

# ТЕСТВАНЕ И ДИАГНОСТИКА

## Упражнение 6

### **Автоматично генериране на тестови вектори за последователни логически схеми**

#### Основни принципи

Почти всички използвани в практиката цифрови схеми са от последователен тип. Проблеми:

1. **Вътрешни състояния на паметите.** В началото състоянието на паметите е неопределено. Трябва да се моделира поведението, така че състоянието на паметите да е функция само на състоянието на първичните входове.
2. **Дълги тестващи последователности.** Тестването на последователна логика има 3 етапа:
  - a. инициализиране на вътрешната памет;
  - b. изработване на тестващи последователности, като се използват алгоритмите, използвани при комбинационната логика;
  - c. ако повреда влияе на повече от един елемент памет, се наблюдава един от всички.

Дадена повреда в последователни схеми се тества чрез подредена последователност от входни вектори (фиг.1).

Уговорки:

1. Схемите се разглеждат като изградени от комбинационна логика + тригери. Така при анализа на поведението на схемата се разглеждат 2 типа входове и изходи:
  - a. първични;
  - b. псевдопървични (вховете и изходите на тригерите).
2. Счита се, че тригерите работят перфектно. Могат да се повреждат само връзките.

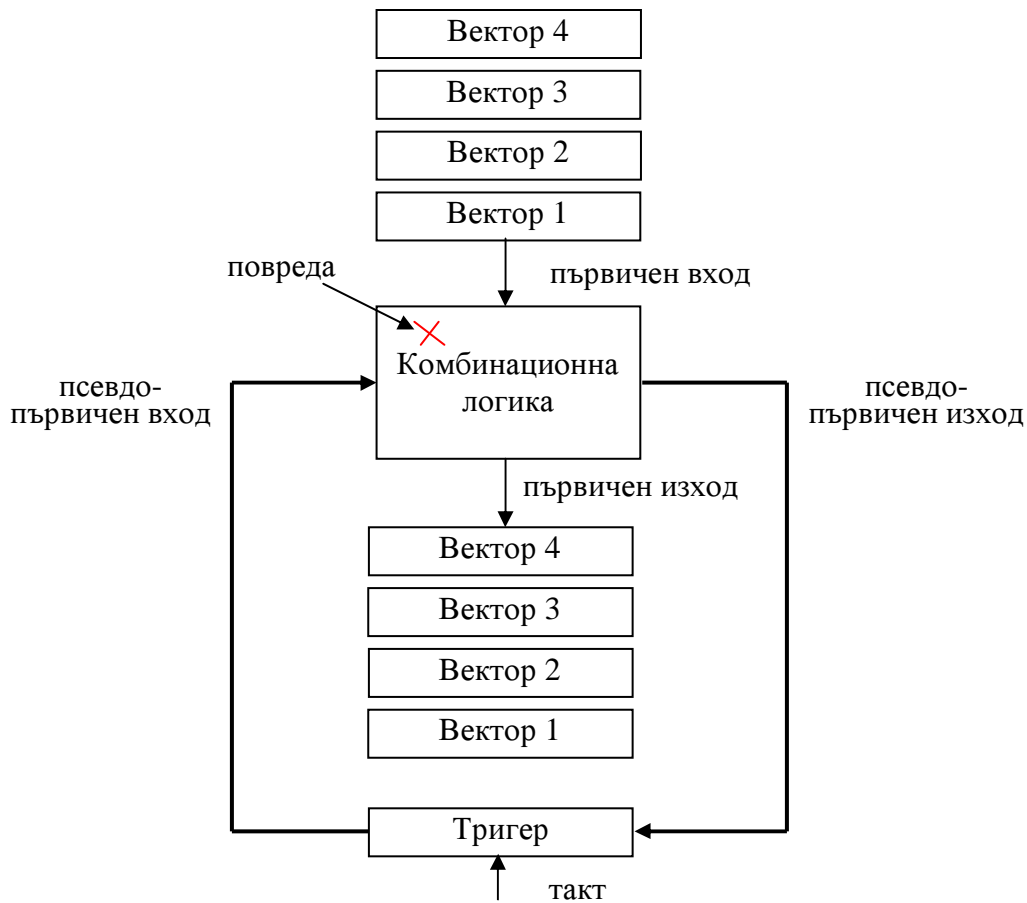
#### Методи:

##### 1. Разгънати времеви рамки (Time-frame expansion).

Последователната схема се преобразува в комбинационна. Схемата се развързва в мястото на тригерите. Всеки изход на тригер се явява псевдопървичен вход, а всеки вход на тригер - псевдопървичен изход. Последователно се свързват толкова копия на схемата, колкото такта са необходими за да се разпространи повреда до някой първичен изход. Методът дава добри резултати при вериги, описани на логическо ниво (чрез булеви елементи). Ефективността на метода намалява при: наличие на циклични структури, множество тактови входове, асинхронни вериги.

##### 2. Методи, базирани върху симулации.

За създаване на тестващите последователности се използват симулатор на грешки и генератор на вектори. Методът може да се приложи върху всички схеми, които могат да се симулират. Могат да се използват модели на схемите от различни нива (логическо, регистрово, транзисторно и др.).



фиг.1

## Метод на разгънатите времеви рамки



**Пример 1** На фиг.2 е показана схема на последователен суматор. Когато  $n$ -тия бит ( $A_n$  и  $B_n$ ) постъпят във веригата, се получава сумата  $S_n$  и пренос  $C_{n+1}$ , която се зарежда в тригера. Така сумирането на две 32-битови двоични числа се свежда до следните процедури:

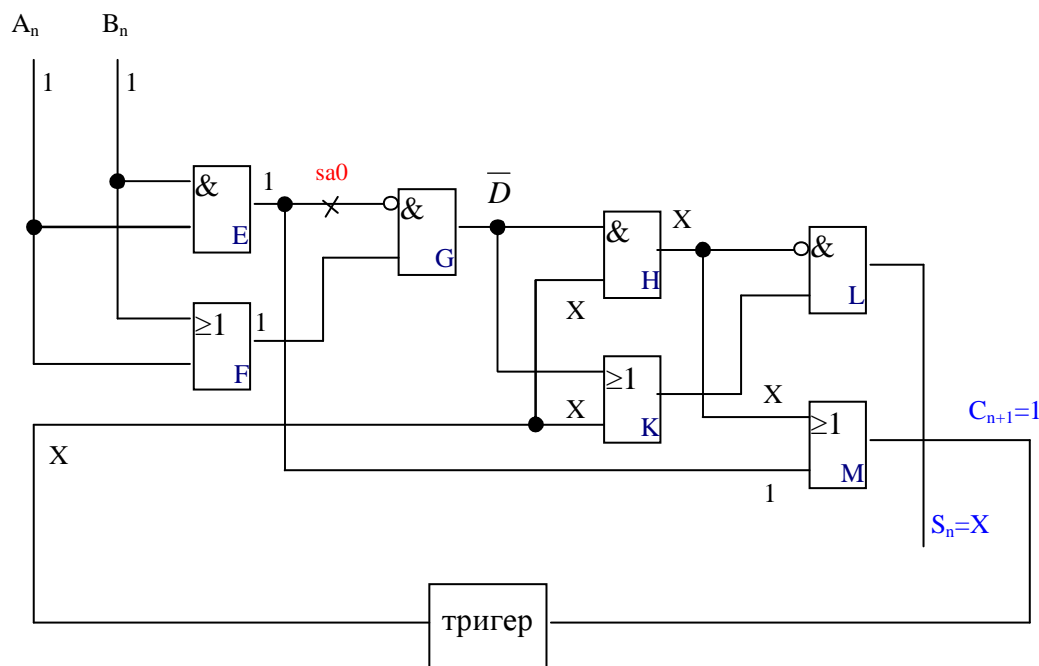
### 1. Инициализация.

Подава се (00) на първичните входове, за да се инициализира тригера в състояние 0 (пренос 0). Първичния изход се игнорира.

### 2. Последователно сумиране.

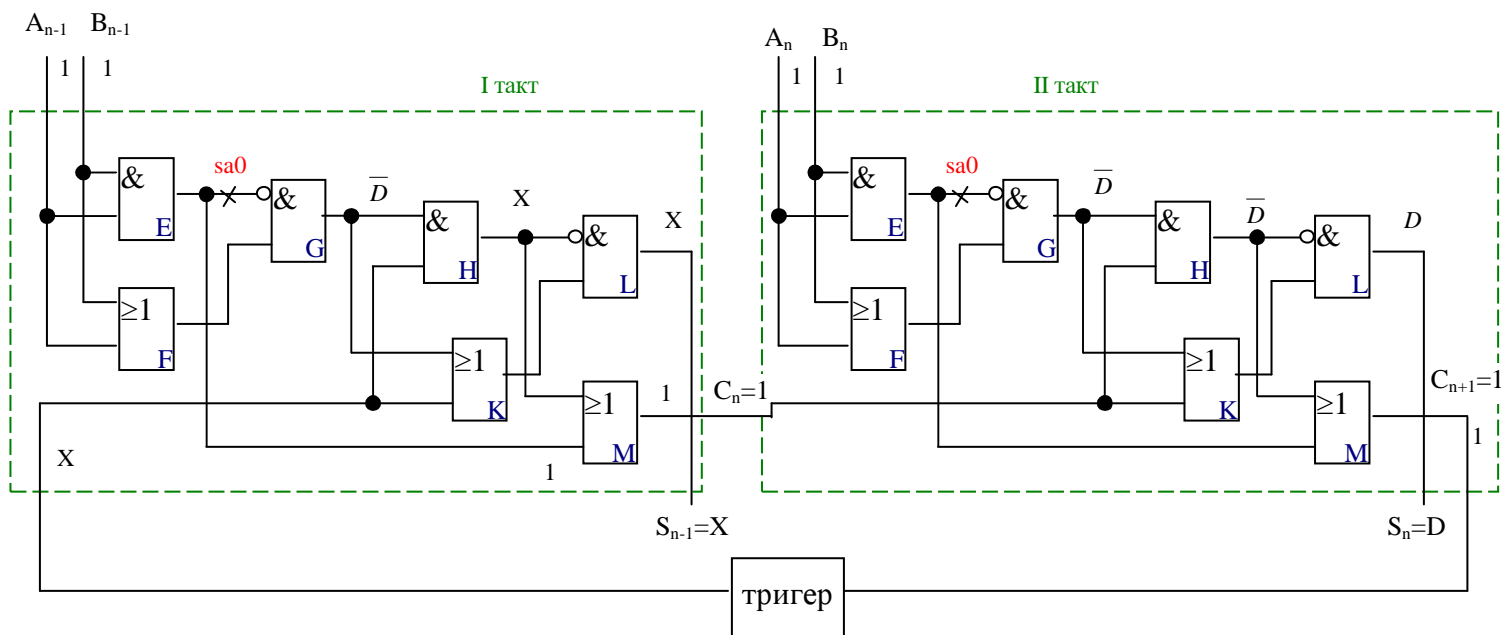
На всеки такт се подават поредните битове ( $A_iB_i$ ), като за всяка двойка се извършва сумиране на двата бита и преноса от предходната операция (от изхода на тригера). На първичния изход излиза сумата, а на тригера се подава преноса.

Върху схемата прилагаме  $D$  – алгоритъма за да намерим тествашите вектори. Правим обратно обхождане и установяваме, че за да се прояви повредата на двата входа трябва да се подаде вектора (11). Продължаваме по пътя на разпространение на повредата и установяваме, че преносът е 1, а на първичния изход  $S_n$  излиза стойност X. Този пренос би стабилизирал поведението на схемата на следващия такт.



фиг.2

Чертаем последователно две копия на схемата – съответно за I и за II такт (фиг.3).  $C_n$  се авява псевдопървичен изход за I такт и псевдопървичен вход за II такт. На изхода  $S_n$  вече излиза състояние D.



фиг.3



**Пример 1** Да се намери тестваща последователност за откриване на повреда от тип слепване към 1 на входа на логическия елемент G (фиг.2).

Правим обратно обхождане от повредената връзка към първичните входове. Достатъчно е поне един от входовете да има стойност 0, за да се прояви слепването в повредената връзка (състояние  $\bar{D}$ ). Следователно допустим е всеки един от следните вектори: (00), (01), (10). Продължаваме по пътя на разпространение на повредата. За да може елемента G да работи като повторител, е необходимо на изхода на елемента F да се получи 1. Това се



И така последователността от входни вектори, които откриват повреда sa1 на входа на елемента G е ((11),(01)) или ((11),(10)).

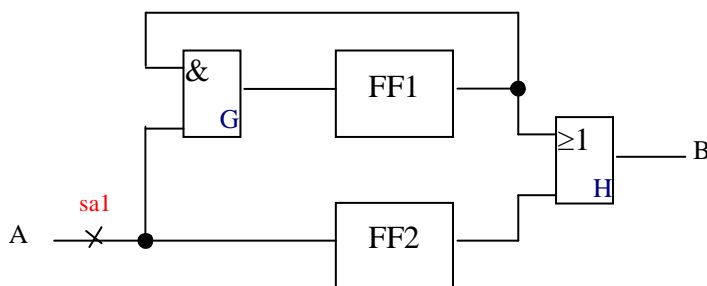
Съществуват ситуации, при които 5-валентните състояния не водят до решение. За това се е наложило използването на 9 валентна логика.

Всяка една връзка може да бъде в някое от следните състояния:

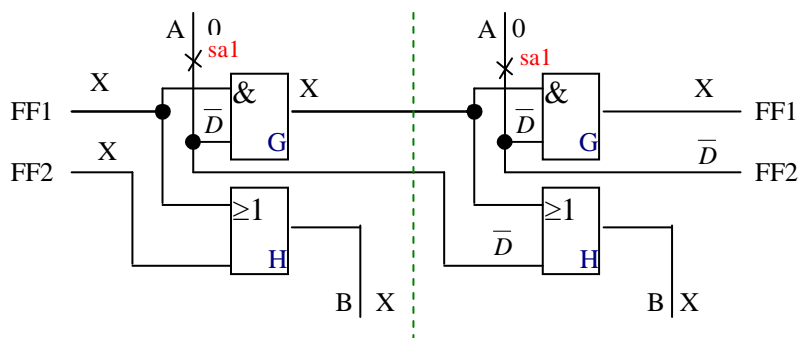
Състояние	Значение	Годна схема	Повредена схема
0	0/0	0	0
1	1/1	1	1
$D$	1/0	1	0
$\bar{D}$	0/1	0	1
X	X/X	X	X
G0	0/X	0	X
G1	1/X	1	X
F0	X/0	X	0
F1	X/1	X	1



**Пример 2** На фиг.6 е показана последователна схема с два тригера. Разгънатите времеви рамки са показани на фиг.7. Петвалентната логика не позволява откриване на показаната повреда.



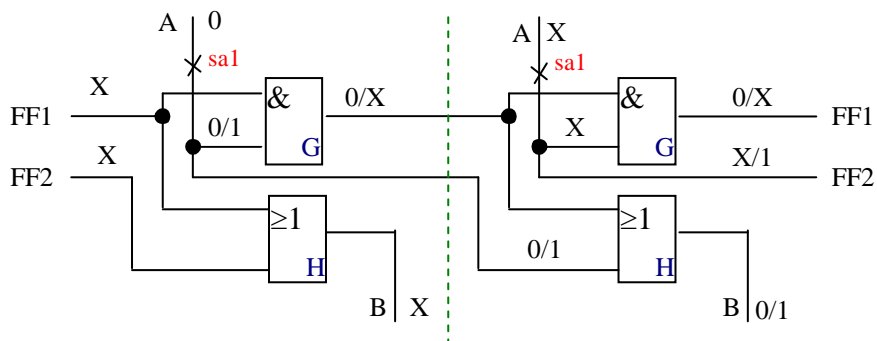
фиг.6



фиг.7

Описваме поведението на схемата като използваме 9 валентна логика (фиг.8). Състоянието на връзките ще описваме с двойките <годна>/<повредена>. Например 0/X – 0 при годна и X при повредена схема. През първия такт подаваме на първичния вход A 0. Вследствие на слепването състоянията на клоновете, излизащи от разклонението са в състояние (0/1). На изхода на G след първия такт излиза (0/X) – без повреда изхода е в състояние 0, а при

повреда повтаря състоянието на FF1. Състоянието на изхода на H и без и при наличие на повреда е X. При втория такт не зависи от състоянието на първичния вход A. Следователно няма значение какво е състоянието на A и затова го маркираме като X. Състоянието на първичния изход B е 0 при изправна схема и 1 при повредена.



фиг.8



**Задача 1.** Използвайте 9-валентна логика за да откриете тестващите последователности за повредата от фиг.2.



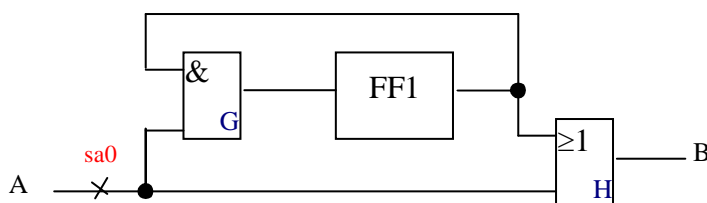
**Задача 2.** Намерете тестващата последователност за повредата sa0 на изхода на тригера (фиг.2).



**Задача 2.** Намерете тестващата последователност за повредата sa0 на изхода на логическия елемент M (фиг.2).



**Задача 3.** Покажете, че посредством 5-валентна логика не може да се намери тестваща последователност за повредата от фиг.9.



фиг.9