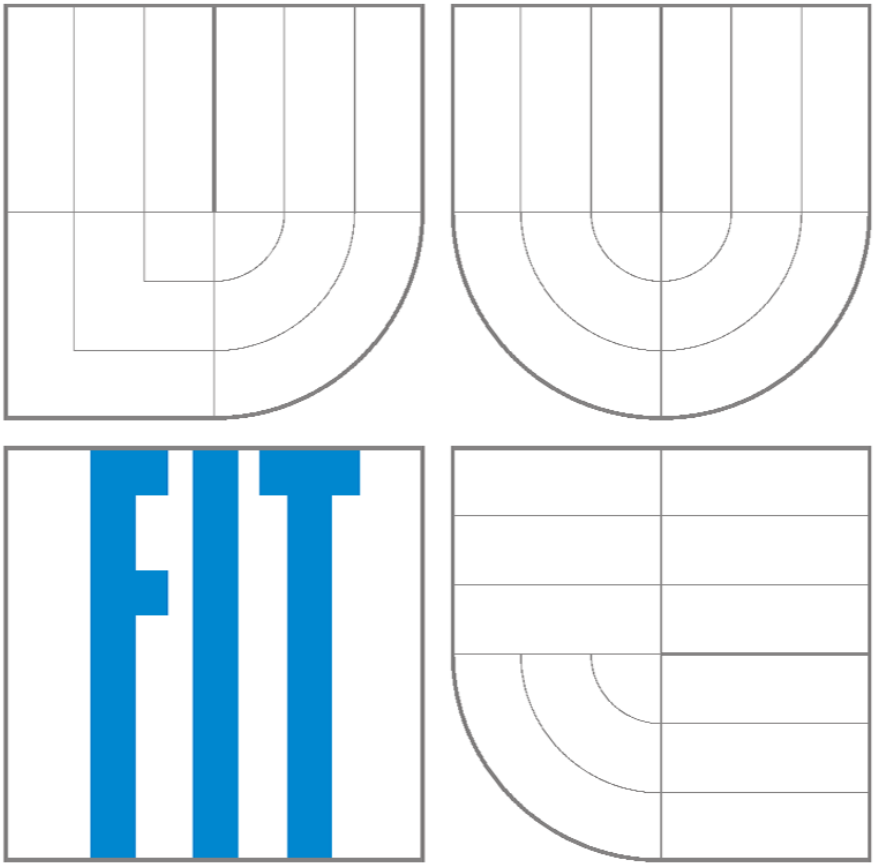


Simulátor číslicových obvodů

Dokumentace projektu pro předmět Modelevání a simulace

Martin Knapovský - xknapo02@stud.fit.vutbr.cz
Tomáš Smetka - xsmetk01@stud.fit.vutbr.cz
Brno, 2011

Obsah

1. Úvod

Tato práce se zabývá problematikou simulace číslicových obvodů na číslicových počítačích a podrobně prochází postupy použité při návrhu, modelování a implementaci modelu, který tyto simulace provádí. Cílem bylo vytvořit **efektivní** a přitom **jednoduchý** a **přenositelný** nástroj pro simulaci těchto obvodů pomocí volně dostupných technologií. Pomocí modelu a simulačních experimentů bude ukázáno chování číslicových obvodů v daných situacích.

Prerekvizity

Vzhledem k větší náročnosti na abstraktní uvažování není téma číslicových obvodů určeno každému. K pochopení tohoto tématu je žádoucí, aby čtenář disponoval určitými znalostmi z příbuzných oborů. Bezpodmínečně nutná je znalost **Booleovy algebry**, funkce a schématické značky **logických hradel**. Čtenář musí být schopen z těchto hradel vytvořit logické obvody a chápat jejich funkci a účel. Pro porozumění implementačních částí této práce je nutná znalost principů některého z objektově orientovaných **programovacích jazyků** a základní principy **diskrétní simulace**.

1.1 Zdroje informací

V objasnění problematiky logických hradel a simulací nám byli velmi nápomocni pánové **Ing. Michal Bidlo, Ph.D.**, **Dr. Ing. Petr Peringer** a **Ing. Martin Hrubý, Ph.D.**, kterým bychom tímto chtěli **poděkovat**. Literaturou, která nás při návrhu doprovázela téměř na každém kroku se stala skripta předmětu **Návrh číslicových systémů** (*INC*), kde byla podrobně popsána témata, kterým se budeme věnovat v rozboru (kapitola 2). Komletní seznam literatury se nachází na konci této práce. K ověření správného pochopení problematiky byl využit volně dostupný nástroj pro simulaci číslicových obvodů *Logisim*. K návrhu a implementaci analyzátoru vstupního souboru byli využity vědomosti nabyté v předmětu **Formální jazyky a překladače**.

1.2 Experimentální ověřování validity modelu

Validita modelu byla ověřována na základě očekávaných výstupních hodnot logických obvodů binárního dekodéru a master-slave klopného obvodu, které jsou přiloženy k aplikaci.

2. Rozbor tématu a použitých metod/technologií

V této kapitole jsou probrána témata důležitá pro pochopení návrhu abstraktního modelu a následného vytvoření modelu simulačního.

Základní vlastnosti logických obvodů

V číslicových systémech se pracuje se signály s konečným počtem diskrétních hodnot, což je odlišuje od systemů analogových, u kterých jsou signály spojité a mohou nabývat nekonečného počtu hodnot ve vymezeném rozsahu. V číslicovém systemu může i čas být veličinou diskrétní, tj. signály se mohou měnit jen v určitých okamžicích. Takovéto číslicové systémy se pak nazývají **synchronní** – na rozdíl od systémů **asynchronních**, u kterých může ke změnám signálů docházet kdykoliv. Synchronní systémy jsou podstatně častější, neboť existence přesně stanovených okamžiků zavádí „pořádek“ do časování signálů v systému a tím usnadňuje jeho konstrukci i výrobu v podobě integrovaných obvodů. Přesné časování je zajištěno hodinovými impulzy, což je velmi významný signál systému.

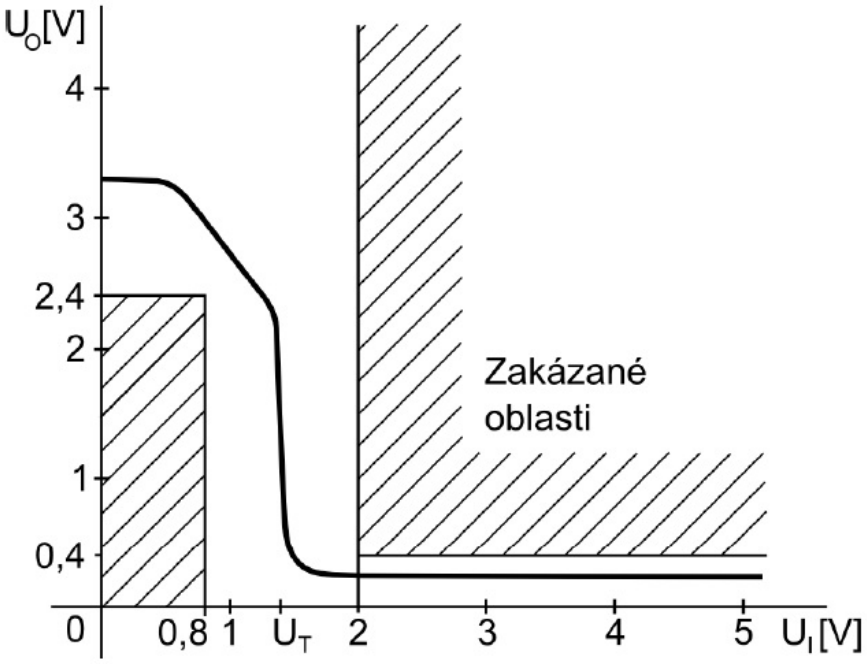
Číslicové systémy je možné rozdělit do dvou skupin – **sekvenční** a **kombinační**. U kombinačních systémů jsou výstupní signály závislé pouze na současných vstupních signálech a u sekvenčních systemů jsou výstupní signály závislé nejen na současných vstupních signálech, ale i na vstupních signálech v minulosti – systém má tedy vnitřní paměť. [Pinker].

Dvoustavové signály

Číslicové součástky jsou napájeny kladným napětím *+Ucc* a existují 2 možnosti mapování diskétních hodnot na napětí.

- **Pozitivní logika** vnímá nižší napětí jako hodnotu “0” a vyšší napětí jako hodnotu “1”
- **Negativní logika** vnímá vyšší napětí jako hodnotu “0” a nižší jako hodnotu “1”

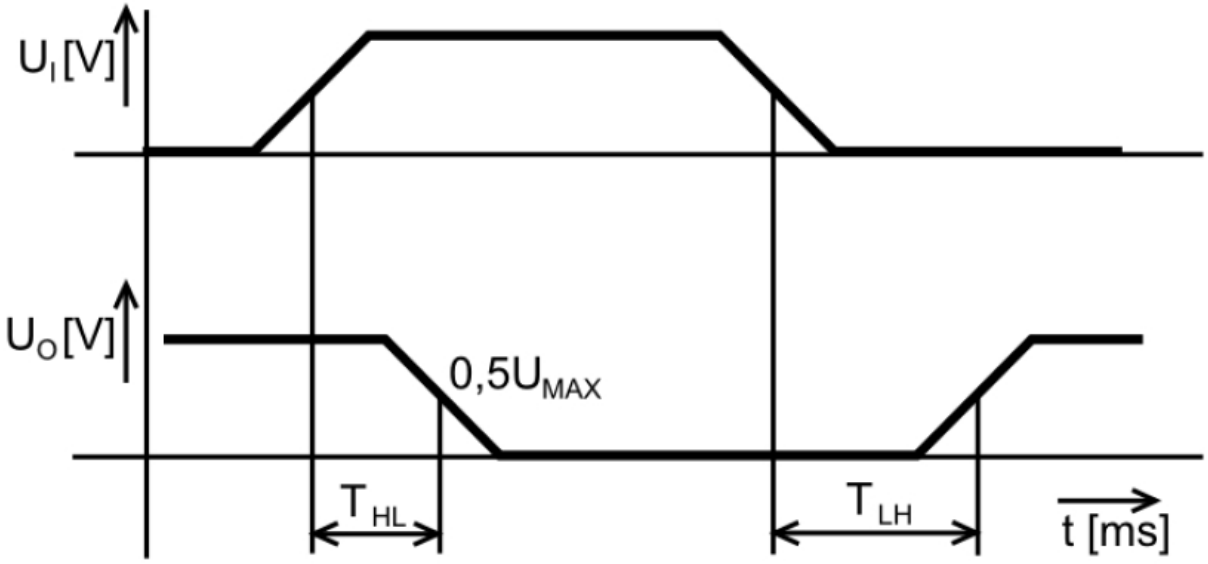
Hodnota signálu, na který je namapována log. 0, nebo 1 se nachází v určitém intervalu, což do výroby součástek vnáší jistou toleranci. Pásmo vyšších napětí se označuje jako *High* a pásmo napětí nižších jako *Low*. Hodnota signálu, která se nachází mezi těmito intervaly se označuje jako *zakázané pásmo*, přes které za ideálních podmínek signál přechází pouze při změně stavu. Při přechodu dochází k zákmitům, do systému mohou pronikat rušivé signály z okolí, nebo je systém může generovat svou činností sám. [Pinker]



Obr. 1 – Intervaly hodnot napětí v číslicových systémech

Dynamické parametry hradel

Dynamické parametry hradla (zpoždění) souvisí s tím, že hradlo nereaguje svým výstupem na přivedený signál okamžitě (přesycování tranzistorů, kapacity přechodů). Zpoždění lze definovat jako časový interval mezi průchodem signálu referenční úrovní na vstupu a na výstupu hradla. [Kolouch]

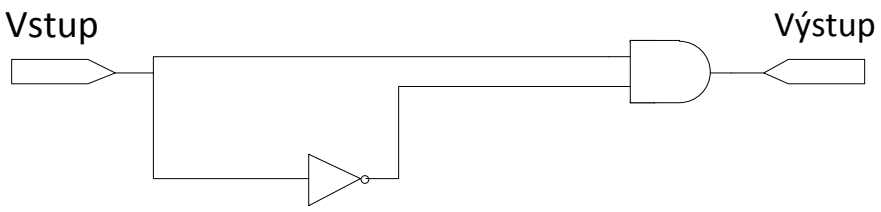


Obr. 2 – Zpoždění hradla

Hodnota časového zpoždění v číslicových obvodech nebývá přesně známa – závisí na teplotě, napájecím napětí a podobných veličinách, což do chování obvodů přináší jistý prvek náhodnosti. V katalozích se uvádí mezní hodnota zpoždění hradel. [Kolouch]

Hazardy

Hazardy jsou způsobeny zpožděním signálu na hradlech, kdy logické obvody, které realizují určité logické funkce, mohou vlivem zpoždění signálu v jednotlivých logických členech po přechodnou dobu vykazovat na výstupech jiné hodnoty, než které odpovídají modelovaným funkcím. Takovou situaci označujeme jako hazard. Harardy jsou dvojího typu – **statické** a **dynamické**, kde statický hazard je takový jev, který může vzniknout v obvodu na základě změny jednoho ze vstupů, pro který logická funkce obvodu nedefinuje změnu výstupu obvodu. Po uplynutí krátkého času se obvod stabilizuje a poskytne již správnou hodnotu. [Zikmund]



Obr. 3 – Ilustrace statického hazardu

Na Obr. 3 je znázorněn kombinační obvod realizující funkci

$$Y = X \,.\, X',$$

kde *Y* je výstupem hradla *AND*, *X* je jeho prvním vstupem a *X'* vstupem druhým, který prochází přes hradlo typu *NOT*. Jakmile dojde ke změně logické hodnoty na vstupu do obvodu, pak se na výstupu z obvodu po dobu zpoždění hradla *NOT* objeví neočekávaná hodnota logické *1*, která se následně opět navrátí do očekávané hodnoty *0*. Tento přechodový stav, který je způsoben zpožděním hradla *NOT* označujeme jako statický hazard. Pokud chceme hazardu předejít, je nutné vyrovnat zpoždění obou větví připojených na vstup hradla *AND* například vložením zpožďujícího prvku do rychlejší větve na vstupu do tohoto hradla. [Zikmund]

Dynamický hazard je pak rozšířením harardu statického. Pokud v obvodu není přítomen žádný statický hazard, pak se nemůže vyskytnout ani hazard dynamický, jelikož ten je propagací jednoho a více statických hazardů obvodem.

Další vlastnosti logických obvodů

Zpoždění běžných číslicových součástek jsou v řádu jednotek *ns*. Při tak krátkých dobách je nutno brát v úvahu i **zpoždění spoje**. Na běžném plošném spoji je zpoždění kolem *1ns* na 10 cm délky spoje, což může hrát významnou roli v rozvodu hodinových pulsů. Dalšími významnými parametry jsou **doba náběhu** a **doba doběhu** impulsu, které jsou obecně definovány při protnutí úrovní 10% a 90% z ustáleného stavu. Doba náběhu a doba doběhu se u běžných číslicových obvodů udáva v jednotkách, nebo zlomcích ns. [Pinker]

Technologie hradel

Prakticky jsou dnes číslicové obvody výhradně vyráběny jako monolitické integrované obvody. Ty mohou být vyráběny v technologiích pracujích s bipolárními tranzistory, s tranzistory unipolárními, nebo i s oběma typy. Obvody jsou členěny do rodin, které jsou dále děleny do technologických variant vzájemně propojitelných obvodů. Kompabilita obvodů různých řad je omezená. V závislosti na použité technologii rozlišujeme dvě hlavní používané technologie – **TTL** a **CMOS**. [Kolouch]

U technologie **TTL** je využito bipolárního tranzistoru, který se vyznačuje vyšší spotřebou, ale zároveň vyšší rychlostí zpracování signálu.

- Napájecí napětí je *5V*
- Signál o hodnotě *0-0,8V* je považován za log. *0* a signál s napětím vyšším jak *2V* jako log. *1*
- Na výstupu je log. *0* reprezentována napětím *0-0,4V* a log. *1* napětím vyšším jak *2,4V*

Technologie **CMOS** používá unipolární tranzistory typu *FET* a *MOSFET* díky nimž mají mnohem nižší spotřebu než TTL hradla, což je vykoupeno nižší rychlostí zpracování signálu (typicky *10ns* pro *NAND* hradlo).

- Napájecí napětí je v intervalu *5-15V*, nebo *3,3V*
- Signál o hodnotě *0-0,3Ucc* je považován za log. *0* a za log. *1* je považován signál s napětím vyšším jak *0,7Ucc*
- Na výstupu reprezentuje log. *0* napětí *0V* a log. *1* napětí *Ucc*

2.1 Popis postupu vytvoření modelu

Popis obvodu

Cílem bylo vytvořit jazyk pro popis obvodu, který by byl uživateli na první pohled srozumitelný a pro popis obvodu jednoduchý. Byl tedy zvolen formát odpovídající specifikaci *XML*, který je v dnešní době velmi rozšířený, jednoduchý na zpracování a zároveň "lidský". Tento popis umožňuje pracovat s dříve popsanými parametry logických obvodů.

Pro přehlednost je možné obvod rozdělit do bloků, které spolu mohou prostřednictvím propojení vzájemně komunikovat. Vstupy obvodu jsou pak popisovány generátory signálu, které umožňují přivést na jakýkoliv počet vstupů hradel periodické či impulsní signály, kterým lze nastavit jejich periodu, resp. délku pulsu a posunutí v simulačním čase. Logická hradla mohou mít 1 a více vstupů a lze jim také nastavit zpoždění v jednotkách simulačního času.

Kompletní syntaxe a popis vstupního souboru lze nalézt v **příloze č. 1**.

Volba prostředí

Uživatelské rozhranní

Jak již bylo uvedeno v úvodu, cílem aplikace bylo vytvořit prostředek, který by pracoval s co možná nejvyšší efektivitou a zůstal ve své koncepci jednoduchý. Proto byl jako výstup programu zvolen **textový** výpis stavu prvků obvodu v jednotlivých simulačních krocích. Tento formát výpisu ovšem nemusí být pro některé z uživatelů příliš přehledný a tudíž byl program doplněn o možnost exportu průběhu simulace do souboru, který lze otevřít a vykreslit programem **GNUPlot**.

Operační systém a vývojové prostředky

Vývojový tým je příznivržencem myšlenky otevřeného softwaru, a tudíž byli pro vývoj zvoleny volně dostupné prostředky prostředí operačního systému Linux.

Pro implementaci přisly v úvahu jazyky Java, C a C++, které jsou dále popsány.

- **Java –** Vysoká produktivita psaní kódu díky svému objektovému přístupu a rozsáhlé knihovny zprostředkovávající nejrůznější i velmi pokročilé funkce. Na každém systému musí být nainstalován interpret tohoto jazyka a výsledný program je charakteristický vyšší paměťovou náročností, což v našem případě rozhodlo v neprospěch tohoto jazyka. [Java Muni]
- **C –** Jazyku C se přičítá snad největší rychlost zkompilovaných programů, což je bezpochyby významný klad. Procedurální programování však ve větších projektech znesnadňuje čitelnost a udržovatelnost kódu [IPP], což je fakt, který by narušil hlavní cíle tohoto projektu.
- **C++ -** Rychlost, kterou disponuje programovací jazyk C, snadná přenositelnost a objektová orientace tohoto jazyka umožňuje vytvořit nástroj, který přímo koresponduje s vytyčenými cíly, a tudíž se stal tento jazyk implementačním prostředkem této aplikace.

2.2 Původ použitých technologií

Pro implementaci bylo využito **standartních knihoven** jazyka C a C++.

- **iostream** - standardní knihovna C++ pro vstup a výstup
- **cstdio** - standardní knihovna C pro vstup a výstup (využito pro tisk do souboru)
- **list** - knihovna C++ pro práci s seznamy
- **ctime** - knihovna pro práci s časem (pro vytváření jména výstupního souboru)

3. Koncepce

Při vytváření konceptuálního modelu je nutno abstrahovat od všech nedůležitých skutečností vzhledem k cíli a účelu modelu. Nedokážeme totiž postihnout reálný svět v celé své komplikovanosti a identifikujeme tak vhodné složky systému, se kterými poté pracujeme. [IMS:43]

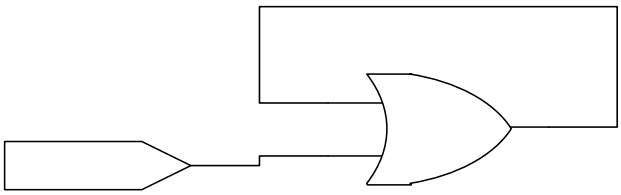
V kapitole 2 byly zmíněny **doby náběhu a doběhu**. Ty jsou však vzhledem ke své délce zanedbatelné a můžeme tak přechod mezi jednotlivými úrovněmi signálu chápat jako **skokový**. Pokud navíc uvažujeme, že se změny v číslicových obvodech provádějí v krocích, pak můžeme využít **diskrétního simulačního modelu**. Jednotlivá hradla jsou pak chápána jako procesy, které jsou posloupností událostí změn na nich vzniklých. [IMS:124]

Jednotka času simulace je zde chápána jako nejkratší okamžik, ve kterém může dojít k nějaké události. V této jednotce je pak měřeno i zpoždění na daném hradle.

Při návrhu modelu bylo abstrahováno od **používané technologie hradel** a tím tedy i možnosti, kdy by byla spojena hradla s různými napěťovými intervaly log. 1, nebo log. 0. Předpokládá se tedy obvod, který je vytvořen z hradel stejné technologie, nebo hradel navzájem kompatibilních.

Taktéž nebyla uvažována možnost rušení signálů vlivem okolního prostředí a vliv teploty na práci obvodu. Předpokládáme obvod v **ideálních podmínkách**.

V modelu je také nutné mimo logických hodnot *0* a *1* uvažovat i **třetí stav**, kdy nevíme, jaká hodnota se aktuálně nachází na vstupu, nebo výstupu hradla. Situace je znázorněna na Obr.4, kde je hradlo typu OR, o kterém nemáme předchozí informace (nevíme jaké hodnoty měl na výstupech a vstupech), a proto je označíme jako **nedefinované**, což je právě onen třetí stav. V okamžiku, kdy bude na vstupu tohoto hradla log. *0*, se bude nedefinovaná hodnota udržovat na jeho výstupu, avšak pokud se hodnota změní na log. *1* alespoň po dobu zpoždění hradla, pak se hodnota na výstupu změní na log. 1 a již se nemění (obvod s pamětí – sekvenční obvod). Více je uvedeno v **příloze 2**.



Obr. 4 – Nedefinovaná hodnota na hradle typu OR

Je také možné abstrahovat od **zpoždění spoje**, jelikož se v číslicových obvodech počítá s tímto jevem a jsou tedy postaveny tak, aby například synchronizační signál, který je rozváděn centrálně zpožděním spoje nezpůsobil nekorektní chování obvodu. [INC]

Pokud bereme v úvahu výše zmíněná fakta, pak je možné sestavit model, který je takovou abstrakcí reality, která realitě odpovídá, avšak zanedbává situace, které nejsou pro model přiliž podstatné a umožňuje vytvořit nástroj, který je **efektivní** v tom, co má dělat.

3.1 Způsob vyjádření modelu

Konceptuální model je vyjádřen **slovním popisem** a **diagramem**, který abstraktně popisuje práci simulátoru. V úvahu připadalo také vyjádření pomocí petriho sítě, avšak ta je v tomto případě mnohem méně přehledná. V modelu je totiž mnoho podmínek, podle kterých se model řídí a tyto podmínky jsou petriho sítí težko modelovatelné.

3.2 Konceptuální model

Model je koncipován tak, že rozlišuje typy prvků na **generátory** a **hradla**. Zpracování těchto prvků je plánováno do **kalendáře událostí**, který určuje jaký prvek se má aktuálně zpracovávat. Zpočátku jsou do kalendáře událostí umístěny prvky generující **vstupní signál** pro simulovaný obvod. Generátory signálu poté svým výstupem určují vstup pro prvky, které jsou k nim připojeny a zároveň se **sami naplánují** do kalendáře událostí na čas změny svého výstupu. Hradla, kterým jsou **záslány zprávy** o změně na jejich vstupu rozhodnou zda mají být naplánovány do kalendáře událostí. Není totiž nutné plánovat hradla, která nemění svou výstupní hodnotu – takovýmto přistupem jsou velmi sníženy nároky na výpočetní výkon a paměťový prostor, který výsledná aplikace vyžaduje od stroje, na kterém simulace probíhá. Pokud však aktuální vstupní hodnoty produkují jinou výstupní hodnotu, pak je hradlo naplánováno do kalendáře událostí na čas který odpovídá součtu aktuálního času simulace a zpoždění hradla. Ve chvíli, kdy dojde ke zpracování tohoto hradla je jeho výstup změněn na hodnotu, která je dle daného typu hradla výsledkem zpracování vstupních hodnot v čase, kdy bylo hradlo naplánováno.

Hradlo však může mít neomezený počet vstupů a tudíž může být plánováno neomezeným počtem pvků jemu předcházejících. Prvek tedy bude v kalendáři událostí umístěn na více místech současně, avšak pokaždé budou 2 za sebou chronologicky jdoucí události jednoho hradla produkovat jinou výstupní hodnotu – jak již bylo dříve uvedeno, není důvod plánovat hradlo, na jehož výstupu neproběhla změna.

Zpracování prvků je pak prováděním naplánovaných událostí v kalendáři. V okamžik, kdy již na daný čas simulace není v kalendáři naplánována žádná událost, se provede krok, který reprezentuje posun v simulačním čase.

Diagram konceptuálního modelu je pro svou velikost vyčleněn do **přílohy 3**. Tento diagram vyjadřuje konceptuální návrh simulátoru, který byl využit k jeho implementaci. Implementací a architekturou simulátoru se zabývá následující kapitola.

Návrh formátu vstupního souboru

Cílem bylo popsat obvod tak, aby byl uživateli na první pohled srozumitelný a zároveň jednoduchý na zpracování programem. Byl tedy zvolen formát odpovídající specifikaci *XML*, který těmto vlastnostem odpovídá a je zároveň velmi rozšířený, čímž u většiny uživatelů odpadá potřeba učit se nový jazyk.

Při vytváření modelu byli vzaty v úvahy veškeré dříve zmíněné vlastnosti logických obvodů, které se promítly do formátu vstupního souboru s popisem prvků a jejich propojení. Kompletní syntaxe vstupního soubrou je uvedena v **příloze 1**.

Návrh analyzátoru vstupního souboru a syntéza obvodu

Vzhledem k tomu, že byl zvolen programu vlastní formát vstupního souboru, bylo nutné navrhnout modul, který by dokázal z tohoto souboru vytvořit takovou reprezentaci obvodu, které simulátor rozumí a dokáže s ní pracovat. Byli identifikovány jednotlivé kroky nutné k dosažení tohoto cíle a byl navžen modul, který tuto činnost vykonává. Jednotlivé kroky jsou následující:

- provést **lexikální** analýzu vstupního souboru,
- výsledek lexikální analýzy využít při analýze **syntaktické**,
- v průběhu syntaktické analýzy provádět analýzu **sémantickou**
- dle rozpoznaných prvků vytvářet vnitřní reprezentaci obvodu.

Diagram znázorňující práci tohoto modulu je možné nalézt v **příloze 4**.

4. Architektura simulačního modelu/simulátoru

Simulátor

Simulátor byl implementován na základě konceptuálního modelu uvedeného v kapitole 3. Byli vytvořeny **třídy** odpovídající generátorům a hradlům.

Objekt třídy *generator* si uchovává informace o svém unikátním identifikačním **čísle**, **typu**, aktuální **hodnotě na výstupu** a dále pak **délku impulsu**, kterou vytváří a **čas**, o který je tento impuls posunutý. Počáteční hodnoty těchto parametrů jsou nastavovány analyzátorem vstupního souboru, jehož implementace je popsána níže. Dále jsou v této třídě implementovány metody, které umožňují pracovat s generátory pomocí zasílání zpráv. Mezi ty nejdůležitější, generátoru specifickoé, patří metoda pro jeho nastavení, se kterou pracuje jak simulátor, tak právě analyzátor vstupního souboru.

Objekt třídy *gate* si taktéž uchovává informace o svém unikátním identifikačním **čísle**, **typu, aktuální hodnotě výstupu** a **zpoždění**. Hradlo navíc oproti generátoru obsahuje vstupy, kterých může být neomezené množství. Je tudíž možné simulovat obvody, které by bylo reálně složité realizovat. Tato třída také obsahuje metody, které reprezentují funkčnost hradel a metody nastavující jejich vstupy. Možnostem nastavení generátorů je vyhrazena **příloha 5**.

Oběma třídám jsou společné (nejsou však zděděné, jelikož jsou pro pro obě třídy specifické) metody a seznamy, které ukládají **historii** výstupu generátorů, nebo hradel. Tyto dále slouží pro výpis průběhu simulace. Společné jsou také seznamy, které uchovávají **propojení** s prvky, na které je přiveden výstup daného generátoru, či hradla. V těchto seznamech jsou uchovávány přímo ukazatele na vytvořené objekty, což k nim umožňuje přímý přístup. Odpadá tak vyhledávání v seznamu hradel, což přispívá k **efektivitě simulace**.

Simulační proces probíhá přesně dle návrhu na **diagramu v příloze 3**. Pro práci s generátory a hradly využívá seznamů. K těmto prvkům přistupuje pomocí přímých odkazů, nastavuje vstupy hradel a zprostředkovává jejich plánování do kalendáře událostí.

Analyzátor vstupního souboru

Analyzátor byl taktéž implementován dle návrhu uvedeného v kapitole 3. Jeho diagram je znázorněn v příloze 4. **Lexikální analyzátor** načítá vstupní soubor po znacích a rozpoznává jednotlivé prvky popisu obvodu – **tokeny**. Ty jsou dále předávány **analyzátoru syntaktickému**, který určuje zda se token, rozpoznaný lexikálním analyzátorem, nalézá na správném místě a podle toho určuje svou další činnost. V průběhu syntaktické analýzy se kontroluje **hlavička XML**, dále se kontroluje **syntaxe bloku**, **vnitřního bloku**, **generátoru**, **hradel** a **propojení**, která jsou ve vstupním souboru popsány. Všechna funkčnost je zajištěna konečnými automaty, které dle stavu a přijatého tokenu rozhodují o své další činnosti.

V průběhu analýzy je prováděna **sémantická analýza**, která např. zaručuje, že identifikační číslo prvku bude v celém obvodu unikátní, nebo např. kontroluje, zda byli zadány všechny potřebné parametry pro popis daného prvku.

Během průchodu vstupním souborem dochází k sestavování **vnitřní reprezentace** obvodu a jeho nastavení.

Po ukončení syntaktické analýzy (po získání tokenu reprezentujícího konec vstupního souboru) je proveden **průchod seznamy** hradel a generátorů a dle popsaných propojení jsou **doplněny ukazatele** na existující objekty.

Výstup simulátoru

Výstup je dvojího typu, avšak ve své podstatě pracují oba stejným způsobem. Na konci simulace se prochází historie všech prvků obvodu a v případě volby **textového** výpisu je vytisknuta na standartní výstup *STDOUT*. Pokud uživatel zvolí možnost uložení souboru pro program **GNUPlot**, pak je nutné tento program spustit a pomocí příkazu *load '%vygenerovaný_soubor'* (kde řetězec *%vygenerovany_soubor* je nutno nahradit za název výstupního souboru vytvořeného simulátorem) nahrát výstup simulace. Záhy je pak zobrazen průběh výstupních signálů na všech prvcích obvodu. – GNUPLOT Nefunkční

5. Podstata simulačních experimentů a jejich průběh

Pomocí experimentů chceme potvrdit validitu simulačního modelu. Experimentováním prokázat funkčnost všech hradel, zpracování jejich zpoždění a s tím spojený výskyt statických a dynamických hazardů. Správné plánování a pomocí jednodušších obvodů simulujících všechny potřebné stavy prokázat funkčnost většího komplexního obvodu. Ve výsledku všemi experimenty dokázat, že simulační model odpovídá reálnému chování a může být použit pro simulování obvodů na abstraktní úrovni, aniž by bylo potřeba obvody fyzicky sestavovat.

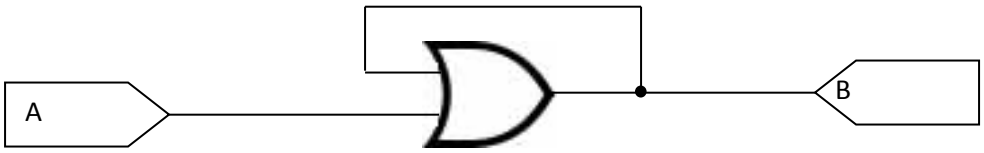
5.1. Postup experimentování

Experimentování probíhalo na řadě jednodušších obvodů, které simulují potřebné chování. Každý obvod byl simulován několikrát a to pokaždé s jinými vstupními parametry (odlišné zpoždění hradel, odlišné délky a logické stavy impulsů od generátorů). V každé nové simulaci s novými parametry probíhala kontrola výstupních hodnot s předpokládanými hodnotami abstraktního modelu. Většina simulací probíhala cíleně, abysme dokázali, že zvolený obvod vykazuje požadované chování (např. statické hazardy, chybovost).

5.2. Dokumentace jednotlivých zajímavých experimentů

5.2.1 Experiment hradla OR jako paměťového prvku

Obvod:



Parametry simulace č.1:

Generátor A: typ – impulzní, délka impulsu – 1, posunutí v čase – 0
Hradlo: typ – OR, zpoždění – 2 ns

Očekávané chování:

- hradlo by si po přijetí prvního signálo o logické hodnotě 1 mělo zpětnou vazbou tuto hodnotu předávat („mělo by si jí zapamatovat“)

Získané výsledky:

A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	2	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20 [ns]

Zhodnocení výsledků:

- Díky inertnímu zpoždění, které je <1ns a zpoždění na hradle, které je 2ns, zpracovává hradlo v čase 0 logickou hodnotu 1 (vrací hodnotu 1), ale než ji díky době zpoždění vrátí, zpracovává

logickou hodnotu 0, díky které v dalším čase vrací nedefinovanou hodnotu (obě tyto hodnoty si zpětnou vazbou plánuje na další časy a hradlo je díky tomu neustále plánováno a neuchová si potřebnou paměť logické hodnoty 1)

Parametry simulace č.2:

Generátor A: typ – impulzní, délka impulsu – 2, posunutí v čase – 0
Hradlo: typ – OR, zpoždění – 2 ns

Očekávané chování:
- byla provedena úprava délky inicializačního impulsu na délku zpoždění hradla, hradlo by si mělo po svém zpoždění vracet pouze logickou hodnotu 1 a protože nedochází ke změně na jeho výstupu, nemělo by se po čase 2 dále plánovat

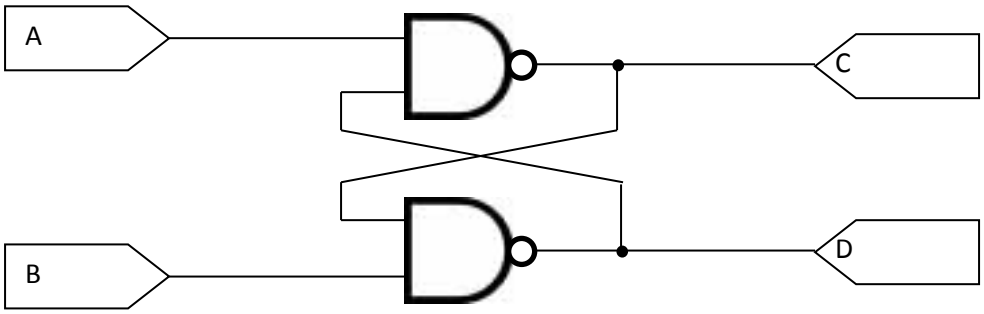
Získané výsledky:

A	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20 [ns]

Zhodnocení výsledků:
- bylo potvrzeno, že pokud je inicializační impuls roven délce zpoždění hradla, dojde zpětnou vazbou k zapamatování hodnoty a hradlo OR se chová jako paměťový prvek
- experimentem bylo ověřeno funkční zpoždění na hradle a zároveň zpětná vazba

5.2.2 Experiment klopného obvodu

Obvod:



Parametry simulace č.1:

Generátor A: typ – impulzní, délka impulsu – 19, posunutí v čase – 1
Generátor B: typ – impulzní, délka impulsu – 20, posunutí v čase – 0
Hradlo: typ – NAND, zpoždění – 3 ns
Hradlo: typ – NAND, zpoždění – 3 ns

Očekávané chování:

- hradlo NAND napojené na generátor A by měl konvertovat logickou hodnotu 0 na log. hodnotu 1 a tu předávat druhému hradlu, které zároveň dostává od generátoru B log. hodnotu 1 -> tedy, mělo by vracet log. hodnotu 0

Získané výsledky:

A	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C	2	2	2	1	2	2	2	2	2	1	2	2	2	2	2	1	2	2	2	2	2
D	2	2	2	2	2	2	0	2	2	2	2	2	0	2	2	2	2	2	0	2	2
simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20 [ns]

Zhodnocení výsledků:
- kvůli zpoždění logického prvku NAND napojeného na generátor A není možné pro pro druhý NAND předat požadované hodnoty

Parametry simulace č.2:

Generátor A: typ – impulzní, délka impulsu – 17, posunutí v čase – 3
Generátor B: typ – impulzní, délka impulsu – 20, posunutí v čase – 0
Hradlo: typ – NAND, zpoždění – 3 ns
Hradlo: typ – NAND, zpoždění – 3 ns

Očekávané chování:
- byla upravena délka impulsu generátoru A tak, aby inicializoval jeho napojené hradlo, které by nemělo vracet nedefinovanou hodnotu

Získané výsledky:

A	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C	2	2	2	1	1	1	2	2	2	1	1	1	2	2	2	1	1	1	2	2	2
D	2	2	2	2	2	2	0	0	0	2	2	2	0	0	0	2	2	2	0	0	0
simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20 [ns]

Zhodnocení výsledků:
- hradlo NAND napojené na generátor A sice svoje zpoždění zpracoval v pořádku, ale protože je seriově napojeno na druhý NAND se svým vlastním zpožděním a je závislý na jeho výstupu, klopný obvod nemůže opět fungovat

Parametry simulace č.3:

Generátor A: typ – impulzní, délka impulsu – 14, posunutí v čase – 6

Generátor B: typ – impulzní, délka impulsu – 20, posunutí v čase – 0
Hradlo: typ – NAND, zpoždění – 3 ns
Hradlo: typ – NAND, zpoždění – 3 ns

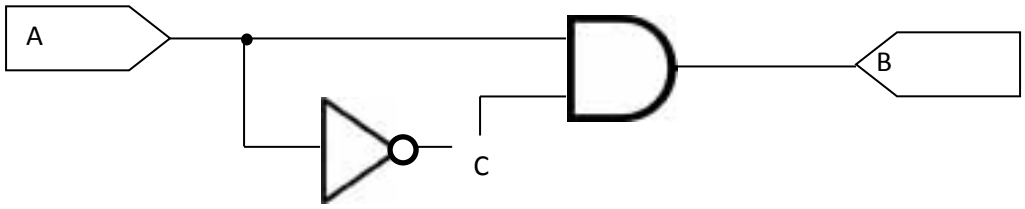
Očekávané chování:
- inicializační signál přivedený z generátoru A na NAND by měl být dostačující, aby hradlo udržel ve správném logickém stavu, dokud nedostane potřebnou logickou hodnotu od druhého hradla NAND

Získané výsledky:

A	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	[ns]

Zhodnocení výsledků:
- klopný obvod funguje v pořádku, od času 7 dostává hradlo NAND od generátoru logickou hodnotu 1, ale to nevadí, protože ve stejném čase mu vrací druhé hradlo potřebnou logickou hodnotu, aby se jeho výstupní hodnota nezměnila
- experimentem bylo ověřeno zpoždění hradel, zároveň také plánování hradla hradlem a také prodleva při zpracování, která je rovna součtu zpoždění všech hradel, které jsou na sobě závislé

5.2.3 Experiment prokazující statický hazard



Generátor A: typ – periodický, délka impulsu – 1, posunutí v čase – 0
Hradlo: typ – NOT, zpoždění – 2 ns
Hradlo: typ – AND, zpoždění – 2 ns

Očekávané chování:
- protože náš simulátor bere v potaz zpoždění (hradla nejsou dokonalá – jinak by výstupní hodnota hradla AND musela být stále 0 kvůli konverzi hodnoty hradlem NOT), očekáváme výskyt statického hazardu

Získané výsledky:

A	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
B	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	2	2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	[ns]

Zhodnocení výsledků:

- v tomto příkladu bylo nevhodně zvoleno časování generátoru nebo zpoždění hradla, nemůžeme pozorovat výskyt statického hazardu

Parametry simulace č.2:

Generátor A: typ – periodický, délka impulsu – 2, posunutí v čase – 0
Hradlo: typ – NOT, zpoždění – 2 ns
Hradlo: typ – AND, zpoždění – 2 ns

Očekávané chování:

- byla provedena úprava délky impulsu generátoru tak, aby se stejné hodnoty od generátoru a hradla NOT potkaly ve stejný čas na hradle AND, měli bysme pozorovat výskyt statických hazardů

Získané výsledky:

A	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
B	2	2	2	2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
C	2	2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	[ns]

Zhodnocení výsledků:

- díky vhodnému nastavení můžeme pozorovat statické hazardy (ve výsledcích zobrazeny červeně)

Parametry simulace č.3:

Generátor A: typ – periodický, délka impulsu – 4, posunutí v čase – 0
Hradlo: typ – NOT, zpoždění – 2 ns
Hradlo: typ – AND, zpoždění – 1 ns

Očekávané chování:

- opět provádíme úpravy parametrů simulace, abysme ověřili, zda se úprava zpoždění na hradle NOT na poloviční hodnotu promítne do výstupů simulace

Získané výsledky:

A	1	1	1	1	0	0	0	0	**1**	1	1	1	0	0	0	0	**1**	1	1	1	0
B	2	2	2	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	**1**	0	0
C	2	0	0	0	0	1	1	1	**1**	0	0	0	0	1	1	1	**1**	0	0	0	0

simT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	[ns]

Zhodnocení výsledků:

- opět můžeme pozorovat výskyt statických hazardů avšak v jiných intervalech
- dále bylo experimentem ověřeno, že je ekvivaletní natažení délky signálu nebo zkrácení doby zpoždění hradel

5.3. Závěry experimentů

Experimentování probíhalo na několika desítkách obvodů a pro každý obvod několik samostatných simulací (s různými parametry). Z experimentů lze odvodit chování simulátoru číslicových obvodů s dostatečnou věrohodností a experimentální prověřování dalších situací již napřinese další výsledky, protože všechny složitější obvody lze dekompozicí rozložit na jednotlivé dílčí jednoduché obvody, které byly simulovány a chyby odladěny tak, aby výsledky odpovídaly našemu abstraktnímu modelu.

6. Shrnutí simulačních experimentů a závěr

V rámci projektu vznikl efektivní a přenositelný simulační nástroj, který dokáže simulovat číslicové obvody popsané vstupním souborem s definici prvků obvodu a jejich vzájemného propojení. Validita modelu byla ověřena pomocí experimentů se sekvenčními a kombinačními obvody.

Použitá literatura

- [Pinker] – Pinker Jiří, Koupa Martin – Číslicové systémy a jazyk VHDL, BEN, 2006
- [Kolouch] – Skripta předmětu Impulzová a číslicová technika – VUT FEKT
- [Zikmund] – Zikmund Vít – Grafické znázornění hazardů, ČVUT FEL, 2007
- [Java Muni] – Přednášky předmětu Programování v jazyce Java – FI MUNI
- [IPP] – Skripta předmětu Principy programovacích jazyků a OOP – VUT FIT
- [IMS] – Materiály k přednáškám předmětu Modelování a simulace – VUT FIT
- [INC] – Skripta předmětu Návrh číslicových systémů – VUT FIT

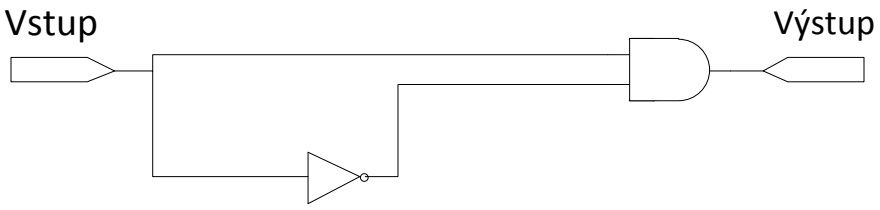
Metriky projektu

3191 řádků kódu

Příloha 1 – Syntaxe vstupního souboru

Vstupem simulačního modelu (dále již jen jako programu) je soubor formát XML popisující prvky a jejich vzájemné propojení. Níže je pomocí ukázky popsána jeho syntaxe.

Dříve uváděný obvod, který ilustroval statický hazard v obvodu bychom popsali následovně.



Obr. XX – Obvod se statickým hazardem

```
<?XML version="1.0" ?>
<BLOCK name="Hazard">
  <GENERATOR id="1" type="PERIODICAL" lambda="5" time="0">
    <OUTPUT id="1" port="1">VYSTUP_NA_AND</OUTPUT>
    <OUTPUT id="2" port="1">VYSTUP_NA_NOT</OUTPUT>
  </GENERATOR>
  <GATE id="1" type="AND" inputs="2" delay="2">
  </GATE>
  <GATE id="2" type="NOT" inputs="1" delay="2">
    <OUTPUT id="1" port="2">VYSTUP_NA_AND</OUTPUT>
  </GATE>
</BLOCK>
```

Vstupní soubor s popisem obvodu má *XML* hlavičku dle její specifikace. Popis obvodu je uzavřen do párových značek *BLOCK*, které ve svém atributu specifikují název obvodu. Uvnitř této značky je možné popsat generátory signálu pomocí značky *GENERATOR* a hradla pomocí značky *GATE*. Vzájemné propojení se pak popisuje pomocí značky *OUTPUT*, kde lze uvést i název výstupu, který však není simulátorem zpracován.

Rodičovská značka **BLOCK** je povinná a je nutné uvést parametr *name,* který pojmenovává sestavovaný obvod. Uzavření probíhá pomocí značky *</BLOCK>*.

Každý prvek v obvodu má pomocí atributu *id* určen unikátní identifikátor (unikátní pro generátory a zvlášť pro hradla), pomocí kterého se dále prvky a jejich spojení na sebe odkazují. Atribut *type* se liší dle popisovaného prvku. Pro prvek reprezentující generátor signálu to jsou

- *IMPULSE* – popis impulsního generátoru signálu
- *PERIODICAL* – popis periodického generátoru signálu

a pro hradla jsou to

- *AND*
- *OR*
- *NOT*
- *NOR*
- *NAND*

Dalšími povinnými parametry generátoru jsou atributy *lambda* a *time*. *Lambda* specifikuje po jak dlouhou dobu se drží signál produkovaný generátorem v hodnotě log. 1 a *time* čas, o který je signál posunutý. Ukázky výstupu generátorů jsou uvedeny v **příloze 2**.

U hradel (*GATE*) jsou pak dalšími povinnými parametry jejich zpoždění a počet vstupů. Zpoždění se popisuje pomocí atributu *delay* a počet vstupů hradla pomocí atributu *inputs*. Do žádného atributu nelze zadat hodnotu *0*, což odpovídá realitě. Pro generování signálu se používají výše popsané prvky *GENERATOR*.

Výstupy příslušejí daným prvků, a proto se zadávají mezi jejich párové značky. Je pak nutno specifikovat, na který prvek je výstup navázán. To je popsáno pomocí atributů *id* a *port*, kde *id* je identifikátor prvku na výstupu a *port* je číslo jeho vstupu.

Tímto souborem je možné popsat jakkoliv složitý logický obvod s různými typy vstupních signálů a hradel. Je také možné každému hradlu zadat jeho zpoždění, což modeluje případ, kdy jsou použity kompatibilní hradla různých technologií.

Příloha 2 – Třístavová logika

Níže jsou uvedeny možné kombinace hodnot na hradlech, které zahrnují nedefinovanou hodnotu.

AND		
A	B	X
U	U	**U**
0	U	**0**
U	0	**0**
1	U	**U**
U	1	**U**
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Graf 1 – Logické hradlo AND

OR		
A	B	X
U	U	**U**
0	U	**U**
U	0	**U**
1	U	**1**
U	1	**1**
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Graf 2 – Logické hradlo OR

NAND		
A	B	X
U	U	**U**
0	U	**1**
U	0	**1**
1	U	**U**
U	1	**U**
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Graf 3 – Logické hradlo NAND

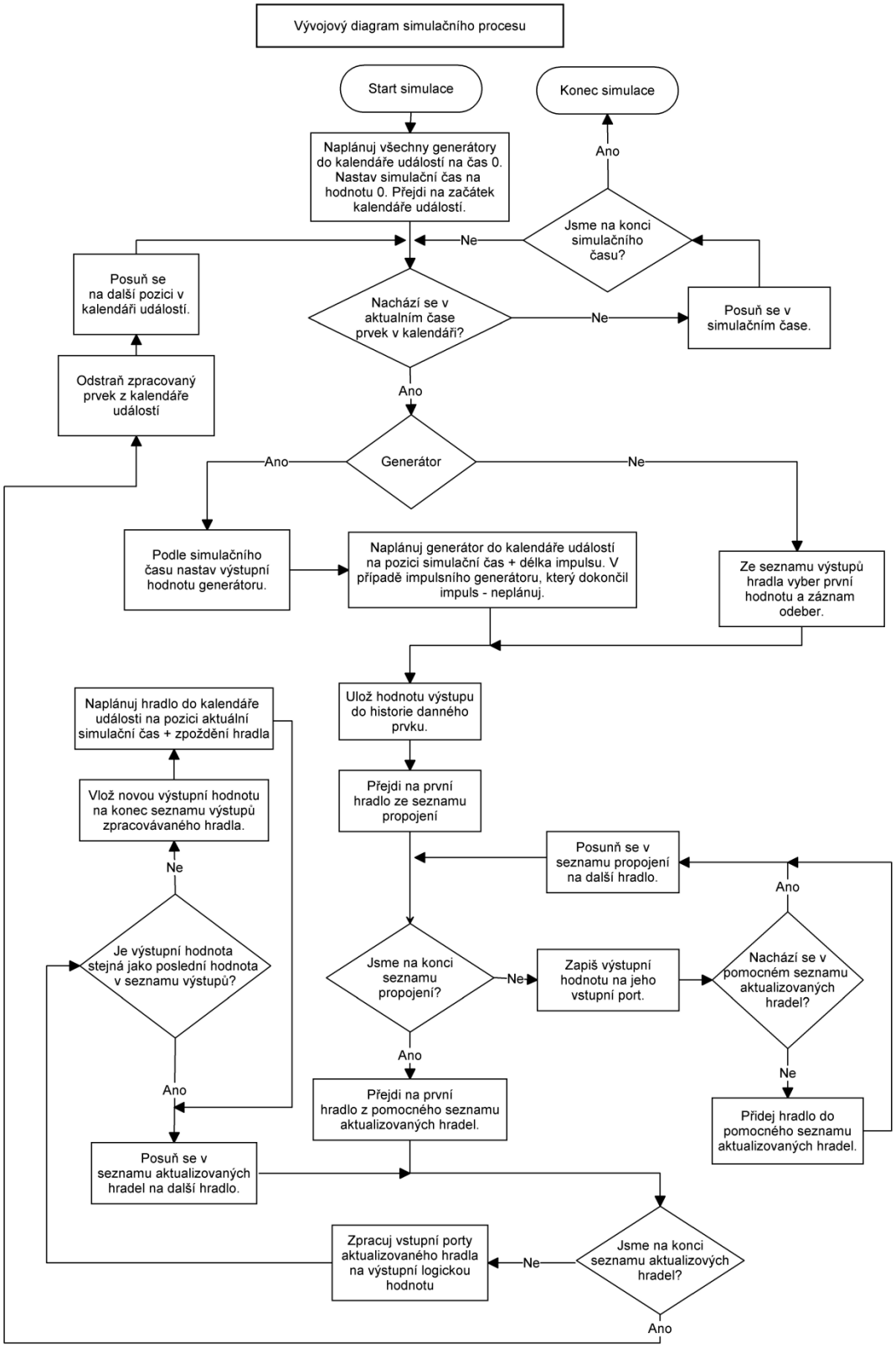
NOR		
A	B	X
U	U	**U**
0	U	**U**
U	0	**U**
1	U	**0**
U	1	**0**
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Graf 4 – Logické hradlo NOR

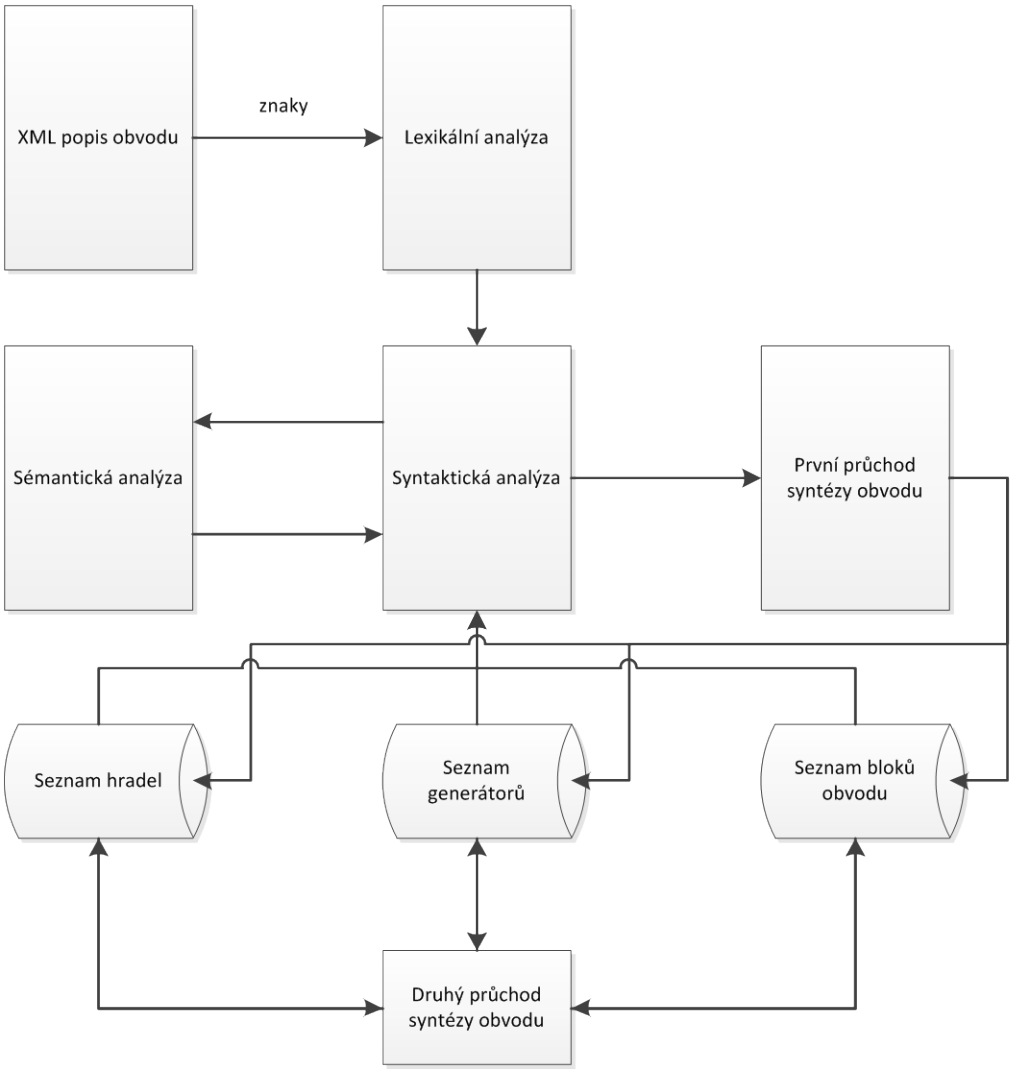
NOT	
A	X
U	**U**
0	**1**
1	**0**

Graf 5 – Logické hradlo NOT

Příloha 3 – Konceptuální diagram simulátoru



Příloha 4 – Diagram analyzátoru vstupního souboru



Příloha 5 – Generátory signálu

Generátory

Aby mohly být simulovány opravdu všechny situace, je nutné vytvořit generátory signálu, které inicializují hradla. V našem modelu jsme proto vytvořili prvek generátor, který může nabývat dvou typů:

- impulsní (odešle signál a dále se neplánuje)
- periodický (po periodě mění svoji výstupní logickou hodnotu)

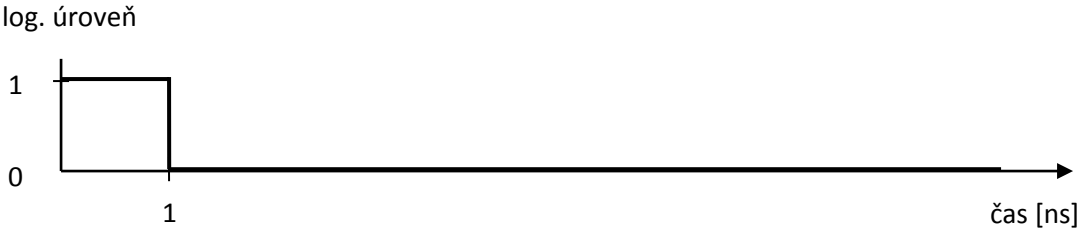
Každému typu generátoru lze nastavit několik povinných parametrů. Díky těmto parametrům lze v programu odsimulovat jakoukoliv reálnou situaci, protože možné nastavení generátoru pokrývá celou množinu jeho chování. Povinné parametry:

- čas, od kterého generátor začíná odesílat signály
- délka signálu

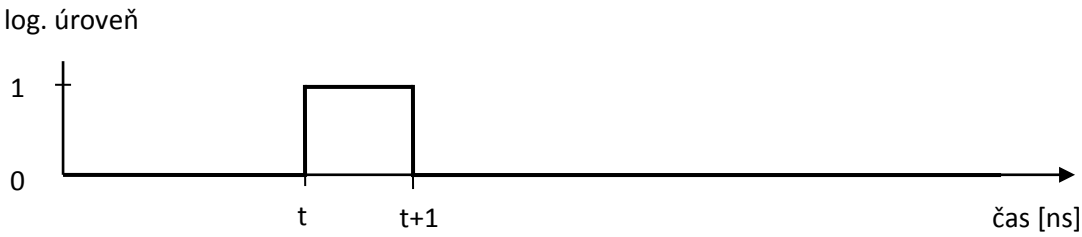
Ukázky možnosti nastavení generátoru

Impulsní generátor

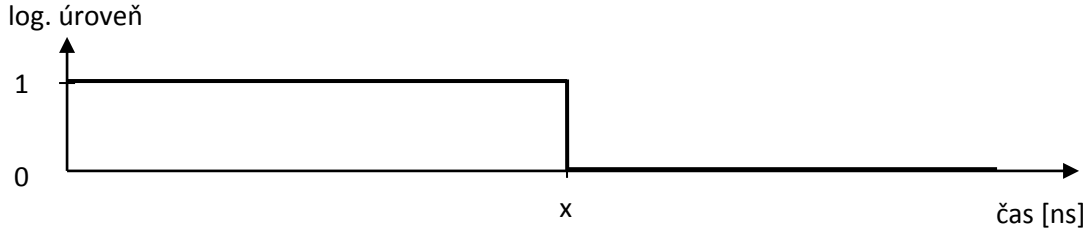
- jednotkový impuls v čase 0



- jednotkový impuls v čase *t*



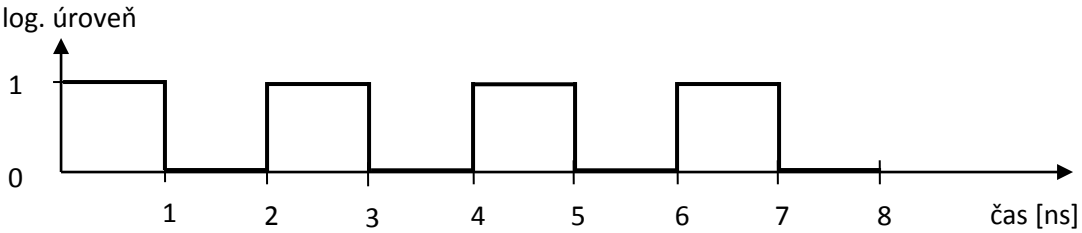
- impuls o délce *x* v čase 0



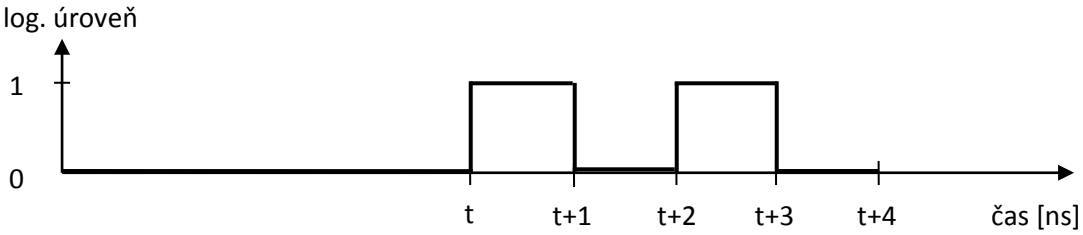
- impuls o délce *x* impuls v čase *t*

Periodický generátor

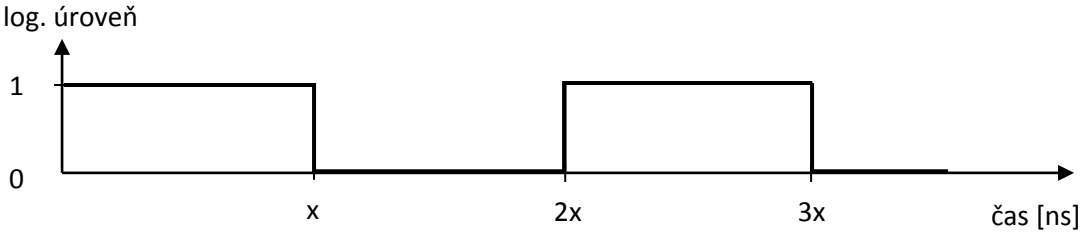
- jednotkový periodický impuls v čase 0



- jednotkový periodický impuls v čase *t*



- periodický impuls o délce *x* v čase 0



- periodický impuls o délce *x* v čase *t*