EESTI NSV TEADUSTE AKADEEMIA TOIMETISED. XIII KÖIDE
FÜÜSIKA-MATEMAATIKA- JA TEHNIKATEADUSTE SEERIA. 1964, NR. 4

ИЗВЕСТИЯ АКАДЕМИИ НАУК ЭСТОНСКОЙ ССР. ТОМ XIII
СЕРИЯ ФИЗИКО-МАТЕМАТИЧЕСКИХ И ТЕХНИЧЕСКИХ НАУК. 1964, № 4

https://doi.org/10.3176/phys.math.tech.1964.4.09

ОДНО КОДИРОВАНИЕ ДЛЯ МОДЕЛИРОВАНИЯ ЦИФРОВЫХ СХЕМ

Х. САЛУМ

Произвольная композиция автоматов (в смысле, данном В. М. Глушковым [1]) может быть охарактеризована

1) операторами отдельных автоматов

2) входными связями (входными векторами) этих автоматов

3) выходными связями (выходными векторами) этих автоматов

и промоделирована на МЦВМ. Считая для каждого конкретного случая кодирование состояний, входов и выходов уже выбранным, можно составить формальное кодирование для описания произвольной структурной схемы. Кодирование упрощается, если выделить некоторое количество элементов памяти, управляемых программой, для моделирования работы схемы.

Основными составными частями кодирования являются:

1) операторы отдельных автоматов, записанные в виде подпрограмм,

2) массивы данных о состояниях, параметрах и связях в схеме.

В данной работе предлагается предварительный вариант кодирования для моделирования на МЦВМ произвольных цифровых схем. Исходные данные о схеме должны быть представлены в виде логической схемы на уровне логических элементов и регистров.

Моделирование работы структурной схемы производится по моментам автоматного времени.

Основными символами синтаксической записи применяемых понятий являются символы, использованные для описания АЛГОЛ-60 в [2]. К этому списку добавлен символ для обозначения адреса второго ранга [3]. Часть обозначений, особенно в разделах 1 и 2, еще не полностью определилась и дана описательно.

Для упрощения кодирования часть элементов памяти выделена и управляется моделирующей программой. Время моделирования одного такта работы АУ ЦВМ, подобной М-3, равно 6—10 *сек.*

Работа выполнена в Институте кибернетики Академии наук Украинской ССР под руководством академика АН СССР доктора физико-математических наук В. М. Глушкова.

1. Элементы

Каждому логическому элементу в подлежащей моделированию схеме сопоставляется кодирующее понятие — *элемент*. Кодовый элемент характеризуется логическим содержанием и обозначается соответствую-

щей этому содержанию большой латинской буквой. Элементы с одинаковым логическим содержанием — для краткости называемым дальше *типом элемента* — объединяются в двухразмерный массив, и каждому отдельному элементу присваиваются 2 индекса. Элементы одного типа с одинаковым первым индексом составляют *регистр*.

Элемент типа *«активизированный»* по своему существу является промежуточным между другими элементами и регистрами. Его назначение — принудительно управлять другими элементами.

При моделировании программа сопоставляет каждому элементу *оператор*, информационно моделирующий работу описываемого элемента схемы. Соответствие между элементами и операторами дано в табл. 1. Для моделирования работы оператор *шины* не нужен, так как шины выполняют только функцию связи между другими элементами.

Таблица 1

№	Название элемента	Обозначение элемента	Название оператора
	Элементы памяти		
1	*Триггер*	*T*	*оператор триггер*
2	*Задержка*	*V*	*оператор задержка*
	Элементы связи		
3	*Шина*	*S*	
	Комбинационные элементы		
4	*Сборка*	*L*	*оператор сборка*
5	*Конъюнкция*	*K*	*оператор конъюнкция*
6	*Дешифратор*	*D*	*оператор дешифратор*
	Активизированные элементы		
7	*Активизированная сборка*	*AL*	*оператор активизированная сборка*
8	*Активизированная конъюнкция*	*AK*	*оператор активизированная конъюнкция*
9	*Активизированная задержка*	*AV*	*оператор активизированная задержка*
10	*Дифф А*	*FA*	*оператор дифф А*
11	*Дифф В*	*FB*	*оператор дифф В*
12	*Активный элемент*	*A*	*оператор выход*

Входящий в состав списка элементов *активный элемент* необходим в основном для моделирования внешних управляющих сигналов и формирователей.

Получаемая при записи модель схемы не содержит в явном виде обозначений отдельных элементов. Кодовый *элемент* является понятием, объединяющим в одно целое соответствующие ему массивы. Индексы, соответствующие каждому элементу, указывают связанные с ним элементы массивов.

В предлагаемом кодировании для записи цифровых схем используются следующие массивы:

1. Входных связей *RS*
2. Выходных связей *RR*
3. Внутренних связей *RE*
4. Состояний *R*1 и *R*2.

Все эти массивы подробно описаны в разделе 3. Первый символ идентификатора массива — *R* — является обобщенным для краткости за-

писи. При записи схемы в кодированном для моделирования виде *R* заменяется соответствующим для типа элемента символом (K|L|D|*T*|*V*|*S*|*AK*|*AL*|*AV*|*FA*|*FB*|*A*).

Выбор набора элементов, использованного в данном предварительном варианте кодирования, сделан эмпирически, причем система является логически полной.

Любой из элементов является частным случаем полного абстрактного автомата (Мили или Мура).

Комбинационный элемент является элементом без памяти (тривиальный случай автоматов Мили).

Сигнал на выходе (двухвходовой) *конъюнкции K* равен конъюнкции сигналов, поданных на входы, а сигнал на выходе (двухвходовой) *сборки L* равен дизъюнкции входных сигналов. Кратной записью двухвходовой сборки и двухвходовой конъюнкции можно описать сборки или конъюнкции с произвольным числом входов.

Дешифратор D является произвольным (р, к)-полюсником без памяти, причем связи между входами и выходами дешифратора описываются массивом внутренних связей *DE* в форме набора конституэнт единицы для каждого из выходов. В случае, если выходы дешифратора прямо управляют элементами памяти и шинами, выходная ступень дешифратора выделяется и запысывается при помощи *AL* или *AK*. Дешифратор является неактивизированным элементом, имеющим выходной массив *DR*.

Триггер T и *задержка V* являются элементарными автоматами Мура с приведенными ниже отмеченными таблицами состояний (табл. 2 и табл. 3 соответственно):

Таблица 2

	0	1
	0	1
x	0	0
y	1	1
z	1	0

Таблица 3

	0	1
	0	1
x	0	0
y	1	1

где x — соответственно гасящий или нулевой,
y — возбуждающий или единичный и
z — счетный вход.

Шина S является элементом связи между активизированными конъюнкциями. Шина возбуждается (*S*1 : : = 1) на время одного цикла моделирования от выхода подсоединенной к ней активизированной конъюнкций (имеющей нагрузку <выход на шины>).

Дифференцирующие элементы дифф A и *дифф B* выдают сигнал для работы входящего в них активного элемента соответственно в случае перехода элемента памяти из состояния 0 в состояние 1 и наоборот.

2. Регистры

Регистром является набор элементов одного типа с одинаковым первым индексом.

Идентификатор элемента нигде, кроме семантических примечаний, не используется самостоятельно, а только в качестве начального символа для обозначения соответствующего массива. Исходя из этого, для

экономии идентификаторов, сделано допущение, что *при записи схемы идентификатору элемента присваивается численное значение*, указывающее количество *регистров*, состоящих *из элементов данного типа*.

В разделе 1 было указано, что для краткости записи свойств, общих для всех элементов, вместо перечисления идентификаторов всех элементов можно писать R. R является максимальным значением первого индекса массива. Регистры из элементов каждого типа нумеруются отдельно, по порядку, начиная с 1 по R (т. е. по численному значению идентификатора типа элементов в регистре).

Длина регистра i — число *ni* — указывает количество объединенных в один регистр элементов. Число *ni* записывается в массиве общих данных элементов $RO[i, 1]$.

В принципе регистр образуется путем произвольного присвоения одинакового первого индекса элементам одного типа. Практически целесообразно объединять в регистры элементы с максимально подобными внешними связями (напр., конъюнкции, на первый вход которых поступает общий управляющий сигнал, триггеры с одинаковым набором входов, соединенных с одними регистрами активных или активизированных элементов, и т. д.). Такие регистры называются однородными и моделирующий их оператор может быть упрощен. Признак *однородности регистра* записывается в массиве общих данных элементом $RO[i, 2]$.

При желании можно отметить и этим исключить регистры, которые при моделировании нужно пропустить. *Используемость регистра* отмечается элементом $RO[i, 3]$ массива общих данных. Предполагается, что неиспользуемые регистры (регистры, которые нужно пропустить) будут отмечены $RO[i, 3] ::= 1$, а остальные нулем или отсутствием признака.

Элементы типа дешифратор не могут образовать регистра. Это сделано для упрощения записи связей. Аналогией регистров у дешифраторов являются набор входов и набор выходов (так как с ними связаны соответствующие массивы DS, DR, $D1$).

3. Массивы информации

3.1. Массив общих данных

<*массив общих данных*> ::= integer array RO [1 : R, 1 : 3], причем:

а) для регистров $R \neq D$

$RO[i, 1] ::=$ <*длина регистра i*>
$RO[i, 2] ::=$ <*однородность регистра i*>
$RO[i, 3] ::=$ <*используемость регистра i*>

б) для дешифраторов

$DO[i, 1] ::=$ <*число входов дешифратора i*>
$DO[i, 2] ::=$ <*число выходов дешифратора i*>
$DO[i, 3] ::=$ <*используемость дешифратора i*>

3.2. Массив входных связей

<*массив входных связей*> ::= array RS[1 : R, 1 : $RO[i, 1]$, 1 : 2 (максимально)]

причем:

а) массивы $DS[i,j]$, $VS[i,j]$ и $AVS[i,j]$ имеют размерность $[1:R, 1:RO[i,1]]$, и в ячейке $RS[i,j]$ может быть записано $R1[k,m]$ или $\neg R1[k,m]$;

б) массивы $KS[i,j,1]$, $KS[i,j,2]$, $LS[i,j,1]$, $LS[i,j,2]$, $AKS[i,j,1]$, $AKS[i,j,2]$, $ALS[i,j,1]$, $ALS[i,j,2]$ соответственно попарно образуют массив размерности $[1:R, 1:RO[i,1], 1:2]$, и в каждой из ячеек $RS[i,j,p]$ может быть записано $R1[k,m] \mid \neg R1[k,m]$;

в) массивы $FAS[i,j]$ и $FBS[i,j]$ имеют размерность $[1:R, 1:RO[i,1]]$, и в каждой из ячеек $RS[i,j]$ может быть записано $T1[k,m] \mid V1[k,m]$.

Массив RS не является АЛГОЛьным, так как записывается не число (**Boolean**, **integer**, **real**), а адрес, имеющий индексы.

В предлагаемом кодировании не предвидено отдельного элемента для моделирования *инвертора*, и запись $\neg R$ означает, что сигнал на данный вход идет или от инверсного выхода триггера, или через инвертор.

3.3. Массив выходных связей

<массив выходных связей (м.вых.с.)> ::= **integer array** RR ::= *<м.вых.с. дешифраторов>|<м.вых.с. активизированных элементов>*

<м.вых.с. дешифраторов> ::= **integer array** $DR[1:D, 1:DO[i,2]]$

<м.вых.с. активизированных элементов> ::= **integer array** $RR[1:R, 1:RO[i,1], 1:3]$

$DR[i,j]$ ::= *<количество конституэнт единицы в выражении для j-го выхода дешифратора i>*

$RR[i,j,1]$ ::= *<возбуждающий вход триггера>|<гасящий вход триггера>|<счетный вход триггера>|<выход на шины>|<выход на задержку>|<прочие нагрузки>|<отключено>*

$RR[i,j,2]$ ::= $[1:R]$ ::= *<номер регистра нагрузки k>*

$RR[i,j,3]$ ::= $[1:RO[k]]$ ::= *<номер элемента в регистре k>*

<возбуждающий вход триггера> ::= *<только возбуждающий вход>|<возбуждающий и гасящий входы>|<возбуждающий и счетный входы>|<возбуждающий, гасящий и счетный входы>*

<гасящий вход триггера> ::= *<только гасящий вход>|<гасящий и счетный входы>|<гасящий и возбуждающий входы>|<гасящий, счетный и возбуждающий входы>*

<счетный вход триггера> ::= *<только счетный вход>|<счетный и возбуждающий входы>|<счетный и гасящий входы>|<счетный, возбуждающий и гасящий входы>*

<только возбуждающий вход> ::= 1

<только гасящий вход> ::= 2

<только счетный вход> ::= 3

<возбуждающий и гасящий входы> ::= 4

<гасящий и счетный входы> ::= 5

<счетный и возбуждающий входы> ::= 6

<возбуждающий и счетный входы> ::= 7

<гасящий и возбуждающий входы> ::= 8

<счетный и гасящий входы> ::= 9

<возбуждающий, гасящий и счетный входы> ::= 10

<*гасящий, счетный и возбуждающий входы*> : : = 11
<*счетный, возбуждающий и гасящий входы*> : : = 12
<*выход на шины*> : : = 13
<*выход на задержку*> : : = 14
<*прочие нагрузки*> : : = 15
<*отключено*> : : = 16

$RR[i, j]$ (для $R \neq D$) характеризует нагрузку, идущую за активизированным элементом.

Нагрузка типа <*только ... вход*> (*режимы* 1, 2, 3) означает, что в схеме не могут одновременно (т. е. в течение одного цикла моделирования) поступить сигналы на разные входы одного и того же триггера. Это соответствует условию надежной работы реальной схемы.

Нагрузка типа <*A и B входы*> (*режимы с* 4 *по* 9) или типа <*A, B и C входы*> (*режимы* 10, 11, 12) (где *A*, *B*, *C* обозначают входы триггера в произвольном порядке) означает, что выход данного активного элемента подсоединен к входу *A* триггера, но одновременно может поступить сигнал на *B* и (или) *C* входы того же триггера.

В режиме <*прочие нагрузки*> активизированный элемент становится эквивалентным неактивизированному, а в режиме <*отключено*> $R1[i, j]$ гасится после проверки режима. Режим <*отключено*> может применяться для исключения не нужных на данном этапе моделирования активизированных элементов без выбрасывания всего регистра и перезаписи остальных, еще нужных элементов в конец массива.

Массивом *RS* обладают только активизированные элементы и дешифратор.

3.4. Массив внутренних связей

3.4.1. Синтаксис. <*массив внутренних связей*> : : = <*массив внутренних связей дешифраторов*>

<*массив внутренних связей дешифраторов*> : : = **boolean array** $DE[1 : D, 1 : DO[i, 1], 1 : DO[i, 2], 1 : DR[i, k]]$

$DE[i, j, k, l] : : = 1 \mid 0$

3.4.2. Семантика. Одна строка по j матрицы DE перечисляет значения входных переменных в l-й конституэнте единицы в выражении для k-го выхода дешифратора i.

3.5. Массив состояний

3.5.1. Синтаксис. <*массив состояний*> : : = <*массив исходных данных*> | <*массив результатов*>

<*массив исходных данных*> : : = **boolean array** $R1[1 : R, 1 : RO[i, 1]]$

<*массив результатов*> : : = **boolean array** $R2[1 : R, 1 : RO[i, 1]]$

3.5.2. Семантика. Массив исходных данных $R1$, свойственный всем элементам, служит информацией для работы программы и выражает состояние схемы в момент начала (нового цикла) моделирования (т. е. в предшествующий момент автоматного времени). В массиве результатов $R2$ накапливается состояние схемы к концу (такта) моделирования (т. е. состояния схемы в настоящий момент автоматного времени). При переходе к новому циклу моделирования (в момент изменения автоматного времени на единицу), информация из массива $R2$ переводится в массив $R1$. Массив $R2$ при этом гасится. Массив $R2$ имеется только у элементов памяти.

ПРИМЕРЫ КОДИРОВАНИЯ

Пример 1

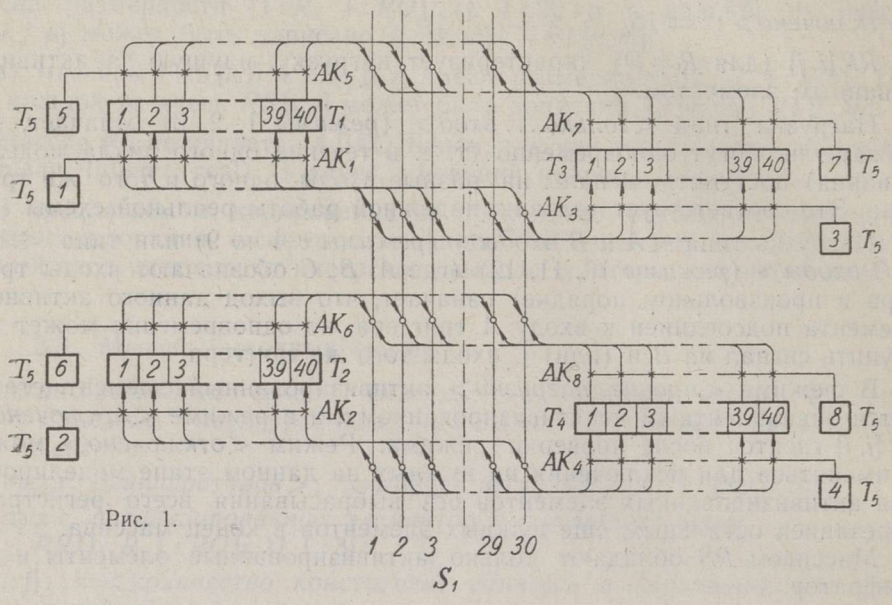

Рис. 1.

Схема на рис. 1, работа которой состоит в пересылке информации между регистрами через общие шины, записывается:

$AK = 8$ — вентили приема и передачи информации

$T = 5$ — 4 регистра хранения и 1 регистр управления

$S = 1$

$A = 4$ — регистры для гашения триггеров

$D = K = L = V = AL = AV = FA = FB = 0$

Массивы общих данных для AK, T и A могут быть записаны АЛГОЛьными процедурами-программами

```
procedure AKO;
 begin
  integer i;
  for i := 1 step 1 until 8 do
    begin AKO[i, 1] := 40;
    AKO[i, 2] := 0; AKO[i, 3] := 0 end
 end;
```

```
procedure TO;
 begin
  integer i;
   begin
    for i := 1 step 1 until 4 do
    begin TO[i, 1] := 40; TO[i, 2] := 0;
    TO[i, 3] := 0 end;
    TO[5, 1] := 8; TO[5, 2] := 1;
    TO [5, 3] := 0
    end;
 end
```

```
procedure AO;
 begin
  integer i;
  for i := 1 step 1 until 4 do
  begin AO[i, 1] := 40; AO[i, 2] := 0;
  AO[i, 3] := 0 end
 end
```

Общие данные для шин

$SO[1, 1] = 40$; $SO[1, 2] = 0$;
$SO[1, 3] = 0$

Если в АЛГОЛьной записи использовать для сокращения *нестандартный опе-*

ратор A ... B и понимать его как запись буквенного кода (адреса) *B* на место (в ячейку) *A*, то массив входных связей активизированной конъюнкции будет:

```
procedure AKS;
 begin
  integer i, j;
   begin
    for i := 1 step 1 until 4 do
    for j := 1 step 1 until 40 do
     begin AKS[i, j, 1] ... T1[5, i];
     AKS[i, j, 2] ... S1[i, j] end;
    for i := 5 step 1 until 8 do
    for j := 1 step 1 until 40 do
     begin AKS [i, j, 1] ... T1 [5, i];
     AKS [i, j, 2] ... T1 [i-4, j] end
   end
 end
```

Массивы выходных связей *AKR* и *AR* образуются:

```
procedure AKR;
 begin
  integer i, j;
  begin
   for i := 1 step 1 until 4 do
   for j := 1 step 1 until 40 do
    begin AKR[i, j, 1] := 1;
    AKR [i, j, 2] := i; AKR [i, j, 3] := j
    end;
   for i := 5 step 1 until 8 do
   for j := 1 step 1 until 40 do
    begin AKR[i, j, 1] := 13;
    AKR [i, j, 2] := 1; AKR[i, j, 3] := j
    end
  end
 end;

procedure AR;
 begin
  integer i, j;
  for i := 1 step 1 until 4 do
  for j := 1 step 1 until 40 do
    begin AR[i, j, 1] := 2;
    AR[i, j, 2] := i; AR[i, j, 3] := j end
 end
```

Пример 2

Последовательный сумматор, изображенный на рис. 2,

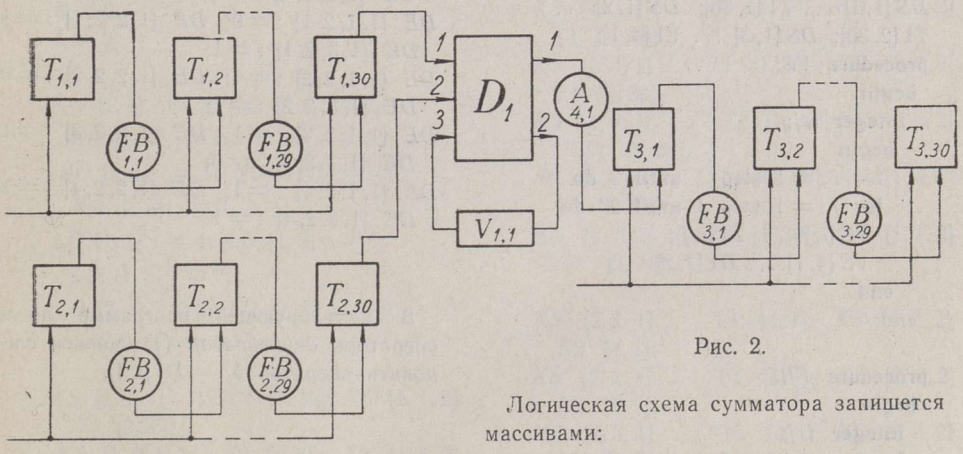

Рис. 2.

может быть записан следующим образом:

$D = 1$ — комбинаторная часть сумматора

$A = 4$

$FB = 3$

$T = 3$ — запоминающая часть сумматора

$V = 4$

Логическая схема сумматора запишется массивами:

```
DO[1, 1] = 3; DO[1, 2] = 2;
DO[1, 3] = 0;
 procedure FBO;
  begin
   integer i;
   for i := 1 step 1 until 3 do
    begin FBO[i, 1] = 29; FBO[i, 2] = 0;
     FBO[i, 3] = 0 end
  end;
```

```
procedure AO;
 begin
  integer i;
  for i := 1 step 1 until 3 do
   begin AO[i, 1] := 30;
   AO[i, 2] := 0; AO[i, 3] := 0 end;
   AO[4, 1] := 1; AO[4, 2] := 0;
   AO[4, 3]: = 0
 end;

procedure TO;
 begin
  integer i;
  for i := 1 step 1 until 3 do
   begin TO[i, 1] := 30; TO[i, 2] := 0;
     TO[i, 3] := 0 end
 end;

procedure VO;
 begin
  integer i;
  begin
   for i := 1 step 1 until 3 do
    begin VO[i, 1] := 29;
      VO[i, 2] := 0; VO[i, 3] := 0 end;
      VO[4, 1] := 1; VO[4, 2] := 0;
      VO[4, 3] := 0
    end
  end;

DS[1, 1] ... T1[1, 30]; DS[1, 2] ...
T1[2, 30]; DS[1, 3] ... V1[4, 1];
procedure VS;
 begin
  integer i, j;
   begin
    for i := 1 step 1 until 3 do
    for j := 1 step 1 until 29 do
     VS[i, j] ... T1[i, j];
     VS[4, 1] ... D1[1, 2]
   end
 end;

procedure FBS;
 begin
  integer i, j;
  for i := 1 step 1 until 3 do
  for j := 1 step 1 until 29 do
    FBS[i, j] ... V1[i, j]
 end;
```

```
procedure AR;
 begin
  integer i, j;
  for i := 1 step 1 until 3 do
  for j := 1 step 1 until 30 do
  begin AR[i, j, 1] := 8;
     AR[i, j, 2] := i; AR[i, j, 3] := j end
     AR[4, 1, 1] := 4;
     AR[4, 1, 2] := 3; AR[4, 1, 3] := 1
 end;


procedure FBR;
 begin
  integer i, j;
  for i := 1 step 1 until 3 do
  for j := 1 step 1 until 29 do
   begin FBR[i, j, 1] := 4;
   FBR[i, j, 2] := i; FBR[i, j, 3] := j + 1
   end
 end;

DR [1, 1] := 4;
DE [1, 1, 1, 1] := 0; DE [1, 2, 1, 1] := 0;
  DE [1, 3, 1, 1] := 1;
DE [1, 1, 1, 2] := 0; DE [1, 2, 1, 2] := 1;
  DE [1, 3, 1, 2] := 0;
DE [1, 1, 1, 3] := 1; DE [1, 2, 1, 3] := 0;
  DE [1, 3, 1, 3] := 0;
DE [1, 1, 1, 4] := 1; DE [1, 2, 1, 4] := 1;
  DE [1, 3, 1, 4] := 1;
DE [1, 1, 2, 1] := 0; DE [1, 2, 2, 1] := 1;
  DE [1, 3, 2, 1] := 1;
DE [1, 1, 2, 2] := 1; DE [1, 2, 2, 2] := 0;
  DE [1, 3, 2, 2] := 1;
DE [1, 1, 2, 3] := 1; DE [1, 2, 2, 3] := 1;
  DE [1, 3, 2, 3] := 0;
DE [1, 1, 2, 4] := 1; DE [1, 2, 2, 4] := 1;
  DE [1, 3, 2, 4] := 1
```

В моделирующей программе после оператора *дешифратор* (1) должен следовать оператор $b := D1[i, 1]$.

Пример 3

В случае замены в примере 2 дешифратора на эквивалентную двухступенчатую схему из трехвходовых конъюнкций и четырехвходовых сборок (рис. 3)*

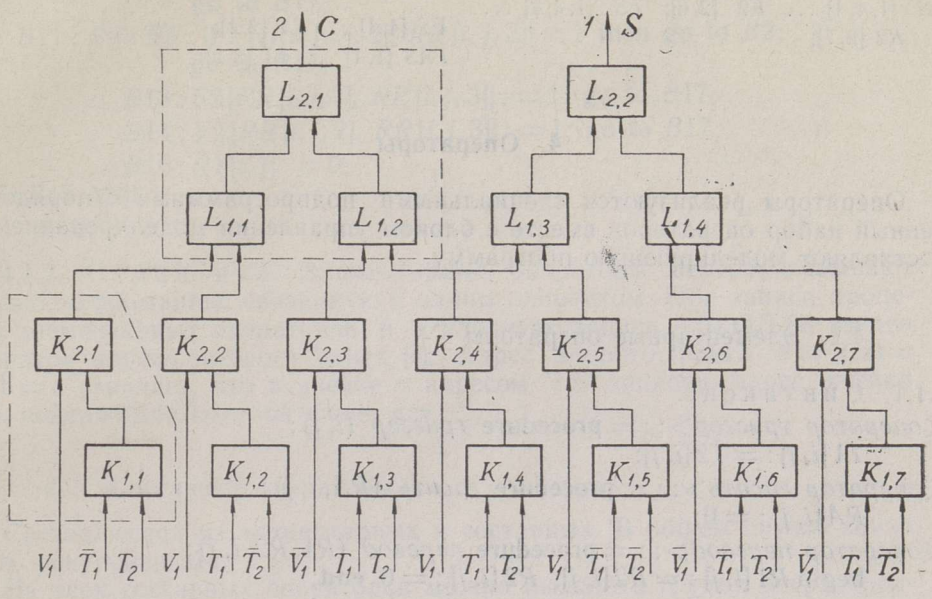

Рис. 3.

массивы для последних запишутся:

$K = 2;$

$L = 2;$

$KO\ [1,1] := 7;\ KO\ [1,2] := 0;$
$KO\ [1,3] := 0;$
$KO\ [2,1] := 7;\ KO\ [2,2] := 0;$
$KO\ [2,3] := 0;$
$LO\ [1,1] := 4;\ LO\ [1,2] := 0;$
$LO\ [1,3] := 0;$
$LO\ [2,1] := 2;\ LO\ [2,2] := 0;$
$LO\ [2,3] := 0;$

$KS\ [1,1,1] \ldots \neg T1\ [1,30];\ KS\ [1,1,2] \ldots$
$T1\ [2,30];$
$KS\ [1,2,1] \ldots T1\ [1,30];\ KS\ [1,2,2] \ldots$
$\neg T1\ [2,30];$
$KS\ [1,3,1] \ldots T1\ [1,30];\ KS\ [1,3,2] \ldots$
$T1\ [2,30];$
$KS\ [1,4,1] \ldots T1\ [1,30];\ KS\ [1,4,2] \ldots$
$T1\ [2,30];$
$KS\ [1,5,1] \ldots \neg T1\ [1,30];\ KS\ [1,5,2] \ldots$
$\neg T1\ [2,30];$
$KS\ [1,6,1] \ldots \neg T1\ [1,30];\ KS\ [1,6,2] \ldots$
$T1\ [2,30];$
$KS\ [1,7,1] \ldots T1\ [1,30];\ KS\ [1,7,2] \ldots$
$\neg T1\ [2,30];$

$KS\ [2,1,1] \ldots V1\ [4,1];\ KS\ [2,1,2] \ldots$
$K2\ [1,1];$
$KS\ [2,2,1] \ldots V1\ [4,1];\ KS\ [2,2,2] \ldots$
$K2\ [1,2];$
$KS\ [2,3,1] \ldots \neg V1\ [4,1];\ KS\ [2,3,2] \ldots$
$K2\ [1,3];$
$KS\ [2,4,1] \ldots V1\ [4,1];\ KS\ [2,4,2] \ldots$
$K2\ [1,4];$
$KS\ [2,5,1] \ldots V1\ [4,1];\ KS\ [2,5,2] \ldots$
$K2\ [1,5];$
$KS\ [2,6,1] \ldots \neg V1\ [4,1];\ KS\ [2,6,2] \ldots$
$K2\ [1,6];$
$KS\ [2,7,1] \ldots \neg V1\ [4,1];\ KS\ [2,7,2] \ldots$
$K2\ [1,7];$

---

* На рис. 3 в обозначениях входов опущен второй индекс, так как это не вызовет неоднозначности. Пунктиром обведены одна из 3-входовых конъюнкций и одна из 4-входовых дизъюнкций.

LS [1, 1, 1] ... *K*2 [2, 1]; *LS* [1, 1, 2] ... *K*2 [2, 2];
LS [1, 2, 1] ... *K*2 [2, 3]; *LS* [1, 2, 2] ... *K*2 [2, 4];
LS [1, 3, 1] ... *K*2 [2, 4]; *LS* [1, 3, 2] ... *K*2 [2, 5];
LS [1, 4, 1] ... *K*2 [2, 6]; *LS* [1, 4, 2] ... *K*2 [2, 7];

LS [2, 1, 1] ... *L*2 [1, 1]; *LS* [2, 1, 2] ... *L*2 [1, 2];
LS [2, 2, 1] ... *L*2 [1, 3]; *LS* [2, 2, 2] ... *L*2 [1, 4];

Соответственно

VS [4, 1] ... *L*1 [2, 1];
FAS [1, 1] ... *L*1 [2, 2];

4. Операторы

Операторы реализуются специальными подпрограммами. Упорядоченный набор операторов вместе с блоком управления моделированием составляют моделирующую программу.

4.1. Элементарные операторы

4.1.1. Синтаксис.

```
<оператор триггер>: : = procedure триггер (i, j);
      T1 [i, j] : = T2 [i, j];
<оператор гасить>: : = procedure гасить (RA, i, j);
      RA [i, j] : = 0;
<оператор перевод>: : = procedure перевод (R1, R2, i, j);
      begin R1 [i, j] : = R2 [i, j]; R2 [i, j] : = 0 end;
<оператор сборка>: : = procedure сборка (i, j);
      L1 [i, j] : = 2LS [i, j, 1] ∨ 2LS [i, j, 2];
<оператор конъюнкция>: : = procedure конъюнкция (i, j);
      K1 [i, j] : = 2KS [i, j, 1] ∧ 2KS [i, j, 2];
<оператор дешифратор>: : = procedure дешифратор (i);
      begin
        for j : = 1 step 1 until DO [i, 1] do
        D3 [j] : = 2DS [i, j];
      B1 : for k : = 1 step 1 until DO [i, 2] do
      B2 : for l : = 1 step 1 until DR [i, k] do
           begin
             integer j;
             D1 [i, k] : = 0;
           B3 : for j : = 1 step 1 until DO [i, 1] do
             if DE [i, j, k, l] = D3 [j] then go to B3
               else go to B2;
             D1 [i, k] : = 1; go to B1
           end
      end;

<оператор задержка>: : = procedure задержка (i, j);
      V2 [i, j] : = 2VS [i, j];
<оператор выход>: : = procedure выход (RR, i, j);
      begin
        switch u1 [RR [i, j, 1]] : = B1, B2, B3, ..., B14, B15, B16;
        begin
            if b = 1 then go to u1 [RR [i, j, 1]]; go to B17;
```

```
        B1 : T2[RR[i, j, 2], RR[i, j, 3]] := 1; go to B17;
        B2 : T2[RR[i, j, 2], RR[i, j, 3]] := 0; go to B17;
B12 : B9 : B6 : B3 : T2[RR[i, j, 2], RR[i, j, 3]] := ¬ T2[RR[i, j, 2],
                     RR[i, j, 3]]; go to B17;
    B10 : B7 : B4 : if T1[RR[i, j, 2], RR[i, j, 3]] = 0 then go to B1;
                    go to B17;
    B11 : B8 : B5 : if T1[RR[i, j, 2], RR[i, j, 3]] = 1 then go to B2;
                    go to B17;
            B13 : S2[RR[i, j, 2], RR[i, j, 3]] := 1; go to B17;
            B14 : V2[RR[i, j, 2], RR[i, j, 3]] := 1; go to B17;
            B16 : R1[i, j] := 0;
B17 : B15 : end
    end;
```

4.1.2. Семантика. Элементарные операторы переобразовывают только информацию, связанную с одним элементом. При записи процедур элементарных операторов в эталонную запись АЛГОЛ-60 введен оператор адреса второго ранга [3]. Адрес второго ранга $^2a = '('a) = 'b = c$ означает, что в ячейке с адресом «a» записан адрес ячейки «b», содержимое которой равно «c».

4.2. Составные операторы

Составляются из элементарных и составных. В общем случае могут быть рекурсивными.

Из всех составных операторов можно выделить группу операторов-регистров, реализующих регистры схемы. В данном предварительном варианте кодирования другие группы не выделены.

4.2.1. Операторы — регистры.

```
<оператор триггерный регистр> : := procedure триггерный регистр (i);
    begin
        integer j;
        for j := 1 step 1 until TO[i, 1] do
        триггер (i, j)
    end;

<оператор регистр сборок> : := procedure регистр сборок (i);
    begin
        integer j;
        own array LS[1 : L, 1 : LO[i, 1], 1 : 2];
        for j := 1 step 1 until LO[i, 1] do
        сборка (i, j)

    end;
<оператор дешифраторы> : := procedure дешифраторы;
    begin
        integer i, j, k, l;
        own array DS[1 : D, 1 : DO[i, 1]];
        own integer array DR[1 : D, 1 : DO[i, 2]];
        own boolean array DE[1 : D, 1 : DO[i, 1], 1 : DO[i, 2], 1 : DR[i, k]];
        for i := 1 step 1 until D do
        дешифратор (i)
    end;
```

```
<оператор регистр задержек>::=procedure регистр задержек (i);
    begin
        integer j;
        own array VS[1:V, 1:VO[i,1]];
        for j:=1 step 1 until VO[i,1] do
        задержка (i,j)
    end;

<оператор регистр конъюнкции>::=procedure регистр конъюнкции (i);
    begin
        integer b, j, m;
        own array KS[1:K, 1:KO[i,1], 1:2];
        switch w1[m]:=Q3, Q4, Q4;
        switch w2[m]:=Q, Q1, Q2;
        comment : метка Q1 является переходом к следующему (т. е.
                  i+1) регистру;
        Q2 : for j:=1 step 1 until KO[i,1] do
             begin
                if j=1 then m:=1; go to w1[m];
        Q3 :    if KO[i,2]=0 then m:=2 else m:=3;
                if KS[i,j,1]=0 then go to w2[m];
                go to w1[m];
        Q4 :    конъюнкция (i,j)
             end
    end;

<оператор активизированный регистр>::=procedure активизирован-
    ный регистр (элемент, R, RO, RS, R1, RR, i);
    begin
        integer j;
        own array RS[1:R, 1:RO[i,1]];
        own integer array RR[1:R, 1:RO[i,1], 1:3];
        for j:=1 step 1 until RO[i,1] do
        активизированный элемент (элемент, R1, RR, i, j)
    end;

<оператор гашение регистра>::=procedure гашение регистра (RA,
    RO, i);
    begin
        integer j;
        for j:=1 step 1 until RO do
        гасить (RA, i, j)
    end;

<оператор перевод регистра>::=procedure перевод регистра (RO,
    R1, R2, i);
    begin
        integer j;
        for j:=1 step 1 until RO do
        перевод (R1, R2, i, j)
    end;
```

4.2.2. Прочие операторы.

```
<оператор активизированный элемент>::= procedure активизированный элемент (элемент, R1, RR, i, j);
    begin
        integer b;
        begin
            элемент;
            b:= R1[i, j];
            выход (RR, i, j)
        end
    end;

<оператор схема>::= procedure схема (регистр, R);
    begin
        integer i, R;
    Q1: for i:= 1 step 1 until R do
        регистр (i)
    end;

<оператор гашение>::= procedure гашение (RA, R, RO);
    begin
        integer R, RO, i;
        for i:= 1 step 1 until R do
        гашение регистра (RA, RO, i)
    end;

<оператор время>::= procedure время (R, RO, R1, R2);
    begin
        integer R, RO, i;
        for i:= 1 step 1 until R do
        перевод регистра (RO, R1, R2, i)
    end;

<оператор счетчик>::= procedure счетчик (i);
        Z[i]:= Z[i] + 1;
<оператор гашение счетчиков>::= procedure гашение счетчиков (ZO);
    begin
        integer ZO, i;
        for i:= 1 step 1 until ZO do
        Z[i]:=0
    end;

<оператор начало>::= procedure начало;
    begin
        гашение счетчиков (ZO);
        гашение (D1[i, j], D, DO[i, 1]);
        гашение (K1[i, j], K, KO[i, 1]);
        гашение (L1[i, j], L, LO[i, 1]);
        гашение (S1[i, j], S, SO[i, 1]);
        гашение (T2[i, j], T, TO[i, 1]);
```

```
        гашение (V2[i, j], V, VO[i, 1]);
        гашение (AK1[i, j], AK, AKO[i, 1]);
        гашение (AL1[i, j], AL, ALO[i, 1]);
        гашение (AV2[i, j], AV, AVO[i, 1]);
        гашение (FA1[i, j], FA, FAO[i, 1]);
        гашение (FB1[i, j], FB, FBO[i, 1])
    end;
```

Пример 4

Для моделирования схемы на рис. 1 программа может иметь вид:

```
begin
  начало;
  ввод;
x1 : счетчик (1);
  схема (активизированный регистр
  конъюнкций, AK);
  схема (активизированный регистр
      конъюнкций, AK);
  if z = N then go to x3 else go to x2;
x2 : схема (регистр триггеров, T);
  go to 1;
x3 : вывод
end;
```

Управление моделированием легче всего осуществить в месте, где поставлена проверка счетчика, а также при возвращении к метке $x1$.

В программе использованы не описанные в статье операторы *ввод* и *вывод*. Их содержание зависит от примененного конкретного АЛГОЛьного транслятора.

Использование дважды подряд процедуры *схема (активизированный регистр конъюнкций, AK)* вызвано наличием в схеме шин.

ЛИТЕРАТУРА

1. Глушков В. М., Синтез цифровых автоматов, Физматиздат, М.-Л., 1962.
2. Бэкус Дж. В. и др., Ж. вычислит. матем. и математич. физики, 1, № 2, 308—342 (1961).
3. Глушков В. М., Ющенко Е. Л., Вычислительная машина «Киев», ГИТЛ УССР, Киев, 1962.
4. Боттенбрух Г., Структура АЛГОЛ-60 и его использование, Изд. ИЛ, М., 1963.

Институт кибернетики
Академии наук Эстонской ССР

Поступила в редакцию
31 I 1964

KODEERING NUMBRILISTE SKEEMIDE KIRJELDAMISEKS JA MODELLEERIMISEKS

H. Salum

Resümee

Esitatakse kodeeringu algvariant. Modelleeritav skeem peab olema esitatud loogilise struktuurskeemina registrite ja elementide nivool. Kodeerimise kirjeldamisel kasutatakse ALGOL-60 [2,4] sümboleid. Peale selle kasutatakse teise järgu aadressi mõistet [3]. Tuuakse näiteid konkreetsete skeemide kirjeldamise kohta.

Eesti NSV Teaduste Akadeemia
Küberneetika Instituut

Saabus toimetusse
31. I 1964

A CODING FOR DESCRIBING AND SIMULATING DIGITAL SCHEMES

H. Salum

Summary

A preliminary variant of coding is given. A scheme to be described and simulated has to be presented as a composition [1] of logical elements and registers. The ALGOL-60 symbols are used to describe the coding. An operator of the second-range-address [3] is used to simplify the descriptions. Some examples of the applications are presented.

Academy of Sciences of the Estonian S.S.R.,
Institute of Cybernetics

Received
Jan. 31st, 1964