Обзор системы программирования ROCCC

Петров А.В., 05.12.2010

Введение

Разработка системы ROCCC (Riverside Optimizing Compiler for Configurable Circuits) если судить по дате первых публикаций началась в 2003 г. в университете Калифорнии (University of California at Riverside) и активно продолжается в настоящее время (последняя версия на момент написания статьи – 0.5.2 от 12 ноября 2010).

Система включает в себя набор инструментов по трансляции программ, написанных на усеченном языке Си, в язык VHDL и среду разработки с графическим интерфейсом, объединяющую эти инструменты и интегрированную в IDE Eclipse. В настоящее время доступны дистрибутивы системы под ОС Linux и Macintosh Snow Leopard.

Документация по данной системе не слишком обширна ([1, 2]), однако имеется ряд видеоуроков [3], примеры [4] и ряд публикаций [5].

Особенностью системы, по мнению её авторов, является поддержка структурного подхода к разработке снизу в верх и возможность разработки системы абстрагированной от конкретного оборудования за счет введения слоя описания интерфейса к аппаратуре(«*two novel features that simplify the design of hardware code accelerators: modular bottom-up designs and platform interface abstractions*»).

Как такового описания «языка» ROCCC не существует, поскольку программы для ROCCC пишутся на языке Си. Однако существует несколько ограничений на использование Си при разработке для ROCCC, а именно:

1. вместо операторов || и && необходимо использовать операторы & и |;
2. запрещено использование указателей;
3. запрещены вызовы Си-функций, неудовлетворяющих требованиям ROCCC;
4. запрещены операции сдвига на переменную величину;
5. поддерживаются только for-циклы;
6. запрещено использование переменных с именем «C»;
7. оператор «?» не поддерживается;
8. максимальная вложенность циклов равна двум;
9. только верхний цикл из двух вложенных может быть бесконечным (при этом отметим, что число итераций цикла может задаваться как константой, так и переменной).

В остальном использование языка Си ничем не отличается от его использования при написании программ для ПК.

ROCCC поддерживает ряд оптимизаций кода, а именно:

1. типы данных переменной разрядности;
2. генерация систолических массивов;
3. устранение избыточных вычислений в циклах;
4. автоматическое распараллеливание вычислительных операций;
5. устранение избыточного копирования данных;
6. «Умные буферы» (см. [1] стр. 48).

Дополнительно отметим, что ROCCC включает инструменты для описания связи генерируемого VHDL кода с конкретной аппаратной платформой (ПЛИС и её «обвязка»), на которой планируется исполнение разработанной программы.

Более подробно структура ROCCC с точки зрения разработчика подобных систем описана в [2].

Примеры на языке ROCCC

Пример 1 – подмодуль для вычисления БПФ

```
// Interface
typedef struct
{
   int realOne_in ;
   int imagOne_in ;
   int realTwo_in ;
   int imagTwo_in ;
   int realOmega_in ;
   int imagOmega_in ;

   int A0_out ;
   int A1_out ;
   int A2_out ;
   int A3_out ;
} FFT_t ;

// Implementation
FFT_t FFT(FFT_t f)
{
   int tmp1 ;
   int tmp2 ;

   tmp1 = f.realOmega_in * f.realTwo_in ;
   tmp2 = f.imagOmega_in * f.imagTwo_in ;

   f.A0_out = f.realOne_in + tmp1 - tmp2 ;
   // The other outputs computations go here...
   return f ;
}
```

Recognized as inputs to the module

Recognized as outputs to the module

Internal registers

Пример 2 – Умножение матриц

```
/*
In order for this example to compile, you must select loop unrolling on the
innermost loop. You may also choose to unroll the other loops as well.
*/

void MatrixSystem()
{
    int A[10][10] ;
    int B[10][10] ;
    int Result[10][10] ;
```

```
    int i ;
    int j ;
    int k ;
    int currentSum ;

    for(i = 0 ; i < 10 ; ++i)
    {
        for(j = 0 ; j < 10 ; ++j)
      {
            currentSum = 0 ;

            L1: for(k = 0 ; k < 10 ; ++k)
            {
                currentSum += A[i][k] * B[k][j] ;
            }

            Result[i][j] = currentSum ;
        }
    }
}
/*
The above code performs matrix multiplication on the two dimensional streams
A and B. Two dimensional streams are declared as two dimensional arrays in
the C code. Since ROCCC only supports up to two control loops, the inner most
loop must be fully unrolled. This can be done by labeling the loop as shown
above and selecting the LoopRooling optimization in the GUI during
compilation.

When unrolled, each two dimensional array is changed into a one dimensional
array. Each of the one dimensional arrays are the individual rows and
individual columns of the original arrays. The result of this multiplication
is output through the Result stream which remains two dimensional since it
was not involved in the loop unrolling like A and B.
*/
```

Пример 3 – Верхний модуль вычисления БПФ (использует модуль из примера 1)

```
#include "roccc-library.h"

typedef struct
{
    int real0_in ;
    int imag0_in ;
    int real1_in ;
    int imag1_in ;
    int real2_in ;
    int imag2_in ;
    int real3_in ;
    int imag3_in ;
    int real4_in ;
    int imag4_in ;
    int real5_in ;
    int imag5_in ;
    int real6_in ;
    int imag6_in ;
    int real7_in ;
    int imag7_in ;

    int realOmega_in ;
    int imagOmega_in ;

    int x0Real_out ;
    int x0Imag_out ;
```

```
    int x1Real_out ;
    int x1Imag_out ;
    int x2Real_out ;
    int x2Imag_out ;
    int x3Real_out ;
    int x3Imag_out ;
    int x4Real_out ;
    int x4Imag_out ;
    int x5Real_out ;
    int x5Imag_out ;
    int x6Real_out ;
    int x6Imag_out ;
    int x7Real_out ;
    int x7Imag_out ;

} FFTOneStage_t ;

FFTOneStage_t FFTOneStage(FFTOneStage_t t)
{
    // The order in which each FFT expects the parameters is:
    // (Real0, RealOmega, Real1, ImagOmega, Imag1, x0Real, Imag0,
    //  x0Imag, x1Real, x1Imag)

    // In the definition of FFTOneStage each block is numbered from
    // the top down in increasing order.  The criss-crossing is done at
    // the higher level when we instantiate these blocks.

    // There are four butterfly operations per stage.
    FFT(t.real0_in, t.realOmega_in, t.real1_in,
        t.imagOmega_in, t.imag1_in, t.x0Real_out,
        t.imag0_in, t.x0Imag_out, t.x1Real_out, t.x1Imag_out) ;

    FFT(t.real2_in, t.realOmega_in, t.real3_in,
        t.imagOmega_in, t.imag3_in, t.x2Real_out,
        t.imag2_in, t.x2Imag_out, t.x3Real_out, t.x3Imag_out) ;

    FFT(t.real4_in, t.realOmega_in, t.real5_in,
        t.imagOmega_in, t.imag5_in, t.x4Real_out,
        t.imag4_in, t.x4Imag_out, t.x5Real_out, t.x5Imag_out) ;

    FFT(t.real6_in, t.realOmega_in, t.real7_in,
        t.imagOmega_in, t.imag7_in, t.x6Real_out,
        t.imag6_in, t.x6Imag_out, t.x7Real_out, t.x7Imag_out) ;

    return t ;
}
/*
The C code for the FFTOneStage is shown above. To be able to use a module,
the module must first exist in the database, either through importing or
compilation. To instantate a module, simply double click the desired module
call in the IPCores view in the ROCCC GUI and it will automatically place the
a skeleton module call in the C code. ROCCC will instantiate these module
calls in the VHDL and hook up all the signals correctly.
*/
```

Литература

1. User Manual - http://roccc.cs.ucr.edu/documentation/files/UserManual-0.5.2.pdf
2. Developers Manual - http://roccc.cs.ucr.edu/documentation/files/DevelopersManual-0.5.1.pdf
3. Видеоуроки по ROCCC - http://roccc.cs.ucr.edu/video_tutorials.php
4. Примеры программ на ROCCC - http://roccc.cs.ucr.edu/examples/index.php
5. Полный список публикаций - http://roccc.cs.ucr.edu/publications.php