ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
Национальный исследовательский университет «Высшая школа экономики»

Московский государственный институт электроники и математики

Департамент компьютерной инженерии

Вычислительная система, управляемая потоком данных

Методические указания
к лабораторным работам по куру
«Математическое и имитационное моделирование»

Москва 2016

Составитель С.М. Салибекян

Вычислительная система, управляемая потоком данных: Метод. указания к Лабораторным работам по курсу «Математическое и имитационное моделирование» / Моск. гос. ин-т электроники и математики Национального исследовательского университета «Высшая школа экономики»; Сост. С.М. Салибекян. М., 2016 – 29 с.

Библиогр.: 7 назв.

Аннотация

Методические указания призваны познакомить студентов с приёмами проектирования и моделирования вычислительных систем с управлением потоком данных (dataflow). Пособие даёт как теоретические знания (представление алгоритмов в виде графов, принципы работы вычислительных систем, управляемых потоком данных), так и знания практические (проектирование конкретной вычислительной системы с управлением потоком данных, моделирование системы с целью определения ее характеристик и оптимальных параметров). Предназначается для студентов бакалавриата 3-го курса дневной формы обучения.

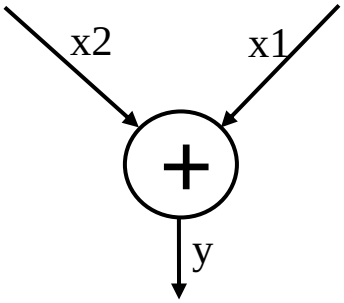
ISBN

Цель работы

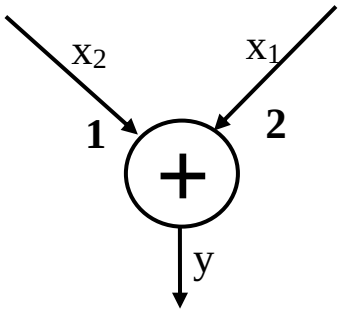
Целью данной курсовой работы является отработка навыков моделирования вычислительных систем с целью оценки их характеристик и выбора оптимальных параметров, и отработка навыков анализа алгоритмов с помощью графовой модели.

Теоретические сведения

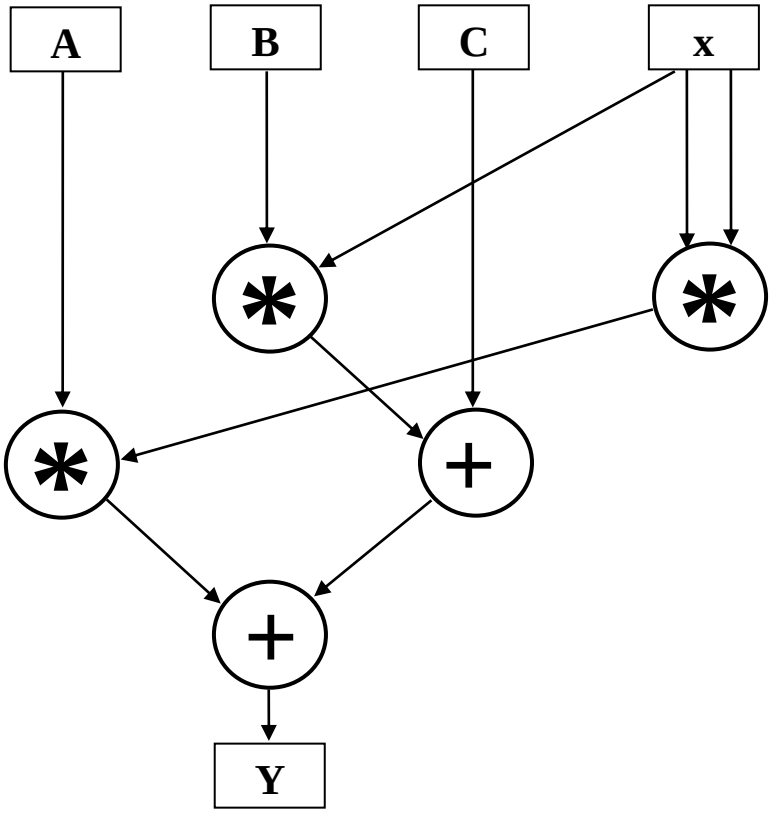
Любой вычислительный процесс можно представить с помощью ориентированного графа, приписав его вершинам вычислительные операции (арифметические, логические и т.д.), а дугам - операнды (данные) этих операций. Например, операцию сложения можно представить так:



Входящие в вершину дуги – входные данные для операции, выходящая дуга – результат выполнения операции (в данном случае сложения). Некоторые операции, например, вычитание, чувствительны к порядку следования операндов: если поменять местами вычитаемое и уменьшаемое, то результат операции будет другим. Поэтому для подобных операций входящие дуги нумеруются: цифра 1 обозначает первый операнд, а цифра 2 – второй:



Результат выполнения операции подается в качестве операнда на одну или сразу на несколько других операций. Таким образом можно описать любой алгоритм. Исходные данные для алгоритма будем обозначать прямоугольниками. Например, арифметическое выражение $y=ax^2+bx+c$ посредством графовой модели можно представить так:



При реализации алгоритма вычислительной системой есть два глобальных подхода: управление командами и управление данными. Первый подход считается классическим - он используется в большинстве современных компьютеров. Заключается он в том, что вычислительный процесс управляется потоком команд: процессор принимает коды,

описывающие команды, интерпретирует их и выполняет. Получается, что для ВС описываются узлы графа алгоритма. Так, типичная команда подобной ВС имеет следующий формат:

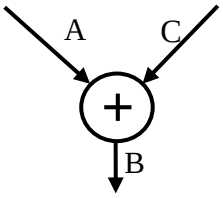
КОП	Операнд 1	Операнд 2	Результат

где КОП - код операции (код, который обозначает, какую операцию надо сделать);

первый операнд, второй операнд - указание, из каких ячеек памяти следует брать операнды;

результат - указание, куда следует поместить результат выполнения операции.

Данная конструкция описывается узлом графа алгоритма, в который входит два операнда, и выходит один результат:

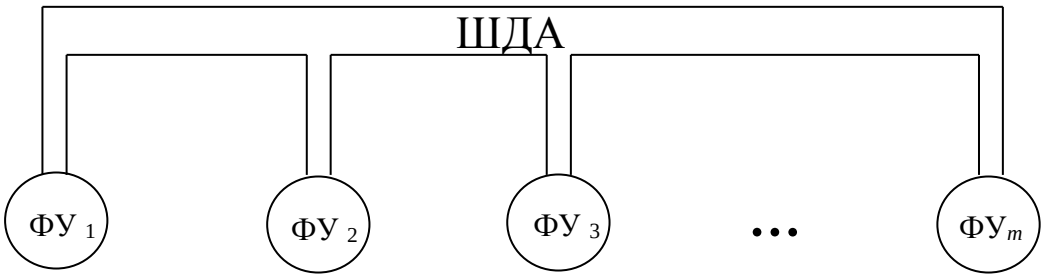


Таким образом, первое поле описывает узел графовой модели, а три других - дуги.

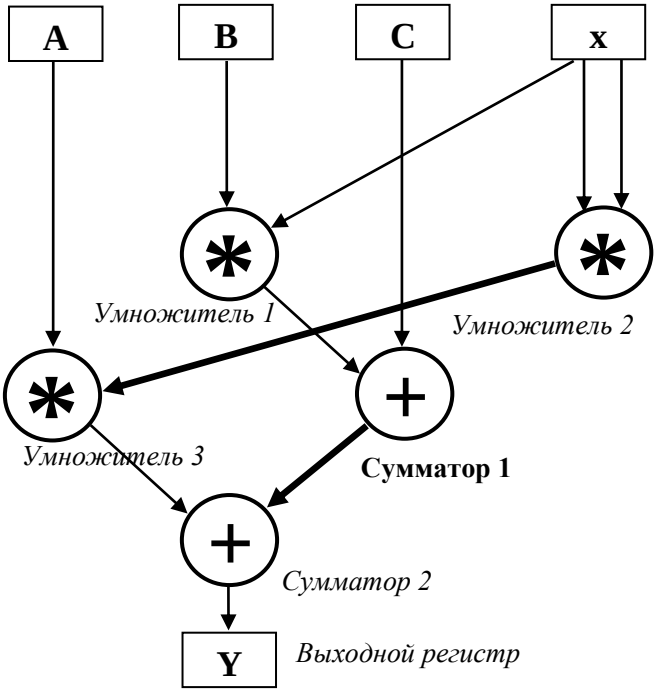
В системах, управляемых данными, акцент делается не на описание команд, а на описание данных, т.е. дуг графа алгоритма. Здесь кодируются не команды, а данные, которые выступают в качестве операндов. Пересылка данных между вычислительными устройствами осуществляется посредством токенов (токен представляет собой один операнд, снабженный служебной информацией). В данном курсовом проекте будет производиться проектирование ВС, управляемой данными.

Проектируемая ВС построена по следующему принципу. Все данные, передаваемые между функциональными устройствами (ФУ), снабжены атрибутом, идентифицирующим эти данные. Все ФУ (сумматоры, умножители, блоки работы с плавающей точкой и т.д.) объединены между собой двумя шиной данных-атрибута (ШДА), по которой происходит передача токена (милликоманды). В состав ФУ входят внутренние регистры (контекст ФУ) для хранения промежуточных данных (так, в состав ФУ «Сумматор» будут входить три внутренних регистра: регистры 1-го и 2-го операндов и регистр результата вычисления). Атрибут милликоманды содержит два поля: первое - код ФУ, которому данные предназначаются; второе - описание данных, где указывается, как ФУ должно эти данные интерпретировать: например, для сумматора указывается, что данные являются 1-м или 2-м слагаемым (если необходимо произвести сложение) или вычитаемым, или уменьшаемым (если необходимо произвести вычитание) и т.д. ФУ, входящие в состав ВС могут не только принимать и обрабатывать данные, но и выдавать на ШДА результат своей работы. Для этого в контекст ФУ включаются регистры, где хранятся атрибуты, из которых формируется милликоманда с результатом. Милликоманда выдается на ШДА. Благодаря этому в ВС реализуется распределенное управление (т.е. нет блока, который централизованно осуществляет планирование вычислений в ВС), что позволяет ВС осуществлять самораспараллеливание вычислений и самостоятельно выбирать наиболее оптимальный путь решения задачи. Однако распределенное управление требует больших аппаратных затрат, ведь практически на каждую операцию нужно выделять отдельное ФУ. Так, для алгоритма вычисления квадратного многочлена, приведенного выше, потребуется пять ФУ: три умножителя и два сумматора.

Описание каждой команды включает в себя атрибуты операндов, и атрибуты, которые необходимо «прикреплять» к результатам вычисления, когда они будут передаваться по ШДА.



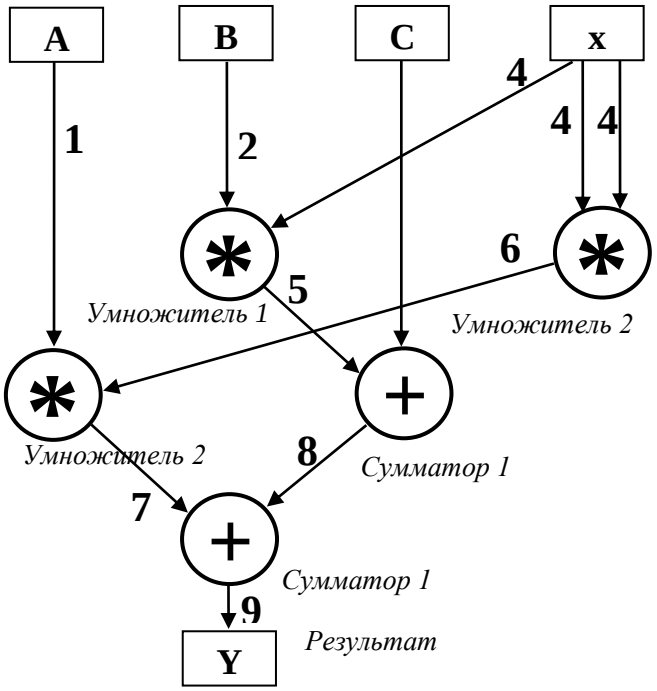
Перед началом работы ФУ следует запрограммировать. Так, для алгоритма вычисления квадратного трехчлена надо задействовать 5 ФУ: 3 умножителя и 2 сумматора.



Однако количество ФУ можно сократить. «Умножитель 2» и «Умножитель 3» всегда будут работать последовательно, т.к. между ними есть непосредственная связь по данным (когда результат вычисления одного ФУ передается в качестве операнда на другой ФУ). Так как эти устройства однотипны, две вычислительные операции можно произвести

на одном ФУ без уменьшения времени выполнения вычислений. Так же в одном ФУ можно реализовать последовательные операции «Сумматор 1» и «Сумматор2». Таким образом, ВС для реализации данного алгоритма понадобятся всего трех ФУ.

Для программирования ВС каждому операнду, который станет передаваться по ШДА, следует присвоить свой уникальный идентификатор, т.е. приписать номер каждой стрелке на потоковом графе программы.

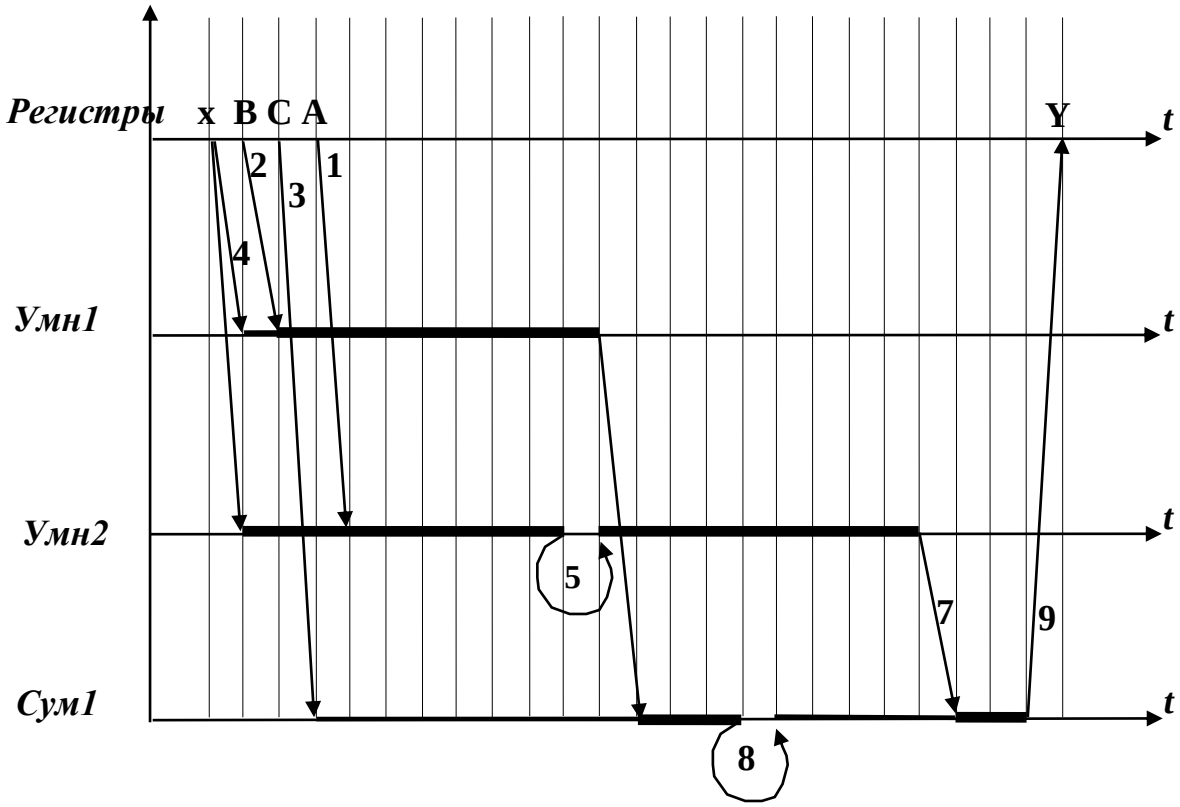


После можно составить таблицу программы для всех ФУ:

Наименование ФУ	Ярлыки операндов	Выполняемая операция	Комментарий	Ярлык выходных данных
Умножитель 1	2,4	*	B*x	5
Умножитель 2	4,4	*	x^2	6
	1,6	*	A* x^2	7
Сумматор 1	3,5	+	B*x+c	8

	7,8	+	A* x^{2+}B*x+c	9

Для анализа работы ВС используется диаграмма Ганта, которая служит для визуализации поведения параллельно работающих вычислительных устройств во времени. Графики работы каждого ФУ располагаются друг под другом. А по оси "x" откладывается время. Пример приведен на рисунке. Передача данных между ФУ обозначается стрелкой.



В данной системе время пересылки – 1 такт, поэтому-то линии, обозначающие передачу, здесь наклонные, что иллюстрирует задержку

пересылки данных между ФУ. Время работы сумматора – 2 такта, а умножителя – 9 тактов.

Существенное преимущество ВС с распределенным управлением – самораспараллеливание алгоритма и самовыбор одного из возможных путей получения конечного результата. Например, вычислить площадь треугольника можно сразу несколькими способами: по двум сторонам и углу, по стороне и двум углам, по трем сторонам (формула Герона) и т.д. По какой ветке алгоритм найдет площадь треугольника, зависит от исходных данных и длины ветви алгоритма: если вычисления по одной из ветвей меньше и для вычислений имеются все необходимые данные, то вычислительный процесс по ней быстрее дойдет до получения окончательного результата.

Для вычисления коэффициента параллелизма (S) следует подсчитать количество параллельно работающих устройств в каждом такте и разделить полученное число на количество тактов работы алгоритма. Данный коэффициент показывает, сколько в среднем в единицу времени ФУ работают параллельно. Если S=1, то ВС работала в последовательном режиме; если S=2, то в среднем в ВС параллельно работало два ФУ и т.д. Для вычислительной системы, рассмотренной выше, S=32/24=4/3=1,33.

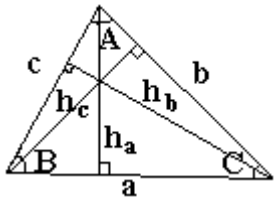
Практическая часть

Задание.

Необходимо разработать ВС с централизованным управлением и ВС распределенным управлением для решения одной из геометрических задач (варианты 1-9).

Варианты:

1. Площадь треугольника,

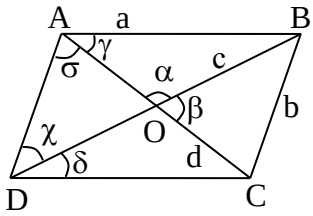


где a, b, c - стороны треугольника;

A, B, C - углы треугольника;

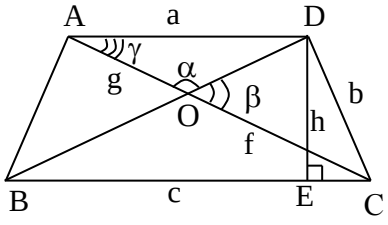
h_a, h_b, h_c - высоты треугольника.

2. Площадь параллелепипеда,



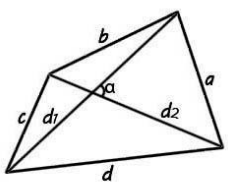
где a,b – стороны параллелепипеда;
c,d – диагонали параллелепипеда;
α,β – углы, образованные пересечением диагоналей;
γ, σ, χ, δ - углы, образованные стороной и диагональю.

3. Площадь равнобокой трапеции,



где a,c – основания;
b – боковая грань;
h – высота;
f,g – отрезок от угла трапеции до пересечения диагоналей;
α,β – углы пересечения диагоналей;
γ – угол пересечения основания и диагонали.

4. Площадь четырехугольника

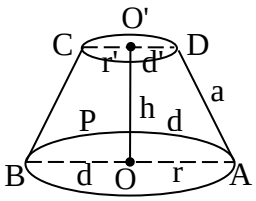


a,b,c,d – стороны четырехугольника;

α – острый угол между диагоналями.

5. Площадь поверхности усеченного конуса,

где a – боковая грань;



r,d – радиус и диаметр основания;

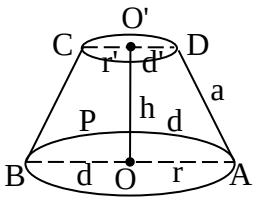
r',d' – радиус и диаметр верхней части пирамиды;

h – высота;

P, P' – периметр основания и верхней части;

S, s – площади основания и верхней части пирамиды.

6. Объем усеченного конуса,



r,d – радиус и диаметр основания;

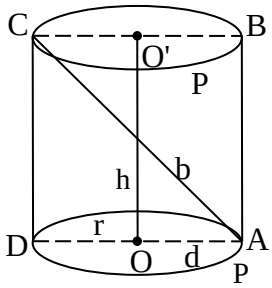
r',d' – радиус и диаметр верхней части пирамиды;

h – высота;

P, P' – периметр основания и верхней части;

S, s – площади основания и верхней части пирамиды.

7. Площадь поверхности цилиндра,



где r,d – радиус и диаметр основания;

h – высота;

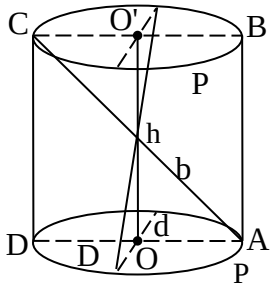
b – диагональ цилиндра;

P – периметр основания;

S – площадь основания.

8. Объем цилиндра,

9. Площадь поверхности цилиндра в основании которого лежит эллипс



r – малый радиус эллипса;

R – большой радиус эллипса;

h – высота;

b – малая диагональ цилиндра;

B – большая диагональ цилиндра.

P – периметр основания;

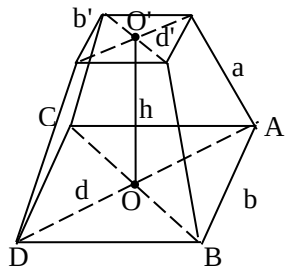
S – площадь основания.

10. Объем цилиндра в основании которого лежит эллипс

11. Площадь поверхности усеченной пирамиды с квадратом в основании,

h – высота;

a – боковая грань;



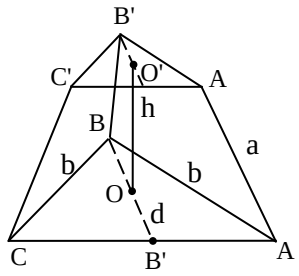
b – грань квадрата, лежащего в основании;

d – диагональ квадрата, лежащего в основании;

S – площадь квадрата, лежащего в основании пирамиды.

12. Объем поверхности усеченной пирамиды с квадратом в основании,

13. Площадь поверхности усеченной пирамиды с треугольником в основании.



h – высота;

a – боковая грань;

b,c – грани треугольника, лежащего в основании;

d – медиана треугольника, лежащего в основании;

S – площадь треугольника, лежащего в основании пирамиды.

14. Объем поверхности усеченной пирамиды с треугольником в основании.

После построения потокового графа алгоритма преподавателем задаются времена работы ФУ и задержка передачи данных по ШДА и моменты прихода входных данных в модельном времени.

Этапы выполнения работы:

Л1. Найти несколько формул (не менее 3-х) для решения поставленной геометрической задачи. Начертить графы алгоритмов, описывающие все формулы нахождения площади треугольника.

Л2. Составить общий граф программы по всем формулам. Составить таблицу программ для ФУ.

Л3. Ознакомление со ОА-средой программирования и моделирования

Л4. Составить и отладить вычислительную программу в среде ОА-программирования и моделирования.

Л5. Дополнить программу для реализации имитационного моделирования ВС. Определить коэффициент параллелизма.

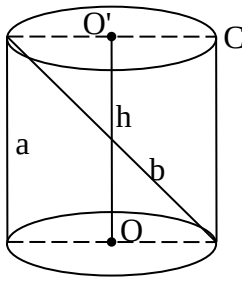
Л6. Выбрать оптимальное количество процессорных ядер для решения вычислительной задачи. Изменить время прихода данных или время вычислений для того, чтобы вычислительный процесс пошел по другой вычислительной ветке.

Пример выполнения работы

Исходные данные

Задание: найти объем цилиндра.

Объем цилиндра (задаются диаметр основания; периметр основания;



высота боковой грани; площадь основания, диагональ цилиндра),

где a – боковая грань;

r,d – радиус и диаметр основания;

P – периметр основания;

h – высота;

S – площадь основания;

b – диагональ цилиндра.

<u>Время поступления входных данных</u>

В 1-ый такт (τ1) приходит операнд b;

Во 2-ой такт (τ2) посылается h;

В 4-ый такт (τ4) посылается p;

На 7-ой такт (τ7) посылается S;

На 9-ый такт (τ9) посылается a;

В 11-ый такт (τ11) посылается r;

На 14-ый такт (τ14) посылается π;

На 16-ый такт (τ16) посылается 2;

На 17-ый такт (τ17) посылается 3;

На 19-ый такт (τ19) посылается 4.

Длительность работы ФУ:

Сумматор – 2 τ;

Умножитель – 9 τ;

Делитель – 10 τ

Тригонометрич. Блок – 10 τ;

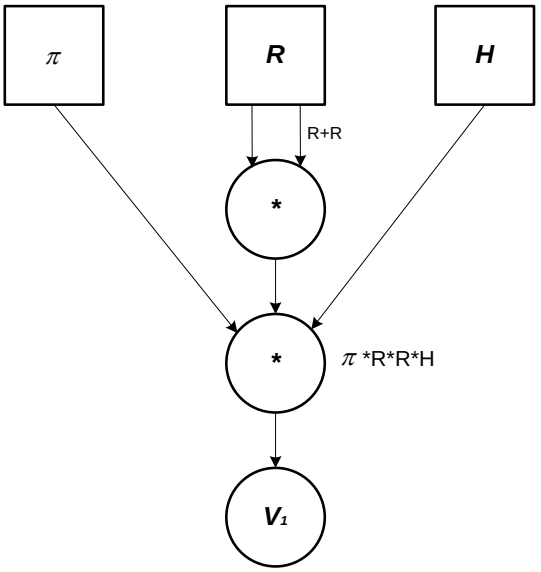
Коренер – 8 τ.

Формулы для вычисления объёма цилиндра:

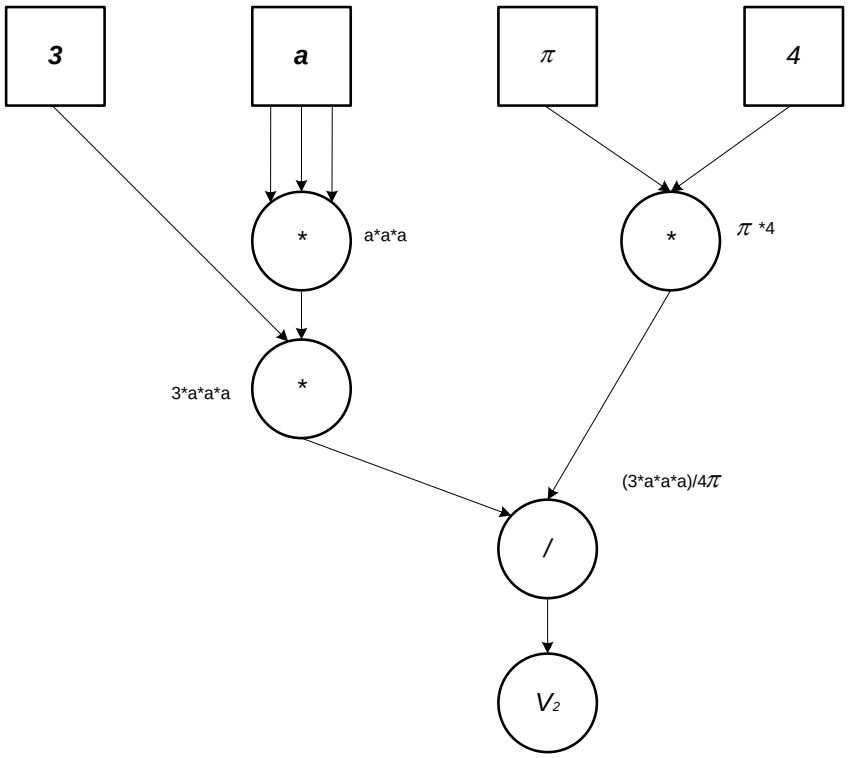
1. $V = \pi R^2 H$
2. $V = \frac{3a^3}{4\pi}$
3. $V = \frac{(b^2 - a^2)a}{4\pi}$
4. $V = \frac{P(H+R) - 2S}{2\sqrt{\frac{\pi}{S}}}$
5. $V = S((R^2 2a\cos((R-H)\setminus R) - (R-H)\sqrt{(H^2 - 2HR)}$

Графы, описывающие формулы нахождения объёма цилиндра

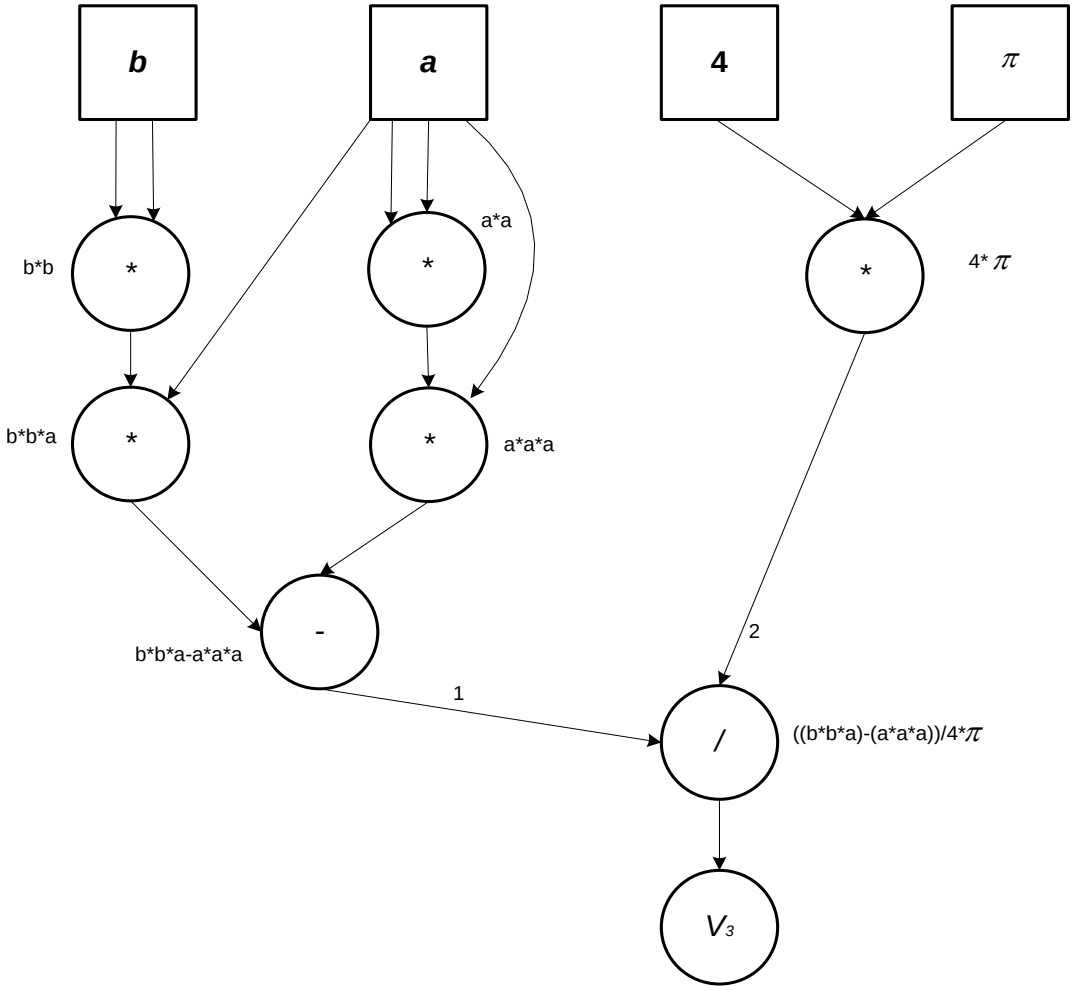
1. V= $\pi R^2 H$



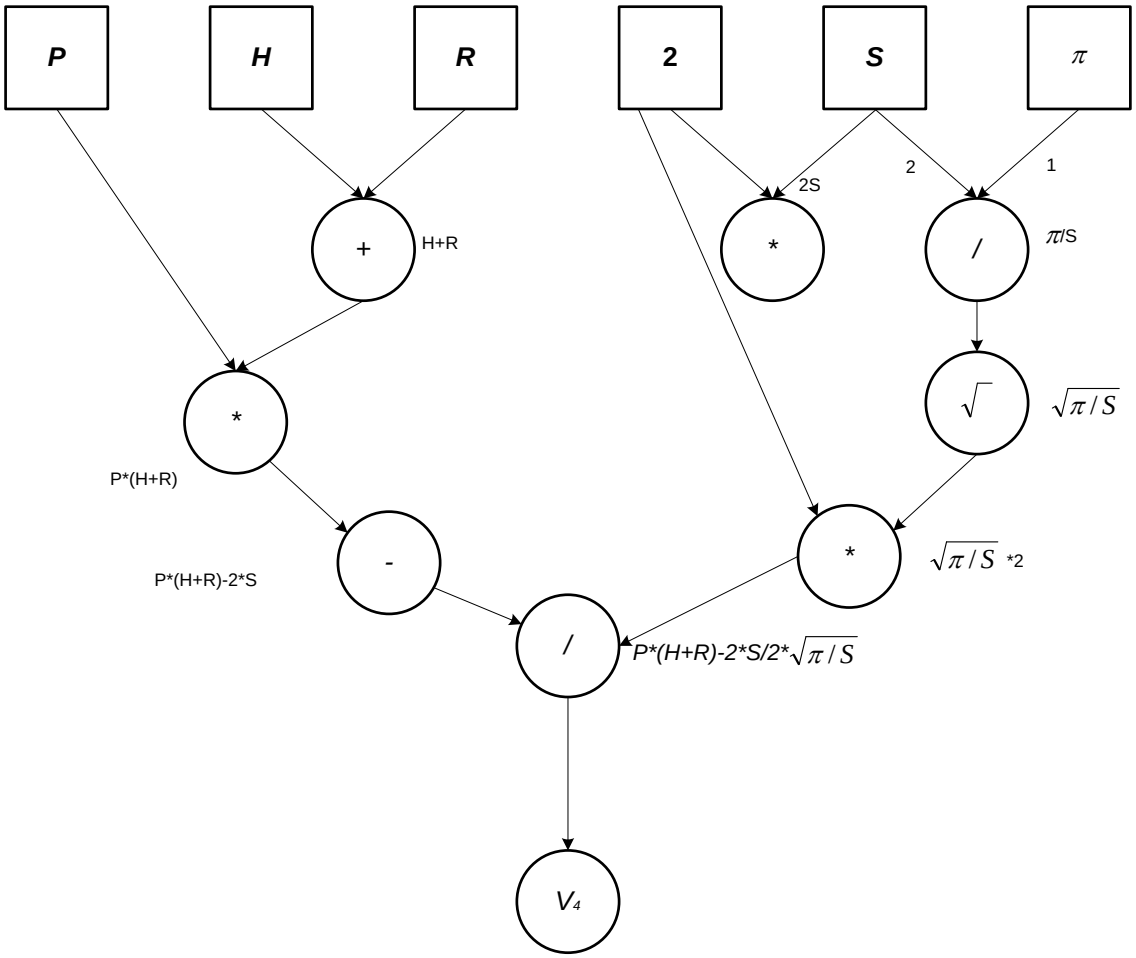
2. V= $\frac{3a^3}{4\pi}$



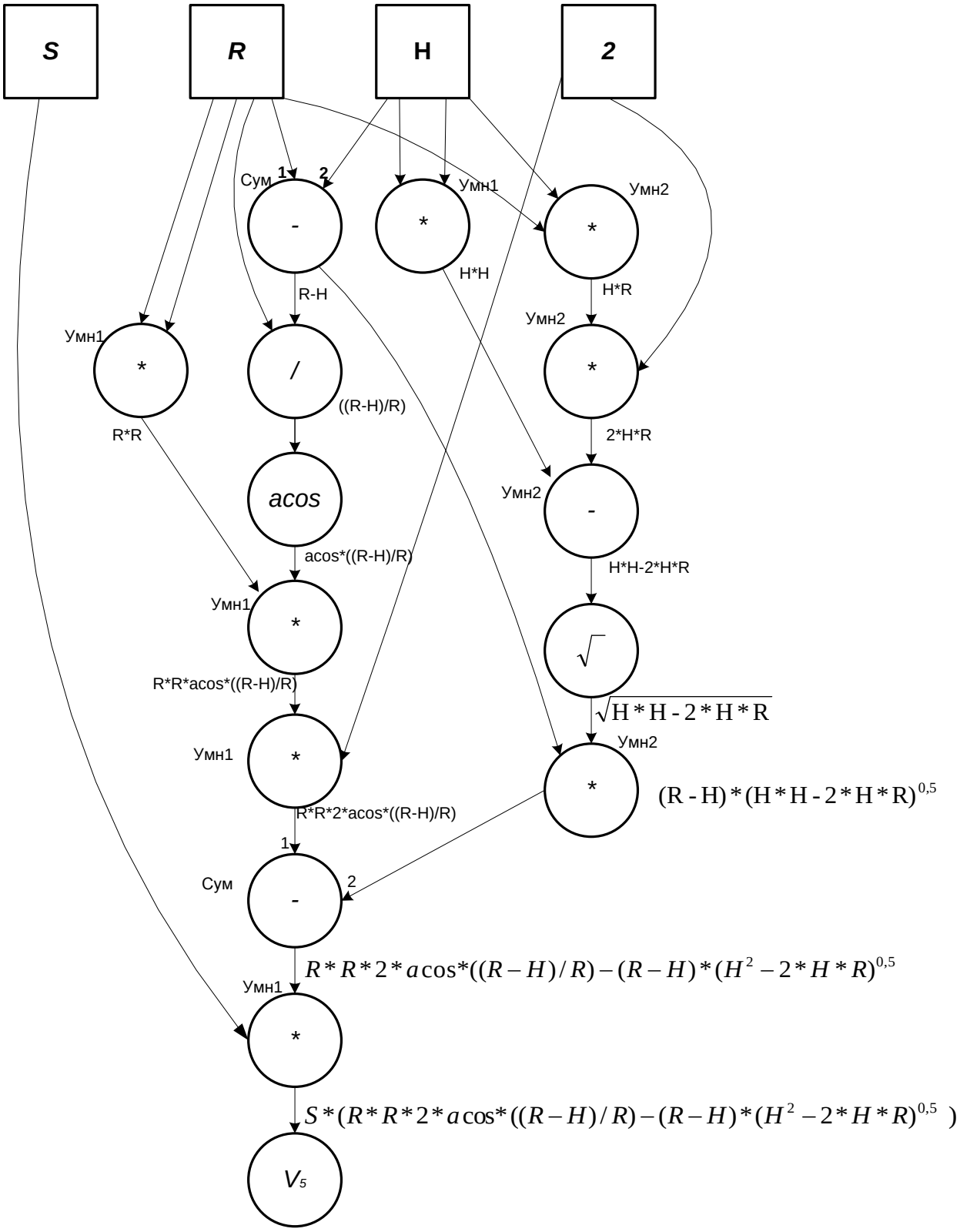
3. V= $\frac{(b^2 - a^2)a}{4\pi}$



4. V= $\frac{P(H+R)-2S}{2\sqrt{\pi/S}}$



5. V=$S((R^2 2a\cos((R-H)/R)-(R-H)(H^2-2HR)^{0,5})$



Граф общего алгоритма нахождения объёма цилиндра

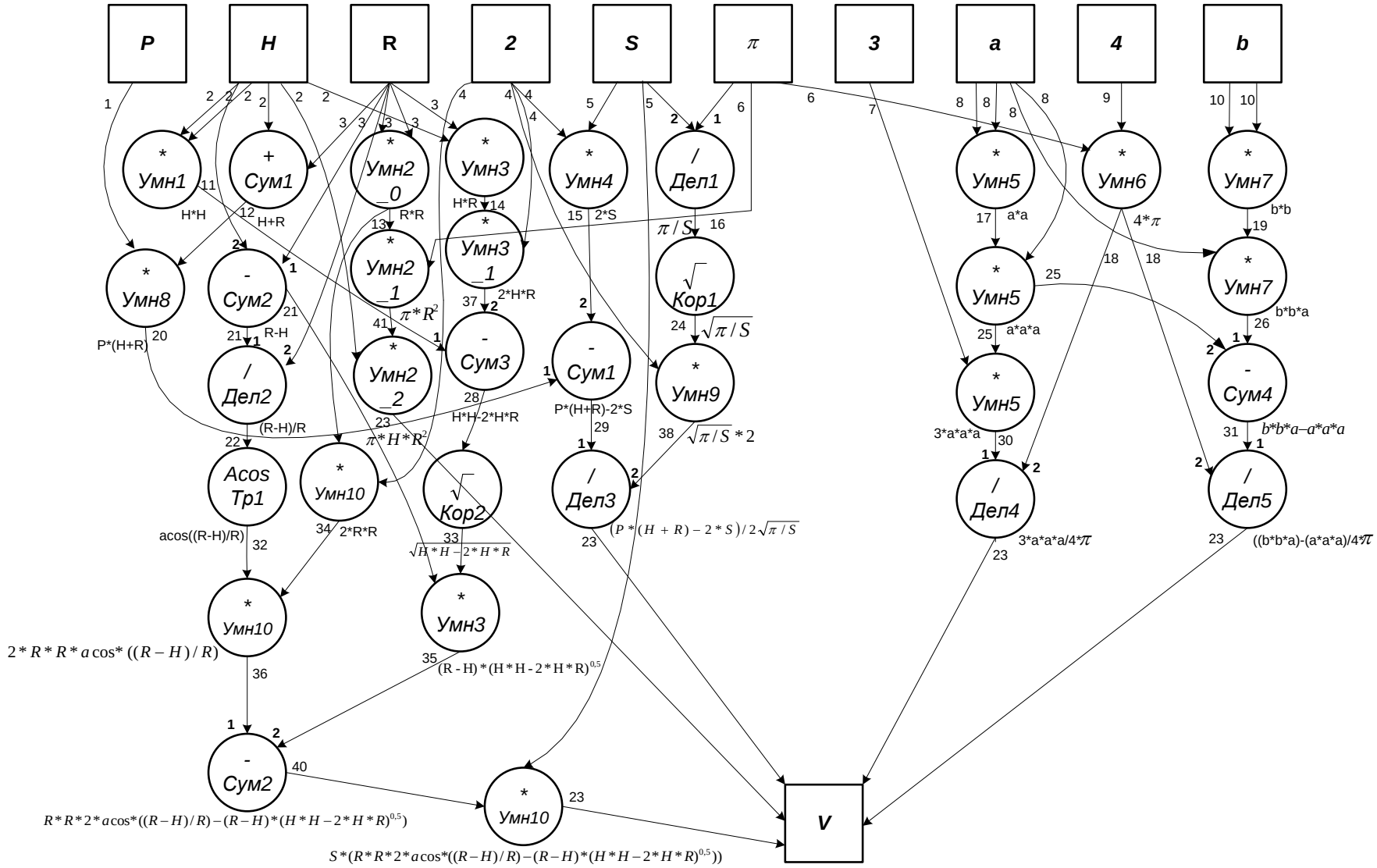


Таблица программ функциональных устройств

Устройство	Ярлыки входных данных	Операция	Комментарии	Ярлык выходных данных
Умножитель 1	2	^2	H*H	11
Умножитель 2	3	^2	R*R	13
	6,13	*	π*R*R	41
	2,41	*	π*H*R*R	21
Умножитель 3	2,3	*	H*R	14
	4,14	*	2*H*R	37
	21,33	*	$(R-H)*(H*H-2*H*R)^{0,5}$	35
Умножитель 4	4,5	*	H*H	15
Умножитель 5	8,8	*	a*a	17
	8,17	*	a*a*a	25
	7,25	*	3*a*a*a	30
Умножитель 6	6,9	*	4π	18
Умножитель 7	10,10	*	b*b	19
	8,19	*	b*b*a	26
Умножитель 8	1,12	*	P*(H+R)	20
Умножитель 9	4,24	*	$\sqrt{\pi / S} * 2$	38
Умножитель 10	4,13	*	$R * R * 2 * a\cos*((R-H)/R)$	36
	32,34	*	2*R*R	36
	5,40	*	S*(R*R*2*acos*((R-H)|R)-(R-H)*(H*H-2*H*R)$^{0.5}$)	23
Сумматор 1	2,3	+	R+H	12
	20,15	-	P*(H+R)-2*S	29
Сумматор 2	3,2	-	R-H	21
	36,35	-	R*R*2*acos*((R-H)|R)-(R-H)*(H*H-2*H*R)$^{0.5}$	40
Сумматор 3	11,37	-	H*H-2*R*H	28
Сумматор 4	26,25	-	b*b*a-a*a*a	31
Коренер 1	16	√	$\sqrt{\pi / S}$	24
Коренер 2	28	√	(H*H-2*H*R)$^{0.5}$	33
Делитель 1	6,5	/	π/S	16
Делитель 2	21,3	/	(R-H)/R	22
Делитель 3	29,38	/	$\left(P*(H+R)-2*S\right)/2\sqrt{\pi / S}$	23
Делитель 4	30,18	/	$3a^3/4\pi$	23
Делитель 5	31,18	/	$b^2*a-3a^3/4\pi$	23
Блок тригонометрических операций 1	22	Acos	acos((R-H)/R)	32

Создание миллипрограммы решения вычислительной задачи

Имитационное моделирование ВС проводится с помощью Среде ОА-программирования и моделирования (документация к среде находится в приложении 1), которая позволяет создавать виртуальную копию компьютерной ВС и проводить ее исследование с целью определения оптимальных параметров ВС. В среде применяется специальный язык программирования (ОА-язык), с помощью которого можно не только задавать вычислительный алгоритм, но и конфигурировать модель ВС, задавать входные данные, настраивать программную модель.

Моделирование можно проводить по следующей схеме.

1. Построение потокового графа решения вычислительной задачи.

2. Составление ОА-программы для решения вычислительной задачи в среде ОА-программирования и её отладка.

3. Дополнение ОА-программы для имитационного моделирования: вводится конфигурация ВС (вычислительные узлы, связи между вычислительными узлами, конфигурация вычислительных узлов, настройки вывода результатов моделирования и т.д.).

4. Проведение серии прогонов имитационной модели с целью определения оптимальной конфигурации ВС.

Составление ОА-программы для решения вычислительной задачи

Первый этап составления ОА-программы – создание виртуальных ФУ, для решения задачи. Для этого необходимо переименовать все вершины потокового графа алгоритма (имена вершин будут являться именами ФУ, созданных в среде ОА-программирования). Имена ФУ необходимо подписывать непосредственно на потоковом графе, т.к. это сделает дальнейшее выполнение работы более удобным. Создание ФУ

осуществляется с помощью милликоманды NewFU (горячая клавиша в среде ОА-программирования - ctrl+d). В капсуле, передаваемой с данной милликомандой могут находится информационные пары (ИП) со следующими атрибутами: Mnemo – название создаваемого ФУ, FUType – тип создаваемого ФУ, Hint – всплывающая подсказка, которая появляется при наведении курсора на название ФУ в среде ОА-программирования (данный параметр является необязательным).

Будем придерживаться следующего правила именования ФУ: сумматор и вычитатель обозначается Sum, умножитель – Mul, делитель Div, ФУ для возведения числа в квадрат – Sqr, для извлечения квадратного корня – Sqrt, ФУ для получения синуса, косинуса, тангенса, котангенса – Sin, Cos, Tg, Ctg соотвественно. Далее за названием, отображающем функционал ФУ, будет следовать порядковый номер ФУ: например, Sum1, Sum2 и т.д.

Для выполнения работы нам будет необходим один тип ФУ – FUStreamFloatAlu (потоковое арифметико-логическое устройство с плавающей точкой.).

Например, создание ФУ с именем Mul1 с типом FUStreamFloatAlu и всплывающей подсказкой «Первый умножитель» будет выглядеть так:

```
NewFU={Mnemo="Mul1" FUType=FUStreamFloatAlu Hint="Первый умножитель"}
```

В нашем примере нам будет необходимо создать 22 ФУ с соответствующими именами (мнемониками) и типом FUStreamFloatAlu.

Далее нам необходимо описать константы необходимые нам для вычислений. Константы задаются с помощью знака «#»: перед знаком помещается мнемоника константы, после – значение. Например, a#10. В нашем примере нам будут необходимы следующие константы: a – боковая грань, b – диагональ, H – высота, b – диагональ, P – периметр основания, S – площадь основания цилиндра:

```
a#10 b#10 H#10 P#10 R#10 S#10
```

Константы необходимо подбирать таким образом, чтобы она удовлетворяли условиям решения поставленной геометрической задачи. Константа Пи является встроенной в среду ОА и обозначается Pi.

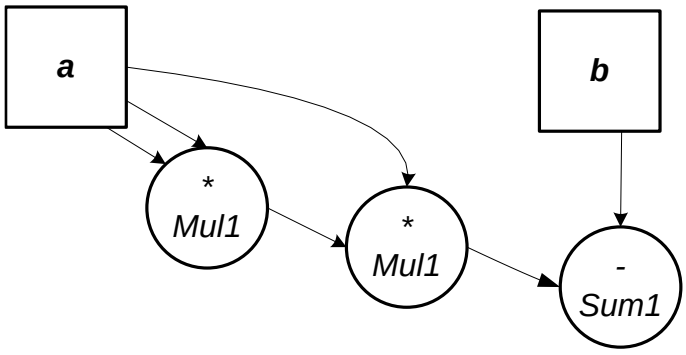
Для того, чтобы ФУ были созданы в среде ОА-программирования необходимо запустить ОА-программу на выполнение. *Примечание: не забудьте сделать вкладку с ОА-программой корневой (Root)*. После запуска ОА-программы имена ФУ должны быть отображены в панели ФУ, а константы – в панели констант.

Следующим этапом является непосредственно описание алгоритма решения вычислительной задачи. Описание алгоритма работы каждого ФУ осуществляется следующим образом:

1. если в узел входит дуга из вершины, обозначающей константу, то необходимо описать милликоманду вида: `Мнемоника_ФУ. Милликоманда =Мнемоника_константы.`

2. если из узла потокового графа выходит дуга, то в ОА-программе её необходимо описать следующим образом:

`Мнемоника_ФУ.ReseiverMkSet=Мнемоника_ФУ-приемника.Милликоманда`



Например, для фрагмента потокового графа, представленного на рис. Следует описать задать такую ОА-программу:

```
1: Mul1.Mul1=a
2: Mul1.Mul2=a
```

```
3: Mul1.ReceiverMkSet=Mul1.1_Mul1
4: Mul1.1_Mul2=a
5: Mul1.1_ReseiverMkSet=Sum1.Sum2
6: Sum1.Sum1=a
7: Sum1.ReceiverMkSet=Console.LnOut
```

В строках 1,2,4 и 6 описывается поступление в качестве входных данных, констант. В строчке 7 описывается вывод результата вычислений на консоль (милликоманда `Console.LnOut`). Рекомендуется вынести все милликоманды, описывающие поступление констант (исходных данных) в конец ОА-программ, что в дальнейшем обеспечит удобство доработки ОА-программы для проведения имитационного моделирования. В том случае, когда на одном ФУ реализуются несколько операций, то для каждой операции перед мнемоникой милликоманды, ставится номер команд с последующим знаком «_»; для первой команды номер не ставит (подразумевается «0_»). В строках 1,2,3 – ведется настройка первой команды ФУ Mul1, в строках 4,5 – второй команды.

ОА-программа, соответствующая разбираемому примеру приведена в приложении 2.

Коэффициент параллелизма определяется как отношение суммы всех работающих устройств на каждом такте выполнения алгоритма к общему числу тактов работы вычислительной системы.

Определение коэффициента параллелизма

Коэффициент параллелизма определяется как отношение суммы всех работающих устройств на каждом такте выполнения алгоритма к общему числу тактов работы вычислительной системы.

Для системы с централизованным управлением коэффициент параллелизма будет:

К=83/64=1,3

Для системы с распределённой системы

К=200/40=5

Рекомендации по написанию ОА-программы

Написание программа лучше производить в следующей последовательности:

1. Объявить все ФУ, необходимые для вычислений.
2. Объявить все необходимые константы.
3. Ввести все исходные данные (раздел «исходные данные» в приложении 1)
4. Затем производить поэтапное программирование информационных пересылок между ФУ (раздел «Описание пересылок данных»). Во избежание совершения ошибок лучше всего производить программирование пересылок по ярусам потокового графа (ярус –

вершины имеющие одинаковый максимальный путь от основания графа (т.е. от вершин, которые представляют исходные данные). При программировании каждого ФУ желательно выводить результаты его работы на консоль. Для этого можно воспользоваться милликомандой ИмяФУ.ReseaverMkSet=Console.LnOut. Например, для проверки того, что на ФУ Mul4 (приложение 2) приходят все операнды, надо записать Mul4.ReceiverMkSet=Console.LnOut. После запуска ОА-программы результат вычислений Mul4 должен появиться в консоли вывода. После этого меняем Mul4.ReceiverMkSet=Console.LnOut на Mul4.ReceiverMkSet=Mul4.C1_Mul, т.е. назаначае для Mul4 его настоящего потребителя и приступаем к проверке следующего ФУ.

Приложение 1. Список милликоманд ФУ «Потоковое АЛУ с плавающей точкой» и ФУ вывода

Общие милликоманды

OperandsReset: Сброс операндов
ReceiverMkSet:Установить милликоманду для потребителя (автоматически создается новая запись описания потребителя
ReceiversReset: Сбросить список потребителей
FireingProgSet: Установить указатель на подпрограмму, запускаемую при активизации вершины (получении результата выполнения операции)"
PopMk: "Выдать результат выполнения операции"
AngleModeSet: Установить режим измерения угла (0- радиены, 1 - градусы).
AutoDisactiveModeSet: Установить режим автоматического перехода ФУ в пассивный режим после получения выходных данных (нужно для того, чтобы предотвратить нежелательную повторную активацию ФУ, которая может нарушить вычислительный процесс).
ActiveSet: Установить режим активности (true на входе – активный режим, false – пассивный).

Арифметические функции

NOperandsSet: Установить число операндов для операций с накоплением
Sum: Сложение с накоплением.
Sub: Вычитание с накоплением.
Mul: Умножение с накоплением.
Div: Деление с накоплением.
Sum1: Первое слагаемое.
Sum2: Второе слагаемое.
Sub1: Вычитаемое.
Sub2: Вычитатель.
Mul1: Умножаемое.
Mul2: Умножитель.
Div1: Делимое.
Div2: Делитель.
Sqr: Квадрат.
Sqrt: Квадратный корень.

Тригонометрические операции

Sin: Синус.
Cos: Косинус.

Tg: Тангенс.
Ctg: Котангенс.
Asin: Арксинус.
Acos: Арккосинус.
Atg": Арктангенс.

Настройки моделирования

RegLoadTimeSet: Установить время записи во внутренний регистр.
SumTimeSet: Установить время суммирования.
SubTimeSet: Установить время вычитания.
MulTimeSet: Установить время умножения.
DivTimeSet: Установить время деления.
SqrTimeSet: Установить время вычисления квадрата.
SqrtTimeSet: Установить время вычисления квадратного корня.
SinTimeSet: Установить время вычисления синуса.
CosTimeSet: Установить время вычисления косинуса.
TgTimeSet: Установить время вычисления тангенса.
CtgTimeSet: Установить время вычисления котангенса.
AsinTimeSet: Установить время вычисления арксинуса.
AcosTimeSet: Установить время вычисления арккосинуса.
AtgTimeSet: Установить время вычисления квадратного арктангенса.

Милликоманды консоли вывода:

Console.LnOut: вывод константы или переменной на консоль с новой строки.
Console.Out: вывод константы или переменной на консоль на той же строке.

Приложение 2. Миллипрограмма расчета объёма конуса

```
NewFU={Mnemo="Mul1" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Mul2" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Mul3" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Mul4" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Mul5" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Mul6" FUType=FUStreamFloatAlu}

NewFU={Mnemo="Sum1" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Sum2" FUType=FUStreamFloatAlu}

NewFU={Mnemo="Div1" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Div2" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Div3" FUType=FUStreamFloatAlu}

NewFU={Mnemo="Sqrt1" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Sqrt2" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Sin1" FUType=FUStreamFloatAlu}
NewFU={Mnemo="Cos1" FUType=FUStreamFloatAlu}

Sin1.AngleModeSet=1 \\ Задать режим измерения угла в градусах
Cos1.AngleModeSet=1 \\ Задать режим измерения угла в градусах

b#14.14213562373095 \\ Длина диагонали
H#10                \\ Высота
P#31.41592653589793 \\ Периметр
R#5                 \\ Радиус
S#78.53981633974483 \\ Площадь основания
a=45                \\ Угол между диагональю и верхней
поверхностью конуса

\\ Описание пересылок данных

\\ Формула 1
Mul4.ReceiverMkSet=Mul4.C1_Mul
Mul4.C1_ReceiverMkSet=Mul4.C2_Mul
Mul4.C2_ReceiverMkSet=Console.LnOut
\\ Формула 2
Mul5.C2_NOperandsSet=3
Mul5.ReceiverMkSet=Sum2.Sub1
Mul5.ReceiverMkSet=Mul5.C1_Mul
Mul5.C1_ReceiverMkSet=Mul5.C2_Mul
Mul5.C2_ReceiverMkSet=Console.LnOut
Mul2.ReceiverMkSet=Mul5.C1_Mul
Sin1.ReceiverMkSet=Div3.Div1
Cos1.ReceiverMkSet=Mul6.Sqr
```

```
Div3.ReceiverMkSet=Mul5.C2_Mul
Mul6.ReceiverMkSet=Mul5.C2_Mul
\\ Формула 3
Div2.ReceiverMkSet=Mul3.Mul
Mul3.ReceiverMkSet=Sum2.Sub2
Sum2.ReceiverMkSet=Sqrt2.Sqrt
Sqrt2.ReceiverMkSet=Mul3.C1_Mul
Mul3.C1_ReceiverMkSet=Console.LnOut
\\ Фориула 4
Div1.ReceiverMkSet=Mul1.Sqr
Mul1.ReceiverMkSet=Div1.C1_Div1
Div1.C1_ReceiverMkSet=Mul2.C2_Mul

Mul1.C1_ReceiverMkSet=Sum1.Sub2
Mul2.ReceiverMkSet=Mul2.C1_Sqr
Mul2.C1_ReceiverMkSet=Sum1.Sub1
Sum1.ReceiverMkSet=Sqrt1.Sqrt
Sqrt1.ReceiverMkSet=Mul2.C2_Mul
Mul2.C2_ReceiverMkSet=Console.LnOut

// Исходные данные
Mul1.Sqr=P
Div2.Div1=S
Mul3.C1_Mul=S
Div2.Div2=pi
Mul2.Mul=pi
Mul4.C1_Mul=pi
Mul4.C2_Mul=H
Mul4.Sqr=R
Div1.Div2=pi
Div1.Div1=P
Div1.C1_Div2=4
Mul3.Mul=4
Div3.Div2=4
Mul5.Sqr=b
Mul2.Mul=b
Sin1.Sin=a
Cos1.Cos=a
Mul1.C1_Sqr=P
```

Приложение 3. Миллипрограмма с настройками для моделирования

```
NewFU={Mnemo="Gant" FUType=FUGant}
NewFU={Mnemo="Eventser" FUType=FUEventser}
NewFU={Mnemo="Scheduler" FUType=FUScheduler}

NewFU={Mnemo="Mul1" FUType=FUStreamFloatAlu}

Eventser.CurrentTimePointPopMk=Gant.CurrentTimeRefSet

Eventser.ContextPopMk=Scheduler.EventserContextSet
Scheduler.ContextPopMk=Mul1.SchedulerContextSet

Eventser.OutProgSet={Eventser.EventCapsPopMk=Gant.EventSet}
Eventser.EventRequestProgSet={Eventser.EventRequestCapsPopMk=Gan
t.OperandsSet}
Eventser.FinProgSet={Gant.Draw
          Console.LnOut="Время выполнения программы: "
Eventser.CurrentTimePointPopMk=Console.Out
}

Scheduler.NCoresSet=4

Mul1.SubTimeSet=1
Mul1.MulTimeSet=4
Mul1.DivTimeSet=5
Mul1.SumTimeSet=1
Mul1.SqrTimeSet=7
Mul1.SqrtTimeSet=8
Mul1.RegLoadTimeSet=0.2

ContextTemplate=nil
Mul1.ManualModeSet=true
Mul1.ContextPop=ContextTemplate

NewFU={Mnemo="Mul2" FUType=ContextTemplate}
NewFU={Mnemo="Mul3" FUType=ContextTemplate}
NewFU={Mnemo="Mul4" FUType=ContextTemplate}
NewFU={Mnemo="Mul5" FUType=ContextTemplate}
NewFU={Mnemo="Mul6" FUType=ContextTemplate}

NewFU={Mnemo="Sum1" FUType=ContextTemplate}
NewFU={Mnemo="Sum2" FUType=ContextTemplate}

NewFU={Mnemo="Div1" FUType=ContextTemplate}
NewFU={Mnemo="Div2" FUType=ContextTemplate}
NewFU={Mnemo="Div3" FUType=ContextTemplate}
```

```
NewFU={Mnemo="Sqrt1" FUType=ContextTemplate}
NewFU={Mnemo="Sqrt2" FUType=ContextTemplate}
NewFU={Mnemo="Sin1" FUType=ContextTemplate}
NewFU={Mnemo="Cos1" FUType=ContextTemplate}

Sin1.AngleModeSet=1 \\ Задать режим измерения угла в градусах
Cos1.AngleModeSet=1 \\ Задать режим измерения угла в градусах

b#14.14213562373095 \\ Длина диагонали
H#10                \\ Высота
P#31.41592653589793 \\ Периметр
R#5                 \\ Радиус
S#78.53981633974483 \\ Площадь основания
a=45                \\ Угол между диагональю и верхней
поверхностью конуса
\\ Формула 1
Mul4.ReceiverMkSet=Mul4.C1_Mul
Mul4.C1_ReceiverMkSet=Mul4.C2_Mul
Mul4.C2_ReceiverMkSet=Console.LnOut
\\ Формула 2
Mul5.C2_NOperandsSet=3
Mul5.ReceiverMkSet=Sum2.Sub1
Mul5.ReceiverMkSet=Mul5.C1_Mul
Mul5.C1_ReceiverMkSet=Mul5.C2_Mul
Mul5.C2_ReceiverMkSet=Console.LnOut
Mul2.ReceiverMkSet=Mul5.C1_Mul
Sin1.ReceiverMkSet=Div3.Div1
Cos1.ReceiverMkSet=Mul6.Sqr
Div3.ReceiverMkSet=Mul5.C2_Mul
Mul6.ReceiverMkSet=Mul5.C2_Mul
\\ Формула 3
Div2.ReceiverMkSet=Mul3.Mul
Mul3.ReceiverMkSet=Sum2.Sub2
Sum2.ReceiverMkSet=Sqrt2.Sqrt
Sqrt2.ReceiverMkSet=Mul3.C1_Mul
Mul3.C1_ReceiverMkSet=Console.LnOut
\\ Фориула 4
Div1.ReceiverMkSet=Mul1.Sqr
Mul1.ReceiverMkSet=Div1.C1_Div1
Div1.C1_ReceiverMkSet=Mul2.C2_Mul

Mul1.C1_ReceiverMkSet=Sum1.Sub2
Mul2.ReceiverMkSet=Mul2.C1_Sqr
Mul2.C1_ReceiverMkSet=Sum1.Sub1
Sum1.ReceiverMkSet=Sqrt1.Sqrt
Sqrt1.ReceiverMkSet=Mul2.C2_Mul
```

```
Mul2.C2_ReceiverMkSet=Console.LnOut

// Исходные данные
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul1.Sqr=P}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Div2.Div1=S}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul3.C1_Mul=S}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Div2.Div2=pi}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul2.Mul=pi}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul4.C1_Mul=pi}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul4.C2_Mul=H}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul4.Sqr=R}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Div1.Div2=pi}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Div1.Div1=P}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Div1.C1_Div2=4}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul3.Mul=4}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Div3.Div2=4}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul5.Sqr=b}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul2.Mul=b}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Sin1.Sin=a}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Cos1.Cos=a}}
Eventser.EventWaitSet={ TimeAtr=0 Mk={Mul1.C1_Sqr=P}}

Eventser.Start
```

Литература

1. Берж К. Теория графов и ее применения. М., изд. Иностранной литературы. 1962.
2. Уилсон Р. Введение в теорию графов. М.: Мир, 1977.
3. Корнеев В.В., Киселев А.В. Современные микропроцессоры. – 3-е издание, перераб. и доп. – СПб.: БХВ-Петербург. 2003. – 448 с.
4. Компьютеры на СБИС: В 2-х кн. Кн.1/Пер. с япон. Мотоока Т., Томита С., Танака Х. и др. – Мир, 1988. – 392 с.
5. Велях Е. Последовательно-параллельные вычисления: Пер. с англ. - М.: Мир, 1985.- 456 с.
6. Введение в алгоритмы параллельных вычислений /Молчанов И.Н.; Отв. ред. Яковлев М.Ф.; АН УССР. Ин-т кибернетики им. В.М. Глушкова. – Киев: Наук. думка, 1990. – 128 с.
7. Параллельные вычисления /Под ред. Г. Родринга: Пер. с англ. /Под ред. Ю.Г. Дадаева. – М.: Наука. Гл. ред. физ.-мат. лит., 1986. – 376 с.

Учебное издание

Вычислительная система, управляемая потоком данных

Составитель САЛИБЕКЯН Сергей Михайлович

Редактор Е.С. Резникова
Технический редактор О.Г. Завьялова

Подписано в печать 20.11.2009. Формат 60x84/16. Бумага офсетная №2. Ризография. Усл. печ. л. 1,9. Уч.-изд. л. 1,5. Изд. №87. Тираж 30 экз. Заказ . Бесплатно.

Московский государственный институт электроники и математики.

109028, Москва, Б. Трёхсвятительский пер., 3.

Отдел оперативной полиграфии Московского государственного института электроники и математики.

113054, Москва, ул. М. Пионерская, 12.