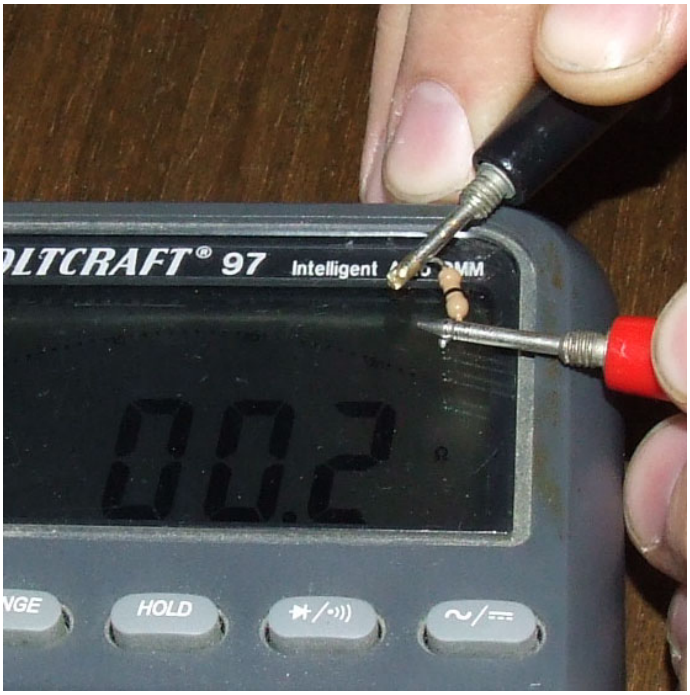
Esercitazioni pratiche di elettronica

Benvenuto nel wikibook:

Esercitazioni pratiche di elettronica

Autore: Rdgmus



Indice

Voci

Esercitazioni pratiche di elettronica

Esercitazioni pratiche di elettronica

Lezioni ed esercitazioni pratiche di elettronica analogica e digitale per il 3° anno dei licei tecnici e professionali

Dedico il presente lavoro di redazione del libro all'insigne filosofo e pedagogo giapponese Tsunesaburo Machiguchi, autore della Teoria del Valore.

Tsunesaburo Machiguchi.

TSUNESABURO MACHIGUCHI [1]

e a Rosa Luxemburg...

Rosa Luxemburg.

Rosa Luxemburg

Libri correlati

- Elettronica pratica

Note

[1] http://www.tmakiguchi.org/

Introduzione

Finalità

Il presente libro multimediale viene realizzato al fine di fornire il materiale didattico che possa essere utilizzato dagli allievi delle classi terminali dell'istituto professionale al fine del conseguimento del diploma di specializzazione quale **OPERATORE ELETTRONICO**, tenendo conto delle teconologie attuali nell'industria, dopo aver riscontrato la totale inadeguatezza dei testi prodotti dall'editoria per tali corsi professionali e del relativo costo che grava sempre più pesantemente sull'economia delle famiglie italiane. Speriamo che possa essere di utilità agli allievi e ci attendiamo contributi positivi a tal fine sotto il coordinamento del professore da parte degli allievi stessi.

Contenuti del programma

- Simulazione & conoscenza dei principali circuiti dell'Elettronica analogica.
- Simulazione & conoscenza dei principali circuiti dell'Elettronica digitale.
- Linguaggio **VHDL** per l'implementazione su schede **SOPC** di circuiti digitali, verifica di laboratorio dei circuiti prodotti
- Progettazione, realizzazione e verifica di semplici sistemi digitali, ciclo della progettazione e verifica di aderenza alle specifiche di sistema.
- La conversione dei segnali dal mondo analogico al mondo digitale, progettazione di semplici convertitori **ADC** e **DAC**.
- Principi di sintesi e elaborazione dei segnali.
- Metodologie di progetto e collaborazione di gruppo.
- Semplici sistemi di controllo.
- Ripasso del programma e approfondimenti specifici alle realizzazioni di gruppo prodotte durante l'anno nel laboratorio.
- Verifica delle conoscenze pratiche e concettuali a preparazione dell'esame di specializzazione.

Lezioni

Simulazione dei principali circuiti con amplificatore operazionale

OBIETTIVI

Facendo uso del software Simplorer procedere alla simulazione dei principali circuiti con amplificatore operazionale, al fine di prendere visione del software e delle sue potenzialità.

- Comparatore a singola soglia

Differenza fra segnali analogici e segnali digitali

- **SEGNALE ANALOGICO**

È un segnale continuo nel tempo che può assumere tutti gli **infiniti valori** della grandezza fisica osservabile, (sia essa una tensione, una corrente, una temperatura o altro...) contenuti all'interno di un determinato **range**, ovvero tra un minimo relativo ed un massimo relativo.

- **SEGNALE DIGITALE**

È un segnale che all'interno di un determinato **range** può assumere solo un **numero discreto (numerabile) di valori**; ad esempio un'onda quadra che assume i valori logici ALTO (HIGH, 1 logico) e BASSO (LOW, 0 logico). È opportuno distinguere tra

- **segnali digitali ideali**

i quali possono solo assumere i valori ALTO oppure BASSO, ed in cui le transizioni tra i due stati logici avvengono in un tempo infinitamente piccolo ovvero nullo, creando dunque all'interno di queste transizioni dei punti di discontinuità tra i due valori limite, (differenza questa importante nei riguardi dei segnali analogici che non presentano discontinuità) e

- **segnali digitali reali**

nei quali le transizioni da ALTO a BASSO oppure da BASSO ad ALTO non avvengono in tempi nulli e che quindi presentano dei **tempi di salita** e **tempi di discesa** dell'onda quadra che non sono nulli ma finiti (avvengono cioè in un tempo finito o meglio ben definito).

Introduzione alle logiche TTL

- Introduzione storica sulla nascita dei circuiti digitali
- Le scale di integrazione dei circuiti digitali
- Le logiche TTL, DTL e loro realizzazione in laboratorio
  - Inverter realizzato con transistor BJT

Vengono visualizzati il segnale ad onda quadra posto in ingresso ed il segnale di uscita prelevato sul collettore del transistor. Si nota lo sfasamento di 180° tra i due segnali.

Classi fondamentali dei circuiti digitali

Vi sono due grandi classi di circuiti digitali che bisogna conoscere:

Circuiti a logica combinatoria

Si tratta di circuiti, che per il momento vediamo come una scatola nera, provvista di INPUTS e OUTPUTS, dove le uscite in ogni momento dipendono dalle variabili poste in ingresso al nostro circuito, nello stesso momento. (Si trascurano i ritardi di propagazione dei segnali che saranno oggetto di successiva discussione)

Logica Combinatoria

Circuiti a logica sequenziale

Questa seconda grande classe di circuiti digitali, comprende tutti quei circuiti nei quali, le uscite dipendono dalle variabili poste in ingresso (uno o più ingressi) nello stesso momento ma anche dalle uscite che il circuito aveva in momenti precedenti. Da questo si evince che tali circuiti sono dotati al loro interno di una **memoria** oppure possiamo dire anche che tali circuiti oltre ad avere ingressi ed uscite, sono dotati al loro interno anche di una altra variabile detta **stato**: una **Macchina a Stati** è un circuito digitale sequenziale dove lo stato successivo nel quale si porrà dipende dallo stato precedente e dalle variabili poste in ingresso.

Logica Sequenziale

I sistemi di numerazione

Discorso introduttivo

"Il saper fare di conto", è una delle operazioni più difficili e che viene invece considerata con sufficienza e superficialità con il risultato che se parli di numerazione in base 2 oppure in base 8 oppure in base 16, inizialmente ti sembra di parlare ad una popolazione incontrata dall'enterprise in una delle sue esplorazioni intergalattiche. Bando agli scherzi! La numerazione avviene accostando un certo numero di cifre numeriche e generalmente siamo abituati a scrivere i numeri in base 10, accostandoli a partire da sinistra con le cifre aventi peso maggiore e muovendosi verso destra con le cifre aventi peso minore. Ad esempio se scrivo 7348 settemilatrecentoquarantotto, sto scrivendo praticamente una quantità pari a 7 migliaia + 3 centinaia + 4 decine + 8 unità. Questo lo abbaimo studiato alle scuole elementari, ma siamo sicuri di averlo capito?

Quanto abbiamo appena detto merita innanzitutto una generalizzazione in modo che la tecnica con la quale siamo abituati a scrivere i numeri in base 10 ci possa essere ugualmente utile per scrivere numeri ovvero saper contare, in sistemi di numerazione che adottano altre basi numeriche.

Chiariamo quindi cosa vuol dire **base**:

Un esempio chiarirà meglio quanto stiamo per dire; affermare che un numero è espresso in base 10 vuol dire che quel numero è stato costruito con un numero di oggetti pari a 10 e questi oggetti sono precisamente le cifre numeriche a partire da zero (0) fino a nove (9).

Se invece un numero è in base 2 allora ho adoperato esclusivamente 2 oggetti per costruirlo che sono le cifre numeriche da zero (0) a uno (1).

In generale possiamo quindi stabilire la regola che:

Un numero si esprime in una certa base se si adopera per la sua costruzione un numero di cifre pari alla base adottata e tali cifre vanno da zero (0) alla (base - 1).

Quindi...

- la base 10 usa le cifre da 0 a 9
- la base 2 le cifre da 0 a 1
- la base 8 le cifre da 0 a 7
- la base 16 le cifre da 0 a 15.

Per le basi maggiori di 10, in particolare noi utilizzeremo spesso e volentieri la base 16, si pone il problema che le cifre numeriche del sistema che utilizziamo noi sono al massimo in numero di 10 (le cifre che vanno da 0 a 9) e quindi sono state introdotto delle lettere che individuino univocamente le cifre numeriche, ovvero gli oggetti, che devono rappresentare il decimo, l'undicesimo, il dodicesimo....fino al quindicesimo oggetto. Ecco perché per la numerazione in base 16, a differenza delle altre basi le cifre numeriche sono:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

La conversione da una base all'altra

In fase progettuale di un hardware o di un software si avrà a che fare spesso e volentieri, con il dover convertire un numero espresso in una certa base, in un'altra base più adatta al sistema che stiamo utilizzando in quel momento.

Ecco perché qui introduco alcuni algoritmi di conversione da una base all'altra che sono molto utili nel nostro bagaglio tecnico.

- Un po' di matematica non guasta...

Prima di iniziare è bene che l'insegnante spieghi la seguente formula che esplode

un numero N in base B nella sommatoria delle cifre che compongono il numero

moltiplicate per la base elevata al peso che la cifra stessa ha nel numero:

$$N_{B_1} = \sum_{P_1=0}^{H-1} C_{P1} * {B_1}^{P_1}$$

e dove **H** è il numero di bit che compongono il numero.

In termini puramente matematici, effettuare una conversione di base equivale a trovare i coefficienti

del numero nella base desiderata a partire dalla seguente equivalenza:

$$N_{B_1} = \sum_{P_1=0}^{h-1} C_{P1} * {B_1}^{P_1} = \sum_{P_2=0}^{k-1} C_{P2} * {B_2}^{P_2} = N_{B_2} = N_{10}$$

dove **h** è il numero di bit che compongono il numero in base B_1

e **k** è il numero di bit che compongono il numero in base B_2

ed entrambe devono dare come risultato lo stesso numero in base 10.

Esempio:

$$76_{10} = 6*10^0+7*10^1 = 1001100_2 = 0*2^0+0*2^1+1*2^2+1*2^3+0*2^4+0*2^5+1*2^6$$

76 IN BASE 10 = 1001100 IN BASE 2.

GLI ALGORITMI DI CONVERSIONE

- Conversione da base 2 in base 10
- Conversione da base 10 in base 2
- Conversione da base 10 in base 16

Conversione da base 2 in base 10

- Sovrappongo a ciascuna cifra componente il numero dato, a partire dalla cifra più a destra, la numerazione in ordine crescente a partire da zero della posizione della cifra stessa.
- Prendo tutte le cifre uguali ad 1 ed effettuo la somma della base 2 elevata all'esponente corrispondente alla posizione della cifra considerata.
- Effettuo il totale: il risultato è il numero convertito in base 10.

Esempio:

Convertire 100011 da base 2 a base 10

- Posizione...5 4 3 2 1 0
- Cifra.....1 0 0 0 1 1
- Somma = 2^5 + 2^1 + 2^0 = 32 + 2 + 1 = 35

Conversione da base 10 in base 2

Dato un numero in base 10 lo voglio convertire in base 2...

La figura mostra l'algoritmo da eseguire per la conversione.

Esempio:

- Convertiamo 35 da base 10 in base 2

N=35

B=10

E=0

Nb=0000

2^E=1

1 è > 35 ? No

Allora E = E+1 = 1

2^E=2

2 è > 35 ? No

Allora E = E+1 = 2

2^E=4

4 è > 35 ? No

Allora E = E+1 = 3

2^E=8

8 è > 35 ? No

Allora E = E+1 = 4

2^E=16

16 è > 35 ? No

Allora E = E+1 = 5

2^E=32

32 è > 35 ? No

Allora E = E+1 = 6

2^E=64

64 è > 35 ? Sì

Allora E = E-1 = 5

Nb=Nb+2^E= 100000

N=N-2^E = 35-32=3

E=0

2^E=1

1 è > 3 ? No

Allora E = E+1 = 1

2^E=2

2 è > 3 ? No

Allora E = E+1 = 2

2^E=4

4 è > 3 ? Sì

Allora E = E-1 = 1

Nb=100000 + 10

N=N-2^E = 3-2=1

E=0

2^E=1

1 è = 1 ? Sì

Nb=100000 + 10 + 1

N=N-2^E = 1-1=0

NB

100000+

000010+

000001=

100011

FINE

- **Più semplicemente.....**

Tenere a portata di mano una tabellina con le potenze in base 2 in ordine crescente a partire dall'esponente zero:

Exp	2^Exp
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
...	...

- Eseguire passo-passo il seguente algoritmo:

Passo	Azione
1	Dato il numero in base 10 N=35
2	Cerco nella tabellina di cui sopra, la potenza **Exp** del **2** tale che **2^Exp <= N**
3	Aggiungi la potenza trovata al risultato => **Risult**
4	N = N - 2^Exp
5	Se N = 0 hai **terminato!!!**
6	Altrimenti torna al punto **2**

- Ad Esempio: Convertiamo 35 da base 10 in base 2...

N = 35

2^5 = 32 < N

Risult = 2^5

N = 35 - 32 = 3

Torno al punto **2**

2^1 = 2 < N

Risult = 2^5 + 2^1

N = 3 - 2 = 1

Torno al punto **2**

2^0 = 1 = 1

Risult = 2^5 + 2^1 + 2^0

N = 1 - 1 = 0

Ho terminato.

Infine trasformo il Risultato espresso in potenze del 2

nel numero binario cercato, sostituendo la cifra 1 nelle posizioni

relative agli esponenti della base 2 che compaiono nel risultato e

pongo a 0 le cifre nelle altre posizioni.

Peso	**5**	**4**	**3**	**2**	**1**	**0**
Numero in base 2	**1**	**0**	**0**	**0**	**1**	**1**

Conversione da base 10 in base 16

Si può effettuare in tre singoli passi:

- Conversione da base 10 in base 2
- Dividere il risultato in base 2 in blocchi di 4 bits a partire da destra; (aggiungere eventualmente degli zeri in coda alle cifre più significative per completare l'ultimo blocco di 4 bits)
- Sostituire ad ogni blocco di 4 bits la cifra esadecimale corrispondente

A_3	A_2	A_1	A_0	**ESA**
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**2**
0	0	1	1	**3**
0	1	0	0	**4**
0	1	0	1	**5**
0	1	1	0	**6**
0	1	1	1	**7**
1	0	0	0	**8**
1	0	0	1	**9**
1	0	1	0	**A**
1	0	1	1	**B**
1	1	0	0	**C**
1	1	0	1	**D**
1	1	1	0	**E**
1	1	1	1	**F**

Il ciclo di progetto

- **Il processo dalla progettazione al rilascio è un processo ciclico**
- **Specifiche di progetto**

Ottenute le specifiche secondo le quali realizzare un progetto, si può scegliere di realizzarlo in ECS (schemi elettrici), VHDL (Linguaggio di programmazione per la definizione di un hardware) oppure FSM (macchine a stati finiti) o comunque utilizzando anche tutti e tre i tipi di files.

- **Correzioni, aggiunte, modifiche**

Man mano che si progetta si procede all'aggiunta di files o alla correzione di quelli già creati apportando le opportune modifiche.

- **Simulazione**

Si può procedere ad una simulazione via software nel momento in cui lo si ritenga più opportuno per osservare l'andamento dei segnali di output in funzione degli inputs.

- **Implementazione e test**

Oppure si può procedere all'implementazione su scheda e relativo test hardware per avere una verifica oggettiva di quanto si è creato.

- **Rilascio**

Si può procedere al rilascio dell'hardware così programmato, quando lo si ritiene necessario oppure a seconda dei tempi di rilascio prestabiliti nelle specifiche di progetto.

- **Nuove specifiche di progetto**

Se dovessero intervenire nuove specifiche riguardanti il progetto, da parte del cliente, non sarà necessario cambiare la scheda, ma semplicemente provvedere alle opportune modifiche e ad una nuova implementazione sulla scheda, dato che l'interfaccia JTAG permette infinite programmazioni della scheda in sede, ovvero senza rimuovere l'hardware o i devices (CPLD, FPGA) ma semplicemente collegandosi ad essa tramite il cavo di collegamento tra PC e interfaccia JTAG della scheda.

Software ISE-XILINX Free Webpack

Tutorial rivolto agli studenti per l'utilizzo del software Xilinx-ISE Free WEBPACK

- Finestra principale
- Creazione di un nuovo progetto
- Creazione di uno schema
- Creazione di un file VHDL
- Creazione di una Macchina a Stati Finiti
- Il file di piedinatura
- La simulazione
- L'implementazione su scheda e il test

Finestra principale

Innanzitutto è importante distinguere all'interno della videata principale del software le finestre che la compongono ed il loro uso e significato.

- **Sources in Project** - files sorgenti del progetto

In questa finestra in alto a sinistra sono elencati il **titolo del progetto** e subito sotto di esso il chip programmabile per il quale vengono creati i files di progetto. Sotto il chip si vede l'elenco dei files che compongono il progetto.

- **Processes for Source** - processi per il file sorgente selezionato

Selezionando uno dei file di progetto di cui si è detto al punto precedente, in questa finestra appaiono i processi, ovvero le operazioni, che si possono eseguire su quel determinato file (ad esempio la compilazione di uno schema in formato VHDL).

- **Console** - log delle operazioni svolte

Questa ultima finestra contiene, in formato testo, il resoconto delle ultime operazioni eseguite nella finestra precedente e l'esito delle stesse operazioni. La consultazione di questa finestra eè di notevole importanza per poter apportare le eventuali correzioni al circuito prima dell'implementazione su scheda SOPC.

Creazione di un nuovo progetto

Dal menù **File** selezionare **New Project**

Inserire il nome del progetto in **Project Name**;

evitare spazi nei nomi!

tale nome verrà automaticamente aggiunto a **Project Location**

si consiglia di non cambiare questa locazione.

Premere **Avanti >**

Inserire il **Device Family**, **Device** e **Package** riferendosi alla scheda che si utilizzerà per l'hardware.

Premere **Avanti >**

Per il momento non creiamo nessun nuovo sorgente.

Premere **Avanti >**

Per il momento non aggiungiamo nessun sorgente già esistente.

Premere **Avanti >**

Questa finestra riepiloga quanto inserito nei passi precedenti.

Premere **Fine**

Riappare la Finestra principale dove nella sotto-finestra **Sources in project** si scorgono il nome del progetto e il device che sarà utilizzato al momento dell'implementazione sull'hardware.

Creazione di uno schema

Digitare il nome che si vuole dare allo schema in **File Name** e selezionare **Schematic** nella finestra di sinistra. Premere **Avanti>**. Appare la finestra di riepilogo.

Premere **Fine** se tutto va bene altrimenti **<Indietro** e correggi.

Creazione di un file VHDL

- Selezionare il progetto e premere il tasto destro del mouse.
- Scegliere: **Create New Source**
- Selezionare **Vhdl Module**, inserire il nome del file e premere **Next**.
- Inserire le informazioni riguardanti l'interfaccia del modulo Vhdl, ovvero le porte di ingresso/uscita. Premere **Next** quando si è terminato. (*Questo passaggio si può anche saltare, nel qual caso l'interfaccia dell'Entity in Vhdl dovra essere specificata a mano nel file che verrà creato.*)
- Appare la finestra di riepilogo: controllare se le informazionidi riepilogo, corrispondono a quanto si desiderava, e premere **Finish**. (*Eventualmente tornare indietro premendo **Back** e correggere*)

Creazione di una Macchina a Stati Finiti

Procedura

- **Create New Source**

Creare un nuovo sorgente di tipo: **State Diagram**. Digitare il nome del file nell'apposito campo, facendo attenzione a non superare gli 8 caratteri per il nome. L'estensione verrà aggiunta ed è ".dia". Premere **Next**.

- **Riepilogo**

Nel riepilogo potete leggere le specifiche del file che verrà creato ede eventualmente premere **Back** per correggere. Premere **Finish**.

- **Finestra StateCad**

Viene lanciata l'applicazione **StateCad** nella quale potete inserire la macchina a stati di **Mealy** oppure di **Moore** con un diagramma del tipo volgarmente detto **pallogramma**. Alla fine il diagramma compilato creerà un file con lo stesso nome ed estensione **vhd**, un file Vhdl che potete caricare nel Vs. progetto, crearne uno schematic symbol ed utilizzarlo come fareste per qualsiasi altro componente.

- **Tutorial**

Nello Statecad c'è un ottimo Tutorial che eventualmente studieremo insieme.

Il file di piedinatura

- Creare un nuovo sorgente...

Inserire il nome de file della piedinatura in **File Name**. È buona norma chiamare il file **icf_nome_del_sorgente** al quale si riferisce, in modo da non far confusione quando si ha a che fare con molti sorgenti, come succede normalmente quando si progetta. Nella finestra di sinistra selezionate **Implementation Constraints File** e premere **Avanti>**.

In questa nuova finestra che compare selezionare il sorgente di cui al punto precedente. (il software non è così intelligente da sapere a priori che il file a cui si riferisce è quello indicato nel nome se non lo selezioniamo espressamente). Premere **Avanti>.**

La solita finestra di riepilogo di quanto inserito e ormai dovreste aver capito cosa fare. Se tutto è a posto premete **Fine**.

- **Esempio di piedinatura nel manuale della scheda**

Sono due colonne nelle quali appare il numero del piedino del CPLD *a sinistra* e la funzione oppure il device di input/output ad esso collegato nella scheda che state utilizzando *a destra*. Dobbiamo sempre tenere a portata di mano questa tabella che troviamo nella documentazione della scheda **Digilent** per poter procedere alla successiva operazione che consiste proprio nello specificare i piedini del device che vogliamo costringere a collegarsi ai nostri interruttori, bottoni o quant'altro a disposizione.

- Modifica del file di piedinatura

Secondo la nostra scelta dei device (switch, bottoni, led, display) di I/O dobbiamo inserire nel file di piedinatura questi **constraints**, ovvero obbligare il CPLD a collegare gli inputs e ouptputs dello schema creato dove vogliamo noi in modo da poter poi effettuare un test hardware sul funzionamento atteso. Vi sono due processi che ritengo utili in questo stadio della progettazione e sono:

- **Edit Constraints (Text)**

Una volta lanciato il suddetto processo si apre una finestra in formato testo ed è possibile inserire la piedinatura creando per ogni **constraint** una riga ma rispettando la sintassi come dal l'immagine di esempio. (Non dimenticare i punti e virgola alla fine di ogni riga)

- **Assign Package Pins**

Oppure si può lanciare il processo della figura e inserire i pin nelle tendine della colonna **Loc** come da quest'altro file di esempio. Verrà così creato automaticamente lo stesso file che se aperto lanciando il processo **Edit Constraints (Text)** mostrerà le scelte fatte in formato testo.

- **Conclusioni**

Ritengo che la procedura migliore sia proprio di creare il file di piedinatura adoperando il processo **Assign Package Pins** che semplifica questa operazione soprattutto quando si ha a che fare con numerosi I/O perché è il software a controllare che non si inserisca due volte lo stesso piedino (errore comune) e solo dopo lanciare **Edit Constraints (Text)** per effettuare le modifiche minori necessarie al nostro caso.

- Se si verifica la situazione per cui il file debba essere scritto manualmente

aggiungo in calce un esempio generico per evidenziare la SINTASSI da rispettare se si vuole che il file venga riconosciuto correttamente:

```
NET "XLXN_1"  LOC = "p2"  ;
NET "XLXN_2"  LOC = "p43"  ;
```

Ogni riga di piedinatura ha la sintassi:

NET "[nome dell'I/O ovvero il nome del filo]**" LOC = "**[piedino dell'FPGA]**" ;**

(attenzione a non dimenticare il punto e virgola che è il carattere di fine-riga)

La simulazione

Per poter effettuare la simulazione del nostro progetto prima o durante le fasi di implementazione e test bisogna aver installato il software **ModelSim XE III 6.0a Xilinx Edition** che potete scaricare gratuitamente dal sito della Xilinx come avete fatto per il software ISE Free WEBPACK, e poi richiedere un file di licenza perpetuo per la versione **Starter Kit** che ModelSim fornisce gratuitamente.

- Creazione di un nuovo sorgente di tipo **Test Bench Waveform**

La procedura è identica a quanto già detto in Il file di piedinatura. L'unica differenza è chimare il file **tbw_nome_file_sorgente** per le stesse ragioni addotte sempre in Il file di piedinatura, e selezionare **Test Bench Waveform** nella finestra di sinistra.

Si ottiene l'apertura di un file sul modello indicato nell'immagine nel quale potete forzare gli input (in verde) al valore booleano che volete prima di lanciare il processo di simulazione in modo da osservare il comportamento degli outputs (in giallo). In questo modo **ModelSim** opera in modo nascosto e la simulazione mostra i valori attesi in uscita ma la simulazione come in questo caso che viene mostrato potrebbe non essere completamente corretta (forse un bug del software).

Se invece si desidera una simulazione con tutti i crismi, nella quale i segnali si possano osservare con la finezza temporale che giustamente è necessaria in un progetto di una certa complessità, allora si deve lanciare **ModelSim** dall'interno dell'**ISE** ottenendo i risultati che appaiono in quest'altra figura. Decisamente più corretti.

- **Post Scriptum**

A proposito di bug del software va sottolineato che se usate il software ISE Free WEBPACK e riscontrate qualche bug, potete mettervi in contatto sul sito della Xilinx con i programmatori e aiutarli a correggere l'errore che avete notato. Riceverete sicuramente un supporto professionale e una disponibiità senza alcun problema da parte della Xilinx che ha tutto l'interesse a correggere i bugs che naturalmente si inseriscono in qualsisasi software. Potreste ricevere indicazione di scaricare un certo **Service Pack** dove il problema è già stato risolto o potreste essere voi i primi ad averlo notato e quindi intraprendere un proficuo dialogo con la Xilinx che ritengo didatticamente e professionalmente molto utili per qualsiasi studente di istituto professionale.

L'implementazione su scheda e il test

Selezionare il sorgente che si vuole implementare nella scheda dalla finestra **Sources in Project** e nella finestra **Processes for Source** lanciare l'applicazione **Configure Device(iMPACT)**.

È fondamentale che la scheda sulla quale si intende operare l'implementazione dello schema sia collegata al computer tramite il cavo seriale o parallelo, prima di iniziare questa operazione, altrimenti i device presenti nella catena JTAG non saranno riconosciuti.

Verranno eseguiti diversi processi, sui quali entreremo in dettaglio successivamente, ed appariranno nella finestra **Processes for Source** dei **segni di spunta**:

- verdi se il processo relativo è andato a buon fine
- rossi se il software ha ricontrato degli errori
- punti esclamativi gialli nel caso si siano verificati degli **warnings**

Trascurare gli **warnings** perché relativi a problemi di ordine minore che non inficiano il test del progetto sulla scheda (questi andranno corretti solo nella fase conclusiva di rilascio dell'applicazione). I segni di spunta rossi invece vanno corretti altrimenti il progetto non viene fisicamente implementato, esaminando nella finestra **Console** il log degli errori (ci vuole qui una certa conoscenza dell'inglese tecnico ed è quindi un buon esercizio per l'allievo al fine di comprendere dove agire nei sorgenti per eliminare l'errore; gli errori seguenti sono di solito un parto del primo errore che in cascata genera tutta un'altra serie di errori).

Di solito risolvendo il primo errore, se si è fortunati, farà scomparire tutti i segni di spunta rossi e lancerà l'applicazione **Configure Device(iMPACT)**

- Scan della catena **JTAG**

La finestra va accettata così come appare; verrà di conseguenza effettuata una ricerca dei device presenti nella catena JTAG. Premere **Fine**.

È fondamentale che la scheda sia già stata collegata al computer sul quale gira il software ISE, affinché i device possano essere correttamente riconosciuti.

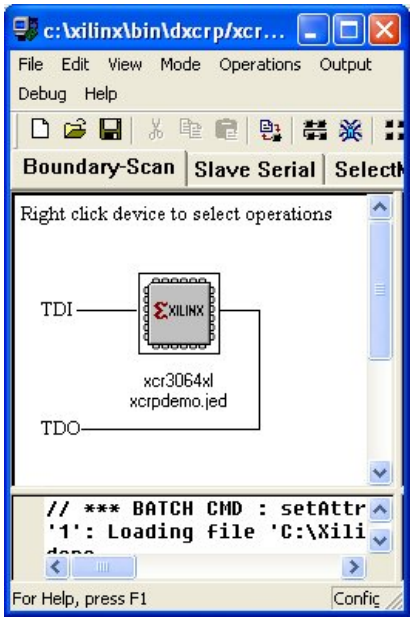
- La catena **JTAG** mostra i device presenti

Appare la finestra che mostra i device presenti nella catena JTAG; premendo **OK** il software si occuperà di aprire una finestra, la successiva, dalla quale caricare il file o i files da associare a ciascun device presente nella catena.

- Caricamento del **file.jed** da implementare

In questa finestra selezionare il **file.jed** da associare al device e premere **Apri**.

- Programmazione di un device della catena



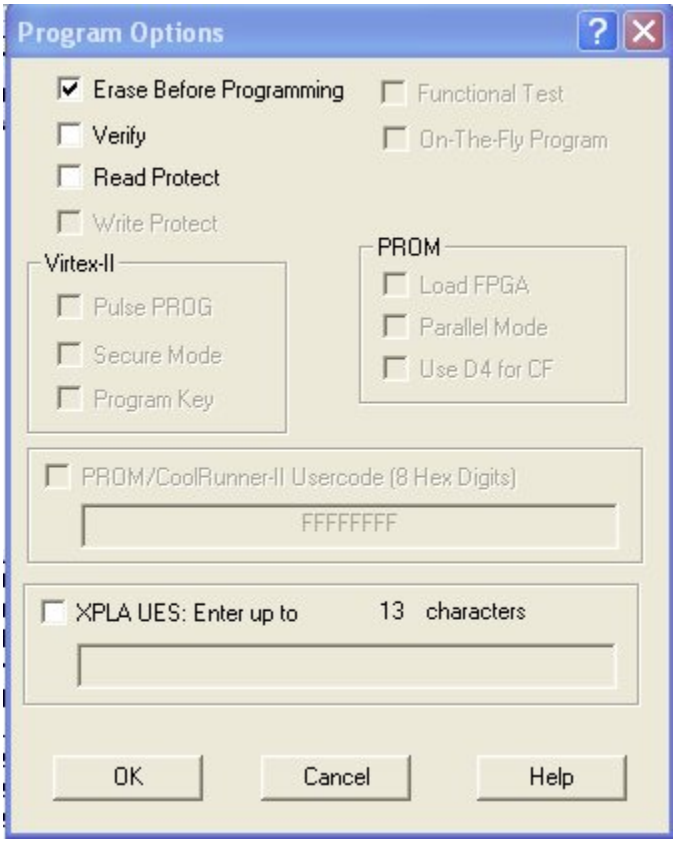
Sotto il device selezionato apparirà il nome del **file.jed** associato ad esso. Ora il device va propriamente programmato tramite le seguenti azioni:

- Selezionare con il mouse il device
- Il device cambia colore mostrando di essere stato selezionato.
- Restando con il mouse sul device selezionato, premere il **tasto destro** del mouse

e selezionare l'operazione **Erase**: tramite questa operazione verrà cancellato dal device qualunque programma che era stato precedentemente implementato al suo interno. Ci attendiamo la risposta **Erase Succeded** ad indicare che l'operazione è andata a buon fine, altrimenti ripetere questo punto.

- Sempre con il mouse sul device selezionato, premere il **tasto destro** del mouse

e selezionare l'operazione **Program**: tramite questa operazione il device verrà programmato con il nuovo programma. NB:L'opzione **Erase Before Programming** è bene che sia selezionata sempre affinché il device non venga danneggiato.



Ci attendiamo la risposta **Programming Succeded** ad indicare che l'operazione è andata a buon fine, altrimenti ripetere questo punto.

Esercitazioni di laboratorio

Logica Combinatoria

Teoria

Al fine di rendere utili le esercitazioni proposte vanno chiariti alcuni concetti di base:

- **Combinazioni di N variabili**

Le combinazioni di **N** variabili in ingresso ad una logica combinatoria sono 2^N**.** La tabella seguente potrà essere utile se ci si rivolge per la prima volta all'implementazione di una logica combinatoria:

N	2^N
1	2
2	4
3	8
4	16
5	32
6	64

Come si vede la crescita delle combinazioni è di tipo esponenziale, il che presenta notevoli difficoltà per valori di **N** già superiori ad $N = 4$; tale difficoltà, vedremo, sarà facilmente sormontabile quando tratteremo delle **logiche sequenziali** tramite l'utilizzo massivo dei dispositivi denominati **Contatori Binari**.

- **Quante variabili di selezione sono necessarie a fronte di N combinazioni?**

La risposta è speculare alla precedente: se si deve selezionare univocamente una ed una sola delle **N combinazioni**, avremo bisogno almeno di **M variabili** di selezione, dove $M <= log_2 N$. Questo concetto risulterà di utilità pratica quando si tratteranno i **Multiplexer**.

- **Logaritmo in base 2:** $M <= log_2 N$

È l'esponente che si deve dare alla base 2 affinché $2^N >= M$**.** A tal fine sarà utile una tabella di esempio:

N Combinazioni	M Variabili	$M <= 2^N$
2	1	$2 = 2^1$
3	2	$3 < 2^2$
4	2	$4 = 2^2$
5	3	$5 < 2^3$
6	3	$6 < 2^3$
7	3	$7 < 2^3$
8	3	$8 = 2^3$
9	4	$9 < 2^4$
10	4	$10 < 2^4$

Esercitazioni di Logica Combinatoria

- Implementazione di una funzione
- Dalla tabella della verità all'implementazione
- Il multiplexer
- Implementazione di una funzione tramite multiplexer
- Il Semisommatore o Half-Adder
- Il Sommatore completo o Full-Adder
- Sommare due parole di 4 bit
- Sommatore a 4 bit di tipo LOOK AHEAD CARRY
- Il Comparatore
- Il decodificatore da BCD a sette segmenti
- Il decodificatore da BCD a 7 segmenti in VHDL
- Multiplexing di più display a 7 segmenti

Implementazione di una funzione

- **La funzione booleana**

Sia data la funzione:

$$y = \bar{A}D + \bar{A}B + \bar{A}\bar{B}\bar{D} + BCD + A\bar{B}CD$$

Ci troviamo di fronte ad una funzione con una singola variabile di uscita **y** e quattro variabili d'ingresso **A, B, C, D**.

- **La tabella di verità**

Costruiamone la **tabella di verità** ovvero rispondiamo alla domanda: a fronte di tutte le combinazioni possibili e univoche di quattro variabili booleane quale sarà la risposta del nostro sistema di logica combinatoria?

Si procede ad esprimere l'uscita **y** analizzando la funzione booleana data, per ogni combinazione delle quattro variabili di ingresso; ovvero se per una riga della tabella esiste un termine dell'equazione per cui si ottiene un **1** (uno logico) come uscita, allora poniamo ad **1** (uno logico) l'uscita **y** della funzione.

Se non abbiamo commesso errori di calcolo la tabella dovrebbe essere la seguente, ma ad ogni modo ci riserveremo di confermare i risultati ottenuti analiticamente tramite il test hardware dopo aver provveduto all'implementazione della funzione stessa sulla nostra scheda:

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0

1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

- **Lo schema realizzato con l'ISE**

Lo schema non sarà altro che la somma logica (OR) dei termini (AND) dati nella funzione booleana che qui ripetiamo per sottolineare quanto andiamo a fare:

$$y = \bar{A}D + \bar{A}B + \bar{A}\bar{B}\bar{D} + BCD + A\bar{B}CD$$

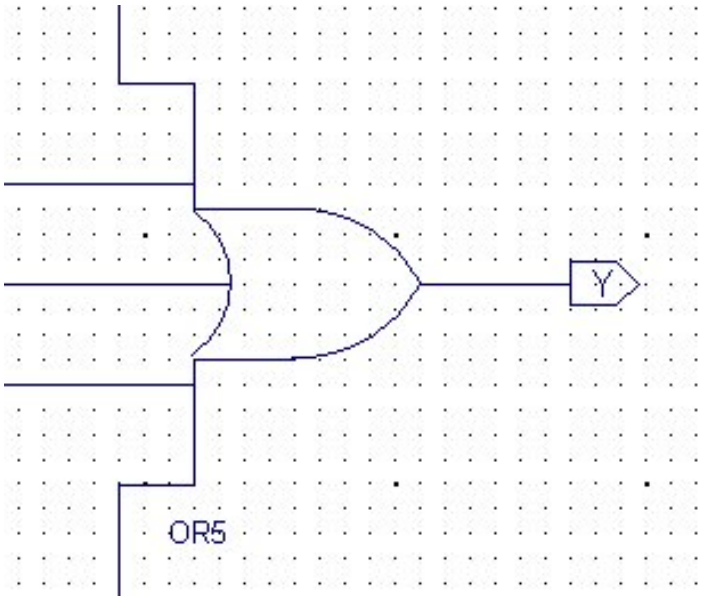
L'ISE ci mette a disposizione tutte le porte (o almeno molte e comunque quelle più necessarie) che servono per esprimere ciascuno dei nostri termini (**minterm** è il nome tecnico di ciascun termine).

- **Gli ingressi**

Notare gli I/O Markers: i marcatori di input/output determinano le porte di ingresso e uscita dello schema.

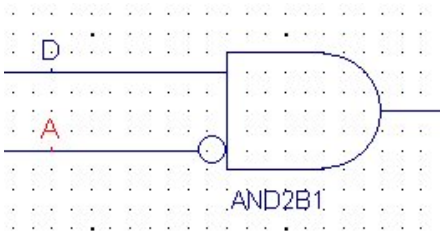
- **La somma logica**

La porta **OR5** è una porta OR a 5 ingressi, notare l'I/O marker di uscita.



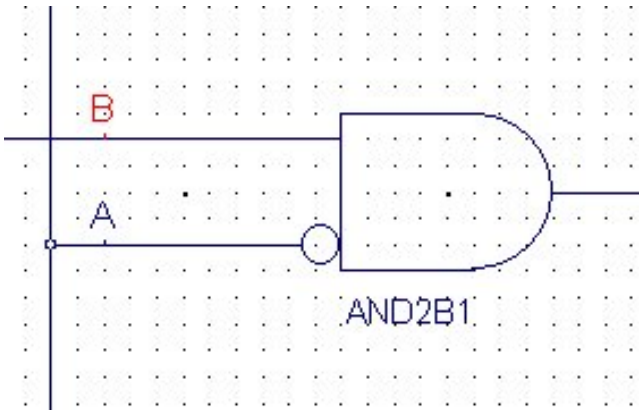
- **I minterms**
  - $\bar{A}D$

La porta **AND2B1** è una porta AND a 2 ingressi (AND2) di cui uno è negato (B1).



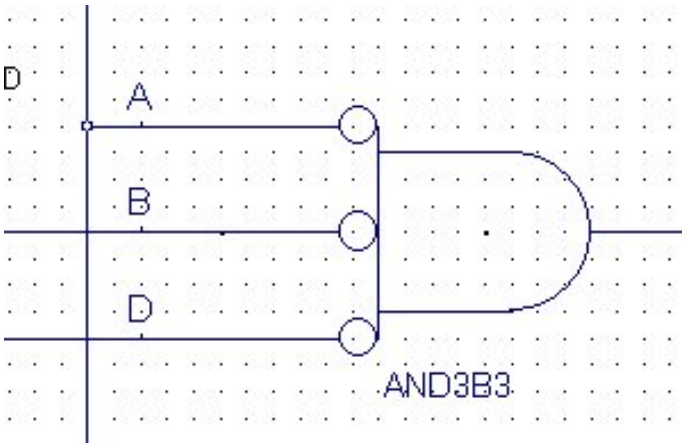
  - $\bar{A}B$

La porta **AND2B1** è una porta AND a 2 ingressi (AND2) di cui uno è negato (B1).



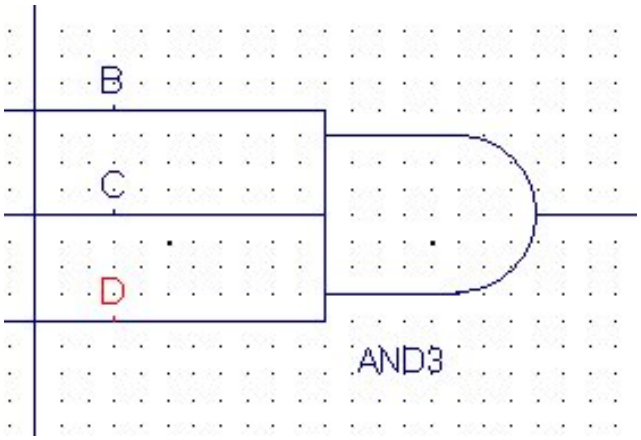
- $\bar{A}\bar{B}\bar{D}$

La porta **AND3B3** è una porta AND a 3 ingressi (AND3) di cui tre negati (B3).



- BCD

La porta **AND3** è una porta AND a 3 ingressi (AND3) di cui nessuno è negato.



- $A\bar{B}CD$

La porta **AND4B1** è una porta AND a 4 ingressi (AND4) di cui uno è negato (B1).

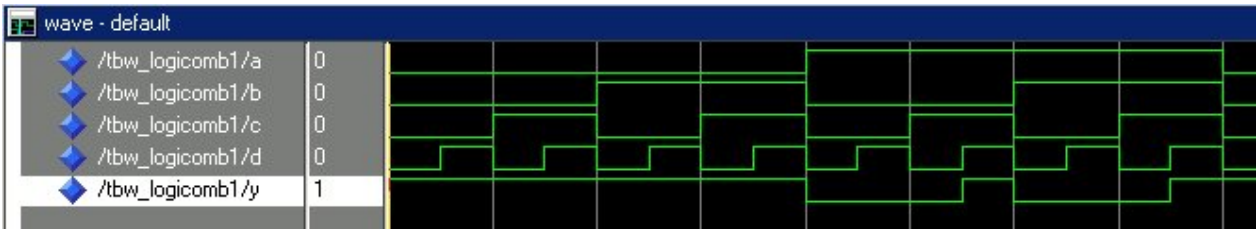
- **La piedinatura**

Si veda Il file di piedinatura dal quale ci si potrà convincere che le porte d'ingresso A,B,C,D sono state indirizzate nella scheda **Digilent XCRP** ai switch SW1,SW2,SW3,SW4 mentre l'uscita Y viene inviata al LED1.

```
3   #PACE: Start of PACE I/O Pin Assignments
4   NET "A"   LOC = "p5"  ;
5   NET "B"   LOC = "p6"  ;
6   NET "C"   LOC = "p8"  ;
7   NET "D"   LOC = "p9"  ;
8   NET "Y"   LOC = "p41"  ;
```

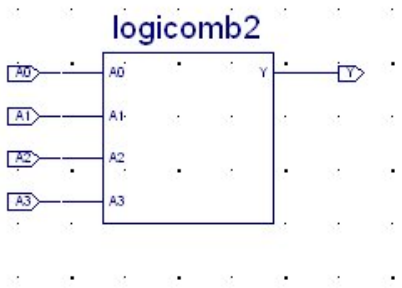
Ora si può passare all'implementazione su scheda, vedasi L'implementazione su scheda e il test

- **La simulazione**



Dalla tabella della verità all'implementazione

È apparentemente un blocco con 4 ingressi, $A_3A_2A_1A_0$ed una sola uscita **Y**.



- **La tabella di verità**

Data la seguente tabella di verità, che rappresenta la risposta desiderata dal nostro blocco di logica combinatoria, si costruisca ed implementi la funzione booleana che la realizza facendo a meno, almeno per il momento di semplificazioni della stessa funzione, le quali necessitano di altre conoscenze che si tratteranno successivamente:

A_3	A_2	A_1	A_0	Y
0	0	0	0	**1**
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0'
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	**1**

1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	**1**

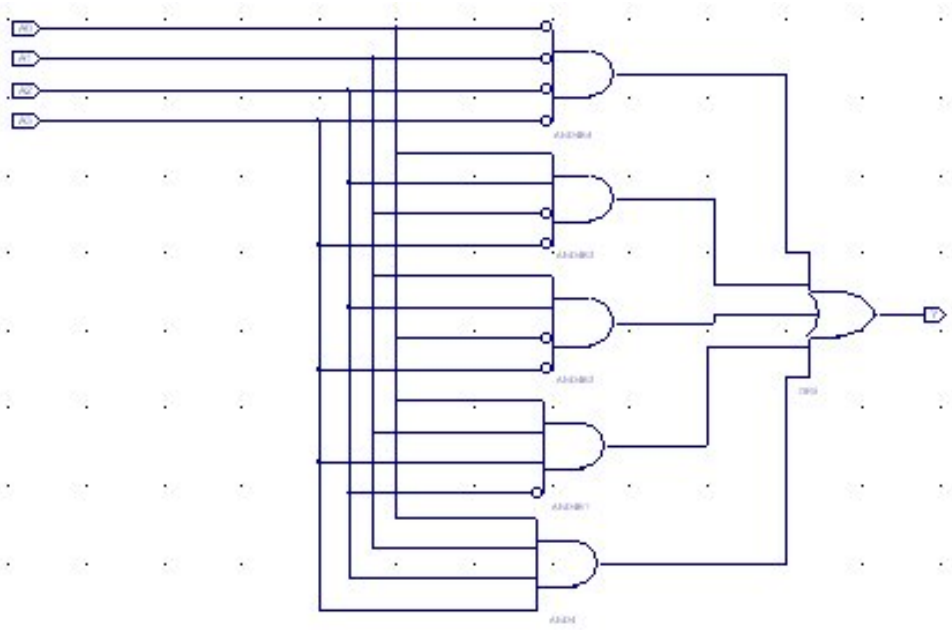
- **Soluzione:**
  - **Si costruisce la funzione booleana**

Per ogni riga della tabella della verità la cui uscita **Y** è uguale ad **1** (uno logico), si costruisce un minterm tramite una porta AND a 4 ingressi, negando gli ingressi delle variabili che nella tabella sono a **0** (zero logico).

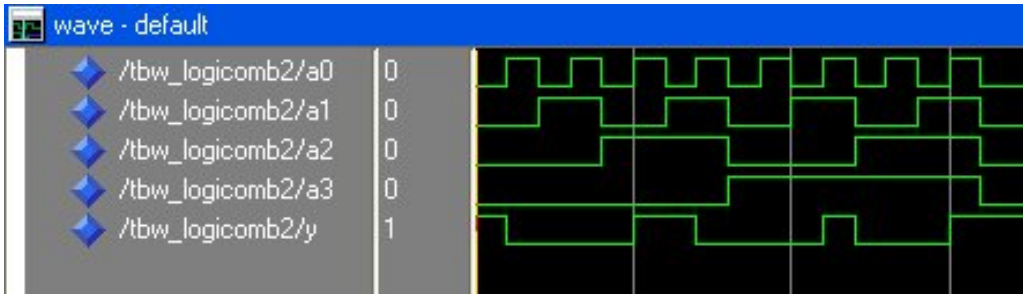
$$y = \bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0 + \bar{A}_3A_2\bar{A}_1A_0 + \bar{A}_3A_2A_1\bar{A}_0 + A_3\bar{A}_2A_1A_0 + A_3A_2A_1A_0$$

Come si vede la funzione booleana può essere molto complicata se nella tabella figurano molte uscite ad 1 ($Y = 1$), ma quello che per ora a noi interessa è che funzioni all'atto dell'implementazione su scheda, e lo vedremo nei prossimi punti.

- **Si implementa e testa con l'ISE**



- **La simulazione**



Il multiplexer

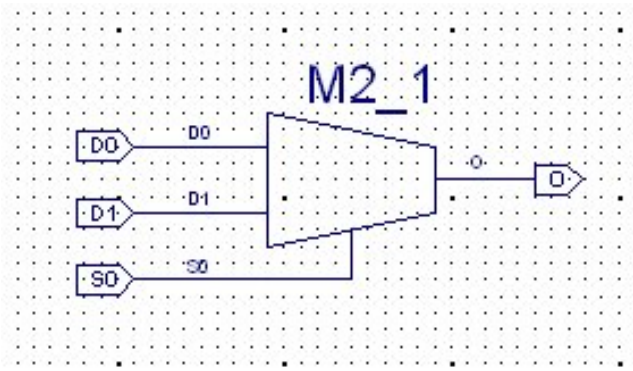
- **Il dispositivo**

Il multiplexer è un dispositivo di logica combinatoria che presenta una interessante innovazione dal punto di vista concettuale, ovvero, oltre agli **N** ingressi di segnale e all'uscita è dotato di uno o più ingressi di selezione, dove il numero **S** degli ingressi di selezione è pari a $S = log_2 N$.

- **Il funzionamento**

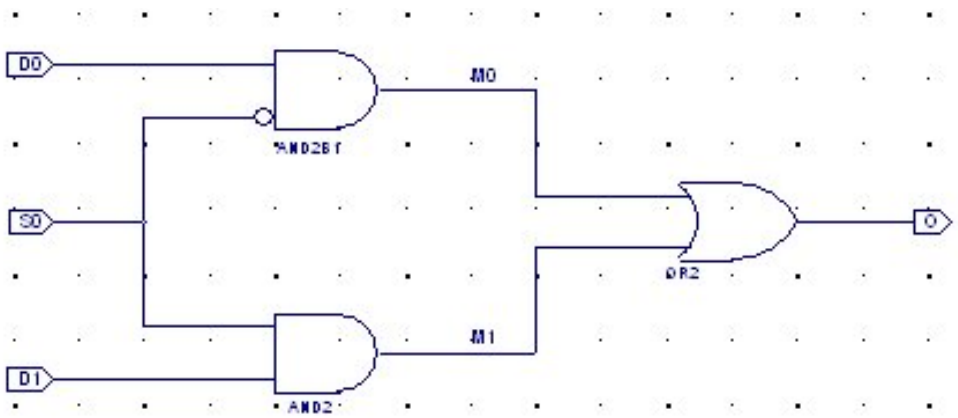
Gli ingressi di selezione determinano in base alla loro configurazione binaria, quale dei segnali posti in ingresso, viene inviato all'uscita. Si comporta quindi come un selettore di segnale o commutatore di segnale.

- **Multiplexer a 2 ingressi**



Avendo due ingressi D_1e D_0, è necessario un solo ingresso di selezione S_0, un solo bit è sufficiente per determinare quale degli ingressi viene inviato all'uscita O; se $S_0 = 0 => O = D_0$, se invece $S_0 = 1 => O = D_1$

- **Schema interno**



- **Tabella della verità**

S_0	D_1	D_0	O
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

In Dalla tabella della verità all'implementazione abbiamo descritto come ottenere la funzione booleana che realizza una tabella di verità del blocco di logica combinatoria. Facciamo la stessa cosa per la tabella qui presentata ed otteniamo la seguente funzione booleana con tre ingressi D_1, D_0, S_0ed un'uscita O :

$O = \bar{S}_0\bar{D}_1D_0 + \bar{S}_0D_1D_0 + S_0D_1\bar{D}_0 + S_0D_1D_0$Sembrerebbe più complicata dello **schema interno** ma possiamo apportare delle semplificazioni alla funzione notando che alcuni termini possono essere messi a **fattor-comune**, ottenendo con un primo passaggio algebrico:

$O = (\bar{S}_0D_0) * (\bar{D}_1 + D_1) + (S_0D_1) * (\bar{D}_0 + D_0)$

E poi osservando che $(\bar{D}_1 + D_1) = 1$e che $(\bar{D}_0 + D_0) = 1$per ogni valore di $D_1 e D_0$(provare per credere!), otteniamo:

$O = (\bar{S}_0D_0) * 1 + (S_0D_1) * 1$

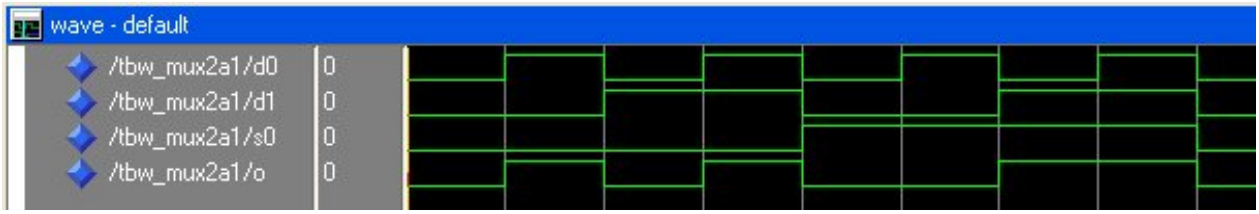
ovvero:

$O = \bar{S}_0D_0 + S_0D_1$

che è proprio la funzione implementata nello **schema interno**.

- **La simulazione**

Evidenzia effettivamente il funzionamento del multiplexer affermato inizialmente, ovvero, se $S_0 = 0 => O = D_0$, se invece $S_0 = 1 => O = D_1$.



Implementazione di una funzione tramite multiplexer

Se dovessimo implementare una funzione booleana espressa tramite la seguente tabella della verità, ci troveremmo di fronte ad un notevole lavoro nel costruire lo schema relativo ad essa, in quanto, come precedentemente affermato, l'implementazione richiederebbe una porta AND a 4 ingressi (di cui negati quelli con valore 0 logico), per ogni riga della tabella che presentasse un'uscita $Y = 1$, e nel nostro caso si dovrebbero utilizzare ben 10 porte AND a 4 ingressi ad una porta OR a 10 ingressi essendo 10 le righe con uscita 1...

S_3	S_2	S_1	S_0	Y
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	**1**
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	**1**

Siete d'accordo?

Beh! è vero che si potrebbero adottare delle semplificazioni, applicare alcuni teoremi (che al momento non conosciamo), ma ad ogni modo il tutto richiederebbe un bel lavoro analitico ed una discreta conoscenza dell'algebra booleana, senza poi tenere conto degli eventuali, e sempre possibili, errori che si potrebbero introdurre durante i passaggi analitici.

Ma esiste un metodo molto più semplice per implementare la funzione data che è facilmente applicabile utilizzando un **multiplexer**, ed è quello che faremo in questo caso per presentare allo studente la **potenza dei multiplexer** ed una delle possibili utilizzazioni che si possono fare di questo interessante dispositivo.

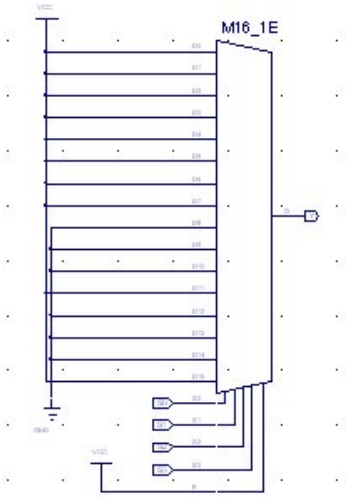
Non ci dimentichiamo che si sta trattando sempre e solo di logica combinatoria, ed uno de nostri scopi è sfruttare al massimo questo tipo di tecnologia in modo da poter scegliere la soluzione migliore e più veloce per risolvere un problema che ci si potrebbe presentare nello sviluppo di sistemi più complicati, come faremo nelle lezioni successive.

I passi da seguire sono i seguenti:

- Ci vuole un multiplexer con **M** ingressi pari al numero delle righe della tabella di verità.
- Il nostro multiplexer avrà **S** ingressi di selezione pari a $S = log_2 M$

- Si collegano i nostri ingressi, in questo caso S_3, S_2, S_1, S_0agli ingressi di selezione del multiplexer.
- Si forzano ad 1 (VCC) o a 0 (GND) gli ingressi del multiplexer secondo le uscite della tabella della verità data.

Schema

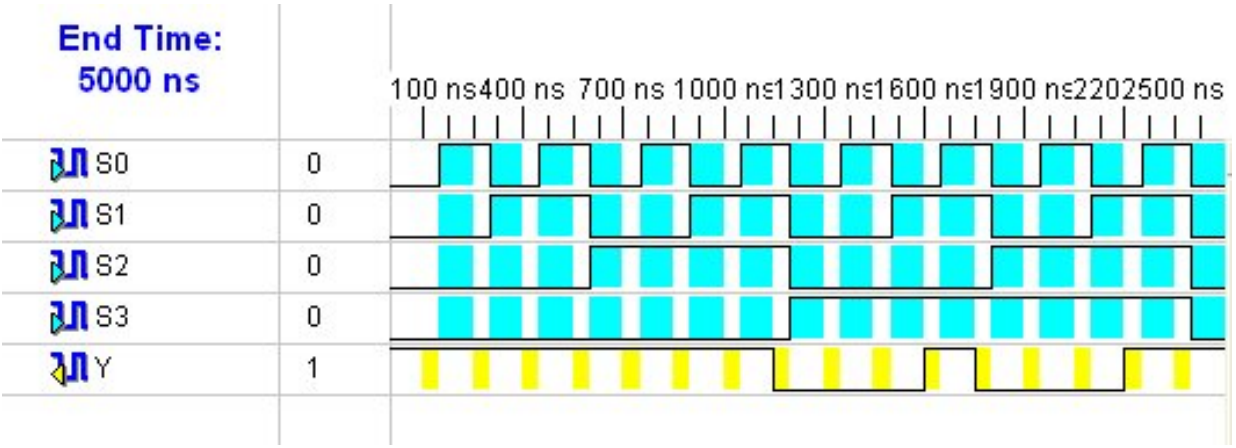


Come funziona?

Secondo la combinazione delle variabili di ingresso S_3, S_2, S_1, S_0viene selezionato il relativo ingresso del multiplexer che abbiamo provveduto precedentemente a forzare ad 1 (VCC) o a 0 (GND), e viene inviato all'uscita **Y**, ottenendo così il funzionamento desiderato dalla tabella di verità.

La simulazione

Mostra che a fronte delle 16 combinazioni dei 4 ingressi S_3, S_2, S_1, S_0l'uscita **Y** è quella desiderata.



Il Semisommatore o Half-Adder

HALF ADDER

La porta XOR - OR esclusivo

Analizziamo innanzitutto il funzionamento di una porta XOR, in quanto risulterà utile nella trattazione del Semisommatore o Half-Adder:

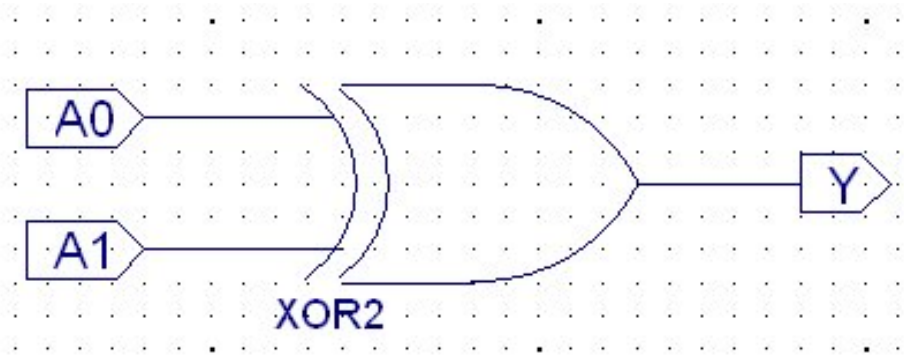


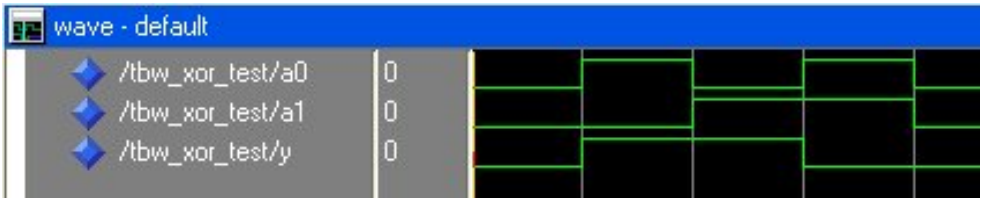
Tabella di verità

La porta ha la seguente tabella della verità:

A_1	A_0	Y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Dove si nota che l'uscita assume il valore 1 logico sole se gli ingressi non sono uguali.

Simulazione



La somma aritmetica

A differenza della somma logica, effettuare la somma aritmetica vuol dire operare come per la somma in base 10, tenendo conto anche di un eventuale riporto che chiameremo CY; se la somma viene effettuata ad una cifra numerica, il riporto interverrà quando la somma supera o eguaglia la base, e nel caso della base 2, effettuando la somma ad una sola cifra (ovvero ad un solo bit per volta) avremo il riporto quando la somma supera 1.

Tabella di verità del Semisommatore

B_0	A_0	SUM	CY
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Come vedete ci troviamo di fronte ad un blocco di logica combinatoria con due ingressi: B_0e A_0, e due uscite: **SUM** (la somma) e **CY** (il riporto). Dovremo quindi realizzare due funzioni booleane distinte, una per ogni uscita del nostro blocco di logica combinatoria, dove ciascuna funzione booleana sarà una funzione a due ingressi ed una sola uscita.

Funzione booleana che realizza la somma - SUM

La funzione è quella che mostra le sue uscite nella terza colonna della tabella a partire da sinistra, ovvero la funzione con uscita **SUM** e ingressi B_0e A_0; scriviamo i minterm:

B_0	A_0	SUM
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

$$SUM = \bar{B}_0 A_0 + B_0 \bar{A}_0$$

E come vedete è identica alla funzione realizzata da una porta **XOR**.

Funzione booleana che realizza il riporto - CY

La funzione è quella che mostra le sue uscite nella quarta colonna della tabella a partire da sinistra, ovvero la funzione con uscita **CY** e ingressi B_0e A_0; scriviamo i minterm:

B_0	A_0	CY
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

$$CY = B_0 A_0$$

E come vedete è identica alla funzione realizzata da una porta **AND**.

Allora andiamo a realizzare lo schema del nostro blocco di logica combinatoria che realizza il **Semisommatore** mettendo insieme i due schemi parziali delle due separate funzioni:

Lo schema del Semisommatore

Questo e lo schema completo di un semisommatore. Si può ora passare all'implementazione sulla scheda e a relativo test, inviando gli ingressi a due switch, ad esempio A_0a SW_1e B_0a SW_2e le uscite SUM e CY ai led LD_1e LD_8.

La simulazione del semisommatore

Il Sommatore completo o Full-Adder

FULL ADDER

Cosa succede se dobbiamo sommare parole aventi più di un bit?

Si dice **parola** un numero binario avente più di una cifra, avremo generalmente a che fare con parole di 4, 8 16, 32 o più bit. Se abbiamo quindi parole con più di un bit, da sommare tra di loro, dobbiamo innanzitutto tenere conto dell'eventuale riporto **CY** ottenuto dalla somma della parola precedente, come facciamo quando si effettua una somma in base 10, per cui oltre che sommare i bit dello stesso peso, tra di loro, si dovrà sommare al risultato ottenuto, cioè al **SUM**, il riporto **CY** ottenuto dalla somma dei bit di peso minore sommati precedentemente?

Come abbiamo visto nel Semisommatore, **CY** può assumere anche il valore 1 logico, e dobbiamo aggiungerlo alla somma dei bit di peso successivo, altrimenti avremo un risultato errato.

È necessario, quindi, tenere conto nella somma di due bit dello stesso peso, del riporto precedente, e dunque dovremo realizzare un sommatore, detto **Sommatore Completo** o **Full-Adder**, apportando dei miglioramenti all' **Half-Adder** visto prima.

Costruiamo quindi la tabella di verità di un sommatore completo, un blocco di logica combinatoria avente come ingressi tre bits, i bit da sommare e il riporto precedente, che avrà come uscite la somma ed un riporto successivo, da sommare ai bits di peso successivo.

Chiamiamo **A**, e **B** i bit da sommare, **CYIN** il riporto precedente, o **CARRY INPUT**, e **CYOUT** o **CARRY OUTPUT** il riporto successivo, e continuiamo a chiamare **SUM** il risultato della somma. La nostra tabella della verità sarà dunque, un'estensione della tabella della verità del **Semisommatore** dove avremo un riporto precedente o **CYIN** che può assumere i valori **0** o **1** logico.

Tabella di verità del Semisommatore

B	A	SUM	CY
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Blocco generico del Semisommatore

Tabella di verità del Sommatore Completo o con riporto

CyIn	B	A	SUM	CyOut
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Come vedete, le prime quattro righe della tabella sono identiche a quelle del Semisommatore avente un riporto precedente, sottinteso, uguale a 0. Nelle ultime quattro righe invece si aggiunge alla somma ottenuta, il riporto che qui è uguale ad 1, il ché produce dei risultati differenti.

Blocco generico del Sommatore Completo

Sintesi dello schema in logica combinatoria

Ora per costruire lo schema relativo a questa tabella possiamo procedere in due maniere differenti:

Costruendo la funzione booleana

Questo è il metodo analitico.

Funzione SUM

$$SUM = \overline{CyIn}\bar{B}A + \overline{CyIn}B\bar{A} + CyIn * \bar{B}\bar{A} + CyIn * BA$$
$$SUM = \overline{CyIn}(\bar{B}A + B\bar{A}) + CyIn * (\bar{B}\bar{A} + BA)$$
$$SUM = \overline{CyIn}(B \oplus A) + CyIn * (\overline{B \oplus A})$$
$$SUM = CyIn \oplus (A \oplus B) = CyIn \oplus A \oplus B$$

Funzione CYOUT

$$CyOut = \overline{CyIn}BA + CyIn * \bar{B}A + CyIn * B\bar{A} + CyIn * BA$$
$$CyOut = \overline{CyIn}BA + CyIn * (B \oplus A) + CyIn * BA$$
$$CyOut = (\overline{CyIn} + CyIn)BA + CyIn * (B \oplus A)$$
$$CyOut = (1)BA + CyIn * (B \oplus A)$$
$$CyOut = BA + CyIn * (B \oplus A)$$

Schema realizzato col procedimento analitico

Utilizzando due semisommatori in cascata

Questo è il metodo intuitivo. Il primo semisommatore per sommare A e B ottenendo una somma ed un riporto, il secondo per sommare alla somma così ottenuta, il riporto precedente.

Schema realizzato col procedimento intuitivo

Esplodendo lo schema si ottiene lo stesso schema ottenuto col procedimento analitico

Simulazione del Full-Adder

Full-Adder realizzato con mux a 3 ingressi di selezione

FULL ADDER CON MULTIPLEXER A 3 INGRESSI DI SELEZIONE

Tabella di verità del Sommatore Completo o con riporto

CyIn	B	A	SUM	CyOut
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Si vuole implementare un sommatore completo (full-adder) tramite l'utilizzo di multiplexer a tre (3) ingressi di selezione. In base a quanto visto nella lezione riguardante i multiplexer utilizzati per implementare una funzione di logica combinatoria, procederemo allo stesso modo. Si tratta (vedi tabella della verità) di implementare una funzione

di logica combinatoria avente tre ingressi, nella fattispecie A, B e CyIn, e due uscite, SUM e CyOut. Quindi in base alla teoria già studiata, il problema si riduce alla risoluzione, tramite naturalmente multiplexer a tre ingressi, di due funzioni di logica combinatoria a tre ingressi ed una uscita. Ovvero risolveremo prima l'uscita riguardante la colonna SUM e poi l'uscita riguardante la colonna CyOut. Infine metteremo insieme le soluzioni trovate al fine di realizzare un unico blocco di logica combinatoria.

Full-Adder realizzato con mux a 2 ingressi di selezione

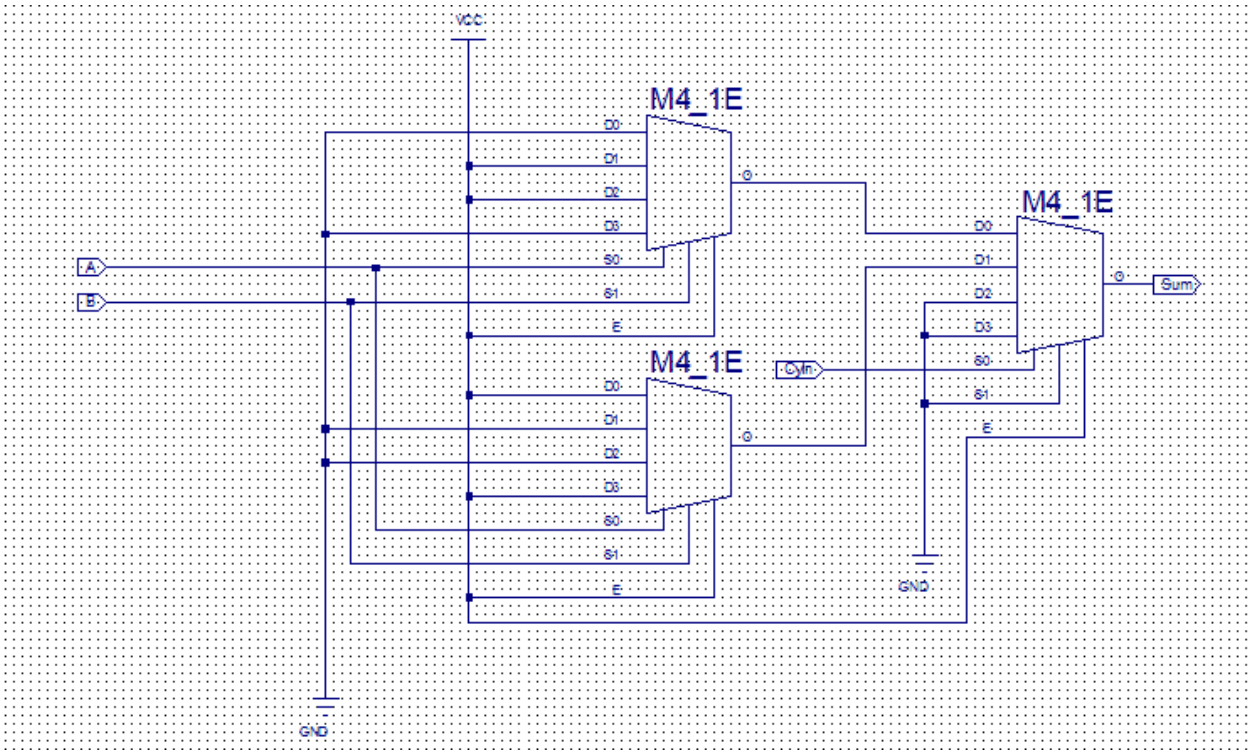
FULL ADDER CON MULTIPLEXER A 2 INGRESSI DI SELEZIONE

Tabella di verità del Sommatore Completo o con riporto

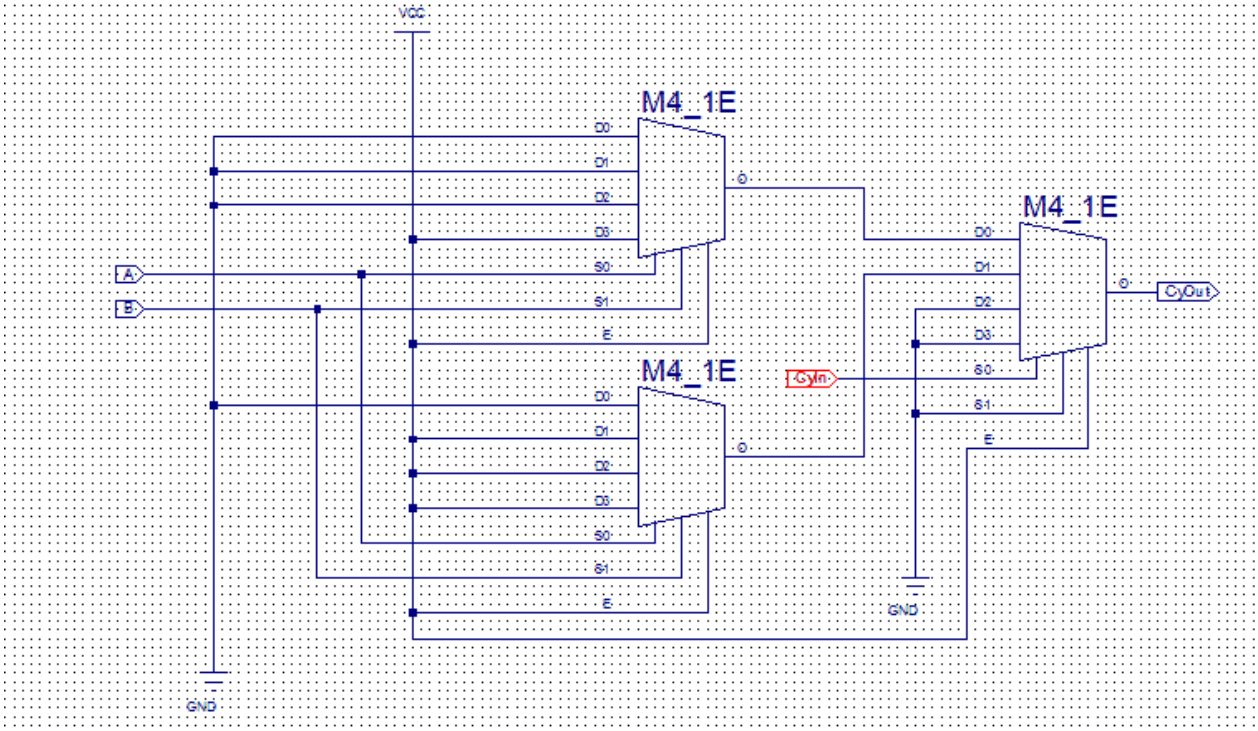
CyIn	B	A	SUM	CyOut
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Si vuole implementare un sommatore completo (full-adder) tramite l'utilizzo di multiplexer a due (2) ingressi di selezione. In base a quanto visto nella lezione riguardante i multiplexer utilizzati per implementare una funzione di logica combinatoria, procederemo allo stesso modo. Si tratta (vedi tabella della verità) di implementare una funzione di logica combinatoria avente tre ingressi, nella fattispecie A, B e CyIn, e due uscite, SUM e CyOut. Quindi in base alla teoria già studiata, il problema si riduce alla risoluzione, tramite naturalmente multiplexer a due ingressi, di due funzioni di logica combinatoria a tre ingressi ed una uscita. Ovvero risolveremo prima l'uscita riguardante la colonna SUM e poi l'uscita riguardante la colonna CyOut. Infine metteremo insieme le soluzioni trovate al fine di realizzare un unico blocco di logica combinatoria. **Sostanziale differenza con la realizzazione del Full Adder che usa i multiplexer a tre ingressi di selezione è che la tabella della verità è stata implementata dividendola in due:** le prime quattro uscite con CyIn a zero (logico) e le seconde quattro uscite con CyIn a uno (logico)

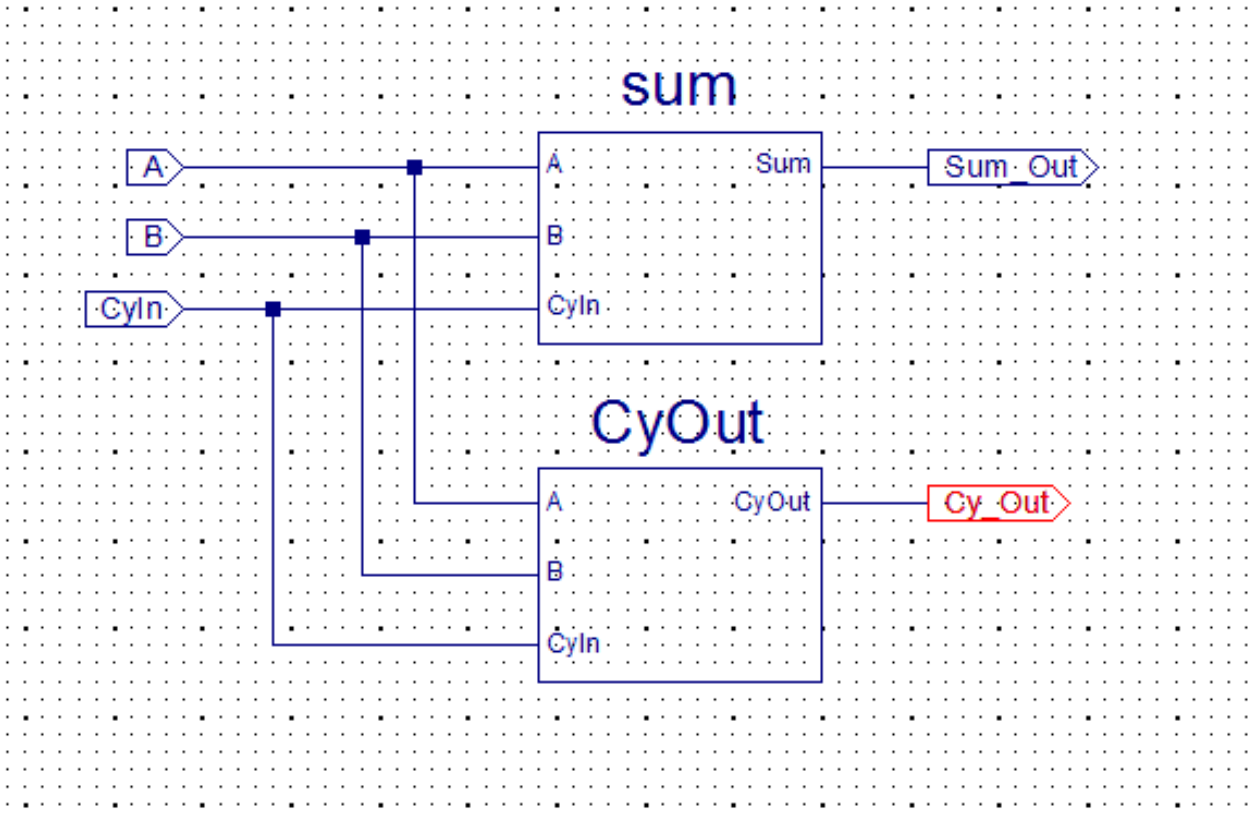
Realizzazione della funzione SUM



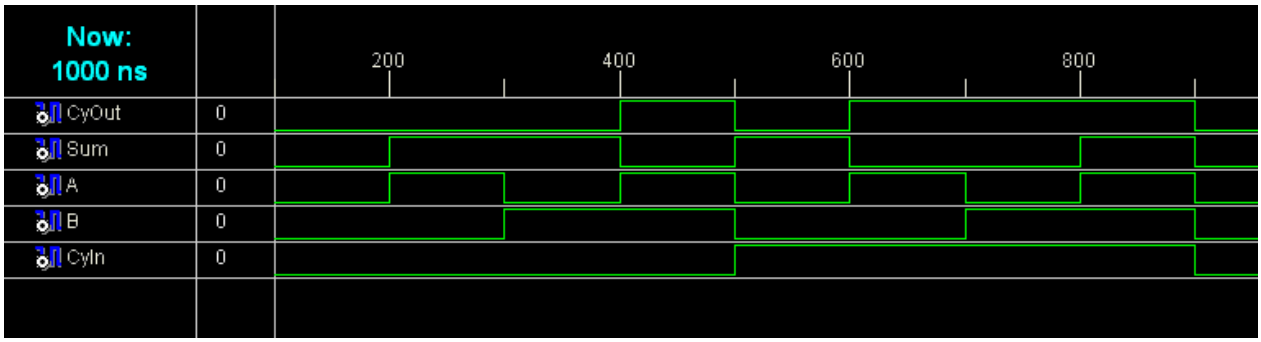
Realizzazione della funzione CyOut



Unione dei due blocchi per le funzioni SUM e CyOut



Simulazione del funzionamento



Sommare due parole di 4 bit

Sommatore completo a 4 bit di tipo RIPPLE CARRY

In questa lezione procederemo ad implementare un **sommatore completo a 4 bit di tipo RIPPLE CARRY** ovvero a propagazione del riporto. Trattasi di un sommatore più lento rispetto alla versione LOOK-AHEAD-CARRY.

Lo schema

La somma delle due parole a bit, **A[3:0]** e **B[3:0],** viene effettuata ponendo in cascata 4 **Full-Adder**, collegati in modo che il riporto della cifra precedente, **CYOUT**, venga inviato all'ingresso di riporto, **CYIN**, della cifra successiva. L'uscita sarà ancora una parola a 4 bit, **SUM[3:0]**, più l'ultimo riporto in uscita, **CYOUT**, ottenendo così un totale di 5 bits in uscita.

- **Non dimenticare mai il riporto in uscita ad una somma se non si vogliono perdere cifre significative.**

Questo risultato è congruente con il fatto che 4 bits contengono i numeri da 0 a 15 e che sommando due parole di 4 bits potremmo ottenere al massimo, quale risultato, il numero 30 che pùo essere contenuto in un numero di almeno 5 bits.

Bus e Tap

Un **BUS** è un insieme di connessioni, **NETS**, che hanno lo stesso nome ma indicizzate. Ad esempio il bus A[3:0], (è questa la sintassi da adoperare per definire un BUS), contiene al suo interno 4 NET che separatamente assumono i nomi **A[0], A[1], A[2], A[3],** e le quali possono essere estratte dal BUS stesso per essere collegate opportunamente tramite gli oggetti che nell'ISE vengono chiamati **TAP**.

NB:la NET che viene collegata al TAP deve essere rinominata in modo congruente con i nomi delle NETS contenute nel BUS dal quale il TAP sta estraendo la singola NET.

A proposito ancora della sintassi, A[3:0] mi dice che il **BUS A** si estende ordinatamente a partire da A[3] fino ad A[0]. In modo equivalente da punto di vista elettronico, avremmo potuto chiamare il BUS A con il nome A[0:3] ottenendo lo stesso risultato, è solo una questione di abitudine. Io preferisco la prima soluzione dove la sintassi è:

NOME_DEL_BUS[MAX_INDEX:MIN_INDEX]

ma come ripeto ugualmente buona è la sintassi:

NOME_DEL_BUS[MIN_INDEX:MAX_INDEX]

Riporto precedente alla cifra successiva

Da questa immagine si vede come il riporto in uscita dalla somma della prima cifra, effettuata col primo sommatore, **CYOUT**, viene inviato all'ingresso del sommatore per le cifre di peso successivo all'ingresso di riporto **CYIN**.

Somma + Riporto

Qui sono evidenziati il BUS somma, SUM[3:0] ed il riporto in uscita dall'ultimo sommatore.

Il riporto in ingresso alla cifra di peso zero viene forzato a GND

Poiché non abbiamo nessun riporto all'inizio della somma è necessaria questa forzatura del primo ingresso **CYIN**.

La piedinatura

Si effettua ora la piedinatura dello schema creando un nuovo file sorgente di tipo **IMPLEMENTATION CONSTRAINTS FILE**. Assegneremo il **BUS A[3:0]** a 4 switch ed il **BUS B[3:0]** a gli altri 4 switch rimanenti; per le uscite faremo in modo di inviare il **BUS SUM[3:0]** a 4 LED ed useremo un quinto LED per il riporto in uscita.

L'implementazione e il test

- Dopo aver effettuato il caricamento del file di programma, **sum4bit.jed** all'interno del device, settare i switch in modo da assegnare a ciascun bus d'ingresso, A[3:0] e B[3:0], un valore numerico da 0 a 15 a scelta.
- Osservare la configurazione binaria dei LEDs; il risultato ottenuto è esatto?
- Rispondere più volte a questa domanda con diverse configurazioni dei bus, in modo da controllare l'esatto funzionamento del sommatore.

La simulazione

Sono stati assegnati alcuni valori ai bus di ingresso; osservare la configurazione del bus di uscita, SUM[3:0] e del bit di riporto, CYOUT.

NB: Una simulazione esaustiva, comprendente cioè tutte le possibili combinazioni dei due bus, qui non viene effettuata perché altrimenti troppo lunga in quanto comprenderebbe $16 * 16 = 256$combinazioni differenti.

Sommatore a 4 bit di tipo LOOK AHEAD CARRY

SOMMATORE DI TIPO LOOK-AHEAD-CARRY

Sintesi del circuito per la Generazione e propagazione del riporto

Nasce da alcune osservazioni sulla tabella della verità del **Full Adder** con l'obiettivo di realizzare un sommatore più veloce del **RIPPLE-CARRY** che è ancora una soluzione provvisoria per un sommatore, in quanto si degna di darci un risultato soltanto quando tutti i circuiti hanno raggiunto una condizione di stabilità dovuta ai ritardi di propagazione interni ai circuiti stessi. Qui si vuole in qualche modo **prevedere in anticipo** quali saranno i riporti in ingresso ai vari stadi del circuito in modo che in un solo passo tutti i **Full-Adder** eseguano la propria somma, in un certo senso **in parallelo**.

Tabella di verità del Sommatore Completo o con riporto

Ci	B	A	SUM	Co
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

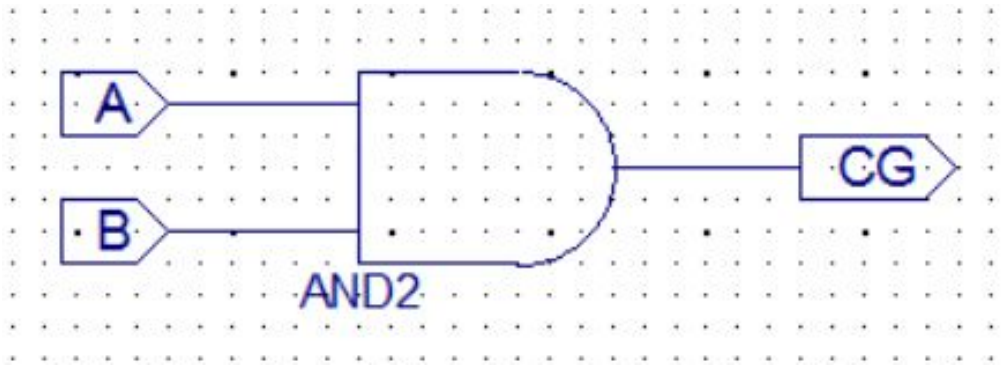
Carry-Generate

- Quando **A = 1** e **B = 1** indipendentemente da **Ci** viene generato un riporto **Co = 1**

Isolo le righe della tabella del Full-Adder che mi interessano nella seguente sotto-tabella:

Ci	B	A	SUM	Co
0	1	1	**0**	**1**
1	1	1	**1**	**1**

Chiamo **Carry-Generate** la funzione **CG = AB** relativa all'ultima colonna della tabella precedente.



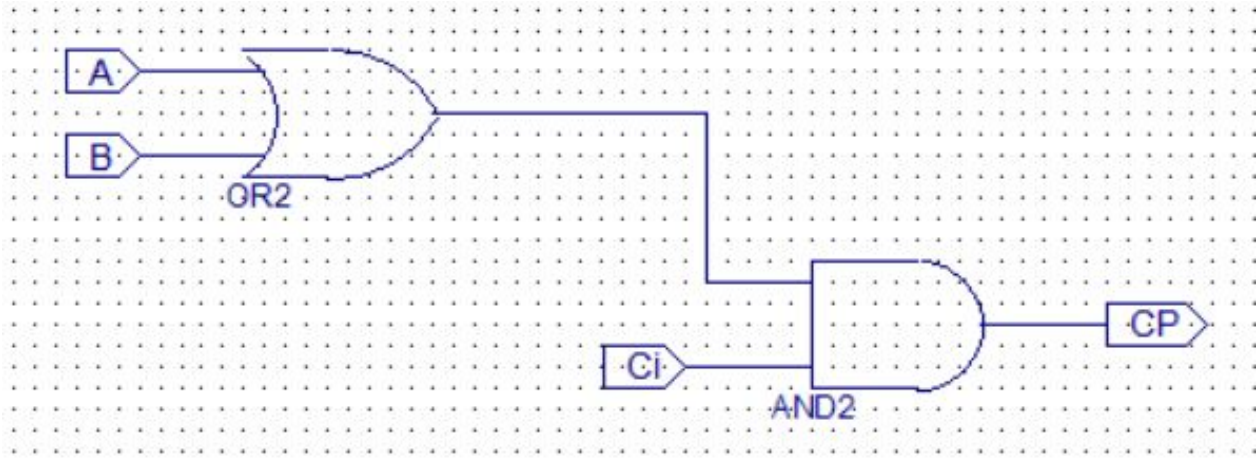
Carry-Propagate

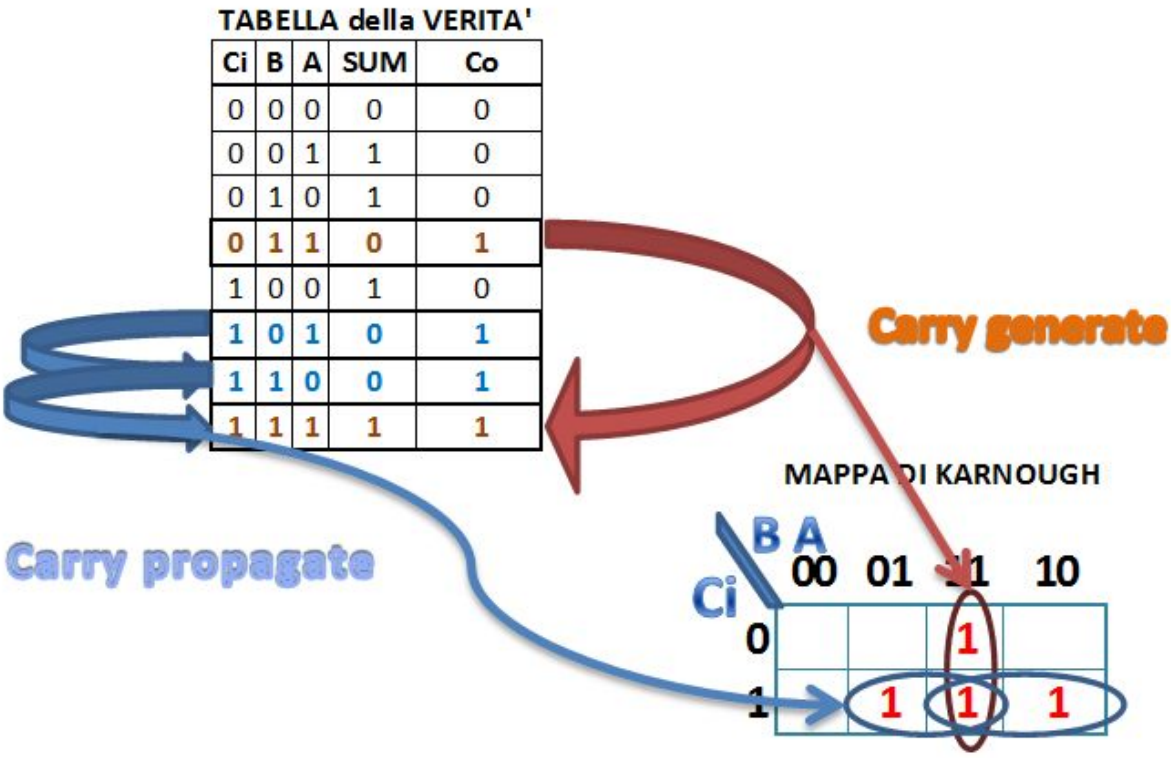
- Se **Ci = 1** allora **Co = 1 sse**(se e solo se) **A = 1** oppure **B = 1** oppure sia **A = 1** che **B = 1**

Anche in questo caso, isolo le righe della tabella del Full-Adder che mi interessano e costruisco un'altra funzione di logica combinatoria:

Ci	B	A	SUM	Co
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Chiamo **Carry-Propagate** la funzione **CP = A + B**; il riporto vale **CP * Ci**



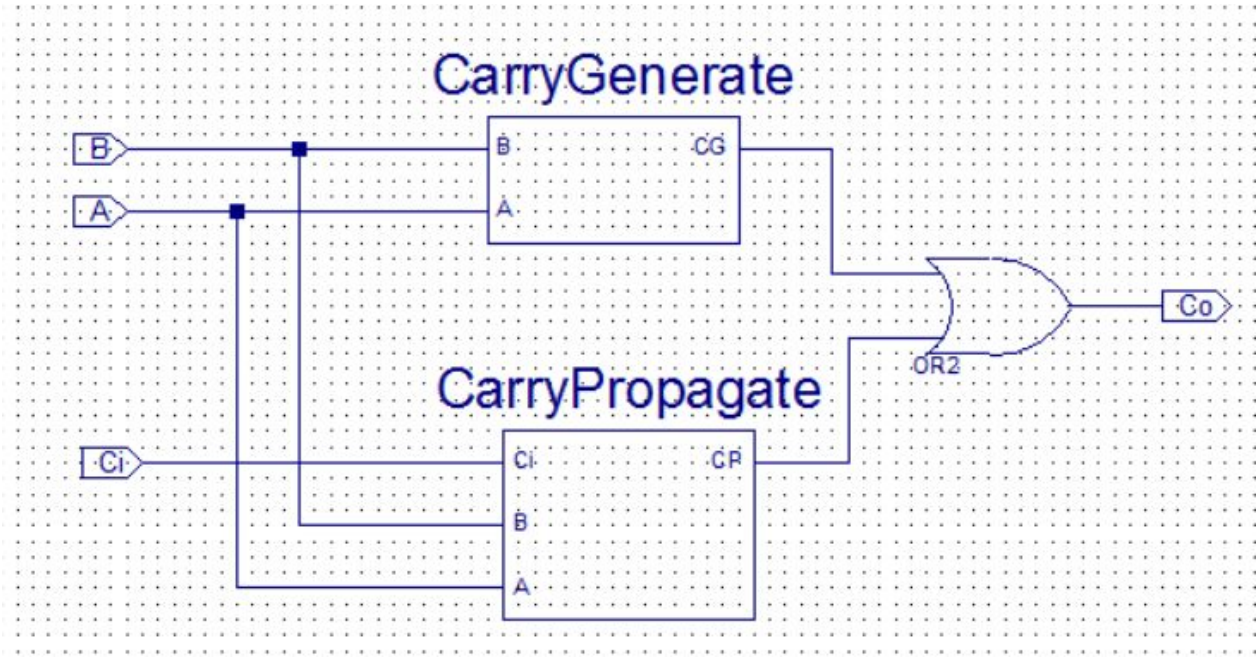


Carry-Out

Tenendo conto di entrambe le conclusioni delle due precedenti osservazioni, otteniamo il riporto in uscita:

Co = CG + CP * Ci

Possiamo ritenere questa conclusione valida? La risposta è Sì! perché abbiamo costruito la logica-combinatoria che realizza la colonna del riporto in uscita al **Full-Adder** tenndo conto di tutte le righe aventi uscita uguale a **'1** e poi le abbiamo sommate anche se abbiamo preso in considerazione qualche riga più di una volta soltanto, e direi che va evidenziato il fatto che non sempre una semplificazione del circuito con **Karnaugh** si rende necessaria, ma che ogni elaborazione di un circuito va sempre presa in considerazione delle finalità, cioè degli obiettivi che ci si propone di raggiungere, come nel nostro caso in cui stiamo cercando di realizzare un sommatore più veloce del tipo **RIPPLE-CARRY** visto in precedenza.



Espressione dei riporti per sommatore a 4 BITS

Sommatore di peso 0

$$Co_0 = CG_0 + CP_0 * Ci_0$$

Sommatore di peso 1

Essendo $Ci_1 = Co_0$ottengo:

$$Co_1 = CG_1 + CP_1 * Ci_1 = CG_1 + CP_1 * (CG_0 + CP_0 * Ci_0) = CG_1 + CP_1 * CG_0 + CP_1 * CP_0 * Ci_0$$

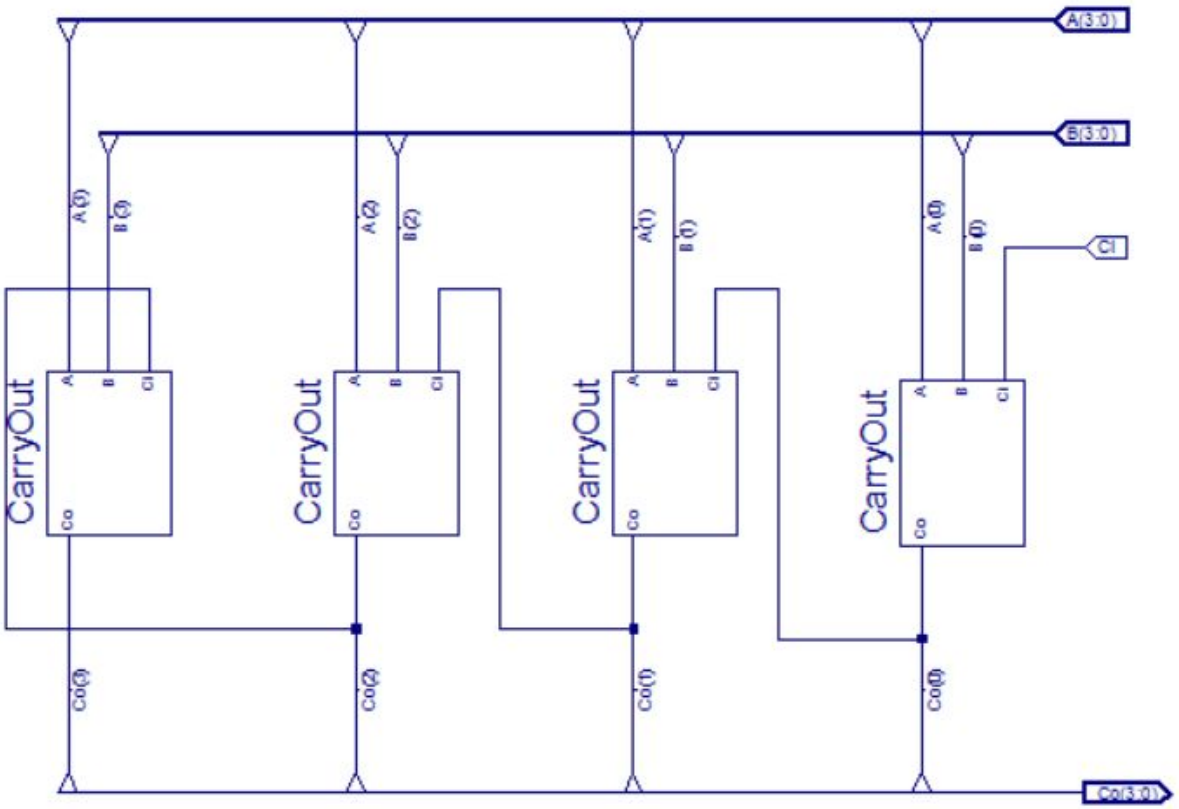
- **Es4**: Ricavare le funzioni successive e costruire il CIRCUITO GENERATORE DI RIPORTI

Lo schema avrà nome **RipGen**

- **Es5**: Realizzare un sommatore Look-Ahead-Carry a 4 bits

Lo schema si chiamerà **Sum4BitLAC** = CG_1+CP_1*CG_0+CP_1*CP_0*Ci_0</math> = CG_1+CP_1*CG_0+CP_1*CP_0*Ci_0</math>

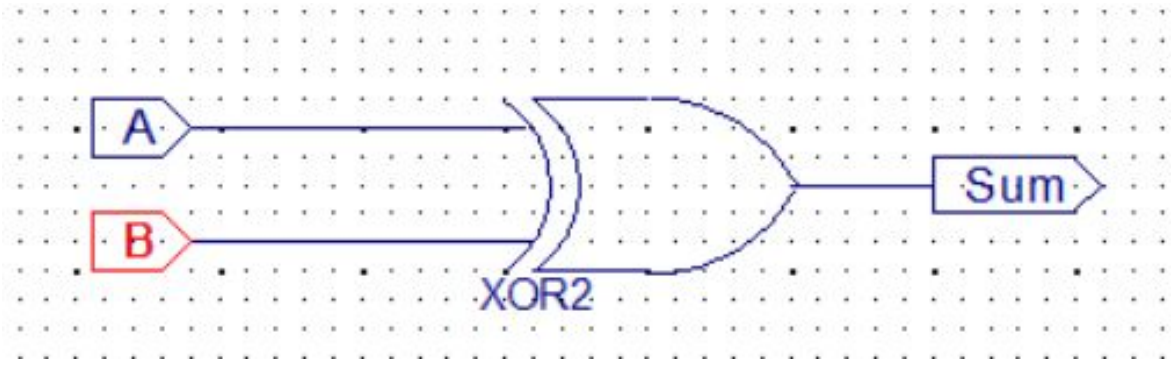
Circuito per la Generazione e propagazione del riporto



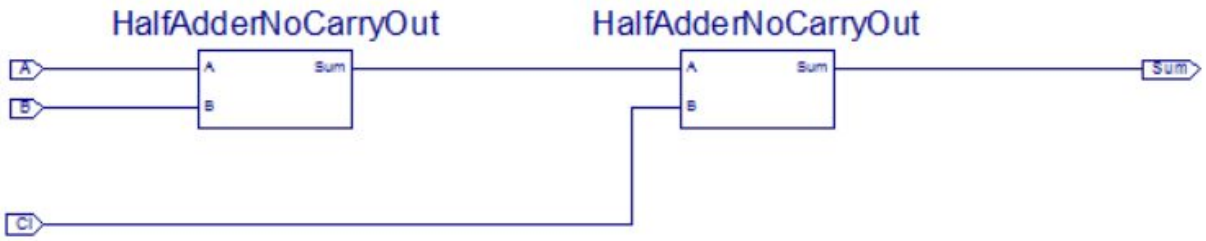
Assemblaggio del sommatore LAC

A questo punto possiamo aggiungere i sommatori completi o full-adder privi della circuiteria preposta alla generazione del carry-out.

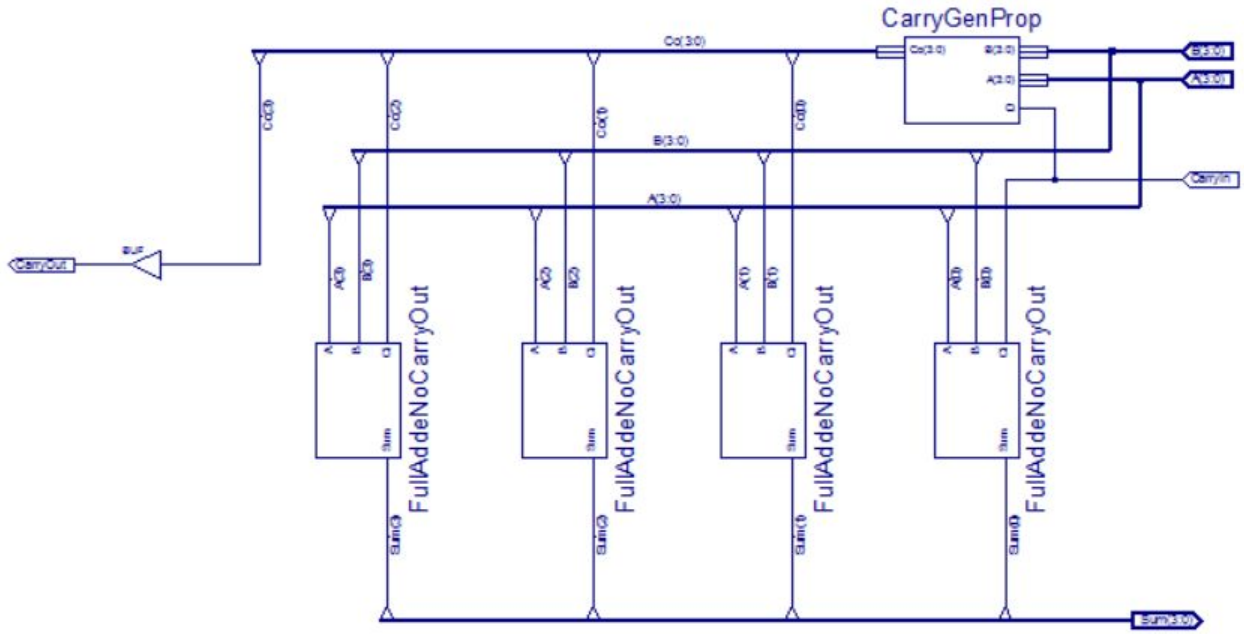
Semi-sommatore privato del carry-out



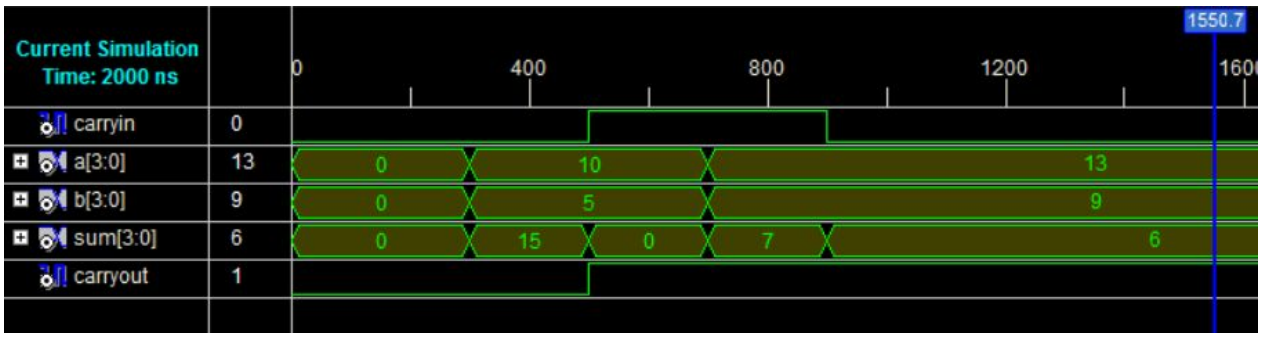
Sommatore completo privato del carry-out



Sommatore a 4 bit di tipo Look-Ahead-Carry assemblato



Simulazione del sommatore LAC



Il Comparatore

Comparatore ad un bit

Vogliamo risolvere la domanda: **il bit A è maggiore, minore o uguale a B**?

La tabella di verità

A	**B**	$(A > B)$	$(A < B)$	$(A = B)$
0	0	**0**	**0**	**1**
0	1	**0**	**1**	**0**
1	0	**1**	**0**	**0**
1	1	**0**	**0**	**1**

Formule booleane per la sintesi del circuito digitale:

```
Scriviamo ora le espressioni booleane che si riferiscono alle
colonne di cui sopra...
```

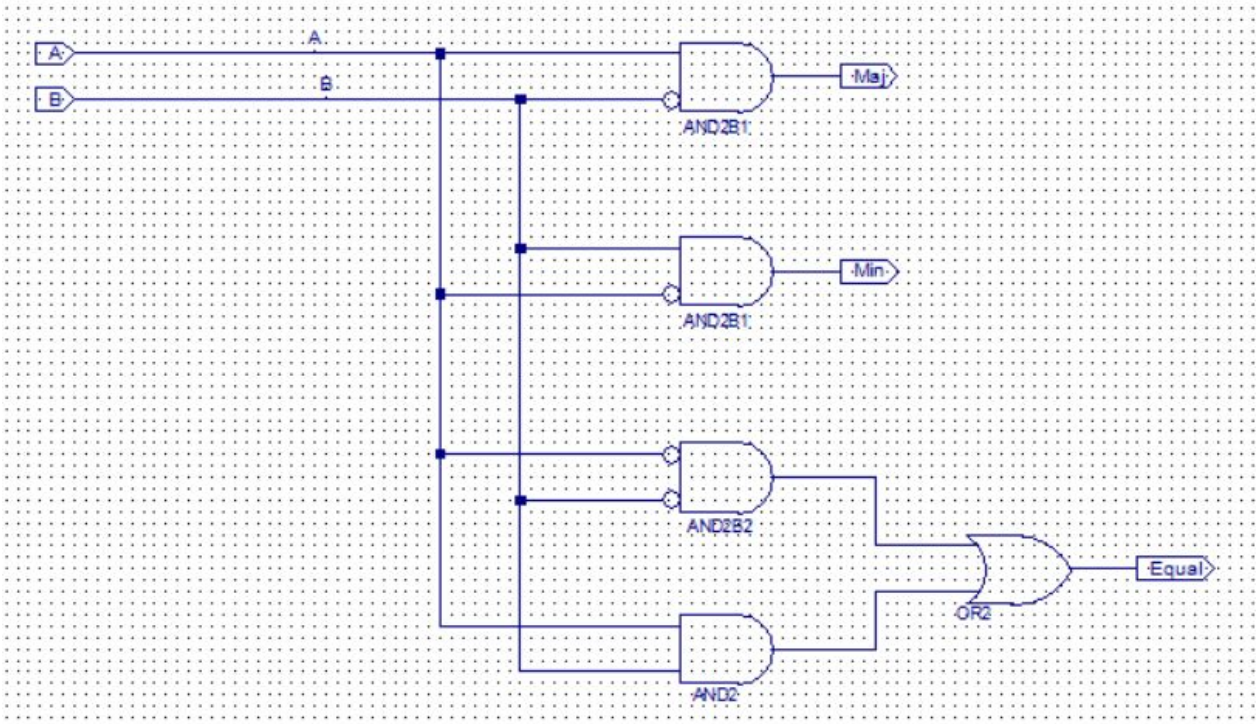
- $(A > B)$

  $$(A > B) = A\bar{B}$$
- $(A < B)$

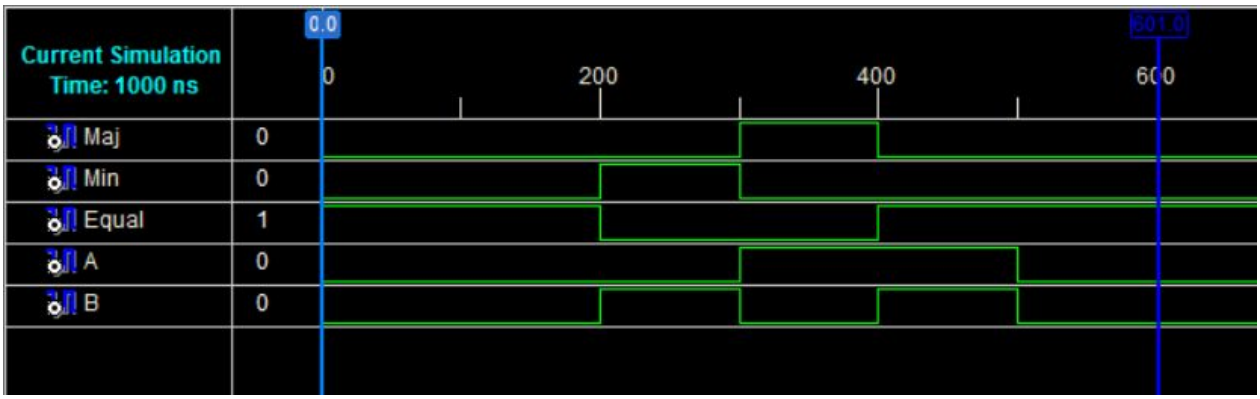
  $$(A < B) = \bar{A}B$$
- $(A = B)$

  $$(A = B) = \bar{A}\bar{B} + AB = \overline{A \oplus B}$$

Lo schema



Il diagramma temporale



Il Comparatore a più bit

Il Comparatore di parole di dimensione maggiore di 1 Bit

Se ci pensi un attimo ti accorgi che comparare parole binarie con dimensione maggiore di 1 bit richiede un confronto bit x bit, richiede in definitiva la ripetizione di un confronto che al massimo richiede **N passi** pari al numero **N** dei bit che compongono la parola di dimensione maggiore tra le due poste a confronto.

È un po' complicato? Beh! allora diciamo che per dire che una parola è maggiore di un'altra basta partire con il confronto delle cifre di peso maggiore **(The High Significative Bit)**: Se i 2 bit messi a confronto sono diversi allora posso dire subito quale delle due parole è maggiore e quale è minore: **È maggiore quella con il bit più significativo ad 1 logico** mentre è minore quell'altra. E fino a qui ci siamo, l'algoritmo termina con un risultato chiaro ed evidente. Se invece i due bit sono uguali allora devo per forza passare ad analizzare i 2 bit di peso minore di quelli appena posti sotto analisi, in definitiva mi devo spostare nelle parole verso destra, verso il bit meno significativo ovvero il bit di peso=0.

Quindi se proprio sono sfortunato, e le parole differiscono di una sola unità dovrò ripetere l'algoritmo di confronto **n volte** cioè un numero di volte pari alla dimensione **n** della parola di dimensione maggiore fra le due.

Comparatore a 2 bit in Logica Combinatoria

- Devo esprimere la Tabella della verità in modo esaustivo, devo cioè considerare tutte

le combinazione di 2 bit e metterli a confronto e poi posso sintetizzare il circuito

con le metodologie già introdotte nella prima parte del corso.

Comparatore a più bits in Logica Sequenziale (Algoritmo)

1. Devo progettare un circuito automatico in logica sequenziale che analizzi la dimensione dei due numeri da confrontare per conoscere quale è la dimensione maggiore.
2. Passo quindi a sottoporre i due numeri al **Comparatore ad 1 Bit** precedentemente spiegato, a partire dai bit di peso più significativo
3. Se sono diversi ho finito e la macchina (che devo ancora progettare) si bloccherà e mi darà l'esito del confronto: quale parola è **maggiore** e quale è **minore**.
4. Altrimenti devo passare in pasto al **Comparatore ad 1 Bit** i due bit successivi.
5. Se erano gli ultimi 2 bit, cioè quelli meno significativi ho finito e dico che **le parole sono uguali**.
6. Altrimenti devo tornare al punto 2.

Progetto di un comparatore automatico a più bit

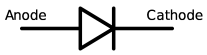
NB: Questa parte di programma richiede che si conoscano gli argomenti nei capitolo di Logica Sequenziale

Logica Sequenziale

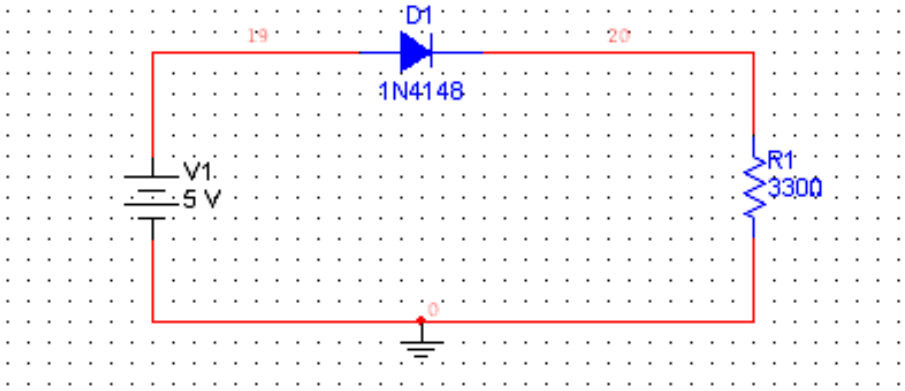
Il decodificatore da BCD a sette segmenti

Il Display a 7 Segmenti

Il Diodo

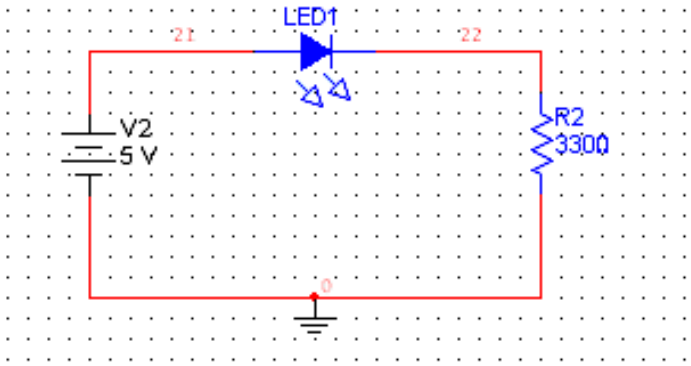


Polarizzazione diretta

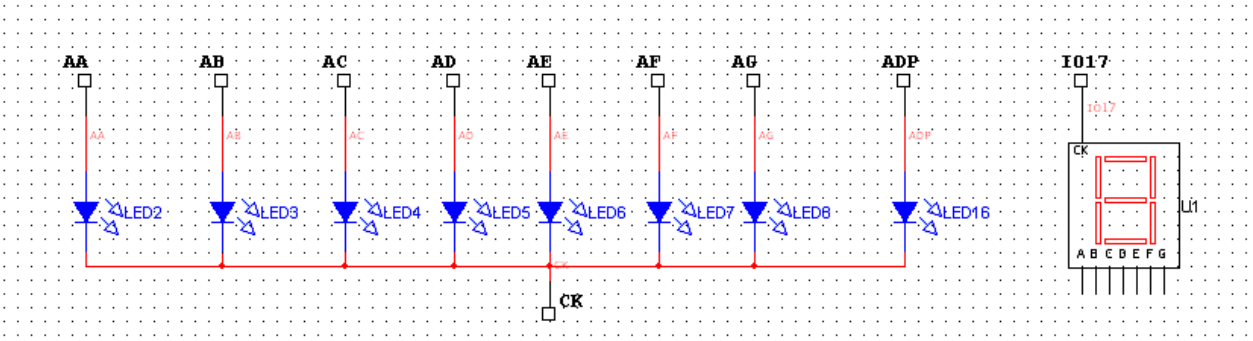


Il Diodo Led

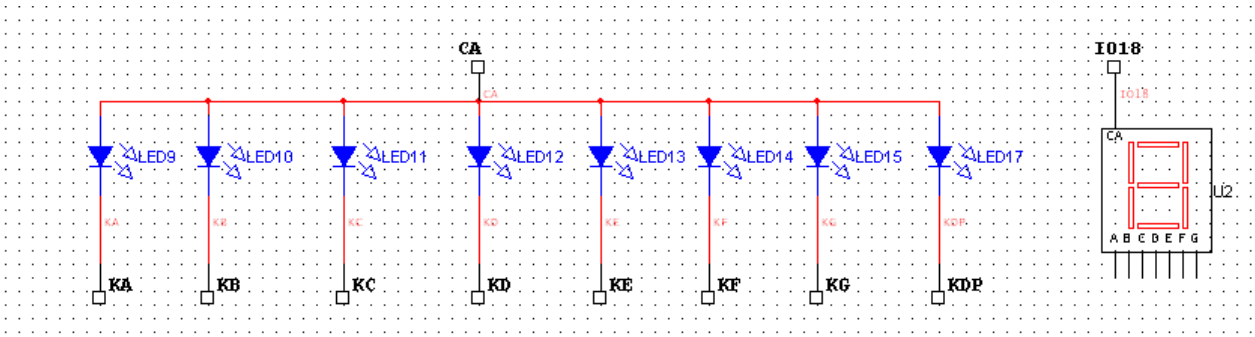
Polarizzazione diretta



Display a catodo Comune (CK sta per Common Katode)



Display a anodo Comune (CA sta per Common Anode)



Come devono essere illuminati i segmenti

I 7 segmenti rappresentano le uscite del nostro **decodificatore da BCD a 7 Segmenti**. La tabella mostra quali segmenti devono essere illuminati (1 logico) per rappresentare la cifra decimale nella colonna di sinistra.

BCD: Binary Coded Decimal

Significa: **la codifica in binario dei numeri decimali**; come si vede necessitano 4 bits per ogni cifra decimale da 0 a 9. Tale codifica rappresenterà gli ingressi del nostro **decodificatore da BCD a 7 Segmenti**, che ci apprestiamo a studiare. Questa lezione sarà anche l'occasione per introdurre il linguaggio VHDL e comprenderne la sua potenza di rappresentazione di un hardware paragonata alla modalità di introduzione dello stesso hardware quale schema elettrico.

La seguente tabella della verità si riferisce ad un display a catodo comune. Infatti per accendere un segmento del display questi deve essere portato al valore logico **1** mentre il catodo-comune deve essere forzato verso massa (gnd) in modo da chiudere il circuito affinché nel segmento scorra la corrente sufficiente a far accendere il diodo led.

A_3	A_2	A_1	A_0	DEC	a	b	c	d	e	f	g
0	0	0	0	**0**	1	1	1	1	1	1	0
0	0	0	1	**1**	0	1	1	0	0	0	0
0	0	1	0	**2**	1	1	0	1	1	0	1
0	0	1	1	**3**	1	1	1	1	0	0	1
0	1	0	0	**4**	0	1	1	0	0	1	1
0	1	0	1	**5**	1	0	1	1	0	1	1
0	1	1	0	**6**	1	0	1	1	1	1	1
0	1	1	1	**7**	1	1	1	0	0	0	0
1	0	0	0	**8**	1	1	1	1	1	1	1
1	0	0	1	**9**	1	1	1	1	0	1	1

In definitiva...

Abbiamo a che fare, con un blocco di logica combinatoria con **4 ingressi e 7 uscite**, si dovranno dunque sintetizzare **7 funzioni booleane da 4 ingressi ad una uscita**.

Realizzazione con Multiplexer

Clicca su un'immagine per ingrandirla...

I Sette Segmenti del display pilotati da 7 logiche combinatorie da 4 a 1 e realizzate con il multiplexer. Gli ingressi di selezione costituiscono gli inputs per il codice BCD da decodificare.

Realizzazione con Porte logiche

Mappe di Karnaugh

Il decodificatore da BCD a 7 segmenti in VHDL

Il file VHDL

Qui di seguito il contenuto di una file in linguaggio vhdl che realizza un decodificatore da BCD a 7 segmenti:

clausola library

library IEEE; **use** IEEE.STD_LOGIC_1164.ALL; **use** IEEE.STD_LOGIC_ARITH.ALL; **use** IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are -- provided for instantiating Xilinx primitive components. --library UNISIM; --use UNISIM.VComponents.all;

clausola entity

entity decBCD7seg is

```
   Port ( d : in std_logic_vector(3 downto 0);
          disp : out std_logic_vector(6 downto 0));
```

end decBCD7seg;

clausola architecture

architecture Behavioral of decBCD7seg is

begin **process**(d) begin

```
    case d is
       when "0000" => disp <= "1111110";
       when "0001" => disp <= "0110000";
       when "0010" => disp <= "1101101";
       when "0011" => disp <= "1111001";
       when "0100" => disp <= "0110011";
       when "0101" => disp <= "1011011";
       when "0110" => disp <= "1011111";
       when "0111" => disp <= "1110000";
       when "1000" => disp <= "1111111";
       when "1001" => disp <= "1111011";
       when others => disp <= "0000000";
    end case;
```

end process;

end Behavioral;

Da cui si evince la relativa facilità di realizzazione di un hardware, se comparata con l'implementazione dello stesso fatta utilizzando blocchi schematici.

Procedura

3- Lanciare il processo Create Schematic Symbol

Ottenendo così il nostro blocco di logica combinatoria pronto per essere inserito in un altro schema.

Schema ottenuto dal processo Create Schematic Symbol

Simulazione del decodificatore

NB:

disp[6] corrisponde al segmento **A** del display

e così via...fino a

disp[0] che corrisponde al segmento **G** del display

Multiplexing di più display a 7 segmenti

- **Schema a Blocchi:**

Il **clock** assumendo alternativamente gli stati logici *1* e *0* quando applicato al multiplexer seleziona il bus a 4 bits da inviare al decodificatore ed essendo anche applicato ai catodi dei display, (con una semplice decodifica ad 1 bit), attiva il display il cui catodo riceve un valore logico *1*.

- **Il multiplexer da 2 a 1 x 4:**

È costituito da 4 mux da 2 a 1 collegati in parallelo ai due bus di dimensione 4.

Logica Sequenziale

Teoria

La logica sequenziale tratta di sistemi digitali le cui uscite non dipendono solo dai valori attuali degli ingressi, ma anche dai valori che le uscite avevano in precedenza.

Ciò significa che, anche se vi è la stessa combinazione (sequenza di "1" e "0") agli ingressi della macchina, non è detto che le uscite assumano sempre gli stessi valori. Le uscite dipendono anche dai valori che erano presenti in uscita prima che fosse modificata la combinazione in ingresso

Per **Macchine a Stati Finiti**, si intendono sistemi digitali sequenziali suscettibili di assumere un numero finito di stati (combinazioni delle uscite interne).

Output della Macchina

L'Output è una funzione degli Input al periodo T e degli Output al Periodo T-1:

$$Output_T = f_{\{}Input_T, Output_{T-1}\}$$

In termini pratici abbiamo a che fare con blocchi di hardware dove le uscite vengono riportate in ingresso dopo essere state sottoposte ad opportune trasformazioni per poter essere elaborate insieme agli altri Input ed in modo da determinare quale sarà lo stato che assumerà la macchina al prossimo passaggio di stato, ovvero in corrispondenza del prossimo scatto della macchina, che avviene in pratica in corrispondenza del fronte di salita del clock.

Esercitazioni di Logica Sequenziale

Iniziamo ad implementare e testare delle piccole macchine a stati , che costiuiranno anche le nostre semplici primitive da cui poi, far scaturire macchine un po' più complesse.

I Flip Flop

- Flip Flop di tipo D
- Flip Flop di tipo JK
- Flip Flop di tipo T

I Contatori e/o Divisori

- Contatore e-o divisore modulo 2
- Contatori in cascata

Flip Flop di tipo D

Il funzionamento

Il flip flop di tipo D è una cella elementare di memoria per un singolo bit. Assume 2 stati:

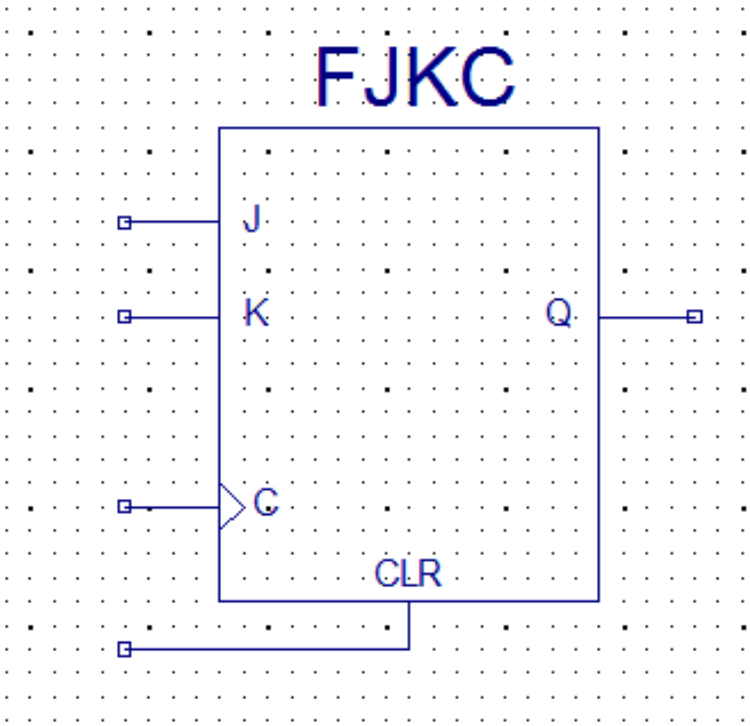
Caricamento

Avviene quando è presente un fronte di salita nell'ingresso **C** (clock); in corrispondenza del fronte di salita, Q <= D ovvero Q assume il valore binario presente in D.

Memoria

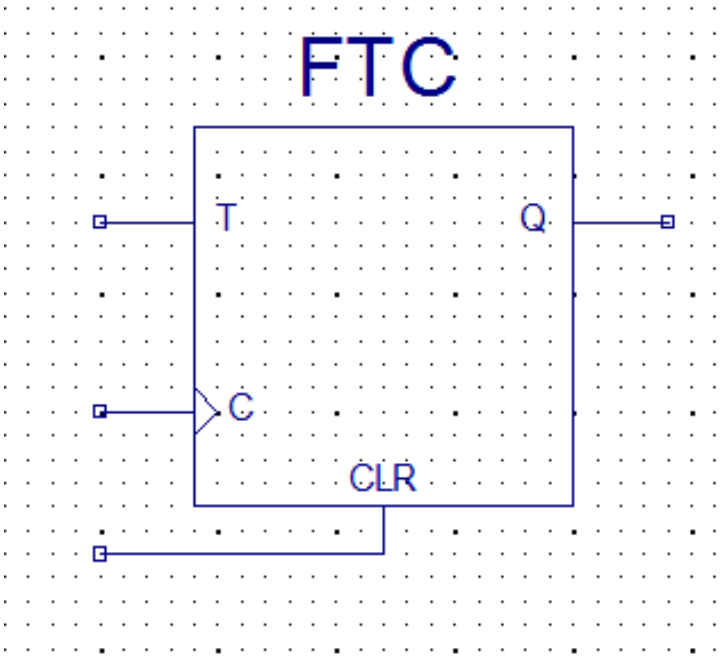
Per tutto il resto del periodo dell'onda quadra nell'ingresso **C** il flip-flop di tipo D si comporta come una cella di memoria e non importa quale valore assuma il segnale all'ingresso **D**. Q continua a mantenere il valore caricato in corrispondenza del fronte di salita.

Flip Flop di tipo JK



- Flip Flop di tipo JK con Clear

Flip Flop di tipo T



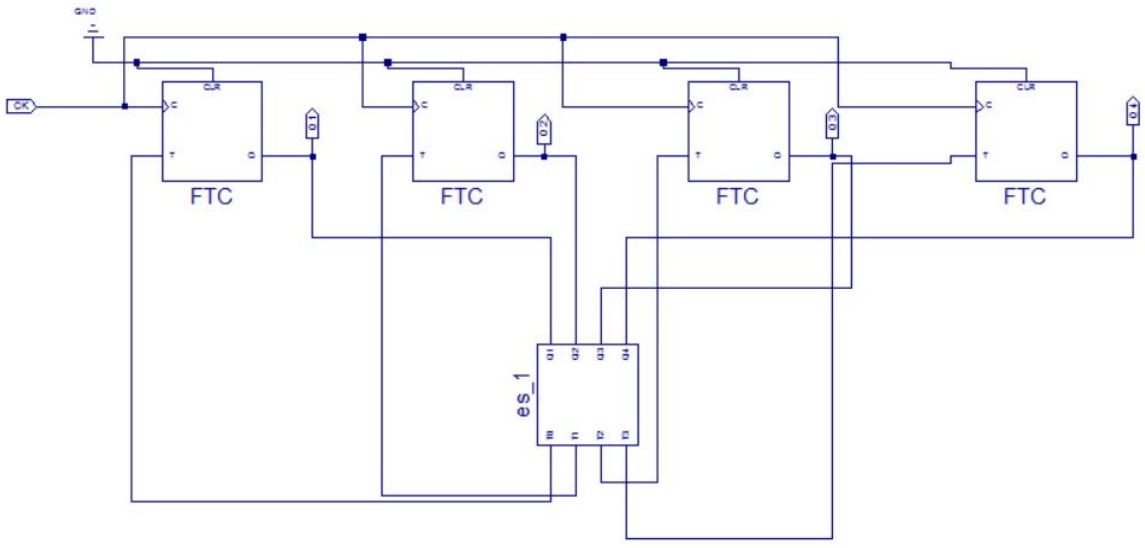
- Flip Flop di tipo T (JK in configurazione "toogle") con Clear

C	T	Q(n)	Q(n+1)	CLR
0->1	0	0	0	0
0->1	0	1	1	0
0->1	1	0	1	0
0->1	1	1	0	0

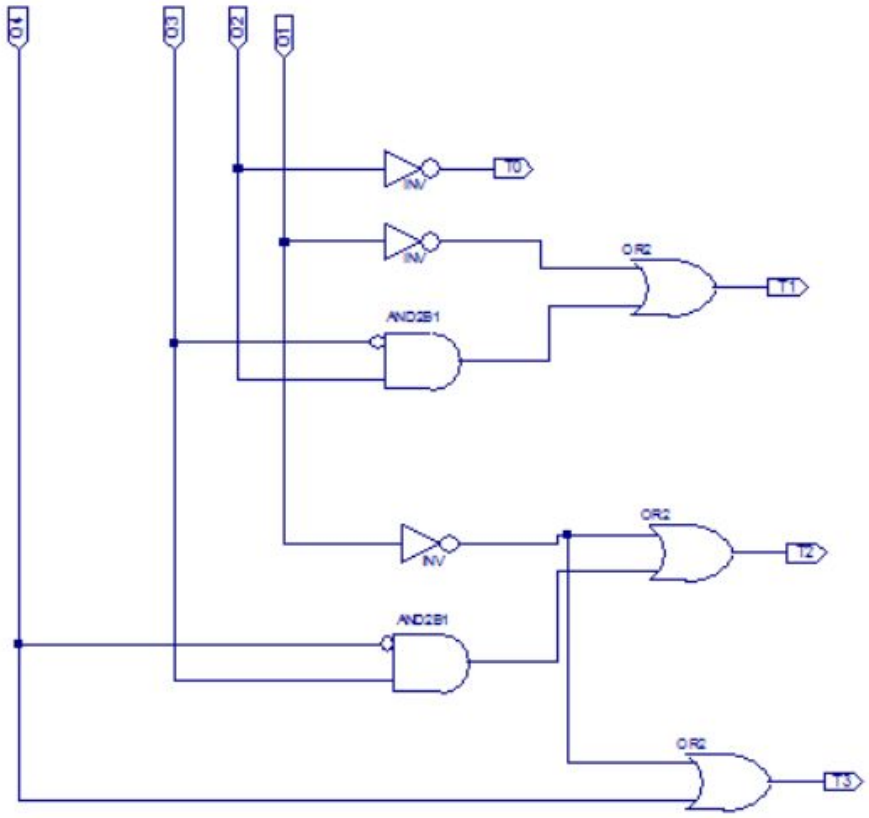
NB: Il CLR è stato disattivato; se viene posto ad 1 in qualsiasi momento

l'uscita andrà a 0, trattandosi di un ingresso asincrono **(ovvero indipendente dal clock C)**

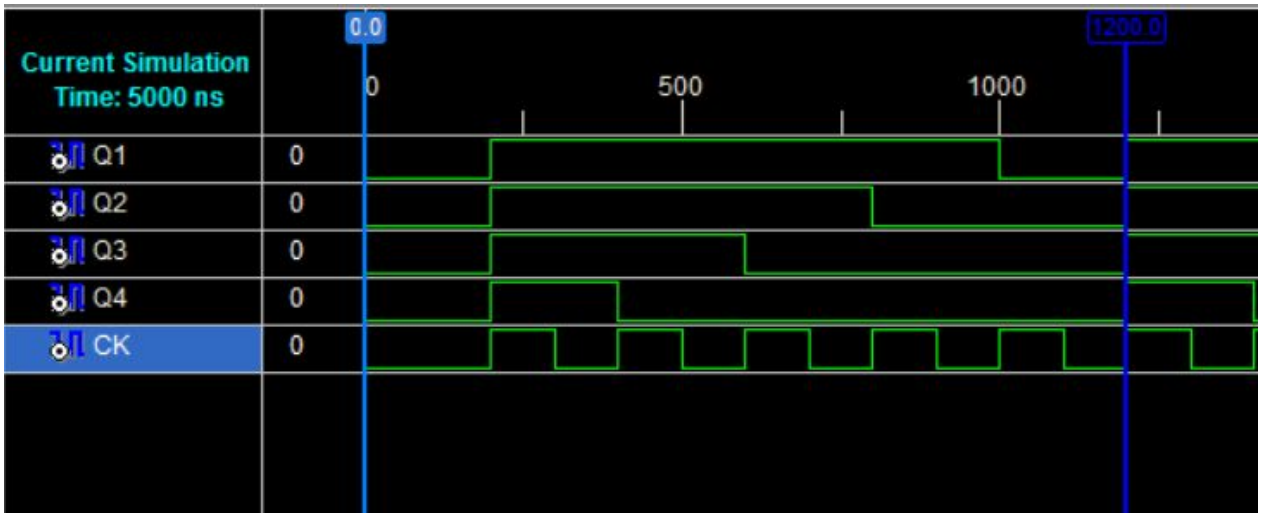
- Esempio di Contatore a 5 stati con FF TOOGLE:



- Logica Combinatoria associata al contatore (Vedi blocco Es_1 della figura precedente):



- Diagramma temporale degli stati del contatore:



- Esempio di file di piedinatura per il trasferimento su hardware:

```
#PACE: Start of PACE I/O Pin Assignments
NET "Q1"   LOC = "p41"  ;
NET "Q2"   LOC = "p40"  ;
NET "Q3"   LOC = "p39"  ;
NET "Q4"   LOC = "p37"  ;
NET "CK"   LOC = "P2"   ;
```

- **NB: Realizzato dalla 4C SISTEMI dell'IPIA ROSA LUXEMBURG A.S. 2009/2010**

Contatore e-o divisore modulo 2

Contatore/Divisore Modulo 2

Modulo

Sta ad indicare il numero di stati che assume il contatore; ad esempio per un contatore modulo 2 si hanno 2 stati possibili e si riferiscono all'uscita del flip-flop che può assumere il valore binario '0' oppure '1'.

Contatore o Divisore?

Entrambi i nomi sono possibili; si chiama **Contatore** se ci mettiamo nell'ottica degli stati che la macchina assume; si chiama **Divisore** se consideriamo la frequenza dell'onda quadra presente all'uscita **Q** del flip-flop in relazione alla frequenza dell'onda quadra applicata all'ingresso di **Clock**. Nel nostro caso viene appunto, divisa per 2.

Realizzazione con FF di tipo D

Simulazione

Come si può vedere, il circuito si comporta come un contatore modulo 2, in quanto l'uscita assume 2 stati **0** oppure **1**; nello stesso tempo è un divisore x 2 in quanto il periodo del segnale presente all'uscita **Q** è esattamente il doppio del periodo del segnale applicato all'ingresso di **Clock**.

Contatori in cascata

Più blocchi di contatori in cascata realizzano un contatore il cui modulo è il prodotto dei moduli dei singoli blocchi. (*Discorso identico vale per i divisori*)

Contatore Modulo 4 all'indietro:

Simulazione

Divisore Modulo 4

Questo contatore è anche un divisore modulo 4. Infatti, dal diagramma temporale si nota che:

$f_{q_0} = f_{Clock}/2$
$f_{q_1} = f_{Clock}/4$

Macchine a Stati Finiti

Macchine a Stati Finiti - FSM

Automi di Mealy e di Moore

La differenza fondamentale tra un ASF di Mealy e un ASF di Moore è la seguente:

```
* Nel modello di Mealy le uscite sono funzione sia degli stati che degli ingressi: W: QxI → U
* Nel modello di Moore le uscite sono funzioni solo degli stati: W: Q → U
```

È stato dimostrato che i due modelli sono equivalenti e che è possibile trasformare una automa di Mealy in uno di Moore che esibisce lo stesso comportamento terminale.

ESEMPI:

Contatore Up & Down

File VHDL della Macchina a Stati

LIBRARY ieee; USE ieee.std_logic_1164.all;

ENTITY SHELL_UPDWCOUN IS PORT (CLK,C,RESET: IN std_logic; Y0,Y1,Y2 : OUT std_logic); END;

ARCHITECTURE BEHAVIOR OF SHELL_UPDWCOUN IS

SIGNAL STATE0, next_STATE0, STATE1, next_STATE1, STATE2, next_STATE2, STATE3 , next_STATE3, STATE4, next_STATE4, STATE6, next_STATE6 : std_logic; SIGNAL next_Y0,next_Y1,next_Y2 : std_logic; SIGNAL Y : std_logic_vector (2 DOWNTO 0); BEGIN

PROCESS (CLK, RESET, next_STATE0, next_STATE1, next_STATE2, next_STATE3, next_STATE4, next_STATE6, next_Y2, next_Y1, next_Y0) BEGIN IF (RESET='1') THEN STATE0 <= '1'; STATE1 <= '0'; STATE2 <= '0'; STATE3 <= '0'; STATE4 <= '0'; STATE6 <= '0'; Y2 <= '0'; Y1 <= '0'; Y0 <= '0'; ELSIF CLK='1' AND CLK'event THEN STATE0 <= next_STATE0; STATE1 <= next_STATE1; STATE2 <= next_STATE2; STATE3 <= next_STATE3; STATE4 <= next_STATE4; STATE6 <= next_STATE6; Y2 <= next_Y2; Y1 <= next_Y1; Y0 <= next_Y0; END IF; END PROCESS;

PROCESS (C,STATE0,STATE1,STATE2,STATE3,STATE4,STATE6,Y) BEGIN

IF ((C='1' AND (STATE1='1')) OR (C='0' AND (STATE6='1'))) THEN next_STATE0<='1'; ELSE next_STATE0<='0'; END IF;

IF ((C='0' AND (STATE0='1')) OR (C='1' AND (STATE2='1'))) THEN next_STATE1<='1'; ELSE next_STATE1<='0'; END IF;

IF ((C='0' AND (STATE1='1')) OR (C='1' AND (STATE3='1'))) THEN next_STATE2<='1'; ELSE next_STATE2<='0'; END IF;

IF ((C='0' AND (STATE2='1')) OR (C='1' AND (STATE4='1'))) THEN next_STATE3<='1'; ELSE next_STATE3<='0'; END IF;

IF ((C='0' AND (STATE3='1')) OR (C='1' AND (STATE6='1'))) THEN next_STATE4<='1'; ELSE next_STATE4<='0'; END IF;

IF ((C='1' AND (STATE0='1')) OR (C='0' AND (STATE4='1'))) THEN next_STATE6<='1'; ELSE next_STATE6<='0'; END IF;

Y<= ((std_logic_vector'(STATE1, STATE1, STATE1)) AND ((std_logic_vector'(C, C, C))) AND (std_logic_vector'("000"))) OR ((std_logic_vector'(STATE6, STATE6, STATE6)) AND ((std_logic_vector'(NOT C , NOT C, NOT C))) AND (std_logic_vector'("000"))) OR ((std_logic_vector'(STATE0, STATE0, STATE0)) AND ((std_logic_vector'(NOT C , NOT C, NOT C))) AND (std_logic_vector'("001"))) OR ((std_logic_vector'(STATE2, STATE2, STATE2)) AND ((std_logic_vector'(C, C, C))) AND (std_logic_vector'("001"))) OR ((std_logic_vector'(STATE1, STATE1, STATE1)) AND ((std_logic_vector'(NOT C, NOT C, NOT C))) AND (std_logic_vector'("010"))) OR ((std_logic_vector'(STATE3, STATE3, STATE3)) AND ((std_logic_vector'(C, C, C))) AND (std_logic_vector'("010"))) OR ((std_logic_vector'(STATE2, STATE2, STATE2)) AND ((std_logic_vector'(NOT C, NOT C, NOT C))) AND (std_logic_vector'("011"))) OR ((std_logic_vector'(STATE4, STATE4, STATE4)) AND ((std_logic_vector'(C, C, C))) AND (std_logic_vector'("011"))) OR ((std_logic_vector'(STATE3, STATE3, STATE3)) AND ((std_logic_vector'(NOT C , NOT C, NOT C))) AND (std_logic_vector'("100"))) OR ((std_logic_vector'(STATE6, STATE6, STATE6)) AND ((std_logic_vector'(C, C, C))) AND (std_logic_vector'("100"))) OR ((std_logic_vector'(STATE0, STATE0, STATE0)) AND ((std_logic_vector'(C, C, C))) AND (std_logic_vector'("110"))) OR ((std_logic_vector'(STATE4, STATE4, STATE4)) AND ((std_logic_vector'(NOT C, NOT C, NOT C))) AND (std_logic_vector'("110")));

next_Y2 <= Y(2); next_Y1 <= Y(1); next_Y0 <= Y(0); END PROCESS;

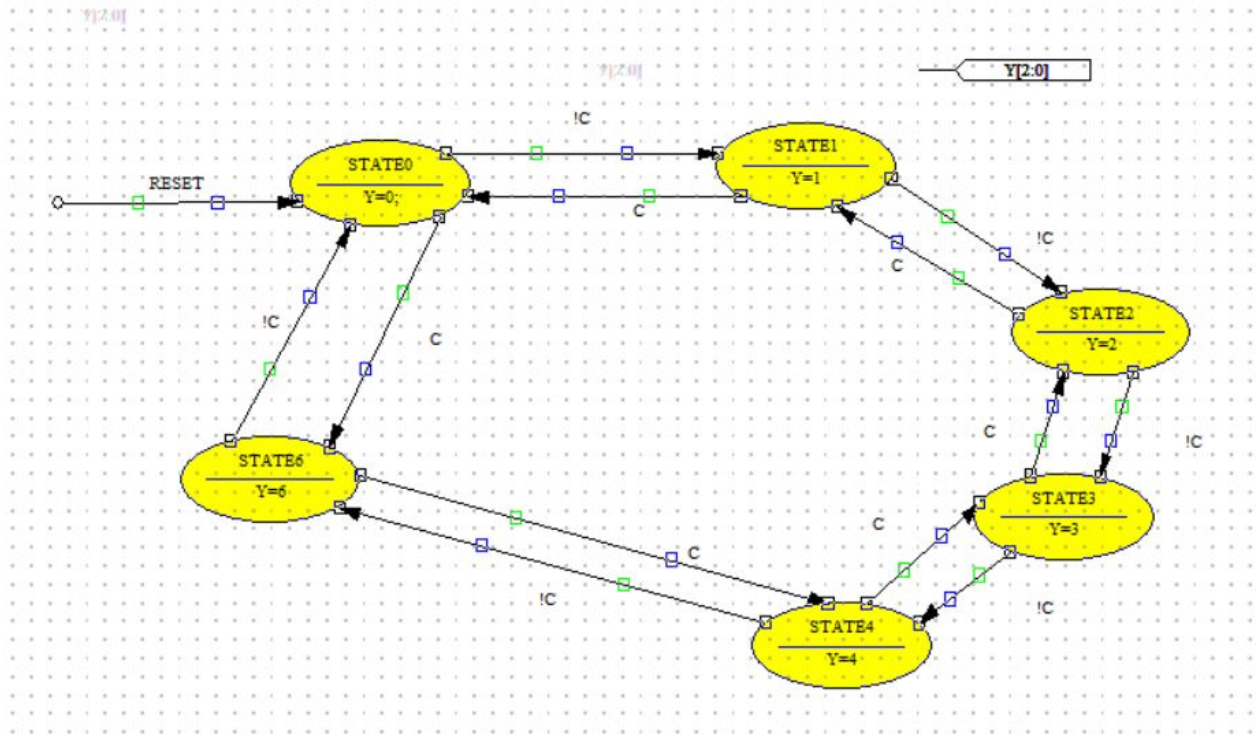
END BEHAVIOR;

LIBRARY ieee; USE ieee.std_logic_1164.all;

ENTITY UPDWCOUN IS PORT (Y : OUT std_logic_vector (2 DOWNTO 0); CLK,C,RESET: IN std_logic); END;

ARCHITECTURE BEHAVIOR OF UPDWCOUN IS COMPONENT SHELL_UPDWCOUN PORT (CLK,C,RESET: IN std_logic; Y0,Y1,Y2 : OUT std_logic); END COMPONENT; BEGIN SHELL1_UPDWCOUN : SHELL_UPDWCOUN PORT MAP (CLK=>CLK,C=>C,RESET=>RESET,Y0=>Y(0),Y1=>Y(1),Y2=>Y(2)); END BEHAVIOR;

Grafo della Macchina a Stati



Test della Macchina a Stati

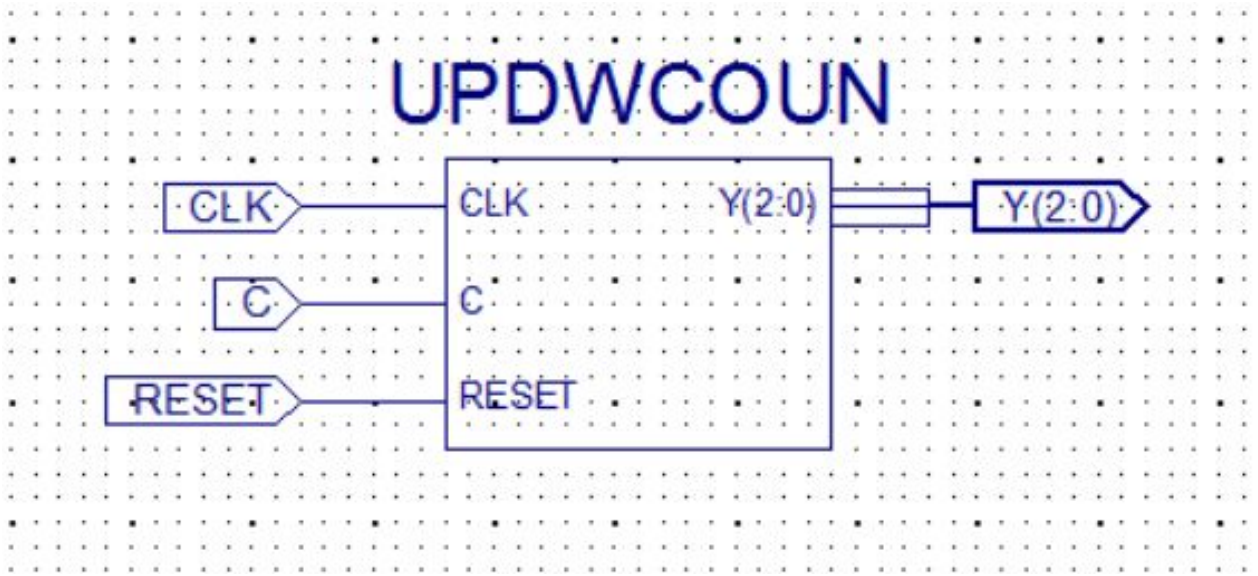
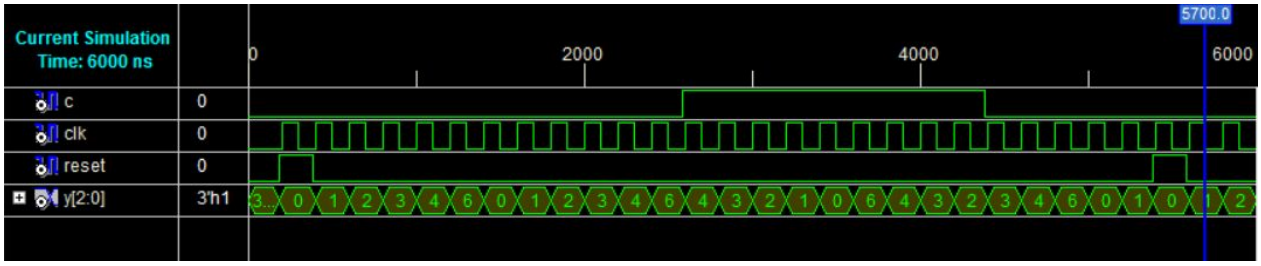


Diagramma Temporale della Macchina a Stati



Appendici

Risorse per il laboratorio

- **Ansoft - Simplorer Student Version**

Software libero scaricabile dal sito della ditta Ansoft [Ansoft http://www.ansoft.com/products/em/simplorer/] per la simulazione di sistemi elettrici, idraulici, magnetici, termici.

Estensione VHDL-AMS contenuta nel programma.

- **Xilinx - Software ISE FREE WEBPACK**

Software per la progettazione di circuiti digitali e per la loro implementazione su schede con chip programmabile (CPLD, FPGA), fornito gratuitamente e scaricabile dal web al sito della Xilinx [Xilinx http://www.xilinx.com/ise/logic_design_prod/webpack.htm]

Creazione di circuiti di tipo Schematico, VHDL, Macchine a Stati Finiti

- **Digilent XCRP - Scheda corredata con CPLD Xilinx XCR64PC44**

Prodotta dalla ditta Digilent, [Digilent http://www.digilentinc.com/products/Programmable.cfm] la XCRP è una scheda per prototipi corredata con chip Xilinx CoolRunner CPLD XCR64PC44 e programmabile tramite cavo seriale su interfaccia JTAG. Si interfaccia egregiamente con il software ISE di cui al punto precedente.

- **GIMP**

Tool per la creazione ed elaborazione professionale di immagini non ha nulla da invidiare a Photoshop, è scaricabile gratuitamente dal sito [Gimp http://www.gimp.org]

- **ArgoUML**

Unified Modeling Language: tool per la creazione di diagrammi secondo lo standard UML. Il tool è scaricabile gratuitamente dal sito [ArgoUml http://argouml.tigris.org]

Da segnalare che in versione Alpha c'è un ottimo plug-in per Netbeans 5 che fornisce l'interfaccia per la realizzazione di diagrammi UML.

Laboratorio di Sistemi

RC passa-basso

Risposta di un sistema costituito da una rete RC passa-basso

Risposta al transitorio

La risposta al transitorio denota il comportamento della rete all'applicazione di uno stimolo a gradino al nostro circuito. Come si vede dal grafico 2D, la tensione sul condensatore si carica al valore della tensione d'ingresso con una costante di tempo $\tau = R * C$. Dopo circa $5 * \tau$ il condensatore si può dire che abbia raggiunto il valore di tensione di ingresso applicata ai suoi capi tramite la resistenza R che ne determina la velocità di carica.

Risposta in frequenza

Per la risposta a regime si osserva il diagramma di **Bode** nel quale appaiono tensione sul condensatore e fase rispetto alla tensione di ingresso. La tensione ci dice che la rete si comporta come un filtro passa-basso con frequenza di taglio di circa 158 Hz. La fase ci dimostra che: mentre a frequenze inferiori alla frequenza di taglio il condensatore è praticamente ininfluente avendo reattanza molto elevata e lo sfasamento tra le due tensioni è zero radianti, all'aumentare della frequenza lo sfasamento aumenta fino ad un massimo di $-\pi/2$ radianti (-90 gradi) e che lo sfasamento alla frequenza di taglio è di $-\pi/4$ radianti (-45 gradi). Questi risultati sono in accordo con l'analisi matematica della rete.

Analisi matematica

Acronimi

- **VHDL**: **V**LSI **H**ardware **D**escription **L**anguage

Linguaggio di programmazione per la definizione di un hardware

- **VLSI**: **V**ery **L**arge **S**cale of **I**ntegration

Un hardware ad elevata scala di integrazione. Si tenga conto che lo stato attuale di questa scala di integrazione e dell'ordine dei 90 nm (nano metri).

- **SOPC**: **S**ystem **O**n **P**rogrammable **C**hip

Sistema implementato su chip programmabile in sede.

- **JTAG**: **J**oint **T**est **A**ction **G**roup

Dall'acronimo del gruppo di aziende riunitesi sotto il nome di **Joint Test Action Group** che hanno collaborato alla sua stesura

- **IEEE**: **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers [1].

Note

[1] http://www.ieee.org/

Autori

Hanno contribuito

- **Docenti**
  - **prof. Roberto Della Grotta (docente c/o IPIA ROSA LUXEMBURG - MILANO)**
- **Allievi**
  - **Giampaolo Roberto** Classe:3C AS:2005/2006
  - **Uga Michael** Classe:3C AS:2005/2006
  - **Lazzarini Luca Danilo** Classe:3C AS:2006/2007
  - **Arosio Luca** Classe:4C AS:2010/2011

Fonti e autori delle voci

Fonte:: http://it.wikibooks.org/w/index.php?oldid=99569 *Autori:*: Pietrodn

Esercitazioni pratiche di elettronica *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=222184 *Autori:*: Diablo, LoStrangolatore, Pietrodn, Ramac, Rdgmus, The Doc, 5 Modifiche anonime

Introduzione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=37339 *Autori:*: Diablo, Pietrodn

Simulazione dei principali circuiti con amplificatore operazionale *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=114610 *Autori:*: Az1568, Pietrodn, Rdgmus, Wim b, 2 Modifiche anonime

Differenza fra segnali analogici e segnali digitali *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=22048 *Autori:*: Pietrodn, 2 Modifiche anonime

Introduzione alle logiche TTL *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=15448 *Autori:*: Pietrodn, Rdgmus

Classi fondamentali dei circuiti digitali *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=185464 *Autori:*: Pietrodn, 3 Modifiche anonime

I sistemi di numerazione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=176042 *Autori:*: Pietrodn, Rdgmus, 4 Modifiche anonime

Conversione da base 2 in base 10 *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=10630 *Autori:*: Pietrodn, Rdgmus

Conversione da base 10 in base 2 *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=174577 *Autori:*: Pietrodn, Rdgmus, 10 Modifiche anonime

Conversione da base 10 in base 16 *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=38818 *Autori:*: Diablo, Pietrodn, Rdgmus, 2 Modifiche anonime

Il ciclo di progetto *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=10633 *Autori:*: Pietrodn, Rdgmus

Software ISE-XILINX Free Webpack *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=10644 *Autori:*: Pietrodn, Rdgmus, The Doc, 4 Modifiche anonime

Finestra principale *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=54668 *Autori:*: Pietrodn, Rdgmus, The Doc

Creazione di un nuovo progetto *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=15463 *Autori:*: Pietrodn, Rdgmus, The Doc

Creazione di uno schema *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=54669 *Autori:*: Pietrodn, Rdgmus, The Doc

Creazione di un file VHDL *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=54520 *Autori:*: Diablo, Rdgmus, 3 Modifiche anonime

Creazione di una Macchina a Stati Finiti *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=176600 *Autori:*: Rdgmus

Il file di piedinatura *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=174473 *Autori:*: Pietrodn, Rdgmus, The Doc, Wim b

La simulazione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=117348 *Autori:*: Pietrodn, Ramac, Rdgmus, The Doc

L'implementazione su scheda e il test *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=22144 *Autori:*: Pietrodn, Rdgmus, The Doc

Logica Combinatoria *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177322 *Autori:*: Diablo, Pietrodn, Rdgmus, The Doc, 1 Modifiche anonime

Implementazione di una funzione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=60411 *Autori:*: Pietrodn, Rdgmus, The Doc, Wim b, 3 Modifiche anonime

Dalla tabella della verità all'implementazione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=77697 *Autori:*: Diablo, Pietrodn, Rdgmus, The Doc

Il multiplexer *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=15451 *Autori:*: Pietrodn, Rdgmus, The Doc

Implementazione di una funzione tramite multiplexer *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=185827 *Autori:*: Franz Liszt, Pietrodn, Ramac, Rdgmus, The Doc

Il Semisommatore o Half-Adder *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177305 *Autori:*: Pietrodn, Rdgmus, The Doc

Il Sommatore completo o Full-Adder *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177304 *Autori:*: Pietrodn, Rdgmus, The Doc

Full-Adder realizzato con mux a 3 ingressi di selezione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=207600 *Autori:*: Rdgmus, 3 Modifiche anonime

Full-Adder realizzato con mux a 2 ingressi di selezione *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=207645 *Autori:*: Rdgmus

Sommare due parole di 4 bit *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177306 *Autori:*: Pietrodn, Rdgmus, The Doc

Sommatore a 4 bit di tipo LOOK AHEAD CARRY *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=180987 *Autori:*: Rdgmus, 1 Modifiche anonime

Il Comparatore *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177483 *Autori:*: Pietrodn, Ramac, Rdgmus, The Doc, 4 Modifiche anonime

Il Comparatore a più bit *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=185560 *Autori:*: Ramac, Rdgmus

Il decodificatore da BCD a sette segmenti *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=209152 *Autori:*: Pietrodn, Rdgmus, The Doc, 5 Modifiche anonime

Il decodificatore da BCD a 7 segmenti in VHDL *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177300 *Autori:*: Diablo, Rdgmus, 2 Modifiche anonime

Multiplexing di più display a 7 segmenti *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=115965 *Autori:*: Diablo, Rdgmus, The Doc, 2 Modifiche anonime

Logica Sequenziale *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=219437 *Autori:*: Diablo, Rdgmus, The Doc, Wim b, 10 Modifiche anonime

Flip Flop di tipo D *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=178583 *Autori:*: Enzo D'Ambrosio, Rdgmus, Wim b, Wutsje, 2 Modifiche anonime

Flip Flop di tipo JK *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=175068 *Autori:*: Ramac, Rdgmus, 1 Modifiche anonime

Flip Flop di tipo T *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=175069 *Autori:*: Ramac, Rdgmus, 1 Modifiche anonime

Contatore e-o divisore modulo 2 *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177309 *Autori:*: Enzo D'Ambrosio, Rdgmus

Contatori in cascata *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177307 *Autori:*: Enzo D'Ambrosio, Rdgmus, 1 Modifiche anonime

Macchine a Stati Finiti *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=180993 *Autori:*: Rdgmus

Risorse per il laboratorio *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=54528 *Autori:*: Pietrodn, Rdgmus, 1 Modifiche anonime

Laboratorio di Sistemi *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=177314 *Autori:*: Pietrodn, Rdgmus

Acronimi *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=83059 *Autori:*: Diablo, Pietrodn, Rdgmus

Autori *Fonte:*: http://it.wikibooks.org/w/index.php?oldid=207641 *Autori:*: Diablo, Pietrodn, Rdgmus, 5 Modifiche anonime

Fonti, licenze e autori delle immagini

File:0-res---messure.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:0-res---messure.jpg *Licenza*: Public Domain *Autori:*: Original uploader was Agapetos at en.wikipedia

File:Makiguchi.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Makiguchi.jpg *Licenza*: Public Domain *Autori:*: Butsushin, Giwa, Reggaeman, Zscout370

Image:Rosa Luxemburg.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Rosa_Luxemburg.jpg *Licenza*: Public Domain *Autori:*: -jha-, Black31, Cantons-de-l'Est, Krinkle, Mogelzahn, 2 Modifiche anonime

Immagine:UCF_SINTAX.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:UCF_SINTAX.gif *Licenza*: sconosciuto *Autori:*: Rdgmus, Wim b

Image:impact4.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Impact4.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Image:Program.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Program.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:LC1 3.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LC1_3.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:LC1 4.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LC1_4.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:LC1 5.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LC1_5.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:LC1 6.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LC1_6.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:LC1 7.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LC1_7.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:LC1 ucf.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LC1_ucf.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:Lc1 msim.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Lc1_msim.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:Lc2_2.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Lc2_2.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:Lc2_1.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Lc2_1.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:Lc2_4.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Lc2_4.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:mux2a1_1.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Mux2a1_1.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:mux2a1_2.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Mux2a1_2.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:mux2a1_3.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Mux2a1_3.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:Mux16a1_1.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Mux16a1_1.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:Mux16a1_7.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Mux16a1_7.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:xor_1.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Xor_1.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

Immagine:xor_2.jpg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Xor_2.jpg *Licenza*: GNU Free Documentation License *Autori:*: Rdgmus, The Doc

File:Sum MUX2s.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Sum_MUX2s.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:CyOut MUX2s.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:CyOut_MUX2s.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:Full Adder MUX2s.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Full_Adder_MUX2s.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:Full Adder MUX2s sim.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Full_Adder_MUX2s_sim.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:Carry Generate.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Carry_Generate.JPG *Licenza*: Creative Commons Attribution-Share Alike *Autori:*: Roberto Della Grotta

File:Carry Propagate.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Carry_Propagate.JPG *Licenza*: Creative Commons Attribution 3.0 *Autori:*: Rdgmus

File:RipGen.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:RipGen.JPG *Licenza*: Creative Commons Attribution-Share Alike *Autori:*: Roberto Della Grotta

File:CarryOut.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:CarryOut.JPG *Licenza*: Creative Commons Attribution 3.0 *Autori:*: Rdgmus

File:CarryGenProp4Bit.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:CarryGenProp4Bit.JPG *Licenza*: Creative Commons Attribution-Share Alike *Autori:*: Roberto Della Grotta

File:HalfAdderNoCarryOut.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:HalfAdderNoCarryOut.JPG *Licenza*: Creative Commons Attribution-Share Alike *Autori:*: Roberto Della Grotta

File:FullAdderNoCarryOut.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:FullAdderNoCarryOut.JPG *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:Sum4BitLAC.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Sum4BitLAC.JPG *Licenza*: Creative Commons Attribution-Share Alike *Autori:*: rdgmus

File:Simul4BitLAC.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Simul4BitLAC.JPG *Licenza*: Creative Commons Attribution 3.0 *Autori:*: Rdgmus

File:Comp_1Bit01.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Comp_1Bit01.JPG *Licenza*: GNU General Public License *Autori:*: IPIA ROSA LUXEMBURG

File:Comp_1Bit02.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Comp_1Bit02.JPG *Licenza*: GNU General Public License *Autori:*: Roberto Della Grotta/IPIA ROSA LUXEMBURG

File:Diode symbol.svg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Diode_symbol.svg *Licenza*: GNU Free Documentation License *Autori:*: Created by User:Omegatron

File:POLARIZ DIODE.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:POLARIZ_DIODE.gif *Licenza*: Public Domain *Autori:*: Roberto Della Grotta

File:LED symbol.svg *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:LED_symbol.svg *Licenza*: GNU Free Documentation License *Autori:*: Er Komandante, Jed, Omegatron, Papa November, Rocket000, Sergey kudryavtsev, 11 Modifiche anonime

File:POLARIZ LED DIODE.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:POLARIZ_LED_DIODE.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:CK DISPLAY.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:CK_DISPLAY.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:CA DISPLAY.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:CA_DISPLAY.gif *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:FJK.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:FJK.gif *Licenza*: sconosciuto *Autori:*: Rdgmus, Wim b

File:FT.gif *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:FT.gif *Licenza*: sconosciuto *Autori:*: Rdgmus, Wim b

File:Es1 Sistemi01.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Es1_Sistemi01.JPG *Licenza*: GNU General Public License *Autori:*: IPIA ROSA LUXEMBURG

File:Es1 Sistemi02.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Es1_Sistemi02.JPG *Licenza*: GNU General Public License *Autori:*: IPIA ROSA LUXEMBURG

File:Es1 Sistemi04.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Es1_Sistemi04.JPG *Licenza*: GNU General Public License *Autori:*: IPIA ROSA LUXEMBURG

File:Es1 Sistemi03.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:Es1_Sistemi03.JPG *Licenza*: GNU General Public License *Autori:*: IPIA ROSA LUXEMBURG

File:UPDWCOUN DIA.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:UPDWCOUN_DIA.JPG *Licenza*: Creative Commons Attribution 3.0 *Autori:*: Rdgmus

File:UPDWCOUN testCir.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:UPDWCOUN_testCir.JPG *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

File:UPDWCOUN SYM.JPG *Fonte:*: http://it.wikibooks.org/w/index.php?title=File:UPDWCOUN_SYM.JPG *Licenza*: Creative Commons Attribution-Sharealike 3.0 *Autori:*: Rdgmus

Licenza