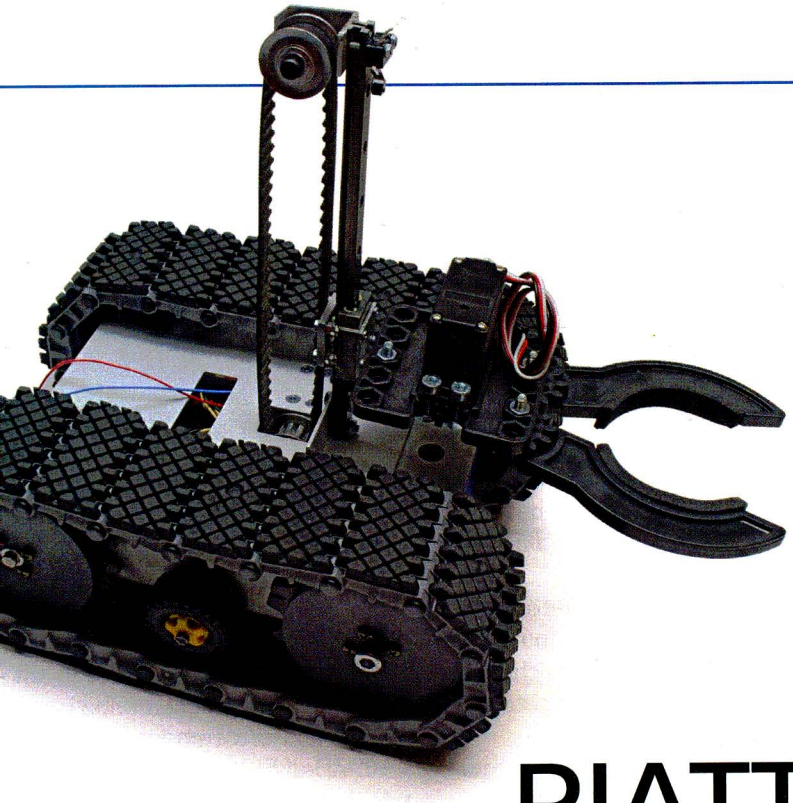


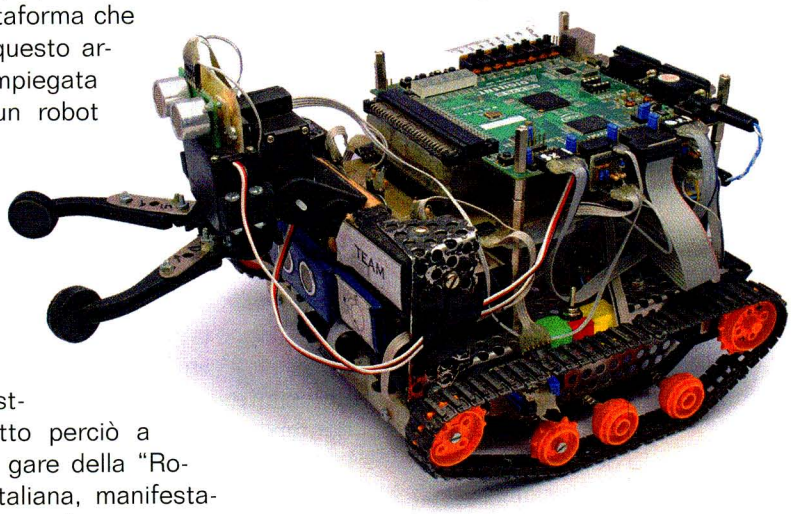
Come nasce e si sviluppa un robot didattico pensato e realizzato presso l'ITCS Erasmo da Rotterdam di Bollate. In questa prima parte vediamo l'architettura su cui si basa il sistema.

UNA PIATTAFORMA FLESSIBILE PER LA ROBOTICA

a cura del prof. GIULIO VITALE

La robotica è una disciplina tanto importante per il presente e promettente per l'immediato futuro, che si sta rapidamente facendo strada nelle scuole italiane, specie negli Istituti Tecnici e Professionali. Alcuni mesi fa abbiamo contattato l'ITCS Erasmo da Rotterdam dove è in corso lo sviluppo di un interessante robot nato a scopo didattico e che ci è sembrato interessante descrivere passo per passo sulla nostra rivista. In questa prima puntata descriveremo l'architettura generale del sistema e le premesse tecnologiche che lo hanno caratterizzato; nelle prossime illustreremo le varie soluzioni adottate, hardware e software, per la realizzazione del sistema su FPGA e della sua interfaccia con i sensori e gli attuatori. Nata due anni fa per scopi puramente didattici, la piattaforma che proponiamo in questo articolo è stata impiegata per realizzare un robot "rescue", ovvero un robot soccorritore capace di ricercare ipotetiche vittime su scenari che simulano una situazione post-catastrofe, adatto perciò a partecipare alle gare della "Robocup Junior" italiana, manifestazione giunta ormai alla sua terza edizione.

La "RoboCup Jr Italia" è la sezio-



La versione 2010 che ha gareggiato alla Robocup Jr di Vicenza.

ne italiana della Robocup internazionale, competizione con scopi didattici riservata ai ragazzi "under 19" e adatta, quindi, alle nostre scuole superiori.
In Italia la Robocup Jr è sostenuta da una "Rete di Scuole" che si prefigge di organizzare la competizione ogni anno, spostandosi di volta in volta nei vari istituti che si candidano ad ospitarla, con lo scopo di promuovere la diffusione della robotica educativa nel sistema scolastico e dando l'opportunità a docenti e studenti di cimentarsi in una competizione nazionale basata essenzialmente sul motto: "L'importante non è vincere ma imparare".
Durante la manifestazione, ragazzi provenienti da tutte le scuole italiane gareggiano con robot da loro costruiti e programmati, suddividendosi in tre categorie diverse: gioco del calcio, ricerca di vittime e danza.
La piattaforma che si vuole proporre in questa serie di articoli è nata con le stesse premesse e nella piena condivisione dello spirito che anima la Rete di Scuole a sostegno della Robocup Jr, e vuole essere un'occasione di discussione e di scambio di esperienze, con lo scopo di crescere insieme sulle diverse e possibili soluzioni tecnologiche, adatte ad affrontare le regole di gara che la manifestazione propone.
Ciò che la distingue da tutte le soluzioni viste alle due ultime competizioni italiane, svoltesi a Torino e Vicenza, è l'architettura dell'hardware, basata sul concetto di "System On Chip" per FPGA, che unisce alla disponibilità di un microprocessore RISC a 32 bit i vantaggi della totale configurabilità dell'hardware.
L'idea originaria si è sviluppata all'interno del gruppo di lavoro del Laboratorio Permanente di Robotica Didattica dell'ITCS Erasmo da Rotterdam di Bollate (MI), ed è nata avendo a mente i seguenti presupposti:

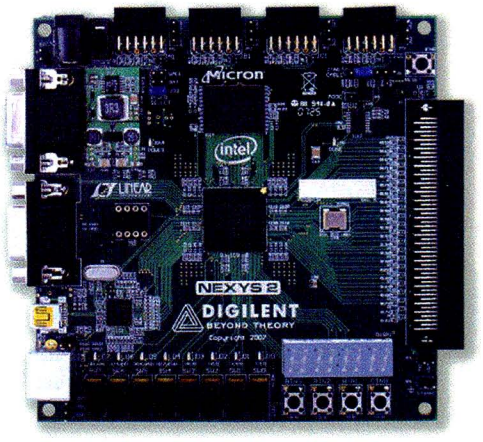
- essere totalmente riconfigurabile, dal punto di vista sia hardware che software, per consentire il massimo di libertà nella scelta della soluzione da usare e garantire la possibilità di riciclare l'hardware di base della piattaforma su altre applicazioni;
- fornire prestazioni elevate pur mantenendo costi contenuti;
- poter aggiungere periferiche senza modificare nella sostanza la struttura del sistema.

IL SISTEMA DI BASE

Vista l'esperienza positiva con la scheda didattica Nexys2 della Digilent Inc. (basata su una FPGA Spartan 3E-1200 della Xilinx e precedentemente utilizzata con buoni risultati nelle attività di laboratorio di Elettronica Digitale e Controlli) si è deciso di mantenere la stessa tecnologia.
La scheda Nexys2 (**Fig. 1**) offre un ambiente aperto, dotato di risorse adatte a valutare circuiti digitali didattici di bassa-media complessità, a costi molto limitati. Essa offre dispositivi di interfaccia come "dip-switch", pulsanti, LED, display a sette segmenti, RAM statica, memoria Flash, e un buon numero di connettori di Input/Output per connettere moduli hardware, progettati dall'utente o acquistabili da Digilent (la ricca serie Pmod), per il controllo di una svariatissima gamma di dispositivi, analogici o digitali, utili per implementare la gran parte dei sistemi didattici impiegati normalmente nei corsi di sistemi, elettronica digitale e telecomunicazioni, sia negli Istituti Tecnici che nei corsi introduttivi di livello universitario.
Il modello da noi impiegato fornisce come risorse di base:

- 8 dip-switch;

Fig. 1



- 4 pulsanti;
- 8 LED;
- 4 display a 7 segmenti;
- 16 MB di Flash;
- 16 MB di RAM;
- 1 porta USB, utile alla configurazione della FPGA e per scambiare dati con un host;
- 1 porta RS232;
- 1 porta PS/2;
- 1 interfaccia VGA;
- 16 I/O disponibili su connettori separati Pmod e 43 su connettore Hirose;
- un generatore di clock a 50 MHz.

LA CPU

Il cuore del nostro robot è il "soft-core" a 32 bit Microblaze della Xilinx, (al quale abbiamo dedicato un breve tutorial sui fascicoli 149 e 150) e che si presenta con una struttura adatta alle prestazioni da noi attese, offrendoci, inoltre, la possibilità di fruire dei vantaggi offerti dall'ambiente di progetto integrato XPS (Xilinx Platform Studio) e dall'SDK (Software Development Kit), dove con un unico "frame work" è possibile sviluppare, parallelamente, hardware e software e verificare i risultati con tecniche di "co-debugging" integrato.
Il Microblaze si presenta come un'architettura Harward dotata di quattro bus principali separati, un set d'istruzioni RISC molto snello con istruzioni uniformi e ortogonali

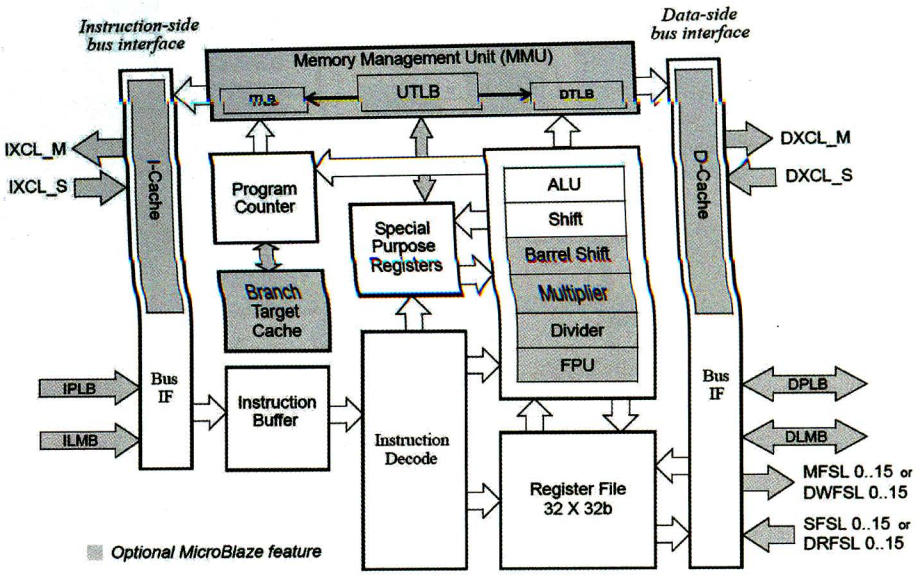
da 32 bit ognuna, un BUS dati a 32 bit e 32 registri della medesima dimensione. Come si può vedere nella **Fig.2** e nella **Fig. 3**, partendo dai blocchi fondamentali, indicati in bianco, e in funzione delle specifiche esigenze, il "core" si espande con i blocchi, indicati in grigio nella figura, che possono man mano essere aggiunti per adattarlo alle particolari prestazioni richieste, senza dover ritoccare l'intero sistema.

In particolare, i moduli che possono essere successivamente connessi sono i seguenti:

- una MMU derivata dal modello Power PC 405, che può supportare "kernel", come quello di Linux, che ne esigono la presenza per il controllo della memoria virtuale;
- una cache dedicata ai salti, associata a un sistema di previsione, per limitare il calo delle prestazioni dovute alle possibili condizioni di stallo della pipeline nei "branch";
- due memorie cache, una per le istruzioni e una per i dati, dimensionabili, ognuna, da 64 a 64K byte a scelta del progettista e connesse con un BUS separato;
- fino a 16 canali di interfacciamento in input/output, gestibili tramite link seriali (FSL), con un meccanismo simile a una FIFO, accessibili direttamente con istruzioni dedicate, per la realizzazione di coprocessori specifici, caratterizzati da bassa latenza nel trasferimento dei dati;
- moltiplicazione e supporto alla divisione implementati in hardware;
- Floating Point Unit a singola precisione compatibile con standard IEEE 754;
- Barrell shifter, per la realizzazione di shift multipli in un solo ciclo macchina.

Nell'applicazione che presentiamo, sono state selezionate soltanto le seguenti risorse basilari:

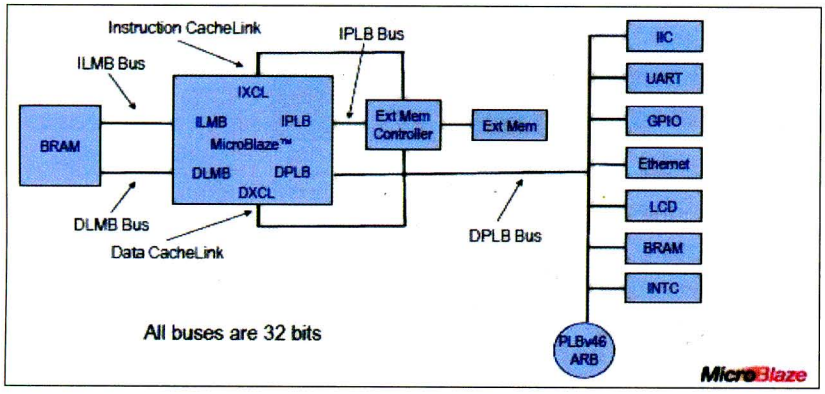
- i due doppi bus, istruzioni/dati, per l'accesso simultaneo alla memoria locale interna e alle periferiche, ovvero il BUS LMB, con 32 kB di memoria interna accessibile con un solo ciclo di clock, e il bus PLB per l'accesso alla memoria esterna e alle periferiche;
- Floating Point Unit, moltiplicatore e divisore hardware, barrel shifter.



Quali sono i vantaggi dell'impiego di una simile architettura? Non è obbligatorio essere dei progettisti di sistemi" embedded" basati su FPGA per rendersi conto di qual è il grado di libertà disponibile con questo approccio, che unisce la flessibilità di poter modellare, oltre al software, anche l'hardware; ma ciò che più si vuole mettere in evidenza in questo caso è legato ad altri tre fattori, tutti egualmente importanti e significativi, che hanno tracciato la strada seguita.

1. Innanzitutto il robot nasce per uno scopo didattico, quindi deve essere aperto, in modo che gruppi di studenti diversi possano pensare simultaneamente alla realizzazione delle diverse periferiche di controllo di sensori e attuatori, controllandone completamente le funzionalità, il progetto e la verifica.
2. Nel dimensionamento delle prestazioni, una FPGA offre costantemente la possibilità di scegliere la strada ottimale, in termini di efficienza di elaborazione, tra la realizzazione in hardware e quella in software di ogni funzione, eliminando tutti gli eventuali colli di bottiglia. Per esempio nel nostro caso è stato scelto un "tick" base di 10 millisecondi e, al massimo, in questo intervallo di tempo tutta la ricca dotazione di sensori deve

Fig. 3



essere stata letta e tutti gli attuatori devono essere aggiornati con i valori coerenti alle condizioni di controllo decise dal programma.

3. Lo stesso hardware deve essere riadattabile alle specifiche condizioni sperimentali che si vogliono affrontare di anno in anno in eventuali corsi didattici differenti, recuperando interamente le parti più costose per riutilizzarle in differenti esperienze.

CARATTERISTICHE DI UN ROBOT "RESCUE" PER LA ROBOCUP JR

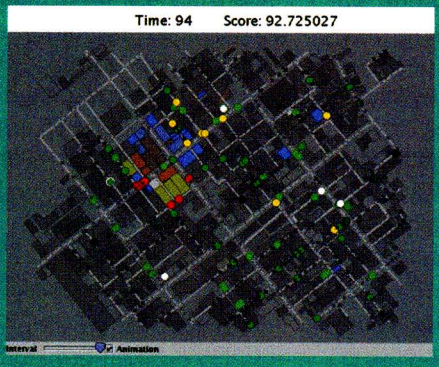
La scelta di realizzare un robot di questo genere è nata circa due anni fa in un progetto di "Scuola Aperta". All'interno di un corso pomeridiano, con partecipazione volontaria da parte degli studenti, ci eravamo posti il problema di pensare in che modo le "macchine" possono ricoprire un ruolo umanitariamente utile. Tra le tante strade percorribili, qualcuno degli studenti partecipanti trovò su Internet il sito della Robocup Jr, indetta quell'anno per la prima volta in Italia proponendo due discipline di gara riservate a robot rispettivamente che giocano al calcio e che simulano la ricerca di vittime (robot "rescue"). Chiaramente i secondi, in quanto dedicati alla ricerca e al soccorso di superstiti di una ipotetica catastrofe, erano la categoria più vicina al nostro obiettivo, pertanto decidemmo di partecipare pensando di costruire un nostro robot dedicato a questa funzione, partendo da quello che avevamo già a disposizione nel nostro magazzino.

Cercammo di identificare i sensori e gli attuatori che potevano essere utili a una macchina candidata a presentarsi con le carte in regola per questo ruolo e decidemmo per le seguenti funzioni:

- **luminosità inferiore**; il robot deve riuscire a seguire un percorso guidato da una linea nera su sfondo bianco, quindi deve essere munito di uno o più sensori di luminosità in grado di rilevare le differenze di intensità di luce bianca necessarie a identificare il riferimento;

Nato per partecipare alla gara

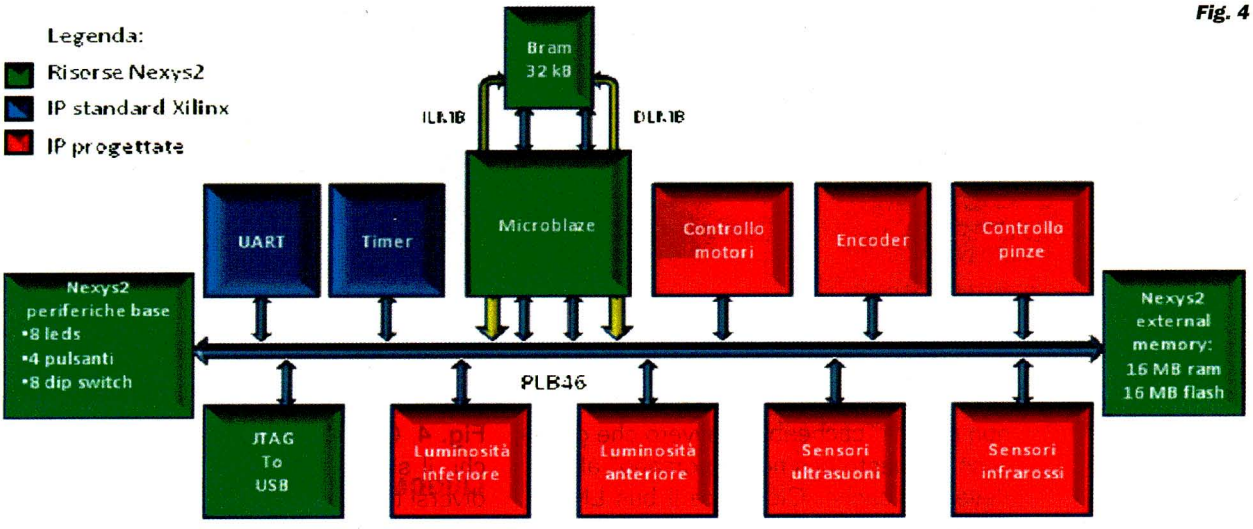
Il robot realizzato dall'Istituto Erasmo da Rotterdam è stato costruito per operare alle condizioni imposte dal regolamento della categoria Rescue della competizione Robocup Jr 2011; gli automi in gara devono salvare una "vittima", costituita da una lattina (capacità 330 ml, diametro 66 mm, altezza 116 mm) rivestita di carta stagnola e del peso di 150 grammi (distribuito in modo da tenere il baricentro al centro dell'oggetto) che può non essere di materiale ferromagnetico, quindi per manipolarla i robot non potranno usare calamite. La vittima sarà inizialmente "in piedi" e dovrà essere spostata fino al punto di deposizione (indicato da un triangolo nero e sollevata di 6 cm rispetto al pavimento) in uno qualunque dei quattro angoli della terza stanza.



La gara ha inizio ponendo il robot all'ingresso della prima "stanza" ed avviandolo manualmente. Ogni robot ha 8 minuti di tempo per esplorare le tre stanze da cui è composto il campo; la struttura portante del campo potrà essere realizzata in acciaio, quindi i robot non devono usare sensori magnetici (bussole, proximity, ecc.) perché potrebbero dare indicazioni alterate; inoltre, siccome le pareti delle stanze potranno essere realizzate in materiale trasparente (tipo plexiglass), anziché in legno o masonite, va tenuto conto della diversa riflessione degli ultrasuoni.

Sul pavimento sarà stampato un reticolo di sottili linee nere (larghezza massima 1 mm) in modo da consentire ai robot di orientarsi; sul cammino verranno

Fig. 4



- **colore**; le vittime da cercare possono essere identificate da un colore, pertanto il robot occorre che abbia l'abilità di discernere le componenti cromatiche della luce riflessa da queste;
- **ostacoli vicini**; il robot deve essere in grado di rilevare eventuali macerie che possono ostacolare il suo cammino e che risulta più conveniente evitare invece che scavalcare;
- **oggetti lontani**; deve essere necessario poter individuare oggetti voluminosi, posti a una certa distanza e che si staccano nettamente dal contesto;
- **inclinazione**; può essere utile la presenza di un giroscopio/accelerometro per identificare un tragitto in piano o la presenza di una inclinazione o di un dislivello nel territorio;
- **pinza**; alcune macerie richiedono la rimozione oppure l'eventuale vittima richiede di essere sollevata e portata in salvo in un luogo sicuro;
- **motori**; devono essere controllati in velocità e deve essere possibile rilevare l'entità dello spostamento.

posti ostacoli (macerie) che possono essere inamovibili ed avere un'altezza da terra di 10 mm. Gli ostacoli mobili possono essere, a scelta, superati oppure spinti fuori dal percorso. L'ingresso alla terza stanza sarà identificato da una striscia di stagnola larga 25 mm posta di traverso sul pavimento per tutta la larghezza della porta; questa stanza non ha uscita. Il salvataggio sarà considerato completato nel momento in cui la lattina verrà appoggiata sulla zona di deposizione (la cui parete laterale sarà dipinta di nero) anche se il robot non abbandonerà la presa. Non esistono più limitazioni di grandezza per i robot, che però devono essere costruiti in modo da attraversare le aperture fra una stanza e l'altra, che hanno dimensioni di 250x250 millimetri.

COME CONNETTERE AL MICROBLAZE LE PERIFERICHE INDIVIDUATE

Come già precedentemente detto, il nostro soft-core è una macchina RISC con un'architettura Harward e ogni suo bus ne mantiene la natura, scomponendosi sempre in due canali paralleli dedicati, l'uno agli scambi tra Unità di Controllo e memoria delle istruzioni, e l'altro al trasferimento dei dati tra memorie/periferiche e Unità Aritmetico-Logica, passando tramite 32 registri "general purpose". Infatti, a tal proposito, essendo il Microblaze una macchina con set d'istruzioni ridotto, tutte le operazioni logico-matematiche sono effettuate tra registro e registro, mentre tutti i trasferimenti da e per la memoria avvengono tramite istruzioni di "load" e "store". Come si può vedere dalla **Fig.2**, il core Microblaze è dotato di due bus paralleli principali dedicati all'interfacciamento con memorie e periferiche; i bus sono quelli descritti di seguito.

1) **LMB** (Local Memory Bus) diviso, a sua volta, in due Bus dedicati: ILMB (Istruction Side LMB) e DLMB (Data Side LMB).

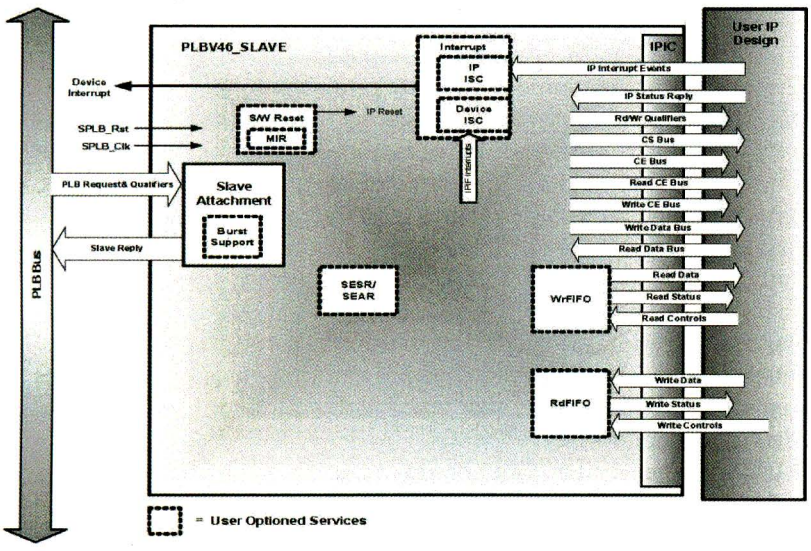
- Bus infrastructure and bridge cores
  - Fabric Co-processor Bus (FCB)
  - Fast Simplex Link (FSL)
  - OCM bridge and bus
  - PLB v46-to-PLB v46 bridge
  - LMB (MicroBlaze™ processor)
  - Debug
  - ChipScope™ Pro tool (ILA, controller, PLB v46, for example)
  - MicroBlaze Debug Module (MDM)
  - PowerPC processor JTAG controller
- Peripherals
  - SPI interface, UART (Lite)
  - Hard-core triple-mode Ethernet MAC
  - 10/100 Ethernet MAC Lite
- Memory and memory controller cores
  - Block RAM controller (PLB v46/_MB)
  - PowerPC 440 processor DDR2 memory controller
  - Multi-port memory controller (SDRAM, DDR, DDR2, DDR3)
  - Multi-channel external memory controller (SRAM, Flash)
  - System ACE™ interface controller (Compact Flash)
- Arithmetic
  - Floating Point Unit (FPU)
- Timers
  - Watchdog, fixed interval
- Inter-processor communication
- External peripheral controller
- DMA controller
- PCI
  - PLB v46 IP interface to PCI Express bridge
  - PCI arbiter
- User core template
  - PLB slave attachment
  - PLB master attachment
- Other cores
  - System monitor
  - Digital-to-analog converter
  - Analog-to-digital converter
  - Clock Generator
  - PLL and DCM clock modules
  - System reset module
  - Interrupt controller

Fig. 5

LMB è un bus dedicato essenzialmente alla connessione tra CPU e le memorie interne alla FPGA (Block RAM), caratterizzato da una bassa latenza e dal fatto di non essere "cacheable", ovvero che gli accessi a questo bus non "sporcano" alcuna delle due cache. Ciò rende il bus LMB il candidato ideale per contenere segmenti di codice dedicati al servizio degli interrupt (ISR) e a funzioni critiche del kernel, oppure a tabelle del sistema operativo. Tutti gli accessi a questo bus durano una singola fase di clock.

2) **PLB46** (Processor Local Bus Vers.4.6), compatibile con le specifiche indicate da IBM nello standard Core Connect, nato apposta per l'interconnessione "on-chip" dei sistemi embedded. Il PLB è un bus a elevate prestazioni suddiviso in tre sezioni: un bus per gli indirizzamenti e due per il trasferimento dei dati (uno dedicato alla scrittura, l'altro alla lettura) e garantisce, quindi, più trasferimenti simultanei. È un bus arbitrabile che accetta connessioni single e multi master ed in esso gli spazi di memoria associati alle memorie esterne sono collegabili a due cache separate (per istruzioni e dati) configurabili a piacere e dotate di un canale di connessione dedicata, detta "Cache Link", attraverso il quale l'aggiornamento delle cache non intralcia il flusso sul bus principale. Un generico "System On Chip", basato sulla concomitante presenza dei due tipi di interconnessione, è quello indicato nella **Fig. 3**.

Fig. 6



SCHEMA A BLOCCHI DEL SISTEMA

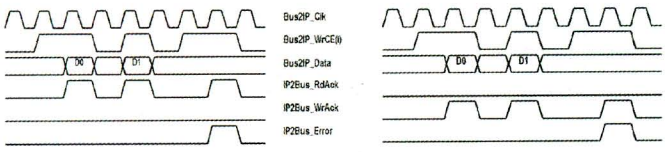
Con queste specifiche in mente è stato deciso di realizzare il sistema indicato in **Fig. 4**. Come si vede dallo schema a blocchi, il sistema ottenuto è una miscela di diversi tipi di interfacciamenti, di cui alcuni sono presenti per impostazione predefinita, come risorse di base fornite sulla scheda madre, mentre altri sono moduli standard, scelti da un carnet di IP (Intellectual Properties) offerti dal sistema di sviluppo della Xilinx, in funzione delle specifiche esigenze di comunicazione e di temporizzazione del robot; infine, tutti i restanti sono le interfacce progettate ad hoc per i nostri specifici sensori e attuatori.

Siamo riusciti in questo modo a "cucirci" addosso un hardware ritagliato sulle nostre specifiche esigenze, scaricando il software dall'enorme carico di lavoro necessario sia a rilevare e condizionare i dati provenienti dai sensori, sia ad effettuare il controllo degli attuatori, lasciandogli soltanto il compito di effettuare la parte "intelligente" del sistema, ovvero la preparazione, la guida e le comunicazioni con l'esterno.

Questa è la chiave che apre la comprensione della soluzione proposta: ognuno dei controllori di periferica è specializzato sulle caratteristiche del sensore/attuatore gestito e opera in modo parallelo alla CPU, richiedendo un intervento trascurabile da parte di quest'ultima. Il Microblaze si viene a trovare così al centro di un insieme di "co-processori dedicati", che evolvono simultaneamente, spezzando la tipica logica sequenziale cui necessariamente deve sottostare un sistema "multi task" governato da un unico processore; il suo unico compito sarà concentrarsi sulla strategia di evoluzione della gara.

I MODULI STANDARD OFFERTI DA XILINX

Per configurare il proprio sistema, Xilinx



offre un ricco menu di moduli configurabili all'interno della propria piattaforma XPS (**Fig. 5**); il progettista, con un semplice sistema di "pick and place", può inserire la periferica desiderata, connetterla al bus e "customizzarla" modificando i parametri configurabili dall'utente.

COME REALIZZARE I PROPRI MODULI D'INTERFACCIAMENTO

Se invece vogliamo creare le nostre periferiche ad hoc, dobbiamo scrivere dei moduli hardware in VHDL da connettere a uno dei bus disponibili: nel nostro caso al bus parallelo di interconnessione con le periferiche, ovvero al PLB46.

Non si può dire che questo sia un bus semplice da interfacciare, anzi il numero dei segnali che lo compongono e il rispetto rigoroso delle specifiche e delle sequenze protocollari, lo rendono non tanto "user-friendly" e immediato, anche perché le prestazioni che fornisce sono abbastanza elevate da rendere critici i vincoli di temporizzazione.

Il problema è stato risolto da Xilinx, mettendo a disposizione dei progettisti un modulo standard, adattabile alle specifiche esigenze di interfacciamento di ognuna delle periferiche, il quale si interpone tra il bus e le periferiche stesse. Esso elimina tutte le difficoltà derivanti dal rispetto delle specifiche di compatibilità dello standard Core Connect e offre un sistema di colloquio con la logica dell'utente semplice e immediato, strutturato a tre livelli, due dei quali configurabili con un wizard guidato e il terzo definibile dall'utente (vedi **Fig.6**):

1. PLB46 bus attachement, disponibile come modulo master o come slave, con possibilità di trasferimento singolo o a burst;
2. IPIC (IP interconnect), semplice bus di connessione alla logica d'utente, che riproduce il classico protocollo di comunicazione parallela sincrona, "data", "address", "control";
3. il modulo VHDL "**User IP Design**",

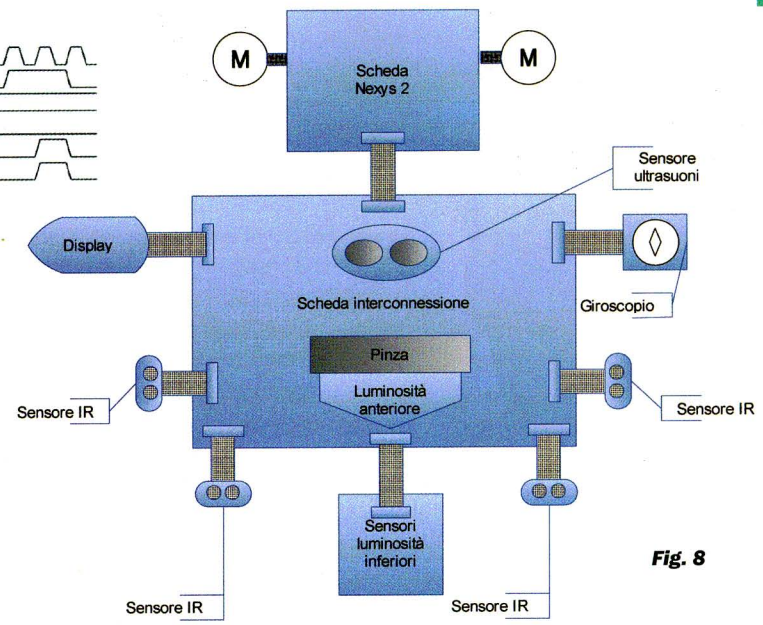


Fig. 8

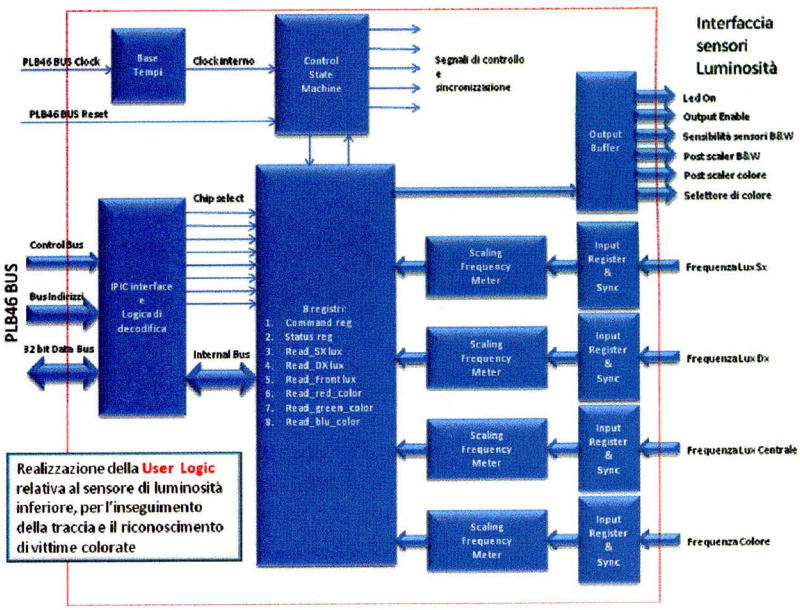


Fig. 9

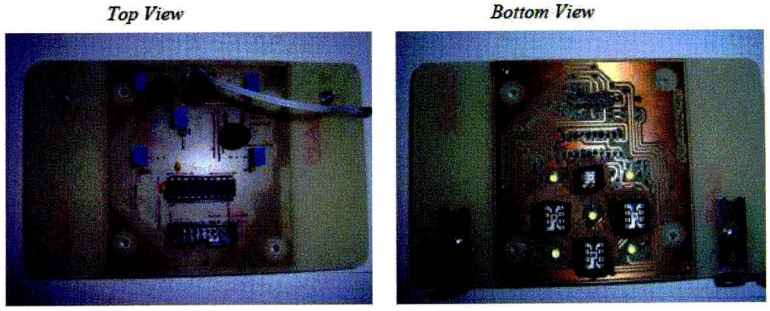
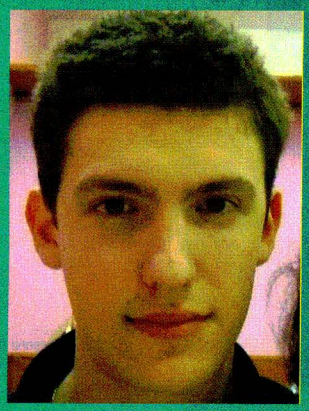


***Fig. 10** - La scheda dei sensori di luminosità.*

Dalla Robocup Jr alla robotica professionale



Stefano Bertolazzo è un giovane che si è diplomato a luglio di quest'anno con una votazione di 100/100, ed è stato un elemento portante per questi due anni di attività del Laboratorio Permanente di Robotica Didattica presso l'ITCS Erasmo da Rotterdam di Bollate, cui ha dato un contributo essenziale. In particolare, la sua competenza e la sua passione sono state trainanti per i suoi compagni nella realizzazione del modello di robot su cingoli di cui si parla in questo articolo.
La storia fin qui vissuta da Stefano è certamente una buona esemplificazione del motto "Robottando s'impara", mutuato dallo spirito della Robocup e pienamente adottato e condiviso dal laboratorio di robotica. Infatti, dopo i due anni di lavoro volontario extra curricolare, con cui ha "giocato imparando", ha avuto l'opportunità di dare subito la giusta buona impressione a chi gli ha fatto il colloquio di assunzione per lavorare presso una grande multinazionale che produce robot industriali. Stefano può ora proseguire il suo percorso, continuando a "giocare" ogni giorno, e a tempo pieno, con robot più grandi, in grado di compiere lavori complessi, dedicati alle più svariate attività e che possono automatizzare intere parti di un processo produttivo. La passione, quindi, ha premiato e reso più labile il confine tra lavoro e divertimento, consentendo anche un passaggio naturale, senza soluzione di continuità, tra la parentesi dello studio e l'ingresso nel mondo del lavoro.
Non ci resta che augurare buona fortuna a Stefano!

scritto dall'utente o acquistato da terze parti, si aggancia alla logica di interconnessione IPIC e contiene il controller della periferica specifica.

Per quanto appaia decisamente sofisticata, questa tecnica è in realtà relativamente semplice, in quanto, visto il grado di personalizzazione consentito, può essere facilmente adattata alle effettive necessità della periferica da interfacciare e ritagliata al livello di complessità desiderato dall'utente, seguendo i passi guidati dal wizard incluso nella piattaforma XPS, che contiene tutti gli strumenti di configurazione "a la carte" per ognuna delle opzioni disponibili.
Nella configurazione più semplice, così come è stata applicata al nostro robot, l'interfaccia tra il bus e le periferiche è il classico sistema "a registri": ogni periferica è vista come un banco di registri leggibili e/o scrivibili, attraverso i quali impartire comandi, trasferire dati in input/output e leggere attributi sullo stato della periferica.
I semplici cicli di lettura e scrittura da e per la periferica, all'interfaccia "IPIC-User Logic", sono riportati nella **Fig. 7**.
La gestione dei segnali di "acknowledge" dei dati in lettura e scrittura, IP2Bus_RdAck e IP2Bus_WrAck, è a discrezione della logica d'utente e può essere riservata a eventuali problemi di adattamento di velocità tra periferica e bus. Nel caso di trasferimenti sincroni con il segnale di clock del bus, Bus2IP_Clk, possono essere semplicemente mantenuti a livello alto e concludere ogni ciclo in due sole fasi di clock.

REALIZZAZIONE DELLA SCHEDA D'INTERCONNESSIONE TRA SCHEDA FPGA E SENSORI

E veniamo, adesso, alla realizzazione effettiva del sistema, il cui primo passo da compiere è stato quello di produrre il legante tra i moduli di controllo realizzati nella FPGA con i circuiti di condizionamento dei sensori.
Come illustrato dallo schema a blocchi della **Fig.8**, che rappresenta le funzionalità implementate dal circuito logico (visibile nella **Fig. 9**) la scheda di connessione, oltre a fungere da canale di comunicazione e adattatore di livelli elettrici, provvede a fornire le necessarie tensioni di alimentazione delle periferiche ed il collegamento con le batterie.
I motori sono connessi, invece, direttamente alla FPGA tramite un ponte di transistors prodotto da Digilent che provvede anche alla consegna dei segnali di feedback degli encoders, necessari al controllo di velocità e di posizione del robot.

UN PRIMO ESEMPIO: I SENSORI DI LUMINOSITÀ PER L'INSEGUIMENTO DELLA TRACCIA

Come si può vedere nella **Fig.10**, è stata progettata un'interfaccia che rappresenta un vero e proprio sottosistema autonomo, che scarica il software da tutti i compiti di misura della frequenza, di scansione dei sin-

goli sensori di luminosità e di colore, ed evita la necessità di dover impostare i normali timer e/o contatori disponibili nei microcontrollori standard.
Una volta scelta la modalità di scansione dei sensori di luminosità e di colore e avviata la sequenza di controllo, l'unico compito che il software deve svolgere è di leggere i registri con le misure correnti e, eventualmente, provvedere alla richiesta taratura della sensibilità e del range di frequenza dei trasduttori di luminosità.
Nella scelta dei sensori di luminosità abbiamo cercato di individuare quelli più semplici da interfacciare e quelli più vicini ai programmi didattici svolti. In particolare, ci siamo orientati verso i componenti della TAOS, che rispondevano meglio alle nostre aspettative e che sono risultati effettivamente molto facili da connettere al nostro "System On Chip" e molto affidabili nel loro impiego. Nelle **Fig. 11** e **Fig. 12**, sono illustrati gli schemi a blocchi di principio dei due componenti scelti: il TSL230-LF, per le misure monocromatiche e il TCS3200 per quella delle tre componenti di colore RGB; i due chip sono prodotti dalla TAOS.

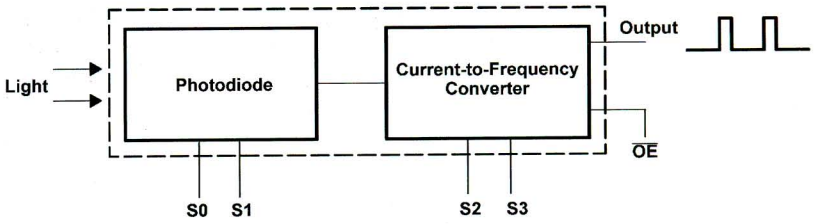


Fig. 11
Schema a blocchi interno del convertitore luce/frequenza TSL230-LF utilizzato nel robot per rilevare la luce monocromatica.

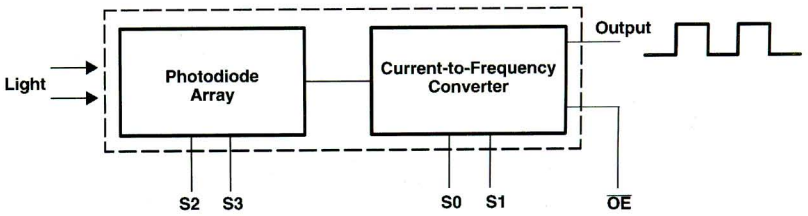


Fig. 12
Schema a blocchi interno del convertitore luce/frequenza TCS3200 usato per la misura delle tre componenti di colore rossa, verde e blu.

I PROSSIMI ARTICOLI

I prossimi due articoli, prendendo ad esempio proprio il sensore di luminosità qui presentato, offriranno lo spunto per illustrare in dettaglio, l'uno, come realizzare in un modulo VHDL la "user logic" da interfacciare al bus PLB per le misure di luminosità e colore, l'altro, le strutture dati nel linguaggio "C" e i relativi driver software necessari all'astrazione dell'hardware prodotto, in modo da rendere autonoma, e quanto più stabile è possibile, la scrittura delle applicazioni. In sostanza si cercherà di esemplificare uno dei punti centrali nel progetto di sistemi embedded, ovvero la rilevanza del rigore nella definizione delle interfacce tra hardware e software, in modo che il primo sia trasparente al secondo e che quest'ultimo sia indifferente ai cambiamenti dell'altro. □