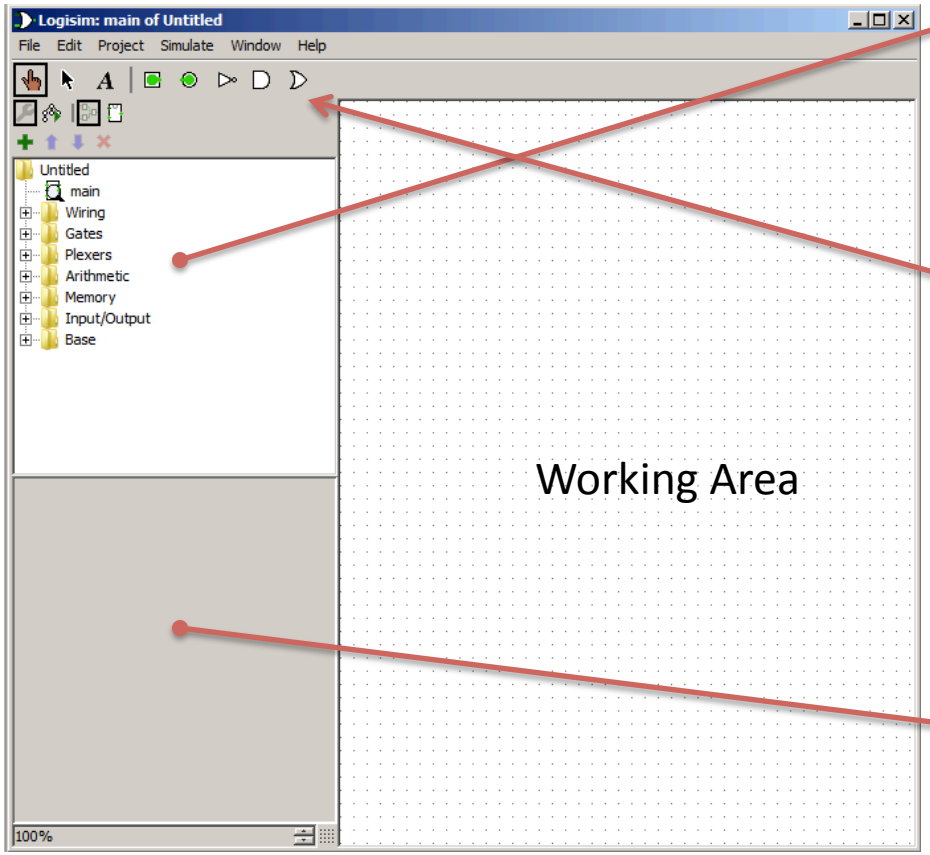
Introduzione a Logisim e circuiti combinatori

Logisim

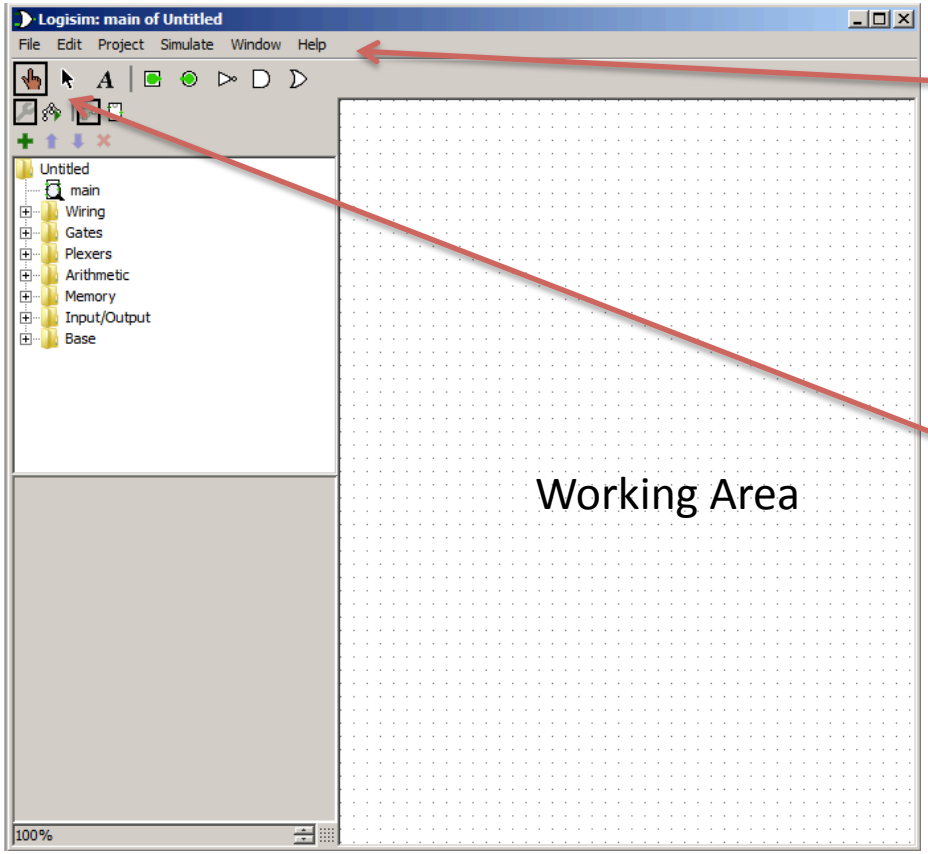
- Logisim (Logic Simulator)
- Sito: http://ozark.hendrix.edu/~burch/logisim/
- Download: http://sourceforge.net/projects/circuit/

Logisim: schermata iniziale



- Nel menu a sinistra, è possibile accede alle porte di base (Basic Gate: NOT, AND, OR & Buffer).
- Le porte devono essere selezionate dal menu e posizionate all'interno del circuito cliccando nel punto scelto.
- In alto a sinistra è possibile selezionare alcuni componenti di uso frequente: pin di ingresso ed uscita, AND, OR, NOT
- La porta di input permette di inserire un segnale true/false. La porta di output permette di leggere un segnale true/false.
- Le porte sono connesse tra di loro semplicemente trascinando con il mouse (tasto sinistro premuto) i connettori di input/output. Da un connettore di output possono uscire più fili!
- Le proprietà di ogni componente possono essere modificate dalla finestra proprietà che compare in basso a sinistra cliccando il componente stesso

Logisim: schermata iniziale

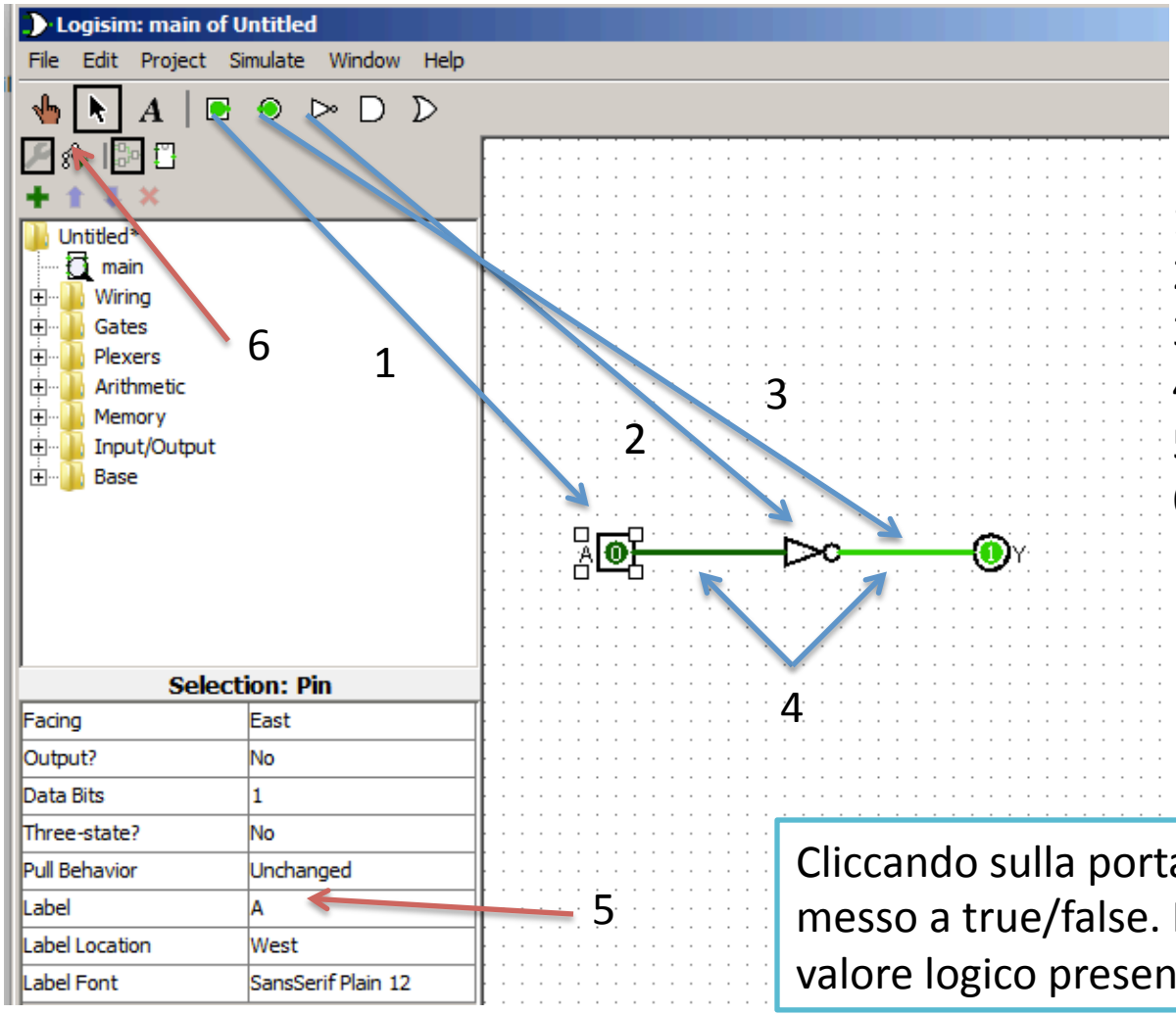


- Nel menù di Help è presente un tutorial che illustra le principali funzionalità
- A sinistra nel menù trovate un puntatore per eseguire la simulazione sul circuito
- Sempre a sinistra nel menù trovate il puntatore per modificare il circuito

Es. 1

- Si disegni il circuito che realizza l'operazione di negazione di un segnale chiamato A in Logisim.

Soluzione e osservazioni



1. Input
2. Not
3. Output
4. Connessioni
5. Rinomina input e output
6. Simlazione

Cliccando sulla porta di input, il segnale viene messo a true/false. Il colore dei fili rispecchia il valore logico presente durante la simulazione.

Es. 2

Si disegni il circuito che realizza:

X = (A and (not B)) or C.

Si derivi la tabella della verità del circuito e si controlli la correttezza dei risultati utilizzando Gatesim.

Soluzione

- Per derivare la tabella della verità, dobbiamo considerare tutte le possibili combinazioni degli ingressi. Dobbiamo quindi calcolare l'output della funzione. E' utile calcolare i risultati intermedi e mettere anch'essi nella tabella.

Soluzione

X = (A and (not B)) or C

A	B	C	Z1 = not B	Z2 = A and Z1	X = Z2 or C
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Soluzione

X = (A and (not B)) or C

A	B	C	Z1 = not B	Z2 = A and Z1	X = Z2 or C
0	0	0	1		
0	0	1	1		
0	1	0	0		
0	1	1	0		
1	0	0	1		
1	0	1	1		
1	1	0	0		
1	1	1	0		

Soluzione

X = (A and (not B)) or C

A	B	C	Z1 = not B	Z2 = A and Z1	X = Z2 or C
0	0	0	1	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	1	0	0	
1	0	0	1	1	
1	0	1	1	1	
1	1	0	0	0	
1	1	1	0	0	

Soluzione

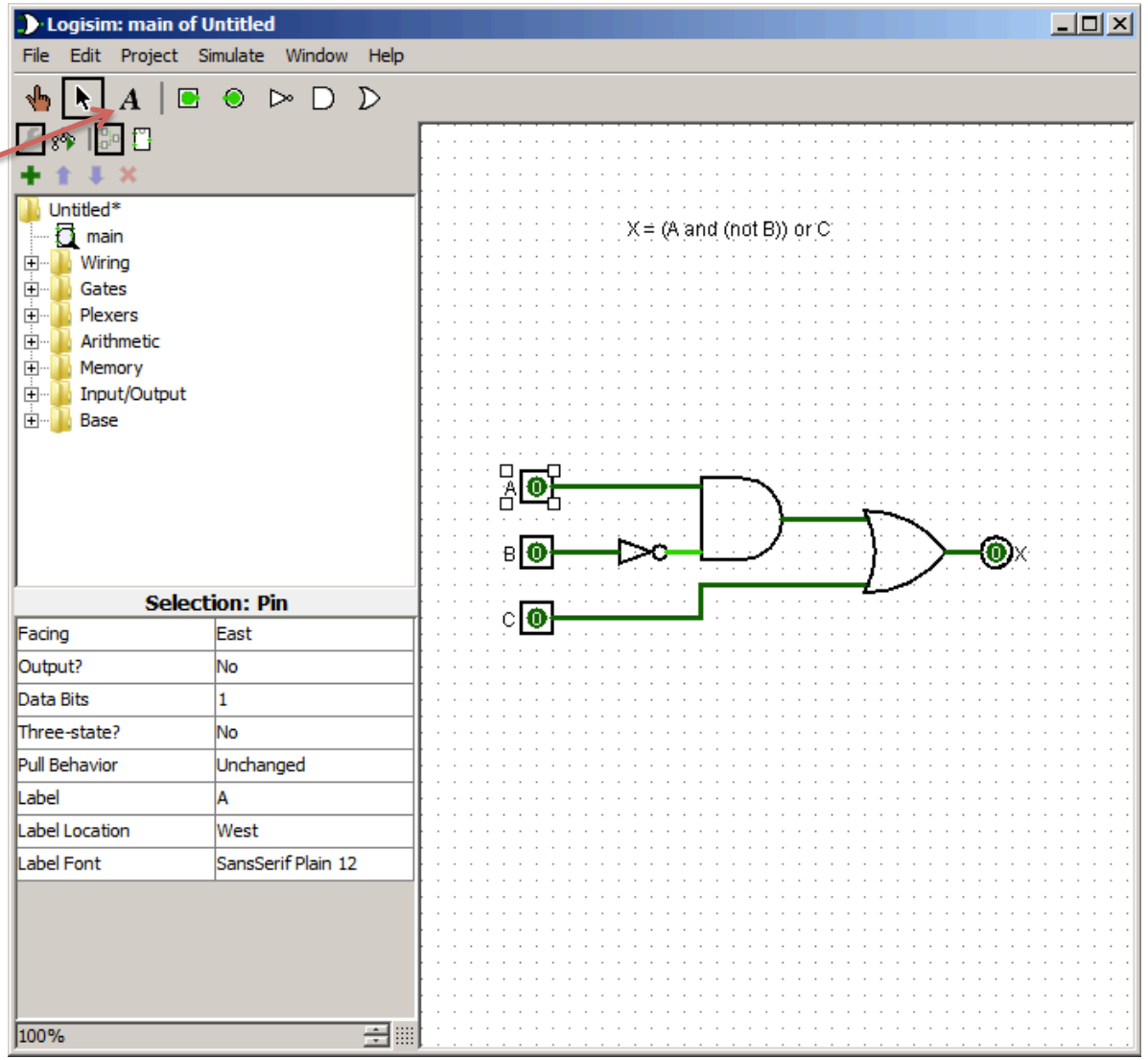
X = (A and (not B)) or C

A	B	C	Z1 = not B	Z2 = A and Z1	X = Z2 or C
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1

Soluzione

- E' possibile inserire dei testi nel progetto attraverso questo pulsante

- Selezionando una porta logica è possibile cambiare il numero di input attraverso la finestra proprietà



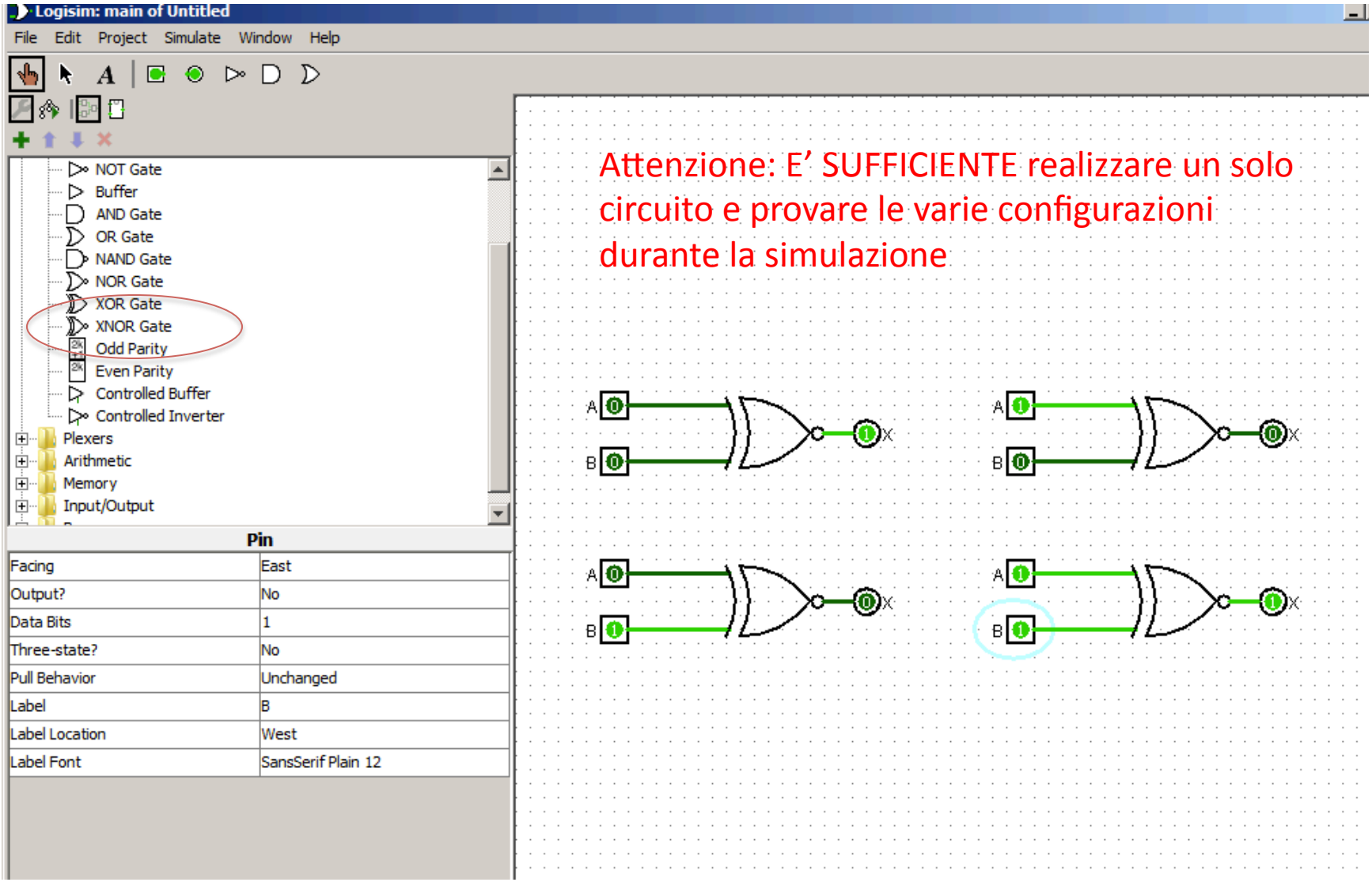
Porte “composte”

- Oltre alle porte di base ed alle porte di I/O, sono presenti delle porte “composte” che rappresentano le funzioni NAND, NOR, XOR e XNOR.
- Queste sono utili, ad esempio, quali porte universali per la realizzazione di un circuito.

Es. 3

- Si definiscano in Logisim due segnali A e B. Si utilizzi la porta XNOR per calcolare X = A XNOR B.
- Si derivi, analizzando l'output X, la tabella di verità di XNOR. A quale funzione logica corrisponde?
- Si implementi in gatesim un circuito equivalente a XNOR utilizzando le porte AND, OR e NOT necessario. Si verifichi la correttezza dell'implementazione confrontando l'uscita di XNOR con l'uscita del circuito implementato.
- hint: le due uscite devono essere uguali per qualsiasi configurazione di ingresso -> è possibile utilizzare la porta XNOR stessa per effettuare questo controllo!

Soluzione


Logisim: main of Untitled
File Edit Project Simulate Window Help
NOT Gate
Buffer
AND Gate
OR Gate
NAND Gate
NOR Gate
XOR Gate
XNOR Gate
Odd Parity
Even Parity
Controlled Buffer
Controlled Inverter
Plexers
Arithmetic
Memory
Input/Output
Pin
Facing East
Output? No
Data Bits 1
Three-state? No
Pull Behavior Unchanged
Label B
Label Location West
Label Font SansSerif Plain 12
Attenzione: E' SUFFICIENTE realizzare un solo circuito e provare le varie configurazioni durante la simulazione
A
B
X

Soluzione

Deriviamo quindi la tabelle di verità per XNOR:

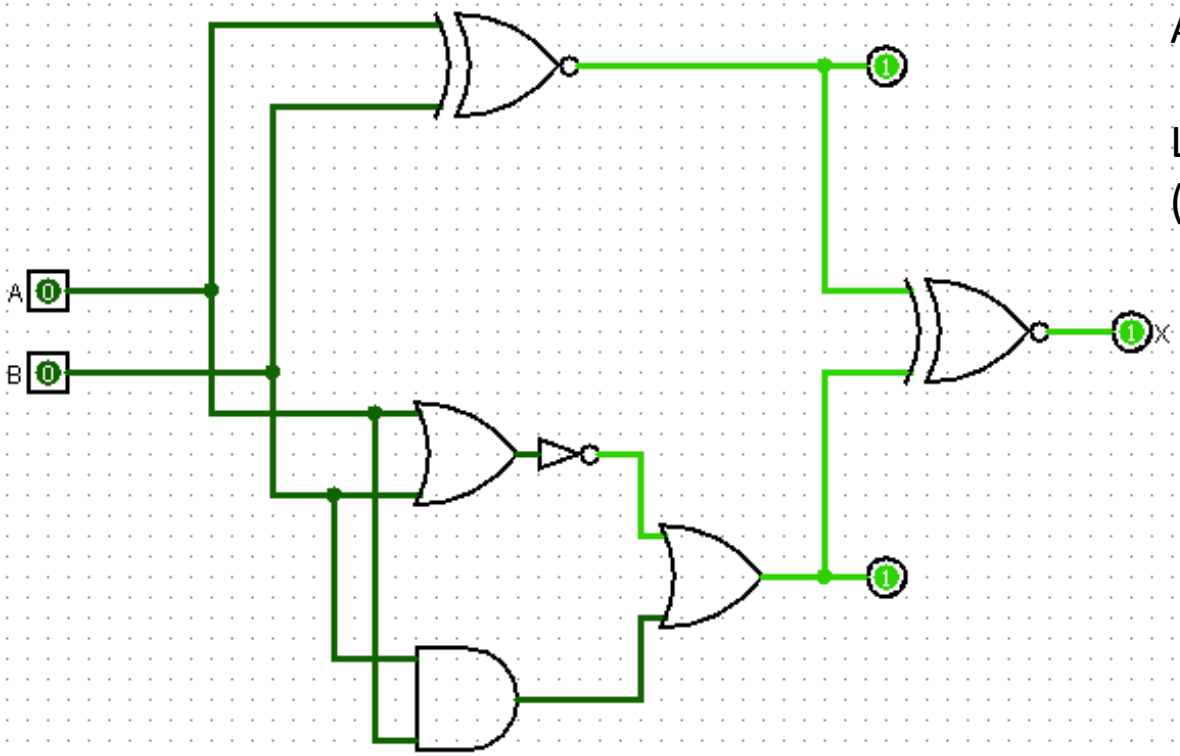
A	**B**	**A XNOR B**
0	0	1
0	1	0
1	0	0
1	1	1

- L'uscita della porta XNOR è pari a 1 quando A e B sono uguali, è pari a 0 altrimenti. La porta determina se i due bit sono uguali o diversi!
- Da qui la possibilità di utilizzare la XNOR per confrontare l'usicta di due circuiti.

Soluzione

- Proviamo a pensare come realizzare il circuito della XNOR (oggi utilizziamo l'intuito – vedremo più che esistono tecnice che rendono il compito di sintesi del circuito molto più semplice).
- Dobbiamo realizzare una porta che va ad 1 quando A=0 e B=0, oppure quando A=1 e B=1.
- La porta OR va a zero solo quando A=0 e B=0, quindi la negazione di OR (detta NOR) va ad 1 solo quando A=0 e B=0.
- La porta AND va ad 1 solo quando A=1 e B=1.
- Possiamo quindi generare due segnali (A NOR B, A AND B), quando uno di questi due va ad 1 anche la porta XNOR va ad 1.
- Possiamo quindi usare la porta OR per legare i due segnali ed ottenere l'output desiderato...
- Dovrebbe quindi valere: (NOT(A OR B)) OR (A AND B) = A XNOR B.

Soluzione



La parte alta del cricuito è:
A XNOR B.

La parte bassa del c circuito è:
(NOT(A OR B)) OR (A AND B).

Soluzione

E’ possibile verificare che l’output dei due circuiti è lo stesso per qualsiasi configurazione di ingresso di A e B. Tale verifica può essere fatta anche ponendo in ingresso ad una porta XNOR l’output dei due circuiti e verificando che l’output della porta XNOR è 1 per qualsiasi configurazione di ingresso.

Quale ulteriore verifica (ed esercizio), ricaviamo la tabella della verità di (NOT(A OR B)) OR (A AND B) e verifichiamo che l’output sia lo stesso di A XNOR B.

A	**B**	**Z1 = A OR B**	**Z2 = NOT Z1**	**Z3 = A AND B**	**X = Z2 OR Z3**
0	0	0	1	0	1
0	1	1	0	0	0
1	0	1	0	0	0
1	1	1	0	1	1

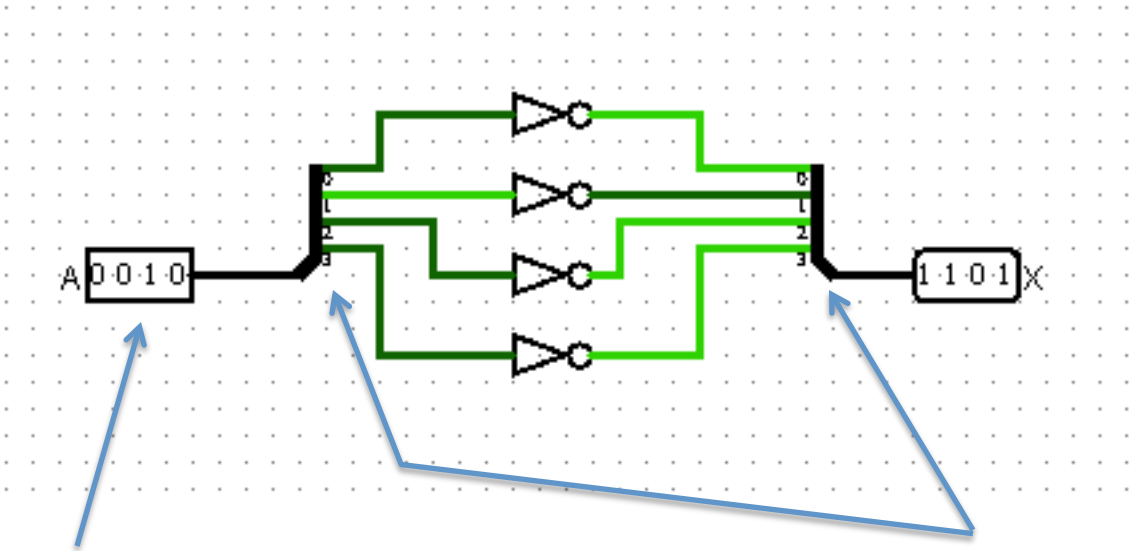
Porte a più bit / salvataggio

- In Logisim è possibile definire delle porte di ingresso con più di un bit. Questa funzionalità è comoda per simulare, ad esempio, l'elaborazione di un byte.
- E' inoltre possibile salvare i circuiti progettati per poterli riutilizzare in futuro.

Es. 4

- Si costruisca con Gatesim un circuito che calcoli il complemento a 1 di una sequenza di 4 bit (il complemento a 1 si ottiene semplicemente invertendo il valore dei singoli bit).
- Si salvi il circuito sviluppato con il nome di C1_4bit.

Soluzione

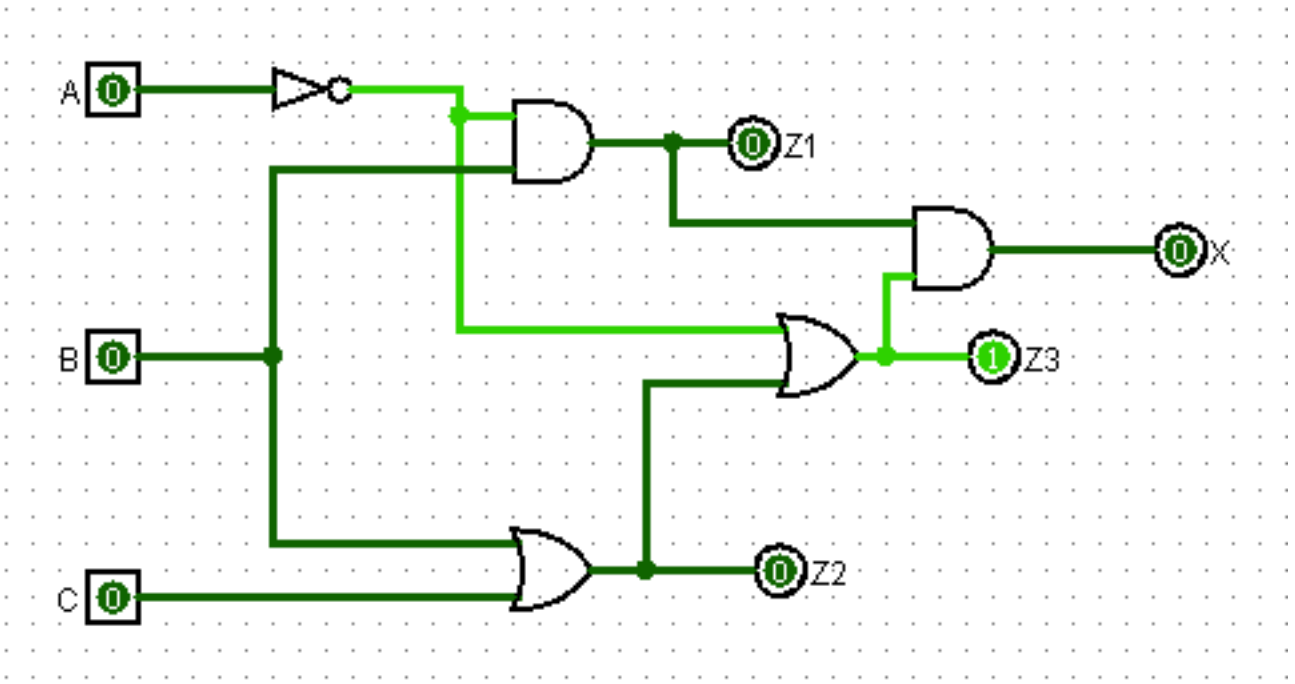


- Il numero di input di un ingresso o di un uscita si può settare nella finestra proprietà in basso a sinistra
- Il numero di linee all'interno di una connessione (bus) è definito automaticamente dalla porta in uscita a cui è collegato.

- Lo splitter permette di suddividere o ragruppare un bus in gruppi distinti di bit
- Il tipo di ragruppamento si definisce scegliendo il numero di linee del bus in ingresso (4 in questo caso) ed il numero di bus in uscita (4 in questo caso)

Es. 5

Si ricavi la tabella di verita del seguente circuito e se ne verifichi la correttezza



Soluzione

- Abbiamo tre ingressi, che chiameremo A, B e C. Per calcolare la tabella che descrive l'uscita X del circuito, calcoliamo prima i "risultati" parziali delle operazioni...
- Z1 = not(A) and B
- Z2 = B or C
- Z3 = not(A) or Z2
- X = Z1 and Z3 = (not(A) and B) and (not(A) or Z2) = (not(A) and B) and (not(A) or (B or C))

Soluzione

A	B	C	Z1	Z2	Z3	X
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Soluzione

A	B	C	Z1	Z2	Z3	X
0	0	0	0			
0	0	1	0			
0	1	0	1			
0	1	1	1			
1	0	0	0			
1	0	1	0			
1	1	0	0			
1	1	1	0			

Soluzione

A	B	C	Z1	Z2	Z3	X
0	0	0	0	0		
0	0	1	0	1		
0	1	0	1	1		
0	1	1	1	1		
1	0	0	0	0		
1	0	1	0	1		
1	1	0	0	1		
1	1	1	0	1		

Soluzione

A	B	C	Z1	Z2	Z3	X
0	0	0	0	0	1	0
0	0	1	0	1	1	0
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	0	1	1	0

L’uscita va a 1 solo quando A=0 e B=1, indipendentemente dal valore di C.

Manipolazioni algebriche

- **Proprietà generali dell'Algebra Booleana:**
-
- **Doppia Inversione** $\sim (\sim x) = x$
- **Identità**: $1\,x = x \quad 0 + x = x$
- **Elemento nullo**: $0\,x = 0 \quad 1 + x = 1$
- **Idempotenza**: $x\,x = x \quad x + x = x$
- **Inverso:** $x \sim x = 0 \quad x + \sim x = 1$
- **Commutativa**: $x\,y = y\,x \qquad x + y = y + x$
- **Associativa:** $(x\,y)\,z = x\,(y\,z) \qquad (x + y) + z = x + (y + z)$
- **Distributiva:** $x\,(y + z) = x\,y + x\,z \qquad x + y\,z = (x + y\,)\,(x + z)$
- **Assorbimento:** $x\,(x + y) = x \qquad x + x\,y = x$
- **De Morgan**: $\sim (x\,y) = \sim x + \sim y \qquad \sim (x + y) = \sim x \;\; \sim y$

Manipolazioni algebriche

- **Precedenza degli operatori logici**:

  In assenza di parentesi, AND ha la priorità sull'OR ed il NOT su entrambi:

  NOT > AND > OR

- **Principio di dualità**:

  Il duale di una funzione si ottiene sostituendo:
  AND con OR,OR con AND, 0 con 1 ed 1 con 0.

Es. 6 (manipolazioni algebriche)

- **Si dimostri che:**

  (A + ~B)(B + C) = AB + AC + ~BC.

Es. 6 (soluzione)

(A + ~B)(B + C) =

(AB)+(AC)+[(~B)B]+(~BC) = ...

(~B)B = 0

... = AB + AC + ~BC.

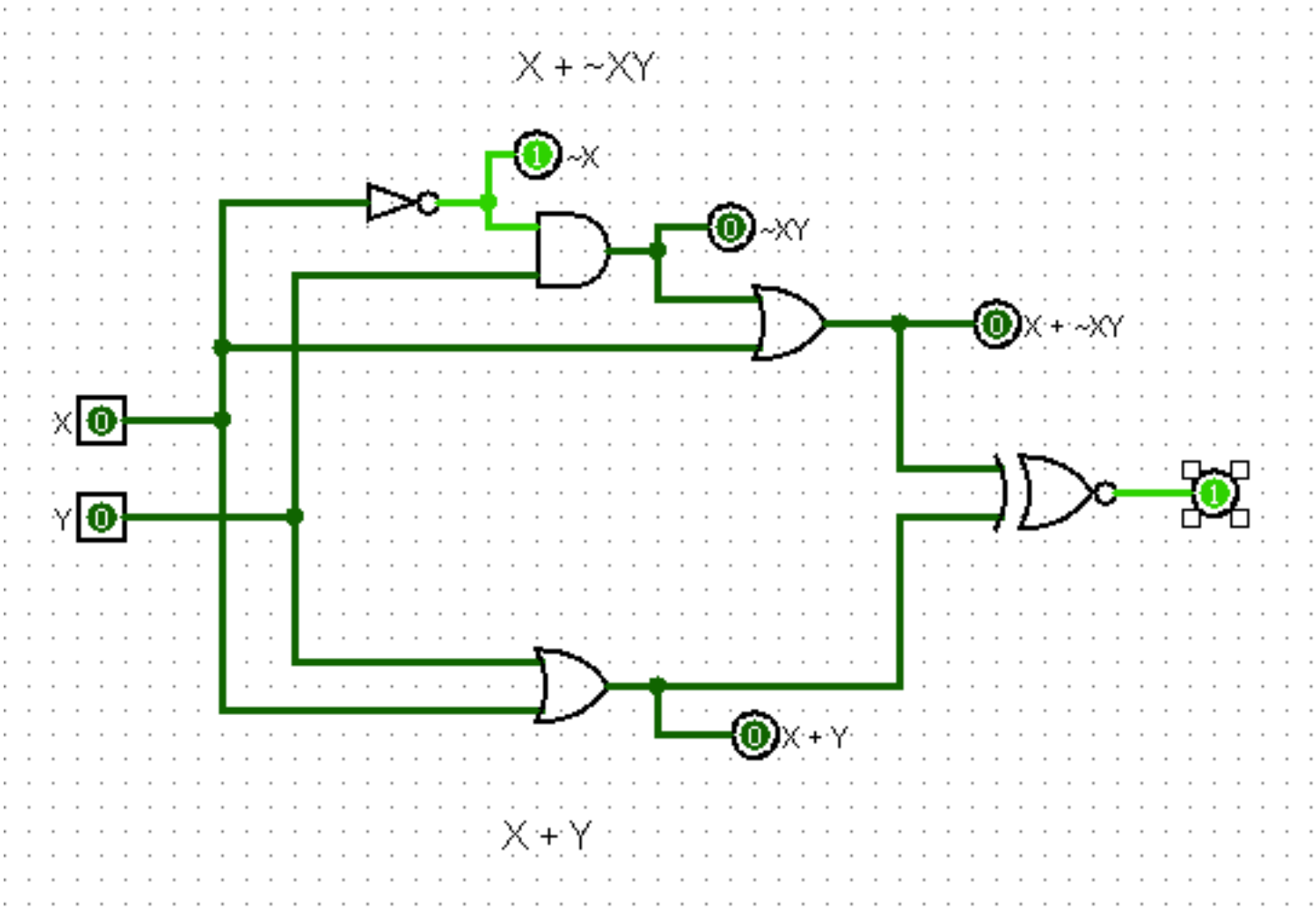
[QED]

Es. 7 (manipolazioni algebriche)

- **Si dimostri che x + ~xy = *x* + *y*.**
- **Si implementino in Logisim i due circuiti corrispondenti a x + ~xy e *x* + *y* e si verifichi la correttezza del risultato.**

Es. 7 (soluzione)

x + ~xy = (x + ~x)(x + y) = 1 (x + y) = x + y. [QED]



Es. 8

- **Usare la sola porta NAND per realizzare la funzione (A or (not(B))) and not(C).**
- **Realizzare lo stesso circuito utilizzando la sola porta NOR.**

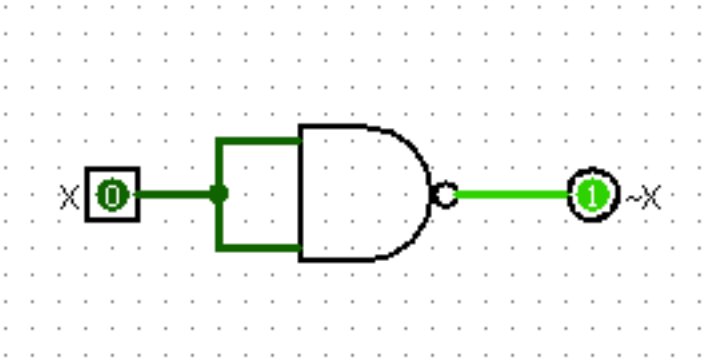
Es. 8 (Soluzione)

- Utilizziamo la sola porta NAND per realizzare la negazione...

  X AND X = X

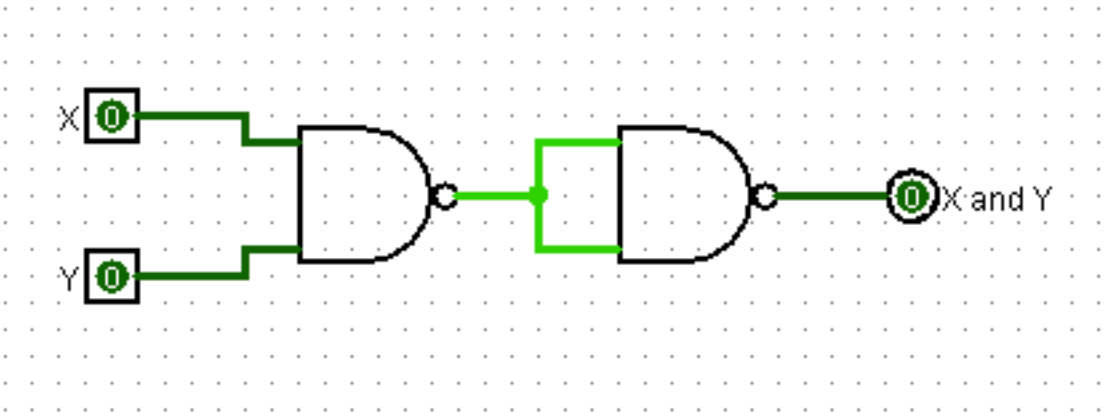
  quindi

  X NAND X = NOT X



Es. 8 (Soluzione)

- Per realizzare la porta AND, basta quindi negare l'uscita della porta NAND...



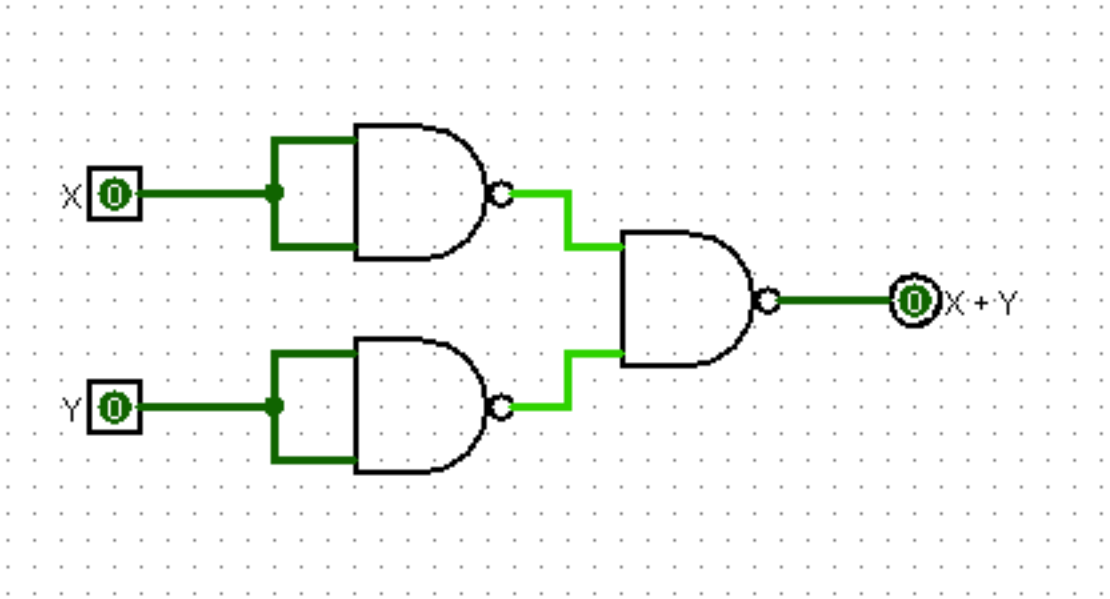
Es. 8 (Soluzione)

- Per realizzare la porta OR, utilizziamo De Morgan:

  X nand Y =

  not (X and Y) =

  not(X) or not(Y)



- quindi

  (nand(X,X)) nand (nand(Y,Y)) = X or Y