Лабораторна робота 5

Опис комбінаційних цифрових схем мовою SystemVerilog

Призначення роботи

Лабораторна робота навчає перетворювати булеву функцію на працездатний модуль SystemVerilog і доводити правильність реалізації симуляцією. Усі дії спочатку виконуються на окремому стандартному прикладі F(A,B,C)=Σm(1,3,6,7), якого немає серед варіантів. Після цього той самий алгоритм застосовується до власного варіанта.

Мета роботи

- скласти таблицю істинності за сумою мінтермів;
- мінімізувати функцію картою Карно;
- описати схему оператором assign;
- описати ту саму схему блоком always_comb і case;
- створити testbench, перевірити вісім комбінацій та прочитати часову діаграму EPWave.

Програмне середовище

Використовується EDA Playground — браузерне середовище для мов опису апаратури. Встановлювати програму не потрібно. Для запуску коду необхідно увійти через Google або зареєструвати безкоштовний обліковий запис.

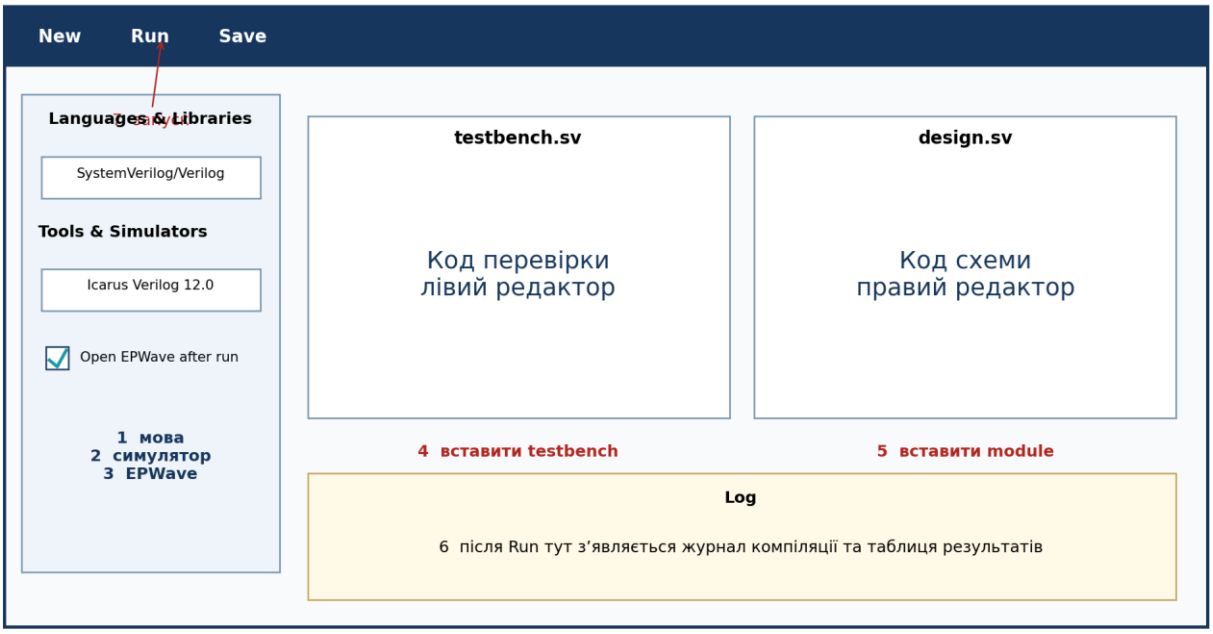
Відкрити середовище https://www.edaplayground.com/

Налаштування	Потрібне значення
Testbench + Design	SystemVerilog/Verilog
Tools & Simulators	Icarus Verilog 12.0 у групі Free Simulators
UVM OVM	None
Other Libraries	None
Open EPWave after run	Увімкнути
Compile options	Залишити порожнім

Підготовка до роботи

1. **Відкрийте EDA Playground.** У верхній частині сторінки натисніть Log In. Оберіть вхід через Google або Register for a full account. Без входу код можна редагувати, але запуск і збереження недоступні.
2. **Перевірте розміщення панелей.** Ліворуч знаходяться Languages and Libraries та Tools and Simulators. У центрі лівий редактор testbench.sv, праворуч редактор design.sv. Унизу відкривається вкладка Log.

3. **Оберіть мову.** У блоці Languages and Libraries знайдіть список Testbench + Design і оберіть SystemVerilog/Verilog. UVM OVM залиште None.
4. **Оберіть симулятор.** Розгорніть Tools & Simulators. У групі Free Simulators оберіть Icarus Verilog 12.0.
5. **Увімкніть EPWave.** У тому самому блоці поставте прапорець Open EPWave after run. EPWave відкриється в новому вікні після успішного запуску; дозвольте спливні вікна для сайту.



Розташування основних елементів EDA Playground

Короткі теоретичні відомості

Комбінаційна схема

Комбінаційна схема (combinational circuit) не має пам'яті: її вихід визначається поточними входами. Тому для функції F(A,B,C) достатньо перевірити всі 2^3=8 комбінацій A, B і C.

Мінтерм і сума мінтермів

Мінтерм (minterm) відповідає одному рядку таблиці істинності. Номер мінтерма дорівнює десятковому значенню двійкового слова ABC, де A — старший біт, C — молодший. Наприклад, m6 відповідає ABC=110.

Засоби опису

Конструкція	Зміст
assign	Безперервно обчислює вихід за виразом праворуч
always_comb	Описує комбінаційну логіку процедурно

Конструкція	Зміст
case	Вибирає гілку за кодом входів
~	Побітове НЕ; у формулах відповідає надрисці
&	Побітове І
|	Побітове АБО
DUT	Design Under Test — модуль, який перевіряється
testbench	Тестове оточення, що подає стимули й читає виходи

Стандартний приклад повне виконання

Крок 1 Читання завдання

Потрібно реалізувати F(A,B,C)=Σm(1,3,6,7). Це означає: F=1 для номерів 1, 3, 6 і 7; для номерів 0, 2, 4 і 5 значення F=0.

m	Двійковий код ABC	F
0	000	0
1	001	1
2	010	0
3	011	1
4	100	0
5	101	0
6	110	1
7	111	1

Крок 2 Побудова таблиці істинності

Рядки розташовуються у порядку від 000 до 111. Для кожного рядка число ABC переводиться в десяткове. Якщо число є у списку Σm, у стовпці F ставиться 1.

m	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Крок 3 Побудова карти Карно

Для трьох змінних карта має два рядки за A і чотири стовпці за BC. Стовпці записуються у коді Грея 00, 01, 11, 10. Такий порядок потрібний, щоб сусідні клітинки відрізнялися лише одним бітом.

A BC	00	01	11	10
A=0	m0 0	m1 1	m3 1	m2 0
A=1	m4 0	m5 0	m7 1	m6 1

- Перша група охоплює m1 і m3. У цих клітинках A=0 та C=1 не змінюються, B змінюється. Отже, група дає $A\overline{C}$.
- Друга група охоплює m6 і m7. У цих клітинках A=1 та B=1 не змінюються, C змінюється. Отже, група дає AB.
- Після об'єднання груп отримуємо $F=A\overline{C}+AB$. Жодний мінтерм з одиницею не втрачено, а клітинки з нулями до груп не включено.

Крок 4 Перетворення формули на SystemVerilog

У математичному записі маємо $F=A\overline{C}+AB$. У коді надриска замінюється оператором ~, добуток — оператором &, сума — оператором |. Кожний добуток береться в дужки, тому отримуємо F=(~A & C) | (A & B).

Крок 5 Створення design.sv через assign

```
module logic_function (
    input  logic A, B, C,
    output logic F
);
    assign F = (~A & C) | (A & B);
endmodule
```

Рядок або фрагмент	Пояснення
module logic_function	Початок модуля та його назва
input logic A, B, C	Три однобітні входи
output logic F	Один однобітний вихід
assign F = ...	Безперервне обчислення виходу
endmodule	Завершення модуля

Дія в інтерфейсі. Виділіть увесь початковий текст у правому редакторі design.sv, видаліть його та вставте наведений module. Переконайтеся, що після assign стоїть крапка з комою.

Крок 6 Створення testbench.sv

```
`timescale 1ns/1ps

module testbench;
    logic A, B, C;
    logic F;
    integer i;

    logic_function dut (
        .A(A),
        .B(B),
        .C(C),
        .F(F)
    );

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0, testbench);

        $display("A B C | F");
        $display("--------");

        for (i = 0; i < 8; i = i + 1) begin
            {A, B, C} = i[2:0];
            #1;
            $display("%0b %0b %0b | %0b", A, B, C, F);
        end

        $finish;
    end
endmodule
```

Фрагмент	Що відбувається
`timescale 1ns/1ps	Одна одиниця #1 дорівнює 1 нс; точність часу 1 пс
logic A B C F	Створюються сигнали тестового оточення
logic_function dut	Створюється екземпляр перевірюваного модуля
$dumpfile	Задається файл dump.vcd для EPWave
$dumpvars	До VCD додаються сигнали testbench
for i=0 i<8	Послідовно перебираються всі вісім наборів
{A B C}=i[2:0]	Три молодші біти і одночасно подаються на входи
#1	Симуляція переходить до наступної наносекунди
$display	Поточні значення друкуються у Log
$finish	Симуляція завершується

Дія в інтерфейсі. Виділіть початковий текст у лівому редакторі testbench.sv, видаліть його та вставте наведений testbench.

Крок 7 Запуск першої симуляції

1. **Перевірте параметри.** Testbench + Design має містити SystemVerilog/Verilog, a Simulator — Icarus Verilog 12.0.
2. **Натисніть Run.** Кнопка розташована у верхній панелі. Комбінація клавіш Ctrl+Enter виконує ту саму дію.
3. **Відкрийте Log.** Якщо нижня панель показує Share, перейдіть на вкладку Log. Спочатку відображаються команди компіляції, потім результат $display.
4. **Знайдіть завершення.** Успішний запуск містить вісім рядків таблиці та повідомлення про $finish. Червоний текст із номером рядка означає помилку, яку потрібно виправити.

Очікуваний результат у Log

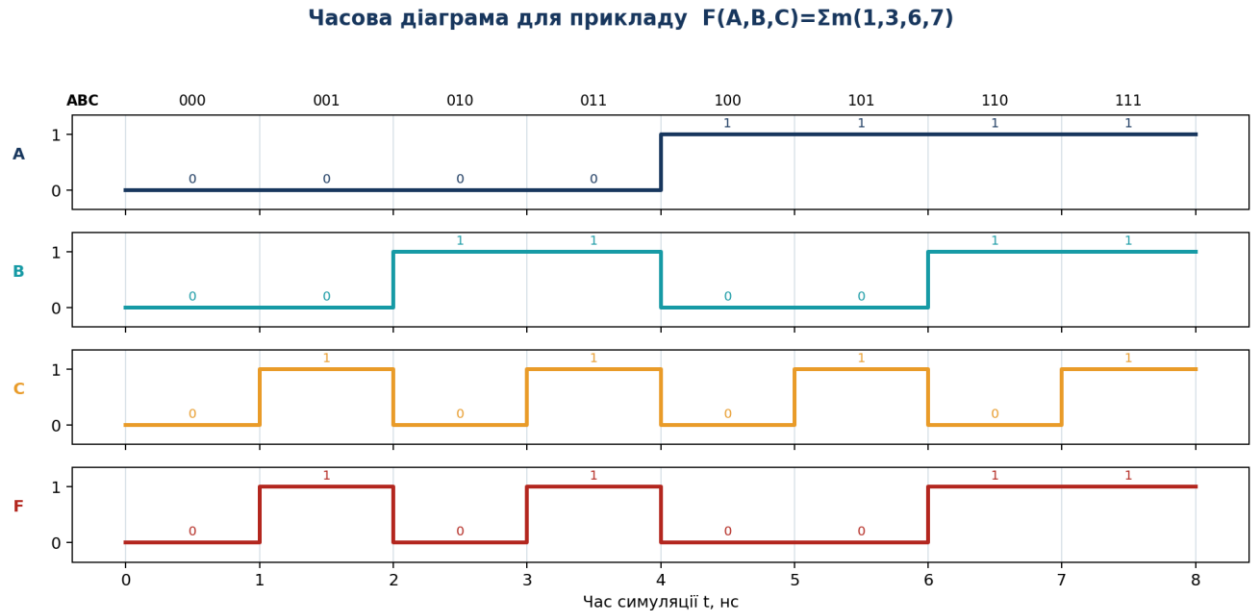
```
A B C | F
0 0 0 | 0
0 0 1 | 1
0 1 0 | 0
0 1 1 | 1
1 0 0 | 0
1 0 1 | 0
1 1 0 | 1
1 1 1 | 1
```

Перевірка виконується рядок за рядком. Наприклад, для ABC=011 десятковий номер дорівнює 3, а m3 є у списку, тому F=1. Для ABC=101 номер дорівнює 5, m5 у списку немає, тому F=0.

Крок 8 Відкриття EPWave

1. **Дочекайтеся нового вікна.** Після успішного Run відкривається EPWave. Якщо вікно не з'явилося, дозвольте pop-ups і повторіть Run.
2. **Натисніть Get Signals.** Відкриється вікно Get Signals to Display зі списком областей Scope і сигналів Signal Name.
3. **Оберіть область testbench.** У списку Scope виберіть testbench. Далі виділіть A, B, C та F.
4. **Додайте сигнали.** Натисніть Append Selected. Якщо список містить лише потрібні сигнали, можна натиснути Append All. Потім закрийте вікно вибору.
5. **Розташуйте сигнали.** Залиште порядок A, B, C, F зверху вниз. За потреби збільшіть масштаб так, щоб було видно інтервал від 0 до 8 нс.

Крок 9 Детальне читання часової діаграми



Очікувана часова діаграма стандартного прикладу

Горизонтальна вісь показує час симуляції. Кожний інтервал шириною 1 нс відповідає одному набору входів. Низький рівень означає 0, високий — 1. Вертикальна зміна лінії є моментом перемикання сигналу.

- A змінюється найповільніше: A=0 протягом наборів 000–011 і A=1 протягом 100–111.
- B змінюється через кожні два набори: 0, 0, 1, 1, 0, 0, 1, 1.
- C змінюється на кожному наборі: 0, 1, 0, 1, 0, 1, 0, 1.
- F має послідовність 0, 1, 0, 1, 0, 0, 1, 1. Одиниці спостерігаються саме для m1, m3, m6 і m7.
- На інтервалі 1–2 нс входи дорівнюють 001, тому F=1. На інтервалі 4–5 нс входи 100, тому F=0. На інтервалі 7–8 нс входи 111, тому F=1.
- Якщо F відрізняється від очікуваного лише для одного інтервалу, потрібно перевірити відповідний мінтерм. Якщо F постійно X, слід перевірити підключення порту F та повноту присвоєнь.

Чому зміна відбувається на межі інтервалу. Команда {A,B,C}=i[2:0] змінює входи, після чого комбінаційний assign перераховує F у межах поточного симуляційного часу. Команда #1 утримує набір одну наносекунду і лише потім переходить до наступної ітерації. Тому рівень F усередині кожного інтервалу відповідає поточному ABC.

Крок 10 Реалізація через always_comb і case

```
module logic_function (
    input  logic A, B, C,
    output logic F
);
    always_comb begin
        case ({A, B, C})
        3'b001: F = 1'b1;
        3'b011: F = 1'b1;
        3'b110: F = 1'b1;
        3'b111: F = 1'b1;
            default: F = 1'b0;
        endcase
    end
endmodule
```

У case аналізується трирозрядний вектор {A,B,C}. Для кодів 001, 011, 110 і 111 встановлюється F=1. Гілка default встановлює F=0 для всіх інших кодів і гарантує повне присвоєння виходу.

1. **Замініть код design.sv.** Видаліть реалізацію з assign і вставте реалізацію always_comb.
2. **Не змінюйте testbench.** Ті самі стимули мають перевірити другий опис.
3. **Натисніть Run.** У Log має повторитися та сама таблиця, а EPWave має показати ту саму послідовність F.
4. **Зробіть висновок.** Однакова таблиця для двох описів підтверджує їх функціональну еквівалентність.

Алгоритм виконання власного варіанта

1. **Перепишіть функцію.** Використовуйте лише свій рядок таблиці варіантів.
2. **Перетворіть номери на ABC.** Запишіть усі числа 0–7 трьома бітами.
3. **Заповніть таблицю.** F=1 тільки для номерів у Σm.
4. **Побудуйте карту Карно.** Стовпці BC розташуйте як 00, 01, 11, 10.
5. **Утворіть найбільші групи.** Спочатку групи з чотирьох, потім із двох, потім поодинокі клітинки.
6. **Запишіть мінімізовану ДНФ.** ДНФ — диз'юнктивна нормальна форма, тобто сума добутків.
7. **Створіть assign.** Перетворіть надриски, І та АБО на ~, & і |.
8. **Запустіть testbench.** Перевірте всі вісім рядків і часову діаграму.
9. **Створіть case.** Перелічіть усі коди, для яких F=1, і додайте default F=0.
10. **Порівняйте результати.** Обидві реалізації повинні дати однакові Log і EPWave.

Варіанти завдань

№	Функція	Способи опису	Набори
1	F(A,B,C)=Σm(1,2,3,5,7)	assign і always_comb case	000–111
2	F(A,B,C)=Σm(0,2,3,6,7)	assign і always_comb case	000–111
3	F(A,B,C)=Σm(1,3,4,5,7)	assign і always_comb case	000–111
4	F(A,B,C)=Σm(0,1,4,6)	assign і always_comb case	000–111
5	F(A,B,C)=Σm(2,3,5,6,7)	assign і always_comb case	000–111
6	F(A,B,C)=Σm(0,2,5,7)	assign і always_comb case	000–111
7	F(A,B,C)=Σm(1,2,4,6,7)	assign і always_comb case	000–111
8	F(A,B,C)=Σm(0,3,4,5,7)	assign і always_comb case	000–111
9	F(A,B,C)=Σm(1,2,3,6)	assign і always_comb case	000–111
10	F(A,B,C)=Σm(0,1,3,4,6)	assign і always_comb case	000–111
11	F(A,B,C)=Σm(2,4,5,6,7)	assign і always_comb case	000–111
12	F(A,B,C)=Σm(0,2,3,5)	assign і always_comb case	000–111
13	F(A,B,C)=Σm(1,3,5,6)	assign і always_comb case	000–111
14	F(A,B,C)=Σm(0,1,2,6,7)	assign і always_comb case	000–111
15	F(A,B,C)=Σm(2,3,4,7)	assign і always_comb case	000–111
16	F(A,B,C)=Σm(0,3,5,6,7)	assign і always_comb case	000–111
17	F(A,B,C)=Σm(1,2,5,6)	assign і always_comb case	000–111

Пошук і виправлення помилок

Ознака	Ймовірна причина	Що зробити
syntax error	Пропущено ;, дужку або end	Перейдіть до номера рядка в Log і перевірте парні конструкції
Unknown module type	Назва module не збігається з DUT	У design.sv і testbench використайте logic_function

Ознака	Ймовірна причина	Що зробити
F постійно X	Вихід не підключено або не присвоєно	Перевірте .F(F), assign або default у case
EPWave не відкрився	Не встановлено прапорець або заблоковано pop-up	Увімкніть Open EPWave after run і дозвольте спливні вікна
EPWave порожній	$dumpfile або $dumpvars відсутні	Поверніть обидва системні виклики у початок initial
Один рядок неправильний	Помилка мінтерма або оператора	Звірте номер ABC з таблицею та карту Карно
assign правильний case ні	Пропущена гілка або default	Перелічіть усі мінтерми F=1 і додайте default F=0

Вимоги до звіту

- тема, мета і номер варіанта;
- початкова функція;
- таблиця істинності та карта Карно;
- групи карти й мінімізована ДНФ;
- код design.sv з assign;
- код design.sv з always_comb і case;
- код testbench.sv;
- консоль із вісьмома наборами;
- EPWave із сигналами A, B, C, F та підписаною очікуваною послідовністю;
- висновок про функціональну еквівалентність двох описів.

Питання для захисту

1. Що таке мова опису апаратури HDL?
2. Чим HDL відрізняється від звичайної програми?
3. Яке призначення конструкції module?
4. Що означають input і output?
5. Для чого використовують тип logic?
6. Що робить оператор assign?
7. Що означають оператори ~ & |?
8. Чим побітовий оператор & відрізняється від логічного &&?
9. Для чого потрібен always_comb?
10. Чому в always_comb потрібно присвоювати вихід на всіх шляхах?
11. Що робить конструкція case?
12. Що таке DUT?
13. Що таке testbench?

14. Навіщо після зміни входів використано затримку #1?

15. Що таке мінтерм?

16. Як читати запис Σm(1,3,6)?

17. Навіщо мінімізувати функцію перед описом?

18. Чим симуляція відрізняється від синтезу?

Контрольний список перед здаванням

- усі вісім наборів перевірено;
- F=1 тільки для заданих мінтермів;
- у формулах використано надриски, у коді — ~;
- відсутні значення X на робочих інтервалах;
- результати assign і always_comb однакові;
- знімки Log і EPWave читаються;
- у звіті є коротке пояснення часової діаграми.

Довідкові ресурси

EDA Playground

Офіційна коротка інструкція EDA Playground

Офіційна документація EPWave