Труды XIV Международного симпозиума
«Методы дискретных особенностей в задачах математической физики»
(МДОЗМФ-2009)
Харьков-Херсон, 8-12 июня 2009, 4 с.

УДК 519.68 + 681.3

Подход к моделированию и реализации параллельных вычислений на языке Ada

С. И. Киркоров, Л. С. Киркорова
Научно-производственное предприятие МедиаСкан, г. Минск

Article is devoted the approach to modeling of parallel algorithms of coding of a video information, protocol CCSDS 133.0-B-1 and their realizations in the testing stand in language Ada. Use of models in language Ada in system of projecting Xilinx. Possibilities of language Ada for realization of hardware-software systems and feature of realizations for MS Windows.

Одной из причин моделирования алгоритмов, реализующих параллельные вычисления, является необходимость их дальнейшего применения в аппаратно-программных или аппаратных системах. В последнее время предлагается относительно много схем аппаратной интерпретации выражений. Решение проблемы тормозится, с одной стороны, технологическими трудностями в реализации, с другой стороны, языковыми проблемами и, наконец, сложной логикой самих решений и необычностью постановки [1]. Особенность аппаратных средств – это возможность параллельного функционирования двух и более модулей системы, причем любой модуль может сам состоять из параллельно работающих частей. С другой стороны существуют прикладные задачи, например, в области технической защиты информации, где требуется как формальное доказательство правильности алгоритма, так и его фактической реализации. В таких приложениях, как дистанционное зондирование Земли (ДЗЗ), где применяются средства сжатия информации параллельно с алгоритмами шифрации и кодирования данных, моделирование алгоритмов, реализующих параллельные вычисления с применением пакетов типа MatLab, применимо только частично, на этапе математического моделирования. Одной из причин ограничений применения такого инструментария это сложность или высокая стоимость обеспечения соответствующего уровня гарантий конечного продукта в области технической защиты информации по всем трем составляющим – конфиденциальность, целостность и доступность [2], [3], [4]. Это связано с различиями оптимизированного под прикладную область алгоритмом и его математической моделью реализованной такими пакетами.

Сложность и стоимость возрастет, когда стоит задача переноса модели алгоритма и ее реализации на другую доверенную платформу. Под доверенной платформой здесь подразумевается собственный вычислитель, например специализированный процессор с другой операционной системой или без нее или программируемые пользователями вентильные матрицы (Field-Programmable Gate Array (FPGA)). Для тестирования доверенной платформы и параллельных вычислений реализуемые ее, как правило, создается стендовое оборудование. Стендовое оборудование как средство измерения также требует соответствующего уровня гарантий правильности своего функционирования.

Правильным выбором в этом случае было бы использовать инструментальное средство, которое:

1. Само могло бы пройти сертификационные испытания в области технической защиты информации, то есть, как минимум имела открытые спецификации, коды программ и так далее.
2. Соответствовало стандарту, строго его выполняло и имело стандарт, регламентирующий эти проверки. Было реализовано (или могло быть адаптировано) для целевых платформ.
3. Обеспечивало поддержку многозадачности (для моделирования алгоритмов, реализующих параллельные вычисления) на уровне языка высокого уровня. Этим достигается стабильность семантики, разработчик избавляется от необходимости использования разнородных внешних библиотек или собственных решения для обеспечения многозадачности.
4. Влияет, конечно, и экономический фактор. Стоимость приобретения и совокупного владения такого инструментария не должна превышать 1/3 от всей стоимости разработки.

Встроенная поддержка многозадачности является уникальной и широко известной особенностью языка программирования Ada, которая выгодно отличает его от большинства современных языков программирования. Следует особо подчеркнуть, что поддержка многозадачности обеспечивается не с помощью каких-либо расширений или внешних библиотек, а с помощью строго стандартизированных средств, которые встроены непосредственно в язык программирования [5], [6].

Критерий выбора инструментария по п. 1 и п. 4 в общем случае зависят от региона и от условий, в которых работает исследователь/разработчик моделей и реализаций параллельных вычислений. Авторы работы использовали инструментарий от фирмы AdaCore (http://www.adacore.com) GNAT GPL 2008 (20080521) для операционной системы Windows XP фирмы Microsoft.

Рассмотрим некоторую обобщенную постановку [1]. Пусть имеется вычислительный фрагмент (например, выражение), выполняющий содержательную функцию в программе. Реализация этого фрагмента требует выполнения набора каких-то элементарных базовых операций и передач данных от одних операций к другим. Выполнение операции при этом зависит только от готовности данных для этой операции. Для анализа рассмотрим пример вычисления произведения нескольких матриц $Y = \prod_{i=1}^{n} X_i$.

Выполнение такого преобразования программно обычно приводит к последовательной программе с циклами и, возможно, условными операторами. Другой путь – в том, что интерпретатор представляет собой параллельную программу (реализованную, возможно, аппаратно), которая «прогоняет» преобразование через имеющиеся у нее ресурсы – сумматоры, умножители, специализированные подсистемы, элементы локальной памяти.

Авторы работы провели исследования по улучшению локальности многомерных алгоритмов с помощью тайлинга. Задачей также было использования одного и того же инструментального средства, языка

программирования Ada, для разработки различных моделей параллельных вычислений и необходимого стендового оборудования. Целью тайлинга является уменьшение накладных расходов на использование иерархической памяти процессора. Тайлинг широко используется в алгоритмах сжатия и восстановления изображения (например, в стандартах JPG и JPG2000), алгоритмах шифрации и кодирования данных.

Под тайлом понимается множество операций алгоритма, выполняемых как одна единица вычислений атомарно. Геометрически тайл можно представить множеством точек многомерной целочисленной решетки. Наиболее частой для использования является форма прямоугольного параллелепипеда.

В модели алгоритма перемножения квадратных матриц для различных модификаций алгоритма: с помощью перестановки циклов и с помощью тайлинга. Исследование было выполнено как продолжение курсовой работы, защищенной на кафедре Математического моделирования и управления Факультета прикладной математики и информатики (ФПМИ) Белорусского государственного университета (БГУ) [7]. Исходными данными к работе послужили способы улучшения локальности многомерных циклов (перестановка циклов, тайлинг) [8], [9].

В результате исследования подтвердилось влияние перестановки циклов и тайлинга на скорость работы алгоритма. Оптимальные решения были разные в зависимости от аппаратной платформы (в данном исследовании сравнение проводилось на одноядерном и двухъядерном процессорах фирмы Intel). Проведенные экспериментальные исследования позволили определить максимальные значения количества тайлингов выполняемых в виде задачи Ada-95 и их оптимум для конкретных сочетаний программных и аппаратных средств.

Ниже приведен текст тестового модуля для оценки производительности аппаратно-программной платформы:

```
with Ada.Text_IO; -- use Text_IO;
with Ada.Calendar; -- use  Ada.Calendar;
with Ada.Exceptions;
with GNAT.Traceback.Symbolic;
--with Ada.Unchecked_Conversion;

with Test; -- use Test;

procedure Taling is

   time_taling : Test.timeDuration := 0.0;
   CalendD1: Ada.Calendar.Time;
   CalendD2 : Ada.Calendar.Time;

   a,b,c : Test.taling_array_Access;

   i,j : integer := 0;

   subtype tiling_count is Integer range 0..12;
--   subtype tiling_count is Integer range 0..17;

   type tiling_type is array (tiling_count) of integer;
     tiling : tiling_type :=
       (2,4,5,8,10,20,25,40,100,125,200,250,500);
   ct : tiling_count :=0;
```

```
   fl : Ada.Text_IO.File_Type ;

begin
   Ada.Text_IO.Create(fl,Ada.Text_IO.Out_File,"result.txt");
   a := new Test.taling_array;
   b := new Test.taling_array;
   c := new Test.taling_array;

   for i in 0..Test.count-1 loop
      for j in 0..Test.count-1 loop
         a(i)(j):= 1.0;
         b(i)(j):= 1.0;
         c(i)(j):= 0.0;
      end loop;
   end loop;

   Ada.Text_IO.Put_Line("matr_multipl_IJKBL");
   Ada.Text_IO.Put_Line(fl,"matr_multipl_IJKBL");
   for ct in tiling_count loop
      Ada.Text_IO.New_line;
      CalendD1 := Ada.Calendar.Clock; -- начальное время
      Test.matr_multipl_IJKBL(a,b,c,Test.count,tiling(ct));
      CalendD2 := Ada.Calendar.Clock; -- конечное время
      time_taling := Test.Time_sec(CalendD1,CalendD2);

      Ada.Text_IO.Put_Line("count =" & tiling_count'Image(tiling(ct)) & ".  ");
      Ada.Text_IO.Put_Line("Time kij =" & Test.timeDuration'Image(time_taling));
      Ada.Text_IO.Put_Line(fl,"count =" & tiling_count'Image(tiling(ct)) & ".  ");
      Ada.Text_IO.Put_Line(fl,"Time kij =" & Test.timeDuration'Image(time_taling));
      Ada.Text_IO.New_line(fl);
   end loop;

   for i in 0..Test.count-1 loop
      for j in 0..Test.count-1 loop
         a(i)(j):= 1.0;
         b(i)(j):= 1.0;
         c(i)(j):= 0.0;
      end loop;
   end loop;

   Ada.Text_IO.Put_Line("matr_multipl_TASK");
   Ada.Text_IO.Put_Line(fl,"matr_multipl_TASK");
   for ct in 5..12 loop
      CalendD1 := Ada.Calendar.Clock; -- начальное время
      Test.matr_multipl_TASK(a,b,c,Test.count,tiling(ct));
      CalendD2 := Ada.Calendar.Clock; -- конечное время
      time_taling := Test.Time_sec(CalendD1,CalendD2);

      Ada.Text_IO.Put_Line("count =" & tiling_count'Image(tiling(ct)) & ".  ");
      Ada.Text_IO.Put_Line("Time kij =" & Test.timeDuration'Image(time_taling));
      Ada.Text_IO.Put_Line(fl,"count =" & tiling_count'Image(tiling(ct)) & ".  ");
      Ada.Text_IO.Put_Line(fl,"Time kij =" & Test.timeDuration'Image(time_taling));
   end loop;

   pragma Warnings (Off);
   return;


exception
   when Error : others =>
      Ada.Text_IO.Put_Line (Ada.Exceptions.Exception_Information (Error));
      Ada.Text_IO.Put_Line (GNAT.Traceback.Symbolic.Symbolic_Traceback (Error));
end Taling;
```

```
with Ada.Calendar;

package Test is

   type timeDuration is delta 0.000001 range 0.0..9000.0;

   function Time_sec(CalD1: Ada.Calendar.Time;
                     CalD2 : Ada.Calendar.Time) return timeDuration;

   count : Integer := 1000;
--   count : Integer := 5000;
   type taling_array_vect is array (0..count-1) of Long_Float;
   type taling_array is array (0..count-1) of taling_array_vect;
   type taling_array_Access is access taling_array;


   procedure matr_multipl_IJK
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer);


   -- procedure matr_multipl_IJKBL
   procedure matr_multipl_IJKBL
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

   -- procedure matr_multipl_IKJBL
   procedure matr_multipl_IKJBL
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

   -- procedure matr_multipl_JKIBL
   procedure matr_multipl_JKIBL
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

   -- procedure matr_multipl_JIKBL
   procedure matr_multipl_JIKBL
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

   -- procedure matr_multipl_KIJBL
   procedure matr_multipl_KIJBL
     (a : access taling_array;
      b : access taling_array;
```

```
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

   -- procedure matr_multipl_KJIBL
   procedure matr_multipl_KJIBL
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

   procedure matr_multipl_TASK
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
      count : Integer;
      tile  : Integer
     );

end Test;


with Ada.Text_IO;
with Ada.Unchecked_Conversion;
with Ada.Exceptions;
with GNAT.Traceback.Symbolic;

package body Test is

--     time_taling : timeDuration := 0.0;
--     CalendD1: Ada.Calendar.Time;
--     CalendD2 : Ada.Calendar.Time;

   function Time_sec(CalD1 : Ada.Calendar.Time;
                     CalD2 : Ada.Calendar.Time) return timeDuration is
      t1, t2, t3 : Long_Long_Integer := 0;
      pragma Warnings (Off);
      function Time_to_Long_Long_Int is
        new Ada.Unchecked_Conversion (Ada.Calendar.Time, Long_Long_Integer);
      pragma Warnings (On);
   begin
      t1 := Time_to_Long_Long_Int (CalD1);
      t2 := Time_to_Long_Long_Int (CalD2);
      t3 := t2 - t1;

      declare
         f  : float := float(t3);
      begin
         f := f  / 1_000_000_000.0;
         return timeDuration (f);
      exception
         when Error : others =>
            Ada.Text_IO.Put_Line (Ada.Exceptions.Exception_Information (Error));
            Ada.Text_IO.Put_Line (GNAT.Traceback.Symbolic.Symbolic_Traceback (Error));
            return 0.0;
      end;
    end Time_sec;
   procedure matr_multipl_IJKBL
     (a : access taling_array;
      b : access taling_array;
      c : access taling_array;
```

```
  count : Integer;
  tile  : Integer
 ) is

  tilesize : Integer := count/tile;
  i1begin, i1end : integer := 0;
  j1begin, j1end, k1begin, k1end : integer := 0;
begin
  for i1 in 0 .. tilesize-1 loop

    i1begin := i1*tile;
    i1end := (i1+1)*tile - 1;

    for j1 in 0 .. tilesize-1 loop

      j1begin := j1*tile;
      j1end := (j1+1)*tile - 1;

      for itc in i1begin .. i1end-1 loop
        for jtc in j1begin..j1end-1 loop
          c(itc)(jtc) := 0.0;
        end loop;
      end loop;

      for k1 in 0..tilesize-1 loop
        k1begin := k1*tile;
        k1end := (k1+1)*tile - 1;
        for it in i1begin..i1end loop
          for jt in j1begin..j1end loop
            for kt in k1begin..k1end loop
              c(it)(jt) := c(it)(jt) + a(it)(kt)*b(kt)(jt);
            end loop;
          end loop;
        end loop;
      end loop;

    end loop;

  end loop;
end matr_multipl_IJKBL;

procedure matr_multipl_TASK
 (a : access taling_array;
  b : access taling_array;
  c : access taling_array;
  count : Integer;
  tile  : Integer
 ) is

  tilesize : Integer := count/tile;
  i1begin, i1end : integer := 0;
  j1begin, j1end, k1begin, k1end : integer := 0;

  task type multipl_tiling (ind1, ind2, ind3 : integer) ;

  task body multipl_tiling is
    i1begin : integer := ind1*tile;
    i1end   : integer := (ind1+1)*tile - 1;
    j1begin : integer := ind2*tile;
    j1end   : integer := (ind2+1)*tile - 1;
    k1begin : integer := ind3*tile;
    k1end   : integer := (ind3+1)*tile - 1;
  begin
```

```
--        Ada.Text_IO.Put_Line("Btsk ="&integer'Image(ind1)&integer'Image(ind2)&integer'Image(ind3));
        for itc in i1begin .. i1end-1 loop
          for jtc in j1begin..j1end-1 loop
            c(itc)(jtc) := 0.0;
          end loop;
        end loop;

        for it in i1begin..i1end loop
          for jt in j1begin..j1end loop
            for kt in k1begin..k1end loop
              c(it)(jt) := c(it)(jt) + a(it)(kt)*b(kt)(jt);
            end loop;
          end loop;
        end loop;
--        Ada.Text_IO.Put_Line("Etsk ="&integer'Image(ind1)&integer'Image(ind2)&integer'Image(ind3));

      end multipl_tiling;

      type multipl_tiling_Ptr is access multipl_tiling;
      type type_ar_tsk is array
        (1..tilesize,1..tilesize,1..tilesize) of multipl_tiling_Ptr;
      ar_tsk : type_ar_tsk;

    begin
      for i1 in 0 .. tilesize-1 loop

        i1begin := i1*tile;
        i1end := (i1+1)*tile - 1;

        for j1 in 0 .. tilesize-1 loop

          j1begin := j1*tile;
          j1end := (j1+1)*tile - 1;

          for k1 in 0..tilesize-1 loop
            k1begin := k1*tile;
            k1end := (k1+1)*tile - 1;

            ar_tsk(i1+1,j1+1,k1+1) := new multipl_tiling(i1,j1,k1);

          end loop;

        end loop;

      end loop;
--      Ada.Text_IO.Put_Line("++++++++++++END++++++++++++");
    end matr_multipl_TASK;


  end Test;
```

Текст программы приведен с некоторыми сокращениями.

Использование тайлинга для улучшения локализации алгоритма значительно повышает скорость расчетов. При разной последовательности циклов в алгоритме оптимальный размер тайла не одинаков для всех случаев. В каждом случае он разный. Поэтому для каждой модификации алгоритма при использовании тайлинга необходимо экспериментально подбирать его размер. И тогда время вычислений будет улучшено.

Аналогичный подход с использование языка Ada как для моделирования алгоритмов, так и для создания стендового оборудования был применен у Заказчика при разработке программных средств сжатия информации

космических систем ДЗЗ и стендового оборудования. Конечным результатом данной работы были средства для исследования аппаратных и программных реализаций алгоритмов сжатия и восстановления (стендовое оборудование), реализованные полностью на языке Ada-95 (за исключением покупных программных, аппаратных средствам и разработанной аппаратной платформы). Программные модели алгоритмов сжатия видеоинформации на языке Ada-95, полностью соответствующие аппаратной реализации на FPGA фирмы Xilinx и имеющие формат данных, аналогичный протоколу CCSDS-133.0-B-1. Программные модели алгоритмов восстановления сжатой видеоинформации реализованы также на языке Ada-95. Программные реализации моделей алгоритмов могут использоваться, вне стендового оборудования, в прикладных системах, реализованных на языке Ada-95, так как являются платформо-независимыми и имеют высокие показатели производительности и использования вычислительных ресурсов целевой аппаратуры.

На рисунке 1 приведен процесс разработки алгоритмов сжатия видеоинформации для целевой аппаратуры дистанционного зондирования Земли.

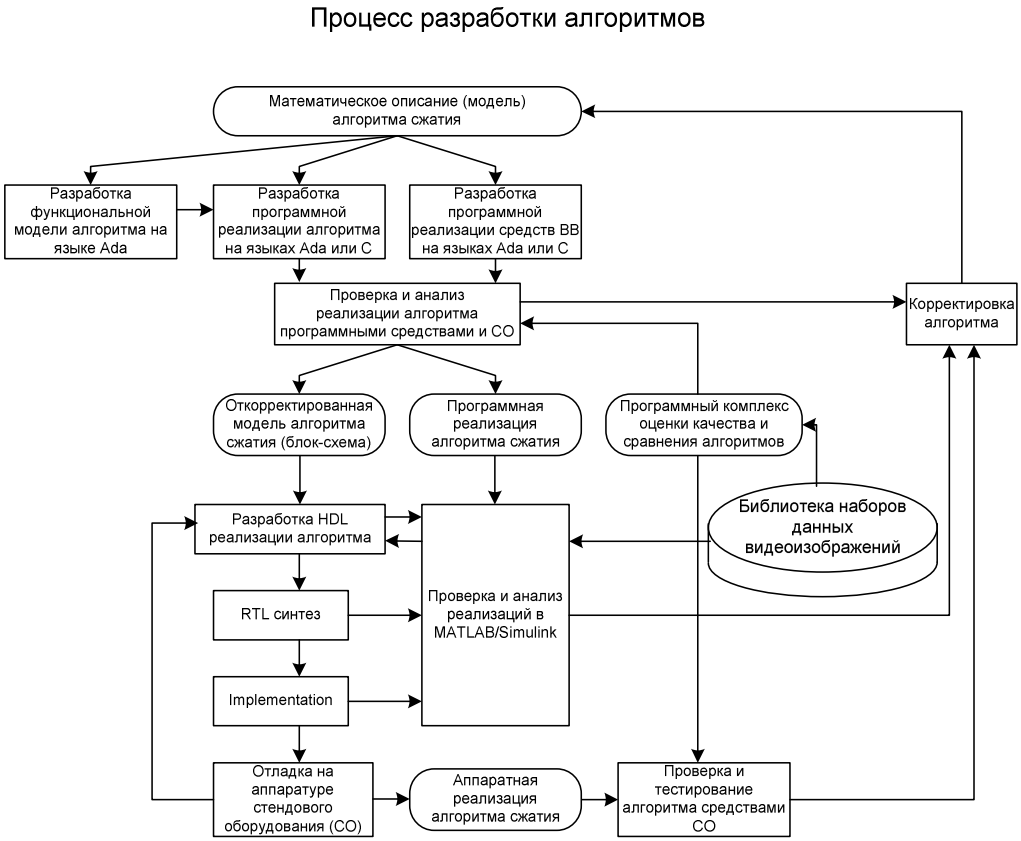

Рисунок 1. Процесс разработки алгоритмов сжатия видеоинформации для целевой аппаратуры.

Уникальностью данного комплекса для Республики Беларусь – это реализация вновь разрабатываемого кода (более 90%) исключительно на языке Ada-95.

Программно-аппаратный комплекс, позволяющий на базе разработанных программных и аппаратных средств производить моделирование, проверку и испытания алгоритмов сжатия и восстановления видеоинформации.

В состав программных средств входит:
программные средства сжатия видеоинформации (ПС СВ);
программные средства восстановления видеоинформации (ПС ВВ);
программные средства стендового оборудования (ПС СО);
библиотека наборов данных видеоинформации (БНДВ).

ПС СВ предназначены для сжатия полутоновых изображений с градациями серого цвета и позволяют осуществлять:

1. выбор алгоритма сжатия видеоинформации;
2. задание режимов обработки видеоинформации: сжатие без потерь, сжатие с потерями;
3. задание коэффициента сжатия (в режиме сжатия с потерями) – от 2 до 10;
4. реализацию используемых алгоритмов в бортовой аппаратуре сжатия (макетном образце бортовой аппаратуры сжатия видеоинформации (МОСВ)).

ПС ВВ предназначены для восстановления видеоинформации, сжатой ПС СВ и МОСВ.

ПС СО предназначены для разработки, моделирования, отладки и испытания ПС СВ, ПС ВВ и обеспечивают процессы:

1. работу и взаимодействие аппаратных и программных средств стендового оборудования;
2. формирование и выдачу информационных потоков, имитирующих работу съемочной аппаратуры космических систем ДЗЗ, со скоростью от 3 до 5 Гбит/с;
3. прием и передачу на обработку видеоинформации для программных средств СВ и ВВ;
4. хранение исходных, промежуточных и результирующих данных обработки видеоинформации;
5. оценку эффективности используемого алгоритма сжатия на различных наборах данных видеоинформации;
6. формирование отчета результатов испытаний ПС СВ и ПС ВВ в соответствии с методикой испытаний;
7. проведение диагностики, отладки и тестирования аппаратных и программных средств стендового оборудования.

БНДВ предназначена для формирования стендовым оборудованием тестовых потоков видеоданных и обеспечивает хранение файлов шаблонов в графическом формате BMP.

Структура программных средств представлена на рисунке 2:

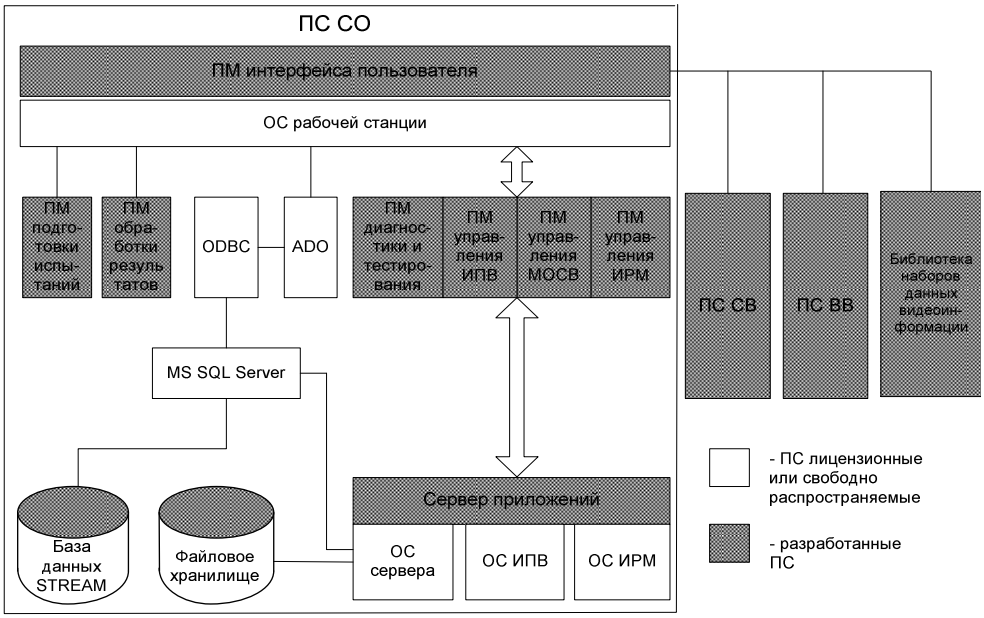

Рисунок 2 - Структура программных средств

Аппаратные средства комплекса (стендовое оборудование) предназначены для моделирования, отладки и испытания программных средств сжатия и восстановления видеоинформации и подтверждения возможности реализации алгоритмов ПС СВ в бортовой аппаратуре сжатия на МОСВ.

Структура аппаратных средств стендового оборудования (АС СО) представлена на рисунке 3:

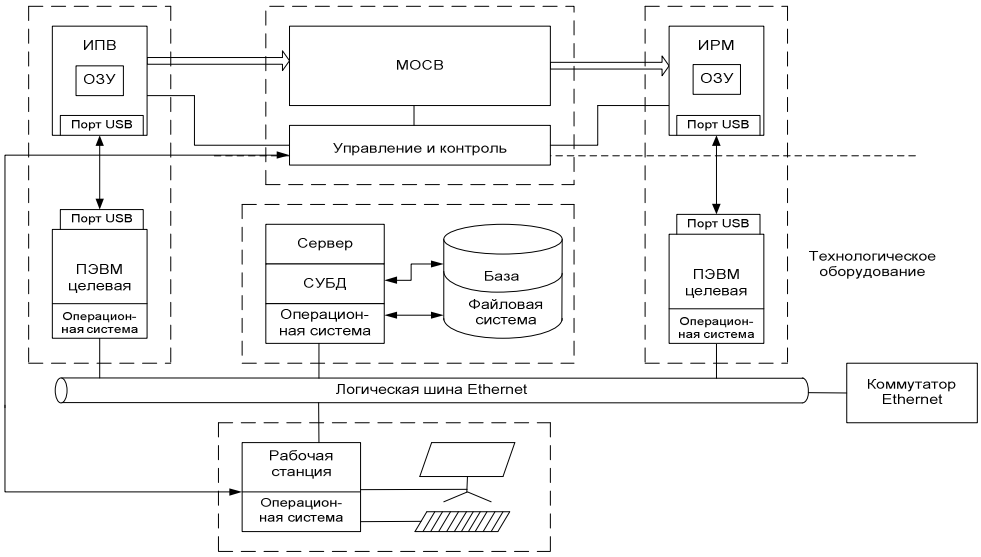

Рисунок 3 - Структура АС СО

ИПВ (имитатор потока видеоинформации) - предназначен для выдачи исходной видеоинформации в виде информационных потоков, имитирующих съемочную аппаратуру;

ИРМ (испытательное рабочее место) - предназначено для приема видеоинформации из МОСВ или ИПВ и питания МОСВ;

МОСВ (макетный образец бортовой аппаратуры сжатия видеоинформации) – предназначен для подтверждения возможности реализации выбранных алгоритмов ПС СВ в бортовой аппаратуре сжатия;

Технологическое оборудование - предназначено для технического обеспечения выполнения ОКР.

Внешний вид стендового оборудования и МОСВ представлены на рисунках 4 и 5 соответственно.

Рисунок 4 - Внешний вид стендового оборудования.

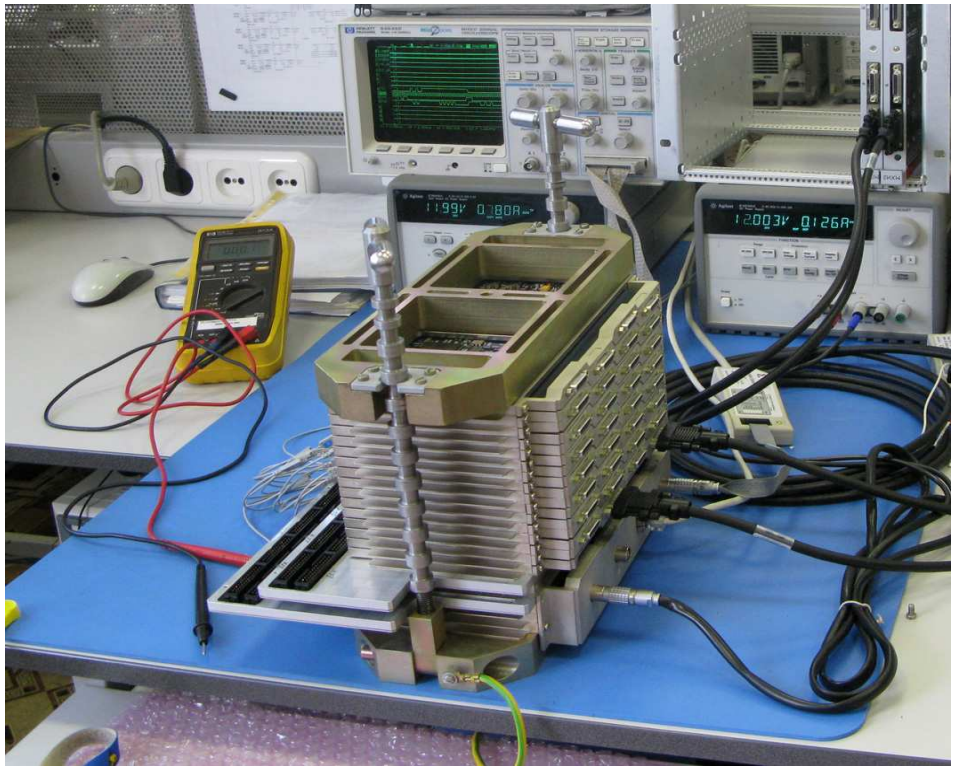

Рисунок 5 - Внешний вид МОСВ в процессе прошивки алгоритмов.

На рисунке 6 приведен МОСВ в собранном виде.

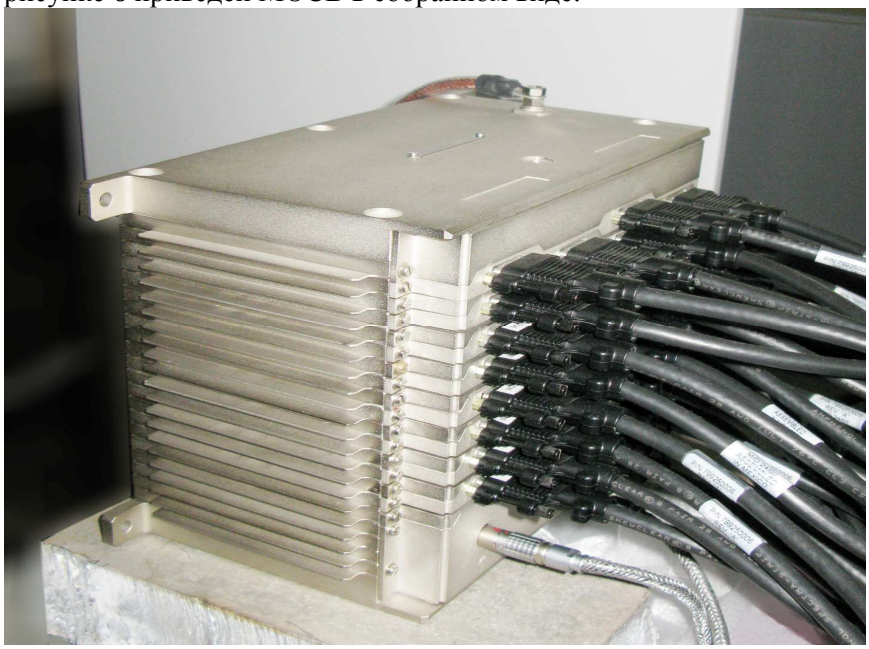

Рисунок 6. МОСВ в собранном виде.

В настоящий момент авторы работы проводят исследования под объединением моделей, разработанных на языке Ada и на языке VHDL. Язык VHDL используемые для моделирования аппаратуры и создания конфигурации для загрузки в FPGA. Такой инструментарий поставляется несколькими фирмами, в том числе и бесплатно, например WEB PACK ICE фирмы Xilinx. Основой при этом служит близость синтаксиса описания типов, функций, процедур и управляющих структур при описании тайлинга на языке Ada и VHDL [10]. Сравнение языков Ada и VHDL можно найти в работе [11].

Таким образом, использовалась программная модель, реализующая основные вычисления на языке Ada. Отработанная программная модель адаптировалась под конкретный кристалл FPGA на языке VHDL. Возможна автоматизация этого процесса. Например, JRS Research Labs [12] создала Ada-to-VHDL Translator.

Подход к моделированию и реализации параллельных вычислений на языке Ada позволяет достичь идеала – получить программный проект для различных платформ и проект микросхемы, который автоматически трансформируется при помощи кремневого компилятора в конфигурацию FPGA (например, WEB PACK ICE фирмы Xilinx) или в набор фотошаблонов для изготовления кристаллов.

ЛИТЕРАТУРА

1. Элементы параллельного программирования/ В.А. Вальковский, В.Е. Котов, А.Г. Марчук, Н.Н. Миренков; Под ред. В.Е. Котова. – М.: Радио и связь, 1983. – 240 с., ил.
2. СТБ 34.101.3, стандарт РБ.
3. ИСО/МЭК 15408-std.
4. Department of Defense Trusted Computer Security Evaluation Criteria. (Orange Book). – 12/85 (Dog 5200.28-std) – Rainbow Series.
5. Ada 95 Language Reference Manual ANSI/ISO 8652.1995-std. http://www.adapower.com/rm95/index.html.
6. Гавва А. Е. “Адское” программирование. Ada-95. Компилятор GNAT. http://www.ada-ru.org.
7. Киркорова Л. С. Экспериментальные исследования локальности алгоритмов перемножения матриц, реализованных на двухъядерных компьютерах. Курсовая работа. – Мн.: БГУ. 2008.
8. Лиходед Н.А. Методы распараллеливания гнезд циклов: Курс лекций. – Мн.: БГУ. 2007. – 100 с. Ser314\ subFaculty\ Каф. Дискр. мат. и алгор\ КУРСЫ ДМА\ 4 курс\ Лиходед\ Лекции\ Распараллеливание гнезд циклов
9. Ahmed N., Mateev N., Pingali K. Synthesizing transformations for locality enhancement of imperfectly-nested loop nests. Proceedings of the International Conference on Supercomputing. 2000. P. 141–152.
10. Язык описания аппаратуры цифровых систем – VHDL. Описание языка (ГОСТ Р 50754-95), стандарт, М., 1995 г.
11. Gregory P. Chapelle, Michael L. Lewis. Software Metrics Lead the Way to Better HDL Coding Practices – Integrated System Design Magazine, 2000.
12. Robert J. Sheraga ANSI C to Behavioral VHDL Translator, Ada to Behavioral VHDL Translator. The RASSP Digest – Vol. 3. September 1996.