УДК 004.31, 004.32
Мельник В., Сарайрех З.
Національний університет „Львівська політехніка”

ЗАСОБИ ГЕНЕРАЦІЇ ПРОГРАМНИХ ПРОЦЕСОРІВ НА ОСНОВІ БІБЛІОТЕКИ ЇХ КОМПОНЕНТ

В статті запропоновано реалізацію генератора спеціалізованих програмних процесорів, який будується на основі бібліотеки компонент програмних процесорів та програмних засобів генерації файлів верхнього рівня мовою опису апаратних засобів. Розглянуто питання конфігурування програмних процесорів, традиційні та альтернативні методи конфігурування спеціалізованих програмних процесорів, проведено їх аналіз.
Ключові слова: програмний процесор, конфігурування, генератор програмних процесорів.

An implementation of the specialized soft-processors generator is proposed in the paper. The generator is built with the library of the soft-processors components and with the software tool that automatically generates top-level files on the hardware description language. The questions of soft-processors configuring, mainstream and alternative methods of configuring, are discussed, their analysis is done.
Key words: soft-processor, configuring, soft-processors generator.

Вступ

Сучасні засоби проектування програмних процесорів дозволяють здійснювати конфігурування їх функціональних вузлів [1]. Конфігурування дозволяє програмно змінювати такі параметри програмного процесора, як розрядність шин даних та команд, об’єми постійної пам’яті та пам’яті з довільною вибіркою, а також виділити з конфігуровного спеціалізованого програмного процесора ту його частину, яка забезпечує реалізацію заданого алгоритму з заданими параметрами.

Останнім часом створюються моделі апаратно-орієнтованих спеціалізованих процесорів з програмованою архітектурою, здатною виконувати деякий набір алгоритмів, характерних для конкретної області використання, що суттєво розширює коло можливих споживачів. Такий підхід спрощує процес проектування програмних процесорів та надає користувачу широкі можливості по їх оптимізації до конкретної задачі. Програмованість архітектури передбачає можливість її нарощування, тобто створення багаторівневої архітектури, яка дозволяє шляхом переходу з нижчого рівня на вищий підняти продуктивність або розширити функціональні можливості. Перехід з рівня на рівень передбачає надання моделі властивості залучення або вилучення відповідної частини апаратних засобів: кількості та потужності трактів обробки даних, об’ємів постійної пам’яті та пам’яті з довільною вибіркою, інших ресурсів. Нарощуваність забезпечується архітектурою обчислювального пристрою та відповідними засобами її опису. При цьому надається можливість задання на рівні опису обчислювального пристрою наступних параметрів: наявності чи відсутності певного функціонального вузла, кількості каналів надходження та видачі даних, кількості трактів обробки даних, схем з’єднання функціональних вузлів і т.д.

В статті коротко розглядаються традиційні методи конфігурування спеціалізованих програмних процесорів, які надаються мовами опису апаратних засобів [1], та альтернативні методи конфігурування на основі бібліотек програмних процесорів та їх компонент [2]. Пропонується генератор програмних процесорів на основі бібліотеки компонент, який побудовано з використанням програмної системи генерації файлів верхнього рівня мовою опису апаратних засобів [3]. Технологія проектування програмних процесорів, що покладена в основу генератора, має ряд переваг порівняно з традиційною технологією їх проектування мовами опису апаратних засобів в частині створення конфігуровних моделей.

1. Конфігурування спеціалізованих програмних процесорів

Коли розробляється надвелика інтегральна схема, то із спеціалізованого програмного процесора виділяється та його частина, яка забезпечує реалізацію заданого алгоритму з заданими параметрами, а все інше обладнання видаляється, що дозволяє мінімізувати затрати обладнання. Це здійснюється спеціальними засобами конфігурування.

Створення конфігуровних спеціалізованих програмних процесорів надає розробнику надвеликих інтегральних схем такі можливості:

1. вибрати таку модель, яка здатна забезпечити потрібні значення продуктивності і споживаної потужності, виходячи з наявних в нього технологій проектування;
2. оптимізувати структуру обчислювального пристрою з метою зменшення об'єму кристалу та споживаної потужності шляхом скорочення переліку виконуваних алгоритмів;
3. отримати модель з набором функціональних характеристик, найбільш близьким до потрібного;
4. отримати модель у найкоротші терміни часу;
5. під час розробки системи на кристалі замінити існуючу модель кращою за характеристиками; у разі використання кількох конфігуровних моделей – складових системи на кристалі – досягти потрібних характеристик системи шляхом підбору та комбінування моделей;
6. побудувати конфігуровну комп'ютерну систему на кристалі шляхом використання конфігуровних моделей, які у ній використовуються.

В свою чергу, розробник конфігуровних спеціалізованих програмних процесорів може:

1. знизити собівартість своїх продуктів, оскільки розробка конфігуровного спеціалізованого програмного процесора коштує менше, ніж розробка набору неконфігуровних, що можуть бути отримані з нього.
2. збільшити кількість замовників завдяки розширенню асортименту своїх продуктів.

2. Аналіз можливостей створення конфігуровних спеціалізованих програмних процесорів механізмами мов опису апаратних засобів

Мови опису апаратних засобів надають розробникові можливість створювати спеціалізовані програмні процесори якісно, зручно та у стислі терміни. Первинними завданнями мов опису апаратних засобів були моделювання та симуляція апаратури. Можливість синтезу апаратури з використанням мов опису апаратних засобів була винайдена розробниками пізніше як один зі шляхів автоматизації процесу розробки апаратних засобів. На сьогодні існує чотири стандарти мови VHDL [4], яка є однією з найпоширеніших – 1987, 1993, 2006 та 2008 років, та ряд модифікацій (наприклад, AHDL – Altera VHDL – оптимізована для синтезу ПЛІС фірми Altera). Щоб задовольнити вимоги розробників, прикладається багато зусиль для розширення мови, вдосконалення її механізмів, її універсалізації.

При розробці простих пристроїв абсолютно достатніми є можливості існуючих мов опису апаратних засобів. Однак при розробці складних пристроїв, а головним чином, конфігуровних програмних процесорів, ці можливості часто бувають недостатніми. В основному це стосується слабкої математичної орієнтації цих мов та недосконалим механізмом параметризації взагалі. Тому проектування конфігуровних програмних процесорів з використанням мов опису апаратних засобів є надзвичайно важкою, а в окремих випадках і взагалі непосильною задачею.

Мова VHDL є єдиною, яка надає можливості створення конфігуровних програмних процесорів. Конфігурування здійснюється з допомогою інтерфейсної константи *generic* та оператора *generate*. Однак, механізми конфігурування програмних процесорів на мові VHDL є громіздкими та незручними, що сильно ускладнює процес розробки.

Інтерфейсну константу *generic* записують у заголовку блока, під час оголошення компоненти чи під час опису її інтерфейсу та використовують для керування розміром портів, кількістю підблоків у блоці, розрядністю шин, кількістю виконання циклів тощо. Вона є інструментом передачі конфігураційної інформації певному блоку ззовні. Значення константи вказують, як правило, ззовні компоненти. Переважна більшість засобів синтезу програмних процесорів підтримують *generic* тільки типу *integer*.

Оператор *generate* – це механізм ітераційної обробки (або обробки при певній умові) фрагмента програми. Він спрощує опис однорідних структур пристрою і звичайно застосовується для побудови груп ідентичних компонент, використовуючи специфікацію однієї компоненти та повторюючи її певну кількість разів.

Оператор *generate* складається з трьох основних частин:

1. схема генерування (що базується на операторах *for* або *if*);
2. деклараційна частина (декларування підпрограм, типів, сигналів, констант, компонент тощо);
3. опис компоненти.

Схема генерування вказує, як саме повинна генеруватися структура, складена з повторюваних компонент. Використовуються два типи схеми генерування, один з яких базується на використанні оператора *for*, інший – оператора *if*. Схема першого типу використовується для описання однорідних структур пристрою і функціонує подібно до послідовного циклу. Схема другого типу використовується у ситуації, коли однорідна структура містить деякі неоднорідні елементи.

3. Бібліотеки програмних процесорів та їх компонент

Існує ряд альтернативних методів конфігурування програмних процесорів. В роботі [2] розглянуто бібліотеку, що містить програмні процесори, описані мовою опису апаратних засобів, та засоби керування бібліотекою. Засоби керування бібліотекою отримують від користувача конфігураційні коди, що задають параметри програмного процесора, та вибирають з множини програмних процесорів такий, який відповідає заданим параметрам. Файли опису цього процесора подаються на вихід. Суттєвим недоліком такої організації бібліотеки є надлишковість, яка зумовлена тим, що певні функціонально подібні програмні процесори часто містять однакові компоненти.

Для зменшення надлишковості у бібліотеку запропоновано вносити не програмні процесори, а їх компоненти [2] таким чином, щоб кожна компонента була присутня у бібліотеці лише в одному примірнику. При такій організації бібліотеки для реалізації конфігурування потрібні додаткові програмні засоби, які, крім вибирання потрібних компонент, повинні виконувати їх компоновку і генерувати файли опису верхнього рівня вибраною мовою опису апаратних засобів. Технологію генерації програмних процесорів на основі бібліотеки їх компонент та програми-генератора ілюструє рис.1.

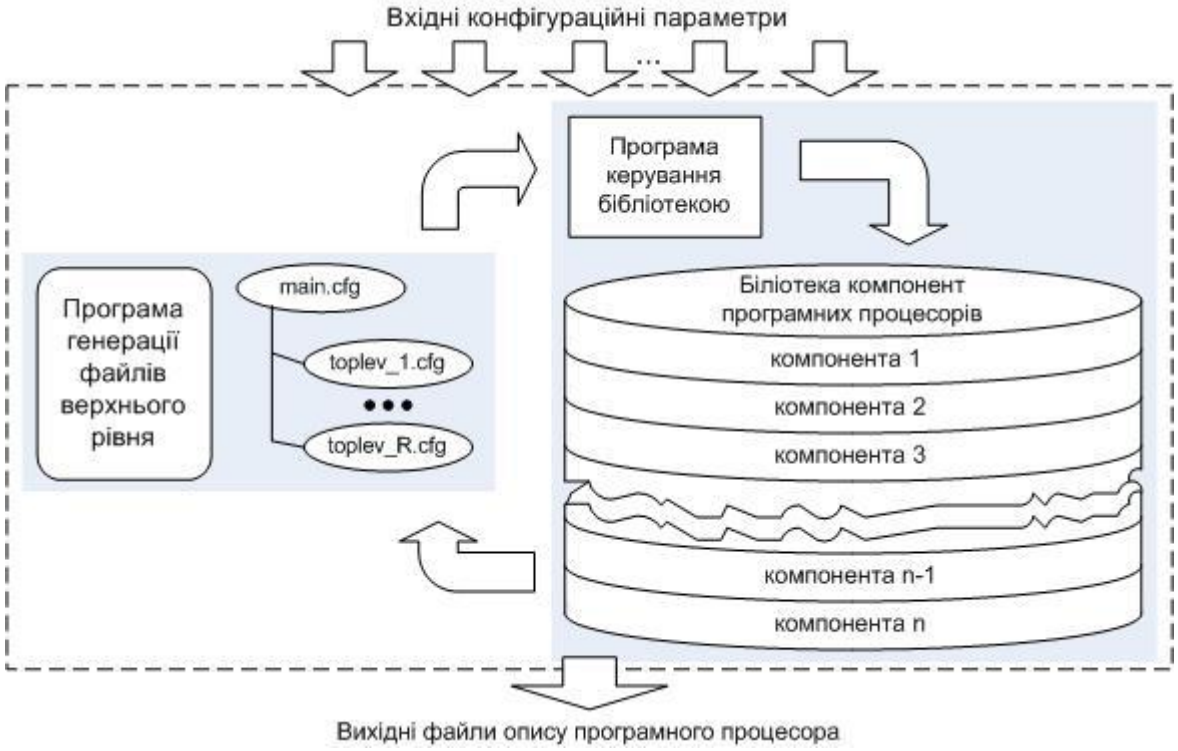

Рис. 1. Технологія генерації програмних процесорів на основі бібліотеки їх компонент та програми-генератора

Програма-генератор у взаємодії з бібліотекою компонент автоматично створює програмні процесори, виконуючи при цьому такі дії:

- отримання конфігураційних кодів, що задають параметри програмного процесора;
- вибір з множини компонент програмних процесорів, розміщених в бібліотеці, таких, які відповідають заданим параметрам;
- генерація файлів опису верхнього рівня, що інтегрують вибрані компоненти;
- подача файлів з описами компонент та файлів опису верхнього рівня на вихід.

3. Структура генератора програмних процесорів на основі бібліотеки компонент

Програмні процесори та їх компоненти можуть складатися з певної кількості рівнів ієрархії. У мовах опису апаратних засобів кожен з рівнів ієрархії формується файлом верхнього рівня (*top level file*). Файли конфігурації, якими оперує програма-генератор, містять інформацію про компоненти, що входять до певного рівня ієрархії програмного процесора. Кількість файлів конфігурації (наприклад, *toplev_x.cfg* ($x = 1, 2, \ldots, R$)) дорівнює кількості ієрархічних структур R. У головному файлі конфігурацій записують команди, що викликають ці файли у порядку зростання рівнів ієрархії. Таким чином програма-генератор визначає компоненти, що входять до конкретного програмного процесора. Імена компонент програма-генератор передає до програми керування бібліотекою, яка вибирає відповідні файли і передає їх на вихід бібліотеки. Разом з тим, програма-генератор копіює з цих файлів фрагменти з описом інтерфейсів компонент і використовує ці фрагменти при генерації файлів опису верхнього рівня.

На рис.2 показано структуру генератора програмних процесорів на основі бібліотеки компонент.

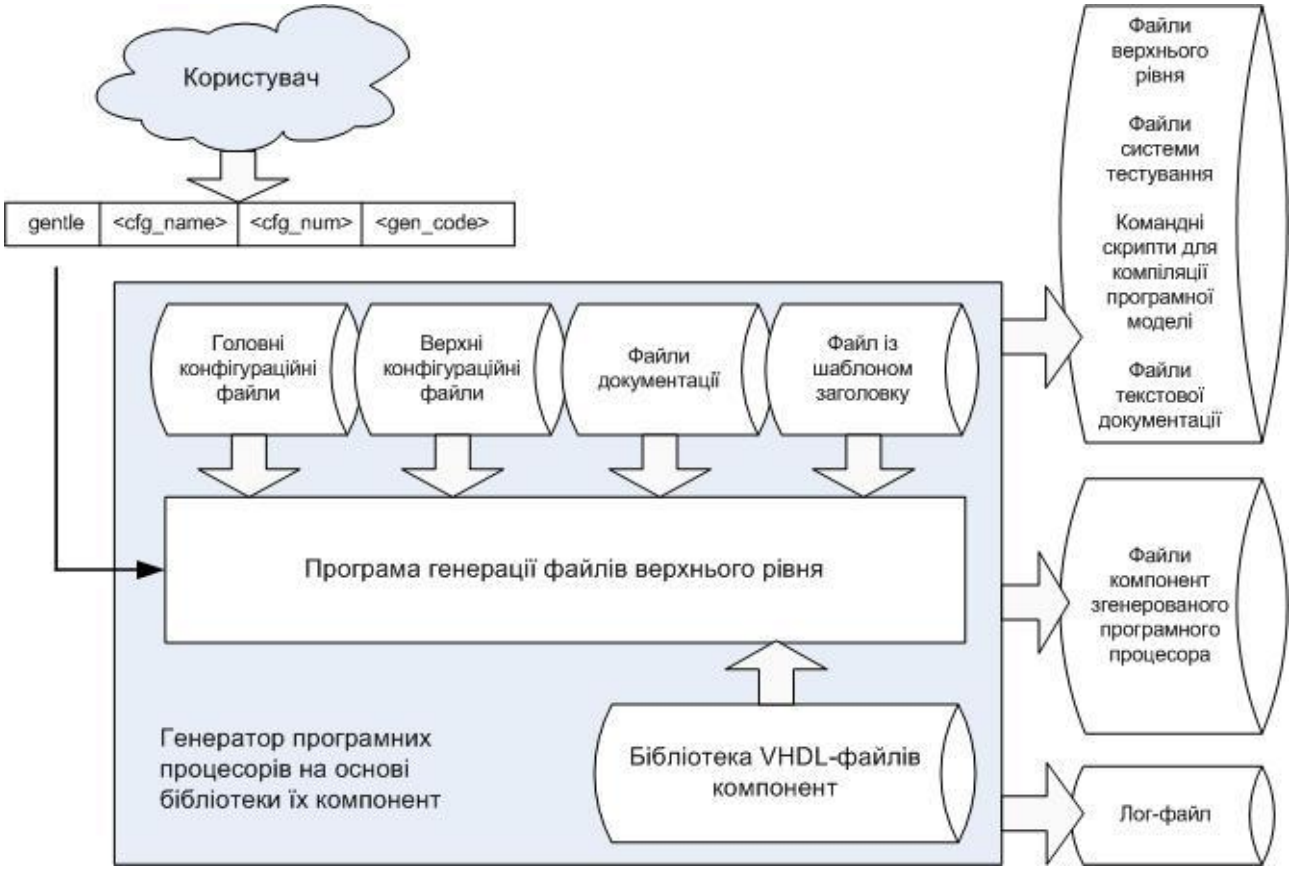

Рис.2. Структура генератора програмних процесорів на основі бібліотеки компонент

Генератор програмних процесорів побудовано з використанням наступних складових:

1. бібліотеки VHDL-файлів компонент програмних процесорів;
2. програми генерації файлів верхнього рівня;
3. набору головних конфігураційних файлів (файлів найвищого рівня ієрархії). Кількість таких файлів фактично визначає кількість програмних процесорів, що можуть бути згенеровані системою з використанням даної бібліотеки компонент;
4. набору верхніх конфігураційних файлів (файлів нижчих рівнів ієрархії). Кожен з цих файлів описує підмножину компонент програмного процесора на певному рівні ієрархії;
5. набору файлів документації (необов'язково), з якого будуть вибрані потрібні файли, що стосуються даної конфігурації, на етапі конфігурування;
6. шаблону заголовку, який буде використано при генеруванні кожного файлу проекту. В генерованому заголовку буде вказано автора проекту, організацію, назву проекту, час генерування та ін.

4. Програма генерації файлів верхнього рівня

Програма генерації файлів опису верхнього рівня виконує наступні функції:

1. сприйняття конфігураційних скриптів та параметрів, заданих користувачем, наприклад, через командну стрічку;
2. ієрархічна обробка компонент програмних процесорів;
3. генерація файлів верхнього рівня мовою опису апаратних засобів;
4. генерація скриптів для виконання компіляції програмних процесорів;
5. копіювання та модифікація файлів на рівні ASCII;
6. генерація систем тестування і тестових послідовностей.

Виклик програми генерації файлів верхнього рівня *gentle* (назва програми, скорочення від *generate top level*) виконується з командного рядка операційної системи UNIX / Windows наступним чином: *gentle* <*cfg_name*> <*cfg_num*> <*gen_code*>. Опис параметрів командного рядка програми генерації файлів верхнього рівня *gentle* наведено у Таблиці 1.

Таб.1. Опис параметрів командного рядка програми генерації файлів верхнього рівня *gentle*

cfg_name	Параметр, що вказує на головний конфігураційний файл <*cfg_name*>.cfg.
cfg_num	Параметр, що задає номер конфігураційної секції, розміщеної у головному конфігураційному файлі, яка визначає номер вибраної конфігурації.
gen_code	Параметр, що налаштовує програму на виконання визначеної операції, а саме: 1 = генерувати файли верхнього рівня; 2 = генерувати систему тестування; 3 = копіювати документацію (передбачено, що файли текстової документації, наприклад, рекламні листи чи інструкції користувача, повинні бути підготовлені наперед); 4 = генерувати список файлів з визначеним порядком компіляції; 4x = генерувати командні скрипти для компіляції файлів.

Програму генерації файлів верхнього рівня *gentle* реалізовано мовою PERL. Під час реалізації в більшості випадків використано глобальні змінні, тому підпрограми не розглядаються як незалежні блоки, а скоріш підтримують чіткішу структуризацію програми. На рис.3. показано алгоритм функціонування підпрограми обробки компонент програми *gentle*.

Підхід дворівневого конфігурування з допомогою конфігураційних скриптів дозволяє генерувати файли верхнього рівня від найвищого до найнижчого рівнів ієрархії. Реалізовано можливість використання згенерованих на попередньому етапі файлів верхнього рівня як компонент у наступному етапі роботи системи. Можна одночасно мати декілька головних конфігураційних файлів, кожен з яких міститиме декілька секцій. Аналогічно працюють конфігураційні файли нижчих рівнів.

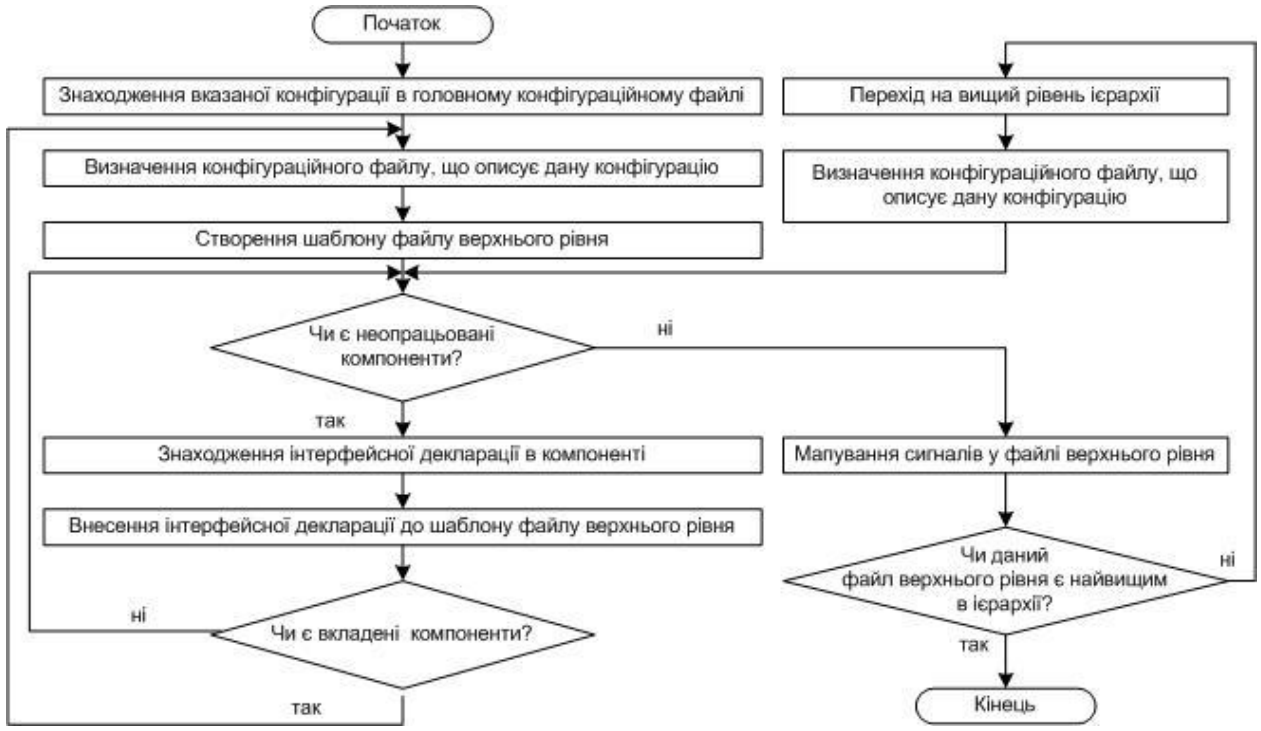

Рис.3. Алгоритм функціонування підпрограми обробки компонент програми *gentle*

Висновки

В статті запропоновано генератор програмних процесорів на основі бібліотеки компонент, який побудовано з використанням програмної системи генерації файлів верхнього рівня мовою опису апаратних засобів. Технологія проектування програмних процесорів, що покладена в основу генератора, має ряд переваг порівняно з традиційною технологією їх проектування мовами опису апаратних засобів в частині створення конфігуровних моделей, до яких, зокрема, можна віднести наступні:

1. Механізм конфігурування програмних процесорів на рівні компонент є значно гнучкішим в порівнянні з механізмами, що надаються мовами опису апаратних засобів, оскільки він не має синтаксичних та структурних обмежень.
2. Запропонована технологія генерації програмних процесорів забезпечує можливість конструювання програмних процесорів з компонент на оптимальному рівні їх деталізації.

Необхідно зазначити, що складність реалізації генератора залежить від рівня деталізації компонент. З одного боку, створення компонент найнижчого рівня деталізації зменшує надлишковість бібліотеки та оптимізує структуру програмних процесорів. З іншого боку, чим нижчий рівень деталізації, тим складнішими будуть конфігураційні файли. Підвищити рівень деталізації компонент без значних втрат в оптимальності структури програмних процесорів можна за умови спеціалізації бібліотеки та шляхом використання наборів спеціалізованих бібліотек.

1. В. Мельник, Мохаммад Аль Хабабсах. Методи конфігурування моделей спеціалізованих процесорів // IV Всеукраїнська науково-практична конференція «Комп'ютерні технології: наука і освіта», Україна, м.Луцьк, 9-11 жовтня 2009 р., Луцький інститут розвитку людини Університету «Україна», с.121-125. 2. В. Мельник. "Технології конфігурування моделей процесорів симетричного блокового шифрування" // Вісник Національного університету "Львівська політехніка" "Комп'ютерні системи проектування. Теорія і практика". – 2003. №471. – с.149-158. 3. Зіад Сарайрех. Програмні засоби синтезу в реконфігуровних прискорювачах процесорів на основі бібліотеки їх складових. Матеріали IV Міжнародної конференції молодих вчених CSE-2010 «Комп'ютерні науки та інженерія», 25-27 листопада 2010, Україна, Львів., с. 182-183. 4. IEEE, Standard VHDL Language Reference Manual. Standard 1076-1993, New York, NY: IEEE, 1993