Лабораторна робота №2
Вступ до моделювання апаратури. Опис та моделювання системи логічних функцій

Мета роботи: отримання базових знань про проектування пристроїв за допомогою мов опису апаратури, отримання навичок створення найпростіших модулів мовою Verilog та вміння написання стендів налагодження (test benches) для верифікації наявних модулів.

Зміст:

Короткі теоретичні відомості

Етапи проектування мовою опису апаратури

Дії «зовнішнього плану» починаються із створення основного підходу та функцій окремих блоків на рівні блок-схеми. Великі проекти, як правило, представляють складну ієрархічну структуру і на першому етапі відбувається створення блок-схеми, яка враховує ієрархічність структури, відбувається розбивка складної структури на невеликі компоненти (модулі), визначаються інтерфейси цих модулів, а також взаємозв'язки між модулями.

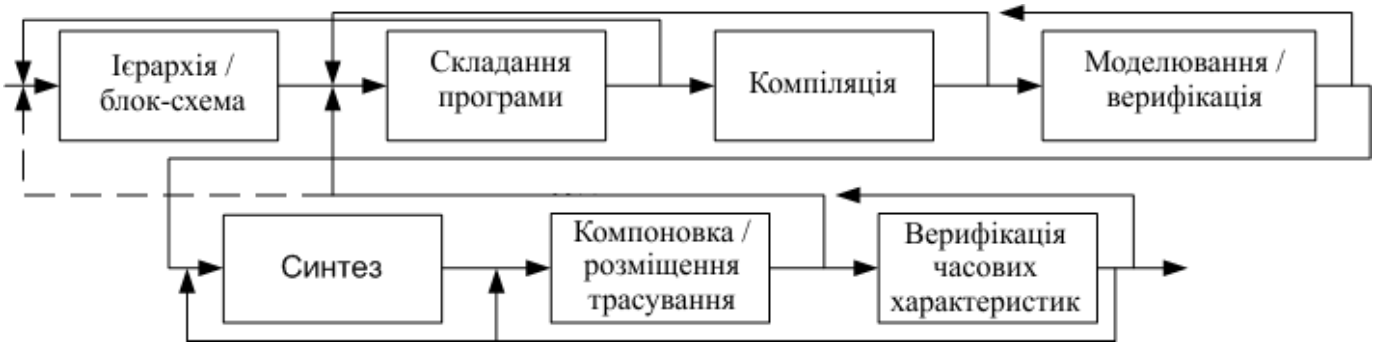


Рис. 1. Етапи процесу проектування мовою опису апаратури.

Наступний крок полягає у фактичному описі кожного з модулів, їх інтерфейсів та деталей внутрішньої будови за допомогою конкретної мови опису апаратури.

Написана програма підлягає трансляції (компіляції). У процесі компіляції текст програми аналізується наявність у ньому синтаксичних помилок, відбувається перевірка сумісності написаної програми з модулями, куди посилається програма. На цьому етапі рекомендується проводити компіляцію з перевіркою модулів, що допоможе уникнути розмноження синтаксичних помилок, дозволить вчасно виявити несумісність імен і т.д.

Етап моделювання дозволяє задавати вхідні сигнали і подавати їх на входи конструкції, що розробляється, а також спостерігати вихідні сигнали, не збираючи схему фізично. Для невеликих проектів вхідні сигнали можна задавати вручну та візуально спостерігати вихідні сигнали. Що стосується великих проектів мовами опису апаратури, зазвичай, використовують т.зв. налагоджувальні стенди (програмні засоби тестування, test benches), в яких вхідні сигнали подаються автоматично, а вихідні сигнали також автоматично порівнюються з очікуваними.

Насправді, моделювання є одним із етапів більш складного та великого етапу верифікації, мета якого переконатися в тому, що розроблені модулі проекту працюють відповідно до технічного завдання. У разі великих проектів на розробку стендів налагодження може йти навіть більше часу, ніж на написання самої програми. Виявлення помилок цьому етапі дуже цінне. У разі виявлення на більш пізніх етапах всі попередні етапи внутрішнього плану доводиться повторювати заново.

У процесі верифікації мають бути перевірені як мінімум два аспекти роботи схеми – має бути виконана функціональна верифікація та перевірка часових співвідношень.

При функціональній верифікації логіка роботи схеми вивчається незалежно від часових співвідношень – затримка у вентилях та інші часові параметри вважаються рівними нулю. При перевірці часових співвідношень робота схеми досліджується з урахуванням передбачуваних затримок – на цьому етапі перевіряють, чи задовольняються вимоги щодо часу встановлення сигналів та їх утримування, як, наприклад, у разі послідовних пристроїв типу тригерів.

Далі, якщо програма створюється для програмованих пристроїв (типу FPGA , CPLD), розробка на цьому закінчується. Якщо схема має бути реалізована «в кристалі», то виконуються етапи т.зв. «внутрішнього плану», які складаються із синтезу схеми, компонування, розміщення та розведення та, зрештою, перевірки часових співвідношень.

Структура модуля мовою Verilog

Основною структурною одиницею пристрою, що описується за допомогою мови опису апаратури, є модуль. Всі модулі є незалежними об'єктами рівня проекту і не можуть включати в себе опис інших модулів. Але модуль може містити в собі посилання на інші модулі.

Для розробки великих програм зручно розміщувати модулі у різних файлах у межах одного проекту.

Структура модуля мовою Verilog має вигляд, як показано на рис. 3.

Рис. 2. Структура модуля мовою Verilog .

Загалом модуль має таку структуру:

```
module Ім'я_Модуля(Інтерфейс_модуля) ;
  Опис_Інтерфейсу
  …
  Внутрішня_реалізація_модуля
  …
endmodule
```

module , *endmodule* – зарезервовані слова, що означають початок і кінець модуля (відкривають та завершують опис модуля),
Ім'я_Модуля – власне ім'я модуля, тобто. ідентифікатор, за яким можна буде звернутися до цього модуля з інших частин програми,
Інтерфейс_модуля – перерахування вхідних та вихідних сигналів модуля. В інтерфейсі сигнали тільки перераховуються, при цьому вказівки типу та напрямку (вхідний або вихідний) в інтерфейсі немає.
Опис_Інтерфейсу – перераховується тип та напрями сигналів, описаних в інтерфейсі модуля.
Внутрішня_реалізація_модуля – це послідовність операторів мови Verilog, яка описує поведінку цього модуля.

Наведемо приклад найпростішого модуля, написаного мовою Verilog .

```
module F ( A1, A2, B1, B2 ) ;
  input A1, A2;
  output B1, B2;

  assign B1 = A1 & A2;
  assign B2 = !(A1 & A2);

endmodule
```

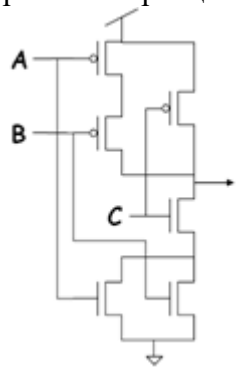
У цьому прикладі:
F – це ім'я модуля,
A1, A2, B1, B2 – інтерфейс модуля, тобто. список вхідних/вихідних сигналів,
input A1, A2; – Опис вхідних сигналів,
output B1, B2; – Опис вихідних сигналів,
два наступні рядки, що починаються з ключового слова « *assign* » - є внутрішня реалізація модуля,
endmodule – інформує про закінчення опису модуля *F* .
Пізніші редакції стандарту мови допускають опис модулів у стилі, подібному до опису функцій мовою С:

```
module F (
  input A1,
  input A2,
  output B1,
  output B2
) ;

  assign B1 = A1 & A2;
  assign B2 = !(A1 & A2);

endmodule
```

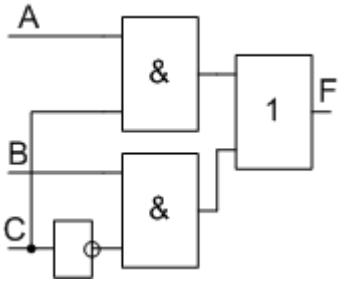
Рівні абстракції під час проектування цифрових схем

При проектуванні пристроїв з урахуванням мікросхем програмованої логіки використовуються кілька рівнів абстракції:

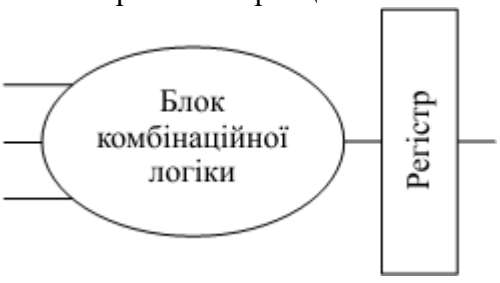
1. Рівень структурної абстракції:
   a. Рівень ключів (Switch level) – це найнижчий рівень абстракції, що оперує на рівні окремих транзисторів. На цьому рівні абстракції проблематично синтезувати логічні схеми, натомість він корисний для розуміння фізичних процесів, що протікають у цифрових колах.
   Приклад моделі транзисторного рівня абстракції:



   b. Вентильний рівень (Gate level) – цьому рівні формування цифрових пристроїв відбувається з допомогою з'єднання стандартних логічних елементів (наприклад, І, АБО і Виключне АБО тощо.).
   Приклад моделі вентильного рівня абстракції:



2. Рівень функціональної абстракції:
   a. Рівень регістрових передач (Register transfer level) – є комбінацією всіх інших рівнів абстракції.
   b. Поведінковий рівень (Behavioral level) – найуживаніший рівень опису цифрових пристроїв. На цьому рівні модель описується як алгоритм роботи пристрою, без деталізації його апаратної реалізації . Розробка моделей на цьому рівні подібна до традиційного програмування мовою Сі.
   Приклад моделі поведінкового рівня абстракції:



Способи опису поведінки схеми

Процесна форма опису поведінки

При проектуванні якогось пристрою його можна розглядати як деяку «чорну скриньку», зв'язок між вхідними та вихідними сигналами якої описується деякою таблицею. У цьому випадку поведінку такої схеми можна запрограмувати безпосередньо використовуючи оператор CASE.

Розглянемо приклад процесного опису схеми з використання v оператора CASE.

Входи		Виходи	
A1	A2	B1	B2
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

```
module F (A1, A2, B1, B2);
input wire A1, A2;
output reg B1, B2;

    always @ (A1, A2)
          begin
                case ({A1, A2})
                2'b11: begin B1 = 1; B2 = 0; end
                default: begin B1 = 0; B2 = 1; end
                endcase
          end

endmodule
```

Потоковий опис поведінки

Потокова форма є короткою формою запису процесів з одним змінним сигналом.

Потоковий стиль опису апаратури передбачає представлення процесу роботи об'єкта як послідовності паралельних операторів перетворення інформації на регістрах.

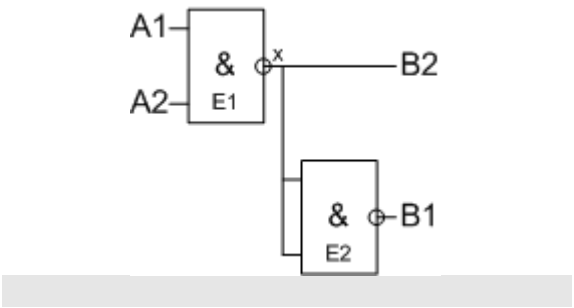
Наведений вище приклад у потоковій формі матиме вигляд:

```
module F ( A1, A2, B1, B2 ) ;
  input A1, A2;
  output B1, B2;

    assign B1 = (({A1, A2}) == 2`b11)? 1:0;
    assign B2 = (({A1, A2}) == 2`b11)? 0:1;

endmodule
```

Структурний опис поведінки

Структурний опис архітектури представляє структуру об'єкта як композицію з компонентів, з'єднаних між собою і сигналами. Функції, реалізовані компонентами, не вказуються. Структурний опис включає імена та типи компонентів, з яких складається схема, та їх зв'язки.

Розглянемо приклад структурного опису схеми:




```
module F ( A1, A2, B1, B2 ) ;
  input A1, A2;
  output B1, B2;
  wire x;

    INE E1(A1, A2, x );       // E1
    INE E2(x, x, B1); // E2
    assign B2 = x;

endmodule
```

У цьому прикладі *E1* і *E2* – це раніше підготовлений модуль *INE* , що реалізує операцію І-НЕ .

Створення стендів налагодження (test benches)

Верифікація проекту (модуля) – це доказ того, що проект (модуль) функціонує відповідно до його специфікації.

Розрізняють 2 основні способи верифікації – це імітаційна та формальна верифікація.

Імітаційна верифікація передбачає моделювання наявного Verilog-опис проектованого об'єкта і відтворення в ході імітаційного експерименту можливих зовнішніх вхідних впливів і внутрішніх станів системи, що моделюється.

Формальна верифікація передбачає використання Verilog-опису як формальної моделі з метою доведення її коректності та еквівалентності чи нееквівалентності специфікації проекту.

Для перевірки правильності функціонування проектованих систем переважно використовуються такі методи тестування:

1. Детерміністичний тест (deterministic) – перевіряє звичайні режими роботи системи, а також функціонування в граничних умовах за допомогою фіксованого набору вхідних дій.
2. Метод випадкового тестування (random) – перевіряє роботу пристрою за допомогою генератора випадкових зовнішніх дій.
3. Метод транзакційного тестування (transaction level) – перевіряє інтерфейси блоків системи, а також перевіряє проходження пакетів даних через ці блоки.

При верифікації об'єкта його можна розглядати як:

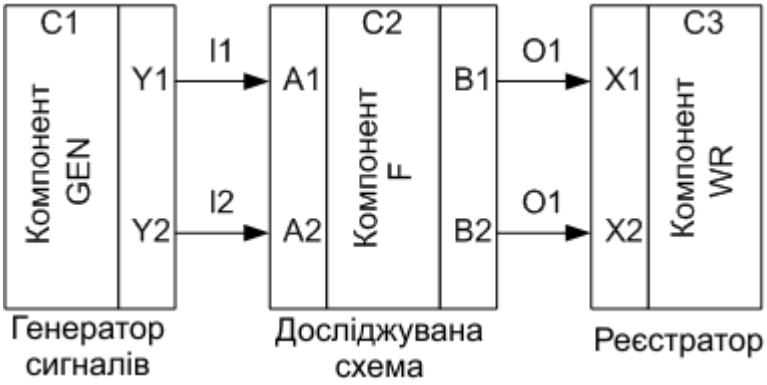
1. «Чорний ящик» – при написанні перевірочних стендів не використовується жодної інформації про внутрішній пристрій об'єкта, що перевіряється.
2. "Сірий ящик" – при написанні перевірочних стендів інформація про внутрішній устрій об'єкта, що перевіряється, частково використовується.
3. «Прозорий ящик» – перевірочні тести базуються на інформації про внутрішній устрій об'єкта, що перевіряється.

Розглянемо приклад імітаційного налагоджувального стенду для прикладу, наведеного вище. Нехай є об'єкт, який описується таблицею та описаний за допомогою CASE -оператора:

Входи		Виходи	
A1	A2	B1	B2
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

```
module F (A1, A2, B1, B2);
input wire A1, A2;
output reg B1, B2;

    always @ (A1, A2)
          begin
                case ({A1, A2})
                2'b11: begin B1 = 1; B2 = 0; end
                default: begin B1 = 0; B2 = 1; end
                endcase
          end

endmodule
```

Структурна схема налагоджувального модуля (TB _ F) матиме такий вигляд:



У цій схемі генератор сигналів моделюватиме зовнішнє середовище, а реєстратор – модель спостерігача.

Щоб знайти всі можливі помилки схеми генератор повинен генерувати повний набір вхідних даних, тобто він повинен генерувати всі можливі не заборонені комбінації вхідних сигналів. На практиці висувається ще одна вимога до вхідного генератора – час тестування має бути прийнятним.

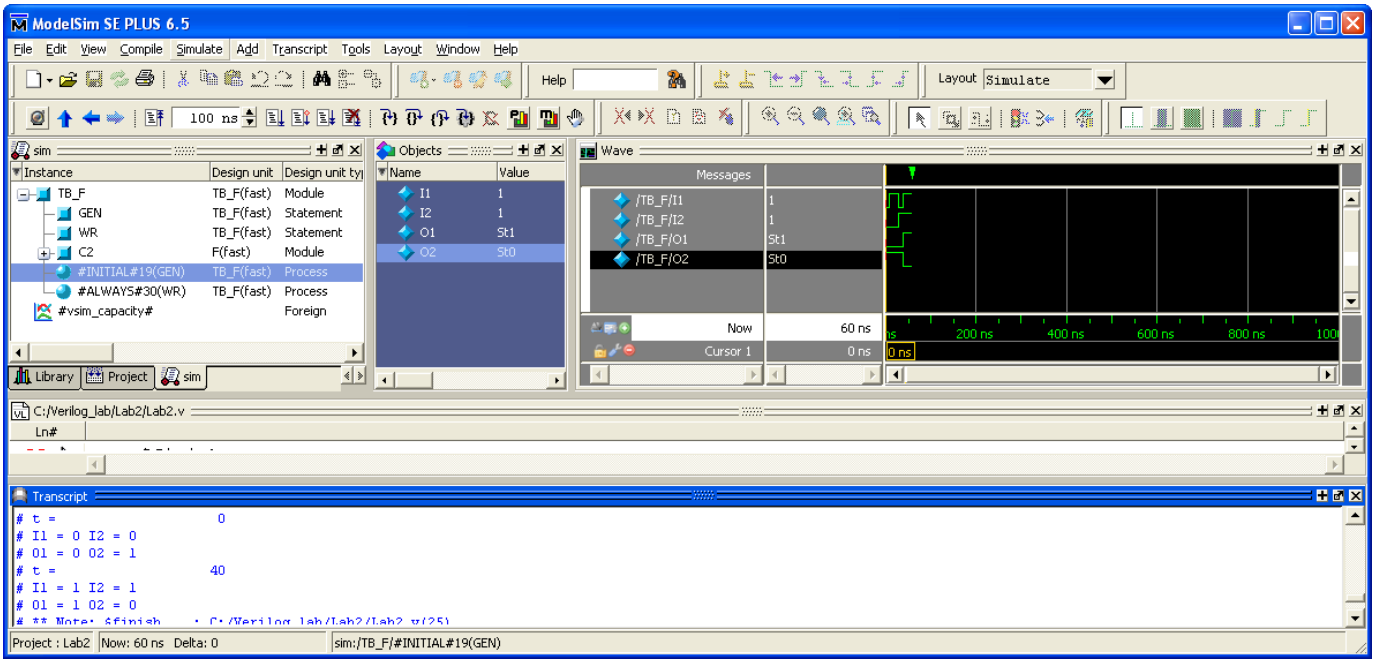
Текст налагоджувального стенду наведено нижче:

```
module TB_F ();
reg I1, I2;
wire O1, O2;

initial begin: GEN
I1 = 0; I2 = 0;
#10 I1 = 1;
#20 I1 = 0; I2 = 1;
#10 I1 = 1;
#20
$finish;
end // GEN

F C2(I1, I2, O1, O2);

always @(O1 or O2) begin: WR
$display("t = %t", $time);
$display("I1 = % b I2 = % b", I1, I2);
$display("O1 = %b O2 = %b", O1, O2);
end // WR

endmodule
```

У даному модулі налагодження генератор сигналів починається з рядка «initial begin: GEN» і закінчується рядком «end // GEN».

Реєстратор починається з рядка "always @ (O1 or O2) begin: WR" і закінчується рядком "end // WR".

Результати його моделювання наведені нижче:



Завдання для виконання

1. За таблицею істинності (згідно з варіантом) скласти Verilog –модель:
   a. Вхідними сигналами є x1, x2, x3 і x4;
   b. Вихідними сигналами є y1, y2, y3;
   c. Використовувати процесну форму опису поведінки модуля.
   d. Реалізувати функції з урахуванням можливості інверсування.
2. Написати перевірочний стенд (test bench) для верифікації, який перебирав би всі можливі комбінації вхідних сигналів.
3. Провести моделювання та переконається у працездатності написаного модуля та його перевірочного стенду.

Варіанти завдання:

1. Варіант 1:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 0 0
0 0 1 1	0 0 0
0 1 0 0	0 0 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 0 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 0 0

2. Варіант 2:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	1 0 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 0
0 1 1 1	1 0 0
1 0 0 0	1 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 1
1 1 1 0	1 0 1
1 1 1 1	1 1 1

3. Варіант 3:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	1 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 0 0
1 0 0 1	0 1 1
1 0 1 0	0 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 0
1 1 0 1	1 0 0
1 1 1 0	0 0 1
1 1 1 1	1 1 0

4. Варіант 4:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	1 0 1
0 0 1 0	1 0 0
0 0 1 1	0 0 0
0 1 0 0	1 0 1
0 1 0 1	0 1 1
0 1 1 0	1 0 1
0 1 1 1	0 0 0
1 0 0 0	0 0 1
1 0 0 1	0 0 1
1 0 1 0	1 0 1
1 0 1 1	1 0 1
1 1 0 0	0 0 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	0 0 0

5. Варіант 5:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 0
0 0 1 0	0 1 0
0 0 1 1	1 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	1 1 0
1 0 0 1	1 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 1
1 1 0 1	0 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

6. Варіант 6:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 0 0
0 0 1 1	0 0 0
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	0 0 0
1 0 1 1	0 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	0 1 0

7. Варіант 7:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 0
0 0 1 0	1 0 0
0 0 1 1	0 0 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	1 0 0
0 1 1 1	1 0 0
1 0 0 0	1 0 0
1 0 0 1	1 1 0
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 0 0
1 1 0 1	1 0 0
1 1 1 0	0 0 1
1 1 1 1	1 0 0

8. Варіант 8:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 1
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 1
0 1 1 0	0 1 1
0 1 1 1	1 0 1
1 0 0 0	0 1 1
1 0 0 1	0 1 1
1 0 1 0	1 0 1
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 1
1 1 1 0	1 0 1
1 1 1 1	1 1 1

9. Варіант 9:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 1 1
0 0 1 0	0 1 1
0 0 1 1	0 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	0 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	0 0 0
1 0 1 1	0 1 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	0 0 1
1 1 1 1	1 1 0

10. Варіант 10:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	1 1 1
0 0 1 1	0 1 1
0 1 0 0	0 0 0
0 1 0 1	1 0 0
0 1 1 0	1 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 1

11. Варіант 11:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 0
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 1
1 0 1 1	1 1 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 1

12. Варіант 12:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	1 1 1
0 0 1 0	1 1 1
0 0 1 1	0 1 1
0 1 0 0	1 0 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 1
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 1 0
1 0 1 1	1 1 1
1 1 0 0	1 1 1
1 1 0 1	1 0 1
1 1 1 0	1 0 1
1 1 1 1	1 1 1

13. Варіант 13:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 1 0
0 0 0 1	0 1 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	0 1 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	0 1 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	0 1 0
1 0 1 1	0 1 1
1 1 0 0	0 1 1
1 1 0 1	0 1 0
1 1 1 0	0 1 1
1 1 1 1	0 1 0

14. Варіант 14:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 0
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 1
0 1 0 1	0 0 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 0 0
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 0
1 1 0 1	1 0 0
1 1 1 0	0 0 1
1 1 1 1	1 1 0

15. Варіант 15:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 0 0
0 0 1 1	0 0 1
0 1 0 0	1 0 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 1 0
1 0 0 0	1 0 0
1 0 0 1	1 0 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 1
1 1 0 1	1 1 0
1 1 1 0	1 1 1
1 1 1 1	1 1 1

16. Варіант 16:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 1
0 1 0 0	1 1 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

17. Варіант 17:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

18. Варіант 18:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 0
0 1 0 1	1 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 1 1
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

19. Варіант 19:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 1 1
0 0 1 0	0 1 0
0 0 1 1	1 0 0
0 1 0 0	0 1 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	1 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 1 0
1 1 1 0	1 0 1
1 1 1 1	1 0 0

20. Варіант 20:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	1 1 1
0 0 1 1	1 0 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 0
0 1 1 1	1 0 0
1 0 0 0	1 1 0
1 0 0 1	1 1 1
1 0 1 0	1 0 0
1 0 1 1	1 1 1
1 1 0 0	0 1 1
1 1 0 1	1 0 1
1 1 1 0	1 0 1
1 1 1 1	1 1 1

21. Варіант 21:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	1 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 0 0
1 0 0 1	0 1 1
1 0 1 0	0 0 0
1 0 1 1	1 0 1
1 1 0 0	0 1 0
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

22. Варіант 22:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 1 0
0 0 0 1	1 0 1
0 0 1 0	1 0 0
0 0 1 1	0 0 0
0 1 0 0	1 0 1
0 1 0 1	0 1 1
0 1 1 0	1 0 1
0 1 1 1	0 0 0
1 0 0 0	1 0 1
1 0 0 1	1 0 1
1 0 1 0	1 1 1
1 0 1 1	1 0 1
1 1 0 0	0 0 1
1 1 0 1	1 1 0
1 1 1 0	1 1 1
1 1 1 1	0 0 0

23. Варіант 23:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	1 0 0
0 0 1 0	0 1 1
0 0 1 1	1 1 1
0 1 0 0	1 0 0
0 1 0 1	0 1 0
0 1 1 0	1 1 1
0 1 1 1	1 0 0
1 0 0 0	1 1 0
1 0 0 1	1 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 1
1 1 0 1	0 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

24. Варіант 24:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 0 1
0 0 1 1	0 0 0
0 1 0 0	0 0 0
0 1 0 1	0 1 1
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	0 1 1
1 0 1 1	0 0 1
1 1 0 0	0 1 1
1 1 0 1	1 1 0
1 1 1 0	1 0 1
1 1 1 1	0 1 0

25. Варіант 25:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	1 0 0
0 0 1 0	1 0 0
0 0 1 1	0 0 1
0 1 0 0	1 0 0
0 1 0 1	0 1 0
0 1 1 0	1 1 0
0 1 1 1	1 0 0
1 0 0 0	1 0 1
1 0 0 1	1 1 0
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 0 0
1 1 0 1	1 0 1
1 1 1 0	0 0 1
1 1 1 1	1 0 0

26. Варіант 26:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 1
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	0 0 0
0 1 0 1	0 1 1
0 1 1 0	0 1 1
0 1 1 1	1 0 1
1 0 0 0	0 1 1
1 0 0 1	0 1 1
1 0 1 0	1 0 1
1 0 1 1	1 0 1
1 1 0 0	0 1 1
1 1 0 1	1 0 1
1 1 1 0	1 0 1
1 1 1 1	1 1 1

27. Варіант 27:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 1 0
0 0 0 1	0 1 1
0 0 1 0	0 1 1
0 0 1 1	0 1 1
0 1 0 0	1 0 0
0 1 0 1	0 1 0
0 1 1 0	1 1 1
0 1 1 1	0 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	0 1 0
1 0 1 1	0 1 1
1 1 0 0	0 1 0
1 1 0 1	1 0 0
1 1 1 0	1 1 1
1 1 1 1	1 1 0

28. Варіант 28:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	1 1 1
0 0 1 1	1 1 1
0 1 0 0	0 0 0
0 1 0 1	1 0 0
0 1 1 0	1 1 1
0 1 1 1	1 0 0
1 0 0 0	1 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 1
1 1 0 1	1 0 0
1 1 1 0	1 1 1
1 1 1 1	1 1 1

29. Варіант 29:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	0 0 0
0 0 1 0	1 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 0
0 1 0 1	0 1 0
0 1 1 0	1 1 1
0 1 1 1	1 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 1 1
1 1 0 0	0 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 1

30. Варіант 30:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	1 1 1
0 0 1 0	1 1 1
0 0 1 1	1 1 1
0 1 0 0	1 0 1
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 1
1 0 0 0	1 1 0
1 0 0 1	0 1 1
1 0 1 0	1 1 0
1 0 1 1	1 1 1
1 1 0 0	1 1 1
1 1 0 1	1 0 1
1 1 1 0	1 1 1
1 1 1 1	1 1 1

31. Варіант 31:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 1 0
0 0 0 1	0 1 1
0 0 1 0	0 1 0
0 0 1 1	0 0 1
0 1 0 0	0 1 1
0 1 0 1	1 1 0
0 1 1 0	0 1 1
0 1 1 1	0 1 0
1 0 0 0	0 1 0
1 0 0 1	1 1 1
1 0 1 0	0 1 1
1 0 1 1	0 1 1
1 1 0 0	0 1 1
1 1 0 1	0 0 0
1 1 1 0	0 1 1
1 1 1 1	0 1 0

32. Варіант 32:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	1 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 1
0 1 0 1	0 0 1
0 1 1 0	0 1 1
0 1 1 1	1 1 0
1 0 0 0	1 1 0
1 0 0 1	0 0 0
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 0
1 1 0 1	1 0 1
1 1 1 0	0 0 1
1 1 1 1	1 1 0

33. Варіант 33:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	0 0 0
0 0 1 1	1 0 0
0 1 0 0	1 0 1
0 1 0 1	1 1 0
0 1 1 0	0 1 1
0 1 1 1	1 1 0
1 0 0 0	1 0 0
1 0 0 1	1 0 0
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 0
1 1 0 1	1 1 0
1 1 1 0	1 1 1
1 1 1 1	1 1 1

34. Варіант 34:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 1
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 1
0 1 0 1	1 1 1
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	1 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 0 1
1 1 0 0	1 1 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

35. Варіант 35:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 0
0 0 1 0	0 1 0
0 0 1 1	0 1 1
0 1 0 0	1 0 0
0 1 0 1	0 1 0
0 1 1 0	0 1 0
0 1 1 1	0 0 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	0 0 0
1 1 0 0	0 1 1
1 1 0 1	0 0 0
1 1 1 0	1 0 0
1 1 1 1	1 1 0

36. Варіант 36:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 1 0
0 0 0 1	0 0 1
0 0 1 0	0 1 0
0 0 1 1	0 1 0
0 1 0 0	1 1 0
0 1 0 1	1 1 0
0 1 1 0	0 1 1
0 1 1 1	1 1 0
1 0 0 0	0 1 0
1 0 0 1	0 1 1
1 0 1 0	1 1 0
1 0 1 1	1 0 0
1 1 0 0	0 1 1
1 1 0 1	1 1 0
1 1 1 0	1 1 1
1 1 1 1	0 1 0

37. Варіант 37:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 0 1
0 0 1 0	1 0 1
0 0 1 1	0 1 0
0 1 0 0	1 0 1
0 1 0 1	0 1 0
0 1 1 0	1 1 1
0 1 1 1	1 0 0
1 0 0 0	0 0 0
1 0 0 1	0 1 1
1 0 1 0	1 0 0
1 0 1 1	1 1 1
1 1 0 0	0 1 1
1 1 0 1	1 0 1
1 1 1 0	1 0 1
1 1 1 1	1 0 0

38. Варіант 38:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	0 0 0
0 0 0 1	0 1 0
0 0 1 0	1 1 0
0 0 1 1	1 1 1
0 1 0 0	0 0 0
0 1 0 1	1 1 0
0 1 1 0	0 1 0
0 1 1 1	1 1 0
1 0 0 0	1 1 0
1 0 0 1	0 1 1
1 0 1 0	1 1 0
1 0 1 1	1 0 1
1 1 0 0	0 1 0
1 1 0 1	1 0 1
1 1 1 0	1 0 0
1 1 1 1	1 1 1

39. Варіант 39:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	1 1 1
0 0 1 0	1 1 0
0 0 1 1	1 1 1
0 1 0 0	0 1 0
0 1 0 1	0 1 0
0 1 1 0	0 1 1
0 1 1 1	1 0 0
1 0 0 0	1 0 0
1 0 0 1	0 1 1
1 0 1 0	0 1 0
1 0 1 1	1 0 1
1 1 0 0	1 1 0
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	1 1 0

40. Варіант 40:

x1 x2 x3 x4	y1 y2 y3
0 0 0 0	1 0 0
0 0 0 1	1 0 1
0 0 1 0	1 0 1
0 0 1 1	1 0 0
0 1 0 0	1 0 1
0 1 0 1	0 1 1
0 1 1 0	1 0 1
0 1 1 1	0 0 0
1 0 0 0	0 0 1
1 0 0 1	0 0 1
1 0 1 0	1 0 1
1 0 1 1	1 1 1
1 1 0 0	0 0 1
1 1 0 1	1 0 0
1 1 1 0	1 0 1
1 1 1 1	0 0 0

Вимоги до оформлення звіту

1. Протокол оформляється кожним студентом групи окремо.
2. Протокол повинен містити:
    a. Титульна сторінка.
    b. Завдання згідно з варіантом.
    c. Лістинги модуля та перевірочного стенду до нього.
    d. Результати моделювання – результати виведення на екран та часові діаграми.
    e. Висновки про виконану роботу.
3. Захист роботи проводиться кожним студентом персонально.

Контрольні питання

1. Назвіть основні етапи проектування мовою опису апаратури.
2. Розкажіть структуру модуля, написаного мовою Verilog .
3. Які методи верифікації апаратури вам відомі?
4. Які методи тестування апаратури вам відомі?
5. Які підходи до розгляду об'єкта, що верифікується, використовуються?

Література

1. Ю.П. Кондратенко, В.В, Мохор, С.А. Сидоренко. VERILOG - HDL для моделювання та синтезу цифрових електронних схем.
2. А.К. Поляков. Мови VHDL та VERILOG у проектуванні цифрової апаратури.