Программирование реконфигурируемой вычислительной системы

ПРЕИМУЩЕСТВА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

1. **Единое языковое пространство** для синтеза вычислительной структуры и программирования информационных потоков

2. **Ресурсонезависимое программирование** – прикладные программы на языке высокого уровня могут портироваться на различные архитектуры и конфигурации реконфигурируемой вычислительной системы

3. **Автоматический контроль** за соответствием входов и выходов различных кристаллов ПЛИС (формирование корректных ucf-файлов)

4. **Автоматическая синхронизация** информационных потоков, проходящих через вычислительную структуру, реализованную на множестве кристаллов ПЛИС

5. Сокращение как времени программирования прикладной задачи в несколько раз, так и числа специалистов: **программированием реконфигурируемой системы занимается только прикладной программист**.

ЯЗЫК COLAMO

Основные отличительные особенности:

1. Массивы различаются по типу доступа: Stream(последовательный доступ) и Vector(параллельный доступ);
2. Переменные различаются по способу хранения в памяти: Mem(мемориальная переменная – ячейка памяти), Reg (регистровая переменная - регистр), Com (коммутационная переменная – точка на графе, физически не хранится, связывает результаты вычислений);
3. Фундаментальные вычислительные структуры: кадр, подкадр, конструкция Let;
4. Наличие правила единственной подстановки – переменная, за исключением регистровой, в теле кадра может получить значение только один раз;
5. Все используемые в теле кадра вычислительные операции соответствуют вычислительным устройствам.

Системное программное обеспечение

Направления исследований:

1) Трансляция языка высокого уровня COLAMO.

2) Синтез схемотехнических решений на уровне логических ячеек ПЛИС.

3) Синтез конфигурации на уровне макрообъектов.

4) Трансляция языка описания софт-архитектур на языке SADL.

5) Оптимизация вычислительной структуры прикладной задачи.

6) Трансляция языка высокого уровня SET@L.

Трансляция языка высокого уровня COLAMO

Области исследований:
– масштабирование параллельной программы;
– преобразование типов данных;
– редукция производительности;
– определение способа организации вычислений;
– управление информационными потоками данных;
– макрообъектное программирование;
– ресурсонезависимое программирование.

Трансляция языка высокого уровня COLAMO

Трансляция параллельной программы с языка высокого уровня COLAMO заключается в создании следующих компонентов параллельной программы:

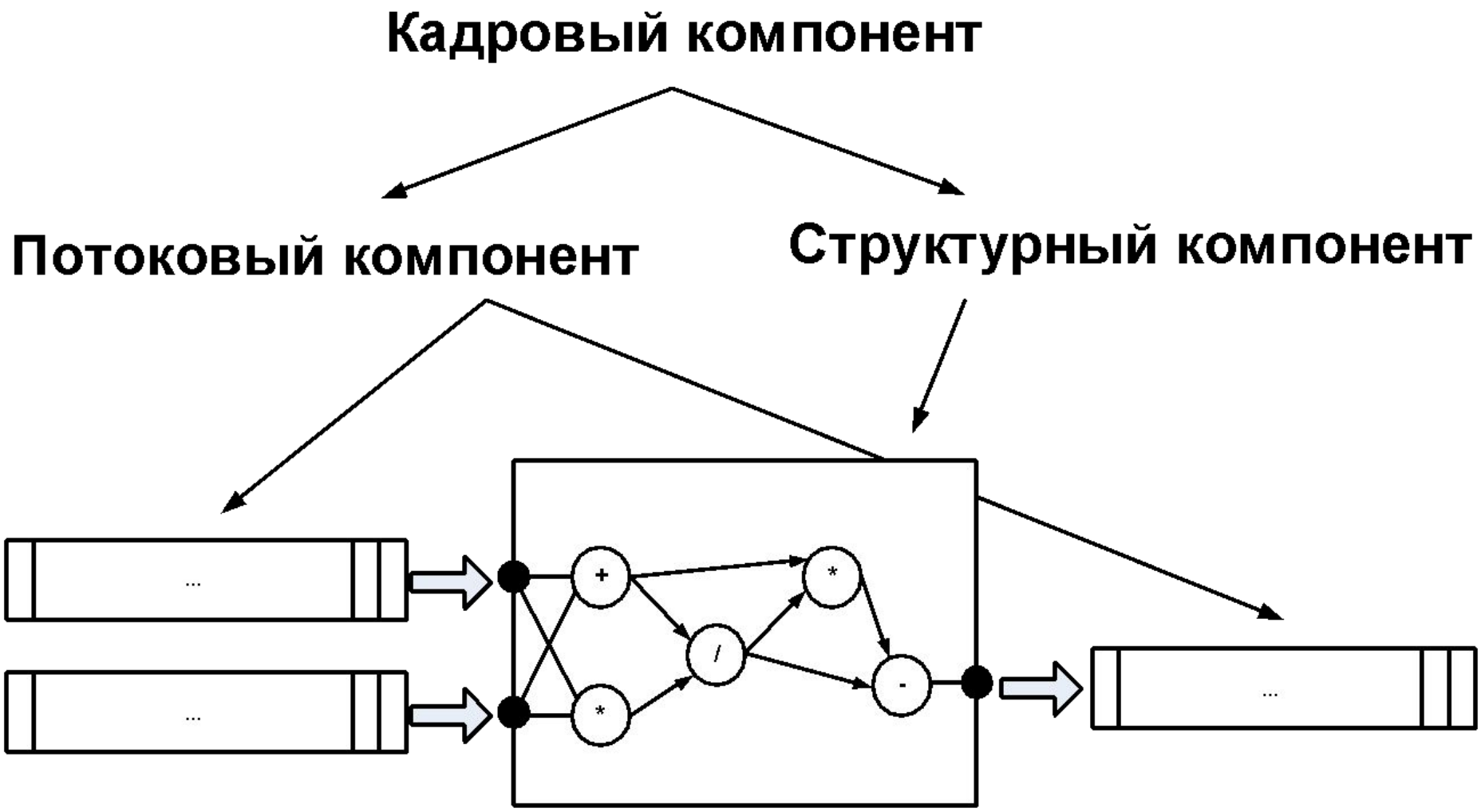
– структурный компонент - описывает структуру вычислительного конвейера для информационного графа задачи;

– потоковый компонент – описывает потоки данных для структурного компонента программы;

– процедурный компонент – описывает последовательность выполнения программы;

– управляющий компонент - осуществляет контроль за процессом выполнения программы и управление процессом загрузки и выгрузки данных.

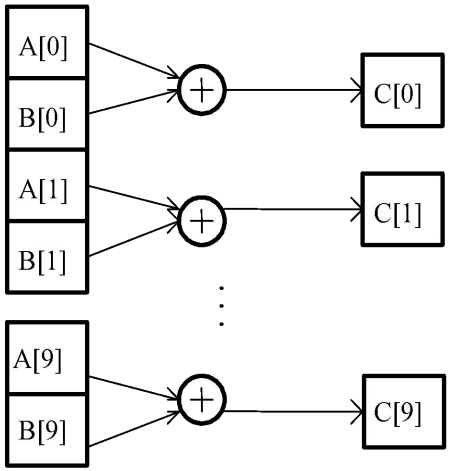
Трансляция языка высокого уровня COLAMO

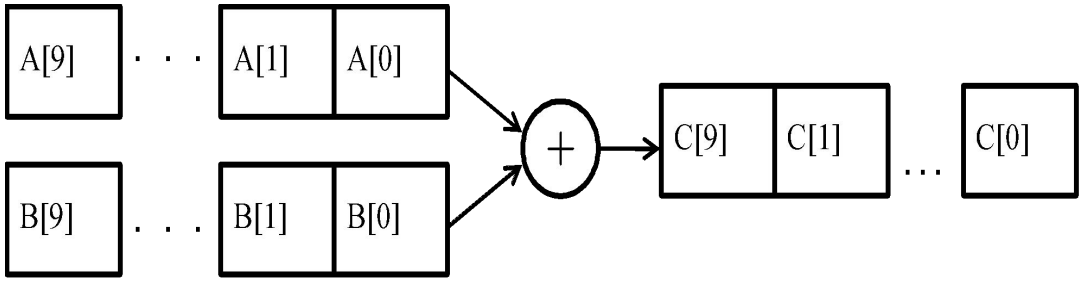


Масштабирование параллельной программы

```
VAR A,B,C : array real [ 10 : Vector] MEM;
VAR I : Number;
CADR summa;
   For I := 0 to 9 do
     C[i]:=A[i]+B[i];
ENDCADR;
```

```
VAR A,B,C : array real [ 10 : Stream] MEM;
VAR I : Number;
CADR summa;
    For I := 0 to 9 do
        C[i]:=A[i]+B[i];
ENDCADR;
```

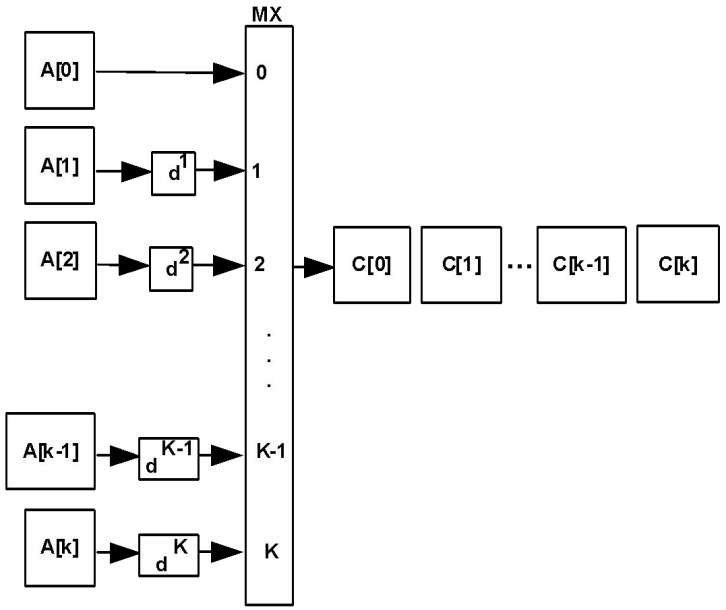




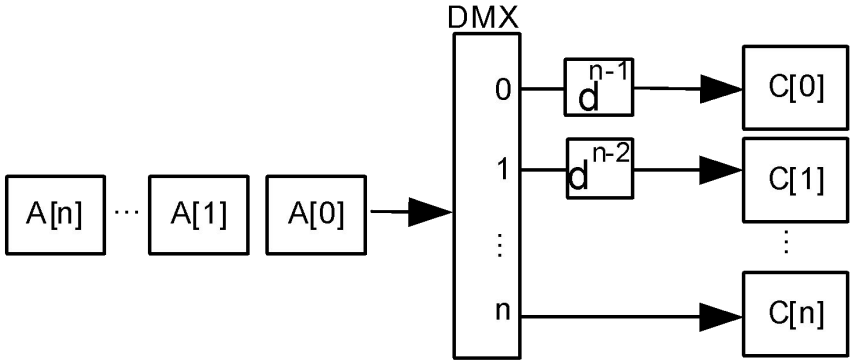
Одна и та же программа в зависимости от типа доступа к элементам массива будет иметь разные информационные графы

Масштабирование параллельной программы

```
Var a : Array Integer [10:Vector] Mem;
Var c : Array Integer [10:Stream] Mem;
Var I : Number;
Const k = 9;
Cadr summa;
    For i:=0 To k Do
        c[i] :=a[i];
Endcadr;
```

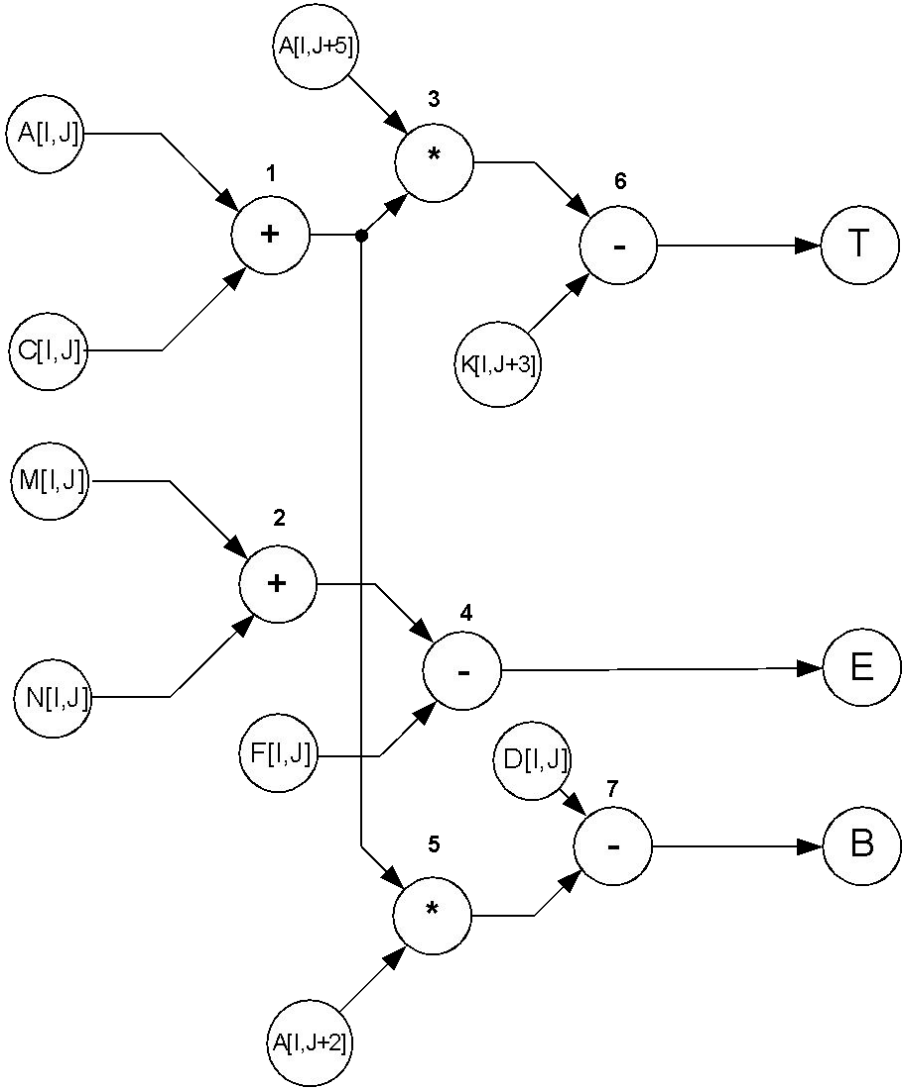



```
Var a : Array Integer [10: Stream] Mem;
Var c : Array Integer [10: Vector] Mem;
Var i : Number;
Const n = 9;
Cadr summa;
    For i:=0 to n do
        c[i] := a[i];
Endcadr;
```



Трансляция языка высокого уровня COLAMO

```
Program Primer;
Include Library;
Var A,B,C,D,E,F,K,N,M,T : array [X : Stream,
Y : Stream] : Mem;
Var Com : Vector Mem;
Define X=5;
Define Y=6;
Cadr
  For I := 0 to X-1 do
    For J := 0 to Y-1 do
      Begin
          Com := ( A[ I , J ]+ C[ I , J ] );
          B[ I , J ] = Com * A[ I , J +2 ] - D[ I , J ];
          E[ I , J ] = M[ I , J ] + N[ I , J ] - F[ I , J ];
          T[ I , J ] = A[ I , J + 5 ] * Com - K[ I , J+ 3];
      End;
EndCadr;
End_Program.
```

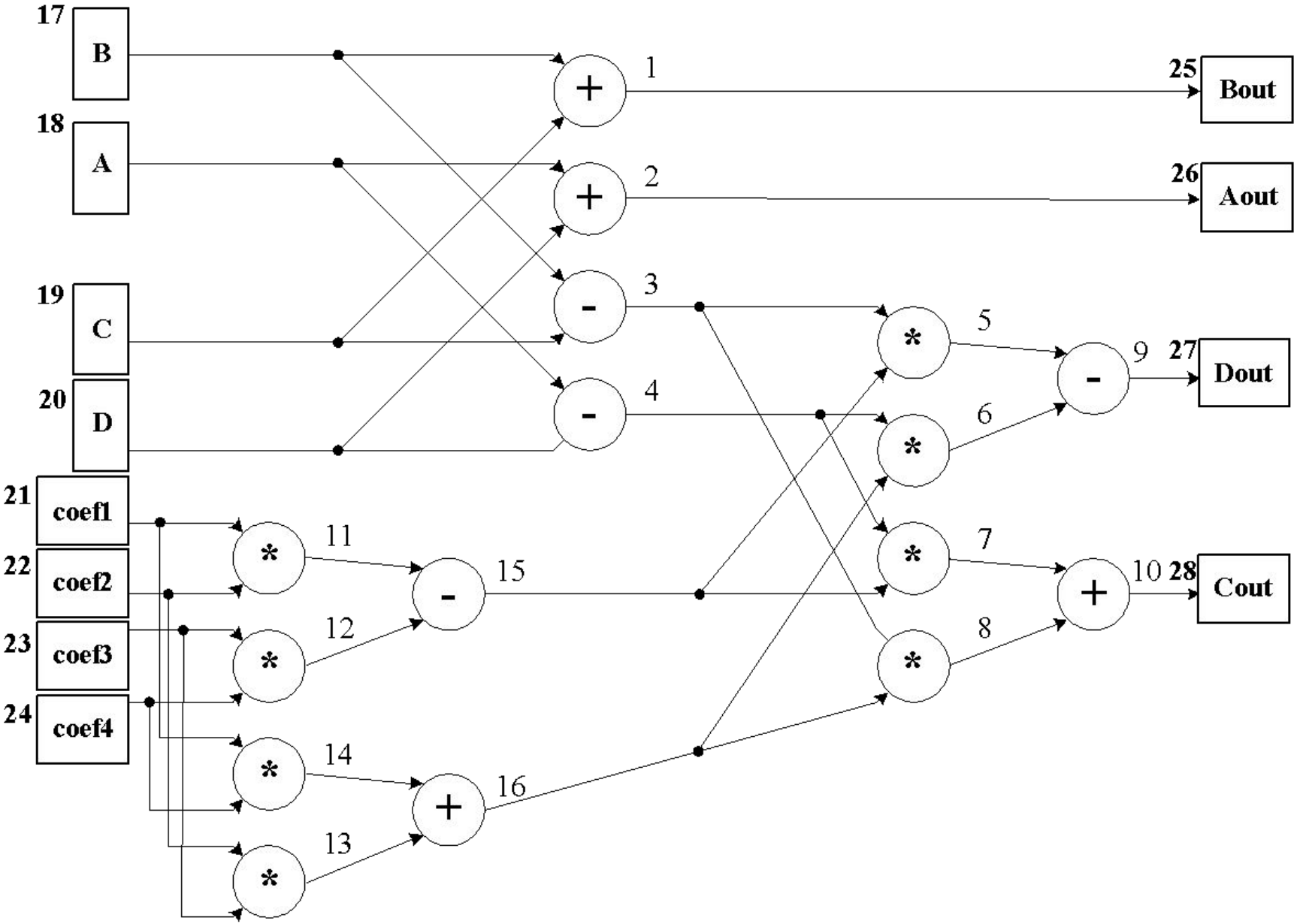


Редукция производительности по функциональным устройствам в виде однокадровой программы

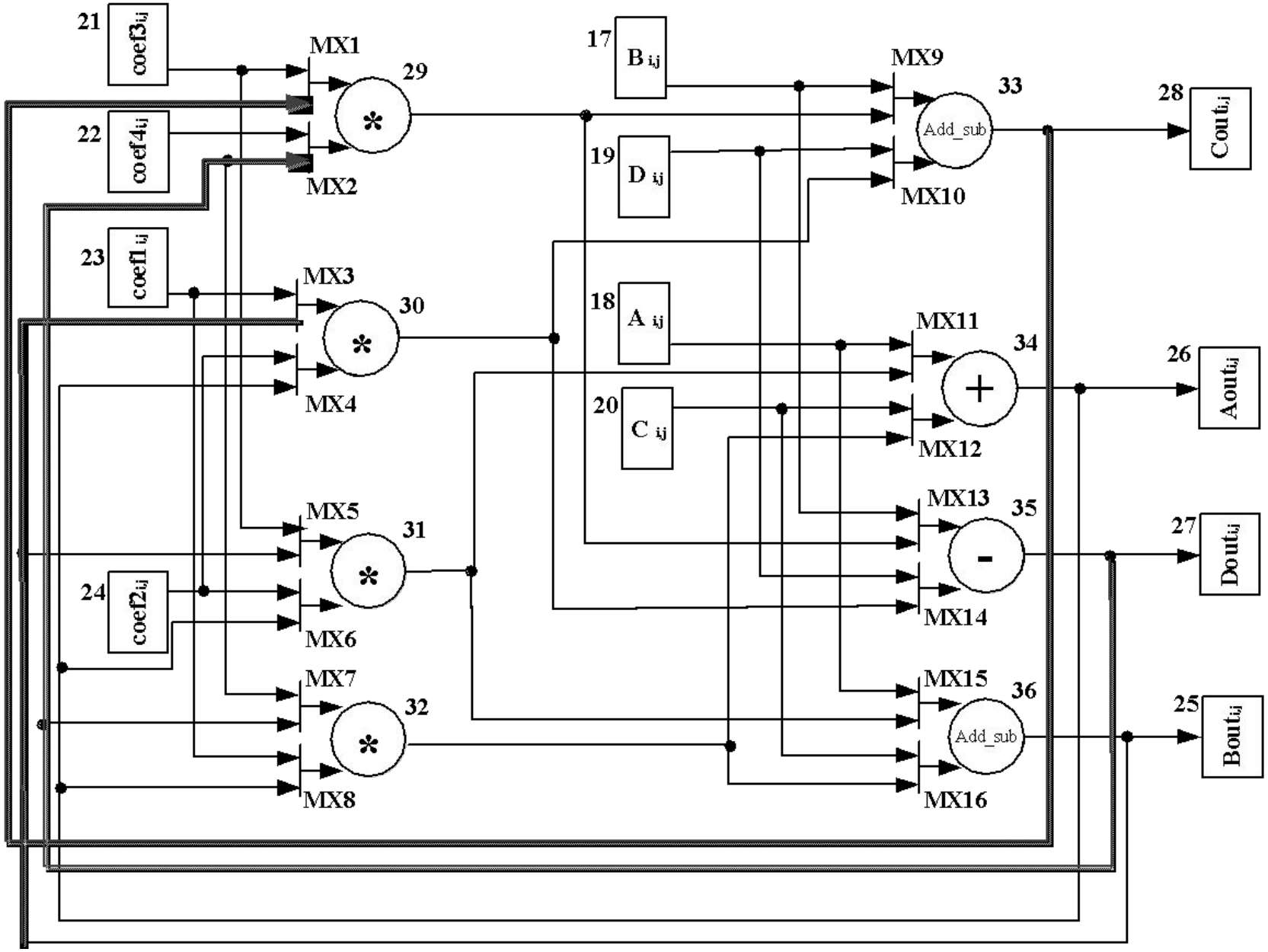
Программная реализации БО БПФ (степень редукции 1)

```
Program Reduction1;
Const X = 1;

subcadr BaseNode (In_ReA, In_ImA, In_ReB, In_ImB, aCoef1, aCoef2, aCoef3, aCoef4, Out_ReA, Out_ImA, Out_ReB, Out_ImB);
Var In_ReA, In_ImA, In_ReB, In_ImB, aCoef1, aCoef2, aCoef3, aCoef4, Out_ReA, Out_ImA, Out_ReB, Out_ImB : Integer Com;
Var Com1, Com2, Com3, Com4, Com5, Com6, Com7, Com8, Com9, Com10, Com11, Com12 : integer Com;
#Reduction of device X;
  Out_ReA := In_ReA + In_ReB;   Out_ImA := In_ImA + In_ImB;
  Com1 := In_ReA - In_ReB; Com2 := In_ImA - In_ImB; Com3 := aCoef1 * aCoef2;
  Com4 := aCoef3 * aCoef4; Com5 := aCoef1 * aCoef4;   Com6 := aCoef2 * aCoef3;
  Com7 := Com3 - Com4;   Com8 := Com5 + Com6; Com9 := Com1 * Com7;
  Com10 := Com2 * Com8;  Com11 := Com2 * Com7;   Com12 := Com1 * Com8;
  Out_ImA := Com9 - Com10;   Out_ReB := Com11 - Com12;
#EndReduction;
EndSubCadr;

Cadr Cadr1;
   for i := 0 to 0 do
     for j := 0 to 99 do
       BaseNode(AIn[i,j], BIn[i,j], CIn[i,j], DIn[i,j], Coef1[i,j], Coef2[i,j], Coef3[i,j],  Coef4[i,j], AOut[i,j], BOut[i,j],COut[i,j], DOut[i,j]);
EndCadr;
End_Program.
```

Информационный граф, соответствующий исходной программе реализации БО БПФ (степень редукции 1)

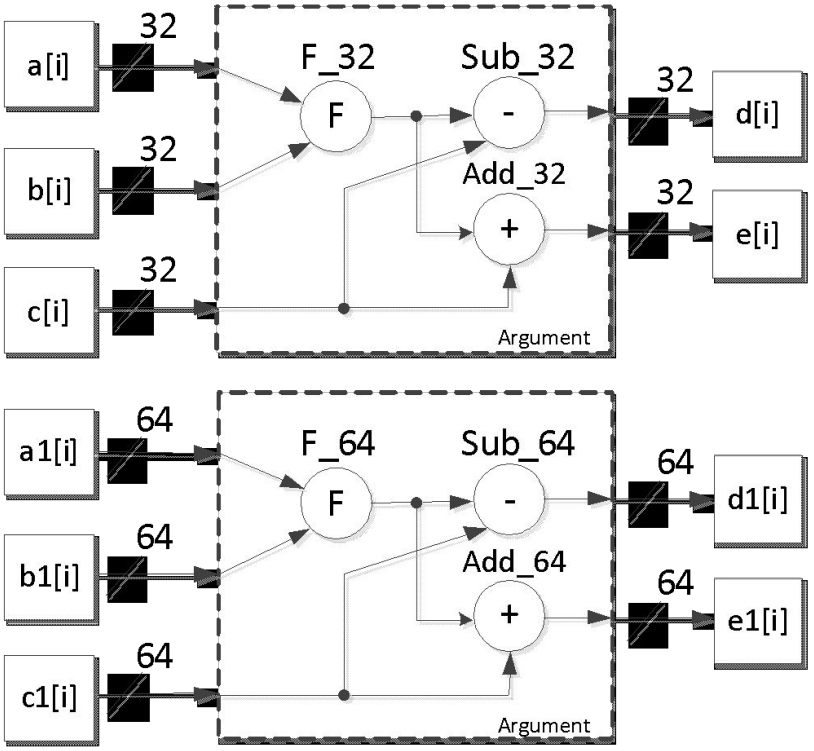


Редукция производительности БО БПФ по функциональным устройствам со степенью 2



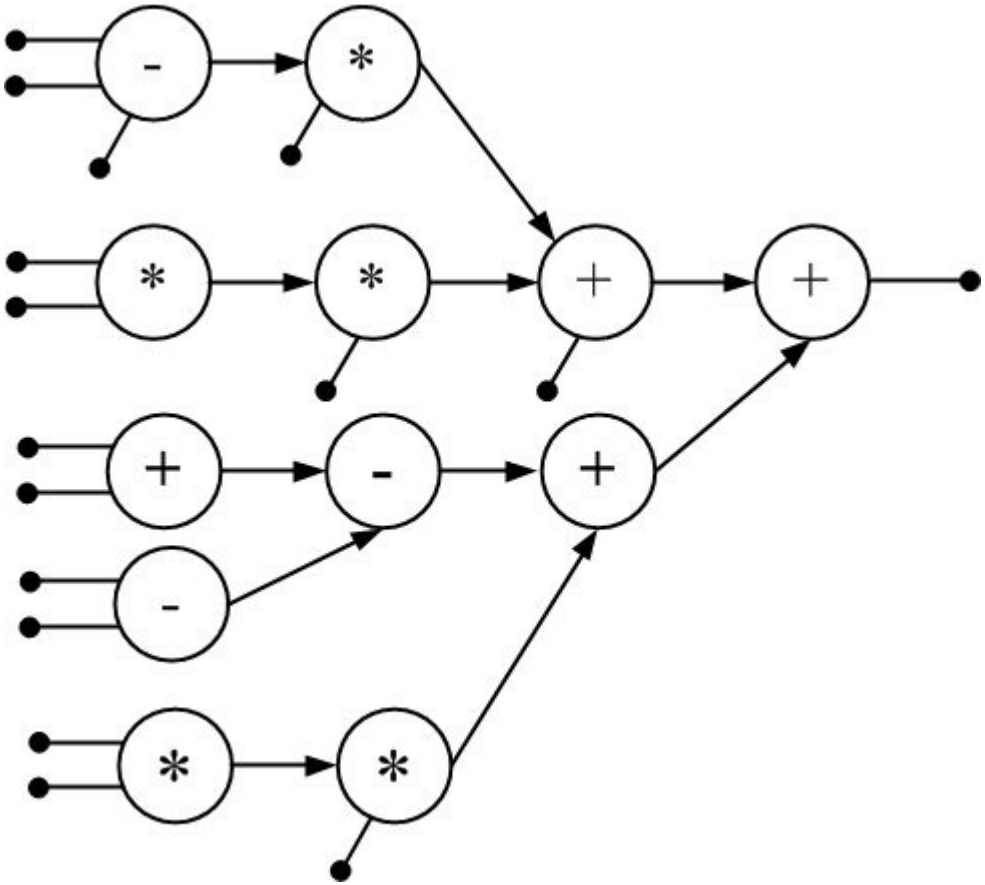
Трансляция языка высокого уровня COLAMO

```
Program VirtSubCadr;
Var a, b, c, d, e : Array Integer [100 : Stream] Mem;
Var a1, b1, c1, d1, e1: Array Int64 [100 : Stream]
Mem;
Var r : Integer Com;
Var i : Number;
Const n = 100;
SubCadr Argument (In : In1, In2, In3; Out : Out1,
Out2);
Var In1, Out1, Out2 : Integer Com;
Var In2, In3, Com1 : Virtual Com;
      Com1 := F(In1, In2);
      Out1 := Com1 + In3;
      Out2 := Com1 - In3;
EndSubCadr;
Cadr One;
   For i := 0 to n - 1 do
      begin
        Argument(a[i], b[i], c[i], d[i], e[i]);
        Argument(a1[i], b1[i], c1[i], d1[i], e1[i]);
      end;
EndCadr;
EndProgram.
```

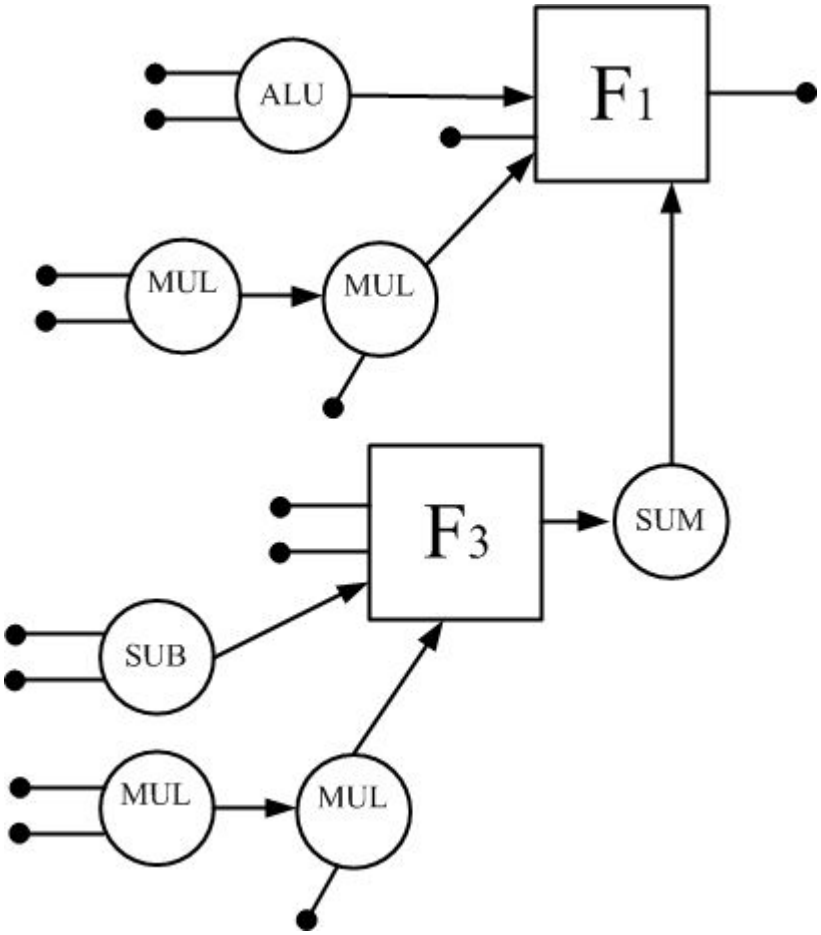


Генерация макрообъектов

Крупные шаблоны объектов софт-архитектуры



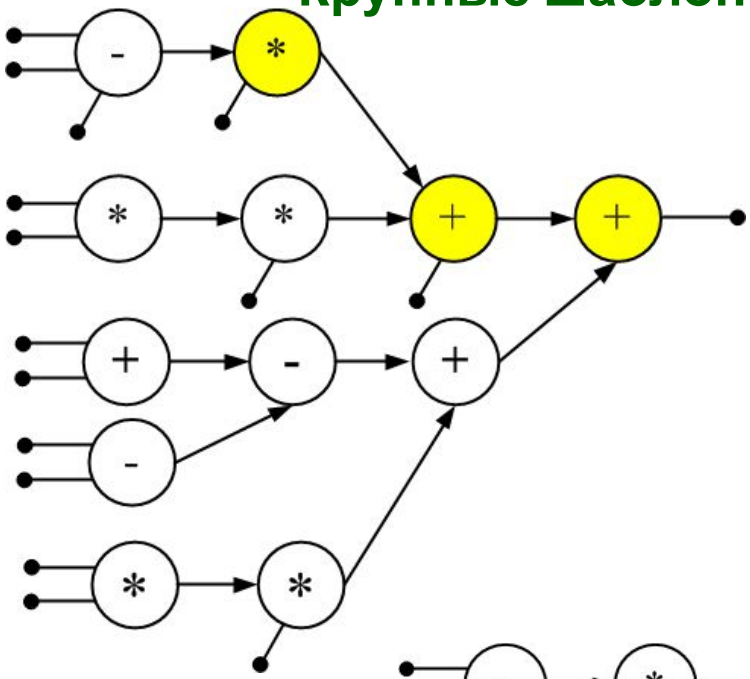
Граф задачи

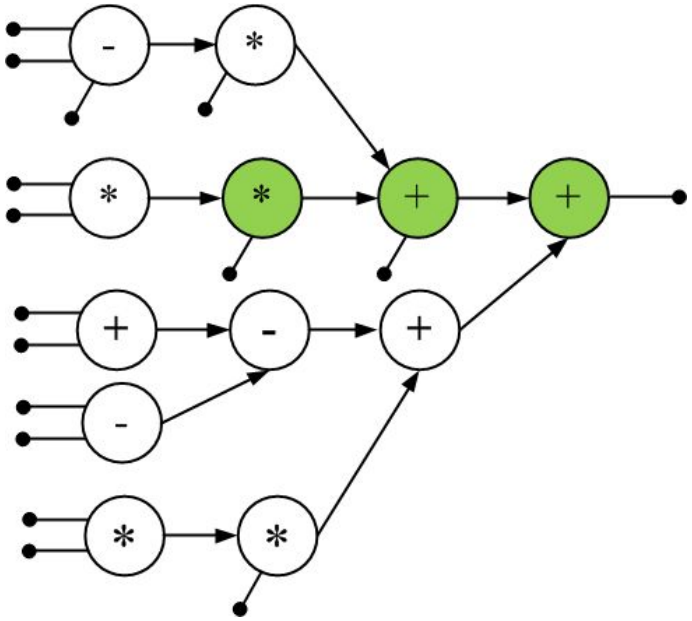


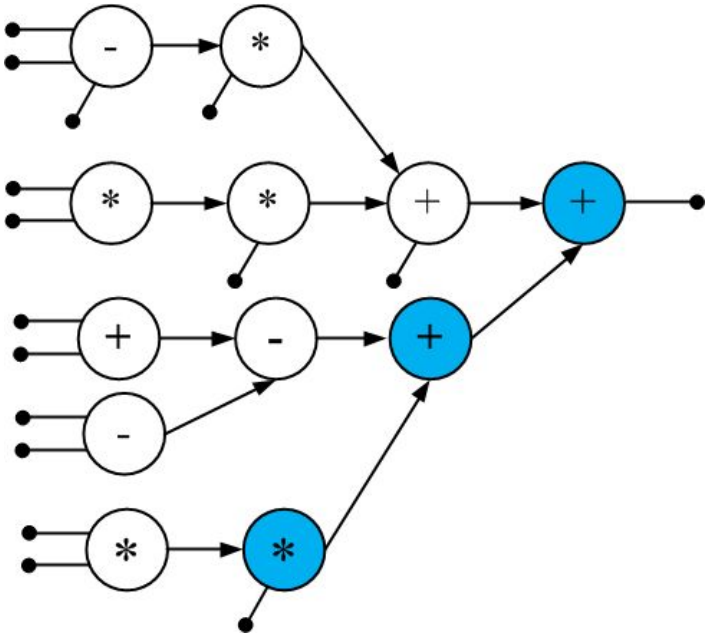
Граф софт-архитектуры

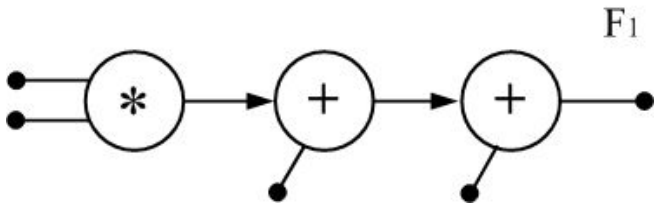
Генерация макрообъектов

Крупные шаблоны объектов софт-архитектуры





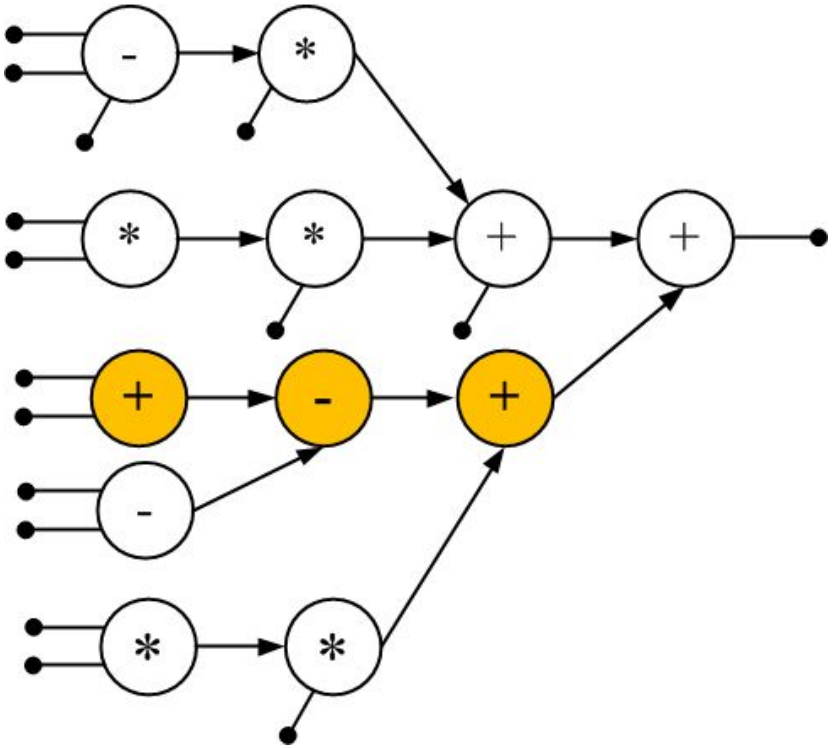


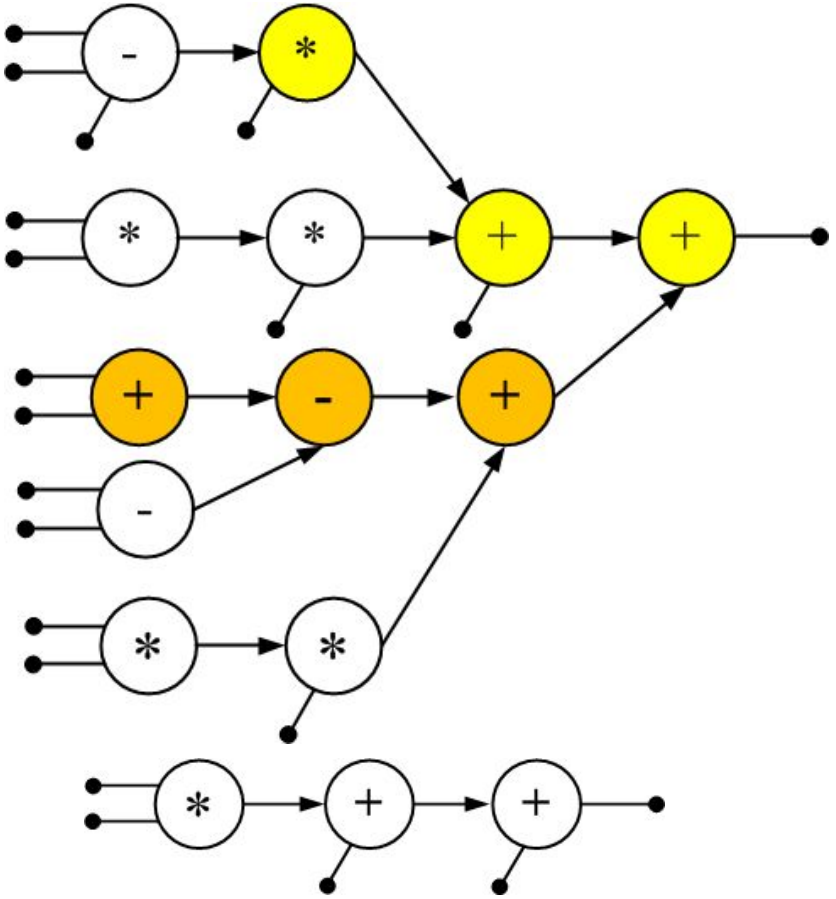


Шаблон

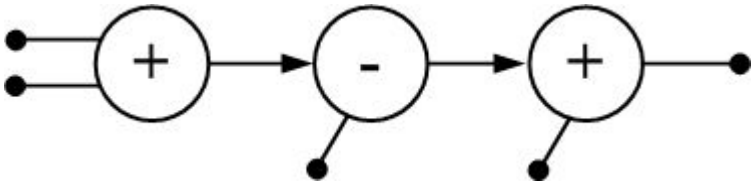
Генерация макрообъектов

Крупные шаблоны объектов софт-архитектуры

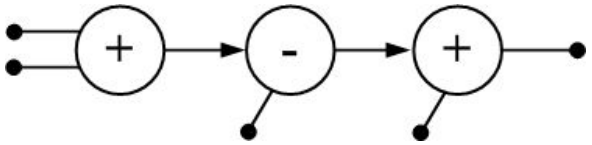




Шаблон объекта F_1



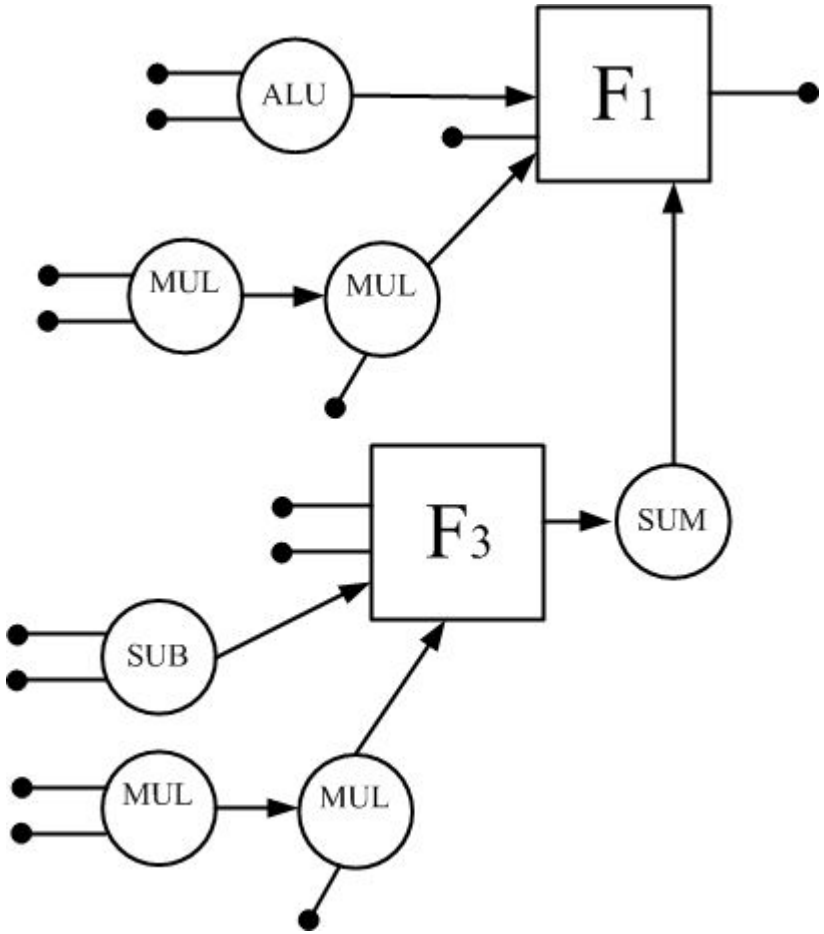
Шаблон объекта F_2



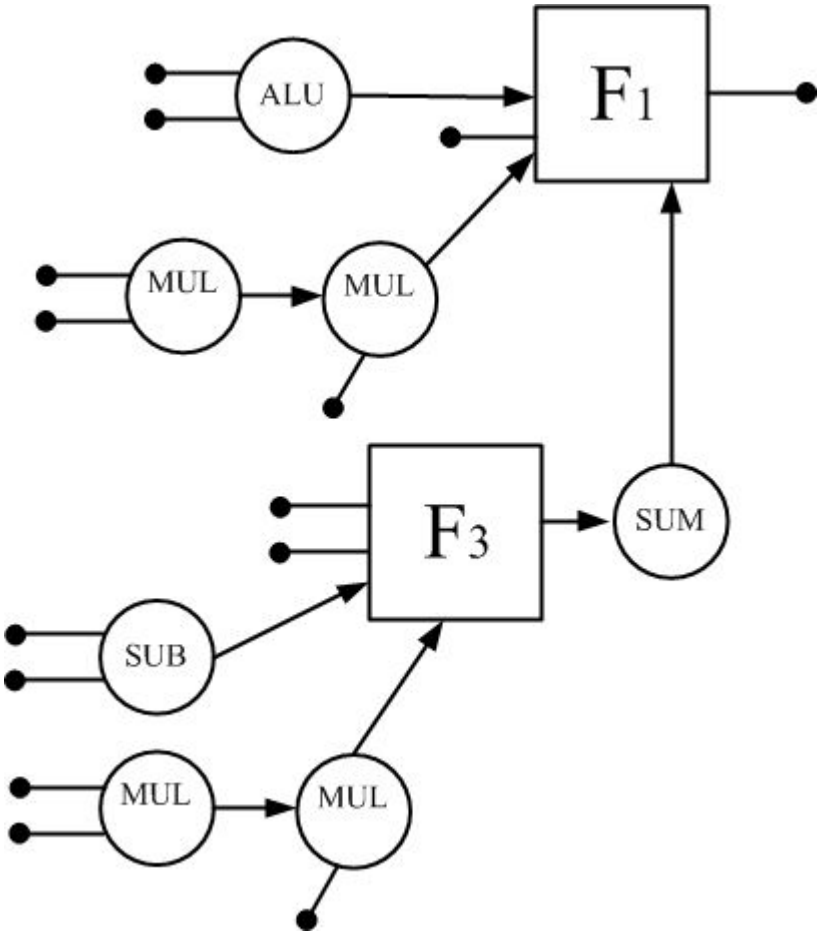
Шаблон объекта F_2

Генерация макрообъектов

Крупные шаблоны объектов софт-архитектуры



Граф задачи, покрытый объектами софт-архитектуры



Граф софт-архитектуры

Синтез схемотехнических решений на уровне логических ячеек ПЛИС

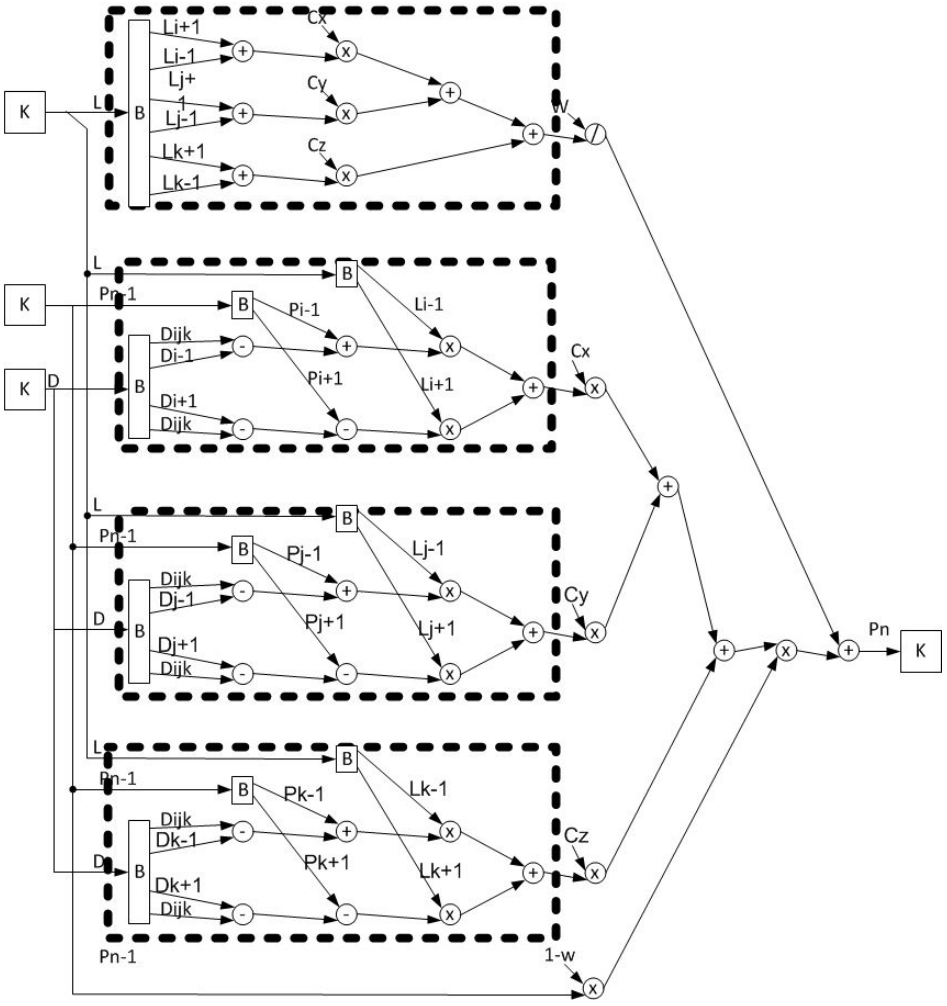
Области исследований:

-поиск сильносвязных фрагментов вычислительных структур параллельных программ;

-разбиение вычислительных структур параллельных программ на заданное число непересекающихся фрагментов методом последовательной рекурсивной бисекции;

-трассировка внешних связей размещённых фрагментов вычислительных структур параллельных программ генетическим алгоритмом.

Поиск сильносвязных фрагментов вычислительных структур параллельных программ

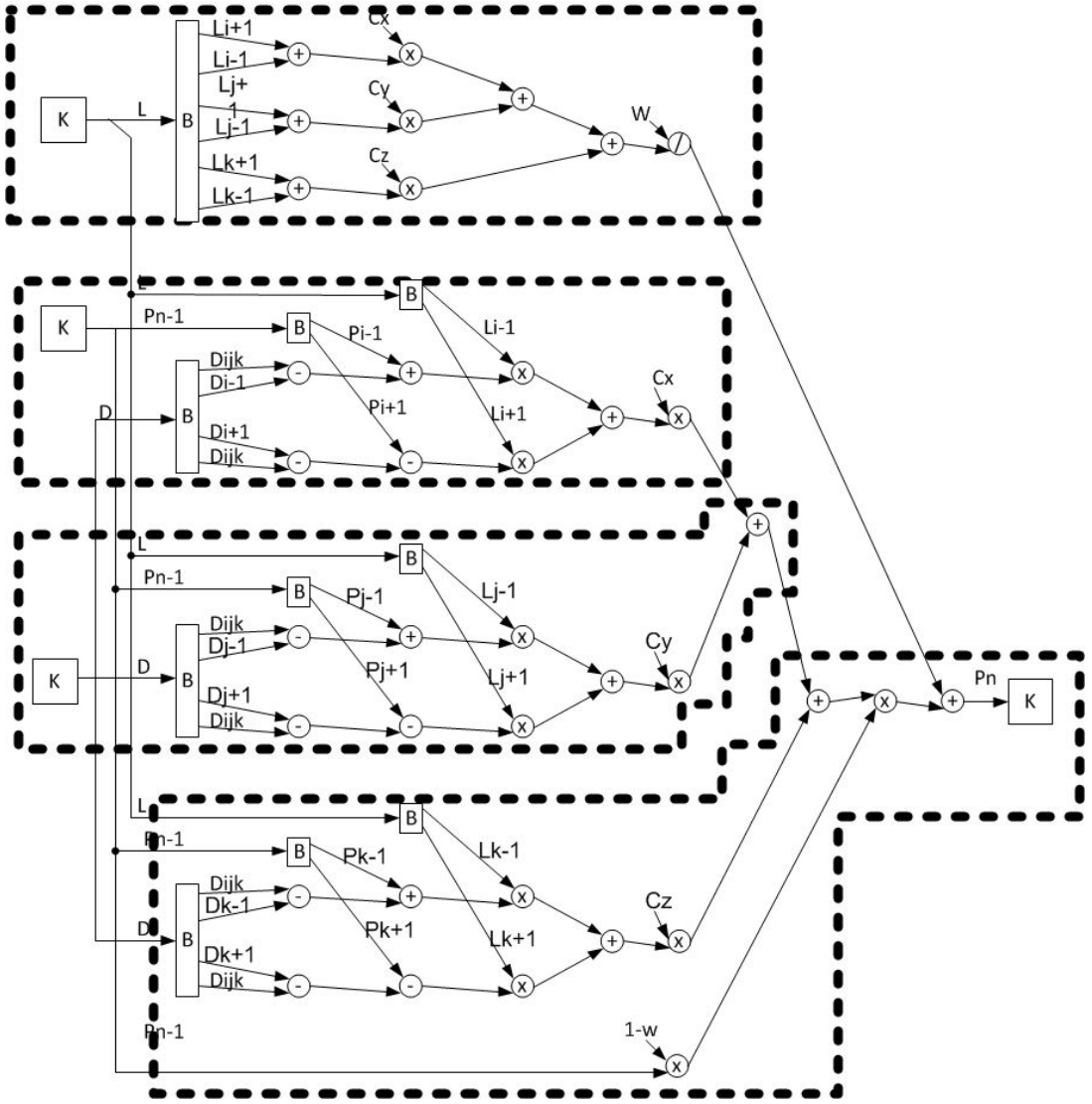


Поиск фрагментов вычислительных структур, для которых отношение

N/L >= coef1,
при ограничениях W(N) <= W(ПЛИС),
L <= coef2

где N – количество блоков вычислительной структуры во фрагменте,
L – количество внешних информационных связей фрагмента,
coef1, coef2 – некоторые заданные значения,
W(N) – суммарный аппаратный ресурс, занимаемый в кристалле ПЛИС блоками фрагмента;
W(ПЛИС) – аппаратный ресурс одного кристалла ПЛИС.

Разбиение вычислительных структур параллельных программ на заданное число непересекающихся фрагментов методом последовательной рекурсивной бисекции

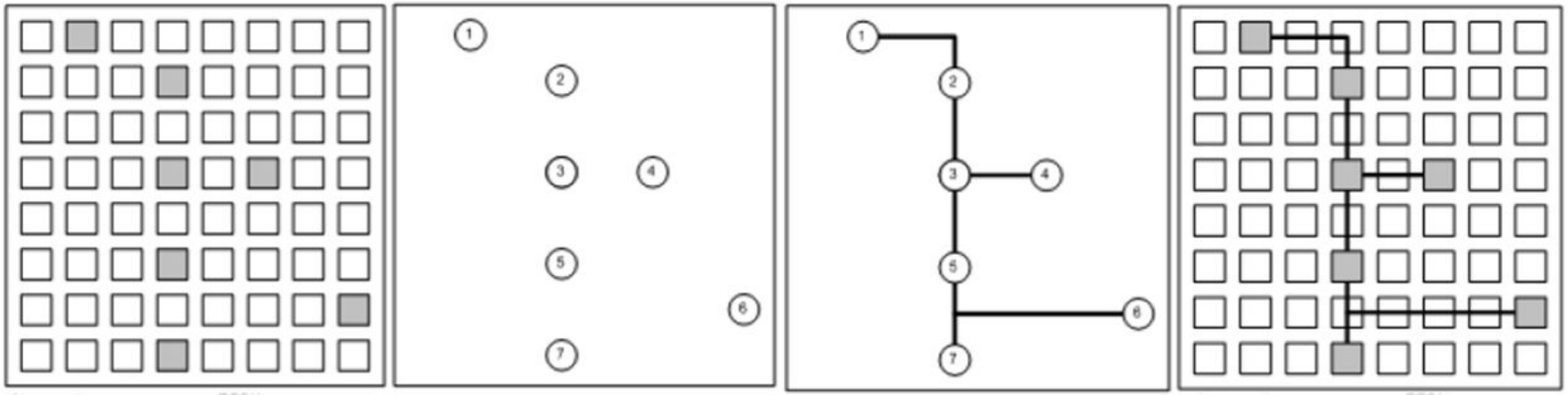


Разбиение вычислительной структуры параллельной программы на M или менее фрагментов при ограничениях:

при ограничениях W(N) <= W(ПЛИС),
L <= coef1

где N – количество блоков вычислительной структуры во фрагменте,
L – количество внешних информационных связей фрагмента,
coef1 – некоторое заданное значение,
W(N) – суммарный аппаратный ресурс, занимаемый в кристалле ПЛИС блоками фрагмента;
W(ПЛИС) – аппаратный ресурс одного кристалла ПЛИС.

Трассировка внешних связей размещённых фрагментов вычислительных структур параллельных программ генетическим алгоритмом

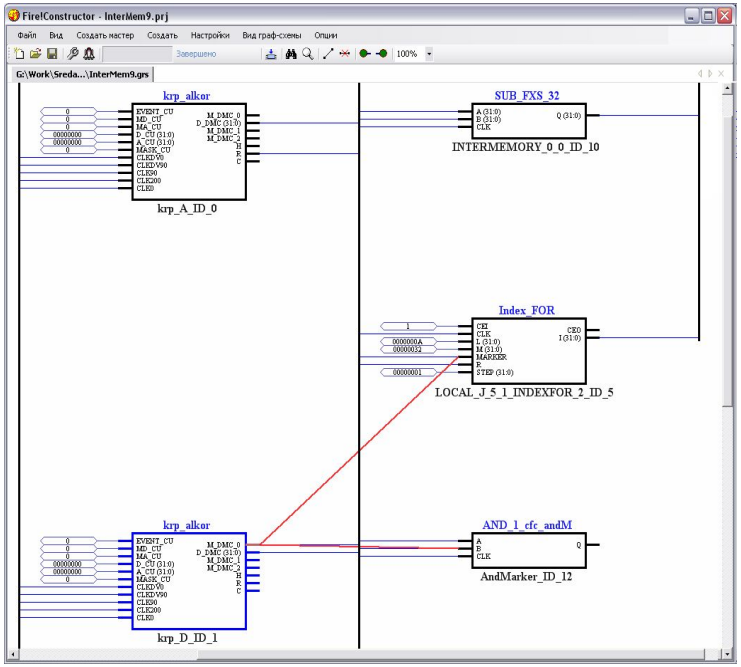


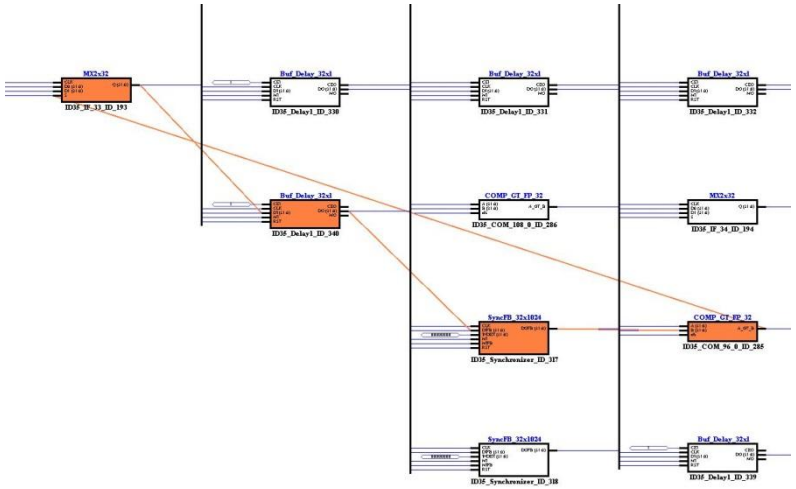
Под качественным решением задачи трассировки подразумевается достижение 100-% разводки всех соединений с минимизацией заданных критериев (минимум суммарной длины соединений, минимум максимальной длинны отдельного соединения и др.).

При последовательном подходе цепи распределяются по областям последовательно. В основе большинства из них лежит волновой алгоритм Ли и его модификации. Качество решения во многом определяется порядком трассируемых соединений. Анализ существующих методов упорядочения показывает, что не существует радикального метода, гарантирующего оптимальную трассировку.

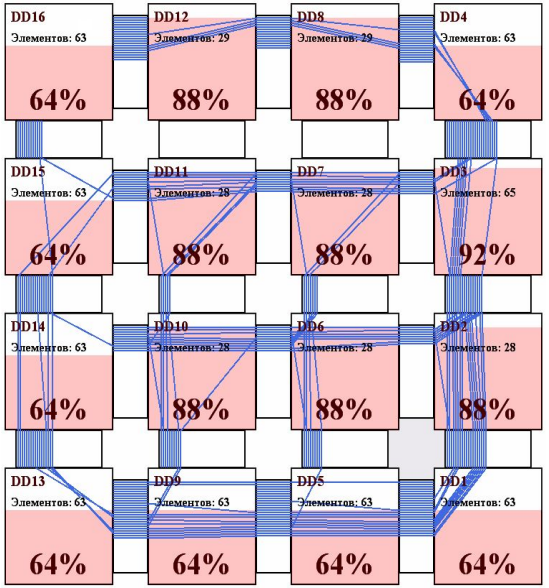
Сущность генетических алгоритмов трассировки заключается в том, что для каждого соединения ti, формируется набор вариантов его реализации, т.е. набор вариантов прохождения его по областям. Цель задачи заключается в нахождении на заданном наборе таких вариантов, которые обеспечивают наилучшее решение задачи трассировки.

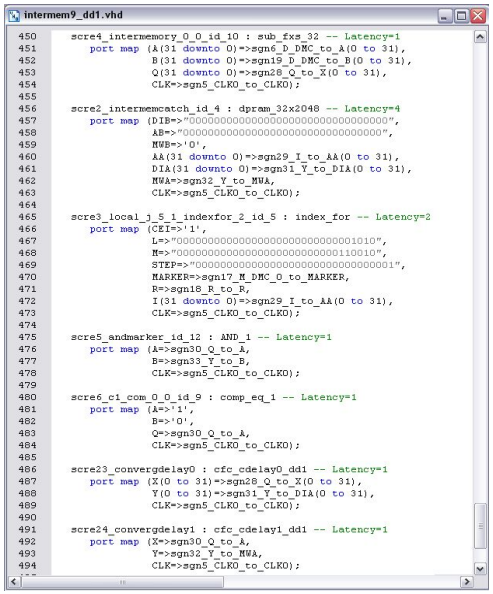
Синтез схемотехнических решений на уровне логических ячеек ПЛИС





Функциональная схема структурной параллельной программы





Синтезированный VHDL-код

Синтез конфигурации на уровне макрообъектов

Области исследований:

-многоуровневая визуализация результатов отображения многокадровых задач на софт-архитектуру РВС;

-проверка возможности успешного отображения многокадровой задачи на софт-архитектуру РВС путем анализа структуры кадров задачи и имеющегося вычислительного ресурса РВС;

-поиск вариантов размещения Complex-структур информационного графа кадра многокадровой задачи на множество объектов софт-архитектуры РВС;

-процедура трассировки внешних и внутренних связей между размещенными вершинами информационного графа кадра многокадровой задачи;

-проверка корректности отображения многокадровой задачи на софт-архитектуру РВС.

Многоуровневая визуализация результатов отображения многокадровых задач на софт-архитектуру РВС

Необходимо обеспечить возможность визуализации:

- софт-архитектуры РВС без распределения объектов по корпусам ПЛИС;
- софт-архитектуры РВС с распределением объектов по корпусам ПЛИС;
- отдельных объектов софт-архитектуры РВС с учетом вложенных объектов с различной степенью вложенности;
- параметров любого объекта софт-архитектуры (шаблоны, параметры объекта, выводы и т.п.);
- каждого графа из списка информационных графов кадров многокадровой задачи (исходного);
- каждого графа из списка информационных графов кадров многокадровой задачи (отображенного на софт-архитектуру РВС);
- отдельных вершин информационного графа выбранного кадра со списком необходимых параметров с учетом того, что вершина может быть комплексной.

Кроме того, необходимо обеспечить возможность выделения другим цветом:

- любой трассы, выбираемой пользователем;
- множества трасс, выходящих из выбранной вершины-источника.

Проверка возможности успешного отображения многокадровой задачи на софт-архитектуру РВС путем анализа структуры кадров задачи и имеющегося вычислительного ресурса РВС

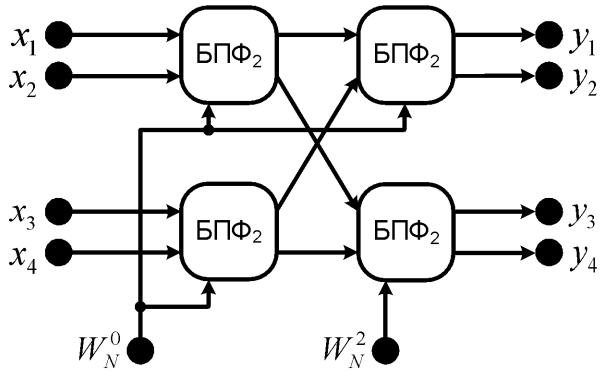
Успешное отображение многокадровой задачи возможно в том случае, если:

- для каждой элементарной вершины существует некоторый объект софт-архитектуры РВС (один или несколько), который реализует выполнение операции, соответствующей вершине, и имеет количество выводов, достаточное для формирования всех связей вершины с другими вершинами информационного графа отображаемого кадра.

- для каждой комплексной вершины существует некоторый набор объектов софт-архитектуры РВС (один или несколько), которые реализуют вычислительный шаблон, заданный для комплексной вершины, имеют достаточное количество выводов для формирования связей комплексной вершины с другими вершинами информационного графа отображаемого кадра и обеспечивают возможность создания связей между собой.

Если для какой либо вершины (элементарной или комплексной) не найдено ни одного объекта для размещения, необходимо выдать сообщение об отказе размещения многокадровой задачи на софт-архитектуру РВС, указав при этом имя кадра, идентификатор вершины и имя соответствующей вычислительной операции.

Поиск вариантов размещения Complex-структур информационного графа кадра многокадровой задачи на множество объектов софт-архитектуры РВС



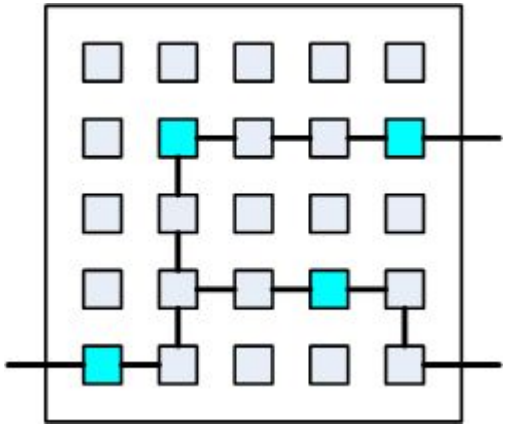
Complex-структуры информационного графа многокадровой задачи или иными словами комплексные вершины, состоящие из множества элементарных вершин, размещаются на группу объектов софт-архитектуры РВС, которые соответствуют вычислительному шаблону комплексной вершины и обеспечивают связи согласно информационному подграфу комплексной вершины.

Вариантов размещения комплексной вершины может быть несколько:

- комплексная вершина может быть размещена только в одном корпусе ПЛИС, и в этом случае необходимо формировать внутренние связи между занимаемыми ею объектами софт-архитектуры;

- комплексная вершина может быть размещена в нескольких корпусах ПЛИС, и в этом случае необходимо формировать как внутренние, так и внешние связи между всеми занимаемыми ею объектами, как внутри корпусов ПЛИС, так и между корпусами ПЛИС.

Процедура трассировки внешних связей между размещенными вершинами информационного графа кадра прикладной задачи



Процедура трассировки должна обеспечивать формирование трасс между корпусами ПЛИС. Необходимо учитывать, что:

- разрядность трасс может быть произвольной;
- несколько трасс могут иметь один и тот же источник.

При этом необходимо формировать трассы таким образом, чтобы количество занимаемых входов/выходов корпусов ПЛИС было **минимальным**.

Трансляция языка описания софт-архитектур на языке SADL

Области исследований и разработки:

–портация прикладных программ на различные софт-архитектуры реконфигурируемых вычислительных систем.

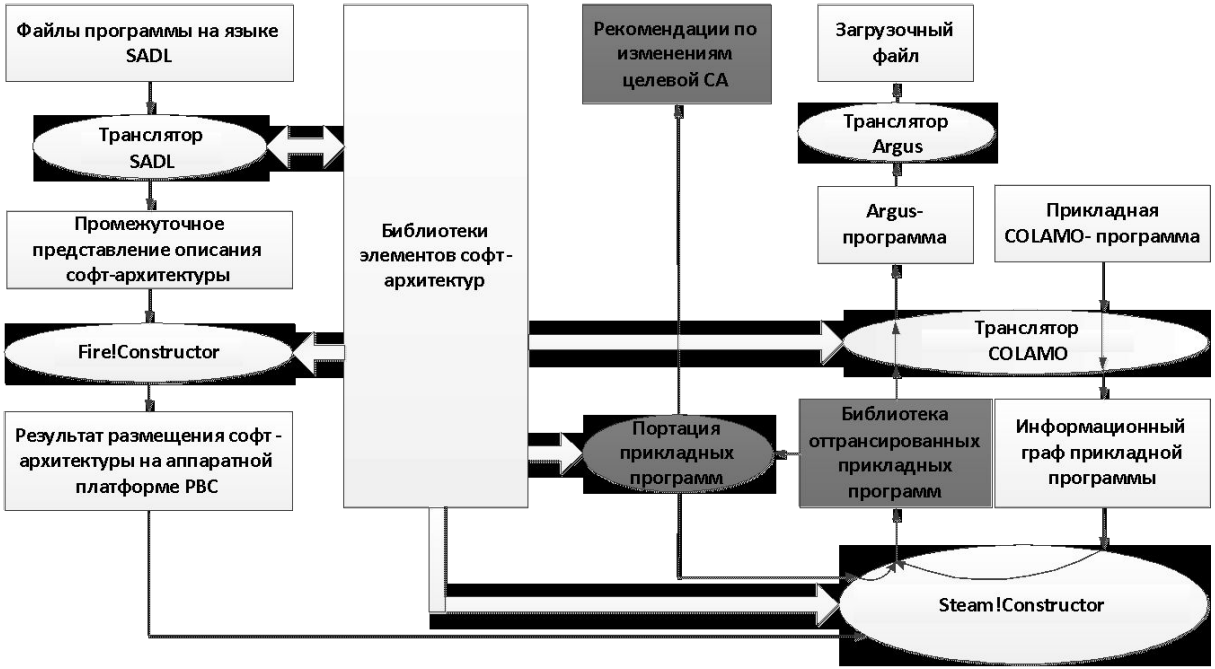
–реализация процессорных объектов и объектов памяти в языке описания софт-архитектур РВС;

–реализация синтаксического и семантического анализа в трансляторе языка SADL;

–реализация механизмов распараллеливания и каскадирования конструкций языка описания софт-архитектур;

–разработка графической оболочки для визуализации софт-архитектур.

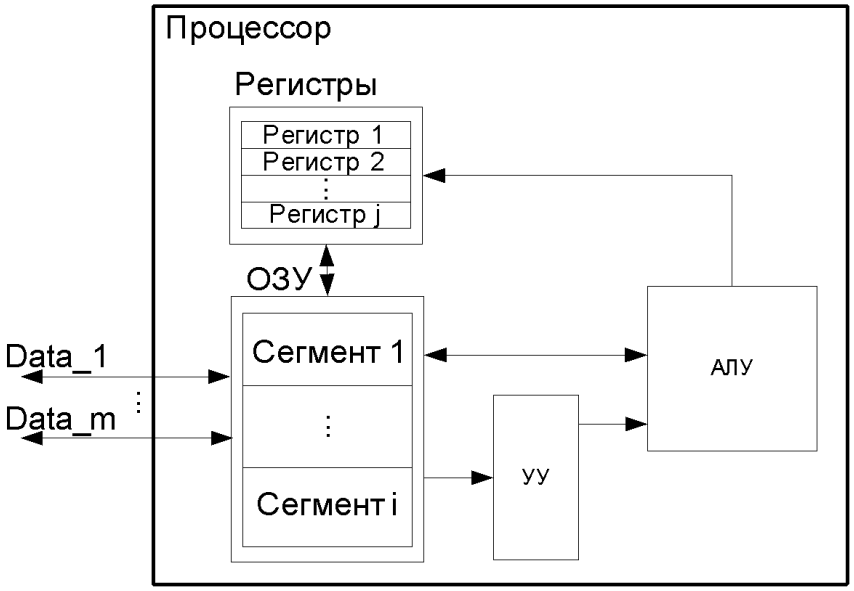
Портация прикладных программ на различные софт-архитектуры реконфигурируемых вычислительных систем



Существует некоторая софт-архитектура CA_1 и набор прикладных параллельных программ на языке Colamo отлаженных и оттранслированы на ней.

Необходимо создать методы, позволяющие определить, возможна ли портация каждой прикладной параллельной программы из данного набора на новую целевую софт-архитектуру CA_2. Если портация возможна, то синтазатор Steam!Constructor выполняет отображение программы на целевую СА. Если портация невозможна, то для каждой из набора программ формируется список рекомендаций по изменению целевой СА.

Реализация процессорных объектов и объектов памяти в языке описания софт-архитектур РВС.



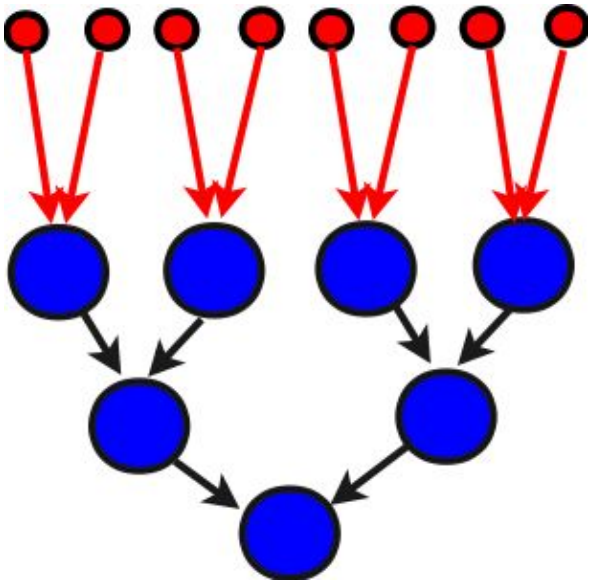
Систему управления процессорными объектами предлагается описывать посредством набора команд ассемблера и используемого при этом интерфейса подключения.

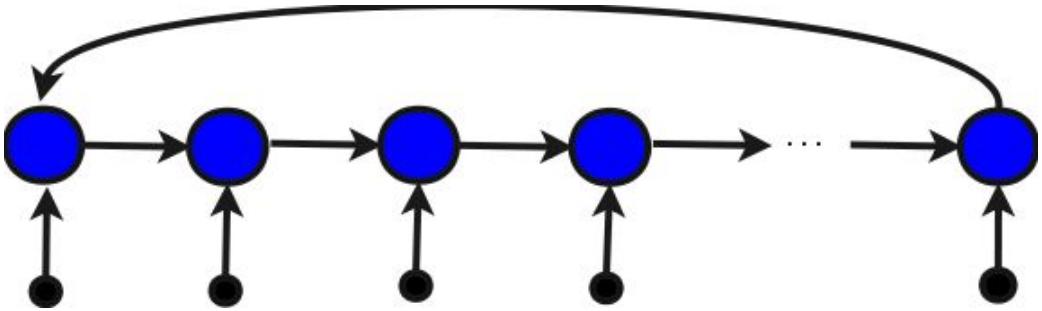
Proc[A, C],

A = {asm_1, asm_2, …, asm_i}– множество систем команд процессора;
C = {int_1, int_2, …, int_i} – множество подключаемых интерфейсов.

Помимо процессоров, выполняющих математические вычисления, к процессорным объектам относятся и различные контроллеры, такие как контроллеры распределенной памяти, отвечающие за создание и управление информационными потоками между объектами и внешней памятью макрообъекта. Контроллеры имеют схожую структуру и так же, как и процессор, работают под управлением системы команд, однако основной их функцией являются не математические расчеты, а генерация динамических обращений к памяти

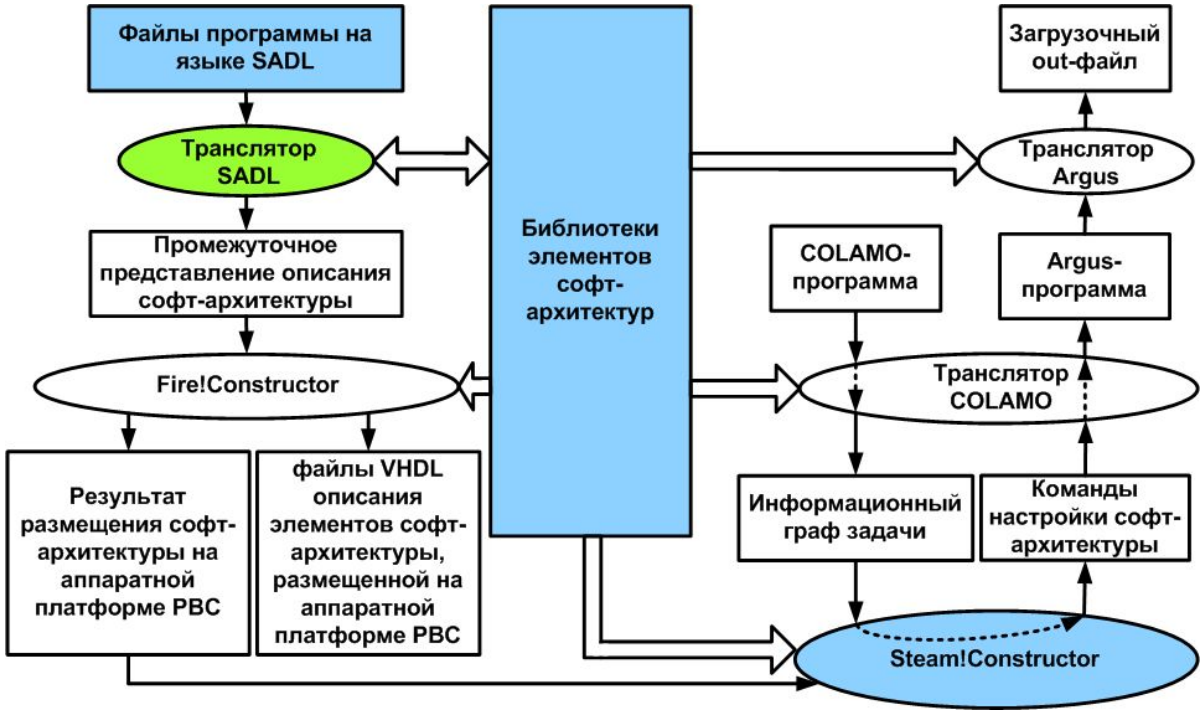
Реализация механизмов распараллеливания и каскадирования конструкций языка описания софт-архитектур.





Введение дополнительных конструкций в язык описания софт-архитектур позволит создавать вычислительные структуры из однотипных элементов, что сократит время описания софт-архитектуры за счет сокращения программного кода и возможных ошибок архитектура, неизбежно возникающих при использовании большого количества однотипных элементов.

Разработка библиотеки элементов и софт-архитектуры для решения задач математической физики.



1. Определение перечня объектов для решения задач предметной области.
2. Разработка схемотехнического описания объектов при помощи стандартных средств разработки.
3. Разработка описания объектов при помощи языка программирования софт-архитектур.
4. Размещение полученных описаний в библиотеке элементов софт-архитектур.
5. Объединение объектов в функционально законченное устройство – макрообъект при помощи языка программирования софт-архитектур.
6. Объединение нескольких макрообъектов в софт-архитектуру при помощи языка программирования софт-архитектур.
7. Трансляция описания софт-архитектуры в промежуточное представление, используемое компонентами системного программного обеспечения.
8. Размещение элементов софт-архитектуры на аппаратной платформе.

Оптимизация вычислительной структуры прикладной задачи

Области исследований и разработки:

–декомпозиция графа;

–модификация смежных комбинационных схем;

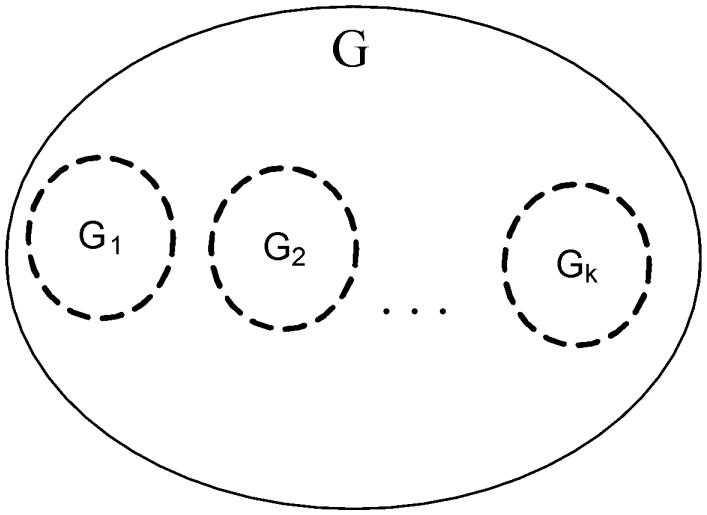
–оптимизация функциональных элементов на основе таблиц истинности;

–поиск и объединение функциональных элементов с однородными операндами;

–разработка графической оболочки для визуализации софт-архитектур;

–разработка библиотеки элементов и софт-архитектуры для решения задач математической физики.

Декомпозиция графа



Необходимо произвести декомпозицию данного графа на k подграфов так, чтобы k было минимальным:

$$G = \bigcup_{i=1}^{k} G_i, k \to \min$$

Подграфы G_i не могут представлять собой пустые множества

$$G_i \neq \varnothing, \forall\, i \in 1,2,...,k$$

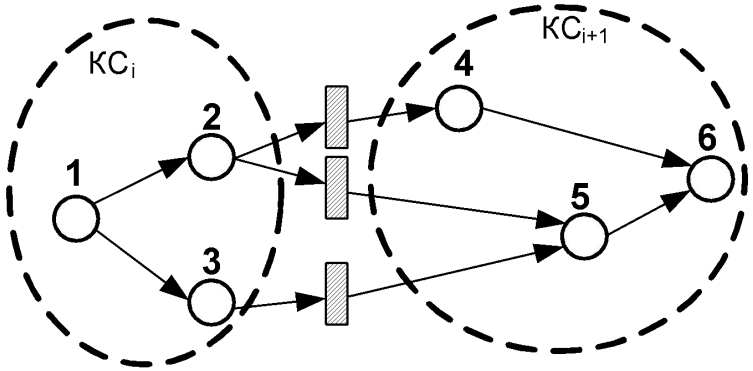
Пересечение двух любых подграфов является пустым множеством:

$$G_i \cap G_j = \varnothing, \ \forall\, i, j \in 1,2,...,k; \ i \neq j$$

Задача модификации смежных КС

Модификация смежных комбинационных схем, приводящая к улучшению ЦФ.

$$ЦФ = \sum_{i=1}^{k} US_i,\ ЦФ \to \min$$



$$US_i = \frac{k_i + p_i}{v_i + p_i},$$

где k_i – количество «внутренних» связей;

p_i – количество «внешних» связей;

v_i – количество вершин.

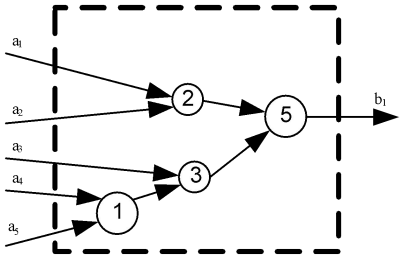
Задача расчета таблицы истинности для функционального элемента

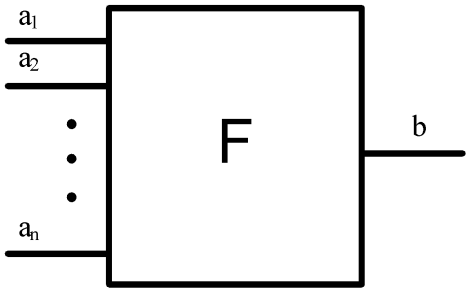
Проанализировать таблицу истинности на:
- тождественное вырождение;
- избыточность.

F(a, b, c) => F(c, a, b)

a	b	c	F
0	0	0	F_1
0	0	1	F_2
0	1	0	F_3
0	1	1	F_4
1	0	0	F_5
1	0	1	F_6
1	1	0	F_7
1	1	1	F_8

c	a	b	F
0	0	0	F_1
0	0	1	F_3
0	1	0	F_5
0	1	1	F_7
1	0	0	F_2
1	0	1	F_4
1	1	0	F_6
1	1	1	F_8





Задача поиска и объединения функциональных элементов с однородными операндами

Параметры поиска

- По количеству параметров;
- По количеству совпадающих параметров;
- По количеству отличающихся параметров;
- По комбинации признаков.