Esercitazione 2 – Introduzione a GATESIM

Gatesim (Logic Gate Simulator) è un simulatore di circuiti logici scritto in in C#/WPF (.NET 3.5 SP1) che permette di creare e simulare semplici circuiti costituiti da porte logiche. Lo stato dei collegamenti tra le porte logiche durante la simulazione è reso graficamente attraverso i colori: bianco per indicare valore logico falso (livello di tensione basso), rosso per indicare valore logico vero (valore di tensione alto). Gatesim simula anche il ritardo di propagazione attraverso le porte quindi ritardando opportunamente le porte è possibile seguire visivamente la convergenza del circuito.

I circuiti creati possono essere salvati in moduli da riutilizzare in circuiti più complessi.

Gatesim può essere scaricato al seguente URL:

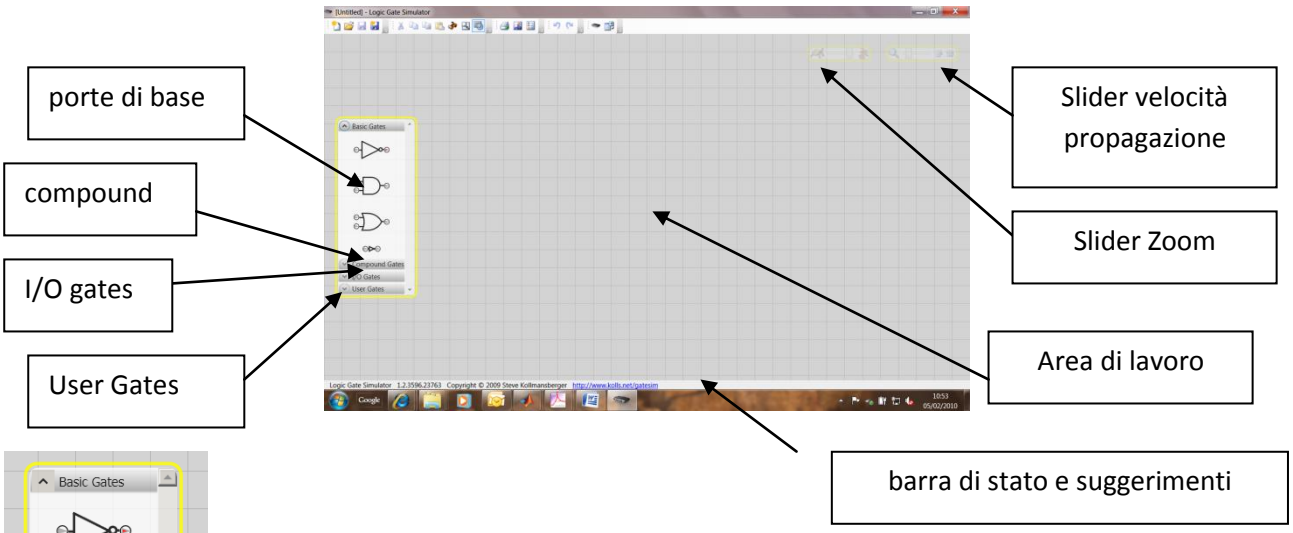
http://www.kolls.net/gatesim/

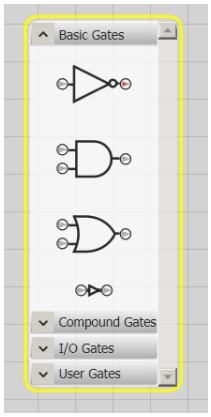
Nello stesso sito trovate un video introduttivo (in inglese) sul funzionamento di Gatesim (porte di input/output, porte loghce, creazione di circuiti personalizzati, velocità di propagazione all'interno del circuito):

http://www.kolls.net/gatesim/gatesim%20demo.swf

Costruire e simulare circuiti con Gatesim

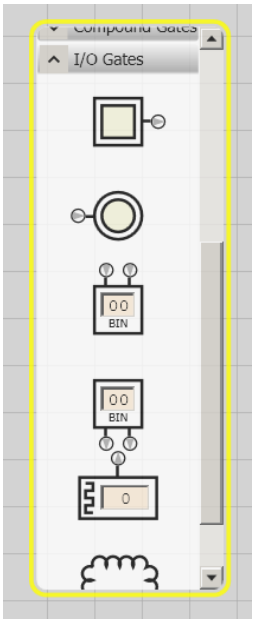
La schermata iniziale di Gatesim:





Nel menù a sinistra è possibile accede alle **porte di base** (Basic Gate: NOT, AND, OR & Buffer). Per inserire le porte all'interno di un circuito occorre trascinarle sull'area di lavoro centrale. Con il tasto destro si possono ruotare i componenti.

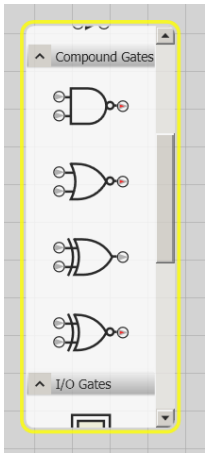
Sempre nel menù a sinistra, oltre alle porte logiche di base, sono presenti dei dispositivi di Input/Output per imporre livelli logici agli ingressi del circuito o per visualizzare i livelli in uscita (I/O Gates).



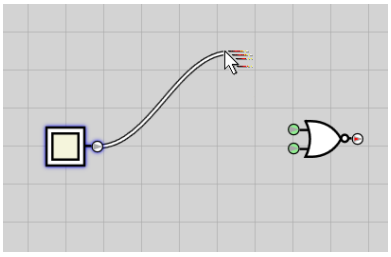
La **porta di input** permette di inserire un segnale true/false. La **porta di output** permette di leggere un segnale true/false. Le porte di input/output possono essere rinominate utilizzando il tasto destro del mouse.

Nel menù I/O gates sono disponibili anche **selettori binari ed indicatori binari** che permettono di avere input ed output su n bit. Al momento del trascinamento sull'area di lavoro viene chiesto di quanti bit deve essere composto l'indicatore/selettore. Cliccando sulla scritta "BIN" è possibile cambiare la base di visualizzazione in decimale, ottale o esadecimale.

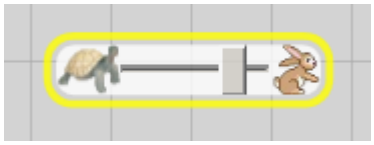
Infine si possono inserire anche **generatori di clock**. Il periodo di clock può essere personalizzato ed è espresso in millisecondi (usate periodi maggiori di 500ms per evitare problemi di visualizzazione).



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

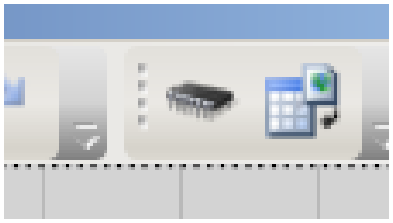


Le porte possono essere **connesse** tra di loro semplicemente trascinando con il mouse (tasto sinistro premuto) i connettori di input/output. Da un connettore di output possono uscire più fili mentre in un connettore di input può entrare un solo filo. Se cliccate su un filo già presente lo eliminate.



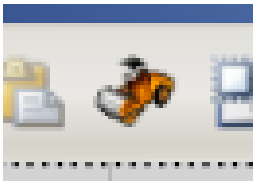
In alto a sinistra è presente uno **slider per il controllo della velocità di propagazione** dei segnali.

Di fianco è presente lo **slider dello zoom** per regolare l'ingrandimento dell'area di lavoro



La barra dei menù mette a disposizione alcune funzionalità di utilità. Oltre alle icone per aprire e salvare file è presente un'icona per registrare il circuito attuale sotto forma di "**chip**", o modulo User Gate. I moduli creati compaiono nel menù User Gates. e possono essere trascinati sull'area di lavoro come le altre porte.

Un altra funzione di utilità è il "**Flatten Circuit**" che permentte di sostituire i moduli con il loro circuito in modo da avere un circuito composto con sole porte logiche.

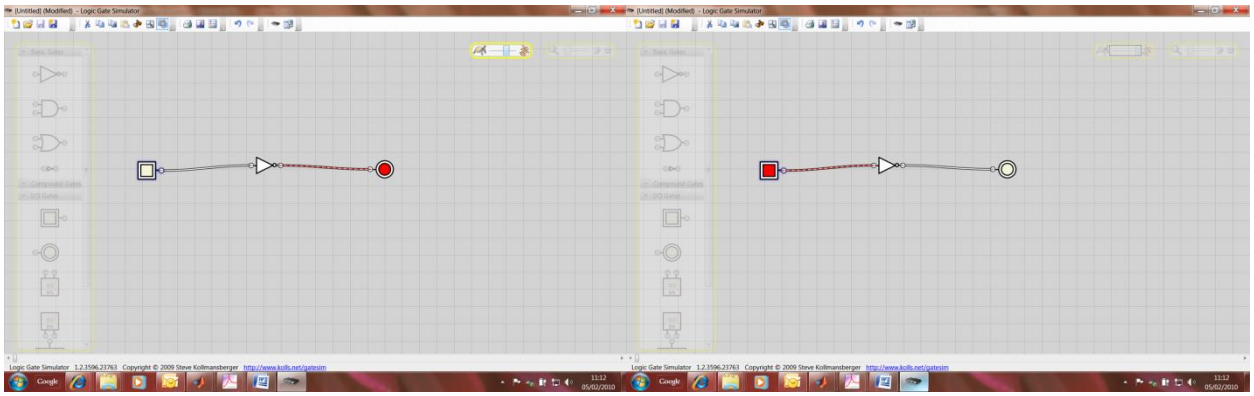


Esercizi con Gatesim

Esercizio 1:

Si disegni il circuito che realizza l'operazione di negazione di un segnale chiamato A in Gatesim e verificarne il corretto funzionamento. Verificare gli effetti dello slider della velocità.

Soluzione:



Dal menù a sinistra trascinare una porta di input, una porta NOT ed un indicatore di output nell'area di lavoro. Eseguire i collegamenti come da schema. Cliccando il tasto destro sulla porta di input è possibile impostare il nome. Cliccando sulla porta di input, il segnale viene messo a true/false e dopo un certo delay l'uscita si modifica di conseguenza. La velocità di propagazione del segnale può essere controllata con l'apposito slider.

Esercizio 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:

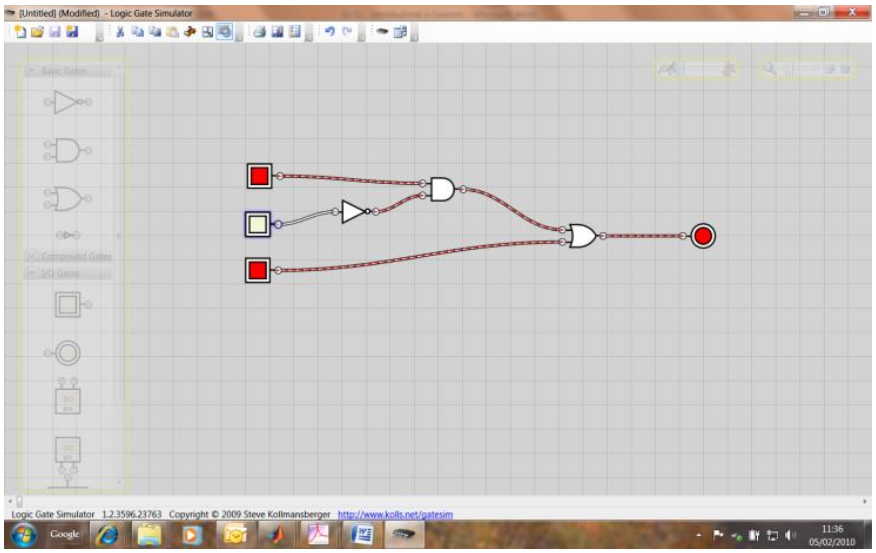
Per derivare la tabella della verità, dobbiamo considerare tutte le possibili combinazioni degli ingressi. Dobbiamo quindi calcolare l'output della funzione. E' utile calcolare i risultati intermedi e mettere anch'essi nella tabella.

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

Il circuito risultante dovrebbe essere del tipo:

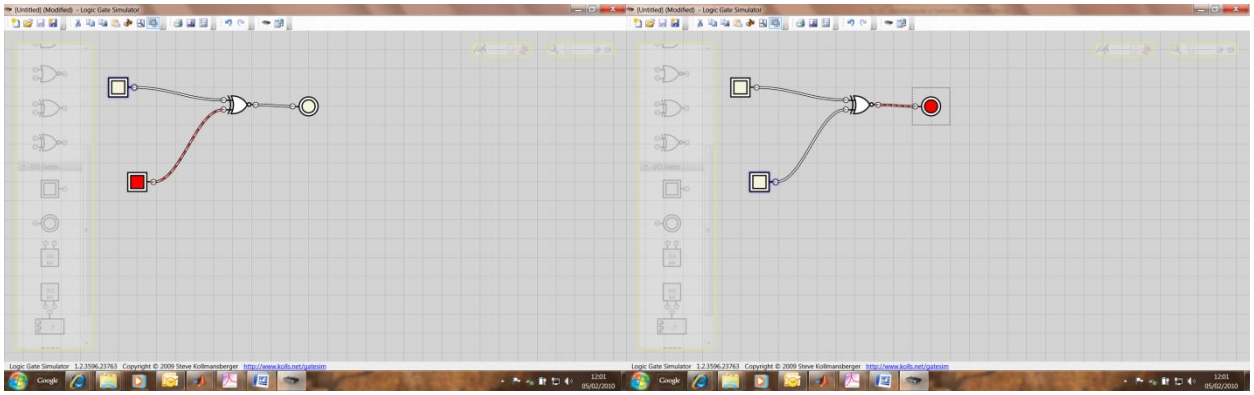


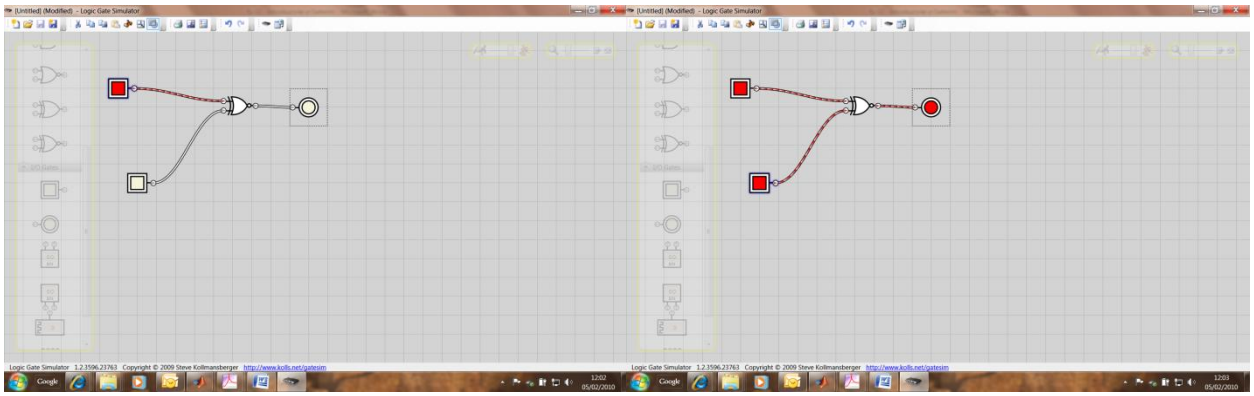
Esercizio 3

Si definiscano in Gatesim 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 esclusivamente le porte AND, OR e NOT. 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:

Analizziamo l'uscita della porta XNOR con gatesim:





Deriviamo quindi la tabelle di verità per XNOR:

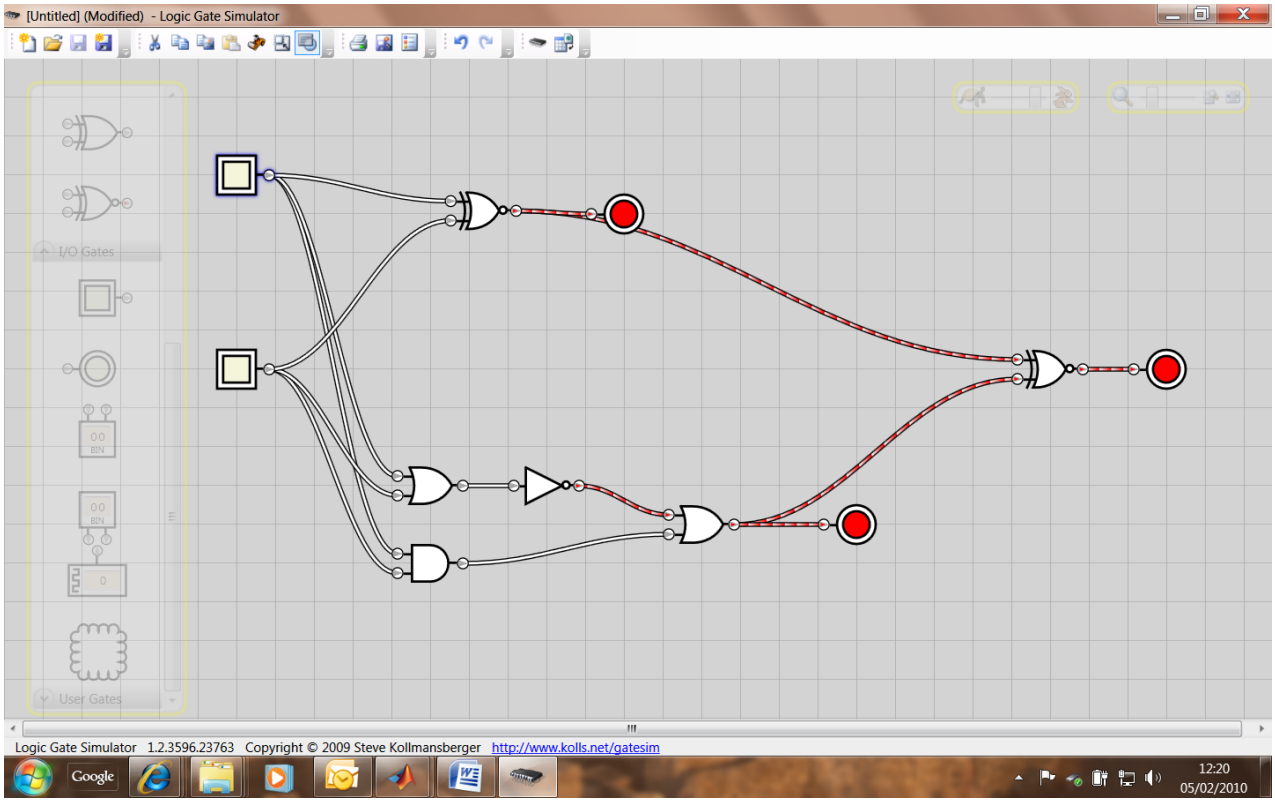
A	**B**	**A XNOR B**
0	0	1
0	1	0
1	0	0
1	1	1

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!

Proviamo a pensare come realizzare il circuito della XNOR (oggi utilizziamo l'intuito – vedremo più che esistono tecniche 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.

Verifichiamo il risultato con Gatesim:



La parte alta del cricuito è A XNOR B.

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

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.

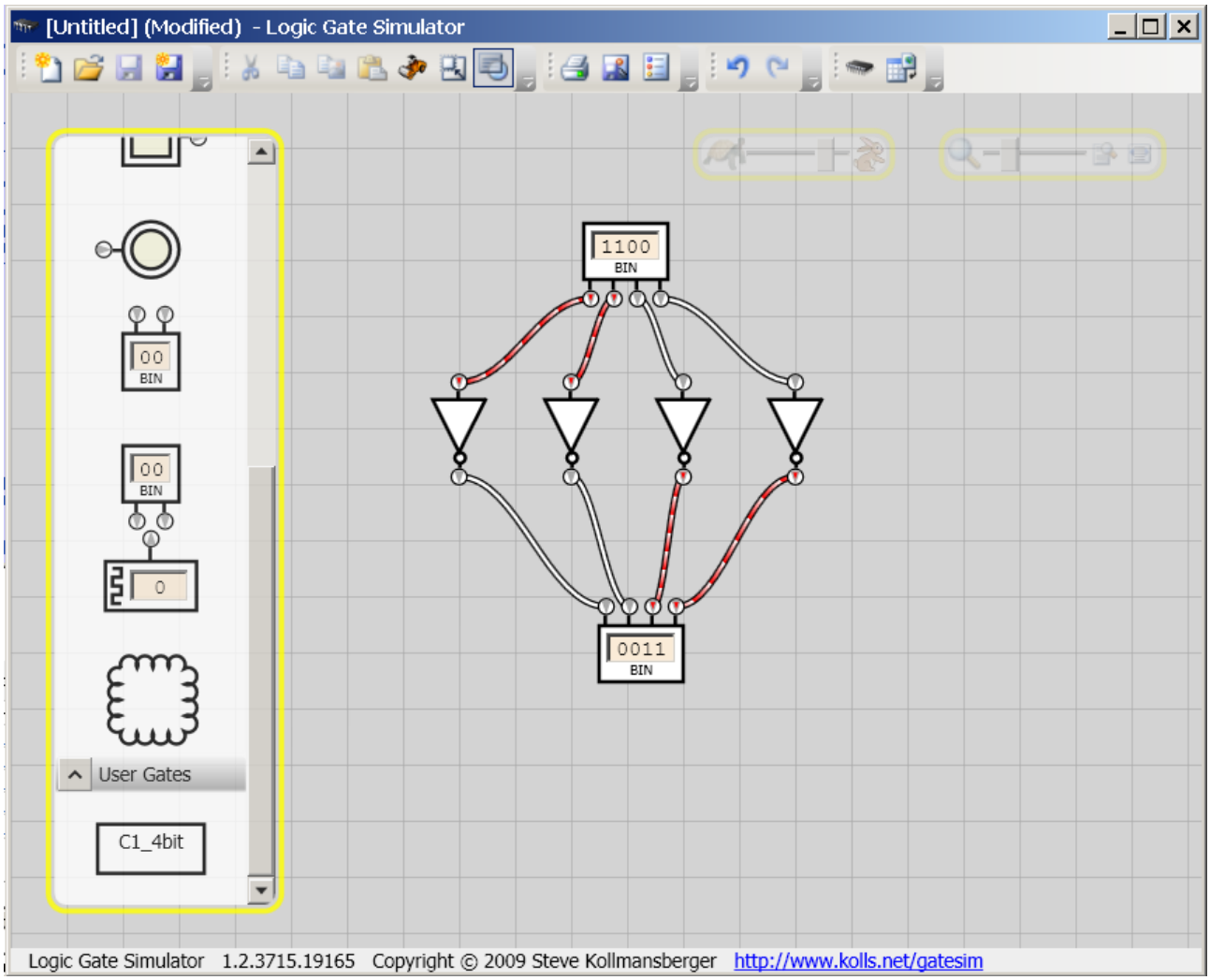
A	**B**	**Z1 = A OR B**	**Z2 = NOT Z1**	**Z3 = A AND B**	**X = Z2 OR Z3**
0	0	0	1	0	1
0	1	1	0	0	0
1	0	1	0	0	0
1	1	1	0	1	1

Come accennato all'inizione, in Gatesim 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.

Esercizio 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) e si salvi il circuito sviluppato con il nome di C1_4bit.

Soluzione:

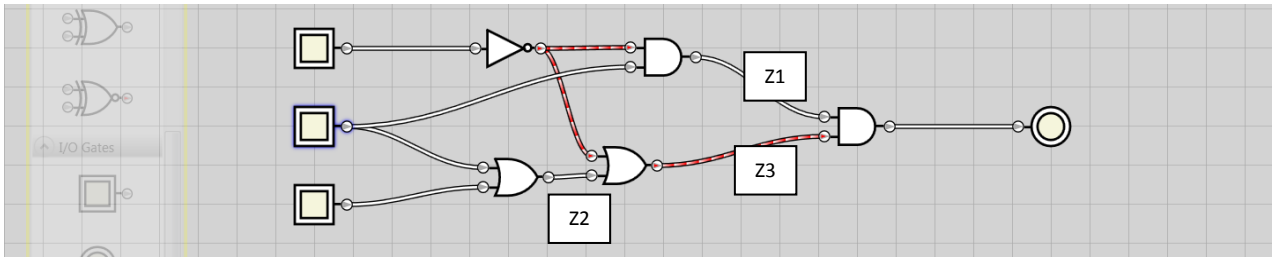


La soluzione proposta può essere salvata in Gatesim (Create IC), ma porta ad un circuito senza input ne output... Meglio utilizzare gli input/output singoli per salvare un circuito riutilizzabile.

Quando si inserisce un nuovo circuito -> tasto dx, in linea -> esplode il circuito.

Esercizio 5:

Si ricavi la tabella di verità 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))

A	**B**	**C**	**Z1**	**Z2**	**Z3**	**X**
0	0	0	0	0	0	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.

Esercizio 6:

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

Soluzione:

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

= A(B+C) + ~B(B+C) *(x+y)z=xz+yz*

= AB + AC + ~BB + ~BC *(x+y)z=xz+yz*

= AB + AC + 0 + ~BC *~xx=0*

= AB + AC + ~BC *x+0=x*

Esercizio 7:

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

Soluzione:

Y **= x + ~xy**

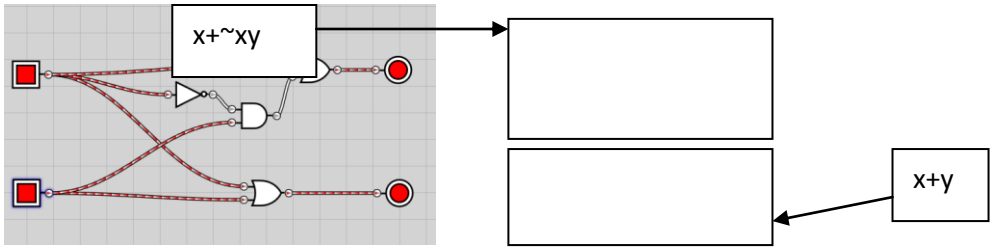
= x + xy + ~xy *x=x+xz*

= x + (x + ~x)y *xy+xz=x(y+z)*

= x + 1y *x+~x=1*

= x + y *1x =x*

Il circuito Gatesim:



Il secondo circuito è migliore del primo sia rispetto al cammino critico (2 per il primo, 1 per il secondo) sia rispetto al costo circuitale (2 per il primo, 1 per il secondo).

Esercizio 8:

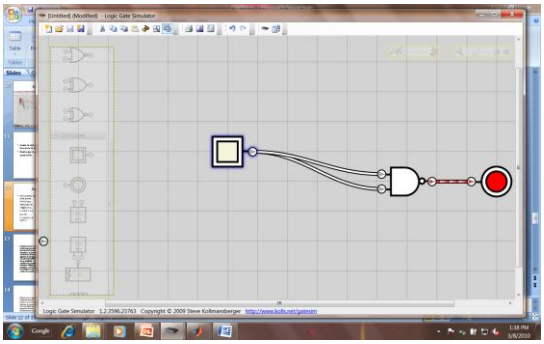
Es. 8 (Soluzione)

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

  X AND X = X

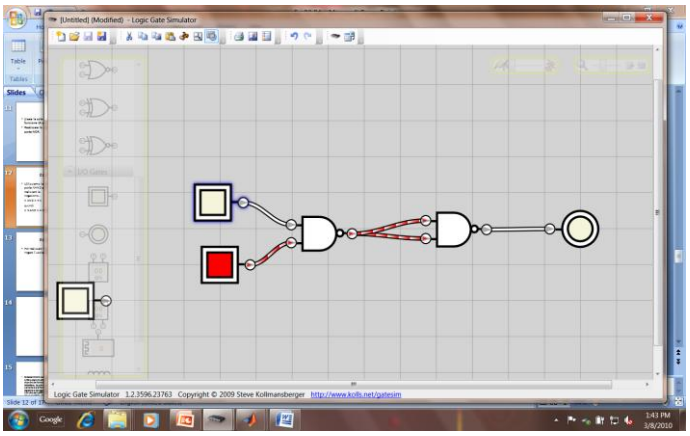
  quindi

  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

  not(X) nand not(Y) =

  X or Y