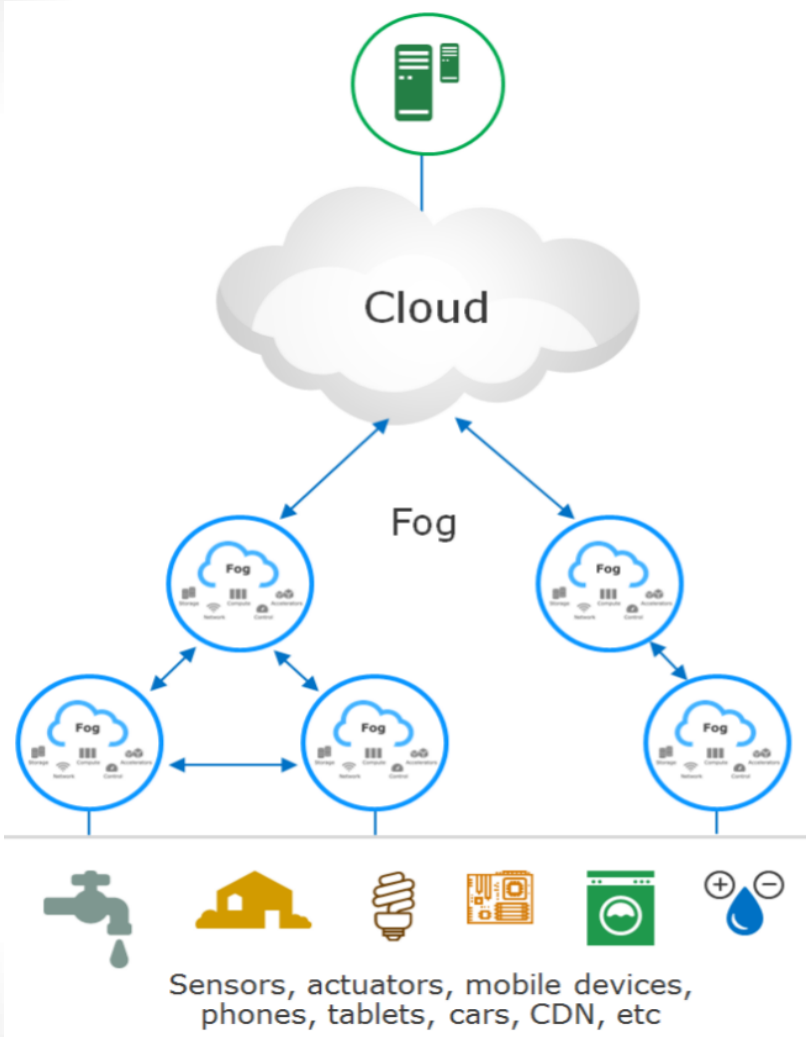
Н.А. Лукин
А.Ю. Филимонов
В.Н. Тришин

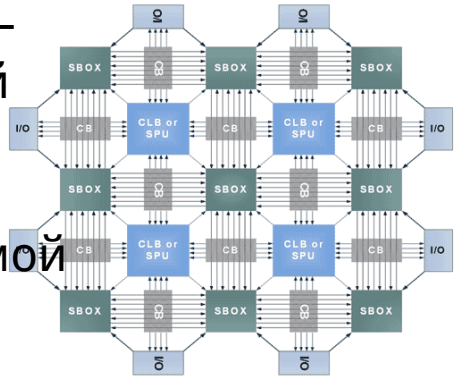
Облачная среда программирования однородных вычислительных систем

Уральский Федеральный Университет
Институт Машиноведения УрО РАН
г. Екатеринбург

Облачные сервисы, fog computing



- 2015г. Инициатива OpenFog *
- 2016г. поручение президента РФ о подготовке инфраструктуры для технологии «туманных вычислений» (fog computing), которая необходима для дальнейшего развития «интернета вещей» **
  IoT процессоры:
- Michigan Micro Mote (8-bit CPU, a 52x40-bit DMEM, a 64x10-bit IMEM, a 64x10-bit IROM, 0.1 МГц)
- OLEAT222-1005 – автомобильный контроллер с блоком программируемой логики



* www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf

** regnum.ru/news/economy/2151919.html

Задачи, решаемые встроенными ВС IoT

- **Вычисления:**

- навигация (ориентация в пространстве);
- обработка сигналов и изображений (распознавание образов)

- **Управление:**

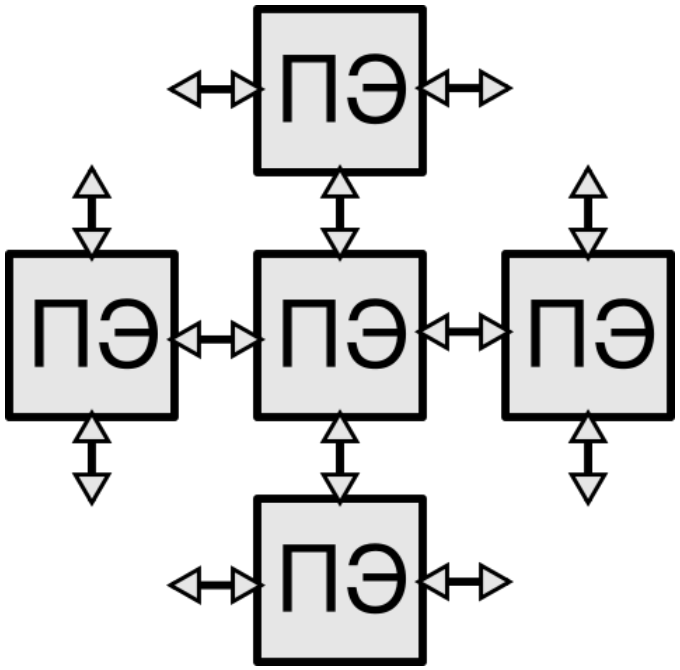
- адаптивные интерфейсы для беспроводных сетей комуникации;
- принятие решений в условиях неопределенности

Требования к встроенным ВС:

- **Производительность — порядка 10^{10} оп/сек**
- **Отказоустойчивость — исправная обработка данных при множественных отказах**
- **Габариты и масса — один кристалл СБИС на всю ВС**

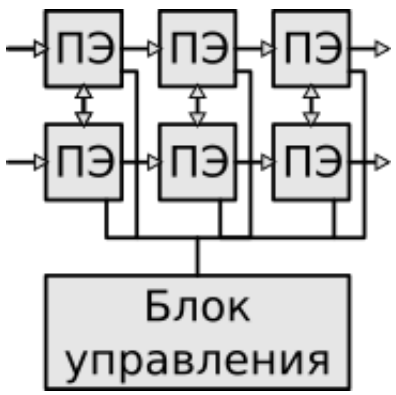
Функционально-ориентированные процессоры на основе ОВС — основные вычислительные модули встроенных ВС для IoT

ОВС – общие понятия

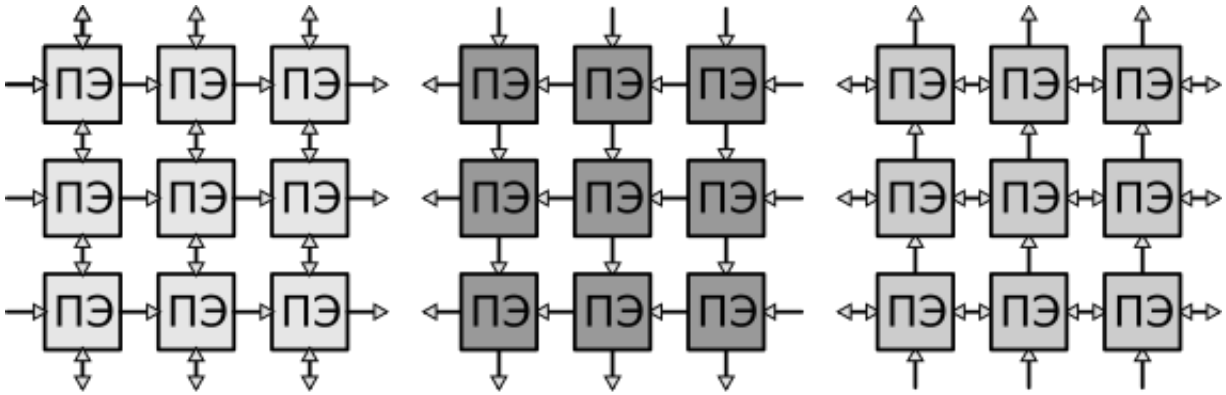


Особенности ОВС:
- Идентичность ПЭ
- Регулярность связей
- ПЭ:
  - простая (примитивная) структура;
  - малая разрядность – 1, 4, 8
- Типы архитектур – SIMD, MIMD, SIMD + MIMD
- Возможность глобального распараллеливания вычислений

SIMD-ОВС. Принципы работы

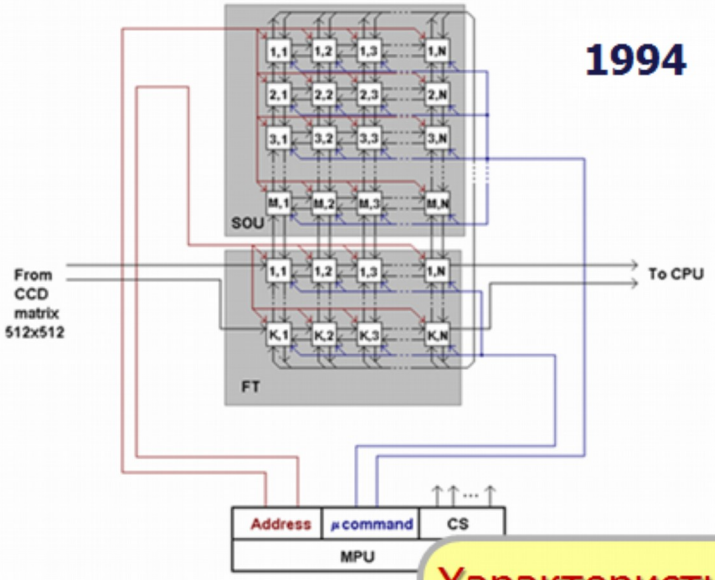


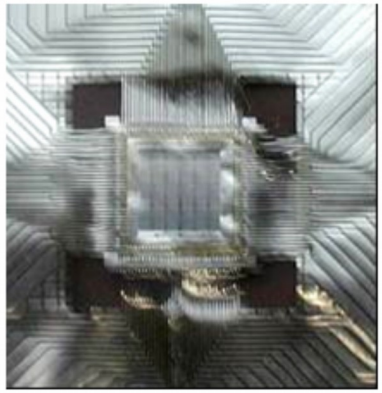
- Кол-во ПЭ связано с размерностью массивов входных данных
- Взаимосвязь между ПЭ отражает структуру массивов данных
- Один поток команд: в любом такте работы ОВС все ПЭ выполняют одну и ту же команду
  Один алгоритм множество операций
  множество данных

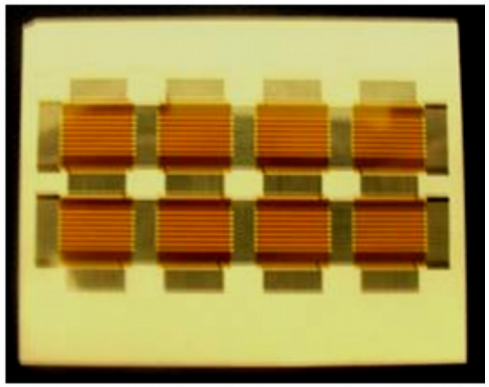


SIMD-ОВС. Физическая реализация

ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СРЕДЫ (SIMD): проект GAPP







Характеристики

Систолический процессор

СБИС СОУ: f = 10 MHz; 64 ПЭ; W = 35 mW

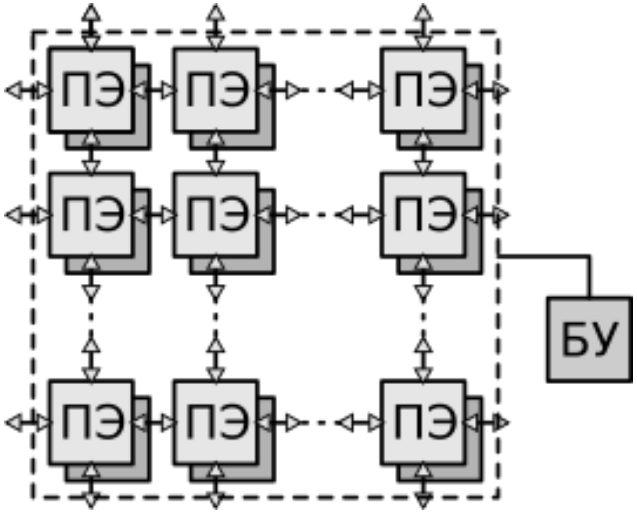
Видео ФОП: S = 128×128 пикс; M = 0.1 кг; T = 3 мс

Основная задача

Межкадровая обработка изображений

Функционально-ориентированные процессоры

MIMD — ОВС. Принципы работы

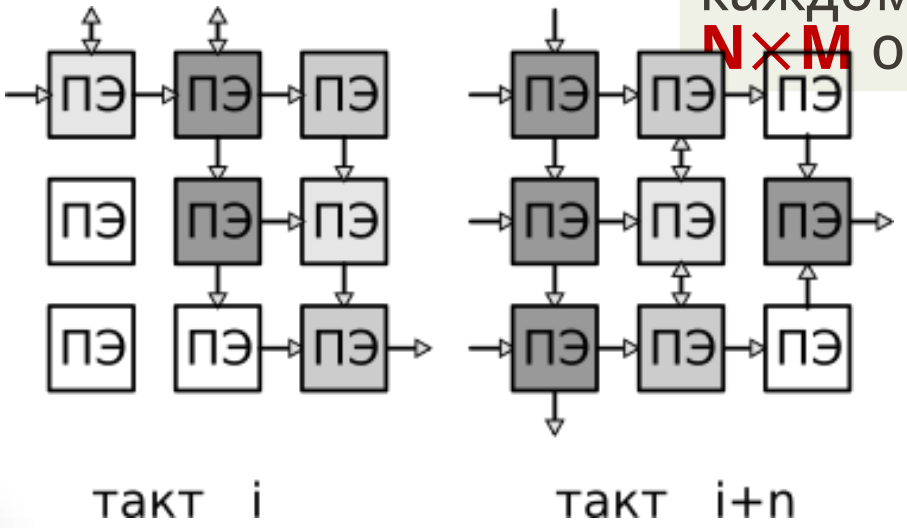


Особенности:

Количество ПЭ связано с числом **вершин** графа алгоритма

Архитектура ОВС отображает **граф** алгоритма

Множество потоков команд: в каждом такте выполняется **N×M** операций

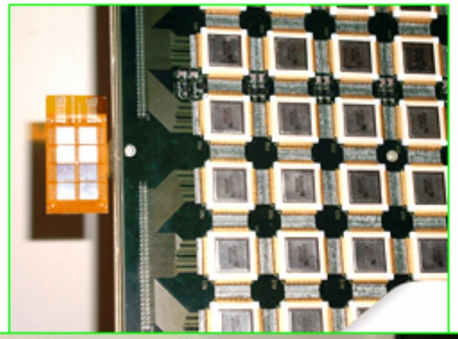


Один алгоритм, множество операций, множество данных

MIMD-ОВС. Физическая реализация

ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СРЕДЫ (MIMD): проект МИНИТЕРА

2004



Характеристики

Систолический процессор

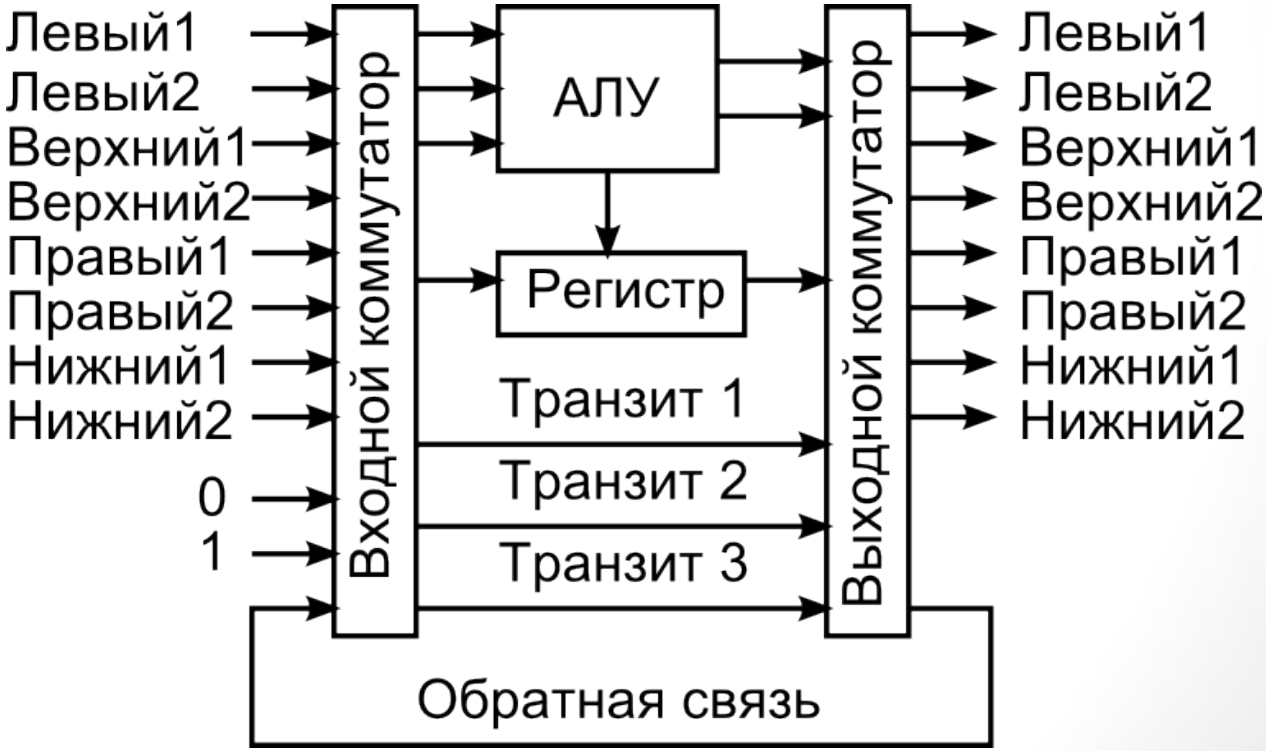
СБИС: 0.6 мкм; f = 50 MHz; 25 ПЭ

Рабочая станция: R = 20 Gglops

Функцио
ориентиро
процессоры

ПЭ ОВС Минитера 2

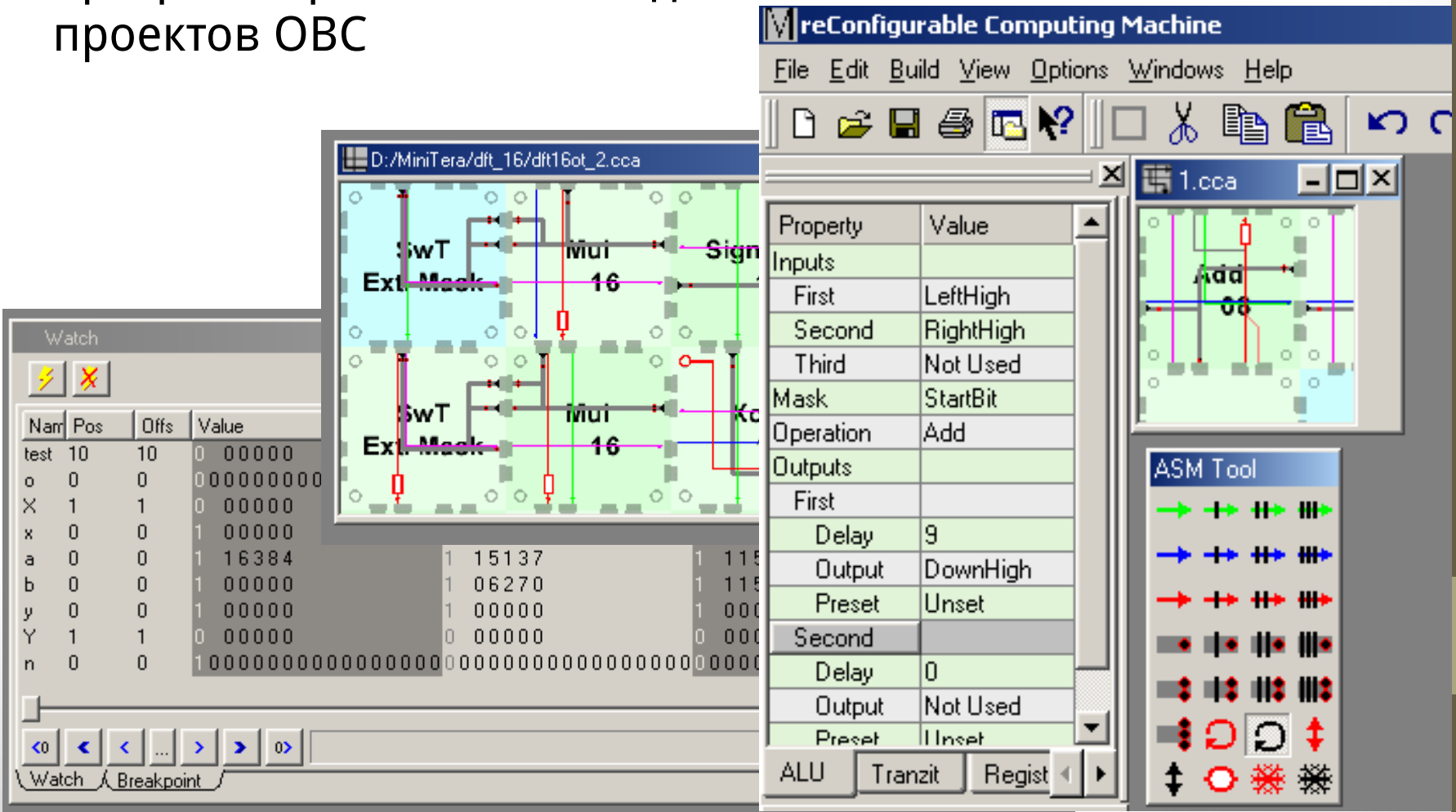
Основные блоки ПЭ:
- битовое АЛУ (49 инструкции)
- 64-битная регистровая память
- 3 транзитные линии



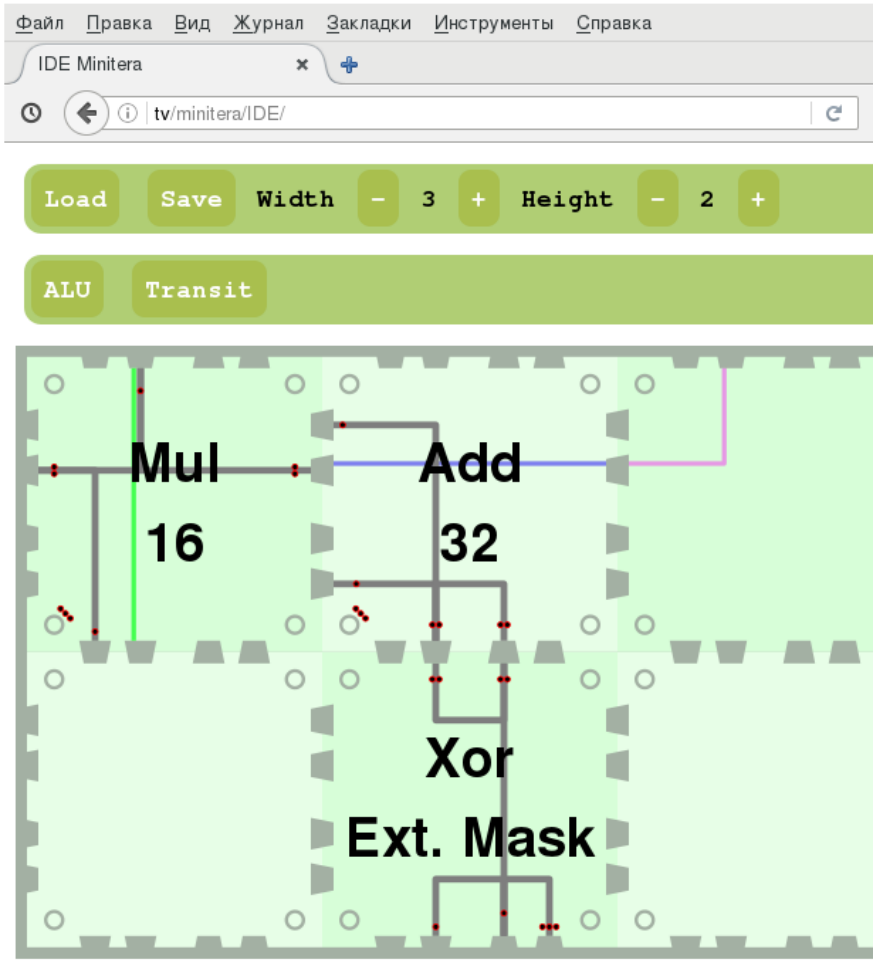
IDE Минитера 2

Графическое программирование и отладка проектов ОВС

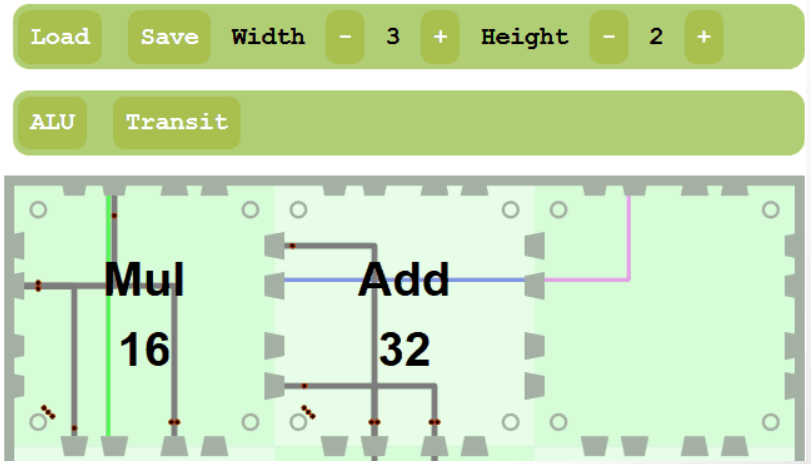
2002 г. Qt 3.0



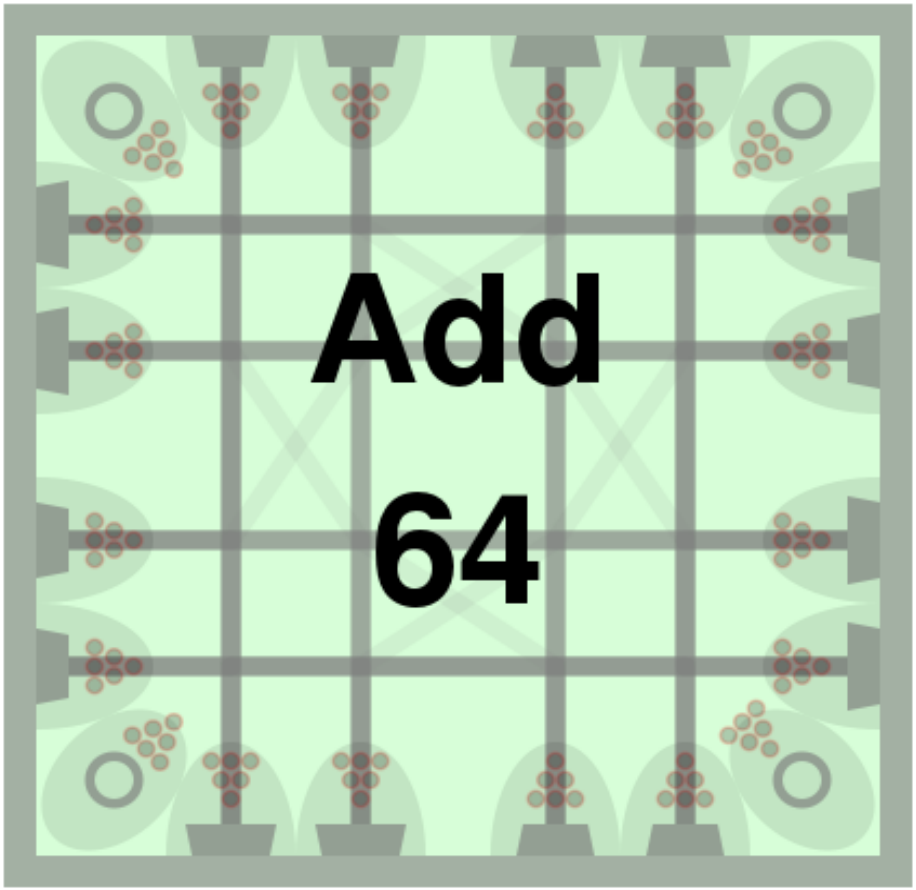
Окно облачной IDE



— редактор поля ПЭ позволяет:
- выбирать размер редактируемого макроса
- назначать выполняемые команды ПЭ и способы синхронизации
- выбирать используемые входы и выходы АЛУ и транзитных линий

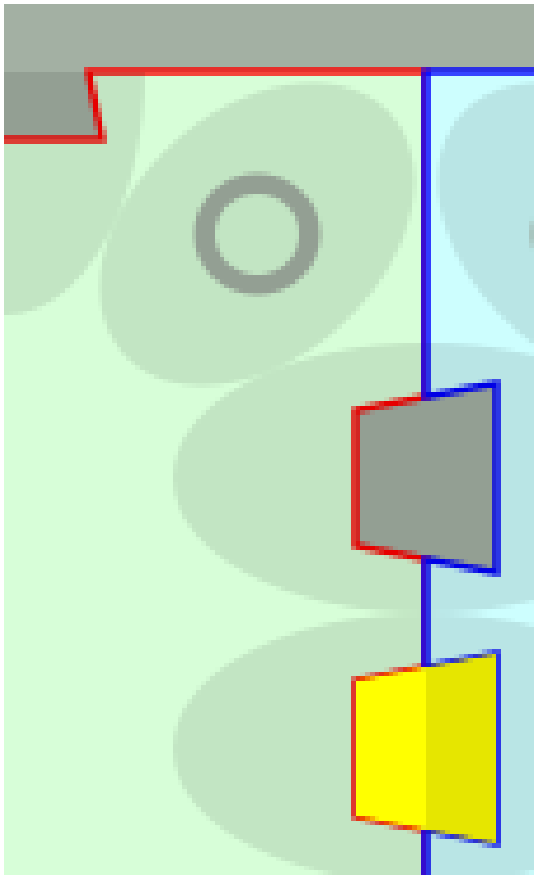


Библиотечный элемент SVG



содержит все необходимые условные обозначения видимость которых определяется таблицей стилей:
- фон
- символы портов
- все возможные линии, соединяющие 2 любых порта
- символы номера порта АЛУ
- имя выполняемой операции

Интерактивная подсветка



выполнена с использованием только таблицы стилей:
- для выделения порта достаточно лишь приблизить к нему курсор (темные овальные области)
- подсветка всего порта реализована

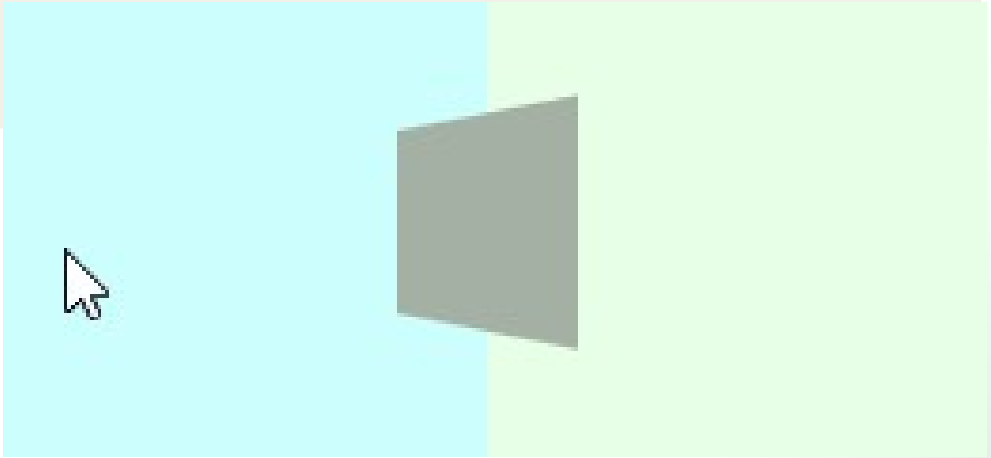
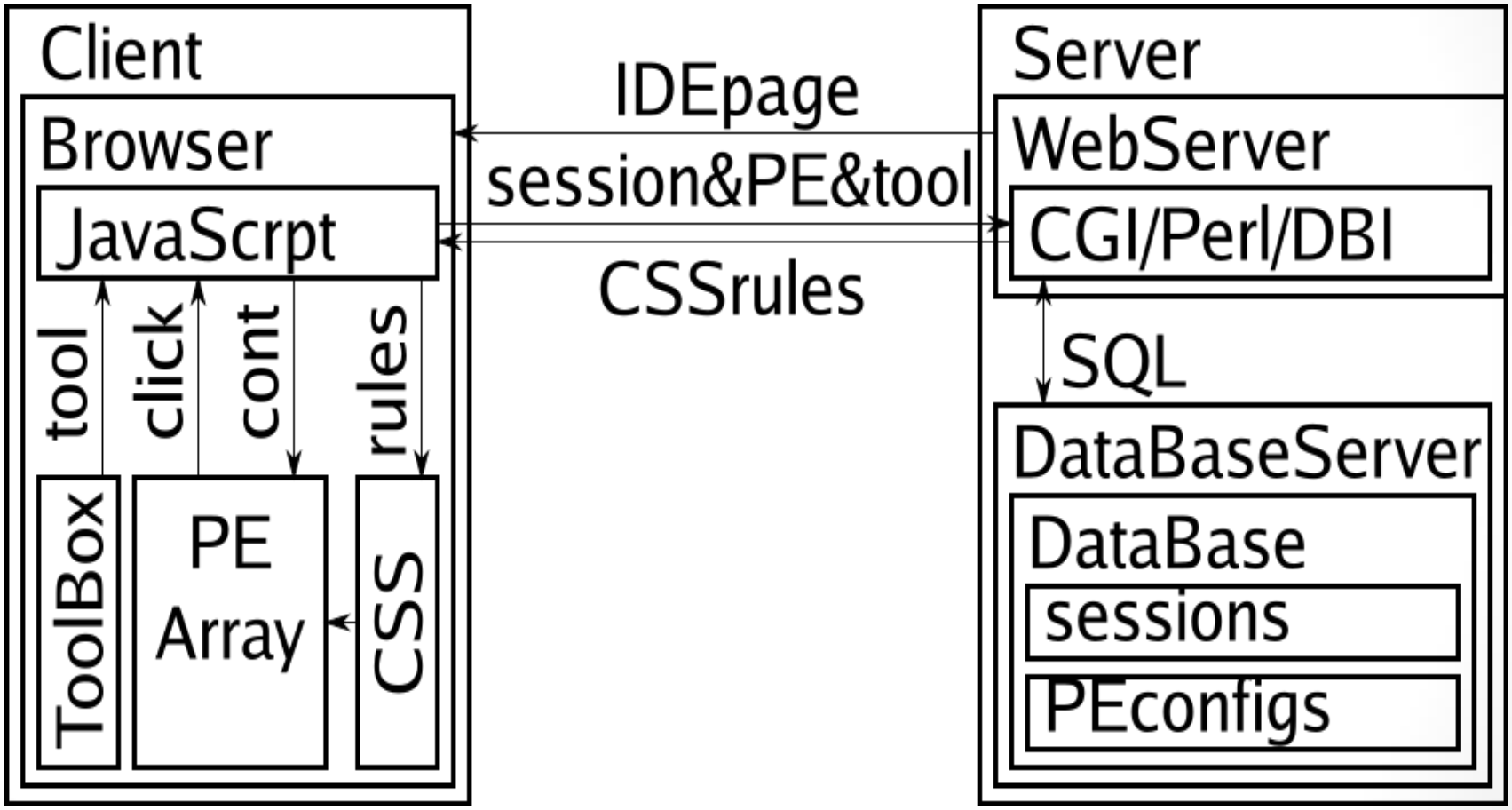
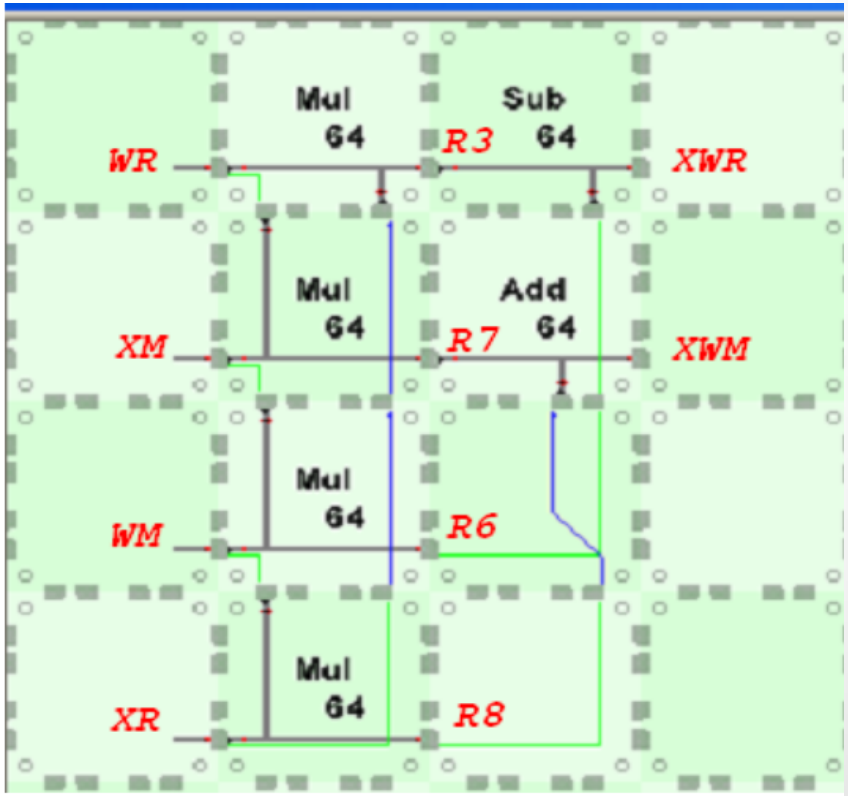


Схема взаимодействия клиент-сервер



Особенности программирования ФОП на базе ОВС

1. Управляющие конструкции языка обеспечивают не столько управление ВП, сколько согласование структур данных и построение маршрутов их передачи .
2. Результат программирования ОВС — Укладка и трассировка ПЭ
3. Стремление сократить время выполнения («высоту» программы -H)
   приводит к увеличению числа задействованных элементов («ширины программы»- L)
   - и наоборот.
4. Понятия «директива управления ВП» и переменная не находят отражения в ФОП.



Концепции построения вычислителей

	Классическая (Фон Нейман)	ОВС
Управление ВП	Поток управляющих операций Control Flow	Поток данных Data Flow
Архитектура *	SISD (SIMD,MIMD)	MIMD
Хранение данных	Общая память	Локальная память
Модель программирования ВП	Императивная	Функциональная (Декларативная) ???

*По таксономии Michael J. Flynn

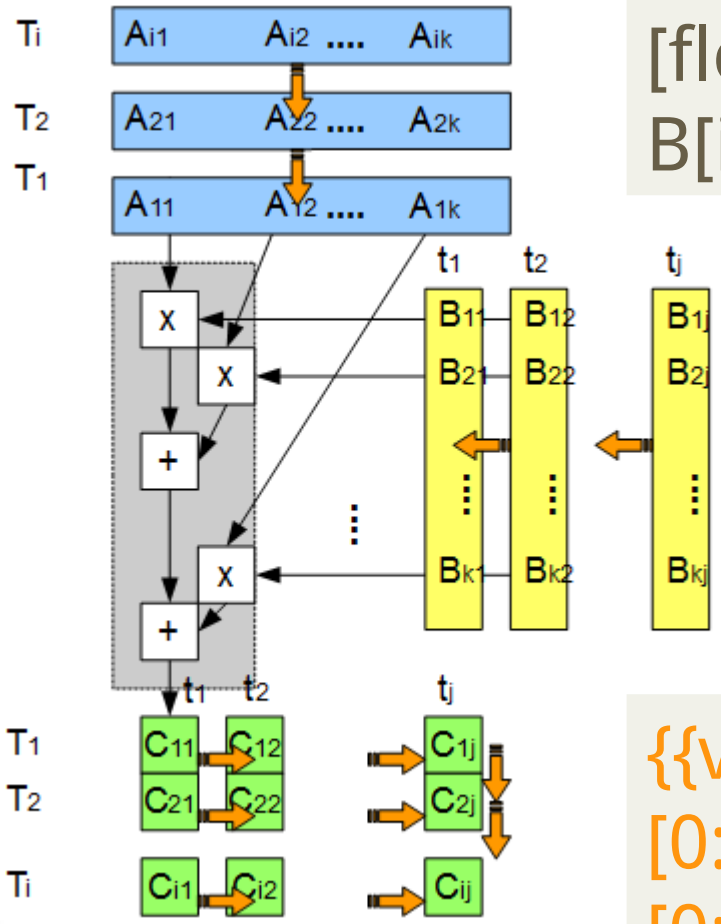
Языки, предназначенные для описания параллельных вычислений

	NESL	Sequencel
Разработчик	Guy E. Blelloch (Carnegie Mellon University)	Daniel E. Cooke, J. Nelson Rushton Texas Multicore Technologies
Распространение	Свободное	Коммерческое
Последняя версия	V.3.1 (1995г.)	V.2.4(2016г.)
Базовый элемент	Последовательность*	Последовательность
Организация параллельных вычислений	Встроенные параллельные функции Векторизация** скалярных функций	Встроенные параллельные функции Комплекс NTD
Сайт	http://www.cs.cmu.edu/~scandal/nesl.html	http://texasmulticore.com/

* a — последовательность 0-го порядка (скаляр);
[a1,a2, ... an] — последовательность 1 -го порядка (вектор) ;
[[a1,a2, ... an], ..[b1,b2, ...bn]]— последовательность 2 -го порядка (матрица);

** ?F(A) | A = [a1,a2, ... an] =>{F(a): a in A} =>[F(a1),F(a2), ... F(an)];// «конвейер»

NESL:Умножение матриц в базисе умножения векторов




```
function vmulv (a,b) :([float],
[float])-> float = sum({A[i] *
B[i]: i in [0:#a]});//аппаратно
```

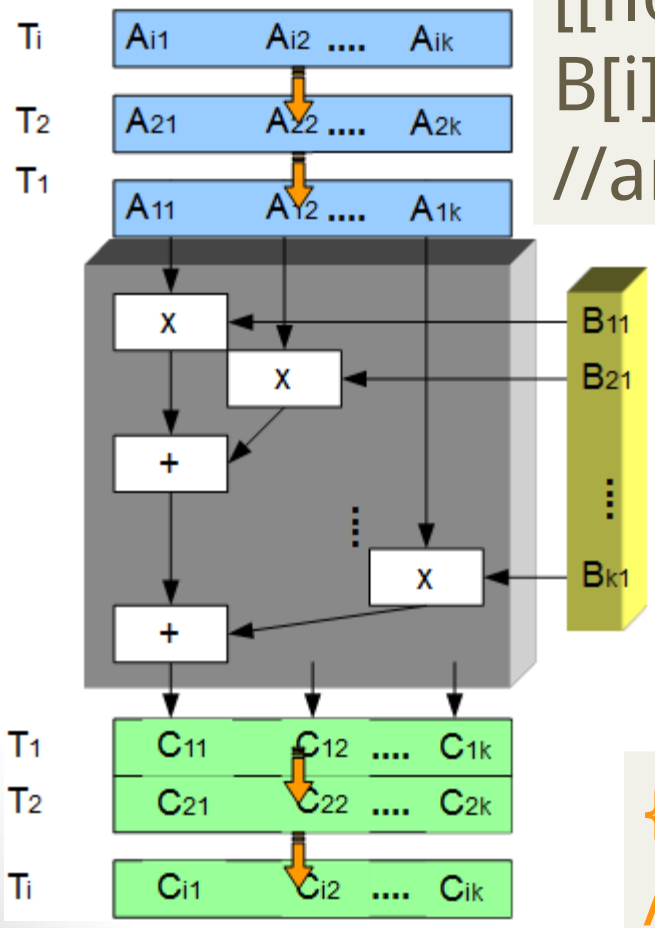
L~число стлб. ||A||

H~число стлб.||B||
* число стр. ||A||

```
{{vmulv(a[i],{b[k][j]: k in
[0:#a[0]]}):j in [0:#b[1]]}: i in
[0:#a]};//программно
```


NESL:Умножение матриц в базисе умножения вектора на матрицу

```
function vmulmat (a,b) :([float],
[[float]])-> [float] = {sum({A[i] *
B[i][k]: i in [0:#a]}): k in [0:#b[1]]};
//аппаратно
```



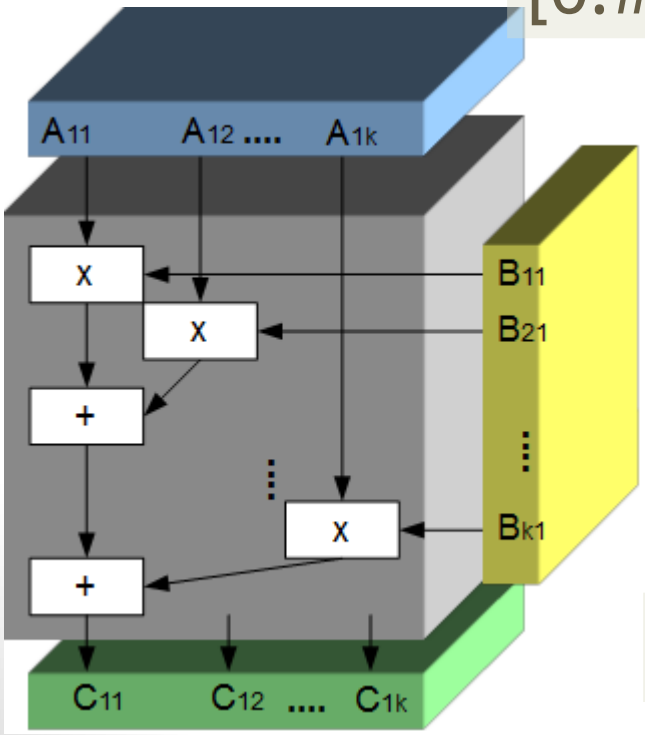
L~число стлб. ||A||
* число стлб. ||B||

H~число стр.||A||

```
{vmulmat(a[r],b):r in [0:#a]};
//программно
```


NESL:Умножение матриц в базисе умножения матриц

```
function matmul (a,b) :([[float]],
[[float]])-> [[float]] = {{sum({A[r][i] *
B[i][k]: i in [0:#a[1]]}): k in
[0:#b[1]] }:r in [0:#a]};//аппаратно
```

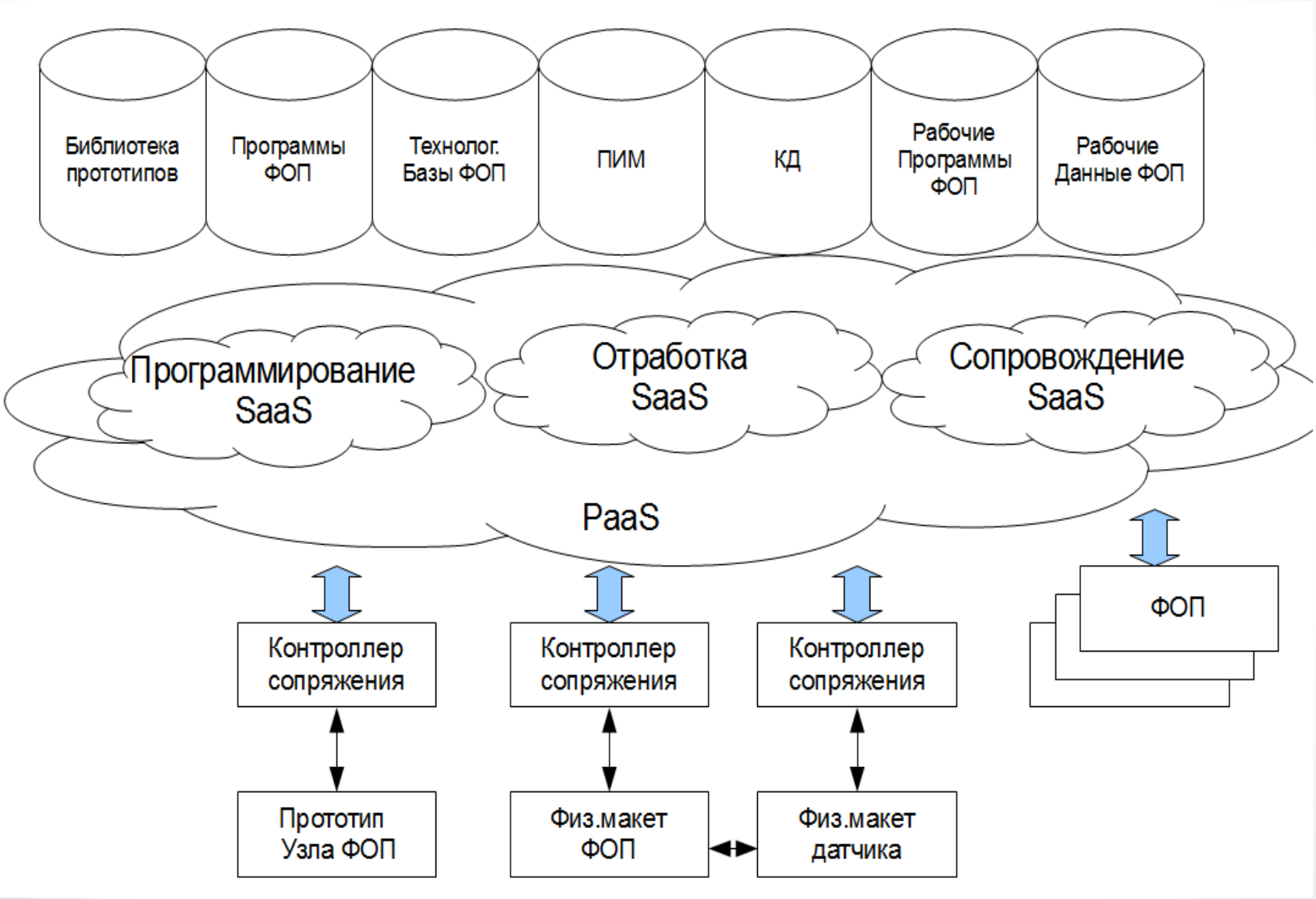


L~число стлб. ||A||
* число стлб. ||B||
* число стр.||A||

H =1;

```
matmul (A,B);//программно
```

Платформа программирования потоковых ФОП



Заключение

1. Процесс программирования ОВС имеет ряд особенностей, которые ограничивают возможности применение традиционных (императивных) языков программирования
2. По своей природе функциональные языки больше приспособлены для программирования ОВС, но их применение сдерживается недостаточным развитием универсальных программных платформ
3. Специальные программные платформы, основанные на ФЯП позволяют решить большинство проблем, связанных с программированием ОВС