ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Актуальність

- Тестування апаратного забезпечення в циклі зазвичай є частиною циклу проектування систем керування. Ефективні та швидкі моделі можуть бути створені на мові опису апаратного забезпечення (HDL – Hardware Description Language), яка реалізована в програмованій матриці вентилів (FPGA – Field-Programmable Gate Array).
- Пропонується робочий процес проектування, який дозволяє досягти продуктивності, подібної до ручної роботи, за допомогою автоматичних інструментів. Він складається з ідентифікації подібних операцій, примусової перевірки знаковості сигналів і пристосування до розмірів вхідних даних множників. Проведено детальне порівняння між трьома робочими процесами: переклад високорівневого коду MATLAB, переклад моделі Simulink і робота безпосередньо в HDL.

Мета та задачі

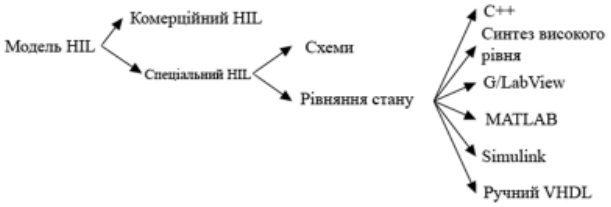
- Метою кваліфікаційної роботи є запропонувати робочий процес для моделювання апаратного забезпечення з використанням автоматичної генерації коду за допомогою MATLAB/Simulink

Задачі

- Проаналізувати дерево рішень для впровадження режиму HIL
- Запропонувати робочій процес (метод) для моделювання апаратного забезпечення
- Провести експерименти генерації та провести відповідну оцінку.

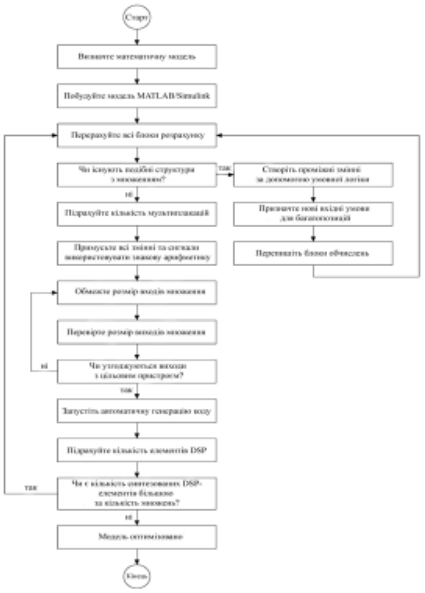
Дерево рішень для впровадження режиму HIL

- На рисунку, для побудови ефективної моделі HIL перший вибір – між комерційною та спеціальною системами. Спеціальні системи пропонують найкращу продуктивність та гнучкість. У них модель описується або на рівні схеми, або на рівні рівняння стану.
- Бібліотека MATLAB/Simulink Simscape дозволяє моделювати на рівні схеми. Це менш вимогливо, але пропонує менше контролю для проектувальника.



Робочий процес проектування для автоматичної генерації коду

Типовий робочий процес проектування



Запропонований робочий процес проектування

5

Вимоги для запропонованого методу

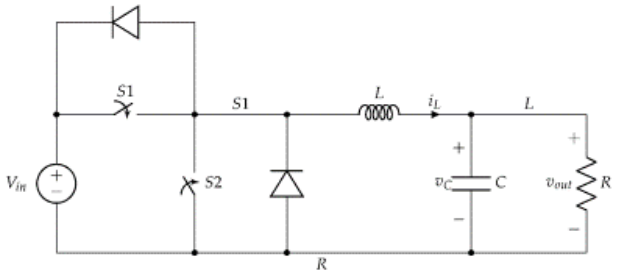
Були розглянуті наступні вимоги для запропонованого метода:

1. Метод повинен бути сумісним з існуючими автоматизованими робочими процесами.
2. Він не повинен змінювати початкову математичну модель.
3. Він не повинен вимагати від інженера-енергетика знання мов опису обладнання.
4. Будь-які зміни в структурі алгоритму повинні бути зрозумілими для енергетика.
5. Не вимагати знання внутрішньої структури цільового пристрою.
6. Він не повинен вимагати знання процесів проектування та оптимізації на мові HDL.
7. Він повинен бути достатньо простим, щоб його варто було використовувати, щоб покращення продуктивності кінцевої моделі було отримано ціною невеликих додаткових зусиль.
8. Він повинен бути сумісним з різними чисельними методами.
9. Він повинен бути сумісним зі складними архітектурами, такими як конвеєрна робота.
10. Він повинен досягати аналогічних показників зайнятості в цільовому пристрої в порівнянні з ручним проектуванням.
11. Він повинен досягати такої ж швидкості, як і при ручному проектуванні.

Процес можна розділити на чотири основні етапи: переписування коду, форсування сигналу, адаптація до розміру множника та верифікація.

6

BUCK-перетворювачі



Обрана синхронна схема з buck-автоматом

Синхронний перетворювач працює шляхом почергового замикання вимикачів S1 і S2, при цьому робочий цикл визначається співвідношенням v_{out}/v_{in}. Щоб уникнути короткого замикання джерела, між замиканням одного вимикача і розмиканням іншого було введено мертвий час, під час якого діоди D1 і D2 забезпечують шлях для розрядки індуктора.

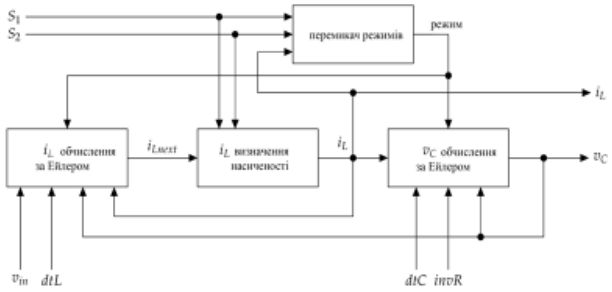
У схемі можна виділити три режими роботи:

- режим 1: S1 замкнений, а S2 розімкнений або S1 і S2 розімкнені (мертвий час), а струм котушки індуктивності (i_L) від'ємний (D1 у стані провідності);
- режим 2: S1 розімкнутий, а S2 замкнутий і під час неробочого ходу, в якому i_L>0 (D2 у стані провідності);
- режим 3: якшо котушка індуктивності розряджається під час неробочого ходу, струм через неї не протікає.

Рівняння схеми були отримані на основі поведінки котушки індуктивності та конденсатора, а також режимів роботи перетворювача. Розв'язок для змінних стану v_C та i_L було отримано три набори звичайних диференціальних рівнянь.

7

Проектування на основі моделей у Simulink



- Simulink – це середовище для моделювання та імітаційного моделювання для інженерії на основі моделей. Воно дозволяє здійснювати графічне проектування систем шляхом з'єднання блоків.
- Для забезпечення трансляції блоків у мову HDL необхідно використовувати бібліотеку HDL Coder Simulink. Simulink-модель, показана на рисунку, складається з чотирьох підсистем: селектора режимів, блоків обчислення Ейлера для v_C і i_L та детектора насичення i_L під час неробочого ходу. Параметри $1/R$, dt/C та dt/L є вхідними константами, а стан перемикачів $S1$ та $S2$ і вхідна напруга vin є вхідними сигналами.
- Блок вибору режиму приймає на входи стани перемикачів $S1$, $S2$ та iL і виробляє вихід (1, 2, 3), який представляє режим роботи. Цей вихід використовується як вхід для інших обчислювальних блоків.

8

Генерація VHDL коду з MATLAB

```
1: Constant declaration
2: dtC ← dt/C
3: dtL ← dt/L
4: invR ← 1/R
5: function CALCULATESTEP(iL, vC, dtC, dtL, invR, S1, S2, v_in)
6:     if S1 = closed OR (S1 = open AND S2 = open AND iL < 0) then        ▷ Mode 1
7:         ΔvC ← (iL − vC × invR) × dtC
8:         ΔiL ← (v_in − vC) × dtL
9:     else if S2 = closed OR (S1 = open AND S2 = open AND iL > 0) then   ▷ Mode 2
10:        ΔvC ← (iL − vC × invR) × dtC
11:        ΔiL ← −vC × dtL
12:    else if iL = 0 then                ▷ Mode 3: iL = 0 during a dead time
13:        ΔvC ← −vC × invR × dtC
14:        ΔiL ← 0
15:    end if
16:    vC_next ← vC + ΔvC
17:    iL_next ← iL + ΔiL
18:    if sign(iL_next) ≠ sign(iL) AND S1 = open AND S2 = open then   ▷ iL crosses zero during a dead time
19:        iL_next ← 0
20:    end if
21: end function
```

Розрахунок наступного значення v_C та i_L методом Ейлера з використанням простого методу насичення iL

Розмір змінних та констант, що використовуються для buck-моделі

Змінна чи постійна	Знак	Цілі біти	Дробові біти	Сумарні біти
i_L	так	7	24	32
i_{Lnext}	так	7	24	32
v_C	так	10	21	32
Δi_L	так	–5	36	32
Δv_C	так	–3	33	32
invR	ні	–4	36	32
dtC	ні	–5	37	32
dtL	ні	–9	41	32
Vin	ні	5	27	32

Використання ресурсів і максимальна швидкість для buck-перетворювача

Тип коду	LUTs	FFs	DSPs	Швидкість (нс)
VHDL з MATLAB	521	68	20	24
VHDL з Simulink	363	64	12	23

Переглянута алгоритмічна модель

- Реалізація на мові VHDL відповідає структурі, з урахуванням розмірів сигналів, наведених у попередній таблиці.
- Результати наведено в четвертому рядку таблиці на слайді. Цей робочий процес генерує найменше використання ресурсів, оскільки VHDL-проектувальник використовує лише те, що необхідно для безпосередньої реалізації алгоритму на мові низького рівня.
- Незважаючи на це, максимальна досягнута швидкість була подібною до інших методів, що свідчить про те, що автоматичні інструменти можуть транслювати арифметичні операції у високопродуктивний HDL-код з ефективним обчислювальним трактом.

Використання ресурсів і максимальна швидкість для buck-моделі

Тип коду	LUTs	FFs	DSPs	Швидкість (нс)
VHDL з MATLAB	521	68	20	24
VHDL з Simulink	363	64	12	23
VHDL з оновленого MATLAB	400	67	12	23
Ручний VHDL	287	64	9	25

Метрики коду та використання ресурсів

- Ці показники коду самі по собі не пояснюють, чому моделі MATLAB і Simulink використовують більше ресурсів. Можна припустити, що кількість сигналів і змінних корелює з більшим використанням LUT і FF.
- Однак у коді є три множення, і вони перетворюються на 12 DSP замість 9, як у випадку ручного VHDL-коду.
- Перевірка згенерованого коду показала, що сигнали зростали в проміжних обчисленнях, а входи до множників не були оптимізовані. Беззнакові сигнали можуть зростати на один біт, якщо вони втручаються в обчислення зі знаковими сигналами.
- Таким чином, за допомогою автоматизованих інструментів користувач не може визначити точний розмір входів множників, якщо не створити проміжну змінну або не розмістити блок перетворення перед множенням.

Метрики VHDL коду трьох потоків проектування

Характеристика	MATLAB (початковий)	MATLAB (оптимізований)	Simulink	Ручний VHDL
Рядки коду	306	284	1007	107
Додавання/відніма ння	5	4	4	4
Множення	6	3	3	3
Блоки If-then-else	13	13	14	5
Сигнали та змінні	102	89	95	13
Процеси	6	6	7	2

11

Висновки

- Ця робота зосереджена на реалізації моделей з використанням їхніх рівнянь, а не на прямій трансляції схем у код HDL, що може мати низьку продуктивність і високе використання апаратного забезпечення. Результати показали, що напівавтоматично згенеровані моделі досягають високої ефективності, якщо дотримуватися запропонованого методу.
- Максимальна швидкість була однаковою у всіх випадках. Використання DSP є критично важливим у HIL-додатках, і неоптимізований алгоритм міг би подвоїти кількість використовуваних DSP, але досяжна швидкість неоптимізованого підходу все одно була подібною до оптимізованого. Найкращі результати показав підхід генерації коду HDL, але він також був найскладнішим і вимагав ручного перетворення у фіксовану крапку.
- Запропоновано модифікацію робочого процесу проектування HIL-моделей, яка призводить до автоматичної генерації дуже ефективних моделей. Це усуває необхідність проектування на рівні HDL і забезпечує результати, дуже близькі до результатів ручного проектування.

12