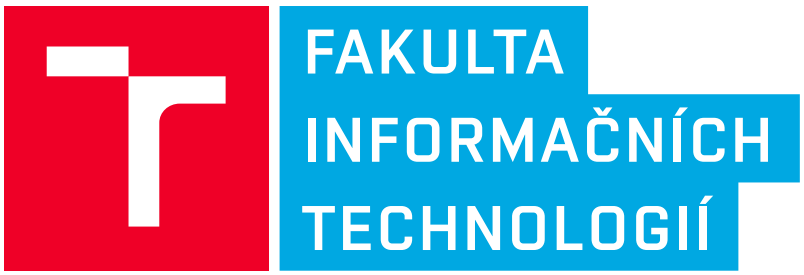


Modelování a simulace 2015/16

Téma č. 3 Simulátor číslicových obvodů

8. prosince 2015

Autoři: Jiří Komárek (xkomar27@stud.fit.vutbr.cz)
Lukáš Pelánek (xpelan03@stud.fit.vutbr.cz)

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

1 Úvod

Tato práce vznikla jako projekt do předmětu Modelování a simulace (IMS) vyučovaným na Fakultě informačních technologií Vysokého učení technického v Brně. Popisuje proces implementace a návrhu simulátoru číslicových obvodů složeného z hradel AND, OR, NAND, NOR a NOT v programovacím jazyce C++. Smyslem projektu je demonstrovat ve grafické formě průběh vnitřních stavových veličin vnitřních prvků v kombinačních a sekvenčních obvodech.

1.1 Autoři a zdroje

Na projektu spolupracovali studenti FIT VUT Jiří Komárek a Lukáš Pelánek. Nepřímo byli do řešení svými přednáškami a cvičeními zapojeni Ing. Martin Hrubý, Ph.D. a Dr. Ing. Petr Peringer. Autoři čerpali informace z výše uvedených přednášek a studijních materiálu předmětu IMS [4] [5], z absolvovaného kurzu Návrh číslicových systémů (INC) [3], z referenční příručky jazyka C++ [1] a v rámci inspirace také ze simulační knihovny SIMLIB. [2]

1.2 Prostředí pro ověřování činnosti simulátoru

Výsledný program byl testován na školním serveru merlin.fit.vutbr.cz (CentOS 5.5 64bit Linux). Pro porovnání výsledků stavových hodnot na hradlech po provedení experimentu byly výsledné hodnoty z grafu porovnány s pravdivostními tabulkami jednotlivých hradel.

2 Rozbor tématu a použitých metod/technologiií

V rámci projektu jsme se zabývali zkoumáním a sledováním vnitřních stavů logických prvků v číslicových obvodech (hradel). Logická hradla jsou elektronické součástky, které mají většinou 2-n vstupů (existují vyjímky - např. negace NOT) a jeden výstup. Vstupní i výstupní hodnotou je logická jednička nebo nula. Chování logických hradel je možno popsat v boolově algebře. V kombinačních nebo sekvenčních obvodech se stavy vnitřních hradel mění (případně zůstávají stejné) podle pravdivostní tabulky pro každé hradlo. [6] Tyto změny stavových veličin sledujeme a vykreslujeme do grafu.

2.1 Použité postupy pro vytvoření modelu

Pro implementaci byl zvolen a použit programovací jazyk C++ s použitím objektově orientovaného přístupu. Objektově orientované programování nám oproti klasickému stylu velmi pomůže s abstrakcí celého systému simulátoru. Pro primární grafický výstup simulátoru, byl zvolen program GNUplot, kvůli jednoduchému překreslení hodnot do grafu.

2.2 Původ použitých technologií

Při implementaci a návrhu kalendáře jsme se inspirovali studijní oporou IMS(od slajdu č. 32 - kapitola 5, odkaz v Kapitole 1.1) a také simulační knihovnou SIMLIB. Pro návrh správného chování hradel jsme využili znalostí získaných z kurzu INC (odkaz v Kapitole 1.1).

3 Koncepce - implementační téma

Cílem projektu bylo implementovat simulátor číslicových obvodů. Celá koncepce výsledného simulátoru je tedy založená na realném principu používání logických hradel v kombinačních a sekvenčních logických obvodech. Program je složen z jednotlivých komponent, které mezi sebou komunikují. Jednotlivé komponenty se vytvoří na základě vstupního souboru, který je ve formátu zjednodušeného NETLISTU. Následně se jednotlivé komponenty propojí a spustí se hlavní programová smyčka. Vše je řízeno pomocí kalendáře, který uchovává aktivační záznamy procesu. Pokud se změní výstupní hodnota nějaké komponenty, naplánují se příslušné procesy připojených komponent a vloží se do kalendáře. Jakmile nastane aktivační čas procesu, proces se aktivuje a vyjme z kalendáře. Simulace končí po uplynutí definovaného modelového času.(IMS opora - Kapitola 5.5.3 - odkaz v Kapitole 1.1)

4 Architektura simulátoru

Následující kapitola popisuje architekturu celého simulátoru a vychází z Kapitoly 3.

4.1 Rozdělení tříd

Dráty
Jednotlivé komponenty jsou propojeny pomocí drátů. Drát představuje třída `Wire`. Každý drát uchovává svoji logickou hodnotu, která se na něm právě nachází. Dále obsahuje index, který jednoznačně identifikuje drát. Tento index je uložen ve vstupní topologii a na základě tohoto indexu jsou jednotlivé komponenty propojeny.
Drát dále obsahuje vector, ve kterém jsou uloženy ukazatele na jednotlivá hradla, která jsou na výstup tohoto drátu připojena a hodnoty typu bool, které značí, zda je vstup či výstup drátu připojen k nějaké komponentě. Všechny atributy jsou typu protected a pro práci s nimi jsou naimplementovány příslušné veřejné metody.
Třída `Wire` dále obsahuje statickou metodu `AddWire`, která vytvoří nový drát a vloží jej do vectoru, který obsahuje ukazatele na dráty.

Procesy
Všechny procesy dědí z abstraktní třídy `Process`, která obsahuje jednu privátní proměnnou `time`, která určuje čas, kdy se má proces aktivovat. Dále obsahuje veřejnou, čistě virtuální metodu `Action`, kterou musí zděděná třída naimplementovat a definuje chování daného procesu.
Z této třídy dále dědí třídy `GateProcess`, `InputProcess` a `ClkProcess`. Vyjmenované třídy obsahují ukazatel na komponentu. Na dané komponentě reprezentují tyto třídy její chování a implementují metodu `Action`, kde se zavolá instanční metoda dané komponenty a provede výpočet nové hodnoty na výstupu komponenty.

Kalendář
Simulace je řízená pomocí kalendáře. Kalendář reprezentuje třída `Calendar`, která je implementována navrhovým vzorem singleton. Třída obsahuje privátní statický ukazatel `instance` na třídu `Calendar`, který bude ukazovat na jedinou instanci této třídy, protože konstruktor je rovněž privátní. Dále obsahuje vector `processes`, který obsahuje ukazetele na procesy.
Třída dále obsahuje veřejné statické metody `getInstance` a `PlanAction`. První zmíněná metoda slouží k získání instance této třídy. Druhá metoda vloží proces do vectoru `processes`. Mezi důležité veřejné metody patří `getFirst`, jenž vrátí ukazatel na proces, který se má vykonat jako první, dále `runProcess` jenž spustí daný proces a `Destroy`, která celý kalendář zruší.

Komponenty
Základem všech komponent je abstraktní třída `Gate`. Obsahuje svoji aktuální hodnotu, zpoždění komponenty a čistě virtuální metodu `connectWire`, která bude sloužit k připojení drátu na vstup či výstup. Uvedené části jsou typu oprávnění protected. Dále obsahuje čistě virtuální metodu `computeOutput`, která bude obsahovat algoritmus pro výpočet výstupní hodnoty komponenty a `getOutputWire`, která vrátí ukazatel na výstupní drát.
Z této třídy dále dědí abstraktní třídy `Generator`, `OneInGate`, `TwoInGate`, jenž tvoří základ jednotlivých komponent. Tyto třídy jsou si velmi podobné, nicméně se liší v privátních proměnných a implementaci virtuálních metod. Třída `Generator` obsahuje ukazazel `output` na výstupní drát a výstupní index `outputIndex`, který je specifikován ve vstupní topologii. Třída `OneInGate` obsahuje dva ukazatele a dva indexy. Pro vstup a výstup. Podobně tak třída `TwoInGate` jenž obsahuje tři ukazatele a tři indexy. Dva vstupní a jeden výstupní. Každá třída implementuje metodu `connectWire`, která porovnává jednotlivé indexy a případně připojí drát na vstup či výstup.
Dále třídy `Input` a `Clk` dědí z třídy `Generator`. Z třídy `OneInGate` dědí třída `NotGate` a ze třídy `TwoInGate` dědí třídy `AndGate`, `OrGate`, `NandGate` a `NorGate`. Tyto třídy reprezentují jednotlivé komponenty a každá třída implementuje metodu `computeOutput`, která byla deklarována jako čistě virtuální ve třídě `Gate`.

5 Podstata simulačních experimentů a jejich průběh

Hlavní důvod provádění simulačních experimentů je ověřit správnost vymyšlené koncepce a následně spravnou funkčnost naimplementovaného simulátoru. U našeho zadání tvoří experimentování velmi důležitou část projektu. Bez prováděných experimentů na vytvořených topologiích z logických členů by nebylo možné odladit možné chyby a celý simulátor by mohl fungovat chybně. Experimentování jsme provedli na námi graficky vytvořených topologiích. Pro účely dokumentace jsme se rozhodli zdokumentovat dvě relativně jednoduché topologie.

5.1 Postup experimentování

Postup experimentování spočíval v modifikování vytvořených topologií skrz vstupní soubor. Po načtení topologie se na vstup požadované komponenty přiřadí zadaná hodnota nebo hodinový signál (podle vstupního souboru) a dále prochází topologií. Každá změna výstupu hradla se s jednotným zpožděním projeví změnou stavu (případně setrváním stavu) hradla v grafickém diagramu. Pokud hodnoty v grafu odpovídají správným hodnotám z pravdivostních tabulek pro hradla, dá se daný experiment považovat za úspěšný. Pokud tento

stav nastane u několika různých topologií, můžeme říci, že se simulátor chová podle reálných předpokladů. Přeložení a spuštění programu, které bylo používáno při experimentech, je popsáno v souboru `README`, který je přiložen v adresáři u projektu. Jednotné zpoždění každého typu hradla je 10 ns.

5.2 Dokumentace experimentů

Vstupní topologie každého zkoušeného experimentu je zadána formou zjednodušeného NETLISTU. Pomoci čísel pinů se vnitřní komponenty topologie propojují dohromady. Každá komponenta má různý počet pinů, v seznamu jsou piny vždy postupně. Pokud tedy komponenta již další pin neobsahuje zapíše se 0. Začíná se vždy vstupnímy piny.

Ukázková topologie:
CLK 1 0 0
CLK 2 0 0
NOT 1 3 0
NAND 3 5 4
NAND 2 4 5

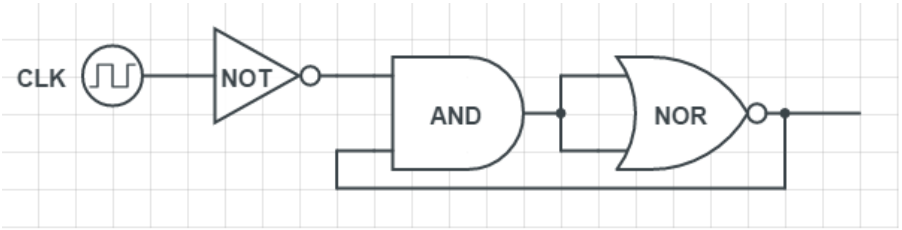
Dostupné komponenty použité při experimentech: IN, CLK, NOT, AND, OR, NAND, NOR
Komponenty IN a CLK obsahují jeden vystupní pin.
Komponenta NOT obsahuje pouze jeden vstupní a jeden vystupní pin.
Komponenty AND, OR, NAND a NOR obsahují dva vstupní a jeden vystupní pin

Experiment 1

Vstupní topologie ve formě NETLISTU ve zdrojovém souboru:

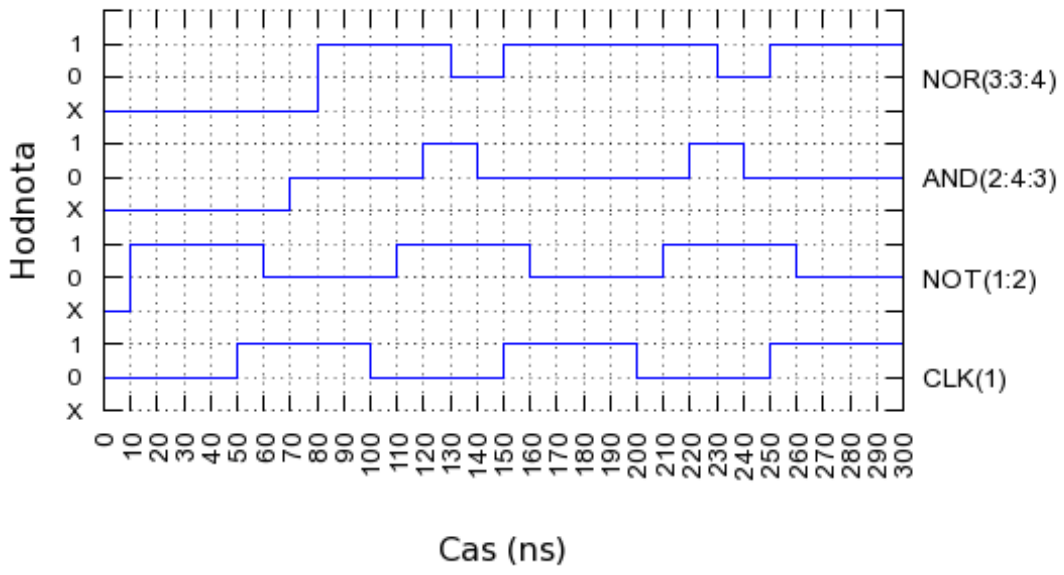
CLK 1 0 0
NOT 1 2 0
AND 2 4 3
NOR 3 3 4

Grafické schéma vstupní topologie:



Obrázek 1: Schéma vytvořeného sekvenčního obvodu

Grafický výstup simulátoru pro sekvenční obvod:



Obrázek 2: Graf simulace sekvenčního obvodu

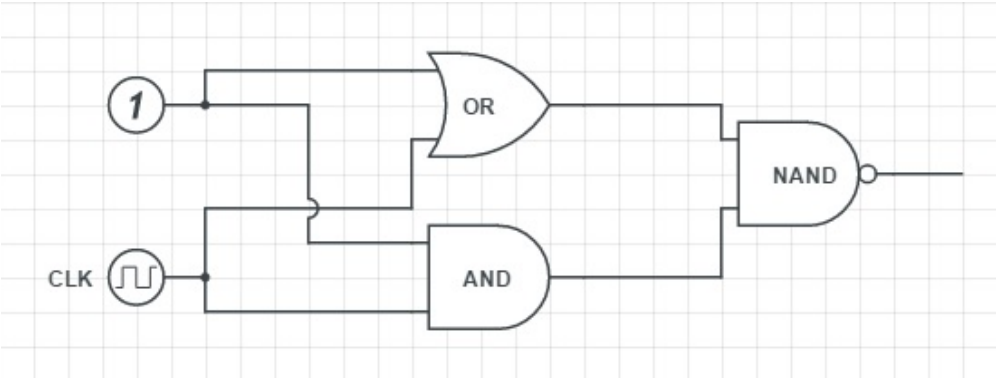
Popis experimentu se sekvenčním obvodem:
V tomto experimentu jsme se rozhodli odsimulovat zapojení obvodu se zpětnou vazbou na schématu 1. Výsledkem simulace vznikl graf 2. Dále jsme podle pravdivostních tabulek jednotlivých hradel kontrolovali stavy výstupů každého logického členu obvodu. Když se zaměříme např. na hradlo NOT, můžeme sledovat že invertuje svůj stav výstupu pokaždé, když se změní stav jeho vstupu v podobě CLK (se zpožděním 10 ns), což odpovídá reálnému chování invertoru. AND změnil svoji hodnotu z nedefinovaného stavu v momentě, kdy na výstupu hradla NOT byla log. 0, což odpovídá pravdivostní tabulce hradla AND, kde log. 0 na jednom ze vstupů znamená pokaždé výstup 0. V momentě kdy se na výstupu AND objevila log. 0 se po zpoždění 10 ns objevila log. 1 na hradlu NOR. NOR měl v čase před změnou stavu na svém vstupu dvě log. 0, což podle pravdivostní tabulky znamená přepnutí do log 1, což se také stalo. Dále se log. 1 z hradla NOR objevila na vstupu hradla AND a v momentě kdy se na výstup hradla NOT dostala log. 1 se výstup hradla AND změnil na log. 1, což znovu odpovídá. Obdobně jsme postupovali u všech prvků topologie a zjistili, že simulace probíha v pořádku.

Experiment 2

Vstupní topologie ve formě zjednodušeného NETLISTU ve zdrojovém souboru:

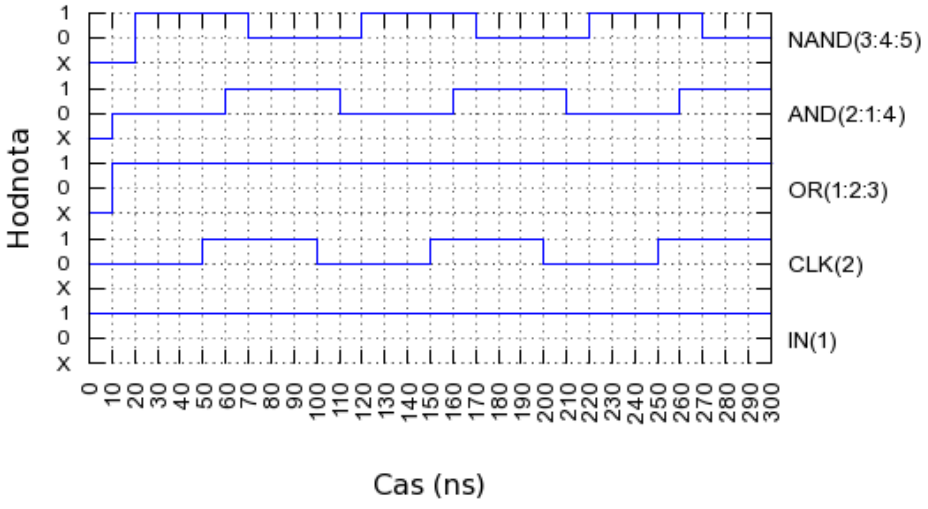
IN 1 0 0
CLK 2 0 0
OR 1 2 3
AND 2 1 4
NAND 3 4 5

Grafické schéma vstupní topologie:



Obrázek 3: Schéma vytvořeného kombinačního obvodu

Grafický výstup simulátoru pro kombinační obvod:



Obrázek 4: Graf simulace kombinačního obvodu

Popis experimentu s kombinačním obvodem:
Ve druhém experimentu jsme zkusili odsimulovat kombinační obvod ze schématu 3. Po odsimulování vznikl graf z obrázku 4. U tohoto experimentu jsme postupovali naprosto stejně jako u předchozího. Opět se potvrdilo, že se hradla chovají podle jejich pravdivostních tabulek a simulace odpovídá realným předpokladům.

5.3 Závěry experimentů

V rámci experimentování s naimplementovaným simulátorem byla provedena serie experimentů různých číslicových obvodů. Při experimentech jsme postupovali od jednoduchých topologií (samotná hradla) až po více komplexnější zapojení. Z experimentů jsme zjistili, že simulátor funguje korektně a další experimenty už by nám nepřinesly nové informace.

6 Shrnutí simulačních experimentů a závěr

V rámci projektu byl navrhnut a naimplementován simulátor logických obvodů složený z hradel. Při návrhu se bylo třeba seznámit s reálnou funkcí logických hradel v číslicových obvodech po logické i elektronické stránce tak, aby výsledný simulátor co nejvíce odpovídal realitě. Funkčnost simulátoru byla ověřena na serii testových topologií, ve kterých byly zastoupeny jak kombinační, tak i sekvenční číslicové obvody.

Reference

[1] *Referenční příručka C++*. [cit. 2015-12-07]. Dostupné na: < <www.cplusplus.com/reference>>.

[2] *Simulační knihovna SIMLIB*. [cit. 2015-12-07]. Dostupné na: < <http://www.fit.vutbr.cz/~peringer/SIMLIB>>.

[3] Fučík, O. *Návrh číslicových systémů*. 2015-02-18 [cit. 2015-12-07]. Dostupné na: < <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/INC-IT/lectures/logicke_obvody.pdf?cid=9371>>.

[4] Peringer, P. *Modelování a simulace - Studijní opora*. 2012-12-17 [cit. 2015-12-07]. Dostupné na: < <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IMS-IT/texts/opora-ims.pdf?cid=10316>>.

[5] Peringer, P. *Modelování a simulace*. 2015-09-17 [cit. 2015-12-07]. Dostupné na: < <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IMS-IT/lectures/IMS.pdf?cid=10316>>.

[6] Sangosanyar, W. *Basic Gates and Functions*. 1997[cit. 2015-12-07]. Dostupné na: < <http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/index.html>>.