

Ім'я користувача:
Volodymyr Donchenko

Дата перевірки:
26.12.2023 10:42:53 EET

Дата звіту:
26.12.2023 10:44:01 EET

ID перевірки:
1016036064

Тип перевірки:
Doc vs Internet

ID користувача:
100012947

Назва документа: Сосєдко Віктор Юрійович (1)

Кількість сторінок: 53 Кількість слів: 7455 Кількість символів: 56559 Розмір файлу: 2.18 MB ID файлу: 1015728563

4.9% Схожість

Найбільша схожість: 1.82% з Інтернет-джерелом (http://homes.esat.kuleuven.be/~mknezevi/documents/sysgen_intro.pdf).

4.9% Джерела з Інтернету

17

Сторінка 55

Пошук збігів з Бібліотекою не проводився

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 9 слів та 0%)

0% Вилучення з Інтернету

1

Сторінка 56

Немає вилучених бібліотечних джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи

1

ВСТУП

Вузькими місцями традиційного підходу до проєктування електронних систем є етапи складання технічного завдання і специфікацій. В даний час широко використовується ітеративний підхід до розробки - цикл Демінга "Plan-Do-Check-Act" (планування, виконання, перевірка, вплив) [1]. При такому підході класичні текстові документи незручні і здатні ускладнювати процес розробки.

Проєктування прототипів і проведення експериментів на ранніх етапах виявляються незручними, складними і витратними. Під час реалізації вручну розроблені програмні рішення та людські помилки роблять процес малозадовільним з точки зору надійності. У той же час традиційне тестування призводить до виявлення помилок лише на пізніших етапах розробки програмного забезпечення[2].

Для вирішення зазначених складнощів використовується інноваційна технологія "модельно-орієнтоване проєктування" (МОП).

Суть даної технології полягає в тому, що спочатку створюється імітаційна модель об'єкта управління в спеціалізованому програмному пакеті. Розроблюваний алгоритм управління відпрацьовується на імітаційній моделі в режимі симуляції (вибирається тип регулятора, налаштовуються коефіцієнти і т. п.).

Модель може досить добре описувати динаміку об'єкта, але може бути і приблизною, яка описує тільки основні рухи і характеристики об'єкта. Найчастіше таке моделювання проводять в середовищі Matlab Simulink компанії Mathworks. У разі досягнення алгоритмом поставленого завдання управління проводиться експорт алгоритму управління в програмний код цільової платформи.

Таким чином, замість фізичних прототипів і текстових специфікацій в модельно-орієнтованому проєктуванні застосовується імітаційна модель, яка й використовується у всіх етапах розробки.

При такому підході можна розробляти і проводити імітаційне моделювання як для всієї системи цілком, так і для її компонентів.

Модельно-орієнтоване проектування – ефективний і економічно вигідний спосіб розробки систем управління, систем обробки сигналів, побудови систем зв'язку, розробок в області робототехніки і створення вбудованих (embedded) систем.

Застосування цього підходу в компаніях ABB, Boeing, Bell Helicopter, Toyota, Tesla, General Motors, Ford, Hitachi, Hyundai, Realtek, Yokogawa та інших дозволило збільшити якість продукції та зменшити час розробки більш ніж в 2 рази [3].

Система численних інженерних розрахунків Matlab є de facto стандартом в області інженерного чисельного моделювання, а Simulink - стандартом в області швидкого прототипування технічних систем, оскільки підтримує в зручній формі (у вигляді графічної мови) створення систем будь-якої складності шляхом їх розбиття на підсистеми (декомпозицію).

Ключовою особливістю середовища Matlab є автоматична генерація C-коду який переноситься для розроблюваних моделей, в тому числі і під широкий діапазон обчислювальних платформ, контролерів і цифрових сигнальних процесорів. Генерований системою код підпорядковується стандартам ANSI C99, які підтримують більшість сучасних компіляторів. Генерація коду можлива практично під всі поширені архітектури мікроконтролерів і мікропроцесорів, що робить проект, що розробляється незалежним від обраної архітектури та дозволяє легко переносити код на інші платформи.

Можлива генерація коду на мові C, на мові ПЛК (програмованого логічного контролера), або на мові опису апаратури HDL (Hardware Description Language) для створення ПЛІС (програмованої логічної інтегральної схеми) в тому числі і для ПЛІС фірми Xilinx.

Однак широкому впровадженню даного підходу перешкоджає відсутність розроблених маршрутів проєктування і швидкого прототипування технічних систем для конкретних реалізацій.

Таким чином, розробка маршрутів модельно-орієнтованого проєктування для реалізації на цільових пристроях фірми Xilinx з використанням додатку MatLab/Simulink актуальна.

Об'єкт дослідження – проєктування кінцевих автоматів в базисі ПЛІС Xilinx.

Предмет дослідження – прискорення проєктування кінцевих автоматів в базисі ПЛІС Xilinx на основі використання MatLab/Simulink.

Мета дослідження - створення кінцевого автомата за допомогою System Generator MATLAB/Simulink на базі програмованих логічних схем Xilinx.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати етапи схемотехнічного проєктування цифрових пристроїв, оцінити рівні абстракції і специфіку проєктування цифрових пристроїв для FPGA з використанням засобів фірми Xilinx;
- розглянути етапи модельно-орієнтованого проєктування в MATLAB/Simulink і основні блоки System Generator;
- розглянути маршрути проєктування цифрових автоматів в MATLAB/Simulink з використанням System Generator;
- Створення кінцевого автомата за допомогою System Generator MATLAB/Simulink на базі програмованих логічних схем Xilinx.

Методи дослідження - експериментальна розробка кінцевого автомата за допомогою System Generator MATLAB/Simulink на базі програмованих логічних схем Xilinx Spartan 3E Starter Board.

РОЗДІЛ 1. АНАЛІЗ ОСНОВНИХ ЕТАПІВ СХЕМОТЕХНІЧНОГО ПРОЄКТУВАННЯ У БАЗИСІ XILINX

1.1. Основні етапи схемотехнічного проєктування цифрових пристроїв

Процес автоматизованого схемотехнічного проєктування радіoeлектронних пристроїв складається з кількох етапів (див. рис.1.1).

На початковому етапі (1) складного пристрою проводиться розбиття на функціонально завершені блоки, для кожного з яких створюються індивідуальні технічні завдання (ТЗ). Ці ТЗ містять опис зовнішніх і внутрішніх параметрів, таких як вхідні та вихідні сигнали, діапазон частот, споживана потужність, умови експлуатації, граничні допуски на основні характеристики тощо. На цьому етапі значною мірою визначається особистістю розробника-конструктора: його знаннями, інтуїцією, інтелектом і широким кругозором.

На другому етапі (2), після сформулювання ТЗ для розглянутого блоку, його принципова електрична схема формується у вигляді початкового (нульового) наближення. Зазвичай це виконується розробником, враховуючи його власний досвід і досвід попередніх розробок. На цьому етапі також обираються компоненти схеми, такі як транзистори, діоди, інтегральні мікросхеми, резистори, конденсатори, котушки індуктивності та інші, а також визначаються номінальні значення та допуски на параметри цих компонентів.

У подальшому, на третьому етапі (3), визначається система автоматизованого схемотехнічного проєктування (моделювання), і вибирається програма, яка оптимально підходить для аналізу заданої електронної схеми та дозволяє оцінити відповідність ТЗ обраній схемі. Часом для здійснення необхідних обчислень у вибраному пакеті використовується не одна, а група програм, таких як аналіз постійного струму, аналіз у часовій та частотній областях.

Після цього принципова схема задуманого блоку підготовлюється для комп'ютерного аналізу та вводиться до ПЕОМ, використовуючи текстовий або графічний метод (етап 4).

Рис. 1.1. Основні етапи автоматизованого проєктування електронних схем

Далі, використовуючи вбудовану бібліотеку моделей компонентів, автоматично генерується математична модель аналізованого пристрою, відповідно до введеної принципової схеми (етап 5). На етапі 6 здійснюється аналіз математичної моделі електронної схеми у режимі діалогу. Наприклад, для аналізу схеми аналогового пристрою проводяться такі обчислення:

- розрахунок схеми для постійного струму;
- розрахунок схеми у частотній області, такий як обчислення амплітудно-частотної характеристики та фазово-частотної характеристики, а також аналіз спектральної щільності шуму;
- розрахунок у часовій області, включаючи визначення перехідних та імпульсних характеристик і проведення спектрального аналізу.

Отримані характеристики порівнюються з вимогами ТЗ та (або) результатами експериментів з макетом (етапи 7 та 8). На основі цього порівняння приймається рішення щодо прийняття чи відхилення розглянутого варіанта проєкту (етап 9). Таке рішення здійснюється неформально, оскільки іноді інженерне розуміння суті дозволяє ігнорувати деяку розбіжність між результатами комп'ютерного аналізу та ТЗ.

У випадку, якщо характеристики незадовільні, принципова схема та (або) моделі компонентів підлягають зміні (етап 10).

Після прийняття проєкту розробляється технічна документація для виготовлення пристрою та проведення випробувань (етап 11).

Цей аналітичний цикл повторюється знову. Зокрема, під час виконання розрахунків для різних варіантів, комп'ютерні програми аналізу електронних схем стають особливо корисними, оскільки вони дозволяють автоматизовано провести аналіз багатьох варіацій за короткий період часу. Модифікацію схеми також можна здійснювати за допомогою спеціальних програм оптимізації на ПЕОМ (етап 12), які використовують методи оптимізації проектних рішень, ґрунтовані на розв'язанні математичних задач (лінійного та нелінійного) програмування. У цих задачах виконується пошук мінімуму або максимуму конкретної цільової функції, яка залежить від багатьох змінних при наявності обмежень на ці змінні. При проєктуванні РЕУ ця цільова функція відображає якість роботи, вартість апаратури та інші характеристики, що залежать від параметрів компонентів. Оптиміальні значення цих параметрів визначаються шляхом вирішення поставленої задачі

оптимізації. Обмеження формуються у вигляді системи взаємозв'язків, які обмежують допустиму область змін параметрів компонентів при вирішенні задачі оптимізації РЕУ.

Після завершення оптимізації стає можливим визначення чутливості схеми, оцінка впливу розкиду параметрів компонентів та отримання інших значущих характеристик. Таким чином, у процесі цього проєктування вирішуються завдання, пов'язані з розрахунком, аналізом та оптимізацією схемних рішень.

Щодо завдання синтезу, це складне завдання, яке можна жорстко алгоритмізувати лише для певних окремих випадків, таких як класичний синтез пасивних і активних аналогових та цифрових частотних фільтрів, або синтез широкопasmових узгоджувальних пристроїв і цифрових автоматів. У більшості випадків завдання синтезу розв'язується евристичним шляхом, базуючись на попередньому досвіді та винахідництві.

Важливо відзначити, що в процесі конструювання та розробки технології може виникнути необхідність корекції схем, структури системи та навіть вихідних даних. Таким чином, процес проєктування не лише має багатоетапний характер, але й піддається повторній корекції у процесі його виконання, тобто, має ітераційний характер.

Треба відзначити, що автоматизоване проєктування електронних схем за допомогою ПЕОМ має переваги перед традиційним методом "вручну" з подальшим доведенням на фізичному макеті. Розробник може використовувати можливості ПЕОМ у кількох областях. По-перше, за допомогою прикладних програм набагато легше вивчити ефект варіювання параметрів схеми, ніж за допомогою експериментів. По-друге, можна аналізувати критичні режими пристрою без фізичного руйнування його компонентів. По-третє, програми аналізу дозволяють оцінити роботу схеми за найгіршого поєднання параметрів, що важко і не завжди можливо здійснити експериментально. По-четверте, програми дозволяють провести такі

вимірювання на моделі електронної схеми, які важко виконати експериментально в лабораторії.

1.2. Поняття рівнів абстракції та ієрархії у проектуванні апаратури

З часу першої появи цифрової мікросхеми у 1958 році розробники спостерігають експоненційне зростання числа транзисторів на інтегральній мікросхемі, що призводить до зростання проєктів та їхнього стрімкого ускладнення. Для спрощення процесу проєктування у проєктах використовуються техніки, такі як абстракція та ієрархія. Ці техніки працюють за принципом «розділяй і владарюй» і виявляються досить ефективними у рамках екстремально великих проєктів.

Функціональні можливості цифрових мікросхем можуть бути представлені за допомогою HDL різних рівнях абстракції. Найнижчим рівнем абстракції цифрових HDL є рівень транзисторних ключів, який визначає здатність описувати схему як таблиці транзисторних ключів. Вище за нього знаходиться рівень вентилів, який описує схему у вигляді таблиці з'єднань простих логічних вентилів та функцій. Обидва описані рівні абстракції можна віднести до структурного подання пристрою.

Наступний, більш складний рівень HDL визначається здатністю підтримувати логічні примітиви. Цей рівень називається функціональним чи RTL (Register Transfer Level), тобто, рівнем регістрових передач. Пристрій на цьому рівні абстракції представляється набором регістрів, пов'язаних між собою елементами комбінаційної логіки.

Найвищим рівнем абстракції вважається поведінковий, який підтримується сучасними версіями HDL і означає можливість описувати поведінку схеми, використовуючи абстрактні логічні структури, наприклад цикли і процеси. Цей рівень також передбачає використання у виразах алгоритмічних елементів, таких як суматори та помножувачі.

У рамках представлення апаратних блоків є системний рівень абстракції, що представляє набори алгоритмів у рамках великих блоків та їх з'єднання між собою. На жаль, цей підхід використовується тільки в

8

модельованні поведінки системи і поки що не придатний для розробки опису реального пристрою.

На рисунку 1.2 показано спрощене уявлення про функціонування апаратних блоків різних рівнях абстракції.

В даний час основним способом представлення апаратури є рівень RTL. Решта уявлень або є допоміжними, або згодом конвертуються в однозначний RTL-код.

Спочатку розглянемо поняття проекту в проектуванні цифрових пристроїв для FPGA. Проектом є набір файлів, що містять графічний опис блоків, описи блоків на мовах HDL, описи обмежень проекту, і т.д. Глобально можна сказати, що будь-який проєкт містить файли вихідного коду, що описують модулі та блоки проекту та ряд допоміжних файлів.

У межах сучасного підходи до проектування будь-який проєкт є ієрархічним, тобто. побудованим за модульним принципом (можна порівняти з матрьошкою). У проєкті є деякий файл, що містить модуль верхнього рівня, який містить деяку логіку роботи і включає модулі нижчого рівня, які в свою чергу здійснюють ту ж послідовність дій-тобто. містять свою внутрішню логіку та підключають модулі нижчого рівня і т.д. На рисунку 1.3. представлено спрощену структуру проекту з урахуванням ієрархії.

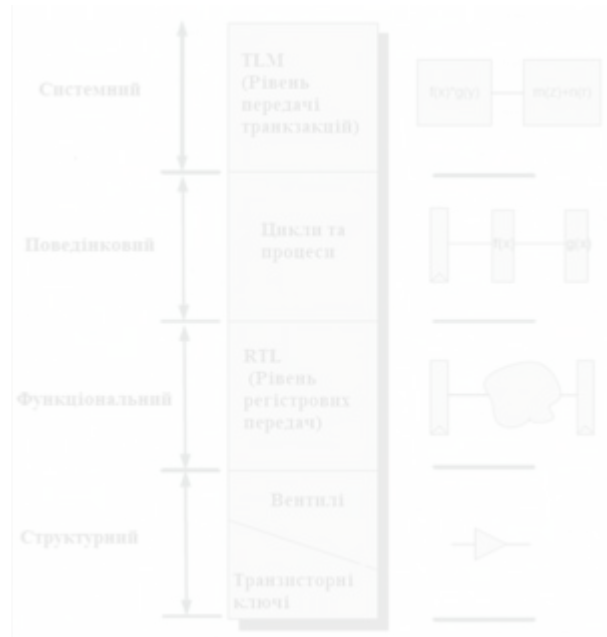


Рис. 1.2. Уявлення про функціонування апаратних блоків на різних рівнях абстракції

Спрощена схема маршруту проектування систем, заснованих на FPGA, з допомогою мов HDL лише на рівні абстракції RTL показано малюнку 1.4. У лівій частині маршруту проектування представлені етапи перетворення текстового HDL опису (RTL code) конфігурацію пристрою рівня Celllevel, яка потім завантажується в цільову FPGA (device programming). У правій частині представлений процес валідації (перевірки правильності), який використовує тестове оточення (testbench) для перевірки, чи система задовольняє технічним вимогам (RTL verification) і вимогам продуктивності (static timing analysis).

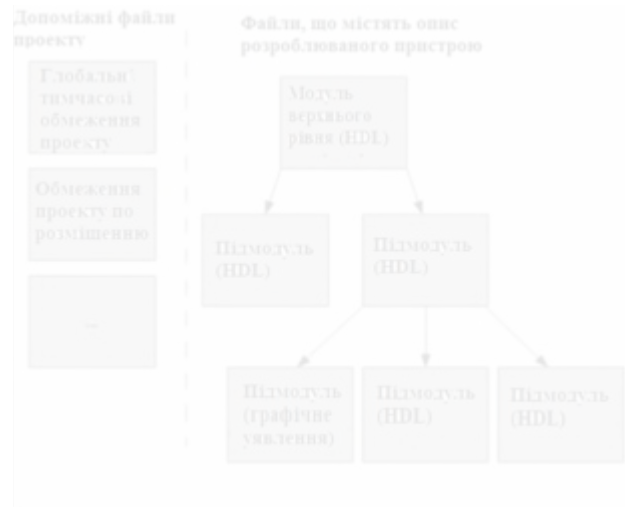


Рис. 1.3. Спрощена структура ієрархії проекту

Головні етапи у маршруті проектування:

1. Проектування системи та отримання HDL файлів. Додавання окремих файлів обмеження (constraints) визначення обмежень реалізації;
2. Створення тестового оточення (testbench) та здійснення моделювання RTL-код;
3. Виконання синтезу (synthesis) та імплементації (implementation). Процес синтезу переважно відомий як логічний синтез (logic synthesis). На цій стадії HDL файли перетворюються на списки з'єднань (netlist).

Процес імплементації складається з трьох ступенів: трансляції (translate), відображення (map), а також розміщення та розведення (place and route). Процес трансляції полягає у перекладі всіх списків з'єднань до єдиного формату та єдиного файлу, що залежить від засобів САПР. Процес, відомий як відображення, відображає універсальні вентиля в логічні осередки, специфічні для певної цільової мікросхеми FPGA.

Процес розміщення та розведення, служить для отримання фізичного розміщення (топології) пристрою, що розробляється всередині мікросхеми FPGA. Цей процес розміщує осередки у фізичній області та визначає маршрути з'єднання різних сигналів. Статичний часовий аналіз (static timing

11

analysis) виконується наприкінці процесу реалізації. Він визначає часові параметри з'єднань, такі як максимальну затримку розповсюдження сигналу та максимальну тактову частоту;

4. Створення та прошивка програмного файлу. На цьому етапі, згідно з остаточним поданням проєкту після фізичного розміщення, створюється файл конфігурації. Цей файл служить для конфігурування логічних осередків та комутаційних матриць цільової FPGA.

Також, за потреби, можна виконати два додаткові кроки: зробити функціональне моделювання (functional simulation), яке може бути виконане після синтезу, а також тимчасове моделювання (timing simulation), яке виконується після етапу імплементації.

Функціональне моделювання використовує отриманий на етапі синтезу netlist для заміни RTL опису та перевірки коректності процесу синтезу. Тимчасове моделювання використовує отримане на етапі імплементації кінцеве уявлення пристрою, що розробляється поряд з детальними тимчасовими даними, також для здійснення перевірки на коректність.



Рис. 1.4. Спрощений маршрут проєктування для FPGA

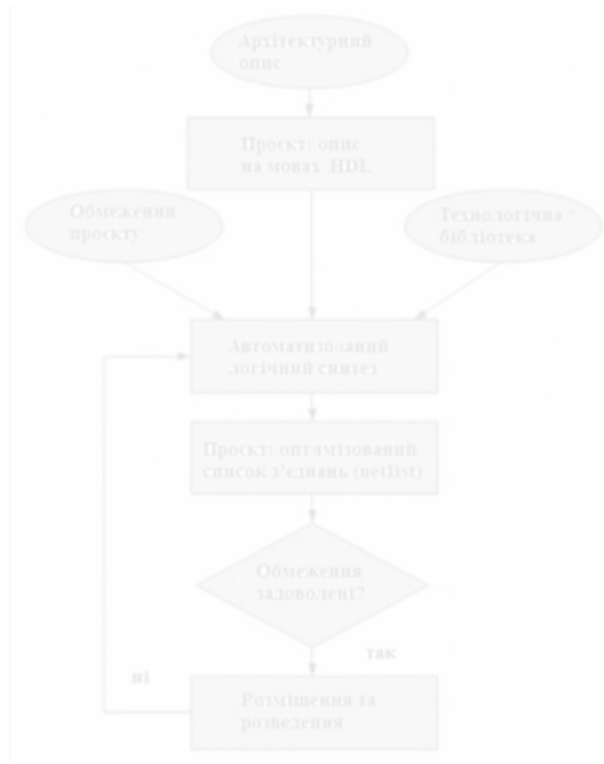
1.3. Процес синтезу

У процесі розробки проєктувальники описують поведінку апаратних блоків технологічно-незалежними високорівневими мовами HDL. Процес синтезу є процесом перекладу високорівневого опису розроблюваного

пристрою в оптимізоване представлення вентильного рівня (netlist) за деяким алгоритмом, ґрунтуючись на обмеженнях проекту та обраній технологічній бібліотеці. Подібна технологічна бібліотека містить як базові логічні вентиля, так і макро-блоки, такі як суматори, мультиплексори, пам'ять та тригери. На перших етапах розвитку цифрової електроніки, розробники здійснювали логічний синтез своїми руками, аналізуючи проєкт і переводячи модулі, що розробляються, і блоки у вентильне уявлення, спираючись на деяку технологічну бібліотеку та обмеження проєкту. На малюнку 2.5.а представлено блок-схему етапів ручного перекладу проєкту до оптимізованого подання у вигляді набору вентилів.



а)



б)

Рис.2.5. Процес логічного синтезу: а) без використання засобів автоматизованих систем проєктування; б) з використанням засобів автоматизованого проєктування

Нині процес логічного синтезу повністю автоматизованим. Це дозволяє розробникам зосередити свою увагу на проєктуванні та використовувати у розробці високорівневі мови HDL. На малюнку 2.5.б представлено блок-схему етапів проєктування з використанням автоматизованих систем проєктування.

Процес імплементації є наступним кроком після процесу синтезу маршруті проєктування. Для даного етапу можна виділити три характерні ступені:

- Трансляція (Translate);

- Відображення (Map);
- Розміщення та розведення (Place and Route).

У зв'язку з тим, що будь-який проєкт може складатися не тільки з безлічі файлів, що містять вихідний код опису модулів проєкту на мовах HDL, але і містити безліч файлів, що є списками з'єднань (netlist), то потрібен певний етап, на якому буде здійснено конвертацію всіх списків з'єднань до одного формату. Цим етапом є процес трансляції. При об'єднанні всіх списків з'єднань в єдине ціле, на цьому етапі відбувається злиття з файлами, що описують обмеження проєкту (design constraints). Також слід зазначити, що формат та структура результуючого файлу, отриманого на цьому етапі, є специфічною від фірми виробника програмного забезпечення.

На стадії відображення проєкт відображається на реально доступні ресурси, характерні для цільового кристала FPGA. У цьому процесі виявляються помилки використання недоступних чи відсутніх для цільового FPGA ресурсів.

Заключною стадією процесу імплементації проєкту є стадія розміщення та розведення. На цьому етапі кошти автоматизованого проєктування розміщують проєкт фізично. Уся логіка Проєкт розміщується на конкретних фізичних структурах цільової FPGA, крім цього між логічними блоками прокладається трасування. На цьому етапі можна виявити помилки щодо переповнення цільової FPGA – тобто. виявити потреба у використанні більшої кількості ресурсів, ніж є. Також причиною помилок можуть стати ситуації, при виникненні яких, неможливість коректно поєднати деяку ділянку логіки із загальним проєктом через брак з'єднань тощо.

Після завершення процесу імплементації та відсутності критичних попереджень та помилок, проєкт стає повністю готовим до заключного етапу, а саме до створення конфігураційного файлу для FPGA.

1.4. Специфіка проєктування цифрових пристроїв для FPGA з використанням засобів фірми Xilinx

Насамперед хотілося б представити огляд сімейств ПЛІС, що є в арсеналі цієї фірми. Кожна із сімейств, представлених нижче, має свою власну специфіку застосування та відповідні цільові ринки:

- Spartan - є масовими FPGA, що мають досить низьку вартість і призначені для вирішення завдань, що не потребують великих обчислювальних потужностей. На жаль, останнім представником даного сімейства є Spartan 6 і Xilinx планує завершити розвиток даного сімейства. На зміну йому прийдуть дві нові родини - Artix і Kintex;

- Artix — це сімейство орієнтоване на масові ринки недорогої продукції, має відносно низьку вартість та найнижче енергоспоживання порівняно з іншими сімействами, що мають однаковий порядковий номер і виконані за однаковою технологічною нормою. Також дане сімейство має малі габарити, що робить його в поєднанні з іншими параметрами дуже привабливим для побудови портативних пристроїв;

- Kintex - кристали цього типу мають великий обсяг пам'яті і розширені ресурси DSP, що робить це сімейство ідеальним для побудови LTE, світлодіодних і 3D цифрових відео дисплеїв, пристрій відображення для медицини та авіації;

- Virtex - являють собою найшвидші і ємніші FPGA, що містять найбагатшу периферію і найбільшу кількість додаткових блоків, таких як DSP-ядра, блокова пам'ять і т.д. Все це робить дану родину FPGA порівнянною з кристалами ASIC;

- CoolRunner – дані сімейства є реалізацією архітектури CPLD, що означає, що ці пристрої є невеликими енергонезалежними кристалами, що служать для організації невеликих, критичних до енергоспоживання цифрових схем.

Нові кристали з архітектурою FPGA від Xilinx спроектовані за технологією high-K metal gate та 28nm техпроцесу, що дозволяє при мінімальному енергоспоживанні досягти максимальної продуктивності.

Для покриття всього стандартного маршруту проектування для FPGA та CPLD фірмою Xilinx було розроблено два програмні пакети: ISE (Integrated Software Environment) та PlanAhead. Обидва ці пакети є закінчені середовища проектування зі зручним графічним інтерфейсом, що дозволяє викликати для різних етапів маршруту проектування спеціалізовані програми та утиліти. Вибір того чи іншого пакету є в основному справою смаку, але все ж таки слід враховувати, що пакет PlanAhead вважається більш професійним і має ряд більш розширених опцій.

На малюнку 2.6 представлено покриття всього маршруту проектування засобами Xilinx, а також на малюнку представлені розширення файлів проєкту, створених на кожному з кроків.

Написання HDL може бути виконане як у вбудованому текстовому редакторі, так і в будь-якому іншому за бажанням розробника. На етапі синтезу можна використовувати як вбудований синтезатор, XST, і зовнішні синтезатори. Як зовнішні синтезатори, рекомендується використовувати Synplify або Precision.

- Процеси імплементації та створення конфігурації реалізовані засобами ISE у вигляді виклику набору відповідних утиліт. Протягом усіх етапів маршруту проектування користувач може контролювати правильність виконання кроків шляхом аналізу різноманітних звітів.



Рис.1.6. Покриття всього маршруту проектування для FPGA засобами Xilinx

1.5. Висновки до розділу

На основі аналізу етапів схемотехнічного проектування цифрових пристроїв та особливостей проектування для FPGA з використанням засобів фірми Xilinx можна зазначити, що автоматизоване проектування електронних схем за допомогою ПЕОМ має кілька переваг порівняно з традиційним методом проектування "вручну", який включає подальше випробування на фізичному макеті:

- з використанням прикладних програм значно легше вивчити ефект варіювання параметрів схеми, ніж у суто експериментальних дослідженнях;
- існує можливість аналізувати критичні режими пристрою без фізичної руйнації його компонентів;

- програми аналізу дозволяють оцінити роботу схеми при найгірших поєднаннях параметрів, що важко і не завжди можливо здійснити експериментально;
- програми дають можливість провести вимірювання на моделі електронної схеми, які було б важко реалізувати в лабораторії експериментально;
- використання новітніх кристалів з архітектурою FPGA від Xilinx, спроектованих за технологією high-K metal gate та 28-нм техпроцесу, дозволяє досягти максимальної продуктивності при мінімальному енергоспоживанні.

РОЗДІЛ 2. МОДЕЛЬНО-ОРІЄНТОВАНЕ ПРОЄКТУВАННЯ. ОСНОВНІ БЛОКИ SYSTEM GENERATOR

2.1. Етапи модельно-орієнтованого проєктування

У модельно-орієнтованому проєктуванні процес розробки зосереджений навколо системної моделі - від фіксації і розробки технічних вимог до впровадження і тестування.

Дана модель системи - основа виконуваною специфікації, яка використовується і розробляється на всьому протязі процесу проєктування (рис.2.1).



Рис.2.1. Етапи модельно - орієнтованого проєктування

Специфікація що виконується може також включати вхідні дані і передбачувані вихідні дані або критерії відповідності та умови експлуатації, а також посилання на вимоги. Мета виконуваної специфікації полягає в

однозначному формулюванні мети розробки, а також в можливості аналізу здійсненності і сумісності вимог за допомогою моделювання.

При модельно-орієнтованому проєктуванні ефективність роботи інженерів підвищується завдяки таким можливостям:

- використання загального середовища проєктування для всіх проєктних груп;
- пряма прив'язка розробок до вимог;
- інтеграція тестування і розробки для безперервного виявлення і виправлення помилок;
- вдосконалення алгоритмів за допомогою багатодоменого моделювання;
- автоматична генерація вбудованого ANSI C, Verilog і VHDL коду;
- розробка і багаторазове використання комплексних тестів;
- автоматичне створення документації;
- багаторазове використання розробок для розгортання системи на багатоядерних процесорах і кластерах.

Модельно-орієнтоване проєктування (model-based design) виявляється ефективним методом, який дозволяє суттєво зменшити витрати часу і фінансові витрати під час розробки, при цьому не втрачаючи якості кінцевого продукту [2, 3]. У випадку модельно-орієнтованого проєктування систем реального часу основною основою розробки є програмна модель об'єкта управління (див. рис. 2.1).

Ця модель є єдиною для розробників з різних галузей знань (інженерів-розробників систем управління, фізиків, математиків, проєктувальників електричних, механічних, гідравлічних систем і т. д.). Вимоги до математичної моделі об'єкта та кінцевого продукту пов'язані з програмною моделлю, що забезпечує прозорість розробки і, відповідно, дотримання одних і тих самих вимог всіма учасниками проєкту.

Алгоритми, розроблені для математичної моделі об'єкта управління, перевіряються на програмній моделі, що дозволяє уникнути витрат на раннє

прототипування (створення апаратних прототипів пристроїв) та поломок прототипів. Після програмної перевірки автоматично генерується програмний код для керуючого пристрою (контролера, ПЛІС, промислового комп'ютера). Керуючий пристрій підключається до вихідної програмної моделі об'єкта управління, що забезпечує перевірку працездатності алгоритму на керуючому пристрої в режимі реального часу (Processor-in-the-Loop, PIL). Автоматична генерація коду з моделі дозволяє уникнути помилок, пов'язаних з людським фактором, і знизити часові витрати на етапі розробки.

Наступний етап - програмно-апаратне моделювання (Hardware-in-the-Loop, HIL) - дозволяє тестувати та досліджувати алгоритм управління, реалізований в пристрої, в реальному часі [2, 4]. На цьому етапі замість моделі об'єкта, реалізованого в математичному середовищі розробки, використовується модель, що виконується в реальному часі. Керуючий пристрій підключається до комп'ютера, який моделює об'єкт управління і працює в реальному часі. Після проходження всіх попередніх етапів керуючий пристрій з алгоритмом управління застосовується на реальному об'єкті.

Середовищі MATLAB/Simulink може бути використане для реалізації всіх етапів модельно-орієнтованого проектування - від урахування вимог до розробки до тестування і сертифікації кінцевого продукту (див. Рис. 2.2).

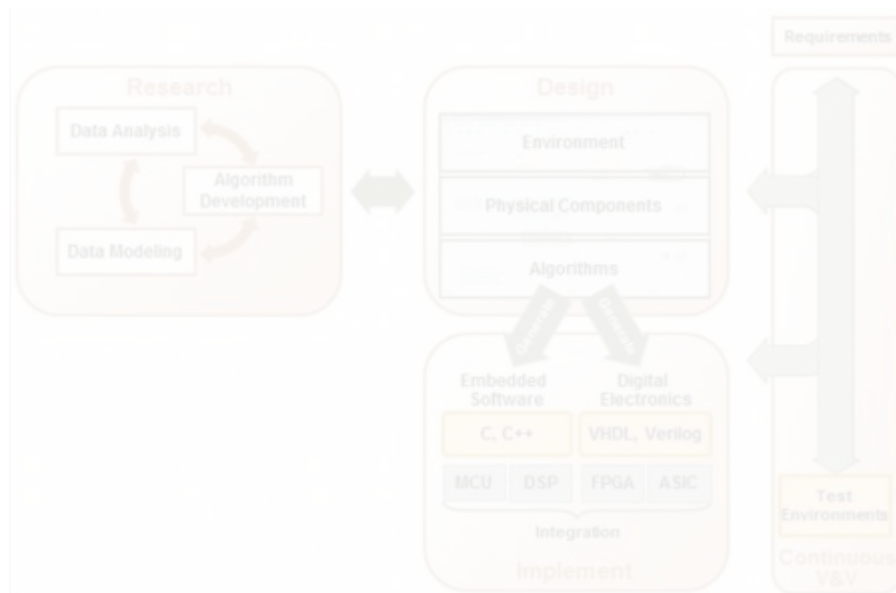


Рис. 2.2. Модельно-орієнтоване проектування вбудованих систем в MATLAB/Simulink

2.2. Модельно-орієнтоване проектування в MATLAB / Simulink с використанням System Generator

Створення моделі в Simulink ґрунтується на методології проектування що полягає в створенні і використанні стандартних блоків Simulink. Ця специфікація може бути розроблена з використанням змінних з плаваючою точкою і без такої. Після того як функціональність і основні потоки даних були визначені, система генератор може використовуватися для визначення апаратної реалізації деталі для Xilinx пристроїв. Система використовує генератор Xilinx DSP Blockset для Simulink і автоматично викликає генератор Xilinx Core™ для створення високо-оптимізованих зв'язків для блоків DSP. System Generator може виконувати всі нижче наведені інструменти реалізації.

Продукт бітового потоку для програмування FPGA. Додатковий випробувальний стенд може бути створений з використанням тестових векторів які витягуються з середовища Simulink для використання з ModelSim або Xilinx ISE® Simulator (рис.2.3)

23



Рис. 2.3. Проектування в MATLAB / Simulink з використанням System Generator

Xilinx DSP Blockset запускається через браузер Simulink Library, який може бути запущений з стандартної MATLAB панелі інструментів. Блоки розділені на підкатегорії для полегшення пошуку. Одні підкатегорії "Index" включають в себе всі блоки і часто це найшвидший спосіб отримати доступ до блоку.

Більше 90 DSP блоків побудови приведені в наборі блоків Xilinx DSP для Simulink. Ці блоки включають в себе загальні DSP блоки побудови, такі як суматори, мультиплікатори, мультиплексори, регістри і інші, показані на малюнку 2.4. Також є набір складних DSP блоків побудови, таких як блоки виправлення помилок, БПФ блоки, ітератори і пристрої, що запам'ятовують. Ці блоки спираються на основні Xilinx IP генератори і забезпечують оптимальні результати для вибраного пристрою.

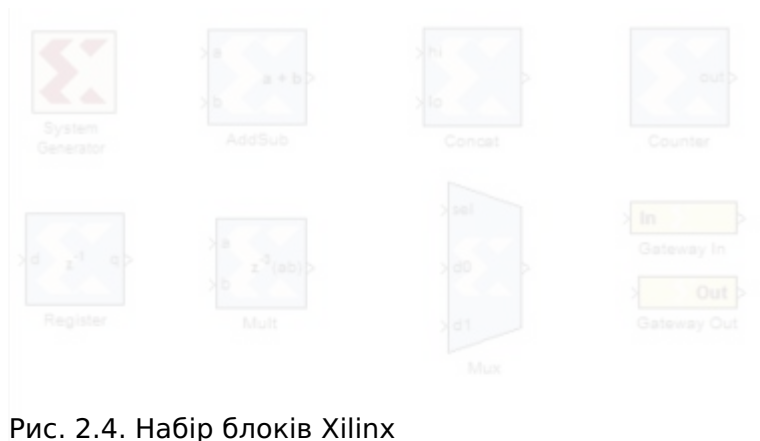


Рис. 2.4. Набір блоків Xilinx

Доступ до набору блоків Xilinx DSP здійснюється через браузер бібліотек Simulink, який може бути запущений з стандартної панелі інструментів MATLAB (рис. 2.5). Блоки розділені на підкатегорії для полегшення пошуку. Одна підкатегорій, Index включає в себе всі блоки і часто є найшвидшим способом отримати доступ до блоку, з яким ви вже знайомі.

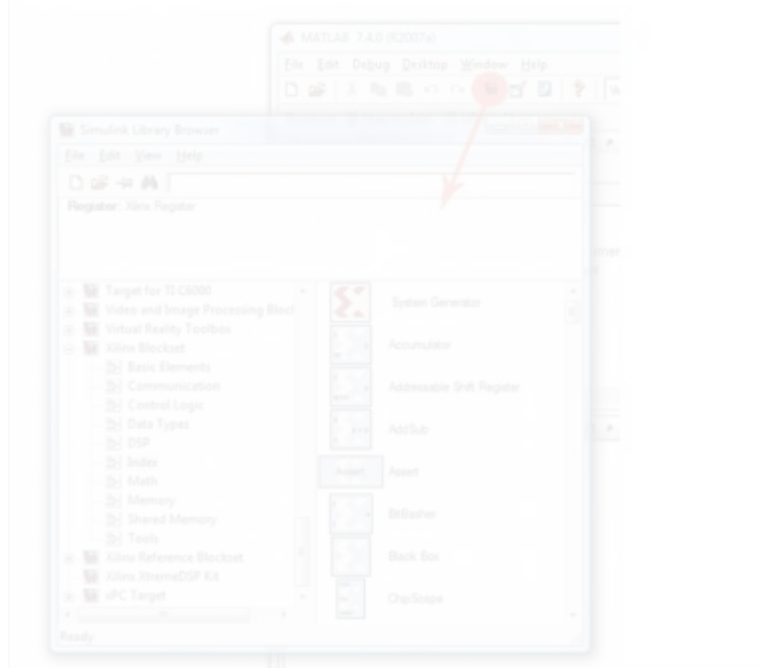


Рис.2.5. Стандартна панель інструментів MATLAB

Системний генератор працює зі стандартними моделями Simulink. Два блоки названі Gateway In і Gateway Out визначають межі FPGA в моделі симуляції Simulink. Подвійний клік на блоках викликає редактор властивостей, де властивості блоку можуть бути повністю визначені (Рис 2.6).



Рис2.6. Gateway In и Gateway Out блоки

Блок Gateway In конвертує вступні типи Simulink integer, double і fixed-point в Xilinx fixed-point число. Xilinx fixed-point типи:

- Boolean;
- Signed (додаток до двох);
- Unsigned.

Якщо обраний тип знаковий або беззнаковий, кількість біт і бітові точки повинні бути визначені. Кількість біт представляє ширину введення, в

той час як параметр довічних точок показує кількість біт праворуч від двійкової точки (розмір ділення). Позиція двійкової точки повинна бути між нулем і певною кількістю біт.

Конвертуючи тип Simulink в System Generator fixed-point тип, Gateway In використовує обрані опції квантування і переповнення. Опції для квантування:

- Округлення - округлити до найближчого представимого значення (або значення найбільш віддалене від нуля, якщо є два еквідистантних найближчих представимо значення);
- Усікти - відкинути біти праворуч від найменш важливого представимо біта;

Опції переповнення:

- Згорнути - відкинути біти зліва від найбільш важливого представимо біта;
- Наситити - наситити до найбільшого позитивного / найменшого від'ємного значення;
- Позначити як помилку - позначити переповнення як помилку Simulink під час симуляції;

Важливо розуміти, що переповнення і квантування для Gateway In не відбувається на рівні заліза - вони відбуваються на рівні ПО блоку, до входу в фазу заліза. На рівні заліза блоки Gateway In стають вступними портами вищого рівня.

Блок Gateway Out конвертує вступні значення типу Xilinx fixed-point у вихідні значення типу Simulink integer, double або fixed-point. На рівні заліза ці блоки стають похідними портами вищого рівня або відкидаються, в залежності від того як вони сконфігуровані.

Блок MCode.

Один з блоків, який заслуговує на особливу увагу, є блок MCode. Це контейнер для виконання написаної користувачем в MATLAB функції в

Simulink. Блок виконує М-код для розрахунку вихідних величин під час моделювання Simulink.

Simulink інтерфейс блоку береться з опису функції в MATLAB і з параметрів маски блоку. Існує один вхідний порт для кожного параметра функції і одним порт для кожного значення, що повертається. Імена та порядок портів відповідає іменам і порядку повертаються значень і параметрів.

MCode блок підтримує обмежена підмножина мови MATLAB, яке є корисним для реалізації арифметичних функцій, кінцевих автоматів і логіки управління. Він має три основних керівних принципи кодування, які повинні бути виконані:

- Всі блоки входів і виходів повинні бути типу Xilinx з фіксованою крапкою;
- Блок повинен мати принаймні один вихідний порт;
- Код для блоку повинна існувати на шляху MATLAB або в тій же директорії, де файл моделі, яка використовує цей блок.

Щоб проілюструвати функціональність блоку MCode, ми покажемо простий приклад який виконує функцію $z = \max(x; y)$. Файл `xlmax.m` містить функцію `xlmax`, реалізовану в такий спосіб:

```
function z = xlmax(x, y)
    if x > y
        z = x;
    else
        z = y;
    end
```

MCode блок на основі функції `xlmax` матиме порти введення x і y і вихідний порт z . Рисунок 2.7. показує, як налаштувати MCode блок щоб використовувати функцію `xlmax`.



Рис 2.7. Властивості блока MCode.

Деякі конструкції мови MATLAB які підтримує блок MCode:

- Призначення звітності;
- Прості і складові if / else / elseif оператори;
- Оператор switch;
- Арифметичні вирази за участю тільки додавання і віднімання;
- Додавання, віднімання, множення, ділення на ступені двійки;
- Реляційні оператори;
- Логічні оператори.

Для інших MATLAB конструкцій / функцій, які можуть бути використані в MCode файлі зверніться до довідкової документації по системному генератору Xilinx.

Іноді модель повинна включати підсистеми, які не можуть бути реалізовані за допомогою Xilinx блоків. Наприклад, Дизайн може зажадати FIR-фільтр, чиї можливості відрізняються від тих, які реалізовані в фільтрі який поставляється в Xilinx. Blockset. bloc box дають можливість включати в себе такі підсистеми Щоб додати bloc box в модель необхідно виконати наступне: Впровадження підсистеми (ваш чорний ящик) в Simulink.

Підсистема може містити будь-яку комбінацію Xilinx і не Xilinx блоків. Встановіть Xilinx чорний блас box на верхньому листі рівні підсистеми. Це вказує System Generator, що користувач буде забезпечувати VHDL або VerilogHDL, необхідні для реалізації цієї підсистеми. Двічі клацніть на маркер, щоб відкрити чорний ящик блок параметрів діалогового вікна. Введіть інформацію, яка описує чорний ящик. Ви повинні вручну ввести VHDL або Verilog HDL в блас box файли.

Розглянемо приклад підключення VerilogHDL до проєкту виконаному в System Generator.

Для оформлення проєкту на VerilogHDL необхідно використовуючи ISE Project Navigator створити новий проєкт (рис 2.8)

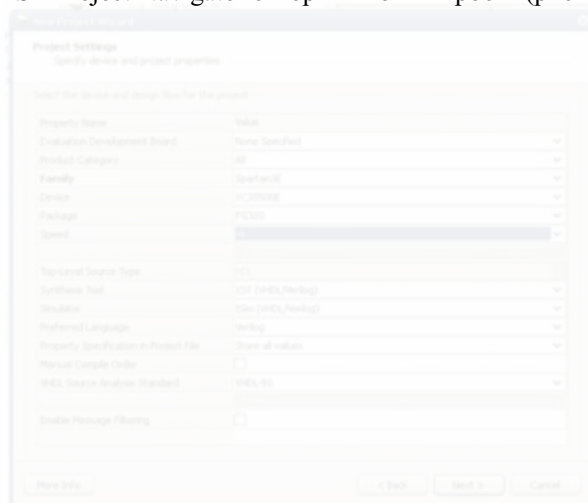


Рис.2.8. Створення проєкту

Далі використовуючи мову проектування пристроїв Verilog HDL створити код процедури який повинен виконуватися (рис 2.9).

Рис.2.9. Код додатка

Потім отриманий код згенерувати і в разі відсутності помилок в розділі проєкту з'явиться файл з розширенням `.v` даний файл необхідно скопіювати в розділ де розміщені м-файли.

Після цього необхідно пов'язати з кодом для чого на іконці blas box два рази натиснути правою кнопкою мишки після чого у вікні виділити додаток з розширенням .v і натиснути ok (рис.2.10).

Після цього у блоку `blac box` визначитися необхідну кількість входів виходів і при подвійному натисканні на нього можна побачити блок з кодом.

Наступним етапом необхідно зібрати схему. При цьому необхідно пам'ятати про відповідність узгодженості вихідних характеристик блоку Gateway In з вхідними bloc box (рис 2.11)

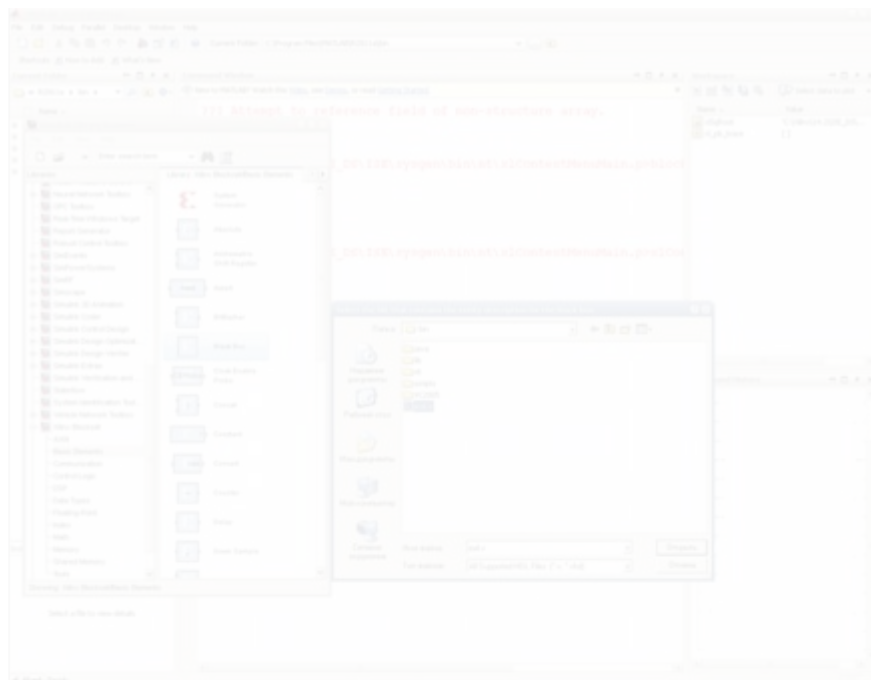


Рис. 2.10. Зв'язування bloc box с кодом

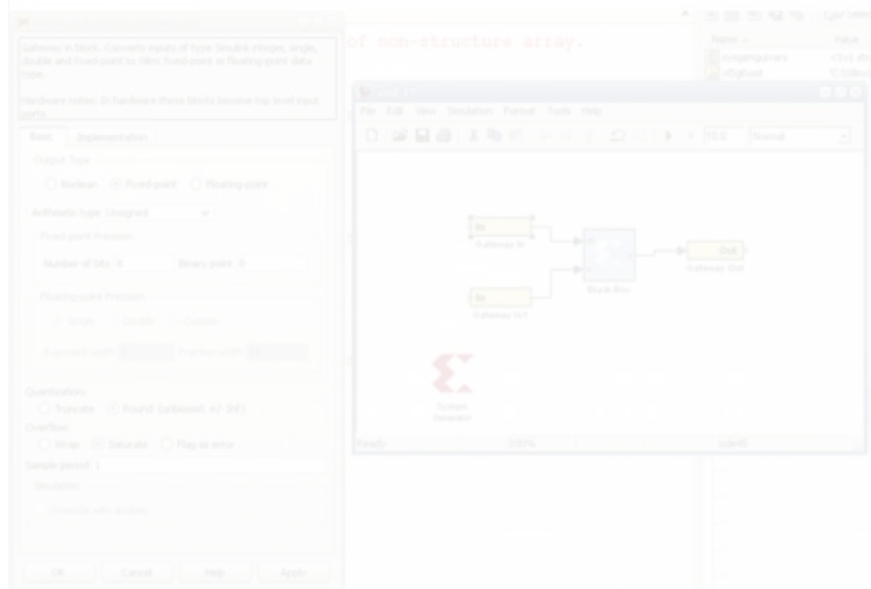


Рис. 2.11. Соголасування розмірності змінних

Далі наведено зразок моделювання для двох діапазонів до 10 и до 100 (Рис 2.12).

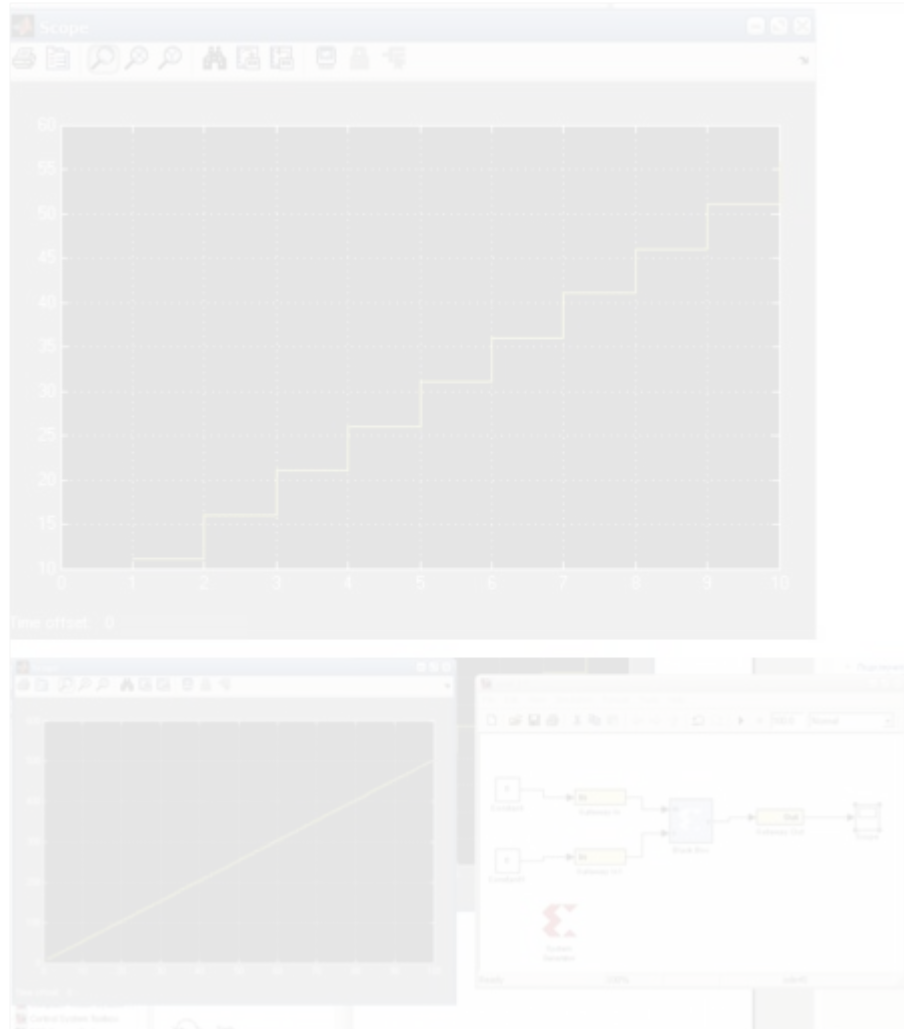


Рис. 2.12. Приклади моделювання для діапазонів 10 и 100

2.3. Висновки по розділу

Виходячи з аналізу, наведеного в розділі можна зробити висновок, що при модельно-орієнтованому проєктуванні кількість етапів процесу проєктування значно менше ніж при використанні традиційних способів.

Серед Simulink дозволяє описувати об'єкти за допомогою виконуваних блок схем. Для роботи в такому середовищі від фахівця не потрібно спеціальних навичок програмування.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЄКТІВ ЦИФРОВИХ АВТОМАТІВ У MATLAB/SIMULINK

3.1. Архітектура інструментального модуля Xilinx Spartan 3E Starter Board

Інструментальний комплект Spartan-3E Starter Kit призначений в першу чергу для практичного вивчення методів проєктування цифрових пристроїв з апаратною реалізацією операцій та розробки вбудованих мікропроцесорних систем на основі ПЛІС передових сімейств FPGA фірми Xilinx. Унікальні функціональні можливості, технічні характеристики та конструктивне виконання інструментального модуля, що входить до цього комплекту, дозволяють не тільки виконувати налагодження проєктів, що реалізуються на основі ПЛІС сімейства Spartan-3E, але й використовувати його як промисловий серійний варіант пристрою, що розробляється.

Архітектура інструментального модуля Xilinx Spartan-3E Starter Board дозволяє використовувати його для реалізації автономних систем управління, збору та обробки інформації, цифрової обробки сигналів, цифрових пристроїв з різними комп'ютерними інтерфейсами. Крім того, цей модуль можна рекомендувати для застосування в навчальних лабораторіях вузів для вивчення сучасних методів проєктування цифрових пристроїв, мікропроцесорних систем з різною архітектурою, а також цифрових пристроїв обробки сигналів. Ресурси інструментального модуля Xilinx Spartan-3E Starter Board дозволяють реалізувати проєкти мікропроцесорних систем, що вбудовуються, що виконуються на основі як 8-розрядних ядер сімейства PicoBlaze, так і 32-розрядних ядер сімейства MicroBlaze. Декілька типових проєктів, спеціально розроблених для аналізованого інструментального комплекту, наочно демонструють не тільки можливості налагоджувальної плати, але і представляють можливі варіанти конфігурації вбудованих систем, створюваних на базі цих мікропроцесорних ядер.

До складу комплекту Spartan-3E Starter Kit входять:

- плата інструментального модуля Xilinx Spartan-3E Starter Board;

35

- мережевий адаптер з вихідною стабілізованою напругою 5 В та аксимальним струмом навантаження 2,5 А, що використовується як первинне джерело живлення інструментального модуля;
- стандартний USB-кабель Type A/Type B, який підключається до відповідного порту комп'ютера та призначений для конфігурування ПЛІС та програмування конфігураційного ППЗУ, встановлених на платі інструментального модуля Xilinx Spartan-3E Starter Board, за допомогою інтегрованої схеми завантажувального кабелю;
- комплект CD-ROM, що містить нову повнофункціональну версію системи проєктування ISE™ (Integrated Software Environment/Integrated Synthesis Environment) WebPACK™, оціночні версії САПР ISE Foundation™ та засобів розробки вбудованих мікропроцесорних систем Xilinx Embedded Development Kit™ (EDK) протягом 60 днів з моменту встановлення, а також Spartan-3/3E Starter Kit Resource CD.

Інструментальний модуль Xilinx Spartan-3E Starter Board характеризується такими відмінними рисами:

- використання в якості основного компонента ПЛІС з сімейства Spartan-3E з об'ємом 500 000 системних вентилів (10 476 логічних осередків) в корпусі FG320, об'єм логічних і трасувальних ресурсів якої в поєднанні з великою кількістю користувальницьких висновків дозволяють реалізувати не тільки окремі функціональні , але систему, що розробляється в повному обсязі;
- можливість підтримки на рівні проєктів стандартних інтерфейсів обчислювальних систем RS-232, PS/2 та VGA, що дозволяє підключати до інструментального модуля зовнішні пристрої з послідовним інтерфейсом, клавіатуру/мишу та дисплей відповідно;

36

- застосування у схемі модуля перетворювача рівнів RS-232, що забезпечує можливість безпосереднього підключення до послідовного порту зовнішніх пристроїв різного типу через роз'єми DB-9 (типу DTE та DCE), встановлені на платі (при реалізації універсального асинхронного приймача UART на основі ресурсів ПЛІС);
- наявність додаткових компонентів, що реалізують фізичний рівень інтерфейсу 10/100 Ethernet PHY, що дозволяє підключати інструментальний модуль через стандартний роз'єм безпосередньо до відповідної мережі при реалізації контролера Ethernet MAC у складі проєкту, що завантажується в кристал сімейства Spartan-3E;
- підключення спеціальних та користувальницьких висновків ПЛІС до контактів чотирьох роз'ємів розширення, що відповідають різним стандартам, що забезпечує можливість гнучкого сполучення із зовнішніми компонентами та пристроями;
- застосування зовнішнього високошвидкісного синхронного динамічного ОЗП з подвоєною швидкістю передачі даних, виконаного у вигляді DDR SDRAM ємністю 64 Мбайт (512 Мбіт), що розширює можливості оперативної пам'яті вбудованих систем, що реалізується на основі відповідних ресурсів ПЛІС;
- використання як стандартного ППЗУ для зберігання конфігураційних даних ПЛІС Flash-пам'яті серії Platform Flash, що програмується в системі, об'ємом 4 Мбіт;
- включення до схеми інструментального модуля додаткової ПЛІС із архітектурою CPLD XC2C64A сімейства CoolRunner-II [11-14], що використовується, зокрема, для комутації конфігураційної пам'яті різного типу та управління режимами конфігурування основного кристала (серії Spartan-3E);

- інтегрована схема завантажувального кабелю, що дозволяє виконувати конфігурування всіх ПЛІС та програмування конфігураційної пам'яті, що входять до складу інструментального модуля, за допомогою стандартного кабелю, що підключається до порту USB комп'ютера;
- наявність додаткового спеціального роз'єму для підключення стандартних завантажувальних кабелів різного типу, що дозволяють виконувати операції конфігурування ПЛІС та програмування ППЗУ у різних режимах, а також зворотного зчитування конфігураційних даних через порт JTAG-інтерфейсу;
- використання у складі інструментального модуля паралельної NOR Flash-пам'яті ємністю 16 Мбайт (128 Мбіт), яка може використовуватися, зокрема, для зберігання конфігураційної послідовності даних ПЛІС сімейства Spartan-3E або програмного коду вбудованої мікропроцесорної системи, що виконується на базі 32-розрядного ядра сімейства MicroBlaze;
- наявність послідовної Flash-пам'яті об'ємом 16 Мбіт з інтерфейсом SPI (Serial Peripheral Interface), що використовується для запису конфігураційної інформації кристала сімейства Spartan-3E або програмного коду, що вбудованої мікропроцесорної системи, що виконується на базі 32-розрядного ядра сімейства MicroBlaze;
- включення до складу схеми інструментального модуля послідовного ППЗУ EEPROM, який підтримує криптографічний алгоритм Secure Hash Algorithm (SHA-1) та призначений для захисту конфігураційних даних від несанкціонованого копіювання;
- застосування двоканального аналого-цифрового перетворювача ADC (Analog-to-Digital Converter) з інтерфейсом управління SPI,

що забезпечує можливість реалізації пристроїв цифрової обробки сигналів;

- наявність чотириканального послідовного цифро-аналогового перетворювача DAC (Digital-to-Analog Converter) з 12-розрядною роздільною здатністю, керованого за допомогою інтерфейсу SPI;
- застосування кварцового генератора із частотою 50 МГц, призначеного для формування основного тактового сигналу для ПЛІС;
- наявність панелі для встановлення додаткового кварцового генератора, що використовується як альтернативне або додаткове джерело сигналу синхронізації;
- присутність на платі елементів індикації різних типів, що забезпечують можливість візуального контролю напруги живлення, процесу конфігурування кристала та функціонування системи, що розробляється;
- наявність чотирьох повзункових перемикачів, чотирьох кнопок і поворотного перемикача, поєднаного з кнопкою, які можуть використовуватися, наприклад, для ручної установки режиму роботи реалізованої системи або в процесі налагодження проєктованої системи, а також для тестування інструментального модуля та прикладного програмного забезпечення;
- використання комплексної схеми управління живленням, що виконує функції формування напруг, необхідних для живлення компонентів модуля, у тому числі для блоків введення/виводу та ядра кристала FPGA, конфігураційного ППЗП, елементів оперативної та постійної пам'яті, ЦАП та АЦП, інтегрованої схеми завантажувального кабелю;
- наявність кнопки, що забезпечує реалізацію режиму примусового завантаження послідовності конфігурації в основну ПЛІС;

- повна сумісність з усім сімейством систем проектування та програмування кристалів фірми Xilinx (ISE WebPACK та ISE Foundation) та підтримка засобами розробки вбудованих мікропроцесорних систем Xilinx EDK. Зовнішній вигляд інструментального модуля подано на рис 3.2.



Рис. 3.1. Зовнішній вигляд інструментальний модуль Xilinx Spartan-3E

Усі компоненти модуля змонтовані на друкованій платі із двостороннім розміщенням компонентів. Структурне уявлення архітектури модуля, що розглядається, показано на рис. 3.2. Основними елементами архітектури модуля Xilinx Spartan-3E Starter Board є:

- головна ПЛІС XC3S500E сімейства Spartan-3E у корпусі FG320, на основі якої реалізується проектувана система;
- програмоване в системі ППЗУ серії Platform Flash XCF04S, призначене для зберігання конфігураційних даних ПЛІС XC3S500E;
- блок завантаження конфігураційних даних;
- схема управління конфігуруванням ПЛІС;
- допоміжна ПЛІС CPLD XC2C64A сімейства CoolRunner-II;
- послідовне ППЗУ EEPROM, піддер SHA-1;

- блок синхронізації, призначений для формування зовнішніх (стосовно ПЛІС) тактових сигналів;
- зовнішнє високошвидкісне ОЗП;
- вузол двоканального аналого-цифрового перетворювача (АЦП);
- вузол чотириканального цифро-аналогового перетворювача (ЦАП);
- модуль паралельної NOR Flash-пам'яті ємністю 16 Мбайт (128 Мбіт);
- модуль послідовної Flash-пам'яті об'ємом 16 Мбіт з інтерфейсом SPI;
- схема формування та контролю живильних напруг;
- блок світлодіодних індикаторів;
- дворядковий шістнадцятизначний рідкокристалічний дисплей;
- блок повзункових перемикачів;
- блок кнопочових перемикачів, з поєднаним поворотним перемикачем;
- схема перетворення рівнів сигналів інтерфейсу RS-232;
- модуль фізичного рівня інтерфейсу 10/100 Ethernet PHY;
- стандартні роз'єми інтерфейсів RS-232, PS/2, VGA та Ethernet;
- чотири роз'єми розширення.



Рис. 3.2. Структурне представлення архітектури інструментального модуля Xilinx Spartan 3E Starter Kit

Тип кристалу сімейства Spartan-3E, що використовується як основна ПЛИС розглянутого інструментального модуля, значною мірою визначає функціональні можливості останнього

Виходячи з того, що вихідна напруга інструментального модуля Xilinx Spartan#3E становить 3,3 а також для того щоб виключити вплив силових схем необхідно розробити блок опторозв'язки.

3.2. Створення кінцевого автомата на основі блоку MCode

Для подальшої ілюстрації функціональних блоків MCode ми наведемо приклад побудови простого кінцевого автомата (FSM), який визначає послідовність 1101 у вхідному потоці бітів. На малюнку 3.3 показана поведінкова функція автомата. М-функція, яка використовується блоком MCode містить перехідну функцію, яка обчислює наступний стан, засноване

на поточний стан і поточних входних значень. М-функція в даному прикладі визначає стійкі змінні стану для зберігання стану кінцевого автомата в блоці MCode. Наступний М-коду, який визначає функцію виявлення 1101 міститься в файлі fsm.m

```
:% This FSM detects the 1101 sequence
% Bits are loaded in a serial manner
function matched = detect 1101 (d in)
    seen none = 0;
    seen 1 = 1;
    seen 11 = 2;
    seen 110 = 3;
% the state is a 2bit register
persistent state, state = xl state(seen none, fxUnsigned, 2, 0g);
    matched = 0;
    switch state
        case seen none
            if d in == 1
                state = seen 1;
            else
                state = seen none;
            end
        case seen 1
            if d in == 1
                state = seen 11;
            else
                state = seen none;
            end
        case seen 11
            if d in == 1
                state = seen 110;
            else
                state = seen none;
            end
        case seen 110
            if d in == 1
                state = seen 1101;
            else
                state = seen none;
            end
    end
end
```

43

```
else
    state = seen 110;
end
case seen 110
if d in == 1
    state = seen 1;
    matched = 1;
else
    state = seen none;
end
otherwise
    state = seen none;
end
end
```

Попередній М-код має внутрішню змінну стану, яка тримає його значення з одного кроку моделювання до наступного. Мінлива стану оголошена з ключовим словом MATLAB persistent і їй спочатку має бути присвоєно значення виклику функції стану xl_state. Мінлива стану оголошена як persistent, і перше завдання для state є результат виклику xl_state. Функція xl_state спочатку приймає два аргументи. Перший являє собою первісне значення і має бути постійною. Другий, точність змінної стану.

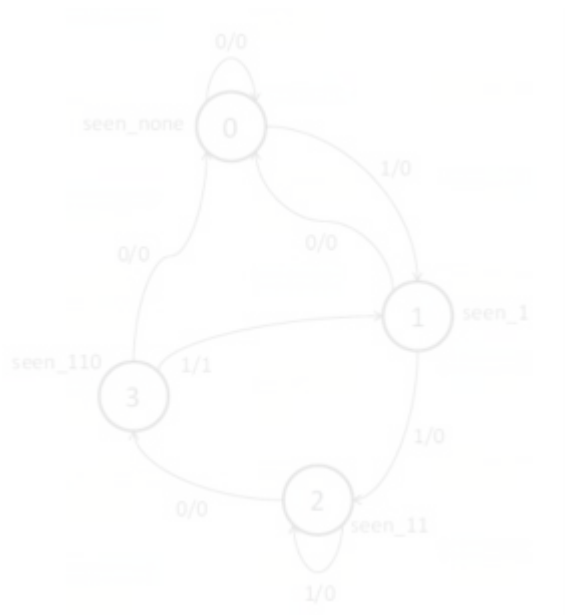


Рис.3.3. Граф кінцевого автомата для детектування послідовності та 1101

У нашому прикладі, лінія постійного стану, стану = xl_state (не знайшов жодного, xlUnsigned, 2, 0) визначає стан змінної у вигляді 2-бітних беззнакових змінних (регістрів) з двійкової точки в положенні 0 і ініціалізує його з значенням «не знайшов ні одного». На рисунку 3.3 показано комплексне рішення для попереднього прикладу.

Зверніть увагу, що, оскільки у нас є 4 стану, 2 біта досить для змінної стану. Немає необхідності у виділенні більшої кількості біт. Однак, вважаючи, що у нас було 5 станів, ми повинні були б виділити 3 біта дають 8 можливих станів. Іншими словами, ми маємо 3 невикористаних стану. Навіть якщо вони не використовуються, дуже важливо включати їх в switch-case блоки. Припустимо, ми тільки визначимо стану від 0 до 4, і після включення живлення, FPGA починається в стані 6.FSM завжди залишатиметься там, так як перехід зі стану 6 не визначений в даному випадку. Таким чином, необхідно також визначити стану від 5 до 7. Логічною реалізацією було б перейти на добре певний стан, наприклад в початковий стан.

Замість того щоб додавати окремий case блок для кожного невикористаного стану, ви також можете використовувати блок otherwise.

Це дозволить переконатися, що ми не забуваємо будь які невикористані стани. У наведеному вище прикладі коду, в блок otherwise був також доданий, хоч з чотирма станами, немає невикористаних станів. Проте, це хороша практика, завжди додавати ще один, так як це дозволяє уникнути помилок при додаванні або видаленні станів надалі. Реалізація автомата представлена на рис 3.4.

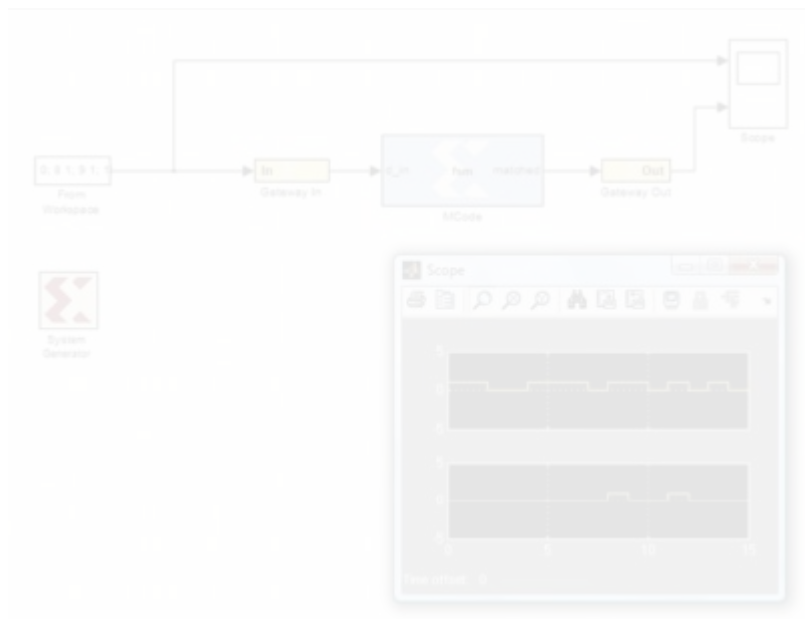


Рис.3.4. Реалізація цифрового автомату

3.3. Розробка кінцевого автомата за допомогою блоків

У цьому прикладі ми створюємо машину станів для кодового замка. Ми використовуємо Xilinx автомат станів Мілі, який є частиною довідкової бібліотеки машин станів. Автомат станів реалізований блоком і розподіленої ОЗУ, що призводить до швидкої і ефективної реалізації. Цей приклад використовує тільки один блок пам'яті і працює на частоті більш ніж 150 МГц.

Бібліотека автомата станів доступна в Xilinx Blockset.

Ми почнемо з визначення кодового замка машини станів. Він має один вхід X і два виходи: UNLOCK і HINT. Висновок UNLOCK повинен бути 1, якщо і тільки якщо $X = 0$ і послідовність матеріалів, отриманих на X в попередніх семи циклах була '0110111'. Висновок HINT повинен бути 1, якщо і тільки якщо поточне значення X є правильним для переміщення машини станів ближче до розблокування станом (з UNLOCK = 1).

Таблиця наступних станів / висновків для автомата станів показана нижче (рис.3.5).

Meaning	Current State	If $X = 0$	If $X = 1$
Seen nothing	0	1, 01	0, 00
Seen 0	1	1, 00	2, 01
Seen 01	2	1, 00	3, 01
Seen 011	3	4, 01	0, 00
Seen 0110	4	1, 00	5, 01
Seen 01101	5	1, 00	6, 01
Seen 011011	6	4, 00	7, 01
Seen 0110111	7	1, 11	0, 00

Next State/Output Table

Рис.3.5. Стан автомату

Таблиця показує, наступний стан і результати від поточного стану і введення X . Наприклад, якщо поточний стан 3 і $X = 0$, наступний стан: 4. Перехід зі стану 3 в 4 означає, що ми виявили послідовність '0110'. Під час цього переходу, вихід UNLOCK дорівнює 0 і вихід HINT дорівнює 1, вказуючи, ми ще на один крок ближче до стану розблокування. Машина станів переходить до наступного стану, якщо ми отримаємо правильний вхід і повертається в стан 0 або 1, якщо ми отримаємо невірний. Стан 6 є винятком, якщо ми отримаємо неправильне введення, тобто, "0", попередні три входи все ще можуть виявитися початком необхідної послідовності. Таким чином, ми повертаємося в стан 4, замість стану 1. У стані 7, ми отримали потрібну послідовність, тому ми встановлюємо UNLOCK в 1, якщо $X = 0$. У кожному

стані, ми встановлюємо HINT в 1 для значення X, яке переміщує машину станів ближче до стану 7.

З наведеного опису видно, що ми повинні користуватися машиною станів Милі для здійснення проектування, так як виходи залежать як від поточного стану, так і від вхідних даних. Блок-схема машини станів цього виду показана нижче.

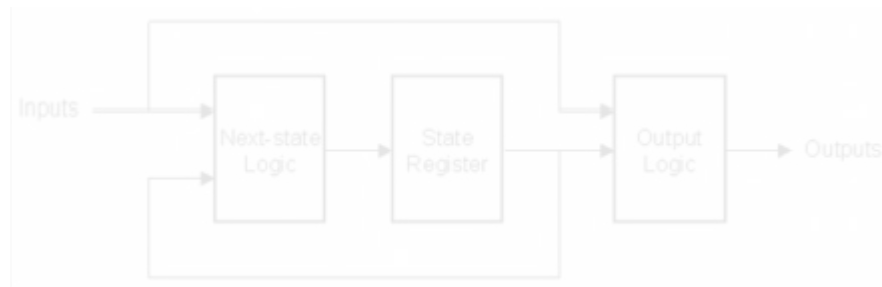


Рис. 3.6. Блок-схема машини станів

Блок конфігурується за допомогою матриць наступного стану та вихідних значень, отримані з таблиці наступних станів / значень виведення показаної вище. Ці матриці будуються таким чином: (рис3.7)

Meaning	Current State	If X = 0		If X = 1	
Seen nothing	0	1	01	0	00
Seen 0	1	1	00	2	01
Seen 01	2	1	00	3	01
Seen 011	3	4	01	0	00
Seen 0110	4	1	00	5	01
Seen 01101	5	1	00	6	01
Seen 011011	6	4	00	7	01
Seen 0110111	7	1	11	0	00

Next State/Output Table

1	0
1	2
1	3
4	0
1	5
1	6
4	7
1	0

Next State Matrix

1	0
0	1
0	1
1	0
0	1
0	1
0	1
3	0

Output Matrix

Рис.3.7.Матриці станів вихідних значень

Рядки матриці відповідають поточному стану, а стовпці відповідають вхідним значенням. Висновок машини станів є N-бітний вектор. (В даному прикладі N = 2.) Елемент матриці виходів є десятковим поданням N-бітного вихідного вектору для конкретного стану. У цьому прикладі, виходи UNLOCK і HINT об'єднуються разом, щоб зробити елементи матриці виходів. Наприклад, коли машина знаходиться в стані 7 і X = 1, UNLOCK і HINT рівні 1, так що відповідний елемент матриці виходів становить 3 (десятькове подання двійкових '11').

Наступне логічне стан і реєстр стану в цьому блоці реалізується з високошвидкісним блоком RAM. Логічний вихід реалізований з використанням розподіленої пам'яті, конфігуровано як таблиця підстановки, і, отже, має нульову затримку.

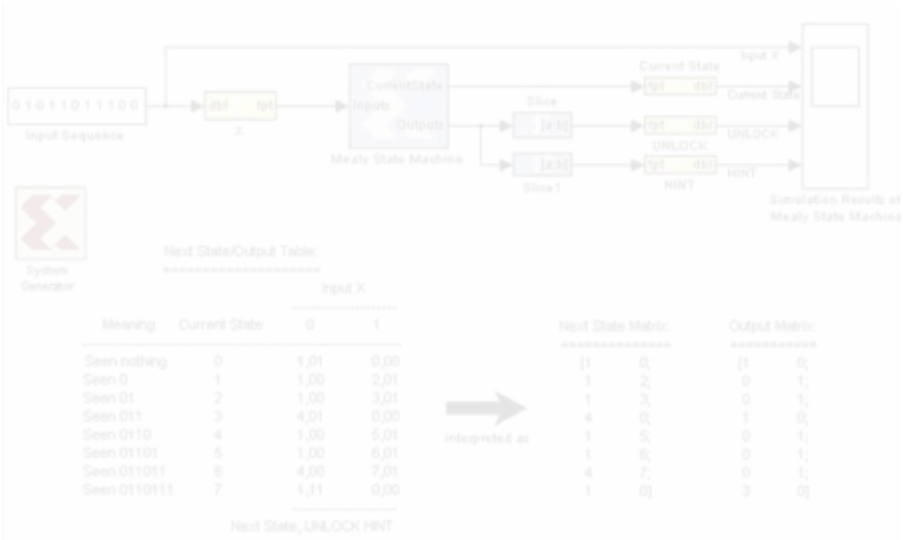


Рис. 3.8. Реалізація автомату

Параметри діалогового вікна блоку можуть бути викликані подвійним клацанням по іконці Милі автомата в вашої моделі. У цьому діалоговому вікні, визначаються матриці стану і виходу, задається і кількість вихідних бітів встановлене в 2. Два слайс-блоку використовуються для вилучення UNLOCK і HINT бітів з виведення.

Коли проект симулюється з [00000000000010110111001100110111] в якості вектора введення, послідовність '0110111' знайдена на зсувах часу 20 і 31. Результати симуляції показані нижче.



Рис.3.9. Результати симуляції

Функція MATLAB називається `calcLockMatrices` надається і може бути використана для створення нових матриць наступного стану і виходів для виявлення іншої послідовності.

```
[Next, out] = calcLockMatrices ('0110101101111')
```

Матриці наступних станів і виходів повинні бути визначені в GUI блоку машини станів в як значення 'Next State Matrix' і 'Output Matrix'. Ви можете змінити входи проєкту, змінивши блок 'Input Sequence'.

3.4. Висновки до розділу

Використання MATLAB/Simulink дозволяє значно скоротити час проєктування та забезпечити моделювання не розміщуючи проєкт на кристалі.

Розроблений проєкт легко завантажується в кристал ПЛІС, що продемонстровано з використанням інструментального модуля Xilinx Spartan_3E Starter Board.

ВИСНОВКИ

В результаті проведених досліджень, присвячених прискоренням проєктування цифрових пристроїв в базисі ПЛІС Xilinx за рахунок використання MatLab/Simulink, були отримані наступні результати:

- ґрунтуючись на аналізі етапів схемотехнічного проєктування цифрових пристроїв, визначені переваги проєктування електронних схем за допомогою ПЕОМ перед традиційним способом проєктування;
- встановлено, що середовище Simulink дозволяє описувати об'єкти за допомогою виконуваних блок схем при цьому кількість етапів процесу проєктування значно менше ніж при використанні традиційних способів;
- встановлено, що використання System Generator MATLAB/Simulink дозволяє здійснювати налагодження і моделювання додатків безпосередньо в середовищі проєктування при цьому завантаження в цільовий пристрій не вимагає значних витрат часу, що й продемонстровано з використанням інструментального модуля Xilinx Spartan_3E Starter Board.

Джерела з Інтернету

17

Сторінка 54 з 55

Вилучення

Вилучення

1

http://student.ch.lu.se/lth/mats/kurser/c++/pdf/cpp_f1-4.pdf

0.11%