УДК 004.43

В. Б. Коваленко[1], канд. техн. наук, ст. науч. сотр., e-mail: vereten@hotbox,
А. И. Дордопуло[2], канд. техн. наук, зав. лаб., e-mail: scorpio@mvs.tsure.ru,
В. А. Гудков[2], канд. техн. наук, ст. науч. сотр., e-mail: gudkov@mvs.tsure.ru,
А. А. Гуленок[2], канд. техн. наук, ст. науч. сотр., e-mail: andrei_gulenok@mail.ru,
Л. М. Сластен[2], канд. техн. наук, ст. науч. сотр., e-mail: lmslasten@yandex.ru
[1] Федеральное государственное бюджетное учреждение науки Южный научныи центр РАН, г. Ростов-на-Дону
[2] НИИ многопроцессорных вычислительных систем им. акад. А. В. Каляева ЮФУ, г. Таганрог

Использование софт-архитектур при решении задач цифровой обработки сигналов на реконфигурируемых вычислительных системах

Рассматривается макрообъектный подход к программированию реконфигурируемых вычислительных систем, позволяющий сократить время программирования задачи за счет уменьшения времени трансляции параллельной программы.

Ключевые слова: *многопроцессорные системы, суперкомпьютеры, реконфигурируемые вычислительные системы, архитектуры вычислительных систем, параллельное программирование, языки программирования*

Введение

Программирование реконфигурируемых вычислительных систем (РВС), обладающих полем связанных программируемых логических интегральных схем (ПЛИС), является сложной и трудоемкой задачей. На данный момент существуют два метода программирования РВС:

- программирование с применением стандартных средств программирования, основанных, как правило, на языках HDL-группы;
- использование языков высокого уровня, таких как COLAMO, Handel-C, SystemC.

Каждый из существующих на данный момент методов программирования РВС имеет свои достоинства и недостатки. Программирование РВС с использованием стандартных средств, предоставляемых производителями ПЛИС, позволяет достигать максимальной производительности, однако требует длительного времени для разработки решений. Кроме того, подобные средства позволяют программировать лишь отдельные ПЛИС, что приводит к возникновению сложностей при объединении нескольких ПЛИС в единую вычислительную структуру.

Более эффективным методом программирования РВС является применение языков высокого уровня, таких как COLAMO, Handel-C, SystemC и др. Сокращение времени программирования при этом достигается за счет существенного ускорения написания текста программ. Современные системы программирования РВС, основанные на языках высокого уровня, зачастую позволяют сократить время программирования до нескольких недель, однако трансляция программ на универсальные процессоры или промежуточные решения, предзагруженные в ПЛИС (Handel-C, SystemC), приводят к сокращению реальной производительности вычислительной системы, а трансляция на уровень логических ячеек ПЛИС (COLAMO) зачастую требует значительных временны́х затрат.

Еще один метод программирования РВС основан на использовании динамически перестраиваемых архитектур, предзагруженных в РВС, что позволяет сократить время разработки прикладной программы РВС за счет уменьшения времени трансляции прикладных программ. При этом производительность вычислительной системы остается на уровне, сопоставимом с производительностью специализированных вычислительных структур. Это достигается путем создания и предзагрузки в РВС вычислительных архитектур, способных решать задачи определенного класса. Опыт создания вычислительных структур на РВС показывает, что для каждого класса задач можно выделить конечный набор вычислительных узлов, с помощью которых может быть построена вычислительная архитектура, эффективно решающая задачи данного класса. Вы-

числительные узлы, управляемые посредством системы команд и связанные в единую вычислительную структуру, составляют софт-архитектуру [1] РВС. При решении задач на РВС с использованием софт-архитектур процесс программирования разбивается на два основных этапа:

- создание софт-архитектуры РВС на основе разработанных ранее вычислительных блоков;
- решение прикладной задачи путем настройки разработанной ранее софт-архитектуры в соответствии с алгоритмом решаемой задачи.

При этом единожды созданная софт-архитектура загружается в ПЛИС РВС, а трансляция прикладной программы осуществляется на уровень команд вычислительных блоков софт-архитектуры. Подобный подход позволяет сократить время отладки прикладных программ в 2—3 раза по сравнению с существующими методами программирования РВС за счет сокращения времени трансляции.

Возможность программирования РВС на уровне пользовательских софт-архитектур ранее была высказана в ряде работ [2—4], однако методы и средства создания и программирования софт-архитектур разработаны не были. В НИИ многопроцессорных вычислительных систем ЮФУ (НИИ МВС ЮФУ) в 2013 г. в рамках государственного контракта по теме "Разработка реконфигурируемой вычислительной системы РВС-7 и организация на ее основе производства реконфигурируемых вычислительных систем с производительностью до 10^{15} операций в секунду в одностоечном конструктиве 47U" был создан программный комплекс, позволяющий разрабатывать различные пользовательские софт-архитектуры и решать на них задачи цифровой обработки сигналов. Это позволило на практике апробировать принципы и методы программирования, сформулированные в более ранних работах [5].

Иерархия софт-архитектур

Софт-архитектуры имеют иерархическое строение, при котором устройства, расположенные на более низких уровнях, являются элементами для построения устройств на более высоком уровне. Иерархия элементов построения софт-архитектур приведена на рис. 1. Построение софт-архитектур осуществляется на основе макрообъектов, имеющих независимое управление и объединенных информационными и синхронизирующими связями. Макрообъект представляет собой программно-аппаратную заготовку, содержащую в себе не только схемотехнические элементы, но и средства их программирования, которые позволяют осуществлять перенастройку без управления извне.

На нижнем уровне иерархии софт-архитектуры находятся объекты, являющиеся неделимыми элементами при построении более сложных элементов архитектуры. При этом объекты являются единственными элементами архитектуры, разрабатываемыми схемотехнически. Все остальные элементы формируются только с помощью языка описания софт-архитектур SADL.

Рис. 1. Иерархия элементов софт-архитектуры

На следующем уровне располагаются узлы, которые формируются из объектов и созданных ранее узлов. Узлы не могут непосредственно участвовать в построении софт-архитектуры, так как не обладают возможностью самостоятельно настраиваться на решение задач и требуют внешнего управления. Использование узлов позволяет структурировать описание софт-архитектуры в целях упрощения понимания и сопровождения кода.

На следующем уровне располагаются макрообъекты. В качестве элементов макрообъектов выступают объекты и узлы. Формирование макрообъектов и связей между ними выполняется с учетом того, что макрообъект должен иметь возможность самостоятельно настраиваться на решение задач. По этой причине в состав макрообъекта включено устройство, осуществляющее управление информационными потоками и настройку макрообъекта на решение задачи.

Разработка софт-архитектур реконфигурируемых вычислительных систем

Транслятор языка SADL вошел в состав комплекса программного обеспечения (КПО) [6], реализующего программирование РВС на уровне логических ячеек ПЛИС с использованием транслятора языка COLAMO. Для программирования РВС с использованием софт-архитектур потребовалась модернизация структуры КПО. В частности, в структуру КПО были добавлены: транслятор языка описания софт-архитектур SADL, синтезатор конфигураций параллельно-конвейерных вычислитель-

ных структур из макрообъектов Steam! Constructor, библиотека элементов софт-архитектур. В настоящий момент программный комплекс состоит из следующих компонентов:

- транслятора языка SADL;
- транслятора языка высокого уровня COLAMO;
- транслятора языка Argus;
- синтезатора разработки многокристальных схемотехнических решений Fire!Constructor;
- синтезатора Steam!Constructor.

На рис. 2 представлена структура системного программного обеспечения, позволяющая создавать софт-архитектуры для РВС и решать с их помощью прикладных задач.

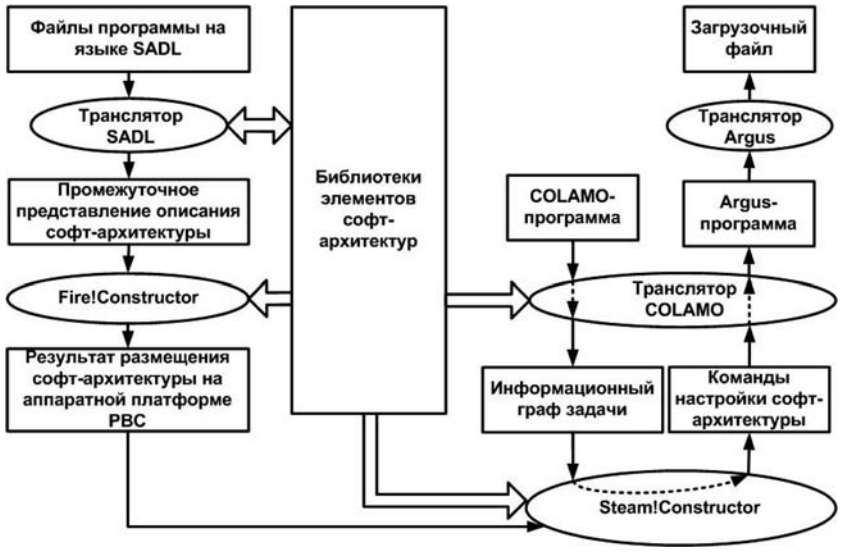

Рис. 2. Структура системного программного обеспечения для решения задач с использованием софт-архитектуры

На основе COLAMO-программы транслятор генерирует информационный граф задачи и указание, на какую из доступных архитектур он должен быть оттранслирован. Информационный граф передается синтезатору Steam!Constructor, который размещает его на софт-архитектуре. Результатом работы синтезатора Steam!Constructor является набор команд для настройки софт-архитектуры. На следующем этапе трансляции в соответствии с командами настройки и потоковой составляющей параллельной COLAMO-программы компилируются загрузочный файл и управляющая программа. Загрузочный файл содержит кодировки для управления софт-архитектурой, а управляющая программа предназначена для управления информационными обменами между вычислительной системой РВС и персональным компьютером.

Работоспособность приведенных методов и комплекса системного программного обеспечения апробирована на задачах цифровой обработки сигналов.

Рассмотрим более подробно процесс создания и программирования софт-архитектур на задаче нахождения спектра комплексного сигнала с использованием алгоритма быстрого преобразования Фурье (БПФ). Разработан следующий перечень элементов, требуемых для решения поставленных задач цифровой обработки сигналов:

- контроллер распределенной памяти, позволяющий управлять информационными потоками;
- функциональные устройства, обеспечивающие выполнение операции суммирования, вычитания, умножения;
- АЛУ, выполняющее операции суммирования или вычитания в зависимости от управляющего сигнала;
- набор мультиплексоров и демультиплексоров для управления информационными потоками внутри кадра;
- память ПЗУ, хранящая коэффициенты W для реализации алгоритма БПФ;
- синхронизирующая память, устанавливаемая между итерациями БПФ, необходимая для формирования информационных потоков в соответствии с графом алгоритма БПФ;
- память ОЗУ, в которой хранятся коэффициенты ядра свертки.

После того как перечень элементов определен, необходимо выполнить их схемотехническое описание. Файлы, содержащие схемотехническое описание каждого объекта, помещаются в библиотеку элементов софт-архитектуры и в дальнейшем используются синтезатором Fire!Constructor для размещения на аппаратной платформе.

На следующем этапе процесса создания софт-архитектуры для каждого из ее элементов разрабатываются файлы описания на языке SADL. В них указывается информация о входах и выходах элемента, его типе, а также описываются вычислительные возможности элемента. В качестве примера приведем описание АЛУ, выполняющего суммирование или вычитание:

```
add_sub_32:ALU;
add_sub_32 (In: in1, in2; Out: out1; InH: H1; VHDLFile:
AS_ALU_32.VHDL);
Var
    in1, in2, outl : integer size 32;
    H1 : integer size 32;
Begin
    Case h1 of
        0: out1 = in1 + in2;
        1: out1 = in1 – in2;
    End;
End;
```

В верхней строке описания элемента указываются его имя (add_sub_32) и тип (ALU). Далее в заголовке выполняется описание его входов/выходов, а также имя VHDL-файла, содержащего схемотехническое описание элемента. В разделе Var

указываются типы входов/выходов и типы переменных, использованных при описании шаблонов вычислений. Далее следует раздел описания шаблонов, которые может выполнять данный элемент.

В приведенном описании АЛУ по управляющему сигналу 0 выполняет суммирование двух чисел из потоков данных in1 и in2, по управляющему сигналу 1 — вычитание.

На основе ранее описанных элементов путем соединения их входов и выходов строятся более сложные объекты: узлы и макрообъекты. Ниже приведен пример узла, выполняющего базовую операцию БПФ:

```
UsesSA;                                  // используемая библиотека элементов софт-архитектуры
Node_BO :Node;                           // имя элемента и тип элемента
Node_BO(In:ReA, ImA, ReB, ImB;           // заголовок элемента с
Out:ReA1, ImA1, ReB1, ImB1; InH: h1);    // указанием типов входов/выходов
Var                                      // описание типов переменных
    AddFU1, AddFU2, AddFU3 : ADD_32X32;
    SubFU1, SubFU2, SubFU3 : SUB_32X32;
    MFU1, MFU2, MFU3, MFU4 : MUL_32X32;
    MemDSP1: KOEF_W64X2048;
    ALU_1, ALU_2 : ADD_SUB_32X32;
    ReA, ImA, ReB, ImB, ReA1, ImA1, ReB1, ImB1: integer size 32;
    ReW, ImW: integer size 32;
    h1: integer size 32;
    a1, a2, b1, b2, b3, b4 : integer size 32;
Begin                                    // описание структуры узла
    MemDSP1(Out: ReW, ImW;               // память коэффициентов БПФ
    Ins: ImB; InH: h1);
    AddFU1(In: ReA, ReB; Out: ReA1);     //сумматор
    AddFU2(In: ImA, ImB; Out: ImA1);     // сумматор
    SubFU1(In: ReA, ReB; Out: a1);       // вычитатель
    SubFU2(In: ImA, ImB; Out: a2);       // вычитатель
    MFU1(In: a1, ReW; Out: b1);          //умножитель
    MFU2(In: a2, ImW; Out: b2);          // умножитель
    MFU3(In: a2, ReW; Out: b3);          // умножитель
    MFU4(In: a1, ImW; Out: b4);          // умножитель
    AddFU3(In: b1, b2; Out: ReB1);       // сумматор
    SubFU3(In: b3, b4; Out: ImB1);       // вычитатель
End;
```

В заголовке описания указываются имя и тип объекта, а также описание его входов и выходов. Так как в состав объекта могут входить различные элементы, описанные ранее, то в разделе описания переменных необходимо указать тип каждого из использованных элементов. Кроме описания типов элементов в данном разделе приводится описание коммутационных переменных, объединяющих элементы в единую вычислительную структуру. В теле описания формируется вычислительная структура будущего узла путем соединения входов и выходов элементов. Из описания видно, что вычислительная структура узла Node_BO состоит из функциональных устройств (сумматоров, вычитателей и умножителей) и памяти DSPMem1, отвечающей за подачу коэффициентов БПФ в соответствии с сигналами управления, поступающими по шине H1. Графическое представление приведенного выше описания приведено на рис. 3.

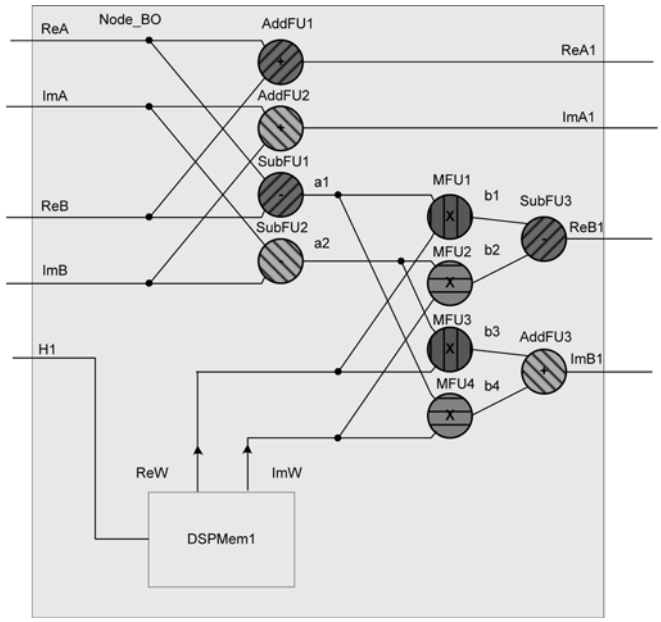

Рис. 3. Узел Node_BO

Софт-архитектура для решения задачи включает в себя следующие объекты (рис. 4):

- восемь контроллеров распределенной памяти (DMC1...DMC8), выполняющих хранение входных, выходных и промежуточных данных алгоритма;
- четыре узла (BONode1, ..., BONode4), реализующих вычисления в соответствии с "бабочкой" БПФ;
- четыре синхронизирующие памяти (DPMem1, ..., DPMem4), формирующие потоки данных в соответствии с алгоритмом БПФ;
- четыре мультиплексора (Mx1, ..., Mx4) и четыре демультиплексора (DMx1, ..., DMx4), с помощью которых программист организует перенаправление информационных потоков.

Узел Node_BO является базовым узлом при решении задач цифровой обработки сигналов. Компонуя различные узлы и элементы в единую вычислительную структуру, архитектор имеет возможность создавать уникальные софт-архитектуры. На рис. 4 приведена типовая структура макрообъекта для решения задач цифровой обработки сигналов, основанных на алгоритме БПФ. Вычислительный конвейер софт-архитектуры SA_DSP состоит из четырех ступеней, каждая из которых выполняет одну итерацию алгоритма БПФ. Во всех ступенях конвейера имеется память типа DPMem для вре-

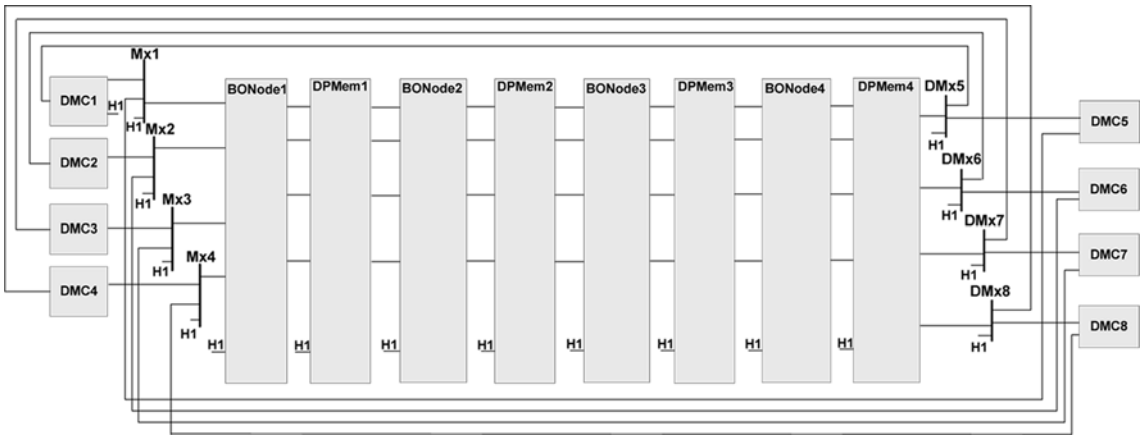

Рис. 4. Структура макрообъекта MO_DSP

менно́го согласования данных при выполнении базовых операций соседних итераций алгоритма БПФ. Задача нахождения спектра сигнала длиной 256 отсчетов с помощью приведенной софт-архитектуры решается за два кадра.

Перед первым кадром софт-архитектура настраивается на выполнение 1, 2, 3, 4 итераций БПФ. При этом промежуточные данные записываются в контроллер распределенной памяти с номерами 5, 6, 7, 8. После этого архитектура перестраивается на выполнение 5, 6, 7, 8 итераций алгоритма БПФ, а результирующие данные записываются в контроллер распределенной памяти с номерами 1, 2, 3, 4.

В соответствии с алгоритмом создания софт-архитектур, после того как архитектура сформирована и оттранслирована в промежуточное представление, она может быть размещена на вычислительном ресурсе РВС. Эту функцию выполняет синтезатор масштабируемых параллельно-конвейерных процедур на уровне логических ячеек ПЛИС Fire!Constructor. В качестве входных данных синтезатор использует: библиотечные файлы элементов софт-архитектуры, описание софт-архитектуры, описание аппаратной платформы РВС. Описания вычислительных модулей РВС включают информацию о числе и типах ПЛИС, а также о существующих физических связях между кристаллами ПЛИС вычислительного модуля.

В результате отображения софт-архитектуры на аппаратный ресурс РВС синтезатором Fire!Constructor осуществляется разбиение иерархичного информационного графа софт-архитектуры на подграфы в соответствии с описанием структуры РВС и выполняется трассировка информационных связей между кристаллами ПЛИС. Для каждого кристалла ПЛИС генерируется описание на языке VHDL, соответствующее фрагменту софт-архитектуры, представленному подграфом информационного графа, размещенного в данном кристалле ПЛИС. Кроме того, для каждой ПЛИС формируется UCF-описание, которое содержит назначение входных/выходных связей фрагмента софт-архитектуры с внешними выводами ПЛИС и дополнительную информацию, необходимую для синтеза конфигурационных файлов ПЛИС. Конфигурационные файлы ПЛИС формируются с помощью средств разработки схемотехнических решений, предоставляемых фирмами-изготовителями ПЛИС, или аналогичным программным обеспечением.

В качестве аппаратной платформы для размещения софт-архитектуры SA_DSP был использован базовый модуль 24V7-750, имеющий следующие технические характеристики:

- число ПЛИС Virtex-7 (58,5 млн вентилей) — 6 шт;
- число элементарных процессоров — 1350 шт.;
- производительность — 720 (340) Гфлопс;
- частота работы платы — 400 МГц;
- частота информационных обменов — 1200 МГц.

Этап получения конфигурационных файлов ПЛИС является последним в алгоритме создания софт-архитектуры. В результате загрузки конфигурационных файлов в РВС на базе реконфигурируемого вычислителя формируется архитектура специализированного вычислительного устройства, соответствующего разрабатываемой софт-архитектуре, на котором могут решаться различные параллельные прикладные задачи без перезагрузки конфигурационных файлов и без необходимости многократной трансляции на уровень логических ячеек ПЛИС при отладке программы.

Решение прикладных задач с использованием софт-архитектур

Процесс решения задач с использованием софт-архитектур начинается с разработки параллельной программы на языке COLAMO. Основная часть программы для вычисления по алгоритму БПФ с длинной сигнала 256 отсчетов представляет собой два кадра. Первый кадр настраивает архитектуру на выполнение 1...4 итераций БПФ, второй кадр настраивает архитектуру на вычисление 5...8 итераций. В качестве примера ниже приведено описание первого кадра программы.

```
ProgramTEST_FFT;
Include V7_Pleyada.Pleayda, V7_SADL.LibMacroObject_SA_Int;
Architecture SA_DSP;
Var DPRam0A, DPRam0B, DPRam0C, DPRam0D : Array Integer
[100 : Stream] mem;
Var DPRam5A, DPRam5B, DPRam5C, DPRam5D : Array Integer
[100 : Stream] mem;
Var DPRam1A, DPRam1B, DPRam1C, DPRam1D : integer com;
Var DPRam2A, DPRam2B, DPRam2C, DPRam2D : integer com;
Var DPRam3A, DPRam3B, DPRam3C, DPRam3D : integer com;
Var DPRam4A, DPRam4B, DPRam4C, DPRam4D : integer com;
Var ReA0, ImA0, ReB0, ImB0 : integer com;
Var ReA1, ImA1, ReB1, ImB1 : integer com;
Var ReA2, ImA2, ReB2, ImB2 : integer com;
Var ReA3, ImA3, ReB3, ImB3 : integer com;
Var ReA4, ImA4, ReB4, ImB4 : integer com;
Vari :number;
Var co1 = $141;Var co2 = $242; Var co3 = $343; Var co4 = $444;
Var coM1 = $0; Var coM2 = $1; Var coM3 = $2; Var coM4 = $3;
Cadr Cadr1;
for i:=0 to 63 do
    Begin
ReA0 := DPRam0A[i];
ImA0 := DPRam0B[i];
ReB0 := DPRam0C[i];
ImB0 := DPRam0D[i];
BONode(ReA0, ImA0, ReB0, ImB0, coM1, ReA1, ImA1, ReB1,
ImB1);
DPMem1(ReA1, ImA1, ReB1, ImB1, co1, DPRam1A, DPRam1B,
DPRam1C, DPRam1D);
BONode(DPRam1A, DPRam1B, DPRam1C, DPRam1D, coM2,
ReA2, ImA2, ReB2, ImB2);
DPMem2(ReA2, ImA2, ReB2, ImB2, co2, DPRam2A, DPRam2B,
DPRam2C, DPRam2D);
BONode(DPRam2A, DPRam2B, DPRam2C, DPRam2D, coM3,
ReA3, ImA3, ReB3, ImB3);
DPMem3(ReA3, ImA3, ReB3, ImB3, co3, DPRam3A, DPRam3B,
DPRam3C, DPRam3D);
BONode(DPRam3A, DPRam3B, DPRam3C, DPRam3D, coM4,
ReA4, ImA4, ReB4, ImB4);
DPMem4(ReA4, ImA4, ReB4, ImB4, co4, DPRam4A, DPRam4B,
DPRam4C, DPRam4D);
        DPRam5A[i] := DPRam4A;
        DPRam5B[i] := DPRam4B;
        DPRam5C[i] := DPRam4C;
        DPRam5D[i] := DPRam4D;
    End;
EndCadr;
```

Константы co1, .., co4 настраивают память коэффициентов БПФ, а константы coM1,..., coM4 настраивают работу синхронизирующей памяти в соответствии с алгоритмом решения задачи.

Результатом работы транслятора COLAMO являются:

- объектная модель вычислительного графа задачи, передаваемая в синтезатор Steam!Constructor;
- проект параллельно-конвейерной программы на языке Argus и конфигурационный файл, содержащий данные о настройке архитектуры на решение текущей задачи;
- управляющая программа на языке высокого уровня VisualC#, организующая обмен данными между персональным компьютером и РВС.

Объектная модель вычислительного графа задачи представляет собой XML-файл с описанием структуры каждого кадра многокадровой прикладной программы. Описание структуры кадра содержит информацию о так называемом плоском графе кадра: об информационных и операционных вершинах; о комплексных вершинах, которые соответствуют некоторому блоку элементарных операций; об информационных связях между всеми описанными вершинами "плоского" графа кадра. Кроме того, XML-описание также содержит информацию о последовательности выполнения кадров в программе, информацию о предварительной загрузке параметров для каждого кадра (в случае, если такая загрузка необходима) и другие параметры прикладной программы, необходимые для ее отображения на архитектуру РВС.

На основании полученных данных синтезатор Steam!Constructor выполняет автоматическое отображение информационных графов многокадровых прикладных программ на софт-архитектуру РВС. В процессе отображения прикладной программы на софт-архитектуру РВС выполняются: одновременное разбиение информационного графа каждого из кадров на подграфы согласно модифицированному описанию софт-архитектуры РВС, размещение данных подграфов в ПЛИС РВС и трассировка информационных каналов внутри каждой ПЛИС и между кристаллами ПЛИС согласно информационным связям текущего кадра. Для каждого кристалла ПЛИС формируется XML-структура, содержащая параметры, описывающие функционирование внутренних объектов данной ПЛИС, задействованных при отображении информационных графов. Аналогичные XML-структуры формируются и для входов/выходов кристаллов ПЛИС, задействованных при формировании информационных каналов. Полученные XML-структуры составляют файл результатов отображения, который передается синтезатором Steam!Constructor транслятору COLAMO для дальнейшей обработки. На основании переданного файла с результатами отображения транслятором COLAMO формируются загрузочный out-файл и управляющая программа. Весь процесс трансляции параллельной программы на софт-архитектуре занимает не более десяти минут, что существенно быстрее, чем при трансляции программы на уровень логических ячеек ПЛИС.

Заключение

Практическое использование комплекса программного обеспечения при создании и модификации софт-архитектуры SA_DSP позволило в два раза ускорить процесс отладки прикладных программ за счет уменьшения времени трансляции. Включение в состав программного обеспечения транслятора языка SADL позволило достаточно быстро разрабатывать и модифицировать пользо-

вательские софт-архитектуры. Использование синтезатора Fire!Conatructor дало возможность автоматического размещения софт-архитектуры на аппаратной платформе и избавило пользователя от необходимости детального изучения аппаратного ресурса РВС. При решении прикладных задач с использованием РВС транслятор COLAMO и синтезатор Water!Constructor автоматически и за короткое время (до 10 мин) преобразуют текст параллельной программы в набор кодов управления софт-архитектуры, что позволяет прикладному программисту получить результат за более короткий срок по сравнению с существующими подходами к программированию РВС.

Список литературы

1. **Gudkov V. A., Gulenok A. A., Kovalenko V. B., Slasten L. M.** Preprints of the 12th IFAC Conference on Programmable Devices and Embedded Systems PDES — 2013, Technical University of Ostrava, Czech Republic. 2013. P. 65—70.

2. **Каляев И. А., Левин И. И., Семерников Е. А., Шмойлов В. И.** Реконфигурируемые мультиконвейерные вычислительные структуры / Под общ. ред. И. А. Каляева. 2-е изд., перераб. и доп. Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. 344 с.

3. **Левин И. И.** Многопроцессорная система с программированием архитектуры на нескольких уровнях // Тр. Первой Всероссийской науч. конф. "Методы и средства обработки информации". М.: Изд. МГУ. 2003. С. 111—118.

4. **Семерников Е. А., Дмитренко Н. Н., Каляев И. А., Левин И. И.** Семейство многопроцессорных вычислительных систем с динамически перестраиваемой архитектурой // Вестник компьютерных и информационных технологий. 2009. Ч. 1. № 6. С. 2—8. Ч. 2. № 7. С. 2—10.

5. **Коваленко В. Б., Семерников Е. А.** Организация многоуровневого программирования реконфигурируемых вычислительных систем // Вестник компьютерных и информационных технологий. 2011. № 9. С. 3--10.

6. **Левин И. И., Дордопуло А. И., Сластен Л. М., Гудков В. А.** Средства программирования реконфигурируемых многопроцессорных вычислительных систем // Известия ТРТУ. 2006. № 16. Специальный выпуск. С. 16--20.

V. B. Kovalenko[1], Senior Researcher, e-mail: vereten@hotbox,
A. I. Dordopulo[2], Head of Laboratory, e-mail: scorpio@mvs. tsure.ru,
V. A. Gudkov[2], Senior Researcher, e-mail: gudkov@mvs.tsure.ru,
A. A. Gulenok[2], Senior Researcher, e-mail: andrei-gulenok@mail.ru,
L. M. Slasten[2], Senior Researcher, e-mail: emslasten@yandex.ru
[1] Southern Scientific Centre of the Russian Academy of Sciences
[2] Academical A. V. Kalyaev Scientific Research Institute of Miltiprocessor Computer at Southern Federal University

Use of Soft-Architectures for Digital Signal Processing Tasks Solved on Reconfigurable Computer Systems

The paper covers the macroobject approach to reconfigurable computer system (RCS) programming, which reduces the programming time of tasks owing to reduction of the translation time of parallel programs. The description of the main advantages and shortcomings of the existing RCS programming levels are given. We have analysed the notion of a soft-architecture, a macroobject, a node and objects of reconfigurable computer systems. We have given the structure of system software owing to which the suggested approach to RCS programming is provided. In addition we have described functions of the elements of the software suit and their interconnection during development of soft-architectures, and use of soft-architectures for implementation of application tasks. The development of the soft-architecture for digital signal processing tasks is analysed, and examples of SADL-descriptions of soft-architecture elements are given. The example of the soft-architecture used for implementation of digital signal processing tasks is given. Besides, we have given the description of one cadr of the parallel application, written in the high-level programming language COLAMO. It implements the FFT algorithm on the base of the DSP soft-architecture.

Keywords: *multiprocessor systems, supercomputers, reconfigurable computer systems, architectures of computer systems, parallel programming, programming languages*

References

1. **Gudkov V. A., Gulenok A. A., Kovalenko V. B., Slasten L. M.** Preprints of the 12th IFAC Conference on Programmable Devices and Embedded Systems PDES — 2013. Technical University of Ostrava, Czech Republic. 2013. P. 65—70.

2. **Kalyaev I. A., Levin I. I., Semernikov E. A., Shmoilov V. I.** Rekonfiguriruemiye multikonveyemiye vychislitelniye struktury. Pod obtsh. red. I. A. Kalyaev. 2nd-e izd., pererab i dop. Rostov-na-Donu: Izd-vo YuNC RAN, 2009. 344 p.

3. **Levin I. I.** Mnogoprotsessomaya systema s programmirovaniyem arkhitektury na neskolkykh urovnyakh. *Tr. Pervoi Vserossiiskoy nauch. konf. "Metody i sredstva obrabotki informatsii"*. M.: MGU, 2003. P. 111—118.

4. **Semernikov E. A., Dmitrenko N. N., Kalyaev I. A., Levin I. I.** Semeistvo mnogoprotsessornykh vychislitelnykh system s dinamicheski perestraiyayemoi arkhitekturoi. *Vestnik kompiuternykh i informatsionnykh tekhnologiy*. 2009. Ch. 1. N. 6. P. 2—8. Ch. 2. P. 2—10.

5. **Kovalenko V. B., Semernikov E. A.** Organizatsiya mnogourovnevogo programmirovaniya rekonfiguriruemykh vychislitelnykh system. *Vestnik kompiuternykh i informatsionnykh tekhnologiy*. 2011. N. 9. P. 3—10.

6. **Levin I. I., Dordopulo A. I., Slasten L. M., Gudkov V. A.** Sredstva programmirovaniya rekonfiguriruemykh mnogoprotsessornykh vychislitelnykh system. *Izv. TRTU*. 2006. N. 16. Spec. vyp. P. 16—20.