 2Антик М.И. 11\_03\_91 МИРЭА

 \_АЛГОРИТМЫ ПРОЦЕДУРНОГО ТИПА. ОПЕРАЦИОННЫЕ УСТРОЙСТВА

Алгоритмы этого типа являются следующим этапом обобщения

описаний вычислительных процессов. Теперь, по сравнению с ал-

горитмами автоматного типа, на каждом шаге, помимо модифика-

ции памяти, идентифицирующей шаг алгоритма, разрешается изме-

нять любую другую память устройства локально (по частям) или

глобально (всю сразу).

Устройство-исполнитель алгоритма этого типа будем назы-

вать операционным устройством (ОУ).

ОУ можно рассматривать как один синхронный автомат со

сложно структурированной памятью - состоянием: часть памяти

используется для идентификации шага алгоритма, остальная па-

мять используется для запоминания промежуточных данных, вы-

числяемых в процессе последовательного, по шагам, выполнения

алгоритма. Такая модель вычислителя особенно удобна для рас-

чета продолжительности одного такта работы устройства.

Другой удобной моделью вычислителя является совокуп-

ность взаимодействующих синхронных автоматов, один из которых

называется управляющим автоматом (УА), а объединение всех ос-

тальных автоматов называется операционным автоматом (ОА).

УА является исполнителем алгоритма автоматного типа, ко-

торый входит составной частью в любой алгоритм процедурного

типа. Кроме того, УА инициирует действия отдельных шагов ал-

горитма и участвует в их выполнении.

ОА выполняет все вычисления на отдельных шагах алгоритма

под управлением УА; кроме того, к ОА удобно отнести все вы-

числения предикатных функций, оставив УА только анализ вычис-

ленных предикатных значений.

Прежде чем переходить к точным терминам, рассмотрим сле-

дующиe примеры алгоритмов процедурного типа.

Например, каноническое описание синхронного конечного

автомата может быть интерпретировано (истолковано) как одно-

шаговый алгоритм процедурного типа.

█

┌──────┐ │

│ ┌──V──V─────┐

│ │ B=FO(S,A) │

│ │ │

│ │ S:=FS(S,A)│

│ └─────┬─────┘

└─────────┘

Исполнитель этого алгоритма состоит только из ОА. На

каждом шаге этого алгоритма изменяется вся память устройства,

поэтому управление памятью не требуется, идентифицировать ша-

ги этого алгоритма не надо.

Например, инкрементор с одноразрядным входом и однораз-

рядным выходом может быть реализацией следующих преобразова-

ний:

█

█ p:=1 █

█

┌────────┐ │

│ ┌─────V─V───────┐

│ │ (p:, b) = a+p │

│ └───────┬───────┘

└──────────┘

- 2 -

ОА, реализующий инкрементор в целом, будет следующим:

┌──┬─┐

a ──────────────────┤HS│S├────\_b

┌─┬─┐p │ │ │

начальное значен.─┤S│T├──┤ │P├──┐

├─┤ │ └──┴─┘ │

SYN ─────────/C│ │ │

┌┤D│ │ │

│└─┴─┘ │

└───────────────┘

Конечно, эта реализация совпадает с реализацией алгоритма ав-

томатного типа, если состояние р1 кодируется 1, а состояние

р0 - 0.

Этот пример показывает,что до начала вычислений может

потребоваться начальная установка памяти. На самом деле это

необходимо было уже в алгоритмах автоматного типа, так как

код начального состояния требует предварительной установ-

ки. Ограничимся здесь обозначением этой проблемы, а решение

ее, связанное прежде всего с корректной синхронизацией ус-

тройства в первом такте его работы, рассмотрим ниже.

При рассмотрении процедуры построения автомата Мура эк-

вивалентного автомату Мили , не обсуждалась простая возмож-

ность ее реализации и на структурном уровне. Например, только

что рассмотренный автомат Мили может быть преобразован в эк-

вивалентный автомат Мура по одному из следующих вариантов:

┌┬─┬┐t┌──┬─┐ ┌──┬─┐ ┌┬─┬┐

a ──\_┤│T│├\_┤HS│S├─\_b a ─────\_┤HS│S├─\_┤│T│├─\_b

─/┴┴─┴┘ │ │ │ │ │ │─/┴┴─┴┘

C │ │ │ C │ │ │ C

─/┬┬─┬┐p│ │ │ ─/┬┬─┬┐p│ │ │

┌\_┤│T│├\_┤ │P├┐ ┌\_┤│T│├\_┤ │P├┐

│ └┴─┴┘ └──┴─┘│ │ └┴─┴┘ └──┴─┘│

└─────────────┘ └─────────────┘

При таких структурных преобразованиях проще истолковы-

вать алгоритмы как процедурные.

█ █

█ p:=1; t:=0 █ █ p:=1 █

█ █

┌────────┐ │ ┌────────┐ │

│ ┌─────V─V───────┐ │ ┌─────V─V───────┐

│ │t:=a;(p:,b)=t+p│ │ │ (p , b):= a+p │

│ └───────┬───────┘ │ └───────┬───────┘

└──────────┘ └──────────┘

БЛОК-ТЕКСТ. Способ описания автоматного алгоритма после

некоторых дополнений может быть использован и для описания

алгоритмов в процедурной форме:

Блок-текст состоит из трех частей:

1)- Описание переменных и начальных значений памяти.

2)- Описания функций и связей. Записываются любые функции и

функциональные связи (в том числе предикатные), не использу-

ющие запоминания. Переменные из левой части операции присва-

ивания таких функциональных описаний используются в блоках

процедуры.

3)- Процедура, состоящая из блоков (микроблоков) операторных

и блоков переходов:

- операторные - в скобках вида {.....},

- перехода - в скобках вида <<...>>;

и те, и другие блоки могут снабжаться метками, стоящими перед

блоком. В блоках перехода используется оператор GO в одной

из двух форм:

GO m - безусловный переход,

GO (P; m0,m1,m2,...) - условный переход.

здесь m0,m1,... - метки блоков,

P - предикатное значение,интерпретируемое оператором GO

- 3 -

как неотрицательное целое число, являющееся порядковым номе-

ром метки в списке меток оператора GO. С этой метки должно

быть продолжено выполнение алгоритма. Блоки условных перехо-

дов эквивалентны предикатным вершинам блок-схемы алгоритма.

На следующем более сложном примере демонстрируется пос-

ледовательность синтеза операционного устройства.

Пример. Вычислитель наибольшего общего делителя (НОД)

двух натуральных чисел (8-разрядных).

1) Разработаем интерфейс вычислителя:

8 ┌──┬─────┬──┐

═══╪═>╡I1│ НОД │ │

│ │ │ │ 8

8 │ │ │D ╞══╪══>

═══╪═>╡I2│ │ │

├──┤ ├──┤

─────>┤rI│ │rO├─────>

├──┤ │ │

─────>┤ C│ │ │

└──┴─────┴──┘

I1[7..0], I2[7..0] -входные информационные шины.

rI -входной сигнал готовности: если rI=1, то на входах I1,

I2 готовы операнды.

D[7..0] -выходная информационная шина .

rO -выходной сигнал готовности: если rO=1, то готов резуль-

тат на шине D, который сохраняется до появления новых операн-

дов.

2) Математическое обоснование алгоритма вычислений:

Идея алгоритма, следуя Евклиду (IIIв. до р.Х.), заключа-

ется в том, что НОД двух натуральных чисел А и В в случае ра-

венства этих чисел совпадает с любым из них, а в случае их

неравенства совпадает с НОД двух других чисел: меньшего и

разности между большим и меньшим. Последовательно, уменьшая

числа, получим два равных числа -значение одного из них и бу-

дет НОД исходных чисел.

3) Блок-схема алгоритма (микропрограмма в содержательном

виде):

- 4 -

█

│

┌──────V──────┐

m1│ rO: = 0 │

└──────┬──────┘

│┌──────────────────┐

││┌─────┐ │

─VVV─ │ │

/ \ 0 │ │

< rI>─────┘ │

\\_/ │

│1 │

┌──────V──────┐ │

│ rO: = 0 │ │

│ │ │

m2│ А: = I1 │ │

│ │ │

│ B: = I2 │ │

└──────┬──────┘ │

┌───────────────────┐│ │

│ ┌─────VV──────┐ │

│ m3│ (p,S)=A - B │ │

│ └──────┬──────┘ │

│ ─V─ m6 │

│ / \ =0 ┌──────────┐│

│ z <S==0>───>┤ rO:=1;D=A├┘

│ \\_\_/ └──────────┘

│ │╪0

│ V

│ 0 / \ 1

│ ┌───────< p >────────┐

│ ┌───────V──────┐ \\_/ ┌───────V──────┐

│m4│ (x,B:)=-A+B │ m5│ (x,A:)=A - B │

│ └───────┬──────┘ └───────┬──────┘

└──────────┴────────────────────┘

Или в виде блок-текста:

I1[7..0], I2[7..0] --входные шины

D[7..0] --выходная шина

rI, rO --сигналы готовности

A[7..0]:, B[7..0]: --память текущих значений

S[7..0] --разность

z, p --предикатные переменные

z=┐(!/S) --сжатие(свертка) S[7..0] по ИЛИ-НЕ

--можно записать иначе z=(S==0)

D=A

-------------------------------------------------------------------

m1{rO:=0}

g1<<GO(rI;g1,m2)>>

m2{rO:=0; A:=I1; B:=I2}

m3{(p,S)=A-B}

<<GO(z;g2,m6)>>

g2<<GO(p;m4,m5)>>

m4{(x,B:)=-A+B}

<<GO m3>>

m5{(x,A:)= A-B}

<<GO m3>>

m6{rO:=1}

<<GO g1>>

- 5 -

4) Разработка функциональной схемы операционного автомата.

В ОА должны быть реализованы все переменные с памятью и

без,а также вычислительные операции,используемые в алгоритме.

A ╔══════════════════════════════>D

─/┬┬──┬┐ ║ ┌────────────┐

C││RG││ ║ │ f1=(A-B) │

││ ││ ║ A│ │

I1═════>══>╡│ │╞══╝ ═>╡ f2=(-A+B)│ ┌─┐

││ ││ │ │S S│1│

││ ││ │ ╞> ═>┤ o───>z

┴┴──┴┘ │ │ │ │

B │ │ └─┘

─/┬┬──┬┐ │ │

C││RG││ │ ├────────────>p

││ ││B B│ │

I2═════>═>╡│ │╞> ═>╡ │ ─/┬┬─┬┐

││ ││ │ │ C││ │├>rO

││ ││ │ │ ││ ││

rI─────>┴┴──┴┘ └────────────┘ └┴─┴┘

Кроме того, в ОА необходимо реализовать все информацион-

ные связи, соответствующие микрооперации коммутации, а также

микрооперации запоминания (загрузки, записи) и обнуления.

╔══════════════════════════════════════════════╗

║ A ╔══════════════════════║═══════>D

║ ┌────┐ ─/┬┬──┬┐ ║ ┌────┐ ┌──────┐ ║

║ │ MUX│ C││RG││ ║ │M2\*8│ 1─>┤cr SM│ ║

╠═>┤0 │ ││ ││ ║ │ │ ├─ │ ║

I1══║═>┤1 ╞══════>╡│ │╞══╩══>╡ ╞═══>╡I1 │ ║ ┌─┐

║ ├ │ ││ ││ A │ │ │ │ ║ │1│

║ │А │ W││ ││ ├─ │ │ S╞═╩>╡ o───>z

║ └A───┘ ─A┴┴──┴┘ └A───┘ │ │ │ │

║ │ │ │ │ │ └─┘

║ umA uA uiA │ │

║ B │ │

║ ┌────┐ ─/┬┬──┬┐ ┌────┐ │ │

║ │ MUX│ C││RG││ │M2\*8│ │ p├─────────>p

╚═>╡0 │ ││ ││ B │ │ │ │

I2════>╡1 ╞══════>╡│ │╞═════>╡ ╞═══>╡I2 │ C

├ │ ││ ││ │ │ │ │ ─/┬┬─┬┐

│А │ W││ ││ ├─ │ │ │1─>┤│T│├>rO

└A───┘ ─A┴┴──┴┘ └A───┘ └──────┘R W││ ││

│ │ │ ─A─A┴┴─┴┘

uMB uB uiB urO uwO

5) Формулировка требований к управляющему автомату.

При формировании управляющих сигналов следует обратить

внимание не только на операции, которые необходимо выполнить

на данном шаге, но и на те оперции, которые нельзя выполнять

на этом шаге, это - как правило, операции изменения памяти.

Будем считать, что операция активна, если значение уп-

равляющего сигнала равно 1.

- 6 -

Для управления вычислениями на каждом шаге алгоритма

потребуются следующие управляющие сигналы:

║umA│umB│uwA│uwB│uiA│uiB│urO│uwO│

══╬═══╪═══╪═══╪═══╪═══╪═══╪═══╪═══╡

m1║ │ │ │ │ │ │ 1 │ 0 │

──╫───┼───┼───┼───┼───┼───┼───┼───┤

m2║ 1 │ 1 │ 1 │ 1 │ │ │ 1 │ 0 │

──╫───┼───┼───┼───┼───┼───┼───┼───┤

m3║ │ │ 0 │ 0 │ 0 │ 1 │ │ 0 │

──╫───┼───┼───┼───┼───┼───┼───┼───┤

m4║ │ 0 │ 0 │ 1 │ 1 │ 0 │ │ 0 │

──╫───┼───┼───┼───┼───┼───┼───┼───┤

m5║ 0 │ │ 1 │ 0 │ 0 │ 1 │ │ 0 │

──╫───┼───┼───┼───┼───┼───┼───┼───┤

m6║ │ │ 0 │ │ │ │ 0 │ 1 │

──╨───┴───┴───┴───┴───┴───┴───┴───┘

В незаполненных клетках сигналы безразличны.

Заметив, что umA = umB , uiB = ┐uiA , окончательно полу-

чаем:

╔══════════════════════════════════════════════╗

║ A ╔══════════════════════║═══════>D

║ ┌────┐ ─/┬┬──┬┐ ║ ┌────┐ ┌──────┐ ║

║ │ MUX│ C││RG││ ║ │M2\*8│ 1─>┤cr SM│ ║

╠═>╡0 │ ││ ││ ║ │ │ ├─ │ ║

I1══║═>╡1 ╞══════>╡│ │╞══╩══>╡ ╞═══>╡I1 │ ║ ┌─┐

║ ├ │ ││ ││ │ │ │ │ ║ │1│

║ │А │ W││ ││ ├─ │ │ S╞═╩>╡ o───>z

║ └A───┘ ─A┴┴──┴┘ └A───┘ │ │ │ │

║ └────┐ ┌─┘ B ┌────┘ ├─ │ └─┘

║ ┌────┐│ │─/┬┬──┬┐ │ ┌────┐ │ │

║ │ MUX││ │ C││RG││ │ │M2\*8│ │ p├─────────>p

╚═>╡0 ││ │ ││ ││ │ │ │ │ │

I2════>╡1 ╞│═══│═>┤│ │╞══│══>┤ ╞═══>╡I2 │

├ ││ │ ││ ││ │ │ │ │ │

│А ││ │ W││ ││ │ ├─ │ │ │ C

└A───┘│ │─A┴┴──┴┘ │ └A───┘ └──────┘ ─/┬┬─┬┐

│ │ │ └─┐ │ ┌─┐│ 1─>┤│T│├>rO

│ │ │ │ ├>┤ o┘ R W││ ││

├────┘ │ │ │ └─┘ ─A─A┴┴─┴┘

umB uwA uwB uiA urO uwO

---│--------│----│-----│----------------------│-│-----

y1 y2 y3 y4 y5 y6

║y1│y2│y3│y4│y5│y6│

══╬══╪══╪══╪══╪══╪══╡

m1║ │ │ │ │ 1│ 0│

──╫──┼──┼──┼──┼──┼──┤

m2║ 1│ 1│ 1│ │ 1│ 0│

──╫──┼──┼──┼──┼──┼──┤

m3║ │ 0│ 0│ 0│ │ 0│

──╫──┼──┼──┼──┼──┼──┤

m4║ 0│ 0│ 1│ 1│ │ 0│

──╫──┼──┼──┼──┼──┼──┤

m5║ 0│ 1│ 0│ 0│ │ 0│

──╫──┼──┼──┼──┼──┼──┤

m6║ │ 0│ │ │ 0│ 1│

──╨──┴──┴──┴──┴──┴──┘

- 7 -

Структура вычислителя:

┌────────────────────────────────┐

══>╡I1 │

│ │

══>╡I2 ОА D╞══>

│ │

┌──/C rO├──>

│ │ │

│ │z p umB uwA uwB uiA urO uwO │

│ └┬──┬──A───A───A───A───A───A─────┘

│ │ │ │ │ │ │ │ │

│ │ │ │ │ │ │ │ │

│ ┌V──V──┴───┴───┴───┴───┴───┴─────┐

│ │z p y1 y2 y3 y4 y5 y6 │

│ │ │

┴──/C │

│ УА │

──>┤rI │

└────────────────────────────────┘

УА должен выполнять следующий алгоритм автоматного типа,

представленный в виде блок-текста:

m1{xxxx10}

g1<<GO(rI;g1,m2)>>

m2{111x10}

m3{x000x0}

<<GO(z;g2,m6)>>

g2<<GO(p;m4,m5>>

m4{0011x0}

<<GO m3>>

m5{0100x0}

<<GO m3>>

m6{x0xx01}

<<GO g1>>

 \_МИКРОПРОГРАММИРОВАНИЕ. ОПРЕДЕЛЕНИЯ.

МИКРООПЕРАЦИЯ - базисное (элементарное) действие, необ-

ходимое для получения (вычисления) значения одной или более

переменных.

Микрооперация присваивания В=А означает, что переменные

левой части получают значения выражения из правой части.

Всегда разрядность левой части равна разрядности правой час-

ти. При этом биты, расположенные на одной и той же позиции в

левой и правой частях, равны.

Неиспользуемые разряды в левой части и произвольные зна-

чения в правой части микрооперации присваивания обозначаются

(х). Например:

(В[7],x,B[6..0]) = (A[7..0],x)

означает арифметический сдвиг влево на один разряд 8-разряд-

ного числа с присваиванием произвольного значения младшему

разряду и с потерей старшего после знака разряда. А, напри-

мер, микрооперация

(B[7..0],d) = (A[7],A[7..0])

означает арифметический сдвиг вправо на один разряд.

Микрооперация

(p,S[3..0]) = A[3..0] + B[3..0] + q

описывает действие, выполняемое стандартным 4-разрядным сум-

матором, если ( А,В,q ) являются его непосредственными входа-

ми, а ( р,S ) - выходами.

Микрооперация ( : ) - двоеточие - означает запоминание

(изменение значения) в памяти устройства. Переменная типа па-

мять сохраняет свое значение между двумя очередными присва-

иваниями.

- 8 -

Микрооперации всегда входят в состав микрооператоров.

МИКРООПЕРАТОР - набор взаимосвязанных микроопераций (или

одна микрооперация ), выполняемых одновременно и необходимых

для получения одного или более значений. Например:

( e,D:) = R1 + R2 + c

Фрагмент аппаратуры, реализующей этот микрооператор, мог бы

быть, например, таким:

┌───┐

c │MUX│

┌┬──┬┐ │ │ ┌───┐

││T │├───>┤0 │ ┌────┐ │MUX│ D

└┴──┴┘ ──>┤1 │ │ SM│ │ │ ┌┬──┬┐

──>┤А ├───>┤cr │ ═══>╡0 ╞═══>╡│RG│╞══>

├───┤ │ S╞═════>╡1 │ └┴──┴┘

R1 │MUX│ │ │ ═══>╡А │

┌┬──┬┐ │ │ │ │ └───┘

││RG│╞═══>╡0 ╞═══>╡I1 │ ┌───┐

└┴──┴┘ ══>╡1 │ │ │ │MUX│

══>╡А │ │ │ │ ├────────────>e

├───┤ │ p├─────>┤0 │

R2 │MUX╞═══>╡I2 │ ───>┤1 │

┌┬──┬┐ │ │ └────┘ ───>┤А │

││RG│╞═══>╡0 │ └───┘

└┴──┴┘ ══>╡1 │

══>╡А │

└───┘

Имена всех переменных, используемых в этом микрооператоре,

означают выполнение микроопераций коммутации ( транспортиров-

ки ). Значения переменных коммутируются на входы суммматора,

а результат суммирования - в места расположения переменных.

МИКРОБЛОК - набор микрооператоров, выполняемых одновре-

менно ( в одном такте ) и синхронно. В одном микроблоке любо-

му из битов присваивается только одно значение.

Синхронность означает, что во всех микрооператорах одно-

го микроблока используется только "старое" значение памяти.

Например:

{ (p,A):= A + B

(C,r):= A + D }

- и в том, и в другом микрооператоре используется одно и то

же старое значение А.

В то же время в микроблоке:

{ (C,x):= A + D

(x,A)= C + B }

в первом микрооператоре используется новое значение А , а во

втором - старое значение С. Разумеется, эти два действия вы-

полняются одновременнo на двух разных сумматорах.

При реализации микроблока { A:= B ; B:= 0 } обязательна

синхронная реализация В:=0 ( хотя обычно такое действие проще

реализовать асинхронно, но это приводит к ошибке ). Другой

правильный вариант: можно выполнить В:=0 асинхронно, но в

следющем такте.

Всегда предполагается, что предикат вычисляется вместе

(в одном такте) с тем микроблоком, за которым непосредственно

следует его использование.Таким образом, здесь, также как и в

микроблоке, используется старое значение памяти, существовав-

шее перед входом в микроблок. Это связано с особенностями

взаимодействия ОА и УА. Например:

- 9 -

█ █

█ CT:=(╪0)█ █ CT:=(╪0)█

█ █

│ │

┌────V───┐ ┌────V───┐

m1│ CT:=3 │ m1│ CT:=3 │

└────┬───┘ └────┬───┘

┌──────>│ ┌──────>│

│ ─V─ │ ─V─

│ / \ =0 │ / \ =0

│ <CT==0>─> │ <CT==0>─>

│ \\_\_\_/ │ \\_\_\_/

│ │╪0 │ │╪0

│ ┌────V───┐ │ ┌────V───┐

│m2│........│ │m2│........│

│ │ │ │ │ │

│ │CT:=CT-1│ │ │CT:=CT-1│

│ └────┬───┘ │ └────┬───┘

└───────┘ │ ┌────V───┐

│m3│........│

│ └────┬───┘

└───────┘

В первом случае цикл будет выполнен 4 раза; во втором - если

в микроблоке m3 нет операций, модифицирующих СТ, - 3 ра-

за. ( Обратите внимание на начальное значение СТ!)

МИКРОКОМАНДА - набор сигналов, поступающий из УА в ОА и

интерпретируемый как управляющий,т.е. необходимый для выпол-

нения всех микроопераций одного микроблока. Сигналы, входящие

в микрокоманду, могут принимать участие в микрооперациях и в

качестве информационных.

Микрокомандой также иногда называют слово управляющей

памяти (обычно ПЗУ), являющееся частью УА. Для различения

этих понятий слово управляющей памяти будем называть МИКРО-

ИНСТРУКЦИЕЙ.

МИКРОПРОГРАММА СОДЕРЖАТЕЛЬНАЯ - алгоритм, представленный

в виде микроблоков и предикатных блоков в одной из принятых

форм, например, в виде блок-схемы или блок-текста.

МИКРОПРОГРАММА КОДИРОВАННАЯ - аппаратная форма содержа-

тельной микропрограммы в виде кодов, заполняющих управляющую

память.

 \_КАНОНИЧЕСКАЯ СТРУКТУРА ОПЕРАЦИОННОГО АВТОМАТА

В общем случае каноническая структура операционного ав-

томата имеет вид:

███████████████████████████████████████████████████████████

█ █

█ ┌──────────┐ ┌┬──────┬┐ ┌──────────┐ ┌───────┐ █

██>╡коммутация│ ││память││ │коммутация│ │функции▐███

│ ▐███>╡│ │▐██>╡ ▐██>╡ │

██>╡ │ ││ ││ │ │ │ ▐███>

└─A────────┘ ─/─┴┴───A──┴┘ └──A───────┘ └──A────┘

█ ┌─┐│CC █ █ █

█ SYN─>┤&├┘ █ █ █

█ ┌┤ │ █ █ █

█ yC│└─┘ █ █ █

└────────────────────────────────────────────────┘

сигналы управления

Столь четкого разграничения операций на зоны (память, комму-

тация, функции) может и не быть. Например, такие широко ис-

пользуемые функции как сдвиги либо хорошо совмещаются с

коммутацией, либо интегрируются с регистрами хранения.Также

часто интегрируются с хранением функции инкремента и

- 10 -

декремента (счетчики обычные и реверсивные).

Особо выделим сигнал yС, управляющий доступом синхросиг-

налов в ОА. Такой вариант управления, называемый условной

синхронизацией, позволяет запретить любые изменения памяти ОА

в каком-либо такте. Причем, если рабочим является срез (зад-

ний фронт) сигнала синхронизации, то используется элемент И,

как и показано на рисунке.Если же рабочим является фронт (пе-

редний фронт) сигнала, то используется элемент ИЛИ. (Первый

перепад сигнала синхронизации в новом такте не должен быть

рабочим.)

 \_ОПТИМИЗАЦИЯ ОПЕРАЦИОННОГО АВТОМАТА

При проектировании вычислительного устройства основными

являются ограничения на:

1)- время вычисления;

2)- объем аппаратуры, реализующей вычисления;

3)- тип применяемых базовых функций.

ОПТИМИЗАЦИЯ АПППАРАТУРЫ ПРИ СОХРАНЕНИИ МИНИМАЛЬНОГО ВРЕМЕНИ

Алгоритм функционального типа обладает максимальным по-

тенциальным параллелизмом (в рамках данной алгоритмической

идеи), и,следовательно, его реализация в виде КС обладает

максимальным быстродействием по сравнению с любыми другими

вычислителями. Невозможность реализации вычислителя в виде КС

может быть обусловлена следующими причинами:

- слишком велик объем аппаратуры КС;

- функциональное представление и его реализация являются

статическим отображением входных объектов на выходные: это

исключает возможность работы с объектами, которые "ведут се-

бя" более сложно во времени; невозможно также реализовать

принципиально рекуррентные алгоритмы (см.,например,алгоритм

Евклида для нахождения НОД).

Тем не менее, если формально алгоритм функционального

типа может быть записан, то проектирование устройства надо

начинать с записи именно такого алгоритма.

Минимизация аппаратуры "сложной" КС с переходом на про-

цедурный вариант реализации связан с "экономией" числа опера-

ционных элементов, т.е. со слиянием некоторых из них в один

функциональный модуль, выполняющий эти операции по очереди.

Такое решение потребует запоминания всех переменных (входных

и выходных),связанных с выполнением этих операций. Требуется

также управление памятью, связанной с этим функциональным мо-

дулем, а также - может быть - управление самим функциональным

модулем, если он объединил существенно различные функции.

Переход к процедурной реализации и, следовательно, к

дискретизации времени неизбежно увеличит время вычисления од-

ного результата - даже при реализации структуры подобной КС.

При этом, как ни странно, может уменьшиться время следующих

друг за другом вычислений именно за счет дискретизации време-

ни и применения так называемых "конвейерных" вычислений - но

об этом речь пойдет в другом курсе.

Рассмотрим возможность сокращения аппаратуры без увели-

чения времени решения, достигнутого в алгоритме функциональ-

ного типа. Сопоставим схеме устройства, реализующего алгоритм

функционального типа, простую модель в виде направленного

графа. Вершины графа будут соответствовать операциям, а дуги

- переменным алгоритма. Назовем такой граф сигнальным (графом

потоков данных). Заметим, что сигнальный граф всегда будет

ациклическим.

Минимальность времени вычислений сохранится, если совме-

щать в один функциональный модуль операции, которые располо-

жены на одном и том же пути в сигнальном графе, либо на аль-

тернативных путях решения.

- 11 -

 \_МИНИМИЗАЦИЯ АППАРАТУРЫ

Может оказаться, что на одном пути в сигнальном графе

расположены операции, плохо или вовсе не совмещаемые друг с

другом (т.е. совмещение не дает экономии в аппаратуре функци-

онального модуля). Возможно также, что проведенная минимиза-

ция, сохраняющая минимальное время, не позволяет выполнить

ограничение на объем аппаратуры. В таком случае необходима

более глубокая минимизация,охватывающая параллельные ветви

сигнального графа. Минимизация должна быть взаимосвязанной по

всем компонентам ОА: по памяти, функциональным модулям и ком-

мутации.

В настоящее время процедуры минимизации не формализованы

и сводятся к перебору "правдоподобных" вариантов.

Можно предложить следующую последовательность действий:

1)- все "похожие" функции (операции) реализовать на одном

функциональном модуле, например, все суммирования выполнять

на одном сумматоре;

2)-скорректировать алгоритм так, чтобы в одном микроблоке не

выполнялось более одной операции на одном и том же функци-

ональном модуле;

3)- минимизировать память автомата, т.е. число запоминаемых

промежуточных переменных;

Выполнение этих этапов может привести к усложнению ком-

мутации, а значит, и к увеличению этой компоненты в аппарату-

ре ОА. Если ограничение по объему аппаратуры все еще наруше-

но, то повторить этапы 1 - 3 с другим вариантом объединения

функциональных модулей и памяти.

При реализации ОА - во избежание ошибок -необходимо бук-

вально следовать описанию алгоритма, но в то же время, при

проектировании самого алгоритма надо представлять себе реали-

зацию микроблоков. Реализация одних и тех же вычислений может

быть существенно различной по объему аппаратуры.

Например, вычисления в цикле потребуют реализации:

─┬─

│

┌───────V───────┐ A ┌────┐

│ J:=0 │ ┌┬──┬─┐ │ MUX│ ┌────┐

└───────┬───────┘ ││RG│0├───>┤0 │ │ f │

┌──────┐ │ ││ │.│ . │. │A[J] │ │

│ ┌────V──V───────┐ ││ │.│ . │. ├────>┤ │

│ │ ..... │ ││ │.│ . │. │ │ │ B

│ │ │ ││ │ │ │ │ │ ╞══>

│ │ B= f(...,A[J])│ ││ │K├───>┤K │ │ │

│ │ │ ││ │.│ │. │ ══>╡ │

│ │ J:=J+1 │ ││ │.│ │. │ │ │

│ └───────┬───────┘ ││ │.│ │. │ │ │

│ ─V─ └┴──┴─┘ ├─ │ └────┘

│ <K / \ =K J═════════>╡А │

└──────<J==K>─────> └────┘

\\_\_/

(реализация счетчика J не показанa).

- 12 -

Запишем этот фрагмент алгоритма иначе так, чтобы нужный

бит извлекался за счет сдвига в регистре D. Тогда получим:

───┬── A D

│ ┌┬──┬┐ ┌┬──┬─┐ A[J] ┌─────┐

┌───────V───────┐ ││RG││ ││RG│0├─────>┤ f │

│ J:=0 │ ││ ││ ││ │.│ │ │

│ │ ││ ││ ││->│.│ │ │ B

│ D:=A │ ││ │╞══════>╡│ │.│ │ ╞══>

└───────┬───────┘ ││ ││ ││ │ │ │ │

┌──────┐ │ ││ ││ ││ │K├ │ │

│ ┌────V──V───────┐ ││ ││ x ──>┤Dn │.│ │ │

│ │ ..... │ ││ ││ ││ │.│ ══>╡ │

│ │ │ ││ ││ S W││ │.│ │ │

│ │ B= f(...,D[0])│ └┴──┴┘ ─v─v┴┴──┴─┘ └─────┘

│ │ │

│ │ (D,x):=(x,D) │

│ │ │

│ │ J:=J+1 │

│ └───────┬───────┘

│ ─V─

│ <K / \ =K

└──────<J==K>─────>

\\_\_/

Промежуточный регистр A введен для общности, если потребуется

сохранить слово А (чаще всего он и не нужен).

Другой пример: фрагмент алгоритма, реализующий регуляр-

ную запись отдельных бит слова и его реализация имеют вид:

───┬── ┌┬─┬┐B[0]

│ a ────────────┬─────>┤│T│├────>

┌───────V───────┐ │ W││ ││

│ J:=0 │ ┌───┐ │ ─A┴┴─┴┘

└───────┬───────┘ │DC │ ┌──┼─────┘| |

┌──────┐ │ │ 0├─┘ │ | |

│ ┌────V──V───────┐ │ .│. │ ┌┬─┬┐B[K]

│ │ ..... │ │ .│. └─────>┤│T│├────>

│ │ │ │ .│. W││ ││

│ │ a=f(...) │ J ══>╡ │ ─A┴┴─┴┘

│ │ │ │ K├──────────┘

│ │ B[J]:=a │ │ .│

│ │ │ │ .│

│ │ J:=J+1 │ │ .│

│ └───────┬───────┘ └───┘

│ ─V─

│ <K / \ =K

└──────<J==K>─────>

\\_\_/

Слово В нельзя реализовать в виде регистра, а только в виде

отдельных триггеров.

Можно формировать слово с использованием операции сдви-

га при обязательном условии D[K..0], тогда алгоритм и его ре-

ализация имеют вид:

- 13 -

───┬──

│ D B

┌───────V───────┐ ┌──┬──┬┐ ┌┬──┬┐

│ J:=0 │ │ │RG││ ││RG││

└───────┬───────┘ │ │->││ ││ ││

┌──────┐ │ a │ │ │╞═════>╡│ ││

│ ┌────V──V───────┐ ──>┤Dk│ ││ ││ ││

│ │ ..... │ S│ │ ││ W││ ││

│ │ │ ─v┴──┴──┴┘ ─v┴┴──┴┘

│ │ a=f(...) │

│ │ │

│ │ (D,x):=(a,D) │

│ │ │

│ │ J:=J+1 │

│ └───────┬───────┘

│ ─V─

│ <K / \ =K ┌────┐

└──────<J==K>──>┤B:=D├>

\\_\_/ └────┘

В этом случае, так же, как и в предыдущем, чаще всего не ну-

жен промежуточный регистр (В).

 \_УНИВЕРСАЛЬНЫЙ ОА

Использование при проектировании универсальных ОА с за-

ранее фиксированной и минимизированной структурой оправдано

тем, что такие универсальные ОА изготавливаются промышлен-

ностью в виде БИС большим тиражом и поэтому они сравнительно

дешевы. Такие универсальные ОА входят в микропроцессорные на-

боры 582, 583, 584, 588, 589, 1800, 1804 и т.д., которые на-

зываются микропрограммируемыми, секционными, разрядно-модуль-

ными.

В основе перечисленных универсальных ОА лежит следующая

структура:

╔══════════════════╦═══════════════════════════╗

║ ║ ║

║ ║ SYN┐ ACC ║

║ ┌─┬─────┬┐ ║ ─/┬┬──┬┐ ┌─────┐ ║

║ │ │ RGF ││ ║ C││RG││ │ ALU │ ║

║ │ │ ││ ║ ││ ││ │ │ ║

║ │ │ ││ ╚════>╡│ │╞═════>╡ │ ║

║ │ │ ││ ││ ││ │ ╞═══╩═>DO

╚═══>╡D│ ││ └┴──┴┘ │ │

│ │ ││ T │ │

│ │ ││ ┌┬──┬┐ │ ╞═════>P

│ │ ││ ││RG││ │ │

│ │ │╞═════════>╡│ │╞═════>╡ │

│ │ ││ ││ ││ │ │

C W│А│ ││ C││ ││ ╔═>╡ │

─o─A┴A┴─────┴┘ ─┬┴┴──┴┘ ║ └──A──┘

SYN┘ │ ║ SYN┘ ║ ║

│ ║ ║ ║

yW YA DI═════╝ YF

ALU - арифметико-логическое устройство - комбинационная

схема с небольшим, но универсальным набором арифметических и

логических операций.

RGF - регистровый файл - адресуемая память RAM со стати-

ческой синхронизацией при записи.

RG'T' - регистр-фиксатор со статической синхронизацией.

RG'АCC' - регистр-аккумулятор с динамической синхрониза-

цией.

DI,DO - входная и выходная информационные шины.

- 14 -

Р - предикатные сигналы (флажки).

YF - сигналы управления выбором функции.

YA - адрес чтения и/или записи RGF.

yW - разрешение записи в RGF.

Память сравнительно большого объема, какой является RGF,

дешевле реализовать со статической синхронизацией. Для то-

го,чтобы такая память могла работать в замкнутом информацион-

ном кольце и при этом не возникали бы гонки, добавляется еше

один промежуточный регистр RG'T' со статической синхрониза-

цией. Если передний фронт является рабочим для регистров уп-

равляющего автомата и RG'ACC', то на первой фазе синхрониза-

ции при SYN=1 информация читается из RGF; при этом RG'T'

прозрачен. На следующей фазе синхронизации при SYN=0 информа-

ция фиксируется в RG'T', т.е. он закрыт для записи, а запись

(если она разрешена) производится в RGF. Фиксируется информа-

ция в RGF и RG'ACC' с началом следующего такта, т.е. на пе-

реднем фронте сигнала.

 \_ВЗАИМОДЕЙСТВИЕ ОА и УА

Для исключения гонок при взаимодействии ОА и УА будем

проектировать УА как автомат Мура. Схема их взаимодействия

может быть представлена в виде:

╔══════════════════════════╗

║┌────┐ ┌┬──┬┐ ┌────┐ ║

╚╡ CS │ ││RG││ │CS ╞<╝

│ ╞<═╦═╡│ │╞<══╡ │

┌───┤ b │ ║ ││ ││ │ c ├<────┐

│ └────┘ ║ └┴──┴┴A─ └────┘ │

│ ┌────┐ ║ └───────────┐ │

│ │CS ╞<═╝ │ │

│┌──┤ a ├<───────────────────┐ │ │

ОА ││ └────┘ │ │ │

----││----------------------------│-│-│--

УА ││РА┌────┐ ┌┬──┬┐ ┌─────┐│ │ │┐

│└─>┤ CS│ ││RG││ │ CS ├┘ │ ││

└──>┤ ╞════>╡│ │╞═>╡ ├──┘ ││Y

РВ │ │ ││ ││ │ ├────┘│

╔>╡ p │ ││ ││ │ y ╞═╗ ┘

║ └────┘ └┴──┴┘ └─────┘ ║

╚════════════════════════════╝

Отметим, что РА(t)=f(Y(t)) зависит без сдвига от сигналов

управления,

PB(t+1)=F(Y(t)) зависит со сдвигом от сигналов

управления,

где РА и РВ - предикатные перемнные.

Продолжительность такта работы схемы определяется наибо-

лее длинными цепями между регистрами. Для данной схемы, кото-

рую будем называть последовательной схемой взаимодействия,

зададимся (так чаще всего бывает), что такой критической

цепью является цепь (CSy,CSa,CSp,RG). Поэтому длительность

такта определяется:

Т > ty + ta + tp + trg,

где tj- время установления соответствующего компонента цепи.

Чтобы сократить длину этой цепи, применяют другой вари-

ант взаимодействия автоматов - конвейерный:

- 15 -

╔══════════════════════════╗

║┌────┐ ┌┬──┬┐ ┌────┐ ║

╚╡ CS │ ││RG││ │CS ╞<╝

│ ╞<═╦═╡│ │╞<══╡ │

┌───────────┤ b │ ║ ││ ││ │ c ├<────┐

│ FF └────┘ ║ └┴──┴┴A─ └────┘ │

│ ┌┬──┬┐ ┌────┐ ║ └───────────┐ │

│┌─┤│RG│╞<══╡ CS ╞<═╝ │ │

││ ││ ││ │ a ├<───────────────────┐ │ │

││ └┴──┴┴A─ └────┘ │ │ │

ОА ││ └──────────────────────────┐ │ │ │

---││----------------------------------│-│-│-│--

УА ││ MK │ │ │ │

││ PA ┌────┬────┐ ┌┬──┬┐│ │ │ │┐

│└────>┤ CS│ CS │ ││RG│├┘ │ │ ││

│ РВ │ │ │ ││ │├──┘ │ ││Y

└─────>┤ │ ╞═══════════>╡│ │├────┘ ││

│ │ │ ││ │├──────┘│

╔>╡ p │ y │ ││ │╞═╗ ┘

║ └────┴────┘ └┴──┴┘ ║

╚═══════════════════════════════╝

При этом варианте взаимодействия такой длинной цепи, как

в предыдущем случае, не возникает.Эта цепь разделена регис-

трами RG'FF' (регистр флажков) и RG'MK' (регистр микрокоман-

ды) на две цепи. Продолжительность такта становится меньше и

ее можно определить следующим образом:

T > max( ta,(tp + ty) )+ trg ,

При конвейерном варианте взаимодействия

PA(t+1)=f(Y(t)), т.е. и эти значения стали зависить со

сдвигом от сигналов управления. Тогда фрагмент микропрограммы

mS{...;pA=f(...)}

<< GO(pA;mi,mj)>>,

выполняемый в последовательной схеме за один такт, в кон-

вейерном варианте за один такт выполнен быть не может и дол-

жен быть модифицирован следующим образом:

mS{...,pA=f(...)}

mS'{нет операции}

<< GO(pA;mi,mj)>>

Таким образом, время выполнения этого фрагмента не только не

уменьшилось, но даже возросло несмотря на уменьшение продол-

жительности каждого из тактов. Зато во всех остальных случа-

ях (при безусловных переходах, при переходах по значению РВ)

время выполнения микропрограммы уменьшается.

 \_НАЧАЛЬНАЯ ИНИЦИАЛИЗАЦИЯ СИНХРОННОЙ СХЕМЫ

Пусть устройство, кроме сигнала синхронизации SYN, имеет

еще один сигнал Н, обозначающий начало и конец синхронной ра-

боты устройства. При Н=0 - нерабочее состояние - можно выпол-

нять начальную установку значений памяти устройства. Измене-

ние значения Н с 0 на 1 происходит в случайный момент времени

(асинхронно), но при этом начальный такт работы устройства

должен быть полным. "Затягивание" асинхронного сигнала Н в

синхронный режим происходит с помощью простейшего синхронного

автомата с диаграммой:

┌──────────┐ ┌────────┐

V 0H/CONST│ V 1H/SYN│

█▀▀▀█────────┘ █▀▀▀█──────┘

>▌ 0 ▐──────────────>▌ 1 ▐──────┐

█▄▄▄█ 1H/CONST █▄▄▄█ 0H/X │

л │

└────────────────────────────┘

У этого автомата простейшей является функция переходов, так

как диаграмма автомата совпадает с диаграммой переходов

- 16 -

D-триггера.

Схема автомата вместе с цепями условной синхронизации

выглядит следующим образом (для синхронизации по фронтам):

а)-по переднему фронту, б)- по заднему фронту:

┌──┐ ┌──┐

SYN ──┬──────────┤ 1├── CC SYN ──┬──────────┤ &├── CC

│ ┌─┬─┐ ┌─┤ │ │ ┌─┬─┐ ┌─┤ │

└─/C│T│ │ └──┘ └─\C│T│ │ └──┘

│ │ ├ │ │ │ ├──┘

┌─┤D│ │ │ ┌─┤D│ │

│ ├─┤ o──┘ │ ├─┤ o─

├─oR│ │ ├─oR│ │

H │ └─┴─┘уст. нач. зн. H │ └─┴─┘уст. нач. зн.

──┴─────────────────── ──┴───────────────────

Такая разница в цепях условной синхронизации, как уже объяс-

нялось выше, определяется тем, что первый перепад сигнала СС

не должен быть рабочим.