Logic Gates
C++ zápočtový

Tomáš Karella

4. května 2017

Téma:

Tento program slouží k tvorbě hradlových sítí, následně k simulaci jejich výpočtu a jejich opakovanému využití v dalších hradlových sítích. Propojení sítě je koncipováno přes načítání kostrukčního souboru, který je popsát dále.

Kompilace a spouštění:

Pro spuštění na Linux distribucích je nutný překladač g++-6 a GNU Make. Na github branch: linuxB

- make compile - zkompiluje zdrojáky do spustitelného souboru "/bin/main"
- make clean - smaže zdrojové zkompilované soubory
- make convert - změní eol všech příkladů z CRLF na LF
- make test1 - make test6 - zkonstruuje příklad a vyzkouší všechny možné vstupy
    1. XORAND ADDW
    2. XORAND ADDW ADD4
    3. allgate
    4. DAND
    5. DXOR
    6. example.txt

Pro spuštění na Windows distribucích je program dostupný pouze jako Visual Studio projekt(VS 2015 a vyšší). Na github branch: master

Uživatelská dokumentace:

Interaktivní režim:

Po spuštění bez parametrů se otevře interaktivní režim, který vás vyzve k zadání cesty konstrukčního souboru. Po jeho úspěšném zkonstruování se přepne do režimu vkládání vstupu, kdy pro daný vstup spočítá a vrátí výstup. Dále poskytuje možnost zkonstruovat hradlo pro další použití(klíčové slovo: c). Po úspěšné konstrukci se opět dostane do režimu načítání souborů. Nyní už může používat jméno prvního konstruovaného hradla jako typ.

Formát vstupu:

Vstupy pro hradlovou síť jsou ve formátu řetězce znaků a to 1 pro logickou 1 a 0 pro logickou 0.

Interaktivní režim - klíčová slova:

exit,e - slouží k ukončení aplikace
const, c - konstruuje zadané hradlo, (jen v režimu, kdy je načtený konstrukční soubor)
h, help, man - zobrazí klíčová slova

Pasivní režim:

Pro spuštění pouze konstrukce a simulace jednotlivých hradel. Lze používat následující argumenty při spouštění main fce. Pro spuštění v pasivní režimu musí být nastaven alespoň jeden konstručkní soubor a alespoň jeden vstup.
-f [file..] - cesty ke konstručním souborům, které se konstrují dle pořadí
-h - vytiskne argumenty fce
-i - vstupy pro poslední generované hradlo, ve formát 1 - pro logickou 1 a 0 - pro logickou nulu, př: pro hradlo DAND: 10
-a - vyzkouší všechny možné vstupy pro poslední konstruované hradlo a -i ignoruje
-d - zoobrazí debug informace

Konstrukční soubor - formát:

Modelový soubor lze nalézt "examples/model.txt". Ve zmíněné složce je i celá řada příkladů k vyzkoušení programu.
Soubor se skládá ze dvou hlavních částí. Pojmenování hradel, kde deklarujete jméno hradla(noCASE sensitive a smí obsahovat pouze číslice a písmena) k jménu typ hradla. Část druhá, kde se řeší jejich vzájemné propojení. Jednotlivé tagy jsou odděleny tabulátorem.

#GATE MYNAME (1)
#INPUT sizeOfInput
#OUPUT sizeOfOutput
NameOfGate Type (2)
NameOfGate Type
#CONNECT (3)
NameOfGate[OutputPinID] -> NameOfGate[InputPinID] (4)
NameOfGate[OutputPinID] -> NameOfGate[InputPinID]
(5)

1. kontrolní tag pro pojmenovácí část souboru a jméno vašeho hradla (oddělené tabulátorem) na dalších řádcích tagy pro zadání velikosti vstupu & výstupu
2. jméno hradla(pouze písmena a číslice) dále typ(predefinovaný či už zkonstruovaný) vzájemně odděleny tabulátorem.
3. kontrolní tag pro začátek propojovací část souboru
4. jméno hradla a v hranatých závorkách číslo výstupního pinu dále "->"oddělená z obou stran tabulátorem jméno hradla s číslem vstupního pinu. Pro připojení vstupu je přednastaveno jméno I a pro připojení výstupu přednastaveno jméno O.
5. kontrolní tag konce konstr. souboru

Pro konstrukci musí být připojeny u hradel všechny vstupní i výstupní piny, konstrukt musí obsahovat alespoň jedno hradlo vstupní a alespoň jedno výstupní.

Kostrukční soubor - seznam předdefinovaných typů hradel

- Základní logické fce:
  - NOT
  - AND
  - OR
  - XOR
  - NAND
  - NOR
  - XNOR

- Ostatní
  - Input
  - Output
  - Blank (má pouze vstup a nikam není dále posílán)
  - ConstIn0 (má pouze výstup a stále nastaven na 1)
  - ConstIn1 (má pouze výstup a stále nastaven na 1)
  - Double (na dva výstupy pouští stejný 1 vstup)

Konstrukční soubor - příklady

- model.txt - obecný model konstručního souboru
- allgate.txt - hradlo využívající všechno předdefinované hradla
- DAND.txt - double and, and s 2 výstupy
- DXOR.txt - double xor, xor s 2 výstupy
- XORAND.txt - xor a and na stejný vstup, první výstup xor druhý and
- ADDW.txt - sčítačka dvou čísel a přetečení, musí být zkonstruovaný XORAND
- ADD4.txt - sčítačka 2x2 bitových čísel, musí být zkonstruovaný XORAND a ADDW

Implementace:

Implemetaci je rozdělena do následujích částí:

- **graph**(graph.h) - implementuje multigrafu, každý vrchol a hrana nese generickou hodnotu.
- **gates** (gates.h, gates.cpp) - deklaruje obecně třídu hradlo a její konkrétní implementace.
- **workbench** (workbench.h, workbench.cpp) - implementuje propojování jednotlivých hradel v hradlovou síť, kontroluje jejich korektnost a konstruuje uživatelsky definovaná hradla.
- **workbenchTUI** (workbenchTUI.cpp, workbench.TUI.h) - řeší komunikaci mezi uživatelem a workbench, načítá konstrukční soubory.
- **main** (main.cpp) - parsuje vstupní parametry a spouští metody workbenchTUI.

graph

Obsahuje následující šablonové třídy s typovými parametry VertexValue, EdgeValue:

- Vertex - drží hodnotu VertexValue
- Edge - orientovaná hrana mezi Vertex s hodnotou EdgeValue
- Graph - orientovaný multigraf nad Vertex a Edge (vtype = Vertex<VertexValue>, etype = Edge<VertexValue,EdgeValue>)
  - vtype * add_vertex(VertexValue value) => přidává vrchol do grafu bez hran
  - etype* connect(vtype* from, vtype* to, EdgeValue value) => vytváří hranu z from do to s hodnotou value
  - void disconnect(etype* e) => smaže v grafu hranu e
  - vector<etype*> edges_from(vtype* a) => vrátí seznam hran z vrcholu a
  - vector<etype *> edges_to(vtype* a) => vrátí seznam hran do vrcholu a
  - unordered_set<vtype *> verticies_from(vtype* a) => vrátí seznam vrcholů dosažitelných z a
  - bool cycle_detection() => testuje, zda daný graf obsahuje cyklus, implementováno DFS, které pokud se vrátí do už uzavřeného vrcholu nahlásí nalezení cyklu
  - bool all_vertices_available_from(const vector<vtype*>& from) => testuje, zda z daných vrcholů je dosažitelný celý graf, pomocí DFS projde graf. Pokud počet uzavřených vrcholů není shodný s velikostí grafu zahlásí false

gates

- výčtový typ Status - Zero, One, Floating(logická 1,0 a nenastaveno)
- třída Signal - obsahuje proměnné : toID, fromID - pořadí pinů u vstupního a výstupního hradla a Status aktuální nastavení signálu
- abstraktní třída Gate -
  - obsahuje vlastnosti velikost vstupu, výstupu, název, result - pravdivý, když jsou nastaveny výstupy, resultValues - hodnota výstupů
  - virtuální metoda Update - spočítá výstup z hradla
- Třídy všech předdefinovaných hradel s přetíženou metodou Update, která počítá jejich logické fce.
  - NOT AND OR XOR NAND NOR XNOR Input Output Blank ConstIn0 ConstIn1 Double
- třídu UserDefinedGate - třída uchovávající hradlovou síť sestavenou uživatelem, drží ukazatel na její graf, vstupní a výstupní hradla
  - metoda Update - nastaví vstupní hradla, simuluje průchod skrz graf hradlovou síť a vrátí hodnoty z výstupních hradel

workbench

- výčtový typ WorkbenchStatus značící stav workbench - UnderConstruction, Constructed, Calculating, Calculated
- třídu Workbech
  - obsahuje Graph<Gate*,Signal> - která je vlastní hradlovou sítí, ukazatele na modely uživatelsky definovaný hradel, na vstupní/výstupní/constatní hradla, ukazatele na ještě nepřipojená hradla(funkce pro jejich výpis), vlastní objekty typu Gate jsou uloženy v interním vektoru třídy.
  - ve stavu konstrukce - jsou k dispozici metody pro přidávání nových hradel, propojování.
    - void Add(const std::string& name, const std::string& typeName) => přidá na workbench nové hradlo s daným jménem "name"typu "typeName", kde lze použít předdefinovaná hradla nebo už hradla zkonstruovaná. Pokud už existuje vrchol daného jména(iname) nebo neexistuje jméno typu(utype) vyhodí vyjímku.
    - Connect(const std::string& fromName, std::size_t fromPin, const std::string& toName, std::size_t toPin) => přidá hranu mezi hradlem "fromName"z výstupu "fromPin"do hradla "toName"do vstupu "toPin". Jména hradel z fce Add. Pokud jména vrcholu neexistují vyhodí vyjímku(uname).
  - Pro konstrukci:
    - void ConstructBench() => Otestuje, zda je hradlová síť korektní. Obsahuje alespoň 1 vstupní a 1 výstupní hradlo, zda nevznikl cykl a zda je hradlo celé dosažitelné. K tomu používá metody graph. Pak se přepne workbench do stavu Constructed. Pokud testy neuspějí vyhodí vyjímku(fpin,npart,dcycle).
  - Při úspěšně zkonstruovaném hradle:
    - void SetInput(const vector<bool>& input) => nastaví vstupní hradla na hodnoty z argumentu, a simuluje průchod sítí v grafu a skončí až budou nastavené všechny výstupní hradla.
    - vector<bool> ReadOutput() => vrátí vypočítané výstupy, pokud byla předtím volána fce SetInput(input)
  - Konstrukce hradla:
    - void ConstructUserGate(const string& name, size_t newInputSize, size_t newOutputSize) =>Zkonstruje z aktuální sítě UserDefinedGateModel, přidá jej do seznamu všech UserDefinedGates, nastaví novému hradlu typ dle name. A zavolá ResetWorkbech s false. Připraví workbench pro další hradlo s danou velikostí vstupu a výstupu.
  - Další funkce:

    * ListOfType(),ListOfNamedVertex, GetTestOutput() => výpisové funkce všech aktuálních typů hradel a výpis z testů při konstrukci hradlové sítě
    * void ResetWorkbench(bool deleteUDG, size_t newInputSize, size_t newOutputSize) => smaže aktuální už zkonstruovanou část hradla, pokud deleteUDG smaže i uživatelské typy. A připraví workbench s danou velikostí vstupů a výstupů.
  - Seznam vyjímek:
    * fpin - Volný pin při konstrukci.
    * dcycle - V grafu detekován cyklus
    * npart - Nedostupná část hradla ze vstupu nebo konst. hradel.
    * istat - Workbench se nachází v jiném stavu než vyžaduje vyvolaná akce.
    * utype - neexistující typ
    * itype - typ s tímto jménem už existuje
    * iname - vrchol s tímto jménem už existuje
    * uname - neexistující jméno vrcholu
    * opin - už obsazený typ
    * isize - nevhodná velikost vstupu či výstupu
    * iinput - špatný formát vstupu

workbenchTUI

- Tvoří vrstvu mezi uživatelem a workbench. Funguje v několika verzích. Vrací výstup na streambuf output a čte streambuf input dané při konstrukci objektu.
  - void InteraktiveMode(),=> spustí interaktivní mód, který načte daný soubor, pak umožňuje konstrukci hradla, čtení vstupů a přidávání dalších hradel
  - void PassiveMode(const vector<string>& filePaths, const vector<vector<bool»& inputSet,bool tryAllInputs); => zkonstruuje daný konstrukční soubor, vypočítá výstup podle inputSettings a vypíše výstup.

main

Vytváří své workbenchTUI, nastaví input na std::cin a output std::cout, dále parsuje argumenty, dle jejich počtu volá příslušné metody na workbenchTUI.

Argumenty aplikace:

- -h = vytiskne všechny argumenty.
- -f [file..] = cesty ke konstručkním souborům, které mají být sestaveny v daném pořadí.
- -i [string..] = vstupy pro poslední konstruované hradlo ve stejném formátu jako při Interaktivním módu
- -a = vygeneruje všechny možné vstupy pro poslední generované hradlo, ignoruje vstupy pro -i