Studio di fattibilità di implementazione di processori open-hardware in grado di integrarsi in una architettura blockchain

Report 1:

Descrizione e analisi di processori open-hardware

Con la collaborazione di:
Università di Trento

UNIVERSITÀ
DI TRENTO
Dipartimento di Matematica
Laboratorio di Matematica Industriale e Crittografia

Fornitore:
MicroFabSolutions s.r.l.

MicroFabSolutions

Attività svolta nell'ambito dell'Avviso promosso dal Ministero dell'Università e della Ricerca per la presentazione di Idee progettuali per Smart Cities and Communities and Social Innovation di cui al D.D. n. 391/Ric. del 5 luglio 2012 e ss.mm.ii. - SIN_00968 THE LEARNING METERS NETWORK: workpackage formativo del SCN_00398 - CUP J49G14000140008.

Indice

1 Introduzione

1.1 Obiettivo dello studio

L'architettura blockchain, letteralmente "catena di blocchi", sfrutta le caratteristiche di una rete informatica di nodi e consente di gestire e aggiornare, in modo immutabile e sicuro, un registro contenente dati e informazioni (per esempio, transazioni) in maniera aperta, condivisa e distribuita senza la necessità di un'entità centrale di controllo e verifica. Dalla sua comparsa nel 2009, la blockchain ha subito ricevuto un forte interesse, per i suoi vari ambiti applicativi: crittovalute e settore finanziario, notarizzazione di dati, tracciamento di beni, memorizzazione di flussi di dati generati da sensori.

Il funzionamento e l'affidabilità della blockchain si fonda sull'utilizzo di diverse primitive crittografiche, come funzioni di hash e firme digitali. Esse basano la loro sicurezza sulla difficoltà di risoluzione di problemi matematici selezionati. In particolare, gli algoritmi di firma digitale maggiormente utilizzati sono protocolli che eseguono efficientemente operazioni definite su gruppi di punti di curve ellittiche, e basano la loro sicurezza sulla difficoltà del problema del logaritmo discreto. Esiste quindi un trade-off tra l'efficienza (selezione di curve ellittiche in cui le operazioni di gruppo possono essere efficientemente eseguite) e la sicurezza (le curve ellittiche devono essere selezionate in modo da evitare vulnerabilità indotte da proprietà algebrico-geometriche). Le funzioni di hash sono invece funzioni in cui la controimmagine di un elemento non è calcolabile in tempo polinomiale. Le funzioni di hash basano la loro sicurezza sulla difficoltà di risoluzione di equazioni nonlineari su campi finiti. Anche in questo caso occorre prestare attenzione al rapporto tra sicurezza e efficienza di esecuzione delle operazioni.

Ai fini della sicurezza e dell'efficienza, risulta utile avere pieno controllo degli algoritmi di cifratura implementati e del design dei microprocessori integrati nell'architettura blockchain.

Diverse piattaforme hardware sono specializzate nell'implementazione di primitive crittografiche, e sul mercato sono disponibili diversi processori crittografici, come, per esempio, generatori hardware di numeri casuali, coprocessori crittografici specializzati sulle versioni di AES e motori crittografici per il calcolo di hash.

Negli ultimi anni, alcune società no profit, come la Risc-V Foundation, si sono costituite con l'obiettivo di promuovere l'adozione e l'implementazione di hardware open-source. Questo tipo di progetto consente a piccole aziende di realizzare processori e microcontrollori personalizzati senza pagare i relativi diritti a società terze, grazie alla natura open-source del progetto stesso.

Lo studio della compatibilità tra dispositivi open-hardware in silicio e architetture blockchain si snoda in due punti chiave:

- Report 1: Descrizione e analisi dei dispositivi open-hardware e relativa architettura.
- Report 2: Selezione e studio delle principali primitive crittografiche utili all'implementazione di una blockchain. Analisi e confronto degli algoritmi crittografici, com-

parando le caratteristiche delle diverse primitive con le limitazioni e potenzialità dei dispositivi open-hardware esistenti.

1.2 Definizioni

Acceleratore esterno	Un componente hardware progettato per migliorare le prestazioni della piattaforma che lo ospita. In alcuni ambiti specifici permette di scaricare la CPU da tutte quelle attività che altrimenti occuperebbero buona parte delle risorse computazionali e che possono essere lasciate libere per svolgere più agevolmente altri compiti.
ASIC	Un circuito integrato creato appositamente per risolvere un preciso problema; la specificità della progettazione consente di raggiungere altissime prestazioni in termini di velocità e consumo elettrico.
Blockchain	Registro dati immutabile composto da una catena di blocchi. Ogni blocco contiene un insieme di transazioni e, escluso il primo, un puntatore (tipicamente un hash crittografico) al blocco precedente.
Cache	Parte della memoria in cui un computer immagazzina le informazioni più frequentemente usate, in modo da rileggerle più velocemente.
Chisel	Un linguaggio di descrizione hardware di tipo open-source, usato per descrivere l'elettronica digitale e i circuiti.
Compilatore	Un software che traduce una serie di istruzioni scritte in un determinato linguaggio di programmazione in istruzioni di un altro linguaggio: il processo di traduzione si chiama compilazione.
Circuito integrato	Un circuito elettronico miniaturizzato dove i vari transistori sono stati formati tutti nello stesso istante grazie a un unico processo fisico-chimico.
Coprocessore	Una tipologia di processore che si contraddistingue per essere ausiliaria ad un altro processore, ad esempio nei moderni computer la CPU è spesso affiancata da processori ausiliari come la GPU.

Core	Il nucleo elaborativo di un microprocessore. Quest'ultimo infatti è costituito in realtà da 2 componenti principali: il core appunto, e il package che lo contiene. In alcune tipologie di processori è visibile la posizione del core, costituito da un piccolo rettangolo nero leggermente sporgente al centro del package: esso contiene i transistor, che ne determinano funzionamento e capacità di elaborazione.
Debian	Un sistema operativo multipiattaforma e composto interamente da software libero.
Fabless	Un modus operandi tipico di alcune aziende, per il quale viene effettuata la progettazione e vendita di dispositivi hardware e circuiti integrati mentre la effettiva fabbricazione viene esternalizzata a una azienda specializzata.
FGPA	Un dispositivo logico programmabile, ovvero un dispositivo hardware elettronico formato da un circuito integrato le cui funzionalità logiche di elaborazione sono appositamente programmabili e modificabili tramite opportuni linguaggi di descrizione hardware.
Firma digitale	Un metodo matematico teso a dimostrare l'autenticità di un messaggio o di un documento digitale inviato mediante un canale di comunicazione non sicuro, garantendo al destinatario che il mittente del messaggio sia chi dice di essere (autenticazione), che il mittente non possa negare di averlo inviato (non ripudio) e che il messaggio non sia stato alterato lungo il percorso dal mittente al destinatario (integrità).
FreeBSD	Un sistema operativo libero basato su Unix, derivato dalla Berkeley Software Distribution.
Funzione di hash	Funzione matematica che associa a un dato di lunghezza arbitraria un valore (digest) di lunghezza fissa. Le funzioni di hash usate in crittografia devono rispettare ulteriori proprietà di sicurezza.
Interfaccia	Il componente fisico o logico che permette a due o più sistemi elettronici di comunicare e interagire.
Kernel	Il software che fornisce ai processi in esecuzione su un dato sistema un accesso sicuro e controllato all'hardware.

Logaritmo discreto	Un problema matematico sulla cui difficoltà si basano diversi protocolli crittografici. Dato un gruppo ciclico moltiplicativo G, un generatore g di G e g^a per un certo $a \in \mathbb{Z}$, il problema consiste nel determinare a.
LowRISC	Una società senza scopo di lucro con sede a Cambridge e molto attiva nell'ambito degli open-hardware.
Microprocessore	Una tipologia particolare di circuito elettronico che si contraddistingue per essere interamente costituita da uno o più circuiti integrati e per questo di dimensioni molto ridotte.
Open-Hardware	Termine generico per indicare hardware elettronici che sono stati progettati con la stessa politica del software libero ed open-source.
Periferica	Un qualsiasi dispositivo hardware che fa parte di un sistema informatico e/o di elaborazione elettronica e che funziona sotto il controllo di una unità centrale e del sistema operativo, alla quale è collegata.
PicoRV32	Un core della CPU che implementa il set di istruzioni Risc-V RV32IMC.
Pipeline	Una catena di fasi di elaborazione dei dati. Ogni stadio di questa catena provvede a ricevere in ingresso un dato o un segnale, ad elaborarlo e poi a trasmetterlo all'elemento successivo.
Pipeline in-order	Un metodo di elaborazione dati per cui la CPU esegue le istruzioni in ordine sequenziale e fino al completamento dell'istruzione corrente non eseguirà l'istruzione successiva.
Pipeline out-of-order	Un metodo di elaborazione dati per cui la CPU esegue le istruzioni in ordine non sequenziale: anche se l'istruzione corrente non è completata, eseguirà l'istruzione successiva se possibile.
Primitiva crittografica	Un algoritmo crittografico che viene utilizzato per creare protocolli crittografici per i sistemi di sicurezza.
Risc-V	Architettura ISA sviluppata nel 2010 dall'Università della California con lo scopo di creare un set di istruzioni estensibile e open-source per uso accademico e commerciale.
RocketChip	Un generatore SoC open-source creato a Berkeley.

Scala	Un linguaggio di programmazione di tipo general-purpose multi-paradigma studiato per integrare le caratteristiche e funzionalità dei linguaggi orientati agli oggetti e dei linguaggi funzionali.
SiFive	Una azienda di tipo fabless che progetta processori Risc-V.
Spike	Un simulatore ISA.
SpinalHDL	Un linguaggio di descrizione hardware.
Transistor	Un dispositivo a semiconduttore largamente usato nell'elettronica analogica.
x86	Un'espressione generica per indicare un'architettura di una famiglia di microprocessori sviluppata e prodotta da Intel.

1.3 Acronimi

ASIC	Application Specific Integrated Circuit.
AXI	Advanced eXtensible Interface.
CHISEL	Constructing Hardware in a Scala Embedded Language.
CISC	Complex Instruction Set Computing.
CPU	Central Processing Unit.
DMA	Direct Memory Access.
GPIO	General Purpose Input/Output.
ISA	Instruction Set Architecture.
I2C	Inter Integrated Circuit.
JTAG	Join Test Action Group.
POWER	Performance OptimizationWith Enhanced RISC.
RISC	Reduced Instruction Set Computing.
RoCC	Rocket Custom Co-Processor.
ROM	Read-Only Memory.
RTL	Register Transfer Level.

SIMD	Single Instruction, Multiple Data.
SoC	System on a Chip.
SPI	Serial Peripheral Interface.
TLB	Translation Lookaside Buffer.
UART	Universal Asynchronous Receiver/Transmitter.
VHDL	VHSIC Hardware Description Language.
VHSIC	Very High Speed Integrated Circuits.

2 Dispositivi open-hardware e relativa architettura

L'*hardware open-source* è l'hardware il cui progetto è reso pubblico in modo che chiunque possa studiare, modificare, distribuire, realizzare, e vendere il progetto o l'hardware basato su di esso. Questo termine si riferisce quindi a dispositivi hardware che sono stati progettati con la stessa politica del software libero ed open-source. L'hardware libero è parte della cultura dell'open source, che espande quest'ideologia al di fuori dell'ambito del software.

2.1 ISA

Il primo passo da fare per la nostra analisi è chiarire il concetto di ISA (Instruction Set Architecture). Un ISA è un modello astratto di un processore, il quale a sua volta è un'implementazione del primo. Un ISA descrive in modo dettagliato e indipendente dall'implementazione i seguenti aspetti:

- **Instruction Set**, ovvero l'insieme delle istruzioni di base che il processore può compiere e che costituiscono dunque il suo linguaggio macchina, a partire dal quale vengono scritti i relativi programmi nei vari linguaggi di programmazione a più alto livello di astrazione.
- **Instruction Format**, ovvero come sono fatte le istruzioni.
- **Data storage**, ovvero dove sono posizionati i dati.
- **Addressing Modes**, ovvero come si accede ai dati.
- **Exceptional Conditions**, ovvero come vengono gestiti i casi eccezionali.

Computer con microarchitetture differenti possono condividere la stessa ISA, ad esempio, l'Intel Pentium e l'AMD Athlon implementano versioni quasi identiche dell'Instruction Set x86, pur essendo al loro interno totalmente diversi.

2.1.1 RISC vs CISC

I termini RISC e CISC indicano due famiglie distinte di modelli architetturali, con un funzionamento di base diametralmente opposto. Un'architettura RISC (Reduced Instruction Set Computing) ha istruzioni semplici che possono essere eseguite in un singolo ciclo di clock del computer, di conseguenza il completamento di un'attività specifica spesso richiede l'esecuzione di più istruzioni. L'elaborazione CISC (Complex Instruction Set Computing) cerca di completare un'attività con il minor numero di righe di codice possibile e pertanto una singola istruzione CISC può richiedere più cicli di clock del computer per essere completata. I processori CISC cercano quindi di ridurre al minimo il numero di istruzioni per programma a scapito del numero di cicli di clock del computer per istruzione. RISC adotta l'approccio opposto, utilizzando semplici istruzioni che possono essere eseguite in un singolo ciclo di clock al costo di un numero maggiore di istruzioni. In generale, l'approccio RISC è più efficace nel ridurre il consumo energetico

complessivo, a volte a scapito di prestazioni inferiori.

Per fare alcuni esempi concreti: i processori Risc-V e ARM si basano sul modello RISC, mentre i processori x86 di Intel e AMD hanno un'impostazione di tipo CISC. Nel corso degli anni il mercato delle CPU è stato dominato da Intel e AMD per quanto riguarda le loro architetture di tipo x86, e da ARM, che pur non essendo un produttore sviluppa i progetti dei processori per dispositivi mobili e microcontrollori, i quali sono poi concessi in licenza ai produttori che li integrano nei propri prodotti o li utilizzano per sviluppare e vendere il proprio SoC. Questi processori sono tipicamente closed-source e pertanto i loro ISA non possono essere utilizzati per sviluppare architetture di terze parti senza autorizzazione. Tutte queste limitazioni rendono lo sviluppo di nuovi processori più costoso e aumentano le difficoltà di condividerne il know-how relativo.

2.1.2 Il Power ISA

Power ISA [1] è un'ISA sviluppato dalla OpenPOWER Foundation e dalla IBM, presentato ora nella versione 3.1, datata maggio 2020. Le radici di questo ISA risalgono tuttavia a oltre un quarto di secolo fa: l'architettura POWER, basata sul modello RISC, è stata introdotta all'inizio del 1990. Successivamente, nell'anno seguente, Apple, IBM e Motorola hanno iniziato una collaborazione per ampliare l'applicabilità dell'architettura; il risultato è PowerPC. Nel 1997 Motorola e IBM hanno iniziato un'altra collaborazione, incentrata sull'ottimizzazione di PowerPC per sistemi embedded, la quale ha prodotto Book E. Nel 2006, Freescale e IBM hanno collaborato alla creazione della versione 2.03 di Power ISA, che rappresentava la riunificazione dell'architettura combinando Book E con PowerPC. Lo stato attuale di Power ISA è la versione 3.1, ed è composta da tre libri più una serie di appendici.

- Il Libro I, Power ISA User Instruction Set Architecture, descrive l'architettura del set di istruzioni per l'utente e copre il set di istruzioni di base e le relative funzionalità disponibili per il programmatore dell'applicazione.
- Il Libro II, Power ISA Virtual Environment Architecture, descrive l'architettura dell'ambiente virtuale e definisce il modello di archiviazione e altre istruzioni e funzionalità che consentono al programmatore dell'applicazione di creare programmi multithread e programmi che interagiscono con determinate realtà fisiche dell'ambiente informatico.
- Il Libro III, Power ISA Operating Environment Architecture, definisce le istruzioni del supervisore e le relative strutture.

Oltre a fornire il supporto per l'aritmetica degli interi e la possibilità di effettuare shift nei registri, sono presenti funzionalità specifiche per la moltiplicazione di polinomi, molto utili per lavorare con i campi finiti in contesti crittografici. Dalla versione 3.0 sono state introdotte due istruzioni (addex e vmsumudm) per accelerare l'aritmetica degli interi a precisione arbitraria e, in particolare, per accelerare l'implementazione nelle blockchain dell'algoritmo di firma basato su curve ellittiche.

2.1.3 L'ISA Risc-V

Nel 2010 a Berkeley, presso l'Università della California, è stata sviluppata una nuova architettura ISA, chiamata Risc-V [2]. Tale architettura ha lo scopo di creare un set di istruzioni estensibile e open-source, non solo per uso accademico, ma anche per prodotti commerciali. Di conseguenza, Risc-V ha ricevuto un supporto significativo dalla comunità open-source, con l'adattamento e lo sviluppo di molti strumenti di programmazione, come un compilatore GCC con supporto GDB, una toolchain LLVM, GNU MCU Eclipse, librerie C (newlib, glibc), un simulatore ISA ufficiale (Spike) e un simulatore in QEMU. In termini di sistemi operativi, ha già il supporto per il kernel Linux, FreeBSD e le porte di Debian. La Risc-V Foundation controlla l'evoluzione di Risc-V e i suoi membri sono responsabili di promuovere l'adozione di questo standard e partecipare allo sviluppo del nuovo ISA. Nella lista dei membri che finanziano l'iniziativa ci sono grandi aziende come Google, NVIDIA, Western Digital, Samsung e Qualcomm, inoltre molte altre aziende offrono o hanno annunciato hardware [3] basato su questo standard:

- Il progetto LowRISC sta lavorando con questo processore e mira a diventare il "linux dell'hardware".
- SiFive e OnChip stanno creando prodotti in silicio personalizzati.
- L'ETH di Zurigo e l'Università di Bologna collaborano per creare un processore all'avanguardia a basso consumo (PULP) e un microcontrollore single-core derivato da quel processore (PULPino).
- In India, il governo ha deciso che Risc-V è il nuovo "ISA nazionale", anche per i sistemi di difesa, e l' IIT-Madras sta sviluppando una gamma di processori ad-hoc.
- Recentemente NVIDIA e Western Digital stanno lavorando sui propri microcontrollori Risc-V [4] per incorporarli nei loro prodotti commerciali, in alternativa alle loro attuali soluzioni ARM. In particolare NVIDIA utilizzerà Risc-V ISA per la sostituzione del proprio processore Falcon.

Con lo sviluppo di Risc-V, ci si aspetta che il paradigma effettivo cambierà attraverso la creazione di un ecosistema software universale e open-source con il contributo di tutti. L'obiettivo principale è supportare più implementazioni, dando la libertà alle aziende, università o centri di ricerca di sviluppare le proprie soluzioni hardware, sapendo che questo livello software esiste, e può essere utilizzato senza limitazioni. In futuro, se questa idea saprà dimostrarsi valida, Risc-V potrebbe essere presente nei processori di diverse piattaforme, come computer, smartphone e altri tipi di microcontrollori.

Nel seguito descriveremo Risc-V, concentrandosi sull'insieme delle istruzioni di base, le estensioni esistenti, e il modo in cui queste cooperano.

Risc-V racchiude un insieme di istruzioni (di seguito anche *base-integer instruction set*) con varianti con 32, 64 e 128 bit, chiamate rispettivamente RV32, RV64 e RV128, inoltre

supporta estensioni opzionali che offrono la possibilità di utilizzare ciò che è necessario in una specifica applicazione. Nonostante la vasta diffusione che sta avendo, la fondazione a capo di Risc-V non ha ancora ratificato il relativo standard ISA, e nella bozza più recente del manuale associato [2] alcuni dei moduli hanno subito modifiche significative, mentre altri sono invece stati ratificati, come RV32I e la *Multiplication and Division extension*. Attualmente l'RV32I dispone di meno istruzioni rispetto a quanto deciso nella sua versione originale, di conseguenza questo significa meno complessità e meno risorse quando è il momento di implementare i processori sottostanti. In generale comunque quando un processore implementa un ISA è fondamentale sapere quali sono le istruzioni che dovrebbero essere supportate e implementate, al fine di definire i requisiti hardware come la dimensione della memoria, la struttura dei file di registro, l'unità aritmetica-logica (ALU), il moltiplicatore e il divisore.

Attualmente l'ISA RV32I (versione 2.1) contiene 40 istruzioni di base [2], ed è stato progettato per ridurre al minimo le risorse hardware e fornire supporto per piccole implementazioni. Le versioni precedenti dell'RV32I includevano le istruzioni per l'accesso ai registri di stato e di controllo (CSR) per accedere a timer e contatori anche usando del codice non privilegiato. Attualmente, queste istruzioni non sono obbligatorie ed è stata creata un'estensione dedicata. L'insieme di istruzioni è diviso nei seguenti cinque gruppi.

- 21 istruzioni per gestire il calcolo con i numeri interi. Nel dettaglio, è possibile effettuare operazioni per gestire l'aritmetica (addizione, sottrazione e shift dei registri), operazioni logiche (operazioni bit-a-bit) e comparazioni (di tipo aritmetico).
- 8 istruzioni per il caricamento e la memorizzazione di dati. Nel dettaglio, è possibile effettuare operazioni di caricamento di un byte (con e senza segno), di una half-word (con e senza segno) e di una word, e di memorizzazione di un byte, di una half-word e di una word.
- 1 istruzione per l'ordinamento. Una sola istruzione, *Fence*, viene utilizzata per ordinare gli accessi alla memoria da parte di altri core o da altri dispositivi.
- 8 istruzioni per il controllo di flusso. Sono supportati sia i rami condizionali che i salti incondizionati.
- 2 istruzioni di sistema. Le istruzioni di sistema vengono utilizzate per richiamare il debugger o per effettuare chiamate di sistema. L'istruzione EBREAK viene utilizzata per interrompere l'esecuzione e tornare al controllo del debugger, mentre l'istruzione ECALL viene utilizzata per effettuare una chiamata esterna.

Uno degli obiettivi principali di Risc-V è quello di essere adatto a tutti i tipi di applicazioni, a partire dalla fascia bassa, con limitazioni hardware, dove l'insieme di istruzioni di base è sufficiente, ma anche raggiungere processori ad alte prestazioni che potrebbero richiedere, ad esempio, operandi in virgola mobile o supporto per operazioni di divisione e moltiplicazione. Questo obiettivo può essere raggiunto definendo estensioni

al set di istruzioni, che può essere implementato o meno, a seconda delle esigenze e delle necessità di ogni architettura. Poiché le estensioni sono indipendenti, possono essere sviluppate in parallelo e possono essere create da terze parti, senza influire sugli ISA di base. La tabella 1 presenta l'elenco di tutte le estensioni in fase di sviluppo da parte della Risc-V Foundation. È possibile osservarne diverse estensioni già ratificate, come la moltiplicazione e la divisione di numeri interi, il registro di controllo e di stato istruzioni, virgola mobile a precisione singola, doppia e quadrupla. Le estensioni contrassegnate come Bozza sono in in fase di sviluppo e pertanto suscettibili di cambiamenti.

Estensione	Descrizione	Versione	Stato
Zifencei	Instruction-Fetch per il comando Fence	2.0	Ratificato
Zicsr	Istruzioni per i registri di stato e di controllo	2.0	Ratificato
M	Moltiplicazione e divisione intera	2.0	Ratificato
A	Istruzioni atomiche	2.0	In sospeso
F	Operazioni in virgola mobile (single-precision)	2.2	Ratificato
D	Operazioni in virgola mobile (double-precision)	2.2	Ratificato
Q	Operazioni in virgola mobile (quad-precision)	2.2	Ratificato
C	Operazioni compresse	2.0	Ratificato
Ztso	Estensione per il TSO (Total Store Ordering)	0.1	In sospeso
Counters	Contatori e timer delle prestazioni	2.0	Bozza
L	Estensione per aritmetica decimale in virgola mobile	0.0	Bozza
B	Estensione per manipolazione dei bit	0.0	Bozza
⋮	⋮	⋮	⋮

Tabella 1: Estensioni per l'ISA Risc-V.

Ai fini del nostro lavoro merita un'attenzione particolare l'estensione standard per moltiplicazione e divisione intera. Questa estensione consente la moltiplicazione e la divisione tra due operandi memorizzati in due registri. In particolare, ci concentriamo solo sulle istruzioni a 32 bit.

- Per quel che riguarda la moltiplicazione sono disponibili quattro istruzioni, la prima delle quali (MUL) viene utilizzata per ottenere i primi 32 bit (meno significativi) del risultato dell'operazione, ed le restanti tre (MULH, MULHSU, MULHU) permettono di ottenere gli ultimi 32 bit, in accordo con quanto descritto in Figura 2. Se sono richiesti entrambi i blocchi da 32 bit è il caso di eseguire le istruzioni sequenzialmente MULH e MUL, assicurandosi allo stesso tempo che i registri degli operandi siano preservati. A livello di architettura, entrambe le operazioni possono

essere eseguite utilizzando una sola operazione di moltiplicazione, riducendo la latenza per ottenere entrambe le parti del risultato.

- Per quel che riguarda la divisione sono disponibili quattro istruzioni, due per la divisione (con e senza segno) e due per ottenere il resto. In queste istruzioni rs1 memorizza il dividendo e rs2 memorizza il divisore. La divisione arrotonda a zero il quoziente, e il resto ha il segno del dividendo, secondo quanto definito dallo Standard C99. Quando sono necessari il quoziente e il resto dovranno essere eseguite entrambe le istruzioni, oppure, in alternativa, la microarchitettura potrebbe implementare un meccanismo per memorizzare entrambi i risultati, visto che una divisione è un'operazione aritmetica che richiede diversi cicli di clock per ottenere il risultato. Nel Risc-V ISA, si è deciso di non lanciare nessuna eccezione quando si cerca di dividere per 0. Per rilevare queste situazioni, un'istruzione condizionale dovrebbe essere inserita subito dopo la divisione per controllare e modificare il flusso del programma qualora necessario.

Istruzioni	Significato
mul rd, rs1, rs2	rs1 x rs2 con calcolo dei primi 32 bit
mulh rd, rs1, rs2	rs1 x rs2 (con segno, con segno) con calcolo degli ultimi 32 bit
mulhsu rd, rs1, rs2	rs1 x rs2 (con segno, senza segno) con calcolo degli ltimi 32 bit
mulhu rd, rs1, rs2	rs1 x rs2 (senza segno, senza segno) con calcolo degli ultimi 32 bit

Tabella 2: Estensioni per l'ISA Risc-V.

È comunque importante ricordare che un open-ISA, come può essere Risc-V, non è un open-hardware, bensì un prerequisito per quest'ultimo.

2.2 Core Basati su Risc-V

Di seguito descriviamo brevemente il funzionamento delle piattaforme open-hardware che implementano Risc-V. Vista la sua importanza all'interno di questo report, e vista anche la notevole documentazione associata, daremo maggior rilievo a PULP. Tale piattaforma costituisce la linea guida della ricerca contemporanea nell'ambito, ciononostante non trascureremo di esporre in modo esaustivo anche gli altri progetti sopra menzionati. In generale è essenziale capire quali sono le funzionalità e gli svantaggi di ciascuna implementazione per capire cosa manca alle soluzioni esistenti e cosa è importante modificare in quelle future. Nel seguito analizzeremo PicoRV32, Rocket Chip, ORCA, Potato, PULP, VexRiscv e SweRV.

PicoRV32. PicoRV32 è un core Risc-V a 32 bit scritto in Verilog e adatto per l'implementazione di FPGA. Supporta le seguenti configurazioni ISA: Integer Reduced (RV32E), Integer (RV32I), Integer Compressed (RV32IC) e Divisione e moltiplicazione di numeri interi (RV32IM). Il core esiste in tre varianti, con diverse interfacce per collegare le periferiche.

- *picoRV32* fornisce una interfaccia elementare da utilizzare in applicazioni semplici.
- *picoRV32 axi* presenta un'interfaccia avanzata (AXI), utile per applicazioni che utilizzano già AXI per connettere periferiche nect.
- *picoRV32 wb* fornisce un'interfaccia master Wishbone.

Ogni core può avere tre configurazioni: piccola, normale e grande. L'architettura più piccola non implementa istruzioni basate su contatori (le counter instructions) e controlli sulle istruzioni errate, mentre la configurazione normale implementa le funzionalità mancanti nella configurazione di base. Infine la configurazione grande supporta moltiplicazioni e divisioni, istruzioni compresse, shift e altre istruzioni personalizzate. Nonostante tutte le informazioni tecniche disponibili per quanto riguarda le sue caratteristiche e le relative prestazioni, non è disponibile una descrizione della sua architettura interna. È difficile capire come è stato costruito il processore analizzando semplicemente il codice Verilog, poiché è assente una documentazione adeguata.

Rocket Chip. Il Rocket Chip [5] è un generatore SoC open-source creato a Berkeley, presso l'Università della California, ed è essenzialmente una libreria di generatori che possono essere configurati e collegati in diversi modi, originando diverse configurazioni di un SoC. Il linguaggio di costruzione hardware che viene utilizzato è Chisel, il quale viene incorporato in Scala per generare core, cache, e interconnessioni per produrre un SoC completo, con un approccio orientato agli oggetti. Chisel può generare simulatori di C++ molto veloci , RTL (Verilog Register Transfer Level) di basso livello per l'emulazione di FPGA o ASIC utilizzando strumenti standard [6]. RocketChip può generare due tipi di core, rispettivamente Rocket e BOOM, che possono essere collegati ai coprocessori utilizzando un'interfaccia denominata RoCC [5].

- Rocket può essere pensato come una "libreria di componenti per un processore" perché diversi moduli possono essere facilmente riutilizzati in altri progetti, ed è configurato come [5] un core a 5 stadi, con supporto per gli ISA RV32I e RV64I, inoltre può essere modificato per includere le estensioni M, A, F e D. Questo core è altamente parametrizzato, offrendo la possibilità di configurare diversi moduli, come il numero di stage della pipeline in virgola mobile, estensioni opzionali, cache e dimensioni TLB. Con l'interfaccia RoCC, Rocket è in grado di controllare coprocessori e acceleratori esterni (unità crittografiche, unità di elaborazione delle immagini). Viene fornito anche il supporto per le implementazioni di FPGA, ma viene utilizzato principalmente per test e simulazioni, raggiungendo frequenze nella gamma di 25-100 MHz, a seconda della configurazione e della FPGA utilizzata per

l'implementazione. Attualmente, Rocket è utilizzato da SiFive, una società creata da alcuni esponenti di Risc-V, la quale produce processori Risc-V in silicio basati su Rocket, e offre strumenti per sviluppare, testare e valutare i loro progetti.

- BOOM [7] è un core di tipo out-of-order[1] che implementa l'ISA RV64G. È stato costruito per utilizzare l'infrastruttura definita da Rocket, e pertanto diversi moduli creati per Rocket vengono utilizzati anche da BOOM, inclusi le cache, le unità funzionali e le TLB. Per istanziare un core BOOM in un progetto generato dal SoC Rocket Chip, il core Rocket è sostituito dal core BOOM. Concettualmente, BOOM è un processore a 10 stadi, ma alcuni di essi sono combinati. BOOM è principalmente ottimizzato per ASIC, ma è utilizzabile anche su FPGA. Sfortunatamente, Rocket Chip non è la soluzione migliore per chi ha bisogno di implementare e modificare un core Risc-V su un FPGA, infatti da un lato sono presenti moltissime funzionalità, ma dall'altro manca una documentazione su come creare un progetto e come modificare il processore, il che rende difficile capire e modificare il codice. Inoltre il Verilog generato non è direttamente adatto per l'implementazione di FPGA, e le implementazioni FPGA esistenti sono state create solo per testare prodotti commerciali che lavoravano a frequenze molto basse.

ORCA. ORCA [8] è un core Risc-V scritto in VHDL e sviluppato da VectorBlox per essere il processore per una piattaforma proprietaria, sebbene possa essere utilizzato anche come processore autonomo. Il core implementa l'ISA RV32IM. ORCA è stato pensato per FPGA e per essere altamente parametrizzato: la pipeline può essere configurata per avere 4 o 5 stadi, migliorando la velocità o l'area, a seconda di ciò che è più importante per ogni implementazione. Oltre al numero di fasi della pipeline c'è anche la possibilità di implementare l'estensione M con supporto hardware. ORCA offre il vantaggio di essere pensato per garantire il supporto per più fornitori, infatti le interfacce di memoria sono progettate per supportare più fornitori di FPGA. La connessione alla memoria cache è fatta tramite AXI, mentre gli altri accessi (uncached) possono essere effettuati tramite AXI4-Lite o utilizzando un'interfaccia ausiliaria (a seconda del fornitore di FPGA), che può essere configurata come Wishbone, Intel Avalon o Xilinx LMB. In termini di area e velocità, l'implementazione RV32I richiede circa 1620 LUT e può raggiungere una frequenza massima di 109 MHz in un FPGA Altera Cyclone IV. Per lo stesso FPGA, la versione del processore con supporto hardware per l'estensione della moltiplicazione e della divisione utilizza circa 2350 LUT e la frequenza massima aumenta a 125 MHz. Nonostante tutto, manca una documentazione associata, così come mancano esempi di implementazione per diversi FPGA.

[1] Il termine tipo out-of-order indica la capacità di alcuni processori di eseguire le singole istruzioni senza rispettare necessariamente l'ordine imposto dal programmatore. Il processore analizza il codice che dovrà eseguire, individua le istruzioni che non sono vincolate da altre istruzioni e le esegue in parallelo. Questa strategia permette di migliorare le prestazioni dei moderni microprocessori dato che permette di riempire unità funzionali del processore che altrimenti rimarrebbero inutilizzate.

Potato. Potato è un core Risc-V scritto in VHDL, con una pipeline a 5 stadi, con un'implementazione completa di RV32I e il supporto per le istruzioni CSR (Control and Status Register). Per consentire il collegamento di periferiche e memoria viene utilizzata l'interfaccia BUS Wishbone. È disponibile un SoC che usa Potato, il quale è pensato per essere implementato in una FPGA Arty con frequenza di 50 MHz, con supporto per un timer a 32 bit con interrupt di confronto, un modulo di tipo UART (Universal Asynchronous Receiver/Transmitter) con velocità di trasmissione configurabile, un modulo di tipo GPIO (General-Purpose Input / Output), memorie che utilizzano BRAM (Block Random-Access Memories) e un file cache di istruzioni. Se confrontato con altre alternative, questo core segue un approccio interessante, utilizzando una interfaccia flessibile (il bus Wishbone) per collegare diverse periferiche e memorie, rendendo il processo di aggiungere nuove periferiche facile. Sfortunatamente, Potato non è documentato e non è fornita nessuna implementazione documentata per le varie FPGA, ovvero non ci sono informazioni specifiche per quanto riguarda l'utilizzo delle risorse.

SweRV. La Western Digital ha deciso di recente di sviluppare un processore Risc-V denominato SweRV, pensato per lavorare e interagire con alcuni dei loro prodotti, i quali normalmente utilizzerebbero una soluzione a pagamento e closed-source. SweRV è un core a 32 bit scritto in SystemVerilog, con supporto per RV32I e le estensioni M, C, Zifencei e Zicsr. Oltre a ciò SweRV ha 4 interfacce per il recupero delle istruzioni, gli accessi ai dati, gli accessi di debug e l'accesso diretto alla memoria esterna (DMA) tramite AXI4 o AHB-Lite. La documentazione è completa, concentrandosi su diversi aspetti, come gli accessi alla memoria (indirizzi, protezione della memoria, gestione delle eccezioni), gestione dell'alimentazione (stati di alimentazione, controllo dell'alimentazione), monitoraggio delle prestazioni (contatori, eventi) e cache. Lo SweRV può raggiungere una frequenza di 1 GHz per nodi a 28 nm, ma sfortunatamente non è adatto per le implementazioni FPGA.

VexRiscv. È un core a 32 bit descritto con SpinalHDL, un linguaggio di descrizione hardware open-source. Questo processore supporta il set di istruzioni RV32IMCA ed è ottimizzato per FPGA di diversi fornitori. Sono supportate diverse configurazioni, che portano a diversi valori consumo e frequenza: la configurazione più piccola (senza alcun modulo opzionale) richiede 496 LUT e raggiunge una frequenza di 324 MHz su un FPGA Artix 7, mentre il core completo, chiamato *VexRiscv full max*, con supporto per moltiplicazione e divisione, modulo di debug, 16 KB di dati e cache di istruzioni ed eccezioni, fa uso di 1758 LUT raggiunge 193 MHz sulla stessa FPGA. Il linguaggio di descrizione dell'hardware utilizzato per costruire questo core non è ben conosciuto dalla comunità, il che crea barriere per comprendere e modificare il codice. Inoltre, l'architettura del core non è descritta nella documentazione disponibile.

PULP. PULP è una piattaforma hardware nata nel 2013 come un progetto di collaborazione fra l'Università di Bologna l'ETH Zürich; questa collega e permette il corretto funzionamento dei seguenti componenti:

- Processore, o core. Inizialmente i primi chip usavano OpenRisc Core, tuttavia a partire da metà 2016 si è preferito utilizzare Risc-V, vista la sua maggiore flessibilità e la grande comunità che anima attivamente questo progetto. Il gruppo di ricerca su PULP ha sviluppato diversi core Risc-V, ottimizzandoli a seconda delle necessità:

  - RI5CY: in origine pensato per garantire una considerevole efficienza energetica nelle applicazioni di processazione del segnale (nel seguito, DSP), è un core a 32 bit con una pipeline a 4 stadi che implementa RV32I e le estensioni M e C, ovvero le estensioni per moltiplicazione e divisione intera e per le operazioni compresse. Volendo questo core è in grado di supportare le estensioni per le operazioni in virgola mobile. In aggiunta, RI5CY implementa molte istruzioni personalizzate, dai loop dell'hardware alle operazioni ALU, alle istruzioni SIMD.
  - Zero Riscy: questo processore è stato sviluppato poichè in genere non tutti i core di un processore vengono utilizzati per le applicazioni DSP, e pertanto l'idea alla base è quella di avere un core piccolo, semplice ed efficiente. Di conseguenza gli obiettivi alla base dello sviluppo di questo core sono l'efficienza e l'ottimizzazione degli spazi. Il core ha una architettura a 32 bit con una pipeline a 2 stadi, e può implementare sia RV32I sia RV32E, più le estensioni M e C.
  - Micro Riscy: per far fronte ad una richiesta di core sempre più piccoli è nato questo processore, il quale è una variazione del precedente con una superficie minore.
  - Ariane: a differenza dei core sopra elencati, che lavorano a 32 bit, questo è l'unico core a 64 bit.

  I processori Zero Riscy e Micro Riscy sono ora parte del progetto Lowrisc, sotto il nome *Ibex*, mentre RI5CY e Ariane entreranno a breve nella famiglia Core-V.

- Periferiche: PULP fa uso di JTAG, SPI, UART, I2S, GPIO e DMA. Quest'ultima ha un ruolo fondamentale in quanto permette ad altri sottosistemi (ad esempio le altre periferiche) di accedere direttamente alla memoria interna per scambiare dati, in lettura e/o scrittura, senza coinvolgere l'unità di controllo centrale.

- Moduli di interconnessione: PULP fa uso di tre di questi moduli: LIN (Logarithmic Interconnect), APB (Advanced Peripheral Bus) e AXI4 (Advanced eXtensible Interface).

- Co-processori (o acceleratori hardware): ovvero componenti progettati per migliorare le prestazioni in settori specifici (nel nostro caso ci concentreremo sugli acceleratori crittografici). Questo permette di scaricare la CPU da tutte quelle attività che altrimenti occuperebbero buona parte delle risorse computazionali e che può essere quindi lasciata libera di svolgere più agevolmente altri compiti.

- Memoria dati.

I componenti sopracitati costituiscono quindi i mattoni base di PULP. Vediamo ora in che modo questa piattaforma può essere utilizzata per realizzare alcuni chip. I microcontrollori più semplici basati su PULP sono PULPino e PULPissimo. Possono essere configurati per utilizzare qualsiasi core Risc-V a 32 bit (RI5CY, Zero-riscy, Micro-riscy). Le versioni avanzate consentono anche l'aggiunta di acceleratori al sistema.

PULPino supporta diverse periferiche attraverso AXI, come la UART, I2C (Integer - Integrated Circuit), SPI, timer a 32 bit, un boot ROM e molto altro. Oltre a questo la piattaforma offre una unità di debug che fornisce un accesso esterno alla sua memoria e alle sue periferiche tramite JTAG (Join Test Action Group). Il SoC è ben documentato e presenta un manuale dedicato, il quale descrive l'architettura, le interfacce fra i componenti, le periferiche esistenti e le opzioni di configurazione. Nonostante esista una versione di PULPino per FPGA, questo SoC è pensato principalmente per simulazioni RTL e ASICs, e di fatto l'implementazione FPGA può essere usata solo come piattaforma di emulazione, dal momento che le performance non sono ottimizzate e la frequenza massima raggiungibile non supera i 40 MHz. PULPissimo si differenzia da PULPino in un dettaglio: la presenza del modulo LIN tra i core e la memoria, il quale consente porte di accesso multiple. Queste vengono poi utilizzate da una periferica uDMA che è in grado di copiare i dati direttamente tra le periferiche e la memoria (eventualmente anche fra acceleratori), svincolando quindi questo lavoro dal processore. Uno schema rappresentativo di questi chip è presentato in Figura 1.

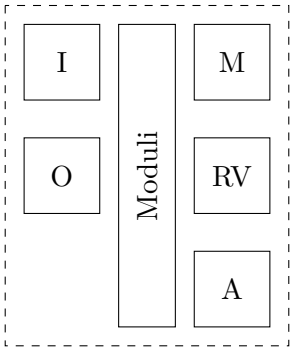

Figura 1: Schema di un microcontrollore basato su PULP, dove I,O indicano rispettivamente le periferiche di input e output, M le locazioni di memoria, RV i core Risc-V e A gli acceleratori.

I sistemi più avanzati si basano su cluster di core Risc-V a 32 bit con accesso diretto a una piccola memoria. Il cluster è supportato da un SoC che ospita una memoria di secondo livello più ampia, periferiche per input e output (nelle versioni successive è presente anche un microcontrollore completo di classe PULPissimo per la gestione dell'alimentazione e le operazioni di base). Architetture basate su questi sistemi sono

Mia Wallace, Honey Bunny, Fulmine e Mr. Wolf. Uno schema di questa architettura è presentato in Figura 2.

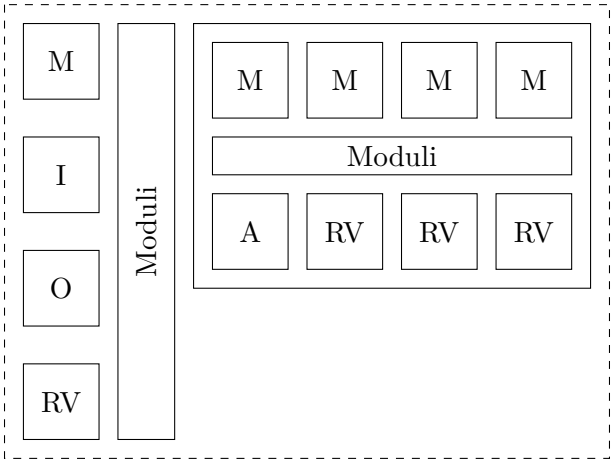

Figura 2: Schema di un sistema cluster-based, dove I,O indicano rispettivamente le periferiche di input e output, M le locazioni di memoria, RV i core Risc-V e A gli acceleratori.

È possibile utilizzare un sistema PULP che contiene più cluster e connetterlo ad un normale nodo di elaborazione. In questo scenario, il cluster PULP viene utilizzato come acceleratore efficiente dal punto di vista energetico per i carichi DSP. La piattaforma Hero è un esempio di tale sistema. Uno schema di questa architettura è presentato in Figura 3.

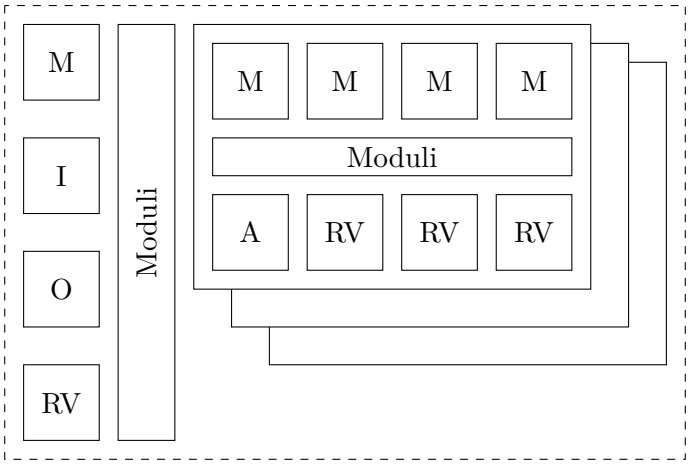

Figura 3: Schema di un sistema multi-cluster. Abbiamo indicato con I,O le periferiche di input e output, con M le locazioni di memoria, con RV i core Risc-V, e con A gli acceleratori.

Descriviamo infine alcuni dettagli riguardo a PULPino [9]. Abbiamo visto che questo SoC ha un'architettura a 32 bit e che è scritto in System Verilog. Inoltre PULPino supporta diverse periferiche attraverso AXI, come la UART, I2C (Integer-Integrated Circuit), SPI (Serial Peripheral Interface), timer a 32 bit, un boot ROM e molto altro. Oltre a questo la piattaforma offre una unità di debug che fornisce un accesso esterno alla sua memoria e alle sue periferiche tramite JTAG (Join Test Action Group).

Nella Tabella 3 sono riassunti in modo preciso e dettagliato tutti i SoC presentati in questa sezione, in modo che sia immediato riconoscerne l'architettura, i modulo di Risc-V che possono implementare, la pipeline del processore e i relativi stadi, il linguaggio di descrizione hardware utilizzato nella descrizione dei core, il supporto per FPGA, la frequenza massima, il consumo, le periferiche e la presenza o meno di una documentazione esaustiva a riguardo. Per ulteriori dettagli si faccia riferimento a [10].

	PicoRV32	Rocket Chip		ORCA	Potato	PULPino		VexRisc	SweRV
		Rocket	BOOM			RIS5Y	ZERO-RISCY		
Architettura	32 bit	32/64 bit	64 bit	32 bit	32 bit	32 bit	32 bit	32 bit	32 bit
Moduli Risc-V	I, E, M, C	I, M, A, F, D	I, M, A, F, D	I, M	I	I, M, C, F	I, E, M, C	I, M, C, A	I, M, C, Zifencei, Zicsr
Pipeline	-	In-order	Out-of-order	In-order	In-order	In-order	In-order	In-order	In-order
Stadi	-	5	10	4/5	5	4	2	2/3/4/5	9
Linguaggio di descrizione hardware	Verilog	Chisel		VHDL	VHDL	SystemVerilog		SpinalHDL	SystemVerilog
Supporto FPGA	Sì	Sì per simulazioni		Sì	Sì	Sì per simulazioni		Sì	No
Freq. max	714 MHz	100 MHz		125 MHz	50 MHz	40 MHz		piccola: 324 MHz grande: 193 MHz	1 GHz per nodi a 28 nm
Consumo	piccola: 761 LUTs normale: 917 LUTs grande: 2019 LUTs	-	-	2350 LUTs	-	-	-	piccola: 496 LUTs grande: 1758 LUTs	-
Periferiche	AXI, UART, SPI	AXI		AXI, Avalon, Wishbone	UART, GPIO	UART, I2C, SPI, timers, JTAG debug		AXI, Avalon, Wishbone	AXI, JTAG
Documentazione	Insufficiente	Insufficiente	Insufficiente	Insufficiente	Ottima	Insufficiente	Insufficiente	Insufficiente	Ottima

Tabella 3: Elenco e comparazione dei core basati su Risc-V descritti nel presente report.

2.3 Core non Basati su Risc-V

Attualmente sono disponibili anche alcuni processori open-source non basati su Risc-V, molti dei quali hanno ciò che manca negli attuali Risc-Vcore, ovvero una documentazione adeguata, un supporto FPGA con velocità operative elevate, un consumo moderato delle risorse e un design che può essere facilmente modificato e riadattato. Esponiamo nel seguito MB-Lite e OpenRISC.

MB-Lite. Il softcore MB-Lite [11] implementa RISC MicroBlaze ISA a 32 bit di Xilinx, tuttavia non tutte le istruzioni di MicroBlaze sono state implementate per semplificare l'architettura: alcune istruzioni che possono essere configurate dai parametri del compilatore, come il moltiplicatore hardware e lo shifter, possono essere aggiunti al processore modificando i parametri di configurazione, secondo i requisiti dell'applicazione. In alternativa queste unità possono essere sostituite da librerie software. I risultati di implementazione riportati per una scheda FPGA Xilinx Virtex 5 hanno mostrato che una versione di base del processore MB-Lite senza moltiplicatore e shifter è in grado di raggiungere una frequenza di 222 MHz, che richiede 843 LUT e 355 FF. Quando il moltiplicatore e lo shifter sono abilitati, la frequenza si riduce a 65 MHz e le risorse aumentano a 1450 LUT. L'architettura MB-Lite è basata sui processori MIPS (Microprocessor without Interlocked Pipeline Stages), caratterizzati da una pipeline a 5 stadi, inoltre ogni fase è stata attentamente sviluppata e descritta in componenti separati, seguendo le stesse metodologie di descrizione hardware e denominazione dei segnali. Per concludere: MB-Lite ha dimensioni ridotte, ma allo stesso tempo può raggiungere frequenze simili a quelle dei suoi concorrenti; il modo in cui il codice è organizzato e commentato, la modularità e le interfacce offerte, insieme alla documentazione fornita e alle buone pratiche seguite durante il suo sviluppo contribuiscono a rendere questo processore un'eccellente opzione da utilizzare nei progetti di ricerca.

OpenRISC. OpenRISC è un progetto di hardware libero di un microprocessore RISC sviluppato da OpenCores e rilasciato sotto LGPL (GNU Lesser General Public License). Il processore è descritto con il linguaggio Verilog ed è sintetizzabile su ASIC o su FPGA. L'instruction set, vale a dire l'insieme delle istruzioni che il microprocessore è capace di eseguire, è molto simile a quello delle architetture MIPS, pertanto include le operazioni matematiche di base (addizione, sottrazione, moltiplicazione, divisione), le operazioni logiche bit-a-bit (AND, OR, XOR, SHIFT), i controlli condizionali, le comparazioni (minore di, ...) e molto altro, così come descritto in [12]. Oltre a ciò è presente un vasto insieme di istruzioni di tipo SIMD (Single Instruction stream, Multiple Data stream) pensate per l'elaborazione di segnali digitali. Questa architettura è stata implementata su vasta scala in diverse piattaforme, tuttavia negli ultimi 5 anni la tendenza è stata quella di trascurare OpenRISC a vantaggio di Risc-V, maggiormente supportato da parte della comunità e notevolmente più flessibile del primo.

3 Conclusioni

Nel present report è stata presentata un'analisi sull'architettura di dispositivi hardware open-source, distinguendo due modelli architetturali per un processore: RISC e CISC. È stato studiato in dettaglio l'ISA Risc-V, con particolare riferimento alla versione RV32I (l'unica ratificata), e alle operazioni matematiche che è possibile effettuare con esso. Fra le 40 operazioni descritte nell'ISA RV32I sono state individuate 21 istruzioni per gestire il calcolo con i numeri interi ed è emerso che è possibile effettuare operazioni per gestire l'aritmetica intera, ovvero addizione, sottrazione e shift dei registri, le operazioni logiche, ovvero le operazioni bit-a-bit, e le comparazioni di tipo aritmetico.

Sono stati elencati quindi i vari SoC open-source presenti attualmente sul mercato e basati su Risc-V, e per ognuno di questi, ovunque la documentazione associata lo ha permesso, sono stati riportati i dati tecnici associati, con particolare enfasi alle estensioni di Risc-V supportate, sulla frequenza massima del processore, sulla struttura della pipeline associata, e sui consumi (vedi Tabella 3). Oltre a ciò, nell'ottica più ampia di descrivere lo stato dell'arte a riguardo dei microprocessori open-source, sono stati descritti anche MB-Lite e OpenRISC, due core non basati su Risc-V.

Nel Report 2, verranno individuate le primitive crittografiche fondamentali per l'implementazione di una blockchain e la fattibilità di implementare le stesse all'interno dei processori open-hardware illustrati nel presente report.

4 Riferimenti bibliografici

[1] Tejas KARKHANIS and José E MOREIRA. Ibm power architecture. 2011.

[2] Andrew WATERMAN and others. The risc-v instruction set manual, volume i: Base user-level isa. *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62, 116*, 2011.

[3] Moritz NÖLTNER-AUGUSTIN. Risc-varchitecture and interfaces the rocketchip. *Computer Engineering, 6.*, 2016.

[4] Brian ZIMMER et al. A 0.11 pj/op, 0.32-128 tops, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16nm. *2019 Symposium on VLSI Circuits. IEEE. p. C300-C301.*, 2019.

[5] Krste ASANOVIC et al. The rocket chip generator. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, 2016.

[6] Jonathan BACHRACH et al. Chisel: constructing hardware in a scala embedded language. *DAC Design Automation Conference 2012*, 2012.

[7] Christopher CELIO, David PATTERSON, and Krste ASANOVIC. The berkeley out-of-order machine (boom) design specification. *University of California, Berkeley*, 2016.

[8] D. HILL et al. Orca: A new architecture for high-performance fpgas. *International Workshop on Field Programmable Logic and Applications. Springer, Berlin, Heidelberg, p. 52-60*, 1992.

[9] Andreas TRABER et al. Pulpino: A small single-core risc-v soc. *3rd RISCV Workshop*, 2016.

[10] Licheng WANG et al. Cryptographic primitives in blockchains. *Journal of Network and Computer Applications, 127: 43-58*, 2019.

[11] Tamar KRANENBURG and Rene VAN LEUKEN. Mb-lite: A robust, light-weight soft-core implementation of the microblaze architecture. *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). IEEE. p. 997-1000.*, 2010.

[12] Charles PRICE. Mips iv instruction set. 1995.