УДК 004.382

В. С. Глухов, В. М. Хоміць
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

ПІДХІД ДО РЕАЛІЗАЦІЇ НА ПЛІС ЗАСОБАМИ ПАКЕТА VIVADO С-ОПИСІВ АЛГОРИТМУ СТИСНЕННЯ ЗОБРАЖЕНЬ

Розглянуто особливості побудови пристроїв для стиснення монохромних зображень без втрат методом JPEG-LS на сучасних ПЛІС. Апробовано можливості пакета Vivado (ф. Xilinx) з перетворення опису алгоритму JPEG-LS мовою С на VHDL-описи, придатні для імплементації в ПЛІС. Визначено конструкції мови С, які не можуть оброблятися вказаними засобами, та можливі способи обходу таких конструкцій.

Ключові слова: стиснення без втрат, ПЛІС, JPEG-LS, програмна реалізація, апаратна реалізація.

V. Hlukhov, V. Khomits
Lviv Polytechnic National University,
Computer Engineering Department

APPROACH TO IMPLEMENTATION ON FPGA OF DATA COMPRESSION ALGORITHM C LANGUAGE DESCRIPTIONS BY THE MEANS OF VIVADO PACKAGE

The features of devices for monochrome images lossless compression by JPEG-LS method in modern element base are discussed. Capabilities of Vivado package (Xilinx) for JPEG-LS algorithm C to suitable for implementation in FPGAs VHDL-descriptions transformation were tested and described. C language structures, which can not be processed by specified means and possible circumvention of such structures were defined.

Key words: lossless compression, FPGA, JPEG-LS, software implementation, hardware implementation.

Вступ

У роботі розглянуто можливості пакета Vivado (ф. Xilinx) з перетворення опису алгоритму JPEG-LS мовою С на VHDL-описи, придатні для імплементації в ПЛІС. Визначено конструкції мови С, які не можуть оброблятися вказаними засобами, та можливі способи обходу таких конструкцій. Апаратна реалізація стиснення зображень методом JPEG-LS є менш гнучкою, ніж програмна, та вимагає конкретики в розмірах даних, що будуть оброблятись. Також, на відміну від програми, вузол стиснення на ПЛІС обробляє відеопотік даних, а не окремі зображення, що зумовлює специфіку методу реалізації такого пристрою. Такі кодери використовують в охоронних системах, системах збирання інформації, супутникових, підводних та інших системах фотовідеоспостережень, а також в астрономічних інструментах і телескопах.

Аналіз літературних джерел

В основу методу стиснення JPEG-LS покладено алгоритм LOCO-I, описаний у роботі [1]. Детальніша інформація про особливості алгоритму та блок-схема його роботи міститься в інтернет-ресурсах [2]. В роботі [3] описано реалізацію методу стиснення JPEG-LS на ПЛІС фірми Xilinx, а в Інтернет-ресурсі [4] міститься його програмна реалізація. Можливу галузь застосування апаратної реалізації описано в [6]. Попередні результати дослідження наведено в [5]. У ресурсах [7] і [8] описано засоби для апаратної реалізації алгоритму JPEG-LS на ПЛІС. Питання стиснення зображень нестандартних форматів у згаданих роботах не розглядалося.

Мета роботи

Метою роботи є дослідження на прикладі програмної реалізації алгоритму JPEG-LS можливостей пакета Vivado (ф. Xilinx) з перетворення описів алгоритмів мовою С на VHDL-описи, придатні для імплементації в ПЛІС.

Розроблення апаратної реалізації методу jpeg-ls засобами Vivado

Відома програмна реалізація алгоритму JPEG-LS [4], яка дає змогу вносити зміни до алгоритму роботи, моделювати його роботу і визначати характеристики методу. Також відомий приклад [3] успішної апаратної реалізації на ПЛІС вузла стиснення методом JPEG-LS і засоби [7], які дають можливість перетворювати описи мовою С на VHDL-описи, придатні для розроблення топології ПЛІС. Метою роботи є дослідження можливості використання наявних та модифікованих за вимогами замовників С-описів алгоритму стискання інформації та засобів їхньої трансляції на VHDL-описи для створення топології ПЛІС, яка задовольняє вимоги замовників. Для цього використовують С-описи, які не створювалися з метою їх подальшого застосування для синтезу топології ПЛІС.

Основним завданням, яке треба вирішити для успішної апаратної реалізації програми jpeg-ls v2.2, є адаптація наявного коду С до його використання у середовищі Vivado HLS. Для цього створюється проект у вищезгаданому середовищі, у якому містяться всі файли з кодами програми jpeg-ls v2.2. Під час створення проекту слід вказати основну функцію проекту (в цьому випадку це main ()) та вибрати або створити файл test-bench для перевірки роботи програми.

На цьому етапі роботи виникають серйозні ускладнення, адже код програмної реалізації та код, який може обробити і синтезувати Vivado HLS, істотно відрізняються. Зокрема, під час синтезу необхідно усунути або замінити конструкції C/C++, які не синтезуються на Vivado HLS. Такі конструкції описано нижче.

Системні виклики

Системні виклики не можуть бути синтезовані, оскільки вони є діями, які пов’язані з виконанням деяких запитів від операційної системи, в якій виконується програма С.

Vivado HLS автоматично ігноруватиме часто використовувані системні виклики, які застосовуються тільки для відображення даних і ніяк не впливають на виконання алгоритму (наприклад, printf () і fprintf (STDOUT)), проте в системні виклики не можуть бути синтезовані й повинні бути видалені з коду (функцій) до синтезу. Іншими прикладами таких викликів є getc (), time (), sleep () і т.д., які здійснюють запити до операційної системи.

Динамічне використання пам’яті

Будь-які системні виклики, які керують розподілом пам’яті в системі, наприклад, malloc (), calloc (), і free () використовують ресурси пам’яті операційної системи і створюються та вивільняються під час виконання програми. Для апаратного синтезу такі конструкції повинні бути змінені так, щоб вони стали повністю автономними, з наперед точно визначеними всіма необхідними ресурсами.

Системні запити щодо розподілу пам’яті повинні бути видалені з конструкції коду до синтезу.

Однак, оскільки динамічний розподіл пам’яті використовують для реалізації функціональних вузлів проекту, його необхідно перетворювати на еквівалентні представлення, що підлягають синтезу.

Рекурсивні функції

Рекурсивні функції не можуть бути синтезовані. Це також стосується функцій, які можуть утворювати нескінченну рекурсію.

Також існують певні обмеження щодо використання вказівників. Визначення вказівників не підтримується в загальному випадку, але підтримується для внутрішніх типів у C.

Під час синтезу C-коду програми jpeg-ls v2.2 виявлено 23 конструкції, про які сказано вище та які необхідно замінити для успішного синтезу VHDL-опису.

На рис. 1 показано вікно консолі Vivado HLS зі списком помилок, що виникають під час оброблення початкового C-коду. Як можна побачити, помилки виникають під час роботи із зовнішніми файлами, системними викликами, такими як clock, putc, та функціями використання пам’яті, що виділяється динамічно.

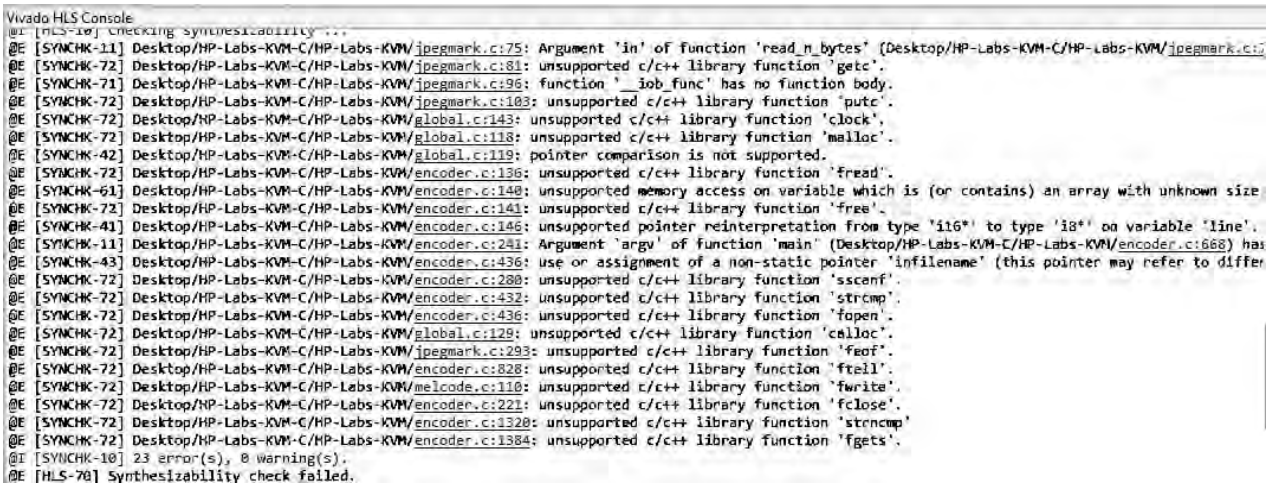

Рис. 1. Вікно програми Vivado HLS зі списком помилок під час синтезу коду

```
infilename = “c:\\hp\\mx18000x1000.pgm”;
/* Assing file pointers to in files */
if ((infilename == NULL) && (multi==0 || components<1) ) {
      usage();
      exit(0);
}
else {
       if (multi==0) {
            #ifndef __SYNTHESIS__
            if ( (in=fopen(infilemane, “rb”)) == NULL ) {
              perror(infilemane);
              exit(10);
            }
            #endif
        }
        else {
            for (i=0;i<components;i++){
                  #ifndef __SYNTHESIS__
                  if ( (in=fopen(infilemane, “rb”)) == NULL ) {
                        perror(infilemane);
                        exit(10);
                  }
                  #endif
            }
      }
```

Рис. 2. Частина коду з прописаним шляхом до робочого файла

Першим кроком усунення цих помилок є чітке визначене ім’я робочого файла та уникнення введення початкової інформації (параметрів) з командного рядка. За апаратної реалізації немає можливості вибирати ім’я файла, оскільки інформація надходитиме неперервно та стискатиметься лише за одним сценарієм. Отже, код треба змінити так, щоб стиснення відбувалось без запитів на введення параметрів з клавіатури. Тому в С-описі необхідно вказати ім’я файла та видалити непотрібну вже конструкцію введення імені з клавіатури (рис. 2).

Параметри алгоритму стиснення (ширина та висота зображення, кількість символів у таблиці, кількість компонентів) під час програмної реалізації зчитуються із вхідного файла зображення.

Ширину і довжину задають в пікселях – вони зчитуються із заголовка вхідного файла.

Значення ширини рядків у пікселях використовується для створення динамічних масивів для буферів обробки зображень. І ця величина застосовується у функціях malloc () і calloc () для задання допоміжних масивів типу void, які надалі використовують для обробки кожного рядка пікселів. Складність заміни функцій malloc () і calloc () полягає в тому, що з їх допомогою створюються масиви елементів для обробки ліній вхідного файла. Якщо ці конструкції з використанням динамічної пам’яті можна замінити на інші, просто визначивши точний розмір даних, то функції, які відповідають за роботу з файлом, такі як: fopen, fclose, fread та інші, потрібно просто ігнорувати за допомогою макроса __SYNTHESIS__ . Макрос використовується як зручний спосіб для виключення частини коду, яка не синтезується, не видаляючи сам код з функції С. Використання такого макроса, однак, робить код С для моделювання та код С для синтезу різними.

```
unsigned int read_n_bytes(FILE *in, unsigned int n)
{
      unsigned int m = 0;
      int i;

      for ( i=0; i<n; i++ ){
            #ifndef __SYNTHESIS__
            m = (m << 8) | ((unsigned char)getc(in));
            #endif
      }
      return m;
}
```

Рис. 3. Частина коду з виключенням функції getc за допомогою макроса “__SYNTHESIS__”

Виділивши проблемні конструкції коду макросом __SYNTHESIS__(рис. 3), можна добитись синтезу коду, однак також і втрати його функціональності (рис. 4). Програма jpeg-ls v2.2 побудована таким способом, що функції роботи із зовнішніми файлами є практично основними і щоб замінити їх, не втративши функціональності, потрібно майже повністю змінити логіку та структуру програми. Оскільки основним завданням цієї роботи є дослідження, а не готовий продукт, макросом __SYNTHESIS__ було виділено всі частини коду С, який не міг бути синтезований, та здійснено синтез VHDL-опису.

Після завершення синтезу Vivado HLS генерує три групи файлів різними мовами опису апаратного забезпечення – VHDL, Verilog та SystemC (рис. 3).

VHDL-код програми вийшов дуже коротким, оскільки Vivado HLS синтезувала функцію main, в якій за допомогою макроса не було синтезовано функції завантаження файла, запису та інших, практично основних у роботі функцій, які не можуть бути синтезовані. Звісно, такий код навряд чи можна назвати функціональним, але все ж його можна відкрити у ще одній програмі фірми Xilinx та переглянути RTL і технологічну схеми такого проекту.

За описаного підходу створюється функціонально обмежений VHDL-код, оскільки з початкового коду вилучено конструкції, які не можуть бути синтезовані пакетом Vivado HLS. Але

результат синтезу уможливлює подальше опрацювання (перегляд самого коду та технологічної схеми, яка йому відповідає) стандартними засобами проектування топології ПЛІС, тобто дає змогу відпрацювати саму технологію проектування.

Після успішного синтезу створеного vhdl-опису можна переглянути RTL та технологічну схему проекту.

Рис. 4. Вікно програми Vivado HLS з результатом синтезу функції main

Рекомендації

Працюючи з файлами наявних С-описів у середовищі Vivado, треба брати до уваги, що не всі конструкції мови С можна синтезувати. Працюючи з готовими програмами, можна зіткнутись із ситуацією, коли через конструкції, які не синтезуються, потрібно змінювати алгоритм роботи програми або її частин. Отримані результати дали змогу визначити ділянки С-описів, які неможливо обробити засобами пакета Vivado HLS. Знайдені ділянки було вилучено з описів за допомогою макросів *__SYNTHESIS__*, що дозволило повністю пройти процес створення топології кристала ПЛІС з початкового опису мовою С. Подальшим кроком є заміна вилучених ділянок початкового коду такими описами, які можуть бути оброблені засобами пакета Vivado HLS, а також створення технологічних засобів, які здатні автоматично шукати визначені проблемні ділянки в початкових С-описах з подальшою також автоматичною (з можливою участю людини) їх заміною на еквівалентні описи, які не порушують роботу алгоритму й обробляються засобами *пакета* Vivado HLS.

Перелік основних ускладень, які виникають під час синтезу, та рекомендовані способи їх вирішення наведено у таблиці. Необхідно пам’ятати, що вказані способи зменшують

функціональність опису, але дають змогу повністю пройти технологічний шлях створення топології ПЛІС.

Продовжуючи дослідження цієї теми, можливо, вдасться знайти більше вирішень проблем синтезу кодів С у Vivado HLS.

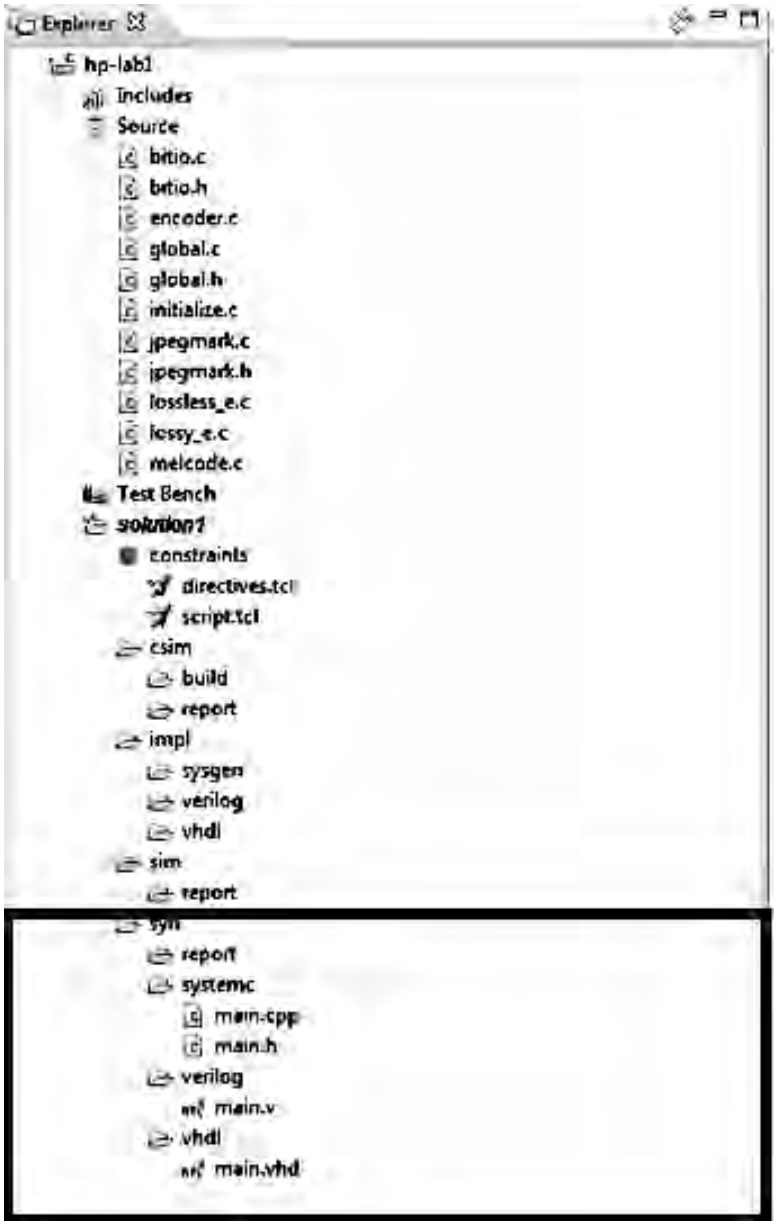

Рис. 5. Вікно програми з виділеним списком створених файлів

Напрям подальших робіт

Отримані результати дали змогу визначити ділянки С-описів, які неможливо обробити засобами пакета Vivado HLS. Знайдені ділянки було вилучено з описів за допомогою макросів __SYNTHESIS__, що дозволило повністю пройти процес створення топології кристала ПЛІС з початкового опису мовою С. Подальшим кроком є заміна вилучених ділянок початкового коду такими описами, які можуть бути оброблені засобами пакета Vivado HLS, а також створення технологічних засобів, які здатні автоматично шукати визначені проблемні ділянки в початкових С-описах з подальшою також автоматичною (з можливою участю людини) їх заміною на еквівалентні описи, які не порушують роботу алгоритму й обробляються засобами пакета Vivado HLS.

Способи вирішення проблем під час синтезу файлів С на Vivado

Місцезнаходження в коді (модуль, номер рядка)	Конструкція мови С	Призначення	Спосіб вирішення
jpegnark.c, 81	функція gets	Робота з потоком даних	__SYNTHESIS__
jpegnark.c, 103	функція puts	Робота з потоком даних	__SYNTHESIS__
global.c, 143	функція clock	Системний виклик	Вилучити з програми або __SYNTHESIS__
global.c, 118	функція malloc	Робота з динамічною пам'яттю	Необхідно визначити розміри пам'яті, яка виділяється
encoder.c, 136	функція fread	Робота з потоком даних	__SYNTHESIS__
encoder.c, 140	вказівник на динамічний масив	Вказівник	Немає. Треба виключити з програми динамічні конструкції або замінити на визначені масиви
encoder.c, 141	функція free	Робота з динамічною пам'яттю	Необхідно визначити розміри пам'яті, яка виділяється
encoder.c, 280	функція sscanf	Читання даних з потоку	__SYNTHESIS__
encoder.c, 436	функція fopen	Робота з зовнішнім файлом	__SYNTHESIS__
global.c, 129	функція calloc	Робота з динамічною пам'яттю	Треба визначити розміри пам'яті, яка виділяється
encoder.c, 828	функція ftell	Робота з потоком даних	__SYNTHESIS__
jpegnark.c, 293	функція feof	Перевірка файла, пов'язаного з потоком	__SYNTHESIS__
melcode.c, 110	функція fwrite	Робота з потоком даних	__SYNTHESIS__
encoder.c, 221	функція fclose	Робота із зовнішнім файлом	__SYNTHESIS__
encoder.c, 1320	функція strncmp	Лексикографічне порівняння двох рядків	__SYNTHESIS__
encoder.c, 1384	функція fgets	Робота з потоком даних та файлом	__SYNTHESIS__

Висновки

У роботі здійснено апробацію реалізації алгоритму засобами пакета Vivado *HLS* (ф. Xilinx) з перетворенням наявних описів алгоритму мовою С на VHDL-описи, придатні для імплементації в ПЛІС, визначено конструкції мови С, які не можуть оброблятися вказаними засобами, та наведено способи корекції початкових описів мовою С. Ускладнення під час опрацювання описів мовою С виникають у разі роботи з потоками даних та файлами, з пам'яттю, що динамічно розподіляється, із системними викликами та вказівниками. Внаслідок ізоляції проблемних ділянок створюється функціонально обмежений VHDL-опис, який дає змогу повністю пройти технологічний шлях створення топології ПЛІС.

1. Vajnberger M., Seroussi G., Schapiro G. The Loco-I:nyzka skladnist, zasnovana na konteksti, stysnennja zobrazhen bez vtrat Algorytm // Proc. Konferentsija IEEE Data Compression, Snowbird, schtat Juta, berezen-kviten 1996 roku 2. Stranichka posvjaschena opisaniju JPEG-LS kodera, realizovannogo avtorom na jazyke VHDL dlja primenenija v proektah na FPGA. [Elektronnyj resurs]. – Rezhym dostupu: http://jpegls.narod.ru. 3. JPEG-LS Encoder Core (Numerically and Near Lossless Compression) Design Specification. 2007 – 2013 ALMA Technologies. 4. Laboratoria Hewlett-Packard [Elektronnyj resurs]. – [Veb-sajt]. – Rezhym dostupu: http://www.labs.hp.com/research/info_theory/ loco/locodown.htm 5. Gluhov V. S., Homits V. M. Stysnennja zobrazhen bez vtrat metodom jpeg-ls na PLIS // Tretij naukovyj seminar "Kiberfizychni sustemy: dosjagnennja ta vyklyky". PROGRAMA. Natsionalnyj universytet "Lvivska politehnika", 13–14 chervnja 2017 r. Lviv, Ukraina. – 26–37 p. 6. Hlukhov Valerii, Lukenyuk Adolf, Shenderuk Sergii. Satellite scientific data collection and accumulation system as a basis for cyber-physical systems construction. Advances in Cyber-Physical Systems. Vol. 1. Number 2. Lviv Polytechnic National University. 2016. p. 77–86. 7. Vivado Design Suite User Guide: High-Level Synthesis (v2014.1) February 04, 2014. 8. Xilinx: All programmable [Elektronnyj resurs] : [Veb-sajt]. – Rezhym dostupu: https://www.xilinx.com/ (data zvernennja 18.05.2017). – Nazva z ekrana.