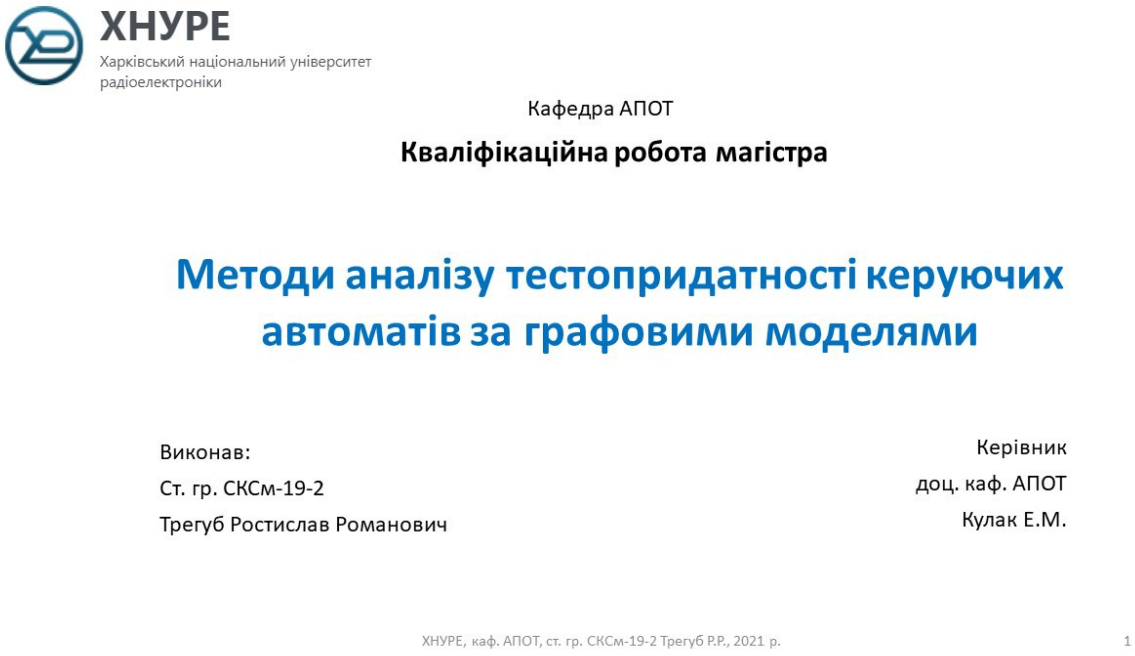
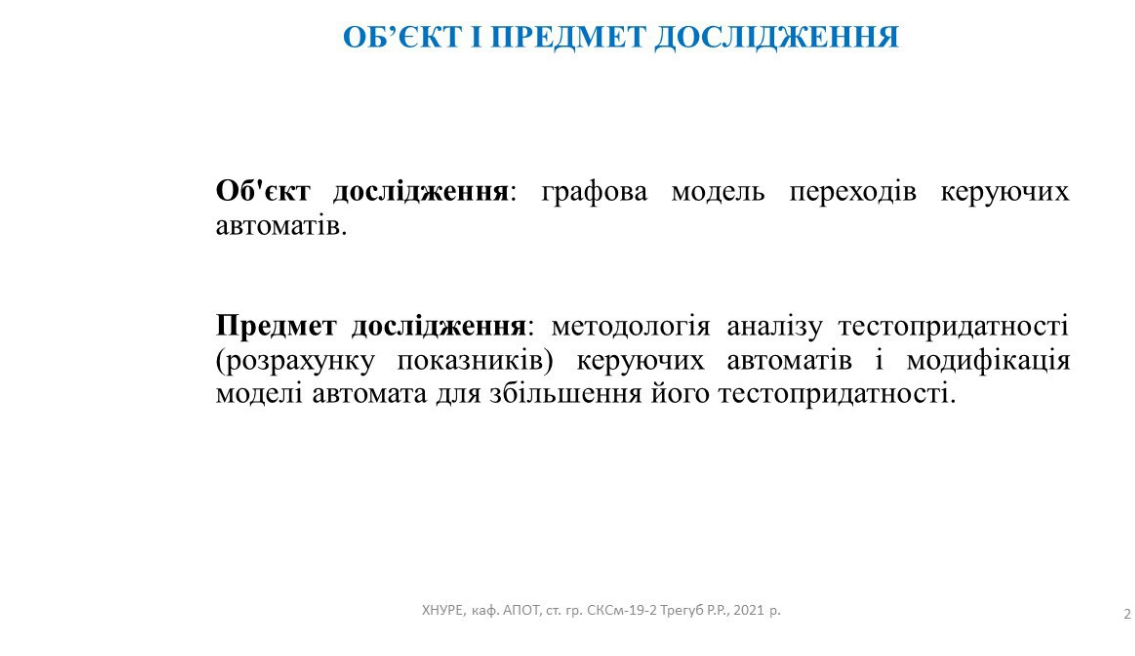
молодіжного форуму «Радіоелектроніка та молодь у 21 столітті» Зб. матеріалів форуму Т.5 – Харків.: ХНУРЕ – 20-22 квітня 2021., с. 31-32.

ДОДАТОК А

Графічний матеріал атестаційної роботи





МЕТА ДОСЛІДЖЕННЯ І ПОСТАНОВКА ЗАДАЧІ

Мета дослідження : Метою дослідження є розробка методу аналізу тестопридатності керуючих автоматів при побудові функціонального тесту, і способу модифікації автомата для підвищення його тестопридатності.

Для досягнення цієї мети необхідно вирішити наступні задачі :

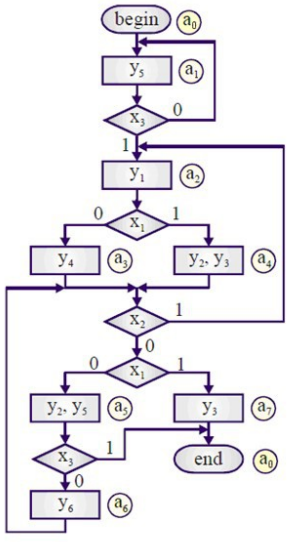
– розробити метод аналізу тестопридатності цифрових автоматів, представлених у вигляді орієнтованого графа для спрощення завдання генерації тестів;

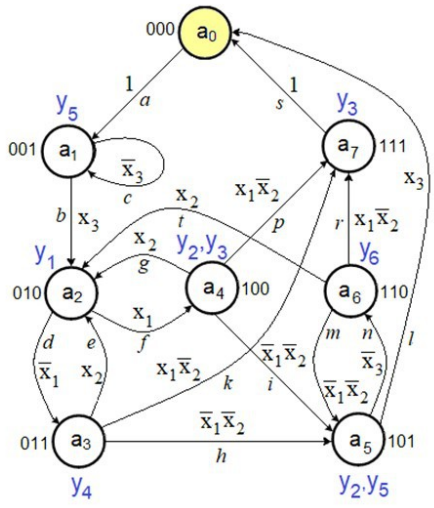
– удосконалити алгоритм побудови дерева рішень для обходу станів ГП автомата при наявності циклів для організації ДЕ;

– розробити методи підвищення тестопридатності керуючих автоматів;

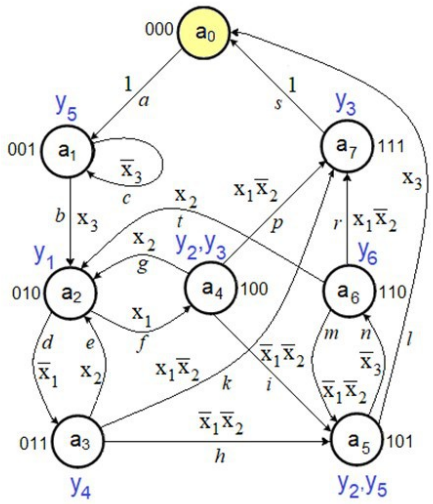
– реалізувати VHDL - моделі для проведення порівняльного аналізу розроблених методів.

ГСА І ГРАФ ПЕРЕХОДІВ АВТОМАТА МУРА





ПОБУДОВА МАТРИЦІ СУМІЖНОСТІ АВТОМАТУ



	a0	a1	a2	a3	a4	a5	a6	a7
a0		a						
a1		c	b					
a2				d	f			
a3			e			h		k
a4			g			i		p
a5	l						n	
a6			t			m		r
a7	s							

АЛГОРИТМ ПОБУДОВИ ДЕРЕВА ОБХОДУ ГРАФА ПЕРЕХОДІВ

Для реалізації стратегії обходу всіх дуг графа (в режимі «ручної» побудови тестів) будується бінарне дерево рішень (дерево обходу графа) за наступним алгоритмом.

1. Побудова дерева обходу графа починається з початкової вершини a_0. Аналізується перший рядок матриці суміжності і складається список вершин-наступників.

2. Зі списку наступників обирається перша вершина і включається в дерево. Відповідна комірка в матриці (дуга) позначається.

3. Для рядка, що відповідає цьому номеру, складається список х наступників. При подальшому аналізу списку наступників перевага віддається непоміченим коміркам-наступникам.

4. Побудова маршруту в графі завершується у наступних випадках :

– чергова вершина-наступник є початковою (термінальною) вершиною а0 (завершений цикл роботи автомата);
– вихідна дуга є петлею (елемент головної діагоналі матриці суміжності);
– у маршруті з'являється вершина, яка вже присутня у цьому маршруті (цикл у графі переходів, фіксується початкова вершина циклу);

5. П. п. 2, 3 та 4 повторюються до тих пір, поки всі маршрути не дійдуть до термінальної (кінцевої) вершини.

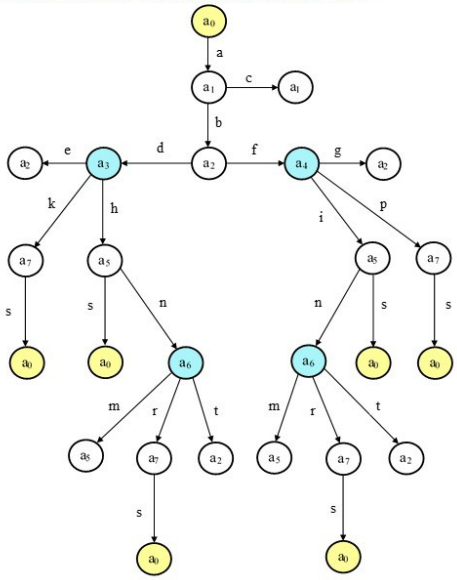
6. Петлі та маршрути з циклами заносяться до окремого стеку. Петлі можуть вставлятися у будь-які маршрути, де є відповідні вершини.

7. Для маршрутів зі стеку циклів добудовуються маршрути повернення в початкову вершину шляхом використання фрагментів маршрутів з цієї вершини до термінальної вершини (такий маршрут завжди буде виходячи зі стандартних умов досяжності графа).

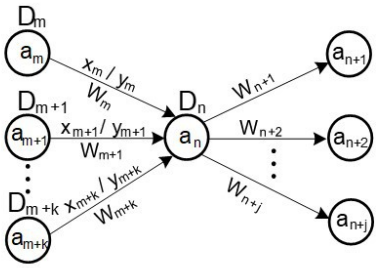
МАТРИЦЯ СУМІЖНОСТІ І ДЕРЕВО РІШЕНЬ АВТОМАТА МУРА

Приклад побудови дерева по матриці суміжності

	a0	a1	a2	a3	a4	a5	a6	a7
a0		a						
a1		c	b					
a2				d	f			
a3			e			h		k
a4			g			i		p
a5	l						n	
a6			t			m		r
a7	s							



КОЕФІЦІЄНТ КЕРОВАНОСТІ



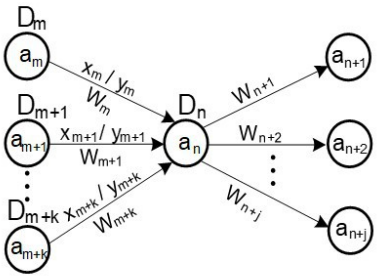
$$W_{m+k} = q_{m+k} * \sum_{i=1}^{R} p_r \qquad (1)$$

де q_{m+k} – кодова відстань між двійковими кодами станів автомата a_{m+k} та a_n (якщо для зберігання станів використовуються тригери типу T, JK), або кількість одиниць у вершини, куди входить дуга (якщо для зберігання станів використовуються тригери типу D),

R – кількість умов переходу на дузі k,

p_r – вага відповідної умови переходу, для мікропрограмних автоматів p_r=1.

МЕТОД ОБЧИСЛЕННЯ КЕРОВАНОСТІ СТАНІВ КА



Керованість D_n вершини графа a_n обчислюється як мінімальна сума керованості вершин-попередників і коефіцієнтів керованості W_{m+k} для дузі $a_{m+k} \to a_n$:

$$D_n = \min_{k=1}^{K}(D_{m+k} + W_{m+k}) \qquad (2)$$

а) Початкова вершина a_0 ГП помічається і їй присвоюється керованість D_0=1.

б) Для кожної вершини, що має метку обчислюються коефіцієнти керованості для всіх вихідних дуг W_j, де j - кількість вихідних дуг, за формулою 1 (слайд 6), ці дуги також помічаються.

в) Для всіх вершин, які мають вхідні помічені дуги обчислюється керованість D_n за формулою 2. Ці вершини помічаються.

г) Пункти б) та в) слід повторювати доти, поки усі вершини графа не отримають мітки.

ПОКАЗНИК СКЛАДНОСТІ ШЛЯХУ

Для забезпечення діагностичного експерименту для путей обходу графа можна використовувати показник складності шляху для вибору путі. Як результат обчислення тестопридатності, показник складності p шляху Sp обчислюється за формулою:

$$S_P = \sum_{i=1}^{H} W_{m,n}$$

де H – кількість дуг для шляху p.

Для діагностичного експерименту з обходу графа переходів обирається той шлях, у якого Sp приймає мінімальне значення.

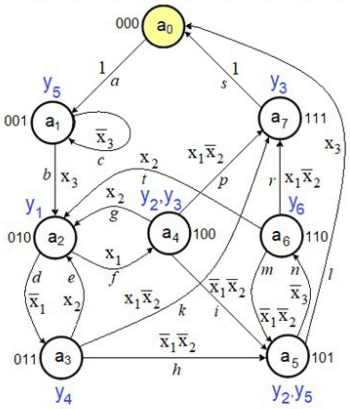
РЕАЛІЗАЦІЯ НЕРУЙНІВНОГО ДЕ ПО ОБХОДУ ВСІХ ВЕРШИН ГРАФА ПЕРЕХОДІВ КА

Умови Дірака (V=8 число вершин)

$\forall v_i \in V \quad deg\, v_i \geq [|V|/2]$

Ступінь кожної вершини $deg\, v_i \geq 4$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
deg	4	6	4	4	5	4	4



Умова Оре

Сума ступенів будь-яких двох несуміжних вершин не менше загального числа вершин в графі (≥ 8)

На основі теореми **Бонді-Хватала** для графа автомата для побудови множини гамільтонових циклів можна додати 4 пари несуміжних вершин, $(a_1 \leftrightarrow a_4)$, $(a_3 \leftrightarrow a_4)$, $(a_3 \leftrightarrow a_6)$, $(a_4 \leftrightarrow a_6)$.

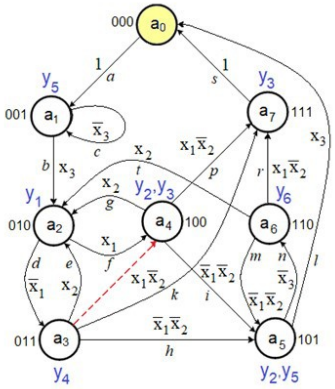
З точки зору мінімізації апаратурних витрат перевага віддається такому циклу для якого функція переваги (вага p-го шляху) буде

$$S_P = \min \sum_{i=1}^{H} W_{m,n}$$

При цьому слід враховувати, що вага додаткових дуг в графі дорівнює 1.

Проаналізувавши запропоновані додаткові дуги в графі можна зробити висновок, що тільки додатковий перехід **$(a_3 \leftrightarrow a_4)$** *дозволяє організувати гамільтонів цикл мінімальної ваги в цьому графі:*

$a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_5 \rightarrow a_6 \rightarrow a_7 \rightarrow a_0$



ОПТИМАЛЬНИЙ ГАМІЛЬТОНІВ ЦИКЛ ЗА МЕТОДОМ ГІЛОК ТА ГРАНИЦЬ

В математичному методі гілок та границь в якості ваги переходу будемо використовувати коефіцієнти ваги в матриці суміжності .

$$W_{m+k} = q_{m+k} * \sum_{i=1}^{R} p_r$$

Початкова матриця суміжності

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_0		1						
a_1			2					
a_2				1	2			
a_3			1			4		2
a_4			2			2		4
a_5	2						2	
a_6			1			4		2
a_7	3							

Підсумкова матриця досяжності

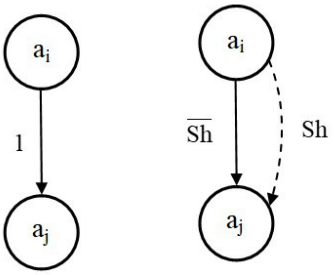
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_0		0						
a_1			0					
a_2				0	0			
a_3			0			3		0
a_4			0			0		1
a_5	0						0	
a_6			0			3		0
a_7	0							

За матрицею складемо цикл а0→а1→а2→а3→а7→а0, який не є гамільтоновим, бо відсутні вершини а4, а5, а6. З матриці видно, що ці вершини між собою пов'язані переходами а4→а5→а6. Оскільки з вершини а6 є перехід до вершини а7, для існування гамільтонова циклу достатньо **штучно** додати дугу, яка б поєднувала обидва фрагменти циклу, тобто дугу **а3 → а4**. Тоді маємо цикл а0→а1→а2→**а3→а4**→а5→а6→а7→а0, що є гамільтоновим. Застосування цього методу підтвердило наші припущення вибору шляху з мінімальною вагою через додаткову дугу **а3 → а4** .

РОЗМІЩЕННЯ Sh В РАЗІ БЕЗУМОВНОГО ПЕРЕХОДУ

Наступний метод модифікації керуючого автомата для підвищення його тестопридатності полягає в запровадженні додаткового вхідного сигналу (*Sh*) в модель автомата між усіма станами автомату. Це дає можливість встановити автомат в потрібний стан не більше ніж за (*n*-1) тактів, де *n* - число станів автомата.

Перший випадок складається в додаванні додаткового сигналу *Sh* до безумовного переходу



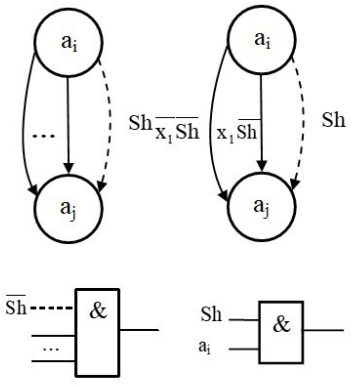
В даному варіанті розміщення функція переходу зі стану a_i в стан a_j матиме вигляд:

$$a_j = a_i Sh \vee a_i \overline{Sh} = a_i$$

Це означає відсутність додаткових витрат апаратури, якщо буде реалізована мінімальна форма рівняння. При реалізації вихідного рівняння додаткові апаратурні витрати становитимуть один вентиль двовходових (АБО) і два вентиля двовходових (І).

РОЗМІЩЕННЯ *Sh* В РАЗІ НАЯВНОСТІ ПЕРЕХОДІВ ТІЛЬКИ МІЖ a_i і a_j

Другий випадок складається в додаванні додаткового сигналу *Sh* до переходів між станами a_i і a_j, якщо відсутні переходи в інші стани, які відрізняються від стану a_j . Це випадок можливий тільки для автомата Мілі, оскільки в автоматі Мура не може бути мультідуг між двома станами.



В даному варіанті розміщення додаткового переходу функція переходу зі стану a_i в стан a_j матиме вигляд:

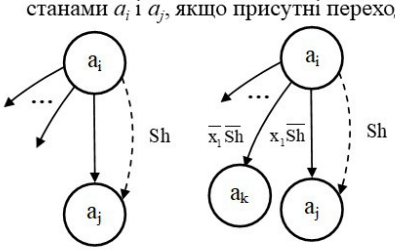
$$a_j = a_i Sh \vee a_i x_1 \overline{Sh} \vee a_i \overline{x_1}\ \overline{Sh} = a_i Sh \vee a_i \overline{Sh} = a_i$$

При реалізації вихідного рівняння додаткові апаратурні витрати становитимуть: додаткові входи інверсні *Sh* в усі вентилі, що реалізують терми вихідних рівнянь функції переходів $a_j = a_i x_1 \vee a_i \overline{x_1}$

і один вентиль, який реалізує терм $a_i Sh$.

РОЗМІЩЕННЯ *Sh* В РАЗІ НАЯВНОСТІ ПЕРЕХОДІВ МІЖ СТАНАМИ a_i і a_j ТА ІНШИХ ПЕРЕХОДІВ З a_i

Третій випадок складається в додаванні додаткового переходу з сигналом *Sh* до переходів між станами a_i і a_j, якщо присутні переходи в інші стани, які відрізняються від стану a_j



В даному варіанті розміщення додаткового переходу функція переходу зі стану a_i в стан a_j і переходу в стан матиме вигляд:

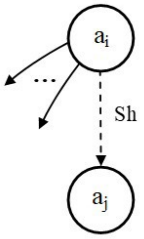
$$a_j = a_i Sh \vee a_i x_1 \overline{Sh} = a_i (Sh \vee x_1 \overline{Sh}) = a_i Sh \vee a_i x_1$$

При реалізації вихідного рівняння (не мінімального) для функції переходу в a_j додаткові апаратурні становитимуть: один вентиль, який реалізує терм a_iSh і один додатковий вхід інверсний *Sh* в вентиль, який реалізує терм первинної функції переходів. Якщо число переходів $a_i \rightarrow a_j$ в вихідному графі переходів більше одиниці, то для кожного з цих переходів з'являється додатковий вхід інверсний *Sh* у вентилі.

На додаток до цього, в даному випадку $\forall(a_i \rightarrow a_k)$, $a_k \neq a_j$, переходи $a_i \rightarrow a_k$ також дають додаткові апаратурні витрати у вигляді одного додаткового входу інверсний *Sh* в вентиль, який реалізує терм первинної функції переходів. Для даного прикладу це $a_k = a_i x_1 \overline{Sh}$.

РОЗМІЩЕННЯ *Sh* В РАЗІ ВІДСУТНОСТІ ПЕРЕХОДІВ МІЖ СТАНАМИ a_i і a_j

Четвертий випадок складається в додаванні додаткового переходу з сигналом *Sh* між станом a_i і станом a_j, якщо взагалі відсутній перехід $a_i \rightarrow a_j$.



В цьому випадку додаткові апаратурні витрати складають: один вентиль, який реалізує терм a_iSh і по одному додатковому входу інверсний *Sh* в ті вентилі, які $\forall(a_i \rightarrow a_k)$, $a_k \neq a_j$ реалізують, терми первинної функції переходів.

При цьому $\forall(a_s \rightarrow a_j)$, $a_s \neq a_i$, додатковий сигнал *Sh* не впливає на переходи $a_s \rightarrow a_j$.

Для вибору шляху розміщення додаткових переходів з сигналом Sh з точки зору мінімізації витрат апаратури найкращим є перший варіант, потім другий і найбільш невигідними є третій або четвертий варіанти.

ФУНКЦІЯ ПЕРЕВАГИ ДЛЯ ВИБОРУ ВЕРШИНИ ПРИ РОЗМІЩЕННІ ДУГ Sh

При обранні переходу ($a_n \rightarrow a_j$), на якому розміщується додатковий перехід *Sh*, у разі наявності двох та більше вихідних дуг з вершини a_n, використовується функція переваги

$$F = \min_{j=1}^{J} \left| D_n - D_j \right|$$

де D_n - досяжність вершини, з якої виходить додаткова дуга
Sh, D_j - досяжність вершини-наступника,
$j = \overline{1,J}$ - кількість вихідних дуг з вершини D_n.

Виходячи з цього додаткова дуга ставиться на переході, для якого функція переваги мінімальна.

Для розглянутого прикладу отримано наступний цикл обходу вершин графа переходів:
$a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_5 \rightarrow a_6 \rightarrow a_4 \rightarrow a_7 \rightarrow a_0$

МЕТОД ВИКОРИСТАННЯ ЗСУВНОГО РЕГІСТРУ

```
library IEEE;
use IEEE.std_logic_1164.all;
-- Описание интерфейса устройства
entity FSM is
   port (Clk, Reset: in STD_LOGIC;
         x1, x2, x3: in STD_LOGIC;
         A, TDI: in STD_LOGIC;
y1, y2, y3, y4, y5, y6: out STD_LOGIC);
end;
```

Сигнал **A** - управління режимами регістра, при **A = 1** виконується нормальний режим функціонування автомата, при **A = 0** виконується зрушення регістра вправо на один розряд.
Сигнал **TDI** - вхід для послідовного введення інформації в зсувний регістр.

```
-- Описание архитектуры устройства
architecture FSM of FSM is
signal  State, NextState: STD_LOGIC_vector (7 downto 0);
begin
-- Блок для формирования последовательностной части
Sreg0_CurrentState: process (Clk, reset, State, x1, x2, x3)
begin
      if Reset='1' then   State <= "00000000";
         elsif Clk'event and Clk = '1' then
           if A='1'   then State <= NextState;
             else   State <= TDI & State(7 downto 1);
           end if;
      end if;
end process;
-- Блок для формирования комбинационнрй части
…
end;
```

ХАРАКТЕРИСТИКИ РОЗГЛЯНУТИХ МОДЕЛЕЙ ДЛЯ ПОРІВНЯЛЬНОГО АНАЛІЗУ

Характеристики	Вихідна FSM	FSM зі зсувним регістром	FSM з Sh (min)	FSM з Sh (max)	FSM з одним Sh $a_6 \rightarrow a_4$	FSM з одним Sh $a_3 \rightarrow a_4$
Тип кодування	Унітарне	Унітарне	Грея	Грея	Унітарне	Унітарне
Швидкодія (max) МГц	362,3	237,1	308,6	275,4	345,1	345,1
Втрати швидкодії, %	-	34	14	24,8	5,24	5,24
Період CLK (min), нс	2,76	4,218	3,24	3,63	2,9	2,9
Кількість Slices	7	15	9	9	6	6
Кількість FF	8	8	3	3	8	8
Кількість LUTs	13	28	15	16	11	11
Кількість IOBs	11	13	12	12	12	12
Витрати по Квайну для КЧ	39	272	108	123	46	42
Загальні витрати по Квайну	143	376	147	165	150	146
Приріст загальних витрат, %	-	162,9	2,79	13,28	4,9	2,1

ВИСНОВКИ

– запропоновано метод аналізу тестопридатності цифрових автоматів, представлених у вигляді орієнтованого графа для спрощення задачі генерації тестів. Метод заснований на топологічному аналізі орієнтованого графа і кодів його вершин;

– удосконалено алгоритм побудови дерева рішень для обходу станів ГП автомата при наявності циклів для організації ДЕ, запропоновано евристичний алгоритм обходу графа за гамільтоновим циклом;

– розроблено методи підвищення тестопридатності керуючих автоматів, перший метод полягає в організації зсувного регістру для установки автомата в будь-який заданий стан з його допомогою;

– другий розроблений метод полягає в додавання сигналу Sh, ініціюючого додаткові переходи між станами автомата для організації гамільтонова циклу при обході станів, він мінімізує додаткові апаратурні витрати за рахунок організації визначеного обходу станів автомату.

– зроблено аналіз апаратурних витрат для другого методу при різних варіантах організації додаткового переходу між станами автомату в залежності від наявності безумовного переходу, умовного переходу та відсутності переходів між станами автомату;

– реалізовано VHDL - моделі для проведення порівняльного аналізу розроблених методів.

Наукова новизна полягає в подальшому розвитку методів оцінки і забезпечення тестопридатності КА та удосконалено алгоритм побудови дерева рішень для обходу станів ГП автомата при наявності циклів для організації ДЕ.

Практична значимість: розроблені методи призначаються для САПР тестопридатних пристроїв та систем, і дають можливість проектувати легкотестовані КА.

ДОДАТОК Б

Листинги VHDL модулів

Листинг 1

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Описание интерфейса устройства
entity FSM is
     port (Clk, Reset: in STD_LOGIC;
          x1, x2, x3: in STD_LOGIC;
           y1, y2, y3, y4, y5, y6: out STD_LOGIC);
end;


-- Описание архитектуры устройства
architecture FSM of FSM is
     type State_type is (a0, a1, a2, a3, a4, a5, a6, a7);
     signal State, NextState: State_type;

begin
-- Блок для формирования последовательностной части
Sreg0_CurrentState: process (Clk, reset)
begin
     if Reset='1' then
          State <= a0;
     elsif Clk'event and Clk = '0' then
          State <= NextState;
     end if;
end process;


-- Блок для формирования комбинационной части
-- Описание переходов состояний по условиям
Sreg0_NextState: process (State, x1, x2, x3)
begin

     case State is
        when a0=> NextState <= a1;
          when a1=> if x3='1' then NextState <= a2;
                                   else    NextState <= a1;
                     end if;
          when a2=> if x1='1' then NextState <= a4;
                      else     NextState <= a3;
                      end if;
          when a3=> if x2='1' then NextState <= a2;
                     elsif x1='1' then  NextState <= a7;
                      else     NextState <= a5;
                    end if;
          when a4=> if x2='1' then NextState <= a2;
                    elsif x1='1' then   NextState <= a7;
                      else     NextState <= a5;
                      end if;
          when a5=> if x3='1' then NextState <= a0;
                      else     NextState <= a6;
```

```
                    end if;
          when a6=> if x2='1' then NextState <= a2;
                    elsif x1='1' then   NextState <= a7;
                      else     NextState <= a5;
                      end if;
          when a7=> NextState <= a0;


          when others => NextState <= a0;
     end case;
end process;

-- Описание выходных сигналов
y1<='1' when State=a2 else '0';
y2<='1' when State=a4 or State=a5 else  '0';
y3<='1' when State=a4 or State=a7 else '0';
y4<='1' when State=a3 else '0';
y5<='1' when State=a1 else    '0';
y6<='1' when State=a6 else '0';

end;
```

Листинг 2

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Описание интерфейса устройства
entity FSM is
     port (Clk, Reset: in STD_LOGIC;
          x1, x2, x3: in STD_LOGIC;
           A, TDI: in STD_LOGIC;
           y1, y2, y3, y4, y5, y6: out STD_LOGIC);
end;


-- Описание архитектуры устройства
architecture FSM of FSM is
        signal  State, NextState: STD_LOGIC_vector (7 downto 0);
begin
-- Блок для формирования последовательностной части
Sreg0_CurrentState: process (Clk, reset, State, x1, x2, x3)
begin
     if Reset='1' then
          State <= "00000000";
     elsif Clk'event and Clk = '1' then
       if A='1'   then
          State <= NextState;
       else   State <= TDI & State(7 downto 1);
     end if;

     end if;
```

```
end process;


-- Блок для формирования комбинационной части
-- Описание переходов состояний по условиям
Sreg0_NextState: process (State, x1, x2, x3)
begin

case State is
 when "00000000"=> NextState <= "00000001";
 when "00000001"=> if x3='1' then  NextState <= "00000010";
                     else     NextState <="00000001";
                     end if;
 when "00000010"=> if x1='1' then  NextState <= "00000100";
                      else     NextState <= "00000011";
                      end if;
 when "00000011"=> if x2='1' then  NextState <= "00000010";
                     elsif x1='1' then  NextState <= "00000111";
                      else     NextState <= "00000101";
                    end if;
 when "00000100"=> if x2='1' then  NextState <= "00000010";
                    elsif x1='1' then   NextState <= "00000111";
                      else     NextState <= "00000101";
                      end if;
 when "00000101"=> if x3='1' then  NextState <= "00000000";
                      else     NextState <="00000110";
                      end if;
 when "00000110"=> if x2='1' then  NextState <= "00000010";
                    elsif x1='1' then   NextState <= "00000111";
                      else     NextState <="00000101";
                      end if;
 when "00000111"=> NextState <= "00000000";

          when others => NextState <= "00000000";
     end case;
end process;

-- Описание выходных сигналов
y1<='1' when State="00000010" else '0';
y2<='1' when State="00000100" or State="00000101" else'0';
y3<='1' when State="00000100" or State="00000111" else '0';
y4<='1' when State="00000011" else '0';
y5<='1' when State="00000001" else'0';
y6<='1' when State="00000110" else '0';

end;
```

Листинг 3

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
 ENTITY test IS
```

```
END test;

ARCHITECTURE behavior OF test IS
     -- Component Declaration for the Unit Under Test (UUT)
     COMPONENT FSM
    PORT(
         Clk : IN  std_logic;
         Reset : IN  std_logic;
         x1 : IN  std_logic;
         x2 : IN  std_logic;
         x3 : IN  std_logic;
         A : IN  std_logic;
         TDI : IN  std_logic;
         y1 : OUT  std_logic;
         y2 : OUT  std_logic;
         y3 : OUT  std_logic;
         y4 : OUT  std_logic;
         y5 : OUT  std_logic;
         y6 : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal Clk : std_logic := '0';
   signal Reset : std_logic := '0';
   signal x1 : std_logic := '0';
   signal x2 : std_logic := '0';
   signal x3 : std_logic := '0';
   signal A : std_logic := '0';
   signal TDI : std_logic := '0';

     --Outputs
   signal y1 : std_logic;
   signal y2 : std_logic;
   signal y3 : std_logic;
   signal y4 : std_logic;
   signal y5 : std_logic;
   signal y6 : std_logic;

   -- Clock period definitions
   constant Clk_period : time := 100 ns;

BEGIN

     -- Instantiate the Unit Under Test (UUT)
   uut: FSM PORT MAP (
          Clk => Clk,
          Reset => Reset,
          x1 => x1,
          x2 => x2,
          x3 => x3,
          A => A,
          TDI => TDI,
```

```
          y1 => y1,
          y2 => y2,
          y3 => y3,
          y4 => y4,
          y5 => y5,
          y6 => y6
        );

   -- Clock process definitions
   Clk_process :process
   begin
          Clk <= '0';
          wait for Clk_period/2;
          Clk <= '1';
          wait for Clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
Reset <='1'; wait for Clk_period;
Reset <='0';  A<='1';wait for Clk_period;
                                    x3 <='1'; wait for Clk_period;
                                    x3    <='0';        wait    for
Clk_period;
                                        x2    <='1';    wait    for
Clk_period;

                      A<='0'; TDI<='1';  wait for Clk_period*8;

      -- insert stimulus here

      wait;
   end process;

END;
```

Листинг 4

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Описание интерфейса устройства
entity FSM is
     port (Clk, Reset: in STD_LOGIC;
          Sh, x1, x2, x3: in STD_LOGIC;
           y1, y2, y3, y4, y5, y6: out STD_LOGIC);
end;
```

```
-- Описание архитектуры устройства
architecture FSM of FSM is
     type State_type is (a0, a1, a2, a3, a4, a5, a6, a7);
     signal State, NextState: State_type;

begin
-- Блок для формирования последовательностной части
Sreg0_CurrentState: process (Clk, reset)
begin
     if Reset='1' then
          State <= a1;
     elsif Clk'event and Clk = '0' then
          State <= NextState;
     end if;
end process;


-- Блок для формирования комбинационной части
-- Описание переходов состояний по условиям
Sreg0_NextState: process (State, x1, x2, x3, Sh)
begin

     case State is
        when a0=>  if Sh ='1' then NextState <= a1;
                              else NextState <= a1;
                              end if;
          when a1=> if Sh ='1' then    NextState <= a2;
                   elsif x3='1' then   NextState <= a2;
                                 else  NextState <= a1;
                    end if;
          when a2=>  if Sh ='1' then   NextState <= a3;
                      elsif x1='1' then NextState <= a4;
                      else    NextState <= a3;
                      end if;
          when a3=> if Sh ='1' then    NextState <= a5;
                     elsif x2='1' then NextState <= a2;
                     elsif x1='1' then NextState <= a7;
                      else    NextState <= a5;
                   end if;
          when a4=> if Sh ='1' then    NextState <= a7;
                    elsif x2='1' then  NextState <= a2;
                    elsif x1='1' then  NextState <= a7;
                      else    NextState <= a5;
                      end if;
          when a5=> if Sh ='1' then    NextState <= a6;
                    elsif x3='1' then  NextState <= a0;
                      else    NextState <= a6;
                      end if;
          when a6=> if Sh ='1' then    NextState <= a4;
                    elsif x2='1' then  NextState <= a2;
                    elsif x1='1' then  NextState <= a7;
                      else    NextState <= a5;
                      end if;
```

```
          when a7=> if Sh ='1' then     NextState <= a0;
                                else NextState <= a0;
                                end if;

          when others => NextState <= a1;
     end case;
end process;

-- Описание выходных сигналов
y1<='1' when State=a2 else '0';
y2<='1' when State=a4 or State=a5 else '0';
y3<='1' when State=a4 or State=a7 else '0';
y4<='1' when State=a3 else '0';
y5<='1' when State=a1 else '0';
y6<='1' when State=a6 else '0';

end;
```

Листинг 5

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Описание интерфейса устройства
entity FSM is
     port (Clk, Reset: in STD_LOGIC;
          Sh, x1, x2, x3: in STD_LOGIC;
           y1, y2, y3, y4, y5, y6: out STD_LOGIC);
end;


-- Описание архитектуры устройства
architecture FSM of FSM is
     type State_type is (a0, a1, a2, a3, a4, a5, a6, a7);
     signal State, NextState: State_type;

begin
-- Блок для формирования последовательностной части
Sreg0_CurrentState: process (Clk, reset)
begin
     if Reset='1' then
          State <= a1;
     elsif Clk'event and Clk = '0' then
          State <= NextState;
     end if;
end process;

-- Блок для формирования комбинационной части
-- Описание переходов состояний по условиям
Sreg0_NextState: process (State, x1, x2, x3, Sh)
begin
```

```
     case State is
        when a0=>  if Sh ='1' then NextState <= a4;
                        else NextState <= a1;
                    end if;
          when a1=> if Sh ='1' then    NextState <= a3;
                   elsif x3='1' then   NextState <= a2;
                        else NextState <= a1;
                    end if;
          when a2=>  if Sh ='1' then   NextState <= a7;
                      elsif x1='1' then NextState <= a4;
                      else    NextState <= a3;
                     end if;
          when a3=> if Sh ='1' then    NextState <= a6;
                    elsif x2='1' then  NextState <= a2;
                     elsif x1='1' then NextState <= a7;
                      else    NextState <= a5;
                    end if;
          when a4=> if Sh ='1' then    NextState <= a2;
                    elsif x2='1' then  NextState <= a2;
                    elsif x1='1' then  NextState <= a7;
                      else    NextState <= a5;
                     end if;
          when a5=> if Sh ='1' then    NextState <= a1;
                    elsif x3='1' then  NextState <= a0;
                      else    NextState <= a6;
                    end if;
          when a6=> if Sh ='1' then    NextState <= a0;
                    elsif x2='1' then  NextState <= a2;
                    elsif x1='1' then  NextState <= a7;
                      else    NextState <= a5;
                     end if;
          when a7=> if Sh ='1' then    NextState <= a5;
                        else NextState <= a0;
                    end if;

          when others => NextState <= a1;
     end case;
end process;

-- Описание выходных сигналов
y1<='1' when State=a2 else '0';
y2<='1' when State=a4 or State=a5 else '0';
y3<='1' when State=a4 or State=a7 else '0';
y4<='1' when State=a3 else '0';
y5<='1' when State=a1 else '0';
y6<='1' when State=a6 else '0';

end;
```

Листинг 6

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
-- Описание интерфейса устройства
entity FSM is
     port (Clk, Reset: in STD_LOGIC;
          x1, x2, x3, Sh: in STD_LOGIC;
           y1, y2, y3, y4, y5, y6: out STD_LOGIC);
end;


-- Описание архитектуры устройства
architecture FSM of FSM is
     type State_type is (a0, a1, a2, a3, a4, a5, a6, a7);
     signal State, NextState: State_type;

begin

-- Блок для формирования последовательностной части
Sreg0_CurrentState: process (Clk, reset)
begin
     if Reset='1' then
          State <= a0;
     elsif Clk'event and Clk = '0' then
          State <= NextState;
     end if;
end process;


-- Блок для формирования комбинационной части
-- Описание переходов состояний по условиям
Sreg0_NextState: process (State, x1, x2, x3, Sh)
begin

     case State is
        when a0=> NextState <= a1;
          when a1=> if x3='1' then NextState <= a2;
                      else    NextState <= a1;
                    end if;
          when a2=> if x1='1' then NextState <= a4;
                      else     NextState <= a3;
                    end if;
          when a3=> if x2='1' then NextState <= a2;
                     elsif x1='1' then  NextState <= a7;
                      else     NextState <= a5;
                   end if;
          when a4=> if x2='1' then NextState <= a2;
                    elsif x1='1' then   NextState <= a7;
                      else     NextState <= a5;
                    end if;
          when a5=> if x3='1' then NextState <= a0;
                      else     NextState <= a6;
                    end if;
          when a6=> if Sh ='1' thenNextState <= a4;
                    elsif x2='1' then   NextState <= a2;
```

```
                    elsif x1='1' then   NextState <= a7;
                      else     NextState <= a5;
                   end if;
          when a7=> NextState <= a0;

          when others => NextState <= a0;
     end case;
end process;

-- Описание выходных сигналов
y1<='1' when State=a2 else '0';
y2<='1' when State=a4 or State=a5 else  '0';
y3<='1' when State=a4 or State=a7 else '0';
y4<='1' when State=a3 else '0';
y5<='1' when State=a1 else    '0';
y6<='1' when State=a6 else '0';

end;
```

ДОДАТОК В

Тези доповідей 24 міжнародного науково-технічної конференції «Проблеми інформатизації»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ
XXIV МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

7 – 9 квітня 2020 р.

Том 5

КОНФЕРЕНЦІЯ
«ВІРТУАЛЬНИЙ ТА ФІЗИЧНИЙ КОМП'ЮТІНГ»

Харків 2020

XXIV Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 5. – Харків: ХНУРЕ. 2020. – 216 с. – pdf 3,4 Mb.

В збірник включені матеріали
XXIV Міжнародного молодіжного форуму
«Радіоелектроніка та молодь у XXI столітті»

Видання підготовлено
факультетом комп'ютерної інженерії та управління
Харківського національного університету радіоелектроніки

61166 Україна, Харків, просп. Науки, 14
тел./факс: (057) 7021397

E-mail: mref21@nure.ua

ДОСЛІДЖЕННЯ ВПЛИВУ ЗАХОДІВ ЩОДО ЗАБЕЗПЕЧЕННЯ ТЕСТОПРИДАТНОСТІ ЦИФРОВИХ КЕРУЮЧИХ АВТОМАТІВ НА АПАРАТУРНІ ВИТРАТИ

Трегуб Р.Р.
Науковий керівник – доц. Кулак Е.М.
Харківський національний університет радіоелектроніки
(61166, Харків, просп. Науки,14, каф. АПОТ, тел. (057) 702-13- 26)
e-mail: rostyslav.trehub@nure.ua, +380501322107

Problems of minimal additional hardware costs during design automation of easy-testable digital finite state machines (FSM) are considered. The reasonable way of the FSM setting into an arbitrary state is to expand the state table, which improves the controllability of FSM' states and leads to the transformation of their HDL-models. Hardware costs for different methods of hardware redundancy introduction to HDL-models of finite state machines are estimated.

Тестопридатніть - це один з найбільш важливих показників, який повинен враховуватися при проектуванні цифрових пристроїв поряд з такими показниками, як швидкодія і вартість пристрою. Низький рівень тестопригодності пристрою призводить до збільшення часу і погіршення якості його тестування. Методи підвищення тестопридатності цифрових пристроїв (ЦП) шляхом внесення в схемну реалізацію апаратурної надлишковості досить розвинені та широко використовуються при проектуванні [1]. Функціональні методи тестопридатного проектування ЦП детально розглянуті в [2], де для цифрових автоматів, представлених у формі таблиць переходів-виходів (ТПВ), введені поняття діагностованих і визначно-діагностлваних класів автоматів та запропоновані способи приведення ТПВ автоматів до зазначених класів. Розглянуто процедури проведення діагностичного єксперіменту з автоматами з використанням установчих, синхронізуючих, діагностичних та характеристичних послідовностей. Крім того, обґрунтована ідея підвищення тестопридатності автомата за рахунок внесення апаратурною надмірності шляхом розширення вхідного алфавіту, вихідного алфавіту та алфавіту станів. В [3] запропонований спосіб кодування станів дозволяє знизити споживану потужність схемних реалізацій кінцевих автоматів. У роботі [4] з метою підвищення тестопридатності керуючого кінцевого автомату авторами викладена концепція введення апаратурної надлишковості в модель абстрактного автомату, яка полягає у в додаванні стовпця Sh у таблицю переходів-виходів (ТПВ) (додаткова дуга Sh у графі переходів автомата), що дає змогу встановити автомат в будь який стан не більше, ніж за n-1 тактів. Зазначено, що додаткові апаратурні витрати при цьому не перевищують 25-30% в залежності від типу автомата та способу кодування його станів, але при цьому використовувався природний порядок обходу

станів автомата, що безумовно впливає на додаткові апаратурні витрати. Таким чином, актуальною є задача мінімізації додаткових апаратурних витрат за рахунок організації оптимального обходу станів автомату.

Об'єкт дослідження: HDL-моделі цифрових керуючих автоматів. Предмет дослідження: аналіз впливу заходів щодо забезпечення тестопридатності цифрових керуючих автоматів на апаратурні витрати. Мета дослідження: знайти найменш витратний спосіб обходу всіх станів автомата при введенні дуги Sh. Задача – дослідити всі можливі способи додавання дуги Sh графа переходів автомата, і вибрати найбільш підходящий.

Проаналізовані апаратурні витрати при різних варіантах організації додаткового переходу між станами автомату Sh в залежності від наявності безумовного переходу, умовного переходу та відсутності переходів між станами автомату. При обранні додаткового переходу обирається той стан-наступник, для якого сумарна оцінка апаратурних витрат для функцій збудження мінімальна з урахуванням кодів станів автомата.

Зазначений підхід підвищує керованість станів автомата, що значно покращує його тестопридатність. Моделювання та синтез розширених моделей засобами САПР XILINX ISE підтвердили отримання тестопридатності автоматів та мінімальних апаратурних витрат. Наукова новизна полягає в подальшому розвитку методу оптимізації апаратурних витрат при підвищенні тестопридатності кінцевих автоматів за рахунок розширення вхідного алфавіту в HDL-моделях в формі автоматного шаблону, що дає можливість автоматизувати процес проектування тестопридатних автоматів з мінімальними апаратурними витратами.

Список використаних джерел:

1. Gorodetsky A. Introduction to JTAG and DFT technology. Testing in edge scanning technologies and testable design. Palmarium Academic Publishing, Germany, 2012, 308 p.

2. Solov'ev V.V. Minimization of mealy finite-state machines by using the values of the output variables for state assignment [Text] / V.V. Solov'ev // Journal of Computer and Systems Sciences International. – January 2017. – Volume 56, Issue 1. – P. 96–104.

3. Solov'ev V.V. Minimization of Power Consumption of Finite State Machines by Splitting Their Internal States [Text] / T.N.Grzes, V.V. Solov'ev // Journal of Computer and Systems Sciences International.– 2015. – Vol. 54, No. 3. – P. 367–374.

4. Shkil A. Design Automation of Testable Finite State Machines [Text] / M.Miroschnyk, Y. Pakhomov, E. German, A. Shkil, E. Kulak, D. Kucherenko // Proceedings of the International Sympos. EWDTS'2017, September 29-October 2, 2017 – Novi Sad, Serbia, 2017. – P.203–208.

ДОДАТОК Г

Тези доповідей 25 міжнародного науково-технічної конференції «Проблеми інформатизації»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ 25-го МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

20 – 22 квітня 2021 р.

Том 5

КОНФЕРЕНЦІЯ «ВІРТУАЛЬНИЙ ТА ФІЗИЧНИЙ КОМП'ЮТІНГ»

Харків 2021

УДК 004.032.2+004.7.032.2](06)

25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 5. – Харків: ХНУРЕ. 2021. – 232с.

В збірник включені матеріали 25-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті».

Видання підготовлено факультетом комп'ютерної інженерії та управління Харківського національного університету радіоелектроніки

61166 Україна, Харків, просп. Науки, 14
тел./факс: (057) 7021397

E-mail: mref21@nure.ua

АЛГОРИТМ ПОСТРОЕНИЯ ДЕРЕВА РЕШЕНИЙ ДЛЯ НЕРАЗРУШАЮЩЕГО ДИАГНОСТИЧЕСКОГО ЭКСПЕРИМЕНТА НАД КОНЕЧНЫМ УПРАВЛЯЮЩИМ АВТОМАТОМ

Трегуб Р.Р.
Научный руководитель – доц. Кулак Э.Н.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, просп. Науки,14, каф. АПВТ, тел. (057) 702-13- 26)
e-mail: rostyslav.trehub@nure.ua

A heuristic algorithm for constructing a decision tree for an FSM transition graph with cycles is offered. The decision tree obtained by the proposed algorithm is intended for constructing a non-destructive diagnostic experiment for the FSM.

Введение. Низкий уровень тестопригодности устройства приводит к увеличению времени и ухудшению качества его тестирования. Функциональные методы тестопригодного проектирования цифровых устройств подробно рассмотрены в [1], где для цифровых автоматов, представленных в форме таблиц переходов-выходов, введены понятия диагностируемых классов автоматов. Рассмотрены процедуры проведения диагностического эксперимента с автоматами с использованием установочных, синхронизирующих, диагностических и характеристических последовательностей. В работе [2] с целью повышения тесто-пригодности конечного управляющего автомата (УА) авторами изложена концепция введения аппаратурной избыточности в модель абстрактного автомата. Для построения теста реализуется стратегия обхода всех дуг графа переходов (ГП) УА, начиная с начальной вершины. Данный подход предусматривает проведение так называемого «неразрушающего» диагностического эксперимента (НДА), в котором в конце каждой проверки автомат логично или принудительно возвращается в исходное состояние. Для реализации стратегии обхода всех дуг ГП по методике, предложенной в [3], строится дерево решений для обхода путей (маршрутов) графа. Эта методика не позволяет обрабатывать нелинейные ГСА, порождающие циклы в ГП. В связи с этим: *объект исследования* – граф переходов УА; *предмет исследования* – построение дерева решений, дающего все пути обхода ГП; *цель исследования* – разработка эвристического алгоритма построения дерева решений для ГП с циклами.

Содержание исследования. Для описания графа переходов автомата предложено использовать модифицированную матрицу смежности размером (M x M), где M - количество вершин в графе автомата. Данная матрица характеризуется тем, что для каждой вершины графа переходов (сроки матрицы) в ячейках матрицы находятся имена дуг, соединяющих данную вершину с вершинами-преемниками (столбцами). При этом в одной ячейке могут быть более одной дуги (мультиграф).

Для реализации стратегии обхода всех дуг графа строится бинарное дерево решений (дерево обхода графа) по следующему алгоритму. 1. Построение дерева начинается с начальной вершины a_0. Анализируется первая строчка матрицы смежности и составляется список вершин-преемников. 2. Из списка преемников избирается первая вершина и включается в дерево, соответствующая ячейка в матрице (дуга) помечается. 3. Для строки, соответствующей этому номеру, составляется список x преемников. При дальнейшем анализе списка преемников преимущество отдается непомеченным ячейкам. 4. Построение маршрута в графе завершается в следующих случаях: очередная вершина-преемник является начальной (терминальной) вершиной a_0 (завершенный цикл работы автомата); исходная дуга является петлей (элемент главной диагонали матрицы смежности); в маршруте появляется вершина, которая уже присутствует в этом маршруте (цикл в графе переходов, фиксируется начальная вершина цикла). 5. П. 2, 3 и 4 повторяются до тех пор, пока все маршруты не дойдут до терминальной (конечной) вершины. 6. Петли и маршруты с циклами заносятся в отдельный стек. Петли могут вставляться в любые маршруты, где есть соответствующие вершины. 7. Для маршрутов из стека циклов достраиваются маршруты возвращения в начальную вершину путем использования фрагментов маршрутов с этой вершины до терминальной вершины (такой маршрут всегда будет, исходя из стандартных условий досягаемости графа).

Выводы. *Научная новизна* состоит в дальнейшем развитии алгоритма построения дерева решений ГП автомата при наличии циклов. *Практическая значимость* – дерево решений, построенное по предложенному алгоритму предназначено для построения НДЭ над УА.

Список использованной литературы:

1. Solov'ev V.V. Minimization of mealy finite-state machines by using the values of the output variables for state assignment / V.V. Solov'ev // Journal of Computer and Systems Sciences International. – January 2017.– Volume 56, Issue 1. – .p. 96–104.

2. Shkil A. Design Automation of Testable Finite State Machines [Text] / M.Miroschnyk, Y. Pakhomov, E. German, A. Shkil, E. Kulak, D. Kucherenko // Proceedings of the International Sympos. EWDTS'2017, September 29-October 2, 2017 – Novi Sad, Serbia, 2017. – p. 203–208.

3. Шкиль, А.С. Диагностирование hdl-моделей микропрограммных автоматов [Текст] / А.С. Шкиль, Э.Н. Кулак, А.С. Серокурова // АСУ приборы автоматики. Вып. 172, – 2015. с. 22-31.