Circuite integrate digitale
Îndrumar de laborator

Asistent drd. Zărnescu George

Introducere în Logisim

Logisim este un program în care se pot simula circuite digitale de complexitate diferită. Pentru a deschide Logisim este de ajuns să dăm dublu click pe pictograma programului, care este evidențiată în figura 1.

Figura 1 – Pictograma programului Logisim.

Acest program are caracter freeware și se găsește la adresa de internet:
http://ozark.hendrix.edu/~burch/logisim/download.html
Un tutorial pentru începători se poate găsi la adresa de internet:
http://ozark.hendrix.edu/~burch/logisim/docs/2.3.0/guide/tutorial/index.html

După ce programul a fost deschis, interfața acestuia ar trebui să arate ca în figura 2. În stânga sus se află pop-up-urile File, Edit, Project, Simulate, Window și Help. Sub aceste pop-up-uri găsim meniul rapid. În partea stângă observăm o pictogramă care ilustrează o mână. Acest buton se va folosi pentru a asigna valori semnalelor la intrările în circuit, adică ne va ajuta să simulam circuitele create. Următorul buton arată ca un cursor de mouse. Cu ajutorul lui putem să modificăm forma circuitelor, să plasăm componente în spațiul de lucru. Următoarea pictogramă, care are forma literei a mare (A) ne va ajuta să adnotăm fiecare circuit pe care îl creăm. După bara despărțitoare se găsesc cinci componente. Primul buton, pătratul cu verde, reprezintă un semnal de intrare pe 1 bit, al doilea buton, cercul cu verde, reprezintă un semnal de ieșire pe 1 bit, următoarele 3 butoane reprezintă proțile logice fundamentale NOT (NU), AND (ȘI) și OR (SAU).

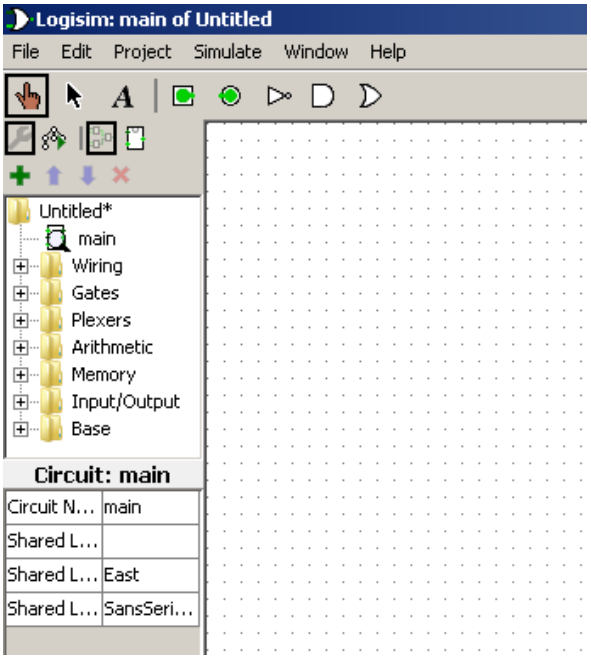


Figura 2 – Interfața programului Logisim.

Butoanele din figura 3 vor fi descrise pe larga în atunci când se va introduce elementul de memorie. Sub aceste butoane este o fereastra în care se găsește folderul cu module și folderele cu librării. Folderul cu module nu este salvat, de aceea se cheamă Untitled*. Acest folder conține un singur modul denumit main. Folderele cu librării vor fi descrise în paragraful următor. Mai jos de fereastra cu foldere se găsește panoul cu proprietăți. Acest

panou va apărea oricând vom selectra un modul din folderul cu module sau o componentă din folderele cu librării. Fiecare modul sau componentă va avea proprietăți specifice.

Figura 3 – Aceste butoane vor fi descrise în finalul documentului.

În figura 4 este evidențiat conținutul folderului Wiring. Din acest folder vom folosi trei componente Splitter, Pin și Clock.

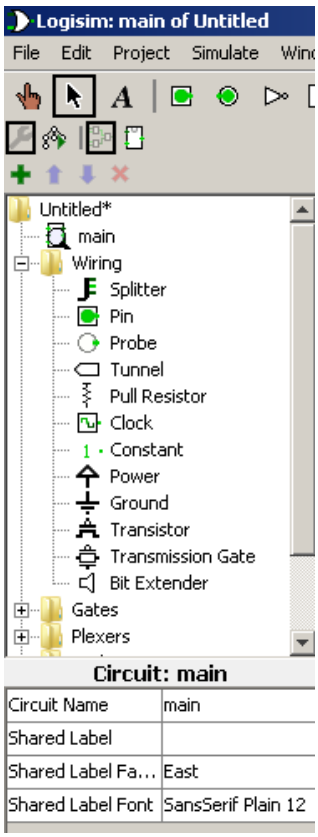


Figura 4 – Componetele folderului Wiring.

Componenta Pin putem să o accesăm și din meniul rapid descris mai sus. Componenta Pin reprezintă un semnal de intrare și este evidențiat în figura 5. Acest semnal de intrare poate fi pe 1 bit sau mai mulți biți. Pentru a genera semnale de intrare pe mai mulți biți, se va selecta componenta Pin și din panoul cu proprietăți se va selecta numărul dorit de biți, Data Bits. I se poate atribui un nume din tabul Label.

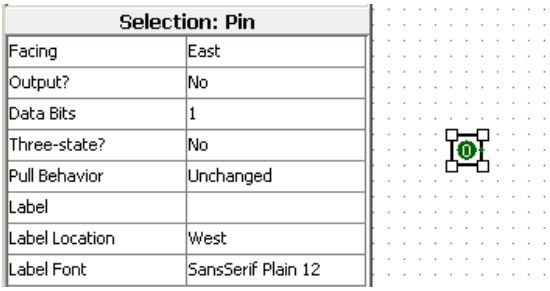


Figura 5 - Componenta Pin și proprietățile ei.

Splitter-ul este o componentă foarte utilă care ne ajută să grupăm mai multe fire pe 1 bit într-un singur fir pe mai mulți biți sau putem ramifica un fir pe mai mulți biți în mai multe fire pe 1 bit. Un fir pe mai mulți biți se numește bus sau magistrală. În figura 5 este evidențiat modul în care se poate conecta un splitter și proprietățile acestuia.

Selection: Splitter	
Facing	East
Fan Out	4
Bit Width In	4
Appearance	Left-handed
Bit 0	0 (Top)
Bit 1	1
Bit 2	2
Bit 3	3 (Bottom)

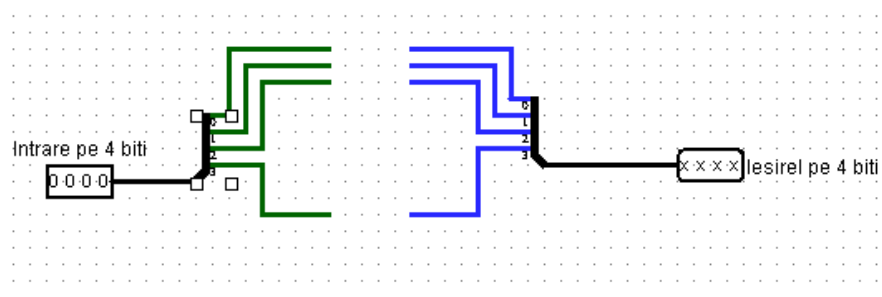


Figura 6 – Utilizarea splitter-ului și proprietățile acestuia.

În stânga figurii 6 sunt evidențiate proprietățile splitter-ului. Acesta este îndreptat către este, Facing East. Este împărțit în 4 fire, Fan Out 4, fiecare fir având 1 bit. Acest splitter acestă semnale de intrare pe 4 biți, Bit Width In 4. Orientarea splitter-ului selectat este spre stânga, Apprearance Left-handed. Cel mai important lucru la un spliter este ordinea biților. După cum se observă biții sunt organizați de sus în jos, Bit 0 0 (Top) ... Bit 3 3 (Bottom). Componenta clock va fi descrisă mai târziu când se va introduce elementul de memorie.

Folderele Gates, Plexers, Arithmetic, Memory conțin circuite logice, cu ajutorul cărora putem construi sisteme digitale de diferite complexități. Folderul Input/Output conține diferite tipuri de intrare sau ieșire. Folderul Base nu va fi folosit.

În continuare se va descrie felul în care se poate construi un circuit logic în spațiul de lucru, în folderul cu module, în modulul main. Din folderul Gates vom alege poarta AND. Vom da un click pe ea, o vom poziționa în spațiul de lucru și vom da încă o dată un click. În momentul de față poarta AND este selectată. În stânga jos a apărut panoul cu proprietăți. Vom alege să o denumim poarta_and4, tabul Label. Numărul de intrări va fi 4, Number of Inputs 4, fiecare intrare având un singur bit, Data Bits 1. Vom plasa în fereastra de lucru un spliter și o intrare pe 4 biți și o ieșire pe 1 bit. Pentru a face o conexiune, se va poziționa mouse-ul pe un terminal și se va trage un fir până la alt terminal. Circuitul ar trebui să arate ca în figura 7.

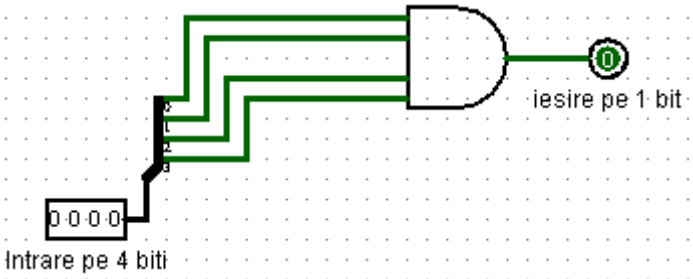


Figura 7 – Un circuit alcătui dintr-o intrare pe 4 biți, un splitter, o poartă AND cu 4 intrări și o ieșire pe 1 bit.

LAB
1

Porţi logice fundamentale

1.1 Obiectivele laboratorului

În acest laborator se vor defini conceptele de ierarhizare, modularitate şi regularitate, concepte ce vor fi utilizate în fiecare din următoarele laboratoare. Vor fi evidenţiate simbolurile, funcţiile logice şi funcţionarea elementelor digitale fundamentale de circuit. Aceste elemente fundamentele sunt denumite porţi logice fundamentale.

1.2 Definiții

1.2.1 Ierarhizare

Acest concept propune împărțirea sistemului digital în module, apoi divizarea acestor module în submodule până în momentul în care acestea sunt alcătuite din fragmente, al căror comportament este ușor de înțeles. Acest concept este evidențiata în figura 1 (vaza din ceară este alcătuită din postamentul, corpul și gâtul vazei).

1.2.2 Modularitate

Acest concept propune ca modulele să aibă un comportament și o interfață bine definite, astfel încât acestea să se conecteze între ele cu ușurință, fără să existe ulterior complicații neanticipate. Acest concept este evidențiat în figura 3 (modulul de bază este reprezentat de o celulă de forma unei prisme hexagonale).

1.2.3 Regularitate

Acest concept propune crearea unor module simple, care să fie reutilizate frecvent în diferite proiecte și reducerea numărului de module, care vor fi proiectate. Acest concept este evidențiat în figura 2 (peretele corpului vazei este alcătuit din modulul de bază, adică celule de forma unor prisme hexagonale).

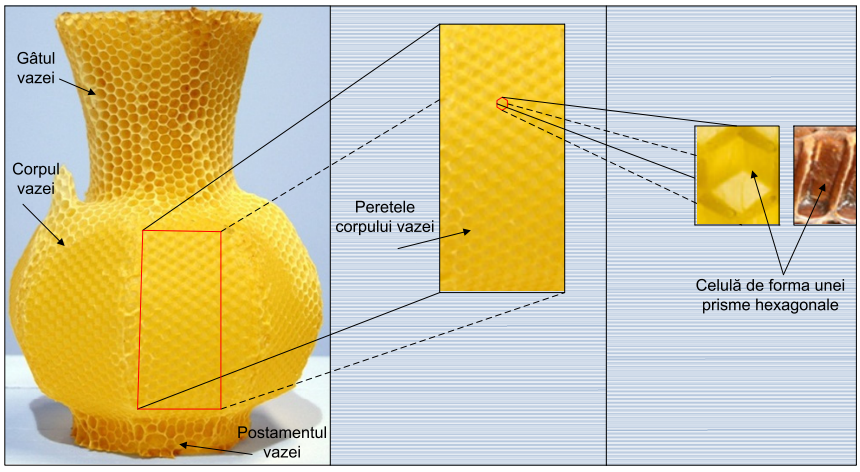


Figura 1 – Ierarhizare. **Figura 3 – Regularitate.** **Figura 2 – Modularitate.**

1.2.4 Simbolul porţilor logice fundamentale şi funcţia logică

Există 7 porţi logice fundamentale numite: NOT, AND, OR, NAND, NOR, XOR şi XNOR.

1.2.4.1 Poarta NOT (NU)

Simbolul acestei porţi este evidenţiat în figura 4. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 1.

Figura 4 – Poarta NOT (NU).

$$y = \mathrm{NOT}(x_1) = \overline{x_1} = !\,x_1 = x_1' \ (1)$$

1.2.4.2 Poarta AND (ŞI)

Simbolul acestei porţi este evidenţiat în figura 5. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 2.

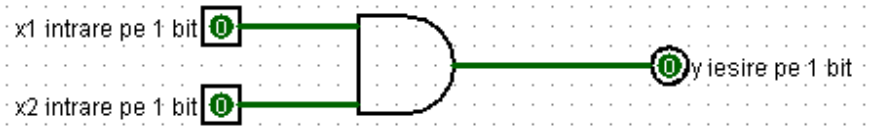


Figura 5 – Poarta AND (ŞI).

$$y = x_1 \mathrm{AND}\, x_2 = x_1 \cdot x_2 \ (2)$$

1.2.4.3 Poarta OR (SAU)

Simbolul acestei porţi este evidenţiat în figura 6. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 3.

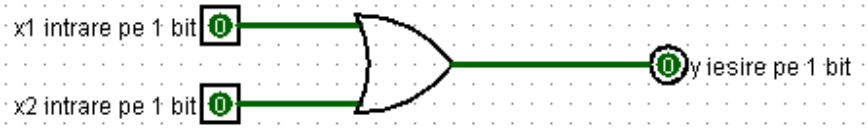


Figura 6 – Poarta OR (SAU).

$$y = x_1 \mathrm{OR}\, x_2 = x_1 + x_2 \ (3)$$

1.2.4.4 Poarta NAND (ŞI NU)

Simbolul acestei porţi este evidenţiat în figura 7. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 4.

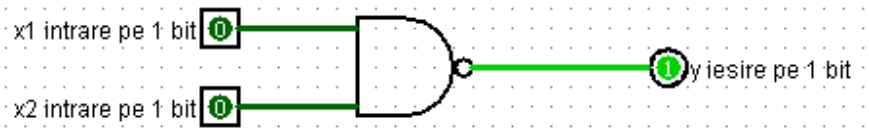


Figura 7 – Poarta NAND (ŞI NU).

$$y = x_1 \text{NAND}\, x_2 = \overline{x_1 \cdot x_2} \quad (4)$$

1.2.4.5 Poarta NOR (SAU NU)

Simbolul acestei porţi este evidenţiat în figura 8. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 5.

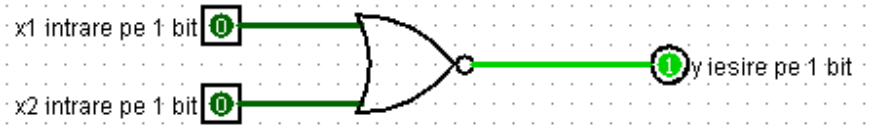


Figura 8 – Poarta NOR (SAU NU).

$$y = x_1 \text{NOR}\, x_2 = \overline{x_1 + x_2} \quad (5)$$

1.2.4.6 Poarta XOR (SAU EXCLUSIV)

Simbolul acestei porţi este evidenţiat în figura 9. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 6.

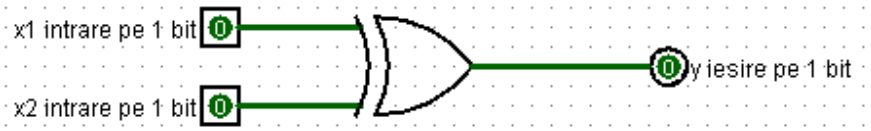


Figura 9 – Poarta XOR (SAU EXCLUSIV).

$$y = x_1 \text{XOR}\, x_2 = x_1 \oplus x_2 \quad (6)$$

1.2.4.7 Poarta XNOR (SAU NU EXCLUSIV)

Simbolul acestei porţi este evidenţiat în figura 10. Funcţia logică pentru această poartă fundamentală este evidenţiată în relaţia 7.

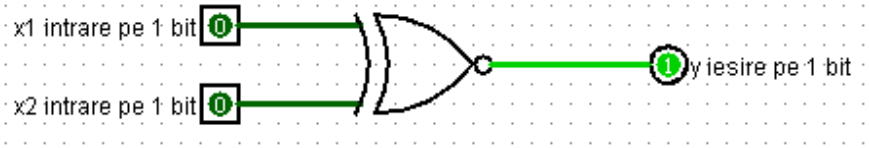


Figura 10 – Poarta XNOR (SAU NU EXCLUSIV).

$$y = x_1 \text{XNOR}\, x_2 = \overline{x_1 \oplus x_2} \quad (7)$$

1.3 Efectuarea laboratorului în Logisim

1.3.1 Porţi logice fundamentale

Exerciţiul 1.

Se vor instanţia în programul Logisim porţile logice fundamentale şi se vor executa conexiunile între intrări, ieşiri şi porţi logice ca în figura 11. Se va denumi fiecare intrare, ieşire şi portă logică. Se vor completa tabelele de adevăr din figura 12.

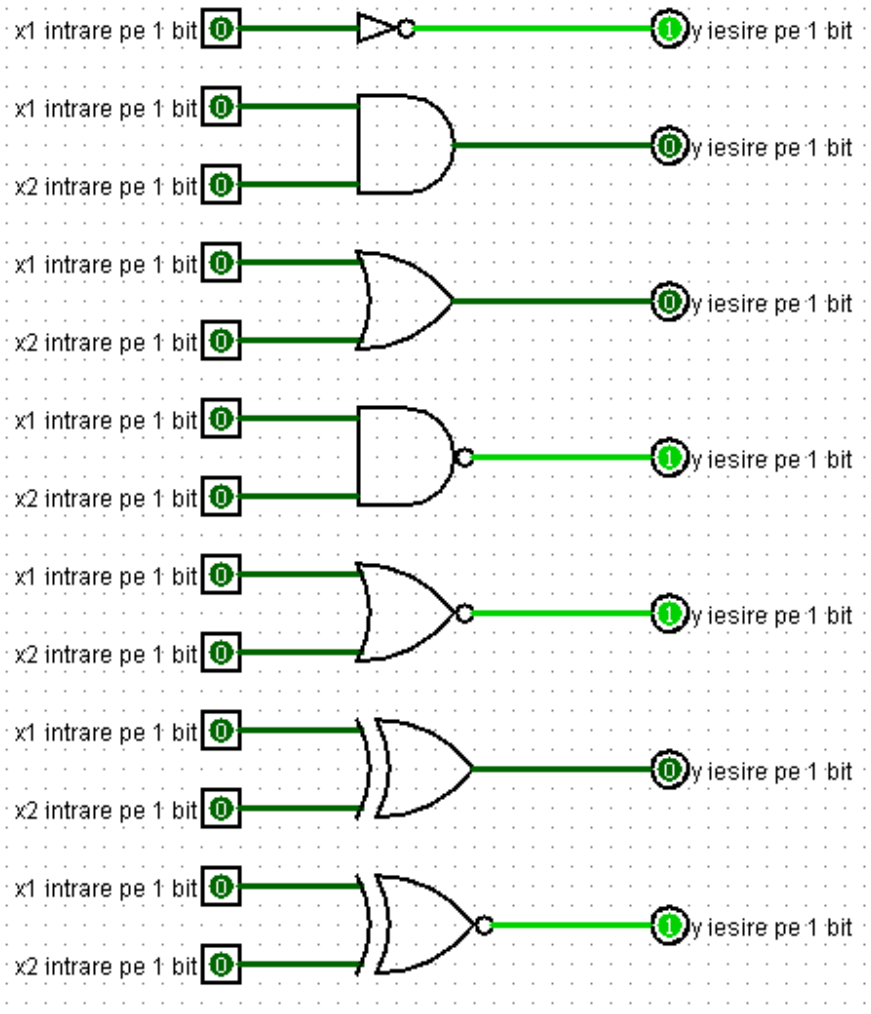


Figura 11 – Porţi logice fundamentale.

Poarta NU	
x	y
0	1
1	0

Poarta ŞI		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta SAU		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta ŞI NU		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta SAU NU		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta SAU EXCLUSIV		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta SAU NU EXCLUSIV		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Figura 22 – Tabelele de adevăr pentru porţile logice fundamentale.

1.3.2 Poarta NAND cu două intrări pe 4 biţi şi o ieşire pe 4 biţi

Exerciţiul 1.

Se va executa circuitul din figura 13 şi se va completa tabelul din figura 15. Circuitul din figura 13 este alcătuit dintr-o poartă NAND cu 2 intrări pe 4 biţi şi o ieşire pe 4 biţi. Acest circuit este echivalent cu circuitul din figura 14 şi este creat utilizând conceptul de regularitate. În figura 14 avem 4 porţi logice, fiecare având 2 intrări pe 1 bit şi o ieşire pe 1 bit.

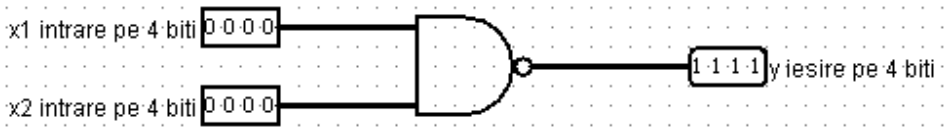


Figura 13 – Circuit logic cu poarta NAND cu 2 intrări pe 4 biţi şi o ieşire pe 4 biţi.

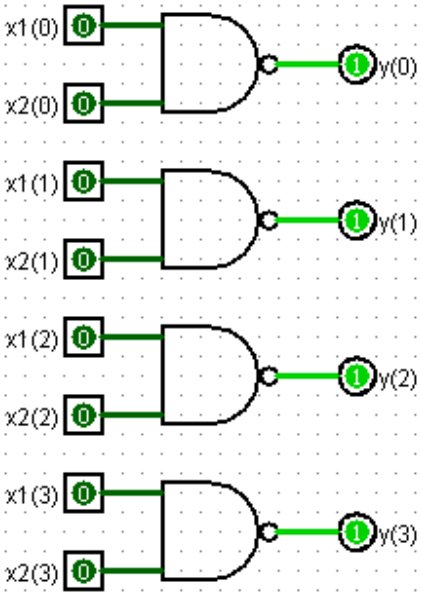
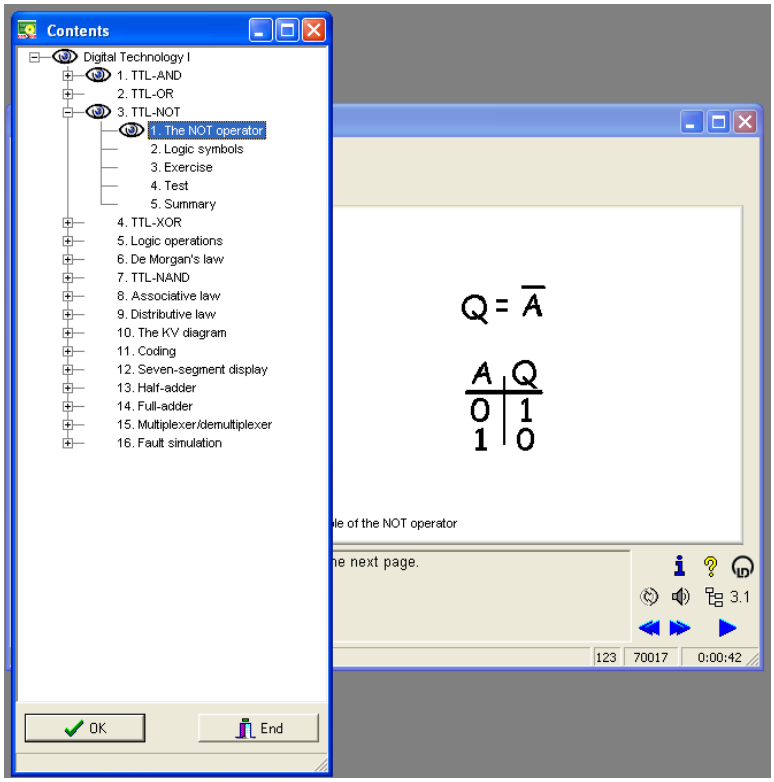


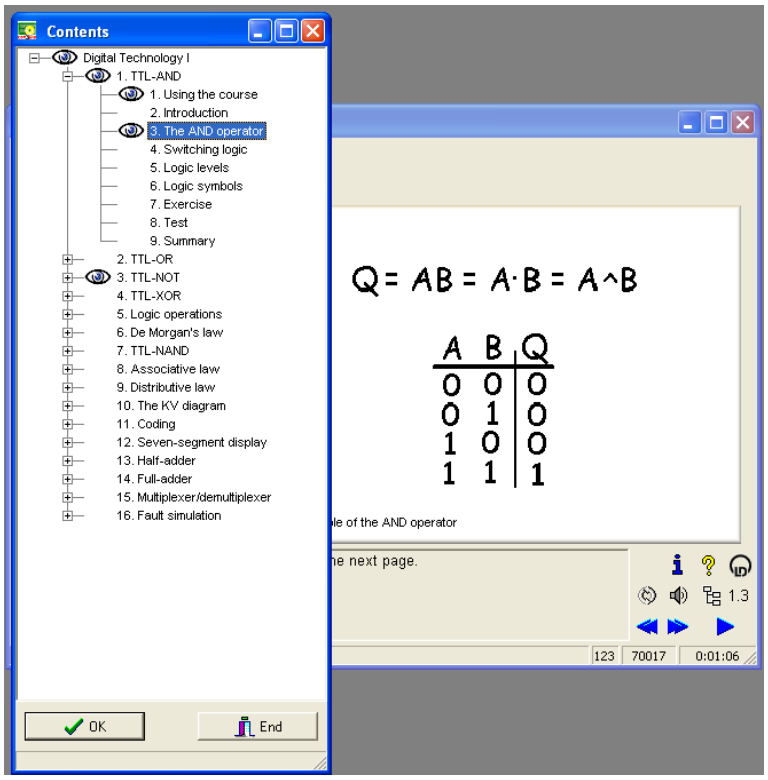
Figura 14 – Circuit cu 4 porţi logice NAND (puse în paralel) cu 2 intrări pe 1 bit şi o ieşire pe 1 bit

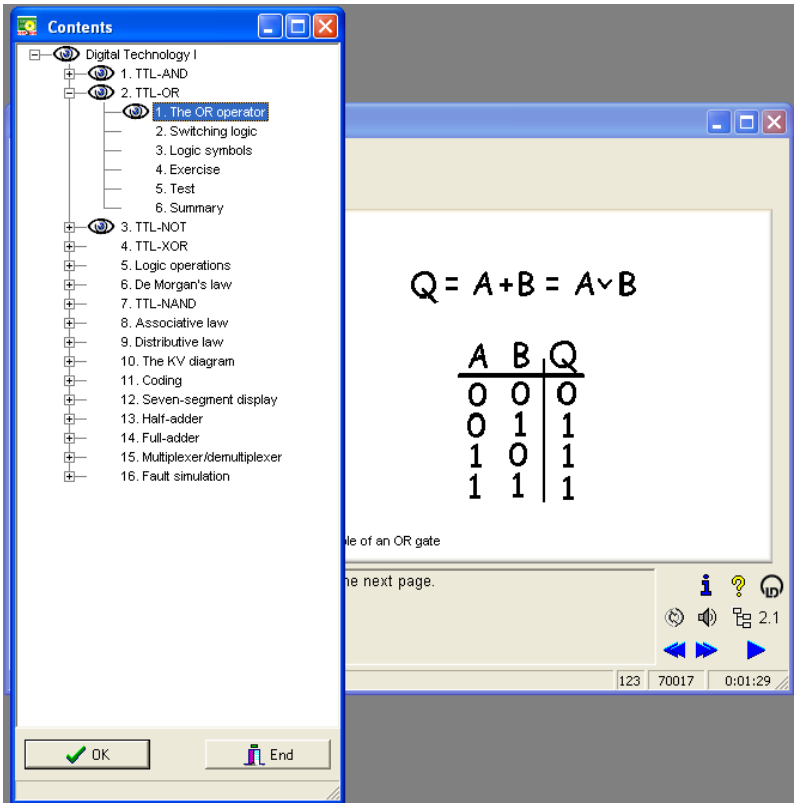
Circuit logic cu poarta NAND cu 2 intrări pe 4 biţi şi o ieşire pe 4 biţi		
x_1	x_2	y
0000	0000	
0001	0001	
1001	0110	
1010	0101	
1110	1110	

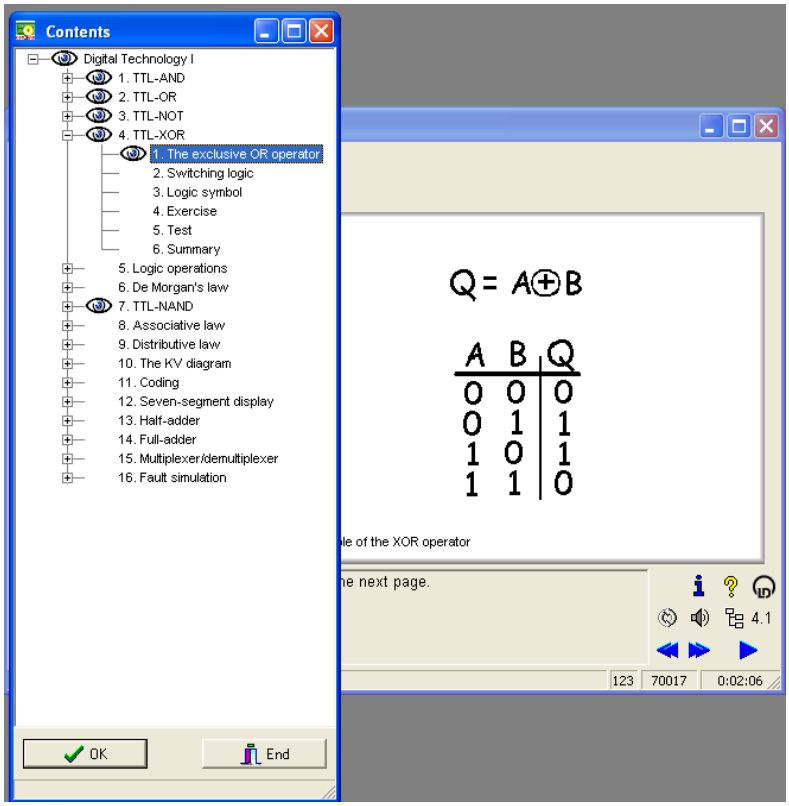
Figura 15 - Circuit logic cu poarta NAND cu 2 intrări pe 4 biţi şi o ieşire pe 4 biţi

1.4 Efectuarea laboratorului în Com3lab






Contents
Digital Technology I
1. TTL-AND
2. TTL-OR
1. The OR operator
2. Switching logic
3. Logic symbols
4. Exercise
5. Test
6. Summary
3. TTL-NOT
4. TTL-XOR
5. Logic operations
6. De Morgan's law
7. TTL-NAND
8. Associative law
9. Distributive law
10. The KV diagram
11. Coding
12. Seven-segment display
13. Half-adder
14. Full-adder
15. Multiplexer/demultiplexer
16. Fault simulation
OK
End
Q = A+B = A∨B
A B Q
0 0 0
0 1 1
1 0 1
1 1 1
le of an OR gate
ne next page.
2.1
123
70017
0:01:29


Contents
Digital Technology I
1. TTL-AND
2. TTL-OR
3. TTL-NOT
4. TTL-XOR
1. The exclusive OR operator
2. Switching logic
3. Logic symbol
4. Exercise
5. Test
6. Summary
5. Logic operations
6. De Morgan's law
7. TTL-NAND
8. Associative law
9. Distributive law
10. The KV diagram
11. Coding
12. Seven-segment display
13. Half-adder
14. Full-adder
15. Multiplexer/demultiplexer
16. Fault simulation
OK
End
Q = A⊕B
A B Q
0 0 0
0 1 1
1 0 1
1 1 0
le of the XOR operator
ne next page.
4.1
123
70017
0:02:06

Nume şi prenume	Grupa	Anul de studiu

Laboratorul 1
Raport de laborator

1.3.1 Porţi logice fundamentale
Exerciţiul 1.

Poarta NOT (NU)	
x	Y
0	1
1	0

Poarta AND (ŞI)		
x_1	x_2	Y
0	0	
0	1	
1	0	
1	1	

Poarta OR (SAU)		
x_1	x_2	Y
0	0	
0	1	
1	0	
1	1	

Poarta NAND (ŞI NU)		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta NOR (SAU NU)		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta XOR (EXCLUSIV OR)		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

Poarta XNOR (SAU NU EXCLUSIV)		
x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

1.3.2 Poarta NAND cu două intrări pe 4 biţi şi o ieşire pe 4 biţi
Exerciţiul 1.

Circuit logic cu poarta NAND cu 2 intrări pe 4 biţi şi o ieşire pe 4 biţi		
x_1	x_2	y
0000	0000	
0001	0001	
1001	0110	
1010	0101	
1110	1110	

LAB
2

Implementarea circuitelor logice combinaționale:

multiplexorul, decodificatorul și convertorul

2.1 Obiectivele laboratorului

În acest laborator se va defini funcționarea multiplexorului, decodificatorului și convertorului și se vor evidenția conceptele de ierarhizare și modularitate. Se va implementa un multiplexor 2 la 1 pe 1 bit ca modul simplu, un multiplexor 4 la 1 pe 1 bit ca modul ierarhizat, un multiplexor 2 la 1 pe 4 biti, un decodificator 4 la 2 și un convertor binar pentru un afișor un șapte segmente.

2.2 Definiții

2.2.1 Multiplexorul

Multiplexorul poate avea N intrări de date pe M biți, cu $N \geq 2$ și $M \geq 1$, o intrare de selecție pe $\log_2 N$ biți și o ieșire de date pe M biți. În funcție de valoarea intrării de selecție, ieșirea de date preia datele de la intrarea selectată. În figura 1 este evidențiat simbolul unui multiplexor cu două intrări, o ieșire și o intrare se selecție ($\log_2 2 = 1$), circuit logic denumit și multiplexor doi la unu pe un bit.

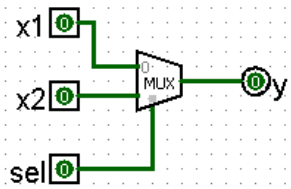


Figura 1 – Multiplexor 2 la 1 pe 1 bit.

2.2.2 Decodificatorul

Decodificatorul este un circuit care transformă o intrare pe N biți, într-o ieșire pe $M = 2^N$ biți, cu $N \geq 1$. În figura 2 este evidențiat simbolul unui decodificator 2 la 4. Decodificatorul va interpreta datele de la intrare și va oferi la ieșire semnale pentru care numai un bit va fi activ. Indexul bitului activ va fi egal cu valoarea semnalului de intrare.

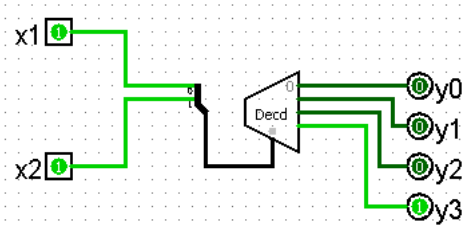


Figura 2 – Decodificator 2 la 4.

2.2.3 Convertorul

Convertorul este un circuit care transformă o intrare pe N biți, cu $N \geq 1$, într-o ieșire pe M biți, cu $M \geq 1$.

2.3 Efectuarea laboratorului

2.3.1 Implementarea multiplexorului 2 la 1 pe 1 bit

Mai jos avem funcția logică a multiplexorului 2 la 1 pe 1 bit

$$y = (x_1 \text{AND sel}) \text{OR} (x_2 \text{AND (NOT sel)})$$

În acest moment acestă funcție logică se va implementa în Logisim ca un modul simplu. Mai jos se va evidenția crearea un modul simplu.

Crearea unui modul simplu

Pasul 1. Se va da click dreapta pe folderul Untitled, care se găsește în partea stângă sus a figurii 3, și se va selecta Add Circuit … .

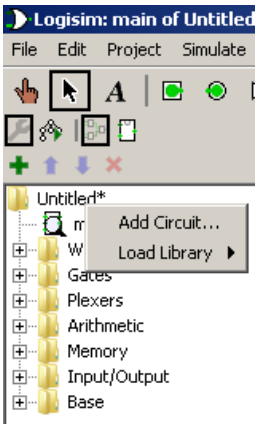


Figura 3 – Crearea unui modul simplu.

Pasul 2. În acest moment va apărea o fereastră, care ne va cere să denumim modulul simplu, Circuit Name:, ca în figura 4. Circuitul va fi denumit mux_2la1_1bit și se va da OK.

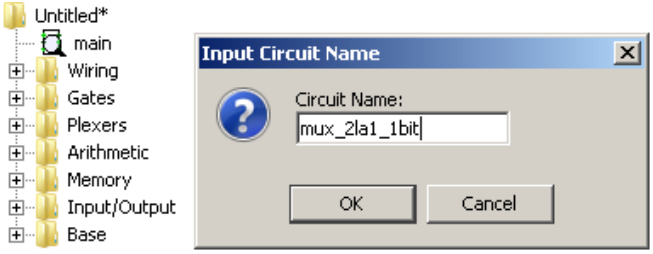


Figura 4 – Asignarea unui nume pentru un modul simplu.

Pasul 3. În acest moment Logisim a creat un modul simplu gol. Acest modul îl putem găsi în stânga sus, sub modulul main, ca în figura 5. În acest momentul ne aflăm în modulul mux_2la1_1bit. La un moment dat putem vedea în ce modul ne aflăm, observând lupa de pe un anumit modul. În cazul nostru lupa se află pe modulul mux_2la1_1bit, deci ne aflăm în modulul creat cu puțin timp în urmă. Dacă dorim să trecem în alt modul, vom da dublu click pe acel modul iar dacă dorim să ne întoarcem în modulul anterior, vom da dublu click pe modulul anterior.

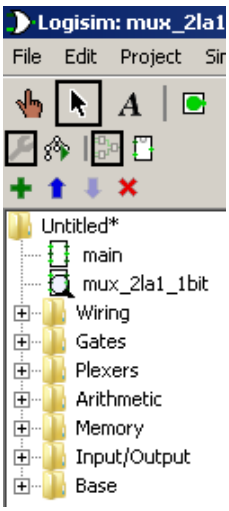


Figura 5 – Am creat un modul simplu.

Pasul 4. Se va crea în Logisim funcția logică a multiplexorului 2 la 1 pe 1 bit ca în figura 6.

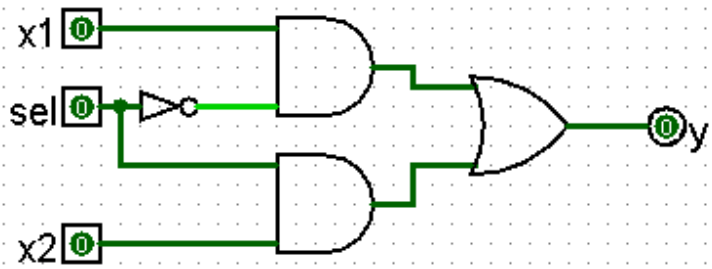


Figura 6 – Crearea circuitului intern al modulului simplu.

Utilizarea unui modul simplu

Pasul 1. După ce am creat modulul simplu se va trece în modulul main, dând dublu click pe el, și se va observa că lupa este acum pe modulul main. Acest lucru este evidențiat în figura 7.

Figura 7 – Acum ne aflăm în modulul main.

Pasul 2. Putem utiliza modulul simplu, mux_2la1_1bit, dând click pe numele lui, după care mai dăm un click în pagina de lucru, adică în modulul main. După ce am executat pasul doi, ar trebui să vedem același lucru ca în figura 8.

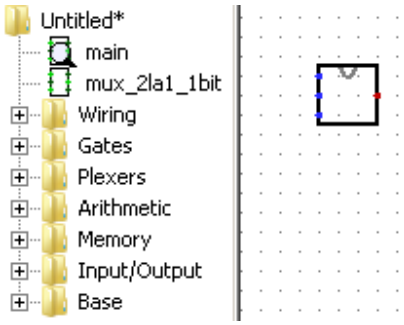


Figura 8 – Plasarea în modulul main a modulului simplu mux_2la1_1bit.

Pasul 3. Se vor conecta semnale la intrările și ieșirea modulului mux_2la1_1bit. După ce am executat pasul trei, ar trebui să vedem același lucru ca în figura 9.

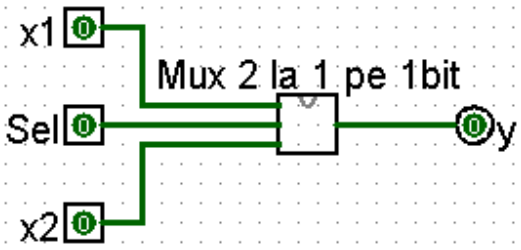


Figura 9 – Conectarea modulului simplu mux_2la1_1bit cu intrări și ieșire.

Execitiul 1.

Să se completeze tabelul de adevăr pentru multiplexorul 2 la 1 pe 1 bit din figura 9. Tabelul de adevăr este reprezentat mai jos

Tabel 1 – Tabel de adevar pentru multiplexorul 2 la 1 pe 1 bit din figura 9.

Sel	x1	x2	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Atunci când intrarea Sel = 0, ce intrare va selecta circuitul, adică y va fi egal cu? Atunci când intrarea Sel = 1, ce intrare va selecta circuitul, adică y va fi egal cu?

2.3.2 Implementarea multiplexorului 4 la 1 pe 1 bit

Multiplexorul 4 la 1 pe 1 bit se va implementa cu ajutorul modulului creat anterior, modulul simplu mux_2la1_1bit. Pentru a crea un multiplexor 4 la 1 pe 1bit mai întâi creăm un modul cu numele mux_4la1_bit. În interiorul acestui modul instanțiem 3 module simple mux_2la1_1bit, adică dăm un click pe numele modulului mux_2la1_1bit, după care ne poziționăm în fereastra de lucru și mai dăm un click. Vom repeta acest lucru de încă două ori. După ce am plasat 3 module mux_2la1_1bit circuitul parțial ar trebui să arate ca în figura 10.

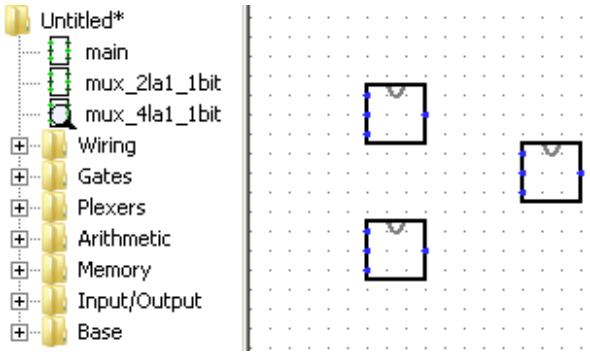


Figura 10 – Instantierea a trei module mux_2la1_1bit pentru a crea un multiplexor 4 la 1 pe 1 bit ca modul ierarhizat.

Multiplexorul 4 la 1 pe 1 bit va fi implementat ca modul ierarhizat. Modulul ierarhizat este alcătuit din alte module, care la randul lor sunt alcatuite din alte module, care la randul lor sunt alcatuite cu porti logice fundamentale. Deci acest circuit este creat utilizând conceptul de ierarhizare.

Legăturile în interiorul multiplexorului 4 la 1 pe 1 bit vor arăta ca în figura 11. Avem 4 intrări pe 1 bit, o intrare de selecție pe 2 biți, pentru care am folosit un splitter, și o ieșire pe 1 bit.

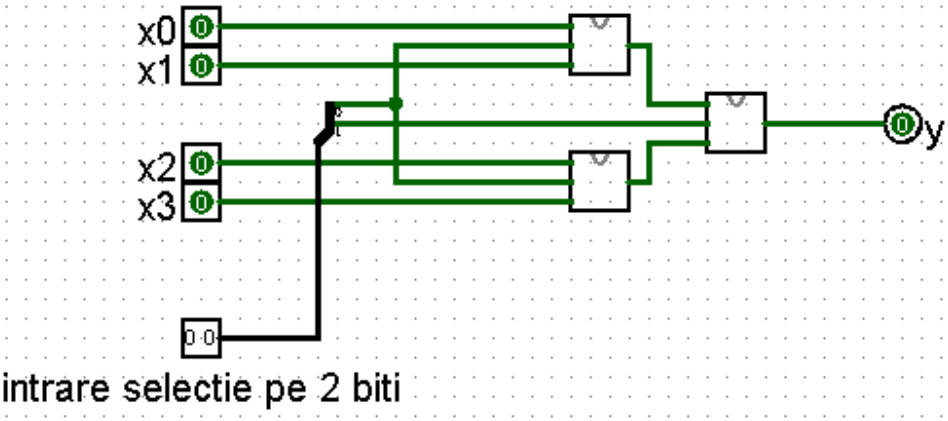


Figura 11 – Multiplexor 4 la 1 pe 1 bit, creat cu trei module simple mux_2la1_1bit.

Exercițiul 1.

În modulul main, se va instanția modulul ierarhizat mux_4la1_1bit, se vor face conexiunile ca în figura 12 și se va completa tabelul de adevăr. Atunci când Sel = 01 ce intrare va fi selectată, atunci când Sel = 10 ce intrare va fi selectată, atunci când Sel = 11 ce intrare va fi selectată?

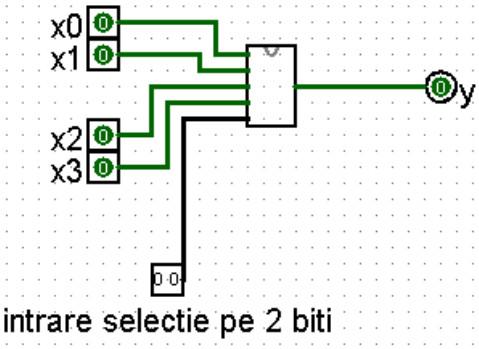


Figura 12 – Multiplexorul 4 la 1 ca modul ierarhizat, instanțiat în modulul main.

Tabel 2 – Tabel de adevar pentru multiplexor 4 la 1 pe 1 bit din figura 12.

Sel	y = ?
00	y = x0
01	
10	
11	

2.3.3 Multiplexorului 2 la 1 pe 4 bit

Multiplexorul 2 la 1 pe 4 biți se va crea în modulul main, prin instanțierea a 4 module simple mux_2la1_1bit. Avem nevoie de 4 biți pentru o intrare, deci vom avem nevoie de 4 module simple. Instanțierea modulelor simple este evidențiată în figura 13.

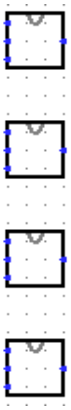


Figura 13 – Instanțierea a 4 module simple mux_2la1_1bit, în modulul main, pentru a crea un multiplexor 2 la 1 pe 4 biți.

Pentru a crea multiplexorul 2 la 1 pe 4 biți, am utilizat 4 module mux_2la1_1bit. Modulul mux_2la1_1bit a fost creat anterior.

Exercițiul 1.

Să se execute legăturile între cele două intrări pe 4 biți, intrarea de selecție pe 1 bit, module și ieșirea pe 4 biți. Vom avea o intrare de selecție pe 1 bit deoarece avem 2 intrări. Intrarea de selecție se modifică în funcție de numărul de intrări și nu de numărul de biți de pe o anumită intrare. Deci dacă aveam 8 intrări pe 32 de biți, aveam nevoie de $\log_2 8 = 3$ biți de selecție.

Ajutor pentru exercițiul 1. Se vor folosi două intrări și o ieșire pe 4 biți. Intrările se vor ramifica spre intrările celor 4 module mux_2la1_1bit cu ajutorul splitterelor. Ieșirile celor 4 module mux_2la_1_1bit se vor strânge cu ajutorul unui splitter și se va conecta la ieșirea pe 4 biți. Vom avea nevoie de un splitter pentru fiecare intrare (nu de un splitter pentru fiecare bit al intrării) și un splitter pentru ieșire, **în total 3 splittere**. Atenție la ordinea intrărilor în module și ordinea firelor care ies sau intră în splitter.

În concluzie utilizând conceptul de regularitate am putut crea circuite mai complexe utilizând numai un modul pe care l-am creat o singură dată și pe care l-am instanțiat de mai multe ori, de câte ori am avut nevoie, fără să fie necesar să în recreăm. Aceasta este puterea regularității.

2.3.4 Decodificatorul 2 la 4

Exercițiul 1.

Din librăria Plexers se va alege decoder. În partea stângă a ecranului, Select Bits se va alege 2, adică 2 biți de intrare, iar Include Enable? se va alege No. Se vor face conexiunile și se va completa tabelul de adevăr. Tabelul de adevăr va avea două intrări pe 1 bit (sau o intrare pe 2 biți) și patru ieșiri pe 1 biți (sau o ieșire pe 4 biți).

Tabel 3 – Tabel de adevăr pentru un decodificator 2 la 4.

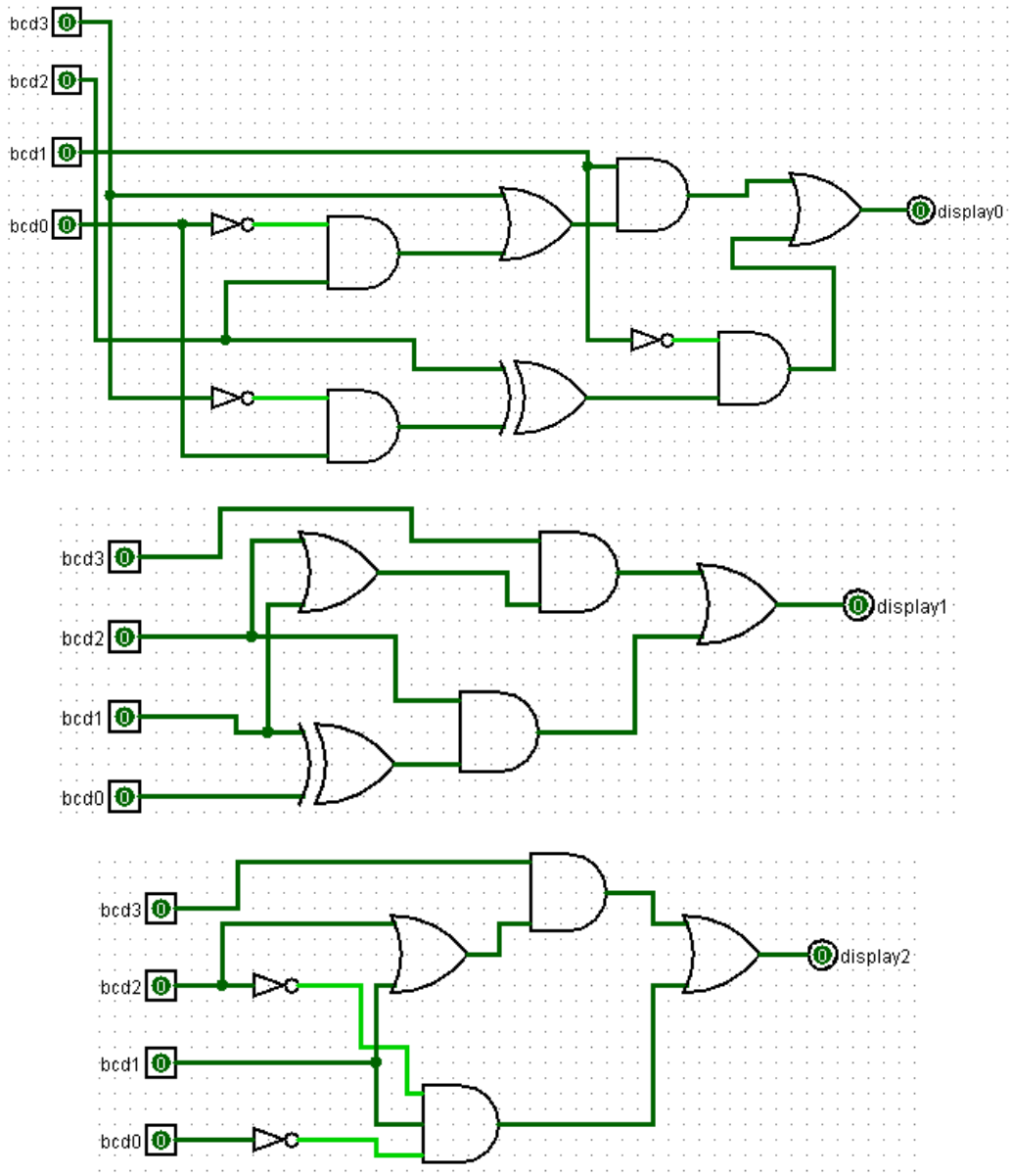
Nr. în baza 10	Intrare pe 2 biți		Ieșire pe 4 biți			
0	0	0				
1	0	1				
2	1	0				
3	1	1				

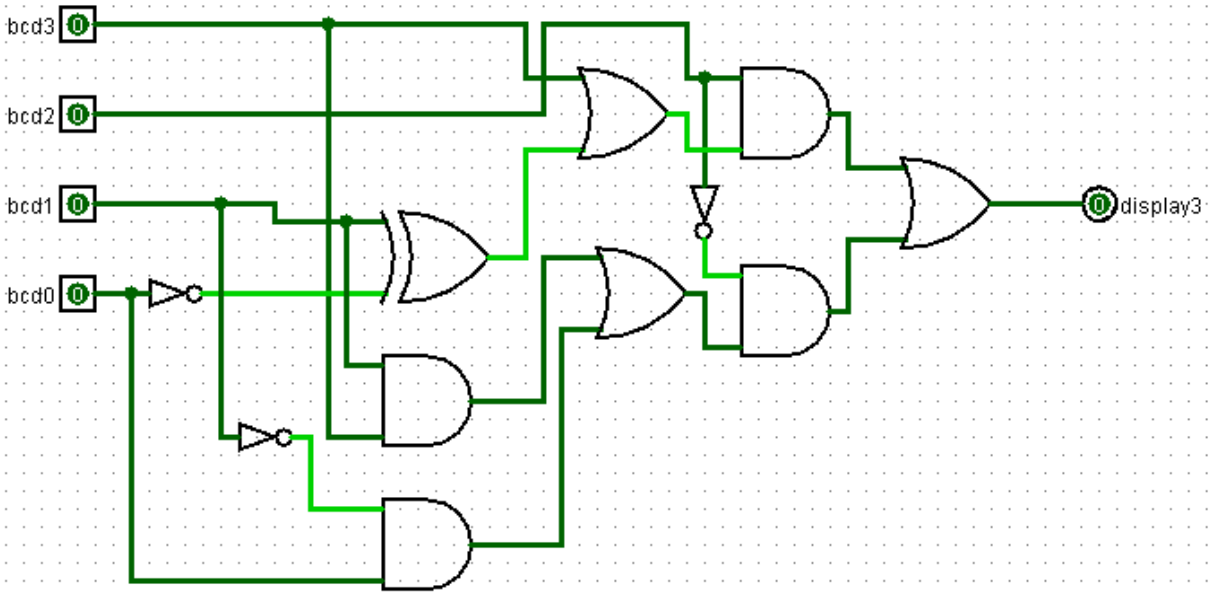
2.3.5 Convertorul binar zecimal

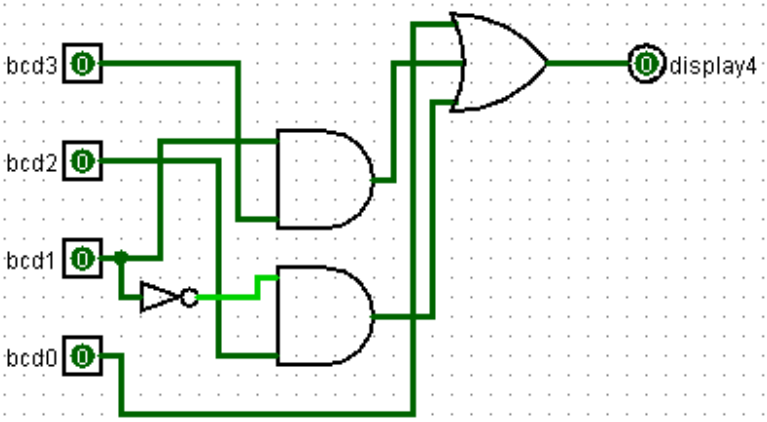
Exercițiul 1.

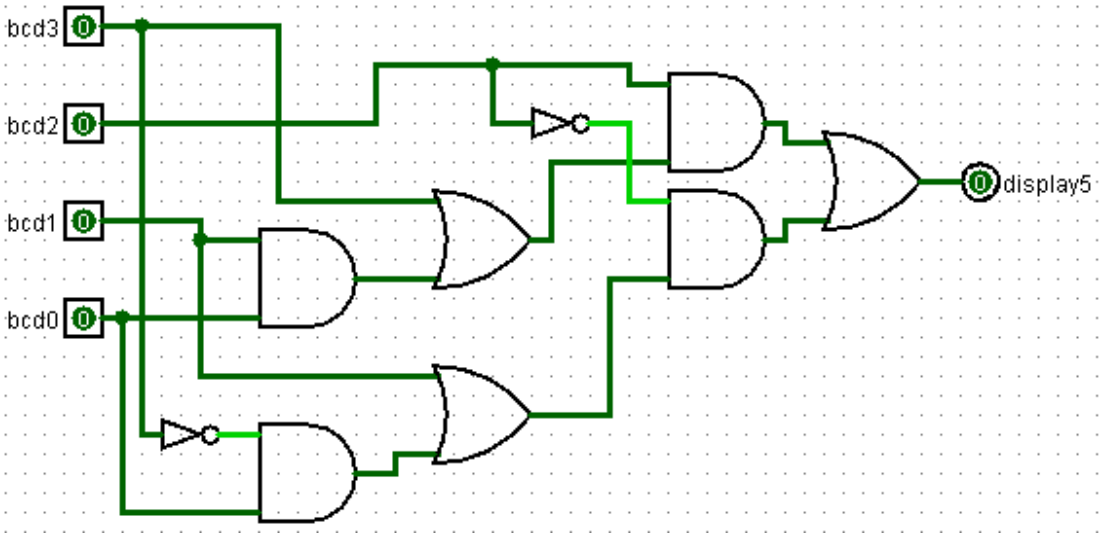
Să se execute în Logisim, circuitul din figura 14. Se vor face 7 module simple, denumite $y_0, \dots, y_6$, pentru fiecare ieșire. Aceste module vor aprinde un afișor cu șapte segmente. Aceste module vor face conversia binar-zecimală.

Ajutor pentru exercițiul 1. Cele 7 module se vor instanția în modulul main, ca în figura 15. Se va introduce o intrare pe 4 biți, un splitter cu 4 fire și se vor interconecta cu intrările în cele 7 module. Ieșirile modulelor se vor lega la afișorul cu șapte segmente, după ordinea evidențiată în figura 16.




bcd3
bcd2
bcd1
bcd0
display3


bcd3
bcd2
bcd1
bcd0
display4


bcd3
bcd2
bcd1
bcd0
display5

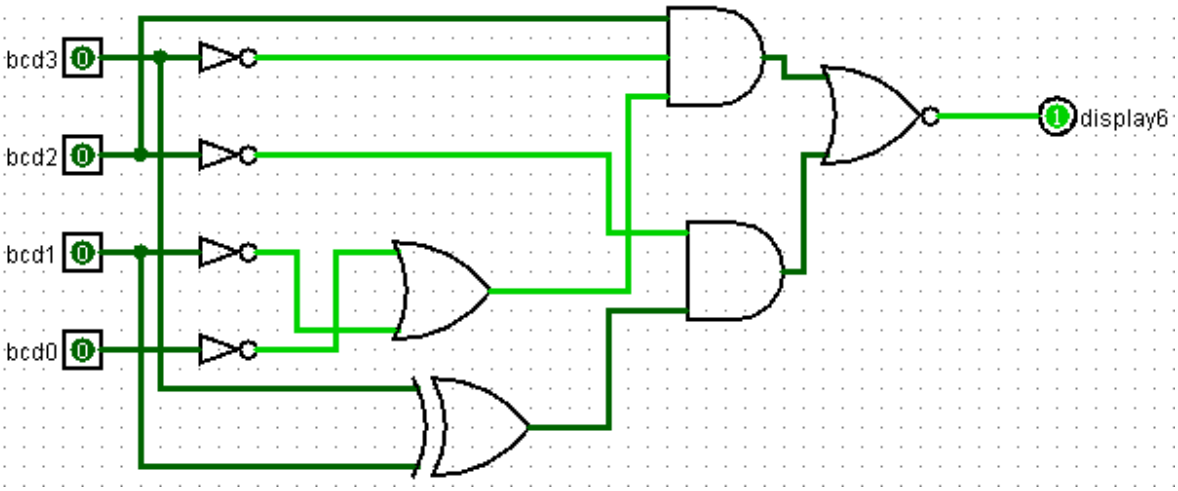


Figura 14 – Cele 7 module ale convertorului binar zecimal.

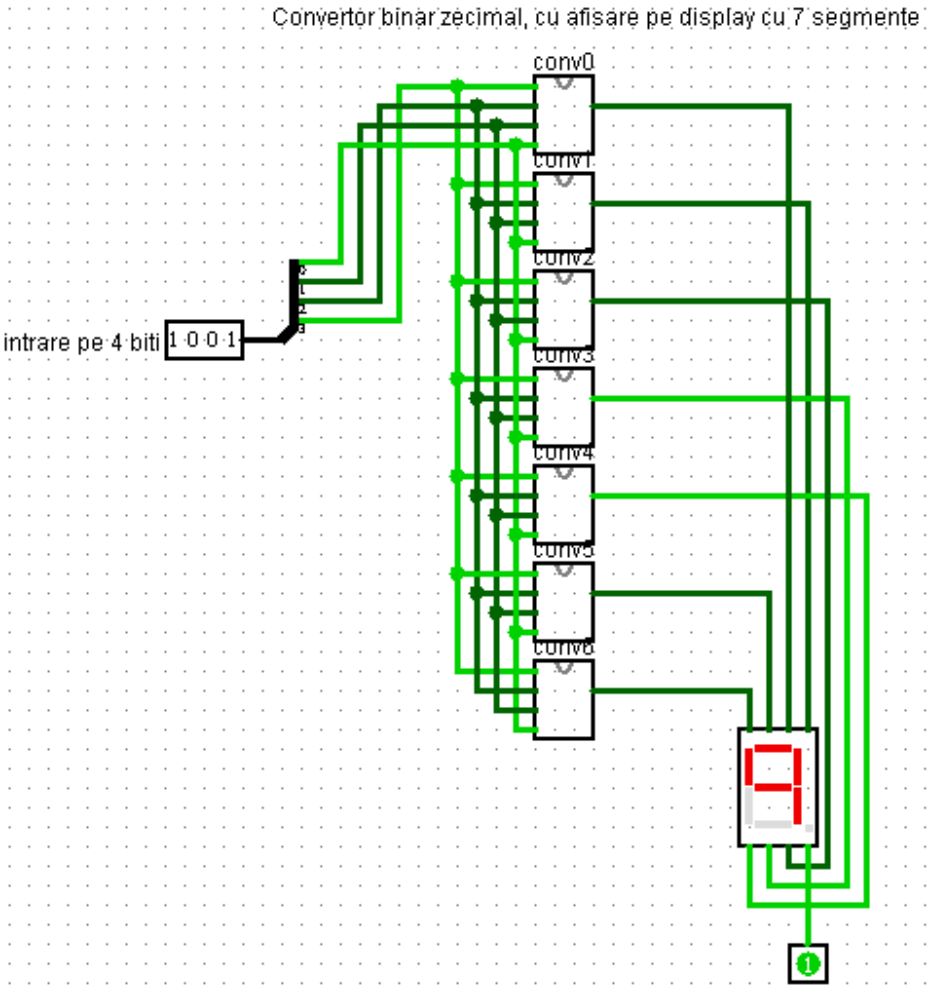


Figura 15 – Instanțierea celor 7 module și interconectarea lor.

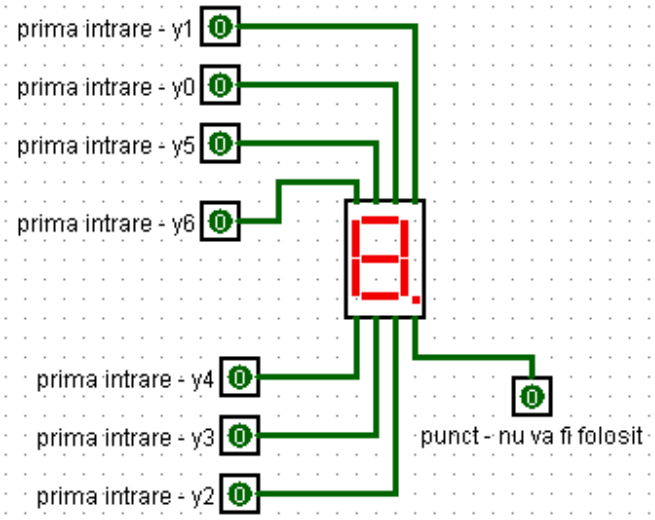
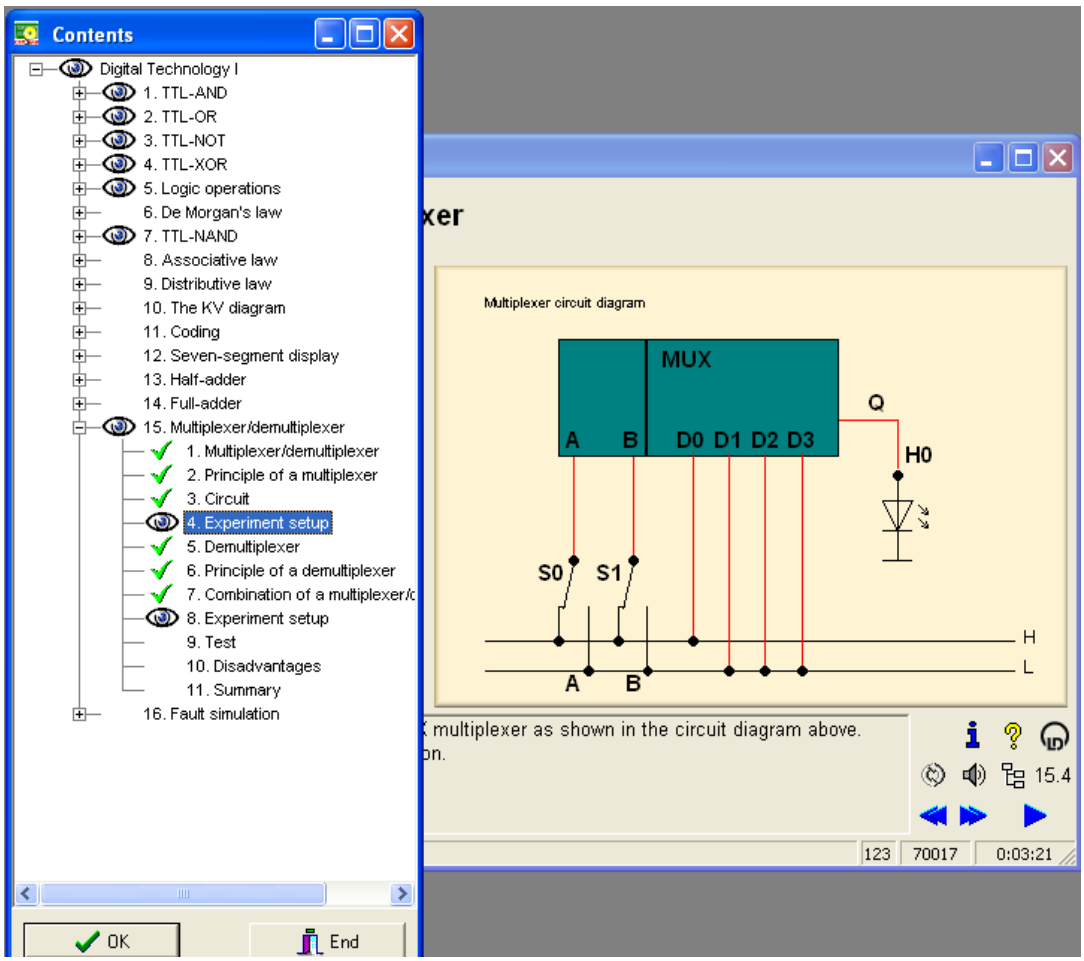


Figura 16 – Ordinea intrărilor în afișorul cu 7 segmente.

2.4 Efectuarea laboratorului în COM3LAB



Nume şi prenume	Grupa	Anul de studiu

Laboratorul 2
Raport de laborator

2.3.1 Implementarea multiplexorului 2 la 1 pe 1 bit
Exerciţiul 1.

Tabel 1 – Tabel de adevar pentru multiplexorul 2 la 1 pe 1 bit.

Sel	x1	x2	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Atunci când intrarea Sel = 0, ce intrare va selecta circuitul, adică y va fi egal cu? Atunci când intrarea Sel = 1, ce intrare va selecta circuitul, adică y va fi egal cu?

2.3.2 Implementarea multiplexorului 4 la 1 pe 1 bit
Exerciţiul 1.

Tabel 2 – Tabel de adevar pentru multiplexor 4 la 1 pe 1 bit.

Sel	y = ?
00	x0
01	
10	
11	

Atunci când Sel = 01 ce intrare va fi selectată, atunci când Sel = 10 ce intrare va fi selectată, atunci când Sel = 11 ce intrare va fi selectată?

2.3.4 Decodificatorul 2 la 4
Exerciţiul 1.

Tabel 3 – Tabel de adevăr pentru un decodificator 2 la 4.

Intrarea reprezentată în baza 10	Intrare pe 2 biţi		Ieşire pe 4 biţi			
0	0	0				
1	0	1				
2	1	0				
3	1	1				

Pentru o anumită ieşire care din biţi sunt 1 şi ce legătură există între aceşti biţi şi intrare?

LAB
3

Optimizarea circuitelor logice combinaționale

3.1 Obiectivele laboratorului

În acest laborator se va defini triunghiul magic și se va exemplifica utilizarea programului Logisim pentru optimizarea circuitelor logice combinaționale.

3.2 Definiții

3.2.1 Triunghiul magic

Avem trei concepte de bază: tabelul de adevăr, funcția logică și circuitul logic. Aceste trei concepte formează triunghiul magic, reprezentat în figura 1. Dacă la un moment dat avem unul din elementele triunghiului magic imediat putem să obținem celelate două elemente.

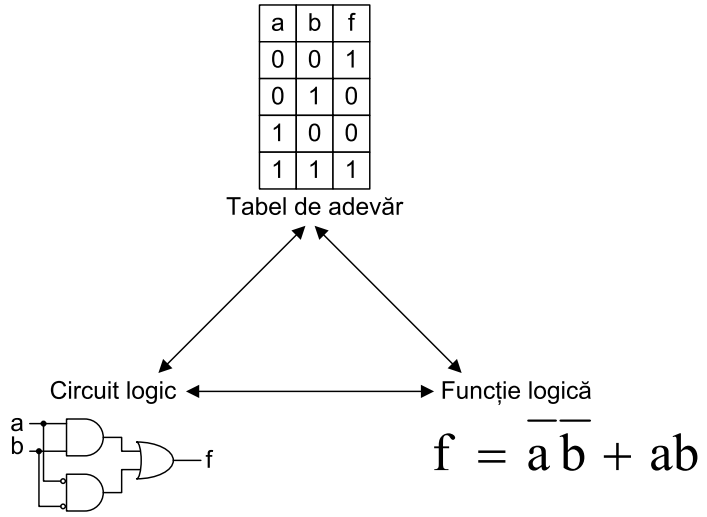


Figura 1 – Triunghiul magic, tabel de adevăr, funcție logică și circuit logic.

3.3 Efectuarea laboratorului

3.3.1 Implementarea și optimizarea circuitelor logice combinaționale

Avem următorul tabel de adevăr, x1 și x2 reprezintă intrări iar y0, y1, y2, y3, y4, y5, y6 reprezintă ieșiri.

Tabel 1 – table de adevăr pentru un convertor 2 la 7.

x1	x2	y0	y1	y2	y3	y4	y5	y6
0	0	0	0	0	1	1	1	0
0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1
1	1	0	1	1	0	0	0	0

Tabelul 1 este tabelul de adevăr pentru un convertor 2 la 7. Acest convertor va aprinde ledurile unui afișor cu 7 segmente și în funcție de combinațiile la intrare, afișorul va reprezenta anumite caractere. Astfel că, pentru combinația la intrare 00, pe afișor va apărea litera L, pentru 01 pe afișor va apărea litera A, pentru 10 pe afișor va apărea litera B, iar pentru 11 pe afișor va apărea valoarea 1. În continuare vom exemplifica modul în care se implementează și optimizează tabelul de adevăr de mai sus cu ajutorul programului Logisim.

Implementarea și optimizarea circuitelor logice în Logisim

Pasul 1. Se deschide programul Logisim, dând dublu click pe pictograma lui.

Pasul 2. Din meniul pop-up se alege Window și apoi se va da click pe Combinational Analysis. În acest moment trebuie să apară o fereastra cu fundal gri ca în figura 2.

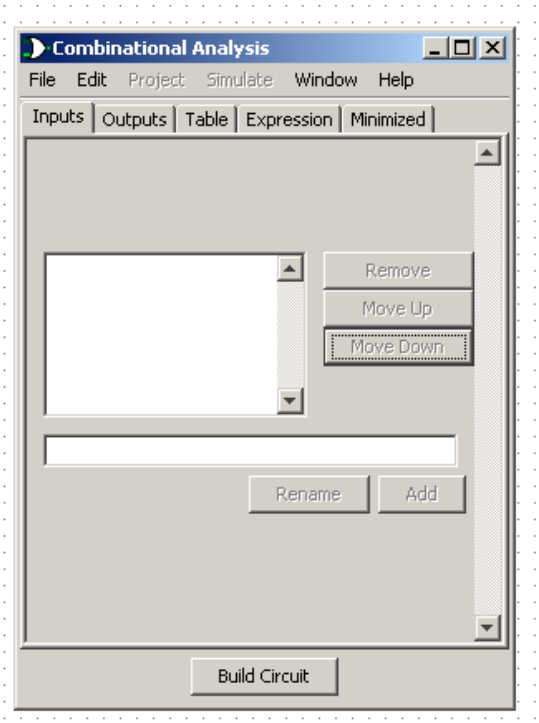


Figura 2 – Fereastra Combinational Analysis cu ajutorul căreia vom putea optimiza funcția din tabelul 1.

Pasul 3. În acest moment se vor introduce intrările x1 și x2 ca în figura 3. Intrările sunt introduse una câte una în fereastra lungă, din tabul Inputs, prin acționarea butonului Add.

Pasul 4. În acest moment se vor introduce ieșirile y0, y1, y2, y3, y4, y5, y6 ca în figura 4. Ieșirile sunt introduse una câte una în fereastra lungă, din tabul Outputs, prin acționarea butonului Add.

Pasul 5. În acest moment se va defini conținutul tabelului de adevăr. În figura 5 este evidențiat tabelul de adevăr, din tabul Table, din fereastra Combinational Analysis. Acele x-uri înseamnă valori nealocate. Noi vom introduce în tabelul de adevăr, din tabul Table, pentru fiecare

combinație la intrare valorile ieșirilor din tabelul 1. După ce am făcut acest lucru, tabelul completat ar trebui să arate ca cel din figura 6.

Pasul 6. Vom apăsa pe buton Build Circuit pentru ca programul să genereze circuitul logic combinațional optimizat. În acest moment apare fereastra din figura 7. În tabul Circuit Name vom introduce un nume sugestiv pentru circuitul creat. Programul va crea un modul cu acest nume în care va plasa circuitul creat. Se va selecta Use two-input gates only și se va la Ok.

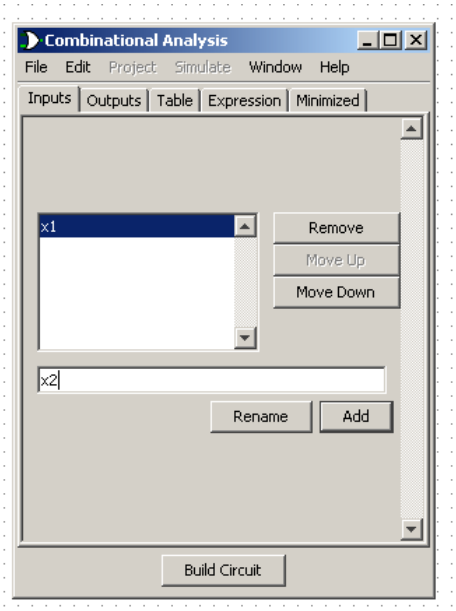


Figura 3 – Introducerea intrărilor x1 și x2 din tabelul de adevăr.

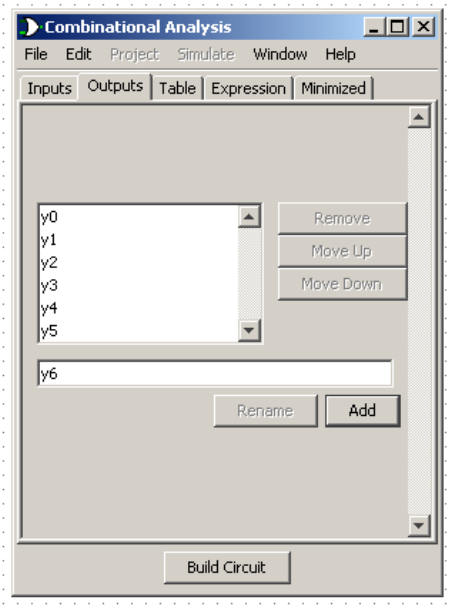


Figura 4 – Introducerea ieșirilor y0, … y6 din tabelul de adevăr.

Pasul 7. În fereastra Combinational Analysis se va da click pe tabul Expression. În acest tab sunt reprezentate funcțiile logice pentru fiecare ieșire. Acest lucru este evidențiat în figura 8. Apoi se va da click pe tabul Minimized pentru a se observa tabelele grafice Karnaugh pentru fiecare ieșire. Acest lucru este evidențiat în figura 9.

Exercițiul 1.

Să se optimize următorul tabel de adevăr și să se explice funcționarea circuitului logic combinațional. Tabelul de adevăr determină funcționarea unui convertor 2 la 7, ieșirile căruia se vor conecta la un afișor cu șapte segmente. Ce litere sunt reprezentate pe afișor?

Tabel 2 – Tabel de adevăr pentru exercițiul 1.

Sel0	Sel1	y0	y1	y2	y3	y4	y5	y6
0	0	0	0	0	1	1	1	0
0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	0	0	1

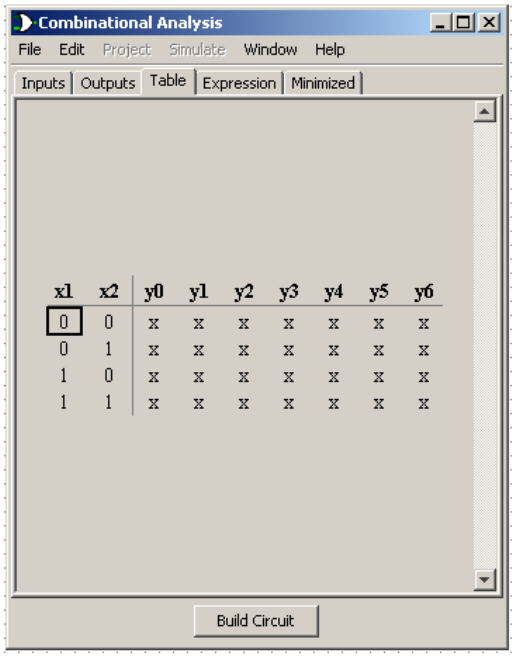


Figura 5 – Tabel de adevăr necompletat.

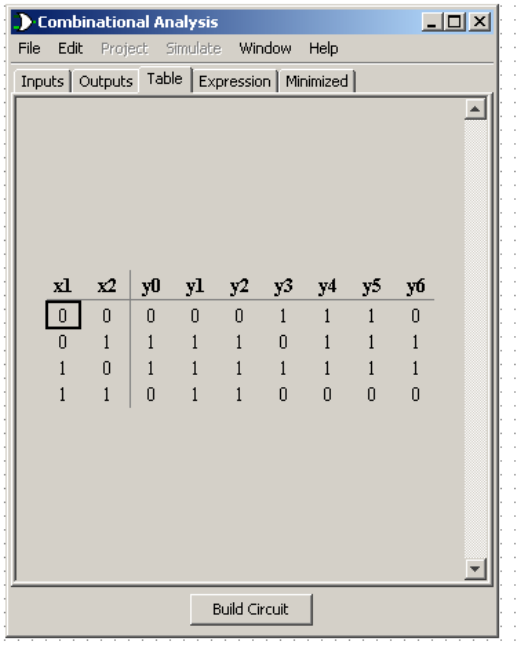


Figura 6 – Tabel de adevăr completat cu valorile ieșirilor din tabelul 1.

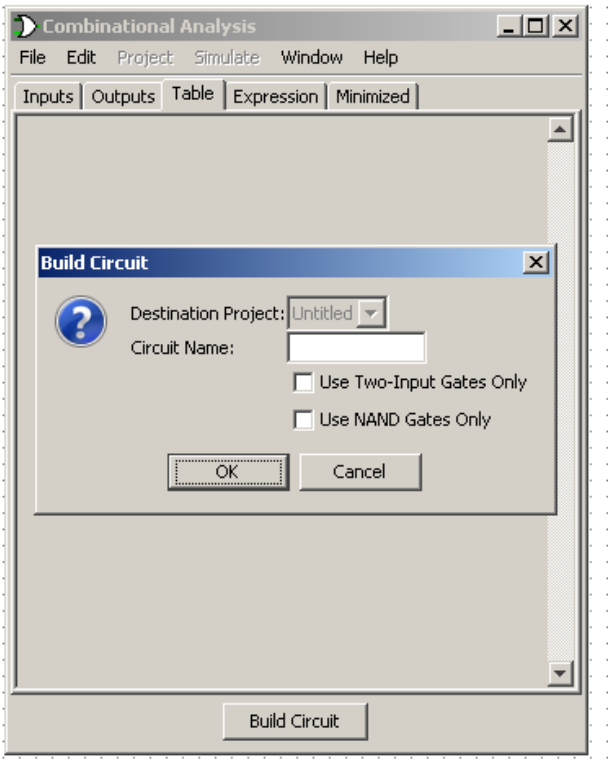


Figura 7 – Fereastră cu proprietăți ce apare după apăsarea butonului Build Circuit.

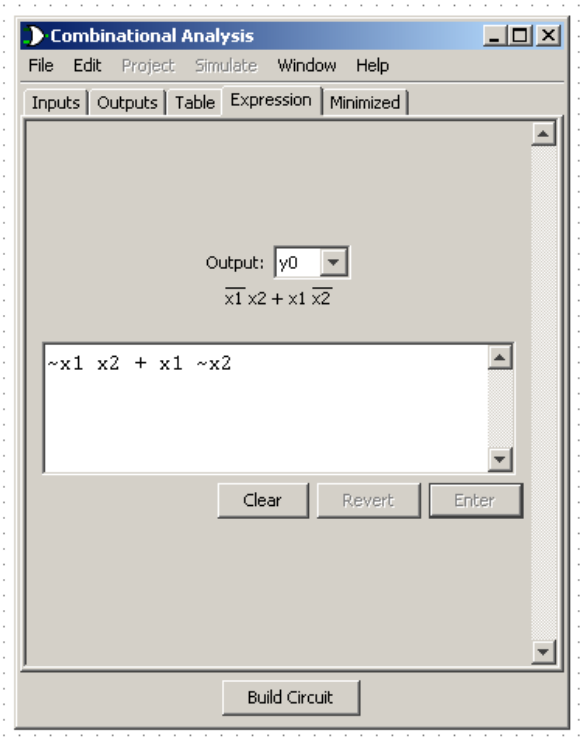


Figura 8 – Tabul Expression unde se găsesc funcțiile logice pentru fiecare ieșire.

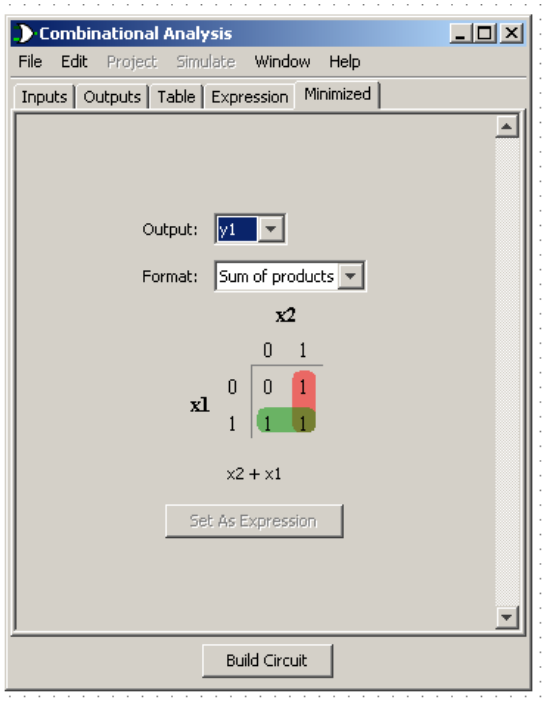


Figura 9 – Tabelul graphic Karnaugh pentru ieșirea y1.

Exercițiul 2.

Utilizând convertorul creat în cadrul exercițiului 1, se va construi circuitul din figura 10. Circuitul Selecție Rotire se va crea utilizând metoda de implementare și optimizare, Combination Analysis, a programului Logisim. Acest circuit este un convertor 2 la 8. Tabelul de adevăr al circuitului Selecție Rotire este evidențiat mai jos.

După ce circuitul va fi creat, ca în figura 10, se va completa tabelul de adevăr, Tabel 4, ținând cont de ordinea biților la intrare, adică 00, 01, 11, 10.

Tabel 3 – Tabelul de adevăr pentru circuitul Selectie Rotire.

Selecție Rotire		Conv1		Conv2		Conv3		Conv4	
Sel0	Sel1	I1	I2	I1	I2	I1	I2	I1	I2
0	0	0	0	0	1	1	0	1	1
0	1	1	1	0	0	0	1	1	0
1	0	0	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0	0	1

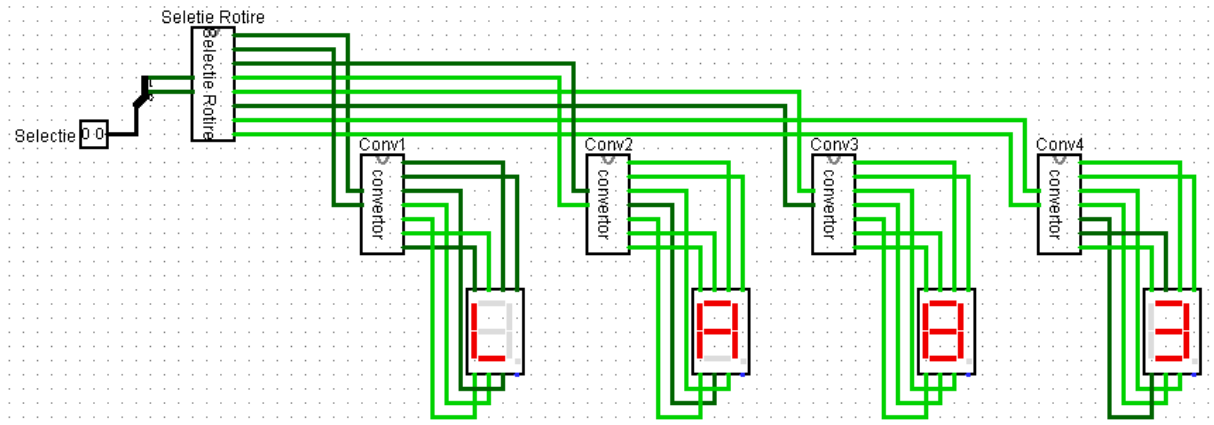


Figura 10 – Circuitul logic pentru exercițiul 2.

Ajutor pentru exercițiul 2. Se observă în figura 10 că avem o intrare de selecție pe 2 biți, un spliter, un modul Selecție Rotire, patru convertoare și patru afișoare cu 7 segmente. Splitterul va fi configurat ca în figura 11.

Selection: Splitter	
Facing	East
Fan Out	2
Bit Width In	2
Appearance	Left-handed
Bit 0	1 (Bottom)
Bit 1	0 (Top)

Figura 11 – Configurarea splitterului din figura 10.

Tabel 4 – Tabelul de adevăr pentru circuitul din figura 10. Se va ține cont de ordinea biților intrării Sel, adică 00, 01, 11, 10.

Sel	Conv1	Conv2	Conv3	Conv4
00	L	A	B	3
01				
11				
10				

Exercițiul 3.

Să se optimizeze următorul tabel de adevăr și să se explice funcționarea circuitului logic combinațional. Tabelul de adevăr determină funcționarea unui convertor 4 la 7, ieșirile căruia se vor conecta la un afișor cu șapte segmente. Ce litere sunt reprezentate pe afișor și ce cuvânt sau cuvinte formează acestea?

Tabel 5 – Tabel de adevăr pentru exercițiul 3.

x1	x2	x3	x4	y0	y1	y2	y3	y4	y5	y6
0	0	0	0	1	0	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	1	1	1	0	1	1	1
0	0	1	1	1	0	0	1	1	1	0
0	1	0	0	1	0	0	1	1	1	1
0	1	0	1	1	0	0	0	1	1	1
0	1	1	0	0	1	1	0	0	0	0
0	1	1	1	1	0	0	1	1	1	1
1	0	0	0	0	0	0	1	1	1	0
1	0	0	1	0	1	1	1	1	0	1

Ajutor pentru exercițiul 3. Convertorul va reprezenta 10 litere, adică vom aveam la intrare primele 10 combinații de biți. Restul combinațiilor din tabelul de adevăr se vor lăsa nedefinite, x.

Nume și prenume	Grupa	Anul de studiu

Laboratorul 3
Raport de laborator

3.3.1 Implementarea și optimizarea circuitelor logice combinaționale

Exercițiul 1.

Tabel 1 – Caracterele ce vor fi reprezentate pe afișorul cu șapte segmente în funcție de intrarea de selecție.

Intrarea de selecție	Caractere
00	
01	
10	
11	

Exercițiul 2.

Tabel 2 – Caracterele ce vor fi reprezentate pe cele 4 afișoare cu șapte segmente. Atenție la ordinea biților de la intrarea de selecție

Intarea de selecție	Conv1	Conv2	Conv3	Conv4
00	L	A	B	3
01				
11				
10				

Exercițiul 3.

Tabel 3 – Caracterele ce vor fi reprezentate pe afișorul cu șapte segmente în funcție de intrarea de selecție.

Intrare de selecție în baza 10	Intrarea de selecție	Caractere
0	0000	
1	0001	
2	0010	
3	0011	
4	0100	
5	0101	
6	0110	
7	1000	
8	1001	
9	1010	

LAB
4

Implementarea circuitelor logice secvențiale sincrone

4.1 Obiectivele laboratorului

În acest laborator se va defini şi utiliza elementul fundamental de memorie, numit bistabil de tip D sau flip-flop de tip D, registrul paralel, registrul serie şi registrul universal. Circuitele amintite mai sus sunt circuite logice secvențiale sincrone.

4.2 Definiții

4.2.1 Circuitul logic secvențial

Circuitul logic secvențial este un circuit logic cu **memorie**. Circuitele logice secvențiale sunt sincrone şi asincrone. Circuitele logice secvențiale includ toate acele circuite logice care nu sunt combinaționale, deoarece se ştie că circuitele logice combinaționale nu au memorie.

4.2.2 Circuitul logic secvențial sincron

Circuitul logic secvențial sincron este un circuit logic cu memorie. Acest circuit memorează datele de la intrare, sincron, cu ajutorul unui impuls de tact, pe frontul crescător al tactului. Circuitul logic secvențial sincron fundamental se numeşte bistabil sau flip-flop. În practică, cel mai utilizat flip-flop este **flip-flop-ul de tip D**. Flip-flop-ul de tip D este elementul fundamental de memorie. Simbolul acestuia este evidențiat în figura 1.

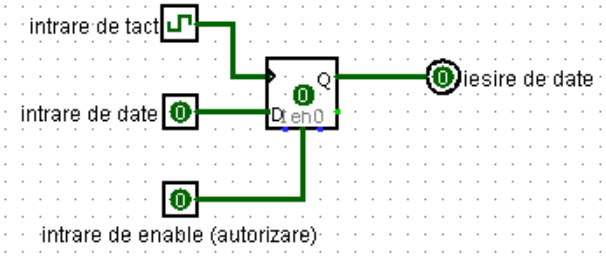


Figura 1 – Flip-flop-ul de tip D.

Cu ajutorul flip-flop-ului de tip D putem crea registre de memorie. Registrul de memorie este un circuit logic secvențial sincron. În practică există registrul paralel, registrul serie, şi registrul universal.

4.3 Efectuarea laboratorului

4.3.1 Flip-flop-ul de tip D

În figura 1 este evidențiat flip-flop-ul de tip D, elementul fundamental de memorie. Se observă că acest circuit are trei intrări. O intrare de tact, fiind intrarea de sincronizare, o intrare de date, fiind intrarea principală a circuitului logic şi o intrarea de enable.

Atunci când intrarea de enable are valoarea logică 0, flip-flop-ul de tip D nu va mai funcționa, adică nu va mai memora valorile de la intrare, dar îşi va reține valoarea anterioară.

Atunci când intrarea de enable are valoarea logică 1, flip-flop-ul de tip D va funcționa normal, adică va memora datele de la intrarea D. Memorarea datelor de către flip-flop-ul de tip D se face pe frontul crescător al semnalului de tact. Frontul crescător se definește ca trecerea de la valoarea logică 0 la valoarea logică 1 a semnalului de tact.

În Logisim trecerea din 0 în 1 este determinată de modificarea culorii semnalului de intrare. Cu ajutorul butonului care reprezintă o mână putem da click pe intrarea de tact pentru a observa modificarea culorilor și implicit trecerea din 0 în 1 a semnalului de tact. Valoarea logică 0 este reprezentată cu verde închis, iar valoarea logică 1 este reprezentată cu verde deschis. Frontul crescător (trecerea din 0 în 1) este reprezentată în figura 2.

a) **b)**

Figura 2 – Reprezentarea frontrului crescător pentru semnalul de tact. a) semnalul de tact are valoarea 0 (culoarea verde închis), b) semnalul de tact are valoarea 1 (culoarea verde deschis).

!!!Atenție. Pentru o trecere completă, care se mai numește **perioadă**, se vor da două click-uri pe simbolul semnalului de tact. Acest lucru înseamnă că semnalul de tact va trece din 0 în 1 și apoi din 1 în 0.

Memorarea datelor de către flip-flop-ul de tip D

Mai întâi se va instanția în Logisim flip-flop-ul de tip D. Flip-flop-ul de tip D se găsește în Logisim, în fereastra cu librării din partea stângă, în folderul Memory. Apoi se vor face conexiunile între intrări, ieșire și flip-flop. Fiecare intrare sau ieșire va avea un Label sau o etichetă, o denumire cum apare în figura 1. Apoi se vor executa pașii din tabelul 1. Tabelul 1 evidențiază pașii pe care trebuie să îi parcurgem în Logisim pentru a putea evidenția funcționarea flip-flop-ului de tip D.

Tabel 1 – Tabelul evidențiază simularea flip-flop-ului de tip D. Circuitul memorează date doar dacă intrarea de enable este 1 și semnalul de tact trece din 0 în 1 (frontal crescător).

Pasul 1	Intrarea de enable este 0 Circuitul nu memorează	Intrarea de date este 0	Frontul crescător	Ieșirea de date va fi 0
Pasul 2	Intrarea de enable este 0 Circuitul nu memorează	Intrarea de date este 1	Frontul crescător	Ieșirea de date va fi 0
Pasul 3	Intrarea de enable este 1 Circuitul memorează	Intrarea de date este 0	Frontul crescător	Ieșirea de date va fi 0 Circuitul memorează 0
Pasul 4	Intrarea de enable este 1 Circuitul memorează	Intrarea de date este 1	Frontul crescător	Ieșirea de date va fi 1 Circuitul memorează 1

Pasul 1 evidențiat în imagini (imaginea din partea stângă evidențiază ce s-a întâmplat anterior, iar imaginea din dreapta evidențiază ce se întâmplă acum)

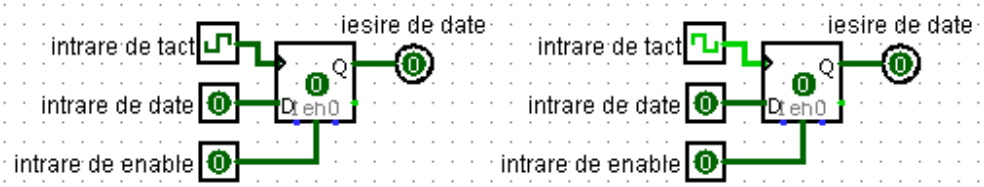


Figura 3 – Intrarea de enable este 0 (circuitul nu funcționează), intrarea de date este 0, intrarea de tact trece din 0 în 1.

Pasul 2 evidențiat în imagini (imaginea din partea stângă evidențiază ce s-a întâmplat anterior, iar imaginea din dreapta evidențiază ce se întâmplă acum)

Figura 4 – Intrarea de enable este 0 (circuitul nu funcționează), intrarea de date este 1, intrarea de tact trece din 0 în 1.

Pasul 3 evidențiat în imagini (imaginea din partea stângă evidențiază ce s-a întâmplat anterior, iar imaginea din dreapta evidențiază ce se întâmplă acum)

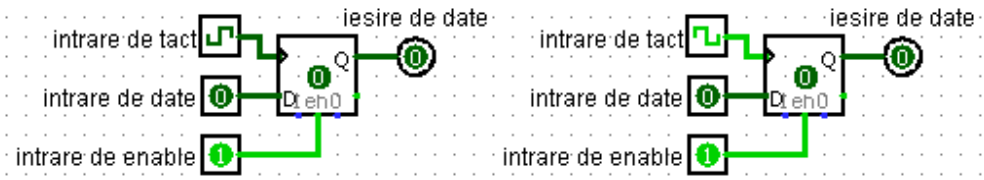


Figura 5 – Intrarea de enable este 1 (circuitul funcționează), intrarea de date este 0, intrare de tact trece din 0 în 1 și circuitul va memora 0.

Pasul 4 evidențiat în imagini (imaginea din partea stângă evidențiază ce s-a întâmplat anterior, iar imaginea din dreapta evidențiază ce se întâmplă acum)

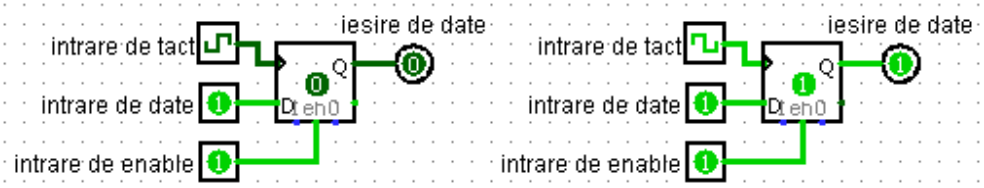


Figura 6 – Intrarea de enanle este 1 (circuitul funcționează), intrarea de date este 1, intrarea de tact trece din 0 în 1 și circuitul va memora 1.

Se observă că bitul din interiorul flip-flop-ului evidențiază valoarea memorată. Deci în figura 5 valoarea memorată este 0 (în interiorul flip-flop-ului vedem valoarea 0). În figura 6, ce era memorat anterior (valoarea logică 0) este înlocuit cu valoarea actuală (valoarea logică 1).

4.3.2 Registrul paralel

În figura 7 este evidențiat un registru paralel pe 4 biți. Se observă că registrul este alcătuit din 4 flip-flop-uri de tip D puse unul lângă altul (în paralel). Fiecare flip-flop are intrare proprie și ieșire proprie. Intrările comune sunt intrarea de reset, intrarea de enable și intrarea de tact. Intrarea de tact este comună deoarece flip-flop-urile trebuie să funcționeze după o referință comună, fiecare element trebuie să fie sincronizat cu celelalte elemente.

Atunci când intrarea de enable va avea valoarea logică 1 circuitul va funcționa și va memora valorile de la intrări. Atunci când intrarea de enable va avea valoarea logică 0, circuitul nu va mai funcționa, nu va mai memora valorile de la intrări, var își va menține starea actuală memorată.

Atunci când intrarea de reset va avea valoarea logică 1, va reseta bitul memorat de flip-flop și îl va înlocuit cu valoarea logică 0.

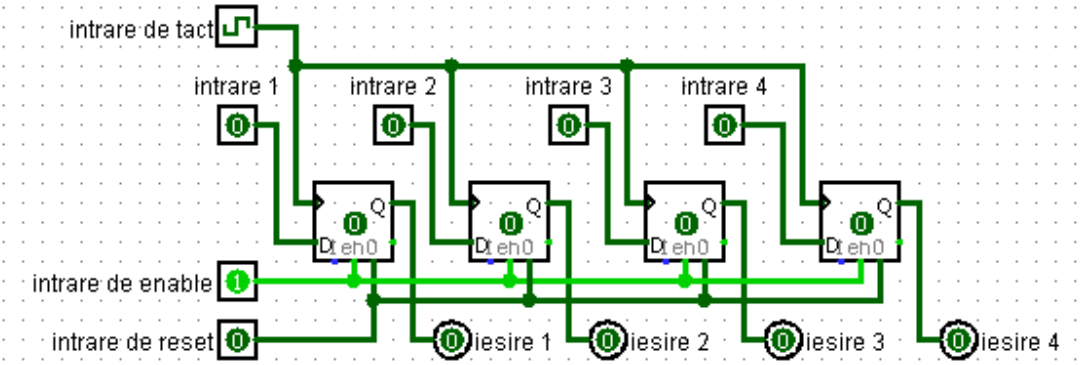


Figura 7 – Registru paralel pe 4 biți. Avem 4 intrări pe 1 bit și 4 ieșiri pe 1 bit.

În figura 8 este evidențiată o altă formă a registrului paralel pe 4 biți. Dacă în figura 7 am avut 4 intrări și 4 ieșiri pe 1 bit, în figura 8 avem o intrare și o ieșire pe 4 biți, deci circuitele sunt echivalente.

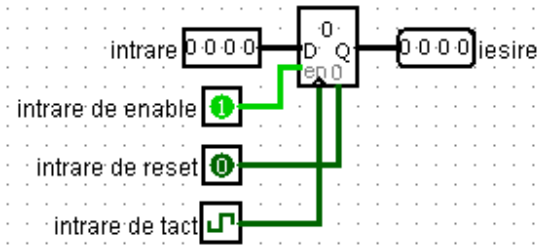


Figura 8 – Registru paralel pe 4 biți. Avem o intrare pe 4 biți și o ieșire pe 4 biți.

Exercițiu 1.

Să se implementeze în Logisim circuitele din figurile 7 și 8 și să se completeze tabelul 2. Ce se va întâmpla cu ieșirea în momentul când intrarea de reset este 0. Dar în momentul când intrarea de enable este 0?

Tabel 2 – Testarea registrului paralel pe 4 biți. Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)

Mai întâi se va seta semnalul de enable, apoi se vor seta intrările, apoi semnalul de reset și apoi se va apăsa de două ori pe simbolul semnalului de tact									
Intrarea de enable va fi 1					Intrare de tact pentru o perioadă				
Intrare 1	Intrare 2	Intrare 3	Intrare 4	Intrare de reset	Se va apăsa de două ori pe simbolul semnalului de tact	Ieșire 1	Ieșire 2	Ieșire 3	Ieșire 4
1	1	1	1	0					
1	0	0	1	0					
1	0	1	0	1					
1	1	1	1	1					
0	1	1	0	0					
Intrarea de enable va fi 0					Intrare de tact pentru o perioadă				
Intrare 1	Intrare 2	Intrare 3	Intrare 4	Intrare de reset	Se va apăsa de două ori pe simbolul semnalului de tact	Ieșire 1	Ieșire 2	Ieșire 3	Ieșire 4
1	1	1	1	0					
1	1	1	0	0					
1	1	1	1	1					

4.3.3 Registrul serial

În figura 9 este evidențiat un registru serial pe 4 biți. Se observă că registrul este alcătuit din 4 flip-flop-uri de tip D conectate în serie. Ieșirea Q a primului flip-flop este conectată cu intrarea D a celui de al doilea flip-flop, ieșirea Q a acestuia este conectată cu intrarea D a celui de al treilea flip-flop și așa mai departe. Registrul serial pe 4 biți are o intrare de date pe 1 bit, o ieșire serială de date și o ieșire paralelă de date. Intrările comune celor 4 flip-flop-uri sunt intrarea de enable și intrarea de reset.

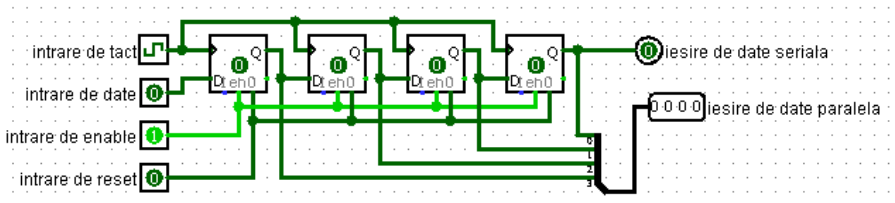


Figura 9 – Registru serial pe 4 biți.

Exercițiu 1.

Să se implementeze în Logisim circuitul din figura 9 și să se completeze tabelul 3. Ce se va întâmpla cu ieșirea paralelă în momentul când intrarea de reset este 0? Dar în momentul când intrarea de enable este 0?

Tabel 3 – Testarea registrului serial pe 4 biți.

<table>
<tr><td colspan="6">Mai întâi se va seta semnalul de enable, apoi se va seta intrarea de date, apoi semnalul de reset și apoi se va apăsa de două ori pe simbolul semnalului de tact</td></tr>
<tr><td colspan="2">Intrarea de enable va fi 1</td><td>Intrare de tact pentru o perioadă</td><td colspan="3">Ieșire de date paralelă</td></tr>
<tr><td>Intrare de date</td><td>Intrare de reset</td><td rowspan="18">Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)</td><td>Bitul 1 din stânga</td><td>Bitul 2</td><td>Bitul 3</td><td>Bitul 4 din dreapta</td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Intrarea de enable va fi 0</td><td>Intrare de tact pentru o perioadă</td><td colspan="3">Ieșire de date paralelă</td></tr>
<tr><td>Intrare de date</td><td>Intrare de reset</td><td rowspan="3">Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)</td><td>Bitul 1 din stânga</td><td>Bitul 2</td><td>Bitul 3</td><td>Bitul 4 din dreapta</td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td></td><td></td><td></td><td></td></tr>
</table>

4.3.4 Registrul universal

În figura 10 este evidențiat un registru universal pe 4 biți. Se observă că registrul universal este acătuit din 4 flip-flop-uri de tip D și 4 multiplexoare 4 la 1 pe 1 bit. În funcție de valoarea intrării de selecție registrul universal se comportă ca un registru serie, registru paralel sau memorează starea actuală. Funcționarea registrului universal este evidențiată în tabelul 4.

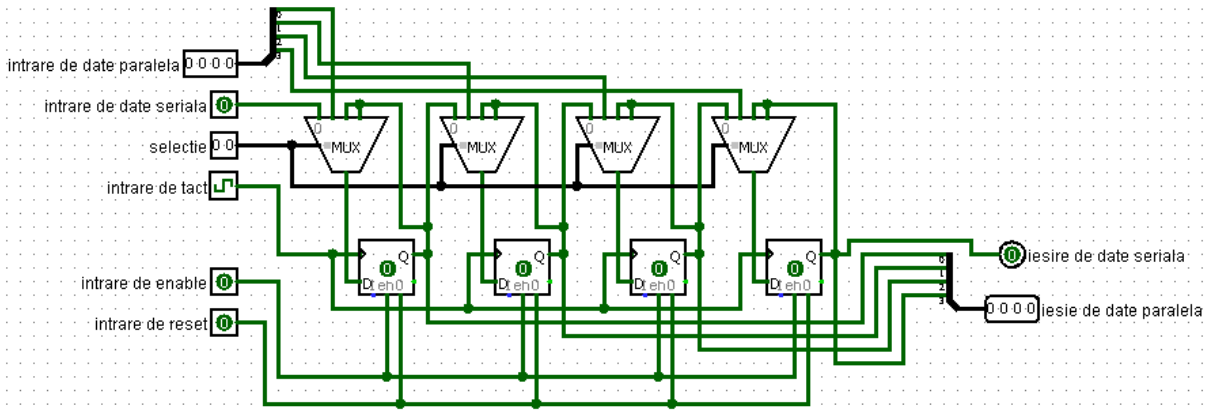


Figura 10 – Registru universal pe 4 biți.

Tabel 4 – Funcționarea resitrului universal din figura 10.

Intrare de selecție	Registrul universal ...
00	devine registru serial
01	devine registru paralel
10	își păstrează starea actuală
11	

Multiplexorul se va confgura ca în figura 11.

Selection: Multiplexer	
Facing	South
Select Location	Bottom/Left
Select Bits	2
Data Bits	1
Disabled Output	Floating
Include Enable?	No

Figura 11 – Configurarea multiplexoarelor registrului universal din figura 10.

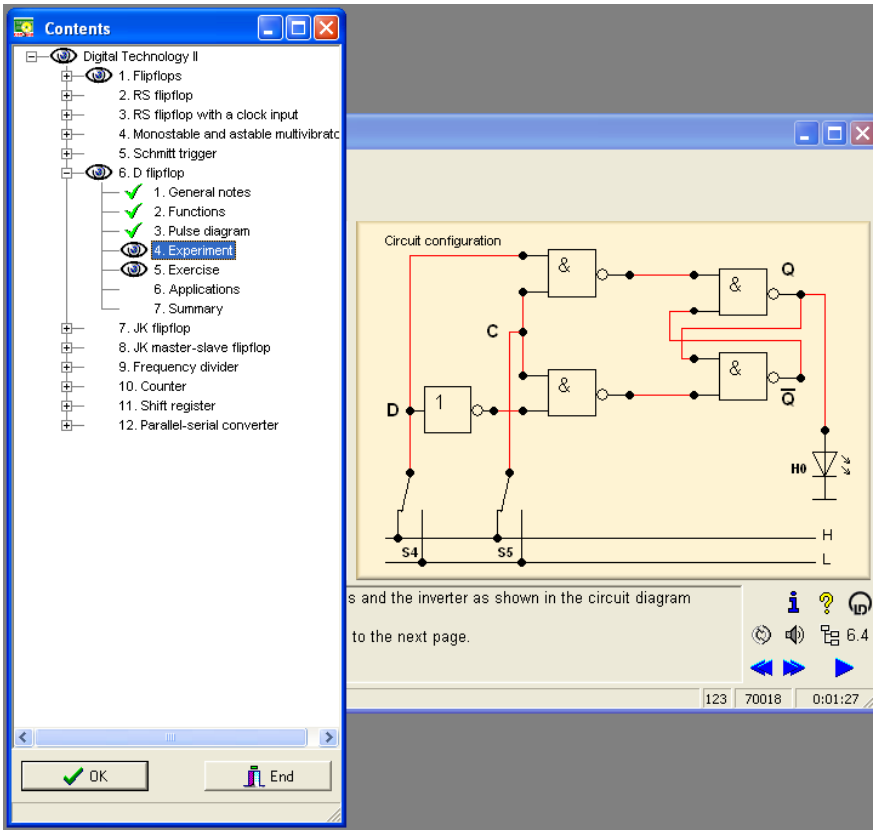
Exercițiul 1.

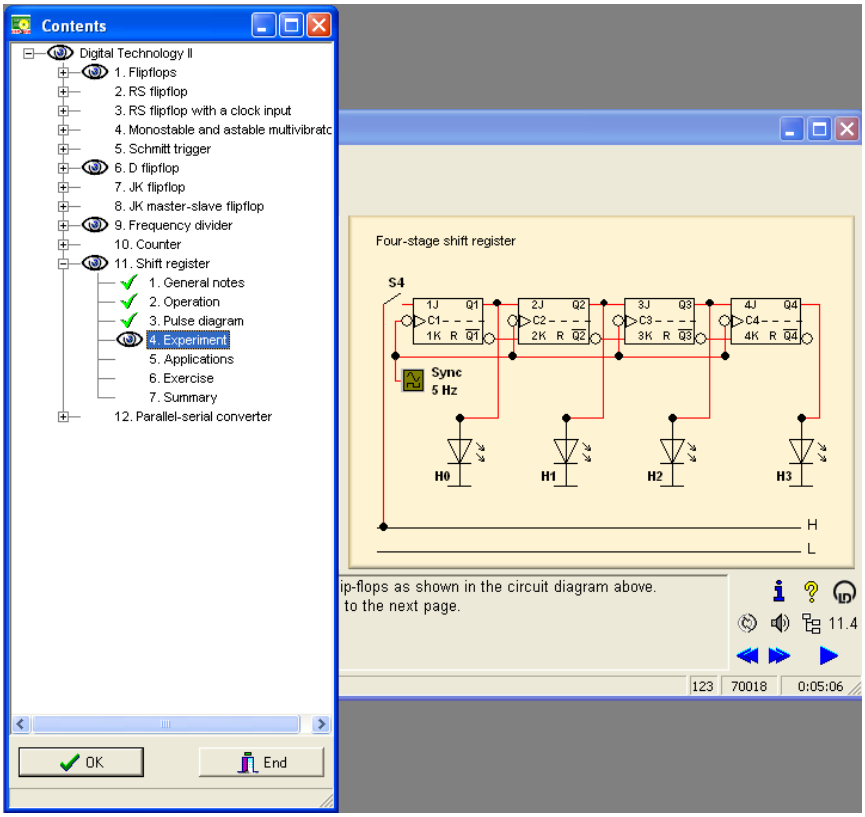
Vom simula funcționarea registrului universal. Intrarea de enable va fi 1, intrarea de reset va fi 0. Vom parcurge pașii din tabelul 5.

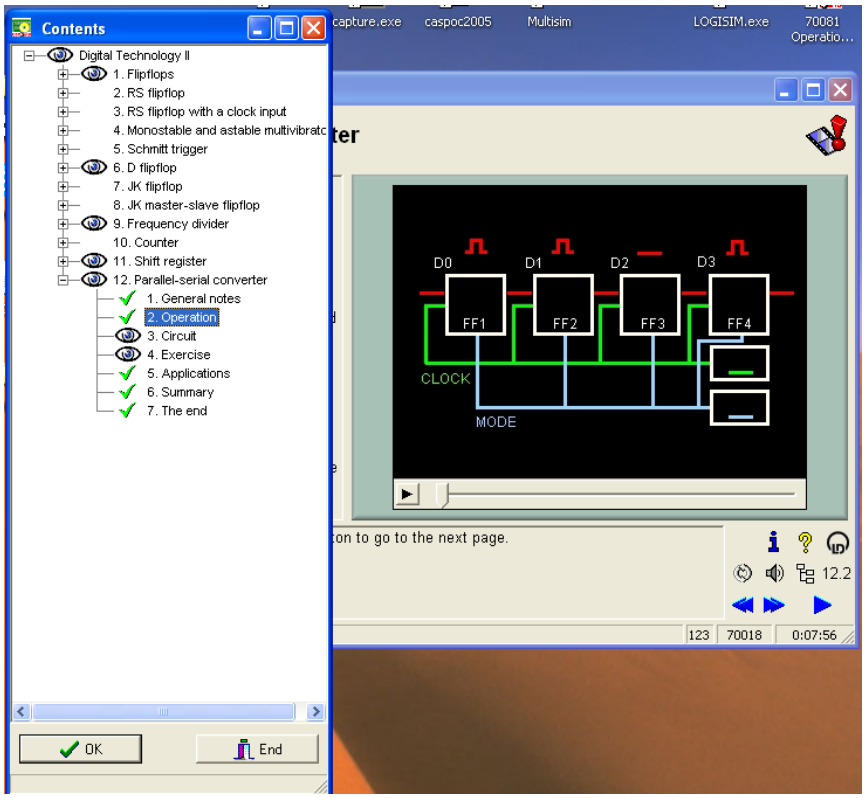
Tabel 5 – Testarea registrului universal.

Pasul	Intrarea de selecție	Intrarea de date serială	Intrarea de date paralelă	Intrarea de tact	Ieșirea paralelă
1	01	0	0110	Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)	
2	00	0	1111		
3	00	0	1111		
4	00	1	1111		
5	00	1	1111		
6	10	1	1111		
7	10	0	1111		
8	01	1	0111		
9	10	1	1111		

4.4 Efectuarea laboratorului în COM3LAB




Contents
Digital Technology II
1. Flipflops
2. RS flipflop
3. RS flipflop with a clock input
4. Monostable and astable multivibratc
5. Schmitt trigger
6. D flipflop
7. JK flipflop
8. JK master-slave flipflop
9. Frequency divider
10. Counter
11. Shift register
1. General notes
2. Operation
3. Pulse diagram
4. Experiment
5. Applications
6. Exercise
7. Summary
12. Parallel-serial converter
OK
End
Four-stage shift register
S4
Sync
5 Hz
H0
H1
H2
H3
H
L
ip-flops as shown in the circuit diagram above.
to the next page.
11.4
123
70018
0:05:06


Contents
Digital Technology II
1. Flipflops
2. RS flipflop
3. RS flipflop with a clock input
4. Monostable and astable multivibratc
5. Schmitt trigger
6. D flipflop
7. JK flipflop
8. JK master-slave flipflop
9. Frequency divider
10. Counter
11. Shift register
12. Parallel-serial converter
1. General notes
2. Operation
3. Circuit
4. Exercise
5. Applications
6. Summary
7. The end
OK
End
capture.exe
caspoc2005
Multisim
LOGISIM.exe
70081
Operatio...
D0
D1
D2
D3
FF1
FF2
FF3
FF4
CLOCK
MODE
on to go to the next page.
12.2
123
70018
0:07:56

Nume şi prenume	Grupa	Anul de studiu

Laboratorul 4
Raport de laborator

4.3.2 Registrul paralel
Exerciţiul 1.

Tabel 1 – Testarea registrului paralel pe 4 biţi. Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)

Pasul	Mai întâi se va seta semnalul de enable, apoi se vor seta intrările, apoi semnalul de reset şi apoi se va apăsa de două ori pe simbolul semnalului de tact									
1	Intrarea de enable va fi 1					Intrare de tact pentru o perioadă				
	Intrare 1	Intrare 2	Intrare 3	Intrare 4	Intrare de reset	Se va apăsa de două ori pe simbolul semnalului de tact	Ieşire 1	Ieşire 2	Ieşire 3	Ieşire 4
2	1	1	1	1	0					
3	1	0	0	1	0					
4	1	0	1	0	1					
5	1	1	1	1	1					
6	0	1	1	0	0					
7	Intrarea de enable va fi 0					Intrare de tact pentru o perioadă				
	Intrare 1	Intrare 2	Intrare 3	Intrare 4	Intrare de reset	Se va apăsa de două ori pe simbolul semnalului de tact	Ieşire 1	Ieşire 2	Ieşire 3	Ieşire 4
8	1	1	1	1	0					
9	1	1	1	0	0					
10	1	1	1	1	1					

4.3.4 Registrul universal
Exerciţiul 1.

Tabel 2 – Testarea registrului universal.

Pasul	Intrarea de selecţie	Intrarea de date serială	Intrarea de date paralelă	Intrarea de tact	Ieşirea paralelă
1	01	0	0110	Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)	
2	00	0	1111		
3	00	0	1111		
4	00	1	1111		
5	00	1	1111		
6	10	1	1111		
7	10	0	1111		
8	01	1	0111		
9	10	1	1111		

4.3.3 Registrul serial
Exercițiul 1.

Tabel 3 – Testarea registrului serial pe 4 biți.

Pasul	Mai întâi se va seta semnalul de enable, apoi se va seta intrarea de date, apoi semnalul de reset și apoi se va apăsa de două ori pe simbolul semnalului de tact						
1	Intrarea de enable va fi 1		Intrare de tact pentru o perioadă	Ieșire de date paralelă			
	Intrare de date	Intrare de reset	Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)	Bitul 1 din stânga	Bitul 2	Bitul 3	Bitul 4 din dreapta
2	1	0					
3	1	0					
4	1	0					
5	1	0					
6	0	0					
7	0	0					
8	0	0					
9	0	0					
10	1	0					
11	1	0					
12	0	0					
13	0	0					
14	1	1					
15	1	0					
16	0	0					
17	0	0					
18	Intrarea de enable va fi 0		Intrare de tact pentru o perioadă	Ieșire de date paralelă			
	Intrare de date	Intrare de reset	Se va apăsa de două ori pe simbolul semnalului de tact (acest lucru se va face pentru fiecare rând al tablelului)	Bitul 1 din stânga	Bitul 2	Bitul 3	Bitul 4 din dreapta
19	1	0					
20	1	0					
21	1	1					

LAB
5

Implementarea maşilor cu stări finite de tip Moore

5.1 Obiectivele laboratorului

În acest laborator se va defini modul de implementarea a maşinilor cu stări finite de tip Moore.

5.2 Definiţii

5.2.1 Maşina cu stări finite de tip Moore

În figura 1 este evidenţiată structura maşinii cu stări finite de tip Moore. Precizăm că vom prescurta maşina cu stări finite prin acronimul FSM (Finite Statea Machine).

Blocul numit starea următoare este un circuit logic combinaţional ce determină starea următoare pe baza stării curente şi a intrării în FSM. Blocul numit ieşire este un circuit logic combinaţional ce determină ieşirea la un moment dat pe baza stării curente. Blocul numit FF este un registru paralel, un circuit logic secvenţial sincron, ce memorează starea următoare şi oferă la ieşire starea curentă. Datorită blocului FF stările FSM-ului de tip Moore se succed sincron cu impulsul de tact (CLK). Intrarea de date poate să aibă P biţi, blocul FF poate memora n biţi, iar ieşirea de date poate să aibe R biţi.

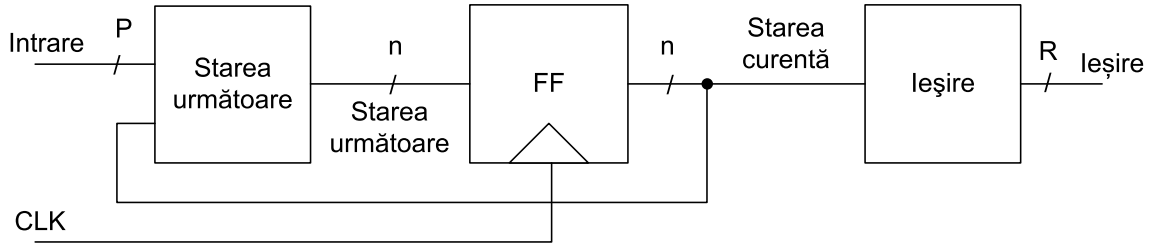


Figura 1 – Structura generală a maşinii cu stări finite de timp Moore.

5.3 Efectuarea laboratorului

5.3.1 Implementarea maşinii cu stări finite de tip Moore în Logisim

Din figura 1 se observă că FSM-ul de tip Moore aduce laolaltă circuitele logice combinaţionale şi circuitele logice secvenţiale sincrone.

Modulul Starea următoare este un circuit logic combinaţional. Acest modul are două intrări: intrarea de date pe P biţi şi intrarea Starea curentă pe n biţi. Acest modul are şi o ieşire numită Starea următoare pe n biţi.

Modulul Ieşire este un circuit logic combinaţional. Acest modul are o intrare numită Starea curentă pe n biţi şi o ieşire de date pe R biţi.

Modul de implementare a mașinii cu stări finite de tip Moore în Logism

Dorim să implementăm un numărător care numără de la 0 la 7. Pentru a număra de la 0 la 7 vom avea nevoie de un registru paralel pe 3 biți pentru ca circuitul să țină minte toate cele 8 valori. Avem nevoie de 3 biți deoarece $3 = \log_2 8$. Dacă doream să numărăm de la 0 la 15 aveam nevoie de 4 biți deoarece $4 = \log_2 16$.

Acest numărător va fi implementat ca o mașină cu stări finite de tip Moore. Numărătorul nu va avea intrări de date. Deci FSM-ul de tip Moore nu va avea intrări de date. Va fi nevoie să implementăm circuitul logic combinațional Starea următoare și circuitul logic combinațional Ieșire și să facem conexiunile cu registrul paralel pe 3 biți FF, ca în figura 1. Va mai fi nevoie de un afișor cu 7 segmente, ce va afișa cele 8 numere.

Tabelul de adevăr al circuitului logic combinațional Starea următoare este evidențiat în tabelul 1, iar tabelul de adevăr al circuitului logic combinațional Ieșire este evidențiat în tabelul 2.

Tabel 1 – Tabelul de adevăr al modulului Starea următoare. Starea curentă reprezintă intrarea, iar Starea următoare reprezintă ieșirea.

Starea curentă	Starea următoare
000	001
001	010
010	011
011	100
100	101
101	110
110	111
111	000

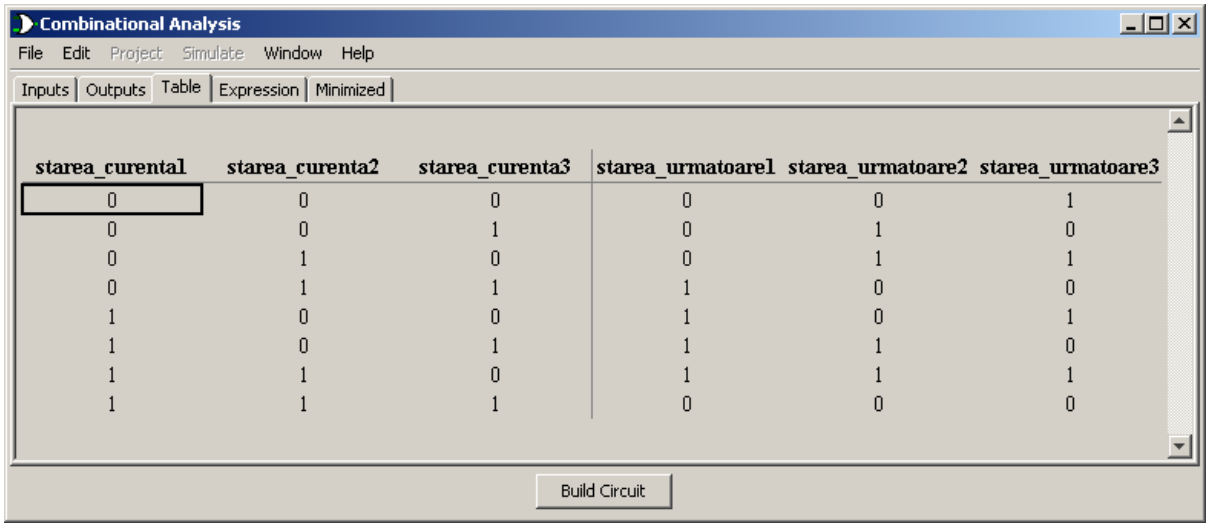
Tabel 2 – Tabelul de adevăr al modulului Ieșire. Starea curentă reprezintă intrarea, iar Ieșire reprezintă ieșirea.

Starea curentă	Ieșire
000	1111110
001	0110000
010	1101101
011	1111001
100	0110011
101	1011011
110	0011111
111	1110000

Se vor implementa cele două tabele în Logisim cu ajutorul Combinational Analysis, care se găsește în meniul pop-up Window.

Pasul 1.

Vom implementa modulul Starea următoare. Tabelul de adevăr în fereastra Combinational Analysis va arata ca în figura 2. Acest circuit va fi creat ca modul simplu și va avea numele Starea următoare ca în figura 3. Circuitul din modulul Starea următoare va arata ca în figura 4.



starea_curental	starea_curenta2	starea_curenta3	starea_urmatoare1	starea_urmatoare2	starea_urmatoare3
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Figura 2 – Tabelul de adevăr pentru modulul Starea următoare implementat în Logim în fereastra Combinational Analysis.

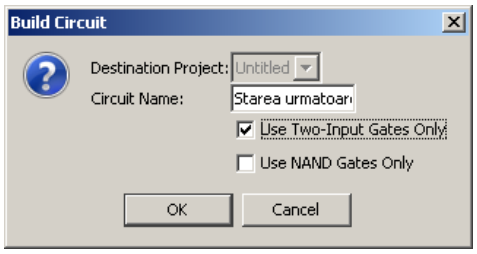


Figura 3 – Modulul se va numi Starea urmatoare.

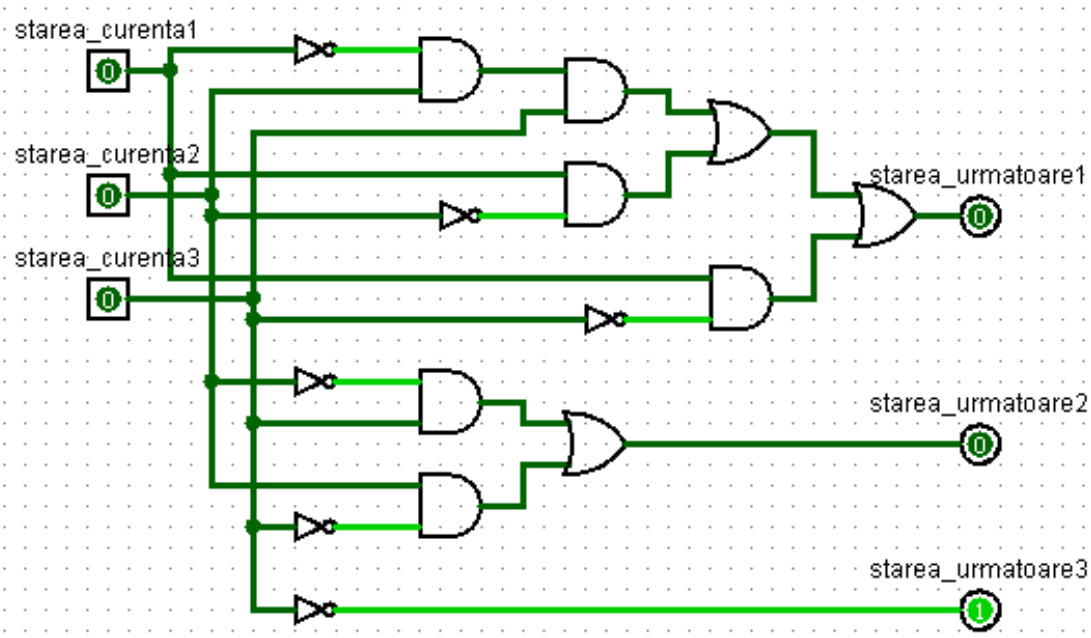
Pasul 2.

Vom implementa modulul Ieșire. Tabelul de adevăr în fereastra Combinational Analysis va arata ca în figura 5. Acest circuit va fi creat ca modul simplu și va avea numele Ieșire ca în figura 6. Circuitul din modulul Ieșire va arata ca în figura 7.

Pasul 3.

Se va crea un modul simplu cu numele FF, care va cuprinde 3 flip-flop-uri de tip D puse în paralel ca în figura 8.

Pasul 4. Se vor instanția în main cele 3 module și se vor conecta ca în figura 9.



Figură 4 - Circuitul din modulul Starea următoare.

Combinational Analysis

File Edit Project Simulate Window Help

Inputs | Outputs | Table | Expression | Minimized

starea_curental	starea_curenta2	starea_curenta3	Iesire0	Iesire1	Iesire2	Iesire3	Iesire4	Iesire5	Iesire6
0	0	0	1	1	1	1	1	0	0
0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	0	1	1	0	1
0	1	1	1	1	1	1	0	0	1
1	0	0	0	1	1	0	0	1	1
1	0	1	1	0	1	1	0	1	1
1	1	0	0	0	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0

Build Circuit

Figura 5 - Tabelul de adevăr pentru modulul Ieşire implementat în Logim în fereastra Combinational Analysis.

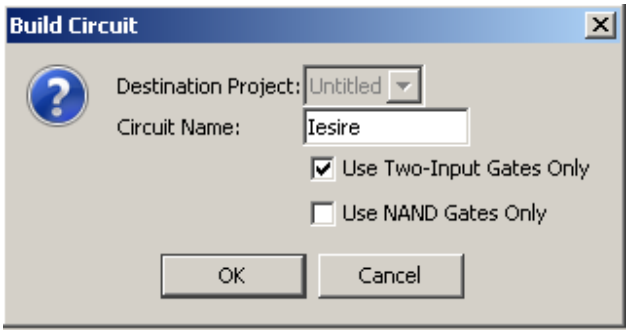


Figura 6 – Modulul se va numi Ieşire.

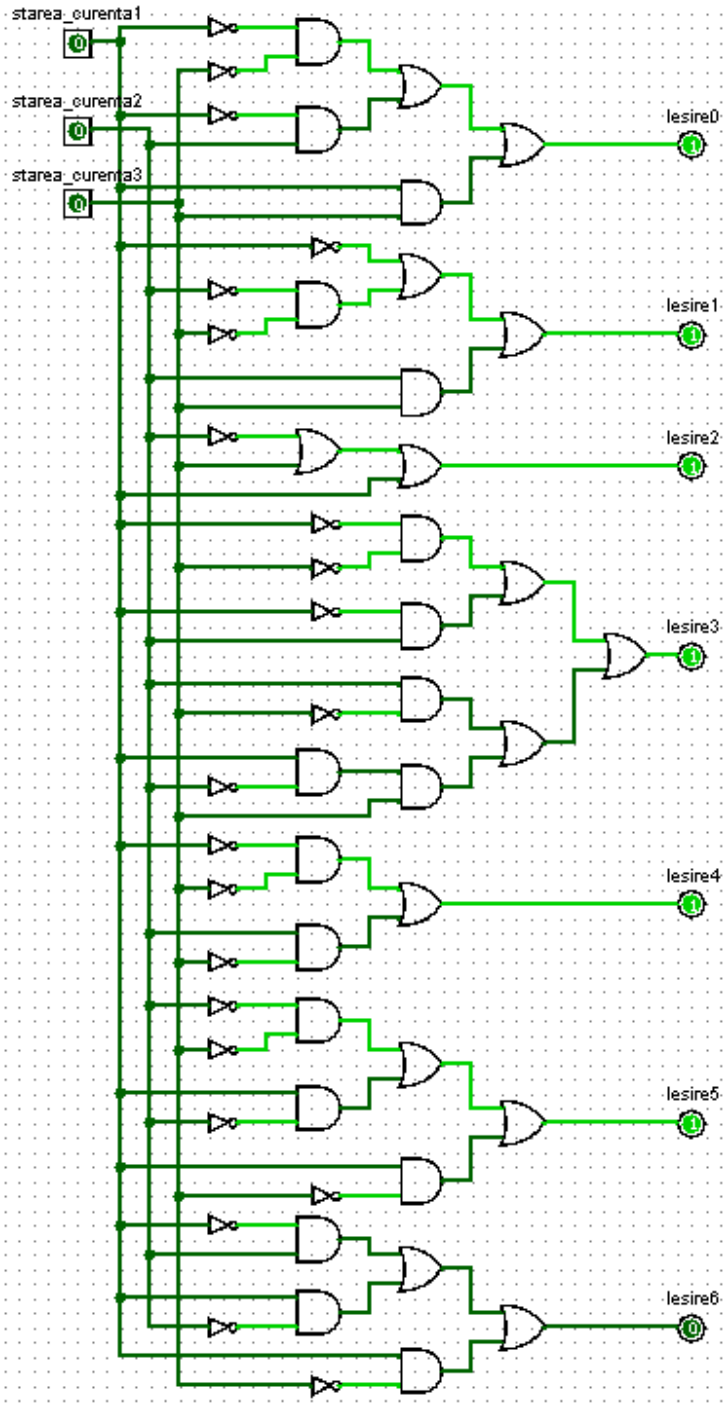


Figura 7 - Circuitul din modulul Ieșire.

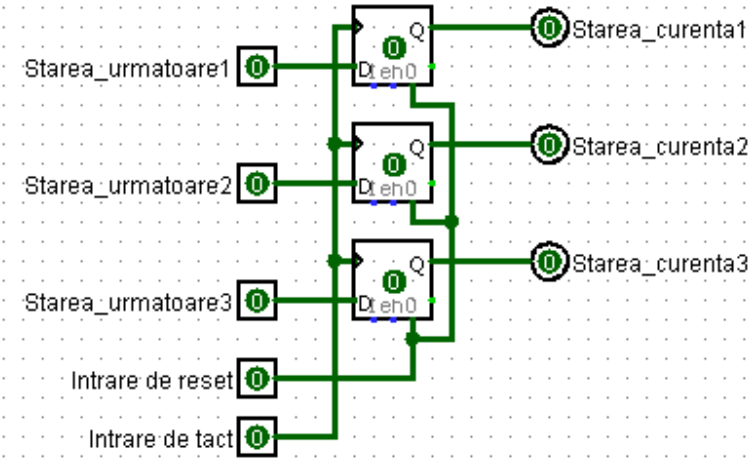


Figura 8 – Registrul paralel pe 3 biți din modulul FF.

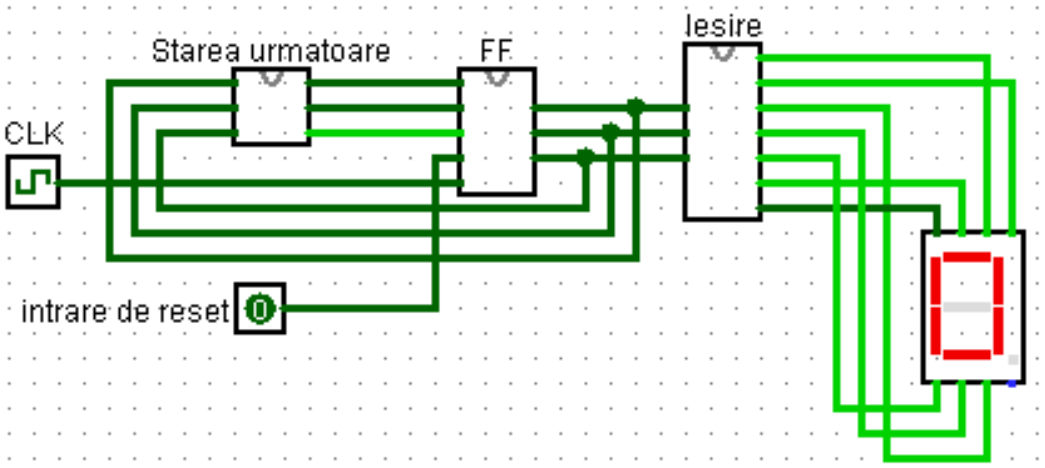


Figura 9 – Numărător pe 3 biți (de la 0 la 7).

Exercițiul 1. Să se implementeze un numărător ca FSM de tip Moore, care să numere de la 0 la 15. Acest numărător va număra în baza 16 (în hexazecimal). Să se completeze tabelele de adevăr pentru modulul Starea următoare și Ieșire. Să se execute în Logisim circuitul. Câte flip-flop-uri de tip D va avea registrul paralel?

Ajutor pentru exercițiul 1. În baza 16 cifra 10 se reprezintă cu A, cifra 11 se reprezintă cu B, cifra 12 se reprezintă cu C, cifra 13 se reprezintă cu D, cifra 14 se reprezintă cu E, iar cifra 15 se reprezintă cu F.

Execițiul 2. Să se implementeze un numărător ca FSM de tip Moore, care să numere din 2 în 2 începând cu 0. Să se completeze tabelele de adevăr pentru modulul Starea următoare și Ieșire. Să se execute în Logisim circuitul. Registrul paralel va reține 3 biți.

Exercițiul 3. Să se implementeze un numărător ca FSM de tip Moore, care să numere de la 0 la 9. Să se completeze tabelele de adevăr pentru modulul Starea următoare și Ieșire. Să se execute în Logisim circuitul. Câte flip-flop-uri de tip D va avea registrul paralel?

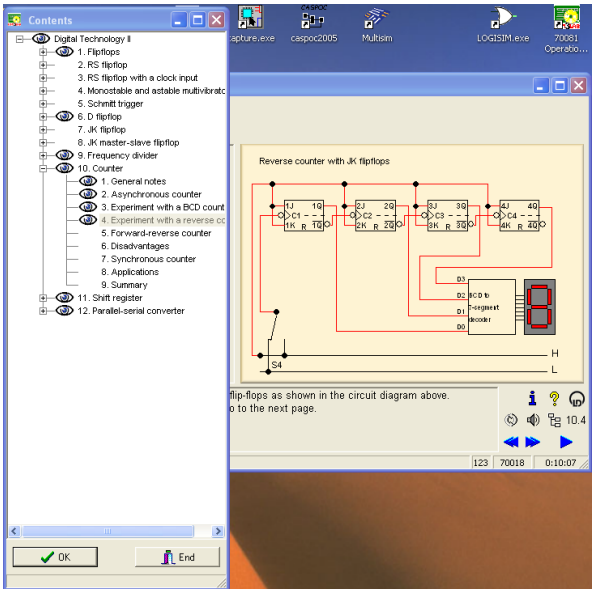
Exercițiul 4. Să se implementeze un automat de cafea ca FSM de tip Moore. Acest circuit va accepta două monede: una de valoarea 25 și alta de valoarea 50. O cafea la acest automat costă 75. Monedele se vor introduce una câte una, deci automatul va accepta la un moment dat o singura moneda. Automatul pe lângă cele două intrări va avea și două ieșiri: o ieșire denumită produs, care va acționa un sistem, care va turna cafea în pahar și o ieșire rest, care va acționa un alt sistem cu monede. Să se completeze tabele de adevăr pentru modulul Starea următoare și Ieșire și să se implementeze mașina cu stări finite de tip Moore în Logisim.

Ajutor pentru exercițiul 4. Între intrările sistemului și intrările mașinii cu stări finite de tip Moore există un decodificator care va avea următorul tabel de adevăr. Atunci când nu va fi introdusă nicio monedă ieșire sistemului va fi 0, atunci când una dintre monezi va fi introdusă sistemul va sesiza valoarea monedei, iar dacă se încearcă introducerea ambelor monede atunci ieșirea sistemului va fi 0.

Tabel 3 – Tabel de aevăr pentru decodificatorul de monede. I50 înseamnă că a fost introdusă moneda cu valoarea 50, I25 înseamnă că a fost introdusă moneda cu valoarea 25.

I50	I25	Y50	Y25
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

5.4 Efectuarea laboratorului în COM3LAB



Nume şi prenume	Grupa	Anul de studiu

Laboratorul 5
Raport de laborator

5.3.1 Implementarea mașinii cu stări finite de tip Moore în Logisim

Exercițiul 1.

Tabel 1 – Tabelul de adevăr al modulului Starea următoare. Starea curentă reprezintă intrarea, iar Starea următoare reprezintă ieșirea.

Starea curentă	Starea următoare

Tabel 2 – Tabelul de adevăr al modulului Ieșire. Starea curentă reprezintă intrarea, iar Ieșire reprezintă ieșirea.

Starea curentă	Ieșire

Exercițiul 2.

Tabel 3 – Tabelul de adevăr al modulului Starea următoare. Starea curentă reprezintă intrarea, iar Starea următoare reprezintă ieșirea.

Starea curentă	Starea următoare

Tabel 4 – Tabelul de adevăr al modulului Ieșire. Starea curentă reprezintă intrarea, iar Ieșire reprezintă ieșirea.

Starea curentă	Ieșire

Exercițiul 3.

Tabel 5 – Tabelul de adevăr al modulului Starea următoare. Starea curentă reprezintă intrarea, iar Starea următoare reprezintă ieșirea.

Starea curentă	Starea următoare

Tabel 6 – Tabelul de adevăr al modulului Ieșire. Starea curentă reprezintă intrarea, iar Ieșire reprezintă ieșirea.

Starea curentă	Ieșire

LAB
6

Implementarea circuitelor aritmetico-logice și a memoriilor

6.1 Obiectivele laboratorului

În acest laborator se va evidenția structura internă a unui circuit aritmetico-logic și se vor prezenta două tipuri de memorii, memoria ROM și RAM.

6.2 Definiții

6.2.1 Unitatea aritmetico-logică (ALU)

Un ALU aduce laolaltă circuite digitale ce implementează operațiile aritmetice și logice. Un ALU este o unitate aritmetico-logică cu două intrări pe un număr M de biți, cu o intrare de selecție, Sel, ce va selecta operația dorită și o ieșire pe același număr de biți, M.

În figura 1 este reprezentat tabelul de adevăr al unui ALU. În funcție de intrarea de selecție se poate opta pentru o operație aritmetică sau logică. Semnalele de intrare și ieșire vor avea o capacitate de 4 biți iar semnalul de selecție va avea o capacitate de 3 biți. Operațiile aritmetice sunt: adunare, scădere, egalitate și mai mic ca. Operațiile logice sunt: OR, AND, NOR, deplasare logică la stânga (înmulțire cu 2) și deplasare logică la dreapta (împărțire cu 2). Unitatea aritmetico-logică are ca operație implicită operația aritmetică de egalitate.

Trebuie menționat că acestă configurație, evidențiată în figura 1, nu este unică. Există ALU-uri ce implementează și alte funcții aritmetico-logice.

Sel	Operația aritmetico-logică	Observații
000	adunare	$A + B$
001	scădere	$A - B$ sau $A + \bar{B} + 1$
010	OR	$A \mid B$
011	AND	$A \,\&\, B$
100	sll	$A \ll B$, deplasare logică la stânga
101	srl	$A \gg B$, deplasare logică la dreapta
110	NOR	$A\ NOR\ B = \bar{A}\ OR\ \bar{B}$
111	mai mic	dacă $A < B$, $S = 1$, dacă nu, $S = 0$
- - -	egalitate	A = = B

Figura 1 - Tabel de adevăr pentru operațiile aritmetico-logice ale unui ALU.

6.2.2 Memoria ROM

O memorie ROM (read only memory) este reprezentată în figura 2. Memoria din figură va avea dimensiunea $n \times m$. Atunci când pe intrarea adresă se va furniza o adresă validă, datele stocate în memoria ROM se vor găsi la ieşirea “date ieşire”.

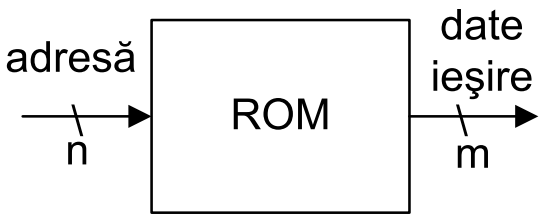
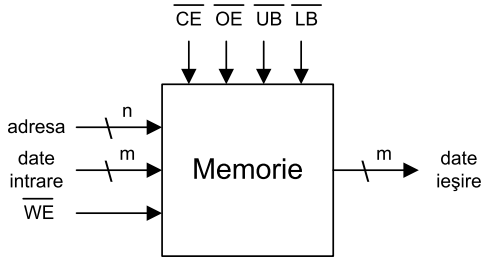


Figura 2 – Memorie ROM.

6.2.3 Memoria RAM

O memorie RAM (random access memory) este prezentată în figura 3. Diferența între memoria ROM și RAM este aceea că memoria ROM este nevolatilă și mai rapidă, iar memoria RAM este volatilă și de obicei este mai lentă. Acest tip de memorie poate avea intrarea de date comună cu ieșirea de date, intrarea de date separată de ieșirea de date, adresa de intrare comună cu cea de ieșire sau adresa de intrare separată de cea de ieșire. O memorie RAM poate avea câteva semnale de control, cum ar fi: intrarea scriere/citire (când această intrare are valoarea logică 0 putem citi memoria, iar când are valoarea logică 1 putem scrie memoria) și intrarea de enable numită chip enable (când această intrare are valoarea logică 0 memoria nu funcționează, iar când are valoarea logică 1 memoria funcționează).



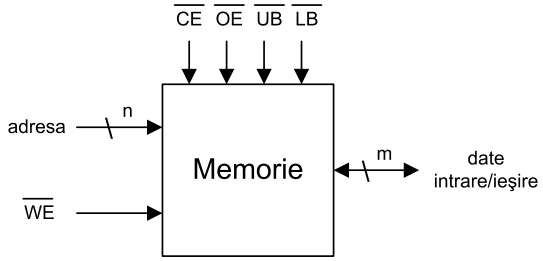
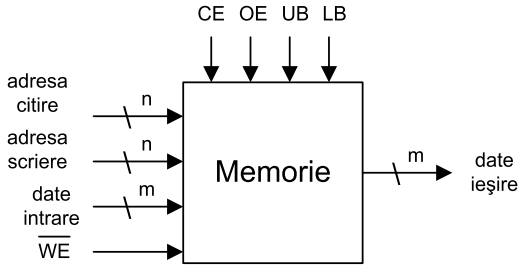


Figura 3.1 – Memorie cu port distinc de intrare şi ieşire. Figura 3.2 – Memorie cu port bidirecţional intrare/ieşire.



Figură 3.3 - Memorie cu port distinc pentru intrarea de adresă.

6.3 Efectuarea laboratorului

6.3.1 Implementarea unui ALU

Exercițiul 1.

Să se implementeze în Logisim circuitul din figura 4 și circuitul din figura 5 ca module ierahizate simple. Circuitul din figura 4 va implementa tabelul de adevăr din figura 6 și reprezintă celula fundamentală a ALU-ului nostru. Circuitul din figura 5, după cum se observă este identic în alcătuire cu circuitul din figura 4, însă are o ieșire în plus, ieșirea m.

După ce aceste două module au fost create, vom avea nevoie de 3 module ALU simple pe 1 bi și un modul ALU extins pe 1 bit pentru a creat în Logisim, în main, circuitul din figura 7.

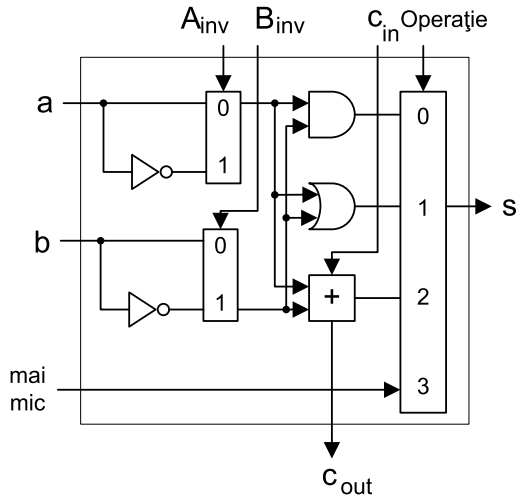


Figura 4 – Modul ALU simplu pe 1 bit.

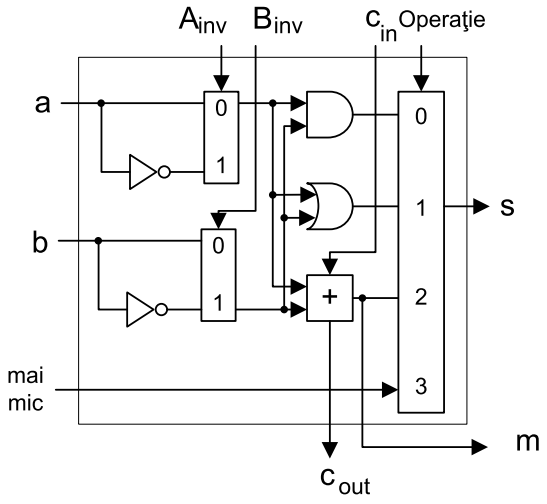


Figura 5 – Modul ALU extins pe 1 bit.

ALU intrări de control	Operație
0 0 0 0	AND
0 0 0 1	OR
0 0 1 0	ADUNARE
0 1 1 0	SCĂDERE
0 1 1 1	MAI MIC
1 1 0 0	NOR

Figura 6 - Tabelul intrărilor de control și a operațiilor implementate de către ALU.

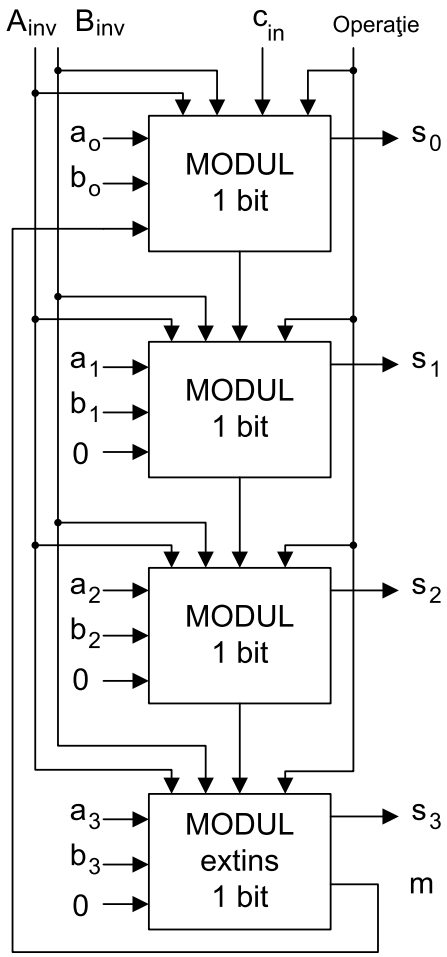


Figura 7 – ALU pe 4 biţi ce implementează tabelul din figura 6.

6.3.2 Implementarea unui ROM

Exerciţiul 1.

Se va implementa circuitul ROM din figura 8. Această memorie are o capacitate de $2^4 \cdot 4$ biți (64 biți sau 8 bytes).

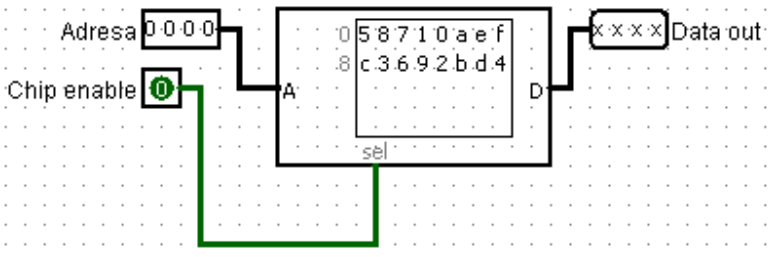
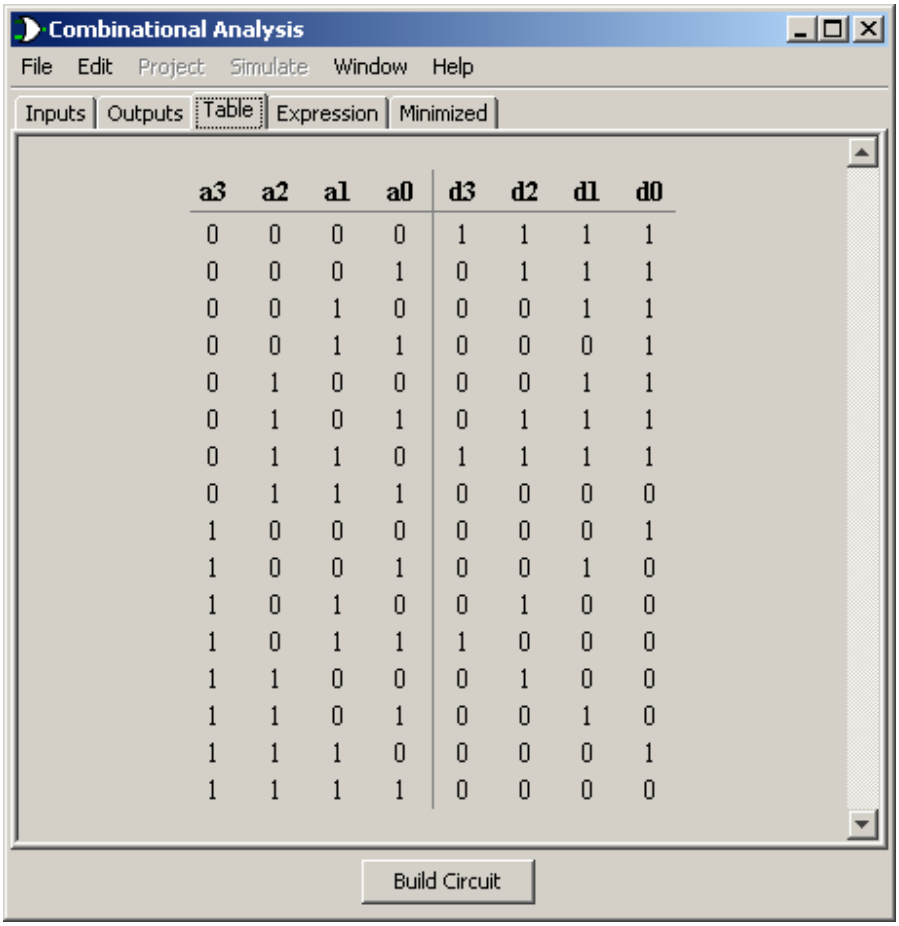


Figura 8 – Memorie ROM.

Exercițiul 2.

Se va implementa tabelul de adevăr din figura 9, cu metoda Combinational analysis. Acesta este un mod de a implementa o memorie ROM.



a3	a2	a1	a0	d3	d2	d1	d0
0	0	0	0	1	1	1	1
0	0	0	1	0	1	1	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	1
0	1	0	1	0	1	1	1
0	1	1	0	1	1	1	1
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	1	0	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

Figura 9 – Implementarea unui ROM prin table de adevăr.

Circuitul în Logisim ar trebui să arate ca în figura 10.

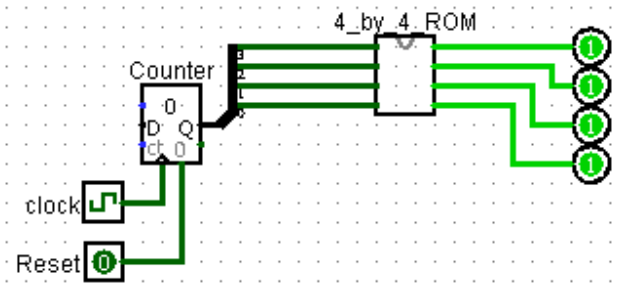


Figura 10 – Circuit cu memorie ROM și numărător.

6.3.3 Implementarea unui RAM

Exercițiu 1.

Se va implementa circuitul RAM din figura 11. Această memorie are o capacitate de $2^4 \cdot 4$ biți (64 biți sau 8 bytes). Are o intrare de adresă comună și bineînțeles intrarea de date este separată de ieșirea de date. Se va implementa modulul denumit Read_Write. Acest modul acceptă o intrare, intrarea Read_0_Write_1 (când intrarea este 0 memoria va fi citită, când intrarea este 1 memoria va fi scrisă), circuit ce este alcătuit ca în figura 12.

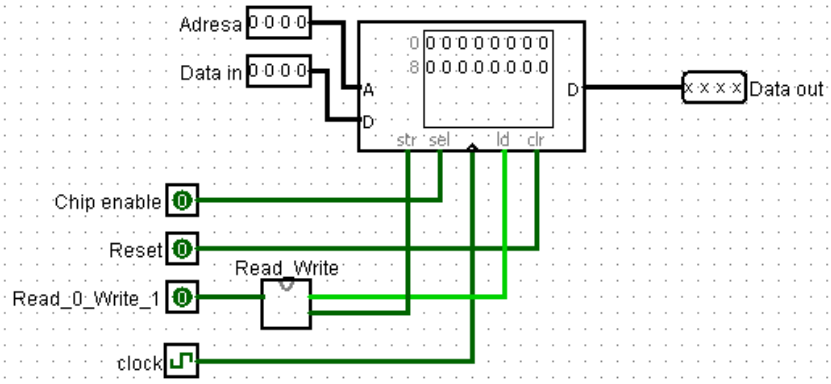


Figura 11 – Memorie RAM.

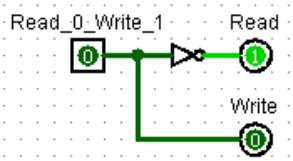


Figura 12 – Modulul Read_Write.

A. Se vor scrie în memorie următoarele date:

Mai întâi se vor configura semnalele de control Chip enable = 1, Reset = 0, Read_0_Write_1 = 1.

1. La Adresa = 0000, Data_in = 1010, dublu click pe clock.

2. La Adresa = 0110, Data_in = 0110, dublu click pe clock.

B. Se vor citi datele scrise anterior în memorie:

Mai întâi se vor configura semnalele de control Chip enable = 1, Reset = 0, Read_0_Write_1 = 0.

1. La Adresa = 0000, Data_out = 1010.

2. La Adresa = 0110, Data_out = 0110.