Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea in Elettronica

Tesi di laurea

Soft-Processor IP per FPGA

Tipologie a confronto

Laureando:
Marioli Michael Simone
Matricola 575203

Relatore:
Vogrig Daniele

Anno Accademico 2009-2010

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

L'obiettivo di questo lavoro di tesi è quello di realizzare un confronto tra i diversi soft-processors disponibili attualmente in commercio.
Nel capitolo 1 vengono forniti alcuni concetti fondamentali sulle FPGA e sui moduli IP indispensabili per una corretta comprensione del resto del lavoro.
Nel capitolo 2 ci si sofferma sulle due diverse modalità in cui è possibile realizzare un processore su di una FPGA indicandone vantaggi e svantaggi.
Nel capitolo 3, il quale costituisce il core di tale lavoro, vengono invece descritti, uno ad uno, i principali soft-processors a disposizione nel mondo dell'elettronica, soffermandosi particolarmente su *MicroBLAZE* e su *Nios II*, che rappresentano i soft-processors maggiormente utilizzati per la realizzazione di sistemi embedded.

CAPITOLO 1

FPGA e moduli IP

Il presente capitolo ha lo scopo di introdurre una serie di definizioni e concetti utili alla comprensione del proseguo di questo lavoro di tesi.

Viene fornita una panoramica sulla FPGA e sulle sue caratteristiche principali, facendo particolare attenzione alla trattazione dei moduli IP.

1.1 FPGA

1.1.1 Introduzione

I dispositivi FPGA (Field Programmable Gate Array) sono dispositivi digitali la cui funzionalità è programmabile via software [28]. Non si tratta della classica programmabilità di un processore ma della possibilità di riconfigurare le risorse logiche disponibili per implementare un progetto complesso a livello di singole interconnessioni tra porte logiche.

Sono elementi che presentano caratteristiche intermedie rispetto ai dispositivi ASIC (Application Specific Integrated Circuit) da un lato e a quelli con architettura PAL (Programmable Array Logic) dall'altro. L'uso di tali componenti comporta alcuni vantaggi rispetto agli ASIC: si tratta infatti di dispositivi standard la cui funzionalità da implementare non viene impostata dal produttore che quindi può produrre su larga scala a basso prezzo. La loro genericità li rende adatti a un gran numero di applicazioni come elettronica di consumo, comunicazioni, automotive, aerospaziale, militare e biomedicale.
Essi sono programmati direttamente dall'utente finale, consentendo la diminuzione dei tempi di progettazione, di verifica mediante simulazioni e di prova sul campo dell'applicazione.

Il grande vantaggio rispetto agli ASIC è che permettono di apportare eventuali modifiche o correggere errori semplicemente riprogrammando il dispositivo in qualsiasi momento.
Di contro sono antieconomici per applicazioni con elevati volumi di produzione (milioni di pezzi), poichè il prezzo unitario del dispositivo è superiore a quello degli ASIC che permettono in questi casi di ammortizzare gli elevati costi di progettazione, hanno prestazioni inferiori e una minore integrazione.

Generalmente le FPGA vengono programmate con linguaggi di alto livello come il *Verilog* o il *VHDL*. Con questi linguaggi ci si puo permettere di usare librerie di componenti predefiniti e lavorare con blocchi e macro-blocchi di alto livello. Non bisogna comunque dimenticare la possibilita di usare la modalità *schematic-entry* che consente un approccio veloce e semplificato a tale tecnologia collegando tra loro blocchi già forniti dal costruttore.
Molte case costruttrici (le principali sono: *Xilinx*, *Altera* ed *Actel*) forniscono gratuitamente sistemi di sviluppo che supportano quasi tutta la loro gamma di prodotti.

Gli sviluppi tecnologici e la crescente capacità di integrazione hanno portato al continuo incremento del numero di porte logiche integrabili su un unico chip tanto che gli ultimi modelli affiancano ai blocchi per la realizzazione di logica combinatoria e sequenziale, memorie dedicate, sommatori e moltiplicatori hardware, complessi sistemi di clock multifase, interfacce di I/O ad elevate prestazioni oltre a veri e propri microprocessori embedded (nei prodotti di fascia più alta), tanto da consentire la realizzazione di interi sistemi sul chip (SoC, System on Chip).

1.1.2 Struttura

Un dispositivo FPGA è un circuito a semiconduttore che contiene sia componenti logici che interconnessioni programmabili.
I componenti logici programmabili (o blocchi logici) permetto di implementare le porte logiche elementari, le funzioni combinatorie più complesse o anche semplici funzioni matematiche. Tramite un blocco logico si può quindi rappresentare una qualunque funzione logica.
L'architettura tipica di un dispositivo FPGA (figura 1.1), è caratterizzata da una matrice di blocchi logici programmabili, interconnessi fra loro con connessioni programmabili, programmate utilizzando delle strutture chiamate switch matrix, delle matrici nelle quali sono realizzati tutti i collegamenti ammissibili. La struttura prevede inoltre, una serie di blocchi di I/O (IOB) che rappresentano la destinazione di tutti i segnali che comunicano con dispositivi posti al di fuori della FPGA, nonchè di tutti i segnali non instradati correttamente all'interno dell'architettura.

Le famiglie di FPGA vengono suddivise in base a 3 aspetti: l'architettura generale, i tipi di blocchi logici e la tecnologia di programmazione.
L'architettura generale può essere organizzata secondo array simmetrici, per riga oppure tramite PLD (Programmable Logic Device) gerarchici. Le FPGA della Xilinx sono organizzate secondo array logici, quelle della Actel per riga, mentre

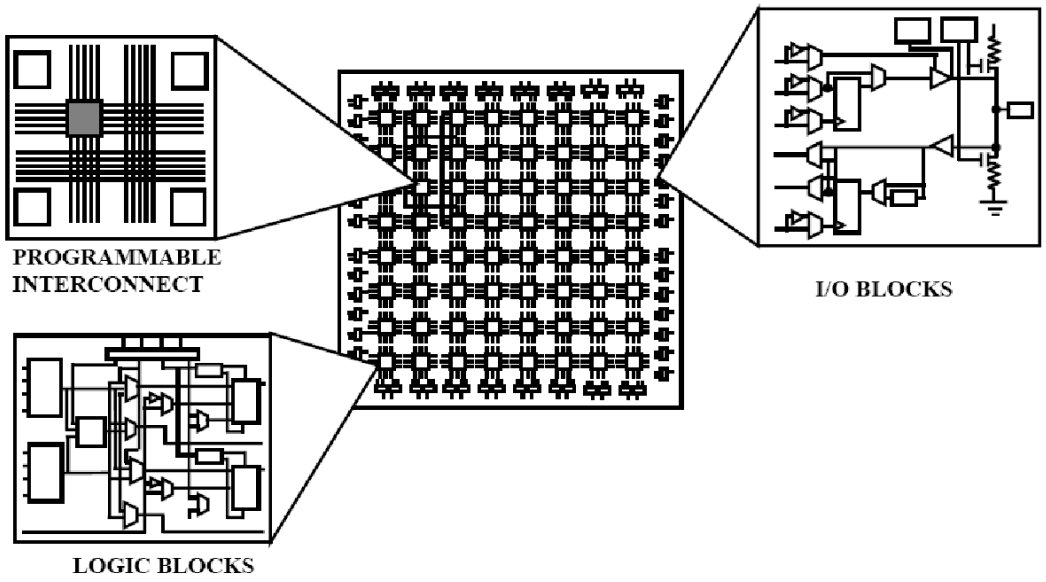
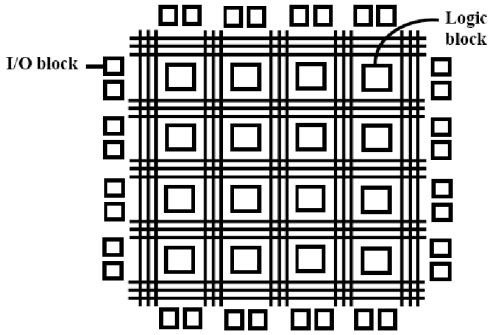
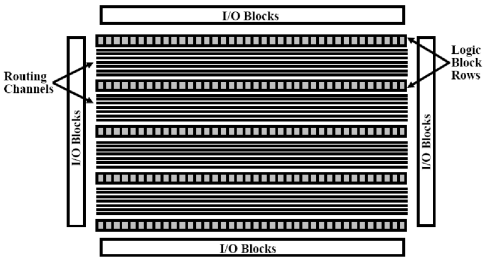


Figura 1.1: Struttura tipica di una FPGA.

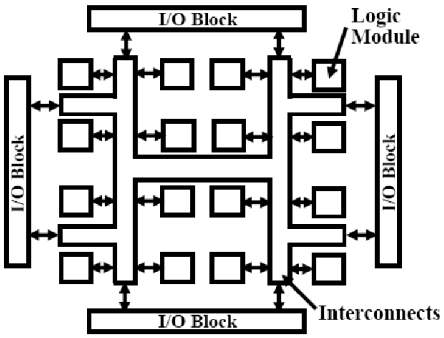
quelle della Altera secondo PLD gerarchici. In Figura 1.2 vengono riportati i 3 tipi di architettura precedentemente descritti.



(a) *Architettura secondo array simmetrici.*



(b) *Architettura organizzata per riga.*

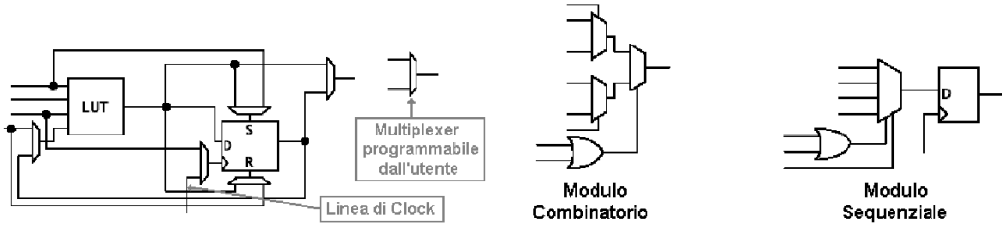


(c) *Architettura a PLD gerarchici.*

Figura 1.2: Architetturegenerali delle FPGA.

Anche i blocchi logici sono suddivisi in tre settori: quelli basati su LUT, Look-Up Table, quelli basati su multiplexer e quelli basati su PLD. I blocchi logici

delle FPGA della Xilinx, chiamati Configurable Logic Block (CLB), sono basati su LUT, quelli della Actel (Logic Modules(LM)) si basano su multiplexer, mentre quelli di Altera (Logic Array Block (LAB)) su PLD (i logic elements di Altera sono comunque molto simili a quelli di Xilinx). In Figura 1.3 vengono riportate le strutture dei blocchi logici adottate dalla Xilinx (a) e dalla Actel (b).



(a) *Xilinx.* (b) *Actel.*

Figura 1.3: Tipi di blocchi logici delle famiglie Xilinx e Actel.

Le tecnologie di programmazione sono tre e sono quelle basate sull'uso di celle RAM Statiche, quelle basate su EPROM ed EEPROM ed infine quelle basate sull'antifuse.

Tecnologia SRAM La soluzione è basata sull'utilizzo di RAM statiche in cui viene memorizzata la configurazione della FPGA.
Quando nella cella di SRAM è memorizzato un valore di tensione alto, il *pass gate* si comporta come un interruttore chiuso e puo essere utilizzato per connettere due linee, mentre quando la cella memorizza un valore basso, il transistor presenta una elevata resistenza tra le linee impedendone il collegamento come raffigurato in Figura 1.4.
A causa della natura volatile della SRAM, la FPGA deve essere configurata al momento dell'alimentazione del chip. I dispositivi basati su questa tecnologia necessitano quindi di una memoria esterna permanente per la conservazione dei bit di programmazione. Il maggior inconveniente di questa tecnologia, utilizzata nei dispositivi di Xilinx (e anche in alcuni di Altera), risulta l'elevata occupazione di area, compensata pero dalla rapida ed efficiente riprogrammabilità del dispositivo.

Tecnologia EPROM ed EEPROM (o a gate flottante) La tecnologia EPROM è un metodo simile a quello utilizzato nelle memorie EPROM. Il programma viene memorizzato senza un supporto di memorizzazione esterno della configurazione. I chip programmabili basati su EPROM non possono però essere riprogrammati *in-circuit* ed hanno bisogno di esser cancellati tramite l'esposizione ai raggi UV. La tecnologia EEPROM, è un metodo simile a quello utilizzato nelle memorie EEPROM. Anche in questo caso il programma viene memorizzato senza un supporto di memorizzazione esterno della configurazione. I chip programmabili basati sulle EEPROM possono essere cancellati elettricamente, ma generalmente non possono

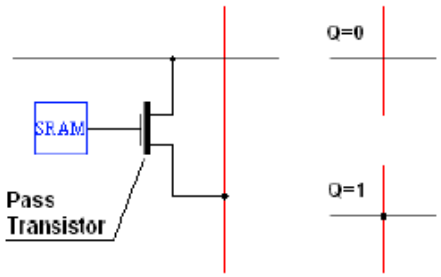


Figura 1.4: Tecnologia SRAM.

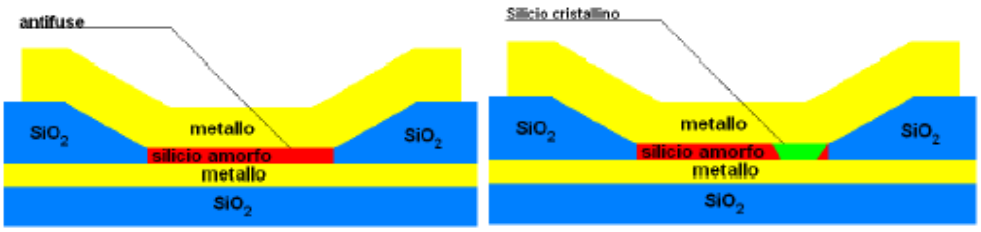
essere riprogrammati in-circuit.
Come per la tecnologia basata su SRAM il maggior vantaggio di questa soluzione (adottata nei dispositivi di Altera) è rappresentato dalla sua riprogrammabilità , inoltre non necessita di una memoria esterna permanente per programmare il chip, la realizzazione su silicio richiede però processi tecnologici piu complessi rispetto a quello CMOS standard.

Tecnologia antifuse La soluzione prevede l'utilizzo di *antifuse devices* per il collegamento dei vari blocchi logici. Un antifuse è un dispositivo a due terminali che presenta, prima della programmazione, un'elevata resistenza tra i suoi capi comportandosi come un circuito aperto, in figura 1.5(a) è rappresentata la sezione di un transistor MOSFET realizzato con tecnologia antifuse. La parte in rosso è composta da silicio amorfo che si comporta come un materiale isolante. Applicando tensioni molto alte ai terminali del dispositivo antifuse si crea un cammino permanente scarsamente resistivo, evidenziato in verde in figura 1.5(b), trasformando il silicio amorfo in silicio policristallino e realizzando in questo modo le interconnessioni sulla FPGA. I principali vantaggi di questa tecnologia sono le dimensioni estremamente ridotte, la resistenza in serie all'antifuse relativamente bassa e capacità parassite molto inferiori rispetto ad altre tecnologie. Va aggiunta anche la resistenza intrinseca di questa tecnologia alle radiazioni, per cui questa tecnologia risulta particolarmente impiegata in condizioni ambientali difficili: come sistemi spaziali, militari, nucleari ed ospedalieri.
La programmazione di dispositivi (ad esempio quelli della Actel) che utilizzano questa tecnologia non è reversibile, tale limitazione puo rappresentare uno svantaggio nel caso sia richiesta la riprogrammazione del dispositivo.

1.1.3 Linguaggi HDL

Introduzione

I linguaggi descrittivi HDL (Hardware Description Language) nascono negli anni '80 con lo scopo di documentazione di progetti hardware complessi. L'esigenza era essenzialmente quella di utilizzare una metodologia standard per descrivere il comportamento dei sistemi elettronici. Questi linguaggi sono molto simili a linguaggi software, ma l'obiettivo è completamente diverso: un HDL descrive comunque un



(a) *Circuito aperto.* (b) *Circuito chiuso.*

Figura 1.5: Sezione circuito antifuse.

sistema fisico, non un algoritmo. Se inizialmente questi processi erano sfruttati al solo scopo di documentazione, ben presto essi sono stati utilizzati per la simulazione al fine di verificare il corretto funzionamento del sistema. Ciò ha reso i linguaggi HDL vera parte attiva del processo di progettazione.
Nel corso degli anni, l'aumento, della complessità dei sistemi, permessa anche dal progredire incessante della tecnologia, ha reso necessario lo sviluppo di uno strumento in grado di descrivere sottosistemi ad alto livello di astrazione. E' così possibile semplificare il carico computazionale degli algoritmi di simulazione. Possiamo dire che, più alto è il livello di astrazione implementato da un blocco HDL, maggiore è la velocizzazione del processo di simulazione dell'intero progetto. Dunque, la tipologia adatta a essere sintetizzata tramite HDL va dal livello logico in su. Solitamente vengono descritti blocchi fino al livello RTL (Register Transfer Level): infatti la sintesi di blocchi appartenenti a un livello di astrazione troppo alto impedirebbe al progettista di vedere l'evoluzione dei diversi segnali all'interno del chip. L'operazione di individuazione degli errori sarebbe decisamente più complessa. Occorre quindi trovare un giusto compromesso in grado di massimizzare le funzionalità del codice.
I linguaggi HDL attualmente più utilizzati sono il VHDL (VHSIC, Very High Speed Integrated Circuits HDL) diffuso soprattutto in Europa, la cui sintassi richiama quella del Pascal e soprattutto dell'ADA, e il Verilog, diffuso soprattutto negli USA deve la sua fortuna al fatto che è più simile al C e inoltre è più semplice rispetto al VHDL.

VHDL

L'acronimo VHDL, come già detto, sta per VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language). VHDL è il risultato del progetto di ricerca VHSIC del Dipartimento della Difesa degli Stati Uniti d'America iniziato negli anni '80. Visto lo sviluppo della tecnologia VLSI, ci si rese conto che serviva un linguaggio ad alto livello che riuscisse a sintetizzare costrutti anche complessi. Nato con scopi militari nel 1987 venne definito lo standard (IEEE-1076-1987) aggiornato poi nel 1993 (IEEE-1076-1993). Osservando un codice VHDL si notano molte somiglianze con il Pascal e soprattutto l'ADA: molte parole chiave sono le stesse, in particolare quelle usate per i cicli e i costrutti condizionati.
La potenza del VHDL sta in costrutti di poche righe che descrivono circuiti di

decine di migliaia di gate, consentendo al progettista di concentrarsi sul comportamento del sistema e non sui dettagli di implementazione. Questo comporta, però, una profonda conoscenza del linguaggio, poiché, se il codice di un determinato blocco, anche se formalmente corretto, è scritto in maniera ridondante, la sintesi potrebbe non essere efficiente poiché è soltanto il sintetizzatore che decide l'implementazione a livello di gate. La differenza sostanziale rispetto ai comuni linguaggi di programmazione è che le istruzioni vengono eseguite non in successione ma in parallelo, richiedendo un approccio mentale al linguaggio totalmente diverso da quello tradizionale.

Un sistema hardware si può sempre rappresentare in questi termini: riceve degli input, esegue delle operazioni, produce degli output. In generale sarà costituito da uno o più sottosistemi rappresentabili in questi termini. Allora per descrivere un sottosistema si devono definire:

- la sua **interfaccia esterna**, ovvero gli ingressi e le uscite che rappresentano le sue relazioni con gli altri sottosistemi o con l'esterno;
- il suo **funzionamento interno**, ovvero che cosa fa il sistema e/o come lo fa.

In VHDL il modello di un sottosistema è detto design entity ed è costituito da due parti: una *entity* (che descrive l'interfaccia) e una *architecture* (che descrive il funzionamento).
Nella descrizione hanno un ruolo fondamentale i *segnali*, rappresentazione dei wires (conduttori), che spostano i dati tra parti diverse del sistema. Le *porte* dell'interfaccia esterna sono segnali particolari perchè spostano i dati da e verso l'esterno.
I *packages* e le *librerie* sono moduli che possono contenere porzioni di codice isolate dal resto della descrizione, allo scopo di rendere il codice riusabile.
Le entity, le architecture, i package (e i package body) sono detti **design units**. Un modello praticamente è fatto di design units al cui interno si costruisce la descrizione dettagliata del sistema.
Il modello di un sistema si descrive in un file sorgente che contiene istruzioni VHDL in formato testo.

Il VHDL supporta tre livelli di astrazione. Il più basso è quello della descrizione *Strutturale* o *Gate-Level*: il sistema viene rappresentato direttamente nella sua struttura come rete di porte logiche. Si indicano i componenti del sistema e le interconnessioni tra di essi. Questa descrizione è equivalente a quella che si ottiene con un disegno schematico, solo che non è grafica ma testuale.
La descrizione *Funzionale* o *Comportamentale (behavioural)* è il livello di astrazione più alto. Nel definire cosa il sistema deve fare, il progettista utilizza descrizioni algoritmiche fatte con istruzioni procedurali simili a quelle dei linguaggi di programmazione software.
Ad un livello intermedio di astrazione c'è il livello *RTL (Register Transfer Level)* o *data-flow* attraverso il quale il sistema viene descritto in termini di registri, logica combinatoria, bus e unità di controllo.

Il modello può contenere descrizioni di tipo diverso per parti diverse del sistema. Inoltre è evidente che, se di una parte del sistema viene data una descrizione strutturale, il funzionamento interno dei sottosistemi corrispondenti deve comunque

essere definito da qualche parte nel progetto o in moduli esterni.
Per poter realizzare il dispositivo, una descrizione funzionale o dataflow deve essere sintetizzata, ovvero tradotta in una strutturale. La sintesi automatica dei costrutti funzionali però non sempre è possibile. Quello che si può fare dipende dal supporto offerto dal tool di sintesi utilizzato.

Verilog

Il Verilog è un linguaggio di descrizione hardware che permette l'implementazione di sistemi digitali in un ampio spettro di livelli di astrazione: da quello logico a quello architetturale. Esso include, inoltre, costrutti gerarchici che permettono al progettista di controllare facilmente la complessità del sistema.
L'elemento base per la descrizione di un blocco in Verilog è il *modulo*. Quest'ultimo rappresenta un costrutto logico preso in uno dei suoi possibili livelli di astrazione. Le informazioni base contenute nel modulo sono: un'*interfaccia*, dove vengono dichiarati gli ingressi e le uscite, e il *corpo*, dove è sintetizzata la concreta descrizione funzionale del modulo stesso.
L'interfaccia di un modulo è divisa in due parti:

1. *Lista delle porte*: le porte sono i terminali di ingresso e uscita del modulo. L'elenco delle porte compare fra parantesi subito dopo il nome del modulo. Nella lista, le porte di input/output possono comparire in qualsiasi ordine;

2. *Dichiarazione delle porte*: subito dopo la prima riga devono essere definite tutte le porte che compaiono nella lista. Per ogni porta si definisce la direzione (input, output, inout), la dimensione (il numero di bit associati alla porta) e il tipo (wire, reg, ecc.). E' importante definire il giusto tipo per ogni variabile in gioco sia essa un ingresso, un'uscita o una variabile ausiliaria interna utilizzata per l'algoritmo.

Per com'è concepito, il Verilog viene spesso considerato più semplice e più vicino all'hardware di quanto non lo sia il VHDL; infatti può essere utilizzato anche per modelizzare circuiti logici a livello di transistor o interruttori, cosa che risulta particolarmente difficile con il VHDL. Per contro, il VHDL ha dei costrutti ad alto livello e delle tipologie astratte che il Verilog non possiede, rendendo il VHDL molto più adatto alla descrizione comportamentale.
Molti programmi di simulazione e di sintesi RTL accettano sia VHDL che il Verilog, ed è quindi possibile utilizzare il VHDL per la progettazione ad alto livello e il Verilog per la temporizzazione a basso livello post-sintesi.

1.1.4 Flusso di progetto

La realizzazione di un progetto su FPGA utilizzando linguaggi come VHDL e Verilog avviene seguendo il flusso descritto in figura 1.6.

Lo sviluppo parte con la descrizione delle specifiche riguardanti le funzionalità da realizzare e le temporizzazioni. Talvolta viene realizzato un modello behavioral a

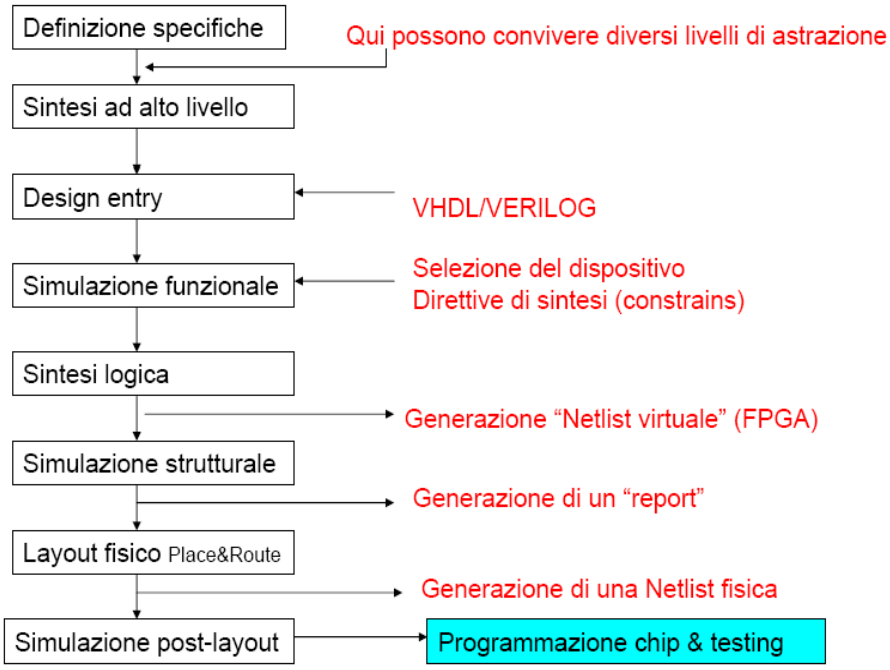


Figura 1.6: Flusso di progetto.

partire dalle specifiche, di norma viene realizzato direttamente un modello RTL sintetizzabile sin dall'inizio.
Il modello VHDL viene simulato e, se mostra il comportamento atteso, viene sintetizzato (passaggio automatico da una descrizione ad alto livello (comportamentale) ad una a basso livello (netlist)). Una volta sintetizzato, il progetto può essere implementato sul dispositivo, ma prima di ciò deve però essere verificata l'aderenza a una serie di vincoli elettrici e temporali, questo permette di determinare a priori se le potenzialità offerte dal dispositivo target consentono di rispettare o meno le specifiche di progetto iniziali.
A questa fase segue il place and route, che si incarica di assegnare le risorse disponibili ai vari blocchi circuitali del dispositivo, l'obiettivo del tool di place and route è di assicurare un uso più efficiente possibile delle connessioni e della logica, sempre nel rispetto dei vincoli temporali ed elettrici delle specifiche iniziali.
Il place and route genera un file binario che viene usato per programmare il chip. In vari punti di tutte queste fasi s'innestano delle opportune verifiche. La verifica finale, nota come simulazione full-timing consente di avere un quadro completo delle prestazioni in quanto sfrutta veri ritardi di routing introdotti dal dispositivo. La tendenza è di anticipare sempre più questa verifica per evitare di arrivare alla fine del progetto e scoprire di dovere ricominciare tutto daccapo.

1.2 Moduli IP

Un modulo IP (da Intellectual Property Module) nel caso più generale, è un blocco circuitale parzialmente o completamente pre-progettato, e disponibile per essere

istanziato all'interno di un circuito più complesso. Esistono 2 tipi di moduli IP:

- *Hard* IP module: si riferisce a un modulo IP progettato completamente (fino al layout), quindi che si presta ad essere utilizzato solo all'interno di una specifica piattaforma tecnologica;
- *Soft* IP module: si riferisce a un modulo IP parzialmente progettato, quindi portabile attraverso diverse piattaforme tecnologiche.

1.2.1 Soft IP

La portabilità di un modulo IP soft è supportata attraverso una descrizione del circuito a un livello abbastanza alto da non dipendere dai dettagli implementativi della tecnologia prescelta: descrizione VHDL o Verilog a livello RTL [1] con strumenti di sintesi semi–automatici, la descrizione può essere implementata in qualsiasi tecnologia (celle standard, gate array, FPGA, ...).
Si va da semplici funzioni aritmetico logiche a core di microprocessori.
Un modulo IP soft viene istanziato all'interno di un modello HDL come un qualsiasi altro componente creato dall'utente:

- Il modello compilato del modulo IP è in una libreria che deve essere visibile alla design unit in cui viene inserita l'istanza;
- Il modulo IP viene sintetizzato e implementato insieme al resto del circuito in cui è inserito.

Dunque qualunque modulo IP soft descritto in VHDL, Verilog o altro HDL sintetizzabile, può essere implementato in una FPGA (purché di dimensioni e prestazioni adeguate).

1.2.2 Hard IP

Un modulo IP hard si presenta nella forma di un blocco circuitale completamente pre-implementato (come un processore, un moltiplicatore...), progettato in modo da essere il più efficiente possibile in termini di: consumo di potenza, area e prestazioni. Di conseguenza un hard IP risulta legato strettamente alla tecnologia con cui viene realizzato il circuito.
Nel caso di realizzazione su FPGA si tratta di un modulo progettato facendo esplicito riferimento sia alle risorse hardware generiche (tipo di Logic Element e interconnessioni) che specifiche (moltiplicatori, DCM, block RAM...) disponibili all'interno del componente.
L'istanza in un modello HDL avviene come nel caso dei moduli soft, la differenza

[1] Register Transfer Level: descrizione del sistema in termini di registri, logica combinatoria, bus e unità di controllo.

principale è che un modulo hard è già sintetizzato e implementato va solo piazzato (come macroblocco) e interconnesso al resto del circuito; la sua struttura interna non cambia.

CAPITOLO 2

Microprocessori su FPGA

In questo capitolo vengono illustrate le modalità con cui è possibile implementare un micropocessore su una FPGA, concentrando l'attenzione sulla differenza tra realizzazione mediante *hard* e *soft*-processor.

2.1 Introduzione

Come accenato nel precedente capitolo su una FPGA è possibile implementare un processore e realizzare quindi un vero e proprio sistema embedded (in questo caso detto anche SOC: System on a chip).
I processori eventualmente disponibili sulle FPGA possono essere realizzati in due modi distinti. Una prima tipologia è rappresentata dagli hard-processor, costituiti da reti hardware fisse. Alcune FPGA sono in grado di fornire un processore di questo tipo, integrato direttamente sul silicio che le compone. In questo modo, all'utente viene offerta la possibilità di sfruttare questo componente mediante l'esecuzione di un software.
Una seconda tipologia di processori, che rivestono il ruolo centrale di questa tesi, sono i soft-processor , che risultano realizzabili anche su FPGA non equipaggiate di un hardware dedicato. Essi infatti vengono sintetizzati sulle celle presenti su tutte le FPGA, opportunamente programmate e collegate tra loro, risultando quindi una soluzione molto comoda e portabile.

2.2 Vantaggi di un sistema embedded su FPGA

La ragione principale nella scelta di una FPGA per la realizzazione di processori e quindi di sistemi embedded è sicuramente la capacità di compensazione tra hardware e software in modo da massimizzare l'efficienza e le prestazioni e garantire

una grande flessibilità. Oltre a questo la realizzazione di un sistema embedded su una FPGA offre una serie di vantaggi rispetto ai normali microprocessori, i più importanti sono [15]:

1. Personalizzazione;
2. Riduzione del costo e del numero di componenti;

Gli svantaggi principali risultano invece la complessità dei tools e dei software di design, i maggiori costi per grandi produzioni e le prestazioni (in genere) ridotte rispetto ad un approccio di tipo ASIC.

2.2.1 Personalizzazione

Il progettista di un sistema digitale integrato su FPGA dispone di ampia flessibilità nel selezionare le combinazioni di periferiche e controllori. Infatti il progettista può inventare nuove, e uniche, periferiche che possono essere direttamente collegate al bus del processore.
Se il progettista non ha particolari richieste (come il rispetto di determinati standard) riguardo al set di periferiche, questo puo essere semplicemente implementato mediante la FPGA.

2.2.2 Riduzione del costo e del numero di componenti

Grazie alla versatilità delle FPGA i sistemi integrati che richiedono numerosi componenti possono essere rimpiazzati da un singolo FPGA. Questo è, per esempio, il caso dei chip di I/O o dei co-processori necessari accanto ad un normale processore. Riducendo il numero di componenti in un progetto è possibile ridurre le dimensioni del circuito e lo studio dell'assemblaggio; entrambe queste caratteriastiche portano ad una diminuzione dei tempi di progetto e dei costi.

2.3 Hard-Processors vs Soft-Processors

Un hard-processor si presenta sotto forma di un blocco direttamente integrato sulla FPGA, quindi costituisce una parte dedicata del circuito. Esistono due diversi approcci per integrare un hard-processor su una FPGA.
Il primo consiste nel posizionarlo in una fascia situata a lato della struttura principale (array di blocchi logici) della FPGA, come mostrato in figura 2.1(a). Uno dei vantaggi di questa implementazione sta nel fatto che la matrice di blocchi risulta invariata rispetto al caso in cui il processore non sia presente, semplificando il lavoro ai tools di sviluppo utilizzati. L'altro vantaggio è che il produttore può aggiungere nella fascia in cui è presente il processore una serie di componenti (come memorie, periferiche...) in modo da aumentare le potenzialità del sistema.
Il secondo consiste invece nell'integrare il processore (o i processori) direttamente nella struttura principale della FPGA, come mostrato in figura 2.1(b). In questo

caso i tool di progetto devono tener conto della presenza del processore. Questo approccio comporta un aumento della velocità di comunicazione tra il processore è il resto del circuito in quanto risulta direttamente integrato nell'array di blocchi logici.

In opposizione all'integrazione diretta su FPGA (hard-processor) è possibile configurare una serie di blocchi logici in modo da farli lavorare come un microprocessore. Questa soluzione è nota con il nome di soft-processor. Un soft-processor è disponibile sotto forma di descrizione in linguaggio HDL, generalmente VHDL o Verilog, sintetizzabile.

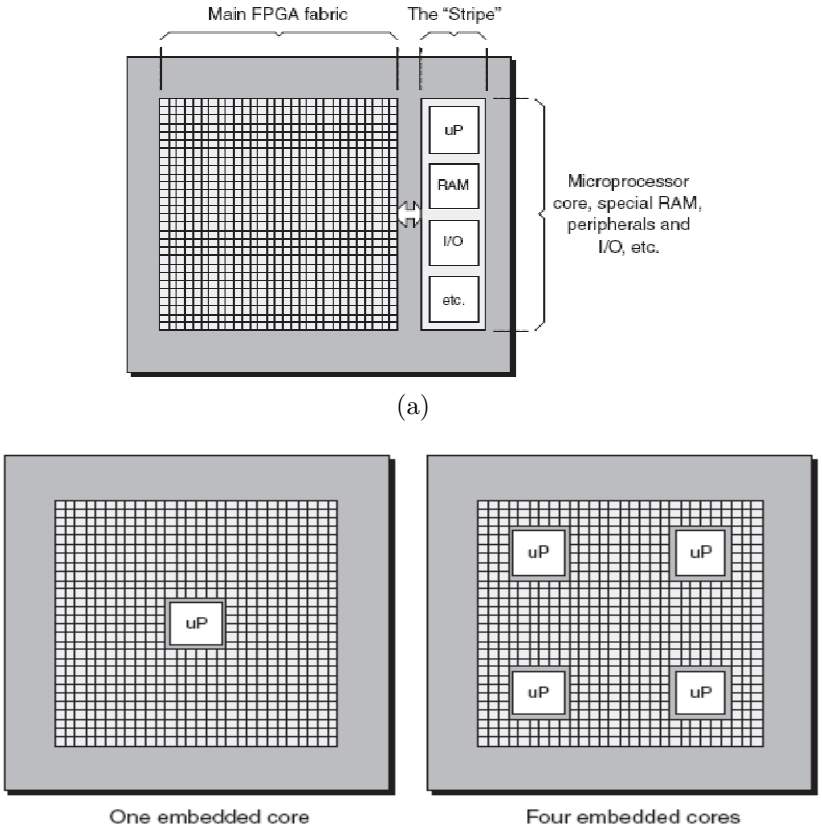


Figura 2.1: Integrazione di un hard-processor su FPGA.

Di seguito vengono riportati i principali vantaggi (e svantaggi) delle due diverse implementazioni. Gli hard-processors possono risultare veloci, piccoli e relativamente economici; tuttavia essi presentano diversi inconvenienti [23];

- Il numero di processori presenti nella FPGA potrebbe non coincidere con il numero richiesto dall'applicazione; ciò può comportare necessità o spreco di risorse;
- Le performance richieste da ciascun processore nell'applicazione potrebbero non coincidere con quelle fornite dai processori della FPGA;
- A causa della struttura stessa degli hard-processors (reti hardware fisse) il collegamento tra i processori e il resto della logica può risultare complicato;

- L'inclusione di o uno più hard-processor all'interno di una FPGA "specializza"il componente stesso, restringendo la clientela di base del prodotto.

Per quanto concerne i soft-processors invece, difficilmente eguagliano le prestazioni in termini di area e potenza degli hard-processors; tuttavia presentano una serie di vantaggi rispetto a quest'ultimi:

- Il progettista può implementare il numero esatto di processori richiesti dall'applicazione e sono i tools di CAD a occuparsi del piazzamento sul chip in modo da ottimizzare i collegamenti;
- Un soft-processor, a differenza di un hard-processor, può essere rimosso dal progetto quando non è richiesto il suo utilizzo, permettendo un notevole risparmio d'area;
- Poichè sono implementati in logica configurabile i soft-processors possono essere messi a punto variando la loro implementazione e complessità per soddisfare le esatte richieste dell'applicazione.

CAPITOLO 3

Soft-Processors

In questo capitolo, costituente la parte centrale di questo lavori di tesi, vengono analizzati in dettaglio i soft-processors, soffermandosi sulle differenze tra i prodotti forniti dalle varie case prodruttici.

3.1 Introduzione

Nel settembre del 2000, *Altera* [1] rilasciò in commercio il primo soft-processor, il `Nios`, a cui fece immediatamente seguito il `Microblaze`, realizzato da *Xilinx*.
Da quel momento la popolarità dei soft-processors è cresciuta a dismisura, tanto che oggi il 16% dei progetti in logica programmabile contengono un soft-processor; il 50% di tali progetti usa Nios, il 40% Microblaze mentre i rimanenti si dividono tra processori personalizzati o disponibili liberamente[27].

La tabella 3.1 riporta i principali soft-processors attualmente in circolazione, indicando inoltre alcune caratteristiche degli stessi (come lo sviluppatore, se si tratta o meno di un prodotto gratuito...)[46].

Dopo aver dato una panoramica generale, si passa ora ad esaminare in dettaglio alcuni dei componenti riportati nella tabella, soffermandosi principalmente sul Microcblaze e il Nios II, i quali rappresentano i soft-processor maggiormente urilizzati nel mondo dei sistemi embedded.

3.2 MicroBLAZE

MicroBlaze è un soft-processor , prodotto da Xilinx ed ottimizzato per le FPGA della medesima casa produttrice. L'inserimento di questo core all'interno di un

[1] Uno dei due maggiori produttori di high-end programmable logic device assieme a Xilinx.

Processor	Developer	Open Source	Bus Support	Notes	Project Home
TSK3000A	Altium	No Royalty–Free	Wishbone	32–bit R3000 style RISC Modified Harvard Architecture CPU	Embedded Design on Altium Wiki[29]
TSK51/52	Altium	No Royalty–Free	Wishbone/ Intel 8051	8–bit Intel 8051 instruction set compatible, lower clock cycle alternative	Embedded Design on Altium Wiki[29]
OpenSPARC T1	Sun	Yes		64–bit	OpenSPARC[30]
MicroBlaze	Xilinx	No	PLB, OPB, FSL, LMB		Xilinx MicroBlaze[31]
PicoBlaze	Xilinx	Yes			Xilinx PicoBlaze[32]
Nios, Nios II	Altera	No	Avalon		Altera Nios II[33]
Cortex–M1	ARM	No			Cortex-M1 Processor[34]
eSi-RISC	EnSilica	No	AMBA AXI and APB	Configurable as 16 or 32-bit. Supports ASIC and FPGA.	EnSilica eSi-RISC[35]
LatticeMico32	Lattice	Yes	Wishbone		LatticeMico32[36]
LEON 3	ESA	Yes	AMBA	SPARC V8 compatible in 25k gates	Gaisler[37]
OpenRISC	OpenCores	Yes		32-bit; Done in ASIC, Altera, Xilinx	OR1K[38]
AEMB	Shawn Tan	Yes	Wishbone	MicroBlaze EDK 3.2 compatible Verilog core	AEMB[39]
OpenFire	Virginia Tech CCM Lab	Yes	OPB, FSL	Binary compatible with the MicroBlaze	
PacoBlaze	Pablo Bleyer	Yes		Compatible with the PicoBlaze processors	Pacoblaze[40]
CPU86	HT-Lab	Yes		8088 compatible CPU in VHDL	cpu86[41]
xr16	Jan Gray	No	XSOC abstract bus	16–bit RISC CPU+SoC featured in Circuit Cellar Magazine # 116-118	XSOC/xr16 [42]
ERIC5	Entner Electronics	No		9-bit RISC, very small size, C-programmable	ERIC5[43]
YASEP	Yann Guidon	Yes AGPLv3	Direct SRAM	16 or 32 bits, VHDL and JavaScript, not ready, sort of anti-F-CPU	yasep.org[44]
ZPU	Zylin AS	Yes	Wishbone	Stack based CPU, configurable 16/32 bit datapath, eCos support	Zylin CPU[45]

Tabella 3.1: Lista dei principali soft processors in commercio.

sistema embedded è immediato utilizzando un software fornito proprio da Xilinx, chiamato EDK[2].
Un sistema embedded basato su Microblaze comprende:

- Un processore Microblaze;
- Una memoria locale on–chip;
- Un sistema di bus;
- Periferiche OPB (on–chip Peripheral Bus).

Microblaze, la cui descrizione è scritta in VHDL, può essere configurato in più di 70 modi diversi, permettendo il suo utilizzo in campi che vanno da semplici macchine a stati, fino a complessi controllori per applicazioni internet.

3.2.1 Caratteristiche principali

Si tratta di un soft processor di tipo RISC[3] a 32 bit le cui caratteristiche principali sono[25]:

- Set d'istruzioni ortogonale;
- Presenza di 32 registri general purpose e fino a 18 special purpose a 32 bit;
- Architettura di memoria di tipo Harvard: separazione tra memoria dati ed istruzioni (quindi presenza di 2 bus distinti per dati e istruzioni);
- Bus indirizzi a 32 bit (spazio d'indirizzamento di 4GB);
- Architettura pipeline a 3 o 5 stadi;
- Rappresentazione dei dati in formato Big endian[4]; tipi di dati supportati: word (32 bit), half word e byte;
- Supporta i bus PLB, OPB, FSL, LMB (per maggiori informazioni vedi appendice A);
- Prevede la gestione di: reset, eccezioni, break ed interrupts;
- L'implementazione richiede almeno 900 CLB;
- Strumenti di sviluppo software e hardware completi e possibilità di debug;

[2]Xililx Embedded Development System è un programma che permette rapidamente di configurare un microprocessore (Microblaze o PowerPc) e caricarlo su una board (Spartan, Virtex, ecc.).

[3]Reduced Instruction Set Computer: architettura caratterizzata da un set di istruzioni in grado di effettuare operazioni semplici che possono essere eseguite in tempi simili.

[4]Memorizzazione dei dati che inizia dal byte più significativo per finire col meno significativo.

- Possibilità di essere programmato usando linguaggi comuni quali ANSI-C e C++; Xilinx stessa ha fornito una versione del compilatore gcc con la quale è possibile compilare il codice sorgente affinchè venga eseguito dal soft processor;

- Disponibilità di dispositivi hardware opzionali: un moltiplicatore, un divisore, un barrel shifter, una FPU (Floating Point Unit) per gestire operazioni a virgola mobile; inoltre si può anche scegliere di utilizzare una memoria cache (sia per dati che per istruzioni) con dimensioni (fino a 64KB) e altre caratteristiche configurabili.

In figura 3.1 è riportato lo schema a blocchi del Microblaze comprensivo anche di alcune periferiche.

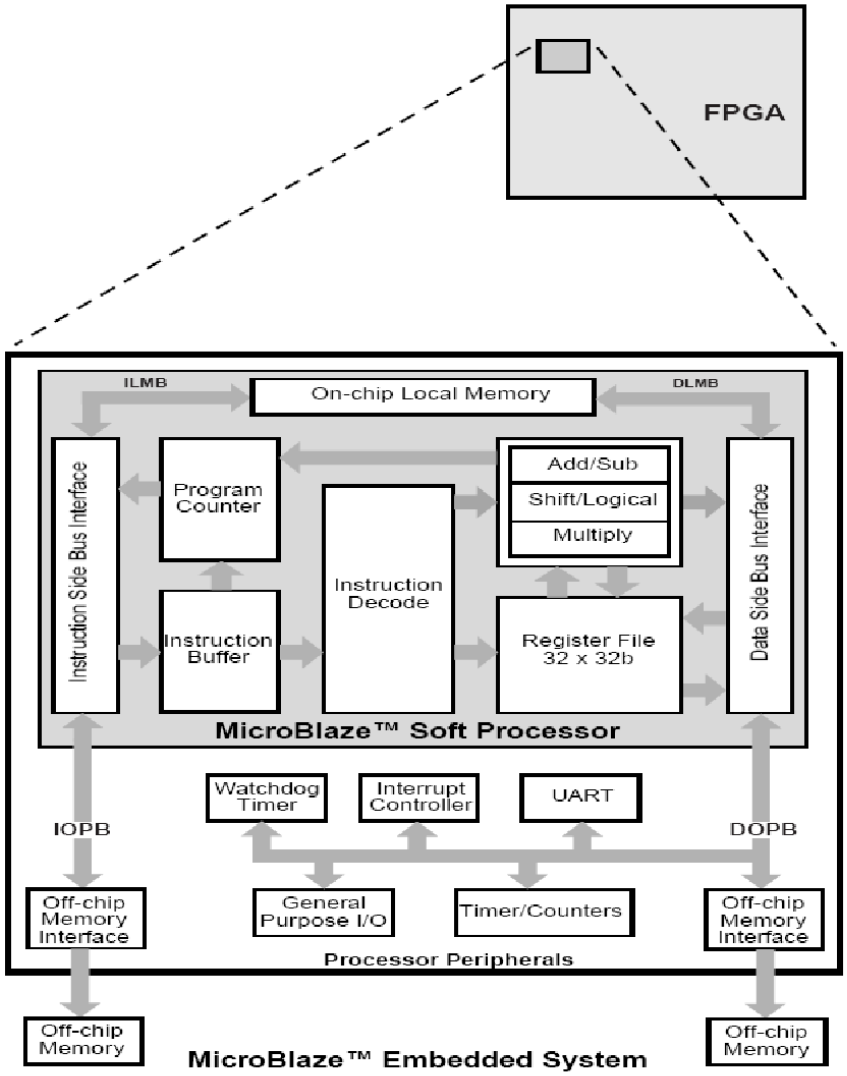


Figura 3.1: Schema a blocchi del microblaze.

3.2.2 Set d'istruzioni

Tutte le istruzioni di microblaze (che si dividono in:aritmetiche, logiche, di salto, di load/store e speciali) sono a 32 bit a 3 operandi (e 2 modalità di indirizzamento) e si dividono in 2 tipi (vedi figura 3.2):

- Tipo A, dette registro-registro, a significare che tutti i dati dell'istruzione provengono dal register file: contiene l'opcode, 2 registri sorgente e un registro destinazione;
- Tipo B, dette immediate: contiene l'opcode, un registro destinazione, uno sorgente e un campo immediato di 16 bit(eventualmente estendibile a 32).

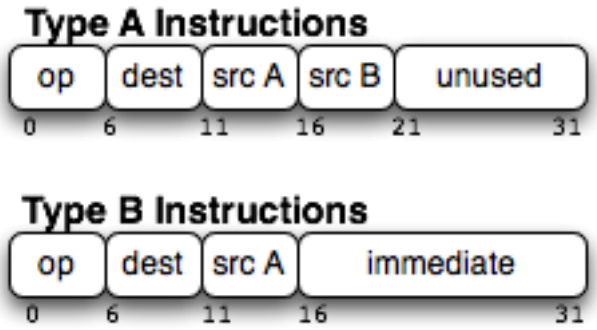


Figura 3.2: Istruzioni Microblaze.

3.2.3 Registri

General purpose I 32 registri general purpose sono indicati con R0–R31. Il register file è resettato al momento della programmazione e il valore di reset è 0x00000000 (mentre non viene resettato dai segnali d'ingresso di reset: `Reset` e `Debug_rst`.).
Nella tabella di figura 3.2 viene riportata una descrizione generale di ciascun registro.

Special purpose Come accennato in precedenza Microblaze dispone fino a 18 registri (il numero dipende dalle opzioni di configurazione) special purpose, tra cui i più importanti sono sicuramente:

- Program counter (PC): contiene l'indirizzo dell'istruzione corrente;
- Machine Status register (MSR): contiene bit di stato, controllo ed errore del processore (permette per esempio di abilitare o disabilitare interruzioni ed eccezioni);

Tra gli altri si possono ricordare l' Exception Address Register (EAR), l'Exception Status Register (ESR) e il Floating Point Status Register (FSR).

Bits	Name	Description	Reset value
0:31	R0	Always has a value of zero. Anything written to R0 is discarded.	0x00000000
0:31	R1 through R13	32-bit general purpose registers.	-
0:31	R14	32-bit register used to store return addresses for interrupts.	-
0:31	R15	32-bit general purpose register. Recommended for storing return addresses for user vectors.	-
0:31	R16	32-bit register used to store return addresses for breaks.	-
0:31	R17	If MicroBlaze is configured to support hardware exceptions, this register is loaded with the address of the instruction following the instruction causing the HW exception, except for exceptions in delay slots that use BTR instead; if not, it is a general purpose register.	-
0:31	R18	through R31 R18 through R31 are 32-bit general purpose registers.	-

Tabella 3.2: Rergistri general purpose del Microblaze.

3.2.4 Architettura pipeline

L'esecuzione delle istruzioni sfrutta la tecnica del pipeline. Per la maggior parte delle istruzioni l'esecuzione di ciascun stadio impiega un ciclo di clock; conseguentemente il numero di cicli per l'esecuzione di ciascuna istruzione dipende dal numero di stadi, ed ad ogni ciclo viene completata un'istruzione. Alcune operazioni possono richiedere più cicli per complentare lo stadio di esecuzione, a ciò viene posto rimedio ponendo la pipeline in stallo.
Esistono 2 versioni di Microblaze: una in cui viene privilegiata l'ottimizzazione dell'area, e una in cui vengono ottimizzate le prestazioni.
Nel primo caso (figura 3.3 la pipeline è divisa in 3 stadi per minimizzare i costi: fetch, decode e execute. Nel secondo caso (figura 3.4 la pipeline è divisa in 5 stadi

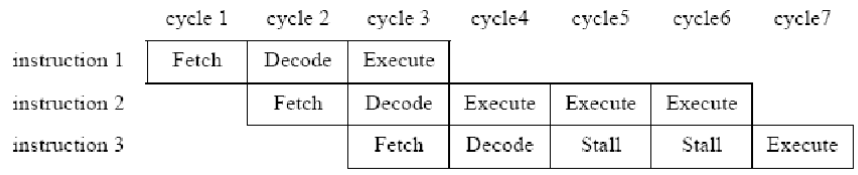


Figura 3.3: Pipeline a 3 stadi.

per massimizzare le prestazioni: Fetch (IF), Decode (OF), Execute (EX), Access Memory (MEM), e Writeback (WB).

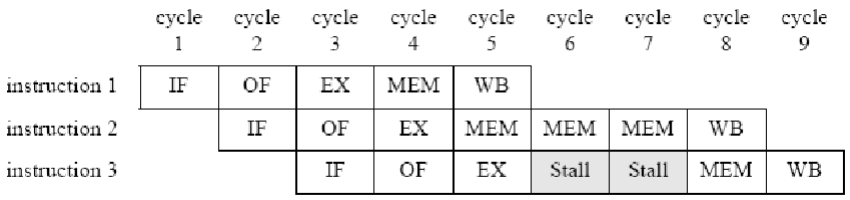


Figura 3.4: Pipeline a 5 stadi.

3.2.5 Interfaccia Esterna e Bus

Le interfacce di MicroBlaze, visibili in Figura 3.5, prevedono da una parte un collegamento con tutte le unità opzionali menzionate in precedenza, e dall'altra un collegamento ai bus esterni. Quest'ultimo è classificabile sotto lo standard IBM CoreConnect.
In MicroBlaze non è prevista la presenza del bus PLB, mentre esiste un'interfaccia che permette al soft processor di comunicare direttamente con il bus OPB. Questa scelta deriva dal fatto che per l'accesso a dati ed istruzioni vengono utilizzati delle soluzioni progettate dalla stessa Xilinx, ed ottimizzate per l'utilizzo con MicroBlaze. Il bus LMB (Local Memory Bus), infatti, permette la lettura in un tempo pari ad un ciclo di clock, dei dati e delle istruzioni presenti sulla memoria della FPGA, ossia sulle BRAM. L'interfaccia verso LMB è duplice: da un lato, ILMB è utilizzato per l'accesso alle istruzioni, dall'altro DLMB è utilizzato per l'accesso ai dati. E' comunque possibile escludere il bus LMB e memorizzare dati ed istruzioni su una memoria interfacciata al bus OPB, al quale MicroBlaze accede tramite due interfacce: DOPB per i dati, e IOPB per le istruzioni.
Altre soluzioni previste nello specifico per MicroBlaze sono i bus XCL (Xilinx CacheLink) ed FSL (Fast Simplex Link). XCL è un'ulteriore interfaccia per l'accesso alla memoria, qualora si volesse accedere a dati ed istruzioni senza utilizzare LMB o OPB. Anche in questo caso l'interfaccia è duplice, ma l'utilizzo è tipicamente limitato a memorie esterne che adottano specificatamente questo tipo di trasferimento. FSL rappresenta invece l'alternativa al DCR bus, in quanto realizza un collegamento diretto tra un registro del processore ed un core presente sulla FPGA, trasferendo un dato in due cicli di clock.

3.2.6 Periferiche OPB

Al processore Microblaze possono essere collegate una serie di periferiche che forniscono un'ampia gamma di funzioni, tra cui si possono menzionare: Watchdog Timer/Timebase, General purpose Timer/Counters, Interrupt Controller, SRAM Controlller, Flash Memory Controller, ZBT Memory Controlller, BRAM Controller, DDR Controller, SDRAM Controller, UART Lite, General purpose I/O, SPI, I2C, UART 16450/550, Ethernet 10/100 MAC.

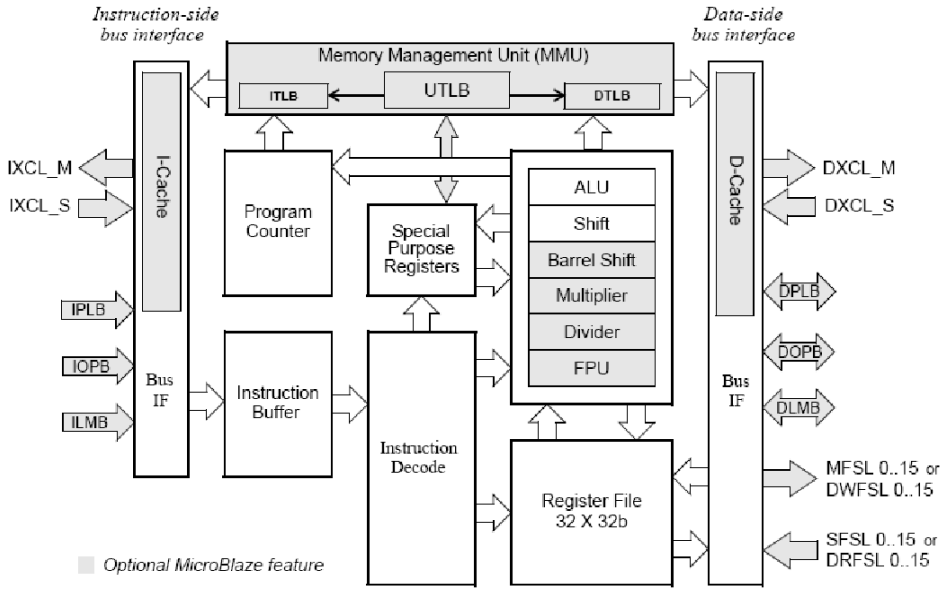


Figura 3.5: Interfacce di Microblaze.

3.2.7 Reset, Eccezioni, Break ed Interrupts

Come detto in precedenza Microblaze prevede la gestione (vettorizzata) del reset, dell'eccezioni, dei break e degli interrupt. L'ordine di priorità (decrescente) di questi è il seguente:

1. Reset
2. Hardware Exception
3. Non-maskable Break
4. Break
5. Interrupt
6. User Vector (Exception)

Nella tabella di figura 3.3 vengono riportati gli indirizzi di memoria dei vettori associati a ciascun evento e il registro contenente il valore dell'indirizzo di ritorno. Da notare come ogni vettore allochi 2 indirizzi, in modo da permettere un salto a qualsiasi locazione di memoria.

3.2.8 Ambiente di sviluppo EDK

Xililx Embedded Development System[5] è un programma che permette rapidamente di configurare un microprocessore (Microblaze) e caricarlo su una board (Spartan,

[5]Di tale pacchetto di tool nella realizzazione di un sistema embedded viene utilizzato lo strumento Xilinx Platform Studio, XPS per la parte hardware, mentre lo strumento Xilinx Software Development Kit (SDK) per la parte software (basato sul GNU toolchain). Un esempio di utilizzo è riportato in [47]

Evente	Vector Adress	Register File Return Address
Reset	0x00000000-0x00000004	-
User Vector (Exception)	0x00000008-0x0000000C	Rx
Interrupt	0x00000010-0x00000014	R14
Break: Non-maskable hardware	0x00000018-0x0000001C	R16
Break: Hardware	0x00000018-0x0000001C	R16
Break: Software	0x00000018-0x0000001C	R16
Hardware Exception	0x00000020-0x00000024	R17 or BTR
Reserved by Xilinx for future use	0x00000028-0x0000004F	-

Tabella 3.3: Indirizzi dei vettori associati a eccezioni, reset, break e interrupts.

Virtex, ecc.). In particolare è possibile settare la velocità del processore, la dimensione della memoria, le sue interfacce con l'esterno e si possono inoltre importare una o più periferiche esterne (descritte in codice VHDL o Verilog) da collegare col processore, effettuandone manualmente le connessioni e settandone gli indirizzi. Tra le periferiche esterne si può contemplare anche un secondo microprocessore da utilizzare come co-processore. Un altro aspetto interessante di Xilinx EDK è costituito dal fatto che è possibile programmare il/i microprocessore/i direttamente in linguaggio C anzichè in assembly, semplificando notevolmente i compiti del programmatore. Una volta caricato il programma nella memoria del processore è possibile effettuare il debug oppure scaricare il tutto nella board. Il progetto viene convertito in un file .bit che rappresenta il bitstream da inviare alla FPGA per programmarne i collegamenti.

3.2.9 Versioni Open Source

In commercio sono disponibili, gratuitamente, dei cloni di MicroBLAZE: l' AEMB e l'OpenFire.

OpenFire Si tratta di una versione open source di MicroBLAZE realizzata in Verilog. Il processore è *binary-compatible* con MicroBLAZE, ovvero qualsiasi applicazione (file binario) realizzata per essere implementata su MicroBLAZE può essere eseguita anche su OpenFire. Tale processore, realizzato da Stephen Craven (studente della Virginia Tech), è rilasciato sotto licenza MIT ed è stato progettato per essere utilizzato nei SCMP (Single Chip, Multiple Processor) e negli ASIP (Application Specific Instruction-set Processor).
Per magggiori informazioni su OpenFire si veda [54].

AEMB Prodotto da Shawn Tan è disponibile sotto licenza LGPL3. Rispetto a MicroBLAZE esso utilizza il bus WISHBONE quindi l'architettura non è comple-

tamente compatibile con l'originale. Per maggiori informazioni su tale processore si veda [24].

3.3 Nios II

Nios II è un soft-processor prodotto da Altera ed ottimizzato per le FPGA della medesima casa produttrice. Rispetto al più vecchio Nios, tale processore presenta una serie di migliorie che hanno portato alla sua utilizzazione in numerose applicazioni: dai DSP fino ai sistemi di controllo.
Un sistema embedded basato su Nios II presenta per lo più le stesse caratteristiche di un sistema embedded basato su Microblaze.

3.3.1 Caratteristiche principali

Si tratta di un soft-processor di tipo RISC a 32 bit che può essere implementato in 3 diverse configurazioni[3]:

- Nios II/f (f sta per fast) progettato per massimizzare le prestazioni; più in dettaglio per:
  - Massimizzare l'efficienza d'esecuzione delle istruzioni;
  - Ottimizzare la latenza degli interrupt;
  - Massimizzare la frequenza di lavoro.

  L'aumento delle prestazioni e delle opzioni di configurazione è ottento a spese delle dimensioni del processore. Il processore risultante viene utilizzato in applicazioni in cui sono richieste elevate prestazioni.
- Nios II/s (s sta per standard) progettato per ottenere un buon compromesso tra risorse utilizzate (quindi costo) e prestazioni.
- Nios II/e (e sta per economy) progettato per ridurre al minimo le dimensioni e quindi le risorse utilizzate. L'ottimizzazione delle dimensioni è ottenuta a spese delle prestazioni ottenibili e del numero di opzioni di configurazione a carico dell'utente.

Le caratteristiche principali di questo processore (comuni a ciascuna configurazione) sono:

- Set d'istruzioni a 32 bit;
- Presenza di 32 registri general purpose, fino a 32 registri di controllo a 32 bit e possibilità di definire insiemi di registri shadow;
- Architettura di memoria di tipo Harvard: separazione tra memoria dati ed istruzioni (quindi presenza di 2 bus distinti per dati e istruzioni);

- Bus indirizzi a 32 bit;
- Rappresentazione dei dati in formato Big endian o little endian (opzione di configurazione); tipi di dati supportati: word (32 bit), half word e byte;
- Supporta il bus avalon(per maggiori informazioni vedi appendice A);
- Prevede la gestione di: reset, eccezioni, break ed interrupts;
- Strumenti di sviluppo software e hardware completi e possibilità di debug;
- Possibilità di essere programmato usando linguaggi comuni quali ANSI-C e C++;
- Due modalità operative;
- Prestazioni fino a 250 DMIPS[6].

In figura 3.6 viene riportato il diagramma a blocchi del processore Nios II, mentre nelle tabelle delle figure 3.8, 3.10 e 3.9 vengono messe a confornto le principali caratteristiche delle tre diverse implementazioni del Nios II.

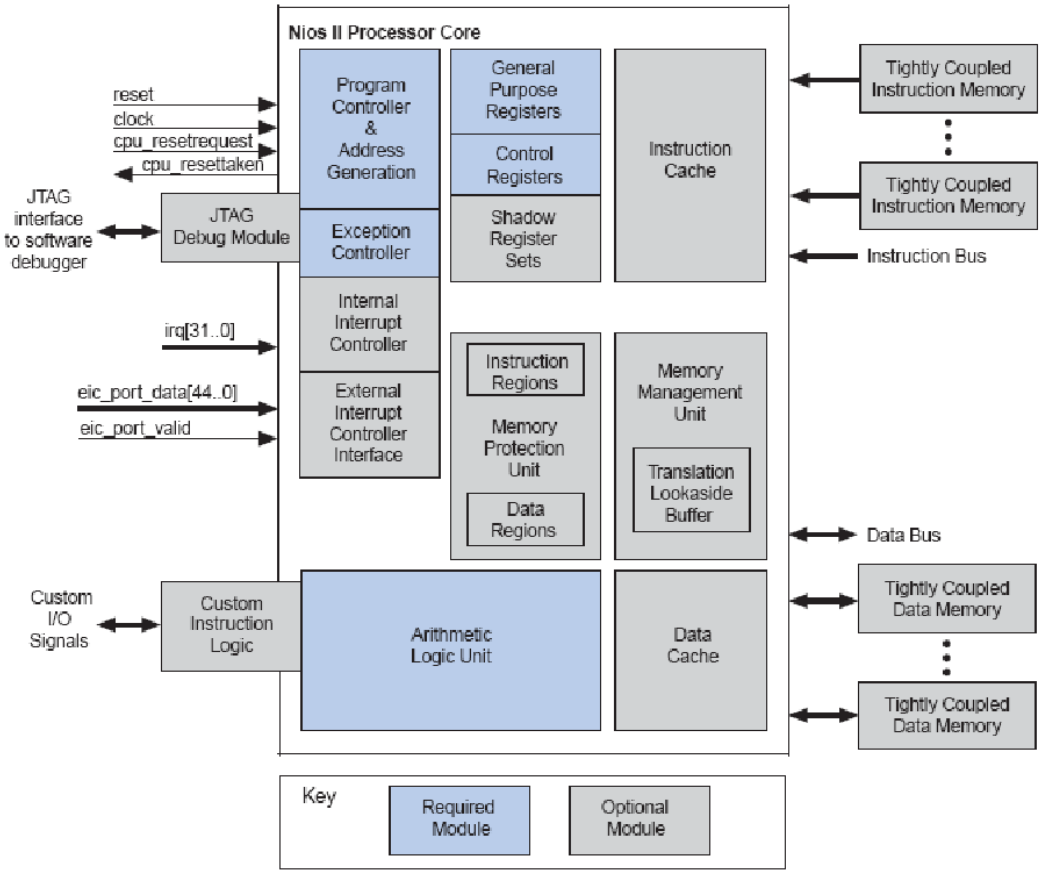


Figura 3.6: Schema a blocchi del processore Nios II.

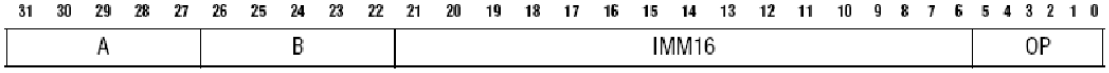
[6]Dhrystone Million Instructions Per Second.

3.3.2 Set d'istruzioni

Tutte le istruzioni del processore sono a 32 bit. In aggiunta alle istruzioni macchina, direttamente eseguite dal processore, il set d'istruzioni del Nios II comprende anche una serie di *pseudoistruzioni* che possono essere usate in programmi assembly come normali istruzioni (assembly). Sarà poi l'assembler che si occuperà di rimpiazzre tali istruzioni con una o più istruzioni macchina[1]. Vi è inoltre la possibilità di definire fino a 256 istruzioni personalizzate.
Esistono tre diversi formati per le istruzioni:

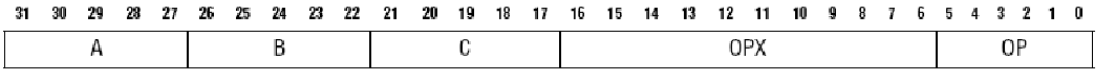
- Tipo–I (la I indica la presenza di un valore immediato), formate da (figura sottostante):
  - Opcode (OP), 6 bit;
  - 2 campi da 5 bit, detti A e B, che specificano i registri a cui l'istruzione fa riferimento;
  - Un campo immediato di 16 bit (IMM16), eventualmente estendibile a 32.

  Nella maggior parte dei casi i campi A e IMM16 specificano gli operandi sorgente mentre il campo B il registro destinazione. Tale formato include istruzioni logico/aritmetiche, di salto, di load/store e di gestione della memoria cache.



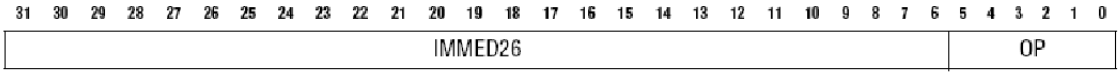
- Tipo–R (la R indica che tutti gli operandi sono specificati da registri), formate da (figura sottostante):
  - Opcode (OP), 6 bit;
  - 3 campi da 5 bit, detti A, B e C, che specificano i registri a cui l'istruzione fa riferimento;
  - Un campo di 11 bit per l'estensione dell'opcode (OPX).

  Nella maggior parte dei casi i campi A e B specificano gli operandi sorgente mentre i il campo C specifica l'operando destinazione. Tale formato include istruzioni logico/aritmetiche, di confronto, personalizzate e altre istruzioni che richiedono solo registri.



- Tipo–J (la R indica che si tratta di istruzioni di salto), formate da (figura sottostante):
  - Opcode (OP), 6 bit;
  - Un campo dati immediato di 26 bit.

Questo tipo di istruzioni trasferisce l'esecuzione in qualsiasi punto in un range di 256–MB.



3.3.3 Registri

General Purpose Come accennato in precedenza il processore Nios II è dotato di 32 registri general purpose indicati con r0–r31. Alcuni di questi registri sono utilizzati per determinati compiti e hanno dei nomi speciali riconusciuti dall'Assambler come mostrato in figura 3.4 (a).

Register	Name	Function	Register	Name	Function
r0	zero	0x00000000	r16		
r1	at	Assembler temporary	r17		
r2		Return value	r18		
r3		Return value	r19		
r4		Register arguments	r20		
r5		Register arguments	r21		
r6		Register arguments	r22		
r7		Register arguments	r23		
r8		Caller-saved register	r24	et	Exception temporary
r9		Caller-saved register	r25	bt	Breakpoint temporary
r10		Caller-saved register	r26	gp	Global pointer
r11		Caller-saved register	r27	sp	Stack pointer
r12		Caller-saved register	r28	fp	Frame pointer
r13		Caller-saved register	r29	ea	Exception return address
r14		Caller-saved register	r30	ba	Breakpoint return address
r15		Caller-saved register	r31	ra	Return address

Tabella 3.4: Rergistri general purpose del Nios II.

Registri di controllo Il Nios II è caratterizzato dalla presenza di fino a 32 registri di controllo, i quali riportano lo stato del processore e modificano il suo comportamento. L'acceso a questi registri è diverso rispetto ai registri general purpose, vengono infatti utilizzate delle istruzioni speciali. La tabelle di figura 3.5 (b) riporta l'elenco di tali registri, per maggiori informazioni vedi [4].

Registri shadow Il Nios II prevede la possibilità di defire uno o più insiemi di registri shadow. Un insieme di registri shadow rappresenta un'alternativa rispetto a un insieme di registri general purpose che può essere usato per mantenere un

Register	Name
0	status
1	estatus
2	bstatus
3	ienable
4	ipending
5	cpuid
6	Reserved
7	exception
8	pteaddr[1]
9	tlbacc[1]
10	tlbmisc[1]
11	Reserved
12	badaddr
13	config[2]
14	mpubase[2]
15	mpuacc[2]
16-31	Reserved

Tabella 3.5: Rergistri di controllo del Nios II.
1 Avviabile solo quando è presente la MMU, altrimenti Reserved. *2* Avviabile solo quando è presente la MPU, altrimenti Reserved.

contesto separato per le routine di servizio degli interrupt (ISR). Un processore Nios II può avere fino a 63 insiemi di registri shadow il cui comportamento è analogo a quello dell'insieme formato dai 32 registri general purpose.

3.3.4 Indirizzamento

Il processore prevede 5 diverse modalità di indirizzamento:

- Indirizzamento Immediato: l'operando, a 16 bit, è parte dell'istruzione quindi non è richiesto nessun accesso in memoria;
- Indirizzamento a registro: l'operando è in un registro del proccessore;
- Indirizzamento con spiazzamento: l'indirizzo dell'operando è dato dalla somma del contenuto di un registro e di un valore (16 bit con segno), detto spiazzamento, presente direttamente nell'istruzione;
- Indirizzamento registro indiretto: l'indirizzo dell'operando è il contenuto di un registro specificato nell'istruzione; tale tipo di indirizzamento coincide con quello precedente assumendo il valore di spiazzamento pari a 0;
- Indirizzamento assoluto: L'indirizzo dell'operando corrisponde al valore immediato di 16 bit specificato nell'istruzione; tale tipo di indirizzamento corrisponde a quello con spiazzamento se si utilizza il registro r0, il quale contiene sempre il valore 0.

3.3.5 Accesso alla memoria e ai dispositivi I/O

La figura 3.7 mostra come il processore Nios II sia in grado di accedere alla memoria ed ai dispositivi di I/O (i quali risultano mappati in memoria).
Per incrementare le prestazioni il Nios II/f può includere memorie cache per dati e istruzioni. Le caches vengono implementate nei blocchi di memoria della FPGA. Il loro uso è opzionale e le loro specifiche (tra cui la dimensione) sono scelte in fase di generazione del sistema. Il Nios II/s, invece, può avere una memoria cache per le istruzioni ma non per i dati, mentre il Nios II/e non è dotato di memorie cache.
Un altro modo di aumentare la velocità di accesso alla memoria e quello di utilizzare memorie per dati e istruzioni di tipo *tightly coupled*; in questo caso il processore accede direttamente a tali memorie senza passare atraverso la rete Avalon. L'acceso a questo tipo di memoria bypassa le memorie cache.
L'acesso alle locazioni di memoria e ai dispositivi di I/O avviene tramite istruzioni di Load/Store.

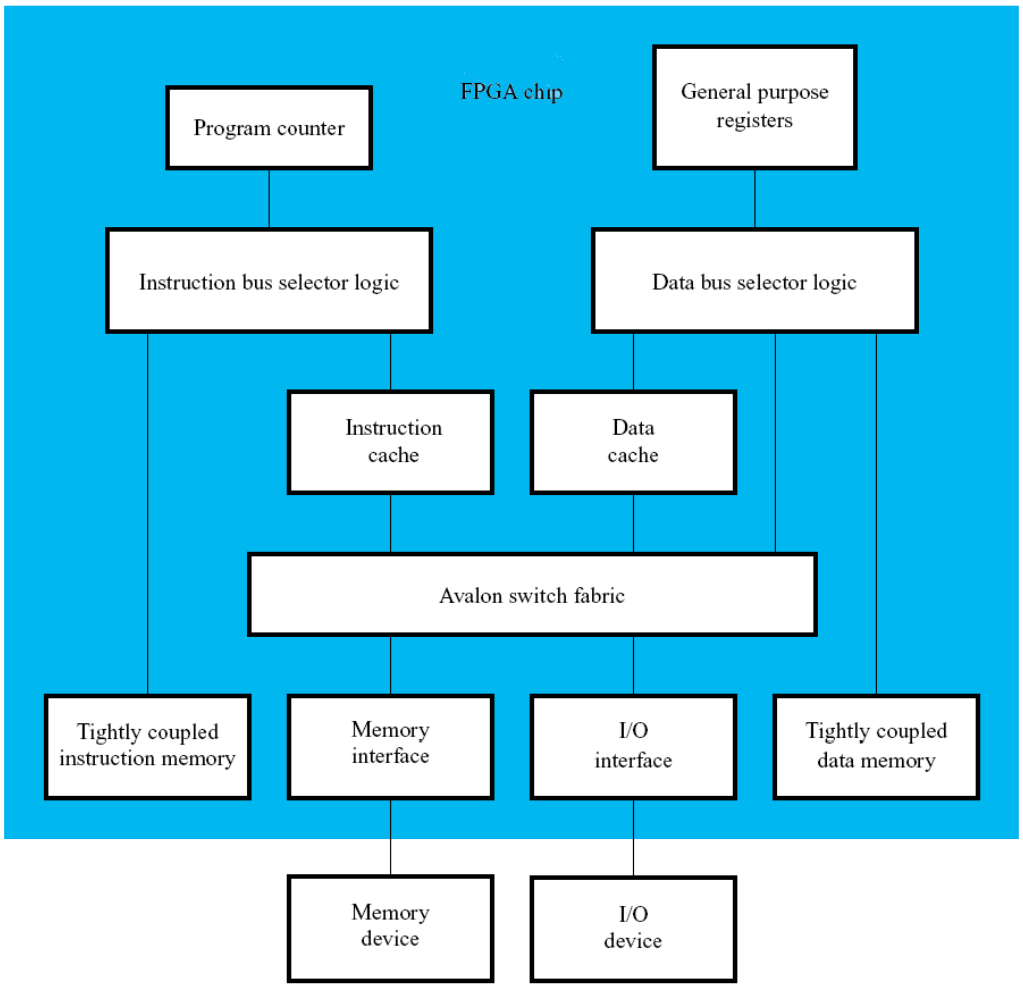


Figura 3.7: Organizzazione Memoria e I/O.

3.3.6 Modalità operative

La modalità operativa di un processore indica il modo in cui opera il processore, come gestisce la memoria e come avviene l'accesso alle periferiche. Il Nios II prevede 2 modalità operative:

- Modo supervisore (supervisor mode): questà modalità permette al processore di eseguire qualunque istruzione e funzione messa a disposizione dall'architettura e di accedere a qualsisasi area di memoria.
  I programmi (codici) che necessitano di accedere e controllare il processore vengnono eseguiti in questa modalità, quindi i codici del sistema operativo vengono eseguiti in tale modalità. Il processore entra in questa modalità anche quando si verificano eccezioni, break e reset.

- Modo utente (user mode): questa modalità è avviabile solo quando il processore è dotato della MMU[7] o della MPU[8] (quindi eventualmente solo nel Nios II/f). Il processore è in modalità utente quando è in esecuzione il codice applicativo, o anche il codice del sistema operativo che non richiede i privilegi propri della modalità supervisore. Questa modalità è sottoposta a limitazioni, tra cui il set d'istruzioni e le funzioni eseguibili. Il sistema operativo determina quali indirizzi di memoria sono accessibili in tale modalità. I codici eseguiti in user mode usano le `chiamate al sistema` per richiedere al sistema operativo l'esecuzioni di operazioni di I/O di gestione della memoria e altre funzioni tipiche della modalità supervisore.

3.3.7 Tool di sviluppo

Lo sviluppo di un sistema basato su Nios II consiste in 2 fasi distinte: generazione dell'hardware e creazione del software[48]. A tale scopo Altera mette a disposizione l'applicazione Embedded Development Suite (EDS) che contiene un ambiente di sviluppo completo per gestire entrambi le fasi:

- Processo di generazione hardware: per configurare e generare il sistema viene utilizzato il SOPC (System-on-a-Programmable-Chip) Builder, un componente del pacchetto Quartus-II. L'interfaccia grafica graphical user interface (GUI), messa a disposizione dal paccchetto, permette di scegliere le caratteristiche del Nios II da implementare e di aggiungere periferiche e blocchi di I/O (come Timers, interfacce seriali...) al sistema. Una volta terminata la configurazione, sempre attraverso Quartus-II, è possibile effettuare la sintesi e il place & route del sistema sulla FPGA.

[7]Memory management unit, è una classe di componenti hardware che gestisce le richieste di accesso alla memoria generate dalla CPU. La MMU può avere vari compiti tra cui la traslazione (o traduzione) degli indirizzi virtuali in indirizzi fisici (necessaria per la gestione della memoria virtuale), la protezione della memoria, il controllo della cache della CPU, l'arbitraggio del bus, e, in architetture più semplici (specialmente nei sistemi a 8-bit), la commutazione di banchi di memoria.

[8]Memory protection unit, insieme di componenti hardware che si occupa della protezione della memoria.

- Processo di creazione del software: per la gestione dello sviluppo del software viene utilizzato il pacchetto EDS (stesso nome dell'applicazione). Basato sull'IDE Eclipse, il pacchetto contiene un compilatore C/C++, un debugger ed un simulatore. EDS permette quindi al programmatore di simulare, caricare ed eseguire sulla FPGA il proprio codice.

Feature		Core		
		Nios II/e	**Nios II/s**	**Nios II/f**
Objective		Minimal core size	Small core size	Fast execution speed
Performance	DMIPS/MHz *(1)*	0.15	0.74	1.16
	Max. DMIPS *(2)*	31	127	218
	Max. f_{MAX} *(2)*	200 MHz	165 MHz	185 MHz
Area		< 700 LEs; < 350 ALMs	< 1400 LEs; < 700 ALMs	Without MMU or MPU: < 1800 LEs; < 900 ALMs With MMU: < 3000 LEs; < 1500 ALMs With MPU: < 2400 LEs; < 1200 ALMs
Pipeline		1 stage	5 stages	6 stages
External Address Space		2 GB	2 GB	2 GB without MMU 4 GB with MMU

Figura 3.8: Caratteristiche delle 3 implementazioni del Nios II: parte 1.

Feature	Core		
	Nios II/e	**Nios II/s**	**Nios II/f**
Shadow Register Sets	No	No	Optional, up to 63
User Mode Support	No; Permanently in supervisor mode	No; Permanently in supervisor mode	Yes; When MMU or MPU present
Custom Instruction Support	Yes	Yes	Yes

Figura 3.9: Caratteristiche delle 3 implementazioni del Nios II: parte 2.

Feature		Core		
		Nios II/e	Nios II/s	Nios II/f
Instruction Bus	Cache	–	512 bytes to 64 KB	512 bytes to 64 KB
	Pipelined Memory Access	–	Yes	Yes
	Branch Prediction	–	Static	Dynamic
	Tightly-Coupled Memory	–	Optional	Optional
Data Bus	Cache	–	–	512 bytes to 64 KB
	Pipelined Memory Access	–	–	–
	Cache Bypass Methods	–	–	■ I/O instructions ■ Bit-31 cache bypass ■ Optional MMU
	Tightly-Coupled Memory	–	–	Optional
Arithmetic Logic Unit	Hardware Multiply	–	3-cycle *(3)*	1-cycle *(3)*
	Hardware Divide	–	Optional	Optional
	Shifter	1 cycle-per-bit	3-cycle shift *(3)*	1-cycle barrel shifter *(3)*
JTAG Debug Module	JTAG interface, run control, software breakpoints	Optional	Optional	Optional
	Hardware Breakpoints	–	Optional	Optional
	Off-Chip Trace Buffer	–	Optional	Optional
Memory Management Unit		–	–	Optional
Memory Protection Unit		–	–	Optional
Exception Handling	Exception Types	Software trap, unimplemented instruction, illegal instruction, hardware interrupt	Software trap, unimplemented instruction, illegal instruction, hardware interrupt	Software trap, unimplemented instruction, illegal instruction, supervisor-only instruction, supervisor-only instruction address, supervisor-only data address, misaligned destination address, misaligned data address, division error, fast TLB miss, double TLB miss, TLB permission violation, MPU region violation, internal hardware interrupt, external hardware interrupt, nonmaskable interrupt
	Integrated Interrupt Controller	Yes	Yes	Yes
	External Interrupt Controller Interface	No	No	Optional

Figura 3.10: Caratteristiche delle 3 implementazioni del Nios II: parte 3.

3.4 Cortex-M1

Cortex-M1 è il primo processore `ARM`[9] progettato per essere implementato sulle FPGAs, principalmente su quelle delle `Actel` (ma anche in quelle di altre case produttrici), caratterizzato da alte prestazioni e piccole dimensioni. Si tratta di un processore RISC a 32 bit le cui caratteristiche principali sono:

- Set d'istruzioni composto da un sottoinsieme del set Thumb-2 (architettura ARMv6-M);
- Frequenza di lavoro fino a 60MHz;
- Presenza di 13 registri general purpose a 32 bit indicati con R0-R12, divisi in *low* (R0-R7), accessibili da tutte le istruzioni che specificano un registro general purpose, e *high* (R8-R12), accessibili solo da certe istruzioni; e dei seguenti registri:
  - Link register (LR, R14);
  - Program counter (PC, R15);
  - Fino a 2 (nel caso in cui venga supportato un sistema operativo) Stack pointer (SP, R13);
  - Program Status register (xPSR).
- Architettura pipeline a 3 stadi;
- Rappresentazione dei dati in formato Big endian o little endian (opzione di configurazione, le istruzioni sono sempre little endian); tipi di dati supportati: word (32 bit), half word e byte;
- Bus indirizzi a 32 bit;
- Presenza di un controllore di interrupts integrato (che permette bassa latenza per gli interrupt), Nested Vectored Interrupt Controller (NVIC), con le seguenti caratteristiche:
  - Possibilità di configurare fino a 32 interruzioni;
  - 4 livelli di priorità;
  - Lo stato del processore viene automaticamente salvato, e in seguito ripristinato, al verificarsi di un interrupt.
- 2 stati operativi:
  - Thumb state: stato in cui il processore si trova durante la normale esecuzione del codice;
  - Debug state: stato in cui il processore si trova durante il debug.

[9] ARM Holdings è una società di alta tecnologia con sede a Cambridge, Regno Unito. La società è nota principalmente per la sua linea di processori sebbene sviluppi e venda system-on-a-chip, piattaforme hardware, infrastrutture e software sotto i marchi RealView e KEIL.

- Supporta 2 modalità operative:
  - Thread mode: il processore entra in questa modalità al reset e può rienetrarci come risultato di ritorno di un'eccezione;
  - Handler mode: il processore entra in questà modalità a seguito di un'eccezione.
- Unità di debug opzionale con le seguenti caratteristiche:
  - L'unità può accedere a tutta la memoria e ai registri del sistema;
  - Debug Access Port (DAP);
  - BreakPoint Unit (BPU) per implementare breakpoints;
  - Data Watchpoint (DW) unit per implementare watchpoints.
- Memorie tightly coupled per dati e istruzioni configurabili (0K - 1024K);
- Supporta il bus AMBA;
- Moltiplicatore hardware a 32 bit: possibilità di scegliere tra un moltiplicatore standard (3 cicli) e uno più piccolo, con prestazioni più basse (33 cicli).

In figura 3.11 è riportato il diagramma a blocchi del Cortex-M1 comprensivo dell'unità di Debug.

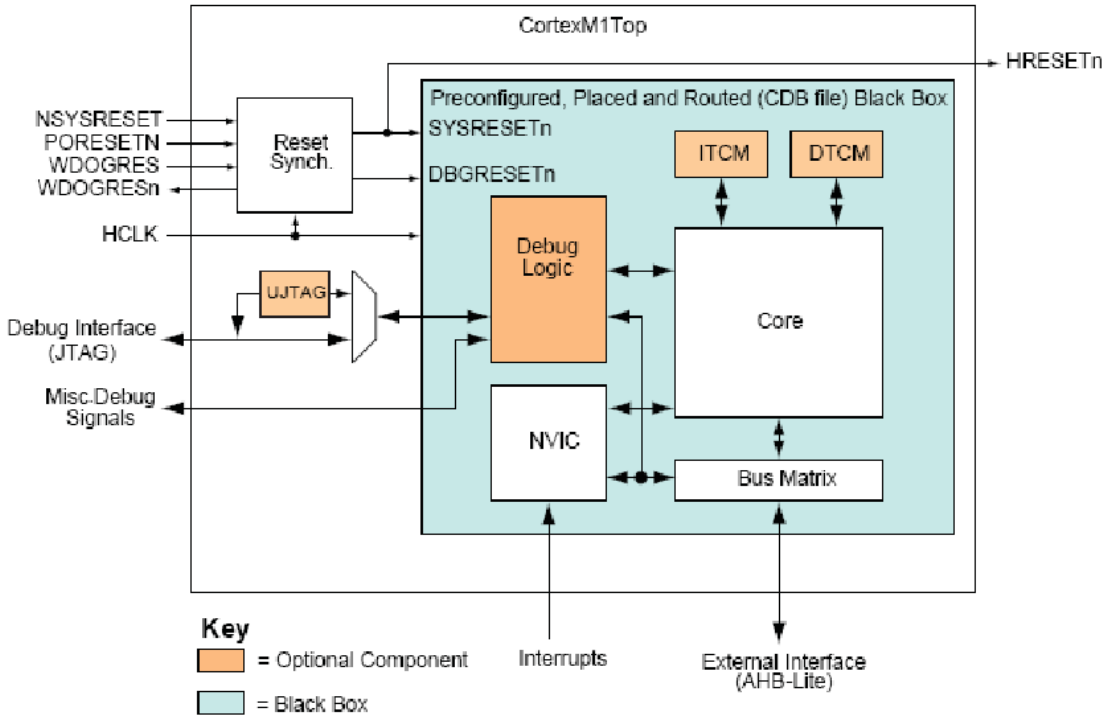


Figura 3.11: Schema a blocchi del Cortex-M1.

3.4.1 Set d'istruzioni

Come accennato in precedenza il set d'istruzioni del Cortex-M1 è un sottoinsieme del Thumb-2.
Il set di istruzioni thumb è un subset del set di istruzioni ARM. Le istruzioni thumb hanno una dimensione che è pari alla metà della dimensione delle istruzioni ARM (16 bit invece che 32). Il risultato che ne deriva è che normalmente può essere ottenuta una maggiore densità di codice (minor fabbisogno di memoria) utilizzando il set Thumb invece che il set ARM. Utilizzando il set di istruzioni Thumb si hanno due limitazioni:

- Per la realizzazione della stessa funzione, il codice thumb usa più istruzioni del codice ARM e quindi il codice ARM è meglio indicato per massimizzare le prestazioni di un codice con criticità di tempi di esecuzione.

- Il set di istruzioni Thumb non include alcune istruzioni necessarie per la gestione delle eccezioni.

La tecnologia Thumb-2 estende le limitate istruzioni a 16 bit con delle addizionali istruzioni a 32 bit per fornire maggior potenza al processore. La tecnologia Thumb 2 fornisce codice con densità (e quindi occupazione di banda) simile a quello del codice Thumb ma con prestazioni più vicine a quello ARM a 32 bit.
Thumb-2 inoltre estende le istruzioni ARM e Thumb con delle nuove istruzioni che permettono la manipolazione dei singoli bit, l'esecuzione condizionata e la gestione di tabelle con salti. Per maggiori informazioni vedi [9].

3.4.2 Interfacce bus

Il processore contiene 2 interfacce bus:
-Interfaccia esterna;
-Interfacce di memoria.
Il Cortex-M1 è dotato inoltre di un bus interno, il private peripheral bus (PPB), per facilitare la comunicazione tra il processore e il NVIC e i blocchi di debug.

Interfaccia esterna: l'interfaccia esterna è di tipo AMBA AHB-Lite[10], l'accesso alle periferiche (AHB) avviene attraverso questo bus. Per maggiori informazioni vedi appendice A e [10].

Interfacce di memoria: il processore possiede 2 interfacce di memoria per accedere alle memorie TCM (ITCM per le istruzioni e DTCM per i dati) nel caso queste abbiamo dimensione diversa da 0.

[10] AHB-lite è un sottoinsieme del protocollo AHB.

3.4.3 Tool di sviluppo

Cortex-M1, a differenza dei soft processor analizzati in precedenza, può essere implementato su FPGA di diverse case produttrici, quindi i tool di sviluppo saranno diversi a seconda delle FPGA che vengono utilizzate. In tabella 3.6 vengono riportati alcuni esempi di FPGA su cui è possibile implementare il processore e i relativi software di sviluppo. Da notare che anche questo soft processore (grazie ai tool messi a disposizioni) può essere direttamente programmato nei linguaggi C/C++. Per maggiori informazioni sui tool di sviluppo per i dispositivi Actel si veda [10] e [49].

FPGA Device Compatibility	**Implementation Tool Compatibility**
Actel ProASIC3L & ProASIC3/E	Actel Libero
Altera Cyclone-II	Altera Quartus-II
Altera Stratix-II	Synopsys Synplify Pro
Xilinx Spartan-3	Mentor Precision
Xilinx Virtex-4	Xilinx ISE

Tabella 3.6: Esempi tool di sviluppo.

3.5 Leon 3

Il LEON3 è un progetto open source di un microprocessore RISC compatibile con l'architettura SPARC V8 sviluppato dalla Gaisler Research[51]. Il microprocessore è molto versatile ed è particolarmente adatto per il progetto di SOC (System On a Chip). La sua versatilità risiede nella caratteristica di poter essere ottimizzato su diversi aspetti, come ad esempio, le performance, il consumo di potenza, I/O throughput, occupazione di risorse ed il costo. Il microprocessore si interfaccia con il bus AMBA-2.0 AHB e supporta l'IP *plug&play* descritta nella libreria GRLIB (Gaisler Research ip LIBrary). Il microprocessore può essere efficientemente implementato nelle tecnologie FPGA e ASIC ed utilizza delle celle standard di RAM sincrona per implementare sia la memoria cache che i register file. Per promuovere l'architettura SPARC e semplicare la progettazione vengono forniti le descrizioni hardware in VHDL sia del processore LEON3 che della libreria IP sotto licenza GNU (General Public License).

3.5.1 Caratteristiche principali

Il LEON3 è un microprocessore a 32 bit conforme allo standard IEEE-1754 (SPARC V8). E' stato progettato per applicazioni embedded e le sue principali caratteristiche sono:

- Pipeline a 7 stadi;
- Architettura di memoria di tipo Harvard;

- Presenza di cache separate per dati e istruzioni (entrambe possono essere configurate in 1-4 set, 1-256 kByte/set, 16-32 byte per linea);
- Frequenza di lavoro su FPGA fino a 125MHz;
- Supporta il bus AMBA;
- Moltiplicatore e divisore hardware;
- 2 modalità operative: supervisore e utente;
- Unità per il supporto del debug on-chip.

L'architettura del LEON3, figura 3.12, è composta da diversi macro blocchi alcuni dei quali vengono brevemente descritti[17].

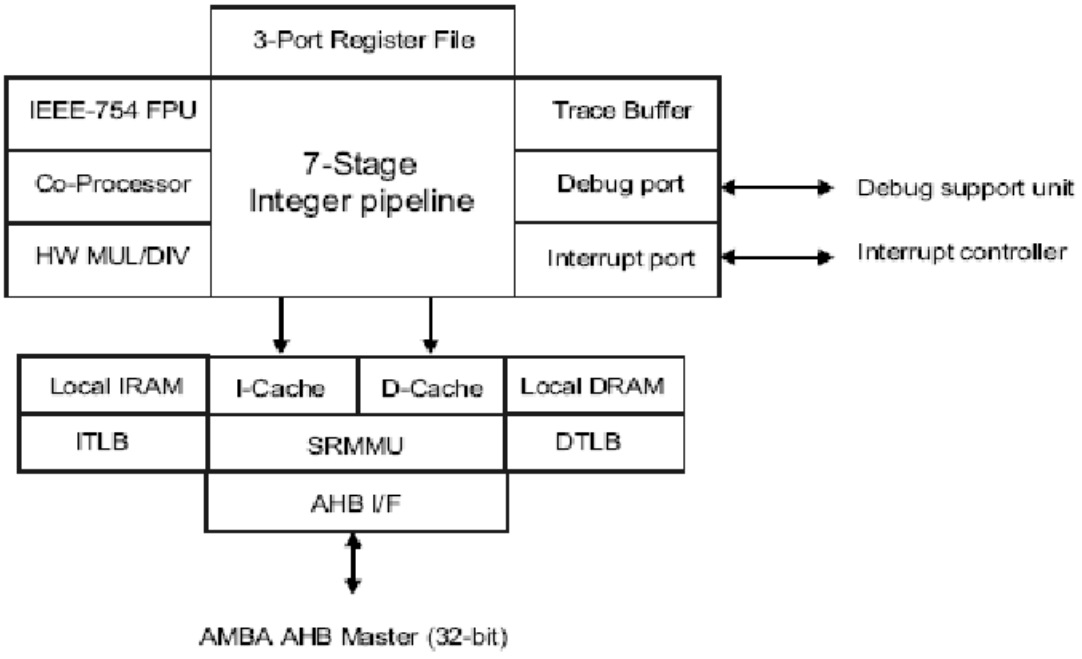


Figura 3.12: Schema a blocchi del Leon3.

Integer Unit L'integer unit del LEON3 implementa in pieno lo standard SPARCV8 comprese le istruzioni per la moltiplicazione e divisione hardware. Il numero di finestre della *register windows*[11] è configurabile, entro i limiti dello standard SPARC

[11]Tecnica utilizzata per incrementare le prestazioni di operazioni particolarmente comuni come le chiamate alle subroutine (o procedure). La register window fornisce un insieme di registri indipendenti ad ogni procedura. In sostanza ogni procedura vede solo un sottoinsieme dei registri totali del processore e quando c'è una chiamata a una subroutine il processore commuta ad un insieme di registri liberi e quindi il cambio di contesto è praticamente istantaneo dato che non richiede accessi alla memoria. Quando la chiamata alla subroutine termina vengono resi visibili i registri della procedura principale e i registri utilizzati dalla subroutine vengono segnalati come liberi. Questa tecnica può essere utilizzata più volte in modo da gestire subroutine che chiamano altre subroutine in modo annidato[52].

(2-32), e di default è pari a 8. Il pipeline è su 7 livelli (Instruction Fetch(FE), Decode(DE), Register access(RA), Execute(EX), Memory(ME), Exception(XC) e Write(WR))con interfaccia separata per le istruzioni e i dati (architettura Harvard).

Floating point unit e co-processor L'integer unit del LEON3 fornisce un'interfaccia per la floating-point unit (FPU) e per un eventuale co-processore. Sono disponibili due FPU, una sviluppata dalla Gaisler Research (GRFPU) e l'altra sviluppata dalla Sun Microsystems (Meiko FPU core). La floating-point unit e l'eventuale co-processore elaborano i dati in parallelo all'integer unit finchè non vi sono dei conflitti nell'utilizzo di risorse o dati in comune.

Memory management unit Può essere opzionalmente abilitata una Memory Management Unit (SRMMU) compatibile con lo standard SPARC V8. La SRMMU provvede a tradurre gli indirizzi, multipli di 32 bit, della memoria vir- tuale nei 36 bit della memoria fisica.

Interrupt interface Il LEON3 supporta fino a un totale di 15 interrupt asincroni. L'interfaccia di interrupt fornisce le funzionalità per la generazione e la gestione degli interrupt.

Multi-processor support Il LEON3 è progettato per essere utilizzato in sistemi multiprocessore. Ciascun processore ha un proprio numero identificativo che permette di indicizzare i processori. I meccanismi di write-through delle memorie cache e di snooping garantiscono la coerenza dei dati condivisi.

3.5.2 La libreria GRLIB IP

La libreria GRLIB IP è una collezione di IP (Intelligent Peripheral) espressamente sviluppate per le architetture SOC (System On Chip). Le IP sono sviluppate per condividere un bus comune e utilizzare un metodo coerente per la simulazione e la sintesi. La libreria è stata realizzata per essere pienamente compatibile con differenti strumenti CAD e differenti tecnologie implementative. Un'unico metodo plug&play viene utilizzato per configurare e connettere le IP in maniera automatica. La GRLIB è un progetto open source ed è distribuita sotto licenza GNU GPL. Questo significa che tutti i componenti della libreria, sviluppati secondo questa licenza, sono distribuiti liberamente e vengono fornite le descrizioni hardware.

Organizzazione della libreria La GRLIB è organizzata in differenti librerie VHDL ciascuna delle quali individua un IP tramite un nome univoco. L'uso di differenti librerie previene il conflitto di nome fra differenti IP e permette di nascondere all'utente finale i dettagli implementativi non essenziali. Ciascuna libreria VHDL contiene parecchi package in cui vi sono le dichiarazioni per l'esportazione degli IP e la loro interfaccia. Gli script per la simulazione e la sintesi sono generati automaticamente attraverso dei makefile globali. L'aggiunta e la rimozione delle

librerie e dei package possono essere fatte senza modificare alcun file globale. Alcune librerie globali provvedono a definire le strutture dati e le funzioni in comune. GRLIB fornisce gli script generator per i simulatori Modelsim, Ncsim e GHDL, e per i sintetizzatori Synopsys, Synplify, Cadence, Mentor, Actel, Altera, Lattice e Xilinx. Il supporto ad altri strumenti CAD può essere facilmente implementato. La libreria GRLIB è sviluppata per essere bus-centric, ovvero è una libreria in cui la maggior parte degli IP sono connessi a un bus comune. Il bus scelto per connettere i vari IP è l'AMBA-2.0 AHB/APB poichè è largamente utilizzato (vedi i processori della ARM), può essere utilizzato liberamente senza restrizioni d'utilizzo ed è disponibile a corredo un'ampia documentazione. La figura 3.13 mostra un'esempio di un sistema basato sul microprocessore LEON3 progettato con la libreria GRLIB. L'utilizzo della libreria GRLIB permette di creare un microcontrollore completo di

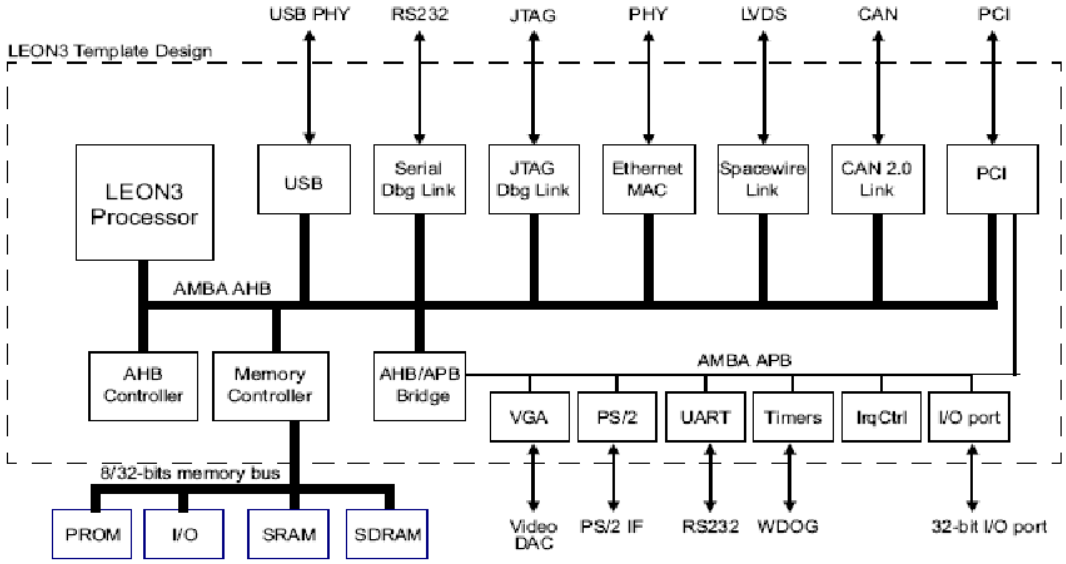


Figura 3.13: LEON3 template design.

tutte le periferiche necessarie. Vengono messe a disposizione componenti come il controllore AMBA AHB/APB, il microprocessore LEON3 SPARC, l'unita in virgola mobile IEEE-754, l'AHB/ABH bridge e molti altri.
GRLIB introduce nell'architettura del sistema delle sostanziose novità per quanto riguarda la decodifica di indirizzo distribuita, la gestione degli interrupts e la funzionalità di plug&play. Per plug&play si intende la capacità di rilevare la configurazione hardware attraverso il software. Tale possibilità permette all'applicazione software o al sistema operativo di configurarsi automaticamente al fine di adattarsi alle caratteristiche hardware. Ciò semplifica notevolmente lo sviluppo delle applicazioni software poichè diventano indipendenti dalla piattaforma hardware. La libreria GRLIB è stata sviluppata per essere facilmente implementata su tecnologia ASIC o FPGA. La portabilità prevede il supporto per componenti come le single-port RAM, le two-port RAM, le dual-port RAM, le single-port RAM, il generatore di clock e i pad. La portabilità è stata implementata per mezzo di componenti virtuali il cui codice VHDL permette di selezionare la tecnologia desidera-

ta. Nell'architettura del componente il comando VHDL `generate` è utilizzato per istanziare i macro blocchi della tecnologia selezionata.

Per maggiori informazioni sull'implementazione di LEON3 si veda [16].

3.6 OpenSPARC T1

OpenSPARC T1, prodotto dalla Sun Microsystems, rappresenta la versione open source (in Verilog) del precedente UltraSPARC T1. Si tratta di un processore *multi core*[12] in grado di garantire il *multithreading*[13] e che implementa l'architettura SPARC V9 a 64 bit.

3.6.1 Caratteristiche principali

La figura 3.14 mostra lo schema a blocchi dell'OpenSPARC T1, le cui caratteristiche principali sono[18]:

- 8 SPARC V9 CPU cores, con 4 threads per core, per un totale 32 threads;
- Disponibilità di memoria cache on-chip:
  - 16 KBytes di memoria cache istruzioni, di livello 1, per core;
  - 8 KBytes di memoria cache dati, di livello 1, per core;
  - 3 MBytes di memoria cache secondaria suddividi in 4 banchi di tipo 12-Way Set-Associative;
- 4 controllori DDR-II SDRAM[14]
- Unità floating point (FPU) condivisa da tutti i core;

[12]Il termine multi core si usa per descrivere una CPU composta da due o più core, ovvero da più nuclei di processori fisici montati sullo stesso package.

[13]Il multithreading in informatica indica il supporto hardware da parte di un processore di eseguire più thread. Questa tecnica si distingue da quella alla base dei sistemi multiprocessore per il fatto che i singoli thread condividono lo stesso spazio d'indirizzamento, la stessa cache e lo stesso translation lookaside buffer. Il multithreading migliora le prestazioni dei programmi solamente quando questi sono stati sviluppati suddividendo il carico di lavoro su più thread che possono essere eseguiti in parallelo. I sistemi multiprocessore sono dotati di più unità di calcolo indipendenti, un sistema multithread invece è dotato di una singola unità di calcolo che si cerca di utilizzare al meglio eseguendo più thread nella stessa unità di calcolo. Le tecniche sono complementari, a volte i sistemi multiprocessore implementano anche il multithreading per migliorare le prestazioni complessive del sistema.

[14]La DDR SDRAM, acronimo di Double Data Rate Synchronous Dynamic Random Access Memory (in italiano memoria dinamica ad accesso casuale sincrona a doppia velocità), è un tipo di memoria RAM su circuiti integrati usati nei computer. Ha una larghezza di banda maggiore rispetto alla SDRAM poiché trasmette i dati sia sul fronte di salita che sul fronte di discesa del ciclo di clock. Questa tecnica consente di raddoppiare la velocità di trasferimento senza aumentare la frequenza del bus di memoria. Quindi un sistema DDR ha un clock effettivo doppio rispetto a quello di uno SDRAM.

- Interfaccia esterna:
  - J-Bus interface (JBI) per i dispositivi di I/O;
  - Serial system interface (SSI) per la boot PROM[15];
- Portabilità su dispositivi di diverse case produttrici.

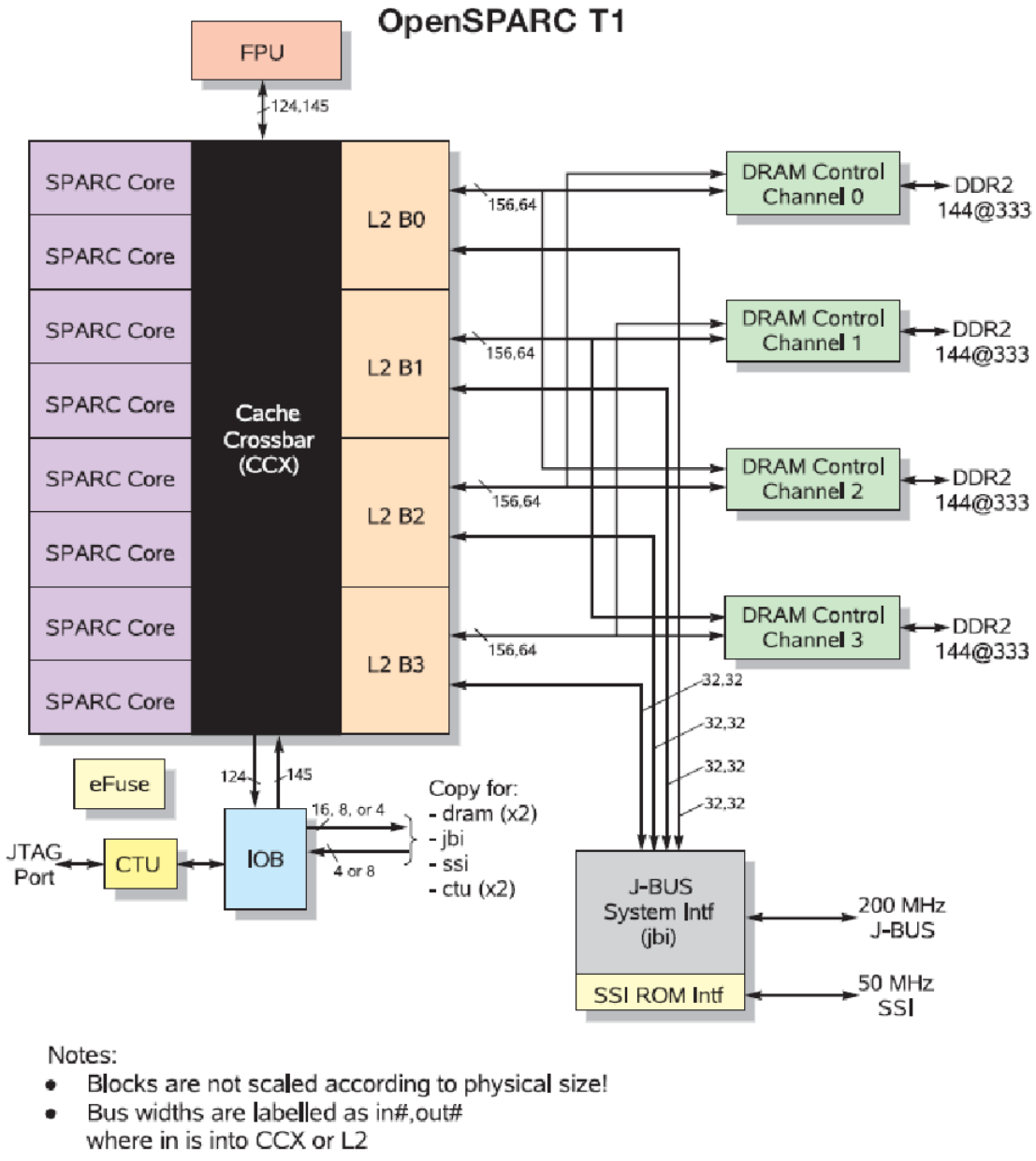


Figura 3.14: Diagramma a blocchi dell'OpenSPARC T1.

[15]Supporto di memoria specifico contenente le informazioni e i programmi per eseguire la procedura di avvio.

3.6.2 SPARC core

Ciascun core è caratterizzato da un hardware in grado di garantire l'esecuzione di 4 threads, che condividono la memoria cache, dati e istruzioni, e i TLBs[16]. Ciascuna cache per le istruzioni ha una dimensione di 16KBytes con linee di 32Bytes; mentre le caches per i dati, di tipo write through, sono caratterizzate da una dimensione di 8KBytes con linee di 16Bytes.
Ogni core è caratterizzato da un'architettura pipeline a 6 stadi (Fetch, Thread Selection, Decode, Execute, Memory e Write Back) ed è costituito dalle seguenti unità:

1. Instruction fetch unit (IFU): realizza il fetch, il thread selection, e il decode dell'istruzione;
2. Execution unit (EXU): esegue l'istruzione;
3. Load/store unit (LSU): esegue le funzioni degli statdi memory e write back (attiva la memoria e scrive il risultato nel registro opportuno);
4. Trap logic unit (TLU): gestisce le eccezioni;
5. Stream processing unit (SPU);
6. Memory management unit (MMU);
7. Floating-point frontend unit (FFU): intefaccia verso la FPU.

3.6.3 CPU-Cache Crossbar

La CPU-Cache Crossbar (CCX) è una matrice di switch che realizza il collegamente tra gli 8 core, i 4 banchi di cache di secondo livello, la FPU e i bridge di I/O.

3.6.4 Floating-Point Unit

Una sola FPU è condivisa da tutti gli 8 core del processore. La FPU è sufficiente per la maggior parte delle applicazioni commerciali , in cui meno dell'1% delle istruzioni sono operazioni floating point.

3.6.5 Interfaccia esterna

J-Bus Interface Il J-Bus Interface è il blocco funzionale che realizza l'interfacciamento con il j-bus (128 bit), permettendo il ytrasferimento di dati tra il processore e il sottosistema di I/O.

[16]Il Translation Lookaside Buffer (TLB) è un buffer, cioè una memoria tampone (o, nelle implementazioni più sofisticate, una cache nella CPU), che l'MMU (Memory Management Unit) usa per velocizzare la traduzione degli Indirizzi Virtuali.

Serial System Interface Il processore OpenSPARC T1 è dotato di una serial system interface (SSI) che permette il collegamento con un ASIC esterno che a sua volta si occupa del collegamento con la boot PROM.

3.7 PicoBLAZE

PicoBlaze è un soft-processor (realizzato in VHDL), disponibile gratuitamente, prodotto da Xilinx ed ottimizzato per le FPGA della famiglia *Spartan-3*; il processore è comunque implementabile anche nelle famiglie *Virtex-5, Spartan-6, and Virtex-6*. Si tratta di un processore estremamente flessibile e compatto, caratterizzato da bassi costi di sviluppo.

3.7.1 Caratteristiche principali

PicoBlaze, il cui diagramma a blocchi è mostrato in figura 3.15, è un soft processor di tipo RISC a 8 bit le cui caratteristiche principali sono[26]:

- Presenza di 16 registri general purpose a 8 bit;
- Memoria di programma (realizzata su una singola block ram) contenente fino a 1024 istruzioni (18 bit), automaticamente caricate durante la configurazione della FPGA;
- Arithmetic Logic Unit (ALU) a 8 bit con flags di CARRY e ZERO;
- RAM scratchpad[17] interna di 64 Byte;
- 256 porte di input e di output per facilitare miglioramenti ed espansione;
- CALL/RETURN stack automatico di 31 locazioni;
- Prestazioni prevedibili, sempre 2 cicli di clock per istruzione, fino a 200 MHz o 100 MIPS in una FPGA Virtex-II Pro;
- Bassa latenza d'interupt: nel caso peggiore 5 cicli di clock;
- Bassa occupazione d'area, solo 96 slices nell'implementazione su Spartan-3;
- Strumenti di sviluppo software e hardware completi.

[17]RAM scratchpad, nota anche solamente come scatchpad o Local Store è un termine informatico che identifica una memoria ad alta velocità utilizzata generalmente per memorizzare dati utilizzati dal microprocessore con frequenza. Può essere considerata simile alla cache di un processore con la differenza che mentre la cache è invisibile al programmatore la scatchpad è indirizzata direttamente dal processore e quindi visibile al programmatore. La scatchpad viene utilizzata per semplificare la gestione della coerenza delle cache nei sistemi multiprocessore.

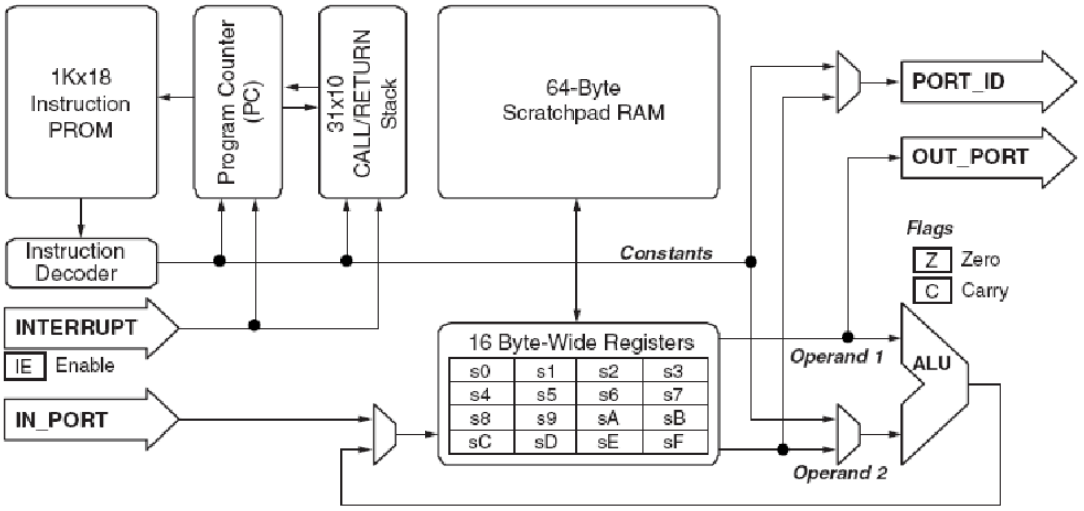


Figura 3.15: Diagramma a blocchi del PicoBLAZE.

3.7.2 Registri

Il PicoBlaze contiene 16 registri general purpose a 8 bit indicati con s0-sF (è possibile tuttavia rinominare tali registri usando delle direttive assembler).
Nessun registro è riservato per compiti speciali o ha priorità rispetto agli altri.

3.7.3 Arithmetic Logic Unit (ALU)

L'unità aritmetico logica si occupa di tutte le operazioni in cui sono coinvolti dei calcoli, tra cui le più importanti risultano:

- Operazioni aritmetiche di base, come addizione e sottrazione;
- Operazioni bitwise (bit a bit), come AND, OR e XOR;
- Operazioni di confornto;
- Operazioni di scorrimento e rotazione.

Tutte le operazioni sono eseguite utilizzando un operando specificato in un opportuno registro (sX). Il risultato viene scritto nel medesimo registro. Nel caso in cui un'istruzione richiedesse un secondo operando, questo può essere fornito da un secondo registro (sY) o da un campo immediato di 8 bit (kk).

3.7.4 64-Byte Scratchpad RAM

Il PicoBlaze è provvisto di una scratchpad RAM di 64 Byte alla quale è possibile accedere usando le istruzioni di STORE e di FETCH.
L'istruzione di STORE scrive il contenuto di uno dei 16 registri in una delle 64 locazioni della scratchpad. L'istruzione di FETCH, invece, legge il contenuto di

una delle 64 locazioni e lo scrive in uno dei 16 registri.
Questo permette di poter maneggiare un numero maggiore di variabili e di riservare lo spazio di I/O per effettivi segnali di input e output.
I 6 bit dell'indirizzo rappresentante la locazione all'interno della scratchpad RAM possono essere specificati nell'istruzione in 2 diversi modi:

- Direttamente attraverso un campo immediato;
- Indirettamente, attraverso il contenuto di uno dei 16 registri.

Vengono utilizzati solo i 6 bit meno significativi e l'indirizzo non deve eccedere il range 00-3F.

3.7.5 Input/Output

Le porte di I/O permettono il collegamento del processore con periferiche personalizzate o con altre FPGA. Il PicoBlaze supporta fino a 256 porte di input e 256 di output.
Durante un'operazione di input il processore legge il contenuto di una porta di input (specificata nell'istruzione) e scrive il valore letto nel registro specificato dall'istruzione stessa. Viceversa durante un'operazione di output il processore scrive il contenuto di un apposito registro nella porta di output specificata dall'istruzione stessa.

3.7.6 CALL/RETURN Stack

IL CALL/RETURN Stack memorizza fino a 31 indirizzi permettendo chiamate nidificate fino a 31 livelli di profondità. Poichè lo stack è utilizzato anche in seguito ad un interrupts, almeno uno di questi livelli dovrebbe essere riservato quando gli interrupts sono abilitati.
Lo stack viene implementato come un beffer di memoria circolare; in questo modo quando risulta pieno viene sovrascritto il dato più vecchio. Di conseguenza non esistono istruzioni per controllare lo stack o lo stack pointer.

3.7.7 Tools di sviluppo

PicoBLAZE è fornito sotto forma di un file VHDL chiamato `KCPSM3.vhd`. Il codice è adatto alla sintesi e alla simulazione ed è stato sviluppato e testato utilizzando il *Xilinx Synthesis Tool* (XST) per la sintesi e *ModelSim* per la simulazione. Il modulo KCPSM3 contiene l'ALU, il register file, la ram, etc.; l'unico modulo non incluso è la memoria di programma (la quale viene generata dall'assembler).

Esistono 3 ambienti di sviluppo software differenti per realizzare codice per PicoBLAZE, come riportato nella tabella 3.7. Due di questi ambienti sono offerti da

Xilinx ed includono un assemblatore command-line. Il Mediatronix pBlazIDE, invece, è un ambiente di sviluppo grafico contente un assemblatore ed un instruction-set simulator (ISS), capace di emulare l'istruction set architecture (ISA) del microprocessore.

	Xilinx KCPSM3	**Mediatronix pBlazIDE**	**Xilinx System Generator**
Platform Support	Windows	Windows 98, Windows 2000, Windows NT, Windows ME, Windows XP	Windows 2000, Windows XP
Assembler	Command-line in DOS window	Graphical	Command-line within System Generator
Instruction Syntax	KCPSM3	PBlazIDE	KCPSM3
Instruction Set Simulator	Facilities provided for VHDL simulation	Graphical/ Interactive	Graphical/ Interactive
Simulator Breakpoints	N/A	Yes	Yes
Register Viewer	N/A	Yes	Yes
Memory Viewer	N/A	Yes	Yes

Tabella 3.7: Ambienti di sviluppo per PicoBLAZE.

E' inoltre disponibile un compilatore C per PicoBLAZE; per maggiori informazioni si veda [53].

3.7.8 PacoBLAZE

PacoBlaze è un'implementazione (comportamentale e sintetizzabile) in Verilog, realizzata da Pablo Bleyer, di PicoBLAZE disponibile sotto licenza BSD.
Le performance del processore sono simili a quelle del PicoBLAZE ma dipendono dalle specifiche del progetto; inoltre le dimensioni, trattandosi di un modello comportamentale, sono maggiori rispetto all'originale del 30-50%. Il vantaggio principale di PacoBLAZE è quello di rendere più facile la modifica e l'espansione del processore per adattarlo alla specifica applicazione. Per maggiori informazioni si veda [55].

3.8 eSi-RISC

ESi-RISC è un'architettura di microprocessori per sistemi embedded altamente configurabile realizzata da *Ensilica*. E' disponibile in tre diverse implementazioni:

eSi-1600 (16 bit), eSi-3200 ed eSi-3250 (entrambe a 32 bit). Tutti e tre i processori sono forniti, gratuitamente, in forma di soft IP core.
Le caratteristiche principali di tale architettura sono:

- Architettura RISC configurabile a 16 o 32 bit;
- Pipeline a 5 stadi;
- Istruzioni sia a 16 che a 32 bit che permettono alta densità di codice;
- Set d'istruzioni configurabile per supportare aritmetica intera, in virgola mobile e in virgola fissa;
- Architettura di memoria configurabile: von Neumann o Harvard;
- Presenza di un numero configurabile di registri general purpose: 8, 16 o 32;
- Presenza di un numero configurabile di *vector registers*: 0, 8, 16 o 32;
- Da 0 fino ad 8 accumulatori;
- Supporta fino a 32 sorgenti d'interrupts;
- Cache (dimensione e associatività configurabili) e MMU opzionali;
- Architettura bus AMBA AXI/APB;
- Due modalità operative: user e supervisor mode;
- Strumenti di sviluppo software e hardware completi e possibilità di debug (JTAG).

La figura 3.16 mostra lo schema a blocchi dell'architettura eSi-RISC.

3.8.1 eSi-1600

L'esi-1600 è un processore a 16 bit a basso costo e a basso consumo. Esso offre prestazioni simili ai processori a 32 bit e un costo comparabile con quello dei processori a 8 bit. Viene particolarmente utilizzato nella produzione di sensori intelligenti, dispositivi medici e reti wireless. Le caratteristiche principali di tale processore sono disponibili nel rispettivo manuale[11].

3.8.2 eSi-3200

L'esi-3200 è un processore a 32 bit a basso costo e a basso consumo ideale per l'integrazione in progetti su FPGA con memorie on-chip. Risulta particolarmente adatto per applicazioni di controllo integrato. Per maggiori informazioni si veda [12].

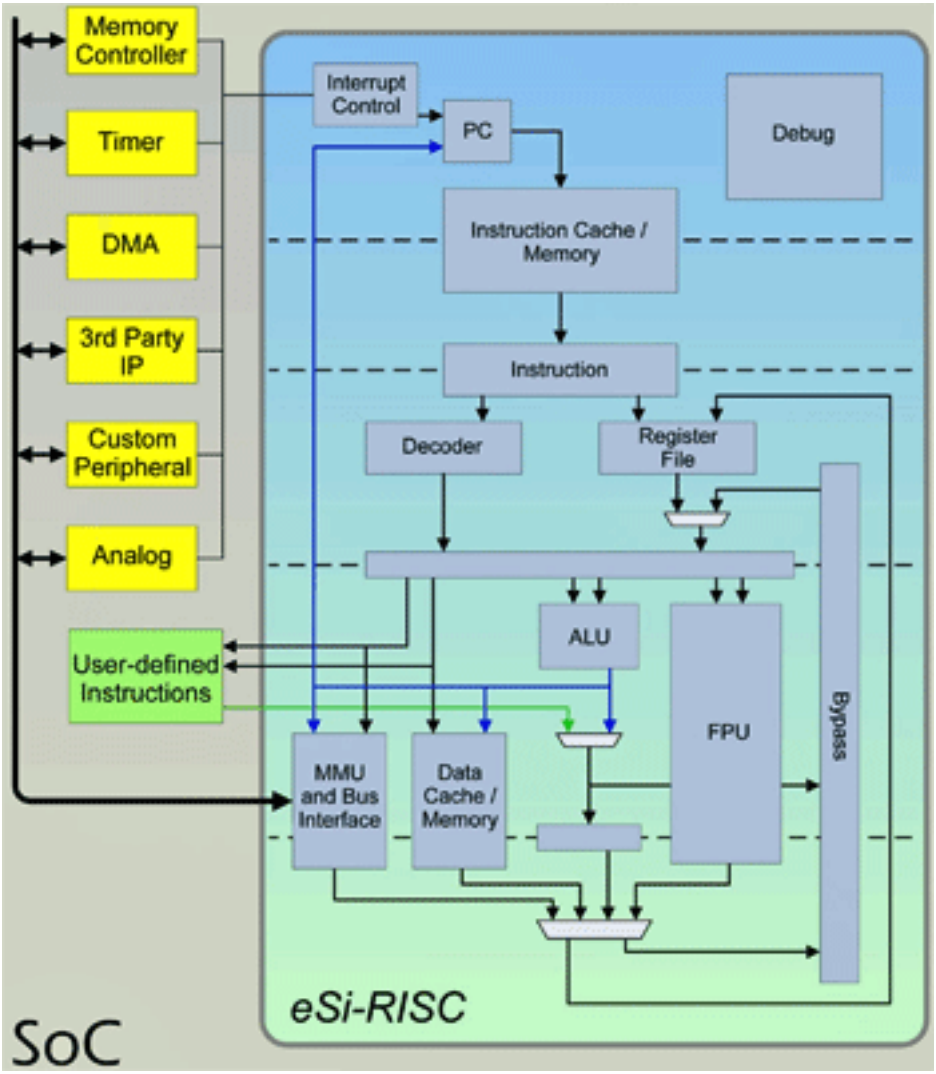


Figura 3.16: Diagramma a blocchi dell'architettura eSi-RISC.

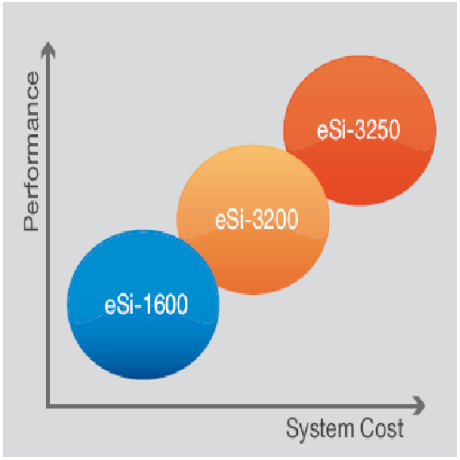


Figura 3.17: Confronto tra le prestazioni dei processori eSi-RISC.

3.8.3 eSi-3250

L'esi-3250 è un processore a 32 bit con alte prestazioni, ideale per l'integrazione in progetti su FPGA con memorie off-chip. E' adatto per un gran numero di applicazioni come ad esempio l'esecuzione di un sistema operativo complesso come linux. Le caratteristiche principali dell'eSi-3250 sono disponibili nel manuale del processore[13].

3.8.4 Tool di sviluppo

E' disponibile un ambiente di sviluppo per la creazione dell'hardware e del software relativo a eSi-Risc: l'eSi-RISC Development Suite; attualmente è disponibile la versione 2.1. Tale ambiente permette la creazione di progetti basati su eSi-RISC, e di programmare il processore direttemente in linguaggi ad alto livello come il C e il C++.

3.9 LatticeMico32

LatticeMico32 è un soft processor altamente configurabile, realizzato dalla Lattice Semiconductor e disponibile gratuitamente. Il processore può essere implementato sulle FPGAs di Xilinx e Altera oltre, ovviamente, su quelle di Lattice (da notare che il processore può essere implementato anche negli ASICs).

3.9.1 Caratteristiche principali

Si tratta di un processore RISC a 32 bit le cui caratteristiche principali sono:

- Istruzioni a 32 bit;
- Architettura di memoria harward;
- 32 Registri general purpose e 10 registri di stato e di controllo a 32 bit;
- Fino a 32 sorgenti d'interrupts esterne;
- Pipeline a 6 stadi: Adress, Fetch, Decode, Execute, Memory e Writeback;
- Supporta il bus WISHBONE: 2 interfacce per la memoria (dati e istruzioni);
- Rappresentazione dei dati in formato Big endian; tipi di dati supportati: word (32 bit), half word e byte;
- I/O Mappato in memoria;
- Cache opzionale (dimensioni e associatività configurabili);
- Strumenti di sviluppo software e hardware completi e possibilità di debug.

Il processore, il cui schema a blocchi è riportato in figura 3.18, è disponibile in 3 diverse configurazioni:

- Basic
  - Moltiplicatore assente;
  - Cache assente;
  - Shifter multiciclo.
- Standard
  - Dotato di moltiplicatore;
  - 8K I-Cache, D-Cache assente;
  - Pipelined Shifter.
- Full
  - Dotato di moltiplicatore;
  - 8K I-Cache,8K D-Cache;
  - Pipelined Shifter.

Per lo sviluppo di veri e propri sistemi embedded al processore possono essere collegate, utilizzando il bus WISHBONE, diverse periferiche. Tra queste le più importanti sono:

- Memory controllers
  - Asynchronous SRAM
  - Double data rate (DDR)
  - On-chip
- Input/output (I/O) ports
  - 32-bit timer
  - Direct memory access (DMA) controller
  - General-purpose I/O (GPIO)
  - Serial peripheral interface (SPI)
  - Universal asynchronous receiver transmitter (UART)

La figura 3.19 mostra un sistema embedded basato su LatticeMico32.

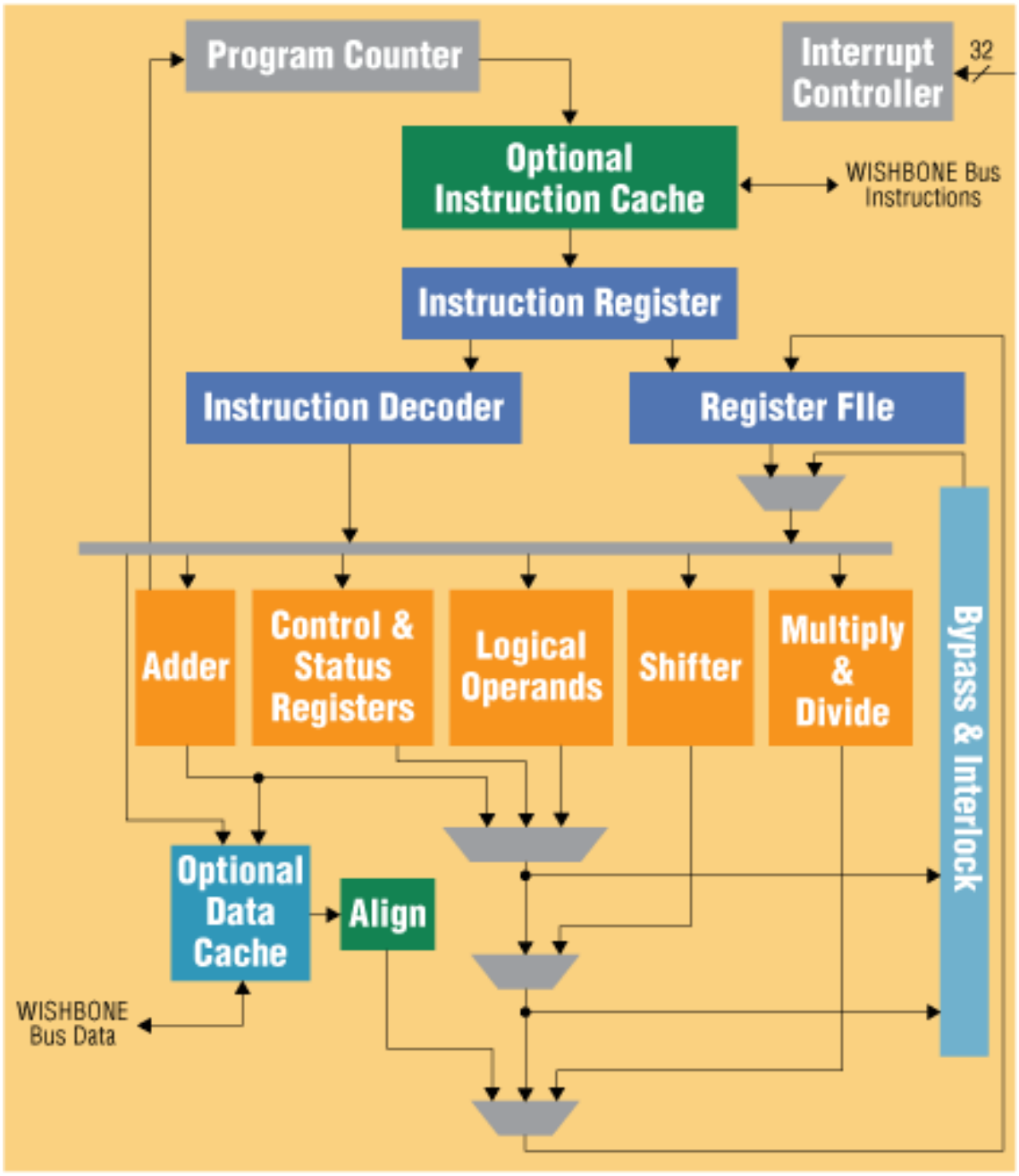


Figura 3.18: Diagramma a blocchi del LatticeMico32.

3.9.2 Registri

Generale purpose Come accennato in precedenza il LatticeMico32 è dotato di 32 registri genral purpose, indicati con r0-r31:

- Il registro r0 deve contenere sempre il valore 0;
- I registri da r1 a r28 sono propriamente general purpose e possono esssere utilizzati come operando sorgente o destinazione in qualsiasi istruzione;

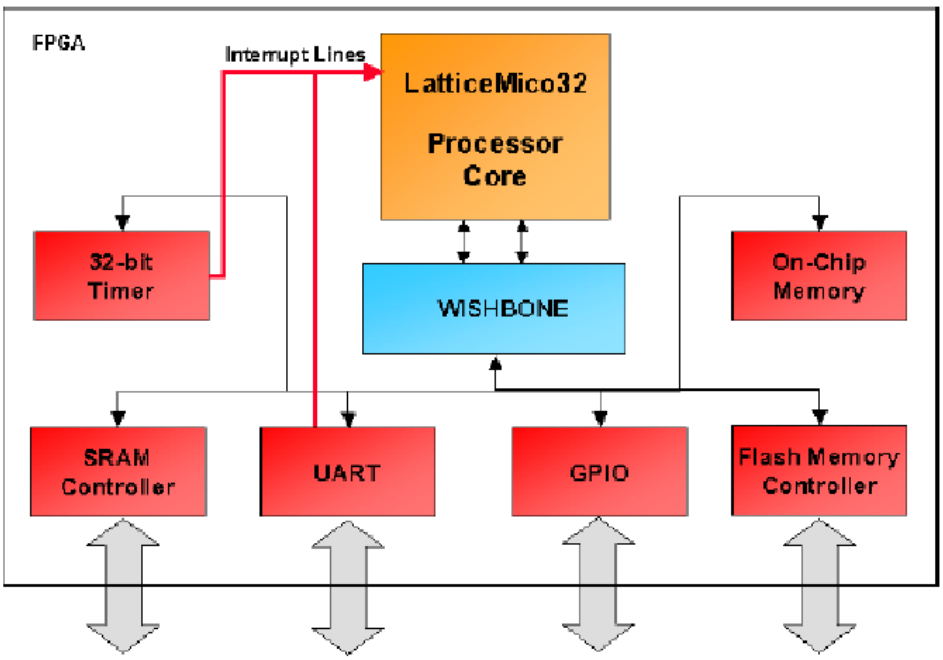


Figura 3.19: Sistema embedded basato su LatticeMico32.

- Il registro r29 (ra) è usato per salvare l'indirizzo di ritorno in seguito ad un'istruzione di call;
- Il registro r30 (ea) è usato per salvare il contenuto del program counter in seguito al verificarsi di un'eccezione;
- Il registro r31 (ba) è usato per salvare il contenuto del program counter in seguito ad un breakpoint o ad un watchpoint.

Dopo il reset il valore memorizzato in questi registri è indefinito quindi, per un corretto funzionamento, occore caricare il valore 0 ne registro r0.

Registri di stato e di controllo La tabella 3.8 mostra i registri di stato e di controllo del processore indicandone alcune delle caratteristiche principali.

3.9.3 Set d'istruzioni

Tutte le istruzioni (aritmetiche, logiche, di confronto...) del LatticeMico32 sono a 32 bit. Esistono 4 diversi tipi di formati per le istruzioni come mostrato in figura 3.20:

- Register Immediate (RI);
- Register Register (RR);
- Control Register (CR);
- Immediate (I).

Name	Access	Index	Optional	Description
PC			No	Program counter
IE	R/W	0x0	Yes	Interrupt enable
IM	R/W	0x1	Yes	Interrupt mask
IP	R	0x2	Yes	Interrupt pending
ICC	W	0x3	Yes	Instruction cache control
DCC	W	0x4	Yes	Data cache control
CC	R	0x5	Yes	Cycle counter
CFG	R	0x6	No	Configuration
EBA	R/W	0x7	No	Exception base address
CFG2	R	0xA	No	Extended configuration

R=READ e W=WRITE.

Tabella 3.8: Registri di stato e di controllo del LatticeMico32.

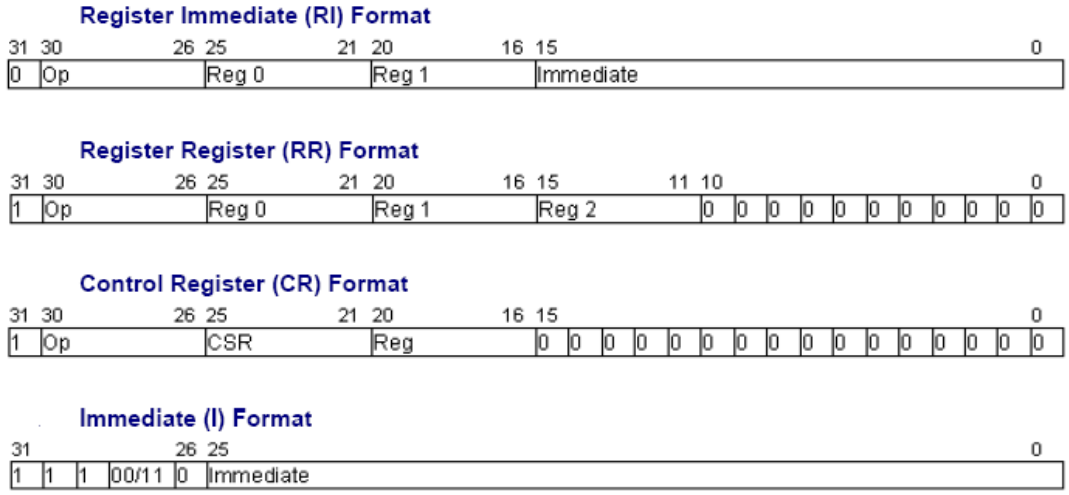


Figura 3.20: Formato delle istruzioni del LatticeMico32.

3.9.4 Tool di sviluppo

Per l'implementazione e lo sviluppo del processore sulle FPGA di Lattice è disponibile il *LatticeMico32 System*. Esso contiene due strumenti, che combinati con l'ambiente *Lattice Diamond* o con l'ambiente *ispLEVER* permettono la realizzazione di un sistema embedded basato su LatticeMico32 e del software per pilotarlo. Esistono due versioni di LatticeMico32 System: una compatibile con Lattice Diamond

e una con ispLEVER. I due strumenti di LatticeMico32 system utilizzati sono:

- **Mico System Builder (MSB)**: utilizzato per l'implementazione hardware del processore sulla FPGA e per la scelta e la connessione delle periferiche;
- **C/C++ Software Project Environment (SPE) and Debugger** : basato sull'ambiente di sviluppo Eclipse C/C++, fornisce gli strumenti (compilatore, assemblatore, linker e debugger) per lo sviluppo del codice da eseguire sul processore.

Per maggiori informazioni sui tool di sviluppo e sulla procedura di implemenrazione si veda [56].

3.10 TSK3000A

TSK3000A è un soft-processor prodotto da *Altium* implementabile in qualsiasi FPGA, indipendentemente dalla casa produttrice. Lo scema a blocchi del processore è riportato in figura 3.21

3.10.1 Caratteristiche principali

Si tratta di un processore RISC a 32 bit le cui caratteristiche principali sono:

- Architettura di memoria di tipo Harvard;
- Pipeline a 5 stadi: Instruction Fetch (IF), Instruction Decode (ID), Execute(EX), Memory Access (MA) e Write Back (WB);
- Indirizzi a 32 bit (spazio d'indirizzamento di 4GB);
- 32 registri general purpose e 10 special function register a 32 bit, oltre al program counter, l'High Word Register (HI) e il Low Word Register (LO);
- Moltiplicatore hardware 32x32→64 bit (con segno e senza segno);
- Divisore hardware 32x32;
- Barrel shifter a 32 bit (1 solo ciclo di clock);
- 32 ingressi d'interrupts, configurabili individualmente per essere sensibili ai livelli o ai fronti, che possono essere usati in 2 diverse modalità:
  - Standard mode: tutti gli interrupt saltano allo stesso indirizzo (interrupt vector adress);
  - Vectored mode: ogni interrupts ha una sua priorità e salta ad un diverso indirizzo (interruzioni vettorizzate);
- Supporta il bus WISHBONE;
- Strumenti di sviluppo software e hardware completi e possibilità di debug.

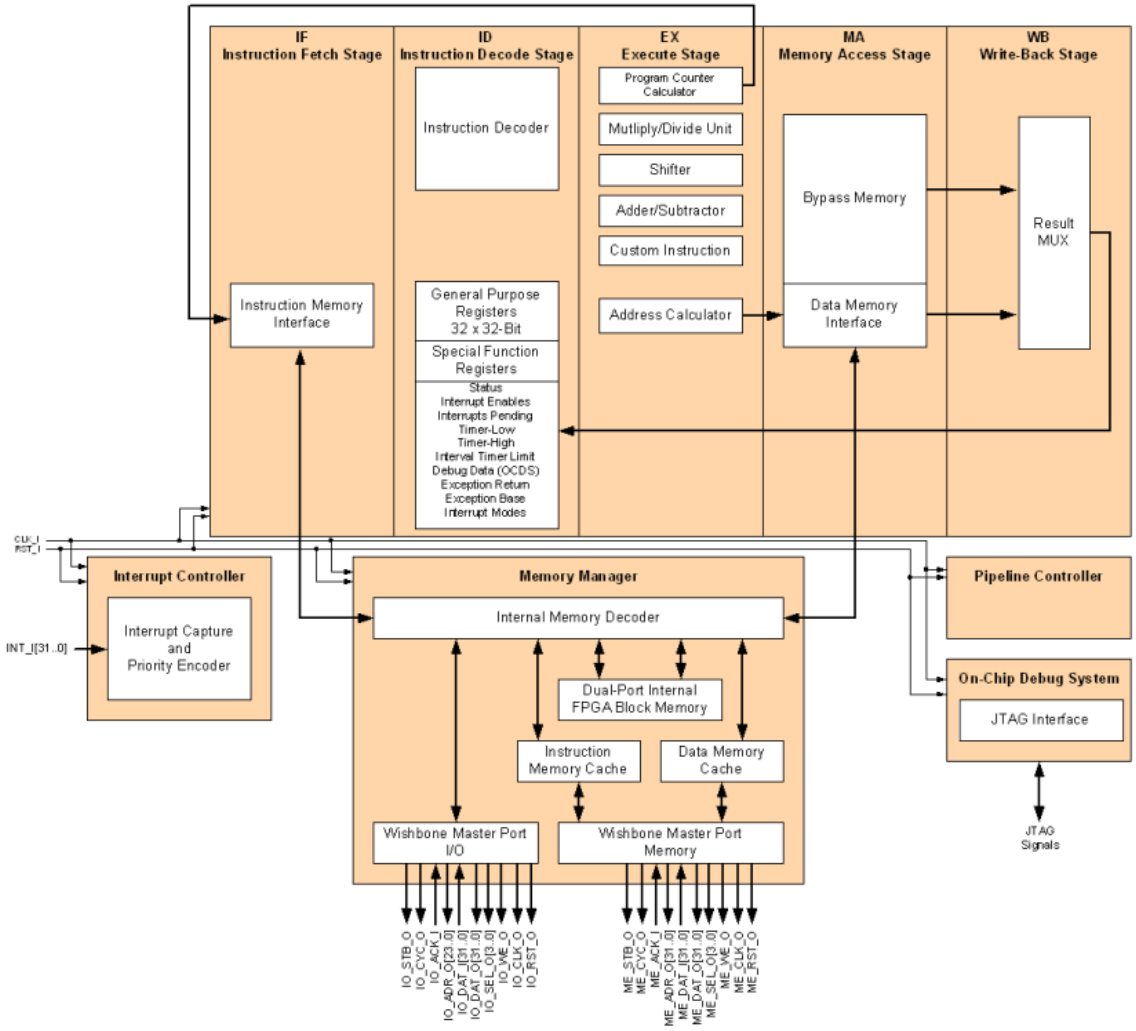


Figura 3.21: Schema a blocchi del TSK3000A.

3.10.2 Registri

General purpose Il processore è dotato di 32 registri general purpose, R0-R31, accessibili mediante istruzioni di tipo R, il cui utilizzo (convenzionale) è mostrato in tabella 3.9. Il valore di inizializzazione all'accensione è 0 per tutti i registri, mentre il reset del dispositivo non modifica il valore memorizzato.

Special function register IL TSK3000A possiede 10 Special function register (SFR) che possono essere letti e scritti (quando possibile) in un solo ciclo di clock. La tabella 3.10 riporta l'elenco di tali registri e delle funzioni da essi implementate. Dopo il reset il valore dei registri è 0000_0000h, fatta eccezione per il PIT che è inizializzato a ffff_ffffh e l'EB che assume il valore 0000_0100h.

Registri addizionali Oltre ai registri precedentemente citati (GPR e SFR) il TSK3000A prevede 3 registri speciali:

Register	Name	Description
R0		Always returns a zero value
R1	at	Assembler Temporary register - used for intermediate macro instruction results
R2-R3	v0-v1	Used for expression evaluations and to hold the integer and pointer type function return values
R4-R7	a0-a3	Used for passing arguments to functions; values are not preserved across function calls. Additional arguments are passed on the stack
R8-R15	t0-t7	Temporary registers used for expression evaluation; values are not preserved across function calls
R16-R23	s0-s7	Saved registers; values are preserved across function calls
R24-R25	t8-t9	Temporary registers used for expression evaluation; values are not preserved across function calls
R26-R27	kt0-kt1	Used by the operating system. kt0 is also used by the Compiler in interrupt handling routines
R28	gp	Global pointer and context pointer
R29	sp	Stack pointer
R30	s8 (or fp)	Saved register (like s0-s7) (or frame pointer)
R31	ra	Return Address register - used by Branch and Link and Jump and Link instructions to store their return address

Tabella 3.9: Registri general prurpose del TSK3000A.

- Program Counter (PC);
- High Word Register (HI): contiene i 32 bit più significativi del risultato nel caso di moltiplicazione e il valore del resto nel caso di divisione;
- Low Word Register (LO): contiene i 32 bit meno significativi del risultato nel caso di moltiplicazione e il quoziente nel caso di divisione;

3.10.3 Set d'istruzioni

Tutte le istruzioni del TSK3000A sono a 32 bit e sono formate dall'opcode, che specifica il tipo d'istruzione e da uno o più operandi. Esistono tuttavia tre diversi formati, come mostrato in figura 3.22:

- Tipo-I: questo tipo di istruzioni include un valore immediato nei 32 bit; sono di questo tipo alcune operazioni aritmetiche e logiche, operazioni di salto e di load/store.
- Tipo-J: questo tipo d'istruzioni è usato quando è richiesto un campo immediato di 26 bit; sono di questo tipo le istruzioni di salto assoluto.
- tipo-R: Questo tipo di istruzioni specifica tutti gli argomenti e i risultati attraverso registri e include un campo immediato di 5 bit utilizzato per le istruzioni di shift e un secondo opcode per distinguere le istruzioni all'interno

Register	Name	Description	Read	Write	Index
Control/Status	Status	Individual Control and Status bits	Yes	Yes	$0
Interrupt Enable	IEnable	Enable/Disable individual interrupts	Yes	Yes	$1
Interrupts Pending	IPending	View of the interrupt values after individual enable gating (i.e. interrupts that are pending or yet to be handled)	Yes	Yes	$2
Time Base Low	TBLO	Least significant 32-bits of the 64-bit time base	Yes	No	$3
Time Base High	TBHI	Most significant 32-bits of the 64-bit time base	Yes	No	$4
Programmable Interval Timer Limit	PIT	Interval length (in clock cycles) of the interval timer	Yes	Yes	$5
Debug Data	Debug	Register for OCDS to exchange data with the processor	Yes	Yes	$6
Exception Return	ER	Register in which to store the return address for interrupts and exceptions	Yes	Yes	$7
Exception Base	EB	Specifies base address for the interrupt vector table	Yes	Yes	$8
Interrupt Mode	IMode	Configures interrupt input pins to operate as either level-sensitive or edge-triggered	Yes	Yes	$9

Tabella 3.10: Special function register del TSK3000A.

di una stessa classe. Sono di questo tipo alcune operazioni aritmetiche e logiche e tutte le istruzioni che operano solo su registri.

3.10.4 Spazio di memoria

Il processore usa un bus indirizzi a 32 bit, quindi è possibile indirizzare uno spazio di 4GByte. Lo spazio di memoria del TSK3000A è diviso in 3 aree: memoria interna (realizzata mediante block ram con dimensione configurabile tra 1K e 16MB) memoria esterna e periferiche di I/O, come mostrato in figura 3.23.

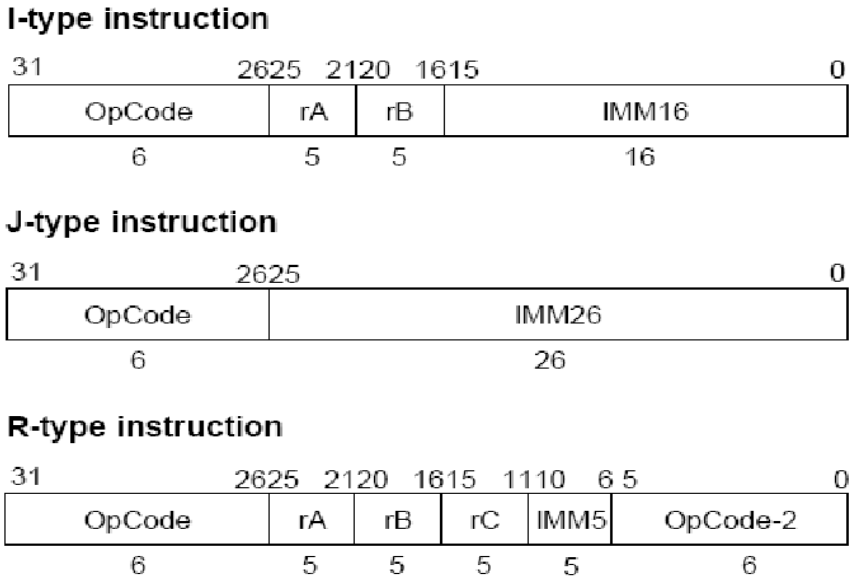


Figura 3.22: Tipi d'istruzioni del TSK3000A.

3.10.5 Tool di sviluppo

Per l'implementazione del processore su FPGA viene utilizzato l'ambiente di sviluppo *Altium designer* (vedi [5]).
Per quanto riguarda il software si fa uso dei tools di *Tasking* (vedi [8]) che permette la programmazione in linguaggi ad alto livello come il C e il C++.

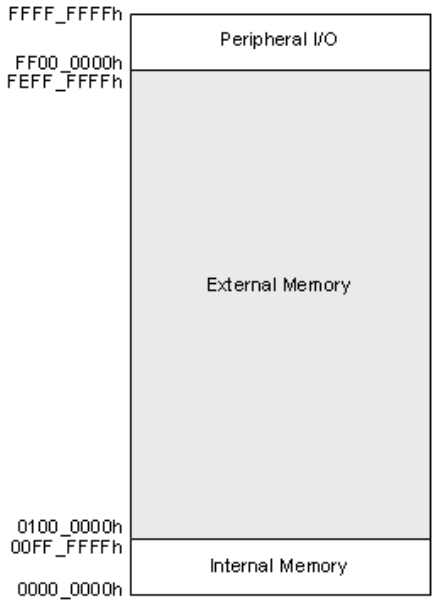


Figura 3.23: Spazio di memoria del TSK3000A.

3.11 TSK51x

Si tratta di un soft-processor prodotto da Altium e disponibile sotto licenza EULA (Altium End-User License Agreement).
Come si può notare dallo schema a blocchi, riporato in figura 3.24, il processore è costituito da quattro componenti fondamentali[7]:

- Control Unit
- Arithmetic Logic Unit
  - Operazioni logiche e aritmetiche a 8 bit;
  - Moltiplicatore a 8 bit;
  - Divisore a 8 bit;
- Memory Control Unit
  - Può indirizzare fino a 64KB di memoria di programma interna;
  - Può indirizzare fino a 64KB di memoria di programma esterna;
  - Può indirizzare fino a 64KB di memoria dati esterna;
- RAM and SFR Control Unit
  - Può indirizzare fino a 256 Bytes di memoria dati;
  - Serve da interfaccia per Special Function Registers off-core;

Altre caratteristiche fondamentali del processore sono:

- Presenza di 2 timer a 16 bit;
- Interrupt controller:
  - 5 sorgenti d'interrupt;
  - 2 livelli di priorità;
- Interfaccia seriale full duplex;
- 4 porte di Input/Output a 8 bit.

Per quanto riguarda gli strumenti di sviluppo vengono utilizzati gli stessi del TSK3000A.

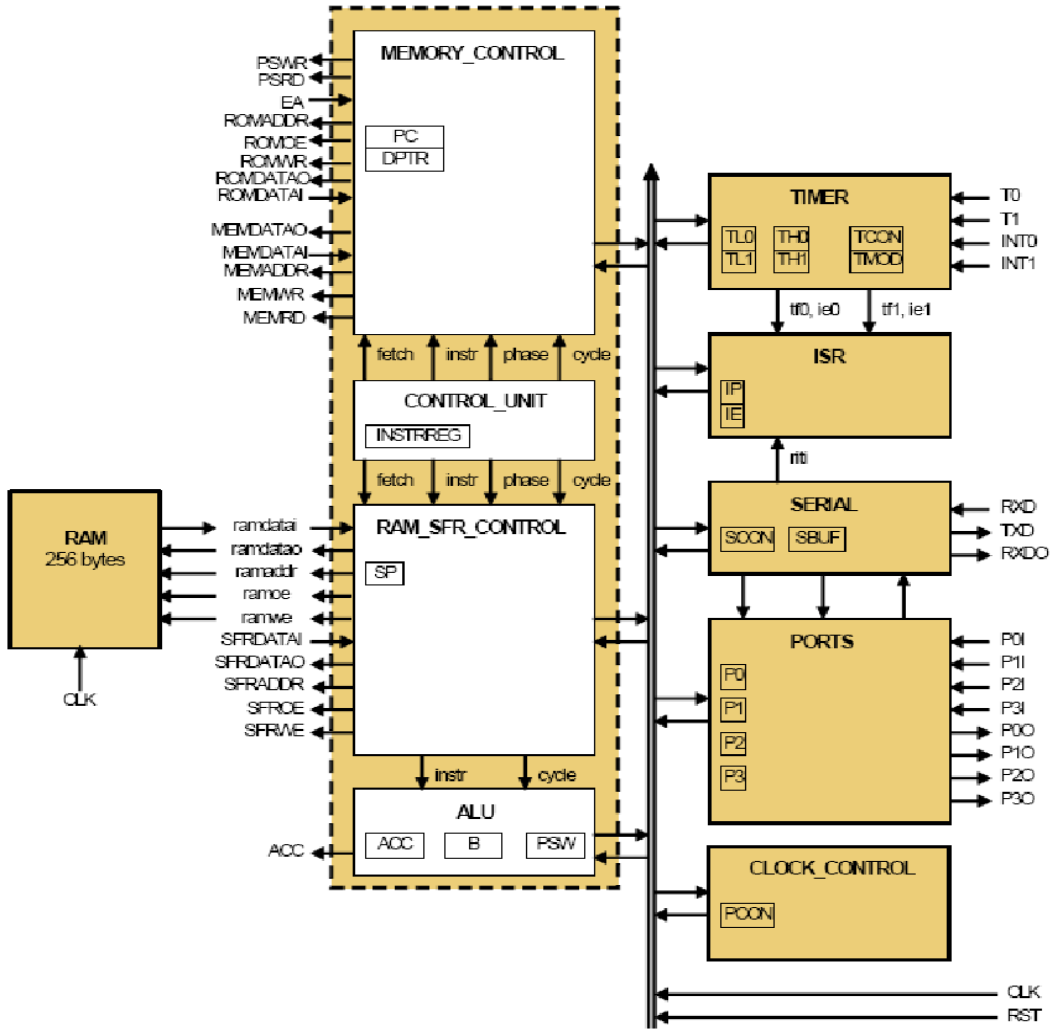


Figura 3.24: Diagramma a blocchi del TSK51x.

3.12 YASEP

Yasep (Yet Another Small Embedded Processor) è un soft-processor realizzato, in VHDL, da Yann Guidon e disponibile gratuitamente sotto licenza GPL. Sono disponibili due versioni del processore, lo YASEP16 e lo YASEP32, che differiscono oltre che per la dimensione del datapath, in alcuni altri punti riguardanti il set d'istruzioni, la memoria e le dimensioni.
Le caratteristiche principali di tale processore, il cui schema a blocchi è riportato in figura 3.25, sono:

- Architettura RISC a 16 o 32 bit a seconda della versione;
- Pipeline a 4 stadi: fetch, decode/read, execute e write back;
- Set d'istruzioni ortogonale: istruzioni a 16 bit con possibilità di estensione a 32; sono permessi quattro formati delle istruzioni:

  - RR: 16 bits con 2 registri;
  - iR: 16 bits con un registro e un valore immediato, con segno, di 4 bits;
  - IRR: 32 bits con 2 registri e un valore immediato, con segno, di 16 bits;
  - RRR: 32 bits con 3 registri. [11] Extended long form (32 bits) with 3 registers (RRR), conditions and other

- 16 registri, R0 R1 R2 R3 R4 NPC A0 D0 A1 D1 A2 D2 A3 D3 A4 D4, divisi in quattro gruppi (rifucendo il numero di opcode necessari):

  - Normal: R0, R1, R2, R3 e R4, usati per memorizzare temporaneamente risultati di operazioni, contatori, parametri di funzioni...
  - Adress: A0, A1, A2, A3 e A4, contengono l'indirizzo in cui verrà letto o scritto il dato;
  - Data: D0, D1, D2, D3 e D4, legati ai registri adresss, contengono il dato memorizzato nella locazione indirzzata dal registro associato (Dx=memory[Ax]);
  - Next PC: NPC, è il puntatore all'istruzione successiva a quella in esecuzione, viene automaticamente incrementato e può essere letto e scritto (la scrittura corrisponde ad un'istruzione di salto).

- Special Registers space (SR) dedicato a operazioni di I/O, configurazione, gestione di eccezioni e interrupt...

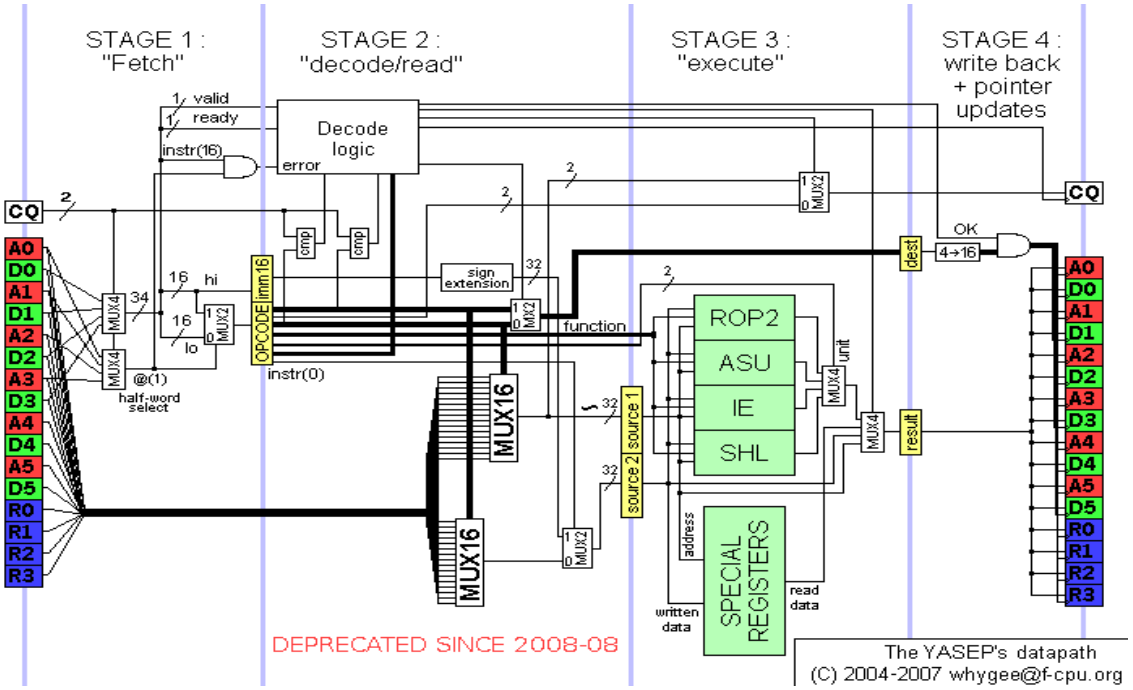


Figura 3.25: Diagramma a blocchi dello YASEP.

3.13 ERIC5

L'ERIC5 è un soft-processor, realizzato in VHDL, prodotto dalla Entner Electronics, implementabile sulle FPGAs di Xilinx, Altera, Lattice e Actel. Viene utilizzato principalmente in applicazioni di controllo in cui è richiesta una bassa occupazione di area e un basso costo.
Si tratta di un processore RISC a 9 bit caratterizzato da piccole dimensioni e basso costo (disponibile sotto licenza il codice VHDL) per cui è disponibile un compilatore C e un assemblatore command-line. Nella sua più piccola configurazione il processore utilizza solamente 110 logic-elements e lavora ad una frequenza massima di 60MHz.
Attualmente sono disponibili 4 diverse implementazioni del processore come mostrato in tabella 3.11, la quale riporta le caratteristiche fondamentali di tali configurazioni. Per maggiori informazioni si veda il manuale del processore [14].

	ERIC5xs	**ERIC5Q**	**ERIC5+**	**ERIC5Q+**
Opcode Width	8bit	8bit	9bit	9bit
9bit Registers	3 (A,B,P)	4 (A,B,P,Q[1])	3 (A,B,P)	4 (A,B,P,Q[1])
Max. Code-Size	1k recommended (128kB possible)	128kB	1k recommended (512kB possible)	512kB
Max. Data-Size	0.5kB	0.5kB	0.5kB	0.5kB
Typ. Fmax	60MHz	60MHz	60MHz	60MHz
Typ. MIPS	25	25	40	40
Multiplier	opt. (18 x 9/18)	opt. (18 x 9/18)	opt. (18 x 9/18)	opt. (18 x 9/18)
Interrupts	n/a	supported	n/a	supported
C-support	no	yes	no	yes

1 Tipicamente usato come stack-pointer.

Tabella 3.11: Configurazioni in commercio dell'Eric5.

3.14 pAVR

Si tratta di un soft-processor realizzato da Doru Cuturela in VHDL, disponibile gratuitamente, compatibile con l'architettura AVR di Atmel. Si tratta di un processore altamente configurabile che permette di simulare la maggior parte dei controllori della famiglia AVR permettendo di avere una velocità tre volte maggiore a parità di tecnologia utilizzata. Le caratteristiche principali del pAVR sono:

- Architettura RISC a 8 bit;

- Pipeline a 6 stadi;
- Quasi tutte le istruzioni vengono eseguite in un ciclo di clock;
- 32 sorgenti d'interupt: ogni interrupt ha una priorità ed un'indirizzo di salto programmabile;
- CPI (clocks per instruction): 1.7 clocks/instruction (tipico), 1 clock/instruction (picco).
- Frequenza tipica di funzionamento pari a 50MHz;
- MIPS a 50MHz: 28 MIPS (tipico), 50 MIPS (picco).

3.15 OpenRISC 1200

L'OpenRISC 1200 è un soft-processor basato sull'architettura OpenRISC 1000(per maggiori informazioni su tale architettura si veda [22] realizzato, in Verilog, da *OpenCores* e disponibile gratuitamente sotto licenza GPL.
Si tratta di un processore RISC a 32 bit utilizzato in diverse applicazioni (internet, elettronica di consumo, etc.), il cui schema a blocchi è riportato in figura 3.27. Da questo si può notare come l'OpenRISC 1200 sia costituito da una serie di unità fondamentali, che vengono di seguito descritte:

- CPU/DSP (figura 3.26)
  - Implementa il set d'istruzioni OpenRISC Basic Instruction Set a 32 bit (ORBIS32);
  - Pipeline a 5 stadi;
  - Un ciclo di clock per quasi tutte le istruzioni;
  - Prestazioni elevate (250 MIPS @ 250MHz nelle condizioni peggiori);
  - 32 registri general purpose a 32 bit;
  - DSP MAC 32x32;
  - Gestione di eccezioni e interrupt: basso tempo di latenza;
  - Possibilità di definire istruzioni personalizzte;
- L1 Caches
  - Architettura Harward: separazione tra cache dati e istruzioni;
  - Dimensione configurabile per ogni cache: 1KB-8KB;
  - Mappatura diretta;
  - Indirizzata fisicamente;
  - Gestione attraverso dei special-purpose registers;

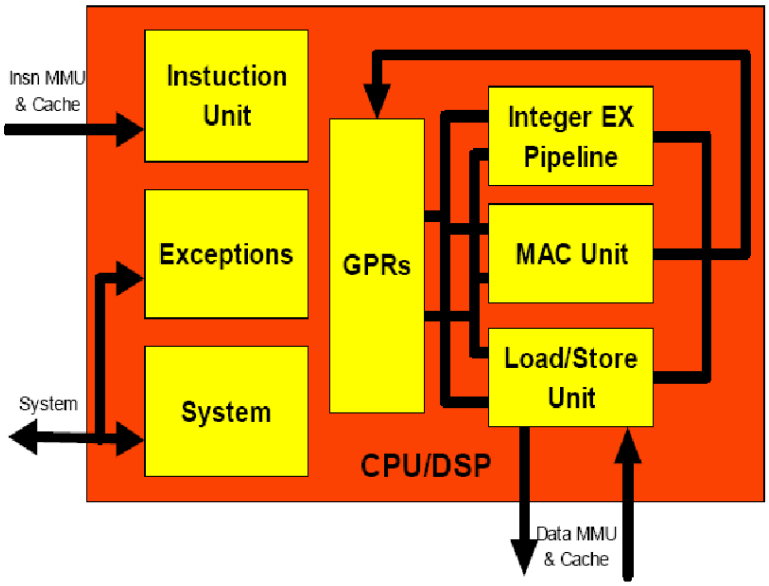


Figura 3.26: Diagramma a blocchi dell'unità CPU/DSP.

- Memory Management Unit
  - Separazione tra MMU dati (DMMU) e MMU istruzioni (IMMU);
  - TLB con dimensione configurabile tra 16 e 128 entries;
  - Indirizzi virtuali a 32 bits e fisici da 24 fino a 32;
  - Dimensione delle pagine pari a 8KB;
- Power Management Unit
  - 4 modalità: slow, idle, doze e sleep;
  - Riduzione consumo di potenza: da 2x a 200x;
  - In modalità slow ed idle il software controlla la riduzione di frequenza permettendo una riduzione del consumo;
  - In modallità doze e sleep il verificarsi di un'interrupt sveglia il processore (il processore ritorna in modalità normale);
- Unità di debug;
- Tick Timer
  - Massimo periodo di conteggio pari a 2^32 cicli di clock;
  - Interrupt mascherabile;
  - Massimo periodo di tempo tra 2 interrupts pari a 2^28 cicli di clock;
  - Single-run, modalità continua o riavviabile;
- Interrupt controller programmabile
  - 2 sorgenti d'interrupt non mascherabili;

  - 30 sorgenti d'interrupt mascherabili;
  - 2 livelli di priorità;

- Unità opzionali e personalizzate
  - Unità addizionali, come la floating point unit, possono essere aggiunte come unità standard;
  - 8 unità personalizzate possono essere agggiunte e controllate attraverso dei registri special purpose o istruzioni personalizzate;

- Interfacce
  - System interface: collegamenti di reset, clock e altri segnali al processore;
  - WISHBONE interfaces : 2 interfacce, una per i dati e una per le istruzioni, permettono il collegamento a periferiche e memorie esterne;
  - Development interface: utilizzata durante il debug;
  - Power management interface: per il collegamento con circuiti esterni per la gestione del consumo;
  - Interrupt Interface: contiene gli ingressi per gli interrupts generati da periferiche esterne.

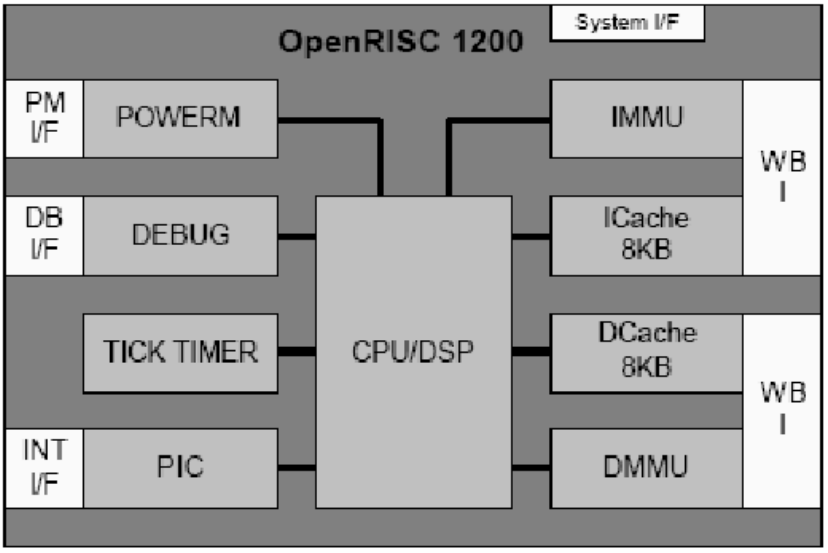


Figura 3.27: Diagramma a blocchi del OpenRisc 1200.

Per quanto riguarda i tool di sviluppo vengono utilizzati OpenIDEA per quanto riguarda il software (permette la programmazione in linguaggi come C e C++) e iNSPIRE/iNCITE per l'hardware.

APPENDICE A

Bus

I bus sono dei canali di comunicazione che permettono lo scambio di dati tra due o più componenti del sistema (detti anche IP core o core, a seconda che siano o meno protetti da licenza) presenti nell'architettura. La scelta del tipo di collegamento, infatti, può sia influenzare il tipo di core che possono essere connessi, sia la dimensione finale di un'architettura. I diversi bus differiscono tra loro per la dimensione dei dati trasportati e per il protocollo di comunicazione, che influenzano il numero di segnali utilizzati[21].
Gli elementi che si affacciano su un BUS sono classificabili in:

- Masters: sono in grado di iniziare e dirigere un operazione di BUS generando gli opportuni indirizzi e segnali di controllo
- Slaves: rispondono alle richieste dei master, leggendo o fornendo i dati richieste. Forniscono inoltre informazioni ai master sull'esito delle operazioni richieste.

Alcuni dispositivi si possono comportare, in tempi diversi, sia da slaves che da master (es. controllore DMA).
Alcuni bus prevedono più master (bus multimaster) quindi sarà presente un meccanismo di arbitraggio che garantisce che un solo master alla volta abbia il controllo del bus fra tutti quelli che ne hanno fatto richiesta.

A.1 IBM CoreConnect

CoreConnect è uno standard semplice e versatile ideato da IBM, e fornisce tutti i mezzi per realizzare un'architettura comprendente un processore ed una serie di

periferiche. Lo standard è utilizzato da un gran numero di processori e di core esistenti, che dunque possono essere facilmente sostituiti tra loro senza modificare l'infrastruttura di comunicazione. IBM CoreConnect prevede quattro elementi fondamentali, che verranno di seguito descritti nel dettaglio.

- **Processor Local Bus (PLB)**: Il bus PLB è caratterizzato da una ampiezza elevata e da basse latenze, e dunque risulta adatto per la connessione di tutti i core altamente performanti, tra cui il processore. Le alte prestazioni sono dovute a diversi fattori, tra cui la scelta di utilizzare due linee separate per i dati in lettura ed in scrittura, che permette il trasferimento di due blocchi di dati per ogni ciclo. Esistono diverse implementazioni che differiscono per il numero di bit che compongono le linee di comunicazione, e che arrivano a proporre 64 bit per gli indirizzi e 128 bit per i dati. Il PLB è un bus sincrono, in quanto esiste un segnale di clock comune a tutti i core collegati che prende il nome di `SYS PLBCLK`. Inoltre, il bus è arbitrato, e dunque è possibile che diverse periferiche (fino ad un limite di 16) possano agire da master sul bus. L'arbitro è un core fornito direttamente da IBM, ed ha una struttura flessibile che permette all'utente di definire diversi schemi di priorità. Tipicamente, il processore possiede due interfacce master per accedere separatamente ai dati ed alle istruzioni, ma possono esistere altre periferiche che fungono da master, come quelle che operano in DMA[1].

- **On-chip Peripheral Bus (OPB)** : Il bus OPB è progettato per la connessione di periferiche più lente, e tipicamente non viene utilizzato per la connessione del processore, anche se questo caso non è escluso. Ha un'ampiezza più limitata del PLB, in quanto prevede indirizzi fino a 64 bit e dati da 32 o 64 bit, e non prevede due trasferimenti contemporanei di dati nello stesso ciclo di clock. Tuttavia, fatto salvo per queste soluzioni che ottimizzano le prestazioni, il protocollo di comunizione sul bus OPB è simile a quello previsto per il PLB. Inoltre, anche OPB è un bus sincrono, ed anche in questo caso è previsto un arbitro distribuito da IBM. La necessità di mantenere i due bus separati, dunque, non nasce tanto da grandi differenze strutturali, quanto dalla volontà di riservare il traffico su PLB per alcune periferiche privilegiate.

- **Device Control Register (DCR) Bus** Il bus DCR è utilizzato per accedere direttamente ai registri di stato e di controllo delle periferiche connesse, il cui contenuto viene trasferito senza passare attraverso il bus OPB e, soprattutto, il più efficiente PLB. Al DCR possono essere connesse periferiche interfacciate sia all'OPB sia al PLB, in quanto il funzionamento del DCR avviene parallelamente a quello degli altri due. Da un punto di vista implementativo, il DCR bus è un bus sincrono con un numero di bit compreso tra 10 e 32 per gli indirizzi, e con 32 bit per i dati.

[1]Il Direct Memory Access (DMA) è una tecnica per trasferire dei dati tra due periferiche senza coinvolgere il processore.

A.2 WISHBONE

Il bus WISHBONE è uno standard con licenza gratuita progettato da Silicore e adottato da OpenCores.org, la cui caratteristica peculiare è la semplicità. Per implementare l'interfaccia WISHBONE, infatti, ad un core è richiesto un numero ridotto di porte logiche, ed è permessa una decodifica parziale degli indirizzi, per non appesantire i core con la logica ridondante. Il bus è sincrono ed ha un'ampiezza variabile, in quanto la dimensione dei dati trasportati non è soggetta a vincoli imposti dallo standard. E' permessa inoltre la presenza di più master in contemporanea ma, a differenza di quanto accade per CoreConnect, l'arbitro deve essere realizzato dall'utente. Infine, l'implementazione del bus non è soggetta a vincoli, e dunque risulta più flessibile rispetto ad OPB. Il bus WISHBONE può essere interfacciato al bus OPB mediante un apposito bridge bidirezionale.

A.3 Avalon

Si tratta di un bus progettato per il collegamento tra processori e periferiche in un system-on-a-programmable chip (SOPC). Avalon è un'interfaccia che specifica i collegamenti tra i vari dispositivi master e slave e fornisce la tempistica con cui tali componenti comunicano tra loro.
Il bus permette il trasferimento di byte, half word e word tra una periferica master e una slave e supporta una serie di caratteristiche avenzate. Per maggiori informazioni si veda [2].

A.4 AMBA

Advanced Microcontroller Bus Architecture (AMBA) è una specifica di bus per dispositivi SoC basati sulle CPU e le celle IP prodotte da ARM[50].
AMBA BUS fornisce specifiche solo a livello di ciclo di clock, pertanto:

- La specifica è indipendente dalla tecnologia che successivamente sarà usata per realizzare il chip su cui il bus viene usato;
- Caratteristiche elettriche: non sono fornite specifiche dato che saranno totalmente dipendenti dalla tecnologia usata;
- Specifiche temporali: non sono fornite informazioni e/o misure temporali di tipo assoluto dato che esse dipendono fortemente dalla tecnologia realizzativa.

AMBA definisce tre BUS diversi per caratteristiche e applicazioni distinte:

- Advanced High-performance Bus (AHB):
  - Operazioni pipelined
  - Supporto a più master

  - Trasferimenti burst
  - Split transactions

- Advanced System Bus (ASB):
  - Operazioni pipelined
  - Supporto a più master

- Advanced Peripheral Bus (APB)
  - Consumi ridotti
  - Latched address and control
  - Interfaccia semplificata verso le periferiche

La figura A.1 mostra l'architettura tipica di sistema basato su AMBA bus:

- Bus AHB: Sistema bus high performance. E' il nucleo di un sistema di interconnessione AMBA. Supporta la connessione a processori, a memorie on-chip e off-chip e l'interfaccia con celle per la gestione low-power della connessione con le periferiche.
- ASB bus: Sistema bus alternativo il cui utilizzo è indicato per le applicazioni in cui le caratteristiche high-performance di AHB non sono richieste. Supporta le stesse connessioni di AHB.
- APB bus: Sistema bus ottimizzato per un minimo consumo di potenza. La complessità dell'intefaccia è ridotta in maniera da supportare la connessione di periferiche lente. Può essere connesso con entrambe le versioni del bus di sistema.

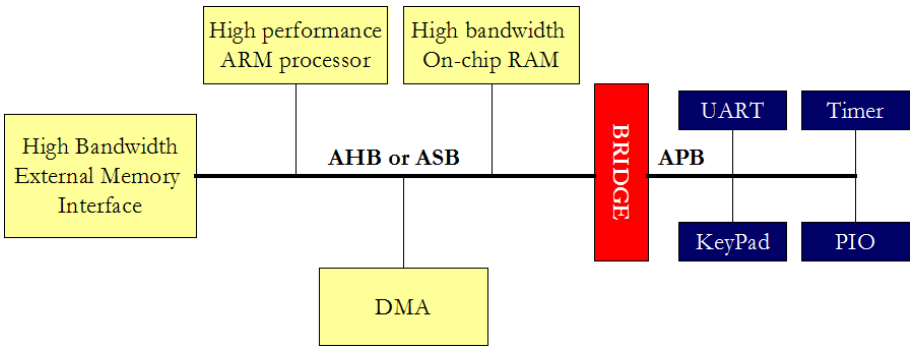


Figura A.1: Architettura di sistema basato su AMBA bus.

Esiste un altro protocollo AMBA per sistemi high-performance e high-frequency: l'AMBA AXI. Le caratteristiche chaive di tale bus sono:

- fasi address/control separate;
- supporto per dati non allineati (byte strobes);

- transazioni burst-based;
- canali separati per read e write per permettere Direct Memory Access (DMA) a basso costo;
- multiple outstanding addresses;
- riordinamento delle transazioni fuori ordine;
- supporto per l'aggiunta di stadi di registi che permettano il rispetto delle constraint di timing.

APPENDICE B

Architettura SPARC

B.1 Caratteristiche

SPARC (Scalable Processor ARChitecture) è il nome di un'architettura per microprocessore big-endian RISC. L'architettura, originariamente disegnata nel 1985 da Sun Microsystems, è anche diventata un trademark registrato da SPARC International, Inc., un'organizzazione nata nel 1989 per promuovere SPARC e per provvedere a rilasciare attestati e test di conformità per processori che si vogliono fregiare di tale titolo.
SPARC International ha voluto che SPARC fosse un'architettura aperta per creare un grande ecosistema per la progettazione, è stato licenziato a vari produttori tra cui Texas Instruments, Cypress Semiconductor e Fujitsu. È risultato che attualmente SPARC è un'architettura aperta e non proprietaria.

Un processore SPARC normalmente contiene almeno 128 registri di uso generico; di questi solo 32 registri sono disponibili direttamente al software, 8 sono globali (il g0 è riservato e quindi non utilizzabile quindi solo 7 sono utilizzabili) e gli altri 24 vengono utilizzati come stack. I 24 registri fungono da register window e quando le funzioni chiamano o ritornano variabili questi registri provvedono a memorizzare i dati, muovendosi in alto o in basso lungo lo stack. Ogni finestra ha 8 registri locali e 8 registri condivisi con le altre finestre. I registri condivisi sono utilizzati per passare i parametri e ritornare le variabili di una funzione.
Lo SPARC è un processore molto scalabile, cioè in grado di essere utilizzato sia come processore per applicazioni embedded che per fornire potenza di calcolo in server aziendali utilizzando sempre lo stesso set di istruzioni. Una delle caratteristiche architetturali che permettono questa scalabilità del processore è il numero di finestre basate su registri che possono essere implementate; le specifiche permettono a queste di variare da un minimo di 3 a un massimo di 32. Un ampio numero aumenta la velocità del codice con molte unità di calcolo mentre un numero ridotto

aumenta la velocità di context switching.
Nel corso degli anni SPARC International ha sviluippato tre versioni dell'architettura: SPARC-V7, SPARC-V8 e SPARC V9, di cui solo le ultime 2 sono attualmente utilizzate nei moderni soft processor. Per maggiori informazioni su queste architetture si vedano [19] e [20].

Bibliografia

[1] *Introduction to the Altera Nios II Soft Processor*.

[2] Altera. *Avalon Bus Specification, Reference Manual*. http://stud4.tuwien.ac.at/~e0125015/nios/lit/avalon_bus_spec.pdf.

[3] Altera. *Nios II Processor Reference Handbook*. http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf.

[4] Altera. *Programming Model, Nios II Processor Reference Handbook*. http://www.altera.com/literature/hb/nios2/n2cpu_nii51003.pdf.

[5] Altium. *Altium Designer-Designing Systems on FPGAs*. http://www.protel.com/files/training/Module6DesigningSystemsonFPGAs.pdf.

[6] Altium. *TSK3000A Core User Guide*. http://www.altium.com/files/learningguides/.%5CCR0121%20TSK3000A%2032%20bit%20RISC%20Processor.pdf.

[7] Altium. *TSK51x Full Reference*. http://wiki.altium.com/download/attachments/3080695/CR0115+TSK51x+MCU.PDF?version=1&modificationDate=1227237294146.

[8] Altium. *Using the TSK3000 Embedded Tools*. http://wiki.altium.com/download/attachments/3080675/GU0111+Using+the+TSK3000+Embedded+Tools.pdf?version=1&modificationDate=1260889653448.

[9] ARM. *ARMv6-M Architecture Reference Manual*. https://silver.arm.com/download/AR585-DC-11001/DDI0419B_arm_architecture_v6m_reference_manual_errata_markup_2_0.pdf.

[10] ARM. *Cortex-M1 handbook*. http://www.actel.com/documents/CortexM1_HB.pdf.

[11] Ensilica. *eSi-1600 16-bit, low-cost and low-power CPU.* http://www.ensilica.com/pdfs/eSi_1600_Product_Brief.pdf.

[12] Ensilica. *eSi-3200 32-bit, low-cost and low-power CPU.* http://www.ensilica.com/pdfs/eSi_3200_Product_Brief.pdf.

[13] Ensilica. *eSi-3250 32-bit, high-performance CPU.* http://www.ensilica.com/pdfs/eSi_3250_Product_Brief.pdf.

[14] Entner-electronics. *Eric5 Core User Guide.* http://www.entner-electronics.com/tl/index.php/eric5.html.

[15] Bryan Fletcher. *FPGA Embedded Processors: Revealing True System Performance.* San Diego, 2005. www.memec.com.

[16] GAISLER. *GRLIB IP Library User's Manual.* http://www.gaisler.com/products/grlib/grlib.pdf.

[17] GAISLER. *LEON3 / LEON3-FT CompanionCore Data Sheet.* http://www.actel.com/ipdocs/LEON3_DS.pdf.

[18] SPARC International. *OpenSPARC T1 Microarchitecture Specification.*

[19] SPARC International. *The SPARC Architecture Manual Version 8.* http://www.sparc.com/standards/V8.pdf.

[20] SPARC International. *The SPARC Architecture Manual Version 9.* http://www.sparc.com/standards/SPARCV9.pdf.

[21] Stefano Bosisio Ivan Beretta. Integrazione del soft processor microblaze nell'architettura riconfigurabile yara. Tesi di laurea in elettronica, Università di Milano, 2006.

[22] OpenCORES. *OpenRISC 1000 Architecture Manual.* http://docs.huihoo.com/openrisc/openrisc1000-arch.pdf.

[23] Gregory Steffan Peter Yiannacouras, Jonathan Rose. The microarchitecture of fpga-based soft processors. 2007.

[24] Shawn Tan. *AEMB 32-bit Microprocessor Core Datasheet.*

[25] Xilinx. *MicroBlaze Processor Reference Guide.* http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf.

[26] Xilinx. *PicoBlaze 8-bit Embedded Microcontroller User Guide.* http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf.

[27] Peter Yiannacouras. The microarchitecture of fpga-based soft processors. Tesi di laurea in elettronica, Università di Toronto, 2005.

Siti consultati

[28] http://en.wikipedia.org/wiki/Fpga

[29] http://wiki.altium.com/display/ADOH/Processor-based+FPGA+Design

[30] http://www.opensparc.net/opensparc-t1/index.html

[31] http://www.xilinx.com/MicroBlaze

[32] http://wiki.altium.com/display/ADOH/Processor-based+FPGA+Design

[33] http://altera.com/products/ip/processors/nios2/ni2-index.html

[34] http://www.arm.com/products/processors/cortex-m/cortex-m1.php

[35] http://www.ensilica.com/ip_esi_risc.htm

[36] http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/index.cfm

[37] http:http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=156&Itemid=104

[38] http://opencores.org/project,or1k

[39] http://www.aeste.my/aemb

[40] http://bleyer.org/pacoblaze

[41] http://www.ht-lab.com/freecores/cpu8086/cpu86.html

[42] http://www.fpgacpu.org/xsoc/index.html

[43] http://www.entner-electronics.com/tl/index.php/eric5.html

[44] http://yasep.org/

[45] http://opensource.zylin.com/zpu.htm

[46] http://en.wikipedia.org/wiki/Soft_microprocessor

[47] http://www.cvorg.ece.udel.edu/cpeg422f08/documents/s3embedded_tutorial_XPS_SDK_flow.pdf

[48] http://en.wikipedia.org/wiki/Nios_II#Development_processes

[49] http://www.actel.com/products/software/softconsole/default.aspx

[50] http://www.cdls-inf.ing.unipi.it/documentiScaricabili/Sistemi%20embedded/File/esercitazione_3.pdf

[51] http://www.gaisler.com

[52] http://it.wikipedia.org/wiki/Register_window

[53] http://space.ednchina.com/upload/2009/5/11/058e8ff7-c1e7-4afd-8852-1b803a8b99b8.pdf

[54] http://klabs.org/mapld05/presento/212_craven_p.ppt

[55] http://bleyer.org/pacoblaze/

[56] http://www.slideshare.net/Flashdomain/lattice-mico32-3734049