

РОЗРОБКА ТА ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМ НА КРИСТАЛІ ТИПУ SOC FPGA CYCLONE V

У даній роботі розглядається розробка та використання програмного забезпечення (ПЗ) для взаємодії Hard Processor System (HPS) із Field-Programmable Gate Array (FPGA) на прикладі системи на кристалі (SoC) типу SoC FPGA Cyclone V. Показано переваги та обмеження використання такого ПЗ, можливості до адаптації та оптимізації систем що використовують даний підхід до обміну даними. Розробка та тестування продемонстровано на базі Arch Linux портованого на платі розробника Intel DE10-Nano, що уможливило доступ HPS до регістрів FPGA частини.

Нині системи на кристалі будуються з допомогою стандартизованих пакетів наданих розробниками цих систем, загальним ПЗ котре орієнтоване на сімейства чіпів або ж в певних випадках спеціалізованих програм котрі орієнтовані виключно на схему на яку й будується проєкт. З розвитком процесів розробки росте рівень абстракції та модульності кожної наступної системи, оскільки необхідно брати до уваги широку сферу використання SoC FPGA. Багато груп розробників фокусуються на підвищенні рівня абстракції своїх проєктів, щоби була можливість охоплення максимально можливого відсотку ринку.

Спеціалізовані ж системи відстають у процесі універсалізації лише тому, що разом із спеціалізацією йдуть жорсткі обмеження на доступні ресурси та їх використання. Проте навіть при цьому, з'являються проєкти про мінімізації ядер оперативних систем [1], дистрибутиви що орієнтовані на мінімальний рівень доступних ресурсів та програмні додатки що можуть бути модифікованими для портування на інші системи [1].

Загальна практика зазвичай полягає у використанні вже існуючих або наданих пакетів ПЗ для роботи із шинами та/або регістрами, рідше через пряме звертання до адресованої комірки пам'яті, проте даний підхід має негативний вплив на здатність системи до адаптації та модифікації під час розробки. У даному випадку, пропонований нами підхід дозволяє проводити переналаштування та модифікацію ПЗ залежно до потреб системи маючи прямий доступ до пам'яті та реалізуючи підхід memory-mapped I/O [2].

Отримуючи прямий доступ до роботи з регістрами, даний підхід абстрагує операції запису та зчитування дозволяючи значно менше залежати

від низькорівневої реалізації системи.[3] А взявши до уваги здатність модифікованого ядра Linux взаємодіяти та конфігурувати FPGA частину, гнучкість та здатність до адаптації буде вищою, що в свою чергу дозволить уникнути частини проблем пов'язаних із переносом системи на іншу SoC.

Було проведено роботу як із прямої інтеграції ПЗ в дистрибутив, так і використання його як тимчасового способу для запису та читання на місці. Використання такого підходу спрощує як роботу з динамічною частиною FPGA так і дозволяє швидко адаптуватися після зміни конфігурації FPGA частини ініційованого з простору користувача ОС.

Розроблене ПЗ здатне взаємодіяти з ПЛІС навіть після зміни конфігурації, при умові що структура даних є попередньо узгодженою, або ж ми здатні обробляти дані на вищому рівні ніж при обміні з регістрами та шинами. Це в свою чергу дозволяє не збільшувати критичну кількість ресурсів, що зарезервовані на процеси ОС та ПЗ в її просторі. Що в свою чергу позитивно впливає на ціну необхідної платформи та на ціну адаптації ПЗ до іншої. Прямий, двосторонній зв'язок ж між компонентами SoC FPGA дозволяє швидко обробляти та взаємодіяти з даними та процесами з простору користувача, в свою чергу спрощуючи роботу із створеною системою через необхідні рівні абстракції.

У подальшому, розглядається оптимізація обміну даними, робота з динамічно присвоєними адресами пам'яті та загалом адаптація до високорівневих мов програмування. В комбінації із здатністю ОС до перепрограмування FPGA частини SoC це дозволить будувати системи що здатні до швидкої адаптації та можуть бути оптимізовані без прямого та явного доступу до конфігурування FPGA. Скомпонована ж система може бути базисом для пришвидшення розробки на різних платформах, оптимізації затрачених ресурсів та підвищення доступності для розробників.

В процесі роботи було розроблено спрощену версію ПЗ для взаємодії HPS із FPGA та проведено порівняння із стандартизованими пакетами, котрі орієнтуються на специфічні платформи. Як результат ми маємо базис для створення динамічних систем що не вимагає значної кількості додаткових ресурсів та позитивно впливає на спрощення роботи з новими системами. У подальшій ж перспективі це дозволяє на вищих рівнях абстракції розробляти проекти, що в своє чергу знижує час на розробку та дозволяє швидко адаптуватися до змін в оточенні систем розробника.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bare-Metal, RTOS, or Linux? Optimize Real-Time Performance with Altera SoCs. Electrical and Computer Engineering | Electrical and Computer Engineering. URL:

https://people.ece.cornell.edu/land/courses/ece5760/DE1_SOC/wp-01245-optimize-real-time-performance-with-altera-socs.pdf

2. Snider R. K. Advanced Digital System Design using SoC FPGAs. Cham : Springer International Publishing, 2023. URL: <https://doi.org/10.1007/978-3-031-15416-4>

3. HPS SoC Boot Guide - Cyclone V SoC Development Kit Altera Altera, "HPS SoC Boot Guide - Cyclone V SoC Development Kit" Altera, 27 1 2016.