

CORSO DI SISTEMI EMBEDDED



Implementazione di un computer su FPGA Altera ad hoc per il controllo di una struttura esapode

Progetto a cura di:

Alessandro Paghi

Luca Di Domenico

Indice generale

1 Premessa	1
2 Componentistica utilizzata	2
2.1 Altera DE10-Lite	2
2.2 Micro Servo Hitec HS-53	4
2.3 Esapode Augusto	5
3 Hexapod Computer	7
3.1 Clock Source	8
3.2 NIOS II Processor.....	10
3.3 On-Chip Memory (RAM)	11
3.4 System ID Peripheral	12
3.5 JTAG UART	13
3.6 Interval Timer	14
3.7 PIO (Parallel I/O)	15
3.8 Motor_DriverV2	16
4 Motor_DriverV2	17
4.1 Schema a blocchi generale	17
4.2 Blocco Enable	19
4.3 Blocco Angle To Time Allocator	20
4.4 Blocco First Allocator	22
4.5 Blocco Second Allocator	23
4.6 Blocco Motor Timer.....	24
4.7 Analysis & Synthesis	25
4.8 Fitter	26
4.9 Simulazione timing.....	27

5 Software	28
5.1 Porting del Codice	28
5.2 Struttura del Codice	29
5.2.1 Main	29
5.2.2 HEXAPOD_FUNCTION	29
5.2.3 ACCESSORIES_FUNCTION	30
6 Conclusioni	31
Bibliografia	31

Capitolo 1: Premessa

L'obiettivo del progetto risiede nella progettazione e nell'implementazione di un sistema embedded ad hoc per il controllo di una struttura esapode.

A tale scopo viene assemblato, mediante l'ausilio del tool Qsys della suite Quartus Prime 16.1, un computer specifico per l'applicazione d'interesse sul quale è integrato un componente dedicato al controllo dei 18 servomotori montati sulla struttura [2].

Successivamente è stato effettuato il porting di parte del codice sorgente presente all'interno del micro controllore Microchip PIC18F4550 [2].

La struttura meccanica non subisce mutamenti, in quanto progettata in modo da poter rendere il dispositivo in grado di svolgere una vasta gamma di movimenti.

Capitolo 2: Componentistica utilizzata

Si descrivono nel seguito i componenti utilizzati per il raggiungimento dell'obiettivo:

- TerasIC DE10-Lite Board;
- Micro servo Hitec HS-53;
- Esapode Augusto.

2.1 TerasIC DE10-Lite Board

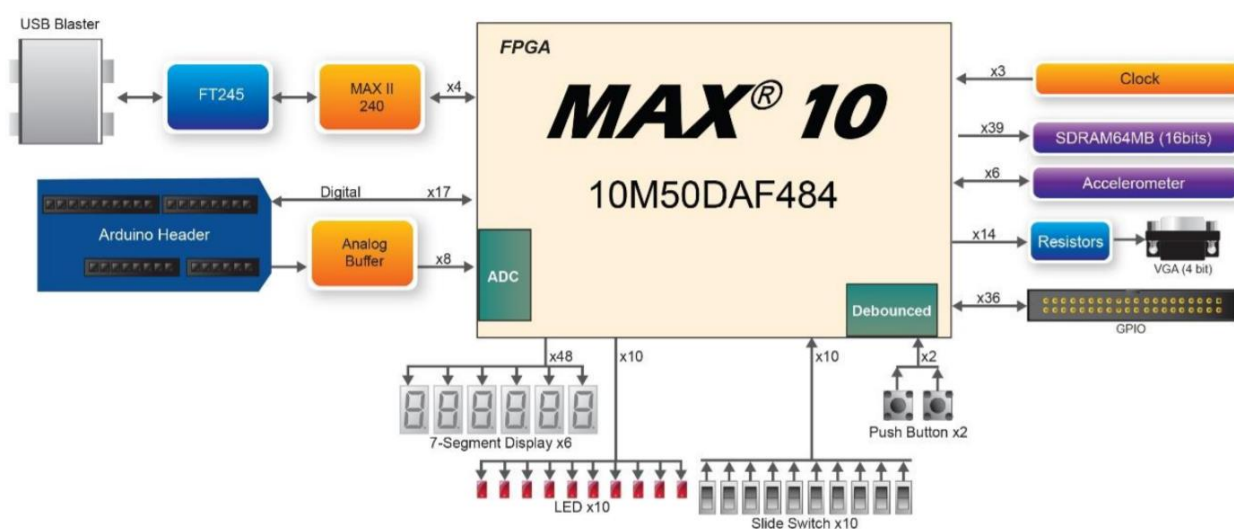


Figura: Diagramma a blocchi della board

La DE10-Lite presenta una piattaforma hardware estremamente robusta costruita attorno ad Altera MAX 10 FPGA.

FPGA MAX 10 è equipaggiato in modo da fornire il miglior rapporto qualità prezzo nello sviluppo di data path application e industry-leading programmable solutions al fine di ottenere estrema flessibilità di progetto.

Con FPGA MAX 10 è possibile ottenere applicazioni a basso costo e consumo con elevate performance.

La DE10-Lite è la soluzione perfetta per progettare applicazioni di valutazione e per il prototyping di tutte le potenzialità del FPGA Altera MAX 10.

Il dispositivo FPGA contiene:

- MAX 10 10M50DAF484C7G Device;
- Due ADC integrati, ogni ADC supporta 1 input analogico dedicato ed 8 dual function pins;
- 50K elementi logici programmabili;
- 1,638 Kbits M9K memory;
- 5,888 Kbits user flash memory;
- 144 18×18 moltiplicatori;
- 4 PLLs.

Programmazione e configurazione:

- On-Board USB Blaster (Normal type B USB connector).

Memory Device:

- 64MB SDRAM, x16 bits data bus.

Connettori:

- 2x20 GPIO Header;
- Arduino Uno R3 Connector, including six ADC channels.

Display:

- 4-bit resistor-network DAC for VGA.

Interruttori, bottoni e LED:

- 10 LEDs;
- 10 Slide Switches;
- 2 Push Buttons with Debounced;
- Six 7-Segments.

Power:

- 5V DC input da USB o connettore di potenza esterno.

2.2 Micro servo Hitec HS-53

Il servomotore si presenta come un piccolo contenitore di materiale plastico da cui fuoriesce un perno in grado di ruotare di un angolo compreso fra 0° e 180° , la cui posizione rimane stabile.

Per ottenere la rotazione del perno è utilizzato un motore in DC e un meccanismo di demoltiplica che consente di aumentare la coppia in fase di rotazione.

La rotazione del motore è effettuata tramite un circuito di controllo interno in grado di rilevare l'angolo di rotazione raggiunto dal perno tramite un potenziometro resistivo e di bloccare, quindi, il motore sul punto desiderato.

Il modello utilizzato viene alimentato con batteria a 6 V.



Figura: Modello CAD del servo motore Hitec HS-53

Caratteristiche tecniche:

Alimentazione	4.8 - 6.0 V
Coppia a 6 V	1.5 Kg/cm
Velocità a 6 V	0.13 sec/60 deg
Ingranaggi	Nylon
Dimensioni	22.8 x 11.6 x 22.6 mm
Peso	8 gr

2.3 Esapode Augusto

La struttura meccanica si presenta come un oggetto estremamente robusto, esteticamente accattivante e di facile montaggio.

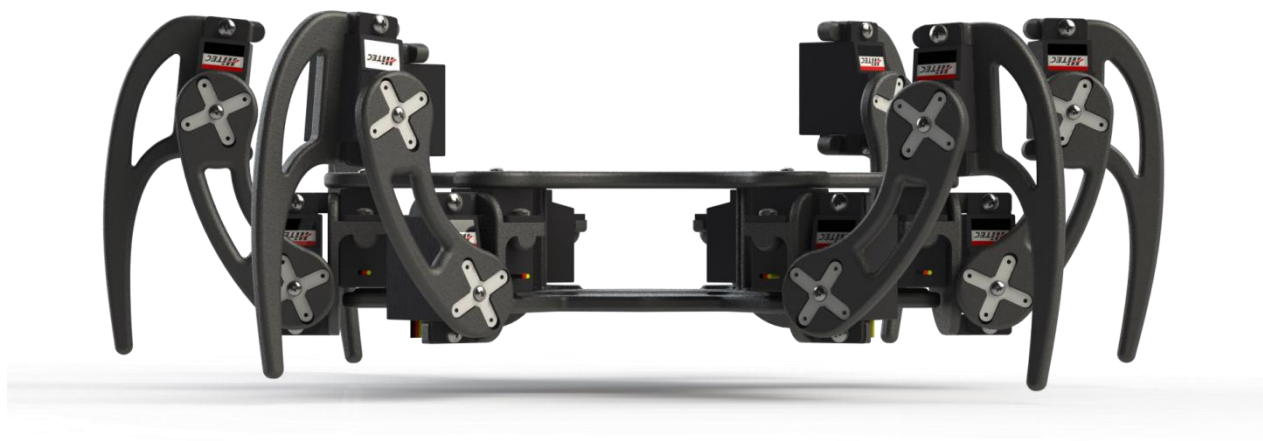


Figura: Modello CAD dell'assieme, vista frontale

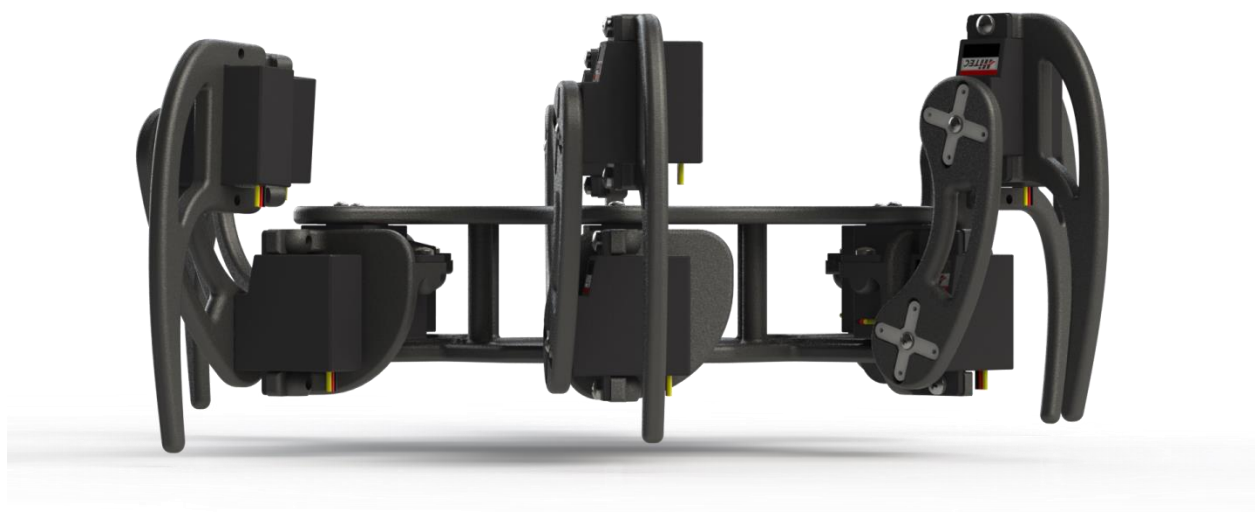


Figura: Modello CAD dell'assieme, vista laterale dx



Figura: Modello CAD dell'assieme, vista superiore



Figura: Modello CAD dell'assieme, vista in prospettiva

Capitolo 3: Hexapod Computer

Il computer è implementato mediante l'ausilio del tool Qsys (System Integration Tool) della suite Quartus Prime 16.1.

Il software permette l'assemblaggio dell'architettura desiderata attraverso l'utilizzo di componenti hardware presenti in libreria o appositamente descritti dall'utente.

Si riporta di seguito il diagramma a blocchi del sistema embedded target realizzato ad hoc per il controllo di una struttura esapode.

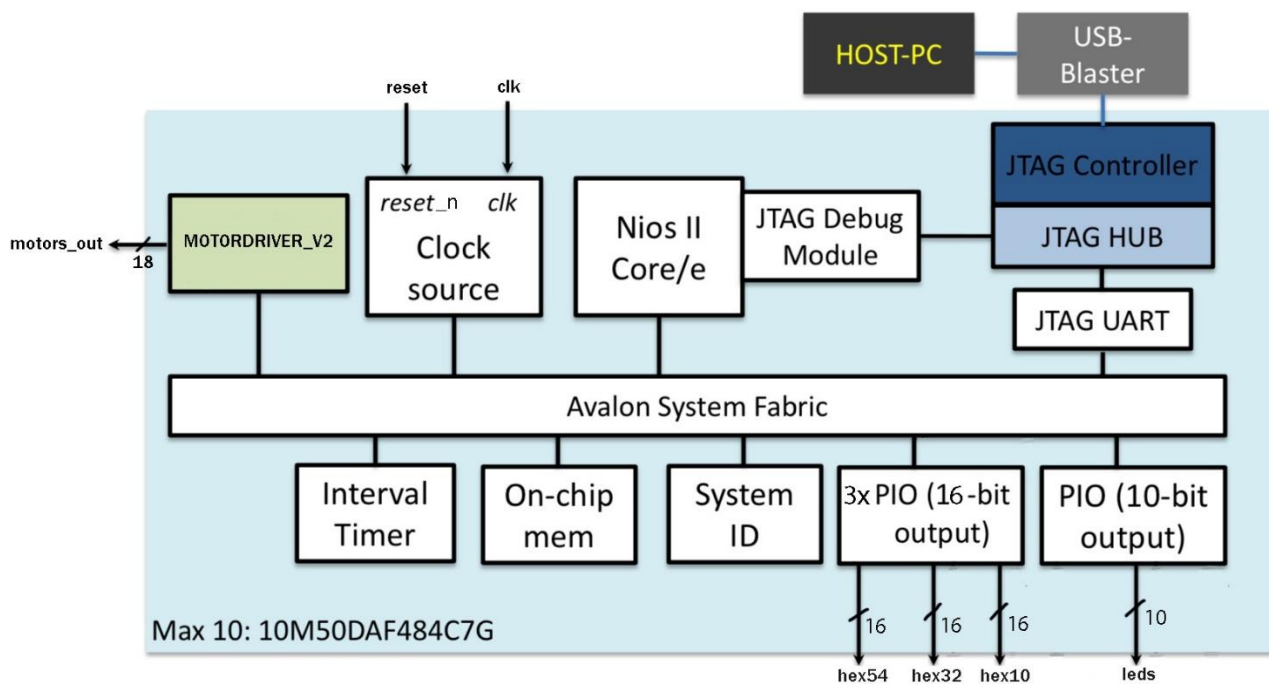


Figura: Diagramma a blocchi Hexapod Computer

Nei capitoli successivi si descrivono, nel dettaglio, i componenti presenti nel precedente schema e le interconnessioni tra di essi.

3.1 Clock Source

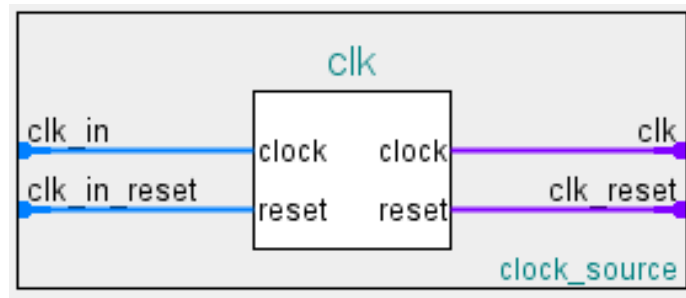


Figura: Componente Clock Source

Blocco Clock Source si occupa di:

- Predisporre un'interfaccia Avalon Conduit pilotata dalla sorgente di clock presente sulla TerasIC DE10-Lite Board. Tale segnale viene internamente fornito ad un'interfaccia Avalon Clock Source, permettendo al System Interconnect Fabric di instradare il clock ai componenti presenti sul computer;
- Predisporre un'interfaccia Avalon Conduit per l'acquisizione del segnale di reset da un push button presente sulla TerasIC DE10-Lite Board. Il rilascio di tale segnale viene opportunamente sincronizzato col clock citato precedentemente, mediante l'ausilio di una rete di Reset Synchronizer.

Si riporta in figura il circuito schematico inerente ai push buttons.

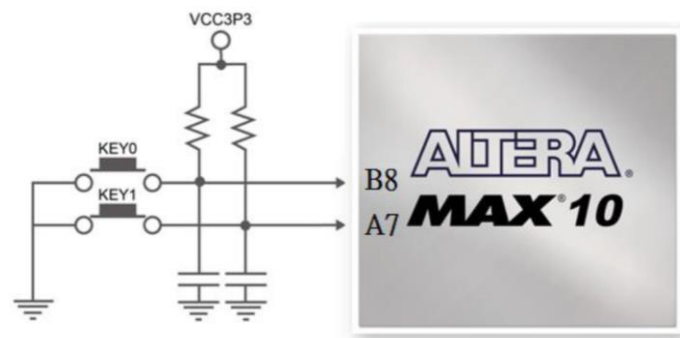


Figura: Circuito push buttons

Si nota immediatamente che la pressione del push button genera un segnale a livello logico 0. Il circuito di sincronizzazione del Reset utilizzerà quindi un segnale `rst_n`.

Reset Synchronizer

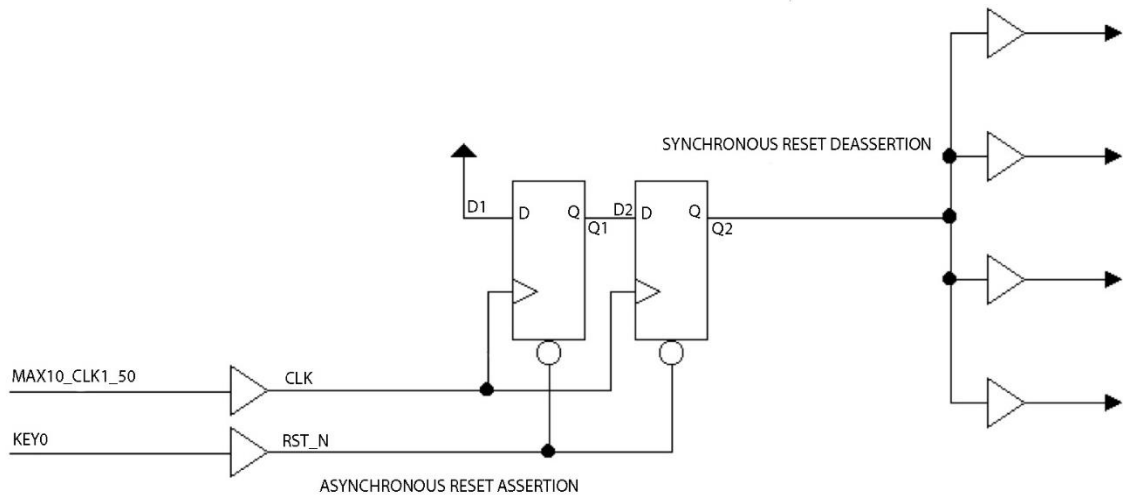


Figura: Circuito di sincronizzazione del segnale di Reset

A seguito della pressione del tasto di reset KEY0, si ha il reset dei Flip Flop e pertanto $Q1=Q2=0$. Al colpo di clock successivo al rilascio del tasto $Q1=1$ e $Q2=0$. Al successivo fronte, $Q1=Q2=1$ e il reset viene rimosso ai componenti presente sul computer.

3.2 NIOS II Processor

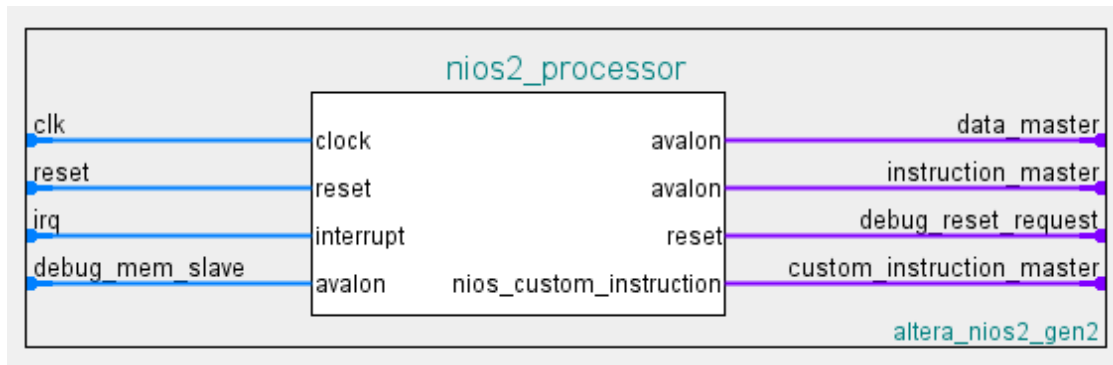


Figura: Componente NIOS II Processor

Il processore in questione presenta un'architettura RISC general purpose con istruzioni e registri implementati su 32 bit.

Blocco NIOS II Processor presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon Interrupt Receiver per l'acquisizione del segnale di Interrupt Request (irq) proveniente dai componenti Interval Timer e JTAG UART;
- Interfaccia Avalon MM Slave debug_mem_slave per la gestione dei break introdotti dall'utilizzo del Debug Module;
- Interfaccia Avalon MM Master Data per la gestione della lettura/scrittura dati address-based da e verso periferiche quali: On-Chip Memory, JTAG UART, Interval Timer, PIO, MotoDriver_V2 e System ID;
- Interfaccia Avalon MM Master Instructions per la gestione di lettura istruzioni address-based dalla On-Chip Memory;
- Interfaccia Avalon Reset Output per il pilotaggio di un segnale di reset, interno al computer, ai componenti presenti su di esso.

Per il computer in questione, il NIOS II Processor si presenta in versione Economy.

3.3 On-Chip Memory (RAM)

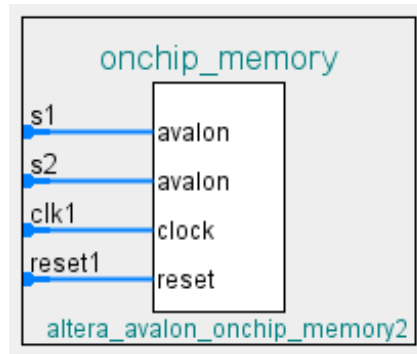


Figura: Componente On-Chip Memory (RAM)

Blocco On-Chip Memory è una memoria del tipo:

- RAM (Writable);
- Dual-port access, al fine di implementare un'architettura Harvard;
- Single clock operation;

La dimensione dei bus è fissata a 32 bit coerentemente con la dimensione dei registri interni al NIOS II Processor e alla dimensione delle word in memoria.

La total memory size è imposta a 174080 B in modo da contenere le funzioni di scrittura e lettura formattate (printf, scanf).

Il contenuto del blocco On-Chip Memory è inizializzato col file .hex contenente il codice steso sulla IDE Eclipse in linguaggio di programmazione C.

Tale operazione permette, al momento della programmazione del FPGA, di trasferire su di esso non solo la descrizione hardware ma anche il sorgente software.

Blocco On-Chip Memory presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon MM Slave s1 per la gestione della lettura/scrittura dati address-based da e verso il NIOS II Processor;
- Interfaccia Avalon MM Slave s2 per la gestione di lettura istruzioni address-based dalla On-Chip Memory verso il NIOS II Processor.

3.4 System ID Peripheral

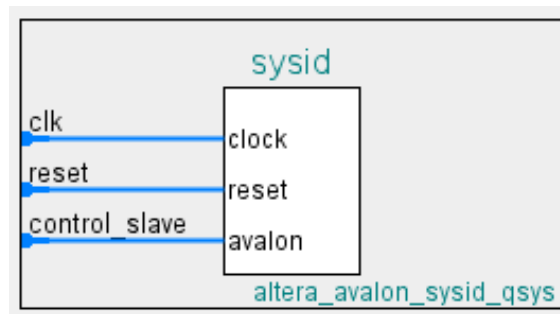


Figura: Componente System ID Peripheral

System ID Peripheral identifica univocamente, mediante un numero esadecimale a 32 bit, il NIOS II Processor.

Blocco System ID Peripheral presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon MM Slave per la gestione di lettura dati address-based da parte del NIOS II Processor.

3.5 JTAG UART

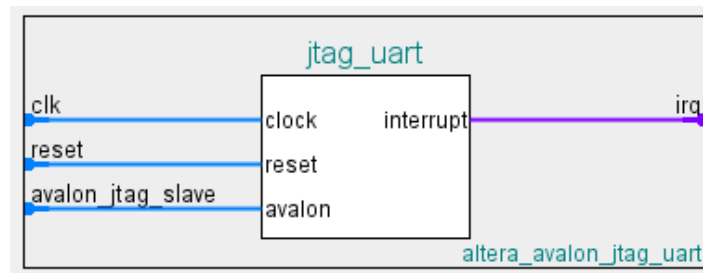


Figura: Componente JTAG UART

Il core JTAG UART implementa un metodo di comunicazione “Serial Character Streams” tra un PC Host e la piattaforma hardware implementata su FPGA Altera. La periferica master (in questo caso NIOS II Processor) comunica con il componente in questione leggendo e scrivendo registri di controllo e dati.

Il componente JTAG UART utilizza la circuiteria JTAG implementata su FPGA e fornisce ad un host l’accesso attraverso i JTAG Pins presenti.

Il software presente all’interno del NIOS II Processor può accedere alle funzionalità messe a disposizione dalla JTAG UART attraverso la HAL system library; è possibile, inoltre, accedere a tali funzioni attraverso l’ANSI C standard library stdio.h .

L’utente può accedere alla JTAG UART attraverso la nios2-terminal command line presente nella NIOS II IDE.

Blocco JTAG UART presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon MM Slave JTAG per la gestione della lettura/scrittura dati address-based da e verso il NIOS II Processor;
- Interfaccia Avalon Interrupt Sender per la generazione del segnale di Interrupt Request (irq) verso il NIOS II Processor.

3.6 Interval Timer

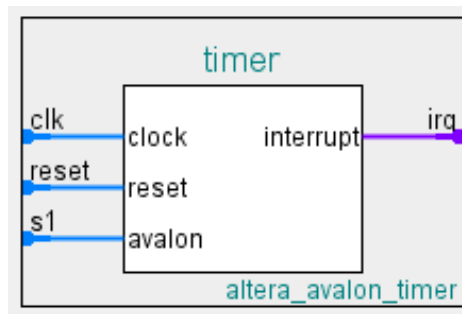


Figura: Componente Interval Timer

Il core Interval Timer è un contatore implementato a 32 bit. Attraverso la HAL system library è possibile utilizzare tale componente come System Clock Timer o Timestamp Timer.

I Timestamp Driver consentono di utilizzare l'Interval Timer come Timestamp attraverso le funzioni fornite dalla HAL: `alt_timestamp_start()`, per l'inizializzazione del conteggio, e `alt_timestamp()`, per la lettura del registro Snap attraverso l'interfaccia Avalon MM Slave.

Blocco Interval Timer presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon MM Slave per la gestione della lettura/scrittura dati address-based da e verso il NIOS II Processor;
- Interfaccia Avalon Interrupt Sender per la generazione del segnale di Interrupt Request (`irq`) verso il NIOS II Processor.

3.7 PIO (Parallel I/O)

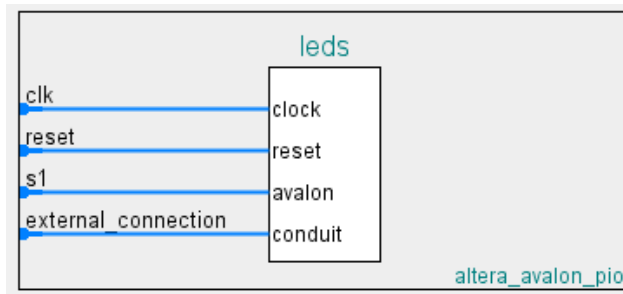


Figura: Componente PIO

Il core Parallel I/O con interfaccia Avalon fornisce un'interfaccia Memory Mapped tra un Avalon MM Slave port e delle general purpose I/O ports.

Nel presente progetto sono state istanziate 4 periferiche PIO (direction: output), rispettivamente per:

- Controllo dei display 7-segmenti HEX0 ed HEX1;
- Controllo dei display 7-segmenti HEX2 ed HEX3;
- Controllo dei display 7-segmenti HEX4 ed HEX5;
- Controllo dei leds LEDR0-LEDR9.

Blocco PIO presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon MM Slave per la gestione della lettura/scrittura dati address-based da e verso il NIOS II Processor;
- Interfaccia Avalon Conduit per la connessione alle general purpose I/O ports.

3.8 MotorDriver_V2

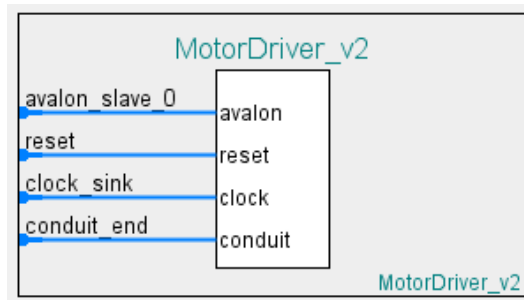


Figura: Componente MotorDriver_V2

Il componente MotorDriver_V2 si interfaccia con il NIOS II Processor attraverso la Avalon MM Slave. Tale interfaccia presenta esclusivamente il segnale di abilitazione di scrittura WRITE e il bus dati WRITE_DATA[31:0].

Tale bus è strutturato in modo da contenere sui bit WRITE_DATA[4:0] l'ID del motore associato all'angolo espresso sui bit WRITE_DATA[12:5].

Il componente si occupa di tradurre gli angoli provenienti dal NIOS II Processor in corrispondenti segnali PWM atti al pilotaggio dei servo motori identificati dal rispettivo ID.

Blocco Interval Timer presenta le seguenti interfacce:

- Interfaccia Avalon Clock Input per la ricezione del segnale di clock dal Clock Source;
- Interfaccia Avalon Reset input per la ricezione del segnale di reset dal Clock Source;
- Interfaccia Avalon MM Slave per la gestione della scrittura dati address-based dal NIOS II Processor;
- Interfaccia Avalon Conduit per la connessione ai PIN signal dei 18 servo motori montati sulla struttura.

Capitolo 4: MotorDriver_V2

Si descrivono nel seguito i blocchi progettati e le tecniche utilizzate per il design del componente hardware MotorDriver_V2.

4.1 Schema a blocchi generale

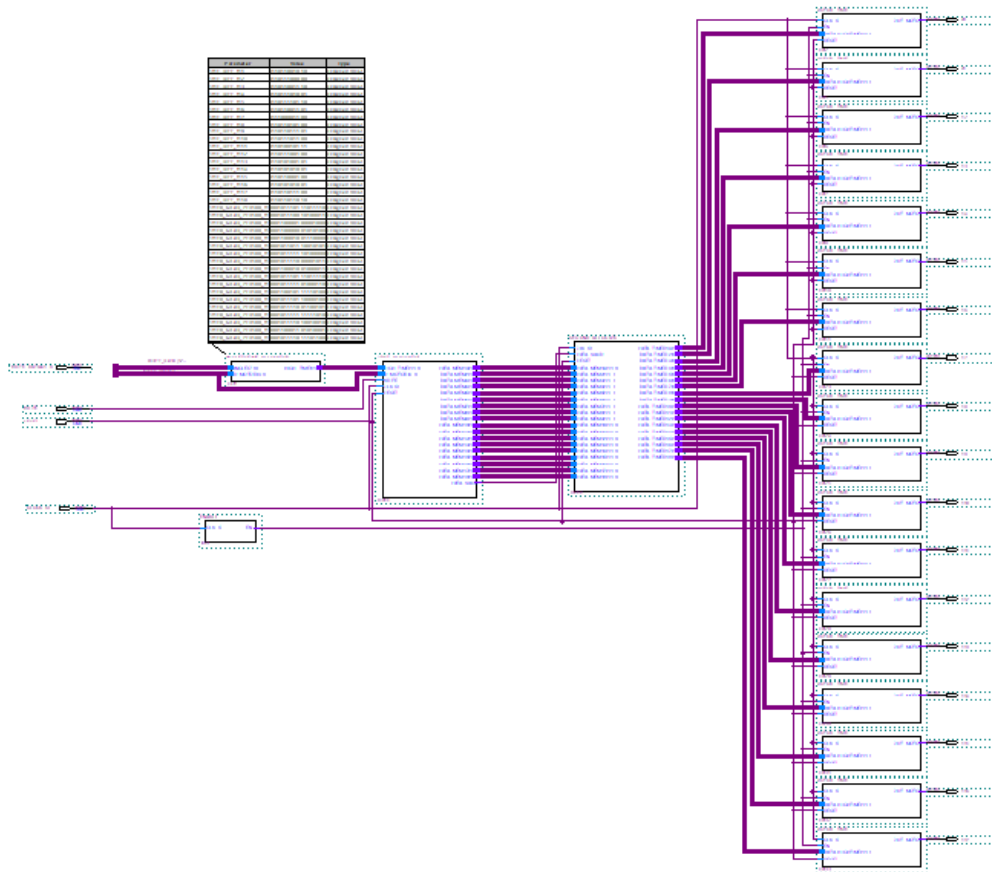


Figura: Schema a blocchi dell'architettura implementata su Quartus Prime 16.1

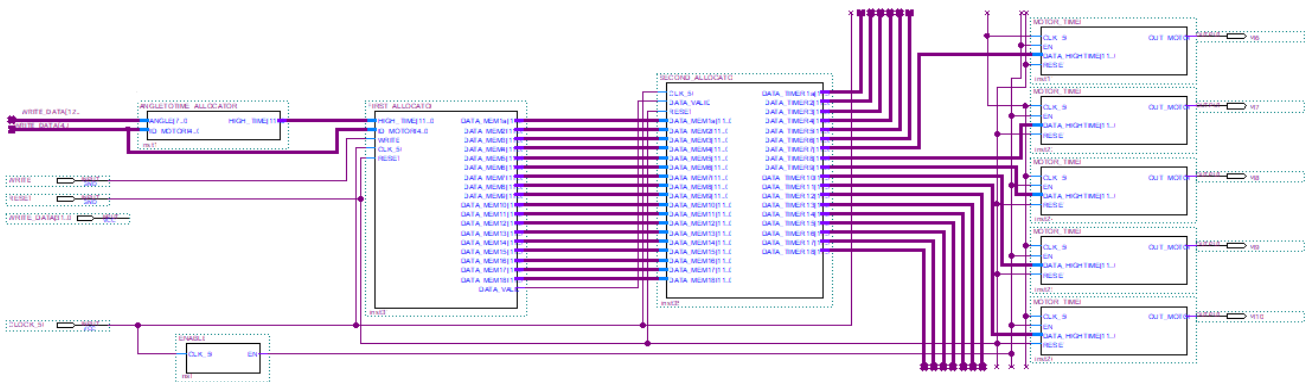


Figura: Dettagli schema a blocchi dell'architettura implementata su Quartus Prime 16.1

Lo schema a blocchi comprende l'utilizzo dei seguenti componenti:

- Blocco Enable;
- Blocco Angle To Time Allocator;
- Blocco First Allocator;
- Blocco Second Allocator;
- 18 x Blocco Motor Timer;

L'intero progetto è realizzato nel pieno rispetto delle regole del progetto sincrono statico: tutti i blocchi costruiti sono clockati dallo stesso fronte dello stesso segnale di clock.

Dal capitolo *Specifiche di risoluzione* [2] è risultato che il periodo di clock necessario per poter generare segnali di controllo PWM in maniera corretta è di $T_{EN} = 1 \text{ us}$, che corrisponde ad una frequenza di clock pari a $f_{EN} = 1 \text{ MHz}$.

È necessaria la presenza di un'unità di abilitazione che gestisca l'attivazione della macchina sequenziale Motor Timer che lavora a frequenza $f_{EN} < f_{CLK}$.

L'attivatore si occupa di abilitare il funzionamento della macchina sequenziale sincrona operante a frequenza f_{EN} , una volta ogni 50 periodi di f_{CLK} .

4.2 Blocco Enable

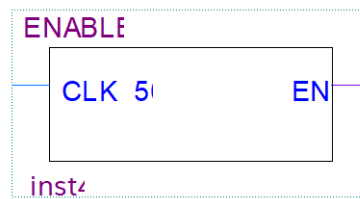


Figura: Blocco Enable

Blocco Enable si occupa di generare il segnale di attivazione delle macchine sequenziali sincrone Motor Timer operanti a frequenza $f_{EN} = 1 \text{ MHz}$., clockate con il segnale di riferimento f_{CLK} .

4.3 Blocco Angle To Time Allocator

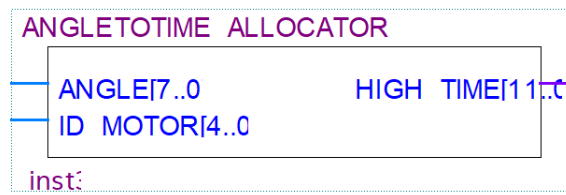


Figura: Blocco Angle To Time Allocator

Blocco Angle To Time Allocator traduce l'angolo rivelato sui bit WRITE_DATA[12:5] in un equivalente intervallo temporale a livello logico 1 T_{ON} necessario al fine di posizionare il servo motore identificato dai bit WRITE_DATA[4:0].

Data l'unicità di ogni servo motore, l'operazione di conversione utilizza una serie di parametri che definiscono le caratteristiche di ogni motore.

[2] Consideriamo legame di linearità tra tempo che l'impulso generato trascorre a livello alto e angolo raggiunto dal servo motore; l'operazione da compiere al fine di effettuare la traduzione è la seguente:

$$(1) T_{ON} = \frac{Time_{180} - Time_0}{180} * Angle + Time_0$$

Dove:

- $Time_0$: durata dell'impulso per il posizionamento a 0° ;
- $Time_{180}$: durata dell'impulso per il posizionamento a 180° ;
- Angle: angolo espresso in gradi ($^\circ$);
- T_{ON} : durata dell'impulso generato per il posizionamento ad Angle.

La traduzione prevede l'utilizzo di operazioni di somma, moltiplicazione e divisione, con conseguente inserimento, a livello di sintesi, di blocchi sommatore, moltiplicatore e divisore.

Di seguito si descrive un metodo al fine di aggirare la necessità della presenza di un blocco divisore.

L'operazione di divisione che si compie è fatta dividendo per una costante.

Al posto di eseguire (1) si elaborano in sequenza queste tre operazioni:

1. $\text{Time}_{APP} = T_{ON} * 180 = (\text{Time}_{180} - \text{Time}_0) * \text{Angle} + \text{Time}_0 * 180;$
2. $\text{Appoggio}_1 = \text{Time}_{APP} * (0.00555\dots);$
3. $T_{ON} = \text{Appoggio}_1[25:14].$

Time_{APP} è un registro di appoggio di dimensione 20 bit ed Appoggio_1 è un registro di appoggio di dimensione 27 bit.

L'operazione 2 consente di approssimare la divisione per 180 con la moltiplicazione per $1/180$.

Il troncamento di $1/180$ prevede l'inserimento di un errore.

$$[1/180]_2 = 0.000\ 000\ 010\ 110\ 110\ 000\ 01\dots$$

$$[1/180]_2 \sim 0.000\ 000\ 010\ 110\ 11$$

Utilizzando 7 cifre significative, dopo la prima diversa da zero, si commette un errore relativo minore di 2^{-12} nel rappresentare $1/180$ dato che i successivi 5 bit da considerare sarebbero stati nulli.

Infine si prelevano dal risultato ottenuto i 12 bit di interesse.

Facendo uso di due nuovi parametri $\text{Time_Diff} = \text{Time}_{180} - \text{Time}_0$ e $\text{Time_Mult} = \text{Time}_0 * 180$, al fine di semplificare le operazioni da svolgere, la frequenza massima di funzionamento del blocco `MotorDriver_V2` incrementa a 57.57 MHz, rispettando largamente i vincoli di timing imposti dal progetto.

4.4 Blocco First Allocator

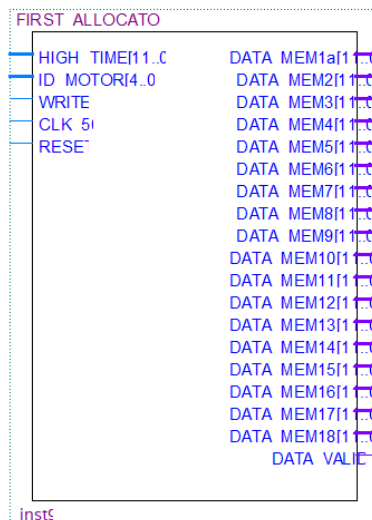


Figura: Blocco First Allocator

Blocco First Allocator memorizza sequenzialmente le traduzioni effettuate dal blocco Angle To Time Allocator .

Viene riempito un buffer di memorizzazione composto da 18 locazioni da 12 bit. L'abilitazione di un flag DATA_VALID evidenzia al blocco successivo il completamento del buffer.

L'attivazione del segnale di Reset permette l'azzeramento dei 18 registri interni.

4.5 Blocco Second Allocator

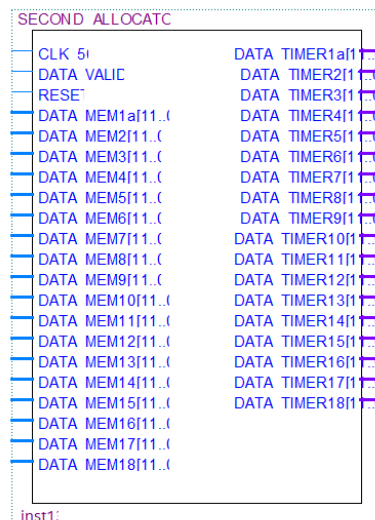


Figura: Blocco Second Allocator

Blocco Second Allocator si occupa di memorizzare parallelamente i dati provenienti dal blocco First Allocator una volta completato il riempimento del primo buffer, segnalato dal livello logico alto del segnale DATA_VALID.

Data l'equivalenza architetturale, anche il secondo buffer di memoria è composto da 18 locazioni a 12 bit.

L'architettura First-Second Allocator consente l'allocazione sequenziale dei T_{ON} provenienti dalla logica combinatoria Angle To Time Allocator e l'aggiornamento parallelo dei valori di conteggio dei 18 Motor Timer. Per un robot esapode è infatti opportuno che la posizione di tutti i giunti sia aggiornata contemporaneamente, come è evidente nel caso di rototraslazioni del corpo.

Anche in questo caso l'attivazione del segnale di Reset permette l'azzeramento dei 18 registri interni.

4.6 Blocco Motor Timer

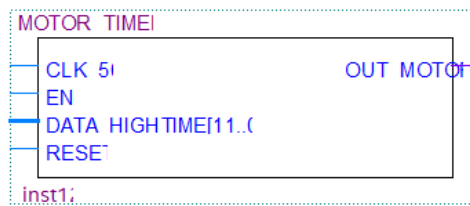


Figura: Blocco Motor Timer

Blocco Motor Timer è una macchina sequenziale sincrona abilitata attraverso il segnale EN. Essa genera il segnale di controllo PWM di un servo motore.

L'attivazione del segnale di Reset permette l'azzeramento del segnale PWM e del corrispondente registro di conteggio.

L'aggiornamento del registro di conteggio interno al blocco avviene ad una frequenza pari a 50 Hz, frequenza di lavoro del servo motore, coerentemente con il periodo del segnale PWM ($T=20$ ms).

4.7 Analysis & Synthesis

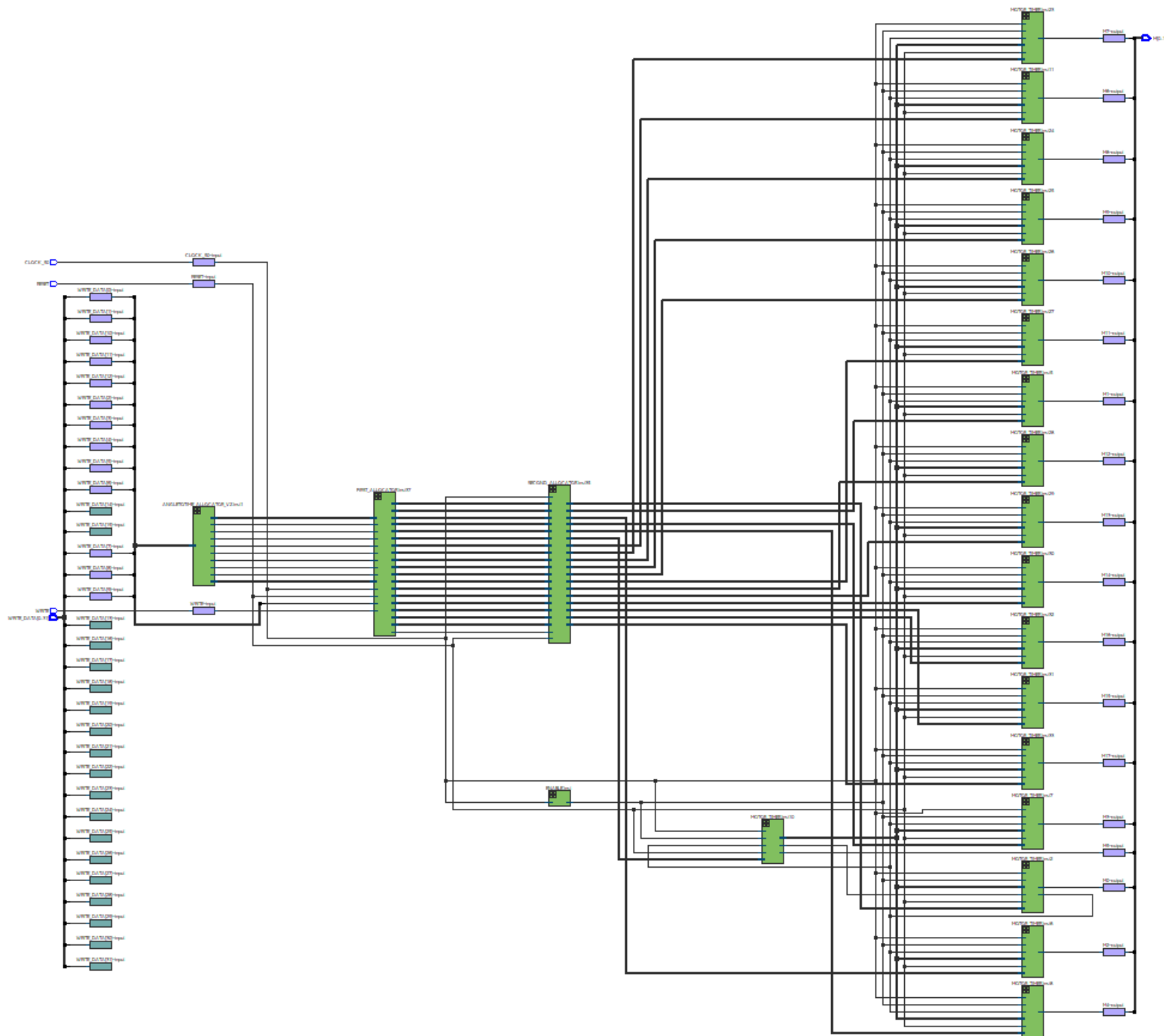


Figura: Technology Map Viewer post sintesi


Analysis & Synthesis Summary	
 <<Filter>>	
Analysis & Synthesis Status	Successful - Thu May 31 17:09:26 2018
Quartus Prime Version	16.1.2 Build 203 01/18/2017 SJ Lite Edition
Revision Name	AUGUSTO_FPGA_PROJECT
Top-level Entity Name	AUGUSTO_FPGA_PROJECT
Family	MAX 10
Total logic elements	1,018
Total combinational functions	378
Dedicated logic registers	689
Total registers	689
Total pins	53
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	5
Total PLLs	0
UFM blocks	0
ADC blocks	0

Figura: Risultati Analysis & Synthesis

4.8 Fitter


Fitter Summary	
 <<Filter>>	
Fitter Status	Successful - Thu May 31 17:09:47 2018
Quartus Prime Version	16.1.2 Build 203 01/18/2017 SJ Lite Edition
Revision Name	AUGUSTO_FPGA_PROJECT
Top-level Entity Name	AUGUSTO_FPGA_PROJECT
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	801 / 49,760 (2 %)
Total combinational functions	379 / 49,760 (< 1 %)
Dedicated logic registers	689 / 49,760 (1 %)
Total registers	689
Total pins	53 / 360 (15 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	5 / 288 (2 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Figura: Risultati Fitter

4.9 Simulazione Timing

Si riportano nel seguito i risultati ottenuti dalle simulazioni timing effettuate sull'architettura implementata.

Clocks						
<<Filter>>						
	Clock Name	Type	Period	Frequency	Rise	Fall
1	CLOCK_50	Base	20.000	50.0 MHz	0.000	10.000

Figura: Segnale di clock

La simulazione fa riferimento al caso in cui:

- $V_{cc} = 1.2V$;
- $Temp = 85^{\circ}C$;

Slow 1200mV 85C Model Fmax Summary			
<<Filter>>			
	Fmax	Restricted Fmax	Clock Name
1	57.57 MHz	57.57 MHz	CLOCK_50

Figura: Risultati sulla frequenza massima di funzionamento

Slow 1200mV 85C Model Setup Summary			
<<Filter>>			
	Clock	Slack	End Point TNS
1	CLOCK_50	2.631	0.000

Figura: Risultati dello slack sui tempi di setup [ns]

Slow 1200mV 85C Model Hold Summary			
<<Filter>>			
	Clock	Slack	End Point TNS
1	CLOCK_50	0.412	0.000

Figura: Risultati dello slack sui tempi di hold [us]

Capitolo 5: Software

Si descrivono nel seguito le scelte progettuali svolte al fine di stendere il codice sorgente dell'applicativo in questione.

5.1 Porting del Codice

Una prima versione del codice è descritta sul micro controllore Microchip PIC18F4550 utilizzato nel progetto [2]. Tale unità di calcolo presenta un'architettura con registri ad 8 bit mentre il soft core NIOS II Processor, implementa un'architettura RISC con registri a 32 bit.

Il porting del codice passa attraverso l'utilizzo della libreria "alt_types.h" che definisce le tipologie integer ad hoc per il microprocessore NIOS II.

Il progetto descritto nell'articolo [2] implementa un'architettura in cui il micro controllore Microchip PIC18F4550 si comporta da master verso un Motor Driver implementato ad hoc su Altera FPGA, il quale ricopre il ruolo di slave. Essendo le due entità separate, è necessaria l'implementazione di un protocollo di comunicazione, tra quelli supportati dal controllore utilizzato. In particolare, la scelta del protocollo UART, comporta la costruzione di una macchina a stati sul Motor Driver necessaria al fine di campionare la linea di comunicazione ed interpretare i dati rivelati.

Nel sistema embedded progettato, invece, Motor Driver e soft core NIOS II Processor si trovano entrambi all'interno dello stesso package contenente l'FPGA MAX10. La macchina sequenziale atta al campionamento della linea di comunicazione presente tra master e slave viene rimossa e sostituita con un'interfaccia Avalon Memory Mapped Slave per la gestione della scrittura dati address-based dal NIOS II Processor.

Il componente MotorDriver_V2 si presenta quindi al processore, tramite un registro a 32 bit write only per la scrittura simultanea di ID e angolo associato.

Il software si prende carico della scrittura di tale registro con i risultati provenienti dalle funzioni implementanti il movimento della struttura esapode.

5.2 Struttura del Codice

Il programma è organizzato in 5 file sorgente:

- main.c;
- HEXAPOD_FUNCTION.h e HEXAPOD_FUNCTION.c;
- ACCESSORIES_FUNCTION.h e ACCESSORIES_FUNCTION.c .

5.2.1 Main

Il programma main si occupa dell'inizializzazione dei valori delle PIO quali Leds, HEX10, HEX32 ed HEX54, e dell'inizializzazione della posizione dei giunti della struttura esapode.

Successivamente, mediante un loop infinito, rimane in attesa delle istruzioni da compiere immesse dall'utente attraverso il nios2-terminal fornito dalla IDE Eclipse.

5.2.2 HEXAPOD_FUNCTION

Il programma HEXAPOD_FUNCTION contiene le funzioni che consentono il movimento del robot esapode. In particolare, la funzione:

```
void IKbody ( float grx, float gry, float grz, float tx, float ty, float tz );
```

consente la rotazione del corpo di grx gradi rispetto all'asse x, gry gradi rispetto all'asse y, grz gradi rispetto all'asse z e la traslazione di tx cm lungo l'asse x, ty cm lungo l'asse y e tz cm lungo l'asse z, nel sistema di riferimento Body [1].

La funzione:

```
void IKLeg( alt_u8 id_leg, float x, float y, float z );
```

consente all'arto id_leg, il raggiungimento del punto di appoggio x cm, y cm e z cm nel sistema di riferimento Leg [1].

La funzione:

```
void sendToMotorDriver( alt_u8 id, alt_u8 degree );
```


consente l'assegnamento dell'angolo degree, calcolato dalle funzioni precedentemente elencate, al servo motore id.

Qualora la posizione desiderata per il robot esapode non fosse raggiungibile, la funzione `sendToMotorDriver` preserva la posizione precedente alla richiesta.

5.2.3 ACCESSORIES_FUNCTION

Il programma `ACCESSORIES_FUNCTION` contiene le funzioni che consentono l'utilizzo delle PIO quali `LEDS`, `HEX10`, `HEX32` ed `HEX54`.

In particolare, la funzione:

```
void Play7Segments( alt_8 first, alt_8 second, alt_8 third );
```

consente la rappresentazione, sulle tre coppie di display a disposizione sulla board, degli angoli di rotazione e dei vettori di traslazione riferiti ai tre assi di riferimento.

La funzione:

```
void PlayLeds( alt_u8 id_gamba, alt_u8 clr );
```

consente l'accensione del led corrispondente alla `id_gamba` in movimento.

Capitolo 6: Conclusioni

Il risultato è un'architettura dedicata estremamente performante che elimina la necessità di un protocollo di comunicazione UART tra unità di calcolo e Motor Driver.

Il dispositivo consente la progettazione di algoritmi di movimento più o meno complessi, data l'elevata ottimizzazione del sistema embedded implementato.

Nonostante il design ad hoc, il numero di posizioni che l'esapode può variare in un'unità di tempo pari ad 1s è determinata dal segnale di controllo dei servo motori, operante a frequenza 50 Hz, sebbene l'architettura progettata permetta un throughput decisamente maggiore. Per tale motivo, infatti, è stato ritenuto opportuno l'utilizzo del NIOS II Processor in versione economy anziché fast, evitando l'utilizzo della tecnica di pipeling e l'implementazione del meccanismo di cache.

Eventuali sviluppi futuri potrebbero portare alla creazione di algoritmi per la camminata del robot esapode, eventualmente basati su Nios II Processor in versione fast, facilmente inseribile all'interno del computer, data l'implementazione dell'architettura di memorizzazione Harvard.

Bibliografia

- [1] Alessandro Paghi, "Progettazione e realizzazione della struttura meccanica e del sistema di controllo di un robot esapode", University of Siena, July 2016.
- [2] Alessandro Paghi, Lorenzo De Marinis "Implementazione di hardware su FPGA Altera dedicato al controllo di servomotori in una struttura esapode", University of Pisa, February 2018.
- [3] TerasIC, "DE10-Lite User Manual", 2017.