UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Impiego di un dispositivo basato su FPGA per misure di power quality in Real-Time

Laureando
Ergest Islamaj

Relatore
Prof. Matteo Bertocco

Correlatore
Ing. Federico Tramarin

Laurea Triennale in
Ingegneria Elettronica

Anno Accademico
2012-2013

Familjes sime
dhe Esiones

Indice

Glossario

ADC	Analog to Digital Converter
AI	Analog Input
AIGND	Analog Input Ground
AO	Analog Output
BRAM	Block Random Access Memory
CAN	Controller Area Network
CLB	Configurable Logic Blocks
DAC	Digital to Analog Converter
DCM	Digital Clock Management
DFT	Discret Fourier Trasform
DIO	Digital Input/Output
DMA	Direct Memory Access
DSP	Digital Signal Processing
DTFT	Discret Time Fourier Trasform
FFT	Fast Fourier Trasform
FPGA	Field Programmable Gate Array
GND	Ground
IDC	Insulation Displacement Connector
IOB	Input/Output Bank
I/O	Input/Output
LSB	Least Significant Bit
LUT	Look-Up Tables
MIO	Multiple Input/Output
NI	National Instruments
PGIA	Programmable Gain Instumentation Amplifier
RAM	Random Access Memory
RSE	Referenced Single Ended

SDHC Secure Digital High Capacity

Capitolo 1

Introduzione

L'incessante crescita della richiesta di energia e il concomitante aumento del riscaldamento globale rendono sempre più pressante e attuale il tema delle fonti rinnovabili. A tal proposito, le soluzioni più diffuse ricorrono ai sistemi fotovoltaici ed eolici. La necessità di ridurre i consumi, contenendo al tempo stesso le emissioni inquinanti, impone profondi cambiamenti nella progettazione e nella realizzazione della rete elettrica. Infatti, non ha più senso parlare di una distribuzione univoca [3], dalle grandi centrali sino agli utilizzatori, siano essi di natura domestica o industriale. Allo stato dell'arte, va affermandosi un nuovo modello di rete elettrica, la cosiddetta rete intelligente o *smart grid*, in cui il consumatore occupa un ruolo di primo piano: da semplice user questo diventa un prosumer, ossia partecipa attivamente alla gestione e alla generazione dell'energia elettrica. Una simile evoluzione richiede il contributo da parte di diversi soggetti, sia da un punto normativo, sia da un punto di vista tecnologico. L'intelligenza di cui è dotata la rete consente un controllo distribuito dei flussi energetici, garantendo interventi più tempestivi e accurati. I protocolli standard vengono via via declinati a seconda delle specifiche caratteristiche della porzione di rete interessata. In tal senso va letta anche la particolare attenzione di cui godono i recenti progetti di *smart microgrid*. Allo stato dell'arte, una rete nazionale presenta numerosi aspetti critici che mal si conciliano con l'innovativo concetto di *smart grid*. Al contrario, una porzione più contenuta, dedicata ad esempio alla fornitura in un quartiere residenziale, rappresenta un ideale banco di prova per l'effettiva fattibilità del progetto. Generazione distribuita, flusso bidirezionale, adozione di sistemi di accumulo e rilascio, sono tutti aspetti che possono essere facilmente testati in ottica locale senza porre a repentaglio l'integrità della rete nazionale.
Il controllo centralizzato delle reti nazionali offre ottime garanzie in termini di stazionarietà della potenza erogata. Di contro, a livello di *smart microgrid* è necessario gestire l'apporto

delle sorgenti distribuite, quasi sempre intermittenti e di difficile predizione nel loro funzionamento. L'inclusione di tali fonti di energia deve rappresentare un miglioramento della rete, ossia non deve pregiudicare gli standard di stazionarietà e qualità del servizio offerto. In tal senso, massima priorità va dedicata alla gestione di eventi quali sovraccarichi, cadute di tensione. Mediante le moderne tecniche di elaborazione del segnale è possibile individuare eventuali anomalie e regolare un corretto equilibrio tra generazione e consumo in tempo reale.

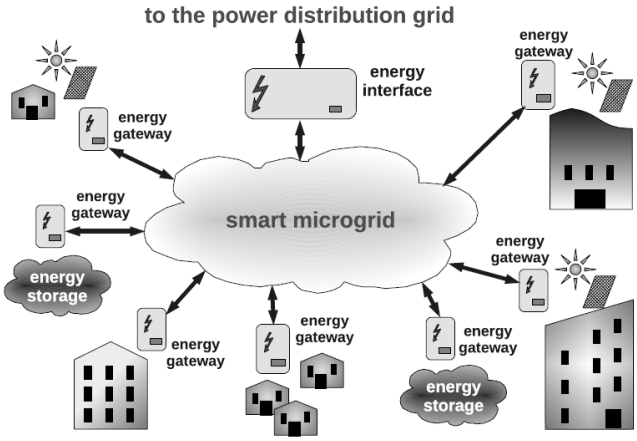


Figura 1.1 – Ambiente smart microgid [3]

Una simile forma di gestione distribuita richiede specifici accorgimenti di natura sia algoritmica, sia implementativa. Vari rami dell'ingegneria dell'informazione forniscono il proprio contributo, basti pensare al progetto dei dispositivi elettronici integrati diffusi lungo la rete o all'implementazione di efficienti algoritmi di elaborazione digitale del segnale.
Il presente elaborato affronta il problema degli sbalzi di tensione, cui spesso corrispondono eventi di sovraccarico o interruzioni di linea. In quest'ottica, tempestività e accuratezza divengono aspetti di primaria importanza: quanto più velocemente il sistema localizza il guasto, tanto più efficacemente potrà porvi rimedio. L'algoritmo qui presentato opera direttamente sul segnale in linea, in particolare ne considera modulo, ampiezza e fase. La stima di questi tre parametri non è immediata, ma richiede successivi passaggi di elaborazione volti ad eliminare i disturbi e i contributi spuri dovuti alla rete. A tal proposito, si adotterà un metodo di interpolazione che garantisce una stima spettrale immune da fenomeni quali lo spectral leakage. Il presente elaborato intende indagare la possibilità di implementare simili metodi di stima spettrale su una scheda NI sbRIO-9636, dotata al suo interno di un circuito FPGA. In particolar modo, l'attenzione si è concentrata sul carico computazionale

delle diverse fasi: un'eventuale progetto di smart microgrid impone tempi di esecuzione quanto più rapidi possibili.

Capitolo 2

Introduzione alla scheda sbRIO 9636 e ambiente LabVIEW

Negli ultimi anni si sta diffondendo l'utilizzo di dispositivi elettronici integrati nella gestione, nel controllo, nell'acquisizione e nell'elaborazione di informazioni sia in ambito delle telecomunicazioni sia in ambito dell'automazone industriale. Il dispositivo Single Board RIO 9636, nel seguito indicato per brevita in sbRIO, rilasciato dalla National Instruments è un dispositivo elettronico che comprende una FPGA, la cui funzionalità è definita via software. Nel seguente capitolo si analizzano prevalentemente le sue caratteristiche principali. Viene descritto brevemente il software LabVIEW utilizzato per la programmazione del dispositivo, ed infine si da un esempio di utilizzo della scheda presentando l'acquisizione di un semplice segnale analogico generato da un generatore di tensione ed evidenziando i vari passi di sviluppo di un primo progetto.

2.1 Componenti necessari

2.1.1 Hardware richiesto

Per utilizzare la scheda NI sbRIO si devono avere a disposizione i seguenti componenti hardware [8]:

- la scheda sbRIO;
- un alimentatore da 9-30 VDC;
- un cavo Ethernet.

Sono inoltre necessari ulteriori connettori utili alla connessione della scheda sbRIO a elementi di trasduzione o altri dispositivi e strumenti esterni.

2.1.2 Software richiesto

I pacchetti necessari per lo sviluppo di applicativi che controllano e interagiscono con la sbRIO 9636 sono [8]:

- labView versione 2011.1 o successivo;
- labView Real Time Module versione 2011.1 o successivo;
- labView FPGA module versione 2011.1 o successivo;
- drivers NI-RIO versione 4.1 o successivo.

2.2 Descrizione della scheda sbRIO 9636

La National Instruments sbRIO (Figura 2.1) è un dispositivo embedded dotato di [8]:

Figura 2.1 – Scheda NI sbRIO 9636 [7]

- processore da 400 MHz;
- memoria RAM da 512 MB;

- DRAM da 256 MB per analisi e controllo deterministico;
- FPGA Xilinx Spartan-6 LX45 riconfigurabile per temporizzazione, elaborazione in linea e controllo personalizzato;
- 16 input analogici a 16-bit, 4 output analogici a 16-bit, che possono operare rispettivamente ad una frequenza massima di 200kHz e 336kHz;
- 28 linee DIO da 3.3 V;
- una porta integrata 10/100BASE-T Ethernet;
- un ingresso USB host;
- un ingresso per memory card SDHC;
- una porta CAN, due porte seriali RS232 e una porta seriale RS485.

L'FPGA, componente principale della scheda sbRIO, è un circuito integrato digitale la cui funzionalità è programmabile via software. Il dispositivo NI monta una FPGA Xilinx Spartan-6 LX45 che ha le seguenti caratteristiche [8]:

- 54.576 flip-flops;
- 27.288 6-nput LUTs;
- 58 moltiplicatori DSP48s;
- blocco RAM disponibile 400kb, blocchi da 2,088 kbits;
- 5 canali DMA;
- il clock interno è impostabile in un range di frequenza 60 - 160 MHz.

La Figura 2.2 mostra uno schema della FPGA Spartan 6, di cui di seguito vengono riportati i blocchi fondamentali [10]:

- CLB: sono le risorse logiche principali per l'attuazione di circuiti sequenziali e combinatori. Ogni CLB è costituito da una matrice di connessione configurabile, alcuni circuiti di selezione (MUX, etc) e flip-flop. La matrice di commutazione è molto flessibile ed è configurata per gestire la logica combinatoria. Ogni elemento di un singolo CLB è collegato alla matrice interruttore la quale regola l'accesso alla matrice di routing (Figura 2.3).

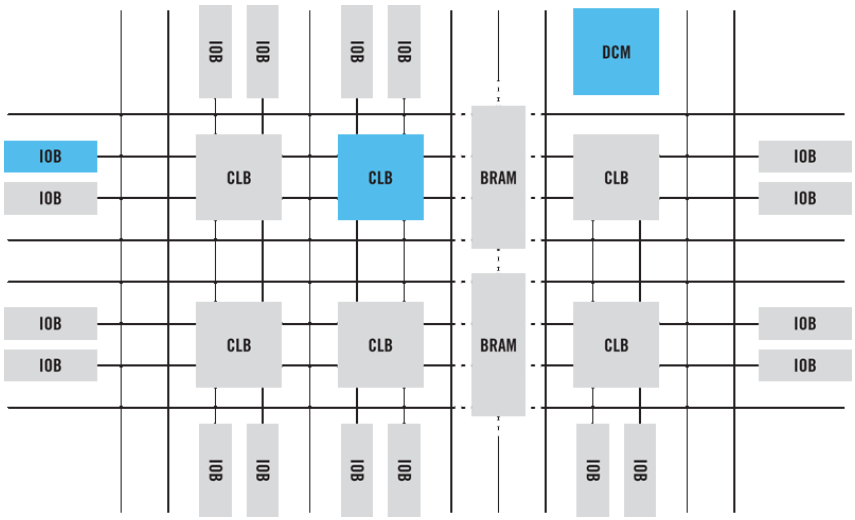


Figura 2.2 – Schema FPGA Spartan 6 [11]

Un CLB contiene due ”slice” (sezioni) le quali non hanno nessun collegamento tra di loro (Figura 2.3). Ogni ”slice” contiene quattro funzioni logiche (o tabelle di consultazione dette LUT) e otto elementi di memorizzazione i quali possono essere flip-flop di tipo D oppure dei latch;

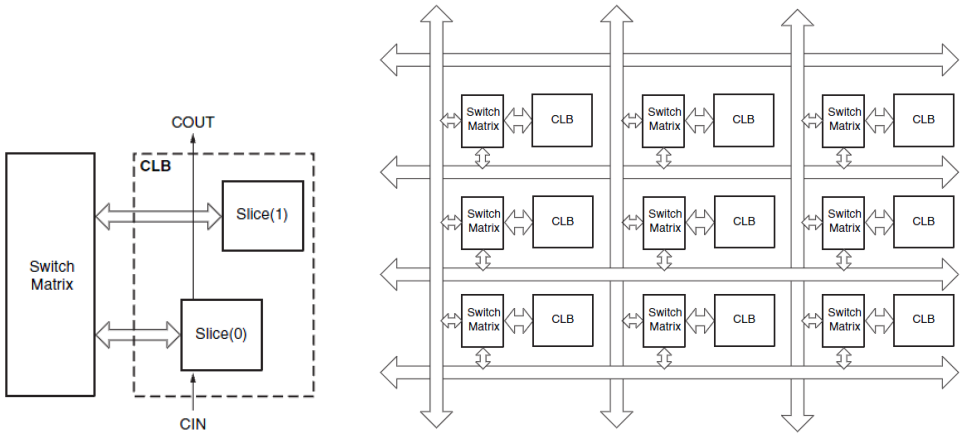


Figura 2.3 – Schema singolo CLB ed organizzazione su FPGA [10]

- IOB: l’FPGA fornisce il supporto per decine di standard I/O, in modo tale da garantire la flessibilità del sistema. Gli I/O della FPGA sono classificati in ”banchi” (IOB) con ciascun ”banco” indipendente ed in tale modo in grado di supportare diversi standard di I/O;

- RAM: la memoria RAM della FPGA è suddivisa in blocchi di memoria. La FPGA

Spartan 6 LX45 è costituita da una memoria RAM da 400Kb sudduvisa in blocchi di memoria (BRAM) da 2,088 kbits;

- DCM: agisce direttamente nella distribuzione del clock globale di rete. In tal modo, DCM risolve una varietà di problemi comuni al clock, in particolare ad alte prestazioni e ad alta frequenza, come ad esempio quello di eliminare ritardi di distribuzione del clock sia all'interno del dispositivo sia su dispositivi esterni oppure di moltiplicare o dividere una frequenza di clock in ingresso.

Nella Tabella 2.4 vengono descritti i connettori consigliati da NI per le connessioni alla scheda. Si evidenzia il connettore 50 Pin IDC Header necessario per l'ingresso analogico/digitale.

Connector	Description	Manufacturer, Part Number	Recommended Mating Connector	NI Solution
Power	2-position, mini-fit JR, H = 0.411 in.	Molex, 46999-0144	Molex, 50-36-1673 w/ 0457501211	NI, Power Plug Assembly 152834-01
RS-232/485/CAN IDC header	10-pin, 0.100 in. CT, shrouded, H = 0.370 in.	Samtec, TST-105-01-L-D	Tyco, 1658622-1	NI, 10-pin to 9-pin D-SUB, 153158-10
50 Pin IDC Header	50-pin, 2 mm CT, Shrouded, H = 0.155 in.	Samtec, STMM-125-02-L-D	Tyco, 2-111626-3	NI, 50 pos. ribbon cable, 154041-12
RMC Connector	240-pin, 40 x 6 pos., high density open pin field SEARAY	Samtec, SEAF-40-06.5-S-06-2	Samtec, SEAM-40-XX.X-S-06-2	—

Figura 2.4 – Descrizione connettori sbRIO [8]

Con riferimento alla descrizione dei connettori della scheda in Figura 2.4 e di quelli in Figura 2.6 il dispositivo sbRIO può essere connesso ad altri dispositivi come mostra la Figura 2.5:

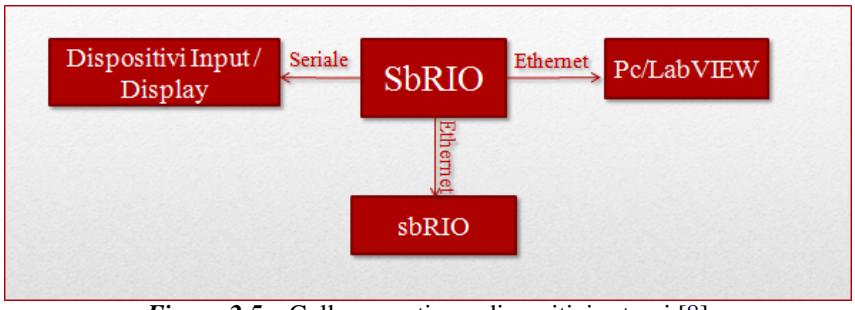


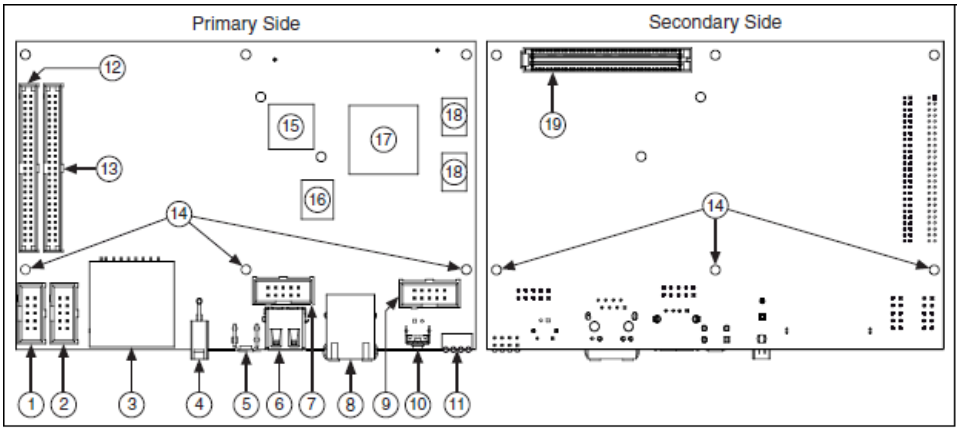
Figura 2.5 – Collegamenti con dispositivi esterni [8]

- trasmissione dati con il Pc solo tramite cavo ethernet;

- collegamento con dispositivi sbRIO tramite cavo ethernet;
- collegamento a dispositivi di input e display tramite cavi seriali SR232 e SR485;
- collegamento a dispositivi di memoria esterna tramite Memory Card SDHC oppure tramite USB Flash.

La Figura 2.6 rappresenta uno schema di tutti i componenti della scheda sbRIO. Gli elementi rilevanti della scheda sono [8]:

- l'alimentazione (punto 4);
- la porta Ethernet (punto 8);
- i LED (punto 11);
- l'FPGA (punto 15);
- il connettore analogico J503, MIO (punto 13);
- il connettore digitale J502, DIO, (punto 12).

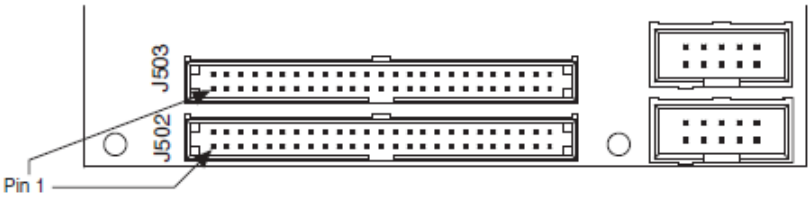


1 W502, RS-485 (COM3)
2 W503, RS-232 (COM2)
3 J504, SDHC
4 J506, Power Connector
5 Chassis Ground Bracket
6 J507, USB Host Port
7 W500, CAN (CAN0)
8 J505, RJ-45 Ethernet Port
9 W501, RS-232 Serial (COM1)
10 Reset Switch
11 LEDs
12 J502, DIO
13 J503, MIO
14 Mounting Holes Connected to Chassis Ground
15 FPGA
16 NAND Flash
17 Processor
18 DDR Memory
19 J1, RIO Mezzanine Card Connector

Figura 2.6 – Componenti della scheda sbRIO [8]

2.2.1 L'ingresso analogico integrato J503

Ai canali analogici di I/O [8] della scheda NI sbRIO 9636 si accede per mezzo del connettore J503, MIO il quale prevede connessioni per ingressi analogici, uscite analogiche e la massa. Nella Figura 2.7 viene evidenziato il connettore e le funzionalità dei 50 Pin a sua disposizione. Si possono notare i pin dedicati alla massa come ad esempio il pin 1, i pin dedicati agli ingressi analogici come ad esempio il pin 6 e i pin dedicati alle uscite analogiche come ad esempio il pin 30.



J502, DIO

D GND	1	2	DIO4
D GND	3	4	DIO5
D GND	5	6	DIO6
D GND	7	8	DIO7
D GND	9	10	DIO8
D GND	11	12	DIO9
D GND	13	14	DIO10
D GND	15	16	DIO11
D GND	17	18	DIO12
D GND	19	20	DIO13
D GND	21	22	DIO14
D GND	23	24	DIO15
D GND	25	26	DIO16
D GND	27	28	DIO17
D GND	29	30	DIO18
D GND	31	32	DIO19
D GND	33	34	DIO20
D GND	35	36	DIO21
D GND	37	38	DIO22
D GND	39	40	DIO23
D GND	41	42	DIO24
D GND	43	44	DIO25
D GND	45	46	DIO26
D GND	47	48	DIO27
+5V	49	50	+5V

J503, MIO

AI GND	1	2	AI0
AI8	3	4	AI GND
AI9	5	6	AI1
AI GND	7	8	AI2
AI10	9	10	AI GND
AI11	11	12	AI3
AI GND	13	14	AI4
AI12	15	16	AI GND
AI13	17	18	AI5
AI GND	19	20	AI6
AI14	21	22	AI GND
AI15	23	24	AI7
AI GND	25	26	AI GND
AO GND	27	28	AO0
AO GND	29	30	AO1
AO GND	31	32	AO2
AO GND	33	34	AO3
AO GND	35	36	NC
AO GND	37	38	NC
AO GND	39	40	NC
AO GND	41	42	NC
D GND	43	44	DIO0
D GND	45	46	DIO1
D GND	47	48	DIO2
D GND	49	50	DIO3

Figura 2.7 – Descrizione ingressi digitali e analogici [8]

La Figura 2.8 mostra lo schema funzionale a blocchi di un canale analogico AI della sbRIO [8].

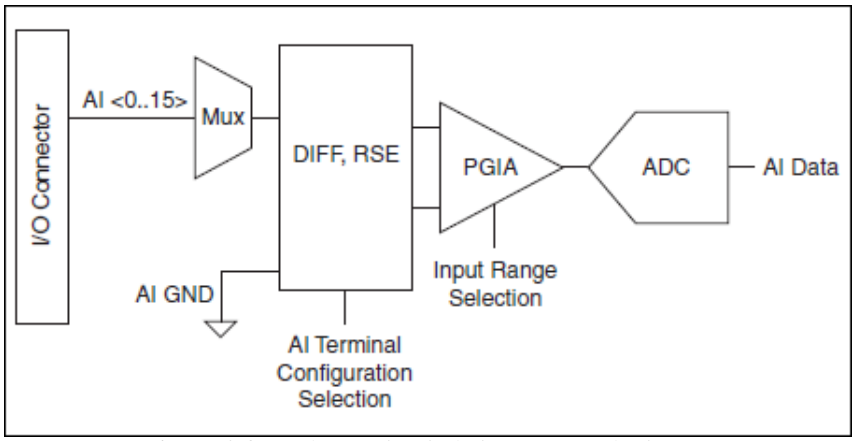


Figura 2.8 – Schema circuitale ingresso analogico [8]

Con riferimento alla Figura 2.8 si evidenziano i seguenti elementi fondamentali:

- I/O Connector: tramite questo blocco è possibile collegare segnali analogici al dispositivo sbRIO attraverso il connettore J503, MIO evidenziato in Figura 2.7;
- un canale di ingresso è collegato al multiplexer analogico MUX il quale determina selettivamente quale ingresso viene trasferito al blocco successivo;
- AI Terminal Configuration Selection: in questa sezione viene selezionato il tipo di misura da effettuare tra differenziale DIFF ed RSE mentre l'AIGND e la massa del segnale analogico preso in considerazione;
- PGIA: si tratta di un amplificatore a guadagno programmabile,il PGIA può amplificare o attenuare un segnale analogico per assicurare la massima risoluzione del convertitore analogico-digitale, presente nel blocco successivo;
- ADC: il convertitore analogico-digitale converte il segnale analogico in ingresso in un segnale digitale, il quale trasmette direttamente il dato.

Configurazione segnali analogici in ingresso

Il dispositivo sbRIO 9636 può effettuare due tipi di misurazione del segnale analogico in ingresso [8]:

- Differenziale: si misura la differenza di potenziale tra due terminali come mostra lo schema circuitale in Figura 2.9 in cui viene evidenziata la differenza tra terminale positivo ($AI_+ = V_+$) e negativo ($AI_- = V_-$) di un generico ingresso analogico AI. Quando

si configura il sistema per eseguire una misura differenziale, le 16 linee di ingresso analogico vengono raggruppati in 8 coppie, ciascuna delle quali è associata a un canale differenziale, come riportato in Tabella 2.1. Ad esempio il canale differenziale 0 è formato dalle linee analogiche AI0 e AI8, mentre il canale differenziale 1 è composto dagli ingressi AI1 e AI9. Sono riportati anche i segni "+" e "-" per indicare quale canale funge da terminale positivo e quale da terminale negativo. L'amplificatore differenziale a guadagno programmabile (blocco PGIA) amplifica oppure attenua la differenza di potenziale tra i terminali positivo e negativo del canale differenziale su cui avviene la misura, come mostrato in Figura 2.8, al fine di adattare il segnale per la conversione analogico-digitale che avviene nel blocco successivo;

Channel	Signal +	Signal -
0	AI0	AI8
1	AI1	AI9
2	AI2	AI10
3	AI3	AI11
4	AI4	AI12
5	AI5	AI13
6	AI6	AI14
7	AI7	AI15

Tabella 2.1 – Ingressi analogici Differenziali [8]

- Referenced Single-Ended: in una misura di tipo RSE viene misurata la differenza di potenziale tra un segnale collegato ad un canale di ingresso e la massa del sistema come mostra lo schema circuitale in Figura 2.9. A differenza della modalità differenziale, un canale è composto da un solo terminale di ingresso, in quanto il secondo terminale viene condiviso da tutti gli altri canali ed è collegato alla massa del sistema, intesa come potenziale comune della scheda e della circuiteria esterna che genera il segnale da acquisire e misurare. In Figura 2.9 è mostrato un esempio di configurazione per una misura riferita a massa. Questa configurazione ha il vantaggio di raddoppiare il numero di canali disponibili per le misurazioni rispetto alla modalità differenziale. Tuttavia una misura differenziale normalmente restituisce dei valori più accurati, in quanto permette all'amplificatore di eliminare eventuali tensioni di disturbo presenti in entrambi i terminali.

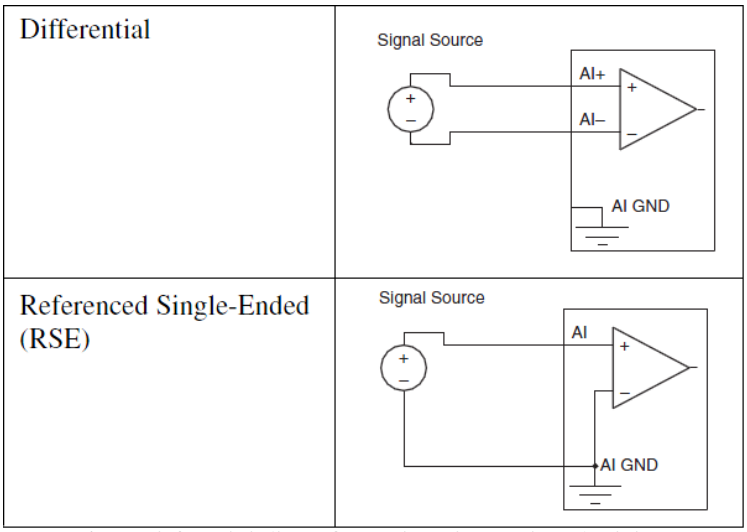


Figura 2.9 – Tipi di configurazione ingressi analogici [8]

Intervallo Ingresso Analogico

Un intervallo di ingresso [8] è un insieme di tensioni di ingresso che un canale analogico può digitalizzare con la risoluzione specificata. I dispositivi NI sbRIO con intervallo d'ingresso selezionabile hanno un PGIA, che amplifica o attenua il segnale analogico AI a seconda del campo di ingresso. E possibile programmare l'intervallo di ingresso per ciascun canale AI indipendentemente.

L'ADC converte gli ingressi analogici in valori digitali discreti. Per un 12-bit ADC ci sono 2^{12} (4096) valori possibili, e per un ADC a 16-bit ci sono 2^{16} (65.536) valori possibili. Questi valori sono distribuiti in modo uniforme in tutto l'intervallo di ingresso, e la differenza di tensione tra i valori è proporzionale all'intervallo di ingresso del canale selezionato ed è pari al valore del bit meno significativo (LSB size) del canale.

La seguente equazione mostra come calcolare il valore del LSB per un set di canali che va da -10V a 10V, con un ADC a 16 bit.

$$\frac{10V - (-10V)}{65,536} = 305\mu V \tag{2.1}$$

La Tabella 2.2 mostra i campi di ingresso e le ampiezze degli LSB che ne derivano per i canali AI sul dispositivo NI sbRIO 9636.

Device	Input Range	Bit Resolution	LSB Size
NI sbRIO	-10 V to 10 V	16	320 μV*
	-5 V to 5 V		160 μV*
	-2 V to 2 V		64 μV*
	-1 V to 1 V		32 μV*
*Includes 5% averranging			

Tabella 2.2 – Intervalli d'ingresso e risoluzioni del dispositivo sbRIO [8]

Range tensione di lavoro

Il PGIA [8] sui dispositivi sbRIO opera normalmente amplificando il segnale di interesse mentre respinge segnali di modo comune secondo le seguenti tre condizioni:

- La tensione di modo comune (V_{cm}), deve essere inferiore a ±10V. V_{cm} è una costante per tutti gli intervalli selezionati.

$$V_{cm} = \frac{V_+ + V_-}{2} \tag{2.2}$$

- Il segnale di tensione differenziale (V_s), deve essere compreso nell'intervallo di valori ammissibili del canale selezionato. Se V_s è al di fuori del range selezionato, il segnale non prosegue ai blocchi successivi.

$$V_s = V_+ - V_- \tag{2.3}$$

- La tensione totale dell'ingresso positivo (V_+), deve essere inferiore alla massima tensione di lavoro specificata per tale intervallo come mostra la Tabella 2.3.

$$V_+ = V_{cm} + \frac{V_s}{2} \qquad V_- = V_{cm} - \frac{V_s}{2} \tag{2.4}$$

Se una qualsiasi di queste condizioni viene superata, la tensione di ingresso è bloccata fino a che la condizione di errore non viene rimossa.

Uscita Analogica Integrata

Il dipositivo sbRIO-9636 [8] ha quattro canali a 16 bit in grado di pilotare uscite analogiche a ±10 V. Tutti i canali AO sono con riferimento a massa (GND). Una descrizione dettagliata

Range	Working Voltage
10 V	± 11V
5 V	± 10.5V
2 V	± 9V
1 V	± 8.5V

Tabella 2.3 – Range Tensione massima di Lavoro (signal + modo comune) [8]

delle uscite analogiche è evidenziata in Figura 2.7 mentre la Figura 2.10 mostra lo schema funzionale a blocchi dell'uscita analogica in cui:

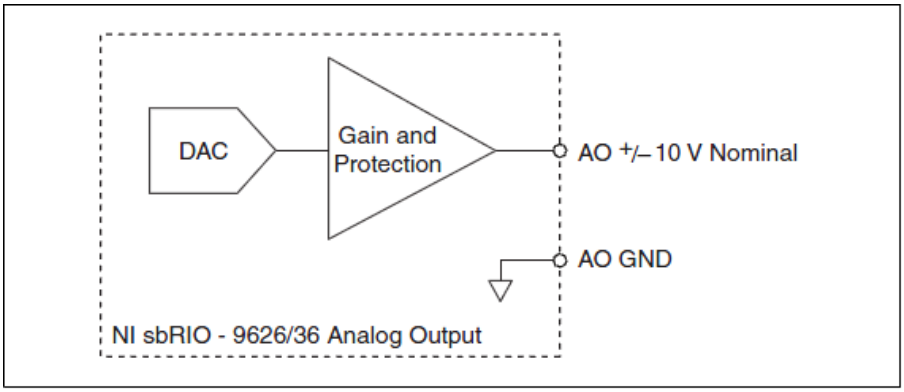


Figura 2.10 – Schema uscita Analogica [8]

- DAC: il primo blocco e costituito da un DAC, il quale converte ogni segnale in uscita da digitale ad analogico
- Gain and Protection: il successivo è un blocco di guadagno, che amplifica il segnale in modo che in uscita sia ±10V, evitando che la tensione di uscita superi l'intervallo di tensione ±10V.

L'uscita analogica viene inizializzata (abilitata e impostata su 0 V) la prima volta che viene caricato un programma nella FPGA in cui o l'AI o l'AO della scheda vengono utilizzati. L'AO viene inizializzata a 0 V ogni volta l'FPGA è programmata in modo da utilizzare o l'AI o l'AO.

2.2.2 I/O Digitali

Il dispositivo sbRIO è costituito da 28 linee digitali di I/O di 3.3 V. Agli ingressi digitali della scheda NI sbRIO 9636 si accede per mezzo del connettore J502, DIO il quale prevede

connessioni per ingressi, uscite e la massa. Facendo riferimento alla Figura 2.7 vengono spiegate le funzionalità dei 50 Pin a disposizione del connettore. Si possono notare i pin dedicati alla massa, tutti i pin dispari a sinistra del connettore e i pin dedicati agli ingressi/uscite digitali, tutti i pin pari a destra del connettore. La Tabella 2.4 mostra i livelli digitali corrispondenti alle tensioni associate [8].

Con riferimento alla Figura 2.11 si ha che:

- tensione in ingresso compresa tra 0 V ÷ 0.8 V, corrisponde ad un livello logico basso, cioè 0;
- tensione in ingresso compresa tra 2 V ÷ 5.25 V, corrisponde ad un livello logico alto, cioè 1;
- tensione in uscita compresa tra 0 V ÷ 0.4 V, corrisponde ad un livello logico basso, cioè 0;
- tensione in uscita compresa tra 2.4 V ÷ 3.465 V, corrisponde ad un livello logico alto, cioè 1;

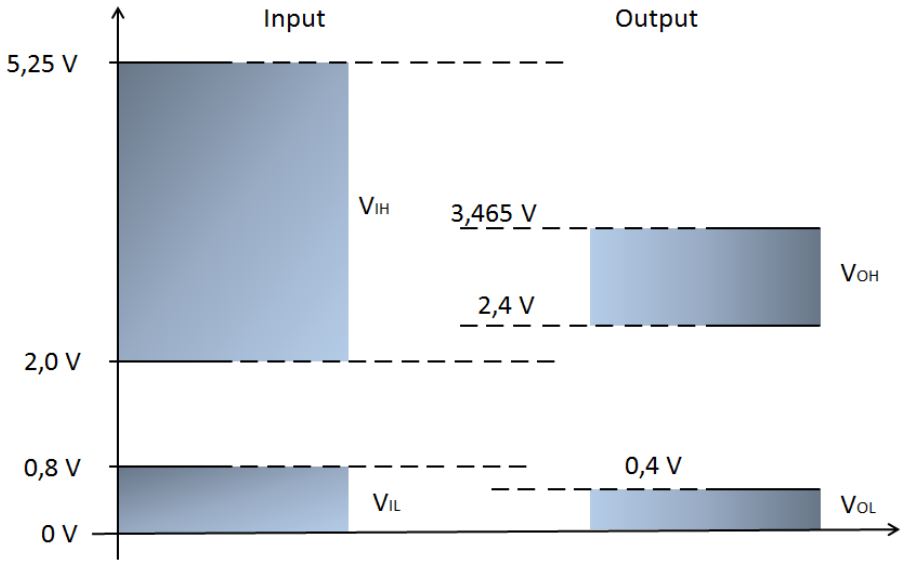


Figura 2.11 – Schema grafico dei livelli di I/O Digitali

Input Voltage	Input Logic Levels
0 V min ÷ 0,8 max 2,0 V min ÷ 5,25 V max	0 1
Output Voltage	**Output Logic Levels**
0 V min ÷ 0,4 max 2,4 V min ÷ 3,465 max	0 1

Tabella 2.4 – Livelli di I/O Digitali [8]

2.2.3 Interpretazione dei LED

Con riferimento alla Figura 2.12 il Power LED [8] si accende quando il dispositivo si collega ad un alimentatore, rimane acceso durante l'utilizzo della scheda ed indica un adeguata alimentazione. Lo Status LED è normalmente spento, quando si accende l'sbRIO esegue un operazione di auto-test. Durante questa operazione il Power e lo Status LED si accendono, se l'operazione va a buon fine la status LED si spegne. Il dispositivo indica errori specifici lampeggiando lo Status LED un certo numero di volte per pochi secondi. Lo User1 LED e lo User FPGA1 LED possono essere impostati dal utente tramite appositi VI in LabVIEW in modo tale da essere usati nelle applicazioni Real Time oppure come indicatore dello stato dell'applicazione in ambiente FPGA.

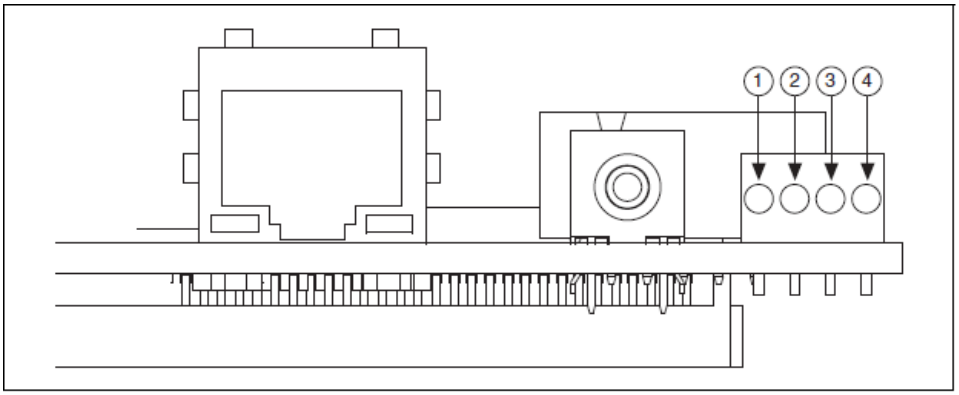


Figura 2.12 – LED del dispositivo sbRIO [8]

2.3 Presentazione ambiente LabVIEW

LabVIEW [6] (Laboratory Virtual Instrumentation Engineering Workbench) è un ambiente di sviluppo integrato per il linguaggio di programmazione visuale realizzato da National Instruments, utilizzato principalmente per acquisizione, analisi dati, controllo di processi e nell'automazione industriale.

I linguaggio di programmazione usato in Labview, chiamato *G* da Graphic-Language, è un linguaggio di programmazione grafico a dataflow (flusso di dati). La definizione di strutture dati avviene con icone e altri oggetti grafici, ognuno dei quali incapsula funzioni diverse, le quali sono unite dal programmatore tramite linee di collegamento, in modo da formare una sorta di diagramma di flusso (o diagramma a blocchi). Tale linguaggio viene definito dataflow in quanto la sequenza di esecuzione è definita e rappresentata dal flusso dei dati stessi attraverso i fili monodirezionali che collegano i blocchi funzionali. Le linee, infatti, gestiscono la propagazione delle variabili e qualsiasi nodo viene eseguito non appena i dati di ingresso sono disponibili. I dati possono anche scorrere in parallelo attraverso blocchi e fili non consecutivi.

Il vantaggio di LabVIEW è quello di essere un programma di semplice utilizzo, di grande versatilità e di facile apprendimento, in quanto presenta una modalità di programmazione a blocchi, di tipo visuale ed intuitivo.

2.3.1 VI (Virtual Instrument)

I programmi realizzati in LabView prendono il nome di strumenti virtuali (Virtual Instrument, VI). Un Virtual Instrument [6] permette l'interazione tra calcolatore e strumentazione fornendo all'utente un opportuno pannello frontale grafico per il dialogo con il VI stesso. Il termine strumento è dovuto al fatto che durante l'esecuzione i programmi sviluppati presentano agli utilizzatori una interfaccia analoga a quella di uno strumento di misura, mentre il termine virtuale è dovuto al fatto che l'utente interagisce con un programma in esecuzione e non con un dispositivo fisico.

Nell'ambiente di sviluppo, i VI sono costituiti da tre componenti principali:

- Il pannello frontale (Front Panel)
- Lo schema a blocchi (Block Diagram)
- Icona/Connettori (Icon/Connector)

Pannello frontale

Il pannello frontale è l'interfaccia utente del VI [6]. Si realizza con controlli e indicatori, che costituiscono i terminali interattivi d'ingresso e d'uscita, rispettivamente. I controlli sono matrici, manopole, potenziometri, pulsanti, quadranti e molti altri; simulano i dispositivi d'ingresso degli strumenti e forniscono dati allo schema a blocchi del VI. Gli indicatori sono grafici, tabelle, LED, termometri e molti altri; simulano i dispositivi d'uscita degli strumenti e visualizzano i dati che lo schema a blocchi acquisisce o genera. Un esempio di panello frontale è quello mostrato in Figura 2.25, in cui i controllori sono il *reource name* con il quale si sceglie la sorgente ed il pulsante di *Stop* con il quale si ferma il processo di acquisizione. Mentre gli indicatori sono il grafico che mostra il segnale acquisito e *AI1* che mostra il valore acquisito in quel istante.

Schema a blocchi

Lo schema a blocchi [6] è il diagramma di flusso che rappresenta il codice sorgente in formato grafico. Gli oggetti del pannello frontale appaiono come terminali di ingresso o uscita nello schema a blocchi. Gli oggetti dello schema a blocchi comprendono:

- funzioni
- strutture
- fili di collegamento
- costanti
- terminali
- chiamate ad altri VI (subVI)
- commenti testuali

Le funzioni sono chiamate esse stesse VI, anche se non hanno un loro pannello frontale e un loro schema a blocchi. Possono avere un numero indefinito di ingressi e di uscite come ogni VI. La Figura 2.13 mostra la funzione di costruzione di un array in cui vengono evidenziati gli ingressi (ad esempio array ed element) e l'uscita che è l'array cotruito.

Le strutture eseguono il controllo di flusso di base. Ad esempio il ciclo FOR (Figura 2.14) è rappresentato da un contenitore quadrato, che ripete N volte la porzione di schema a blocchi che si trova al suo interno. Mentre il ciclo While (Figura 2.14) ripete lo schema a blocchi al suo interno finché non si verifica una determinata condizione di uscita dal ciclo.

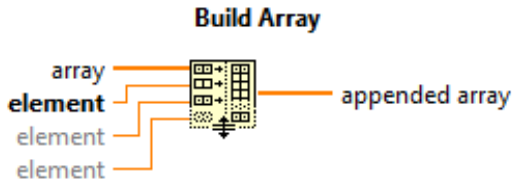


Figura 2.13 – Esempio di funzione considerata come subVI

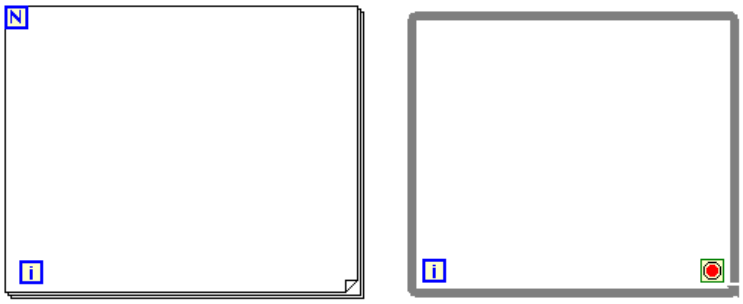


Figura 2.14 – Struttura ciclo FOR e ciclo While in LabView

I fili di collegamento possono trasportare quantità di dati di qualunque tipo, anche aggregati definiti dal programmatore. Il colore e lo spessore del filo cambiano di conseguenza per permetterne una facile identificazione. Come mostra la Figura 2.15, valori interi scorrono su fili blu e stringhe su fili rosa.

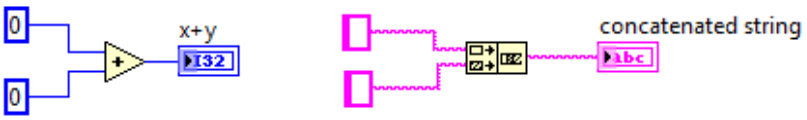


Figura 2.15 – Esempio di collegamenti tra tipologie diverse di dati in LAbVIEW

Lo schema a blocchi può essere reso visibile anche durante l'esecuzione, cosa molto utile in fase di debug, in quanto se richiesta si può visualizzare con un'animazione al rallentatore il movimento dei dati lungo i fili e il loro valore momentaneo.

La Figura 2.16 mostra un esempio di VI in cui si ha lo schema a blocchi ed il corrispondente panello frontale. Si può notare l'utilizzo di funzioni, di subVI e di collegamenti tramite fili tra di essi.

Icone/Connettori

Ogni VI [6] può essere a sua volta utilizzato come subVI e comparire all'interno dello schema a blocchi di altri VI, proprio come una qualsiasi funzione, e come tale può avere ingressi

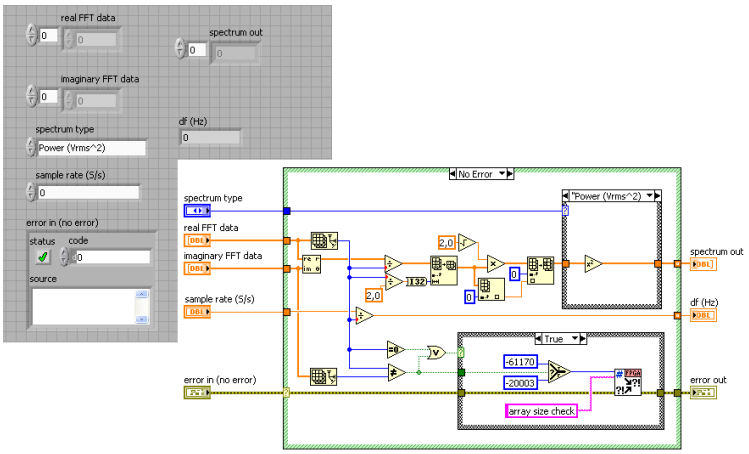


Figura 2.16 – Esempio di panello frontale e corrispondente diagramma a blocchi

e uscite a cui collegare le linee di flusso. I VI hanno un ”riquadro connettori” che serve a definire qual'è l'aspetto del VI quando appare come subVI in uno schema a blocchi, l'anteprima dell'icona, ma soprattutto come e dove vanno collegate le linee per permettere il passaggio dei dati. In generale ogni controllo può essere associato a un ingresso e ogni indicatore può essere associato a un'uscita. La Figura 2.17 mostra la barra di stato del panello frontale del VI in cui a destra si può notare il ”riquadro connettori” in cui è stato evidenziato un quadratino nero al quale poi sarà associato un ingresso. Affianco al riquadro connettori si trova l'anteprima dell'icona del VI che solitamente descrive quale sia la funzione del VI o subVI.

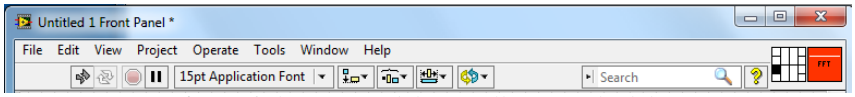


Figura 2.17 – Barra del Pannello Frontale del VI

2.4 Configurazione di NI sbRIO 9636

Prima di effettuare un acquisizione bisogna configurare il dispositivo.

- Si collega la scheda sbRIO tramite cavo Ethernet a un PC.

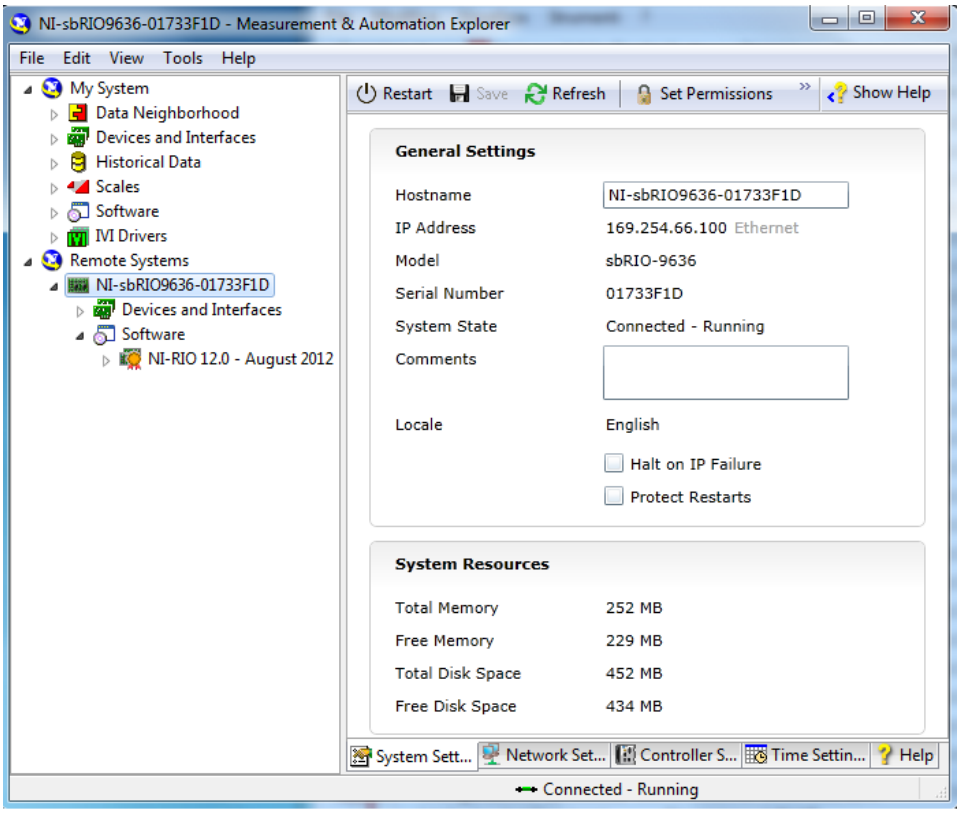


Figura 2.18 – Measurement and Automation Explorer

- Si apre l'applicativo MAX software installato contemporaneamente all'installazione di LabView. Espandendo la voce My System dovremmo vedere il nome del dispositivo il quale ha già un IP auto-assegnato (Vedi Figura 2.18). Se questo non avviene allora vanno verificate le impostazioni di sicurezza di windows e fare in modo che il programma Measurement ed Automation Explorer abbia accesso alla rete.

- Cliccando su Software vengono visualizzate informazioni sul software installato (ad esempio NI-RIO 12.0-August 2012) sulla scheda sbRIO come mostra la Figura 2.19. Se nessun software è installato, si clicca con il tasto destro su Software → Add/Remove Software in questo modo che si installi sul dispositivo il software necessari per l'utilizzo, come mostra la Figura 2.20. Il software NI-RIO 12.0 - August 2012 (Fi-

gura 2.20) normalmente viene installato sul PC contemporaneamente a quella di LabVIEW(paragrafo 2.1).

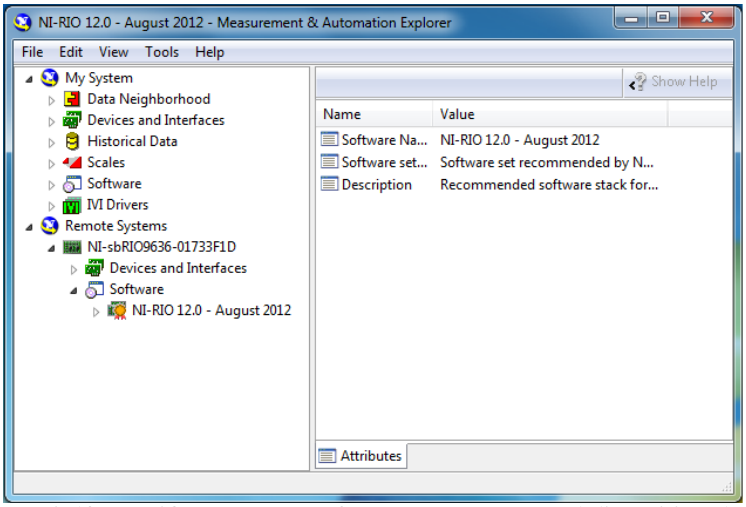


Figura 2.19 – Verifica presenza software NI-RIO 12.0 nel dispositivo sbRIO

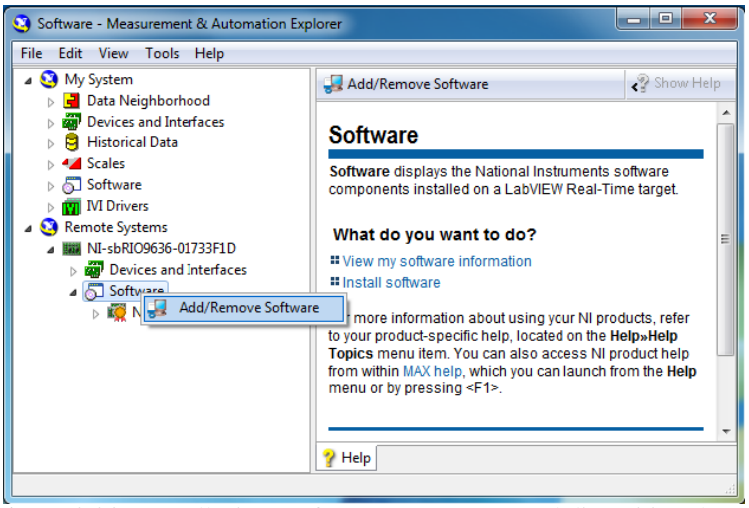


Figura 2.20 – Installazione software NI-RIO 12.0 nel dispositivo sbRIO

2.5 Acquisizione di un segnale analogico

Per l'acquisizione di un segnale analogico lo start-up da seguire e quello mostrato in Figura 2.21. Il segnale analogico è generato da un generatore di tensione il quale è collegato alla scheda e ad un DSO tramite cavi coassiali. Il dispositivo è connesso al PC tramite cavo ethernet. La sbRIO in questo modo acquisisce il segnale, lo elabora ed invia i dati al PC mentre il DSO visualizza il segnale generato come verifica. Tramite computer si ha accesso al panello frontale dell'applicazione realizzata in labVIEW con la quale si può comunicare con la scheda ed impostare i dati di front end del dispositivo stesso.

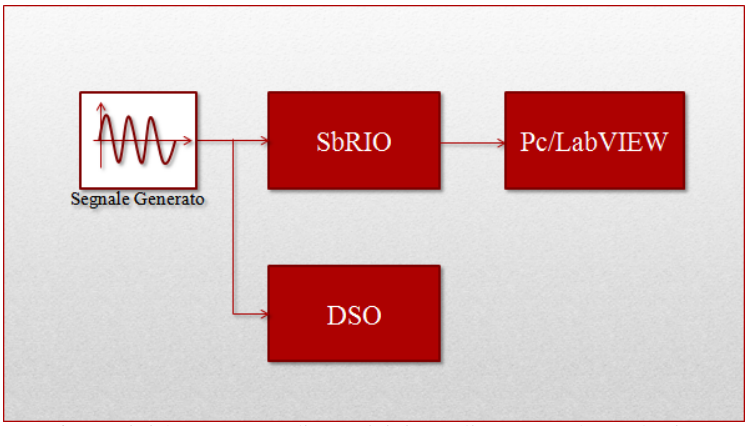


Figura 2.21 – Start-up di acquisizione di un segnale analogico

Per creare un nuovo progetto di acquisizione del segnale in LabVIEW si eseguono i seguenti passi:

- si apre LabVIEW e si selezione New Project → LabView FPGA Project;
- si apre una finestra in cui si deve selezionare il dispositivo. Si seleziona Single-Board RIO Embedded System;
- nella finestra successiva selezionare Discover Existing System. Nella finestra successiva LabView riconosce il dispositivo in Real Time Single Board Rio, si clicca next per proseguire;
- la finestra successiva fornisce un'anteprima del progetto in cui si vede il nome della scheda sbRIO9636 e il suo indirizzo IP. Cliccando su finish si aprira il progetto come mostra la Figura 2.22.

Nel progetto si distinguono due ambienti di lavoro:

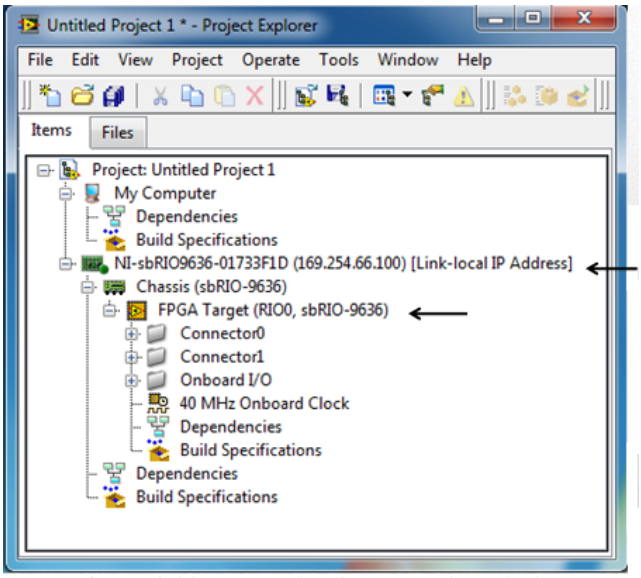


Figura 2.22 – Esempio di progetto in LabView

- Ambiente Real-Time (NI-sbRIO9636-01733F1D in Figura 2.22);
- Ambiente FPGA (FPGA Target Figura 2.22).

2.5.1 VI in ambiente FPGA

Come prima cosa si deve creare il Block Diagram del VI in ambiente FPGA che sia in grado di acquisire un segnale analogico per poi usare tale VI nel ambiente Real-Time.

- si clicca con il tasto destro su FPGA Target e facciamo New → VI e si aprirà il modulo FPGA dell'ambiente di lavoro LabVIEW;
- si crea il Block Diagram come mostra la Figura 2.23. All'interno del ciclo while è presente una sequenze con due passi. Il primo permette all'utente di determinare un ritardo sul periodo di ripetizione del ciclo while impostando il tempo di acquisizione del segnale. Il secondo passo acquisisce il segnale attraverso il nodo AI1 il quale è collegato al pin AI1 scelto come ingresso analogico della scheda. Il ciclo while termina se ci sono eventuali errori in fase di acquisizione del segnale oppure viene fermato manualmente dall'utente tramite il controllo booleano di Stop;
- una volta creato il VI, cliccando su Run, LabVIEW chiederà di compilare. Nella operazione di compilazione, che richiede parecchi minuti, il compilatore programma

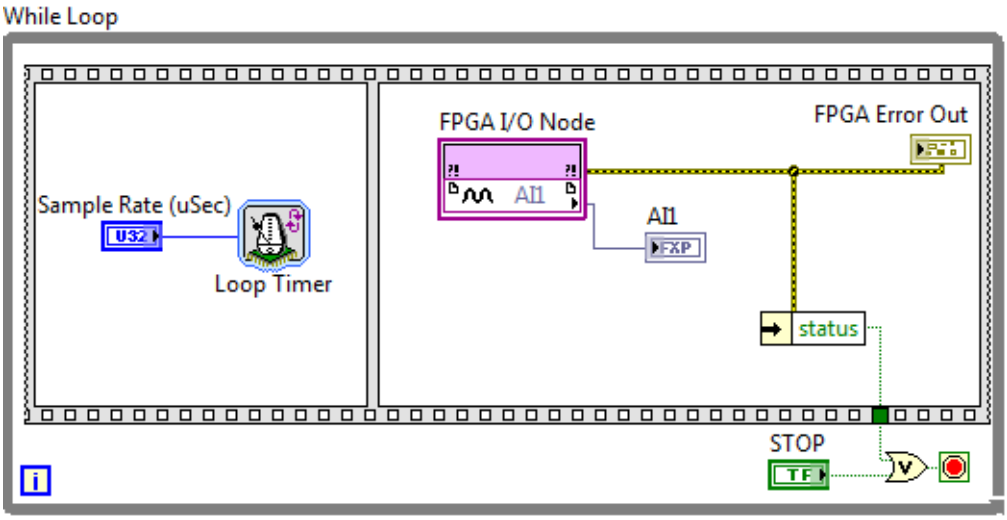


Figura 2.23 – VI di acquisizione in ambiente FPGA

il componente FPGA interno alla scheda, traducendo prima il programma scritto in labVIEW in VHDL.

La Figura 2.24 mostra il panello frontale del Block Diagram creato in cui si può notare il Sample Rate con cui l'utente può gestire il periodo di acquisizione del segnale e il pulsante di Stop con il quale l'utilizzatore può fermare l'acquisizione manualmente.

Figura 2.24 – Panello frontale in ambiente FPGA

2.5.2 VI in ambiente Real-Time

In ambiente Real Time si realizza il VI da usare per visualizzare e manipolare il segnale tramite PC.

- dalla finestra di Project Explorer (Figura 2.22)si clicca con il tasto destro su NI-sbRIO9636-01733F1D e si faccia New → VI, si aprirà l'ambiente di lavoro LabVIEW;

- si crea il VI come mostra la Figura 2.25, utilizzando l'interfaccia FPGA presente nell'elenco delle funzioni;

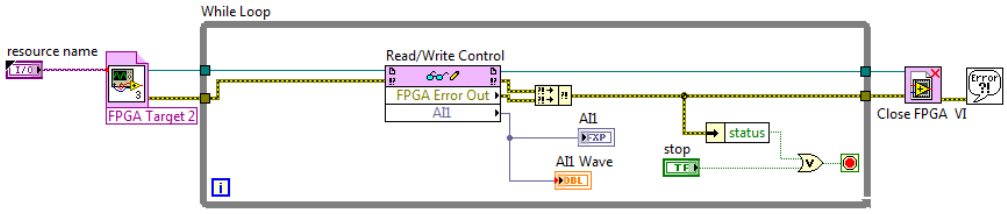


Figura 2.25 – VI di acquisizione in ambiente real-time

- il *Source name* indica la sorgente del segnale (in questo caso la sbRIO). L'icona *FPGA Target* è un riferimento al VI creato in ambiente FPGA . Schiacciando con il tasto destro, sopra l'icona *FPGA Target* si apre una finestra di configurazione con cui si può scegliere il VI realizzato in FPGA da utilizzare. Il segnale entra all'interno del ciclo while e viene letto da un *nodo*, che riprende gli indicatori e i controllori del VI in FPGA che si sta usando,il quale ha come uscite il dato acquisito e l'errore (eventuali errori in fase di acquisizione). Il dato infine, viene visualizzato su un grafico. Fuori dal ciclo while si chiude l'FPGA Target e si gestisce un eventuale errore, nel caso di Figura 2.25, con un messaggio all'utente.

La Figura 2.26 mostra il panello frontale del VI in Real Time. Nell'esempio segnale è una sinusoide con ampiezza 1 Vpp e frequenza impostata dall'utente. Il panello rappresenta l'interfaccia utente con cui l'utilizzatore interagisce con l'applicazione realizzata in LabVIEW e con il dispositivo. Tramite l'icona *resource name* si può selezionare la sorgente del segnale, cliccandoci sopra infatti, si apre una finestra con elencato i dispositivi disponibili per l'acquisizione. Attraverso il pulsante di STOP si può terminare l'esecuzione del VI mentre l'indicatore AI1 mostra il valore del dato che viene acquisito in quel istante.

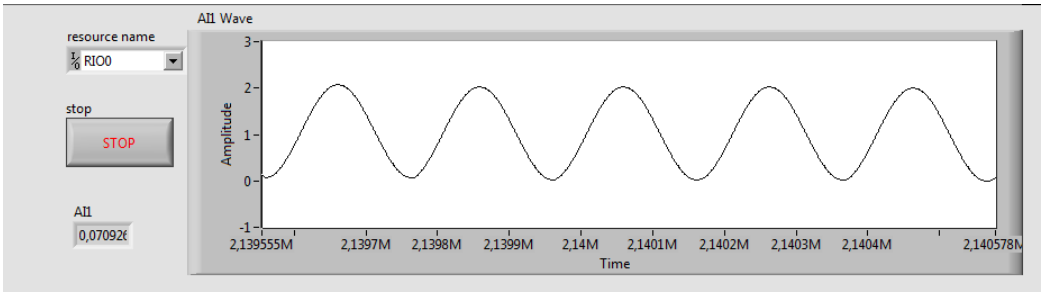


Figura 2.26 – Esempio di una sinusoide acquisita

Sul progetto finale dovrebbero apparire i due VI creati come mostra la Figura 2.27. Il VI in ambiente Real Time deve fare parte della voce NI-sbRIO9636-01733F1D mentre il VI in ambiente FPGA deve apparire all'interno della FPGA Target.

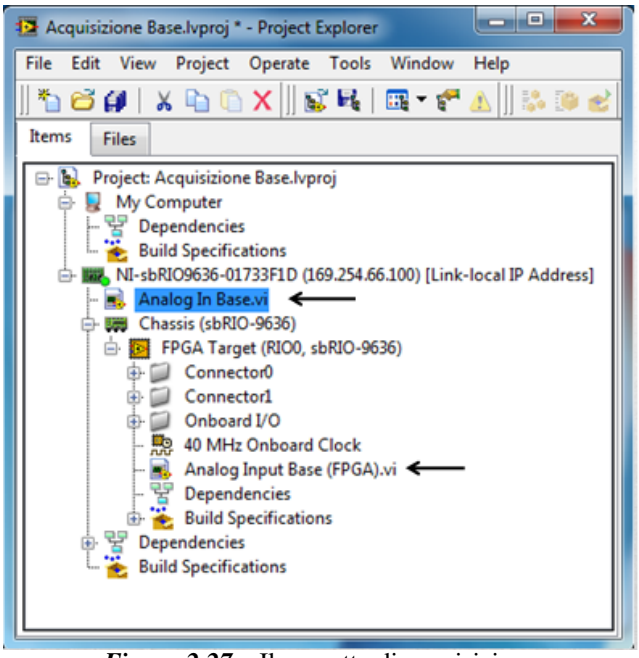


Figura 2.27 – Il progetto di acquisizione

Capitolo 3

Teoria dei Segnali ed Interpolazione Spettrale

Un segnale fisico [5] è una grandezza fisica che varia nel tempo (o nello spazio) e che fornisce informazioni su un aspetto qualsiasi del mondo reale. In generale esistono diversi tipi di segnali, di cui molti dei quali sono accomunati dall'essere in natura segnali casuali e continui. Nelle telecomunicazione i segnali contenenti l'informazione sono segnali aleatori, quelli il cui valore non è prevedibile poiché varia in maniera casuale nel tempo ma su cui è possibile ottenere soltanto delle proprietà statistiche. Nello studio dei segnali, quelli periodici possono essere trattati mediante l'astrazione in uno spazio vettoriale lineare tramite l'utilizzo della serie di Fourier, quelli non periodici invece, mediante la trasformata di Fourier [5]. Attualmente esistono diversi metodi di analisi spettrale dei segnali, argomento di grande studio negli ultimi tempi in quanto interessa la realizzabilità fisica della trasformazione dal tempo alla frequenza. La difficoltà principale è quella di conoscere tutto il dominio del segnale, richiesta dalla teoria, ma di difficile realizzazione nelle applicazioni reali. Si sono cosi studiati metodi in grado di far fronte a questo problema. Uno di questi è l'Interpolazione Spettrale.

3.1 Cenni base

I segnali [5] in generale hanno un dominio di definizione continuo, $\mathbb{R}$, per questo vengono chiamati "segnali a tempo continuo" o anche "segnali continui" come mostra la Figura 3.1 e cosi definiti:

$$s : \mathbb{R} \to \mathbb{C} \quad t \in \mathbb{R} \quad e\ si\ scrive \quad s(t), t \in \mathbb{R} \tag{3.1}$$

in cui il codominio è il campo dei complessi $\mathbb{C}$.

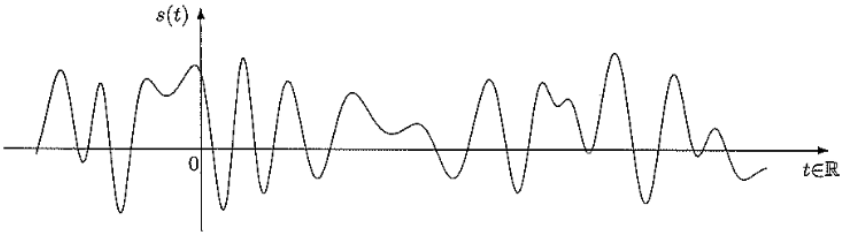


Figura 3.1 – Esempio del andamento di un segnale continuo [5]

Nella maggior parte dei casi succede che non è necessario conoscere l'andamento dettagliato del segnale ad ogni istante ma è sufficiente conoscerlo a intervalli regolari di tempo. In questi casi il dominio del segnale a tempo continuo, viene ristretto a un sottoinsieme discreto costituito da un insieme infinito ma numerabile di punti. Questa operazione prende il nome di campionamento. Esistono però anche casi in cui il segnale nasce limitato a un insieme di istanti separati da intervalli temporali regolari. Si definiscono in questo caso i "segnali a tempo discreto" in cui il dominio è un sottoinsieme dell'asse $\mathbb{R}$, costituito da istanti equispaziati di una quantità T > 0. Tali istanti, essendo multipli di T, possono essere messi in corrispondenza biunivoca con l'insieme $\mathbb{Z}$ dei numeri interi come mostra la Figura 3.2 e cosi definiti:

$$s : \mathbb{Z}(T) \to \mathbb{C} \quad t \in \mathbb{Z}(T) \quad e\ si\ scrive \quad s(t), t \in \mathbb{Z}(T) \tag{3.2}$$

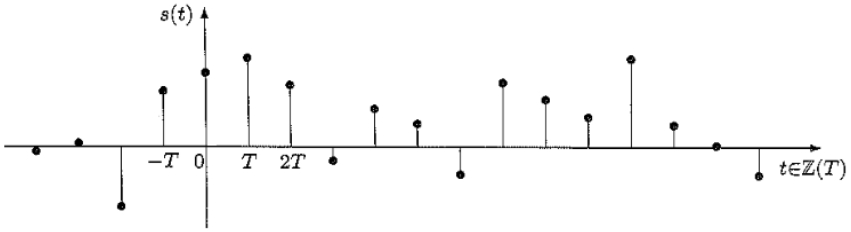


Figura 3.2 – Esempio del andamento di un segnale discreto [5]

Segnali periodici

Rilevante importanza ha la distinzione tra segnali periodici e aperiodici [5]. Un segnale continuo si dice periodico (Figura 3.3) se esiste un numero reale $T_p > 0$ tale che:

$$s(t + T_p) = s(t) \quad t \in \mathbb{R} \tag{3.3}$$

la quantità T_p è detta periodo del segnale. Questo implica che il segnale si ripete costantemente per multipli interi del suo periodo per cui si può ricavare l'andamento del segnale stesso tramite un suo periodo partendo da un istante t_0 arbitrario. Si ha:

$$s(t), \qquad s \in [t_0, t_0 + T_p] \tag{3.4}$$

Se un segnale continuo non presenta nessuna periodicità nel sua andamento temporale allora si dice che è aperiodico (Figura 3.3). Possiamo estendere il concetto di segnale periodico e aperiodico anche ai segnali discreti. Un segnale discreto si dice periodico (Figura 3.3) se esiste un numero reale T_p = NT, con N numero intero positivo, tale che:

$$s(t + NT) = s(t) \quad t, NT \in \mathbb{Z}(T) \tag{3.5}$$

Come nel caso dei segnali continui anche in questo caso si può dire che il segnale si ripete periodicamente nel tempo con periodo T_p = NT partendo da un intero k_0 arbitrario. Si ha:

$$s(k_0T), ((k_0 + 1)T), ..., s((k_0 + N - 1)T) \tag{3.6}$$

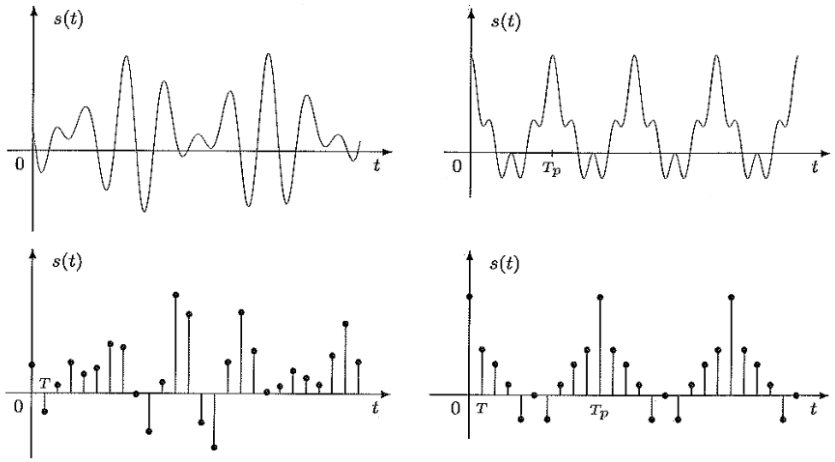


Figura 3.3 – Esempio del andamento di un segnale continuo aperiodico (periodico) e discreto aperiodico (periodico) [5]

Rappresentazione complessa

La teoria dei segnali opera però con segnali complessi, per cui si rende necessario dare una definizione del segnale in campo complesso:

$$s(t) = \Re s(t) + i\Im s(t) \tag{3.7}$$

in cui il segnale si scrive come la somma della sua componente reale e della sua componete immaginaria in campo complesso. Una rappresentazione alternativa del segnale s(t) in campo complesso è data mediante il modulo $|s(t)|$ e la fase φ_s(t):

$$s(t) = |s(t)|e^{i\varphi_s(t)} \tag{3.8}$$

Le due rappresentazioni complesse sono tra loro legate grazie alle formule di Eulero [5]:

$$s(t) = |s(t)|cos\varphi_s(t) + i|s(t)|sin\varphi_s(t) \tag{3.9}$$

da cui si ottiene:

$$\Re s(t) = |s(t)|cos\varphi_s(t) \qquad \Im s(t) = |s(t)|sin\varphi_s(t) \tag{3.10}$$

Inversamente si ha:

$$|s(t)| = \sqrt{(\Re s(t))^2 + (\Im s(t))^2} \tag{3.11}$$

3.2 Trasformata di Fourier

La trasformata di Fourier permette, nell'analisi di un segnale, di passare dal dominio del tempo al dominio della frequenza. Il vasto utilizzo di Fourier in ambito della teoria dei segnali e dovuta al fatto che permette di scomporre e successivamente ricombinare, tramite la formula inversa di antitrasformata, un segnale generico in una somma infinita di sinusoidi. Consideriamo un segnale continuo nel tempo, ad esempio una sinusoide del tipo:

$$s(t) = A_0 sin(2\pi f_0 t + \varphi_0) \tag{3.12}$$

La quale ha un andamento come quella di Figura 3.4

Applicando la definizione di trasformata di Fourier [5] si ha:

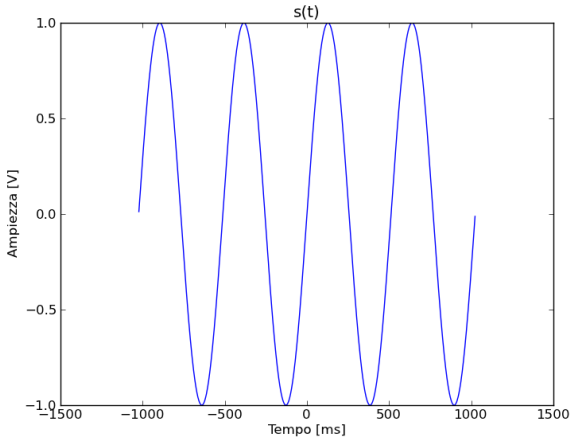


Figura 3.4 – Esempio del andamento di un segnale sinusoidale

$$\begin{aligned} S(f) &= \int_{-\infty}^{+\infty} s(t)e^{-j2\pi ft}\,dt \\ &= -\frac{j}{2}\delta(f-f_0)e^{j\varphi_0} + \frac{j}{2}\delta(f-f_0)e^{-j\varphi_0} \end{aligned} \tag{3.13}$$

dove δ si è ottenuto dalla seguente condizione di ortogonalità [5]:

$$\int_{-\infty}^{+\infty} e^{j2\pi ft}e^{-j2\pi\lambda t}\,dt = \int_{-\infty}^{+\infty} e^{j2\pi(f-\lambda)t} = \delta(f-\lambda) \tag{3.14}$$

Il risultato è la funzione in frequenza S(f) a valori complessi. Di questa funzione si può ricavare il modulo e la fase come mostra la Figura 3.5.

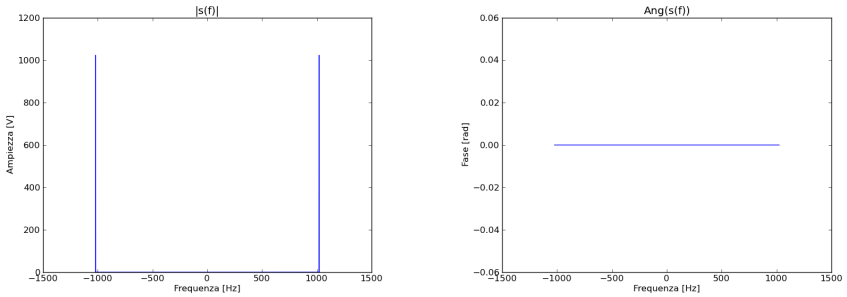


Figura 3.5 – Grafico del modulo e della fase del segnale sinusoidale preso in considerazione

Nelle applicazioni reali e nei dispositivi che vengono utilizzati per l'acquisizione dei segnali viene imposto un tempo minimo tra un campione ed il successivo per cui impossibile analizzare un segnale continuo ma si deve analizzare un segnale campionato nel tempo, per cui si ha:

$$s(kT_p) = A_0 sin(2\pi f_0 kT_p + \varphi_0) \tag{3.15}$$

dove k è un numero intero e T_p è il passo di campionamento, cioè l'inverso della frequenza di campionamento F_p la quale deve sempre rispettare il teorema di Nyquist il quale afferma che: si è in grado di ricostruire il segnale originale solo nel caso in cui il campionamento sia effettuato con una frequenza doppia della frequenza massima del segnale che si vuole osservare. Dopo aver effettuato un operazione di campionamento il segnale non è più continuo quindi invece di usare la trasformata di Fourier si utilizza la DTFT che è la corrispettiva della trasformata continua di Fourier nel dominio discreto illimitato cosi definita:

$$\begin{aligned} S(f) &= \sum_{k=-\infty}^{k=+\infty} T_p s(kT_p) e^{-j2\pi f k T_p} \\ &= -\frac{j}{2}\delta(f - f_0)e^{j\varphi_0} + \frac{j}{2}\delta(f - f_0)e^{-j\varphi_0} \end{aligned} \tag{3.16}$$

Come si può notare il risultato è lo stesso ottenuto nel caso in cui il segnale è continuo. Quanto detto fino ad adesso succede nel caso ideale in cui il modulo di S(f) costituisce uno spettro ideale del segnale. In questo caso per stimare la frequenza precisa delle armoniche basta trovare il massimo sulla rappresentazione del modulo e poi in base alla frequenza dell'armonica considerata si leggono i valori di modulo e fase. In questo modo i valori ottenuti sono da considerarsi corretti. Ma come detto precedentemente questo succede nel caso ideale. Nel caso reale invece non è possibile acquisire un numero infinito di campioni, per cui ci si deve limitare ad un numero finito di N campioni su cui effettuare un operazione di trasformata. A questo proposito si definisce la DFT che è la corrispettiva della trasformata continua di Fourier nel dominio discreto limitato definita come:

$$S(nF) = \sum_{k=k_0}^{k_0+N-1} T_p s(kT_p) e^{-j2\pi k T_p n F} \tag{3.17}$$

in cui k_0 è l'istante iniziale di campionamento. Valgono per cui le relazioni di reciprocità le quali mettono in relazione il dominio della frequenza con il dominio del tempo in cui la frequenza di campionamento è data da $F_p = \frac{1}{T_p}$ e deve sempre rispettare il teorema di Nyquist. F è la risoluzione dello spettro definita come $F = \frac{F_p}{N} = \frac{1}{T_p N}$. Considerando che la DFT lavora su un numero finito di campioni allora consente l'osservazione del segnale in una finestra limitata della durata di $T_w = \frac{N}{F_p} = NT_p$. La DFT effettua operazioni di calcolo su un numero di campioni $N = 2^b$ con un tempo di calcolo proporzionale a $o(N^2)$. Per

ridurre la complessità temporale della DFT si può utilizzare la FFT la quale, grazie ad una serie di algoritmi veloci è in grado di ridurre il numero di operazioni associato ad un numero di campioni con andamento $N = 2^b$ riducendo la complessità di calcolo proporzionale a $o(Nlog_2N)$ [9]. La FFT è la funzione che verrà usata nella implementazione dell'algoritmo di interpolazione spettrale sul dispositivo di nostro utilizzo.

I passi eseguiti dalla DFT sono principalmente tre: campionamento, finestratura ed estensione per periodicità nel tempo. Il campionamento è il primo passo del processo di DFT in cui il segnale viene acquisito con frequenza costante rispettando le condizioni di non distorsione ed evitando il fenomeno di aliasing [5] campionando con una frequenza almeno il doppio della frequenza massima che si vuole osservare nello spettro. Il segnale in uscita è in questo caso discretizzato nel tempo dal quale verranno acquisti gli N campioni da utilizzare nella DFT. Gli N campioni in questo caso vengono selezionati tramite una funzione finestra $w(kT_p)$ cosi definita:

$$w(kT_p) = \begin{cases} 1 & n_0 < k < n_0 + N - 1 \\ 0 & altrove \end{cases} \tag{3.18}$$

il risultato della moltiplicazione con la funzione finestra sarà un segnale uguale a quello discretizzato all'interno del periodo di acquisizione e zero altrove. La moltiplicazione nel tempo equivale ad una convoluzione in frequenza tra la funzione S(f) e lo spettro della funzione finestra W(f) che diventa:

$$W(nF) = \sum T_w w(kT_p) e^{-j2\pi kT_p nF} = T_w sinc(n) e^{-j\pi n} \tag{3.19}$$

Infine il terzo e ultimo passo è l'estensione per periodicità nel tempo dei campioni acquisiti nell'intervallo T_w il quale causa la comparsa di massimi secondari nella rappresentazione dello spettro del segnale. Dopo aver applicato il processo di DFT alla sinusoide presa in esame si ha che:

$$S(nF) = -\frac{jA_0}{2} e^{j\varphi_0} W(nF - f_0) + \frac{jA_0}{2} e^{-j\varphi_0} W(nF + f_0) \tag{3.20}$$

la quale si differenzia dal caso ideale per la presenza della funzione W(nF). Il problema principale di questa procedura è l'estensione per periodicità, in quanto se il campionamento non è eseguito in modo corretto porta alla presenza di distorsioni nella spettro con sovrapposizioni della stesso dando origine ad armoniche che non sono riconducibili al segnale di partenza. Questo fenomeno si distingue in campionamento coerente e campionamento incoerente. Il campionamento coerente [4] si ha quando in corrispondenza di un segnale

di ingresso periodico la durata dell'intervallo di acquisizione è esattamente al periodo del segnale o ad un suo multiplo:

$$f_0 = l_0 F \quad con\ l_0\ intero\ diverso\ da\ 0\ ed\ F\ risoluzionespettrale \tag{3.21}$$

sostituendo nella 3.20 si ha:

$$S(nF) = -\frac{jA_0}{2} e^{j\varphi_0} W[(k - l_0)F] + \frac{jA_0}{2} e^{-j\varphi_0} W[(k + l_0)F] \tag{3.22}$$

Con riferimento alla Figura 3.6 si osserva che in queste condizioni i punti campionati coincidono con i zeri della funzione finestra ad eccezione della frequenza in corrispondenza della quale il campione ha ampiezza massima. In questo modo si è evitata la distorsione introdotta dovuta al finestra di osservazione limitata. Dalla misurazione del campione non nullo visualizzato è possibile risalire al valore esatto dell'ampiezza e della frequenza esatta del campione.

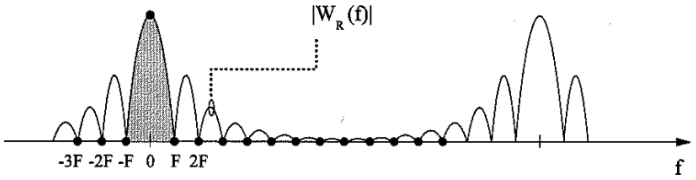


Figura 3.6 – Esempio di campionamento coerente [4]

Nelle applicazioni reali però si ottiene sempre campionamento incoerente il quale è cosi definito:

$$f_0 = \lambda_0 F = (l_0 + \delta_0)F \quad con\ l_0\ intero\ diverso\ da\ 0, |\delta_0| \leq \frac{1}{2} \tag{3.23}$$

sostituendo nella 3.20 si ha:

$$S(nF) = -\frac{jA_0}{2} e^{j\varphi_0} W[(k - l_0 - \delta_0)F] + \frac{jA_0}{2} e^{-j\varphi_0} W[(k + l_0 + \delta_0)F] \tag{3.24}$$

Con riferimento alla Figura 3.7 si può notare come i punti di campionamento non coincidono con la funzione di finestratura causando un deterioramento della traccia spettrale detta anche spectral leakage definito nel paragrafo successivo.

3.2.1 Finestratura

La dispersione spettrale o spectral leakeage [4] e un fenomeno di distorsione che ha origine da un operazione di DFT effettuata in condizioni di campionamento non coerente. Si ma-

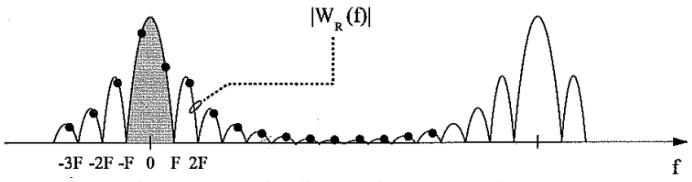


Figura 3.7 – Esempio di campionamento incoerente [4]

nifesta come una dispersione dell'energia sulla traccia spettrale ottenuta. Un modo efficace di ridurre gli effetti della dispersione spettrale, consiste nell'applicare alla sequenza di campioni del segnale acquisito appena prima dell'operazione di DFT, una opportuna funzione di pesatura detta funzione finestra. Tale funzione e costruita in modo da smorzare gradualmente i campioni all'inizio e alla fine dell'intervallo di acquisizione come mostra la Figura 3.8.

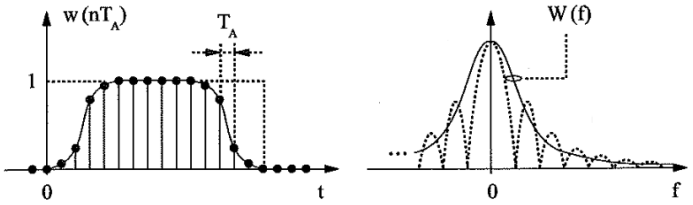


Figura 3.8 – Esempio di una finestratura e della sua trasformata fi Fourier [4]

Le finestre presentano spettri abbastanza simili tra loro, e sempre presente un lobo principale centrato nell'origine di ampiezza molto maggiore rispetto al resto dello spettro. In base alla struttura dell'equazione che genera la finestra, sono presenti una successione di lobi laterali con ampiezze decrescenti con l'aumentare della distanza dal lobo principale come mostrato in Figura 3.9 in cui vengono evidenziate anche le caratteristiche principale di una funzione finestra quali la larghezza del suo lobo principale indicata con B_6, la velocità di decadimento dei lobi secondari r_L e l'altezza massima dei lobi laterali $\triangle_L$.

L'utilizzo della tipologia delle finestra e molto studiato e di grande interesse. Ciascuna funzione presenta una specifica finalità di misurazione, per esempio alcune finestre presentano lobi principali molto stretti che ben si adattano per la stima di frequenza, altre invece presentano un'attenuazione molto elevata tra il lobo principale e il lobo secondario per eliminare il più possibile l'interferenza con componenti vicine. Molte di esse fanno parte delle finestre cosinusoidali che presentano l'andamento in (3.26)le quali sono di nostro interesse in quanto permettono di stimare con semplicità i parametri spettrali grazie all'espressione maneggevole del loro spettro.

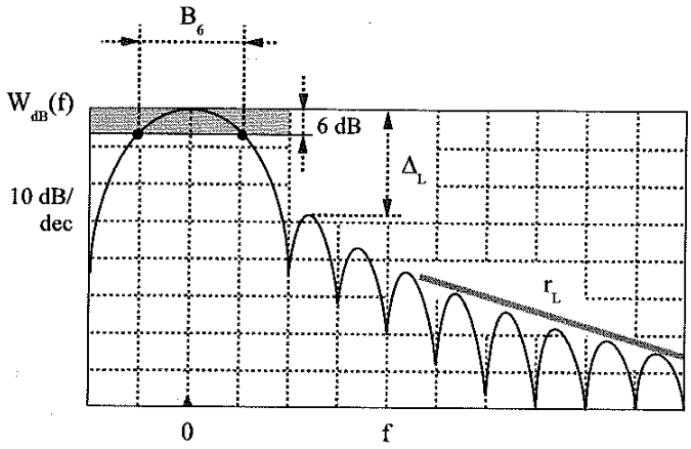


Figura 3.9 – Esempio di una funzione finestra [4]

$$w(nk_p) = \begin{cases} \sum_{l=0}^{H-1} a_l cos\left(\frac{2\pi klT_P}{N}\right) & |k| \leq \frac{N}{2} \\ 0 & |k| \geq \frac{N}{2} \end{cases} \tag{3.25}$$

dove N e il numero di campioni su cui applicare la finestra, H e il numero di termini che vengono combinati e a_l e la serie di coefficienti reali per scalare le componenti. La spettro delle finestre cosinusoidali è calcolabile in forma chiusa e vale [1]:

$$\begin{aligned} W(f) = sin(\pi f)e^{-j\pi f}e^{j\frac{\pi}{N}f}\sum_{l=0}^{H-1}(-1)^l\frac{a_l}{2} \\ \left[\frac{e^{-j\frac{\pi}{N}f}}{sin(\frac{\pi}{N}(f-l))} + \frac{e^{j\frac{\pi}{N}f}}{sin(\frac{\pi}{N}(f+l))}\right] \quad f \in [0, N) \end{aligned} \tag{3.26}$$

Per le analisi del segnale si e scelto di utilizzare le finestra di "MSLD Hanning". Con l'acronimo MSLD si indica la tipologia di finestre a massimo decadimento dei lobi laterali. L'utilizzo delle finestre MSLD è dovuto all'espressione matematica del loro spettro dove la (3.26) può essere semplificata [1] ottenendo una formula gestibile analiticamente:

$$W(f) \simeq \frac{Nsin(\pi f)}{2^{2H-2}\pi f}e^{-j\pi f}e^{j\frac{\pi}{N}f}\frac{(2H-2)!}{\prod_{q=1}^{H-1}(q^2-f^2)} \tag{3.27}$$

3.3 Interpolazione Spettrale

Il principio di funzionamento che sta alla base dell'interpolazione spettrale è quello di utilizzare la conoscenza del comportamento della funzione per ricostruire l'andamento dello spettro stesso. L'analisi in questo caso è composto principalmente sul calcolo della tra-

sformata e sulla stima dei parametri dello spettro che consiste nella stima di δ_m presente nella formula 3.24. Il calcolo della trasformata è stato analizzato nei paragrafi precedenti di seguito vedremo brevemente la stima dei parametri usando il metodo che si basa sulla conoscenza del comportamento della finestra [1]. La funzione interpolatrice in questo metodo è lo spettro della finestra W(f) e gli n-punti da interpolare sono i punti della FFT. In questo modo, partendo dai punti elaborati dalla FFT, si ha uno spettro continuo su cui calcolare δ_m. La funzione interpolatrice viene semplificata per rendere una espressione matematica semplice la quale viene approssimata ad un polinomio nell'origine del lobo principale. Il numero di punti interno al lobo principale per una finestra cosinusoidale è sempre pari a 2H-1, pertanto l'elaborazione numerica aumenta al crescere di H ma aumenta anche la precisione dell'interpolazione. Il caso più semplice da analizzare è l'interpolazione a due punti in cui si utilizza una funzione finestra con una banda maggiore di 2bin. Con riferimento alla Figura 3.10 si calcola l'ampiezza massima al lobo principale, indicando con i_0 l'indice e A_{i0} l'ampiezza. In seguito si confrontano i punti di indice adiacenti ad i_0 e si considera il maggiore dei due che si indica con i_1 e A_{i1} .

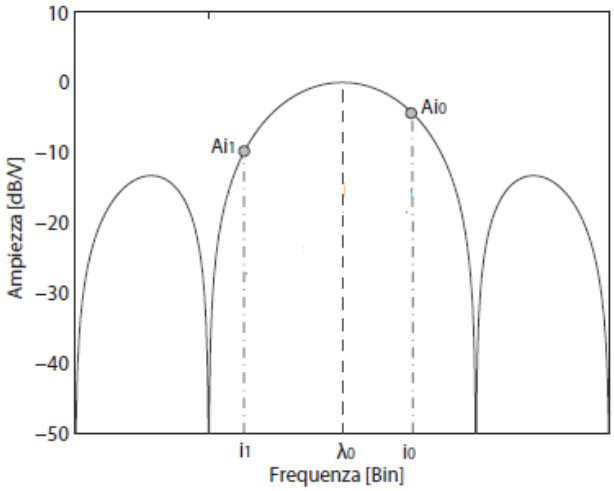


Figura 3.10 – Esempio di metodo di interpolazione spettrale

Supponiamo $i_1 = 1_0 - 1$ allora possiamo impostare il seguente sistema di equazioni:

$$\begin{cases} A_{i0} = A_0 W(i_0 - \lambda_0) \\ A_{i1} = A_0 W(i_1 - \lambda_0) \end{cases} \tag{3.28}$$

è un sistema di due equazioni indipendenti in due incognite, A_0 e λ_0, per cui ammette soluzione. Considerando la 3.23 si può riscrivere il sistema in funzione di δ_0:

$$\begin{cases} A_{i0} = A_0 W(\delta_0) \\ A_{i1} = A_0 W(\delta_0 - 1) \end{cases} \tag{3.29}$$

Risolvendo il sistema è possibile stimare il valore di δ_0 il cui valore stimato verrà indicato con un cappuccio per essere distinto dal valore ideale:

$$\frac{A_{i0}}{A_{i1}} = \alpha_0 = \frac{W(\delta_0)}{W(\delta_0 - 1)} = \frac{H - 1 + \hat{\delta_0}}{H - \hat{\delta_0}} \tag{3.30}$$

Considerando il caso delle finestre in esame si sostituisce a W(f) la sua espressione matematica si può esplicitare $\hat{\delta_0}$ come:

$$\hat{\delta_0} = \frac{H\alpha_0 - H + 1}{\alpha_0 + 1} \tag{3.31}$$

Con riferimento alla Figura 3.10 il centro della finestra può essere sempre trovato in mezzo ai due valori A_{i0} e A_{i1} e quindi a seconda che il picco secondario sia a sinistra o a destra del del massimo principale, δ_0 assumerà rispettivamente valori negativi o positivi. Si può stimare in questo modo il valore stimato di f_0 e di conseguenza dei restanti parametri:

$$\hat{f_0} = (i_0 \pm \hat{\delta_0})F = \hat{\lambda_0}F \tag{3.32}$$

$$\hat{A_0} = \frac{2}{|W(-\hat{\delta_0})|}|A_{i0}| \tag{3.33}$$

$$\hat{\varphi_0} = arg[A_{i0}] - arg[W(-\hat{\delta_0})] \tag{3.34}$$

Come si può notare la stima dell'ampiezza e della fase dipende dalla stima di δ_0. Maggiore è il numero di punti considerati migliore sarà la stima dei parametri, quindi si può dire che lo stesso ragionamento si può ripetere per n punti. Si tende però a considerare numeri sempre dispari pari a 2H - 1 poiché le finestre cosinusoidali presentano sempre una banda pari a 2H bin, quindi il ragionamento si ripete su 3,5,7...n punti. Vengono riportate di seguito solo le formule per una stima a 3 punti [2] [1].

$$\delta_0 = H\frac{1 - a_0}{1 + a_0} \tag{3.35}$$

in cui:

$$a_0 = \frac{\sum_{i=0}^{J} C_J^{J-i} |S(i_0 - i)|}{\sum_{i=0}^{J} C_J^{J-i} |S(i_0 + i)|} \tag{3.36}$$

$$J = \frac{n-1}{2} \qquad C_m^p = \frac{m!}{(m-p)!p!} \tag{3.37}$$

Capitolo 4

Analisi spettrale in real-time

Lo scopo del lavoro svolto durante questi mesi è stato quello di implementare sulla FPGA del dispositivo sbRIO 9636 il calcolo in real-time della trasformata discreta di Fourier tramite l'algoritmo di FFT. Ciò permette lo studio dei segnali, acquisiti nel dominio del tempo, tramite una rappresentazione nel dominio della frequenza. Tale calcolo fornisce una forma alternativa che permette di analizzare i fattori componenti il segnale stesso.

4.1 Acquisizione ed FFT

L'applicativo sviluppato si compone di acquisizione del segnale ed elaborazione su FPGA per fornire la corrispondente visualizzazione dello spettro del segnale in Real Time (PC). In ambiente FPGA si è dato importanza ai tempi di calcolo della FFT con relativo visualizzazione su DSO della tempistica di elaborazione, alle prestazioni e alla efficienza del dispositivo. Di seguito si descrivono i VI realizzati in accordo alle fasi di sviluppo presentate nel Paragrafo 2.4.

4.1.1 Allestimento sperimentale

Con riferimento alla Figura 4.1 il banco di test è costituito da un generatore di segnali collegato tramite un cavo coassiale ad un ingesso analogico della scheda sbRIO la quale a sua volta è collegata tramite cavo dati ethernet al PC, con il quale verrà controllata la parte real-time del progetto. Il dispositivo, inoltre, è collegato tramite cavo coassiale ad un DSO in cui viene visualizzato l'andamento di un uscita digitale.

Il generatore fornisce il segnale da analizzare e lo invia al dispositivo sbRIO il quale esegue le operazioni di acquisizione, elaborazione e di comunicazione con il PC trasmettendo

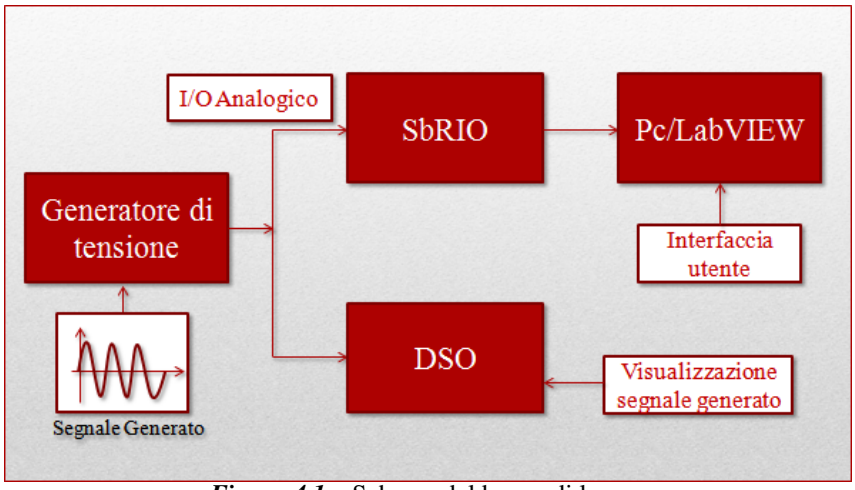


Figura 4.1 – Schema del banco di lavoro

i dati elaborati. Il PC permette l'interazione con l'operatore tramite il Panello Frontale del progetto.

4.1.2 VI FFT in FPGA

La Figura 4.2 riassume brevemente il VI FFT utilizzato per il calcolo della trasformata di Fourier tramite FPGA:

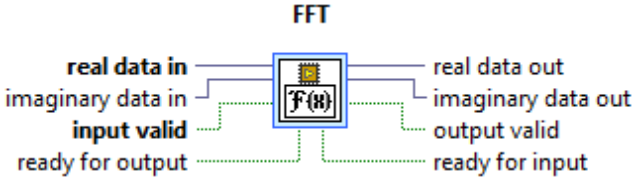


Figura 4.2 – La funzione FFT

Nella Figura 4.2, gli elementi di I/O hanno i seguenti significati:

- real data in: è il dato in ingresso su cui verrà effettuato il calcolo della FFT;
- input valid: specifica, con un valore booleano, al VI FFT se il dato in ingresso è stato processato oppure no in modo tale da permettere al VI di ricevere un altro valore;
- ready for output: invia in uscita un valore di True/False per comunicare che il VI FFT ha fatto il calcolo della trasformata di Fourier ed è pronto per inviare il dato in uscita;
- ready for input: invia in uscita un valore booleano per comunicare che il VI FFT è pronto per ricever dati in ingresso;

- output valid: esprime, con un valore booleano, se il dato trasmesso dal VI FFT è stato processato correttamente oppure no. In tal modo si specifica se può essere utilizzato dai nodi successivi per l'elaborazione;

- imaginary data out: parte immaginaria della FFT calcolata;

- real data out: parte reale della FFT calcolata;

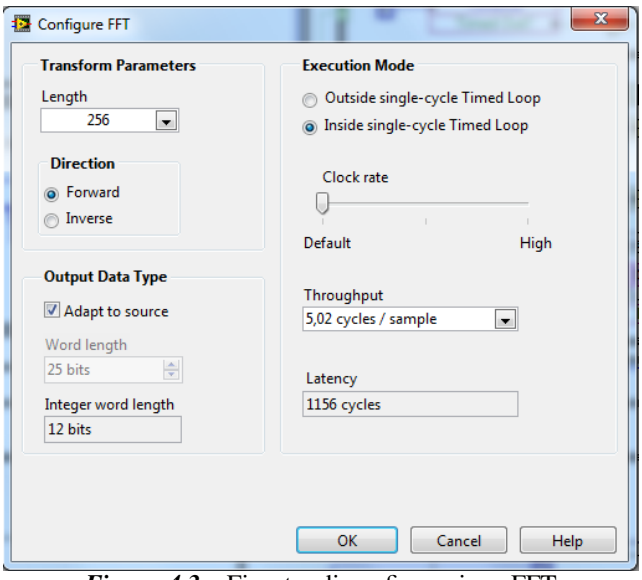


Figura 4.3 – Finestra di configurazione FFT

Facendo un doppio clic sul VI FFT si apre una finestra di configurazione come mostra la Figura 4.3 in cui si può notare la voce *Length* che permette di impostare il numero di campioni su cui verrà calcolata la trasformata di Fourier, in questo caso si esegue un calcolo della FFT su 256 campioni. Nella voce *Execution Mode* si può scegliere il metodo di esecuzione del VI FFT in questo caso impostata come interno a un singolo ciclo del while temporizzato, ciò vuol dire che il calcolo della FFT viene eseguito con un ciclo di clock del dispositivo. La voce *Throughput* da informazioni sul tempo di calcolo con cui il dispositivo esegue la trasformata di Fourier mentre la voce *Latency* indica il tempo che la scheda impiega per fare l'acquisizione dei primi 256 campioni per poi dare in uscita il primo valore valido di FFT.

4.1.3 VI in ambiente FPGA

Diagramma a Blocchi del VI in FPGA

Facendo riferimento alla Figura 4.12 di pag. 56, si possono osservare due blocchi principali costituiti da due cicli while i quali lavorano in parallelo. Il primo è un ciclo *While* semplice costituito da una sequenza di due parti mentre il secondo è un *While Timed Loop*, il quale esegue il blocco al suo interno con cicli temporizzati eseguendo un ciclo in un tempo di clock della scheda. Di seguito si spiegano le varie componenti del VI in cui l'elenco numerato si riferisci ai corrispettivi numeri presenti nelle Figure 4.4 e 4.5

1 Il punto "1." consiste nel controllo di una uscita digitale la quale viene abilitata appena il VI comincia la sua esecuzione. Il suo controllo si è reso necessario per fare in modo di visualizzare, grazie ad un DSO, la tempistica di calcolo della FFT da parte della FPGA.

2 Nel punto "2." la prima parte della sequenza nel ciclo while acquisisce il segnale analogico attraverso il nodo AI1 il quale è collegato al pin AI1 dell'ingresso analogico della scheda e lo manda ad un *FIFO Method Node* il quale agisce nella coda FIFO di nome *Wave*. Il nodo invoca il metodo *write* il quale esegue un operazione di scrittura su *Element*. La voce *Timeout* indica il tempo di attesa che la coda deve aspettare prima di terminare la sua funzione di scrittura in questo caso è collegato la costante zero, in quanto la coda deve essere in scrittura continua poiché l'acquisizione è in tempo reale. La voce *Timed Out?* chiede continuamente se il tempo di attesa è trascorso. Questo vuol dire che se il valore di *Timed Out?* è maggiore di *Timeout* la coda termina l'esecuzione dell'operazione di scrittura. In questo caso è collegato all'uscita dal ciclo while e fa in modo che se non arriva nessun dato da scrivere il ciclo while termina non facendo niente. Prima che i valori acquisiti siano inviati alla coda Wave, essi vengono finestrati in base alla finestra selezionata, nel nostro caso la scelta e tra quella di *Hanning* e quella di *Backman-Harris*. Tale operazione è utile quando si va ad implementare l'elaborazione del metodo si interpolazione.

3 Nel punto "3." viene gestito il periodo di campionamento del segnale in tick, cioè in cicli di clock del dispositivo, indicando con un valore booleano quando questo supera il valore inserito dall'utente. La frequenza di campionamento viene inserita manualmente dall'utilizzatore attraverso il controllore *Sample Rate*. Il *Loop Timer* restituisce in uscita i tick trascorsi tra un ciclo e l'altro, quindi la differenza di due valori in uscita dovrebbe essere uguale a quella inserita dall'utente inizialmente. La differenza tra il

valore precedente in uscita dal *Loop Timer*, tenuto in memoria dal *Feedback Node*, e l'attuale in uscita viene continuamente confrontato con il tempo di campionamento inserito. Se i due valori non coincidono allora il valore booleano assume il valore di vero indicando che il tempo di campionamento inserito non rispetta quello effettivo.

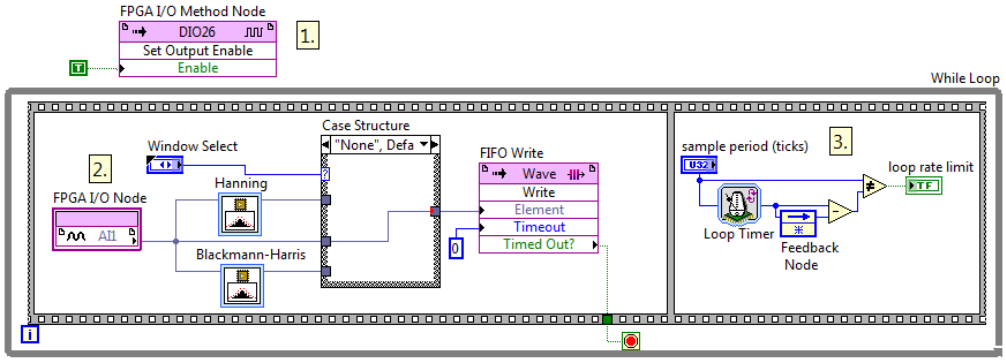


Figura 4.4 – VI creato in ambiente FPGA, cilco while

4 - 5 Il punto "4." è costituito da una *Struttura Case* che racchiude due casi, uno *False* ed uno *True*. Se la funzione FFT è pronta per ricevere dati, invia alla Struttura Case un valore True attraverso il Feedback Node (punto 5 in Figura 4.5) che controlla l'uscita *ready for input* del VI FFT. In questo caso la coda Wave legge il dato e lo invia alla FFT in caso contrario non fa niente ed il ciclo while termina. Quando l'applicazione viene mandata in acquisizione il VI FFT è pronto per ricevere dati quindi lo stato del Feedback Node cambia subito in True in questo modo vengono letti immediatamente i valori dalla coda Wave.

6 Nel punto "6." i due FIFO Method Node corrispondenti alle code *Real FFT* e *Image FFT* richiamano il metodo delle code *Number of Elements to Write* controllando se nelle code in qui verranno inviati in scrittura la parte reale e la parte immaginaria della FFT si possono scrivere elementi, impostando per entrambe la condizione che il numero di elementi da scrivere sia maggiore di zero. In questo modo, se le due condizioni sono vere entrambe, cioè se il numero di elementi da scrivere nelle rispettive code FIFO e maggiore di zero, viene comunicato alla funzione FFT, impostando l'ingresso *ready for output* a vero, che può inviare elementi in scrittura alle due code Real FFT e Imag FFT. In caso contrario non fa niente aspetta finché non si verifica la condizione di vero.

7 Il punto "7." è costituito da due Strutture Case. La prima racchiude la coda *Signal* la quale, nel caso True scrive gli elementi provenienti dalla coda Wave, ricostruendo il

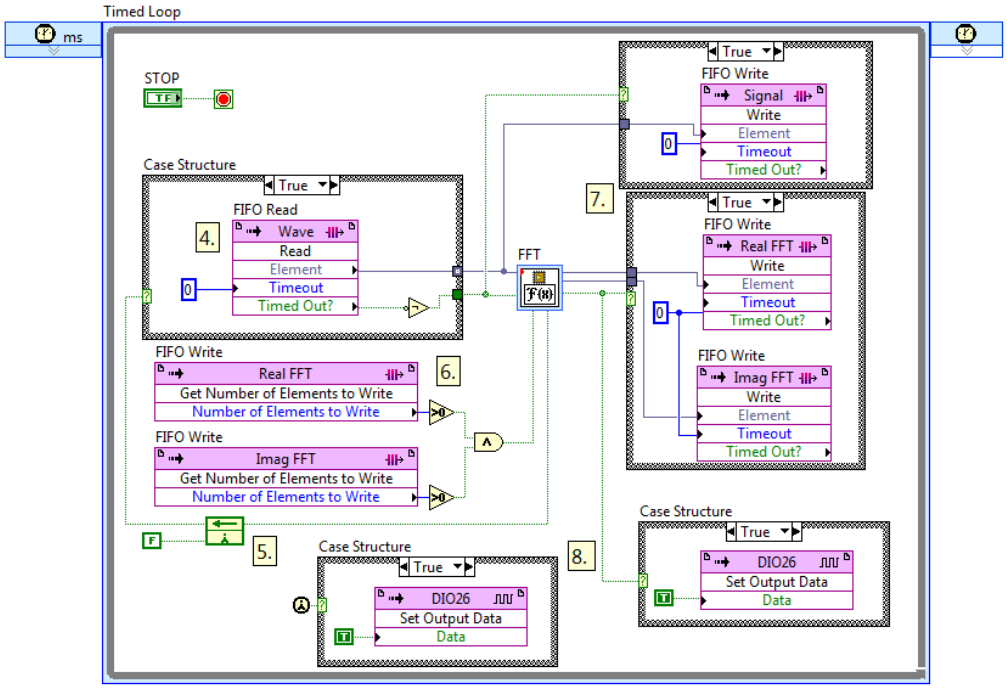


Figura 4.5 – VI creato in ambiente FPGA, cicle while temporizzato

segnale senza elaborazione. Nel caso contrario vuol dire che il valore di *Timed Out?* è maggiore di *Timeout* per cui la coda Wave non sta ricevendo dati, la Struttura Case esegue il caso False non facendo niente. La seconda Struttura Case è costituita da due code Real FFT ed Image FFT le quali scrivono i dati proveniente dalla FFT solamente se i dati sono stati processati correttamente e sono pronti per essere usati dai nodi successivi, altrimenti la struttura case esegue il caso False non facendo niente.

8 L'ultimo punto esegue un controllo sulla uscita digitale che è stata abilitata al punto "1." La Struttura Case a sinistra del punto "8." (Figura 4.5) fa in modo che alla prima esecuzione del cilco while temporizzato l'uscita digitale DIO26 sia impostata alta assegnandoli un valore vero. La Struttura Case a destra del punto "8." fa in modo che il valore dell'uscita digitale cambi in True/False in base alla uscita *output valid* del VI FFT. Di conseguenza il valore dell'uscita cambia da alto a basso in base al valore di uscita dalla FFT permettendo una visualizzazione del suo stato su un DSO.

Panello Frontale del VI in FPGA

La Figura 4.6 mostra il panello frontale del VI realizzato in ambiente FPGA. Si possono notare: la selezione di quale delle due operazioni finestra eseguire, il tempo di campionamento

in tick e il led di controllo del limite del tempo di campionamento. Il pulsante STOP fa in modo che il ciclo while temporizzato si fermi facendo terminare l'esecuzione del VI, infatti con riferimento alla Figura 4.5 si può notare il controllo di STOP collegato alla condizione di esecuzione del ciclo while.

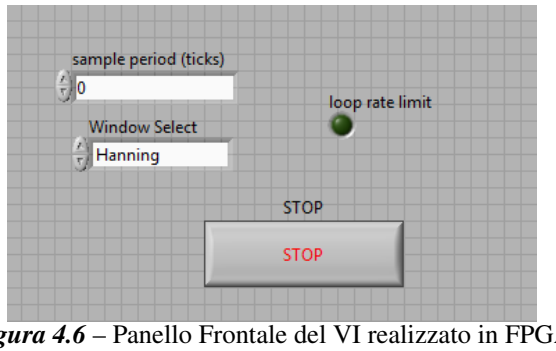


Figura 4.6 – Panello Frontale del VI realizzato in FPGA

4.1.4 VI in ambiente Real-Time

Facendo riferimento alla Figura 4.11, il VI realizzato è stato diviso in due parti per permettere una visualizzazione migliore. Il blocco superiore presenta una sequenza di *FIFO node* i quali richiamano i metodi di inizializzazione delle code FIFO, il blocco inferiore è un ciclo while in qui avviene l'elaborazione e successivamente la visualizzazione del segnale in frequenza. L'elenco numerato che segue fa riferimenti ai numeri presenti in Figura 4.11.

1 Nel punto "1." viene impostato il device sorgente dal quale provengono i dati per l'elaborazione. La *FPGA Target* riprende il VI realizzato in FPGA il quale restituisce in uscita un riferimento al VI contenuto e uno ad eventuali errori. L'errore viene portato su ogni nodo fino alla chiusure del VI stesso in modo tale che se si verifica durante l'esecuzione esso viene gestito con la terminazione del codice. Il FIFO node richiama il metodo *FPGA VI Execution Mode* il quale chiede se il VI è in modalità di esecuzione. Se questo è vero effettua un *reset* del VI altrimenti esegue il caso false nella Struttura case proseguendo senza fare niente.

2 Il punto "2." è costituito da una serie di FIFO node i quali inizializzano le code impostando la lunghezza necessaria. In questo caso la lunghezza viene impostata a 512 elementi visto che i calcoli della FFT vengono effettuati su 256 campioni. Mettere una lunghezza superiore sarebbe uno spreco di memoria del dispositivo.

3 Al punto "3." i FIFO Node richiamano il metodo delle code *start* i quali impostano le FIFO usate in FPGA in modalità di pronto utilizzo. Il nodo successivo richiama

il metodo *run* in modo tale che con l'esecuzione del VI in real-time viene eseguito automaticamente anche il VI FPGA.

4 Considerando il punto "4." il VI (Figura 4.7) effettua una conversione della frequenza di campionamento inserita in base alla velocità del processore e da in uscita la velocità in tick/s la quale va ad inserirsi nel nodo di controllo di scrittura/lettura della FPGA. Il VI restituisce in uscita anche la frequenza di campionamento "attuale", intesa come quella che si utilizza durante l'esecuzione del VI in Real-Time, la quale sarà utile in seguito per rappresentare graficamente lo spettro del segnale acquisito.

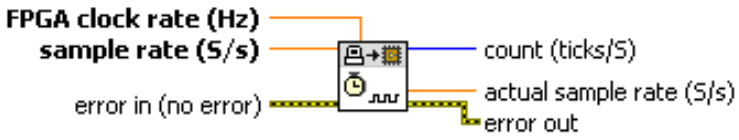


Figura 4.7 – VI di conversione della frequenza di campionamento in basa alla frequenza del processore della scheda

Il *Read/Write Control* ha in ingresso un riferimento al VI realizzato in FPGA e uno alla scelta della finestra da applicare ai segnali acquisiti. In uscita invece presenta un riferimento al led che esprime la condizione di campionamento.

5 In questo punto vengono richiamate le due code FIFO usate in ambiente FPGA in cui viene scritta la parte reale e la parte immaginaria della FFT. I due FIFO Node ricevono in ingresso, oltre all'errore, il riferimento al VI realizzato in ambiente FPGA da cui richiamare le code. Restituiscono in uscita un array della dimensione del numero di elementi ricevuto in ingresso il quale verrà utilizzato per la rappresentazione dello spettro del segnale acquisito dal VI *FFT Spectrum*. Quest'ultimo, come mostra la Figura 4.8 riceve in ingresso parte reale e parte immaginaria sotto forma di array, il tipo di spettro che si vuole visualizzare (l'ampiezza) e la frequenza di campionamento in Sample/s. Restituisce in uscita lo spettro visualizzato su un grafico (Figura 4.9).

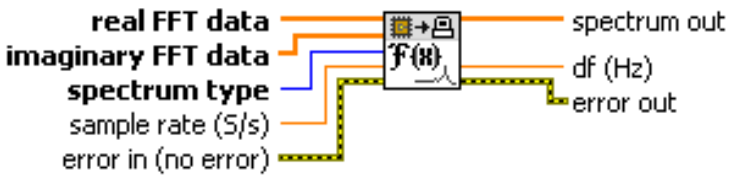


Figura 4.8 – subVI di ricostruzione dello spettro della FFT

6 Come nel punto precedente il FIFO node invoca il metodo di lettura sulla coda Signal usata su FPGA per scrivere i valori del segnale campionato senza alcuna elaborazione.

In questo modo si riesce a visualizzare su un grafico il segnale che stiamo acquisendo con il dispositivo.

Panello Frontale del VI in Real-Time

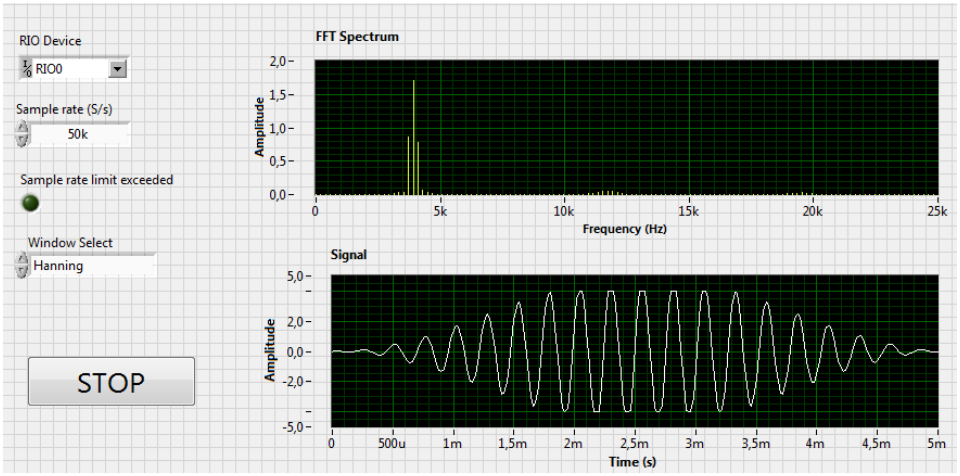


Figura 4.9 – Esempio di acquisizione di un segnale sinusoidale con finestratura di Hanning

Con riferimento alla Figura 4.9 con il controllo *RIO Device* si seleziona il dispositivo che si intende utilizzare come stumento di acquisizione. Sul controllo *Sample rate* viene impostato manualmente dall'utilizzatore la frequenza di campionamento in Samples/s che si vuole utilizzare. Il led *Sample rate limit*, come già detto nei punti precedenti si accende quanto la frequanza di campionamento supera quella inserita. Il selezionatore di finestra consente all'utente di selezionare se applicare una funzione di pesatura di Hanning, di Blackmann-Harris oppure nessuna finestratura. In Figura 4.9 viene mostrato l'esempio di acquisizione di una sinusoide con la rispettiva rappresentazione in frequenza sulla quale viene eseguita una funzione finestra di Hanning. In Figura 4.10 viene mostrato un esempio di acquisizione dello stesso segnale della figura precedente e della corrispettiva rappresentazione in frequenza in cui non viene applicata un operazione di finestratura. Si può notare la differenza nel dominio delle frequenza dell'onda sinusoidale con finestratura e senza. In Figura 4.10 lo spettro del segnale è costituito da un singolo picco principale di valore massimo mentre in Figura 4.9 lo spettro è caratterizzato da un valore principale e da due massimi secondari. Questo è dovuto al fatto che l'operazione di finestratura ha il vantaggio di estendere il segnale per periodicità nel tempo causando però un degrado nel dominio della frequenza come già accennato nel paragrafo 3.2.

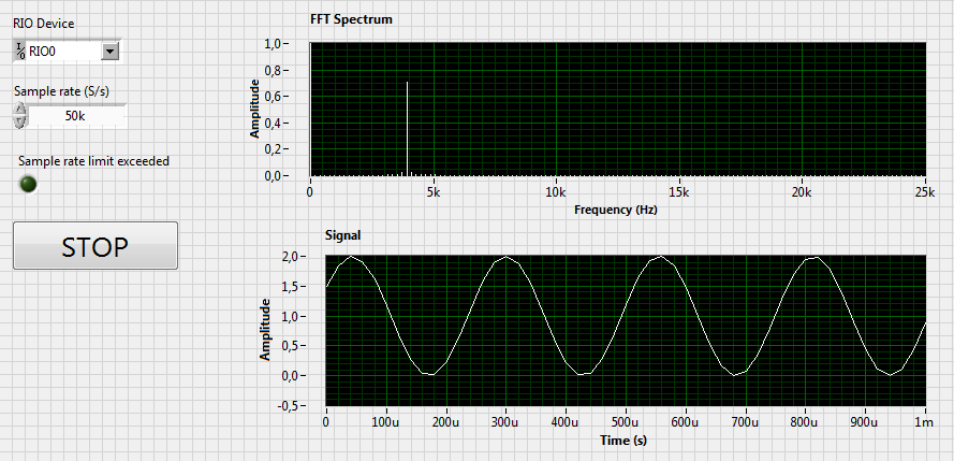


Figura 4.10 – Esempio di acquisizione di un segnale sinusoidale senza finestratura

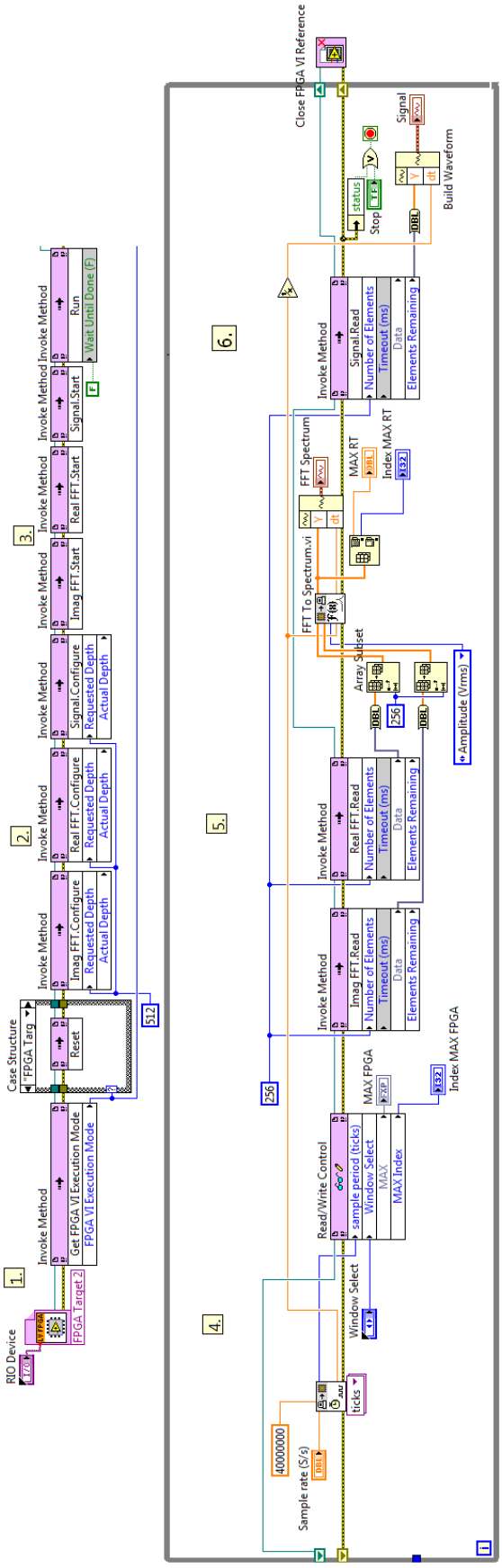


Figura 4.11 – VI creato in ambiente Real-Time

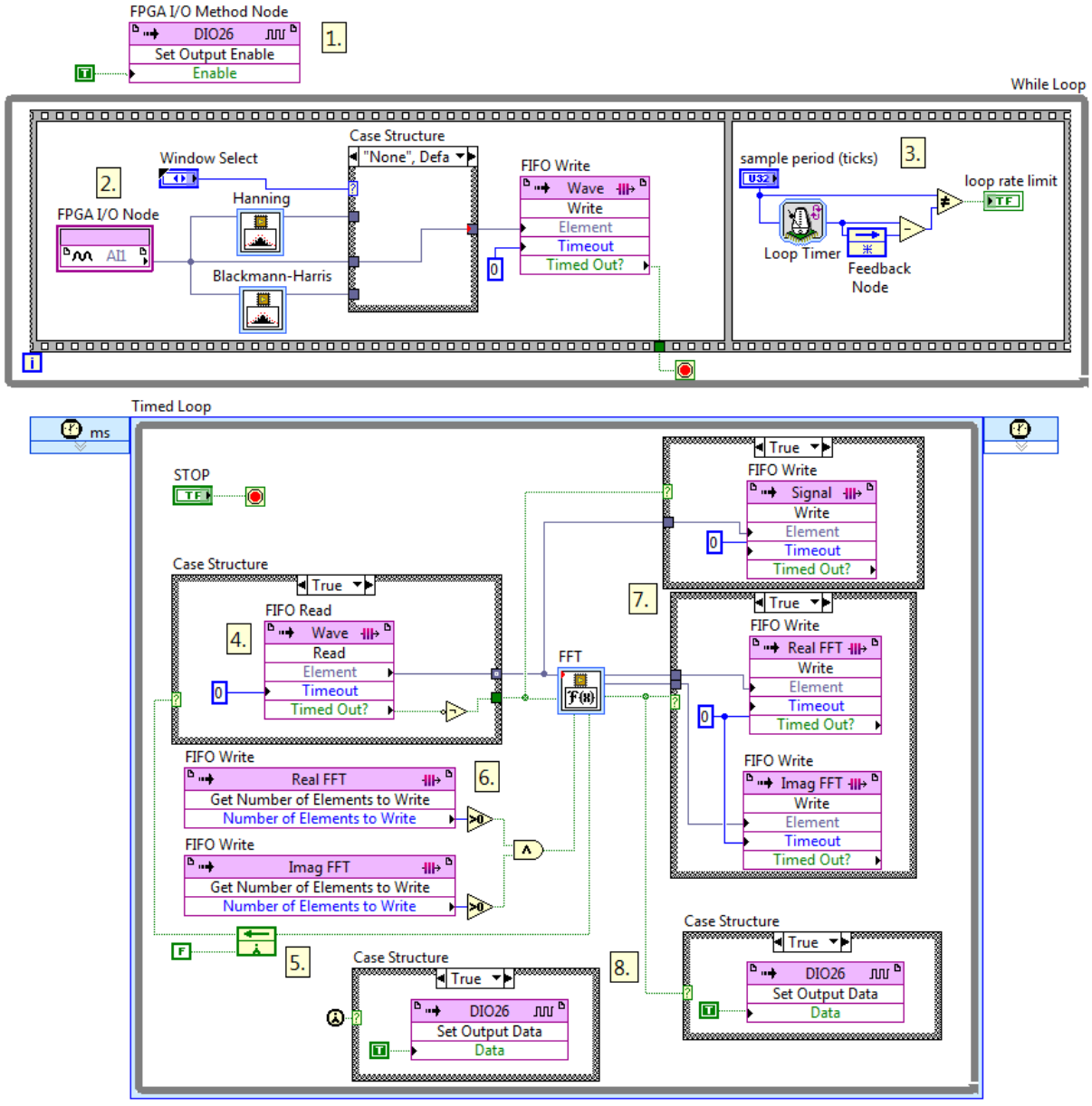


Figura 4.12 – VI completo creato in ambiente FPGA per il calcolo della FFT

4.2 Analisi delle prestazioni

Passando all'analisi delle prestazioni della scheda, la Figura 4.13 mostra il comportamento del dispositivo durante un test di acquisizione ed elaborazione. Si può notare, indicato dalla freccia, un primo gradino il quale corrisponde ad un ciclo di clock del dispositivo.

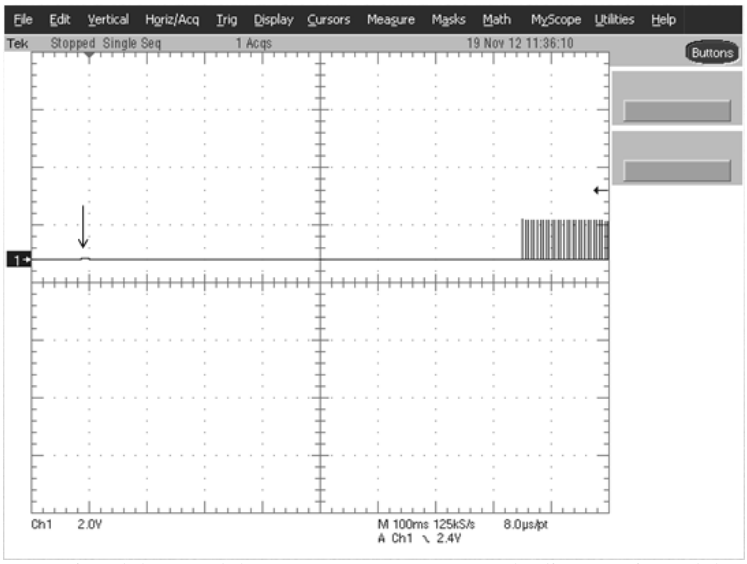


Figura 4.13 – Immagine del DSO del comportamento temporale di esecuzione del VI da parte della sbRIO

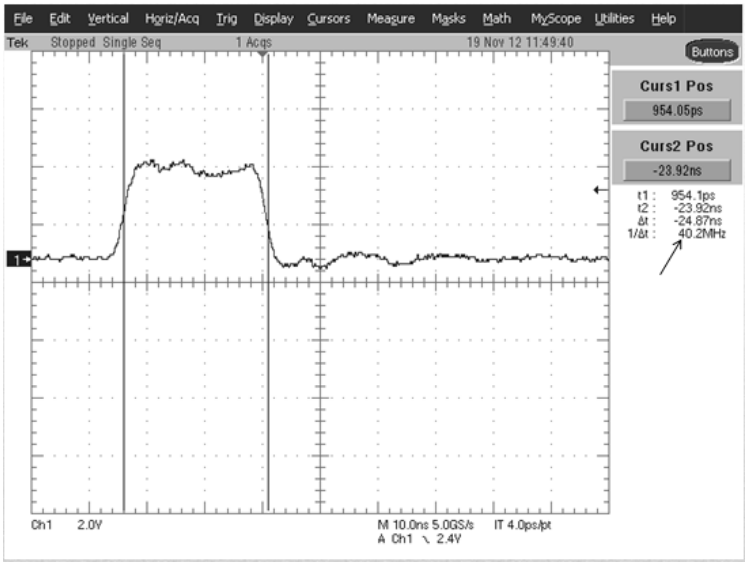


Figura 4.14 – Immagine dal DSO della larghezza del ciclo di inizializzazione del dispositivo

Nella Figura 4.14 il gradino è stato ingrandito ed e stato misurato tramite i cursori del DSO. Si è verificato che corrisponde ad un ciclo di clock pari a 40MHz. Dal momento del gradino passa un periodo in cui il dispositivo procede con l'acquisizione dei 256 campioni, che è il tempo indicato nel paragrafo 4.1.2 come *Latency*. In Figura 4.15 viene misurato il tempo impiegato dalla scheda uguale a 746,7ms il quale non corrisponde a quanto indicato nelle proprietà del calcolo della FFT. Infatti nel paragrafo 4.1.2, dove viene analizzato il VI FFT la velocità di acquisizione dei primi campioni viene associato un valore di 1156 cicli, nel caso di 256 campioni, che corrispondono a 0.0289 ms. Questo probabilmente è dovuto al fatto che la scheda deve prima eseguire delle operazioni di inizializzazione interne prima di cominciare con l'acquisizione dei 256 campioni ma non si può dire con certezza visto che, osservando la Figura 4.13 non vengono dati segni di elaborazione da parte del dispositivo dopo il primo gradino eseguito in un ciclo di clock e prima del primo campione di FFT disponibile.

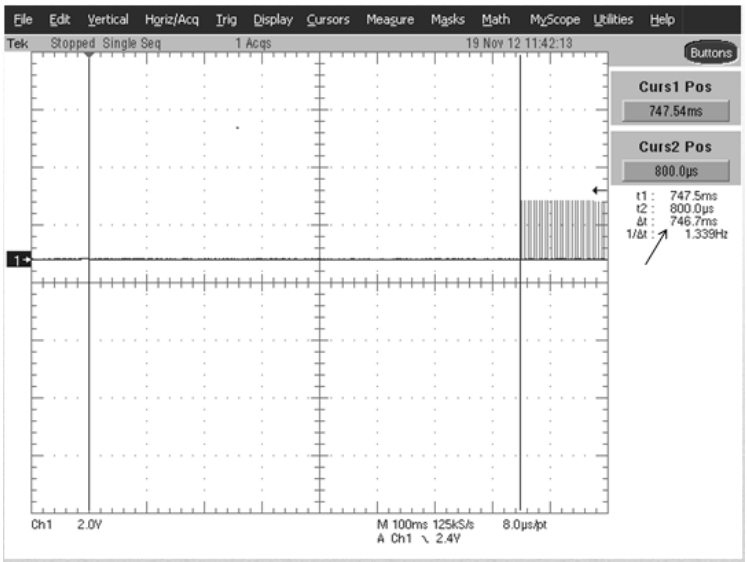


Figura 4.15 – Immagine dal DSO con il tempo trascorso per l'acquisizione dei primi 256 campioni

Considerando invece il calcolo della FFT, cioè l'intervallo temporale che trascorre tra un campione ed il suo successivo, misurato con il DSO (Figura 4.16) corrisponde a circa quello che ci aspettavamo:

$$
\begin{aligned}
&F_s = 50kHz \quad \text{Frequanza di campionamento} \\
&T_s = 1/Fs = 20\mu s \quad \text{Tempo di campionamento} \\
&T_w = T_s * 256samples = 5,12ms \quad \text{Finestra temporale di visualzizzazione}
\end{aligned} \tag{4.1}
$$

Come mostra la Figura 4.16 viene misurato con i cursori del DSO il tempo trascorso tra due campioni consecutivi di FFT in cui l'intervallo temporale misurato è di 5.138ms che è circa quello calcolato considerando gli errori di misura.

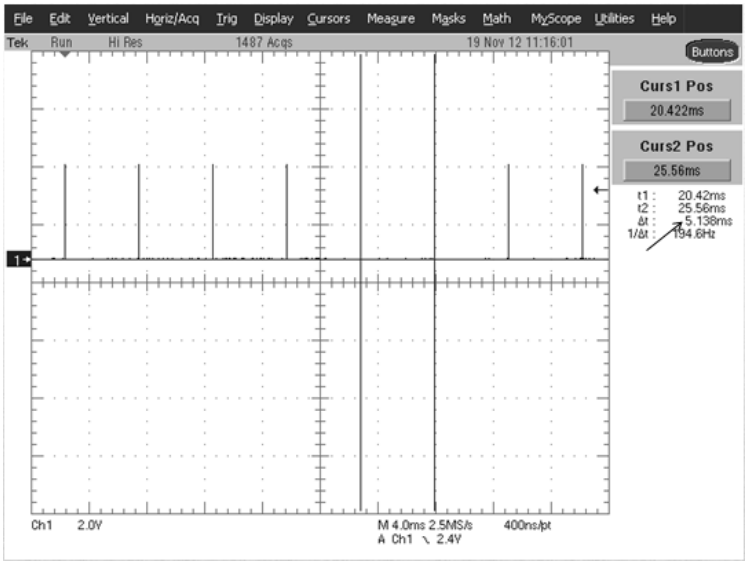


Figura 4.16 – Immagine dal DSO con evidenziato il tempo trascorso tra due campioni di FFT

La Tabella 4.17 mostra i tempi di esecuzione della FFT con 256, 512 e 1024 campioni in cui i tempi di esecuzione sono quelli attesi.

Campioni per FFT	Tempo Esecuzione
256	5,138 ms
512	10,24 ms
1024	20,48 ms

Figura 4.17 – Tabella riassuntiva dei tempi di esecuzione della FFT

Uno strumento utile per la visualizzazione delle risorse del dispositivo sbRIO durante

la compilazione è la finestra di dialogo messa a disposizione dal compilatore (Figura 4.18) sulla quale viene visualizzata un stima di analisi sull'utilizzo delle risorse che la FPGA userà per la gestione del VI creato. Con riferimento alla Figura in oggetto si può notare che vengono utilizzate l'8, 8% dei *Slice Registers*, il 23, 6% delle *Slice LUTs* e il 16, 4% dei blocchi RAM.

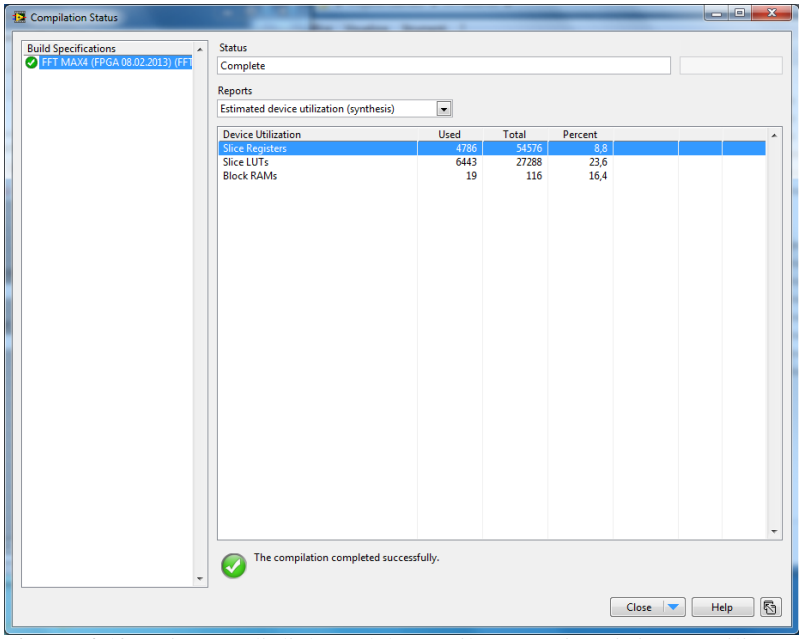


Figura 4.18 – Finestra di dialogo del compilatore - Sintesi risorse utilizzate

Già a questo punto possiamo vedere se la FPGA sta utilizzando una percentuale di risorse superiore alle sue capacità oppure no. In questo modo possiamo fermare la compilazione per evitare un tempo prolungato di inutile compilazione in quanto verrà terminata dal compilatore stesso nei passi successivi. In Figura 4.19 viene mostrata la sintesi del passaggio successivo che è quello della mappatura di tutte le risorse utilizzate dal dispositivo. Le percentuali in questo punto sono quelle effettive, infatti dall'analisi alla mappatura il dispositivo ottimizza l'utilizzo delle sue risorse in base al VI creato.

Infine la Figura 4.20 mostra il riassunto finale dopo aver terminato la compilazione in cui si possono notare le percentuali delle risorse utilizzate dal dispositivo, i tempi di compilazione di ogni processo ed il tempo totale impiegato per la compilazione.

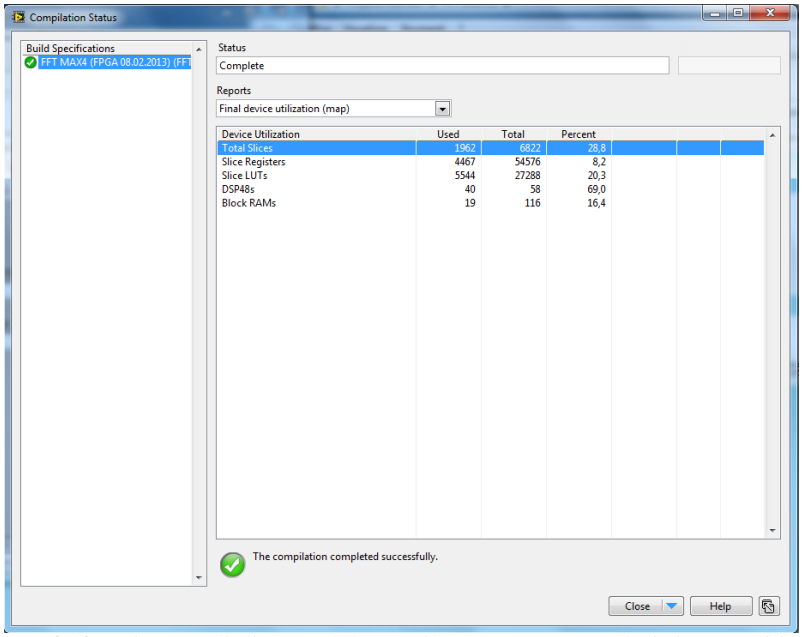


Figura 4.19 – Finestra di dialogo del compilatore - Percentuali di risorse utilizzate

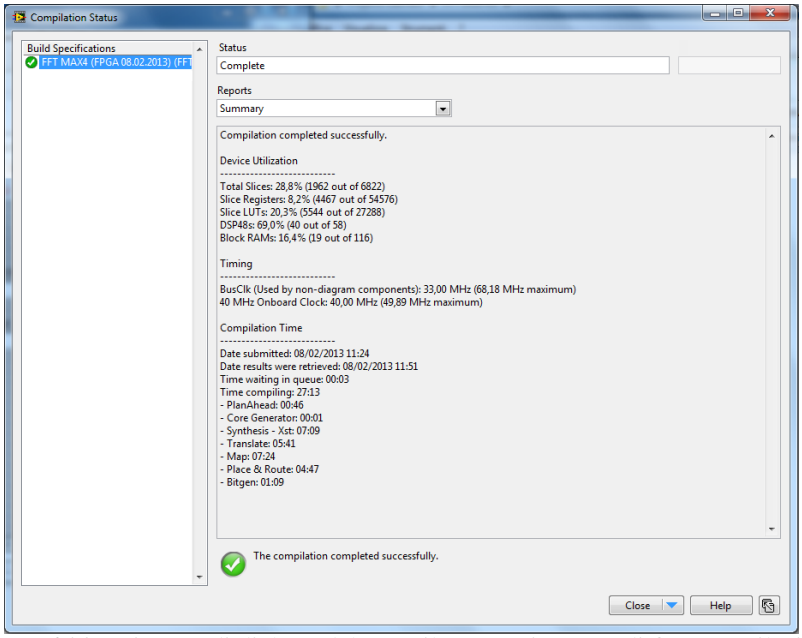


Figura 4.20 – Finestra di dialogo del compilatore - Riassunto di fine compilazione

4.3 Sviluppi futuri

L'implementazione futura all'interno della scheda potrebbe essere quella del metodo di interpolazione spettrale descritto nel paragrafo 3.3. Una possibile strada di implementazione è quella della ricerca del massimo dello spettro del segnale e dell'indice in cui esso è posizionato. Dopo aver trovato il massimo, si considera il valore di ampiezza più grande tra il precedente ed il successivo, in questo modo si può applicare:

$$\frac{A_{i0}}{A_{i1}} = \alpha_0 = \frac{H - 1 + \hat{\delta_0}}{H - \hat{\delta_0}} \tag{4.2}$$

in cui A_{i0} e A_{i1} sono rispettivamente il massimo ed il successivo valore più grande trovato. Scrivendola in funzione di $\hat{\delta_0}$ si ha:

$$\hat{\delta_0} = \frac{H\alpha_0 - H + 1}{\alpha_0 + 1} \tag{4.3}$$

in questo modo si ha una stima di quanto sia lo spostamento della frequenza all'interno della rappresentazione spettrale e dove H è un numero ricavato dal tipo di funzione finestra che si vuole utilizzare. A questo punto si può dare una stima della frequenza:

$$\hat{f_0} = (i_0 \pm \hat{\delta_0})F = \hat{\lambda_0}F \qquad \text{F risoluzione spettrale} \tag{4.4}$$

dell'ampiezza:

$$\hat{A_0} = \frac{2}{|W(-\hat{\delta_0})|}|A_{i0}| \tag{4.5}$$

e della fase:

$$\hat{\varphi_0} = arg[A_{i0}] - arg[W(-\hat{\delta_0})] \tag{4.6}$$

$|W(-\hat{\delta_0})|$ è la funzione della finestra utilizzata calcolata in $(-\hat{\delta_0})$. In pratica si utilizza la (3.27) in modulo. Consideriamo ad esempio una finestra di Hanning in cui H = 2, si ha che:

$$\begin{aligned} |W(-\hat{\delta_0})| &\simeq \left| \frac{Nsin(\pi f)}{2^{2H-2}\pi f} e^{-j\pi f} e^{j\frac{\pi}{N}f} \frac{(2H-2)!}{\prod_{q=1}^{H-1}(q^2-f^2)} \right| \\ &= \frac{Nsin(\pi(-\hat{\delta_0}))}{2\pi(-\hat{\delta_0})} \frac{1}{1-\hat{\delta_0^2}} = \frac{Nsinc(-\hat{\delta_0})}{2(1-\hat{\delta_0^2})} \end{aligned} \tag{4.7}$$

I passi sopra descritti dovrebbero essere gli sviluppi da fare sulla FPGA in modo tale da implementare il metodo di interpolazione. Di seguito si da una possibile configurazione

della ricerca del massimo e dell'indice in cui si trova nella rappresentazione spettrale del segnale.

La Figura 4.21 e la Figura 4.22 mostrano il codice LabVIEW implementato per la ricerca del massimo dello spettro e del suo indice. In Figura 4.21, rispetto al codice in Figura 4.5, sono state aggiunte due ulteriori code FIFO in cui vengono scritti la parte reale e la parte immaginaria provenienti dal calcolo della FFT ed utilizzate per portare i valori al while in Figura 4.22 eseguito in parallelo insieme agli altri due cicli while in Figura 4.12. Per chiarire meglio, i codici implementati in FPGA fanno parte dello stesso VI, sono stati divisi in immagini differenti per evitare un'immagine molto piccola ed incomprensibile.

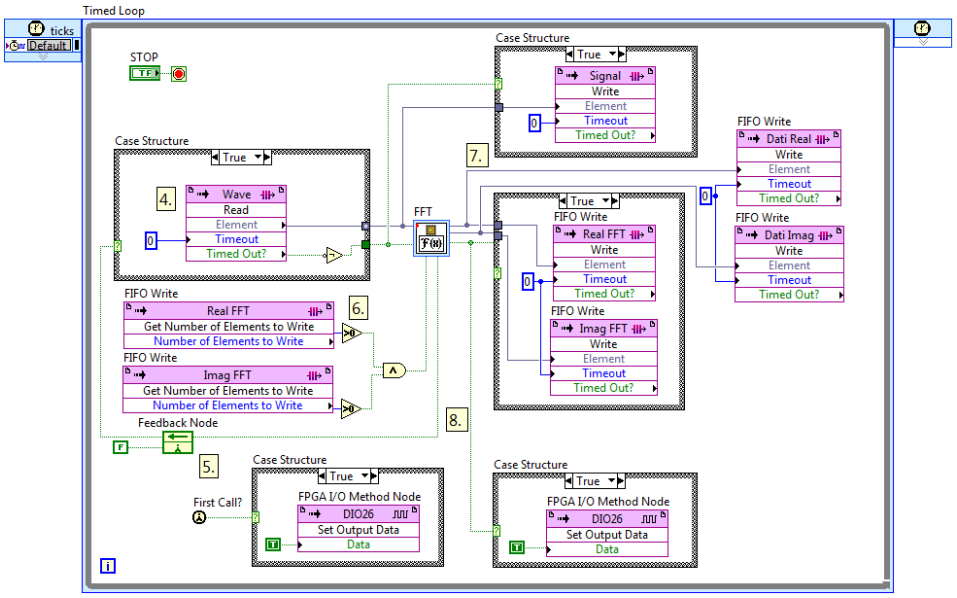


Figura 4.21 – Vi realizzato in FPGA modificato per l'elaborazione della ricerca del valore massimo del modulo

Con riferimento alla Figura 4.22:

- parte reale e parte immaginaria della trasformata di Fourier vengono letti dalle due code FIFO e viene fatto il loro modulo;
- si ha un ciclo for che si ripete per 256 volte, in modo tale tenere traccia degli indici degli elementi letti dalle code;
- viene fatta una normalizzazione degli elementi dividendoli per 256 e poi moltiplicandoli per il valore efficace;

- la condizione in cui si verifica che l'indice sia maggiore di 128 implica la scelta del valore massimo ai secondi 128 elementi dello spettro del segnale;
- se l'indice è minore di 128 la struttura case esegue il caso falso e non facendo niente. Quando l'indice del ciclo for è maggiore di 128 la struttura case esegue il caso vero nel quale comincia la ricerca del massimo, in cui si usano dei *shift register* per tenere in memoria il valore precedente e confrontarlo con il successivo;
- si confronta il valore precedente con il successivo. Se questo è maggiore allora viene eseguito il caso vero della seconda struttura case in cui il valore più grande passa allo *shift register* e di conseguenza anche il suo indice. Altrimenti la seconda struttura case esegue il caso falso in cui passa allo *shift register* il valore che era già presente al suo interno insieme al valore dell'indice presente nello *shift register* usato per l'indice;
- quando il ciclo for è stato eseguito per 256 volte si ha in uscita il valore massimo ed il suo indice.

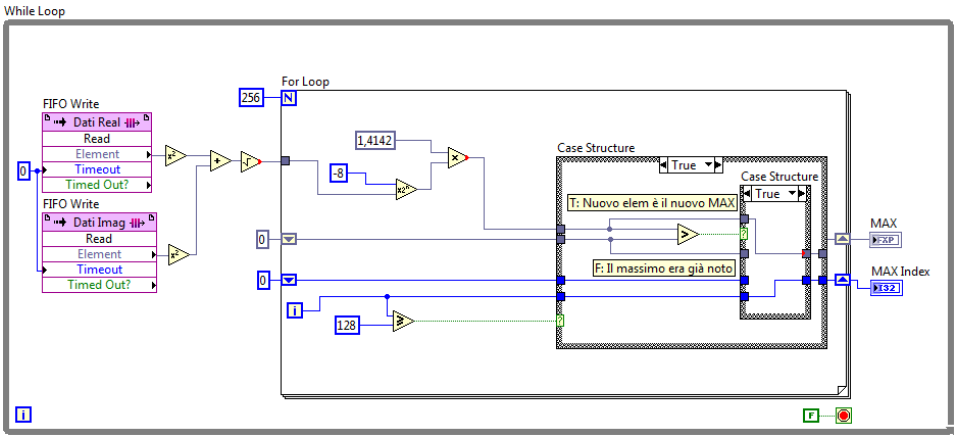


Figura 4.22 – Vi realizzato in FPGA per il calcolo del valore massimo del modulo e del suo indice

Mentre è stato provato che in ambiente Real-Time il codice LabVIEW funziona dando come valori finale il massimo e l'indice corrispondente rimane da verificare in ambiente FPGA. Il tempo a disposizione non ci ha permesso di confermare il funzionamento anche su scheda quindi una delle sfide future sarà quella di controllare l'esecuzione su dispositivo.

Capitolo 5

Conclusioni

La prima parte dell'elaborato descrive il contesto operativo in cui si inserisce il presente lavoro di tesi. In particolare, vengono presentati il dispositivo sbRIO 9636 e il codice LabVIEW utilizzato per programmarlo. Successivamente, si affrontano gli aspetti di natura teorica e implementative, quali la trasformata di Fourier e i metodi di interpolazione spettrale. La sezione conclusiva, infine, descrive l'esperienza svolta in laboratorio, dando ampio spazio ai VI approntati in ambiente FPGA e Real-Time.

In sede di stesura del codice sono emersi numerosi aspetti critici, in special modo in ambiente FPGA. A margine delle simulazioni effettuate si è riscontrato come un utilizzo intensivo di strutture array produce un sensibile incremento dei tempi di compilazione, fino a superare i novanta minuti. Simili prestazioni sono motivate dal fatto che ogni array occupa una notevole porzione di memoria della FPGA. La loro gestione richiede l'impiego di numerosi registri logici, ragion per cui si è scelto di adottare delle code FIFO per ridurre al minimo la permanenza dei dati all'interno dei registri. Operazioni immediate in ambiente Real-Time si rivelano molto complicate in ambiente FPGA. La causa principale va rintracciata nelle specifiche tecniche del dispositivo in uso: il potere di calcolo non è proporzionato alla mole di dati in esame.

Ciò nonostante, i tempi di esecuzione della FFT sono molto contenuti e nel medesimo ciclo di lavoro è possibile inserire ulteriori elaborazioni. In fase di sviluppo, si è rivelata molto utile la finestra del compilatore, che consente un controllo costante delle risorse impiegate.

Il tempo a disposizione non ha consentito di raggiungere tutti gli obbiettivi prefissati. In particolare, il lavoro di tesi si è concluso con il calcolo della FFT di un segnale appositamente fornito da un generatore di forme d'onda. I possibili sviluppi futuri riguardano

l'implementazione di algoritmi di interpolazione spettrale. A tal proposito, il problema della ricerca dei punti di massimo viene introdotto nel paragrafo 4.3. D'altro canto, particolare rilievo va destinato alla sincronizzazione delle diverse operazioni all'interno del dispositivo, ciascuna caratterizzata dal proprio tempo di esecuzione e carico computazionale.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] D. Belega and D. Dallet, "Estimation of the multifrequency signal parameters by interpolated dft method with maximum sidelobe decay," *Intelligent Data Acquisition and Advanced Computing Systems, Technology and Applications, pp. 294-299*, 2007.

[2] ——, "Frequency estimation via weighted multipoint interpolated dft," *Science, Measurement and Technology, IET pp. 1-8*, 2008.

[3] M. Bertocco, G. Giorgi, C. Narduzzi, and F. Tramarin, "A case for ieee std. 1451 in smart microgrid environments," *Conference Publications, pp. 124-129*, 2011.

[4] M. Bertocco and A. Sona, *Introduzione alle Misure Elettroniche*, 2010.

[5] G. Cariolaro, G. Pierobon, and G. Calvagno, *Segnali e Sistemi*. McGraw Hill, 2005.

[6] N. Instruments, "http://en.wikipedia.org/wiki/labview," 2012.

[7] ——, "http://www.ni.com/singleboard/i/," 2012.

[8] ——, "Operating instruction and specifications ni sbrio 9636," 2012.

[9] H. S. Stone, "An algorithm for the machine calculation of complex fourier series," *Electronic Computers,IEEE Transactions on, pp. 680-681*, 2011.

[10] Xilinx, "Spartan-6 fpga configurable logic block," 2010.

[11] ——, "http://www.xilinx.com/fpga/index.htm," 2012.

Ringraziamenti

Vorrei ringraziare il prof. Matteo Bertocco e l'Ing. Federico Tramarin per la loro professionalità e disponibilità, entrambe sempre presenti in questi mesi di tesi.

Un particolare ringraziamento va alla National Instruments in maniera speciale all'Ing. Simone Suaria, sempre disponibili per assistenza tecnica.

Desidero ringraziare la mia famiglia, la mia ragazza, i miei amici e tutti coloro che in questi anni mi hanno sostenuto a dare sempre il massimo di me stesso. Infine un ringraziamento va anche al personale del laboratorio di Compatibilità Elettromagnetica e Misure Elettroniche.