MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA Universitatea Tehnică a Moldovei Facultatea Calculatoare, Informatică şi Microelectronică Departamentul Inginerie Software și Automatică

Copta Adrian | FAF-223

Raport

Lucrare de laborator n.2

Arhitecturi de calculatoare

Verificat:

Voitcovschi Vladislav *asist.univ*

Chișinău – 2024

1. **Scopul lucrării:**

Laboratorul 3 reprezinta un ansamblu de exercitii in LogiSim. Fiecare nivel de exerciții are ponderea sa. Easy – 0.2 p Medium – 0.4 p Hard – 0.8 p Necesita sa executați exerciții din toate 3 nivele Scopul fiecărui este de a acumula 10 p.

2. **Introducere:**

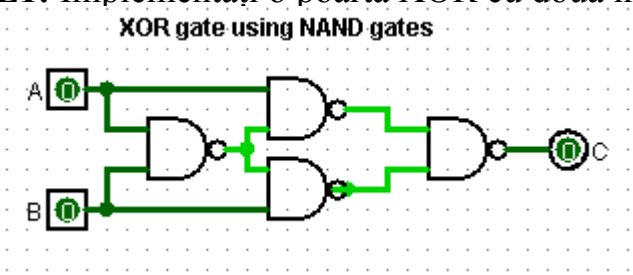
Logisim este o unealtă educațională folosită pentru proiectarea și simularea circuitelor digitale. Cu o interfață simplă în bara de instrumente și simularea circuitelor pe măsură ce sunt construite, este suficient de accesibil pentru a facilita învățarea celor mai fundamentale concepte legate de circuitele logice. Datorită capacității de a construi circuite mai mari din subcircuite mai mici și de a trasa fascicule de fire cu o singură mișcare a mouse-ului. Logisim poate fi folosit (și este folosit) pentru a proiecta și simula întregi unități centrale (CPU-uri) în scopuri educaționale.

Studenții de la colegii și universități din întreaga lume utilizează Logisim în diverse scopuri, printre care:

- Un modul în cursurile generale de informatică
- O unitate în cursurile de organizare a calculatoarelor la nivelul al doilea de studiu
- Pe parcursul unui semestru întreg în cursurile superioare de arhitectură a calculatoarelor

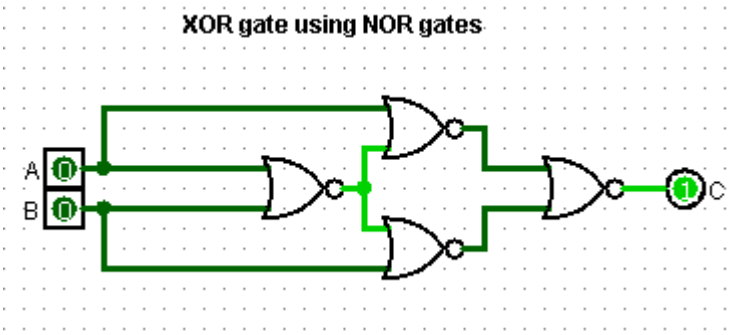
3. **Exercitii executate:**

E1: Implementați o poartă XOR cu două intrări folosind poarta NAND.



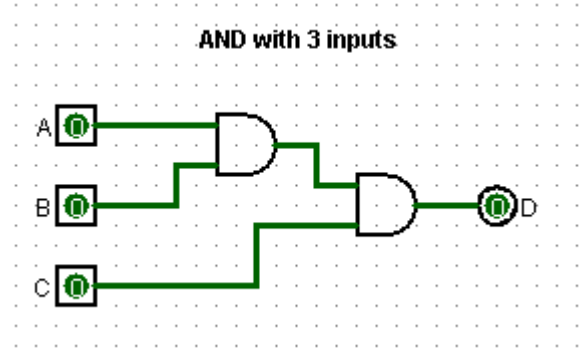
Acest circuit este un exemplu de cum se poate realiza un XOR gate folosind NAND gates. Intrările sunt marcate cu “A” și “B”, iar ieșirea este marcată cu “C”. Există patru NAND gates în total, aranjate astfel încât să realizeze funcția logica XOR.

E2: Implementați o poartă XNOR cu două intrări folosind poartă NOR



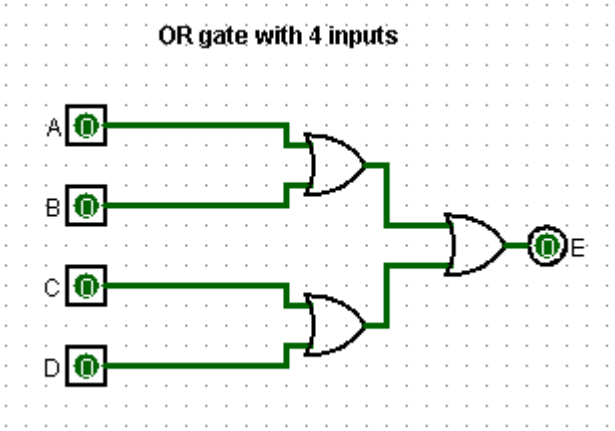
Acest circuit este un exemplu de cum se poate realiza o poartă XOR folosind porți NOR. Are două intrări, A și B, și o ieșire C. Funcționează astfel încât ieșirea C este adevărată (1) dacă una dintre intrările A sau B este adevărată (1), dar nu ambele.

E3: Implementați o poartă AND cu trei intrări folosind poarta AND.



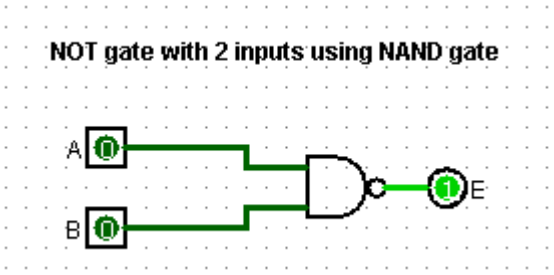
Acesta este un circuit logic AND cu trei intrări. Circuitul are trei intrări etichetate ca A, B și C și o ieșire D. Toate cele trei intrări trebuie să fie active (sau “true”) pentru ca ieșirea D să fie activată.

E4: Implementați o poartă OR cu patru intrări folosind poarta OR.



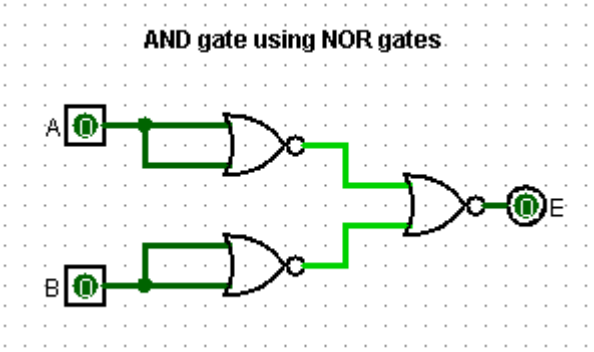
Acesta este un circuit logic care reprezintă o poartă OR cu 4 intrări. Intrările sunt etichetate ca A, B, C și D, iar fiecare intrare este conectată la poarta OR. Dacă cel puțin una dintre intrările A, B, C sau D este înaltă (ON), atunci ieșirea E va fi de asemenea înaltă (ON).

E5: Implementați o poartă NOT cu două intrări folosind poarta NAND.



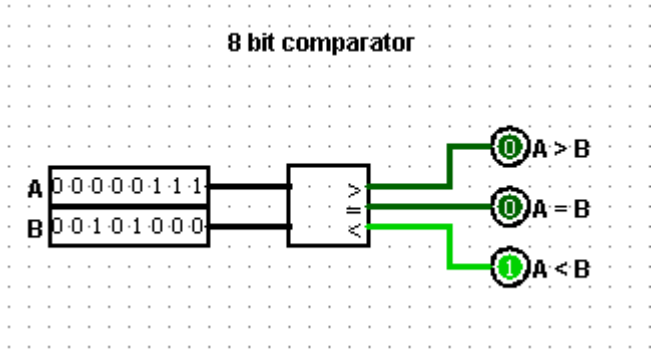
Configurează o poartă NAND cu o intrare legată la ieșirea sa pentru a inversa semnalul de intrare, creând efectiv o poartă NOT.

E6: Implementați o poartă AND cu două intrări folosind poartă NOR.



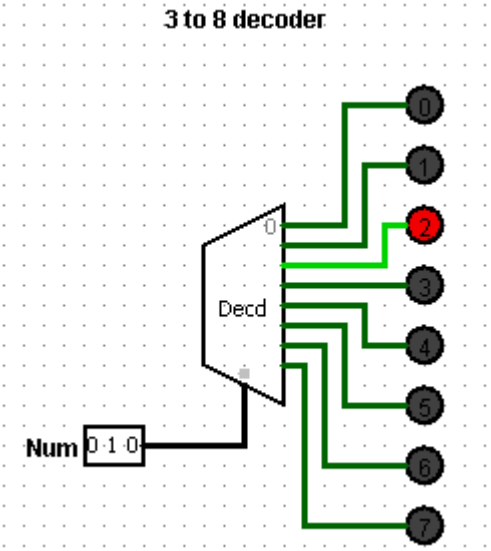
Configurează porți NOR într-o amenajare specifică pentru a obține operația logică AND, producând o ieșire adevărată numai când ambele intrări sunt adevărate.

M1: Implementați un comparator cu 8 biți folosind porți logice.



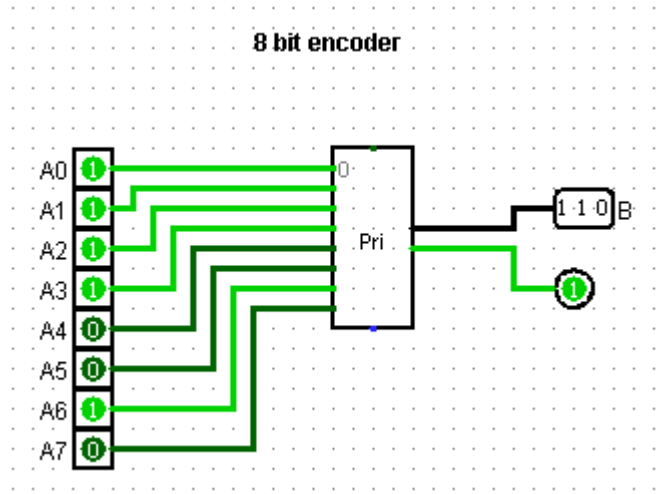
Acest circuit este un comparator de 8 biți care compară două numere binare de 8 biți și indică dacă un număr este mai mare, egal sau mai mic decât celălalt. În acest exemplu, două numere binare sunt introduse: A (00000111) și B (01010000). Există trei ieșiri: A > B, A = B și A < B, care devin active în funcție de comparația dintre cele două valori binare.

M2: Implementați un decodor cu 3 biți și 8 ieșiri.



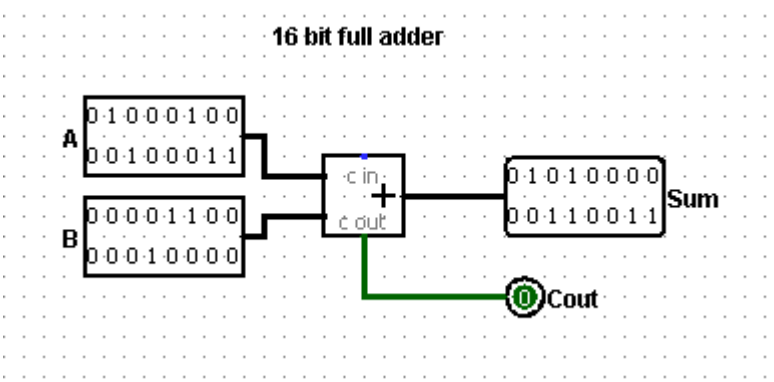
Acest circuit este un decodor de la 3 la 8. Înseamnă că are trei intrări binare și opt ieșiri, fiecare reprezentând o combinație unică a celor trei biți de intrare. Decodorul activează una dintre ieșirile sale în funcție de combinația binară pe care o primește.

M3: Implementați un codificator prioritar pentru 8 intrări, care să aibă prioritatea dată de valoarea intrărilor.



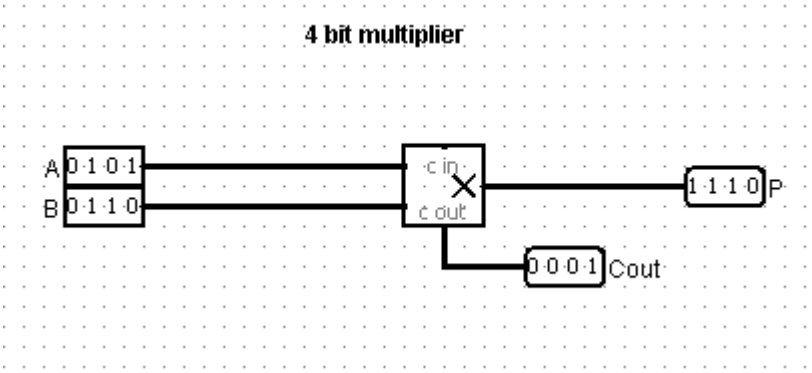
Acest circuit este un encoder de 8 biți. El convertește unul dintre cele 8 semnale de intrare într-un cod binar pe 3 biți la ieșire.

M4: Implementați un circuit de adunare completă (full-adder) cu 16 biți.



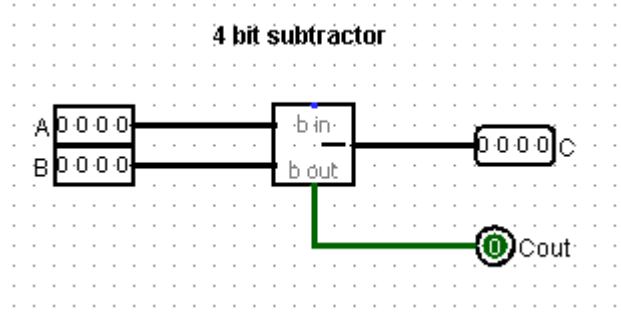
Ai dreptate, circuitul este un adunator complet de 16 biți. Acesta are două intrări, A și B, fiecare cu 16 biți de date. Există și o intrare “cin” pentru bitul de transport în. Rezultatul adunării este reprezentat în zona “Sum”, iar bitul de transport ieșit este marcat ca “Cout”.

M5: Implementați un circuit de înmulțire pentru 2 numere de 4 biți.



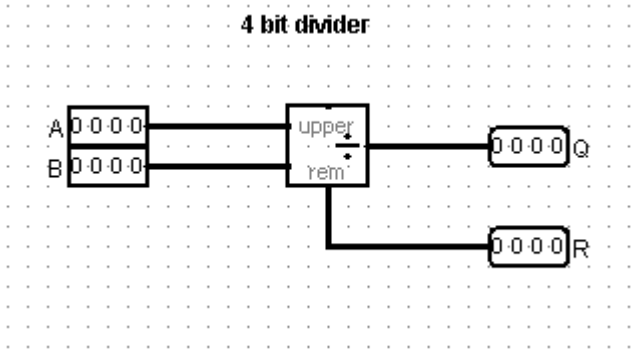
Acest multiplicator de 4 biți funcționează prin înmulțirea a două numere binare de 4 biți. Cele două numere sunt înmulțite împreună, iar rezultatul este reprezentat în zona “P” (produs). Rezultatul în binar este “1110”. Carry-out (Cout): Există și un bit de transport ieșit (Cout), care apare în cazul în care există un carry over după înmulțire. În acest caz, Cout este “0001”.

M6: Implementați un circuit de divizare pentru 2 numere de 4 biți.



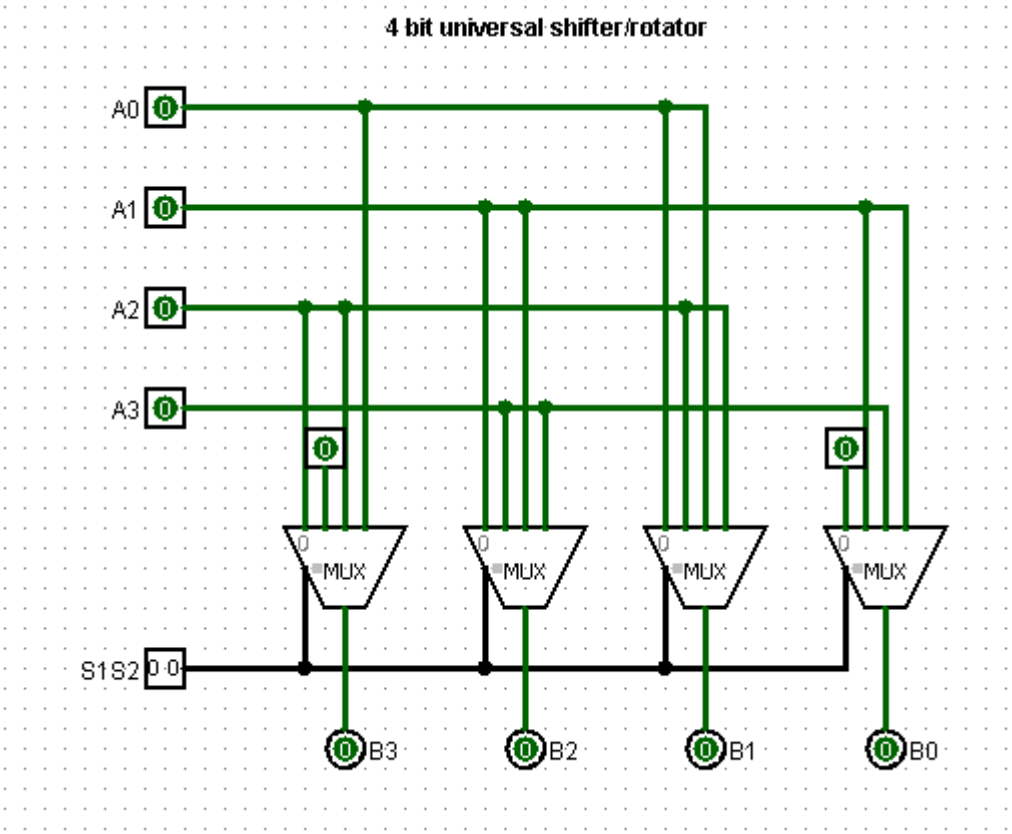
Imaginea prezintă o diagramă bloc a unui subtractor binar de 4 biți. Subtractoarele binare sunt utilizate pentru a scădea un număr binar din altul. Diagrama bloc evidențiază componentele principale ale subtractorului, dar nu detaliază modul de funcționare al fiecărei componente.

M9: Implementați un circuit care realizează operația de împărțire între două numere de 4 biți.



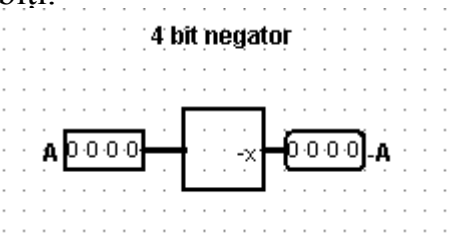
Un divizor de 4 biți este un circuit digital conceput să împartă un dividend binar (număr de intrare) la un divizor binar (un alt număr de intrare) și să furnizeze un cât (rezultat) binar și o rest (remnant) binară.
Imaginea reprezinta o diagramă bloc a unui divizor de 4 biți, dar îi lipsesc detaliile despre funcționarea internă a fiecărui bloc.

M10: Implementați un circuit care realizează operația de rotire dreapta pentru un număr de 4 biți.



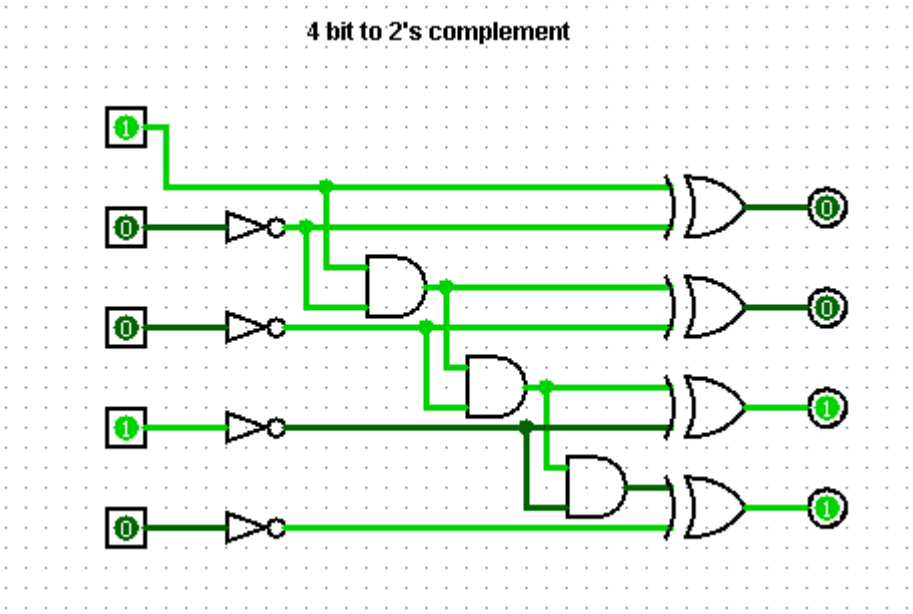
Circuitul dat reprezinta un universal shifter si rotator, care efectuează operațiunile de shift right/left și rotate right/left a unui numar de 4 biti folosind 4 multiplexoare.
În dependenta de inputul S1 si S2, el efectuează următoarele operații:
0 0 Shift left and fill with 0
0 1 Shift right and fill with 0
1 0 Rotate left
1 1 Rotate right

M11: Implementați un circuit care realizează operația de negare a unui număr de 4 biți.



Imaginea data prezintă un circuit negator de 4 biți, cunoscut și ca invertor de 4 biți. Circuitul are rolul de a inversa valorile fiecărui bit al unui număr binar de 4 biți.
Componentele circuitului:
4 porți NOT: Principalele componente ale circuitului sunt 4 porți logice NOT. Fiecare poartă NOT are o intrare (A) și o ieșire (Q).
Intrări (A0, A1, A2, A3): Circuitul primește un număr binar de 4 biți ca intrare, reprezentat de biții A0, A1, A2 și A3.
Ieșiri (Q0, Q1, Q2, Q3): Circuitul produce un număr binar de 4 biți ca ieșire, unde Q0, Q1, Q2 și Q3 reprezintă valorile negate ale biților de intrare corespunzători.

M12: Implementați un circuit care realizează operația de complementare la doi pentru un număr de 4 biți.



Imaginea prezintă un circuit pentru complemetarea la 2 a unui număr binar de 4 biți. Complementarea la 2 este o operație importantă în aritmetica binară, utilizată pentru a reprezenta numere negative sau pentru a efectua scăderi.

Funcționarea circuitului:

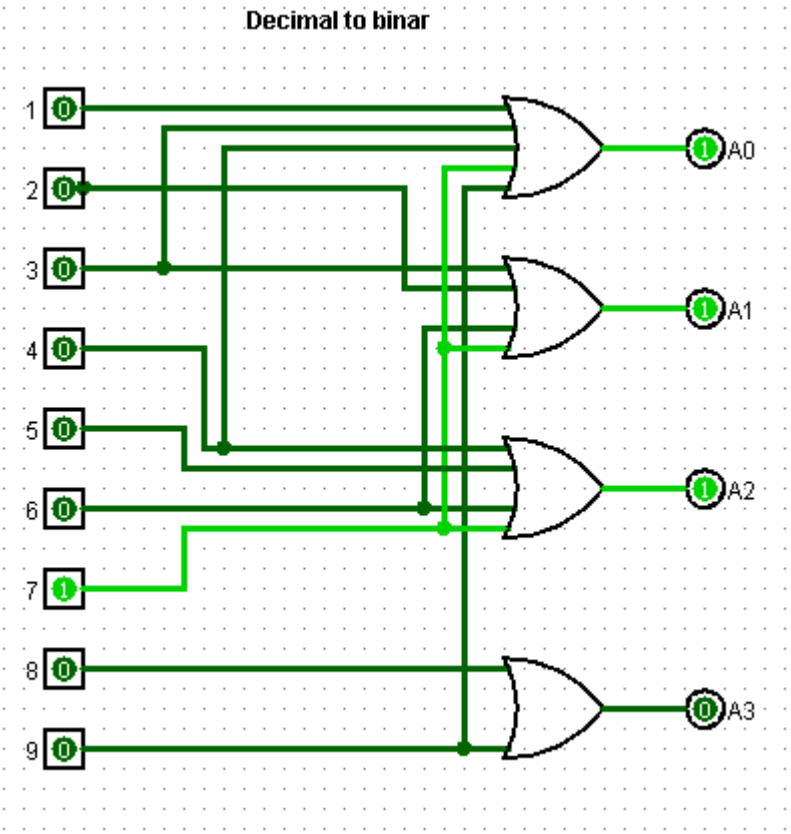
1. **Inversarea biților:** Primele 4 porți NOT inversează valorile biților de intrare, similar cu circuitul negator.
2. **Adunarea cu 1:** Fiecare poartă OR combină bitul negat cu un bit constant "1". Această operație are ca rezultat complementul la 1 al bitului de intrare.
3. **Complementul la 2:** Deoarece complementul la 2 este complementul la 1 plus 1, circuitul realizează automat această operație prin adunarea cu 1 la complementul la 1.

Să considerăm un număr binar de intrare A = 1011 (11 în zecimal).

- A0 = 1, deci Q0 = ~A0 + 1 = 0 + 1 = 1.
- A1 = 0, deci Q1 = ~A1 + 1 = 1 + 1 = 10.
- A2 = 1, deci Q2 = ~A2 + 1 = 0 + 1 = 1.
- A3 = 1, deci Q3 = ~A3 + 1 = 0 + 1 = 1.

Ieșirea circuitului va fi Q = 1110 (14 în zecimal), care este complementul la 2 al lui A.

M15: Implementați un circuit care realizează operația de conversie a unui număr de 4 biți din zecimal în binar.

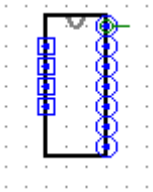


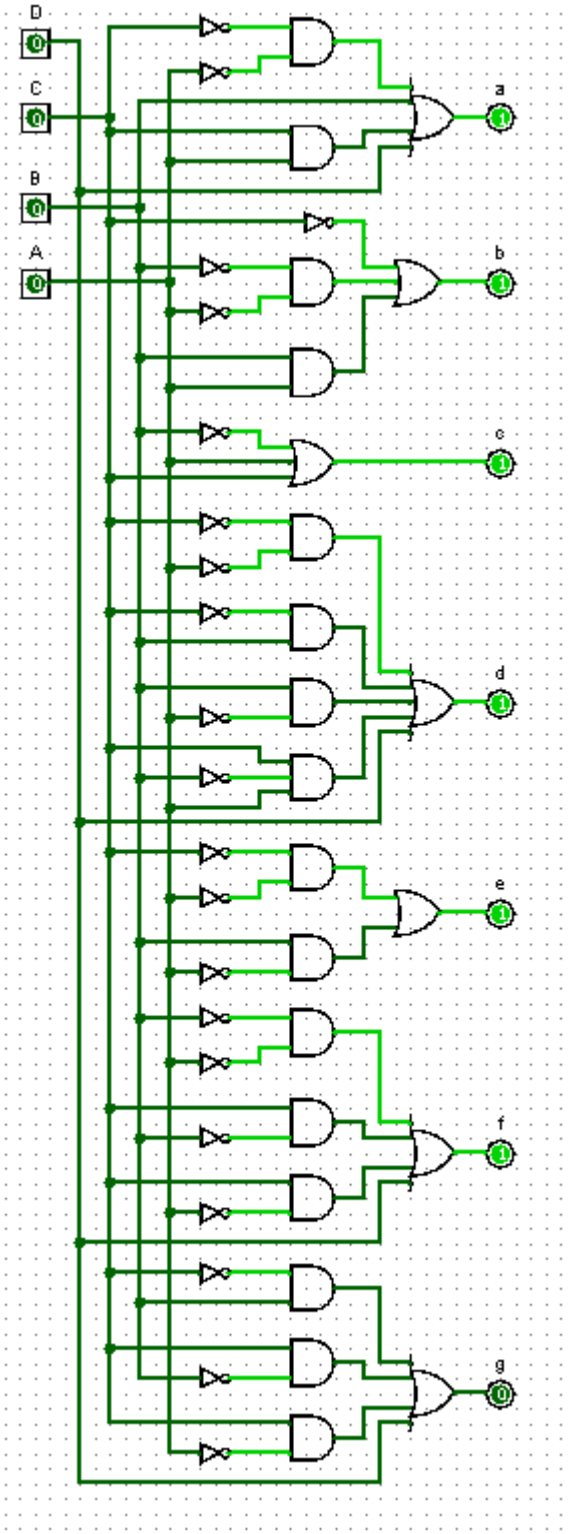
Imaginea prezintă un circuit pentru conversia unui număr zecimal (decimal) în binar (binar) pe 4 biți. Circuitul utilizează o combinație de porți logice și elemente de memorie pentru a realiza conversia.

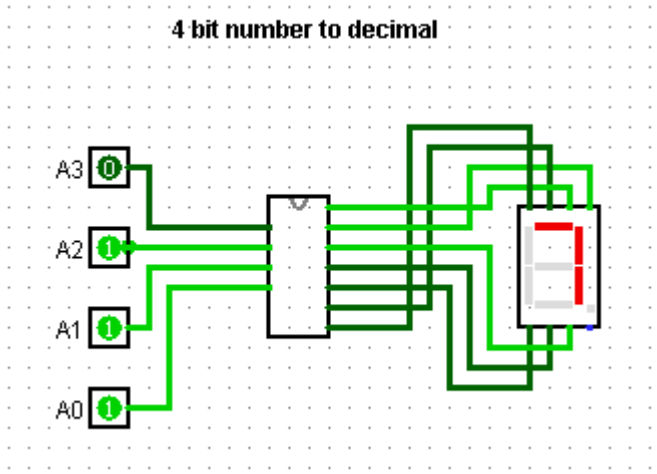
Funcționarea circuitului:

1. **Decodificarea zecimală:** Fiecare decodor BCD-la-binare primește un bit zecimal de la intrare (D0, D1, D2, D3) și generează 4 biți de ieșire. Acești 4 biți corespund codului binar echivalent pentru cifra zecimală respectivă.
2. **Stocarea biților binari:** Registrele D stochează valorile biților binari calculați de decodoare.
3. **Formarea numărului binar:** Biții stocați în registre sunt combinați pentru a forma un număr binar de 4 biți la ieșirea circuitului (Q0, Q1, Q2, Q3).

M16: Implementați un circuit care realizează operația de conversie a unui număr de 4 biți din binar în zecimal.




D
C
B
A
a
b
c
d
e
f
g



Imaginile prezintă un circuit pentru conversia unui număr binar (binar) de 4 biți în zecimal (decimal). Circuitul utilizează o combinație de porți logice și elemente de memorie pentru a realiza conversia.

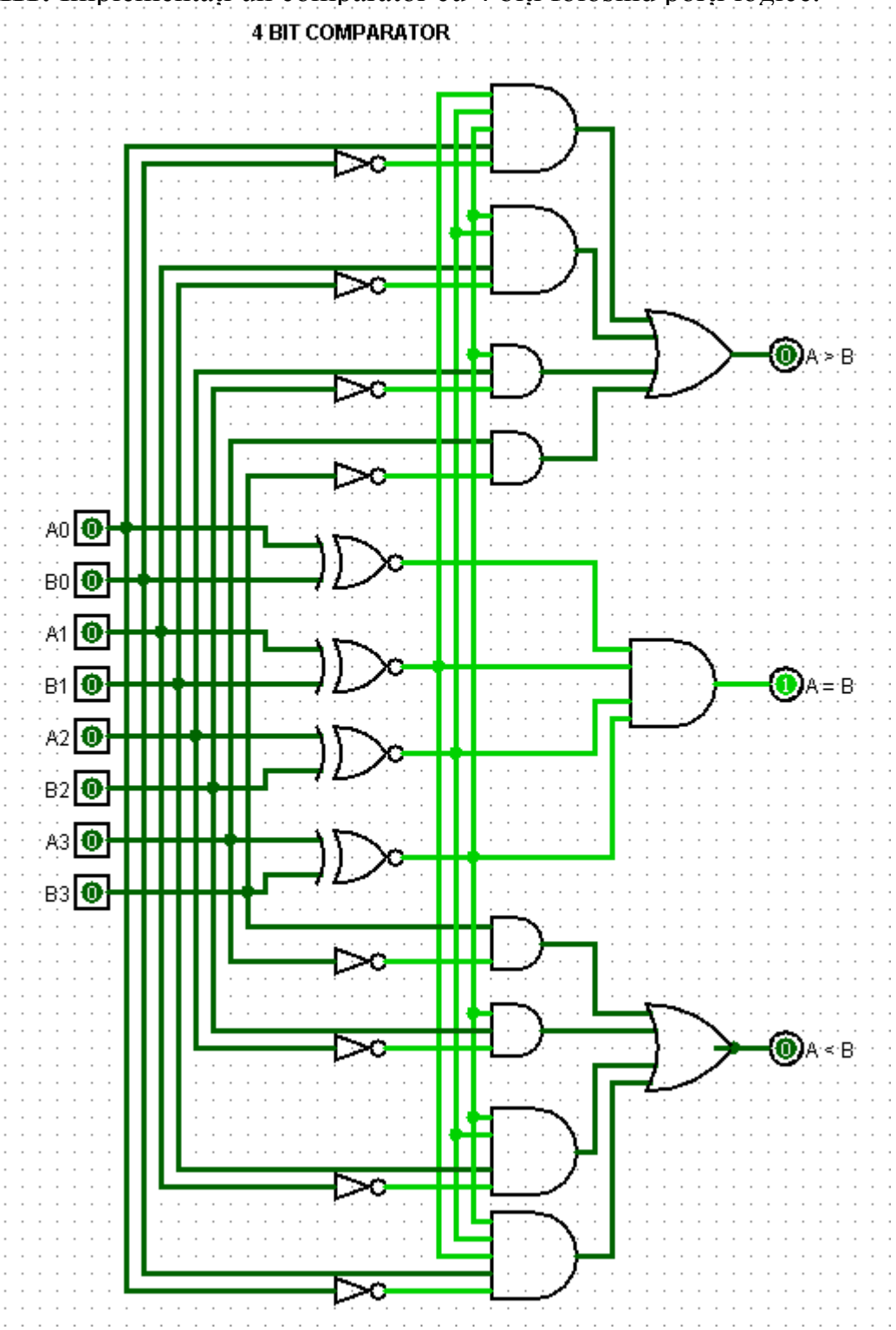
Componentele circuitului:

- **4 decodoare 2-la-4:** Circuitul utilizează 4 decodoare logice 2-la-4 pentru a converti fiecare bit binar de intrare în 4 biți de ieșire, unde doar o linie de ieșire este activă (1) în funcție de bitul de intrare.
- **4 ponderi (rezistențe):** Fiecare decodor are asociată o pondere specifică (2^3, 2^2, 2^1, 2^0) sub formă de rezistență, care determină contribuția fiecărui bit la valoarea zecimală finală.
- **Sumator:** Un sumator binar adună valorile ponderate generate de decodoare pentru a obține numărul zecimal final.
- **Ieșire (Z):** Ieșirea circuitului afișează numărul zecimal echivalent al numărului binar de intrare.

Funcționarea circuitului:

1. **Decodificarea binară:** Fiecare decodor 2-la-4 primește un bit binar de la intrare (A0, A1, A2, A3) și activează una dintre cele 4 linii de ieșire în funcție de bitul respectiv.
2. **Ponderarea biților:** Valorile de pe liniile active ale decodoarelor sunt ponderate de către rezistențele asociate, reflectând puterile lui 2 (2^3, 2^2, 2^1, 2^0).
3. **Sumarea ponderată:** Sumatorul binar adună valorile ponderate ale biților binari, rezultând numărul zecimal echivalent.
4. **Afișarea rezultatului:** Ieșirea Z afișează numărul zecimal final.

H1: Implementați un comparator cu 4 biți folosind porți logice.



Imaginea prezintă un circuit comparator de 4 biți, utilizat pentru a compara două numere binare de 4 biți. Circuitul analizează biții corespunzători din ambele numere și determină dacă sunt egali, dacă primul număr este mai mare sau dacă al doilea număr este mai mare.

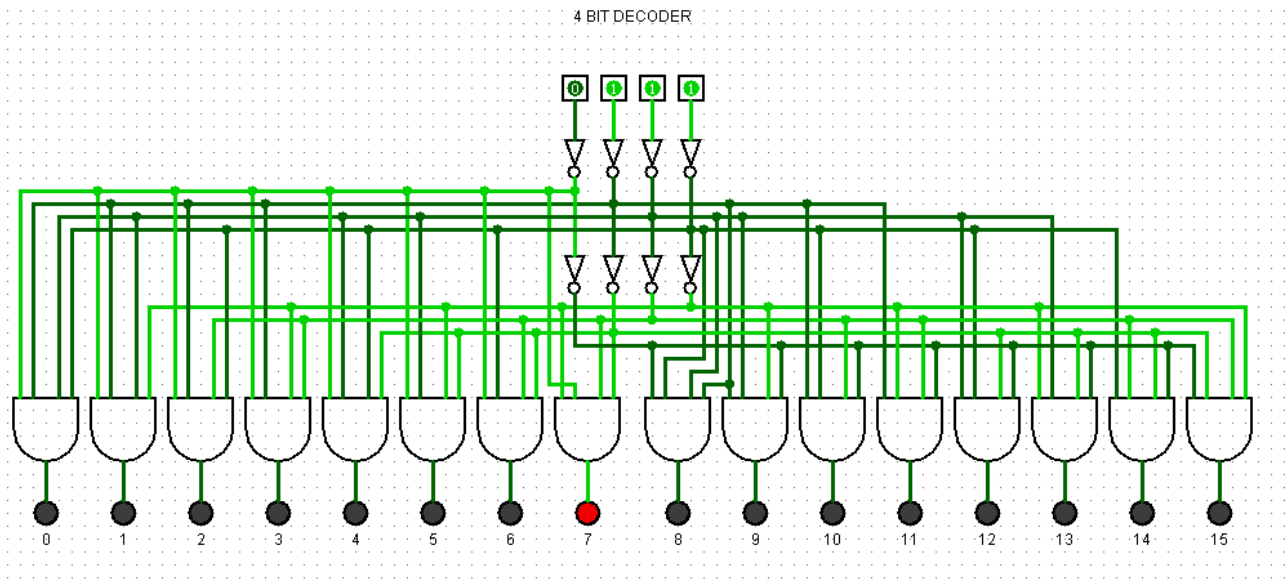
Componentele circuitului:

- **4 porți XOR:** Fiecare bit al ambelor numere de intrare (A și B) este comparat cu ajutorul unei porți XOR. O ieșire "1" la poarta XOR indică biți diferiți, iar o ieșire "0" indică biți identici.
- **4 porți AND:** Porțile AND combină ieșirile porților XOR pentru a determina relația dintre cele două numere.
- **3 porți OR:** Porțile OR combină rezultatele porților AND pentru a genera semnale de ieșire finale.
- **Intrări (A0, A1, A2, A3, B0, B1, B2, B3):** Circuitul primește două numere binare de 4 biți ca intrare, A și B, reprezentate de biții A0-A3 și B0-B3 respectiv.
- **Ieșiri (Egalitate, A_mai_mare, B_mai_mare):** Circuitul are 3 ieșiri:
  - **Egalitate:** Indică dacă cele două numere de intrare sunt identice.
  - **A_mai_mare:** Indică dacă numărul A este mai mare decât numărul B.
  - **B_mai_mare:** Indică dacă numărul B este mai mare decât numărul A.

Funcționarea circuitului:

1. **Compararea biților:** Porțile XOR compară biții corespunzători din A și B.
2. **Determinarea relației:** Porțile AND combină ieșirile porților XOR pentru a determina relația dintre cele două numere:
   - **A0 și B0 identici:** AND(A0, ~B0) = 1 dacă A0 = B0.
   - **A1 și B1 identici:** AND(A1, ~B1) = 1 dacă A1 = B1.
   - **Toți biții identici:** AND(Egalitate_parțială1, Egalitate_parțială2, Egalitate_parțială3) = 1 dacă toți biții sunt identici.
   - **A mai mare:** AND(A_mai_mare_parțială1, A_mai_mare_parțială2, A_mai_mare_parțială3) = 1 dacă A este mai mare.
   - **B mai mare:** AND(B_mai_mare_parțială1, B_mai_mare_parțială2, B_mai_mare_parțială3) = 1 dacă B este mai mare.
3. **Generarea semnalelor de ieșire:** Porțile OR combină rezultatele intermediare pentru a genera semnalele de ieșire finale:
   - **Egalitate:** OR(Egalitate_parțială1, Egalitate_parțială2, Egalitate_parțială3) = 1.
   - **A_mai_mare:** OR(A_mai_mare_parțială1, A_mai_mare_parțială2, A_mai_mare_parțială3) = 1.
   - **B_mai_mare:** OR(B_mai_mare_parțială1, B_mai_mare_parțială2, B_mai_mare_parțială3) = 1.

H2: Implementați un decodor cu 4 biți și 16 ieșiri.



Imaginea prezintă un circuit decoder de 4 biți. Un decoder convertește un cod binar de intrare pe 4 biți în 16 linii de ieșire decodate, unde doar o singură linie este activă (1) la un moment dat, corespunzând codului de intrare specific.

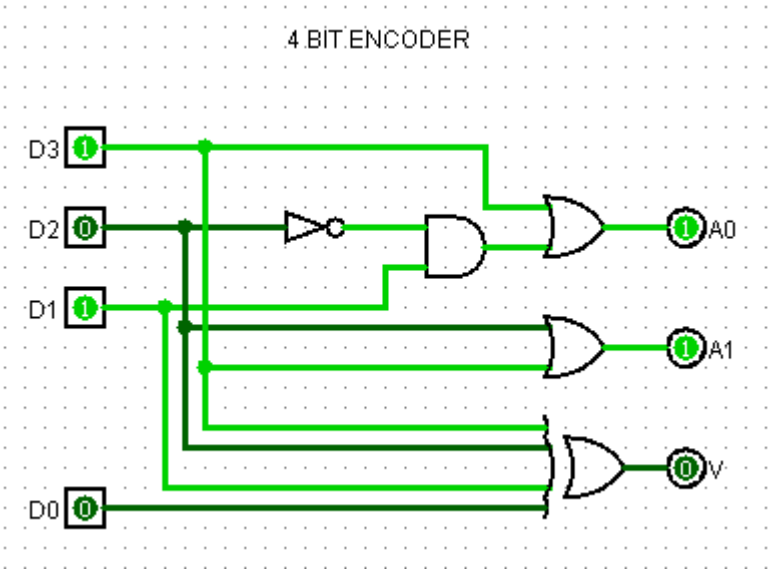
Componentele circuitului:

- **4 porți AND:** Circuitul utilizează 4 porți logice AND pentru a decoda fiecare dintre cele 4 biți de intrare.
- **16 decodoare 2-la-4:** Fiecare poartă AND este conectată la un decodor 2-la-4, care generează 4 linii de ieșire din 2 linii de intrare.
- **Intrări (A0, A1, A2, A3):** Circuitul primește un număr binar de 4 biți ca intrare, reprezentat de biții A0, A1, A2 și A3.
- **Ieșiri (Y0, Y1, Y2, ..., Y15):** Circuitul are 16 ieșiri, unde doar o singură ieșire va fi activă (1) pentru codul binar de intrare specific.

Funcționarea circuitului:

1. **Decodificarea biților:** Fiecare poartă AND decodifică un bit de intrare specific:
   - A0 este decodificat de AND1, activând Y0 sau Y8.
   - A1 este decodificat de AND2, activând Y1 sau Y9.
   - A2 este decodificat de AND3, activând Y2 sau Y10.
   - A3 este decodificat de AND4, activând Y3 sau Y11.
2. **Selectarea ieșirii:** Decodoarele 2-la-4 combină biții decodificați pentru a selecta o singură ieșire activă:
   - Y0 este activată dacă A0 = 0 și A1 = 0.
   - Y1 este activată dacă A0 = 0 și A1 = 1.
   - ...
   - Y15 este activată dacă A3 = 1 și A1 = 1.

H3: Implementați un codificator prioritar pentru 4 intrări, care să aibă prioritatea dată de valoarea intrărilor.



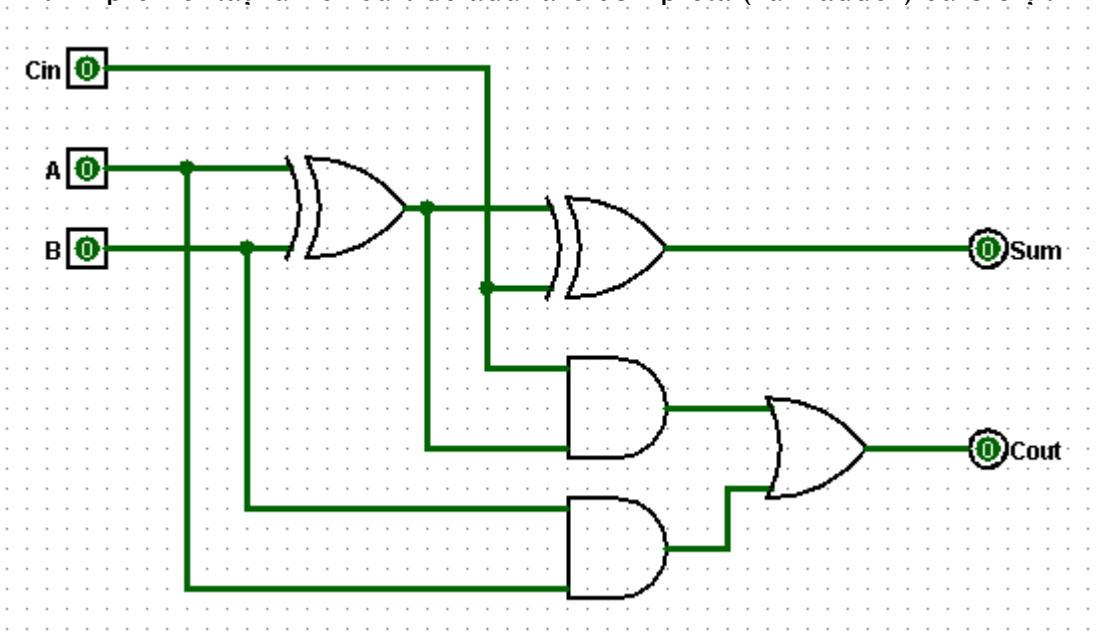
Imaginea prezintă un circuit encoder prioritar de 4 biți. Un encoder convertește un număr binar de 4 biți de intrare în codul său zecimal echivalent, dar cu o particularitate: prioritatea este acordată biților cu indici superiori.

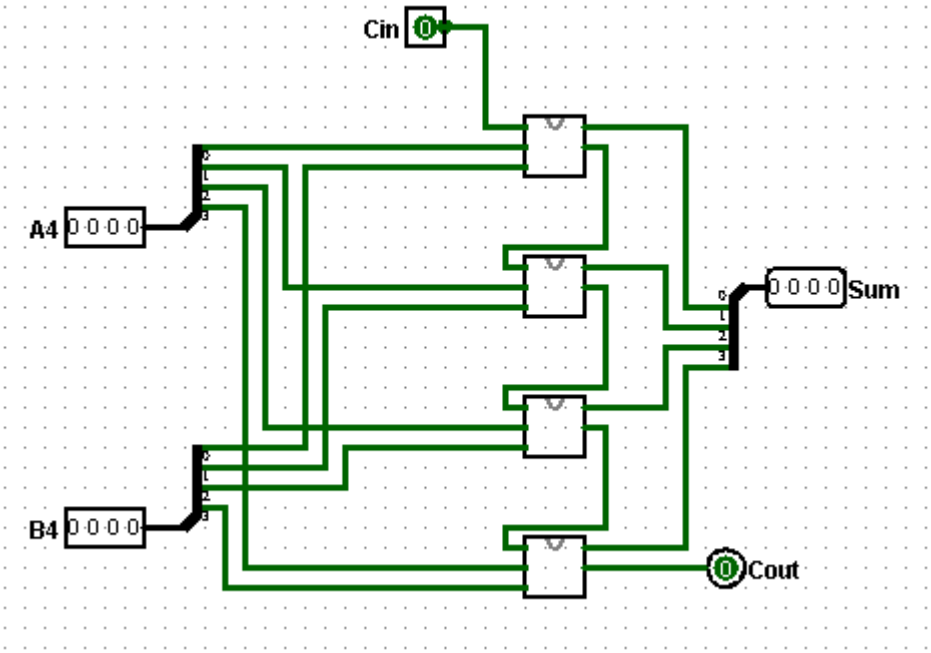
Componentele circuitului:

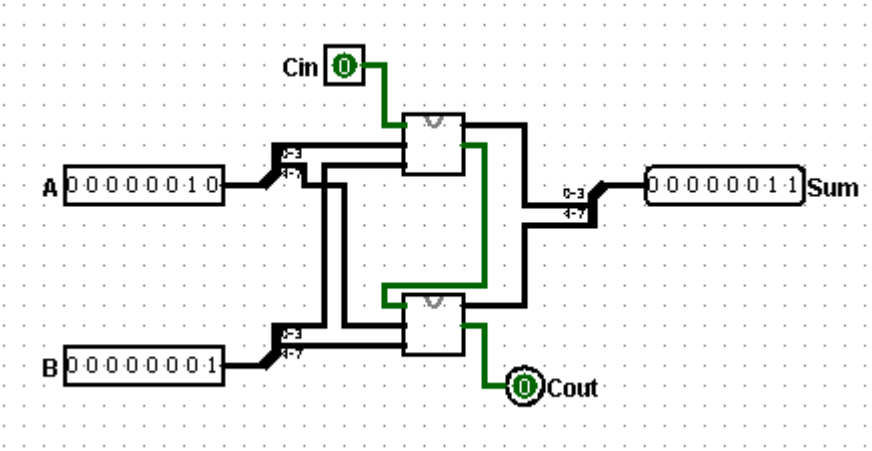
- **4 porți OR:** Circuitul utilizează 4 porți logice OR pentru a combina biții de intrare cu priorități diferite.
- **4 decodoare 2-la-4:** Fiecare poartă OR este conectată la un decodor 2-la-4, care generează 4 linii de ieșire din 2 linii de intrare.
- **Intrări (D0, D1, D2, D3):** Circuitul primește un număr binar de 4 biți ca intrare, reprezentat de biții D0, D1, D2 și D3.

- **Ieșiri (Y0, Y1, Y2, Y3):** Circuitul are 4 ieșiri care codifică zecimal numărul binar de intrare, luând în considerare prioritatea biților.

H4: Implementați un circuit de adunare completă (full-adder) cu 8 biți.

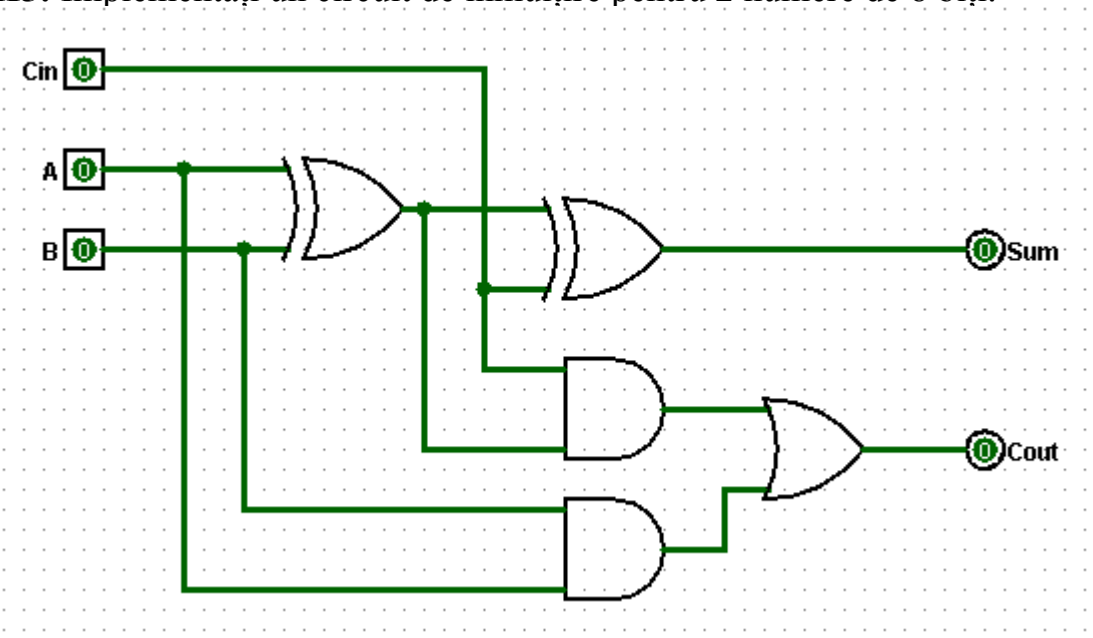


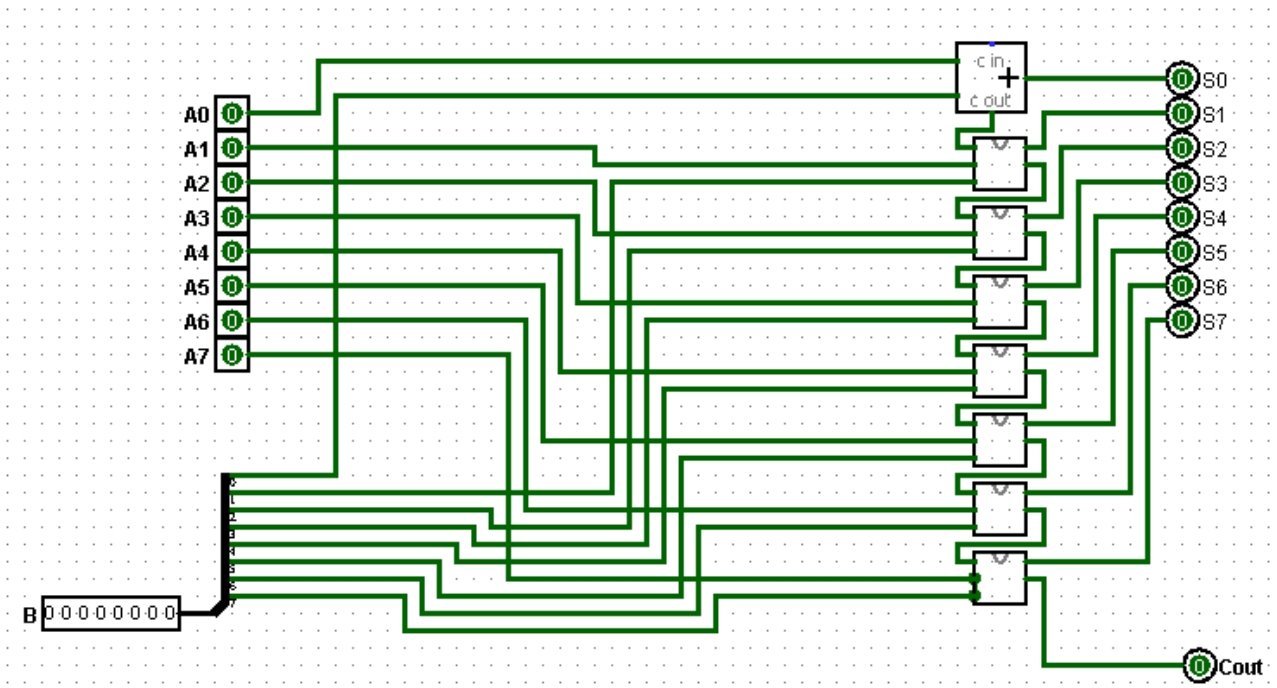


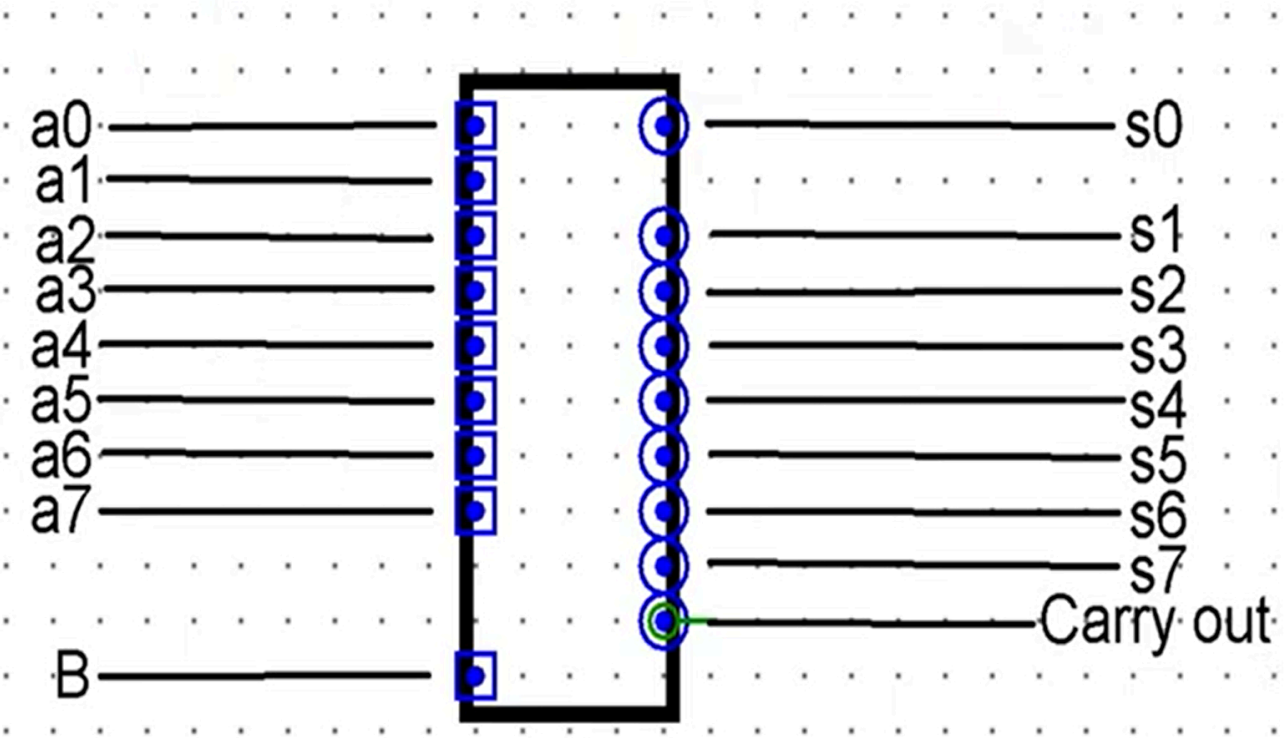


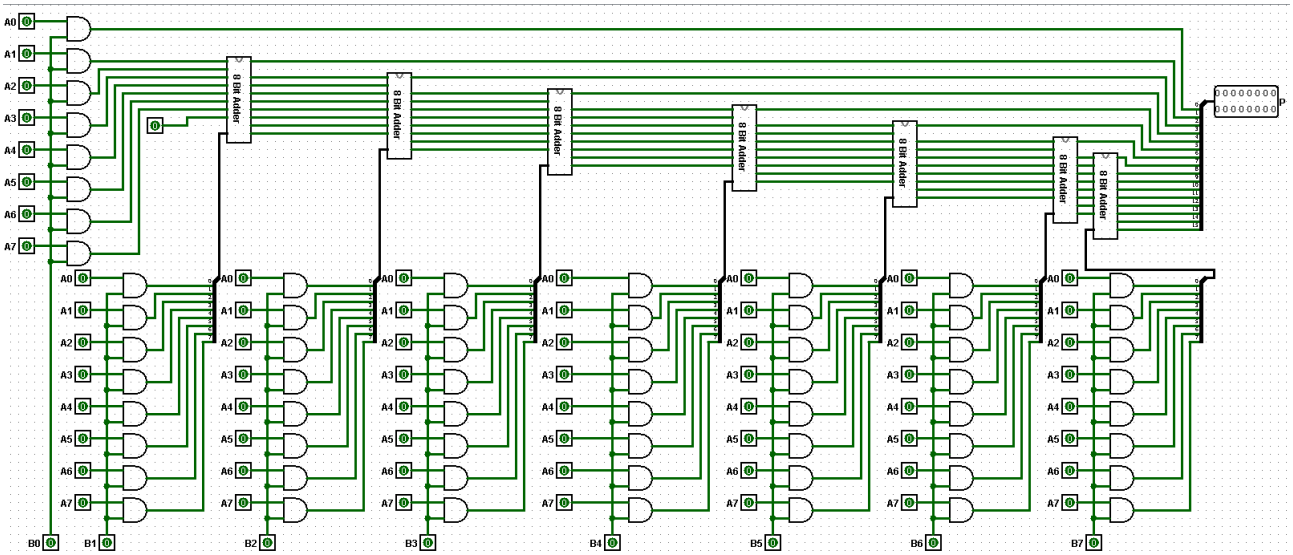
Un full adder de 8 biti consta din 2 4-bit full adders, care la randul lor constau din 2 full adders si care la randul lor constau din 2 half adders. Un sumator complet de 8 biți este un circuit logic care adună două numere reprezentate pe 8 biți și un bit de transport (carry) de intrare. Acesta generează un rezultat de 8 biți și un bit de transport de ieșire, care indică dacă adunarea a produs un depășire a capacității de stocare de 8 biți.

H5: Implementați un circuit de înmulțire pentru 2 numere de 8 biți.




A0
A1
A2
A3
A4
A5
A6
A7
B 00000000
c in
c out
S0
S1
S2
S3
S4
S5
S6
S7
Cout


a0
a1
a2
a3
a4
a5
a6
a7
B
s0
s1
s2
s3
s4
s5
s6
s7
Carry out



Imaginea trimisă prezintă un circuit înmulțitor de 8 biți care utilizează 7 blocuri de adunători compleți de 8 biți pentru a realiza înmulțirea a două numere binare de 8 biți. Circuitul se bazează pe algoritmul de înmulțire manuală și utilizează adunări succesive pentru a obține produsul final.

4. **Concluzie**

În concluzie, laboratorul de Arhitecturi de Calculatoare a fost un succes remarcabil, oferindu-ne o înțelegere profundă a modului în care funcționează operațiile cu numere de 4 și 8 biți. Prin intermediul acestui curs, am acumulat abilități esențiale în efectuarea operațiilor de adunare, înmulțire și conversie între reprezentările binare și zecimale ale numerelor. De asemenea, am avut oportunitatea de a explora și de a lucra cu diverse blocuri de circuite precum multiplexoarele, decodoarele și encodoarele, consolidand astfel cunoștințele teoretice cu experiență practică în domeniul arhitecturii de calculatoare. Această experiență ne-a pregătit în mod semnificativ pentru a aborda și rezolva probleme mai complexe în domeniul informaticii și al ingineriei electronice.