUNNINGHAM, P. A., MULLINS, R. D., ROBINSON, P. Using Stoppable Clocks to Safely Interface Asynchronous and Synchronous Subsystems. In *AINT (Asynchronous INTerfaces)*, Delft, Netherlands, July 2000.

[32] BREJ, CH. An automatic synchronous to asynchronous circuit convertor. In *the 11th UK Asynchronous Forum*, 2001.

[33] DAVIS, A., NOWICK, S. M. *An Introduction to Asynchronous Circuit Design*, Technical Report UUCS-97-013, Computer Science Department, University of Utah, 1997.

[34] NOWICK, S. M. *Automatic synthesis of burst-mode asynchronous controllers*, Technical Report CSL-TR-95-686, Department of Electrical Engineering and Computer Science, Stanford University, 1995.

[35] TAYLOR, M. S. *Data-driven handshake circuit synthesis*. PhD thesis, Department of Computer Science, University of Manchester, 2007. 223 p.

[36] BARDSLEY, A. *Implementing Balsa Handshake Circuits*. PhD thesis, Department of Computer Science, University of Manchester, 2000.

[37] BREJ, C. F., GARSIDE, J. D. Reduction in synchronisation in bundled data systems. In *15th UK Asynchronous Forum*, University of Cambridge, 2004.

[38] TOMS, B. W. *Synthesis of Quasi-Delay-Insensitive Datapath Circuits*. Ph.D thesis, Department of Computer Science, University of Manchester, 2006.

[39] OBERG, J., PLOSILA, J., ELLERVEE, P. Automatic Synthesis of Asynchronous Circuits from Synchronous RTL Descriptions. In *Proceedings of the 23rd NORCHIP Conference*, 2005. p. 200-205. ISBN: 1-4244-0064-3.

[40] SMITH, D. J. *HDL Chip Design: A practical guide for designing, synthesizing and simulating ASICs and FPGAs using VHDL or Verilog*. Doone Publications, Madison, AL, USA, 1996, 456 p. ISBN 0-9651934-3-8.

[41] ŠŤASTNÝ, J. *Návrh obvodů založených na programovatelných hradlových polích*. Automatizace, květen 2008, roč. 51, č. 5, s. 317-321.

[42] GARAVEL, H., SALAUN, G., SERVE, W. On the semantics of communicating hardware processes and their translation into LOTOS for the verification of asynchronous circuits with CADP. *Science of Computer Programming*, 2009, Volume 74, Issue 3, 100-127 p., ISSN:0167-6423.

[43] MARTIN, J. A. *Communicating Hardware Processes* [online], 1995, [cit. 2009-08-04]. Available from http://www.cse.ttu.edu.tw/~cheng/courses/async/readings/chp95.pdf.

[44] KESSELS, J., PEETERS, A. The Tangram Framework: Asynchronous Circuits for Low Power, In *Proceedings of the 2001 conference on Asia South Pacific design automation*, p.255-260, January 2001, Yokohama, Japan. ISBN: 0-7803-6633-6.

[45] EDWARDS, D., BARDSLEY, A. BALSA - AN ASYNCHRONOUS HARDWARE SYNTHESIS SYSTEM. *The Computer Journal*. Oxford University Press, 2002, Volume 45, Issue 1. p. 12-18.

[46] BARDSLEY, A., EDWARDS, D. The Balsa Asynchronous Circuits Synthesis System. In *Forum on Design Languages*, September 2000.

[47] TUOMINEN, J., SANTTI, T., PLOSILA, J. *Feasibility Report on Asynchronous Synthesis*. Technical Report 765, TUCS, Apr 2006.

[48] Handshake Solutions. *Haste: Programming Language Manual*. 2008.

[49] MURATA, T. Petri Nets: Properties, Analysis and Applications. In *Proceedings of the IEEE*, 1989, Volume 77, Number 4, p.541-580.

[50] STARODOUBSTEV, A. N., YAKOVLEV, V. A., PETROV, Y. S. *Use of VHDL Enviroment for Interactive Synthesis of Asynchronous Circuits*. Department of Computing Science, University of Newcastle upon Tyne, 1995.

[51] SHANG, D., et al. Asynchronous system synthesis based on direct mapping using VHDL and Petri nets. In *IEEE Proceedings of Computers and Digital Techniques*, 2004, Volume 151, Number 3. 209-220 p.

[52] EDWARDS, D., et al. *Balsa: A Tutorial Guide* [online]. printed 19/05/2006, ver. 3.5, [cit. 2009-08-04], Available from ftp://ftp.cs.man.ac.uk/pub/amulet/balsa/3.5/BalsaManual3.5.pdf.

[53] SAIFHASHEMI, A., BEEREL, P. High Level Modeling of Channel-Based Asynchronous Circuits Using Verilog. In *28th WOTUG(World Occam and Transputer User Group) Technical Meeting*. Eindhoven, 2005.

[54] CORTADELLA, J., et al. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE transactions on information and systems*. 1997, p. 315-325.

[55] ENDECOTT, P., FURBER, S. Modelling and Simulation of Asynchronous Systems using the LARD Hardware Description Language. In *Proceedings of the 12th European Simulation*

Multiconference on Simulation - Past, Present and Future. SCS Europe, 1998. p. 39-43. ISBN:1-56555-148-6.

[56] HOARE, R. A. C. *Communicating Sequential Processes* [online]. 2004, [cit. 2009-08-06]. Available from http://www.usingcsp.com/cspbook.pdf.

[57] GARNICA, O., LANCHARES, J., HERMIDA, R. A New Methodology to Design Low-Power Asynchronous Circuits. In *Proceedings of the 12th International Workshop on Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*. Springer-Verlag, London, 2002. p. 108-117. ISBN:3-540-44143-3.

[58] LLOYD, W. D., GARSIDE, D. J. A Practical Comparison of Asynchronous Design Styles. In *Proceedings Seventh IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'01)*, Salt Lake City, Utah, 2001. p. 36-45.

[59] HAUCK, S. Asynchronous Design Methodologies: An Overview. In *Proceedings of the IEEE*, Vol. 83, No. 1, 1995, p. 69-93.

[60] LIGTHART, M., FANT, K., SMITH, R., TAUBIN, A., KONDRATYEV, A. Asynchronous design using commercial HDL synthesis tools. Advanced Research in Asynchronous Circuits and Systems. In *Proceedings of Sixth International Symposium ASYNC*, 2000. p. 114-125. ISBN: 0-7695-0586-4.

[61] SOKOLOV, D. *Automated synthesis of asynchronous circuits using direct mapping for control and data paths*, Technical Report Series NCL-EECE-MSD-TR-2006-111, School of Electrical, Electronic & Computer Engineering, University of Newcastle, 2006.

[62] DEDOU, J.O., CHILLET, D., SENTIEYS, O. Behavioral synthesis of asynchronous systems: a methodology. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, Volume 6, Issue , 1999, p. 370-373 vol.6. ISBN: 0-7803-5471-0.

[63] KONDRATYEV, A., LWIN, K. Design of Asynchronous Circuits Using Synchronous CAD Tools. In *IEEE Design & Test of Computers*, Volume 19, Issue 4, 2002, p. 107–117, ISSN: 0740-7475.

[64] EDWARDS, A. D. *Design, Automation and Test for Asynchronous Circuits and Systems*. 3rd ed. Technical Report, 2004.

[65] SEMENOV, A., KOELMANS, A. M., LLOYD, L., YAKOVLEV, A. Designing an asynchronous processor using Petri nets. In *Proceedings of the IEEE Micro*, 1997, Volume: 17, Issue: 2, p. 54-64, ISSN: 0272-1732.

[66] SUTHERLAND, E. I., LEXAU, K. J. Designing Fast Asynchronous Circuits. In *Proceedings Seventh IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'01)*. 2001, p. 184–193.

[67] JOSEPHS, B. M., NOWICK, M. S., VAN BERKEL, H. C. Modeling and Design of Asynchronous Circuits. In *Proceedings of the IEEE*, Vol. 87, No. 2, 1999, p. 234–242.

[68] FANT, M. K., BRANDT, A. S. *NULL Convention Logic*. Theseus Research, 2002.

[69] SMITH, S. J. M. *Application-Specific Integrated Circuits*. Addison-Wesley, 1997. ISBN 0-201-50022-1.

[70] CHEN, W. *The VLSI Handbook*. CRC Press LLC, 2000, ISBN 0-8493-8593-8.

[71] BREEDING, J. K. *Digital Design Fundamentals*. Second Edition. Prentice Hall, 1992. 460 p.

[72] WAKERLY, F. J. *Digital Design Principles and Practices*. Third Edition. Prentice Hall, 1999, 830 p. ISBN 0-13-769191-2.

[73] SOKOLOV, D., et al. Design and Analysis of Dual-rail Circuits for Security Applications. In *IEEE Transactions on Computers*, Vol. 54, No. 4, 2005, p. 449-460. ISSN: 0018-9340.

Zoznam skratiek

ACK	potvrdenie (*acknowledge*)
ASIC	aplikačne špecifické zákaznické integrované obvody (*application-specific integrated circuit*)
CAD	počítačom podporovaný návrh (*computer-aided design*)
CHP	procesy pre komunikáciu hardvéru (*communicating hardware processes*)
CMOS	komplementárny polovodič typu kov-oxid (*complementary metal-oxid semiconductor*)
CSP	komunikácia sekvenčnými procesmi (*communicating sequential processes*)
DI	obvody necitlivé na oneskorenie (*delay-insensitive*)
EAIC	externe asynchrónne vnútorne taktované systémy (*externally asynchronous internally clocked*)
EME	elektromagnetické emisie (*elektromagnetic emission*)
FF	klopný obvod (*flip-flop*)
FIFO	posuvný register (*first in first out*)
FPGA	hradlové pole (*field programmable gate array*)
GALS	globálne asynchrónne lokálne synchrónne systémy (*globally asynchronous locally synchronous*)
GCL	jazyk chránených príkazov (*guarded command language*)
HDL	jazyk pre popis hardvéru (*hardware description language*)
LUT	priraďovacia tabuľka (*look-up table*)
MIC	viacnásobná zmena vstupov (*multiple input change*)
MOS	polovodič typu kov-oxid (*metal-oxid semiconductor*)
NCL	logika nulovej konvencie (*null convention logic*)
NRZ	signál bez návratu k nule (*non return to zero*)
PLL	fázová slučka (*phase locked loop*)
PN	Petriho siete (*Petri nets*)
QDI	obvody takmer necitlivé na oneskorenie (*quasi delay-insensitive*)
REQ	požiadavka (*request*)
RTL	úroveň abstrakcie pre medziregistrové prenosy (*register transfer level*)
SB	synchrónny blok (*synchronous block*)
SI	obvody nezávislé na rýchlosti (*speed-independent*)
SoC	systém na čipe (*system on chip*)

ST	samospúšťané obvody (*self-timed*)
STG	graf signálových prechodov (*signal transition graph*)
TS	prechodové systémy (*transition systems*)
VHDL	jazyk pre popis hardvéru veľmi rýchlych integrovaných obvodov (*very high-speed integrated circuit hardware description language*)
VLSI	veľmi veľká miera integrácie (*very-large-scale integration*)
XST	nástroj pre syntézu firmy Xilinx (*Xilinx synthesis tool*)

Zoznam symbolov

P_D	dynamický stratový výkon
C	kapacita
F, f	frekvencia
t	čas
P	výkonová spotreba
A	plocha
δ	oneskorenie
l	dĺžka vodičov
V_{dd}	napájacie napätie

Curriculum Vitae

Name: Michal KOVÁČ

Born: October 21th 1982 in Piešťany

Contact: xkovac03@stud.feec.vutbr.cz

Education

2006 – 2009 **Technical University of Brno / Department of Radio Electronics**
Ph.D. study of Electronics
State exam passed in June 2008

2001 – 2006 **Technical University of Brno / Department of Radio Electronics**
Pre-graduate study of Electronics and Communication
State exam passed in June 2006
Diploma thesis BPSK, QPSK Modulator and Demodulator defended in June 2006

Experience

7/05 – 8/05 **TEV-EV Piešťany**
study stay
realisation of electronic control circuits for lamps

Languages

English

Specialization

FPGA and CPLD programmable logic devices.