Затверджено

482.362. 705010201-79 33-5 ЛЗ

Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра комп’ютерних систем та мереж

**СИМУЛЯТОР ТРІЙКОВИХ КВАНТОВИХ МЕРЕЖ**

**482.362. 705010201-79 13 33-5**

**(Опис програми)**

Сторінок 13

2015

**АНОТАЦІЯ**

В даному документі описана програма, призначена для симуляції зворотніх трійкових мереж з можливістю переглядати результат складеної схеми. Розглянуто алгоритми роботи програми в цілому та представлено графічний вигляд екранних форм.

Опис програми містить: 7 розділів, 13 сторінок, 7 рисунків.

**ЗМІСТ**

1. ЗАГАЛЬНІВІДОМОСТІ………………………………………...………….….4

# 2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ..........................................…4

# 3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ.......................................................5

3.1. Структура і функції програми………………………………………..5

3.2. Алгоритм обчислювальної частини програми……..........…………..6

3.3. Діаграма прецедентів……………. …………………............………...8

3.4. Алгоритм виконання обчислень контрольованих елементів …....…9

4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ……………………………….13

5. ВИКЛИК І ЗАВАНТАЖЕННЯ………………………………………...…….13

# 6. ВХІДНІ ДАНІ………………………………………………………13

# 7. ВИХІДНІ ДАНІ…………………………………………………….13

# 1. ЗАГАЛЬНІ ВІДОМОСТІ

Метою розробленої програми є симуляція зворотних трійкових мереж із можливістю переглядати результат складеної схеми, створити бібліотеку примітивів, задавати вхідні сигнали в трійковій системі числення. Розробити зручний інтерфейс з візуалізацією вхідних та вихідних станів.

Дане програмне забезпечення розроблено на таких технологіях: HTML5, Css, Javascript з використанням бібліотеки JQuery.

# 2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програма призначена для симуляції роботи зворотних трійкових мережіз можливістю переглядати результат складеної схеми.

Програма забезпечує введення користувачем певних вхідних трійкових сигналів, які в процесі проходження квантової мережі змінюють своє значення. Результати обчислень можна переглянути в покроковому режимі, як дані будуть мінятися відносно вхідних в процесі обчислень. На виході ми отримуємо деяке вихідне значення цих сигналів.

Результати роботи розробленої програми можуть застосовуватись при дослідженні й розробленні квантових мереж, та квантового комп’ютингу.

# 3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

**3.1. Структура і функції програми**

Для створення квантової мережі використовувався компонент div з класом box. Ці бокси – це лінії нашої мережі, які представляються у вигляді таблички розміром 8 на n, де n – це «довжина» квантової мережі, яку ми можемо програмно збільшувати чи зменшувати за допомогою кнопок «Додати» і «Видалити». Отже в нас будується таблиця розмірністю 8 на n. При ініціалізації її комірки заповнюються лініями, які в сукупності утворюють одну суцільну лінію в кожному рядочку таблиці, таким чином у нас створюється «пуста» мережа. Далі одночасно зі ініціалізацією таблиці ініціалізується масив розмірністю 8 на n який одразу заповнюється нулями, таким чином що кожна комірка таблиці відповідає певному елементу в матриці.

Далі нам необхідно заповнити мережу примітивами. Для цього вибираємо потрібне місце на мережі і натискаємо лкм на цю комірку. Комірка, в свою чергу, підсвітиться.

Тільки після цього нам потрібно перетащити потрібний нам примітив на вибране місце. Одночасно з цим елемент в матриці, який відповідає за обрану комірку змінює своє значення на деяке інше ціле значення. В результаті створюється мережа.

Таким чином кожен елемент моє своє унікальне значення, або кілька унікальних значень. Алгоритм заповнення показаний на рис. 3.1.

Ініціалізація мережі;

Ініціалізація масиву;

Вибір комірки(лінії)

Вибір примітива

Додавання примітива в мережу

Рис. 3.1. Алгоритм заповнення мережі примітивами

**3.2. Алгоритм обчислювальної частини програми**

Кожен симулятор і взагалі кожну логічну схему можна уявити в вигляді таблиці де рядочки це розрядність, а стовпчики це елементи розміщені в певному порядку.

Для розміщення елементів на мережі доцільно використовувати табличку в якій стовпчики відповідають тактам, а рядочки лініям. Тоді кожна комірка має свою адресу і позицію в яку можна занести певний однокубітовий вентиль. Тоді як двокубітові вентилі будуть займати 2 комірки і т д.

Вхідні дані представленні у вигляді вектора довжиною 8 значень (0 або 1 або 2).

Починається сканування мережі на наявність в ній контрольованих елементів. Цикл розміром n який представляє довжину мережі і вміщує в себе інший цикл, який в свою чергу представляє комірки в одному стовпчику. В результаті подвійний цикл який сканує мережу, спочатку по комірках стовпчика, - потім по рядках. Якщо в одному такті зустрічається елемент, або контрольований спін, викликається відповідна функція, яка відповідає примітиву. І ця функція записує в матрицю відповідне примітиву значення(0,1,2), після чого всі елементи матриці заносяться в функцію, яка додає всі елементи матриці по модулю три. Результат, який виходить після додавання елементів матриці, записується в відповідний рядку матриці span з классом "spans\_for\_result". Для кожного рядка матриці є свій span для результатів з відповідним класом. Наприклад, для першого рядка матриці, після обчислень результат буде записуватися в «<span id ="line\_0" class = "spans\_for\_result"></span>».

Алгоритм обчислювальної частини програми представлений на рис. 3.2.

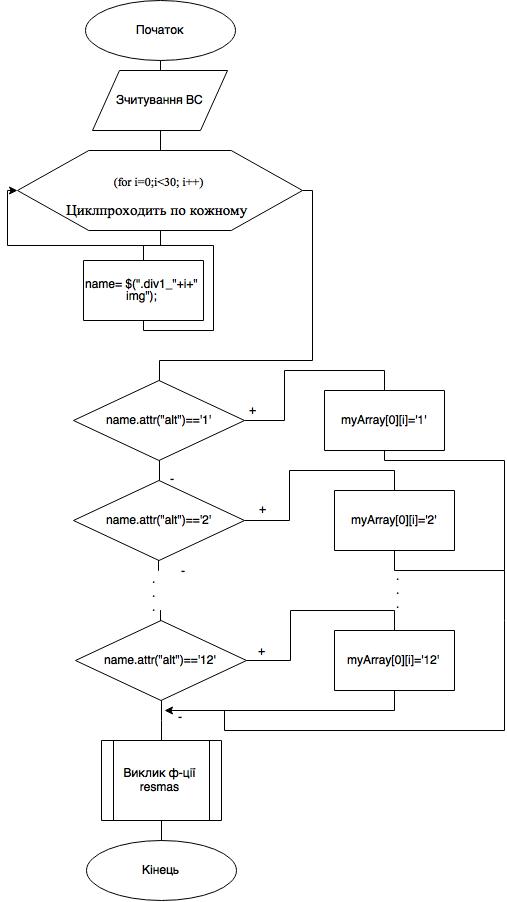


Рис. 3.2.Алгоритм виконання обчислень програми

**3.3. Діаграма прецедентів**

Діаграма прецедентів є [графом](http://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами.[[1]](http://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BF%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2#cite_note-rumbaught-1) Діаграми прецедентів відображають елементи моделі варіантів використання.Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання.

З точки зору моделі прецедентів структура програми виглядає наступним чином. Користувачеві надається можливість виконувати основні дії, передбачені в програмі, а саме:

1. Задавати вхідні кутріти.
2. Формувати мережу.
   * + Вибирати позицію на лінії.
     + Вибирати необхідний примітив із переліку на формі.
       - +1.
       - +2.
       - 01.
       - 02.
       - 12.
       - Контрольовані елементи.
3. Переглядати загальні результати.
4. Змінювати розмір мережі.
5. Видаляти обраний елемент.
6. Очищувати мережу.

Діаграма прецедентів програми наступна (рис. 3.3).

Примітиви

Рис. 3.3. Діаграма прецедентів

**3.4. Алгоритм виконання обчислень контрольованих елементів**

Сканується мережа на наявність контрольованих елементів. Для цього ми виконуємо два цикли : один перевіряє рядки, інший – стовпці. Якщо на одному стовпці цикл знаходить примітив і на цьому ж стовпці знаходить вентель, то виконується відповідна функція, яка обчислює цей контрольований елемент і також будує його.

Алгоритм виконання обчислень контрольованих елементів показаний на рис. 3.3.

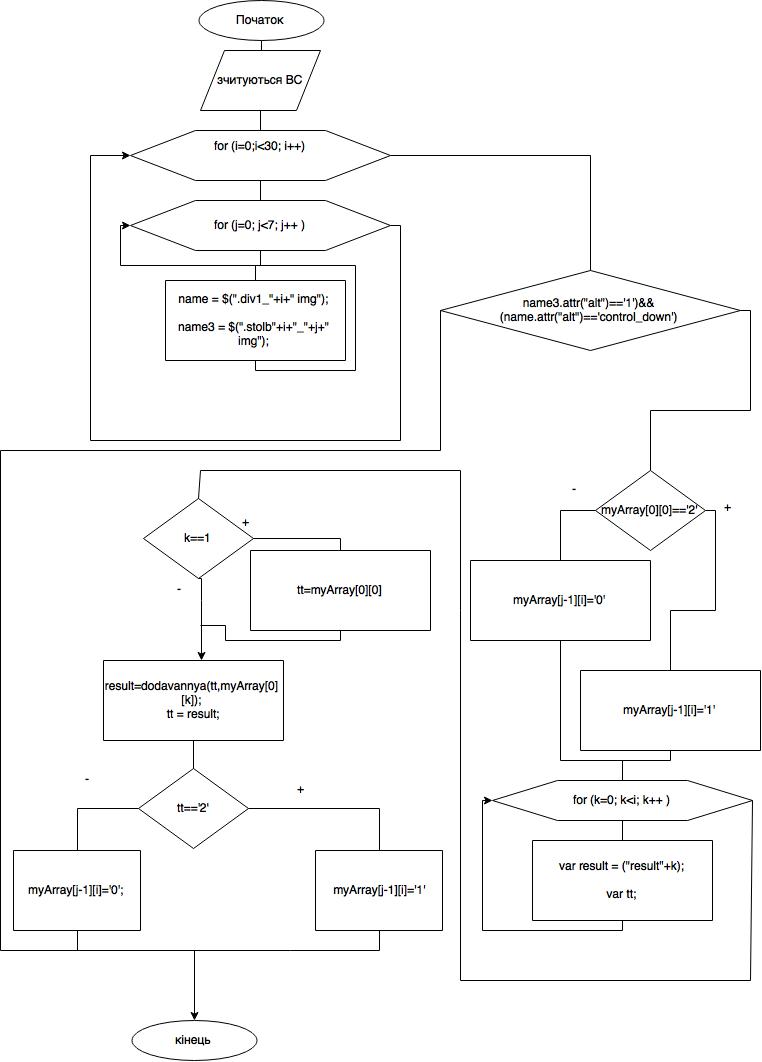


Рис 3.4. Алгоритм виконання обчислень контрольованих елементів

Початкова форма програми зображена на рис. 3.5.

Вхідними даними для програми є значення 0,1 або 2. (Рис. 3.6.), які задає користувач.

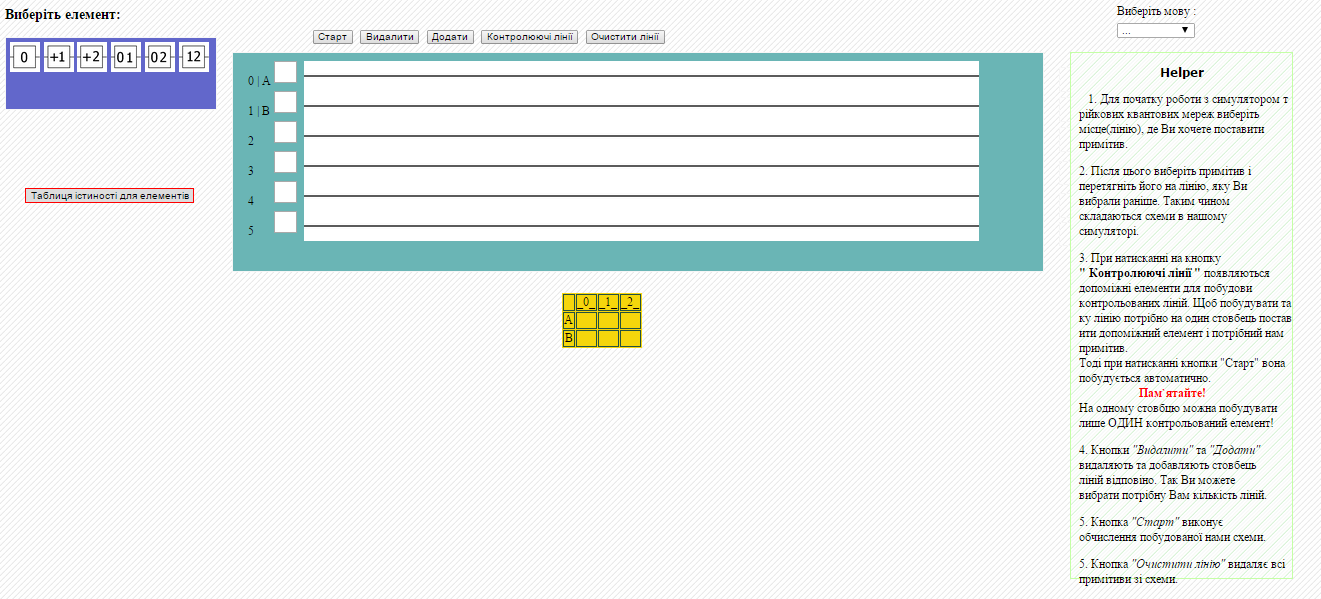


Рис 3.5. Початкова форма програми

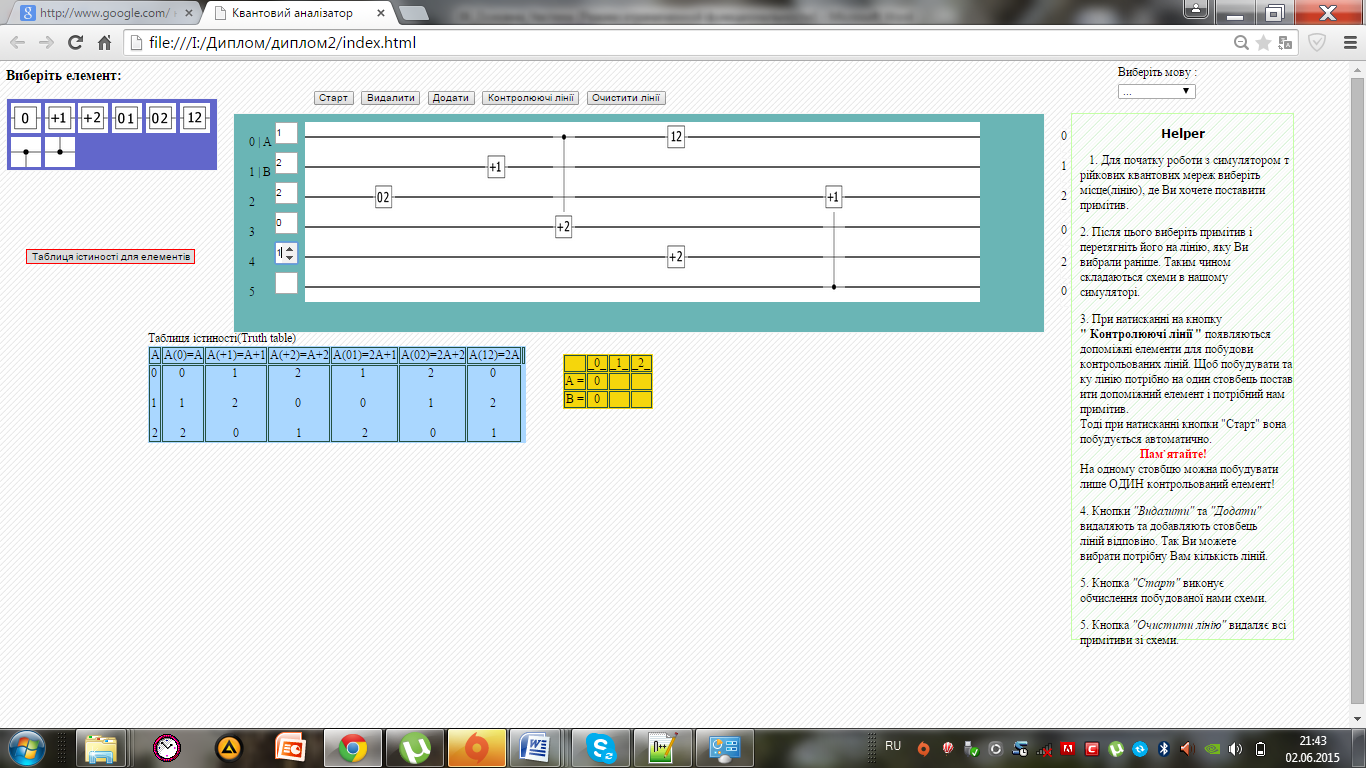


Рис 3.6. Задання початкових значень

В програмі передбачено дві мови: англійська та українська. Користувач може на власне бажання змінити мову в програмі (рис.3.7.)

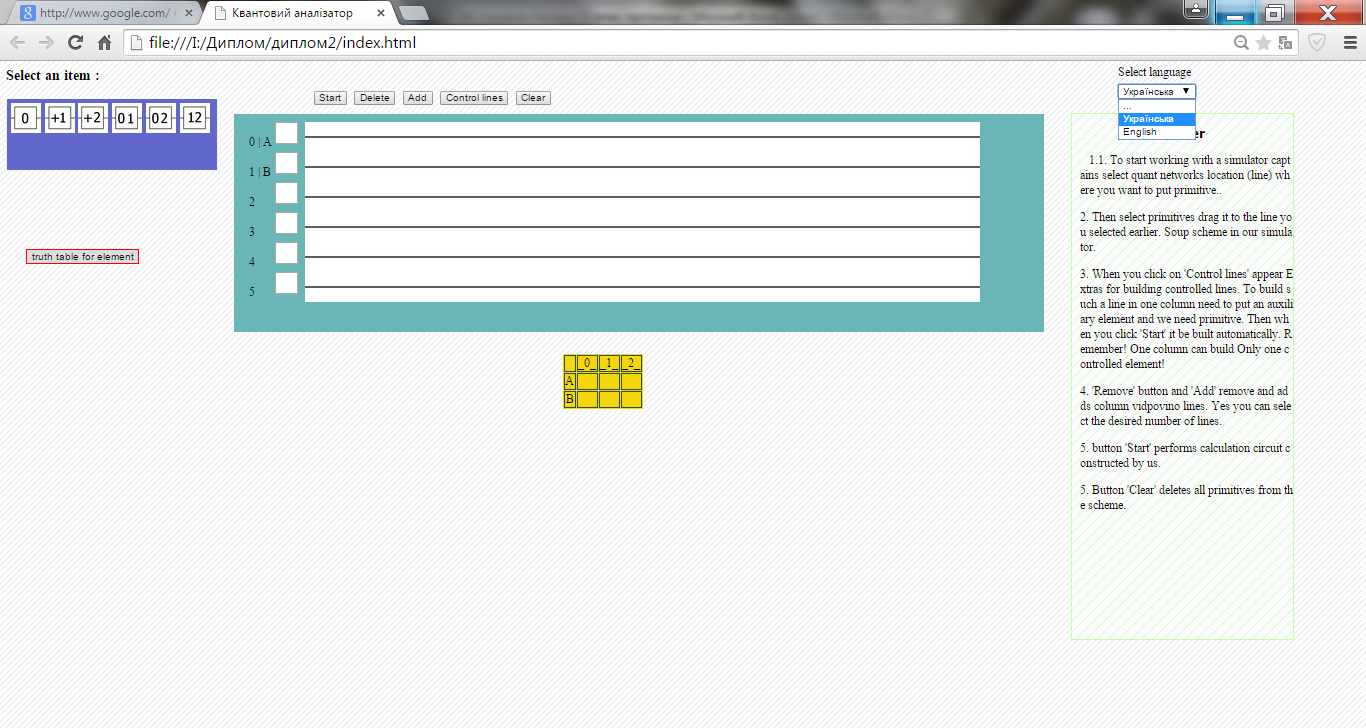


Рис 3.7. Зміна мови програми

Також є таблиця істинності для кожного примітива(рис 3.8.)

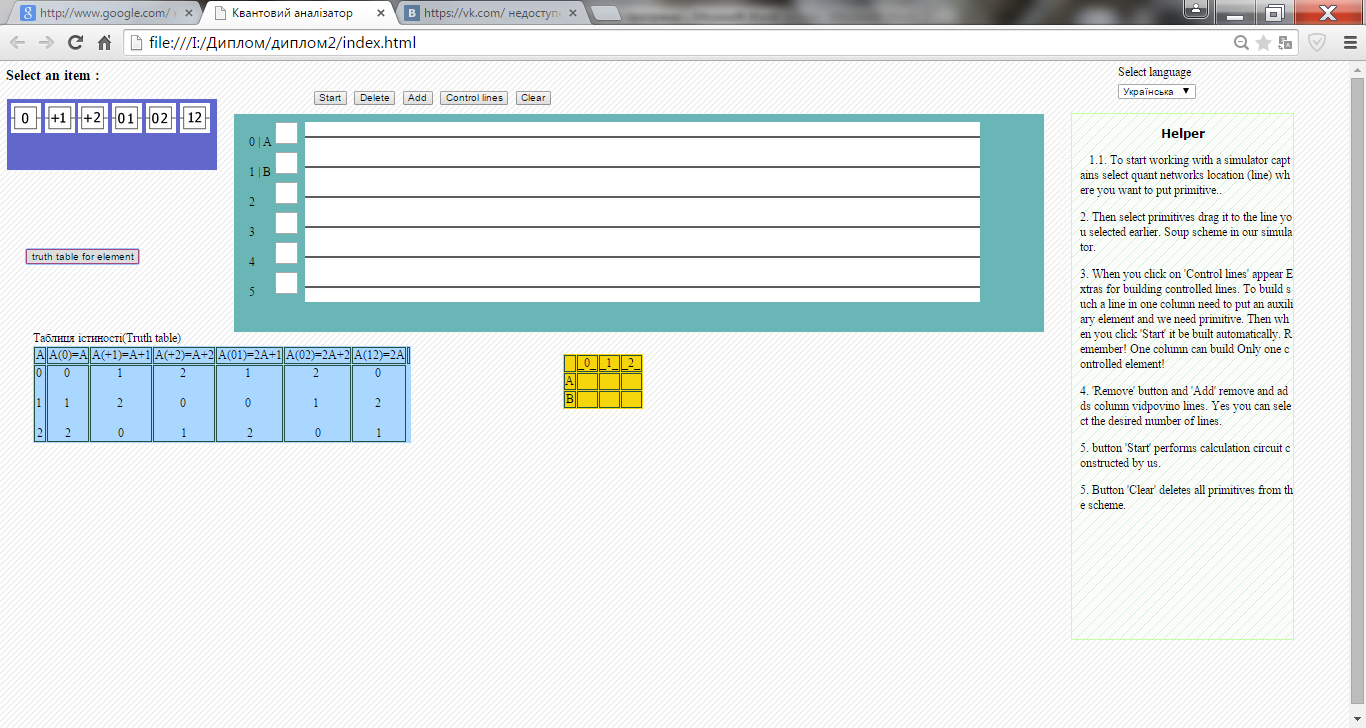


Рис 3.8. Таблиця істинності для елементів

# 4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для розробки програмного продукту було використано комп’ютер наступної конфігурації:

• процесор - AMD Athlon II X4 760K 3.8GHz;

• материнська плата: ASRock Fatal1ty FM2A88X;

• оперативна пам’ять: Kingston DDR3-1600 8192 MB;

• жорсткий диск: Toshiba 2TB 7200rpm 64MB 3.5 SATA III

• графічний адаптер: AMD Radeon HD 6850, 1 Гб;

• монітор: LG 22MP55 (1920x1080);

• блок живлення: CoolerMaster Silent Pro M2 520W;

• швидкість інтернет-з’єднання: 10 Мбіт/с.

# 5. ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма може бути викликана стандартними методами, прийнятими в операційній системі Microsoft Windows, а саме: подвійним натисканням лівої кнопки маніпулятора «миша» в момент перебування його вказівника на значкові програми або ярлика до неї; натисканням на клавіатурі клавіші «Enter» або «Return» після активування значка або ярлика (підсвічування синім кольором). Для цієї мети доцільно використовувати можливості багаточисленних файлових менеджерів (напр., Far Manager або Total Commander).

# 6. ВХІДНІ ДАНІ

Вхідними (початковими) даними програми є трійкові вхідні сигнали. (рис. 3.6).

# 7. ВИХІДНІ ДАНІ

Вихідними даними програми є сигнали, які змінювались при проходженні по лініям створеної мережі, у трійковій формі.