МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра ЕОМ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторної роботи №3

з дисципліни

" МОДЕЛЮВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ "

для студентів першого (бакалаврського) рівня вищої освіти спеціальності

123 "**Комп'ютерна інженерія**"

Укладачі:

Юрчук А. Ф. асистент каф. ЕОМ

Цигилик Л.О., ст. викл. каф.ЕОМ

Львів – 2020

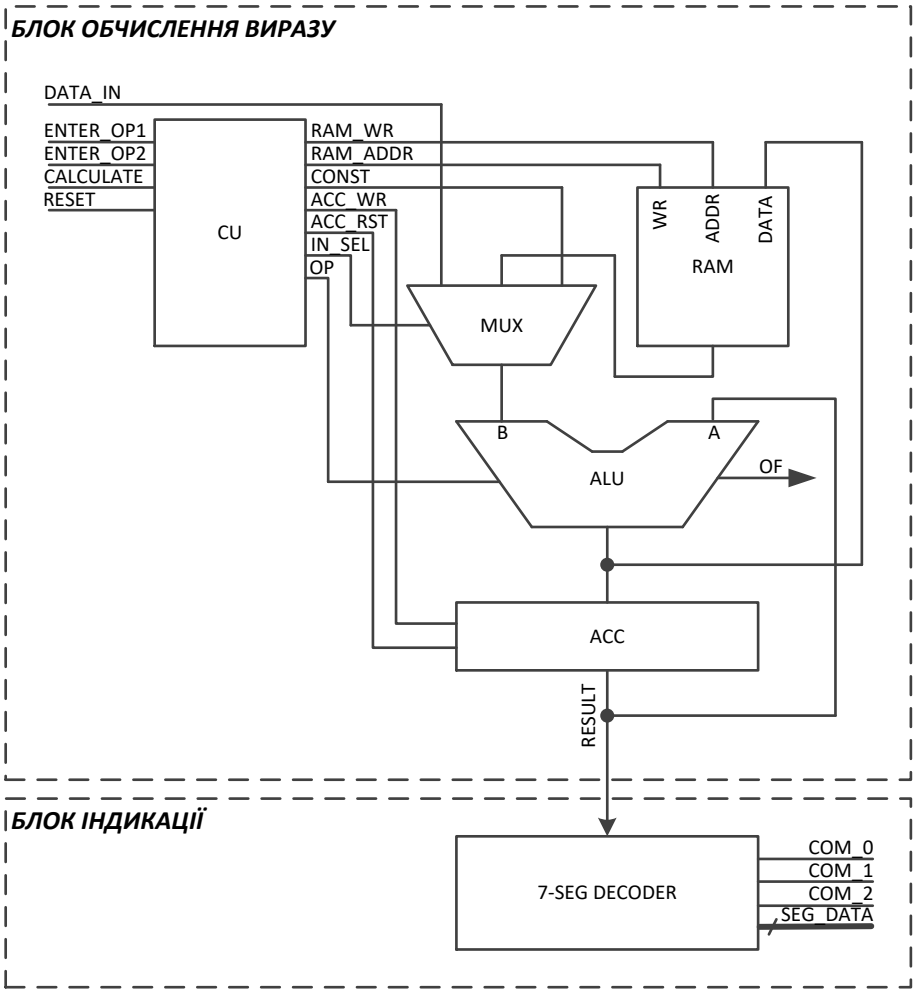
Тема роботи:

Поведінковий опис цифрового автомата. Перевірка роботи автомата за допомогою стенда *Elbert V2 – Spartan 3A FPGA.*

Мета роботи:

На базі стенда *Elbert V2 – Spartan 3A FPGA,* реалізувати цифровий автомат для обчислення значення виразу дотримуючись наступних вимог:

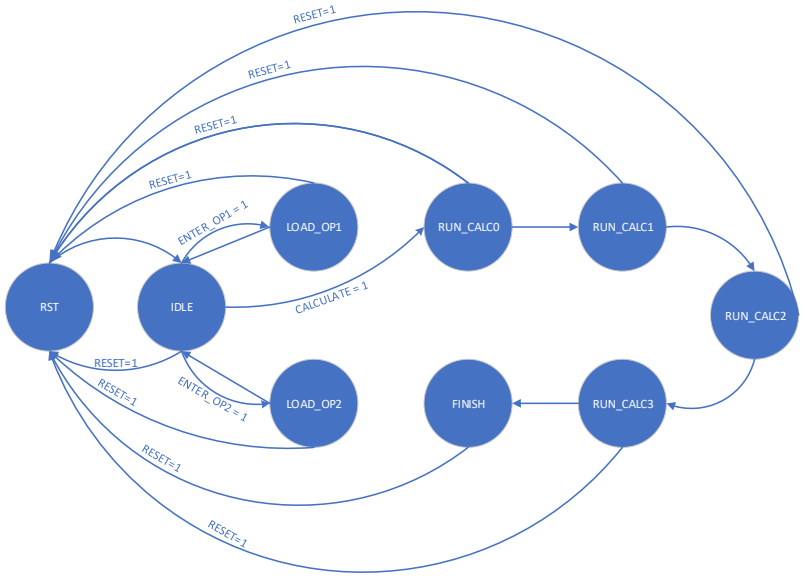
1. Функціонал пристрою повинен бути реалізований згідно отриманого варіанту завдання. Дивись розділ ЗАВДАННЯ:.
2. Пристрій повинен бути ітераційним (АЛП *(ALU)* повинен виконувати за один такт одну операцію), та реалізованим згідно наступної структурної схеми (*Малюнок 1*):



Малюнок 1 - Структурна схема автомата.

Опис елементів структурної схеми:

- *CU*: блок керування. Призначений для керування роботою решти елементів (*MUX*, *RAM*, *ALU*, *ACC*) блока обчислення виразу. Приклад графа станів блока керування для реалізації варіанту №0 (*Таблиця 1*) наведено нижче (*Малюнок 2*):



Малюнок 2 - Приклад графа станів блока керування.

  - *RST* – скидання значення регістра *ACC*.
  - *IDLE* – очікування команд *ENTER_OP1*, *ENTER_OP2* або *CALCULATE*.
  - *LOAD_OP1* – запис операнда *OP1* в *RAM(0x00)*;
  - *LOAD_OP2* – запис операнда *OP2* в *RAM(0x01)*;
  - *RUN_CALC0* – запис операнда *OP1* в регістр *ACC*:
    $ACC = RAM(0x00);$
  - *RUN_CALC1* – знаходження різниці *(OP1 – OP2)*. Значення *OP1* береться з регістра *ACC*. Значення *OP2* береться з *RAM(0x01)*. Результат операції зберігається в *ACC*:
    $ACC = ACC - RAM(0x01)$
  - *RUN_CALC2* - Додавання константи «*4*» до *(OP1 – OP2)*. Значення різниці *(OP1 – OP2)* береться з *ACC*. Результат операції зберігається в *ACC*:
    $ACC = ACC + CONST$
  - *RUN_CALC3* – операція зсуву ліворуч на *OP2* розрядів значення *((OP1 – OP2) + 4)*. Значення *((OP1 – OP2) + 4)* береться з *ACC*. Значення *OP2* береться з *RAM(0x01)*. Результат операції зберігається в *ACC*:
    $ACC = ACC << RAM(0x01)$
  - *FINISH* – індикація кінцевого результату обчислень.

- *MUX*: мультиплексор. Призначений для комутації входу *B* арифметико-логічного пристрою з різними джерелами вхідних даних.
- *RAM*: пам'ять пристрою. Призначена для зберігання вхідних операндів, а також – проміжних результатів.
- *ALU*: арифметико-логічний пристрій. Призначений для виконання арифметичних і логічних операцій над вхідними операндами *A* і *B*.
- *ACC*: акумулятор. Регістр, призначений для зберігання безпосередніх результатів виконання арифметичних і логічних операцій, а також – кінцевого результату обчислень.
- *7-SEG_DECODER*: блок індикації. Призначений для перетворення двійкового позиційного коду *(BIN)* в двійково-десятковий код *(BCD)* і подальшого відображення за допомогою 7-сегментних індикаторів.

Опис входів:

- *DATA_IN*: шина вхідних даних.
- *ENTER_OP1*: запис першого операнда в пам'ять даних *(RAM)* автомата.
- *ENTER_OP2*: запис другого операнда в пам'ять даних *(RAM)* автомата.
- *CALCULATE*: запуск процесу обчислення.
- *RESET*: скидання автомата в початковий стан та скидання регістра *ACC* в *«0»*.
- *CLK*: сигнал синхронізації.

Опис виходів:

- *OF*: ознака переповнення в АЛП *(ALU)*.
- *COM_0*: активація 7-сегментного індикатора *№0* (одиниці).
- *COM_1*: активація 7-сегментного індикатора *№1* (десятки).
- *COM_2*: активація 7-сегментного індикатора *№2* (сотні).
- *SEG_DATA*: керування сегментами *A*, *B*, *C*, *D*, *E*, *F*, *G* та *DP*, активного індикатора.

Опис виходів блока управління автомата (*CU*):

- *RAM_WR*: Сигнал запису пам'яті даних *(RAM)* автомата.
- *RAM_ADDR*: адреса для запису/читання пам'яті даних *(RAM)* автомата.
- *CONST*: константний параметр виразу.
- *ACC_WR*: сигнал запису акумулятора.
- *ACC_RST*: сигнал скидання значення акумулятора в *«0»*.
- *IN_SEL*: сигнал вибору входу мультиплексора *MUX*.
- *OP*: код операції АЛП *(ALU)*.

3. Кожен блок структурної схеми повинен бути реалізований на мові *VHDL* в окремому файлі. Дозволено використовувати всі оператори.
4. Для кожного блока структурної схеми повинен бути згенерований *Schematic* символ.
5. Інтеграція структурних блоків в єдину систему та зі стендом *Elbert V2 – Spartan 3A FPGA* повинна бути виконана за допомогою *ISE WebPACK™ Schematic Capture.*
6. Кожен структурний блок і схема вцілому повинні бути промодельовані за допомогою симулятора *ISim.*
7. Формування вхідних даних на шині *DATA_IN* повинно бути реалізовано за допомогою *DIP* перемикачів (елемент *P7* на стенді *Elbert V2 – Spartan 3A FPGA* Див. *Додаток – 2* (інформація про стенд)):
    a. *P7[8]* – наймолодший розряд значення операнда.
    b. *P7[1]* – найстарший розряд значення операнда.
8. Керування пристроєм повинно бути реалізовано за допомогою *PUSH BUTTON* кнопок (елементи *SW1*, *SW2*, *SW3*, *SW6* на стенді *Elbert V2 – Spartan 3A FPGA* Див. *Додаток – 2* (інформація про стенд)):
    a. *SW1* – запис першого операнда в пам'ять даних автомата.
    b. *SW2* – запис другого операнда в пам'ять даних автомата.
    c. *SW3* – запуск процесу обчислення.
    d. *SW6* – скидання автомата у початковий стан.
9. Індикація значень операндів при вводі, та вивід результату обчислень повинні бути реалізовані за допомогою семи сегментних індикаторів *S1-S3*. Індикація переповнення в АЛП - за допомогою *LED D8* на стенді *Elbert V2 – Spartan 3A FPGA*. Див. *Додаток – 2* (інформація про стенд)
10. Підготувати і захистити звіт.

ЗАВДАННЯ:

ВАРІАНТ	ВИРАЗ
0	((OP1 – OP2) + 4) << OP2
1	((OP1 + 2) * OP2) << OP1
2	((OP1 or OP2) + OP2) – 3
3	((OP2 and 5) + OP2) – OP1
4	((4 + OP1) xor OP2) – OP1
5	((1 << OP1) + OP2) – OP1
6	((OP1 + OP2) – 2) << OP2
7	((OP1 << 2) – OP2) + 4
8	((OP1 * OP2) >> 1) + OP1
9	((OP2 - 4) + OP1) or 2
10	((OP2 – OP1) << 1) + OP1
11	((OP1 * 2) + OP2) >> 1
12	((OP1 xor OP2) + OP2) – 1

Таблиця 1 – Варіанти завдань.

МЕТОДИЧНІ ВКАЗІВКИ:

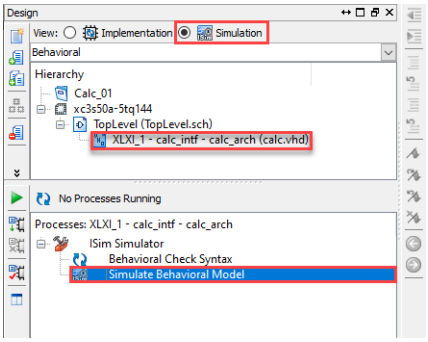
Даний розділ надає рекомендовану послідовність дій для вивчення роботи прикладу що додається до даного документа, та розробки власного пристрою згідно індивідуального варіанту завдання (*Таблиця 1*).

Приклад, що додається до даного документа, реалізує обчислення виразу Варіант №0 (*Таблиця 1*) з наступними обмеженнями:

- Усі блоки пристрою реалізовані в одному **.vhd* файлі і мають спільний символ.
- Не реалізовано індикацію переповнення в АЛП *(ALU)*.

Рекомендована послідовність виконання роботи

1. Ознайомитись з будовою та принципом роботи семи сегментного індикаторного модуля (*Додаток – 1* (7-сегментний LED індикатор)).
2. Відкрити, та промоделювати роботу прикладу, який додається до даного документа:
   - Запустити симулятор для файлу *calc.vhd*

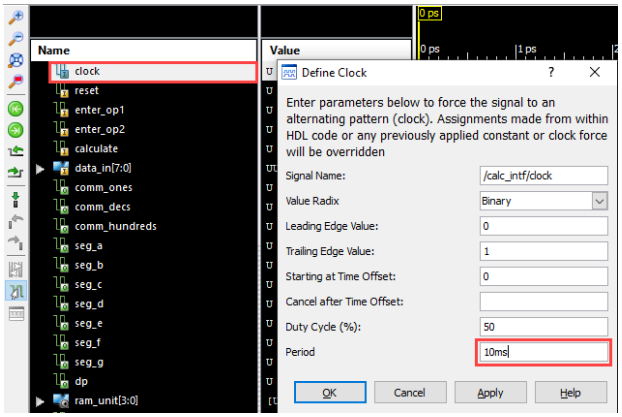


Малюнок 3 – Вікно менеджера проекту.

   - На панелі команд симулятора встановити крок *10ms* та натиснути *Restart* для скидання в початковий стан

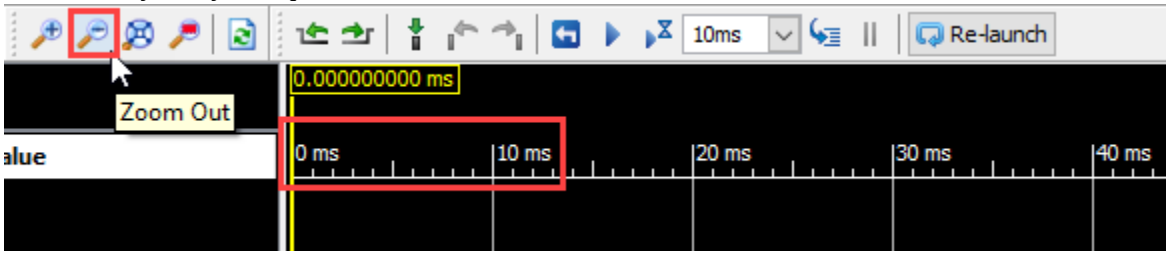
Малюнок 4 - Панель інструментів симулятора.

   - Сконфігурувати стимулятор входу *clock* для генерації тактового сигналу з періодом *10ms*:



Малюнок 5– вікно налаштувань стимулятора входу clock.

- Виконавши декілька разів команду *Zoom-Out*,встановити значення *10ms* для кроку шкали часу симулятора

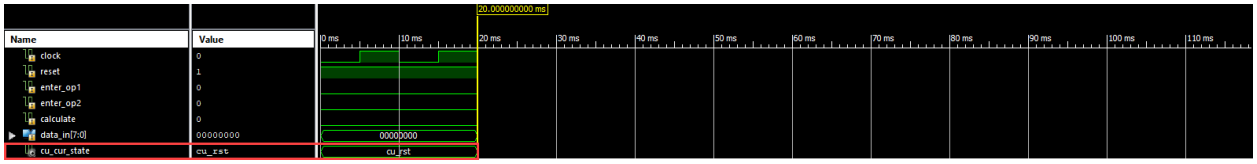


Малюнок 6 – Значення кроку шкали часу симулятора.

- Виконати 2 такти симуляції з наступними значеннями вхідних сигналів:

СИГНАЛ	ЗНАЧЕННЯ
reset	1
enter_op1	0
enter_op2	0
calculate	0
data_in	0

Керуючий автомат (*CU*) блока обчислення виразу буде переведено в стан *RST* для скидання значення регістра ACC в «0».



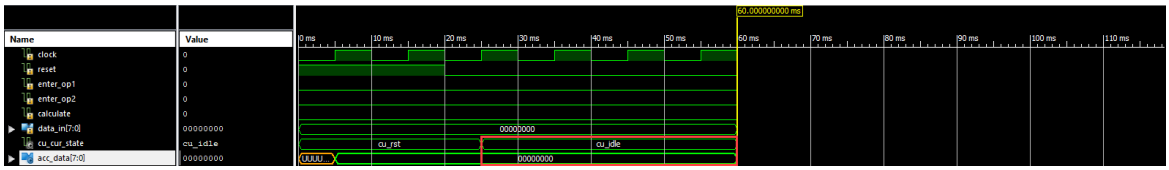
Малюнок 7 – Значення сигналів, коли автомат знаходиться в стані RST.

- Виконати 4 такти симуляції з наступними значеннями вхідних сигналів:

СИГНАЛ	ЗНАЧЕННЯ
reset	0
enter_op1	0
enter_op2	0
calculate	0

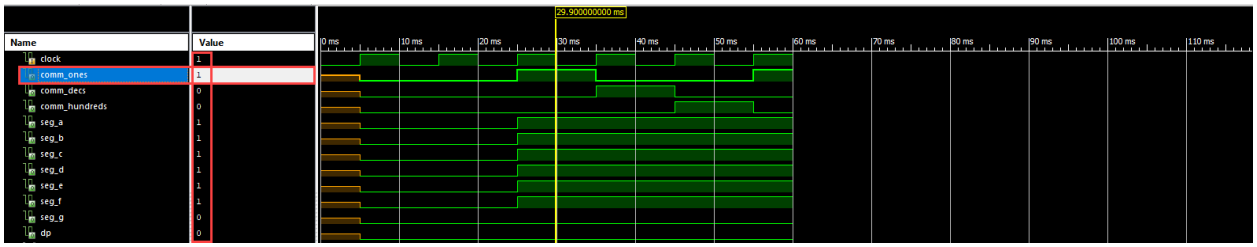
СИГНАЛ	ЗНАЧЕННЯ
data_in	0

Керуючий автомат *(CU)* блока обчислення виразу буде переведено в стан *IDLE* і очікуватиме команд (ввід параметрів, запуск виконання обчислень). Регістр *ACC* міститиме значення *«0»*.



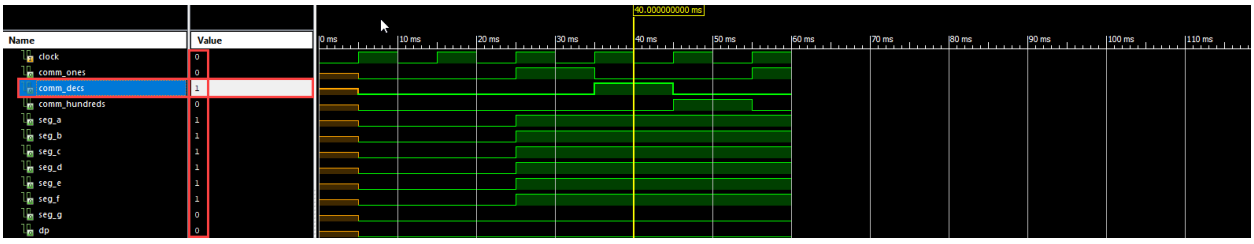
Малюнок 8 – Значення сигналів, коли автомат знаходиться в стані IDLE.

Семи сегментні індикаторні модулі відображають поточне значення регістра *ACC (*Додаток – 1 (7-сегментний LED індикатор)) відображає стани сигналів в момент індикації розряду одиниць. В стані логічної *«1»* перебувають: сигнал керування виводом *COM* розряду одиниць (розряд одиниць є активним), та сигнали керування елементами індикації *A*, *B*, *C*, *D*, *E* і *F* (цифра *«0»*).



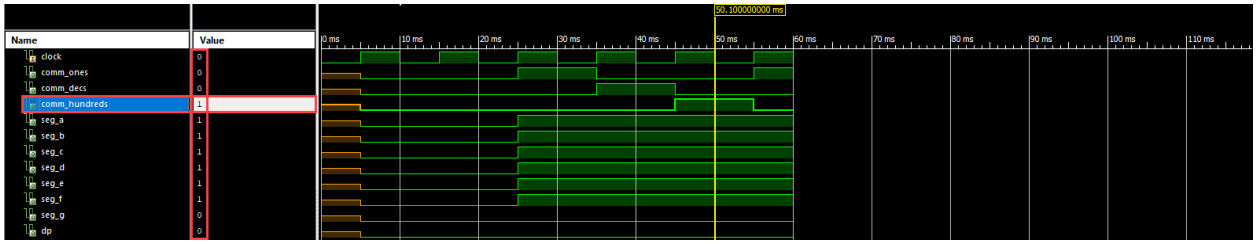
Малюнок 9 - Індикація розряду одиниць (цифра «0»).

Малюнок 10 відображає стани сигналів в момент індикації розряду десятків. В стані логічної *«1»* перебувають: сигнал керування виводом *COM (comm_decs)* розряду десятків (розряд десятків є активним), та сигнали керування елементами індикації *A*, *B*, *C*, *D*, *E* і *F* (цифра *«0»*).



Малюнок 10 – Індикація розряду десятків (цифра «0»).

Малюнок 11 відображає стани сигналів в момент індикації розряду сотень. В стані логічної *«1»* перебувають: сигнал керування виводом *COM (comm_hundreds)* розряду сотень (розряд сотень є активним), та сигнали керування елементами індикації *A*, *B*, *C*, *D*, *E* і *F* (цифра *«0»*).



Малюнок 11 – Індикація розряду сотень (цифра «0»).

- Виконати 2 такти симуляції з наступними значеннями вхідних сигналів:

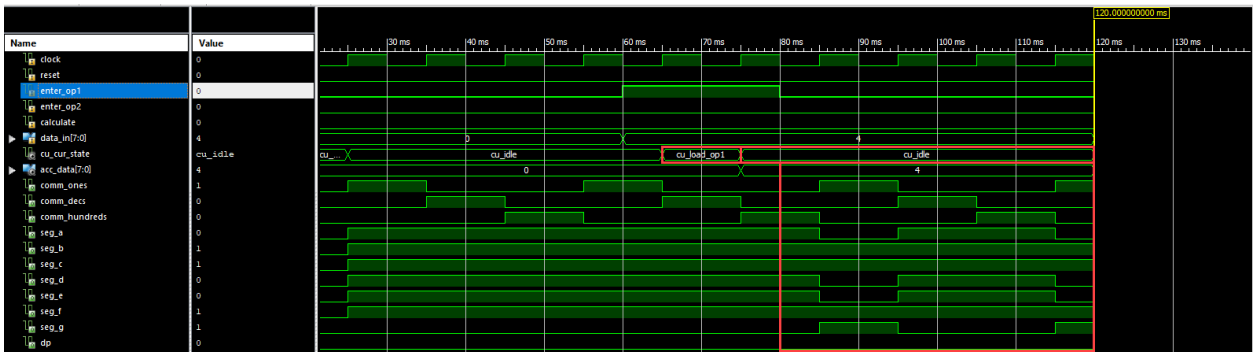
СИГНАЛ	ЗНАЧЕННЯ
reset	0
enter_op1	1
enter_op2	0
calculate	0
data_in	4 (OP1 = 4)

За перший такт керуючий автомат *(CU)* блока обчислення виразу буде переведено в стан *LOAD_OP1*, для запису значення операнда *OP1* в *RAM* автомата і в регістр *ACC*. За другий такт керуючий автомат *(CU)* блока обчислення виразу буде переведено назад в стан *IDLE* для очікування нових команд. Регістр *ACC* міститиме значення «*4*».

- Виконати 4 такти симуляції з наступними значеннями вхідних сигналів:

СИГНАЛ	ЗНАЧЕННЯ
reset	0
enter_op1	0
enter_op2	0
calculate	0
data_in	4 (OP1 = 4)

Керуючий автомат *(CU)* блока обчислення виразу буде знаходитись в стані *IDLE*. Семи сегментні індикатори відображатимуть поточне значення регістра *ACC* (*OP1* = 4). Даний крок необхідний для оновлення значення індикаторів на часовій діаграмі (*Малюнок 12*).



Малюнок 12 – Результат виконання кроків 2.7 і 2.8.

- Виконати 2 такти симуляції з наступними значеннями вхідних сигналів:

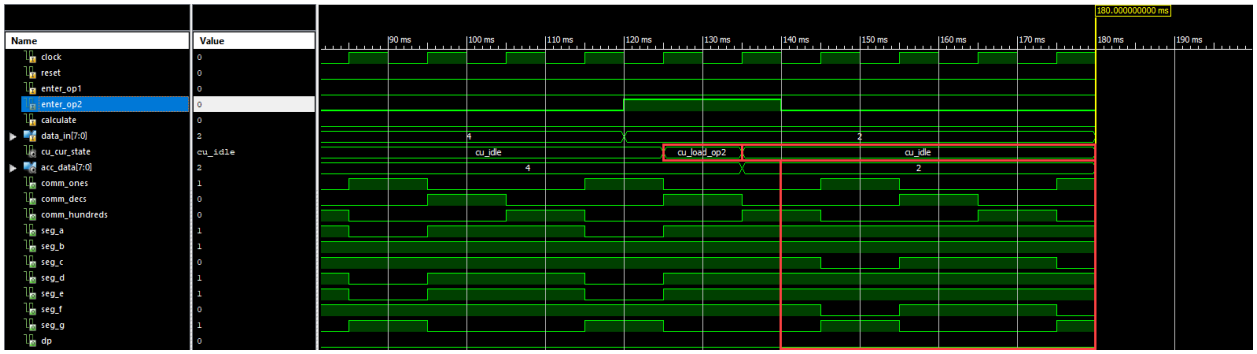
СИГНАЛ	ЗНАЧЕННЯ
reset	0
enter_op1	0
enter_op2	1
calculate	0
data_in	2 (*OP2* = 2)

За перший такт керуючий автомат *(CU)* блока обчислення виразу буде переведено в стан *LOAD_OP2*, для запису операнда *OP2* в *RAM* автомата і в регістр *ACC*. За другий такт керуючий автомат *(CU)* блока обчислення виразу буде переведено назад в стан *IDLE* для очікування нових команд. Регістр *ACC* міститиме значення «*2*».

- Виконати 4 такти симуляції з наступними значеннями вхідних сигналів:

СИГНАЛ	ЗНАЧЕННЯ
reset	0
enter_op1	0
enter_op2	0
calculate	0
data_in	2 (*OP2* = 2)

Керуючий автомат *(CU)* блока обчислення виразу буде знаходитись в стані *IDLE*. Семи сегментні індикатори відображатимуть поточне значення регістра *ACC* (*OP2* = 2). Даний крок необхідний для оновлення значення індикаторів на часовій діаграмі (*Малюнок 13*).

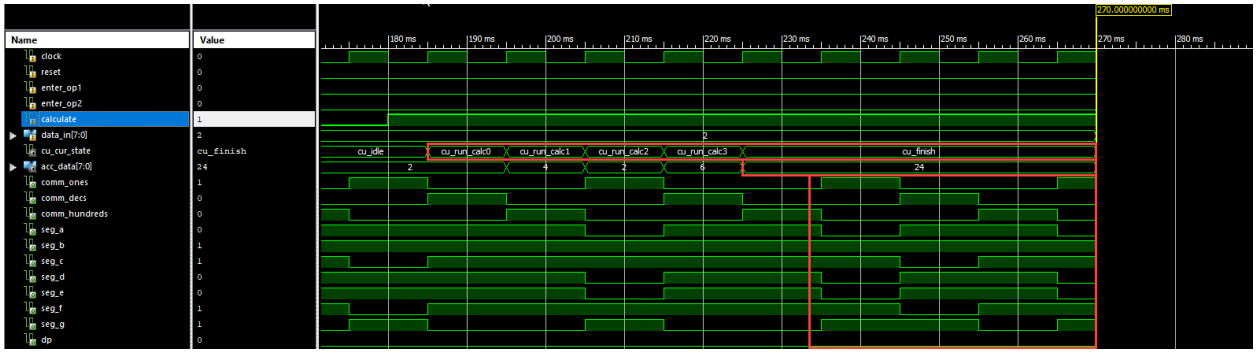


Малюнок 13 – Результат виконання кроків 2.9 і 2.10.

- Виконати 9 тактів симуляції з наступними значеннями вхідних сигналів:

СИГНАЛ	ЗНАЧЕННЯ
reset	0
enter_op1	0
enter_op2	0
calculate	1
data_in	2 (*OP2* = 2)

Керуючий автомат *(CU)* блока обчислення виразу перейде через стани *RUN_CALC0*, *RUN_CALC1*, *RUN_CALC2* та *RUN_CALC3* до стану *FINISH* (кінець обчислення). Регістр *ACC* міститиме результат обчислень (число «*24*»). Семи сегментні індикатори відображатимуть поточне значення регістра *ACC* (число «*24*» - результат обчислень).



Малюнок 14 – Результат виконання кроку 2.11.

3. Створити новий **.vhd* файл, та реалізувати в ньому мультиплексор *MUX* (*Малюнок 1*).
4. Перевірити роботу мультиплексора за допомогою симулятора *ISim*.
5. Створити новий **.vhd* файл, та реалізувати в ньому регістр *ACC* (*Малюнок 1*).
6. Перевірити роботу регістра *ACC* (запис/скидання) за допомогою симулятора *ISim*.
7. Визначити набір операцій, необхідних для обчислення індивідуального варіанту виразу. Наприклад, для реалізації варіанту №0 (*Таблиця 1*) потрібна підтримка наступних операцій: «-», «+», «<<», *nop*. Операція *nop* просто передає дані з входу *B* арифметико-логічного пристрою на його вихід. Використовується, якщо потрібно записати дані в регістр *ACC* без жодних змін.
8. Створити новий **.vhd* файл, та реалізувати в ньому АЛП *(ALU)* (*Малюнок 1*) з підтримкою визначеного набору операцій.
9. Перевірити роботу АЛП *(ALU)* за допомогою симулятора *ISim*.
10. Визначити множину станів та умови переходів пристрою керування *(CU)*, необхідних для обчислення виразу.
11. Створити новий **.vhd* файл, та реалізувати в ньому пристрій керування *(CU)* (*Малюнок 1*), відповідно до визначеного алгоритму.
12. Перевірити роботу пристрою керування *(CU)* за допомогою симулятора *ISim*.
13. Створити новий **.vhd* файл, та реалізувати в ньому пам'ять пристрою *(RAM)* (*Малюнок 1*).
14. Перевірити виконання операцій читання/запису за різними адресами пристрою *(RAM)* за допомогою симулятора *ISim*.
15. Створити новий **.vhd* файл, та реалізувати в ньому блок індикації. *(7-SEG DECODER)* (*Малюнок 1*). Для реалізації блока індикації можна скористатися фрагментами коду з прикладу, що додається до даного документа.
16. Перевірити роботу блока індикації *(7-SEG DECODER)* за допомогою симулятора *ISim*.

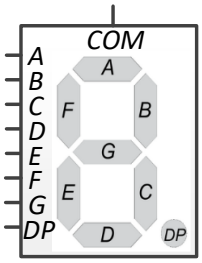
17. Згенерувати символи для імплементованих компонентів (*MUX*, *ACC*, *ALU*, *CU*, *RAM*, *7-SEG DECODER*).
18. Створити *Schematic* файл верхнього рівня (файл *TopLevel.sch)* та виконати інтеграцію компонентів системи між собою та зі стендом *Elbert V2 – Spartan 3A FPGA*.
19. Перевірити роботу пристрою вцілому за допомогою симулятора *ISim*.
20. Підготувати та захистити звіт.

Додаток – 1 (7-сегментний LED індикатор)

Будова:

В найрозповсюдженішому варіанті семи сегментний індикатор являє собою модуль, який містить 8 окремих елементів відображення:

- 7-сегментів, розміщених у вигляді цифри *8*, включення яких у різних комбінаціях дозволяє відображати різні цифри. Сегменти зазвичай позначаються латинськими літерами: *A*, *B*, *C*, *D*, *E*, *F*, *G*
- Десяткової крапки (*DP*), розміщеної праворуч від цифри внизу. Крапка призначена для відображення дробових чисел.



Малюнок 15 – Модуль семи сегментного індикатора.

Кожен елемент відображення – окремий світлодіод. Існує 2 основні способи є'єднання світлодіодів в складі модуля:

- Схема із загальним анодом (аноди всіх елементів індикації об'єднані спільним виводом *COM*, катод кожного елемента індикації має окремий вивід).

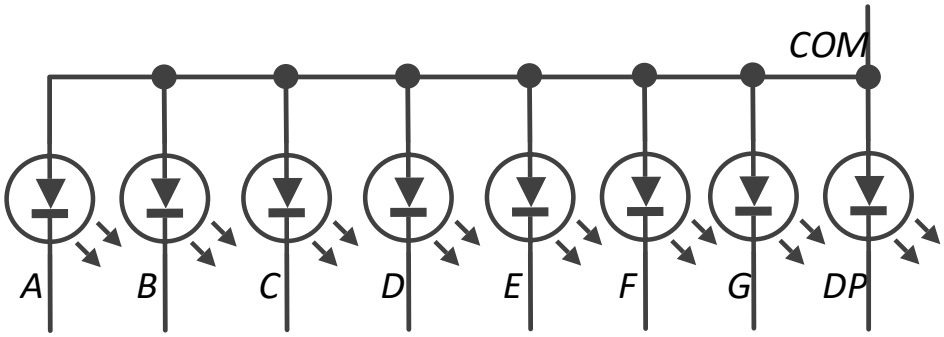
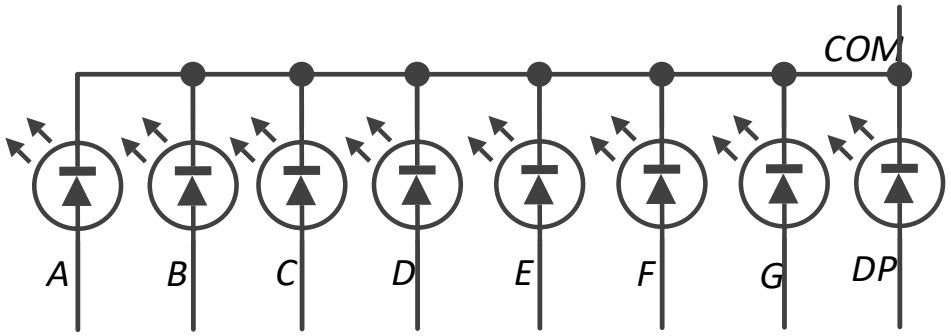


Figure 16 – Схема із загальним анодом.

Якщо модуль семи сегментного індикатора реалізований згідно даної схеми, то керування елементами індикації здійснюється наступним чином:

- Логічна *«1»* (вхід *A..DP* приєднано до *VDD* через обмежуючий резистор) – вимкнути.
- Логічний *«0»* (вхід *A..DP* приєднано до *GND* через обмежуючий резистор) – увімкнути.

- Схема із загальним катодом (катоди всіх елементів індикації об'єднані спільним виводом *COM*, анод кожного елемента індикації має окремий вивід).



Малюнок 17 – Схема із загальним катодом.

Якщо модуль семи сегментного індикатора реалізований згідно даної схеми, то керування елементами індикації здійснюється наступним чином:

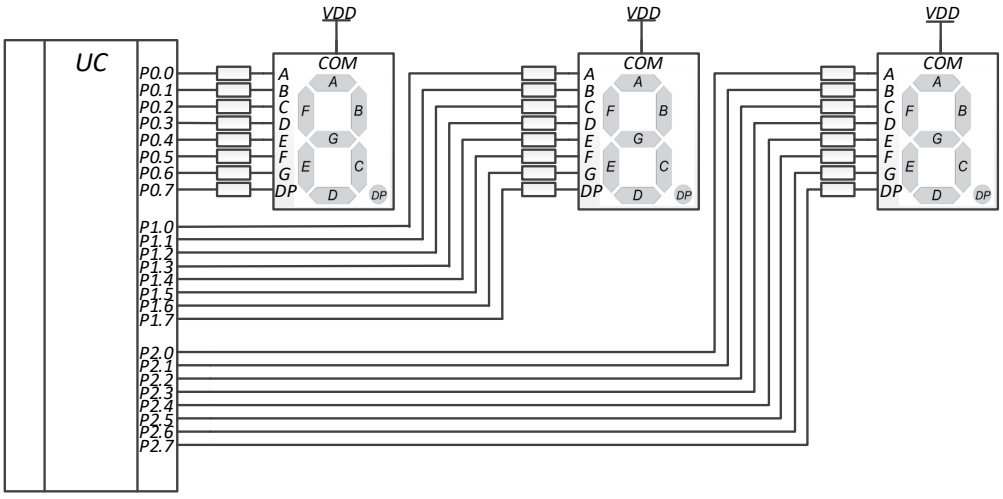
- Логічна «*1*» (вхід *A..DP* приєднано до *VDD* через обмежуючий резистор) – увімкнути.
- Логічний «*0*» (вхід *A..DP* приєднано до *GND* через обмежуючий резистор) – вимкнути.

Схеми підключення:

Є *2* основних підходи до використання *7*-сегментних індикаторів:

- ***Метод статичної індикації.***
- ***Метод динамічної індикації.***

Метод статичної індикації полягає в тому, що кожен окремий модуль має окремі сигнали керування елементами індикації *A..DP*.



Малюнок 18 – Схема для статичної індикації.

Основні недоліки методу:

- Вимагається більша кількість виводів мікросхеми (*8* виводів для кожного окремого модуля індикації).
- Вимагається більша кількість ліній на платі (*8* для кожного окремого модуля індикації)
- Вимагається більша кількість зовнішніх компонентів (*8* резисторів для кожного окремого модуля індикації).

Основні переваги методу:

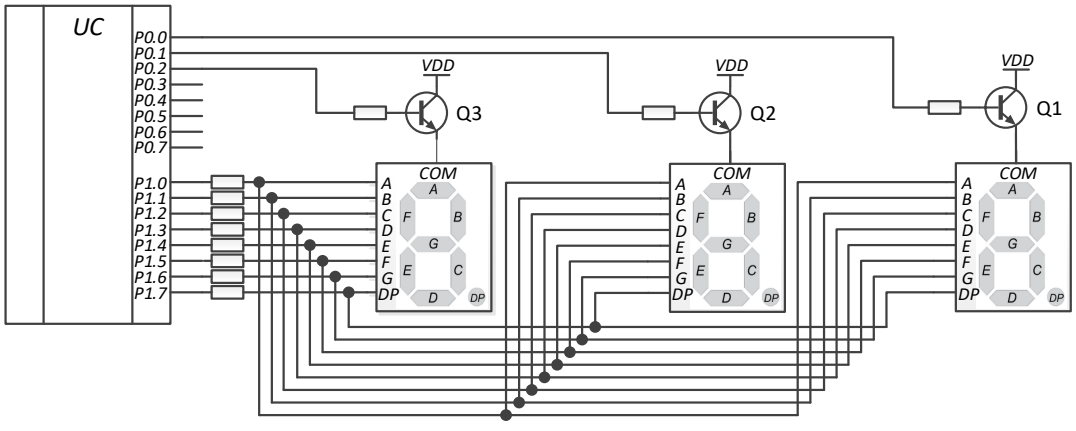
- Простота реалізації керування. Не вимагається динамічна зміна стану портів при незмінному числовому значенні, що відображається.

Приклад застосування:
Щоб вивести число «*127*», необхідно встановити порти *P0..P2* в наступний стан:

	ЦИФРА-1		ЦИФРА-2		ЦИФРА-7	
ЕЛЕМЕНТ	ПІН	ЗНАЧЕННЯ	ПІН	ЗНАЧЕННЯ	ПІН	ЗНАЧЕННЯ
A	P0.0	1	P1.0	0	P2.0	0
B	P0.1	0	P1.1	0	P2.1	0
C	P0.2	0	P1.2	1	P2.2	0
D	P0.3	1	P1.3	0	P2.3	1
E	P0.4	1	P1.4	0	P2.4	1
F	P0.5	1	P1.5	1	P2.5	1
G	P0.6	1	P1.6	0	P2.6	1
H	P0.7	1	P1.7	1	P2.7	1

Елементи відображення активуються логічним «*0*», оскільки у схемі використовуються модулі індикації, реалізовані за схемою зі спільним анодом.

Метод динамічної індикації полягає в тому, що всі модулі мають спільні сигнали керування елементами індикації *A..DP*. Але в кожен момент часу лише один модуль є активним (активація відбувається за допомогою керування станом виводу *COM*). Усі інші модулі – вимкнені.



Малюнок 19 – Схема для динамічної індикації.

Основні недоліки методу:

- У порівнянні з методом статичної індикації – вища складність реалізації керування. Вимагається динамічна зміна стану портів при незмінному числовому значенні, що відображається.

Основні переваги методу:

- У порівнянні з методом статичної індикації - вимагається менша кількість виводів мікросхеми (*8* загальних виводів і по одному для кожного окремого модуля індикації).
- У порівнянні з методом статичної індикації - вимагається менша кількість ліній на платі (8 загальних і по одній для кожного окремого модуля індикації)
- У порівнянні з методом статичної індикації - вимагається менша кількість зовнішніх компонентів (*8* резисторів і по одному транзистору для кожного окремого модуля індикації).

Приклад застосування:
Щоб вивести число «*127*», необхідно реалізувати циклічне переключення наступних станів:

1. Активація розряду одиниць:

	ЦИФРА-7	
ЕЛЕМЕНТ	ПІН	ЗНАЧЕННЯ
A	P1.0	0
B	P1.1	0
C	P1.2	0
D	P1.3	1
E	P1.4	1
F	P1.5	1
G	P1.6	1
H	P1.7	1
База Q1	P0.0	1
База Q2	P0.1	0
База Q3	P0.2	0

2. Активація розраду десятків:

	ЦИФРА-2	
ЕЛЕМЕНТ	ПІН	ЗНАЧЕННЯ
A	P1.0	0
B	P1.1	0
C	P1.2	1
D	P1.3	0
E	P1.4	0
F	P1.5	1
G	P1.6	0
H	P1.7	1
База Q1	P0.0	0
База Q2	P0.1	1
База Q3	P0.2	0

3. Активація розряду сотень:

	ЦИФРА-1	
ЕЛЕМЕНТ	ПІН	ЗНАЧЕННЯ
A	P1.0	1
B	P1.1	0
C	P1.2	0
D	P1.3	1
E	P1.4	1
F	P1.5	1
G	P1.6	1
H	P1.7	1
База Q1	P0.0	0
База Q2	P0.1	0
База Q3	P0.2	1

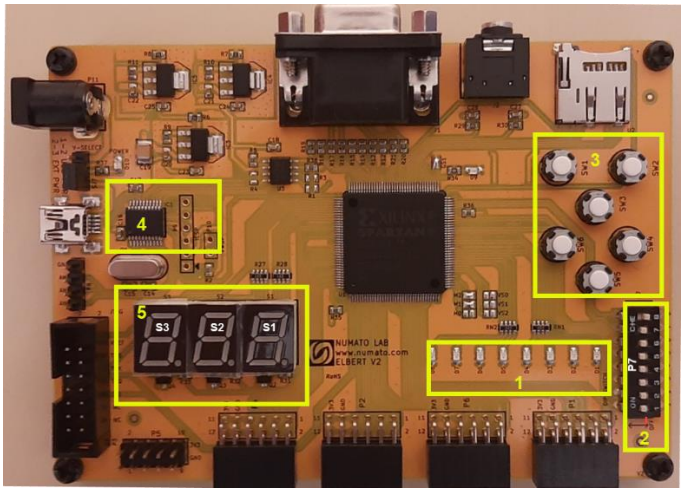
- Елементи відображення активуються логічним «*0*», оскільки у схемі використовуються модулі індикації, реалізовані за схемою зі спільним анодом.
- Транзистори відкриваються логічною «*1*».

Інтеграція зі стендом Elbert V2 – Spartan 3A FPGA

- У складі стенда є 3 семи сегментні *LED* модулі індикації. Див. *Додаток – 2* (інформація про стенд).
- Семи-сегментні LED модулі, які містяться на стенді, реалізовані за схемою зі спільним анодом.
- Стенд реалізує метод динамічної індикації. Всі *3* семи сегментні модулі індикації мають спільні сигнали керування елементами індикації *A..DP*. Активація модулів реалізована за допомогою керування станом виводу *COM*.
- Таблиця відповідності між виводами *FPGA* і сигналами керування індикаторами приведена в *Додаток – 2* (інформація про стенд).

Додаток – 2 (інформація про стенд)

Розміщення елементів на стенді:



1. Світлодіоди *D1-D8*. Конфігурацію виводів *FPGA*, до яких приєднано світлодіоди, описано в розділі *LED* оригінального *UCF* файла (elbertv2.ucf). Світлодіоди вмикаються подачею логічної «1» на відповідний вивід *FPGA*.
2. Регістр з 8 *DIP* перемикачів. В положенні *ON* перемикач приєднує вивід *FPGA* до *GND*. В положенні *OFF* – коло розірване і вивід нікуди не підключений. Конфігурацію виводів *FPGA*, до яких приєднано перемикачі, описано в розділі *DP Switches* оригінального *UCF* файла (elbertv2.ucf).
   Оскільки *UCF* конфігурує виводи як *PULL UP* то положення *ON* (вивід приєднано до *GND*) буде сприйматись як логічний «0», а положення *OFF* (коло розірване, проте вивід підтягнутий до *VDD* за допомогою внутрішнього резистора), буде сприйматись як логічна «1».
3. 5 кнопок *PUSH BUTTON*. В положенні *ON* перемикач приєднує вивід *FPGA* до *GND*. В положенні *OFF* – коло розірване і вивід нікуди не підключений. Конфігурацію виводів *FPGA*, до яких приєднано кнопки, описано в розділі *Switches* оригінального *UCF* файла (elbertv2.ucf).
   Оскільки *UCF* конфігурує виводи як *PULL UP*, то положення *ON* (вивід приєднано до *GND*) буде сприйматись як логічний «*0*», а положення *OFF* (коло розірване, проте вивід підтягнутий до *VDD* за допомогою внутрішнього резистора), буде сприйматись як логічна «*1*».
4. Мікроконтролер, який призначений для програмування стенда і генерування тактового сигналу для *FPGA*. Частота генерованого тактового сигналу становить *12MHz*. Конфігурацію виводу *FPGA*, до якого приєднано сигнал *Clk* від мікроконтролера, описано в розділі *UCF for ElbertV2 Development Board* оригінального *UCF* файла (elbertv2.ucf).

5. Семи сегментні індикаторні *LED* модулі *S1-S3*. Оскільки модулі, які містяться на стенді, реалізовані за схемою зі спільним анодом, то для увімкнення заданих елементів індикації (*A..DP*), необхідно подати логічну *«1»* на вивід *COM* індикаторного модуля, та логічний *«0»* на відповідний вхід керування (*A..DP*). Конфігурацію виводів *FPGA*, до яких приєднано індикаторні модулі, описано в розділі *Seven Segment Display* оригінального *UCF* файла (elbertv2.ucf).

Більше інформації та схему електричну принципову можна знайти за наступним посиланням:
https://numato.com/product/elbert-v2-spartan-3a-fpga-development-board

Таблиці відповідності між позначеннями в оригінальному UCF файлі (elbertv2.ucf) і елементами на стенді *Elbert V2 – Spartan 3A FPGA.*

Розділ *LED*

Позначення в UCF файлі	*Маркування на стенді Elbert V2*
LED[0]	D8
LED[1]	D7
LED[2]	D6
LED[3]	D5
LED[4]	D4
LED[5]	D3
LED[6]	D2
LED[7]	D1

Розділ *DP Switches*

Позначення в UCF файлі	*Маркування на стенді Elbert V2*
DPSwitch[0]	P7[8]
DPSwitch[1]	P7[7]
DPSwitch[2]	P7[6]
DPSwitch[3]	P7[5]
DPSwitch[4]	P7[4]
DPSwitch[5]	P7[3]
DPSwitch[6]	P7[2]
DPSwitch[7]	P7[1]

Розділ *Seven Segment Display*

Позначення в UCF файлі	*Елемент на стенді Elbert V2*
SevenSegment[7]	Сегмент A
SevenSegment[6]	Сегмент B
SevenSegment[5]	Сегмент C
SevenSegment[4]	Сегмент D
SevenSegment[3]	Сегмент E
SevenSegment[2]	Сегмент F
SevenSegment[1]	Сегмент G
SevenSegment[0]	Сегмент DP
Enable[2]	Загальний (COM) S3
Enable[1]	Загальний (COM) S2
Enable[0]	Загальний (COM) S1