

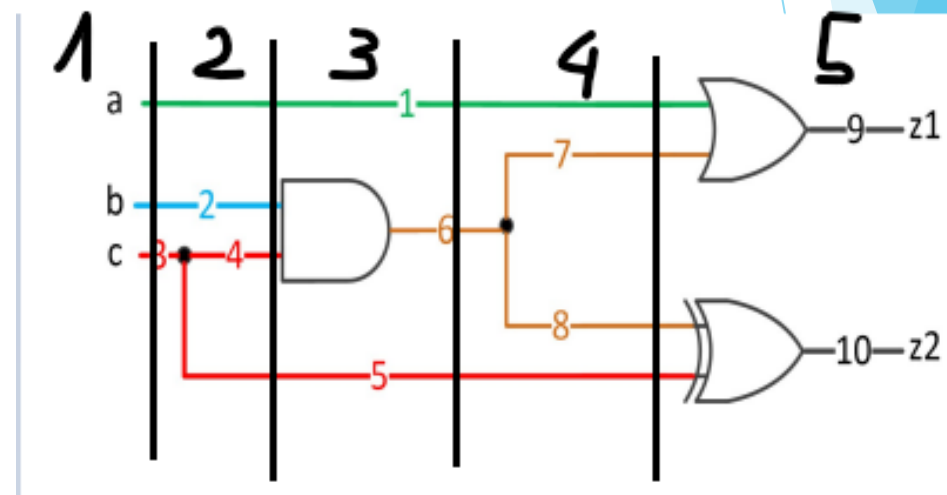
Program de simulare și testare a structurilor logice combinacionale

Autor: Negoită Tudor

Coordonator științific: Prof. Dr. Ing. Petru Cașcaval

Structura circuitului

- ▶ La circuitul studiat porțile se dispun pe niveluri și poziții consecutive în cadrul nivelului.
- ▶ Se impune condiția ca la intrarea unei porți să nu fie conectate decât intrări primare sau porți de pe niveluri inferioare. Astfel se evită ca intrarea unei porți să depindă direct sau indirect de valoarea logică de la ieșire.



Notații folosite pentru tipurile de porți studiate

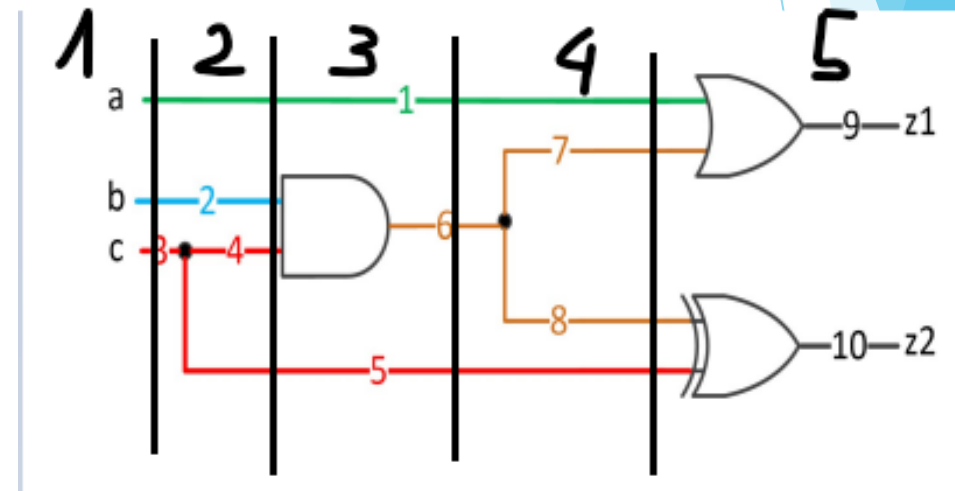
- ▶ Pentru a descrie porțile în fișierul de descriere vom folosi notații specifice pentru tipul porții.
- ▶ Intrările primare - In
- ▶ Porțile AND și NAND - a și A
- ▶ Porțile OR și NOR - o și O
- ▶ Porțile XOR și XNOR - x și X
- ▶ Conexiunile de tip fan-out - s

Structura fișierului de descriere a circuitului studiat

- ▶ Pentru a descrie circuitul fiecare poartă va fi descrisă după următorul format <tipul_porții poziția_porții numărul_de_intrări pozițiile_intrărilor>.
- ▶ În cazul intrărilor primare e suficient să se precizeze poziția acestora.
- ▶ În cazul ieșirilor primare acestea se vor menționa la finalul fișierului.

Structura fișierului de descriere a circuitului studiat - Exemplu

```
In 1,1
In 1,2
In 1,3
s 2,1 1 1,3
s 2,2 1 1,3
a 3,1 2 1,2 2,1
s 4,1 1 3,1
s 4,2 1 3,1
o 5,1 2 1,1 4,1
x 5,2 2 4,2 2,2
Out 5,1
Out 5,2
```



Reprezentarea internă a circuitului studiat

- ▶ Se face printr-o matrice de structuri care conțin următoarele câmpuri:
- ▶ type - tipul porții
- ▶ ni - numărul de intrări
- ▶ valFN - valoarea porții în funcționare normală
- ▶ valCD - valoarea porții când există un defect în circuit
- ▶ inputs[][] - pozițiile intrărilor în matrice.
- ▶ stuckAtFault - un eventual defect

```
struct gate {  
    char type;  
    int ni;  
    int valFN;  
    int valCD;  
    int inputs[4][2];  
    int stuckAtFault;  
};
```

Simularea circuitului

- ▶ Combinația curentă aplicată la intrare se memorează în structurile de pe prima coloană a matricei de descriere.
- ▶ Parcurgerea circuitului se face de la ieșire către intrări folosind o funcție recursivă.
- ▶ Valoarea logică de la ieșirea unei porți se memorează în structura corespunzătoare.
- ▶ La parcurgerea recursivă pentru o poartă se urmăresc valorile dominante.


Funcționalități

- ▶ La rularea programului utilizatorului îi sunt prezentate 4 opțiuni:
- ▶ Generarea tabelii de adevăr a circuitului.
- ▶ Generarea tabelii de adevăr a circuitului în prezența unui defect specificat.
- ▶ Inserarea defectelor pentru un vector de test specificat.
- ▶ Generarea unor vectori de test care să acopere toate defectele posibile.

```
Enter the name of the file to be read: 3I_10_4G.txt
Select an option:
1. Generate truth table without fault
2. Generate truth table with a specified fault
3. Insert faults one by one for certain inputs
4. Generate a test vectors set that detect all faults
5. Exit
Enter your choice:
```


Generarea tabelii de adevăr a circuitului în prezența unui defect specificat

- ▶ Dacă utilizatorul alege a doua opțiune, acestuia i se va cere să specifice defectul pentru care se va genera tabela.
- ▶ Se va utiliza formatul <poziția_porții tipul_defectului>.
- ▶ Tabela generată va indica vectorii de test pentru care defectul este detectabil, marcând ieșirile la care acesta e vizibil.

 3,1_s0.txt - Notepad

File	Edit	Format	View	Help
x1	x2	x3	z1	z2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0*	1*
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1*

Inserarea defectelor pentru un vector de test specificat

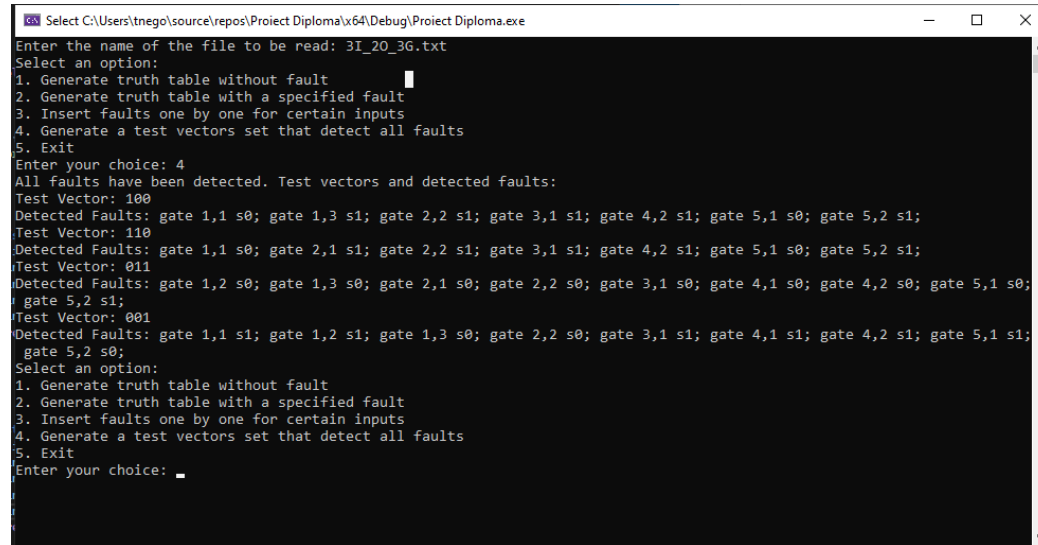
- ▶ Dacă utilizatorul alege a treia opțiune acestuia i se va cere să introducă un vector de test în ordinea intrărilor.
- ▶ Se evaluează circuitul fără defecte și se afișează output-urile.
- ▶ Utilizatorul are posibilitatea să insereze blocaje pe rând și pentru fiecare se vor afișa valorile ieșirilor și se va menționa dacă defectul este detectat la acestea.

Inserarea defectelor pentru un vector de test specificat

```
Enter your choice: 3
Enter the values for the primary inputs (e.g., 110 for 3 inputs). Enter -1 to stop.
Primary inputs: 10110
Evaluating circuit without faults...
Circuit outputs with the current fault:
Output at (6, 1): 0
Fault not detected
Output at (6, 2): 1
Fault not detected
Enter faults in the format 'row,col fault_type' (e.g., '3,2 s0' for stuck-at-0, '3,2 s1' for stuck-at-1). Enter -1 to stop.
Insert fault: 1,1 s0
Evaluating circuit with fault: 1,1 s0...
Circuit outputs with the current fault:
Output at (6, 1): 1
Fault detected
Output at (6, 2): 1
Fault not detected
Insert fault: 1,1 s1
Evaluating circuit with fault: 1,1 s1...
Circuit outputs with the current fault:
Output at (6, 1): 0
Fault not detected
Output at (6, 2): 1
Fault not detected
Insert fault:
```

Generarea unui set de vectori de test care să acopere toate defectele posibile

- ▶ La selectarea acestei opțiuni programul generează în mod aleatoriu vectori de test și simulează circuitul pentru toate defectele posibile.
- ▶ Când au fost detectate toate defectele se vor afișa vectorii de test și defectele pe care aceștia le acoperă.



```
Select C:\Users\tnego\source\repos\Proiect Diploma\src\Debug\Proiect Diploma.exe
Enter the name of the file to be read: 3I_20_3G.txt
Select an option:
1. Generate truth table without fault
2. Generate truth table with a specified fault
3. Insert faults one by one for certain inputs
4. Generate a test vectors set that detect all faults
5. Exit
Enter your choice: 4
All faults have been detected. Test vectors and detected faults:
Test Vector: 100
Detected Faults: gate 1,1 s0; gate 1,3 s1; gate 2,2 s1; gate 3,1 s1; gate 4,2 s1; gate 5,1 s0; gate 5,2 s1;
Test Vector: 110
Detected Faults: gate 1,1 s0; gate 2,1 s1; gate 2,2 s1; gate 3,1 s1; gate 4,2 s1; gate 5,1 s0; gate 5,2 s1;
Test Vector: 011
Detected Faults: gate 1,2 s0; gate 1,3 s0; gate 2,1 s0; gate 2,2 s0; gate 3,1 s0; gate 4,1 s0; gate 4,2 s0; gate 5,1 s0;
gate 5,2 s1;
Test Vector: 001
Detected Faults: gate 1,1 s1; gate 1,2 s1; gate 1,3 s0; gate 2,2 s0; gate 3,1 s1; gate 4,1 s1; gate 4,2 s1; gate 5,1 s1;
gate 5,2 s0;
Select an option:
1. Generate truth table without fault
2. Generate truth table with a specified fault
3. Insert faults one by one for certain inputs
4. Generate a test vectors set that detect all faults
5. Exit
Enter your choice: _
```

Modalitate de îmbunătățire

- ▶ La funcționalitatea prin care se generează vectorii de test care detectează toate defectele sunt analizate în total un număr de defecte egal cu dublul numărului de conexiuni.
- ▶ Acest lucru poate reprezenta o problemă pentru circuitele mari.
- ▶ Se poate reduce numărul de defecte undeva la jumătate prin aplicarea relațiilor de dominanță și echivalență dintre defecte.
- ▶ Prin reducerea numărului de defecte analizate se reduce mult și numărul de vectori de test necesari.

Relații de dominanță și echivalență - Poarta AND

- ▶ Se observă tabela de adevăr a porții AND.
- ▶ În cazul relației de echivalență se observă coloanele identice și se păstrează doar ieșirea porții.
- ▶ În cazul relației de dominanță observăm defectele ale căror coloane sunt incluse în coloana altui defect și se elimină efectul dominant.
- ▶ Se pot defini pentru poarta AND regulile acestea (se elimină a_{b0} , b_{b0} și z_{b1}).

a	b	z	a_{b0}	a_{b1}	b_{b0}	b_{b1}	z_{b0}	z_{b1}
0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0	1
1	0	0	0	0	0	1	0	1
1	1	1	0	1	0	1	0	1

Relații de dominanță și echivalență - Poarta OR

- ▶ Se observă tabela de adevăr a porții OR.
- ▶ În cazul relației de echivalență se observă coloanele identice și se păstrează doar ieșirea porții.
- ▶ În cazul relației de dominanță observăm defectele ale căror coloane sunt incluse în coloana altui defect și se elimină efectul dominant.
- ▶ Se pot defini pentru poarta OR regulile acestea (se elimină a_{b1}, b_{b1} și z_{b0}).

[illegible]

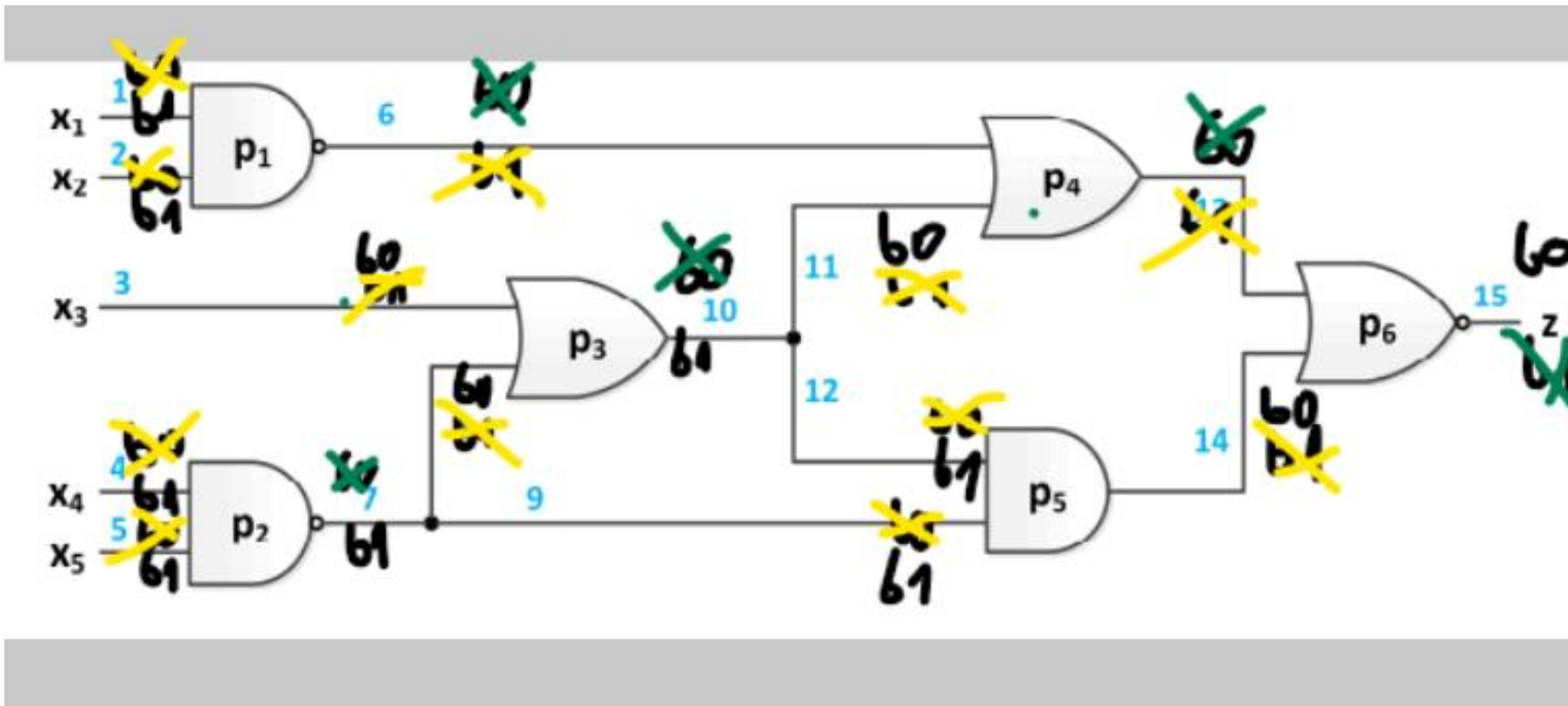
Relații de dominanță și echivalență - Poarta XOR

- ▶ În cazul porții XOR observăm că nu există relații de dominanță și echivalență între defecte.
- ▶ Trebuie analizate toate defectele.

a	b	z	a_{b0}	a_{b1}	b_{b0}	b_{b1}	z_{b0}	z_{b1}
0	0	0	0	1	0	1	0	1
0	1	1	1	0	0	1	0	1
1	0	1	0	1	1	0	0	1
1	1	0	1	0	1	0	0	1

Relații de dominanță și echivalență - Porțile negate

- ▶ Pentru poarta NOT se elimină a_{b0} și a_{b1} .
- ▶ Pentru porțile NAND, NOR și XNOR se aplică aceleași reduceri ca și pentru porțile lor complementare, doar că blocajul de la ieșire eliminat prin dominanță este opus (de exemplu dacă la poarta AND se elimină a_{b0}, b_{b0} și z_{b1} la poarta NAND se elimină a_{b0}, b_{b0} și z_{b0}).



Exemplu

Se poate observa cum prin aplicarea regulilor menționate anterior pentru acest circuit din 30 de defecte totale ar rămâne doar 13.

Concluzii

- ▶ Programul realizat are scop didactic și este util în analiza și sinteza circuitelor combinaționale ca și la proiectarea unui experiment de testare pentru defectele de tip blocaj.
- ▶ Relațiile de echivalență și de dominanță între defecte reduc studiul de testabilitate undeva la jumătate.
- ▶ Pentru determinarea vectorilor de test se pot aplica și algoritmi mai eficienți cum ar fi cel bazat pe metoda Roth.