УДК 621.31

О.Б. Одарущенко[1], О.М. Одарущенко[2], В.О. Бутенко[3], В.В. Москалець[2], О.Ю. Стрюк[2]

[1] *Полтавський національний технічний університет імені Юрія Кондратюка, Полтава*
[2] *Науково-виробниче підприємство "Радій", Кропивницький*
[3] *Національний аерокосмічний університет імені М.Є. Жуковського "ХАІ", Харків*

МОДЕЛІ МАТЕМАТИЧНИХ БЛОКІВ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ІНФОРМАЦІЇ ДЛЯ ВЕРИФІКАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГРАМОВАНИХ ЛОГІЧНИХ КОНТРОЛЕРІВ

Моделювання є важливим етапом під час розробки сучасних технічних систем, особливо систем критичних для безпеки, адже дозволяє надати відповіді на велику кількість питань без необхідності проводити додаткові дослідження над коштовним обладнанням. Перевірка коректності роботи базових математичних алгоритмів, що використовуються під час побудови логіки додатків, які виконуються програмованими логічними контролерами (ПЛК), є необхідним завданням етапу їх розробки. У даній статті наведені результати експериментальних досліджень, щодо можливостей використання, у вигляді верифікаційного механізму, потужності одного із найбільш розповсюджених пакетів комп'ютерної математики – Matlab. Даний пакет, а також його компонент Simulink, використовується із метою верифікації роботи математичних алгоритмів розроблених за допомогою мови VHDL, що є базовими блоками при побудові логіки роботи ПЛК, які розробляються із використанням технології FPGA.

***Ключові слова**: програмований логічний контролер, модель, тест-кейс, блок, порт, Simulink, Matlab*

Вступ. Загальні відомості щодо ПЛК та розробки програм керування технологічними процесами

В даний час широкого поширення набули програмовані логічні контролери (ПЛК), які є ядром автоматизованих систем управління технологічних процесів і виробництв. Програмований логічний контролер - це електронний спеціалізований пристрій, що працює в реальному масштабі часу, для автоматизації технологічних процесів. В якості основного режиму роботи ПЛК виступає його тривале автономне використання, часто в несприятливих умовах навколишнього середовища, без серйозного обслуговування і практично без втручання людини. Основними галузями застосування ПЛК є: автоматизація технологічних процесів промислового виробництва; в системах протиаварійного захисту та сигналізації (зокрема АЕС); в верстатах з числовим програмованим управлінням; управління дорожнім рухом; в системах життєзабезпечення будівель; в системах охорони; в медичному обладнані; для керування роботами; в системах зв'язку та багатьох інших.

Типовий ПЛК складається з: процесорного модуля, модулів дискретних і аналогових входів / виходів, та ін. Аналіз сучасного ринку ПЛК дозволяє їх класифікувати наступним чином: за кількістю входів і виходів: наноконтроллери - до 15-20 входів / виходів; малі контролери - до 100 входів / виходів; середні - 100-300; великі - 300-2000; надвеликі - більше 2000 входів / виходів.

Крім цього параметра, що характеризує «потужність» ПЛК, при розробці систем управління і виборі ПЛК враховують також його швидкодію, обсяг різних видів пам'яті і кількість мережевих інтерфейсів.

Конфігурація ПЛК визначається номенклатурою модулів в залежності від цілей управління.

Це модулі: дискретного введення; дискретного виводу; релейного виходу; цифроаналогові та аналогоцифрові перетворювачі по струму і напрузі; вводи з термопар; частотні входи та інш.

Для програмування ПЛК використовуються стандартизовані мови МЕК (IEC) стандарту IEC61131-3 [1]. Найбільш поширеними є наступні:

-LD (Ladder Diagram) - мова релейних схем;

-FBD (Function Block Diagram) - мова функціональних блоків;

-SFC (Sequential Function Chart) - мова діаграм станів - використовується для програмування кінцевих автоматів.

Основні операції ПЛК відповідають комбінаційному управлінню логічними схемами спеціальних агрегатів - механічних, електричних, гідравлічних, пневматичних і електронних. У процесі управління контролери генерують вихідні сигнали (включити або вимкнути) для управління виконавчими механізмами (електродвигунами, клапанами, електромагнітами і вентилями) на підставі результатів обробки сигналів, отриманих від датчиків, або пристроїв верхнього рівня.

Сучасні програмовані контролери виконують також й інші операції, наприклад, поєднують функ-

ції лічильника і таймера, обробляють затримку сигналів. Більшість ПЛК пристосовані для роботи в типових промислових умовах, з урахуванням забрудненої атмосфери, рівнів сигналів, термо- і вологостійкості, ненадійності джерел живлення, а також механічних ударів і вібрацій. З цією метою апаратна частина розміщується в міцний корпус, здатний мінімізувати негативний вплив ряду виробничих та природних факторів.

Зважаючи на високу відповідальність функціонування ПЛК, міжнародними стандартами висуваються вимоги до процесів їх розробки та тестування [2, 3]

Однією із вимог стандартів до цих процесів (наприклад, процесу верифікації програмного коду) є необхідність виконання альтернативних обчислень основних математичних функцій, які реалізує програмне забезпечення ПЛК.

Метою цієї статті є опис одного із можливих підходів до рішення цієї задачі. З метою верифікації обчислень, які містяться у досліджуваних математичних блоках ПЛК необхідно було створити бібліотеки засобами комп'ютерної математики та виконати тестування розроблених альтернативних блоків.

Сучасний маркет систем комп'ютерної математики представлено такими найбільш розповсюдженими середовищами як Matlab, Maple, Mathematica, MathCad та ін. В рамках проведених експериментальних досліджень використовувалась система Matlab [5] у зв'язку із її певними перевагами, а саме орієнтацією на широке коло технічних задач, перш за все на чисельні розрахунки, візуалізацію і технічні додатки. Додаток Simulink — це інтерактивна система, яка використовується для моделювання об'єктів різної природи. Simulink працює з лінійними, нелінійними, неперервними, дискретними, багатовимірними системами [6].

1. Розробка моделі математичного блоку засобами Matlab

В якості функціонального блоку для розробки моделі було обрано обчислювач, який виконує математичні операції додавання, віднімання, множення та ділення.

Механізм роботи базується на міжнародному стандарті "IEEE Standard 754 for Floating-Point Arithmetic" [4], згідно якого було обрано формат представлення чисел з плаваючою комою з одинарною точністю: ***Floating Point 32 bits*** (FP32b). Також модель дозволяє виконувати обчислення для цілих знакових чисел формату ***Signed Integer 32 bits*** (SI32b).

Для побудови моделі було використано підсистему типу *"Triggered Subsystem"*, що знаходиться в бібліотеці *"Simulink/ Ports & Subsystems"*. Особливістю такої підсистеми є те, що вона спрацьовує тільки в момент зміни значення тригера (*"i_start"*) з '0' на '1'. Тобто систему легко контролювати за допомогою сигналу типу *"clock"* (тактовий годинник), який широко використовується у роботі сучасних електронно-обчислювальних механізмів, наприклад, в FPGA.

"Black-box" моделі наведена на рис. 1.

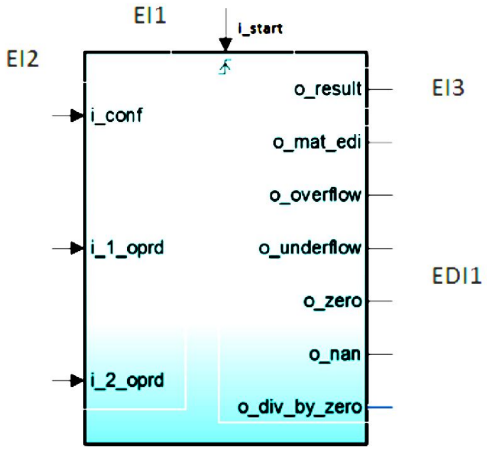

Рис. 1. Зовнішній вигляд моделі МАТ-блока

Модель побудована для того, щоб виконувати одну з заданих операцій (+ - * /) над операндами *"i_1_oprd"* та *"i_2_oprd"*.

Вибір операції здійснюється за допомогою параметра *"i_conf"*. Для цього на вхід *"i_conf"* необхідно подати число, що відповідає бажаній операції, а саме:

1 – додавання (SI32b);
2 – віднімання (SI32b);
3 – множення (SI32b);
4 – ділення (SI32b);
5 – додавання (FP32b);
6 – віднімання (FP32b);
7 – множення (FP32b);
8 – ділення (FP32b).

Результатом виконання операції є числове значення (виводиться на порт *"o_result"*) та прапорці діагностики (*"o_mat_edi"*, *"o_overflow"*, *"o_underflow"*, *"o_zero"*, *"o_nan"*, *"o_div_by_zero"*).

Детальний опис інтерфейсів (рис. 1) блока розглянемо в табл. 1.

Розглянемо структурні схеми кожної операції (рис. 2 – 9).

Дані на вхідні порти блоку потрапляють у вигляді шини даних (набір 0 та 1) і в залежності від обраного формату даних конвертуються в числа (SI32b чи FP32b). Результат обраховується за допомогою окремого блоку, в налаштуваннях якого уже обрана потрібна операція та тип даних.

Особливості формування прапорців діагностики наведені далі:

Таблиця 1

Зовнішні інтерфейси (EI) моделі MAT-блока

ID	Назва сигналу	I/O	Тип	Опис
EI1	i_start	In	Trigger	Стартовий сигнал
EI2	i_conf[4..0]	In	Data	Параметр конфігурації
	i_1_oprd[31..0]	In	Data	Перший (Лівий) операнд
	i_2_oprd[31..0]	In	Data	Другий (Правий) операнд
EI3	o_result[31..0]	Out	Data	Числовий результат
EDI1	o_mat_edi	Out	Flag	Сигнал діагностики – приймає значення ‘1’, якщо параметр конфігурації заданий некоректно
	o_overflow	Out	Flag	Прапорець переповнення – приймає значення ‘1’, якщо результат обчислень перевищує допустимий діапазон
	o_underflow	Out	Flag	Прапорець спустошення – приймає значення ‘1’, якщо результат обчислень потрапляє в діапазон денормалізованих чисел
	o_zero	Out	Flag	Прапорець отримання нуля – приймає значення ‘1’, якщо результат обчислень набуває значення 0
	o_nan	Out	Flag	Прапорець NaN – приймає значення ‘1’, якщо результат обчислень NaN (Не число)
	o_div_by_zero	Out	Flag	Прапорець ділення на 0 – приймає значення ‘1’, якщо другий операнд (дільник) рівний 0. Актуально тільки для операції ‘/’.

1) o_mat_edi – основний прапорець (якщо параметр конфігурації заданий некоректно, то результат роботи блоку вважається "порожнім");

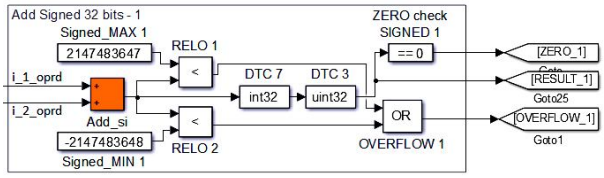

Рис. 2. Структурна схема операції додавання (SI32b)

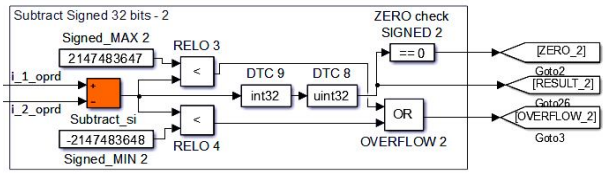

Рис. 3. Структурна схема операції віднімання (SI32b)

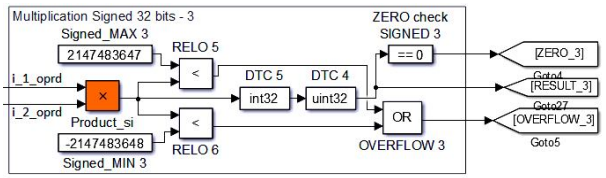

Рис. 4 .Структурна схема операції множення (SI32b)

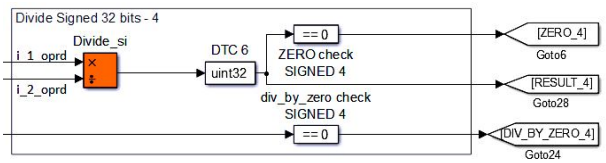

Рис. 5. Структурна схема операції ділення (SI32b)

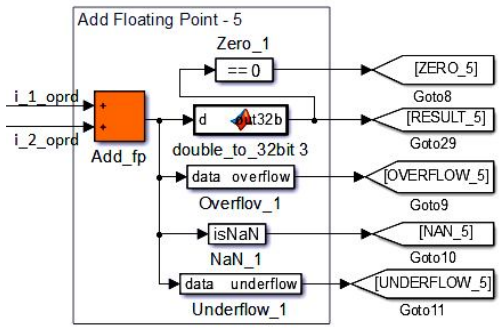

Рис. 6. Структурна схема операції додавання (FP32b)

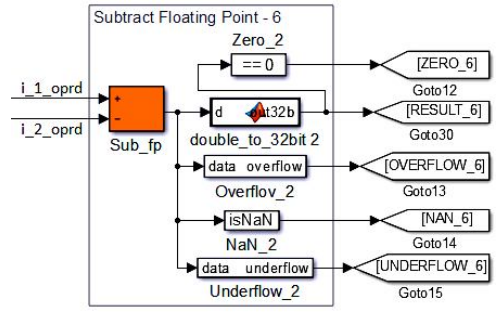

Рис. 7. Структурна схема операції віднімання (FP32b)

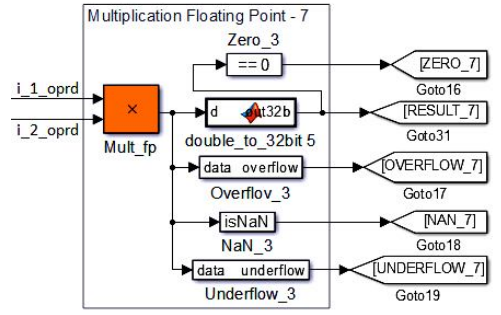

Рис. 8. Структурна схема операції множення (FP32b)

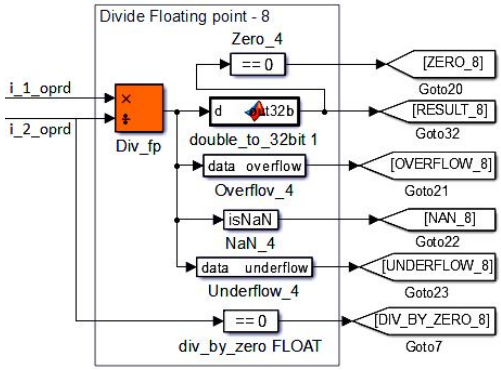

Рис. 9. Структурна схема операції ділення (FP32b)

2) o_overflow – перевірка переповнення виконується порівнянням результату з верхньою та нижньою межею діапазону (для SI32b та FP32b межі відрізняються, рис. 2 та рис. 10);
3) o_underflow – перевірка спустошення виконується порівнянням результату з верхньою та нижньою межею діапазону спустошення (актуально тільки для FP32b, рис. 11);
4) o_zero – визначається прирівнюванням результату обчислень до 0;
5) o_nan – якщо вся експонента числа заповнена '1' та хоча б один біт мантиси рівний '1', то результат вважається не числом (актуально тільки для FP32b);
6) o_div_by_zero – перевіряємо чи не рівний 0 дільник (актуально тільки для операції ділення).

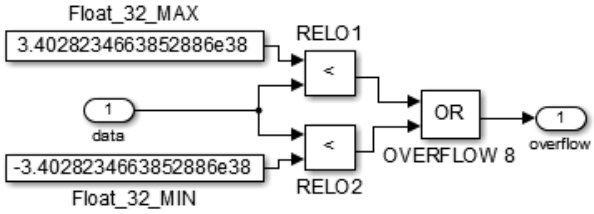

Рис. 10. Структурна схема обчислення Прапорця переповнення (FP32b)

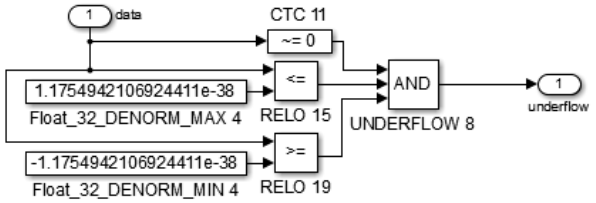

Рис. 11. Структурна схема обчислення Прапорця спустошення (FP32b)

Таблиця 2

Актуальність прапорців діагностики в залежності від типу даних та типу операції

	Overflow	Underflow	Zero	NaN	Div_by_zero
SI32b +	✔		✔		
SI32b -	✔		✔		
SI32b *	✔		✔		
SI32b /			✔		✔
FP32b +	✔	✔	✔	✔	
FP32b -	✔	✔	✔	✔	
FP32b *	✔	✔	✔	✔	
FP32b /	✔	✔	✔	✔	✔

В цілому актуальність прапорців діагностики описана в табл. 2.

Тест-кейс має вигляд, котрий наведений на рис. 12.

2. Дослідження результатів роботи моделі математичного блоку

Для перевірки коректності роботи моделі MAT-блока засобами Simulink було створено унікальний *тест-кейс*: у нашому випадку це додаткова модель, у яку поміщається блок. З її допомогою подаються вхідні дані та виконується порівняння отриманого результату з очікуваним.

Розроблений набір тестових векторів складається з двох частин: *набору вхідних даних* та *очікуваного результату,* які можна детально переглянути в табл. 3.

Кожен тестовий вектор призначений для перевірки унікальної ситуації. Наприклад, подаємо на вхід *i_conf* значення 10 і таким чином очікуємо, що спрацює діагностика (Прапорець *o_mat_edi* буде рівний '1').

Після запуску моделі у спеціальний візуальний блок *Scope* заносяться результати відпрацювання тестових векторів (рис. 13).

Перший графік відповідає за тактовий годинник, який подається на тригер моделі *i_start*.

Другий графік показує номер тестового вектора.

Третій графік відповідає за порівняння отриманого результату з очікуваним. Якщо результати будуть відрізнятися хоча б по одному виходу, то прапорець прийме значення '1', інакше він рівний '0'.

Таблиця 3

Набір тестових векторів

№ тест-вектора	Набір вхідних даних			Очікуваний результат						
	i_conf	i_1_oprd	i_2_oprd	o_result	o_mat_edi	o_overflow	o_underflow	o_zero	o_nan	o_div_by_zero
1	0	7	7	0	**1**	0	0	0	0	0
2	10	7	7	0	**1**	0	0	0	0	0
3	1	2	2	4	0	0	0	0	0	0
4	1	2E+09	2000000000	2147483647	0	**1**	0	0	0	0
5	1	-3	3	0	0	0	0	**1**	0	0
6	2	6	2	4	0	0	0	0	0	0
7	2	-2,13E+09	2106975000	-2147483648	0	**1**	0	0	0	0
8	2	-2,04E+08	-203789218	0	0	0	0	**1**	0	0
9	3	2	2	4	0	0	0	0	0	0
10	3	2,118E+09	2097558029	2147483647	0	**1**	0	0	0	0
11	3	2	0	0	0	0	0	**1**	0	0
12	4	2	2	1	0	0	0	0	0	0
13	4	0	1	0	0	0	0	**1**	0	0
14	4	1	0	2147483647	0	0	0	0	0	**1**
15	5	1,5	3,5	5	0	0	0	0	0	0
16	5	3E+38	3E+38	inf	0	**1**	0	0	0	0
17	5	-1,1E-37	1,10003E-37	2,993E-42	0	0	**1**	0	0	0
18	5	-25,6	25,6	0	0	0	0	**1**	0	0
19	5	nan	25,6	nan	0	0	0	0	**1**	0
20	6	3,3	1,3	2	0	0	0	0	0	0
21	6	-3,2E+38	3,1E+38	-inf	0	**1**	0	0	0	0
22	6	-1,11E-37	-1,0098E-37	-1,002E-38	0	0	**1**	0	0	0
23	6	-2,5	-2,5	0	0	0	0	**1**	0	0
24	6	nan	1,5	nan	0	0	0	0	**1**	0
25	7	1,1	2,6	2,86	0	0	0	0	0	0
26	7	2,6E+38	3,1E+38	inf	0	**1**	0	0	0	0
27	7	-1,11E-37	0,01	-1,11E-39	0	0	**1**	0	0	0
28	7	0	3,1	0	0	0	0	**1**	0	0
29	7	nan	2,45	nan	0	0	0	0	**1**	0
30	8	2,6	2,6	1	0	0	0	0	0	0
31	8	3,4E+37	0,00001	inf	0	**1**	0	0	0	0
32	8	-1,11E-37	100	-1,11E-39	0	0	**1**	0	0	0
33	8	0	1,4	0	0	0	0	**1**	0	0
34	8	nan	3,2	nan	0	0	0	0	**1**	0
35	8	1,89	0	-inf	0	0	0	0	0	**1**

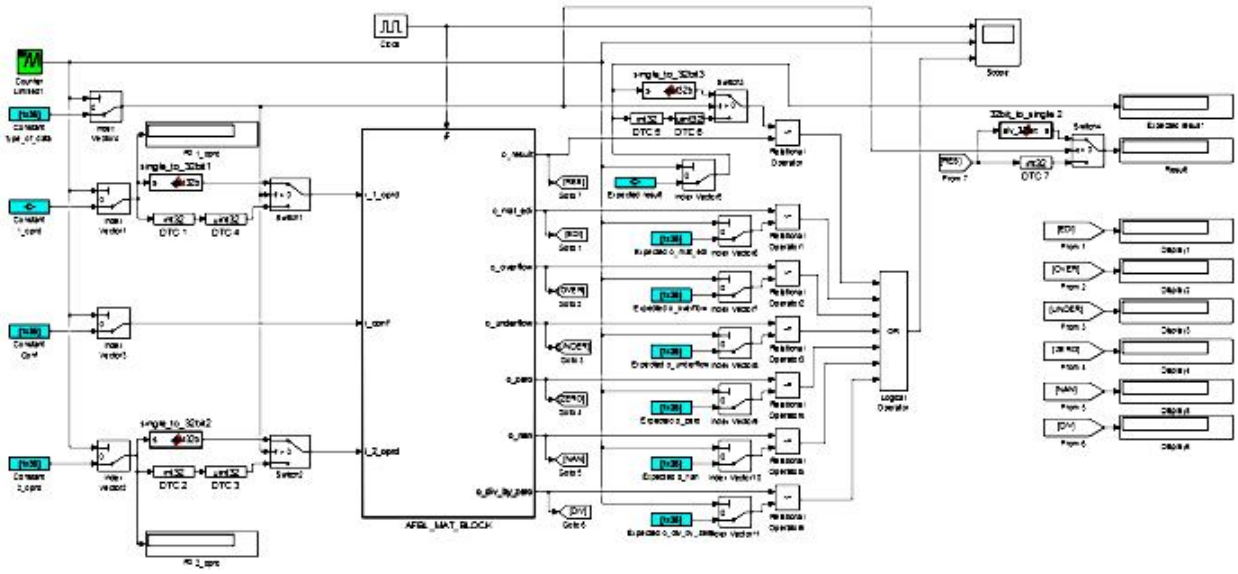

Рис. 12. Модель тест-кейса

Так як прапорець на всьому тест-кейсі залишався рівним ‘0’, робимо висновок, що усі очікувані результати співпадають з обчисленими в MAT-блоці.

Отже, модель розроблена коректно.

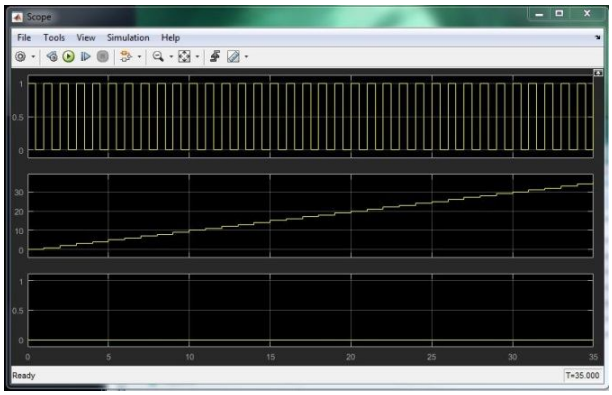

Рис. 13. Результат роботи тест-кейса

Висновки

Результатом проведених досліджень є розроблена засобами комп’ютерної математики, досліджена та верифікована модель математичного блоку із складу бібліотеки математичних блоків. Ці моделі можуть бути використані для розробки алгоритмів програмних додатків, які обробляють програмовані логічні контролери.

Проведені дослідження відкривають шлях до розробки та впровадження процедури альтернативних обчислень, виконання якої вимагають міжнародні стандарти для складних технічних систем, що розробляються для критичних галузей промисловості.

Список літератури

1. *IEC 61131-3: 2013. Programmable controllers-Part 3: Programming languages.*

2. *IEC 61508:2010. Functional safety of electrical/electronic/programmable electronic safety-related systems.*

3. *ASME NQA-1-2008. Quality Assurance Requirements for Nuclear Facility Applications, Part 1, Requirement 11, Test Control.*

4. *IEEE Computer Society IEEE Standard 754 for Floating-Point Arithmetic. — New York : IEEE, 2008. — 70 с. — ISBN 978-0-7381-5752-8.*

5. *Сивохин А. В., Мещеряков Б. К. Решение задач оптимального управления с использованием математической системы MATLAB и пакета имитационного моделирования SIMULINK. Лабораторный практикум по основам теории управления. – Пенза: Изд-во Пенз. гос. ун-та, 2006 – 120с.*

6. *Терёхин В.В. Моделирование в системе MATLAB: Учебное пособие /Кемеровский государственный университет. – Новокузнецк: Кузбассвузиздат, 2004. -376с.*

Надійшла до редакції 24.04.2017

Рецензент: д-р техн. наук, проф. Л.І. Лєві, Полтавський національний технічний університет імені Юрія Кондратюка, Полтава.

МОДЕЛИ МАТЕМАТИЧЕСКИХ БЛОКОВ ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ ДЛЯ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ

Е.Б. Одарущенко, О.Н. Одарущенко, В.О. Бутенко, В.В. Москалец, А.Ю. Стрюк

Моделирование является важным этапом при разработке современных технических систем, особенно систем критичных для безопасности, поскольку позволяет дать ответы на множество вопросов без необходимости проводить дополнительные исследования над ценным оборудованием. Проверка корректности работы базовых математических алгоритмов, используемых при построении логики приложений, которые выполняются программируемыми логическими контроллерами (ПЛК), – необходимая задача этапа их разработки. В данной статье приведены результаты экспериментальных исследований возможностей использования в виде верификационного механизма мощности одного из самых распространенных пакетов компьютерной математики – Matlab. Данный пакет, а также его компонент Simulink, используется с целью верификации работы математических алгоритмов разработанных с помощью языка VHDL, которые являются базовыми блоками при построении логики работы ПЛК, разрабатываемых с использованием технологии FPGA.

Ключевые слова: *программируемый логический контроллер, модель, тест-кейс, блок, порт, Simulink, Matlab.*

MATHEMATICAL MODEL OF BLOCKS DISCRETE TRONSFORM INFORMATION FOR SOFTWARE VERIFICATION PROGRAMMABLE LOGIC CONTROLLERS

O.B. Odarushchenko, O.M. Odarushchenko, V.O. Butenko, V.V. Moscalets, O.Yu. Strjuk

Modelling is an important step during the development of advanced technology systems, especially safety-critical systems, as it helps to answer many questions without the need to conduct additional studies on the expensive equipment. Checking the accuracy of the basic mathematical algorithms used in the construction of application logic performed by programmable logic controllers (PLC), is a necessary task stage during their development. This article presents the results of experimental studies on the possibilities of use Matlab as a verification mechanism. This package and its components Simulink were used to verify the mathematical algorithms developed with VHDL language, which are the basic blocks in the PLC logic developed using FPGA technology.

Keywords: *programmable logic controller, model, test cases, power port, Simulink, Matlab.*