Моделювання логічних елементів за допомогою симулятора логічних схем BUMMEL

Михалевич І.,Рикалюк Р., Тимчук Ю.

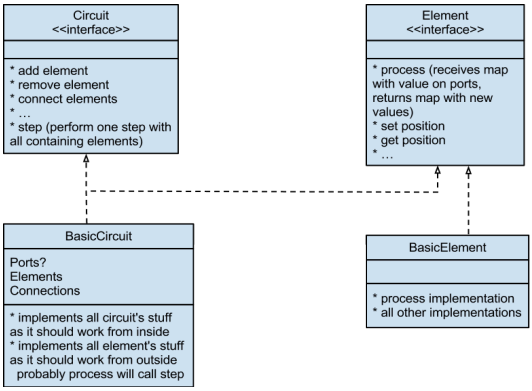
Львівський національний університет імені Івана Франка
rer@franko.lviv.ua

Applied interface approach to creating logical elements as software objects, which simplifies the process of programming of new types of elements and doesn't load too much the processor of the machine on which the executable program is running. Using this approach to setting new element types helped to create the basic elements "AND", "OR", "NOT", analyzer and generator of logic signal.

Відкритий проект зі створення симулятора логічних схем BUMMEL у Львівському університеті триває упродовж двох років. Набуто певний досвід колективного програмування, подолано низку проблем. Для візуалізації роботи кожного з учасників, оцінювання просування роботи над проектом в цілому використано методологію розробки програмного забезпечення Kanban [1]. Для графічного інтерфейсу використовувався ресурс Trello, який дав змогу створити дошку з різними списками, створювати завдання у цих списках, додавати до завдань учасників проекту і цим самим візуалізовувати роботу учасників проекту.

Для симуляції роботи схеми логічних елементів та аналізу її стану застосовують певну математичну модель. Крім того, з точки зору гнучкості платформи, потрібно укласти алгоритм створення нових типів логічних елементів.

Математична модель побудови логічного елемента показана на рисунку.



У такому варіанті існують два інтерфейси - модель та елемент, які передбачають наступну функціональність: схема (Circuit) – володіє можливістю додавання, видалення, з’єднання та роз’єднання елементів, а також можливістю запуску зміни за її правилами. Елемент виконує свою

основну функцію - приймає інформацію на вхід, опрацьовує її і подає на вихід, а також опрацьовує інформацією, яку потрібно безпосередньо для якогось представлення елемента - координати, назва, стан елемента.

Реалізація елементу - інтерфейс елементу, а реалізація схеми - інтерфейс схеми та інтерфейс елементу, оскільки схема може бути елементом всередині якоїсь іншої схеми.

З одного боку алгоритми додавання нових типів елементів можуть передбачати задання елементу повністю у вихідному коді основної мови програмування, та компіляції його з рештою програми. З іншого боку можна створити унікальну скриптову мову для опису типів елементів, і після токенізації скриптів, написаних на цій мові, додавати до базових елементів властивості за допомогою ін’єкції.

Використана модель схеми на кожному відрізку часу опрацьовує кожен елемент, який в свою чергу використовуючи булівські значення на всіх контактах та свою внутрішню функцію обчислює нові значення на контактах. Потім опрацьовуються всі з’єднання елементів, на з’єднанні встановлюється значення, яке дорівнює кон’юнкції значень на його краях. Потім значення на з’єднаннях використовуються як значення на контактах, до яких вони під’єднанні.

Застосовано щось посередині між двома крайніми варіантами. Кожна частина елементу описана відповідним чином. Функція обчислення нових значень на контактах задана кодом основної мови програмування, мапування індексів значень переданих/повернутих функцією до графічного відображення - у файлі з XML форматом та визначеним нами DTD, графічне відображення у форматі векторної графіки SVG.

Модель схеми доволі достовірно відображає статус схеми. Проблеми виникають у циклах, побудованих на тригерах, так як вони можуть змінювати своє значення. Для виправлення цієї проблеми потрібно розширити функціональність, реалізувавши регулярне опрацювання схеми та відображення результатів у реальному часі.

Такий підхід не вимагає великого знання основної мови програмування від творця нових типів елементів і не навантажує надто сильно процесор ЕОМ, на якій виконується програма. Також такий підхід дуже добрий з точки зору програмування так як всі частини зрозуміло вписуються в загальну картину і не заважають сприйняттю та розробці коду. Використовуючи реалізований підхід до задання нових типів елементів нам вдалось створити базові елементи такі як “І”, “АБО”, “НЕ”, а також аналізатор та генератор логічного сигналу.

Джерела:

Р. Рикалюк, Ю.Тимчук, І.Михалевич Особливості організації робочого потоку при програмуванні симулятора BUMMEL. // XVIII Всеукраїнська наук. конфер. «Сучасні проблеми прикладної математики та інформатики», 4-5 жовтня 2012 р. – Львів, 2012. – С. 136-137.