Суперкомпьютерные дни в России 2019

Комплекс средств программирования реконфигурируемых вычислительных систем на основе ПЛИС

А.И. Дордопуло,

НИЦ СЭ и НК

И.И. Левин, В.А. Гудков, А.А. Гуленок,
А.В. Бовкун, Г.А. Евстафьев, К.Н. Алексеев

Компилятор программ на языке С в программы на COLAMO с последующей трансляцией в VHDL

Преобразовать последовательную программу на языке программирования С (стандарт ISO/IEC 9899:1999 для компилятора gcc) в информационный граф, описанный на языке программирования COLAMO, который транслятором COLAMO транслируется в конфигурационные файлы ПЛИС для заданного вычислительного ресурса.

Выходные данные: конфигурационные файлы кристаллов ПЛИС с параллельно-конвейерной реализацией алгоритма программы на специализированном вычислителе потоковой архитектуры.

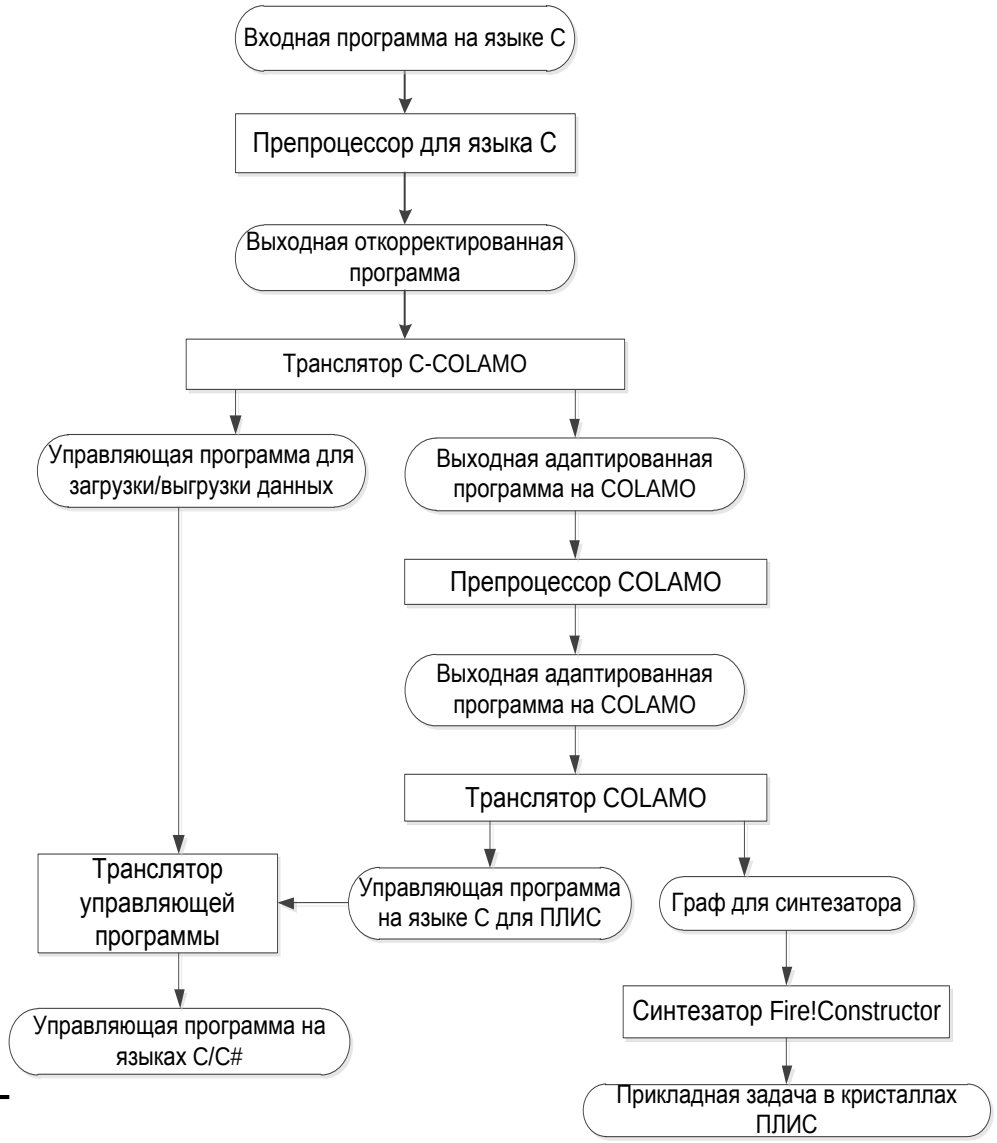
Заданные показатели: удельная производительность получаемых решений не менее 70% при уровне заполнения кристалла не выше 90%.

Аналоги: Vivado HLS, Quartus HLS, Mitrion-C, Catapult-C

Отличия: создание решения задачи на выбранной РВС, а не IP-ядра

Общая схема предлагаемого решения

1. Преобразование входной программы к каноническому виду
2. Построение абсолютно-параллельного информационного графа входной программы, отображение его в синтаксис COLAMO.
3. Преобразование абсолютно-параллельного информационного графа входной программы к параллельной, параллельно-конвейерной или конвейерной форме.
4. Адаптация программы к вычислительному ресурсу.
5. Трансляция программы в VHDL и синтез конфигурационных файлов.



Структура программного комплекса

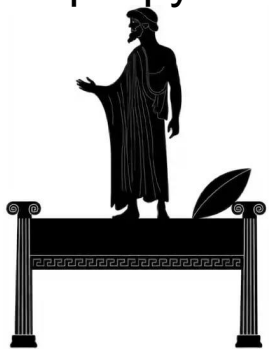
Транслятор «Ангел»

Преобразует последовательную программу на C в информационный граф на COLAMO

Процессор «Русалка»

Преобразует абсолютно параллельную программу на COLAMO в параллельно-конвейерную ресурсонезависимую форму

Процессор «Прокруст»



Автоматически подбирает параметры параллельно-конвейерной программы на COLAMO для заданного вычислительного ресурса и преобразует программу для эффективной реализации

Процессор «Щелкунчик»

Редуцирует программу при нехватке аппаратного ресурса для ее эффективной реализации на имеющемся

Транслятор языка COLAMO + синтезатор Fire!Constructor + описание архитектуры РВС

Синтезатор загрузочных конфигурационных файлов Xilinx Vivado/ISE

РВС: загрузочные конфигурационные файлы ПЛИС

Транслятор «Ангел»

1. Преобразует входную программу на языке С к каноническому виду
2. Строит абсолютно-параллельный информационный граф входной программы и отображает его в синтаксис языка COLAMO.

Состоит из 2 модулей: препроцессора языка С и транслятора С-COLAMO.

Модуль препроцессора языка С:

- проверяет синтаксическую правильность входной программы на С;
- раскрывает директивы условной компиляции и макросы, преобразует трехместные операции к двуместным, преобразует SWITCH и CASE к оператору IF, приводит все циклы к оператору FOR, преобразует доступ по указателям в адресацию массивов;
- проверяет синтаксическую правильность преобразованной программы на С.

Транслятор «Ангел»

Модуль транслятора C-COLAMO:

- выполняет анализ информационных зависимостей переменных и массивов исходной последовательной программы;
- на основе результатов анализа строит абсолютно-параллельный информационный граф;
- преобразует последовательную программу с произвольным обращением к памяти к потоку данных;
- определяет характерные для последовательных программ нарушения правил однократного присваивания и единственной подстановки;
- устраняет нарушения правил в тексте программы на COLAMO (расщепляет скалярные переменные и растягивает массивы по итерациям с копированием необходимых данных);
- для каждой индексной переменной каждого массива определяет тип цикла (итерационный или слоевой) для всех вхождений переменной в тексте программы;

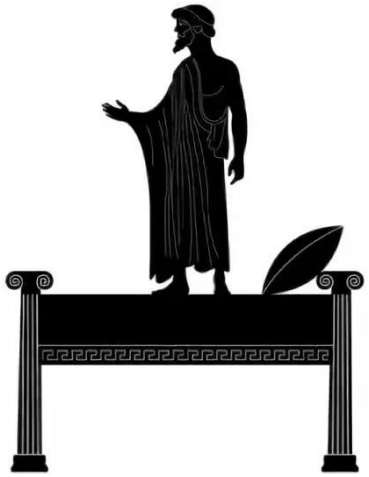
Выходные данные: абсолютно параллельная программа в синтаксисе COLAMO с большим числом каналов КРП.

Процессор «Русалка»

Преобразует абсолютно-параллельный информационный графа входной программы на COLAMO к параллельной, параллельно-конвейерной или конвейерной форме:

- выполняет разбиение каждой размерности массивов данных на параллельную (Vector) и потоковую (Stream) составляющую;
- все циклы в программе на COLAMO разбивает на параллельную (Vector) и потоковую (Stream) составляющую;
- выполняет вынос зависящих только от итераций переменных из слоевых циклов;
- определяет и преобразует базовый подграф вычислений в вычислительную структуру LET или подкадр (SubCadr);
- преобразует базовый подграф в структуру Implicit;
- вносит необходимые корректировки в текст программы на COLAMO с учетом сделанных изменений;

Выходные данные: параметризуемая по параллельной и потоковой составляющим параллельно-конвейерная программа на языке COLAMO, информационно-эквивалентная входной абсолютно-параллельной программе.



Процессор «Прокруст»

1. Автоматически рассчитывает параметры параллельно-конвейерной программы на COLAMO для заданного вычислительного ресурса;
2. Преобразует программу для эффективной реализации на заданном вычислительном ресурсе:

- выполняет подбор параметров ресурсонезависимой программы в зависимости от критического ресурса: время обработки, число КРП, доступный аппаратный ресурс;
- для рассчитанных параметров преобразует входную ресурсонезависимую программу к параллельной, конвейерной или параллельно-конвейерной форме;
- преобразует зависящие от итераций переменные к типам Intermem или Reg;
- выполняет организацию и синхронизацию обратных связей для реализуемой формы организации вычислений;
- синтезирует многокадровые программы для структур SubCadr и Let;
- организует вычисления по схеме «конвейер в конвейере».

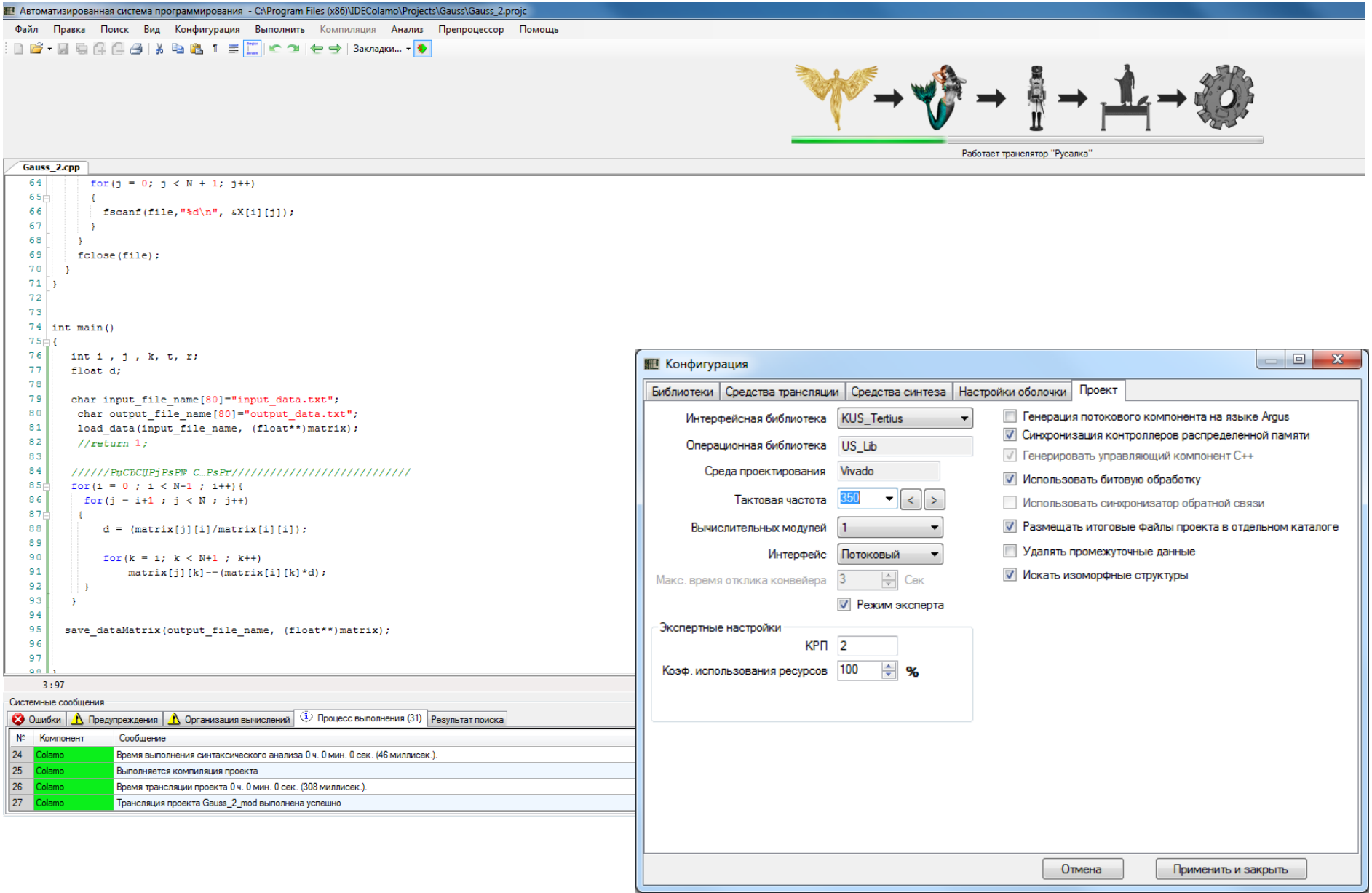
Процессор «Щелкунчик»

1. Редуцирует программу при нехватке аппаратного ресурса для реализации базового подграфа вычислений (если «Прокруст» не может подобрать параметры даже для минимального уровня параллелизма);
2. Преобразует программу на COLAMO для эффективной реализации на имеющемся вычислительном ресурсе:

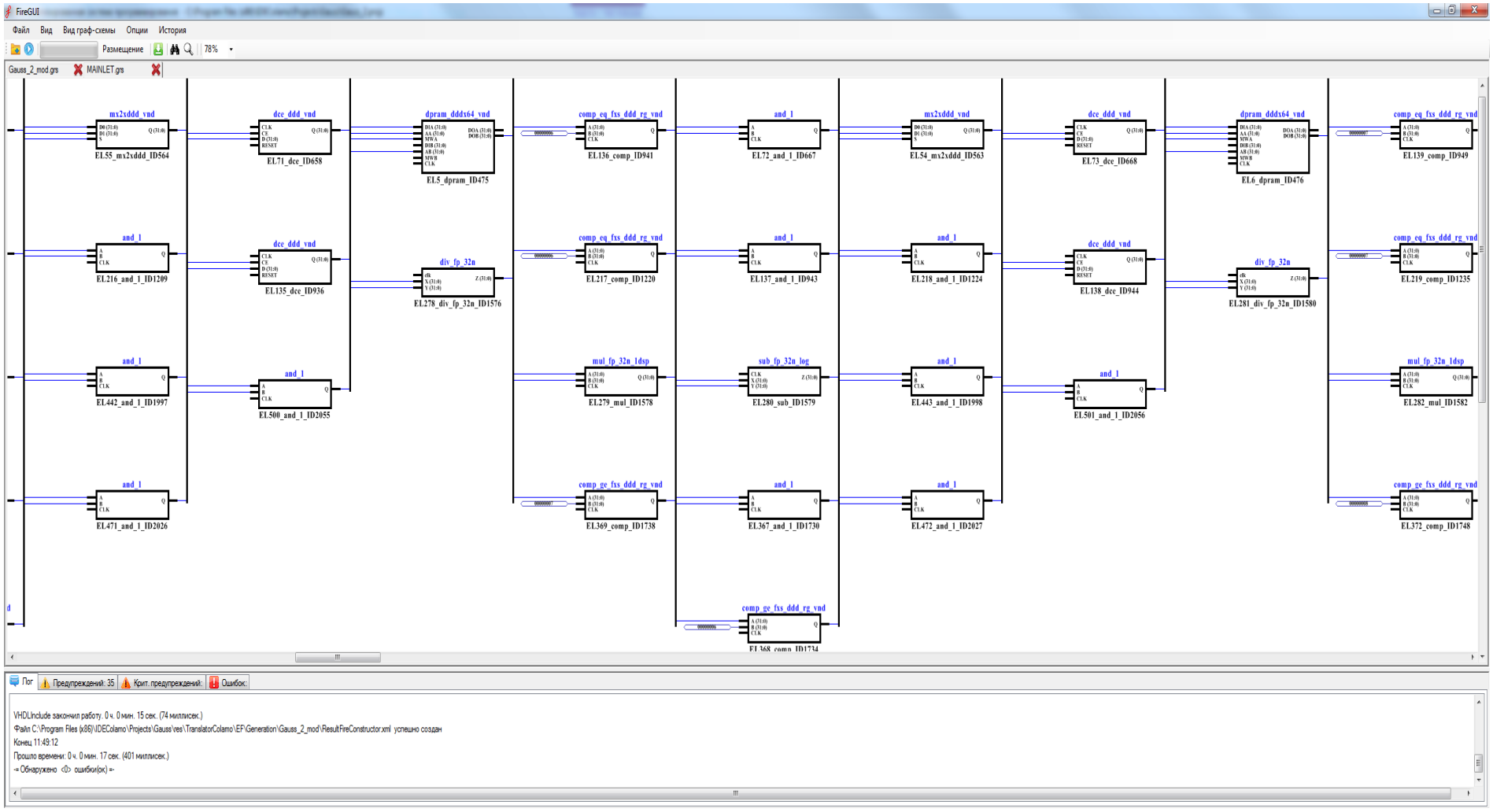
- рассчитывает требуемый коэффициент сокращения аппаратных затрат задачи;
- выполняет редукцию производительности базового подграфа по числу подграфов, вычислительных операций и разрядности;
- выбирает наиболее рациональный вариант реализации задачи по критерию общего времени обработки.
- преобразует текст программы на COLAMO.

Выходные данные: редуцированная программа в синтаксисе COLAMO, которая может быть выполнена на имеющемся вычислительном ресурсе.

Пример работы комплекса при трансляции задачи решения СЛАУ методом Гаусса

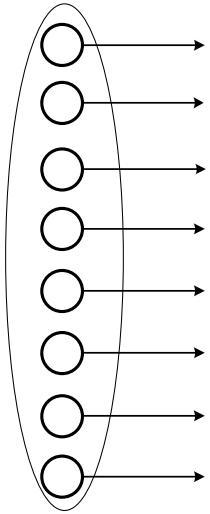


Пример работы комплекса при трансляции задачи решения СЛАУ методом Гаусса

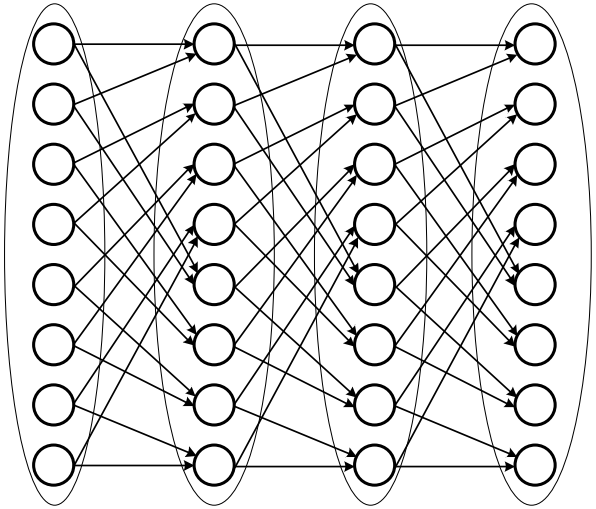


Виды прикладных задач

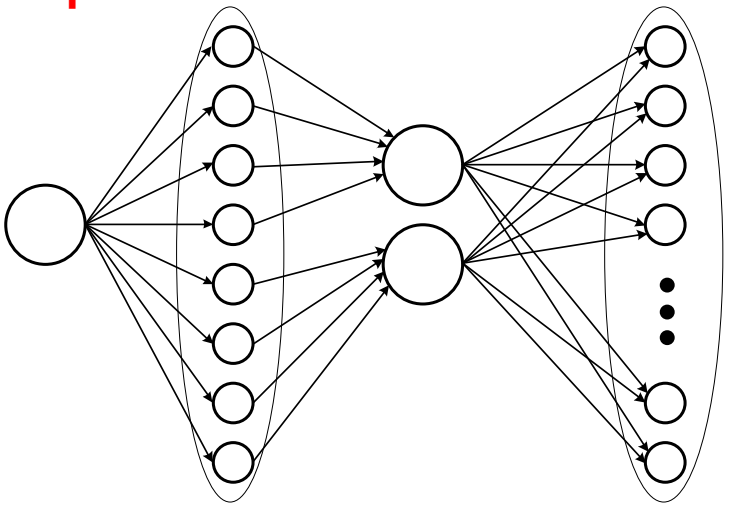
Линейные



Потоковые с постоянной интенсивностью



Потоковые с переменной интенсивностью



- **Линейные** - **символьная обработка**;
- **Потоковые** – **линейная алгебра** (решение СЛАУ методом Гаусса, решение СЛАУ методом Якоби для полных и 3-диагональных матриц, LU-разложение);
- С переменной интенсивностью потоков данных **- докинг ингибиторов**, молекулярное моделирование и др.

Сравнение реализации задачи символьной обработки

на COLAMO, Vivado HLS и транслятором C-COLAMO

Для микросхемы Xilinx Virtex 7 XC7VX485T (РВС «Тайгета»).

Частота 250 МГц. САПР Xilinx Vivado 2018.2

Наименование ресурса	Реализация программиста на COLAMO (двойной конвейер)	Автоматическая реализация программы С в Vivado HLS	Реализация программы С в трансляторе C-COLAMO	Эффективность C-Colamo	
				К COLAMO (один конвейер)	К Vivado HLS
Slice Registers (всего 607,200)	3 475	12 545	2 152	0,81	5,83
Slice LUTs (всего 303,600)	2 336	8 586	1 493	0,78	5,75
Occupied Slices (всего 75,900)	627	2 371	394	0,8	6,02
Максимальное число конвейеров в ПЛИС	**120*2=240**	30	**150/195** (130%)	0,81 (195/240)	6,5 (195/30)
Время трансляции проекта в Vivado	12 ч. 10 мин.	7 ч. 24 мин.	15 ч. 34 мин.		

Сравнение реализации задачи символьной обработки

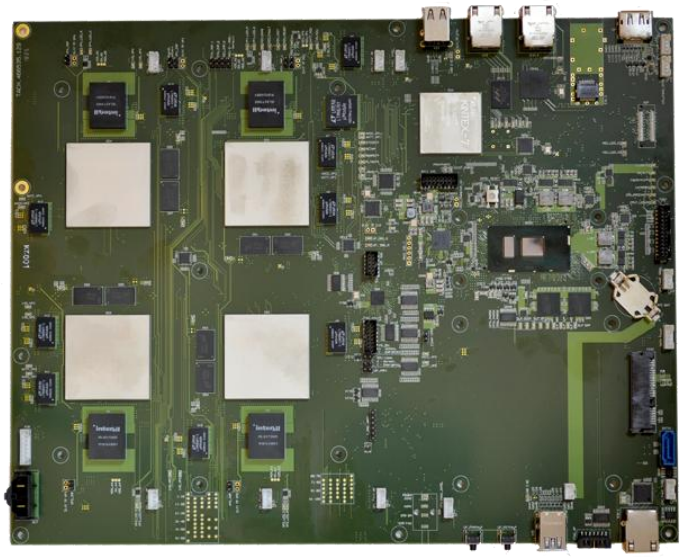
на COLAMO, Vivado HLS и транслятором C-COLAMO

Для микросхемы Xilinx Kintex UltraScale XCKU095 (РВС «Неккар»). Частота 400 МГц. САПР Xilinx Vivado 2018.2

Наименование ресурса	Реализация программиста на COLAMO	Реализация программы С в Vivado HLS	Автоматическая реализация программы С в трансляторе C-COLAMO	Эффективность C-Colamo	
				К COLAMO (один конвейер)	К Vivado HLS
CLB Registers (всего 1,075,200)	2 900	16 512	2 921	0,5	5,65
CLB LUTs (всего 537,600)	2 076	1 863	1 518	0,68	1,23
CLB (всего 67,200)	336	1 101	210	0,8	5,24
Максимальное число конвейеров в ПЛИС	**205*2=410**	60*	**266/313** (118%)	0,76 (313/410)	5,21 (313/60)
Время трансляции проекта в Vivado	6 ч. 11 мин.	6 ч. 21 мин.	6 ч. 28 мин.		

*в результате разводки 60 конвейеров, полученных с помощью Vivado HLS(плотное заполнение), САПР Xilinx Vivado HLS выдал 54037 временных ошибок. Для 50 конвейеров - 30743 временные ошибки.

Результаты реализации задач линейной алгебры на НРК «Терциус»



Микросхема Xilinx Virtex UltraScale XCKU095T

Частота 250 МГц.
САПР Vivado 2018.2

Алгоритм	Число ступеней на плате (4 ПЛИС)	
	Автоматический расчет (100% ресурса)	Vivado HLS
Гаусс 8000*8000	821	1*
LU 8000*8000	767	1*
Якоби 3-д 8000*8000	950	1*

Компилятор программ на языке С в программы на COLAMO с последующей трансляцией в VHDL

Основные отличия от существующих трансляторов C-VHDL и Vivado HLS:

1. Входной текст на «чистом» С в стандарте ISO/IEC 9899:1999.

2. Создание масштабируемой как по данным, так и по итерациям параллельно-конвейерной программы.

3. Автоматическая адаптация программы под доступный аппаратный ресурс.

4. Поддержка многокристалльных проектов.

5. Автоматический синтез управляющей программы для регистрового интерфейса.

Спасибо за внимание!

Если остались вопросы:

dordopulo@superevm.ru

Текущие требования к исходному тексту программ на языке С

1. Текст программы не должен содержать следующих операторов и конструкций языка С: операторов **goto**, конструкций **union** и **struct**, функций **strlen** и функций работы с памятью (**calloc**, **memcpy** и др).

2. В тексте программы должны обязательно присутствовать процедуры загрузки и сохранения данных с фиксированными именами (**load_data**, **load_data32**, **load_data64** и **load_data_str**) для определения входных и выходных переменных.

3. В циклах должны использоваться обычные индексные переменные без косвенной адресации:

Неправильно: for(**ind[LenP_]**=0 ; ind[LenP_] < AlphPower[LenP_] ; **ind[LenP_]**++)

Правильно: for(**i**=0 ; i < AlphPower[LenP_] ; **i**++)

4. По возможности избегать неоправданно частого использования операторов **break** и **return** для немедленного (аварийного) выхода из циклов.

Общая методика преобразования последовательной программы на С в параллельную программу на COLAMO

1 этап (Ангел): перевод программы на языке программирования С в абсолютно-параллельную форму на языке программирования COLAMO

1. Все инициализируемые на С массивы (для чтения или записи данных) становятся мемориальными массивами в COLAMO с параллельным типом доступа (векторными по всем измерениям):

int M [R][C]	Var M : Array Integer [R: Vector, C: Vector] Mem;

2. Для каждого мемориального массива в COLAMO объявляется коммутационный массив-копия для использования в базовом подграфе вычислений:

int M [R][C]	Var M : Array Integer [R: Vector, C: Vector] Mem;
	Var MCom : Array Integer [R:Vector,C:Vector] Com;

3. Коммутационные массивы-копии источников данных инициализируются значениями мемориальных массивов. В тексте программы все мемориальные массивы заменяются на коммутационные массивы-копии.

	MCom[*,*] := M [*,*];

Общая методика преобразования

2 этап (Ангел): анализ информационных зависимостей исходной программы

1. Поиск явной и неявной информационных зависимостей массивов данных и определение вида цикла (слоевой или итерационный) для каждой индексной переменной :

Явная	**Неявная** (псевдоитерационная)
`x[i] := f(x[i-k])`	`x[i] := f(x[i])` `a := f(a)`

2. Критерии определения вида цикла для индексной переменной.

2.1. По умолчанию все циклы абсолютно-параллельной программы считаются итерационными;

2.2. Цикл для каждого измерения массива считается слоевым, если :

- в массив, индексируемый переменной цикла, в теле цикла производится только запись (без операций чтения);
- если в массив, индексируемый переменной цикла, в теле цикла производится и чтение, и запись при неявно заданной информационной зависимости.

2.3. Если в массив, индексируемый переменной цикла, в теле цикла производится и чтение, и запись, при явно заданной информационной зависимости цикл остается итерационным.

Общая методика преобразования

2 этап(Ангел): расщепление скалярных переменных и растягивание массивов по итерациям

1. Растягивание коммутационного массива-копии по итерациям:

```
for i = 1 to n do
  for j = 1 to m do
    x[j] = a*x[j-1]+
           y[i];
```

```
Var xCom : Array Integer [n: Vector, n+1:
Vector] Com;
…
xCom[0,1] = x[0];
for i = 1 to n do
  begin
    for j = 1 to m do
       xCom[j,i] = a*xCom[j-1,i] + y[i];
 // переписывание данных на следующую
итерацию по i
     xCom[*, i+1] = xCom[*, i];
   end;
```

2. Разрешение правила однократного присваивания:

```
for i = 1 to n do
  for j = 1 to m do
    x[j] = a*x[j-1]+
           y[i];
```

```
xCom[*,1, 0] = x [*];
for i = 1 to n do
  begin
    for j = 1 to m do
      xCom[j,i,1]=a*xCom[j-1,i,0] + y[i];

    xCom[*,i+1,0] = xCom[*,i,1];
  end;
```

Общая методика преобразования

3 этап(Русалка): переход в параметризуемую параллельно-конвейерную форму

1. **Каждое** измерение всех мемориальных и коммутационных массивов разбивается на 2 составляющие – **векторную** (с параллельным) и **потоковую** (с последовательным) типами доступа. Длина потоковой составляющей по умолчанию задается как R = 1, а длина векторной – как (исходная длина измерения массива +R-1)/R).
2. Все циклы программы также разбиваются векторную и потоковую составляющие.
3. Подготовка входных данных для масштабирования программы процессором «Прокруст».

```
Var xCom : Array Integer
[n: Vector, n+1: Vector] Com;
…
for i = 1 to n do
  for j = 1 to m do
    xCom[j,i] =
         a*xCom[j-1,i]+y[i];
```

```
Var xCom : Array Integer
[(n+R1-1)/R1: Vector,R1: Stream,
(n+1+R2-1)/R2: Vector, R2: Stream] Com;
…
for vi = 1 to (n+R1-1)/R1 do
  for si = 1 to R1 do
    for vj= 1 to (n+1+R2-1)/R2 do
      for sj= 1 to R2 do
        xCom[vj,sj,vi,si] =
          a*xCom[vj-1,sj-1,vi,si] + y[vi,si];
    end;
```

Общая методика преобразования

4 этап (Прокруст): расчет и подбор рациональных параметров реализации задачи

1. Первым критическим ресурсом при расчете параметров является скважность решения – необходимо обеспечить скважность, равную 1. Для этого в цепь обратной связи добавляются регистры для организации вычислений по схеме «конвейер конвейеров».

2. Следующий критический параметр - число каналов КРП. Масштабируем число необходимых для вычислений каналов до имеющегося количества добавлением регистров в цепь обратной связи.

3. Если имеющихся КРП недостаточно для реализации базового подграфа задачи, необходимо редуцировать производительность задачи по критическому ресурсу (процессор «Щелкунчик»).

4. После определения минимального числа каналов КРП для реализации задачи, рассчитывается число устройств для распаралелливания по итерациям (если это возможно по задаче), либо процедурно (перезапуском кадров).
5. Адаптация программы под рассчитанные параметры: обработка аргументов с итерационным циклом (через регистр или внутреннюю память), организация обратной связи, организация «конвейера в конвейере»