Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №6**

з дисципліни «Комп’ютерна схемотехніка»

**Тема: «ПРИСТРОЇ ДЛЯ ПЕРЕТВОРЕННЯ ЧИСЕЛ.**

**ТИПОВІ ВУЗЛИ КОМП’ЮТЕРА»**

Виконав:

 студент групи ІО-31

Устимчик П.Я..

Перевірив:

Нікольський С.С.

Київ 2025 р.

**Хід роботи**

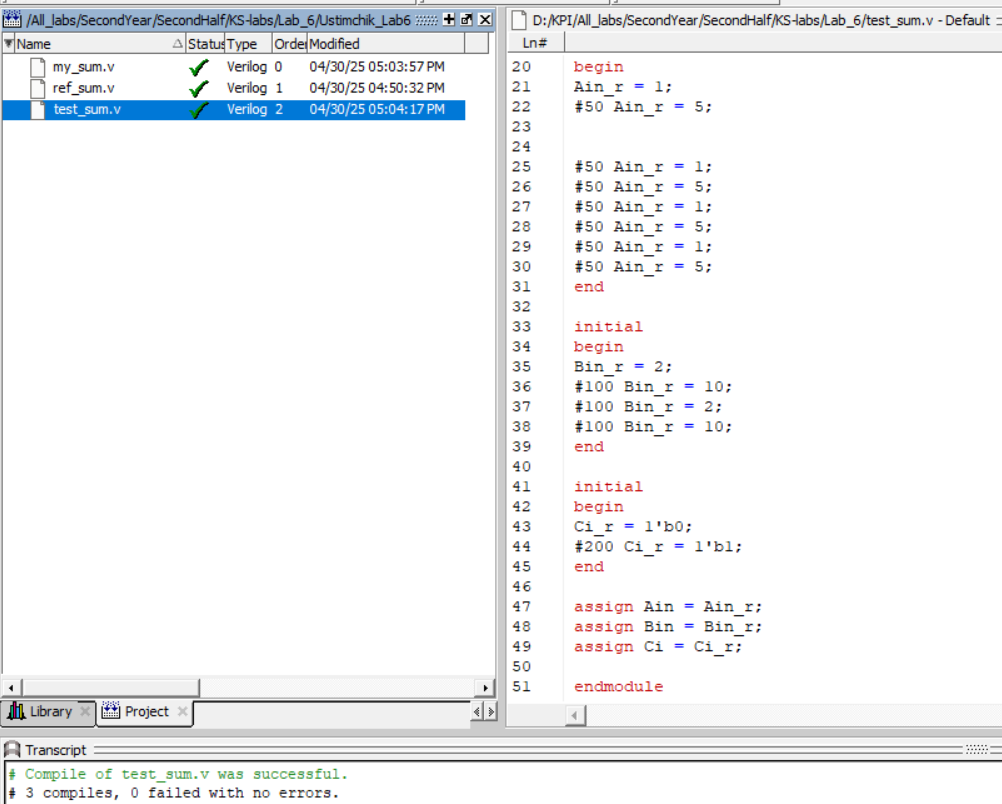
Варіант:

312210 - 110000110010₂, звідси :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 | 0 |
| h6 | h5 | h4 | h3 | h2 | h1 |

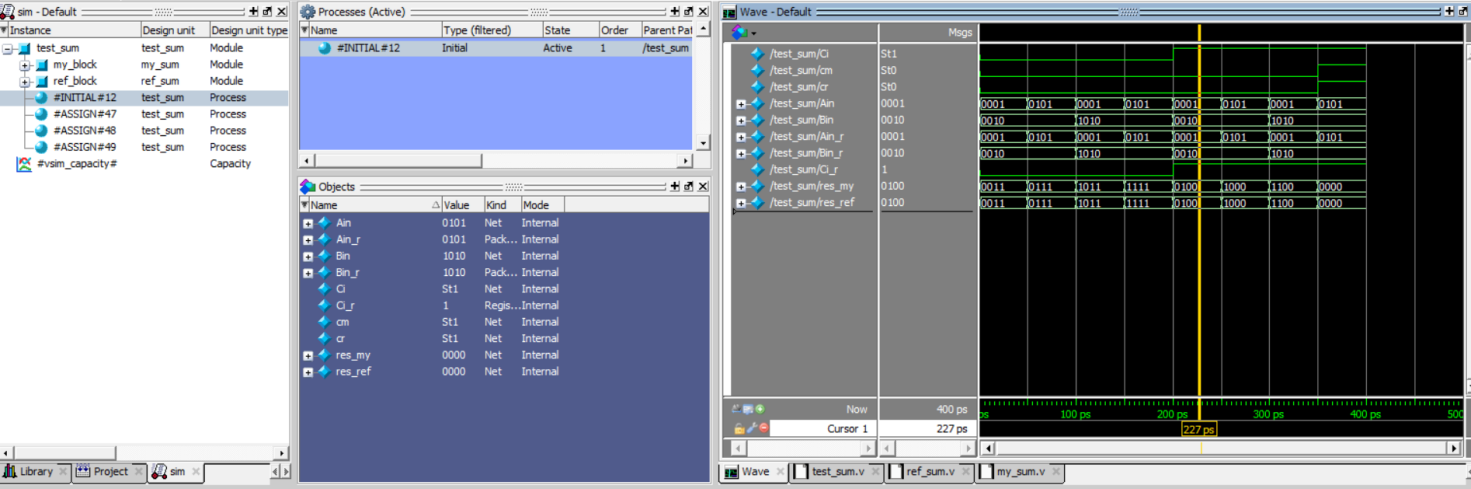
|  |  |
| --- | --- |
| h1 h2 h3 | Розрядність суматора |
| 0 1 0 | 6 |

Створюємо проект, а також файли на мові Verilog відповідно методички.Прописуємо код та проводимо компіляцію.

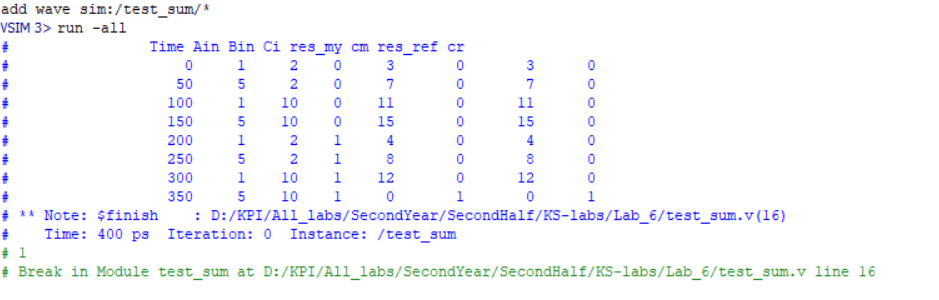


Компіляція пройшла успішно

Перейдемо у режим моделювання :



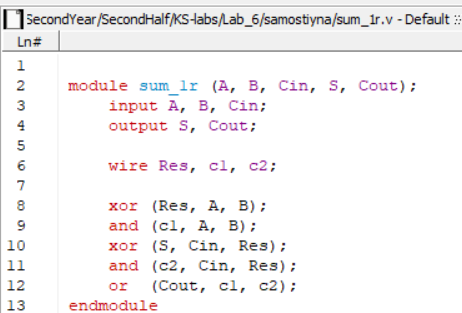
Результати з консолі:



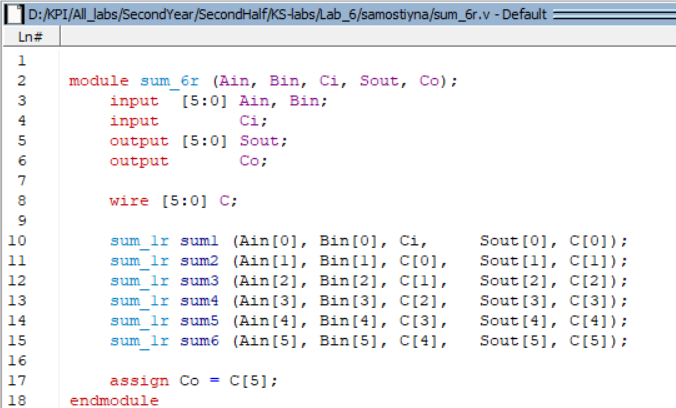
Ми бачимо, що на входи суматора подаються операнди Ain, Bin та сигнал переносу Ci. Результат додавання виводиться на res\_my — це результат нашого суматора, і cm — сигнал переносу з нього. Також паралельно працює модуль ref\_sum, який реалізований на поведінковому рівні, і виводить res\_ref та cr.Під час симуляції бачимо, що для кожної комбінації входів результати res\_my і res\_ref повністю збігаються, як і сигнали переносу cm і cr. Це підтверджує, що реалізація мого суматора my\_sum працює правильно. Симуляція виконана коректно.

**Самостійна робота**

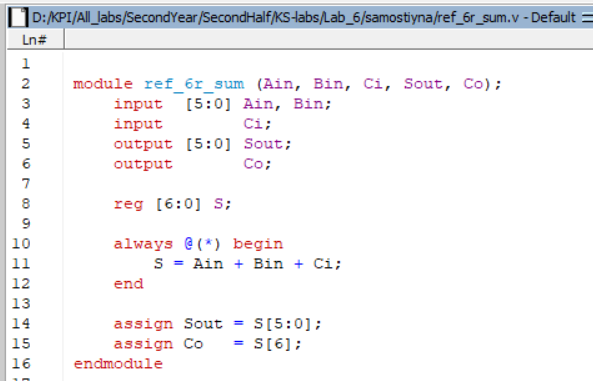
Створимо новий проєкт та створимо там файл на мові Verilog та напишемо там код для 6-ти розрядного суматора. За основу візьмемо код, який використовувався у прикладі вище . У результаті маємо отакий код:



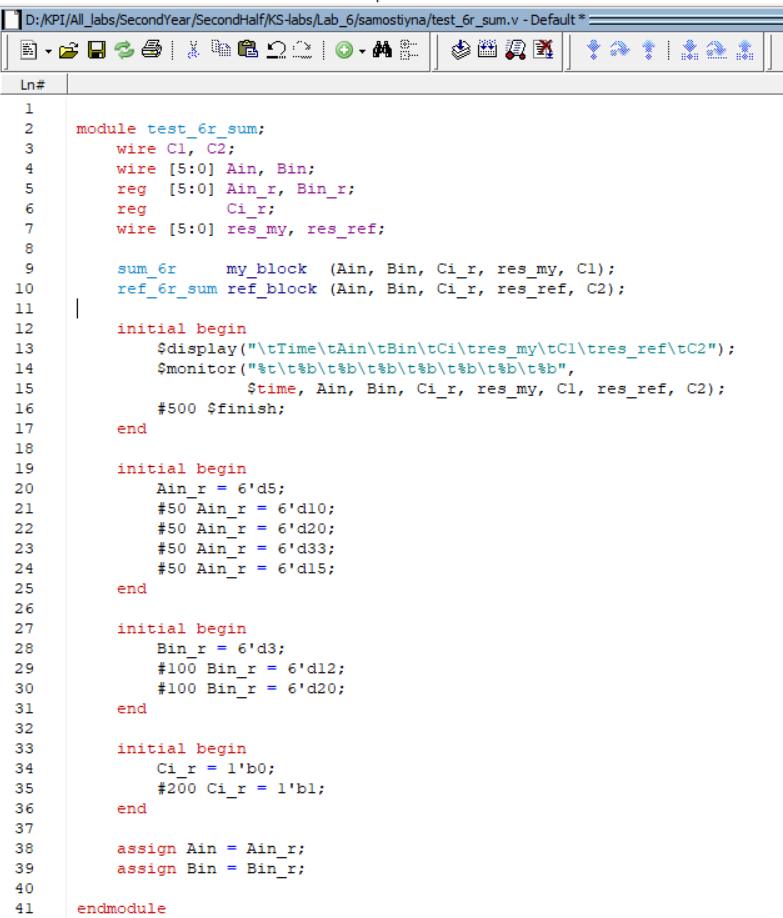
У цьому модулі реалізовано логіку повного однорозрядного суматора з урахуванням вхідного переносу. Сума обчислюється через два XOR, а перенос — через два AND і один OR. Цей модуль є базовим елементом для побудови багаторозрядного суматора.



У цьому модулі реалізовано 6-розрядний суматор на основі послідовного з'єднання 6 повних однорозрядних суматорів (sum\_1r). Кожен розряд враховує перенос із попереднього, що дозволяє точно передавати перенесення по всьому розряду. Структурна побудова дозволяє краще уявити внутрішню логіку додавання.

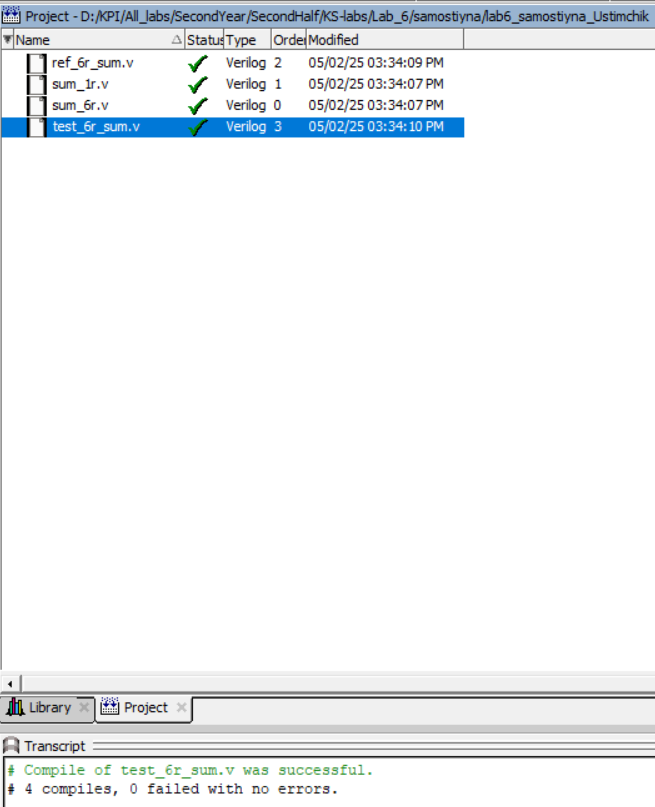


У даному модулі використовується поведінковий опис суматора, де результат обчислюється за допомогою оператора +. Це дозволяє швидко і просто реалізувати додавання з урахуванням переносу, без логічної деталізації. Такий модуль зручно використовувати як еталон для перевірки.

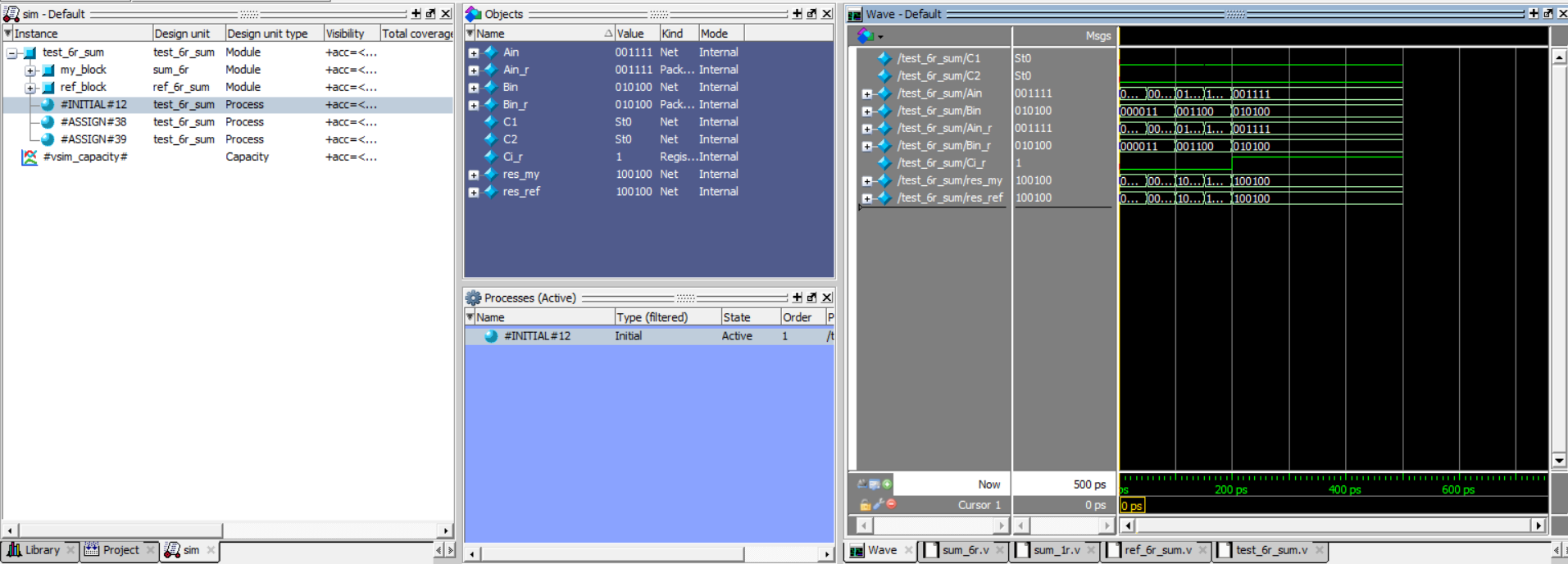


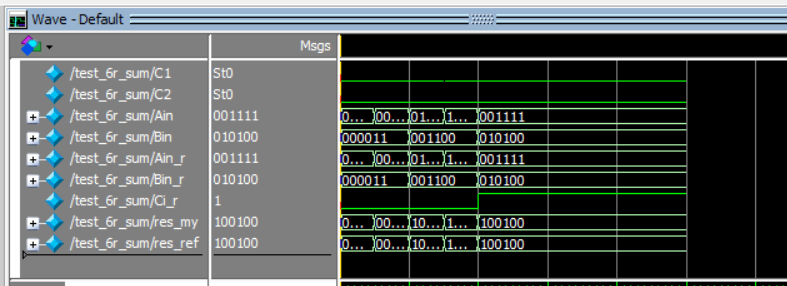
У модулі тестбенча реалізовано перевірку роботи суматора шляхом подачі різних комбінацій вхідних значень Ain, Bin та Ci. Результати виводяться в консоль через $monitor. Порівняння результатів структурної та поведінкової реалізацій дозволяє впевнено перевірити правильність роботи власного суматора.

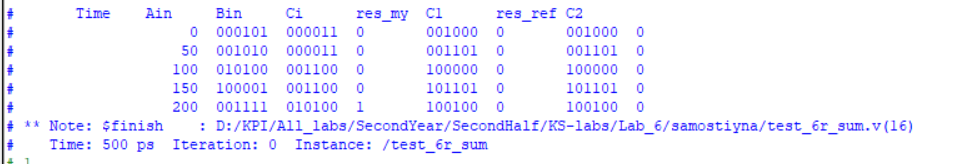
Тепер скомпілюємо всі ці файли .



Перейдемо у режим модулювання та перевіримо чи правильні результати:







Все виконано коректно

**Висновок:**

У ході виконання лабораторної роботи було реалізовано 6-розрядний суматор двома способами: структурно та поведінково. Структурна реалізація (sum\_6r) була побудована на основі шести однорозрядних повних суматорів (sum\_1r), які з'єднані послідовно. Поведінкова модель (ref\_6r\_sum) виконувала ті ж самі обчислення за допомогою оператора +.

Для перевірки коректності роботи було створено тестовий модуль (test\_6r\_sum), який подавав різні комбінації вхідних даних та порівнював результати обох реалізацій. Під час симуляції в середовищі ModelSim результати збігалися для всіх заданих комбінацій, що підтверджує правильність структурної побудови суматора.

Таким чином, поставлене завдання виконано повністю: суматор працює коректно, а тестування показало повну відповідність поведінковій моделі.

**Посилання на git\_hub :**