УДК 004.272, 004.274, 004.383

В. А.Мельник[1,2], І. І. Лопіт[1], А. Ю. Кіт[1]
[1] Національний університет “Львівська політехніка”,
кафедра безпеки інформаційних технологій;
[2] Люблінський Католицький Університет Івана Павла II, м. Люблін, Польща,
Інститут математики і інформатики

ПРОТОКОЛ ОБМІНУ ІНФОРМАЦІЄЮ ДЛЯ АВТОМАТИЧНОГО СТВОРЕННЯ КОМП’ЮТЕРНИХ ПРИСТРОЇВ У РЕКОНФІГУРОВНИХ АПАРАТНИХ ПЛАТФОРМАХ ОБЧИСЛЮВАЛЬНИХ ВУЗЛІВ КФС

В доповіді запропоновано протокол обміну інформацією для автоматичного створення комп’ютерних пристроїв у реконфігуровних апаратних платформах обчислювальних вузлів кіберфізичних систем. Протокол враховує особливості автоматичного генерування кодів програмних моделей комп’ютерних пристроїв, їх компіляції і логічного синтезу, і не залежить від середовища передачі інформації.

Ключові слова – кіберфізична система, реконфігуровна апаратна платформа, програмовна логічна інтегральна схема, протокол обміну інформацією.

INFORMATION EXCHANGE PROTOCOL FOR COMPUTER DEVICES AUTOMATIC CREATION IN RECONFIGURABLE HARDWARE PLATFORMS OF THE CYBER-PHYSICAL SYSTEMS COMPUTING NODES

The report embraces the information exchange protocol for computer devices automatic creation in reconfigurable hardware platforms of the cyber-physical systems computing nodes. The protocol considers features of the computer devices programming models automatic generation, compilation and logic synthesis, and is independent of the information transmission medium.

Key words – cyber-physical system, reconfigurable hardware platform, field-programmable gates array, software interface.

Вступ

За сучасного стану комп’ютерної елементної бази підвищення продуктивності комп’ютерних систем в більшості випадків вирішується екстенсивним методом, тобто збільшенням кількості ядер універсальних процесорів та підвищенням частоти їх роботи. Разом з тим, підхід застосування універсальних процесорів для досягнення високих показників продуктивності має принципові недоліки, головними з яких є висока споживана потужність і низька ефективність використання обладнання. Для уникнення вказаних недоліків створюють комп’ютерні системи із спеціалізованими процесорами, однак вони є ефективними лише на вузьких класах алгоритмів, а їх побудова потребує значних зусиль і ресурсів. Один з варіантів вирішення цієї проблеми полягає в генеруванні програмних моделей спеціалізованих процесорів з високорівневого подання алгоритму їх роботи та їх

реалізації в програмовних логічних інтегральних схемах (ПЛІС) [1]. В результаті отримують спеціалізовані процесори (СП), які використовують як прискорювачі комп'ютерних систем, і функції яких можна змінювати шляхом реконфігурування ПЛІС і створення в ний іншого СП.

Реконфігуровною апаратною платформою (РАП) називають апаратну платформу для здійснення обчислень на базі пристроїв реконфігуровної логіки (наприклад, ПЛІС). Для автоматичного створення конфігурацій, які призначені для реконфігуровних апаратних платформ, системі генерування програмних моделей спеціалізованих процесорів необхідна така інформація: характеристики РАП (типи і кількість ПЛІС), характеристики конфігурації та високорівневий опис алгоритму.

До основних характеристик РАП на основі ПЛІС можна віднести такі: тип ПЛІС, організація вбудованих блоків пам'яті, організація арифметико-логічних пристроїв, характеристики логічних комірок, максимальна тактова частота, енергоспоживання. Кожен тип ПЛІС має свою архітектуру, і тому без цієї інформації неможливо згенерувати конфігурацію. Налаштування конфігурації РАП на основі ПЛІС задається такими характеристиками спеціалізованого процесора: тактова частота, критерій оптимізації, граничне використання ресурсів, кількість обчислювальних модулів, енергоспоживання.

Постановка задачі

Окрім високопродуктивних комп'ютерних систем, одним із найперспективніших напрямків застосування РАП є кіберфізичні системи (КФС), в яких вони використовуються для реалізації обчислювальних вузлів. При цьому створення комп'ютерних пристроїв в РАП обчислювальних вузлів КФС повинно виконуватись автоматично. Виконання цього завдання потребує розроблення спеціального протоколу обміну інформацією для автоматичного створення комп'ютерних пристроїв у реконфігуровних апаратних платформах обчислювальних вузлів КФС.

1. Протокол обміну інформацією для автоматичного створення комп'ютерних пристроїв у реконфігуровних апаратних платформах обчислювальних вузлів КФС

Авторами запропоновано протокол обміну інформацією між реконфігуровними апаратними платформами обчислювальних вузлів КФС для автоматичного створення в них комп'ютерних пристроїв. Цей протокол базується на клієнт-серверному підході. Клієнтом є РАП, в якої є необхідність змінити конфігурацію, а сервером – система автоматичного створення конфігурацій для РАП, яка надає дану послугу. Стан з'єднання керується за допомогою наступних 4 компонент: повідомлення, середовище передачі, скінчений автомат сервера та скінчений автомат клієнта.

1.1. Повідомлення і середовище їх передачі

Комунікація між сервером і клієнтом здійснюється способом передачі повідомлень. Поняття «повідомлення» нероздільно зв'язане із середовищем їх передачі. До середовища передачі є тільки одна вимога: воно повинно гарантувати передачу даних. Наприклад, при реалізації даного протоколу в моделі *OSI* [2] ним може слугувати протокол *TCP* [3] чи інші протоколи гарантованої передачі даних.

Повідомлення в протоколі можна розділити на 2 групи:

- група повідомлень для керування станом з'єднання. До цієї групи входять повідомлення, які відповідають за встановлення з'єднання, авторизацію, його перебіг, контроль і завершення.

§ група повідомлень для передачі даних. До цієї групи входять повідомлення, які містять інформацію про тип ПЛІС, основні характеристики конфігурації РАП і власне конфігурацію РАП.

1.2. Скінчений автомат сервера

Розглянемо скінчений автомат сервера, який відповідає за керування станом з'єднання зі сторони сервера (Рис. 1). Автомат працює за наступним алгоритмом. Початковий стан автомата, з якого сервер починає свою роботу – *Init*. Після виклику *start*() сервер переходить до стану *WaitForConnection*, в якому він очікує на підключення клієнта. Якщо відбулось з'єднання, викликається метод *DoAcceptConnection*() і сервер переходить у стан *Connected*. Якщо в результаті з'єднання сталась помилка або перевищено кількість спроб з'єднання, сервер перейде в стан *ConnectionFailed* за допомогою методів *OnMaxConnectionAttempt*() і *OnConnectionFailed*() відповідно. Якщо кількість спроб з'єднання не перевищено, за допомогою методу *WaitForNewConnection*() відбудеться перехід у стан *WaitForConnection*. В іншому випадку відбудеться перехід до стану *Error* за допомогою методу *HandleConnectionFailed*(), який відповідає за незворотну помилку з'єднання. При успішному з'єднанні сервер перейде у стан *WaitForAuthorization* – стан автомата, в якому сервер очікує на авторизацію клієнта. Якщо клієнт авторизований, сервер переходить у стан *Authorized* за допомогою *DoAuthorizationAccept*(). У разі помилки авторизації відбудеться виклик методу *OnAuthorizationFailed*() і сервер перейде в стан *AuthorizationFailed*. Якщо кількість спроб на авторизацію не перевищено, із стану *AuthorizationFailed* можливе повернення до стану *WaitForAuthorization* викликом методу *WaitForNewAuthorizationAttempt*(), в іншому випадку виконається метод *HandleAuthorizationFailed*() і сервер перейде до стану *Error*. Якщо перевищено ліміт перебування сервера в стані *WaitForAuthorization*, відбудеться виклик методу *OnAuthorizationTimeout*() і перехід до стану *AuthorizationFailed*().

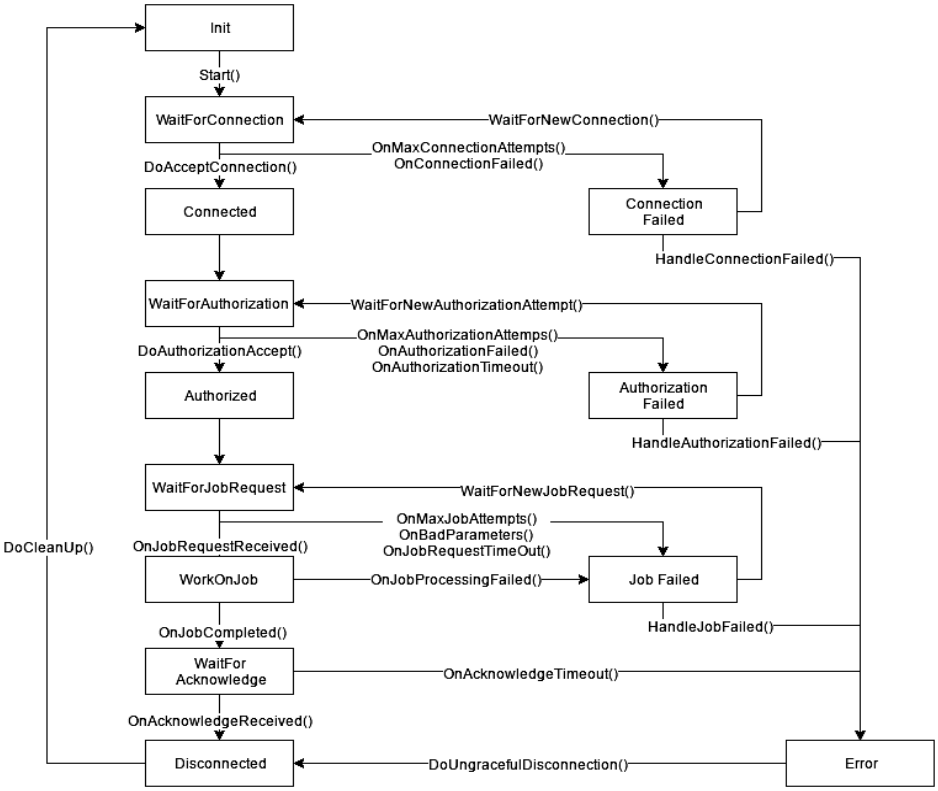

Рис. 1. Скінчений автомат сервера

У випадку успішної авторизації сервер перейде до стану *WaitForJobRequest*. В цьому стані сервер очікує на надходження запиту на виконання генерування конфігурації для РАП. Якщо перевищений ліміт перебування сервера в стані *WaitForJobRequest*, відбудеться виклик методу *OnJobRequestTimeOut*() і перехід до стану *JobFailed*. У випадку, коли були передані неправильні параметри для синтезу конфігурації, відбудеться перехід до стану *JobFailed* за допомогою *OnBadParameters*(). Зі стану *JobFailed* можливий перехід до стану *WaitForJobRequest* за допомогою *WaitForNewJobRequest*(), якщо кількість запитів на виконання не перевищена, в іншому ж випадку відбудеться перехід до стану *Error* і з'єднання буде закінчене. Якщо успішно отримано запит на синтез *OnJobRequestReceived*(), сервер перейде у стан *WorkOnJob*, в якому здійснюється генерування конфігурації для РАП. Під час виконання можливе виникнення помилок, і в цьому випадку сервер перейде зі стану *WorkOnJob* до стану *JobFailed*. В результаті успішного синтезу виконається метод *OnJobCompleted*() і сервер перейде до стану *WaitForAcknowledge*, в якому він буде очікувати на відповідь клієнта про успішне отримання конфігурації. Якщо цього не відбудеться, за допомогою методу *OnAcknowledgeTimeout*() сервер перейде в стан *Error*. В випадку отримання підтвердження відбудеться виклик *OnAcknowledgeReceived*(). Сервер перейде у стан *Disconnected*(), в якому відбувається закінчення з'єднання. Після закінчення з'єднання відбувається скидання стану методом *DoCleanUp*() і сервер перейде в стан *Init*. У випадку незворотної помилки сервер перебуватиме в стані *Error*. З цього стану за допомогою методу *DoUngracefulDisconnection*() відбувається перехід в стан *Disconnected*.

1.3. Скінчений автомат клієнта

Цей скінчений автомат відповідає за керування з'єднанням із сторони клієнта (Рис. 2).

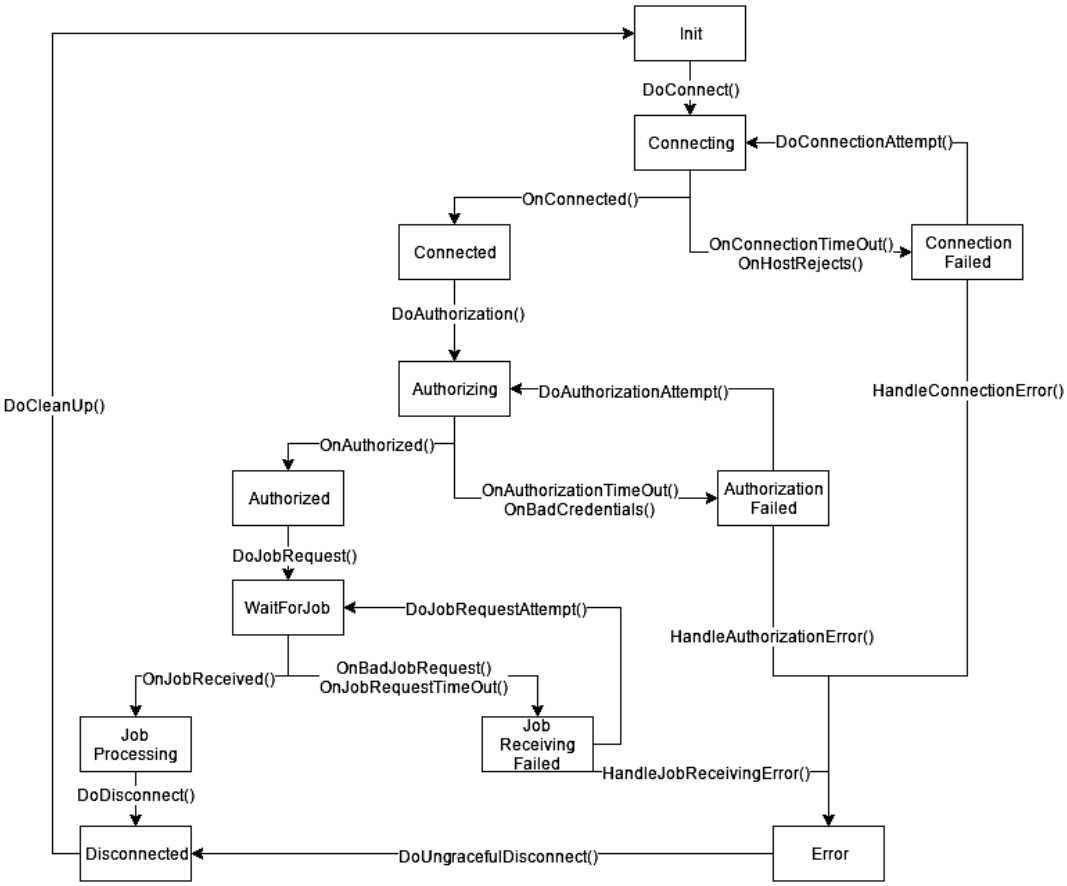

Рис. 2. Скінчений автомат клієнта

Скінчений автомат клієнта працює за таким алгоритмом. Клієнт починає роботу із стану *Init*. Встановлення з'єднання з сервером відбувається за допомогою методу *DoConnect*() і клієнт переходить в стан *Connecting*, очікуючи на відповідь сервера. Якщо перевищений час очікування на відповідь, відбудеться виклик методу *OnConnectionTimeOut*(), і клієнт перейде в стан *ConnectionFailed*. У випадку, коли сервер не може прийняти з'єднання, відбудеться виклик методу *OnHostRejects*() і клієнт перейде в стан *ConnectionFailed*. Зі стану *ConnectionFailed* можливий перехід до стану *Connecting* за допомогою методу *DoConnectionAttempt*(), якщо кількість спроб не перевищила ліміт. Якщо з'єднання відбулось успішно, клієнт переходить до стану *Connected*. Зі стану *Connected* за допомогою виклику методу *DoAuthorization*() відбувається перехід до стану *Authorizing*, в якому клієнт очікує на авторизацію від сервера. Якщо авторизація не пройшла успішно, або перевищений час очікування відповіді сервера, то за допомогою методів *OnAuthorizationTimeOut*() або *OnBadCredentials*() відбудеться перехід до стану *AuthorizationFailed*. Якщо ліміт спроб не перевищений, відбудеться виконання методу *DoAuthorizationAttempt*(), в іншому випадку за допомогою методу *HandleAuthorizationError*() відбудеться перехід до стану *Error*.

Якщо авторизація пройшла успішно, відбудеться виклик *OnAuthorized*() і сервер перейде в стан *Authorized*. Із цього стану клієнт за допомогою виклику *DoJobRequest*() здійснить перехід в стан *WaitForJob* і буде очікувати на результат виконання синтезу. Якщо під час запиту виникнуть помилки, за допомогою методів *OnBadJobRequest*() *OnJobRequestTimeOut*() відбудеться перехід в стан *JobReceivingFailed*. Коли ліміт запитів не вичерпаний, за допомогою *DoJobRequestAttempt*() відбудеться перехід в стан *WaitForJob*.

Після виконання синтезу відбудеться виклик методу *OnJobReceived*() і клієнт перейде в стан *JobProcessing*. У цьому стані відбувається зберігання конфігурації. За допомогою виклику методу *DoDisconnect*() клієнт перейде в стан *Disconnected*, з якого, викликавши *DoCleanUp*(), перейде в початковий стан *Init*. У випадку незворотної помилки сервер перебуватиме у стані *Error*. З цього стану за допомогою методу *DoUngracefulDisconnection*() відбувається перехід у стан *Disconnected*.

2. Приклад комунікації клієнта із сервером

Розглянемо приклад комунікації клієнта із сервером (Рис. 3).

Після того, як сервер перейде в стан *WaitForConnection*, клієнт відправляє повідомлення *ConnectionRequest* і переходить в стан *Connecting* в якому очікує на відповідь сервера. Сервер в свою чергу отримує повідомлення, переходить у стан *Connected*, відправляє *ConnectionAccept* і переходить у стан *WaitForAuthorization*, в якому очікує на авторизацію. Клієнт, після отримання *ConnectionAccept*, переходить у стан *Connected*, відправляє повідомлення *AuthorizationRequest* і переходить у стан *Authorizing*. Після отримання *AuthorizationRequest* сервер переходить у стан *Authorized*, відправляє повідомлення *AuthorizationAccepted*, і переходить у стан *WaitForJob*, в якому очікує на запит виконання компіляції. Отримавши *AuthorizationAccepted*, клієнт переходить у стан *Authorized*, робить запит на компіляцію і далі переходить у стан *WaitForJob*. Після отримання запиту на компіляцію – *RequestForCompilation*, сервер переходить у стан *WorkOnJob* і повідомленням *CompilationAccepted* повідомляє, що запит прийнятий. Після завершення компіляції сервер переходить до стану *WaitForAcknowledgment*, в якому очікує на відповідь від клієнта. Отримавши *OnJobDone*, клієнт виконує обробку отриманої конфігурації, відправляє повідомлення про успішне отримання конфігурації *JobAcknowledgment* і закінчує

свою роботу, переходячи до стану *Disconnected*. Сервер, отримавши повідомлення *JobAcknowledgment*, закінчує свою роботу, також переходячи до стану *Disconnected*.

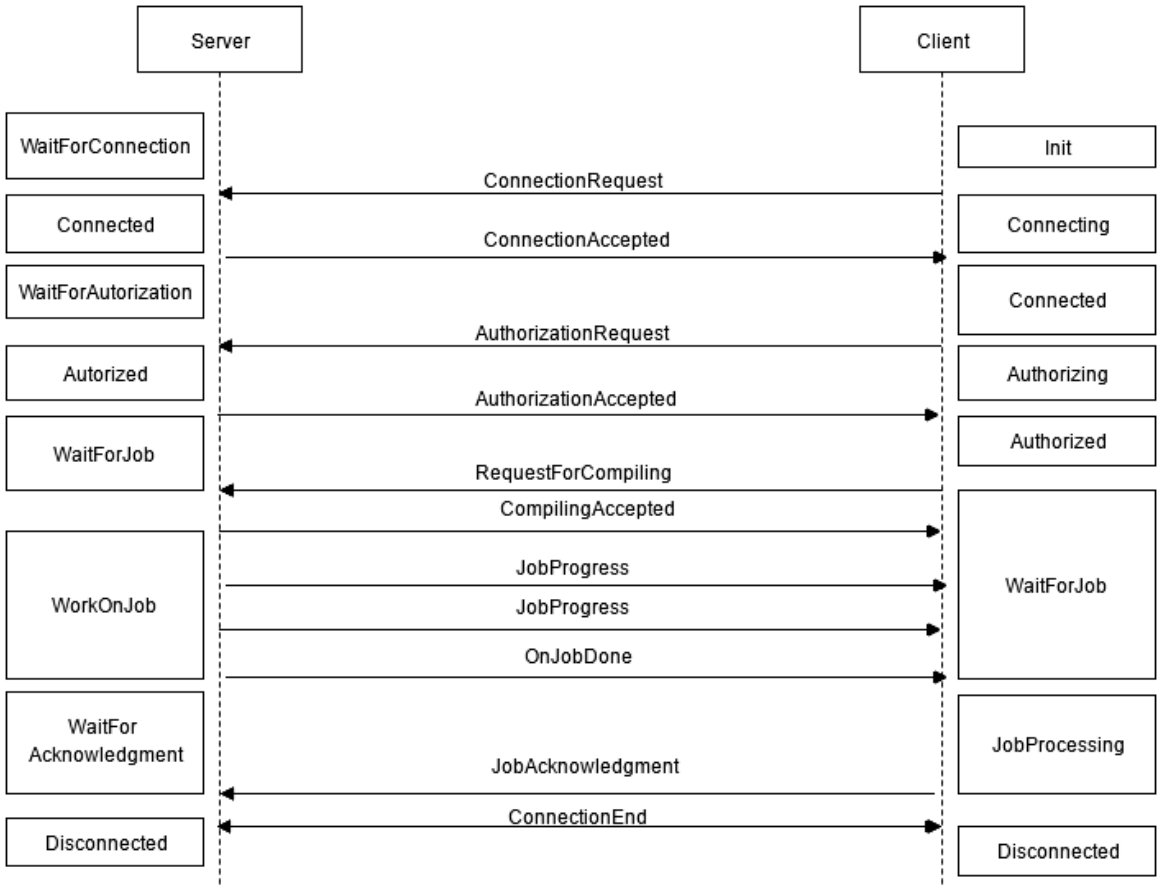

Рис. 3. Приклад комунікації клієнта із сервером

Висновок

В доповіді запропоновано протокол обміну інформацією для автоматичного створення комп'ютерних пристроїв у реконфігуровних апаратних платформах обчислювальних вузлів кіберфізичних систем. Протокол не залежить від середовища передачі інформації, що дозволяє використовувати його поряд із іншими протоколами, які гарантують передачу даних, наприклад *TCP*. Запропонований протокол в подальшому можна буде використати не тільки для автоматичного створення комп'ютерних пристроїв, але й для вирішення інших актуальних для КФС задач, так як він оперує абстрактним поняттям «завдання», яке може бути змінене залежно від поставленої мети.

1. Chameleon – the System-Level Design Solution. [Електронний ресурс] / – Режим доступу: http://intron-innovations.com/?p=sld_chame. *2. Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. International Standart. ISO/IEC 7498-1. Second edition. 1994-11-15. 3. Transmission Control Protocol. Darpa Internet Program. Protocol Specification. September, 1981.*

Наукові результати, подані у цій доповіді, було отримано в рамках дослідницького проекту ДБ/КІБЕР з реєстраційним номером 0115U000446, 01.01.2015 – 31.12.2017, фінансово підтриманим Міністерством науки та освіти України.