Ministerul Educaţiei și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică

Laboratory work 3:

Elaborat:
st. gr. FAF-221 Cuzmin Simion

Verificat:

asist. Univ. Voitcovschi Vladislav

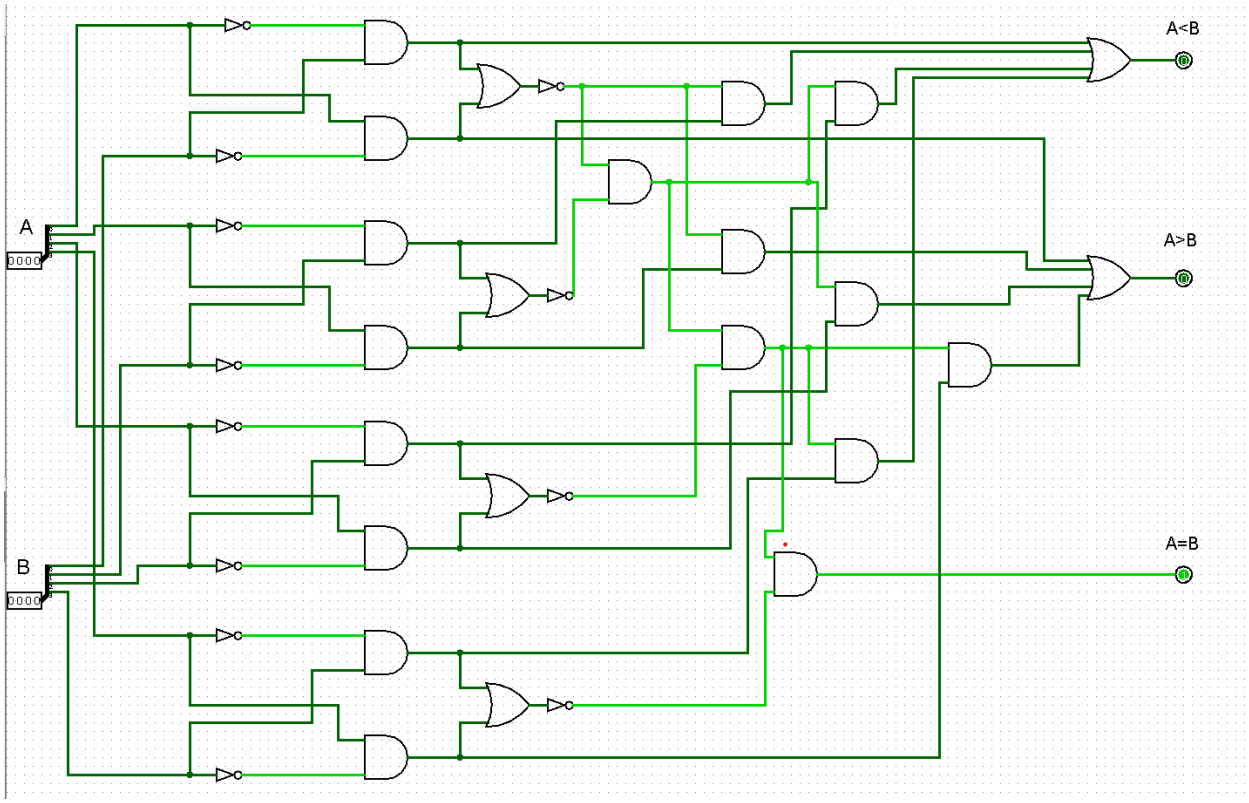
Scopul lucrării:

Scopul acestei lucrări este să mă familiarizeze cu conceptele de bază ale proiectării și implementării circuitelor digitale folosind Logisim. Prin exercițiile practice propuse, intenția este să învăț cum să utilizez diverse componente logice pentru a construi circuite digitale funcționale și să dobândesc competențe în simularea acestora.

Exercitiu 1H:

Implementați un comparator cu 4 biți folosind porți logice

Rezultatul:



Exercițiul propus vizează implementarea unui comparator cu 4 biți folosind porți logice. Acesta compară două numere de 4 biți și determină relația dintre ele (egalitate, numărul unu mai mare decât numărul doi sau invers).

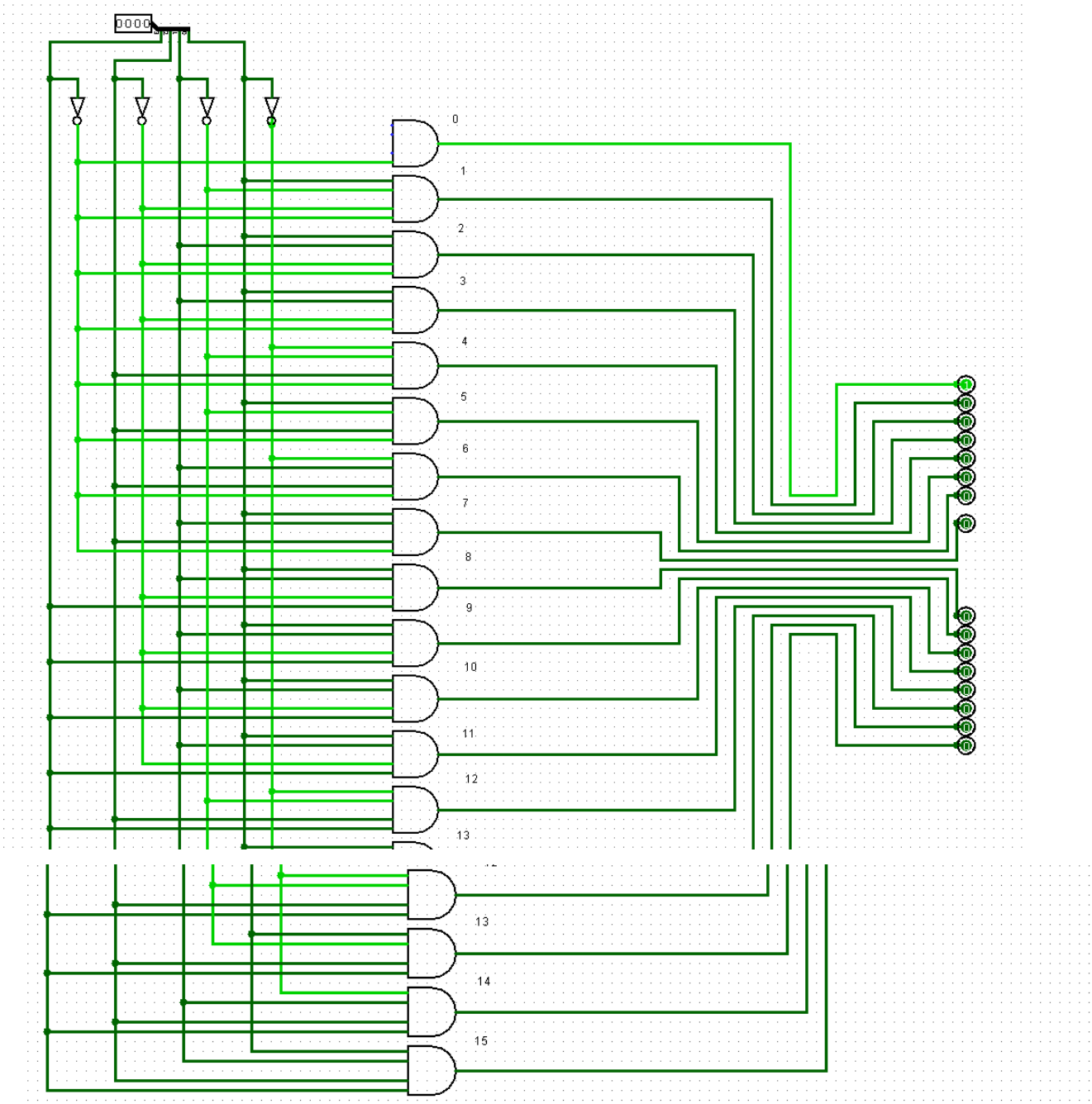
Pentru a realiza acest lucru, putem împărți problema în mai multe etape și folosi diverse porți logice pentru a efectua operațiile necesare.

1) Intrări: Două grupuri de patru biți fiecare, reprezentând cele două numere pe care dorim să le comparăm.
2) Splitter: Fiecare grup de patru biți este împărțit în patru biți individuali pentru a putea fi procesați separat.
3) Not-uri: Pentru a determina negația fiecărui bit, adică pentru a crea complementul fiecărui bit.
4) And-uri: Fiecare bit este comparat între cele două numere folosind porți AND pentru a obține rezultatele individuale ale comparării.
5) Or-uri și NOT-uri suplimentare: Rezultatele comparării sunt combinate pentru a obține rezultatele finale. Sunt folosite porți OR pentru a detecta dacă un număr este mai mare decât celălalt sau dacă sunt egale. De asemenea, se pot folosi NOT-uri pentru a inverte rezultatele pentru a obține corect relațiile de ordin.
6) Ieșire: Rezultatul final indică dacă cele două numere sunt egale sau care dintre ele este mai mare.
Prin combinarea acestor elemente, se realizează un circuit logic care compară cele două numere și determină relația dintre ele.

Exercitiu 2H:

Implementați un decodor cu 4 biți și 16 ieșiri

Rezultatul:



Un decodor cu 4 biți și 16 ieșiri este un circuit logic care primește o intrare de 4 biți și activează una dintre cele 16 ieșiri în funcție de valoarea binară a intrării.

1) Intrarea cu 4 biți: Primește un grup de 4 biți ca intrare. Acest grup de biți reprezintă valoarea pe care dorim să o decodăm.

2) Splitter 4x4: Acesta împarte grupul de 4 biți în 4 biți individuali, astfel încât fiecare bit poate fi procesat separat.

3) Not-uri (optional): Pentru a asigura că toate combinările sunt acoperite, este posibil să fie necesare unele operații NOT pentru a completa funcționarea circuitului. În funcție de implementare, acestea pot fi sau nu necesare.

4) And-uri: Se folosesc 16 porți AND pentru a compara fiecare bit de intrare cu o combinație specifică de biți. Fiecare ieșire a splitter-ului este conectată la toate cele 16 porți AND. Astfel, fiecare combinație de biți de intrare activează doar o singură dintre cele 16 porți AND.

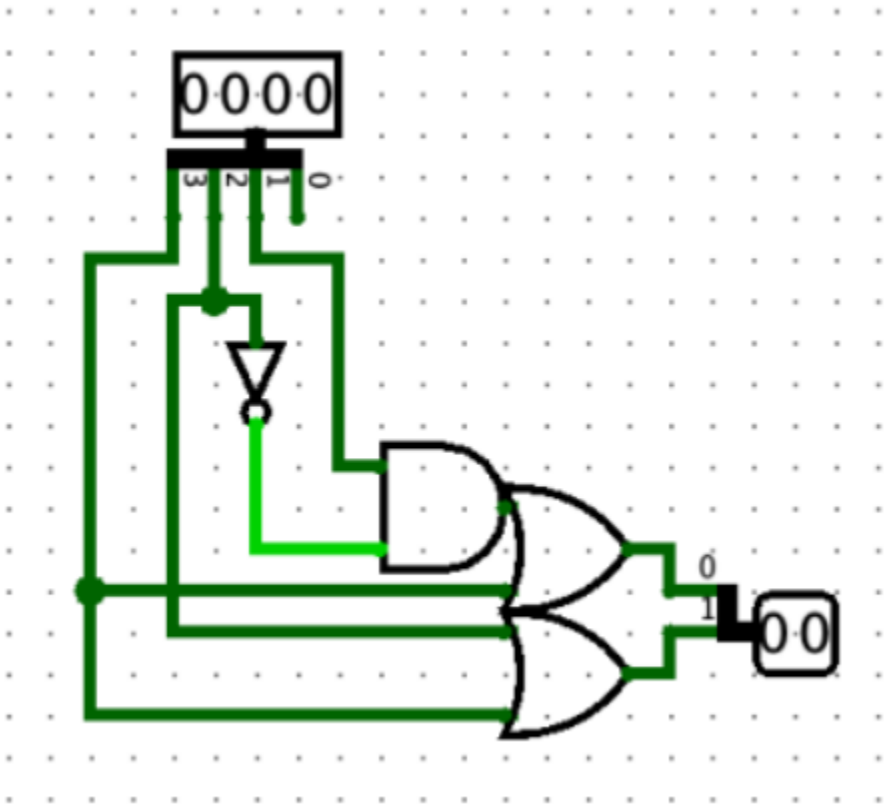
5) Ieșiri: Fiecare porți AND este asociată cu o ieșire a decodorului. Prin activarea unei porți AND specifice, se activează și ieșirea corespunzătoare a decodorului.

În esență, fiecare combinație posibilă de biți de intrare activează una dintre cele 16 ieșiri ale decodorului, făcând posibilă decodarea valorii de intrare în funcție de configurarea sa binară.

Exercitiu 3H:

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

Rezultatul:



Un codificator de prioritate este un circuit logic combinațional care codifică mai multe linii de intrare într-o reprezentare binară, luând în considerare prioritatea valorilor de intrare. În acest caz, dorim să implementăm un codificator de prioritate pentru 4 intrări, unde intrarea cu cea mai mare valoare are cea mai mare prioritate.

Funcționalitatea circuitului este de a detecta intrarea activă cu cea mai mare prioritate și de a produce o ieșire binară care reprezintă poziția sa (valoarea codificată). Dacă mai multe intrări sunt active în același timp, cea cu cea mai mare prioritate (cea mai mare valoare) este codificată.

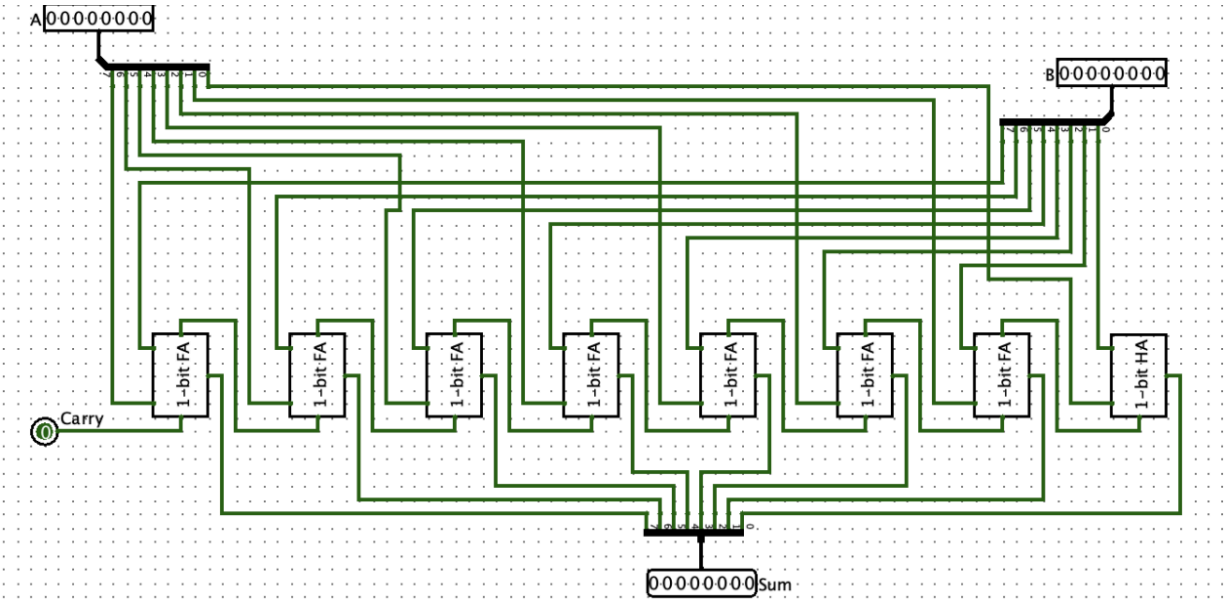
Pentru a implementa acest circuit, putem compara intrările în ordinea priorității și codifica poziția intrării active cu cea mai mare prioritate. Acest lucru poate fi realizat folosind o combinație de porți logice precum AND, OR și NOT. Prin analizarea valorilor de intrare și a

priorității lor, putem determina expresiile logice corespunzătoare pentru a implementa codificatorul de prioritate.

Exercitiu 4H:

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

Rezultatul:



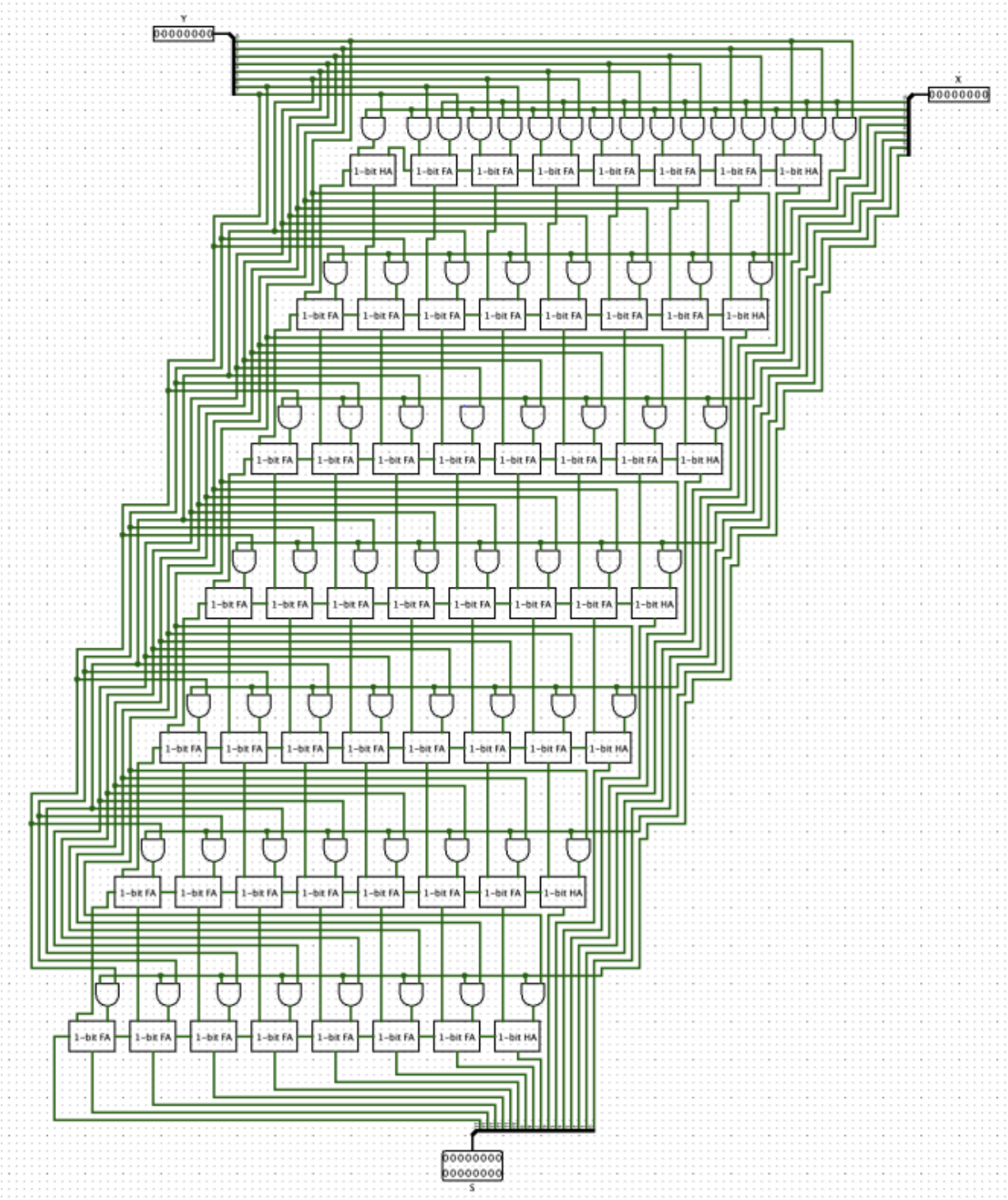
Pentru a implementa acest lucru, va trebui să combinați opt adunătoare complete împreună. Fiecare adunător complet primește trei intrări: două biți de adunat (A și B) și o intrare de transport (C_in), și produce două ieșiri: o sumă (S) și un transport-ieșire (C_out).
Iată cum puteți implementa acest lucru:
Conectați transportul-ieșire (C_out) al fiecărui adunător complet la transportul-intrare (C_in) al următorului adunător complet.
Conectați cei mai puțin semnificativi biți (LSB-uri) ai numerelor pe care doriți să le adunați (A și B) la cele mai puțin semnificative intrări (A0 și B0) ale primului adunător complet.
Conectați cei mai semnificativi biți (MSB-uri) ai numerelor pe care doriți să le adunați la intrarea de transport (C_in) a primului adunător complet. Cel mai semnificativ bit (MSB) al

rezultatului (transportul-ieșire al ultimului adunător complet) va fi transportul-ieșirea întregii adunări de 8 biți.
Această configurație formează un circuit de adunător complet de 8 biți, capabil să adune două numere binare de 8 biți împreună cu posibilitatea de a trece la următorul bit dacă suma depășește 1.
Fiecare adunător complet din circuit calculează suma și transportul-ieșirea pe baza intrărilor sale, iar transportul-ieșirea de la un adunător complet alimentează transportul-intrarea următorului, adăugând efectiv fiecare bit al celor două numere împreună cu orice transport de la adunarea anterioară.

Exercitiu 5H:

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

Rezultatul:



Un circuit de multiplicare pentru două numere de 8 biți este un tip de circuit logic combinațional care efectuează operația de înmulțire asupra două numere binare de 8 biți.

Pentru a construi acest circuit, este necesar să utilizăm o combinație de porți logice și aritmetice de bază, precum porți AND, OR și XOR, împreună cu alte circuite auxiliare.

Scopul circuitului este să înmulțească cele două numere binare de 8 biți și să genereze o ieșire de 16 biți care reprezintă rezultatul înmulțirii. Această ieșire poate conține produsul a două numere de 8 biți, care variază în intervalul de la 0 la 255 * 255 = 65025.

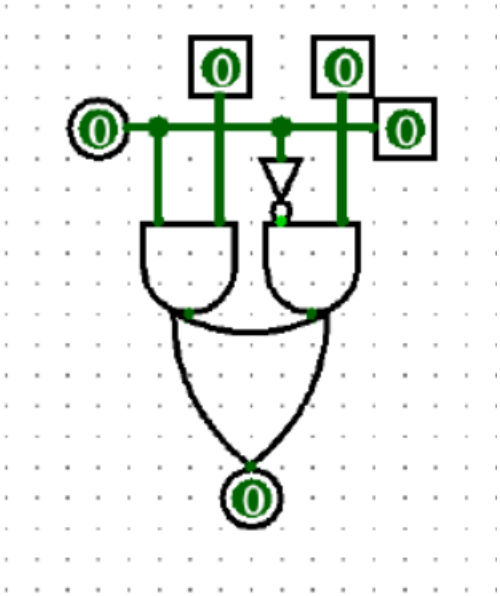
Circuitul este construit pe mai multe etape, în care se generează produse parțiale și apoi sunt adunate pentru a obține rezultatul final. Fiecare etapă implică deplasarea unuia dintre factorii de multiplicare în diverse poziții și efectuarea operațiilor logice pentru a genera produse parțiale, care sunt ulterior adunate folosind circuite de adunare.

În ansamblu, circuitul de multiplicare pentru două numere de 8 biți necesită o serie de operații logice și aritmetice pentru a calcula rezultatul în mod eficient și precis.

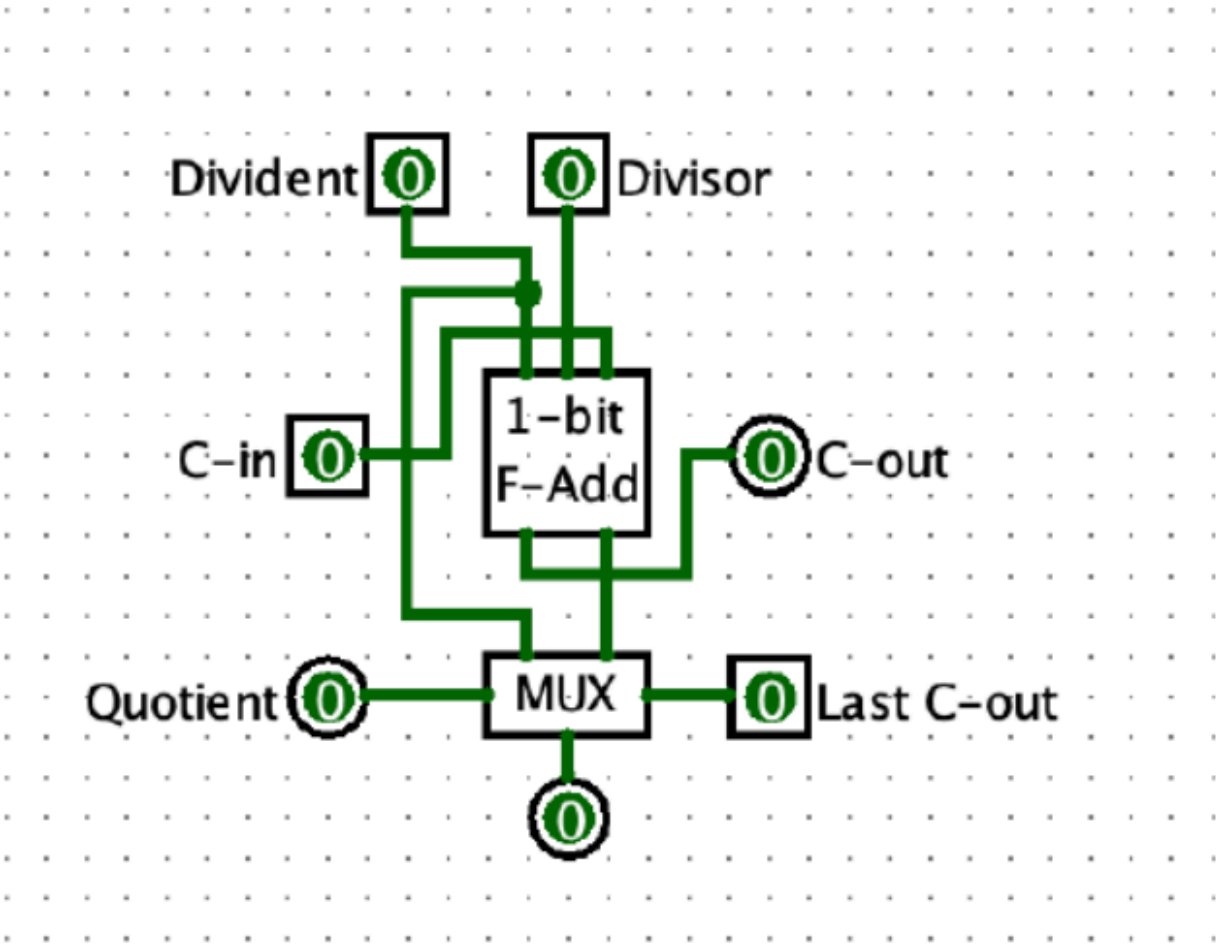
Exercitiu 6H:

Implementați un circuit de divizare pentru 2 numere de 8 biți.

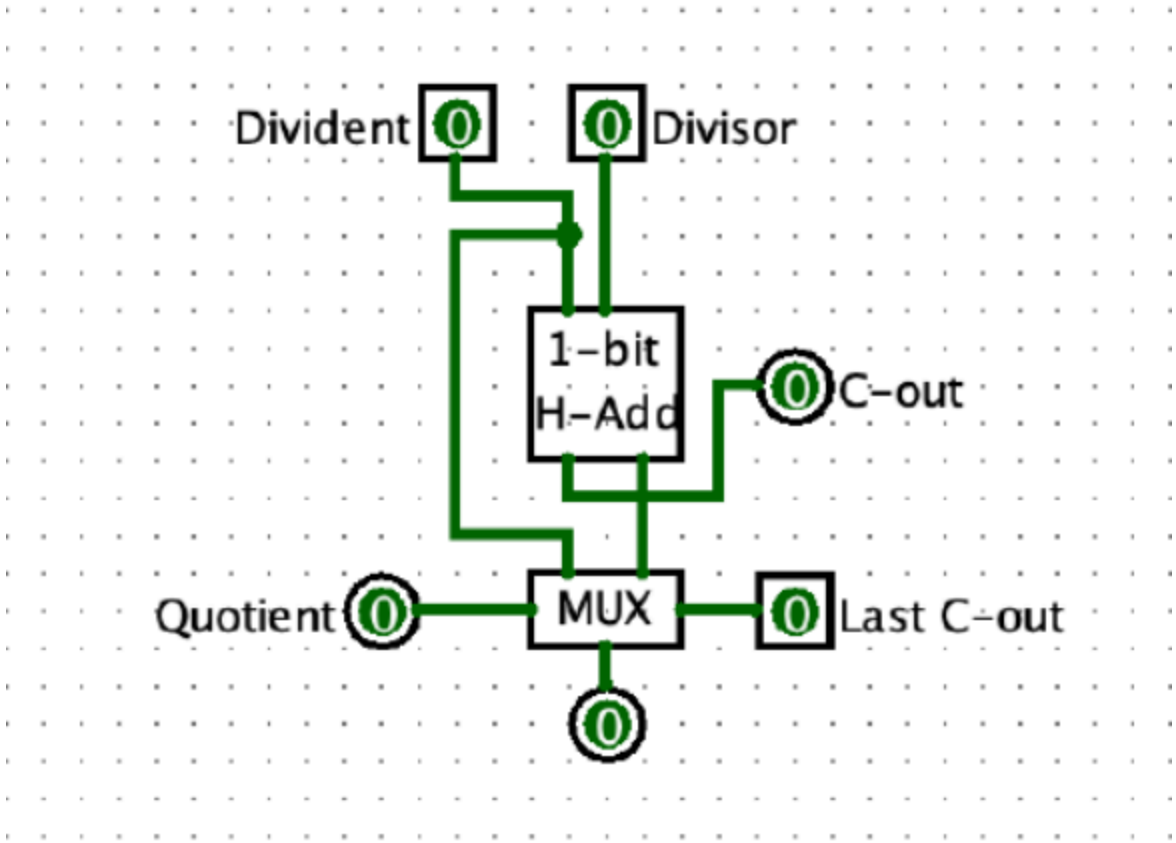
Rezultatul:



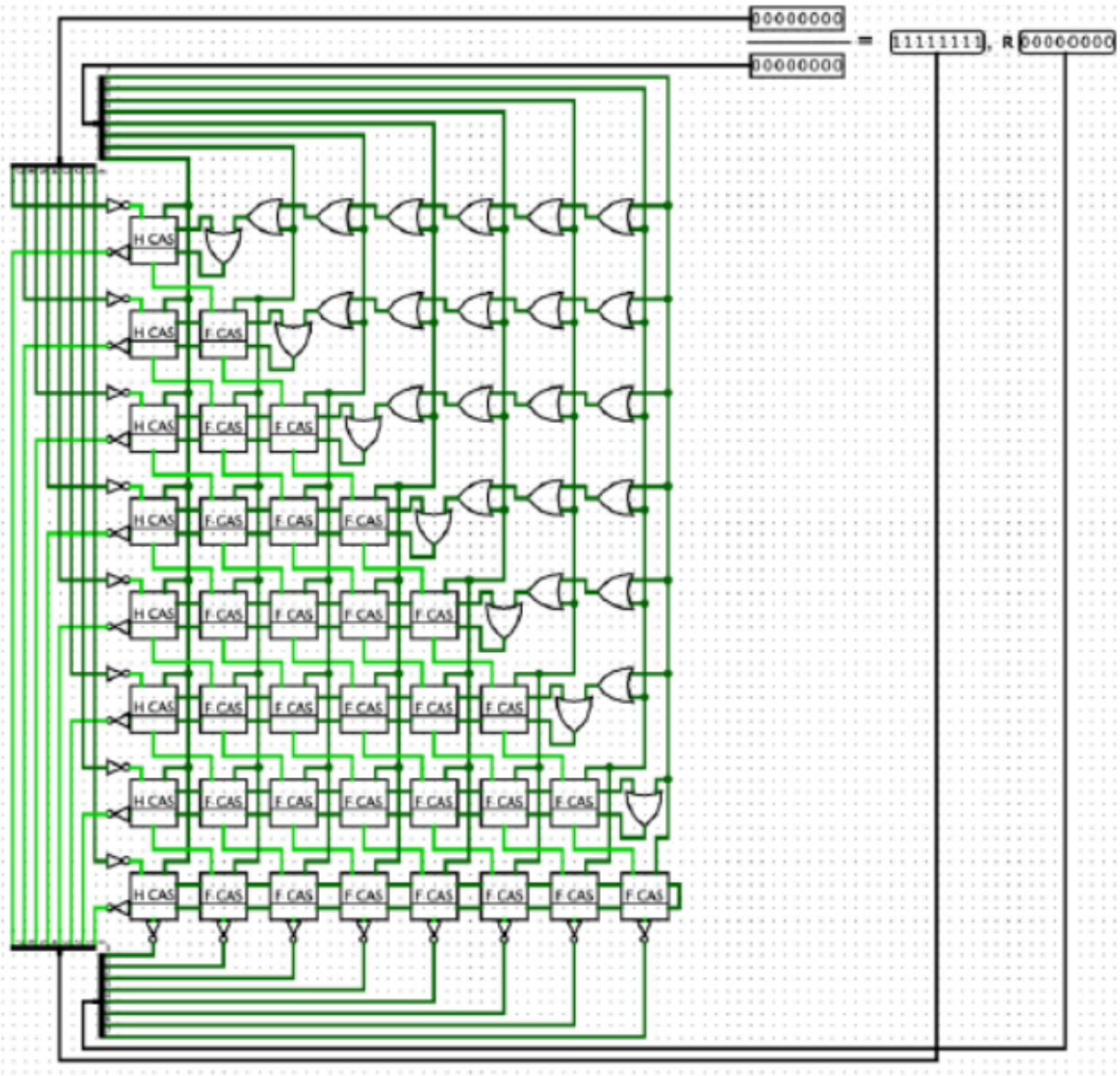
2-to-1 Multiplexer



Celulă divizor cu adunător complet



Celulă divizor cu semiadunător



Divizor de 8 la 8

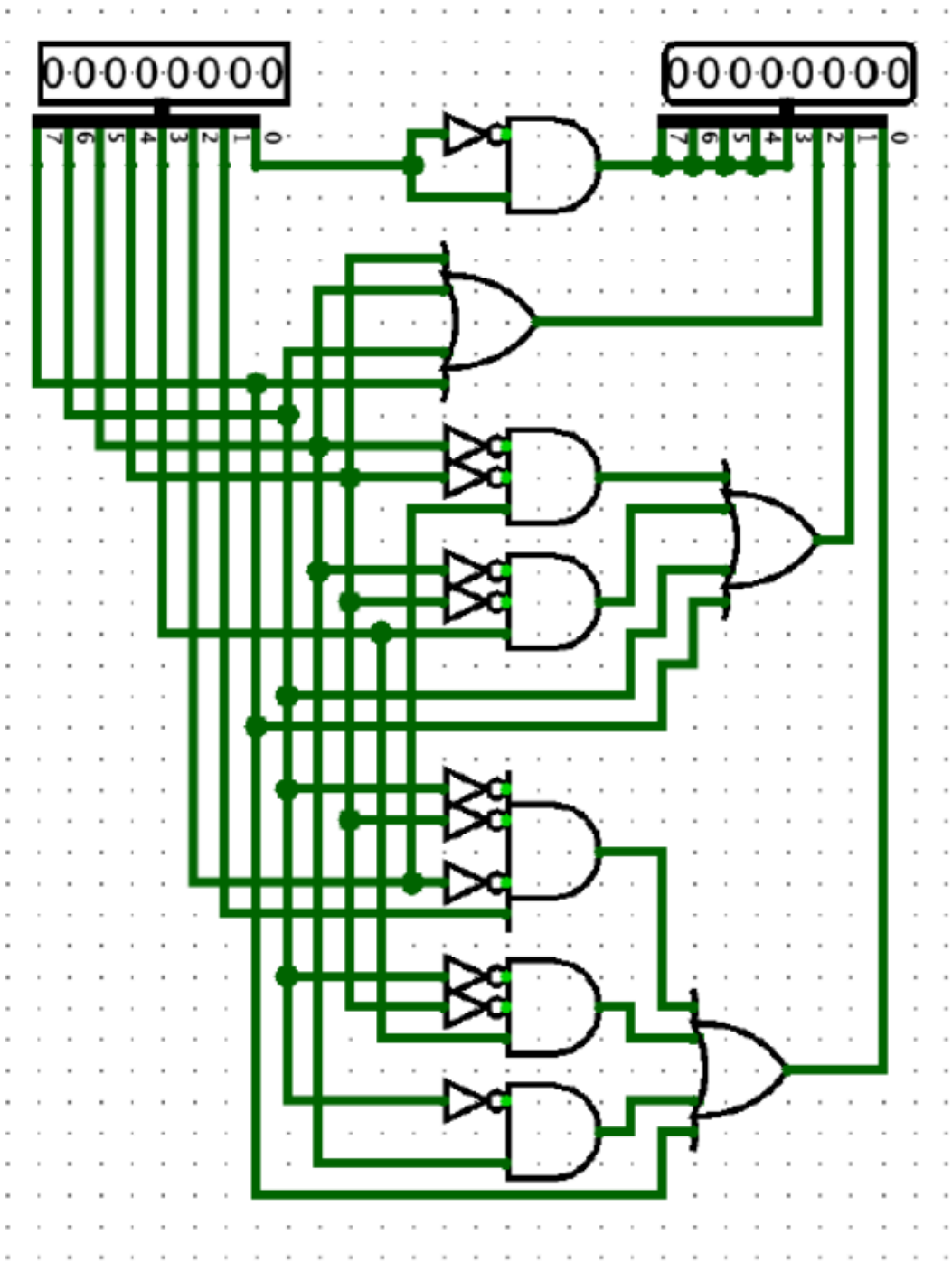
Pentru a înțelege funcționarea unui divizor de 8 pe 8 biți format din 8 semiadunătoare și 28 de adunătoare complete, să îl descompunem pas cu pas. Divizorul este conceput pentru a împărți un dividend de 8 biți printr-un divizor de 8 biți, rezultând un cotaț de 8 biți. Utilizează o serie de etape, fiecare conținând un semiadunător și un număr variabil de adunătoare complete într-o manieră piramidală. Există în total 8 etape, fiecare corespunzând unui bit din cotaț. Fiecare etapă calculează un bit din cotaț. Un semiadunător adaugă două cifre binare și produce o sumă și un transport. În acest divizor, fiecare etapă începe cu un semiadunător. Fiecare etapă, în afară de prima, conține un număr variabil de adunătoare complete, aranjate într-o

manieră piramidală. Prima etapă conține 0 adunătoare complete, a doua etapă conține 1 adunător complet, a treia etapă conține 2 adunătoare complete, și așa mai departe, până când a opta etapă conține 7 adunătoare complete. Scopul acestor adunătoare complete este de a efectua calculul principal pentru fiecare bit din cotaț. Fiecare etapă calculează un bit din cotaț prin efectuarea repetată a scăderii. La fiecare etapă, restul parțial (rezultatul scăderii anterioare) este deplasat la stânga cu un bit, și următorul bit din dividend este adăugat. Apoi, divizorul este scăzut din acest rest parțial. Rezultatul acestei scăderi este stocat ca un bit din cotaț. Procesul continuă pentru toate cele 8 etape până când întregul cotaț este calculat. Ieșirea fiecărei etape reprezintă un bit din cotaț. După cele 8 etape, cotațul și restul de 8 biți sunt formați.

Exercitiu 7H:

Implementați un circuit care realizează operația logaritmă în baza 2 a unui număr de 8 biți.

Rezultatul:



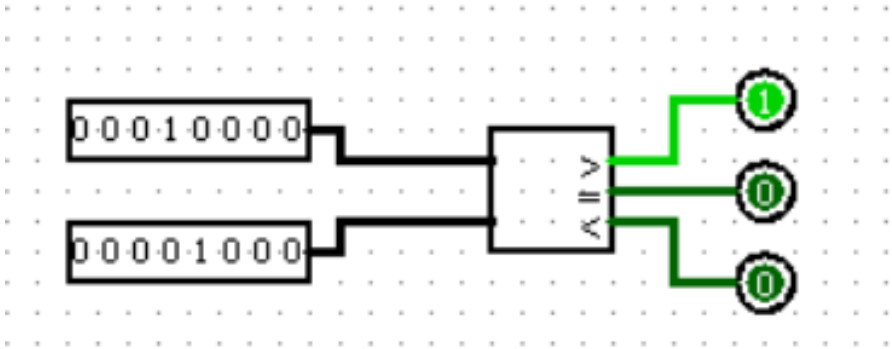
Un encoder prioritar este un circuit digital care preia mai multe intrări binare și oferă ca ieșire numărul intrării de ordine superioară (cea mai semnificativă) care este activată (setată pe 1). În acest caz, encoderul prioritar preia o intrare de 8 biți și oferă ca ieșire un număr binar de 4 biți indicând poziția bitului cel mai semnificativ care este setat pe 1. Un extractor de logaritm este

un circuit sau algoritm care calculează poziția bitului cel mai la stânga (cel mai semnificativ) care este setat pe 1 într-un număr binar. Într-un număr binar de 8 biți, acest lucru corespunde găsirii poziției bitului cel mai semnificativ care este setat pe 1, numărând de la stânga la dreapta (începând de la poziția 7 și terminând la poziția 0). Extractorul de logaritm de 8 biți implementat ca un encoder prioritar de 8 la 4 funcționează folosind encoderul prioritar pentru a determina poziția bitului cel mai semnificativ care este setat pe 1. Intrarea de 8 biți este conectată la intrările encoderului prioritar. Ieșirea encoderului prioritar este un număr binar de 4 biți reprezentând poziția bitului cel mai semnificativ care este setat pe 1. Informația de poziție furnizată de encoderul prioritar corespunde direct logaritmului (baza 2) al numărului de intrare. De exemplu, dacă ieșirea encoderului prioritar este "0110" (binar pentru decimal 6), indică faptul că bitul cel mai semnificativ setat pe 1 în intrare este la poziția 6, ceea ce corespunde logaritmului baza 2 al numărului de intrare. Ieșirea encoderului prioritar reprezintă direct logaritmul (baza 2) al numărului de intrare. Oferă o reprezentare binară de 4 biți a poziției bitului cel mai semnificativ setat pe 1 în intrare.

Exercitiu 1M:

1M) Implementați un comparator cu 8 biți folosind porți logice.

Rezultatul:

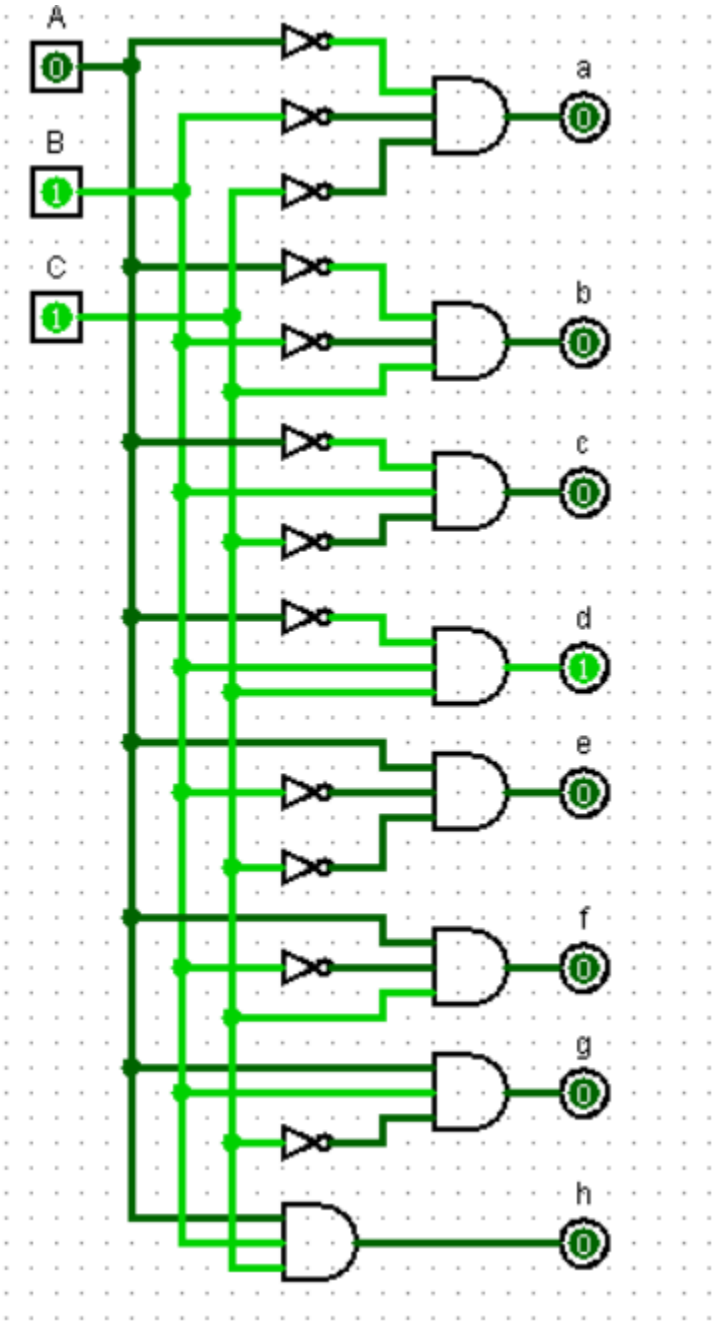


Acest circuit evaluează două numere de 8 biți folosind comparatorul implicit din Logisim. În acest exemplu, se compară 16 cu 6, iar ieșirea este mai mare.

Exercitiu 2M:

Implementați un decodor cu 3 biți și 8 ieșiri

Rezultatul:

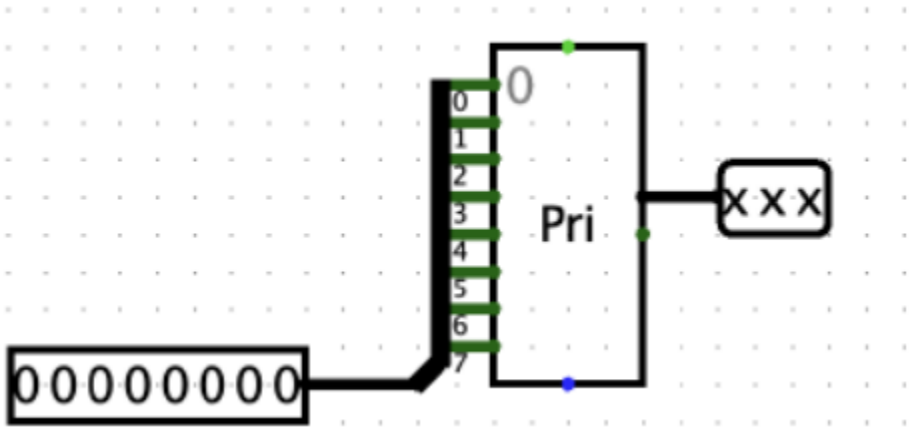


În circuitul de mai sus, cele trei intrări pot fi decodate în 8 ieșiri, unde fiecare ieșire reprezintă unul dintre termenii mijlocii ai celor trei variabile de intrare. Cele 3 inversoare din circuitul logic de mai sus vor furniza complementul intrărilor și fiecare dintre porțile AND va genera unul dintre termenii mijlocii. Acest tip de decodor este folosit în principal pentru a decoda orice cod binar de 3 biți și generează opt ieșiri, echivalente cu cele 8 combinații diferite pentru codul de intrare. Acest decodor este cunoscut și sub numele de decodor binar la octal deoarece intrările acestui decodor reprezintă numere binare de trei biți în timp ce ieșirile reprezintă cele 8 cifre din sistemul de numerație octal.

Exercitiu 3M:

3M). Implementați un codificator prioritar pentru 8 intrări, care să aibă prioritatea dată de valoarea intrărilor.

Rezultatul:

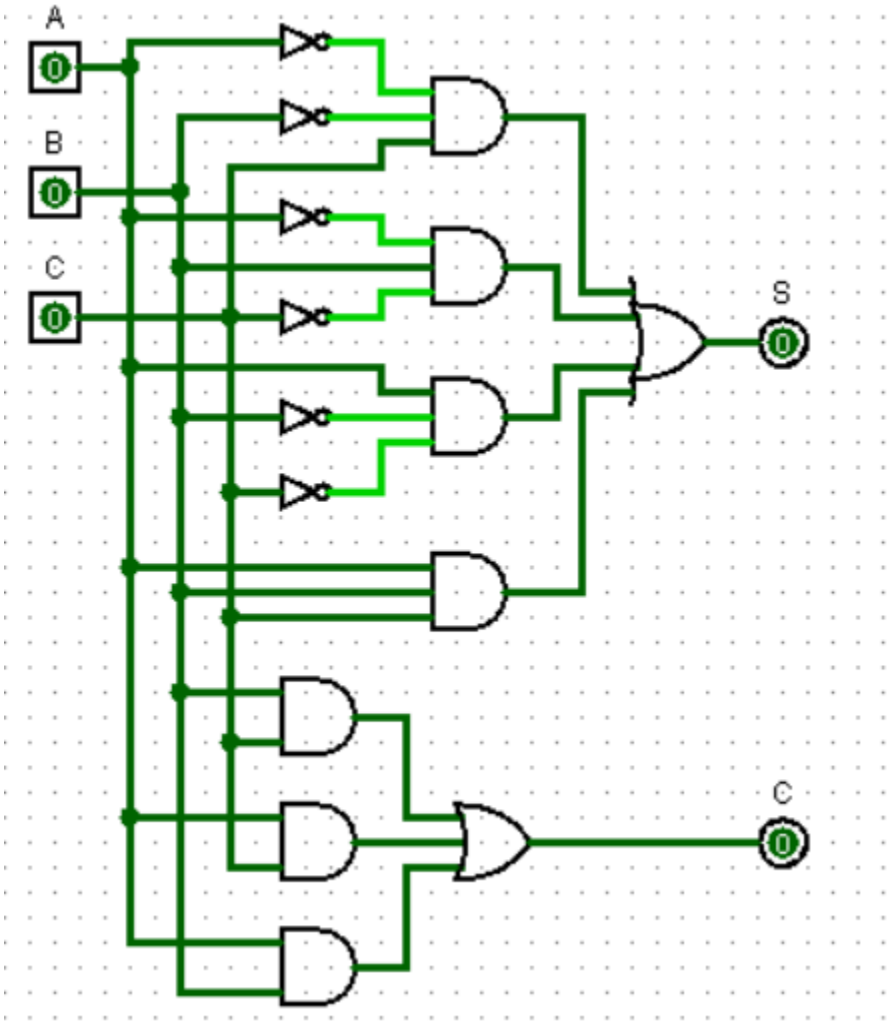


Pentru a implementa encoderul prioritar, veți compara fiecare intrare cu celelalte pentru a determina prioritatea cea mai înaltă. Ieșirea va reprezenta codul binar corespunzător intrării cu prioritatea cea mai mare. Folosiți o serie de comparatoare pentru a compara fiecare valoare de intrare cu toate celelalte valori de intrare. Ieșirea fiecărui comparator indică dacă intrarea este mai mare sau egală cu celelalte intrări. Conectați ieșirile comparatorilor la intrările multiplexoarelor. Multiplexoarele selectează intrarea cu prioritatea cea mai mare pe baza ieșirilor comparatorilor și oferă ca ieșire codul său binar.

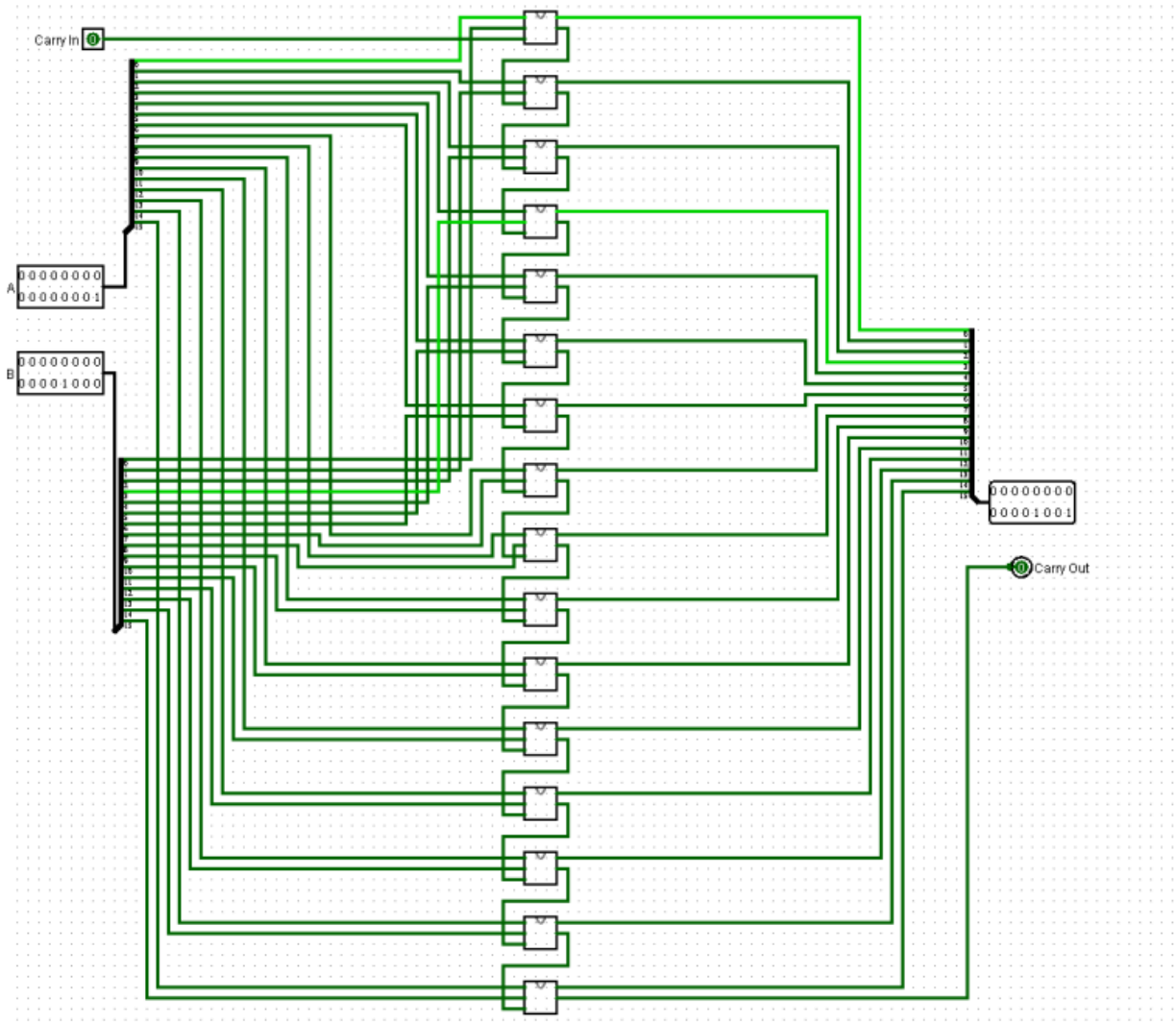
Exercitiu 4M:

4M) Implementați un circuit de adunare completă (full-adder) cu 16 biți.

Rezultatul:



Adunător complet

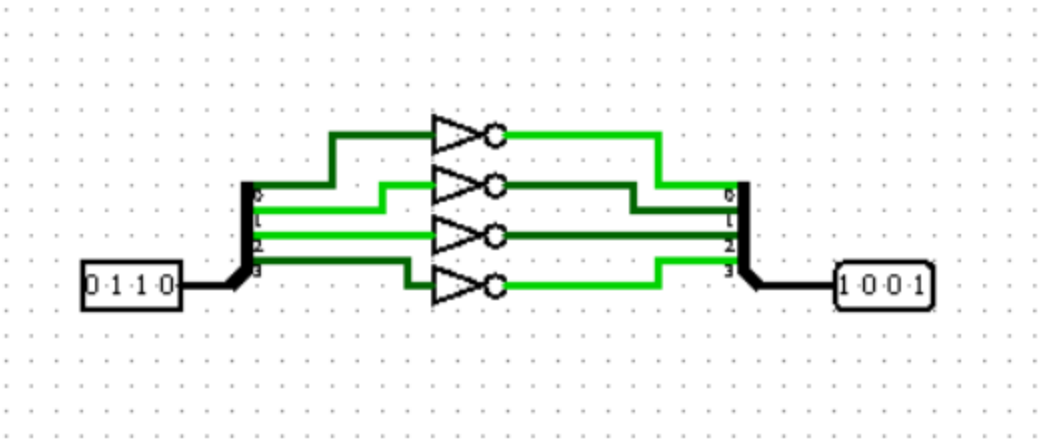


Pentru acest circuit folosesc un adunător complet personalizat (figura 13) generat cu ajutorul unui tabel de adevăr. Apoi, folosesc 16 adunătoare complete pentru a obține suma numerelor de până la 16 biți. În figură, obțin suma dintre 1 și 8. După cum este intenționat, ieșirea este 9.

Exercitiu 11M:

11M) Implementați un circuit care realizează operația de negare a unui număr de 4 biți.

Rezultatul:

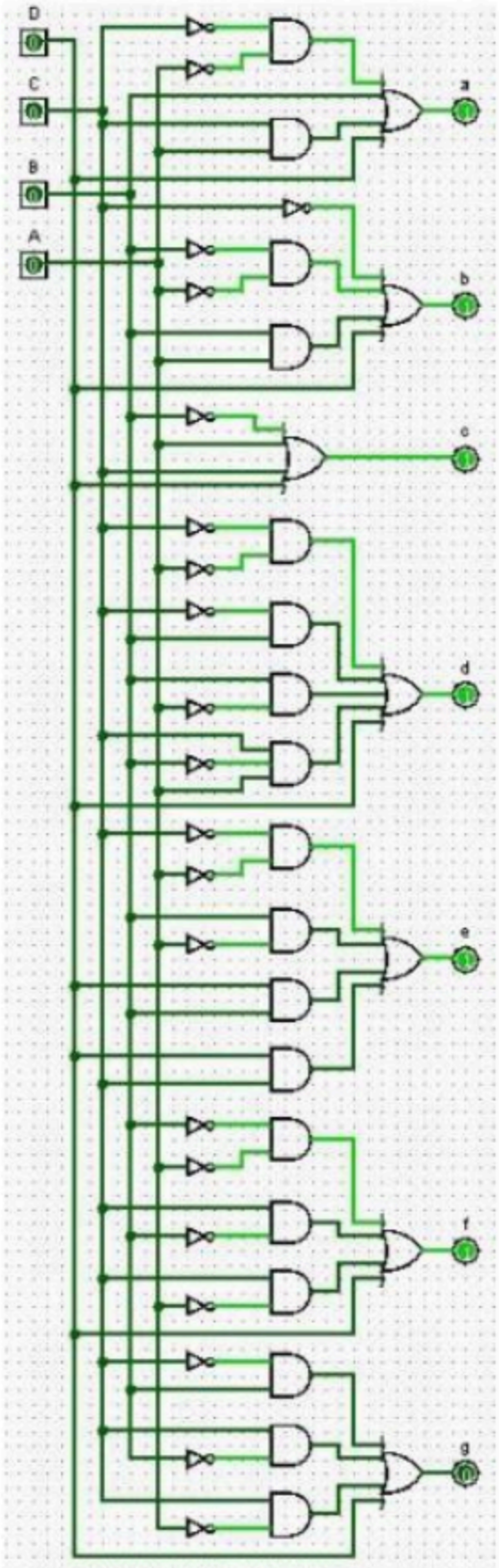


Pentru acest circuit am folosit o intrare de 4 biți. Apoi intră într-un divizor. În continuare, am folosit 4 porți NOT pentru a nega fiecare valoare. Apoi, valorile sunt introduse printr-un alt divizor spre ieșire.

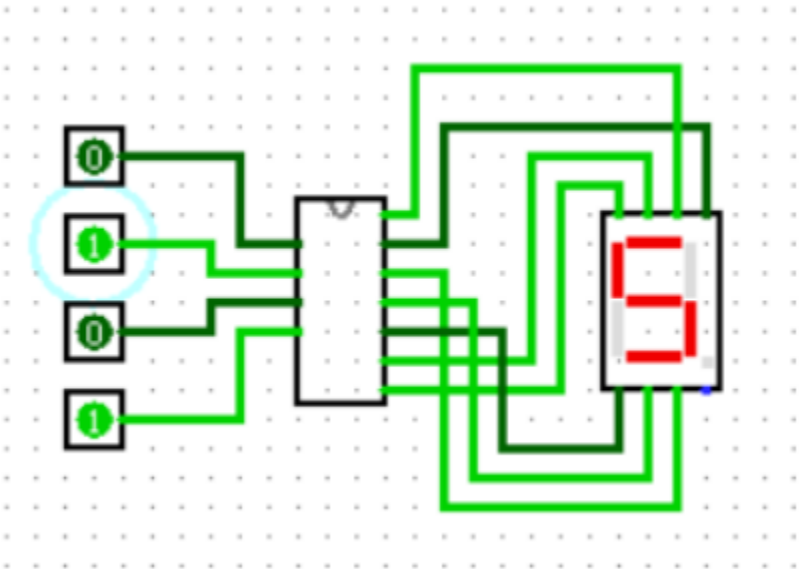
Exercitiu 16M:

7M) Implementați un circuit logic care va activa ieșirea dacă ambele intrări sunt 1.

Rezultatul:


D
C
B
A
a
b
c
d
e
f
g

circuit personalizat pentru conversie

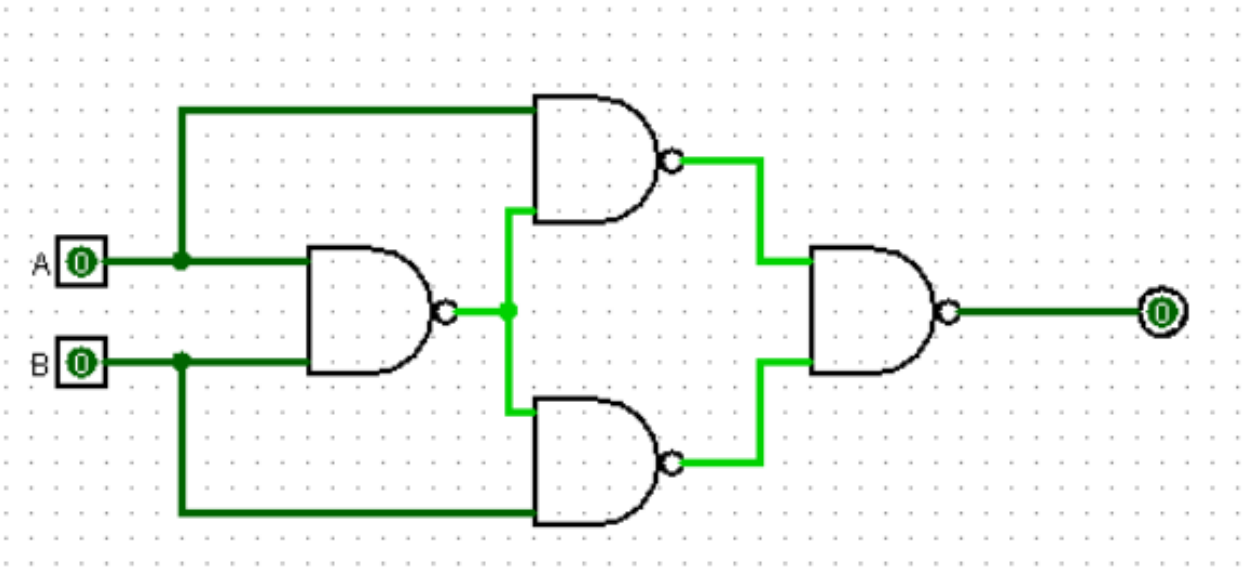


Circuitul funcționează cu intrări binare etichetate A, B, C și D, care în total reprezintă un număr binar de 4 biți care variază de la 0000 (0 în zecimal) la 1111 (15 în zecimal). Ieșirile circuitului, notate ca a, b, c, d, e, f și g, corespund segmentelor unui afișaj standard cu 7 segmente. Fiecare segment este desemnat cu o literă și poate fi iluminat în diverse combinații pentru a reprezenta numere zecimale de la 0 la 9 și, în anumite cazuri, caractere hexazecimale de la A la F. Pentru a determina care segmente ale afișajului cu 7 segmente ar trebui iluminate pe baza intrării binare, circuitul utilizează o combinație de porți logice. Porțile NOT, reprezentate de triunghiuri inversate, inversează semnalul de intrare; porțile AND, reprezentate ca D-uri cu o parte plată, produc o ieșire adevărată doar dacă toate intrările sunt adevărate; în timp ce porțile OR, cu o curbă și un capăt ascuțit, produc o ieșire adevărată dacă cel puțin o intrare este adevărată. Pentru a proiecta un astfel de circuit, de obicei se începe cu un tabel de adevăr care definește care segmente ar trebui iluminate pentru fiecare combinație de intrare posibilă. Apoi, adesea se utilizează hărți Karnaugh pentru a simplifica ecuațiile logice rezultate din tabelul de adevăr. De exemplu, pentru a ilumina segmentele b și c pentru a afișa numărul zecimal 1 pe afișajul cu 7 segmente, circuitul logic ar produce un semnal înalt (logic 1) la ieșirile b și c, și semnale joase (logic 0) la toate celelalte ieșiri atunci când intrarea binară corespunde numărului zecimal 1.

Exercitiu 1E:

1E) Implementați o poartă XOR cu două intrări folosind poarta NAND.

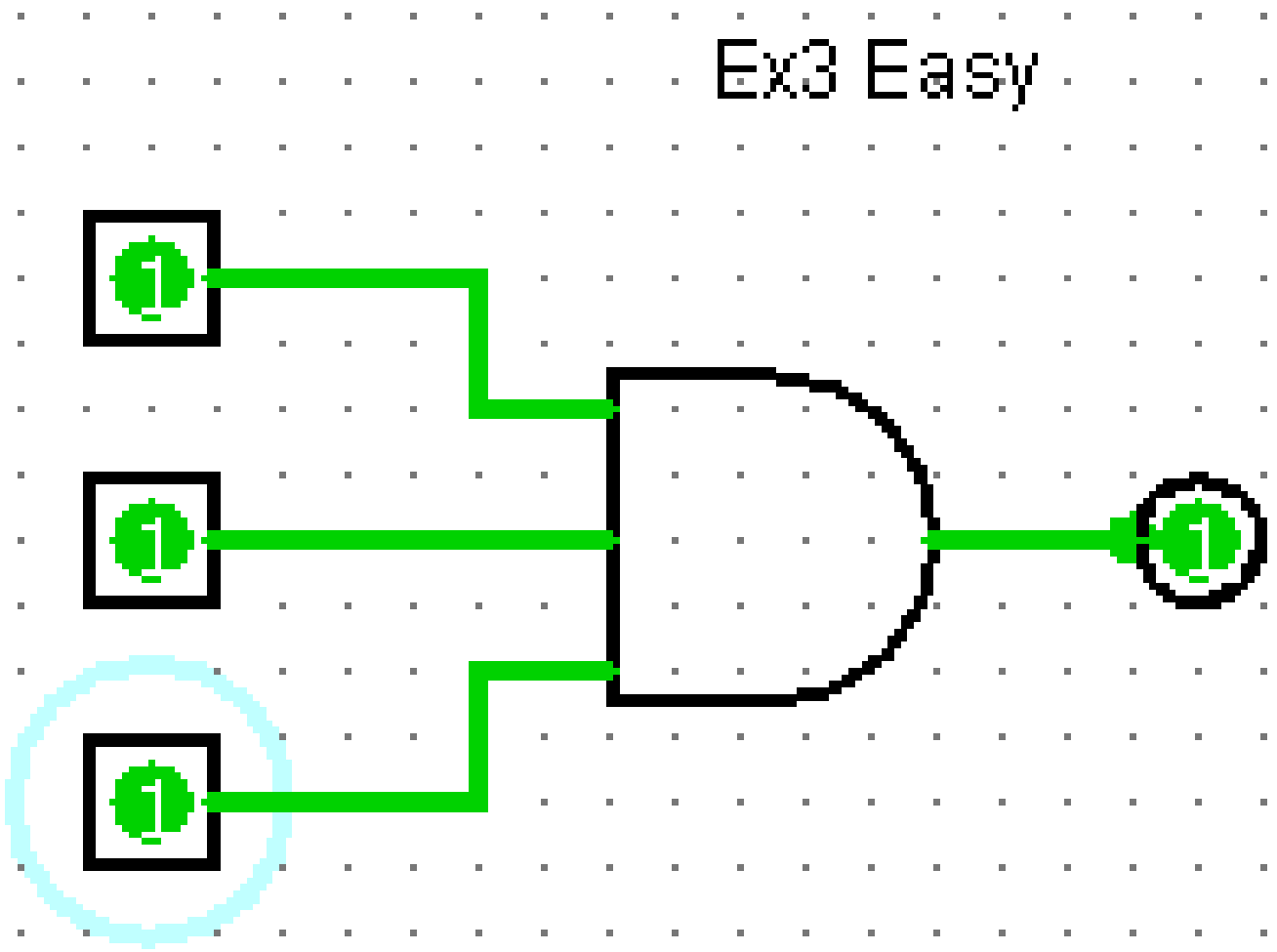
Rezultatul:



Exercitiu 3E:

3E) Implementați o poartă AND cu trei intrări folosind poarta AND

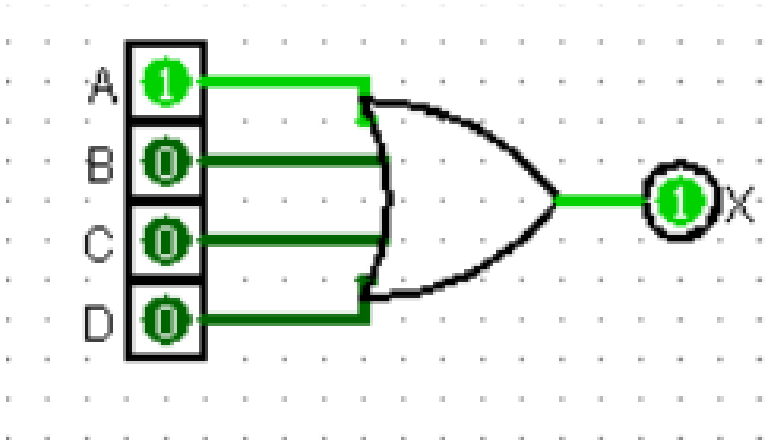
Rezultatul:



Exercitiu 4E:

4E) Implementați o poartă OR cu patru intrări folosind poarta OR

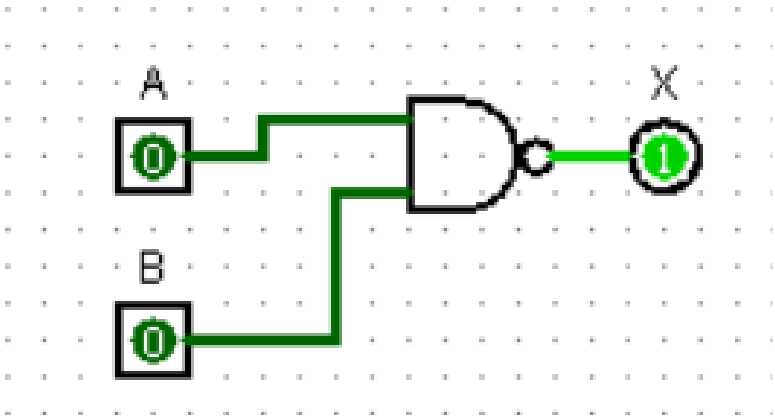
Rezultatul:



Exercitiu 5E:

5E) Implementați o poartă NOT cu două intrări folosind poarta NAND.

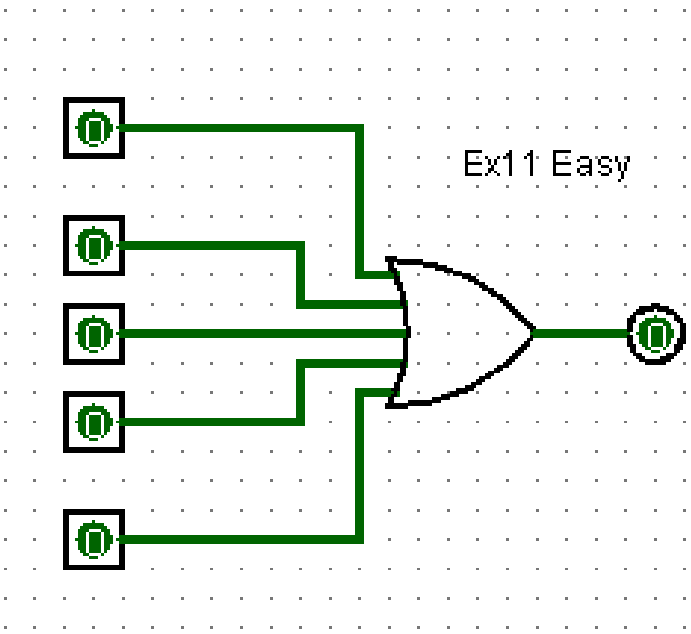
Rezultatul:



Exercitiu 11E:

11E) Implementați o poartă OR cu cinci intrări folosind poarta OR.

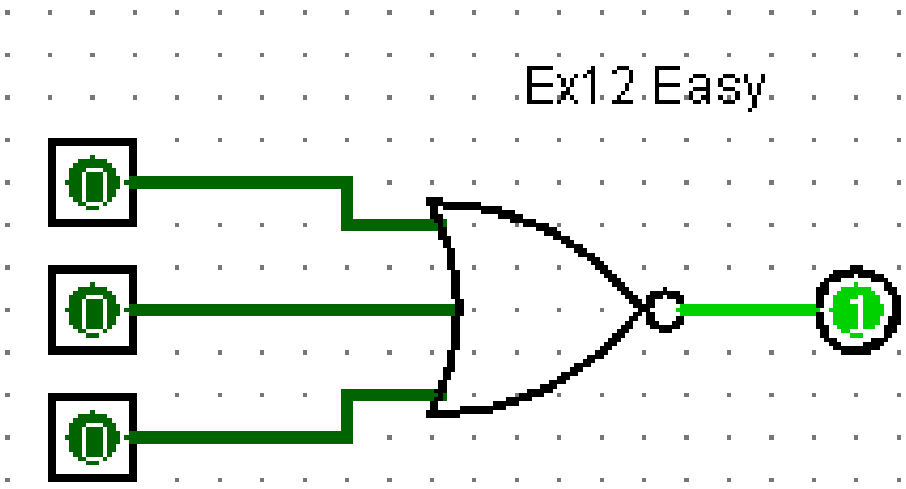
Rezultatul:



Exercitiu 12E:

12E) Implementați o poartă NOT cu trei intrări folosind poarta NOR

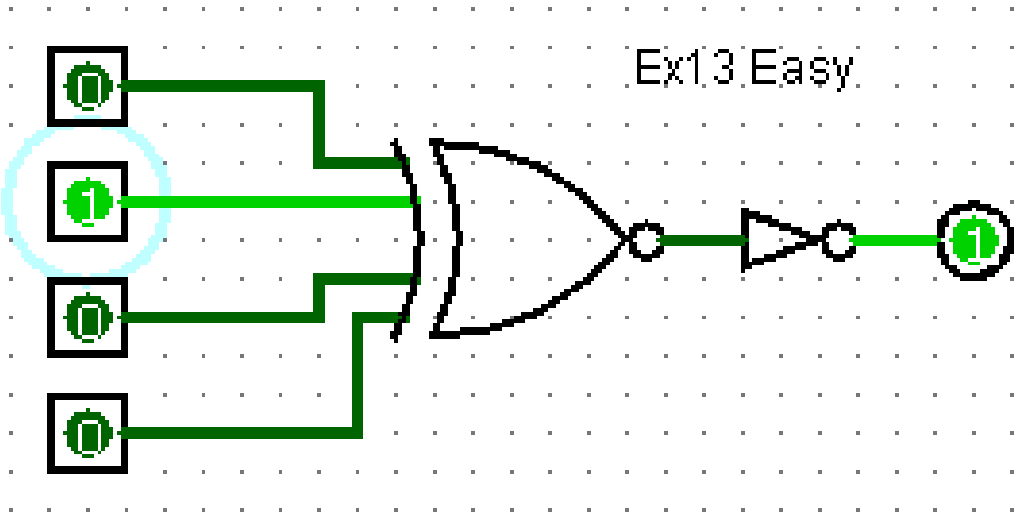
Rezultatul:



Exercitiu 13E:

13E) Implementați o poartă XOR cu patru intrări folosind poarta XNOR.

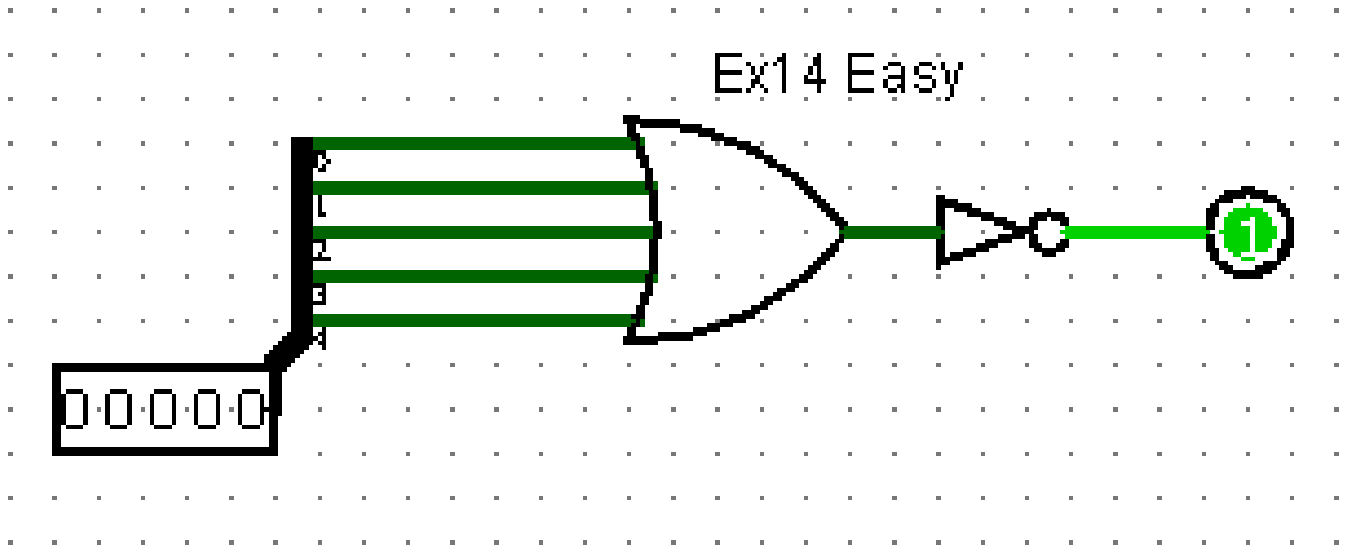
Rezultatul:



Exercitiu 14E:

14E) Implementați o poartă XNOR cu cinci intrări folosind poarta OR.

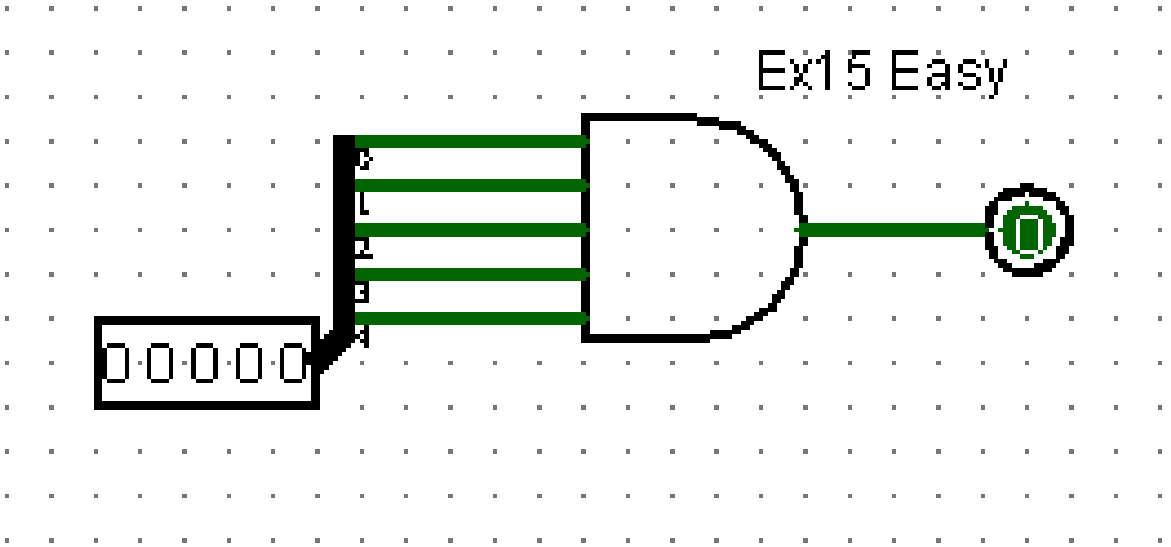
Rezultatul:



Exercitiu 15E:

15E) Implementați un circuit logic care va activa ieșirea dacă intrarea este nulă.

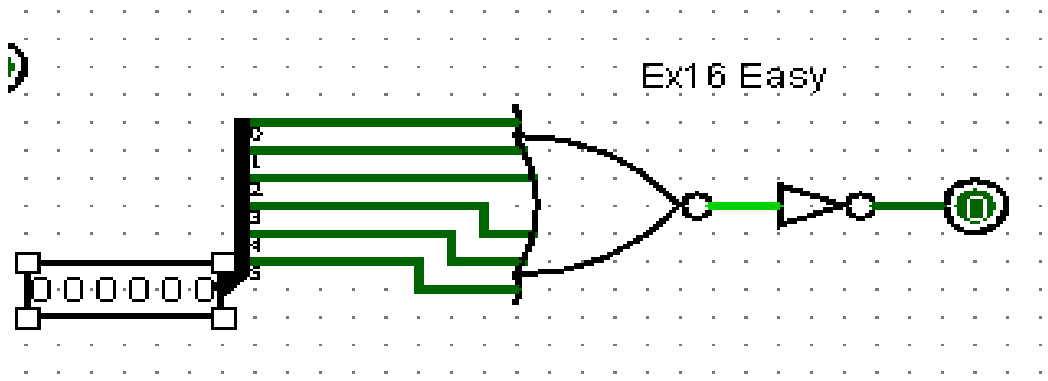
Rezultatul:



Exercitiu 16E:

16E) Implementați o poartă OR cu șase intrări folosind poarta NOR

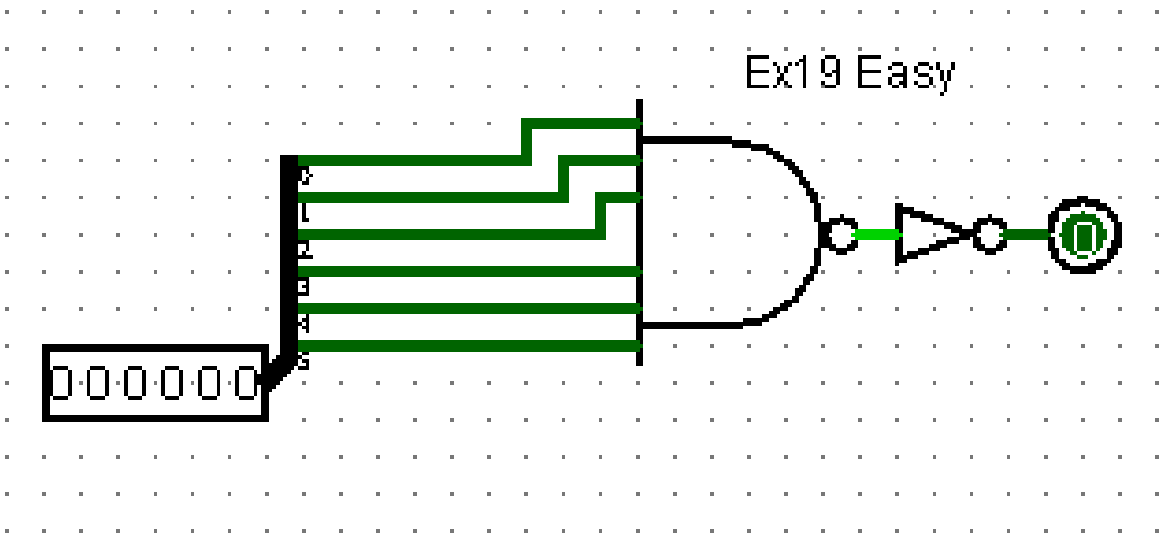
Rezultatul:



Exercitiu 19E:

19E) Implementați o poartă AND cu șase intrări folosind poarta NAND.

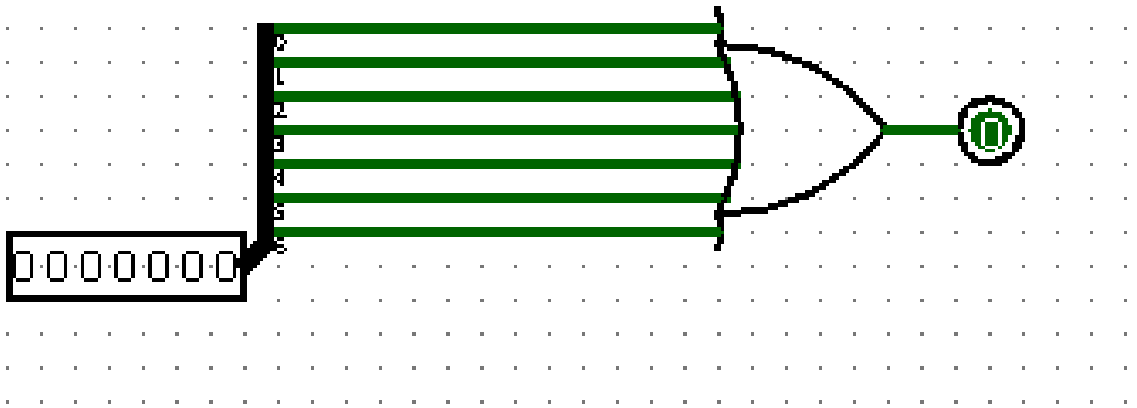
Rezultatul:



Exercitiu 20E:

20E) Implementați o poartă OR cu șapte intrări folosind poarta OR.

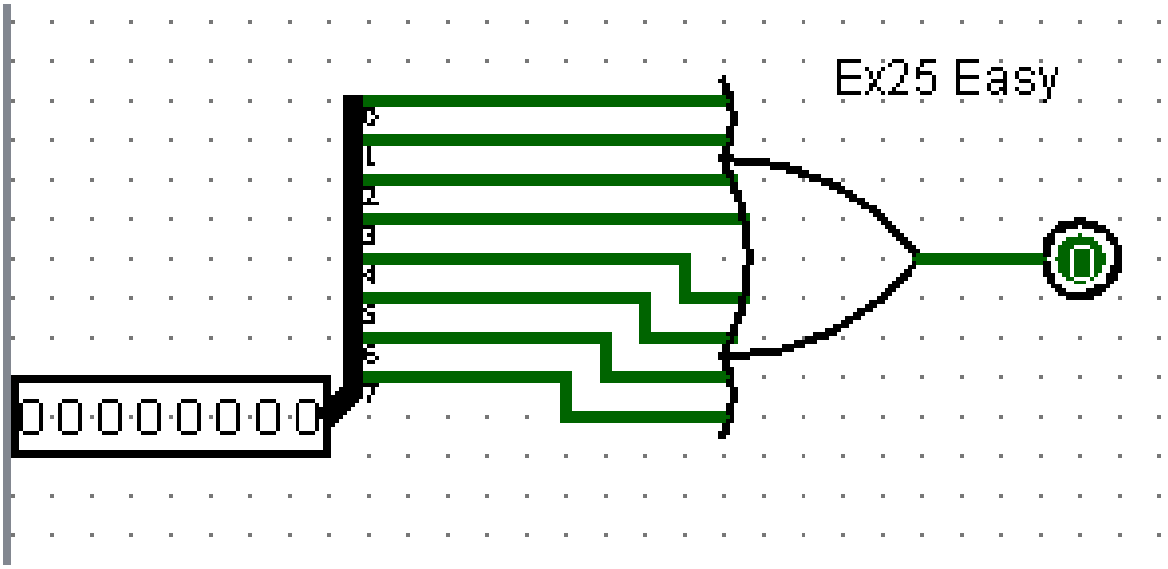
Rezultatul:



Exercitiu 25E:

25E) Implementați o poartă OR cu opt intrări folosind poarta OR.

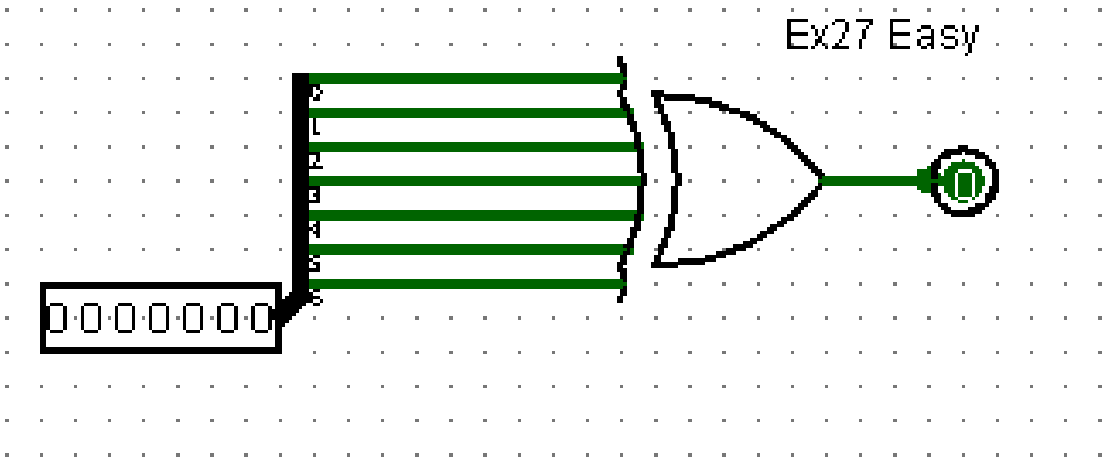
Rezultatul:



Exercitiu 27E:

27E) Implementați o poartă XOR cu șapte intrări folosind poarta XOR.

Rezultatul:



Concluzie:

În concluzie, finalizarea acestui laborator de practică pe Logisim a oferit o oportunitate inestimabilă de a explora detaliile designului și implementării de circuite digitale. Prin intermediul unei serii de exerciții diverse, care au cuprins de la porți logice de bază până la circuite aritmetice complexe și convertoare de date, nu numai că mi-am consolidat înțelegerea principiilor electronicii digitale, dar am și îmbunătățit abilitățile practice în utilizarea Logisim ca instrument de simulare.

Construind meticulos fiecare circuit, analizându-le comportamentul și identificând și rezolvând potențiale probleme, am dezvoltat o apreciere profundă pentru conceptele fundamentale care guvernează sistemele digitale. De la operațiile logice de bază precum porțile AND, OR și NOT, până la componente mai avansate cum ar fi adunătoarele, comparatoarele și codificatoarele, fiecare exercițiu a oferit o provocare unică care m-a încurajat să explorez diferite aspecte ale circuitelor digitale.

În plus, munca de laborator a stimulat gândirea critică și abilitățile de rezolvare a problemelor, deoarece am trebuit să găsesc soluții inovatoare pentru a obține funcționalitatea dorită în cadrul mediului Logisim. Prin experimentare și iterație, am dobândit cunoștințe despre optimizarea și eficiența circuitelor, învățând să minimizez complexitatea în timp ce maximizez performanța.

Privind în viitor, cunoștințele și abilitățile dobândite din acest laborator de lucru vor servi ca o bază solidă pentru proiectele viitoare în domeniul electronicilor digitale și al ingineriei calculatoarelor. Fie că urmează să urmez studii academice suplimentare sau să mă implic în proiecte profesionale, sunt echipat cu o înțelegere cuprinzătoare a principiilor de design al circuitelor digitale și cu expertiza practică necesară pentru a aborda provocările reale.

În esență, acest laborator de lucru a fost o experiență de învățare transformatoare, care m-a împuternicit să navighez în lumea intricate a sistemelor digitale cu încredere și pricepere.