

Corso di laurea in Ingegneria delle Telecomunicazioni

Misure Elettroniche per le Telecomunicazioni II

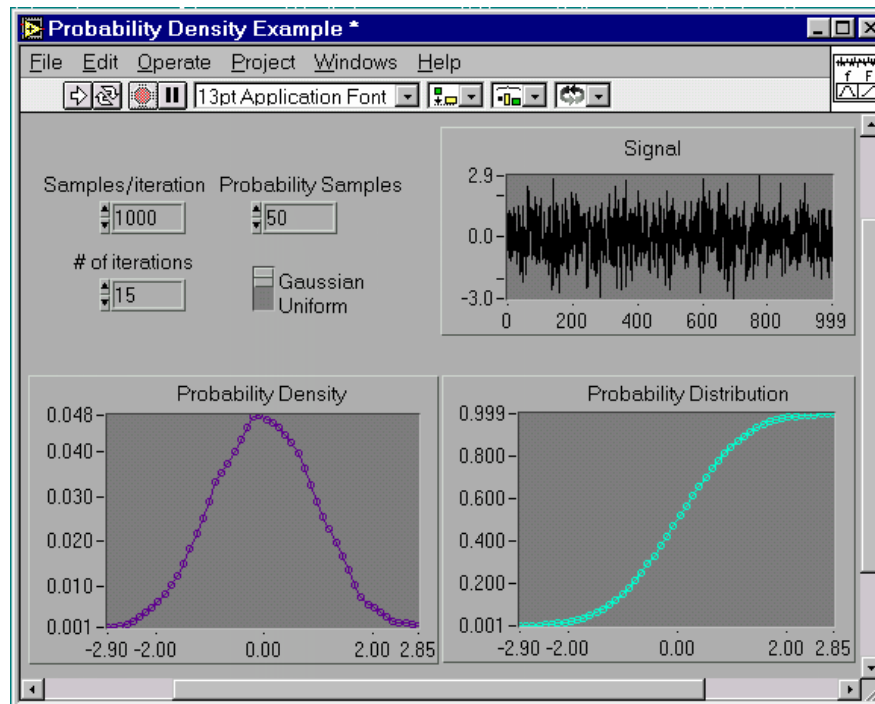
LabView

Gianfranco Miele
g.miele@unicas.it

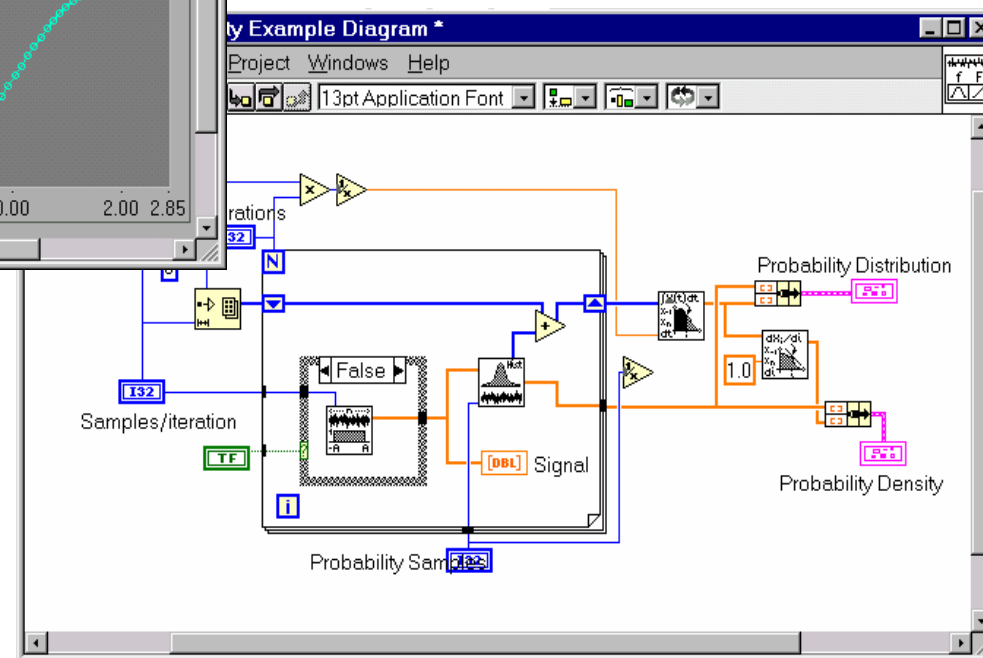
- **Ambiente di sviluppo grafico:**
 - Codice
 - Interfaccia utente
- **Modulare**
- **Compilatore 32 bit**
- **Elevato numero di funzioni**
- **Librerie per la gestione della strumentazione**
- **Librerie per l'acquisizione dati**
- **Possibilità di includere *routine* in altro linguaggio**

- Calcolo, confronto, cicli di controllo
- Operazioni su aggregati di dati (stringhe, array, ...)
- I/O su file
- Gestione strumentazione (RS232, IEEE-488, VXI, ...)
- Acquisizione dati
- Analisi (Generazione ed elaborazione segnali, Statistica, Fitting, Algebra lineare, ...)
- Comunicazione (DDE, TCP/IP, Bluetooth, ActiveX)
- Gestione applicazione (DLL, Timers, Sincronizzazione)

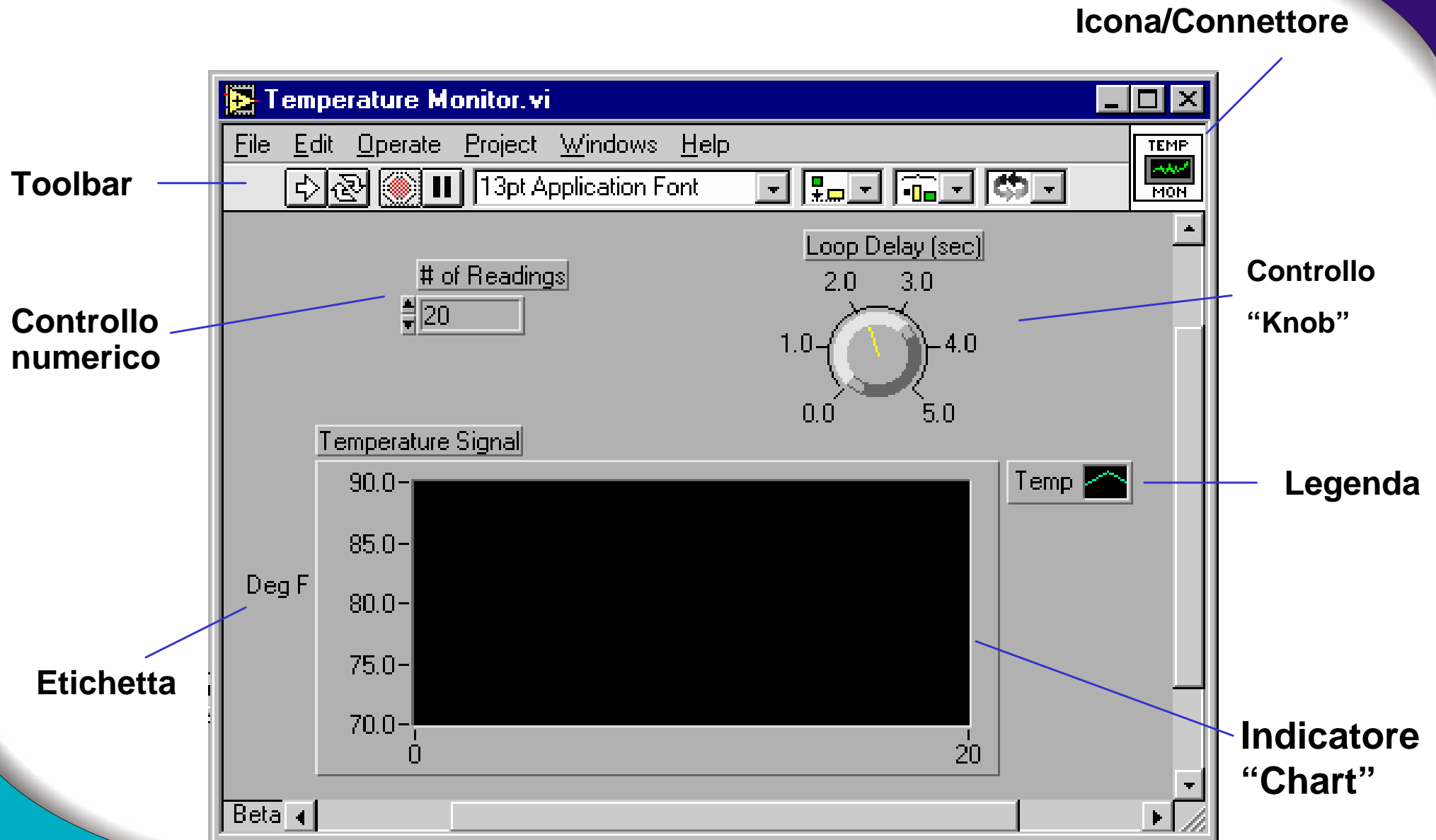
Virtual Instrument (VI)



- Il Front Panel
- Il Block Diagram
- L'icona/connettore

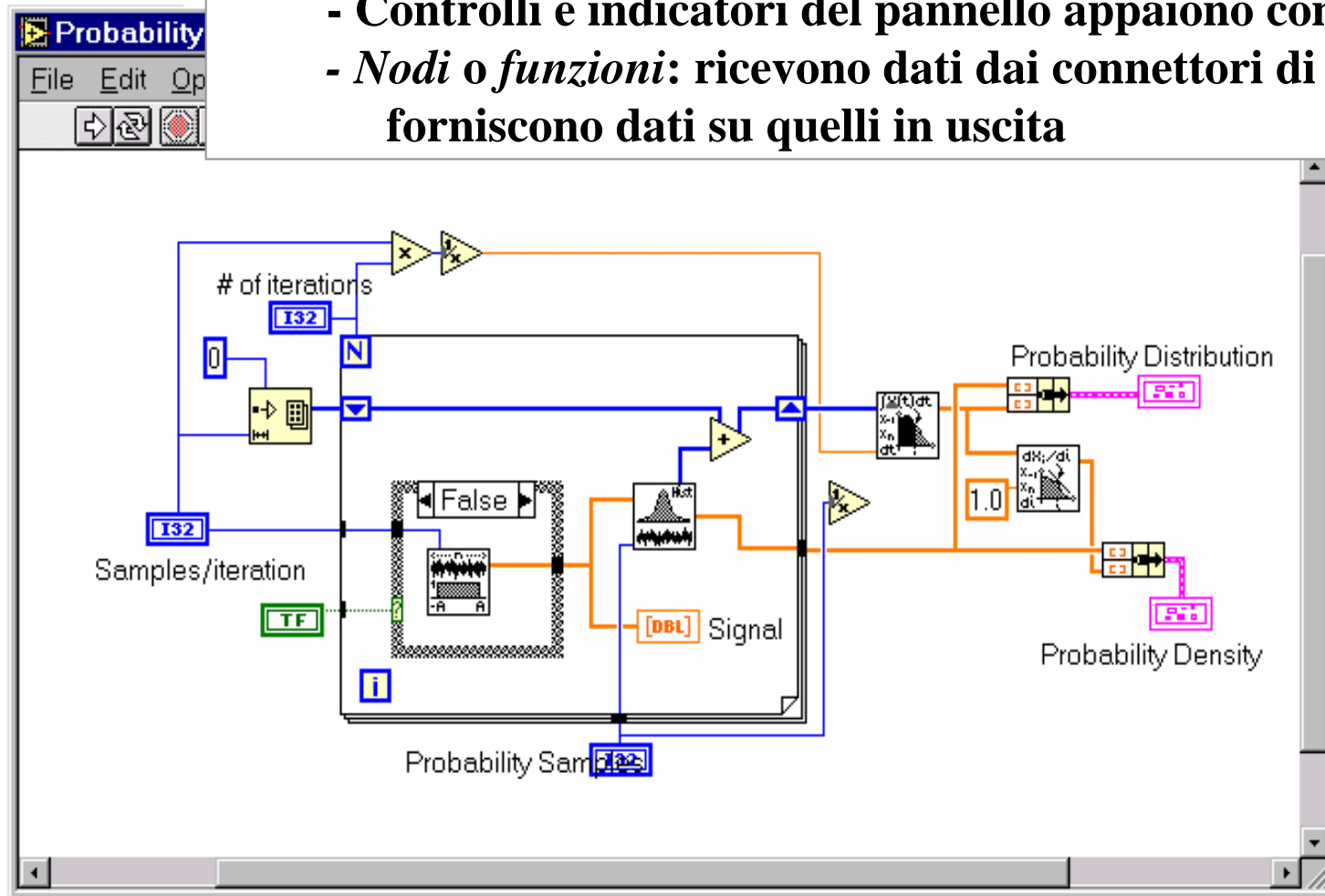


Front Panel



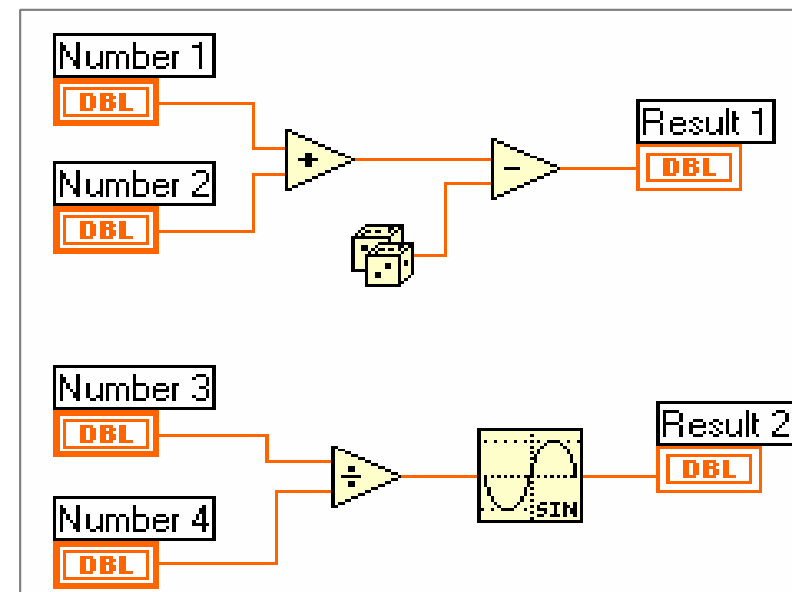
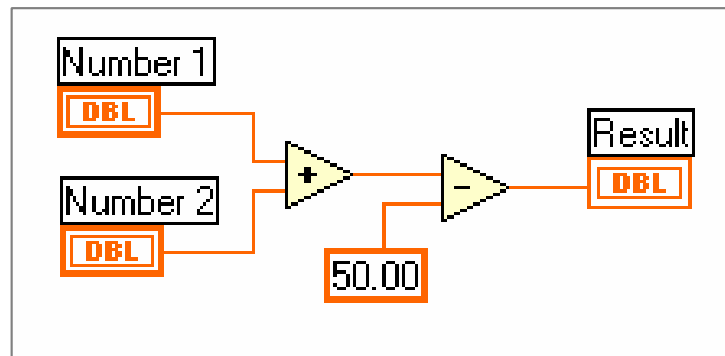
Block Diagram

- E' il codice sorgente dell'applicazione (in *Linguaggio G*)
- Controlli e indicatori del pannello appaiono come *Terminali*
- *Nodi o funzioni*: ricevono dati dai connettori di ingresso e forniscono dati su quelli in uscita



Dataflow programming

- Un nodo è eseguito quando i dati sono disponibili a TUTTI i suoi terminali di ingresso
- Un nodo fornisce i dati ai suoi terminali di uscita quando è terminata la sua esecuzione
- Se non esiste dipendenza tra due nodi, l'ordine con cui verranno eseguiti non è prevedibile !















Il modello di programmazione di LabView è

- *Data driven*

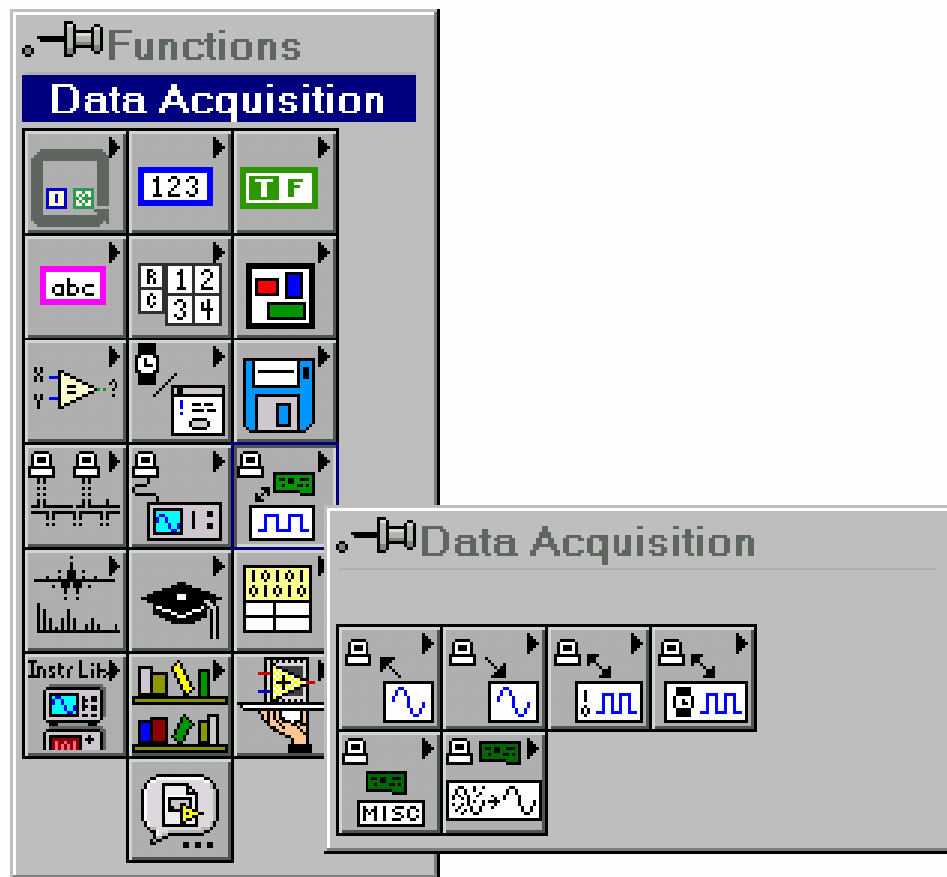
Dataflow programming

- Un filo collega un'unica sorgente di dati con uno o più nodi
- Il colore e lo spessore del filo individuano il tipo di dato

	Scalar	Array 1D	Array 2D	
Numeri				Arancione (floating point)
				Blu (interi)
Booleani				Verde
Stringhe				Viola

L'ambiente di sviluppo: Le palette

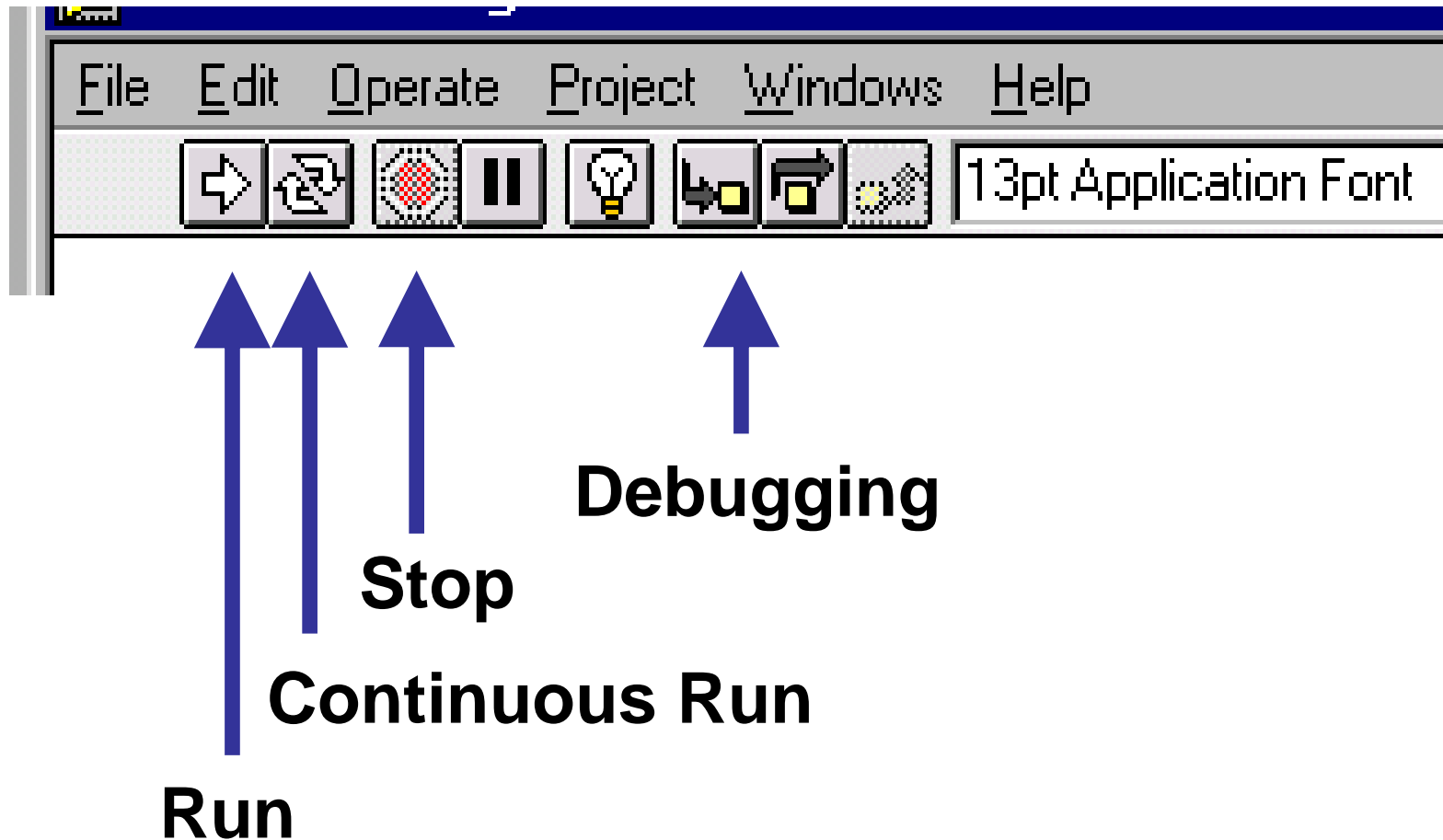
Palette delle Funzioni (Diagramma)



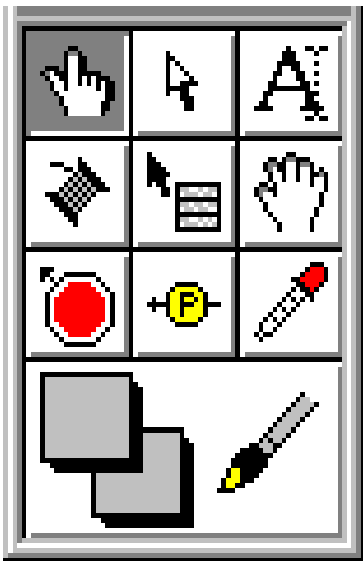
Palette dei Controlli (Pannello)



L'ambiente di sviluppo: La toolbar



L'ambiente di sviluppo: i tools



Operating: Inserimento dati

Positioning: Posizionamento e ridimensionamento

Labeling: Gestione etichette

Wiring: Collegamento (nel Block Diagram)

Object pop-up: Menu dell'oggetto

Scrolling: Scorrimento della finestra

Breakpoint: Inserimento stop (debug)

Probe: Visualizzazione dati nei fili (debug)

Color copy: Copia di un colore

Coloring: Per colorare un oggetto

Tecniche di debug



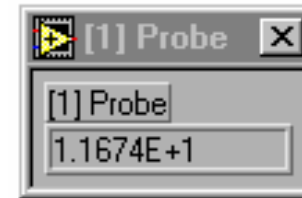
Click sulla freccia spezzata: Info sull'errore



Highlighting button: Esecuzione animata



Probe: Mostra il valore dei dati mentre scorrono lungo i fili



Breakpoint: ferma l'esecuzione quando i dati sono disponibili sul filo



Single Step: Esecuzione passo-passo

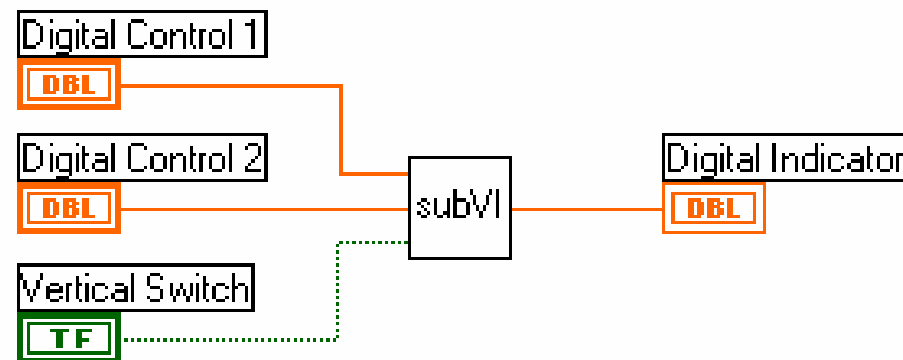


Step Over: Esecuzione passo-passo, non entra nei nodi



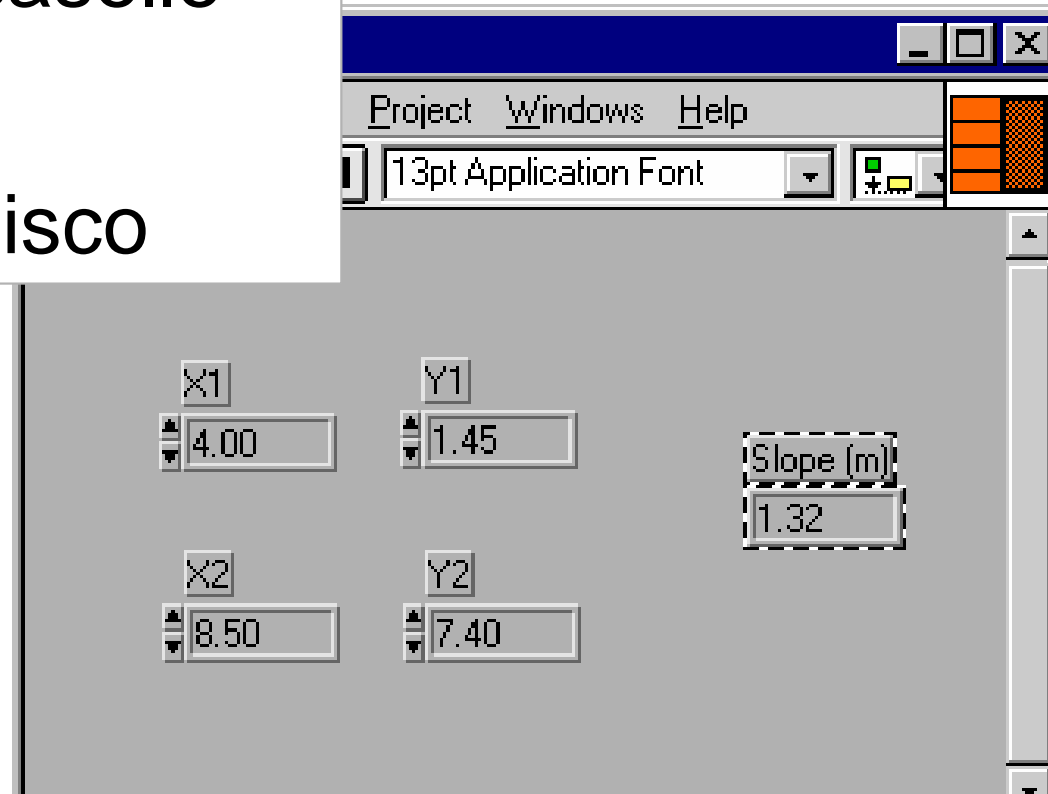
Step Out: Esecuzione passo-passo, esce dai nodi nei quali è entrato

- Un VI può essere usato nel Block Diagram di un altro VI di livello più alto
- Nel VI chiamante, un SubVI è rappresentato con la sua icona

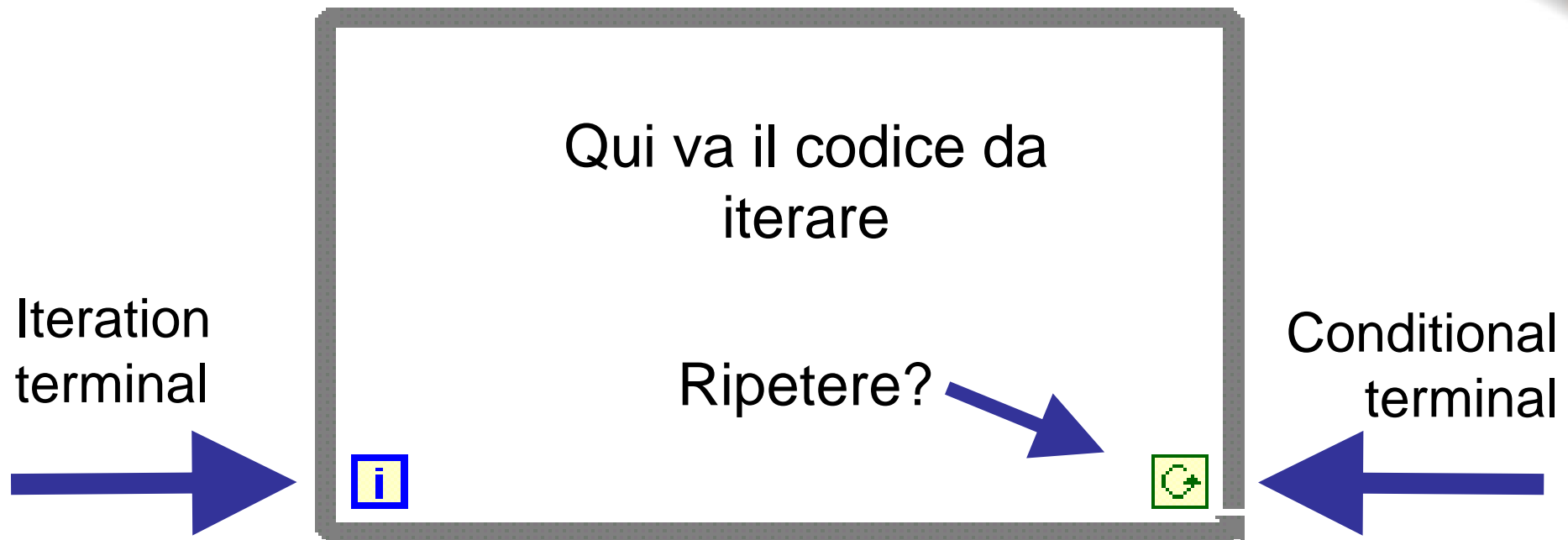


Realizzazione di un SubVI

- Disegno dell'icona
- Associazione di controlli e indicatori alle caselle del connettore
- Salvataggio su disco



Ciclo While

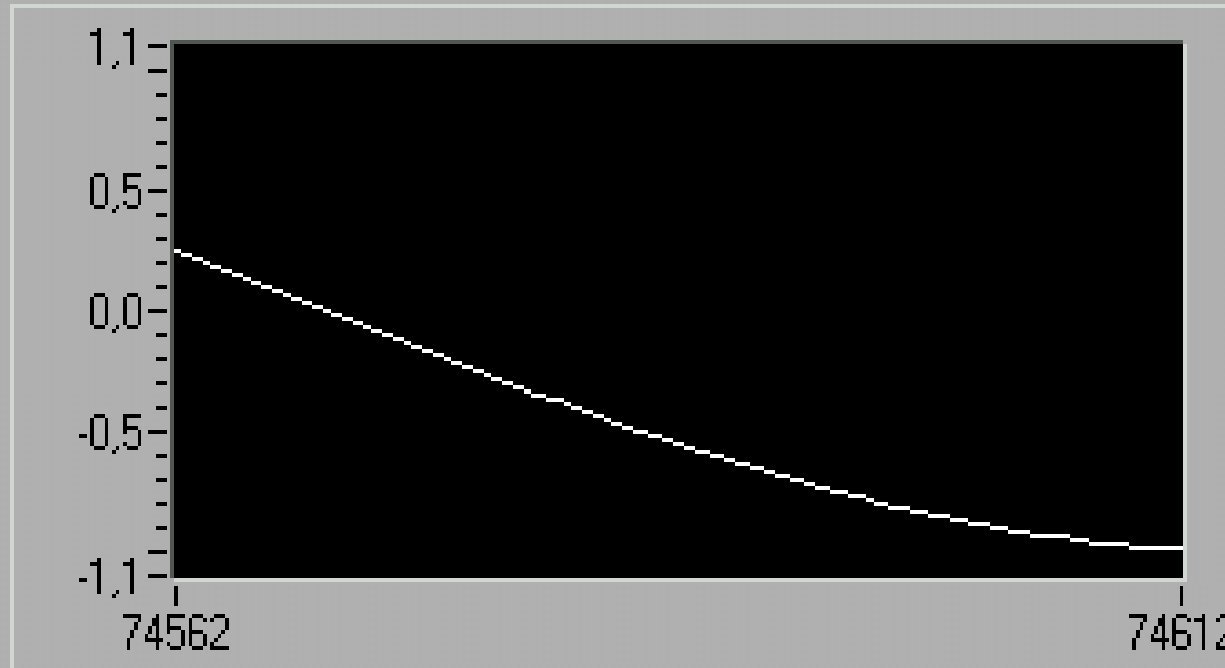


- Il ciclo termina quando si fornisce FALSE al conditional terminal
- Il codice è eseguito almeno una volta
- L'iteration terminal restituisce il numero di iterazioni raggiunto

Waveform Chart

Label

Waveform chart



Plot 0

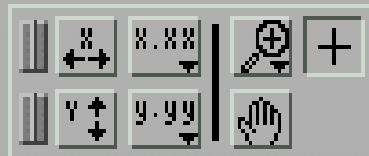


Legenda

Aggiornamento:

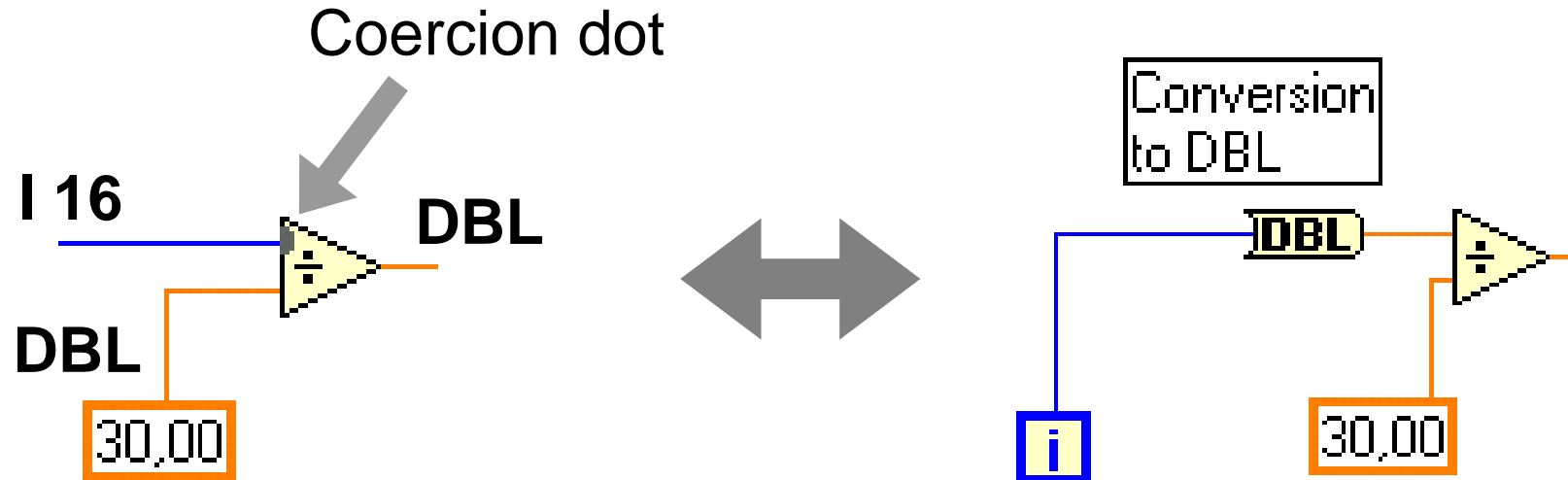
- Strip chart
- Scope chart
- Sweep chart

Palette



Tipi numerici e conversioni

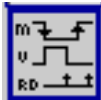
- Interi con segno: I8, I16, I32
- Interi senza segno: U8, U16, U32
- In virgola mobile: SGL (32), DBL (64), EXT
- Complessi



Comportamento dei controlli booleani

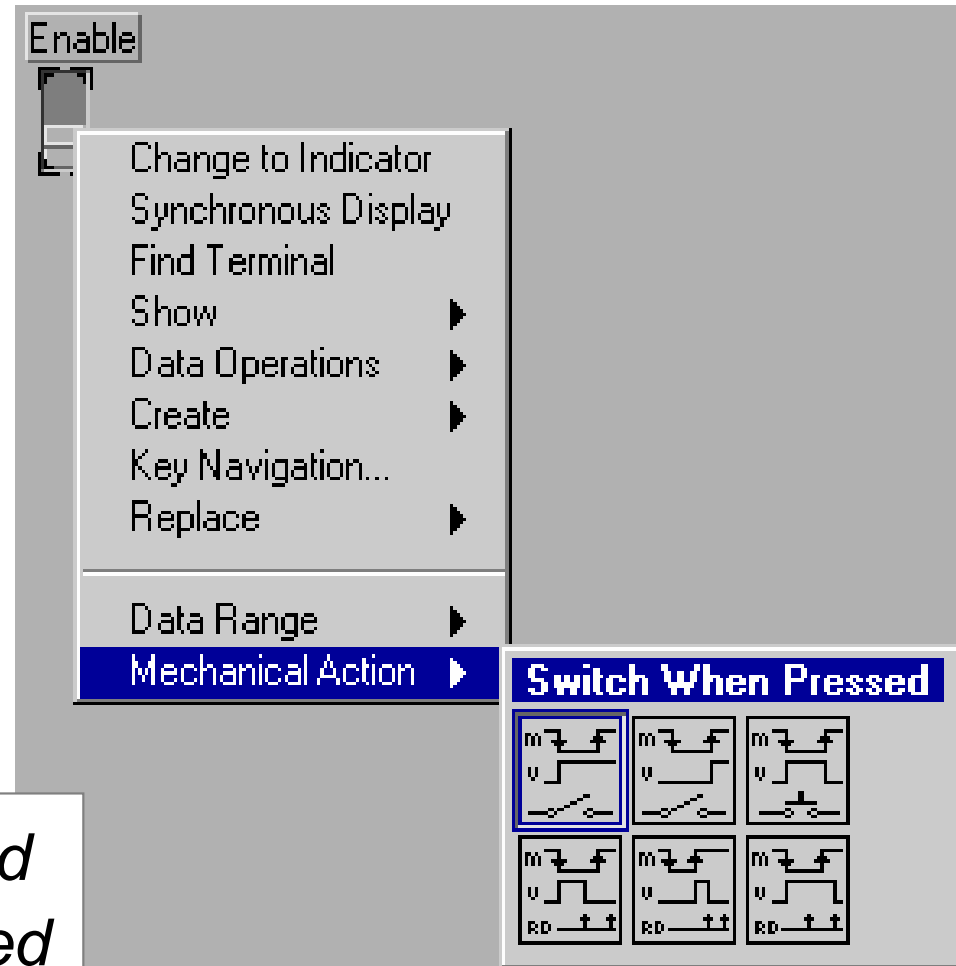


Switch - Commuta
per pressione e
rilascio

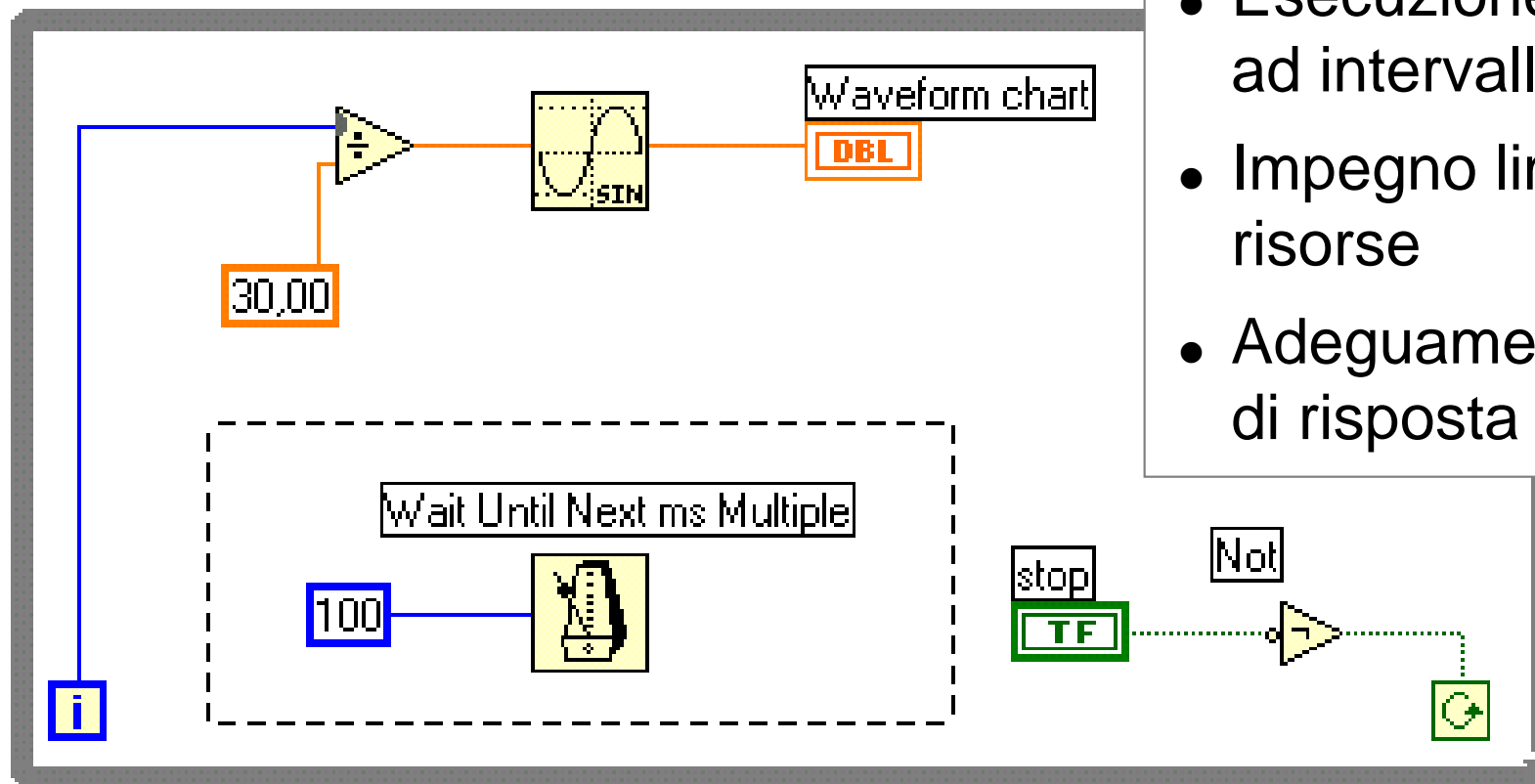


Latch – Ritorna
nello stato di
default quando
viene letto dal
diagramma

- *When pressed*
- *When released*
- *Until released*

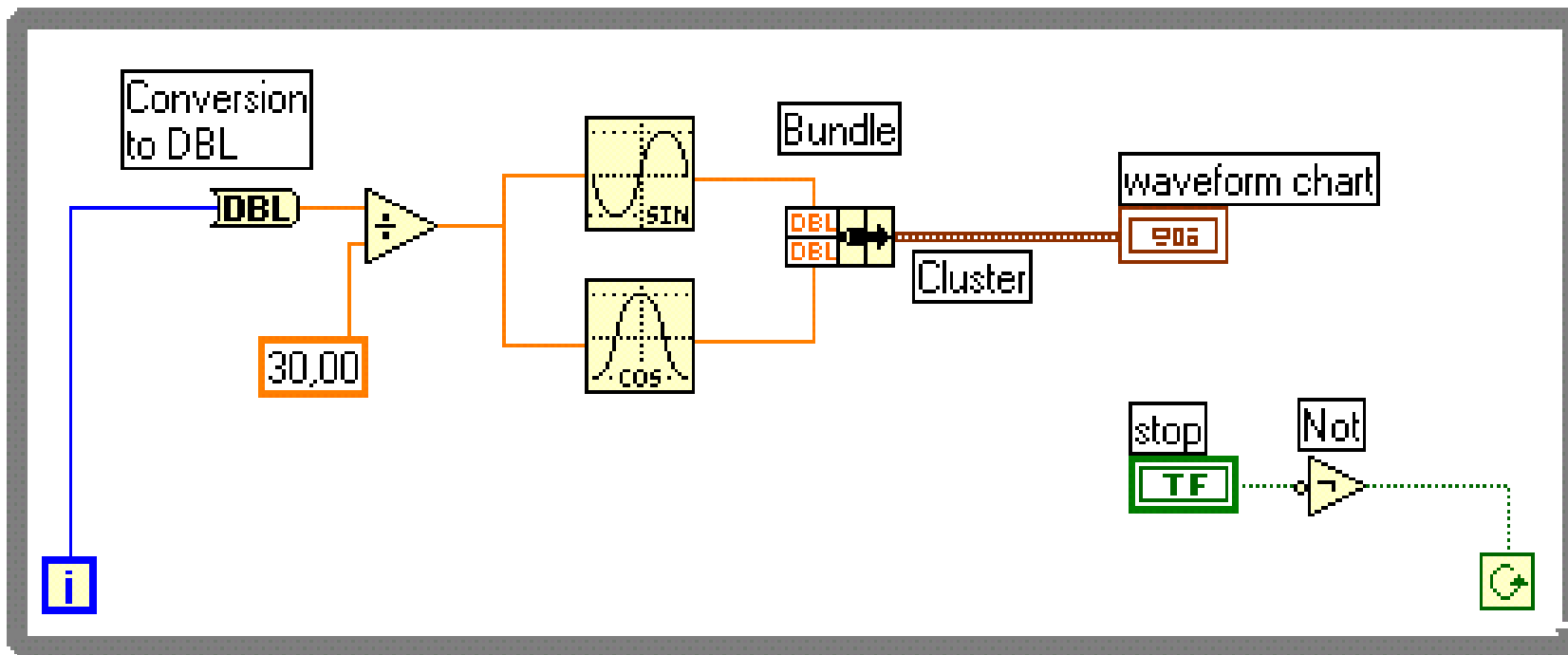


Ciclo While temporizzato



- Esecuzione di azioni ad intervalli prestabiliti
- Impegno limitato di risorse
- Adeguamento ai tempi di risposta dell'utente

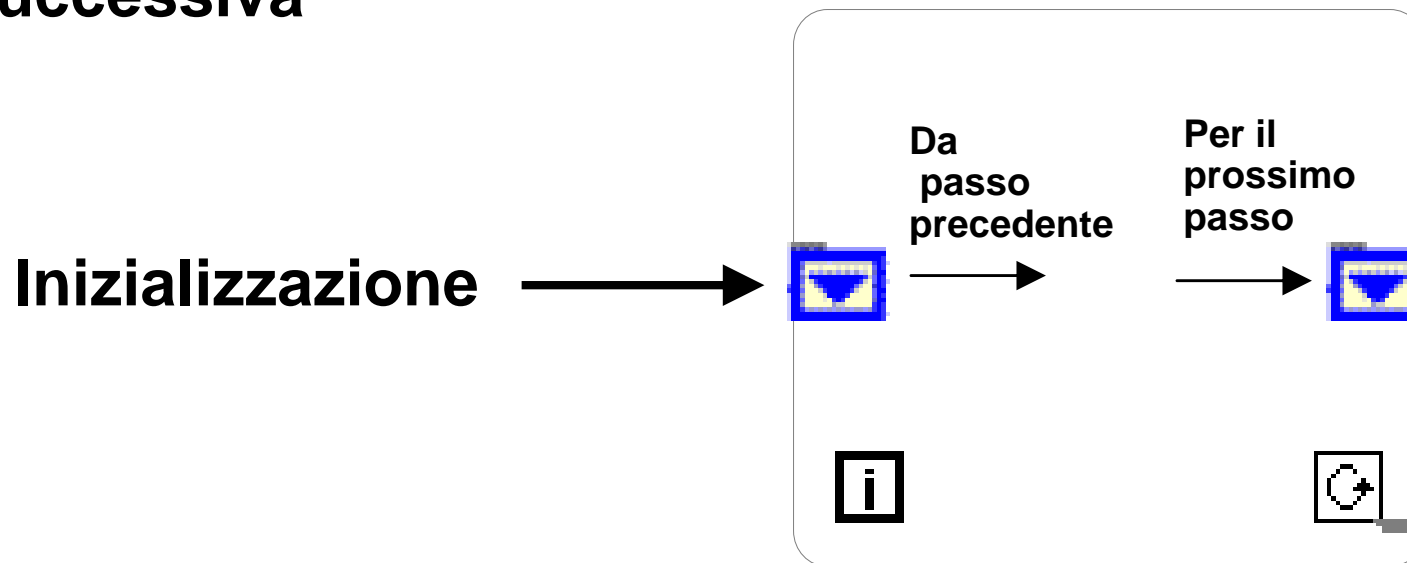
Grafici multiple-plot



- **Bundle** (nella palette Functions >> Cluster): assembla i suoi ingressi in un unico *cluster*
- Un **cluster** è una struttura che raccoglie più elementi in un nuovo tipo di dato

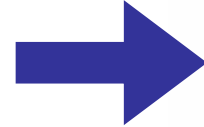
Shift register

- Trasferiscono dati da una iterazione alla successiva
- Si inseriscono con un *right-click* sul bordo del *loop*
- Il terminale destro conserva il dato alla fine della iterazione
- Il terminale sinistro fornisce il dato alla iterazione successiva



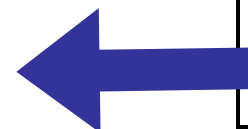
Ciclo for

Numero di iterazioni
da eseguire



N

Numero corrente di
iterazioni



i

Esegue il codice al suo interno un numero
prestabilito di volte

Array

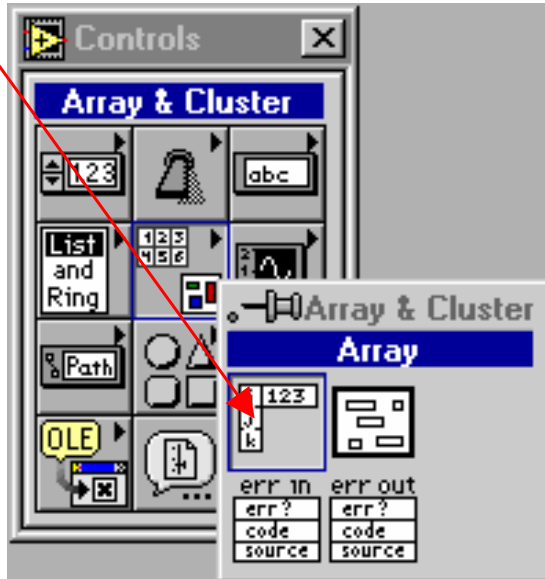
- Collezioni di elementi (dati) dello stesso tipo
- Una o più dimensioni, fino a 2^{31} elementi per dimensione
- Si accede agli elementi con un indice per dimensione
- Il primo elemento ha sempre indice 0

Indice	0	1	2	3	4	5	6	7	8	9
array di 10 elementi	1.2	3.2	8.2	8.0	4.8	5.1	6.0	1.0	2.5	1.7

	0	1	2	3	4	5	6
Array bidimensionale	0						
	1						
	2						
	3						
	4						

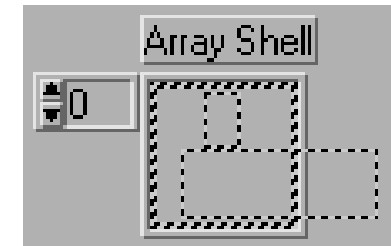
Controlli ed indicatori per gli array

1. Selezionare **Array** dalla palette dei controlli



Procedura analoga per le costanti array

2. Inserire un controllo del tipo desiderato nella **Array Shell**



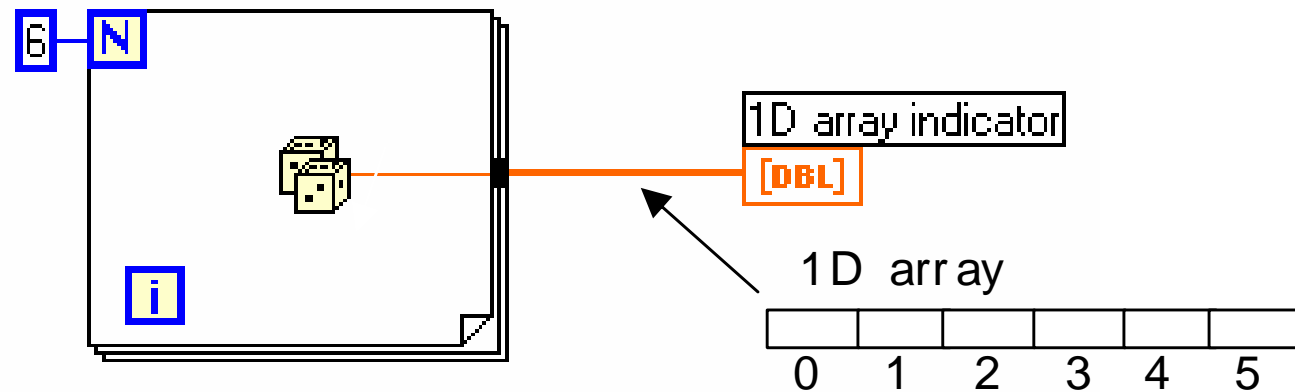
Add Dimension
per 2D arrays



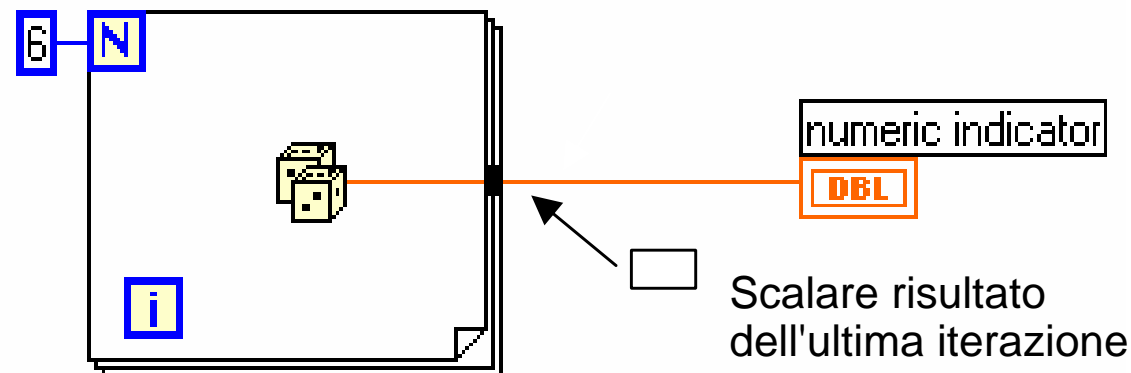
Autoindexing

Il loop accumula l'array al suo margine

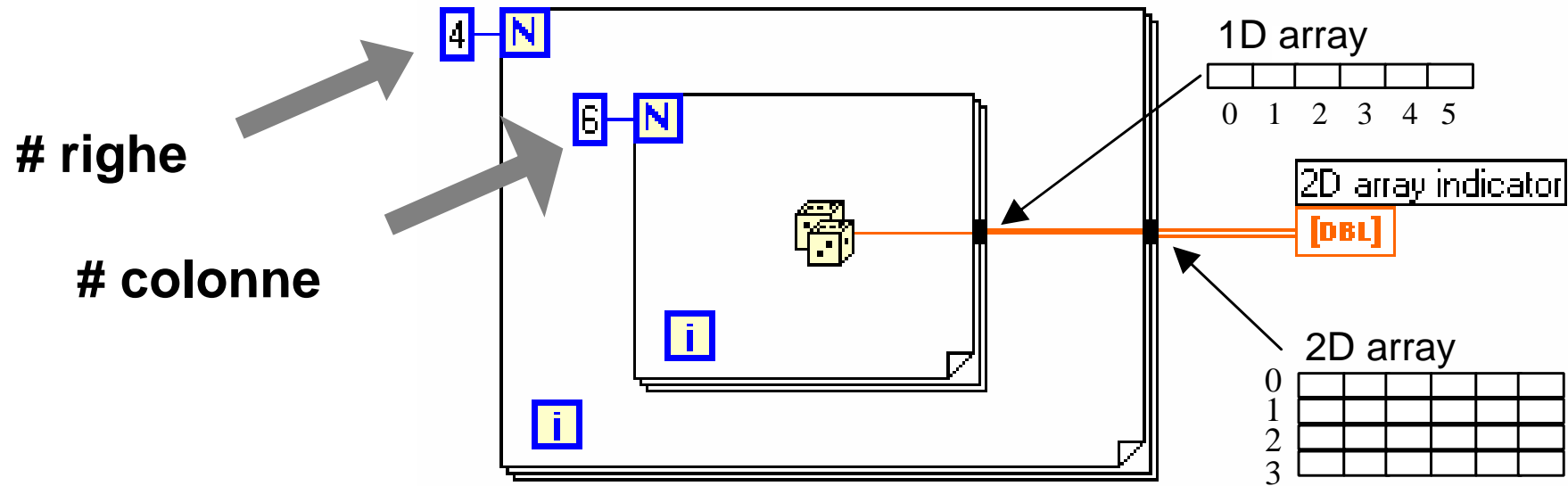
Auto-Indexing
abilitato
(Default nel For)



Auto-Indexing
disabilitato
(Default nel While)



Creazione array 2D

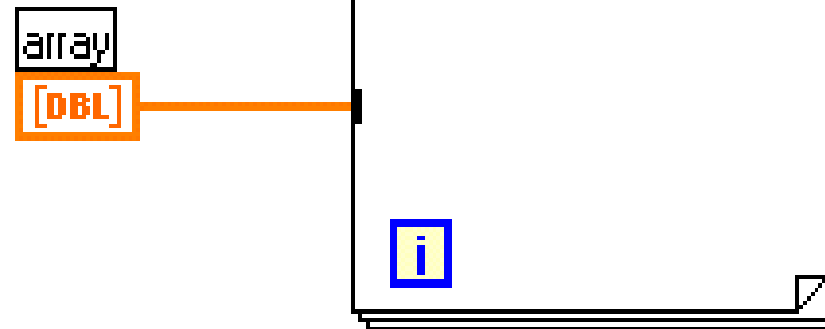
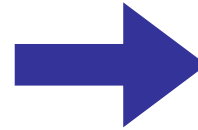


Il loop interno crea gli elementi di ogni riga

Il loop esterno sovrappone le righe

Autoindexing e lettura di array

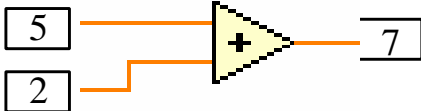
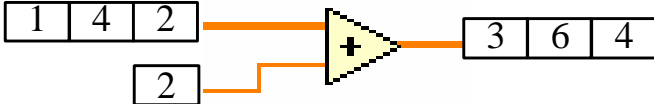
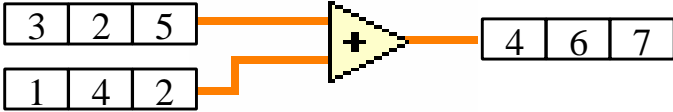
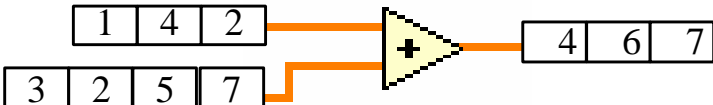
Non deve essere impostato
direttamente



Il ciclo viene eseguito per un numero di volte
pari al numero di elementi dell'array collegato

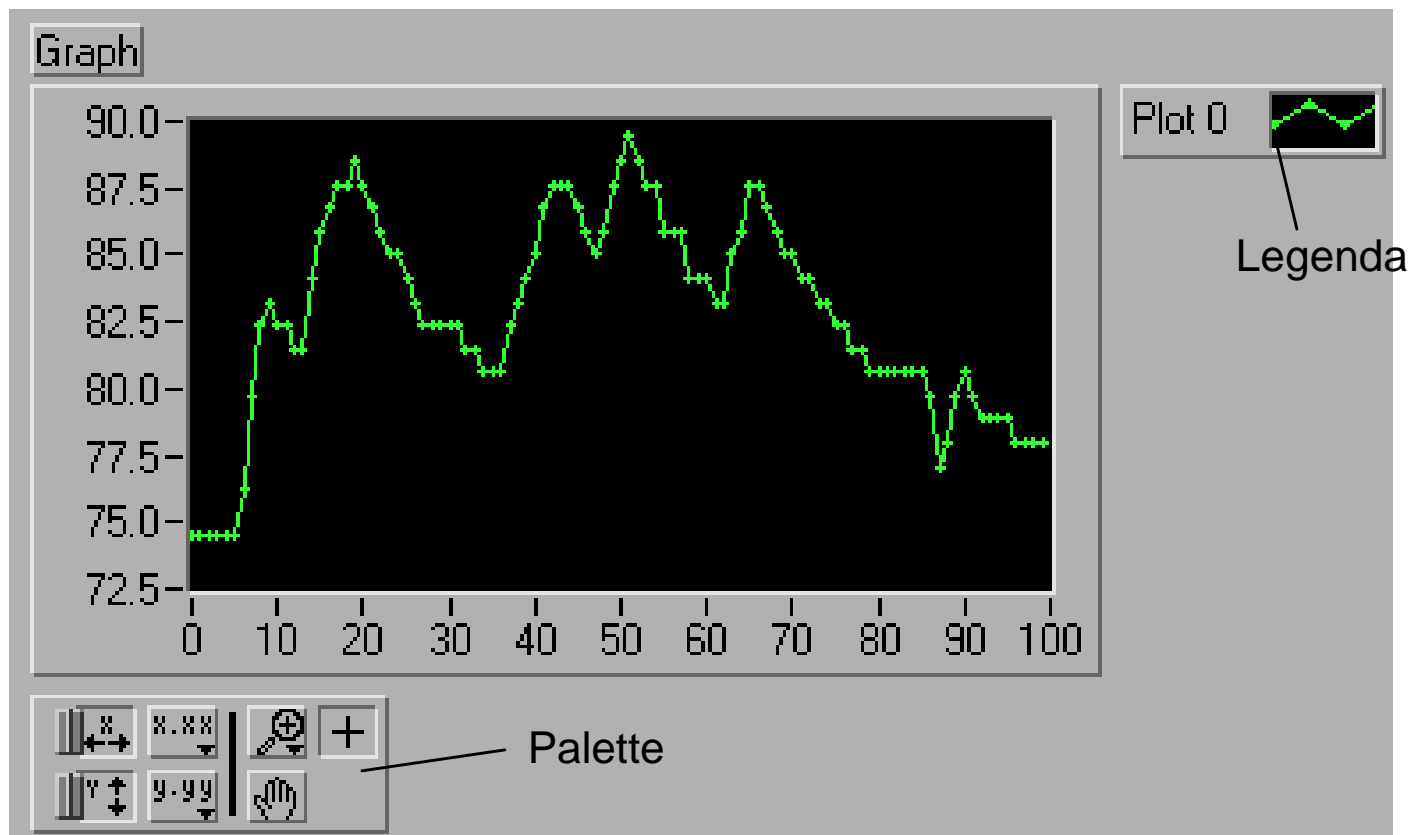
Polimorfismo

- Ingressi delle funzioni di tipo diverso
- Comportamento dipendente dal tipo degli ingressi
- Tutte le funzioni aritmetiche sono polimorfiche

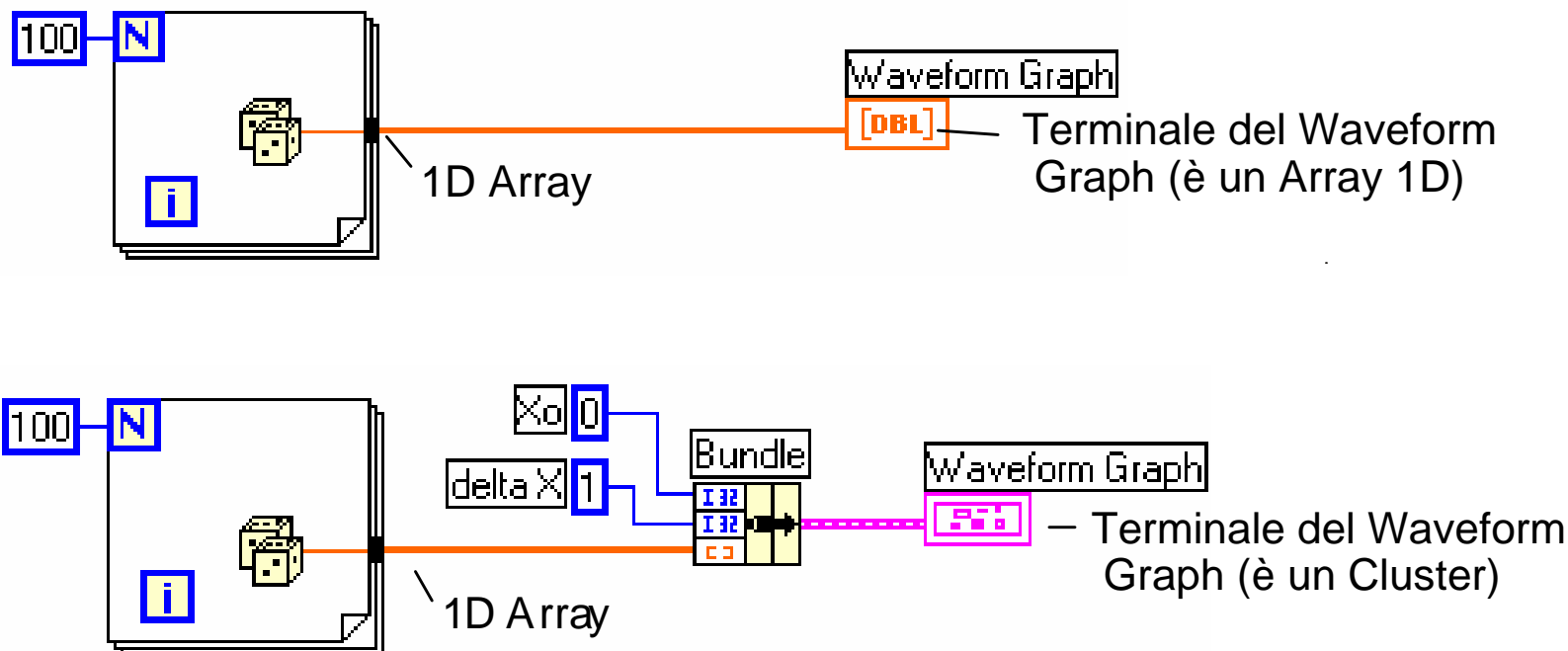
<u>Combinazione</u>		<u>Risultato</u>
Scalare + Scalare		Scalare
Array + Scalare		Array
Array + Array		Array
Array + Array		Array

Waveform graph

Tracciano un array di valori numerici rispetto all'indice

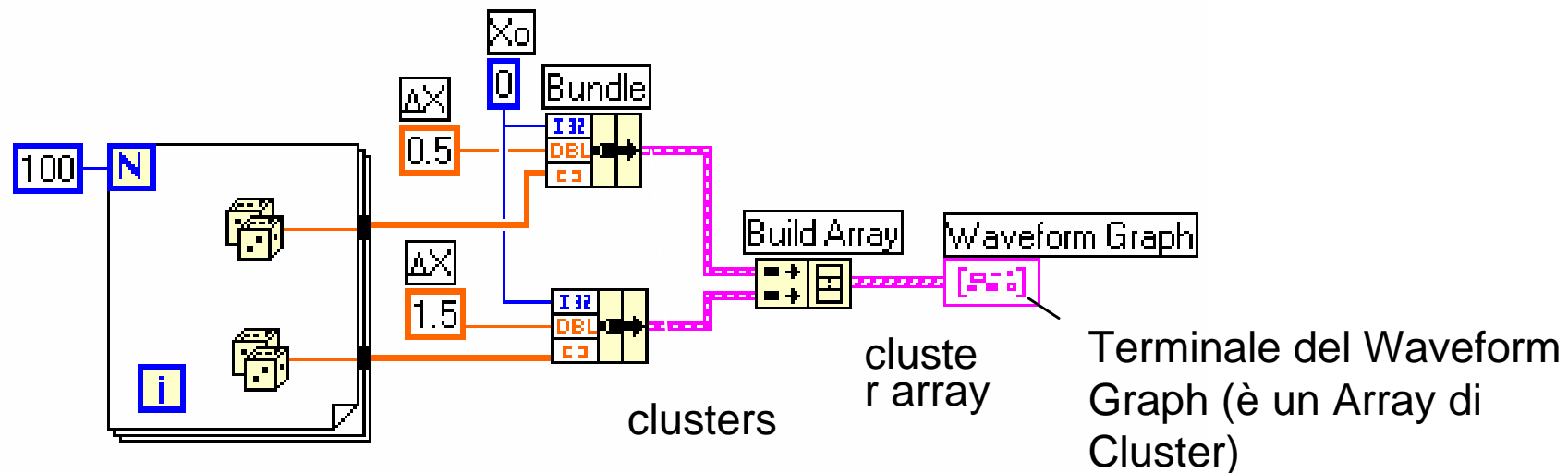
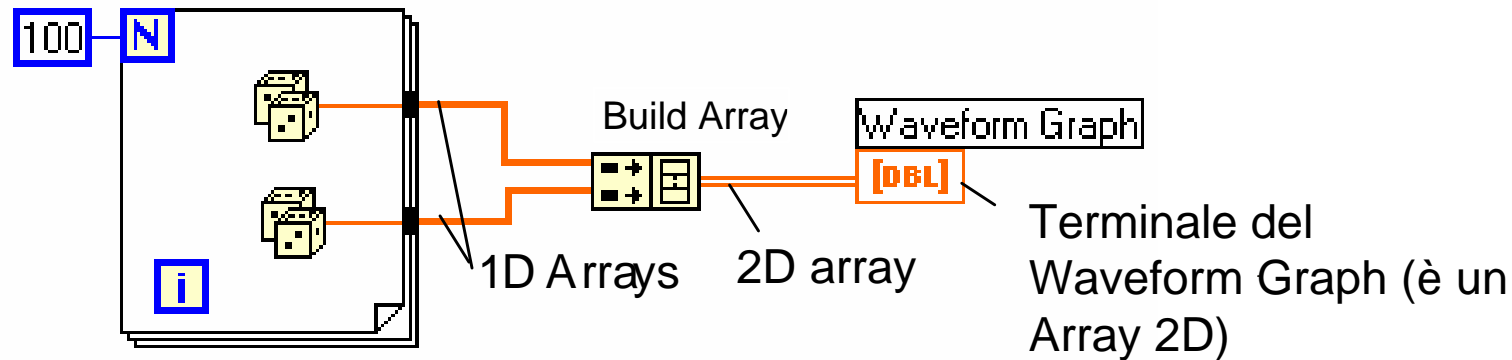


Single plot waveform graph

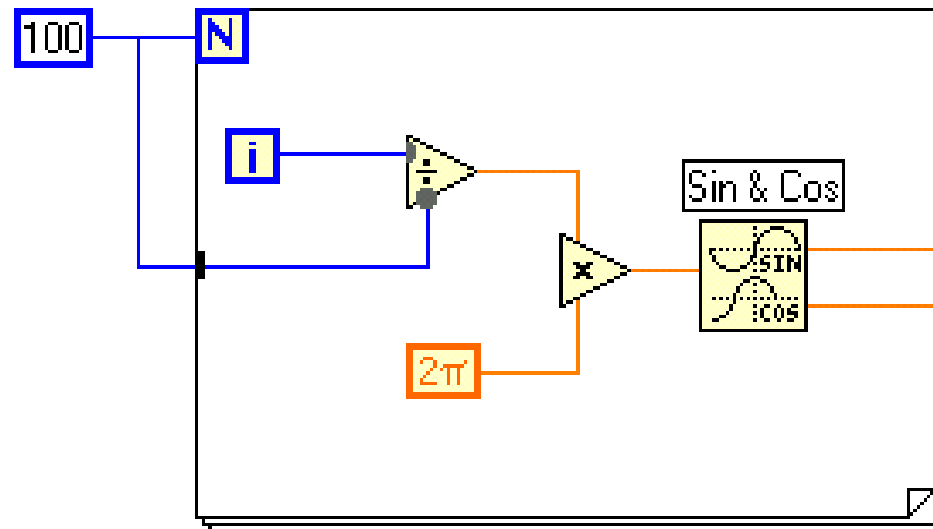


- **X₀** è il valore iniziale per le ascisse
- **deltaX** è la distanza orizzontale tra i punti del grafico

Multiple plot waveform graph

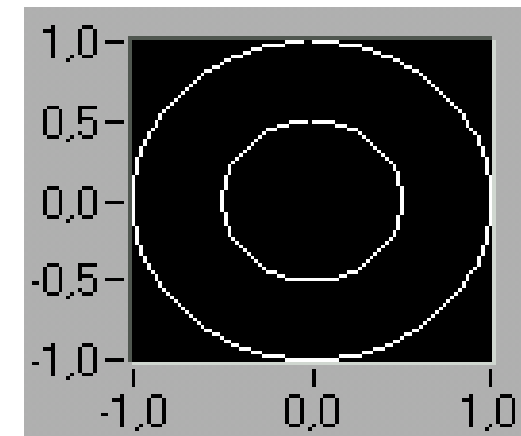
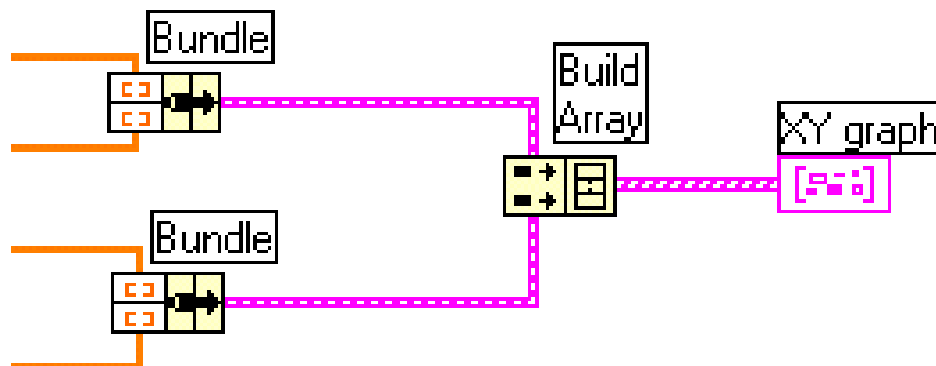


XY graph



Single-plot XY Graph

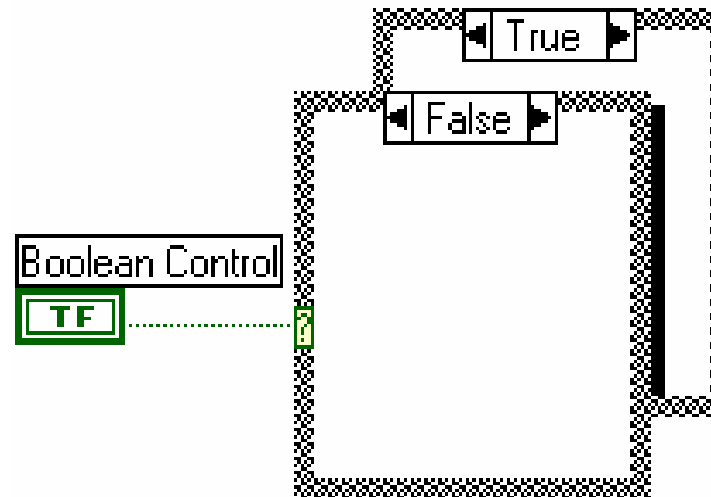
Multi-plot XY Graph



Case

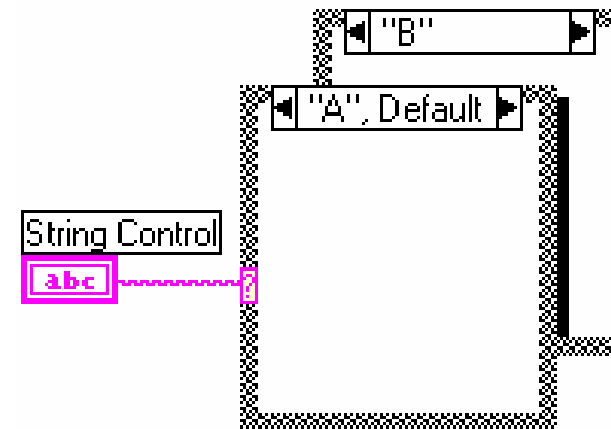
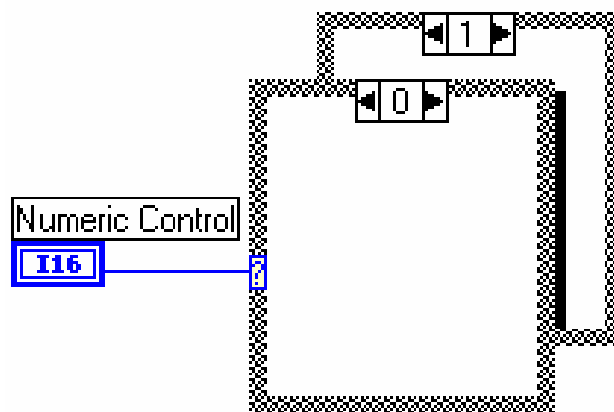
- Solo un sottodiagramma viene eseguito
- Il valore passato al selettore determina il codice da eseguire
- Il selettore può essere booleano, numerico o stringa

Selettore booleano:
if...then...else

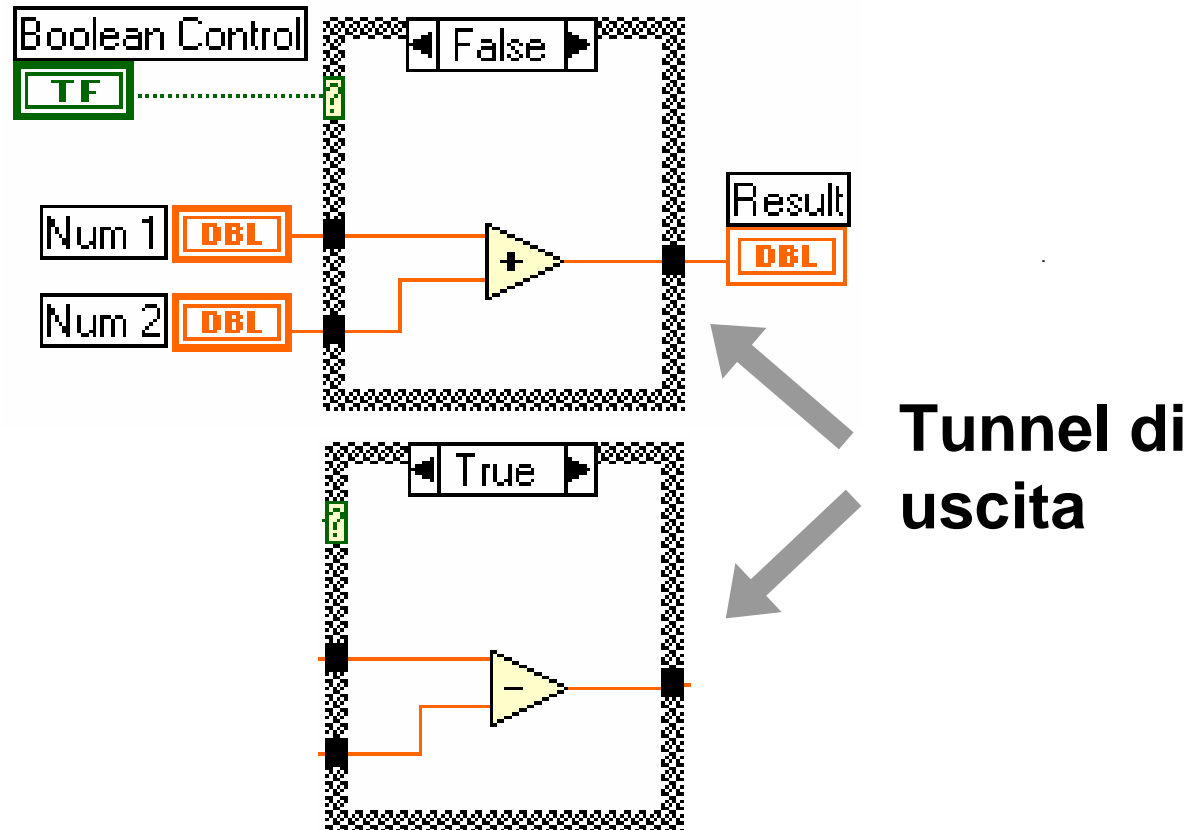


Case

- Con selettori di tipo numerico o stringa si possono gestire scelte multiple
- “Add Case” dal menu di contesto per aggiungere nuove scelte
- Un *Case* deve essere quello di *default*



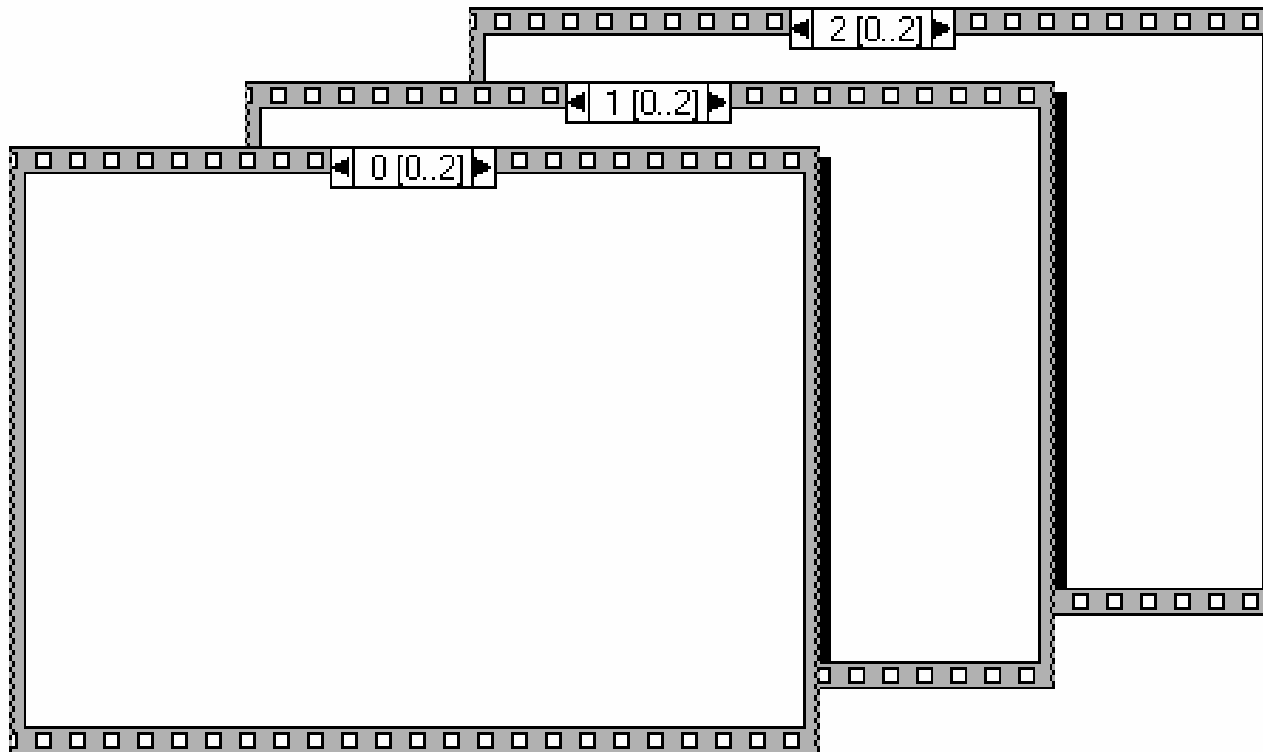
Ingressi ed uscite del case (tunnel)



- In TUTTI i case devono essere collegati TUTTI i tunnel

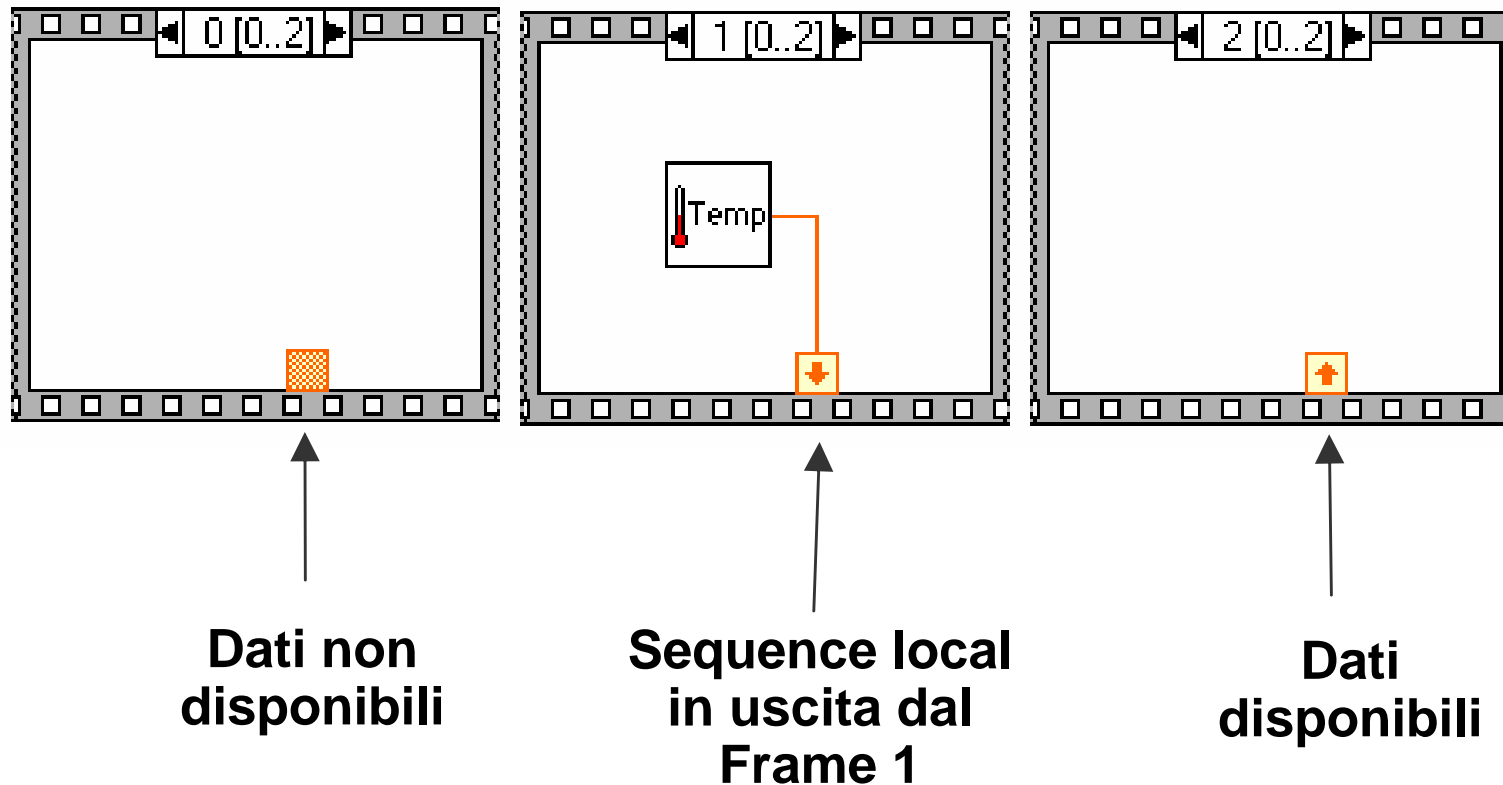
Sequence

- I sottodiagrammi vengono eseguiti uno dopo l'altro
- Consente di fissare l'ordine di esecuzione di VI non dipendenti



Sequence locals

- Passano dati da un frame al solo frame successivo
- Si definiscono sul bordo della Sequence



Formula Node

- Implementazione di equazioni complicate
- Variabili create sul bordo
- I nomi delle variabili sono *case sensitive*
- Ogni istruzione deve terminare con un punto e virgola (;)
- <Ctrl-H> per l'elenco delle funzioni disponibili

