УДК 621.382

П.Н. Бибило

Нахождение теста для режима максимального энергопотребления комбинационной схемы

Предложены формализация задачи и алгоритм нахождения тестовых векторов, обеспечивающих режим максимального энергопотребления комбинационной логической схемы, синтезированной в базисе проектирования заказной КМОП СБИС.

A formalization of the problem and an algorithm of finding the test vectors that allow to find the mode of the maximal power consumption in a combinational logic circuit synthesized in the custom CMOS VLSI design basis are suggested.

Запропоновано формалізацію задачі та алгоритм знаходження тестових векторів, які забезпечують режим максимального енергоспоживання комбінаційної логічної схеми, синтезованої у базисі проектування заказної КМОП НВІС.

Введение. Оценка энергопотребления – важная задача на стадии синтеза проекта цифровой сверхбольшой интегральной схемы (СБИС). В работах [1, 2] показано, что основная доля энергопотребления КМОП-схем приходится на переключения транзисторов. Для комбинационных логических блоков, входящих в состав заказной СБИС и представляющих собой логические схемы из КМОП-элементов библиотеки проектирования СБИС, в данной статье предложено оценивать энергопотребление проекта логической схемы на основе подсчета числа переключений транзисторов, из которых состоят логические элементы схемы. Каждому такту функционирования логической схемы соответствует упорядоченная пара <*предыдущий набор i*, *текущий набор j*> входных сигналов, а каждой такой упорядоченной паре соответствует число $S_{i,j}$ переключающихся транзисторов в схеме. Поскольку подсчет числа переключений транзисторов может быть выполнен достаточно быстро с помощью логического моделирования, оперирующего с *VHDL*-моделями [3, 4] логических элементов и схемы в целом, то для каждого такта моделирования можно получить соответствующее число переключений транзисторов. Для схем с ограниченным числом $n \le 15$ входов можно провести логическое моделирование на всех упорядоченных парах входных наборов и выбрать *последовательность из k наборов* (тест *T*), характеризующихся наибольшим числом переключений транзисторов, и, следовательно, наибольшим энергопотреблением. Заметим, что число всех пар входных наборов для комбинационной схемы, имеющей *n* входов, равно $2^n(2^n - 1)$. Найдя тест *T*, можно использовать его для трудоемкого схемотехнического моделирования, которое и позволяет сделать более точную оценку энергопотребления.

В данной статье предложен способ составления *VHDL*-моделей логических элементов (и схемы в целом) для подсчета числа переключений транзисторов логической схемы. Предлагается также формализация задачи формирования теста *T*, состоящего из заданного числа *k* входных наборов и обеспечивающего режим максимального потребления тока схемой. Искомый тест *T* строится путем анализа всех пар входных наборов и соответствующих им чисел переключений транзисторов в предположении, что максимальному числу переключений (в заданном такте) соответствует максимальное потребление тока. Если же паре входных наборов поставлено в соответствие значение потребляемого тока, то найденный тест обеспечит не приближенное, а реальное (с точки зрения схемотехнического моделирования) потребление тока. Величина значения потребляемого тока определяет минимальную ширину проводников в сетях питания и заземления СБИС, «правильная» ширина таких проводников важна для предотвращения эффектов электромиграции, приводящих к разрыву проводников и сбоям функционирования СБИС [2, с. 558].

Подсчет числа переключений транзисторов

Рассмотрим логическую схему *circ* (рис. 1), на примере которой будем подсчитывать число переключений транзисторов. В схеме использу-

ются элементы следующих типов: *A*2, *A*3 – двухвходовый и трехвходовый элемент И соответственно; *O*2 – двухвходовый элемент ИЛИ; *N* – инвертор; *XOR*2 – «сумма по модулю 2». Суть предлагаемого подхода для подсчета числа переключающихся транзисторов заключается в преобразовании *VHDL*-моделей элементов и преобразовании схемы (см. рис. 1) в схему, изображенную на рис. 2. При составлении *VHDL*-моделей логических элементов полагается, что все элементы имеют одинаковую задержку, в данном примере выбранную равной 1 *ns*. Каждый элемент схемы (см. рис. 2), дополняется средством (*VHDL*-процессом) для подсчета числа переключившихся транзисторов в этом элементе. Для этого *VHDL*-модель элемента снабжается дополнительным выходом *z*, значение которого задает число переключившихся транзисторов в данном сеансе моделирования. При этом учитывается схемотехника (иерархия на уровне *VHDL*-описаний) элементов, например, элемент *A*2 представляет собой каскадное соединение логического элемента *NA*, реализующего функцию И–НЕ, и инвертора *N*. Пример модели иерархически описанного элемента *A*2 и базового элемента *NA* приведен в лист. 1.

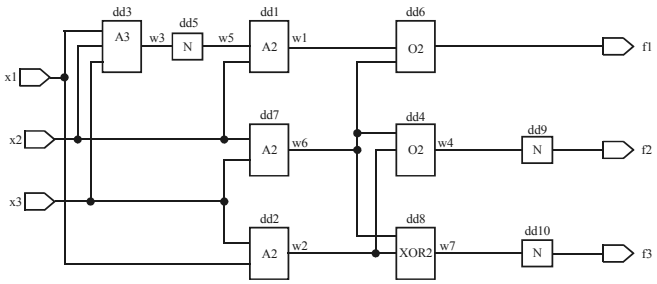

Рис. 1. Логическая схема *circ*

Л и с т и н г 1. *VHDL*-модели логических элементов *A*2, *NA*

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity A2 is
port (A:IN std_logic;
    B:IN std_logic;
    Y:OUT std_logic;
Z : out integer);
end;
architecture str of A2 is
component    NA
  port (A:IN std_logic;
      B:IN std_logic;
      Y:OUT std_logic;
  Z : out integer);
  end component;
  component    N
  port (A:IN std_logic;
      Y:OUT std_logic;
  Z : out integer);
  end component;
  signal W : std_logic;
  signal k1,k2 : integer;
  type MAS1 is array (1 to 2) of integer;
  signal V : MAS1;
  function sum(DATA: in MAS1) return inte-
ger is
  variable S : integer:= 0;
  begin
   for I in DATA'range loop
   S:= DATA(I) + S;
     end loop;
   return S;
   end  sum;
  begin
  p0: NA port map (A,B,W,k1);
  p1: N port map (W,Y,k2);
  V <= (k1,k2);
  p2 : process (V)
   begin
   Z <= sum(V);
  end process;
  end;

  library IEEE;
  use IEEE.STD_LOGIC_1164.all;
  use IEEE.NUMERIC_STD.all;
  entity NA is
  port (A:IN std_logic;
      B:IN std_logic;
      Y:OUT std_logic;
  Z : out integer);
  end;
  architecture BEHAVIOR of NA is
  begin
  Y <= not (A and B) after 1ns;
  p1:  process (A, B)
  variable z1 : integer :=0;
  begin
  if ((A xor A'delayed (1 ns)) = '1')
then Z1:= Z1+2;
  end if;
  if ((B xor B'delayed (1 ns)) = '1')
then Z1:= Z1+2;
  end if;
  Z <= z1;
  end process;
  end;
```

В *VHDL*-модели элемента *NA* полагается, что изменение значения (0 на 1, 1 на 0) входного сигнала влечет переключение двух транзисторов, а с помощью атрибута *A'delayed* (1 *ns*) вычисляется значение, которое имел сигнал *A* одну наносекунду назад относительно текущего времени моделирования. Заметим, что подсчет числа переключений осуществляется с помощью функции *sum*, написанной для логического элемента с многими входами, для двухвходовых элементов можно обойтись более простыми средствами.

VHDL-модель схемы в целом дополняется процессом (см. рис. 2), осуществляющим суммирование переключений транзисторов по всем элементам схемы. Такое суммирование выполняет функция *sum_percl*, по сути та же функция *sum*. Пример описания схемы *circ* для подсчета переключений триггеров приведен в лист. 2. Жирным шрифтом выделены операторы, ответственные за подсчет перекючений триггеров. В этом же листинге приводится пакет *perecl*, содержащий декларации компонент и функцию *sum_percl*.

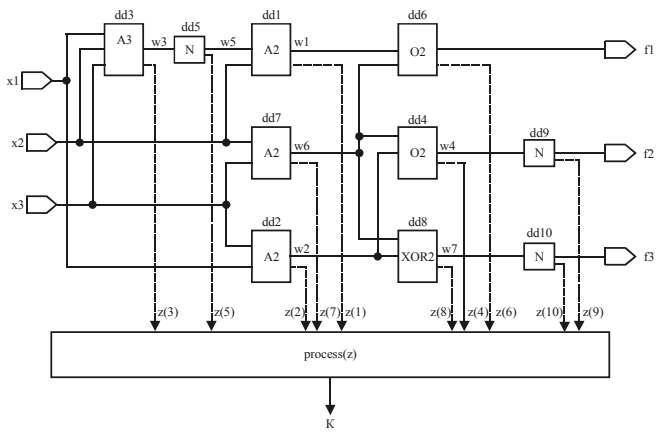

Рис. 2. Преобразованная логическая схема *circ*

Л и с т и н г 2. *VHDL*-описание схемы ***circ*** для подсчета числа переключений транзисторов

```
library ieee;
use ieee.std_logic_1164.all;
use work.perecl.all;

entity circ is
port(x1, x2, x3 : in std_logic;
     F1, F2, F3 :out std_logic;
                 K : out integer);
end;

architecture circ_arch of circ is
signal W : std_logic_vector (1 to 8);
signal Z : MAS;

begin
dd1 : A2 port map (w(5),x2,w(1), Z(1));
dd2 : A2 port map (x3,x1,w(2), Z(2));
dd3 : A3 port map (x1,x3,x2,w(3), Z(3));
dd4 : O2 port map (w(7),w(2),w(4), Z(4));
dd5 : N port map (w(3), w(5), Z(5));
dd6 : O2 port map (w(1),w(7),F1, Z(6));
dd7 : A2 port map (x2,x3,w(7), Z(7));
dd8 : XOR2 port map (w(7),w(2),w(8),
Z(8));
dd9 : N port map (w(4), F2, Z(9));
dd10 : N port map (w(8), F3, Z(10));
 p1 : process (Z)
 begin
 K <= sum_percl(z);
 end process;
end;

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

package perecl is
constant S : integer := 10;
type MAS is array (1 to S) of integer;
component    A2 is
port (A:IN std_logic;
    B:IN std_logic;
    Y:OUT std_logic;
Z : out integer);
end component;
.........
-- декларации компонент A3,O2,N,XOR2

function sum_percl (DATA: in MAS)
return integer;

package body perecl is
function sum_percl (DATA: in MAS)
return integer is

variable S : integer:= 0;
begin
 for I in DATA'range loop
 S:= DATA(I) + S;
   end loop;
 return S;
 end  sum_percl;
  end perecl ;
```

Результат логического моделирования схемы на упорядоченной последовательности 57 на-

боров входных сигналов представлен в таблице, где заданы числа $S_{i,j}$ переключающихся транзисторов для каждой пары <*i*, *j*> сменяемых наборов значений входных сигналов.

Номер пары	Входные наборы	Вес $S_{i,j}$	Номер пары	Входные наборы	Вес $S_{i,j}$	Номер пары	Входные наборы	Вес $S_{i,j}$
	7			1			3	
1		26	21		22	41		30
	5			6			1	
2		34	22		16	42		30
	7			0			3	
3		**58**	23		16	43		36
	4			6			0	
4		**50**	24		36	44		36
	7			5			3	
5		34	25		32	45		24
	3			3			2	
6		26	26		32	46		12
	7			5			0	
7		**62**	27		**40**	47		12
	2			2			2	
8		**42**	28		**40**	48		18
	7			5			1	
9		**56**	29		22	49		6
	1			1			0	
10		**48**	30		22	50		6
	7			5			1	
11		**62**	31		28	51		18
	0			0			2	
12		**54**	32		28	52		24
	7			5			3	
13		**58**	33		24	53		**40**
	6			4			4	
14		12	34		16	54		**24**
	4			2			5	
15		12	35		16	55		36
	6			4			6	
16		28	36		10	56		38
	3			1			7	
17		28	37		10			
	6			4				
18		4	38		4			
	2			0				
19		4	39		4			
	6			4				
20		22	40		**40**			
	1			3				

Формализация задачи нахождения теста

Пусть задана комбинационная схема *R*, состоящая из логических КМОП-элементов и имеющая *n* входов x_1, x_2, x_n. Булево пространство $V^{\underline{x}}$ над переменными вектора $\underline{x} = (x_1, x_2, \ldots, x_n)$ содержит 2^n двоичных наборов. Каждому двоичному набору $\underline{x}_i^* \in V^{\underline{x}}$ поставим в соответствие число *i*, равное десятичному эквиваленту этого набора. Если при моделировании схемы *R* входной набор *i* сменяется входным набором *j*, то упорядоченной паре <*i*, *j*> соответствует число $S_{i,j}$ переключений транзисторов (вес). Сформулируем формальную постановку задачи нахождения теста *T*, обеспечивающего максимальное суммарное число переключений транзисторов.

Пусть задано множество *V* чисел: $V = \{0, 1, 2, \ldots, 2^n - 1\}$. Рассмотрим множество *L* всех $2^n(2^n - 1)$ упорядоченных пар <*i*, *j*>, составленных из элементов множества *V*. Каждой паре <*i*,*j*> соответствует неотрицательное целое число $S_{i,j}$ – вес пары <*i*,*j*>. Каждой упорядоченной последовательности *P*

$$P = < i_1, i_2, i_3, i_4, \ldots i_{k-2}, i_{k-1}, i_k > \qquad (1)$$

элементов (не обязательно различных) множества *V* соответствует множество

$$< i_1, i_2 >, < i_2, i_3 >, < i_3, i_4 >, \ldots., < i_{k-2}, i_{k-1} >, < i_{k-1}, i_k > \qquad (2)$$

упорядоченных пар, составленных из *соседних* элементов последовательности (1). *Правильной* *k*-последовательностью назовем такую упорядоченную последовательность (1), что все упорядоченные пары вида (2) *различны*.

Задача нахождения теста *T* имеет следующую формальную постановку.

Задача. Для заданного числа *k* требуется составить из элементов множества *L* правильную *k*-последовательность *P* с максимальной суммой весов

$$S = \sum_{q=2}^{k-1} (S_{i_{q-1}, i_q} + S_{i_q, i_{q+1}}). \qquad (3)$$

Если каждому элементу множества *L* поставить в соответствие вершину полного ориентированного графа *G*, то задача может быть переформулирована в графовой постановке: в полном ориентированном графе *G*, дуги которого взвешены неотрицательными целыми числами, требуется найти простую цепь *M*, состоящую из *k* дуг и имеющую максимальную сумму *S* весов входящих в нее (т.е. в цепь) дуг.

Данная задача и алгоритмы ее решения хорошо известны в теории графов [5].

Модификации этой задачи появляются тогда, когда в качестве весов дуг могут быть как неотрицательные, так отрицательные числа. Другой особенностью может быть условие $S_{i,j} = S_{j,i}$ для весов дуг, пример такого графа дан на рис. 3. Граф G (см. рис. 3) получен при моделировании элемента, реализующего логическую операцию «сумма по модулю 2». Одно из решений задачи для $k = 3$ выделено на графе G штриховыми дугами.

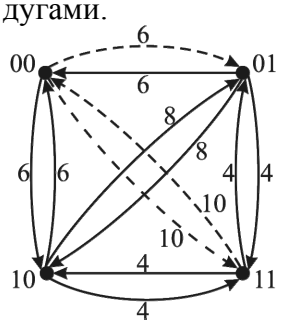

Рис. 3. Граф G

Алгоритм нахождения теста

Поскольку в практических ситуациях число 2^n вершин графа G может быть огромным, то о точном решении задачи не может быть и речи, поэтому предлагается эвристический алгоритм решения задачи нахождения теста T, ориентированный на практическую размерность, когда $n \le 15$. Элементы булева пространства будем представлять числами – десятичными эквивалентами, а алгоритм иллюстрировать на примере последовательности наборов, заданных в табл. 1. Пусть для данного примера $k = 16$, тогда число искомых пар входных наборов равно 15. Алгоритм состоит из следующих шагов.

Ш а г 1. Расположение без повторений всех пар множества L в линейном порядке [6], соответствующую последовательность расположения пар множества L обозначим через Z.

В рассматриваемом примере логической схемы (см. рис. 1) исходным является булево пространство от трех переменных, а десятичные представления двоичных наборов образуют множество {0, 1, 2, 3, 4, 5, 6, 7}. Расположить без повторений все пары, т.е. получить упорядоченную последовательность Z комбинаций входных сигналов можно согласно «правилу треугольника»:

7, 5, 7, 4, 7, 3, 7, 2, 7, 1, 7, 0, 7
6, 4, 6, 3, 6, 2, 6, 1, 6, 0, 6,
5, 3, 5, 2, 5, 1, 5, 0, 5,
4, 2, 4, 1, 4, 0, 4,
3, 1, 3, 0, 3,
2, 0, 2,
1
0, 1, 2, 3, 4, 5, 6, 7

Расположив строки «треугольника» в линейном порядке, получим последовательность Z, которая для рассматриваемого примера задана в столбце «Входные наборы» табл. 1. Аналогичным образом, т.е. без повторений пар, можно перечислить все пары элементов булева пространства и большей, чем три, размерности.

Ш а г 2. Нахождение пар с максимальным весом.

Находим в Z последовательно одну за другой пары $< i_q, j_q >$ с максимальным весом S_{i_q, j_q}. Если в последовательности Z очередная пара $S_{i_{q+1}, j_{q+1}}$ располагается рядом с одной уже из найденных пар, то ищется дополнительно еще одна пара.

В примере максимальный вес 62 имеет пара <7,2> = $< i_1, j_1 >$ с номером 7. Затем находится пара $< i_2, j_2 >$=<7,0> с номером 11, имеющая тот же вес 62, и пары $< i_3, j_3 >$ = <7,4> (номер 3), $< i_4, j_4 >$ = <7,6> (номер 13), имеющие вес 58. Следующей будет пара $< i_5, j_5 >$=<7,1> с номером 9 и весом 56. Среди оставшихся пар максимальный вес 50 имеет пара $< i_6, j_6 >$=<4,7> с номером 4. Так как эта пара – соседняя ($j_6 = i_3$) с уже найденной парой $< i_3, j_3 >$=<7,4>, то для того, чтобы искомая последовательность содержала 15 пар, требуется найти еще одну пару и т.д.

Последовательно находя пары с максимальным весом, построим цепочки c_1, c_2, c_3, c_4 пар, содержащие 16 элементов: цепочке 1 соответствует последовательность <7,4,7> из трех эле-

ментов; цепочке 2 – последовательность <7,2, 7,1,7,0,7,6> из восьми элементов; цепочке 3 – последовательность <5,2,5> из трех элементов; цепочке 4 – последовательность <3,4> из двух элементов. В данном примере только одна цепочка 4 выродилась до пары. Однако может случиться и так, что каждая из цепочек может состоять из одной пары.

Ш а г 3. Соединение найденных цепочек в одну последовательность.

Для этого строится граф H (рис. 4), отражающий варианты соединения (конкатенации) цепочек. Легко видеть, что при соединении цепочек может потребоваться «добирать» пары, так как последний элемент цепочки, к которой будет добавляться следующая, может совпадать с первым элементом подсоединяемой цепочки, поэтому в последовательности Z потребуется искать еще пары с максимальным весом, которые нужно будет включить в искомую последовательность.

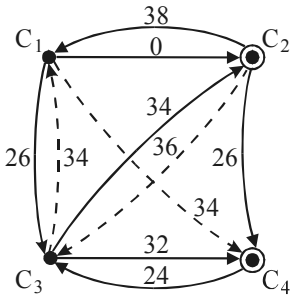

Рис. 4. Граф H

Вернемся к примеру. Построим граф H, каждой из вершин которого соответствует найденная цепочка, а ориентированная дуга $<c_i, c_j>$ взвешена весом S пары, образованной последним элементом цепочки c_i и первым элементом цепочки c_j. Например, дуга $c_2 \to c_3$, исходящая из вершины c_2 и заходящая в вершину c_3, имеет вес 36, так как этой дуге графа H соответствует пара <6,5> с номером 24. Некоторые дуги имеют вес ноль, например, такой дугой является дуга $c_1 \to c_2$. Этой дуге соответствует пара <7,7> – последним элементом цепочки c_1 есть элемент 7, а первым элементом цепочки c_2 есть также элемент 7. Граф H не является полным, некоторые дуги могут отсутствовать, например, между вершинами c_4, c_2 дуга $c_4 \to c_2$ отсутствует – этой дуге соответствует пара <4,7>, которая включена в цепочку c_1. Для рассматриваемого примера и найденных цепочек c_1, c_2, c_3, c_4 граф H изображен на рис. 4.

Задача нахождения последовательности соединения цепочек на шаге 4 алгоритма сводится к нахождению последовательности обхода всех вершин графа H, причем сумма весов проходимых дуг должна быть максимальна, т.е. на этапе 4 требуется решить задачу нахождения в ориентированном графе H гамильтонова цикла с максимальным суммарным весом дуг. Это также известная задача в теории графов [5].

Рассмотрим маршрут $<c_1, c_2, c_3, c_4>$, приводящий к следующей конкатенации <7,4,**7**>, <**7**,2,7,1,7,0,7,6>, <5,2,5>, <3,4> цепочек и сумме S весов

$$S = 58+50+\mathbf{0}+62+42+56+48+62+ \\ +54+58+\mathbf{36}+40+40+\mathbf{32}+40 = 678.$$

Последний элемент 7 цепочки <7,4,**7**> совпадает с первым элементом 7 цепочки <**7**,2,7, 1,7,0,7,6>, поэтому в последовательности <7,4, 7,2,7,1,7,0,7,6,5,2,5,3,4> имеется только 15 элементов. Следовательно, требуется добавить одну пару, например, ту, которая имеет максимальный вес и первый элемент которой равен четырем. Такой есть пара <4,5> (номер 54) с весом 24. Для последовательности <7,4,7,2,7,1, 7,0,7,6,5,2,5,3,4,5> вес

$$S = 58+50+62+42+56+48+62+54+ \\ +58+\mathbf{36}+40+40+\mathbf{32}+40+\mathbf{24}=702.$$

Рассмотрим маршрут $<c_1, c_2, c_3, c_4>$ обхода всех вершин: этому маршруту соответствует конкатенация цепочек <7,2,7,1,7,0,7,6>, <5,2,5>, <7,4,7>, <3,4>, и последовательность <7,2,7,1, 7,0,7,6,5,2,5,7,4,7,3,4>, а этой последовательности соответствует сумма

$$S = 62+42+56+48+62+54+58+\mathbf{36}+ \\ +40+40+\mathbf{34}+58+50+\mathbf{34}+40=714$$

весов, вычисляемая по формуле (3). Для данного маршрута добавлять пары не понадобилось, так как дуга с нулевым весом в графе H не была пройдена.

Последний маршрут имеет больший вес и является лучшим. На шаге 4 после соединения цепочек, если это необходимо, последовательно до-

бавляются пары согласно «жадной» эвристики – каждая из добавляемых пар должна обеспечить максимальный вклад в суммарный вес.

Результаты схемотехнического моделирования

Чтобы проверить результаты логического моделирования и предложенный алгоритм формирования теста, было составлено схемотехническое (*Spice*-описание) логической схемы (см. рис. 1) и проведено моделирование на той же последовательности из 57 наборов входных сигналов, для которой проводилось логическое *VHDL*-моделирование. Результат схемотехнического моделирования схемы `circ` в системе *Accusim* (разработка фирмы *Mentor Graphics*) на той же последовательности 57 входных наборов представлен на рис. 5, среднее значение потребляемого тока равно –0,092902 ма (микроампер). Пары наборов, вошедшие в цепочки c_1, c_2, c_3, c_4, помечены на рис. 5 символом «*».

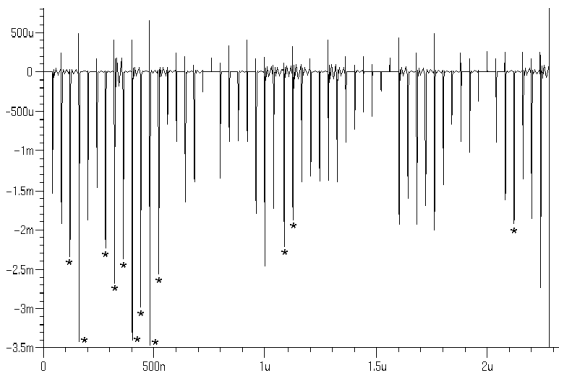

Рис. 5. Результат аналогового моделирования на наборах последовательности *Z*

Если же провести схемотехническое моделирование на 16 наборах найденного теста *T* (рис. 6), то среднее значение потребляемого тока равно –0,150054 ма, т.е. значительно выше. Таким образом, тест *T*, найденный на основе предложенного алгоритма, позволяет с достаточной для практики точностью находить режим максимального энергопотребления, что подтверждается результатами схемотехнического моделирования.

Если внимательно проанализировать рис. 5, то можно заметить, что в некоторых тактах срабатывания схемы график функции потребляемого тока заходит в область положительных значений, при формализации задачи это будет выражено отрицательными значениями весов $S_{i,l}$, если в качестве таковых взять не числа переключений транзисторов, а значения тока, полученные при схемотехническом моделировании.

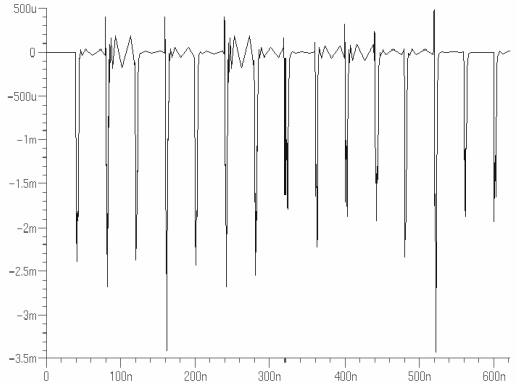

Рис. 6. Результат аналогового моделирования на наборах теста *T*

Заключение. Предложенный алгоритм на основе подсчета числа переключений транзисторов позволяет быстро находить режим максимального энергопотребления комбинационной логической схемы, состоящей из КМОП-элементов. Данный способ пригоден для задач практической размерности и может быть использован для оценки вариантов схемной реализации структур СБИС на этапе синтеза проекта СБИС.

1. *Estimation* of Average Switching Activity in Combinational and Sequential Circuits / A. Ghosh, S. Devadas, K. Keutzer et al. // Proc. 29th ACM/IEEE Design Automation Conf. – 1992. – P. 253–259.
2. *Рабаи Ж.М.*, *Чандракасан А.*, *Николич Б.* Цифровые интегральные схемы. – М.: ООО «И.Д. Вильямс», 2007. – 912 с.
3. *Суворова Е.А.*, *Шейнин Ю.Е.* Проектирование цифровых систем на *VHDL*. – СПб.: БХВ–Петербург, 2003. – 576 с.
4. *Сергиенко А.М. VHDL* для проектирования вычислительных устройств. – К.: ЧП «Корнейчук», ООО «ТИД “ДС”», 2003. – 208 с.
5. *Харари Ф.* Теория графов. – М.: Мир, 1973. – 300 с.
6. *Закревский А.Д.* Минимизация перебора ориентированных пар. Танаевские чтения // Докл. Четвертой Междунар. науч. конф. (29 марта 2010 года, Минск). – Минск: ОИПИ НАН Беларуси, 2010. – С. 89–97.

Поступила 12.04.2010

E-mail: *bibilo@newman.bas-net.by*