УДК 004.274, 004.451

В. А. Мельник, А. Ю. Кіт
Національний університет “Львівська політехніка”,
кафедра безпеки інформаційних технологій

ОПЕРАЦІЙНІ СИСТЕМИ РЕКОНФІГУРОВНИХ КОМП’ЮТЕРІВ: БУДОВА І ОРГАНІЗАЦІЯ ФУНКЦІОНУВАННЯ

Проаналізовано підходи до організації функціонування реконфігуровних комп’ютерів на рівні їх операційної системи. Продемонстровано особливості взаємодії операційної системи з реконфігуровним середовищем та виділено механізми забезпечення ефективного виконання її сервісів в умовах нової функціональності.

Ключові слова: реконфігуровна комп’ютерана система, операційна система.

OPERATING SYSTEM FOR RECONFIGURABLE COMPUTERS: STRUCTURE AND ORGANIZATION OF OPERATION

In a paper an analysis of approaches to organization of operation of reconfigurable computers at the level of the operating system is given. We demonstrated features of the interaction of the operating system with reconfigurable logic and identified ways to ensure effective implementation of its services respecting the new functionality.

Key words: reconfigurable computer system, operating system.

Вступ

Вперше концепцію реконфігуровних обчислень розробив в шістдесяті роки Джеральд Естрін, проте вона значно випереджала тогочасні можливості цифрової техніки. Лише починаючи з середини дев’яностих років двадцятого століття реконфігуровні обчислення отримали повноцінне практичне застосування в комп’ютерній техніці. На відміну від нейманівських архітектур, у реконфігуровних комп’ютерних системах змінними є не лише алгоритми, за якими працює апаратне забезпечення, але й саме апаратне забезпечення [1]. Сьогодні створення реконфігуровних комп’ютерних систем (РККС), які позбавлені таких недоліків, притаманних універсальним комп’ютерам, як низька реальна продуктивність, висока споживана потужність та низька ефективність використання обладнання, – є одним із найперспективніших напрямів діяльності в сфері високопродуктивних обчислень.

Огляд літературних джерел

Вчений Райнер Хартенштайн ще в 2001 році вперше проаналізував підходи до організації реконфігуровних обчислень [2], розглядаючи їх з позиції антимашини як фундаментально новий підхід до організації роботи обчислювальних машин. Однією з перших операційних систем (ОС) для реконфігуровних комп’ютерів була ОС, розроблена для Xputer [3]. Модель ОС для Xputer – XOS – була програмноцентричною, тобто керування обчислювальним процесом здійснювала виконувана на центральному процесорі програма мовою високого рівня. Схожу концепцію втілюють такі програмні засоби та підходи, як HybridOS [4], StarPU [5], програмна парадигма Molen [6]. Інший підхід, коли пристрій реконфігуровної логіки є рівноправним з універсальним процесором, повною мірою втілено зокрема в операційних системах The Berkeley Operating system for ReProgrammable Hardware [7] та ReconOS [8].

Постановка задачі

Для ефективного функціонування РККС постає питання розроблення уніфікованих програмних середовищ та програмних парадигм, орієнтованих на організацію роботи гібридних комп'ютерних систем. Сьогодні такий клас програмних засобів не дуже поширений, але вже здатен вирішувати широке коло завдань. Вони можуть забезпечувати та підтримувати модульність, одночасно виконувати декілька додатків (багатозадачність), а також паралельно виконувати в ПЛІС послідовності операцій (що є важливою і потужною особливістю ПЛІС), якщо у них немає залежності за даними. Більшість з наведених вище програмних засобів та парадигм реалізують ці завдання по-своєму та зорієнтовані на певну конкретну архітектуру, що значно знижує їх гнучкість та часто вимагає від розробників апаратного забезпечення та користувачів (програмістів) підтримки того чи іншого підходу. Постає завдання проаналізувати наявні парадигми та підходи, на основі яких будуються відповідні програмні засоби, щоби виявити притаманні їм проблеми, вирішивши які можна буде створити уніфікований продукт із широким колом можливостей.

Підходи до організації функціонування ОС реконфігуровних комп'ютерних систем

Основні компоненти реконфігуровної комп'ютерної системи – це універсальний процесор та реконфігуровне середовище на базі ПЛІС. Залежно від того, якою є роль кожного компонента системи, існують два підходи до реалізації системного програмного забезпечення:

1. Використання реконфігуровного середовища як співпроцесора;
2. Використання реконфігуровного середовища як основного обчислювального ресурсу.

Розглянемо, як впливають ці підходи на будову і організацію функціонування операційної системи РККС.

Реконфігуровне середовище як співпроцесор

Перший підхід полягає в застосуванні реконфігуровного середовища на базі ПЛІС як співпроцесора. В таких системах управляють доступом до реконфігуровного середовища так само, що і для будь-якого іншого апаратного забезпечення: марежевої карти, жорсткого диска, графічних прискорювачів. Операційна система, що реалізує цей підхід, повинна завантажити конфігурацію (у вигляді файла конфігурації) в ПЛІС, після чого ПЛІС готова до виконання конкретного завдання. І потім, подібно до того, як програма взаємодіє з будь-якими іншими апаратними пристроями системи, вона взаємодіє з реалізованим у реконфігуровному середовищі спеціалізованим процесором за допомогою відповідного драйвера. Проте такий підхід має декілька слабких місць. Так, реконфігуровне середовище підпорядковується центральному процесору, який ініціює зв'язок з ним. Програма ініціює апаратне прискорення, відправляючи блок даних прискорювачу, реалізованому в ПЛІС, і, використовуючи механізм переривання, очікує результату. Такий підхід виправдовує себе при здійсненні апаратного прискорення виконання додатків, та для багатьох завдань залежність реконфігуровного середовища від універсального процесора є надлишковою. Для зв'язку з будь-яким пристроєм в системі необхідний драйвер. Той факт, що кожен додаток має свої унікальні вимоги щодо взаємодії програмної частини з апаратною, неминуче призводить до необхідності написання нового драйвера пристрою для кожної програми, що не є тривіальним для розробників програмного забезпечення. Частково цю проблему можна вирішити через стандартизовані інтерфейси.

До першого підходу організації функціонування РККС можна віднести HybridOS та програмну парадигму Molen. У Molen операції у реконфігуровному середовищі здійснюють у два етапи: встановлення (SET) та виконання (EXECUTE). Інструкції SET необхідний лише один параметр, а саме – початкова адреса в пам'яті мікрокоду конфігурації ПЛІС. Після початку виконання інструкції SET арбітр (елемент системи, що виконує часткове декодування команд, вибраних із основної пам'яті, та приймає рішення, у якому з обчислювачів – універсальному процесорі чи пристрої реконфігуровної логіки – виконуватиметься та чи інша операція чи її частина) послідовно зчитуватиме адреси пам'яті, доки не зустріне мікрокоманду *end_op*, яка означає завершення інструкції SET. Ця операція буде фактично здійснювати апаратну конфігурацію. На етапі EXECUTE власне виконується потрібна операція. Крім того, використовується регістр обміну

для передачі параметрів операції до реконфігуровного середовища на початку виконання певної операції та одержання параметрів з реконфігуровного середовища після виконання операції. Регістр обміну може отримати дані безпосередньо із регістрів універсального процесора, для чого необхідно додати відповідні команди переміщення (MOVTX та MOVFX). Для виконання програм у РККС система повинна містити арбітра та декілька додаткових команд, але для виконання програм, написаних мовою високого рівня в реконфігуровному середовищі, цього недостатньо. Необхідно також здійснити розширення компілятора відповідно до вимог парадигми Molen.

На рис. 1 показано: ліворуч – приклад програми, написаної мовою С; у центрі – об'єктний код цієї програми, згенерований стандартним компілятором С, і праворуч – об'єктний код, згенерований розширеним компілятором, а також доповнений операціями, зорієнтованими на реконфігуровне середовище.

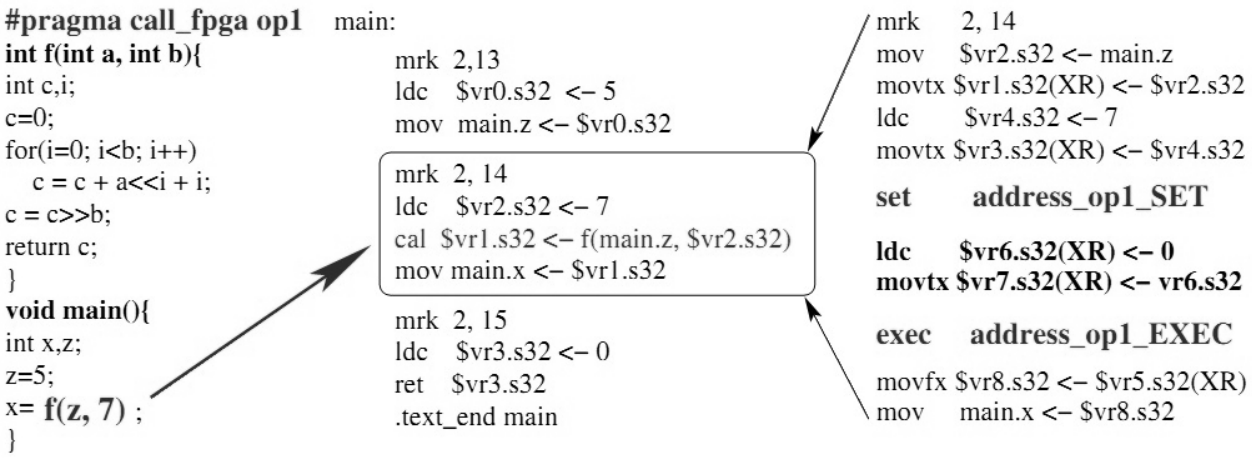

Рис. 1. Генерація об'єктного коду в Molen

Як бачимо, функція, що реалізовуватиметься в ПЛІС, позначена директивою pragma, яку звичайний компілятор С проігнорує. Це у деяких випадках може стати слабким місцем, оскільки програміст повинен сам вирішувати та явно вказувати, яка частина коду працюватиме з реконфігуровним середовищем. Отже, розробник під час написання додатків повинен явно орієнтуватись на певну платформу, універсальний комп'ютер або ж РККС. Звичайно, це можна вирішити шляхом аналізу та розширення коду додатка перед його виконанням у РККС, але не слід забувати, що цю процедуру потрібно буде повторювати для кожного додатка. Одним із рішень може бути написання додаткового модуля в системному програмному забезпеченні, який і здійснюватиме цю функцію.

Реконфігуровне середовище як основний обчислювальний ресурс

Другий підхід розглядає реконфігуровне середовище як основний обчислювальний ресурс. У таких системах універсальний процесор не створює з реконфігуровним середовищем зв'язків типу "master-slave", а інструкції апаратного забезпечення є рівноправними з програмним забезпеченням і можуть бути ініціатором з'єднання з універсальним процесором. Реалізовано такий підхід в ReconOS та The Berkeley Operating system for ReProgrammable Hardware (BORPH).

Розглянемо, як цю концепцію реалізовано в BORPH. В операційній системі BORPH з'являється поняття апаратного процесу, який поводиться як процес звичайної користувацької програми, за винятком того, що це конструкція обладнання, яка працює в ПЛІС. За звичайною термінологією ОС процес переважно визначається як безпосереднє виконання інструкцій програми, запущеної на процесорі. BORPH розширює це поняття відповідно до реконфігуровного середовища, визначаючи процес також як виконання інструкцій апаратного забезпечення. Апаратні процеси поводять себе так, як і будь-які інші процеси, створені ОС при виконанні звичайного програмного забезпечення. Такий процес створюється тоді, коли виконується об'єктний файл BORPH (BORPH Object File), що має розширення BOF. BOF-файл має двійковий формат, який інкапсулює, окрім іншої інформації, конфігурацію ПЛІС. ОС BORPH може бути реалізована на базі

ядра Linux з відповідним розширенням для роботи з ПЛІС. Коли апаратний процес створюється за допомогою програмного, використовуються ті самі системні виклики fork-exec. Оскільки створює апаратний процес ядро, то виконання в ПЛІС може ініціювати будь-яка програма, для якої є можливим створення звичайного програмного процесу, зокрема програма мовою С. Створений апаратний процес має свою ділянку в пам’яті, отже, “перегонів” між процесами, що повинні виконуватись у реконфігуровному середовищі та універсальному процесорі, виникати не повинно. До апаратних процесів можна застосовувати всі ті команди linux, що і до звичайних процесів.

Особливості взаємодії операційної системи із реконфігуровним середовищем

Операційна система, завданням якої є організація ефективної роботи реконфігуровної комп'ютерної системи, повинна виконувати також функції, притаманні й операційним системам, що керують роботою універсальних комп’ютерів архітектури Фон Неймана, такі як доступ до файлової системи, робота в мережі Internet, складні механізми взаємодії з користувачами, управління процесами та пам’яттю. Так, фундаментальні сервіси, що надає операційна система універсального комп'ютера, це завантажувач, планувальник, віртуальна пам'ять, управління кешем, міжпроцесна взаємодія, механізми введення/виведення (I/O), забезпечення захисту, – у реконфігуровних системах відрізнятимуться. Наприклад, складнішою буде робота завантажувача, адже він повинен не лише завантажувати до виконання програми, написані мовою високого рівня, але і завантажувати конфігурації в ПЛІС. Механізм планування процесів також має бути пристосований до умов нової функціональності – він повинен не допускати простою ЦП при очікуванні програмою результатів завдання, виконуваного в ПЛІС.

Зміни, передбачені архітектурою та завданнями, які стоять перед реконфігуровними комп’ютерами, можуть стосуватись не лише внутрішніх сервісів операційної системи, але і розроблення користувацьких додатків. На рис. 2 показано ліворуч зміни в частині користувача, а праворуч – розширення, передбачені в частині операційної системи та системного програмного забезпечення для роботи сервісів ОС за умов нової функціональності.

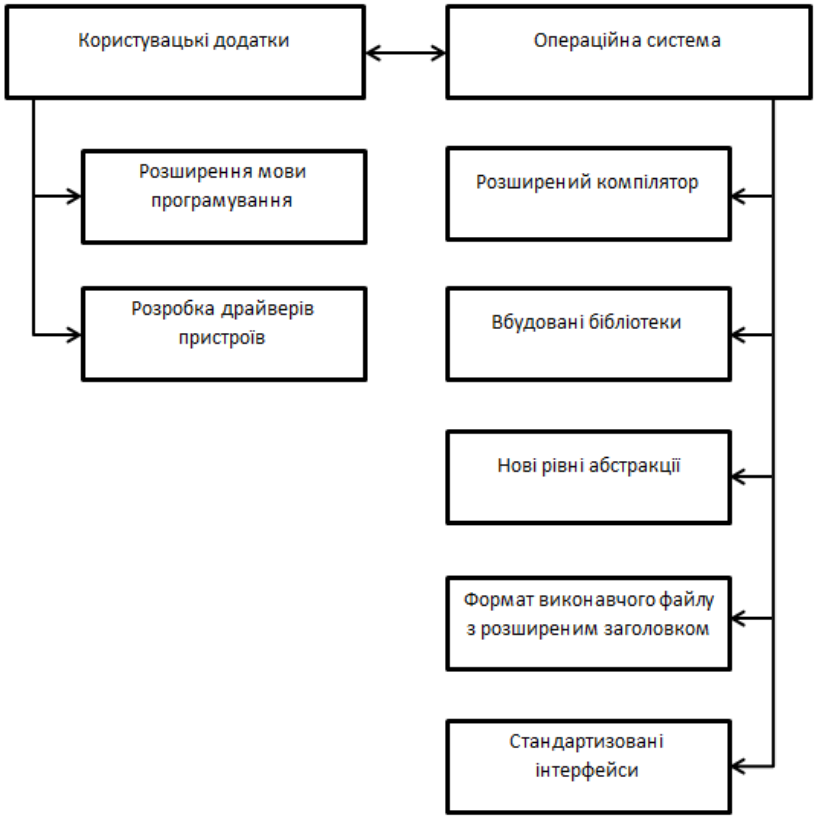

Рис. 2. Розширення ОС та користувацьких додатків для реконфігуровних комп’ютерних систем

Важливим є також питання гнучкості операційної системи. Лише така ОС, яка працює на різних архітектурах і не передбачає використання великої кількості додаткових апаратних модулів, може претендувати на широке застосування. Так, наприклад, BORPH здатна працювати на РККС із слабозв'язаною реконфігуровною логікою, РККС – із тіснозв'язаною реконфігуровною логікою, під'єднаною до шини пам'яті, та під'єднаною до шини процесора. Не менш важливим завданням є розроблення стандартизованої семантики введення/виведення та комунікації в системі програмного забезпечення з апаратним.

Порівняльний аналіз підходів до організації функціонування ОС реконфігуровних комп'ютерних систем

Основні переваги та недоліки проаналізованих систем та підходів до їх побудови знаходяться в двох площинах:

1. Зручність взаємодії користувачів із системою;
2. Ефективність роботи системи.

Як було сказано вище, організація взаємодії з РККС передбачає необхідність внесення змін до процесу розроблення програмного забезпечення. Внесення значних змін та перекладання на програміста нетривіальних для нього завдань є небажаним. Недоліком процесу розроблення додатків, які б виконувались в ОС, що розглядає реконфігуровне середовище на базі ПЛІС як співпроцесор, є необхідність написання нового драйвера пристрою для кожної програми. Можливим способом вирішення цієї проблеми є розроблення спеціальних засобів інтеграції додатків – API (application programming interface). Другий підхід позбавлений такого недоліку, але, як і за першим підходом, програміст повинен сам вирішувати та явно вказувати, яка частина програми працюватиме з реконфігуровним середовищем.

Яка б роль не відводилась реконфігуровному середовищу в РККС, співпроцесора чи основного обчислювального ресурсу, розглянуті підходи передбачають можливість розширення ядра операційної системи та системного програмного забезпечення персонального комп'ютера для організації ефективної роботи реконфігуровної системи. Побудова операційної системи на основі ядра Linux зберігає знайомі користувачу Unix-подібних ОС інтерфейс та механізми виконання багатьох базових функцій.

Ефективна організація роботи системи насамперед означає максимально ефективне використання всіх ресурсів системи. Недоліком підходу, що використовує поняття апаратного процесу, є збільшення кількості виконуваних у системі процесів та комунікацій між ними, оскільки перемикання між процесами також вимагає затрат ресурсів (перемикання з користувацького режиму в режим ядра, зберігання значень регістрів у таблиці процесів, запуск планувальника, перезавантаження блоку керування пам'яттю і, нарешті, запуск іншого процесу). Вирішенням цієї проблеми може бути буферизація – використання додаткових регістрів для тимчасового зберігання даних, які необхідно передати іншому процесу.

Кожен з розглянутих підходів може мати свої області доцільного застосування. Другий підхід є ефективнішим у системах з великою кількістю пристроїв реконфігуровної логіки, де більшість додатків зорієнтовані на виконання завдань у ПЛІС, наприклад, для високошвидкісної обробки цифрових сигналів. У таких системах на універсальний процесор насамперед покладено функції курування виконанням обчислень. Використання підходу, що розглядає реконфігуровне середовище як співпроцесор, буде доцільнішим у системах, які виконують велику кількість перемикань між процесами та постійно взаємодіють з користувачем, який у багатьох завданнях чекає негайної відповіді системи. Такими можуть бути персональні реконфігуровні системи.

Проте найбільшим недоліком цих підходів, на думку авторів, є не стільки недостатня ефективність роботи певних сервісів операційної системи чи організації взаємодії користувача та системи з реконфігуровним середовищем, а модель обчислень, яку покладено в основу розглянутих підходів. Незалежно від того, яка роль відводиться реконфігуровному середовищу в системі, керують роботою РККС командами користувача за посередництвом операційної системи, а не за допомогою самих даних. Тому важливим кроком на шляху вдосконалення операційної системи є

підтримка нею механізмів самоконфігурування. Можна провести аналогію з об’єктно-орієнтованою парадигмою програмування, де конкретні дії над даними виконуються залежно від типу та організації самих даних.

Висновки

Проаналізовано підходи до організації функціонування реконфігуровних комп’ютерних систем на рівні операційної системи, наведено приклади програмних засобів та парадигм, що реалізують ці підходи. На основі проведеного аналізу було розглянуто основні особливості та необхідні розширення операційних систем реконфігуровних комп’ютерів, що дадуть змогу організувати ефективну роботу їх сервісів та користувацьких додатків. Наведено переваги та недоліки розглянутих підходів, області доцільного застосування ОС кожного типу та шляхи вдосконалення ОС РККС.

1. N. Tredennick: The Case for Reconfigurable Computing; Microprocessor Report, Vol. 10 No. 10, 5 August 1996, pp 25–27. 2. Hartenstein, R. 2001. A decade of reconfigurable computing: a visionary retrospective. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2001) (Munich, Germany). W. Nebel and A. Jerraya, Eds. Design, Automation, and Test in Europe. IEEE Press, Piscataway, NJ, 642–649. 3. Hartenstein, R. W. and Kress, R. and Nageldinger, U. An Operating System for Custom Computing Machines based on the Xputer Paradigm. Proceedings of the 7th International Workshop on Field Programmable Logic, FPL 97, September 1997. 4. Kelm, J. H., Lumetta, S. S.: HybridOS: Runtime Support for Reconfigurable Accelerators. International Symposium on Field-Programmable Gate Arrays, Monterey, California (2008) 5. Augonnet, C., Thibault, S., Namyst, R., Wacrenier, P.-A.: StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. 15th International Euro-Par Conference (2009) 6. Mojtaba Sabeghi, Koen Bertels: Interfacing Operating Systems and Polymorphic Computing Platforms Based on the MOLEN Programming Paradigm. ISCA Workshops 2010: 311-323 7. So, H.K.-H., Brodersen, R.: A unified hardware/software runtime environment for FPGA-based reconfigurable computers using BORPH. ACM Transactions on Embedded Computing Systems (TECS) 7 (2008) 8. Lubbers, E., Platzner, M.: Reconos: An operating system for dynamically reconfigurable hardware. Dynamically Reconfigurable Systems, 269–290 (2010)