УДК 621.004.9

д-р техн. наук Толюпа С. В. ORCID: 0000-0002-1919-9174 (КНУ)
канд. техн. наук Штаненко С. С. ORCID: 0000-0001-9776-4653 (ВІТІ)
Побережець Т. В. ORCID: 0000-0001-8007-8614 (ВІТІ)
Лозунов В. К. (в/ч А0307)

МЕТОДИКА ПРОЄКТУВАННЯ РОБОТИЗОВАНИХ СИСТЕМ В БАЗІСІ САПР *INTEL QUARTUS PRIME*

У даний час при розробці роботизованих систем все більше застосовуються програмовані логічні інтегральні мікросхеми (ПЛІС).

Істотною перевагою ПЛІС є їхня універсальність і можливість швидкого програмування під виконання функцій практично будь-якого цифрового пристрою роботизованої системи. ПЛІС являє собою напівфабрикат, на основі якого розробник, що володіє персональним комп'ютером, має можливість проєктування цифрового пристрою в рекордно короткі терміни. Забезпечується це нескладними і відносно недорогими апаратними засобами програмування та спеціальним програмним забезпеченням, що називається системою автоматизованого проєктування (САПР).

ПЛІС – це електронний компонент, який використовується для створення цифрових інтегральних схем. На відміну від звичайних цифрових мікросхем, логіка роботи ПЛІС задається за допомогою програмування спеціальних засобів: програматорів і програмного забезпечення. Програмування на ПЛІС здійснюється за допомогою мов опису апаратури Verilog HDL і VHDL. На верхньому рівні ці мови дуже схожі – модель апаратури описується у вигляді взаємодіючих блоків (модулів) і для кожного з них визначається інтерфейс і реалізація. Інтерфейси модулів описують вхідні, вихідні і двосторонні порти, завдяки яким модулі з'єднуються один з одним з метою обміну даними, а також управління сигналами. Реалізація задає елементи внутрішнього стану і порядок обчислення значень вихідних інтерфейсів на основі цього стану і значень вхідних портів, а також правила поновлення внутрішнього стану.

У статті розкриті етапи проєктування цифрових пристроїв роботизованих систем за допомогою ПЛІС, розглянуті принципи побудови і функціонування основних вузлів комбінаційних схем, на логічних елементах реалізована одна із заданих функцій, яка в подальшому запрограмована на ПЛІС за допомогою САПР Quartus Prime із вбудованим симулятором ModelSim-Altera .

Ключові слова: *логічна схема, функція, мова опису апаратури, програмована логічні інтегральна мікросхема, система автоматизованого проєктування.*

S. Toliupa, S. Shtanenko, T. Poberezhets, V. Lozunov. Methodology for designing robotic systems based on CAD Intel Quartus Prime.

At present, programmable logic integrated circuits (FPGAs) are increasingly used in the development of robotic systems. A significant advantage of FPGAs is their versatility and the ability to quickly program to perform the functions of almost any digital device of a robotic system. FPGA is a semi-finished product, on the basis of which a developer with a personal computer has the ability to design a digital device in record time. This is provided by simple and relatively inexpensive software hardware and special software called computer-aided design (CAD). FPGA is an electronic component used to create digital integrated circuits. Unlike conventional digital chips, the logic of FPGA operation is set by programming using special tools: programmers and software. FPGA programming is performed using the description languages Verilog HDL and VHDL. At the upper level, these languages are very similar - the hardware model is described in the form of interacting blocks (modules) and for each of them is defined interface and implementation. Module interfaces describe the input, output, and two-way ports through which modules connect to each other for data exchange as well as control signals. The implementation sets the elements of the internal state and the order of calculating the values of the output interfaces based on this state and the values of the input ports, as well as the rules for updating the internal state. The article reveals the stages of designing digital devices of robotic systems using FPGA, considers the principles of construction and operation of the main nodes of combinational circuits, logical elements implemented one of the specified functions, which is subsequently programmed on FPGA using CAD Quartus Prime with built-in simulators Models.

Keywords: *logic circuit, function, hardware description language, programmable logic integrated circuit, computer-aided design system.*

Постановка завдання. На сьогоднішній день на театрі воєнних дій все більше спостерігається використання роботизованих систем, які вже без сумнівів відіграють вирішальну роль на полі бою, а згодом, на думку експертів, в подальшому повинні частково замінити людину. Створення таких систем висуває жорсткі вимоги до самих цифрових пристроїв зі швидкодії, функціональних можливостей, габаритів, потужностей, надійності, вартості та інших параметрів. Найбільш ефективним підходом вирішення цієї проблеми є орієнтація при створенні сучасних роботизованих систем на новітню елементну базу.

Яскравим прикладом такої елементної бази є програмовані логічні інтегральні мікросхеми (ПЛІС), які органічно поєднують в собі широкі можливості і гнучкість замовних інтегральних схем з доступністю і зручністю застосування традиційної «жорсткої» логіки.

Мікросхеми такого типу являють собою матрицю програмованих логічних елементів з *CPLD* (*Complex Programmable Logic Device*), *FPGA* (*Field-Programmable Gate Array*), *FLEX* (*Flexible Logic Element Matrix*) та *SoC* (*System-on-Chip*) архітектурою, між якими прокладені електричні комутовані з'єднання. Це дозволяє конфігурувати окремі компоненти і створювати зв'язку між ними шляхом завантаження в ПЛІС потоку даних, що включає необхідні електричні кола і вузли комутації. В результаті з існуючих у складі ПЛІС програмованих логічних елементів створюється необхідна цифрова схема, яка за необхідності може бути легко модифікована [1].

Аналіз останніх досліджень. На сьогоднішній день існує багато наукових робіт, присвячених проєктуванню цифрових пристроїв. Зокрема, робота [2] присвячена придбанню початкових відомостей з проєктування цифрових пристроїв як з використанням ручних методів мінімізації булевих функцій, так і з використанням системи автоматизованого проєктування (САПР) ПЛІС, а також вивчення деяких повноважень з управління логічним синтезом схем.

Робота [3] являє собою введення в САПР і дає загальне уявлення про типові етапи проєктування ПЛІС, від ідеї до готового виробу. Описано два підходи до проєктування схем в ПЛІС: перший – графічне введення, коли користувач креслить схеми електричних кіл, використовуючи готові шаблони, другий – кодовим описом логічних схем.

У роботі [4] розглянуті питання проєктування софт-процесорів, що конфігуруються, або які здатні надати проєкту в базісі ПЛІС всі елементи стандартної мікроконтролерної системи, включаючи можливість програмування отриманого пристрою за допомогою звичайних мов високого рівня.

Мета статті. У статті авторами на відмінність від попередніх робіт розглянуто трудомісткість етапів проєктування цифрових пристроїв, а саме опис алгоритму та його декомпозиція на частини, враховуючи специфіку застосування, а також розглянутий процес проєктування, що полягає у вирішенні задачі синтезу структури на прикладі комбінаційної схеми, яка задається булевою функцію як невід'ємною складовою роботизованої системи з подальшою реалізацією в базісі САПР *Intel Quartus Prime*.

Виклад основного матеріалу. Проєктування цифрових пристроїв, як основа будь-якої роботизованої системи на основі ПЛІС, містить наступні основні етапи:

формулювання концепції – постановка задачі та концептуальний опис алгоритму функціонування пристрою, що розробляється;

введення проєкту – представлення цифрового пристрою, що проєктується в «зрозумілому» для програмного засобу вигляді (принципова схема, часові діаграми, текстовий файл на спеціальній мові програмування *AHDL*, *VHDL*, *Verilog HDL*);

компіляція проєкту – логічний синтез, мінімізація, розведення і укладання проєкту в ПЛІС, а також створення файлів в спеціальному форматі, що містять всю інформацію для програмування мікросхеми;

верифікація проєкту – функціональне або тимчасове моделювання, а також часовий аналіз цифрового пристрою, що проєктується. Зазвичай, це робиться за допомогою побудови часових діаграм, де стан входів задаються користувачем, а стан виходів визначаються за допомогою програмного засобу, виходячи з закладеного алгоритму функціонування пристрою, що розробляється. При виявленні помилок або збоїв в роботі пристрою проводиться повернення до етапу створення проєкту з метою виправлення помилки. Процес повторюється до виправлення всіх помічених неточностей, завдяки чому ще до програмування ПЛІС вдається локалізувати в проєкті переважну більшість помилок;

програмування і тестування – кінцевий етап проєктування. Залежно від типу ПЛІС, що використовується і схемотехнічного рішення її апаратного обрамлення в пристрою, що розробляється, цей етап здійснюється або за допомогою програматора, або безпосередньо на робочій платі, в тому числі і динамічно, під час роботи пристрою.

Процес проєктування та верифікації цифрових пристроїв виконується засобами САПР. Одним з трудомістких етапів на етапі проєктування є постановка завдання, що полягає в описі алгоритму і його декомпозиції на частини, кожна з яких являє собою алгоритм, який має відоме апаратне уявлення [5].

Будь-який алгоритм можна представити послідовністю функціональних операторів:

$$F_i(X,\ Y) = (y_1^i,\ y_2^i,\ \ldots,\ y_r^i),$$

де X – множина вихідних даних;

Y – множина проміжних результатів, отриманих у процесі виконання оператора – F_i.

Узагальнена архітектура реконфігурованого пристрою може бути представлена у вигляді:

$$S = < P,\ A,\ F >,$$

де $P = \{P_i\}$ – множина об'єктів управління ($i = \overline{1, n}$);

$A_i = \{A_{ij}\}$ – множина алгоритмів управління, що реалізують функцію відображення $A_i : X_{ij} \to Y_{ij}$ множина вхідних сигналів $\{X_{ij}\}$ в множині вихідних сигналів $\{Y_{ij}\}$ для i-го об'єкта ($j = \overline{1, m}$);

$F = \{F_k\}$ – множина файлів конфігурації ($k = m \times n$), що визначають структури реалізації алгоритмів A_{ij} об'єктів управління P_i.

Якщо при апаратній реалізації алгоритм A_{ij} не вдається розмістити в один кристал, то цей алгоритм розбивається на фрагменти, що виконуються послідовно. Складність фрагментів алгоритму при цьому визначається логічною ємністю кристала. Відповідні цим фрагментам файли конфігурації F_{kl} завантажуються в кристал послідовно. Конфігурація кристала ПЛІС здійснюється шляхом запису файла конфігурації, сформованого за допомогою системи САПР *Intel Quartus Prime*.

Обсяг пам'яті, необхідний для зберігання множини F файлів конфігурації, буде визначатися величиною:

$$\Xi = q \times z \times k,$$

де z – число фрагментів алгоритму A_{ij};

q – обсяг пам'яті, необхідний для зберігання одного файла конфігурації.

Розглянутий алгоритм може бути основою для побудови моделі цифрового пристрою, що проєктується.

Враховуючи той факт, що сучасні роботизовані системи відносяться до програмно-реконфігурованих пристроїв (ПРП), в яких фіксоване поле заданої розмірності, налаштоване спеціально для виконання певного заданого алгоритму або його частини, постає питання забезпечення реалізації цього алгоритму оптимальним способом з точки зору часу його виконання і витрат апаратних ресурсів.

Для визначення оптимальної кількості рівнів програмованих компонентів (за які використовується ПЛІС) необхідно розглядати обробляючу систему (процесор) ПРП як інформаційну систему, вся інформація в якій віднесена до трьох сфер станів: зберігання, транспортування і перетворення. Очевидно, що при певних співвідношеннях між об'єктами інформації в цих сферах можна отримати оптимальні технічні параметри ПРП. Оптимальною вважається така структурна реалізація моделі ПРП, для якої відповідно до прийнятих критеріїв знайдені оптимальна кількість рівнів і оптимальне співвідношення між узагальненими характеристиками компонент на кожному рівні, а також між відповідними характеристиками компонент сусідніх рівнів.

Модель цифрового пристрою, що проєктується, може бути представлена четвіркою [6]:

$$S = < \Omega,\ A,\ B,\ D >,$$

де Ω – множина математичних методів для предметної області, що лежать в основі функціонування пристрою;

A – множина алгоритмів реалізації методу;

$B = \{b\}$ – алфавіт конструктивів, з яких синтезується структура;

D – процедура опису проєкту (опис об'єкта).

Таким чином, процес проєктування полягає у вирішенні задачі синтезу структури на основі конструктивів $\{b\}$ алфавіту B для виконання певного алгоритму, що реалізує метод Ω, що лежить в основі функціонування структури, відповідно до вимог специфікацій. Результатом процедури є опис проєкту мовою програмування в базісі САПР *Intel Quartus Prime.*

Запропоновано синтез структурної реалізації послідовності алгоритмів, коли метод (M) представляється послідовністю алгоритмів:

$$A_i, \forall i = \overline{1, \dots, n};$$
$$\Omega = \bigcup_i A_i .$$

У програмно-реконфігурованих пристроях спочатку задана базова (нульова) архітектура, реалізована на ПЛІС у вигляді функціонального поля фіксованої розмірності, контролера шини *host*-комп'ютера, поля пам'яті, а також добре структурованої бібліотеки файлів конфігурацій (БФК) структурних реалізацій методів (алгоритмів), що виконують відображення алгоритму в структурну реалізацію ($F : A_i \to B_i$). Кожен алгоритм має відображення ($F : A_i \to B_i$) в структурну реалізацію (B_i), яка являє собою файл конфігурації для кристала ПЛІС. У загальному випадку є кілька варіантів реалізації алгоритму (наприклад, послідовний, послідовно-паралельний і паралельний):

$$B_i = \bigcup_i A_{ij} .$$

Кожен варіант характеризується параметрами швидкодії (час виконання t_{ij}) й апаратними витратами (q_{ij}).

Припускаємо, що потужність множини t_{ij} є достатньою для реалізації широкого набору алгоритмів. У тому випадку, якщо необхідна реалізація t_{ij}-го алгоритму в бібліотеці відсутній, то необхідно за допомогою інструментальних засобів САПР створити її і включити як стандартний елемент в бібліотеку.

Таким чином, завдання оптимізації зводиться до впорядкованого призначення кожної i-ї вершини графа реалізованого алгоритму B_{ij}-го елемента бібліотеки з метою отримання екстремального значення деякого критерію якості. Будь-який оператор відображається тільки одним елементом з бібліотеки. В результаті визначається структура, яка реалізує заданий граф.

Тоді рішення задачі може бути отримано методами цілочисельного математичного програмування.

Завдання оптимізації полягає у визначенні мінімуму цільової функції, а критерієм якості є сумарний час виконання всіх алгоритмів і витрати обладнання:

$$\sigma_1 \sum_i \sum_j t_{ij} \lambda_{ij} + \sigma_2 \sum_i \sum_j q_{ij} \lambda_{ij} = \min \left(\forall i = \overline{1, \dots, n};\ j = \overline{1, \dots, k}\right)$$

при обмеженнях

$$\sum_i \sum_j t_{ij} \lambda_{ij} \le T_0 ; \quad \sum_i \sum_j q_{ij} \lambda_{ij} \le Q_0 ; \quad \sum_j x_{ij} = 1 (\forall i = \overline{1, \dots, n};\ j = \overline{1, \dots, k}),$$

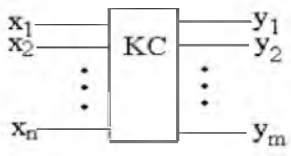
де σ_1, σ_2 – вагові коефіцієнти, які можуть бути визначені експертним шляхом;

Q_0 – гранично допустимі апаратні затрати;

T_0 – гранично допустимі часові затрати.

Теоретичною базою при проєктуванні цифрових пристроїв як основа сучасної роботизовано системи є булева алгебра, двійкова арифметика та теорія кінцевих автоматів. Функцією алгебри логіки (булева функція) називається функція, яка як і її аргументи, може приймати лише два значення: 0 або 1. Булеві функції є описом комбінаційних схем.

Розглянемо синтез комбінаційної схеми. Логічну схему, яка має n входів $X = \{x_1, x_2, \dots, x_n\}$ та m виходів $Y = \{y_1, y_2, \dots, y_m\}$, можемо представити у вигляді (рис. 1) [7].



$$\left.\begin{aligned} y_1 &= y_1(x_1, x_2, \dots, x_n), \\ y_2 &= y_2(x_1, x_2, \dots, x_n), \\ &\dots\dots\dots\dots\dots\dots \\ y_m &= y_m(x_1, x_2, \dots, x_n). \end{aligned}\right\} \quad (1)$$

Рис. 1. Комбінаційна схема

Логічна схема називається комбінаційною, якщо значення множини $Y = \{y_1, y_2, \dots, y_m\}$ її виходів можуть бути виражені як система m булевих функцій від множини вхідних змінних $X = \{x_1, x_2, \dots, x_n\}$ наступного вигляду:

Кожна функція $y_i = y_i(x_1, x_2, \dots, x_n)$ при $i = 1, 2, \dots, m$ визначає значення на виході схеми $y_i \in \{0, 1\}$ для будь-якого двійкового набору $(e_1, e_2, \dots, e_n)$, $e_j \in \{0, 1\}$ при $j = 1, 2, \dots, n$, який подається на незалежні входи схеми.

Система (1) описує залежність між входами та виходами схеми, але не дає представлення про її внутрішню структуру. Функціонування комбінаційної схеми можемо представити також у вигляді таблиці істинності, яка має 2^n рядків (по рядку для кожного набору вхідних змінних) та $(n+m)$ стовпців (n стовпців для входів та m стовпців для виходу схеми). Задача синтезу комбінаційної схеми складається з побудови схеми для заданої булевої функції або системи булевих функцій на основі певної системи логічних елементів. Як правило, початковий опис для синтезу схеми задається або у вигляді таблиці істинності, або в аналітичній формі у вигляді системи (1).

При вирішенні задачі синтезу комбінаційної схеми, що реалізує задану булеву функцію, попередньо проводиться мінімізація булевої функції та подальше спрощення мінімальної форми шляхом факторизації і декомпозиції. Комбінаційна схема будується в заданому базисі, як правило, з урахуванням коефіцієнта об'єднання по входах і коефіцієнта розгалуження по виходу.

В якості прикладу розглянемо булеву функцію

$f(x, y, z) = xy \vee \bar{y}z \vee \bar{x}y\bar{z} \vee yz \vee \overline{x \vee \bar{y} \vee z}$, для якої:

1. Побудуємо логічну схему, яка реалізує булеву функцію $f(x, y, z)$ за допомогою логічних елементів «І», «АБО», «НІ» та знайдемо затримку та ціну за Квайном.

2. Напишемо таблицю істинності даної функції.

3. Знайдемо фактичні змінні функції $f(x, y, z)$.

4. Використовуючи основні еквівалентності, перетворимо дану формулу в еквівалентну до неї, при цьому не містить фіктивних змінних. Побудуємо логічну схему, знайдемо її затримку та ціну за Квайном [8–11].

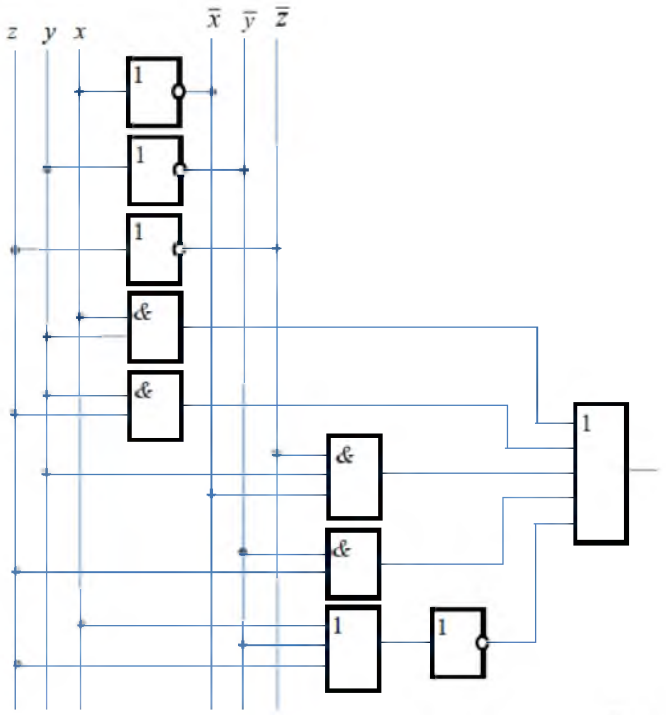


Рис. 2. Логічна схема, яка реалізує булеву функцію $f(x, y, z) = xy \vee \bar{y}z \vee \bar{x}y\bar{z} \vee yz \vee \overline{x \vee \bar{y} \vee z}$

Розв'язок

1. Для схеми на рис. 2 затримка – $T = 4\tau$, ціна за Квайном – $S_Q = 21$.

2. Побудуємо таблицю істинності (табл. 1), визначимо порядок виконання операцій відповідно до їх пріоритетності: $f_1 = \bar{x}$; $f_2 = \bar{y}$; $f_3 = \bar{z}$; $f_4 = x \vee f_2 \vee z$; $f_5 = \bar{f}_4$; $f_6 = xy$; $f_7 = f_2 \cdot z$; $f_8 = f_1 \cdot y \cdot f_3$; $f_9 = yz$; $f = f_5 \vee f_3 \vee f_7 \vee f_8 \vee f_9$.

Таким чином, кількість операцій, необхідних для отримання векторного значення функції, дорівнює десяти. Виходячи з цього, таблиця істинності має наступний вигляд (табл. 1):

Таблиця 1

x	y	z	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f
0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	1	1	1	0	1	0	0	1	0	0	1
0	1	0	1	0	1	0	1	0	0	1	0	1
0	1	1	1	0	0	1	0	0	0	0	1	1
1	0	0	0	1	1	1	0	0	0	0	0	0
1	0	1	0	1	0	1	0	0	1	0	0	1
1	1	0	0	0	1	1	0	1	0	0	0	1
1	1	1	0	0	0	1	0	1	0	0	1	1

3. Розглянемо пари наборів, які є сусідами для змінної x, із значеннями функцій на цих наборах:

$$f(0, 0, 0) = f(1, 0, 0) = 0;\ f(0, 0, 1) = f(1, 0, 1) = 1;$$
$$f(0, 1, 0) = f(1, 1, 0) = 1;\ f(0, 1, 1) = f(1, 1, 1) = 1.$$

Таким чином x – фіктивна змінна.

Розглянемо пари наборів по змінній y. Так як $f(0, 0, 0) \neq f(0, 1, 0)$, то y – істотна змінна.

Розглядаючи пари наборів по змінній z, знаходимо, що $f(0, 0, 0) \neq f(0, 0, 1)$, тому z – істотна змінна.

Отримали, що $f(x, y, z) = g(y, z)$, при цьому таблиця істинності функції $g(y, z)$ має вигляд (табл. 2):

Таблиця 2

y	0	0	1	1
z	0	1	0	1
$g(y, z)$	0	1	1	1

За таблицею 2 визначаємо $g(y, z) = y \vee z$, тобто $f(x, y, z) = y \vee z$.

Застосовуючи основні еквівалентності, перетворюємо формулу до вигляду, яка не містить фіктивної змінної:

$$f(x, y, z) = xy \vee \bar{y}z \vee \bar{x}y\bar{z} \vee yz \vee \overline{x \vee \bar{y} \vee z} = xy \vee \bar{y}z \vee \underline{\bar{x}y\bar{z}} \vee yz \vee \underline{\bar{x}y\bar{z}} = xy \vee \underline{\bar{y}z} \vee \bar{x}y\bar{z} \vee \underline{yz} =$$

$$= xy \vee z(\bar{y} \vee y) \vee \bar{x}y\bar{z} = xy \vee z \cdot 1 \vee \bar{x}y\bar{z} = xy \vee z \vee \underline{\bar{x}y\bar{z}} = xy \vee (z \vee \bar{z})(z \vee \bar{x}y) = xy \vee 1 \cdot (z \vee \bar{x}y) =$$

$$= \underline{xy} \vee z \vee \underline{\bar{x}y} = y(x \vee \bar{x}) \vee z = y \cdot 1 \vee z = y \vee z.$$

Таким чином функція $f(x, y, z) = y \vee z$, а логічна схема представлена на рис. 3, при цьому затримка схеми складає $T = \tau$ та ціна за Квайном – $S_Q = 2$.

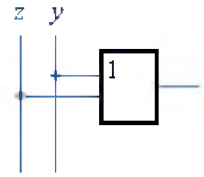
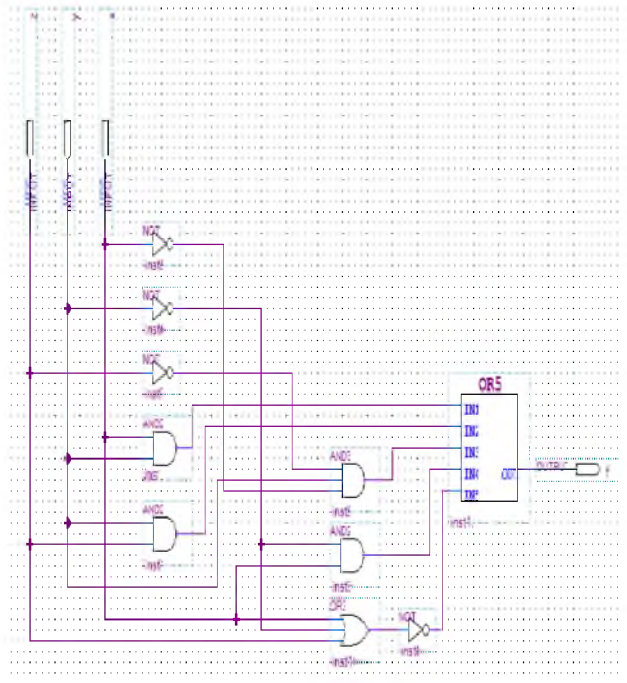


Рис. 3. Логічна схема, яка реалізує булеву функцію

$$f(x, y, z) = xy \vee \bar{y}z \vee \bar{x}y\bar{z} \vee yz \vee \overline{x \vee \bar{y} \vee z}$$

Проєктування комбінаційної схеми, яка реалізує булеву функцію роботизованої системи, здійснюється в середовищі САПР *Intel Quartus Prime* з вбудованим симулятором *ModelSim-Altera* та представлена у вигляді схемотехнічного редактора (рис. 4, *а*) та вихідного коду на мові опису апаратури *Verilog HDL* (рис. 4, *б*) [12–15].



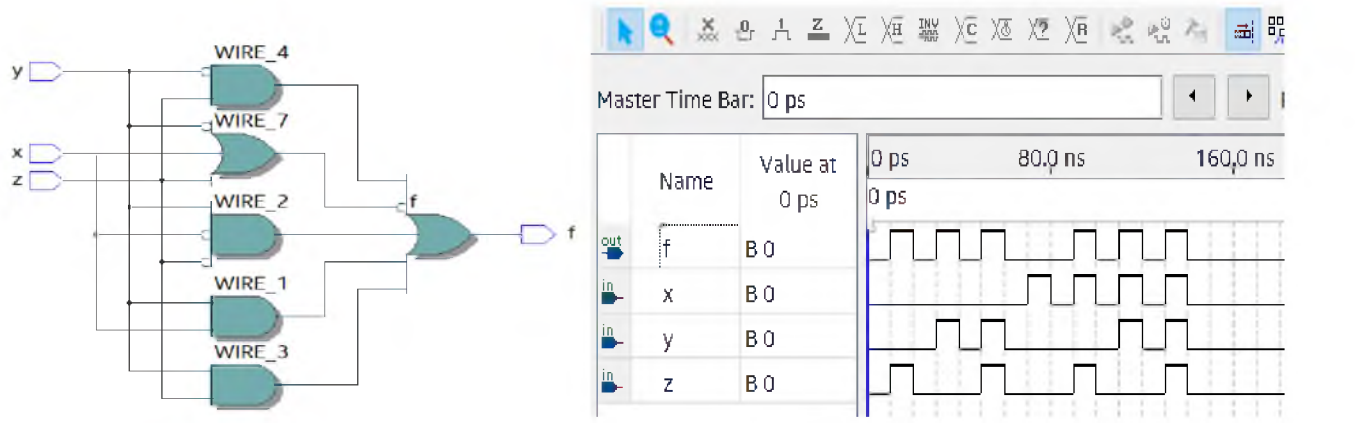

```
module logic_function(z, y, x, f);
input wire z; input wire y; input wire x;
output wire f;
wire WIRE_10;
wire WIRE_1; wire WIRE_2; wire
WIRE_3; wire WIRE_4;
wire WIRE_5; wire WIRE_7; wire
WIRE_8; wire WIRE_9;
assign  WIRE_1 = x & y;
assign  WIRE_3 = y & z;
assign  WIRE_7 = WIRE_10 | z | x;
assign  f = WIRE_1 | WIRE_2 | WIRE_3 |
WIRE_4 | WIRE_5;
assign  WIRE_4 = WIRE_10 & z;
assign  WIRE_9 =  ~x;
assign  WIRE_10 =  ~y;
assign  WIRE_8 =  ~z;
assign  WIRE_5 =  ~WIRE_7;
assign  WIRE_2 = WIRE_8 & y & WIRE_9;
endmodule
```

а) *б*)

Рис. 4. Комбінаційна схема по заданій логічній функції (*а*) та її *Verilog HDL*-код (*б*)

Далі представлено дизайн комбінаційної схеми, яка реалізує булеву функцію на рівні регістрових передач – *Register Type Level* (*RTL*) (рис. 5, *а*), що дає можливість в подальшому структурувати вихідний код та провести функціональне тестування, шляхом подачі на вхідні сигнали тестових комбінацій з подальшим аналізом вихідних сигналів у вигляді часових діаграм (рис. 5, *б*), які повною мірою відображають таблицю істинності.

Порівнюючи таблицю істинності (табл. 1) та результати моделювання (рис. 5, *б*), бачимо, що комбінаційна схема, яка проєктується, працює правильно.



а) *б*)

Рис. 5. *RTL* – дизайн функції $f(x, y, z) = xy \vee \bar{y}z \vee \bar{x}y\bar{z} \vee yz \vee \overline{x \vee \bar{y} \vee z}$ (*а*)
та функціональне моделювання по заданій логічній функції (*б*)

Впевнившись в коректності роботи комбінаційної схеми, переходимо до програмування та конфігурації ПЛІС шляхом використання конфігуруючого файлу, який згенерований асемблером *Intel Quartus Prime*.

В подальшому за кінцевий функціональний пристрій, який програмується, використовується налагоджена плата *DE10-Nano* на основі *FPGA Cyclone V* 5CSEBA6U2317.

Висновок. Таким чином, використаний в роботі підхід налагодження логічної схеми спрощує тестування моделі та робить дану методику прийнятною, а також доступною для більшості програмно-реконфігурованих цифрових пристроїв, які використовуються в роботизованих системах шляхом підтвердження адекватності отриманих результатів на часових діаграмах *ModelSim*. За рахунок використання засобів налагодження можливо запобігти ряду помилок, які виникають в процесі створення роботизованих системах, а також підтвердити або спростувати адекватність роботи представленої симульованої моделі, яка в подальшому програмується на кристалі ПЛІС.

ЛІТЕРАТУРА

1. Семенець В. В. Проектування цифрових систем з використанням мови VHDL: навч. посібник / В. В. Семенець, І. В. Хаханова, В. І. Хаханов. Харків: ХНУРЕ, 2003. 492 с.

2. Строгонов А. В. Проектирование комбинационных схем в базисе ПЛИС // Компоненты и технологи. 2008. № 5. С. 148–151.

3. Акчурин А. Д. Основы работы в среде Quartus II: уч.-метод. пособ. / А. Д. Акчурин, К. М. Юсупов, А. А. Колчев. Казань: КФУ, 2017. 49 с.

4. Тарасов И. Е. Проектирование конфигурируемых процессоров на базе ПЛИС // Компоненты и технологи. 2006. № 2. С. 78–83.

5. Попов А. Ю. Проектирование цифровых устройств с использованием ПЛИС: учеб. пособ. Москва: Изд-во МГТУ им. Н. Э. Баумана, 2009. 80 с.

6. Слюсарь В. В. Методика проектирования программно-реконфигурируемых устройств на базе ПЛИС / В. В. Слюсарь, Р. М. Романов // Оборонный комплекс – научно-техническому прогрессу России. 2010. № 1. С. 48–52.

7. Довгий П. С. Синтез комбинационных схем. Учебное пособие к курсовой работе по дисциплине «Дискретная математика» / П. С. Довгий, В. И. Поляков. Санкт-Петербург: СПбГУ ИТМО, 2009. 64 с.

8. Исмагилова Е. И. Булевы функции и построение логических схем: учеб. пособие / Е. И. Исмагилова. Москва: МИРЭА, 2015. 160 с.

9. Харрис Д. М. Цифровая схемотехника и архитектура компьютера: учеб. пособие. 2-е изд., перераб, и доп. USA: Morgan Kaufman, 2013. 1625 с.: ил.

10. Сергиенко И. В. Задачи дискретной оптимизации. Проблемы, методы решения, исследования / И. В. Сергиенко, В. П. Шило. Киев: Наукова думка, 2003. 261 с.

11. Тарасов И. Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования // Горячая линия – Телеком, 2022. 358 с.

12. Vaibbhav Taraate. PLD Based Designwith VHDL RTL Design, Synthesisand Implementation – Springer Nature Singapore Pte Ltd, 2017. 423 p.

13. Строгонов А. В. Реализация Verilog-проектов в базисе академических ПЛИС с применением САПР VTR7.0 // Компоненты и технологи. 2017. № 5. С. 12–17.

14. Bogdan Belean. Application-Specific Hardware Architecture Designwith VHDL. Springer International Publishing, 2018. 191 p.

15. Ратушний П. М. ПЛІС та їх програмування: лабораторний практикум / П. М. Ратушний, О. М. Жагловська, К. В. Огородник. Вінниця: ВНТУ, 2018. 57 с.