ITIS-LS "Francesco Giordani" Caserta

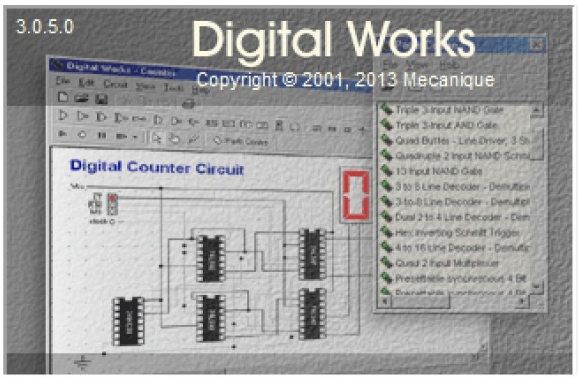
prof. Ennio Ranucci

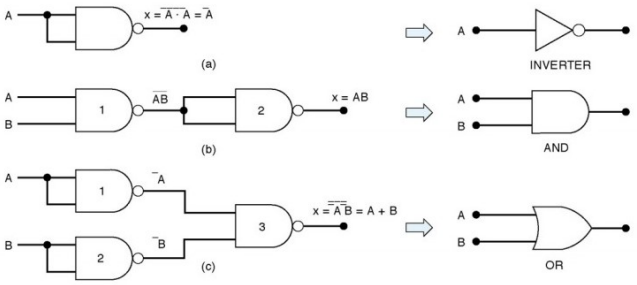
a.s. 2019-2020

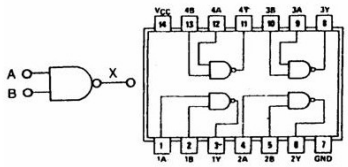
Semplici circuiti digitali

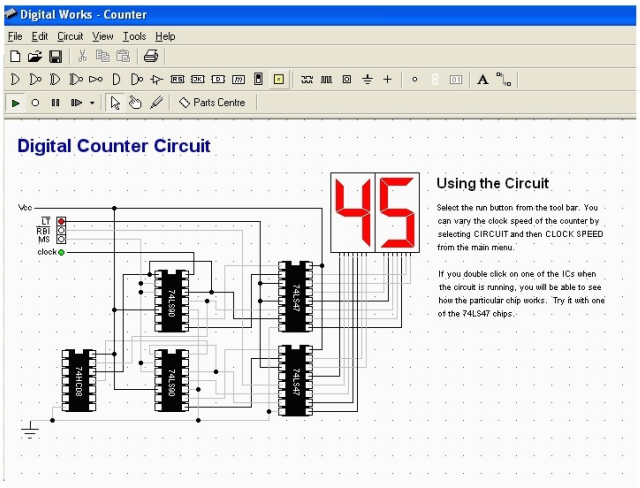
Esercitazioni in ambiente DigitalWorks

versione 3.0.5.0









Fonte Luca Andrea Ludovico “Informatica Generale”

OPERATORE NOT

Si tratta di un operatore unario, che accetta un unico ingresso e produce un’uscita. Rappresenta la negazione logica, o il complemento.

Equazione logica:

$z = \text{NOT } a$ [informatica]

$z = \overline{a}$ [algebra]

$z = \neg a$ [logica]

Simbolo circuitale:

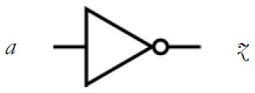


Tabella della verità:

a	z
0	1
1	0

OPERATORE AND

Si tratta di un operatore binario, che accetta due ingressi e produce un’uscita. Rappresenta il prodotto logico.

Equazione logica:

$z = a \text{ AND } b$ [informatica]

$z = a \times b = a \cdot b = a\,b$ [algebra]

$z = a \wedge b$ [logica]

Simbolo circuitale:

Tabella della verità:

a	b	z
0	0	0
0	1	0
1	0	0
1	1	1

OPERATORE OR

Si tratta di un operatore binario, che accetta due ingressi e produce un'uscita.
Rappresenta la somma logica.

Equazione logica:

$z = a \text{ OR } b$ [informatica]

$z = a + b$ [algebra]

$z = a \vee b$ [logica]

Simbolo circuitale:

Tabella della verità:

a	b	z
0	0	0
0	1	1
1	0	1
1	1	1

OPERATORE XOR

Si tratta di un operatore binario, che accetta due ingressi e produce un'uscita.
Rappresenta l'OR esclusivo (*aut aut*) ed è un operatore di disuguaglianza.

Equazione logica:

$z = a \text{ XOR } b$ [informatica]

$z = a \oplus b$ [algebra]

Simbolo circuitale:

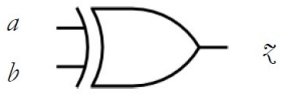


Tabella della verità:

a	b	z
0	0	0
0	1	1
1	0	1
1	1	0

OPERATORE NAND

Si tratta di un operatore binario, che accetta due ingressi e produce un'uscita.
Rappresenta la negazione del prodotto logico.

Equazione logica:

$$z = a \text{ NAND } b = \text{NOT } (a \text{ AND } b) \qquad \text{[informatica]}$$

$$z = \overline{a \times b} = \overline{ab} \qquad \text{[algebra]}$$

Simbolo circuitale:

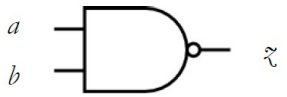


Tabella della verità:

a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

OPERATORE NOR

Si tratta di un operatore binario, che accetta due ingressi e produce un'uscita.
Rappresenta la negazione della somma logica.

Equazione logica:

$$z = a \text{ NOR } b = \text{NOT } (a \text{ OR } b) \qquad \text{[informatica]}$$

$$z = \overline{a + b} \qquad \text{[algebra]}$$

Simbolo circuitale:

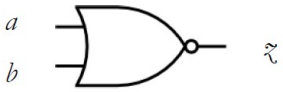


Tabella della verità:

a	b	z
0	0	1
0	1	0
1	0	0
1	1	0

OPERATORE XNOR

Si tratta di un operatore binario, che accetta due ingressi e produce un'uscita.
Rappresenta la negazione dell'OR esclusivo (*aut aut*) ed è un operatore di uguaglianza.

Equazione logica:

$z = a \text{ XNOR } b$ [informatica]

$z = a \otimes b$ [algebra]

Simbolo circuitale:

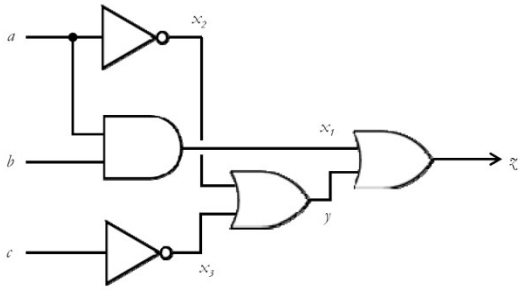
Tabella della verità:

a	*b*	*z*
0	0	1
0	1	0
1	0	0
1	1	1

Data un'equazione logica, è sempre possibile fornire una rappresentazione grafica dello schema circuitale sotteso. Analogamente, è possibile partire dallo schema circuitale per ricavare la corrispettiva equazione logica.
Nelle nostre convenzioni grafiche, i segnali si propagano da sinistra verso destra, pertanto gli ingressi saranno posti sul lato sinistro e le uscite su quello destro.
Volendo rappresentare lo schema circuitale di un'equazione logica, si parte dagli operatori con massima priorità che agiscono direttamente sugli ingressi, per poi procedere nel disegnare gli operatori con priorità più bassa, fino a giungere all'uscita o alle uscite.
Si ricorda che l'operatore con priorità più alta è il NOT, seguito dall'AND e infine dall'OR. Per modificare l'ordine naturale di valutazione degli operatori, si possono introdurre le parentesi tonde. Le versioni negate *a* NAND *b* e *a* NOR *b* ai fini della priorità possono essere rispettivamente considerate come NOT (*a* AND *b*) e NOT (*a* OR *b*).

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es1*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Si fornisca una rappresentazione grafica del circuito relativo alla seguente equazione logica:
$z = ab + (a + c) = a$ AND b OR (NOT(a) OR NOT(c))
Soluzione
I primi operatori da valutare sono
$x1 = a$ AND b
$x2$ = NOT(a)
$x3$ = NOT(c)
Al secondo livello viene valutato $y = x2$ OR $x3$.
Al terzo livello, si giunge infine a $z = x1$ OR y.

Realizzare e verificare il circuito con Digital Works (costruire la tabella di verità)

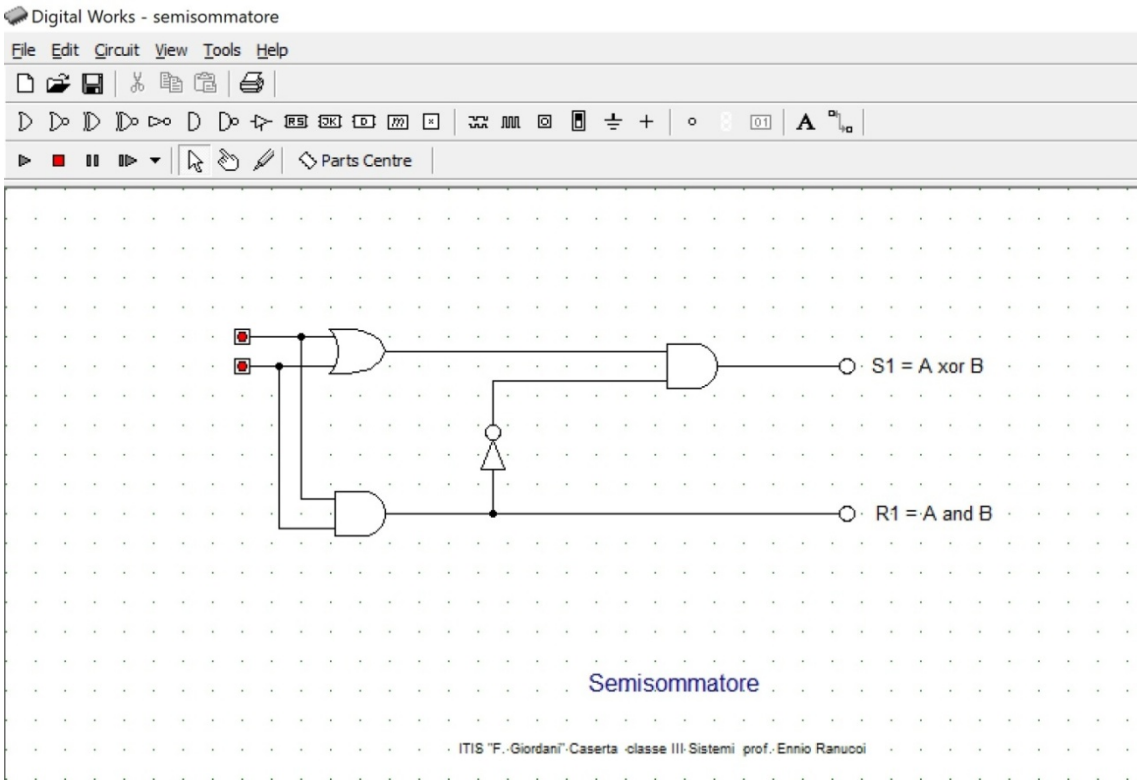


ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es2*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Realizzare il semisommatore

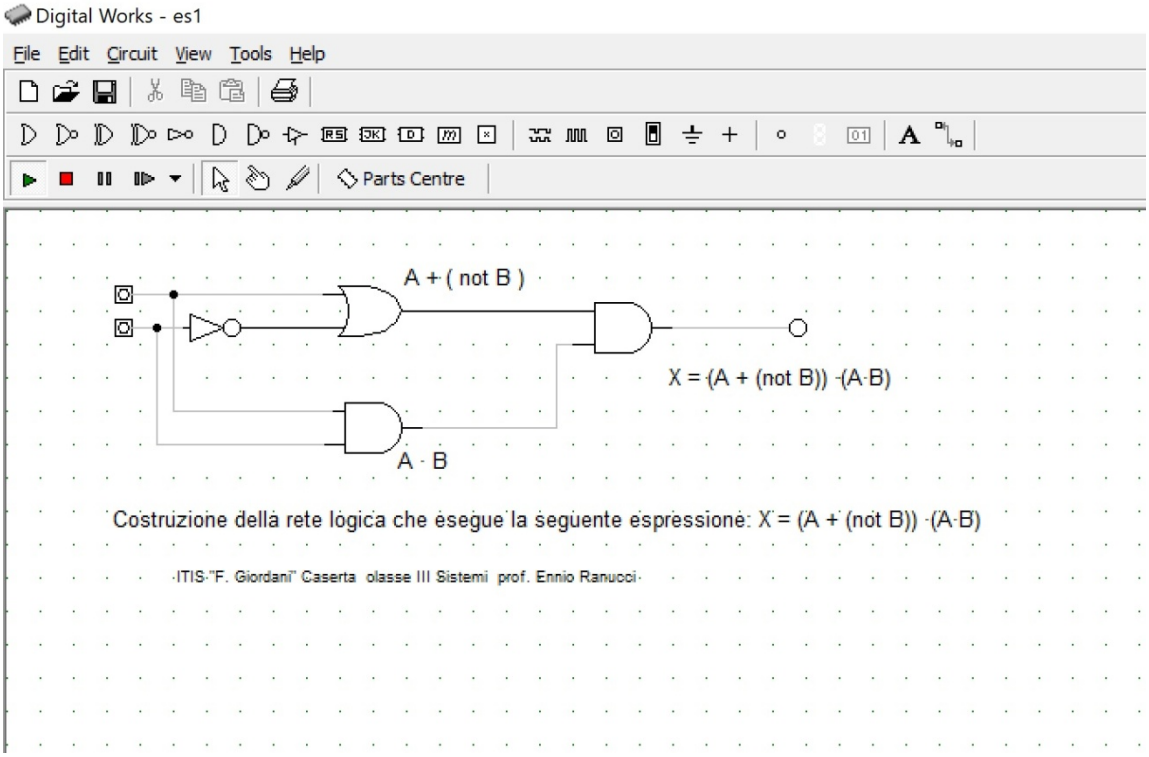
SOMMA

+	**0**	**1**
0	0	1
1	1	10

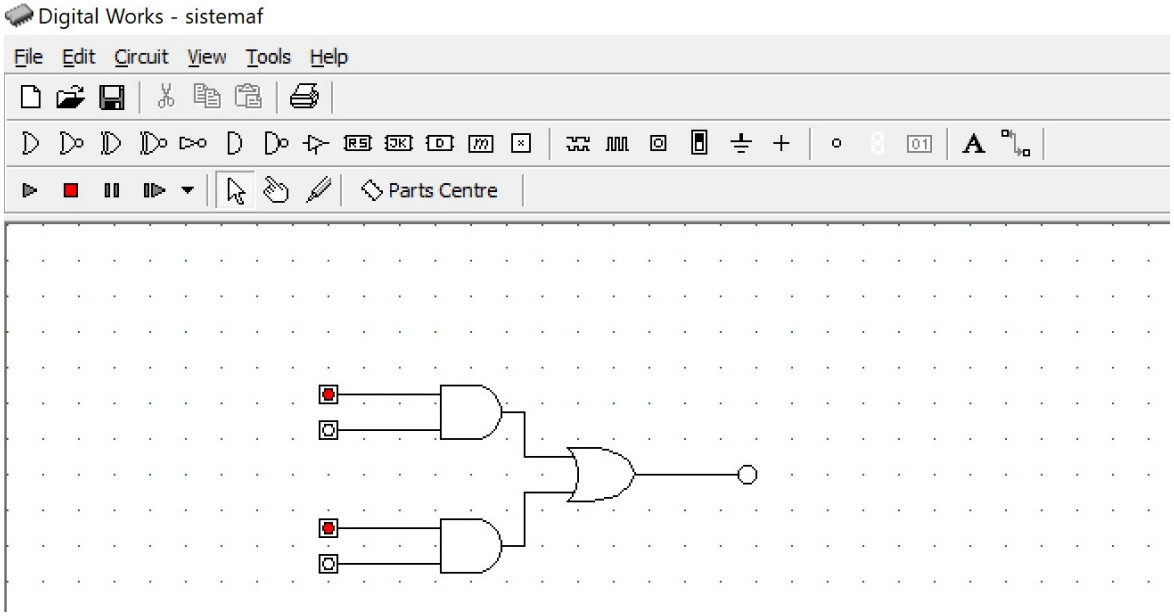
ovvero 0 con riporto di 1



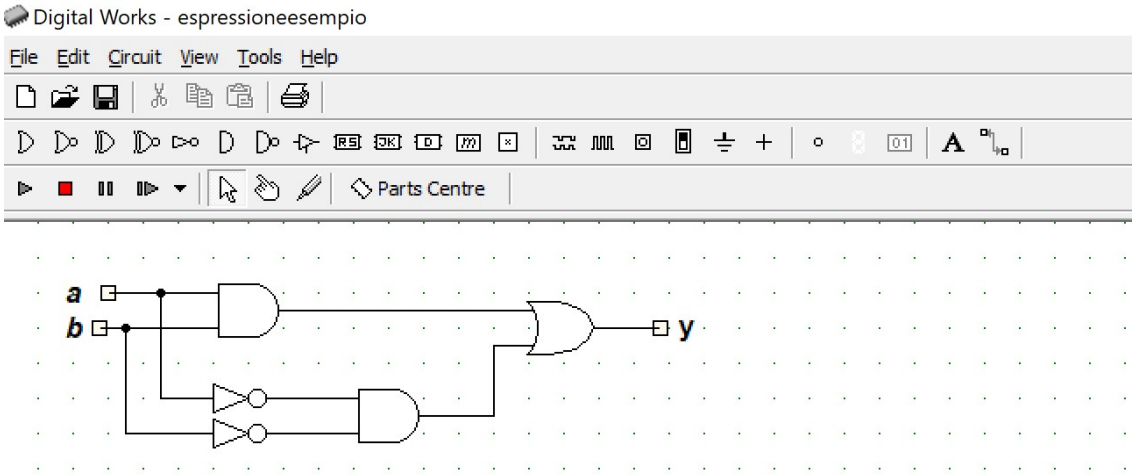
ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es3*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Costruire la rete logica dell'espressione descritta in figura



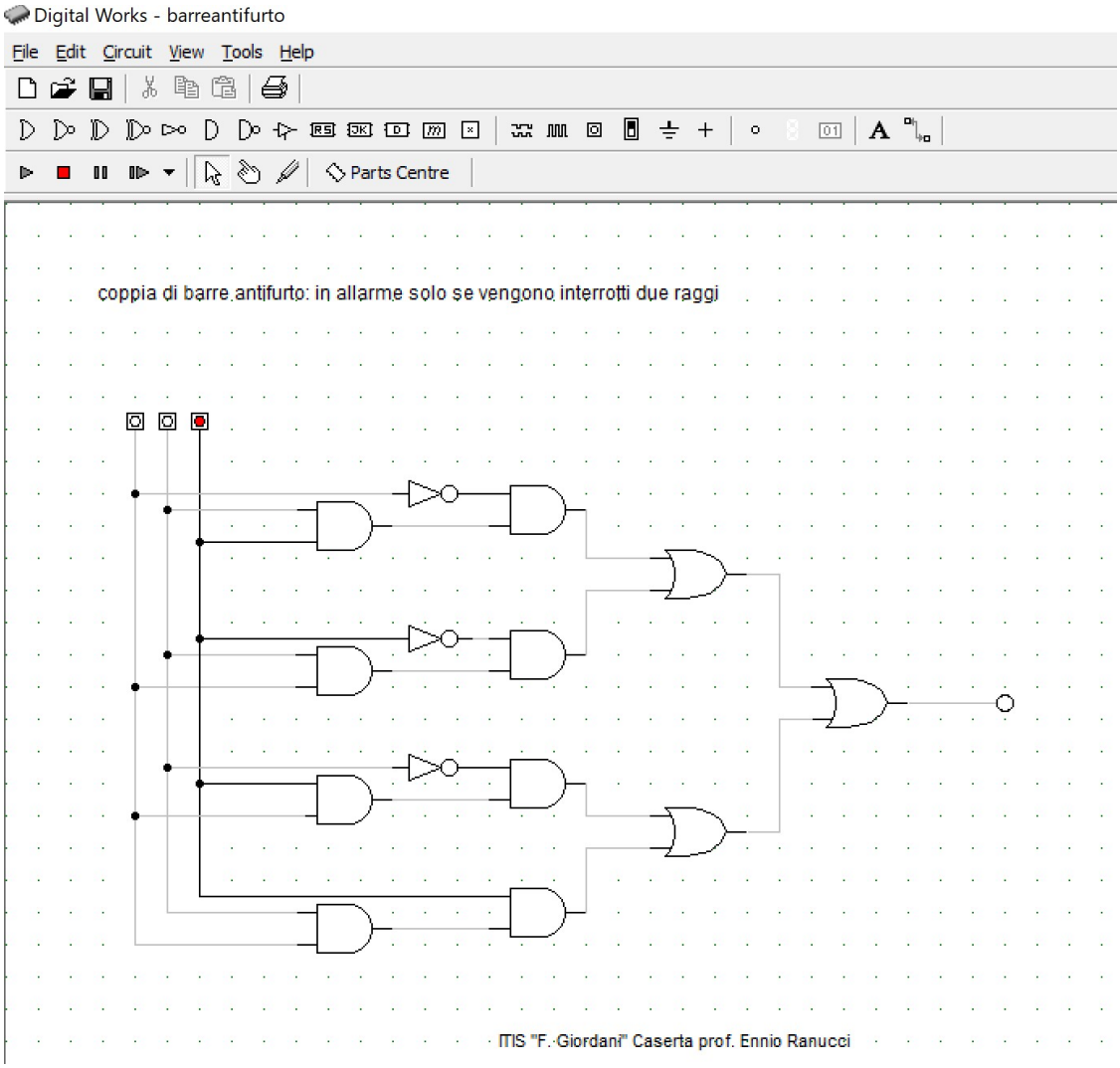
ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es4*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Scrivere l'espressione logica che rappresenta il circuito in figura



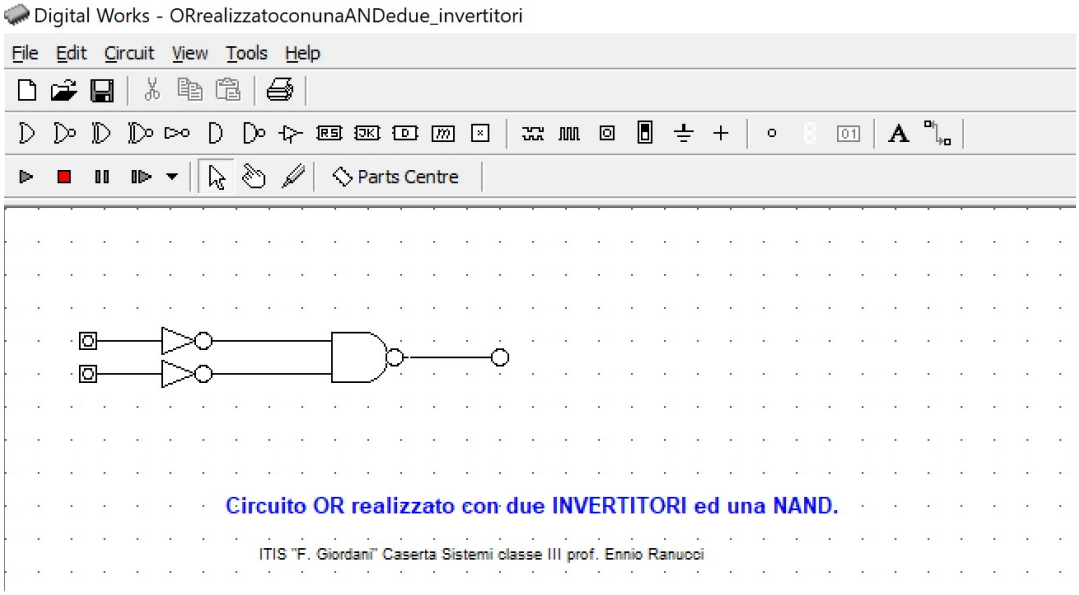
ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es5*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Scrivere la tabella di verità del circuito logico



ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es6*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Realizzare una coppia di barre antifurto in allarme solo se vengono interrotti due raggi



ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es7*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *DigitalWorks versione 3.0.5.0*
Obiettivo didattico:
La logica ed i circuiti
Obiettivo del programma:
Realizzare la porta OR mediante due inveritori ed una NAND



Algebra di Boole e i suoi modelli:

Algebra delle Proposizioni e Algebra dei Circuiti

La logica delle proposizioni, l'algebra dei circuiti,

simulazione di circuiti digitali con "Digital Works"

I.T.I.S. "Francesco Giordani" Caserta

Specializzazione informatica ABACUS

prof. Ennio Ranucci

- *L'algebra booleana è un sistema formale logico-deduttivo e, come ogni sistema formale, parte da alcuni concetti fondamentali e da un insieme di postulati.*
  Un esempio di sistema formale logico – deduttivo è la geometria euclidea.

- *La logica delle proposizioni*
  La logica è la scienza che si occupa delle strutture formali del ragionamento. Oggetto della logica è il linguaggio costituito dalle proposizioni logiche che godono della proprietà di essere vere o false e un caso esclude l'altro (tertium non datur).
  Sono, per esempio, proposizioni logiche:
  <<Caserta è una città della Campania>>,
  <<17 è un numero>>,
  << Napoli è la capitale d'Italia>>.
  Le prime due sono proposizioni logiche vere, la terza è falsa.
  *Non sono proposizioni logiche le interrogazioni (*hai studiato?*), le esclamazioni (*che bel libro!*), ecc., perché non si può esprimere un giudizio di verità o falsità.*

  Pertanto se "p" è una proposizione, si ha:
  p=1 se p è vera

p=0 se p è falsa.
Una proposizione può essere:

a) semplice,
b) composta.
Il concetto di proposizione semplice è primitivo. Una proposizione è composta quando è ottenuta da due o più proposizioni semplici collegate fra loro da particolari operatori detti connettivi *quali "e", "o", "se…allora".*

Esempio

"La Reggia di Caserta è stata proclamata Patrimonio dell'umanità dall'UNESCO" è una proposizione semplice.

"La Reggia di Caserta è stata proclamata Patrimonio dell'umanità dall'UNESCO" e "Luigi Vanvitelli era un architetto" è una proposizione composta.

Date due proposizioni x *e* y *chiamiamo disgiunzione logica o addizione logica una operazione che opera su* x *e* y *producendo una proposizione* z *che è vera se almeno una delle due proposizioni è vera.*

Si indica:

$$z = x \vee y \text{ } \textit{oppure} \text{ } z = x + y$$

*e si legge <<*x *oppure* y*>>*

[logica matematica ($\vee$), informatica (**OR**), italiano (**O**), latino (**VEL**)]

Poiché x *e* y *possono assumere solo i valori: 0,1, si può scrivere per* z *la tavola di verità:*

x	y	$z = x \vee y$
0	*0*	*0*
1	*0*	*1*

Date due proposizioni x *e* y *chiamiamo congiunzione logica o prodotto logico una operazione che opera su* x *e* y *producendo una proposizione* z *che è vera se* x *e* y *sono entrambe vere.*

Si indica:

$$z = x \wedge y \text{ } \textit{oppure} \text{ } z = x \cdot y$$

*e si legge <<*x *e* y*>>*

[logica matematica ($\wedge$), informatica (**AND**), italiano (**E**), latino (**ET**)]

Tavola di verità della congiunzione logica:

x	y	z = x ∧ y
0	*0*	*0*
1	*0*	*0*

La negazione logica di una proposizione logica x *è una proposizione* ¬x *(*e si legge non x*) che è falsa se* x *è vera e viceversa.*

x	¬x
0	*1*
1	*0*

*[logica matematica (¬), informatica (**NOT**), italiano (**NON**), latino (**NON**)]*

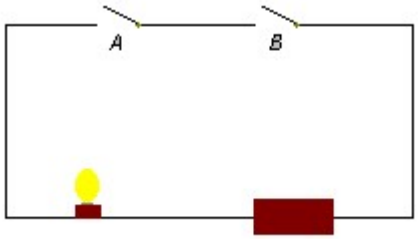
Date due proposizioni x *e* y *chiamiamo somma logica esclusiva una operazione che opera su* x *e* y *producendo una proposizione* z *che ha valore 1 se* x *oppure* y, *ma non entrambe, hanno valore 1.*

Si indica:

$$z = x \oplus y$$

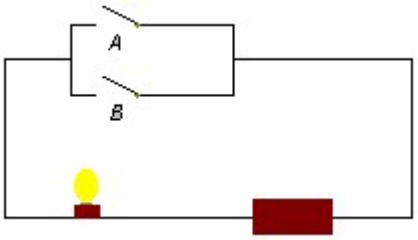
[informatica (**XOR**), latino **(AUT)**]

- L’algebra dei circuiti
  Osservando il circuito elettrico schematizzato in figura,



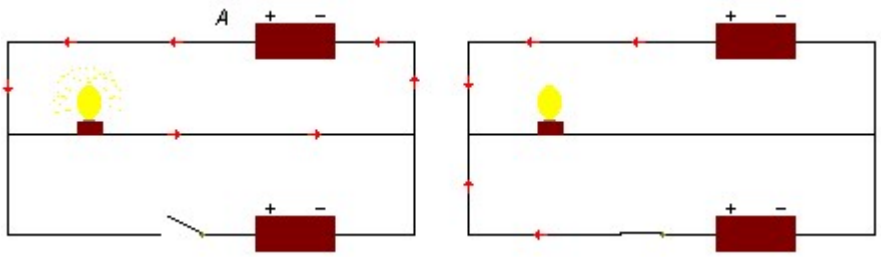
è facile riconoscere che la lampadina si accenderà solo se entrambi gli interruttori *A* e *B* verranno abbassati in modo da chiudere il contatto elettrico. Questo circuito, corrisponde ad un operatore logico AND.

Ora, come secondo passo, esaminiamo un altro circuito:



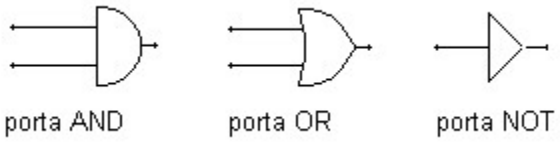
in questo caso, è facile riconoscere che la lampadina si accenderà solo se almeno uno degli interruttori *A* e *B* verrà abbassato in modo da chiudere il contatto elettrico. Questo circuito, corrisponde all'operatore logico OR.

Il circuito NOT inverte il segnale in ingresso: se vale 1, diventa 0 e viceversa.

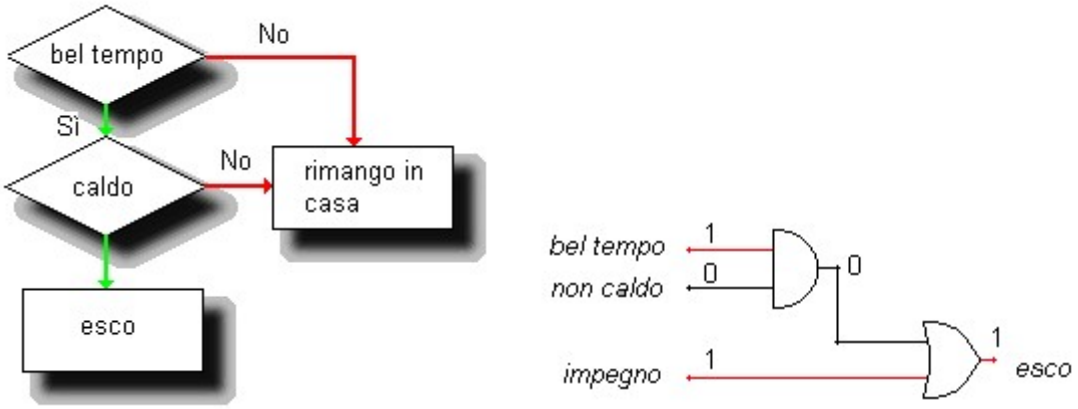


Il circuito NOT comprende due alimentatori con polarità opposta. Quando il circuito inferiore è aperto (0), la batteria *A* alimenta il circuito superiore e la lampadina è accesa; quando il circuito inferiore è chiuso, la seconda batteria fornisce una corrente uguale e opposta che impedisce il passaggio di corrente per cui la lampadina si spegne.
La tavola di veridicità corrispondente a questo circuito, comprende un solo ingresso.

Le porte logiche descritte, sono rappresentate simbolicamente con i simboli in figura:



combinandoli fra loro, si può codificare in linguaggio binario qualsiasi diagramma di flusso.
Per esempio, la frase "se è bel tempo ed è caldo esco; tuttavia, se ho un impegno esco in ogni caso" richiede una porta AND ed una porta OR unite tra loro come in figura:



La combinazione di più porte logiche, permette di ottenere risultati più articolati.

Per esempio, nella figura sotto è mostrata una porta NOR, costituita dalla combinazione di due porte AND, due porte NOT ed una porta OR. Questa porta permette di selezionare un valore positivo (1) se e solo se uno dei due dati in ingresso è positivo (a differenza della porta OR che fornisce un valore unitario anche se entrambi i dati in ingresso sono positivi).

Quello che abbiamo esaminato finora, può definirsi un semplice elaboratore dedicato, capace cioè di eseguire un compito particolare: verificare la presenza di due eventi (bel tempo, caldo) o di un evento alternativo (impegno). E' ovvio che un simile elaboratore è troppo semplice per avere qualche utilità. Tuttavia, si può facilmente intuire che gli elementi discussi costituiscono la base per combinare tra loro più porte logiche in modo da ottenere rapidamente, e senza confusione, decisioni immediate. Inoltre, si possono costruire circuiti in grado di effettuare operazioni matematiche.

Esempi di reti logiche simulate con Digital Works

- *Digital Works (Windows, freeware)*

Esperienze di simulazione di circuiti combinatori con Digital Works

(dopo aver scaricato aprirli con digital works *)*

- *Impiego di un circuito NAND come INVERTITORE*
- *Impiego di un circuito XOR come invertitore controllato*

- *Circuito OR realizzato con un NAND e due INVERTITORI*
- *Circuito OR realizzato con tre NAND*
- *Semisommatore*

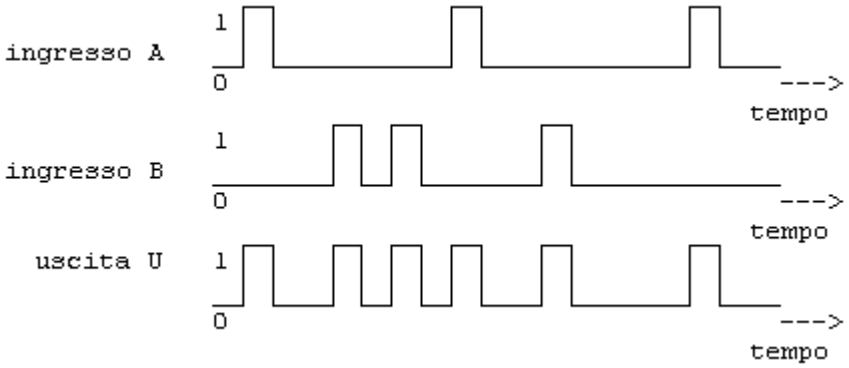
Funzioni e formule:

AND: $U = AB$ *NAND:* $U = \overline{AB}$

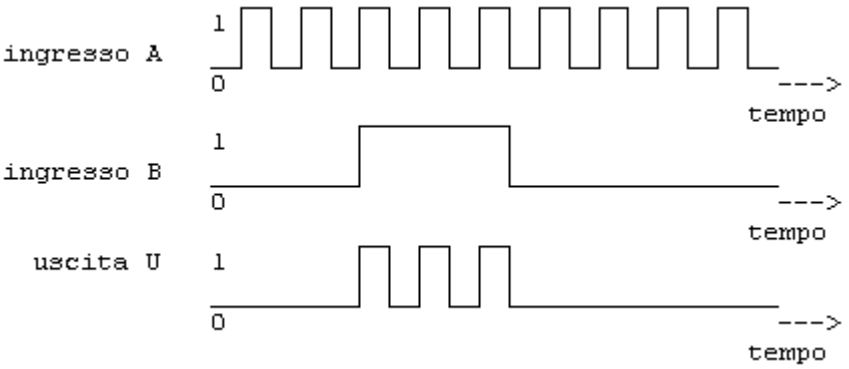
OR: $U = A + B$ *NOR:* $U = \overline{A + B}$

NOT: $U = \overline{A}$ *EX-OR:* $U = A \oplus B$

L'applicazione più importante del circuito OR: combinare mediante la somma logica segnali diversi in modo da ottenere un segnale più complesso, come si può osservare nella figura seguente:



Invece il circuito AND può funzionare come elemento che lascia passare o meno un segnale, come si può notare nel diagramma temporale del segnale presente sui terminali del circuito AND:



Dalla formula logica alla rete logica

- *Costruire la rete logica che esegue la seguente espressione: X = (A + ¬B) ·(A·B) (soluz.)*

Conversione delle tabelle di verità in formule

- *In un circuito combinatorio si considera **Trasmissione parziale di un dato Bit** il fatto che il bit prescelto compaia in uscita quando agli ingressi si presentano le necessarie condizioni.*
  Per esempio, nella tabella sottostante sono segnate con un asterisco le due trasmissioni parziali del bit 1:

A	*B*	*C*	*U*
0	*0*	*0*	*0*
0	*0*	*1*	*1**
0	*1*	*0*	*0*
0	*1*	*1*	*0*
1	*0*	*0*	*0*
1	*0*	*1*	*0*
1	*0*	*1*	*0*
1	*1*	*0*	*1**
1	*1*	*1*	*0*

L'insieme delle trasmissioni parziali del bit prescelto costituisce la trasmissione

complessiva del circuito: { $\overline{A}\,\overline{B}\,C \;\; A\,B\,\overline{C}$

Formula logica U = $\overline{A}\,\overline{B}\,C \cdot A\,B\,\overline{C}$

Criteri adottati per la conversione dei valori logici in simboli letterali:

1. *se la trasmissione è riferita al bit 1, si deve fare in modo che agli ingressi si presentino tutti 1;*
2. *se invece la trasmissione è riferita al bit 0, si deve fare in modo che agli ingressi si presentino tutti 0;*

Nel momento in cui si presentano tutti 1 agli ingressi, si trasmette 1 in uscita soltanto quando il circuito funziona logicamente come AND.

Diversamente, nel momento in cui si presentano tutti 0 agli ingressi, si trasmette 0 in uscita soltanto quando il circuito funziona logicamente come OR

L'espressione simbolica ottenuta è detta FORMA CANONICA. Quella costituita dalla trasmissione complessiva OR e dalle trasmissioni parziali AND è chiamata Forma Canonica a MINTERM; quella costituita dalla trasmissione complessiva AND e dalle trasmissioni parziali OR è chiamata Forma canonica a MINTERM.

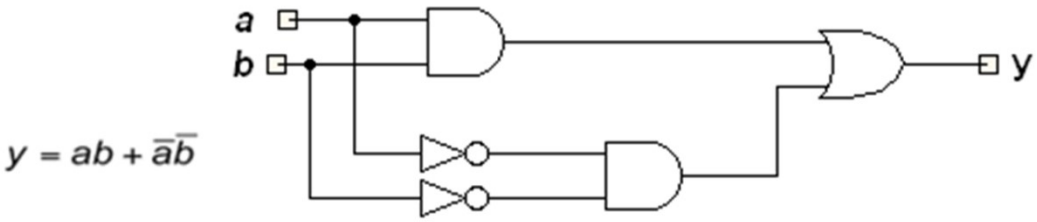
Simulazione di barre antifurto con digital works:

- *barre antifurto*

"Digital Works": ***Quick guide***

"Digital Works" è utile per realizzare circuiti digitali e per analizzare il loro funzionamento.

1. *È uno strumento interattivo per la progettazione e la simulazione di circuiti logici;*
2. *Utile allo studio del comportamento funzionale dei circuiti;*
3. *Consente di comprendere la "corrispondenza" tra i concetti teorici dell'algebra di Boole ed i suoi aspetti circuitali. Nell'immagine che segue si vede un esempio di corrispondenza biunivoca tra rete logica ed espressione algebrica.*



Nella Tool Bar sono disponibili porte (AND, OR, NAND, NOR, XOR, XNOR, NOT), semplici flip-flops(D, RS e JK), tri-state e memorie.

Come si realizza un circuito?

1 passo: con il puntatore del mouse selezionare un oggetto della Tool Bar e inserirlo nello spazio di lavoro (workspace), inserire anche gli altri oggetti necessari per disegnare il circuito;

2 passo: collegare gli oggetti con la penna della Tool Bar (Wiring tool). Con lo strumento penna fare click sul punto di partenza di un oggetto (segnalato con "wire"), premere il pulsante sx del mouse e disegnare il collegamento che deve terminare nel punto di arrivo di un altro oggetto (segnalato con "wire").
Per fare un collegamento con una linea spezzata lasciare il pulsante sx del mouse, fare click e continuare il disegno.

Si possono collegare tra loro due o più collegamenti (appare un puntino nero nell'intersezione delle due linee).

3 passo: premere il tasto Run della Tool Bar per avviare la simulazione, selezionare lo strumento "manina" della Tool Bar per cambiare il valore della "Interactive Input".

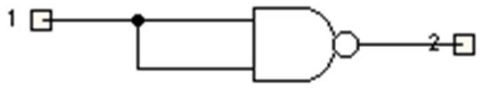
Il circuito realizzato può essere salvato in un file con estensione .dwm. I file .dwn sono macro e le macro possono essere visualizzate in due modalità:

1. *massimizzata (possono essere osservati/modificati i dettagli interni alla black box);*
2. *minimizzata (è visibile solo la forma esterna della black box e i pin per gli input e gli output).*

Si possono creare macro per trasformare un circuito in una black box *ed utilizzarla in circuiti più complessi. Ad esempio si può realizzare il circuito integrato MM74C32 che contiene 4 porte OR.*

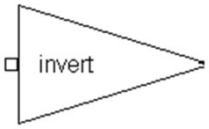
Realizziamo un INVERTITORE mediante l'impiego di un NAND:

1 passo: Disegniamo il seguente circuito e salviamo nella macro denominata Invertmacro



I quadratini di input (1) e di output (2) si ottengono premendo il tasto dx del mouse. Il quadratino (1) sostituisce l'oggetto "interactive input" e il quadratino (2) il led. I quadratini 1 e 2 saranno i pin *del nuovo oggetto che stiamo creando con la macro.*

2 passo: Selezionare "macro tags" nella Tool Bar e poi click con il pulsante dx del mouse sul quadratino (1) e selezionare "Template Editor". Disegnare il nuovo oggetto ad esempio



- *Il triangolo è stato disegnato utilizzando lo strumento "Polygon" della tool bar del Template Editor. Il quadratino è stato inserito scegliendo lo strumento* pin *nella tool bar del Template Editor.*

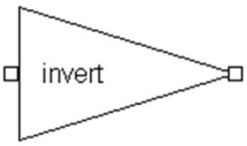
  Poi fare click con il pulsante dx del mouse e scegliere "associate with tag" che associa il quadratino (1) del circuito in modalità " massimizzata" con il pin *in modalità " minimizzata" (la conferma che i due "punti di input" sono stati associati viene segnalata riempiendo di colore giallo il* pin*.*

  Chiudere il Template Editor.

3 passo: Ripetere le operazioni del passo due per inserire il secondo pin *che rappresenta l'output.*

Il Template Editor si riapre presentando il disegno precedente e questa volta rappresenta in rosso il pin *relativo al quadratino (1) disegnato al passo 2.*

Otteniamo il seguente schema completo



4 passo: Aprire una macro vuota e scegliere "Embed Macro" nella Tool Bar per inserire l'INVERTITORE creato(in modalità minimizzata") nello spazio di lavoro ed utilizzarlo in combinazione di altre porte per realizzare circuiti più complessi.

- ***ESERCIZIO***

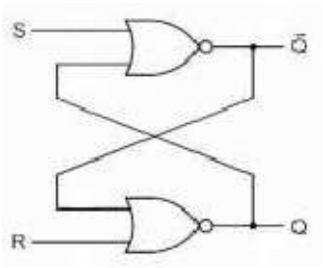
Creare il circuito integrato MM70C32 utilizzando la modalità "minimizzata" collegarlo a 4 led ed a 8 "Interactive input" per analizzarne il funzionamento.

I piedini 7 e 14 non devono essere rappresentati.

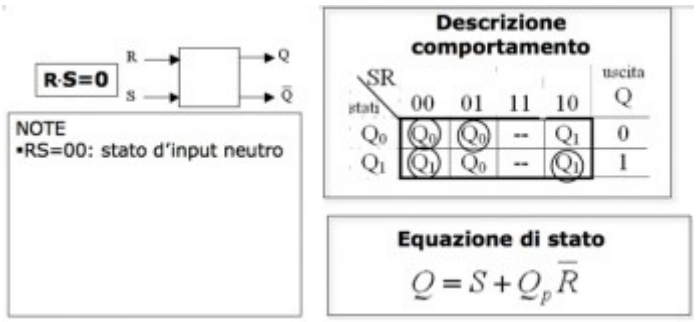
CIRCUITI SEQUENZIALI

Seppure le reti combinatorie permettono di realizzare funzioni aritmetico-logiche, esse non sono in grado di realizzare una delle funzionalità più importanti per un calcolatore elettronico, cioè la possibilità di memorizzare l'informazione e di usare tale memoria per elaborare in modo opportuno quest'ultima. Le reti sequenziali costituiscono lo strumento concreto per la realizzazione di tale possibilità.

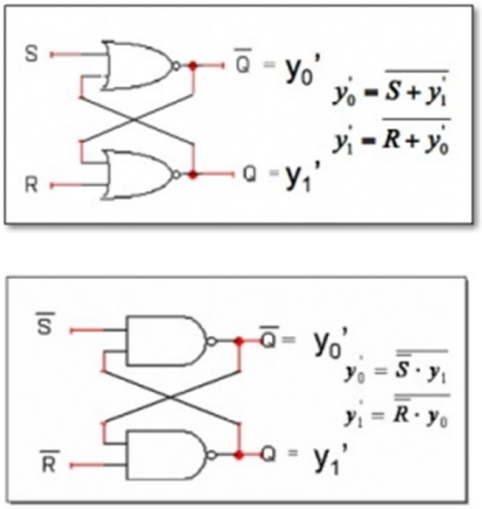
Un primo esempio di circuito sequenziale che permette la memorizzazione di un singolo bit è il Latch SR:



Flip flop RS fondamentale



Realizzazione a NOR o a NAND



Simulazione Digital works:

- *Bistabile RS*
- *Flip flop RS con due circuiti NAND*
- *Flip flop RS con due circuiti NOR*
- *Confronto tra flip flop RS D JK*
- *Memoria*
- *Circuito a scorrimento serie – parallelo*

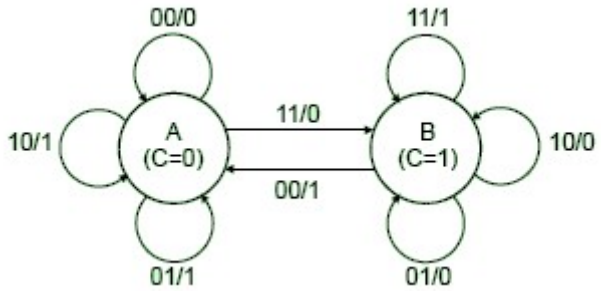
Esercitazioni da eseguire con digital works 3.4

- *Trasmissione dati di tipo serie sincrona*
- *Trasmissione dati di tipo serie sincrona 2*
- *Codice BCD e display a sette segmenti*
- *Memory*
- *Contatore modulo 10*
- *Contare fino a 100*
- *Bus conteso*

Il flip-flop Set/Reset è **la più semplice rete sequenziale possibile**, è anche un automa a stati finiti molto semplice, visto che ha due soli stati interni.

AUTOMI

Disegnamo il diagramma relativo ad un addizionatore a due bit. L'operazione è condizionata da due sole situazioni di rilievo e cioè se c'è stato o no un riporto dallo stadio precedente. I due possibili stati della corrispondente rete sono caratterizzati completamente dalla condizione C=0, C=1. Il corrispondente "***diagramma degli stati***" è riportato in figura.



L'interpretazione del grafo va inquadrata in un opportuno contesto logico.
Bisogna immaginare che una coppia di registri a scorrimento con un numero di celle sufficienti a contenere i due operandi siano stati precaricati, magari in parallelo, con i due numeri da sommare ed ad ogni periodo di clock presentino ordinatamente al sommatore i due bit corrispondenti, partendo da quelli meno significativi.

Automa che effettua l'addizione tra due cifre binarie.

Gli ingressi sono le due cifre binarie da sommare

Lo stato indica il riporto (0 oppure 1)

L'uscita indica il risultato della somma (0 oppure 1)

Tabella degli stati

(Insieme degli stati) ↓

	00	***01***	***10***	***11***	***←(insieme degli ingressi)***
0	*0*	*0*	*0*	*1*	***←(riporto)***
1	*0*	*1*	*1*	*1*	

Tabella delle uscite

	00	***01***	***10***	***11***
0	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*1*

Il caso più semplice è l'addizione delle due ultime cifre di due numeri binari; infatti può dare un riporto verso la colonna di sinistra ma non riceve riporto da destra.
Tabella delle possibili combinazioni di valori degli addendi e i corrispondenti valori della somma (S) e del riporto (R).

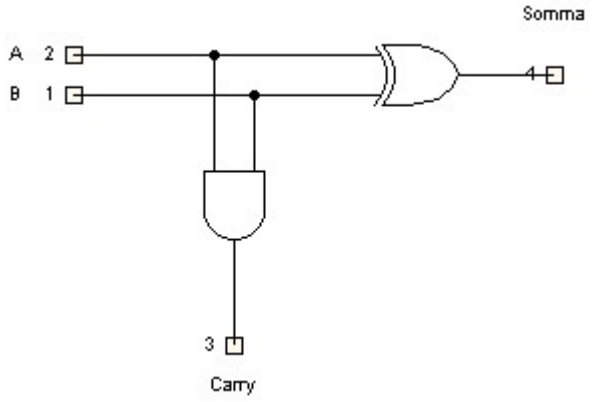
A	*B*	*S*	*R*
0	*0*	*0*	*0*
1	*0*	*1*	*0*
0	*1*	*1*	*0*
1	*1*	*0*	*1*

↗ (XOR) *↗ (AND)*

Dalla tabella si può rilevare che il valore di S si ottiene dalla somma logica esclusiva

di A e B ed il riporto R dal prodotto logico di A e B.

Possiamo realizzare, quindi, la seguente rete logica detta ***Semisommatore***



La somma di due cifre di una colonna intermedia si realizza con una rete logica detta

Sommatore

Tale rete somma tre cifre binarie C D R1 con C e D le cifre della colonna intermedia e R1 il riporto della colonna di destra.

Il sommatore si compone di due semisommatori il primo dei quali fornisce la somma S2 e il riporto R2 di C+D, il secondo, avente come ingressi il riporto R1, proveniente da destra, e la somma parziale S2, restituisce la somma S e il riporto R della colonna intermedia.