ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2016. Т. 20, № 3 (73). С. 122–128 http://journal.ugatu.ac.ru

УДК 004.65

ПАРАЛЛЕЛЬНО-КОНВЕЙЕРНАЯ ФОРМА ПРОГРАММЫ ДЛЯ ПРОГРАММИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ГИБРИДНОГО ТИПА

А. И. Дордопуло[1], И. И. Левин[2], И. А. Каляев[3], В. А. Гудков[4], А. А. Гуленок[5], Г. А. Евстафьев[6]

[1] scorpio@mvs.tsure.ru, [2] levin@superevm.ru, [3] kaliaev@mvs.sfedu.ru, [4] slava_gudkov@mail.ru, [5] andrei_gulenok@mail.ru, [6] gaevstafiev@yahoo.com

[1,3-5] Научно-исследовательский институт многопроцессорных вычислительных систем Южного федерального университета, г. Таганрог, Россия
[2,6] ООО «Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров», г. Таганрог, Россия

Поступила в редакцию 21.07.2016

Аннотация. В статье предлагается использовать единую параллельно-конвейерную форму (ПКФ) для адаптации прикладной программы под аппаратный ресурс вычислительной системы гибридного типа. Единая параллельно-конвейерная форма программы дает возможность: масштабировать программу на уровне данных и на уровне разрядов, описывать различные формы организации вычислений, а также осуществлять быстрые преобразования между различными формами организации вычислений. Приведение исходной программы к ПКФ дает возможность автоматизировано выполнять преобразование программы для быстрой адаптации прикладной программы под изменившуюся архитектуру или конфигурацию вычислительной системы гибридного типа.

Ключевые слова: язык программирования высокого уровня, программирование вычислительных систем гибридного типа, технологии ресурсонезависимого программирования, единая параллельно-конвейерная форма.

ВВЕДЕНИЕ

Рост сложности решаемых прикладных задач требует постоянного повышения производительности вычислительных систем. Для достижения высокой реальной производительности разработчики ведут постоянный поиск новых архитектурных решений, а также создают новые технологии и средства программирования для эффективного решения прикладных задач.

Одним из перспективных способов достижения высокой реальной производительности вычислительной системы (ВС) при решении задач является адаптация архитектуры ВС под структуру решаемой задачи и создание специализированного вычислительного устройства, эффективно реализующего алгоритм вычислений.

Многообразие решаемых задач требует от ВС возможности изменения специализированного устройства для реализации типовых для различных предметных областей (математическая физика, символьная обработка, молекулярная динамика и др.) фрагментов вычислений. Большинство практических задач требует совмещения в едином вычислительном контуре как последовательных, так и параллельных вычислительных фрагментов для эффективной реализации как структурных, так и процедурных фрагментов вычислений [1], для выполнения которых необходимы различные по своей архитектуре вычислительные устройства. Решение этой проблемы многие разработчики видят в создании вычислительных систем с гибридной организацией вычислений, содержащих различные по архитектуре вычислительные узлы, на которых можно реализовать как структурные, так и процедурные вычисления в едином вычислительном контуре.

Симбиоз узлов с разной архитектурой в рамках единой вычислительной системы позволит повысить реальную производительность вычислительной системы за счет возможности эффективной реализации как структурных, так и процедурных фрагментов вычислений на узлах различной архитектуры вычислительной

системы гибридного типа (ВСГТ), которая содержит архитектурно-различные вычислительные узлы с различным типом организации вычислений и одинаковым способом обработки информации. ВСГТ может содержать реконфигурируемые вычислительные узлы и узлы универсальных микропроцессоров, в роли которых могут выступать универсальные процессоры, графические процессоры или ускорители Intel Xeon Phi.

В настоящее время для программирования таких вычислительных систем зачастую используются технологии программирования гетерогенных вычислительных систем: CUDA, OpenACC, OpenCL и т.д., в основе которых лежат расширения языков программирования C, C++, FORTRAN, учитывающие архитектуру специализированного микропроцессорного узла.

К существенным недостаткам этих технологий программирования относятся плохая переносимость готовых решений между ВС различной архитектуры и конфигурации и плохая масштабируемость программ. Основной причиной указанных недостатков, по нашему мнению, является подход к программированию ВС, при котором осуществляется разбиение задачи на отдельные фрагменты, каждый из которых реализуется на отдельном узле (на отдельном устройстве) гибридной вычислительной системы.

Фактически каждый задействованный узел ВС программируется отдельно, в результате чего любое изменение конфигурации ВС или изменение исходного кода прикладной программы приводит к необходимости повторного переразбиения задачи на фрагменты, созданию локальных программ для каждого узла ВС и их последующей синхронизации, которая также возлагается на программиста. Высокая сложность программирования является сдерживающим фактором широкого применение вычислительных систем гибридного типа для решения практических прикладных задач, поскольку для эффективного использования архитектурных преимуществ всех вычислительных узлов ВСГТ необходима разработка подпрограмм для каждого узла ВСГТ с учетом различных вариантов организации вычислений.

ТЕХНОЛОГИЯ РЕСУРСОНЕЗАВИСИМОГО ПРОГРАММИРОВАНИЯ ВСГТ

Создаваемые для ВСГТ программы, как правило, ориентированы на конкретную конфигурацию вычислительной системы, имеют конвейерную или параллельную организацию вычислений и ориентированы на фиксированную разрядность обрабатываемых данных. Адаптация подобных программ под текущую конфигурацию ВСГТ может потребовать изменения способа организации вычислений, модификации вычислительной структуры и организации информационных потоков данных и как правило приводит к частичной или полной переработки исходной программы, что значительно увеличивает время портации программы на ВСГТ другой архитектуры.

Для программирования ВСГТ необходимы средства описания различных вариантов организации вычислений в едином для различных архитектур языковом пространстве и средства трансляции параллельных прикладных программ, объединенные в технологию ресурсонезависимого программирования ВСГТ.

Для обеспечения функционирования унифицированных процессорных и реконфигурируемых вычислительных узлов в едином контуре необходима новая технология ресурсонезависимого программирования ВСГТ [2], базирующаяся на следующих принципах:

– адаптация программы под текущую конфигурацию ВСГТ выполняется в автоматизированном режиме специальной программой - препроцессором на основе рассчитанных параметров масштабирования программы [3];

– определение эффективных для текущей конфигурации ВСГТ параметров масштабирования программы (индукции или редукции производительности) должно выполняться автоматически средствами программирования без участия пользователя;

– для автоматизированного преобразования под текущую конфигурацию ВСГТ прикладная программа должна быть представлена в единой параллельно-конвейерной (канонической) форме.

Преобразование программы в единую параллельно-конвейерную форму является основой для применения автоматизированных средств технологии ресурсонезависимого программирования, обеспечивающих возможность как увеличения параллелизма задачи (индукцию) при увеличении аппаратного ресурса, так и возможность сокращения (редукции) при сокращении вычислительного ресурса.

Для практической реализации технологии ресурсонезависимого программирования ВСГТ необходим язык программирования высокого уровня, который позволит описывать различные формы организации вычислений и программиро-

вать унифицированные процессорные и реконфигурируемые вычислительные узлы в едином вычислительном контуре.

Специализированные языки программирования высокого уровня и технологии, используемые в настоящее время для вычислительных систем гибридного типа (OpenCl, OpenAcc и др.) обладают привычным для большинства программистов персональных ЭВМ синтаксисом языка С и отличаются между собой семантическими особенностями вызова и использования операторов. При этом для описания параллельных процессов в этих языках используется изначально последовательная парадигма языка С, ориентированная на взаимодействие последовательных процессов, что и не позволяет в полной мере использовать все возможности ВСГТ при разработке параллельных программ на этих языках.

Для программирования реконфигурируемых вычислительных систем и описания параллельных вычислений на основе принципов структурно-процедурной организации вычислений используется язык программирования высокого уровня COLAMO [1] с неявным описанием параллелизма.

Для реализации вычислений на универсальных процессорах в языке COLAMO есть средства описания процедурной организации вычислений (конструкция LocalProc) и возможность быстрого перехода от процедурной реализации вычислений на универсальных процессорах к структурной организации вычислений (конструкция SubCadr) с помощью неявного указания типа организации вычислений (конструкция Implicit). Переопределение способа реализации конструкции Implicit позволяет прикладному программисту без существенного изменения текста параллельной программы перейти от структурной организации вычислений к процедурной и обратно, что дает возможность программисту создавать единую прикладную программу для различных по архитектуре вычислительных узлов (ПЛИС и микропроцессоров) [4, 5].

Это позволяет рассматривать язык программирования высокого уровня COLAMO как основу для реализации технологии ресурсонезависимого программирования как реконфигурируемых вычислительных узлов, так и универсальных узлов ВСГТ.

КАНОНИЧЕСКАЯ ФОРМА ПРЕДСТАВЛЕНИЯ ПРОГРАММЫ

Под параллельно-конвейерной формой (ПКФ) представления переменных и массивов здесь и далее понимается описание данных на языке программирования высокого уровня COLAMO, обладающее одновременно как параллельным, так и последовательным типами доступа. На языке высокого уровня COLAMO такие типы доступа задаются с помощью ключевых слов Vector/Stream для элементов массивов данных [1] и BitVector/BitStream для битов [6] в скалярных значениях. Пример одновременного использования параллельного и последовательного типов доступа по данным и по разрядам к массивам A, B и C в канонической форме приведен на рис. 1.

```
Const N = 10;
Const M = 100;
Const BV = 32;
Const BS = 1;
Type Type32 [BV : BitVector, BS : BitStreamJ of Int;
Var a, b, c : Array Type32 [N: Vector, M: Stream] Mem;
Var i, j, k, t: Number;
Cadr ExpParallelStream;
For i :=0toN -1 do
  For j ;= 0 to M - I do
    For k:=0to BV -1 do
      Fort:=0to BS - 1 do
        Begin
          C[i, j][k, t] := A[i,j][k,t] - B[i,j][k,t];
        end;
EndCadr;
```

Рис. 1. Пример канонической формы программы на языке программирования COLAMO

Такая форма представления программы позволяет легко менять основные параметры параллельной программы на языке COLAMO (число одновременно реализуемых подграфов вычислений с помощью типов доступа Vector/Stream, разрядность обрабатываемых данных с помощью типов доступа BitVector/BitStream, число операций и др.). Эти преобразования можно выполнить автоматизировано с помощью программы-препроцессора без участия пользователя для адаптации под текущую конфигурацию ВСГТ. Одним из основных требований к выполняемым преобразованиям является их информационная эквивалентность, под которой понимается полное совпадение результатов выполнения исходной и преобразованной параллельной прикладной программы.

Приведение исходной программы к канонической форме представления выполняется в два этапа. На первом этапе выполняется преобразование переменных и их разрядности, а на втором этапе – преобразование конструкций программы к параллельно-конвейерной обработке. Метод преобразования программы на

языке COLAMO к канонической форме представления можно представить следующим образом.

1. Все массивы в исходной программе на языке программирования высокого уровня COLAMO преобразуются в ПКФ (при необходимости добавляются параллельный (Vector) или последовательный (Stream) типы доступа).

2. Все переменные и элементы массивов параллельной программы на языке COLAMO (кроме счетчиков цикла) преобразуются в ПКФ, содержащую параллельный (bitvector) и последовательный (bitstream) типы доступа.

3. Все подкадры параллельной программы на языке COLAMO, описанные конструкцией SubCadr, преобразуются в конструкции Implicit.

4. Все процедурные фрагменты параллельной программы на языке COLAMO, описанные конструкцией LocalProc, преобразуются в конструкции Implicit.

5. Все операторы, конструкции и циклы программы преобразуются согласно модифицированным параметрам переменных, при необходимости добавляются дополнительные циклы для доступа к данным или битам.

Преобразование переменных к параллельному и последовательному типу доступа (смешанному типу доступа) выполняется по следующим правилам:

– если для доступа к элементам массива использовался только последовательный тип доступа, то в объявлении массива добавляется параметр Vector единичной размерности, описывающий параллельный тип доступа;

– если для доступа к элементам массива использовался только параллельный тип доступа, то в объявлении массива добавляется параметр Stream единичной размерности, описывающий последовательный тип доступа;

– если для доступа к элементам массива использовался смешанный тип доступа, то преобразования для данного массива не выполняются.

Таким образом, при модификации описания массива выполняется расширение его размерности. Так, при объявлении массива A следующего вида:

Var A : Array Integer [N : Stream] Mem,

должно быть выполнено преобразование к объявлению вида

Var A : Array Integer [M : Vector, K : Stream] Mem,

где $M * K = N$.

Для обеспечения эквивалентности информационных графов исходной программы и модифицированной программы начальное значение M=1.

На рис. 2 показан пример преобразования исходной программы (*а*) к канонической форме (*б*).

```
Const N = 10;
Var a, b, c, d : Array
        Integer [N: Vector] Mem;
Var i : Number;
Cadr ExpParallel;
   For i := 0 to N - 1 do
    Begin
      If (A[i] > 5)
        C[i] := A[i] - B[i];
      Else
        C[i] := A[i] + D[i];
    end;
EndCadr;
```

а – исходная программа

```
Const N = 10;
Const M = 10;
Const K = N/M;
Var a, b, c, d : Array Integer [M : Vector, K : Stream] Mem;
Var i, vc_1 : Number;
Cadr ExpParallelConvData;
   For vc_1 := 0 to K - 1 do
    Begin
     For i := 0 to M - 1 do
      Begin
        If (A[i, vc_1] > 5)
          C[i, vc_1] := A[i, vc_1] - B[i, vc_1];
        Else
          C[i, vc_1] := A[i, vc_1] + D[i, vc_1];
      end;
    end;
EndCadr;
```

б – преобразованная программа

Рис. 2. Преобразование программы к канонической форме на уровне типа доступа к данным

Как видно из текста параллельно-конвейерной программы (рис. 2, *б*), все одномерные массивы были преобразованы к двумерным массивам, имеющим смешанный тип доступа (M : Vector, K : Stream), добавлен новый оператор цикла с индексной переменной VC_1 и выполнена модификация всех обращений к переменным.

Для эффективной адаптации программы для ВСГТ такие же преобразования должны быть выполнены и для разрядов. Преобразование по разрядам выполняется для всех переменных исходной программы и заключается в создании произвольного типа данных с параллельным (ключевое слово BitVector) и последовательным (ключевое слово BitStream) типом доступа, общая размерность которого не должна превышать разрядности исходной преобразуемой переменной.

Для создания нового типа переменных в языке высокого уровня COLAMO используется оператор Type следующего вида:

Type Type16 [N : Bitvector, X: BitStream] of TypeData;,

где Type16 – идентификатор нового типа данных, BitVector – указывает на параллельный тип доступа к разрядам данных, BitStream – указывает на последовательный тип доступа к разрядам данных, TypeData – определяет знак и формат данных.

На рис. 3 показан пример программы, обеспечивающей масштабирование по разрядам, полученный в результате преобразования исходной программы, представленной на рис. 2, *б*.

```
Const N = 10;
Const M = 10;
Const K = N/M;
Const BV = 32;
Const BS = 32/BV;
Type Type32 [BV : BitVector, BS : BitStream] of Int;
Var a, b, c, d : Array Type32 [M : Vector, K : Stream] Mem;
Var i, vc_1, vc_2, vc_3 : Number;
Cadr ExpParallelConvBit;
For vc_1 := 0 to K - 1 do
  For i := 0 to M - 1 do
    For vc_2 := 0 to K - 1 do
      For vc_3 := 0 to M - 1 do
        Begin
          If (A[i, vc_1][vc_2, vc_3] > 5)
           C[i, vc_1][vc_2, vc_3] :=
    A[i, vc_1] [vc_2, vc_3] - B[i, vc_1][vc_2, vc_3];
          Else
           C[i, vc_1][vc_2, vc_3] :=
    A[i,vc_1] [vc_2, vc_3] + D[i, vc_1][vc_2, vc_3];
        end;
EndCadr;
```

Рис. 3. Преобразование программы к канонической форме на уровне типа доступа к данным и битам

Для описания масштабирования по разрядам для каждой переменной или группы переменных, связанных информационной зависимостью между собой, создается новый тип данных с параллельным и последовательным типом доступа, при этом размер параметра с последовательным типом доступа должен быть равен 1 для обеспечения эквивалентности информационных графов исходной программы и модифицированной программы, т.е. X=1. Из текста параллельно-конвейерной программы (рис. 3) видно, что стандартный тип данных Integer был заменен на новый тип данных Type32, описывающий возможность параллельного и последовательного доступа к разрядам (BV : BitVector, BS : BitStream), добавлены новые операторы цикла с индексными переменными VC_2 и VC_3 и выполнена модификация всех обращений к переменным. Таким образом, переход к новым типам данных позволяет быстро изменять степень масштабирования по разрядам путем модификации значения константы BV.

Следующим этапом преобразования программы к ПКФ является преобразование вычислительных конструкций подкадр и LocalProc к универсальной вычислительной конструкции Implicit для организации быстрого перехода от процедурной организации вычислений к структурной и обратно. Конструкция Implicit является конструкцией с неявным указанием реализуемой в фрагменте формы организации вычислений (структурной или процедурной).

На рис. 4 представлен пример преобразования конструкции подкадр к конструкции Implicit с указанием способа организации вычислений (структурная или процедурная).

```
var q, w, e, r, t: Array Integer [N1:
Stream] Mem;

subcadr BaseNode(in: In_q, In_w, In_e,
In_r; out: Out_X);
  var In_q, In_w, In_e, In_r, Out_X:
Integer Com;
  var com1, com2: Integer Com;
    com1 := In_q + In_w;
    com2 := In_e + In_r;
    Out_X := com1 * com2;
EndSubCadr;

Cadr Cadr1;
  for i := 0 to N1-1 do
  begin
    BaseNode(q[i], w[i], e[i], r[i], t[i]);
  end;
EndCadr;
```

а – Реализация в виде подкадра (SubCadr)

```
Define BASENODE is TSubCadr;
Implicit BASENODE( In: IN_Q, IN_W, IN_E,
IN_R; Out: OUT_X ) ;

Var IN_Q, IN_W, IN_E, IN_R: Integer Com;
Var OUT_X : Integer Com;
Var COM1, COM2 : Integer Com;

COM1 := IN_Q + IN_W ;
COM2 := IN_E + IN_R ;
OUT_X := COM1 * COM2 ;
EndImplicit;

Cadr Cadr1;
 For I:= 0 To N1-1 Do
  Begin
   BASENODE(Q[I], W[I], E[I], R[I] , T[I]) ;
  End;
EndCadr;
```

б – Реализация в виде конструкции Implicit

Рис. 4. Преобразование подкадра и LocalProc к универсальной вычислительной конструкции Implicit

Как видно из текста параллельно-конвейерной программы (рис. 4, *б*) подкадр BaseNode (рис. 4, *а*) был преобразован к универсальной конструкции Implicit с указанием структурной реализации вычислений в операторе Define.

Переопределение способа реализации конструкции Implicit позволяет прикладному программисту без существенного изменения текста параллельной программы перейти от структурной организации вычислений к процедурной и обратно. Наличие конструкций описания как структурных, так и процедурных фрагментов вычислений прикладной задачи дает возможность создавать единую прикладную программу для всех узлов ВСГТ.

ПРЕОБРАЗОВАНИЕ ПРОГРАММЫ ПОД ТЕКУЩУЮ КОНФИГУРАЦИЮ ВСГТ

После преобразования исходной параллельной программы в каноническую форму модуль анализа препроцессора языка COLAMO подсчитывает необходимый для ее реализации аппаратный ресурс и сопоставляет его с текущей конфигурацией ВСГТ для определения максимальной степени и типов необходимых преобразований. Если текущего аппаратного ресурса ВСГТ достаточно для выполнения программы, то синтезируются загрузочные модули для задействованных узлов ВСГТ, в противном случае выполняются специальные редукционные преобразования [2, 3], сбалансированно сокращающие производительность программы и задействованный аппаратный ресурс ВСГТ. Редукция производительности выполняется в следующем порядке: редукция по одновременно выполняемым подграфам программы, редукция по разрядности, редукция по командам (устройствам), редукция по скважности (частоте). Сокращение занимаемого аппаратного ресурса для каждого вида редукции с учетом ее теоретически допустимой степени выполняет модуль анализа препроцессора, который определяет наиболее рациональный вариант использования редукции для обеспечения максимально возможной производительности программы для текущей конфигурации ВСГТ. Полученный в автоматизированном режиме после препроцессора текст редуцированной параллельной программы передается транслятору языка программирования COLAMO, который создает развернутый информационный граф прикладной задачи. Информационный граф прикладной задачи, содержащий фрагменты, реализуемые структурно на реконфигурируемых вычислительных узлах и процедурно на микропроцессорных вычислительных узлах, передается программе-синтезатору для автоматического распределения фрагментов задачи по доступным в текущей конфигурации ВСГТ реконфигурируемым и микропроцессорным вычислительным узлам. Согласование потоков данных между различными по типам организации вычислений узлам ВСГТ осуществляется на основе файла описания конфигурации ВСГТ, содержащего данные о типах синхронизируемых вычислительных узлов, разрядности шины данных, частоте их работы, скважности подачи данных и др. После установки необходимых интерфейсов и элементов синхронизации программа-синтезатор создает загрузочные конфигурационные файлы *.bit для реконфигурируемых вычислительных узлов и загрузочные файлы *.exe для микропроцессорных узлов ВСГТ и единую для всех вычислительных модулей ВСГТ управляющую вычислительным процессом программу.

ЗАКЛЮЧЕНИЕ

Описание в едином вычислительном контуре различных форм организации параллельных вычислений, необходимых для эффективного программирования ВСГТ, позволяет использовать язык программирования высокого уровня COLAMO как основу программирования таких вычислительных систем.

Разработанная единая параллельно-конвейерная форма позволяет описывать различные формы организации вычислений, масштабировать программу как на уровне потоков данных, так и на уровне разрядности обрабатываемых данных, что обеспечивает ресурсонезависимость программирования и возможность адаптации исходного текста программы под любую конфигурацию ВСГТ.

Использование автоматизированных средств (препроцессора) для преобразования программы без участия программиста обеспечивает пользователя набором необходимых средств для быстрой разработки эффективных масштабируемых параллельных программ, что снижает сложность программирования ВСГТ, позволяет рационально использовать ресурсы узлов с разной архитектурой и повышает скорость разработки параллельных прикладных программ.

СПИСОК ЛИТЕРАТУРЫ

1. **Гузик В. Ф., Каляев И. А., Левин И. И.** Реконфигурируемые вычислительные системы: учеб. пособие / под общ. ред. И. А. Каляева. Ростов-на-Дону: Изд-во ЮФУ, 2016. 472 с. **[** V. F. Guzik, I. A. Kalyaev, I. I. Levin. *Reconfigurable computer systems: tutorial* / Edited by I. A. Kalyaev, (in Russian). Rostov-on-Don: SFU Publishing, 2016. 472 p.**]**.

2. **Программирование** вычислительных систем гибридного типа на основе метода редукции производительности / И. И. Левин [и др.] // Параллельные вычислительные технологии ПаВТ'2016: Междунар. науч. конф. (Архангельск, 28 марта–1 апреля 2016): тр. конф. Челябинск: Издательский

центр ЮУрГУ, 2016. С.131–140. **[** I. I. Levin, *et al.*, “Programming of hybrid computer systems based on the method of performance reduction”, (in Russian), in *Proc. Int. Workshop on Parallel computational technologies (PCT-2016)*, Arkhangelsk, Russia, 2016, pp. 131-140. **]**.

3. **Каляев И. А., Дордопуло А. И., Левин И. И., Гудков В. А., Гуленок А. А.** Технология программирования вычислительных систем гибридного типа // Вычислительные технологии. 2016. Новосибирск: Изд-во ИВТ СО РАН, 2016. Т. 21, № 3. С. 33–44. **[** I. A. Kalyaev, *et al*., “Programming technology fo hybrid computer systems“, (in Russian), in *Vichislitelniye tekhnologii*, vol. 21, no. 3, pp. 33-44, 2016.**]**

4. **Kalyaev I. A., Levin I. I., Dordopulo A. I., Slasten L. M.** Reconfigurable Computer Systems Based on Virtex-6 and Virtex-7 FPGAs // IFAC Proceedings Volumes (**ISSN** 14746670), Programmable Devices and Embedded Systems. 2013. Vol. 12, part 1. P. 210–214. **[** I. A. Kalyaev, *et al*., “Reconfigurable Computer Systems Based on Virtex-6 and Virtex-7 FPGAs”, in *IFAC Proceedings Volumes (**ISSN** 14746670), Programmable Devices and Embedded Systems*, 2013, vol. 12, part 1., pp. 210-214. **]**

5. **Dordopulo A., Kalyaev I., Levin I., Slasten L.** High-performance reconfigurable computer systems // Lecture Notes in Computer Science, Volume 6873, **Chapter** Parallel Computing Technologies, P. 272–283. **[** Dordopulo A., *et al*., “High-performance reconfigurable computer systems“, in *Lecture Notes in Computer Science, **Chapter** Parallel Computing Technologies,* 2015, vol. 6873, pp. 272-283.]

6. **Семерникова Е. Е., Левин И. И., Гудков В. А.** Организация битовой обработки данных для реконфигурируемых вычислительных систем на языке программирования высокого уровня // Вестник компьютерных и информационных технологий. М.: Изд-во Машиностроение, 2015. № 5. С. 3–9. **[** E. E. Semernikova, I. I. Levin, and V. A. Gudkov, “Organization of data bit processing for reconfigurable computer systems in a high-level programming language”, (in Russian), in *Vestnik kompjuternikh I informatsionnykh tekhnologiy*, no. 5, pp. 3-9, 2015. **].**

ОБ АВТОРАХ

ДОРДОПУЛО Алексей Игоревич, зав. лаб., инж. по спец. «Организация и технология защиты информации» (ТРТУ, 2000). Канд. техн. наук (ТРТУ, 2003). Иссл. в обл. высокопроизводительных МВС и их программного обеспечения.

ЛЕВИН Илья Израилевич, дир., инж.-конструктор-технолог ЭВА (ТРТИ, 1984). Д-р техн. наук (НИИ МВС ТРТУ, 2005), проф. (НИИ МВС ЮФУ, 2013). Иссл. в обл. высокопроизводительных МВС.

КАЛЯЕВ Игорь Анатольевич, главный науч. сотр., инж.-системотехник (ТРТИ, 1980). Чл.-кор. РАН (2003), д-р техн. наук (ЛИАП, 1989), проф. (НИИ МВС ТРТУ, 1998). Иссл. в обл. многопроцессорных вычислительных и информационно-управляющих систем.

ГУДКОВ Вячеслав Александрович, С.н.с., инж. по программному обеспечению (ТРТУ, 2005). Канд. техн. наук (НИИ МВС ЮФУ, 2010). Иссл. в обл. параллельного программирования на РВС.

ГУЛЕНОК Андрей Александрович, С.н.с., инж. по программному обеспечению (ТРТУ, 2005). Канд. техн. наук (НИИ МВС ЮФУ, 2011). Иссл. в обл. параллельного программирования на РВС.

ЕВСТАФЬЕВ Георгий Александрович, Программист., инж. по программному обеспечению (ТРТУ, 2005). Иссл. в обл. параллельного программирования на РВС.

METADATA

Title: A parallel-pipeline form of application for programming of hybrid computer systems

Authors: A.I. Dordopulo[1], I.I. Levin[2], I.A. Kaliaev[3], V.A. Gudkov[4], A.A. Gulenok[5], G.A. Evstafiev[6]

Affiliation:

[1345] Scientific Research Institute of Multiprocessor Computer Systems at Southern Federal University, Taganrog, Russia.

[26] Scientific Research Centre of Supercomputers and Neurocomputers, Co Ltd, Taganrog, Russia

Email: [1]scorpio@mvs.tsure.ru.

Language: Russian.

Source: Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 20, no. 3 (73), pp. 122-128, 2016. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract:

In the paper we suggest to use a single parallel-pipeline form (PPF) for adaptation of the application to the HCS hardware resource. The single parallel-pipeline form of the application allows: scaling of the application on data level and on bit level, description of various forms of organization of calculations, and fast transformations between various forms of organization of calculations. Transformation of the initial application to the PPF provides automatic transformation of the application for its fast adaptation to the modified architecture or configuration of the hybrid computer system (HCS).

Key words: high-level programming language, programming of hybrid computer systems, technologies of resource independent programming, single parallel-pipeline form.

About authors:

DORDOPULO Alexey Igorevich, head of laboratory, engineer in the area “Organization and technology of information security” (TSURE, 2000). Candidate of technical sciences (TSURE, 2003). Researcher in the area of high-performance MCS and their software.

LEVIN Ilya Izrailevich, director, design and production engineer of computer equipment (TSIRE, 1984). Doctor of technical sciences (SRI MCS TSURE, 2005), professor (SRI MCS SFU, 2013). Researcher in the area of high-performance MCS.

KALYAEV Igor Anatolievich, chief research engineer, system analyst (TSURE, 1980). Correspondent member of the RAS (2003), Doctor of technical sciences (LIA, 1989), professor (SRI MCS TSURE, 1998). Researcher in the area of multiprocessor computer and data control systems.

GUDKOV Vyacheslav Alexandrovich, senior staff scientist, software engineer (TSURE, 2005). Candidate of technical sciences (SRI MCS SFU, 2010). Researcher in the area of parallel RCS programming.

GULENOK Andrey Alexandrovich, senior staff scientist, software engineer (TSURE, 2005). Candidate of technical sciences (SRI MCS SFU, 2011). Researcher in the area of parallel RCS programming.

EVSTAFIEV Georgiy Alexandrovich, programmer, software engineer (TSURE, 2005). Researcher in the area of parallel RCS.