РАСШИРЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ COLAMO ДЛЯ БИТОВОЙ ОБРАБОТКИ

И.И. Левин, А.И. Дордопуло, В.А. Гудков, А.А. Гуленок

Непрерывный поиск новых способов достижения высокой производительности при решении задач различных классов, в первую очередь сильносвязанных задач, обусловил применение суперЭВМ на основе программируемых логических интегральных схем (ПЛИС) для выполнения трудно- или неэффективно реализуемых фрагментов вычислений. С расширением класса эффективно решаемых на РВС задач требуется своевременная модернизация средств программирования РВС.

В НИИ МВС ЮФУ программирование реконфигурируемых вычислительных систем осуществляется на языке высокого уровня COLAMO (Common Oriented Language for Architecture of Multi Objects), разработанным И.И. Левиным в 1987 году [1]. Язык COLAMO предназначен для описания реализации специализированной вычислительной структуры в архитектуре РВС на основе принципов структурно-процедурной организации вычислений [2], которая предполагает последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов.

В 2010 году для поддержки задач символьной обработки данных в язык высокого уровня COLAMO были введены следующие конструкции: тип данных Bit, операторы логических функций (and, or, xor, not) и логические поразрядные операции (операции сдвига влево и вправо, операции циклического сдвига влево и вправо и др.), функции преобразования данных: Separate и Combine, а также вычислительные структуры для структурной и процедурной реализации вычислений LocalProc и Implicit [3].

На рис. 1 рассмотрен пример использования введенных в 2010 году расширений языка (битовых массивов, функций Separate и Combine) в программе, а также граф синтезированной вычислительной структуры.

```
Const n = 1;
Const k = 32;
Var a, b : Array Integer [100: Stream] Mem;
Var d, e : Array Bit [k : Vector, 100: Stream] Com;
Var i : Number;
Cadr BitUse;
  For i := 0 to 99 do
    Begin
      d[*,i] := Separate (a[i]);
      e[*,i] := d[*,i] << 3;
      b[i] := Combine(e[*,i]);
    End;
EndCadr;
```

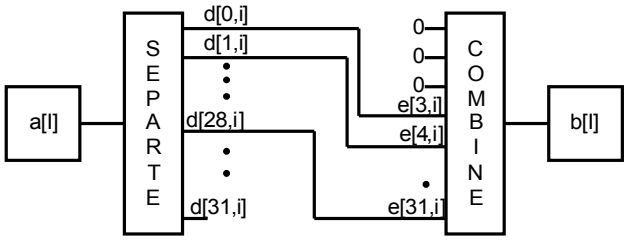

Рис. 1. Пример программы и граф синтезированной вычислительной структуры

В рассмотренной на рис. 1 программе функция Separate выполняет отображение битового представления элемента a[i] в векторную компоненту битового массива d, обозначенного срезом [3]. Функция Combine объединяет соответствующие срезу элементы битового массива в единое 32-разрядное слово, соответствующее элементу b[i].

Введение в язык битовых массивов и операций над ними, с одной стороны, расширило возможности программиста, поскольку позволило оперировать массивами бит, а с другой стороны, обязывало его использовать только 32-разрядные слова для хранения битовых массивов, что связано с аппаратной реализацией существующих блоков распределенной памяти.

Однако опыт практического решения задач цифровой обработки сигналов и задач символьной обработки показал, что для успешного решения задач такого класса на РВС требуется создание более мощных инструментов в языке программирования COLAMO. Поэтому для повышения функциональности и эффективности языка COLAMO при решении задач цифровой обработки сигналов в 2013 году было разработано расширение языка, включающее следующие типы переменных и конструкции: виртуальные переменные, оператор выделения бит (FlexBit), поддержка внешних констант (ExternalConst) в подкадрах.

Для расширения возможностей использования битовых массивов в язык COLAMO введен новый тип переменной – Virtual. Переменная типа Virtual является виртуальной переменной, и в отличие от статических типов данных инициализация виртуальной переменной происходит только один раз при первом присваивании ей данных, и далее при объявлении переменной ее разрядность, знак и формат не переопределяются.

Рассмотрим программу, содержащую различные варианты инициализации параметров виртуальной переменной.

```
Var Virt1, Virt2, Virt3, Virt3 : Virtual Com;
Var a : UInteger Mem;
Const b = $12AB;
Const c = (Double) $12AB;
Cadr Cadr1
   Virt1 = 10;
   Virt2 = a;
   Virt3 = b;
   Virt4 = c;
EndCadr;
```

В таблице 1 представлены определенные транслятором языка COLAMO параметры виртуальных переменных рассмотренной программы.

Таблица 1 - Параметры виртуальных переменных

Переменная	Разрядность	Знак	Формат
Virt1	32	Знаковый	Фиксированная точка
Virt2	32	Беззнаковый	Фиксированная точка
Virt3	32	Знаковый	Фиксированная точка
Virt4	64	Знаковый	Плавающая запятая

Для задания произвольной разрядности виртуальной переменой в языке COLAMO используется оператор FlexBit.

Оператор FlexBit позволяет пользователю изменять разрядность переменной путем выделения необходимо числа бит, начиная с указанной позиции. В общем виде использование оператора FlexBit выглядит следующим образом:

A = FlexBit (B, K, N),

где A – результат выполнения функции, B – обрабатываемое данное, K – начальный номер бита, а N – количество выделяемых бит, начиная с позиции K.

Разрядность переменной B и содержимое переменной B зависит от параметров K и N. Для простоты рассмотрения будем считать, что разрядность переменной B равна значению R.

Если K < 0 или N < 0 или K >= R, то параметры функции FlexBit являются некорректными и такое обращение к функции воспринимается транслятором COLAMO как семантическая ошибка с выдачей сообщения об ошибке.

Если N < R и N + K <= R, то в переменной A начиная с нулевого разряда по разряд N - 1 включительно, будут располагаться биты соответствующие битам переменной B начиная с бита с номером K по бит с номером K + N включительно.

Если N < R и N + K > R, то в переменной A начиная с нулевого разряда по разряд R – K – 1 включительно будут располагаться биты соответствующие битам переменной B начиная с бита с номером K по бит с номером R включительно, а начиная с разряда R – K до разряда N - 1 будут расположены биты со значением – ноль.

Если N > R, то в переменной A, начиная с разряда K по разряд R + K включительно, будут располагаться биты, соответствующие битам переменной B, а начиная с нулевого разряда по разряд K и начиная с разряда R + K + 1 по разряд N - 1, будут расположены биты со значением – ноль.

Рассмотрим на примерах различные варианты значений параметров K и N оператора FlexBit и результаты его работы.

На рис. 2 показана параллельная программа и схема выделения бит оператором FlexBit при K= 3 и N=5.

```
Var A : Virtual Com;
Var B, C : Integer Mem;
Cadr Cadr1
    A = FlexBit(B, 3, 5);
EndCadr;
```

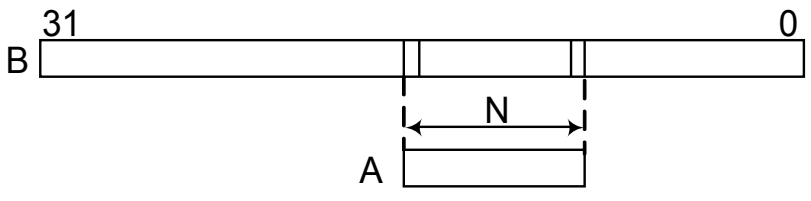

Рис. 2. Схема выделения бит оператором FlexBit при K= 3 и N=5

После трансляции данной программы функция FlexBit выделит из переменной B 5-разрядное данное, начиная с 3-его бита.

На рис. 3 показаны параллельная программа и схема выделения бит оператором FlexBit при K= 27 и N=10.

```
Var A : Virtual Com;
Var B, C : Integer Mem;
Cadr Cadr1
        A = FlexBit(B, 27,
10);
EndCadr;
```

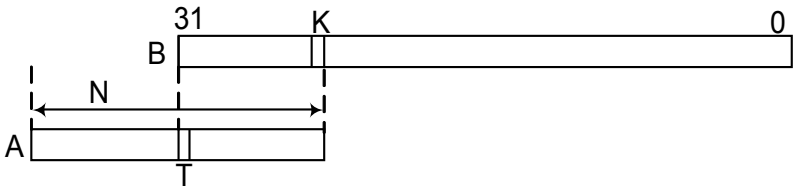

Рис. 3. Схема выделения бит оператором FlexBit, при K= 27 и N=10

В рассмотренной программе, в отличие от программы на рис. 3, сумма значений параметров K и N превышает разрядность переменной B (32 разряда), при этом значение N меньше разрядности переменной B. В данном случае после выполнения оператора присваивания переменная A будет иметь разрядность 10, где первые 5 бит будут соответствовать битам с номерами 27 - 31 переменной B, а биты, начиная с 5 по 9, будут иметь значение – ноль.

Рассмотренные примеры использования оператора FlexBit на рис. 2 и рис. 3 продемонстрировали случаи понижения разрядности путем выделения необходимого числа бит, начиная с начальной позиции, т.е. N<R, в противном случае оператор FlexBit будет выполнять расширение разрядности, как показано на рис. 4.

```
Var A : Virtual Com;
Var B, C : Integer Mem;
Cadr Cadr1
    A = FlexBit(B, 5, 45);
EndCadr;
```

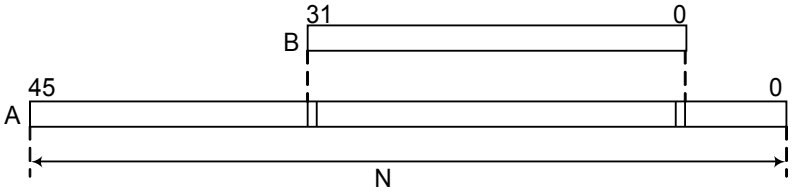

Рис. 4. Схема расширения разрядности оператором FlexBit, при K= 5 и N=45

В рассмотренной программе, переменная A будет иметь разрядность, равную 45, где биты, начиная с бита с номером 5 по 36, будут соответствовать битам переменной B, а биты, начиная с 0 по 5 и с 37 по 45, будут иметь значение ноль.

При расширении разрядности значение K определяет начальное положение всех бит переменной B в переменной A, при этом начиная с нулевого бита по бит K-1 и начиная с бита K+P+1 по N включительно будут расположены ноли.

При использовании функции Combine и FlexBit для виртуальных переменных необходимо учитывать тот факт, что для полученного слова разрядности N не определены знак и формат, что может привести к генерации некорректной вычислительной структуры.

Для приведения виртуальной переменной к знаковой или беззнаковой форме представления в язык вводятся функции SetSign и SetUnSign, а для указания формата данных - функции SetInt и SetFloat. Функция SetSign указывает на хранение в виртуальной переменной знаковых данных, функция SetUnSign – беззнаковых. Использование функции SetInt указывает на хранение в виртуальной переменной данных с фиксированной точкой, а функция SetFloat – данных с плавающей запятой. Функции непосредственного указания знака и формата данных, хранимых в виртуальной переменной, могут расцениваться транслятором как директивы и не участвовать в формировании вычислительной структуры в явном виде.

Введение в язык 2010 года макросов, в частности, макросов, реализующих операции сдвига, позволило расширить функциональность языка высокого уровня COLAMO, но опыт программирования на языке COLAMO показал неэффективность использования макросов, реализующих операции сдвига в подкадрах, в силу особенностей их реализации. В отличие от традиционных языков программирования операции сдвига реализуются в виде фиксированной структуры на этапе трансляции параллельной программы, что требует от программиста указания величины сдвига в виде константы.

Данное требование приводит к неудобству при программировании и увеличению исходного кода параллельной программы при необходимости использования одной и той же операции сдвига на разную величину при вызове одного и того же подкадра.

Подобная проблема возникает и в случае, когда необходимо использовать один и тот же подкадр, но с разными параметрами внутри, например, изменять параметр распараллеливания фрагмента программы при описании операторов цикла. В существующем синтаксисе языка подобное действие не выполнимо и требует для каждого параметра описывать новую конструкцию «подкадр».

Для решения рассмотренных проблем в язык COLAMO вводится новый тип переменной ExternalConst. Переменная типа ExternalConst ничем не отличается от переменной типа «Const» с точки зрения синтаксиса языка, но отличается семантикой.

Использование переменной типа ExternalConst в объявлении входов подкадров или конструкции LET требует генерации новой вычислительной структуры и выполнения автоподстановки ее фактического значения,

указанного при вызове вычислительной структуры во все операторы в теле обрабатываемой структуры где выполняется обращение к ней.

На рис. 5 в левой части представлена программа, демонстрирующая использование переменной типа ExternalConst, а в правой части - параллельная программа после обработки данной переменой.

Программа Constant1 содержит описание подкадра Sub1, в параметрах которого описана переменная Con, имеющая тип ExternalConst. Наличие данного параметра в подкадре указывает на необходимость обработки данного подкадра. Обработка подкадра Sub1 заключается в разборе фактических параметров, используемых при вызове данного подкадра и создании нового подкадра Sub2 (см. программу Constant2), операторами которого являются модернизированные операторы рассматриваемого подкадра Sub1. После создания нового подкадра Sub2 параметр Con удаляется из списка параметров подкадра, а вместо вызова подкадра Sub1 используется подкадр Sub2.

```
Program Constant1;
Var a, b, c : Array Integer [10 : Stream] Mem;
Var i : Number;

SubCadr Sub1 (in : a1, b1, Con; out : c1);
Var a1, b1, c1 : Integer Com;
Var Con : ExternalConst;
c1 := (a1 + b1) * Con;
EndSubCadr;

Cadr BeginEnd1;
   For i := 0 to 9 do
      Begin
         Sub1(a[i], b[i], 3,c[i]);
      End;
EndCadr;
EndProgram.
```

```
Program Constant2;
Var a, b, c : Array Integer [10 : Stream] Mem;
Var i : Number;

SubCadr Sub2 (in : a1, b1; out : c1);
Var a1, b1, c1 : Integer Com;
c1 := (a1 + b1) * 3;
EndSubCadr;

Cadr BeginEnd1;
   For i := 0 to 9 do
      Begin
         Sub2(a[i], b[i], c[i]);
      End;
EndCadr;
EndProgram.
```

Рис. 5. Выполнение этапа модификации констант

Такой подход позволяет использовать в языке одну и ту же конструкцию, содержимое которой будет зависеть от фактических параметров при ее вызове, что значительно упрощает разработку параллельных прикладных программ.

Описанные в настоящей работе новые типы переменных и операторы языка высокого уровня реализованы в трансляторе языка COLAMO версии 2.1 и были апробированы на задаче цифровой обработки радиолокационных данных. Реализация задачи с использованием представленных в настоящем докладе расширений языка привела к сокращению объема исходного текста параллельной программы на 20%, что улучшило читабельность программы и значительно упростило отладку параллельной прикладной программы.

ЛИТЕРАТУРА:

1. И.А. Каляев, И.И. Левин, Е.А. Семерников, В.И. Шмойлов Реконфигурируемые мультиконвейерные вычислительные структуры /Изд. 2-е, перераб. и доп. / Под общ. ред. И.А. Каляева. - Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
2. А.В. Каляев, И.И. Левин Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. - М.: Янус-К, 2003. – 380 с.
3. И.И. Левин, В.А. Гудков Расширение языка высокого уровня COLAMO для программирования реконфигурируемых вычислительных систем на уровне логических ячеек ПЛИС // Вестник компьютерных и информационных технологий. – М.: Машиностроение, 2010. - №12. – С.10-17.