ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Scuola di Scienze
Corso di Laurea in Fisica

Scheda di controllo JTAG per l'elettronica di readout del Pixel Detector dell'esperimento ATLAS

Relatore:
Prof. Alessandro Gabrielli

Correlatore:
Dott. Luca Lama

Presentata da:
Stefano Righi

Sessione I
Anno Accademico 2014/2015

“Ma nulla si conosce interamente finché

non vi si è girato tutt’attorno per arrivare

al medesimo punto provenendo dalla parte opposta”

- ***A. Schopenhauer***

Abstract

Il sistema di acquisizione dati del nuovo layer IBL di ATLAS conta attualmente 15 schede ROD attive sull'esperimento. In ognuna di queste schede sono presenti due catene JTAG per la programmazione e il test. La prima catena è facilmente accessibile da remoto tramite uno standard VME o Ethernet, mentre la seconda è accessibile solo tramite un programmatore JTAG. Accedere alla catena secondaria di tutte 15 le ROD è in primo luogo sconveniente poiché sarebbero necessari 15 programmatori diversi; in secondo luogo potrebbe risultare difficoltoso doverli gestire tutti da un unico computer. Nasce così l'esigenza di sviluppare un'elettronica aggiuntiva con funzione di controllo che riesca, tramite un unico programmatore, a distribuire un segnale JTAG in ingresso a 15 uscite selezionabili in maniera esclusiva. In questa tesi vengono illustrati i vari passaggi che hanno portato alla realizzazione del progetto ponendo attenzione alla scelta, al funzionamento e all'eventuale programmazione dei componenti elettronici che lo costituiscono. Per ogni parte è stato realizzato un ambiente hardware di prototipazione che ne ha garantito il test delle funzionalità. La scheda, basata su un microcontrollore ATmega 328-P, è attualmente in fase di completamento nel laboratorio di progettazione elettronica dell'INFN di Bologna.

Il prototipo studiato e realizzato tramite il lavoro di questa tesi verrà anche utilizzato in ambiente CERN una volta che ne sarà convalidata l'affidabilità e potrà anche essere facilmente adattato a tutti gli esperimenti che usano un protocollo JTAG per la programmazione di dispositivi remoti.

Indice

Introduzione

In seguito allo spegnimento dell'acceleratore LHC avvenuto nei primi mesi del 2013, è stato effettuato un importante upgrade sul *pixel detector* dell'esperimento ATLAS. Questo aggiornamento che ha visto l'aggiunta di un rivelatore supplementare nello strato più interno dell'*inner detector* di ATLAS, garantirà in primo luogo di ripristinare l'effettiva efficacia del sistema di rivelazione iniziale che era via via degradata soprattutto nei layer più interni di pixel a causa degli effetti delle radiazioni ionizzanti, e in secondo luogo la raccolta di un numero più elevato di informazioni, permettendo così di effettuare misurazioni più precise dal punto di vista spaziale. Questo apparato di rivelazione ha portato con sé anche un nuovo sistema hardware e software per l'acquisizione dei dati. Le schede hardware, dette *ROD*, che si occupano della formattazione dei dati dispongono di due catene *JTAG* per la loro programmazione e gestione. Una di queste catene è facilmente accessibile da remoto tramite protocollo *VME* o protocollo *Ethernet*. La catena secondaria, invece, è accessibile esclusivamente tramite protocollo *JTAG* e, di conseguenza, tramite un programmatore hardware compatibile. Al momento in cui viene redatta questa tesi sono presenti 15 schede *ROD* all'interno del pozzo sperimentale di ATLAS e non abbastanza programmatori *JTAG* per poter accedere a tutte. Si è deciso così di sviluppare una scheda elettronica che permettesse, con un solo programmatore *JTAG,* di accedere in maniera selettiva a tutte le 15 schede *ROD*.

In questa tesi verrà descritto il lavoro di progettazione e di sviluppo della scheda avvenuto nel laboratorio di progettazione elettronica dell'*INFN* di Bologna. Per ragioni di chiarezza e di completezza verranno anche illustrate le parti fondamentali che caratterizzano l'LHC, l'esperimento ATLAS e l'elettronica di readout di IBL, il nuovo rivelatore del *pixel detector*.

Capitolo 1

Cenni su LHC ed esperimento ATLAS

In questo capitolo, precursore di una trattazione più specifica, ci si occuperà di descrivere, senza alcuna pretesa di completezza, alcuni principi di funzionamento dell' LHC ed in particolare dell'esperimento ATLAS necessari alla comprensione della parte principale di questo trattato.

Tutto il lavoro di sviluppo svolto in laboratorio è circoscritto ad una sezione estremamente ridotta dell'esperimento ATLAS; in particolare, in questa tesi si farà solo riferimento al primo di una serie di rivelatori di ATLAS, detto Pixel Detector, che verrà illustrato in questo stesso capitolo.

1.1 LHC

LHC, acronimo di "Large Hadron Collider", è il più grande ed energetico acceleratore di particelle al mondo. Composto da un anello di magneti superconduttori di $27\ Km$ di lunghezza, situato ad una profondità media di $100\ m$ nel sottosuolo franco-svizzero, è, al giorno d'oggi, l'opera di dimensioni più grande che l'essere umano abbia mai compiuto.

Il nome LHC racchiude le tre più importanti caratteristiche di LHC stesso:

- **Large**

Come appena accennato le grandi dimensioni ne giustificano l'aggettivo

- **Hadron**

Fa riferimento alle particelle che vengono accelerate; per la precisione protoni ottenuti dalla ionizzazione dell'idrogeno, e ioni di piombo.

- **Collider**

Indica che gli urti, sotto forma di collisioni, avvengono tramite due fasci, detti beam, controrotanti di particelle.

1.1.1 Caratteristiche LHC

Le caratteristiche principali utili alle sperimentazioni per cui è nato l'LHC sono l'energia del centro di massa e la luminosità, quest'ultima definita in questo modo:

$$L = n_b \frac{N_1 N_2 f}{A_{eff}} \tag{1.1}$$

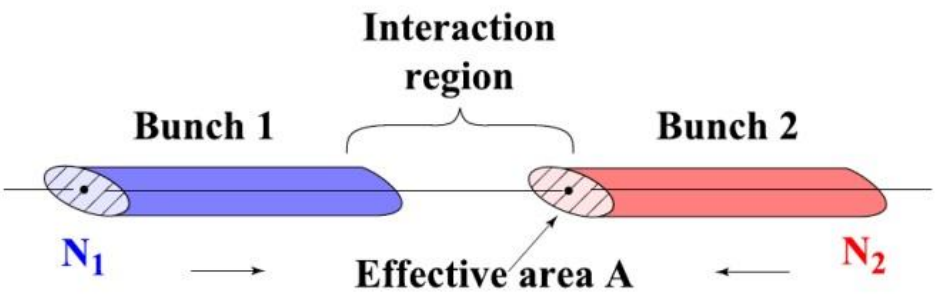

Fig. 1-1 Schematizzazione di una interazione tra pacchetti

Definito come il numero di collisioni che possono essere prodotte nell'unità di tempo e di superficie efficace; dove N_1 ed N_2 sono il numero

di particelle del pacchetto (Bunch) considerato, n_b numero di bunch, A_{eff} la superficie efficace per la collisione che dipende dalla sezione d'urto σ dei fasci di particelle secondo la relazione:

$$A_{eff} = 4\pi\, \sigma_1 \sigma_2 \tag{1.2}$$

Un'elevata energia nel centro di massa consente di produrre particelle di più grande massa. Questa grandezza viene espressa come la somma delle energie dei fasci incidenti $E_1 + E_2$ e, a livello teorico, in linea con l'attuale upgrade di LHC, la massima energia raggiungibile per collisioni è di $14\,TeV$. Questa energia è raggiungibile anche grazie ai 1232 dipoli magnetici, costituiti da superconduttori raffreddati ad elio superfluido alla temperatura di circa $1.9\,K$ che, con un campo magnetico pari ad 8.3 T, garantiscono la forza centripeta necessaria alle particelle per compiere il percorso ellittico dell'LHC. Inoltre, per garantire una giusta collimazione dei fasci sono presenti 392 quadrupoli magnetici.

1.1.2 Sistema di accelerazione del fascio di protoni

L'energia massima, ottenibile in via teorica, del fascio di particelle viene raggiunta a seguito di una catena di acceleratori. Il fascio di protoni viene accelerato, in prima istanza, dall'acceleratore lineare *Linac 2* fino ad un'energia pari a $50\,MeV$. La seconda fase di accelerazione è svolta dal *Proton Synchrotron Booster (PSB)* che accelera il fascio fino ad un'energia di 1.4 GeV. Questo booster garantisce una maggiore iniezione di protoni nel sincrotrone successivo, il *Proton Synchrotron (PS)* che spinge il fascio di protoni fino ad energie di $25\,GeV$. Raggiunta questa energia i fasci vengono deviati nel successivo sincrotrone, il *Super Proton Synchrotron (SPS)*, dove vengono accelerati fino a $450\,GeV$ e iniettati nell'anello principale, l'LHC.

In questo anello i protoni vengono divisi in due fasci (*beam pipes*), un fascio circolante in senso orario e uno in senso antiorario, e portati al valore finale di energia pari a 7 TeV.

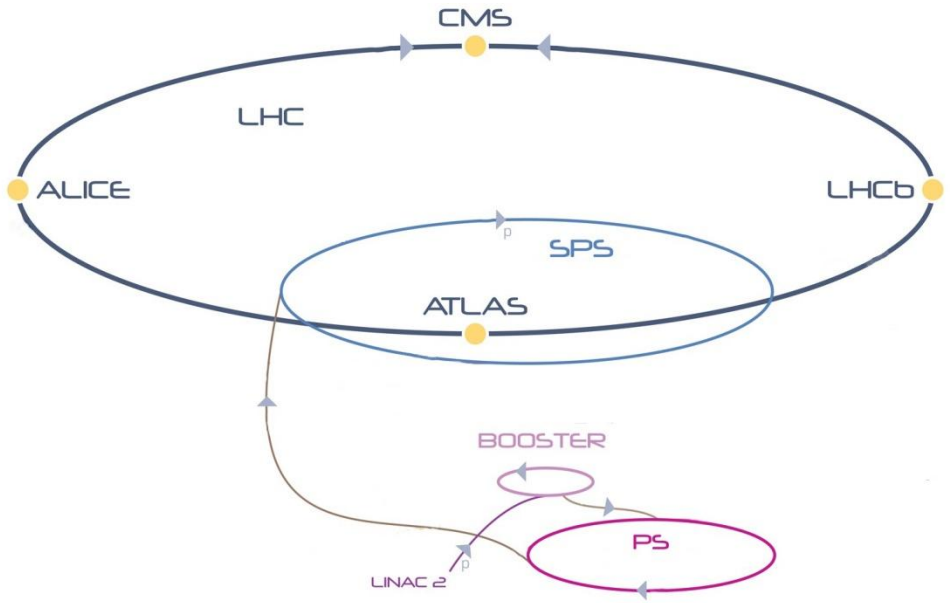

Fig. 1-2 Sistema di accelerazione di protoni

In Fig. 1-2 si possono inoltre vedere i quattro esperimenti che operano sull'acceleratore principale.

1.2 Esperimento ATLAS

ATLAS, uno dei quattro detector lungo l'anello principale dell'LHC (Fig. 1-3), è un dispositivo general purpose lungo 44 m e di diametro 22 m costituito da rivelatori disposti in maniera simmetrica lungo il suo asse longitudinale (Fig. 1-4).

In seguito verrà dato un breve accenno sui principi di funzionamento dei rivelatori che lo costituiscono e poi, al fine di rendere più comprensibile il lavoro svolto in funzione di questa tesi, verrà approfondito qualche

dettaglio interessante sull'*inner detector*, il rivelatore più interno di ATLAS.

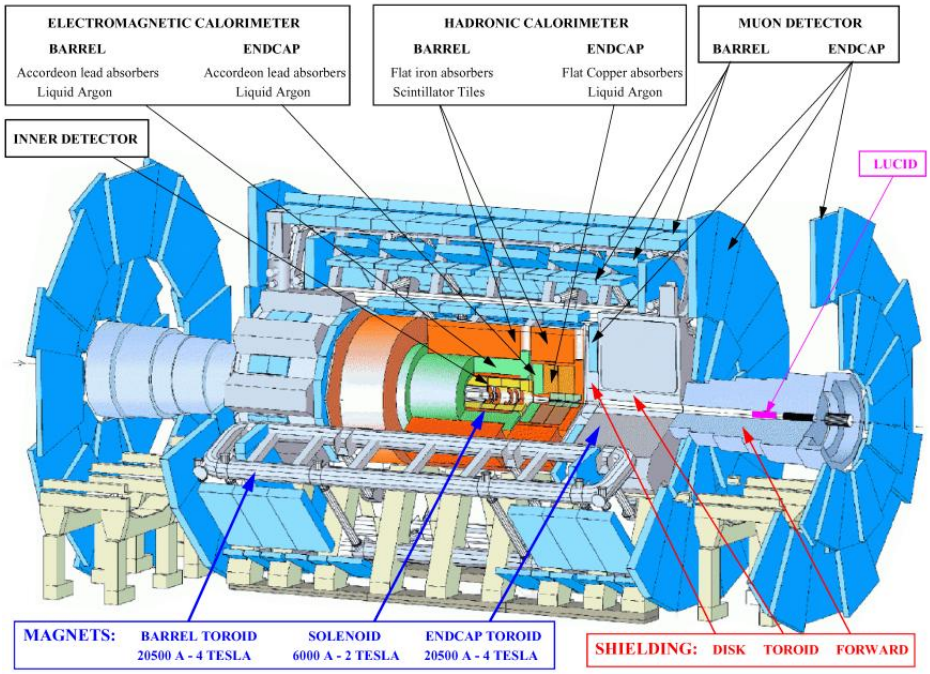

Fig 1-3 Spaccato in prospettiva di ATLAS

1.1 Rivelatori di ATLAS

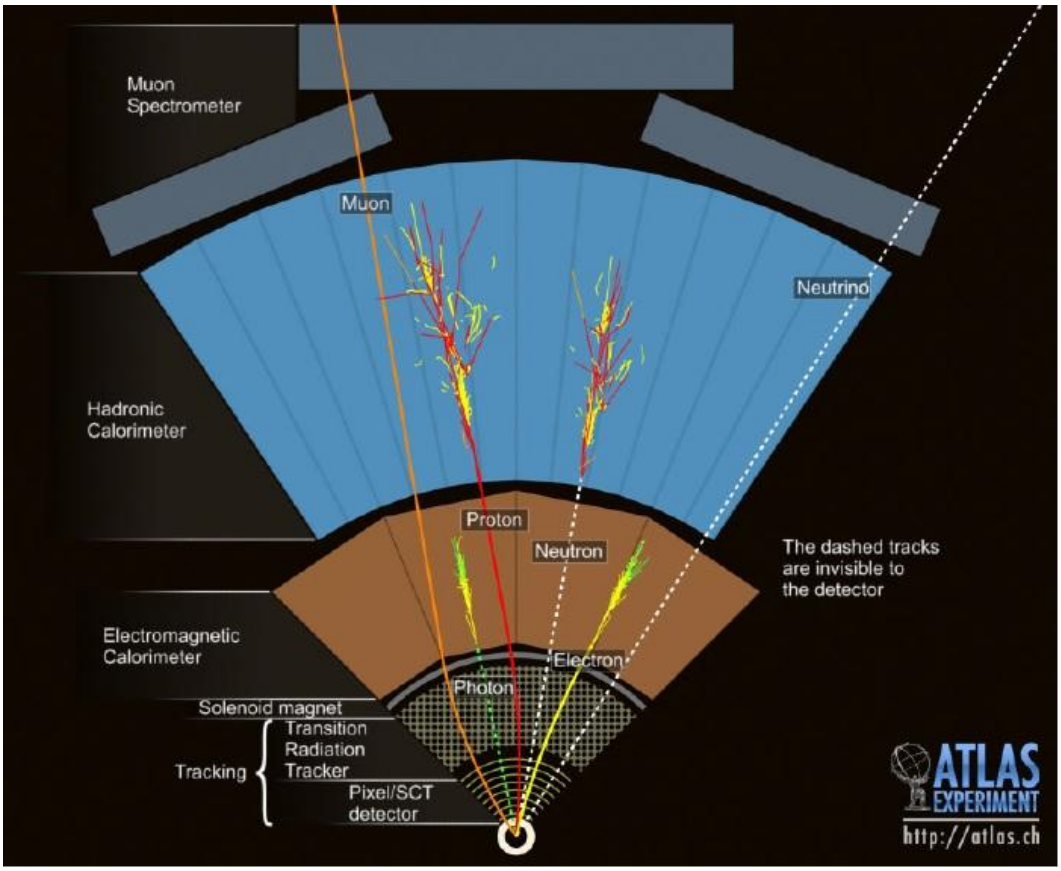

Fig. 1-4 Sezione verticale dei rivelatori

Partendo dai rivelatori più esterni, ovvero quelli più lontani dalla beam pipe (Fig. 1-4), si hanno:

- ***Muons spectrometer***

  I muoni sono le uniche particelle, oltre ai neutrini, a non essere tracciate dai calorimetri a causa della loro elevata massa. Il rivelatore di muoni è formato da due sistemi di rivelazione: le camere di trigger, a loro volta formate da due camere dette *Resistive Plate Chambers* e *Thin Gap Chambers* che si occupano di

una rapida misura dell'impulso dei muoni, e le camere di tracciamento per una lenta ma più accurata misura dell'impulso dei muoni.

- ***Hadronic - Electromagnetic calorimeter***

I calorimetri riescono a rivelare la maggior parte delle particelle note ad esclusione, come già detto, dei muoni che vengono rilevati successivamente, e dei neutrini la cui presenza è dimostrata solo tramite bilancio energetico. Particelle di energia sufficientemente elevata interagiscono con il materiale presente nei due calorimetri generando così uno sciame elettromagnetico di particelle (*particles shower)* che a loro volta interagiscono ionizzando il materiale nei calorimetri. Misurando l'energia di ionizzazione e di eccitazione nel calorimetro si può risalire all'energia della particella iniziale.

Nel calorimetro più vicino alla beam pipe, detto calorimetro elettromagnetico e composto da argon liquido racchiuso tra strati di piombo, si possono misurare le energie degli elettroni e fotoni incidenti

Nel calorimetro adronico, composto invece da masse metalliche, si possono misurare le energie delle particelle sensibili all'interazione forte come ad esempio i pioni.

In questo modo è possibile riconoscere non solo l'energia della particella che attraversa i calorimetri ma anche la natura della particella stessa.

- ***Solenoid magnets***

Il sistema di magneti di ATLAS è composto da quattro magneti superconduttori solenoidali e toroidali permettono ai detector

successivi di misurare la carica e la quantità di moto delle particelle misurandone la loro curvatura.

Il solenoide avvolge completamente l'Inner detector generando un campo magnetico pari a 2 T e mantenuto ad una temperatura costante di $4.5\,K$ grazie ad un sistema di refrigerazione ad elio liquido.

Il toroide, posto nella zona del *Muons Spectrometer*, è composto da 8 bobine rettangolari posizionate a simmetria cilindrica e anch'esse raffreddate con lo stesso sistema di refrigerazione del solenoide.

- ***Inner detector***

È il rilevatore che circonda la beam pipe e di conseguenza più prossimo ad essa. Come si vede dalla Fig. 1-5 l'inner detector è formato da più strati tra cui i *Pixel detectors*, che verranno meglio discussi nel prossimo capitolo.

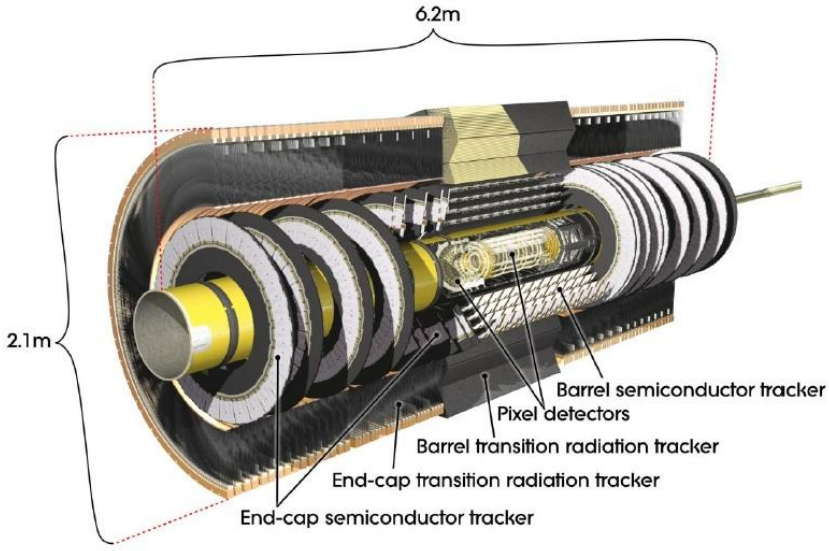

Fig. 1-5 Inner detector

Anch'esso a simmetria cilindrica è immerso in un campo magnetico con linee di campo parallele alla beam pipe e di intensità

pari a 2T. L'inner detector può misurare la quantità di moto e ricostruire la traiettoria delle particelle cariche che lo attraversano. Attraversando i vari strati dell'inner detector (Fig. 1-6) le particelle rilasciano energia lasciando una traccia nei rilevatori. Tale traccia sarà poi fondamentale per ricostruire la traiettoria delle particelle stesse.

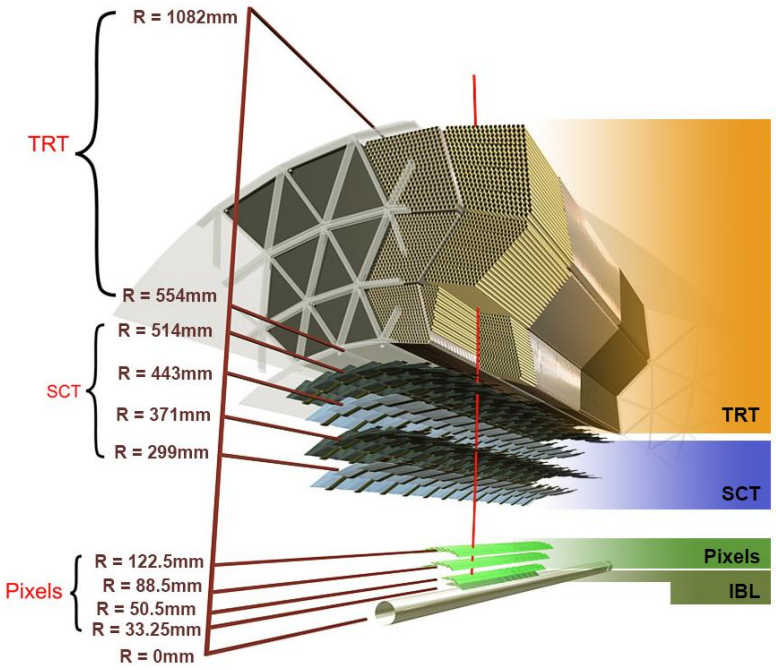

Fig 1-6 Sezione dell'inner detector

La parte più esterna del rilevatore, denominata *Transition Radiation Tracker (TRT)* e composta da strati di tubi di 4 *mm* di diametro, viene utilizzata per misurare la radiazione elettromagnetica che viene emessa quando una particella attraversa un mezzo non omogeneo.

Il secondo rivelatore, più vicino alla beam pipe, è il *Semi Conductor Tracker (SCR)* composto da layers di rivelatori al silicio.

Ultimo, ma di certo non di importanza per questa testi, è il *Pixel detector*. Più vicino alla beam pipe tra tutti i detectors, il pixel detector è stato progettato per sopportare le forti sollecitazioni

elettromagnetiche durante l'acquisizione dei dati. Il detector, prima dell'upgrade datato maggio 2014, era composto da tre layers denominati, partendo dal più esterno, *Layer 2*, *Layer 1*, *Layer B0 (B-Layer)*, composti da strutture metalliche dette *staves*, che contengono i più importanti chip di front-end FEI3 che si occupano della lettura del segnale di carica dei pixel. Anche il *pixel detector*, come si vede in Fig. 1-5, è costruito con una simmetria cilindrica con tre cilindri concentrici di rivelatori e tre endcaps discoidali.

Capitolo 2

IBL ed elettronica di DAQ di IBL

Dopo lo shut down programmato dell'LHC il 13 febbraio 2013 per lavori di upgrade che hanno consentito di raggiungere, ad oggi, un'energia di collisione dei fasci di protoni nel centro di massa pari a $13\,TeV$, è stato installato un nuovo rivelatore dentro all'inner detector di ATLAS.

Tale rivelatore, chiamato *Insertable B-Layer (IBL),* è il quarto layer dei pixel detector inserito tra la beam pipe, opportunamente ridotta di dimensioni, ed il vecchio *Layer B0.*

Le ragioni dell'aggiunta di questo layer sono molteplici ma tra le più importanti si possono citare:

- Limiti di banda.

  Aumentando l'energia di collisione aumenta di conseguenza il parametro *L* di luminosità (1.1)

  I vecchi layers del *pixel detector* non sono stati progettati per supportare l'aumento di del flusso di informazioni dovuto all'incremento di luminosità.

- Danni dovuti alla radiazione elettromagnetica.

  Si è stimato che il *B-Layer* con le attuali energie e quindi con luminosità dell'ordine di $10^{34} cm^{-2} s^{-1}$ degradi fino a rendersi inutilizzabile nell'arco di 3 anni. Inoltre nel tempo si sono verificati danni irreparabili ai sensori a pixel. L'inserimento di *IBL* permette così di recuperare la totale efficienza del *pixel detector*.

Per ottenere quindi una maggiore sensibilità si è deciso di utilizzare nell'IBL il chip FEI4 che garantisce circa il 90% di area sensibile contro il 70% di FEI3.

Per ridurre i danni dovuti alla radiazione, invece, si è provveduto a minimizzare lo spessore dell'elettronica e di tutti i supporti di IBL.

2.1 Elettronica di DAQ di IBL

Il percorso dei dati acquisiti dai chip FEI4 segue pochi e semplici passaggi a favore di una maggior affidabilità di tutto il sistema. Dai 32 chip FEI4 escono 32 connessioni ottiche seriali verso ad una scheda detta *BOC* con un bitrate di $160\ Mbit/s$. Le informazioni vengono poi trasferite alla scheda *ROD* tramite 8 linee parallele di $12\ bit$ ciascuna, formattati e rispediti nuovamente alla *BOC.* A questo punto i dati possono essere trasferiti, nuovamente tramite collegamento ottico, ai computer che si occupano dell'analisi e dello stoccaggio vero e proprio. Un percorso alternativo usato in fase di calibrazione del sistema prevede che i dati vengano trasferiti direttamente ai computer senza nuovamente ritornare alla scheda *BOC.*

I chip di front-end FEI4 possiedono, inoltre, una linea ottica con un bitrate massimo di 40 $Mbit/s$ utile alla configurazione dei chip stessi. I dati per la configurazione vengono inviati tramite un computer alla scheda *ROD.* Dalla scheda *ROD* vengono trasferiti tramite bus parallelo alla scheda *BOC* e, per finire, dalla scheda *BOC* vengono trasferiti tramite fibra ottica al chip di front-end da controllare.

2.1.1 IBL BOC

La *BOC* (Fig. 2-1) è costituita da tre Xilinx Spartan-6 FPGA, due dei quali, chiamati *BMF*, si occupano di prelevare i segnali dai chip di front-end, processarli ed inviarli alla *ROD;* l'altro, chiamato *BCF*, si occupa del protocollo ethernet e della comunicazione con gli altri dispositivi.

La *BOC* inoltre si occupa di distribuire un clock a 40 MHz a tutto il sistema, dai chip FEI4 alla *ROD*.

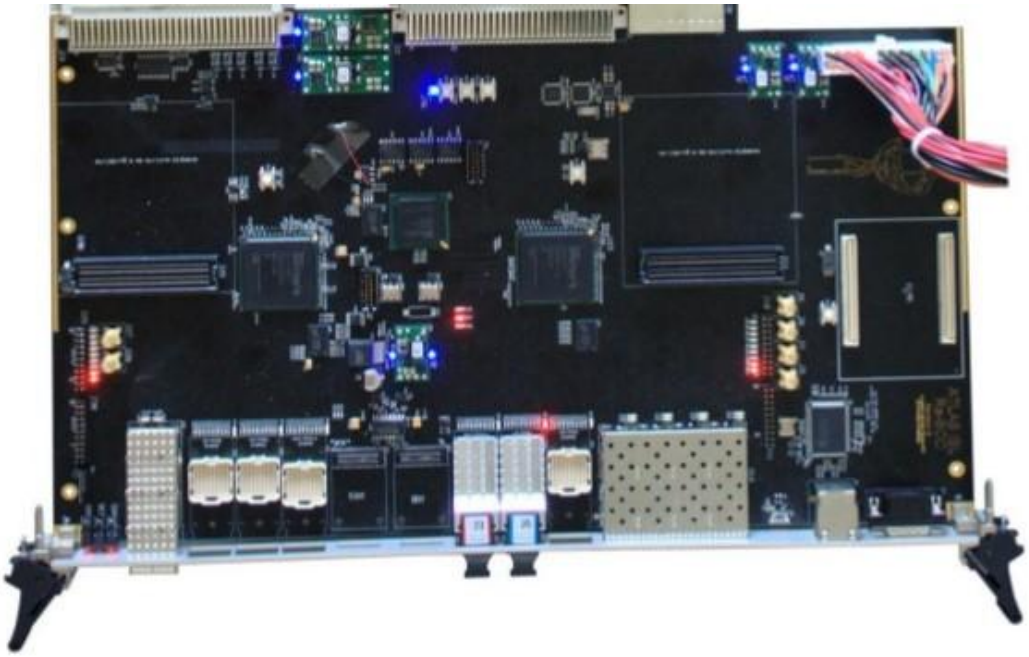

Fig. 2-1 Scheda BOC

2.1.2 IBL ROD

Come già accennato, le mansioni principali della scheda ROD (Fig. 2-2) sono:

- Inviare i segnali di trigger ai chip di front end;
- Inviare le configurazioni ai chip di front end;
- Ricevere le informazioni di un evento dai chip FEI4 ed incapsularli in un singolo data frame.

Sulla scheda, composta da 14 layer, sono presenti i seguenti componenti;

- SDRAM DDR 32 MB
- Memoria FLASH da 64 $Mbit$ Atmel AT45DB642D
- 2 GB DDR2 SODIMM
- Memoria FLASH da 4 $Mbit$ SST39VF040-70-4C-NH
- Interfaccia Ethernet 3 $Gbit$ DP83865
- Processore digitale TMS320C6201-GJC200
- FPGA Xilinx Spartan-6 XC6SLX45-FGG484. Si tratta di un dispositivo *PRM (Program Reset Manager)* che permette alla *ROD* di venire pilotata tramite interfaccia *VME*. Inoltre questa FPGA si occupa delle operazioni di reset e della programmazione della scheda stessa.
- FPGA Xilinx Virtex-5 XC5VFX70T-FF1136. È il controller della *ROD* e contiene un core PPC all'interno.
- 2 FPGA Xilinx Spartan-6 XC6SLX150-FGG900. Si occupano dell'elaborazione dei dati all'interno della *ROD*. Ad ognuna di queste FPGA è collegata una memoria statica più veloce utilizzata

come cache e una memoria dinamica DDR2 più lenta per l'elaborazione vera e propria dei dati.

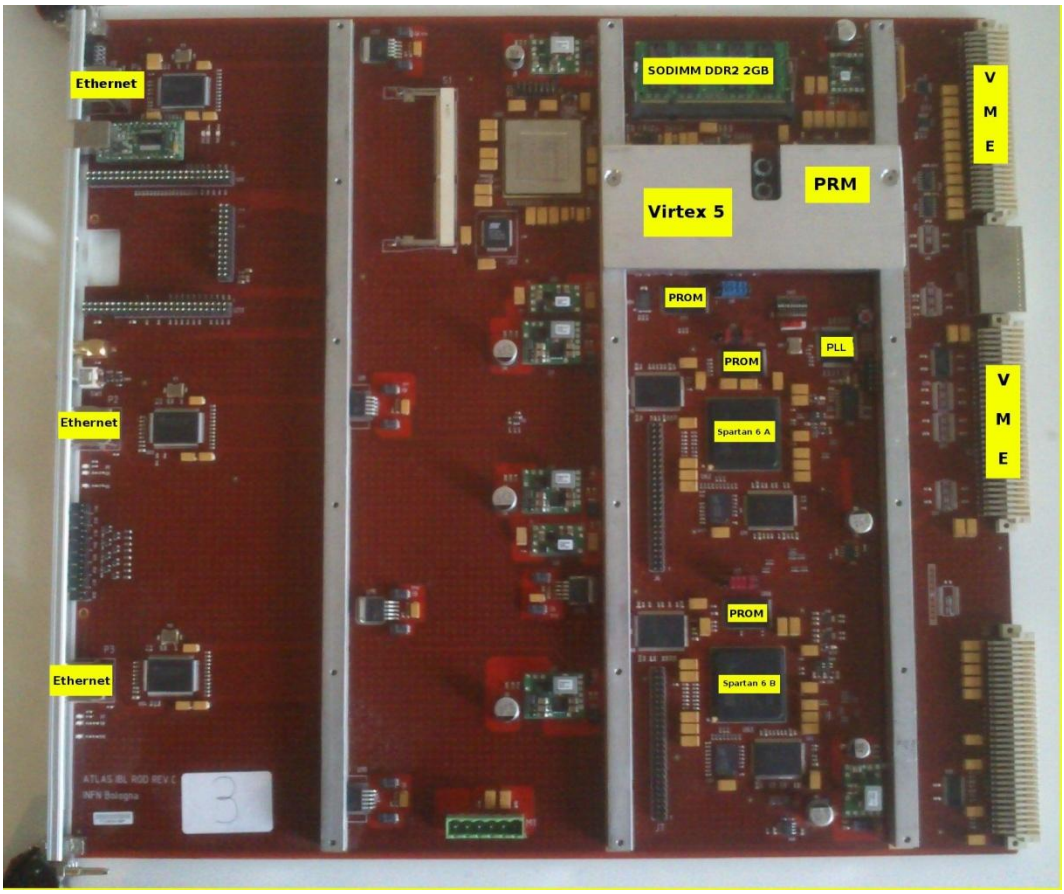

Fig. 2-2 Scheda ROD

Per finire, sul pannello frontale della *ROD* sono presenti le seguenti interfacce per il collegamento esterno:

- Porta USB
- Connettore JTAG per la programmazione delle FPGA
- Pulsante di reset
- 3 Porte Gbit Ethernet
- 8 status led

Sulla scheda sono presenti due clock: uno interno, utilizzato esclusivamente in fase di programmazione, ed uno secondario ricevuto

direttamente dalla scheda *BOC* pari a 40 MHz utilizzato in fase di acquisizione dati dai chip di front-end.

2.2 Protocollo JTAG

JTAG (*Join test action group*) nasce nella seconda metà degli anni ’80 come standard per il debug di schede elettroniche che, con l’avanzare delle tecnologie, diventavano sempre più complesse e difficili da controllare. Più avanti lo standard divenne utile anche alla programmazione vera e propria di più dispositivi elettronici, dal caricamento di firmware in microcontrollori, alla scrittura di dati in memorie flash. Il protocollo prevede un collegamento seriale, detto catena (*Daisy chain*), tra i dispositivi da controllare (Fig. 2-3).

I segnali previsti dal protocollo standard JTAG sono:

- **TDI** (*Test Data in*) Ingresso dei dati nella scheda
- **TDO** (*Test Data out*) Uscita dei dati dalla scheda
- **TMS** (*Test Mode Select*) Selettore della modalità di test
- **TCK** (*Test Clock*) Selettore del clock

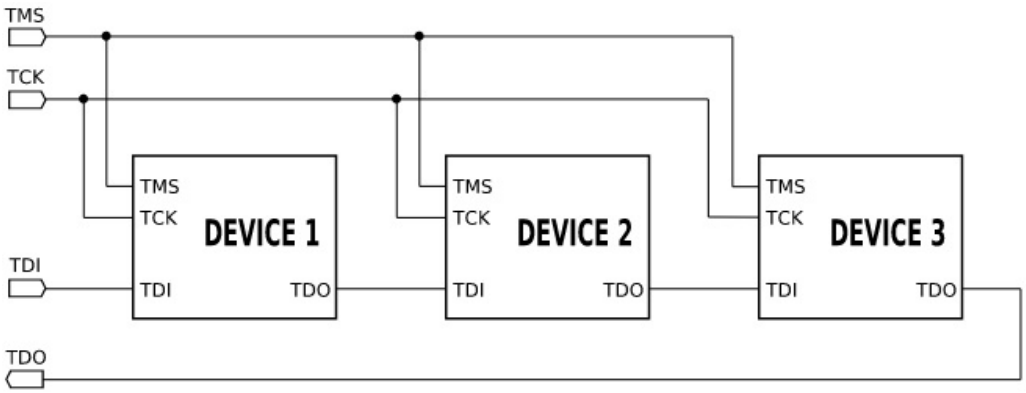

Fig. 2-3 Standard JTAG

I segnali TDI e TDO trasferiscono i dati dall'ingresso della scheda ai vari device che costituiscono la catena. Inoltre la trasmissione seriale è sincrona e pilotata da segnale TCK applicato contemporaneamente a tutti i dispositivi.

2.1 Catena JTAG nelle schede ROD

Nelle schede *ROD* sono presenti due catene *ROD* di cui una primaria, accessibile da remoto tramite protocollo VME o Ethernet IEEE 802.3, che si occupa dell'elaborazione dei dati all'interno della *ROD*, e una catena secondaria, accessibile solo ed esclusivamente da programmatore JTAG, che si occupa del *PRM*.

La catena principale (Fig. 2-4) è composta da:

- XCF32P (Spartan-6A PROM)
- XCF08P (Spartan-6A PROM)
- XC6SLX150 (Spartan-6A FPGA)
- XCF32P (Spartan-6B PROM)
- XCF08P (Spartan-6B PROM)
- XC6SLX150 (Spartan-6B FPGA)
- XCF32P (Virtex 5 PROM)
- XC5VFX70T (Virtex 5 FPGA)

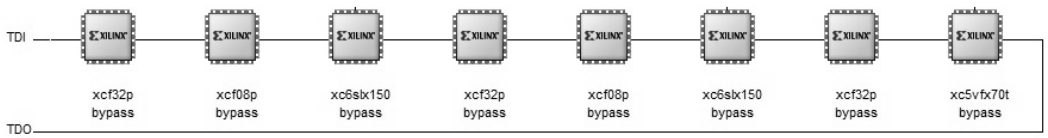

Fig. 2-4 Catena JTAG Principale

La catena secondaria è composta da:

- XCF16P (Spartan-6 PROM)
- XC6SLX75 (Spartan-6 FPGA)

Capitolo 3

Sviluppo scheda di controllo per la catena JTAG secondaria

Il principale svantaggio della catena *JTAG* secondaria della *ROD*, composta dal XCF16P e dal XC6SLX75, risiede nell'impossibilità di venire controllata o programmata a distanza dalle interfacce *VME* o dalle interfacce *Ethernet*. L'unico metodo utile per il controllo della catena secondaria è attraverso un programmatore *JTAG*. Il lotto, ad oggi attivo nell'esperimento *ATLAS*, conta 15 schede *ROD* e si renderebbe necessario un acquisto di ben 15 diversi programmatori. Si è pensato dunque di fare fronte a questo problema sviluppando una scheda elettronica che permetta, con un solo programmatore, di accedere in maniera selettiva ad ogni *ROD* desiderata. Così facendo non solo si andrebbe incontro ad un forte risparmio in termini economici, ma anche ad una forte ottimizzazione di spazio all'interno del crate dove alloggiano le schede e non si avrebbero problemi di risorse di indirizzi nei computer utilizzati per il controllo e per la programmazione delle schede stesse.

Nel seguito verranno illustrati tutti i procedimenti che hanno portato allo sviluppo e alla realizzazione di un prototipo funzionante di questa scheda.

3.1 Programmatore JTAG

Per l'accesso alla catena *JTAG* si è deciso di fare affidamento al dispositivo *Platform Cable USB DLC9G* (Fig. 3-1) prodotto da Xilinx. Questo dispositivo, compatibile con la maggior parte delle *FPGA* in commercio, garantisce un collegamento adattato da standard *JTAG* (Fig. 3-2) a standard USB (Fig. 3-3).

Le principali caratteristiche del programmatore sono:

- Driver con supporto di sistemi operativi windows e Linux Red Hat
- Autosense dei valori di tensione in entrata e in uscita
- Indicatore di stato

Fig. 3-1 Xilinx Platform Cable USB DLC9C

- Compatibilità con FPGA Spartan
- Compatibilità con FPGA Virtex
- Compatibilità con ambiente di sviluppo iMPACT

Come per l'alimentazione, anche la frequenza di clock fornita dal programmatore può variare in funzione della FPGA da controllare. Nel caso della Spartan-6 XC6SLX45-FGG484 la frequenza di clock massima raggiungibile dal programmatore è di 33 MHz.

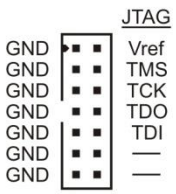

Fig. 3-2 Connettore Standard JTAG

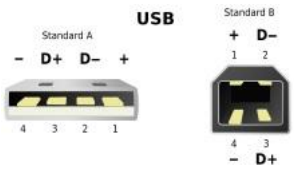

Fig. 3-3 Connettore Standard USB

3.2 Cenni sui Microcontrollori

Il microcontrollore è un dispositivo elettronico integrato su un unico chip utilizzato prevalentemente in sistemi embedded. È un'alternativa più economica e completa rispetto ad un singolo microprocessore poiché, a differenza di quest'ultimo, il microcontrollore è dotato sia di memoria ad accesso causale (*RAM* o *EEPROM*) che di memoria permanente (*ROM*, *EPROM*, o *FLASH*).

Il microcontrollore è inoltre composto da un microprocessore interno con set di istruzioni *CISC* (*Complex Instruction Set Computer*) o *Risc* (*Reducted Instruction Set Computer*) progettato per interagire con mondo esterno tramite linguaggio *assembly* dedicato. Analogamente ad un normale microprocessore, il processore del microcontrollore è dotato di un'unità *ALU* (*Arithmetic & Logical Unit*) che si occupa delle operazioni logiche ed aritmetiche, e di un'unità *CPU* (*Central Processing Unit*) che gestisce il flusso di dati e di indirizzi dall'unità *ALU* ai dispositivi esterni.

3.2.1 Microcontrollore ATmega 328-P

È il microcontrollore (Fig. 3-4) prodotto da *Atmel*, utilizzato per sviluppare la scheda descritta in questa tesi.

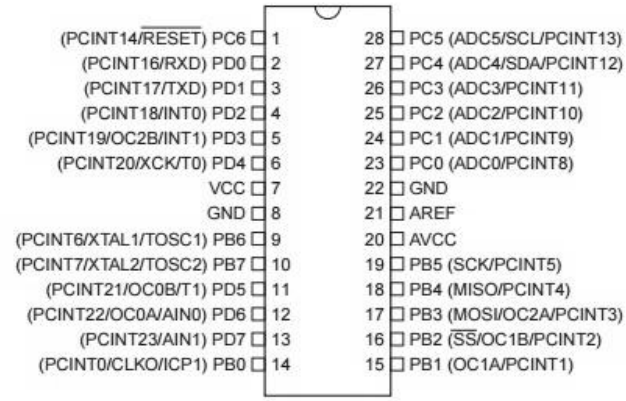

Fig. 3-4 Schema della piedinatura dell'ATmega 328-P

Basato su un processore *AVR* da 8-bit *RISC*, questo microcontrollore è composto da una memoria *FLASH* da $32\ KB$ con possibilità di lettura anche durante la scrittura, una *EEPROM* da $1024\ B$, una *SDRAM* da $2\ KB$, 23 linee di *I/O* general purpose.

Le principali caratteristiche, riassunte in tabella, sono:

Parametri	**Valori**
Flash	$32KB$
Pin	32
Frequenza massima di clock	$20MHz$
Clock interno	$8MHz$
CPU	8-bit AVR
Numero massimo di pin di I/O	23

Parametri	**Valori**
Canali ADC	8
Risoluzione ADC	10 bit

Durante la fase di programmazione del microcontrollore, che verrà meglio specificata nei prossimi paragrafi, è stato deciso di non utilizzare il clock interno del ATmega 328-P ma di utilizzare un quarzo con frequenza di oscillazione pari a $16MHz$ collegato ai pin 9 e 10 del microcontrollore (Fig. 3-5)

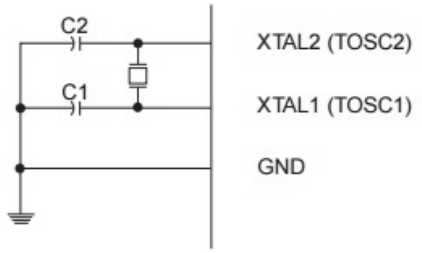

Fig. 3-5 Circuito di oscillazione

Come si vede dall'immagine sono stati utilizzati due condensatori ceramici $C1$ e $C2$ rispettivamente da 12 e 22 pF. Questi condensatori, consigliati nel datasheet del microcontrollore hanno il compito di stabilizzare il segnale di clock fornito dall'oscillatore riducendo al minimo gli *spike* nei fronti d'onda del clock.

Per concludere questo paragrafo verranno illustrate (Fig. 3-6) i parametri caratteristici e i range massimi sopportabili (Fig. 3-7) dal ATmega 328-P[1].

[1] Informazioni più dettagliate e datasheet sono reperibili nella web page http://www.atmel.com

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$ $0.3V_{CC}$	V
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$ $0.6V_{CC}$		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.8V_{CC}$ $0.7V_{CC}$		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$	V
V_{IH2}	Input High Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V		$0.9V_{CC}$		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$ $0.3V_{CC}$	V
V_{IH3}	Input High Voltage, $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$ $0.6V_{CC}$		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{OL}	Output Low Voltage except RESET pin	I_{OL} = 20mA, V_{CC} = 5V	T_A=85°C			0.9	
			T_A=105°C			1.0	
		I_{OL} = 10mA, V_{CC} = 3V	T_A=85°C			0.6	
			T_A=105°C			0.7	V
V_{OH}	Output High Voltage except Reset pin	I_{OH} = -20mA, V_{CC} = 5V	T_A=85°C	4.2			
			T_A=105°C	4.1			
		I_{OH} = -10mA, V_{CC} = 3V	T_A=85°C	2.3			
			T_A=105°C	2.1			V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)				1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)				1	µA
R_{RST}	Reset Pull-up Resistor			30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor			20		50	kΩ
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2			<10	40	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2		-50		50	nA
t_{ACID}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V			750 500		ns

Fig. 3-6 Parametri caratteristici ATmega 328-PU

Operating Temperature. . . . . -55°C to +125°C

Storage Temperature . . . . . -65°C to +150°C

Voltage on any Pin except $\overline{\text{RESET}}$
with respect to Ground . . . . . -0.5V to V_{CC}+0.5V

Voltage on $\overline{\text{RESET}}$ with respect to Ground-0.5V to +13.0V

Maximum Operating Voltage 6.0V

DC Current per I/O Pin 40.0mA

DC Current V_{CC} and GND Pins 200.0mA

Fig. 3-7 Valori massimi ATmega 328-PU

3.3 Arduino UNO *Rev.3*

Arduino UNO *Rev.* 3 (Fig. 3-8) è una scheda di prototipazione elettronica open source, general purpose che consente, tra i tanti impieghi, la programmazione ad alto livello di decine di microcontrollori diversi[2].

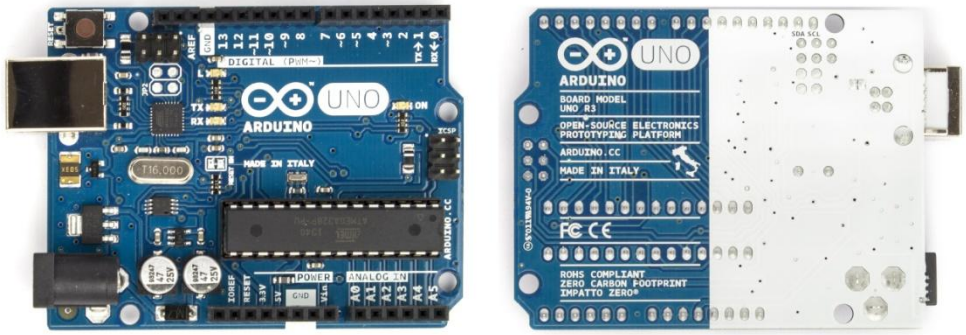

Fig. 3-8 Arduino UNO Rev. 3 Front-Back

In passato la programmazione dei microcontrollori passava inevitabilmente attraverso il linguaggio assembly. Anche se tale linguaggio garantiva grande libertà dal lato della programmazione in quanto estremamente vicino al linguaggio macchina, dall'altro i codici scritti dal programmatore risultavano lunghi, dispendiosi di risorse e di difficile sviluppo e debug. Essendo poi l'assembly un linguaggio di basso livello, non era in alcun modo possibile utilizzare le caratteristiche che rendono forti e indispensabili i linguaggi di programmazione ad alto livello, come le funzioni o gli oggetti. A tutto questo si aggiungeva poi l'impossibilità di avere uno standard universale per la compilazione del codice per diverse famiglie di microcontrollori che disponevano ognuna di un proprio compilatore (*assembler*).

[2] Impieghi principali, caratteristiche tecniche e vari esempi si possono reperire nella web page ufficiale https://www.arduino.cc

Con la nascita delle schede Arduino questi problemi vengono risolti attraverso un'interfaccia hardware già predisposta per la programmazione dei microcontrollori più diffusi. Per semplificare maggiormente ogni aspetto di natura tecnica le schede Arduino sono dotate di un convertitore USB/Seriale in maniera tale da poter essere collegate a qualsiasi moderno PC. Dal lato software, Arduino fornisce un ambiente di programmazione ad alto livello, anch'esso open source, estremamente maturo grazie alla forte community che prende parte allo sviluppo offrendo librerie ricche di oggetti, funzioni e strumenti utili alla programmazione.

Arduino UNO *Rev.* 3 integra nativamente l'ATmega 328-P ragion per cui è stato preso in considerazione durante la stesura del progetto principale di questa tesi.

3.3.1 Programmazione ATmega 328-P con Arduino UNO *R3*

In fase di progettazione è stato scelto il microcontrollore ATmega 328-P in maniera tale da velocizzare il processo di programmazione tramite la piattaforma Arduino. Il programma scritto con *Arduino Software 1.6.4* (*IDE*) *open source* per sistemi operativi Windows-Based, è stato scritto per fare riconoscere alla scheda Arduino UNO, e di conseguenza al microcontrollore ATmega 328-P, un carattere in codifica ASCII all'entrata di un pin della board. A seguito di ciò la scheda deve rispondere alzando il livello logico di un unico pin in uscita deciso a priori dal programmatore.

Per entrare più nello specifico verrà ora riportato il codice:

```
#include <SoftwareSerial.h>
SoftwareSerial PortaSeriale(7, 8); //RX - TX
char EchoChar;

void setup()  {

pinMode(7, INPUT); //setup Input Pin
pinMode(8, OUTPUT); //setup Output Pin

PortaSeriale.begin(9600); //setup BaudRate
pinMode(0, OUTPUT); //char 0
pinMode(1, OUTPUT); //char 1
pinMode(2, OUTPUT); //char 2
pinMode(3, OUTPUT); //char 3
pinMode(4, OUTPUT); //char 4
pinMode(5, OUTPUT); //char 5
pinMode(6, OUTPUT); //char 6
pinMode(A0, OUTPUT); //char 7
pinMode(A1, OUTPUT); //char 8
pinMode(9, OUTPUT); //char 9
pinMode(10, OUTPUT); //char a
pinMode(11, OUTPUT); //char b
pinMode(12, OUTPUT); //char c
pinMode(A2, OUTPUT); //char d
pinMode(A3, OUTPUT); //char e
pinMode(A4, OUTPUT); //char f
 }

void loop() {

EchoChar = PortaSeriale.read(); //lettura da rs232

      switch(EchoChar)
        {
          case '0':
          PORTD = B00000001;
          PORTB = B00000000;
          PORTC = B000000;
          break;
          case '1':
          PORTD = B00000010;
          PORTB = B00000000;
          PORTC = B000000;
          break;
          case '2':
          PORTD = B00000100;
          PORTB = B00000000;
          PORTC = B000000;
          break;
          case '3':
          PORTD = B00001000;
          PORTB = B00000000;
          PORTC = B000000;
          break;
          case '4':
          PORTD = B00010000;
          PORTB = B00000000;
          PORTC = B000000;
          break;
          case '5':
          PORTD = B00100000;
          PORTB = B00000000;
          PORTC = B000000;
          break;
          case '6':
          PORTD = B01000000;
          PORTB = B00000000;
          PORTC = B000000;
          break;
```

```
    case '7':
    PORTD = B00000000;
    PORTB = B00000000;
    PORTC = B000001;
    break;
    case '8':
    PORTD = B00000000;
    PORTB = B00000000;
    PORTC = B000010;
    break;
    case '9':
    PORTD = B00000000;
    PORTB = B00000010;
    PORTC = B000000;
    break;
    case 'a':
    PORTD = B00000000;
    PORTB = B00000100;
    PORTC = B000000;
    break;
    case 'b':
    PORTD = B00000000;
    PORTB = B0001000;
    PORTC = B000000;
    break;
    case 'c':
    PORTD = B00000000;
    PORTB = B0010000;
    PORTC = B000000;
    break;
    case 'd':
    PORTD = B00000000;
    PORTB = B00000000;
    PORTC = B000100;
    break;
    case 'e':
    PORTD = B00000000;
    PORTB = B00000000;
    PORTC = B001000;
    break;
    case 'f':
    PORTD = B00000000;
    PORTB = B00000000;
    PORTC = B010000;
    break;
    case 'r':
    PORTD = B00000000; // azzera i pin digitali da 0 a 7
    PORTB = B00000000; // azzera i pin digitali da 8 a 13
    PORTC = B000000; // azzera i pin analogici da A0 a A5
    break;
   }
 delay(100);
}
```

Pur essendo un linguaggio open source sviluppato dal team Arduino, si può subito intuire che le analogie col linguaggio *C/C++* sono molto forti. Gli utenti a conoscenza di tale linguaggio hanno la possibilità di poter programmare da subito senza requisiti preliminari.

Alcune procedure come ad esempio il setup della frequenza di clock a 16 MHz del microcontrollore, la possibilità di dialogare col software di programmazione, l'associazione dei pin fisici del microcontrollore con quelli della scheda Arduino (Fig. 3-9), vengono automaticamente definiti

dal *bootloader*. Il *Bootloader* non è altro che una parte di codice scritto dall' *IDE Arduino* in linguaggio macchina e precaricato sul microcontrollore della board. Tale codice non è ovviamente visibile tra quello sopra riportato. Utilizzare nel progetto globale un microcontrollore identico a quello nativo sulla scheda Arduino UNO *R3* ha permesso una notevole riduzione dei tempi di sviluppo del software risparmiando la scrittura, di per se piuttosto complicata, di un *bootloader* ad-hoc.

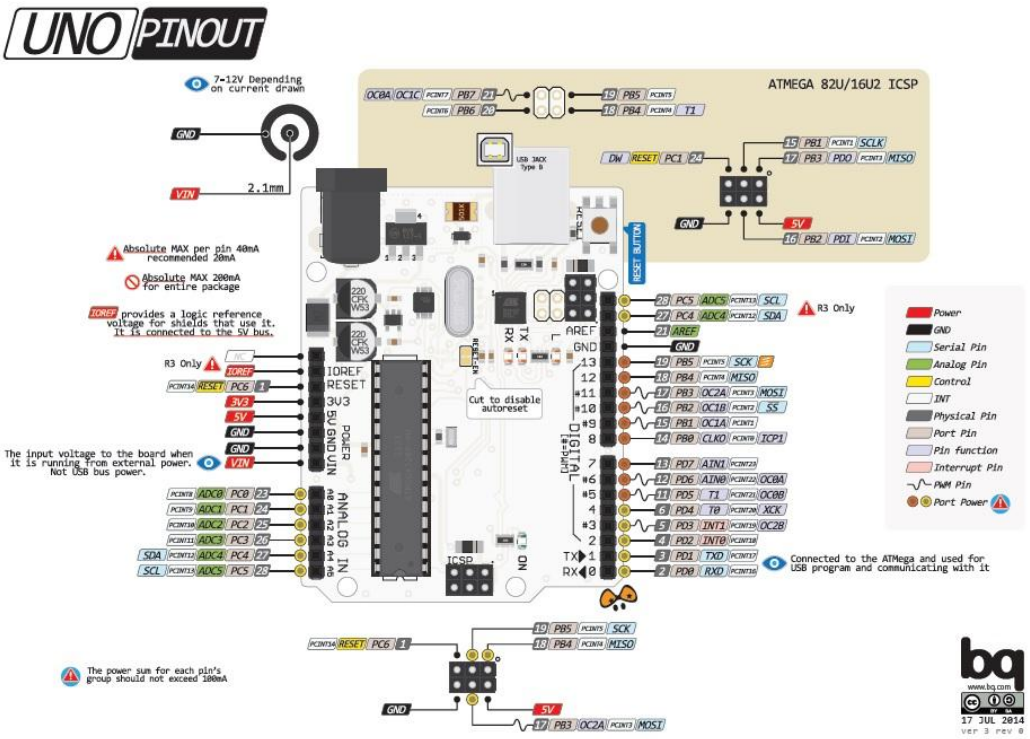

Fig. 3-9 Pinout Arduino UNO *R3*

Risulta importante specificare che durante la programmazione ad alto livello tutti gli oggetti del codice, come ad esempio pinMode, fanno riferimento ai pin della scheda Arduino (Fig. 3-9) e non ai pin del microcontrollore.

Durante la progettazione dello schematico della scheda JTAG, che verrà descritta nei prossimi paragrafi, si renderà quindi necessaria una particolare attenzione ai collegamenti dei pin del microcontrollore ATmega 328-P. Questi ultimi infatti non corrispondono a quelli della scheda

Arduino. L'associazione dei pin del microcontrollore con quelli della board Arduino è semplificata dagli schematici messi a disposizione dal team Arduino essendo esso stesso un progetto open source.

Tutti gli oggetti e le funzioni che permettono il dialogo di Arduino UNO *R3* con una porta seriale, sono situati all'interno di un file header nominato SoftwareSerial.h[3]. All'inizio del software viene dichiarata una variabile di tipo char e di nome *EchoChar* che permetterà in seguito di memorizzare al suo interno il carattere ASCII letto da una porta seriale. Viene inoltre inizializzato un oggetto di tipo *SoftwareSerial* e di nome *PortaSeriale* che permette di dichiarare quali pin di Arduino riceveranno il segnale seriale e quali lo trasmetteranno[4].

Dentro alla procedura *Void setup()* che indica il blocco di configurazione dei pin, vengono dichiarati e configurati 16 pin come output di tensione. All'interno di questo blocco viene inoltre impostato il baudrate di comunicazione tra porta seriale e board Arduino pari a $9600\ bps$.

Dentro alla procedura *Void loop()* c'è il nucleo del programma. Il codice all'interno di questa procedura viene ripetuto costantemente dal microcontrollore mentre è alimentato. All'inizio di questo ciclo la funzione .read legge lo stream di dati standard RS232 dal convertitore UM232R che verrà introdotto nel prossimo paragrafo. Ogni volta che nello stream RS232 passa un carattere ASCII standard, esso viene inserito nella variabile EchoChar. A questo punto comincia un controllo condizionale strutturato a forma di case che, una volta riconosciuto il carattere letto dalla board, esegue un azione specifica. Le azioni eseguite sono tutte del tipo *Port manipulation*, ovvero azioni che permettono di impostare in maniera veloce i registri del microcontrollore andando così ad alzare o abbassare i livelli logici di un determinato pin della board; in particolare il comando PORTD

[3] Le informazioni sulla libreria sono liberamente accessibili nel *Language Reference* del sito di Arduino
[4] La dichiarazione di un pin di trasmissione non è necessaria ai fini di questo progetto. Si è deciso comunque di riservare un pin del microcontrollore alla trasmissione del segnale seriale nel caso in cui in futuro si presenti questa necessità.

permette di controllare i registri dei *digital pin 0 to 7*, PORTB controlla i registri dei *digital pin 8 to 13*¸ mentre PORTC i registri degli *analog pin 0 to 5*[5].

Eseguita la condizione del case il ciclo riparte dalla funzione .read dopo un intervallo di $100\ ms$.

[5] Di norma si userebbero dei valori esadecimali per configurare un registro. In questo caso si è preferito usare dei valori binari semplificando così la lettura del programma. Essendo l'ATmega 328-P un microcontrollore a 8bit sono richiesti solo 8 valori binari per poter indirizzare i registri a piacimento. Impostando ad 1 il valore del registro si otterrà un livello logico alto nel pin corrispondente a tale registro; viceversa, impostando a 0 il registro si abbasserà il livello logico del pin corrispondente.

3.4 UM232R USB–Serial UART

È un ricevitore-trasmettitore asincrono (Fig. 3-10) che consente di convertire uno standard USB in uno standard RS232. È utilizzato nel progetto come convertitore USB collegato ad un PC che consente l'invio di un segnale seriale codificato RS232 al pin d'ingresso dell'ATmega 328-P.

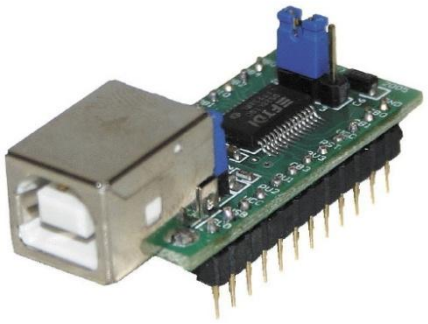

Fig. 3-10 UM232R USB-Serial UART

Il dispositivo è pienamente compatibile con la tecnologia TTL e di conseguenza col microcontrollore. Una volta collegato al PC non necessita di una tensione di alimentazione aggiuntiva per il funzionamento. Ai fini del progetto vengono utilizzati esclusivamente i pin 1 e 24 (Fig. 3-11) rispettivamente TXD e GND

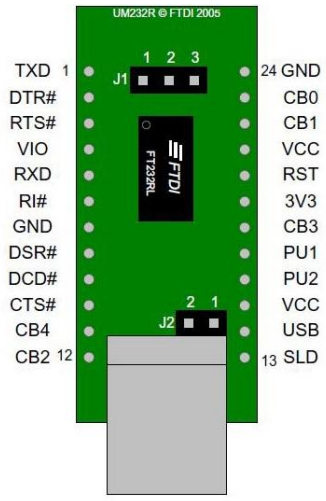

Fig. 3-11 Pin out UM232R USB-Serial UART

A seguito vengono riportati i parametri caratteristici di questo dispositivo

Parameter	Description	Minimum	Typical	Maximum	Units	Conditions
VCC1	VCC Operating Supply Voltage	4.0	---	5.25	V	
VCC2	VCCIO Operating Supply Voltage	1.8	---	5.25	V	
Icc1	Operating Supply Current	---	15	---	mA	Normal Operation
Icc2	Operating Supply Current	50	70	100	µA	USB Suspend

Fig. 3-12 Parametri caratteristici UM232R USB-Serial UART nel range -40 to 85 °C

Parameter	Value	Unit
Storage Temperature	-65°C to 150°C	Degrees C
Floor Life (Out of Bag) At Factory Ambient (30°C / 60% Relative Humidity)	168 Hours (IPC/JEDEC J-STD-033A MSL Level 3 Compliant)*	Hours
Ambient Temperature (Power Applied)	-40°C to 85°C	Degrees C
VCC Supply Voltage	-0.5 to +6.00	V
D.C. Input Voltage – USBDP and USBDM	-0.5 to +3.8	V
D.C. Input Voltage – High Impedance Bidirectionals	-0.5 to + (VCC +0.5)	V
D.C. Input Voltage – All Other Inputs	-0.5 to + (VCC +0.5)	V
D.C. Output Current – Outputs	24	mA
D.C. Output Current – Low Impedance Bidirectionals	24	mA
Power Dissipation (VCC = 5.25V)	500	mW

Fig. 3-13 Valori massimi UM232R USB-Serial UART

3.5 SN74CBT3384ADW 10-BIT FET Bus Switch

L'SN74CBT3384ADW (Fig. 3-14) è uno switch digitale composto da transistor ad effetto di campo in tecnologia MOS con elevata velocità di commutazione. Compatibile con lo standard TTL e quindi col microcontrollore ATmega 328-P, è realizzato in logica NMOS ad arricchimento (Fig. 3-15)

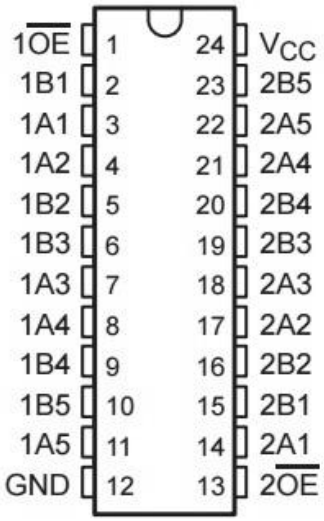

Fig. 3-14 SN74CBT3384ADW Top View

È dotato di 10 linee di ingresso (1A1-1A5 ; 2A1-2A5) e 10 linee di uscita (1B1-1B5 ; 2B1-2B5) controllate da due *OE* (*Output Enable*) attivi bassi.

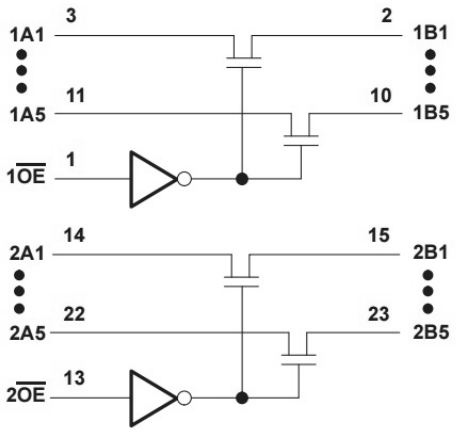

Fig. 3-15 Schema logico SN74CBT3384ADW

Ogni *OE* attivo basso è collegato al *gate* di ogni NMOS passando prima per un invertitore di logica. Se sull'OE è presente un livello di tensione basso, le 5 linee di Input corrispondenti vengono collegate alle 5 linee di Output. Viceversa, se la tensione sull'OE è ad un livello logico alto, allora lo switch si comporta come un interruttore aperto tra le linee di Input e quelle di Output.

Le principali caratteristiche[6] verranno ora riportate in Fig. 3-16

		SN74CBTD3384		UNIT
		MIN	MAX	
V_{CC}	Supply voltage	4.5	5.5	V
V_{IH}	High-level control input voltage	2		V
V_{IL}	Low-level control input voltage		0.8	V
T_A	Operating free-air temperature	−40	85	°C

Fig. 3-16 Principali caratteristiche SN74CBT3384ADW

[6] Altre caratteristiche, datasheet e schematici si possono consultare nella web page del produttore. http://www.ti.com

3.6 Schema e principio di funzionamento della scheda di controllo.

Per concludere l'ultimo capitolo di questa trattazione, dopo aver discusso il funzionamento di ogni componente che costituisce la scheda è bene fornire un'idea del funzionamento globale della scheda stessa (Fig. 3-17).

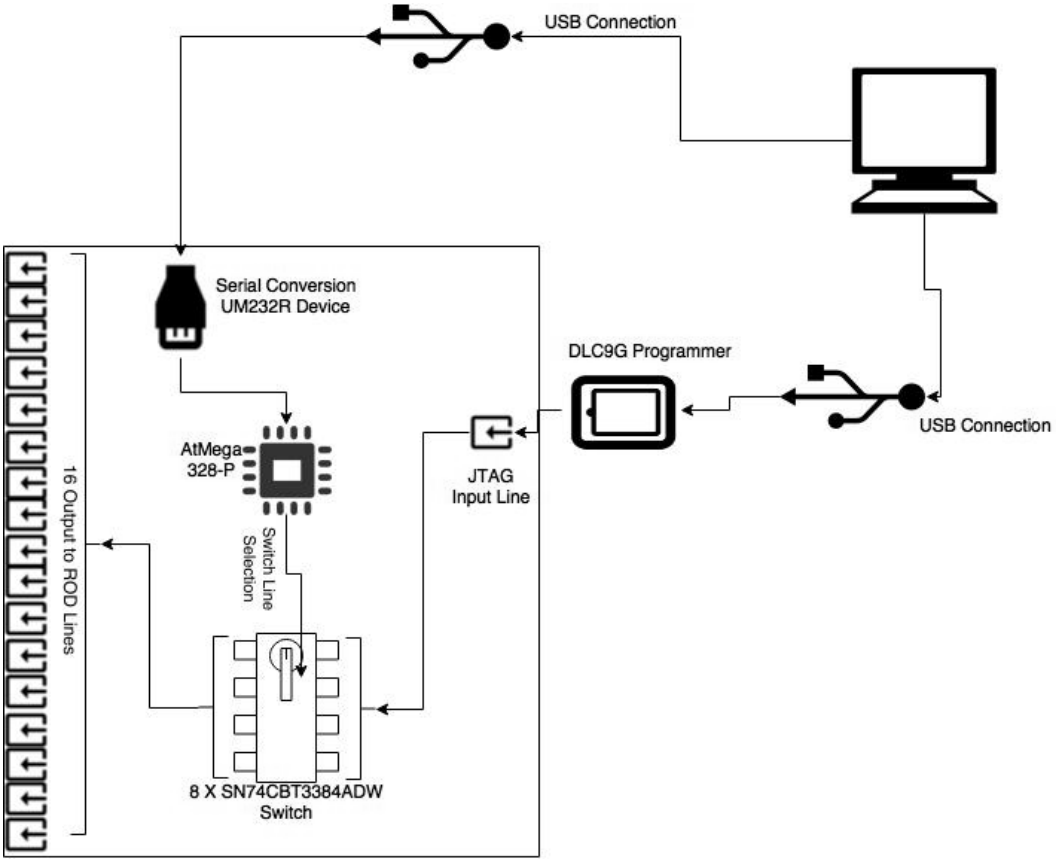

Fig. 3-17 Diagramma del progetto globale

Da un computer è possibile inviare un segnale di programmazione JTAG al programmatore *DLC9G*. Questo segnale viene trasferito all'interno della scheda di controllo alle entrate di tutti gli 8 switch. Per abilitare un'uscita dello switch e mettere in collegamento fisico il segnale JTAG in entrata con una delle 16 uscite della scheda è necessario inviare un carattere ASCII compreso tra *'0'* e *'16'* alla scheda. Il carattere inviato tramite

protocollo USB viene prima convertito in protocollo seriale RS232 ed inviato al microcontrollore. Il microcontrollore, riconoscendo il carattere ricevuto, attiverà di conseguenza un *OE* di uno degli 8 switch SN74CBT3384ADW, consentendo così la trasmissione del segnale JTAG dall'entrata fino all'uscita. Se dal computer venisse inviato il carattere ASCII *'r'* alla scheda, il microcontrollore verrebbe resettato bloccando così ogni Output della scheda.

È bene inoltre specificare che gli *OE* degli switch si attivano ad un livello logico basso, mentre il firmware scritto sul microcontrollore, per semplicità di lettura, è stato progettato per fornire un livello alto di Output. Ovviamente questo problema è facilmente risolubile eseguendo un'inversione di logica nel codice scritto nel microcontrollore.

Lo schema elettrico della scheda (Fig. 3-18) è stato disegnato con la suite open source KiCAD[7]. Questo software, oltre ad essere open source, è sostenuto da una grande community di utilizzatori, garantisce un supporto multipiattaforma ed è particolarmente facile seppur estremamente completo e versatile.

[7] http://www.kicad-pcb.org

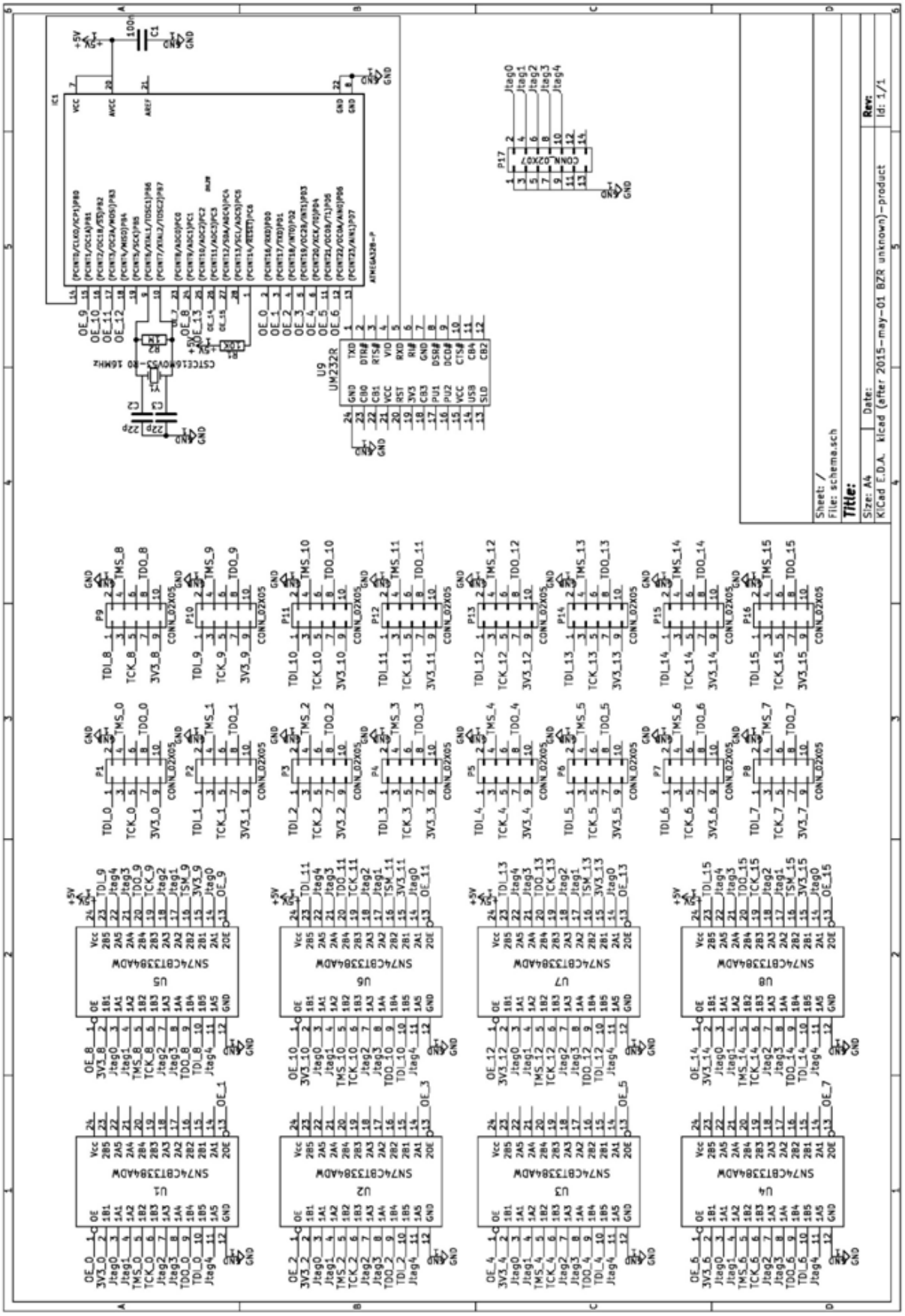

Fig. 3-18 Schema elettrico Scheda di controllo JTAG

Conclusioni

Ad oggi, luglio 2015, è stato concluso il progetto della scheda e sono stati testati i vari componenti in maniera indipendente costruendo degli ambienti hardware ad hoc di prototipazione e di test. Lo schema elettrico (Fig. 3-18) per il progetto finale della scheda è già stato sviluppato e corretto. Il passo successivo sarà quello di produrre ed ottimizzare un footprint che permetterà di realizzare il circuito stampato (*PCB*) vero e proprio.

Durante la stesura del progetto si è posta particolare attenzione non solo alla scelta dei componenti ma anche all'ottimizzazione dei costi. Ciò rende possibile un'eventuale produzione in serie qualora dovesse essere prevista.

È evidente che la scheda in questione, pur essendo nata per uno scopo specifico, può essere utilizzata in un qualsiasi ambito in cui si renda necessario il test o la programmazione di dispositivi che utilizzino il protocollo *JTAG*. Con un solo programmatore *JTAG* è infatti possibile, tramite questa scheda, pilotare fino a 16 dispositivi, permettendo un risparmio sia dal punto di vista economico che da quello delle risorse. Una singola porta di comando, infatti, difficilmente riuscirebbe a gestire gli indirizzi di 16 diversi programmatori *JTAG* collegati simultaneamente alle sue porte USB; con questa scheda si potrebbero snellire processi di questo genere.

Bibliografia

[1] CERN. “The Large Hadron Collider”. http://home.web.cern.ch/topics/large-hadron-collider Web. 06/2015

[2] INFN. “ATLAS – A Thoroidal Lhc ApparatuS”. http://lhcitalia.infn.it/index.php/atlas Web. 06/2015

[3] Xilinx. “Platform Cable USB datasheet” http://www.xilinx.com/support/documentation/data_sheets/ds300.pdf 10/12/2004. Web. 06/2015

[4] Arduino. “Arduino UNO R3”. https://www.arduino.cc/en/Main/arduinoBoardUno Web. 06/2015

[5] Milmann J. Grabel A. Terreni P. *Elettronica di Milmann 4/ed*, McGraw-Hill, 2008

[6] Atmel. “ATmega328”. http://www.atmel.com/devices/ATmega328.aspx Web. 06/2015

[7] C. Preti, “Test hardware del primo batch di schede ROD per l’esperimento ATLAS IBL”, 2013.

[8] L. Lama, “Development and Testing of the ATLAS IBL ROD preproduction boards”, 2013.